**VPC PEERING IN AMAZON WEB SERVICES (AWS)**

**Introduction to VPC Peering:**

Virtual Private Cloud (VPC) peering is a networking connection between two VPCs that enables them to communicate securely as though they are part of the same network. This allows resources within each VPC to interact with one another seamlessly, facilitating data sharing, application deployment, and resource utilization across multiple environments.

**Benefits:**

1. Simplified Networking: VPC peering simplifies network architecture by enabling direct communication between VPCs without the need for complex configurations or additional hardware.
2. Resource Sharing: It allows resources, such as instances, databases, and services, to communicate across VPC boundaries, facilitating collaboration and resource utilization.
3. Cost-Effective: By leveraging existing AWS infrastructure, VPC peering eliminates the need for costly external connections or dedicated networking hardware.
4. Improved Security: Traffic between peered VPCs remains private and isolated from other networks, enhancing data security and compliance.
5. Scalability: VPC peering supports horizontal scaling by allowing resources to be distributed across multiple VPCs, accommodating growth and evolving business needs.

In this document we will see how to connect two VPC in AWS using VPC peering. When we need to connect two VPC, we can use VPC Peering, when we need to connect more than two VPC we need to do it by Transit Gateway. But in this document, we will see only VPC Peering.

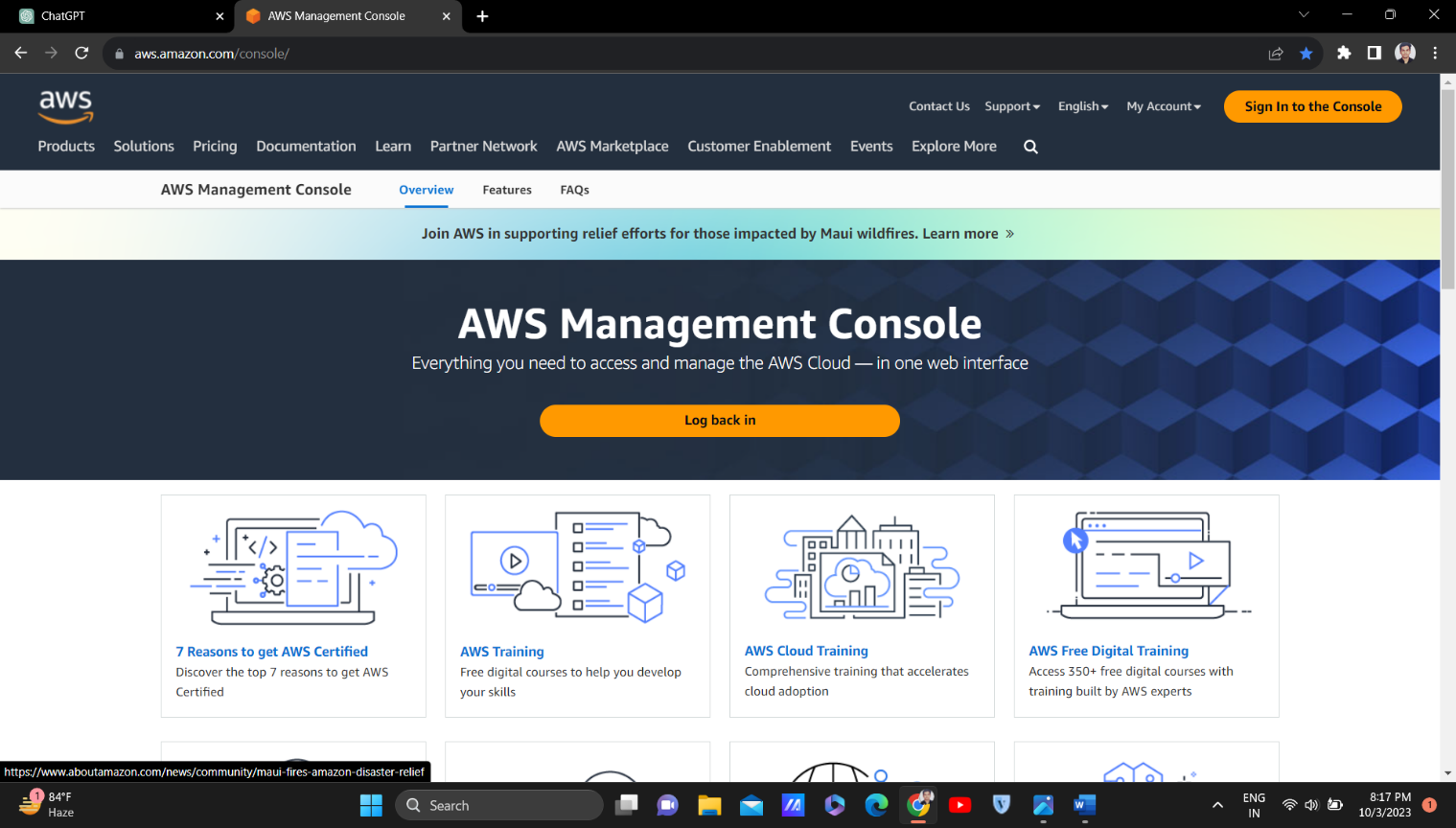
**Prerequisites for Setting up VPC Peering Connections:**

