Lab 3 – AWS – VPC Peering

**Goal** – Establish point-to-point access from VPC A to VPC B

**Task** – Create a peering connection and VPC routes so EC2 Instance-A can ping Instance-B

**Validation** – Confirm point-to-point VPC connectivity using ping.

**Introduction**

In this task, there are multiple VPCs in the same region that have one instance each. VPC peering and the appropriate VPC routes are already configured for VPC-B to VPC-C. VPC peering and VPC routes will need to be configured between VPC-A and VPC-B. Then traffic will be tested to confirm what traffic routing is and is not supported with VPC peering.

**Topology**

A screenshot of a computer

AI-generated content may be incorrect.

**Pre-Work:**

* Delete the FGTHA stack by navigating to the CloudFormation console

A screenshot of a computer

AI-generated content may be incorrect.

Graphical user interface, text

AI-generated content may be incorrect.

* Create an IAM role prior to deploying the CloudFormation template by navigating to the IAM console. Once there click on **Access management -> Roles -> Create role**

A screenshot of a computer

AI-generated content may be incorrect.

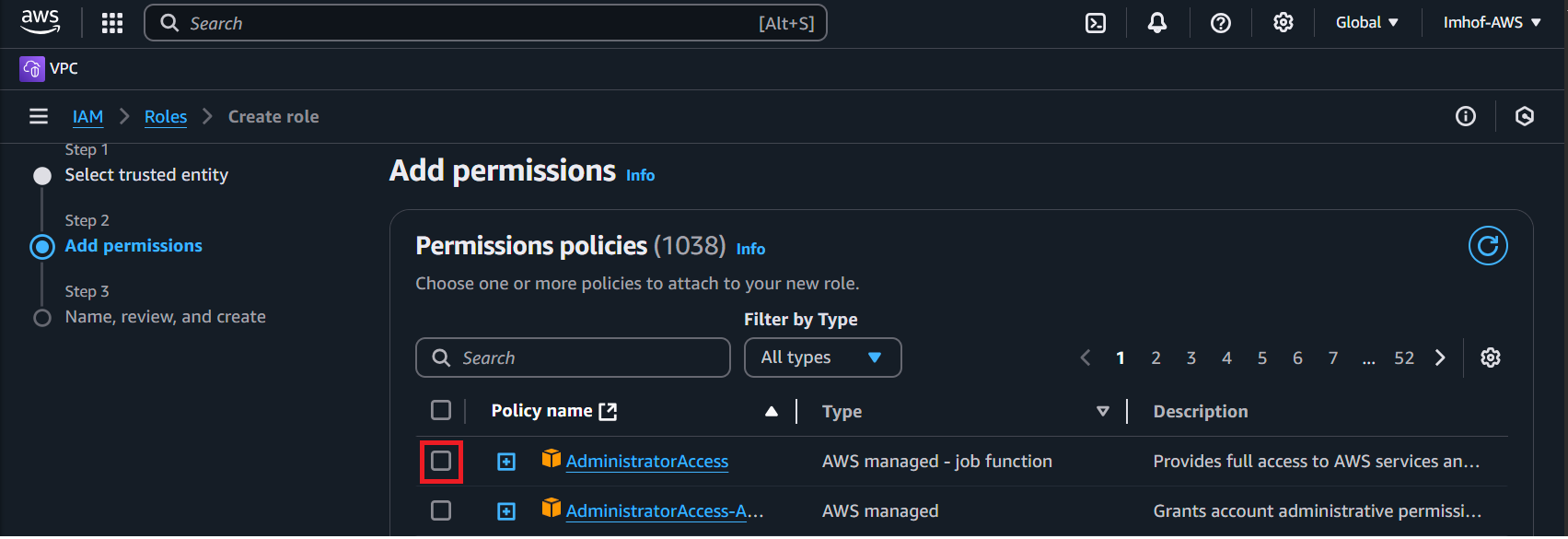
Use the following parameters for your test user:

* Click on **AWS service**
* Use case: **CloudFormation**
* Click on **Next**

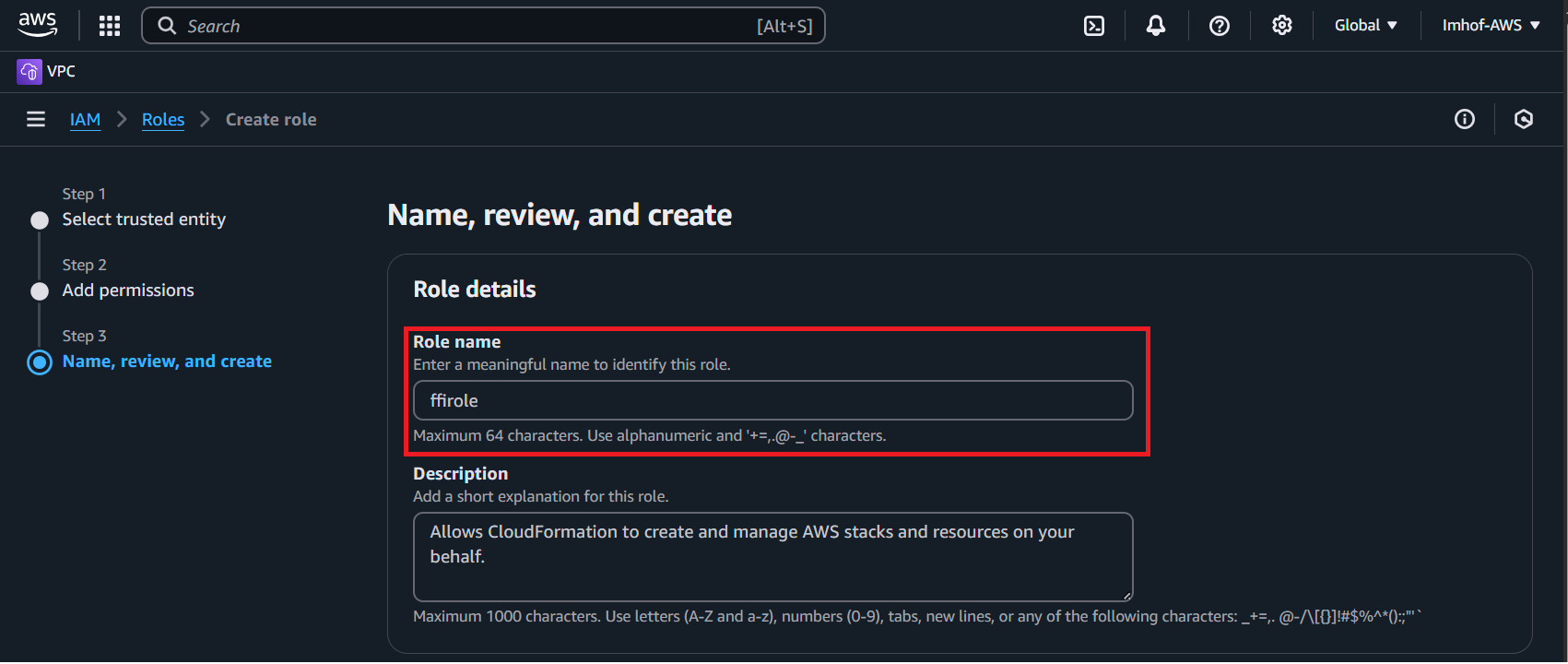
A screenshot of a computer

AI-generated content may be incorrect.

* Select **AdministratorAccess** for the role
* Click on **Next**



* Role name: **ffirole**
* Click on **Create role**



1 – Create a new stack by navigating to the CloudFormation console and importing the VPCpeering.json file:

