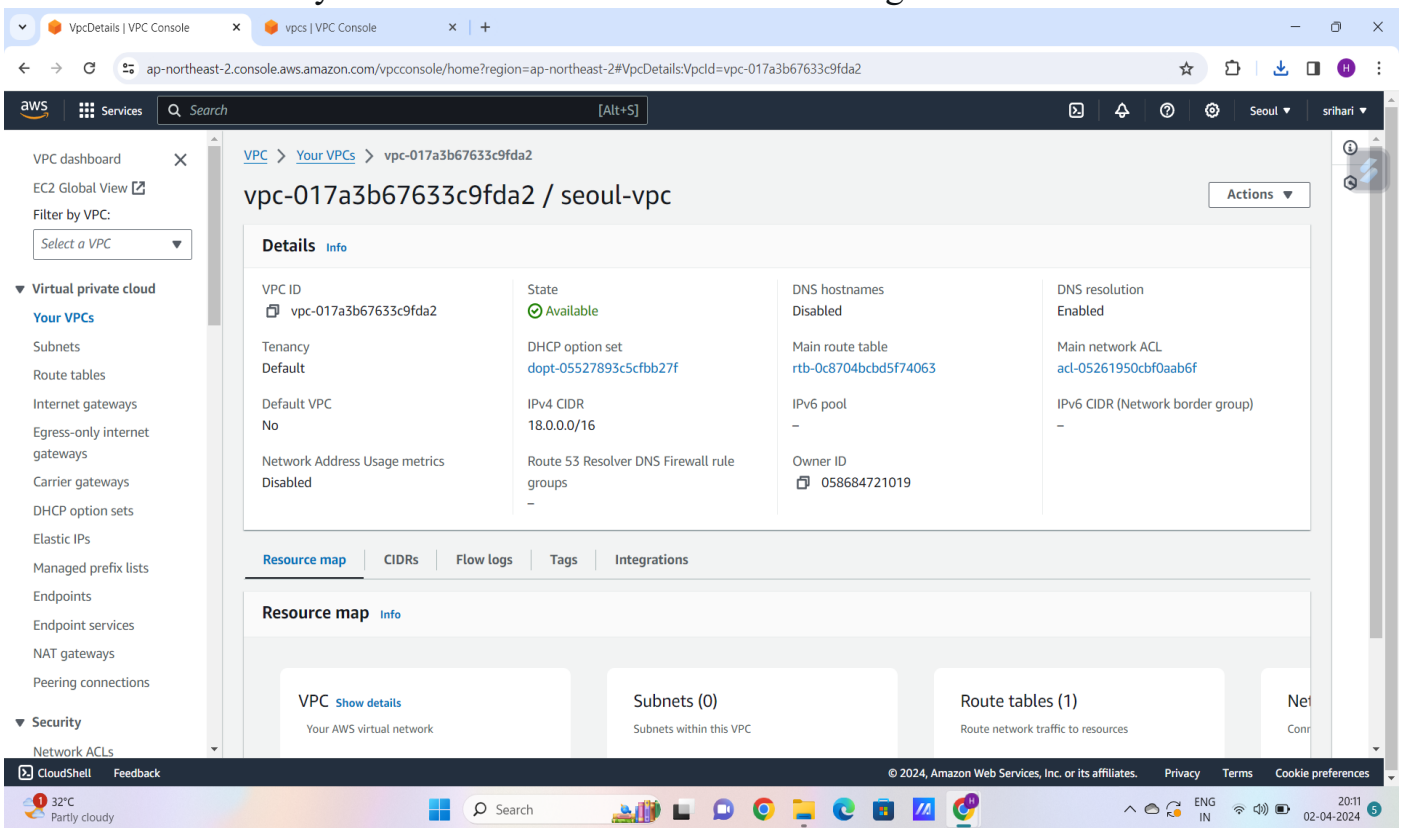


CONNECT TWO VPC'S IN DIFFERENT REGIONS THROUGH PEERING

By K.Manichandu

CREATING A PEERING CONNECTION BETWEEN TWO VPC's IN TWO DIFFERENT REGIONS

- Peering is the exchange of data directly between internet service providers, rather than via the internet.
- First we need two VPC's in two different regions and launch an EC2 instance in the respective subnets of the two regions.
- So firstly create two VPC's in two different regions.



