EpicReads

Module Overview:

This Project follows the **AWS Compute Services** module of the AWS Mastery course. It is designed to provide you with the foundational knowledge and skills required to start using Amazon Compute Services. This module covers a range of introductory topics, including:

- **Compute:** Learning how computers operate and complete tasks within the cloud setup.
- **Compute Services:** Discovering different cloud services that enable computing for various needs.
- **Virtual Machine:** An overview of virtual machines, Simulated computers that operate within a real computer system, performing tasks similarly to actual computers.
- **Elastic Compute Cloud (EC2):** Details about a service called EC2 that offers flexible and resizable computing power on the internet.
- **Component of Amazon EC2:** Different parts or elements that make up the Amazon EC2 service, each contributing to its functionalities and configurations.
- **EC2 Instance Naming Convention:** Guidelines or rules used to name and identify specific instances created within the EC2 service.
- **Instance Types:** Different categories or variations of virtual machines available on AWS, each with its own specific configurations and capabilities to suit diverse computing requirements and workloads.
- **Launch Your First EC2 Instance:** Step-by-step guidance on starting your initial virtual server within the AWS cloud environment.
- **SSH EC2 Instance and Install Httpd:** Guidance on securely accessing an EC2 instance using SSH and setting up an HTTP server (Httpd) on it.

- **Creating a Windows EC2 Instance:** The process of setting up a virtual Windows-based computer within the AWS cloud environment.
- **Elastic Container Service (ECS):** An explanation of AWS ECS service, that is used for efficiently managing and deploying containers, simplifying application development and scaling in the cloud environment.
- **AWS Elastic Beanstalk:** A look into AWS Elastic Beanstalk, a platform that streamlines the deployment and management of applications in the cloud by automating the deployment process, making it easier to handle application-related tasks without the need for manual configuration.
- **Serverless Computing:** Understanding of Serverless Computing, where cloud providers manage the infrastructure and resources, allowing developers to focus solely on writing and deploying code without the need to manage servers or infrastructure configurations.
- **AWS Lambda:** A look into AWS Lambda serverless computing service that enables users to run code without provisioning or managing servers.

Project Objectives:

After completing this project, you should be able to do the following:

- Launch and configure a Linux-based EC2 instance precisely customized to fulfill the operational needs of EpicReads, an organization operating in the cloud environment.
- To familiarize yourself with Virtual Machines, Elastic Compute Cloud (EC2) services, Elastic Container Service (ECS), and explore the fundamentals of serverless computing for a comprehensive understanding.

DURATION

This project requires approximately 90 minutes to complete.

ICON KEY

Throughout this Project, we use various icons to highlight different types of instructions and notes. Here's what each icon signifies:

- 📓 **Note:** Offers a helpful hint, tip, or piece of crucial guidance.
- ℹ️ **Learn More:** Directs you to additional sources for more detailed information.
- ⚠️ **Caution:** Points out information that, while not critical, is important and missing it might necessitate redoing steps.
- 🔥 **WARNING:** Indicates an action that's permanent with potential consequences that could cause process failure or irreversible changes.
- ✅ **Expected Output:** Shows a sample of the expected result to help confirm correct execution of a command or file edit.
- 💻 **Command:** Denotes a specific command that you're required to execute.
- 💡 **Consider:** Encourages you to reflect on how to apply a concept in your scenario or to spark a discussion related to the topic at hand.

COMMON ERRORS

1. **Misconfiguration of Security Groups:** Not setting up appropriate inbound and outbound rules in security groups can lead to connectivity issues or security vulnerabilities.
2. **Improper Setup of SSH and HTTPd:** Incorrectly configuring SSH access or setting up an HTTP server (Httpd) on an EC2 instance can result in security gaps or functionality issues.

Refer to COMMON ERRORS - EXPLANATION SECTION for the solution.

Start Project: Welcome to Our AWS Compute Solutions Page!

My Greetings and a warm welcome to our Second AWS Project Solutions Page. Your journey into harnessing AWS compute power begins here, and I am happy to be a part of this exciting journey.

Let's start on this Compute journey together!

In this solution page, you will walk through the step-by-step process of the tasks:

1. Launch a Linux-based EC2 instance using Amazon Linux 2 AMI.
2. Configure the instance with the necessary settings for web server operations.
3. Establish a secure SSH connection and perform basic network tests.

Let’s start now!

Task 1: Setting Up an AWS Virtual Private Cloud (VPC) with Public and Private Subnets for EpicReads

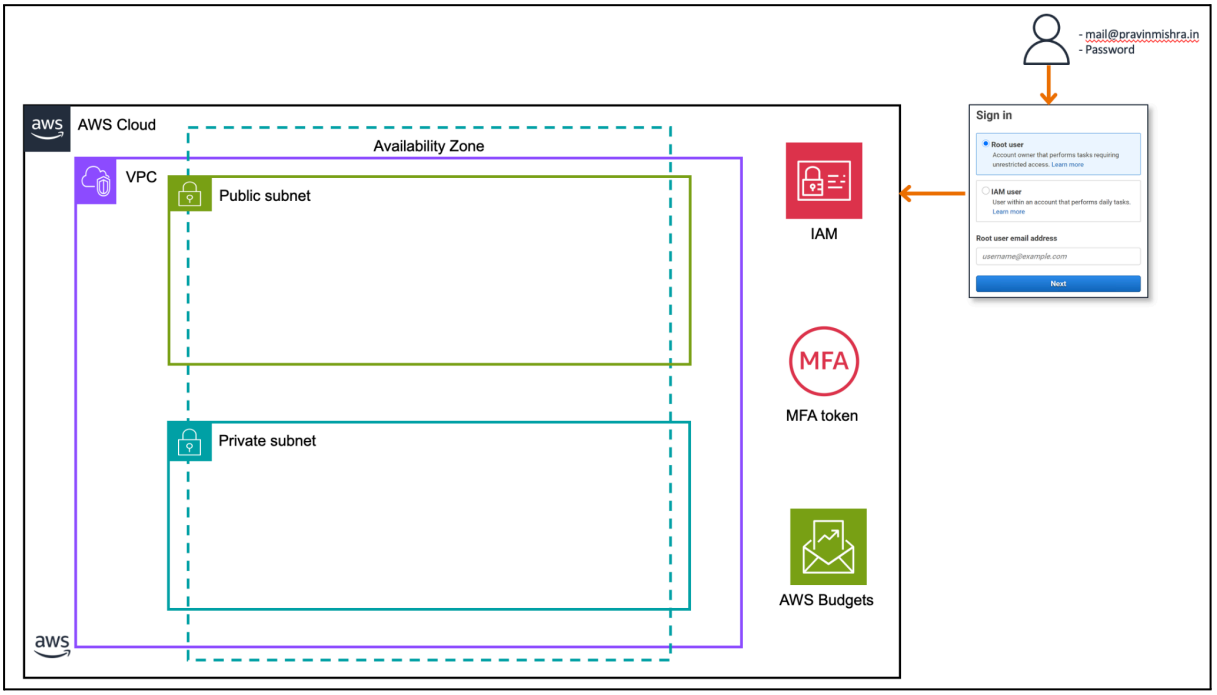


Figure 1.1.1: Virtual Private Cloud

Project Overview:

This project focuses on creating a VPC within the AWS environment for EpicReads, complete with public and private subnets. This infrastructure will provide a secure and isolated environment for hosting the company's applications and services.

Project Objectives:

- Establish a VPC that offers enhanced security and network isolation for EpicReads' AWS resources.
- Create public subnets for resources that need internet access and private subnets for resources that require restricted access.

- Ensure that the VPC design aligns with best practices for security and scalability.

STEP 1: Planning and Design:

Determine the network range (CIDR block) for the VPC and subnets.

1. **Understand CIDR Notation Basics:**
    - CIDR notation represents an IP address range using a combination of an IP address and a suffix (e.g., /16, /24) denoting the number of bits in the network prefix.
    - A smaller suffix represents a larger number of available IP addresses for hosts.
2. **List Required Subnets and Their Sizes:**
    - Plan the number of subnets you need and their respective sizes (number of IP addresses).
    - Separate these subnets into public and private based on their need for internet accessibility.
    - When deciding the IP address range for your AWS Virtual Private Cloud (VPC), you can use tools like a CIDR calculator available online or in Google to help set up the network. For instance, we chose a CIDR block of 10.0.0.0/16 for our VPC.
    - Within this VPC, the plan is to create subnets that are like smaller segments within the larger network. For example, we aim to make two public subnets (e.g., 10.0.0.0/24, 10.0.1.0/24) and two private subnets (e.g., 10.0.2.0/24, 10.0.3.0/24). (*Figure 1.1.2: CIDR block selection)*
    - The '10.0.0.0/16' CIDR block allows us a range of IP addresses for all our subnets. Each subnet gets its own portion of this range, defining the number of available IP addresses for devices or instances within that subnet.

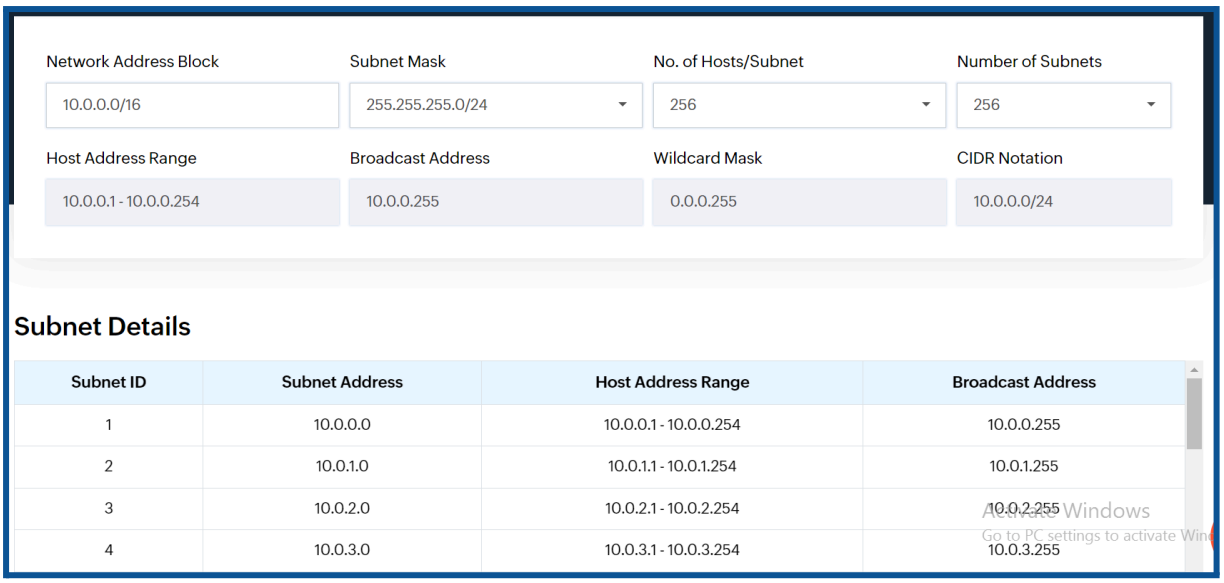


Figure 1.1.2: CIDR block selection

⚠️ **Caution:** Incorrect CIDR block selection or subnet planning may lead to IP address conflicts, network connectivity issues, or security vulnerabilities. Pay close attention to details to avoid complications.

STEP 2: Creating the VPC:

1. **Access the AWS Console:**
    - Log in to your AWS account and navigate to the AWS Management Console.
2. **Choose the Desired Region:**
    - At the topmost corner of the AWS Management Console, select the region where you want to create your VPC. For instance, if you're selecting the Asia Pacific region, choose the appropriate region from the dropdown menu (*Figure 1.1.3: Region Selection)*

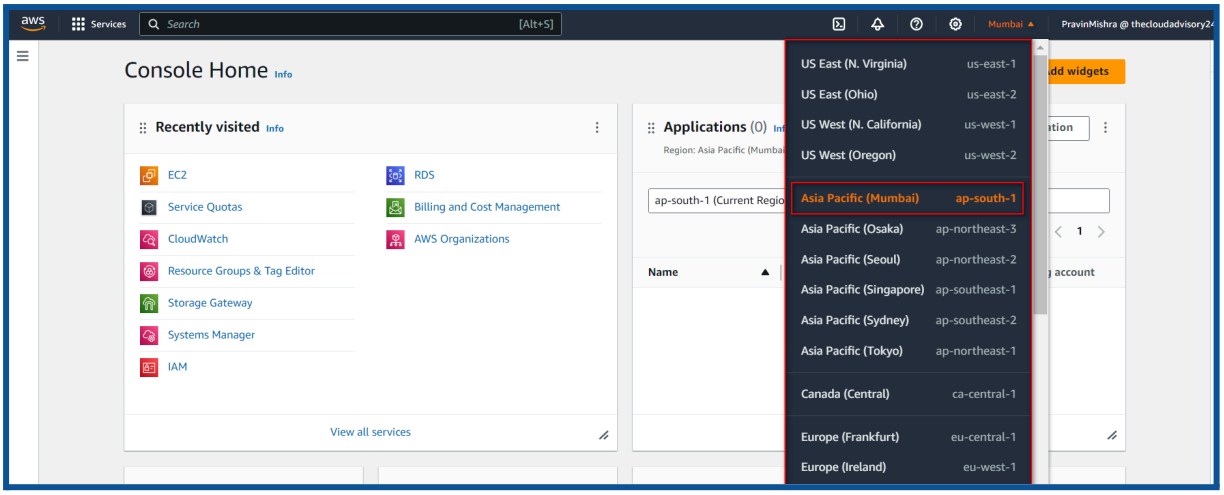


Figure 1.1.3: Region Selection

3. Go to the VPC Dashboard:

- In the AWS Management Console, locate the "Services" menu and select "VPC" under the "Networking & Content Delivery" section.

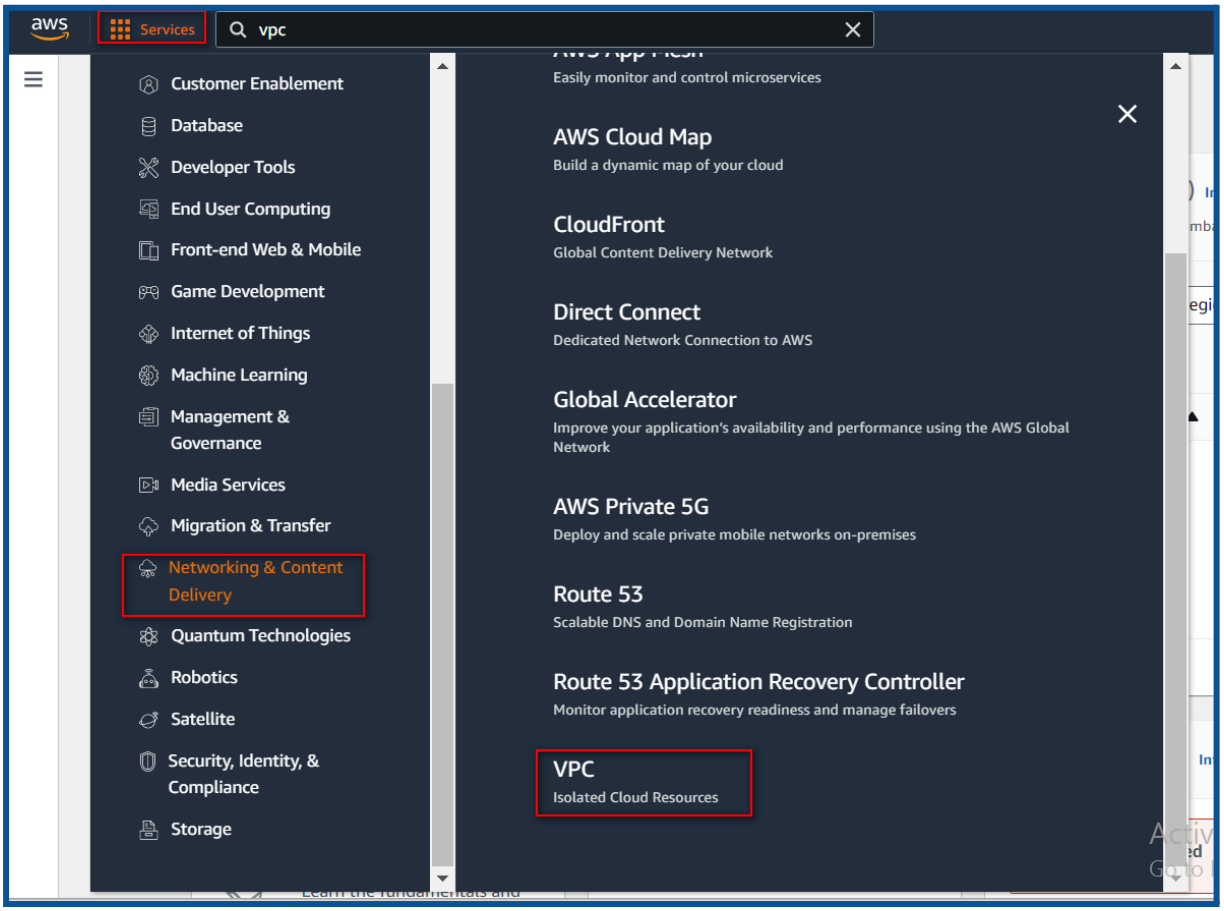


Figure 1.1.4: Select VPC

4. Create a New VPC:

- Click on "Create VPC" in the left sidebar of VPC Dashboard. (*Figure 1.1.5: Create VPC)*

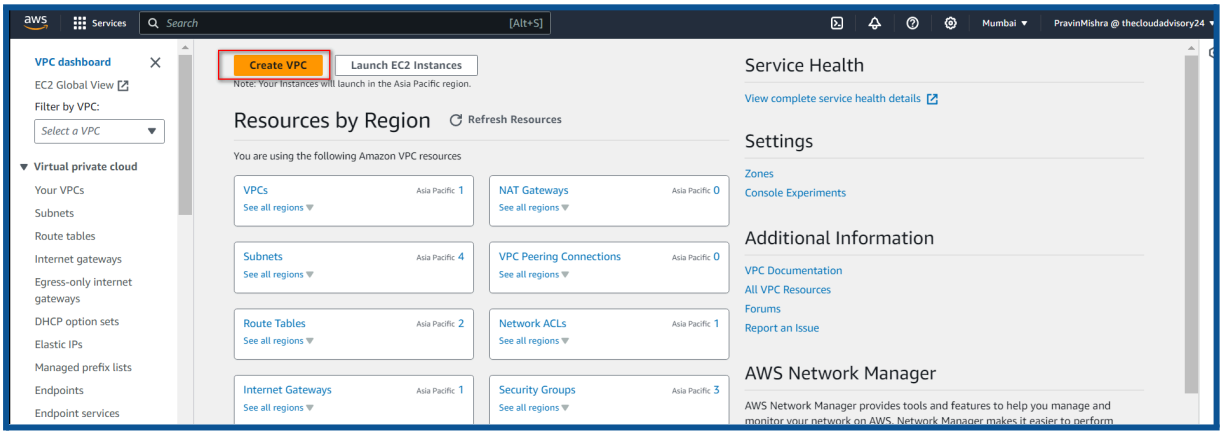


Figure 1.1.5: Create VPC

- In the resource to create option click on "**VPC only**" and provide the name of the VPC in the name tag.
- Enter the IPv4 CIDR block you previously selected in the earlier steps (for example, **10.0.0.0/16**) in the "**IPv4 CIDR block**" field. This defines the range of private IP addresses for your VPC's instances. (*Figure 1.1.6: VPC Settings)*

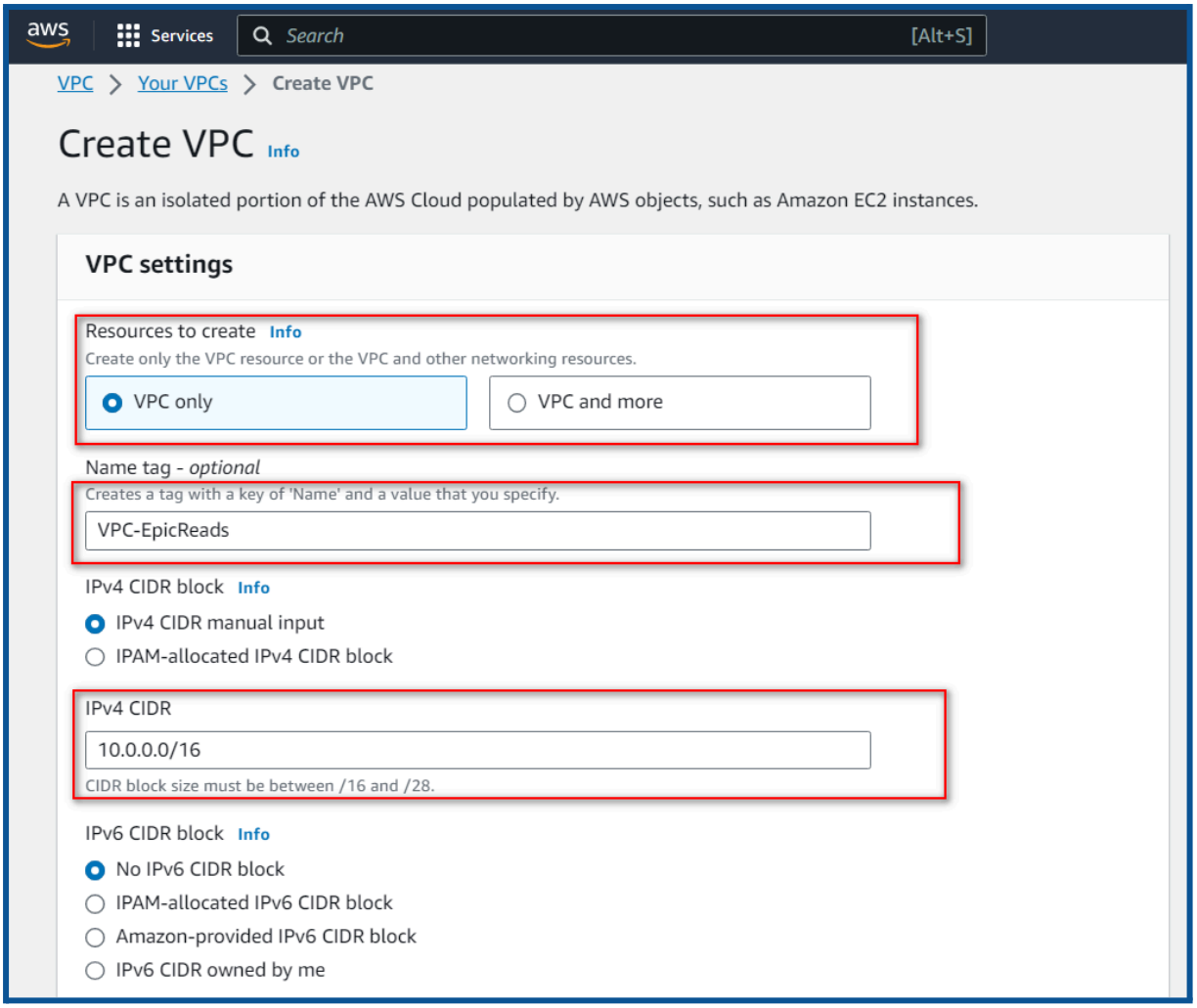


Figure 1.1.6: VPC Settings

- Select "**Default**" for the tenancy option. This indicates that your instances will be provisioned on shared hardware.
- Tags are optional but helpful for identification purposes. You can add tags to your VPC by clicking "Add tag" and providing a key-value pair. (*Figure 1.1.7: VPC Tenancy)*

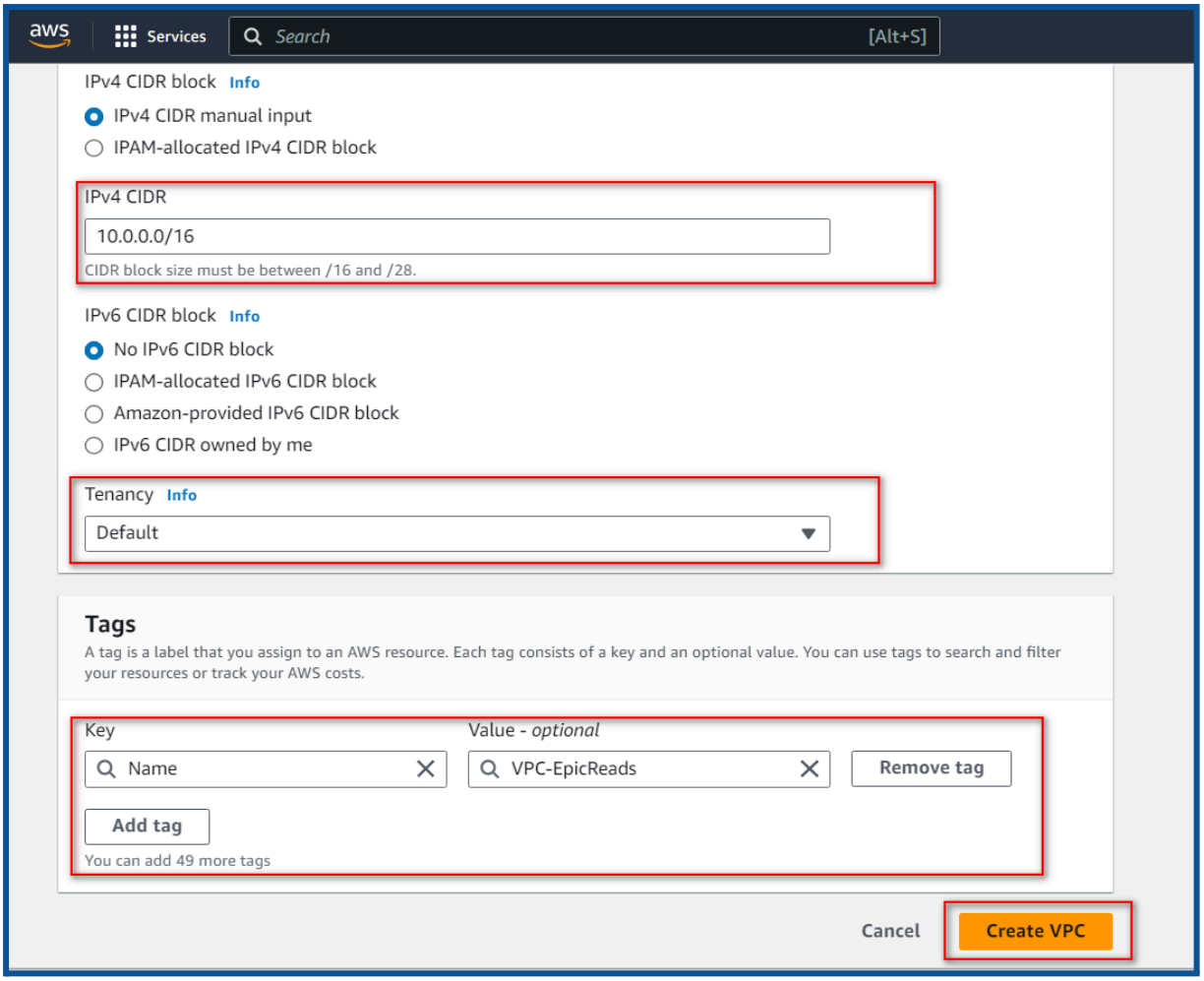


Figure 1.1.7: VPC Tenancy

- Review the details you've entered for the VPC.
- Once everything looks correct, click on "**Create VPC**" to initiate the creation process.
- VPC now creates and views the details in the VPC Dashboard. (*Figure 1.1.8: VPC Details)*

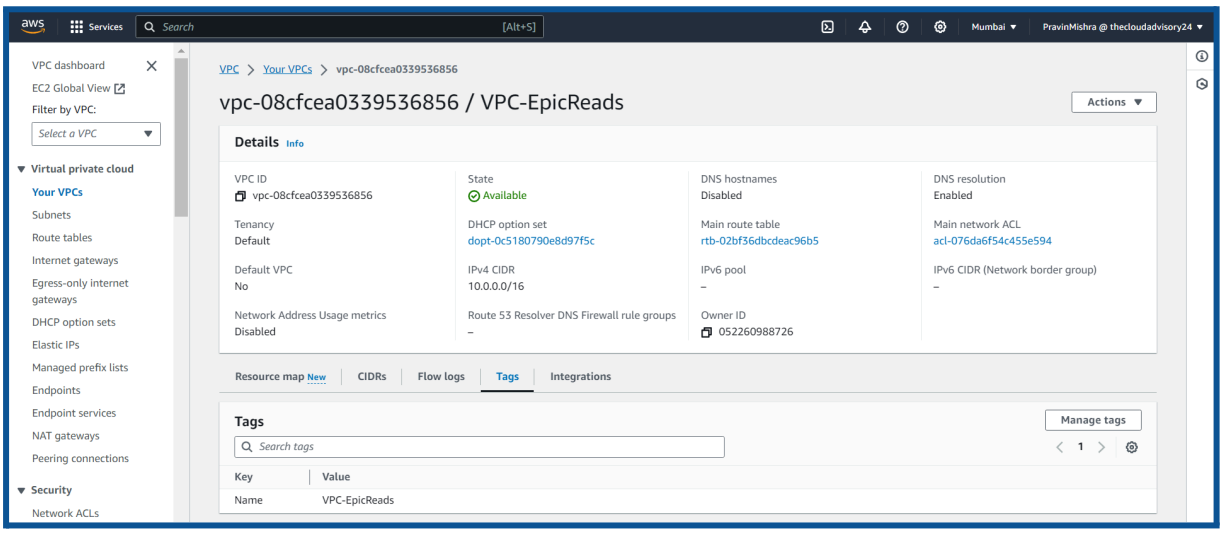


Figure 1.1.8: VPC Details

📓 **Note:** While creating your VPC and configuring subnets, always consider the specific requirements of your applications and services.

STEP 3: Setting Up Subnets:

1. Create Public Subnets:

- In the VPC Dashboard, navigate to "Subnets" on the left sidebar and click on "Create subnet." (*Figure 1.1.9: Create Subnet)*

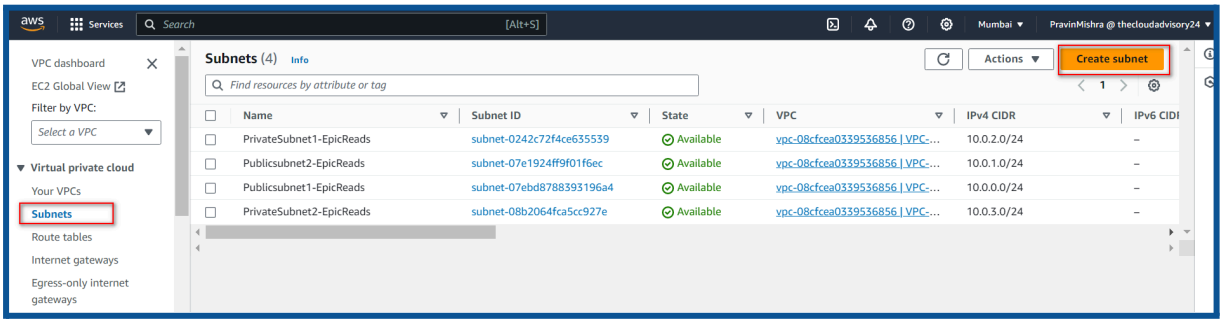


Figure 1.1.9: Create Subnet

- Choose the VPC you created earlier from the dropdown menu. (*Figure 1.1.10: Subnet settings)*

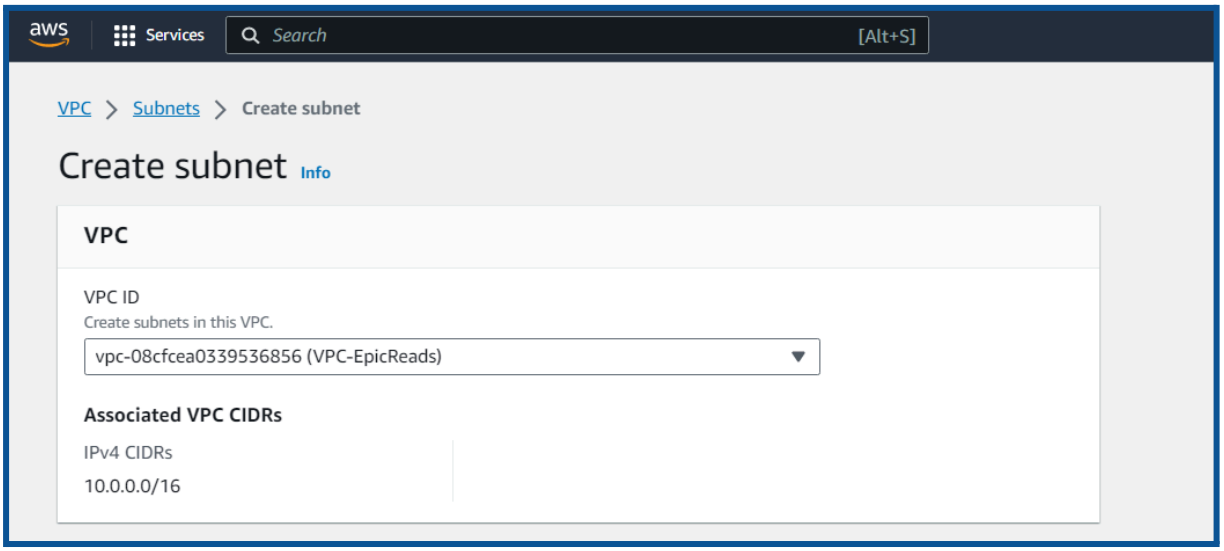
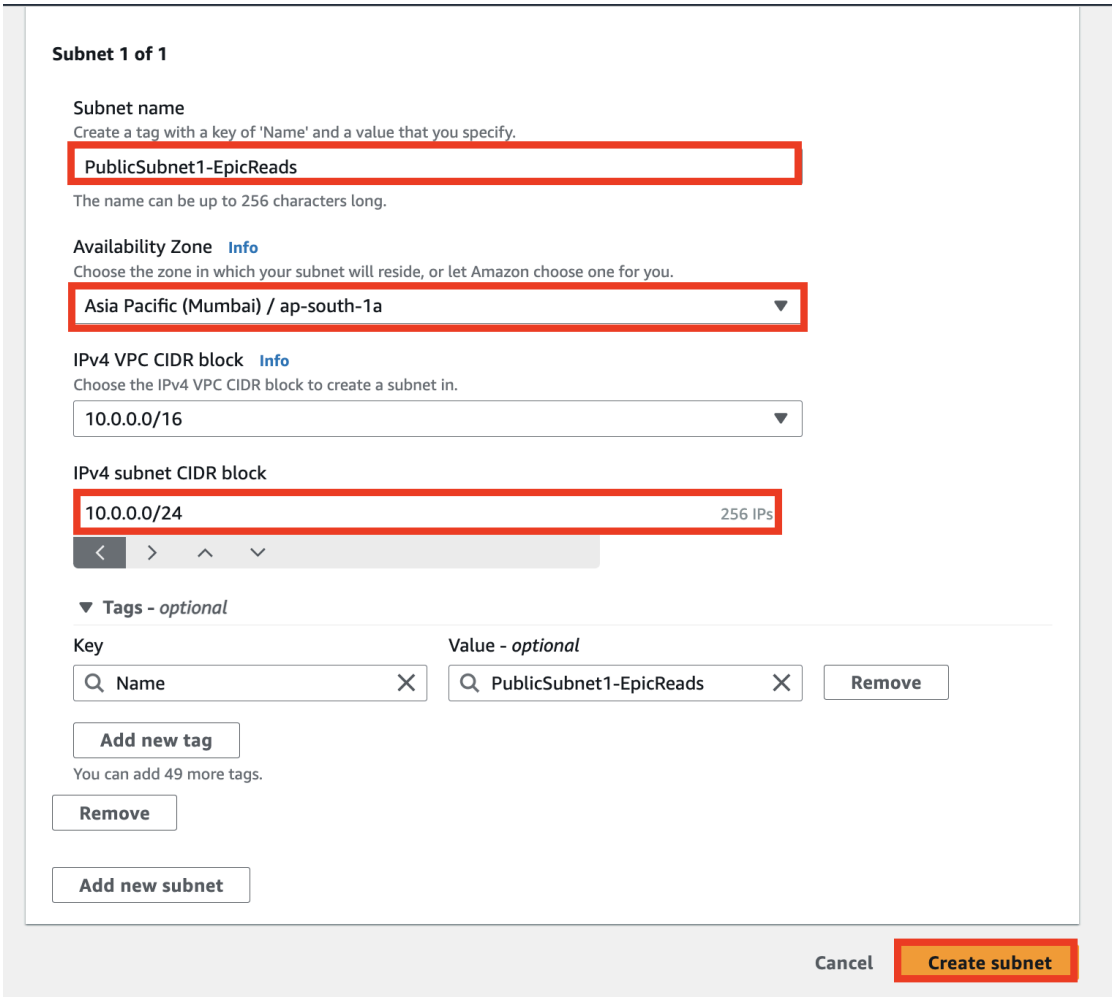


Figure 1.1.10: Subnet settings

- Provide a name for the subnet in the "Name tag" field to identify it easily.
- Assign an appropriate CIDR block for the public subnet (for example, 10.0.0.0/24).

