CCT College Dublin

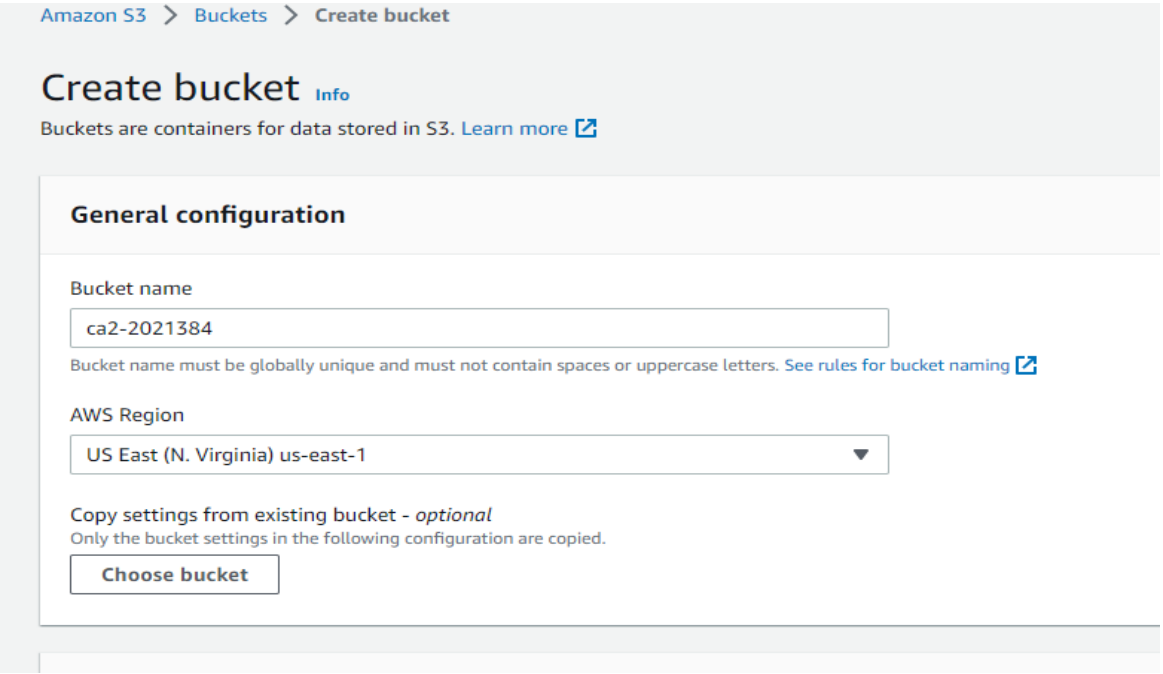
Assessment Cover Page

Module Title:	Cloud Computing Fundamentals
Assessment Title:	Autoscaling and NLB in a cloud-based environment
Lecturer Name:	Michael Weiss
Student Full Name:	Muhammad Sajjad Haider
Student Number:	2021384
Assessment Due Date:	Sunday 11 December 23:59
Date of Submission:	Sunday 11 December

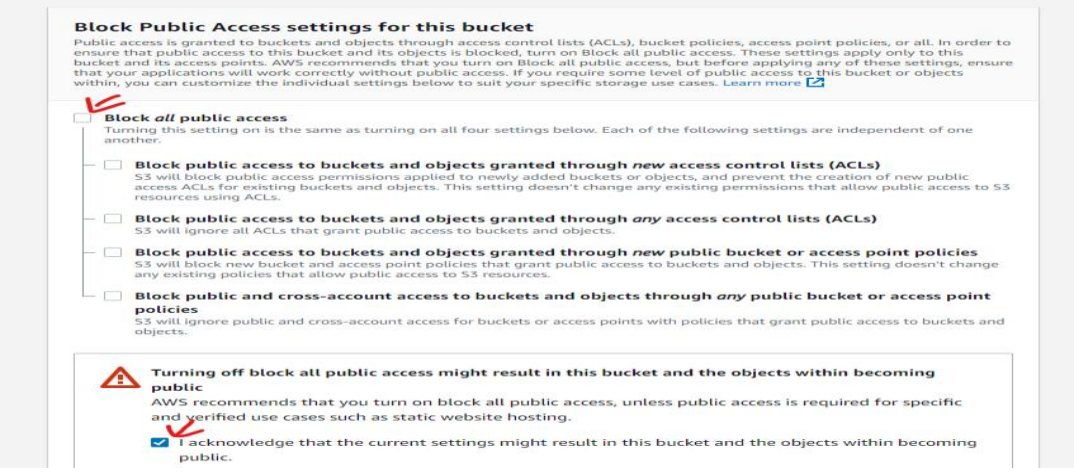
Table of Contents

Part 1: AWS Bucket

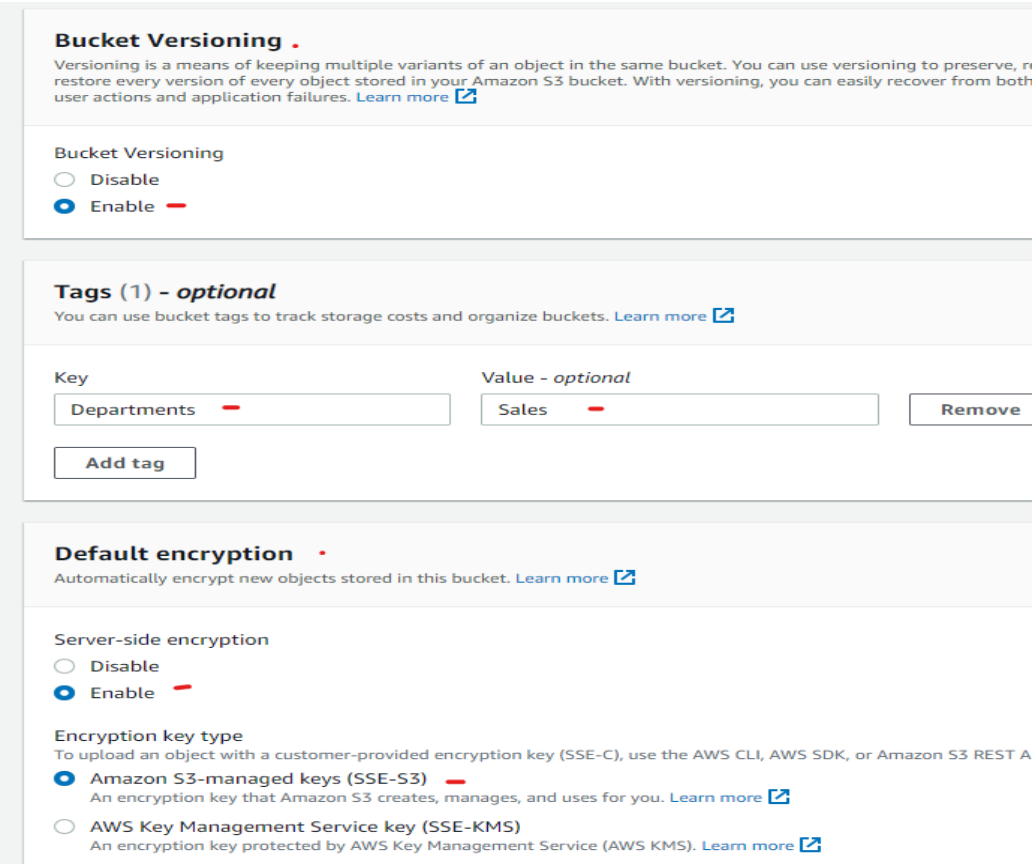
- Go to **Services**, Select **Storage** and navigate **S3**.
- In Amazon S3 select **Buckets**.
- Click on **Create bucket**.
- I gave **bucket name** as **ca2-2021384**.
- Make sure **AWS Region** in **US East (N. Virginia) us-east-1** as shown below:



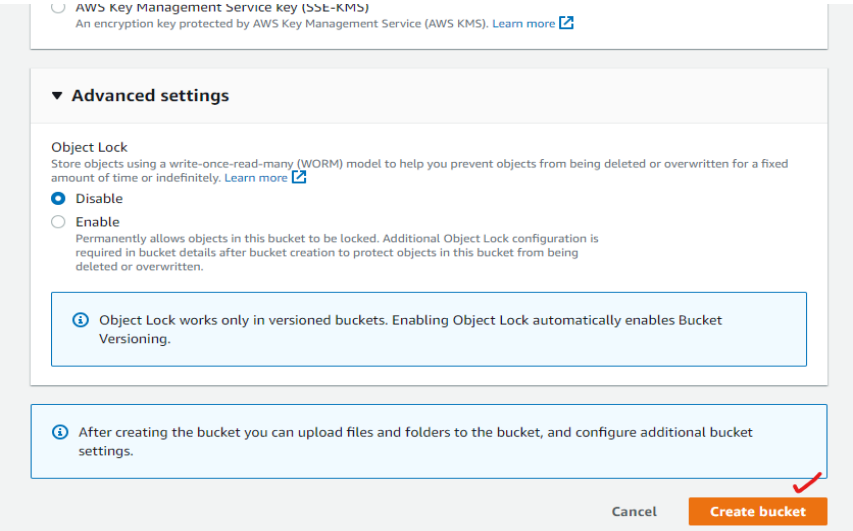
- **Untick** the option **Block all public access.**
- Then, **tick** the **acknowledgment** below:



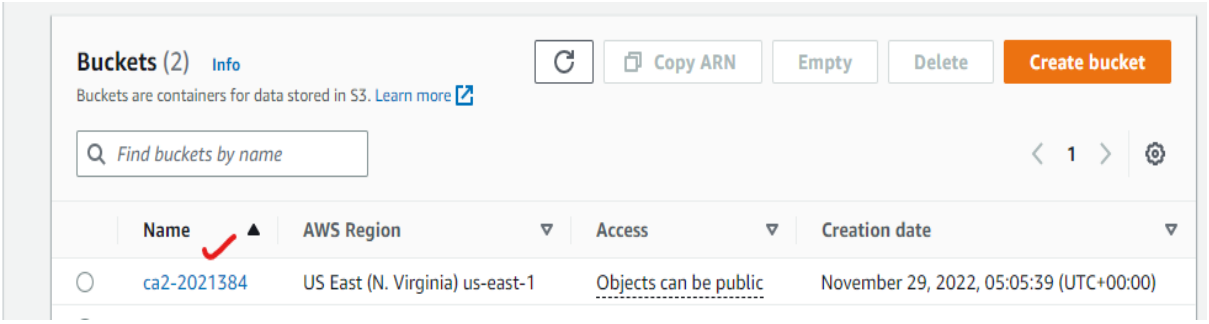
- **Bucket Versioning** should be **enabled**.
- **Tags** are **optional** but I gave **Key** as **Departments** and **Value** as **Sales**.
- **Default Encryption** should also be **enabled**.
- **Encryption key type** should set to first option **SSE-S3**.



