Lab : Virtual Private Cloud(VPC)

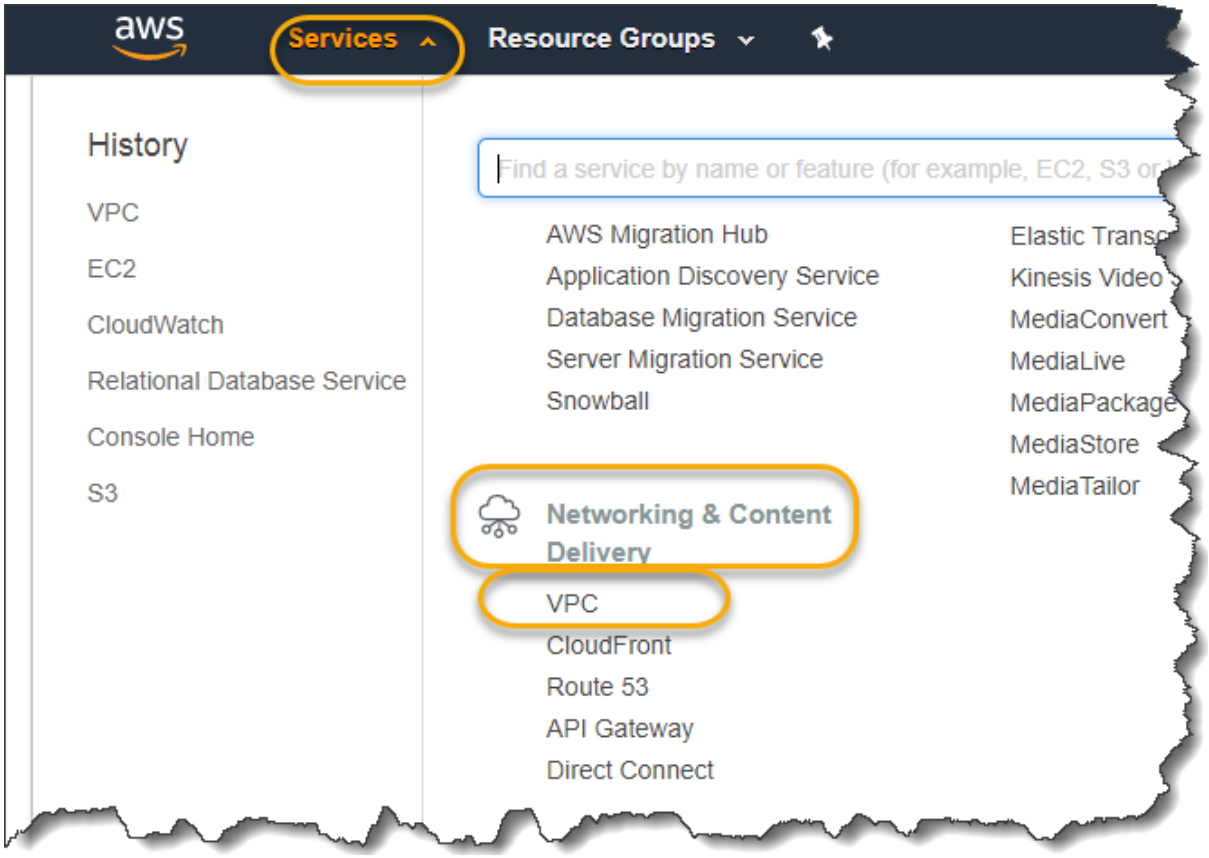
In this lab we will create a VPC, create subnets, configure public and private access, configure network address translation and then perform tests to demonstrate the working of our configuration.

Task Breakdown

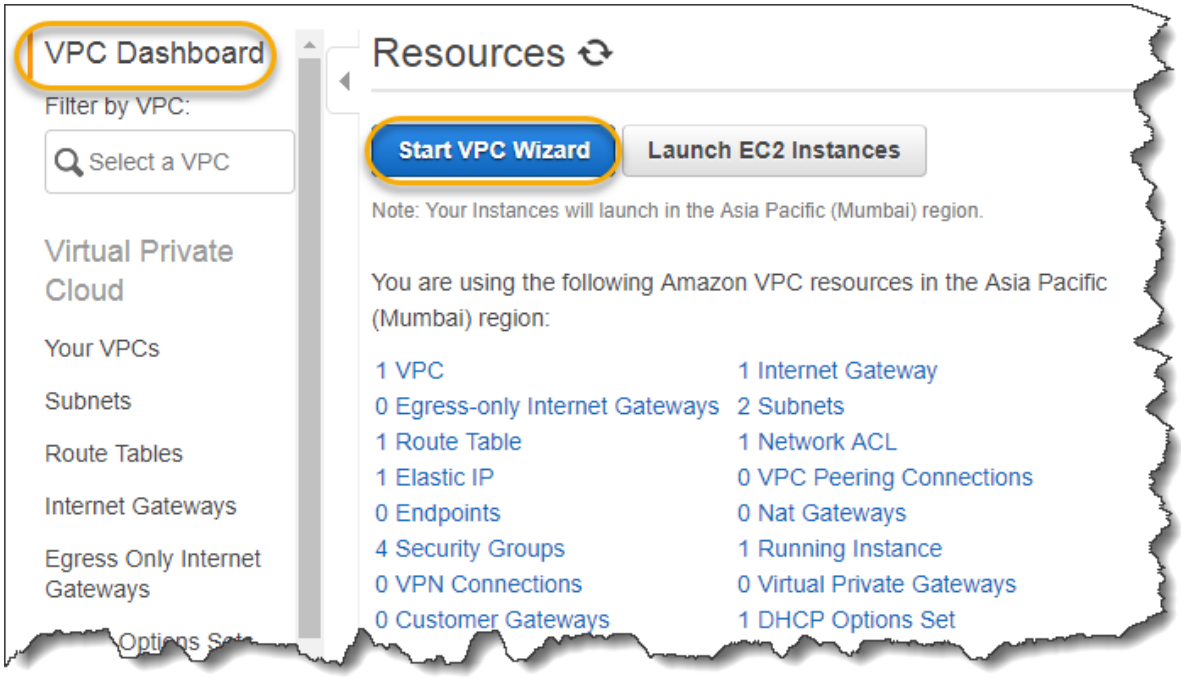
- Task 1: Create a VPC with a Public subnet
- Task 2: Create a Private subnet
- Task 3: Launch a web server into the Public subnet
- Task 4: Launch a database server into the Private subnet
- Task 5: Test out accessibility of subnets
- Task 6: Launch a NAT instance.
- Task 7: Modify the private route table to Route traffic to NAT
- Task 8: Download Security Credentials
- Task 9: Setup the AWS CLI
- Task 10: Manage VPC through CLI

Task 1: Create a VPC with a Public subnet

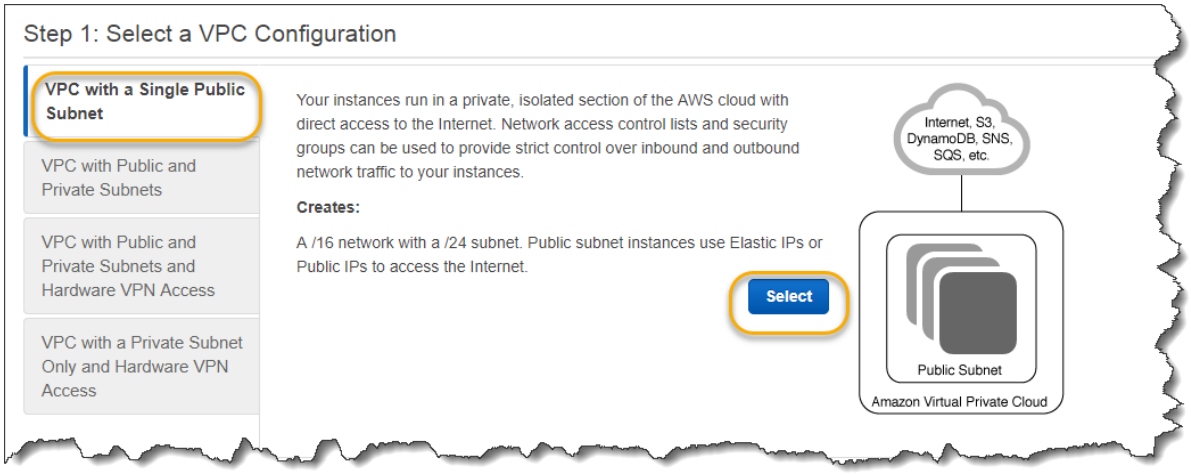
1. On the top of your screen Click **Services**, and under **Networking & Content Delivery** Click **VPC**



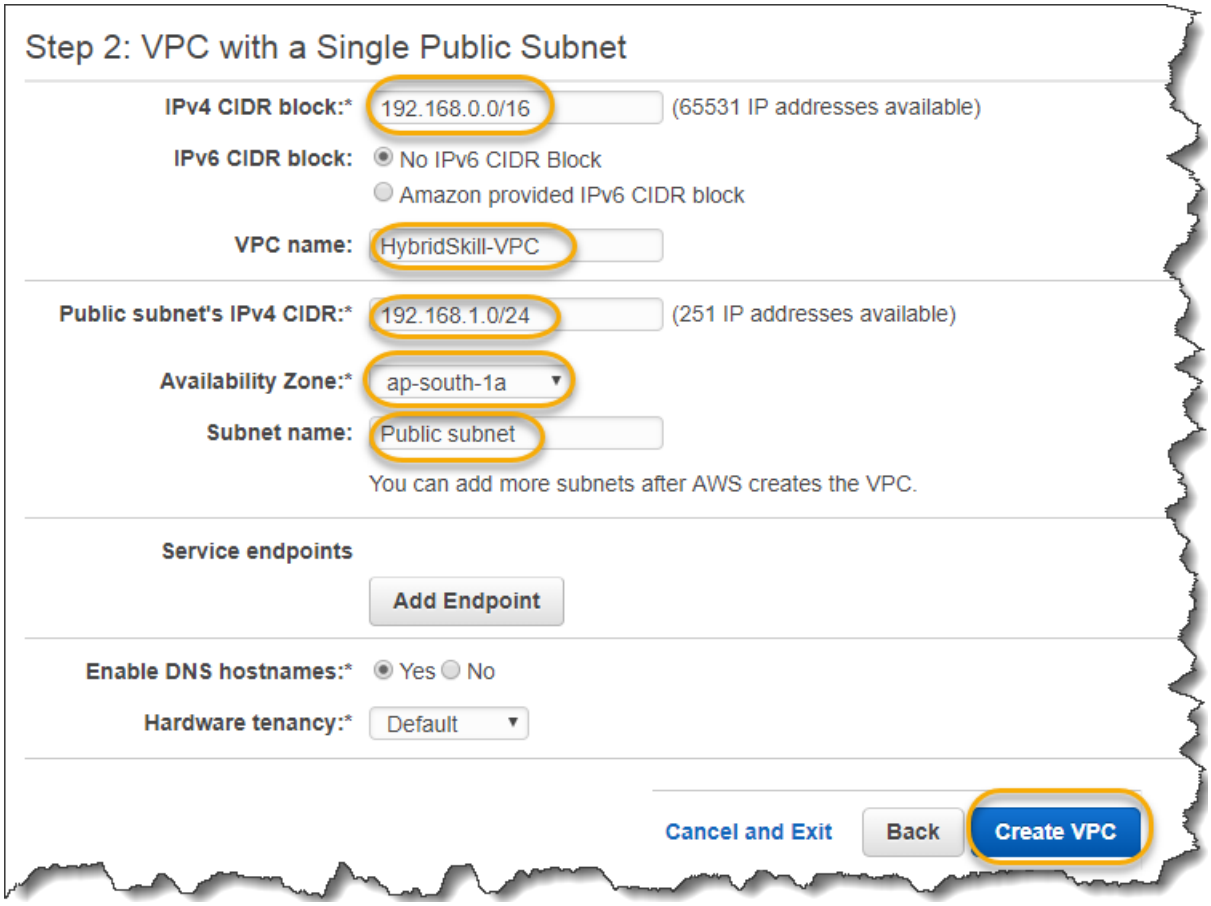
2. On the Main **VPC Dashboard** click **Start VPC Wizard**



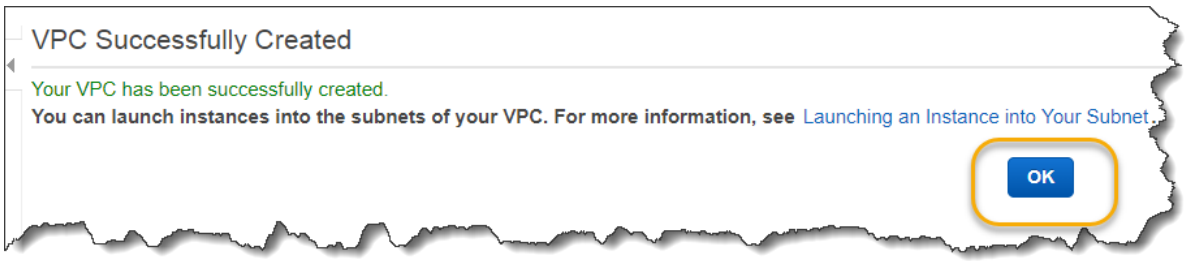
3. **Select** the configuration includes a **VPC with a Single Public Subnet**



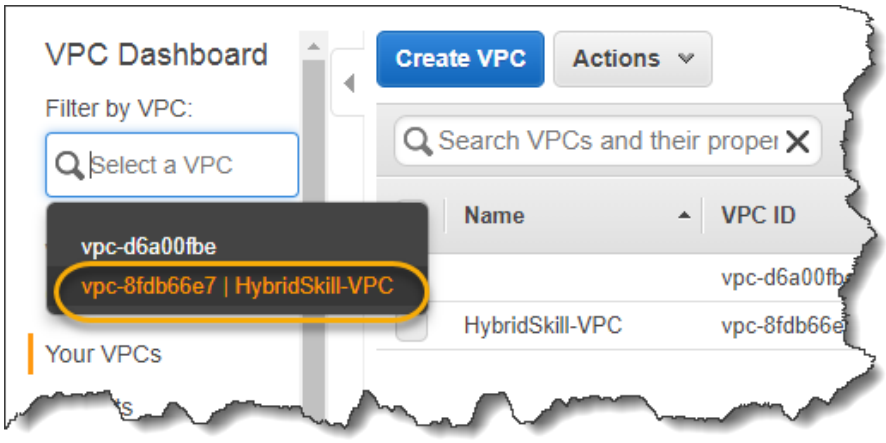
4. Enter **192.168.0.0/16** as the **IPv4 CIDR block** of the VPC and enter **HybridSkill-VPC** as the **VPC name.** Enter **192.168.1.0/24** as the **IPv4 CIDR block** of the subnet, select **ap-south-a (any zone in your region)** as the **Availability Zone** and enter **Public Subnet** as the **Subnet name.** Finally click **Create VPC**



5. Your VPC is created successfully. Click the **OK** button

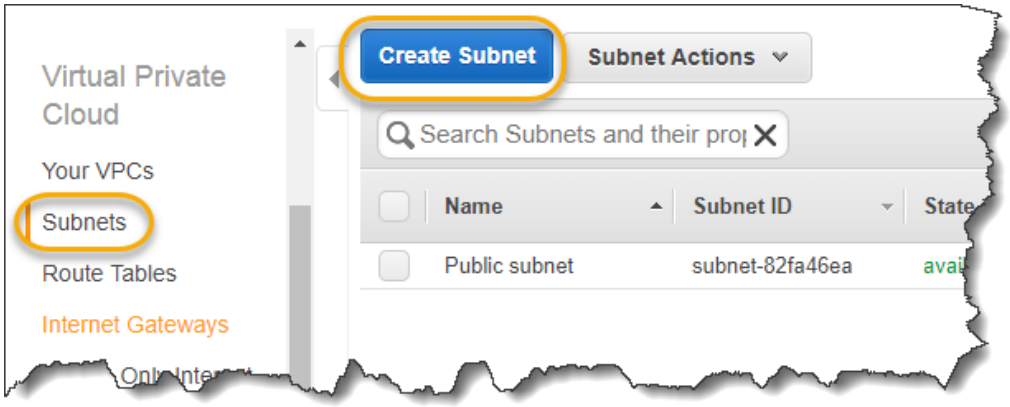


6. In order to prevent confusion between the Default VPC and the one you just created use **the Filter By VPC** option at the top left of your screen. Select Your VPC from the dropdown here.

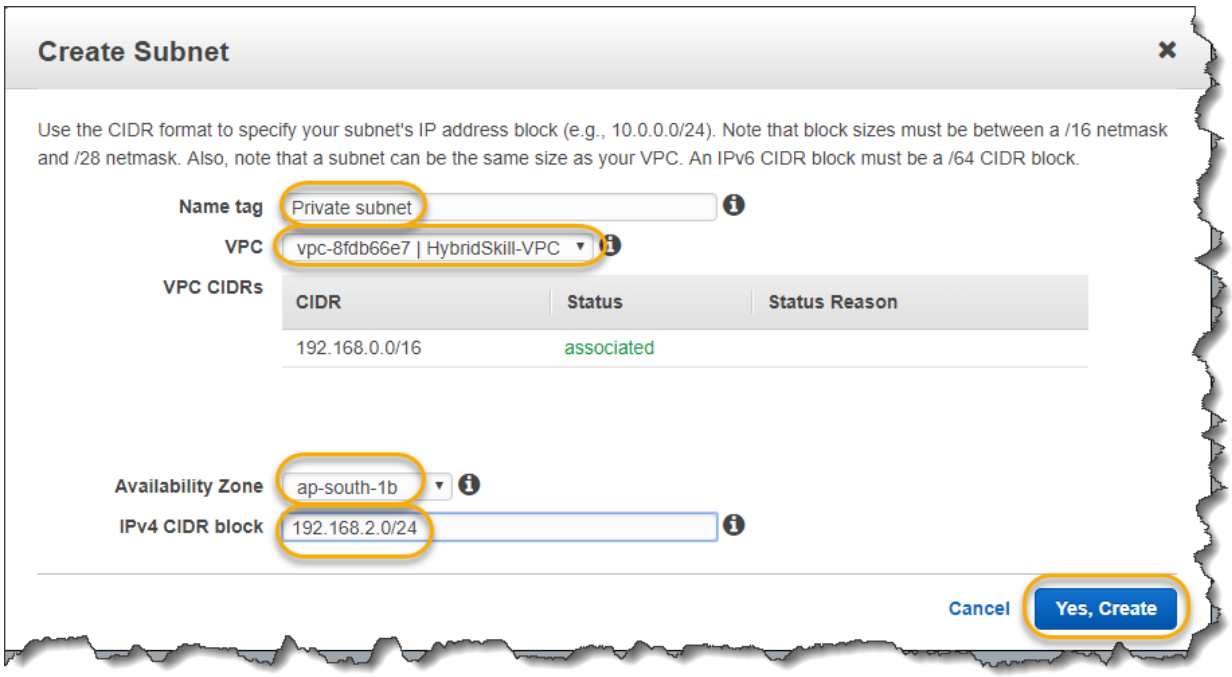


Task 2: Create a Private subnet

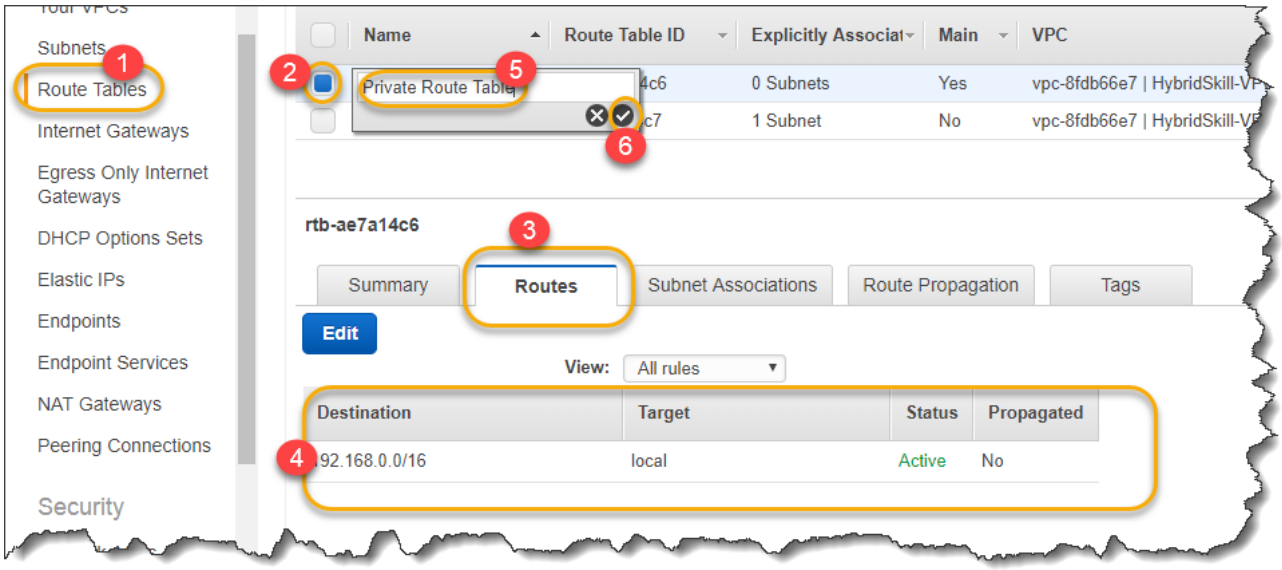
1. Click **Subnets** and then click **Create Subnet.**



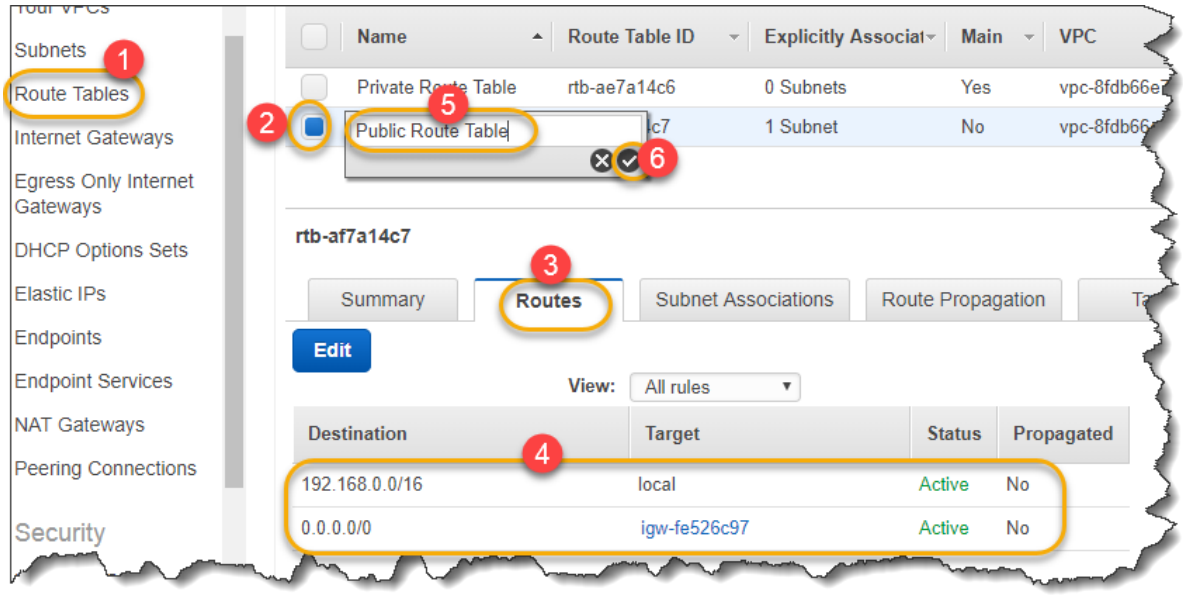
2. Enter **Private subnet** as the **Name** of the subnet, select your **VPC(HybridSkill-VPC)** from the dropdown. Select **ap-south-1b (any AZ from your region)** as the **Availability Zone**. Enter **192.168.2.0/24** as the **IPv4 CIDR block** of the subnet and click **Yes, Create.**