- Here I created a vpc in seoul region and its name is Seoul-vpc with CIDR 18.0.0.0/16
- Now I have to create a subnet, route table and an internet gateway(igw) for this vpc in seoul.
- Now a subnet is created in this vpc named Seoul-subnet.

SubnetDetails | VPC Console

ap-northeast-2.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-2#SubnetDetails:subnetId=subnet-0a1f19b64ece2071b

Subnet-0a1f19b64ece2071b / seoul-subnet

Details

Subnet ID subnet-0a1f19b64ece2071b	Subnet ARN arn:aws:ec2:ap-northeast-2:058684721019:subnet/subnet-0a1f19b64ece2071b	State Available	IPv4 CIDR 18.0.0.0/16
Available IPv4 addresses 65531	IPV6 CIDR -	Availability Zone ap-northeast-2a	Availability Zone ID apne2-az1
Network border group ap-northeast-2	VPC vpc-017a3b67633c9fda2 seoul-vpc	Route table rtb-0c8704bcd5f74063	Network ACL -
Default subnet No	Auto-assign public IPv4 address No	Auto-assign IPv6 address No	Auto-assign customer-owned IPv4 address No
Customer-owned IPv4 pool -	Outpost ID -	IPv4 CIDR reservations -	IPv6 CIDR reservations -
IPv6-only No	Hostname type IP name	Resource name DNS A record Disabled	Resource name DNS AAAA record Disabled
DNS64 Disabled	Owner 058684721019		

- Now create an internet gateway and attach to this vpc.

Attach Internet gateway | VPC Console

ap-northeast-2.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-2#AttachInternetGateway:internetGatewayId=igw-02ff3709fae6a5161

The following internet gateway was created: igw-02ff3709fae6a5161 - seoul-igw. You can now attach to a VPC to enable the VPC to communicate with the internet.

Attach to VPC (igw-02ff3709fae6a5161)

VPC

Attach an internet gateway to a VPC to enable the VPC to communicate with the internet. Specify the VPC to attach below.

Available VPCs

Attach the internet gateway to this VPC.

Select a VPC

AWS Command Line Interface command

Cancel Attach internet gateway

- The attached igw to a vpc should be looking like this.

InternetGateway | VPC Console

ap-northeast-2.console.aws.amazon.com/vpcconsole/home?region=ap-northeast-2#InternetGateway:internetGatewayId=igw-02ff3709fae6a5161

igw-02ff3709fae6a5161 / seoul-igw

Details

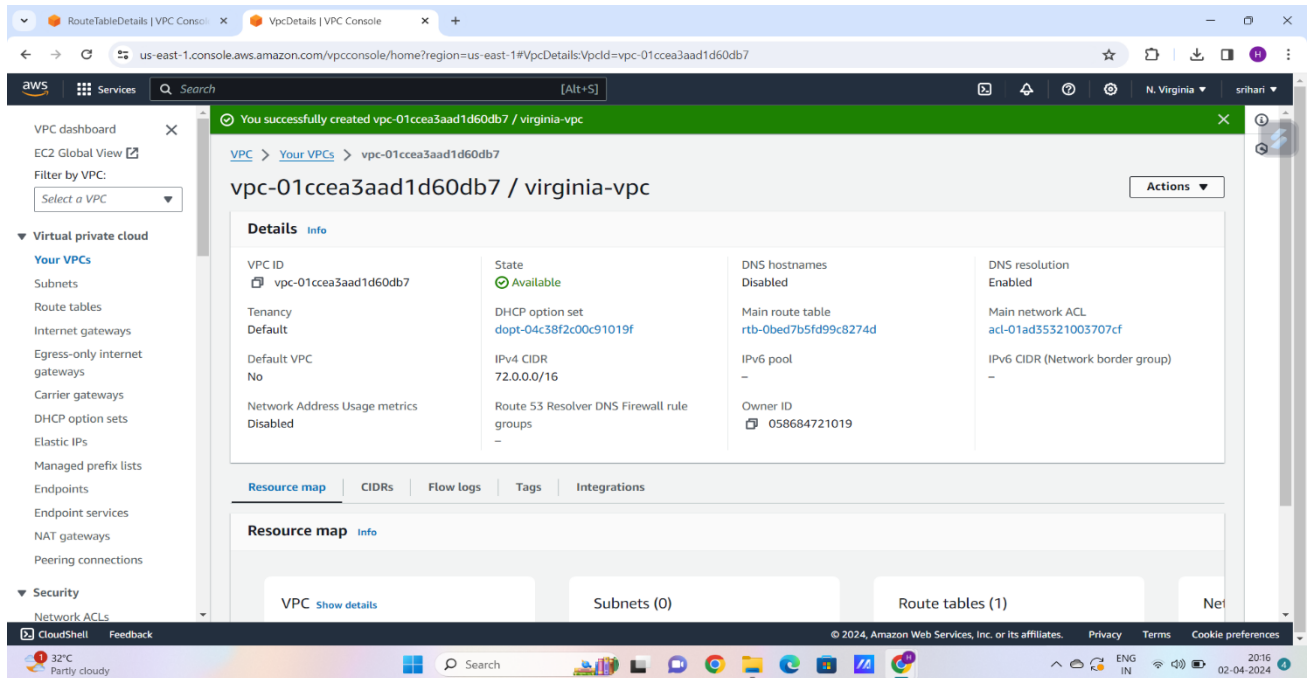
Internet gateway ID igw-02ff3709fae6a5161	State Attached	VPC ID vpc-017a3b67633c9fda2 seoul-vpc	Owner 058684721019
--	-------------------	---	-----------------------