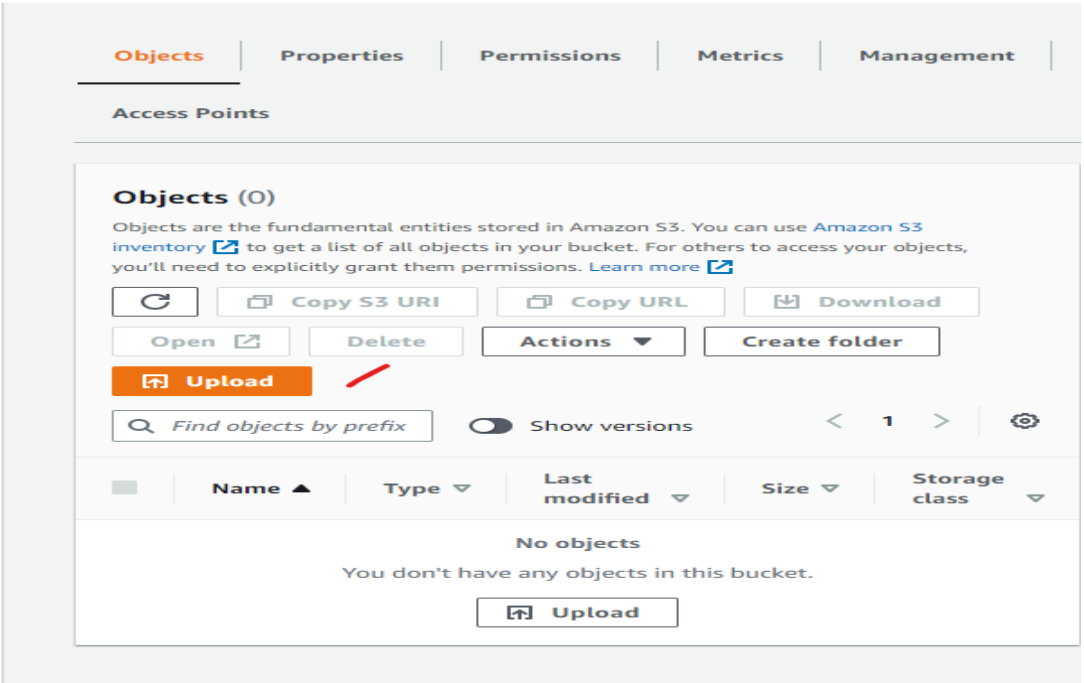
- **Advanced settings** remain **same** and **click** on **Create bucket**.



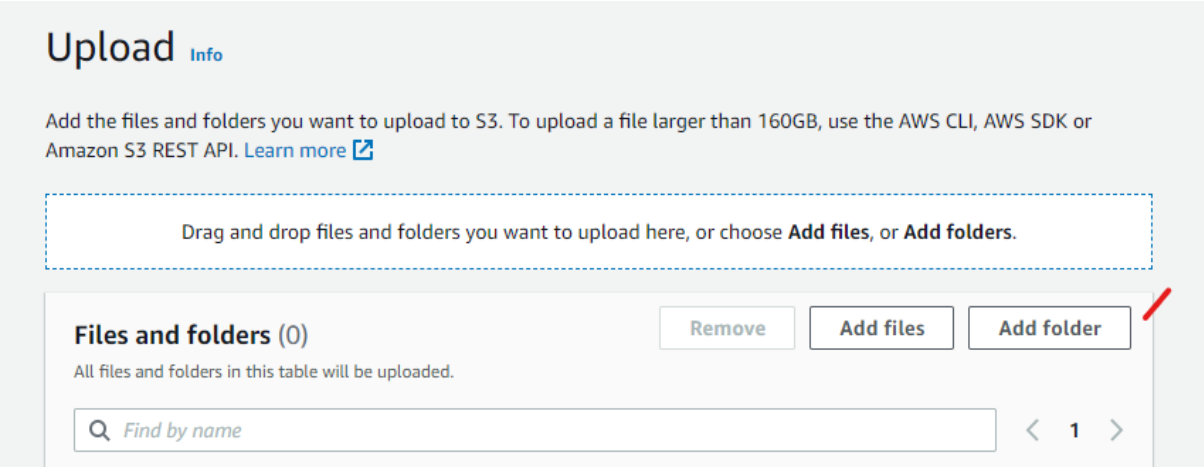
- **Ca2-2021384** bucket has successfully **created** as shown in shot below:



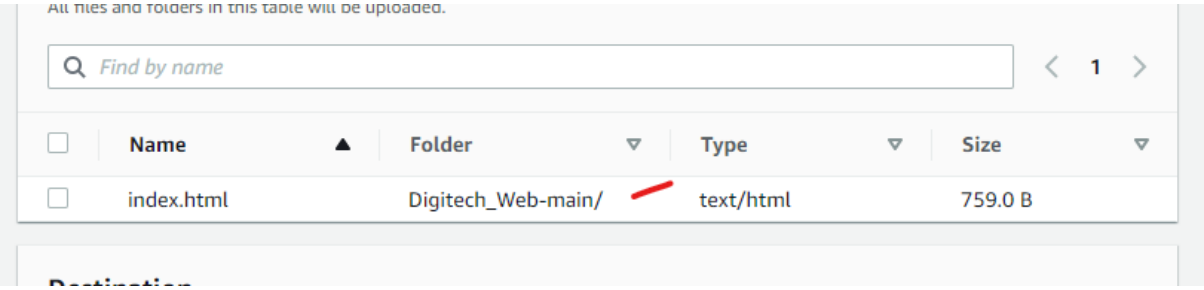
- **Open** the bucket **ca2-2021384**.
- Click on **Upload**.



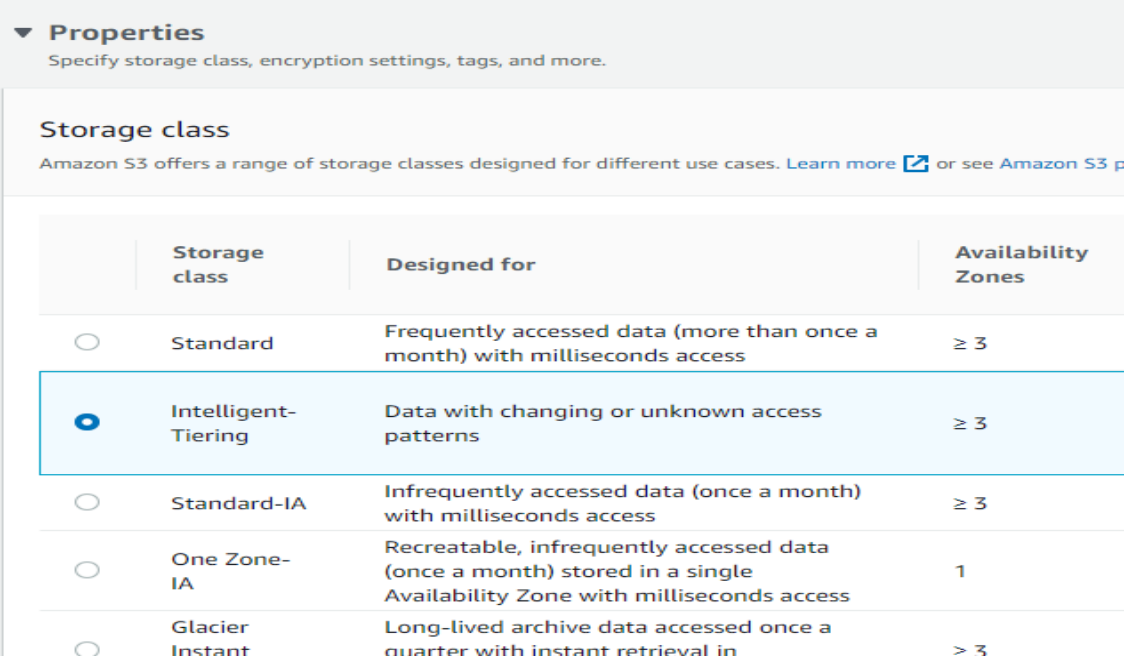
- Inside Upload, click on **Add folder**.