1. **AWS Account**: Access to an AWS account with appropriate permissions to create and manage VPC resources.

**STEP 1. Login to AWS console**

* <https://aws.amazon.com/console/> using this website login to AWS Console, if you do not

have an AWS account please sign up first and then login.



**STEP 2. Create VPC**

* After logging into the AWS account, you will see the home page of AWS Console.
* On the left top there is a search bar.
* In the search bar search for VPC and press enter button.
* You will see the interface as shown in fig 2.1
* Now click on Create VPC

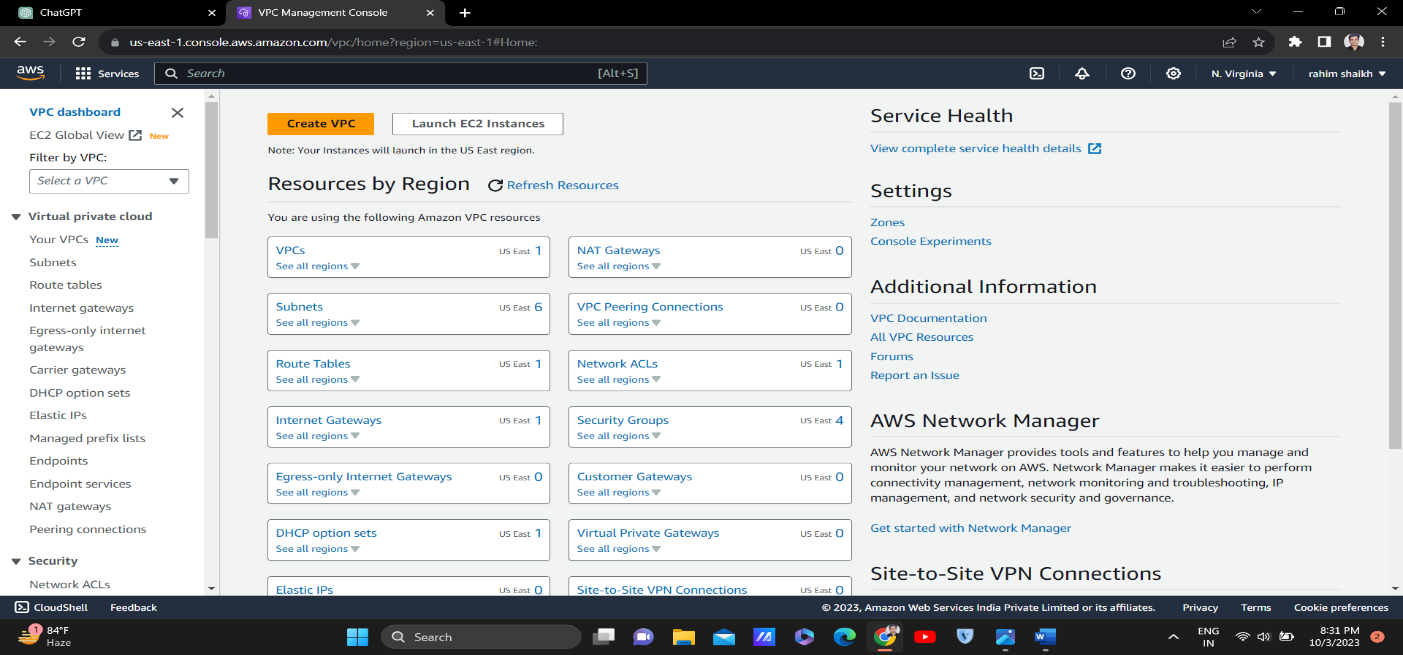


Fig 2.1

* After you click on create VPC, there will be two options VPC only or VPC and more, click on VPC only.
* Now give a name to your VPC by your choice (for eg. Testvpc1)
* Now in IPv4 CIDR type this 10.0.0.0/16
* Rest other settings keep default do not change any of the settings.
* Then click on Create VPC.
* Now repeat the above process and create another VPC (for eg. Testvpc2), but make sure that give a different IPv4 CIDR in this VPC (for eg. 10.0.0.0/24) to avoid  overlapping CIDRs.
* Now go in subnet, click on create subnet, give the vpc id Testvpc1 and give the name of subnet and create a subnet in that vpc.
* Similarly create one subnet by selecting Testvpc2.

**STEP 3. Create EC2 Instance**

When setting up VPC peering, deploying instances in both VPCs allows you to test and verify connectivity between the two environments, ensuring resources in one VPC can access those in the other. It also facilitates network testing and troubleshooting, enabling you to diagnose and resolve any connectivity issues more effectively. Overall, deploying instances in both VPCs enhances the effectiveness and reliability of your VPC peering setup.

* After creating 2 VPC now its time to create 2 EC2 Instances.
* Search for EC2 and press enter, you will see the interface as shown in fig 3.1
* Now click on Launch Instance
* Now name your instance (for eg. Testec21)
* Now in Application and OS image, select Amazon Linux.
* In Amazon Machine Image, select Amazon Linux 2023 AMI free eligible tier. Remember, always select the free tier to avoid getting charged for EC2 instance.
* In Instance type, select t2 micro free eligible tier. Remember, always select the free tier if you do not have a heavy task to perform. Amazon will charge for other instance types except the free ones.

