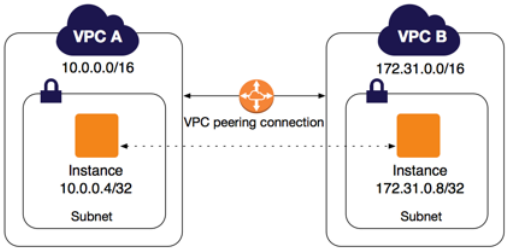
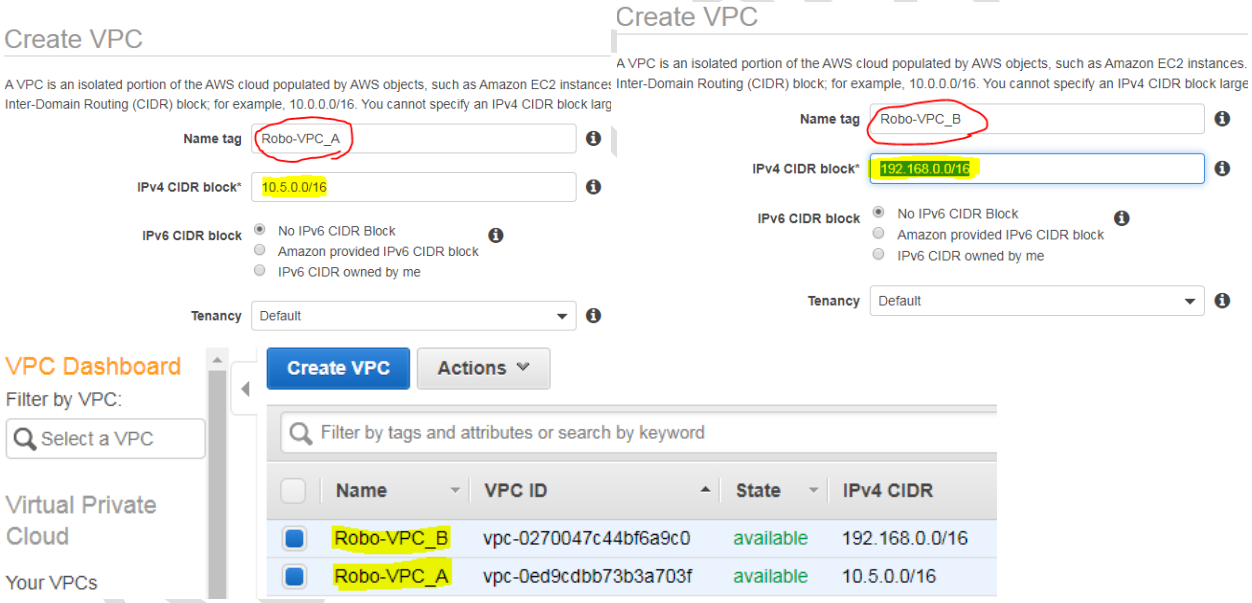
Amazon VPC Peering

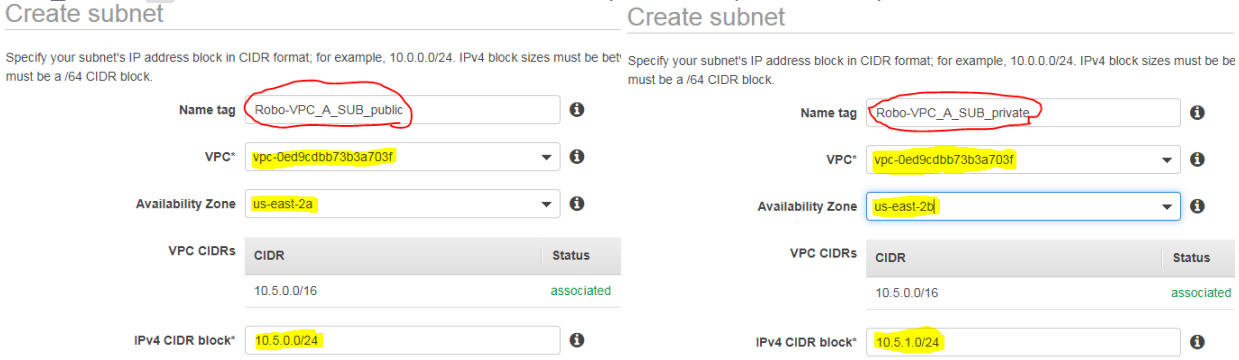
A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IPv4 addresses or IPv6 addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, or with a VPC in another AWS account. The VPCs can be in different regions (also known as an inter-region VPC peering connection).
https://docs.aws.amazon.com/vpc/latest/peering/working-with-vpc-peering.html