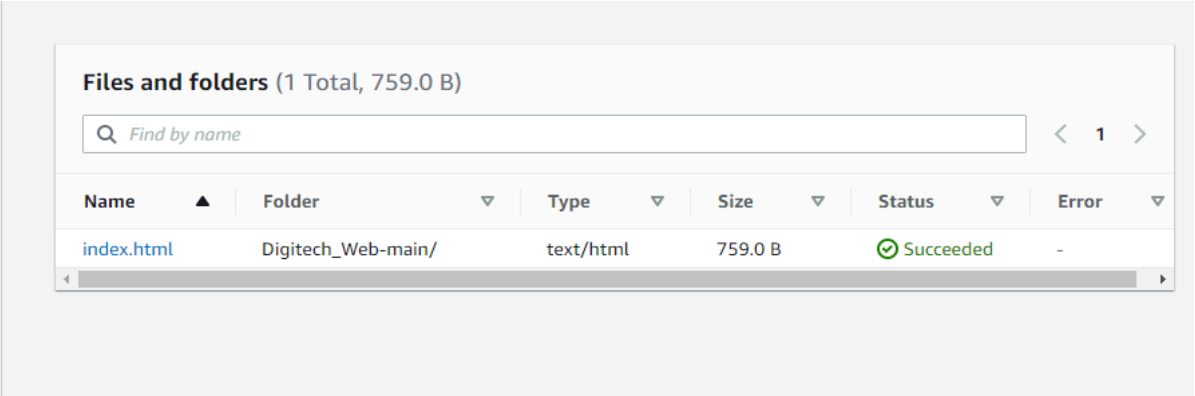
- Find the **Digitech html page** from the device and **upload** it.



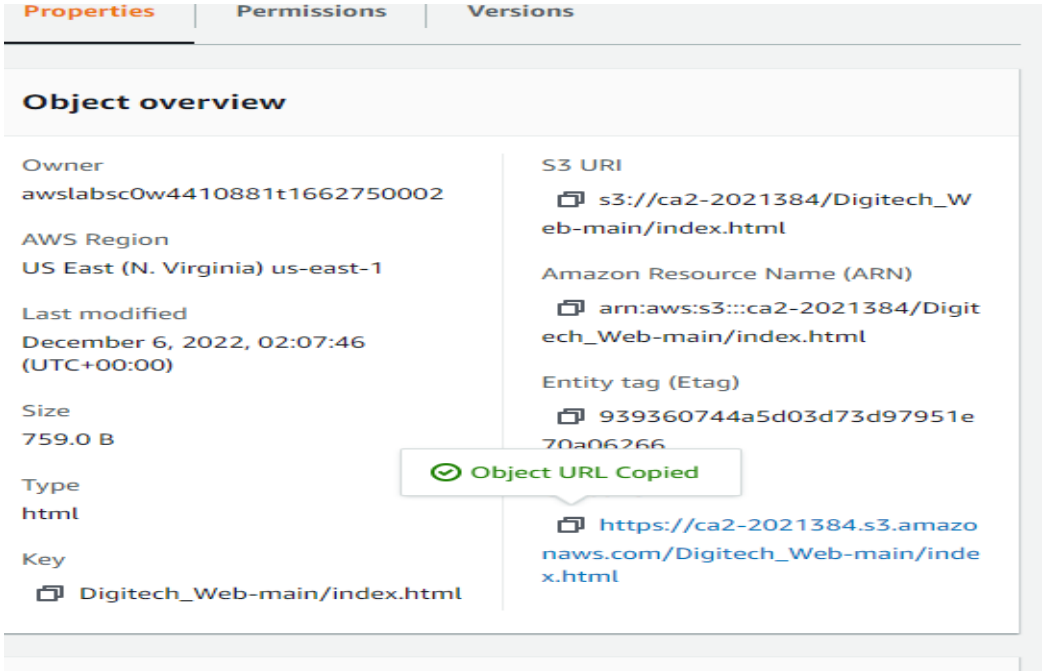
- Scroll down, before clicking on **upload**, select **Intelligent-Tiering** form **properties**.



- It can be seen **Digitech web page** has successfully **uploaded**.

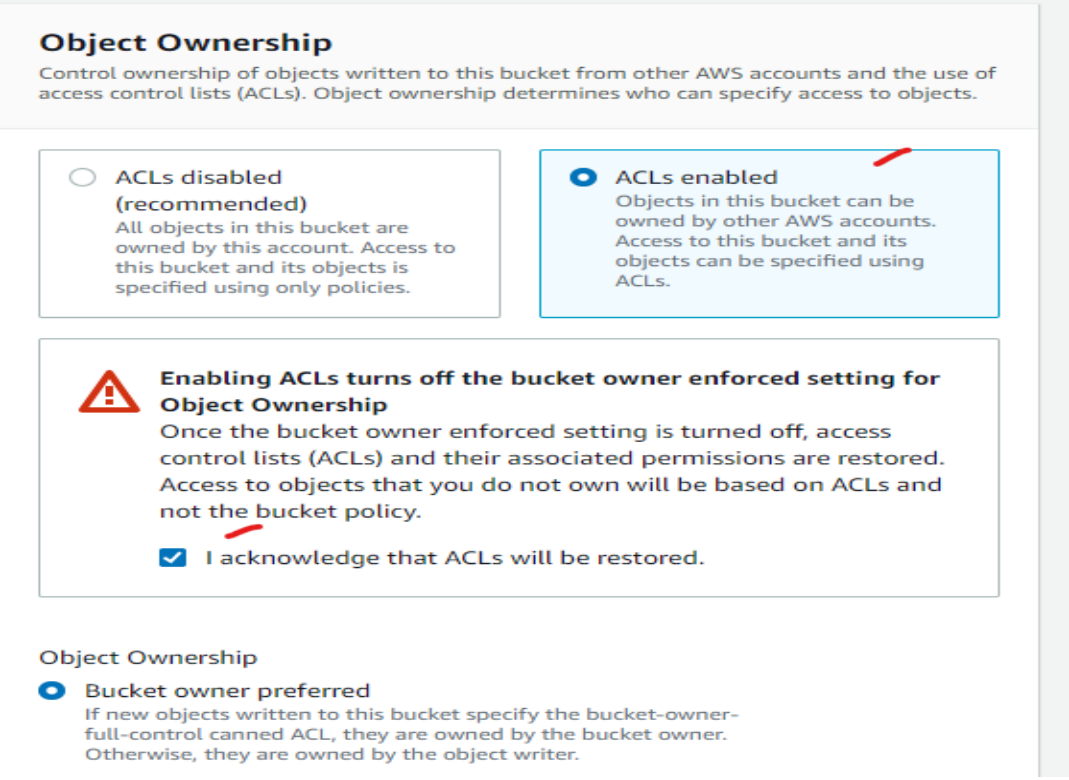


- **Open** the **Digitech html**, and **copy** its **URL**.
- **Search** it on **google**.
- It **gave error**.



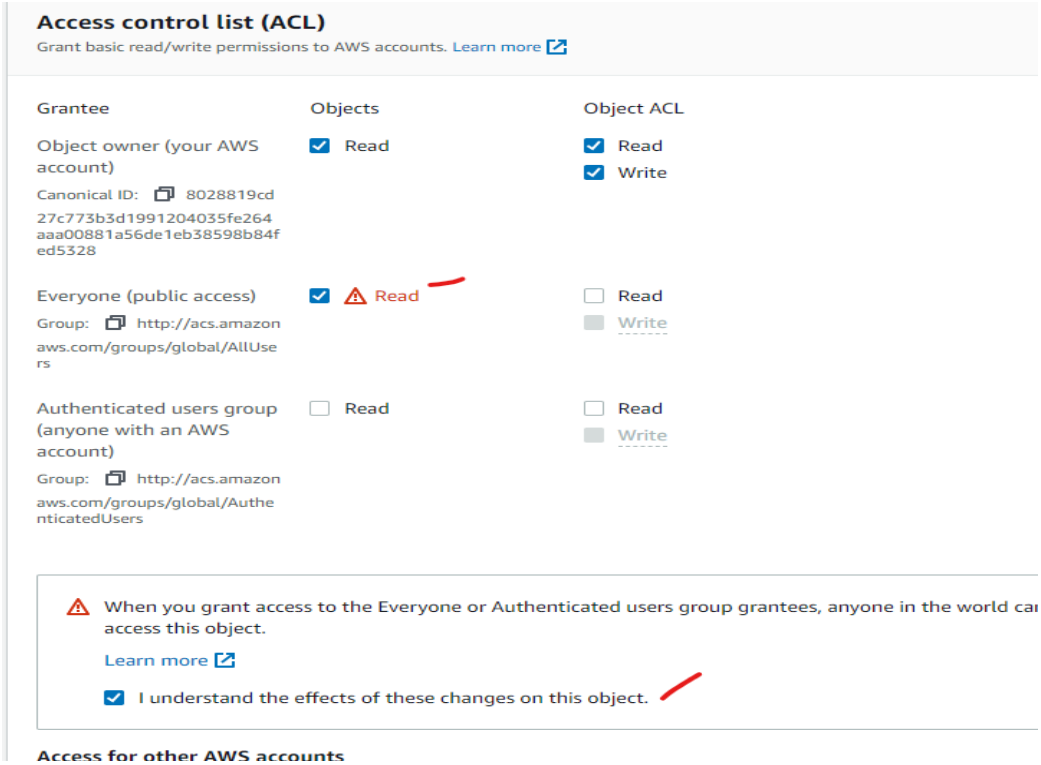
- Go back to **Bucket**, **opened ca2-2021384**.
- Go to **permissions**.
- **Click** on **Edit Block public access**.
- **Untick** the **Block all public access**.

- **Stick** in **permissions**.
- **Click** on **Edit Object Ownership**.
- **Enable** the access controller **ACLs.**
- **Make sure to tick** the **acknowledgement below**.

