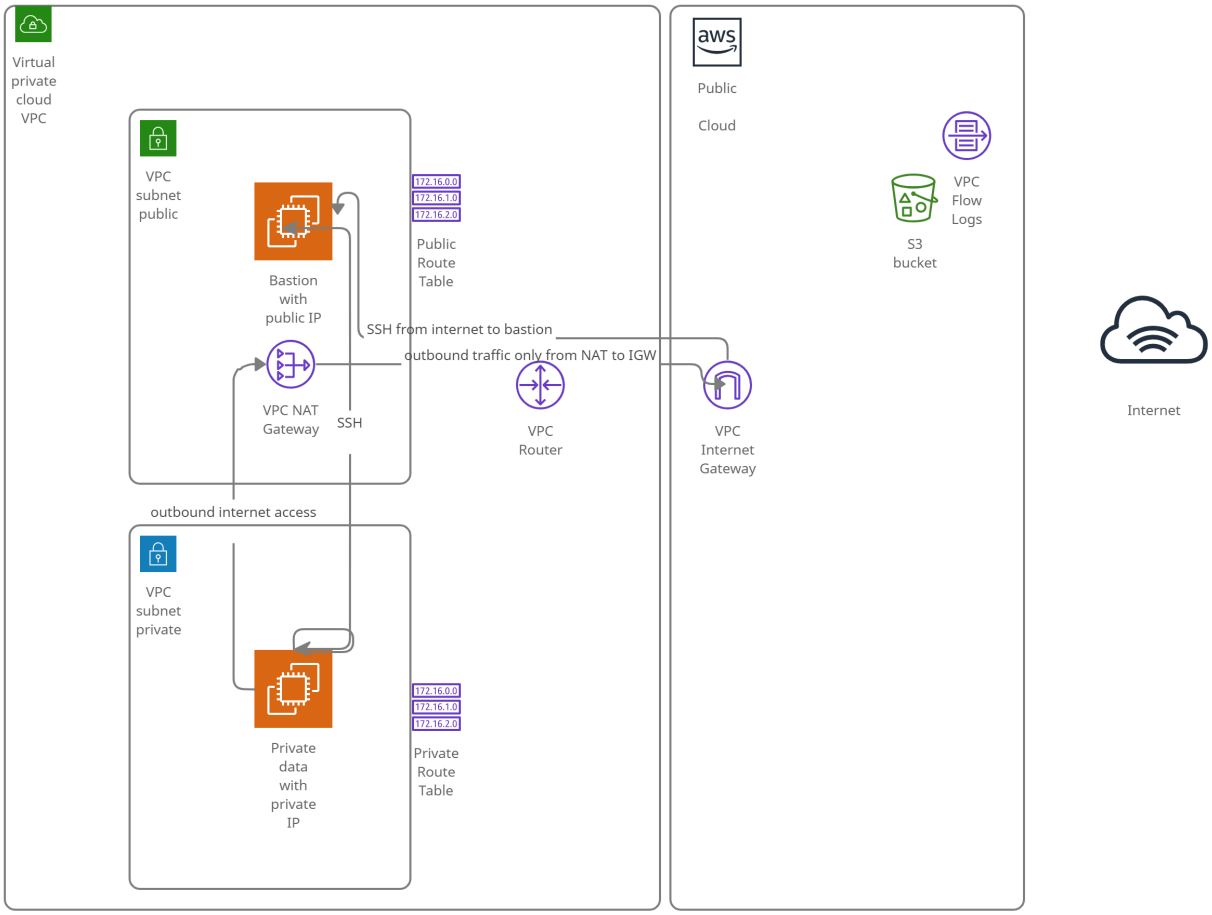
# GL

**Custom VPC with public and private subnets allowing private instance with SSH connectivity via bastion host and external traffic via IGW and NAT Gateway. Access monitored via VPC Flow logs stored in S3**




## Outline

- Creating a private virtual cloud to have more control over the security of our resources. Creating two subnets, one public and one private
- Public subnet contains a NAT Gateway for a private instance without a public IP to send data out, as well as a bastion host to SSH into from the public internet to

then be forwarded into private instance adding layers of security to hide away access.

- Attaching VPC flow logs to monitor access which will be stored in S3 bucket.
- Going to utilize local zones to get lower compute latency.
- SSH Agent forwarding to be able to forward us from bastion host to the private instance without the need of storing a key onto the instance for added security and means we do not have to continuously copy the key onto instances. The Agent will keep the private key portion and the Agent talks to the bastion host as well as the user to authenticate. Private instance talks to the bastion for the ok of user, and you are forwarded.

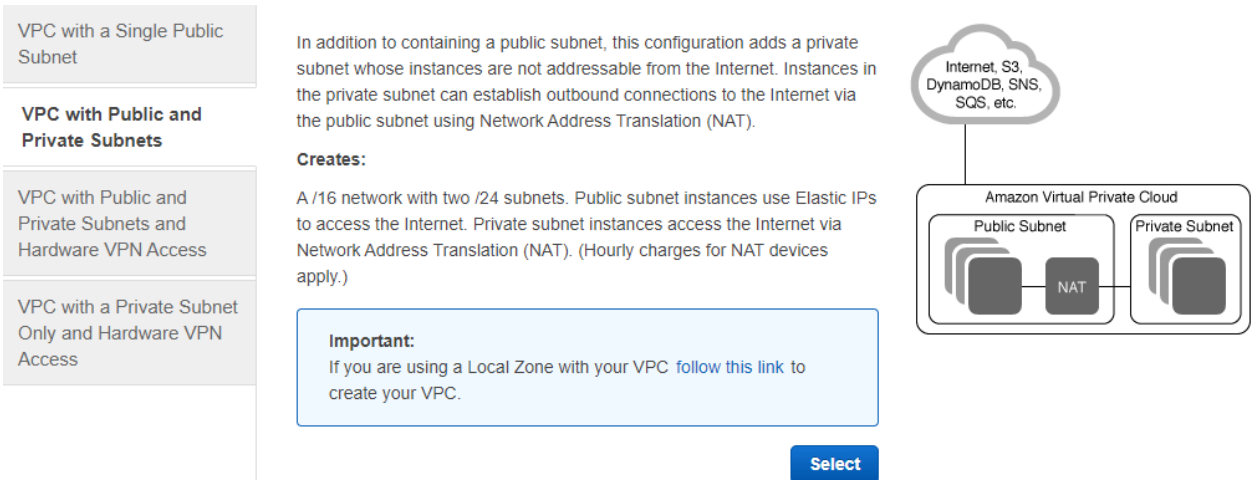
**Services:**

- AWS VPC:
  - Regional level, if the region goes down your VPC goes down
  - Creating an isolated private virtual network so we can have control over IP address range, subnets, route tables, network traffic
- VPC Routing
  - How we allow traffic to enter the public cloud for S3 and private cloud for your private instance as well as the internet to be able to SSH into.
  - We route traffic between subnets and this is controlled by route tables which is defaulted to the main route table made by AWS if not created. Route tables come with one noneditable local route that matches the CIDR range of the VPC It is in, if IPv6 is enabled it will have one for that too.
  - VPC router sends the packets via destination IP which is instructed by the destination field route table. If multiple routes match the higher the prefix the higher the priority route so /32 beats /16.
- Bastion host or jump box
  - All admin connections arrive and only way to SSH into internal resources
  - Acts as a middleman to filter out connections and to add a layer of security for the private instance in this case.
- VPC Flow logs