- Ensure to select a different Availability Zone for each subnet if you plan to have multiple subnets for high availability. (*Figure 1.1.11: Subnet name)*



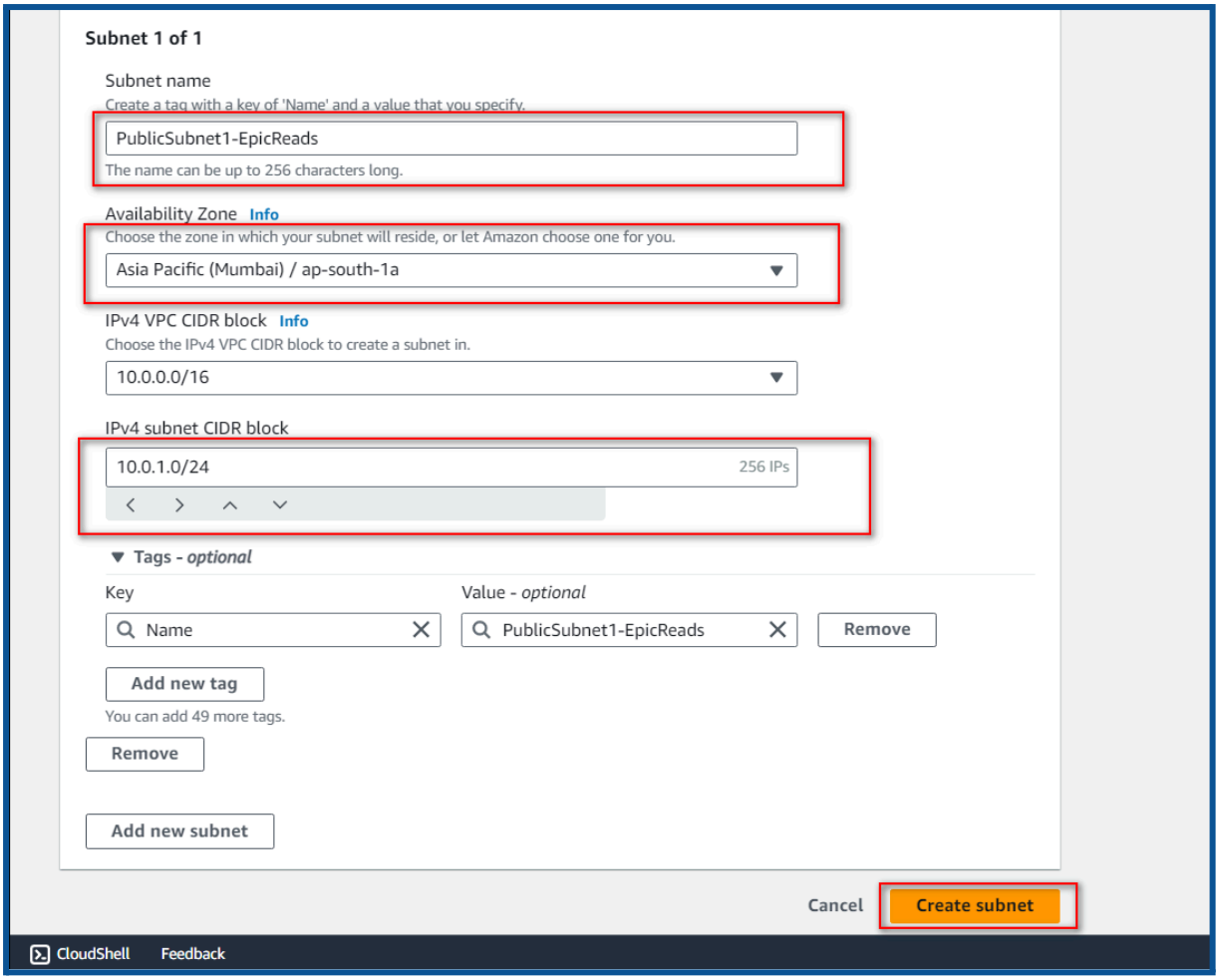


Figure 1.1.11: Subnet name

Repeat the same for *PublicSubnet2-EpicReads* with CIDR block *10.0.1.0/24*

2. **Create Private Subnets:**
    - Similarly, click on "Create subnet" again within the VPC Dashboard.
    - Provide a name for the private subnet in the "Name tag" field.
    - Choose the same VPC as before from the dropdown menu.
    - Assign a different CIDR block for the private subnet (for example, 10.0.1.0/24).
    - Remember to select different Availability Zones for the private subnets to ensure distribution across multiple zones. (*Figure 1.1.12: Subnet configuration)*

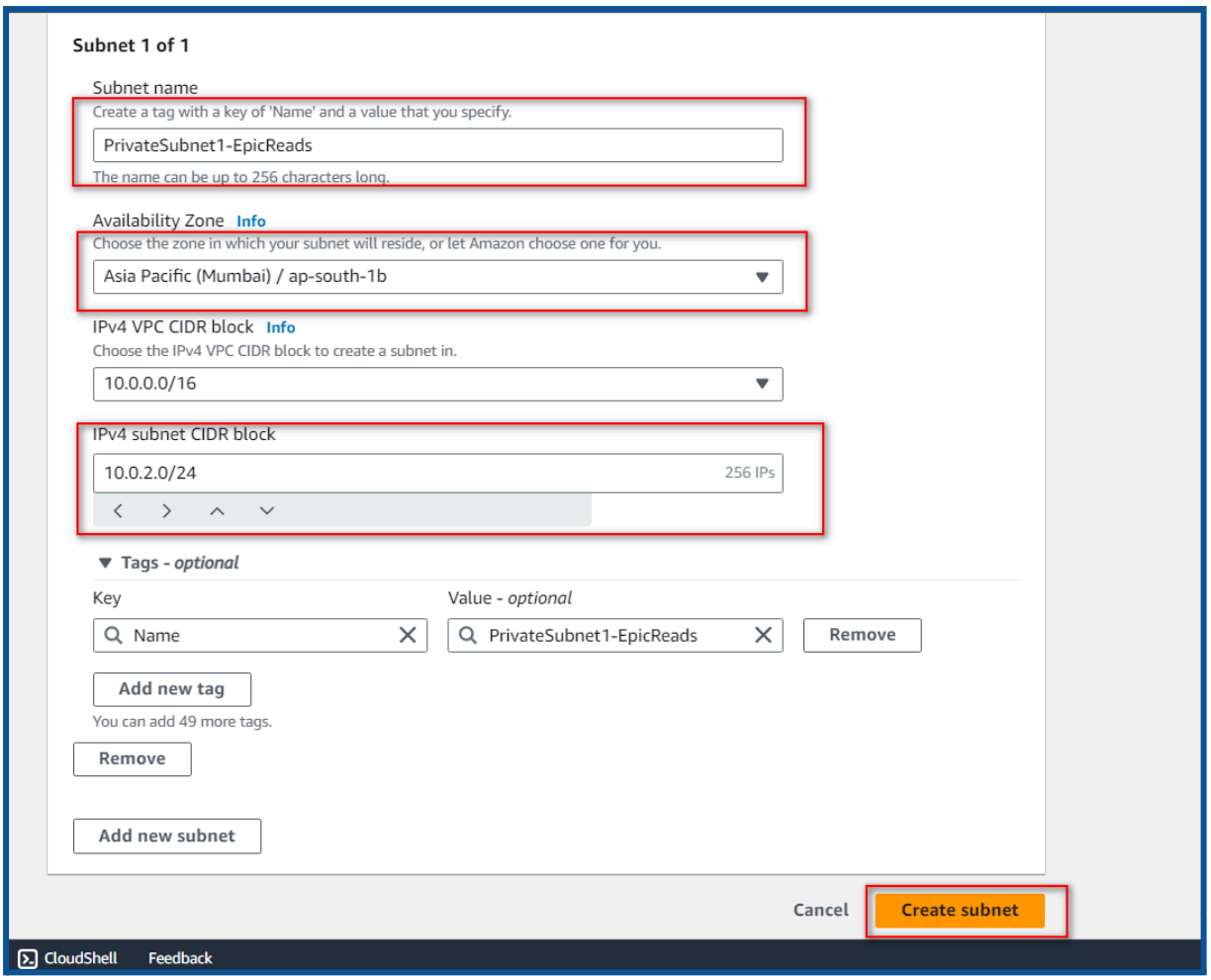


Figure 1.1.12: Subnet configuration

Repeat the same for *PrivateSubnet2-EpicReads* with *CIDR block 10.0.3.0/24*

4. **Review and Create:**
    - Double-check the details you've entered for the subnets.
    - Ensure that each subnet has its unique CIDR block and is associated with the appropriate VPC and Availability Zone.
    - Click on "Create subnet" to complete the creation of both public and private subnets.

Get back to the VPC console, Select the Subnets and you will find the Subnets that you have created a while ago! *(Figure 1.1.13: Subnets of EpicReads)*

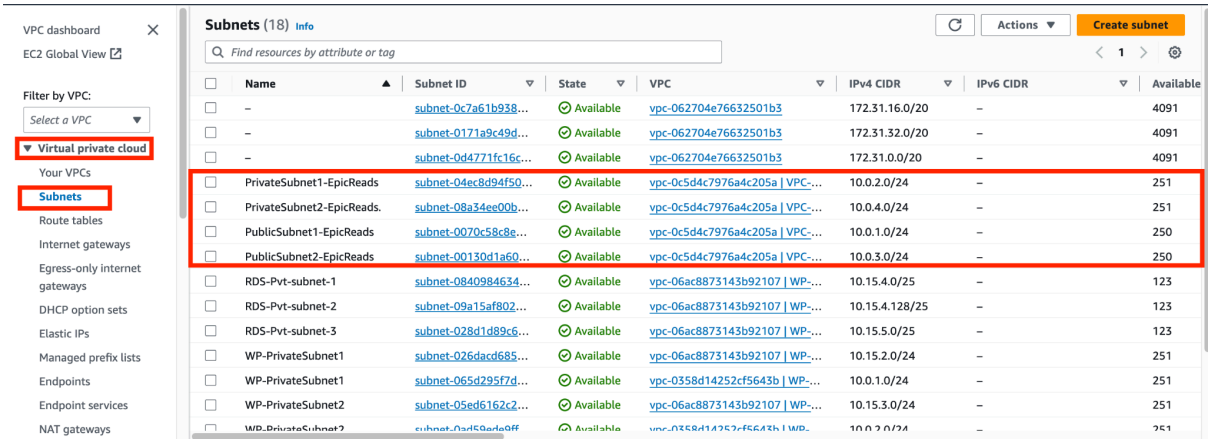


Figure 1.1.13: Subnets of EpicReads

💡 **Consider:** Reflect on the specific requirements of your applications and services when planning subnets. Consider future scalability and potential changes in network demands.

STEP 4: Configuring Internet Access:

1. Set Up Internet Gateway (IGW) for Public Subnets:

- In the VPC Dashboard, click on "Internet Gateways" in the left sidebar. (*Figure 1.1.14: Create Internet Gateway)*

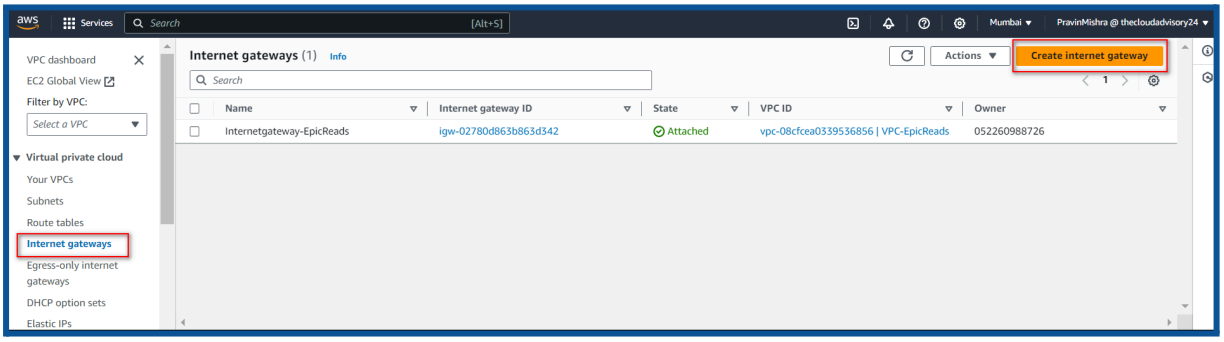


Figure 1.1.14: Create Internet Gateway

- Click on "Create internet gateway" and provide a name for the IGW. (*Figure 1.1.15: Internet Gateway Settings)*

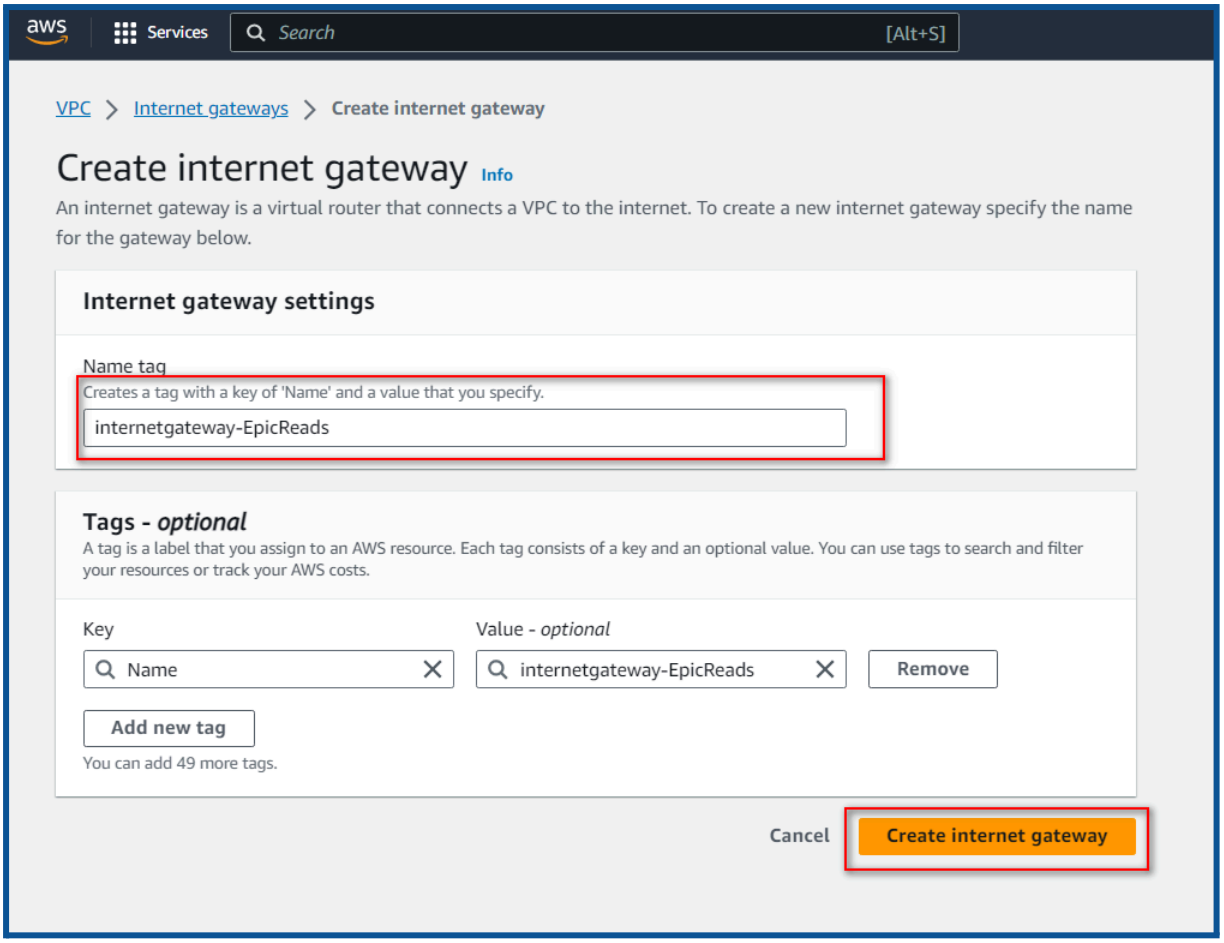


Figure 1.1.15: Internet Gateway Settings

- Select the newly created IGW and click on "Attach to VPC," then choose your VPC from the list. (*Figure 1.1.16: Attach to VPC)*

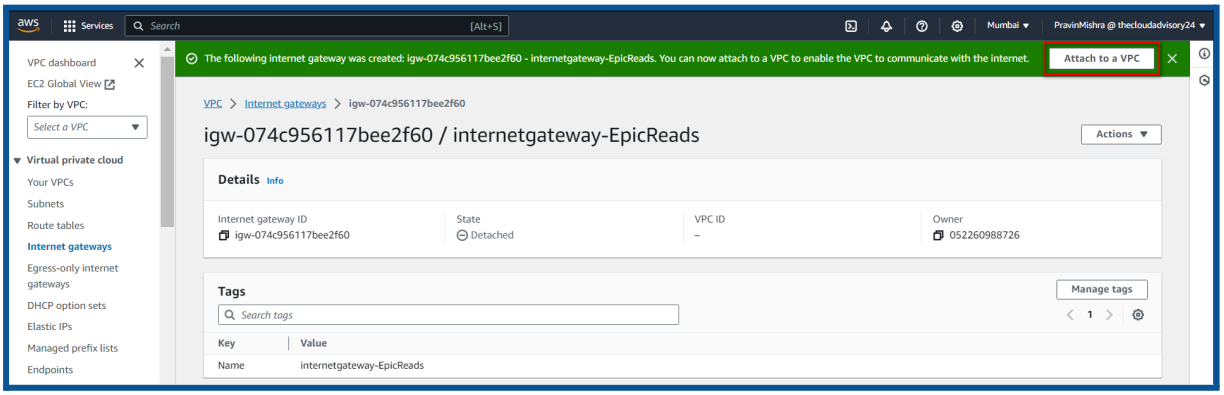


Figure 1.1.16: Attach to VPC

- Once VPC is attached, you can see the details in the internet gateways Tab. (*Figure 1.1.17: IGW Details)*

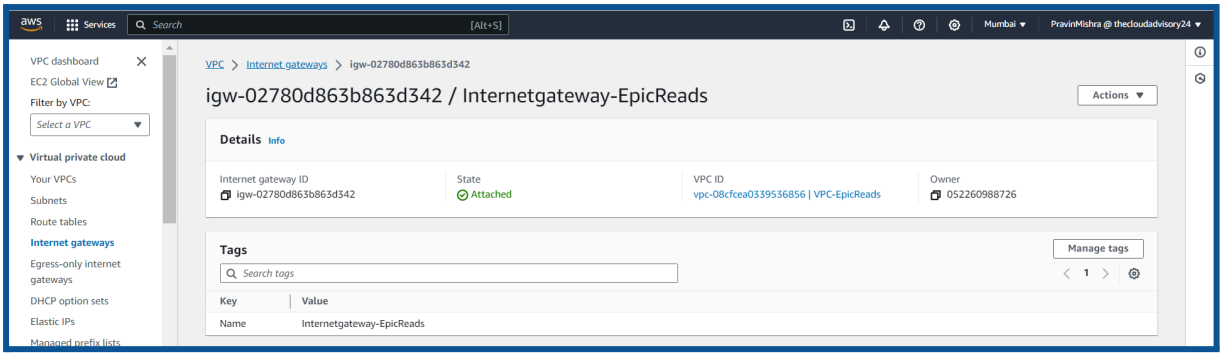


Figure 1.1.17: IGW Details

2. Update Route Tables for Public Subnets:

- Go back to the VPC Dashboard and click on "Route Tables" in the left sidebar. (*Figure 1.1.18: Route Tables)*

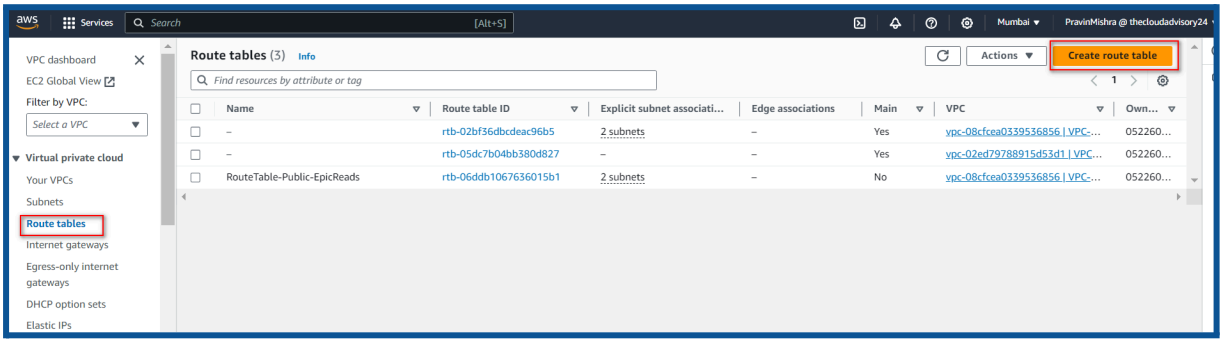


Figure 1.1.18: Route Tables

- Select the route table associated with your public subnets (usually the main route table for the VPC) (*Figure 1.1.19: Route Table Settings).*

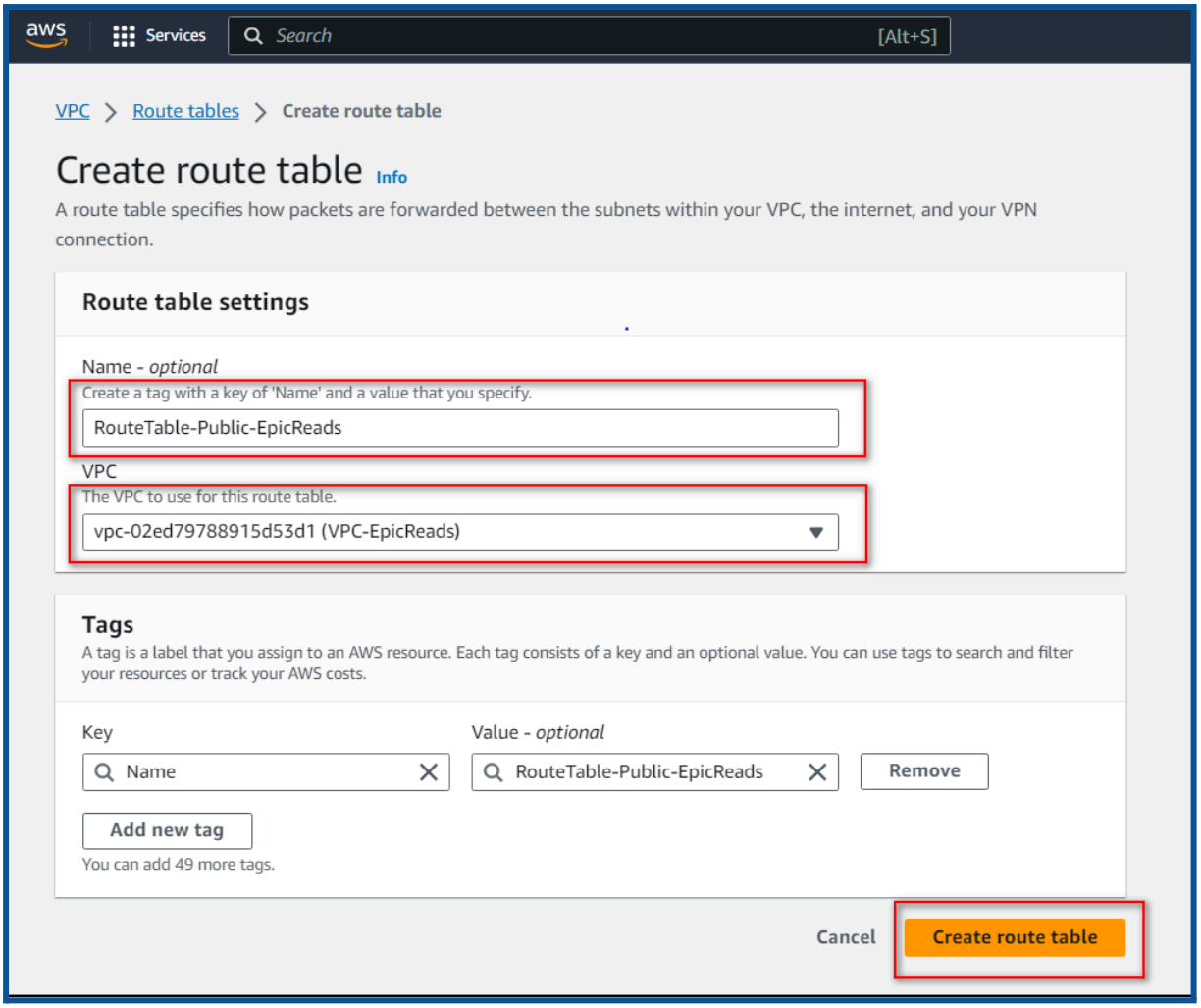


Figure 1.1.19: Route Table Settings

- The Route Table is created now.
- Click on the "**Routes**" tab and then on "**Edit routes**."
- Add a new route with destination **0.0.0.0/0** and target as the ID of the IGW you attached earlier. (*Figure 1.1.20: Edit Routes)*

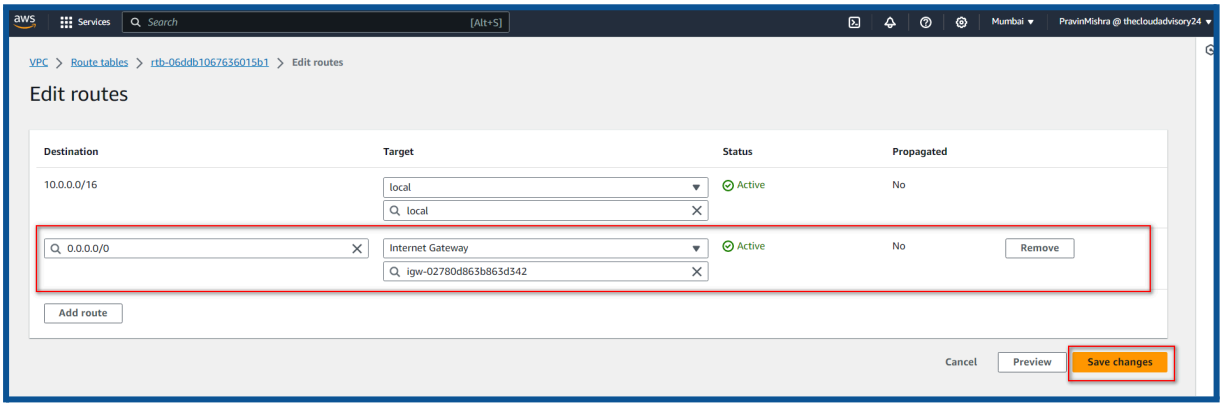


Figure 1.1.20: Edit Routes

- Save the changes.

3. Associate Public Subnets to the Routetable

- As the Public Subnets need the internet access do associate *PublicSubnet1-EpicReads* and *PublicSubnet2-EpicReads* to the created Route table *Routetable-Public-EpicReads*
- Select the Route table *Routetable-Public-EpicReads,* Choose *Subnet Associations* and Click on *Edit subnet association (Figure 1.1.21: Subnet Association)*

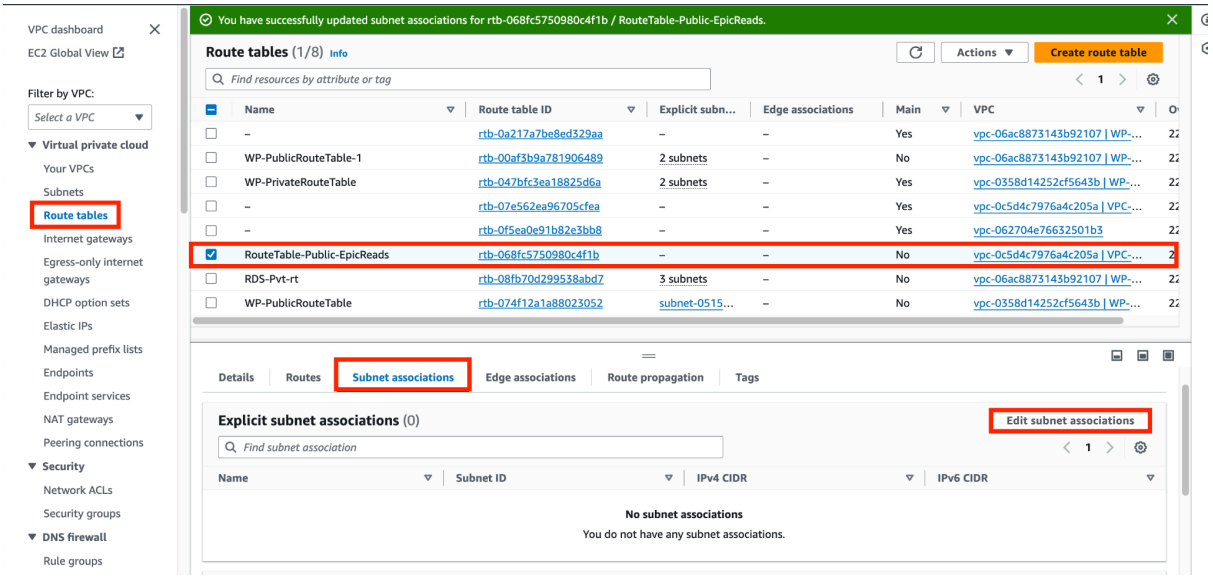


Figure 1.1.21: Subnet Association

Select both the *Public subnets* of EpicReads and click on the button *Save association (Figure 1.1.22:Choose Public Subnets)*

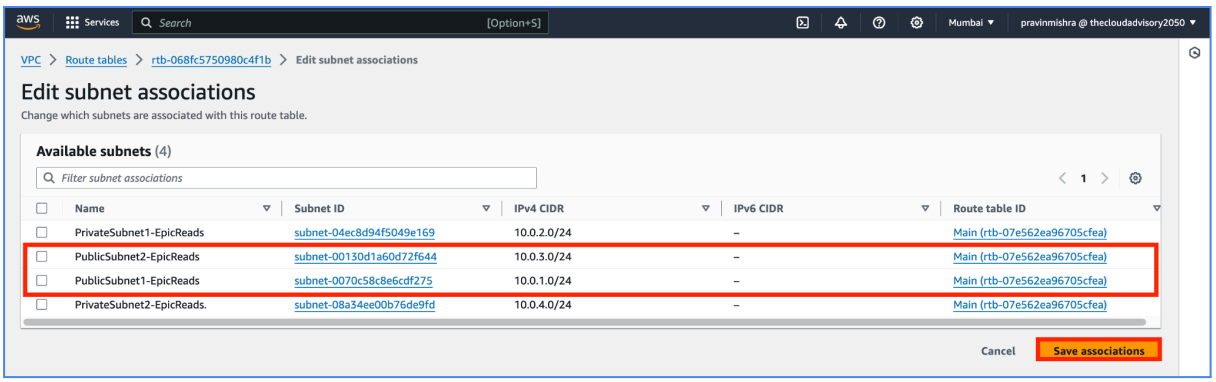


Figure 1.1.22:Choose Public Subnets

4. **Review and Confirm:**
    - Double-check that the IGW is attached to the VPC and the route table for public subnets has the necessary route to the IGW.
    - Confirm that the route table is associated with the public subnets you want to have internet access.

