Deploying a Highly Available Web Application and Bastion Host in AWS

Introduction

Bastion Host

- **A bastion host is a system** that is exposed to the internet.
- In terms of security, Bastion is the only server that is exposed to the internet and should be highly protective to malicious attacks.
- **A Bastion host** is also **known as a Jump Box**. It is a computer that acts like a proxy server and that allows the client machine to connect to the remote server.
- It usually resides outside the firewall.
- The Bastion server filters the incoming traffic and prevents unwanted connections entering the network thus acting as a gateway to maintain the security of bastion hosts, all unnecessary software, and daemons.

High Availability

- Consider your application is running on a single EC2 instance. If the traffic to your application increases and you need further resources, we can launch multiple EC2 instances from an already running server and then **use Elastic Load Balancing** to distribute the traffic to your application among the newly-created servers.
- We can also **eliminate the Fault tolerance** in your application by placing the servers (EC2 instances) across different availability zones.
- In the event of **Failure of one Availability zone**, your application will serve or handle the **traffic from another availability zone.**
- High **Availability and fault tolerance** can be **achieved using Elastic Load balancers.**

Task Details

1. Launch a **Bastion Host instance** along with **two web application instances, two web application instances should be launched in the private subnet.**
2. Set up a **Load Balancer** and associate the two web instances to the Load Balancer.
3. **SSH** into the web servers **via** the **Bastion server**.
4. Publish a test **index.html** on both of the web servers.
5. Access the webpage using the load balancer's DNS endpoint.
6. Check the responses to see the **Load Distribution** between the 2 servers.
7. Stop or terminate one of the web servers.
8. Check the responses to see how to Load Distribution changes

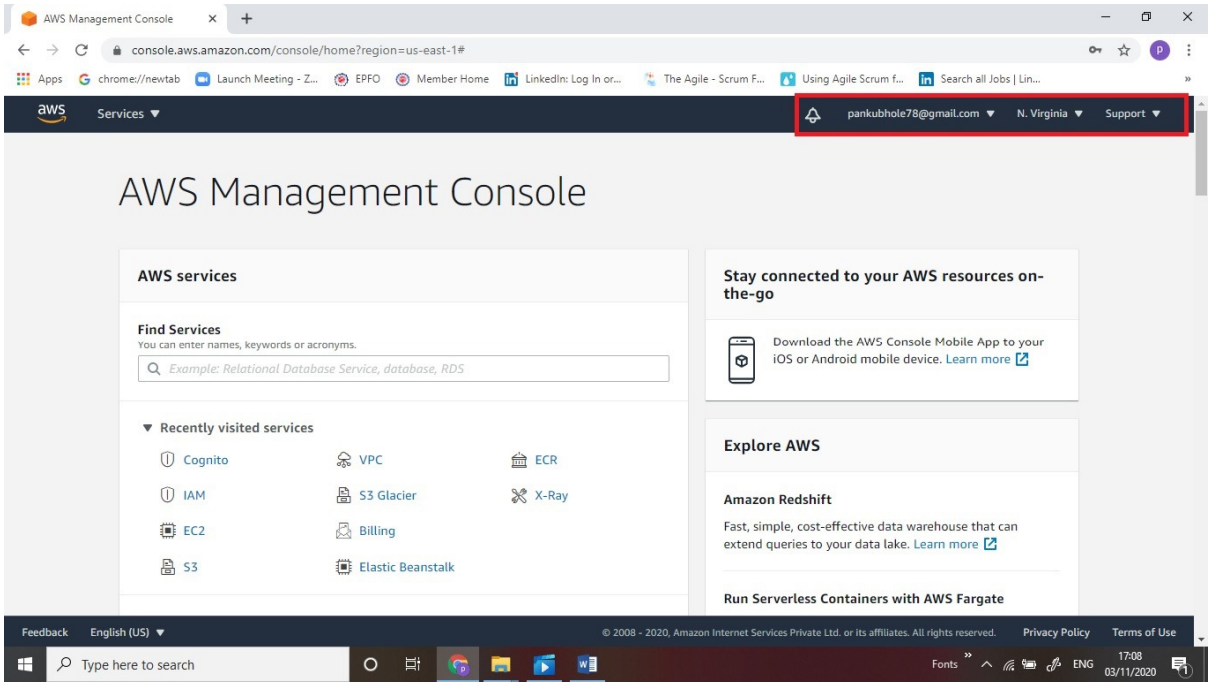
Following Steps following making in this projects

1)vpc
2)public and private subnet
3)nat gateway deployment
4)Bastion host deployment
5)launching web servers securely in private subnet
6)ssh into remote servers in the private subnet
7)Creating customized security groups
8)creating and accessing a load balancer
9)RSA private key login procedure
10)changing file permissions in linux

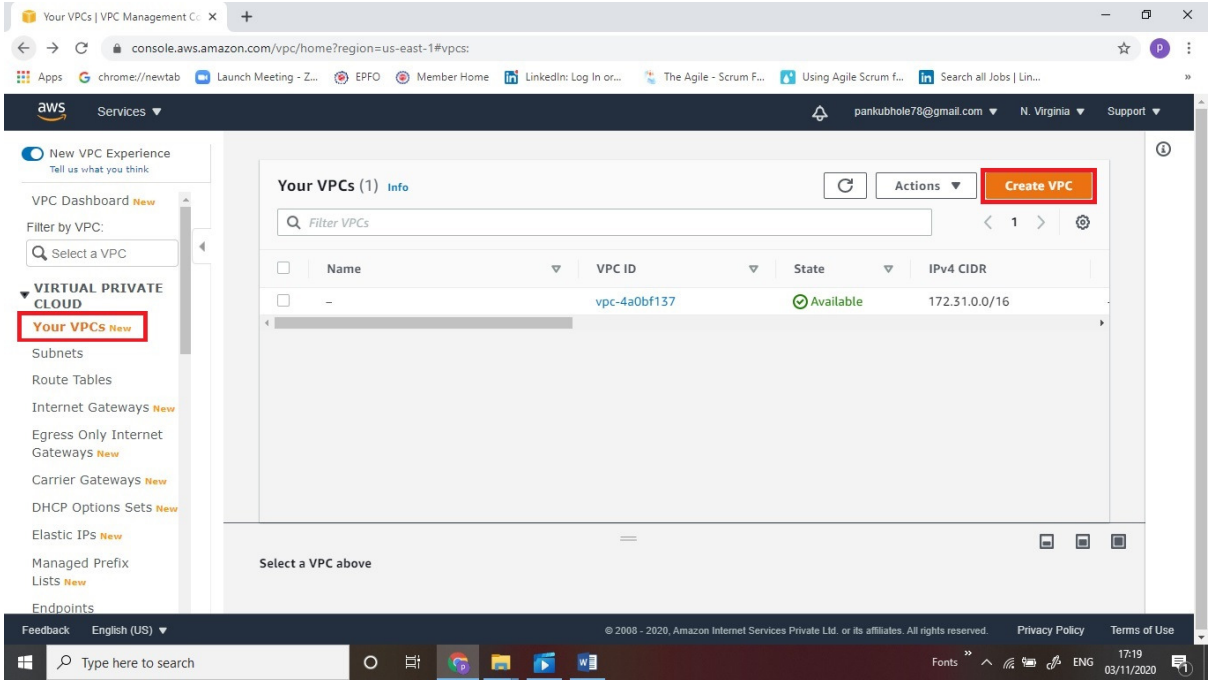
webservers-private subnet
bastion server-public subnet

Create VPC:

1. Log into AWS Management Console.
2. Make sure you are in the **US East (N. Virginia) us-east-1** region.



3. Navigate to **VPC** under the services menu. Click on **Your VPCs**.



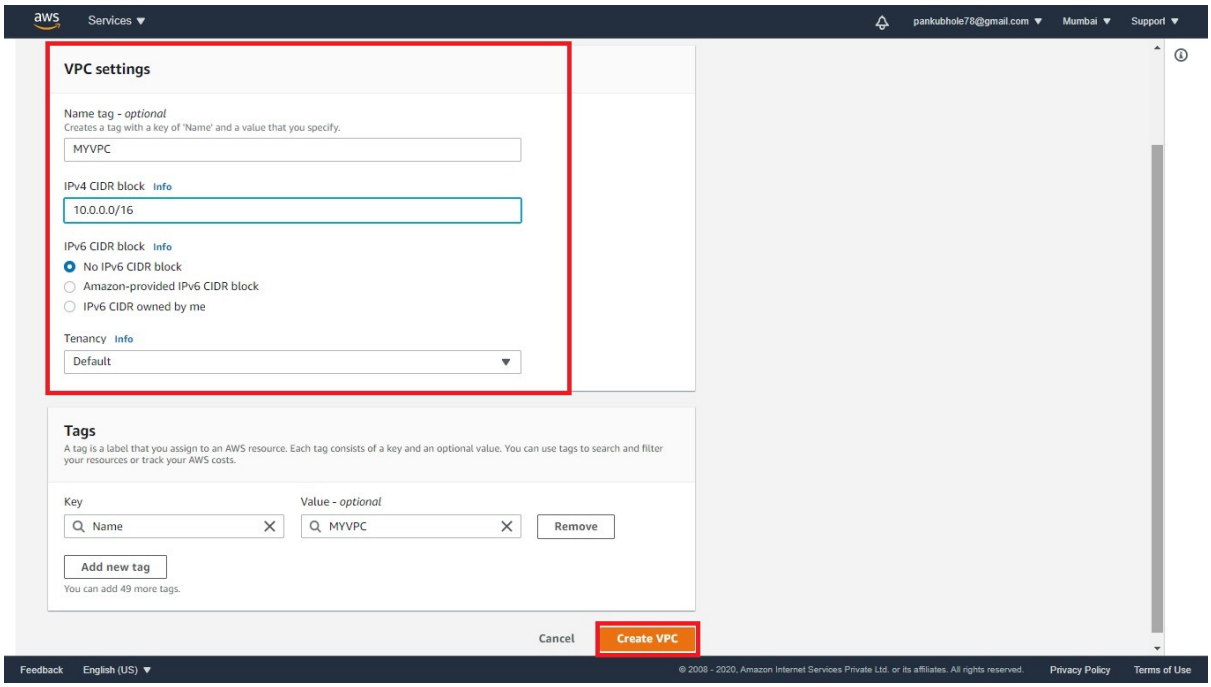
Click on Create VPC Button and add on following details,

Name Tag : ***MyVPC***

IPv4 CIDR block : **10.0.0.0/16** (You can also put any other CIDR range)

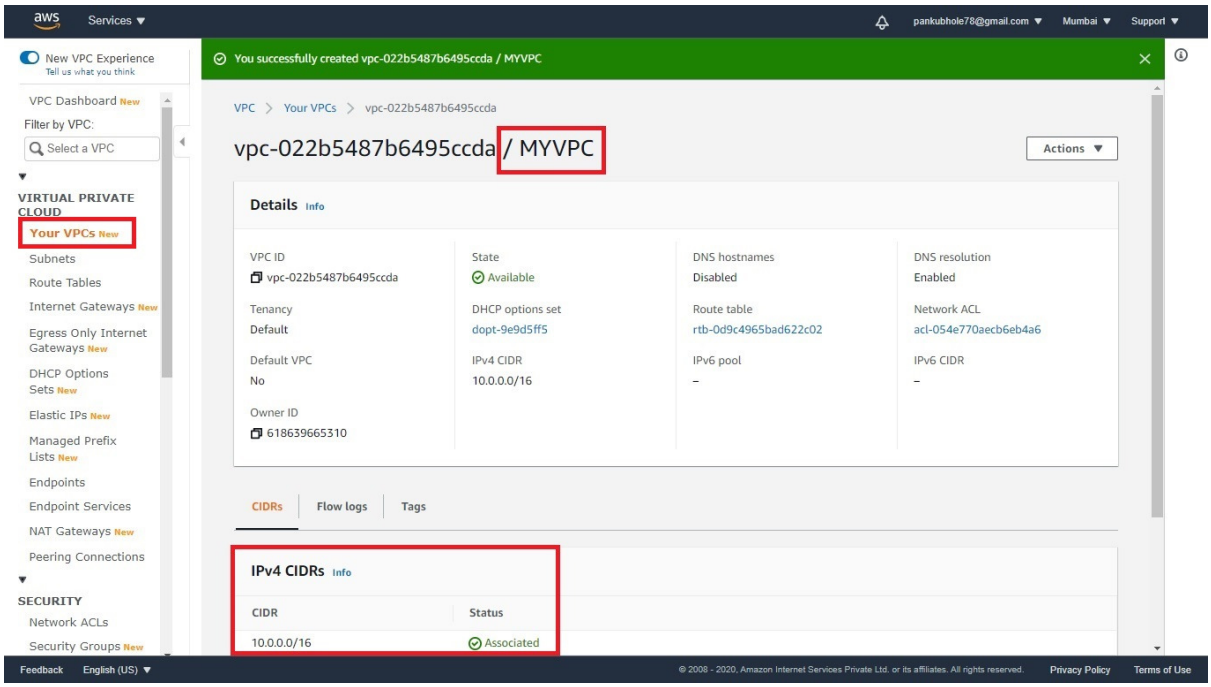
IPv6 CIDR block : Select **No IPv6 CIDR Block**

Tenancy : **Default**



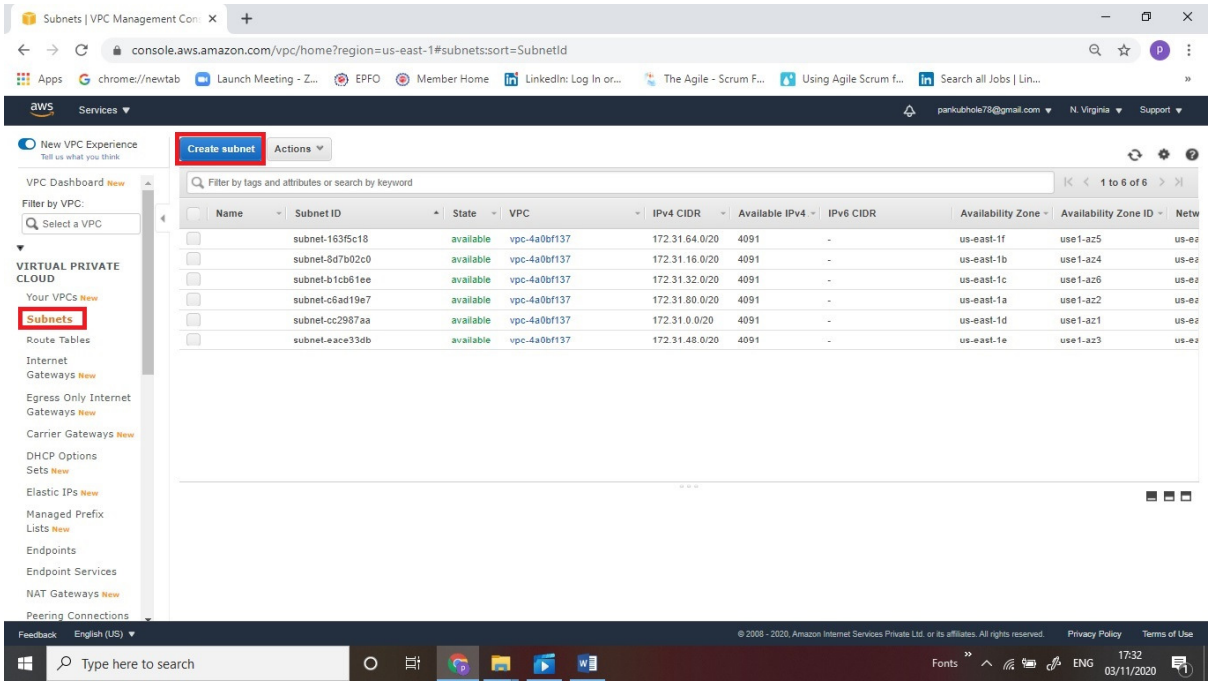
Click on in above screen selected “Create VPC “Button.

4. The VPC is now created and showing VPC details in following screen.



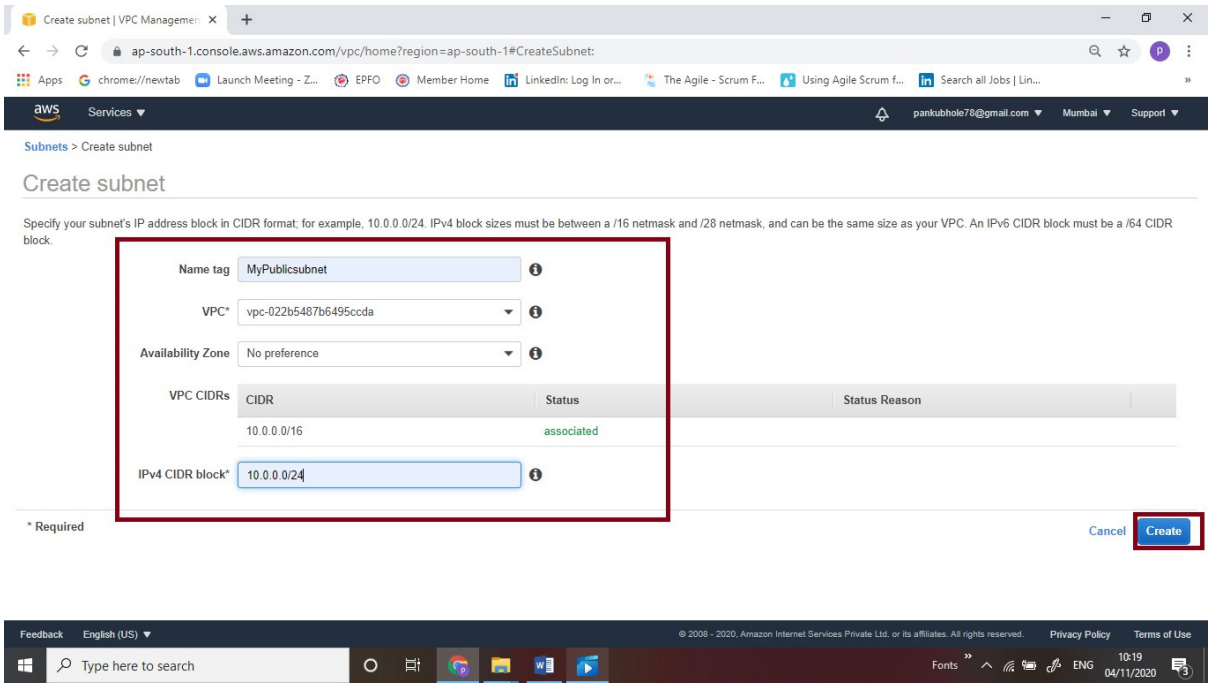
Create Public and Private Subnets:

1. Navigate to **Subnets** in the left panel of the VPC page.



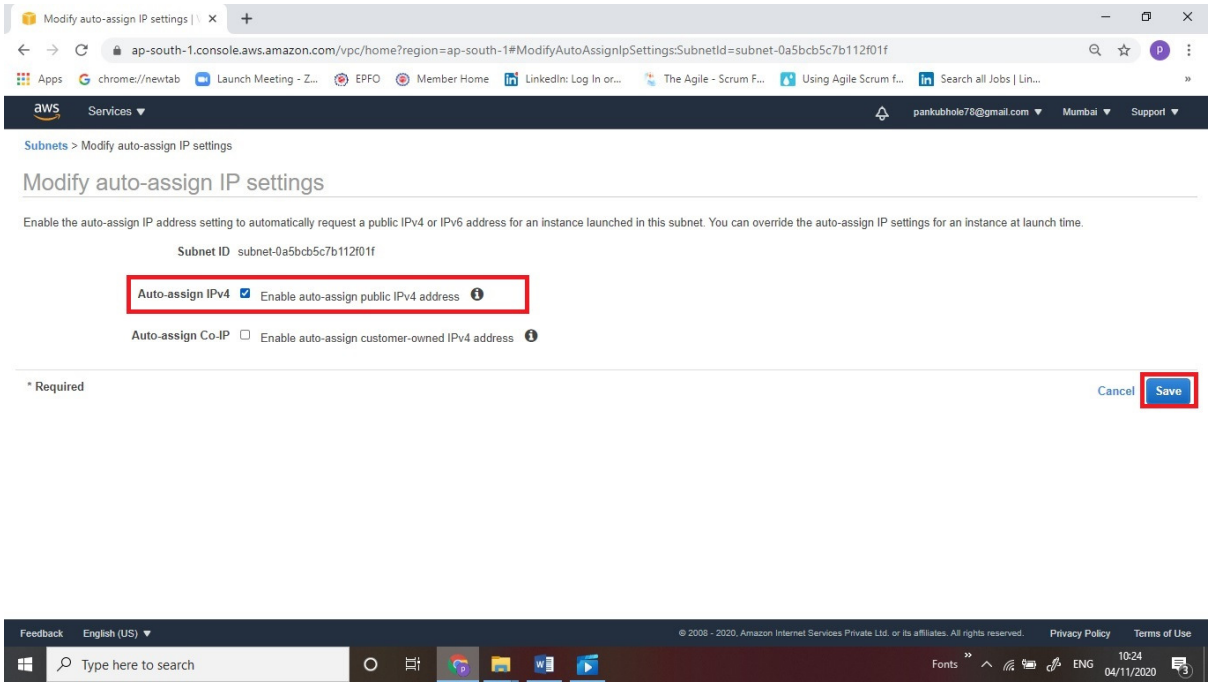
1. **Let's create a Public subnet. Click on Create subnet button**.
    - Name tag : ***MyPublicSubnet***
    - VPC : ***MyVPC***
    - Availability Zone : **No Preference**

- IPv4 CIDR block : **10.0.0.0/24**



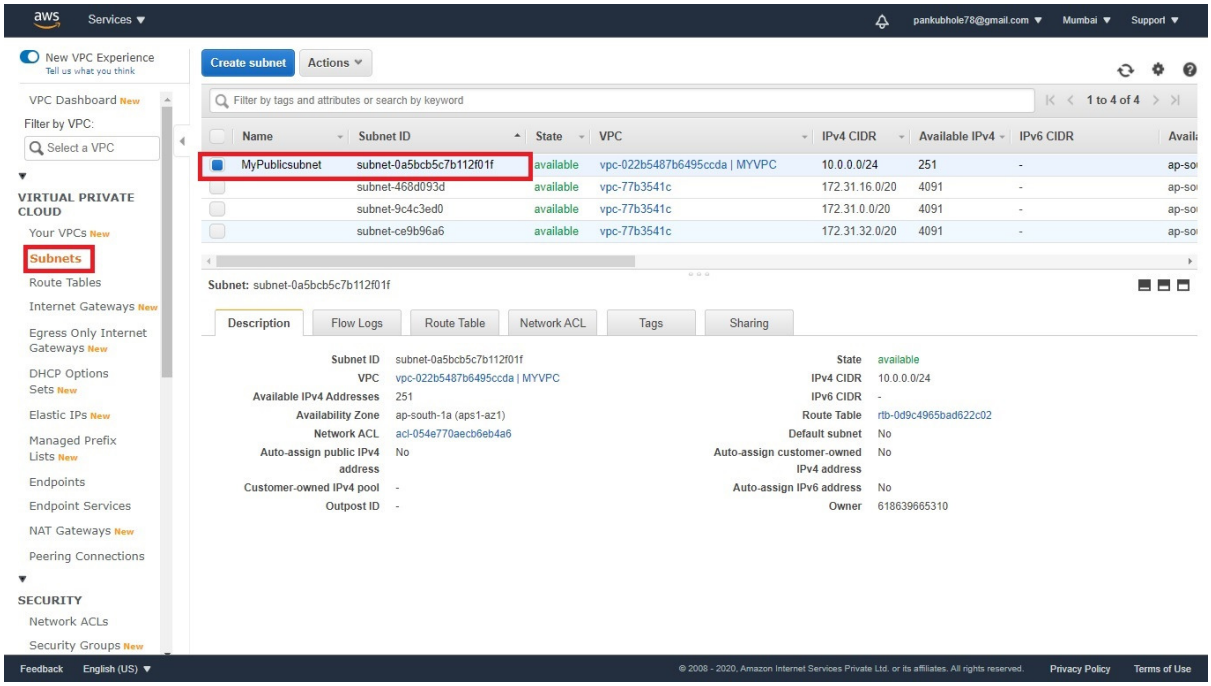
- Make sure add on above details click on Create button on above screen.