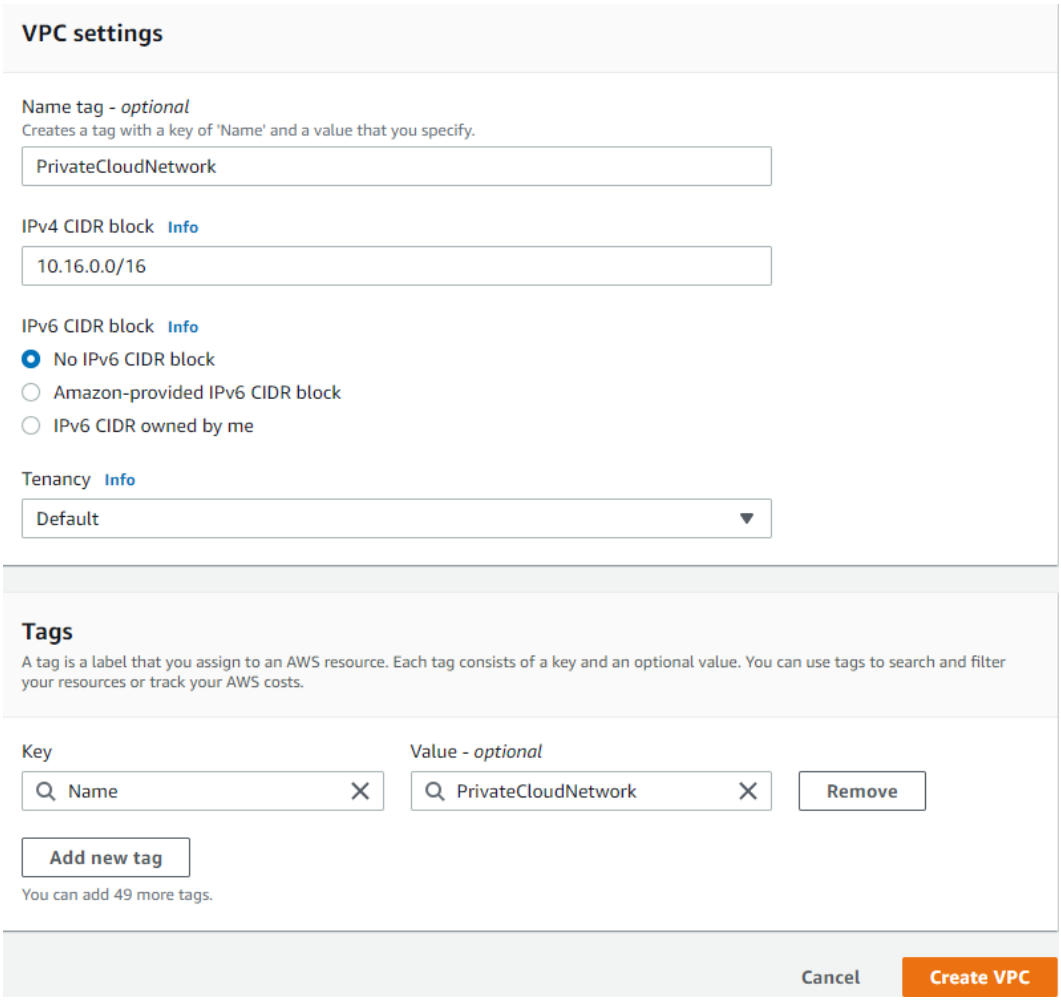
    - Enables us to capture information about IP Traffic to and from network interfaces and subnets and the VPC itself.
    - Pushes the data to CloudWatch logs or S3, in our case we will store in S3.
    - Not real time
- Internet Gateway (IGW)
    - Regional level same as VPC but managed by AWS
    - Runs from public cloud to be able to connect it with your private cloud network and or public internet. This allows full usage of AWS services.
    - IGW contains in the backend a record that matches the private instance IP to public IP. The IP is never attached directly to the instance instead it is associated with the IGW which adds a layer of security. If you are inside the instance you can only view its private IP address because it doesn't know it BUT IPv6 does know the public IP.
- Keypair for SSH
    - Remember you need a keypair, private key is stored with user and public key stored on AWS instances which allows for SSH.
    - ED25519 type keys do not support windows instances, EC2 instance connect and EC2 serial console, RSA Supports all.

**Steps:**

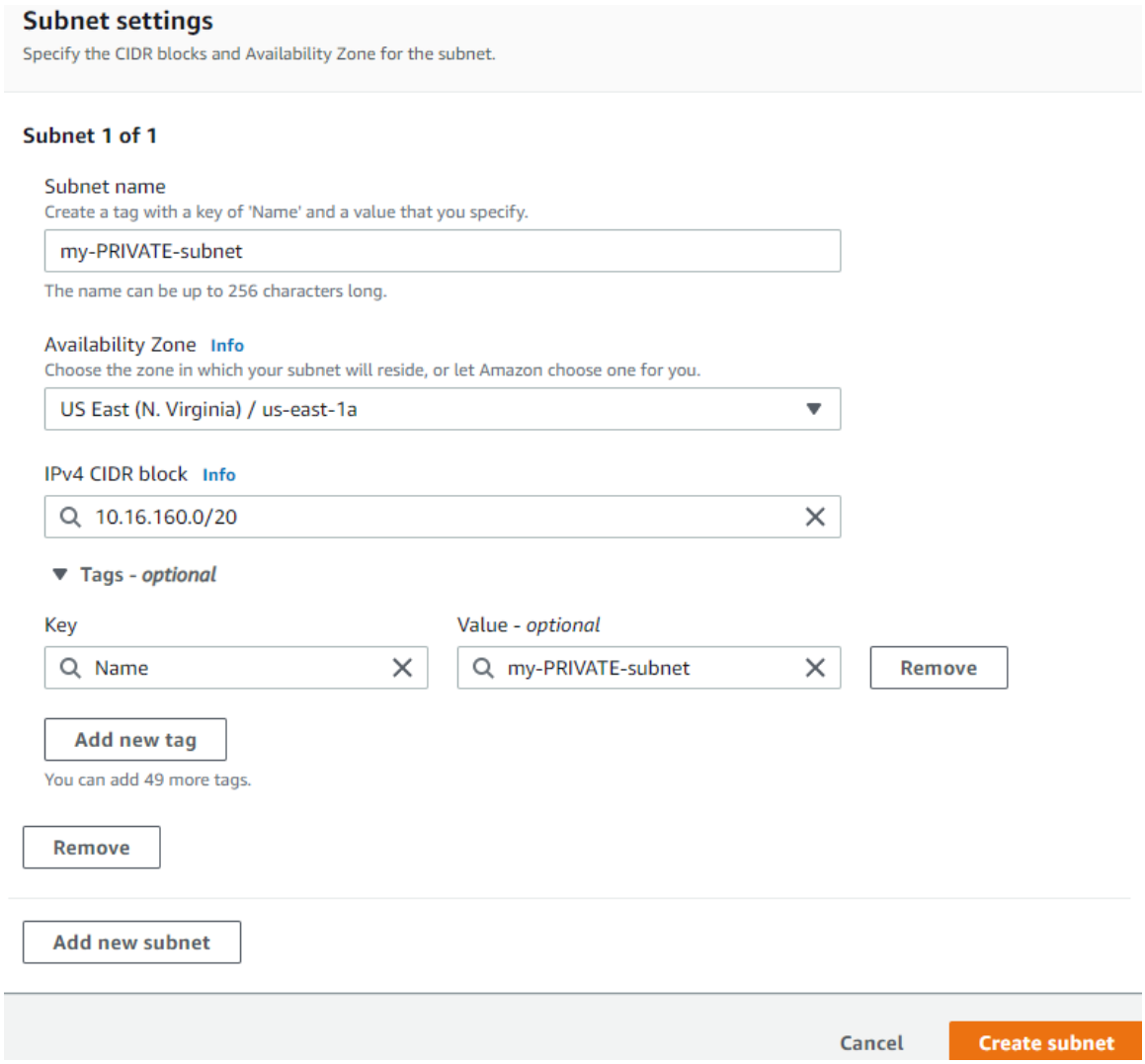
- **When creating VPC, we want one with public + private subnets, public for the bastion host and private subnet contains our testing instance. We will be choosing local zone as we want low compute latency speeds, this allows the instances to be closer to the end user by utilizing edge locations.**