STEP 5: Implementing Network Security:

1. **Create Network Access Control Lists (NACLs):**
    - In the VPC Dashboard, click on "Network ACLs" in the left sidebar.
    - Click on "Create network ACL." (*Figure 1.1.23: Create Network ACL)*

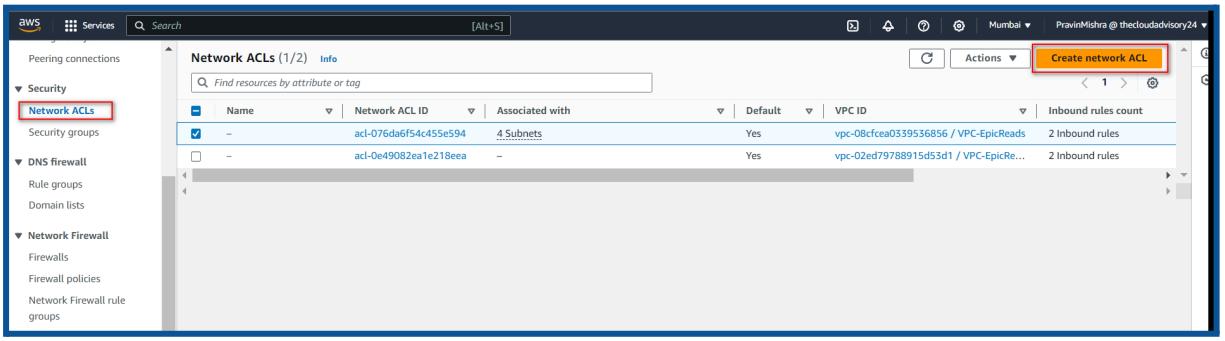


Figure 1.1.23: Create Network ACL

- Provide a name for the NACL and associate it with your VPC. (*Figure 1.1.24: Network ACL Settings)*

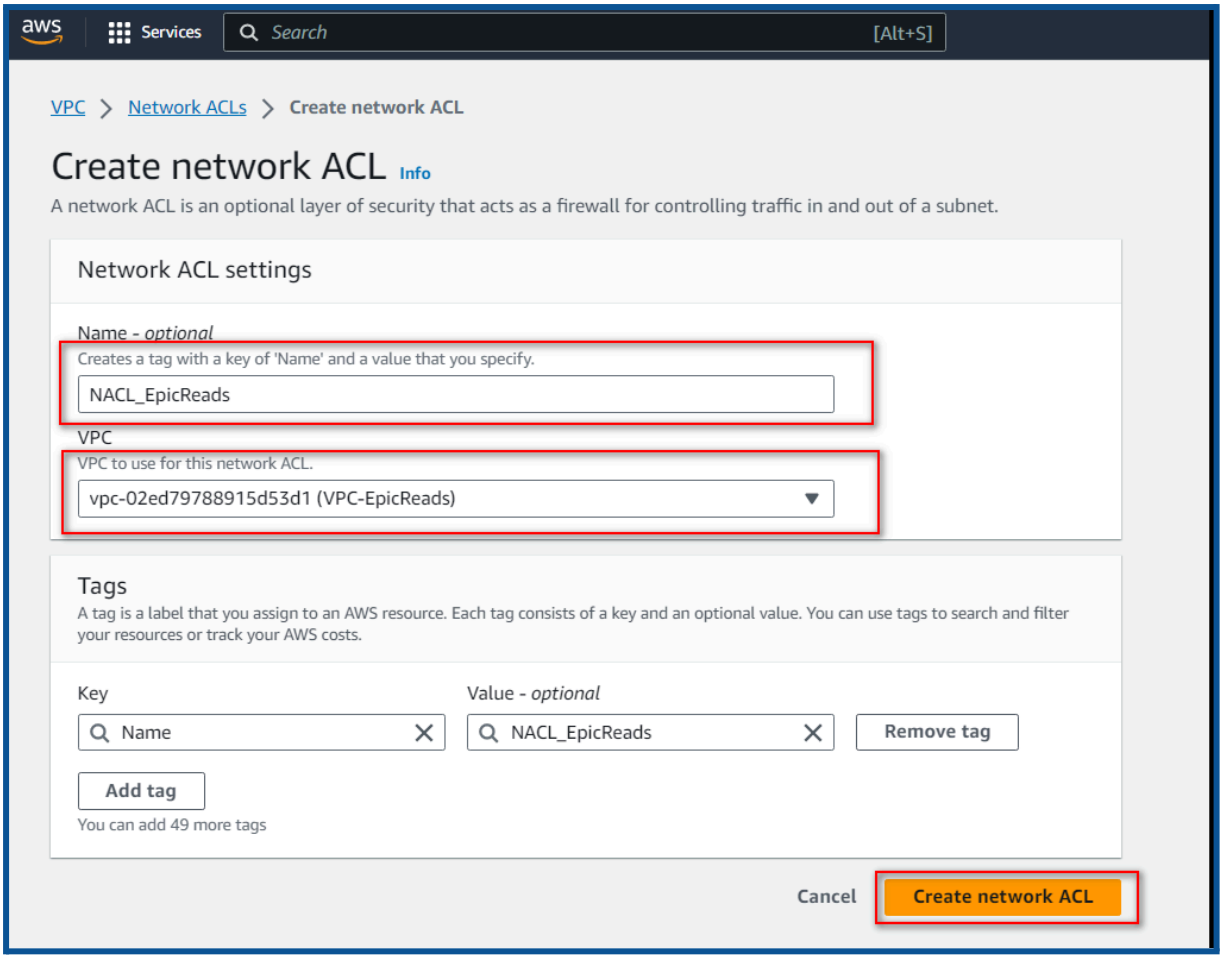


Figure 1.1.24: Network ACL Settings

- Configure inbound and outbound rules to control traffic.
- Remember that NACLs are stateless, so you'll need to define rules for both inbound and outbound traffic.(*Figure 1.1.25: Edit Inbound Rules)*

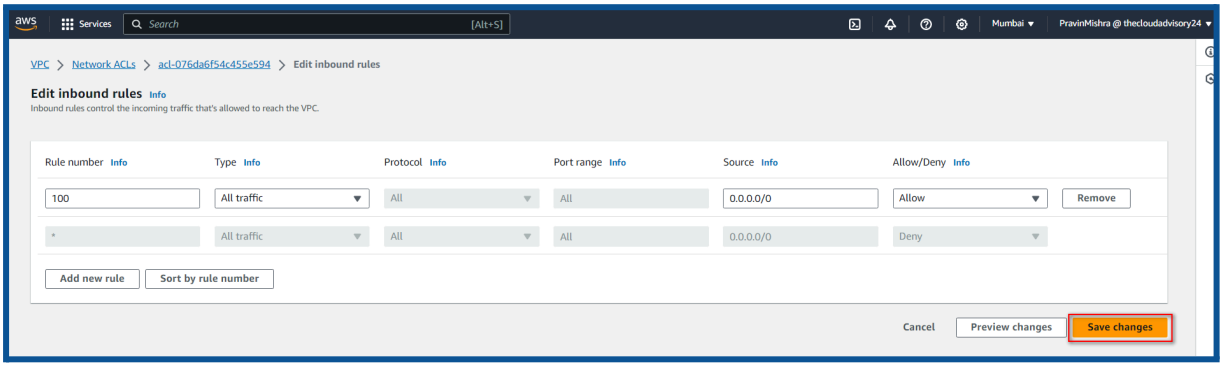


Figure 1.1.25: Edit Inbound Rules

2. Create Security Groups:

- Go back to the VPC Dashboard and click on "Security Groups" in the left sidebar.
- Click on "Create security group." (*Figure 1.1.26: Create security group)*

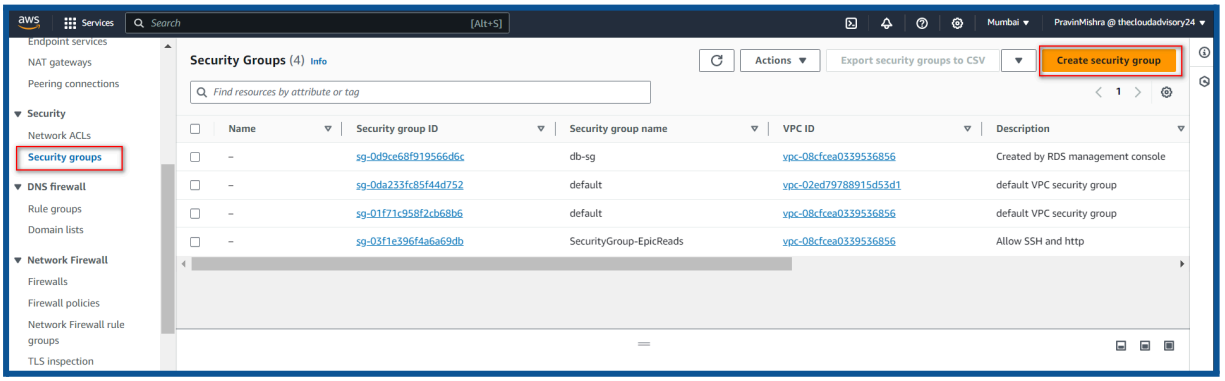


Figure 1.1.26: Create security group

- Create security group
- Provide a name and description for the security group. (*Figure 1.1.27: security group Details)*

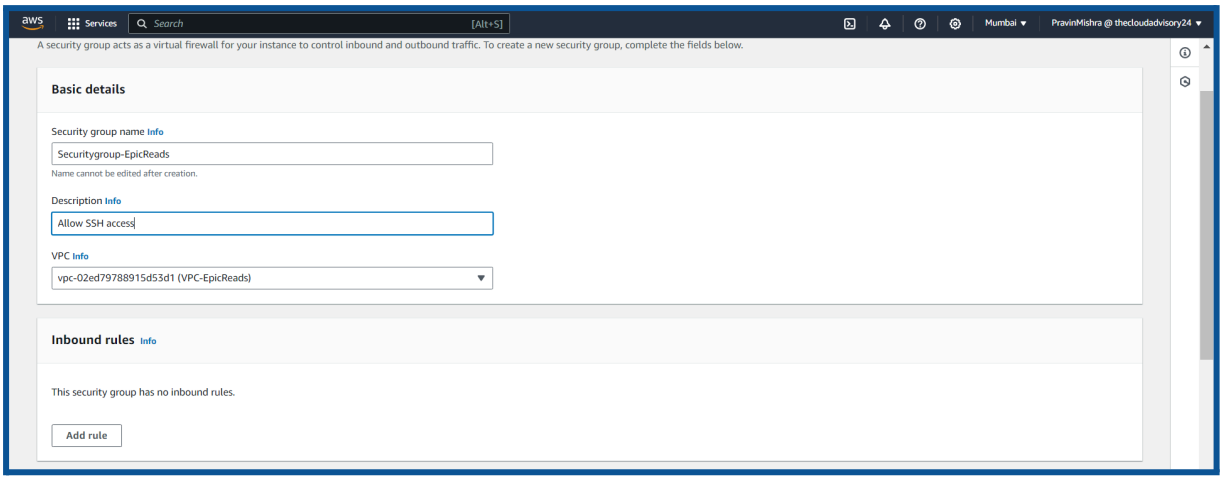


Figure 1.1.27: security group Details

- Define or Edit inbound and outbound rules for the security group, specifying the allowed traffic types and sources/destinations. (*Figure 1.1.28: security group inbound)*

(Here in the below inbound rules Mysql/ Aurora can be avoided and can be shown in the *Assignment 2: Enhancing Security with Security Groups for EpicReads*

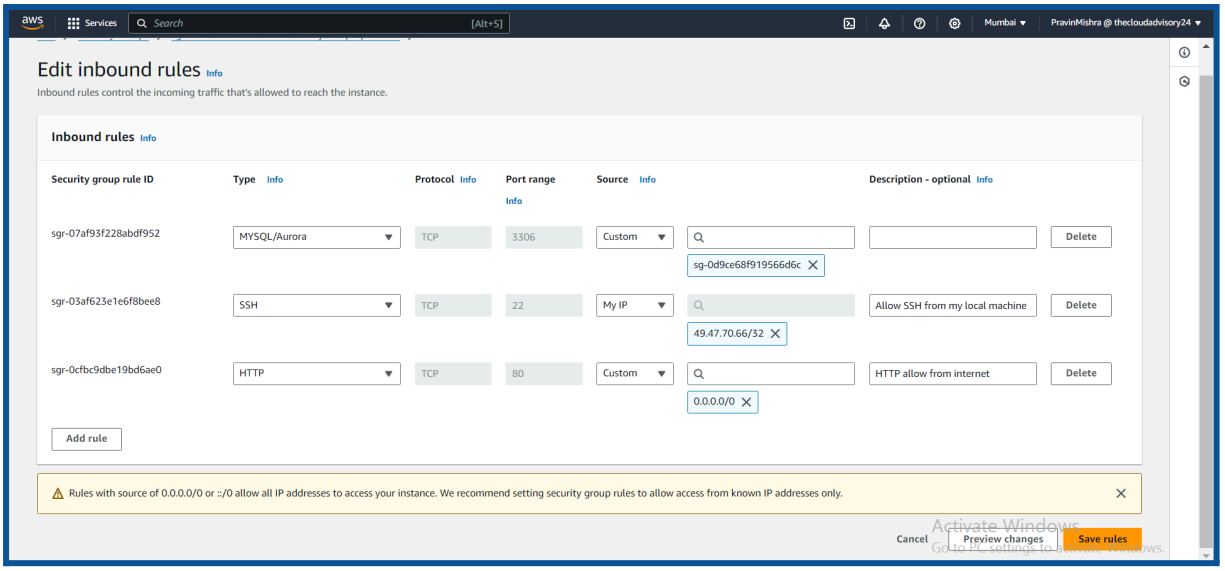


Figure 1.1.28: security group inbound

3. Apply NACLs and Security Groups to Subnets:

- Associate the NACLs and Security Groups with the appropriate subnets.
- In the VPC Dashboard, click on "Subnets" in the left sidebar.
- Select a subnet, go to the "Network ACL" tab, and associate the relevant NACL.
- Similarly, select a subnet, go to the "Security" tab, and associate the relevant Security Group. (*Figure 1.1.26: security group Details) (This step and the Figure not much clear)*

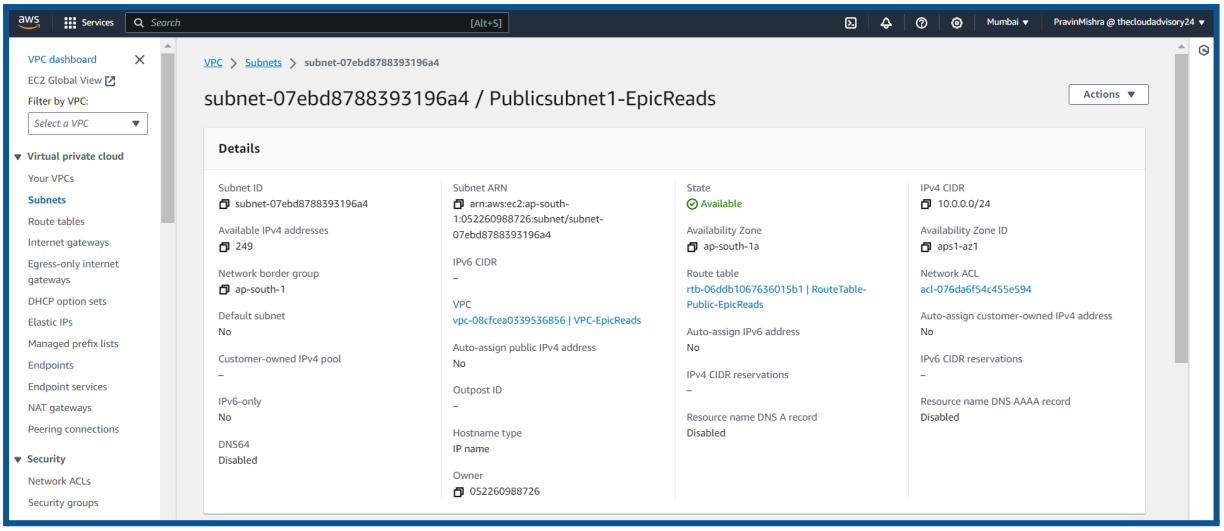


Figure 1.1.26: security group Details

4. **Security Best Practices for Private Subnets:**
    - For private subnets, ensure that the associated NACL and Security Group configurations restrict inbound traffic to only what is necessary.
    - Specifically, deny inbound traffic that is not required for the internal functioning of your resources.
    - This helps ensure that resources in private subnets are not directly accessible from the internet.
5. **Review and Confirm:**
    - Double-check the configurations of NACLs and Security Groups for both public and private subnets.

✅ **Expected Output:** After completing this task, you should have a VPC with properly configured public and private subnets, an attached Internet Gateway (IGW), and updated route tables. The network should follow best practices, providing secure and scalable infrastructure.

Task 2. Launching and Configuring a Linux EC2 Instance at EpicReads

STEP 1: Launching an EC2 Instance in AWS

a. **Sign in to AWS Console:** Go to the AWS Management Console at https://console.aws.amazon.com/ (*Figure 1.1: Search result*).

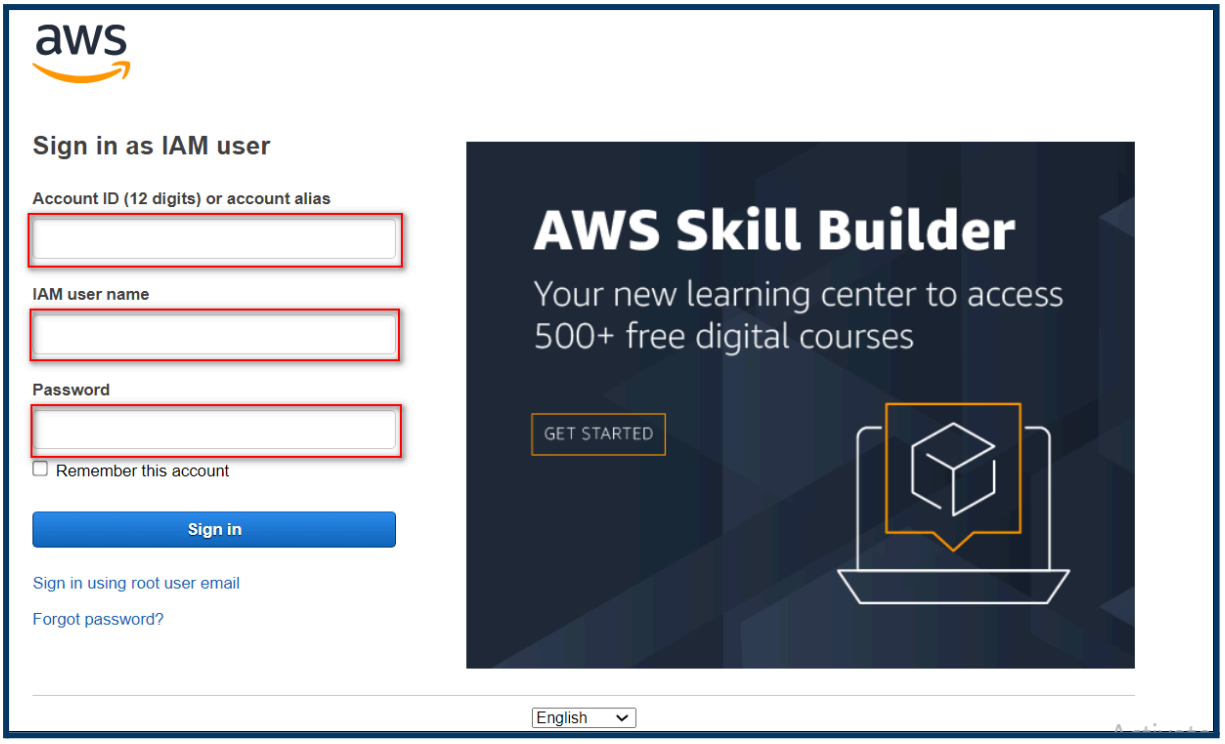


Figure 1.1: Search result

📓 ***Note***: Log in with your credentials. (**Always user IAM user to login** followed by Account ID and password).

ℹ️ **Learn More (IAM user):** IAM (Identity and Access Management) users in AWS are entities that you create in AWS that are associated with specific permissions to access and manage AWS services and resources. These users are separate from your AWS account and allow you to grant or restrict access to AWS resources for individuals, applications, or services. You can also learn more from the IAM User Guide.

b. Navigate to EC2 Dashboard:

Click on the "Services" dropdown in the top left corner.

Select "EC2" under the "Compute" section. (*Figure 1.2: EC2 Search result)*

Figure 1.2: EC2 Search result

c. Launch Instance:

Click on the "Instances" link in the EC2 Dashboard.

Press the "Launch Instance" button.(*Figure 1.3: Launch Instance Tab)*

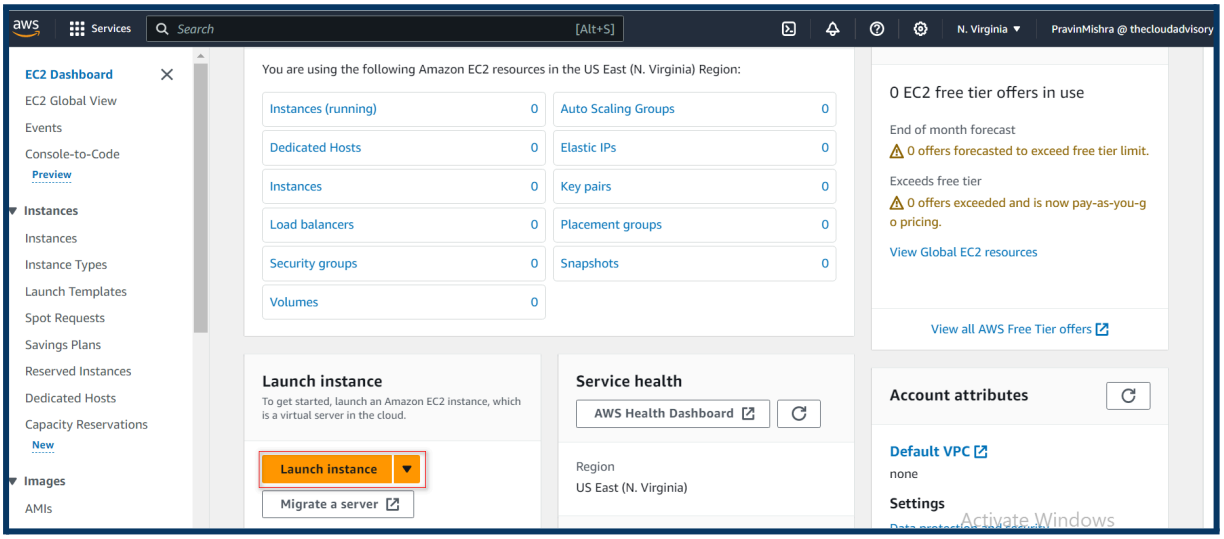


Figure 1.3: Launch Instance Tab

Name the instance you are going to create!

d. Choose an Amazon Machine Image (AMI):

Select "Amazon Linux 2 AMI" from the list of available AMIs.
Click on "Select" to proceed. *Figure 1.4: Choose an Amazon Machine Image (AMI)*

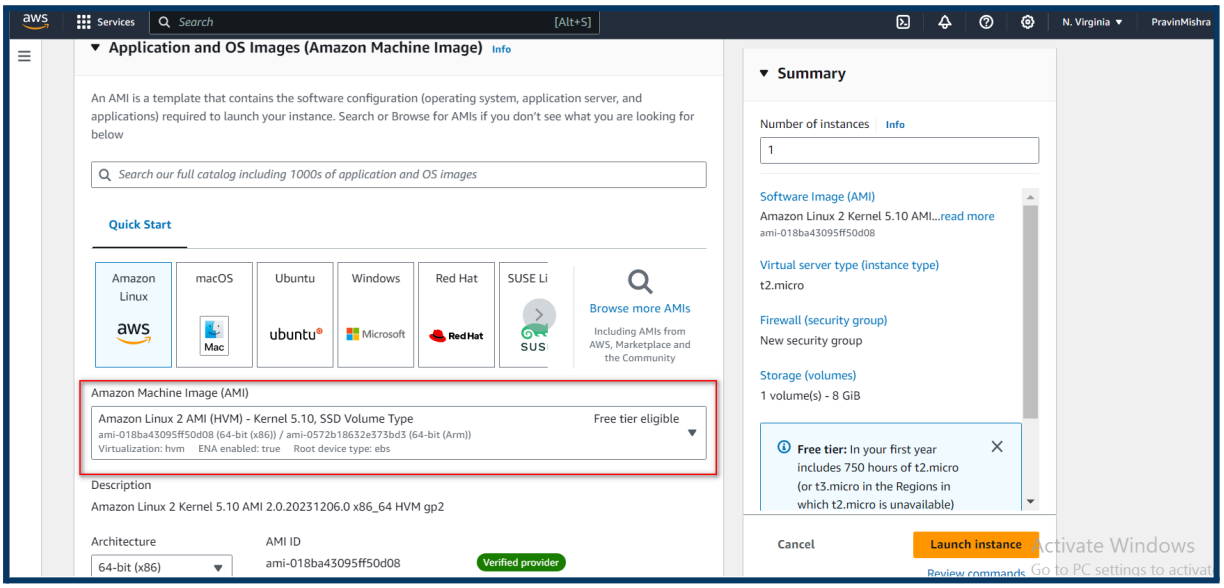


Figure 1.4: Choose an Amazon Machine Image (AMI)

✅**Expected Output**: Successful selection of the instance type.

💡 **Consider**: Always ensure that the selected AMI aligns with your specific application or system requirements.

ℹ️ **Learn More (AMI): AMI** stands for **Amazon Machine Image**. It's a pre-configured template that contains the necessary information to launch an instance, which is a virtual server in the Amazon Elastic Compute Cloud (EC2). AMIs include the **operating system**, **application server**, and any additional software needed to deploy an application. Read more about AWS Documentation on Amazon Machine Images (AMIs)

e. Choose `t3.micro` as the instance type for this exercise.
(Figure 1.5: Choose an Instance Type)

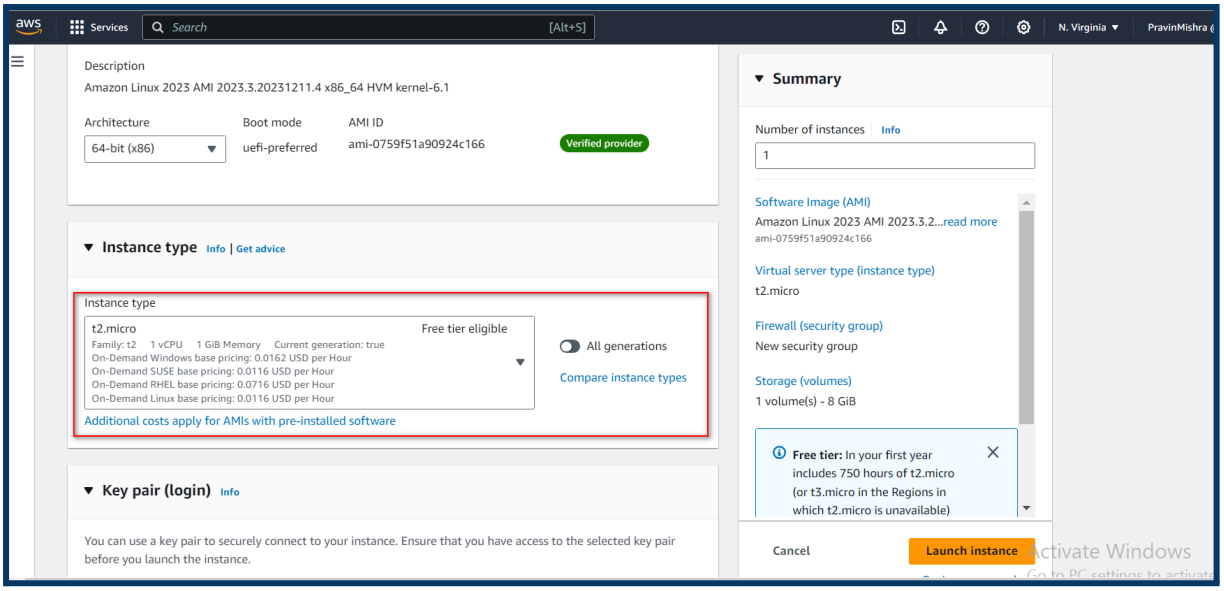


Figure 1.5: Choose an Instance Type

✅**Expected Output:** Successful selection of the instance type.

💡 **Consider**: Always ensure that the chosen instance type (t3.micro) meets the resource requirements and capacity needed for your specific use case or workload.

ℹ️ **Learn More (Instance Type): Instance Type**, refers to the **configuration and specifications of the virtual machine you'll deploy in AWS**. The t3.micro instance type is part of the T3 family, designed to offer a balance of compute, memory, and network resources suitable for various workloads. Read more about AWS EC2 Instance Types

STEP 2: Key Pair Creation:

a. Create a new key pair during the instance setup process. This key pair will be used to securely SSH into the EC2 instance.(*Figure 1.6: Key Pair Creation)*

- Select "Create a new key pair" in the Key pair(login).
- Enter a name for the key pair.
- Download the key pair file (.pem) and keep it secure.

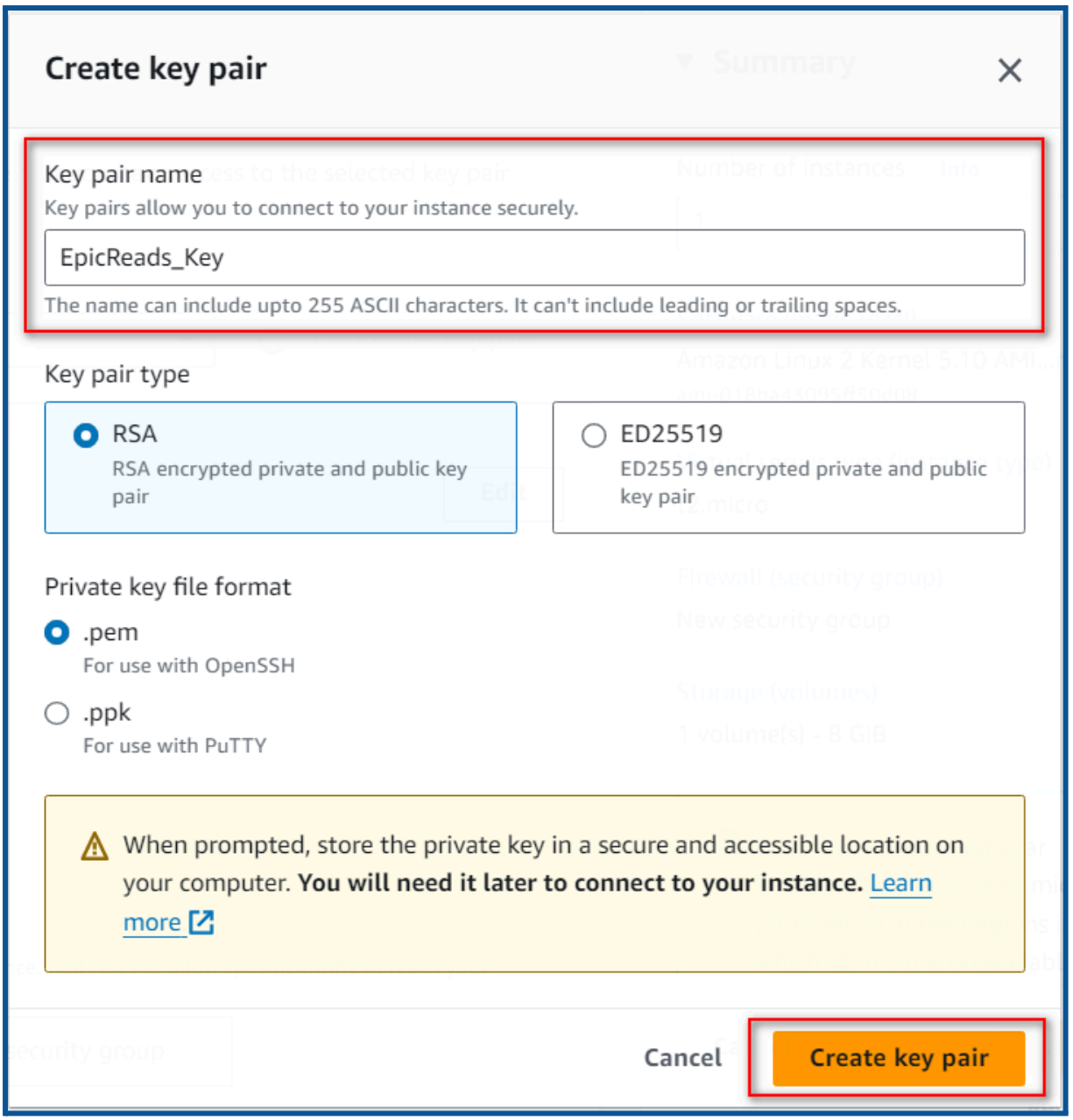


Figure 1.6: Key Pair Creation

✅**Expected Output**: A newly generated key pair file (.pem) downloaded and stored securely.

📓***Note***: Key pairs in AWS are used for secure access to instances running on Amazon EC2.

ℹ️ **Learn More (Key pair Creation): Key pairs** in AWS are used for **secure access to instances** running on Amazon EC2. They consist of a public key that AWS stores, and a private key file that you can download to access your instances.Read more about EC2 Key Pairs Documentation.