Tags

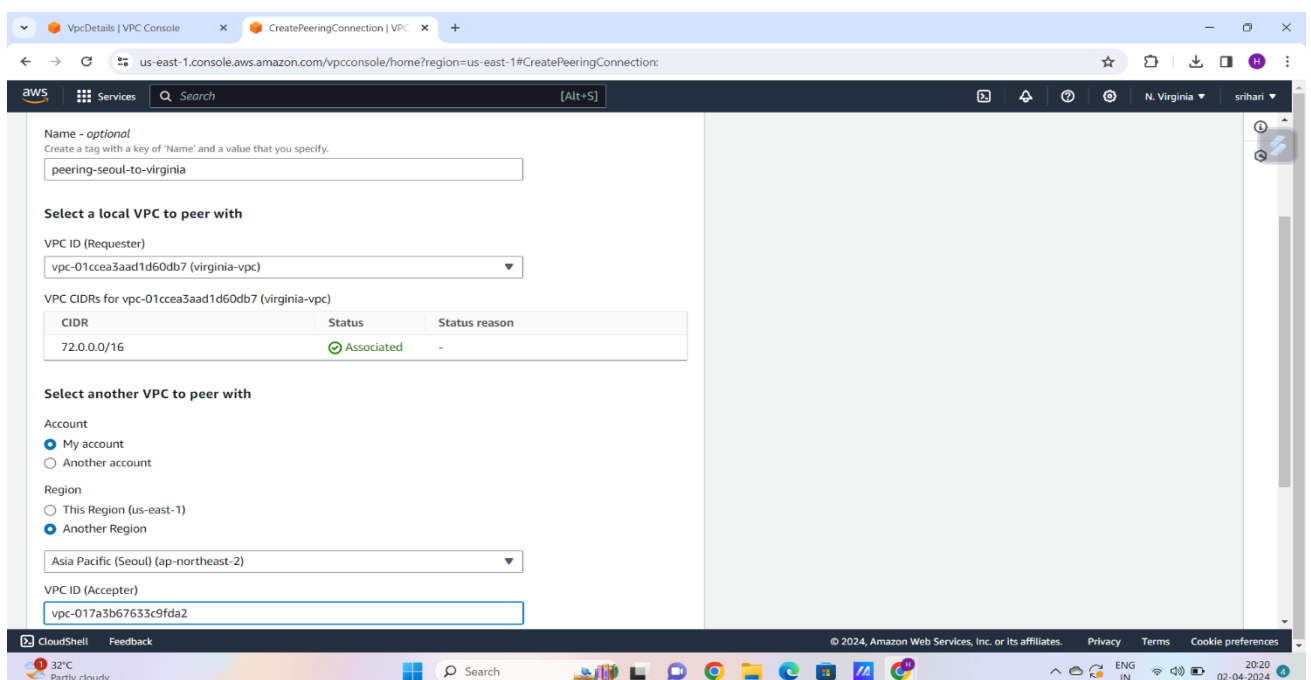
Search tags

Key	Value
Name	seoul-igw

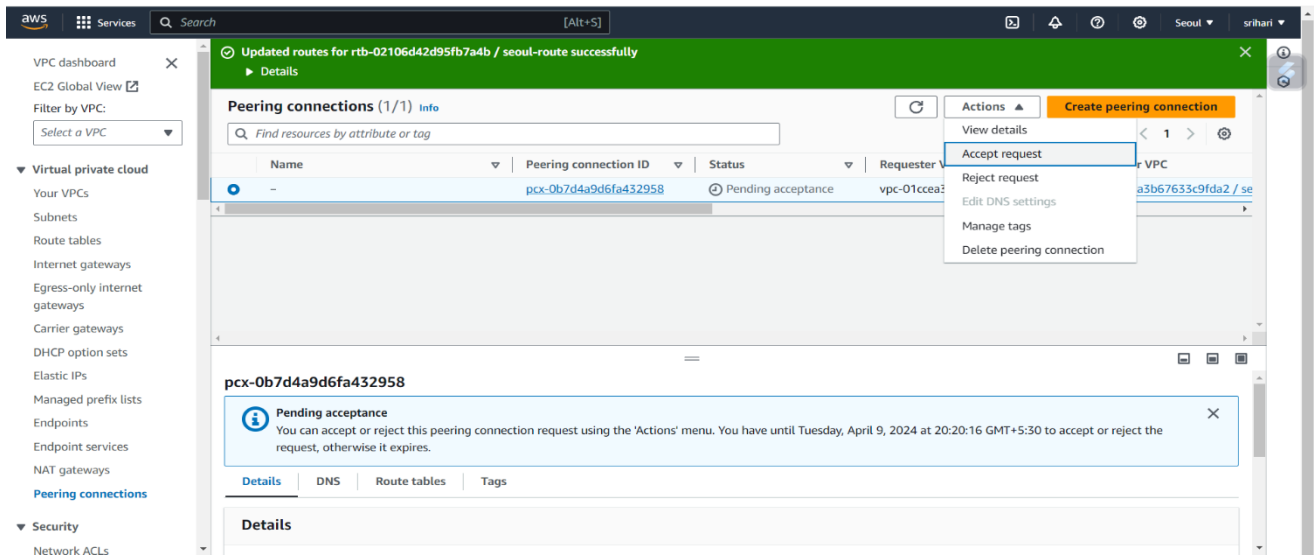
- Now create a route table for this subnet and associate the subnet and go to edit routes and add internet gateway connection in the route.
- Create another VPC in another region and repeat the same process for this VPC.



- Here I created a VPC in virginia region with CIDR 72.0.0.0/16
- Now add a peering connection for these two VPC's by going to the peering connections in VPC section.
- Here the requestor is the vpc you are present working and acceptor is the vpc in another region.

