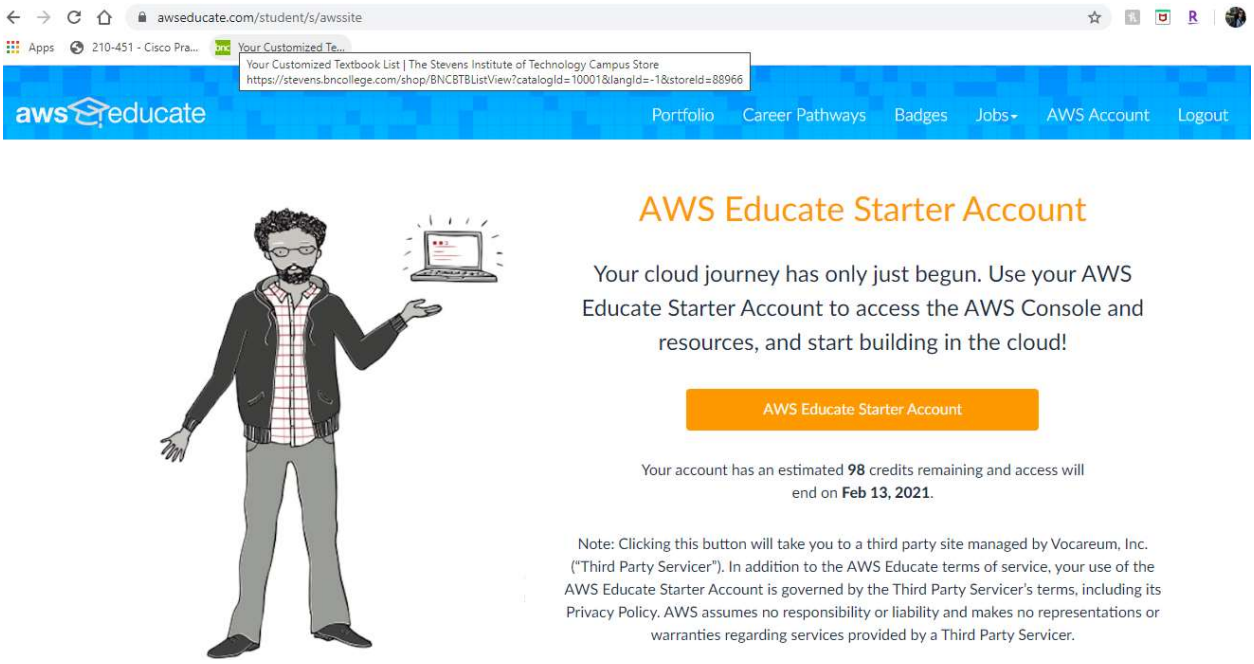
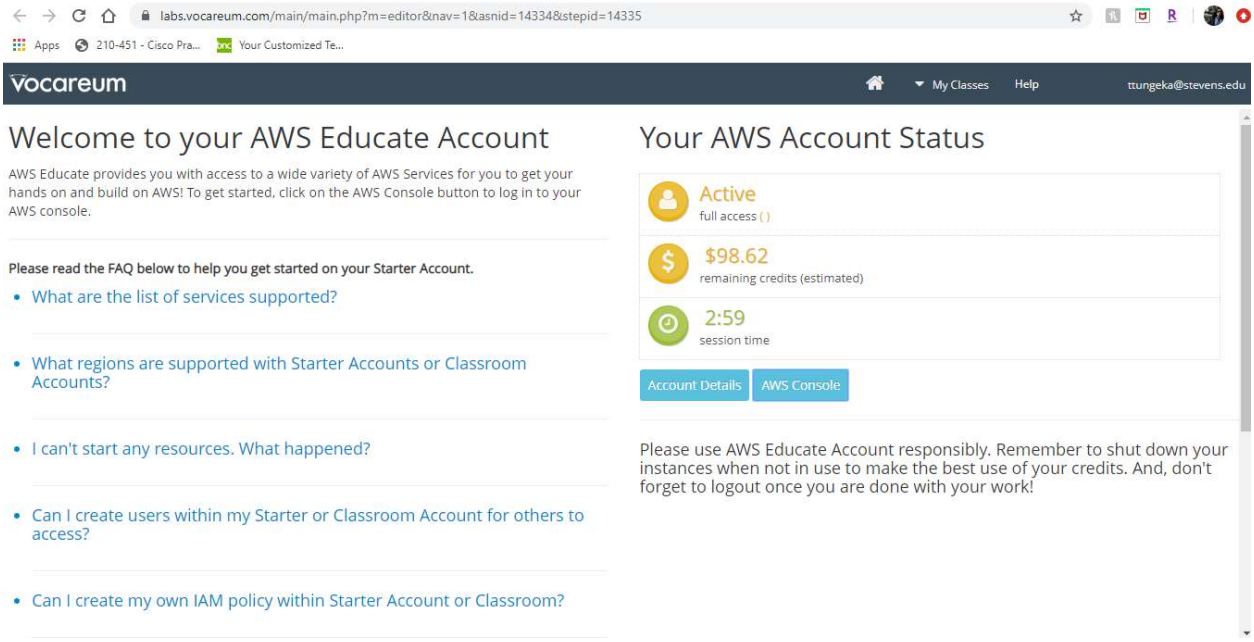
Lab #2

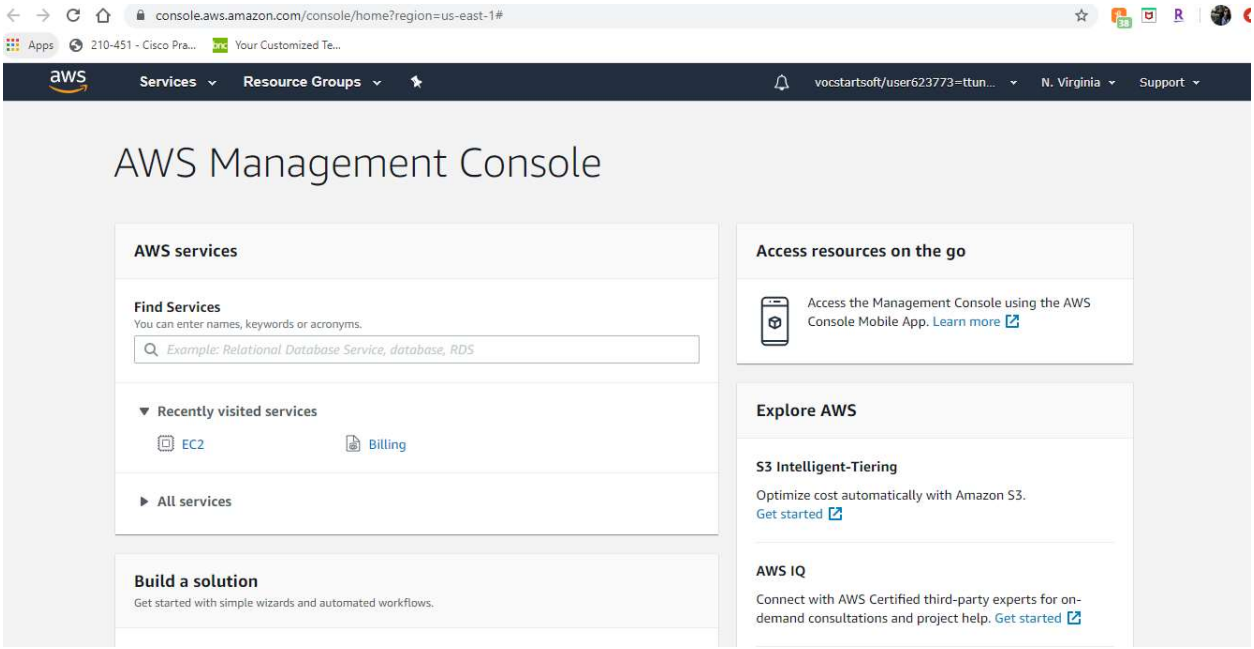
1)Create Amazon EC2 Instances:

Step 1: Sign in Amazon AWS account:



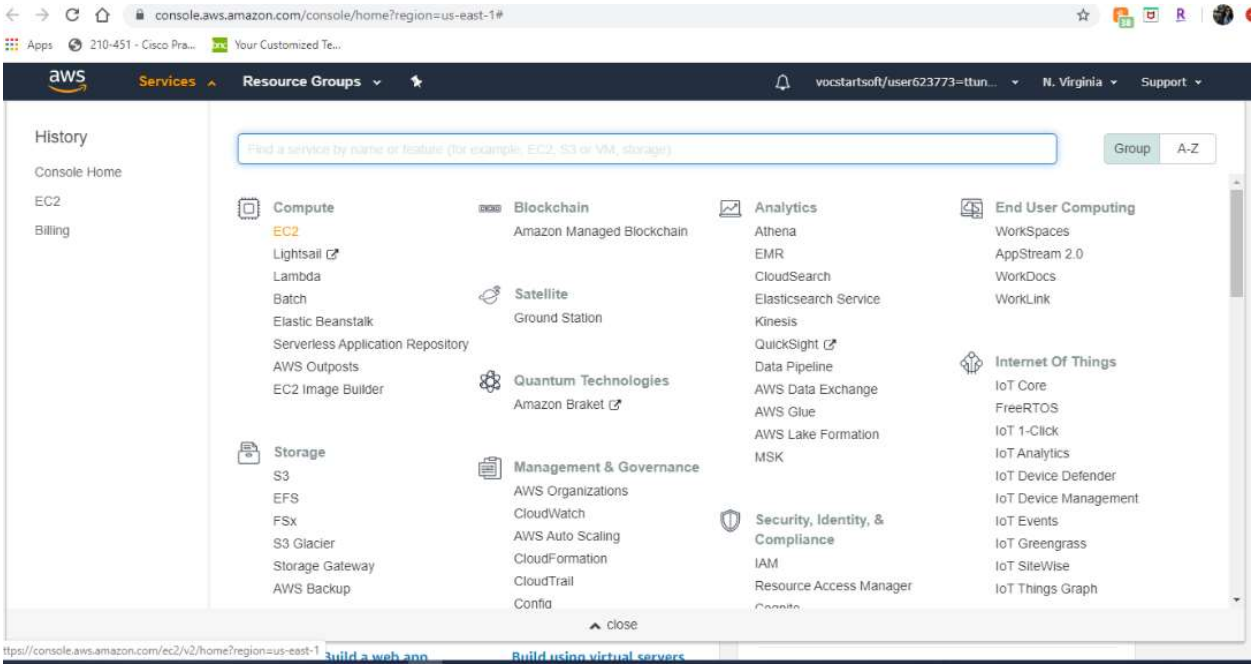


Step 2:

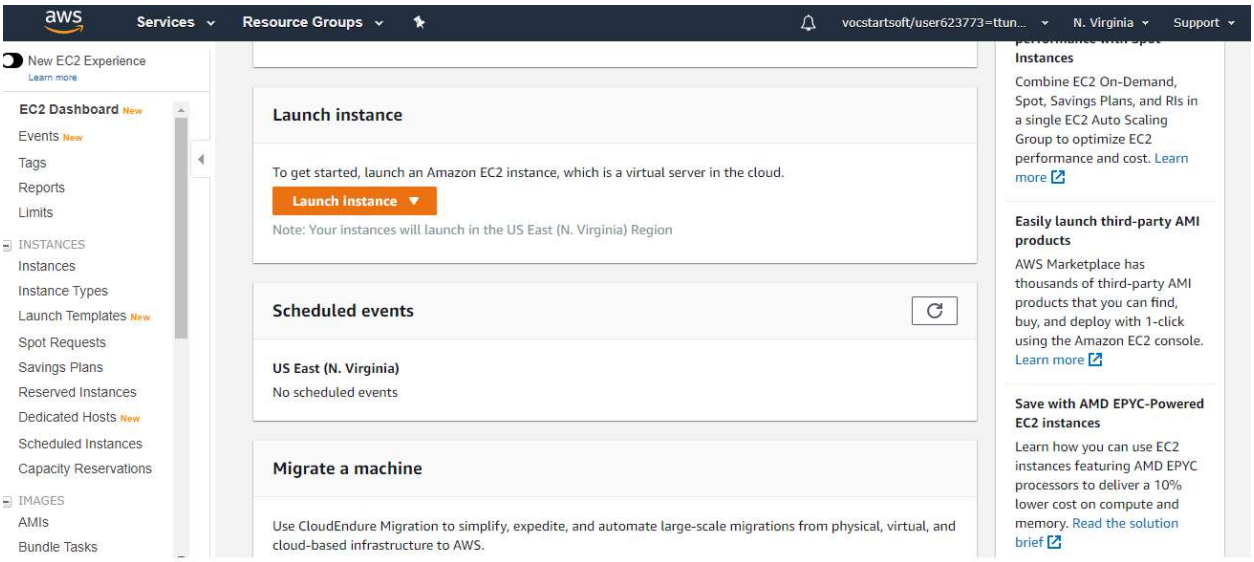


Step 3:

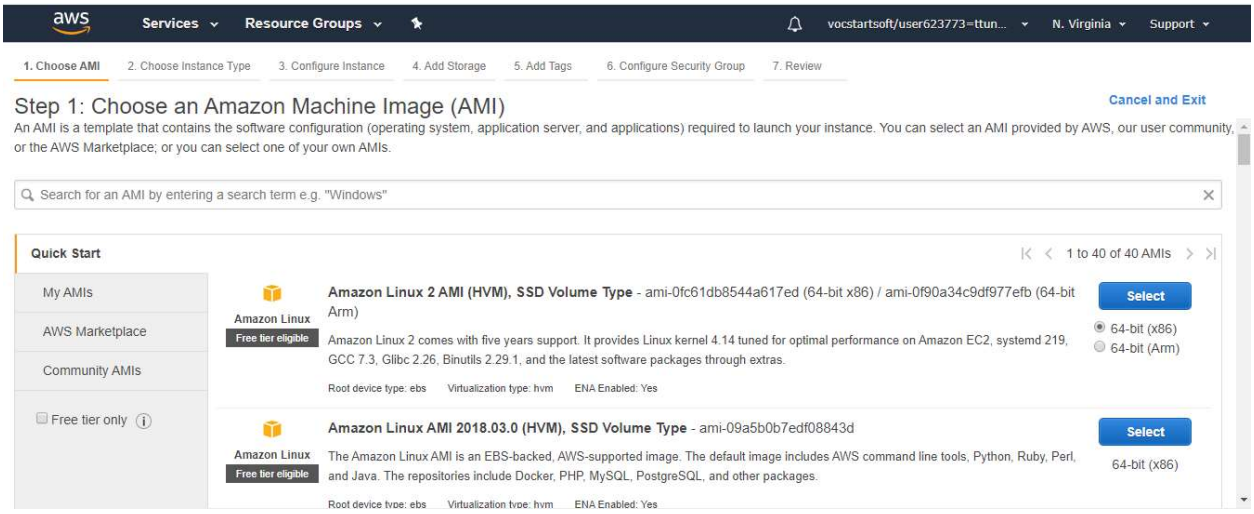
Click on Services dropdown menu, Select EC2 under Compute:



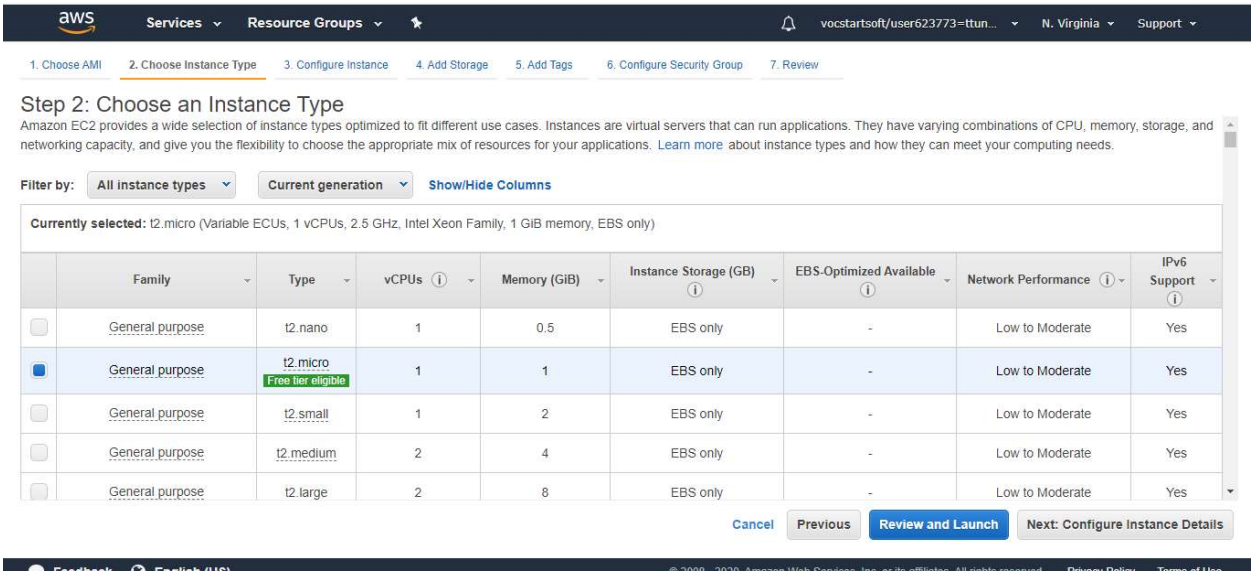
Step 4: On EC2 Dashboard, Click on Launch Instance:



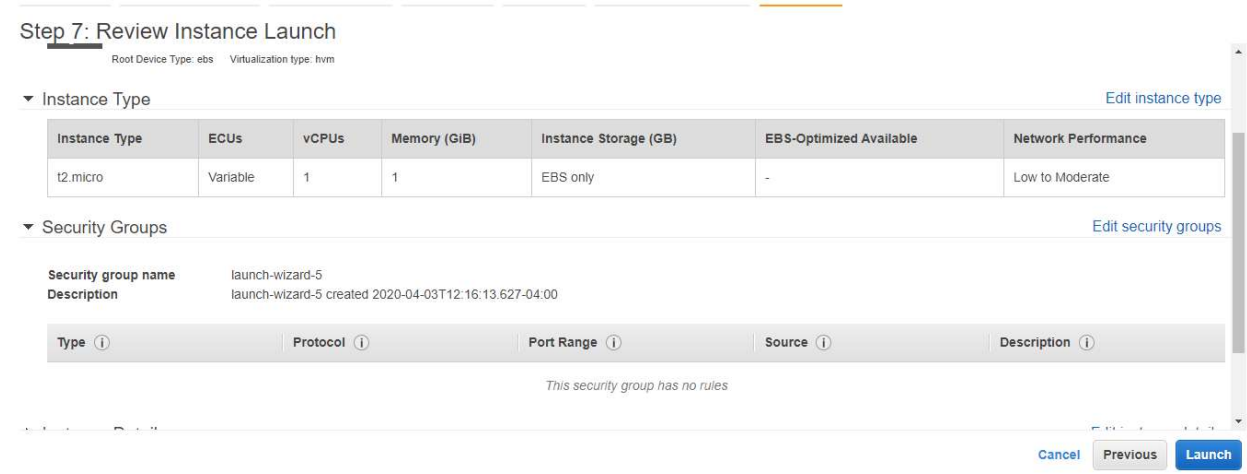
Select the Amazon Linux 2 AMI as shown below:



Select t2.micro as instance type and Click on Preview and Launch button:

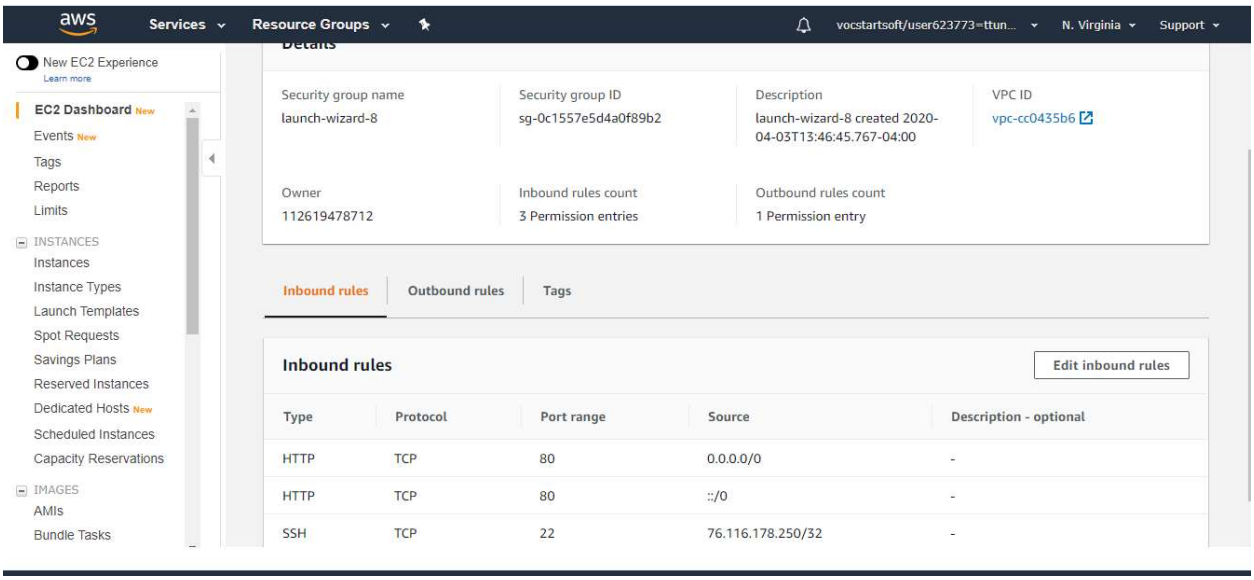


Click on Edit Security Group on the next page:



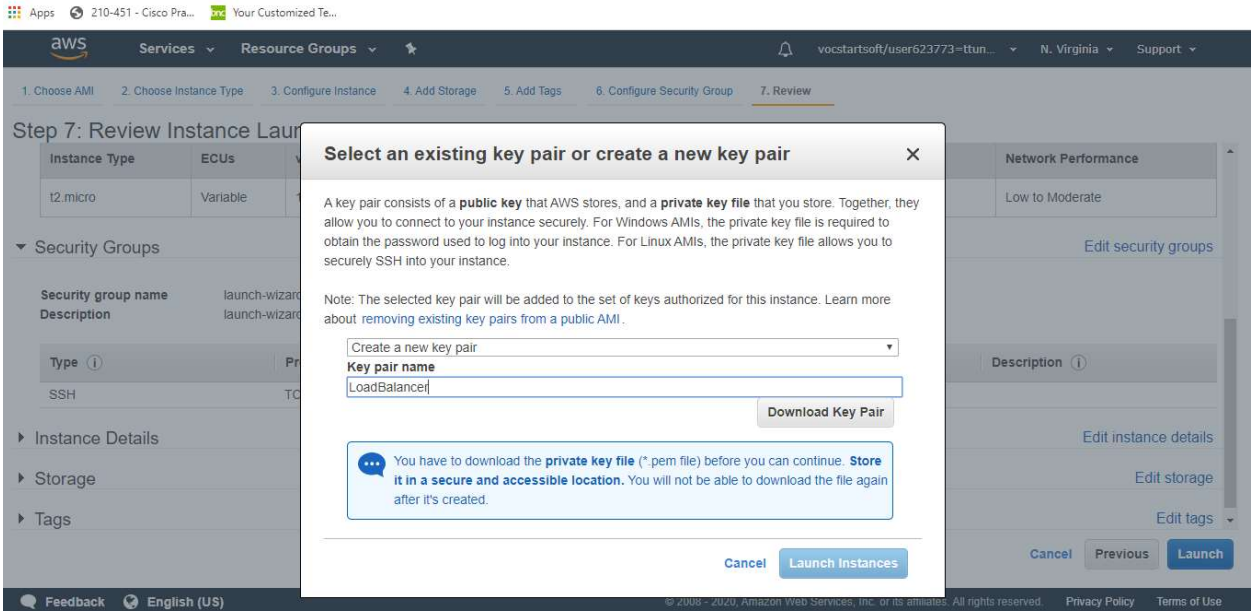
Review the Security Group:

Open port 80 and 22:

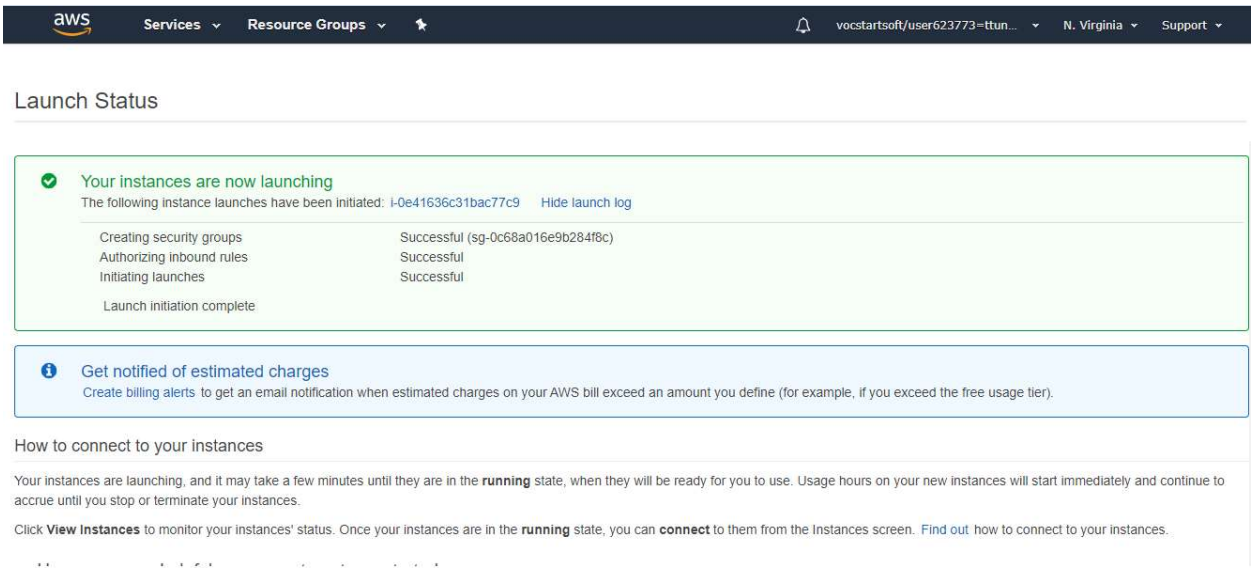


Click on Launch:

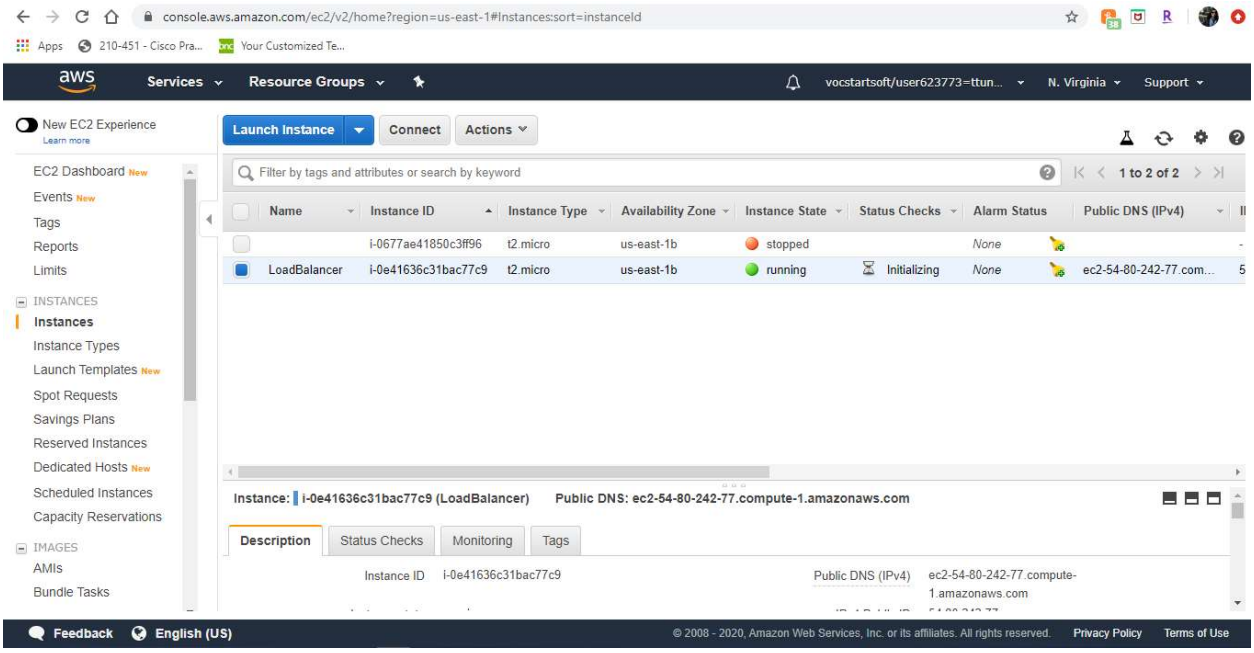
Since this instance is for LoadBalancer, give the key that name and download the key pair:



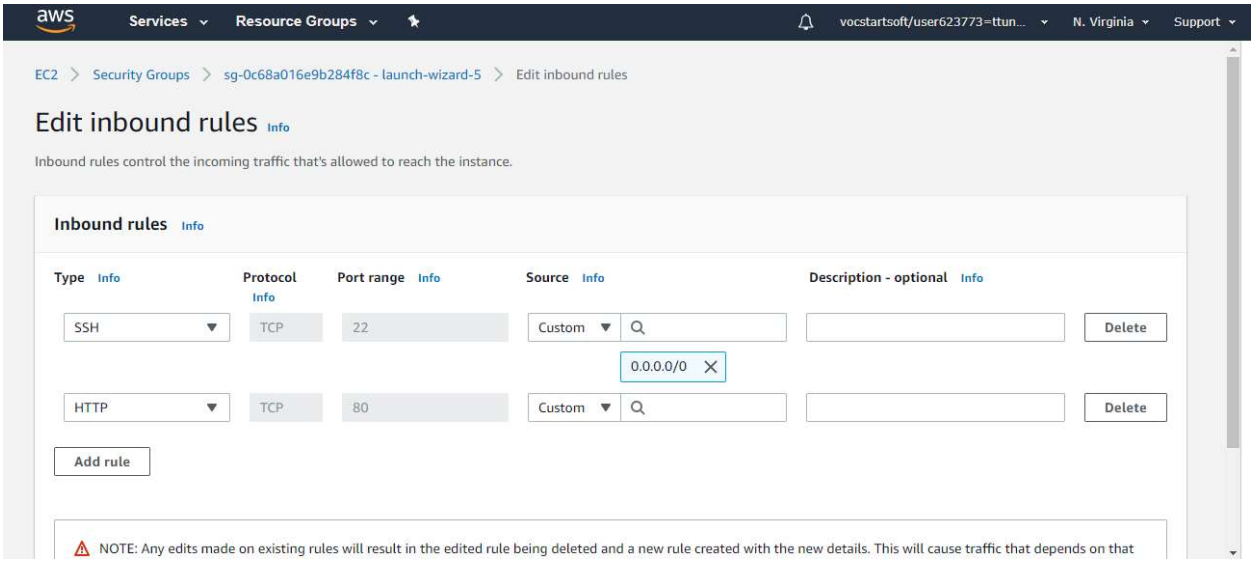
After downloading the key-pair, click on Launch Instances:



In the View Instances tab on EC2 Console, this instance can be viewed. I have named this one as Load Balancer:

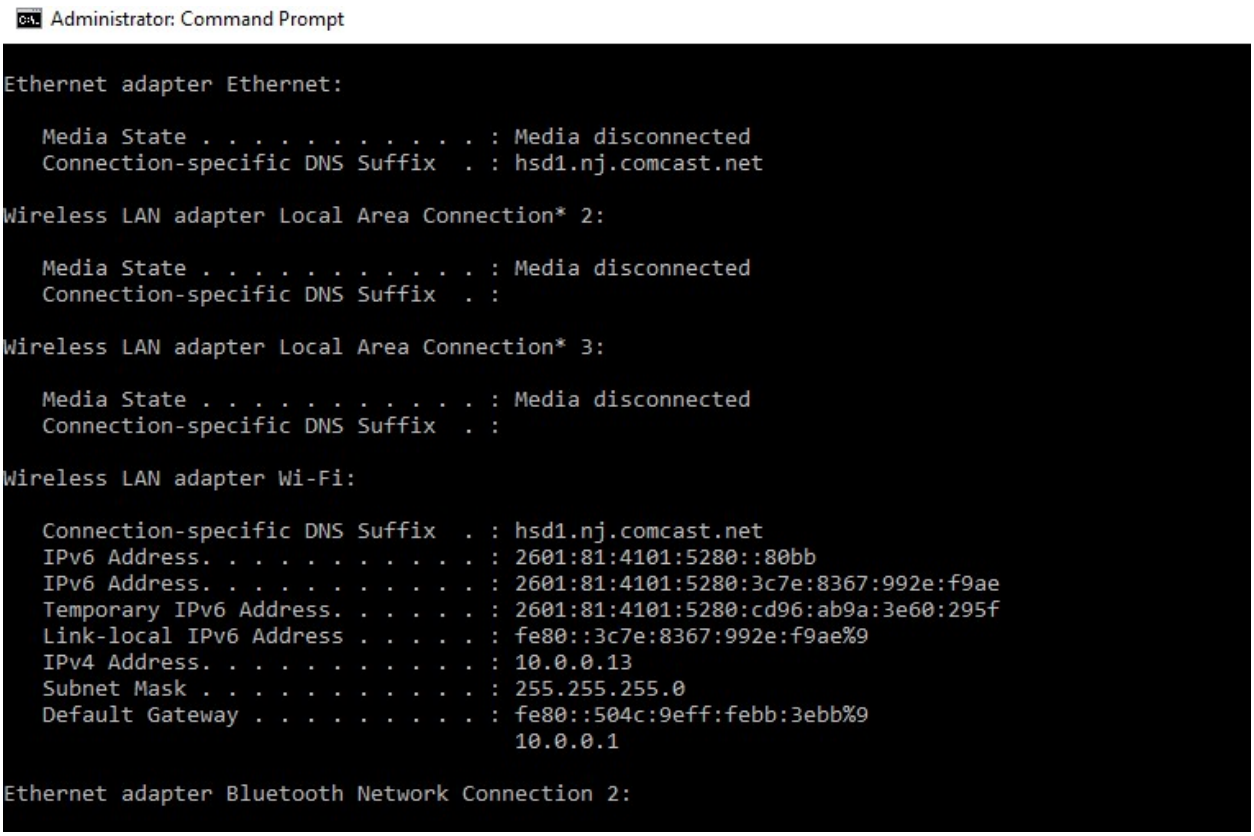


Note: For Load Balancer, security group should be:

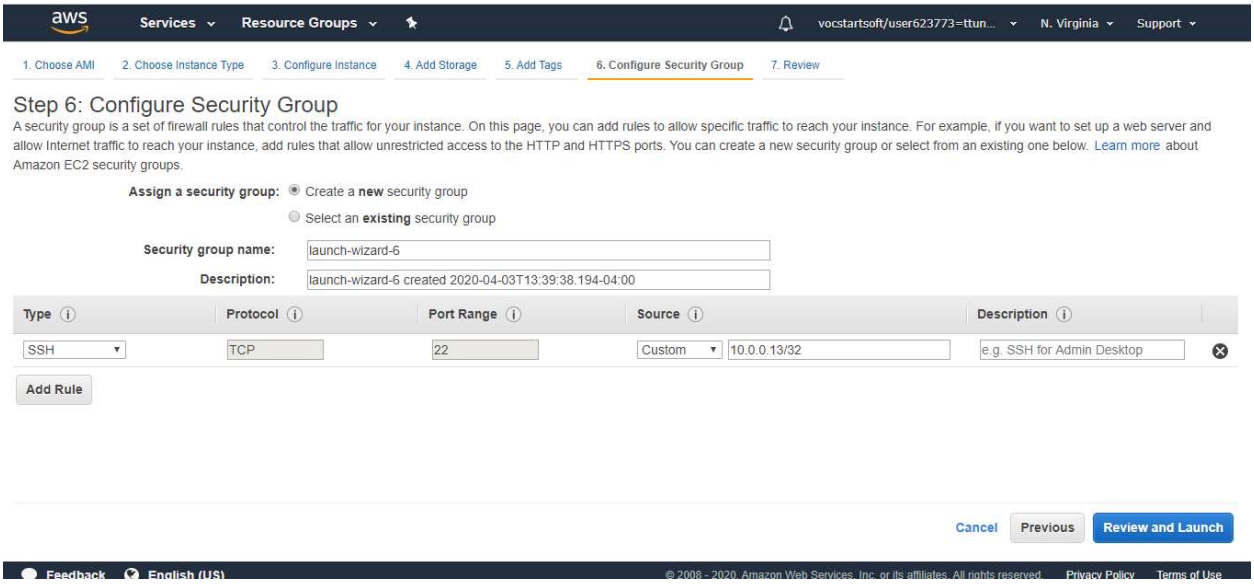


Similarly, repeating the above steps, I have created 4 more instances, namely, Server1, Server2, Server 3 and Server 4 which can be viewed in the View Instances tab on EC2 Console:

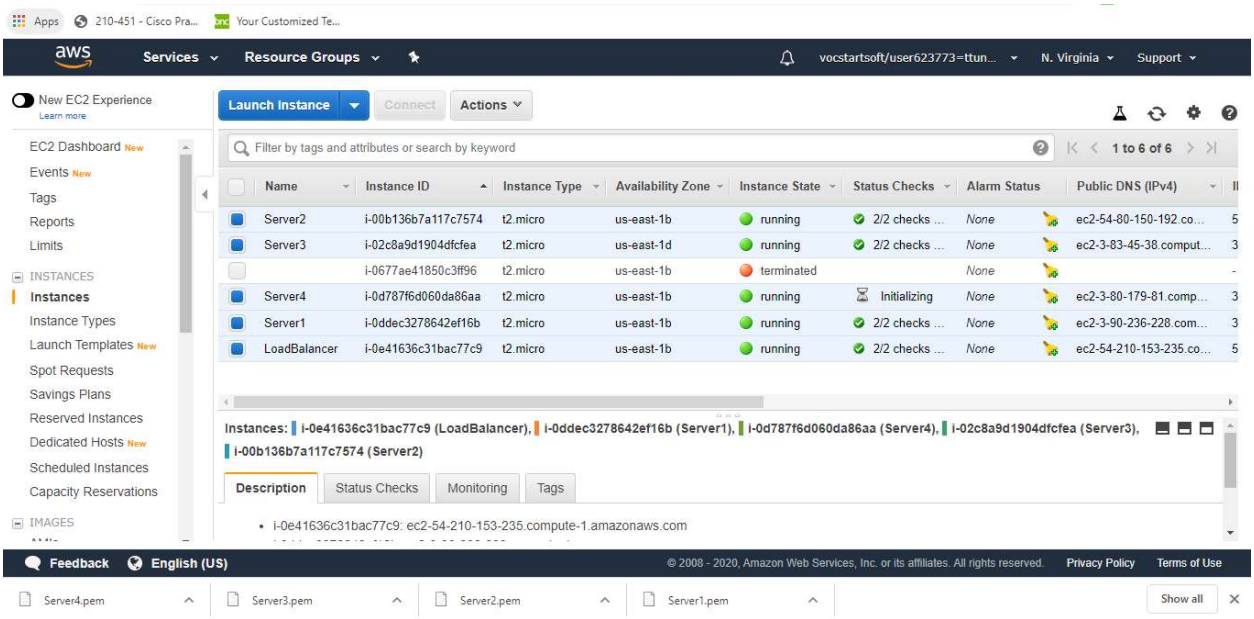
Step 4: For Server 1, Server 2, Server 3, Server 4; IP address of my machine should be in the Security Group:



So I used 10.0.0.13/32(which was not my IP address)in Security group so only my machine would be able to communicate with the instances, but I was not able to connect. This step has to be done for all servers 1 to 4. Also, their corresponding Key Pairs have to be downloaded.

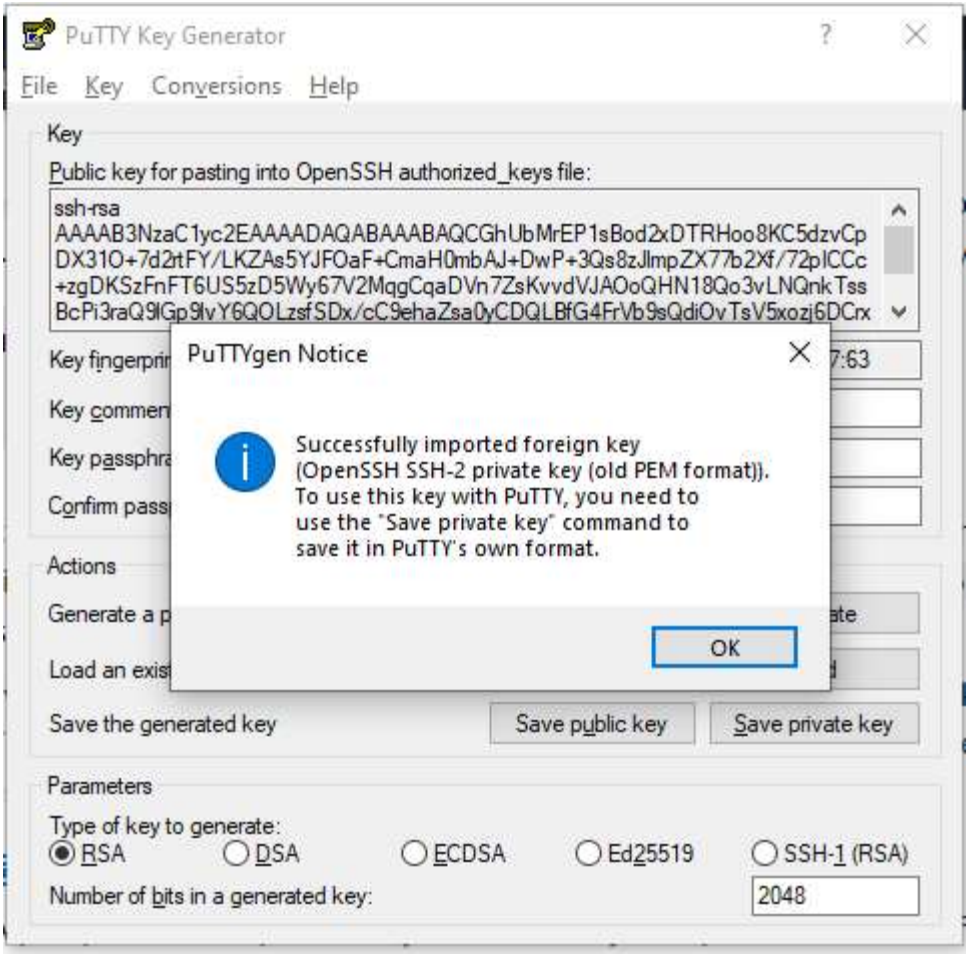


Step 5: The highlighted ones show four servers and one load balancer that was configured.

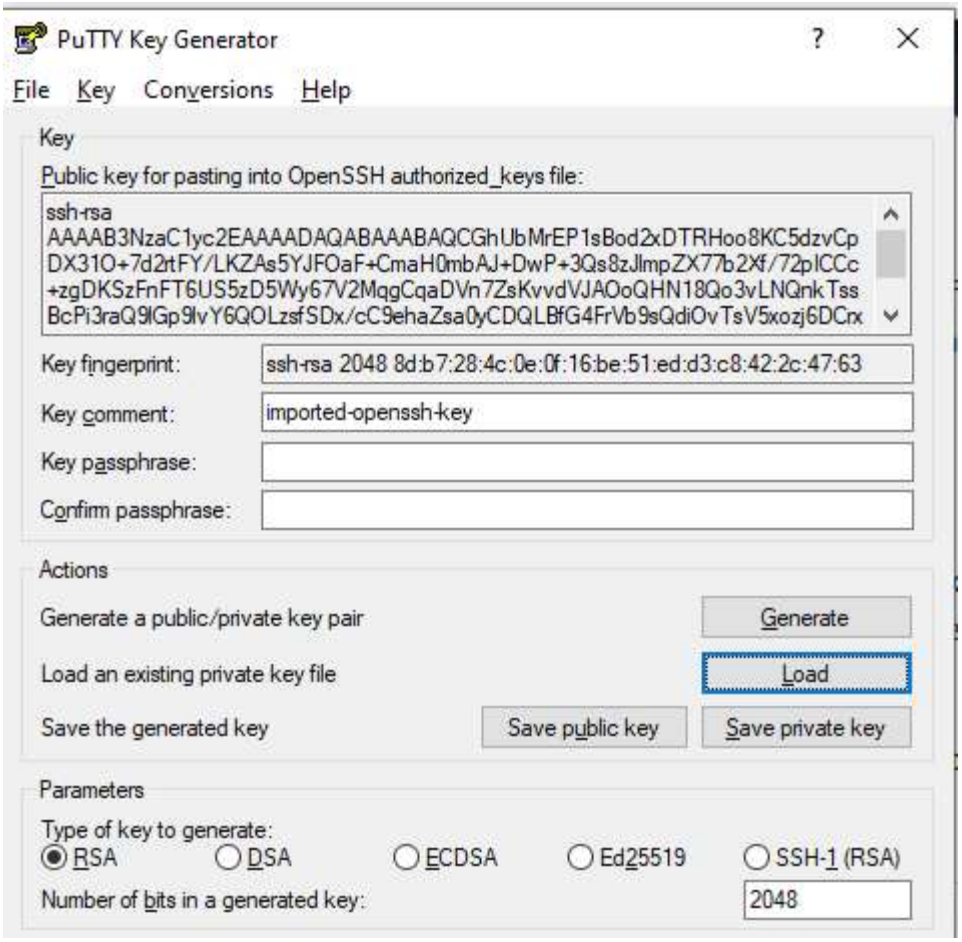


Step 6:

In order to connect to LoadBalancer Console, firstly the .pem key has to be converted to .ppk using pUTTYGen:



Click on Save Private Key:

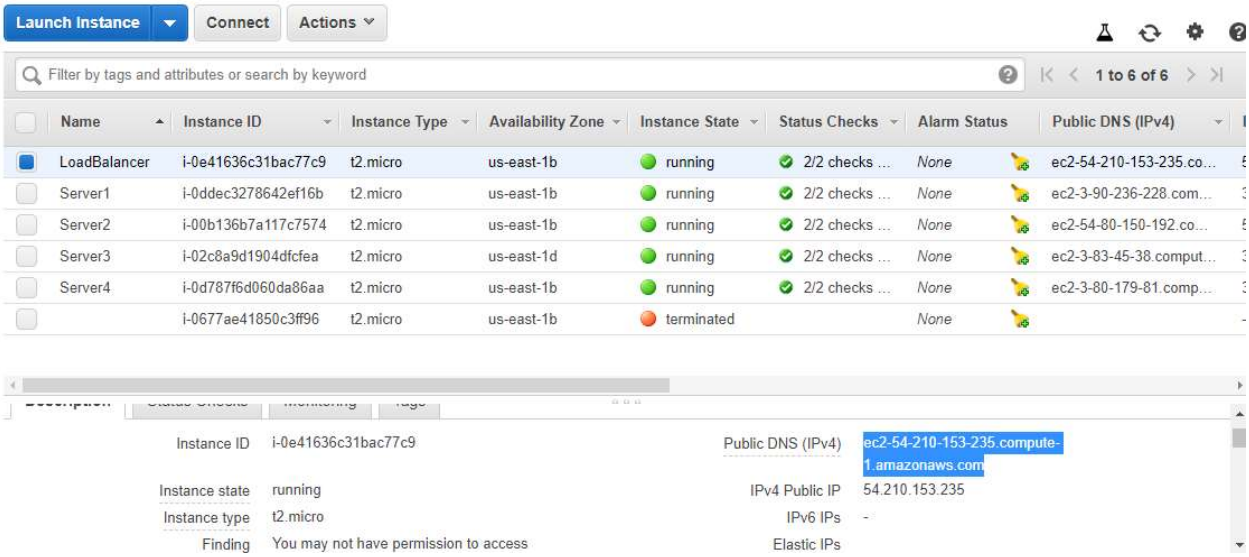


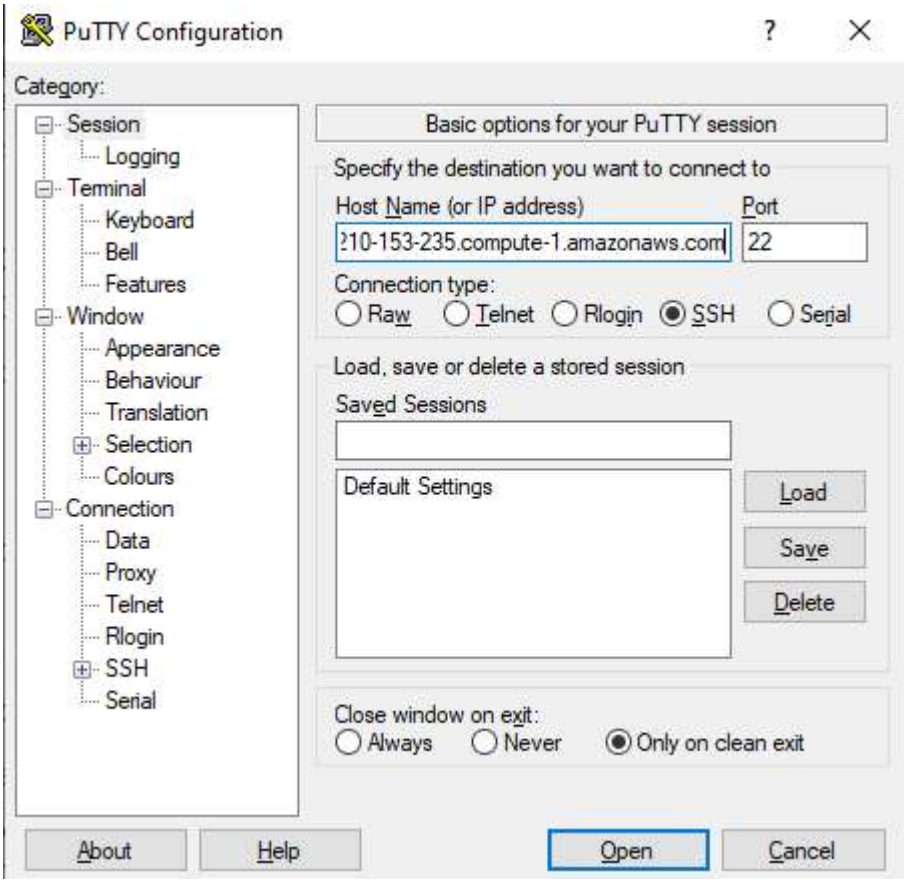
In this way, LoadBalancer.ppk file is created.