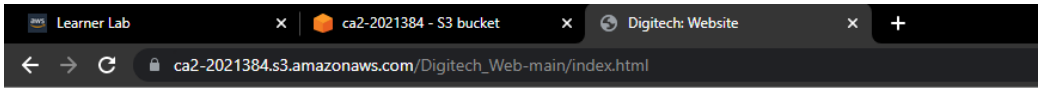


- Go back to **inside** the **Index.html file**.

- **Edit** the **access control**.
- **Tick** the **Read** option of **Object**, in **public access**.
- Tick the **acknowledgement** below of changes**.**
- Click on **Save.**



- **Paste** the **URL** of **Digitech html page** on **google** and **search** it.
- It can be seen that **google** is **showing successfully** the **Digitech html page.**



Digitech Company

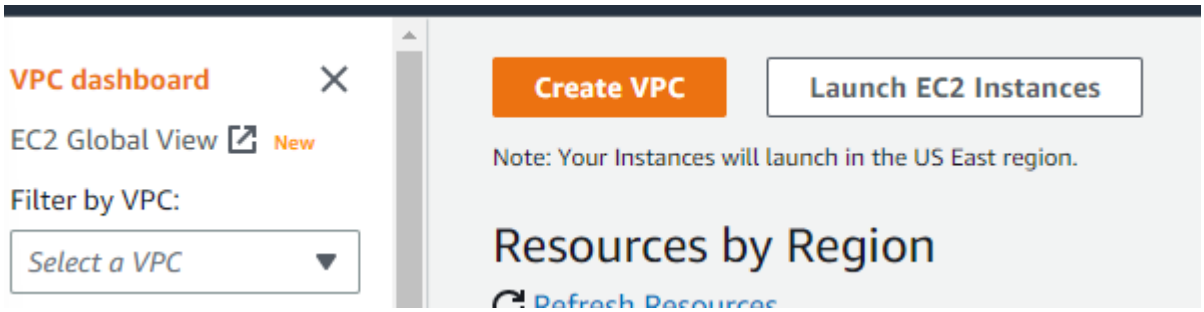
Digitech
Digitech

Webmaster : Muhammad Sajjad Haider

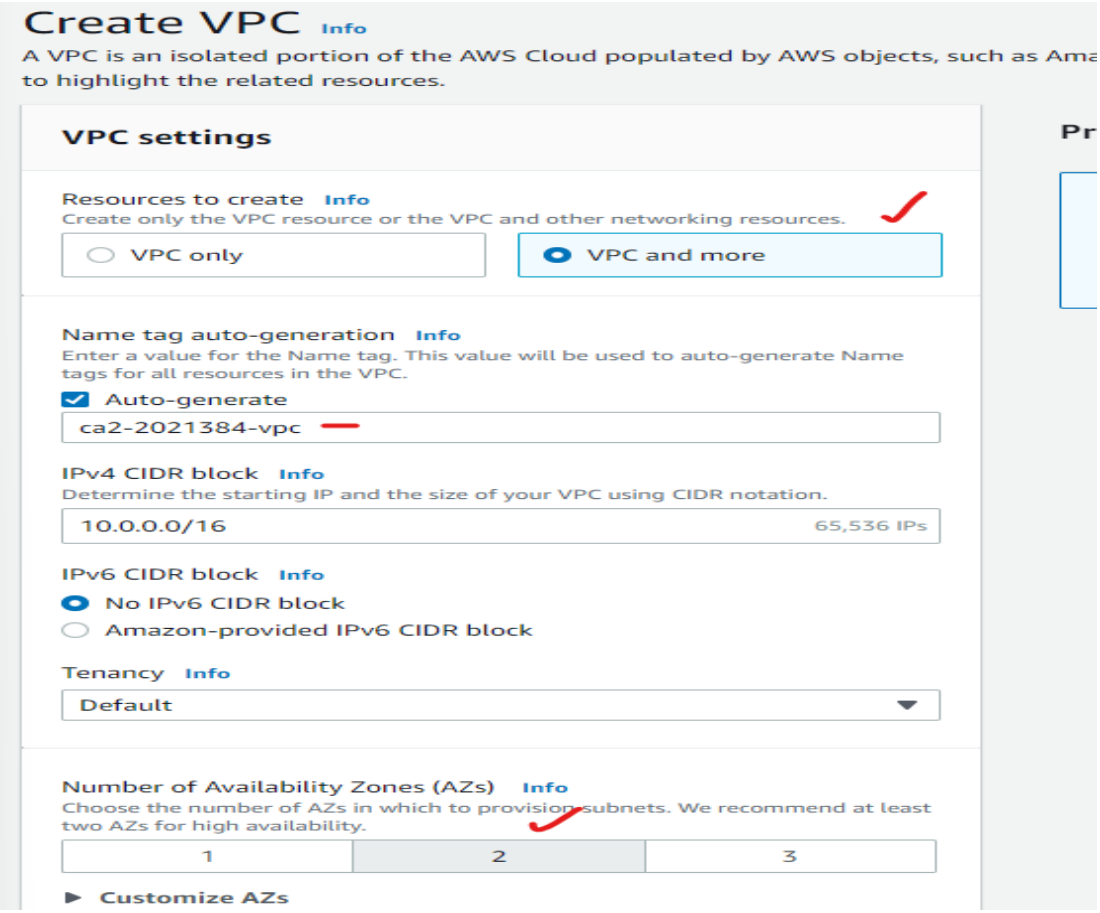
Student Number : 2021384

Part 2: VPC in one region with two availability zones

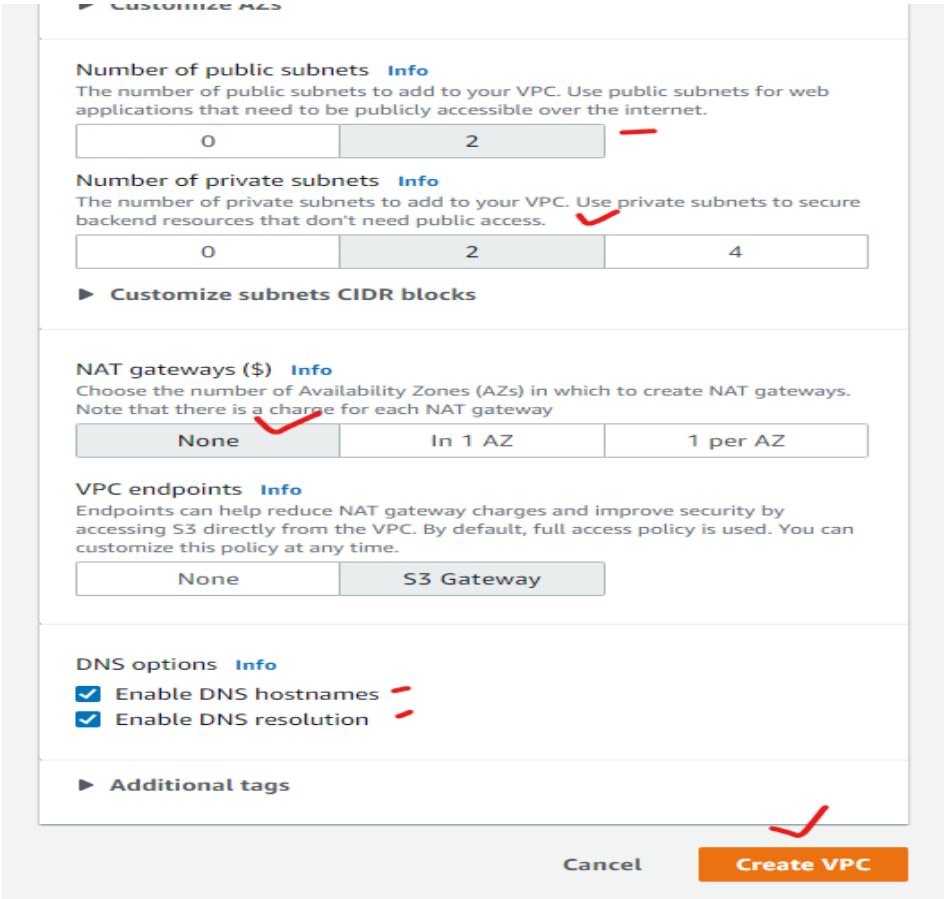
- Go to **Services**, Select **Networking & Content Delivery** and open **VPC**.
- **Click** on **create VPC**.



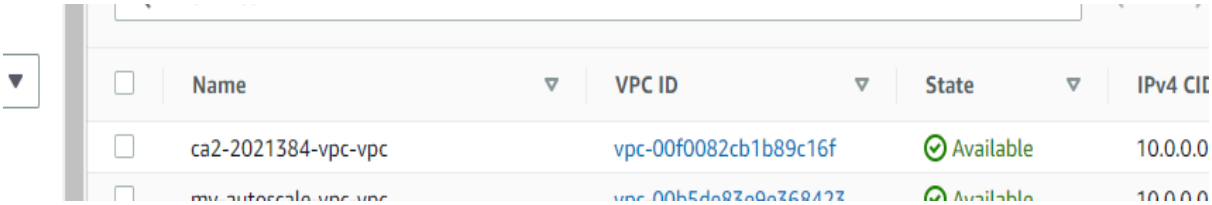
- First, I selected **VPC and more**.
- I gave **name** tag as **ca2-2021384-vpc**.
- Select **2** for number of **availability zones**.
- **Tenancy** will be remained **Default**.