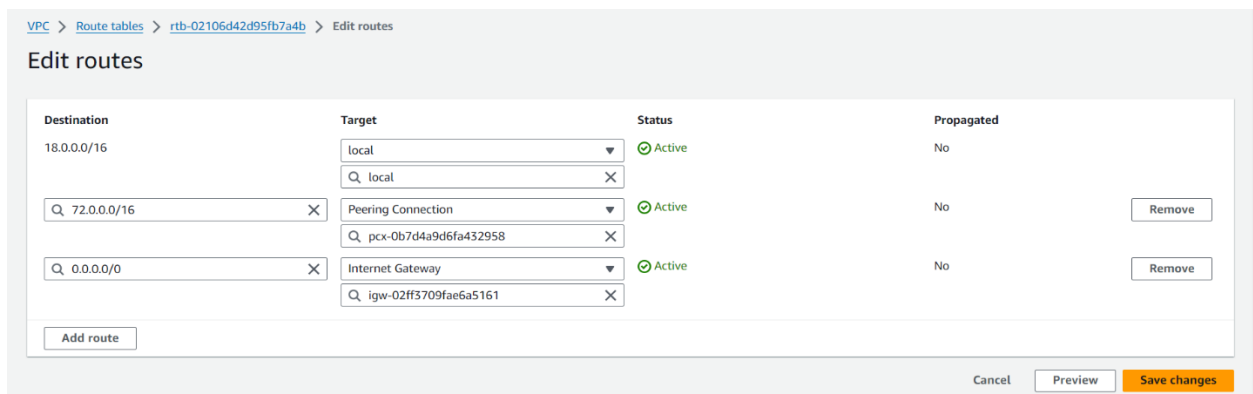
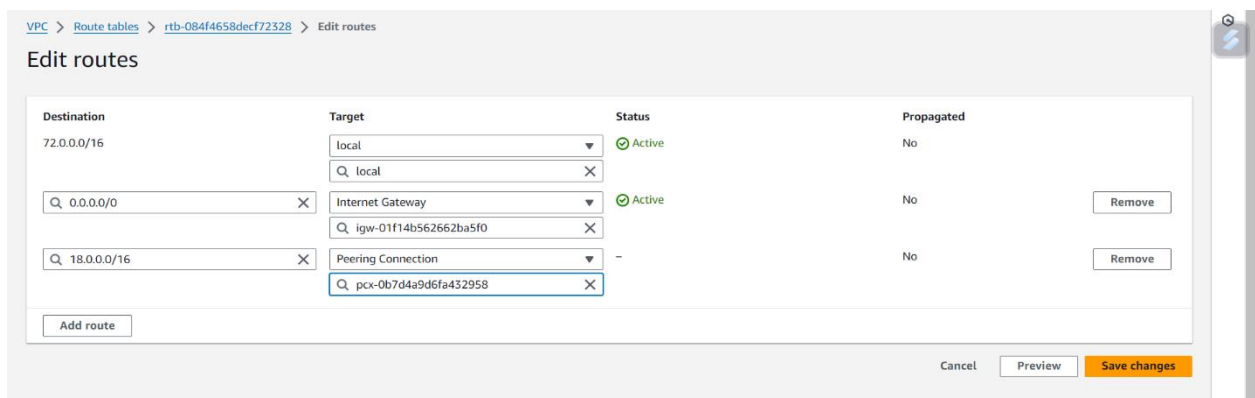


- Here select another region option for you to connect VPC in another region.
- Now the request to connect has been sent and go to another region and accept the peering connection.
- After you accept the request, you have to go the route tables of both