Step 7:

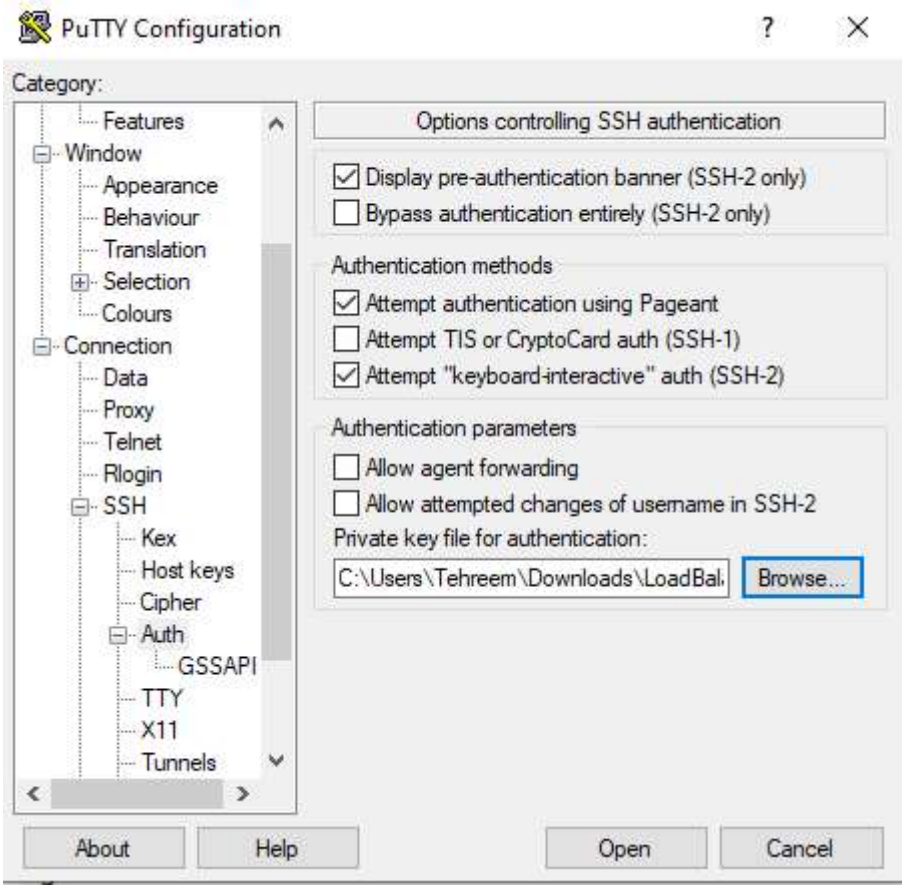
Start PuTTY (from the **Start** menu, choose **All Programs, PuTTY, PuTTY**)

In Session Category, For Host Name or IP Address, input the Public DNS of the LoadBalancer, highlighted in the figure below: Ensure that the **Port** value is 22 and Under **Connection type**, select **SSH**.





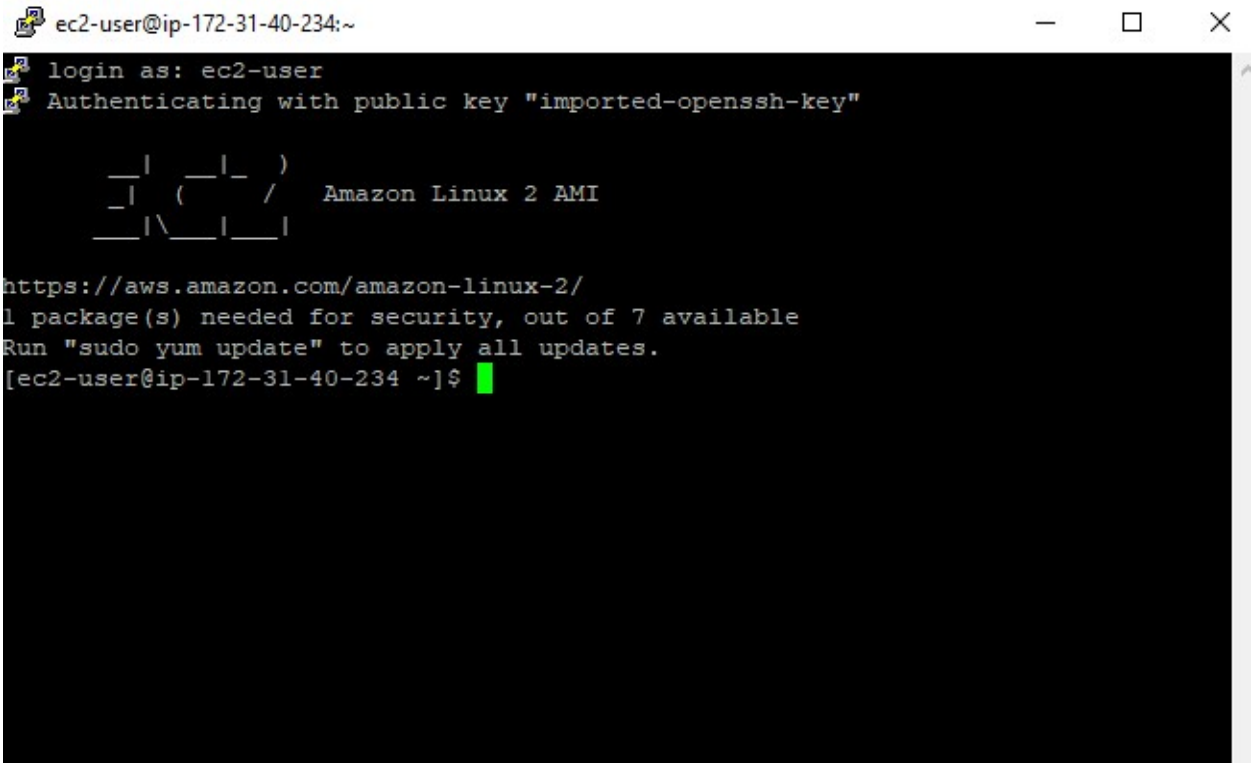
Under Category, Choose Connections and expand it. Now Choose Auth and browse the LoadBalancer.ppk file created in the previous step:



Click Open.

Click Yes on the Security Warning as I trust this instance.

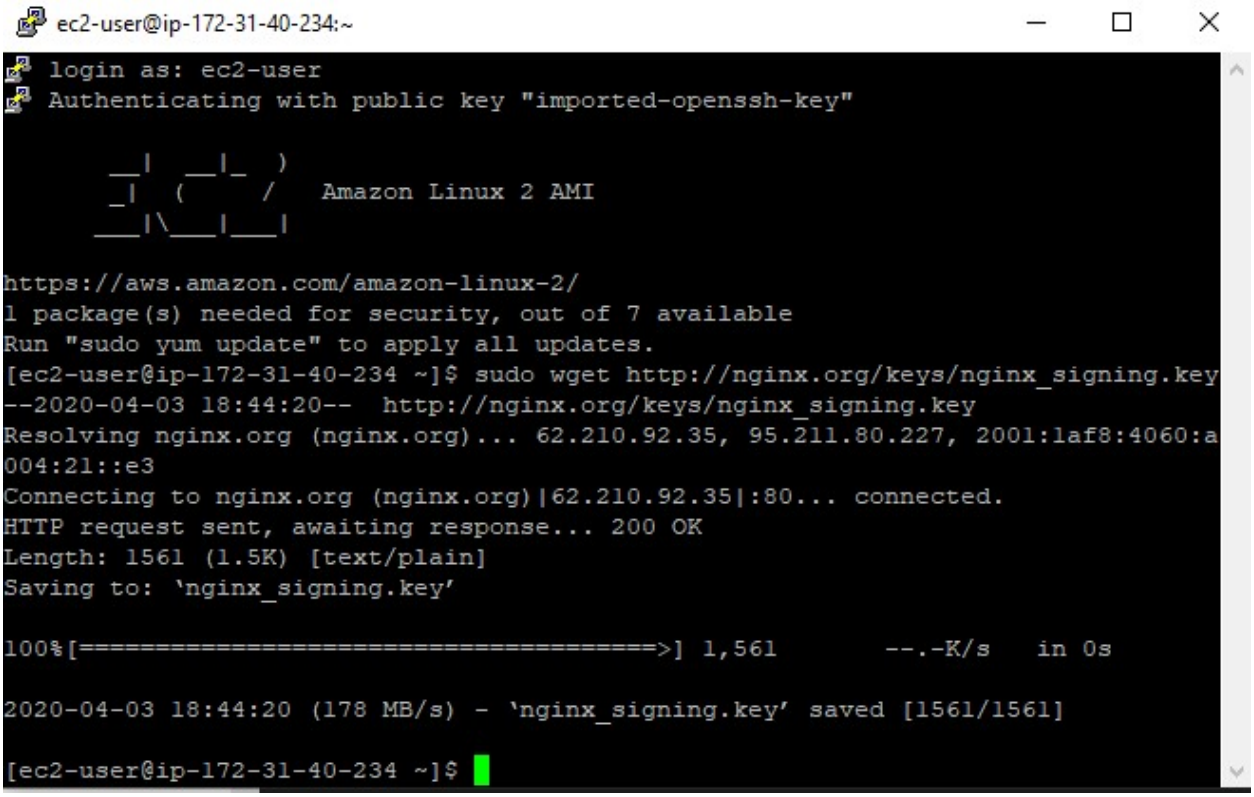
Enter login as ec2-user:



Now run this command:

To download Nginx key:

sudo wget http://nginx.org/keys/nginx_signing.key



Step 7:

Since sudo apt-key command was not working on my machine, I installed it using amazon-linux-extra:

root@ip-172-31-40-234:~

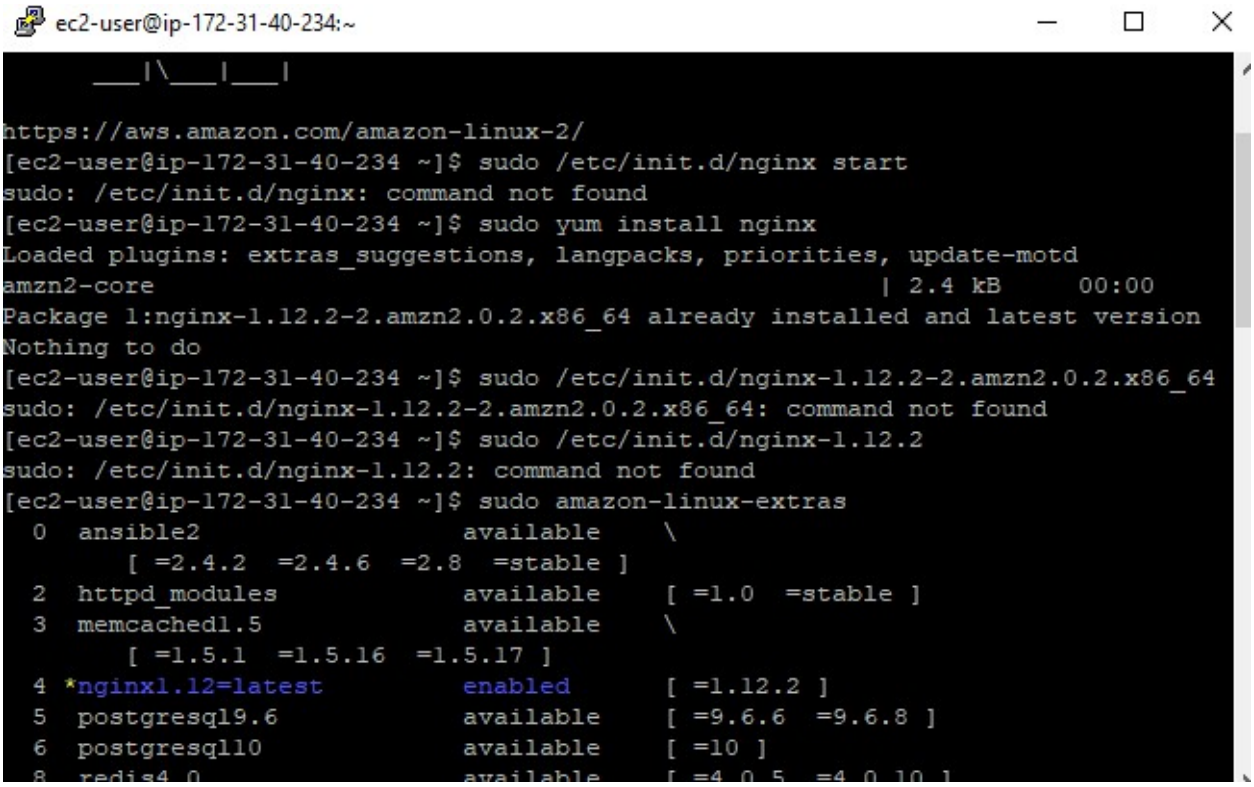
```
 42  php7.4                   available    [ =stable ]
[root@ip-172-31-40-234 ~]# sudo amazon-linux-extras install nginx1.12
Topic nginx1.12 has end-of-support date of 2019-09-20
Installing nginx
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker amzn2extra-nginx1.12
12 metadata files removed
4 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core                                               | 2.4 kB     00:00
amzn2extra-docker                                        | 1.8 kB     00:00
amzn2extra-nginx1.12                                     | 1.3 kB     00:00
(1/6): amzn2-core/2/x86_64/group_gz                        | 2.5 kB   00:00
(2/6): amzn2-core/2/x86_64/updateinfo                      | 199 kB   00:00
(3/6): amzn2extra-docker/2/x86_64/updateinfo               |   69 B   00:00
(4/6): amzn2extra-docker/2/x86_64/primary_db               |  64 kB   00:00
(5/6): amzn2extra-nginx1.12/2/x86_64/primary_db            |  23 kB   00:00
(6/6): amzn2-core/2/x86_64/primary_db                      |  39 MB   00:00
Resolving Dependencies
--> Running transaction check
---> Package nginx.x86_64 1:1.12.2-2.amzn2.0.2 will be installed
--> Processing Dependency: nginx-filesystem = 1:1.12.2-2.amzn2.0.2 for package:
1:nginx-1.12.2-2.amzn2.0.2.x86_64
--> Processing Dependency: nginx-all-modules = 1:1.12.2-2.amzn2.0.2 for package:
 1:nginx-1.12.2-2.amzn2.0.2.x86_64
--> Processing Dependency: nginx-filesystem for package: 1:nginx-1.12.2-2.amzn2.
0.2.x86_64
--> Processing Dependency: libprofiler.so.0()(64bit) for package: 1:nginx-1.12.2
-2.amzn2.0.2.x86_64
--> Running transaction check
---> Package gperftools-libs.x86_64 0:2.6.1-1.amzn2 will be installed
---> Package nginx-all-modules.noarch 1:1.12.2-2.amzn2.0.2 will be installed
--> Processing Dependency: nginx-mod-stream = 1:1.12.2-2.amzn2.0.2 for package:
1:nginx-all-modules-1.12.2-2.amzn2.0.2.noarch
--> Processing Dependency: nginx-mod-mail = 1:1.12.2-2.amzn2.0.2 for package: 1:
nginx-all-modules-1.12.2-2.amzn2.0.2.noarch
--> Processing Dependency: nginx-mod-http-xslt-filter = 1:1.12.2-2.amzn2.0.2 for
 package: 1:nginx-all-modules-1.12.2-2.amzn2.0.2.noarch
--> Processing Dependency: nginx-mod-http-perl = 1:1.12.2-2.amzn2.0.2 for packag
e: 1:nginx-all-modules-1.12.2-2.amzn2.0.2.noarch
--> Processing Dependency: nginx-mod-http-image-filter = 1:1.12.2-2.amzn2.0.2 fo
r package: 1:nginx-all-modules-1.12.2-2.amzn2.0.2.noarch
--> Processing Dependency: nginx-mod-http-geoip = 1:1.12.2-2.amzn2.0.2 for packa
```

```
Dependencies Resolved

================================================================================
 Package                 Arch   Version              Repository            Size
================================================================================
Installing:
 nginx                   x86_64 1:1.12.2-2.amzn2.0.2 amzn2extra-nginx1.12 533 k
Installing for dependencies:
 fontconfig              x86_64 2.10.95-11.amzn2.0.2 amzn2-core           231 k
 fontpackages-filesystem noarch 1.44-8.amzn2         amzn2-core            10 k
 gd                      x86_64 2.0.35-26.amzn2.0.2  amzn2-core           147 k
 gperftools-libs         x86_64 2.6.1-1.amzn2        amzn2-core           274 k
 libX11                  x86_64 1.6.5-2.amzn2.0.2    amzn2-core           614 k
 libX11-common           noarch 1.6.5-2.amzn2.0.2    amzn2-core           164 k
 libXau                  x86_64 1.0.8-2.1.amzn2.0.2  amzn2-core            29 k
 libXpm                  x86_64 3.5.12-1.amzn2.0.2   amzn2-core            57 k
 libpng                  x86_64 2:1.5.13-7.amzn2.0.2 amzn2-core           214 k
 libxcb                  x86_64 1.12-1.amzn2.0.2     amzn2-core           216 k
 libxslt                 x86_64 1.1.28-5.amzn2.0.2   amzn2-core           243 k
 nginx-all-modules       noarch 1:1.12.2-2.amzn2.0.2 amzn2extra-nginx1.12  17 k
 nginx-filesystem        noarch 1:1.12.2-2.amzn2.0.2 amzn2extra-nginx1.12  17 k
 nginx-mod-http-geoip    x86_64 1:1.12.2-2.amzn2.0.2 amzn2extra-nginx1.12  24 k
 nginx-mod-http-image-filter
                         x86_64 1:1.12.2-2.amzn2.0.2 amzn2extra-nginx1.12  27 k
 nginx-mod-http-perl     x86_64 1:1.12.2-2.amzn2.0.2 amzn2extra-nginx1.12  37 k
 nginx-mod-http-xslt-filter
                         x86_64 1:1.12.2-2.amzn2.0.2 amzn2extra-nginx1.12  26 k
 nginx-mod-mail          x86_64 1:1.12.2-2.amzn2.0.2 amzn2extra-nginx1.12  55 k
 nginx-mod-stream        x86_64 1:1.12.2-2.amzn2.0.2 amzn2extra-nginx1.12  76 k
 stix-fonts              noarch 1.1.0-5.amzn2        amzn2-core           1.3 M

Transaction Summary
================================================================================
Install  1 Package (+20 Dependent packages)

Total download size: 4.2 M
Installed size: 11 M
Is this ok [y/d/N]:
```

*sudo yum install command was not working for my system.

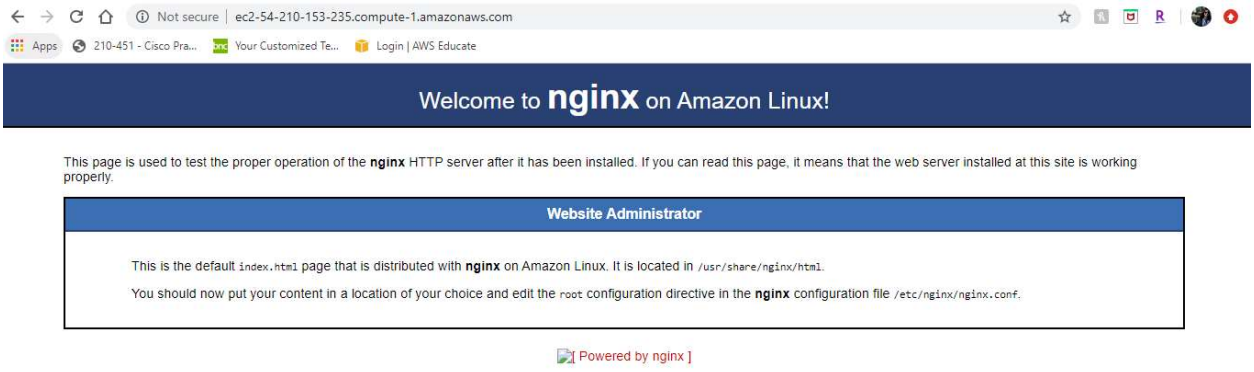
Step 8:

The following screen shows that nginx is running on Load Balancer machine



On putting public IP Address of LoadBalancer as URL:

So nginx is running successfully on LoadBalancer.



