Project in AWS
Practice Lab

Build Solutions across VPCs with Peering

Andra-Diana Popescu

2025

ABOUT THIS LAB

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses. In this lab, you will create a new VPC for your WordPress blog to run from. You will then create a VPC peering connection between the new VPC and an existing database VPC. By the end of this lab, you will understand how to create a new VPC from scratch, attach internet gateways, edit routing tables, and peer multiple VPCs together.

LEARNING OBJECTIVES

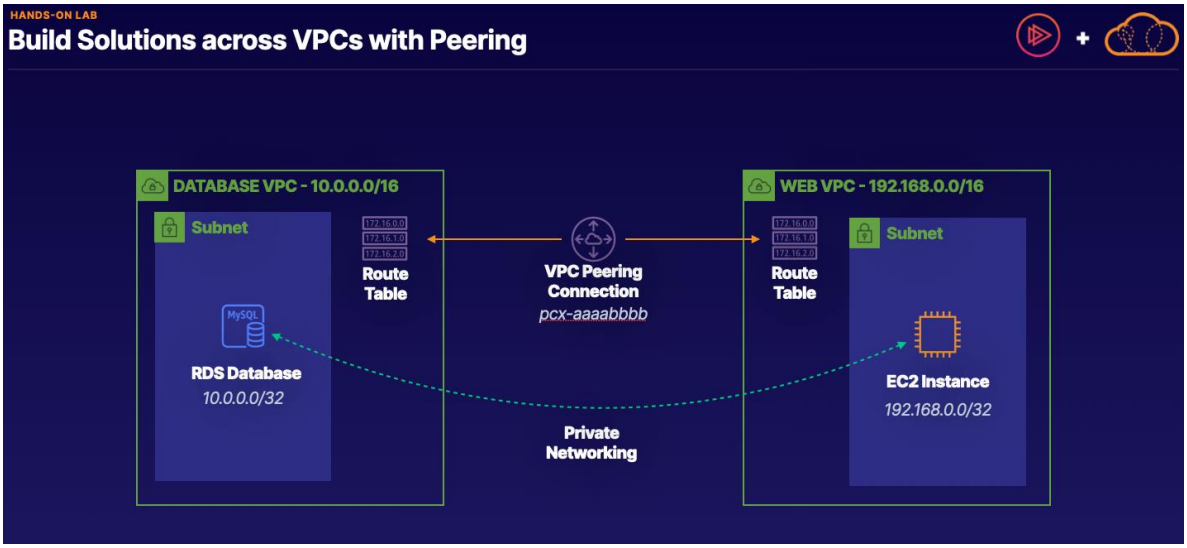
- Create Web_VPC Subnets and Attach a New Internet Gateway
- Create a Peering Connection
- Create an EC2 Instance and configure Wordpress
- Modify the RDS Security Groups to Allow Connections from the Web_VPC VPC
- Test WordPress

AWS Documentation about VPC and subnets:
https://docs.aws.amazon.com/vpc/latest/userguide/configure-subnets.html

Source: https://learn.acloud.guru/course/certified-solutions-architect-associate/

Table of Contents

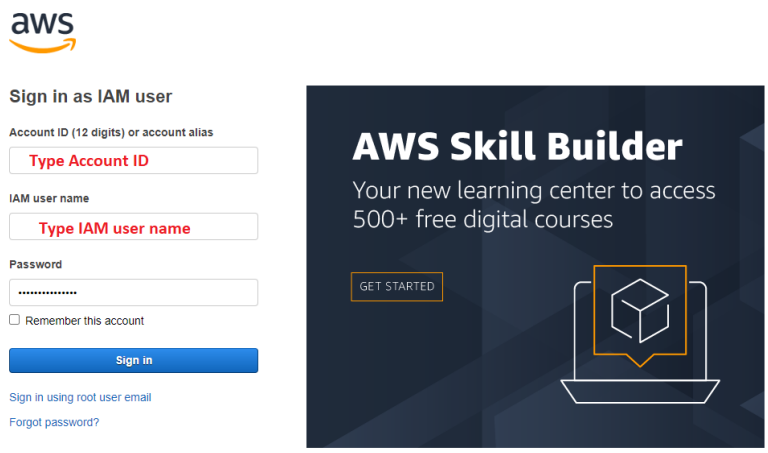
Lab Diagrams



We have the AWS account in **us-east-1** Region, and inside our AWS account, we have a VPC that we'll call it DATABASE VPC. Inside that, we have a subnet and a route table and there's also an RDS MySQL database, which will be created it automatically inside the subnet as well.

Our scenario is that our company is looking to set up a WordPress blog as part of a new internal site. Your database administrator created this RDS database in the VPC that was dedicated to only hosting databases. We need to set up a new WEB VPC with a subnet and routing table, and in the subnet, we'll have an EC2 instance for our blog. Then, we'll configure a VPC peering connection between the 2 VPCs to ensure private networking connection is possible between the EC2 instance and the RDS database.

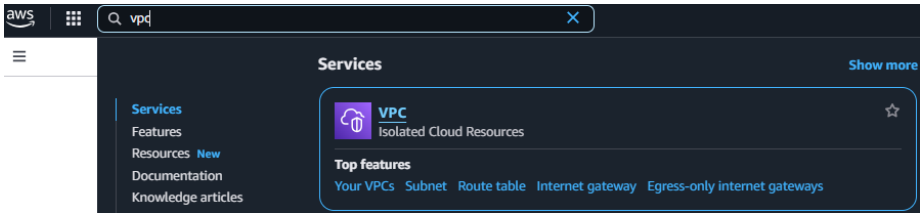
Log in to your AWS account



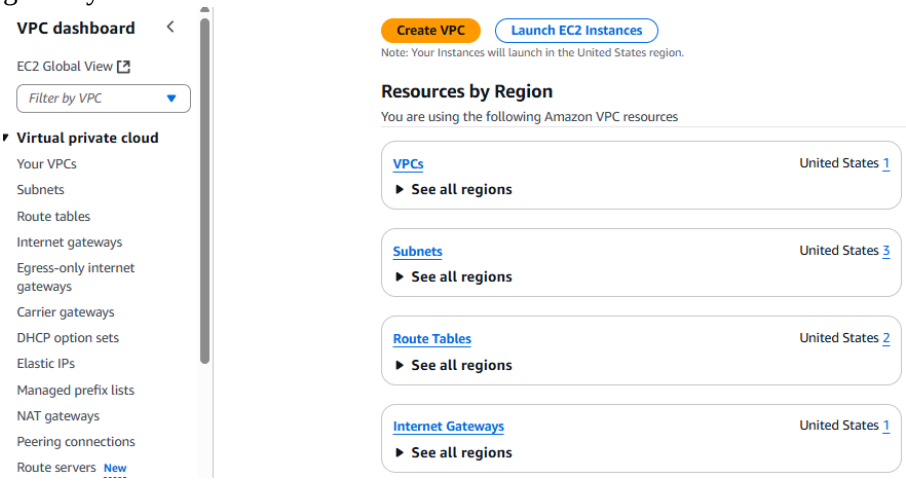
1. Create **Web_VPC** Subnets and Attach a New Internet Gateway

1.1. Create a VPC

1. Once you are logged in to the AWS Management Console, navigate to **VPC**.

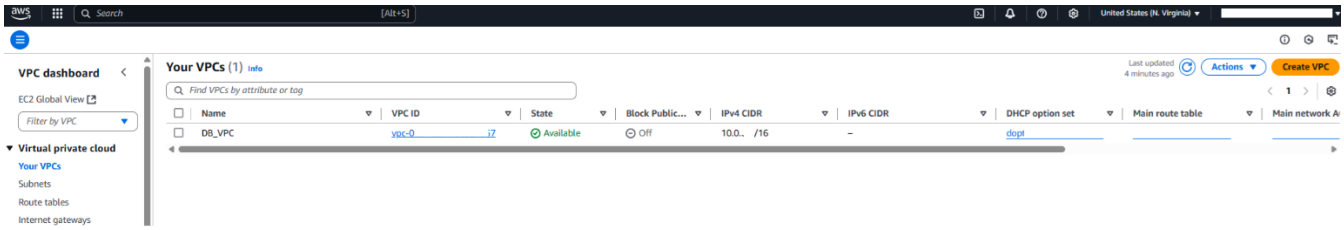


2. On this page, we can see that we already have 1 VPC, 3 subnets, 2 route tables and an internet gateway.

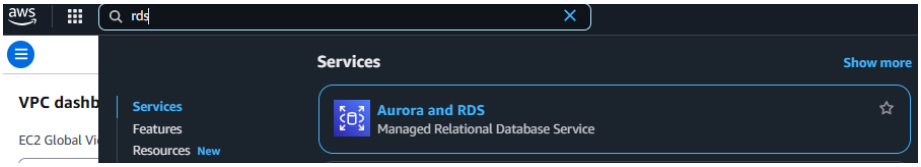


3. Under ***Resources by Region***, click **VPCs**.

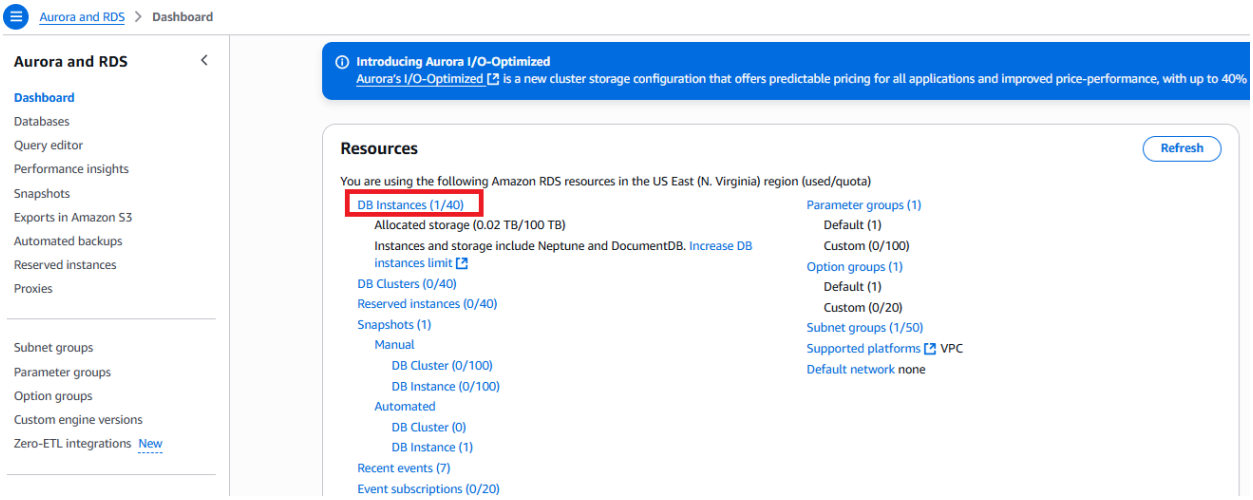
4. As you can see, we have our database VPC that was created with this hands-on lab.