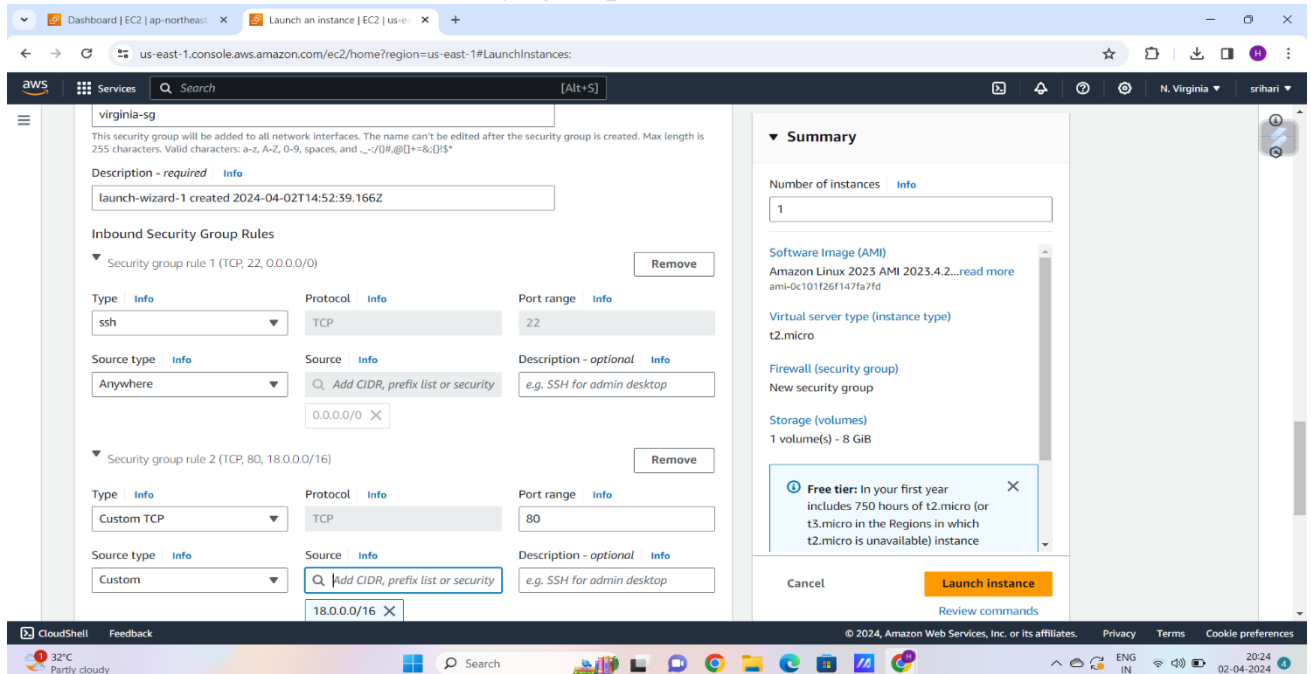


Subnets of VPC's and enable peering.

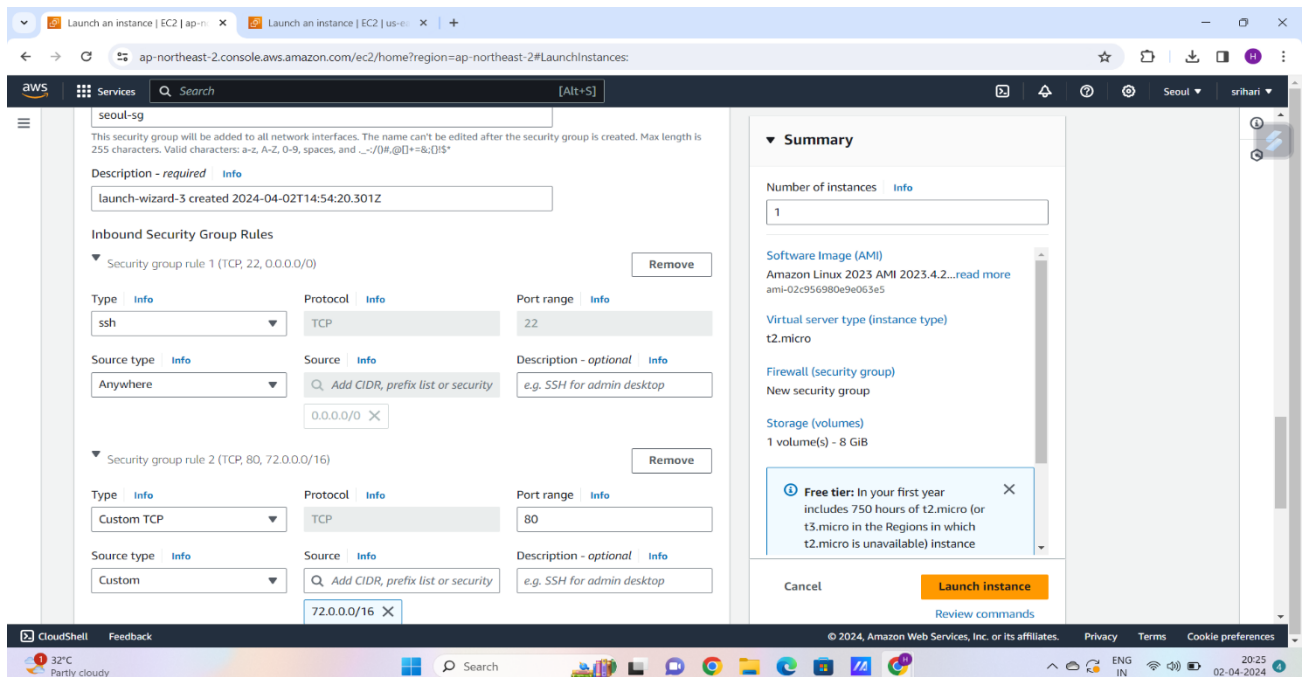
- To enable peering go to edit routes and give CIDR of other VPC as destination and peering as target.



- So as shown above both route tables has to be given these routes. Now the peering connection has been enabled.
- Now launch instances in both of the subnets in these two VPC's.
- Before launching these two instances create security groups for these instances and add a security group rule for these instances.



- Here the port range will be 80 for these two security groups and source type will be custom TCP and source will be the CIDR of the other VPC.



- After creating the security groups for both of these instances , launch these instances.