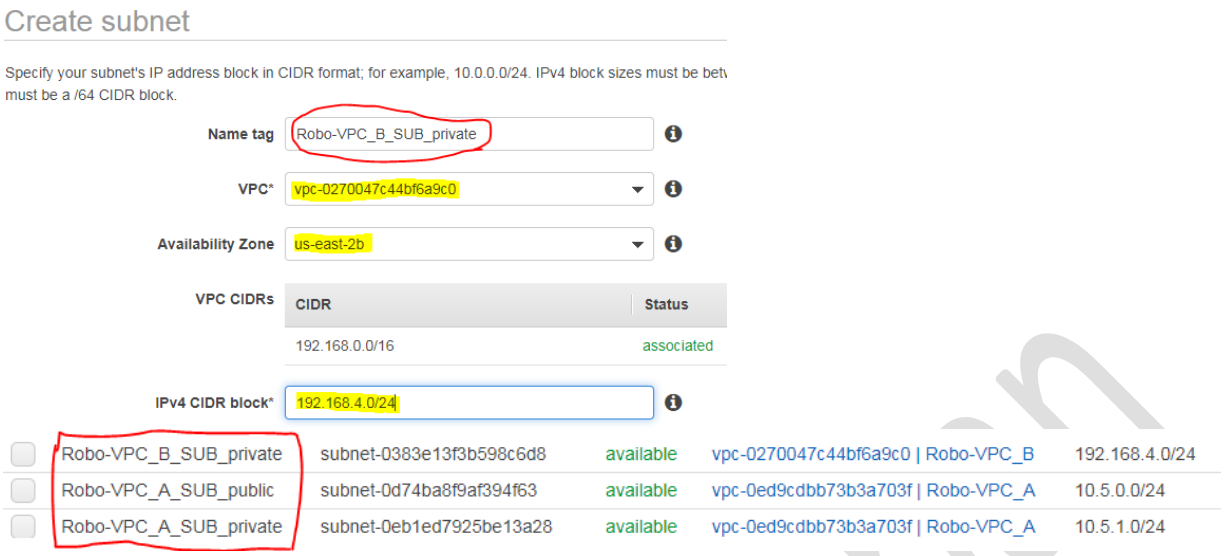


Step 1:- Create two VPC with different range of ip address.
Name:- Robo-VPC_A (10.5.0.0/16) and Robo-VPC_B (192.168.0.0/16)



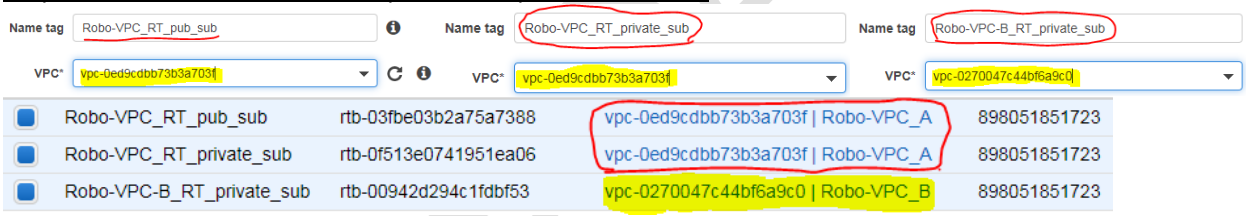
Step2:-Create three subnets in that for Robo-VPC_A assign two subnets and for assign one for Robo-VPC_B and make sure to select different Availability Zone for private and public.