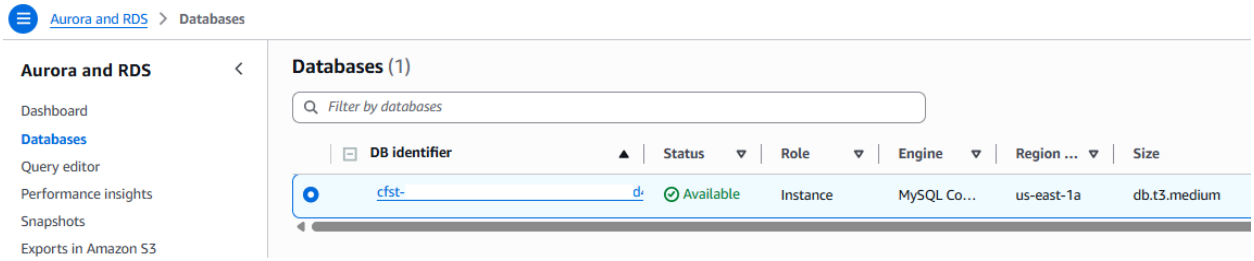
5. Use the top search bar to look for and navigate to **RDS** in a new tab.



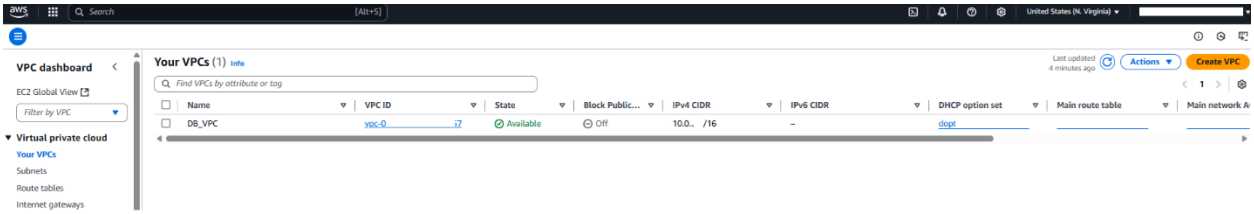
6. Click **DB Instances**, and observe the instance created for this lab.



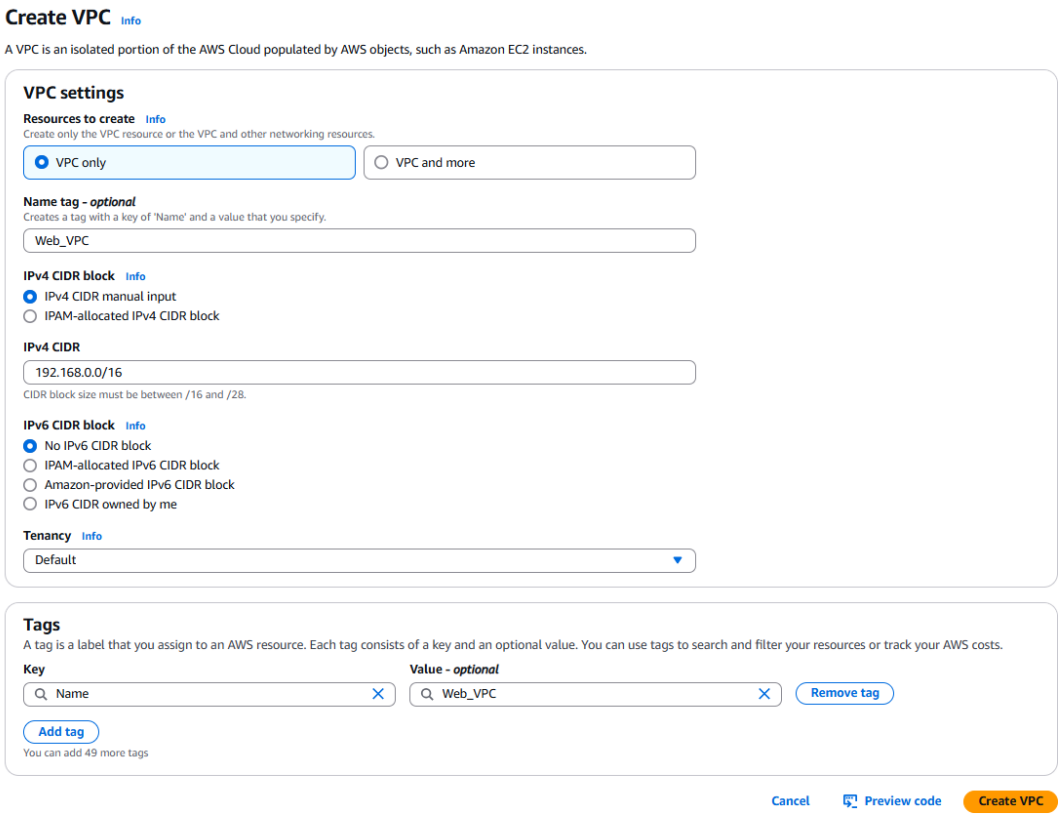
Note: Keep this tab open for use later in the lab.



7. Go back to your VPC tab and click Create **VPC**.

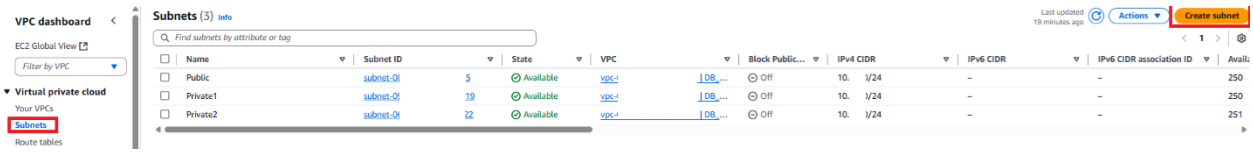


8. Ensure the **VPC only** option is selected.
9. Set the following values:
    a. ***Name tag***: Enter **Web_VPC**.
    b. ***IPv4 CIDR block***: Enter **192.168.<number>.<number>/16**. The IPv4 CIDR block will need to be different from the database VPC (**10.0.<number>.<number>/16**) that was created with this lab.
10. Leave the rest of the settings as their defaults, and click **Create VPC**.



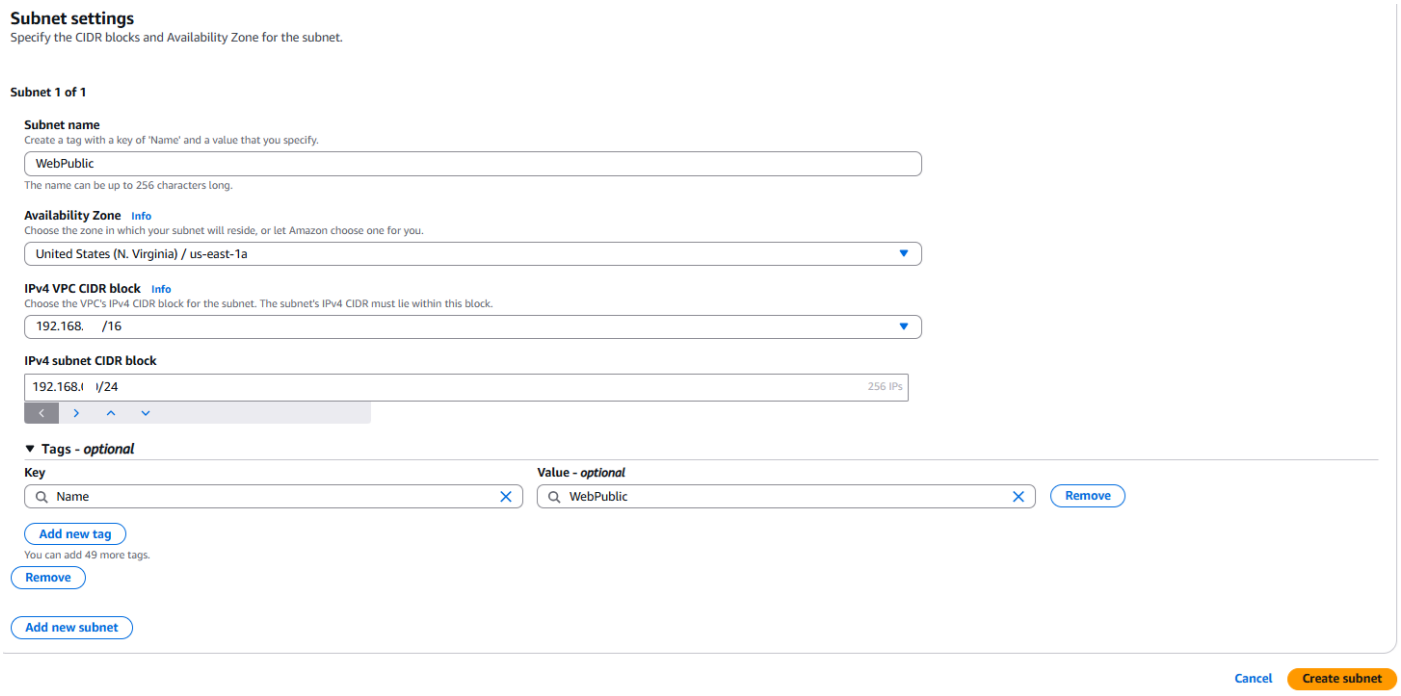
1.2. Create a Subnet

1. On the left menu under ***VIRTUAL PRIVATE CLOUD***, select **Subnets**.
2. Click **Create subnet**.



3. For ***VPC ID***, select the newly created **Web_VPC**.

4. Under ***Subnet settings***, set the following values:
    a. ***Subnet name***: Enter **WebPublic**.
    b. ***Availability Zone***: Select **us-east-1a**.
    c. ***IPv4 CIDR block***: Enter **192.168.<number>.<number>/24**.
5. Click **Create subnet**.



