😊 Site-to-Site VPN Connection

A **site-to-site VPN (Virtual Private Network)** is a secure connection established between two or more separate networks, typically located in different physical locations, allowing them to communicate as if they were on the same local network. This type of VPN is commonly used by businesses to connect branch offices, data centers, or remote offices with the main corporate network.

Key Features:

- **Encryption**: Data transmitted between the sites is encrypted to ensure privacy and security.
- **Tunneling**: A virtual "tunnel" is created over the public internet or other networks to connect the sites securely.
- **Persistent Connection**: The VPN connection is usually always on, providing a constant link between the sites.
- **Network Access**: Devices on one site can access resources (like files, printers, databases) on another site as if they were on the same local network.

Common Use Cases:

- **Branch Office Connectivity**: Connecting remote offices to the central office securely.
- **Multi-Data Center Connectivity**: Linking multiple data centers to act as a single unified network.
- **Partner Network Integration**: Securely connecting the networks of business partners or suppliers.

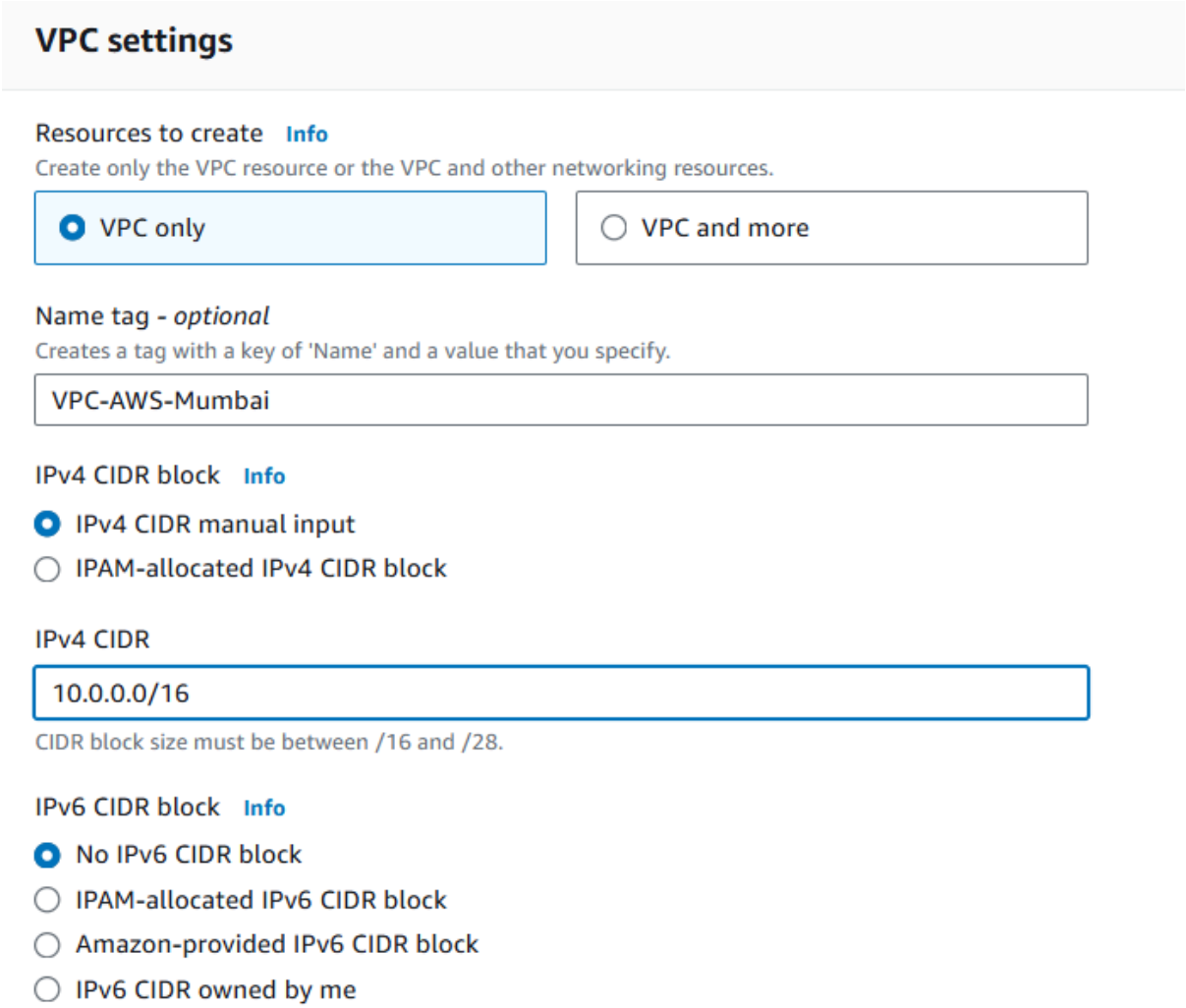
Example:

Imagine a company with headquarters in New York and a branch office in London. A site-to-site VPN allows employees in both locations to access the same corporate resources and communicate securely over the internet as if they were in the same building.

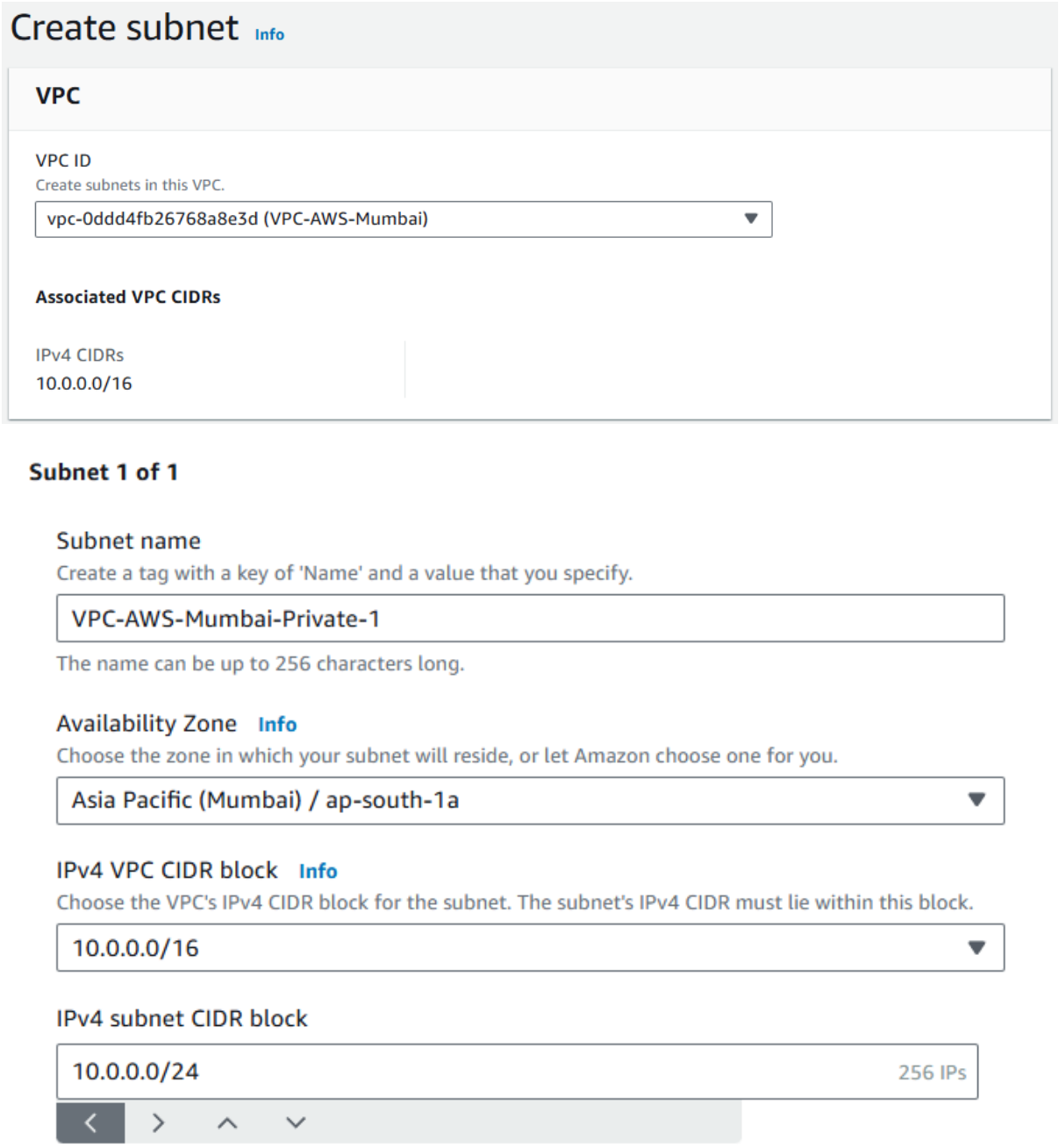
😄 To begin with the Lab:

1. For setting up the VPN connection, we need a set of networks which is AWS VPC. So, for this exercise, we have chosen the Mumbai region and we will create a VPC here.
2. We'll also have a customer network representing another VPC in the North Virginia Region.
3. In Mumbai Region we will have a private subnet and EC2 instance. This instance will just have a private IP address. Now you know that we can't reach this instance from over the internet because there is no internet gateway and EC2 also has a private IP.
4. So, the purpose of this exercise is to get access to this EC2 instance privately from the corporate data centre or say from North Virginia.