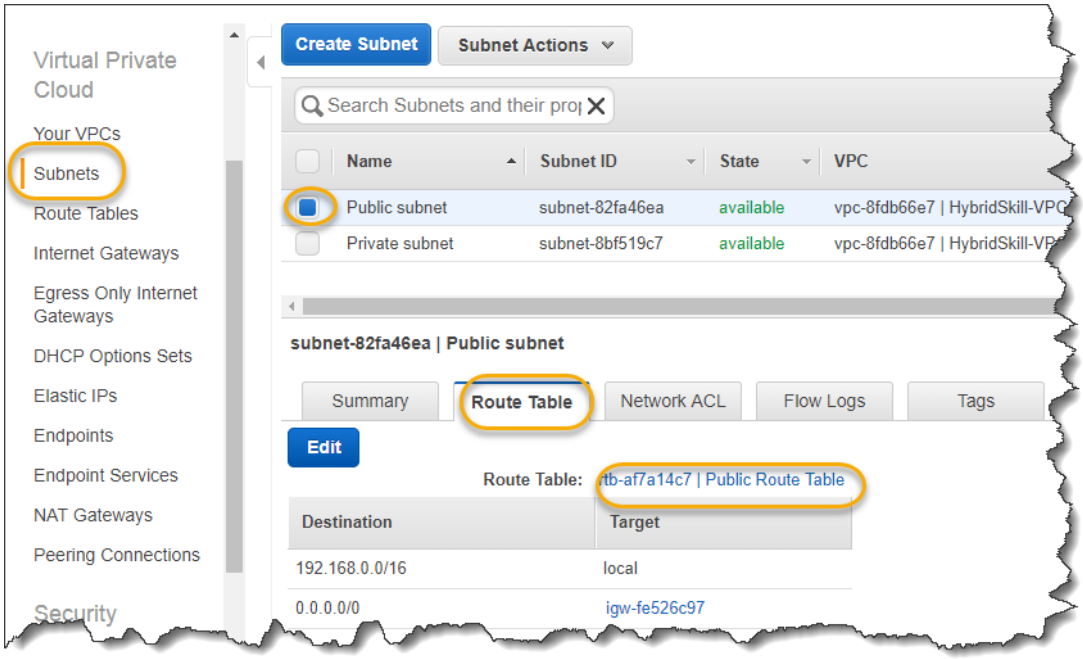
3. Next let's review our Route Tables. Rename them as public and Private to prevent confusion. Select **Route Tables**. Select a route table. On the bottom click **Routes** and if see only **one entry** in the Route Table **(no IGW entry),** then rename it as **Private Route Table.**



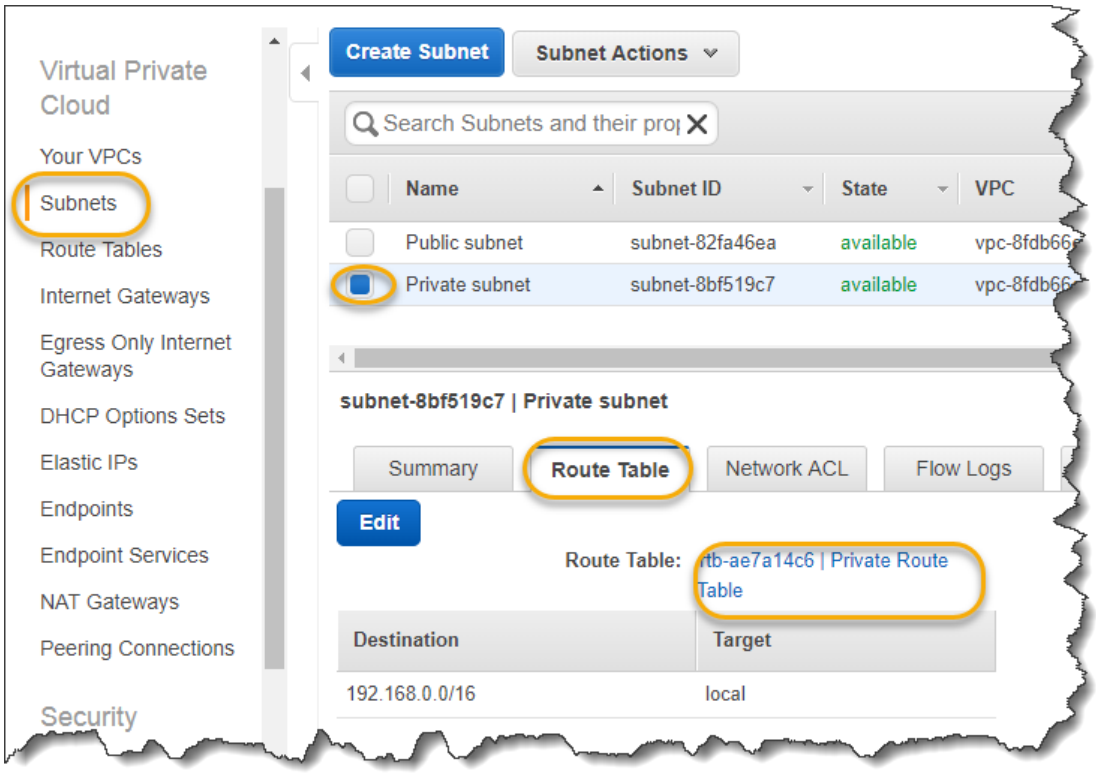
4. Similarly, if you see **2 routes** in the **Route Table (a route to the IGW)** then rename it to **Public Route Table**



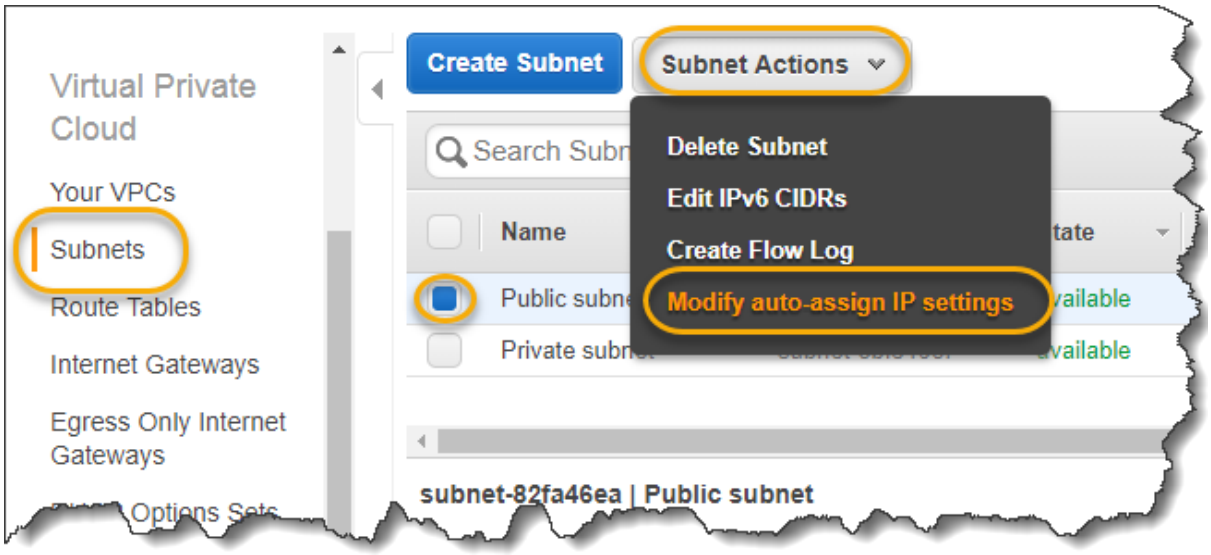
5. Select **Subnets,** select **Public subnet,** click **Route table** tab at the bottom and make sure it is using the **Public Route Table.**



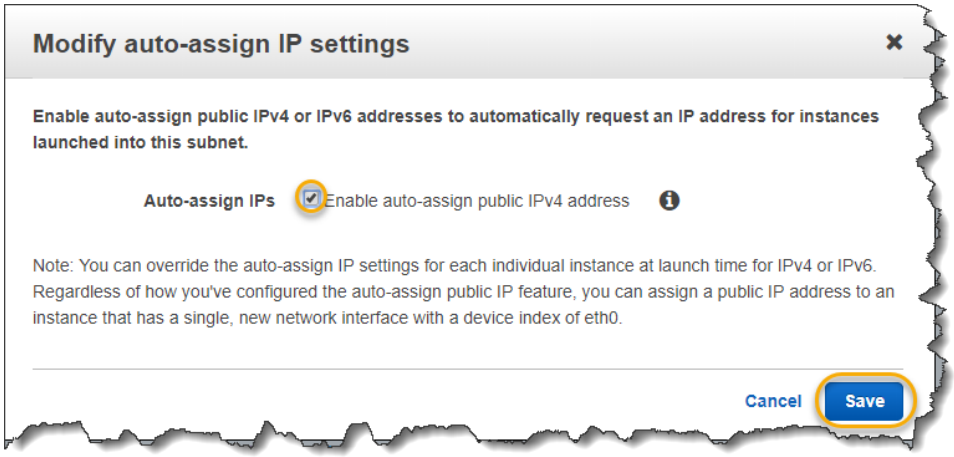
6. Select **Subnets,** select **Private subnet,** click **Route table** tab at the bottom and make sure it is using the **Private Route Table.**



7. Select your **Public Subnet**, click **Subnet Actions** and click **Modify auto-assign IP settings**

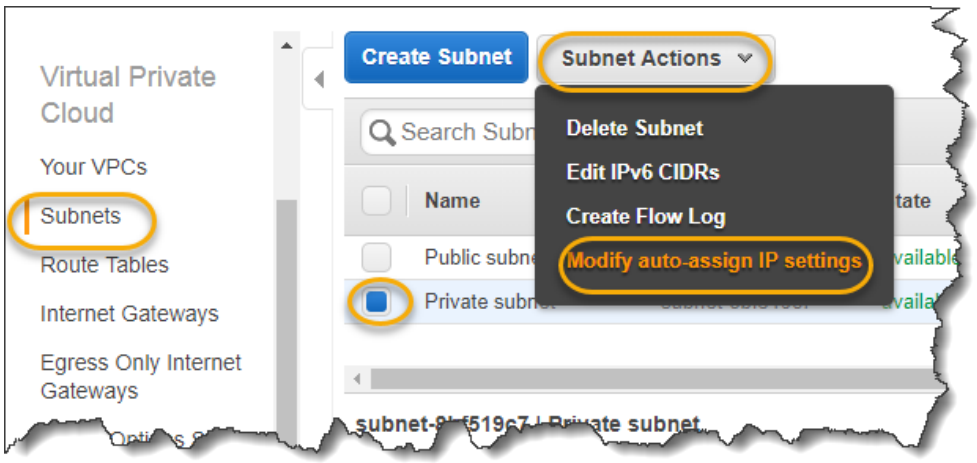


8. Make sure the option of **Enable auto assign public IPv4 address** is checked and click **Save**

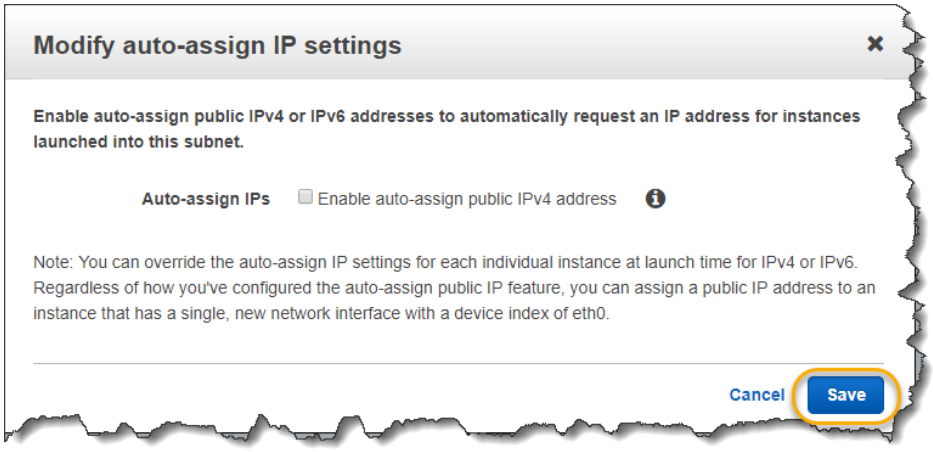


9. Click **Close**

10. Similarly select your **Private Subnet**, click **Subnet Actions** and click **Modify auto-assign IP settings**



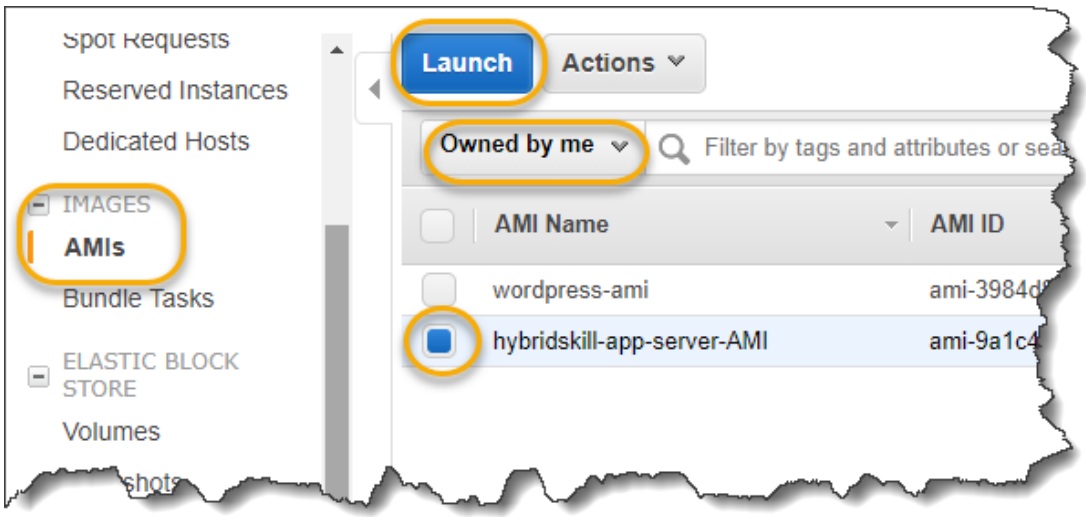
11. Make sure the option of **Enable auto assign public IPv4 address** is unchecked and click **Save**



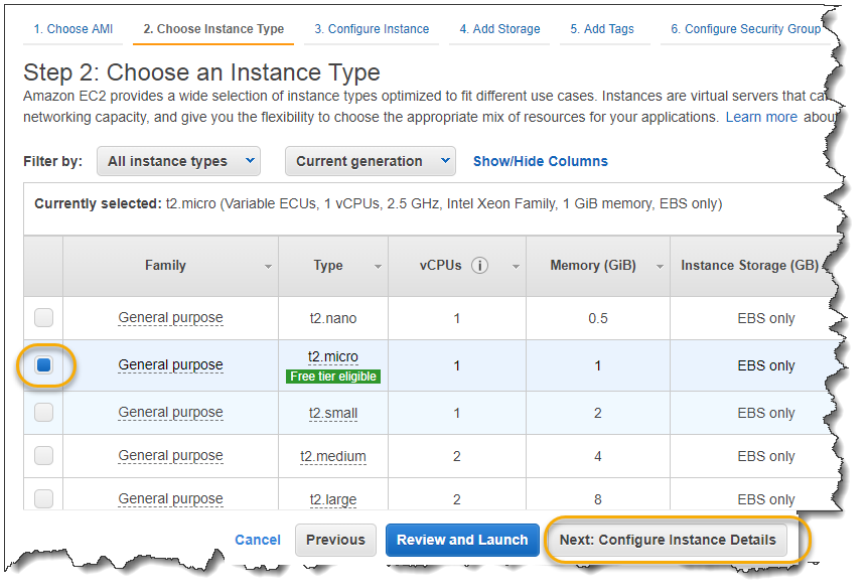
12. Click **Save**

Task 3: Launch a web server into the Public subnet

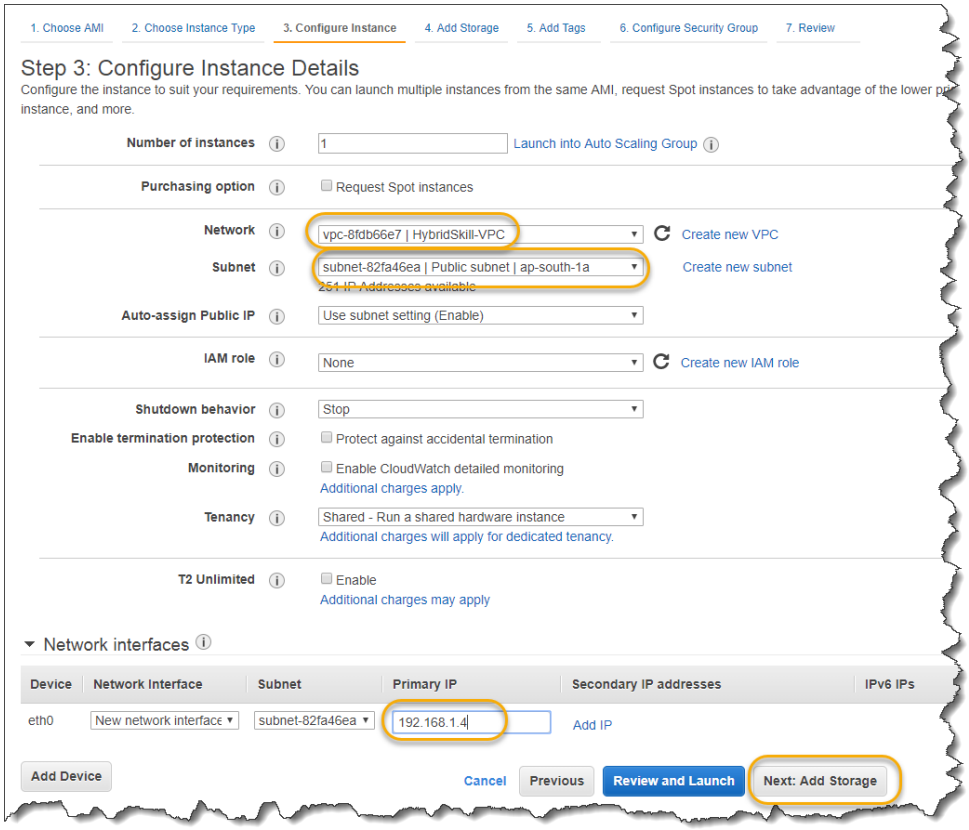
1. Go the **EC2 Dashboard**. Click **AMIs,** select **Owned by me,** select your **Hybrid-skill-app-server-AMI** you created earlier and click **Launch.**



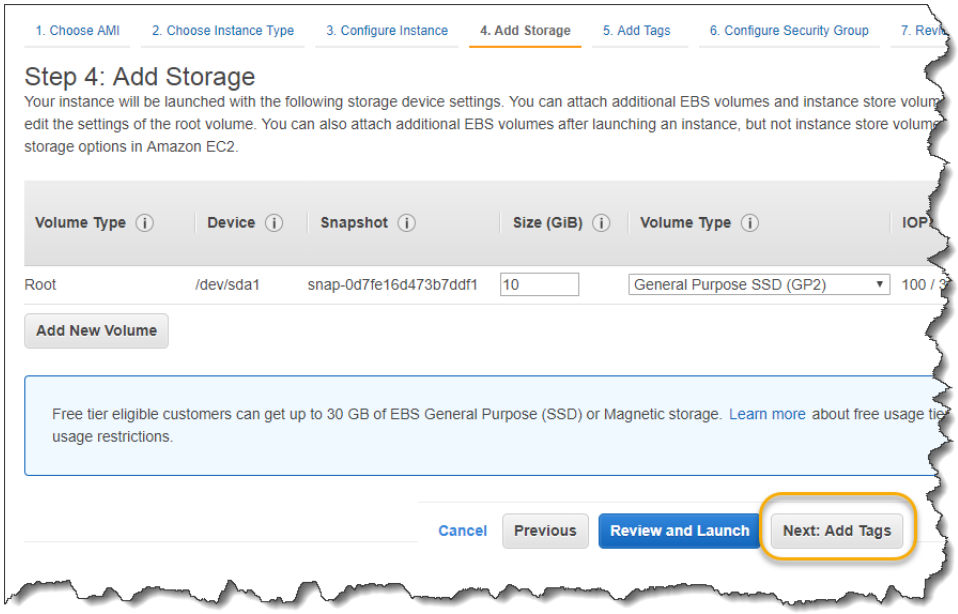
2. Select the instance type as **t2.micro** and click **Next: Configure Instance Details**



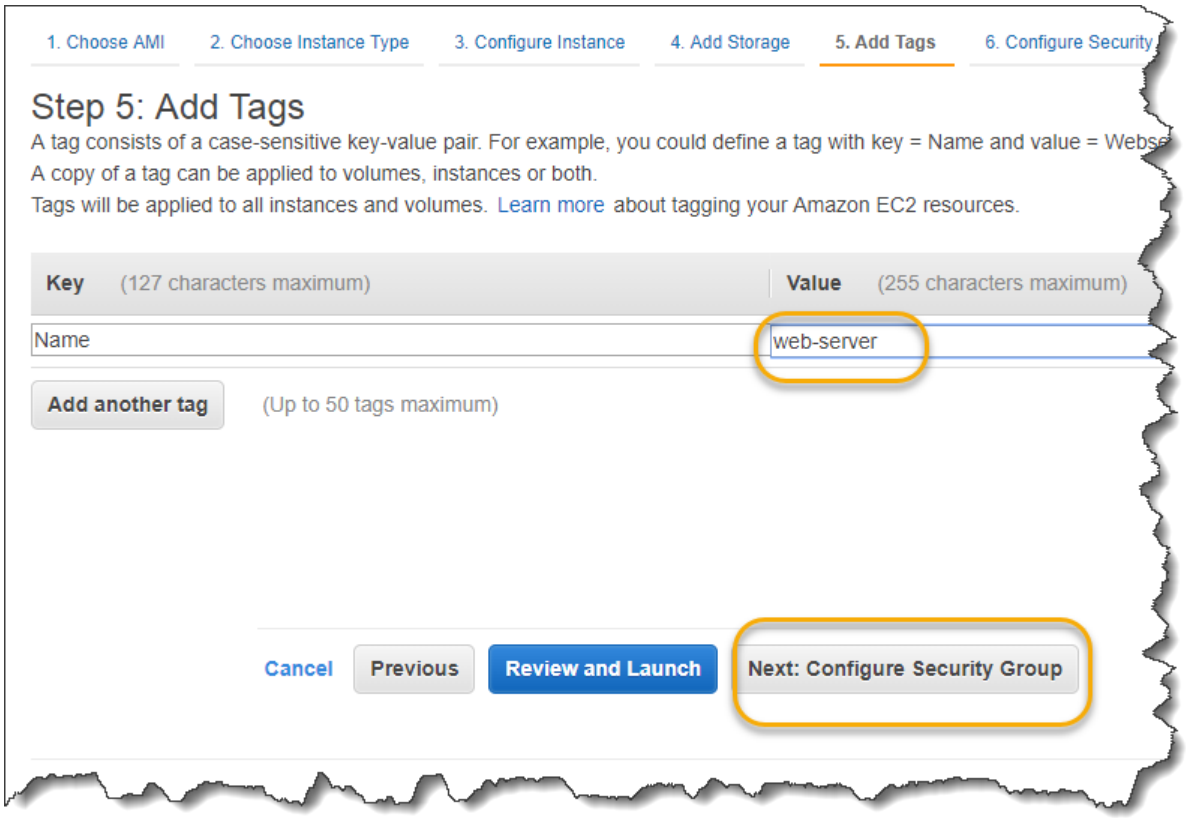
3. For Network select your **HybridSkill-VPC,** select Public Subnet you created earlier from the dropdown. Enter **192.168.1.4** as the **Primary IP** of the server and click **Next: Add Storage**



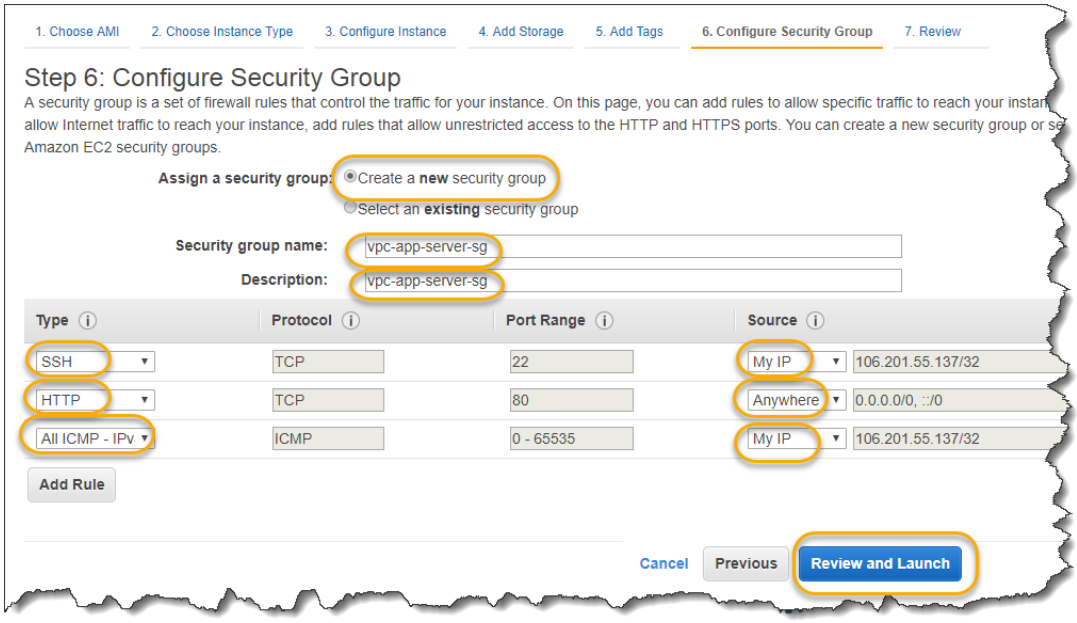
4. Click **Next: Add Tags**



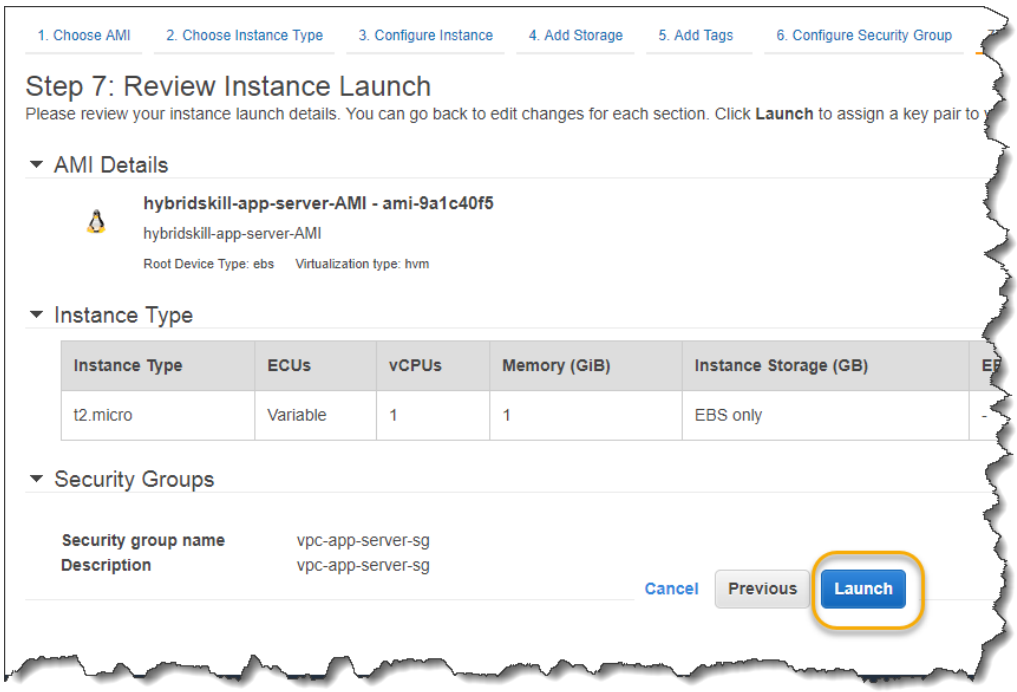
5. Enter **web-server** as the Name tag of the server



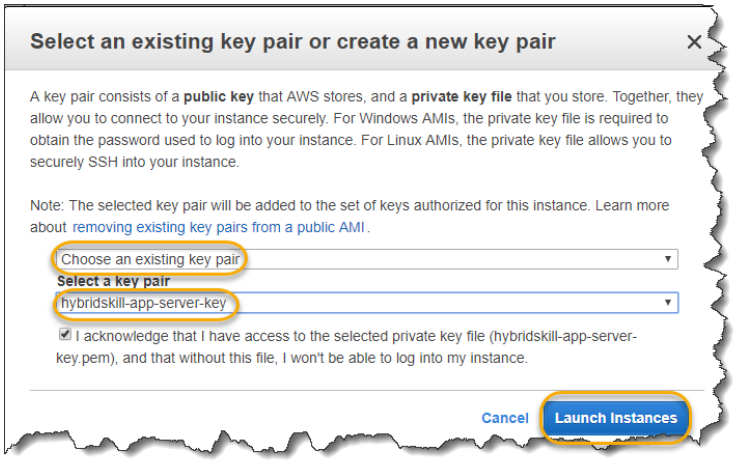
6. Enter **vpc-app-server-sg** as Security group name, enter a description. Click on add rule and from the dropdown add **HTTP, ICMP-IPv4** and **SSH** as rules. For **SSH** select the source as **My IP** from the dropdown. For **HTTP** leave the source as **Anywhere** and for **ICMP-IPv4** select the source as **My IP**. Finally click on **Review and Launch**.