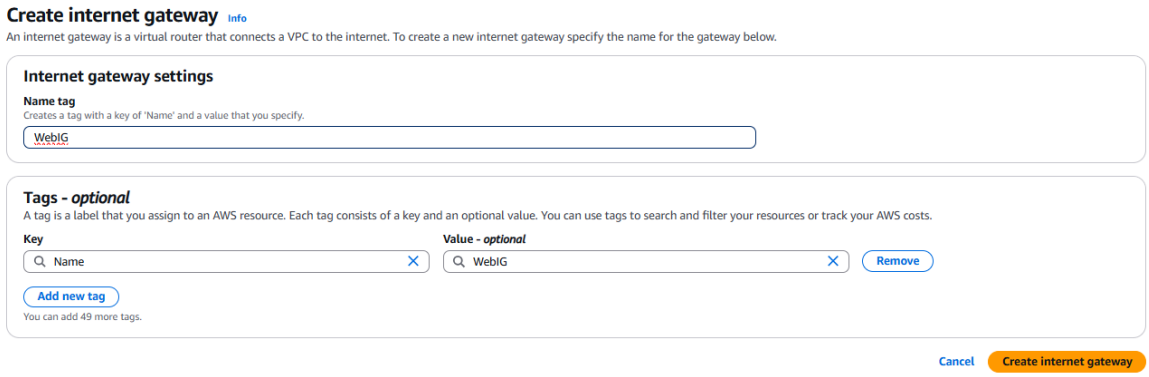
Note: For testing purposes, we need our WordPress instance to be available to the internet just to make sure it works, even though our scenario said it's going to be a private internal blog. To do this, we need to set up an internet gateway.

1.3. Create an Internet Gateway

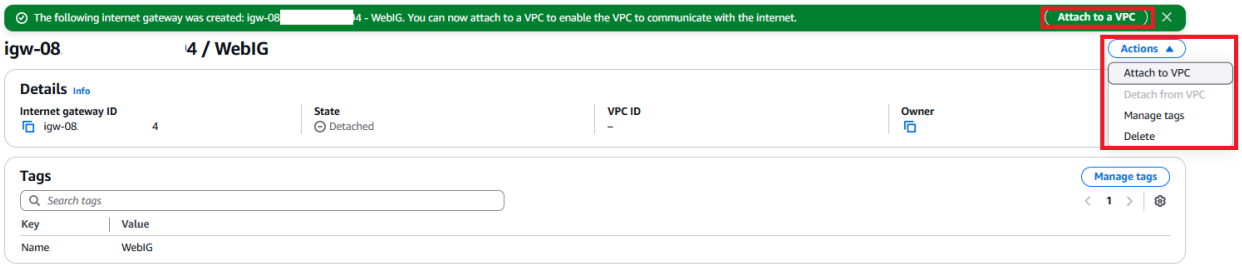
1. On the left menu, select **Internet Gateways**.
2. Click **Create internet gateway**.

3. For ***Name tag***, enter **WebIG**.

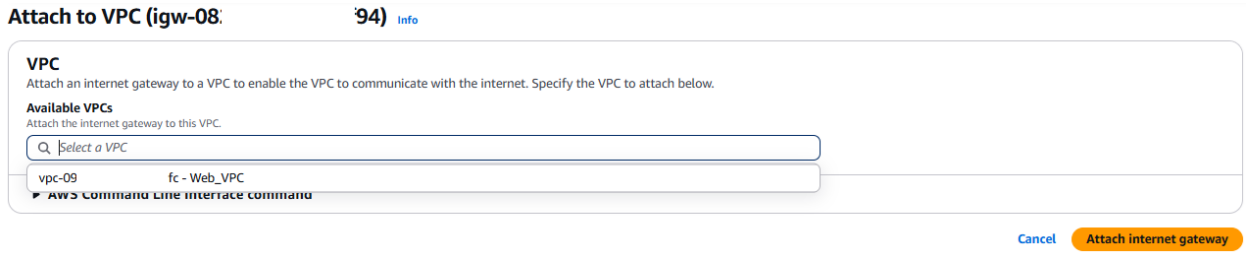
4. Click **Create internet gateway**.



5. In the green notification at the top of the page, click **Attach to a VPC**. Another option is **Actions → Attach to a VPC**.



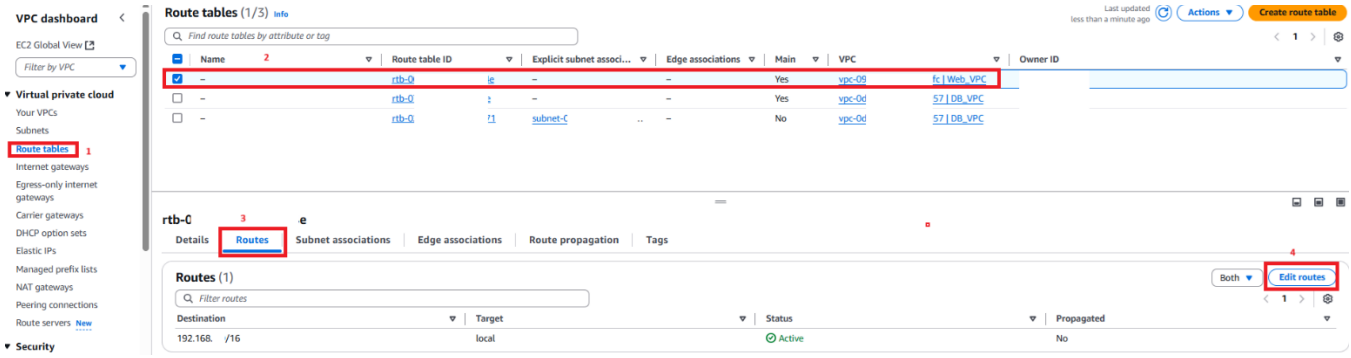
6. In ***Available VPCs***, select the **Web_VPC** and click **Attach internet gateway**.



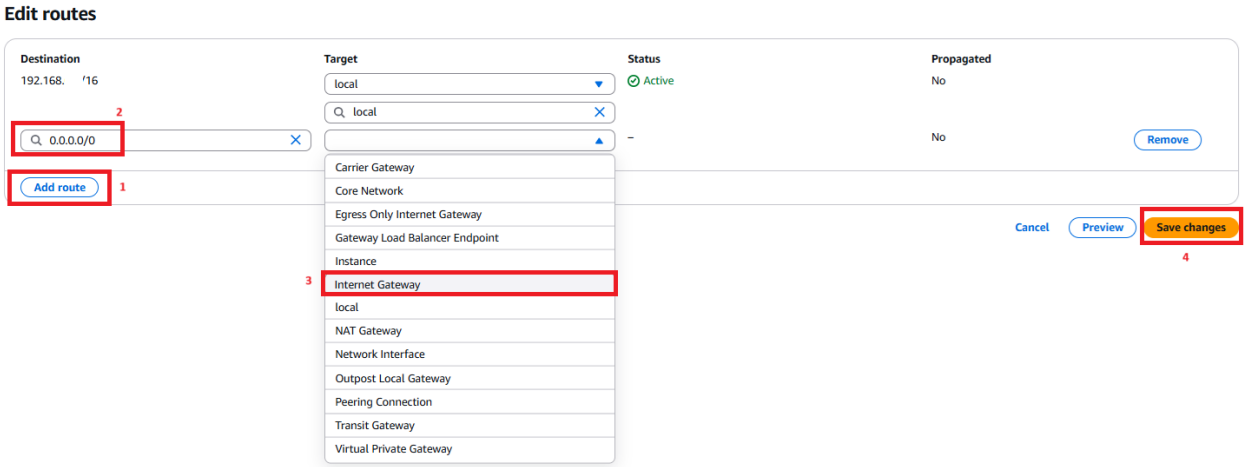
7. On the left menu, select **Route Tables**.

8. Select the checkbox for the **Web_VPC**.

9. Underneath, select the ***Routes*** tab and click **Edit routes**.



10. Click **Add route**.
11. Set the following values:
    a. ***Destination***: Enter **0.0.0.0/0**.
    b. ***Target***: Select **Internet Gateway**, and select the internet gateway that appears in the list.
12. Click **Save changes**.



Note: So now, anything in this subnet that tries to get to anything that isn't the 192.168 address, which is our local address, will go out to the internet via this internet gateway. Next, we'll link our 2 VPCs together with the peering connection.