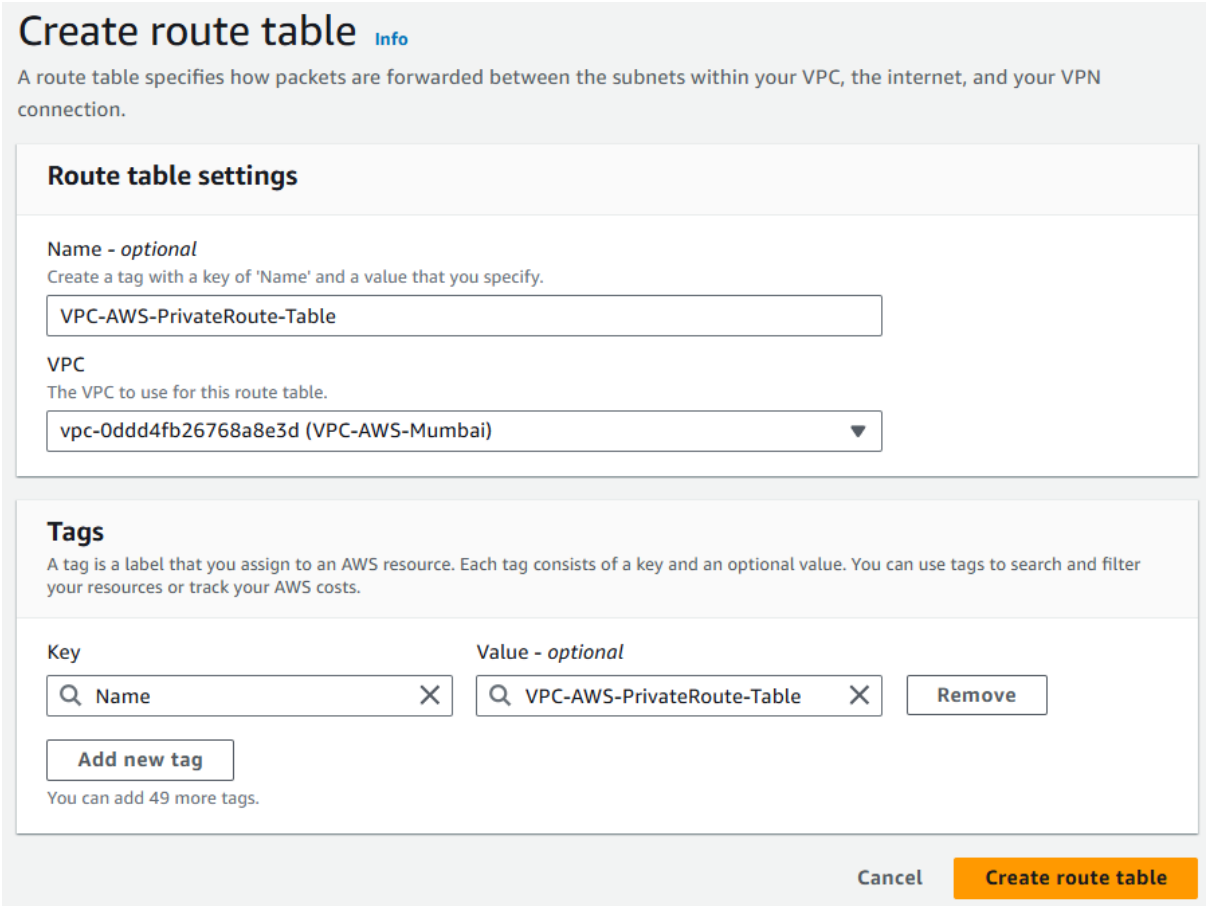
5. Now for this we need a VPN connection and on the AWS side VPN terminates at a Virtual Private Gateway or it is also called a VPN gateway.
6. In your **AWS Console navigate** to VPC in Mumbai Region. Create a new VPC.
7. Here you need to give VPC a name then in the IPv4 CIDR block give the same if you want. VPC-AWS-Mumbai (10.0.0.0/16).



8. Then we need to create a Private Subnet. First, choose your VPC then scroll down, now give your subnet a name and choose the AZ, give the IPv4 subnet CIDR block, create your subnet.

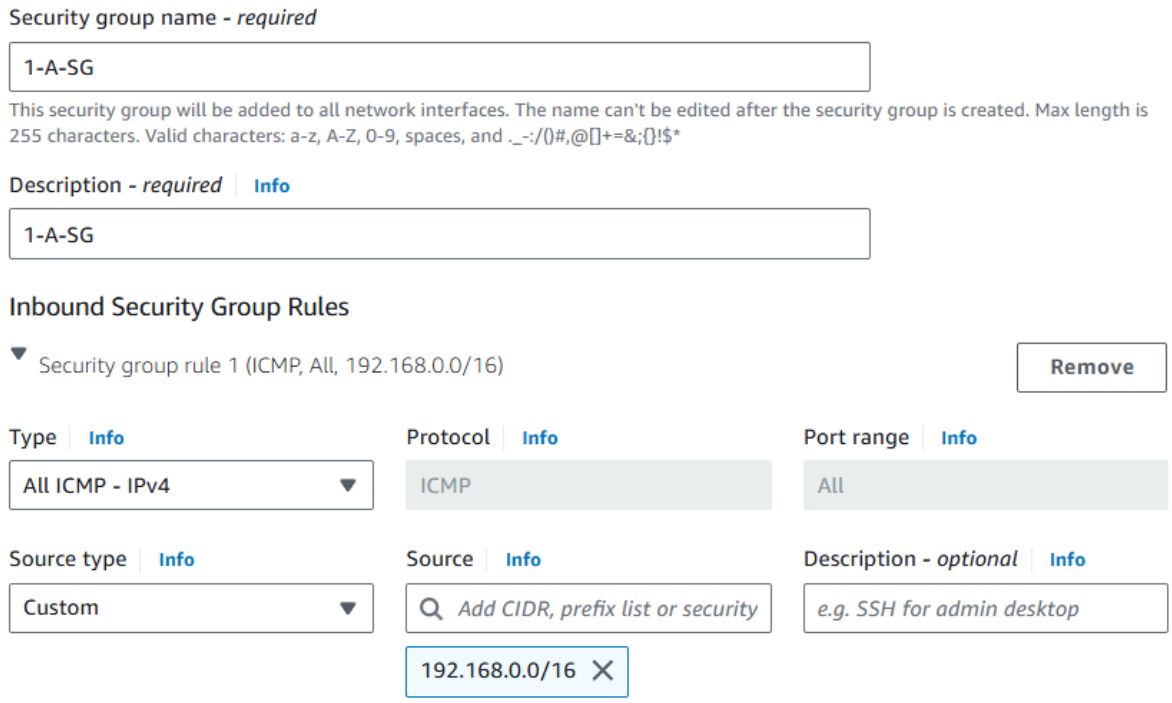


9. Now create a Private Route table. Once your route table is created you need to associate your private subnet with your route table. **This is a must step don't forget it.**

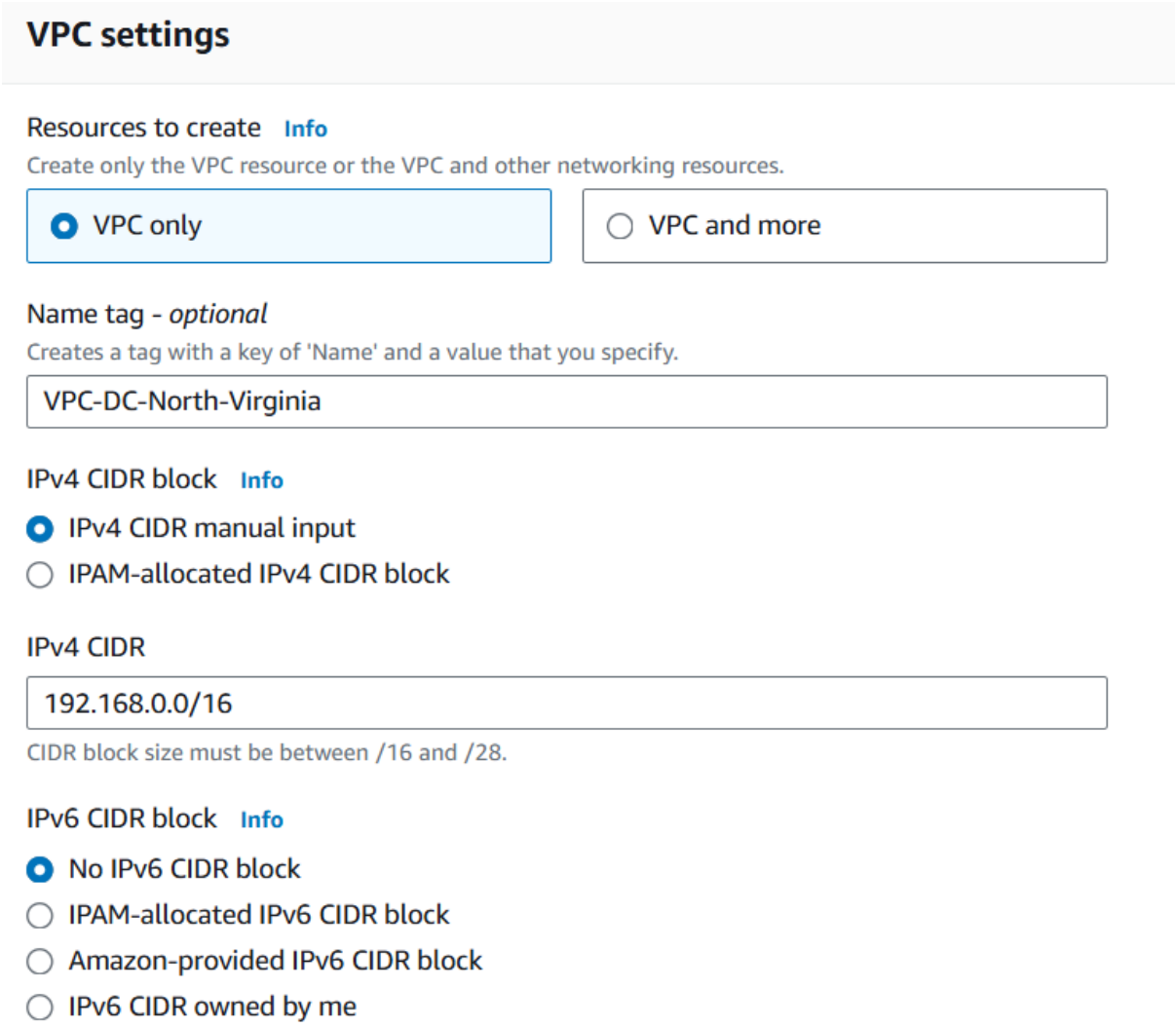


10. Go to EC2 and launch an instance, give it a name, choose Amazon Linux AMI, and create a key pair. In the VPC choose your new VPC and your private subnet disable Public IP for your instance.