2. Let's enable Auto Assign public IP to Instances created within this subnet,
   - Select *MyPublicSubnet* ,
   - Click on **Modify auto-assign IP settings**
   - Auto-assign IPv4 : **Check**

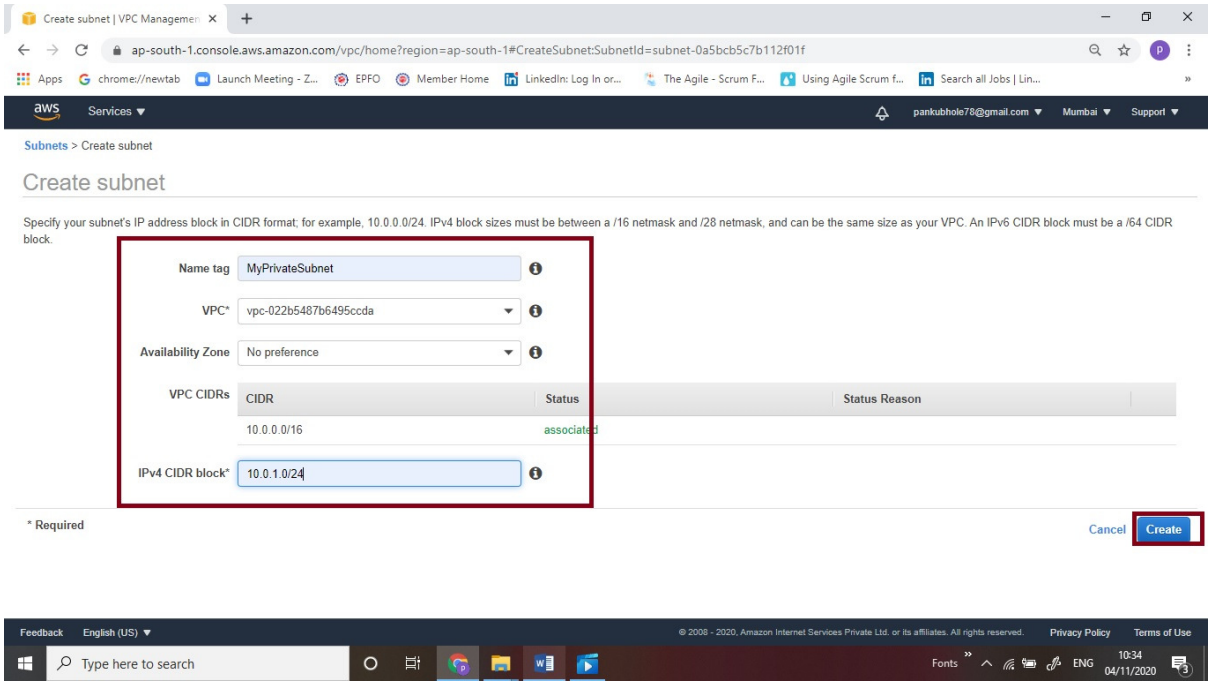


Click on save button and review below screen **Public Subnet details**

0



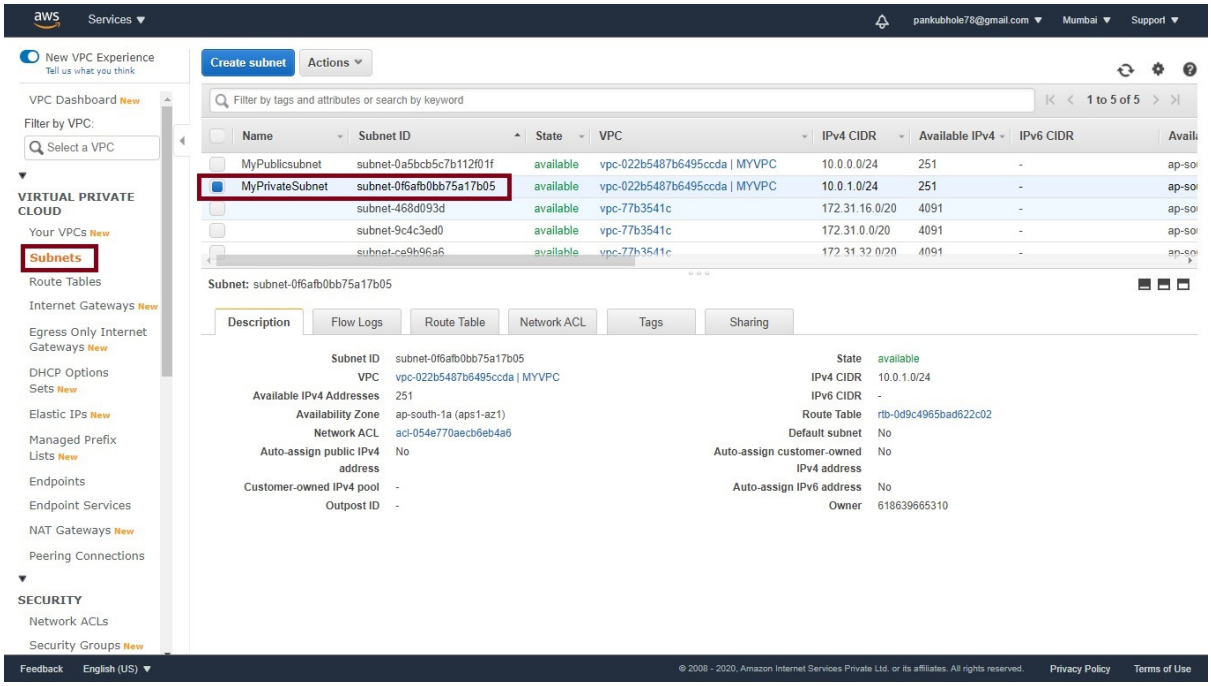
3. Let's create a private subnet. Click on Create subnet button



Added in above screen Private Subnet details

- Name tag : **MyPrivateSubnet**
- VPC : ***MyVPC***
- Availability Zone : **No Preference**
- IPv4 CIDR block : **10.0.1.0/24**

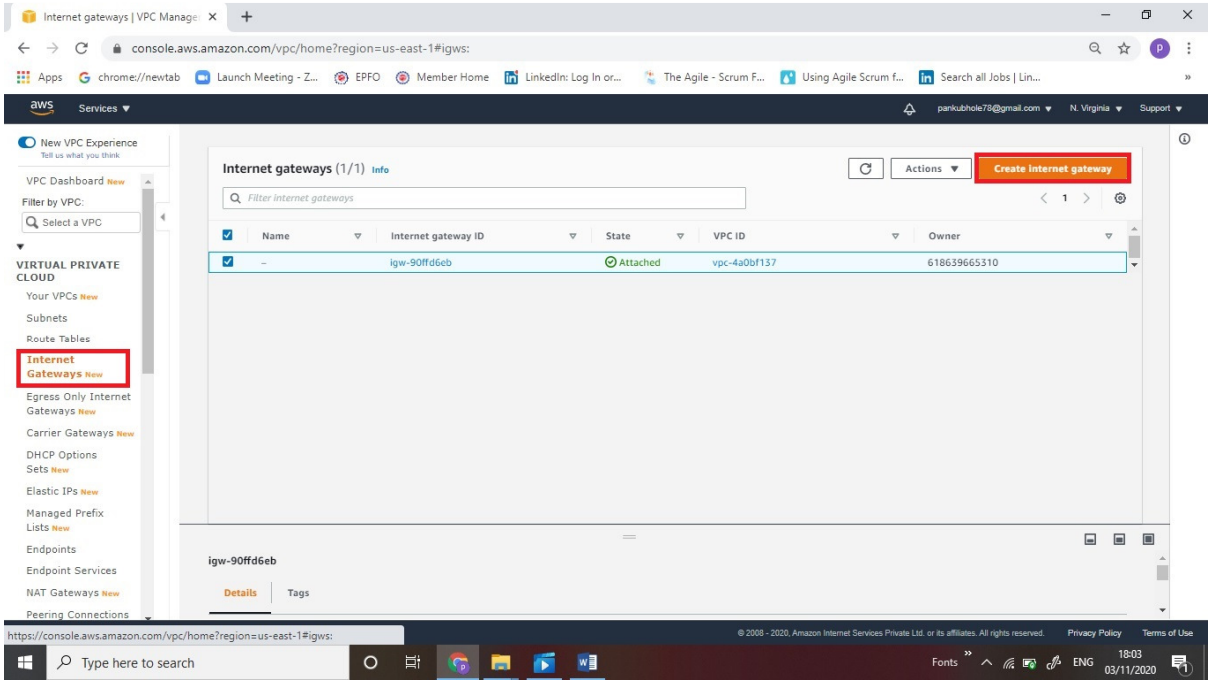
Click on save button and review below screen **Private Subnet details**



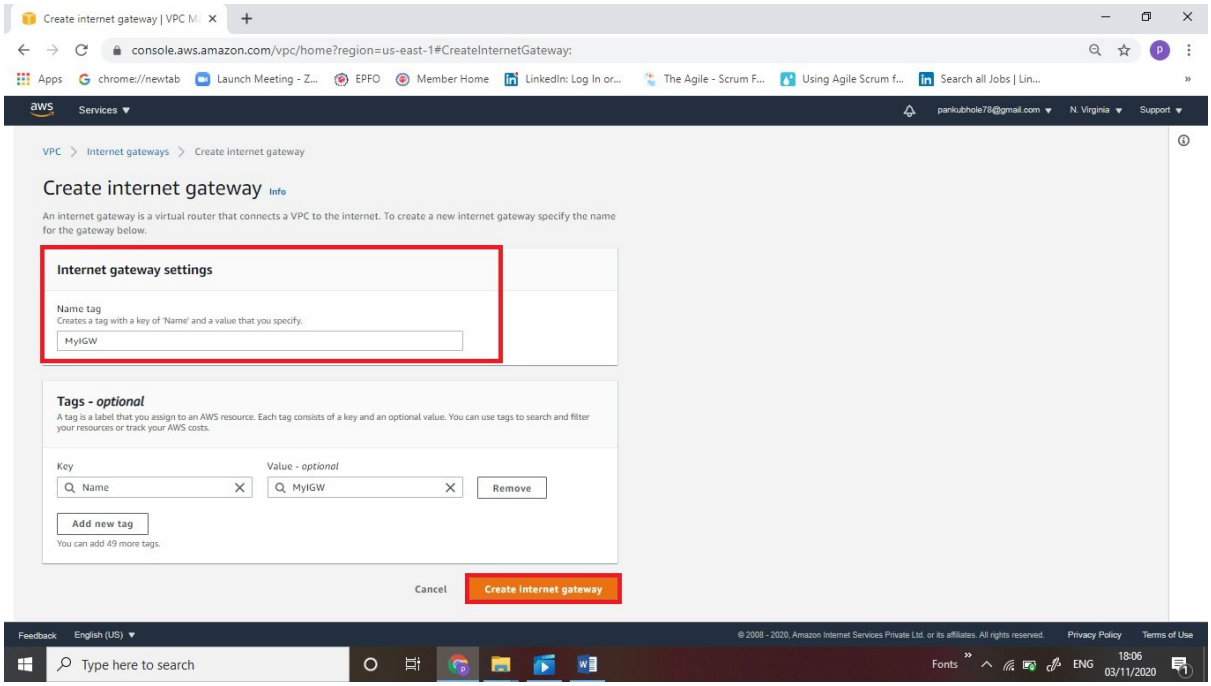
4. Now, two subnets are created

Create Internet Gateway:

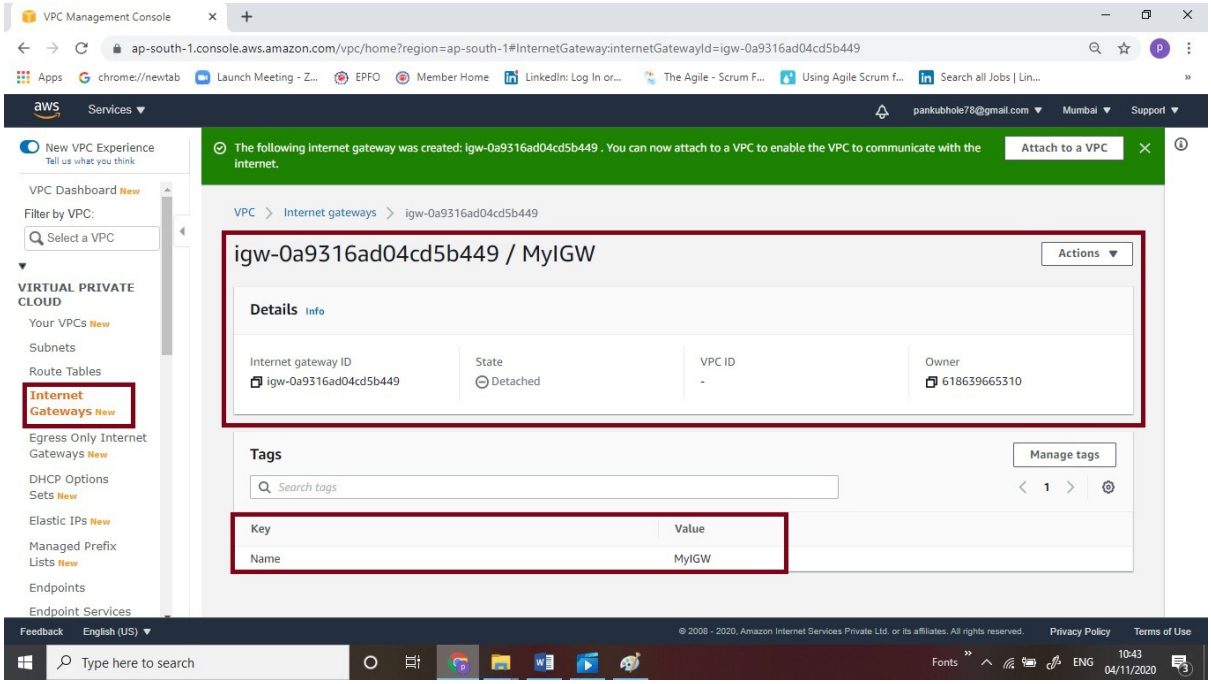
1. Navigate **Internet Gateways** in the left panel of the VPC page.



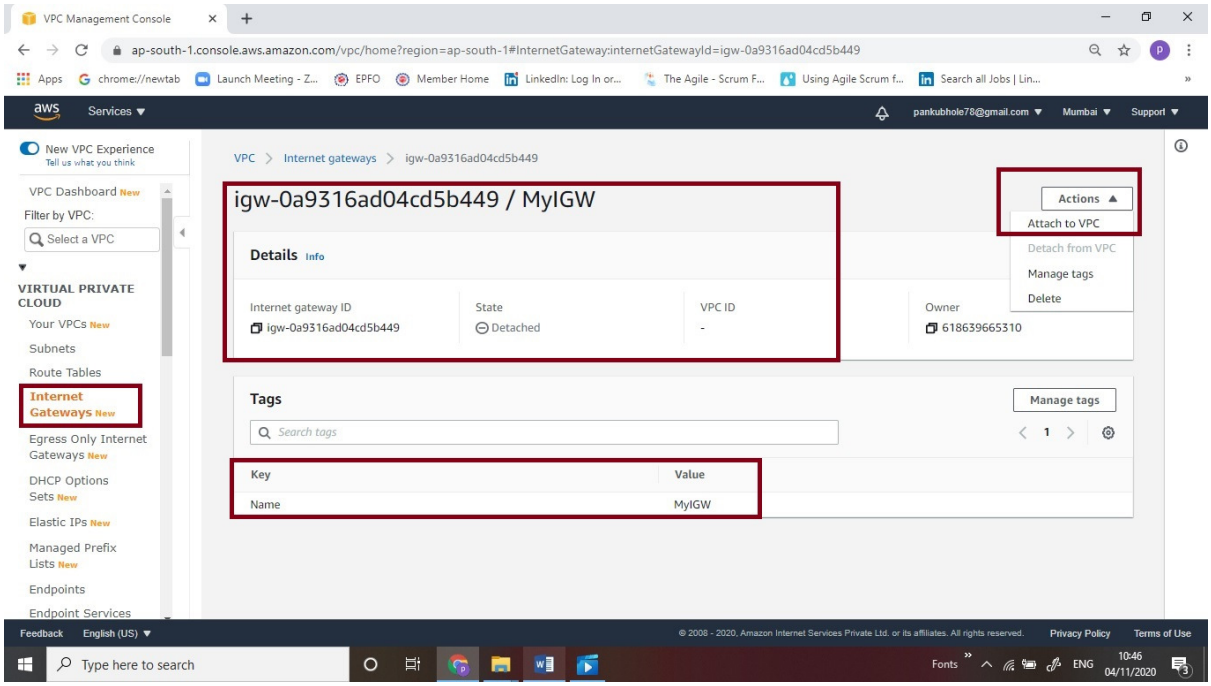
2. Click on Create Internet Gateway button and add following gateway details.
    - Name tag : ***MyIGW***
    - ***Click on Create internet gateway button.***



3. An Internet Gateway is now created.

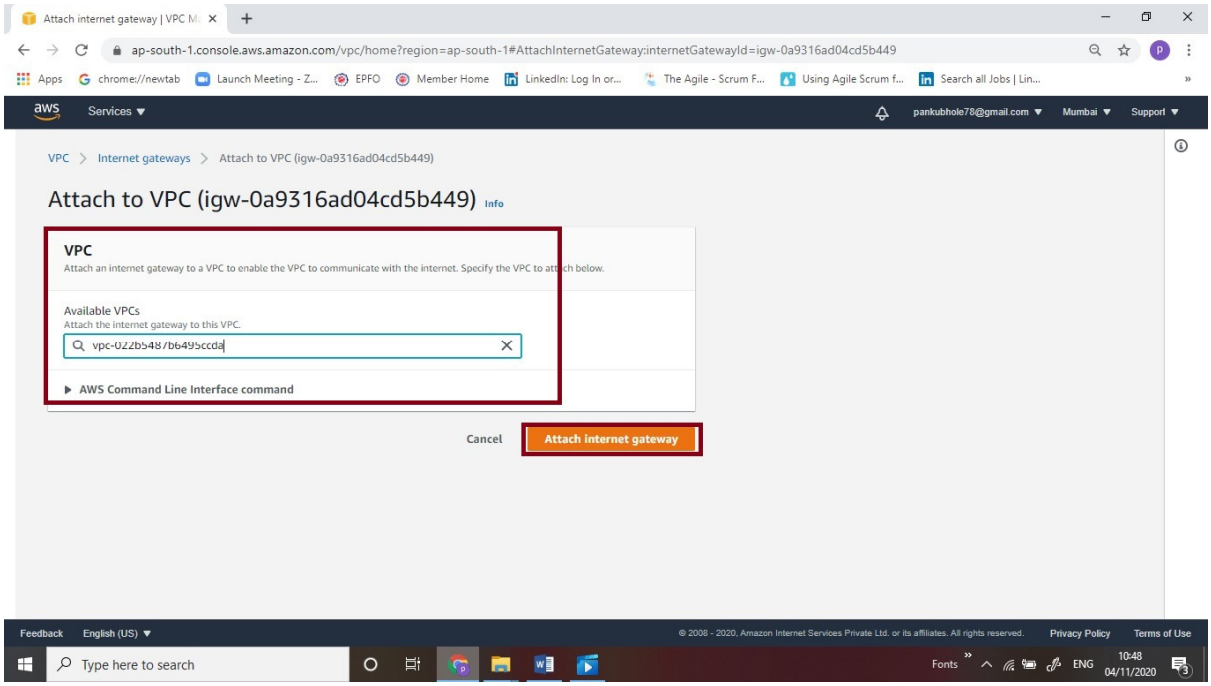


4. To attach Internet Gateway to a VPC,
    - Select the Internet Gateway *MyIGW*.
    - Click on **Actions**. Select **Attach to VPC**.
    - VPC : ***MyVPC***



Click on Attached VPC button.

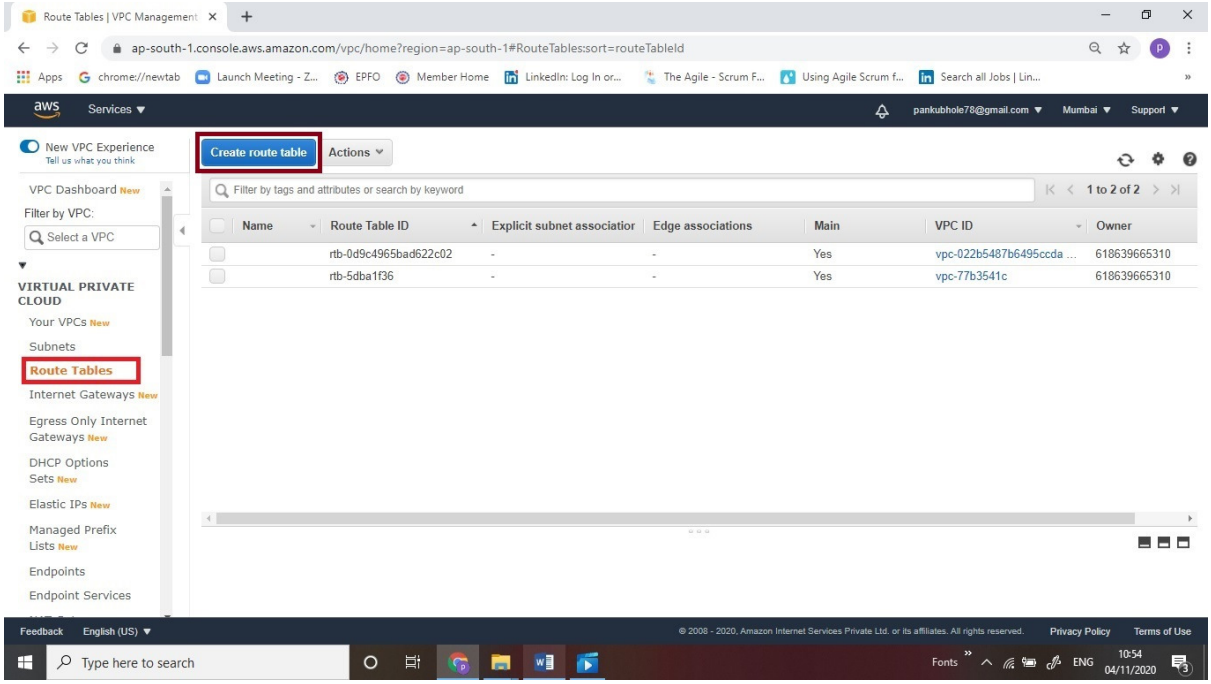
5. Now *MyIGW* is attached to *MyVPC*.



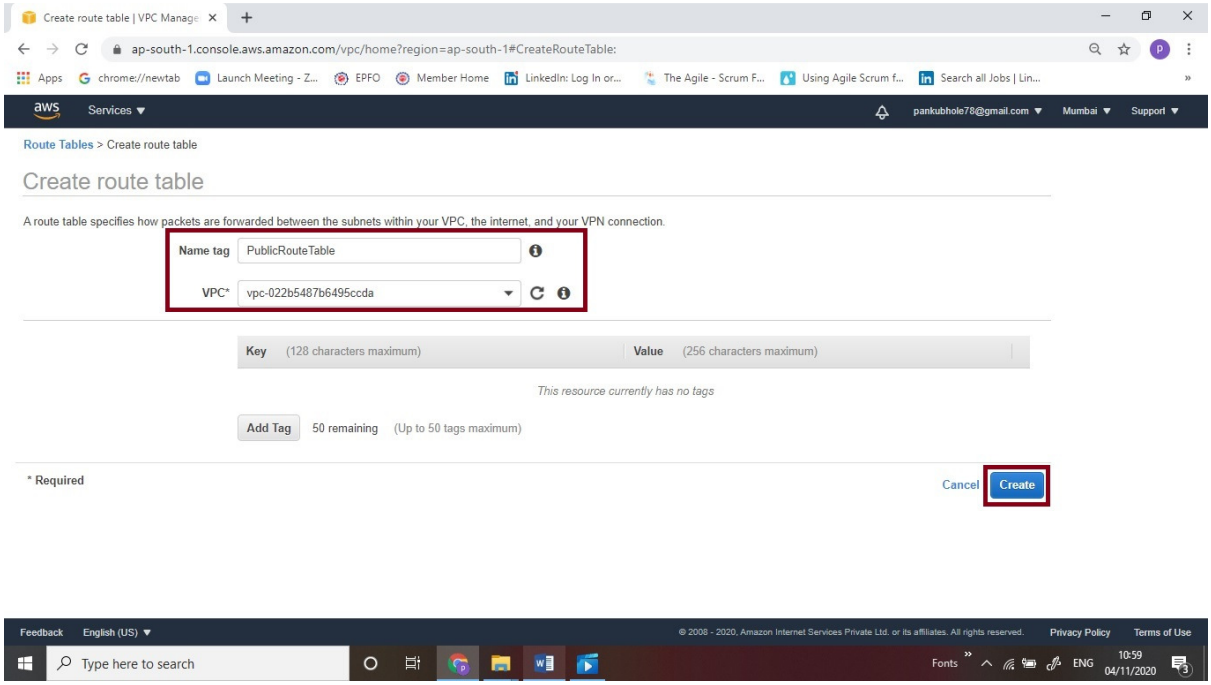
Create Public Route Table and Configure:

We will create a route table and attach a public subnet to it. Instances launched within this subnet will have access to the Internet.