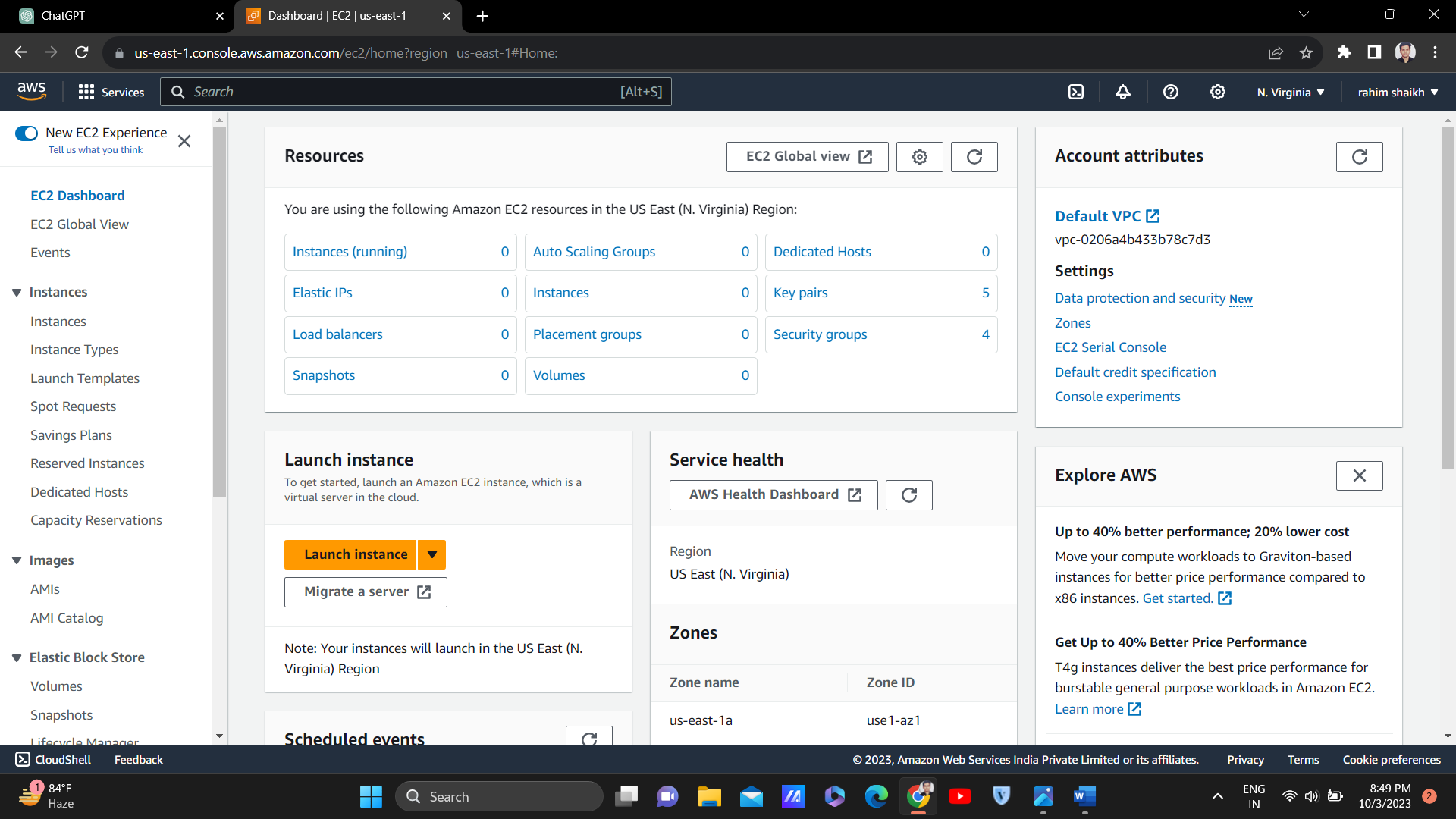
