First Name: **SaiGopi**

Last Name: **Pachipala**

Contents

1.EC2 Instances

Steps:

1. Open AWS and under search box select EC2
2. Click on launch instance
3. Now enter name of your machine (Machine1-SaiGopi)
4. under AMI select Amazon linux
5. under instance type select t2.micro
6. under key pair >> create a new key pair (TOKYO.pem)
7. under firewall security group click on create security group
8. and click on launch instance
9. Repeat the same process and create one more instance (Machine1-Pachipala)

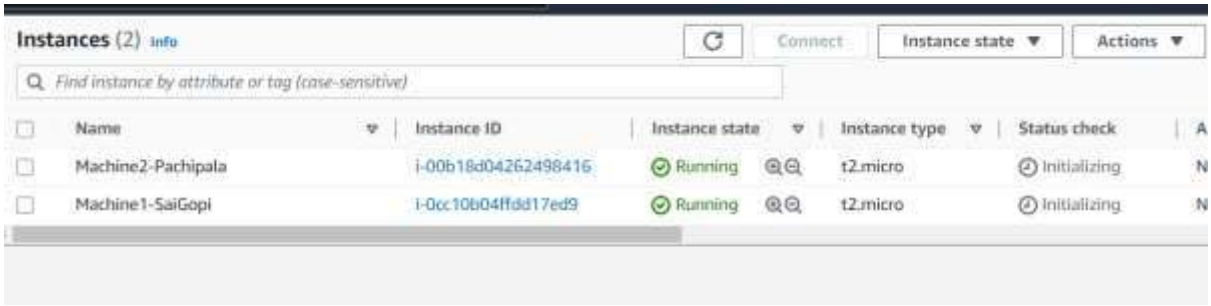


Fig1:Ec2 instances

10.Steps to connect to ec2 machines.

- Select machine1 under instances tab and click on connect.
- Under connect to instance select SSH client
- Now copy the ssh command shown under example
- Now go to the .pem file location directory and open command prompt terminal
- Now paste the ssh command and click on enter
- Type yes to connect and you can see your Machine1-Saigopi instance running

Fig2.Machine1-SaiGopi

Fig3.Machine2-Pachipala

2.EBS volume

Steps:

1. When ever you want to provide an extra storage to your machine you opt for this Elastic Block store (EBS).
2. Created Two machines SaiGopi-Machine A and SaiGopi-Machine B in sydney region.

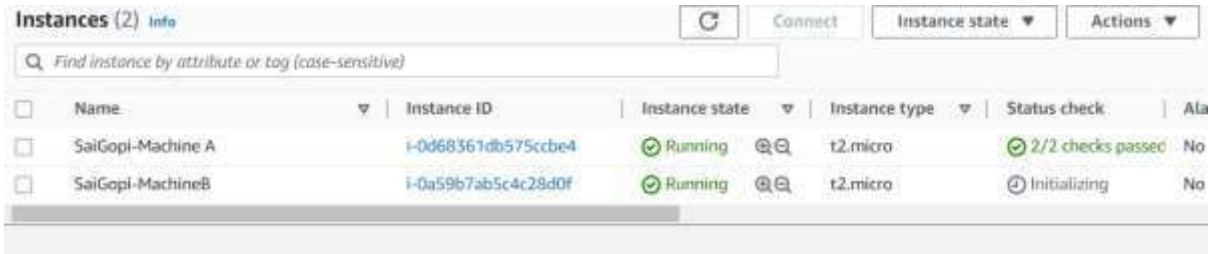


Fig4: Instances for EBS

3.Under EBS select volumes and you can see default storage allocated for your EC2 machines.

4.Now click on create volume

- Under volume type select any type you want (General purpose SSD (gp2))
- Under size select the amount of GB (1GB)
- Under Availability zone you can select available zone in which your instance got created.
- Now click on create volume.
- Now click on volumes and you can see all volumes and newly created EBS.
- Now select the EBS and click on actions and click on attach volume.
- Under Basic details select your instance and click on attach volume.

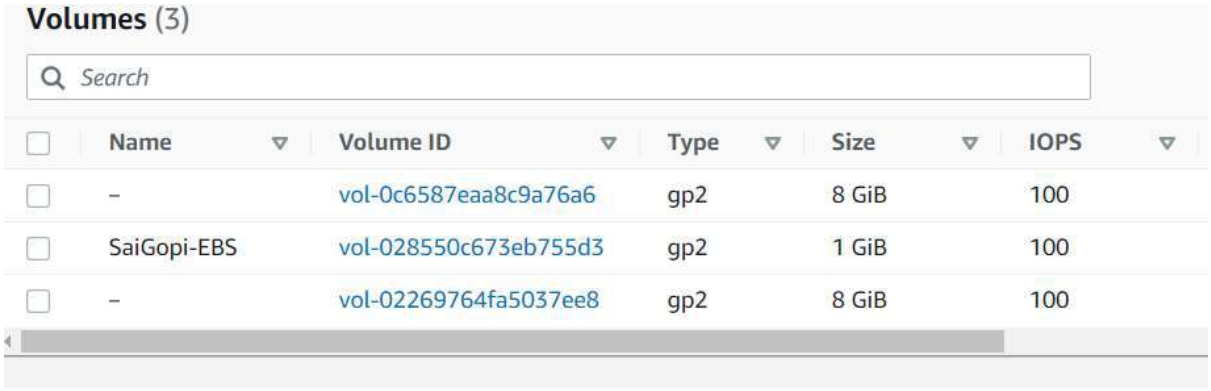


Fig5: EBS volume of 1GB

6.Now log on to SaiGopi-MachineA and make a file system and mount it.

- lsblk – to list all file systems
- mkdir <directory name> to create a storage directory
- mkfs -t xfs /dev/sdf
- mount -t xfs /dev/sdf storage
- created a storage directory named SaiGopi-Storage
- mounted it to file system and created ten .txt files in it
- umounted the file system.

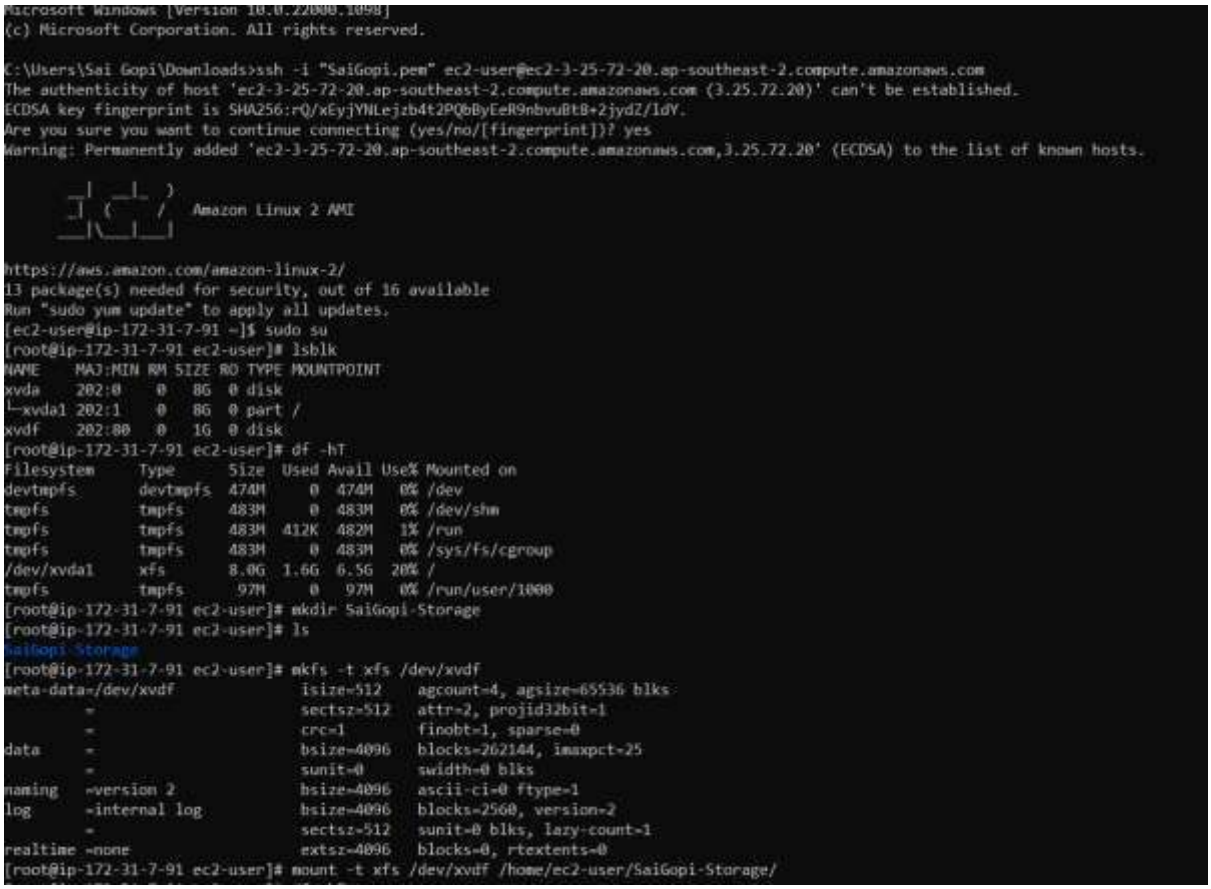


Fig 6: File system created for SaiGopi-MachineA and mounted it

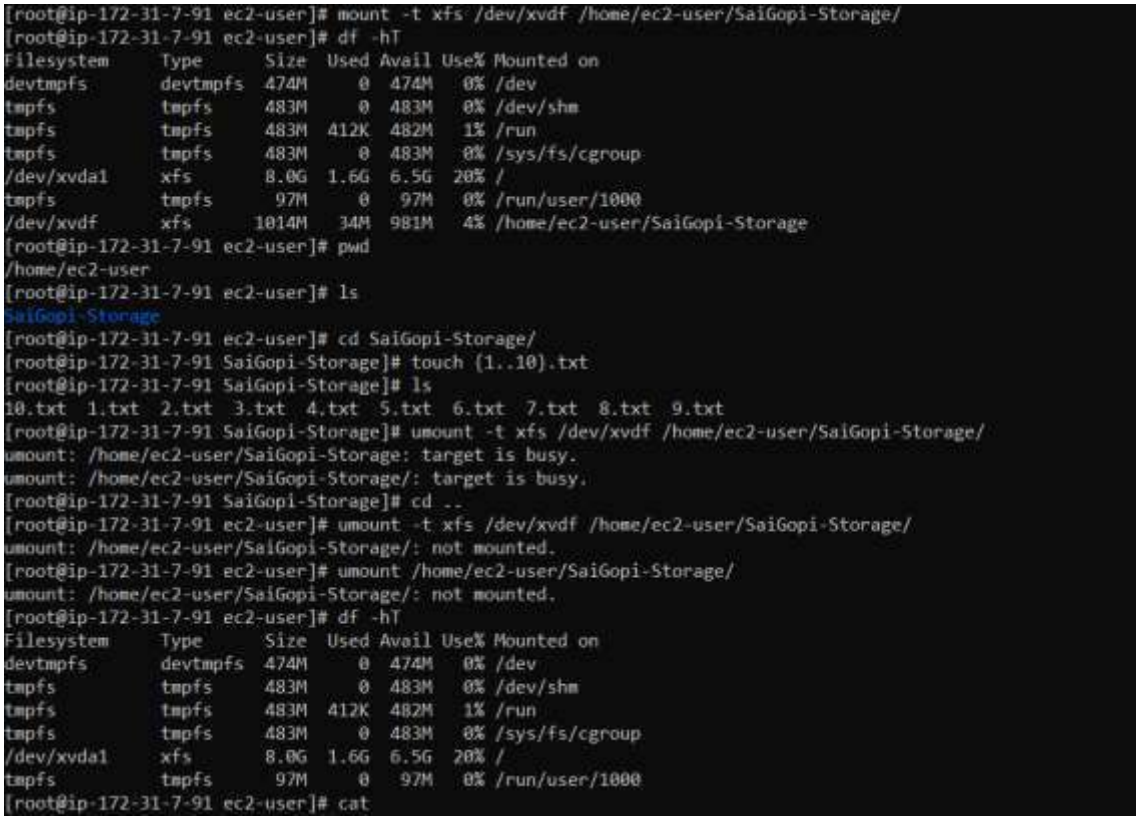


Fig7: Created 10 files in SaiGopi-Storage and unmounted it

7.Now detach the EBS volume from machine A and attach it to Machine B

8.Now connect to Machine B, create a new directory and mount the same to it.