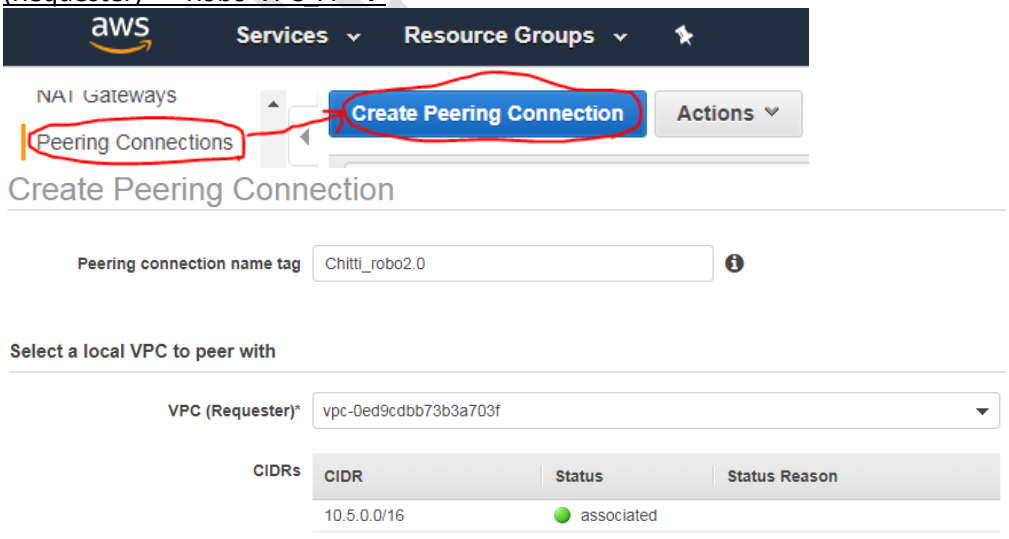
Note:- Here for testing purpose I have selected Availability Zone:-<US-east-2b> for private subnets on both VPC and Availability Zone:-<US-east-2a> for public subnet in Robo-VPC_A

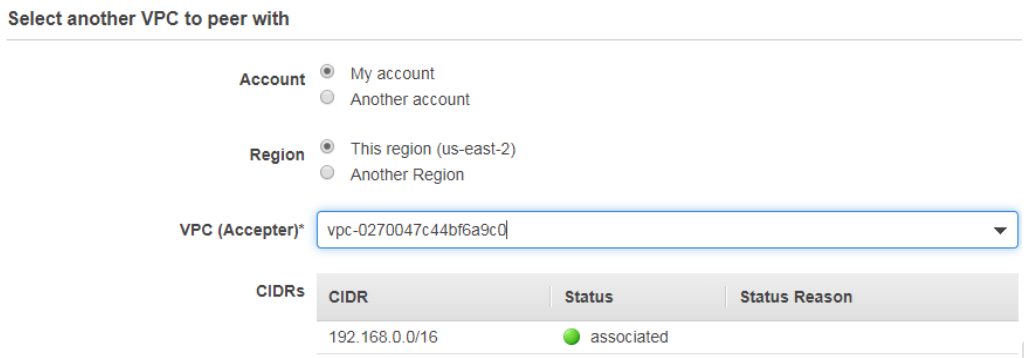
Step3:-Create a route tables for public and private subnets.

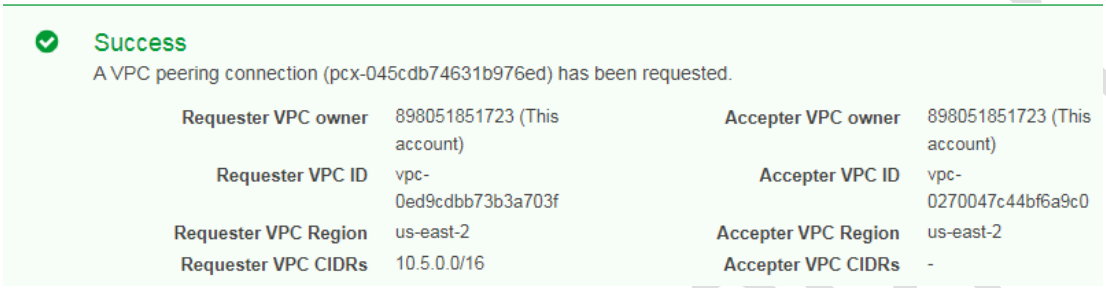


Note:- Here two route table(private&public) has been created Robo-VPC_A and one route table (private) has been created for Robo-VPC_B