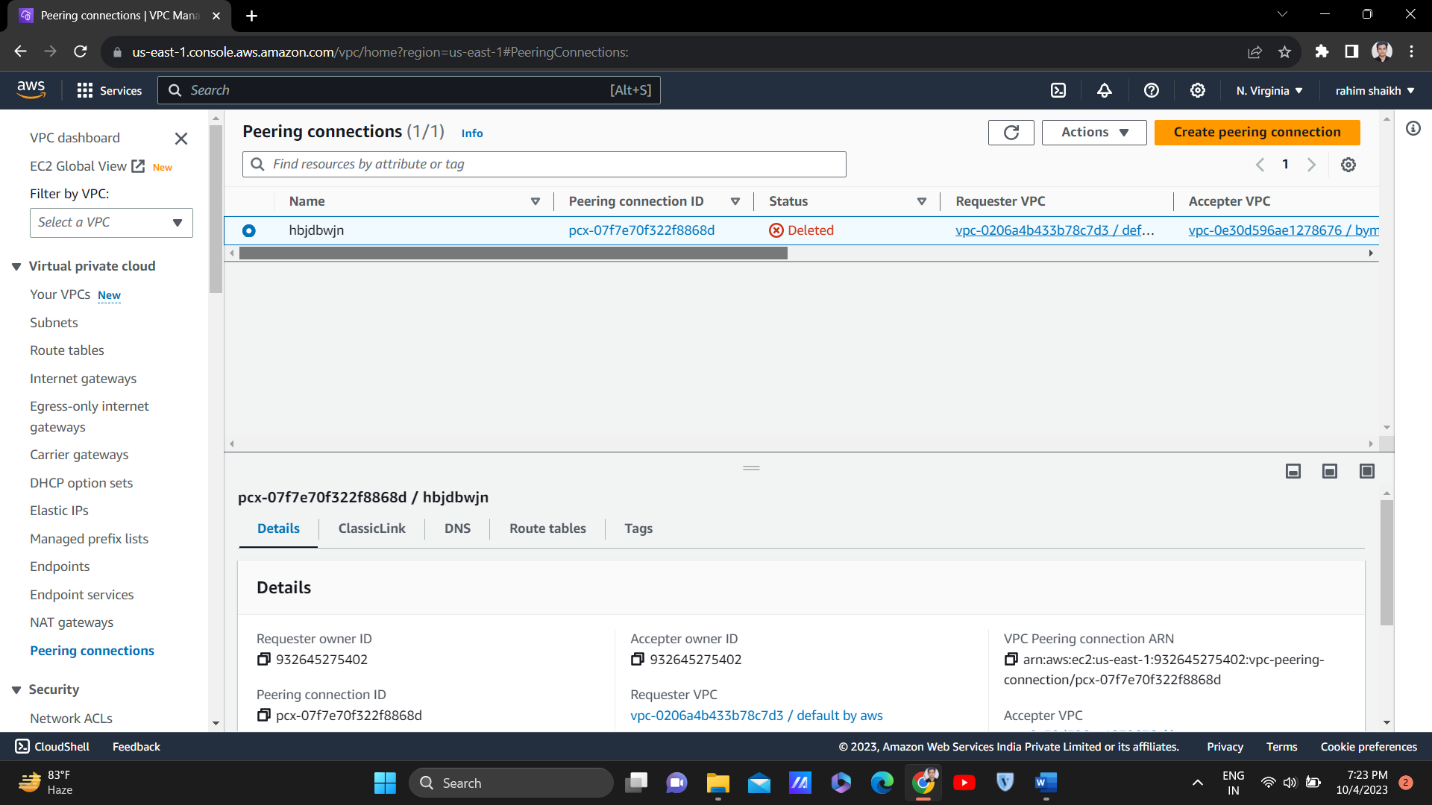