1. Navigate to Route Table in the left panel of the **VPC** page.



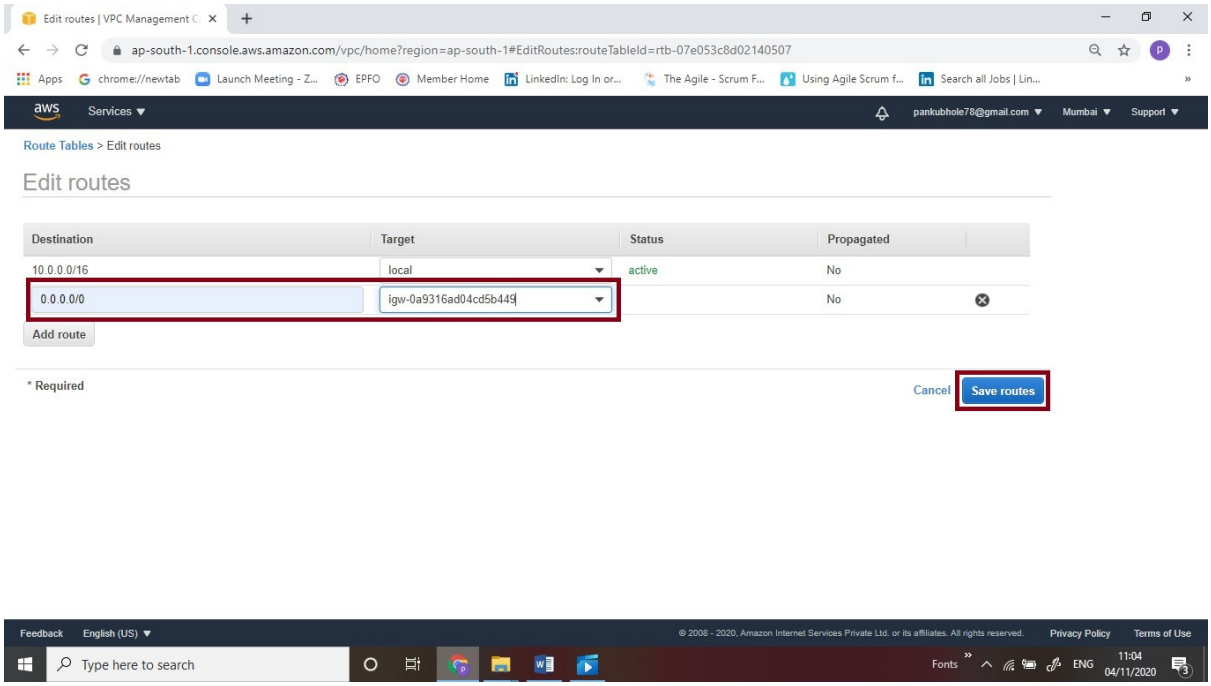
2. Click on to Create Route Table Button.
    - Name tag : ***PublicRouteTable***
    - VPC : ***MyVPC***
    - Click on Create button



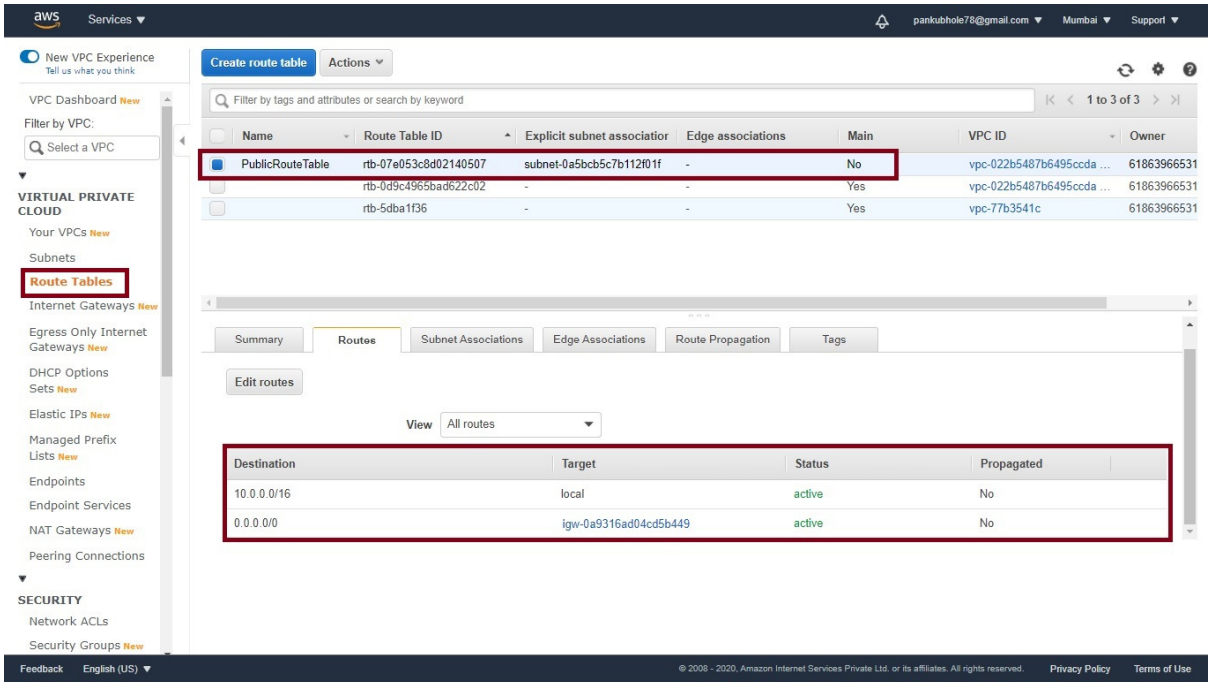
3. A route table by name *PublicRouteTable* will be created.
4. To attach an Internet Gateway, select *PublicRouteTable*.
5. In the **Routes** tab below:

- Click on Edit Route

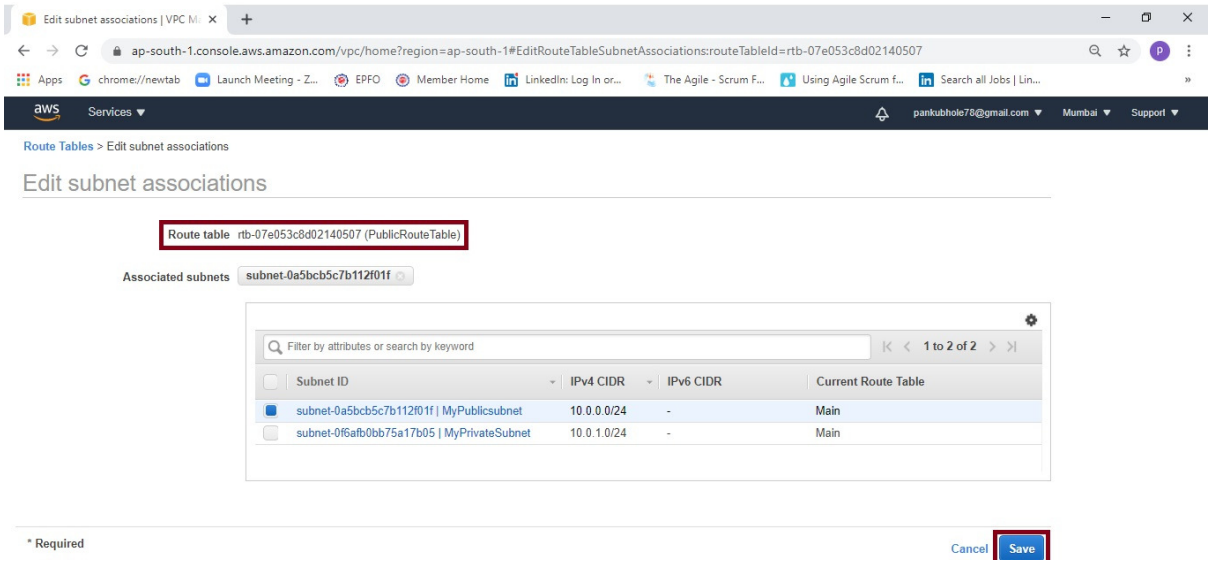
- On the next page, click on
- Destination : Enter **0.0.0.0/0**
- Target : Select **Internet Gateway**, and once the internet gateways have been created, select ***MyIGW***
- Click on Save Button.



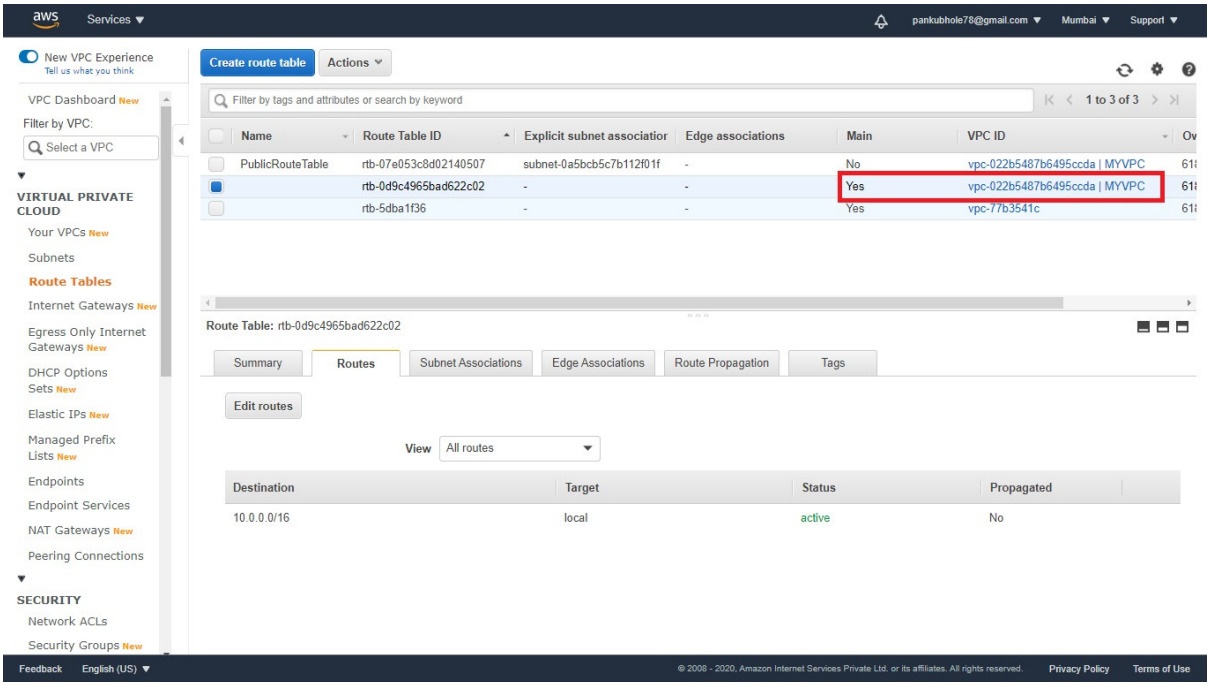
6. To associate the Public Subnet to the route table, Select *PublicRouteTable*.
   - Click on the Action and select **Edit Subnet Associations** tab.
   - Click on Save Button.
   - On the next page, select *MyPublicSubnet* from the list displayed.



Click on Save Button

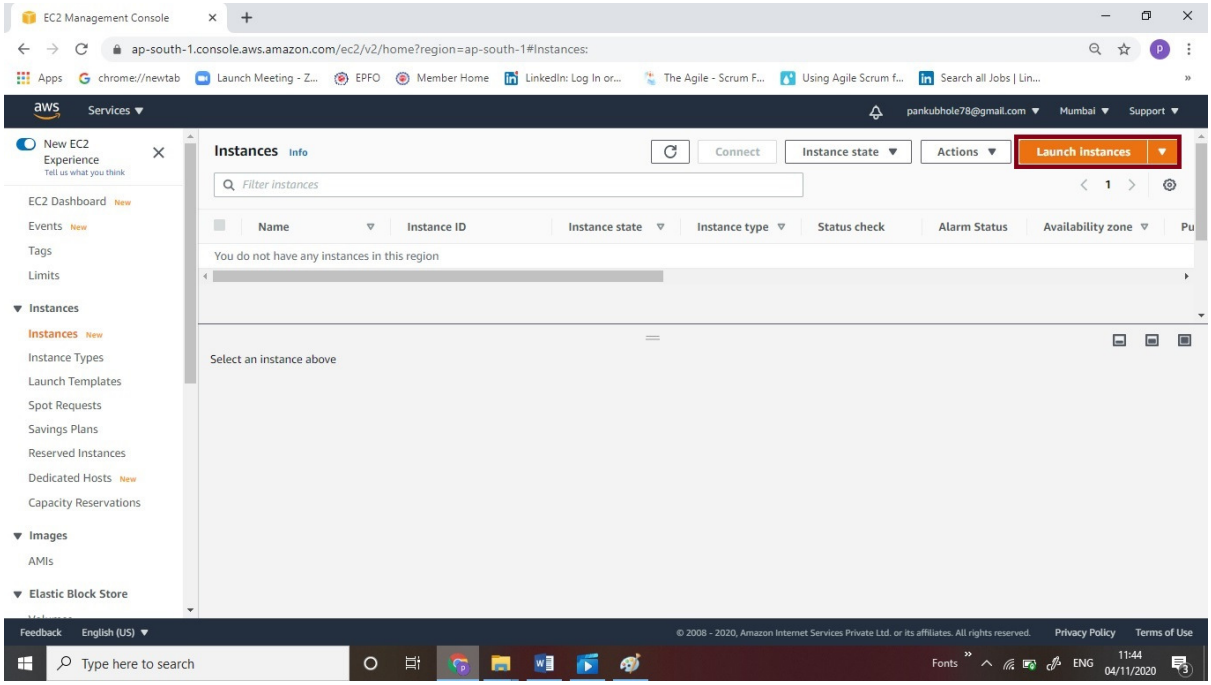


7. Once all the configurations are completed, it should look like below .selected details are Main Route Table details.
8. Now the Instances launched within *MyPublicSubnet* will have access to the Internet.
9. As you can see, there is another existing route table already available for *MyVPC*. It is a main route table created at the time the VPC was created. We will use it while creating the **NAT Gateway**.

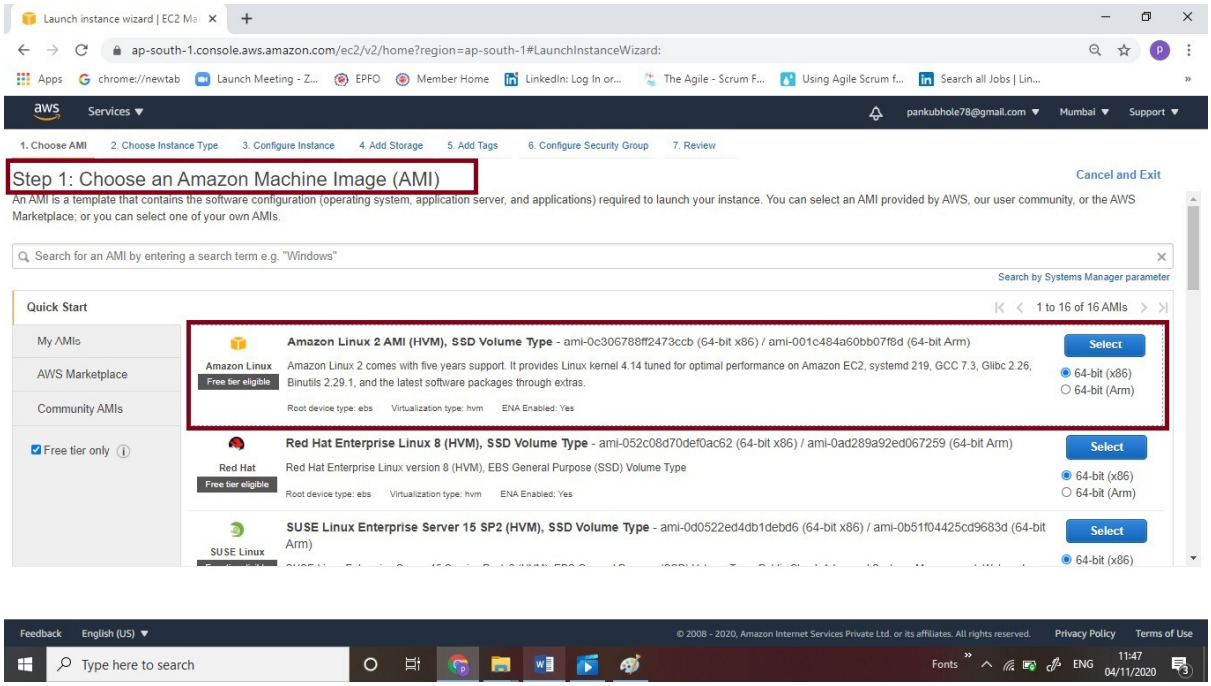


Launching an EC2 Instance in Public Subnet

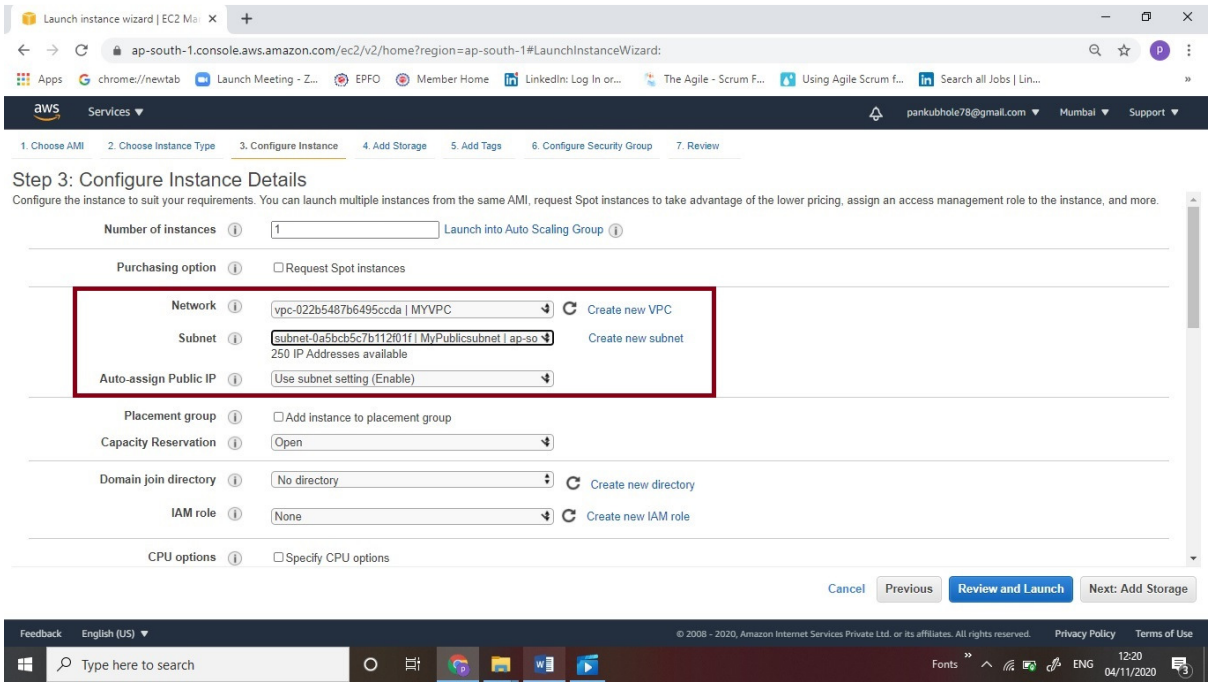
1. Make sure you are in the **Mumbai** region.
2. Navigate to the menu in the top, click on in the section.
3. Click on Launch Instance button.



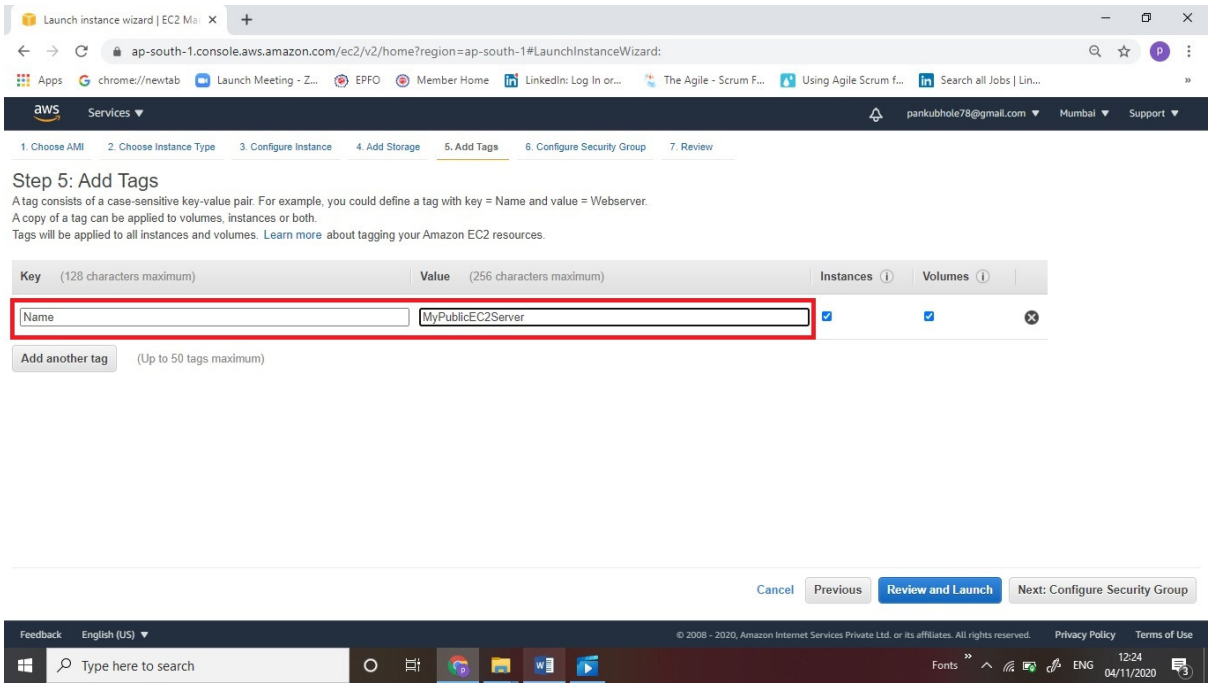
4. Choose an Amazon Machine Image (AMI): Search for **Amazon Linux 2 AMI** in the search box and click on the **select** button.



5. Choose an Instance Type: select **Amazon Linux 2 AMI** and click on the
6. Configure Instance Details:
    - Network : ***MyVPC***
    - Subnet : ***MyPublicSubnet***
    - Auto-assign Public IP : *Use Subnet Setting(Enable)* - default
    - Leave all other settings as default.
7. Click on Configure Instance details button.

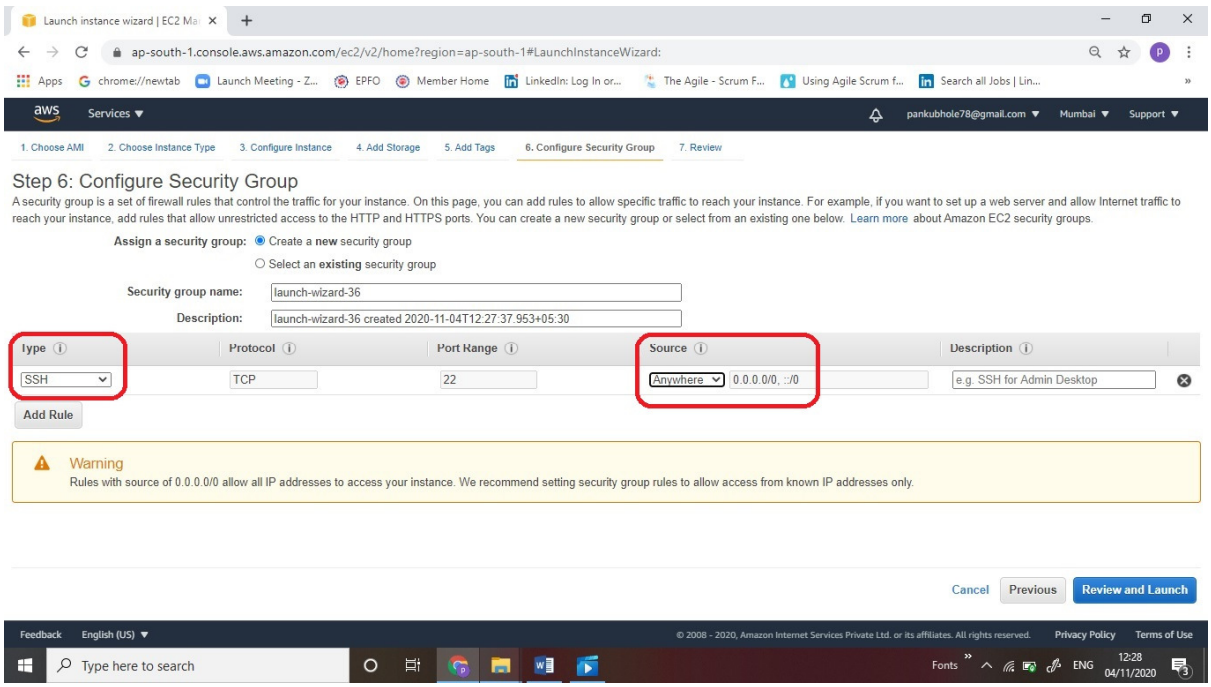


8. Add Storage: No need to change anything in this step. Click on configure security button and click Add Tag button.



9. Configure Security Group:
    - SSH is already available:
        - Choose Type: **SSH**
        - Source: **Anywhere**

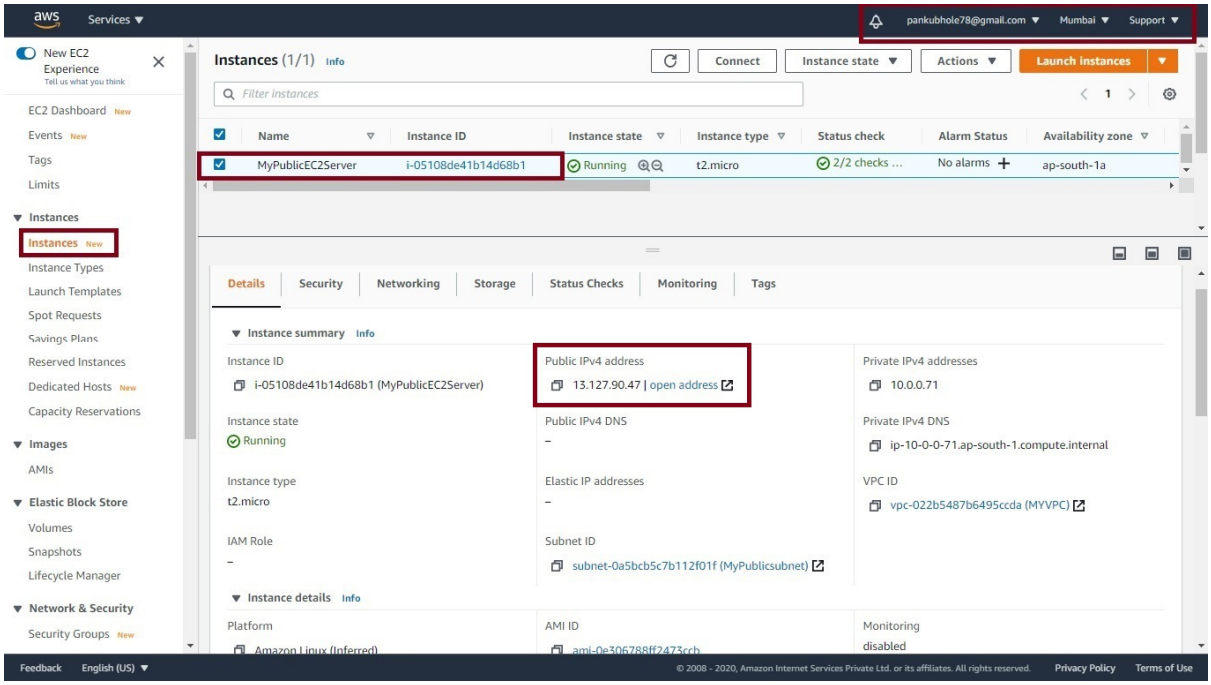
Click on Review and Launch button.