2. Create a Peering Connection

1. On the left menu, select **Peering Connections**.
2. Click **Create peering connection**.

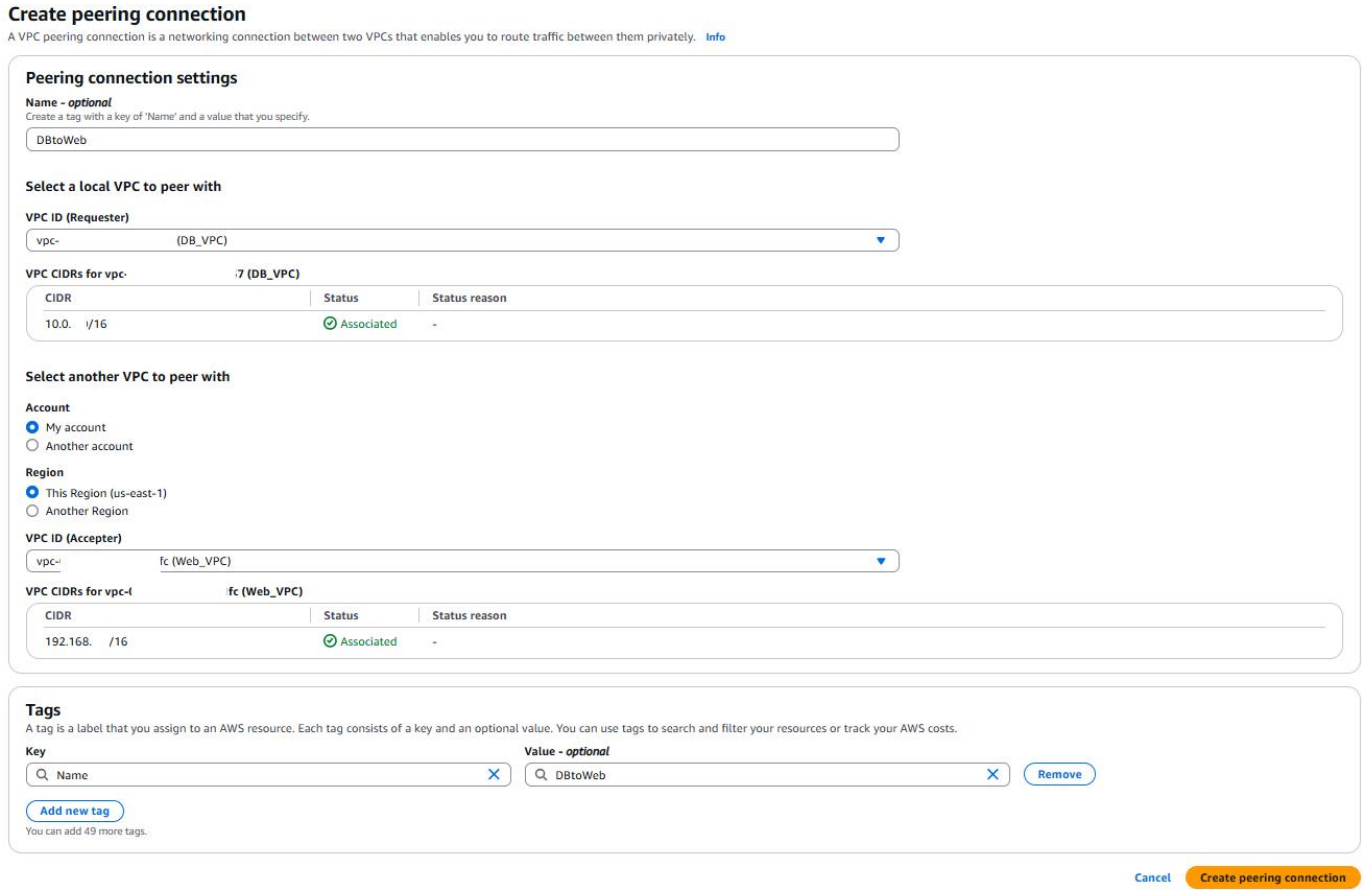
3. Set the following values:

    a. ***Name***: Enter **DBtoWeb**.

    b. ***VPC (Requester)***: Select the **DB_VPC**.

    c. ***VPC (Accepter)***: Select the **Web_VPC**.

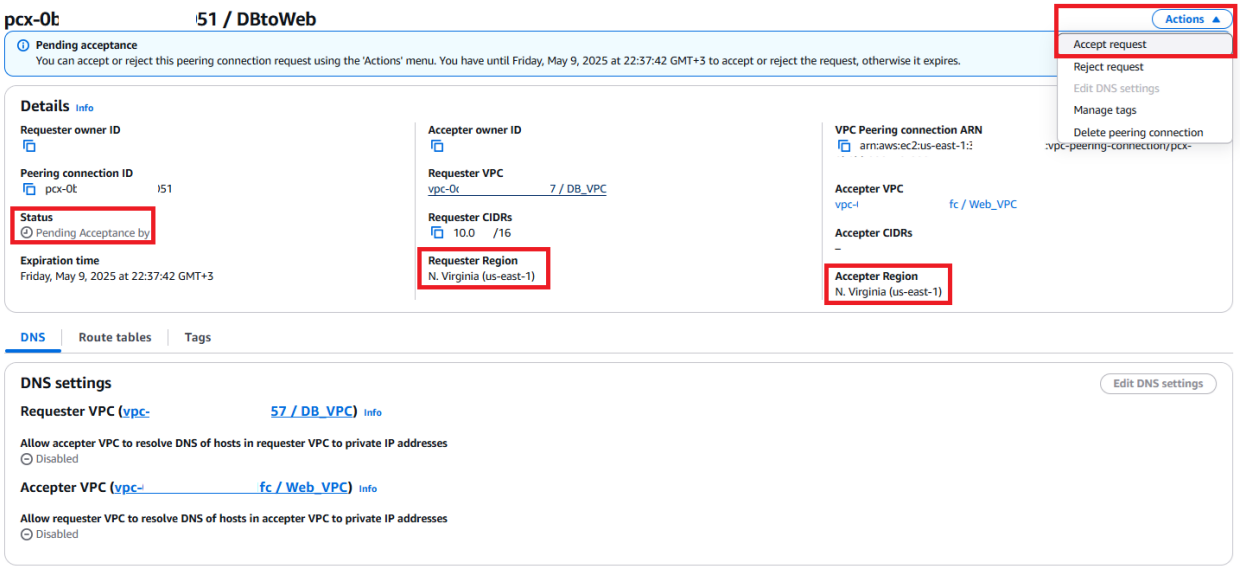
4. Click **Create peering connection**.



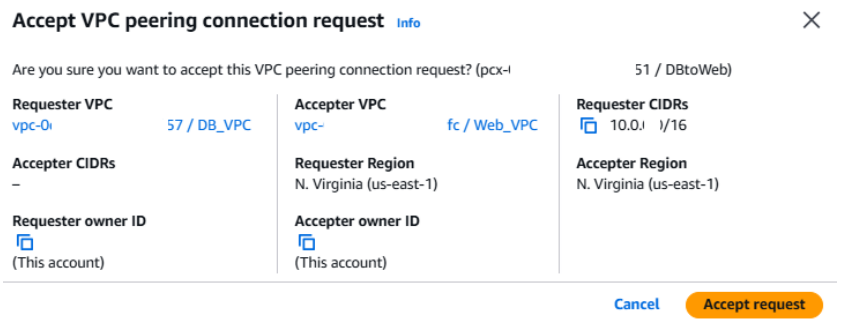
5. This will attempt to link these 2 VPCs together. Here on the details page for our new peering connection, we see that we have a notice that our peering connection is pending acceptance. You can imagine if you were working between different accounts, once you've created a peering

connection, someone on another account would have to accept it. In this case since we control both sides of this VPC peering connection, we can accept the connection ourselves. The status will change to active.

6. At the top of the page, click **Actions → Accept request**.



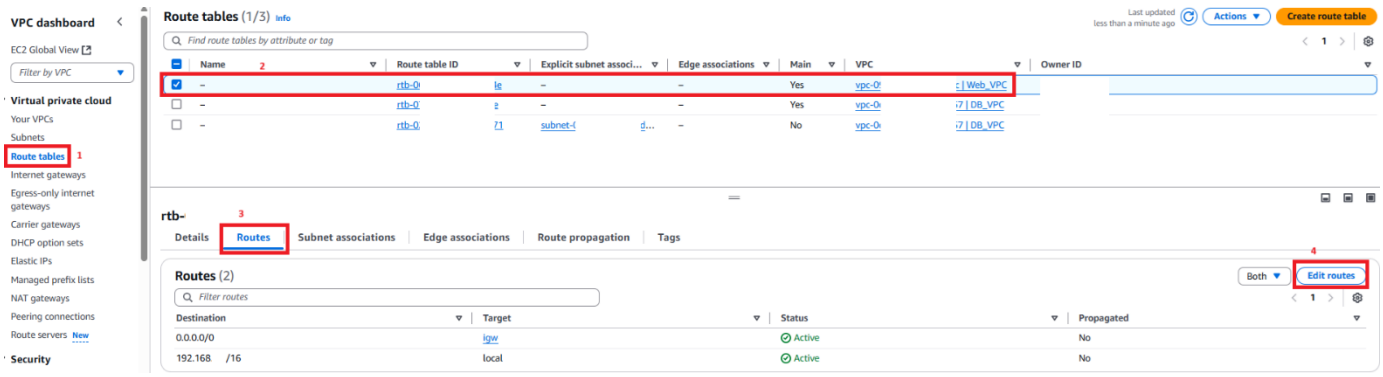
7. Click **Accept request**.