Fig 3.1

* Now in Key Pair, click on create on a new key pair, name the key pair (for eg. ec21key), in key pair type click RSA and in Private key file format click on .ppk and then click on Create Key Pair. After creating, your key pair will automatically get downloaded.
* In Network Settings, edit the network settings and in VPC select the Testvpc1 which we have created before. Make sure that Auto assign public IP is enabled. Rest of the setting keep by default.
* Now click on Launch Instance and your Testec21 instance will be created
* Now in the same way create one more ec2 instance (for eg. Testec22), but make sure that in network setting for this ec2, in VPC section select another VPC that is Testvpc2 which we have created before.

**STEP 4. Create a VPC Peering Connection**

* After successfully creating VPC and EC2, now it is time to create Peering Connection between the VPC that we have created.
* Search VPC and press Enter, now go in peering connection.
* You will see the below interface once you go in peering connection.



* Now click on Create peering connection on the top right corner.
* Name your peering connection (for eg. peering1)
* In VPC Id (Requester), select any one VPC (for eg Testvpc1)
* In VPC Id (Accepter), select another VPC (for eg Testvpc2)
* Keep other settings same if the VPC is in same account and in same region, if not then change the settings accordingly.
* Now click on create peering connection.
* As soon as the peering connection is created go in action and accept the request.

