CREATING THE LOAD BALANCER

A load balancer accepts incoming traffic from clients and routes requests to EC2 instances (Targets).

The load balancer also monitors the health of its registered targets and ensures that it routes traffic only to healthy targets.

When the load balancer detects an unhealthy target, it stops routing traffic to that target. It then resumes routing traffic to that target when it detects that the target is healthy again.

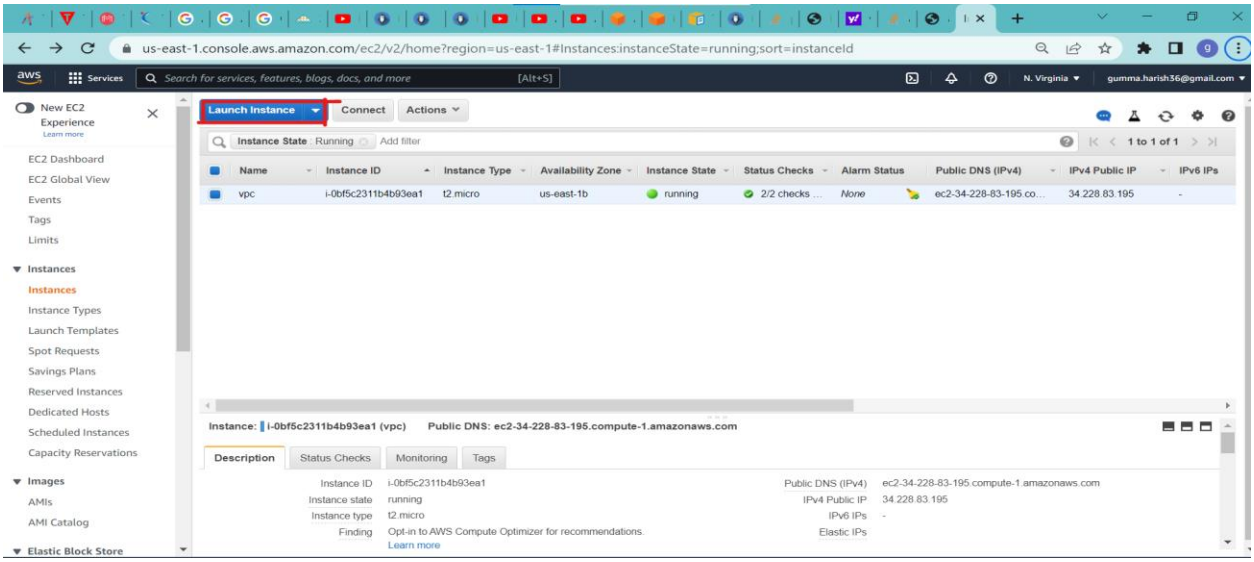
Step-1 Create Linux Machine

Launch instance --- Amazon Linux -- No of intances - 1 --- Name Tag- Lin-1 --- Security Group - LinSG09