STEP 3: VPC and Network Configuration:

a. Use an existing VPC, ensuring it's connected to a public subnet.
b. Enable "Auto-assign public IP" to ensure the instance is accessible over the internet. (*Figure 1.7: VPC and Network Configuration)*

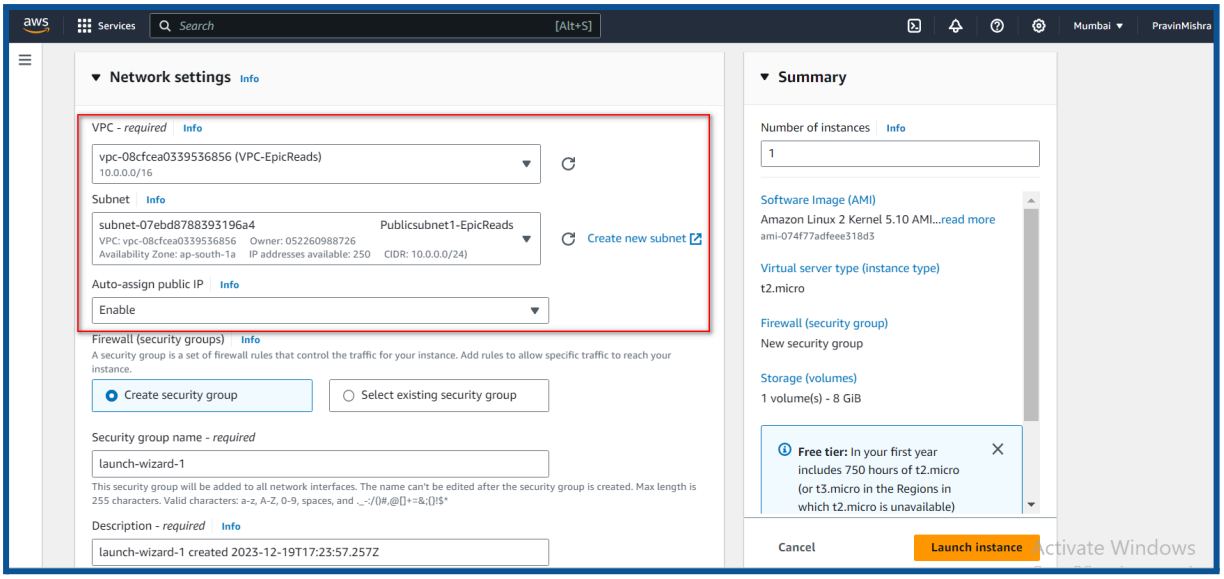


Figure 1.7: VPC and Network Configuration

✅**Expected Output:** Confirmation that the VPC is connected to a public subnet and 'Auto-assign public IP' is enabled for internet accessibility.

ℹ️ **Learn More (VPC and Network Configuration): VPC (Virtual Private Cloud)** in AWS allows you to create a **virtual network environment, including subnets, route tables, and security settings**, to launch AWS resources like EC2 instances..Read more about [AWS VPC User Guide](#).

STEP 4: Security Group Setup:

a. Create a new security group for instance and specify the group name and description. (*Figure 1.8: Security Group Setup)*

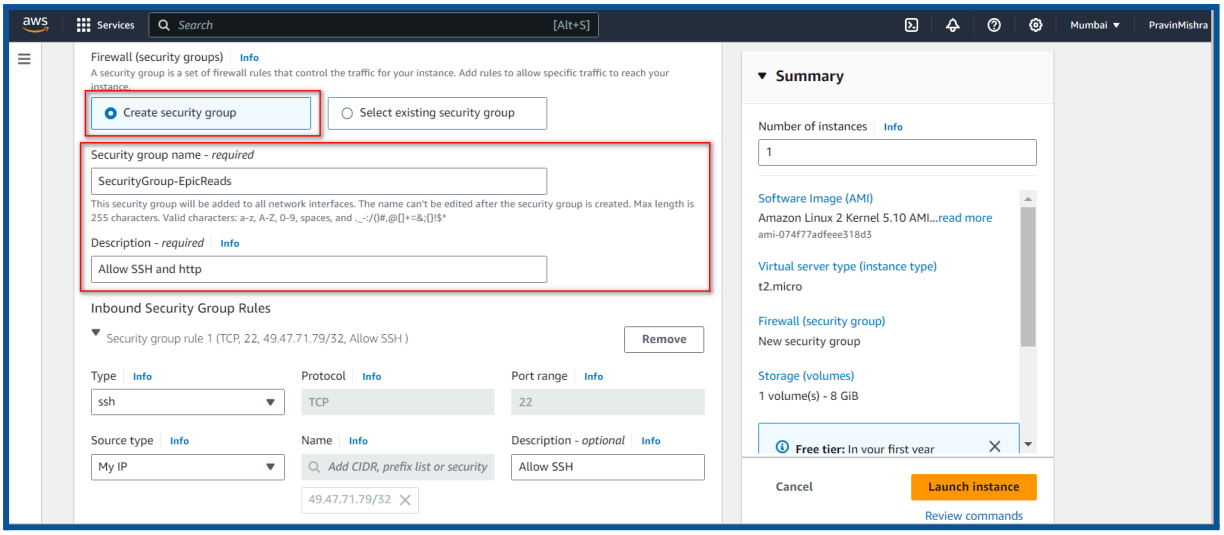


Figure 1.8: Security Group Setup

b. Configure inbound rules to allow traffic on ports 22 (SSH) and 80 (HTTP) from the internet. (*Figure 1.9: Security Group Rules).*

Port 22:

Port 22 is similar to having a secret key that allows us to securely access and control our computer or server from anywhere, like opening a locked door with a special code. It helps us manage our computer or server even when we are far away.

Port 80:

Port 80 acts as the main entrance to the internet, just like the front door of a house where websites come and go. When we want people to see a website or application we have made, the information travels through this entrance.

In Amazon Web Services (AWS), we use port 22 as a special key to remotely access our computer or server (like using a secret code), and port 80 to showcase websites or applications on the internet (like the main entry point). We create a website and use Apache for a new project, it will need port 80 to display the application on the internet.

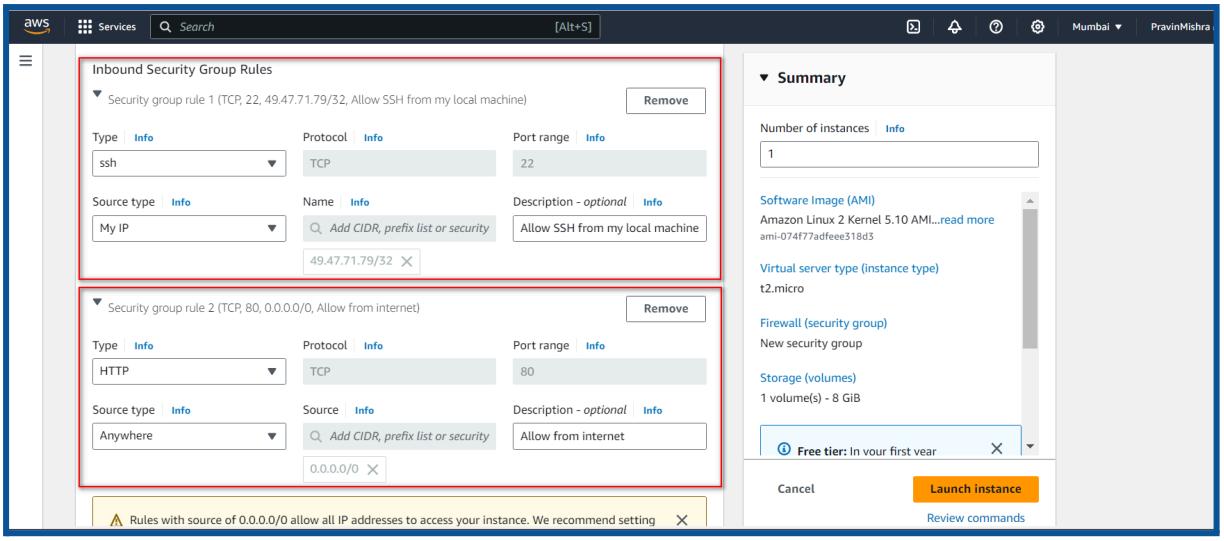


Figure 1.9: Security Group Rules

✅**Expected Output:** Verification of newly configured security rules confirming SSH access limited to 'My IP' on port 22 and allow HTTP access from 'Anywhere' on port 80.

⚠️ **Caution**: It's crucial to accurately configure inbound rules to prevent potential security vulnerabilities or connectivity issues.

ℹ️ **Learn More (Security Group Setup): Security Groups** in AWS act as **virtual firewalls that control inbound and outbound traffic** for AWS instances. Read more about EC2 Security Groups Documentation.

STEP 5: Storage Configuration:

a. Modify the instance's storage settings, setting the size to 10 GB.(*Figure 1.10: Storage Configuration)*

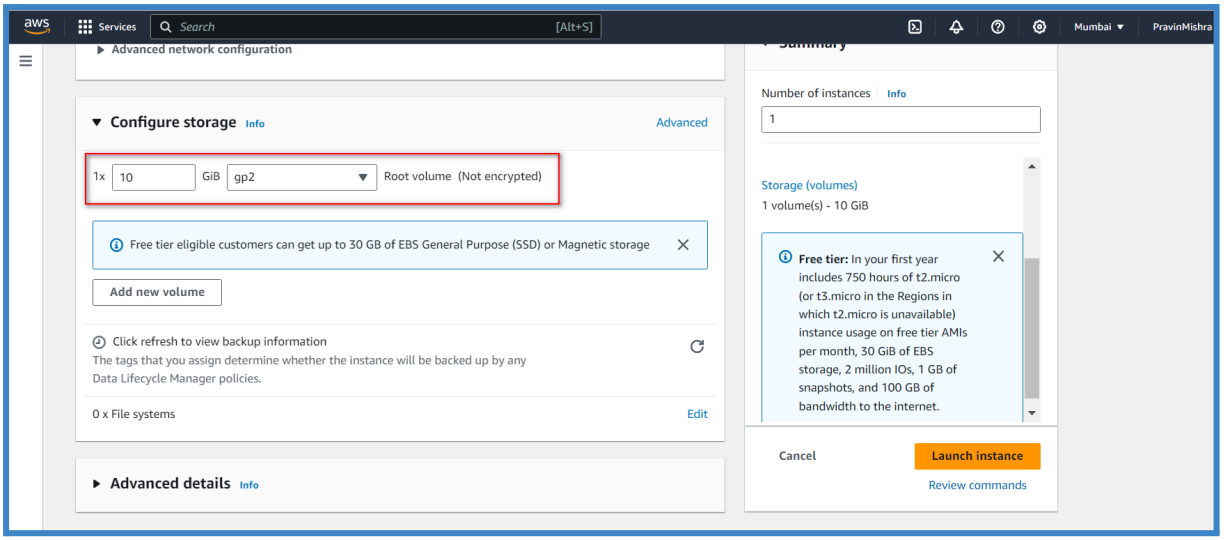


Figure 1.10: Storage Configuration

🚀 Fantastic work! You're moments away from launching your instance. Simply click on the 'Launch Instance' button to proceed with the final step." (*Figure 1.10: Launch Instance Final)*

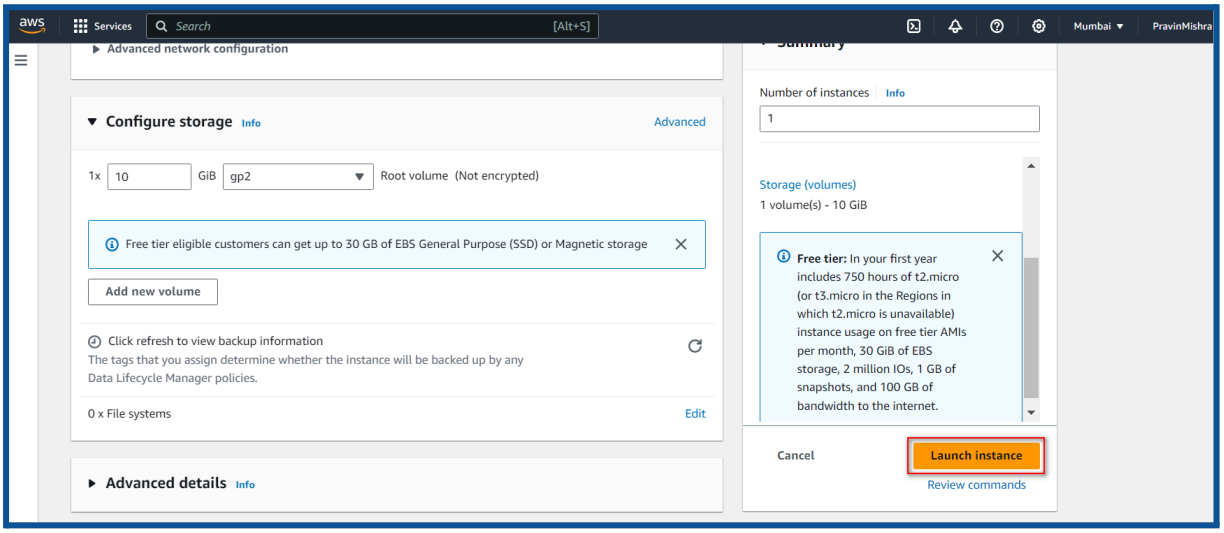


Figure 1.10: Launch Instance Final

STEP 6: SSH Connection and Testing:

a. Select your EC2 instance: Click on the checkbox next to your EC2 instance to select it.
b. Click on the "Connect" button: This should be located at the top of the EC2 Dashboard when you have your instance selected.
c. Go to the SSH client section: Once you're in the "Connect" window, you should see an option for an SSH client. There, you'll find an example of the SSH connection string.

d. Copy the connection string along with the username: In the SSH client section, you'll typically find a command that looks similar to this:
ssh -i /path/to/your-key.pem ec2-user@your-instance-public-ipv4
(*Figure 1.11: SSH Client Console)*

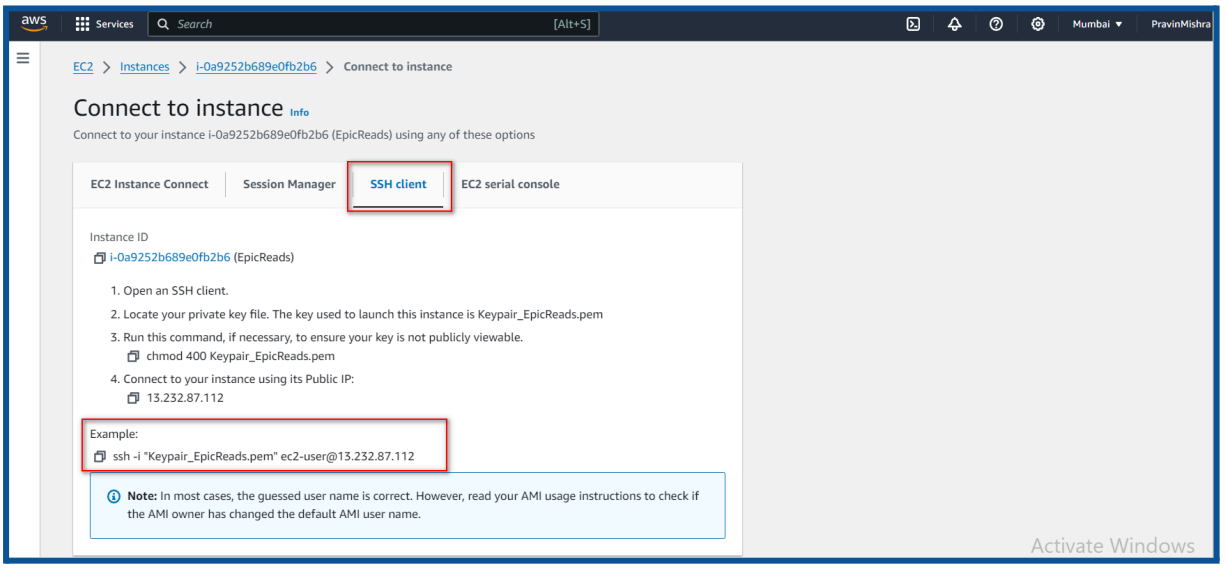


Figure 1.11: SSH Client Console

STEP 7: Connecting via Command Line (Terminal):

a. Open a terminal window on your local machine (Linux/macOS) or use an SSH client like PuTTY on Windows or Git Bash.
b. Navigate to the directory where your private key (.pem) file is located: Use the `cd` command to navigate to the directory where your key is stored.
For example: cd /path/to/your-key-pair-directory
c. Change permissions for the private key: Ensure that the permissions for the private key are secure.

💻 **Command:** *chmod 400 your-key-pair.pem*

The command "chmod 400 your-key-pair.pem" helps protect a file called "your-key-pair.pem" by allowing only the owner to read it. This keeps the information inside safe and secure from others who shouldn't have access to it.

d. Connect to the EC2 instance using SSH: Use the SSH command in the terminal to connect to your EC2 instance (*Figure 1.12: Connecting via Command Line)*. The format is:

💻 **Command**: ssh -i your-key-pair.pem ec2-user@your-instance-public-ipv4

Example: ssh -i "Keypair_EpicReads.pem" ec2-user@13.232.87.112

e. Once connected, perform a basic network test by pinging an external site like google.com.

📓 **Note**: Always ensure the correct path to the key pair file and instance public IP when connecting via SSH. (common errors)

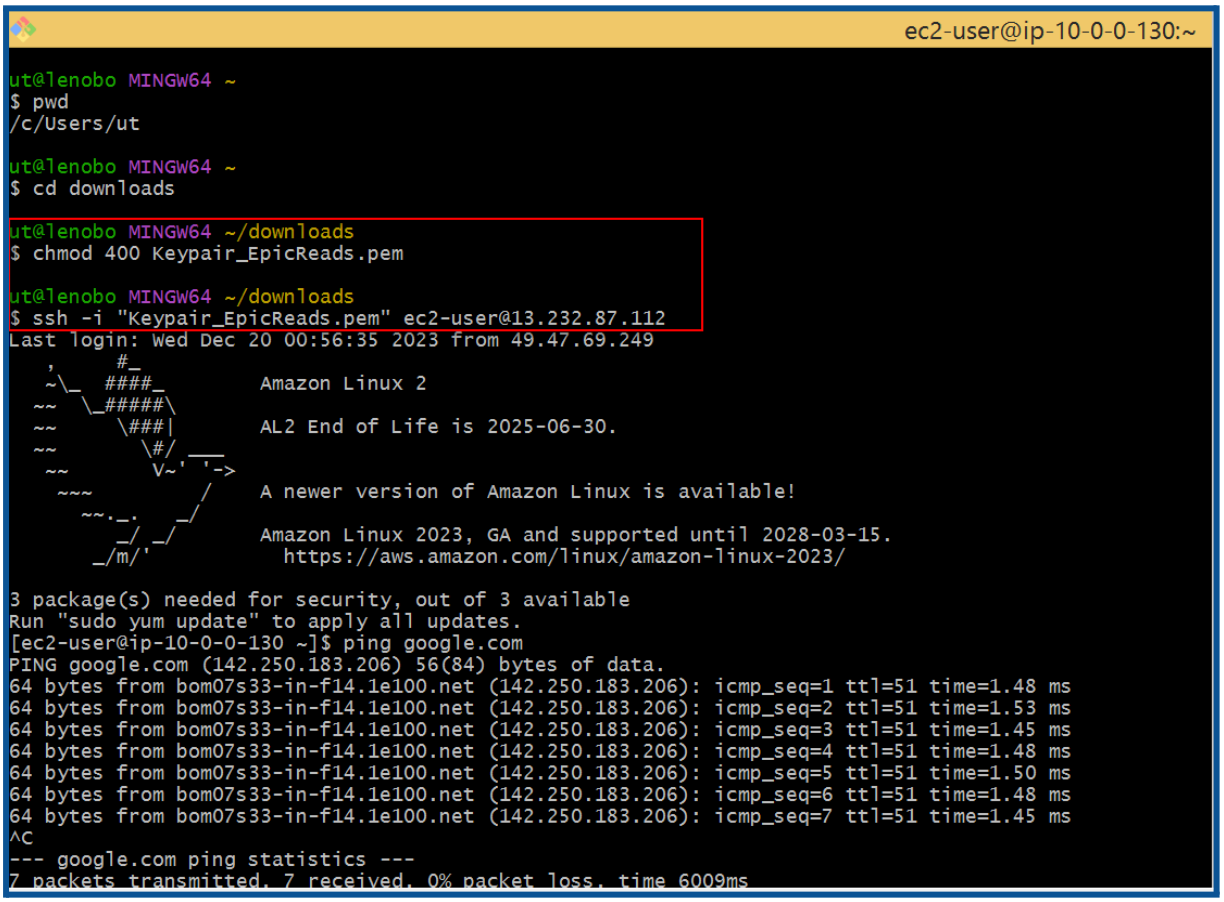


Figure 1.12: Connecting via Command Line

✅**Expected Output:** Successful SSH connection established to the EC2 instance using the provided key pair and IP, followed by a successful network test by pinging an external site like google.com.

Congratulations on launching your first EC2 instance!

Task 3 : Launching, Configuring, and Installing WordPress on a Linux EC2 Instance at EpicReads

Task Overview:

Once you've successfully launched and connected to your Linux EC2 instance, the subsequent step involves installing and configuring WordPress. This will replicate a real-life scenario, allowing EpicReads to oversee digital content, blogs, or an online catalog of books.

This guide covers the following:

1. Installing and configuring a WordPress application on the Amazon Linux 2 EC2 instance.
2. Verifying that the WordPress site is accessible and fully functional from the internet.

COMMON ERRORS

1. **YUM Update Failure**:
- **Error**: `sudo yum update` fails to update packages due to network issues, conflicting repositories, or missing dependencies.
- **Solution**: Check network connectivity, verify repositories, resolve dependency issues, and rerun the command. Use `sudo yum clean all` to clear the package cache and try updating again.
2. **Apache Installation Failure:**
- **Error**: `sudo yum install httpd -y` fails due to repository configuration problems, incorrect package names, or interrupted installation.
- **Solution:** Ensure repositories are configured correctly. Verify the package name (`httpd` in this case). Recheck the command and ensure it completes without interruptions.
3. **Apache Service Not Starting:**
- **Error**: `sudo systemctl start httpd` or `sudo systemctl status httpd` commands show errors or indicate that the service is inactive.

- **Solution**: Check logs (`sudo journalctl -xe` or Apache logs) for errors. Verify the configuration files and permissions. Rectify any issues and try restarting the service.

4. **Browser Doesn’t Display Apache Test Page:**

- **Error**: Entering the EC2 instance's public IP address in the browser doesn't load the Apache test page or shows an error.
- **Solution**: Verify the EC2 instance's security group settings; ensure that HTTP (port 80) traffic is allowed. Double-check the public IP address used in the browser. Confirm that the Apache service is running and properly configured.

5. **Misconfigured EC2 Instance:**

- **Error**: Misconfigurations in the EC2 instance settings or network settings prevent Apache from displaying the test page.
- **Solution**: Review EC2 instance settings, networking (VPC, subnet, route tables), and ensure the instance is properly associated with the desired security groups.

6. **Misconfigured Inbound Rules:**

- **Error**: Misconfiguring the inbound rules, such as selecting the wrong type (e.g., choosing "SSH" instead of "MYSQL/Aurora") or setting the wrong source.
- **Solution**: Double-check the rules and ensure the correct type (MYSQL/Aurora, HTTP) is selected. Confirm the source addresses (e.g., "My IP" or specific IP ranges) are set accurately.

7. **Missing Database Configuration:**

- **Error:** Incorrect MySQL database settings in wp-config.php.
- **Solution:** Double-check database details (name, username, password, host) and ensure they match the actual database setup.

STEP 1: Update and Upgrade Packages

Once connected to your EC2 instance, update the package lists and upgrade installed packages by using sudo yum update: (*Figure 2.1: Update and Upgrade Packages*).

💻 ***Command:***

```
sudo yum update
```

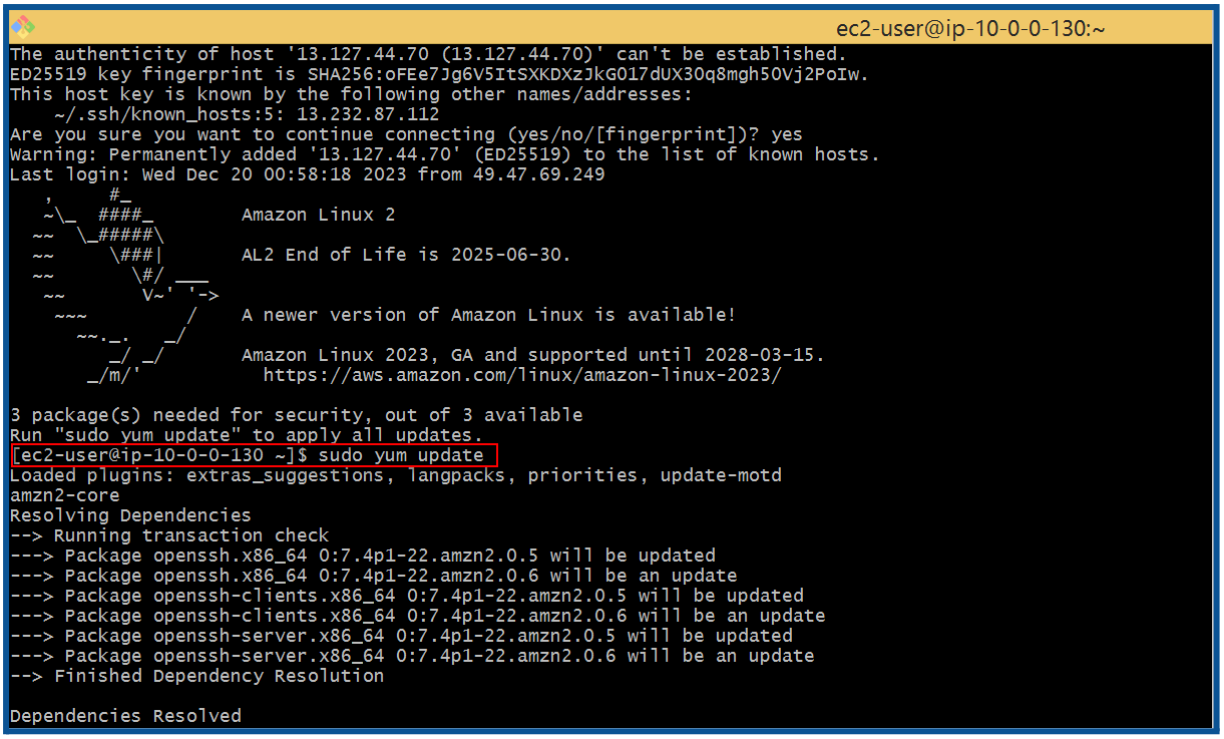


Figure 2.1: Update and Upgrade Packages

✅**Expected Output:** The expected output after executing sudo yum update would display information about the package dependencies being checked and updated.

📓 **Note**: Before proceeding further, it's essential to update the package lists and upgrade installed packages on your EC2 instance.

ℹ️ **Learn More:** For more detailed information about managing packages on Amazon Linux instances, you can refer to the AWS documentation on YUM commands for Amazon Linux.

STEP 2: Install Apache Web Server

Install the Apache web server by using the below command (*Figure 2.2: Install Apache Web Server).*

💻 ***Command:***

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

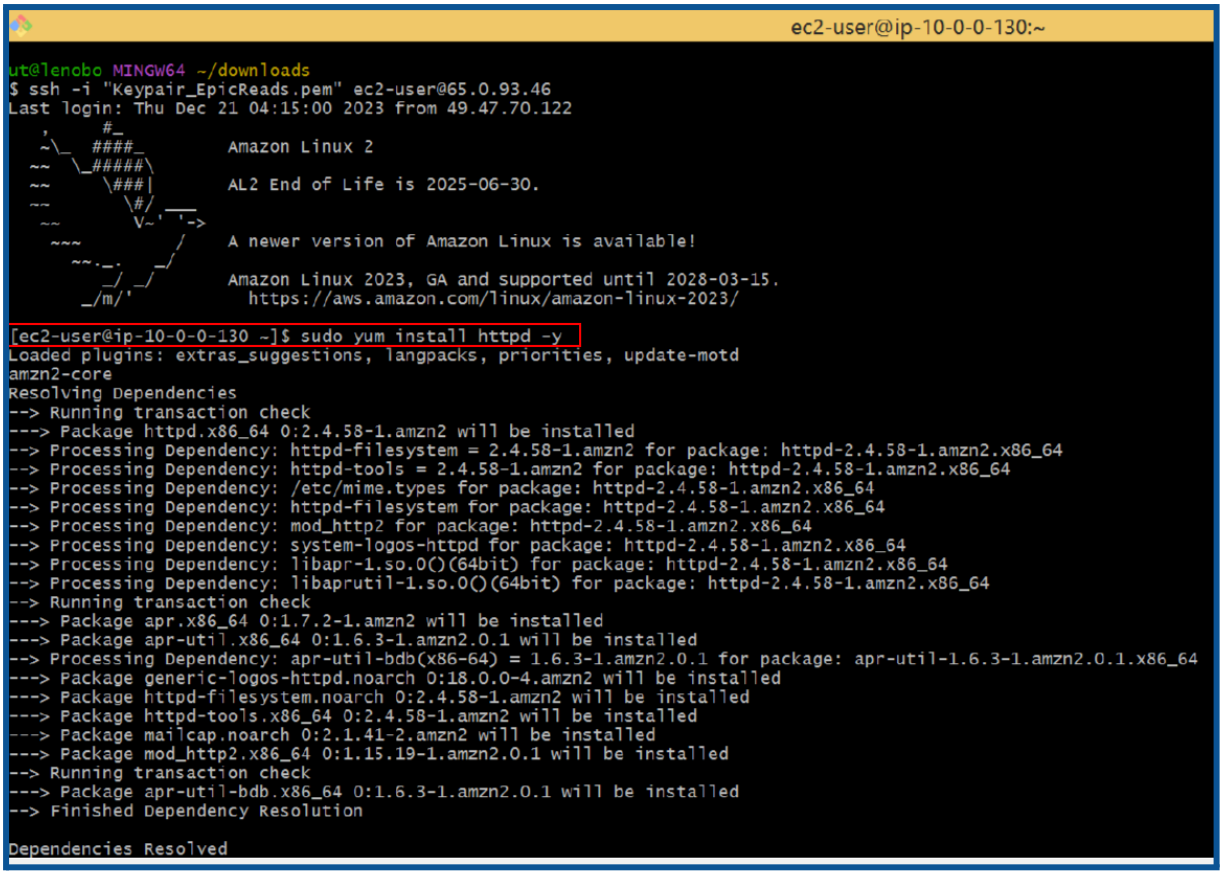


Figure 2.2: Install Apache Web Server

Start the Apache service and enable it to start on boot (*Figure 2.3: Apache service status)*

💻 ***Command:***

```
sudo systemctl start httpd
sudo systemctl status httpd
```

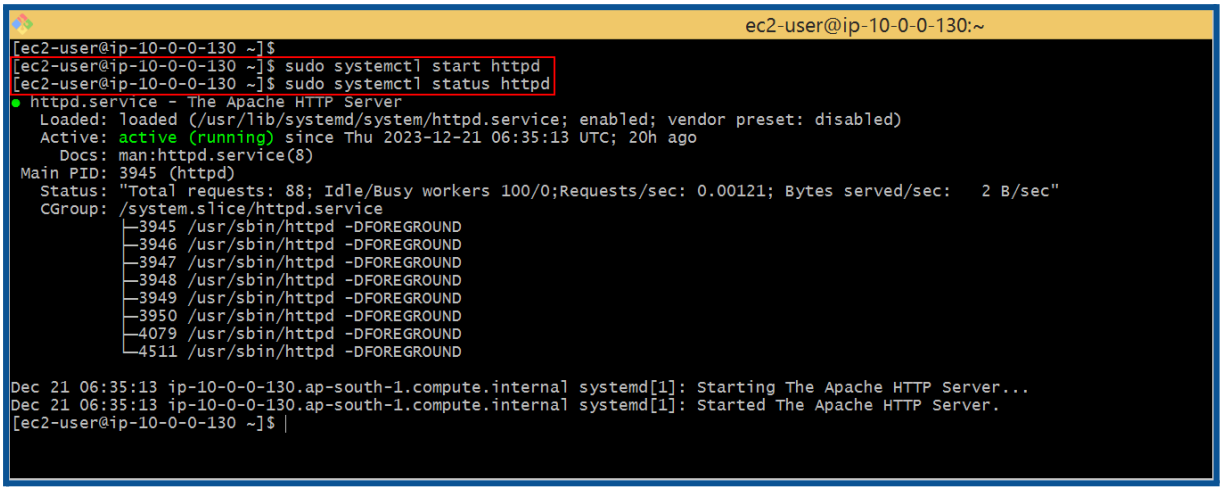


Figure 2.3: Apache service status

✅**Expected Output:** `sudo systemctl status httpd` should indicate that the Apache service is active and running without any errors.

To verify the deployment of Apache HTTPD through the browser:

- Obtain the public IP address of your EC2 instance (*Figure 2.4: Public IP address).*

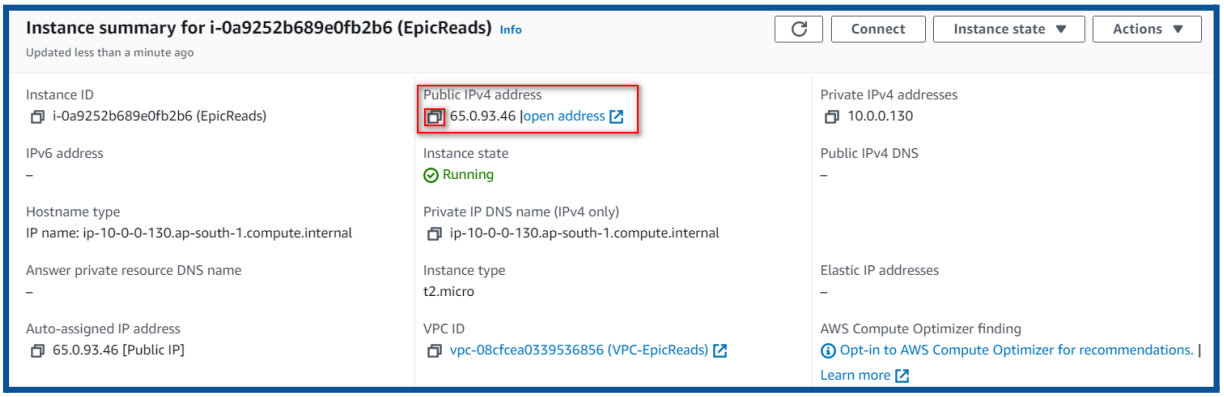


Figure 2.4: Public IP address

- Open your web browser.
- Enter the EC2 instance's public IP address into the browser's address bar.
- Press "Enter" to load the page (*Figure 2.5: Apache Test Page).*

Example: `http://[EC2_Public_IP]`

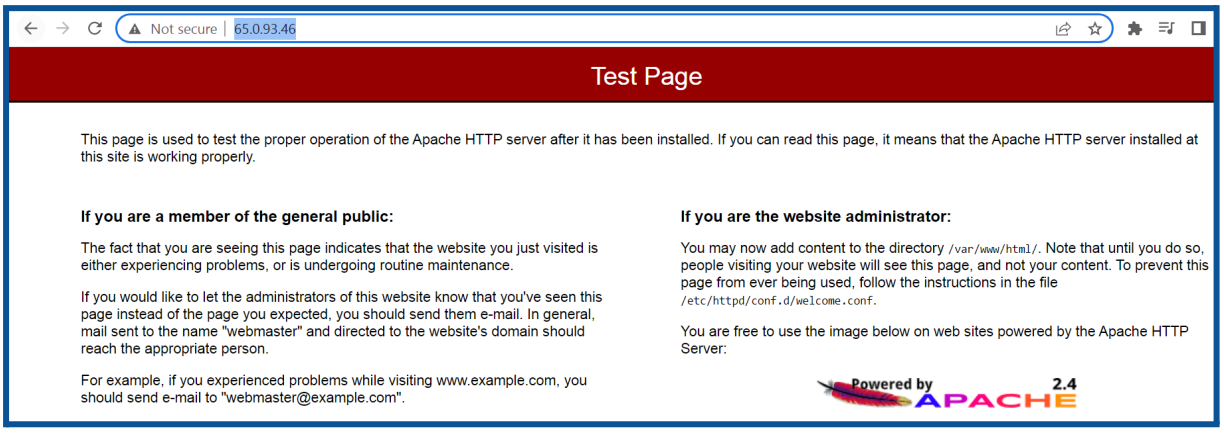


Figure 2.5: Apache Test Page

✅**Expected Output:** The the expected output in the browser should display the default Apache test page, indicating that Apache HTTPD has been successfully deployed and is functioning correctly.