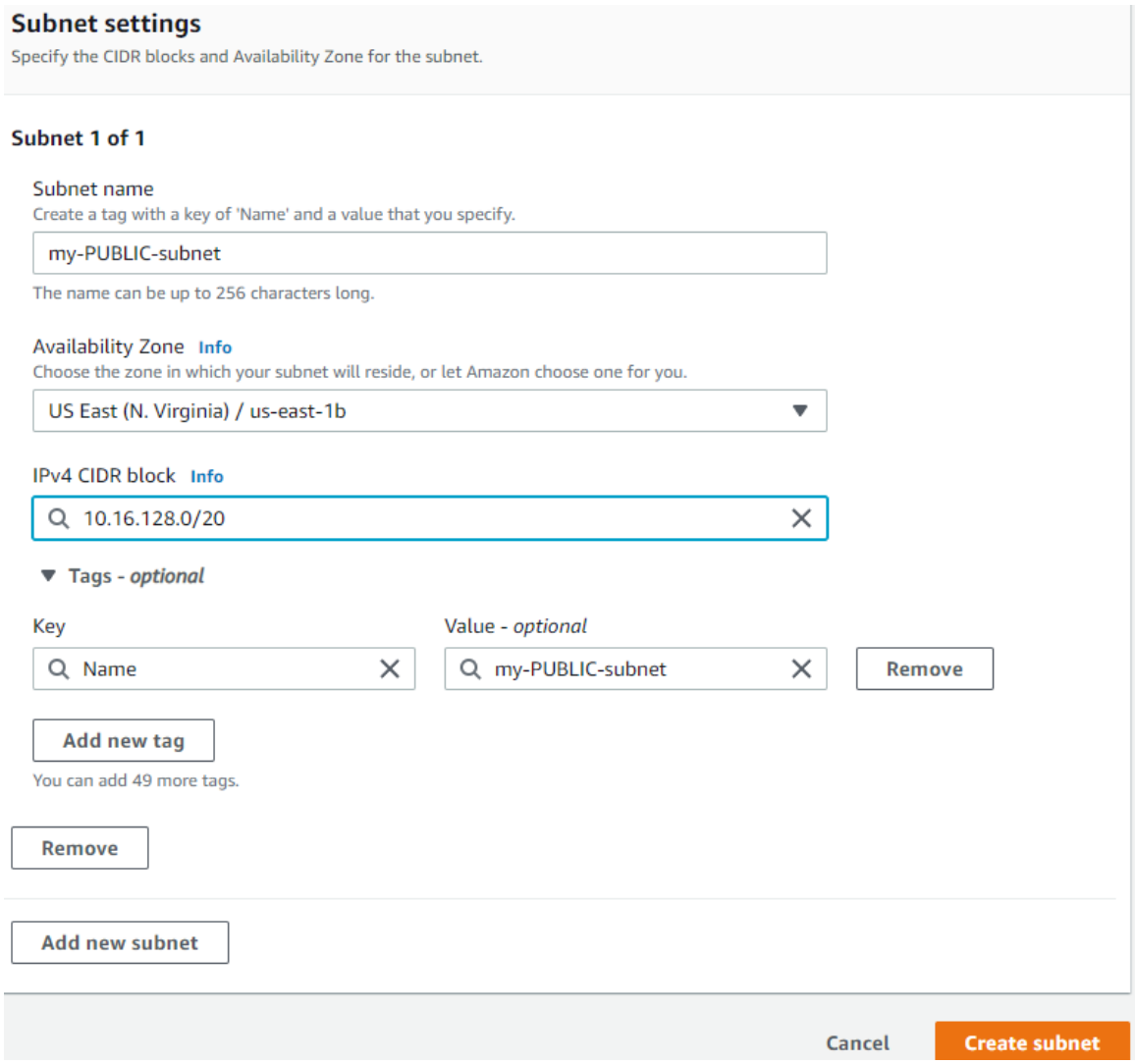

- **Adding name and for the CIDR block we are going to use 10.0.0.0/24 which is recommended to give us access to subnets.**



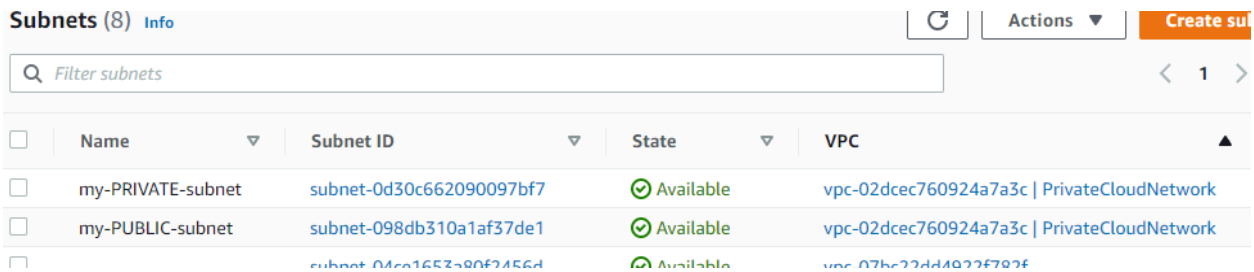
- **Now we need 2 subnets to place into our private cloud. First we'll make the private subnet and we'll put it in AZ-1a.**



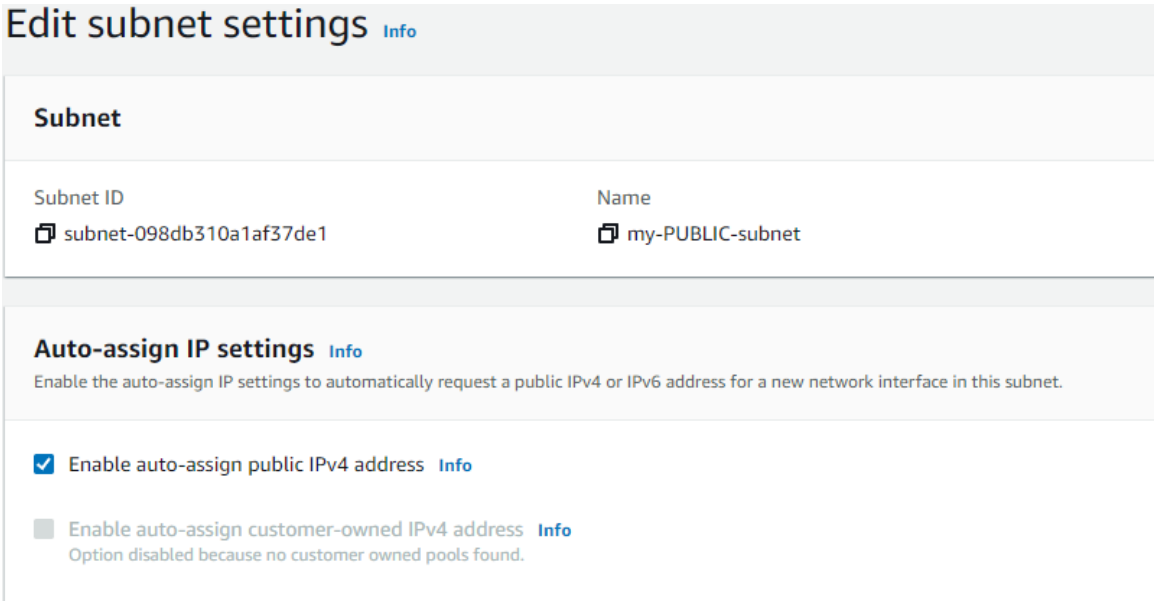

- **Now same thing for the public subnet, but this time put it in a different AZ, I chose AZ-1b.**