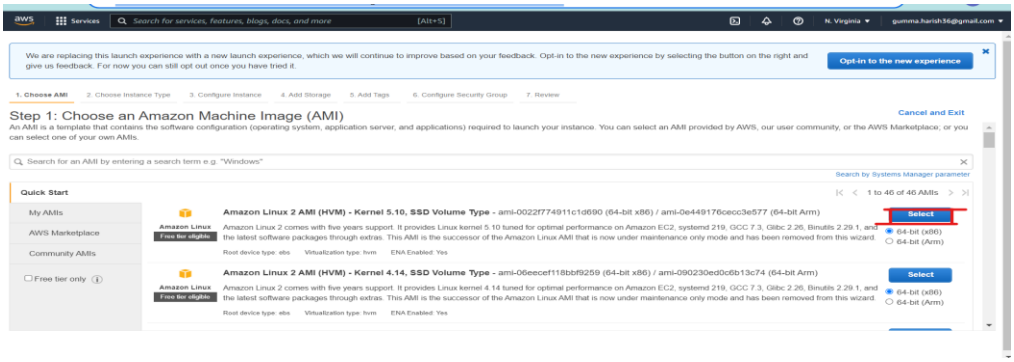
Description – LinSG

Add Rule

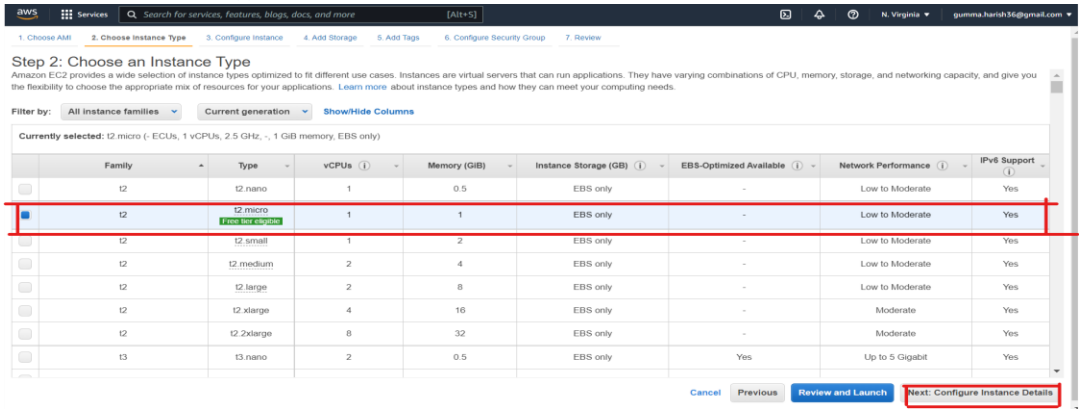
HTTP



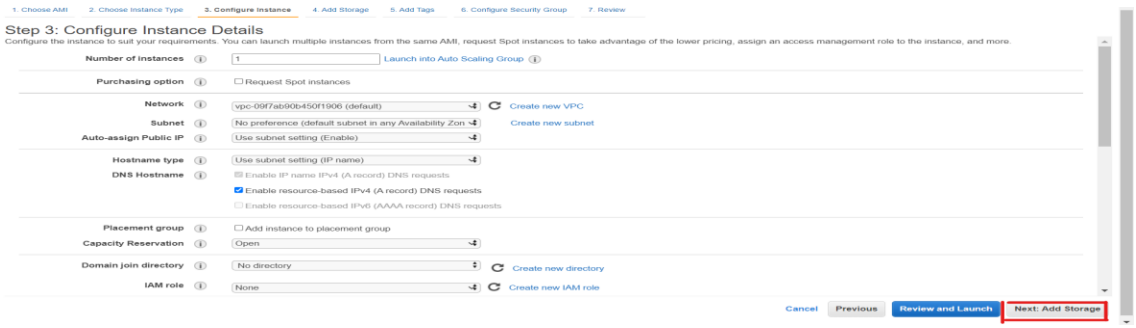
Choose machine image



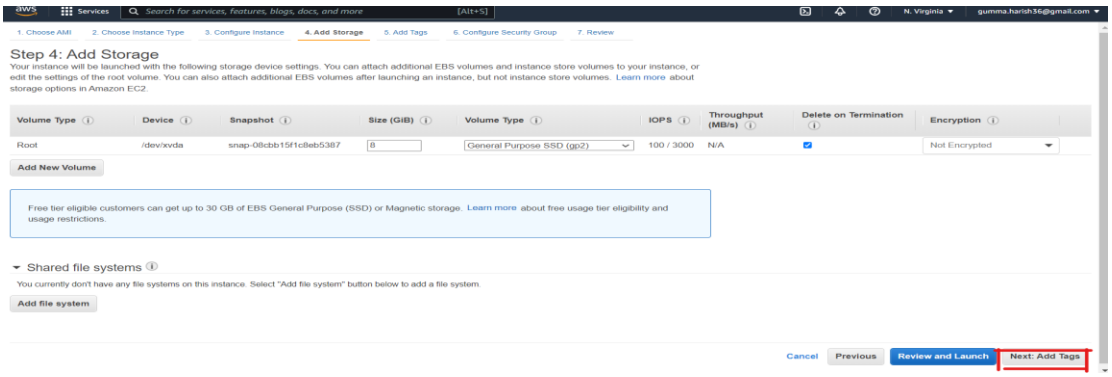
Choose an instance type



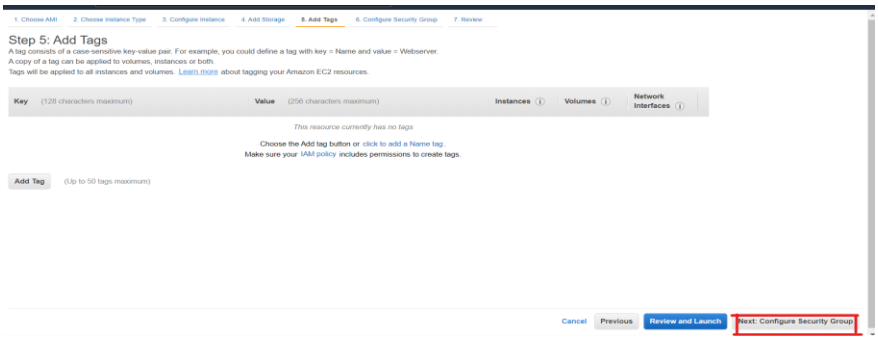
Configure instance details



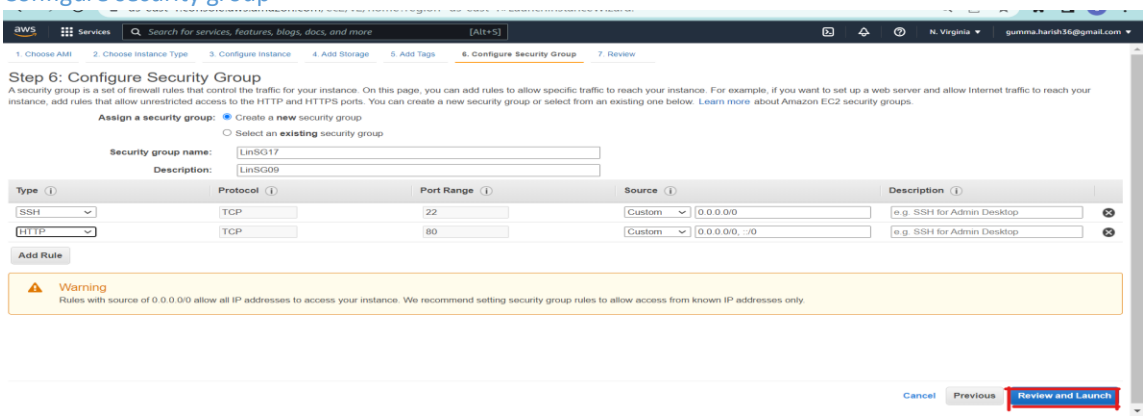
Add storage



No need to add the tags

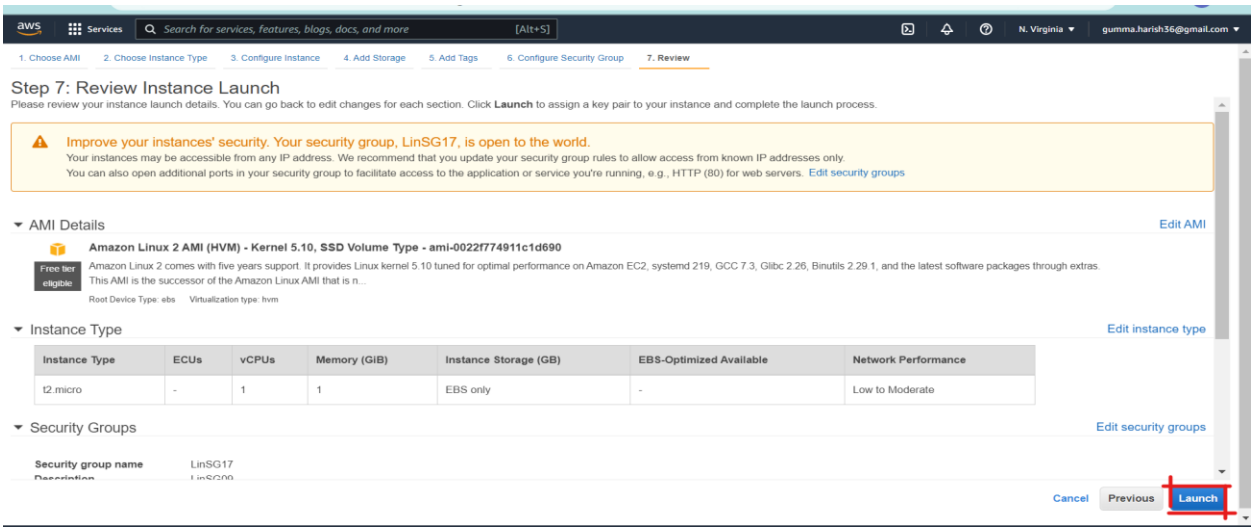


Configure security group

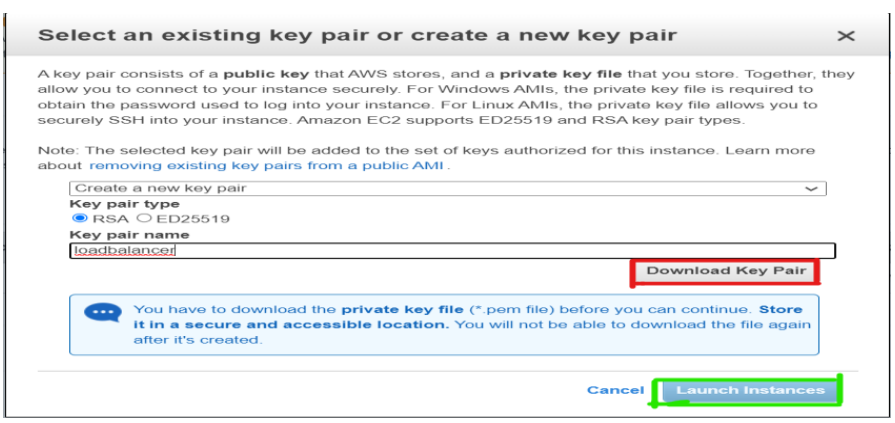


Add the HTTP port

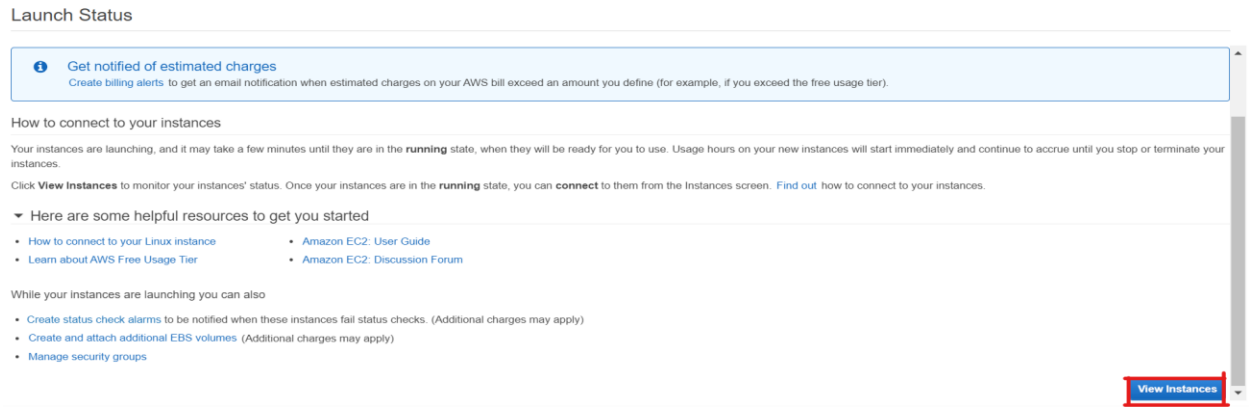
Review instance launch



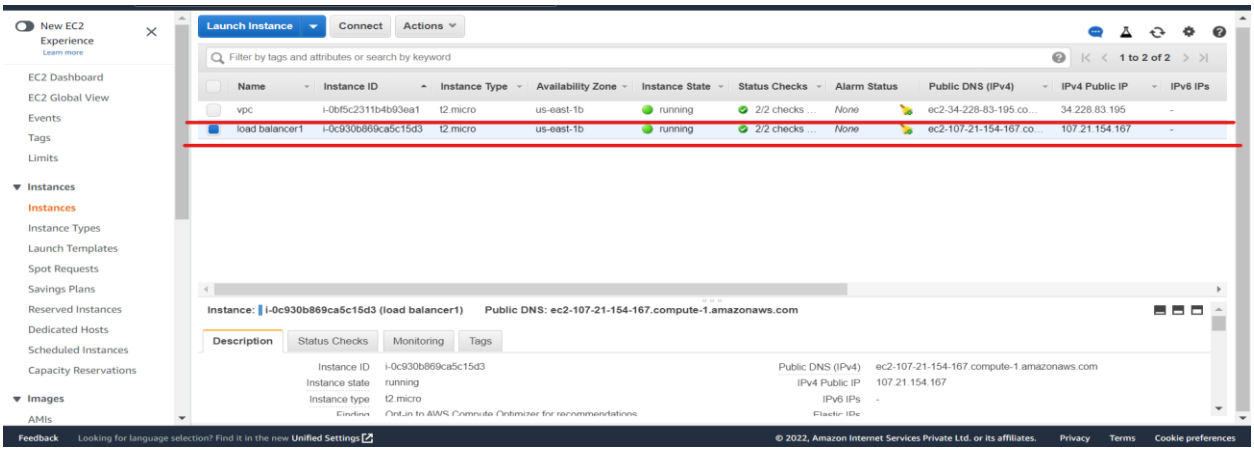