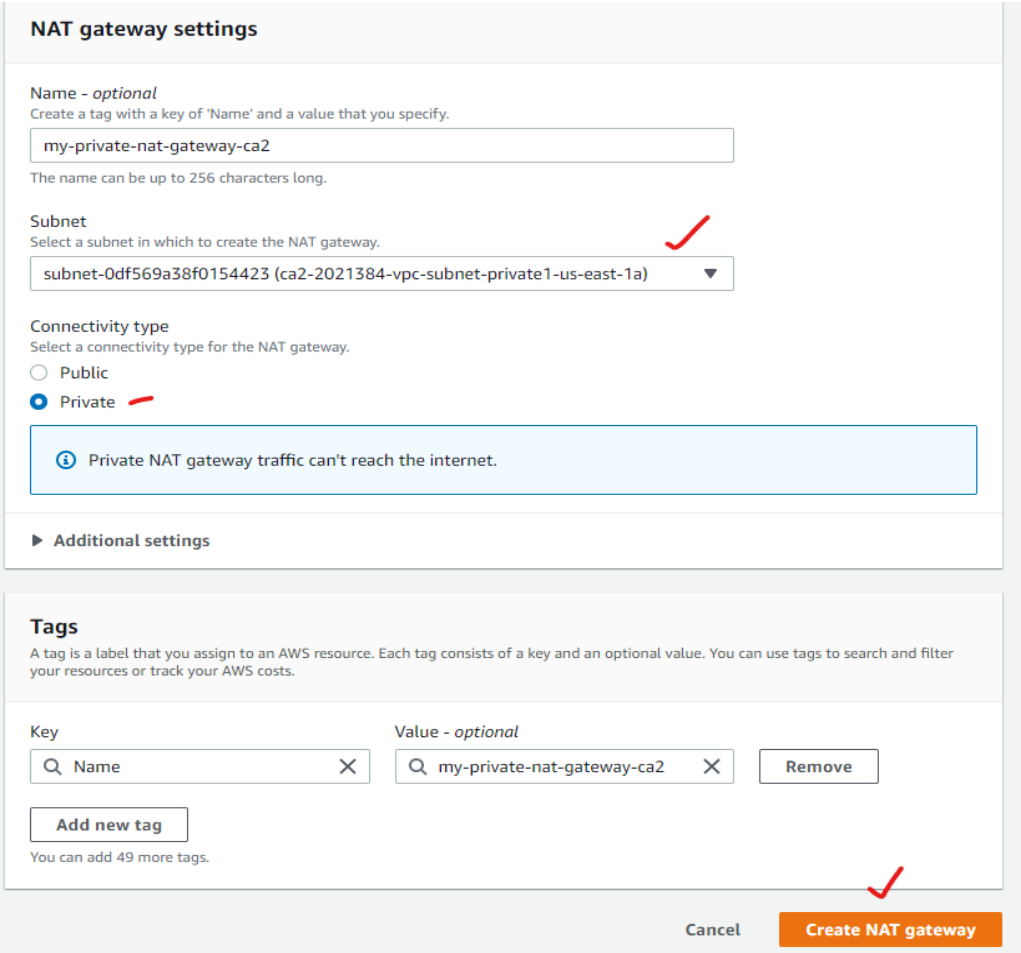
- I chose number of **Public** and **Private subnets** at **2**.
- **Nat gateway** at **None**.
- **DNS hostnames** and **DNS resolution** both must be **enabled**.
- **Click** on create **VPC.**



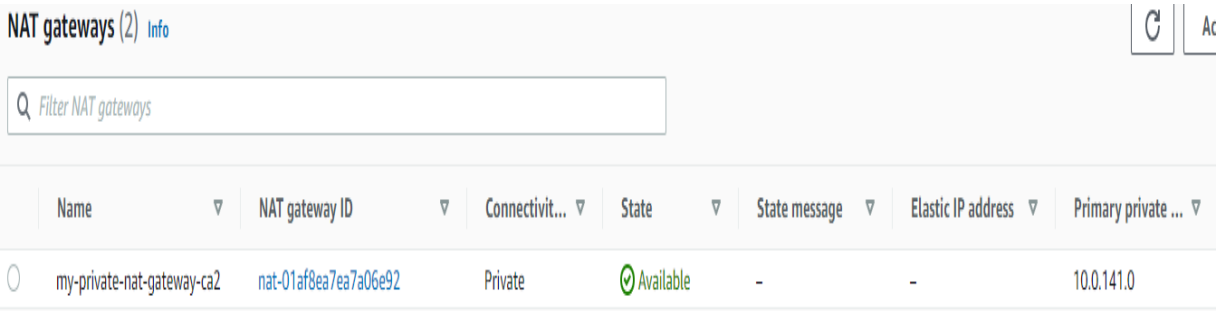
- **Ca2-2021384-vpc** has **created** successfully as shown below:



- Navigate the **NAT gateway** from left menu of VPC.
- **Click** on **create NAT gateway**.
- I gave **name** as **my-private-nat-gateway-ca2**.
- Then, I selected the **Private1 subnet**.
- **Connectivity type** on **Private**.
- **Click** on **create**.



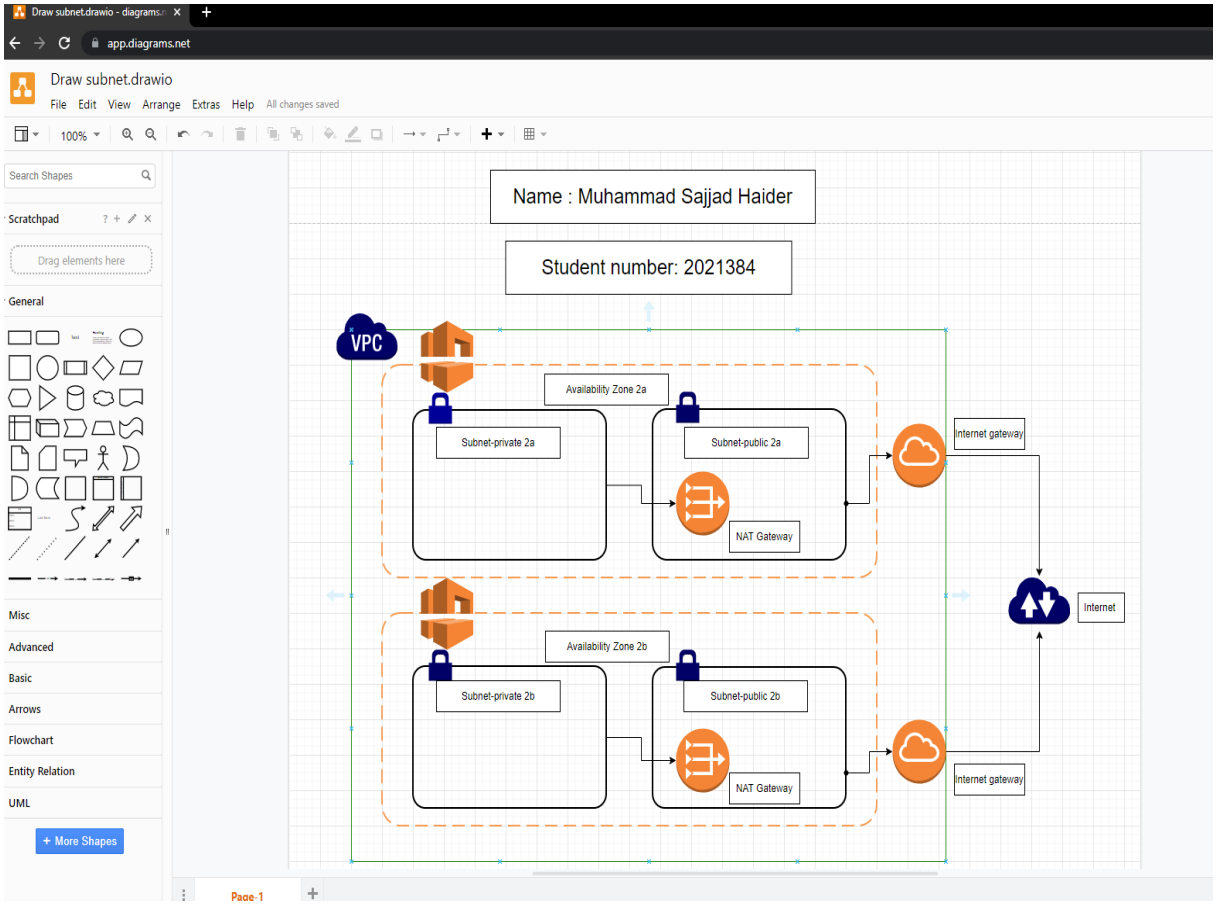
- **Private NAT gateway** has **created** successfully.



Part 3: AWS VPC diagram

- I used **Draw.io** and **draw** the **AWS VPC** with my **name** and **student number** which I just **made** in **Part 2**.