8. On the left menu, select **Route Tables**.

9. Select the checkbox for the **Web_VPC**.

10. Underneath, select the ***Routes*** tab, and click **Edit routes**.



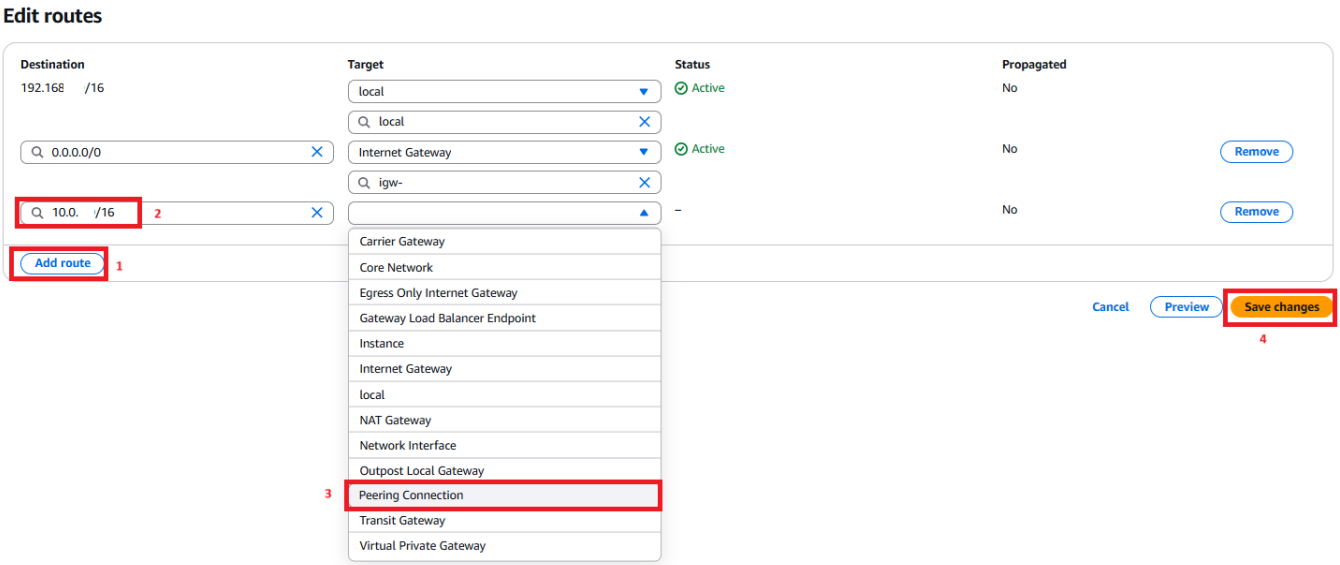
11. Click **Add route**.

12. Set the following values:

    a. ***Destination***: Enter **10.0.<number>.<number>/16**.

    b. ***Target***: Select **Peering Connection**, and select the peering connection that appears in the list.

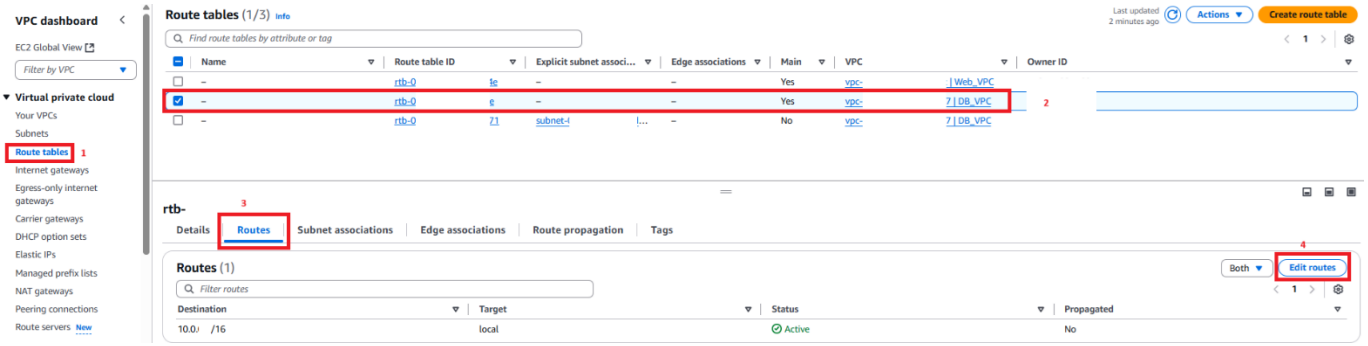
13. Click **Save changes**.



Note: We're going to do the same thing for our database VPC.

14. Go back to ***Route Tables***, and select the checkbox for the **DB_VPC** instance with a **Main** column value of **Yes**.

15. Underneath, select the ***Routes*** tab, and click **Edit routes**.



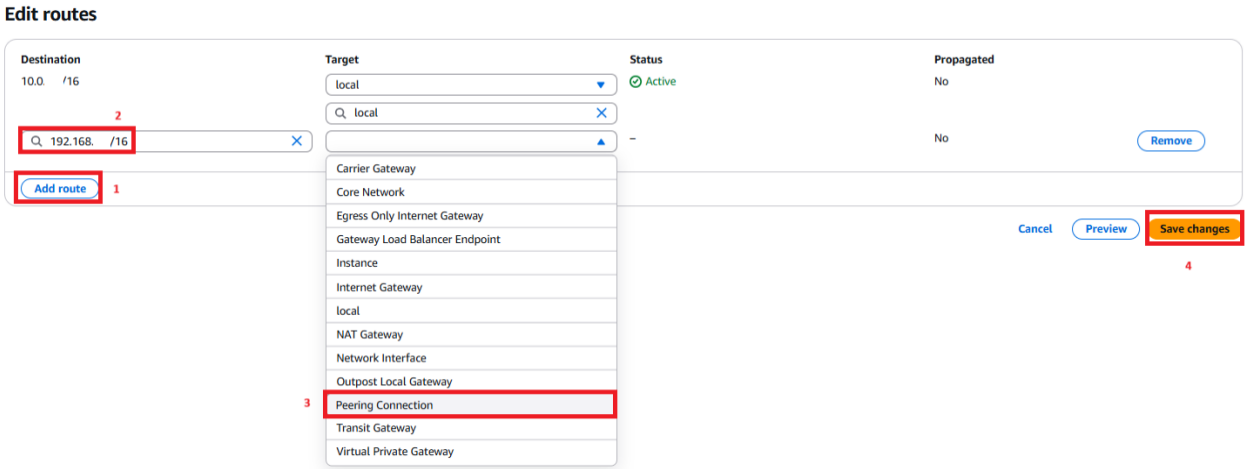
16. Click **Add route**.

17. Set the following values:

    a. ***Destination***: Enter **192.168.<number>.<number>/16**.

b. ***Target***: Select Peering Connection, and select the peering connection that appears in the list.

18. Click **Save changes**.

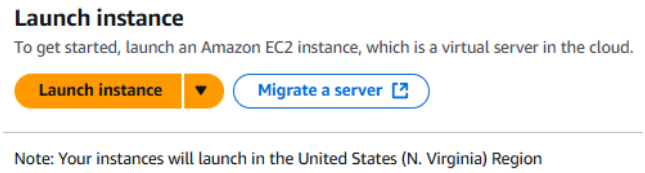


Note: Our VPCs can communicate with each other now.

3. Create an EC2 Instance and Configure WordPress

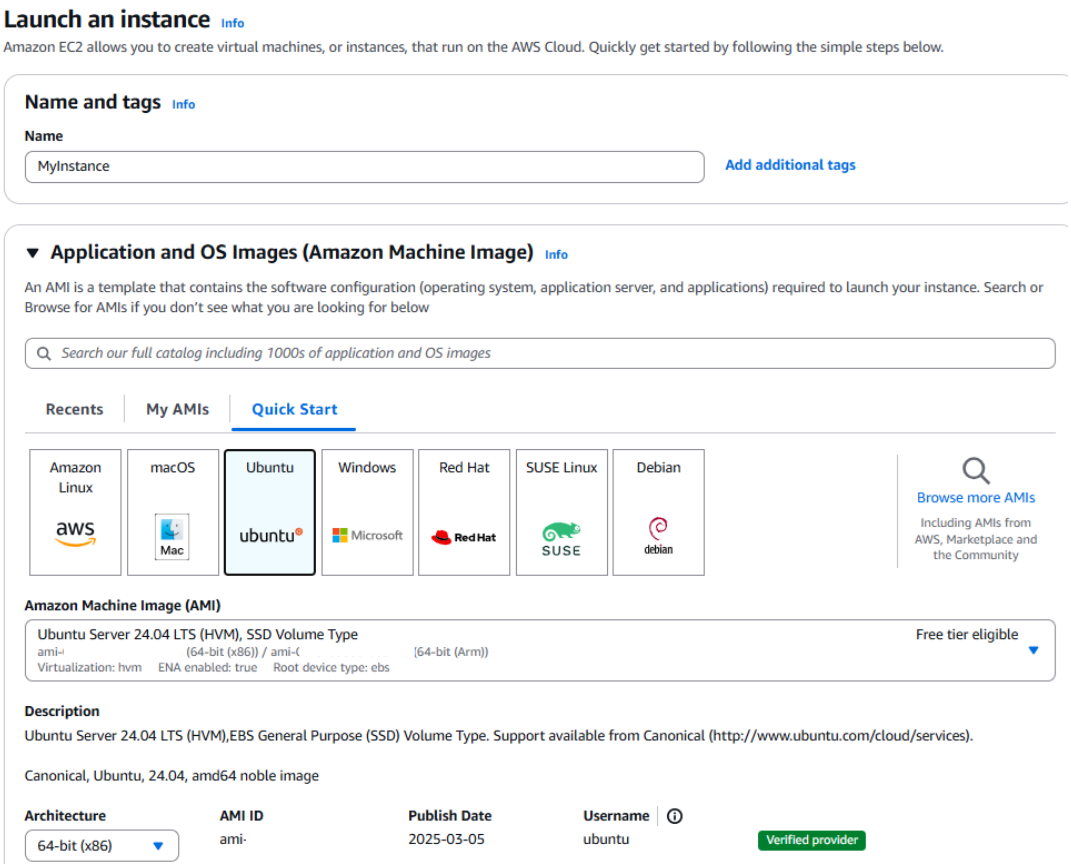
1. In a new browser tab, navigate to **EC2**.

2. Click **Launch instance → Launch instance**.

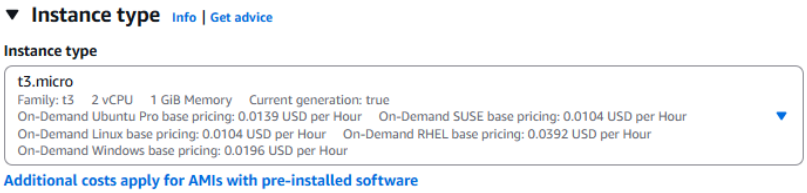


3. Give a name to the instance. Scroll down and under ***Quick Start***, select the **Ubuntu** image box.

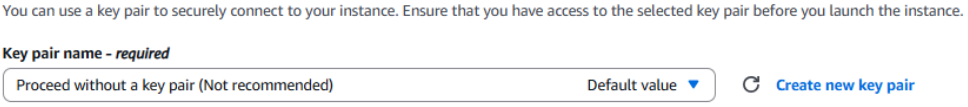
4. Under ***Amazon Machine Image (AMI)***, click the dropdown and select **Ubuntu Server 24.04 LTS**.