3) Change loadbalancer config file:

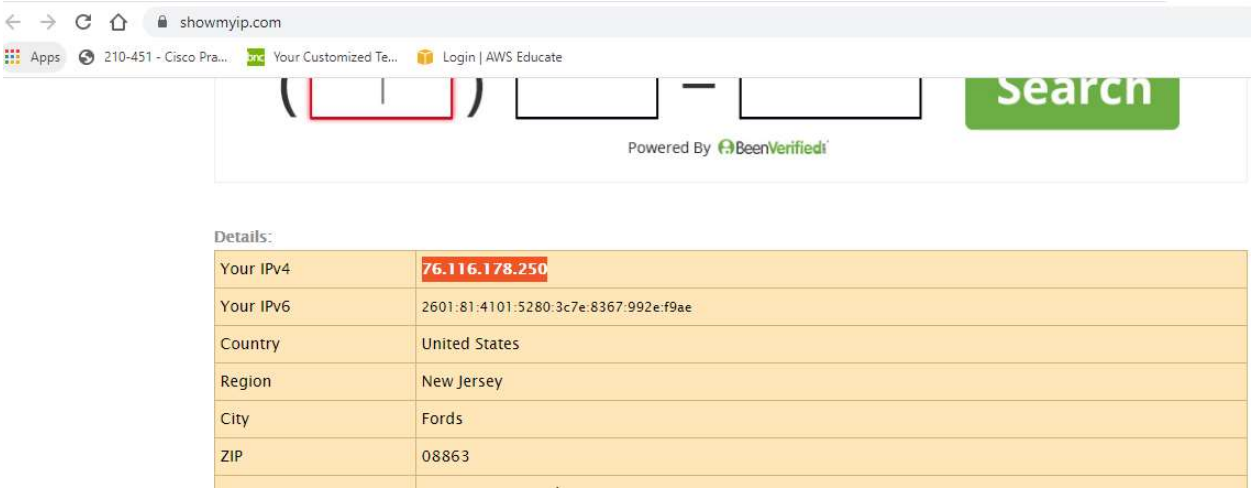
Step 1:

Editing the configuration file on LoadBalancer and viewing it:

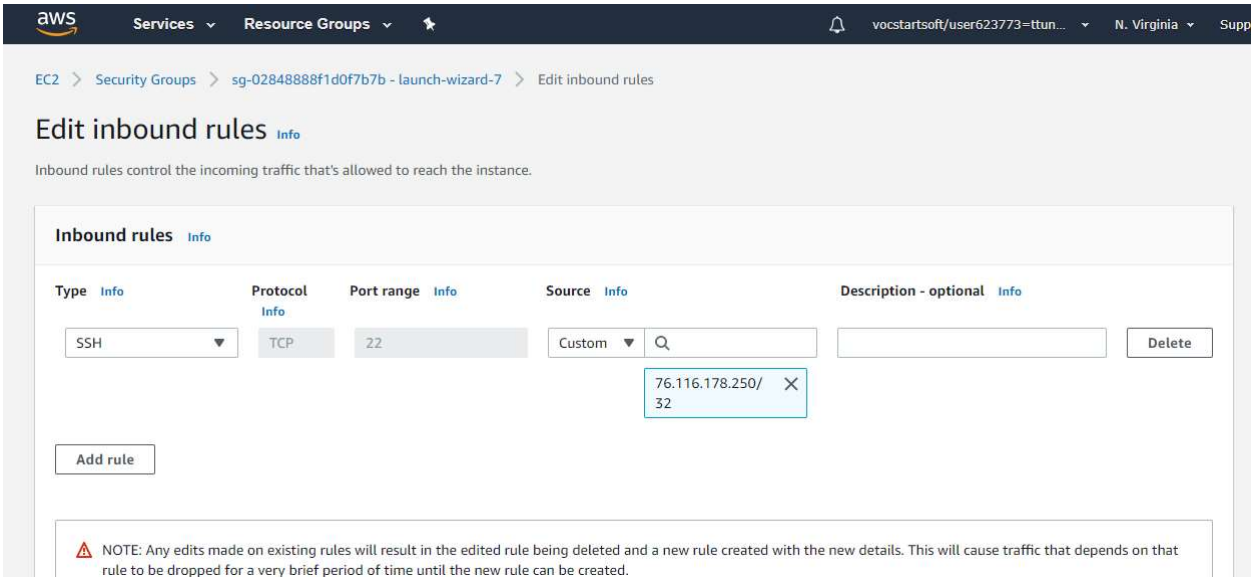
Step 9:

Since I put my machine address as 10.0.0.13/32; which was not my public IP Address; I was not able to connect to any of the servers.

So now my public IPAddress is:



I changed this by going in the security group of my server 1 , server 2, server 3 and server 4 and now I can connect successfully to my ec2 instance! Now only my machine will be able to access these servers.



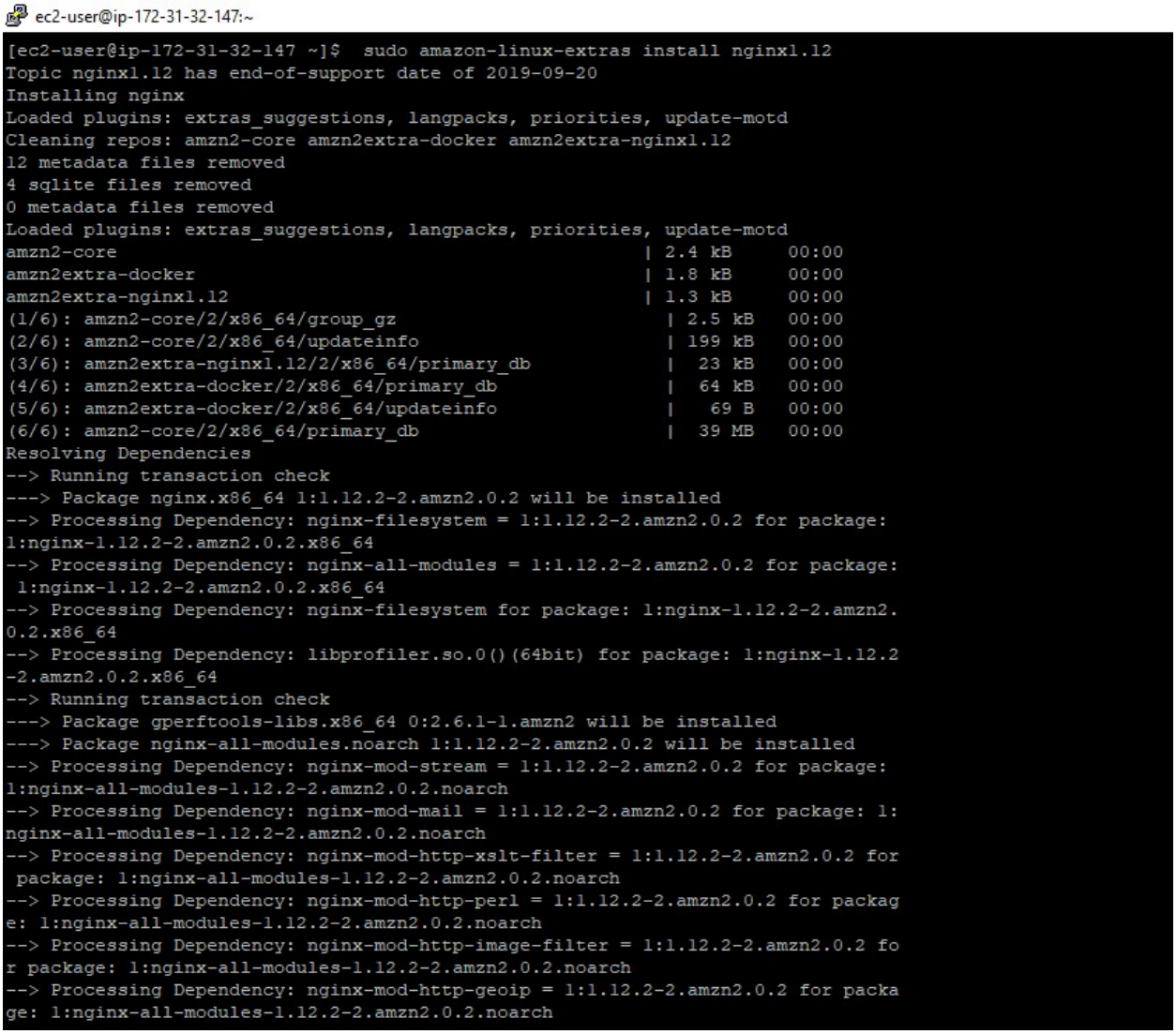
2) Install Nginx on each instance:

Step 1:

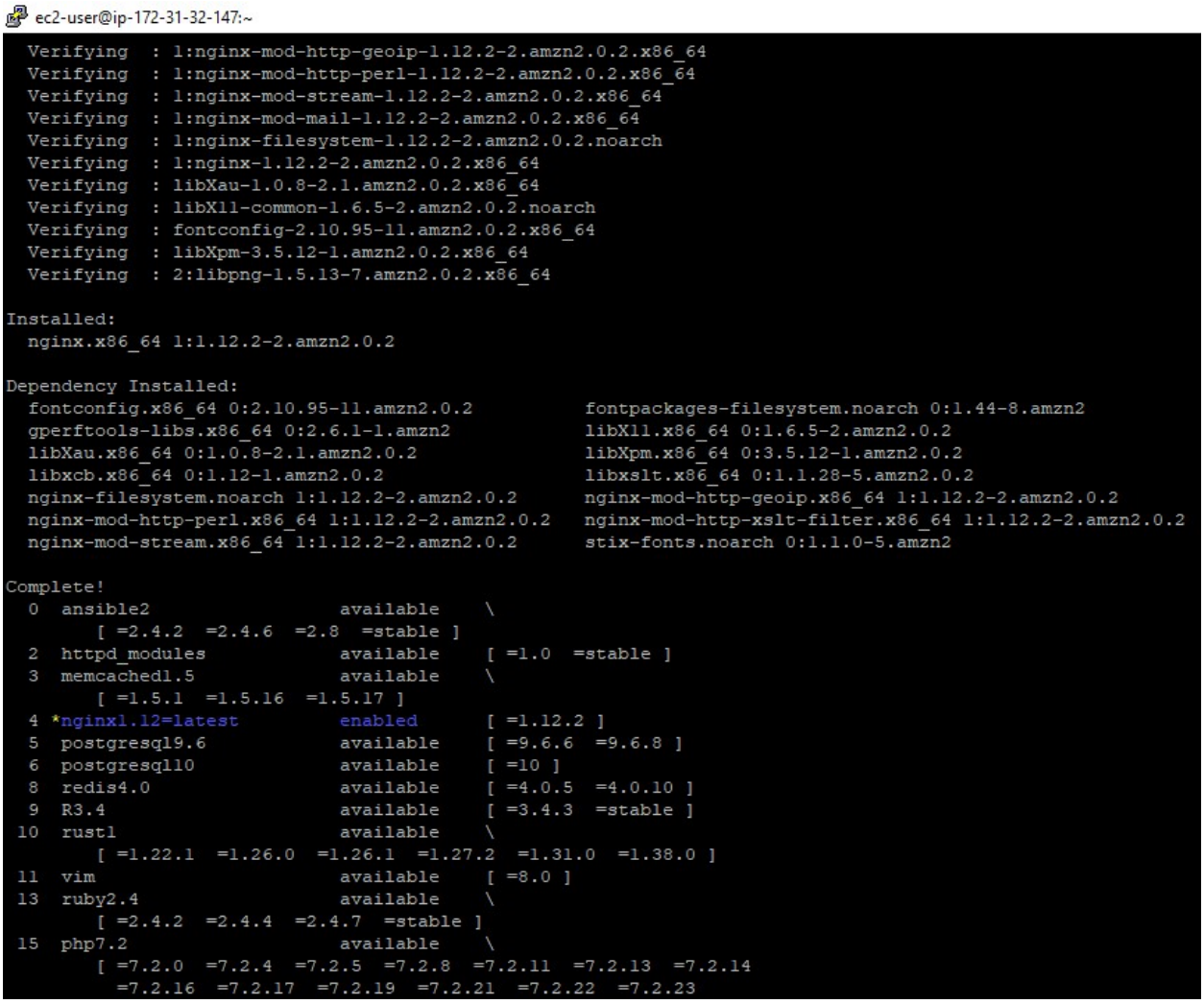
Install nginx on servers 1 to 4 using command **sudo amazon-linux-extras install nginx1.12**

Like the one used in step 8. Now all servers have nginx installed and enabled.

Example for server 1:



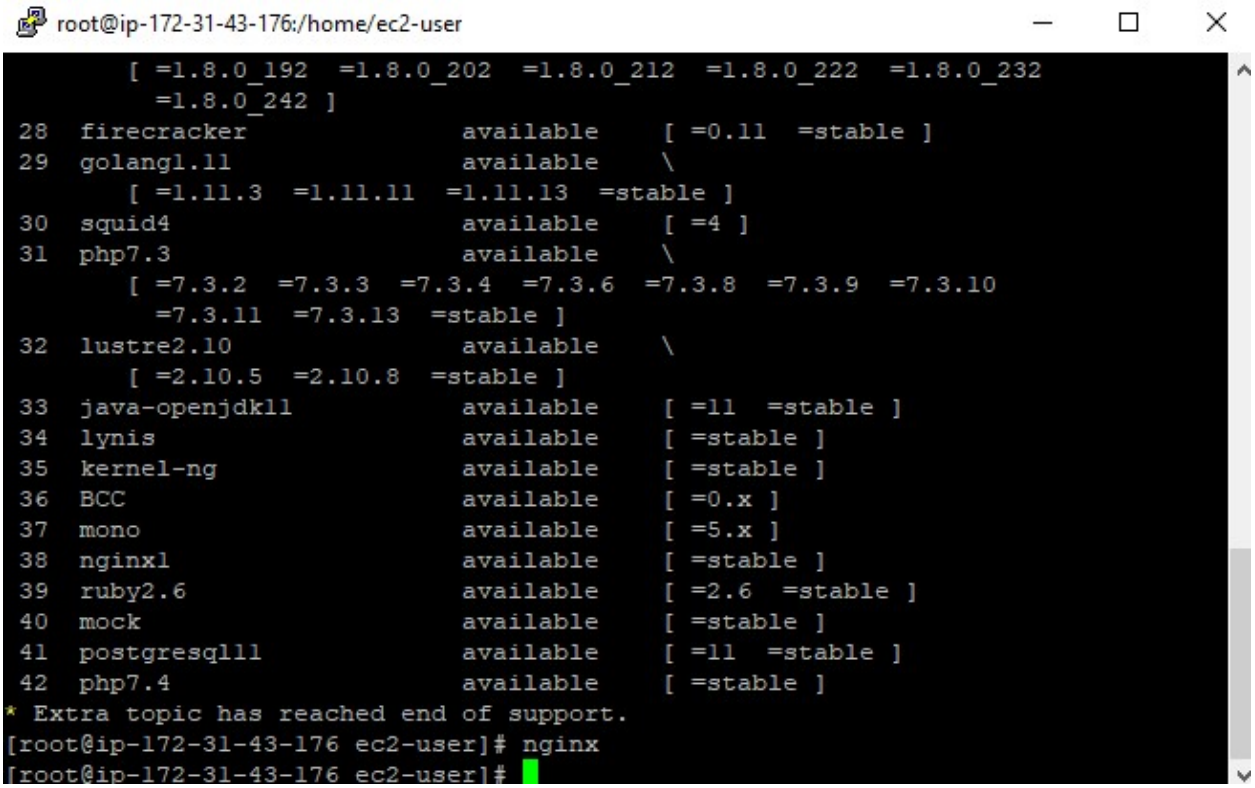
This below screen shows it is enabled/running:

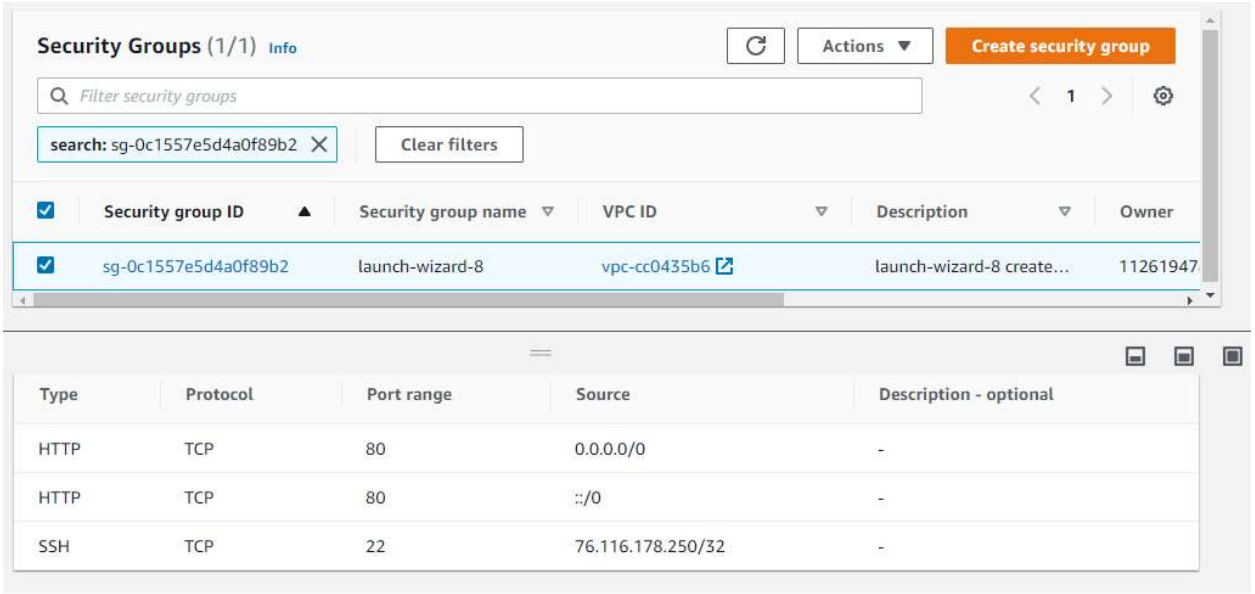


Testing if nginx has started on server 1:

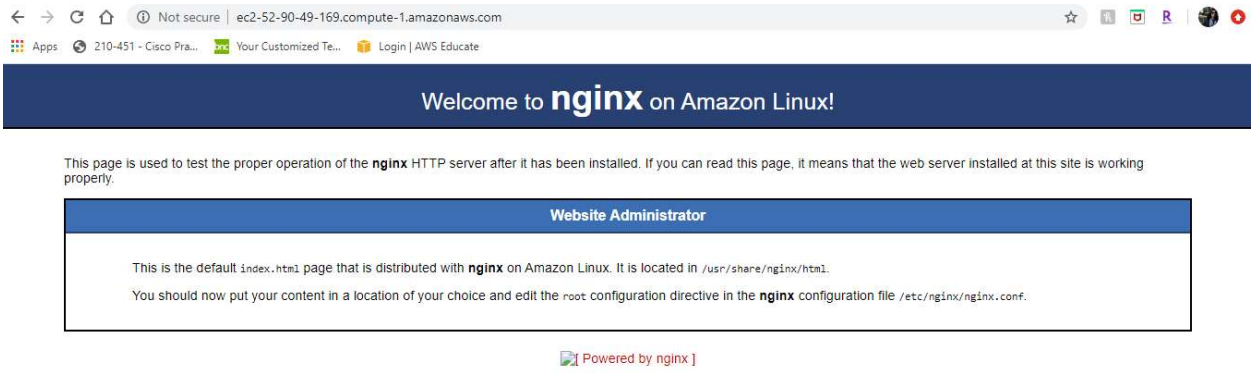
In browser, type http:// ec2-52-91-113-8.compute-1.amazonaws.com

In case of server 1, ec2-52-91-113-8.compute-1.amazonaws.com is the public DNS.





For some reason in my machine, in order to start nginx I had to go to root using sudo su command and then type command nginx and then only the web-page loaded! Also, I made sure to add port 80 in inbound security rules for my instances.



Repeat this step for all servers 1,2,3, and 4. And make sure nginx is working by inputting public DNS in the browser for each one of them!

Step 2:

Now index.html has to be edited for Server1:

```
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Sat Apr  4 19:24:05 2020 from 76.116.178.250

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

https://aws.amazon.com/amazon-linux-2/
1 package(s) needed for security, out of 7 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-32-147 ~]$ sudo su
[root@ip-172-31-32-147 ec2-user]# vi /usr/share/nginx/html/index.html
[root@ip-172-31-32-147 ec2-user]# ^C
[root@ip-172-31-32-147 ec2-user]# vi /usr/share/nginx/html/index.html
[root@ip-172-31-32-147 ec2-user]#  vi /usr/share/nginx/html/index.html
[root@ip-172-31-32-147 ec2-user]# ~
bash: /root: Is a directory
[root@ip-172-31-32-147 ec2-user]#  vi /usr/share/nginx/html/index.html
[root@ip-172-31-32-147 ec2-user]# vi /usr/share/nginx/html/index.html
[root@ip-172-31-32-147 ec2-user]# vi /usr/share/nginx/html/index.html
[root@ip-172-31-32-147 ec2-user]# vi /usr/share/nginx/html/index.html
[root@ip-172-31-32-147 ec2-user]#
```

Write permissions were not given in ec2user mode so I switched to root mode:

Edited the index.html file to include Server 1 header in h1 tag:

root@ip-172-31-32-147:/home/ec2-user

```
            hr {
                display: none;
            }
            .content {
                padding: 1em 5em;
            }
            .alert {
                border: 2px solid #000;
            }

            img {
                border: 2px solid #fff;
                padding: 2px;
                margin: 2px;
            }
            a:hover img {
                border: 2px solid #294172;
            }
            .logos {
                margin: 1em;
                text-align: center;
            }
            /*]]>*/
        </style>
    </head>

    <body>
        <h1>Welcome to <strong>My Server 1 </strong> on Amazon Linux!</h1>

<header><h1>Server1</h1></header>
        <div class="content">
            <p>This is the first server's page named
            <strong>Server 1</strong>Congratulations! Our server 1 is running successfully!</p>

            <div class="alert">
                <h2>Tehreem Tungekar</h2>
                <div class="content">
                    <p>This is the default <tt>index.html</tt> page that
                    is distributed with <strong>nginx</strong> on
                     Amazon Linux.  It is located in
                    <tt>/usr/share/nginx/html</tt>.</p>

                    <p>You should now put your content in a location of
```

Not secure | ec2-52-91-113-8.compute-1.amazonaws.com

Apps | 210-451 - Cisco Pra... | Your Customized Te... | Login | AWS Educate

Welcome to **My Server 1** on Amazon Linux!

Server1

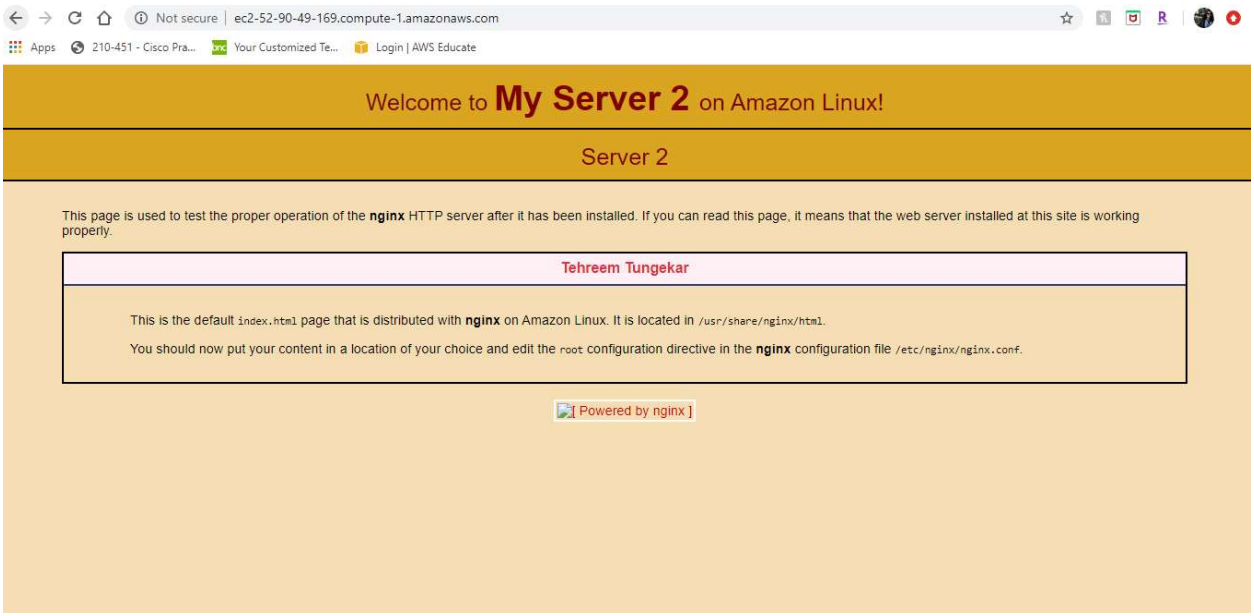
This is the first server's page named **Server 1**Congratulations! Our server 1 is running successfully!