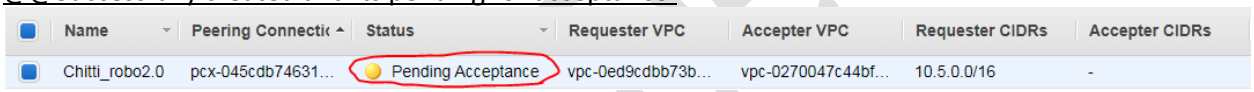
Step4:-Create peering connection between Robo-VPC_A and Robo-VPC_B
@Click on Peering Connections → Create Peering Connection → enter the Name tag → select VPC (Requester)* < Robo-VPC_A> →



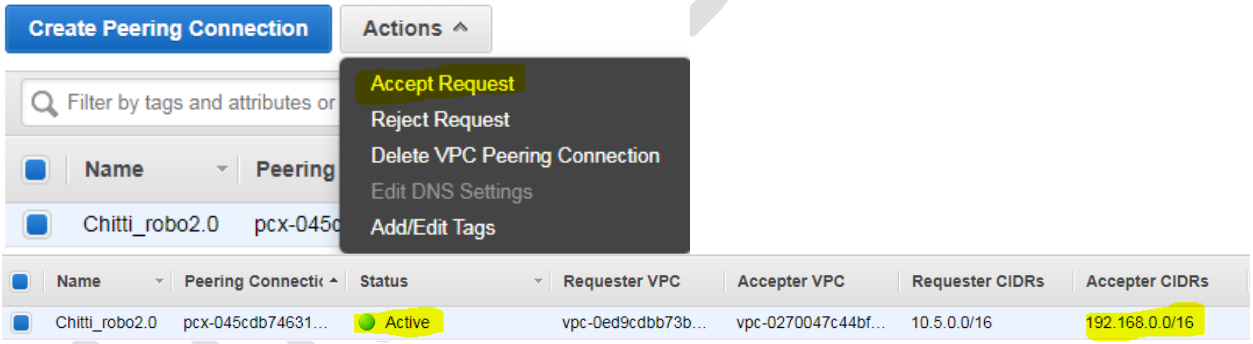




@@Successfully created and its pending for acceptance.

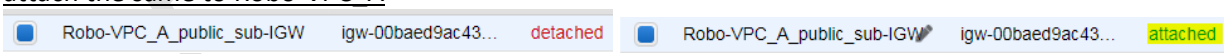


@Click on Actions → then select Accept Request



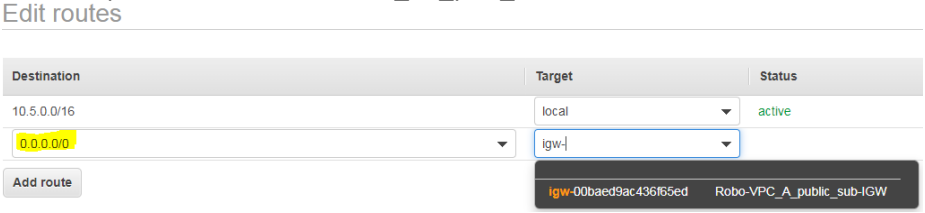
Step5:-Create a internet gateway and attach the same VPC

@Click on Internet Gateways → Create internet gateway → enter the name tag → after creation then attach the same to Robo-VPC_A