9.SaiGopi-MachineB EBS volume contains all the ten txt files.

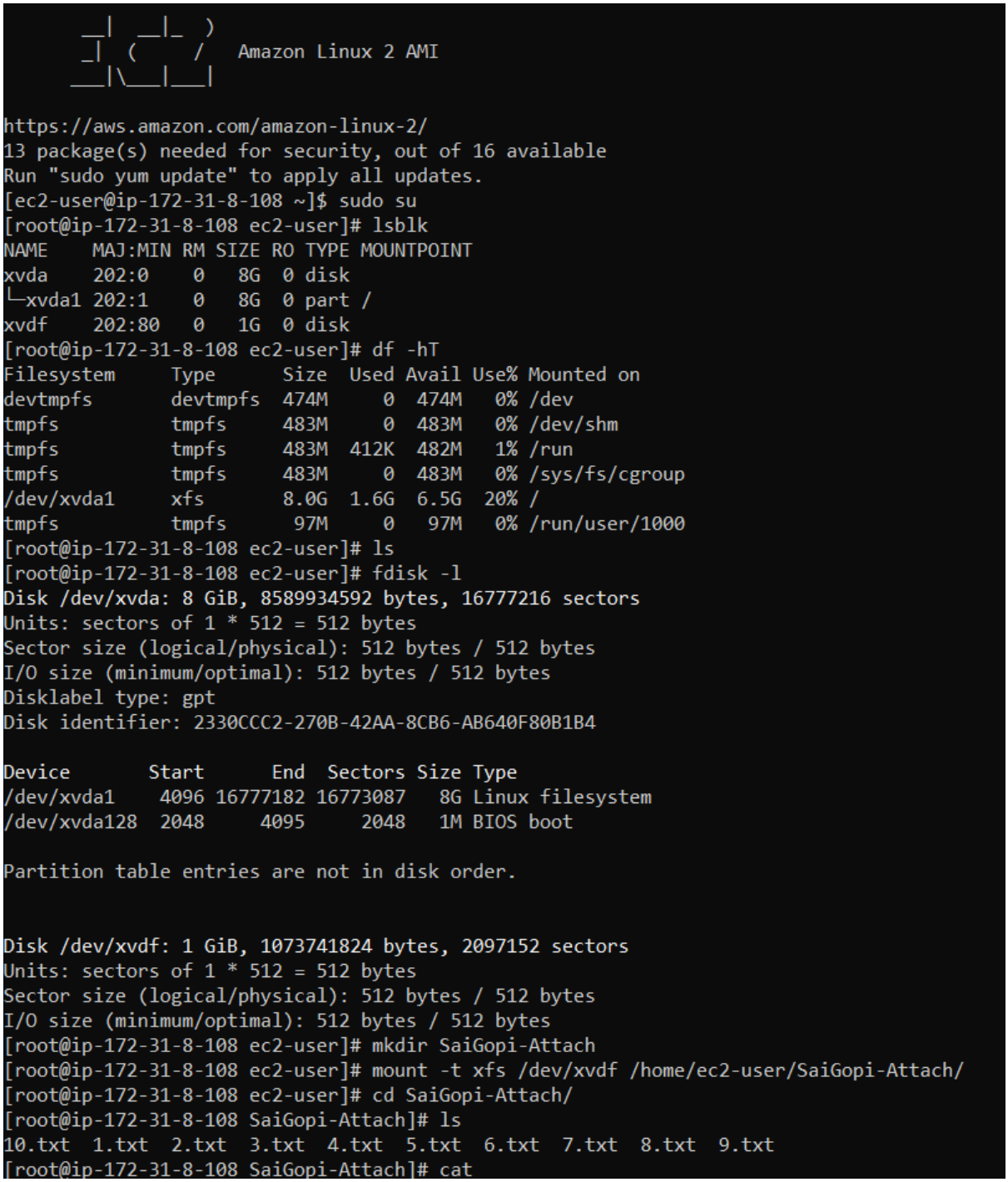


Fig8:SaiGopi-MachineB EBS

3.Snapshot

Steps:

1. Under EC2 Elastic Block store click on Snapshot
2. Click on create snapshot
3. Under volume id select your volume (SaiGopi-EBS) in Sydney region
4. Under description enter name of snapshot
5. Now click on create snapshot

6. Now click on snapshots and you can able to see your created snapshot
7. Select your snapshot and click on actions and click on copy snapshot
8. In settings page of copy snapshot ,under Destination region select the region where you want to create Tokyo (ap-northeast1)
9. Now click on copy snapshot

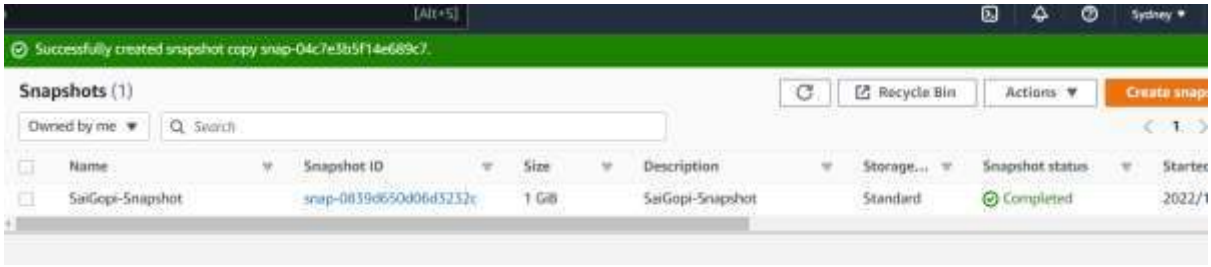


Fig 9: Snaphot created in Tokyo region from SaiGopi-EBS volume

Fig 10: volume created from copy snapshot in Tokyo region

10.Create a SaiGopi-Machine C in Tokyo region and attachthe EBS volume created from Snapshot copy

11.Now connect to SaiGopi-Machine C and create a new storage directory named SaiGopi-SnaphotVolume and mount it.

12.switch to the SaiGopi-SnaphotVolume directory and check the list of files in it.

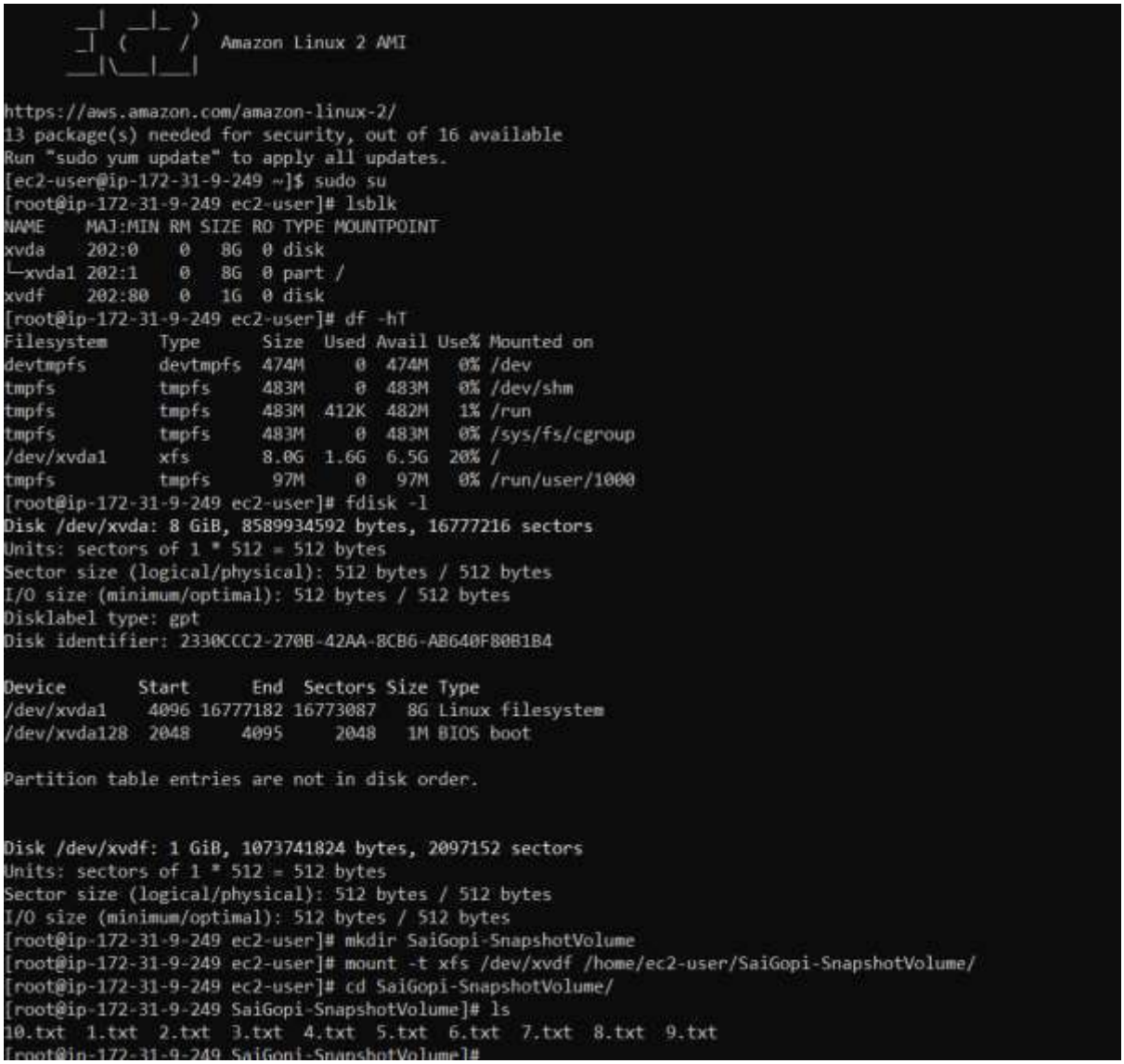


Fig 11: SaiGopi-Machine C SnapshotVolume

4.AMI

An Amazon Machine Image (AMI) is a template that contains a software configuration (for example, an operating system, an application server, and applications). From an AMI, you launch an instance, which is a copy of the AMI running as a virtual server in the cloud.