5. Under ***Instance type***, click the dropdown and select **t3.micro**.



6. In this lab scenario, for ***Key pair***, click the dropdown and select **Proceed without a key pair**. For production environment you should create a key pair or use an existing key pair.



7. In the ***Network settings*** section, click the **Edit** button.

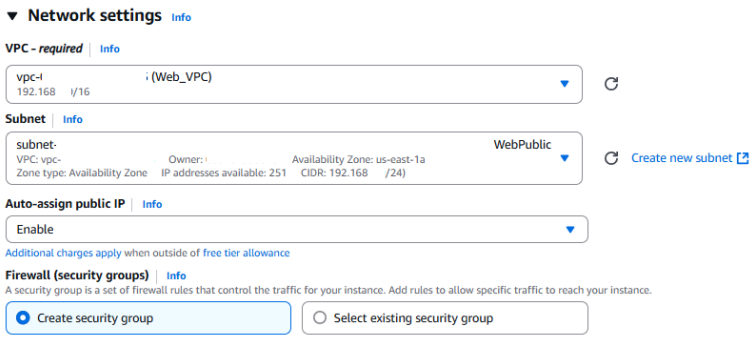
8. Set the following values:

    a. ***VPC***: Select the **Web_VPC**.

    b. ***Subnet***: Ensure the **WebPublic** subnet is selected.

    c. ***Auto-assign public IP***: Select **Enable**.

9. Under ***Firewall (security groups)***, ensure **Create security group** is selected (the default value).

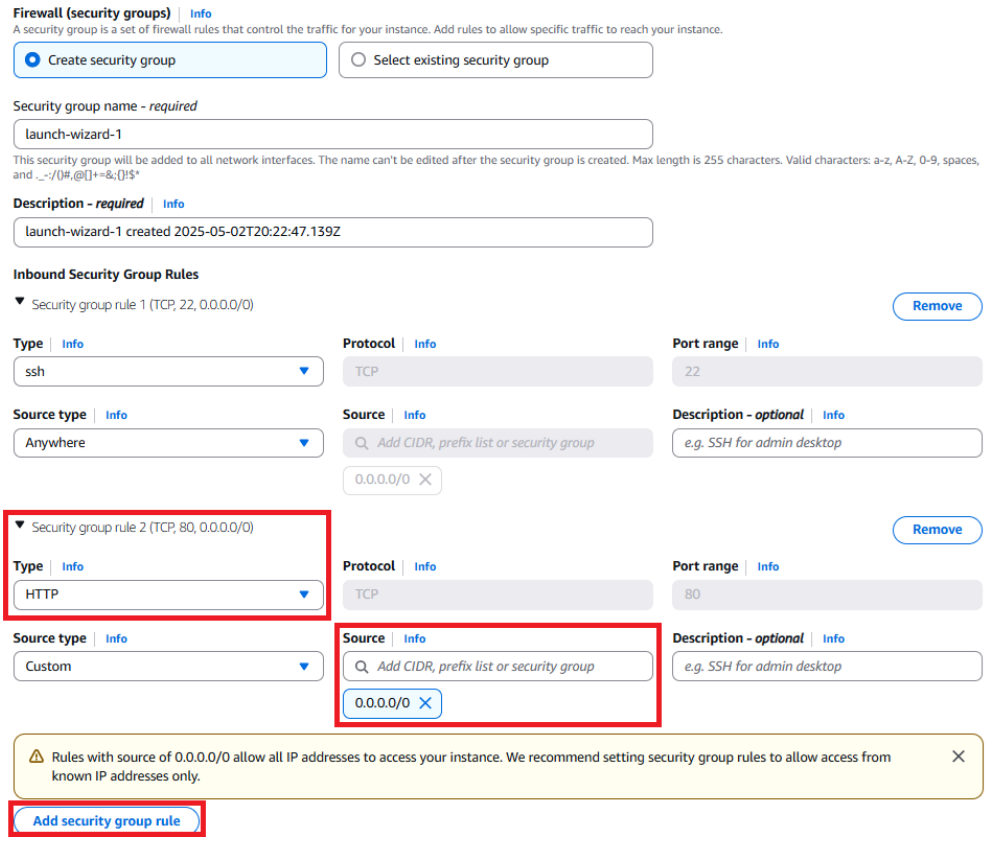


10. Scroll down and click **Add security group rule**.

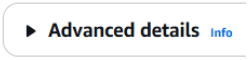
11. Set the following values for the new rule (i.e., **Security group rule 2**):

    a. ***Type***: Select **HTTP**.

    b. ***Source***: Select **0.0.0.0/0**.



12. Scroll to the bottom, and expand **Advanced details**.



13. At the bottom, under ***User data***, copy and paste the following bootstrap script:

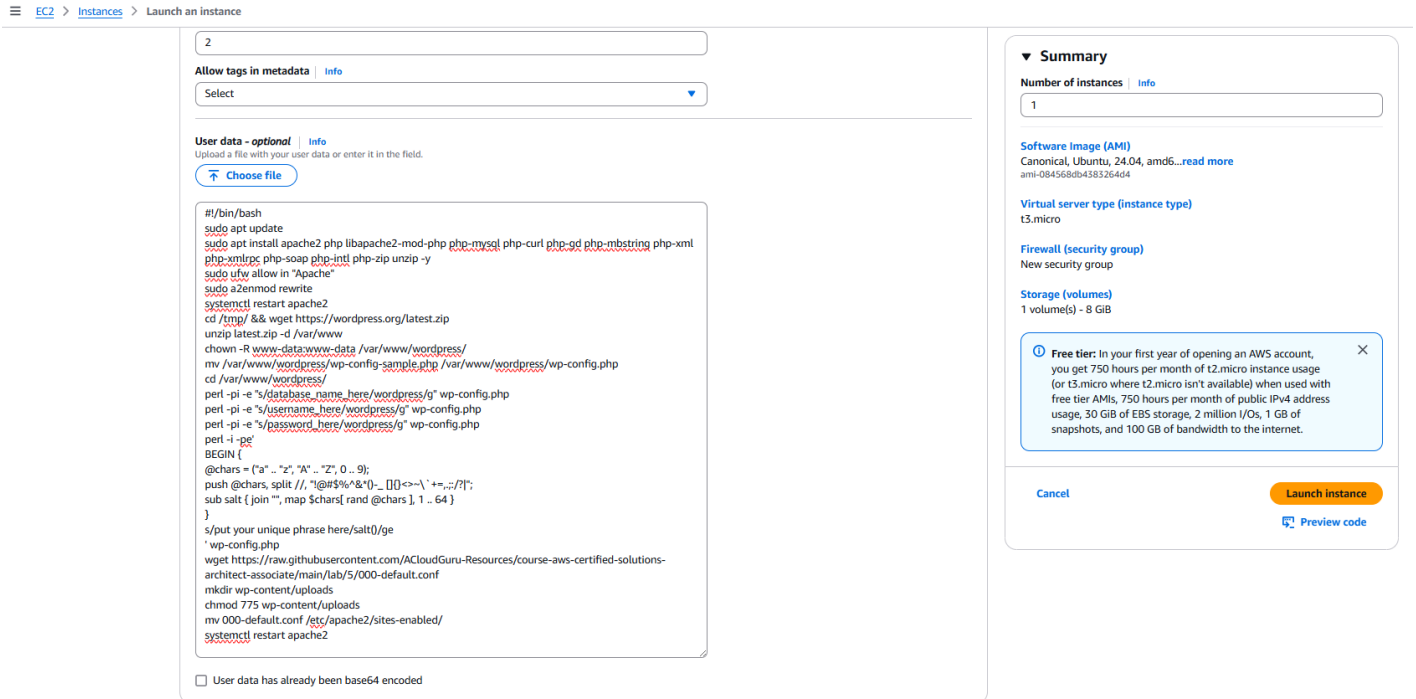
```
#!/bin/bash

sudo apt update

sudo apt install apache2 php libapache2-mod-php php-mysql php-curl php-gd php-mbstring
php-xml php-xmlrpc php-soap php-intl php-zip unzip -y

sudo ufw allow in "Apache"

sudo a2enmod rewrite

systemctl restart apache2

cd /tmp/ && wget https://wordpress.org/latest.zip

unzip latest.zip -d /var/www

chown -R www-data:www-data /var/www/wordpress/

mv /var/www/wordpress/wp-config-sample.php /var/www/wordpress/wp-config.php

cd /var/www/wordpress/

perl -pi -e "s/database_name_here/wordpress/g" wp-config.php

perl -pi -e "s/username_here/wordpress/g" wp-config.php

perl -pi -e "s/password_here/wordpress/g" wp-config.php

perl -i -pe'

BEGIN {

@chars = ("a" .. "z", "A" .. "Z", 0 .. 9);

push @chars, split //, "!@#$%^&*()-_ []{}<>~\`+=,.;:/?|";

sub salt { join "", map $chars[ rand @chars ], 1 .. 64 }

}

s/put your unique phrase here/salt()/ge

' wp-config.php

wget https://raw.githubusercontent.com/ACloudGuru-Resources/course-aws-certified-
solutions-architect-associate/main/lab/5/000-default.conf

mkdir wp-content/uploads

chmod 775 wp-content/uploads
```

mv 000-default.conf /etc/apache2/sites-enabled/

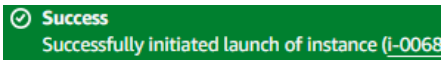
systemctl restart apache2

14. At the bottom, click **Launch Instance**.

    Note: It may take a few minutes for the new instance to launch. The script is going to install our Apache web server and get WordPress installed when the instance is created, and then we can go in and make a few changes to allow our WordPress installation to work properly. We will need to point our WordPress instance to the RDS database that was created with this lab environment.