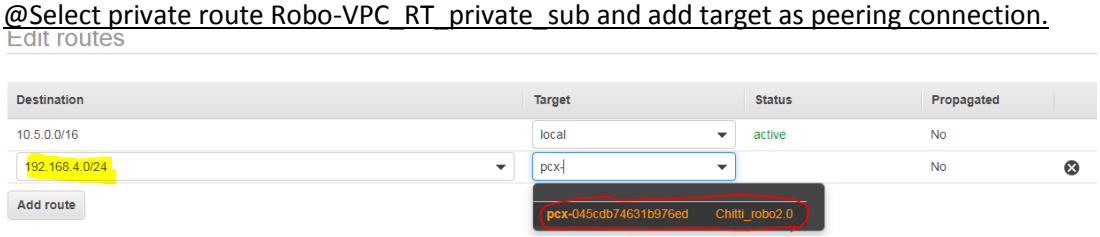


Step5.1 click on route tables and add rule for target IGW. Routes → Edit routes → Add route

@Select public routes Robo-VPC_RT_pub_sub and edit route.



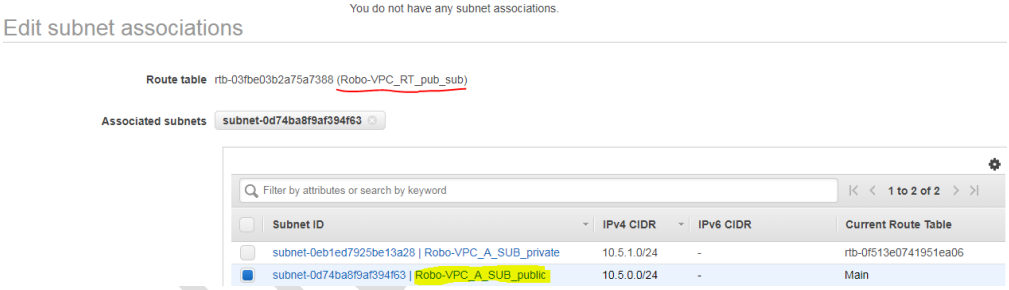
@Select private route Robo-VPC_RT_private_sub and add target as peering connection.



@Select private route Robo-VPC-B_RT_private_sub add

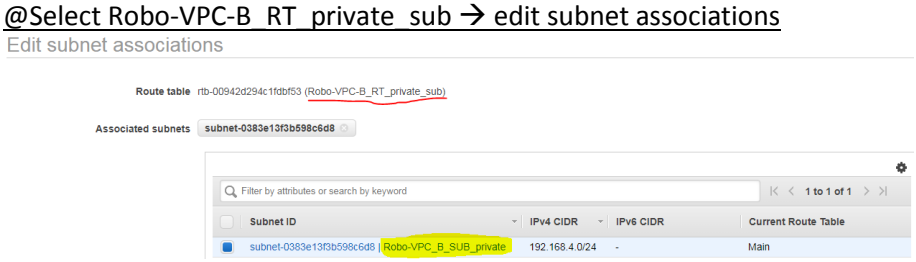
Step5.1 Associate respective subnet.

@Select Robo-VPC_RT_pub_sub → edit subnet associations



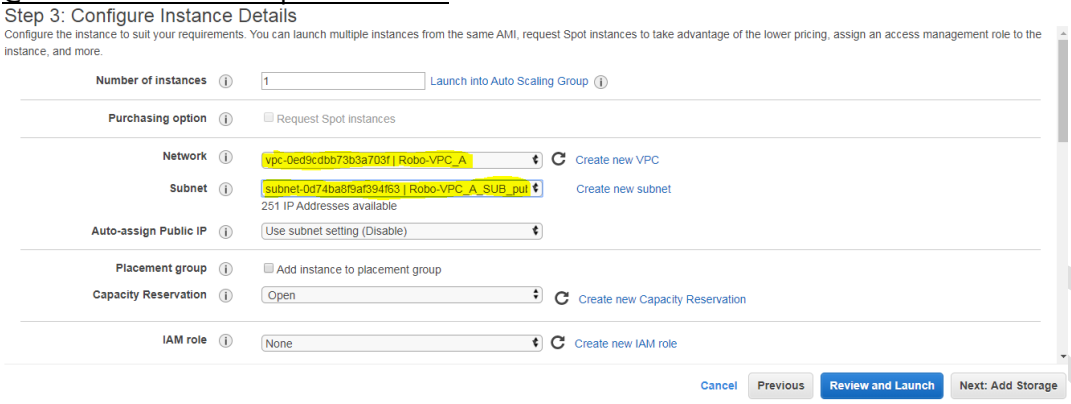
@Select Robo-VPC_RT_private_sub → edit subnet associations