- Now connect to the instances and write the script as follows

```
Sudo -i
Yum install nginx -y
cd /usr/share/nginx/html
rm index.html
vi index.html
```

- The script above is given to install nginx in both of the ec2 instances and modify the content in the index.html.
- The last command is given to edit the index.html and you can write whatever you want in the index page.
- Now after completing this process copy the IP of seoul EC2 and give a curl operation in virginia EC2 with the seoul IP.
- It should display the index page of seoul EC2 and viceversa.
- The picture shows us the index of seoul EC2 in virginia EC2. And viceversa is achieved. Now PEERING is SUCCESSFUL between two VPC'S.

The image shows two screenshots of AWS terminal windows. The top screenshot shows the installation of nginx on the 'seoul' EC2 instance. The bottom screenshot shows the installation of nginx on the 'virginia' EC2 instance and a successful curl test from the virginia instance to the seoul instance.

```
aws Services Search [Alt+S] N. Virginia srihari
Running scriptlet: nginx-1:1.24.0-1.amzn2023.0.2.x86_64
Verifying : nginx-1:1.24.0-1.amzn2023.0.2.x86_64
Verifying : nginx-core-1:1.24.0-1.amzn2023.0.2.x86_64
Verifying : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
Verifying : libunwind-1.4.0-5.amzn2023.0.2.x86_64
Verifying : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch
Verifying : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
Verifying : nginx-filessystem-1:1.24.0-1.amzn2023.0.2.noarch
Installed:
generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 libunwind-1.4.0-5.amzn2023.0.2.x86_64
nginx-1:1.24.0-1.amzn2023.0.2.x86_64 nginx-core-1:1.24.0-1.amzn2023.0.2.x86_64 nginx-filessystem-1:1.24.0-1.amzn2023.0.2.noarch
nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch
Complete!
[root@ip-72-0-0-97 ~]# cd /usr/share/nginx/html
[root@ip-72-0-0-97 html]# rm index.html
rm: remove regular file 'index.html'? yes
[root@ip-72-0-0-97 html]# vim index.html
[root@ip-72-0-0-97 html]# systemctl restart nginx
[root@ip-72-0-0-97 html]# curl 18.0.140.47:80
this is seoul server
[root@ip-72-0-0-97 html]# exit
logout
[ec2-user@ip-72-0-0-97 ~]$ curl 18.0.140.47:80
this is seoul server
[ec2-user@ip-72-0-0-97 ~]$
```

```
aws Services Search [Alt+S] Seoul srihari
Running scriptlet: nginx-1:1.24.0-1.amzn2023.0.2.x86_64
Verifying : nginx-1:1.24.0-1.amzn2023.0.2.x86_64
Verifying : nginx-core-1:1.24.0-1.amzn2023.0.2.x86_64
Verifying : gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64
Verifying : libunwind-1.4.0-5.amzn2023.0.2.x86_64
Verifying : nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch
Verifying : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
Verifying : nginx-filessystem-1:1.24.0-1.amzn2023.0.2.noarch
Installed:
generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch gperftools-libs-2.9.1-1.amzn2023.0.3.x86_64 libunwind-1.4.0-5.amzn2023.0.2.x86_64
nginx-1:1.24.0-1.amzn2023.0.2.x86_64 nginx-core-1:1.24.0-1.amzn2023.0.2.x86_64 nginx-filessystem-1:1.24.0-1.amzn2023.0.2.noarch
nginx-mimetypes-2.1.49-3.amzn2023.0.3.noarch
Complete!
[root@ip-18-0-140-47 ~]# cd /usr/share/nginx/html
[root@ip-18-0-140-47 html]# rm index.html
rm: remove regular file 'index.html'? y
[root@ip-18-0-140-47 html]# vi index.html
[root@ip-18-0-140-47 html]# systemctl restart nginx
[root@ip-18-0-140-47 html]# curl 72.0.0.97:80
this is virginia server
[root@ip-18-0-140-47 html]# exit
logout
[ec2-user@ip-18-0-140-47 ~]$ curl 72.0.0.97:80
this is virginia server
[ec2-user@ip-18-0-140-47 ~]$
```

i-02c6907564cb79b56 (seoul-ec2)
PublicIPs: 3.38.209.80 PrivateIPs: 18.0.140.47