7. Review your settings and click **Launch**

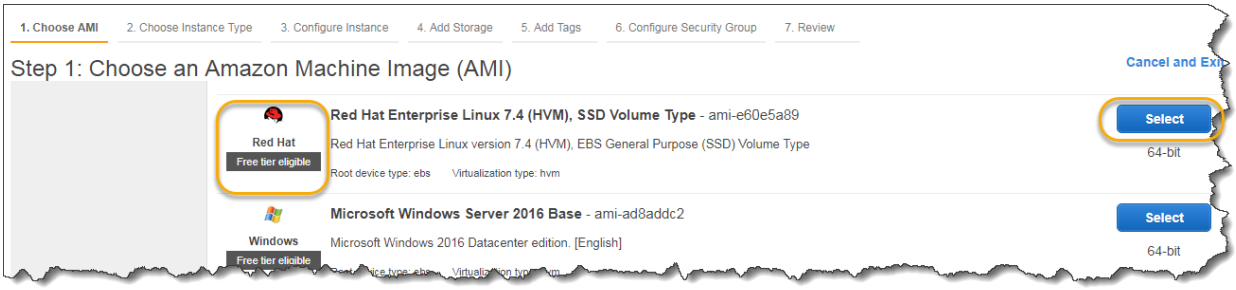


8. Select **Choose an existing key-pair** select the key you created earlier and Click **Launch Instances**

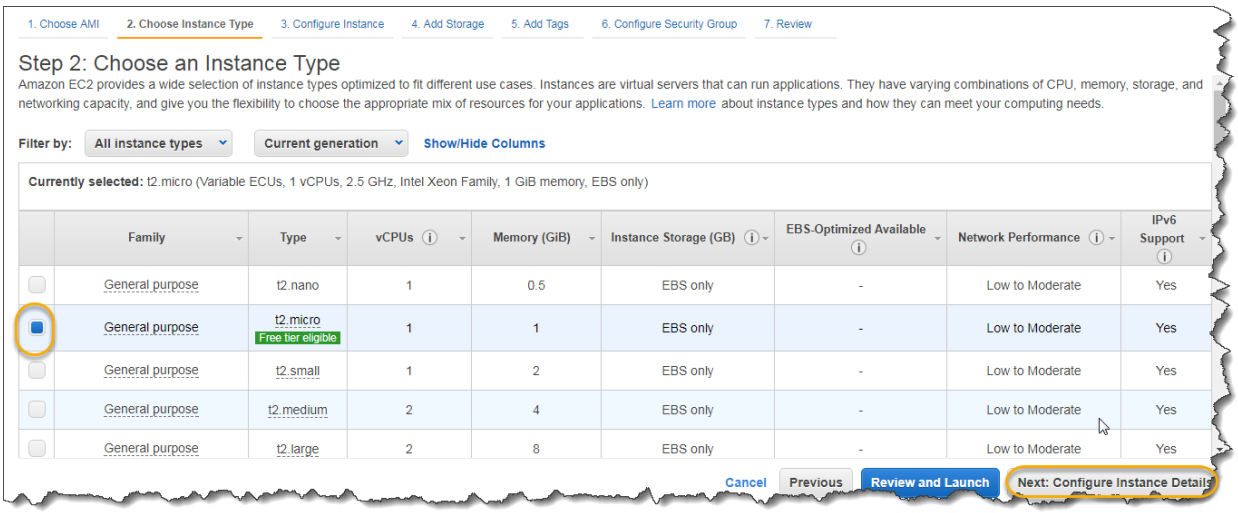


Task 4: Launch a database server into the Private subnet

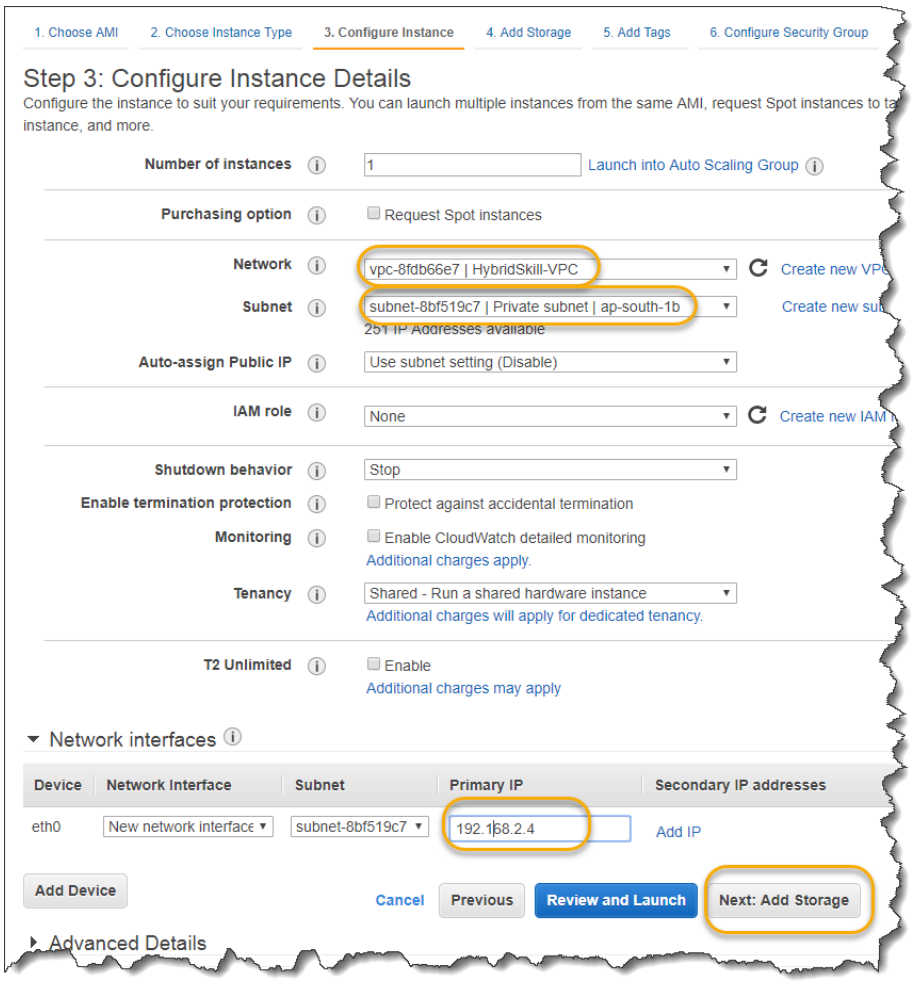
1. Choose **Red Hat Enterprise Linux 7.4** as your **AMI** by clicking the **Select** button.



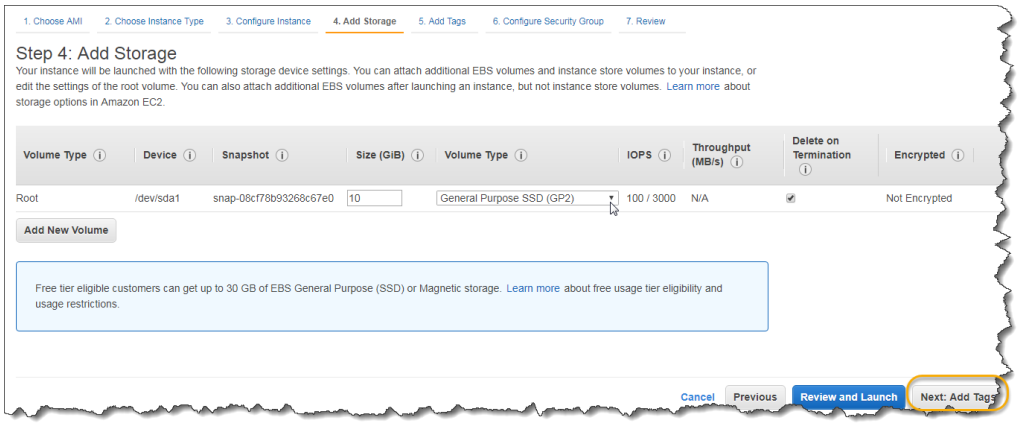
2. Select **t2.micro** as your Instance Type and click **Next: Configure Instance Details**



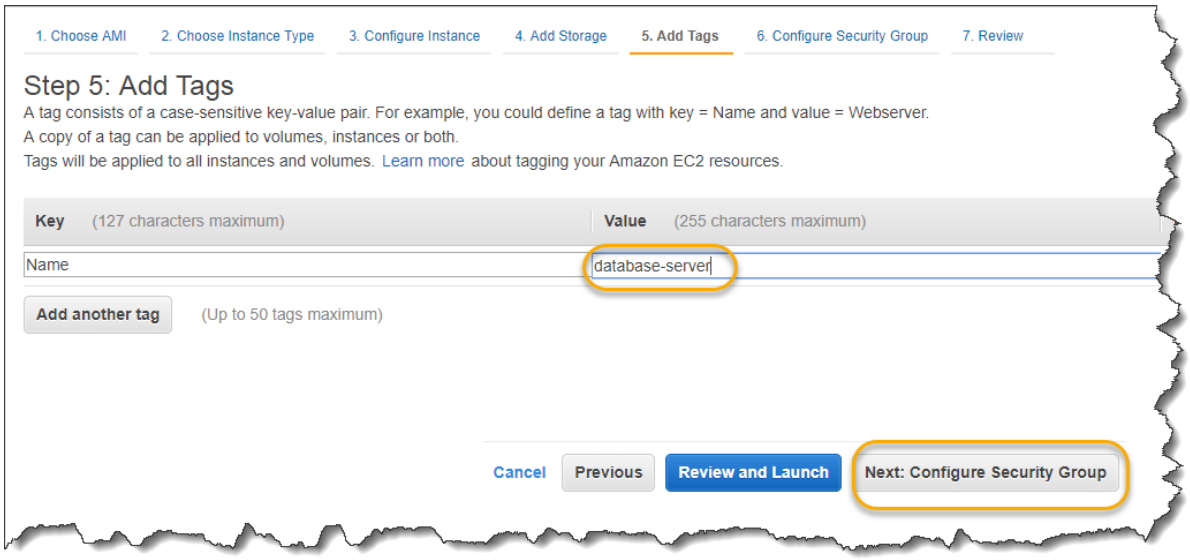
3. For **Network** select your **HybridSkill-VPC,** this time select the **Private Subnet** you created earlier from the dropdown. Enter **192.168.2.4** as the **Primary IP** of the server and click **Next: Add Storage**



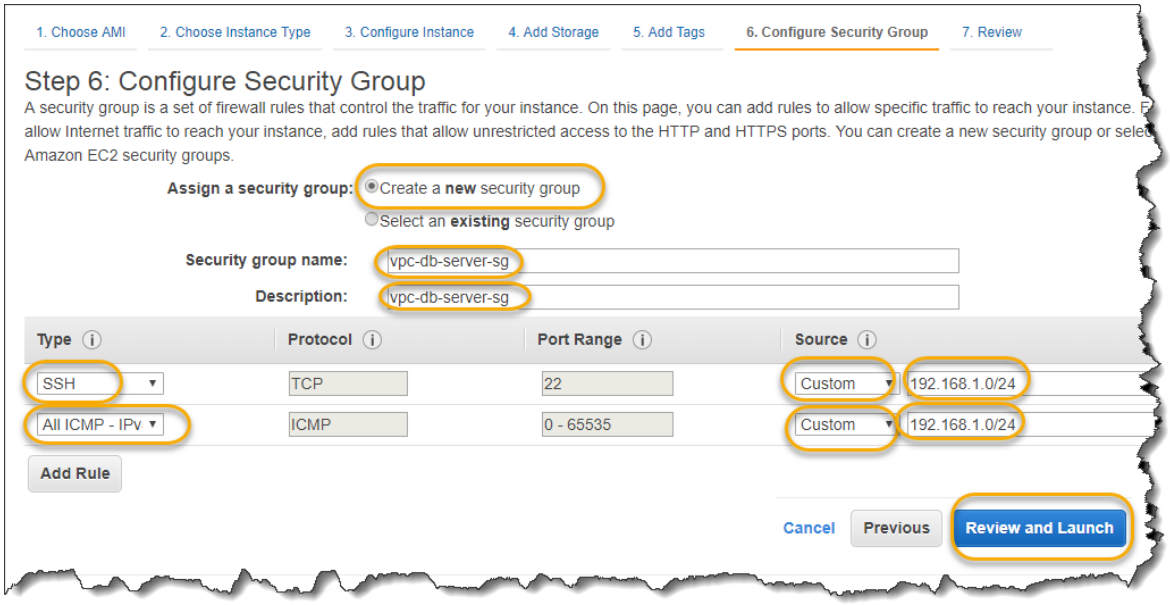
4. Leave all the options as default and click on **Next: Add Tags**



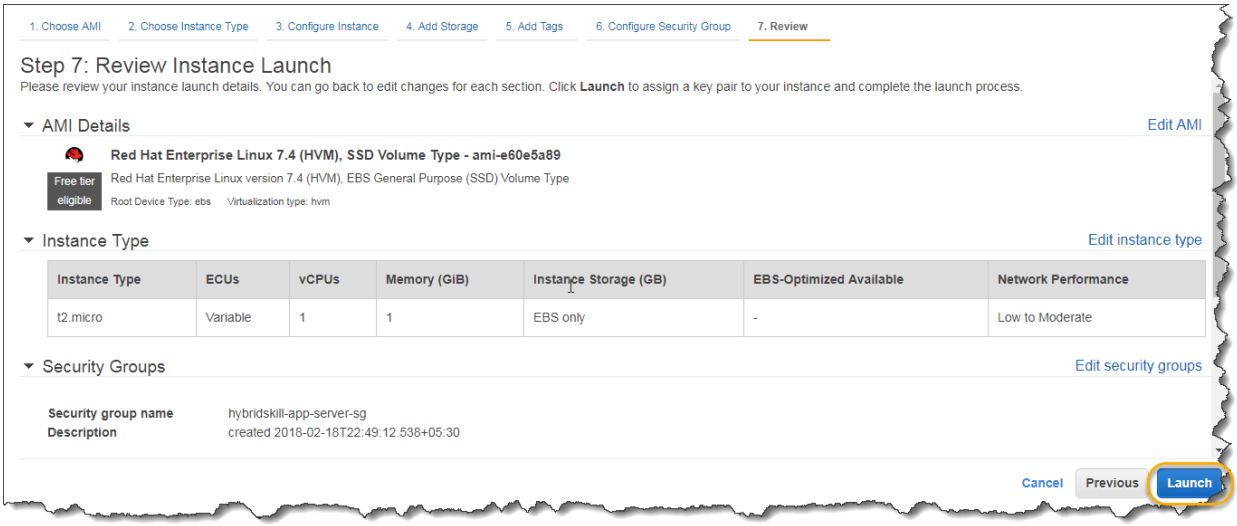
5. Click **Create** name tag and enter **database-server** as the name of your EC2 instance click **Next: Configure Security Group**



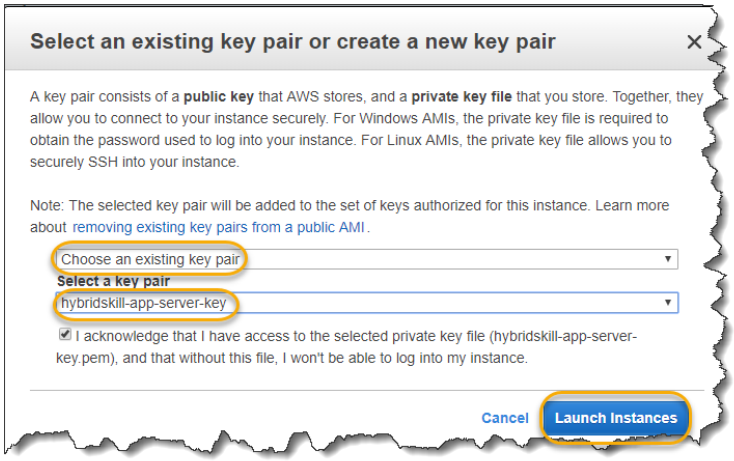
6. Enter **vpc-db-server-sg** as Security group name, enter a description. Click on add rule and from the dropdown add **ICMP-IPv4** and **SSH** as rules. For **both the rules,** select the source as **Custom** and enter **192.168.1.0/24** as the IP range to allow traffic from. *(You could alternatively set the **Source** as the security group of the **web-server).*** Finally click on **Review and Launch**.



7. Review your settings and click **Launch**



8. Select **Choose an existing key-pair** select the key you created earlier and Click **Launch Instances**



Task 5: Test out accessibility of subnets

1. Open a command prompt and ping the public IP address of the webserver. You should get a response.

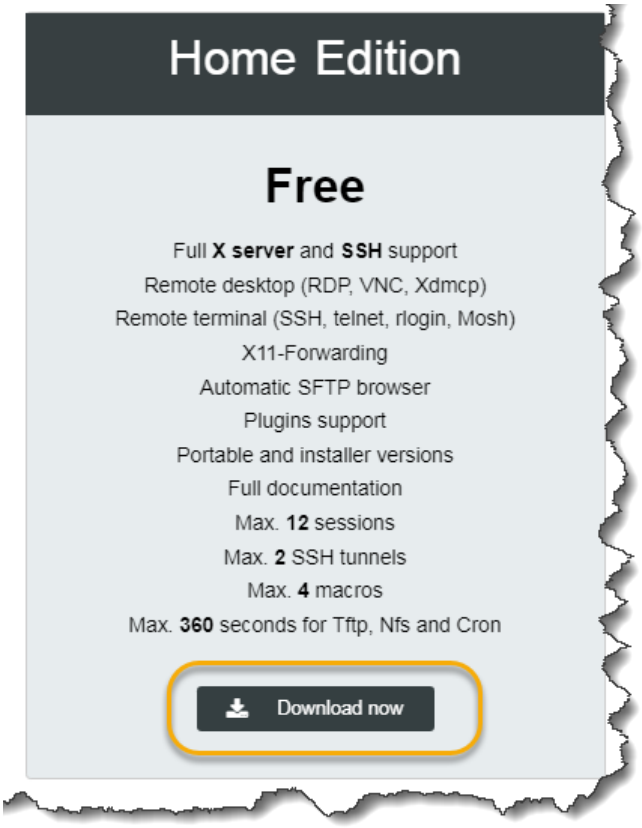
```
1.  ping 13.127.205.144
2.
3.  Pinging 13.127.205.144 with 32 bytes of data:
4.  Reply from 13.127.205.144: bytes=32 time=52ms TTL=53
5.  Reply from 13.127.205.144: bytes=32 time=53ms TTL=53
6.  Reply from 13.127.205.144: bytes=32 time=52ms TTL=53
7.  Reply from 13.127.205.144: bytes=32 time=53ms TTL=53
8.
9.  Ping statistics for 13.127.205.144:
10.     Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
11. Approximate round trip times in milli-seconds:
12.     Minimum = 52ms, Maximum = 53ms, Average = 52ms
```

2. Visit the public IP of the web server on browser. You should see the wordpress home page displayed
3. Log in to the webserver using SSH.

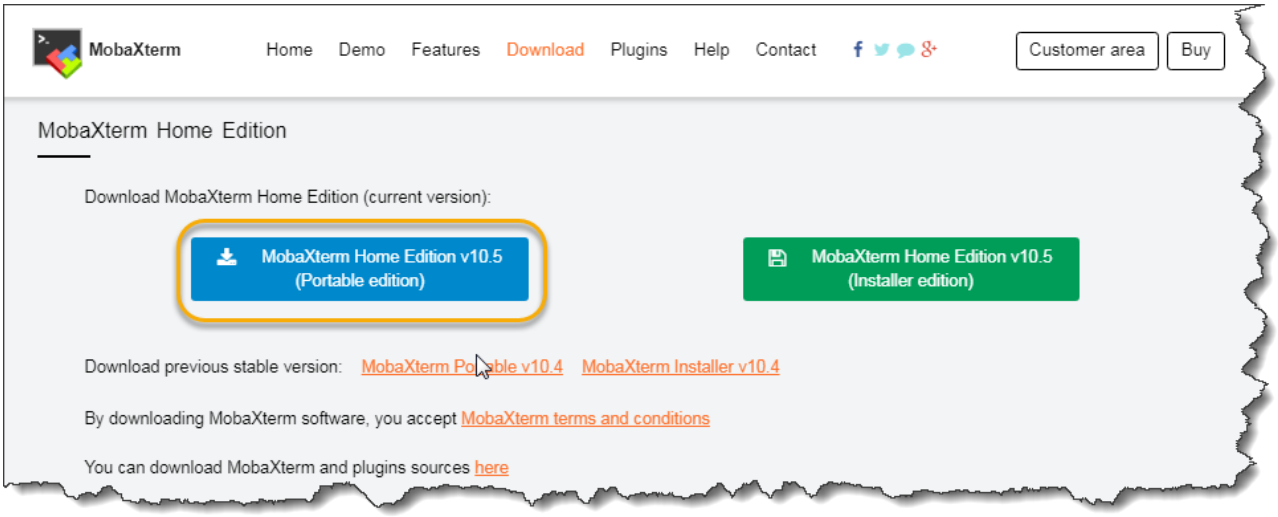
   . For Mac and Linux users run the following command to connect your instances. Make sure to use your public **IPv4** and your **Key pair** in the command.

```
1. chmod 500 hybridskill-app-server-key.pem
2. ssh -i hybridskill-app-server-key.pem ec2-user@13.126.115.88
```

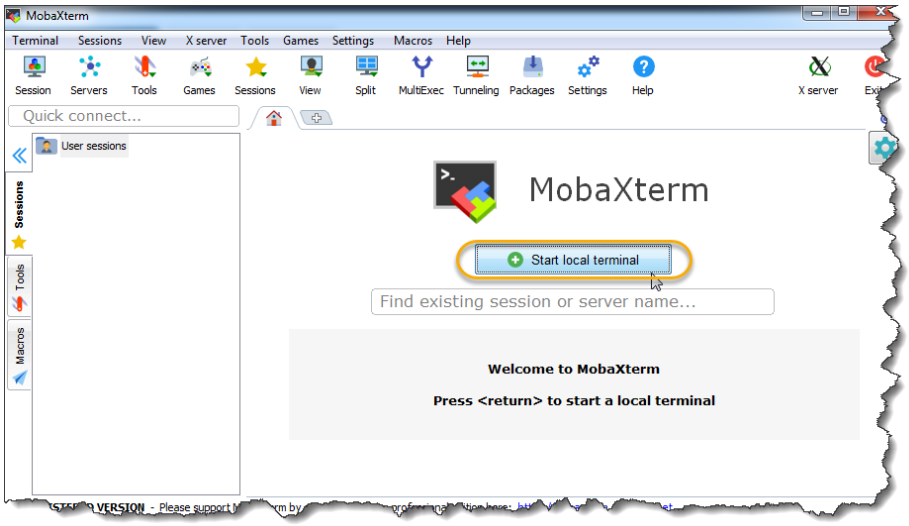
For Windows users we are going to use a SSH client call MobaXterm. Visit this link https://mobaxterm.mobatek.net/download.html Download the home edition.