10. Review and Launch: Review all settings and click on Launch button.

11. Key Pair: Create a new key pair and click on .
    - Name KeyPair as ***MyKey.pem***

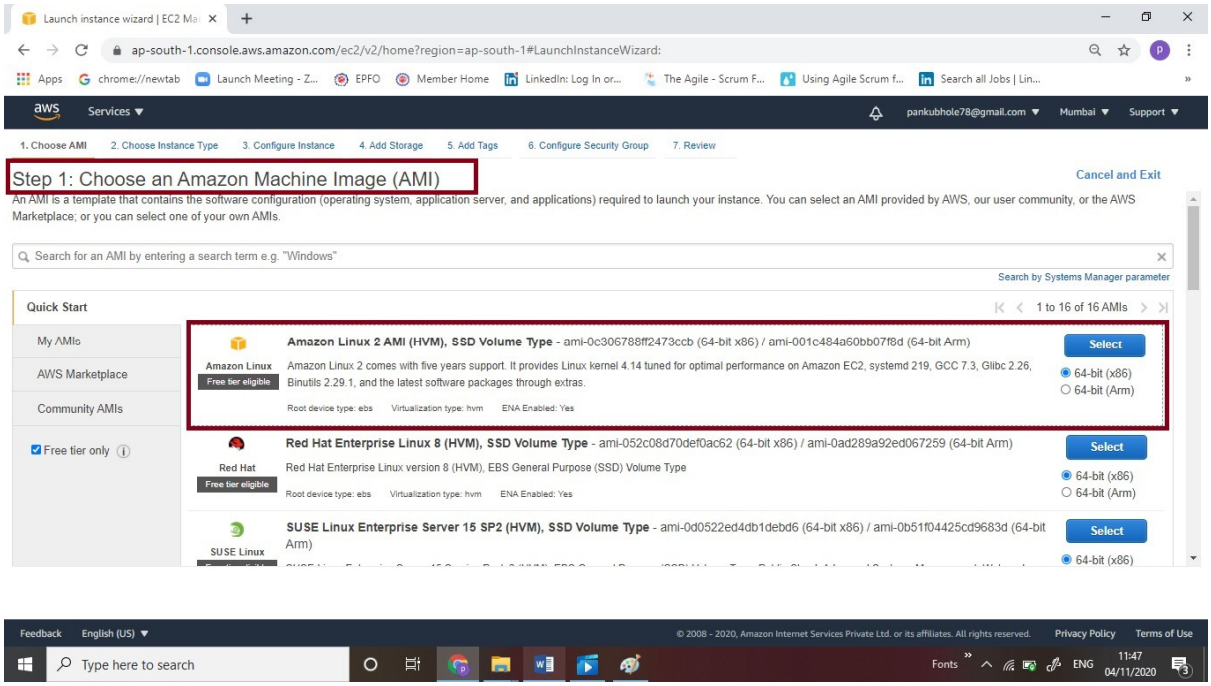
12. Click on Instance details.
13. Launch Status: Your instance is now launching, Select the instance and wait for it to change status to **Running 2/2 check.**



14. Note down the Public IP address of *MyPublicEC2Server*:**13.127.90.47.**

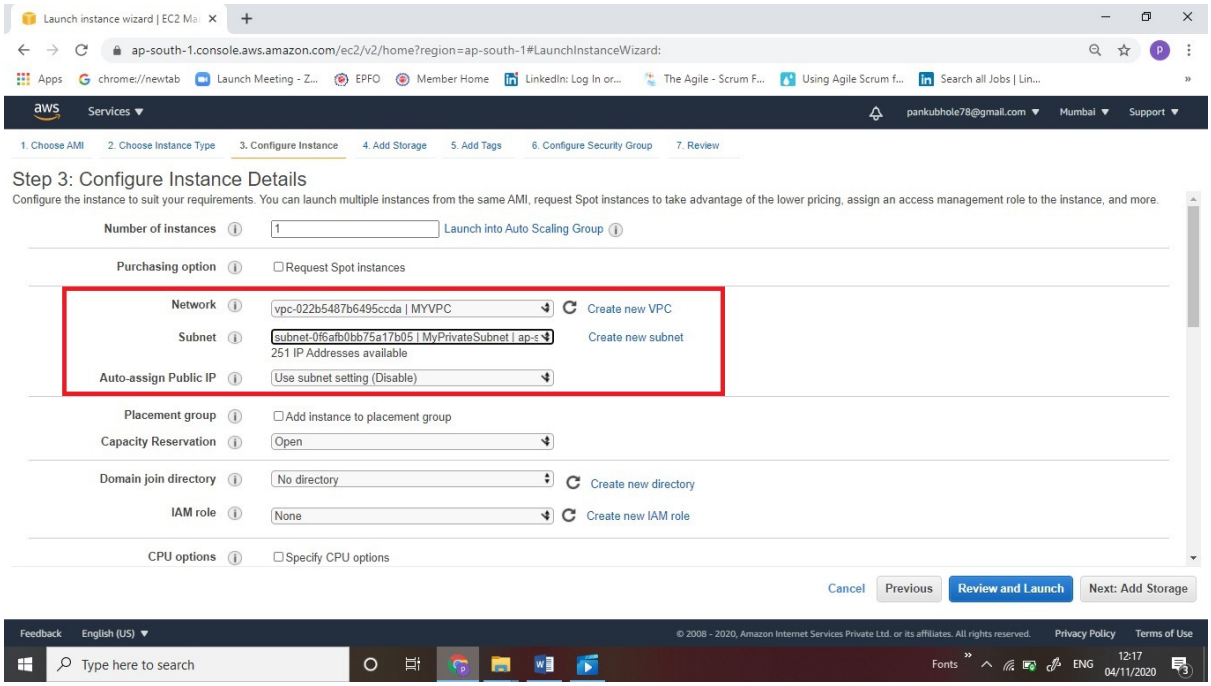
Launching an EC2 Instance in Private Subnet:

1. Click on Launch Instance button.
2. Choose an Amazon Machine Image (AMI): Search for Amazon Linux 2 AMI in the search box and click on the select button.
3. Choose an Instance Type: select Amazon Linux 2 AMI and click on the

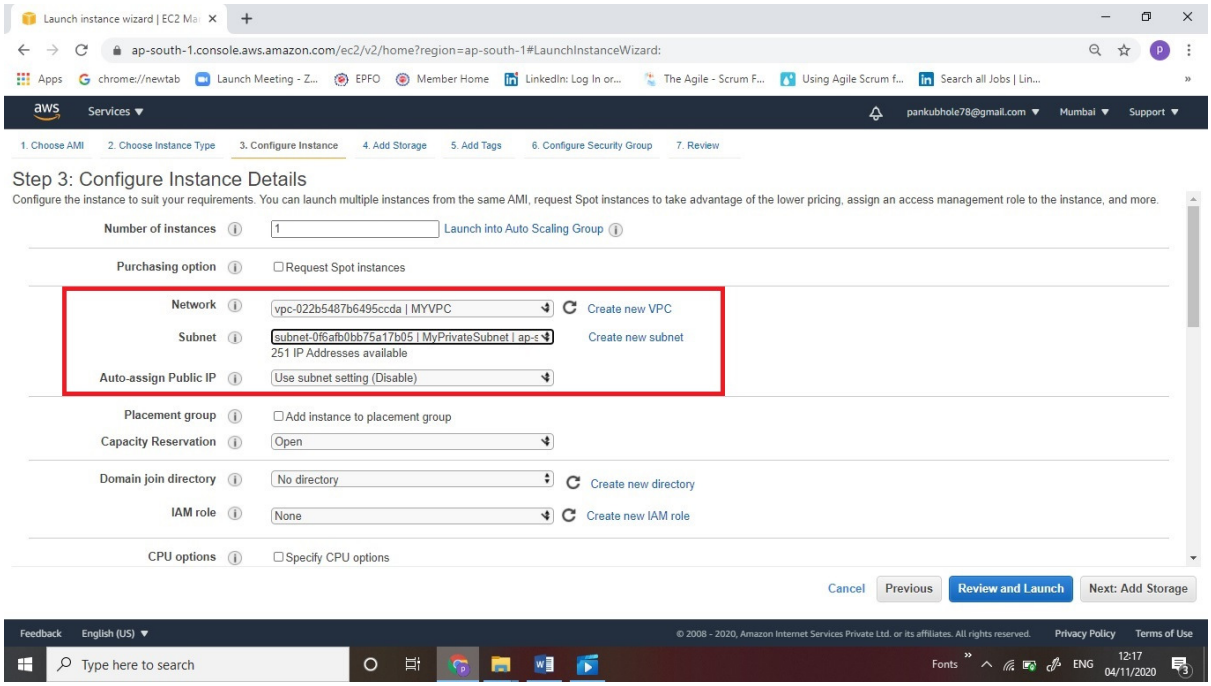


4. Configure Instance Details:

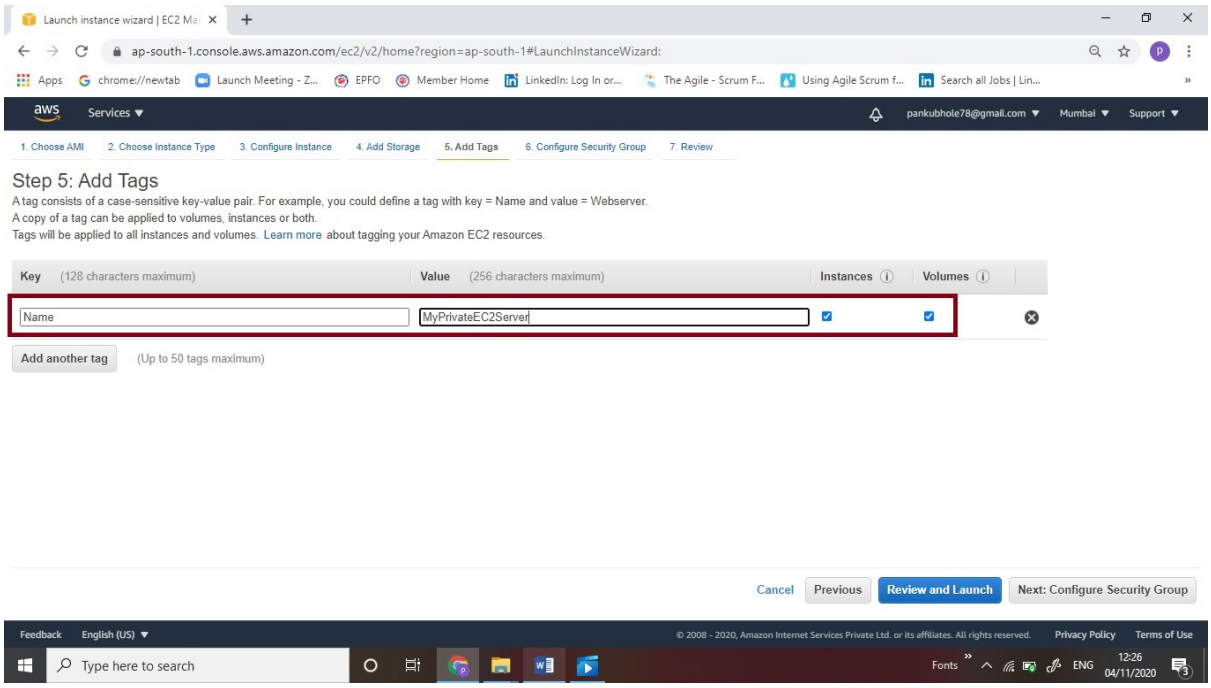
- o Network : MyVPC
- o Subnet : MyPrivateSubnet
- o Auto-assign Public IP : Use Subnet Setting(Enable) - default
- o Leave all other settings as default.



5. Click on Configure Instance details button.



6. Add Storage: No need to change anything in this step. Click on configure security button and click Add Tag button.



7. Configure Security Group: this step same as above instance Publicserver.

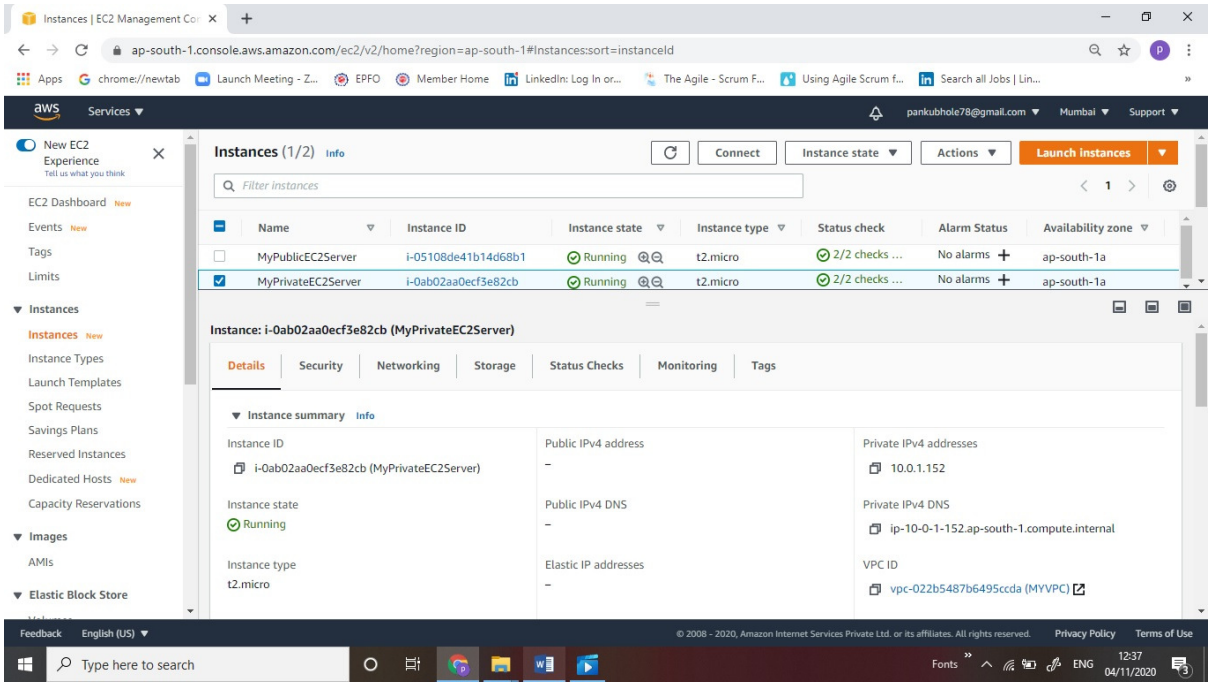
8. Review and Launch: Review all settings and click on Launch button.

9. Key Pair: Create a new key pair and click on.

O Name KeyPair as MyKey.pem

10. Click on Instance details

11. Launch Status: Your instance is now launching, Select the instance and wait for it to change status to **Running 2/2 check**.

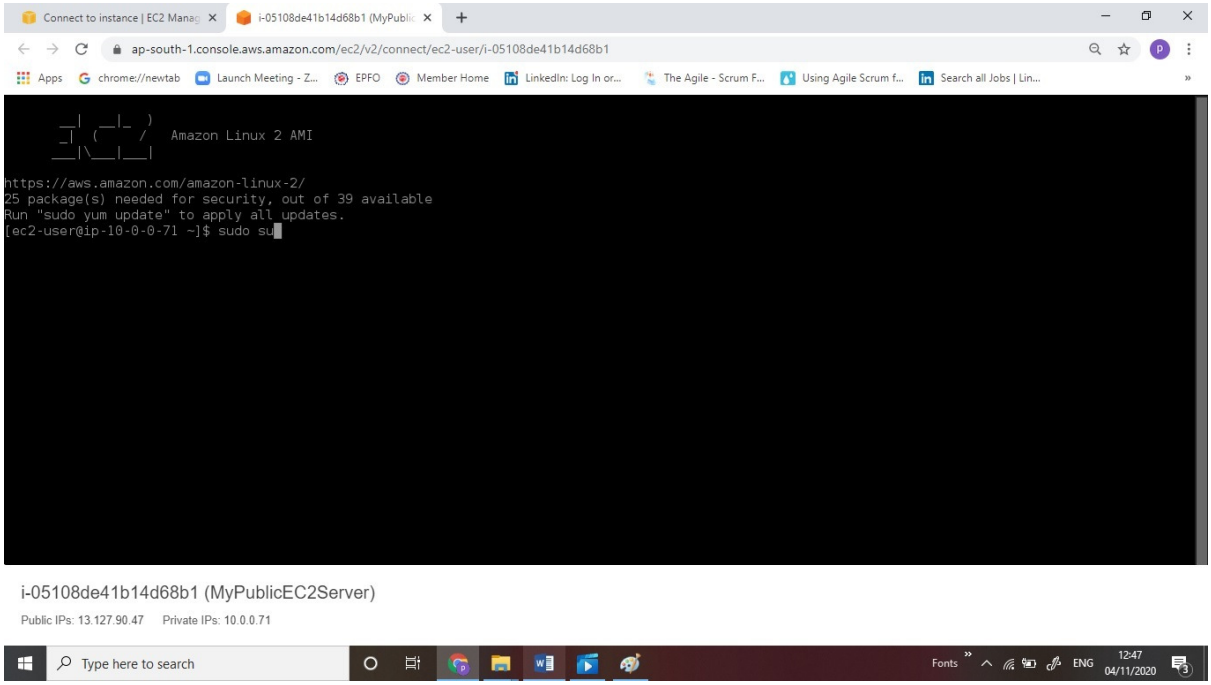


12. Note down the Private IP address of MyPrivateEC2Server:10.0.1.152.

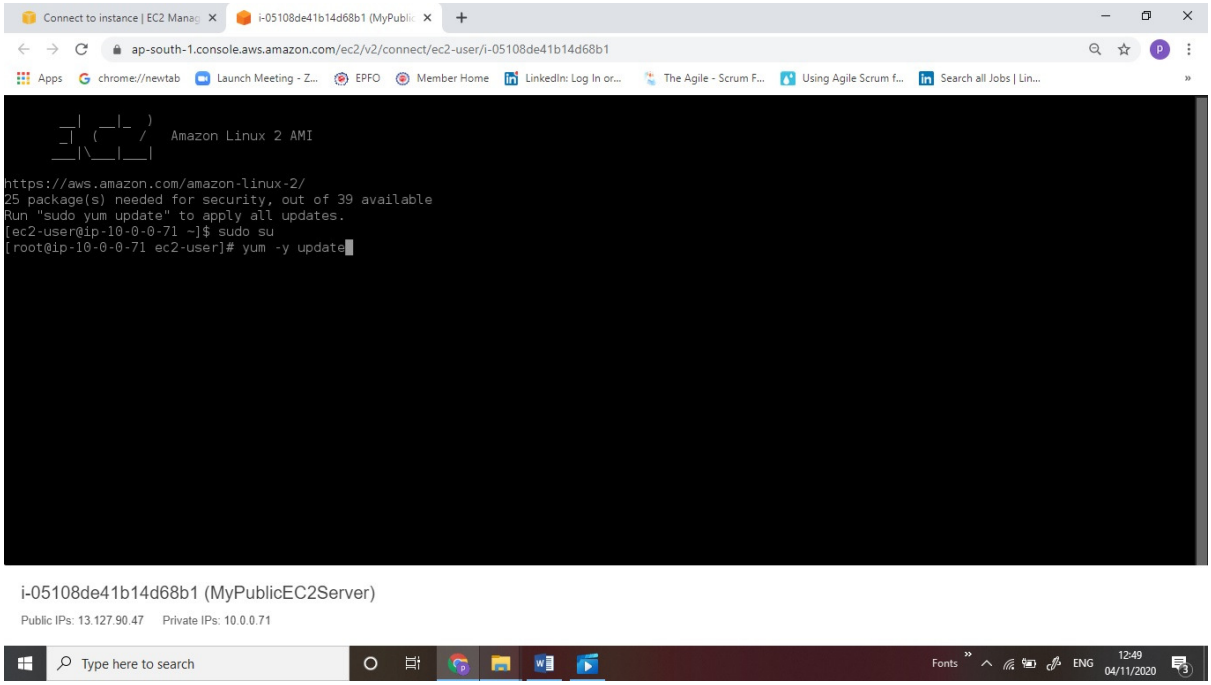
13. Two servers are now launched and ready.

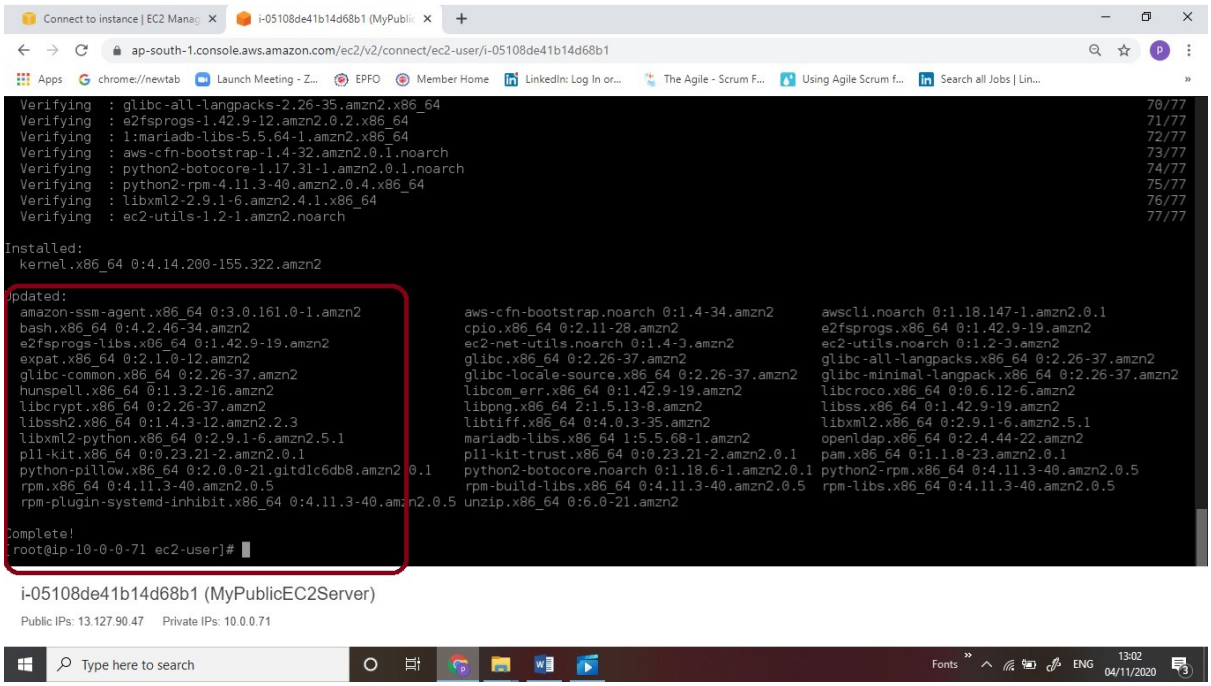
SSH into Public and Private EC2 Instance and Test Internet Connectivity:

1. Using the public IP address, SSH into *MyPublicEC2Server.*
2. Please follow the steps in SSH into EC2 Instance.
3. Switch to root user: **sudo –su**

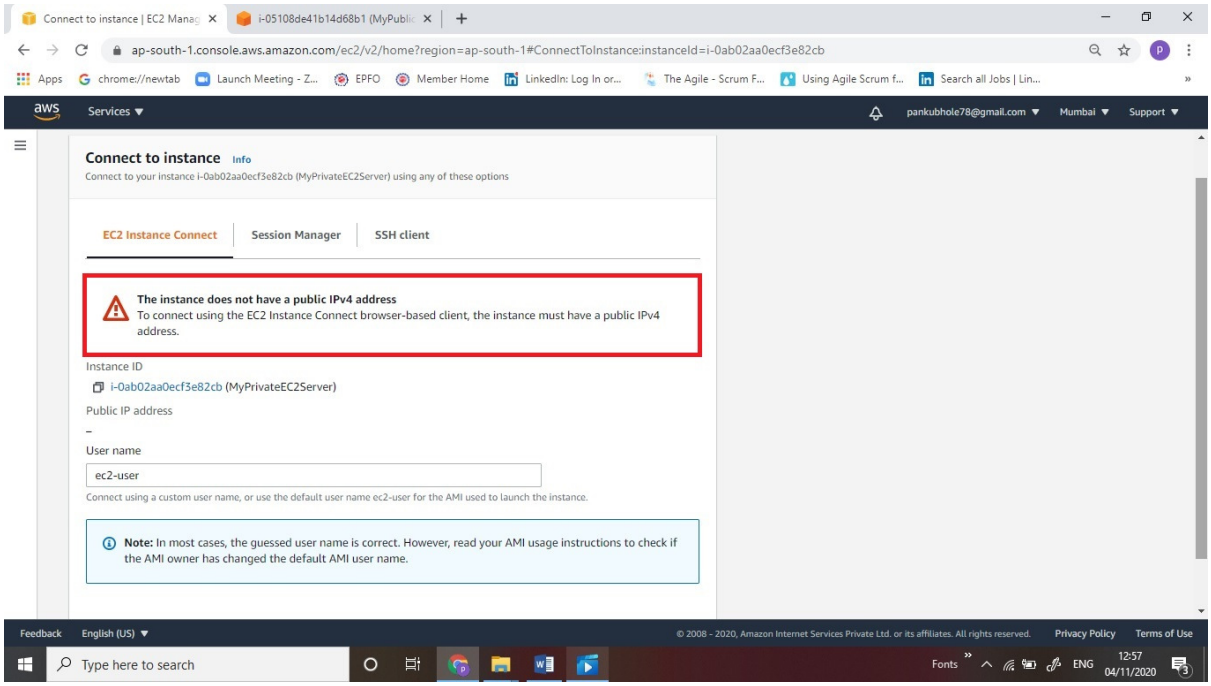


4. Run the updates using the following command:
    - **yum -y update**



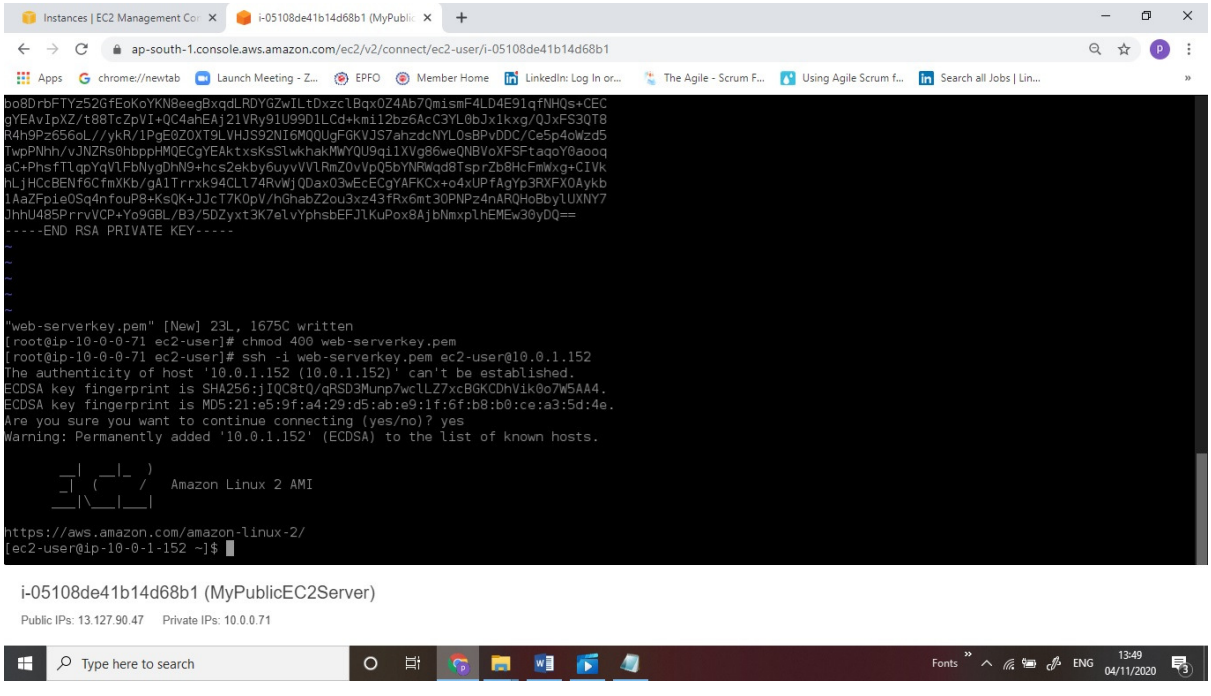


5. Since the Internet Gateway *MyIGW* is connected to *MyPublicSubnet*, updates will be completed successfully.
6. Let's SSH into ***MyPrivateEC2Server*** from ***MyPublicEC2Server***.
7. In order to SSH into ***MyPrivateEC2Server,*** first we need to create the PEM file in the public EC2 ie, ***MyPrivateEC2Server*** and copy the data from our *MyKey.pem* in the local machine.



