

**Build Solutions across VPCs with Peering**

Introduction

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, we create a new VPC for our WordPress blog to run from. We then create a VPC peering connection between the new VPC and an existing database VPC. By the end of this lab, the user will understand how to create a new VPC from scratch, attach internet gateways, edit routing tables, and peer multiple VPCs together.

Solution

Log in to the AWS Management Console using the credentials provided on the lab instructions page. Make sure you're in the N. Virginia (us-east-1) region throughout the lab.

Create Web\_VPC Subnets and Attach a New Internet Gateway

Create a VPC

1. Navigate to VPC.
2. Under *Resources by Region*, click **VPCs**.
3. Click **Create VPC**.
4. Set the following values:
   * *Name tag:* **Web\_VPC**
   * *IPv4 CIDR block:* **192.168.0.0/16**
5. Leave the rest as their defaults and click **Create VPC**.

Create a Subnet

1. On the left menu under *VIRTUAL PRIVATE CLOUD*, select **Subnets**.
2. Click **Create subnet**.
3. Select the newly created Web\_VPC.
4. Under *Subnet settings*, set the following values:
   * *Subnet name:* **WebPublic**
   * *Availability Zone:* **us-east-1a**
   * *IPv4 CIDR block:* **192.168.0.0/24**
5. Click **Create subnet**.

Create an Internet Gateway

1. On the left menu, select **Internet Gateways**.
2. Click **Create internet gateway**.
3. In *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**.
6. In *Available VPCs*, select the Web\_VPC and click **Attach internet gateway**.
7. On the left menu, select **Route Tables**.
8. Select the Web\_VPC.
9. Underneath, select the *Routes* tab and click **Edit routes**.
10. Click **Add route**.
11. Set the following values:
    * *Destination:* **0.0.0.0/0**
    * *Target:* **Internet Gateway** > **WebIG**
12. Click **Save changes**.

Create a Peering Connection

1. On the left menu, select **Peering Connections**.
2. Click **Create peering connection**.
3. Set the following values:
   * *Name:* **DBtoWeb**
   * *VPC (Requester):* **DB\_VPC**
   * *VPC (Accepter):* **Web\_VPC**
4. Click **Create peering connection**.
5. At the top of the page, click **Actions** > **Accept Request**.
6. Click **Accept request**.
7. On the left menu, select **Route Tables**.
8. Select the Web\_VPC.
9. Underneath, select the *Routes* tab and click **Edit routes**.
10. Click **Add route**.
11. Set the following values:
    * *Destination:* **10.0.0.0/16**
    * *Target:* **Peering Connection** > **DBtoWeb**
12. Click **Save changes**.
13. Go back to *Route Tables* and select the DB\_VPC instance with a *Main* column value of *Yes*.
14. Under the *Routes* tab, click **Edit routes**.
15. Click **Add route**.
16. Set the following values:
    * *Destination:* **192.168.0.0/16**
    * *Target:* **Peering Connection** > **DBtoWeb**
17. Click **Save changes**.

Create an EC2 Instance and Configure WordPress

1. In a new browser tab, navigate to EC2.
2. Click **Launch instance** > **Launch instance**.
3. Scroll down to *Ubuntu Server 20.04 LTS* and click **Select**.
4. Select **t3.micro** as the instance type.
5. Click **Next: Configure Instance Details**.
6. Set the following values:
   * *Network:* **Web\_VPC**
   * Subnet:\* **WebPublic**
   * *Auto-assign Public IP:* **Enable**
7. At the bottom under *User data*, paste in the following bootstrap script:
8. #!/bin/bash
9. sudo apt update -y
10. sudo apt install php-curl php-gd php-mbstring php-xml php-xmlrpc php-soap php-intl php-zip perl mysql-server apache2 libapache2-mod-php php-mysql -y
11. wget https://github.com/ACloudGuru-Resources/course-aws-certified-solutions-architect-associate/raw/main/lab/5/wordpress.tar.gz
12. tar zxvf wordpress.tar.gz
13. cd wordpress
14. wget https://raw.githubusercontent.com/ACloudGuru-Resources/course-aws-certified-solutions-architect-associate/main/lab/5/000-default.conf
15. cp wp-config-sample.php wp-config.php
16. perl -pi -e "s/database\_name\_here/wordpress/g" wp-config.php
17. perl -pi -e "s/username\_here/wordpress/g" wp-config.php
18. perl -pi -e "s/password\_here/wordpress/g" wp-config.php
19. perl -i -pe'
20. BEGIN {
21. @chars = ("a" .. "z", "A" .. "Z", 0 .. 9);
22. push @chars, split //, "!@#$%^&\*()-\_ []{}<>~\`+=,.;:/?|";
23. sub salt { join "", map $chars[ rand @chars ], 1 .. 64 }
24. }
25. s/put your unique phrase here/salt()/ge
26. ' wp-config.php
27. mkdir wp-content/uploads
28. chmod 775 wp-content/uploads
29. mv 000-default.conf /etc/apache2/sites-enabled/
30. mv /wordpress /var/www/

apache2ctl restart

1. Click **Review and Launch**.
2. Scroll down to *Security Groups* and click **Edit security groups**.
3. Click **Add Rule**.
4. Select **HTTP** and click **Review and Launch**.
5. Click **Launch**.
6. On the dropdown, select **Proceed without a key pair**.
7. Select the checkbox acknowledging that you will not be able to connect to this instance unless you already know the password built into this AMI.
8. Click **Launch Instances** and then click **View Instances**.

**Note:** It may take a few minutes for the new instance to launch.

1. In a new browser tab, navigate to RDS.
2. Select the provisioned RDS instance.
3. Under *Connectivity & security*, copy the RDS endpoint for later use.
4. Navigate back to EC2.
5. Select the new instance and click **Connect**.
6. Click **Connect**.
7. To confirm WordPress installed correctly, view the configuration files:
8. cd /var/www/wordpress

ls

1. To configure WordPress, open wp-config.php:

sudo nano wp-config.php

1. Scroll down to /\*\* MySQL hostname \*/ and replace localhost with the RDS endpoint previously copied.
2. To save, press **Ctrl+X**, and then type Y and press **Enter**.

Modify the RDS Security Groups to Allow Connections from the Web\_VPC VPC

1. Navigate to RDS.
2. In *Connectivity & security*, click the active link under *VPC security groups*.
3. Select the *Inbound rules* tab and click **Edit inbound rules**.
4. Click **Add rule**.
5. Under *Type*, type and select **MYSQL/Aurora**.
6. Under *Source*, type and select **192.168.0.0/16**.
7. Click **Save rules**.
8. Return to the terminal.
9. At the bottom of the terminal window, copy the public IP address of our server.
10. Open a new browser tab and paste the public IP address in the address bar. You should now see the WordPress installation page.
11. Set the the following values :
    * *Site Title:* **A Blog Guru**
    * *Username:* **guru**
    * *Your Email:* Your email address
12. Click **Install WordPress**.
13. Reload the public IP address in the address bar to view our newly created WordPress blog.

Conclusion

Congratulations — you've completed this hands-on lab!

Troubleshooting

If the website isn't loading the way you'd expected at the end of this guide, here are some tips to help troubleshoot:

* Check the status of the lab objectives - are any not yet completed?
* Has everything we've setup successfully become ready to use? Check things like the VPC Peering Connection, which requires you to specifically accept the connection request.
* Does the database error page load after a minute or so of waiting, or does no page load at all? This gives a hint to whether the issue may be with the peering, or the security groups.