11. Then you need to create a new Security group whose inbound rule should be **all ICMP IPv4** with the source 192.168.0.0/16. This IP address is of our corporate data centre VPC which is in North Virginia. Create your EC2 instance.



12. Now go to the North Virginia region and create a VPC. Here choose VPC only, give it a name then give the IPv4 CIDR block and click on create.



13. Now go to Internet gateway and create it, then attach it with your VPC.

VPC > Internet gateways > Create internet gateway

Create internet gateway Info

An internet gateway is a virtual router that connects a VPC to the internet. To create a new internet gateway specify the name for the gateway below.

Internet gateway settings

Name tag

Creates a tag with a key of 'Name' and a value that you specify.

VPC-DC-NV-IGW

Tags - *optional*

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

Key	Value - *optional*	
Name	VPC-DC-NV-IGW	Remove

Add new tag

You can add 49 more tags.

Cancel | Create internet gateway

VPC > Internet gateways > Attach to VPC (igw-0ab206ad39924d060)

Attach to VPC (igw-0ab206ad39924d060) Info

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.

vpc-0b4e17b68cbd89400

▶ AWS Command Line Interface command

Cancel | Attach internet gateway

VPC > Internet gateways > igw-0ab206ad39924d060

igw-0ab206ad39924d060 / VPC-DC-NV-IGW

Actions

Details Info

Internet gateway ID	State	VPC ID	Owner
igw-0ab206ad39924d060	Attached	vpc-0b4e17b68cbd89400 | VPC-DC-North-Virginia	878893308172