Steps:

- Created an SaiGopi-Machine1 Instance and in the security-groups add inbound rule http port 80 for this machine.
- Connect to the above instance and perform the below commands

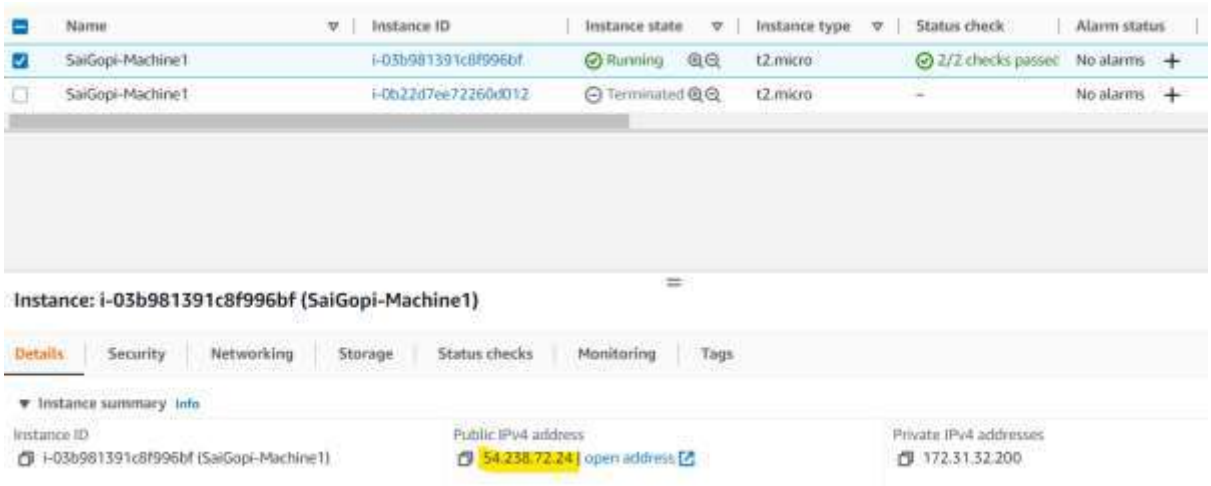


Fig12: SaiGopi-Machine1

- Preparing your Ubuntu server
    1. sudo apt update
    2. sudo ufw allow ssh
    3. sudo ufw allow 80
    4. sudo ufw allow 443
    5. sudo ufw enable

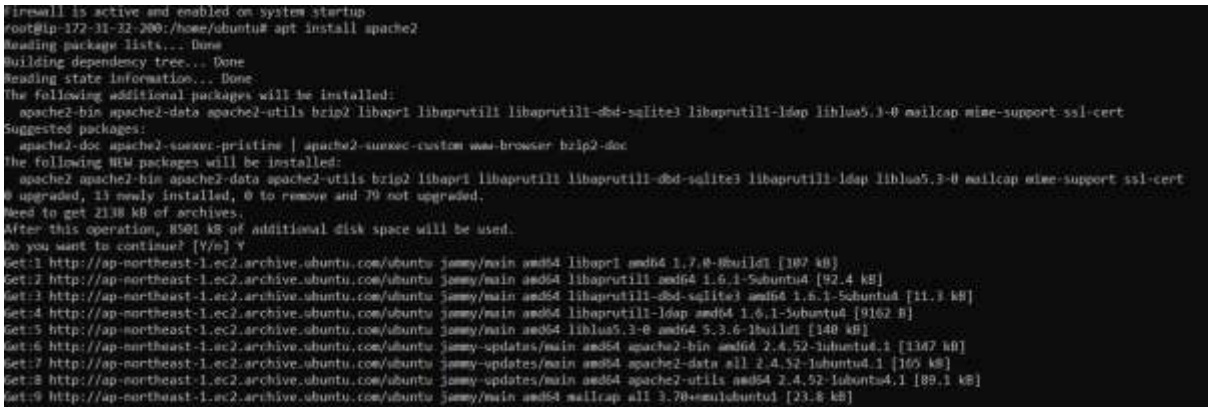


Fig13: Preparing your Ubuntu server

- Installing and testing Apache2
    1. sudo apt install apache2
    2. sudo systemctl status apache2
    3. http://YOURSERVERIPADDRESS/

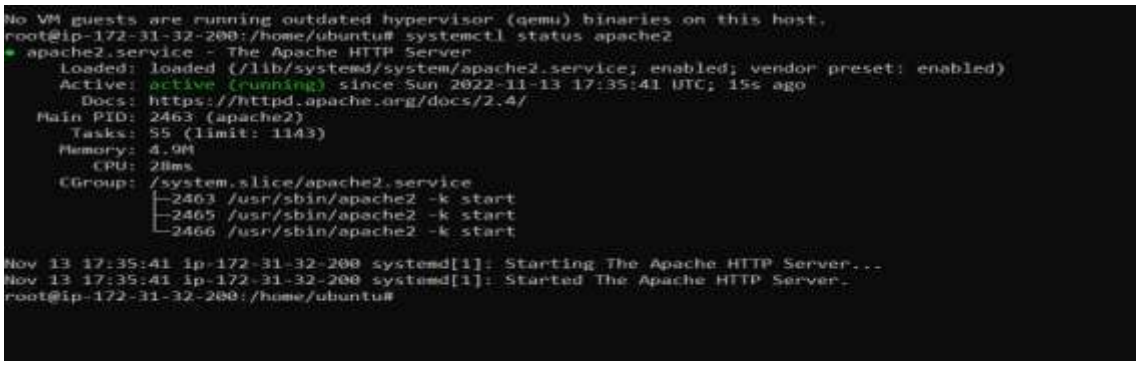


Fig 14: Testing apache2

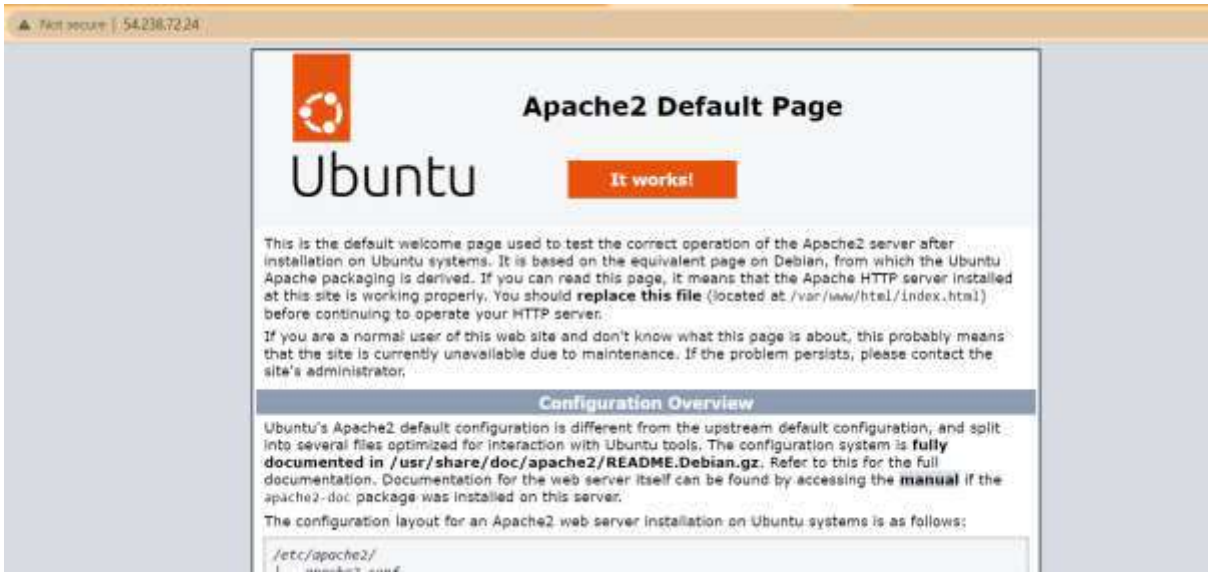


Fig 15: Testing apache2 on browser

- Installing and testing PHP
    1. sudo apt install php8.1
    2. php --version
    3. sudo systemctl restart apache2
    4. echo '<?php phpinfo(); ?>' | sudo tee -a /var/www/html/phpinfo.php > /dev/null
    5. http://YOURSERVERIPADDRESS/phpinfo.php

Fig 16: Testing PHP on browser

- Now create AMI from existing instance
- From this AMI create a new instance and in the security-group add inbound rule for http port 80
- Connect to this instance and try to access the ubuntu and php on browser

```
C:\Users\Sai Gopi\Downloads>ssh -i "TOKYO.pem" root@ec2-18-176-55-205.ap-northeast-1.compute.amazonaws.com
The authenticity of host 'ec2-18-176-55-205.ap-northeast-1.compute.amazonaws.com (18.176.55.205)' can't be established.
ED25519 key fingerprint is SHA256:oryzIHyJXsD6kCy2RyngH2uEapj5R/kq34hFYFz4bIM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-18-176-55-205.ap-northeast-1.compute.amazonaws.com' (ED25519) to the list of known hosts.
Please login as the user "ubuntu" rather than the user "root".


ssh -i "TOKYO.pem" root@ec2-18-176-55-205.ap-northeast-1.compute.amazonaws.comConnection to ec2-18-176-55-205.ap-northeast-1.comp

C:\Users\Sai Gopi\Downloads>ssh -i "TOKYO.pem" ubuntu@ec2-18-176-55-205.ap-northeast-1.compute.amazonaws.com
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1019-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Nov 13 10:57:58 UTC 2022

  System load:  0.3046875         Processes:             112
  Usage of /:   22.4% of 7.57GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for eth0: 172.31.32.196
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

84 updates can be applied immediately.
54 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Sun Nov 13 17:33:41 2022 from 175.101.107.225
ubuntu@ip-172-31-32-196:~$ sudo su
root@ip-172-31-32-196:/home/ubuntu# cat
```

Fig 17: SaiGopi-Machine2

Fig 18: Testing ubuntu for SaiGopi-Machine2

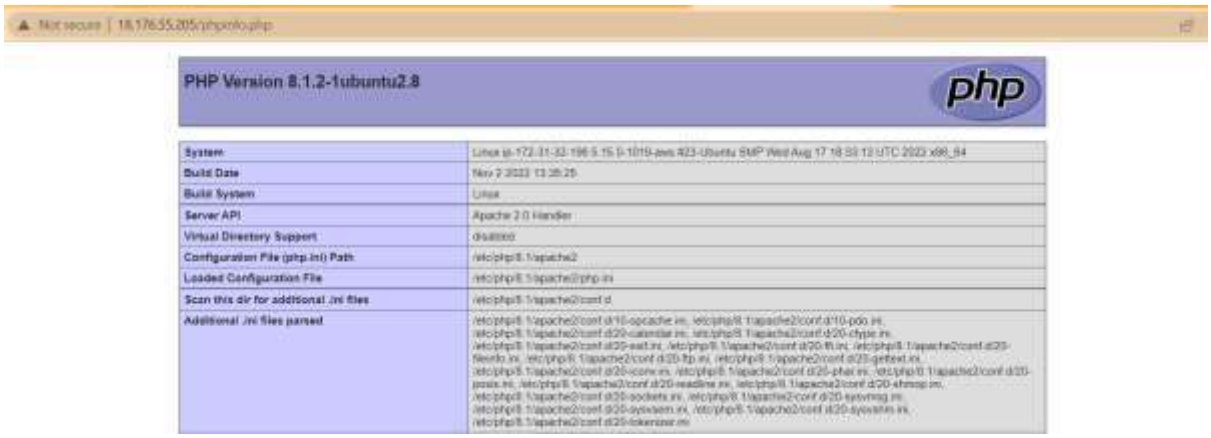


Fig 19: Testing ubuntu for SaiGopi-Machine2

5. Load Balancer

Steps:

- Create a EC2 machine (SaiGopi-A) and add security group with inbound rule allowing SSH and HTTP port.
- Prepare your ubuntu server and install and test apache2
- Install and test PHP8.1
- Create an AMI and create two instances from AMI with security group allowing inbound rule for SSH and HTTP port.