Download and install the portable edition.



Click on **Start local terminal**



Copy your key pair to your desktop. Run the following command to connect your instance. Make sure to use your public **IPv4** and your **Key pair** in the command

```
1. cd Desktop
2. chmod 500 hybridskill-app-server-key.pem
3. ssh -i hybridskill-app-server-key.pem ec2-user@13.126.115.88
```

4. Ping google.com. You should get a response.

```
1. [ec2-user@ip-192-168-1-4 ~]$ ping google.com
2. PING google.com (209.85.202.101) 56(84) bytes of data.
3. 64 bytes from dg-inf101.1e100.net (209.85.202.101): icmp_seq=1 ttl=36 time=239 ms
4. 64 bytes from dg-inf101.1e100.net (209.85.202.101): icmp_seq=2 ttl=36 time=239 ms
5. 64 bytes from dg-inf101.1e100.net (209.85.202.101): icmp_seq=3 ttl=36 time=239 ms
6. 64 bytes from dg-inf101.1e100.net (209.85.202.101): icmp_seq=4 ttl=36 time=239 ms
7. 64 bytes from dg-inf101.1e100.net (209.85.202.101): icmp_seq=5 ttl=36 time=239 ms
8. ^C
9. --- google.com ping statistics ---
10. 6 packets transmitted, 5 received, 16% packet loss, time 5006ms
11. rtt min/avg/max/mdev = 239.837/239.856/239.882/0.438 ms
```

5. Next try and ping the database server. You should get response too.

```
[ec2-user@ip-192-168-1-4 ~]$ ping 192.168.2.4
PING 192.168.2.4 (192.168.2.4) 56(84) bytes of data.
64 bytes from 192.168.2.4: icmp_seq=1 ttl=64 time=0.781 ms
64 bytes from 192.168.2.4: icmp_seq=2 ttl=64 time=0.967 ms
64 bytes from 192.168.2.4: icmp_seq=3 ttl=64 time=0.893 ms
64 bytes from 192.168.2.4: icmp_seq=4 ttl=64 time=0.908 ms
^C
--- 192.168.2.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.781/0.887/0.967/0.070 ms
```

6. Next log into the database server from your webserver. You will the need the key of the database server placed on the webserver for this to happen.
   a. For Windows users, **MobaXterm** has a provision for this. Open a new terminal and run following command use IP of webserver.

```
scp -i hybridskill-key.pem hybridskill-key.pem ec2-user@13.126.115.88:.
```

   b. For MAC users you need to user the built in **SCP** command. Replace the IP with the Public IP of your webserver, and hybridskill-key.pem with the path to your key on your system. You need to

```
scp -i hybridskill-key.pem hybridskill-key.pem ec2-user@13.127.205.144:.
Warning: Permanently added '13.127.205.144' (RSA) to the list of known hosts.
hybridskill--key.pem
```

6. After the copy is done run the following commands to log into your database instance.

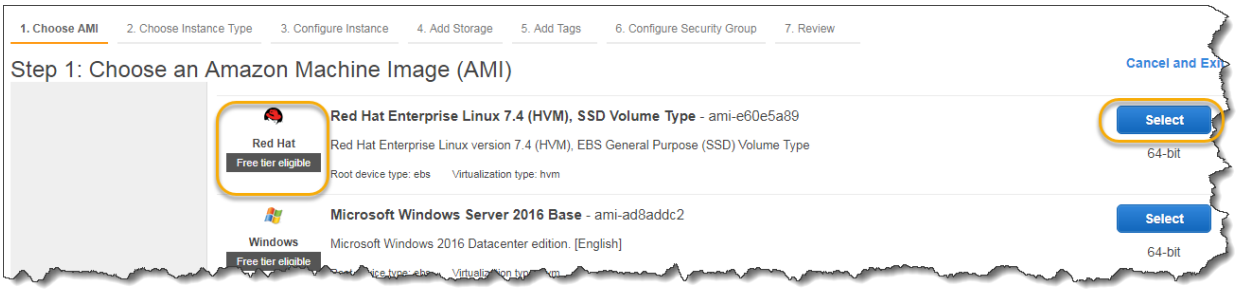
```
[ec2-user@ip-192-168-1-4 ~]$ sudo chmod 500 hybridskill-app-server-key.pem
[ec2-user@ip-192-168-1-4 ~]$ ssh -i hybridskill-app-server-key.pem ec2-user@192.168.2.4
The authenticity of host '192.168.2.4 (192.168.2.4)' can't be established.
ECDSA key fingerprint is SHA256:xOjxcpTh0X01v0IBG30BCLnQbCy2O1aFUQppliNEwg4.
ECDSA key fingerprint is MD5:10:a3:d7:37:af:de:56:41:40:ba:9d:7d:20:11:b3:d6.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.4' (ECDSA) to the list of known hosts.
[ec2-user@ip-192-168-2-4 ~]$
```

7. Once you are logged into the database serve try and ping google.com. You should not be able to get a response.

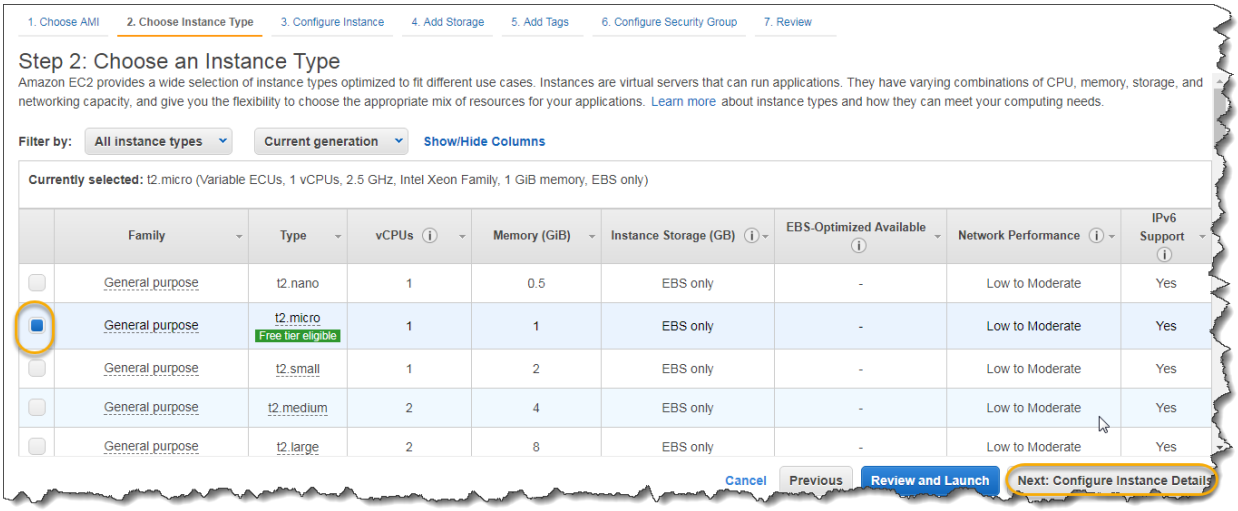
```
[ec2-user@ip-192-168-2-4 ~]$ ping google.com
PING google.com (172.217.27.206) 56(84) bytes of data.
^C
--- google.com ping statistics ---
40 packets transmitted, 0 received, 100% packet loss, time 38999ms
```

Task 6: Launch a NAT instance.

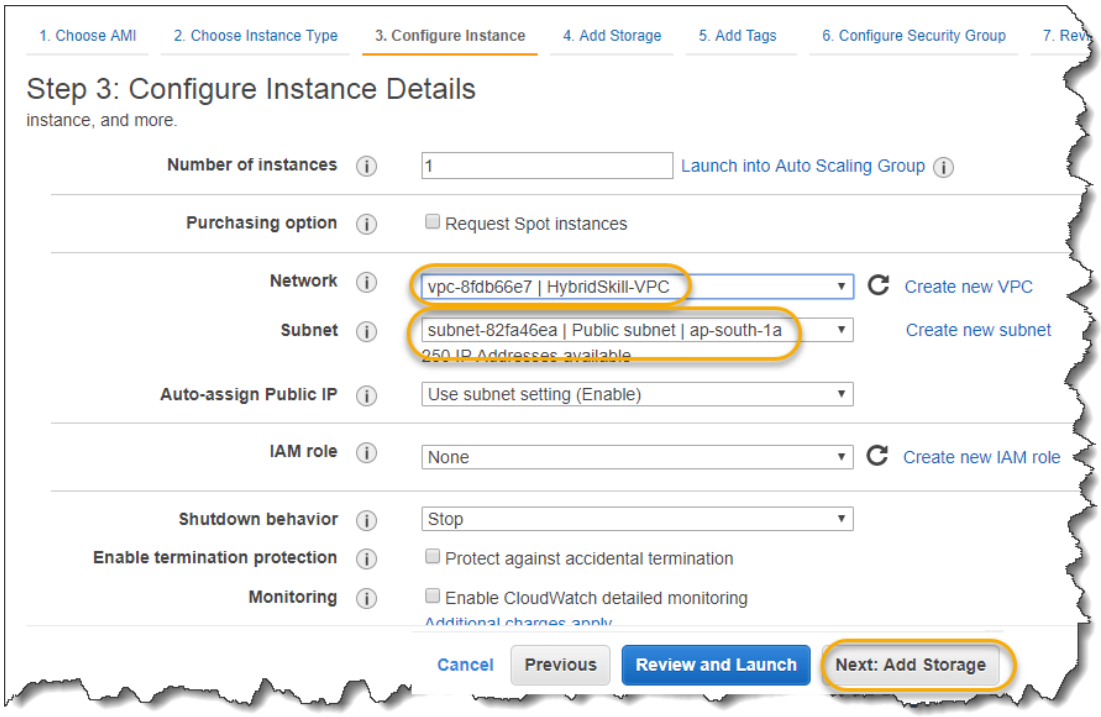
1. Choose **Red Hat Enterprise Linux 7.4** as your **AMI** by clicking the **Select** button.



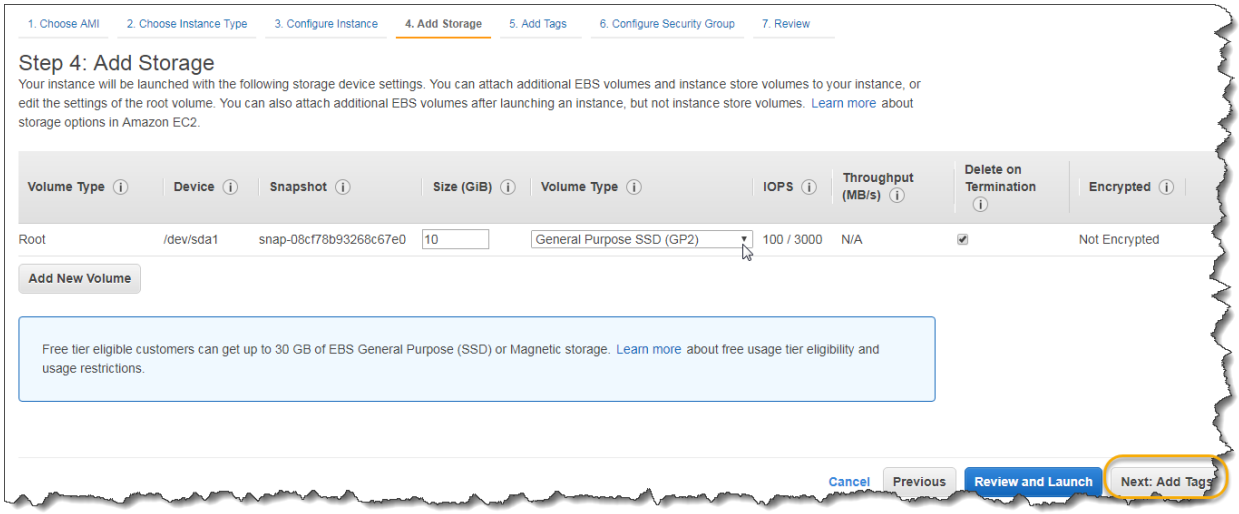
2. Select **t2.micro** as your Instance Type and click **Next: Configure Instance Details**



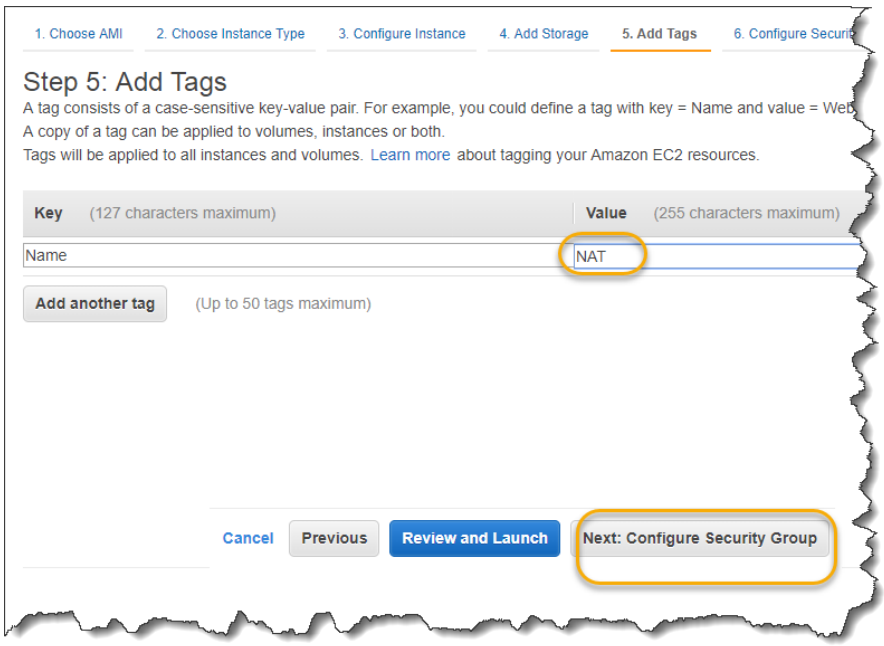
3. For Network select your **HybridSkill-VPC,** select **Public Subnet** you created earlier from the dropdown. Leave all other options as default and click **Next: Add Storage**



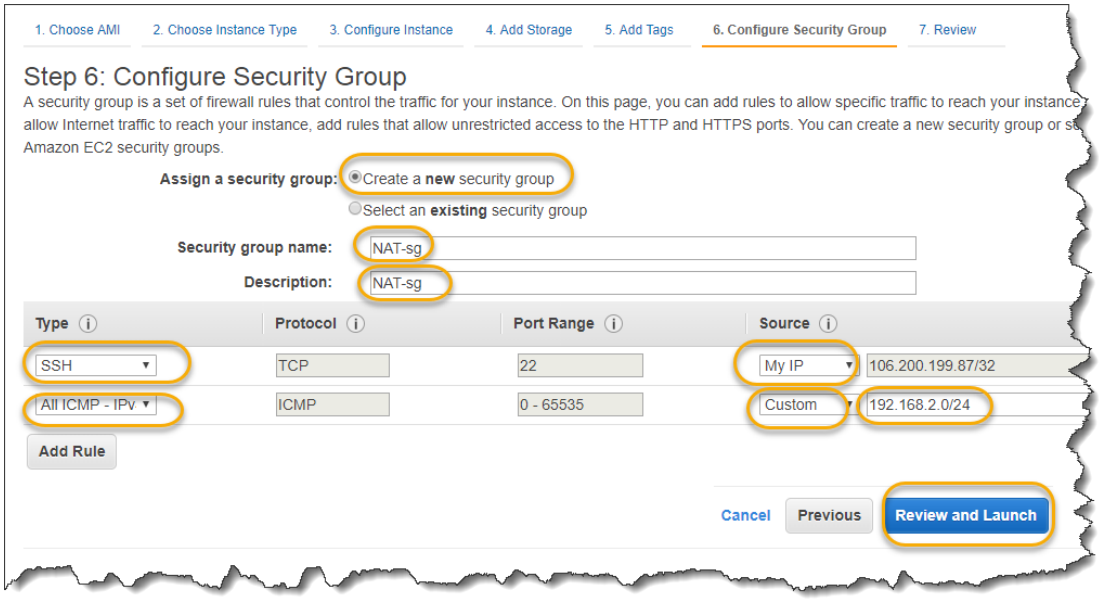
4. Leave all the options as default and click on **Next: Add Tags**



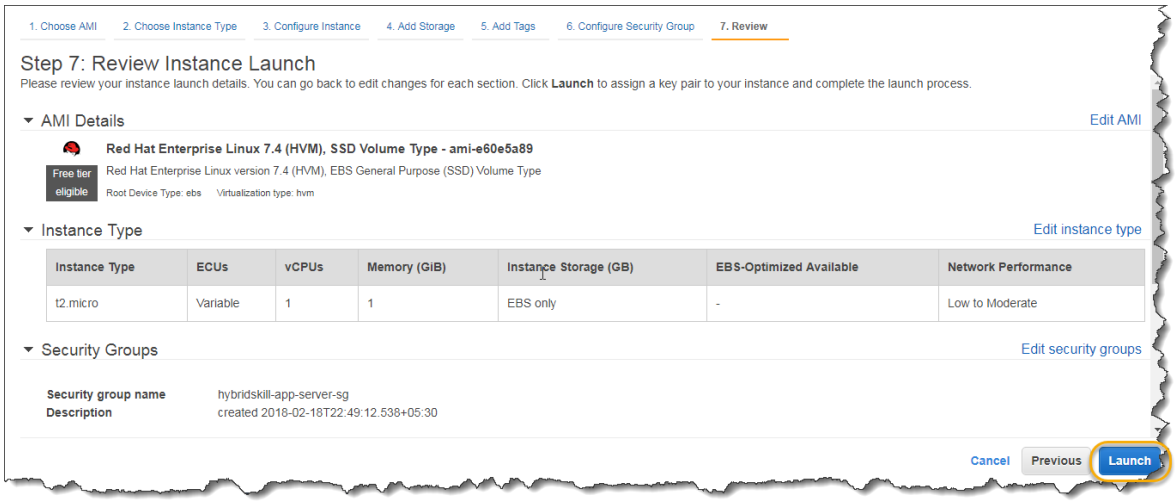
5. Click **Create** name tag and enter **NAT** as the name of your EC2 instance click **Next: Configure Security Group**



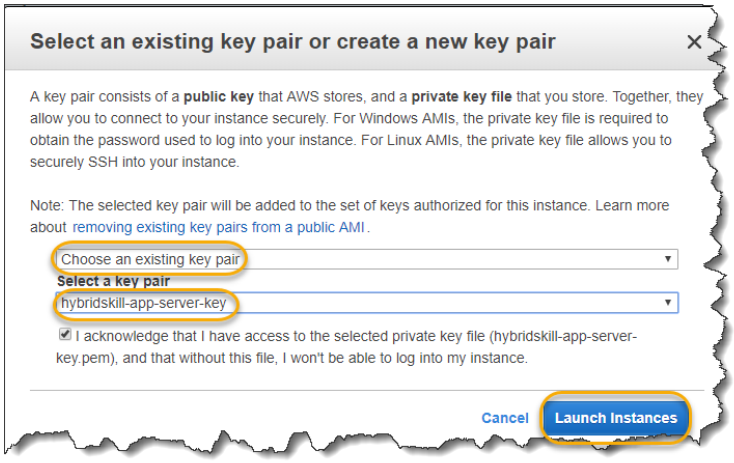
6. Enter **NAT-sg** as Security group name, enter a description. Click on add rule and from the dropdown add **ICMP-IPv4** and **SSH** as rules. For **SSH** select the source as **My IP** from the dropdown. For **ICMP-IPv4** select the source as **Custom** and enter **192.168.2.0/24(IP range of Private Subnet)** as the IP range to allow traffic from. *(You could alternatively set the **Source** as the security group of the **database-server).*** Finally click on **Review and Launch**.