Tags

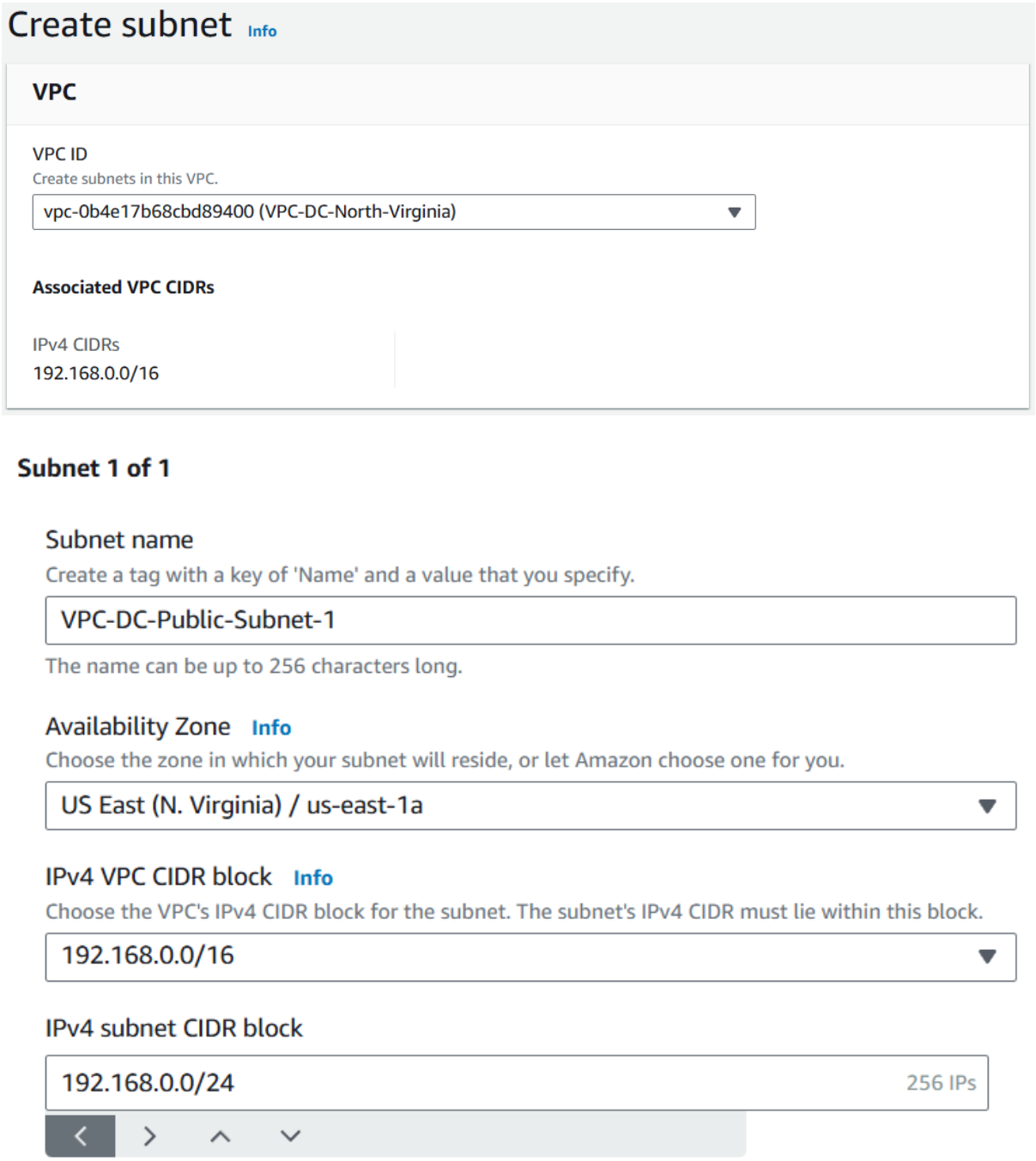
Manage tags

Search tags

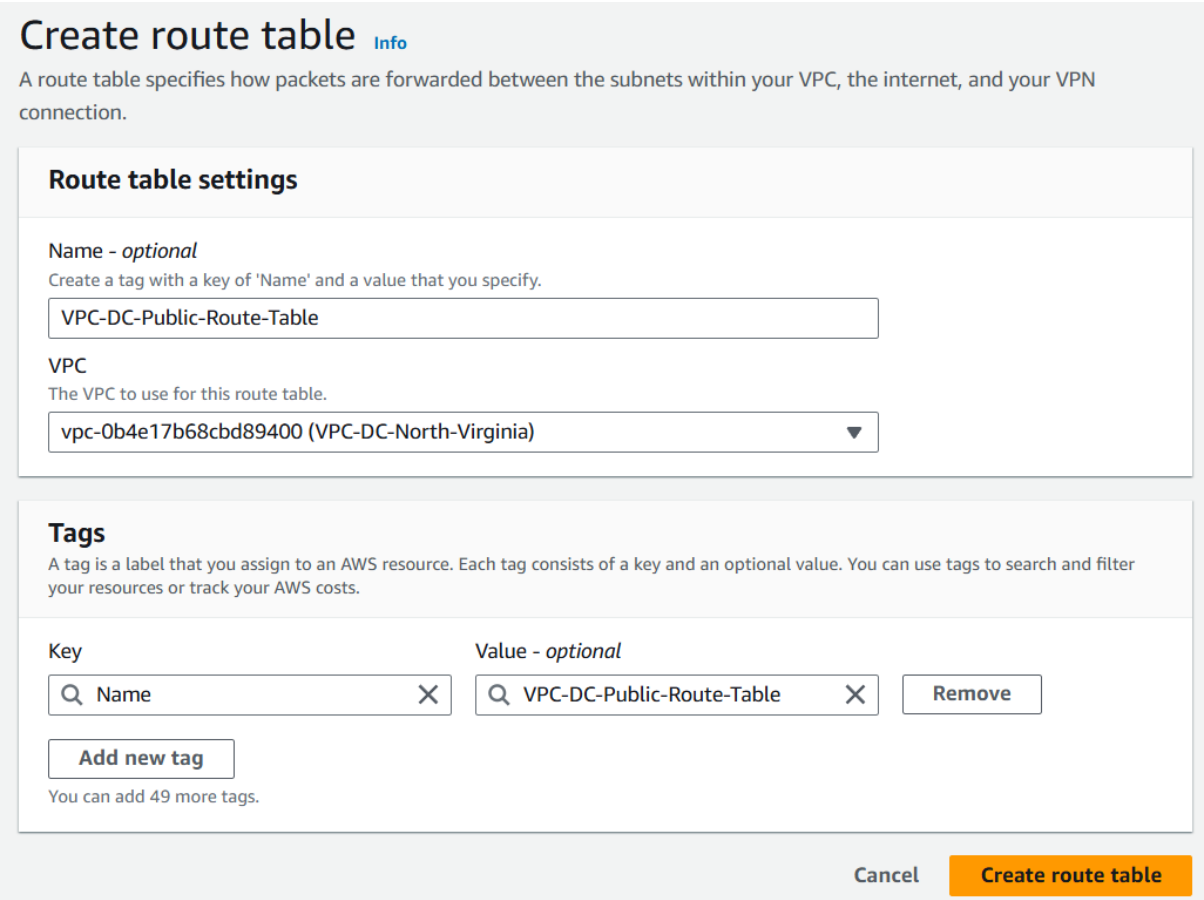
1

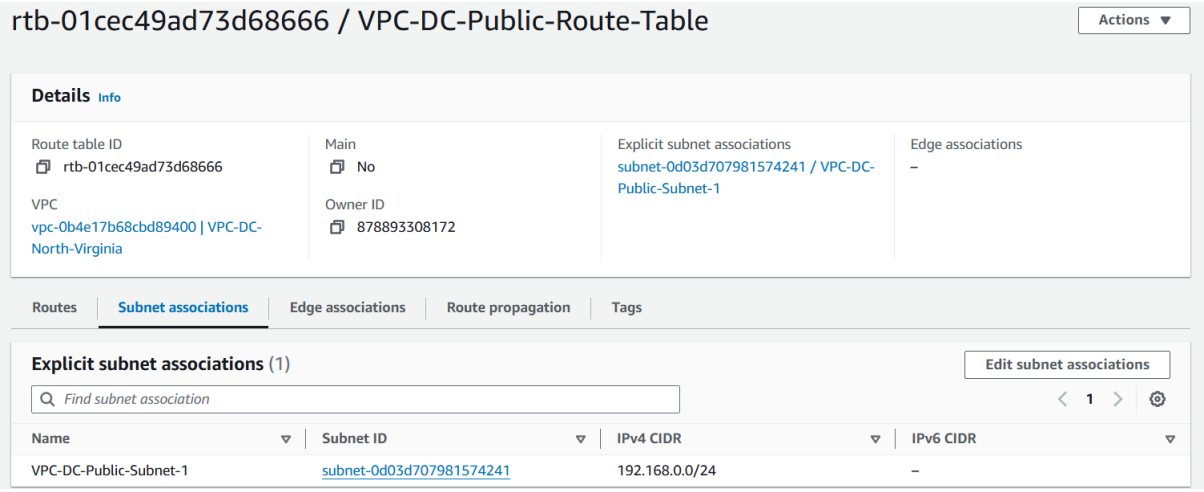
Key	Value
Name	VPC-DC-NV-IGW

14. Go to Subnets and create a public subnet. First, choose your VPC, give it a name, choose any AZ, and give it an IPv4 subnet CIDR block. Create your Subnet.

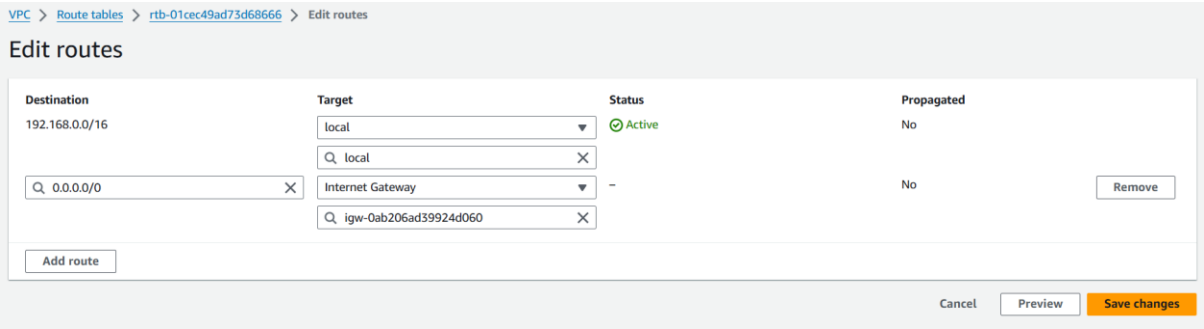


15. After that go to the route table and create a public route table for your VPC. Once your route is created then associate your public subnet with your route table.

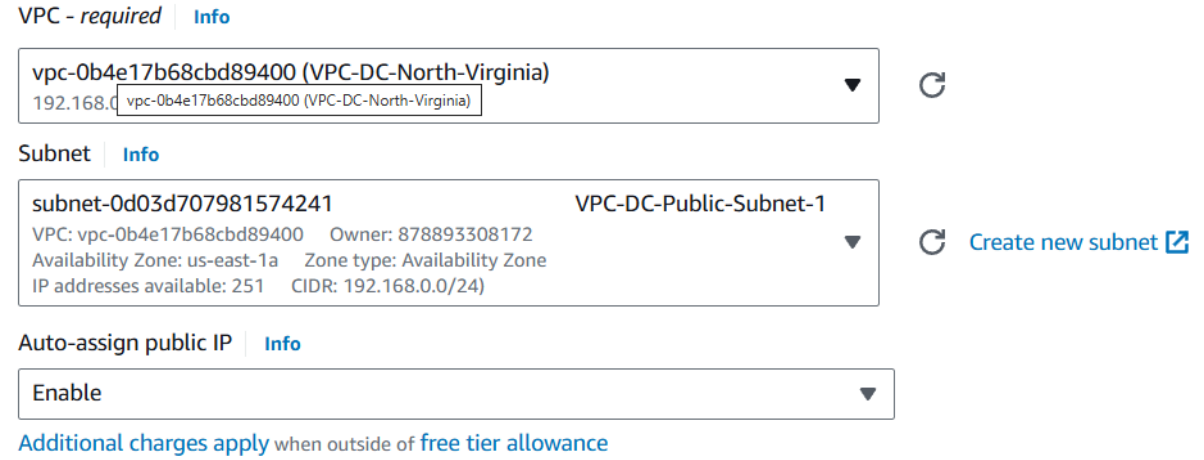




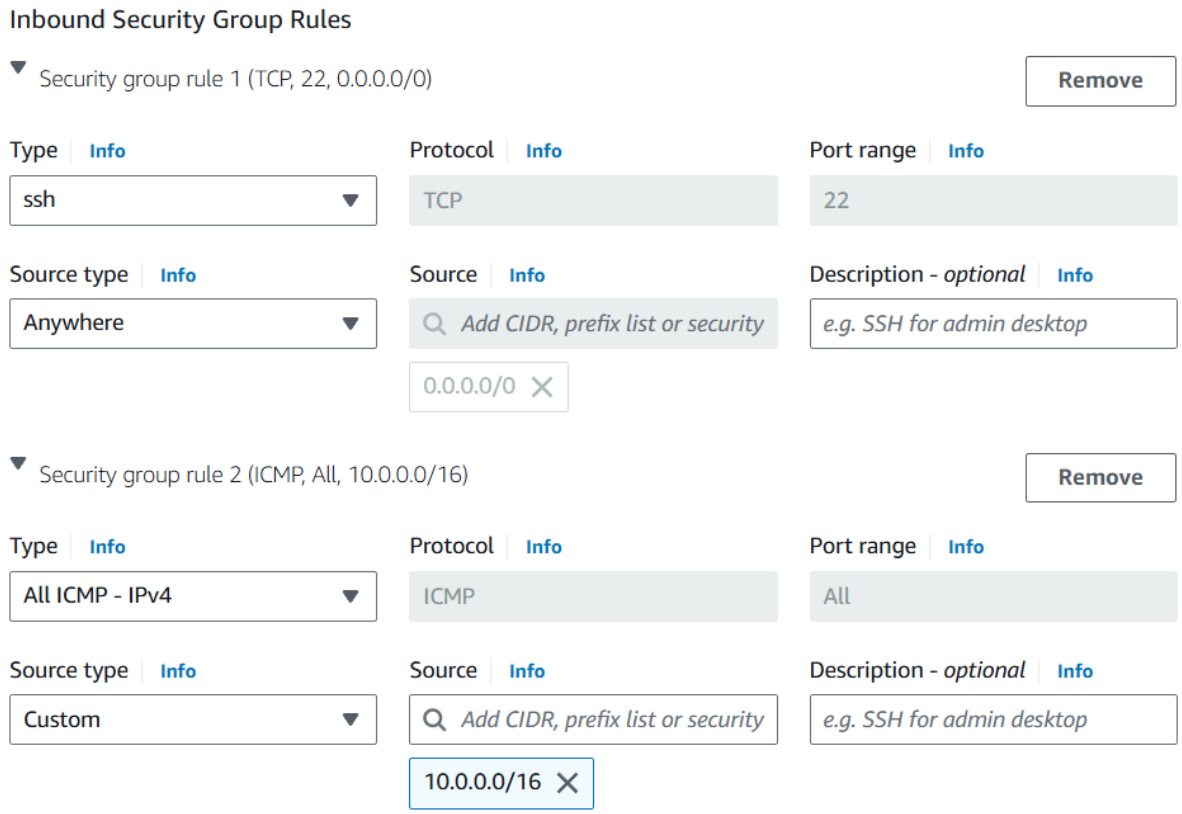
16. Also, you need to make sure that you add a route for the Internet gateway using destination 0.0.0.0/0.