* Click on **Create stack**
* Select **With new resources**

A screenshot of a computer

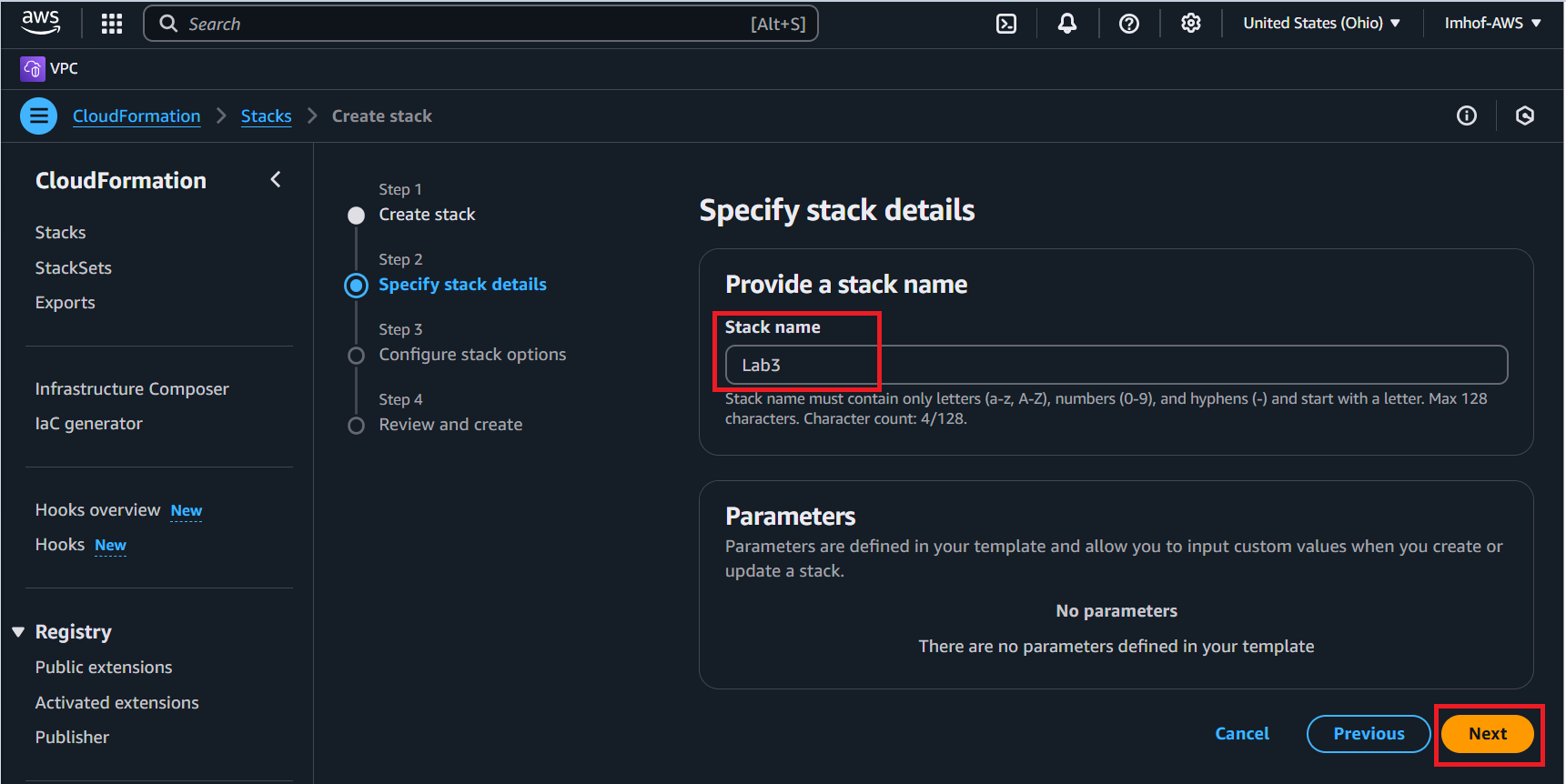
AI-generated content may be incorrect.

* Click **Choose an existing template**
* Choose **Upload a template file**
* Choose **VPCpeering.json** as the template file
* Click **Next**

A screenshot of a computer

AI-generated content may be incorrect.

* Name the stack as **Lab3**
* Click **Next**



* IAM Role: **ffirole**
* Behavior on provisioning failure: **Roll back all stack resources**
* Delete newly created resources during a rollback: **Delete all newly created resources**

A screenshot of a computer

AI-generated content may be incorrect.