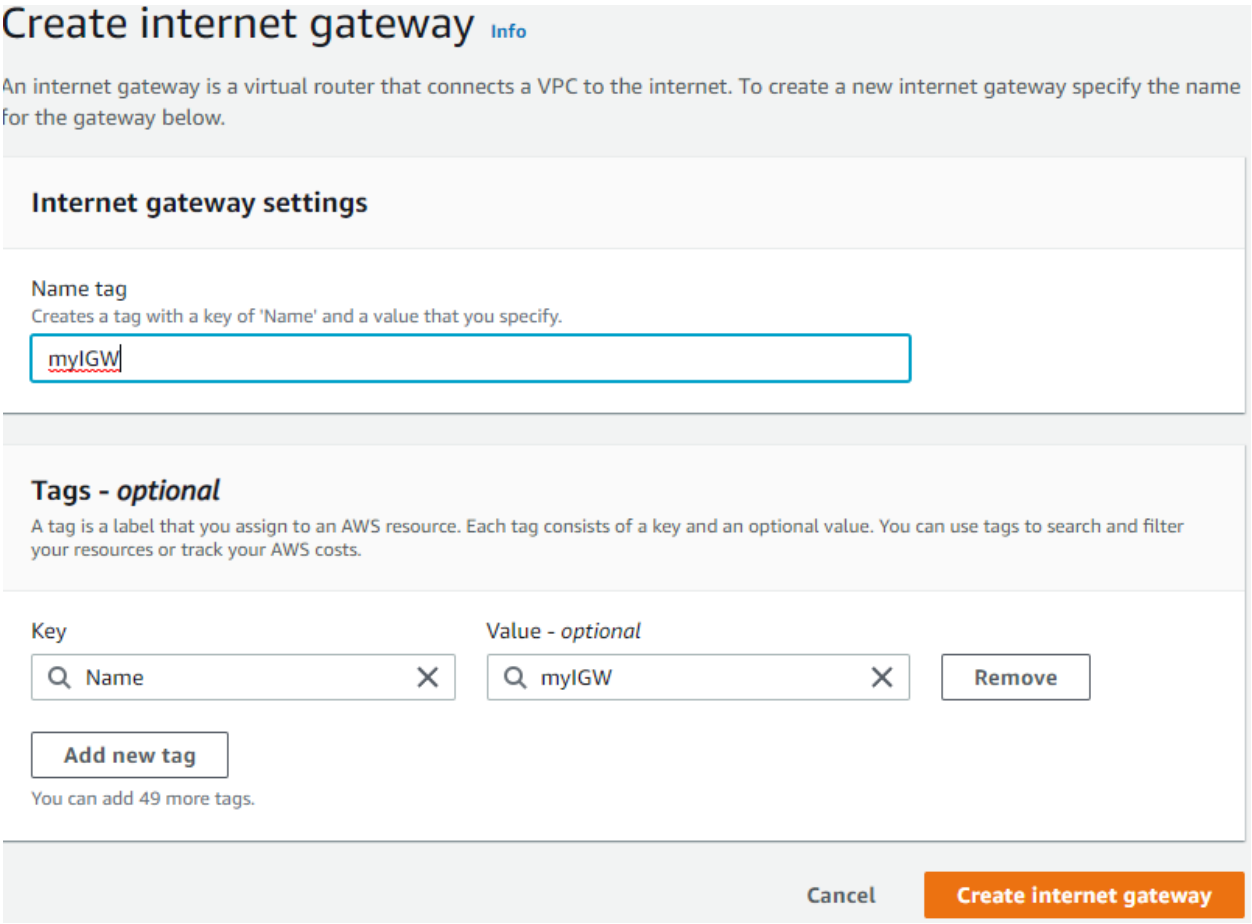

- **And to confirm you should be able to filter via your VPC and see your 2 subnets.**



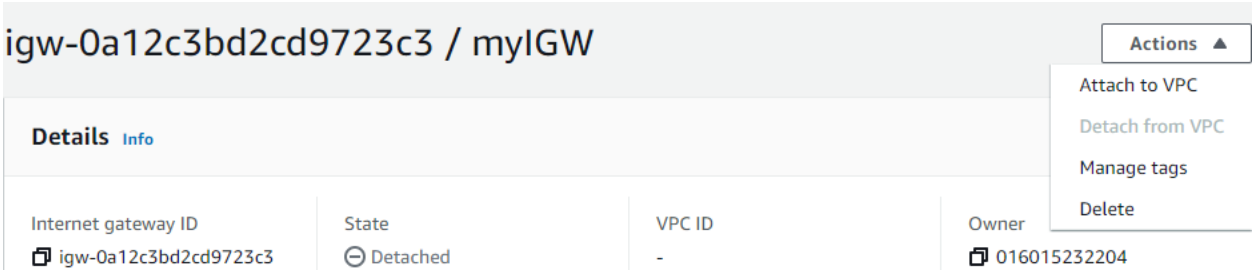
- **We need to make our public subnet auto assign IP so we have an IP for our future bastion host and for it to be able to talk to the IGW.**




- **We are going to create our internet gateway to able to access public cloud and public internet.**