📓 **Note**: The appearance of the Apache test page confirms the s**uccessful deployment of Apache HTTPD on your EC2 instance.** If

the test page does not load, ensure that the installation and configuration steps have been executed correctly. Troubleshoot any issues by revisiting the installation commands and verifying the server status.

ℹ️ **Learn More:** To dive deeper into the Apache HTTP Server, AWS provides additional information about setting up Apache on Amazon Linux.

STEP 3: Install PHP and MySQL

Install PHP along with necessary PHP extensions required by WordPress(*Figure 2.6: install PHP).*

💻 ***Command:***

```
sudo amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2
```

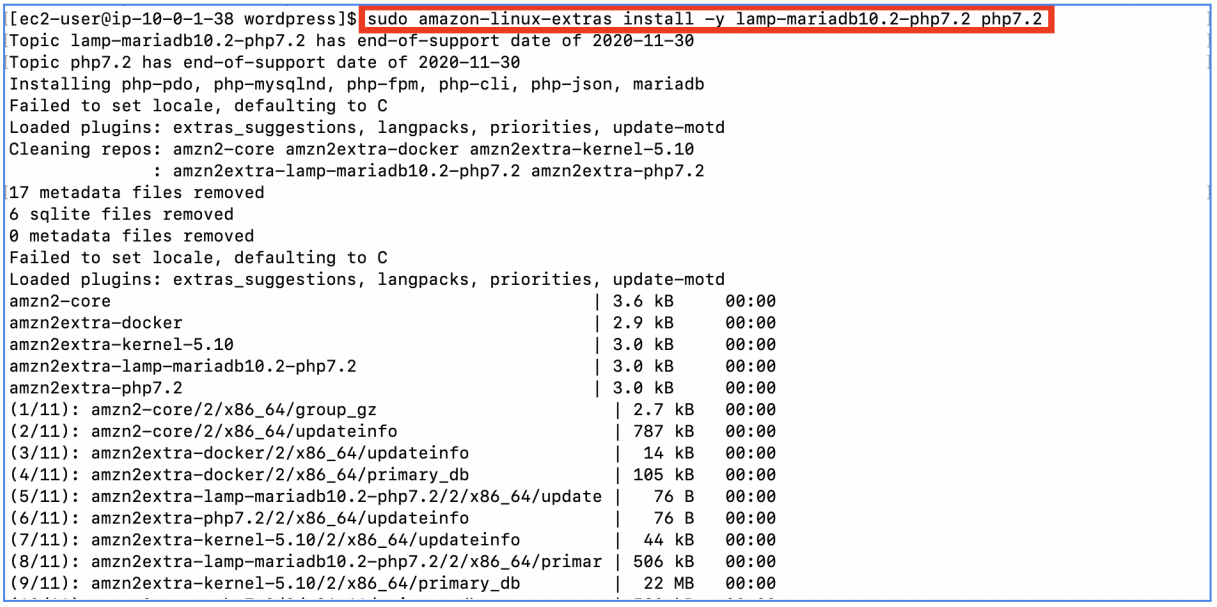


Figure 2.6: install PHP

This command installs PHP along with various extensions that are essential for WordPress to function properly. Here's a breakdown of the extensions installed:

- php-mysqlnd: Enables PHP to communicate with MySQL databases.
- php-gd: Provides image manipulation functionalities for WordPress.

- php-xml: Allows PHP to handle XML data.
- php-mbstring: Necessary for multibyte string support in PHP.

Once these steps are completed, PHP will be installed and ready to support your WordPress installation on your system.

✅ **Expected Output:** This command installs PHP along with essential extensions necessary for **WordPress functionality**. No specific output is expected if the installation is successful.

📓 **Note:** These PHP extensions provide critical functionalities to WordPress. They enable PHP to communicate with MySQL, handle XML data, support multibyte strings, and perform image manipulation tasks.

ℹ️ **Learn More:** For further details about PHP and its modules, you can refer to the AWS documentation on PHP on Amazon Linux.

Install MySQL:

Install MySQL server package for the WordPress database(*Figure 2.7: install MySQL).*

💻 ***Command:***

```
sudo yum install -y mysql
```

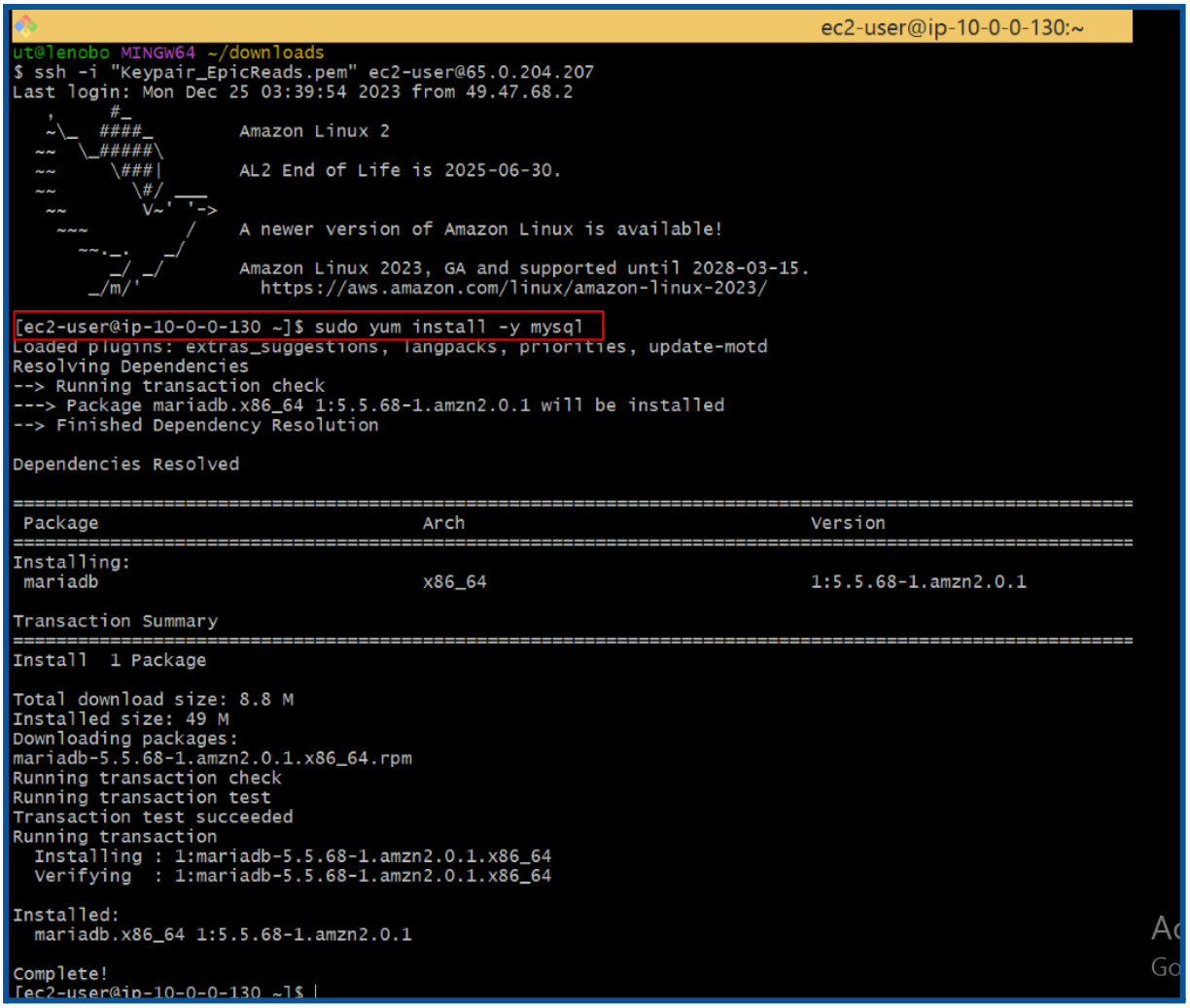


Figure 2.7: install MySQL

⚠️ **Caution:** The command mentioned might not specifically install MySQL but could potentially **install a package** that includes MySQL or MySQL client.

📓 **Note**: It's crucial to install MySQL correctly to establish the WordPress database. Ensure the accuracy of the MySQL package name before executing the installation command.

STEP 4: WordPress Setup

1. Download and install WordPress on the EC2 instance(*Figure 2.8: Download WordPress)*

a. Navigate to the Web Directory:

- Move to the directory where you want to install WordPress. For instance, in

```
cd /var/www/html/
```

b. Download WordPress:

- Fetch the latest version of WordPress.

💻 ***Command:***

```
sudo wget https://wordpress.org/latest.tar.gz
```

c. Extract WordPress Archive:

- Unzip the downloaded WordPress file.

💻 **Command:**

```
sudo tar -xzvf latest.tar.gz
```

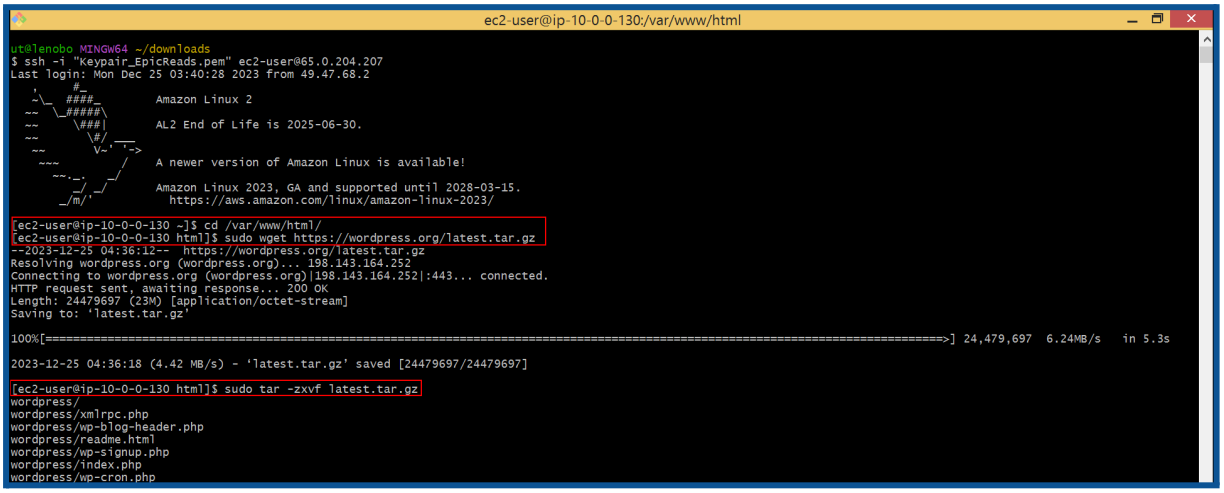


Figure 2.8: Download WordPress

✅**Expected Output:** Successful downloading of the WordPress archive (`latest.tar.gz`) indicated by completion message.

📓 **Note:** Ensure you download WordPress from a reliable source to avoid issues with the installation package.

ℹ️ **Learn More:** For detailed information about downloading WordPress, refer to AWS documentation on *Installing WordPress on Amazon Linux*.

d. Create WordPress Configuration(*Figure 2.9: Permission for Wordpress file).*

- Navigate into the WordPress directory.

💻 **Command:**

```
cd wordpress
```

Create the configuration file by copying the sample configuration.

```
sudo cp wp-config-sample.php wp-config.php
```

Set Permissions: Adjust the permissions for WordPress files.

```
sudo chown -R apache:apache /var/www/html/wordpress
sudo chmod -R 755 /var/www/html/wordpress
```

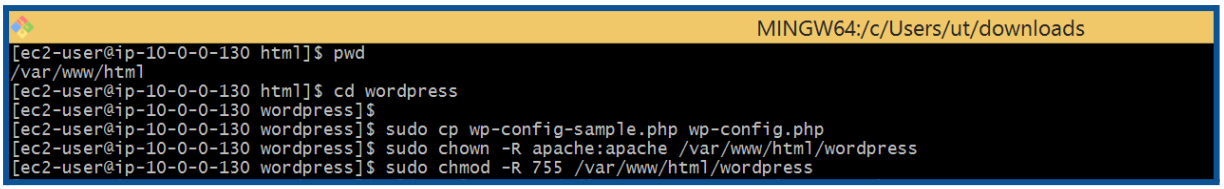


Figure 2.9: Permission for Wordpress file

✅**Expected Output:** Successful creation of the WordPress configuration file (*`wp-config.php`*) and adjustment of permissions on WordPress files without any errors.

📓 **Note:** Proper file ownership and permissions are crucial for WordPress functionality. Ensure these settings are correctly applied.

STEP 5: Create Wordpress Database

Before even creating the Database, creating a subnet group is important

Navigate to the left navigation pane of the **RDS dashboard** and click on the *Subnet groups* (*Figure 2.10: Create Subnet groups*)

Figure 2.10: Create Subnet groups

Navigate to the RDS console and click on "Create Database."(*Figure 2.10:Create Database Tab)*

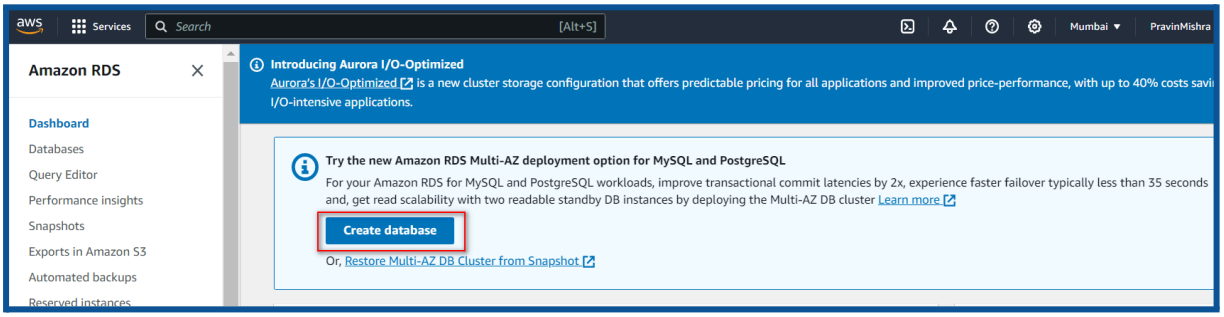


Figure 2.10:Create Database Tab

Under Engine options, choose "MySQL." (*Figure 2.11:Create Database Option)*

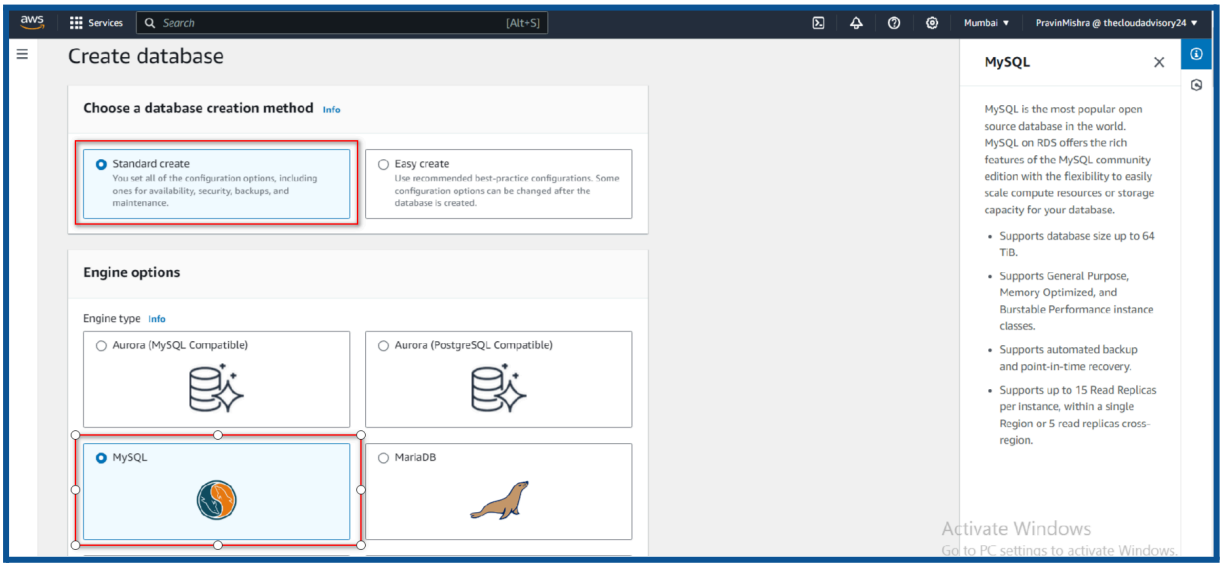


Figure 2.11:Create Database Option

Select the "Free tier" template in the Templates section(*Figure 2.12: Free Tier Template)*

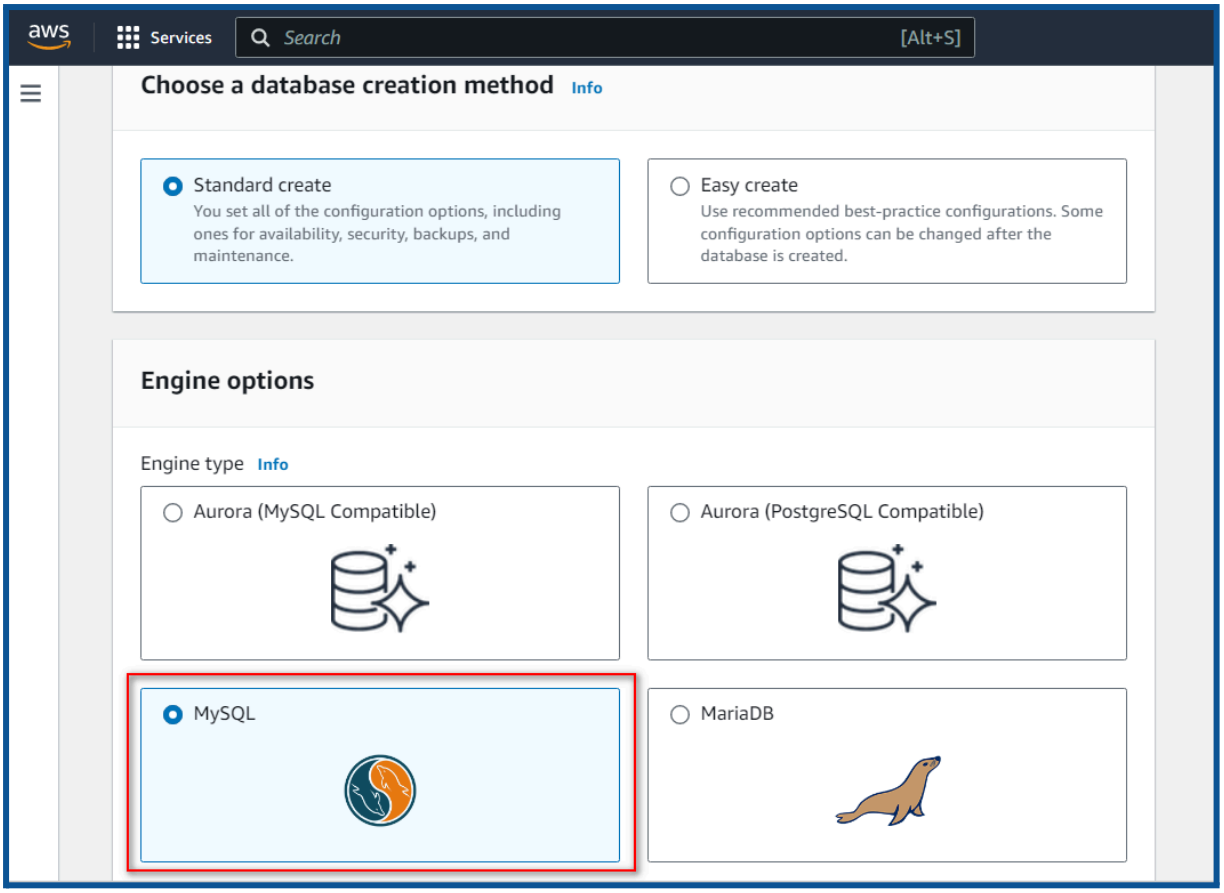


Figure 2.12:Free Tier Template

In the Settings section(*Figure 2.13:Credentials Settings):*

- Enter "wordpress" as the DB identifier.
- Set your desired Master username.
- Define a secure Master password (e.g. Happynewyear#24) for access and also enter the same password on the confirm password tab.

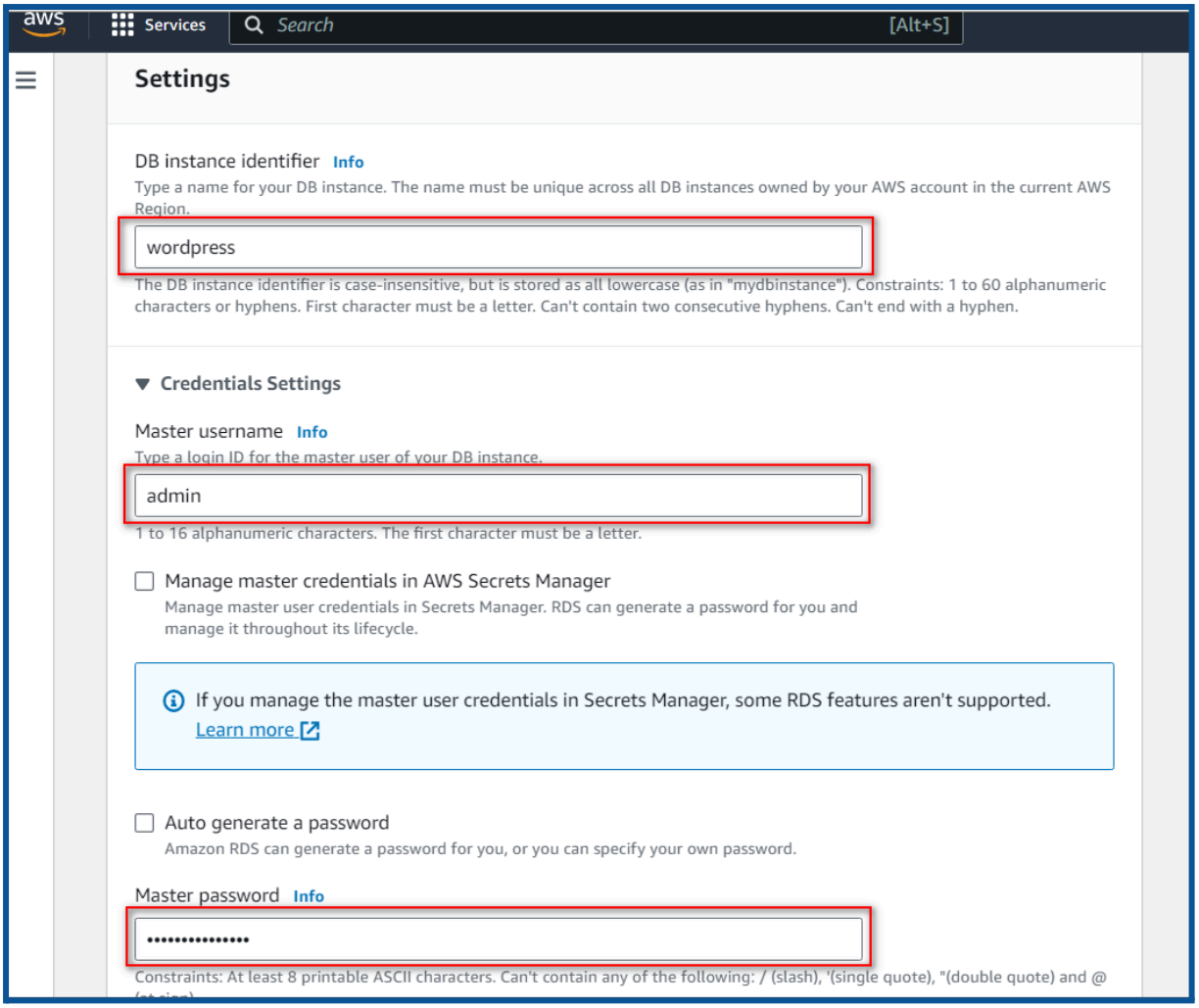


Figure 2.13:Credentials Settings

The instance configuration and storage should remain the same at their default selections(*Figure 2.14:Instance Configuration).*

- Use the default configuration for **Burstable Classes** with the **db.t3.micro** instance type.
- Specify the storage type as **General Purpose SSD (gp2)**.
- Allocate **20 GB** of storage capacity.

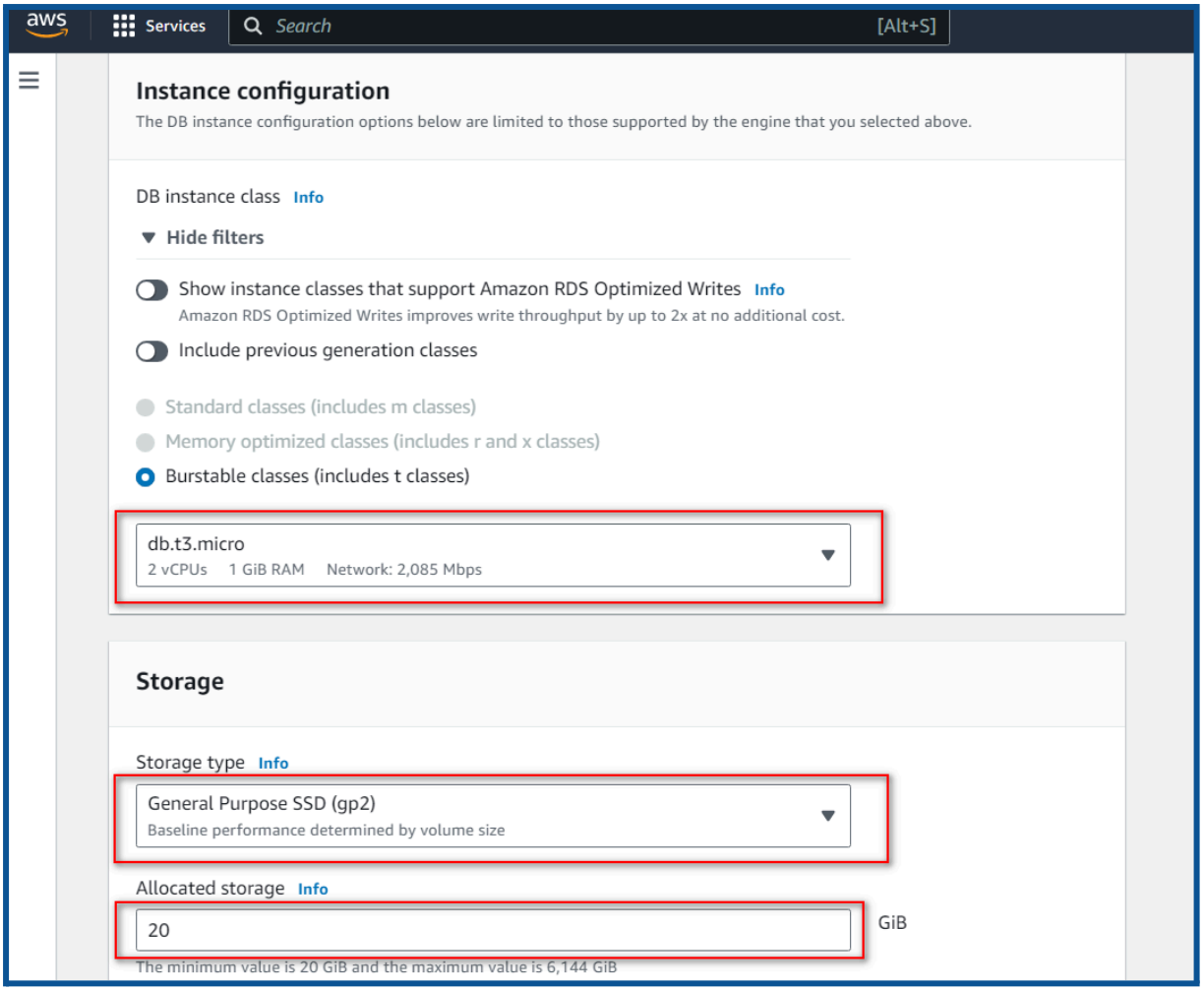


Figure 2.14:Instance Configuration

In the Connectivity section(*Figure 2.15:Compute resource Configuration)*:

- Choose the VPC named "VPC-EpicReads" created in the preceding step.
- For "Public access," select "No" to limit public access to the instance.

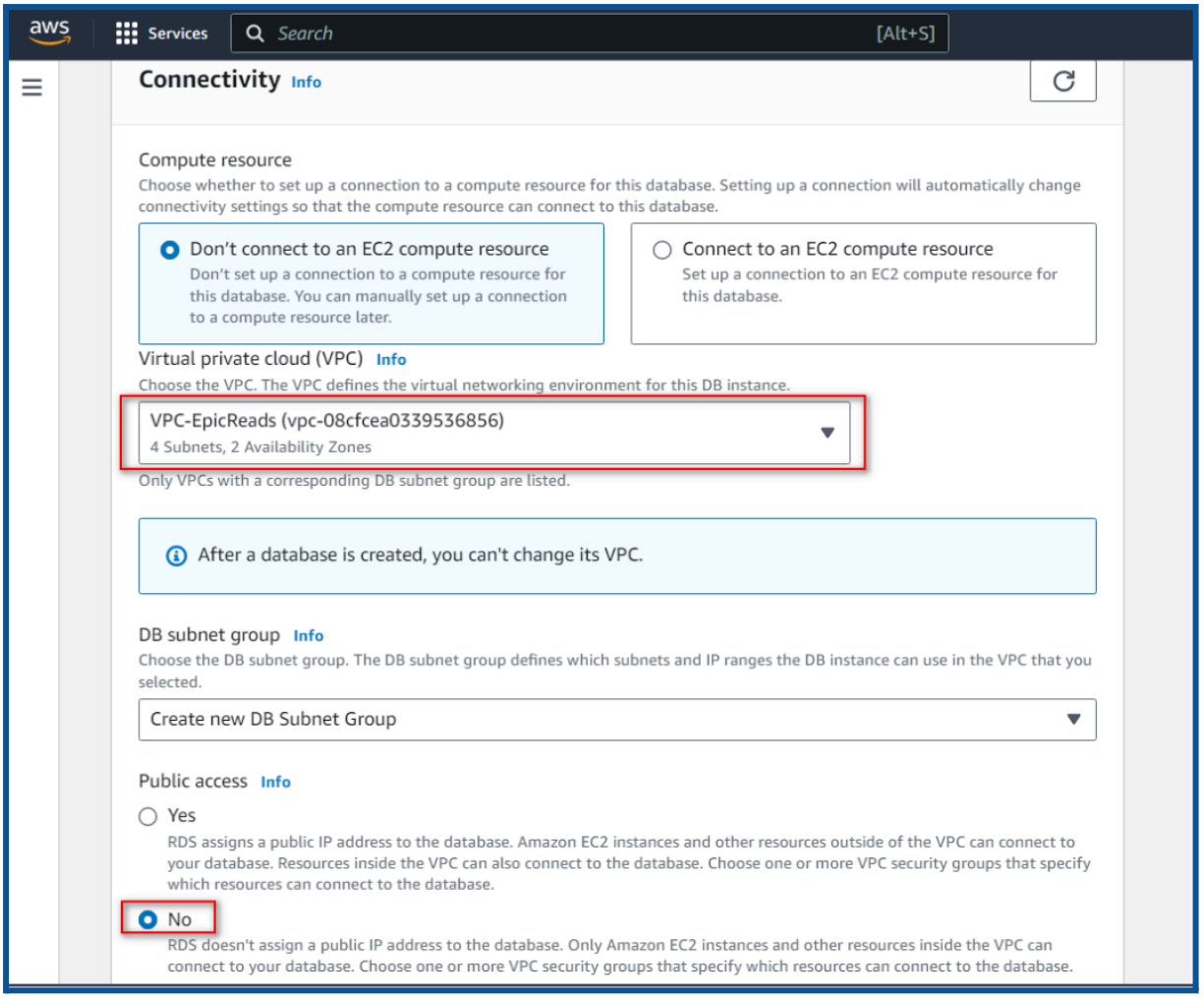


Figure 2.15:Compute resource Configuration

Under "VPC security group," choose "Create new" and input "db-sg" as the name for the new VPC security group (*Figure 2.16: VPC Security Group).*

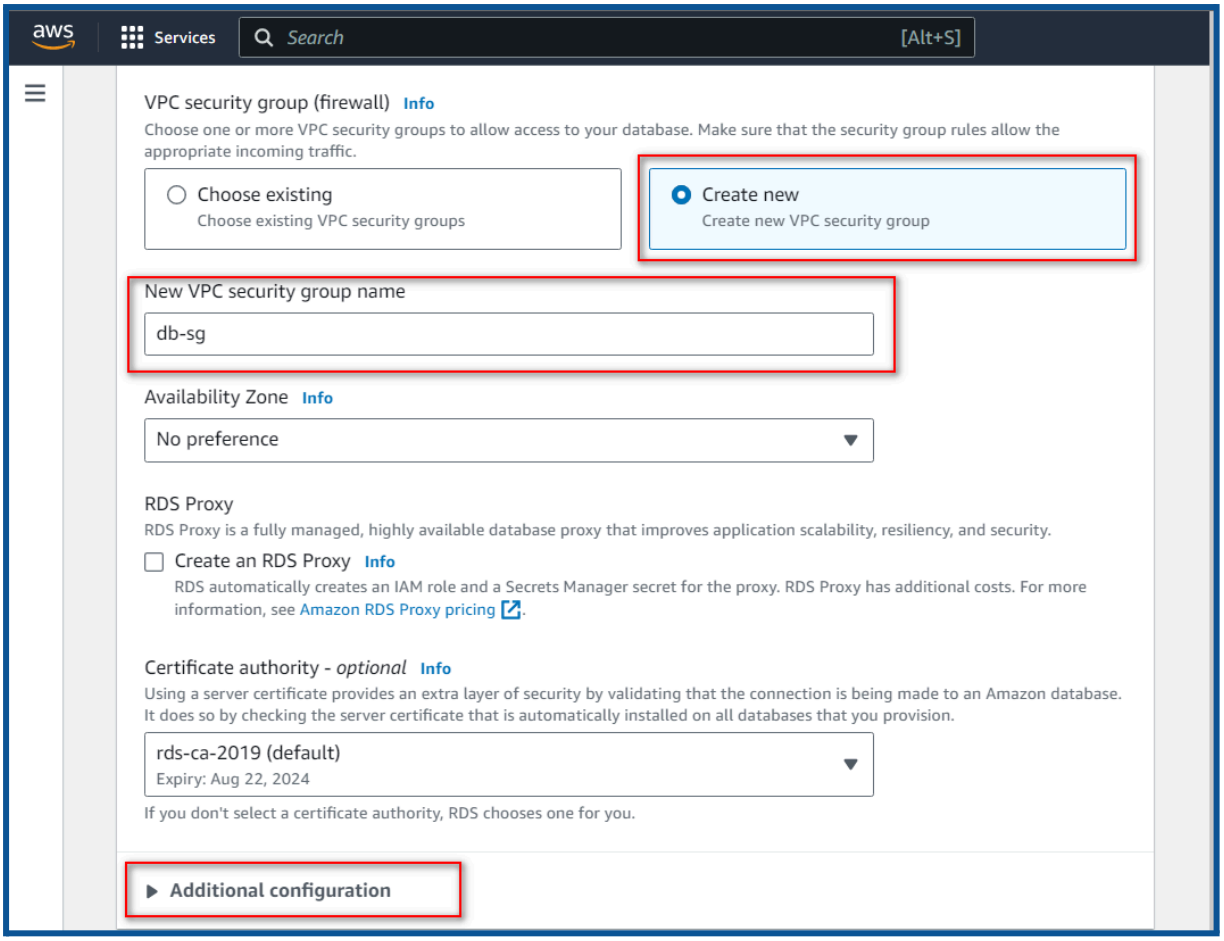


Figure 2.16: VPC Security Group

🔥 **WARNING:** Be cautious with the security group settings as it directly impacts network access to your database instance.