7. Review your settings and click **Launch**



8. Select **Choose an existing key-pair** select the key you created earlier and Click **Launch Instances**

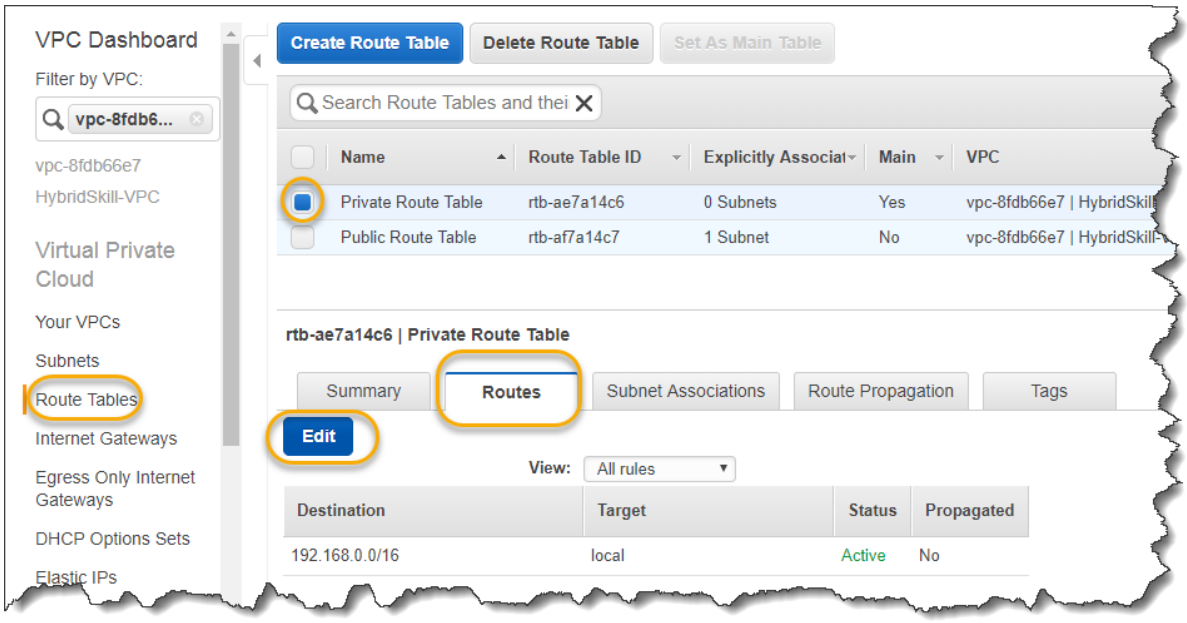


9. Log into the NAT instance and run the following commands to setup NAT

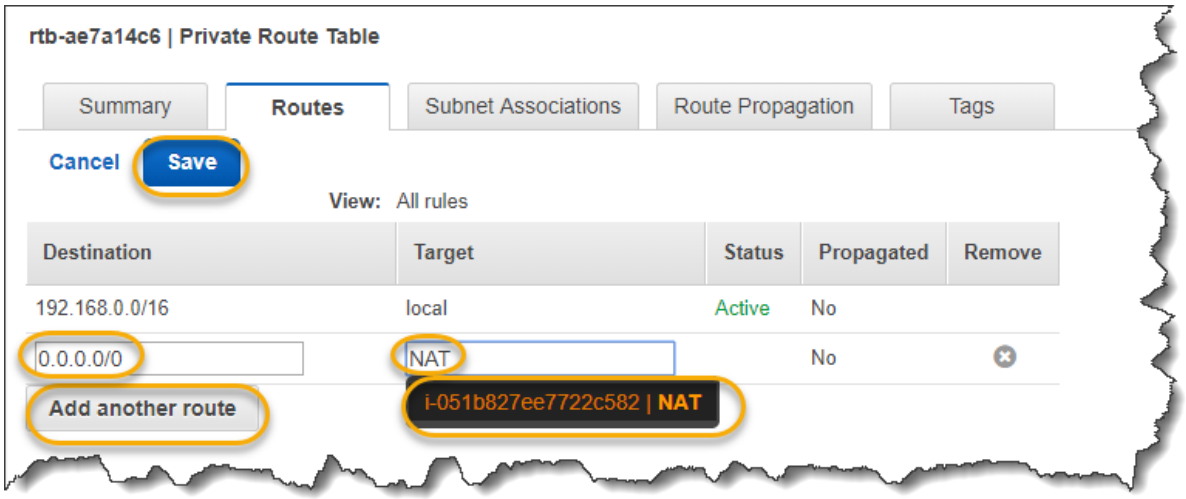
```
1. sudo sysctl -w net.ipv4.ip_forward=1
2. sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Task 7: Modify the private route table to Route traffic to NAT

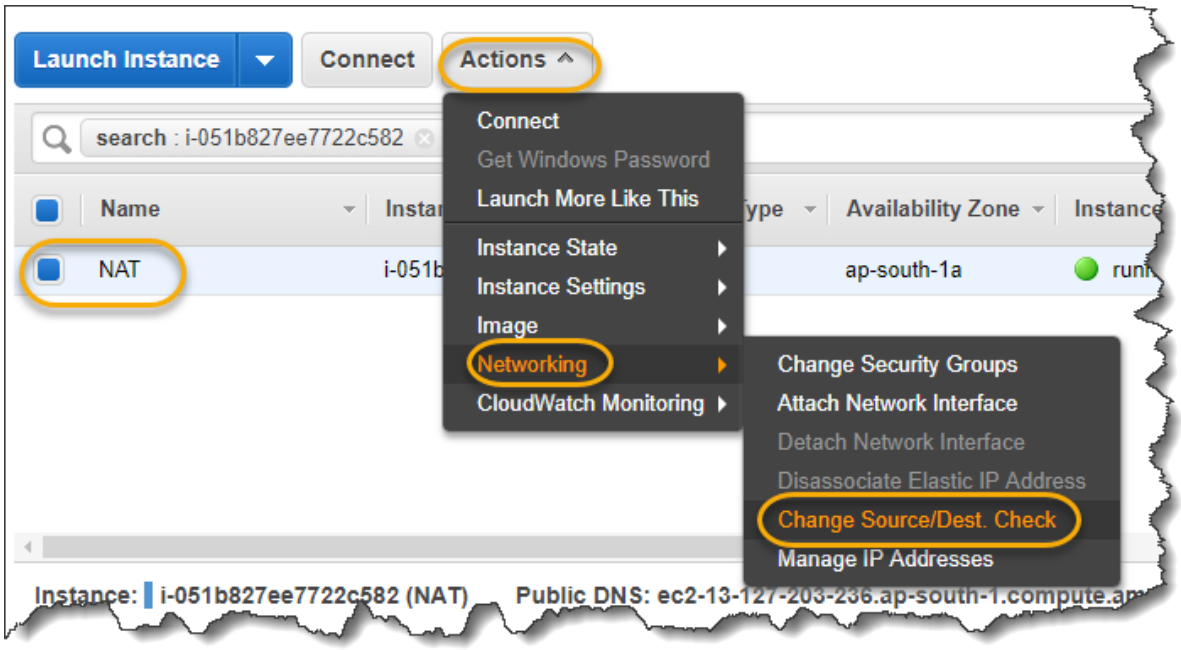
1. Go to **VPC** -> **Route Tables**, select **Private Route Table**. Next on the bottom of the screen, Select the **Routes Tab** and click **Edit.**



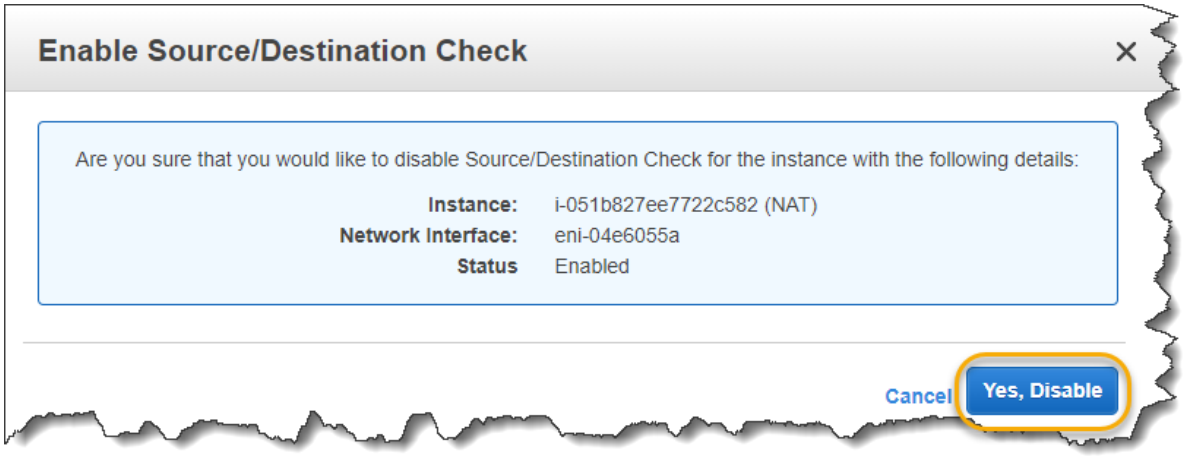
2. **Click Add another Route. Enter 0.0.0.0/0 as the Destination.** For the **Target** search and select your **NAT instance** you created earlier. Finally Click **Save.**



3. There is one final setting to be done. select your **NAT instance** click **Actions**, scroll down to **Networking** and click on **Change Source/Dest Check.**



4. Click **Yes, Disable**

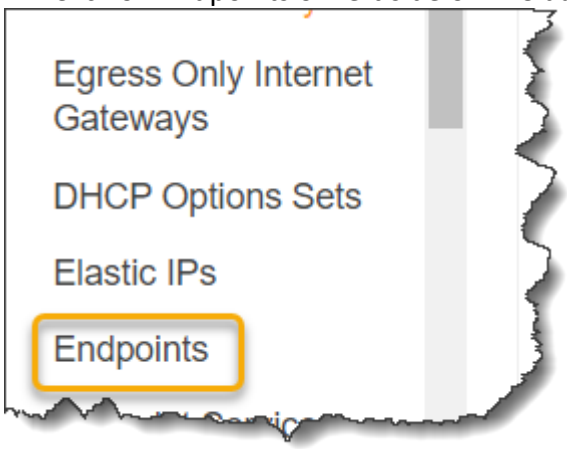


5. Now try and ping google.com from your database server. You should get successful ping.

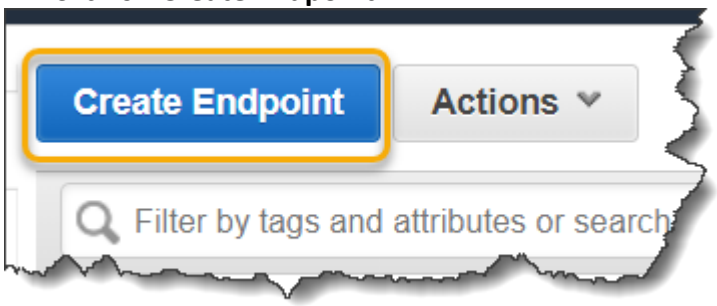
```
ping google.com
PING google.com (172.217.27.206) 56(84) bytes of data.
64 bytes from bom07s15-in-f14.1e100.net (172.217.27.206): icmp_seq=6 ttl=52 time=2.69 ms
64 bytes from bom07s15-in-f14.1e100.net (172.217.27.206): icmp_seq=7 ttl=52 time=2.63 ms
64 bytes from bom07s15-in-f14.1e100.net (172.217.27.206): icmp_seq=8 ttl=52 time=2.70 ms
64 bytes from bom07s15-in-f14.1e100.net (172.217.27.206): icmp_seq=9 ttl=52 time=2.73 ms
^C
--- google.com ping statistics ---
9 packets transmitted, 4 received, 55% packet loss, time 8005ms
rtt min/avg/max/mdev = 2.636/2.692/2.735/0.036 ms
```

Task 8: S3 Endpoint

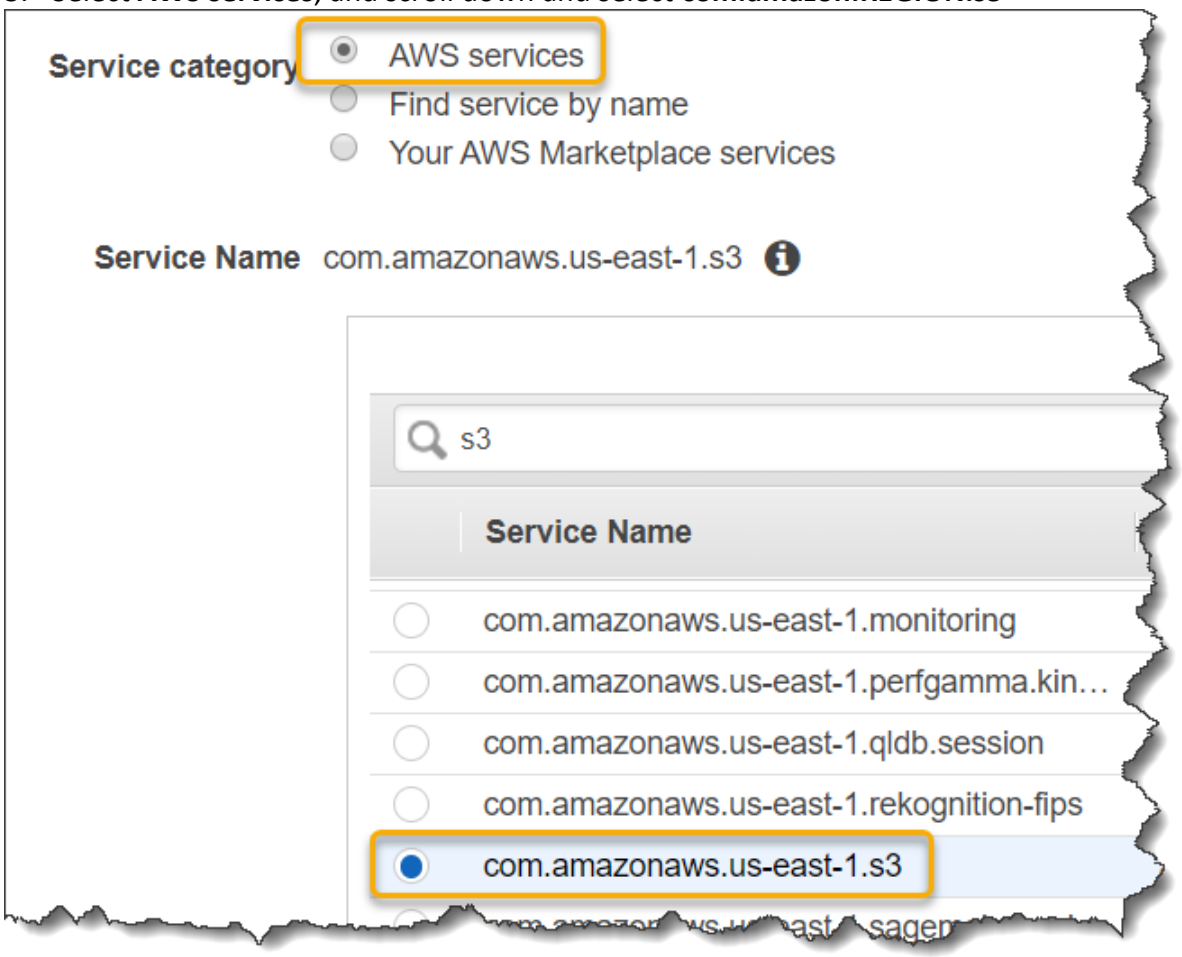
1. Click on Endpoints on left side of VPC dashboard



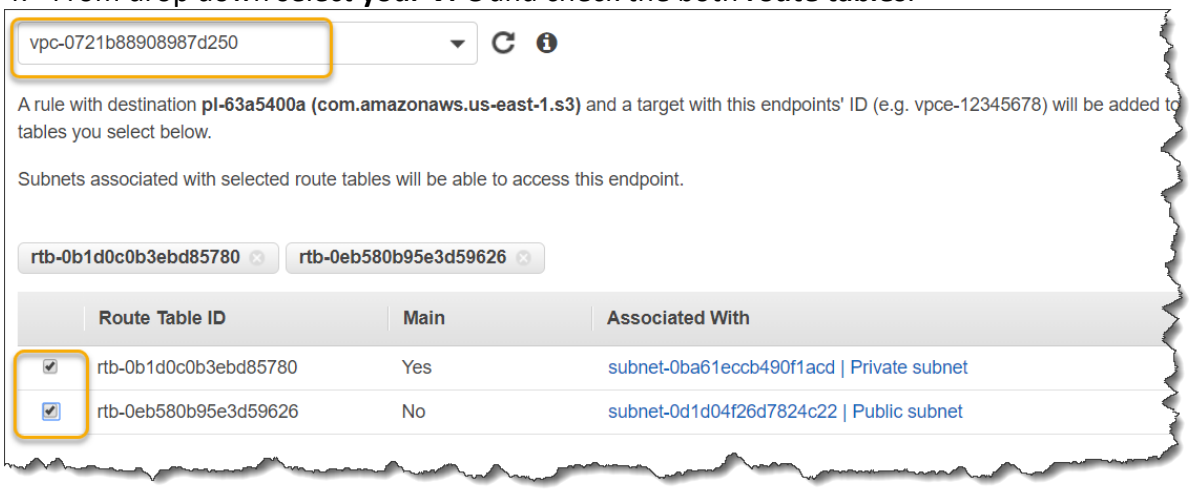
2. Click on **Create Endpoint**



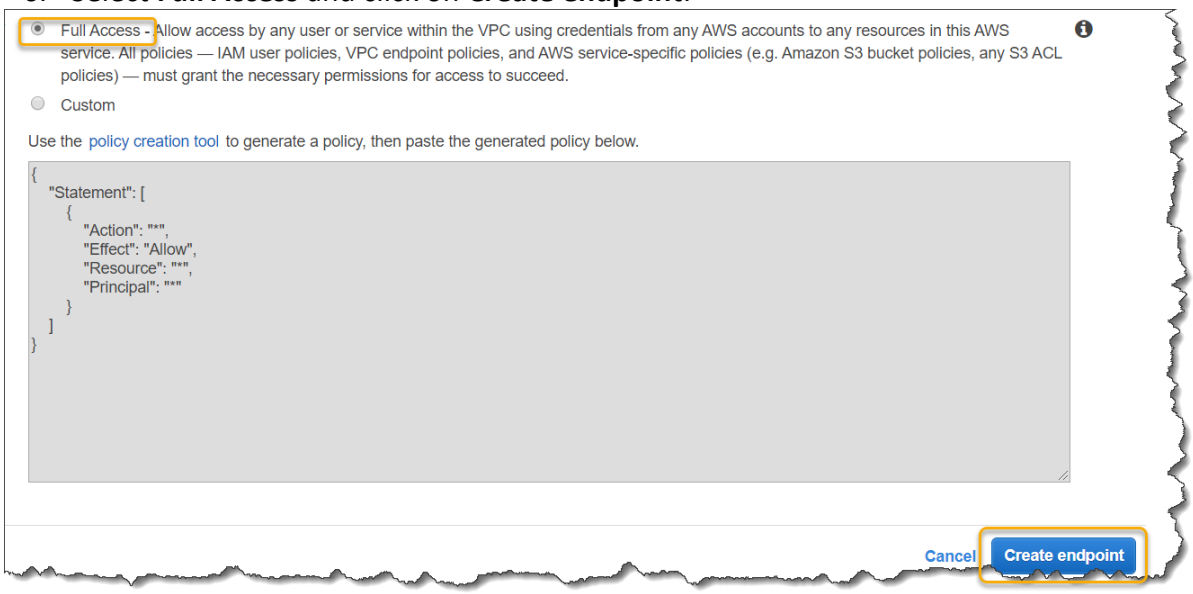
3. Select **AWS services**, and scroll down and select **com.amazon.REGION.s3**



4. From drop down select **your VPC** and check the both **route tables**.



6. Select **Full Access** and click on **Create endpoint**.



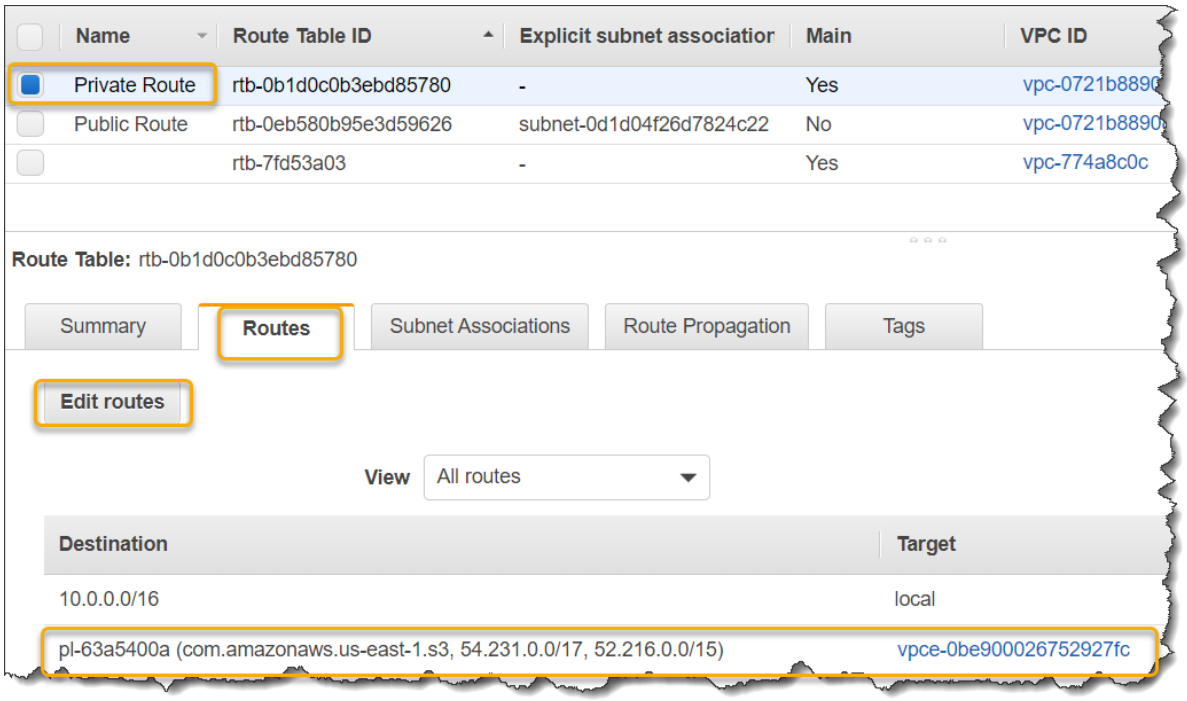
7. Now install AWS-cli on this machine by running following commands

```
curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o "awscli-bundle.zip"
sudo yum install zip unzip
unzip awscli-bundle.zip
cd awscli-bundle
sudo ./install -i /usr/local/aws -b /usr/local/bin/aws
aws --version
```

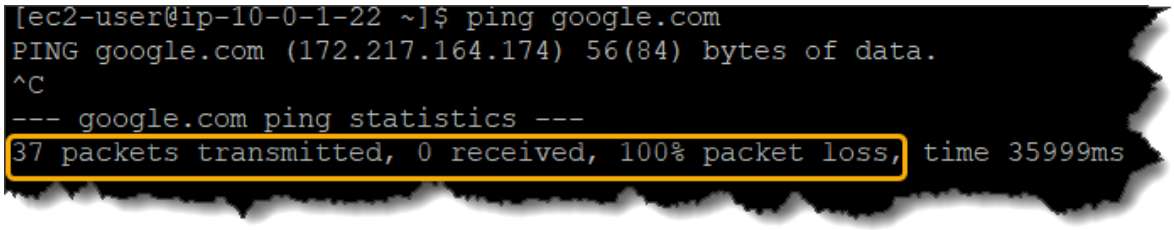
8. Now configure aws and provide **your access key** and **secret key** and **your region**.