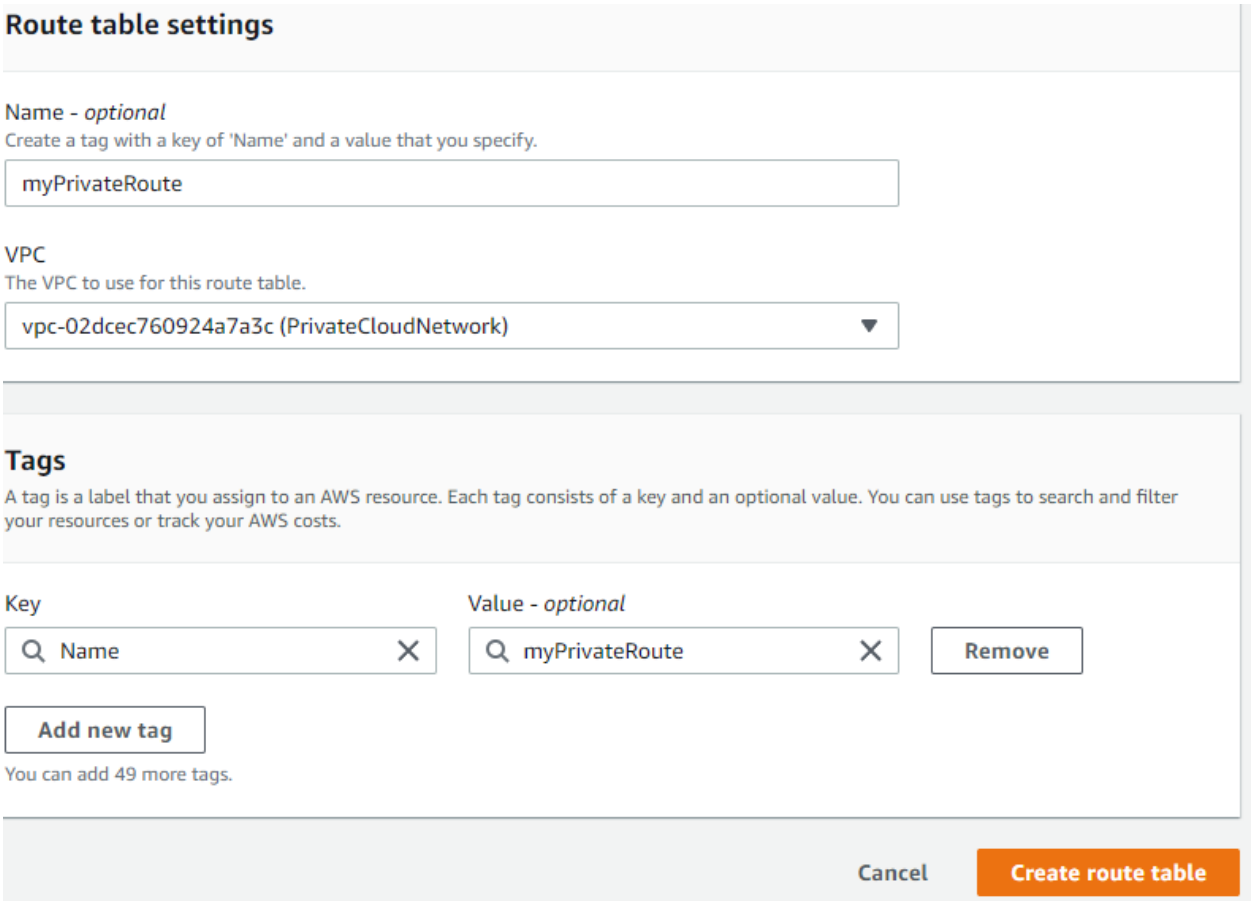



- **Remember to attach your IGW!**

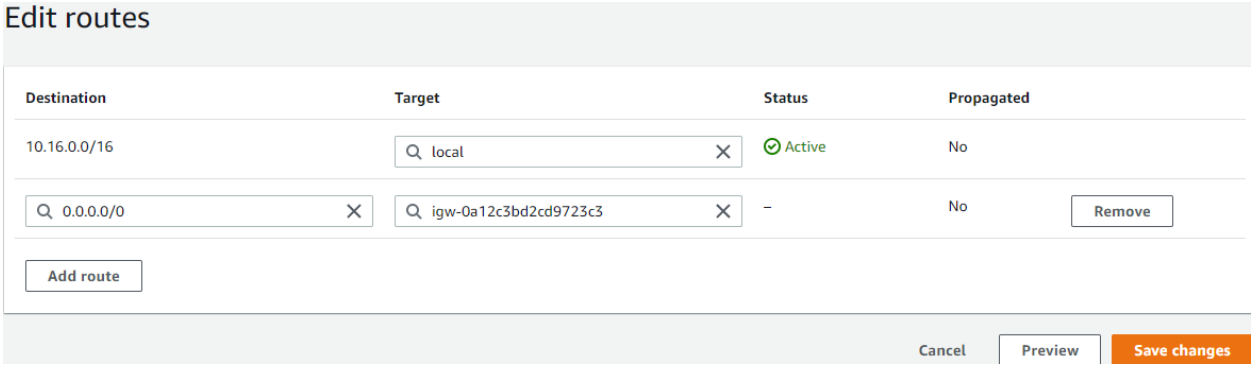



- Should show attached like below with the correct VPC

- **Route tables: we are going to create destinations for our addresses that are not wanted within our VPC outbound towards the IGW**

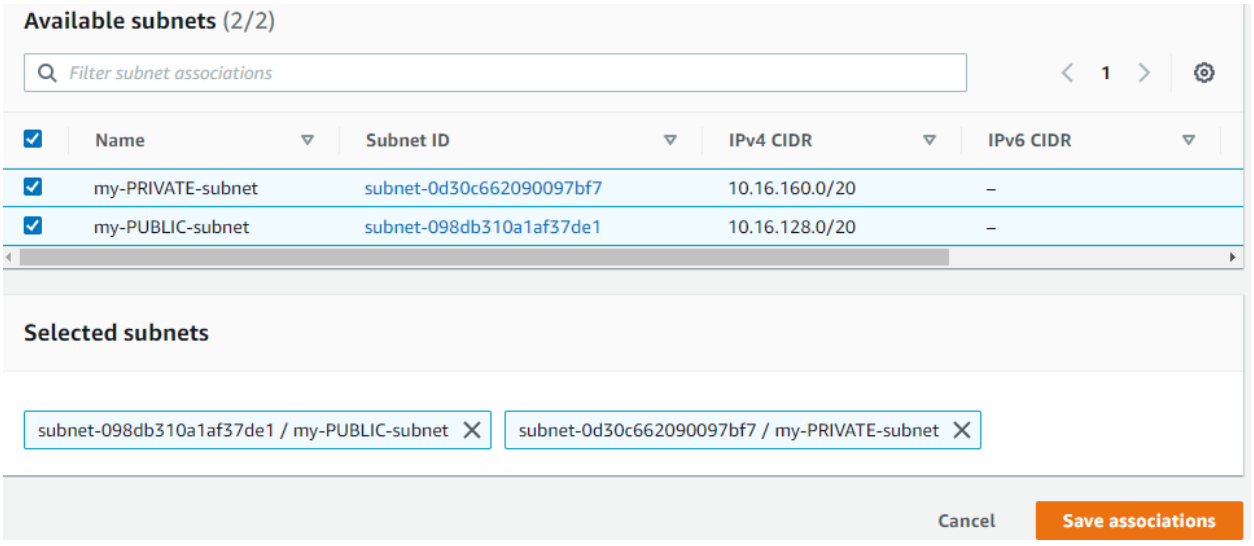



- **So we need to now edit the routes so we can add a destination for our internet gateway. By default it creates a local target meaning everything within the private VPC and by default it uses the IP range that is associated with the VPC.  With the proposed destinations it shows anything that is not in the IPv4 CIDR range will be sent to our IGW.**
- **This is why we told our public subnet to assign public IP addresses for instances created, so traffic can be sent out for updates.**

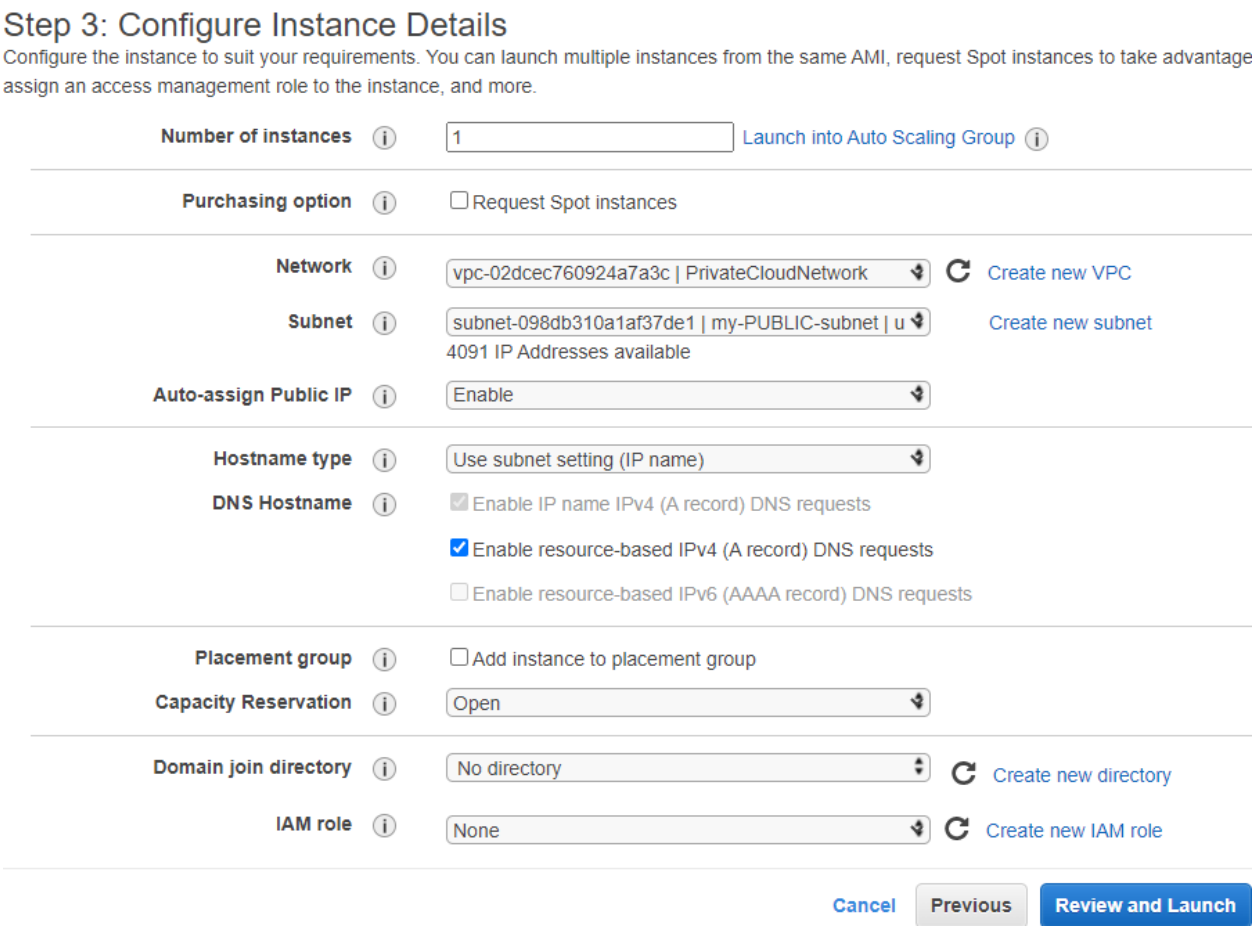



- **By default the route table will not be associated with any subnets, so you must explicitly attach it. Remember a subnet can only have one route table.**