📓 **Note**: The security group configuration controls inbound and outbound traffic for your RDS instance.

Scroll down and click on "Additional connectivity configuration" to reveal further settings (*Figure 2.17: Additional Configuration).*

Enter "wordpress" as the initial database name.
Disable backups and encryption, as they are not required for this project.

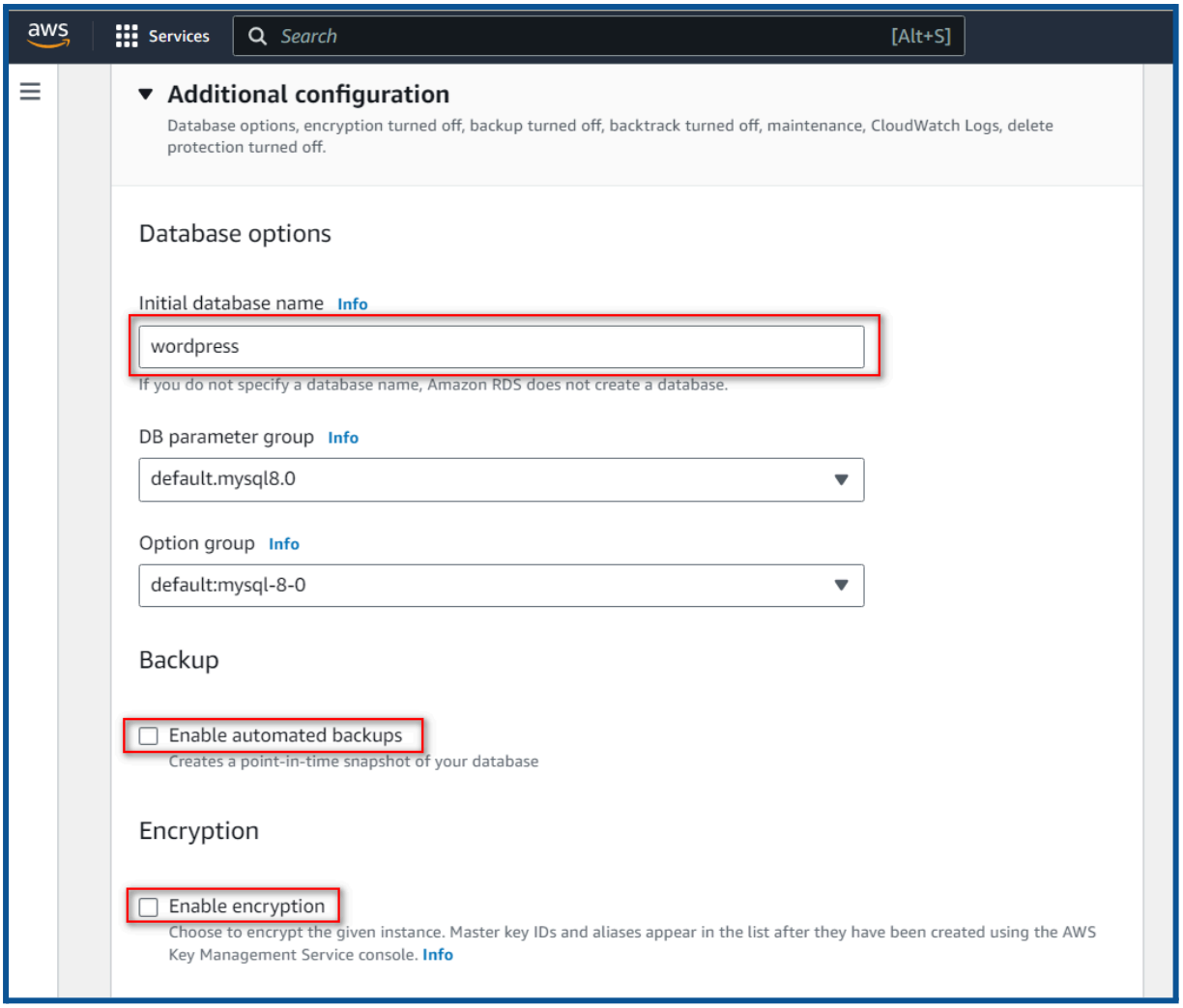


Figure 2.17: Additional Configuration

Awesome! We've reached the final step. Click 'Create database' to bring our database go live (*Figure 2.18: Create Database Final Step)*

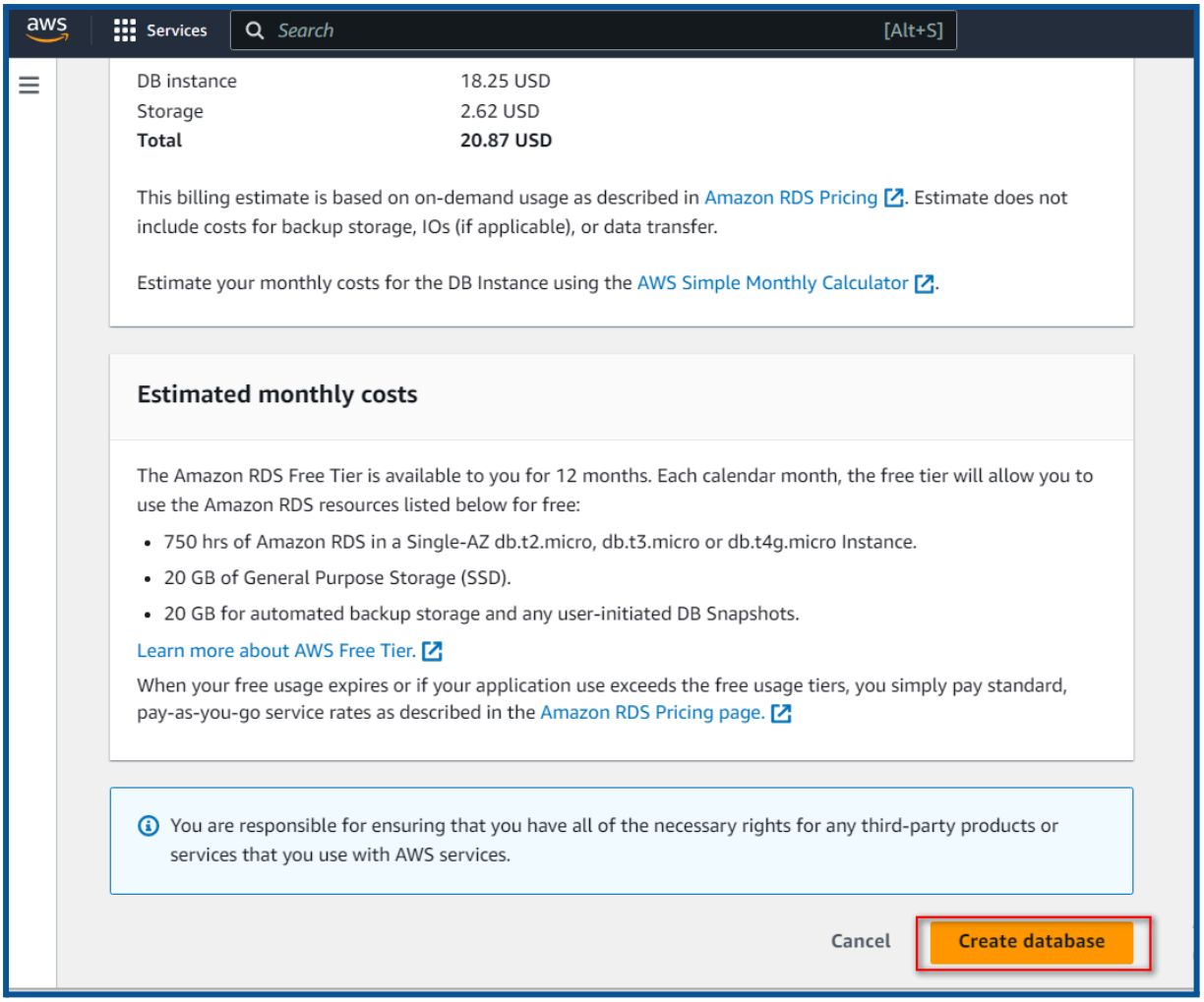


Figure 2.18: Create Database Final Step

Once created, you will see it on the RDS dashboard as shown in the below figure(*Figure 2.19: RDS Dashboard).*

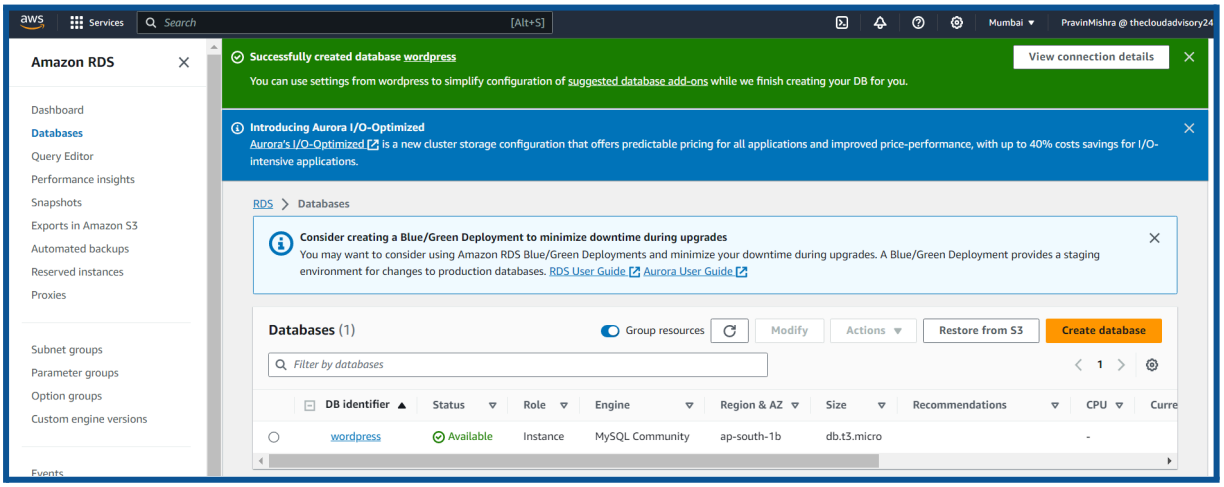


Figure 2.19: RDS Dashboard

✅**Expected Output:** Successful creation of the database instance, visible on the RDS dashboard.

📓 **Note:** Review the RDS dashboard to verify the database instance's successful creation.

STEP 6: Security Groups Settings

1. **Visit Security Groups Page**(*Figure 2.20: Security Group EpicReads):*
    - Access the Security Groups page and choose "public-instance-sg."

a. Edit Inbound Rules:

- Click on the "Edit inbound rules" button.

b. Add MySQL/Aurora Rule:

- Click "Add rule."
- Choose "MYSQL/Aurora" for Type.
- Select "Custom" for Source and locate "db-sg," then save the rules.

c. Add HTTP Rule:

- Click "Add rule" again.
- Select "HTTP" for Type.
- Choose "My IP" for Source, and save the rules(*Figure 2.17: Security Group EpicReads).*

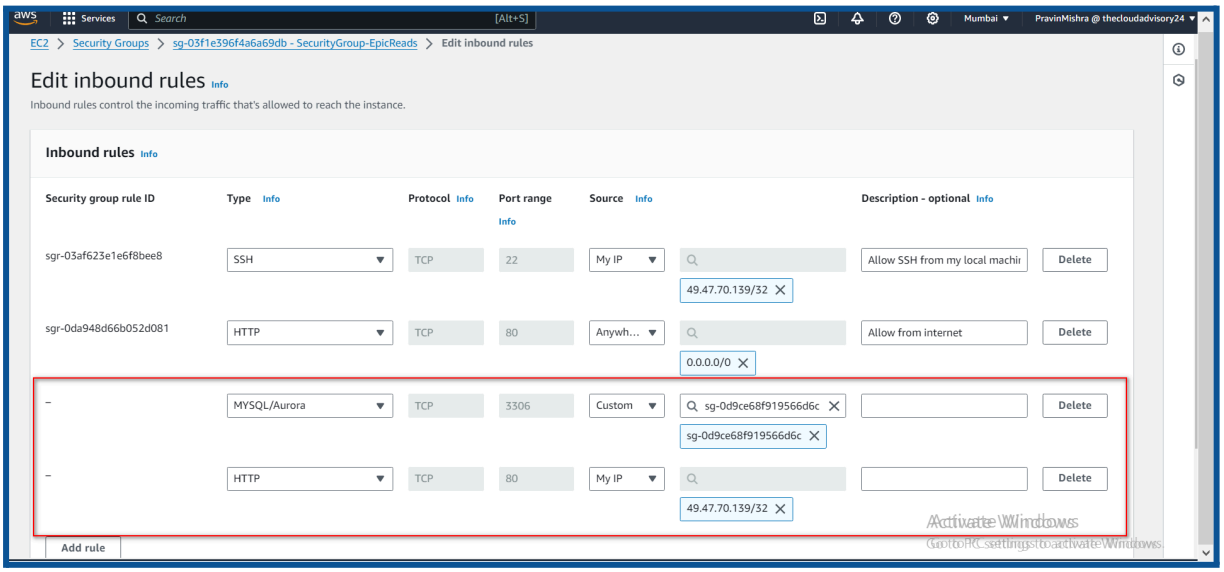


Figure 2.20: Security Group EpicReads

Inbound Rules of Instance Security Group: These rules often include allowing specific types of incoming traffic like HTTP (port 80) or SSH (port 22) from specific IP addresses or ranges. In the above steps, we added rules for MySQL/Aurora (port 3306) to allow communication between the

EC2 instance and the database server, as well as an HTTP rule to allow web traffic.

2. Access Security Groups of DB*(Figure 2.21: Security Group DB):*

- Navigate to the Security Groups page and choose "db-sg."

a. Edit Inbound Rules:

- Click on the "Edit inbound rules" button.

b. Delete Existing Rule:

- Remove any existing rule.

c. Add MYSQL/Aurora Rule:

- Click "Add rule."
- Choose "MYSQL/Aurora" for Type.
- Select "Custom" for Source and locate "SecurityGroup-EpicReads," then save the rules(*Figure 2.18: Security Group DB).*

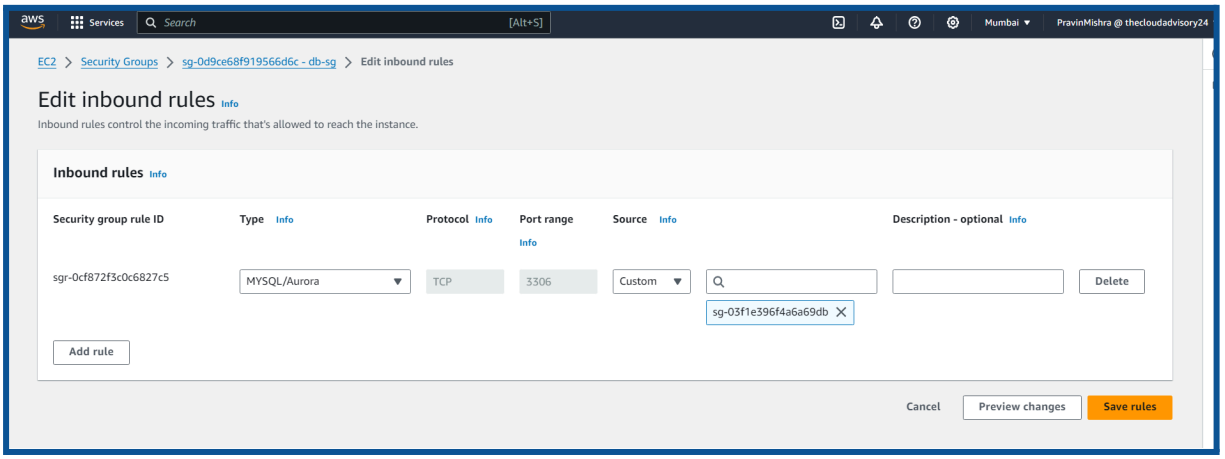


Figure 2.21: Security Group DB

Inbound Rules of DB Security Group: These rules define the source of incoming traffic allowed to access the database server. In the example, we allowed MySQL/Aurora traffic (port 3306) from the "public-instance-sg" to permit the EC2 instance to communicate with the database.

📓 **Note:** Correctly configuring security groups ensures proper network access between resources and enhances security.

ℹ️ **Learn More:** Explore AWS documentation on Security Groups for detailed information.

STEP 7: Configure the WordPress installation and connect it to a database

To configure the WordPress installation and connect it to a database, follow these steps(*Figure 2.22: DB Connection page);*

1. Set MySQL Environment Variable:
   Replace `<your-endpoint>` with your RDS endpoint. Run the following command in your terminal.

   💻 **Command:**

   ```
   export MYSQL_HOST=<your-endpoint>
   ```

   📓 **Note:** This command sets the MySQL environment variable to your RDS endpoint.

2. Connect to the WordPress Database:
   Replace `<your-username>` and `<your-password>` with your MySQL username and password. Use the following command to connect to the WordPress database:
   💻 **Command:**

   ```
   mysql --user=<your-username> --password=<your-password> wordpress
   ```

📓 **Note:** Connects you to the WordPress database using your MySQL credentials.

3. Create New Database and Grant Permissions:
   Inside the MySQL prompt, create a new DB for WordPress and grant permissions:
   💻 **Command:**

   ```
   CREATE USER 'wordpress'@'localhost' IDENTIFIED BY 'wordpress-pass';

   GRANT ALL PRIVILEGES ON wordpress.* TO wordpress;
   FLUSH PRIVILEGES;
   ```

   📓 **Note:** Creates a new database and grants necessary permissions to the WordPress user.

4. Exit MySQL Prompt:
   Exit the MySQL prompt:

   💻 **Command:** *EXIT;*

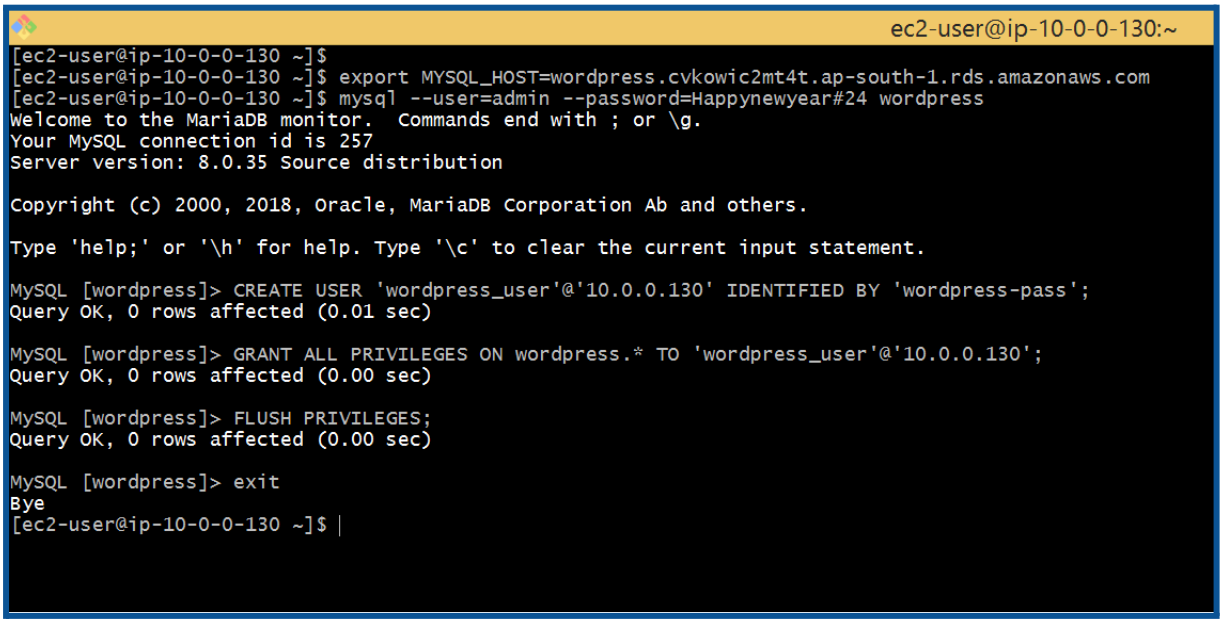


Figure 2.22: DB Connection page

5. Access the wp-config.php file: Use the nano text editor to open the wp-config.php file(*Figure 2.23:* Nano text editor)

   💻 **Command:**

   sudo vi wp-config.php

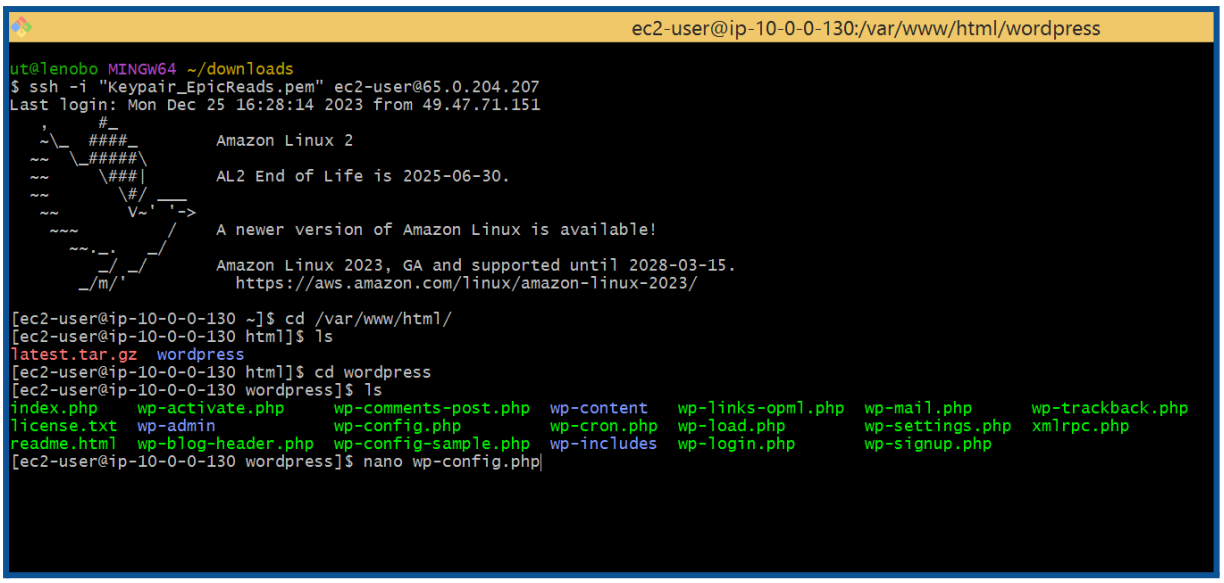


Figure 2.23: Nano text editor

6. Locate and Modify the Database Settings(*Figure 2.24: Nano text editor page*):
   Inside the wp-config.php file, find the section with MySQL settings.
   Replace the following placeholders:

- `database_name_here` with the actual name of your WordPress database ('wordpress' in your case).
- `username_here` with the MySQL database username ('wordpress' in your case).
- `password_here` with the MySQL database password ('wordpress-pass' in your case).
- Replace `localhost` with your `RDS endpoint`.

Here is an example of how it might look after modification:

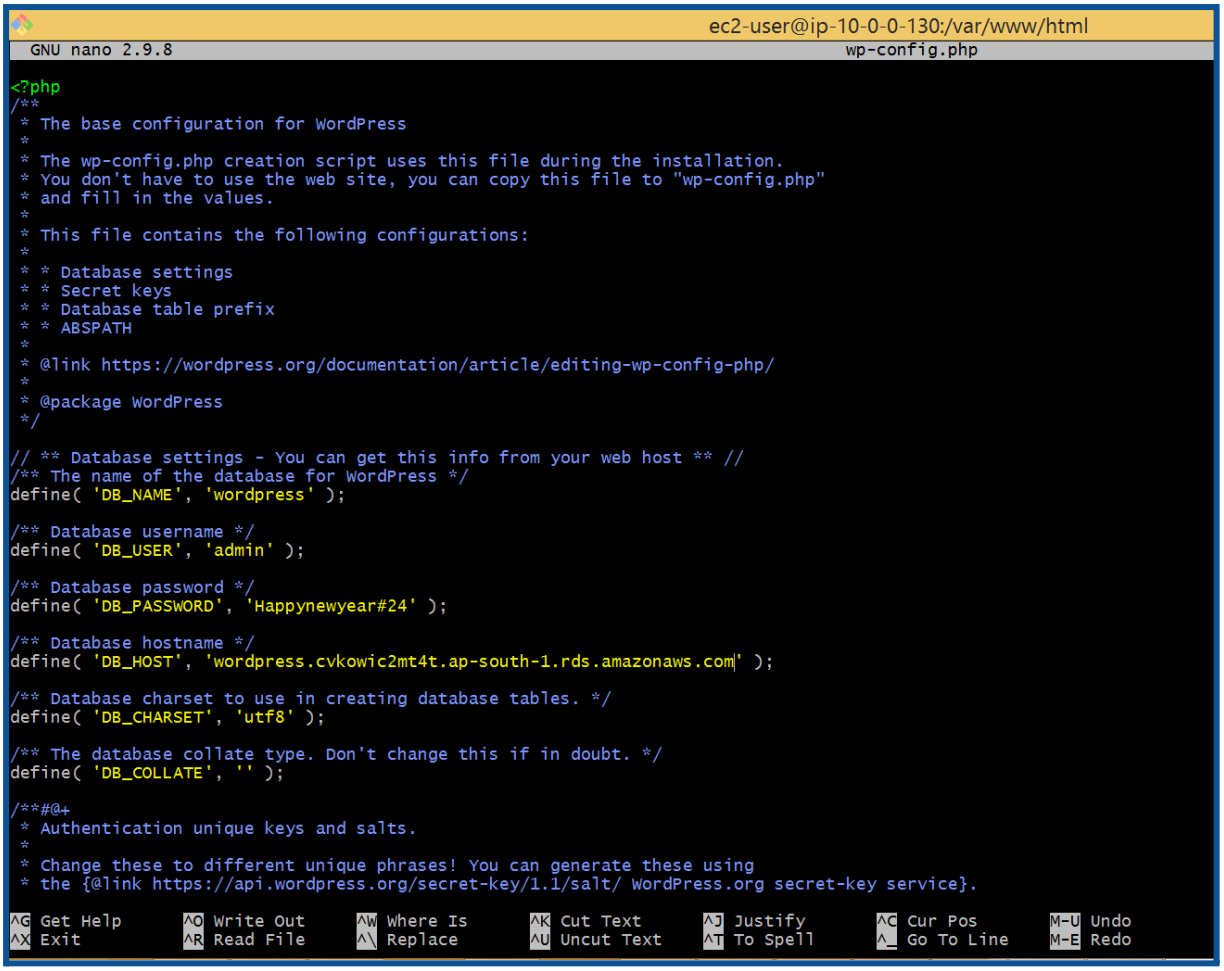


Figure 2.24: Nano text editor page

📓 **Note:** Modify database settings in the `wp-config.php` file. Replace placeholders with actual database information.

7. Navigate to the WordPress Secret Key Generator available at https://api.wordpress.org/secret-key/1.1/salt/ and copy the content generated on the page. Replace the current script labeled as '/**#@+' in your configuration file with the newly acquired values(*Figure 2.25: Nano text editor secret key page)*

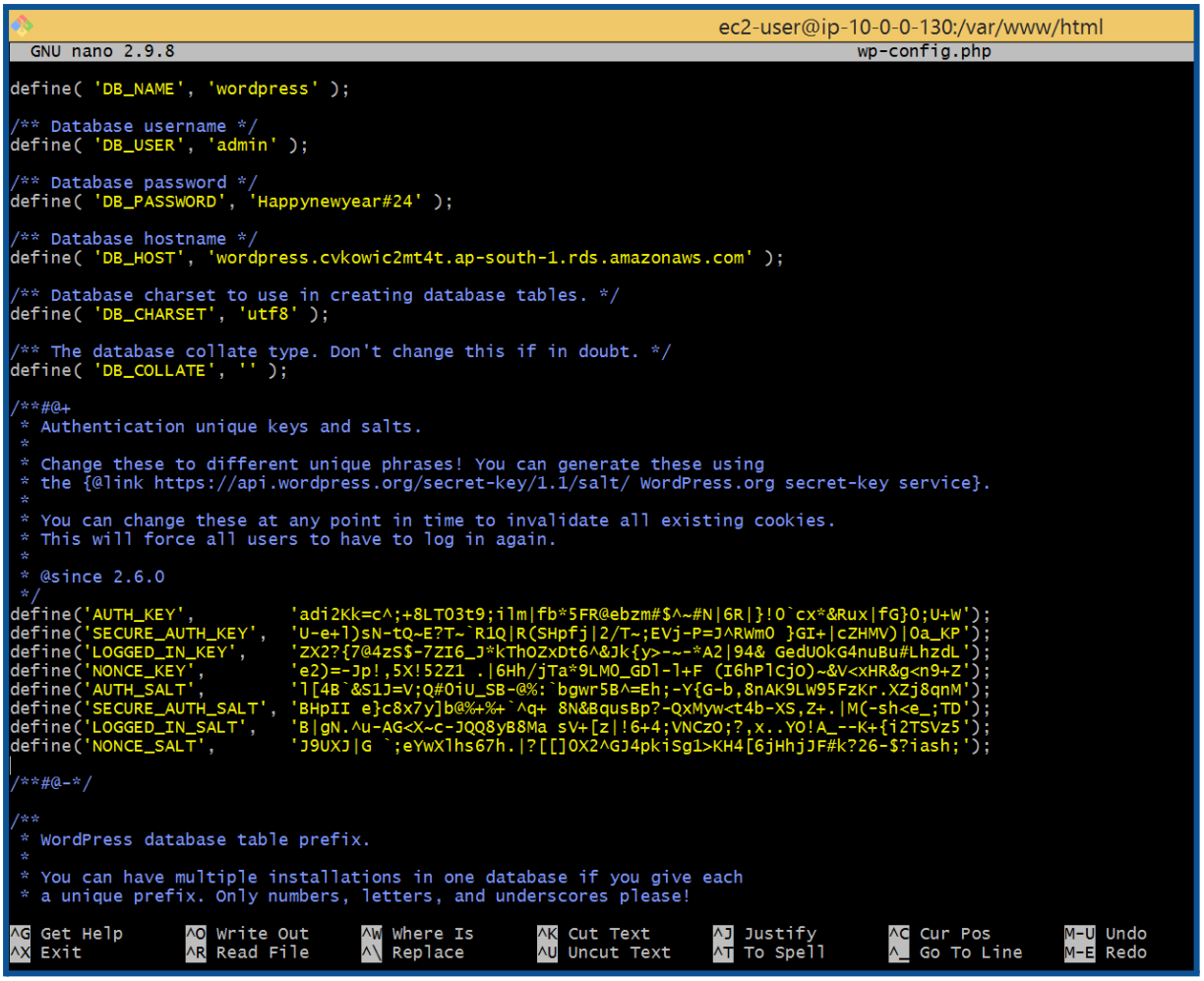


Figure 2.25: Nano text editor secret key page

8. To allow 'W3TC' plugin write the configuration data into DB

   Paste the below command at the end of the script *(Figure 2.26: 'W3TC' plugin config)*

   💻 **Command:**

   ```
   define( 'W3TC_CONFIG_DATABASE', true );
   ```

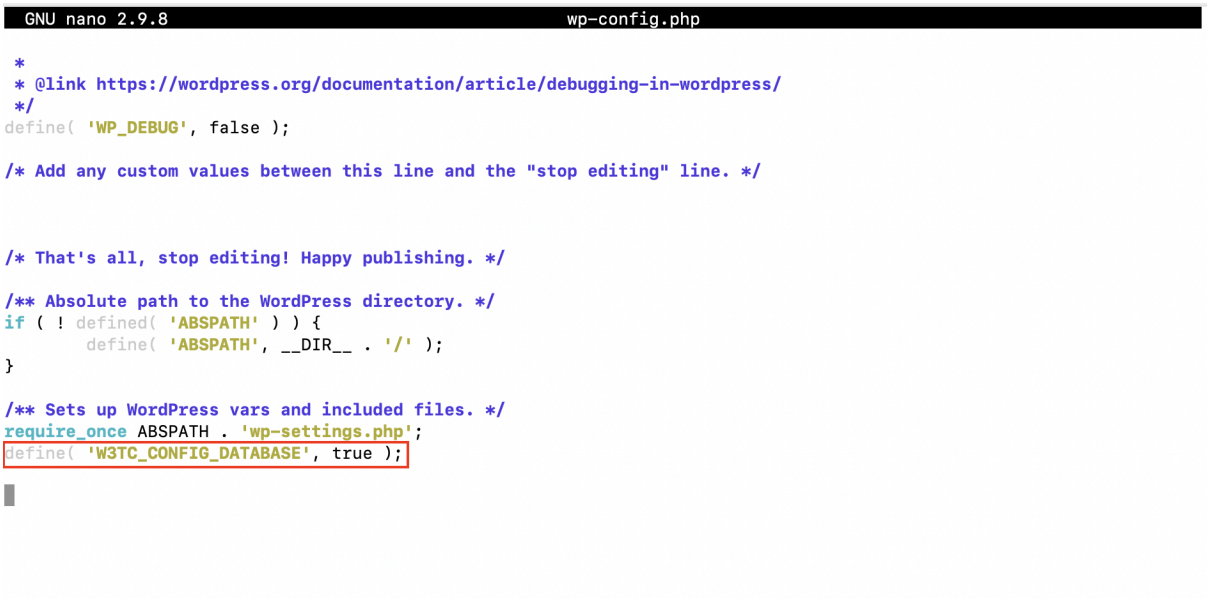


Figure 2.26: 'W3TC' plugin config

9. Save the Changes:
   After making the modifications, press `Ctrl + O` to save the file.
   Press `Enter` to confirm the file name.
   To exit nano, press `Ctrl + X`.