**Acknowledge the capabilities** and then click **Next**

A screenshot of a computer

AI-generated content may be incorrect.

**Review the template parameters then click on Submit.**

It will take 5-10 minutes for the template to be fully deployed. Once it is finished the main/root stack will show as complete:

A screenshot of a computer

AI-generated content may be incorrect.

2 – Find EC2 Instance-A and verify that it cannot access Instance-B or C

* Navigate to the EC2 console
* Find Instance-A and connect to it using the following serial console directions:
  + Select the instance.
  + Click **Connect > EC2 serial console**.
  + **Copy the instance ID** as this will be the username and click connect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

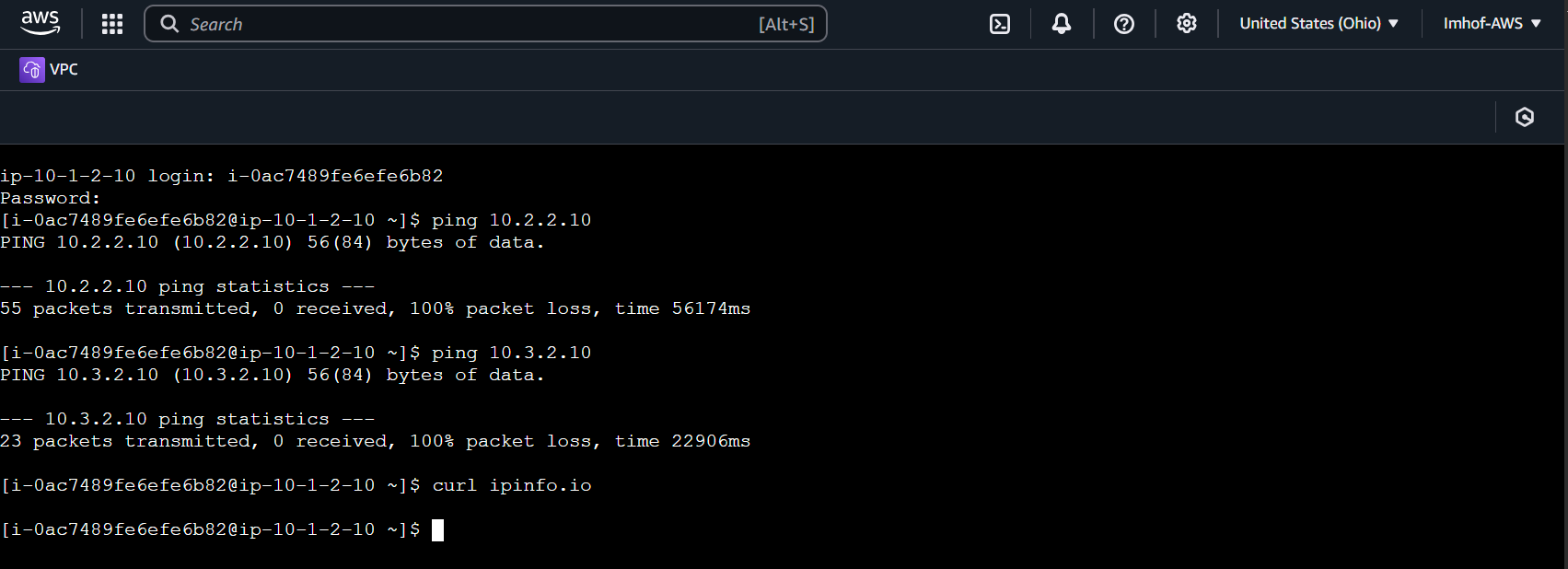
AI-generated content may be incorrect.

* Login to the EC2 instance:
  + You may need to hit enter to get a login prompt
  + Username: **«copied Instance ID from above»**
  + Password: **FORTInet123!**

Graphical user interface, application

AI-generated content may be incorrect.