- **Now our subnets will know how to route traffic.**



- **Bastion host: Create an EC2 instance and make sure it is in your private VPC and public subnet as this is going to be our jump box we SSH into then allows us to be forwarded into the private instance.**




- **Remember to change the source IP accordingly, in my case I'm going to use 'MY IP'.**


- **SSH time, but before we do that you must change permissions for your pem file for the bastion host. For Linux it's easy just do chmod 400 keyname.pem but for windows it's a few lines in the terminal. I used PowerShell and finally after the few lines we can connect to the instance via SSH because we have the necessary permissions otherwise you will be denied for being too open.**

```
PS C:\Users\ginoo\Documents\awsCredentials> icacls.exe admin_ssh.pem /grant:r "$($env:username):(r)"
processed file: admin_ssh.pem
Successfully processed 1 files; Failed processing 0 files
PS C:\Users\ginoo\Documents\awsCredentials> icacls.exe admin_ssh.pem //inheritance:r
Invalid parameter "//inheritance:r"
PS C:\Users\ginoo\Documents\awsCredentials> icacls.exe admin_ssh.pem /inheritance:r
processed file: admin_ssh.pem
Successfully processed 1 files; Failed processing 0 files
PS C:\Users\ginoo\Documents\awsCredentials> ssh -i "admin_ssh.pem" ec2-user@3.85.223.167
The authenticity of host '3.85.223.167 (3.85.223.167)' can't be established.
ECDSA key fingerprint is SHA256:ieEmsUDRDy6sZtAi7XtPOFUSHxo1VsKrW8cf1dPtXEE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.85.223.167' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-16-137-13 ~]$
```

- **SSH Agent must be enabled, on Windows it is a service in stop mode, must start it but it does have a few bugs when starting so change the start up type to manual which means you choose to start it and you should have no problems like below:**

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> Get-Service ssh-agent

Status   Name               DisplayName
------   ----               -----------
Stopped  ssh-agent          OpenSSH Authentication Agent


PS C:\Windows\system32> Start-Service ssh-agent
Start-Service : Service 'OpenSSH Authentication Agent (ssh-agent)' cannot be started due to the following error:
Cannot start service ssh-agent on computer '.'.
At line:1 char:1
+ Start-Service ssh-agent
+ ~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError: (System.ServiceProcess.ServiceController:ServiceController) [Start-Service],
   ServiceCommandException
    + FullyQualifiedErrorId : CouldNotStartService,Microsoft.PowerShell.Commands.StartServiceCommand

PS C:\Windows\system32> Set-Service ssh-agent -StartupType Manual
PS C:\Windows\system32> Start-Service ssh-agent
PS C:\Windows\system32> Get-Service ssh-agent

Status   Name               DisplayName
------   ----               -----------
Running  ssh-agent          OpenSSH Authentication Agent
```

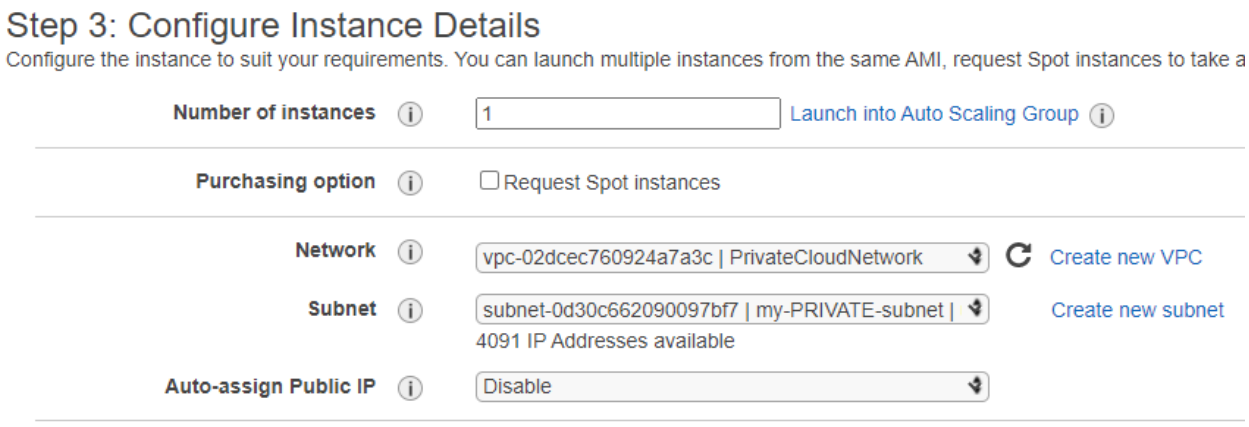
- **Lastly, we must add the SSH key to the Agent via ssh-add keyname.pem**

```
PS C:\> cd .\Users\ginoo\Documents\awsCredentials\
PS C:\Users\ginoo\Documents\awsCredentials> ssh-add admin_ssh.pem
Identity added: admin_ssh.pem (admin_ssh.pem)
PS C:\Users\ginoo\Documents\awsCredentials>
```

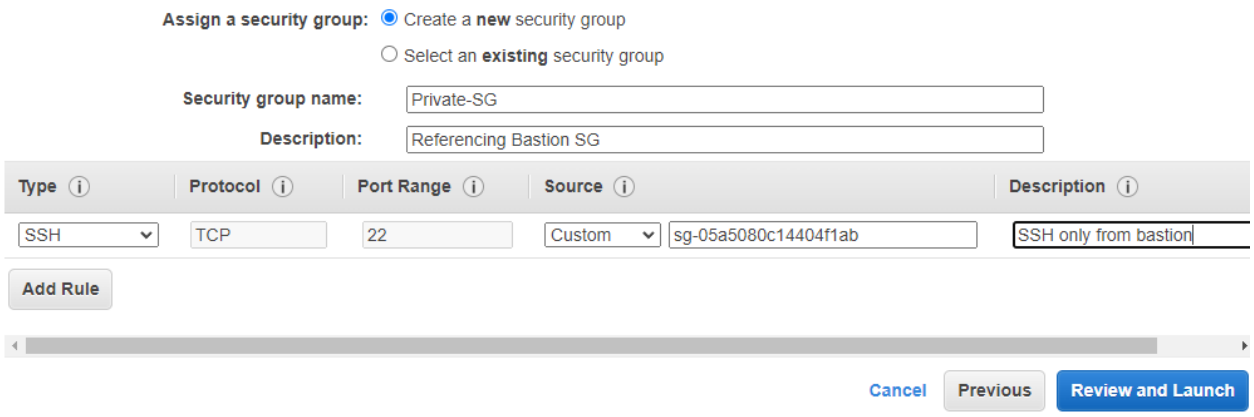
- **So now when we SSH in, we add -A into the SSH command .**

