* Setting up a private network in AWS using VPC and provisioning remote machines:

1. Create a Virtual Private Cloud (VPC):

- Log in to your AWS Management Console.

- Navigate to the VPC dashboard.

- Click on "Create VPC" and specify the details, including the CIDR block for your VPC --🡪( i.e., 10.0.0.0/16).

2. Ensure Network Isolation:

- During VPC creation, do not attach an Internet Gateway to keep the network isolated from the public internet.

3. Create Subnets:

- Within the VPC, create two private subnets in different availability zones.(Also we can create a public subnet within the VPC)

- Specify the CIDR blocks for each subnet (e.g., 10.0.1.0/24, 10.0.2.0/24).

4. Configure Routing Tables:

- Create a custom route table for each subnet.

- Define routes to enable communication within the VPC and between subnets.

5. To ensure that the network is secure and isolated from the public internet

- We created a NAT gateway and routed into the private subnet

6. Provision Remote Machines (EC2 Instances):

- Navigate to the EC2 dashboard.

- Launch two EC2 instances: one for the web application and one for the PostgreSQL database.

7. Launch EC2 Instance for web application:

- Choose an Amazon Machine Image (AMI) for each instance, such as Amazon Linux or Ubuntu.

- Select the instance type based on the requirements of your applications.

- Configure instance details, including selecting the VPC and subnet for each instance.

- Since as per the requirement instances had to be with the same private subnet, hence we assigned the same private subnet of same region to both the EC2 instances

- Configure security groups for each instance to control inbound and outbound traffic.

8. Install Web Application:

- For one instance to install web application we added a security rule with type ***http port 80.***

*Now to install the web application from backend:*

*Below are the steps->*

Step1: Login to SSH Client.

Step 2: Commands to execute.

***sudo su***

***yum update -y***

***yum install httpd -y***

***service httpd start***

***chkconfig httpd on*** (To configure the web server to restart if it gets stooped)

Step 3: Create an index.html file in the path -> ***/var/www/html***

Step 4: To launch any webpage, we will have edit the index.html file with the html script to route to html page

9. Installation of PostgreSQL:

- Choose an Amazon Machine Image (AMI) for each instance, such as Amazon Linux or Ubuntu.

- Select the instance type based on the requirements of your applications.

- Configure instance details, including selecting the VPC and subnet for each instance.

- Since as per the requirement instances had to be with the same private subnet, hence we assigned the same private subnet of same region to both the EC2 instances

- Configure security groups for each instance to control inbound and outbound traffic.

- Now create the RDS for PostgreSQL type and connect it to the above mentioned instance.

10. Ensure Communication Between Instances:

- Ensure that both EC2 instances are in the same private subnet.

- Update security group rules to allow inbound traffic from the web application instance to the PostgreSQL database instance on the PostgreSQL port (default is 5432).

- Configure the PostgreSQL instance to accept connections from the IP address of the web application instance.

11. Test Communication:

- Test the communication between the web application and the PostgreSQL database to ensure they can communicate successfully within the private network.

**Result:** From the above steps, we have successfully set up a private network in AWS VPC and I have provisioned EC2 instances for a web application and PostgreSQL database, and ensured they can communicate securely within the private subnet.

* Install and configuring the web application and PostgreSQL database, as well as setting up routing via an Application Load Balancer (ALB) and DNS for the application:

1. Web Application Installation and Configuration:

Step 1: SSH into the first EC2 instance designated for the web application.

Step 2: Install necessary dependencies for your web application, such as Node.js or Python Flask, and any other libraries or frameworks required.

Step 3: Clone or copy your web application codebase onto the EC2 instance.

Step 4: Configure environment variables, including database connection strings, API keys, and any other settings required for the web application to function properly.

Step 5: Set up a reverse proxy server (e.g., Nginx) to forward incoming requests to your web application.

Step 6: Test the web application to ensure it's functioning correctly on the EC2 instance.

2. PostgreSQL Installation and Configuration:

Step 1: SSH into the second EC2 instance designated for the PostgreSQL database.

Step 2: Install PostgreSQL using the package manager or by downloading and installing it manually.

Step 3: Configure PostgreSQL by editing the postgresql.conf and pg\_hba.conf files to specify settings such as data directory, authentication methods, and listen addresses.

Step 4: Create a new database and user for the web application to use. Grant necessary privileges to the user.

Step 5: Optionally, configure additional settings such as replication, logging, and security measures based on your requirements.

Step 6: Start the PostgreSQL service and ensure it's running properly.

3. Connection between Web Application and PostgreSQL Database:

Step 1: Update the web application's configuration to use the PostgreSQL database, providing the correct host, port, database name, username, and password.

Step 2: Test the connection between the web application and the PostgreSQL database to ensure they can communicate successfully.

4. Routing with ALB and DNS Setup:

Step 1: Set up an Application Load Balancer (ALB) in front of the web application EC2 instance(s) to distribute incoming traffic.

Step 2: Configure the ALB to route incoming requests to the EC2 instances hosting the web application.

Step 3: Obtain a domain name for your application (e.g., example.com) and configure DNS settings to point to the ALB's DNS name.

Step 4: Test the DNS configuration to ensure the domain name resolves to the ALB's DNS name and that requests are routed correctly to the web application.

Step 5: Optionally, configure SSL/TLS certificates for secure HTTPS communication between clients and the ALB.

**Result**: From the above steps, we have successfully installed, configured, and connected the web application and PostgreSQL database, as well as set up routing via an ALB and DNS for your application.