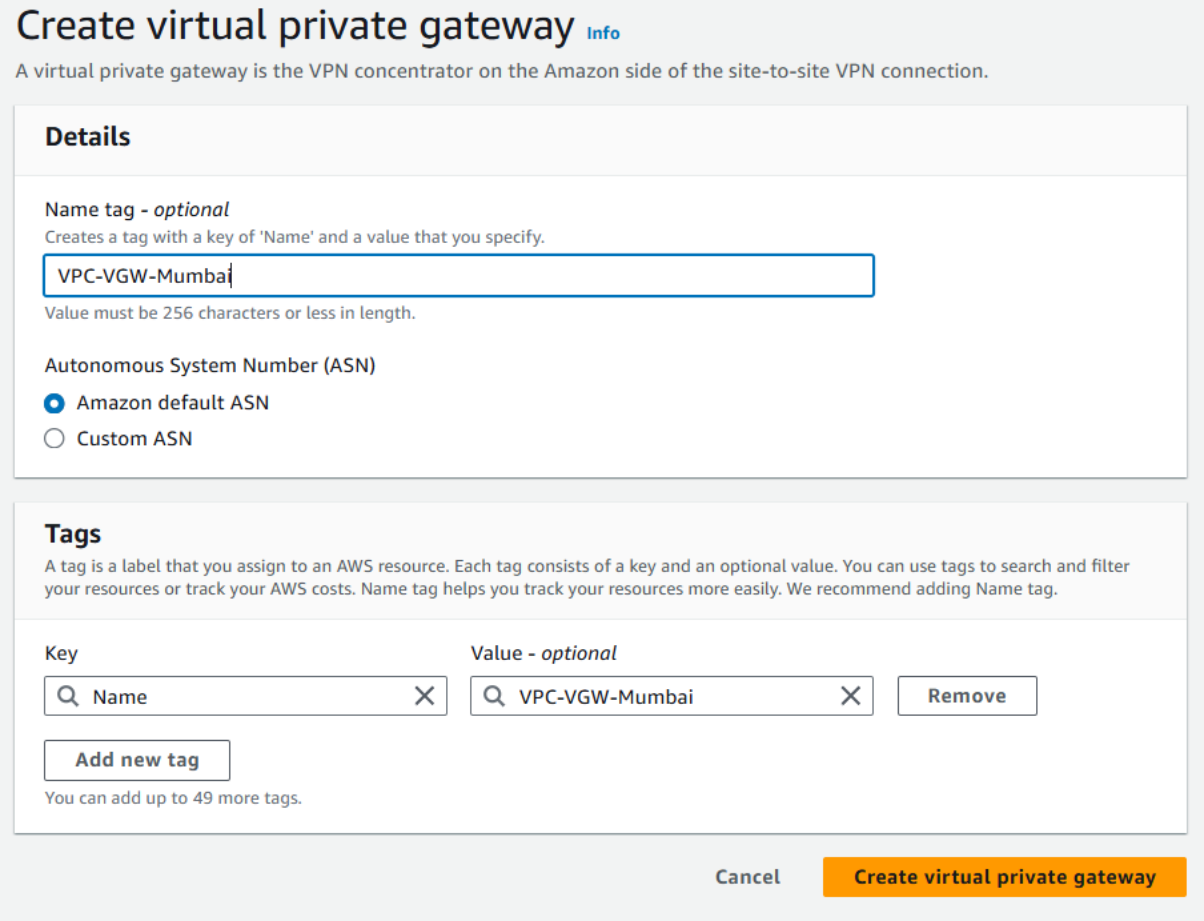
17. After that you must launch an EC2 instance. Give it a name, choose Amazon Linux 2023 as your AMI, and create a key pair. Then choose your VPC and the subnet, enable Public IP.



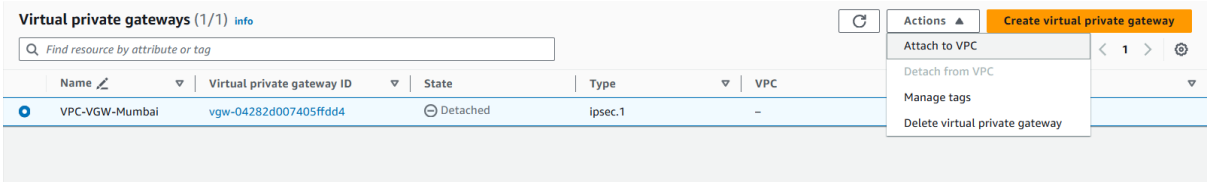
18. Create a new security group, add port 22 for SSH, add All ICMP IPv4 choose the source as 10.0.0.0/16 which is your Mumbai region VPC network CIDR.
19. Then just launch your instance.

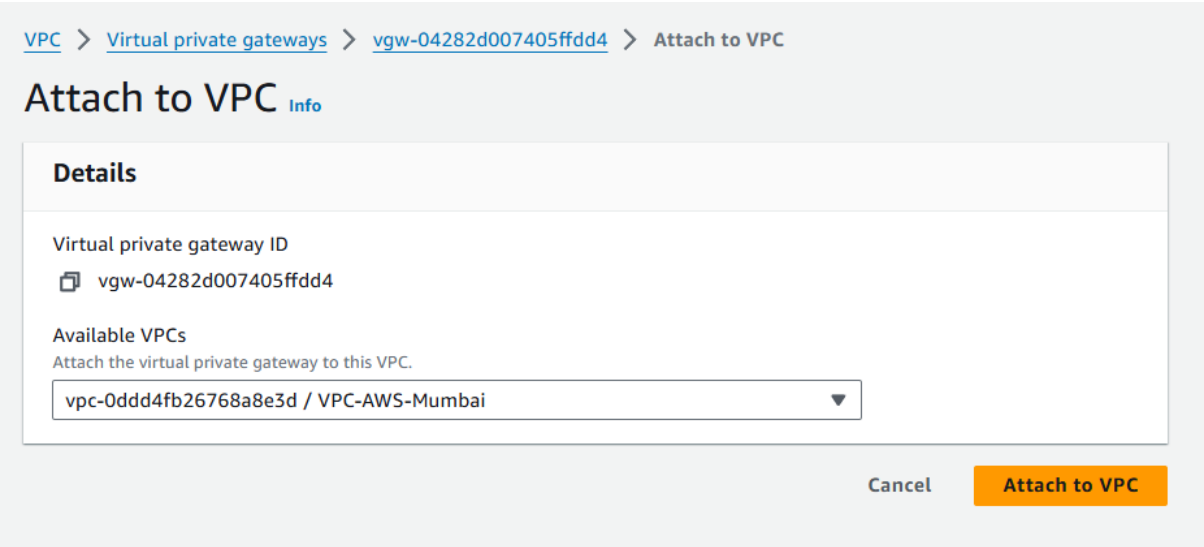


20. Now you need to go to Virtual Private Gateway in the VPC of Mumbai region and click on Create.

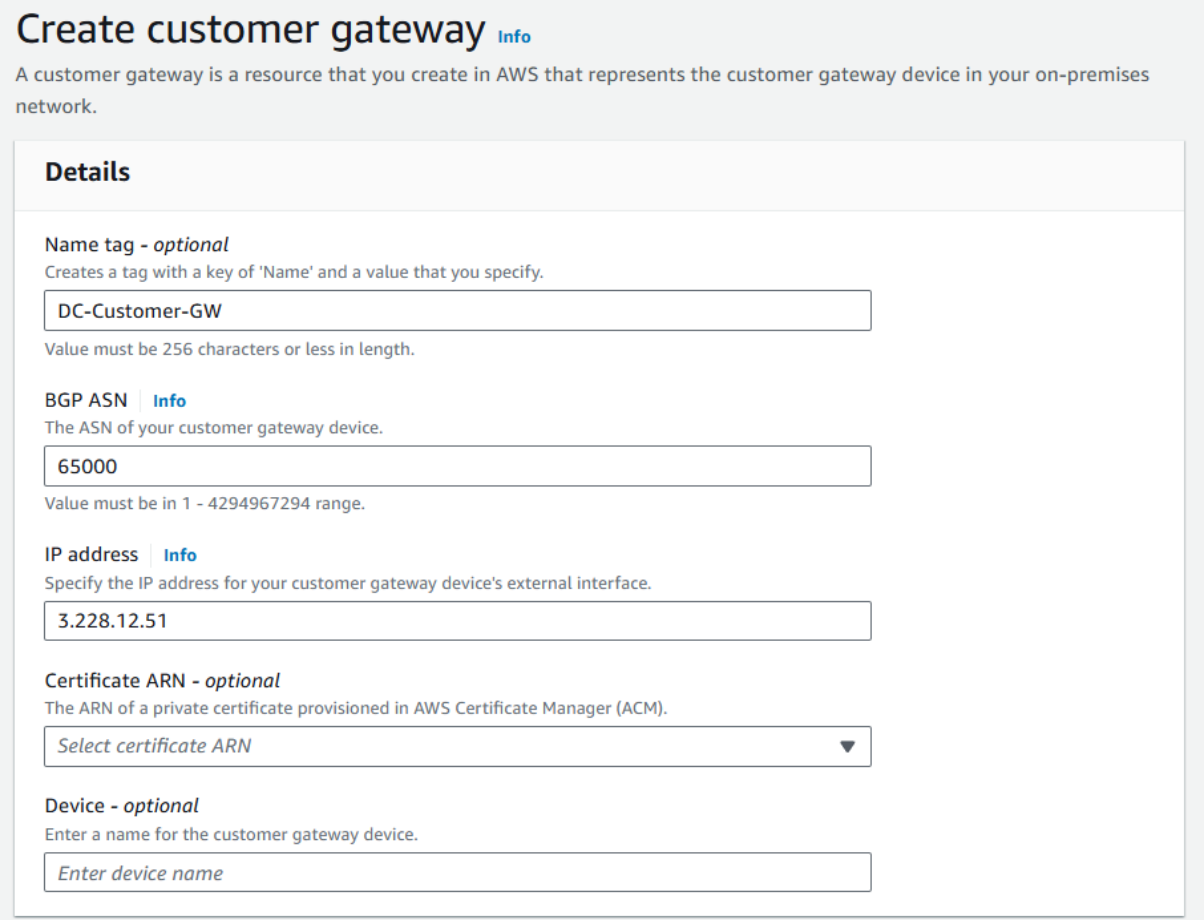


21. Once it is created you will see that it is not attached to our VPC. So, we need to attach it, select it, and click on actions, choose Attach to VPC.