First create a new key pair and download the key pair .after downloading the key pair than we need to launch the instance



Launch status



Now new instance had launched (with the name of load balancer1)

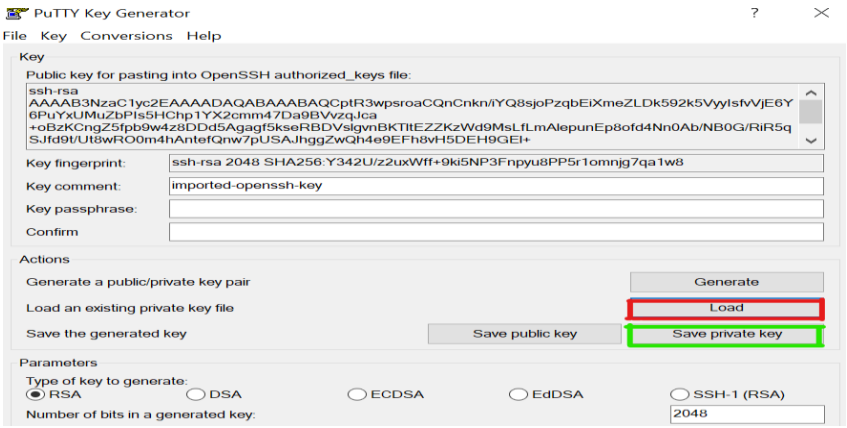


Step-2

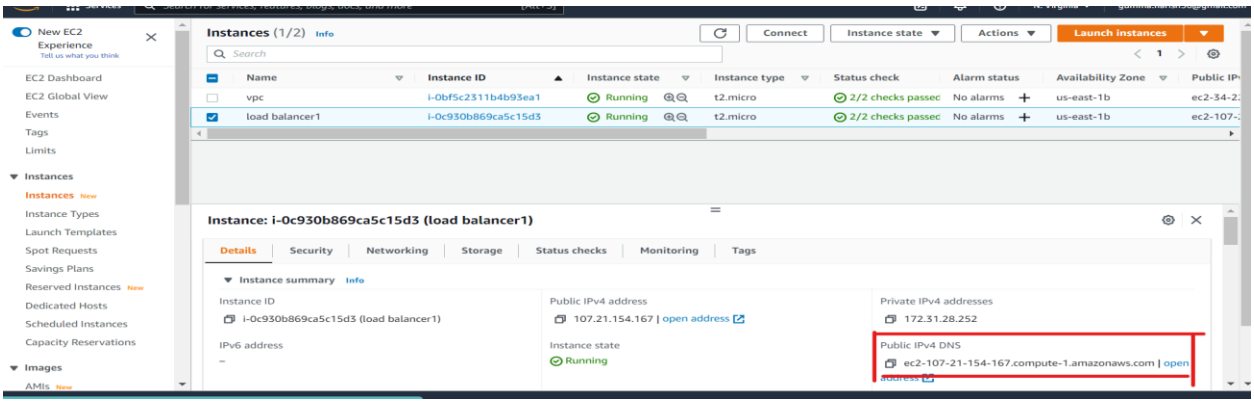
Convert pem into ppk file

Using the puttygen we can convert the pem file into ppk file

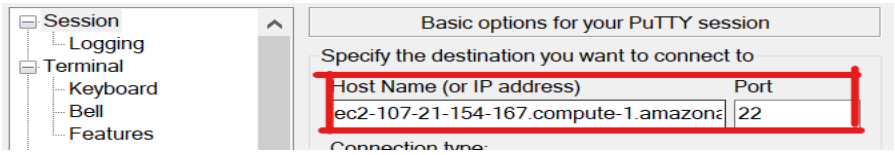
- First load the pem file into the putty gen
- After loading the file we need to save the ppk file



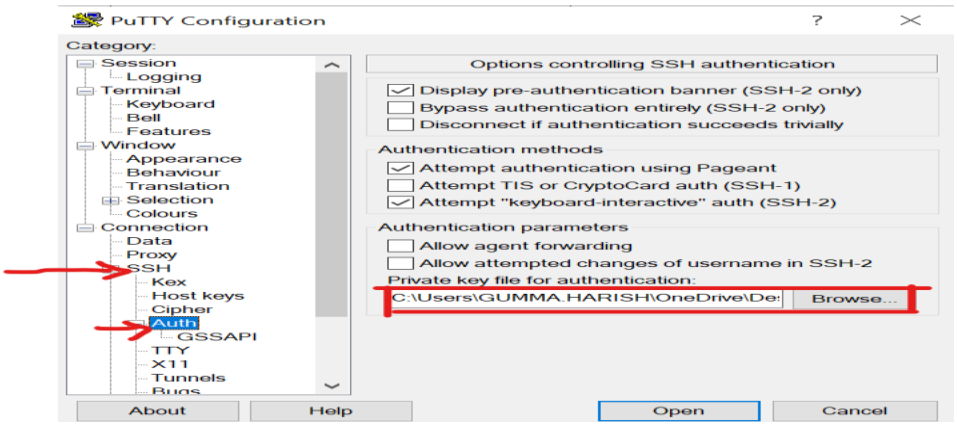
Now access the machine



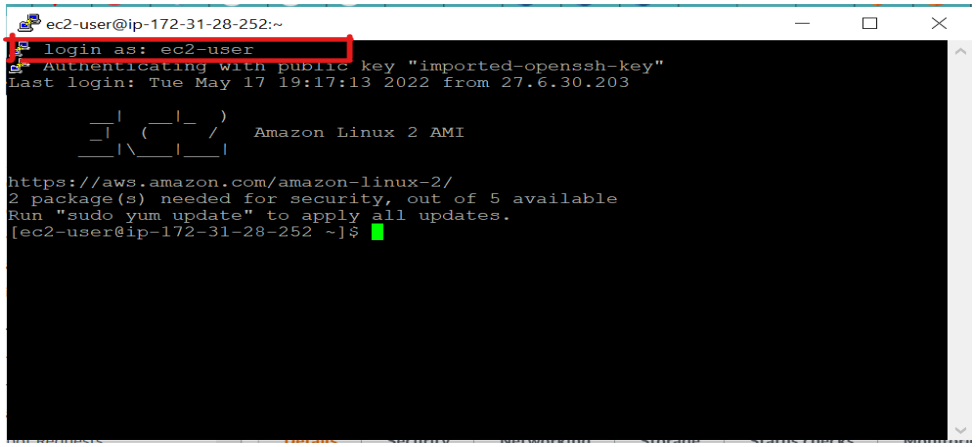
- Take the public DNS and paste into putty
- Open the putty under hostname we need add the public DNS of the instance