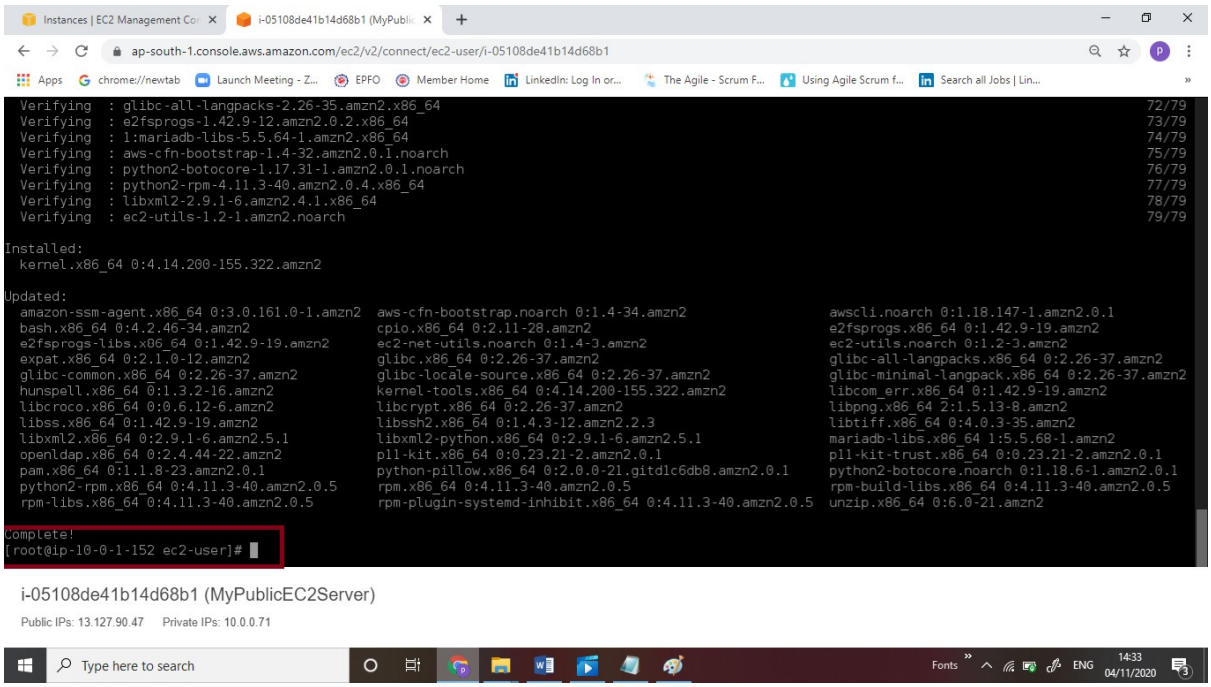
When connect the privateEC2 server then system is showing on above screen selected error message.

8. We need *MyKey.pem* inorder to SSH. We shall copy key details from the MyKey.pem from your local machine (which was downloaded earlier while launching EC2 instances).
9. To create the MyKey.pem in ***MyPublicEC2Server***, run
    - **nano MyKey.pem**
10. In the editor, copy and paste the key that looks similar to the example below



11. Update Permissions for the MyKey.pem
    - **chmod 400 MyKey.pem**

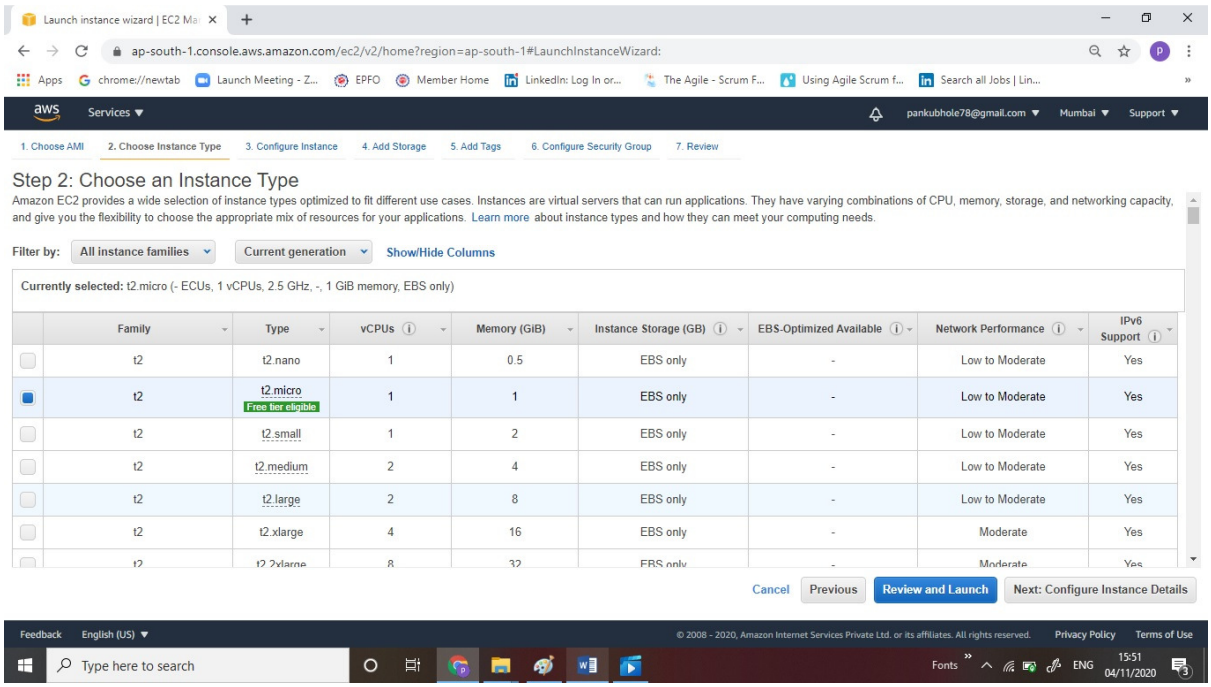
12. Use the Private IP address **10.0.1.152**to SSH.
    - **ssh -i MyKey.pem ec2-user@10.0.1.152**
    - **Note**: Incase if this message shows **Are you sure you want to continue connecting (yes/no)?** : Enter **yes**
13. Switch to root user
    - **Sudo su**
14. Run the updates using the following command:
    - **yum -y update**
15. Since no internet access is provided for EC2 instances in a private subnet, you will not be able to get updates. After some time, it will fail with the following message.



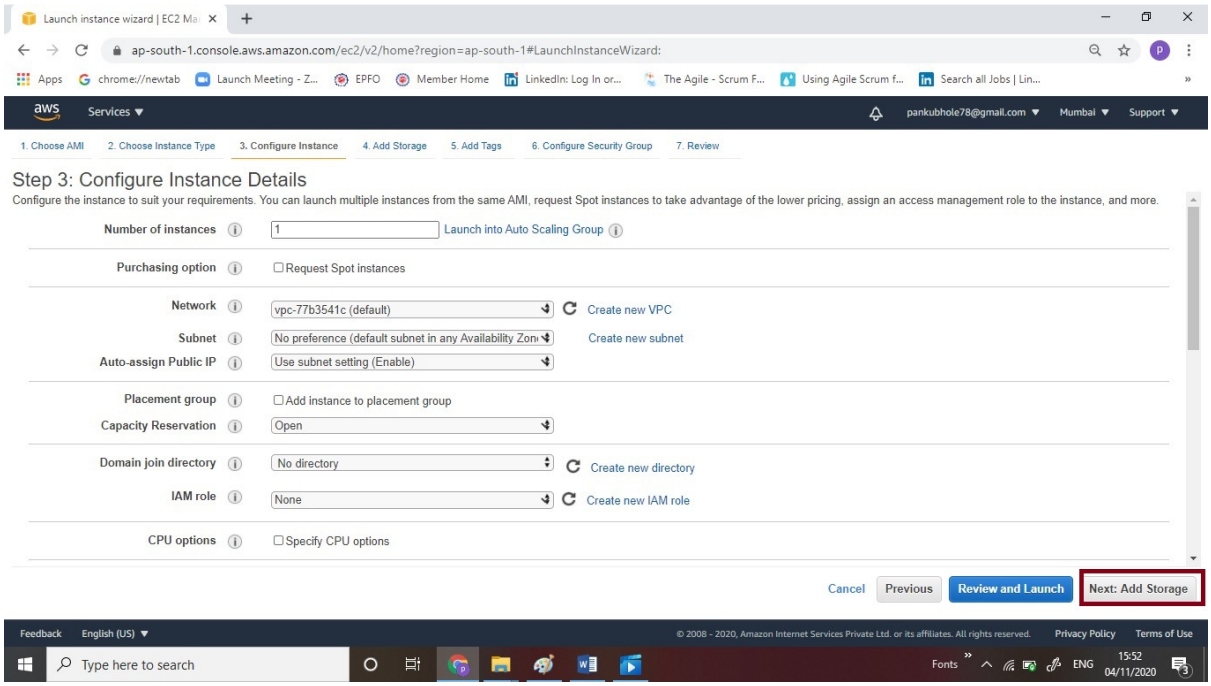
16. You can see that the updates have been completed successfully in the terminal.
17. This shows that ***MyPrivateEC2Server*** has internet access.
18. Use **exit** command to close the private server connection.

Steps to Create Bastion Server:

1. Make sure you are in the **N.Virginia** Region.
2. Navigate to EC2 by clicking on the menu at the top, then click on in the section and click on Launch instance button.
3. Choose the first Amazon Machine Image (AMI)**: Amazon Linux 2 AMI (HVM), SSD Volume Type** click on the **Select** button
4. Instance Type : **t2.micro**

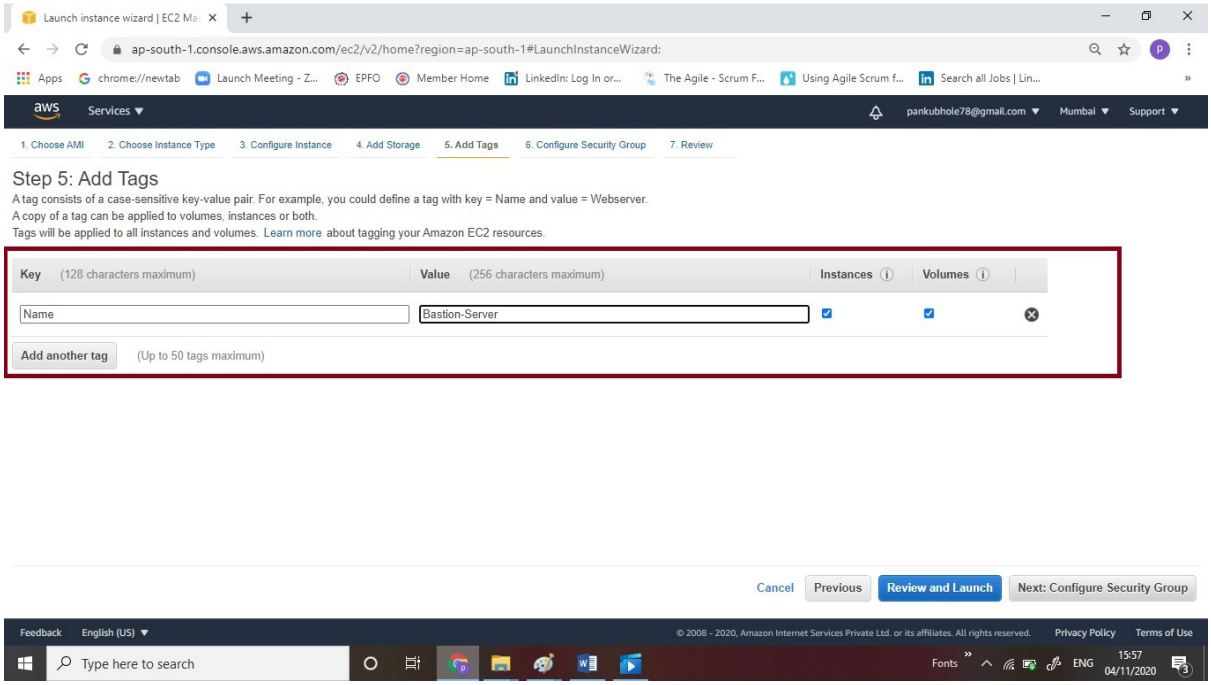


5. In the **Configure Instance Details**, leave all fields with the default values and then click on configure instance details button.

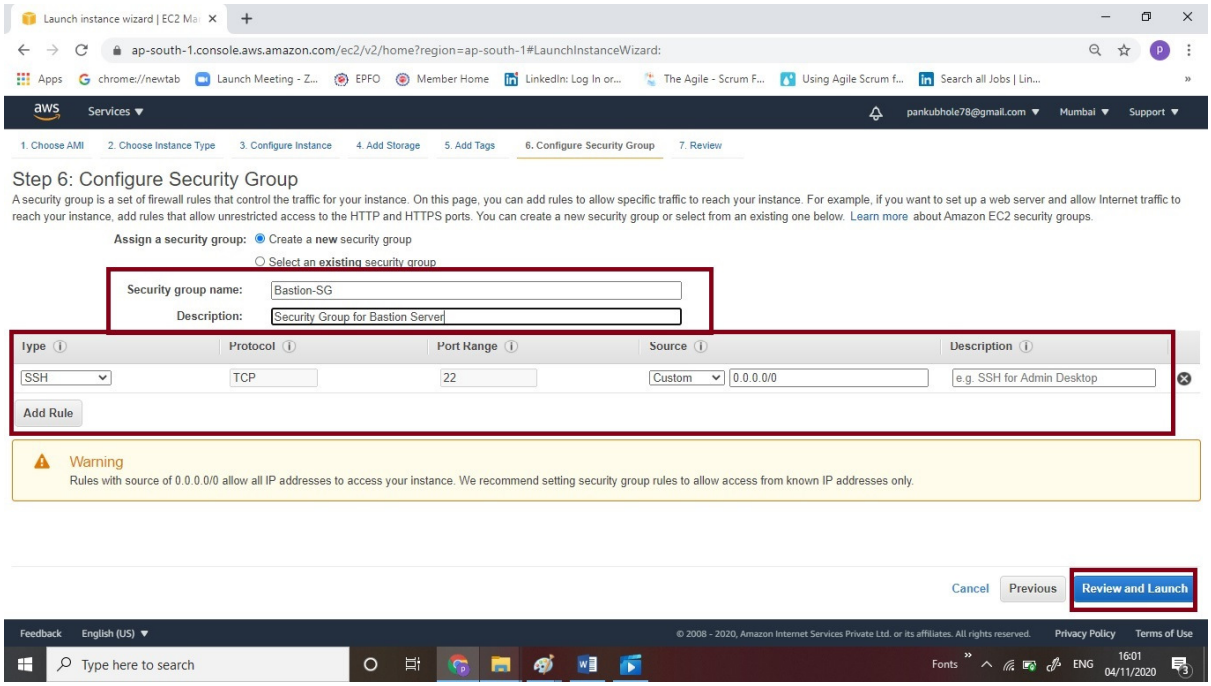


6. No need to change anything in this step, click on Add storage details.

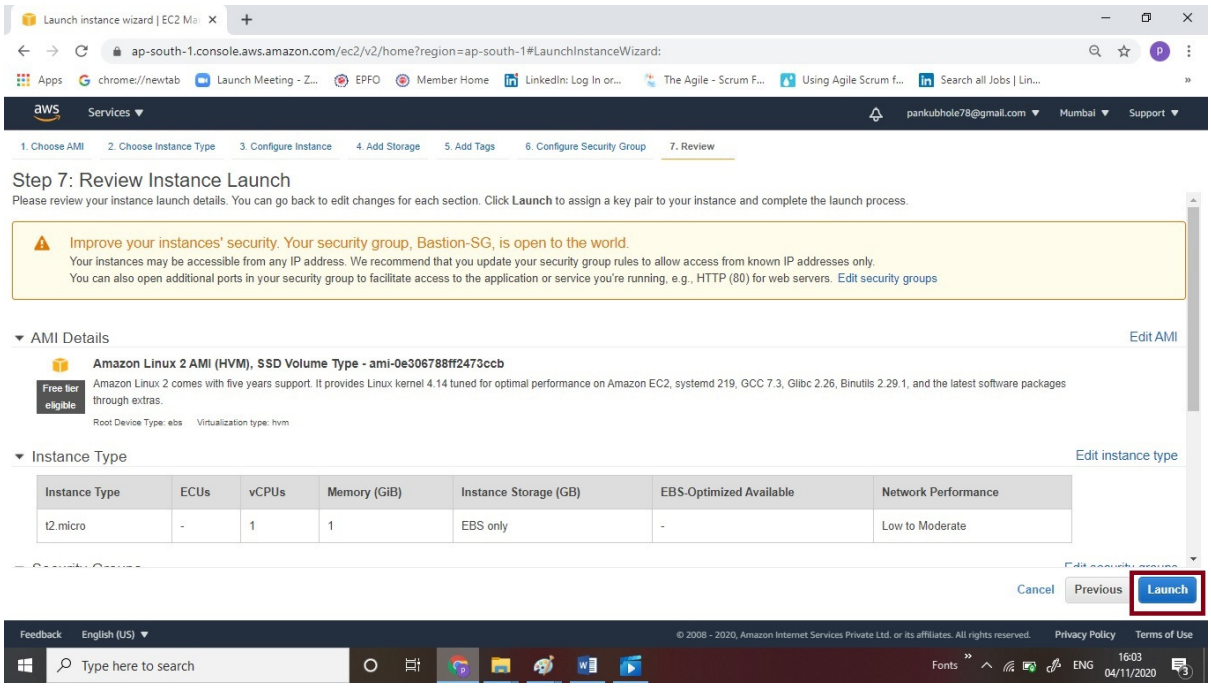
7. **Add Tags:** Click on
    - Key : ***Name***
    - Value : **Bastion-Server**



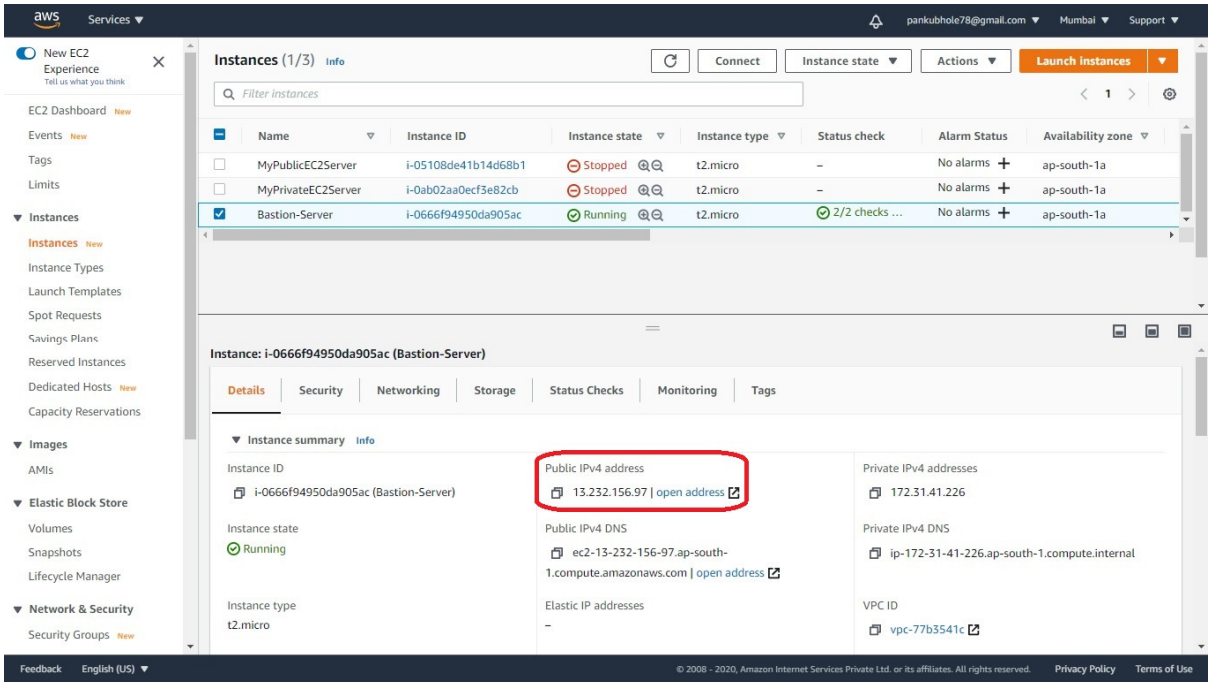
8. **Configure Security Group:**
    - Assign a security group: Choose to **Create a new security group**
    - Security group name: **Bastion-SG**
    - Description: **Security group for Bastion-server**
    - To add **SSH:**
    - Choose Type : SSH
    - Source : **Custom(Allow specific IP address) - 0.0.0.0/0**



10. **Review and Launch:** Review all settings and click on .
11. **Key Pair:** Create a new key pair named **bastionkey,** click on Launch button.



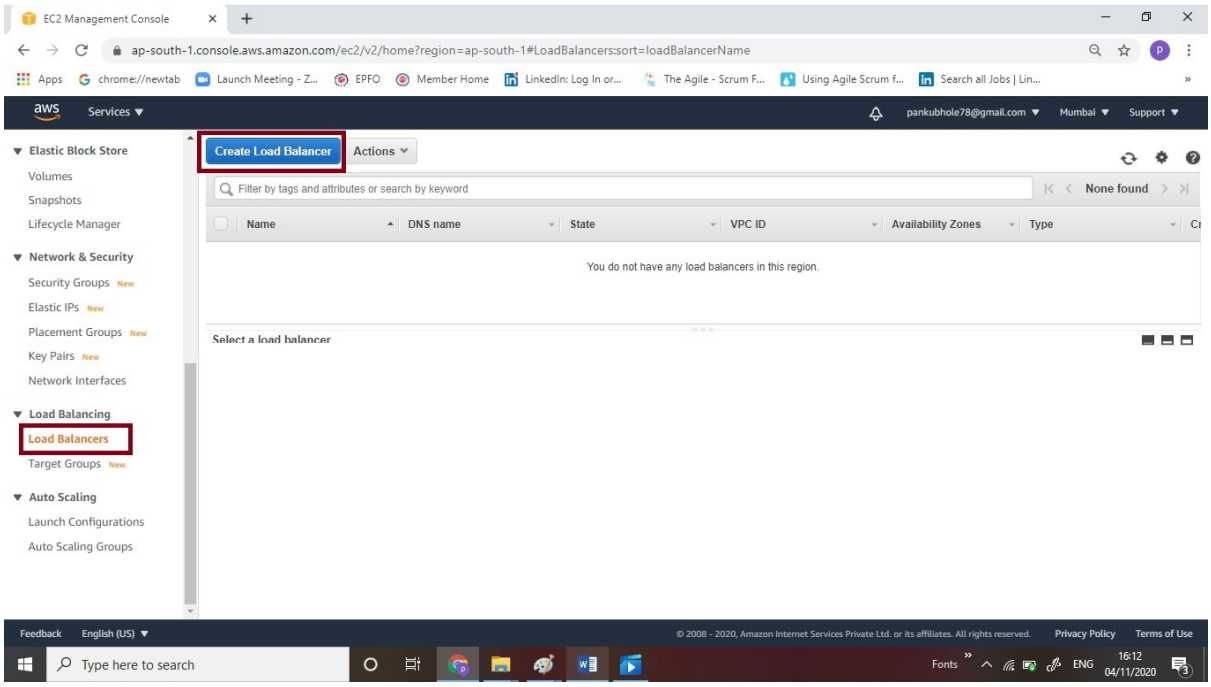
12. Navigate to **Instances** and wait for 1-2 minutes (until the Bastion-server's status changes from pending to running).



- **Bastion-server Public IP:** 13.232.156.97.