**STEP 5. Modifying Security Groups**

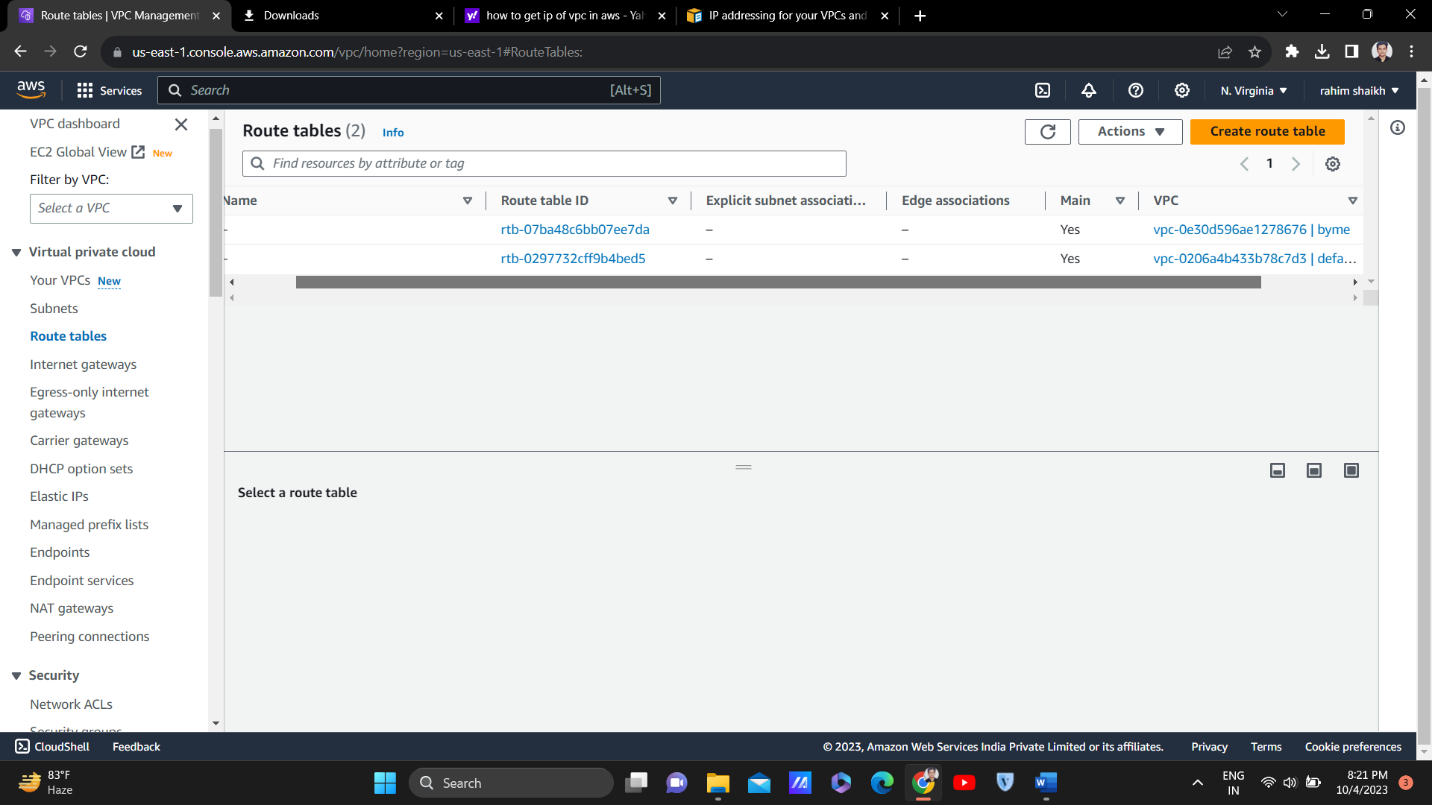
Modifying the security groups of instances during VPC peering is crucial to regulate and facilitate the flow of traffic between resources across the peered VPCs. Security groups serve as virtual firewalls controlling inbound and outbound traffic for instances, and by adjusting their rules, you can ensure that the necessary communication between instances in different VPCs is permitted while maintaining security. This modification allows you to enforce specific access control policies, aligning with security requirements and compliance standards. Additionally, it enables you to accommodate application requirements, ensuring that instances can communicate effectively across VPC boundaries. Whether for enforcing policies, accommodating application needs, or troubleshooting connectivity issues, modifying security groups provides the flexibility needed to manage access between peered VPCs efficiently.

* Go in VPC, copy the IPv4 CIDR of Testvpc2.
* Go in EC2 > go in instance id of Testec21 > scroll down and go in security > click on the security group linked with it.
* Now edit inbound rules > Add rule > in type select All Traffic in IP paste the Testvpc2 IPv4 CIDR copied before and save rule.
* Now repeat the same process, copy IP of Testvpc1 > go to instance id of Testec22> go in security groups> edit inbound rules> add rule > All traffic > paste the IP of Testvpc1 > save rule.