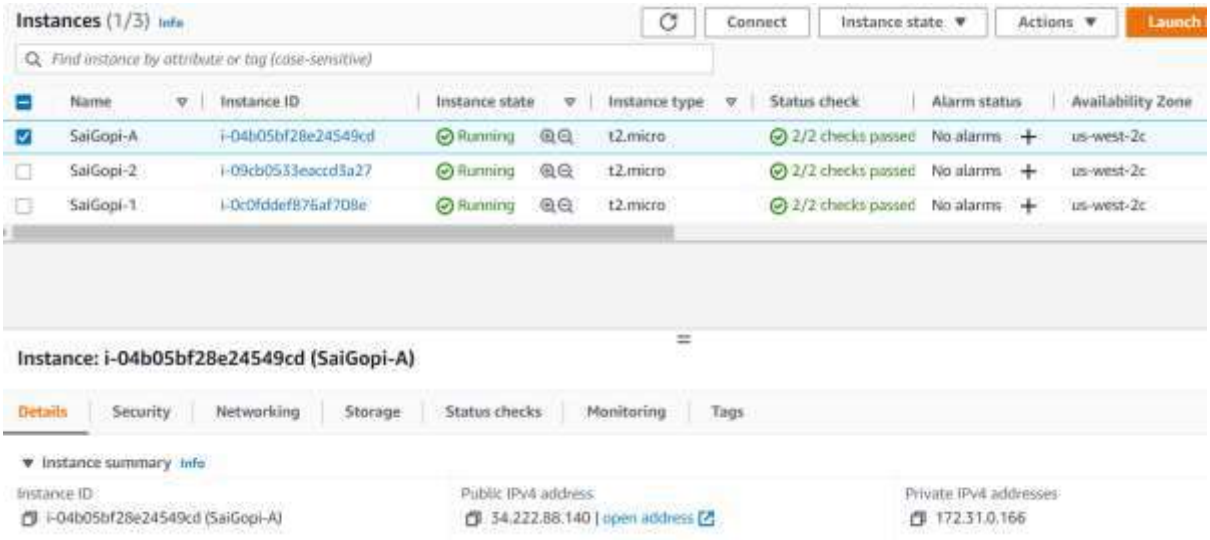


Fig20: SaiGopi-A

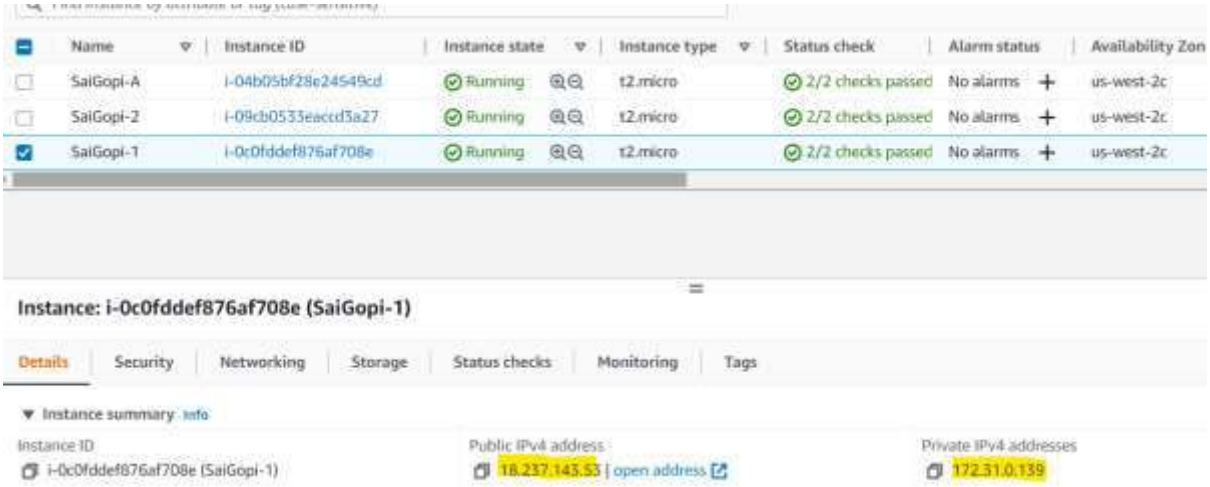


Fig21: SaiGopi-1

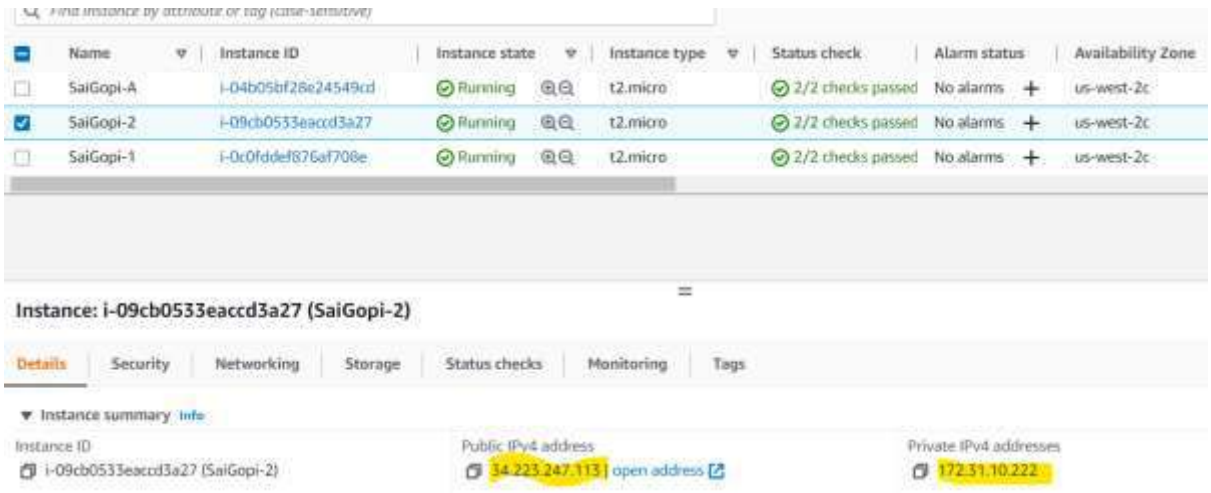


Fig22: SaiGopi-22

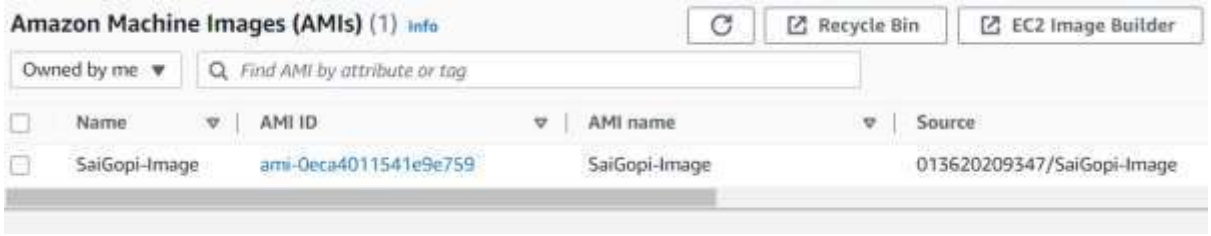


Fig23: AMI

- Under Load balancing from EC2 service click on Load Balancer and click on create a load balancer.
- Click on create Application load balancer and Give name to your load balancer (SaiGopi-APLB) and select all mappings under Network Mapping.
- Under security groups create a new security group allowing inbound rules for SSH and HTTP port.
- Under Listeners and routing, need to create a new target group (SaiGopi-APLBTG) and include your target machines under it.
- Now connect your Target Group to your Load balancer and click on create.

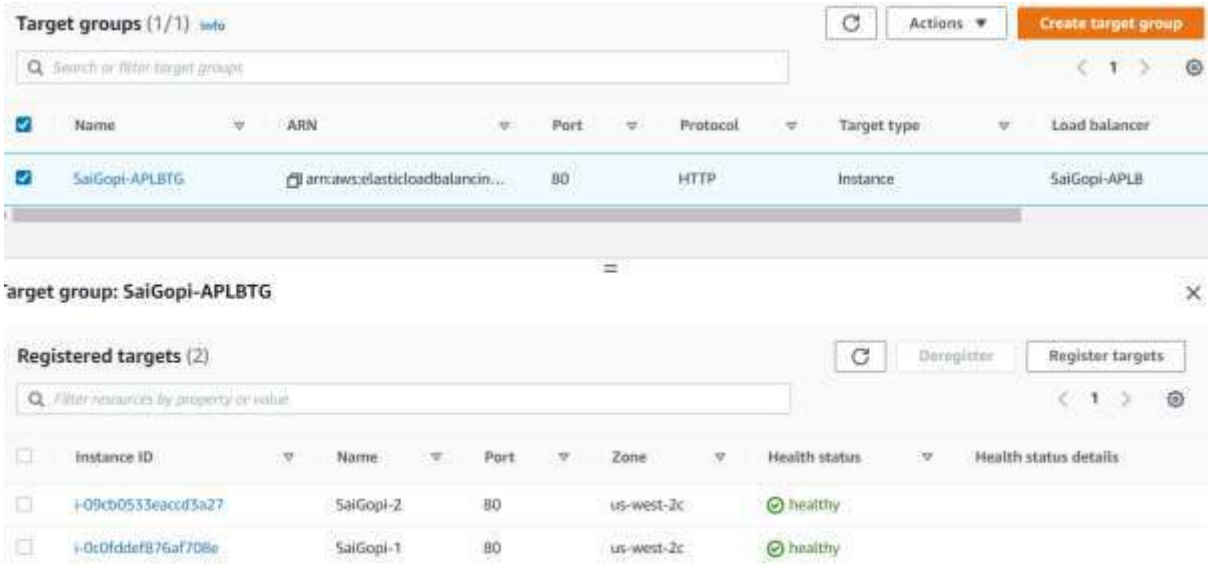


Fig24: Target Group

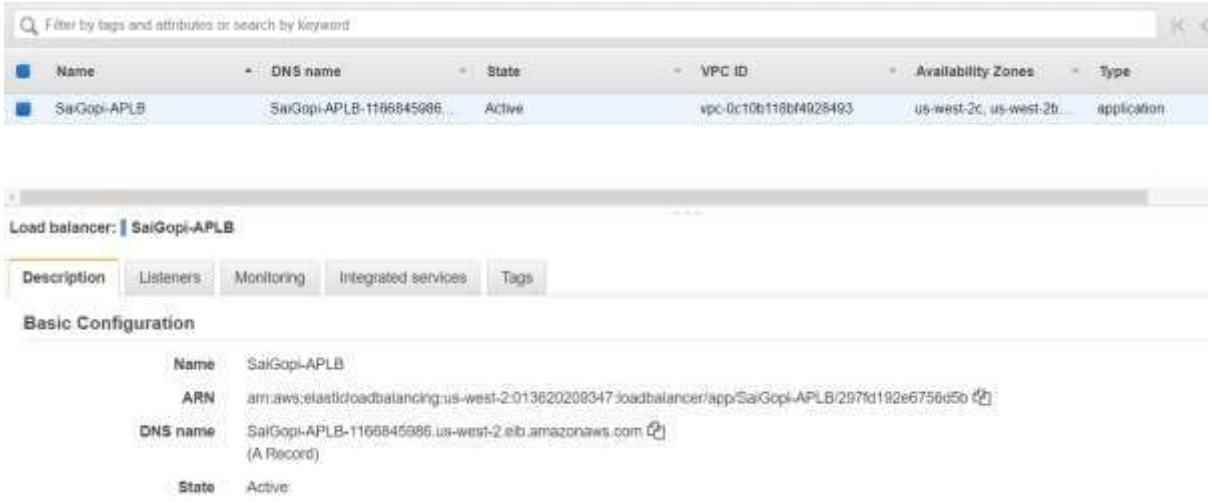


Fig25: Load Balancer

- Now connect to your Load balancer by copying the DNS name and pasting in the browser.
- You can also check to which machine it is being connected using DNS name/phpinfo.php and you can check the ip address of your machine to which it is being connected.