Creating a Security Group for the Load Balancer:

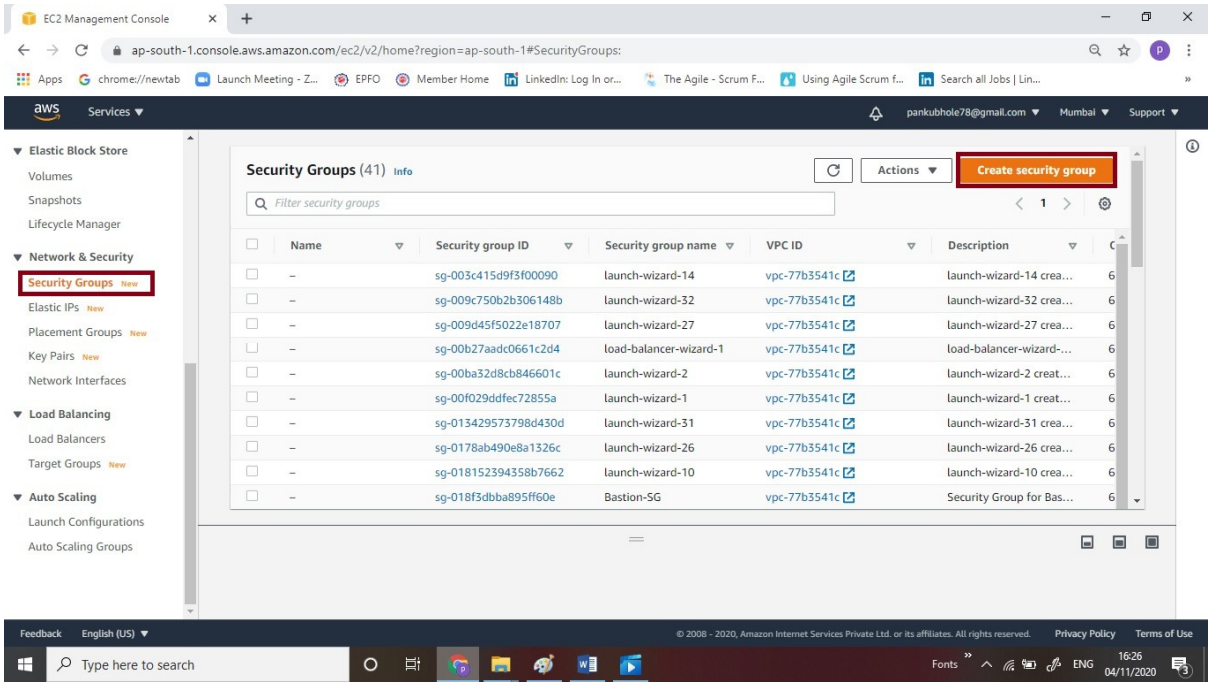
1. Navigate to the Ec2 Dashboard, scroll down to in left menu and click on Create Load Balancer button

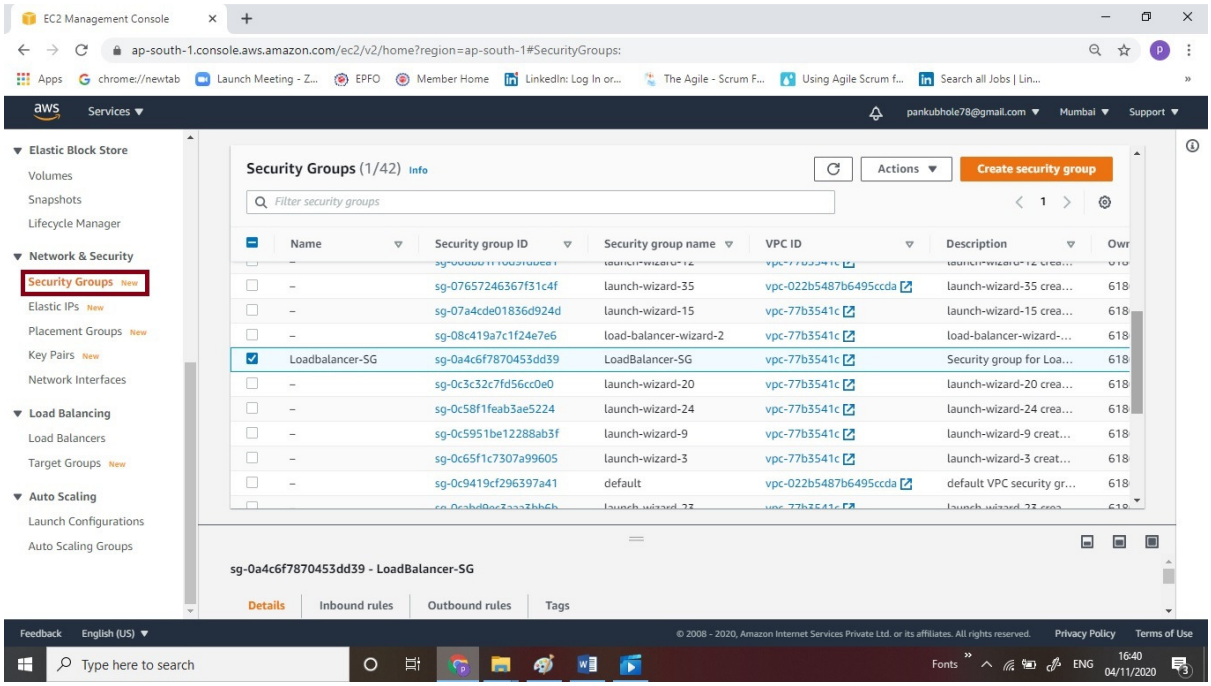


2. **Configure the security** group as follows:

   Security group name: **LoadBalancer-SG**

   - Description: **Security group for the Load balancer**
   - VPC: **Leave as the default**
   - Click on Create Security group and add the port as follows:
   - Type: **HTTP**
   - Make sure, Protocol is **TCP** and **Port range** is **80**
   - Source: Custom and enter **0.0.0.0/0**



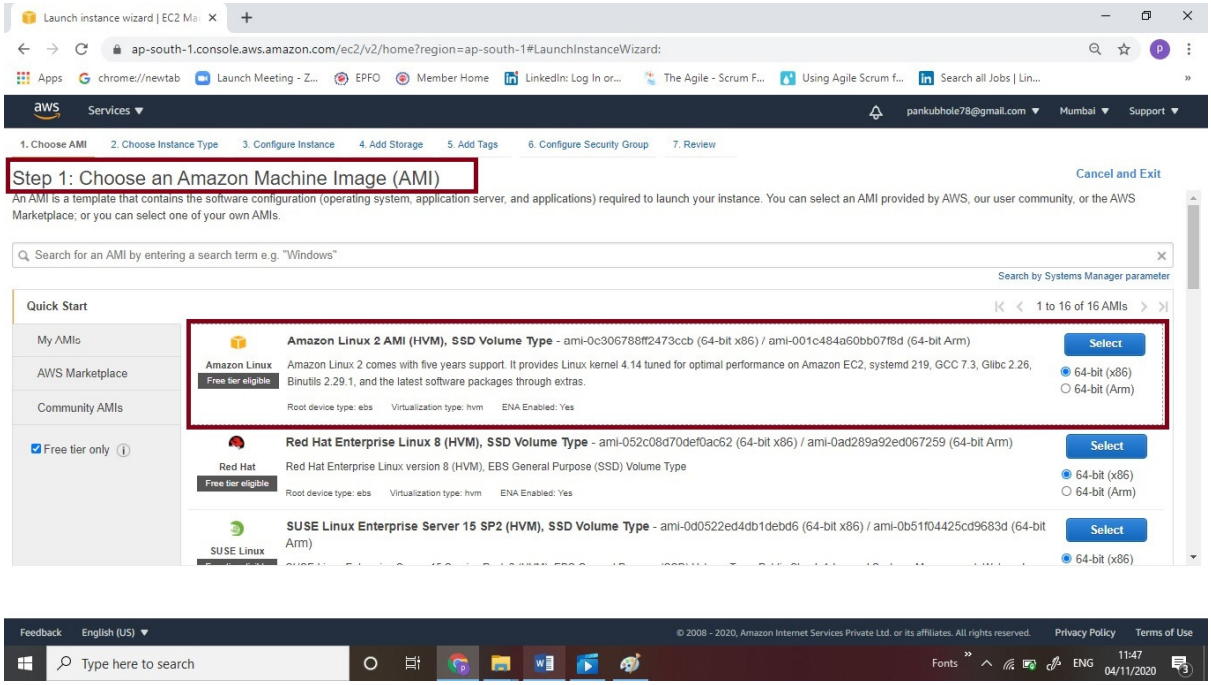


3. Leave everything by default in **Outbound rules** and **Tags - optional**

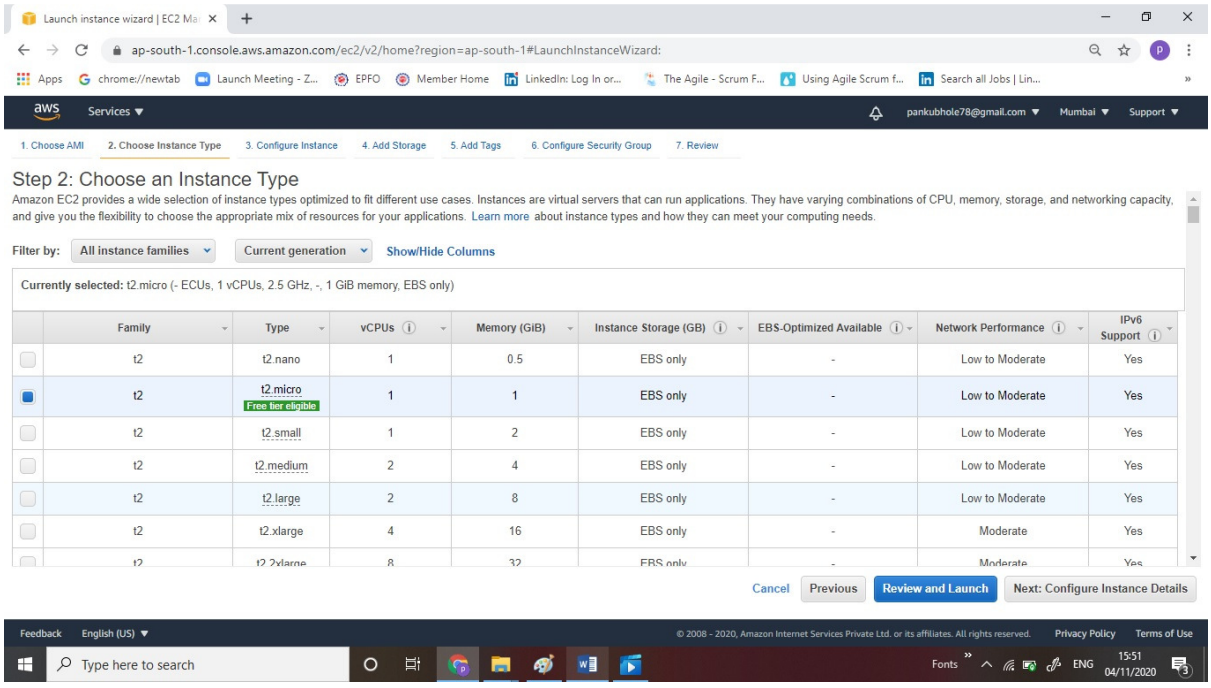
4. Click on save button.
5. The security group for the load balancer will be created see as above screen.

Steps to create Web-servers:

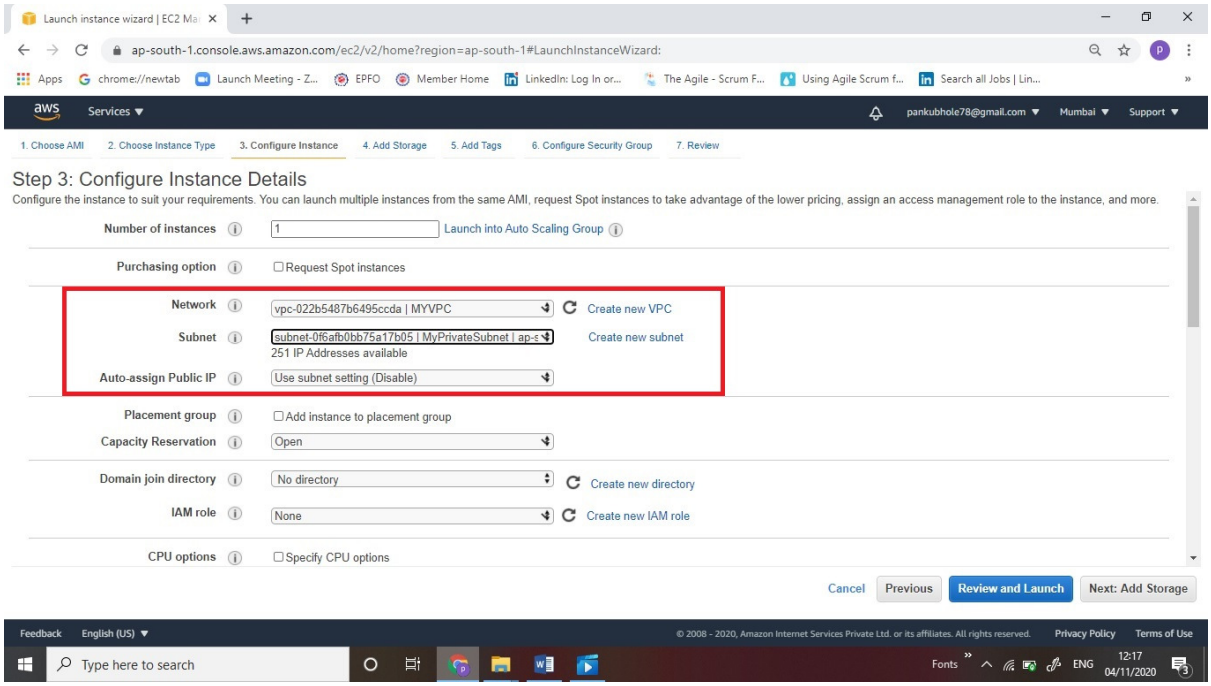
1. Click on Launch **instance button**.
2. Choose the first Amazon Machine Image (AMI)**: Amazon Linux 2 AMI (HVM), SSD Volume Type** click on the **Select** button.



3. Instance Type : **t2.micro**

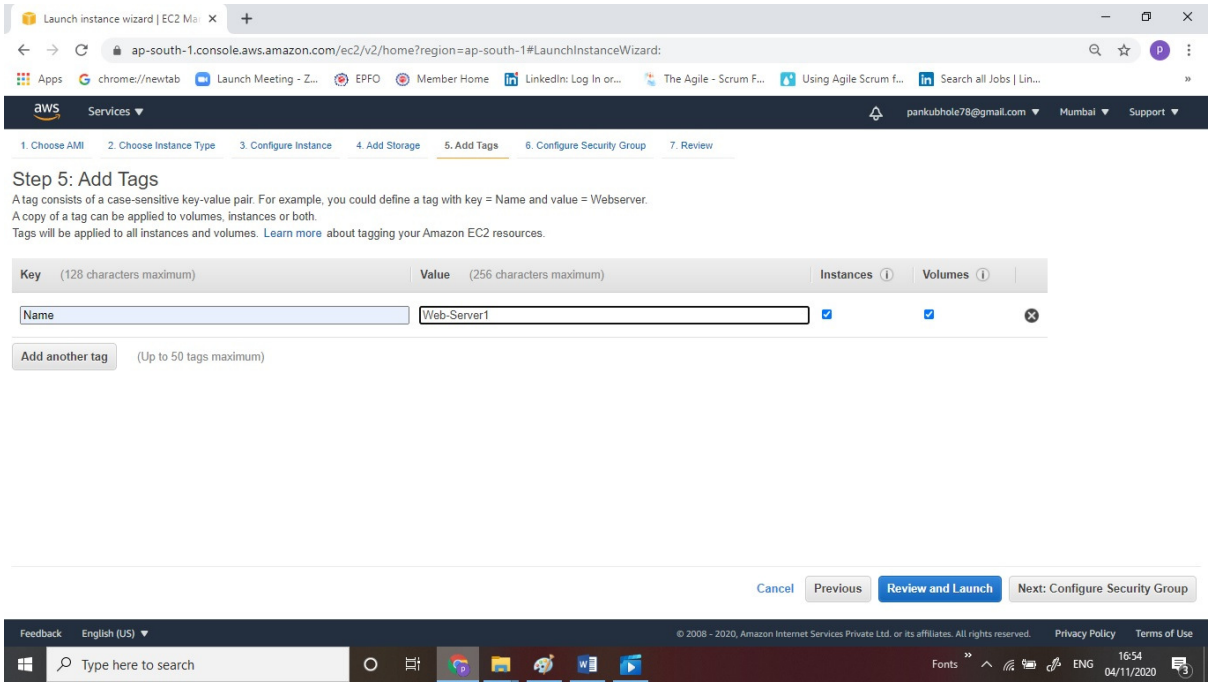


4. In Configure Instance Details scroll down to Subnet and choose the Private subnet as shown below:



- Leave other fields default and click on Add Storage button.

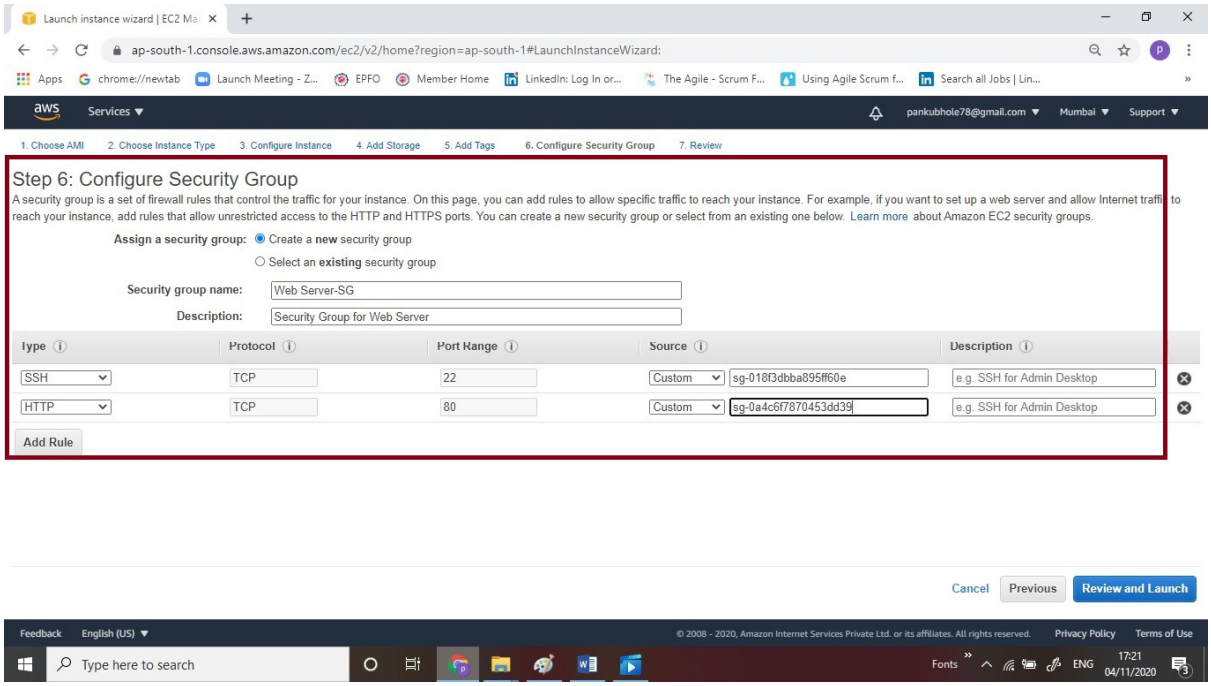
6. **Add Storage:** No need to change anything in this step, click on Configure Security Group.



7. **Add Tags:** Click on

- Key : ***Name***
- Value : **Web-server-1**
- Click on : **Configure Security Group**

8. **Configure security group:**



Create New Security Group details as follow:

- Name**: web-server-SG**
- Description: **Security group for web servers**
- On **port 22**, we choose the **Bastion-SG security group as its source to allow SSH connection to web servers from only the bastion server** by restricting the public SSH connection. Type bastion in source and select the **Bastion-SG.**
- On port 80, choose the **Load Balancer-SG as its source** to serve the traffic coming through the load balancer. Type Load in source and select **Load Balancer-SG.**

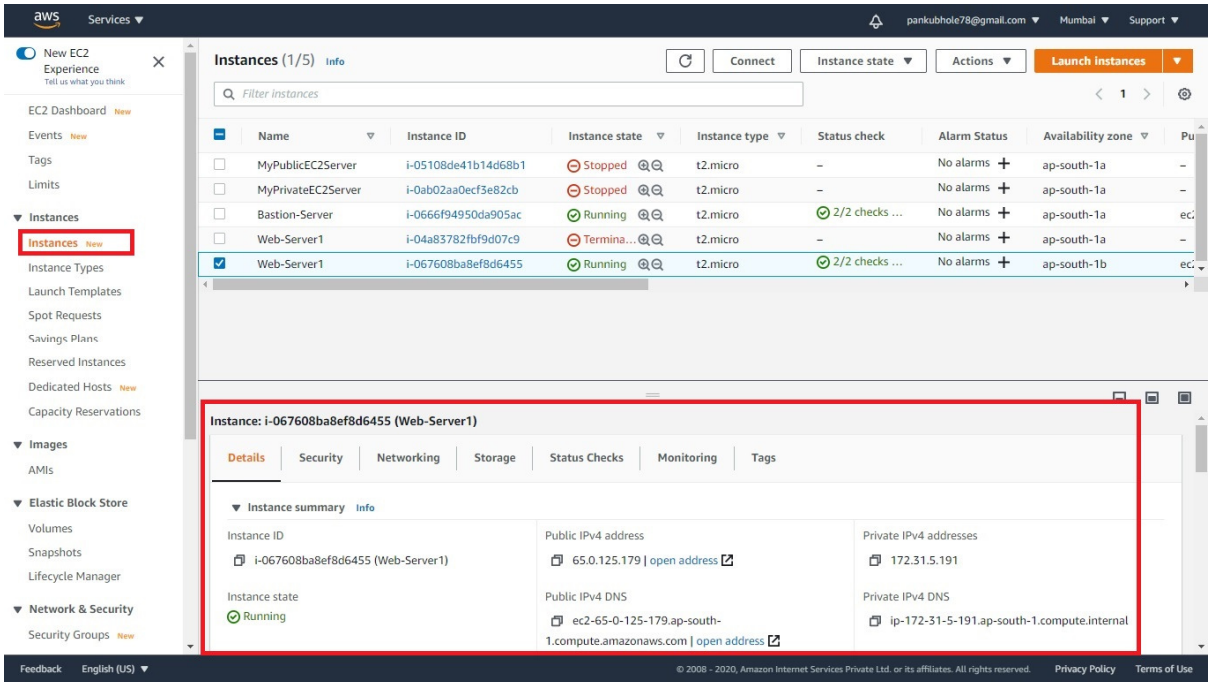
- To add **SSH:**

  - Choose Type **: SSH**
  - Source: **Bastion-SG**

- To add **HTTP:**
  - Choose Type**: HTTP**
  - Source: **Load Balancer-SG**

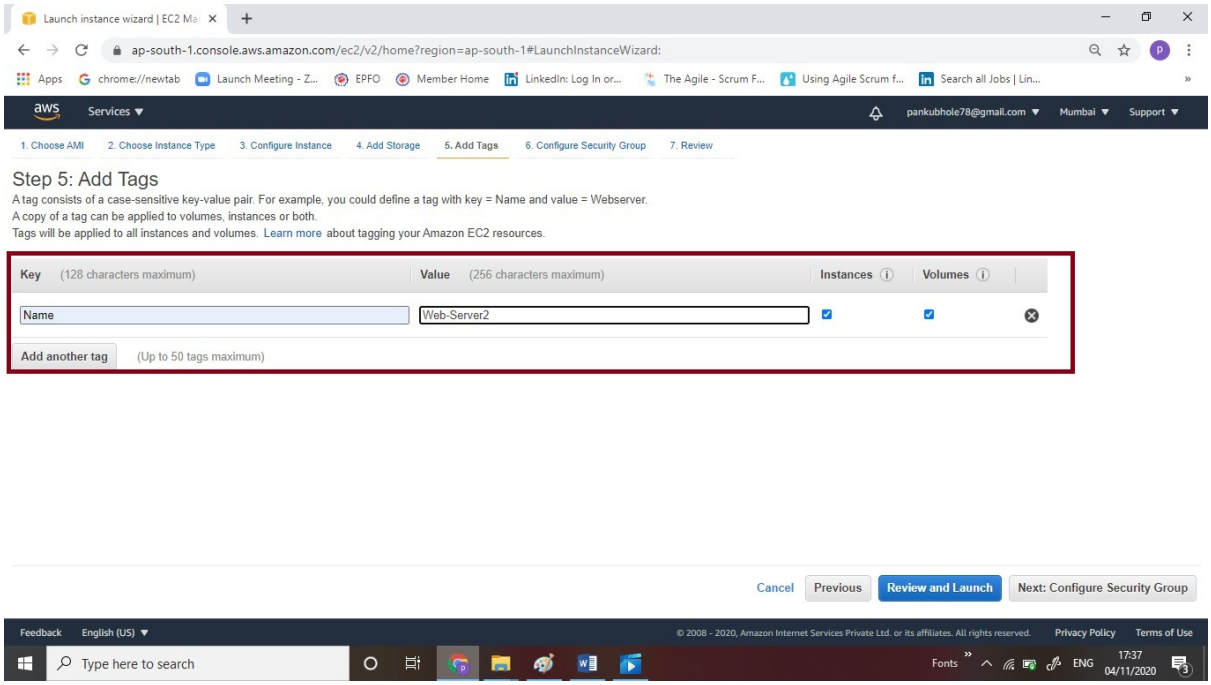
- After that, click on Review and Launch button.

9. **Key Pair:** Create a new key pair named **web-server key,** click download button and the **key will be downloaded to your local system. A**fter that click on .
10. After a few minutes, you will see the new instance named **web-server-1** running along with the **Bastion-server** created in the earlier step.