**STEP 6. Modifying Route Table**

Modifying the route table of a VPC during VPC peering is essential to establish proper routing between the peered VPCs. Route tables determine how network traffic is directed within a VPC, specifying the destination for packets based on their destination IP addresses. By updating the route tables of each VPC involved in the peering arrangement, you ensure that traffic destined for the other VPC is correctly directed through the peering connection. This adjustment allows instances within each VPC to communicate with resources in the other VPC by routing traffic through the peering connection rather than relying on default routes or external gateways. Thus, modifying the route tables enables seamless and efficient communication between instances across the peered VPCs, enhancing the connectivity and functionality of the overall network architecture.

* Route table are automatically created as we create VPC
* Search VPC and press enter
* Go to route table you will see the below interface.



* Go in route table associated with Testvpc2 > click on edit routes > add route > in destination add the IPv4 CIDR of Testvpc1> in target add peering connection and select the peering connection you made> then save changes.
* Now repeat the same process, go in route table associated with Testvpc1 > click on edit routes > add route > in destination add the IPv4 CIDR of Testvpc2> in target add peering connection and select the peering connection you made> then save changes.

**STEP 7. Launch EC2 Instance**

* Copy the Private IP address of any of the instance (for eg Testec22)
* Now go in instance id of Testec21 and click on connect then scroll down and click on connect.
* As soon as you get connected type a command: ping ‘private IP’ and press enter.
* You can see the interface as shown in fig 7.1 which means that the connection between 2 VPC was established successfully.
* If you did not see the interface as shown below then it is possible that you have made mistake somewhere.
* Repeat the process again if your connection was not established successfully