@Select Robo-VPC-B_RT_private_sub → edit subnet associations

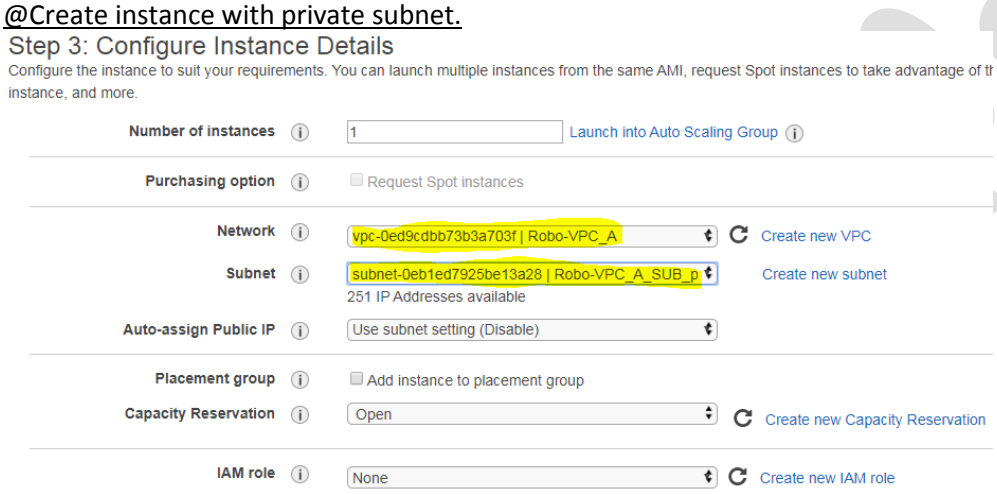


Step6:-Launch instance

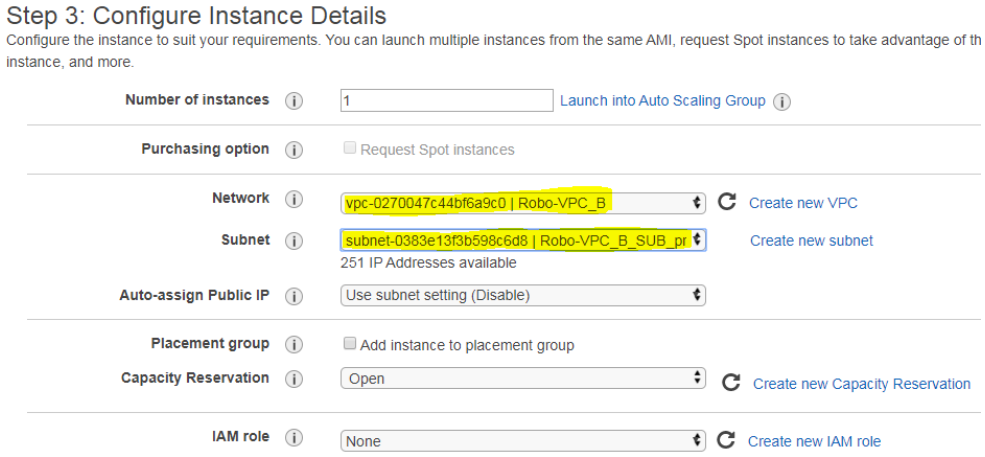
@Create instance with public subnet.



@Create instance with private subnet.