```
PS C:\Users\ginoo\Documents\awsCredentials> ssh -A -i "admin_ssh.pem" ec2-user@3.85.223.167
```

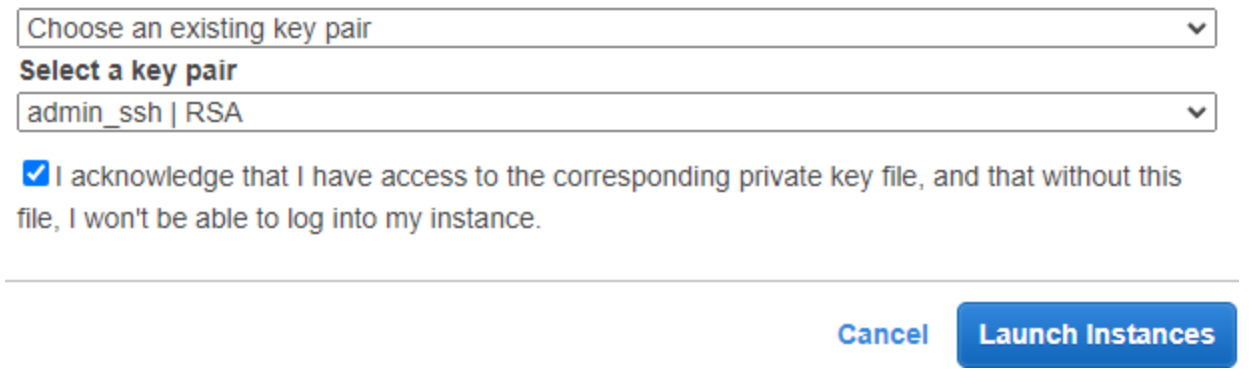
- **Now that we have a running bastion host we can create the private instance that we are hiding away in a private subnet away from the public therefore remember to disable auto-assign public IP.**



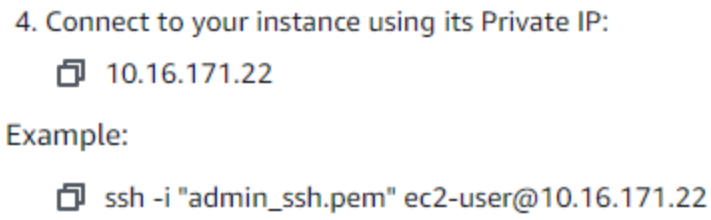

- **For the Security Group, we are going to reference the SG from bastion host therefore we are only allowing SSH traffic that is equivalent to the one attached to bastion. This means we can SSH from bastion and nothing else.**



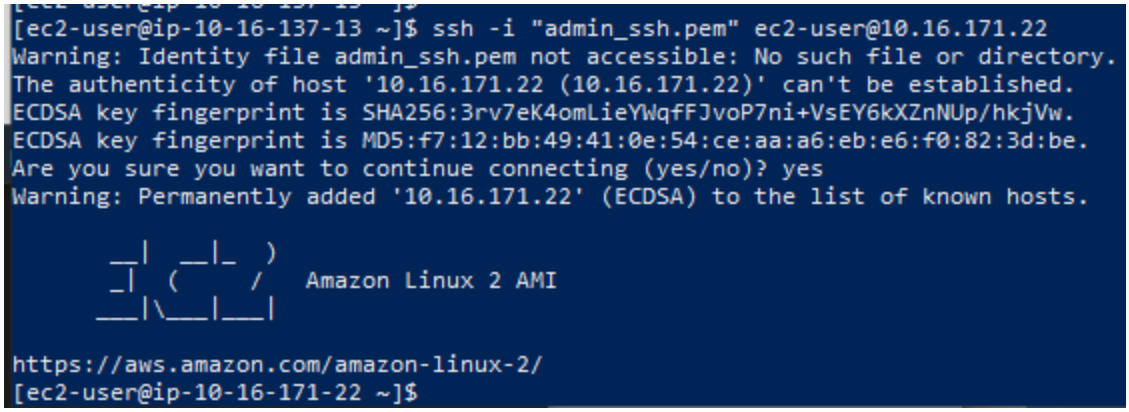

- **Lastly, attach the same pem key**



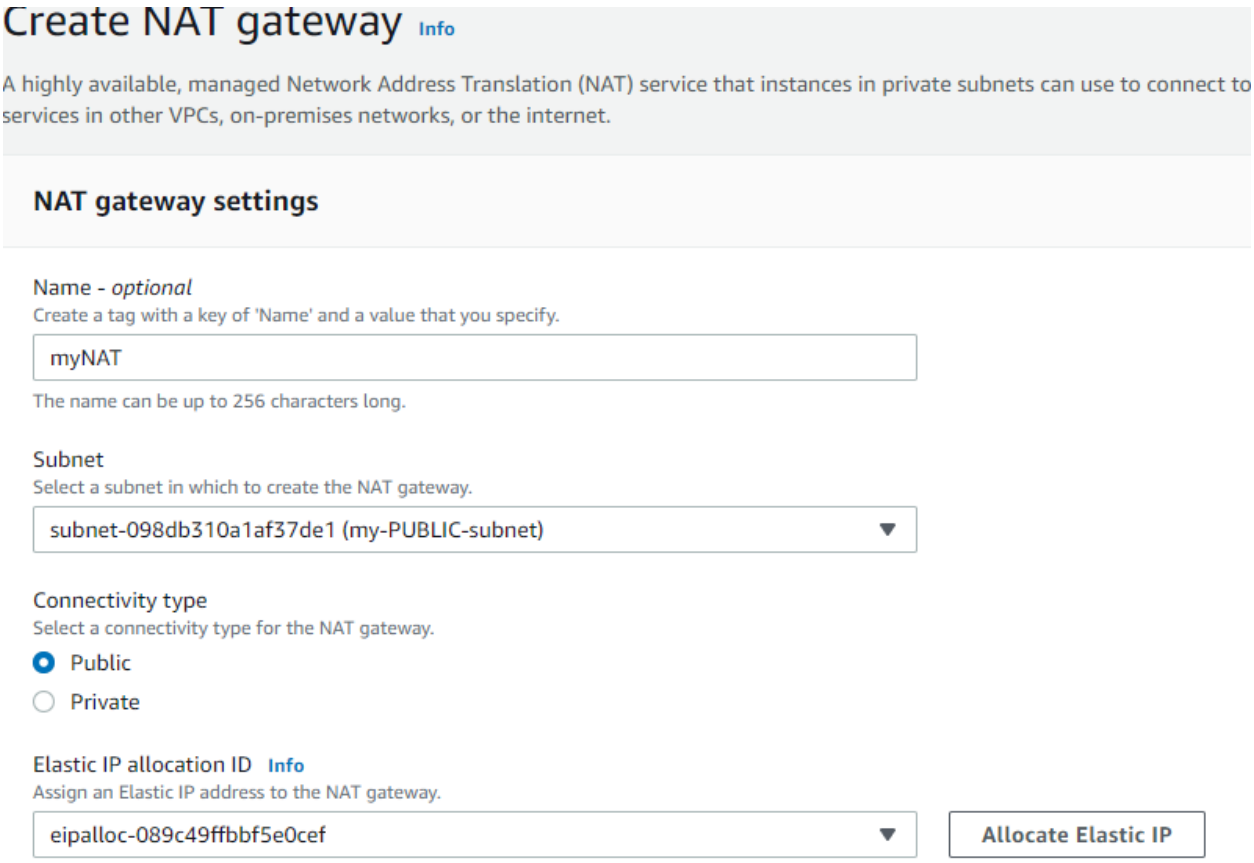
- **When we connect to the private EC2 instance which is does not have a public IP to reach, we can connect via private IP. We can use the private IP internally, so we can connect to it from the bastion host which is connected via public IP using the public internet.**
- **As you can see it notes Private instead of Public for this instance.**




- **Within the shell of the bastion instance we can now SSH into the private one.**
- **We now have access from the public internet specifically from my IP to a privately secured instance unavailable to anyone else.**




- **We now need to add a NAT Gateway to have outbound internet access so we can retrieve logs and other data. Remember NAT Gateways are AZ specific. We must add one into the public subnet so the private subnet can route its destination addresses to NAT is able to talk to IGW**

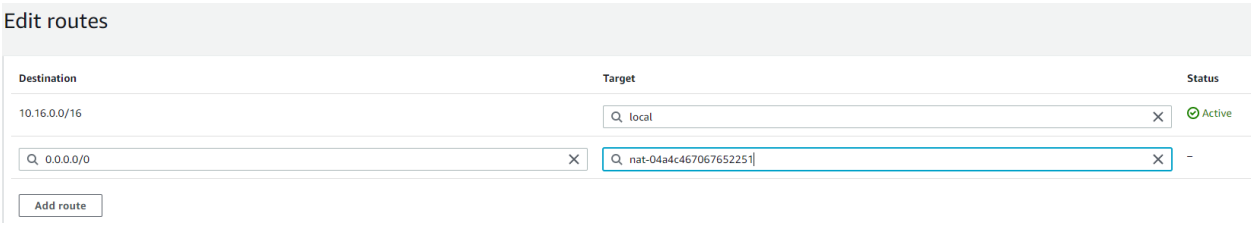



- **Right now we have no connection from the private instance due to no internet access via public IP as shown by pinging.**

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-16-171-22 ~]$ ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
^C
--- 1.1.1.1 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8173ms
```

- **We need to create a separate route table for the private subnet because we are attaching a NAT Gateway to give a private instance outbound internet access by rerouting traffic to the NAT Gateway in the public subnet.**
- **The first route table will be associated now with only the public subnet which has destinations internally and external to the IGW**