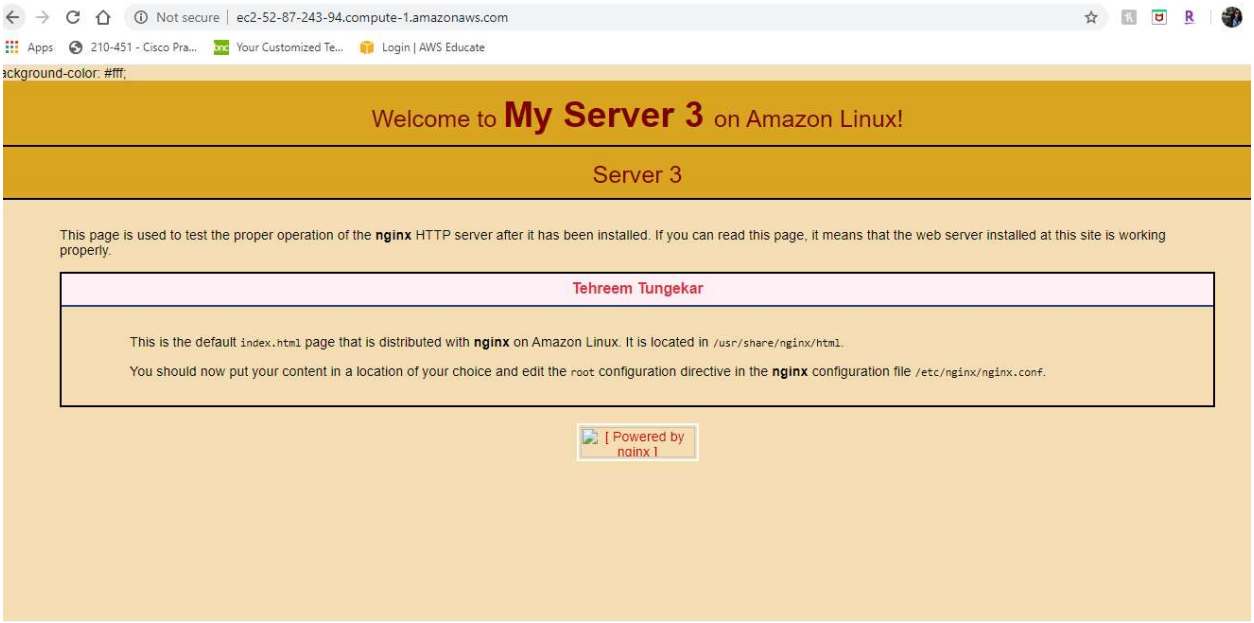
Tehreem Tungekar

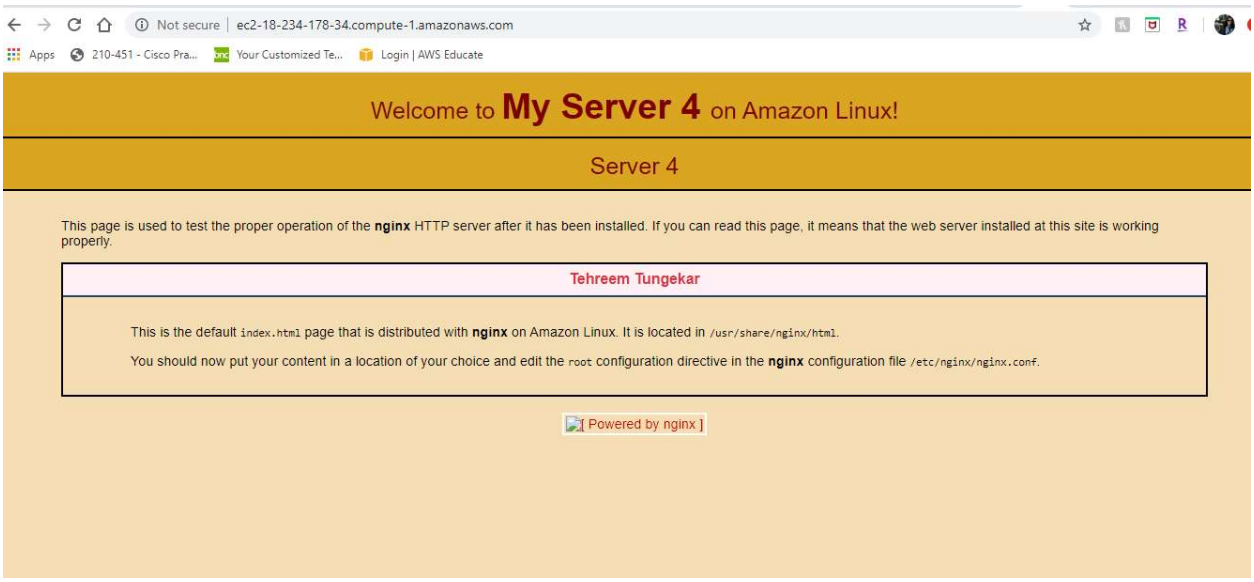
This is the default `index.html` page that is distributed with **nginx** on Amazon Linux. It is located in `/usr/share/nginx/html`.

You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.

[Powered by nginx]


Not secure | ec2-52-90-49-169.compute-1.amazonaws.com
Apps
210-451 - Cisco Pra...
Your Customized Te...
Login | AWS Educate
Welcome to My Server 2 on Amazon Linux!
Server 2
This page is used to test the proper operation of the nginx HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly.
Tehreem Tungekar
This is the default index.html page that is distributed with nginx on Amazon Linux. It is located in /usr/share/nginx/html.
You should now put your content in a location of your choice and edit the root configuration directive in the nginx configuration file /etc/nginx/nginx.conf.
[Powered by nginx]


Not secure | ec2-52-87-243-94.compute-1.amazonaws.com
Apps
210-451 - Cisco Pra...
Your Customized Te...
Login | AWS Educate
ackground-color: #fff;
Welcome to My Server 3 on Amazon Linux!
Server 3
This page is used to test the proper operation of the nginx HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly.
Tehreem Tungekar
This is the default index.html page that is distributed with nginx on Amazon Linux. It is located in /usr/share/nginx/html.
You should now put your content in a location of your choice and edit the root configuration directive in the nginx configuration file /etc/nginx/nginx.conf.
[Powered by nginx]

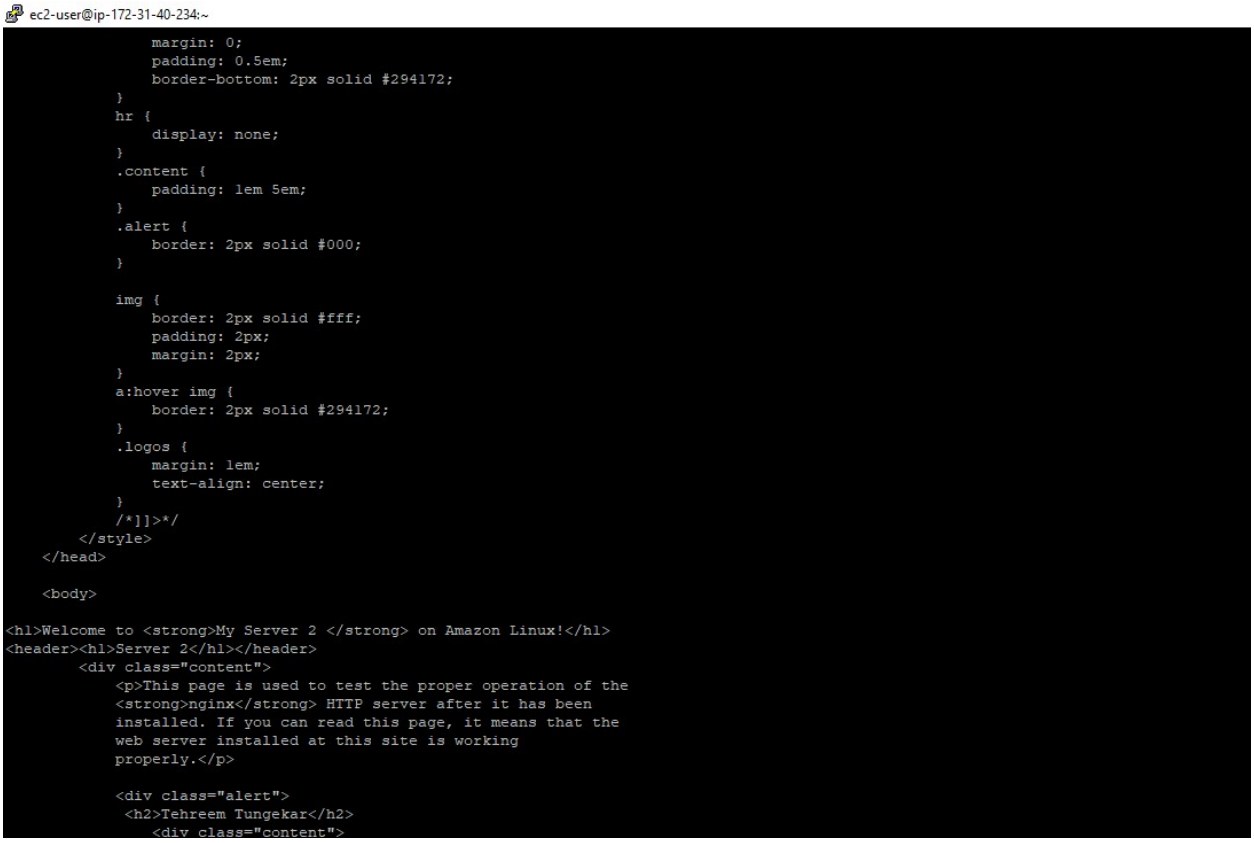


Step 3:

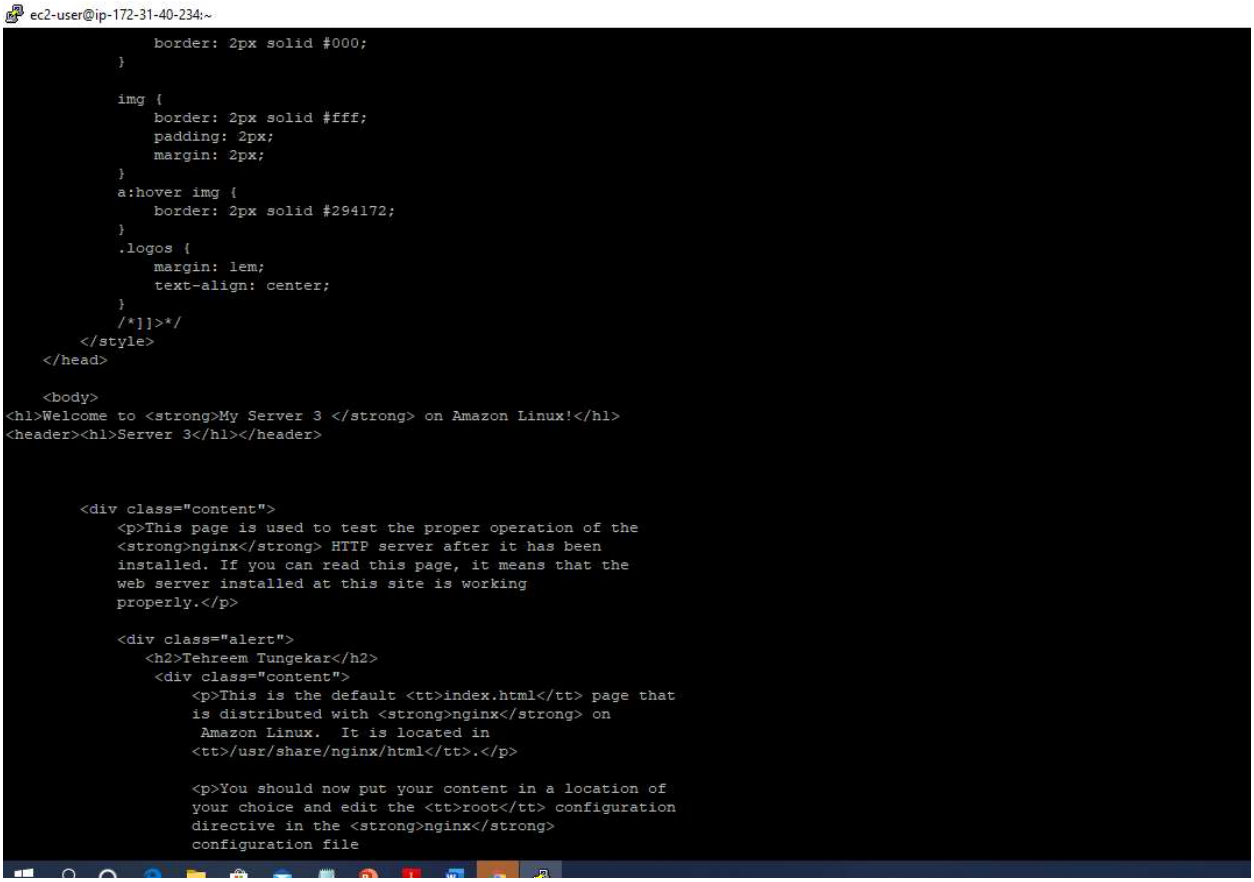
Using $curl dnsnameofloadbalancer:

```
ec2-user@ip-172-31-40-234:~
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-40-234 ~]$ curl ec2-54-210-153-235.compute-1.amazonaws.com
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/
DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>Test Page for the Nginx HTTP Server on Amazon Linux</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <style type="text/css">
            /*<![CDATA[*/
            body {
 background-color: wheat;

                font-size: 0.9em;
                font-family: sans-serif,helvetica;
                margin: 0;
                padding: 0;
            }
            :link {
                color: #c00;
            }
            :visited {
                color: #c00;
            }
            a:hover {
                color: #f50;
            }
            h1 {
                text-align: center;
                margin: 0;
                padding: 0.6em 2em 0.4em;
               background-color:goldenrod;
               color:maroon;

                font-weight: normal;
                font-size: 1.75em;
                border-bottom: 2px solid #000;
            }
            h1 strong {
```

As it can be seen, first , load is sent to Server 2 .



Next time, load is sent to Server 3:



4) Collect information about visits to your site:

Step 1:

```
#!/usr/bin/env ruby
#
# This program is used for collecting web server visit information.
#
# Author: A. Genius
#
require 'optparse'
def print_usage
 puts "USAGE: visit_server -d DNS_NAME"
 exit
end
# add option switch and handler
options = {}
option_parser = OptionParser.new do |opts|
# DNS_NAME argument
 options[:dns_name] = nil
 opts.on('-d', '--dns-name DNS_NAME', 'Specify a DNS NAME') { |dns_name|
```

```
options[:dns_name] = dns_name }
# HELP argument
 options[:help] = nil
 opts.on('-h', '--help', 'Display usage') { |help| options[:help] = help }
end
option_parser.parse!
# verify arguments
if options[:dns_name] then
 dns_name = options[:dns_name]
else
 puts "Please set a balancer's DNS."
 print_usage
 exit
end
if options[:help] then
 print_usage
 exit
end
# Keep STDOUT
#orig_stdout = $stdout
# redirect stdout to /dev/null
#$stdout = File.new('/dev/null', 'w')
server1_visit_count = 0
server2_visit_count = 0
server3_visit_count = 0
server4_visit_count = 0
# starting to visit load balancing server
puts "Starting to visit load balancing server"
2000.times do
# visit load balancer
#o = `curl #{dns_name}`
o = `curl -s #{dns_name}`
if o =~ /server\s*1/i
server1_visit_count += 1
elsif o =~ /server\s*2/i
server2_visit_count += 1
elsif o =~ /server\s*3/i
server3_visit_count += 1
elsif o =~ /server\s*4/i
server4_visit_count += 1
end
print "."
end
# redirect output to stdout
#$stdout = orig_stdout
# print visit information
```

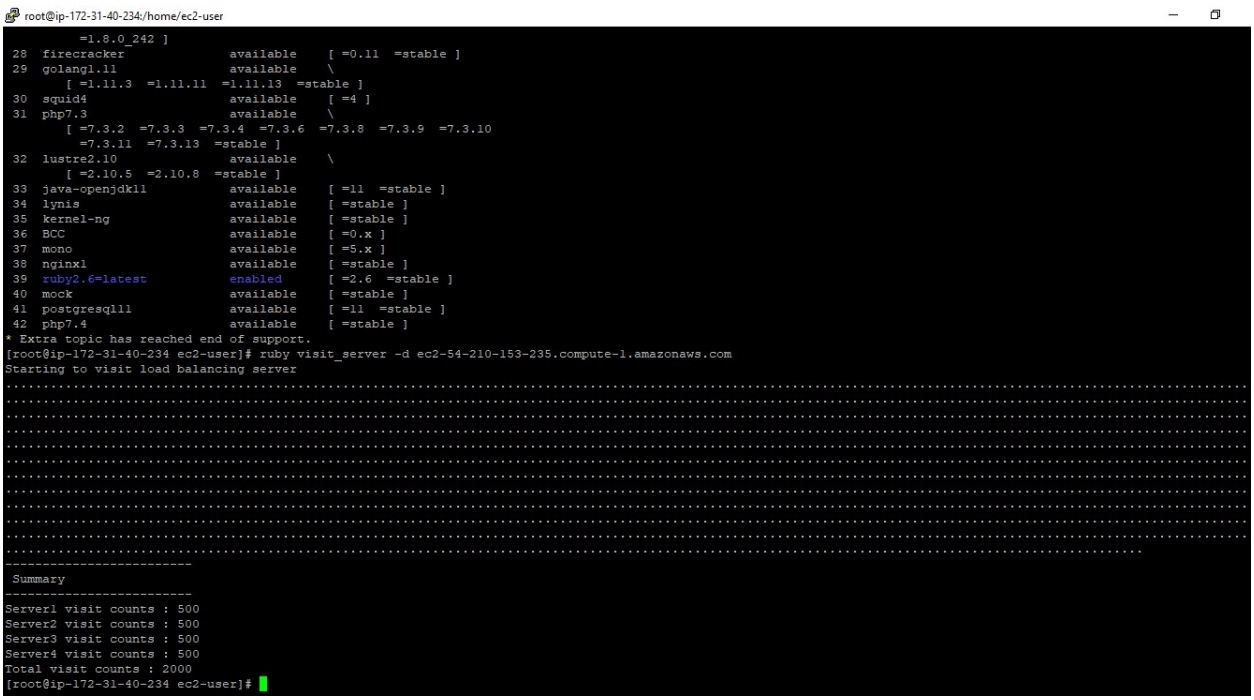
```
puts
puts '-------------------------'
puts ' Summary'
puts '-------------------------'
puts "Server1 visit counts : " + server1_visit_count.to_s
puts "Server2 visit counts : " + server2_visit_count.to_s
puts "Server3 visit counts : " + server3_visit_count.to_s
puts "Server4 visit counts : " + server4_visit_count.to_s
puts "Total visit counts : " + (server1_visit_count + server2_visit_count + server3_visit_count + server4_visit_count).to_s
```

Installed ruby 2.6, as it can be seen in the below screen and wrote a program using command $ vim visit_server and pasted the above highlighted code in it.

Scenario 1:

root@ip-172-31-40-234:/home/ec2-user

```
events {
    worker_connections 768;
}
http {
    upstream myapp {
     #ip_hash;
     server ec2-52-91-113-8.compute-1.amazonaws.com weight=1;
     server ec2-52-90-49-169.compute-1.amazonaws.com weight=1;
     server ec2-52-87-243-94.compute-1.amazonaws.com weight=1;
     server ec2-18-234-178-34.compute-1.amazonaws.com weight=1;
    }
    server {
        listen 80;
        server_name myapp.com;
        location / {
            proxy_pass http://myapp;
        }
    }
}
```

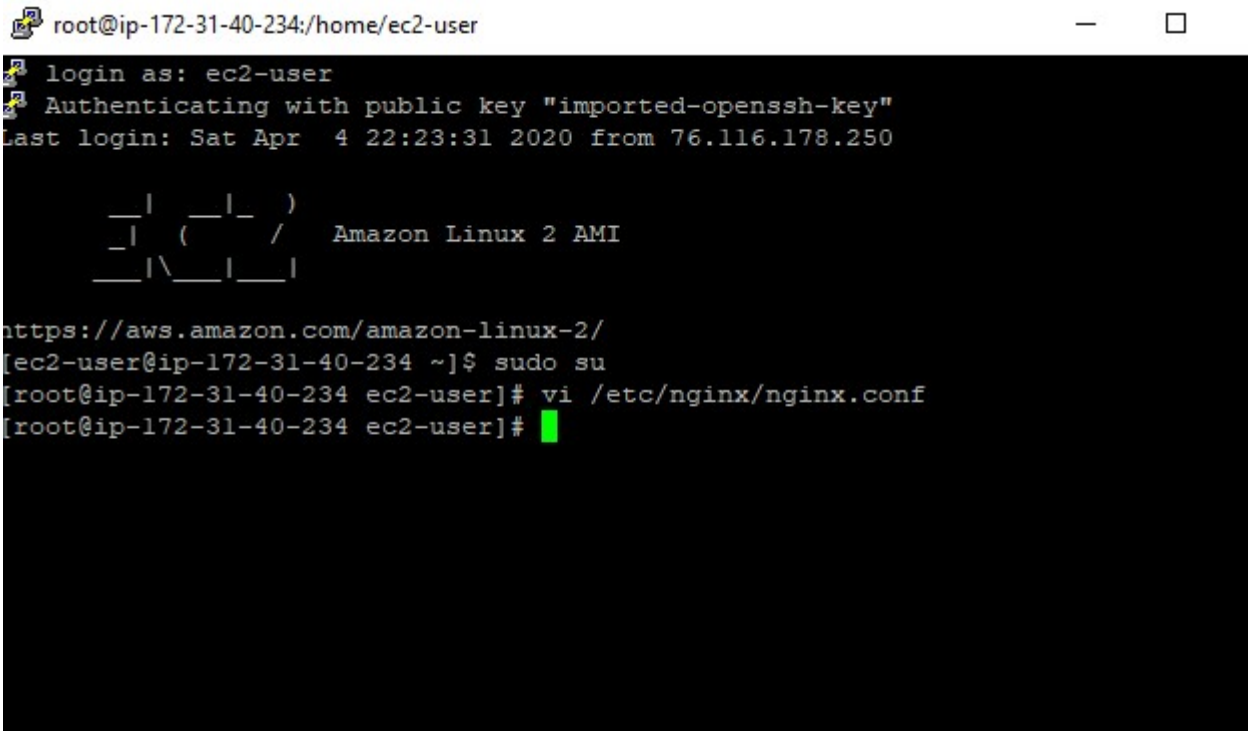


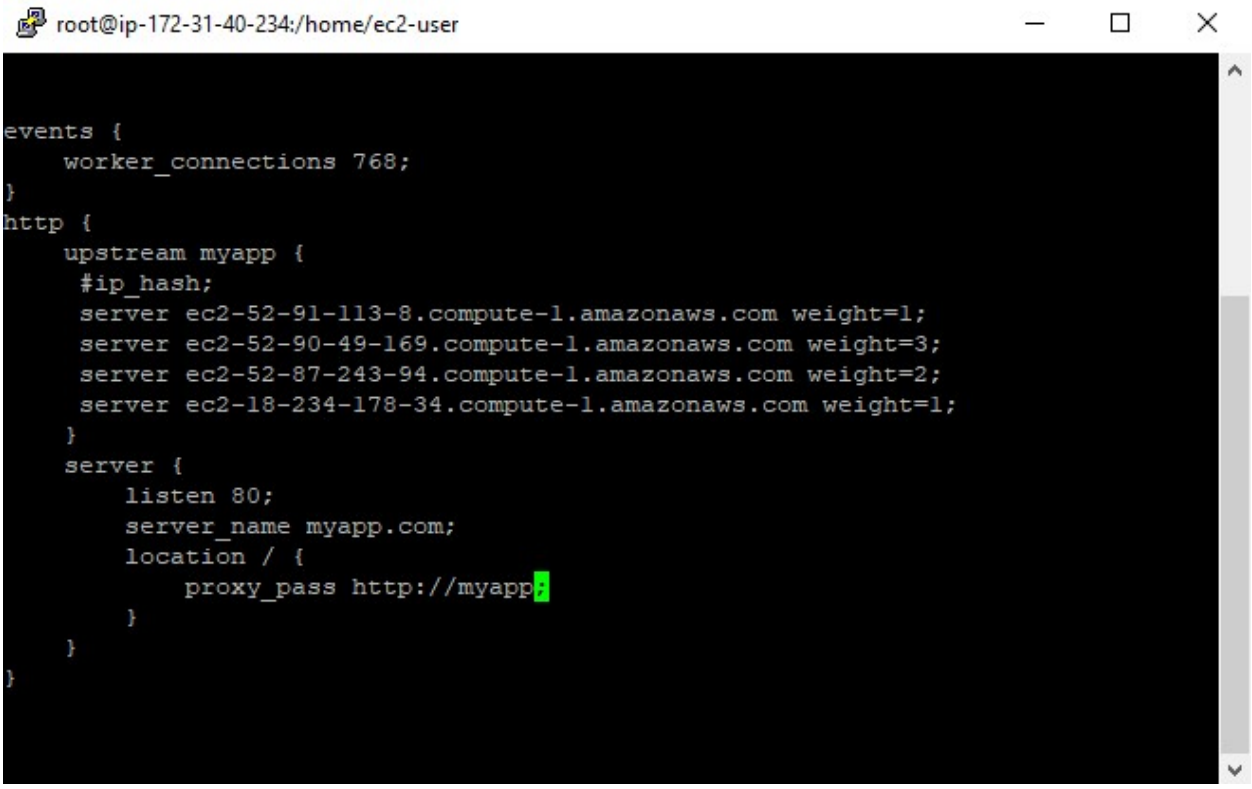
As it can be seen from the above screen, since in this scenario, equal weights are assigned, 1,1,1and 1 to all the servers, load is equally distributed amongst all the servers as 500 each.