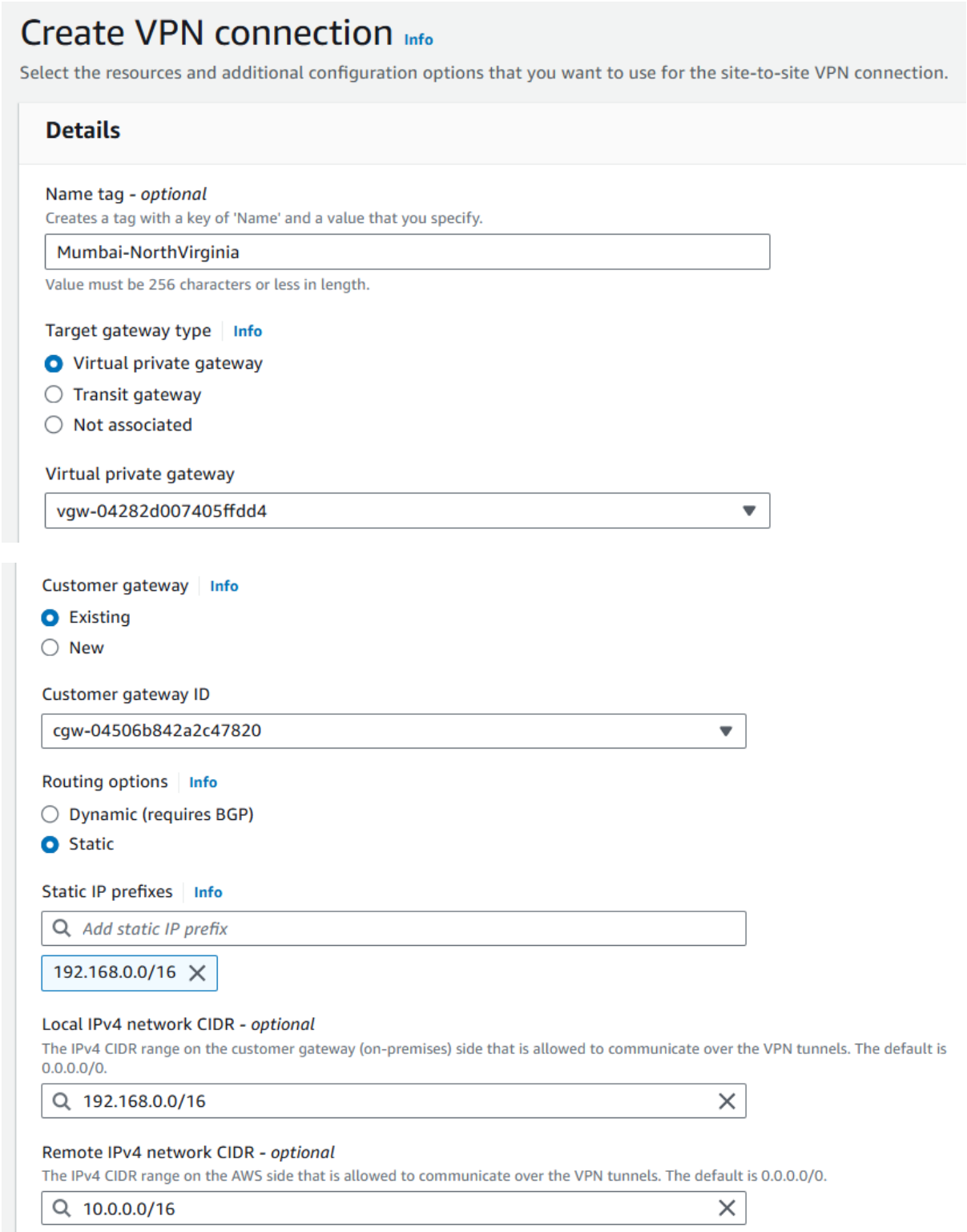




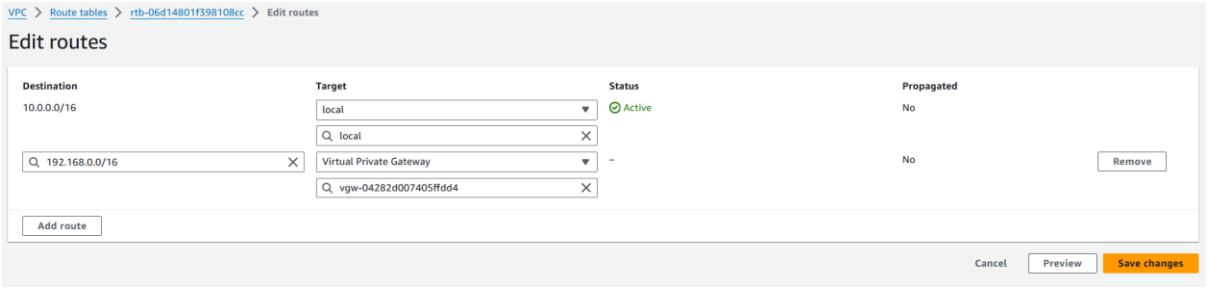
22. After that go to Customer Gateway in the Mumbai region and click on Create. Here you need to give it a name then in the IP address you need to give the public IP of your North Virginia instance. Keep the rest of the settings to default and Click on Create CGW.

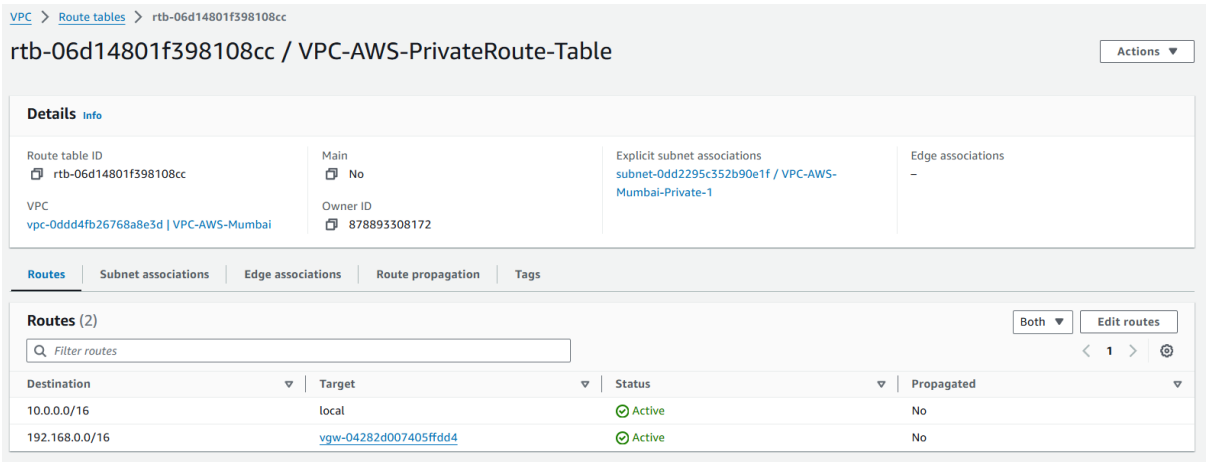


23. Now move to Site-to-Site VPN connections in the Mumbai region and click on Create. Here you need to give it a name then choose Virtual Private Gateway, in the VPG choose the Mumbai side, in the customer gateway choose the North Virginia side.
24. Then in the routing options choose static, in the static IP prefix give the CIDR block of the North Virginia region, this is the same for local IPv4 network CIDR and in the remote give the CIDR block of the Mumbai region.
25. After that keep rest of the settings to default and create your VPN.

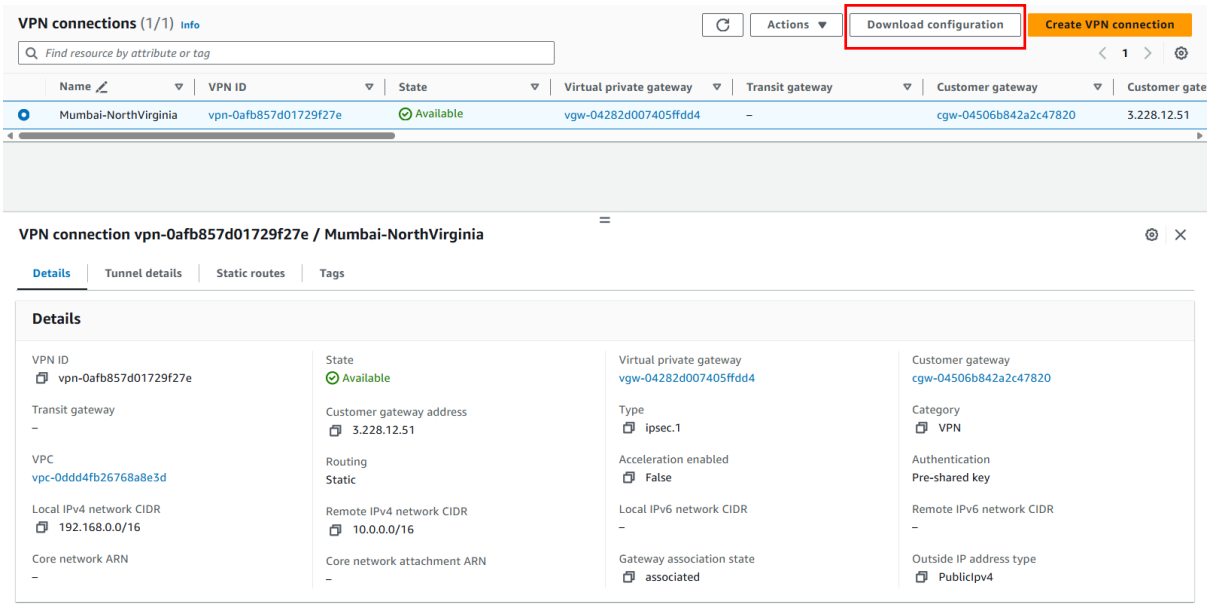


26. So, your VPN will take some time to create till then go to the Private route table of the Mumbai region and here you must add a route for 192.168.0.0/16 and the target is Virtual Private Gateway.