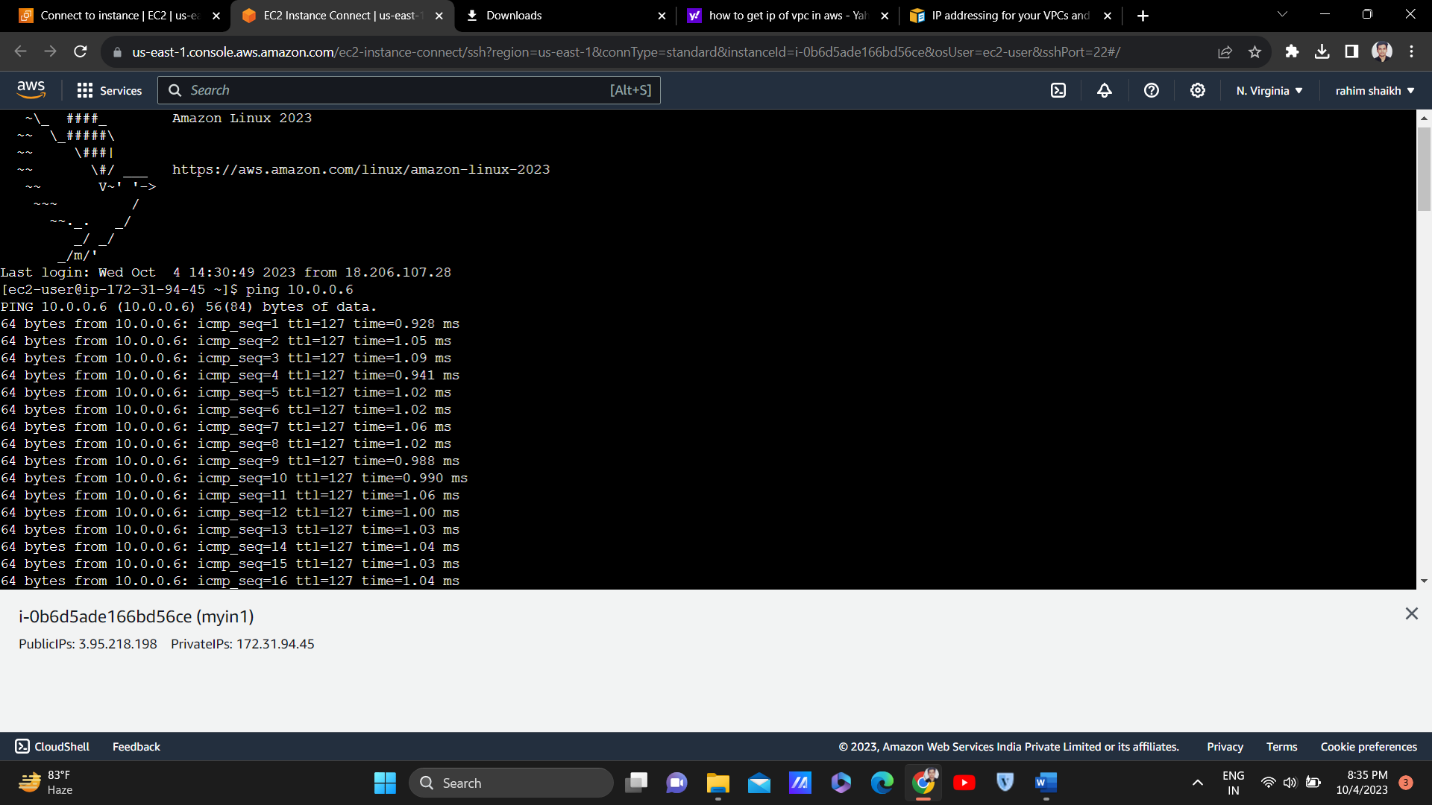


Fig 7.1

By following these steps, you can establish VPC peering connections between two VPCs using the AWS Management Console. VPC peering enables seamless communication between resources in different VPCs, enhancing the flexibility and scalability of your AWS infrastructure.

**Best Practices**

Best practices for configuring security groups, route tables, and network ACLs for optimal communication between peered VPCs involve implementing secure, efficient, and granular controls to manage traffic flow effectively while maintaining a high level of security. Here are some key recommendations:

1.Security Groups:

Assign security groups to resources based on their function and access requirements, following the principle of least privilege.

Define inbound and outbound rules that allow only necessary traffic between peered VPCs, such as specific ports or protocols required for application communication.

Regularly review and update security group rules to align with changing application requirements and security policies.

2.Route Tables:

Update route tables in each VPC to include routes directing traffic destined for the peered VPC's CIDR block through the VPC peering connection.

Avoid overlapping CIDR blocks between peered VPCs to prevent routing conflicts.

Consider implementing custom routes or route priorities to control traffic routing more effectively, especially in complex network architectures with multiple route tables.

3.Logging and Monitoring:

Enable logging for security groups, route tables, and NACLs to capture and analyze traffic patterns, security events, and potential anomalies.

Implement centralized logging and monitoring solutions to gain visibility into network traffic and security posture across peered VPCs, facilitating timely detection and response to security incidents.

By adhering to these best practices, you can ensure that your security groups, route tables, and network ACLs are configured optimally to facilitate secure and efficient communication between peered VPCs while minimizing potential security risks and vulnerabilities.

**Troubleshooting tips and common issues encountered during VPC peering setup:**

When troubleshooting VPC peering setup, it's essential to understand common issues and employ effective strategies to resolve them promptly. Here are some troubleshooting tips and common issues encountered during VPC peering setup:

1.Verify Peering Status: Check the status of the VPC peering connection in the AWS Management Console to ensure it is in the "active" state. If the status is "pending" or "failed," investigate the reason for the failure, such as incorrect settings or connectivity issues.

2.Review Route Tables: Ensure that the route tables in each VPC are correctly configured to route traffic to the peered VPC. Check for any missing or incorrect routes that may prevent communication between instances.

3.Security Group Rules: Review the security group rules associated with instances in the peered VPCs. Ensure that the rules allow the necessary inbound and outbound traffic for the desired communication to occur.

4.Subnet CIDR Overlaps: Verify that the CIDR blocks of the subnets in the peered VPCs do not overlap. Overlapping CIDR blocks can cause routing conflicts and prevent successful communication between instances.

5.Check Logs and Monitoring: Utilize AWS CloudWatch Logs and VPC Flow Logs to monitor network traffic and identify any anomalies or errors. Analyze the logs to pinpoint the source of the issue and troubleshoot accordingly.

6.Consult AWS Documentation and Forums: If troubleshooting efforts are unsuccessful, refer to AWS documentation, FAQs, and community forums for additional guidance and insights from other AWS users who may have encountered similar issues.

By following these troubleshooting tips and addressing common issues encountered during VPC peering setup, you can effectively diagnose and resolve connectivity issues, ensuring seamless communication between peered VPCs within your AWS environment.