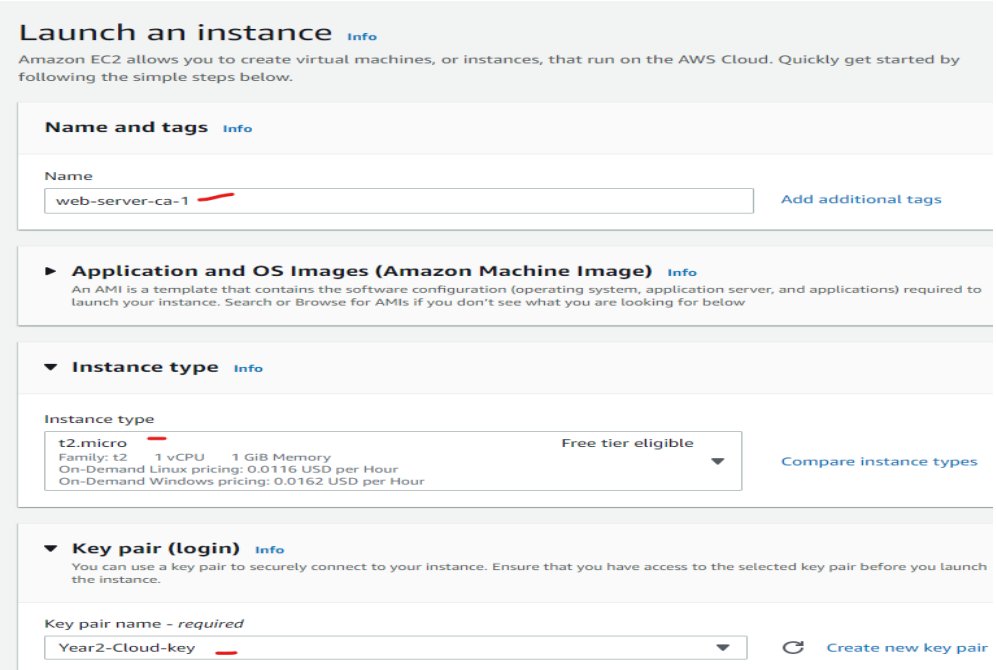
- It can be seen that **VPC** in **one region** with **two availability zones**.
- **Each AZ** have **one public** subnet and **one private** subnet.
- There is an **Internet Gateway for** the **public** subnet and a **NAT Gateway for** the **private** subnet.



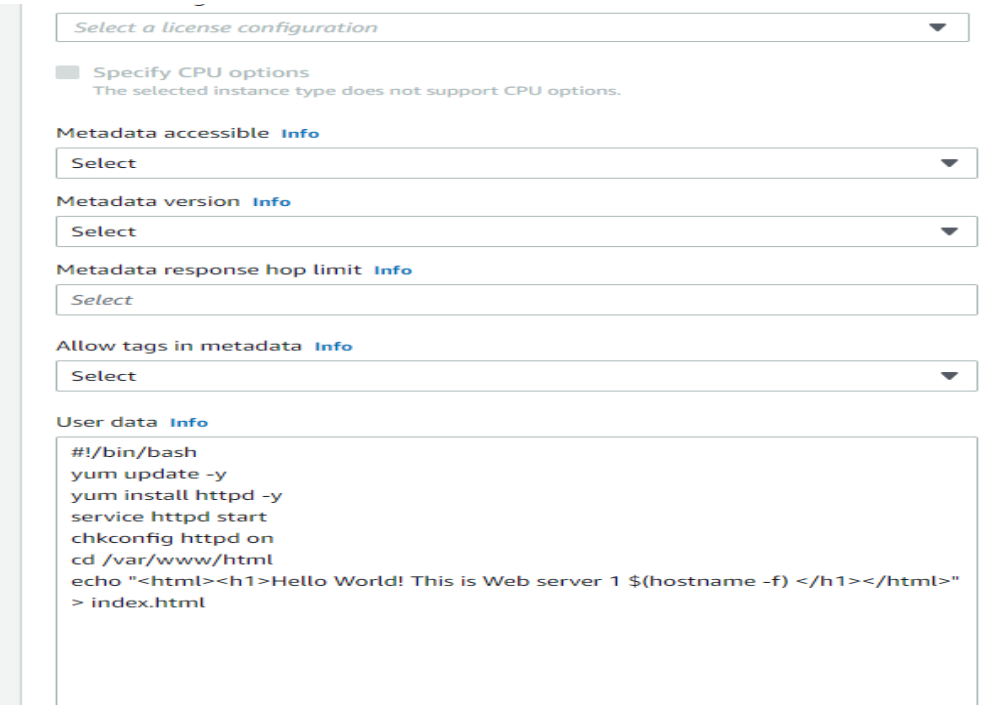
Part 4: Load Balancer

- For the **load balancer**, First I made **five Linux servers**.
- I made them with the **t2.micro instance type** and **Year-2-cloud-key**.

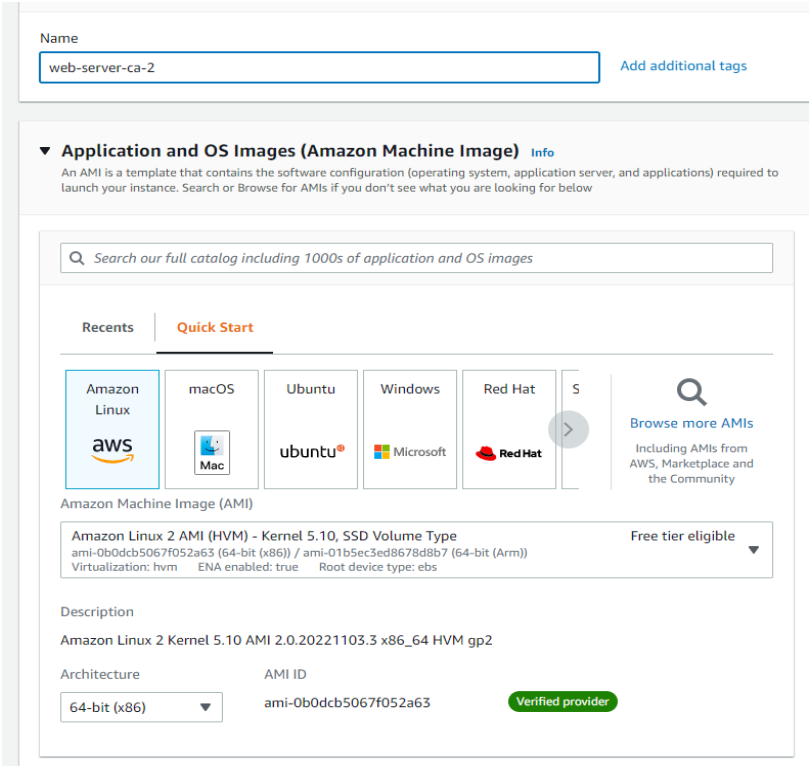
- **First server** name is **web-server-ca-1** as shown below:



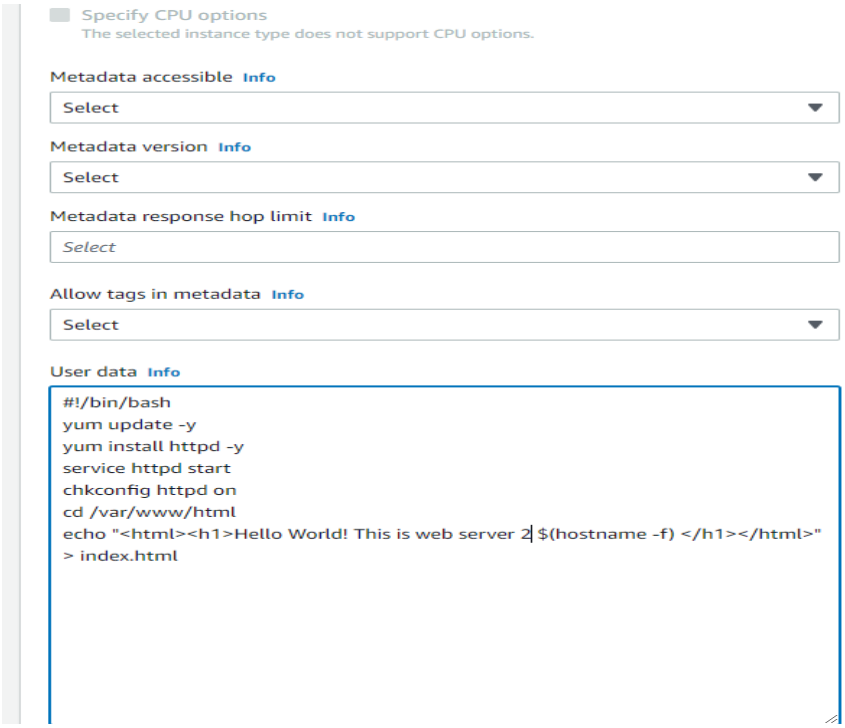
- I used the **bootstrap** which is given on **Moodle** and I **changed** the statement as **"Hello World! This is Web server 1"**.



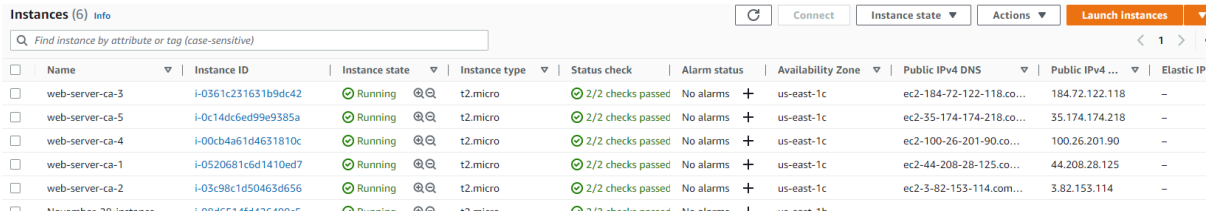
- I created **second instance** with the name **web-server-ca-2** with same procedure.