Now SSH under auth and browse for the ppk file (putty need the ppk file)



After open putty we will get the ec2 terminal . we need to login with the ec2-user



Step 4: Run the commands to install web package

sudo su

yum update -y

yum install httpd -y

cd /var/www/html

echo "MyGoogle-1" > index.html

ls

service httpd start

chkconfig httpd on

1.sudo -i

```
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-28-252 ~]$ sudo su
[root@ip-172-31-28-252 ec2-user]# sudo -i
[root@ip-172-31-28-252 ~]#
```

2.yum update -y

```
Installed:
  kernel.x86_64 0:5.10.112-108.499.amzn2

Updated:
  curl.x86_64 0:7.79.1-2.amzn2.0.1
  iproute.x86_64 0:5.10.0-2.amzn2.0.2
  kernel-tools.x86_64 0:5.10.112-108.499.amzn2
  libcurl.x86_64 0:7.79.1-2.amzn2.0.1
```

3. yum install httpd -y

Output

```
Installed:
  httpd.x86_64 0:2.4.53-1.amzn2

Dependency Installed:
  apr.x86_64 0:1.7.0-9.amzn2
  apr-util.x86_64 0:1.6.1-5.amzn2.0.2
  apr-util-bdb.x86_64 0:1.6.1-5.amzn2.0.2
  generic-logos-httpd.noarch 0:18.0.0-4.amzn2
  httpd-filesystem.noarch 0:2.4.53-1.amzn2
  httpd-tools.x86_64 0:2.4.53-1.amzn2
  mailcap.noarch 0:2.1.41-2.amzn2
  mod_http2.x86_64 0:1.15.19-1.amzn2.0.1
```

4.cd /var/www/html

```
[root@ip-172-31-28-252 ~]# cd /var/www/html
[root@ip-172-31-28-252 html]#
```

5. echo "MyGoogle-1" > index.html

```
[root@ip-172-31-28-252 html]# echo "MYGOOGLE-1" > index.html
```

6.ls

```
[root@ip-172-31-28-252 html]#
[root@ip-172-31-28-252 html]# ls
index.html
[root@ip-172-31-28-252 html]#
[root@ip-172-31-28-252 html]#
[root@ip-172-31-28-252 html]# cat index.html
MYGOOGLE-1
```

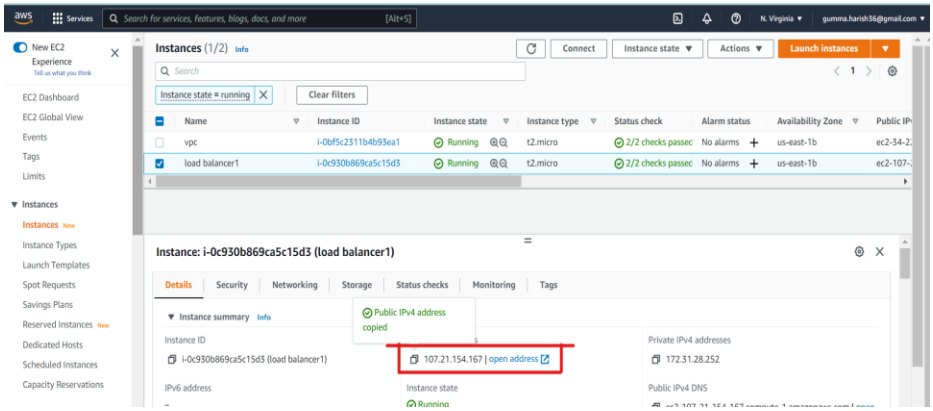
7. service httpd start

8. chkconfig httpd on

```
MYGOOGLE-1
[root@ip-172-31-28-252 html]# service httpd start
Redirecting to /bin/systemctl start httpd.service
[root@ip-172-31-28-252 html]# chkconfig httpd on
Note: Forwarding request to 'systemctl enable httpd.service'.
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-28-252 html]#
```