Extra Scenario:

Now we need to change the configuration file of load balancer server in order to assign unequal weights to all servers and then we can see the output:

Firstly, modifying the conf file:





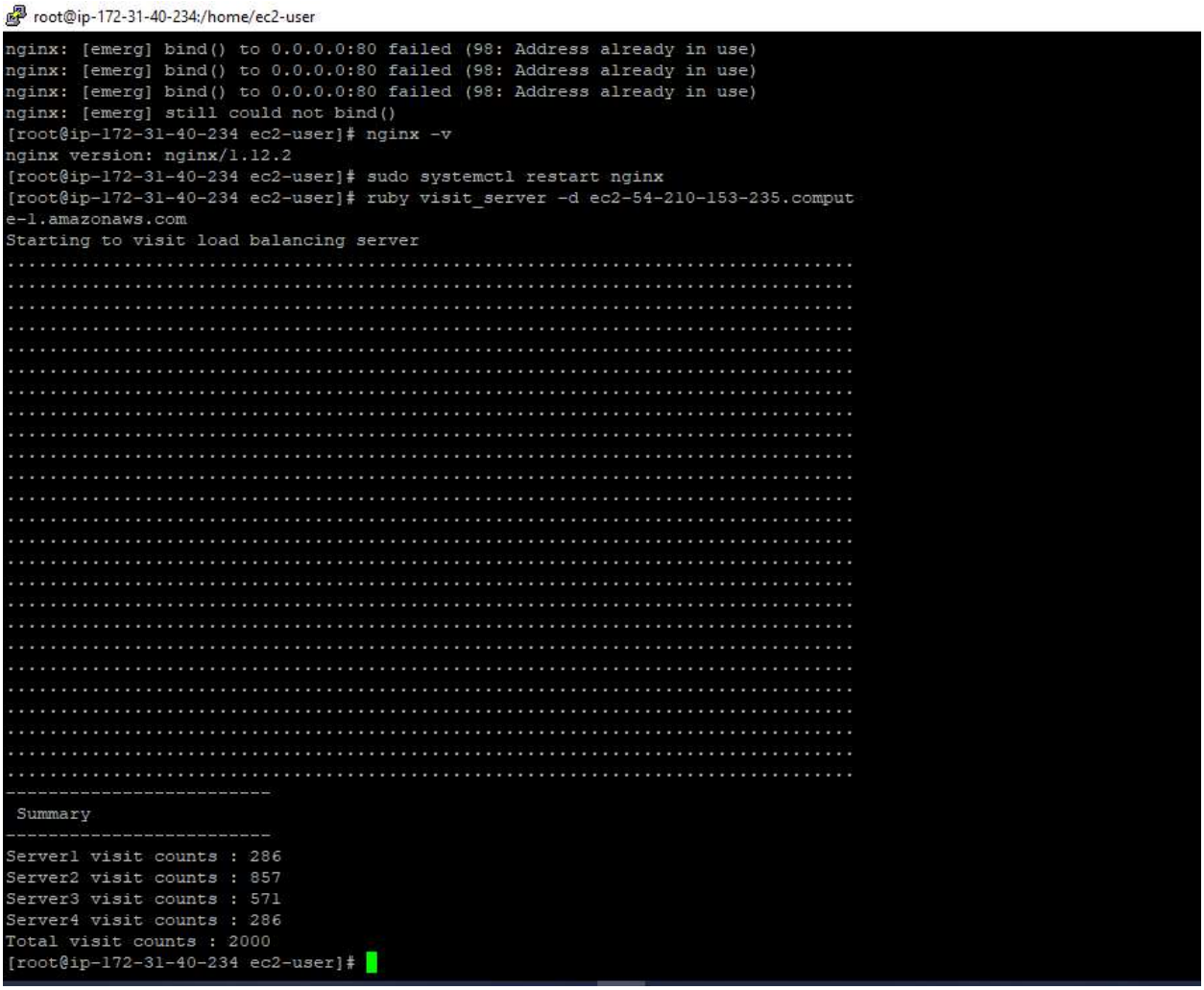
In this case, Server 1 and Server 4 are assigned weight 1, server 2 is assigned weight 3 and server 3 is assigned weight 2.

Now checking using visit server script: ruby visit_server -d ec2-54-210-153-235.compute-1.amazonaws.com

First, since the config file has been modified, restart the nginx service using command:

sudo systemctl restart nginx

Then check using ruby script:



Since the weight is unevenly distributed, the counts have also been unevenly distributed among the four servers.

Scenario 2:

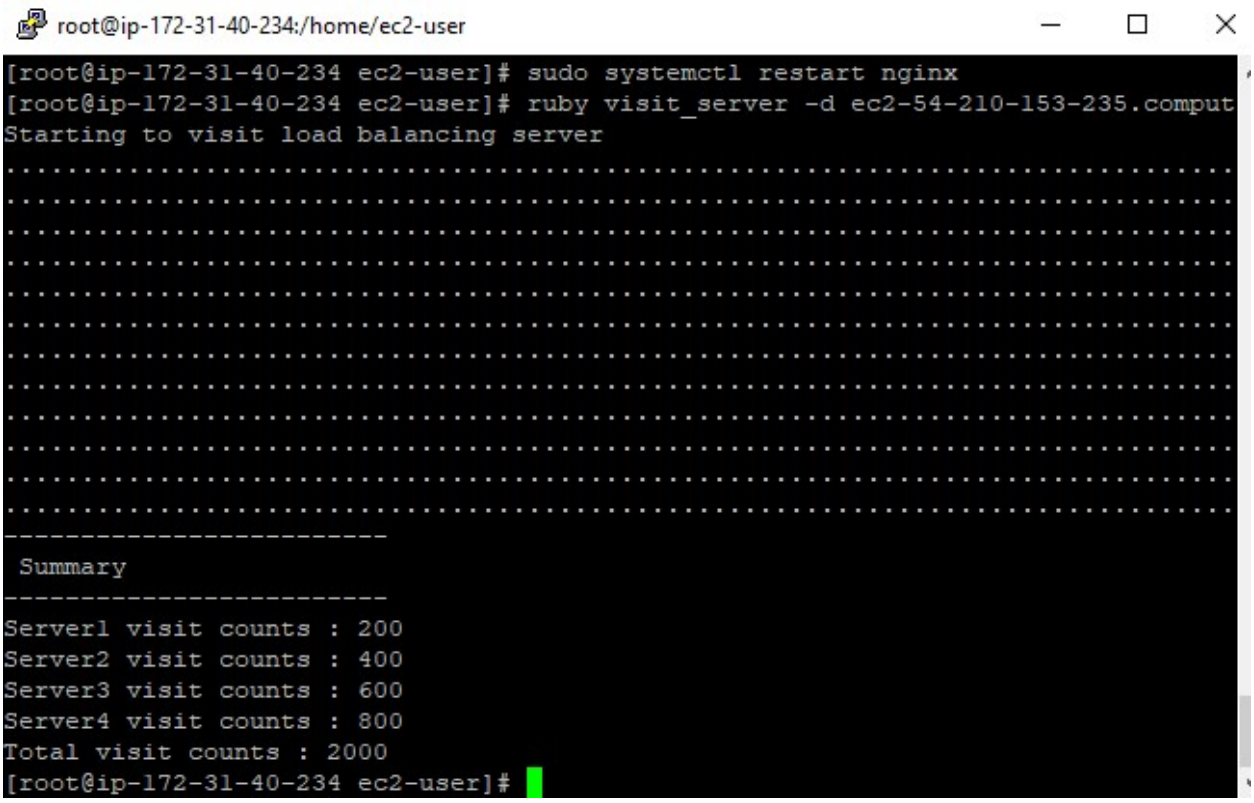
Using command: **vi /etc/nginx/nginx.conf**

Assign weight 1 to server 1, weight 2 to server 2, weight 3 to server 3 and weight 4 to server 4:

root@ip-172-31-40-234:/home/ec2-user

```
events {
    worker_connections 768;
}
http {
    upstream myapp {
     #ip_hash;
     server ec2-52-91-113-8.compute-1.amazonaws.com weight=1;
     server ec2-52-90-49-169.compute-1.amazonaws.com weight=2;
     server ec2-52-87-243-94.compute-1.amazonaws.com weight=3;
     server ec2-18-234-178-34.compute-1.amazonaws.com weight=4;
    }
    server {
        listen 80;
        server_name myapp.com;
        location / {
            proxy_pass http://myapp;
        }
    }
}
```

Restarting the service and checking the visits:



Since more weight is assigned to Server 4, it gets the highest counts, i.e. 800 counts, followed by server 3 getting 600 counts as it has weight 3; server 2 gets 600 counts as it has weight 2 and the least counts are given to server 1 as it has the minimum weight 1 assigned to it.

Scenario 3:

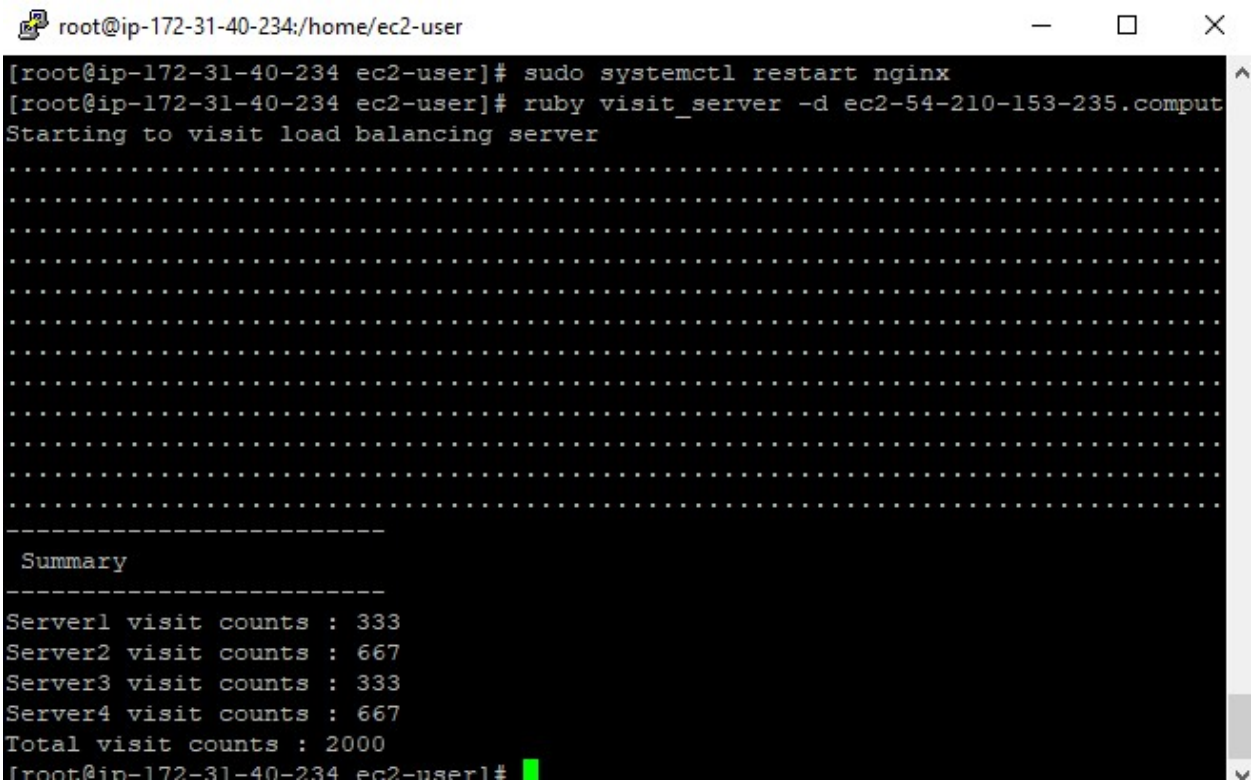
In this scenario, server 1 and server 3 get weight 1, server 2 and server 4 get weight 2.

root@ip-172-31-40-234:/home/ec2-user

```
events {
    worker_connections 768;
}
http {
    upstream myapp {
     #ip_hash;
     server ec2-52-91-113-8.compute-1.amazonaws.com weight=1;
     server ec2-52-90-49-169.compute-1.amazonaws.com weight=2;
     server ec2-52-87-243-94.compute-1.amazonaws.com weight=1;
     server ec2-18-234-178-34.compute-1.amazonaws.com weight=2;
    }
    server {
        listen 80;
        server_name myapp.com;
        location / {
            proxy_pass http://myapp;
        }
    }
}
```

Saving this file and restarting the nginx service.

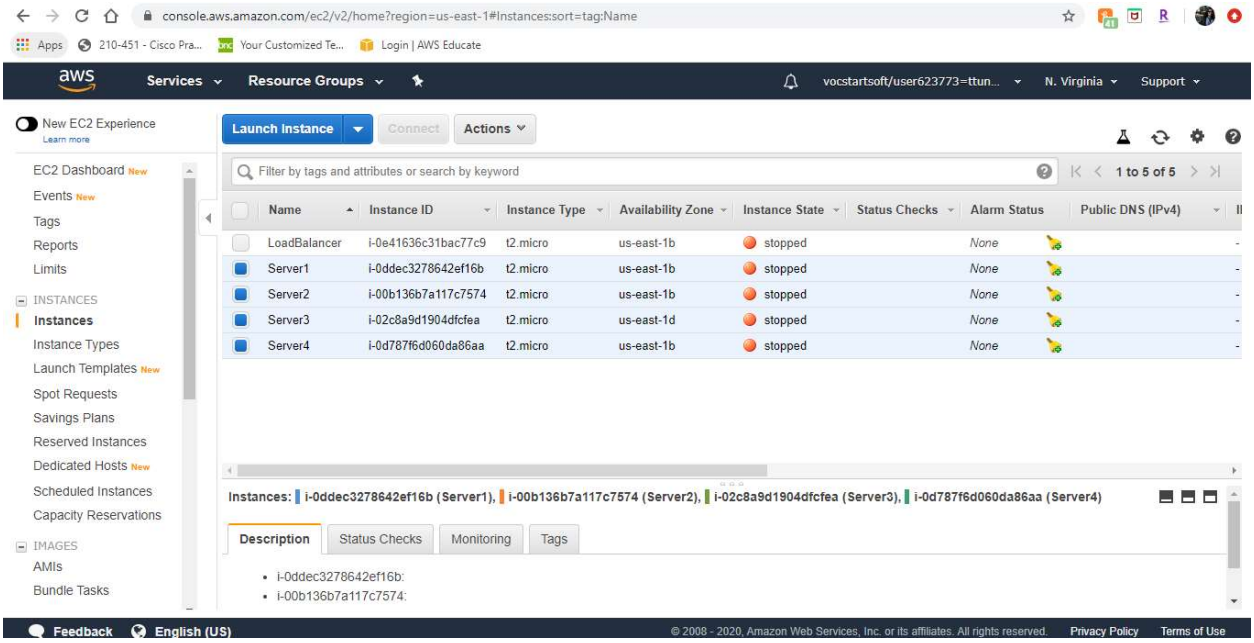
Visit server script shows:



As it can be seen, Server 1 and 3 which have a weight 1 assigned get 333 counts each. While servers 2 and 4 get double the count as compared to servers 1 and 3 because their weights are double.

Additional Steps:

Lastly, terminate all instances:

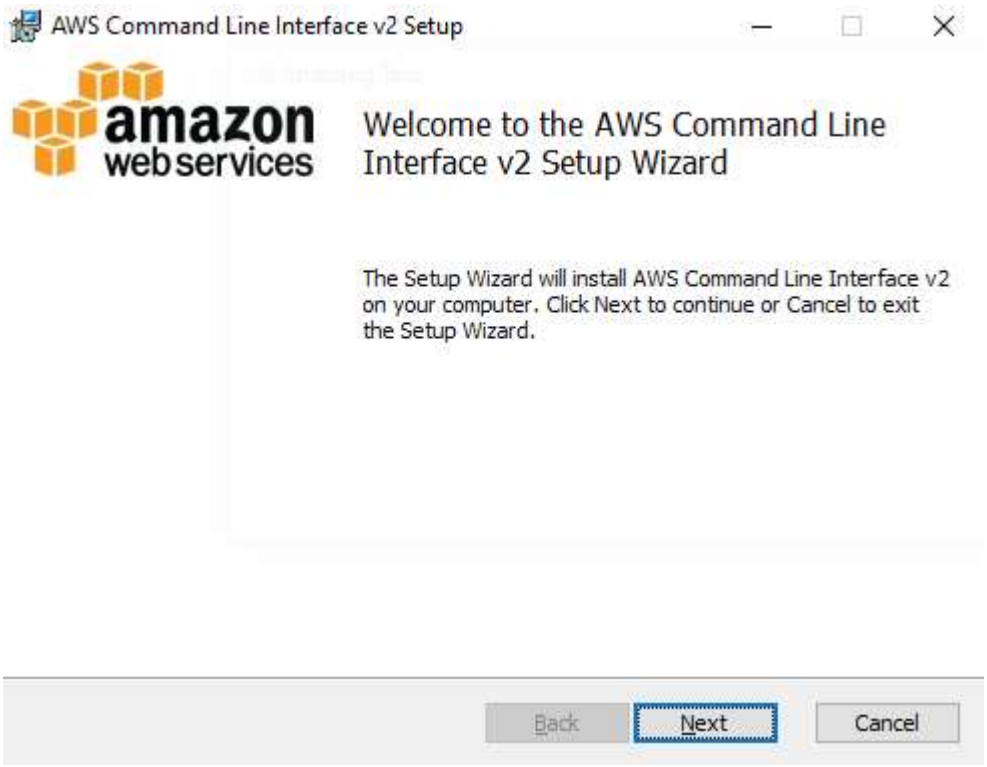


Steps for creating EC2 instance through CLI:

Step 1:

Download the AWS CLI MSI installer for Windows (64-bit) at https://awscli.amazonaws.com/AWSCLIV2.msi

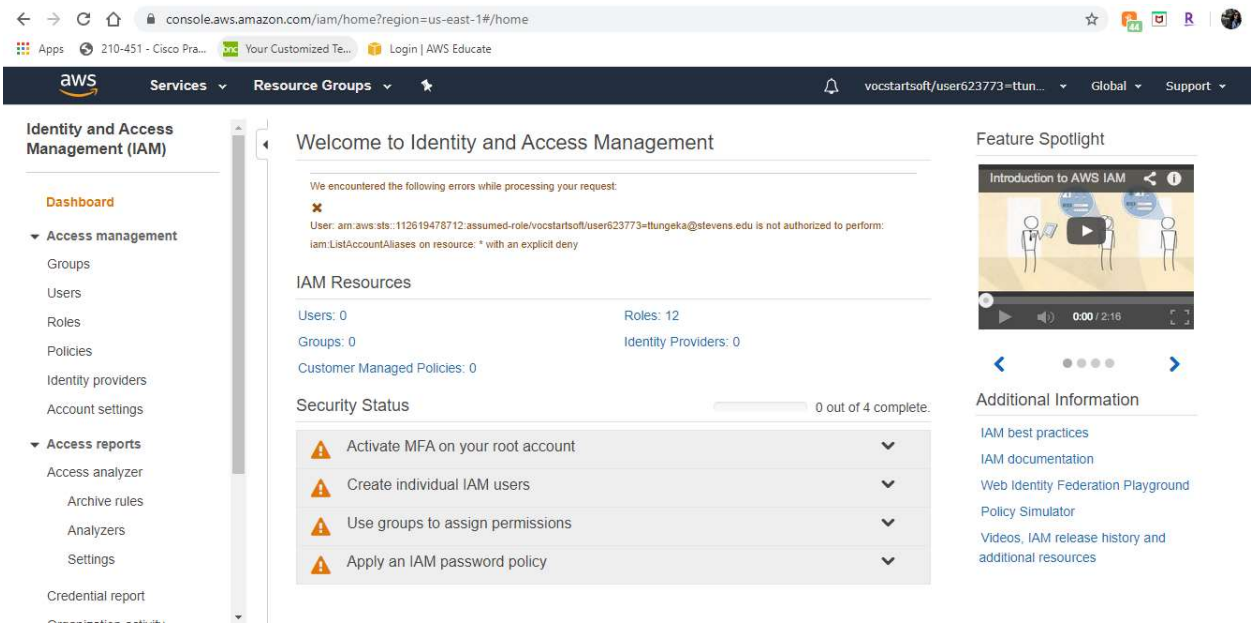
Run the downloaded file



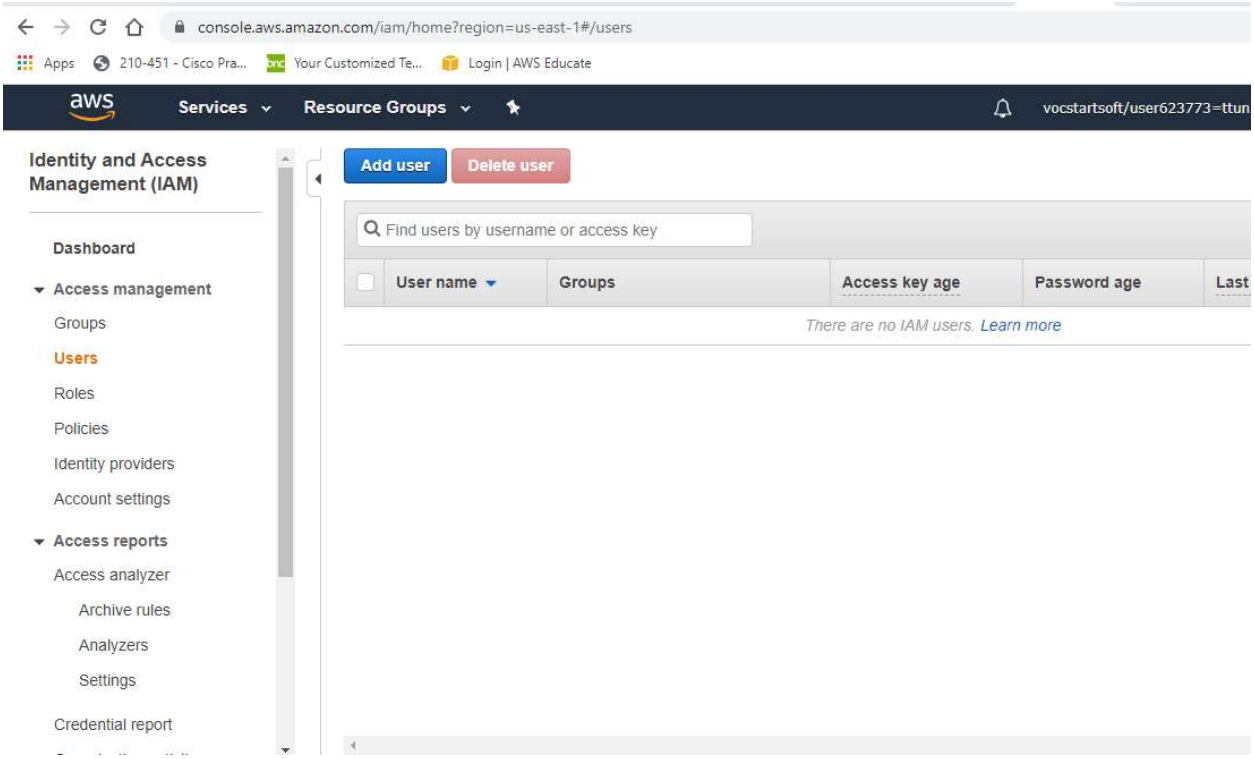
Click Next, Accept the terms, Check the Path and click Install.