Fig26: Connecting to Load balancer

Fig 27: Connecting to Machine1 using LB

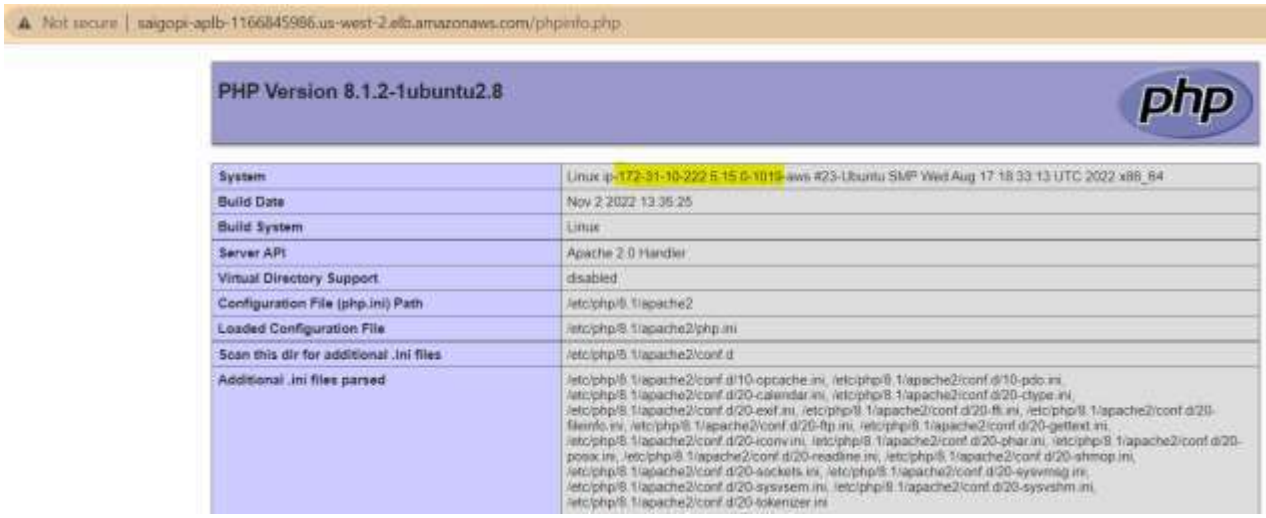


Fig 28: Connecting to Machine2 using LB

6. VPC with 2 public subnets & 2 private subnet having Internet gateway and NAT gateway

Steps:

- Create a VPC with 192.168.0.0/16 range.
- Create an Internet Gateway and attach it to the VPC created

Name	VPC ID	State	IPv4 CIDR
SaiGopi-VPC A	vpc-0d69682daf9b795a0	Available	192.168.0.0/16
–	vpc-0ebc6663e8377aeb3	Available	172.31.0.0/16

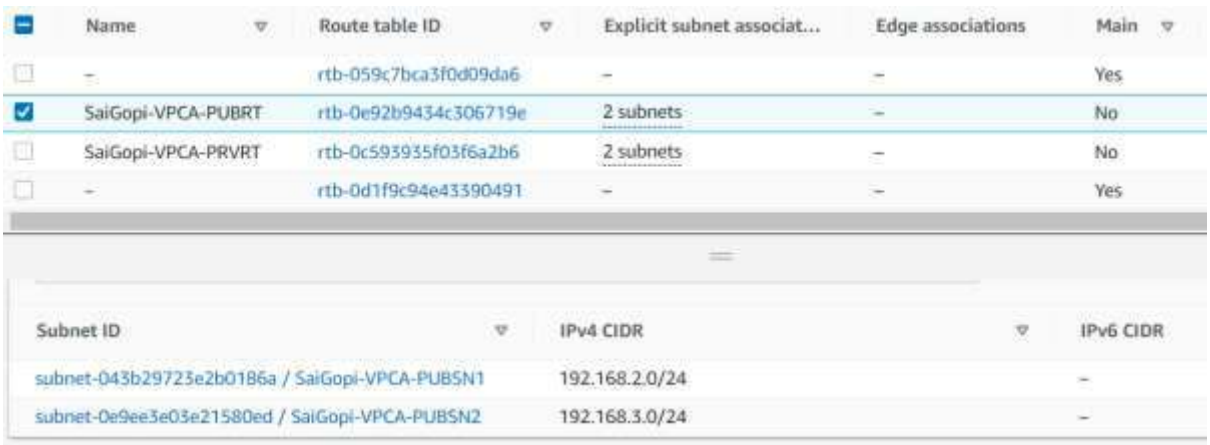
Fig 29: VPC

- Create Two private and two public subnets with the range:
    1. SaiGopi-VPCA- PRVSN1 >> 192.168.0.0/24
    2. SaiGopi-VPCA- PRVSN2 >> 192.168.1.0/24
    3. SaiGopi-VPCA- PUBSN1>> 192.168.2.0/24
    4. SaiGopi-VPCA- PUBSN2 >> 192.168.3.0/24

Name	Subnet ID	State	VPC	IPv4 CIDR
SaiGopi-VPCA-PUBSN1	subnet-043b29723e2b0186a	Available	vpc-0d69682daf9b795a0 | Sai...	192.168.2.0/24
SaiGopi-VPCA-PUBSN2	subnet-0e9ee3e03e21580ed	Available	vpc-0d69682daf9b795a0 | Sai...	192.168.3.0/24
–	subnet-0426a3f3a7ede3100	Available	vpc-0ebc6663e8377aeb3	172.31.16.0/20
SaiGopi-VPCA-PRVSN1	subnet-0329b24ec6e173b7b	Available	vpc-0d69682daf9b795a0 | Sai...	192.168.0.0/24
–	subnet-00549a52040bbed36	Available	vpc-0ebc6663e8377aeb3	172.31.0.0/20
SaiGopi-VPCA-PRVSN2	subnet-007b8487b82f840ce	Available	vpc-0d69682daf9b795a0 | Sai...	192.168.1.0/24
–	subnet-06444667c2ef027e4	Available	vpc-0ebc6663e8377aeb3	172.31.32.0/20

Fig 30: Subnets

- Create Two route tables i.e one public and one private
    1. SaiGopi-VPCA-PUBRT
    2. SaiGopi-VPCA-PRVRT



Name	Route table ID	Explicit subnet associat...	Edge associations	Main
–	rtb-059c7bca3f0d09da6	–	–	Yes
SaiGopi-VPCA-PUBRT	rtb-0e92b9434c306719e	2 subnets	–	No
SaiGopi-VPCA-PRVRT	rtb-0c593935f03f6a2b6	2 subnets	–	No
–	rtb-0d1f9c94e43390491	–	–	Yes

Subnet ID	IPv4 CIDR	IPv6 CIDR
subnet-043b29723e2b0186a / SaiGopi-VPCA-PUBSN1	192.168.2.0/24	–
subnet-0e9ee3e03e21580ed / SaiGopi-VPCA-PUBSN2	192.168.3.0/24	–

Fig 31: Public Route Table

- Associate the private route table to the existing private subnets
- Associate the public route table to the existing public subnets and the Internet Gateway

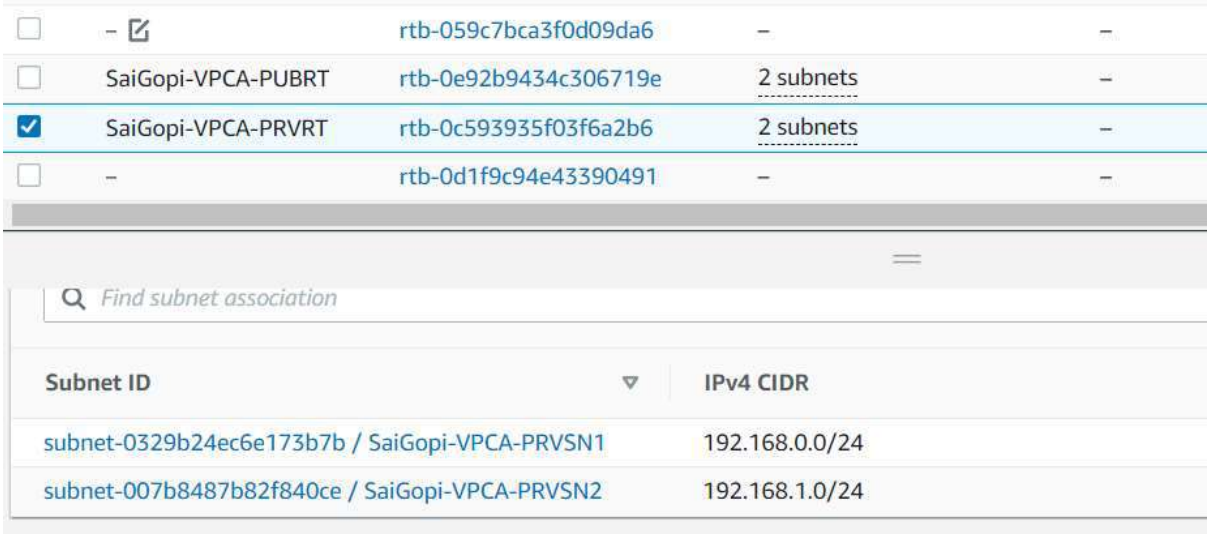


Fig 32: Private Route Table

- Create a New Security Group and add rules in the Inbound Traffic allowing SSH, All ICMP IPV4 and All TCP types .
- Now create Two EC2 instances with the newly created security group and under network edit option choose the Newly created VPC and the corresponding subnets
    1. SaiGopi-PRVSN1
    2. SaiGopi-PUBSN1