Step-5 Now take the public ip and paste into the browser

Access the webserver by using public_ip

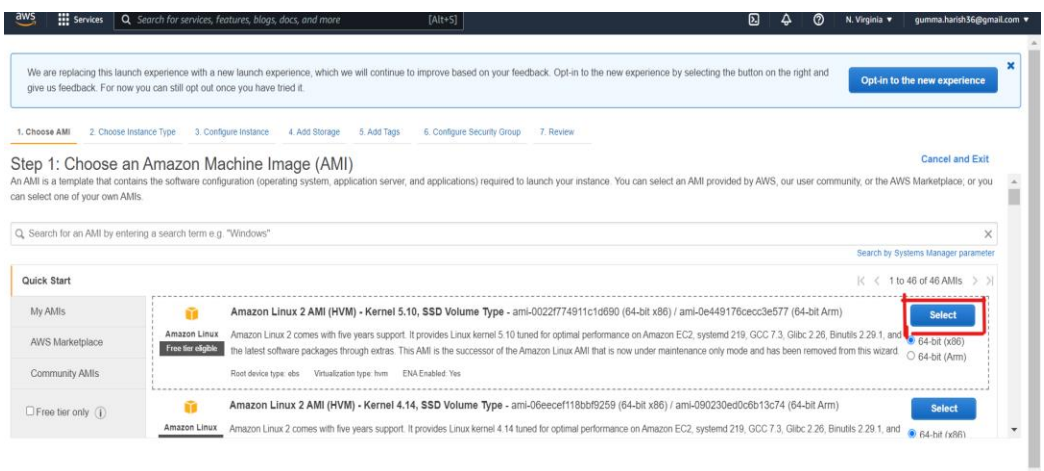


Output and final result

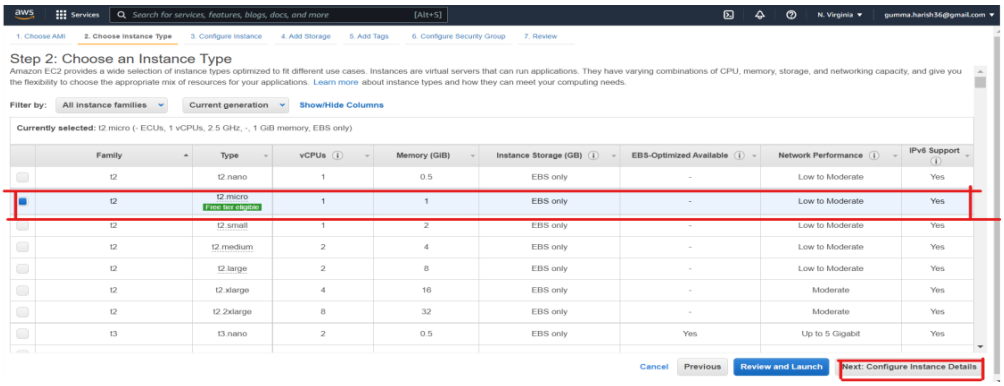
MYGOOGLE-1

Step-6

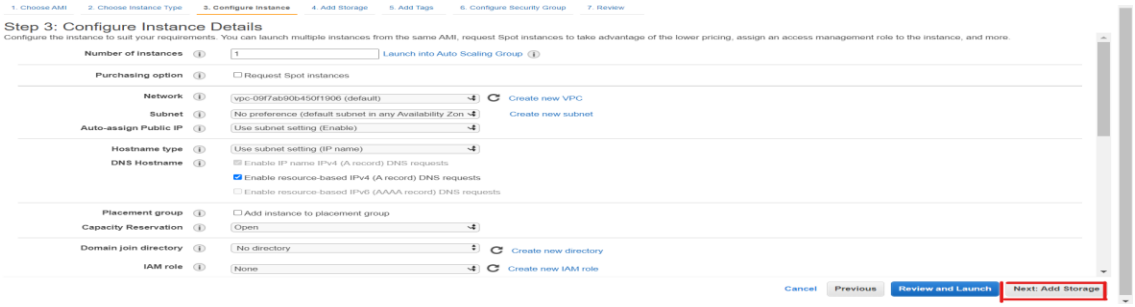
Create a one more instance



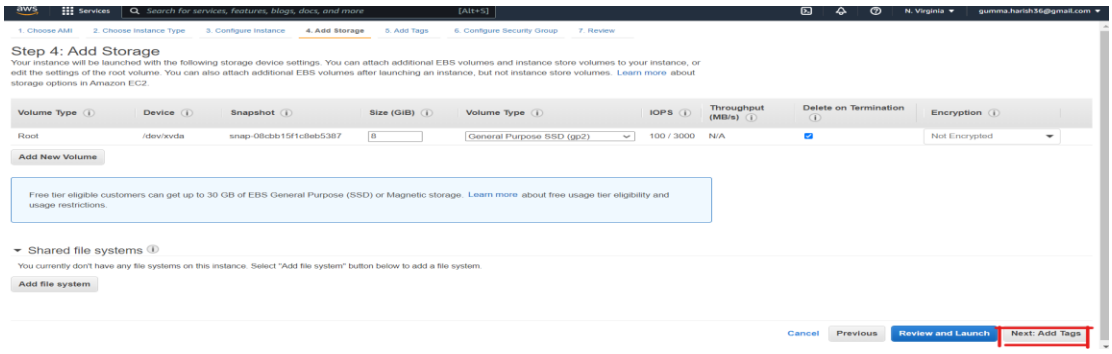
Choose an instance type



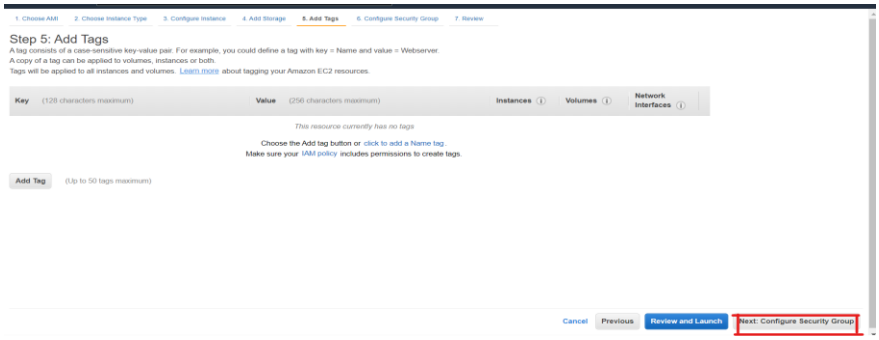
Configure instance details



Add storage

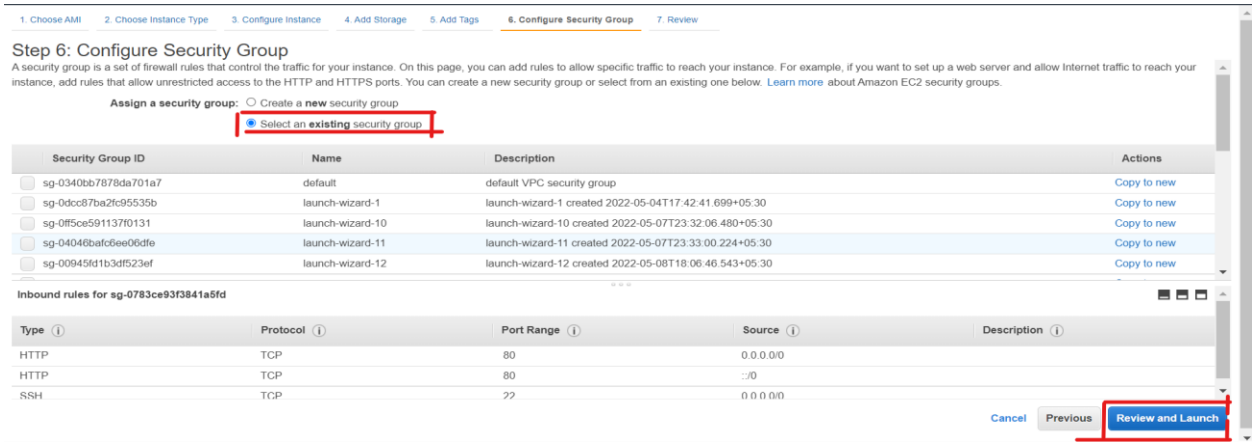


No need to add the tages

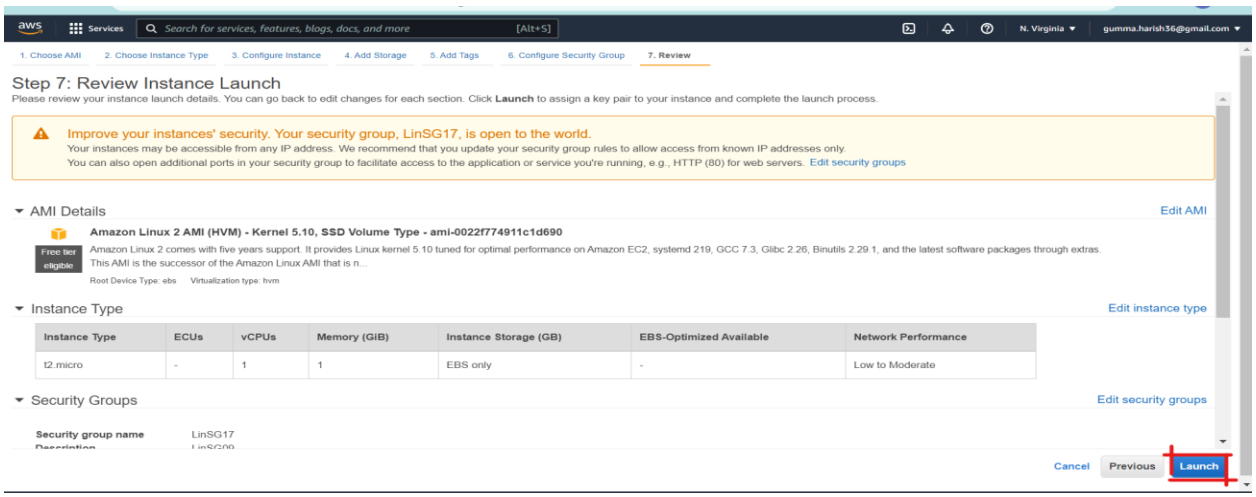


Configure security Group

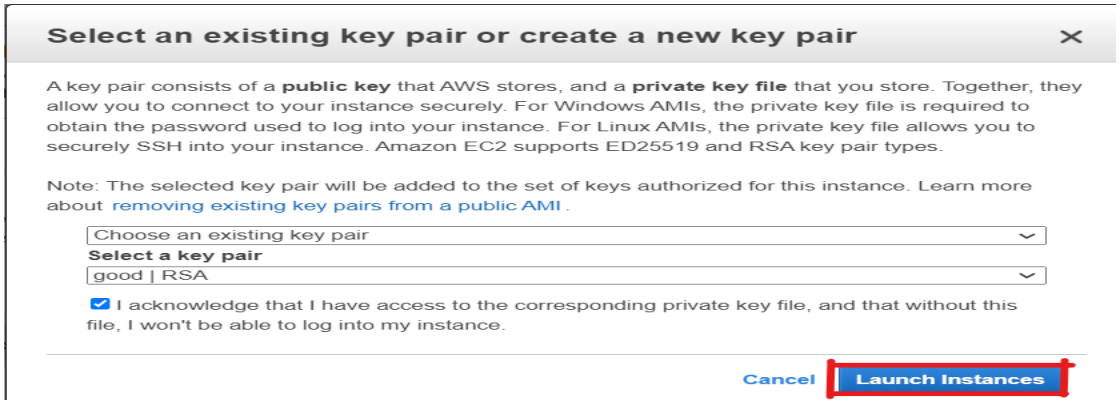
We need to select the existing security group (use the previous security group what we used for the above instance)