10. Run commands to deploy Wordpress on your computer

```
sudo yum install php-xml -y

cd ..

sudo cp -r wordpress/* /var/www/html

sudo chown -R apache:apache /var/www/html
```

Finally, start hosting the Apache web server

```
sudo systemctl start httpd.service

sudo systemctl enable httpd.service

sudo systemctl restart php-fpm
```

11. **Install WordPress through the Browser:**
    Open a web browser and enter your EC2 instance's **public IP address/wp-admin** or domain name(*Figure 2.27: Wordpress language page).*

    `http://{ec2-instace-public-IP-address}/wp-admin/`

    Follow the WordPress setup wizard:

    a. Select the language and provide database connection details (database name, username, password, and host).

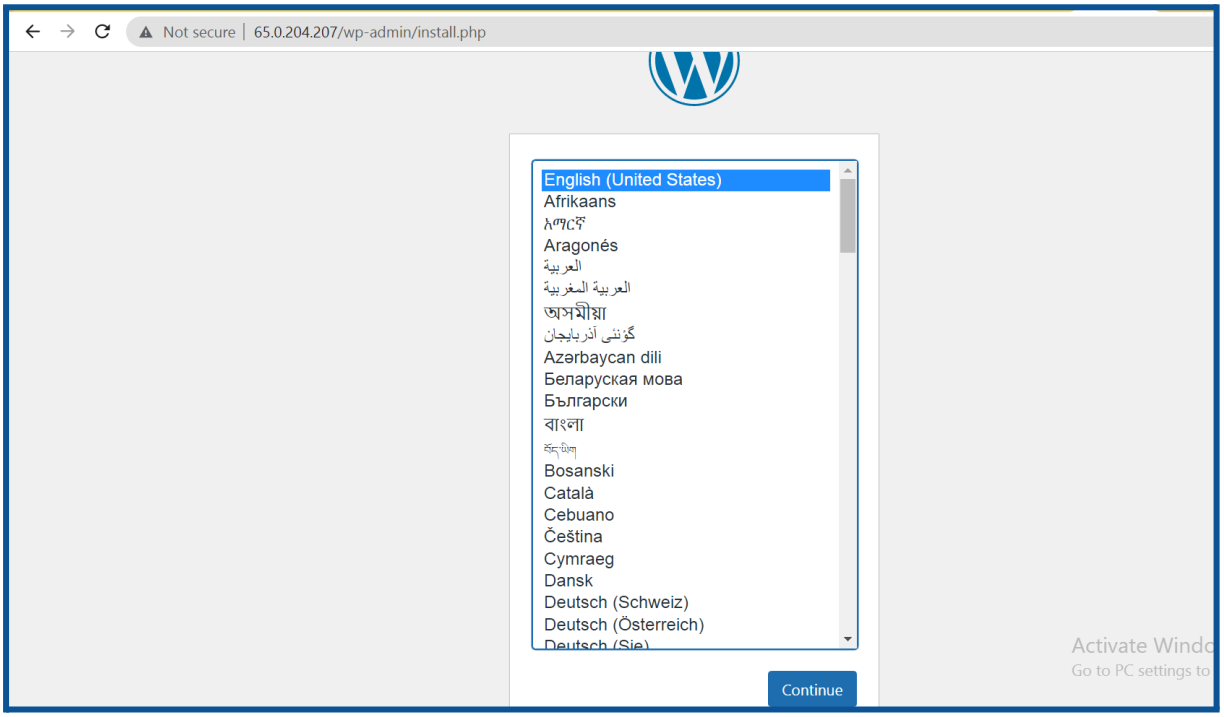


Figure 2.27: Wordpress language page

    b. Complete the WordPress setup by providing site details (site title, admin username, password, email, etc.)(*Figure 2.28: Wordpress site details).*

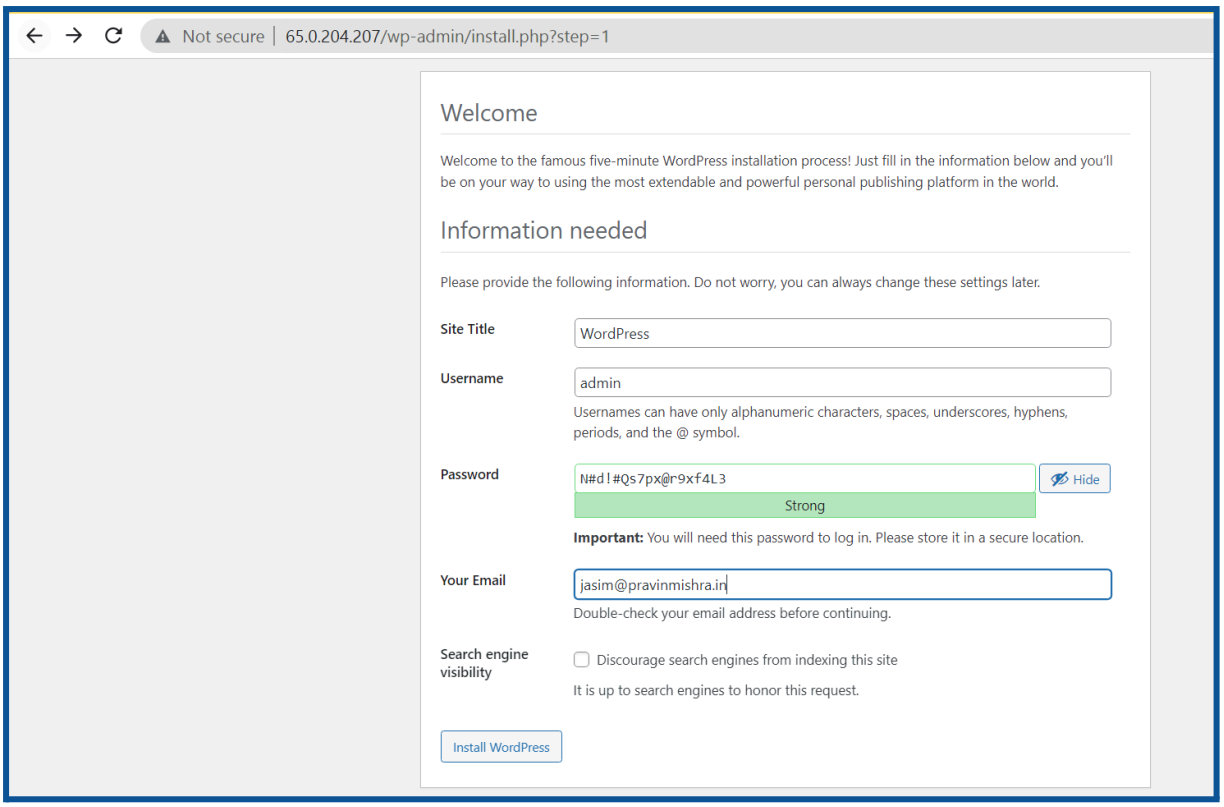


Figure 2.28: Wordpress site details

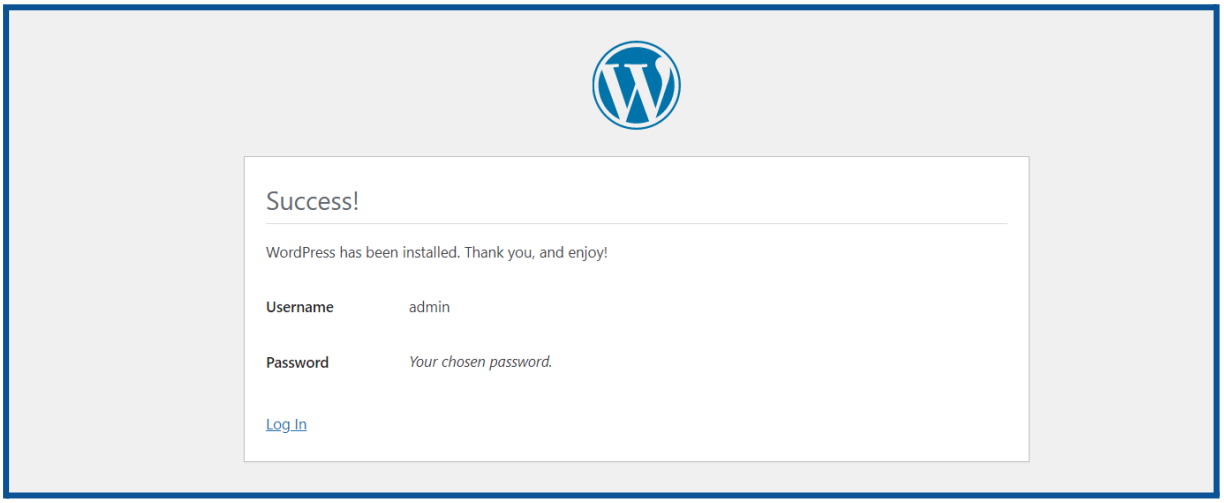


Figure 2.29: Wordpress Success Page

c. Login to the WordPress admin dashboard(*Figure 2.30:* WordPress admin dashboard*)*

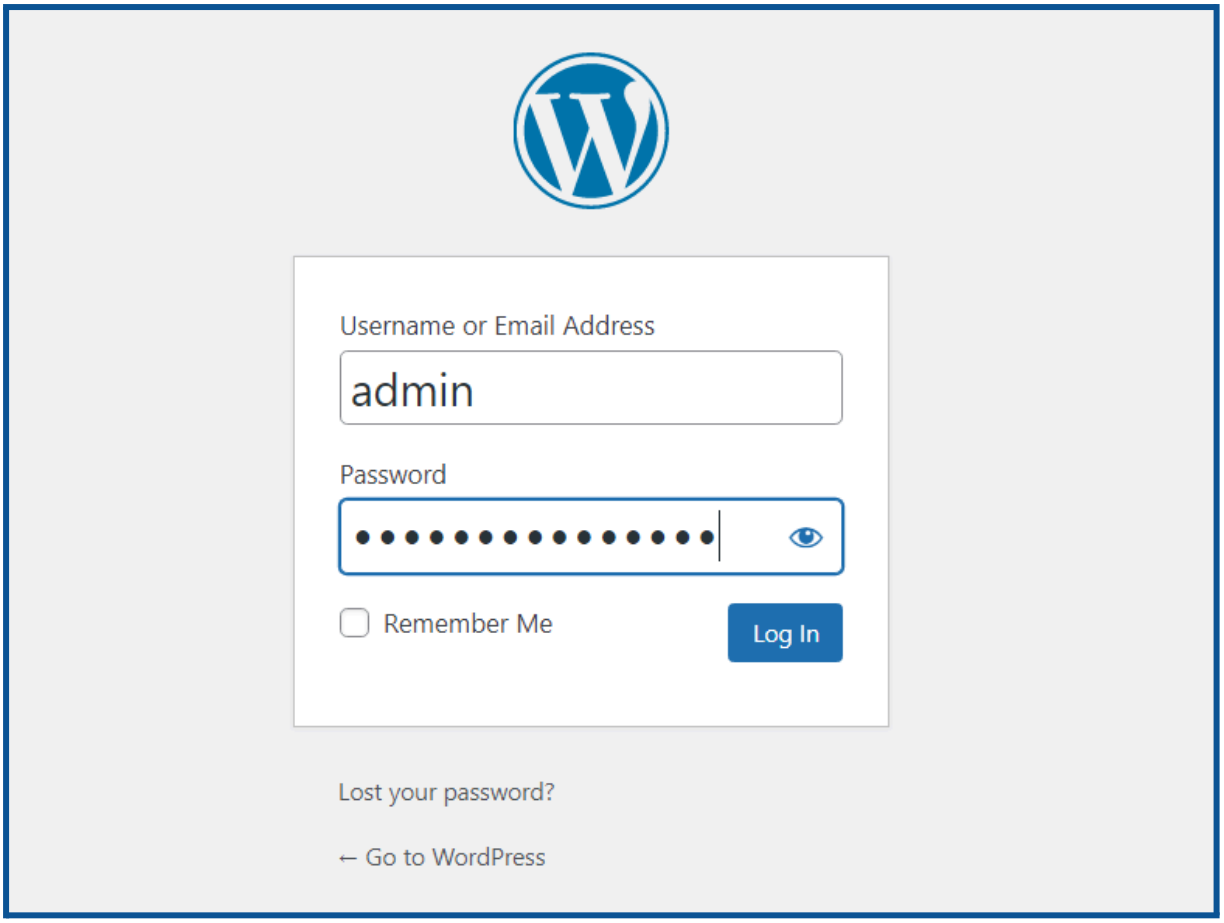


Figure 2.31: WordPress admin dashboard

"**Great job!** You're almost there! This is the final step where you'll access the WordPress app's main page. You've successfully launched, set up, and installed WordPress on your Linux EC2 Instance at EpicReads. Let's take the last exciting step together!" (*Figure 2.32:* WordPress Welcome Page)

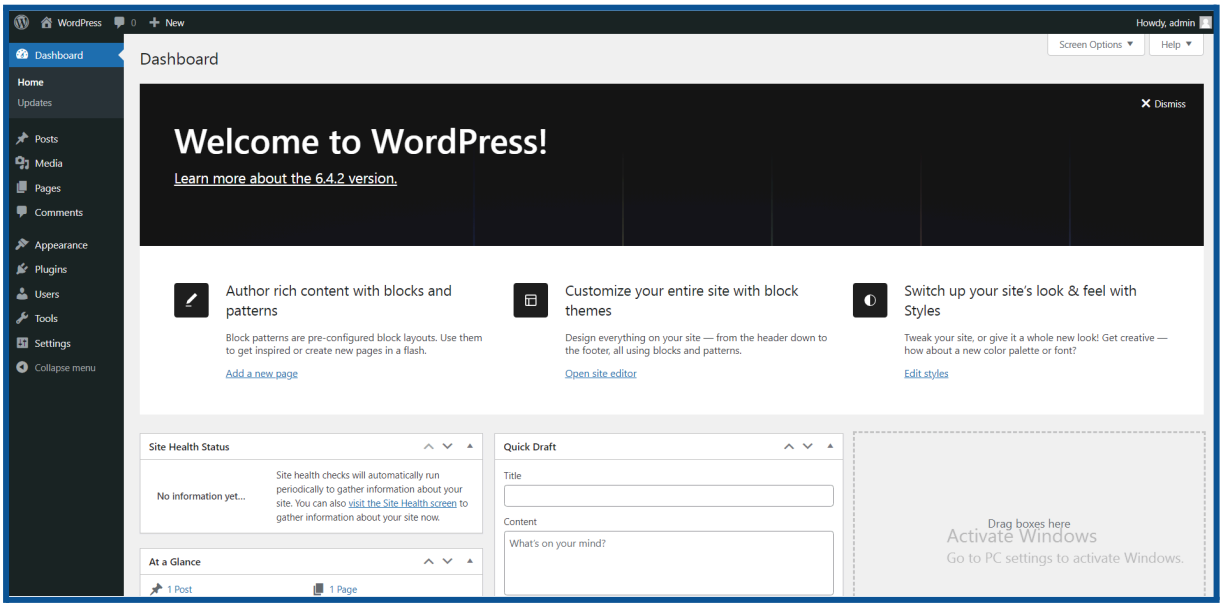


Figure 2.32: WordPress Welcome Page

12. Ensure that the WordPress installation is functioning correctly by creating a test post or page(*Figure 2.33:* Add New Post Page).
Steps to Create a Test Post/Page:

a. **Navigate to Posts or Pages:** In the WordPress Dashboard sidebar, click on "Add New Post" to create a test post.

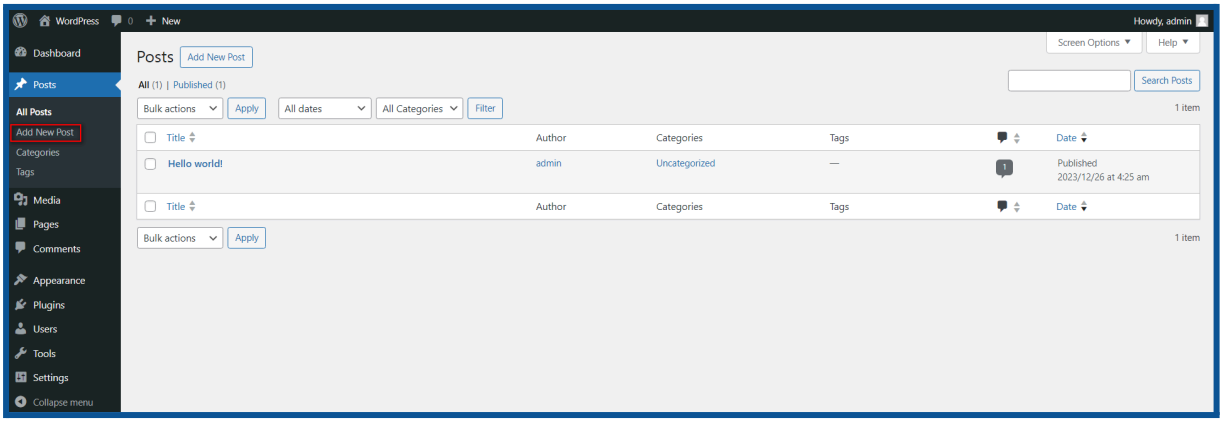


Figure 2.33: Add New Post Page

b. **Fill in Test Post/Page Details:** Enter a title for the test post or page in the designated field(*Figure 2.34:* Post creation Page).

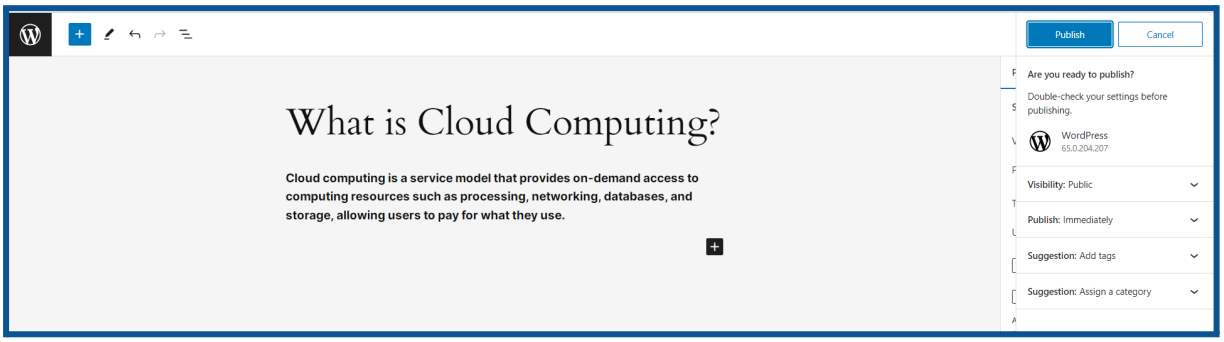


Figure 2.34: Post creation Page

c. **Publish Test Post/Page:** Once satisfied with the test content, click the "Publish" button to make the post/page live on the website.

d. **Verify the Test Post/Page:** Visit the live website and navigate to the newly created test post or page(*Figure 2.35:* Post Sample Page)

Figure 2.35: Post Sample Page

📓 **Note:** Confirm successful WordPress deployment and publication of a test blog on your EC2 Instance at EpicReads.

Kudos! On successfully publishing your latest blog via WordPress on your Linux EC2 Instance at EpicReads!

STEP 8: Domain Configuration (Optional):

This step is the optional step ,if you want to configure a domain name to point to the EC2 instance, providing a user-friendly way to access the WordPress site.

1. **Purchase a Domain Name:** Register a domain name through a domain registrar like **GoDaddy, Namecheap, Google Domains** etc.
2. **Access Domain Registrar's DNS Settings:** Log in to your domain registrar's website and find the DNS (Domain Name System) settings or Domain Management section.
3. **Obtain Public IP of the EC2 Instance:** Go to the AWS Management Console, navigate to EC2, and find your EC2 instance's public IP address or public DNS.
4. **Allocate Elastic IP (Optional, but Recommended):** For a more stable IP address, allocate an Elastic IP address within the AWS EC2 console and associate it with your EC2 instance.
5. **Update DNS Records:** In your domain registrar's DNS settings, add an 'A record' or an 'Alias record' (if supported) that points to the Elastic IP address or the public IP of your EC2 instance.

- For 'A record': Enter the hostname (e.g., www) and the IP address of your EC2 instance.
- For 'Alias record' (if supported): Select the AWS service (like EC2) and choose the instance you want to associate with the domain.

6. **Configure WordPress:** Once the DNS settings propagate (which might take up to 48 hours), update the WordPress settings to reflect the new domain.
   - Log in to the WordPress admin dashboard.
   - Go to Settings > General and update the WordPress Address (URL) and Site Address (URL) to the new domain.
7. **Test Your Domain:** Open a web browser and enter your domain name (e.g., www.yourdomain.com). It should now load the WordPress site hosted on your EC2 instance.

ℹ️ **Learn More:** Explore further details on domain configuration using AWS Managed Domain.

STEP 9: Security Enhancements:

Enhance security measures for the WordPress installation by implementing various methods, including configuring the .htaccess file, establishing correct file permissions, and integrating essential security plugins.

1. **Keep WordPress Updated:** Ensure your WordPress core, themes, and plugins are always up-to-date. Regularly check for updates and apply them promptly to patch any security vulnerabilities.
2. **Use Strong Credentials:** Avoid using default usernames like "admin" and use complex passwords that include a combination of letters, numbers, and special characters. Consider using a password manager to generate and store secure passwords.
3. **Limit Login Attempts:** Implement a plugin (e.g., Limit Login Attempts Reloaded) to restrict the number of login attempts. This prevents brute-force attacks by locking out users or bots after multiple failed login attempts.
4. **Enable Two-Factor Authentication (2FA):** Use a plugin (such as Google Authenticator or Wordfence) to add an extra layer of security by requiring a second form of verification, typically a temporary code sent to your mobile device.

5. **Protect wp-config.php:** Add the following code to your .htaccess file to restrict access to wp-config.php:

   ```
   <files wp-config.php>
   order allow,deny
   deny from all
   </files>
   ```
6. **Secure .htaccess File:** Protect your .htaccess file by ensuring it has the correct permissions (e.g., 644) to prevent unauthorized access. Additionally, regularly check for any unusual modifications.
7. **File Permissions:** Set appropriate file permissions for WordPress files and folders:

- **Directories: 755**
- **Files: 644**
- **wp-config.php: 600**

8. **Install Security Plugins:** Consider using security plugins such as Wordfence Security, Sucuri Security, or iThemes Security. These plugins offer features like firewall protection, malware scanning, and real-time threat detection.
9. **Enable HTTPS:** Encrypt data transmitted between users and your website by installing an SSL certificate. This can be done through your hosting provider or using services like Let's Encrypt. Update your WordPress settings to use HTTPS.
10. **Disable Directory Listing:** Prevent directory browsing by adding this line to your .htaccess file:
    ```
    Options -Indexes
    ```
11. **Regular Backups:** Set up regular backups of your WordPress site (files and database) using plugins or your hosting provider's backup solutions. This ensures you can restore your site if needed.

Task 3.1: Secure Access to EC2 Instance Using AWS Systems Manager Session Manager

Project Overview:

This project focuses on setting up and utilizing AWS Systems Manager Session Manager to securely manage and access Linux EC2 instances without exposing SSH ports to the internet. It aims to create a safer method of connecting to these instances while avoiding the traditional practice of opening SSH ports for external access.

Project Objectives:

- The primary goal is to establish a secure means of accessing and managing EC2 instances hosted on AWS using Systems Manager Session Manager.
- No ports are needed to be allowed in security groups.
- There is no need for SSH keys.
- You can delegate access to manage EC2 instances using IAM roles.

Usage of Systems Manager Session Manager:

- Systems Manager Session Manager acts as a secure gateway to access EC2 instances. It offers controlled and monitored access without directly opening SSH ports to make it more secure.

This guide covers the following:

- Prepare the EC2 instance for access through Session Manager by ensuring necessary prerequisites are met.
- Use Session Manager for secure and auditable instance management

STEP 1: Systems Manager Prerequisites:

IAM Role Setup: Ensure that the EC2 instance has an IAM role associated with the `AmazonSSMManagedInstanceCore` policy or a custom policy granting necessary permissions for AWS Systems Manager.

1. **Access the AWS Management Console:**
    - Open your web browser and go to the AWS Management Console by clicking *here*

2. **Navigate to the IAM Service:**
    - Click on "Services" in the top left corner.
    - Under "Security, Identity, & Compliance", select "IAM" to access the IAM Dashboard.
3. **Select "Roles" from the Left Navigation Pane:** (*Figure 3.1:* IAM Dashboard)
    - In the IAM Dashboard, click on "Roles" in the left-hand side panel.

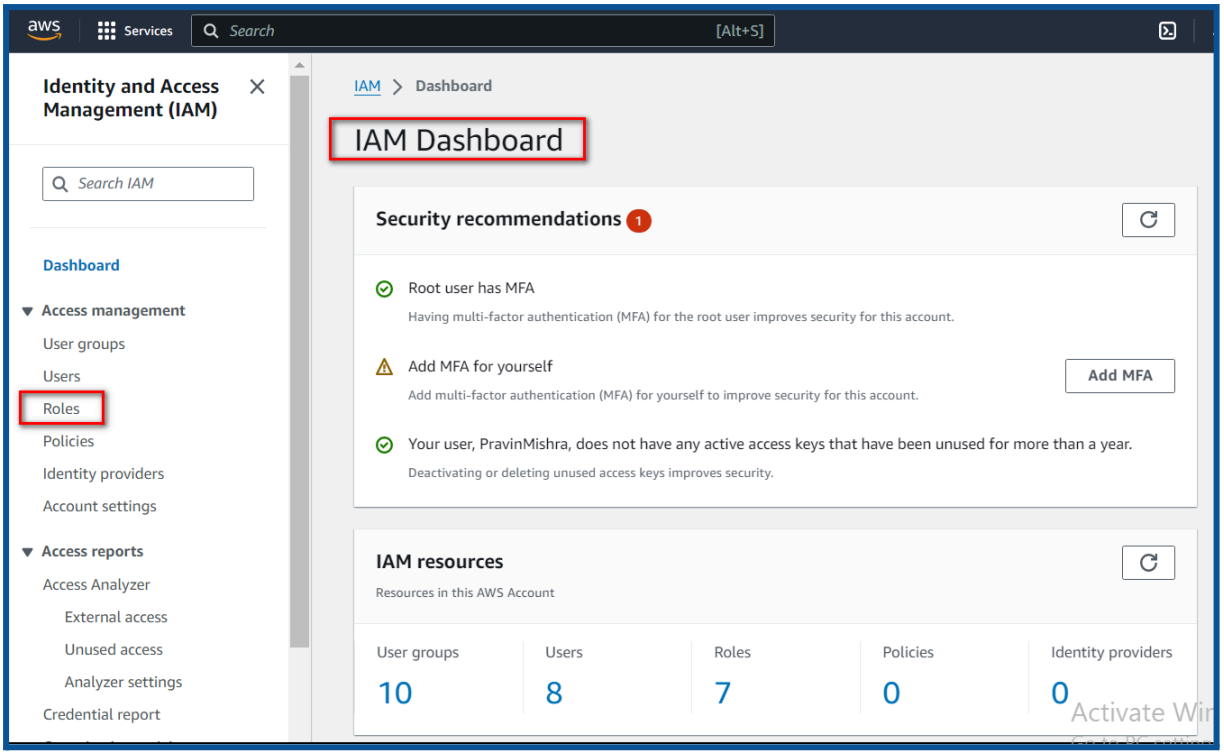


Figure 3.1: IAM Dashboard

4. **Create a New Role:** *(Figure 3.2: Create Role Tab)*
    - Click on the "Create role" button.

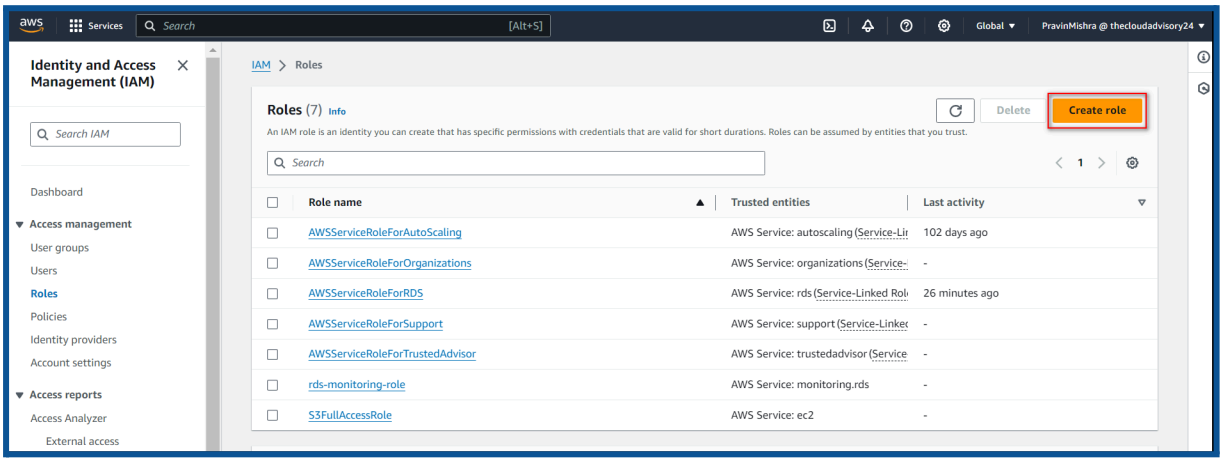


Figure 3.2: Create Role Tab

5. **Choose the Trusted Entity Type:** (*Figure 3.2: Trusted Entity Type)*
    - Select the service that will use this role. In this case, as it's for an EC2 instance, choose "AWS service" as the trusted

entity and then select "EC2" from the list of use cases and click on the *Next* button

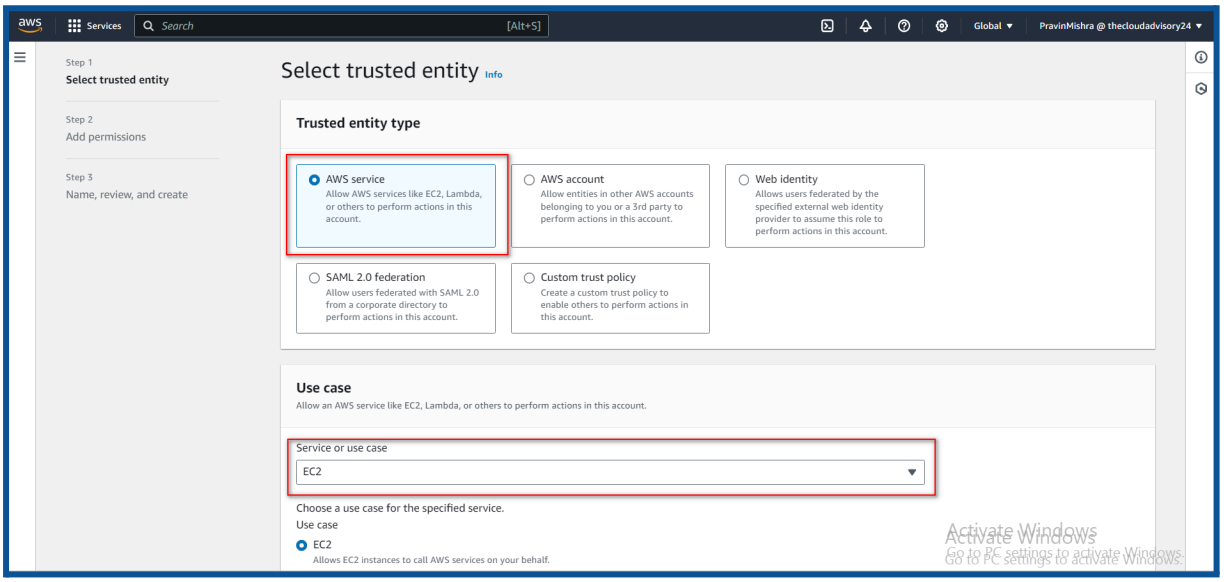


Figure 3.2: Trusted Entity Type

6. **Attach Permissions Policies:** *(Figure 3.3: Add Permissions Page)*
    - In the "**Permissions**" section, you can attach policies to the role. Here, you can either attach existing policies or create a custom policy to grant Systems Manager access.
    - To attach the `AmazonSSMManagedInstanceCore` policy, click on "**Attach policies directly**" and search for `AmazonSSMManagedInstanceCore`. Select the checkbox next to it.