Step 2: Creating IAM access key:

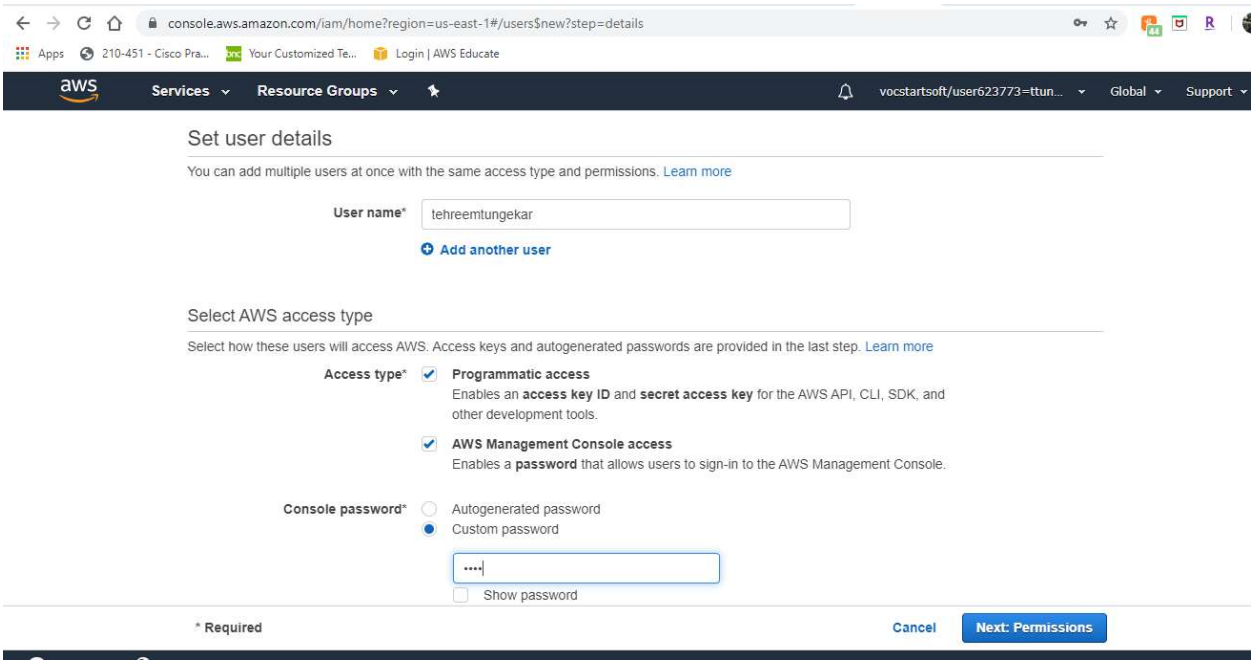
Search IAM in AWS Console :



Click Users in the left side:

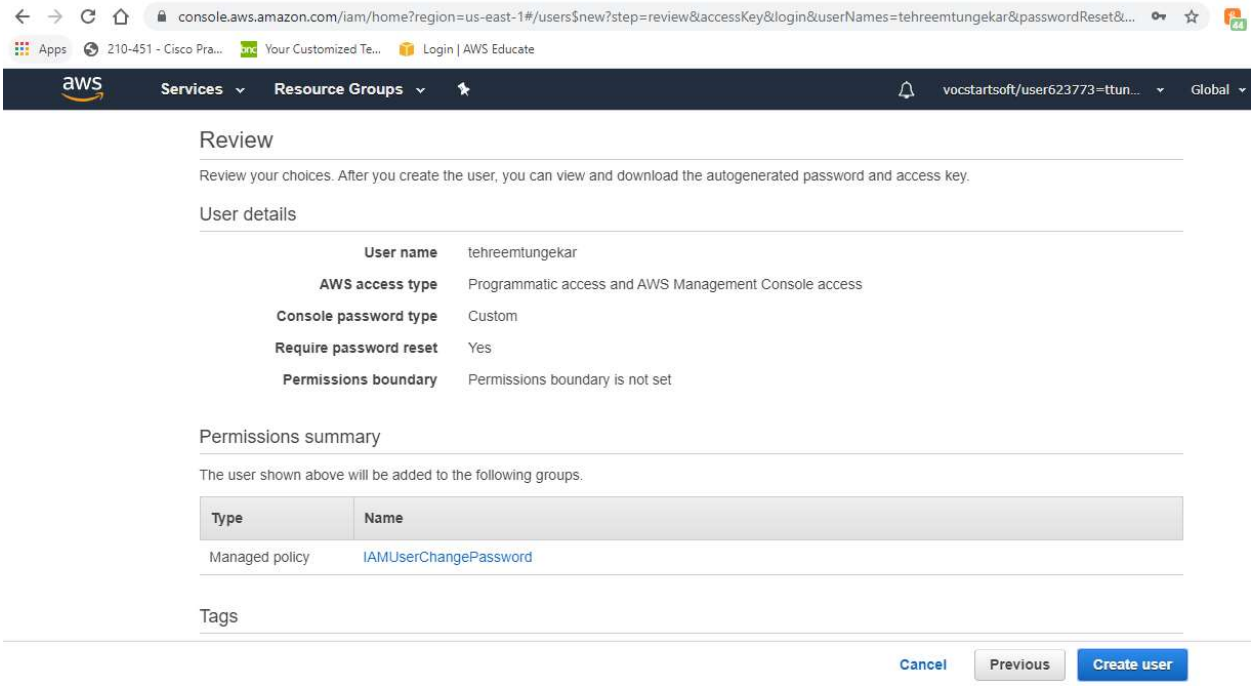


Click on Add User:

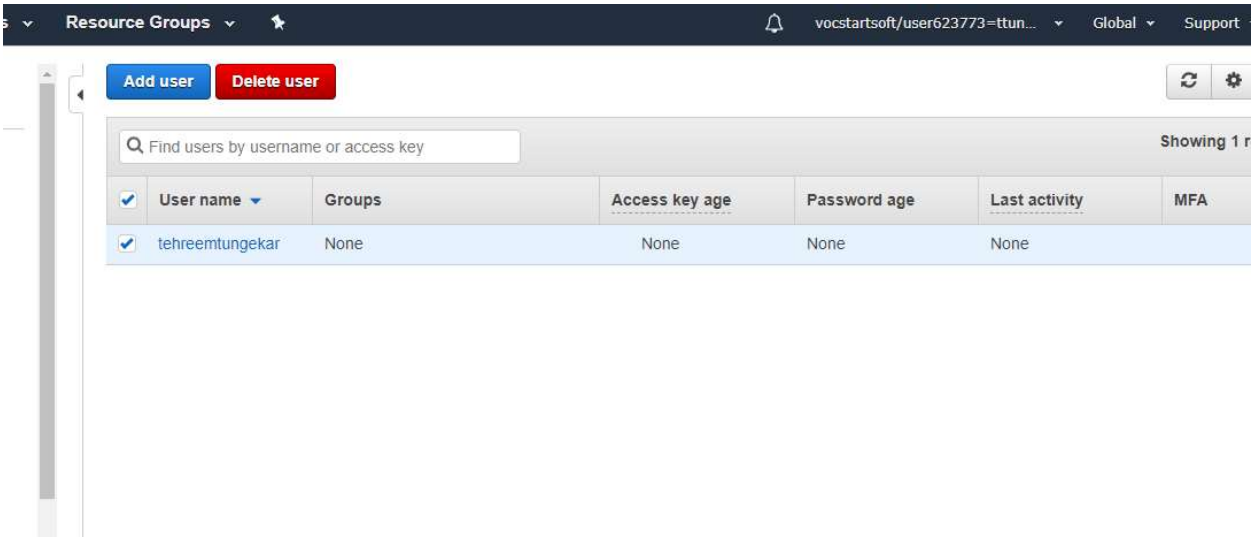


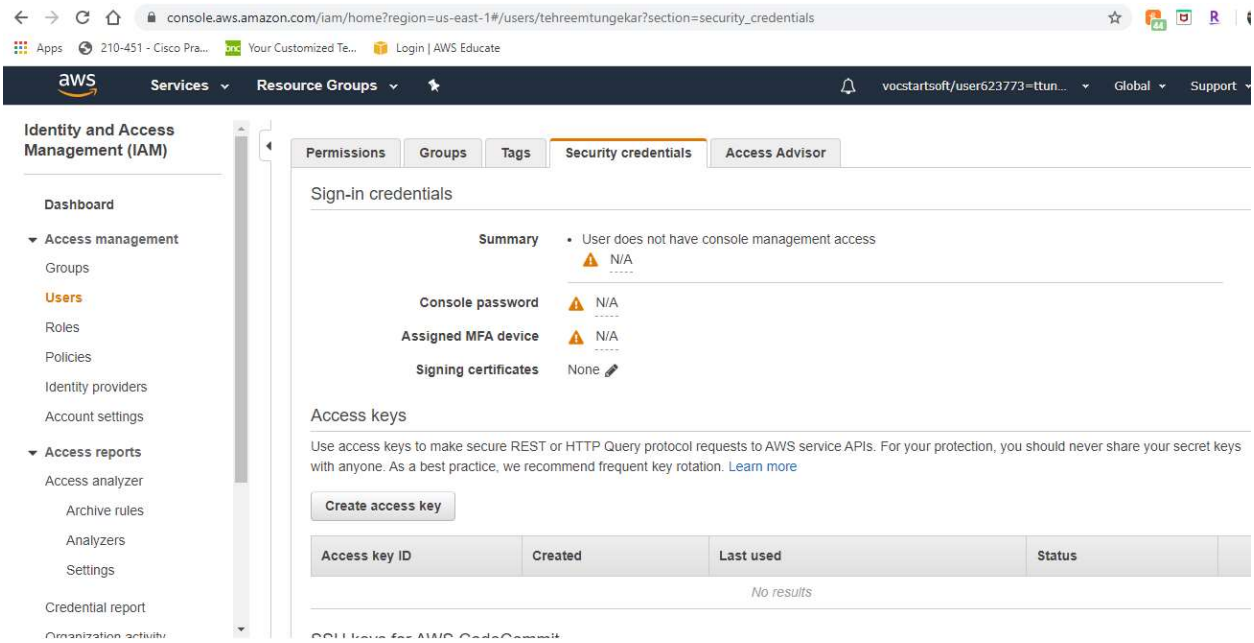
Click on Next after giving a username and creating a password.

Since I don't want to create any group, I clicked Next,also, I did not create any Tags

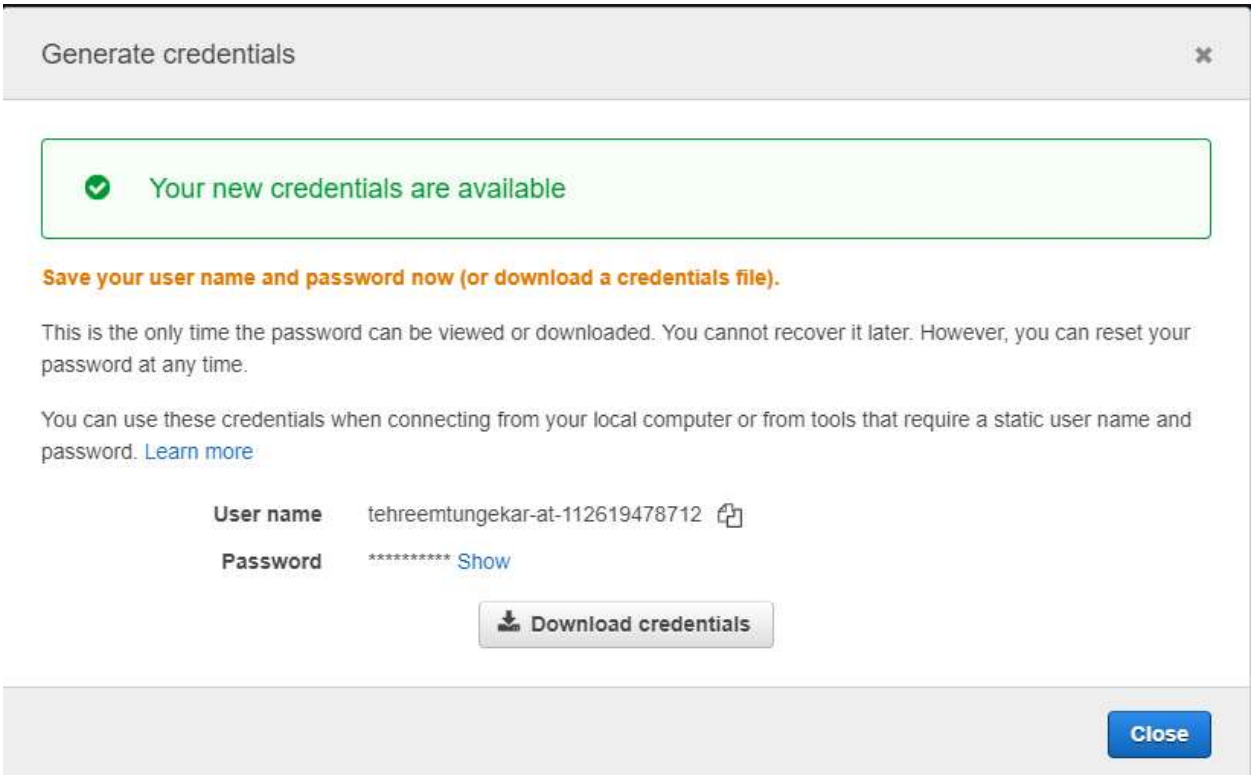


Click on Create uSer



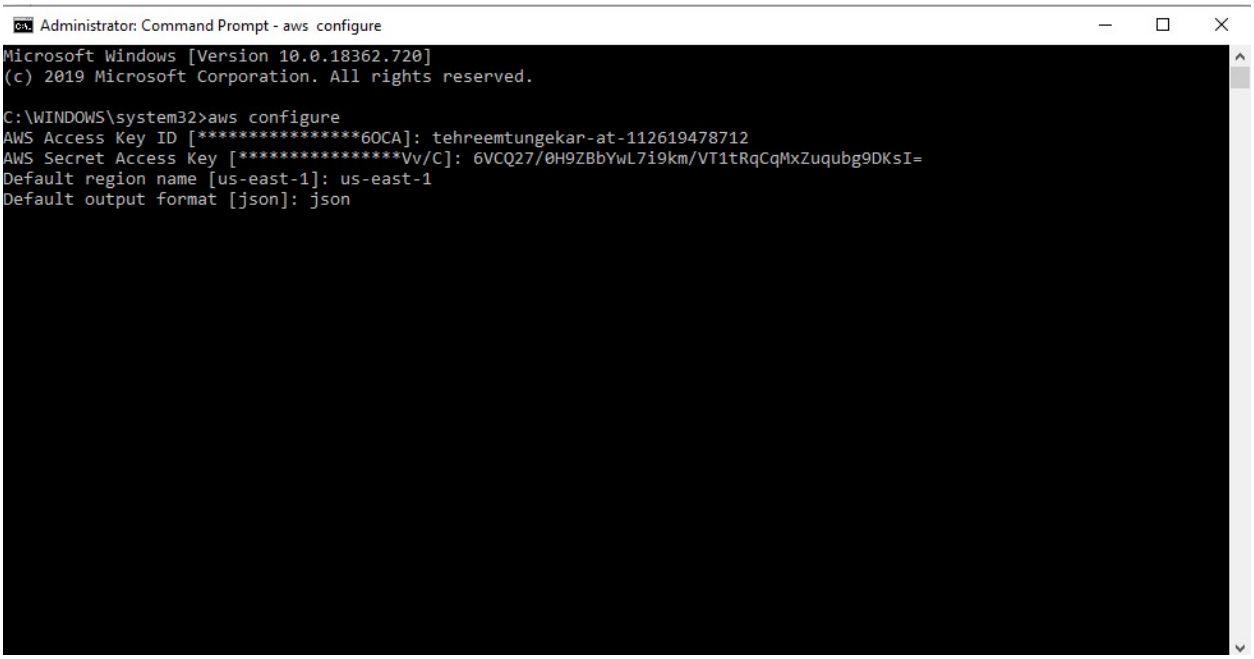


Due to limited access, the warnings are shown. Click on Create Access and Generate Credentials:

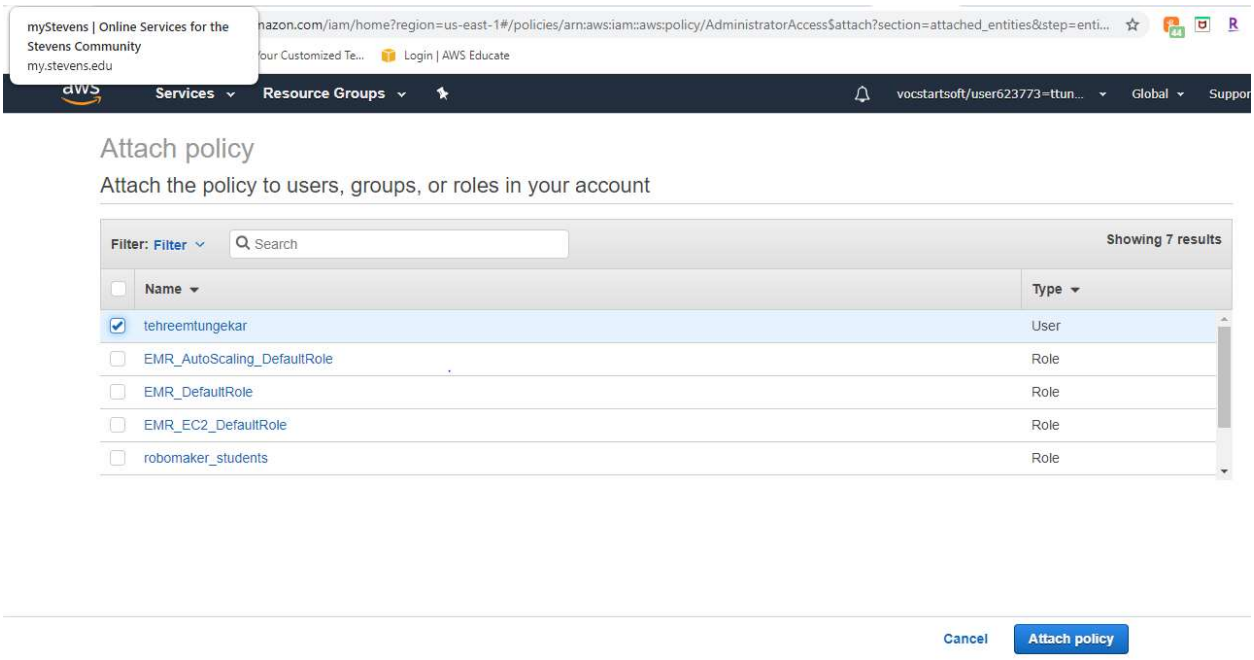


Click on Download Credentials.

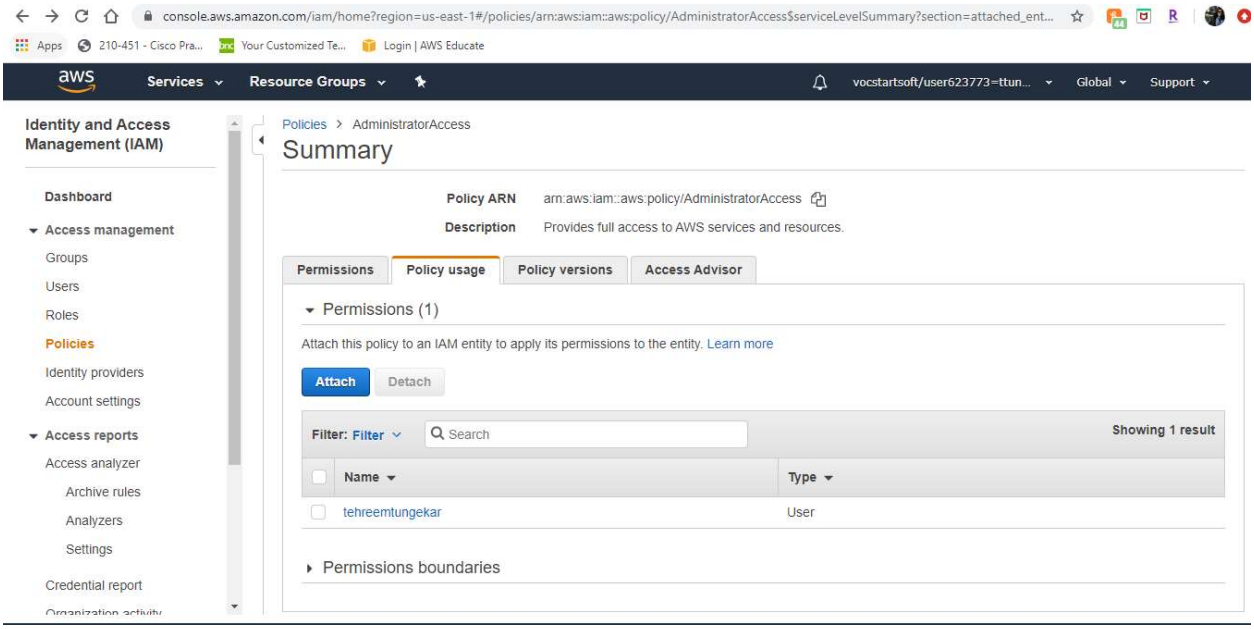
Step 3:Open command prompt and enter command aws configure. Then fill data as per downloaded credential file



Step 4: Creating security group:



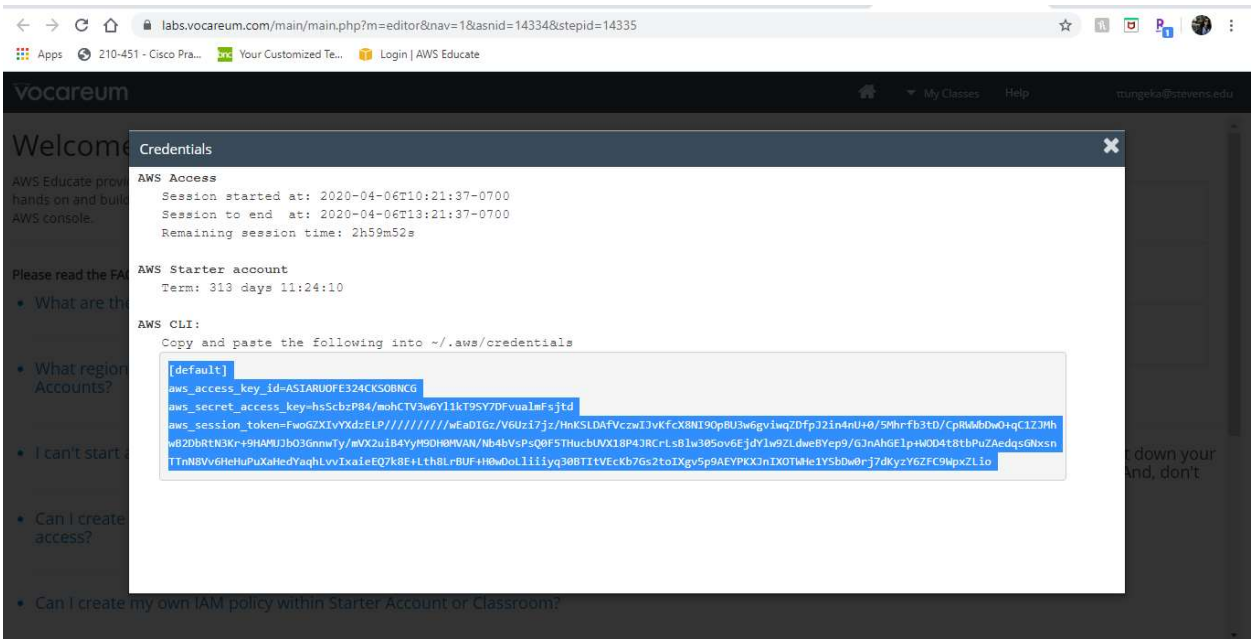
Click on attach policy and check if the policy is attached successfully.