```
aws configure
```

9. Now go to your **Private Route table** click on **Routes** and then on **Edit routes** and remove **Nat instance route rule**, you can also notice that **s3 endpoint route** is also added to your route table now. It will look like below



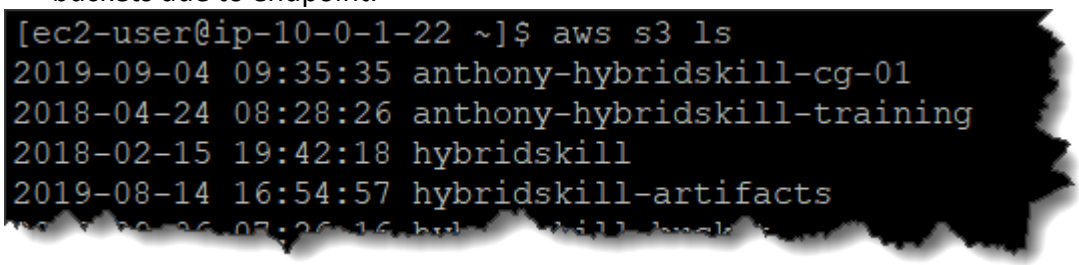
10. Now login to your DB instance in private subnet and ping google.com as expected it will fail so it is clear that there is no internet connection.



11. Now try to list s3 buckets

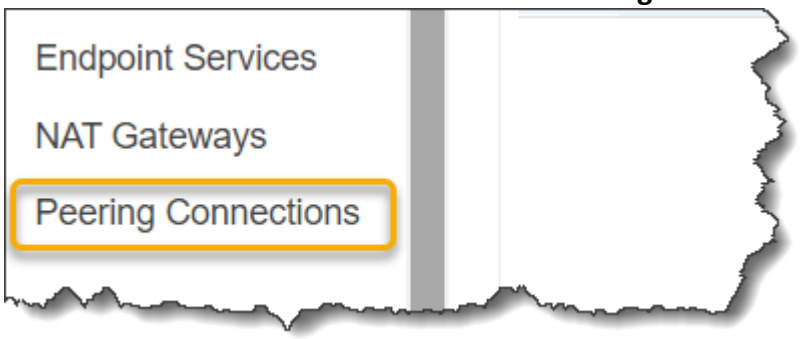
```
aws s3 ls
```

12. As you can see even though there is no intenet route even then we are able to list s3 buckets due to endpoint.

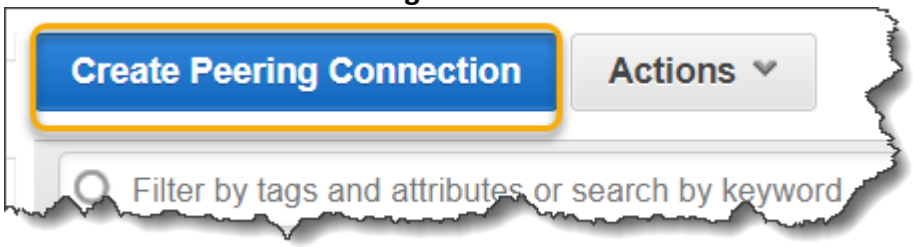


Task 9: VPC Peering

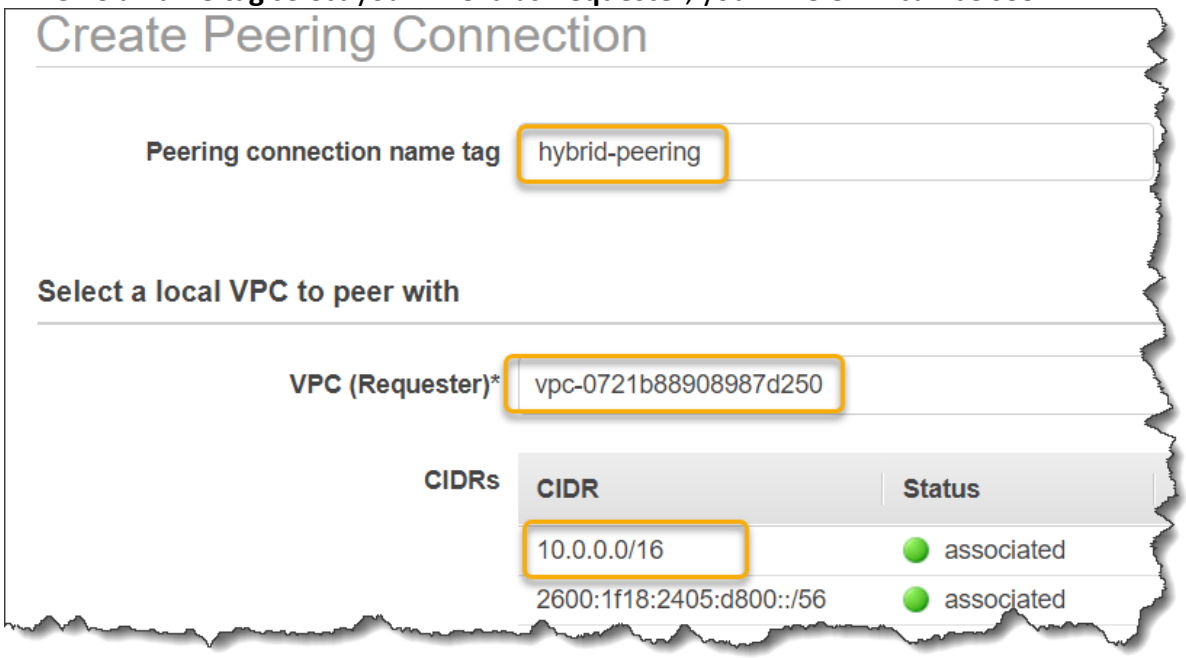
1. Create **another VPC** if you do not have two VPCs or ask your friend for his/her **VPC id**. But make sure that range of both **VPC CIDRs** do not coincide for example if one VPC has **10.0.0.0/16** CIDR other should have different CIDR like **192.168.0.0/16** etc.
2. On left side of VPC dashboard click on **Peering Connections.**



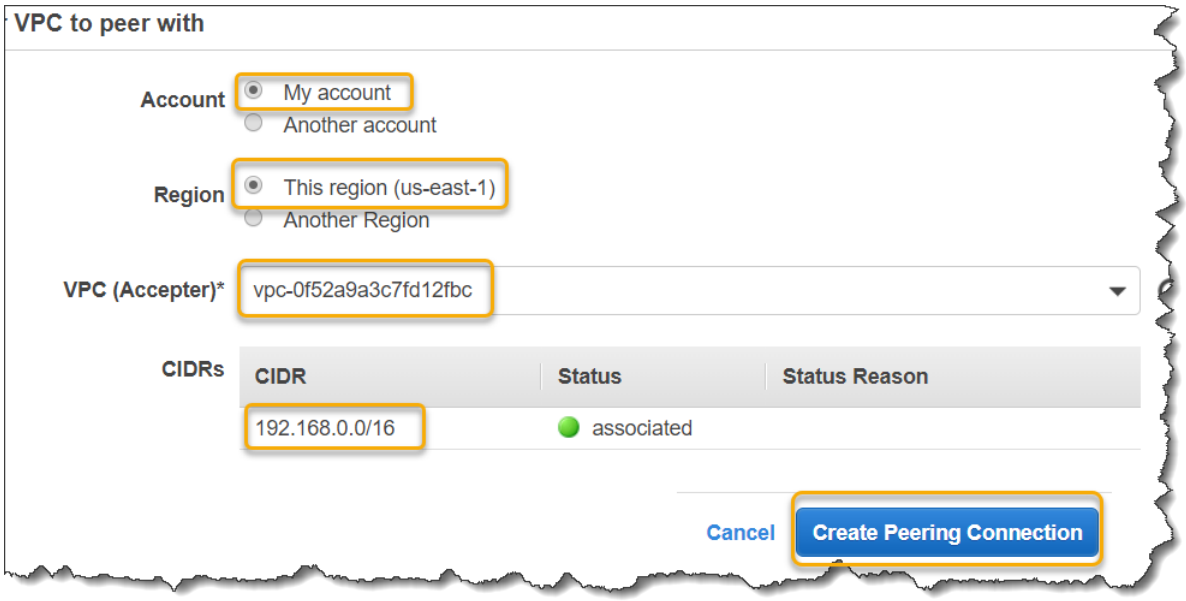
3. Now click on **Create Peering Connection.**



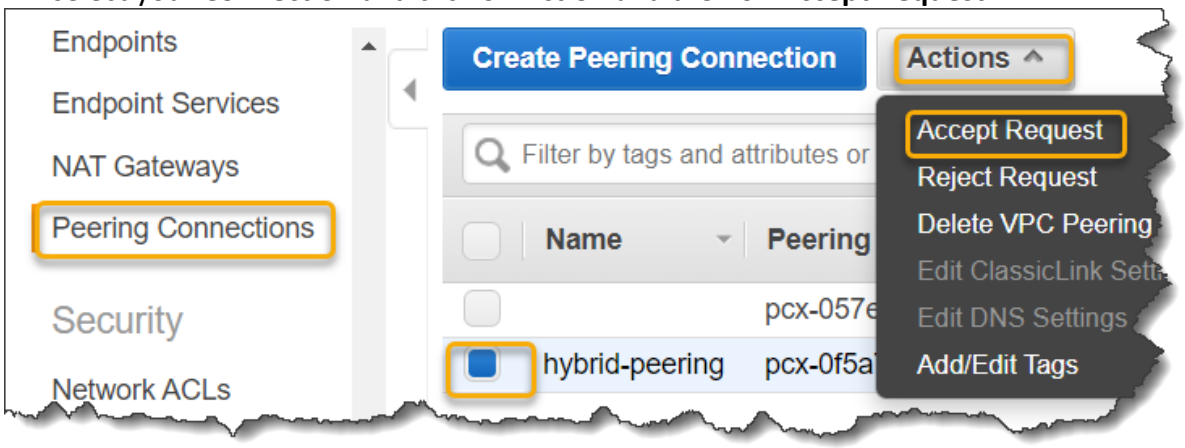
4. Give a **name tag** select your **VPC id** as **Requester**, your **VPC CIDR** can be seen.



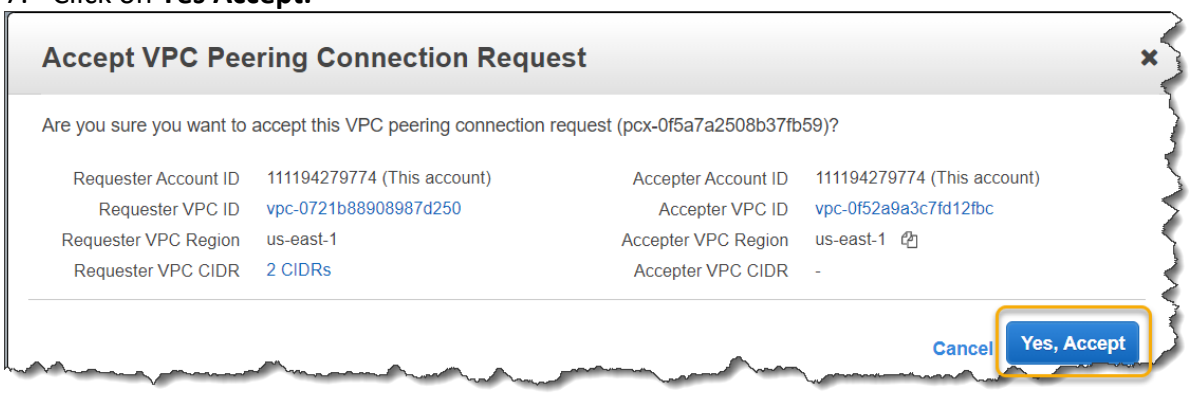
5. Choose **Account** in this case My account. Choose **Region** choose **Accepter VPC** and you can see its **CIDR is different** from Requester CIDR now click on **Create Peering Connection**



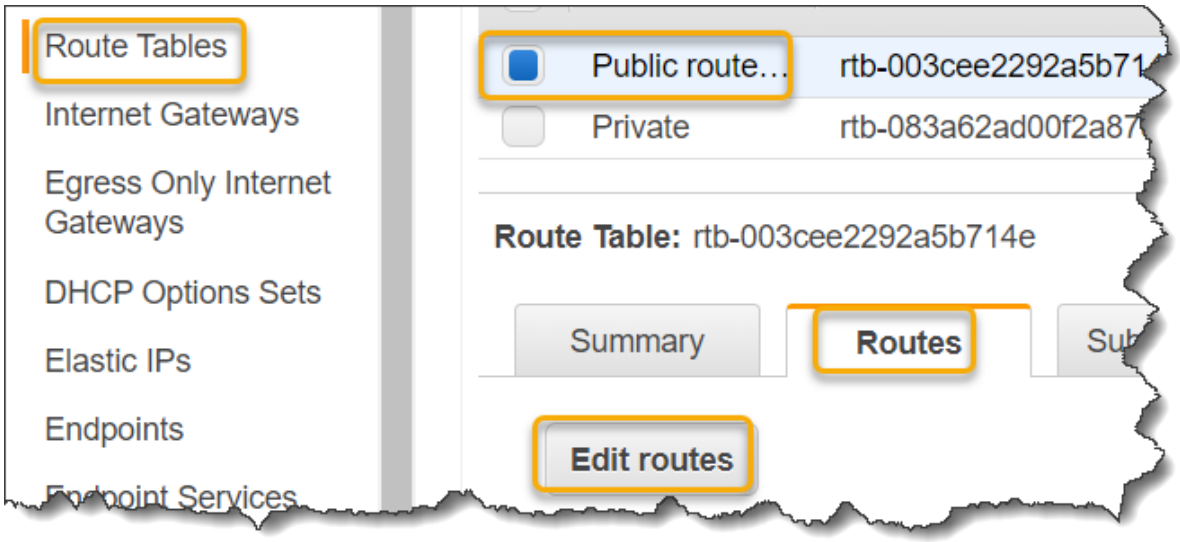
6. If accepter VPC was in diff Region/Account visit there and click on **Peering connections** select your **Connection** and click on **Action** and then on **Accept Request**.



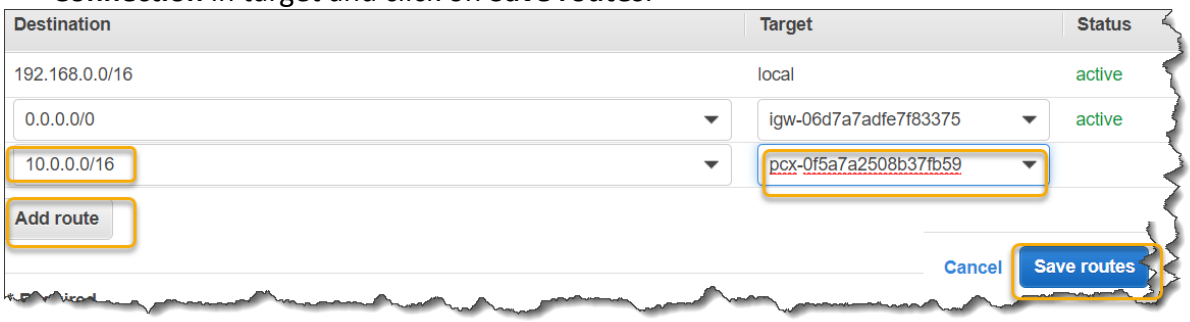
7. Click on **Yes Accept.**



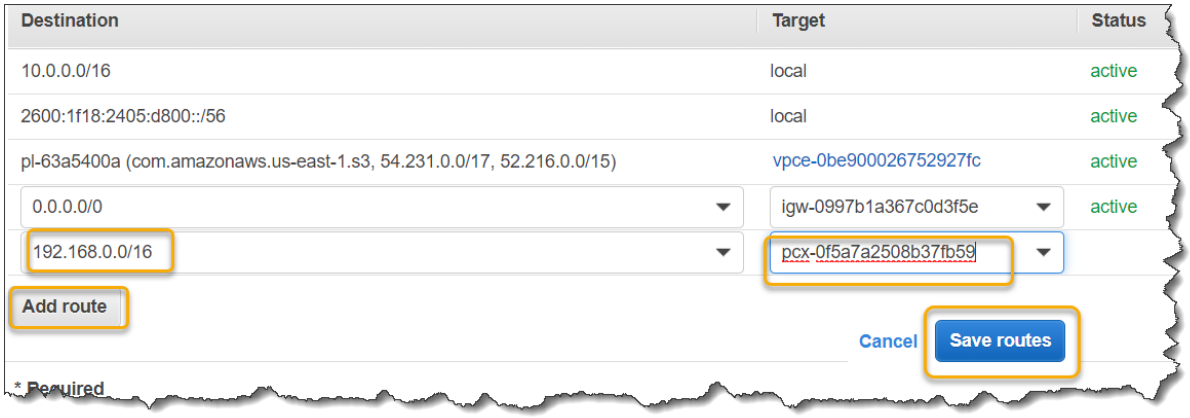
8. Now as final step we have to add routes to the Route tables for both VPCs click on **Route Tables** on left hand side of VPC dashboard, click one **Public route table**, then click on **Routes** and then on **Edit routes**



9. Click on **Add route** and provide **CIDR of second VPC** in Destination and select **Peering Connection** in target and click on **Save routes**.



10. Repeat for all Route Tables.
11. Now do the same for other VPC Route tables also. In destination provide CIDR of first VPC.



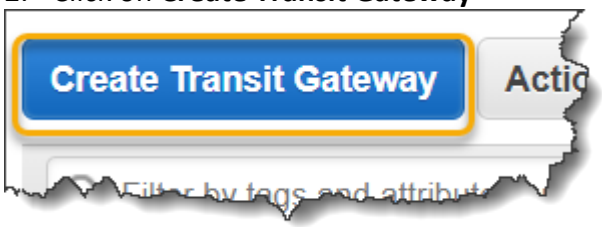
13. Now try to ping one of instance's **private IP** from an instance in other VPC ping should be successful.

Task 10: Setup Transit Gateway

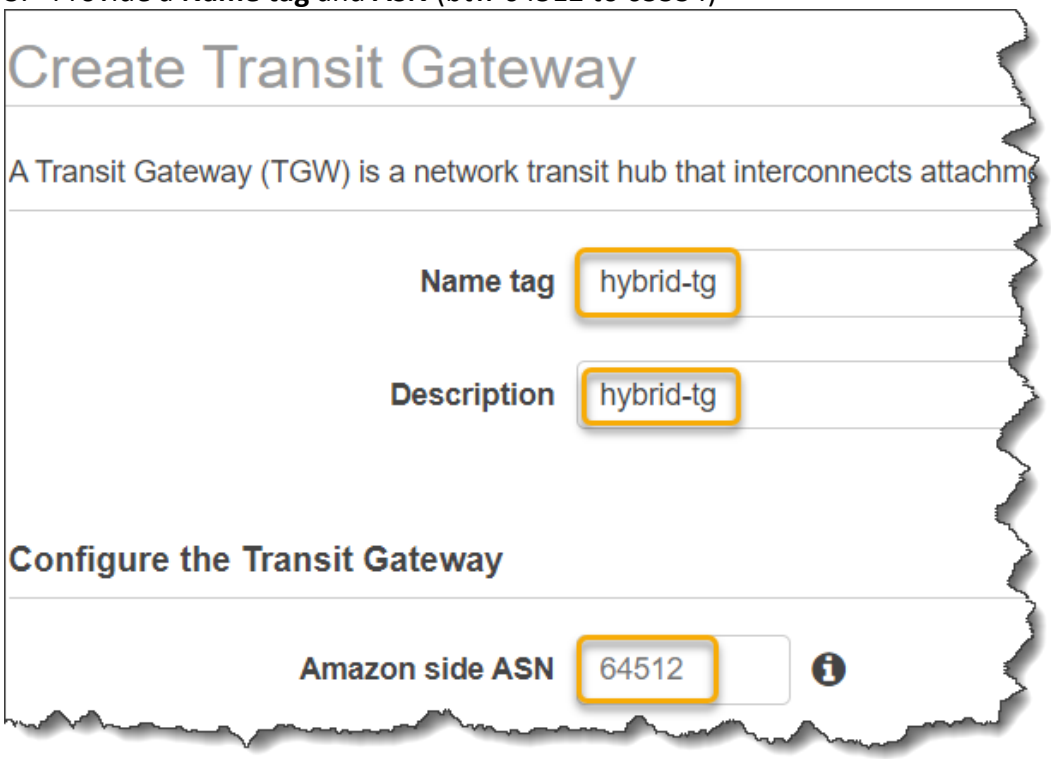
To demonstrate a simple example of using a transit gateway, create three VPCs one Dev, one prod and one staging in the same Region. The VPCs cannot have overlapping CIDRs. Launch one EC2 instance in each VPC. Through transit gateway we will make such arrangement that Dev and prod can talk to Staging VPC and vice-versa, but prod and dev will not talk to each another.

1. Go to your VPC dashboard and from left hand side scroll down and select **Transit Gateways.**

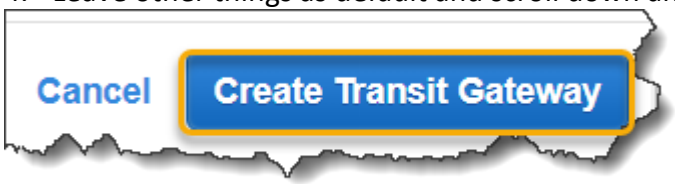
2. Click on **Create Transit Gateway**



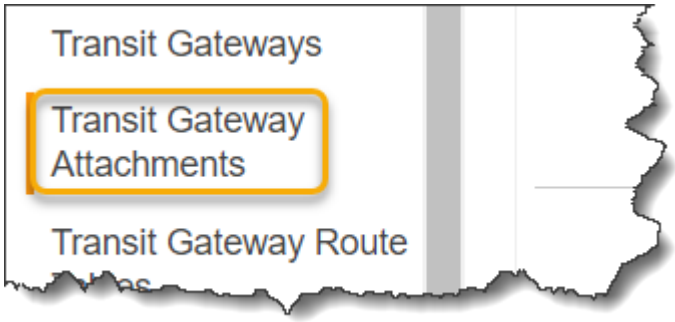
3. Provide a **Name tag** and **ASN** (btw 64512 to 65534)



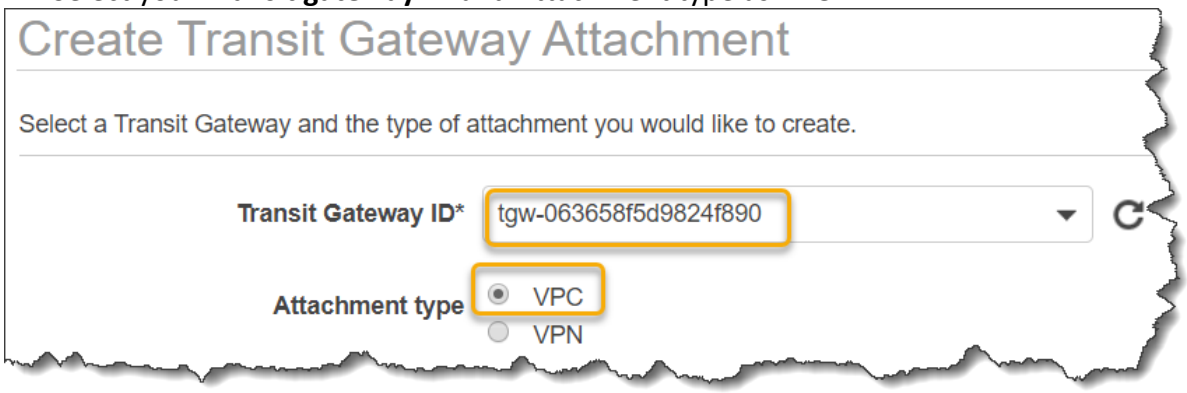
4. Leave other things as default and scroll down and click Create Transit Gateway



5. After you see the message **Create Transit Gateway request succeeded**, choose **Close**. The initial state of the transit gateway is **pending**. Let it become **available**.
6. Now from left hand side scroll down and select **Transit Gateway Attachments**

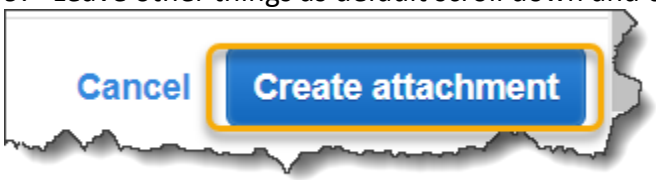


7. Select your **Transit gateway ID** and Attachment type as **VPC**.



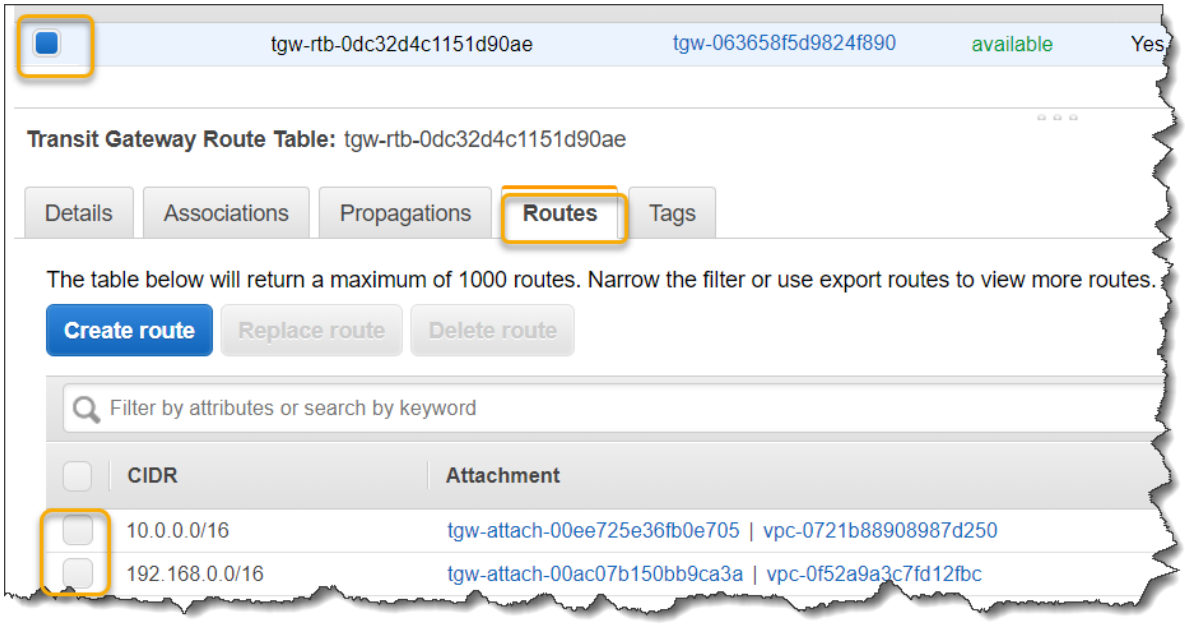
8. Give a **name tag** select your **VPC ID** and select **Availability Zones** in which you have subnets

9. Leave other things as default scroll down and click on **Create attachment**.

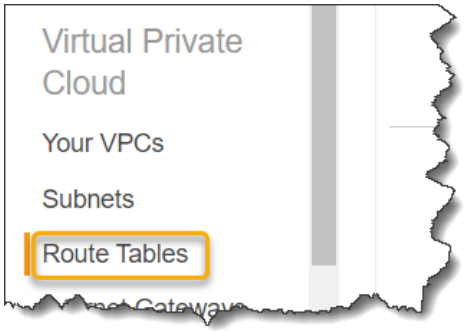


10. Make another attachment for the other **VPC** also.
11. Now click on transit **Gateway Route Tables**.

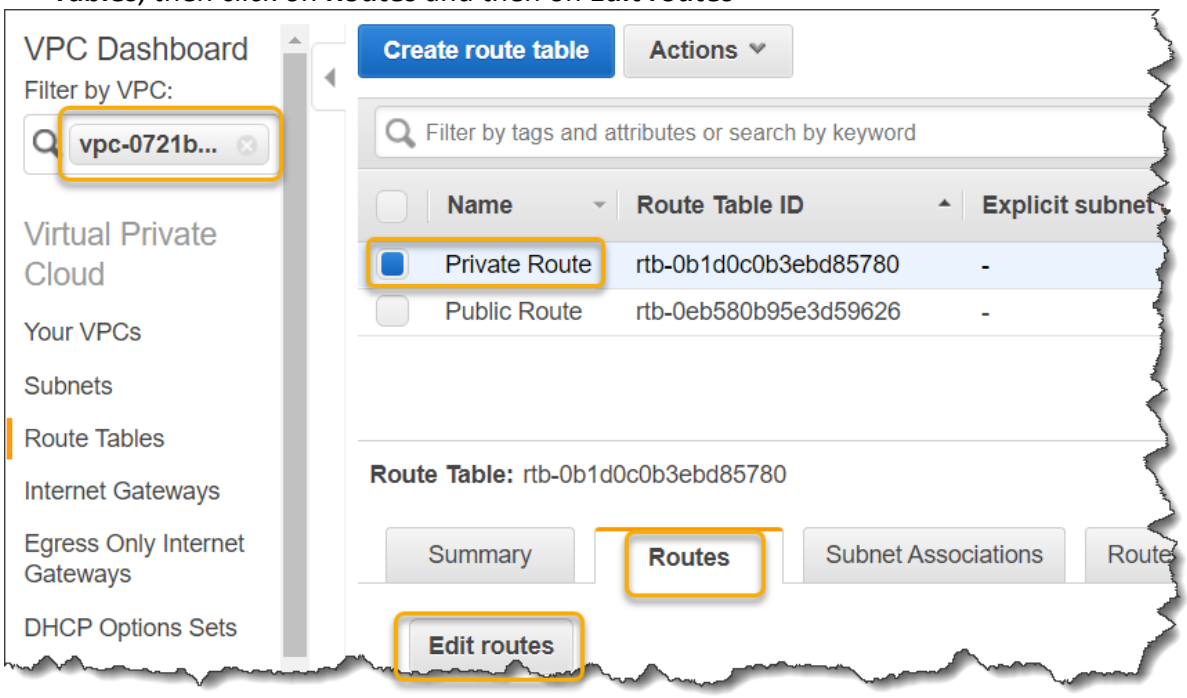
12. Now select the **Route table** click on **Routes** we can see both **VPC CIDR** are added to the Routes.



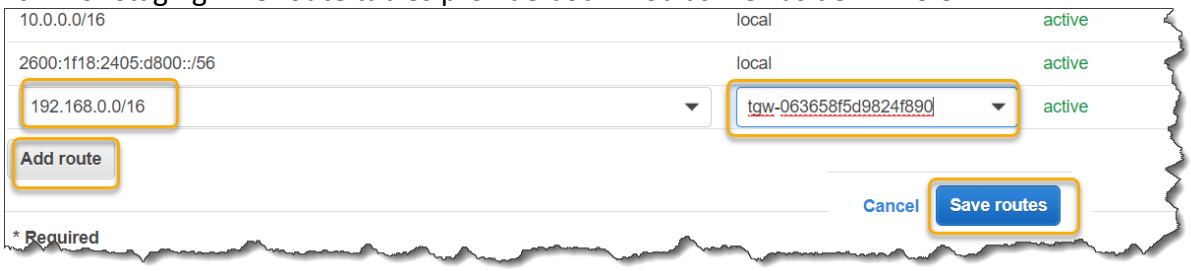
14. Now we have to edit routes to Route Tables of VPCs, click **Route Tables** on left side of VPC Dashboard



15. So that you do not get confused from filter select your VPC and click on one of the **Route Tables**, then click on **Routes** and then on **Edit routes**



16. Click on **Add route** provide other **VPC CIDR** select target as **Transit gateway** and click on **Save routes,** Repeat for all route tables in both VPCs.
    a. For prod- VPC route tables provide staging VPC CIDR
    b. For dev-VPC route tables provide staging VPC CIDR
    c. For staging-VPC route tables provide both Prod as well as dev. VPC CIDR