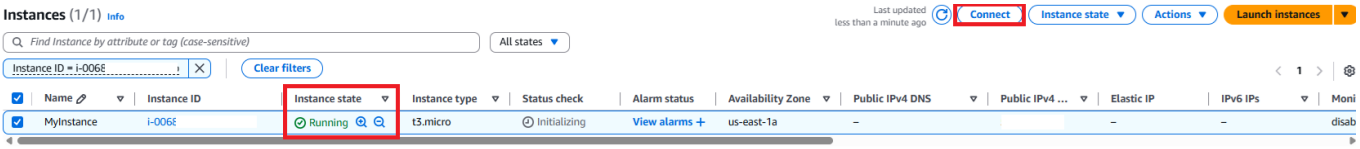


15. From the green box that appears after the instance launches, open the link for the instance in a new browser tab.



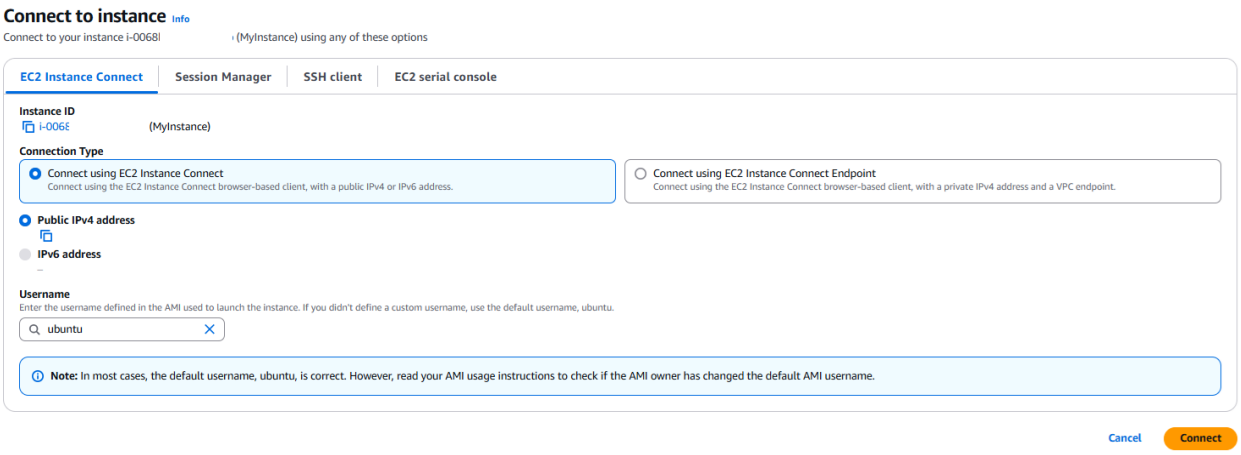
16. Observe the **Instance state** column, and check to ensure it is **Running** before you proceed.

17. Select the checkbox for the new instance and click **Connect**.



18. Click **Connect**.

    Note: The startup script for the instance may take a few minutes to complete and you may need to wait for it to complete before proceeding with the next step.



19. To confirm WordPress installed correctly, view the configuration files in the CLI:

    cd /var/www/wordpress

    ls

```
ubuntu@ip-192-168-      :~$ cd /var/www/wordpress
ubuntu@ip-192-168-      :/var/www/wordpress$ ls
index.php    readme.html    wp-admin           wp-comments-post.php  wp-content   wp-includes        wp-load.php   wp-mail.php      wp-signup.php    xmlrpc.php
license.txt  wp-activate.php  wp-blog-header.php  wp-config.php         wp-cron.php  wp-links-opml.php  wp-login.php  wp-settings.php  wp-trackback.php
ubuntu@ip-192-168-      :/var/www/wordpress$
```

    Note: Now we need to configure WordPress and we're going to edit a config file.

20. To configure WordPress, open ***wp-config.php***: **sudo vim wp-config.php**

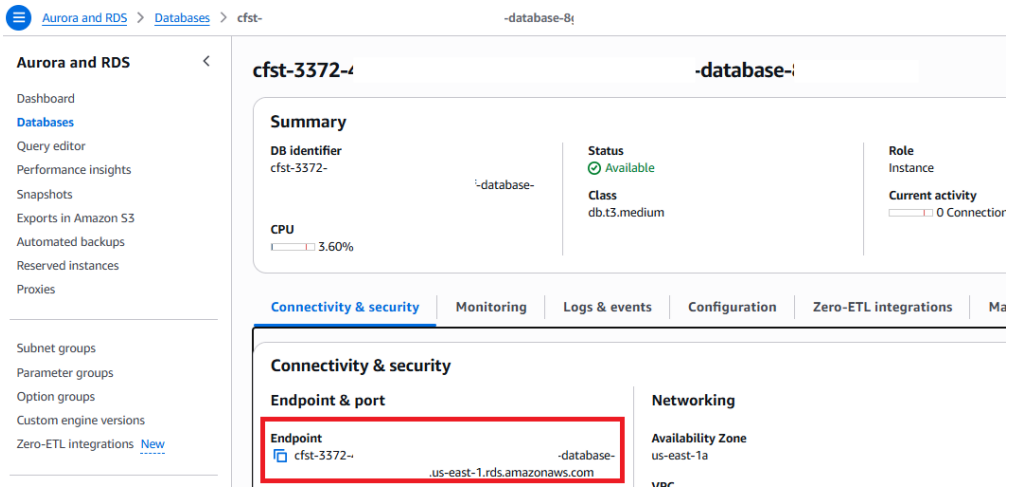
```
sudo vim wp-config.php
```

```
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the installation.
 * You don't have to use the website, you can copy this file to "wp-config.php"
 * and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * Database settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://developer.wordpress.org/advanced-administration/wordpress/wp-config/
 *
 * @package WordPress
 */

// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** Database username */
define( 'DB_USER', 'wordpress' );

/** Database password */
define( 'DB_PASSWORD', 'wordpress' );

/** Database hostname */
define( 'DB_HOST', 'localhost' );

/** Database charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The database collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );
```