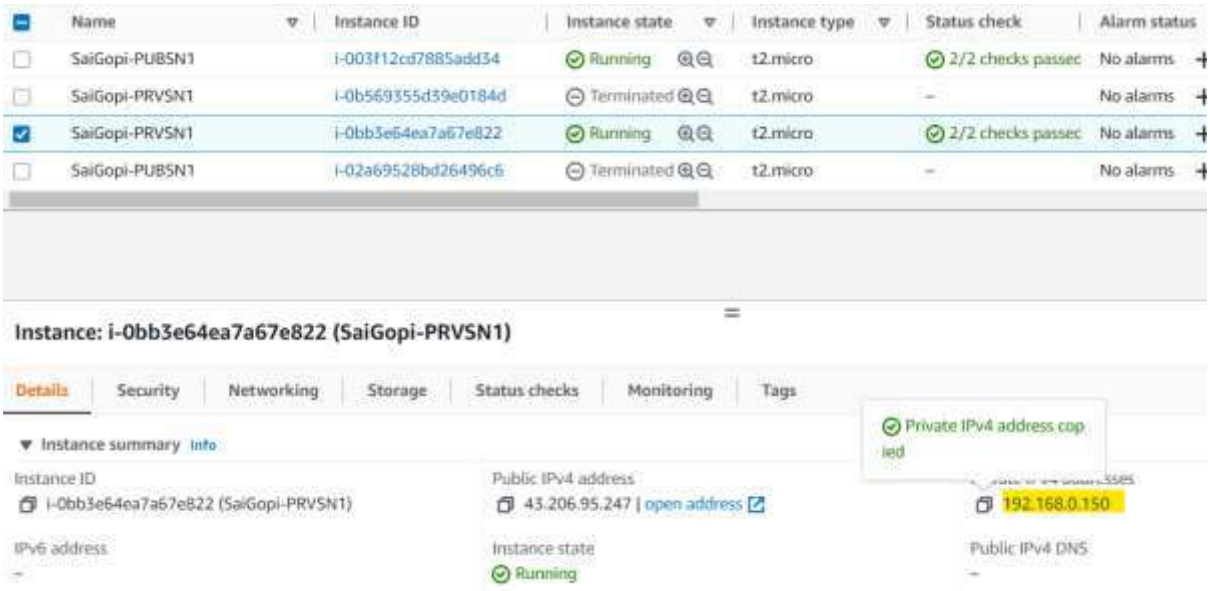


Fig 33: EC2 machines

- Now connect to the EC2 Public machine from SSH client and ping from it
- We can able to ping from this machine
- Now connect to the EC2 Private machine from SSH client and try to ping
- We can see we are not able to connect to the machine and also we are not able to ping from it.

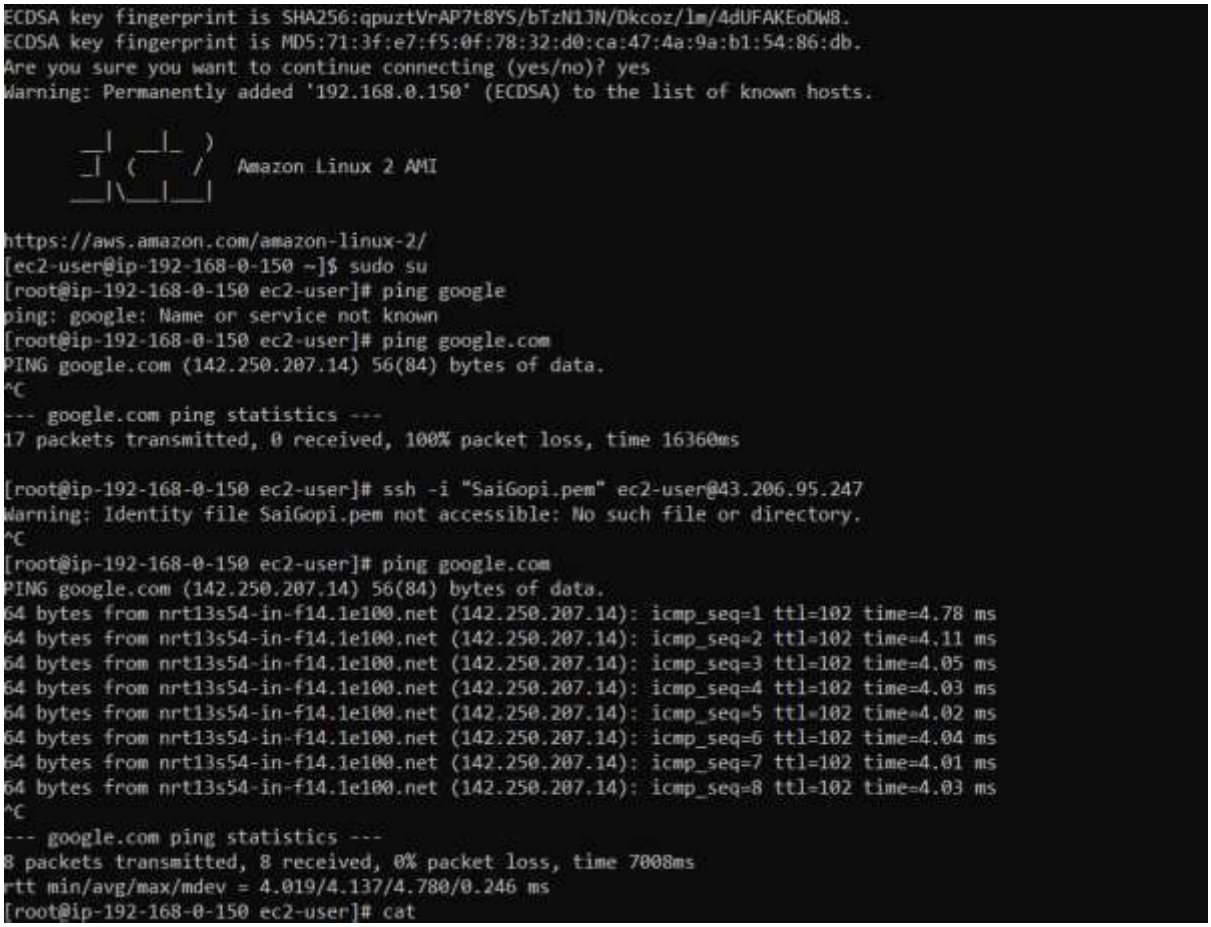


Fig 34: Connecting to Public machine

- So We try to ping the Private machine from public machine , which will give an error of key not exists.
- So we copy the pem file into the Public machine using secure copy
  Scp -i .\SaiGopi.pem -r .\SaiGopi.pem ec2-user@public machine private ip address :/present working directory of Public machine.
- Grant the permission to the pem file chmod 700
- Now you can able to connect the private machine from the public machine.
- Now you are able to connect to the private machine but still you are not able to ping since its not connected to any internet Gateway.
- So for this purpose we use Nat Gateway in order to provide internet for the private machines.
- Create a Nat Gateway and allocate an elastic Ip .
- Now go to the Private route table and under routes add the Nat Gateway and now connect to this machine and ping from it

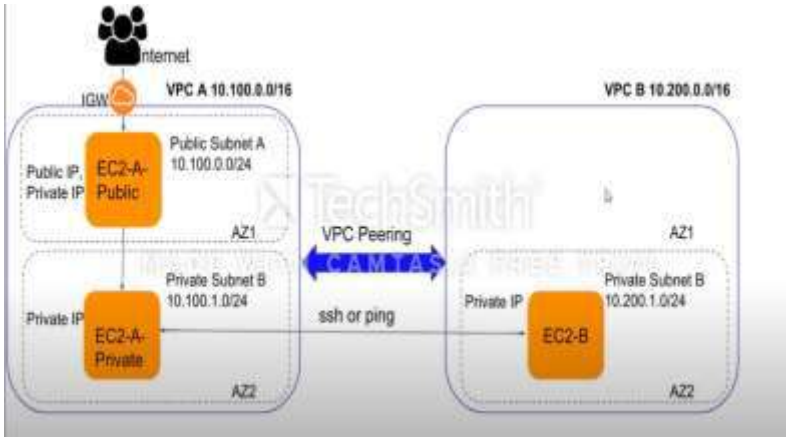
```
C:\Users\Sai Gopi\Downloads>ssh -i "SaiGopi.pem" ec2-user@35.77.9.139
The authenticity of host '35.77.9.139 (35.77.9.139)' can't be established.
ED25519 key fingerprint is SHA256:Z6JctaKbnoMbU8zSUSOi4sqSpjQTde1Dq7Jg1LL1fes.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '35.77.9.139' (ED25519) to the list of known hosts.

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

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 1 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-192-168-2-140 ~]$ sudo su
[root@ip-192-168-2-140 ec2-user]# ping 192.168.0.150
PING 192.168.0.150 (192.168.0.150) 56(84) bytes of data.
64 bytes from 192.168.0.150: icmp_seq=1 ttl=255 time=0.983 ms
64 bytes from 192.168.0.150: icmp_seq=2 ttl=255 time=0.511 ms
64 bytes from 192.168.0.150: icmp_seq=3 ttl=255 time=0.485 ms
64 bytes from 192.168.0.150: icmp_seq=4 ttl=255 time=0.567 ms
64 bytes from 192.168.0.150: icmp_seq=5 ttl=255 time=0.557 ms
64 bytes from 192.168.0.150: icmp_seq=6 ttl=255 time=0.561 ms
64 bytes from 192.168.0.150: icmp_seq=7 ttl=255 time=0.499 ms
64 bytes from 192.168.0.150: icmp_seq=8 ttl=255 time=0.537 ms
64 bytes from 192.168.0.150: icmp_seq=9 ttl=255 time=0.603 ms
64 bytes from 192.168.0.150: icmp_seq=10 ttl=255 time=0.506 ms
64 bytes from 192.168.0.150: icmp_seq=11 ttl=255 time=0.422 ms
64 bytes from 192.168.0.150: icmp_seq=12 ttl=255 time=0.532 ms
64 bytes from 192.168.0.150: icmp_seq=13 ttl=255 time=0.528 ms
64 bytes from 192.168.0.150: icmp_seq=14 ttl=255 time=0.525 ms
64 bytes from 192.168.0.150: icmp_seq=15 ttl=255 time=0.650 ms
64 bytes from 192.168.0.150: icmp_seq=16 ttl=255 time=0.505 ms
 64 bytes from 192.168.0.150: icmp_seq=17 ttl=255 time=0.524 ms
^C
--- 192.168.0.150 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 16348ms
rtt min/avg/max/mdev = 0.422/0.558/0.983/0.118 ms
```

Fig 35: Connecting to Private machine

7. VPC Peering



Steps:

1. Creation of two VPC's
    - VPC A >> 10.100.0.0/16
    - VPC B >> 10.200.0.0/16

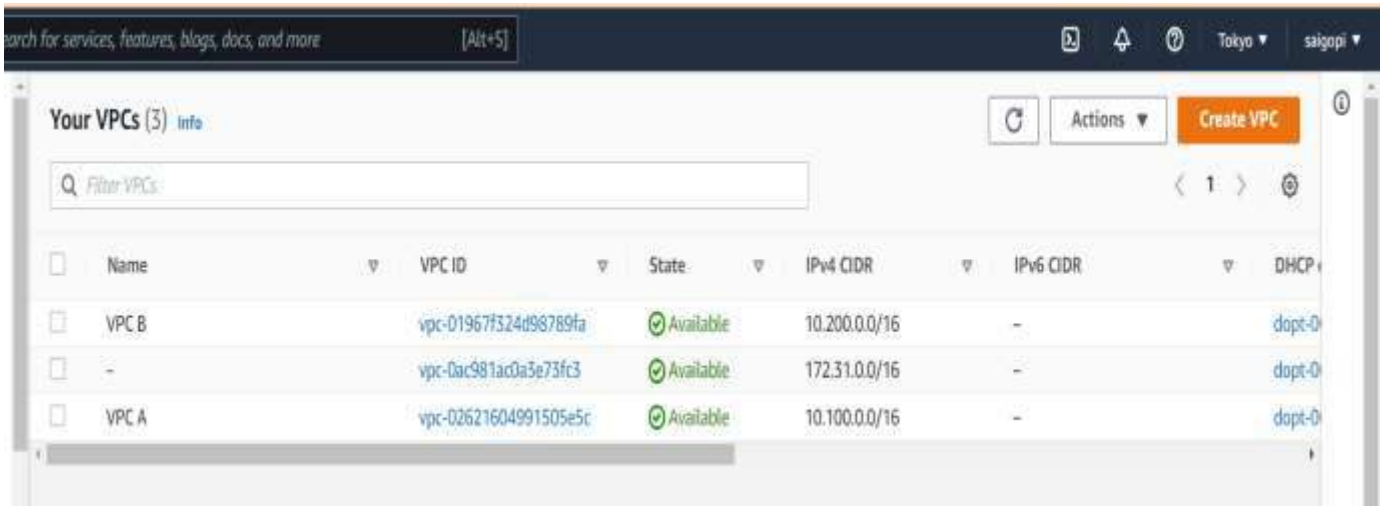


Fig 36: VPCs

2. Creation of Route Tables
    - VPC A-PUB RT
    - VPC A-PRV RT
    - VPC B PRV RT

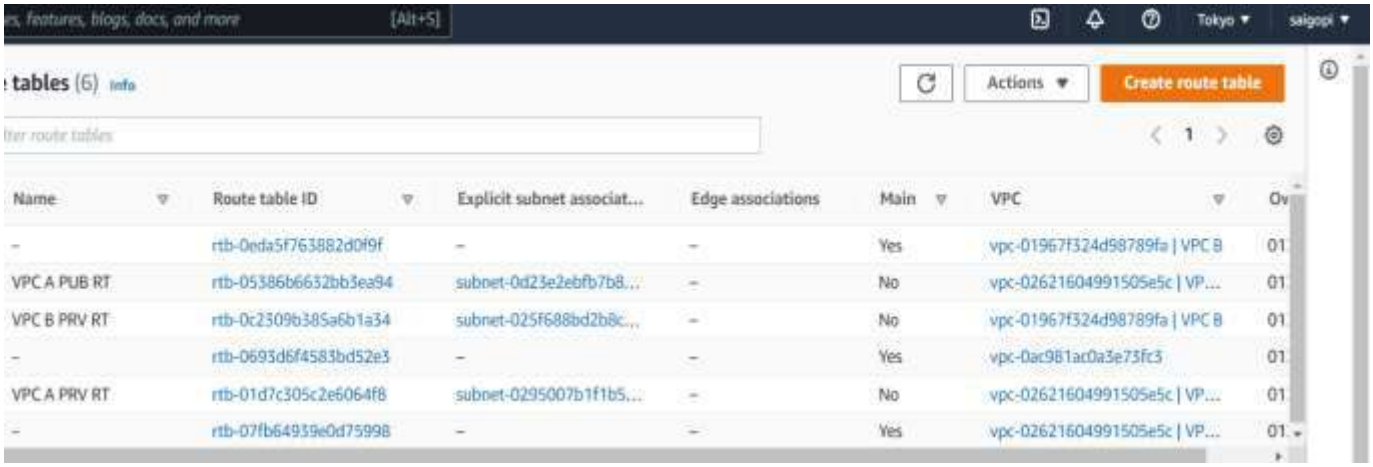


Fig 37: Route tables

3. Creation of Subnets
    - VPC A-PUBSN A >>10.100.0.0/24
    - VPC A-PRVSN B >>10.100.1.0/24
    - VPC B PRV B >>10.200.1.0/24

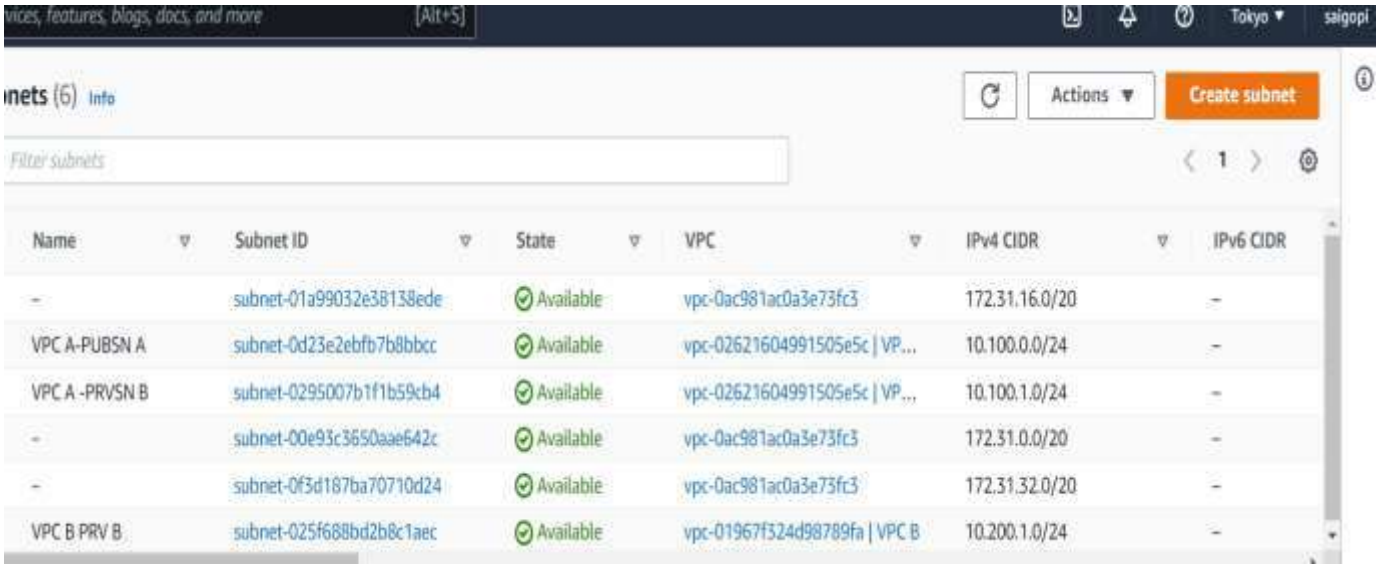


Fig 38: Subnets

4. Creation of Internet Gateway
    - VPC A IGW

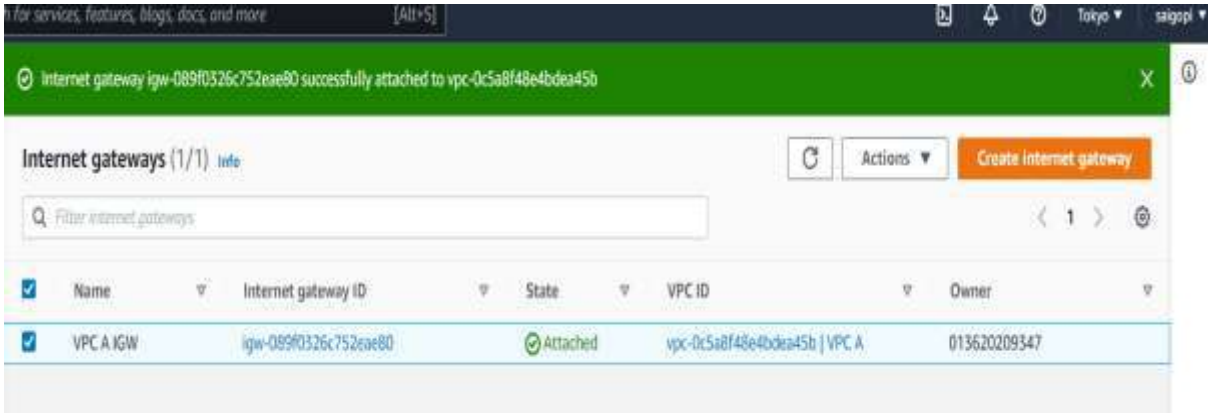


Fig 39: Internet Gateway

5. Attaching the subnets to corresponding route tables.
   - VPC A-PUBSN A to VPC A-PUB RT
   - VPC A-PRVSN B to VPC A-PRV RT
   - VPC B PRV B to VPC B PRV RT
6. Attaching Internet gateway to VPC
   - VPC A IGW to VPC A
7. Attaching route table to internet gateway
   - VPC A PUB RT to VPC A IGW
8. Creation of EC2 instances
   - OPEN EC2 instances and click on launch instance
   - Give EC2 instance a name and create a new key pair and select t2.micro instance type.
   - Under Network settings select the related VPC ,subnet and enable the public IP
   - Create a new security group and later edit the inbound and outbound rules to enable all IPV4 addresses.
   - And click on create instance.
   - ➢ EC2 VPCA-PUB
   - ➢ EC2 VPCA-PRV
   - ➢ EC2 VPCB-PRV