17. Now try to ping **Private IP** of one of instances in **Staging** VPC from instances in prod and dev VPC it will work and vice-versa.
18. But when u try ping **Dev VPC** instance from **Prod VPC** instance and vice-versa it will **fail**.

Task 11: VPC Private Links

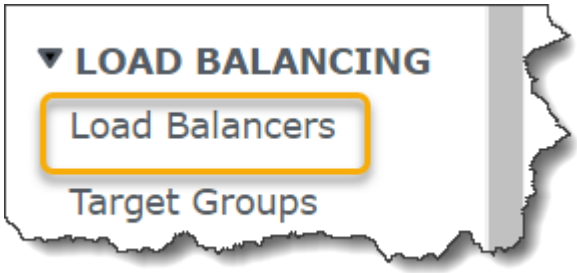
1. Create two VPCs with a public subnet in a same availability zone name one as **Producer** and one as **Receiver**
2. In **Producer** VPC create webserver in Public Subnet, In **Security Group** open TCP port 80 to receive inbound traffic from any IP and SSH port 22 to your IP.

3. Now login into the VM and install HTTP webserver by running following commands and provide some **Text** in **index.html** like 'this is webserver'

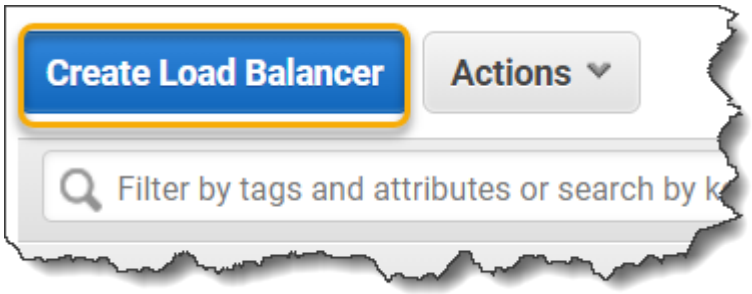
```
sudo yum install -y httpd
sudo service httpd start
cd /var/www/html/
sudo vi index.html
```

4. Now open Public IP of VM in a new browser tab you should be able to see your Text.

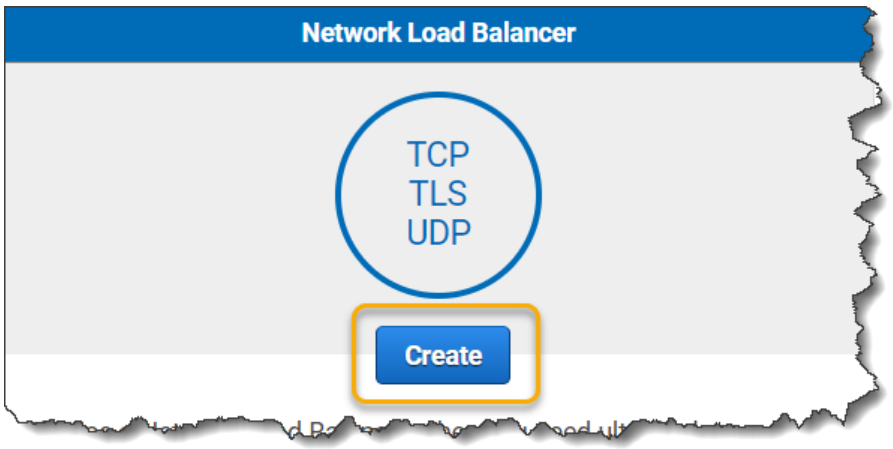
5. On left hand side of EC2 console click on **Load Balancers** under Load Balancing



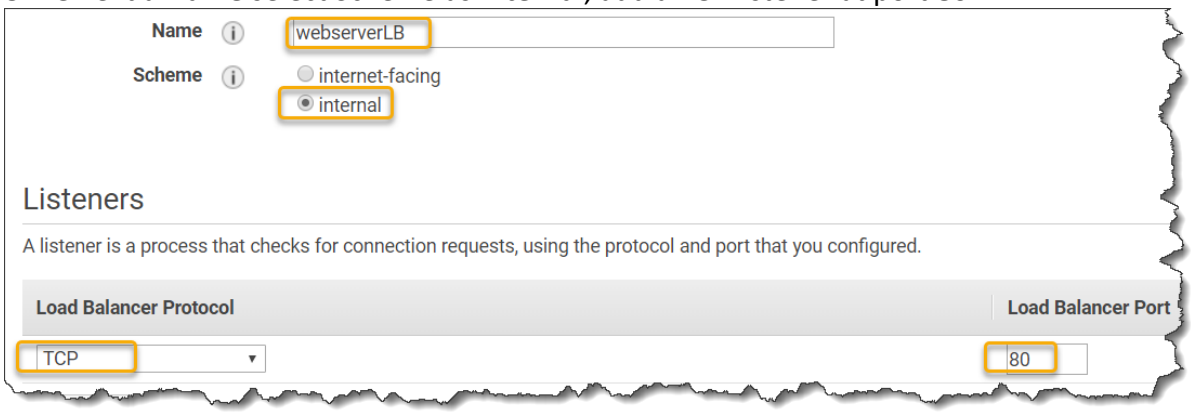
6. Click on **Create Load Balancer**



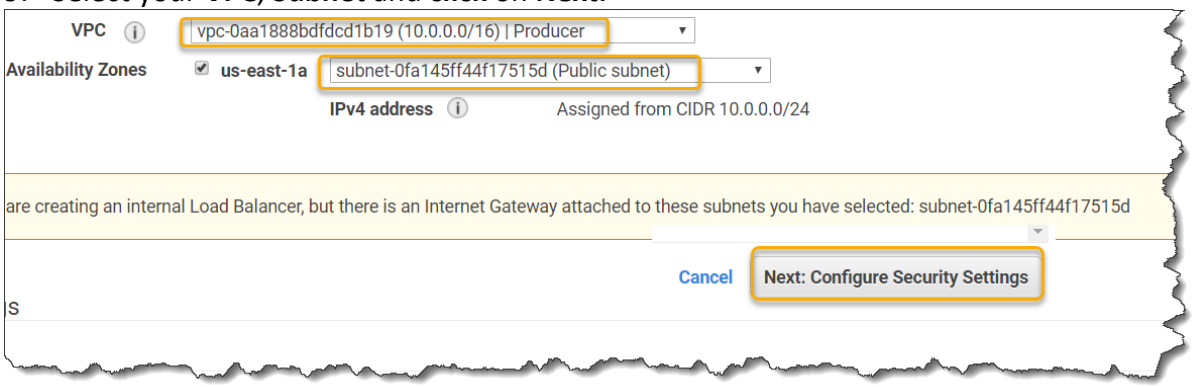
7. Under Network Load Balancer click on **Create**



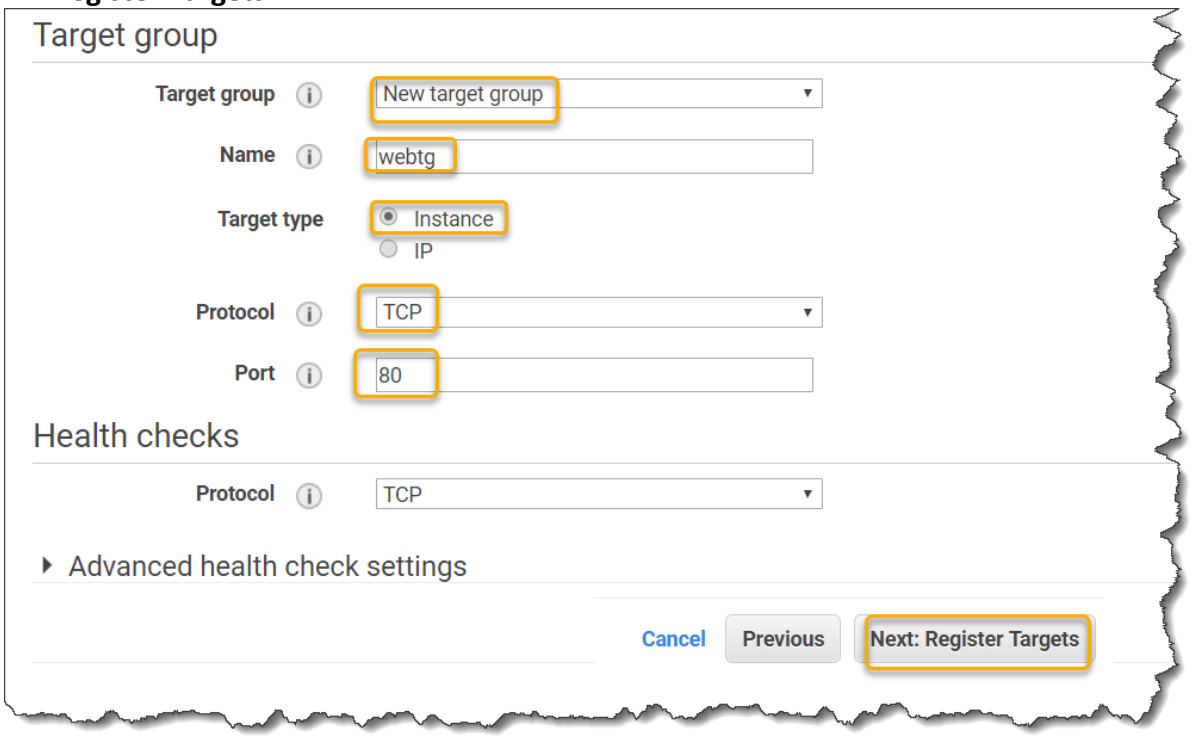
8. Give it a **Name** select Scheme as **internal**, add a **TCP** listener at port **80**



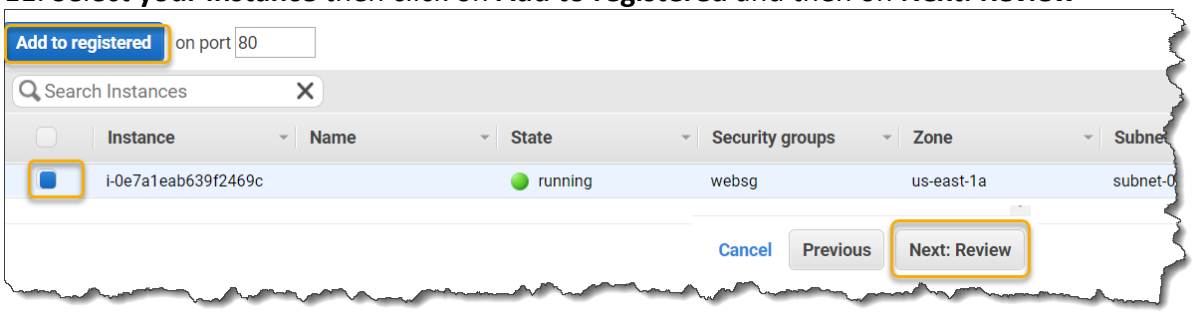
9. Select your **VPC**, **Subn**et and **click** on **Next:**



10. Click on **Next: Create Target Group** provide a **Name**, type as **instance** and click on **Next: Register Targets**

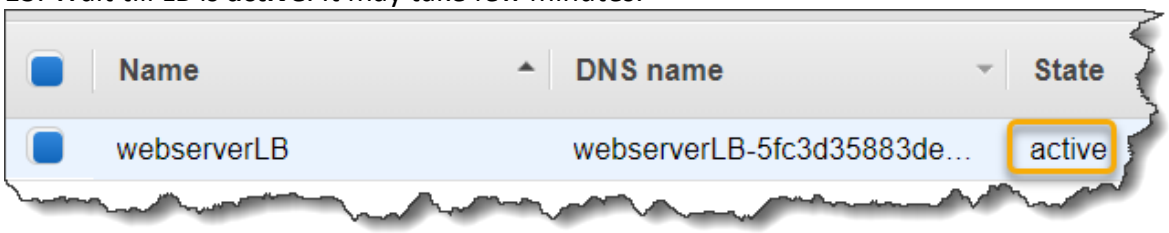


11. Select **your instance** then click on **Add to registered** and then on **Next: Review**

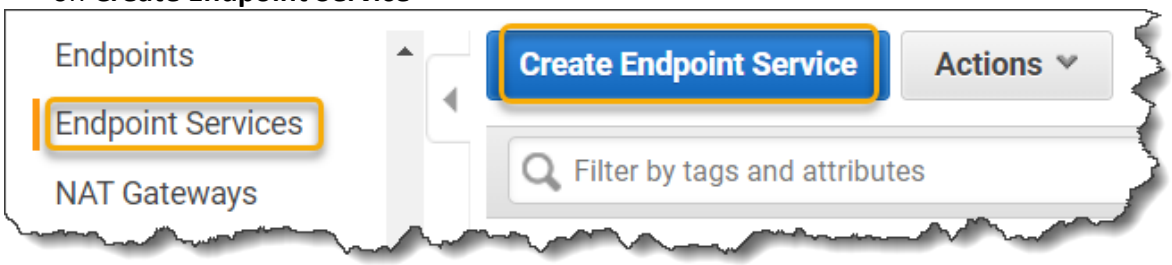


12. Review and scroll below and click on **Create**.

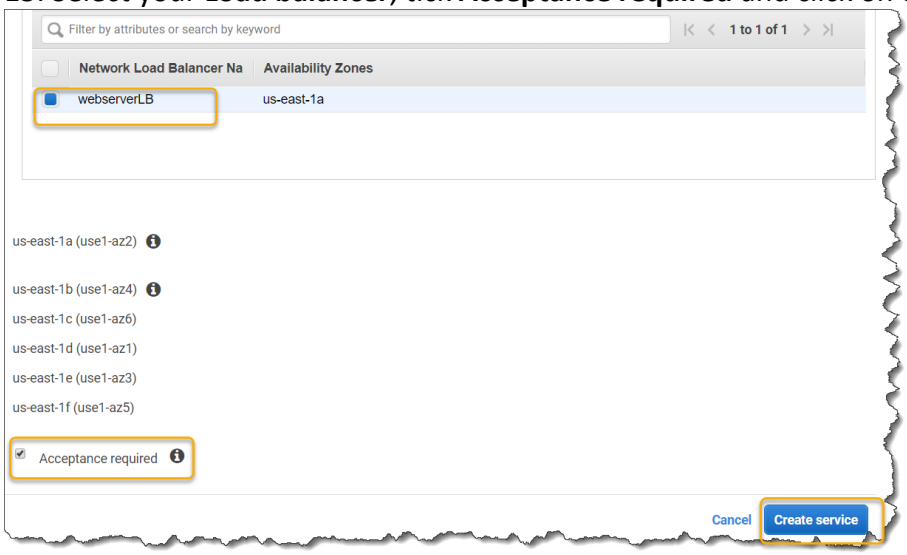
13. Wait till LB is **active**. It may take few minutes.



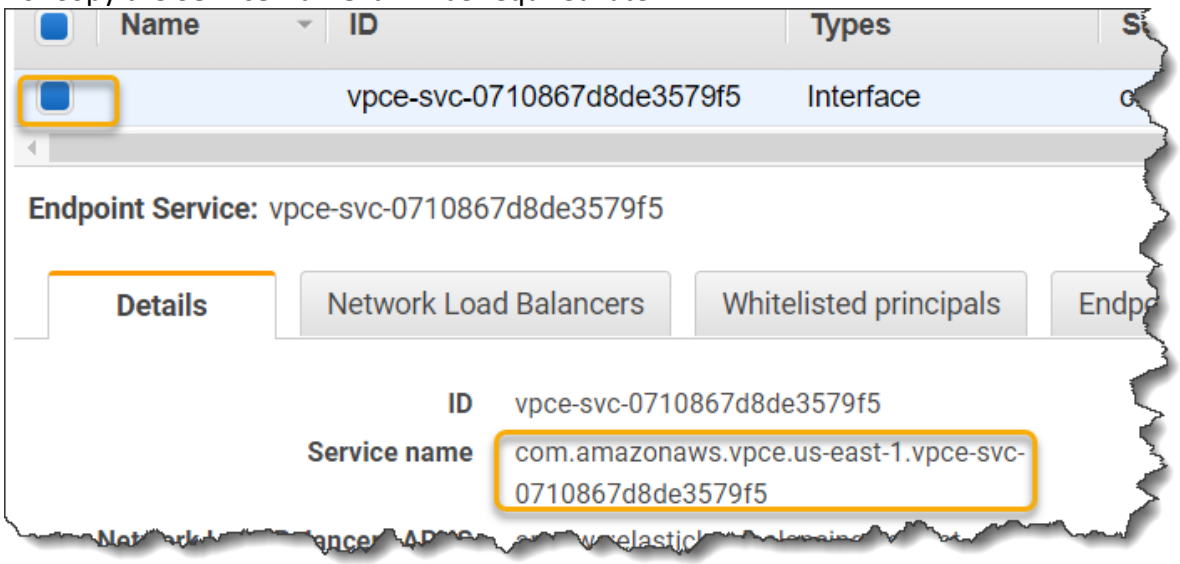
14. From left hand side of VPC console scroll down and click on **Endpoint Services** then click on **Create Endpoint Service**



15. Select your **Load balancer**, tick **Acceptance required** and click on **create service**.

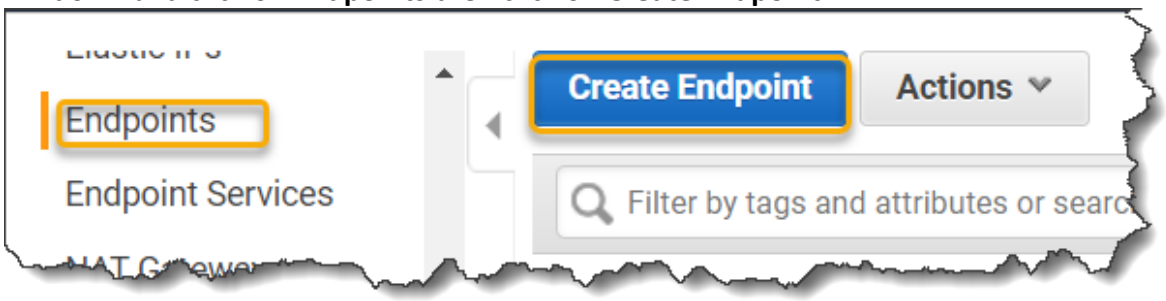


16. Copy the **Service Name** it will be required later

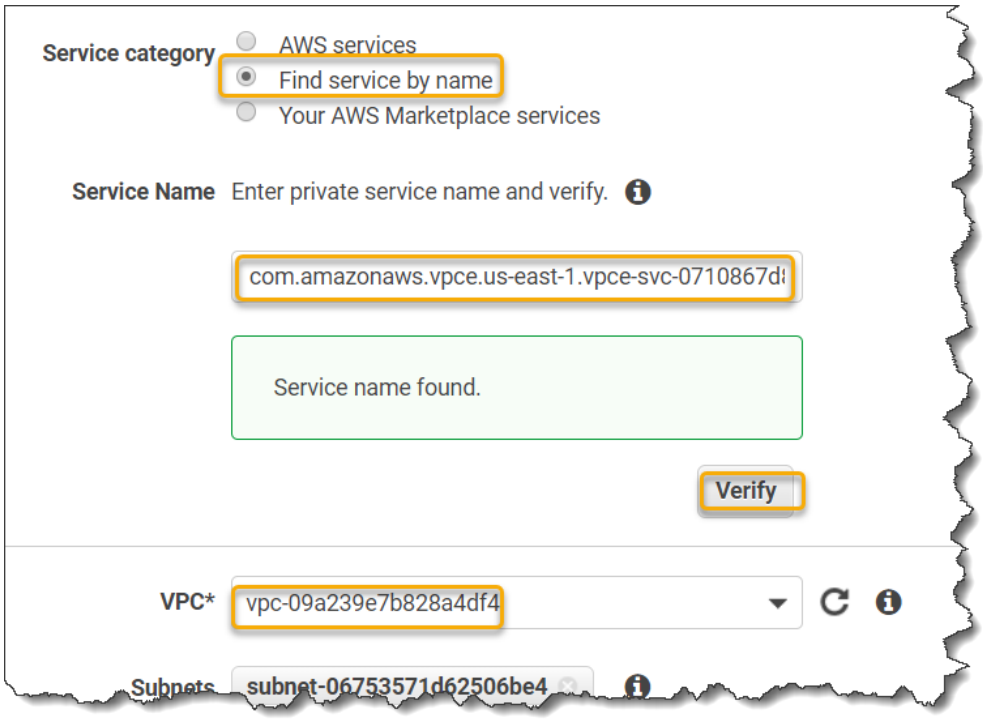


17. In consumer Launch **another instance**.

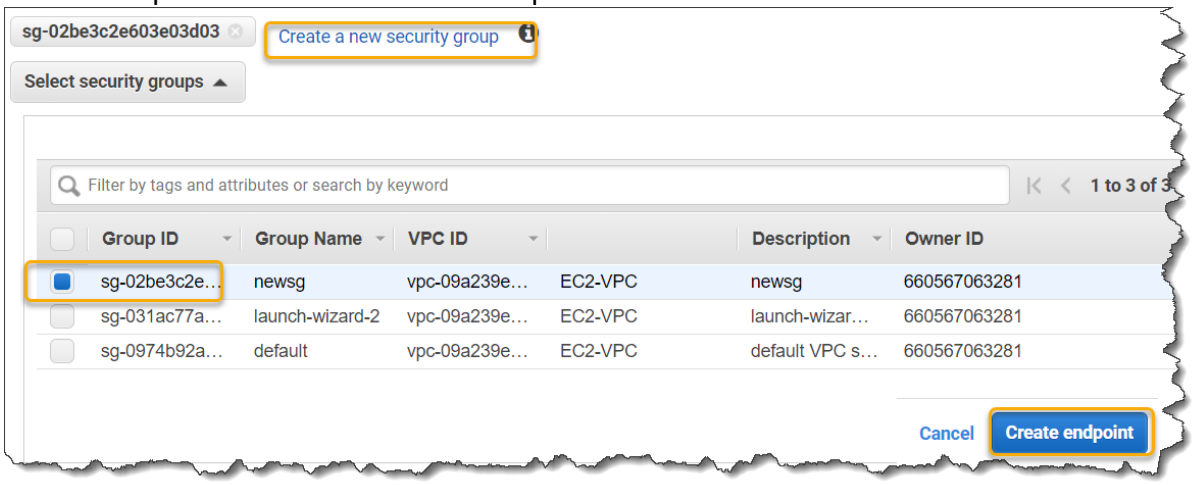
18. While instance is being launched go to VPC, from left hand side of VPC console scroll down and click on **Endpoints** then click on **Create Endpoint**



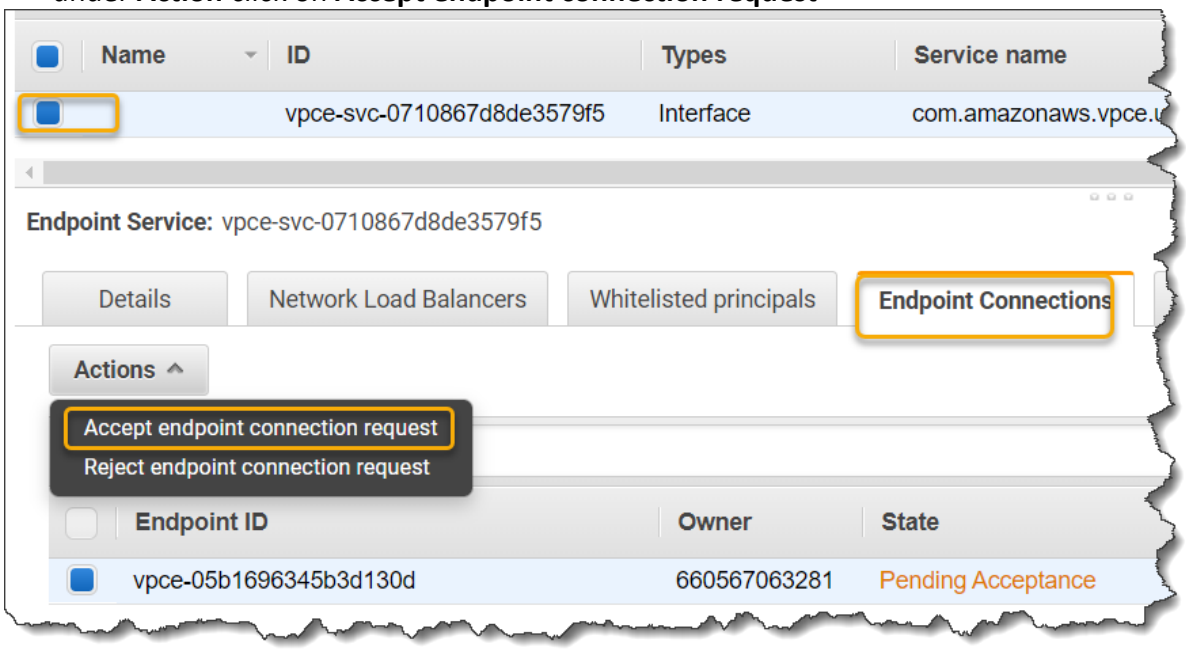
19. Select Find by service name and paste service name copied earlier and click on verify in VPC select Receiver VPC



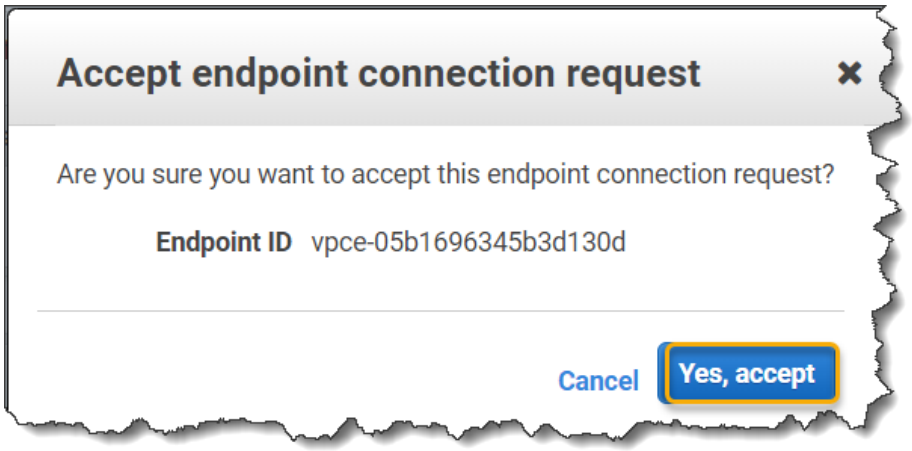
20. **Create a new Security group** which allow TCP at port 80 from anywhere and add it to the Endpoint and click on Create endpoint



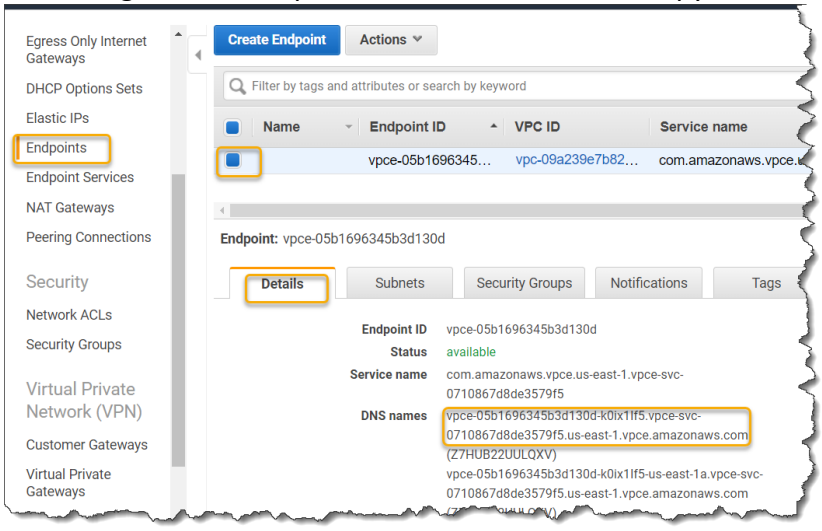
21. Now go again to Endpoint Services select **your service**, click on **Endpoint connections** under **Action** click on **Accept endpoint connection request**



22. Click on **Yes accept**



23. Now go to the endpoints on VPC console and copy DNS name



24. Now SSH to your test instance and run following command

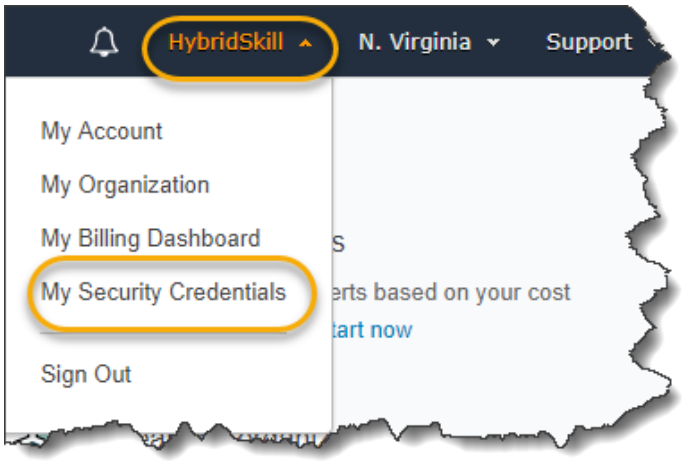
```
Curl vpce-05b1696345b3d130d-k0ix1lf5.vpce-svc-0710867d8de3579f5.us-east-1.vpce.amazonaws.com
```

Task 12: Download Security Credentials

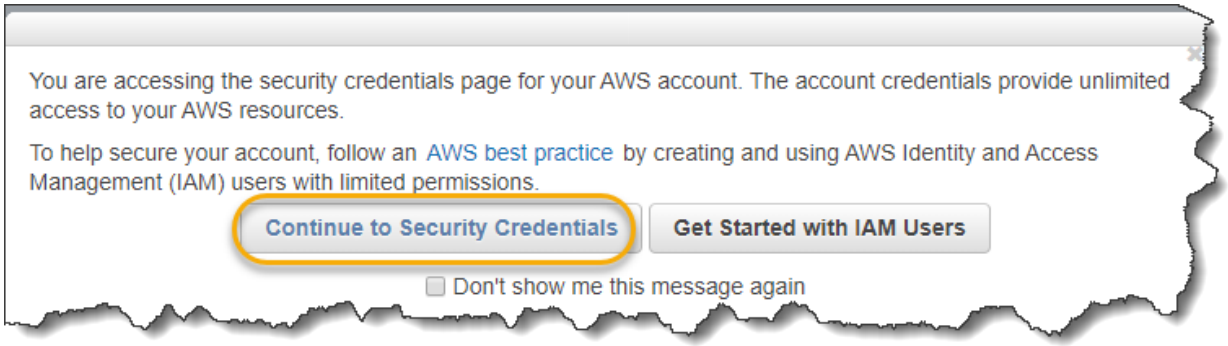
First we will download an Access key and Secret Key to sign our API calls. Depending on whether you have created your own account or are sub account under a corporate account follow either step A or B.

A. For Root account users

1. Log into the console, at the top left click on your **login name** and from the dropdown click on **Security Credentials.**

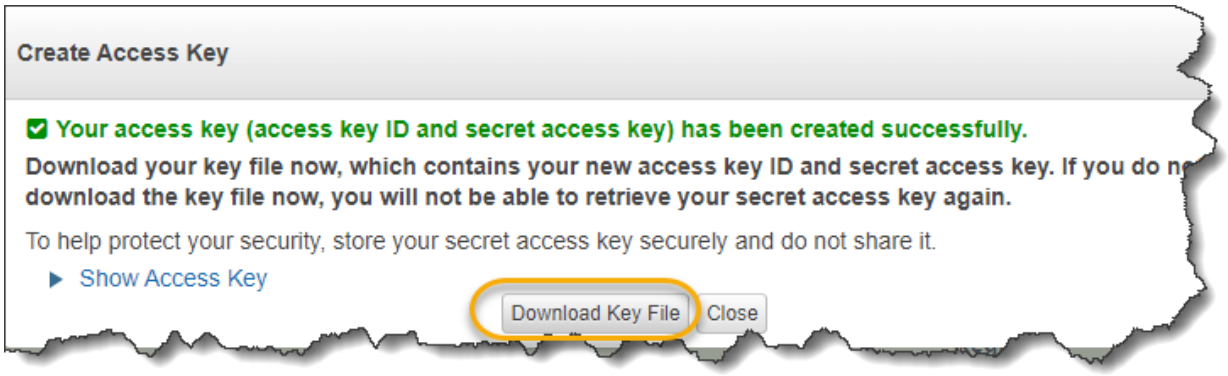


2. Click **Continue to Security Credentials**



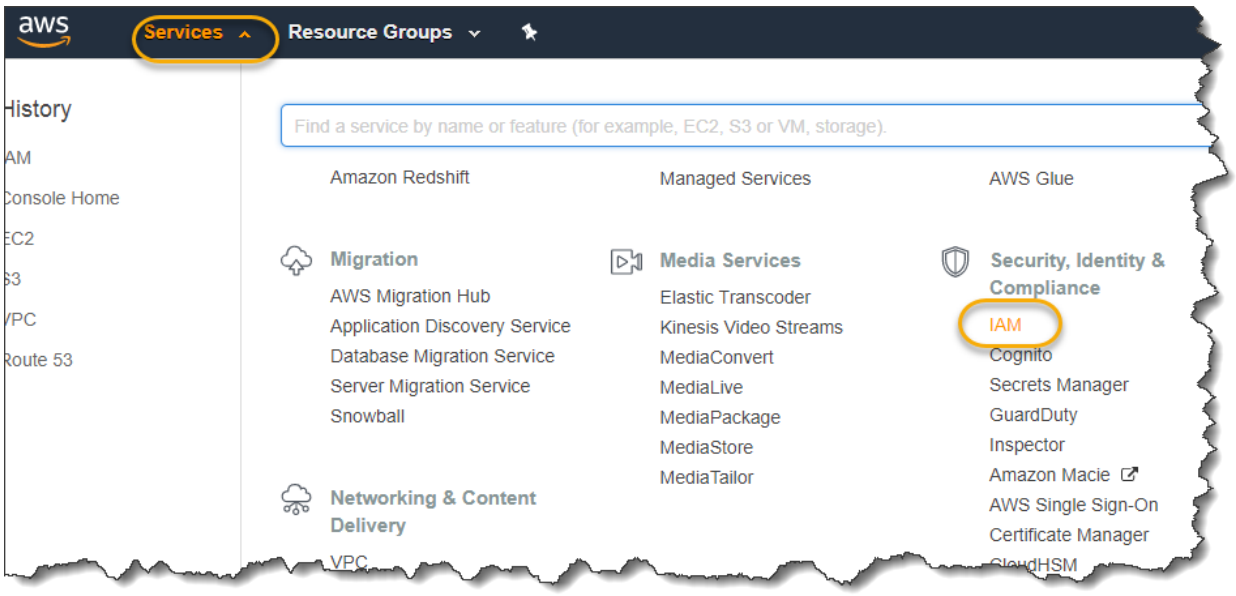
3. Expand **Access keys (access key ID and secret access key)** and click **Create New Access Key**

4. Click **Download Key File**

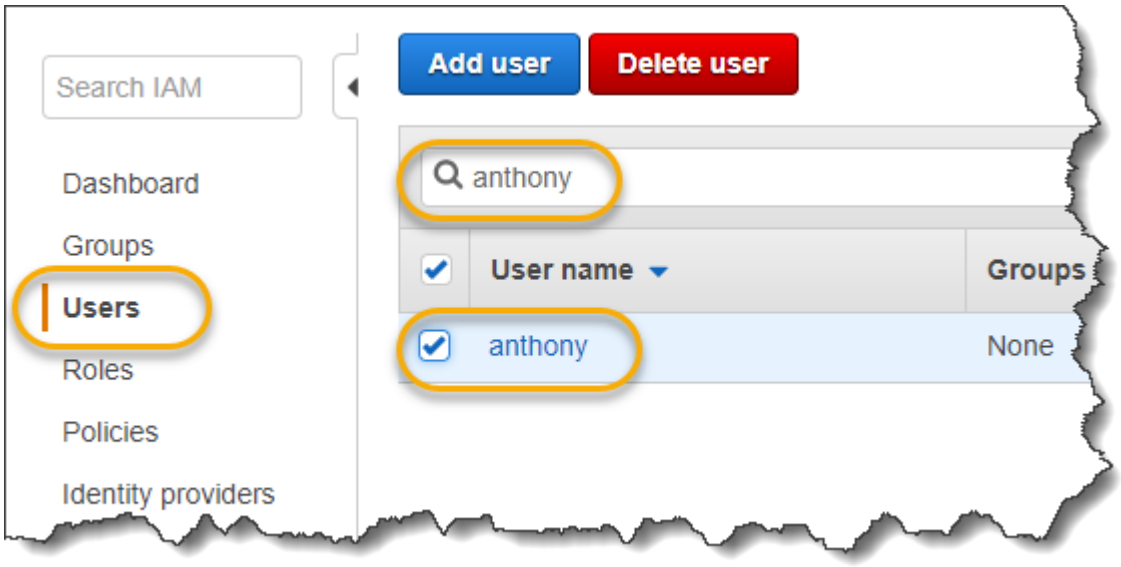


B. For IAM subaccount users

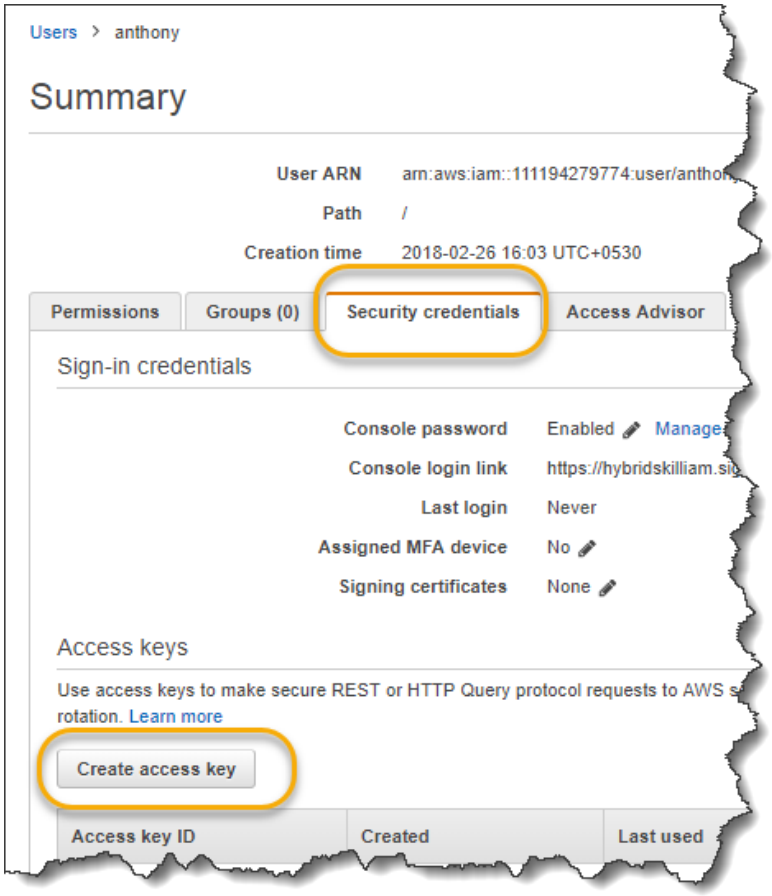
1. Click **Services** and under **Security Identity and Compliance** click **IAM**



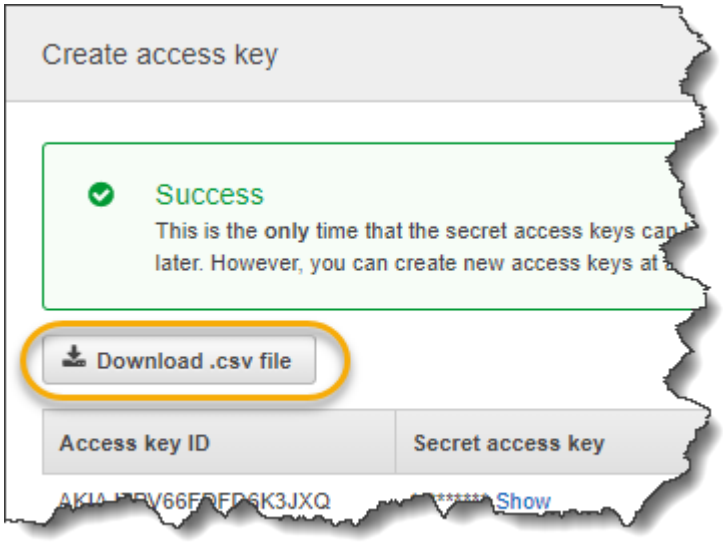
2. Click **Users** search for your **username** and click on it for more details



3. Click on the **Security credentials** tab and click on **Create access key**



Finally click on **Download .csv file**



Task 13: Setup the AWS CLI

In this task we are going to setup the AWS CLI and authorize it to perform command line API calls using the Access keys you downloaded earlier

1. For Windows, Users go to http://aws.amazon.com/cli/ and download and run the 64-bit or 32-bit Windows Installer

2. For MAC users type the following commands