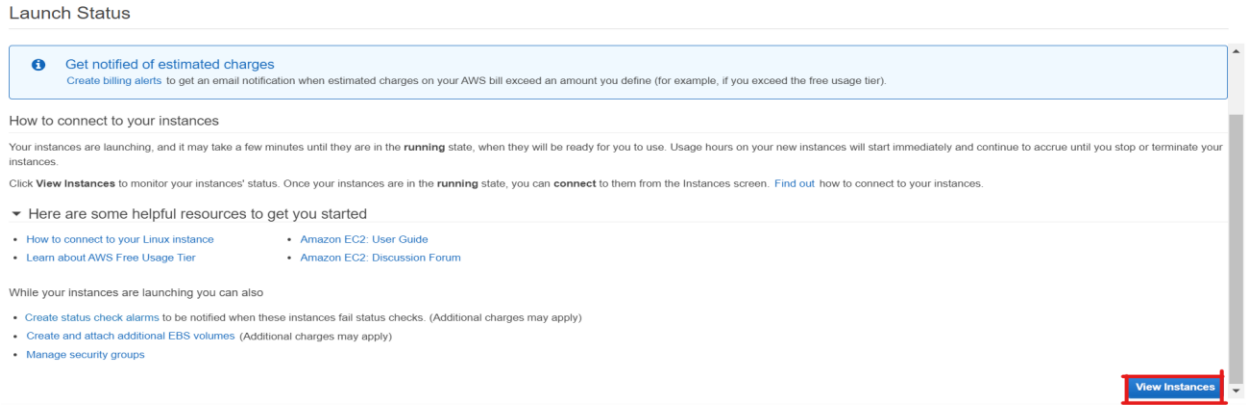
Review instance launch



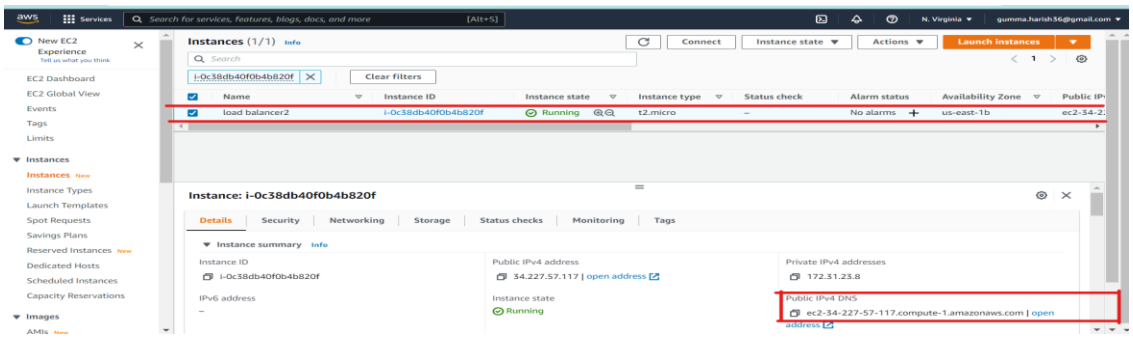
Now select the existing key pair and launch the instance



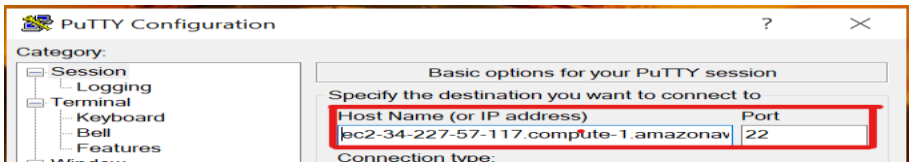
Launch status



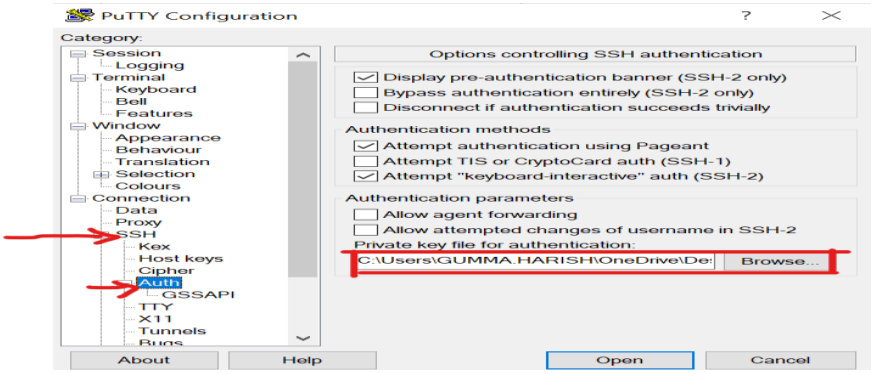
Now new instance had launched (with the name of load balancer2)



- Take the public DNS and paste into putty
- Open the putty under hostname we need add the public DNS of the instance



Now SSH under auth and browse for the ppk file (putty need the ppk file)



After open putty we will get the ec2 terminal . we need to login with the ec2-user

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

https://aws.amazon.com/amazon-linux-2/
2 package(s) needed for security, out of 5 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-23-8 ~]$
```

Step 7: Run the commands to install web package

sudo su

yum update -y

yum install httpd -y

cd /var/www/html

echo "MyGoogle-2" > index.html

ls

service httpd start

chkconfig httpd on

1.sudo su

2. yum update -y

```
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
2 package(s) needed for security, out of 5 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-23-8 ~]$ sudo su
[root@ip-172-31-23-8 ec2-user]# yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
```

```
Installed:
  kernel.x86_64 0:5.10.112-108.499.amzn2

Updated:
  curl.x86_64 0:7.79.1-2.amzn2.0.1      iproute.x86_64 0:5.10.0-2.amzn2.0.2      kernel-tools.x86_64 0:5.10.112-108.499.amzn2      libcurl.x86_64 0:7.79.1-2.amzn2.0.1

Complete!
[root@ip-172-31-23-8 ec2-user]#
```

3.yum install httpd -y

```
[root@ip-172-31-23-8 ec2-user]# yum install httpd -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.53-1.amzn2 will be installed
--> Processing Dependency: httpd-tools = 2.4.53-1.amzn2 for package: httpd-2.4.53-1.amzn2.
--> Processing Dependency: httpd-filesystem = 2.4.53-1.amzn2 for package: httpd-2.4.53-1.a
--> Processing Dependency: system-logos-httpd for package: httpd-2.4.53-1.amzn2.x86_64
--> Processing Dependency: mod_http2 for package: httpd-2.4.53-1.amzn2.x86_64
--> Processing Dependency: httpd-filesystem for package: httpd-2.4.53-1.amzn2.x86_64
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.53-1.amzn2.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.4.53-1.amzn2.x8
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.4.53-1.amzn2.x86_64
--> Running transaction check
---> Package apr.x86_64 0:1.7.0-9.amzn2 will be installed
```

4.cd /var/www/html

```
[root@ip-172-31-23-8 ~]# cd /var/www/html
[root@ip-172-31-23-8 html]#
```

5. echo "MyGoogle-2" > index.html