* Run the following ping tests, they should all fail (Ctrl+C terminates the command):
  + **ping 10.2.2.10**
  + **ping 10.3.2.10**
  + **curl ipinfo.io**



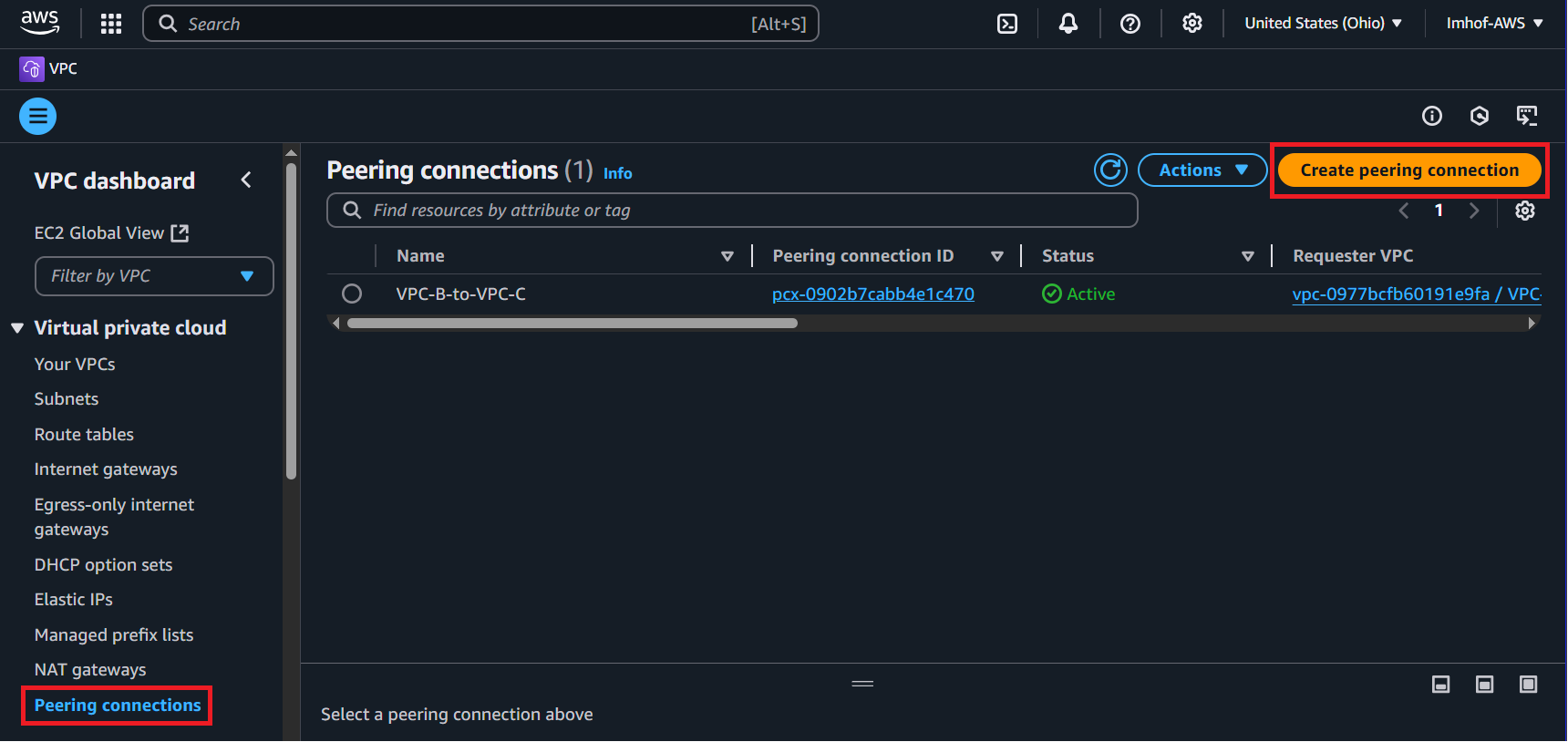
* Run the following commands to note the instance IPv4 address, default route, and gateway IP. Note that the default route points to 10.1.2.1, this is the AWS intrinsic router.
  + **ifconfig ens5**
  + **route -n**

A screenshot of a computer

AI-generated content may be incorrect.