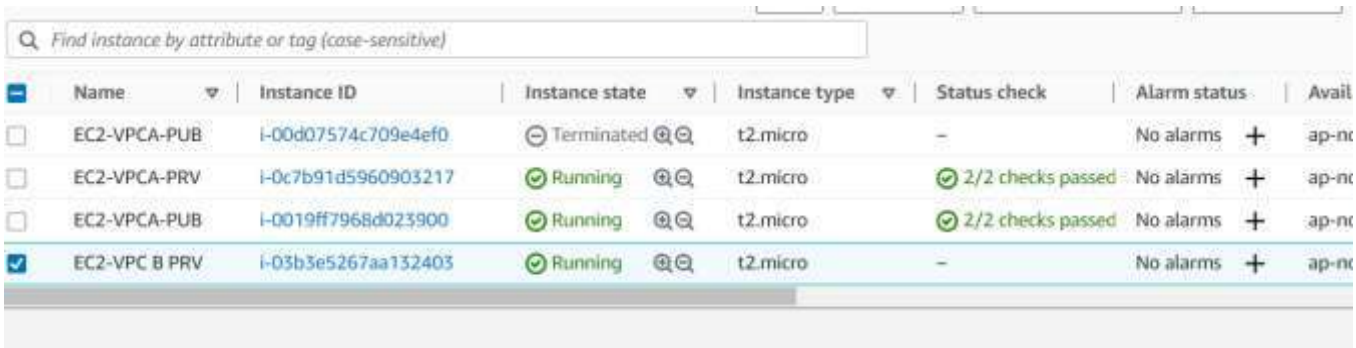


Fig 40: EC2 machines

9. Connecting to first EC2 machine >> EC2 VPCA-PUB

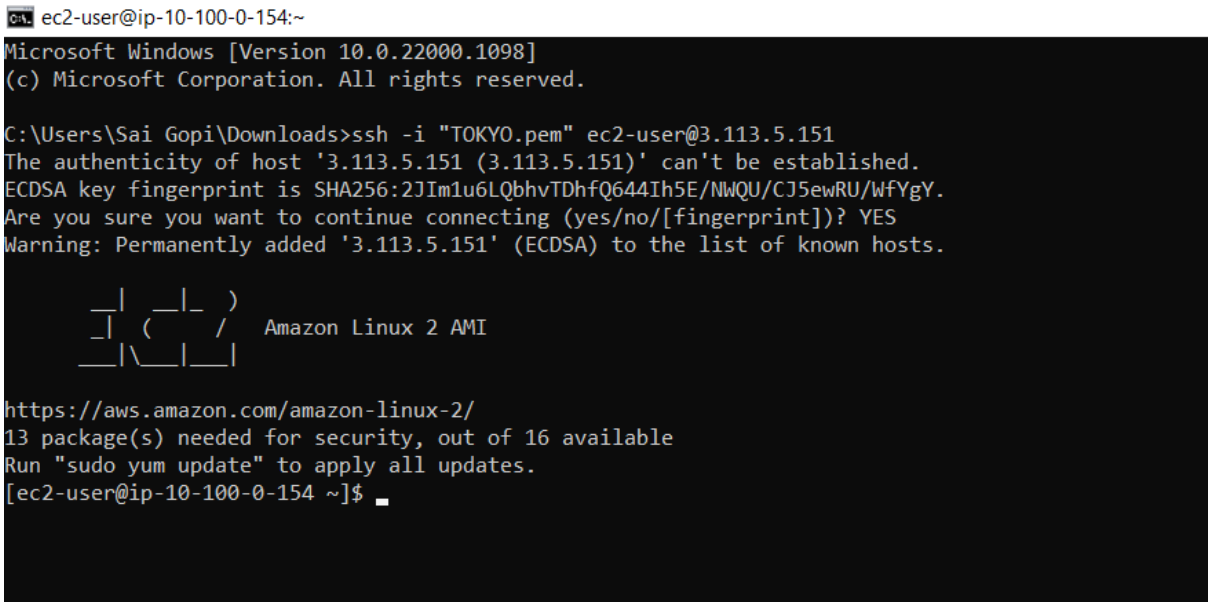


Fig 41: connecting to Public machine

10. Now trying to ping the EC2 VPCA-PRV machine from EC2 VPCA-PUB machine by using private IP of EC2 VPCA-PRV machine

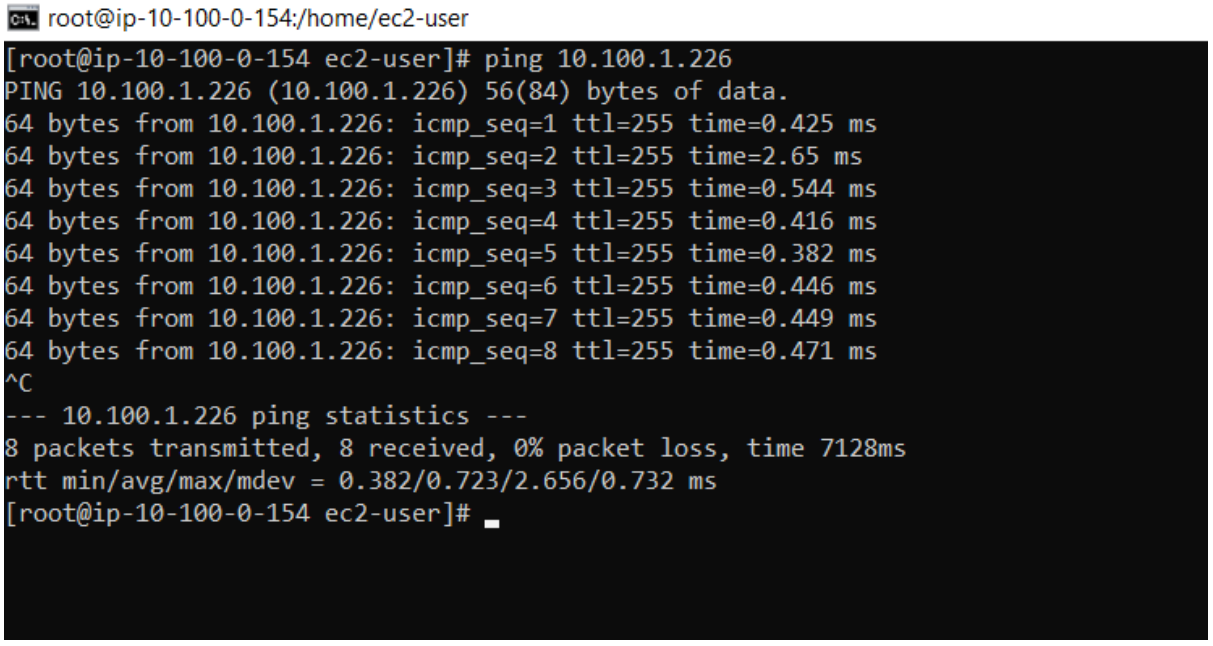


Fig 42: connecting to private machine using VPC peering

11. I am unable to ssh the machine as it requires the .pem file .so copying the .pem file to EC2 VPCA-PUB 10-100-0-151 ec2user.
    - scp -i .\TOKYO.pem -r .\TOKYO.pem ec2-user@publicmachine ip address :/Present working direcorty.
    - Connect to the machine and ping from it.

```
64 bytes from 10.100.1.226: icmp_seq=3 ttl=255 time=0.589 ms
64 bytes from 10.100.1.226: icmp_seq=4 ttl=255 time=0.532 ms
^C
--- 10.100.1.226 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3049ms
rtt min/avg/max/mdev = 0.392/0.500/0.589/0.072 ms
[root@ip-10-100-0-154 ec2-user]# ssh -i "TOKYO.pem" ec2-user@54.178.163.32
^C
[root@ip-10-100-0-154 ec2-user]# ls -l
total 4
-rw-rw-r-- 1 ec2-user ec2-user 1674 Oct 27 18:26 TOKYO.pem
[root@ip-10-100-0-154 ec2-user]# chmod 700 TOKYO.pem
[root@ip-10-100-0-154 ec2-user]# ls-l
bash: ls-l: command not found
[root@ip-10-100-0-154 ec2-user]# ls -l
total 4
-rwx------ 1 ec2-user ec2-user 1674 Oct 27 18:26 TOKYO.pem
[root@ip-10-100-0-154 ec2-user]# ^Ch -i "TOKYO.pem" ec2-user@54.178.163.32
[root@ip-10-100-0-154 ec2-user]# ssh -i TOKYO.pem ec2-user@10.100.1.226

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-100-1-226 ~]$
```

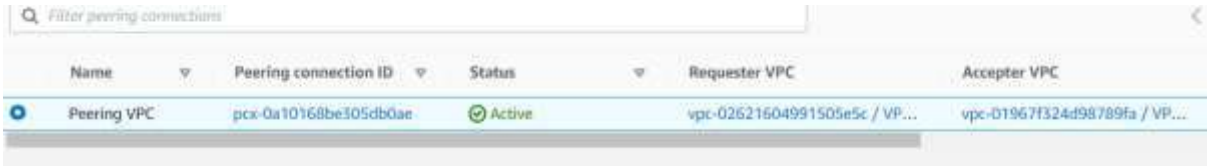


Fig 43: VPC Peering

12.Now connecting to the private EC2 machine from VPC peering

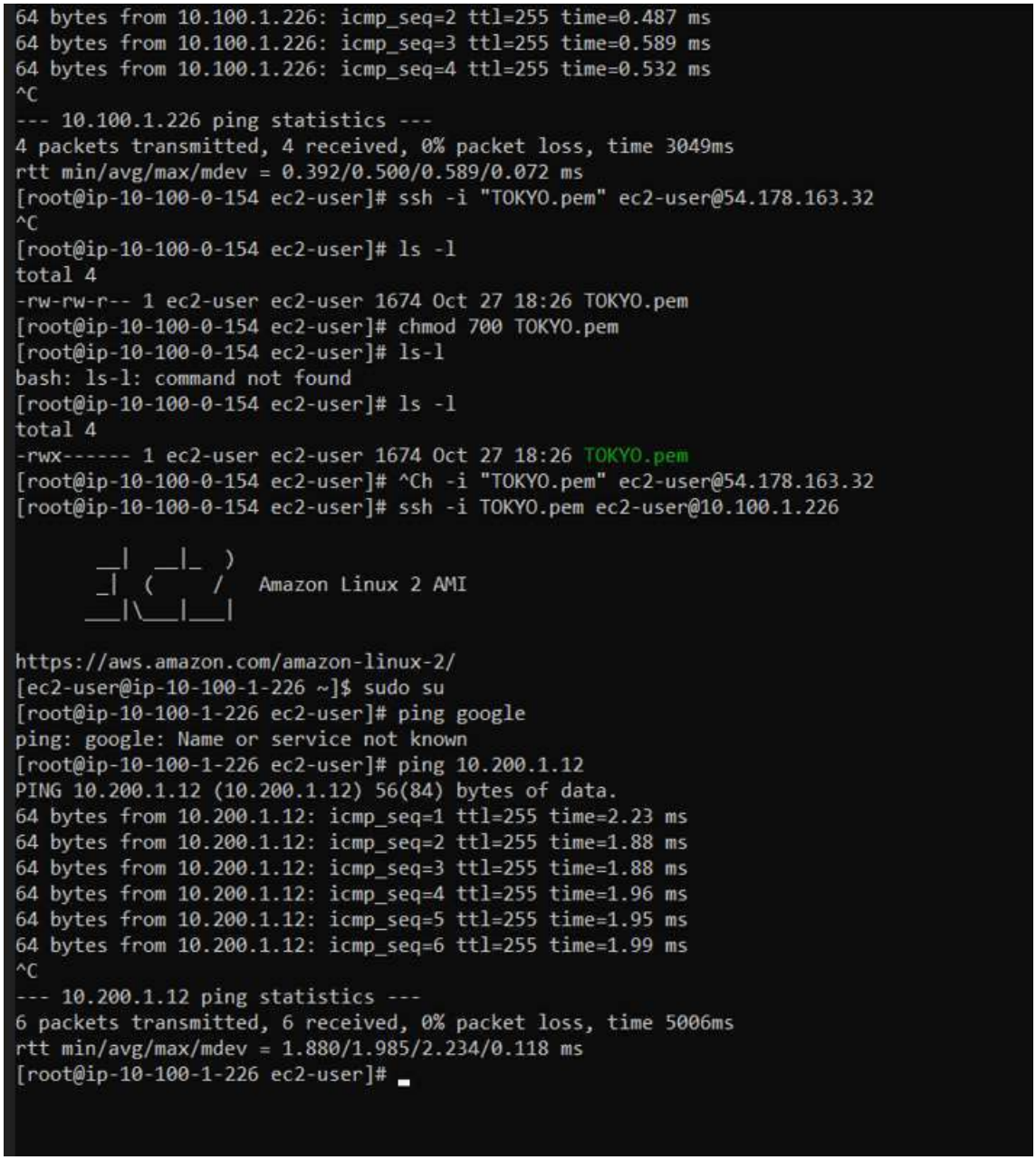


Fig 44: EC2 machine by VPC peering