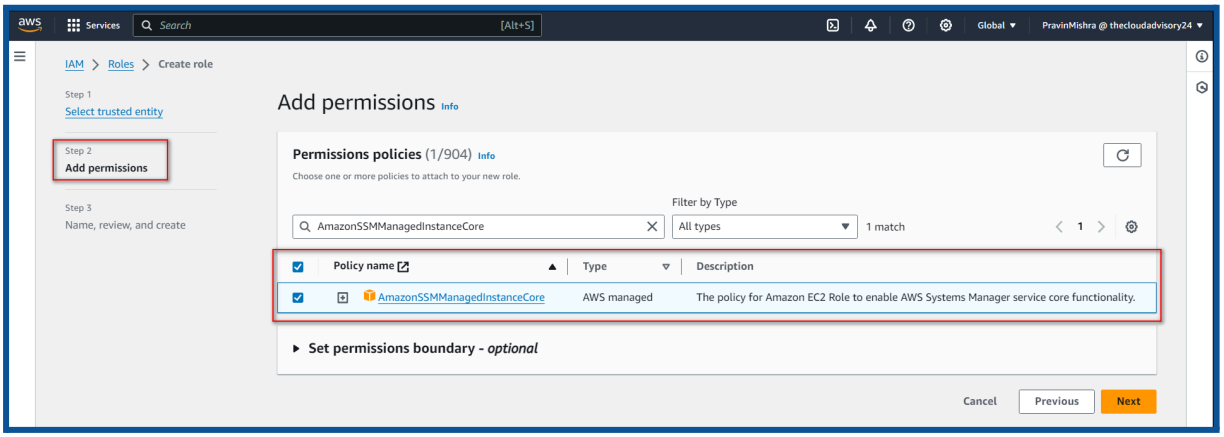


Figure 3.3: Add Permissions Page

✅ **Expected Output:** After selecting the **AmazonSSMManagedInstanceCore policy**, it should be successfully

attached to the role, enabling Systems Manager access for the EC2 instance.

💡 **Consider:** Consider reviewing the permissions granted by the attached policies to ensure they align with the requirements of Systems Manager and the EC2 instance's operational needs.

ℹ️ **Learn More:** Explore more about AWS managed policies and creating custom policies to suit specific access requirements.

7. **Add Tags (Optional):**
    - Optionally, you can add tags to the role for better organization and management. Tags are key-value pairs.

8. **Review and Name the Role:** *(Figure 3.4: Name, review Page)*
    - Give the role a meaningful name that describes its purpose, such as "EC2-SSM-Access-Role-EpicReads".
    - Optionally, add a description to clarify the role's usage.

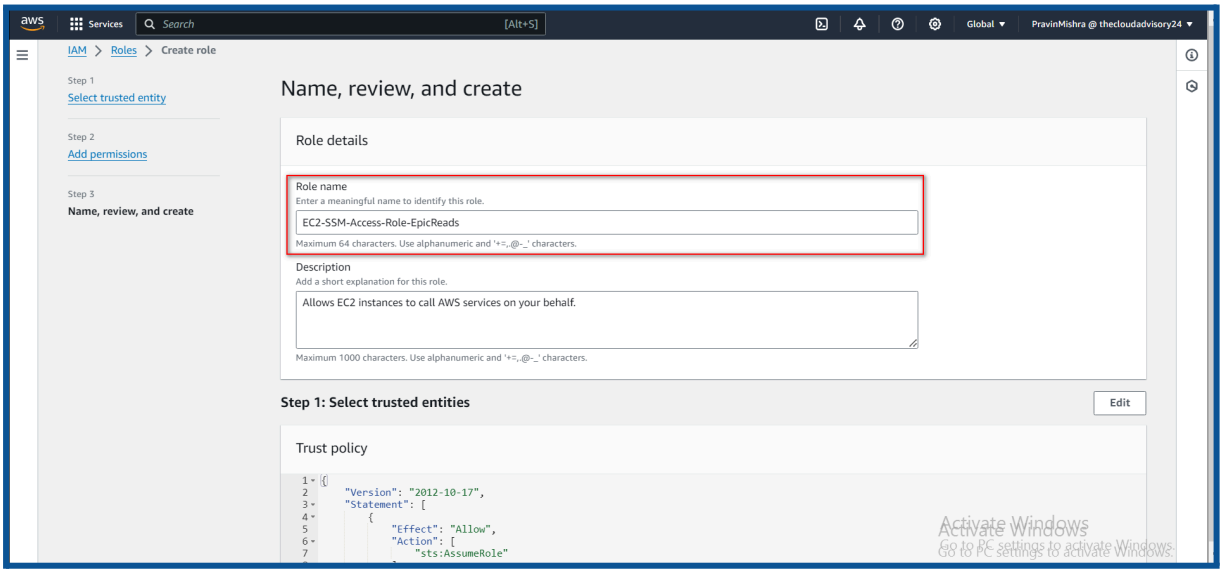


Figure 3.4: Name, review Page

9. **Create the Role:** (*Figure 3.5: Create Role)*
    - Click on the "Create role" button at the bottom of the page.

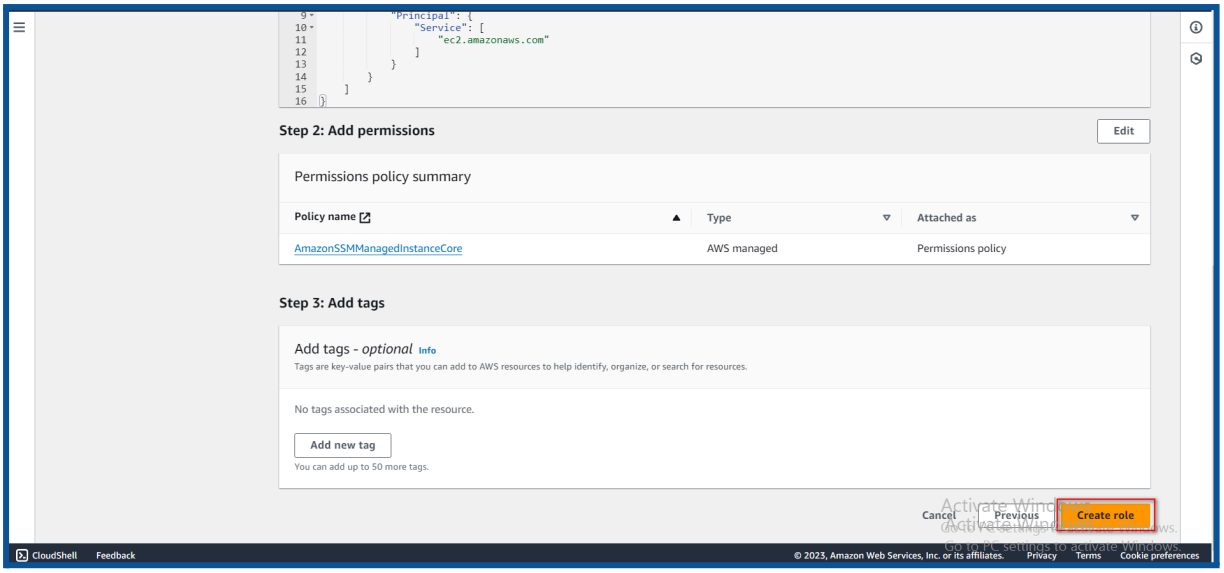


Figure 3.5: Create Role

10. **Role Creation Confirmation:** (*Figure 3.5: Role Creation Confirmation Page)*
    - You will be redirected to the summary page for the newly created role, showing details including the role ARN and policies attached.

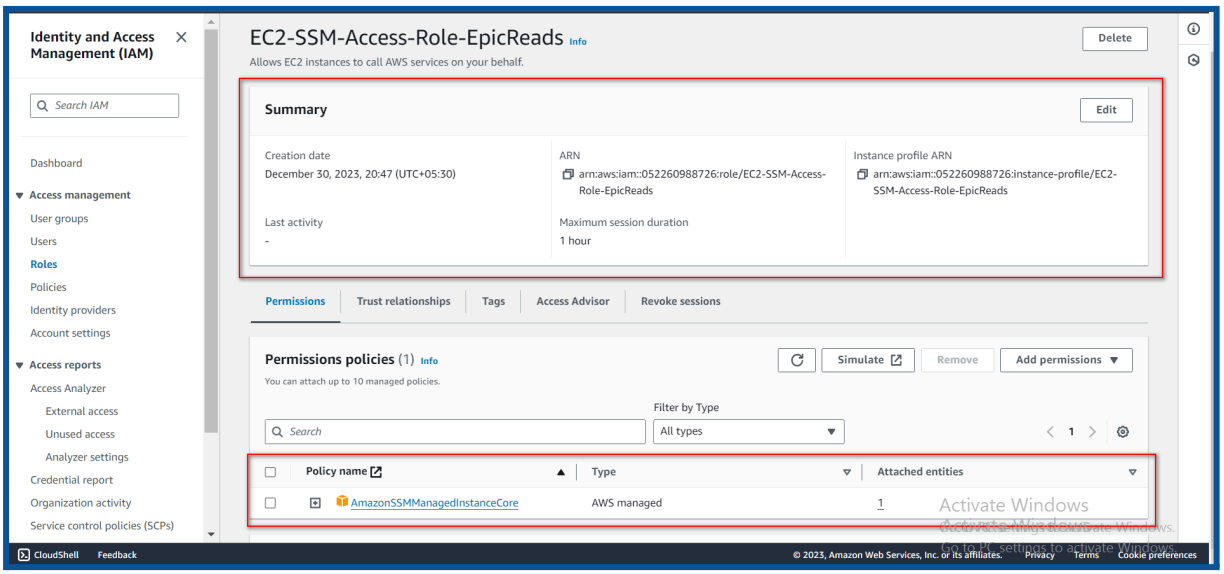


Figure 3.5: Role Creation Confirmation Page

✅**Expected Output:** Upon successful role creation, you should see a confirmation page displaying details like the **role ARN and attached policies.**

💡 **Consider:** Reflect on the purpose of the role you're creating and consider naming it based on its function or purpose within your

infrastructure. Additionally, think about incorporating specific tags for better organization and management of roles.

STEP 2 : Attach the IAM Role to the EC2 Instance:

1. **Go to the EC2 Dashboard:**
    - In the **EC2 Dashboard**, select the instance.
2. **Attach IAM Role via Instance Settings:** (*Figure 3.6:* Action Tab)
    - Under "**Actions**" dropdown, select "**Security**" -> "**Modify IAM role**".

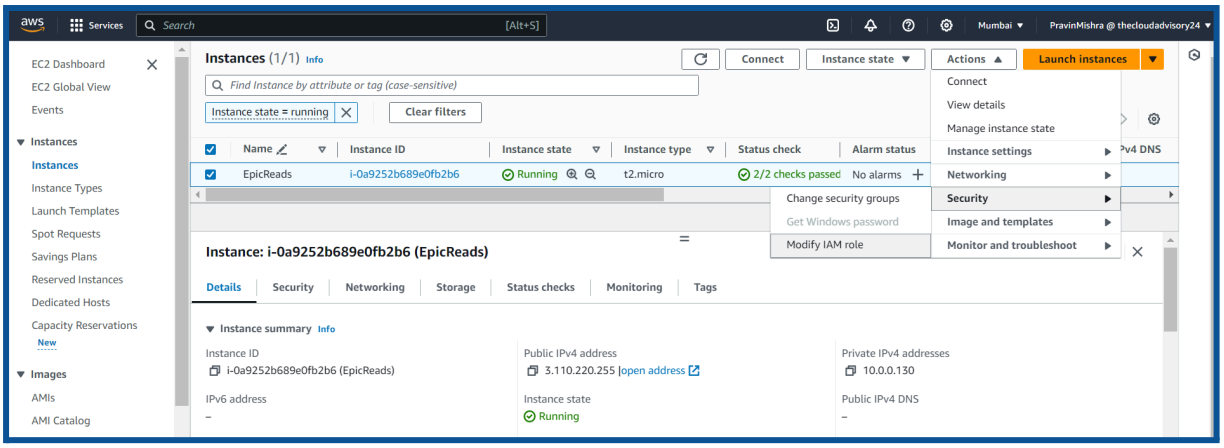


Figure 3.6: Action Tab

3. **Choose the IAM Role:** (*Figure 3.7:* Modify IAM Role)
    - In the "**Modify IAM role**" dialog box, you'll see a list of available IAM roles.
    - Choose the IAM role you created or updated from the drop-down menu and save the changes.

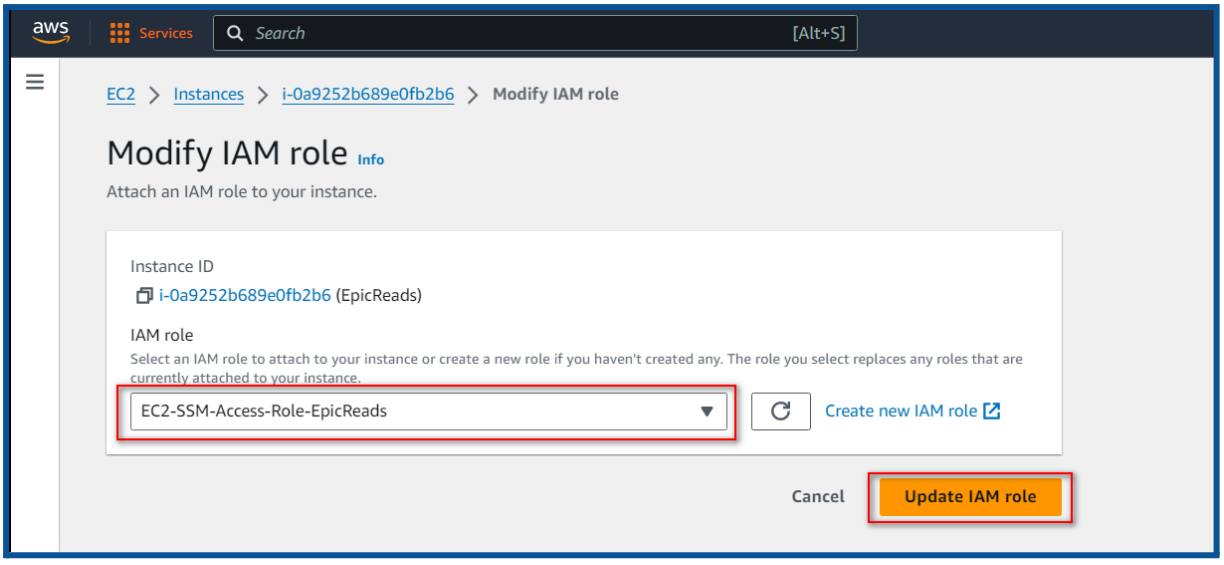


Figure 3.7: Modify IAM Role

🔥 **WARNING:** Assigning incorrect permissions or missing this step could lead to failures in utilizing Systems Manager effectively or causing unintended changes to your system.

4. **Confirm Role Attachment:** (*Figure 3.8:* IAM Role Confirmation)
    - confirm the role attachment by checking the "Description" tab of the EC2 instance details. You should see the IAM role listed under "IAM role".

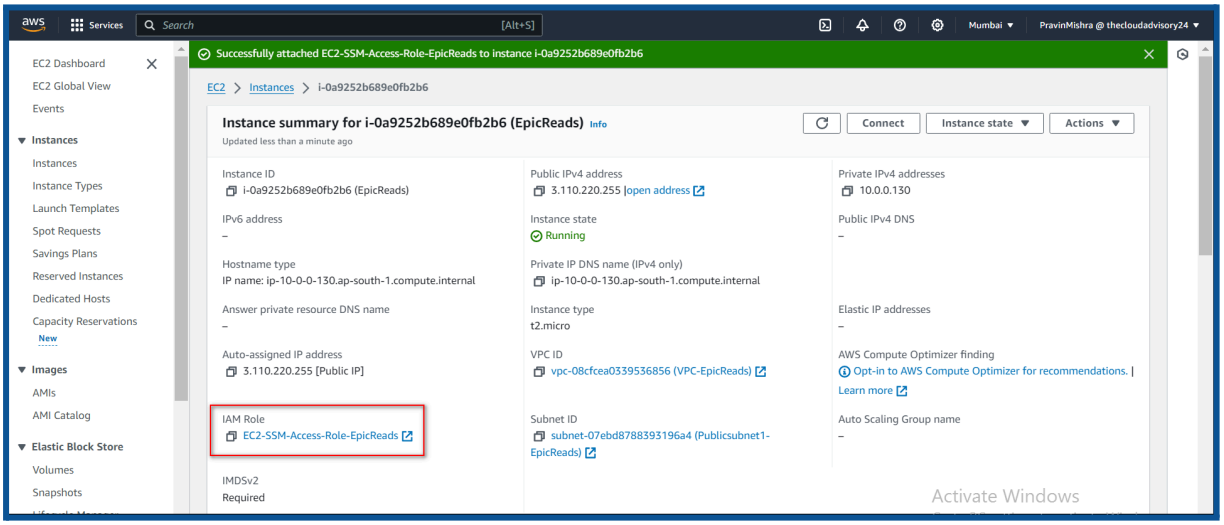


Figure 3.8: IAM Role Confirmation

📓 **Note:** Ensure the EC2 instance has an IAM role associated with the **AmazonSSMManagedInstanceCore policy** or a custom policy granting **necessary permissions** for AWS Systems Manager.

ℹ️ **Learn More**: For more detailed information on IAM roles and permissions, refer to the [AWS documentation on IAM roles].

Final Steps:

After assigning the IAM role to your EC2 instance. Proceed with Installing the SSM Agent on your EC2 instance if it's not already installed.

STEP 3 : Check SSM Agent Installation

1. **Connect to the EC2 Instance Using SSH:**
    - Use SSH to connect to your EC2 instance.

2. **Check SSM Agent Status:** (*Figure 3.9: Check SSM Agent Status)*
    - Run the following command to verify if the SSM Agent is installed and running:

    💻 **Command:** sudo systemctl status amazon-ssm-agent

    If the **SSM Agent** is installed and running, the output will indicate its status as "**active (running)**".

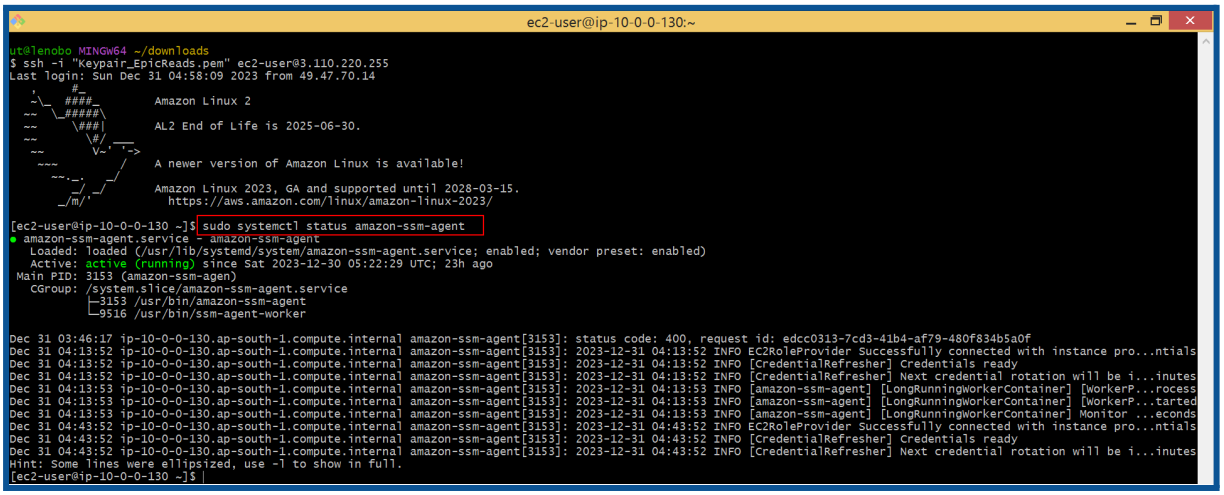


Figure 3.9: Check SSM Agent Status

✅**Expected Output:** Upon running the '**sudo systemctl status amazon-ssm-agent**' command, an "**active (running)**" status indicates the successful installation and running state of the SSM Agent.

📓 **Note:** SSH into the EC2 instance to perform checks on the SSM Agent status.

STEP 4: Install SSM Agent (if not installed):

1. **Download and Install the SSM Agent:**
    - Run the following commands to download and install the SSM Agent (if it's not already installed):

    💻 **Command:** sudo yum install -y https://s3.amazonaws.com/ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm

2. **Start the SSM Agent:**
    - After installing the SSM Agent, start the service by executing:

💻 **Command:** sudo systemctl start amazon-ssm-agent

- Run the following command to verify if the SSM Agent is installed and running:

💻 **Command:** sudo systemctl status amazon-ssm-agent

✅ **Expected Output:** Upon successful execution of 'sudo yum install...' and 'sudo systemctl start...' commands, '**sudo systemctl status amazon-ssm-agent**' should display an "**active (running)**" status, confirming SSM Agent installation and running status.

📓 **Note:** These commands assist in downloading, installing, and starting the SSM Agent, enabling secure connections to your EC2 instance using **Session Manager**.

ℹ️ **Learn More:** For detailed information on the installation process or troubleshooting, refer to AWS documentation on SSM Agent installation.

Congratulations on completing all the necessary steps! You're now ready to securely connect to your EC2 instance using Session Manager.

Note: You have to wait for sometime (usually 10 -15 minutes)until the configure completes!.
Do not panic if you get to see (*Figure 3.10: Error*)

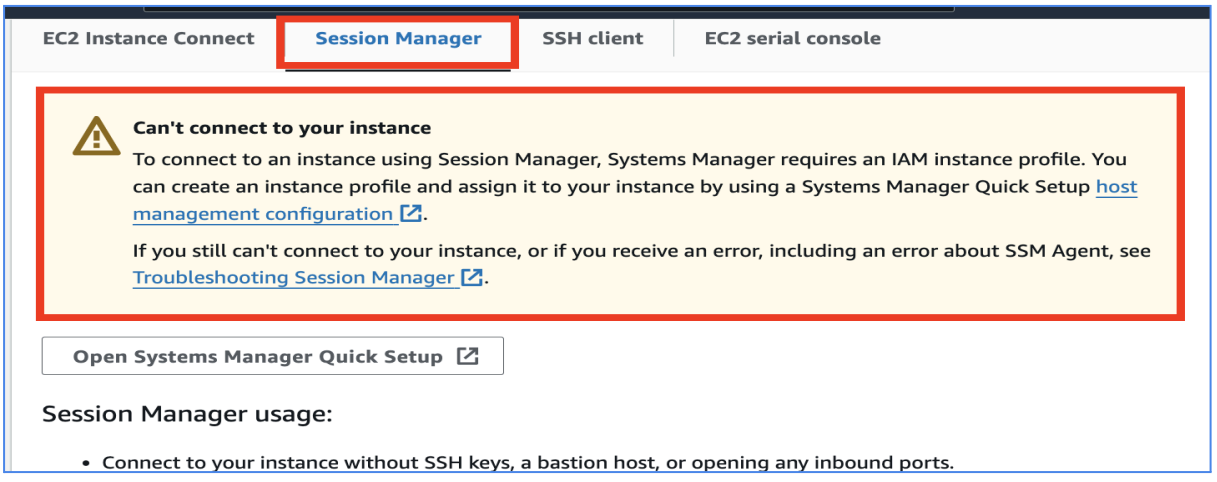


Figure 3.10: Error

Step 5: Connect to EC2 Instance using Session Manager:

1. **Select the EC2 Instance:**
    - From the list of instances, select the EC2 instance of EpicReads.

2. **Choose "Connect" Tab :** (*Figure 3.11: EC2 Connect Tab)*
    - Click on the connect tab at the upper right corner of the page.

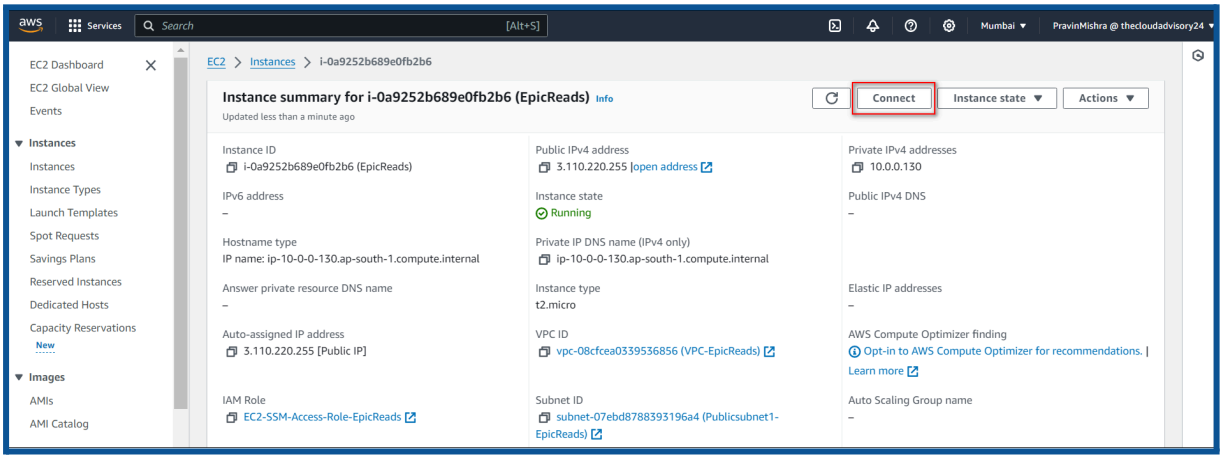


Figure 3.11: EC2 Connect Tab

3. **Connect to Instance**: *(Figure 3.12:connect to Instance)*
    - In the "**Session Manager**" tab, Click on connect to create a session.

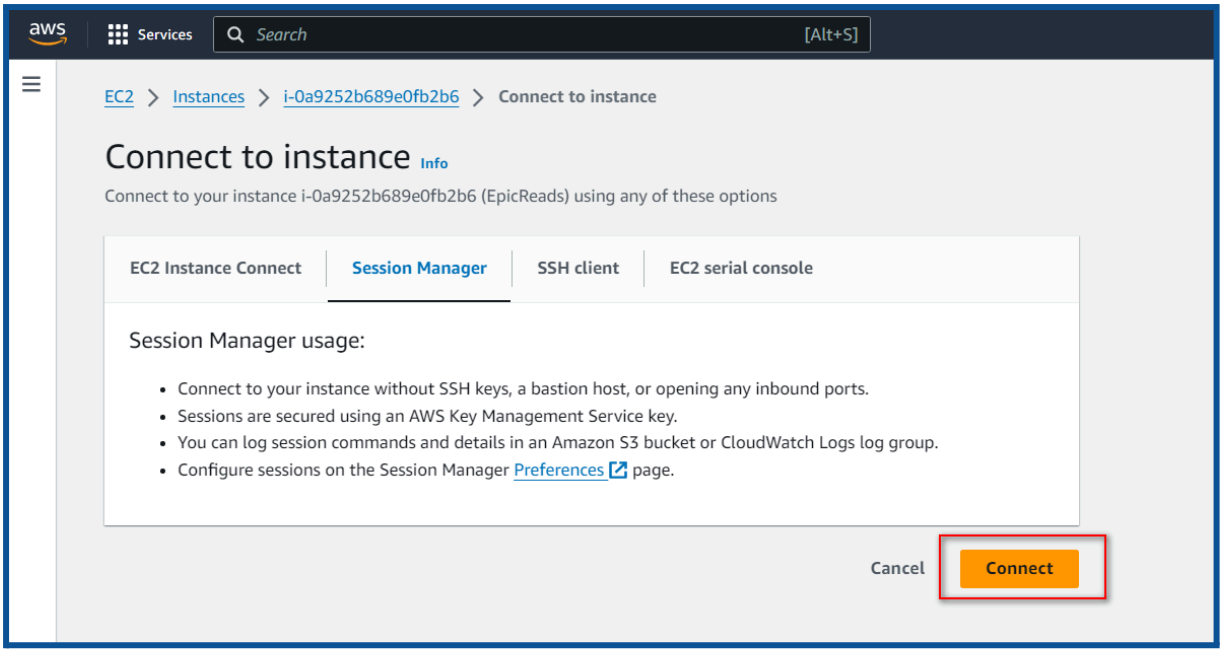


Figure 3.12:connect to Instance

4. **Interact with the EC2 Instance:** (*Figure 3.13: EC2 Instance Session)*
    - Once the session is started, a terminal-like interface will open directly in the browser, allowing you to interact with the EC2 instance's shell.
    - You can execute commands, perform administrative tasks, configure settings, etc., directly through this interface.

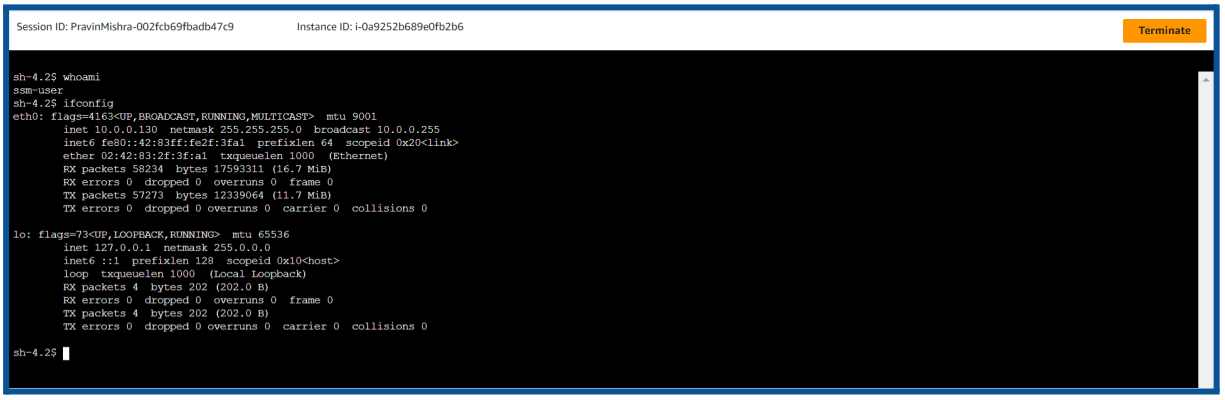


Figure 3.13: EC2 Instance Session

5. **End Session:**

- After you've completed your tasks, click on the "**Terminate session**" button to end the session securely.

✅ **Expected Output:** Upon successful connection, you should see a **terminal-like interface within the browser**, enabling direct interaction with the EC2 instance's shell.

📓 **Note:** Session Manager allows **secure and convenient access to EC2** instances **without requiring inbound traffic or SSH keys**.

🔥 **WARNING:** Leaving sessions open unnecessarily can pose security risks. Always terminate sessions after completing tasks to safeguard your EC2 instance.

ℹ️ **Learn More:** Explore AWS documentation on Session Manager for more detailed information on establishing and managing sessions.

Congratulations! You can now securely access the EC2 Instance of Epic Reads utilizing the robust capabilities of AWS Systems Manager Session Manager.

Step 6: Security Analysis: Benefits of AWS Systems Manager Session Manager Over SSH

1. **Elimination of SSH Port Exposure**:
    - **SSH Vulnerabilities Mitigated:** Session Manager eliminates the need to expose SSH ports on EC2 instances to the internet, significantly reducing the attack surface area. This mitigates common vulnerabilities associated with open SSH ports, such as brute-force attacks and unauthorized access attempts.
2. **Centralized and Controlled Access:**
    - **Centralized Management:** Session Manager provides a centralized access control mechanism through AWS IAM roles and policies. Access to EC2 instances is managed centrally, reducing the risk of unauthorized access by implementing fine-grained access controls.
3. **Integration with AWS Services:**

- **Integrated Security:** Session Manager is an integral part of AWS Systems Manager, leveraging the security features and compliance standards offered by AWS. This integration ensures adherence to best security practices and continuous updates by AWS.

- 📓 **Note: AWS Systems Manager Session Manager** offers a robust and secure alternative to traditional SSH access by **eliminating the exposure of SSH ports** on EC2 instances. Its **centralized access control, encrypted communication, comprehensive logging, and reduced attack** surface make it a more secure and manageable option for accessing and managing EC2 instances within AWS environments.

ℹ️ **Learn More**:To understand the recommended security best practices for using Session Manager effectively. Read more about this Session Manager Security Best Practices.

COMMON ERRORS:

1. **Instance Not Listed in Session Manager:**
   - **Error:** `Instance Not Listed in Session Manager` - This happens when an EC2 instance fails to appear or is inaccessible within the Session Manager console for establishing a session.
   - **Solution:** Ensure the instance is running and connected to the internet. Confirm that the Systems Manager Agent (SSM Agent) is properly installed and running on the instance. Verify that the instance is associated with the correct IAM role with the necessary permissions for Session Manager.
2. **IAM Role Permission Issues:**
   - **Error:** `IAM Role Permission Issues` - Occurs when the IAM role associated with the EC2 instance lacks the necessary permissions for Systems Manager Session Manager, resulting in an inability to initiate a session.

- **Solution:** Double-check the IAM role associated with the EC2 instance and ensure it includes the required policies (like AmazonSSMManagedInstanceCore) or a custom policy with appropriate permissions for Systems Manager. Review the role's policies, ensure they are correctly attached, and grant any missing permissions related to Session Manager.

3. **SSM Agent Not Responding:**
   - **Error:** `SSM Agent Not Responding` - This error occurs when the SSM (Systems Manager) Agent on the EC2 instance becomes unresponsive or fails to establish a connection with the Systems Manager service.
   - **Solution:** First, check the status of the SSM Agent by running the command `sudo systemctl status amazon-ssm-agent`. If the agent status indicates it's not running or inactive, restart the SSM Agent service using `sudo systemctl start amazon-ssm-agent`.
4. **Session Manager Connection Failure:**
   - **Error:** `Session Manager Connection Failure` - This occurs when attempts to establish a connection to the EC2 instance via AWS Systems Manager Session Manager fail.
   - **Solution:** Start by checking the IAM role attached to the EC2 instance. Ensure that the IAM role associated with the instance includes the necessary permissions required for Systems Manager Session Manager access.