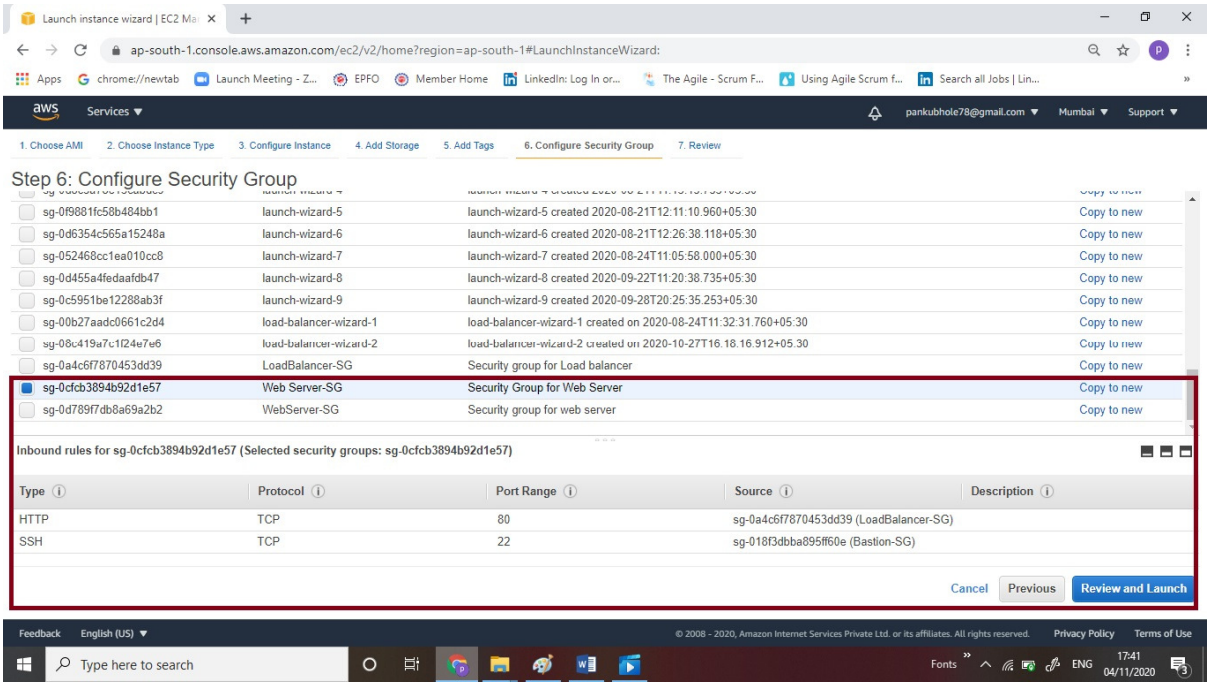


12. Repeat the above steps from Step1 to create **Web-server-2.** You need 2 instances for this lab.

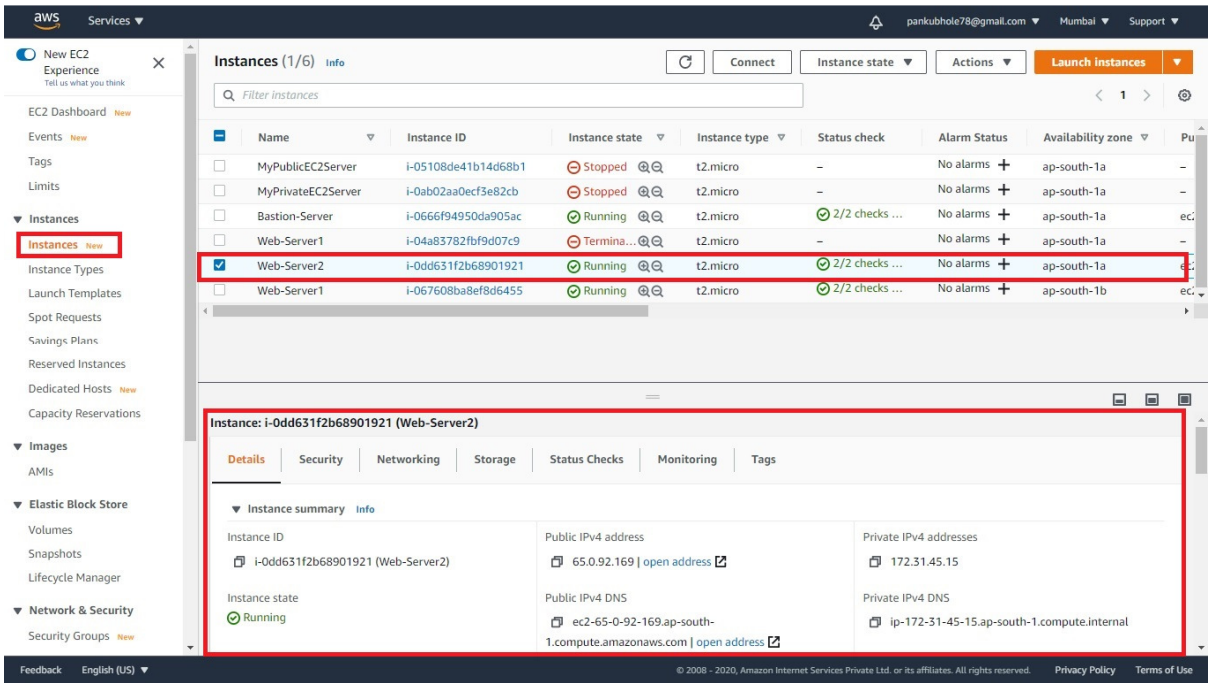
13. On **Add Tags** page, Click on **Add Tag** button, and enter below details.
    Name: **Web-server-2**



14. In, Security Group section, by selecting existing security group **web-server-SG.**



15. Click on **Review and Launch** and then Click on **Launch** and select a key pair, **web-server key,** and **Launch**. **Web server 2**

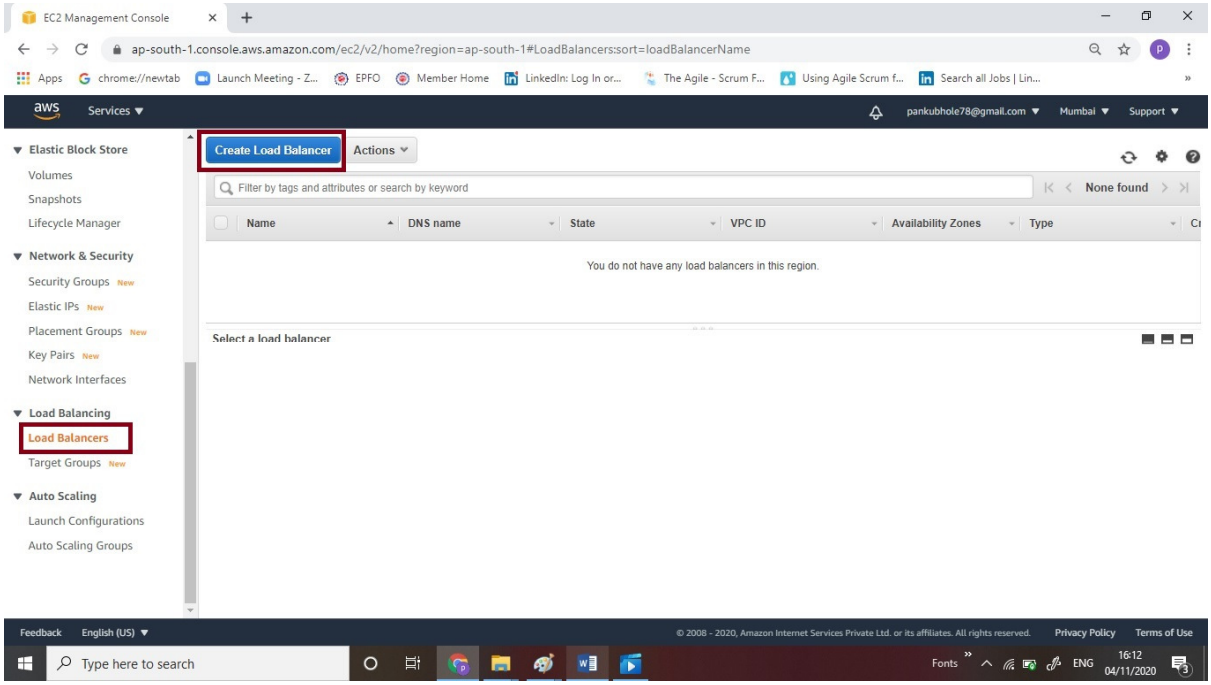


16. Now you will see three servers running namely **Bastion-server**, **Web-server-1, and Web-server-2.**

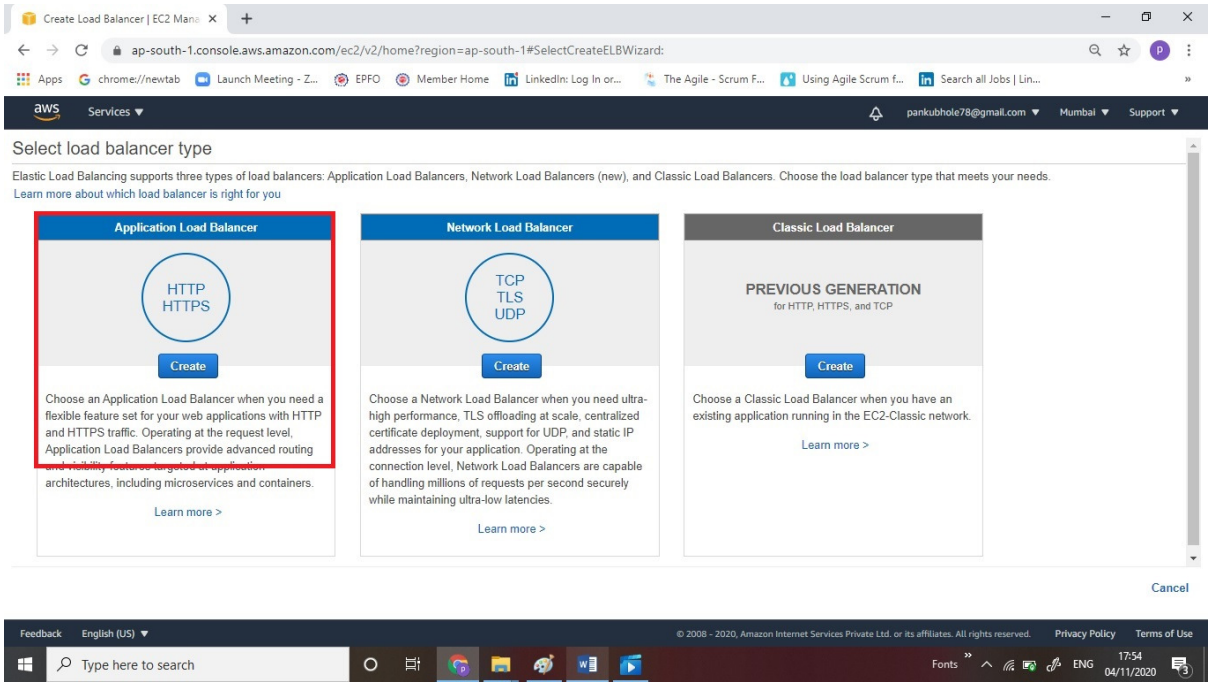
- **Web-server-1 Private IP :** 172.31.5.191
- **Web-server-2 Private IP :** 172.31.45.15

Creating a load balancer:

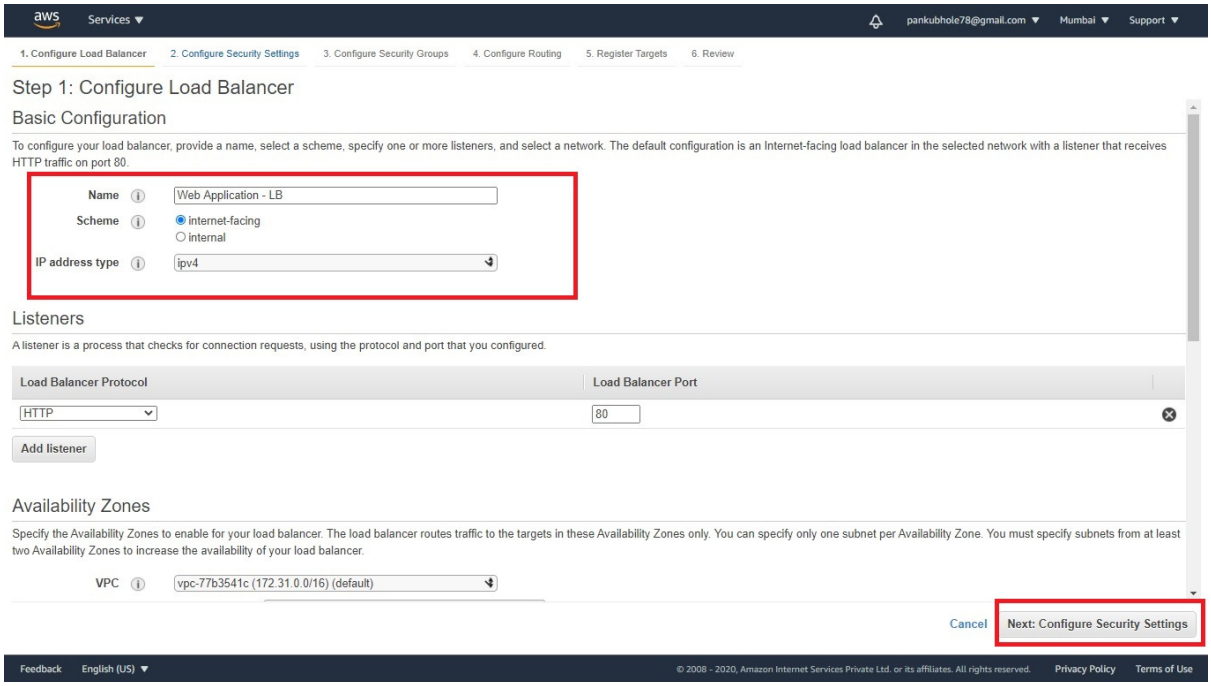
1. In the EC2 console, navigate to in Network & Security-> Load balancer link the left side panel.



2. Click on at the top left to create a new load balancer for our web servers.



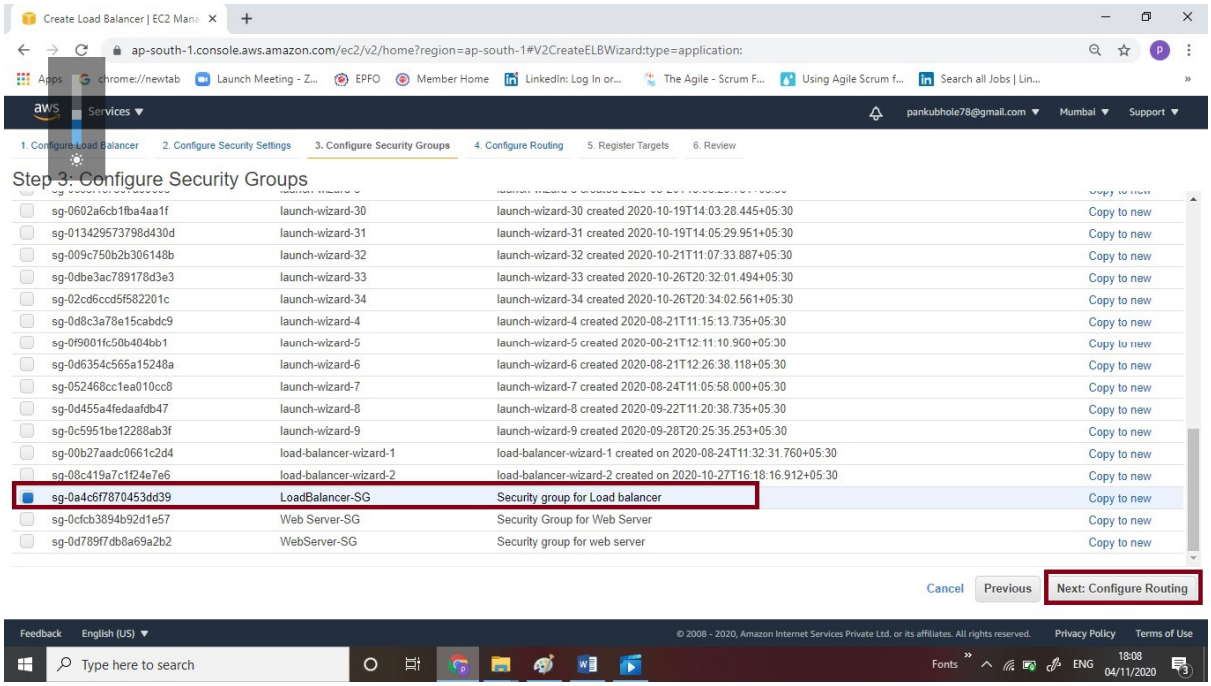
3. On the next screen, choose since we are testing the high availability of the web app.



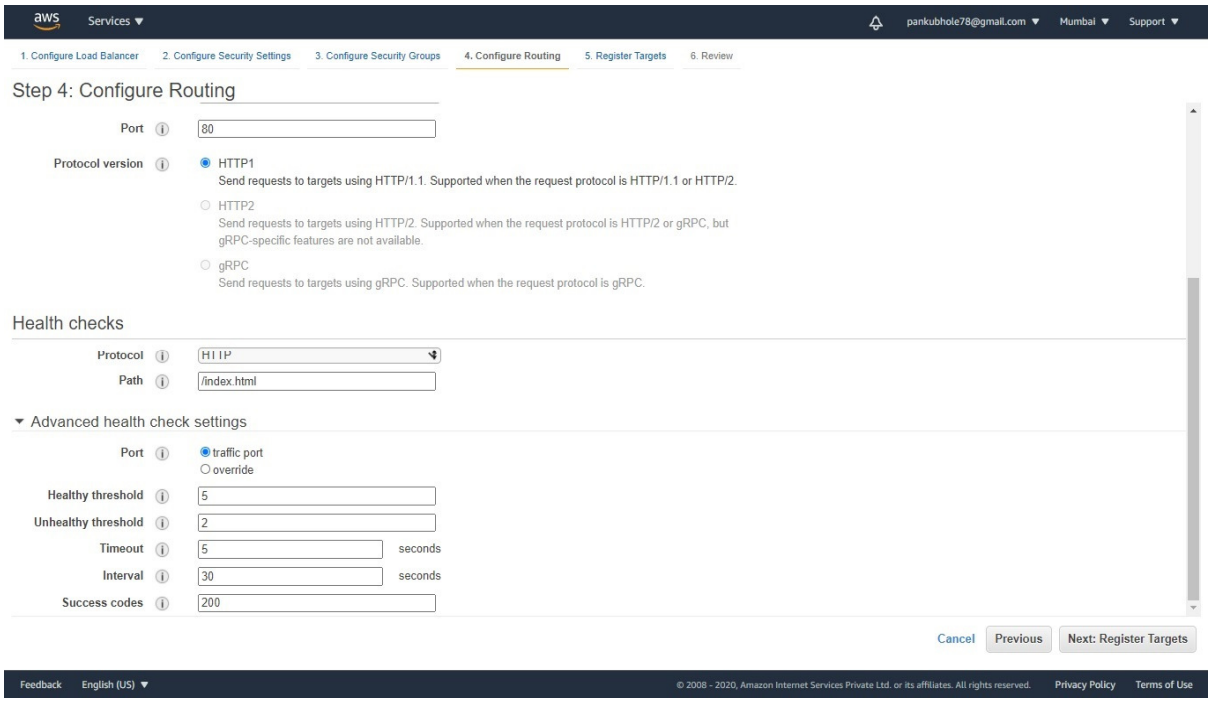
4. In **configure the load balancer** enter the details below:
    - Name: Enter **Web-application-LB**
    - Scheme: Select **Internet-facing**
    - IP address type: Choose **ipv4**
    - Listener: **Default** (HTTP:80)
    - **Availability Zones**
    - VPC: Choose **Default**
    - Availability Zones: Select **All Availability Zones,**
    - Make sure you select the **Public Subnet in the Availability zone us-east-1a.**

Note: we must specify the availability zones in which your load balancer needs to be enabled, making it routing the traffic only to the targets launched in those availability zones. You must include **subnets from a minimum of two Availability zones** to make our Load balancer **Highly Available.**

5. After filling in all the details above, click on **Confirm Security Settings button**.
6. On the next page, ignore the warning and click on.
7. **Configure Security Settings:**
    - Select an **existing** security group and chose the security group **Load Balancer-SG** (we created this one in the step above)



8. **Configure Routing**



- Target Group: Select New target group (default)
    - Name : Enter **web-app-TG**
    - Target Type: Select **Instance**
    - Protocol : Choose **HTTP**
    - Port : Enter **80**

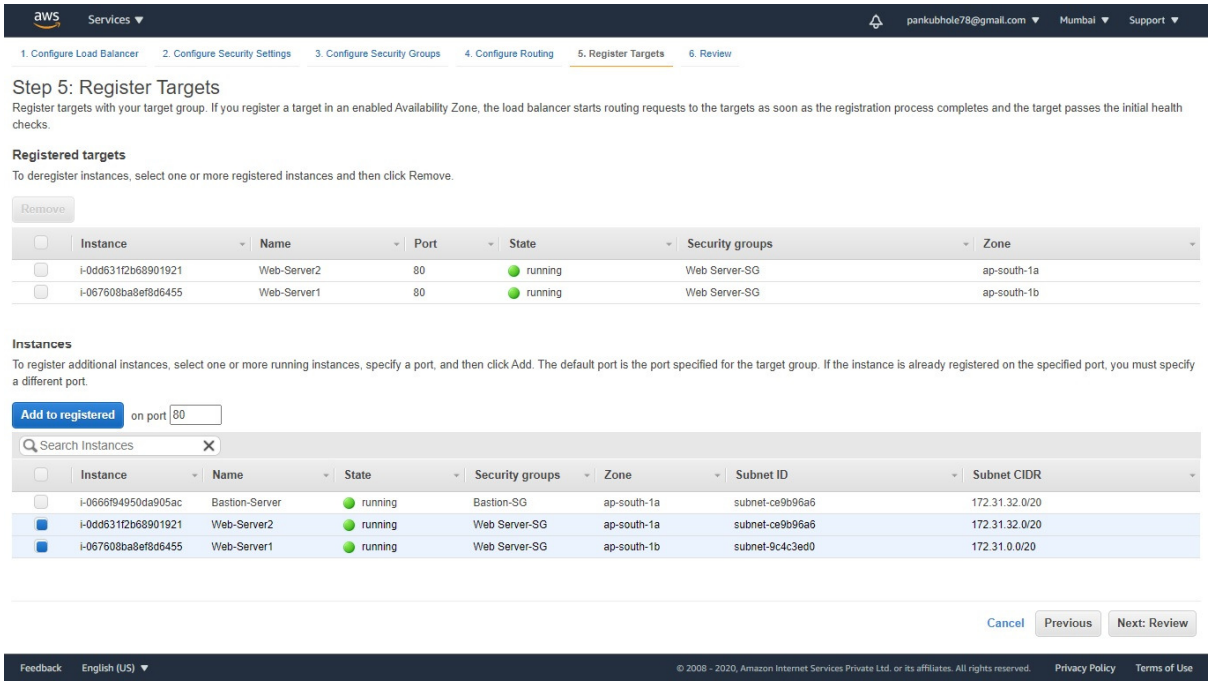
    Note: The target group is used to **route requests** to one or more registered targets.

- Health check:
    - Protocol : **HTTP**
    - Path : **/index.html**

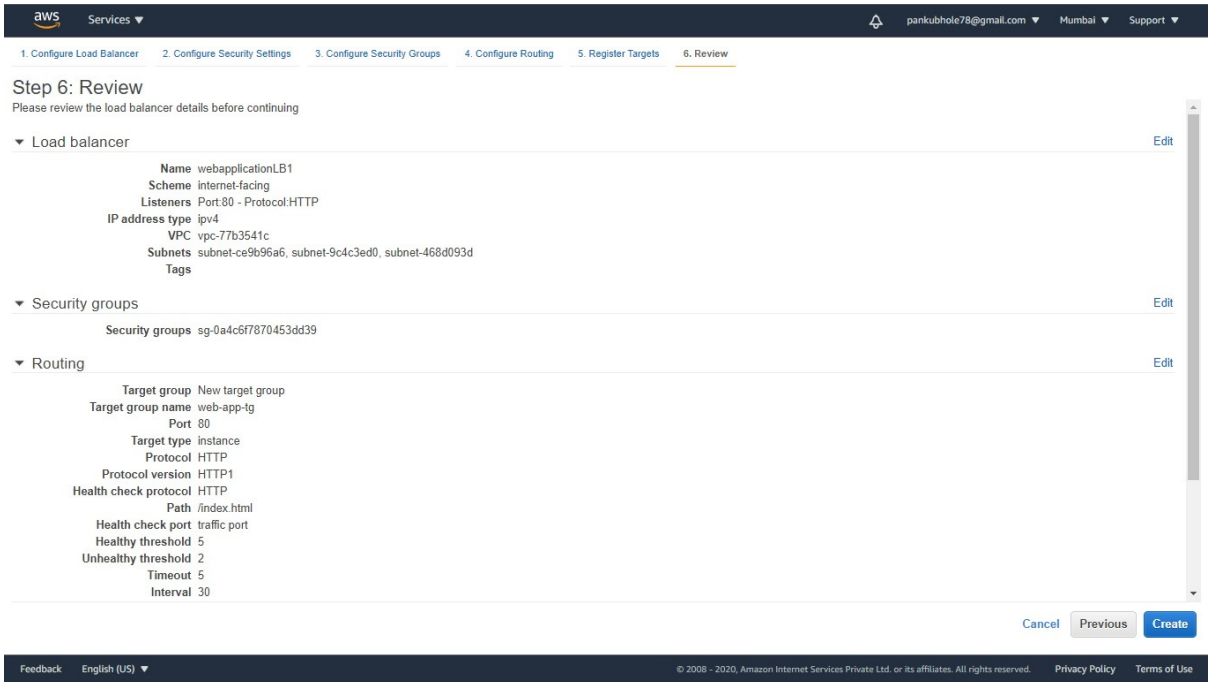
      Note: The load balancer periodically sends pings, attempts connections, or sends requests to test the EC2 instances. These tests are called health checks.

    - In the upcoming steps, we will create an **index.html** in the root directory of the Apache web servers (/var/www/html) to pass this health check.

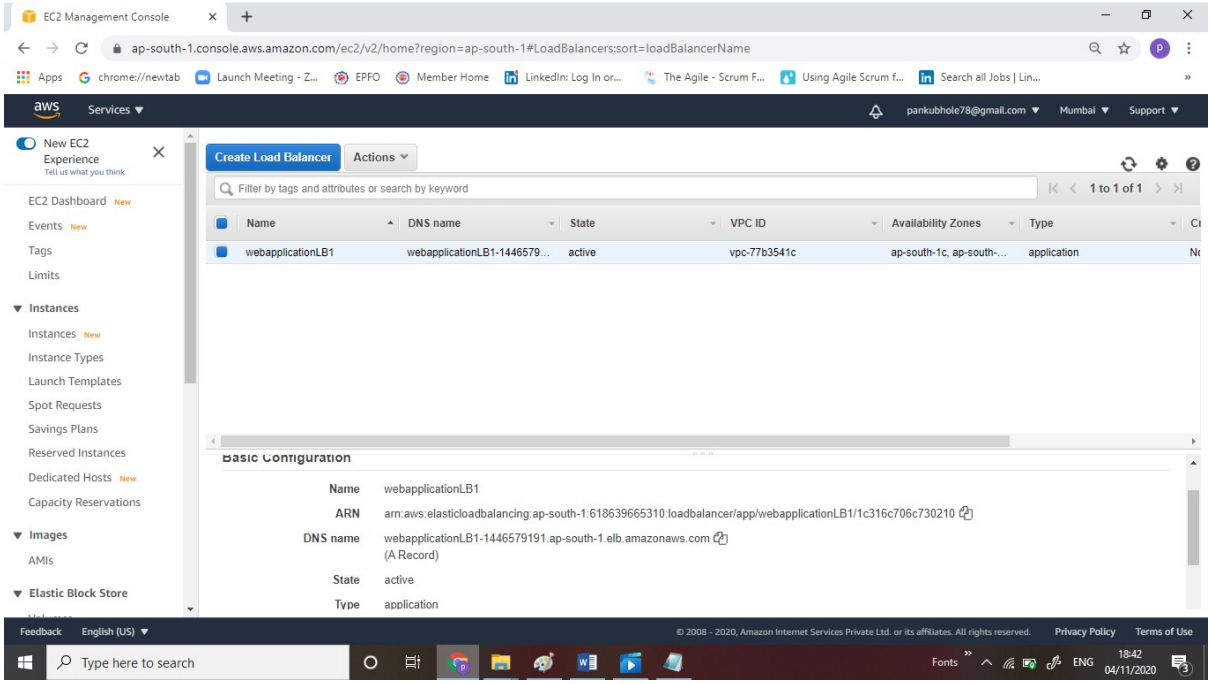
9. **Registering Targets:**
    - Choose the two web instances and then click on and click on.