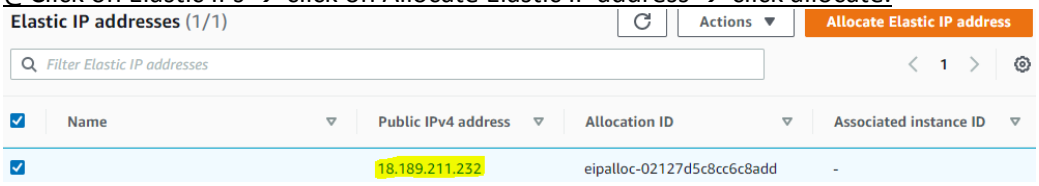


@Create instance with private subnet of VPC B

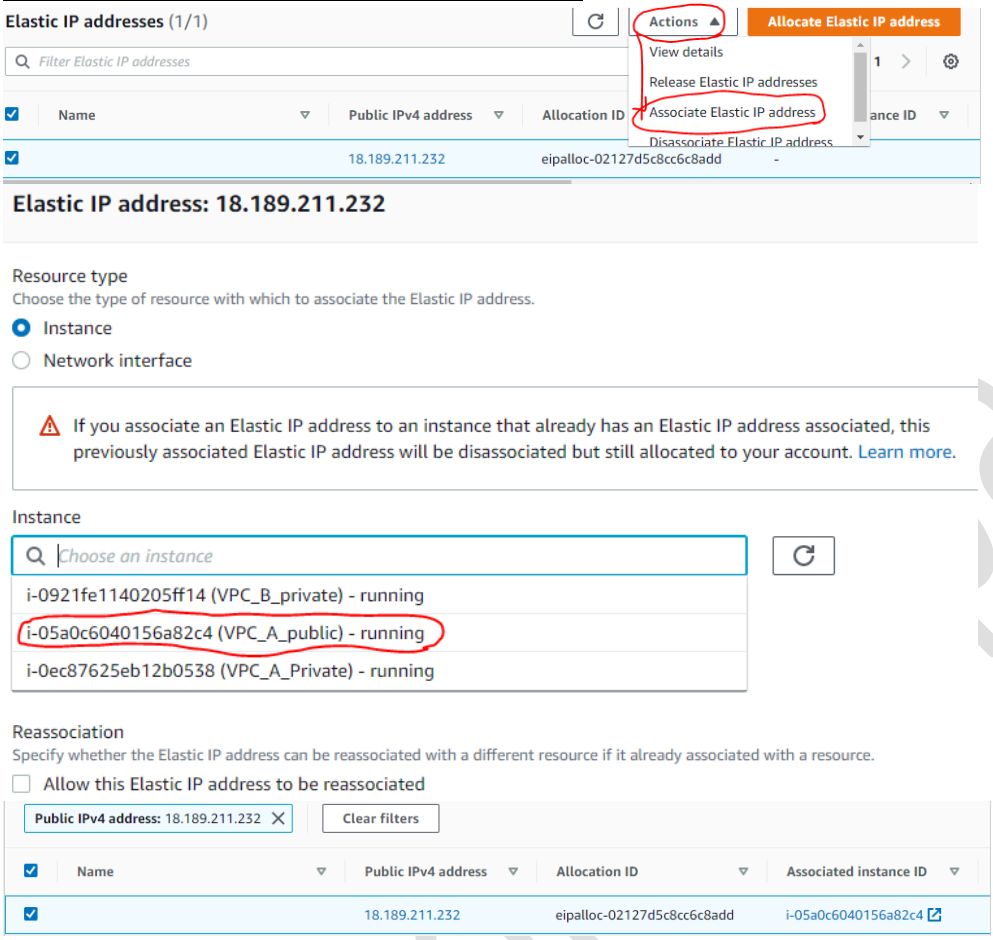


Step 7:-Assign EIP to public instance subnet.

@Click on Elastic IPs → click on Allocate Elastic IP address → click allocate.



@Click on Actions → Associate Elastic IP address



Step 8:- Now try to access server via public ip.

```
[centos@ip-10-5-0-192 ~]$ uptime
 03:50:07 up  1:03,  1 user,  load average: 0.14, 0.08, 0.05
[centos@ip-10-5-0-192 ~]$ uname -a
Linux ip-10-5-0-192.us-east-2.compute.internal 3.10.0-957.1.3.el7.x86_64
 x86_64 GNU/Linux
[centos@ip-10-5-0-192 ~]$ hostname
ip-10-5-0-192.us-east-2.compute.internal
[centos@ip-10-5-0-192 ~]$
```

@Copy the .pem key via winscp to public instance and then try to access private instance.