```
1. sudo curl https://bootstrap.pypa.io/ez_setup.py -o - | sudo python
2. sudo easy_install pip
3. sudo pip install awscli
4. export LC_ALL=en_US.UTF-8
```

Next open a Command prompt or Terminal and type the below command. Note that "**YOUR_AWS_ACCESS_KEY**" should be replaced by your actual access key and "**YOUR_AWS_SECRET_KEY**" should be replaced by an actual secret key that was downloaded.

```
aws configure

AWS Access Key ID [None]: YOUR_AWS_ACCESS_KEY

AWS Secret Access Key [None]: YOUR_AWS_SECRET_KEY

Default region name [None]: ap-southeast-1

Default output format [None]:
```

Task 14: Manage VPC through CLI

Now that we have explored VPC through the console, let's do the same through the CLI. Run the following commands on the command line interface, you had setup earlier.

1. Create VPC and note down its vpc id:

```
:~$ aws ec2 create-vpc --cidr-block 10.10.0.0/16

{
   "Vpc": {
      "CidrBlock": "10.10.0.0/16",
       ...
      "VpcId": "vpc-602b9308",
      "CidrBlockAssociationSet": [ {
             ..
            "CidrBlock": "10.10.0.0/16",
             ..
        } ] }
}
```

2. Create Subnet and note down subnet id

```
:~$ aws ec2 create-subnet --vpc-id vpc-602b9308 --cidr-block 10.10.1.0/24

{
   "Subnet": {
          ..
          "SubnetId": "subnet-ecd56d84",
          ..
             }
}
```

3. Create gateway and note down the gateway ID:

```
:~$ aws ec2 create-internet-gateway
{
   "InternetGateway": {
      "Attachments": [],
      "InternetGatewayId": "igw-49576c20",
      "Tags": []
   }
}
```

4. Attach Internet gateway to VPC

```
:~$ aws ec2 attach-internet-gateway --vpc-id vpc-602b9308  --internet-gateway-id igw-49576c20
```

5. Create Route Table

```
:~$ aws ec2 create-route-table  --vpc-id vpc-602b9308
{
   "RouteTable": {
      ..
      "RouteTableId": "rtb-4fbcc827",
      ..
   }
}
```

6. Add a rule to route table for internet gateway

```
:~$  aws ec2 create-route --route-table-id rtb-4fbcc827 --destination-cidr-block 0.0.0.0/0 --gateway-id
igw-49576c20
{
   "Return": true
}

:~$ aws ec2 describe-route-tables --route-table-id rtb-4fbcc827
```

7. Associate to route table to subnet

```
:~$ aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-602b9308" --query
'Subnets[*].{ID:SubnetId,CIDR:CidrBlock}'

[
   {
```

```
      "ID": "subnet-ecd56d84",
      "CIDR": "10.10.1.0/24"
   },
   {
      "ID": "subnet-0ed36b66",
      "CIDR": "10.10.2.0/24"
   }
]
```

8. This will make a subnet public:

```
:~$ aws ec2 associate-route-table  --subnet-id  subnet-ecd56d84 --route-table-id rtb-4fbcc827
{
   "AssociationId": "rtbassoc-adc92dc6"
}
```

9. Enable public IP addressing :

```
:~$ aws ec2 modify-subnet-attribute --subnet-id  subnet-ecd56d84 --map-public-ip-on-launch
```

10. Launch wordpress server from AMI:
    a. Create Security group

```
:~$ aws ec2 create-security-group --group-name webserver-nondefault  --vpc-id vpc-602b9308 --description "for non default vpc"
{
   "GroupId": "sg-4cea5827"
}
```

11. Add ingress rules

```
:~$ curl ipinfo.io/ip
106.200.201.58

:~$  aws ec2 authorize-security-group-ingress --group-id sg-4cea5827 --protocol tcp --port 22 --cidr 106.200.201.58/32

:~$ aws ec2 authorize-security-group-ingress --group-id sg-4cea5827 --protocol icmp --port all --cidr 106.200.201.58/32

:~$ aws ec2 authorize-security-group-ingress --group-id sg-4cea5827 --protocol tcp --port 80 --cidr 0.0.0.0/0
```

    c. Launch box

```
:~$aws ec2 run-instances --image-id ami-0e356a61  --instance-type t2.micro --key-name hybridskill-test  --security-group-ids sg-4cea5827 --subnet-id subnet-ecd56d84
```

12. Launch db box in private subnet:
    a. The other subnet with default route table attached is the private subnet.
    b. Create security group for db

```
:~$ aws ec2 create-security-group --group-name db-nondefault  --vpc-id vpc-602b9308 --description "for non default vpc"
```

13. Add ingress rules

```
:~$ aws ec2 authorize-security-group-ingress --group-id sg-f7dd6f9c  --protocol tcp --port 22 --cidr 106.200.201.58/32

:~$ aws ec2 authorize-security-group-ingress --group-id sg-f7dd6f9c --protocol icmp --port all --cidr 10.10.0.0/16
```

14. Start db box:

```
:~$ aws ec2 run-instances --image-id ami-e60e5a89  --instance-type t2.micro --key-name hybridskill-test  --security-group-ids sg-f7dd6f9c --subnet-id subnet-0ed36b66
```

15. Ping WebServer from local box:

```
:~$ ping 13.126.142.213
PING 13.126.142.213 (13.126.142.213): 56 data bytes
64 bytes from 13.126.142.213: icmp_seq=0 ttl=53 time=72.152 ms
64 bytes from 13.126.142.213: icmp_seq=1 ttl=53 time=72.130 ms
…
```

16. Login to Web Server:

```
:~$ ssh -i hybridskill-test.pem ec2-user@13.126.142.213

[ec2-user@ip-10-10-1-135 ~]$ ping google.com
PING google.com (216.58.211.174) 56(84) bytes of data.
64 bytes from dub08s01-in-f174.1e100.net (216.58.211.174): icmp_seq=1 ttl=43 time=243 ms
64 bytes from dub08s01-in-f174.1e100.net (216.58.211.174): icmp_seq=2 ttl=43 time=243 ms
^X64 bytes from dub08s01-in-f174.1e100.net (216.58.211.174): icmp_seq=3 ttl=43 time=243 ms
64 bytes from dub08s01-in-f174.1e100.net (216.58.211.174): icmp_seq=4 ttl=43 time=243 ms
```

17. Login to db server via Web Server:

```
[ec2-user@ip-10-10-1-135 ~]$ ssh ec2-user@10.10.2.110
Last login: Mon Mar  5 21:14:39 2018 from 10.10.1.135
[ec2-user@ip-10-10-2-110 ~]$ ping google.com
PING google.com (172.217.166.78) 56(84) bytes of data.

^C
--- google.com ping statistics ---
74  ackets transmitted, 0 received, 100% packet loss, time 72999ms
```

18. Launch **NAT** instance. First Create security group for NAT

```
:~$ aws ec2 create-security-group --group-name nat-nondefault  --vpc-id vpc-602b9308 --description "for non default vpc"
```

```
{
   "GroupId": "sg-80cc7eeb"
}
```

19. Create security group Ingress rule

```
:~$ aws ec2 authorize-security-group-ingress --group-id sg-80cc7eeb  --protocol tcp --port 22 --cidr 106.200.201.58/32

:~$ aws ec2 authorize-security-group-ingress --group-id sg-80cc7eeb --protocol icmp --port all --cidr 10.10.0.0/16
```

20. Launch box in public subnet using the R**ed Hat Enterprise Linux 7.4 AMI**

```
:~$ aws ec2 run-instances --image-id ami-e60e5a89  --instance-type t2.micro --key-name hybridskill-test  --security-group-ids sg-80cc7eeb --subnet-id subnet-ecd56d84
```

21. Login into the NAT box:

```
~:$  ssh -i hybridskill-test.pem ec2-user@13.126.142.213
~:$ sysctl net.ipv4.ip_forward=1
~:$  iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

22. Add rule in private route table for nat instance ID

```
~:$ aws ec2 create-route --route-table-id  rtb-e4b3c78c --destination-cidr-block 0.0.0.0/0  --instance-id i-0663e364fdb4dc43f
{
   "Return": true
}
```

23. Disable Source/Dest-check

```
~:$  aws ec2 modify-instance-attribute  --instance-id i-0663e364f5b48c67f --no-source-dest-check
```

Not login to db server and try to and ping the Internet.