21. Go back to your browser tab with RDS.

22. Click the link to open the provisioned RDS instance.

23. Under ***Connectivity & security***, copy the RDS **Endpoint**.

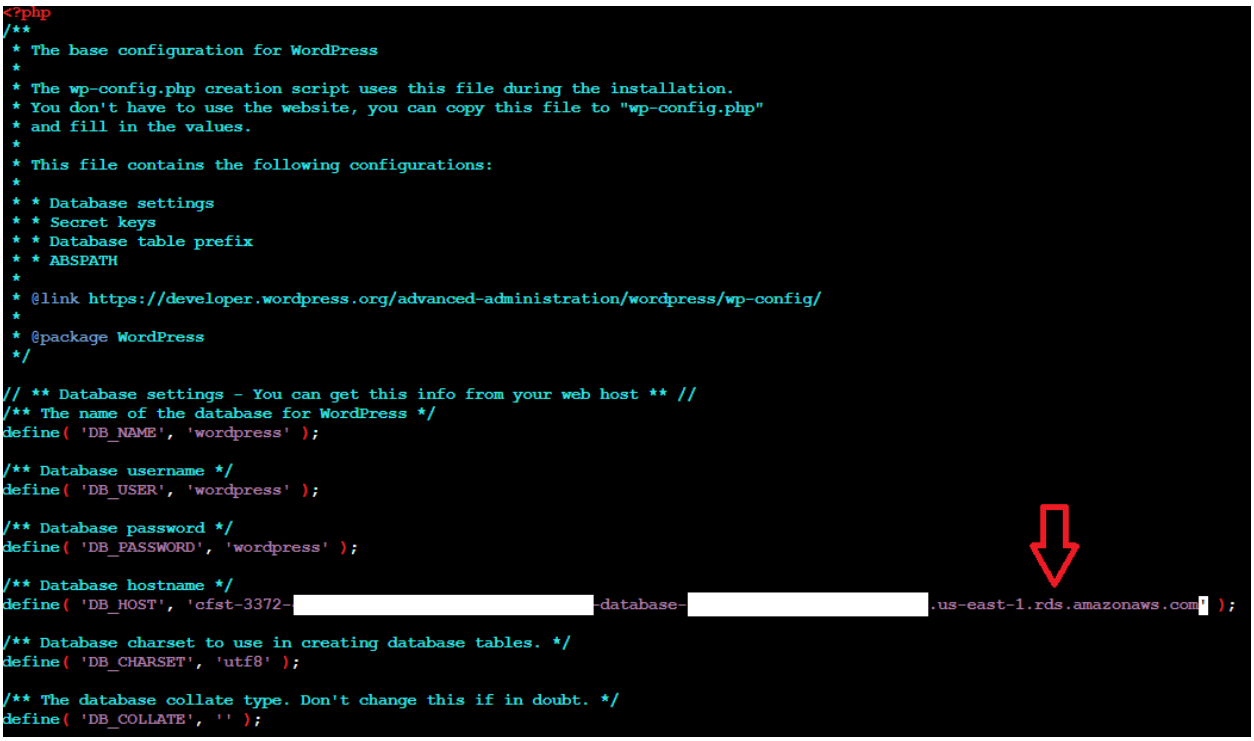


24. Go back to the tab with the terminal, and scroll down to **/** Database hostname **/**.

25. Press "i" to enter Insert mode.

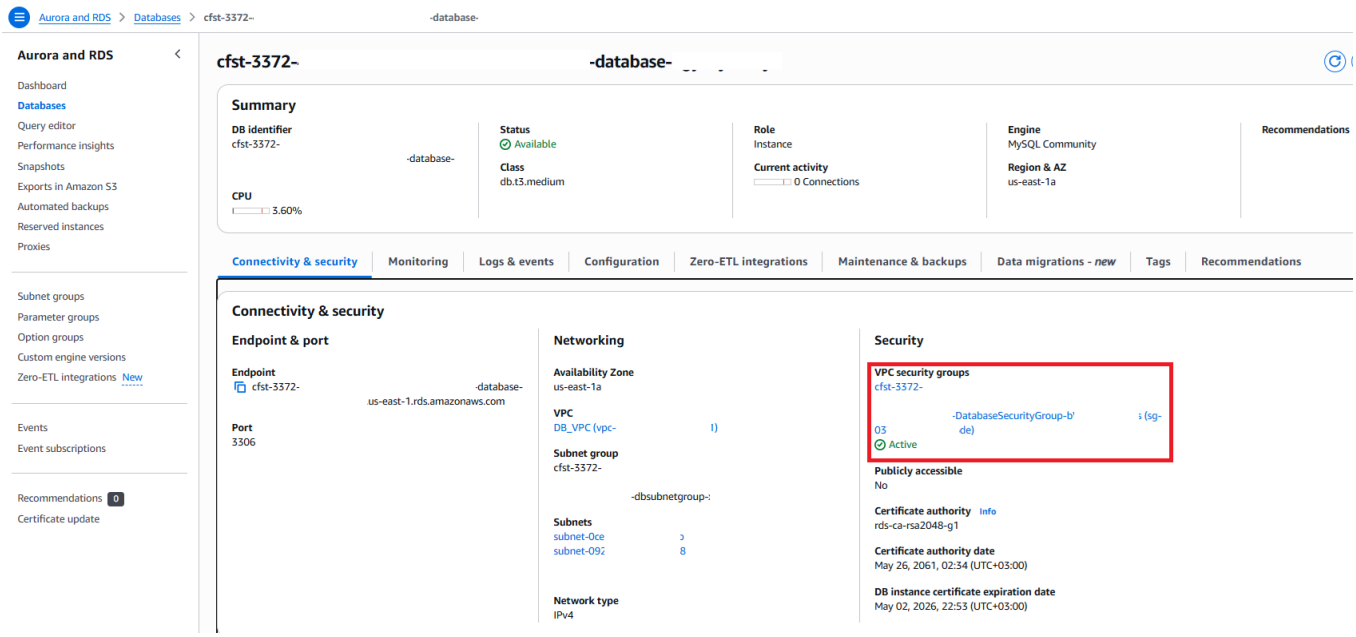
26. Replace **localhost** with the RDS endpoint you just copied. Ensure it remains wrapped in single quotes.

27. Press "**ESC**" followed by "**:wq**", and press **Enter**. Leave this tab open.

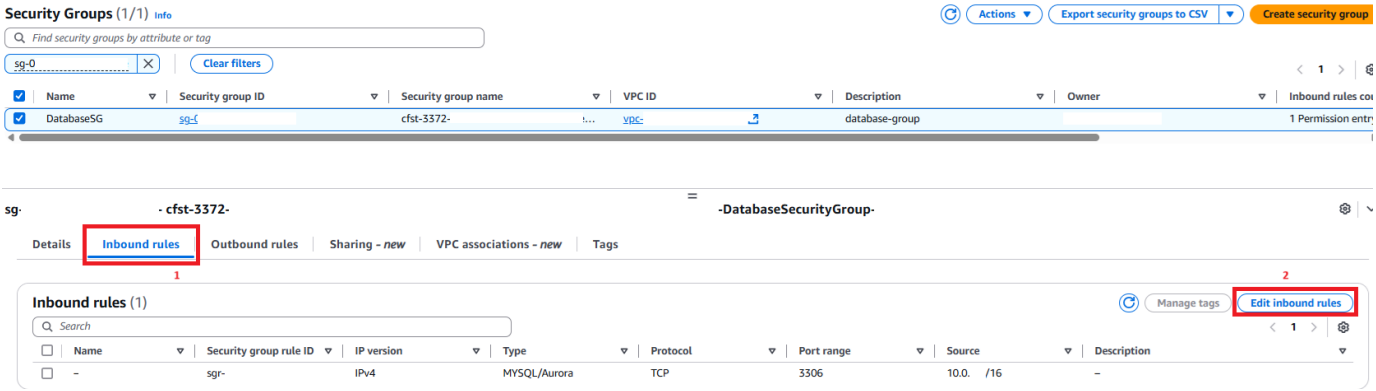


4. Modify the RDS Security Groups to Allow Connections from the **Web_VPC** VPC

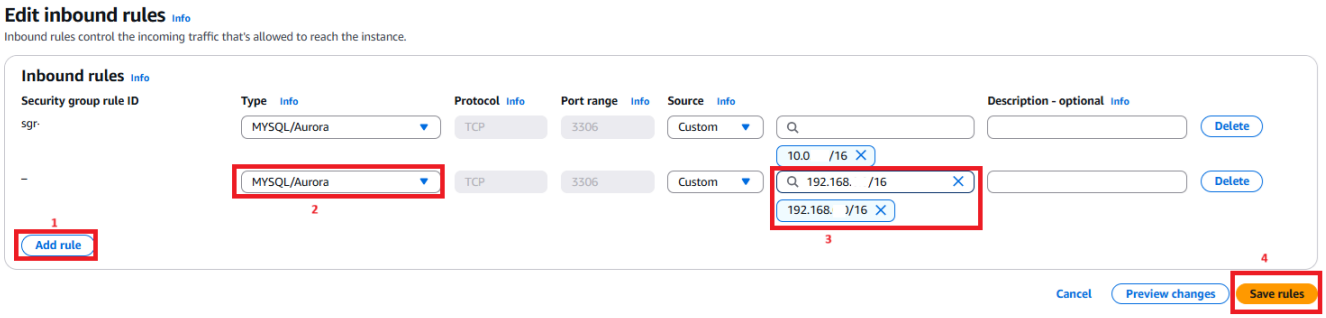
1. Go back to your RDS browser tab.
2. In ***Connectivity & security***, click the active link under **VPC security groups**.



3. Checkmark the **DatabaseSG** Security Group.
4. At the bottom, select the **Inbound rules tab**.
5. Click **Edit inbound rules**.



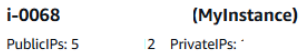
6. Click **Add rule**.
7. Under ***Type***, search for and select **MYSQL/Aurora**.
8. Under ***Source***, search for and select **192.168.<number>.<number>/16.**
9. Click **Save rules**.



Note: These 2 rules will allow anyone from the ***10.0.*** or the ***192.168.*** VPCs to connect to the database.

10. Return to the terminal page.

11. Below the terminal window, copy the public IP address of your server.



12. Open a new browser tab and paste the public IP address in the address bar. You should now see the WordPress installation page.

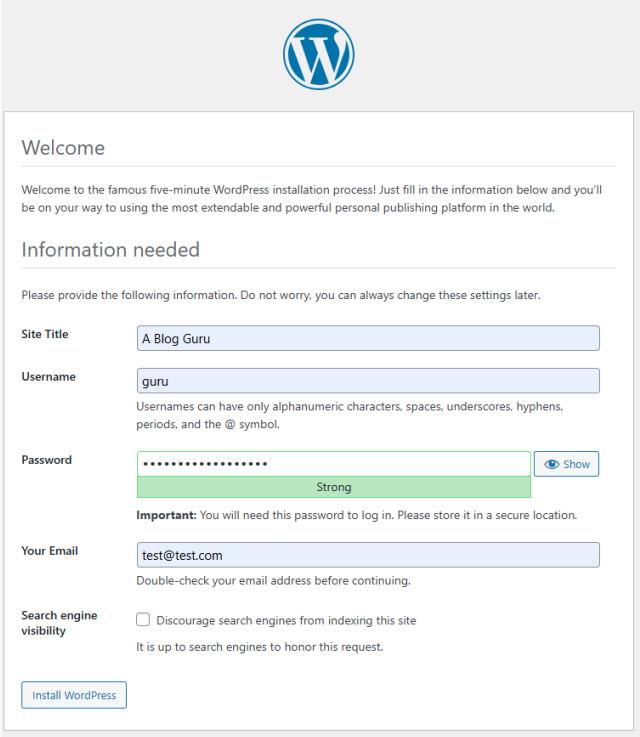
13. Set the following values:

    a. ***Site Title***: Enter **A Blog Guru**.

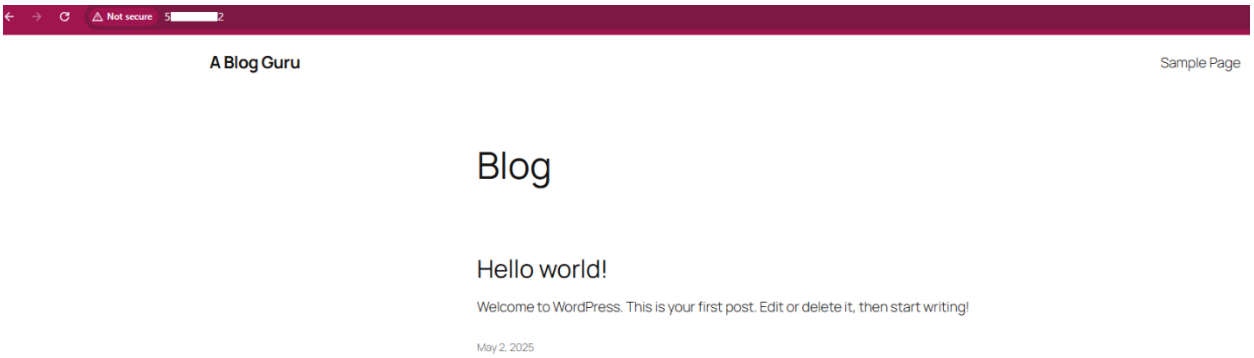
    b. ***Username***: Enter **guru**.

    c. ***Your Email***: Enter **test@test.com**.

14. Click **Install WordPress**.



15. Reload the public IP address in the address bar to view your newly created WordPress blog.



In this lab, we had an existing database VPC, and we created a new website VPC. We created a subnet, we edited our route table, and then we peered our 2 VPCs together so they could talk to each other. We also edited our route tables to ensure that traffic was flowing in the correct directions. We created our EC2 instance in the web VPC, and we configured WordPress to connect to our RDS instance, which exists in our database VPC.