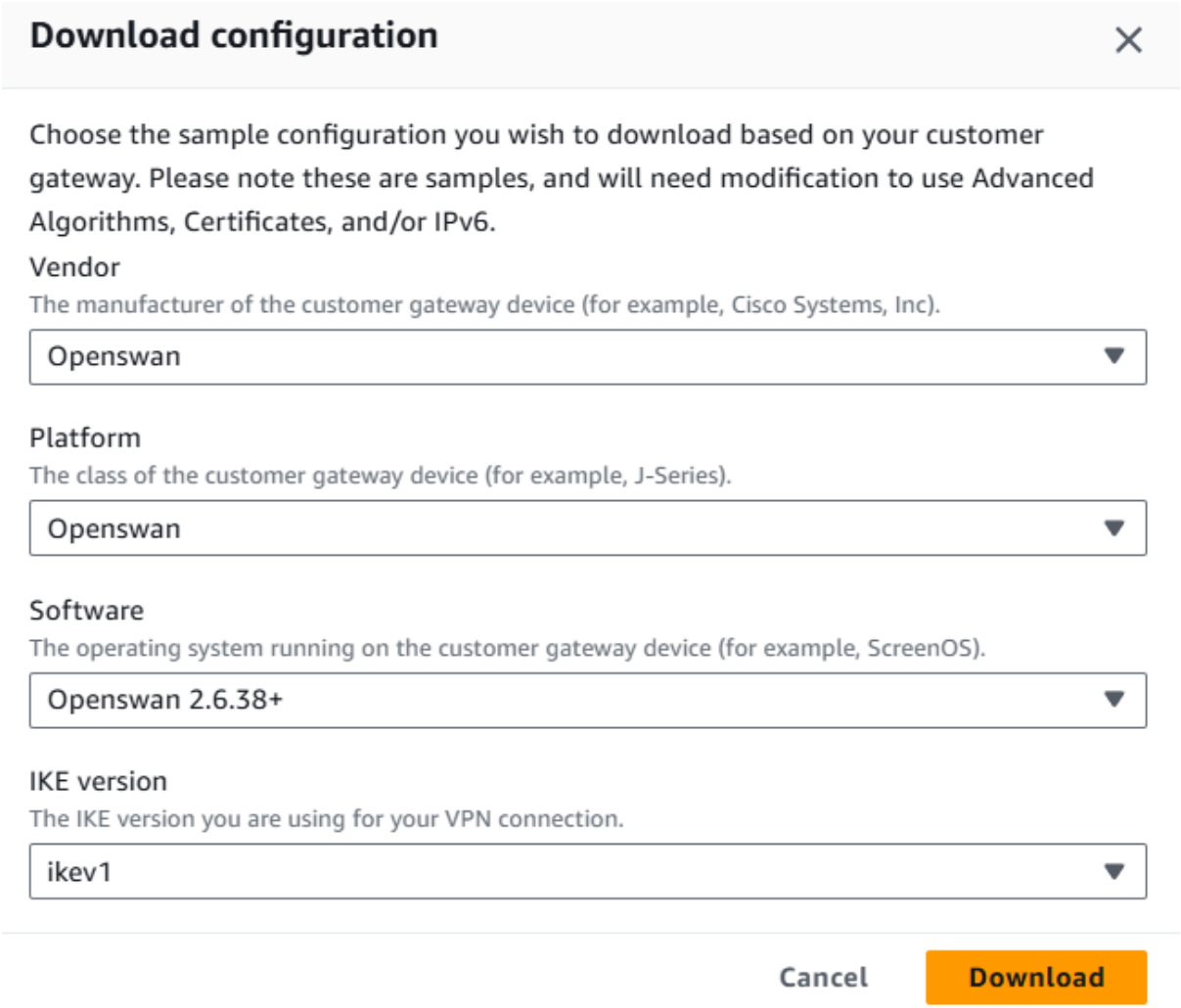




27. Come back to site-to-site VPN, select it and click on download the configuration file.



28. While downloading the configuration file you need to choose Openswan as your Vendor and click on Download.



29. So, this file has all the information to set up a tunnel and you must perform all the steps given in this file.
30. First, we need to connect with our instance in the North Virginia region. Then you need to run some commands.
31. By using the below command, we are creating a repository, and in that we are putting some data. As you can see below.

    sudo vi /etc/yum.repos.d/fedora.repo

    [fedora]
    name=Fedora 36 - $basearch
    #baseurl=http://download.example/pub/fedora/linux/releases/36/Everything/$basearch/os/
    metalink=https://mirrors.fedoraproject.org/metalink?repo=fedora-36&arch=$basearch
    enabled=0
    countme=1
    metadata_expire=7d
    repo_gpgcheck=0
    type=rpm

gpgcheck=1
gpgkey=https://getfedora.org/static/fedora.gpg
skip_if_unavailable=False

```
[fedora]
name=Fedora 36 - $basearch
#baseurl=http://download.example/pub/fedora/linux/releases/36/Everything/$basearch/os/
metalink=https://mirrors.fedoraproject.org/metalink?repo=fedora-36&arch=$basearch
enabled=0
countme=1
metadata_expire=7d
repo_gpgcheck=0
type=rpm
gpgcheck=1
gpgkey=https://getfedora.org/static/fedora.gpg
skip_if_unavailable=False
~
```

32. Then run this command below to install libre swan on your instance.

sudo dnf --enablerepo=fedora install libreswan -y

```
  ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-192-168-0-204 ~]$ sudo vi /etc/yum.repos.d/fedora.repo
[ec2-user@ip-192-168-0-204 ~]$ sudo dnf --enablerepo=fedora install libreswan -y
Fedora 36 - x86_64                                                    64 MB/s |  69 MB     00:01
Last metadata expiration check: 0:00:24 ago on Mon Sep  2 12:13:01 2024.
Dependencies resolved.
================================================================================
 Package                 Architecture        Version                 Repository
================================================================================
Installing:
 libreswan               x86_64              4.12-3.amzn2023.0.2     amazonlinux
Installing dependencies:
 ldns                    x86_64              1.8.3-2.amzn2023.0.1    amazonlinux
 nss-tools               x86_64              3.90.0-6.amzn2023.0.1   amazonlinux
 unbound-libs            x86_64              1.17.1-1.amzn2023.0.5   amazonlinux
Installing weak dependencies:
 unbound-anchor          x86_64              1.17.1-1.amzn2023.0.5   amazonlinux

Transaction Summary
================================================================================
Install  5 Packages
```

33. After that you are going to add some data at his location in your instance mention below.

sudo vi /etc/sysctl.conf

net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.accept_source_route = 0

```
# sysctl settings are defined through files in
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.accept_source_route = 0
~
```

34. Then you need to run this command below which applies the changes basically.

    sudo sysctl -p

```
[ec2-user@ip-192-168-0-204 ~]$ sudo vi /etc/sysctl.conf
[ec2-user@ip-192-168-0-204 ~]$ sudo sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.default.accept_source_route = 0
[ec2-user@ip-192-168-0-204 ~]$
```

35. Open /etc/ipsec.conf and look for the line below. Ensure that the # in front of the line has been removed, then save and exit the file.

    #include /etc/ipsec.d/*.conf

36. Below you can see that the entry is not hashed out. So just exit from here.

```
        # broken and you might need to disable DNSSEC.
        # dnssec-enable=no
        #
        # To enable IKE and IPsec over TCP for VPN server. Requires at least
        # Linux 5.7 kernel or a kernel with TCP backport (like RHEL8 4.18.0-291)
        # listen-tcp=yes
        # To enable IKE and IPsec over TCP for VPN client, also specify
        # tcp-remote-port=4500 in the client's conn section.

# if it exists, include system wide crypto-policy defaults
include /etc/crypto-policies/back-ends/libreswan.config

# It is best to add your IPsec connections as separate files
# in /etc/ipsec.d/
include /etc/ipsec.d/*.conf
```

37. Now open the VPN configuration which we downloaded from site-to-site VPN.
38. So, we have completed our steps from 1 to 3, now we are going to perform step 4.

    sudo vi /etc/ipsec.d/aws.conf

```
This configuration assumes that you already have a default openswan installation in place on the Amazon Linux operating system (but may also work with other distros as well)

1) Open /etc/sysctl.conf and ensure that its values match the following:
   net.ipv4.ip_forward = 1
   net.ipv4.conf.default.rp_filter = 0
   net.ipv4.conf.default.accept_source_route = 0

2) Apply the changes in step 1 by executing the command 'sysctl -p'

3) Open /etc/ipsec.conf and look for the line below. Ensure that the # in front of the line has been removed, then save and exit the file.
    #include /etc/ipsec.d/*.conf

4) Create a new file at /etc/ipsec.d/aws.conf if doesn't already exist, and then open it. Append the following configuration to the end in the file:
 #leftsubnet= is the local network behind your openswan server, and you will need to replace the <LOCAL NETWORK> below with this value (don't include the brackets). If you have multiple subnets, you can use 0.0.0.0/0 instead.
 #rightsubnet= is the remote network on the other side of your VPN tunnel that you wish to have connectivity with, and you will need to replace <REMOTE NETWORK> with this value (don't include brackets).

conn Tunnel1
        authby=secret
        auto=start
        left=%defaultroute
        leftid=3.228.12.51
        right=13.126.252.75
        type=tunnel
        ikelifetime=8h
        keylife=1h
        phase2alg=aes128-sha1;modp1024
        ike=aes128-sha1;modp1024
        auth=esp
        keyingtries=%forever
        keyexchange=ike
        leftsubnet=<LOCAL NETWORK>
        rightsubnet=<REMOTE NETWORK>
        dpddelay=10
        dpdtimeout=30
        dpdaction=restart_by_peer
```

39. So, first we need to remove the highlighted line from the file because this is not supported in libre swan and there are some more changes.

```
conn Tunnel1
        authby=secret
        auto=start
        left=%defaultroute
        leftid=3.228.12.51
        right=13.126.252.75
        type=tunnel
        ikelifetime=8h
        keylife=1h
        phase2alg=aes128-sha1;modp1024
        ike=aes128-sha1;modp1024
        auth=esp
        keyingtries=%forever
        keyexchange=ike
        leftsubnet=<LOCAL NETWORK>
        rightsubnet=<REMOTE NETWORK>
        dpddelay=10
        dpdtimeout=30
        dpdaction=restart_by_peer
~
```

40. We need to change some entries, below you can find those. Also, you need to provide the same CIDR as mentioned below, in the left subnet you have to give the CIDR of North Virginia VPC and in the right subnet you have to give the CIDR of Mumbai VPC. Save this file.

    phase2alg=aes_gcm
    ike=aes256-sha1

```
conn Tunnel1
        authby=secret
        auto=start
        left=%defaultroute
        leftid=3.228.12.51
        right=13.126.252.75
        type=tunnel
        ikelifetime=8h
        keylife=1h
        phase2alg=aes_gcm
        ike=aes256-sha1
        keyingtries=%forever
        keyexchange=ike
        leftsubnet=192.168.0.0/16
        rightsubnet=10.0.0.0/16
        dpddelay=10
        dpdtimeout=30
        dpdaction=restart_by_peer
~
```

41. Below you can see that our changes were made successfully.
42. After that we need to create secrets file whose value will taken from the VPN configuration file.
43. So, this is our step 5, Create a new file at /etc/ipsec.d/aws.secrets if it doesn't already exist, and append this line to the file (be mindful of the spacing!):

    sudo vi /etc/ipsec.d/aws.secrets

    3.228.12.51 13.126.252.75: PSK "kEcQjJsW46w3xX7Nody0t_UxFPZq1LwS"

```
[ec2-user@ip-192-168-0-204 ~]$ sudo vi /etc/ipsec.d/aws.secrets
[ec2-user@ip-192-168-0-204 ~]$ sudo cat /etc/ipsec.d/aws.secrets
3.228.12.51 13.126.252.75: PSK "kEcQjJsW46w3xX7Nody0t_UxFPZq1LwS"

[ec2-user@ip-192-168-0-204 ~]$
```

44. So, everything is done. Now the last step that we need to do is to start the IPsec service. For that run the below command to start the service and check the status.