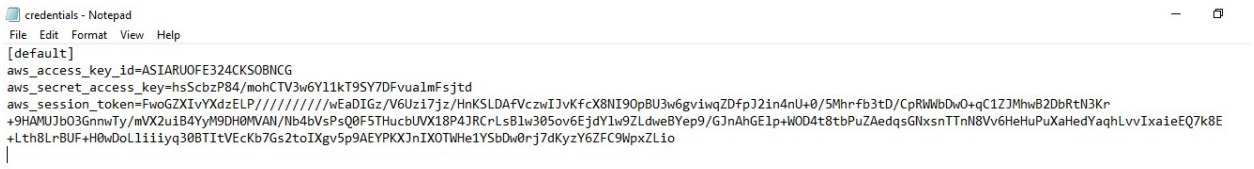
So it seems there were some issues with IAM users, which can be resolved as:

First login to AWS account and click on Account details.

Then click on Account Details and get the details below:



Paste this in C:/Users/Tehreem/.aws/credentials file



Save this file. Now run aws configure through cmd in run as administrator mode:

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>aws configure
AWS Access Key ID [****************BNCG]: ASIARUOFE324CKSOBNCG
AWS Secret Access Key [****************sjtd]: hsScbzP84/mohCTV3w6Yl1kT9SY7DFvualmFsjtd
Default region name [us-east-1]: us-east-1
Default output format [json]: json

C:\WINDOWS\system32>aws ec2 create-security-group --group-name my-sg1 --description "My new sec"
{
    "GroupId": "sg-0b9d6a7d80863478e"
}

C:\WINDOWS\system32>script
'script' is not recognized as an internal or external command,
operable program or batch file.

C:\WINDOWS\system32>
```

After creating the security group, we need to

```
C:\WINDOWS\system32>aws ec2 authorize-security-group-ingress --group-id sg-0b9d6a7d80863478e --protocol tcp --port 80 --cidr 76.116.178.250/32

C:\WINDOWS\system32>aws ec2 authorize-security-group-ingress --group-id sg-0b9d6a7d80863478e --protocol tcp --port 22 --cidr 76.116.178.250/32

C:\WINDOWS\system32>aws ec2 describe-security-groups --group-ids sg-0b9d6a7d80863478e
{
    "SecurityGroups": [
        {
            "IpPermissionsEgress": [
                {
                    "IpProtocol": "-1",
                    "PrefixListIds": [],
                    "IpRanges": [
                        {
                            "CidrIp": "0.0.0.0/0"
                        }
                    ],
                    "UserIdGroupPairs": [],
                    "Ipv6Ranges": []
                }
            ],
            "Description": "My new sec",
            "IpPermissions": [
                {
                    "PrefixListIds": [],
                    "FromPort": 80,
                    "IpRanges": [
                        {
                            "CidrIp": "76.116.178.250/32"
                        }
                    ],
                    "ToPort": 80,
                    "IpProtocol": "tcp",
                    "UserIdGroupPairs": [],
                    "Ipv6Ranges": []
                },
                {
                    "PrefixListIds": [],
                    "FromPort": 22,
```

The aws ec2 describe-security-groups command displays the ports just added.

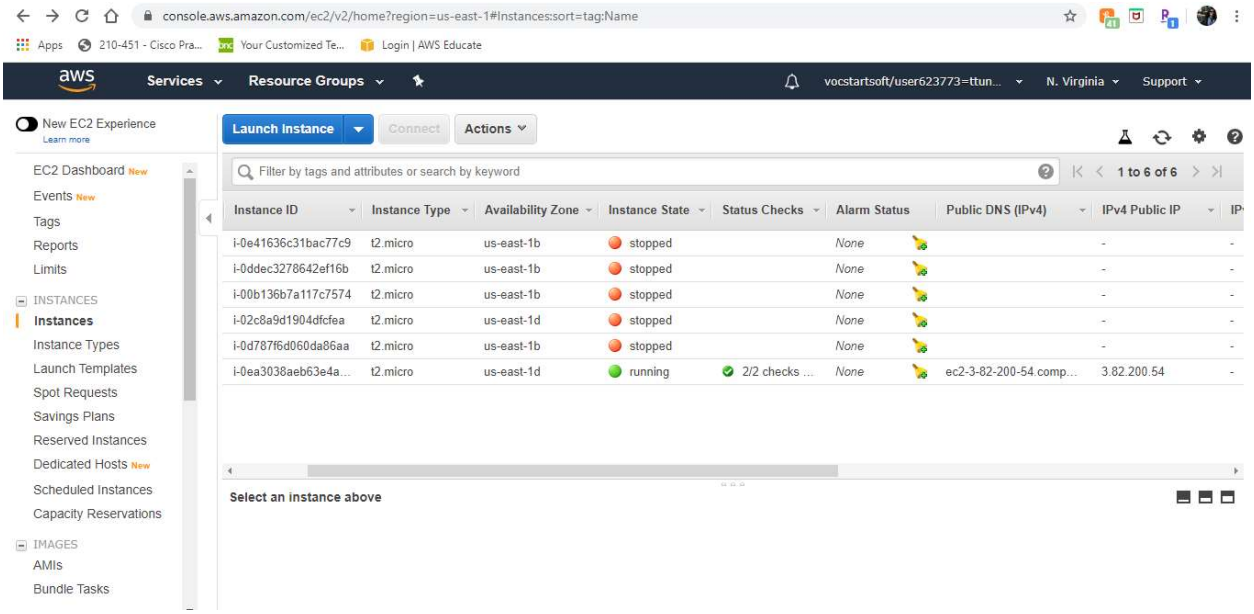
I have added port 3389 just for testing it.

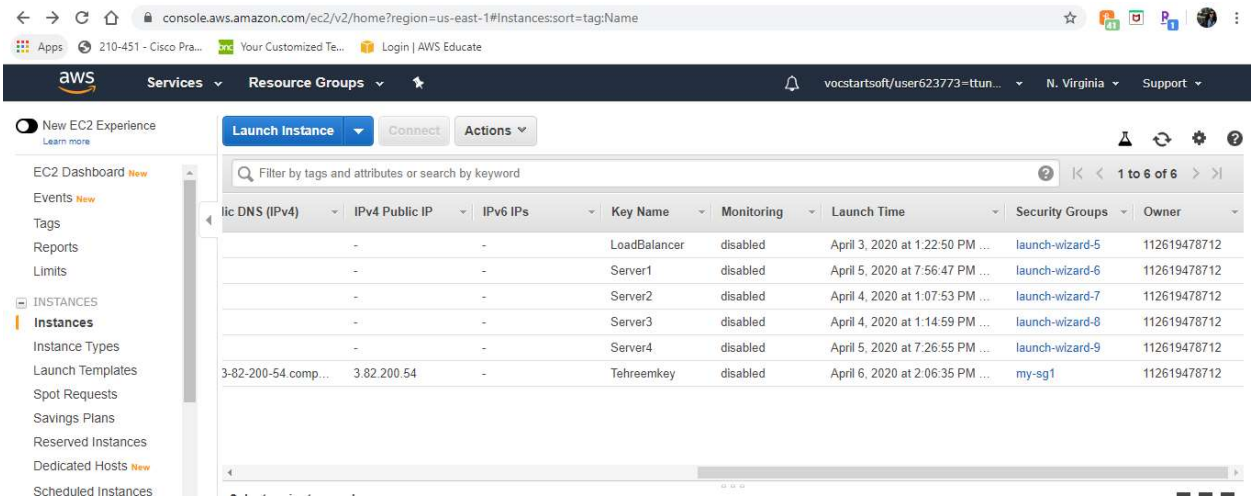
Now we need to create a key-pair for this instance.

Take ami id from the EC2 console: ami-0fc61db8544a617ed

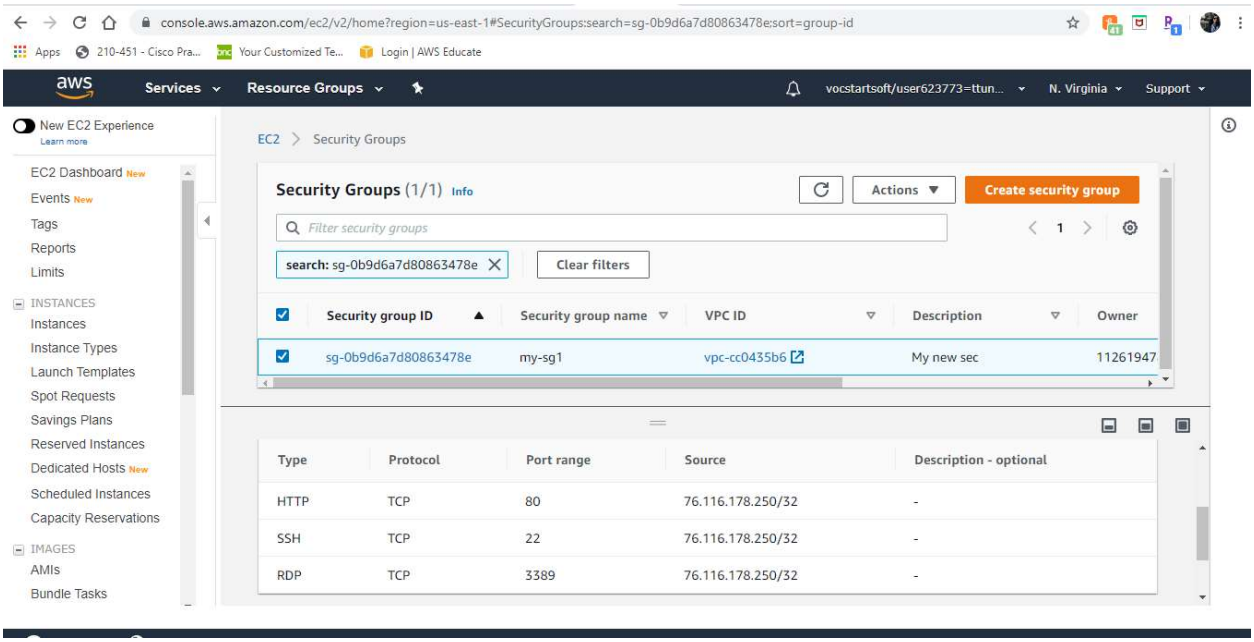
```
C:\WINDOWS\system32>aws ec2 run-instances --image-id ami-0fc61db8544a617ed --count 1 --instance-type t2.micro --key-name Tehreemkey --security-group-ids sg-0b9d6a7d8086
3478e
{
    "Instances": [
        {
            "Monitoring": {
                "State": "disabled"
            },
            "PublicDnsName": "",
            "StateReason": {
                "Message": "pending",
                "Code": "pending"
            },
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "EbsOptimized": false,
            "LaunchTime": "2020-04-06T18:06:35.000Z",
            "PrivateIpAddress": "172.31.80.162",
            "ProductCodes": [],
            "VpcId": "vpc-cc0435b6",
            "StateTransitionReason": "",
            "InstanceId": "i-0ea3038aeb63e4a6b",
            "ImageId": "ami-0fc61db8544a617ed",
            "PrivateDnsName": "ip-172-31-80-162.ec2.internal",
            "KeyName": "Tehreemkey",
            "SecurityGroups": [
                {
                    "GroupName": "my-sg1",
                    "GroupId": "sg-0b9d6a7d80863478e"
                }
            ],
            "ClientToken": "",
            "SubnetId": "subnet-ddb457fc",
            "InstanceType": "t2.micro",
            "NetworkInterfaces": [
                {
                    "Status": "in-use",
                    "MacAddress": "12:ed:42:83:a6:dd",
```

Instance has been created in AWS EC2 dashboard:





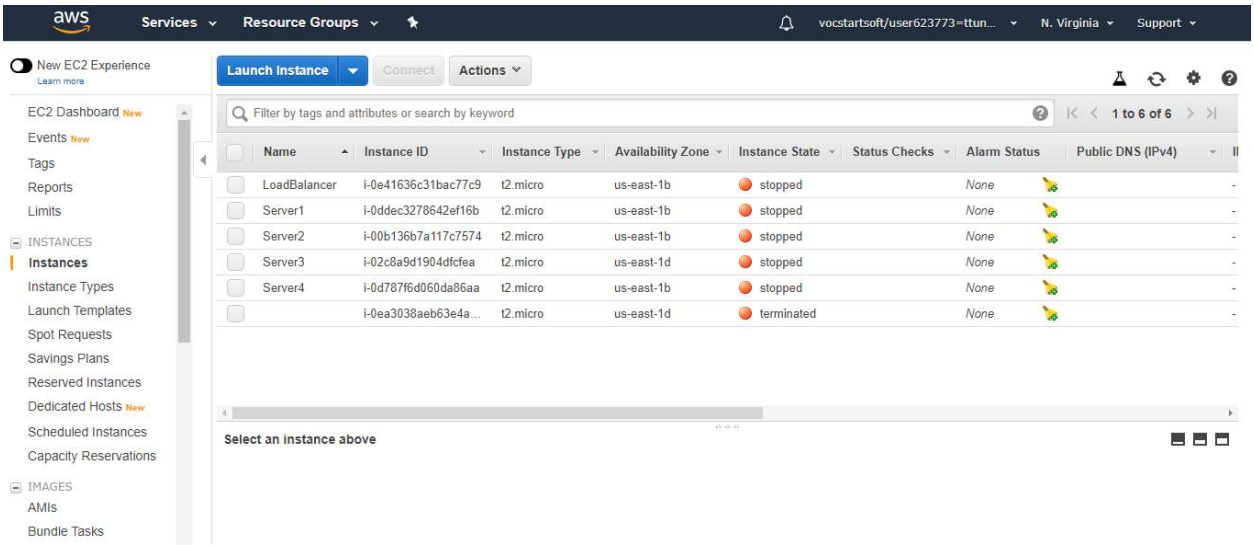
Tehreemkey is the key, as specified in Cmd and my sg1 security group has been assigned successfully.



The three security parameters specified have also been added. Port 3389 is just for testing purpose.

Shutting down instances:

```
C:\WINDOWS\system32>aws ec2 terminate-instances --instance-ids i-0ea3038aeb63e4a6b
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-0ea3038aeb63e4a6b",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}

C:\WINDOWS\system32>
```



Successfully terminated!

To record all the commands that were executed:

```
C:\WINDOWS\system32>doskey/history>history.txt

C:\WINDOWS\system32>
```

Now go to C:\WINDOWS\system32. There should be a file named history:

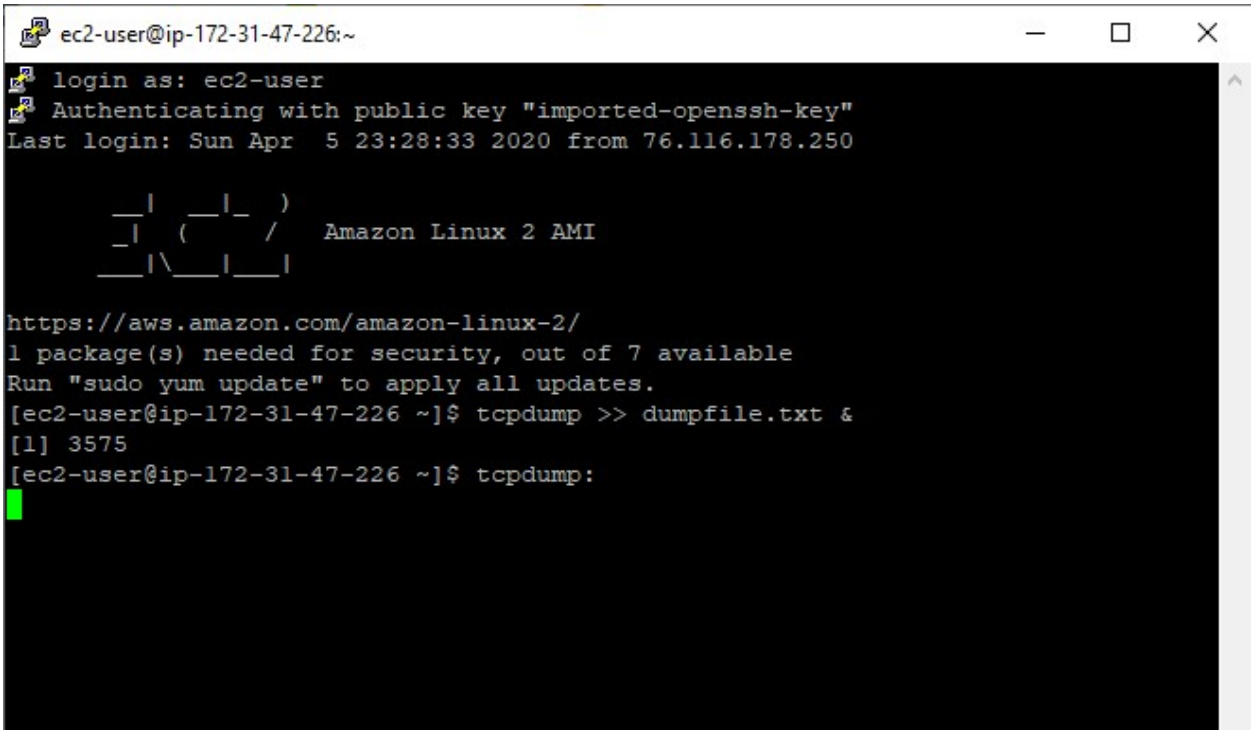
history - Notepad

File Edit Format View Help

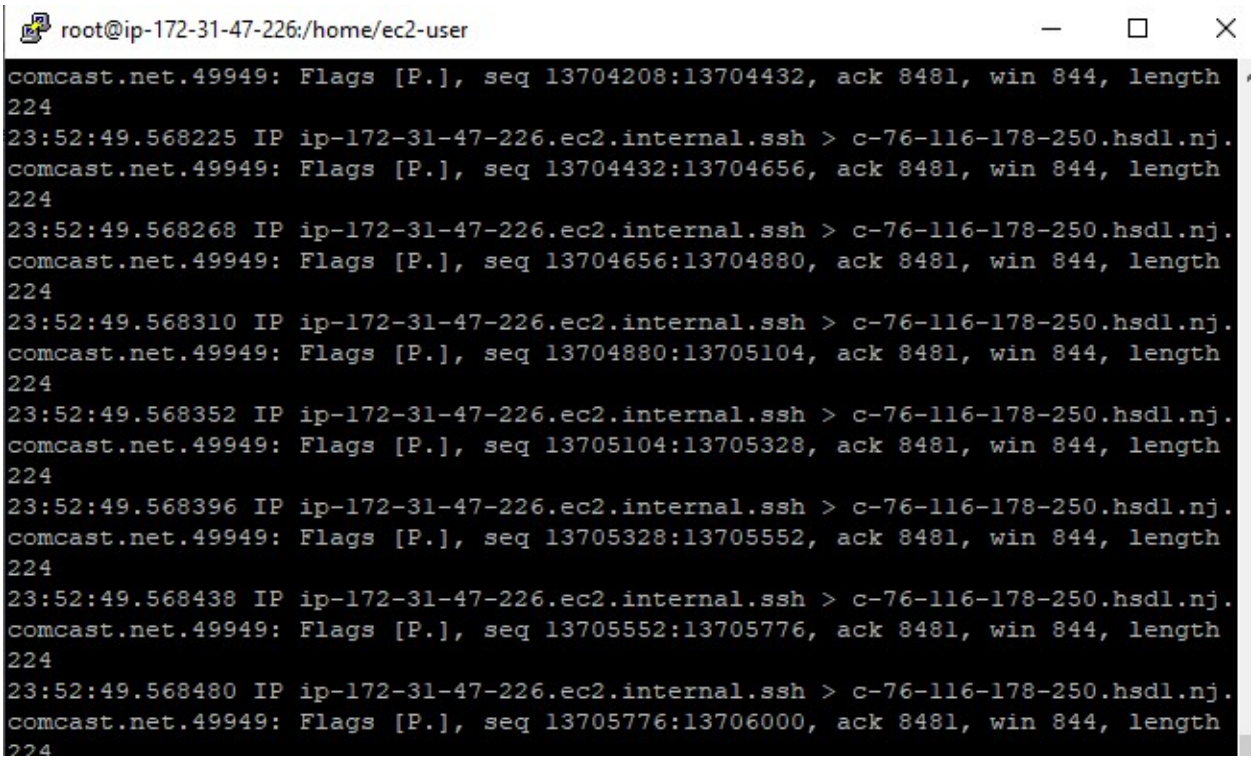
```
aws configure
aws ec2 create-security-group --group-name my-sg1 --description "My new sec"
script
start-transcript
Start-Transcript
     [[-Path] <String>]
     [-Append]
     [-Force]
     [-NoClobber]
     [-IncludeInvocationHeader]
     [-WhatIf]
     [-Confirm]
Start-Transcript -Path "C:\transcripts\transcript0.txt" -NoClobber
doskey/history>history.txt
aws ec2 authorize-security-group-ingress --group-id sg-0b9d6a7d80863478e --protocol tcp --port 3389 --cidr 76.116.178.250/32
aws ec2 authorize-security-group-ingress --group-id sg-0b9d6a7d80863478e --protocol tcp --port 80 --cidr 76.116.178.250/32
aws ec2 authorize-security-group-ingress --group-id sg-0b9d6a7d80863478e --protocol tcp --port 22 --cidr 76.116.178.250/32
aws ec2 describe-security-groups --group-ids sg-0b9d6a7d80863478e
aws ec2 create-key-pair --key-name Tehreemkey --query 'KeyMaterial' --output text > Tehreemkey.pem
aws ec2 describe-key-pairs --key-name Tehreemkey
aws ec2 run-instances --image-id ami-0fc61db8544a617ed --count 1 --instance-type t2.micro --key-name Tehreemkey --security-group-ids sg-0b9d6a7d80863478e
aws ec2 create-tags --resources i-0ea3038aeb63e4a6b --tags Key=Created through CLI,Value=MyInstance
aws ec2 create-tags --resources i-0ea3038aeb63e4a6b --tags Key=CreatedthroughCLI,Value=MyInstance
aws ec2 terminate-instances --instance-ids i-0ea3038aeb63e4a6b
doskey/history>history.txt
```

So all commands were recorded successfully!

Step 17: Running tcpdump command:



If this command is run on the server, it captures all the packets endlessly and the server has to be stopped forcefully.



So I made the packet capture go to a dumpfile named dumpfile.txt.