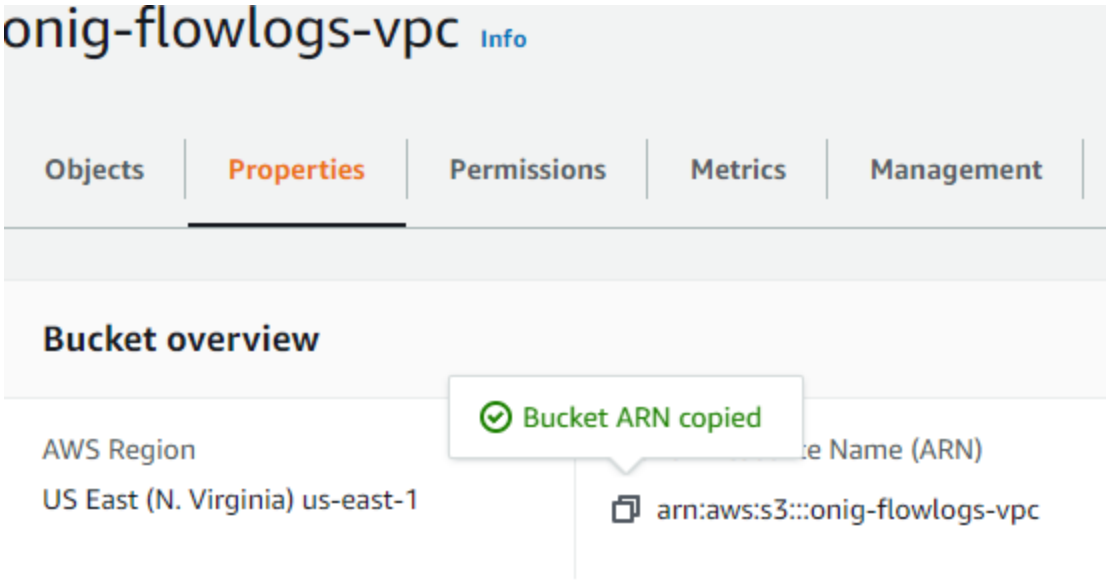
- **The new route table is now associated with the private subnet which has destinations internally and the NAT Gateway we made in the public subnet.**
- **When you associate it with the private subnet it will replace the previous route table.**

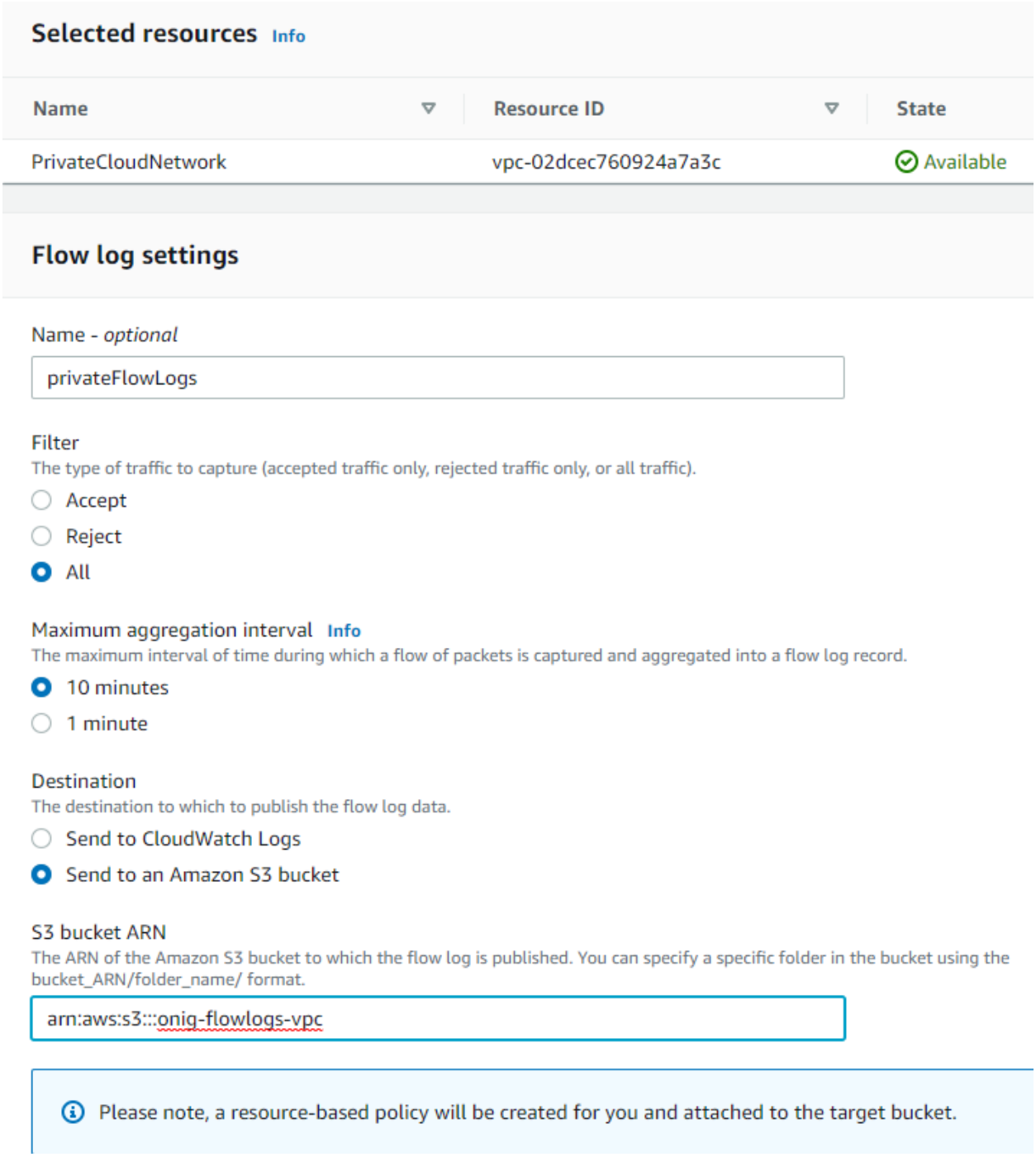



- **As shown below, ping now works. We have outbound connection from the private instance.**

```
[ec2-user@ip-10-16-171-22 ~]$ ping google.com
PING google.com (142.250.65.78) 56(84) bytes of data.
64 bytes from iad23s91-in-f14.1e100.net (142.250.65.78): icmp_seq=1 ttl=111 time=3.62 ms
64 bytes from iad23s91-in-f14.1e100.net (142.250.65.78): icmp_seq=2 ttl=111 time=2.47 ms
64 bytes from iad23s91-in-f14.1e100.net (142.250.65.78): icmp_seq=3 ttl=111 time=2.41 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 2.412/2.839/3.629/0.562 ms
```

- **Finally we need to create VPC Flow logs to see IP information about accessing our private VPC**
- **We're going to store them  in S3, so create a bucket**


onig-flowlogs-vpc
Info
Objects
Properties
Permissions
Metrics
Management
Bucket overview
Bucket ARN copied
AWS Region
US East (N. Virginia) us-east-1
e Name (ARN)
arn:aws:s3:::onig-flowlogs-vpc




- **Now we can SSH into the bastion instance and then the private one to create some logs.**

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> ssh -A -i "admin_ssh.pem" ec2-user@3.85.223.167
Warning: Identity file admin_ssh.pem not accessible: No such file or directory.
Last login:

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
Last login:

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-16-137-13 ~]$ ssh -i "admin_ssh.pem" ec2-user@10.16.171.22
Warning: Identity file admin_ssh.pem not accessible: No such file or directory.
Last login: Wed Dec  1 13:10:16 2021 from 10.16.137.13

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-16-171-22 ~]$ ^C
[ec2-user@ip-10-16-171-22 ~]$ exit
logout
Connection to 10.16.171.22 closed.
[ec2-user@ip-10-16-137-13 ~]$ exit
logout
Connection to 3.85.223.167 closed.
PS C:\Windows\system32>
```

- **As we can see logs have generated. DONE.**