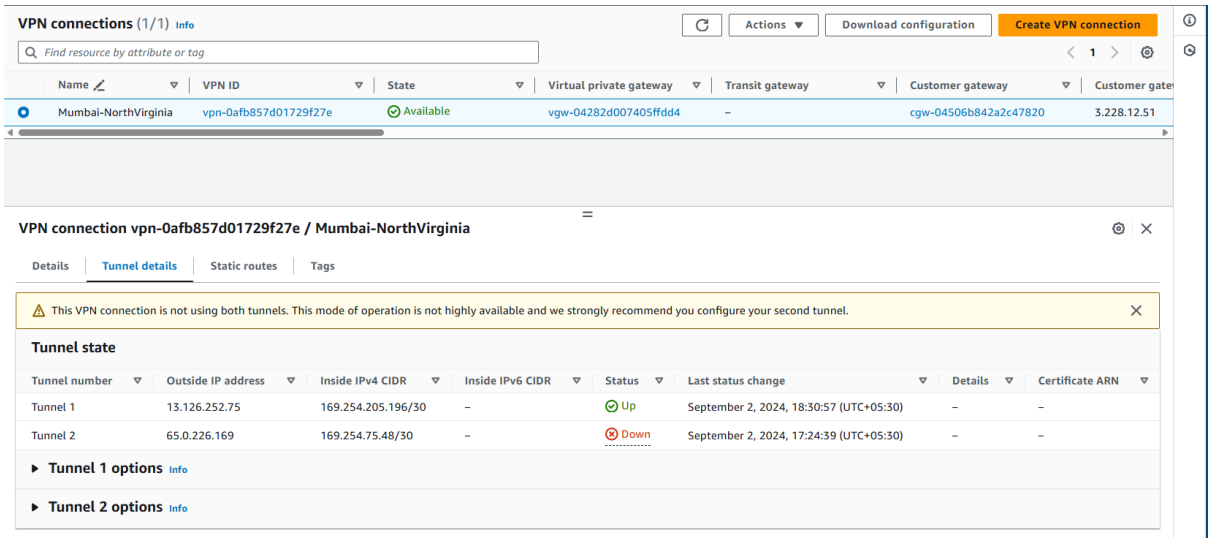
    sudo systemctl start ipsec.service

    sudo systemctl status ipsec.service

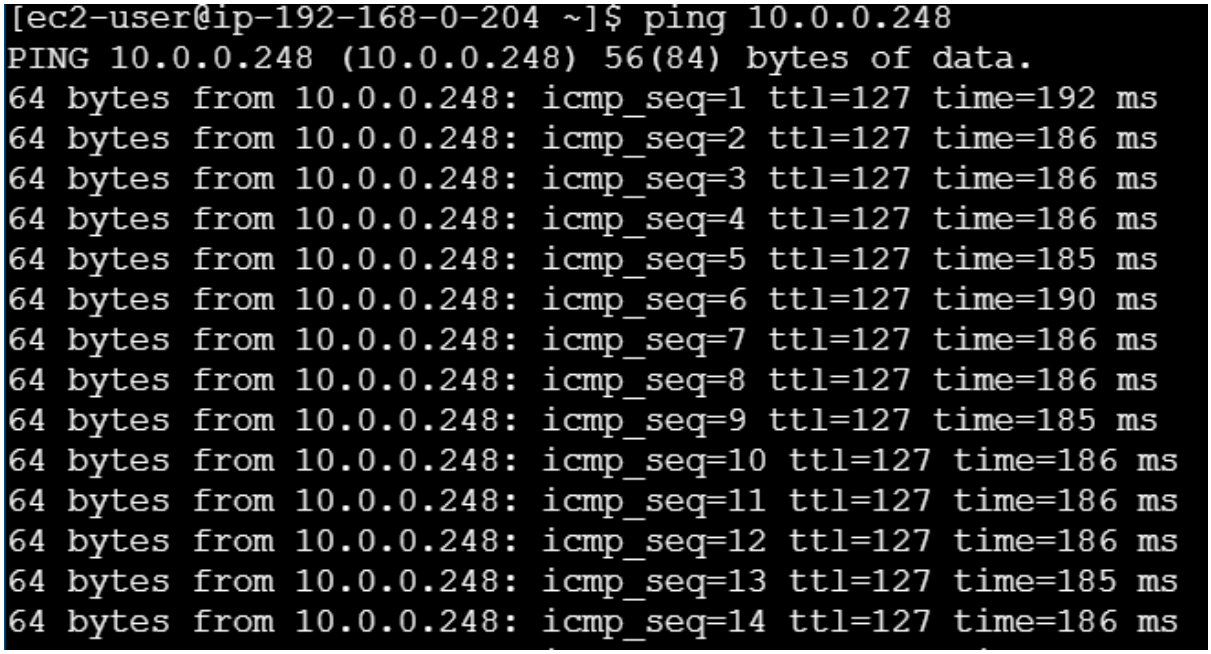
```
[ec2-user@ip-192-168-0-204 ~]$ sudo systemctl start ipsec.service
[ec2-user@ip-192-168-0-204 ~]$ sudo systemctl status ipsec.service
● ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
     Loaded: loaded (/usr/lib/systemd/system/ipsec.service; disabled; preset: disabled)
     Active: active (running) since Mon 2024-09-02 13:00:11 UTC; 11s ago
       Docs: man:ipsec(8)
             man:pluto(8)
             man:ipsec.conf(5)
    Process: 28766 ExecStartPre=/usr/libexec/ipsec/addconn --config /etc/ipsec.conf --checkconfig (code=exited, status=0/SUCCESS)
    Process: 28767 ExecStartPre=/usr/libexec/ipsec/_stackmanager start (code=exited, status=0/SUCCESS)
    Process: 29211 ExecStartPre=/usr/sbin/ipsec --checknss (code=exited, status=0/SUCCESS)
    Process: 29216 ExecStartPre=/usr/sbin/ipsec --checknflog (code=exited, status=0/SUCCESS)
   Main PID: 29227 (pluto)
     Status: "Startup completed."
      Tasks: 2 (limit: 1112)
     Memory: 8.7M
        CPU: 586ms
     CGroup: /system.slice/ipsec.service
             └─29227 /usr/libexec/ipsec/pluto --leak-detective --config /etc/ipsec.conf --nofork

Sep 02 13:00:11 ip-192-168-0-204.ec2.internal pluto[29227]: adding UDP interface lo 127.0.0.1:4500
Sep 02 13:00:11 ip-192-168-0-204.ec2.internal pluto[29227]: adding UDP interface lo [::1]:500
Sep 02 13:00:11 ip-192-168-0-204.ec2.internal pluto[29227]: adding UDP interface lo [::1]:4500
Sep 02 13:00:11 ip-192-168-0-204.ec2.internal pluto[29227]: loading secrets from "/etc/ipsec.secrets"
Sep 02 13:00:11 ip-192-168-0-204.ec2.internal pluto[29227]: loading secrets from "/etc/ipsec.d/aws.secrets"
Sep 02 13:00:11 ip-192-168-0-204.ec2.internal pluto[29227]: "Tunnel1" #1: initiating IKEv2 connection
Sep 02 13:00:11 ip-192-168-0-204.ec2.internal pluto[29227]: "Tunnel1" #1: sent IKE_SA_INIT request to 13.126.252.75:500
Sep 02 13:00:11 ip-192-168-0-204.ec2.internal pluto[29227]: "Tunnel1" #1: sent IKE_AUTH request {cipher=AES_CBC_256 integ=HMAC_SHA1_96 prf=HMAC_SHA1 group=MODP2048
Sep 02 13:00:12 ip-192-168-0-204.ec2.internal pluto[29227]: "Tunnel1" #1: initiator established IKE SA; authenticated peer using authby=secret and ID_IPV4_ADDR '13
Sep 02 13:00:12 ip-192-168-0-204.ec2.internal pluto[29227]: "Tunnel1" #2: initiator established Child SA using #1; IPsec tunnel [192.168.0.0-192.168.255.255:0-6553
```

45. Now if you go towards site-to-site VPN in Mumbai region you will see that our 1 tunnel is up.



46. Once your tunnel is up then you need to copy the private IP of Mumbai region instance and try to ping it from North Virginia region.
47. And below you can see that our lab is now completed.



After successful VPN connectivity, delete all the resources that you created during this lab

- **Delete VPN Connection in (Mumbai region)**
- **Delete Customer Gateway (Mumbai region)**
- **Delete Virtual Private Gateway (Mumbai region)**
- **Terminate both EC2 instances (both the regions)**