3 – Create a VPC peering connection and VPC routes to allow Instance-A in VPC-A to reach Instance-B in VPC-B

* Navigate to the VPC console then go to the Peering connections page and click **Create peering connection**:



* Name: **A-to-B**
* VPC ID (Requester): **VPC-A**
* VPC ID (Accepter): **VPC-B**
* Click **Create peering connection**

A screenshot of a computer

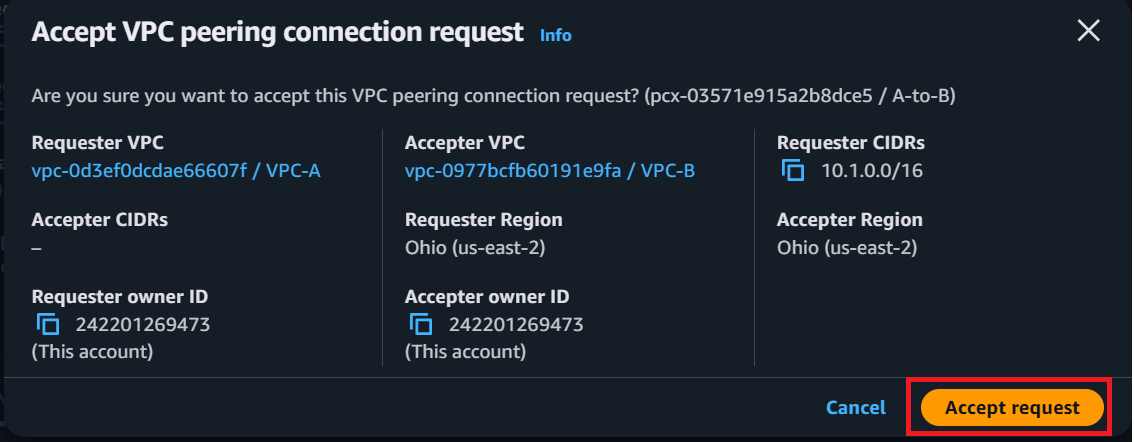
AI-generated content may be incorrect.

* Click on **Actions**
* Select **Accept request**

Graphical user interface, text, application

AI-generated content may be incorrect.

* **Accept request** on the pop-up window



* Navigate to the Route tables page and edit the VPC-A-PrivateRouteTable.
  + Click on **Routes**
  + Click on **Edit Routes**