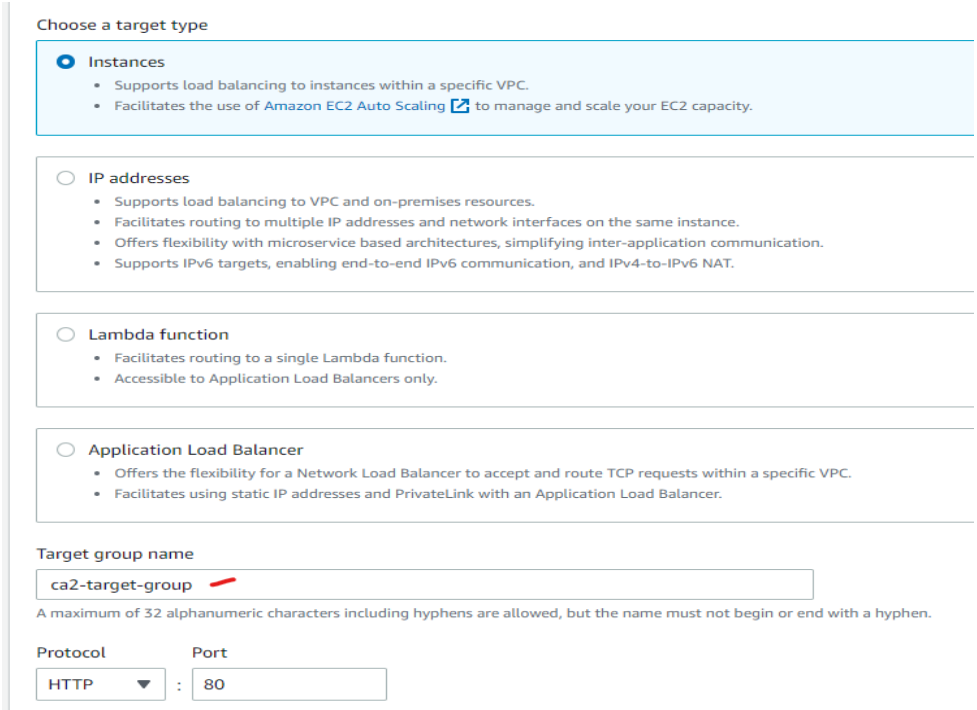
- Here is the **bootstrap** of **Second instance** and I **changed** the **statement** as "**Hello World! This is Web server 2**".



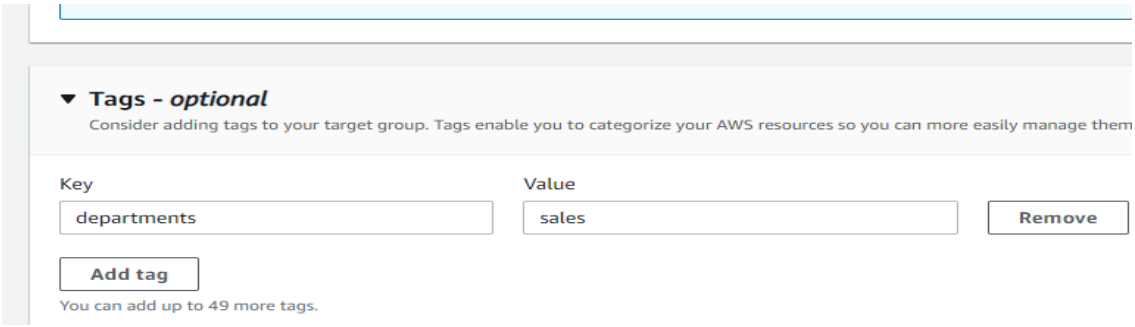
- By following the **same procedure**, I **made five instances** with the **numbering 1 to 5** as shown in the screenshot below:



- Then, I created **Target group** with the name **ca2-target-group**, **sticking** with **instances** in **target type**.

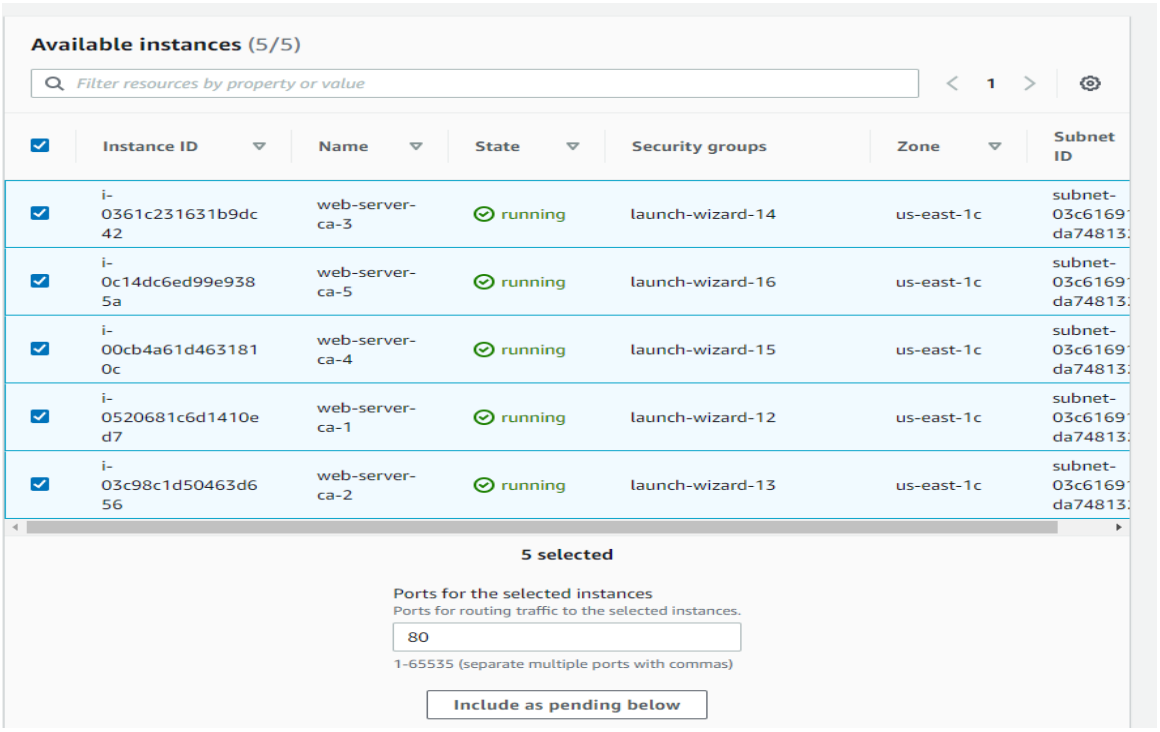


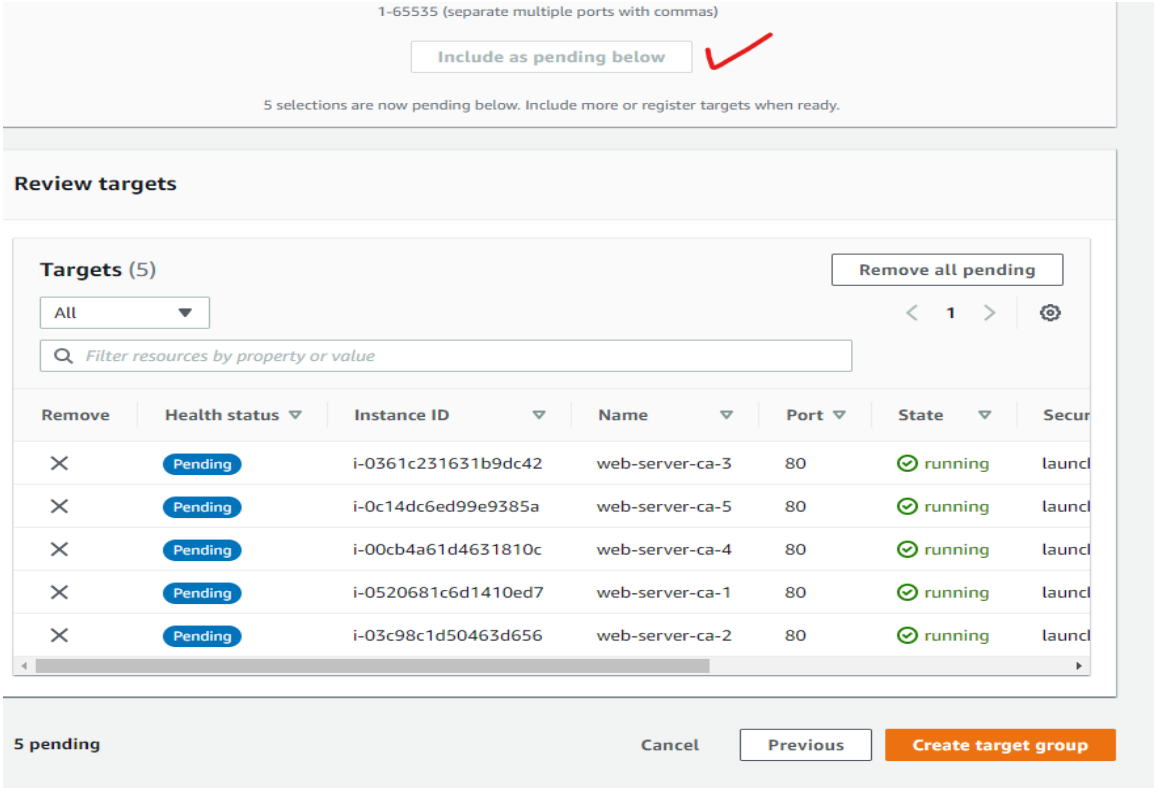
- I also added **tags** with **Key** as **Departments** and **Value** as **Sales** but its **optional**.
- Click on **Next**.



- I **selected** all **five instances** in **available instances**.
- **Click** on **include as pending below** option at bottom.