```
[root@ip-172-31-23-8 html]# echo "MYGoogle-2" >index.html
```

6.ls

```
[root@ip-172-31-23-8 html]# ls
index.html
```

7.service httpd start

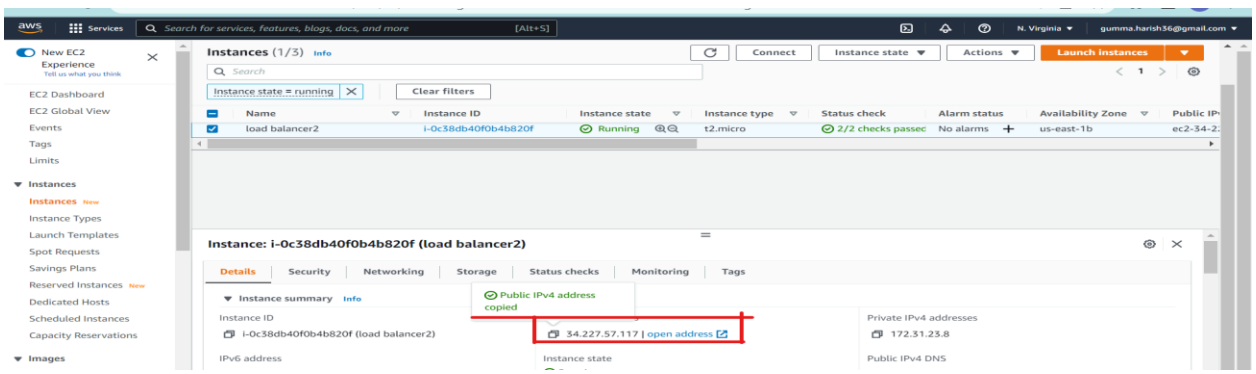
```
[root@ip-172-31-23-8 html]# service httpd start
Redirecting to /bin/systemctl start httpd.service
```

8.chkconfig httpd on

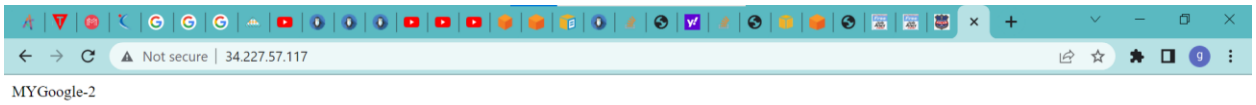
```
[root@ip-172-31-23-8 html]# chkconfig httpd on
Note: Forwarding request to 'systemctl enable httpd.service'.
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
[root@ip-172-31-23-8 html]#
```

Step -9 Now take the public ip and paste into the browser

Access the webserver by using public_ip

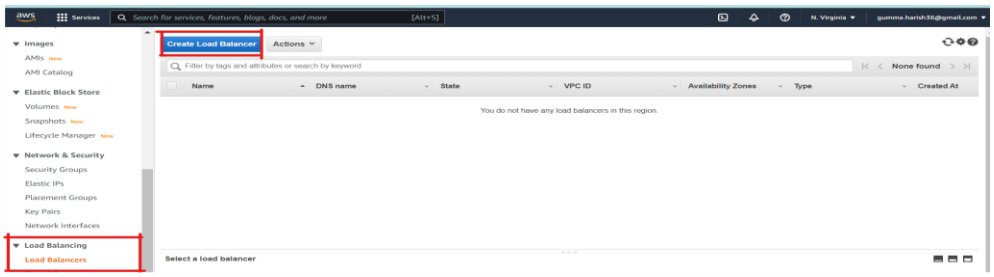


Final out put

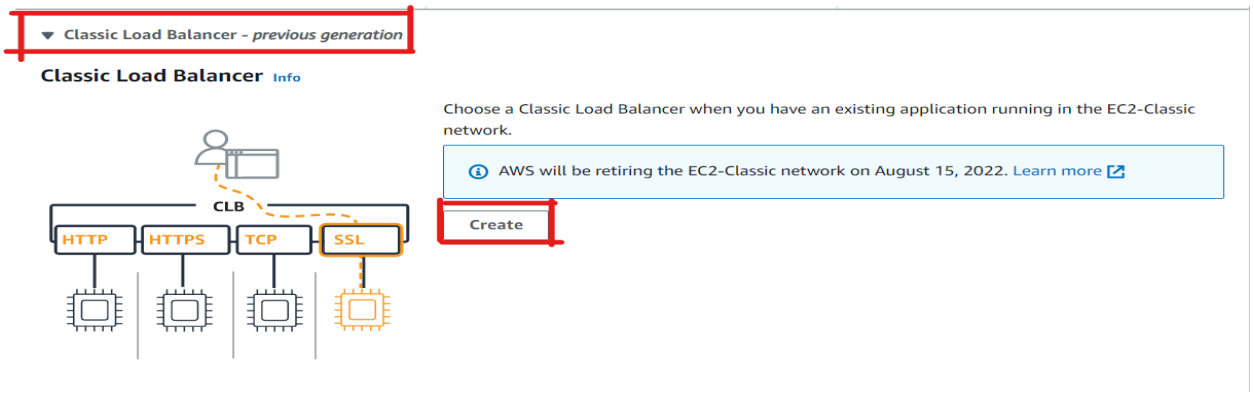


Step-10 : Create load balancers

Select classic load balancer



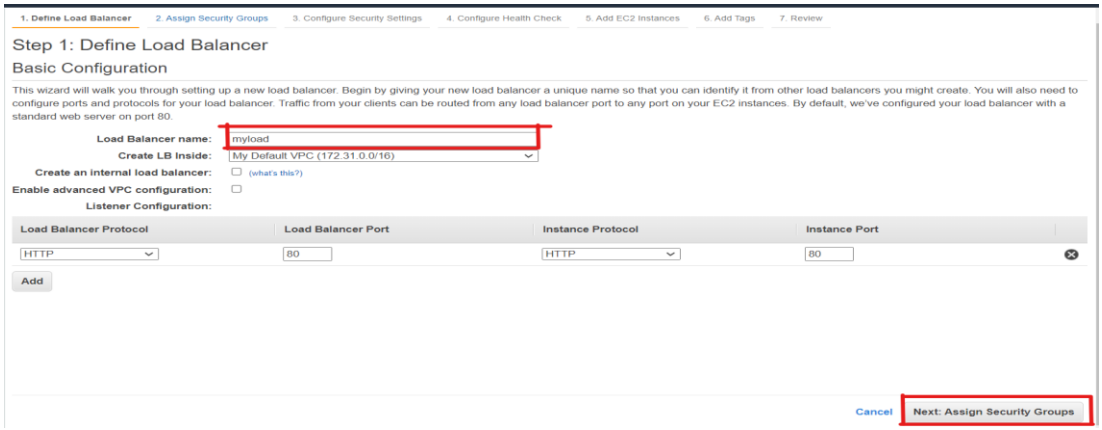
Select the classic load balancer



Step-11

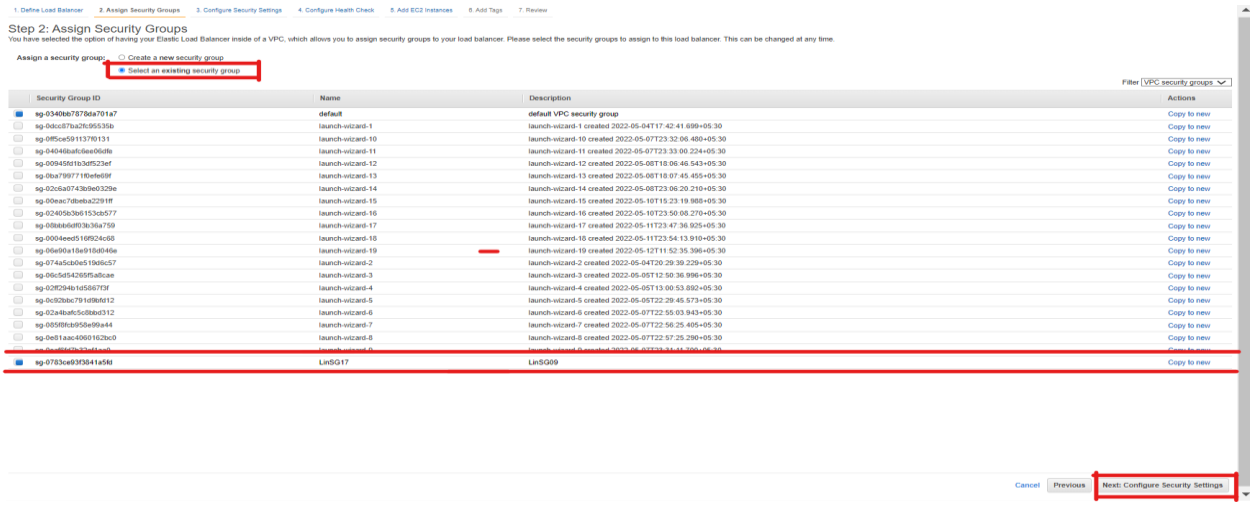
Load Balancer Name - Myload--> Next ----> select existing security group ---> NEXT --

Define load balancer

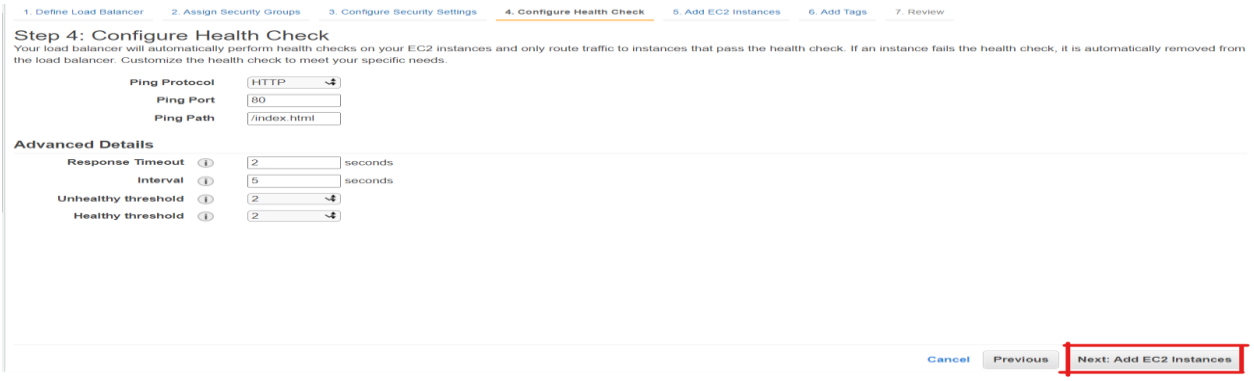