A screenshot of a computer

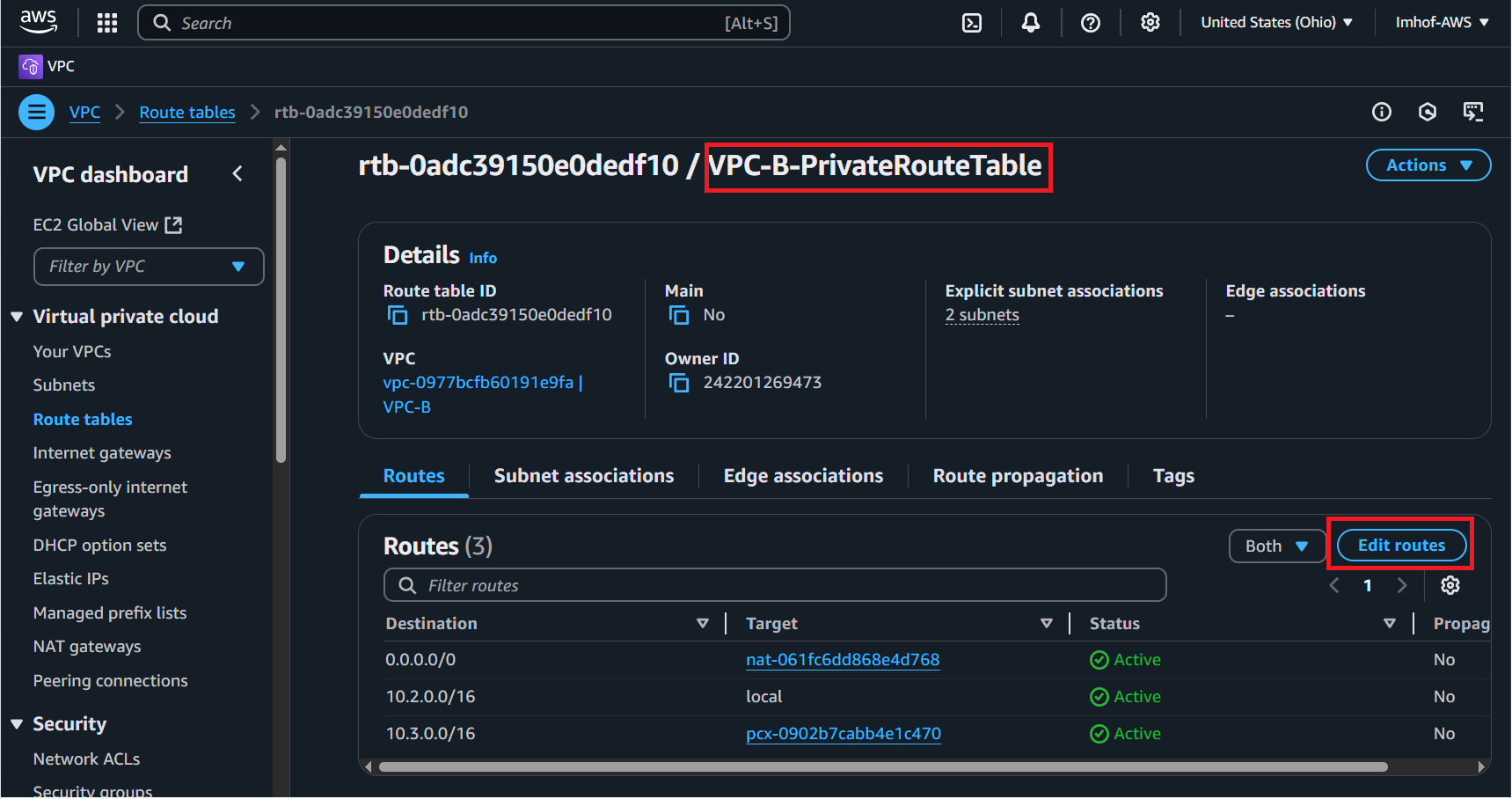
AI-generated content may be incorrect.

* Destination: **0.0.0.0/0**
* Target: **Peering Connection**
* Select **A-to-B**
* Click **Save changes**

A screenshot of a computer

AI-generated content may be incorrect.

* Edit the VPC-B-PrivateRouteTable to allow reply traffic:



* Destination: **10.1.0.0/16**
* Target: **Peering Connection**
* Select **A-to-B**
* Click **Save changes**

A screenshot of a computer

AI-generated content may be incorrect.

4 – Verify communication over the VPC Peering connection

* Navigate to the EC2 console.
* Access Instance-A using the serial console.
* Run the following command to verify connectivity to Instance-B:
  + **ping 10.2.2.10**

Graphical user interface

AI-generated content may be incorrect.

**Discussion Points:**

* VPC peering is a point to point connection only (no transitive routing).
* Full mesh is required to connect all VPCs together.
  + For example, connecting 10 VPCs would require (10\*9)/2 = 45 connections.
* VPC peering supports connections between VPCs:
  + In the same or different AWS Accounts
  + In the same (intra) or across (inter) regions
* When using inter-region peering, encryption is used.
* No Internet Gateway is required for VPC Peering (even when inter-region)
* Jumbo frames (9001 bytes) are only supported for intra-region connections, inter-region is limited to 1500 bytes.
* The VPC peering connection is at the **VPC level**. This means the VPC peering connection is not directly tied to any VPC subnet or route table explicitly. AWS routing for VPC peering connections will only deliver traffic to an IP address that is within the destination VPC CIDR. The routes you created in both VPC-A and B’s private route tables only direct traffic out of the local VPC to the target destination VPC.

Text

AI-generated content may be incorrect.

Step-by-step packet handling for the traffic from Instance-A to Instance-B:

Text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Step-by-step packet handling for the traffic from Instance-B to the internet:

Text, table

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.