10. Once you reviewed the settings, click on **Review Button**.

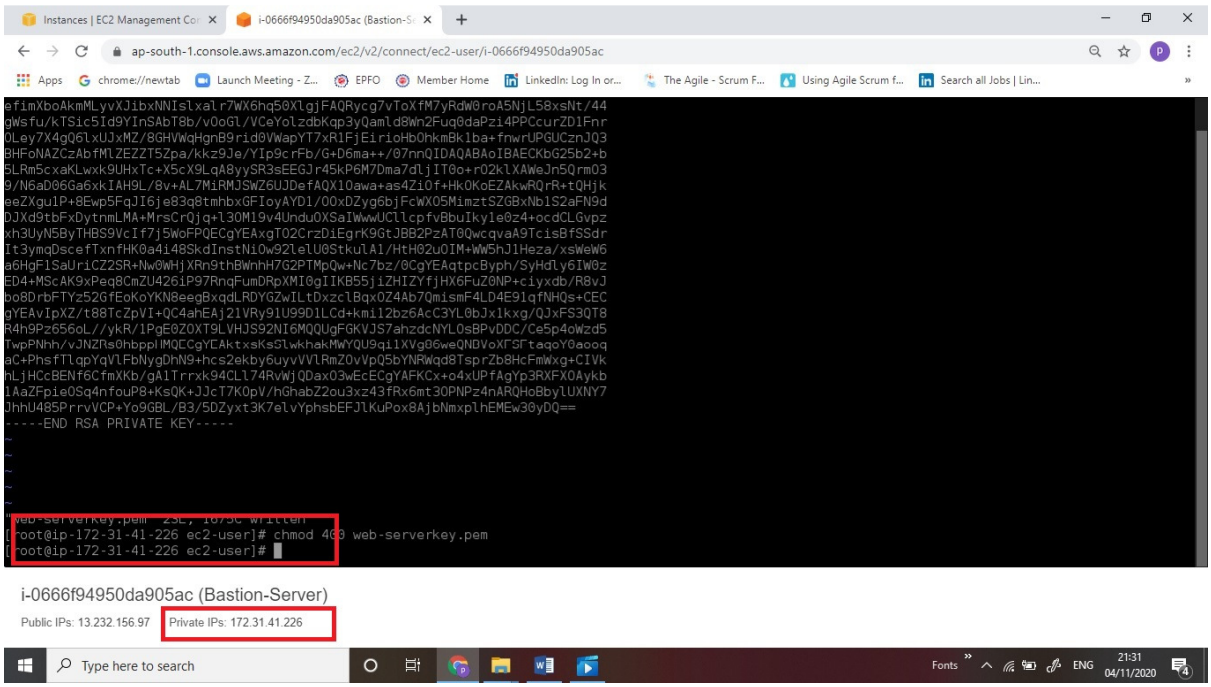


11. You have **successfully created** the **Application Load balancer.** Please wait for 3-4 minutes to make this **ALB into Active** state.



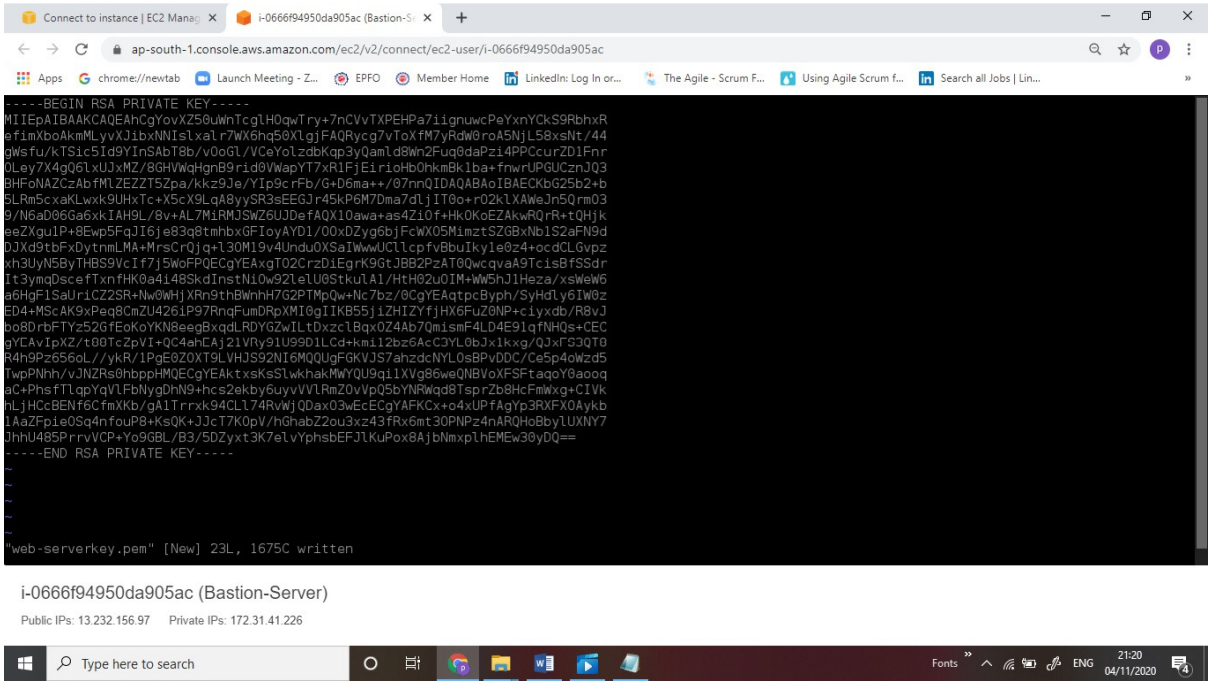
Connecting to web server via Bastion (Optional, if you have pasted the commands in User data):

1. SSH into the Bastion server using the Bastion PEM key: **bastionkey.pem**



2. To SSH into web servers via Bastion server, we need the web server key that we used to launch the previous web servers (**web-serverkey.pem**).
3. Open the **web-serverkey.pem** file on your local system and then **copy the text content**.
4. Navigate to the Bastion server and create a file named **web-serverkey.pem** using below command:

   - **vi web-serverkey.pem**



5. Paste the content and save it by pressing **shift+colon followed by :wq!** and then enter to save your private key.
6. Make sure you have changed the **permission of the key file to 400**. You can change the permission using below command:

- **chmod 400 web-serverkey.pem**

7. Now **you can log into the web servers** using the private key copied to the bastion server with the help of below commands.

- **Note:** You **don't have a public IPs** for the web servers since we them in a private **subnet.**
- Syntax **: ssh -i web-serverkey.pem ec2-user@**<Web-server-2 private IP>
  - Example: **ssh -i web-serverkey.pem ec2-user@ 172.31.45.15**

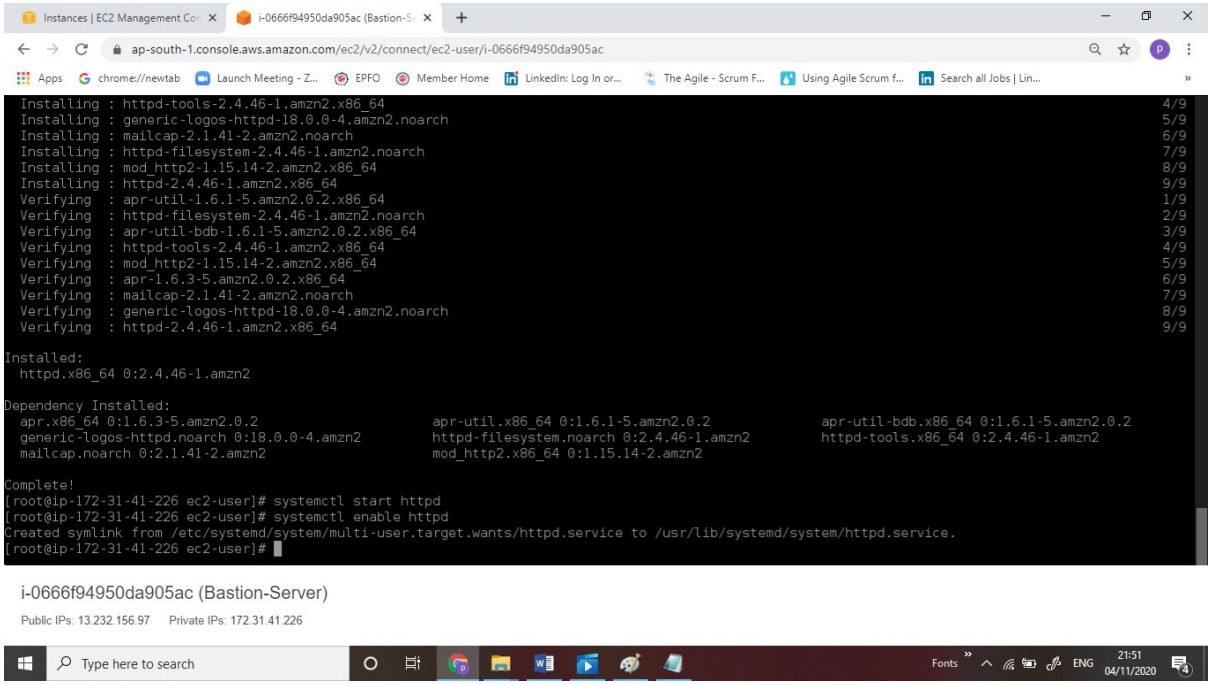
8. Now **install the apache service** using the below commands and **create a test index.html file,** which will be **used for a health check.**

- **Installing Apache:**
  - **sudo su**
  - **yum update -y**

- yum install httpd -y
- systemctl start httpd
- systemctl enable httpd

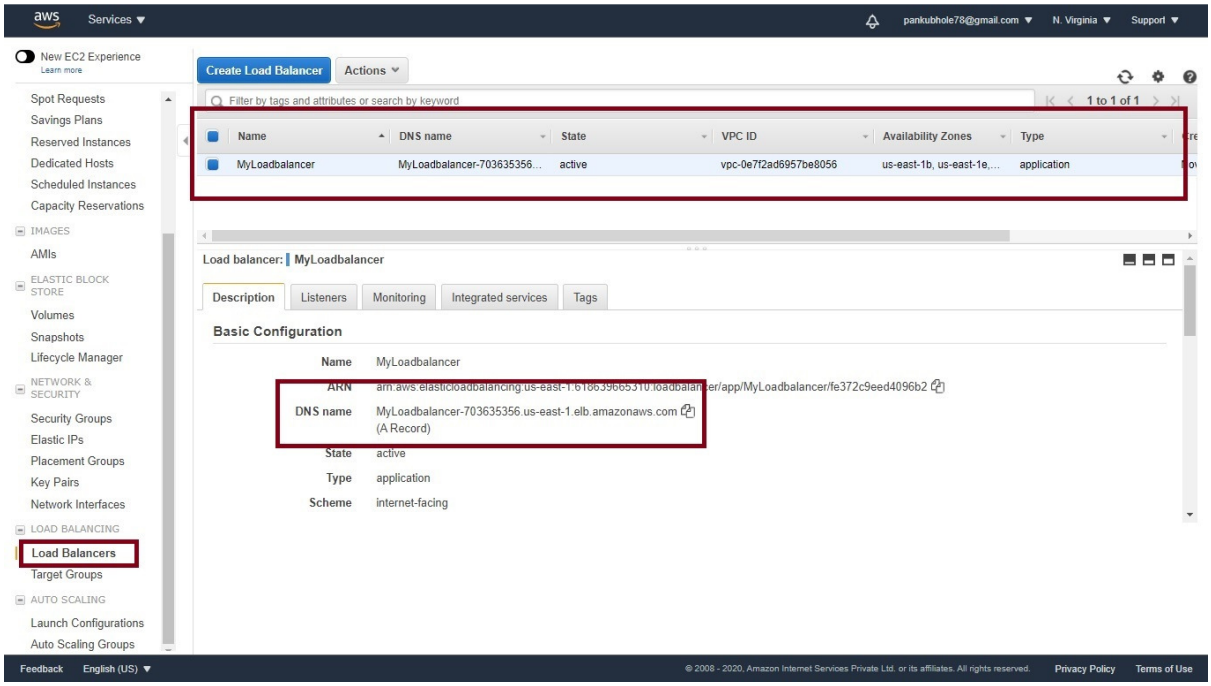
- **Creating the example homepage :**
  - **echo " REQUEST HANDLING BY SERVER 1" > index.html**
- **?Exit from webserver to Bastion server**
  - **?**To come out of 2nd instance, type **exit** command for coming out of root user, and **exit** command again for coming out of the instance.

9. Repeat steps 7 & 8 for web server 2 **with its respective private IP** , making sure to change the content of index.html to "**REQUEST HANDLING BY SERVER 2**"
10. To come out of 1st instance, type **exit** command for coming out of root user, and **exit** command again for coming out of the instance.

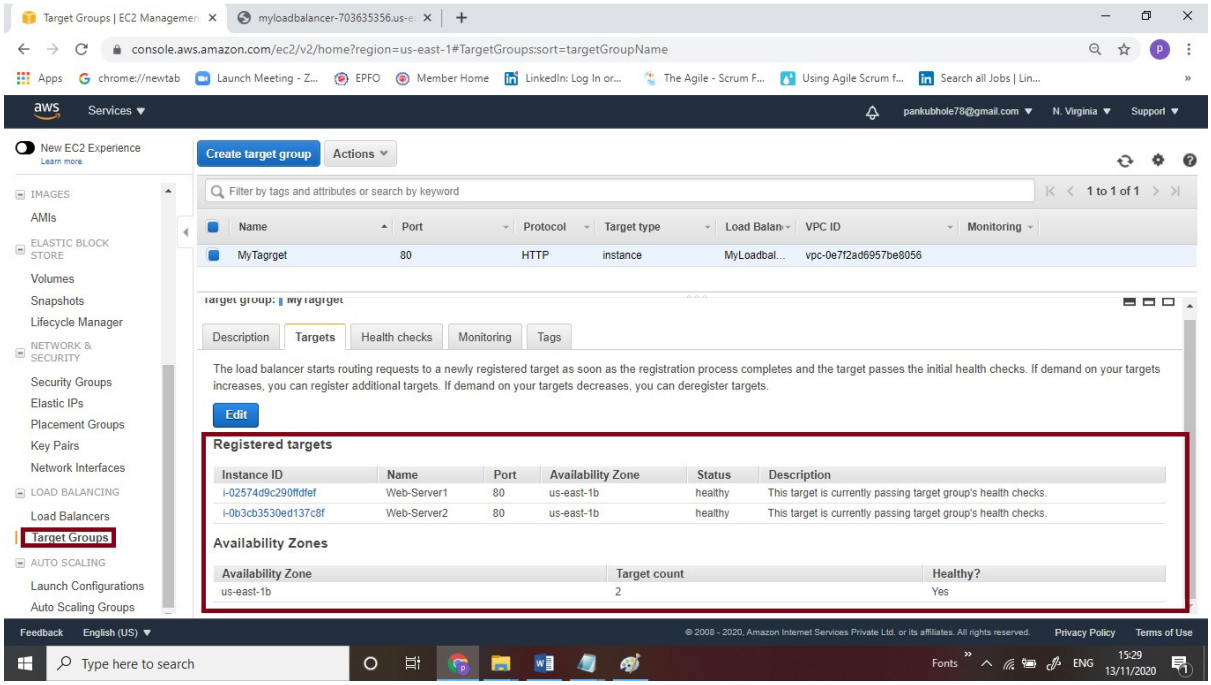


Checking the health of the load balancer

1. Navigate to the Load balancer and created the one Myloadbalancer for the system health checkup.

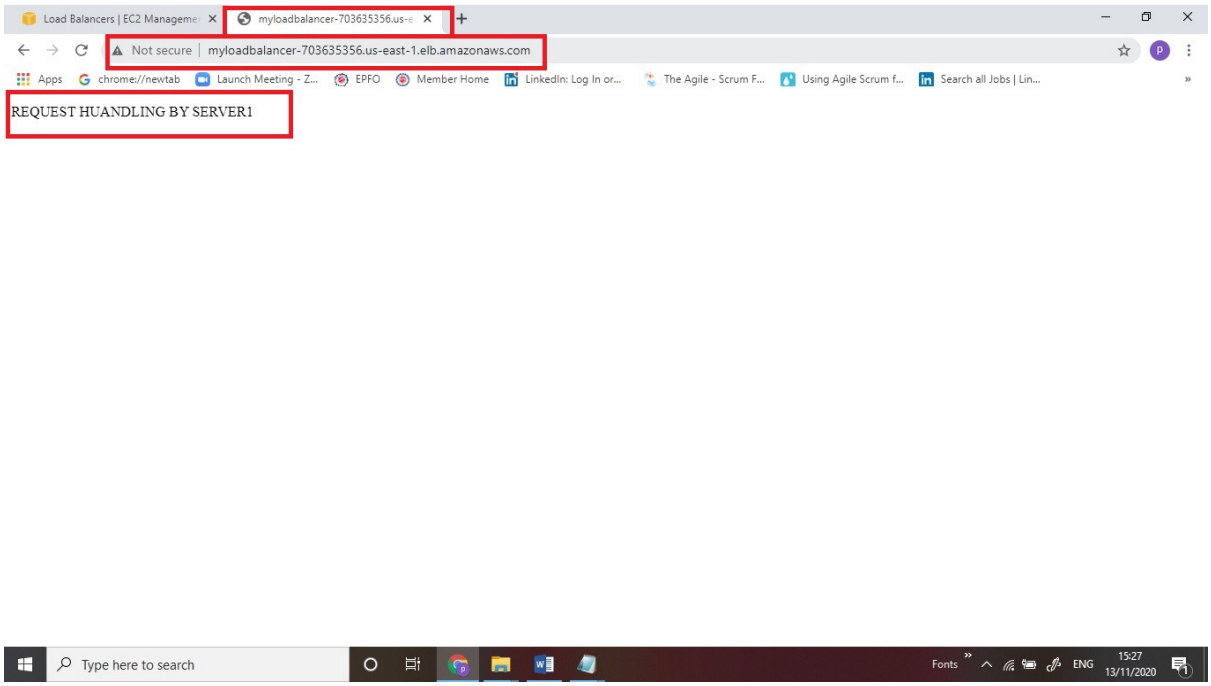


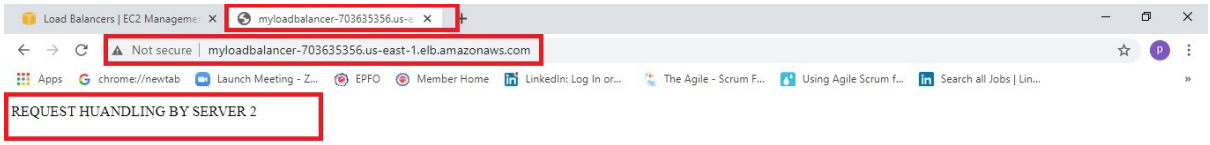
2. Select the target group you created Mytagrget and then click on **Registered Target** and to see the **Status** of the attached targets.

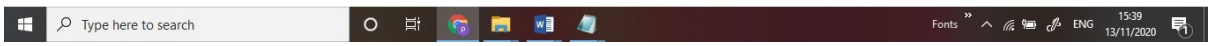


3. It should show **Healthy** for the Load Balancer to work properly. You may need to wait for 2-5 minutes before the load balancer's status updates to "Healthy"

4. Now navigate to **load balancer** and select **DNS Link** that you created earlier.
5. Click on **DNS Link, copy the link** and paste it into the browser.
    - DNS URL: MyLoadbalancer-703635356.us-east-1.elb.amazonaws.com







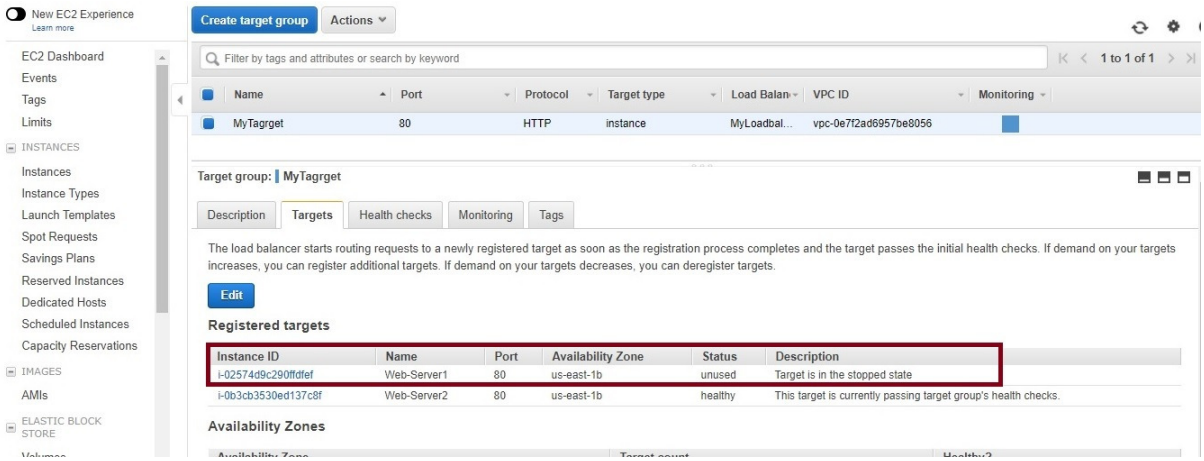
5. Refresh the browser a couple of times to see the requests being served from both servers. Seeing output similar to **REQUEST HANDLING BY SERVER 1 & REQUEST HANDLING BY SERVER 2** implies that load is shared between the two web servers via Application Load Balancer.
6. Now we have successfully created a **bastion server, two web servers and an Application Load balancer,** registered the targets to the load balancer and tested the working of Load Balancer.

Test case for High Availability

1. To check for high availability, we will make one of the instances unhealthy and test whether we get response from the other server.
2. If your instance is shown as Unhealthy then it's status would be one of the following:

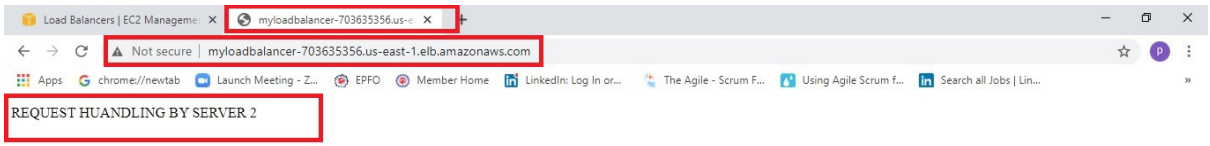
- stopping
- stopped
- terminating
- Terminated

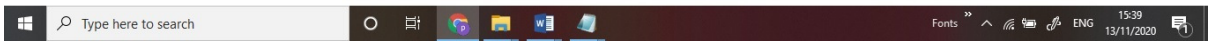
3. Navigate to the EC2 dashboard and select **Web-server-1.** Click on Action-> Instance state -> select and then click on **stop.**



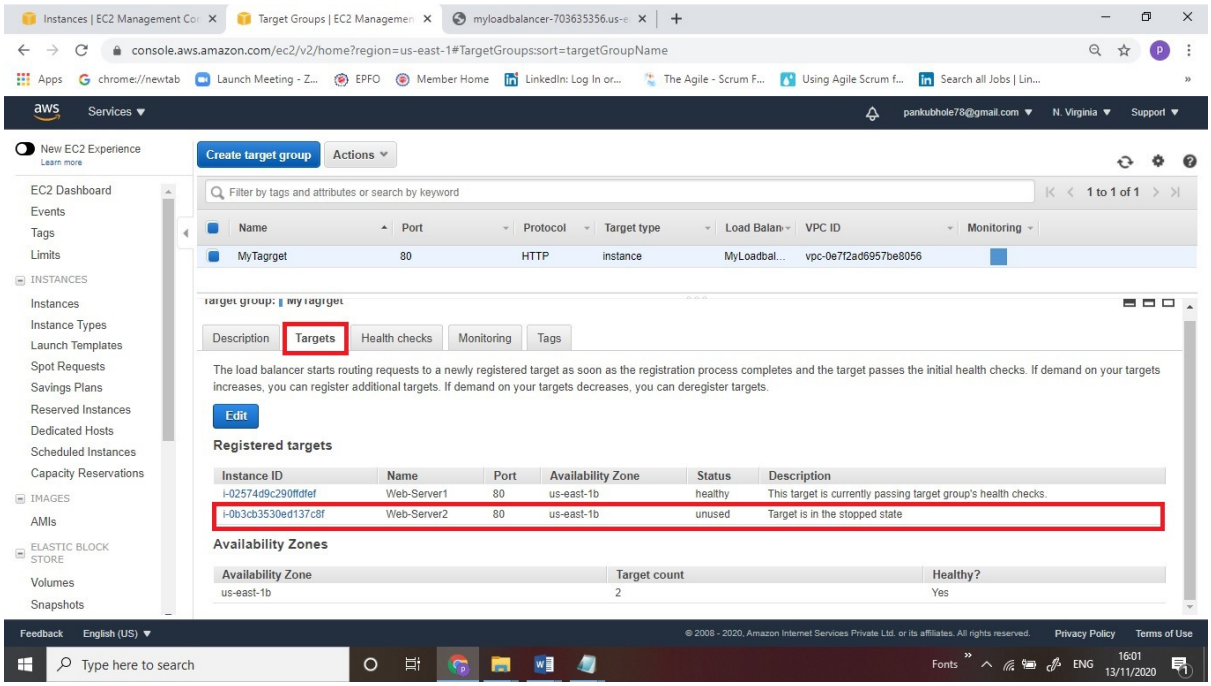
4. Navigate to **Target group** and click on **targets.** Here you will find the status of Web-server-1 (which should be unhealthy because it is unused).

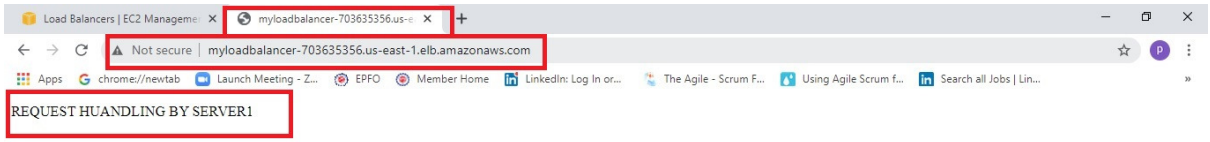
5. Navigate to **Load balancers-->Description-->DNS name.** Copy the DNS name and paste it into your browser. You should see the response "REQUEST HANDLING BY SERVER 2" FROM WEB-SERVER-2**.**

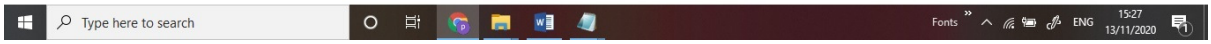




6. If you refresh a few times, you will continue to see the response only from Web-server-2
7. Repeat step 3 by stopping *Web-server-2* and starting Web-server-1 back up. This time you should see the response "REQUEST HANDLING BY SERVER" from Web-server-1.







Completion and Conclusion

1. We have launched a Bastion server and two web-servers. We were able to SSH into the servers via Bastion Server successfully.
2. We launched an Application Load Balancer and associated our web servers with the load balancer.
3. We tested the load sharing between web servers.
4. We successfully tested the high availability of the web application by making one of the web servers unhealthy.