#chmod 600 AWS-SSHKEY.pem

#ssh -i AWS-SSHKEY.pem centos@10.5.1.67

```
[centos@ip-10-5-0-192 ~]$ ssh -i AWS-SSHKEY.pem centos@10.5.1.67
[centos@ip-10-5-1-67 ~]$ uname -a
Linux ip-10-5-1-67.us-east-2.compute.internal 3.10.0-957.1.3.el7.x86_64
x86_64 GNU/Linux
[centos@ip-10-5-1-67 ~]$ w
 04:02:45 up  1:12,  1 user,  load average: 0.00, 0.01, 0.03
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
centos   pts/0    10.5.0.192       04:02    5.00s  0.01s  0.01s w
[centos@ip-10-5-1-67 ~]$
```

@Now come out from VPC-A private instance and try to copy .pem key from public instance to private.

#scp -r -i AWS-SSHKEY.pem AWS-SSHKEY.pem centos@10.5.1.67:~

```
[centos@ip-10-5-0-192 ~]$ scp -r -i AWS-SSHKEY.pem AWS-SSHKEY.pem centos@10.5.1.67:~
AWS-SSHKEY.pem                                100% 1692   869.9KB/s   00:00
[centos@ip-10-5-0-192 ~]$
```

@again ssh to VPC-A private instance and from there try to copy .pem key to VPC-B private and access via ssh.

#ssh -i AWS-SSHKEY.pem centos@10.5.1.67

```
[centos@ip-10-5-1-67 ~]$ w
 04:09:18 up  1:19,  1 user,  load average: 0.00, 0.01, 0.03
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
centos   pts/0    10.5.0.192       04:07    6.00s  0.02s  0.01s w
[centos@ip-10-5-1-67 ~]$
```

#ping 192.168.4.104

```
[centos@ip-10-5-1-67 ~]$ ping 192.168.4.104
PING 192.168.4.104 (192.168.4.104) 56(84) bytes of data.
64 bytes from 192.168.4.104: icmp_seq=1 ttl=64 time=0.422 ms
64 bytes from 192.168.4.104: icmp_seq=2 ttl=64 time=0.545 ms
```

#ssh -i AWS-SSHKEY.pem centos@192.168.4.104

```
[centos@ip-10-5-1-67 ~]$ ssh -i AWS-SSHKEY.pem centos@192.168.4.104
The authenticity of host '192.168.4.104 (192.168.4.104)' can't be established.
ECDSA key fingerprint is SHA256:LTwp3RF8TKb/KA72d/eWVt3dSVsY+3l4aPfjY00OPD8.
ECDSA key fingerprint is MD5:8d:0f:8b:22:c5:88:8c:8c:4b:e9:c2:c1:50:b2:0b:57.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.4.104' (ECDSA) to the list of known hosts.
```

#w

#hostname -i

```
[centos@ip-192-168-4-104 ~]$ w
 04:11:44 up  1:16,  1 user,  load average: 0.00, 0.01, 0.04
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
centos   pts/0    10.5.1.67        04:11    0.00s  0.02s  0.02s w
[centos@ip-192-168-4-104 ~]$ hostname -i
fe80::4c8:d9ff:fe2d:de42%eth0 192.168.4.104
[centos@ip-192-168-4-104 ~]$
```

Note:- Now we have created VPC peering and You can establish peering relationships between VPCs across different AWS Regions (also called Inter-Region VPC Peering). This allows VPC resources including EC2 instances, Amazon RDS databases and Lambda functions that run in different AWS Regions to communicate with each other using private IP addresses, without requiring gateways, VPN connections, or separate network appliances. The traffic remains in the private IP space. All inter-region traffic is encrypted with no single point of failure, or bandwidth bottleneck. Traffic always stays on the global AWS backbone, and never traverses the public internet, which reduces threats, such as common exploits, and DDoS attacks. Inter-Region VPC Peering provides a simple and cost-effective way to share resources between regions or replicate data for geographic redundancy.