Assign security groups

(select the existing security group which we used for instances)



Configure Health Check



Response Timeout - 2 Seconds

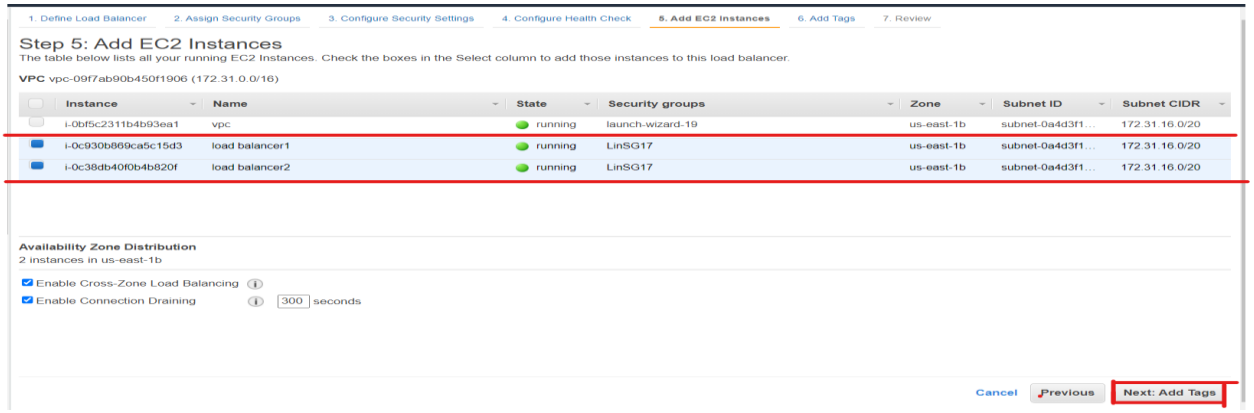
Interval - 5 Seconds

Unhealthy threshold - 2

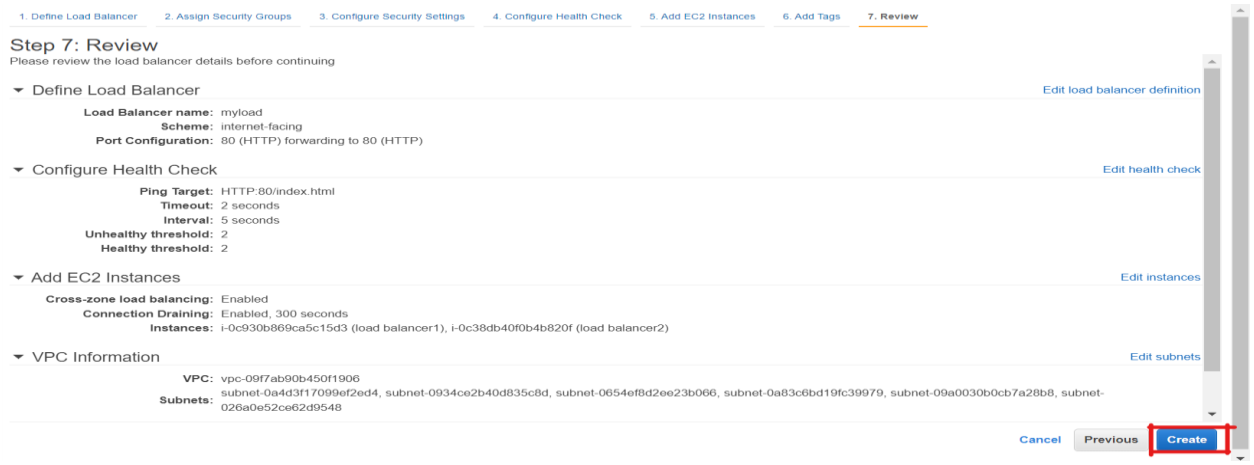
Healthy threshold – 2

Next -- Attach both the instances

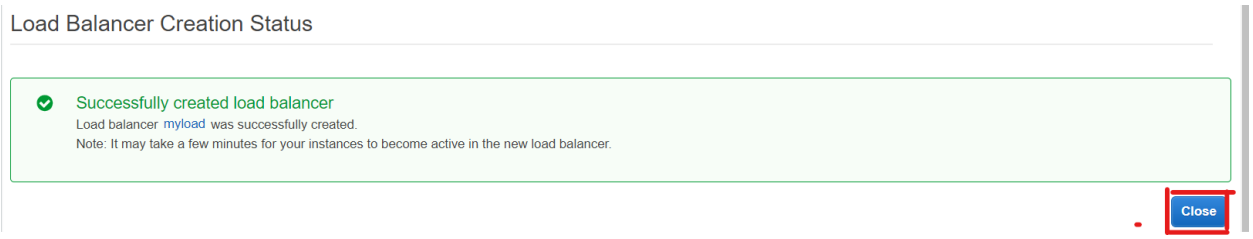
Add EC2 instances (what we created)

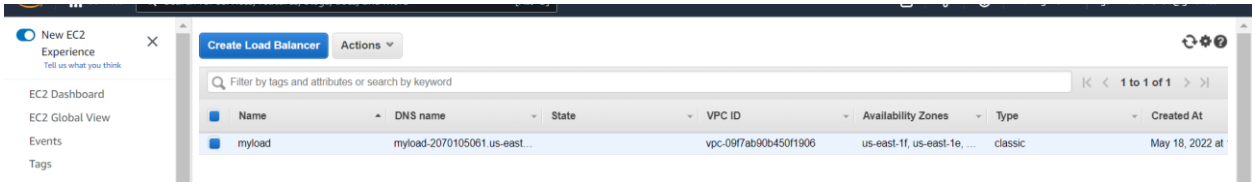


Review



Load Balancer Creation Status





Step-12

Access the load balance by using DNS

and experience the load balancer.

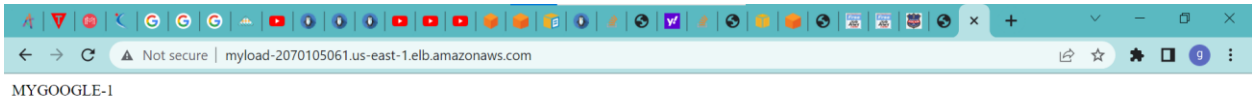
Take the DNS of load balancer

Using the load balancer DNS we are getting the content

The main aim of the load balancer If one server is down, it should redirect the traffic to another server.

Now load balance will start sending the traffic

Load balancer will also monitor the health checks of ec2 instances



Same DNS we are getting different content because load balancer will distribut the traffic .if one server goes download it automatically route the traffic to another server