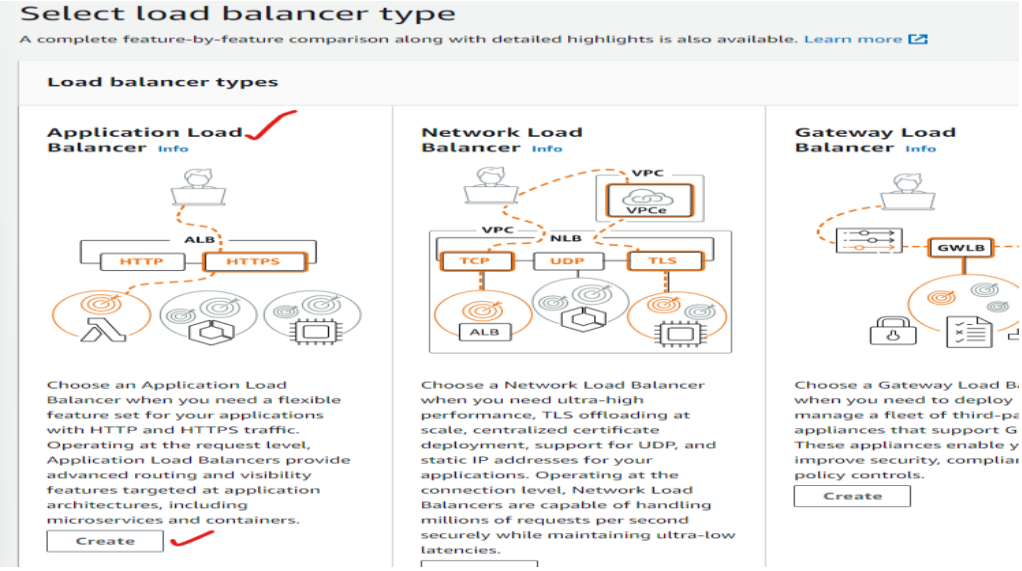
- And **click** on **create target group**.



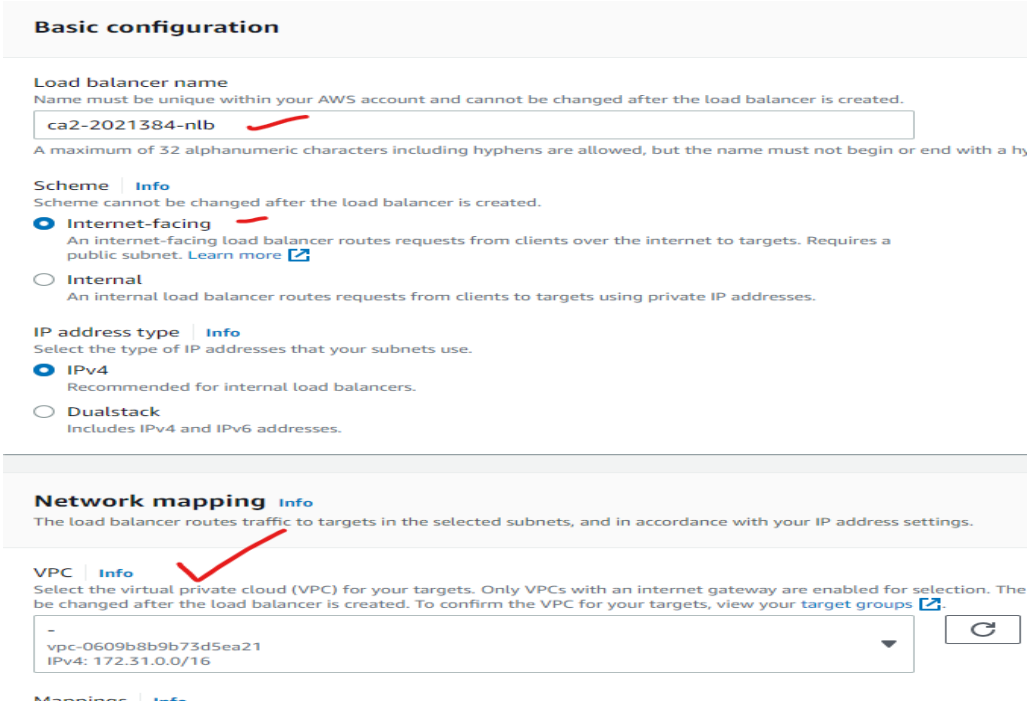


- After that, navigate the **load balancer** from the left menu and click on **create** load balancer.
- In **load balancer type**, select **Application Load Balancer**.

- And click on **create**.

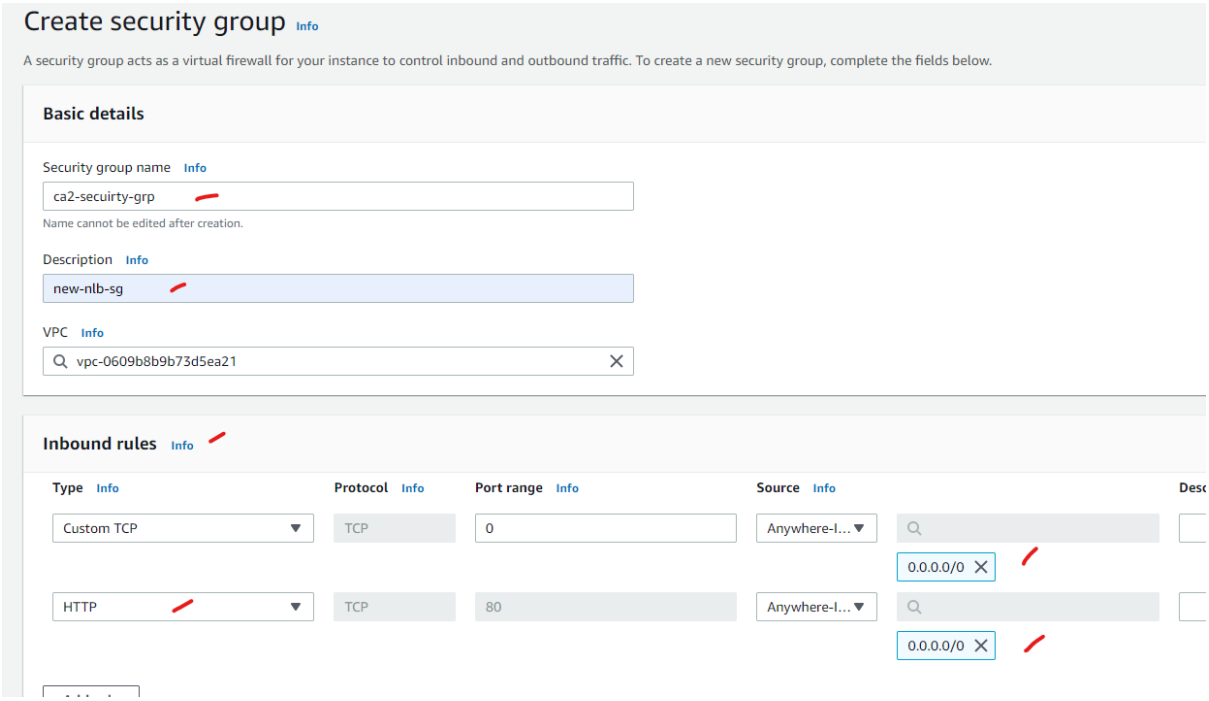


- In **Basic configuration**, I gave **load balancer** name as **ca2-2021384-nlb**.
- **Stick** with **Internet-facing** in **Scheme** and **IPv4** in **IP address type**.
- I selected **Default VPC** in **Network mapping**.

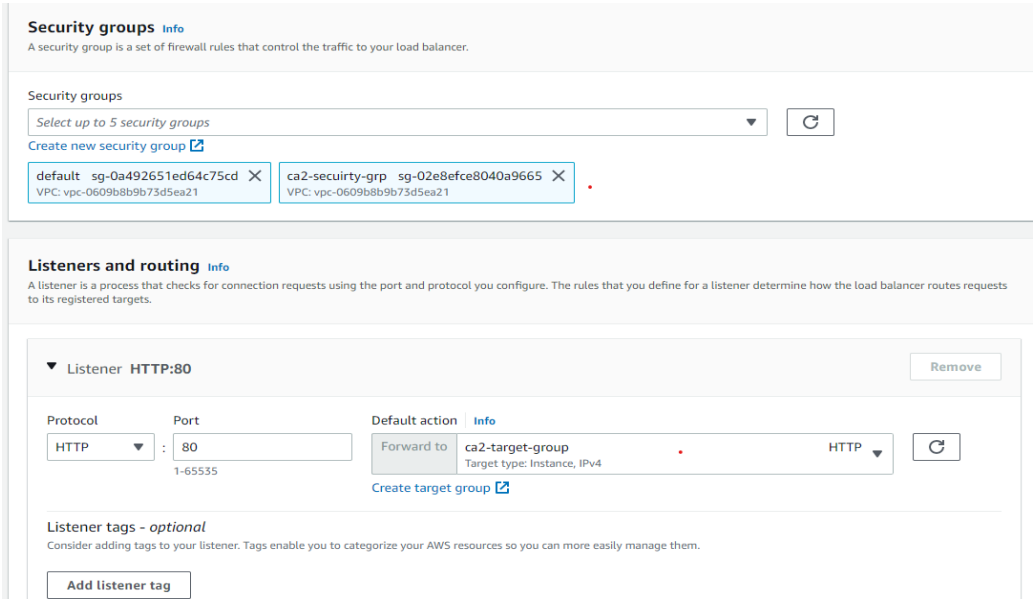


- Then, I **created new Security group** with the name **ca2-security-grp** and **Description** (Optional) as **new-nlb-sg**.
- I also set **Inbound rule** with **HTTP**.

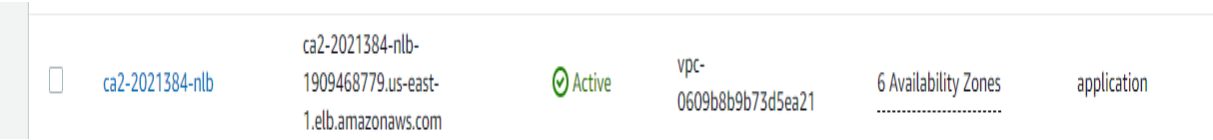
- Click **next**.



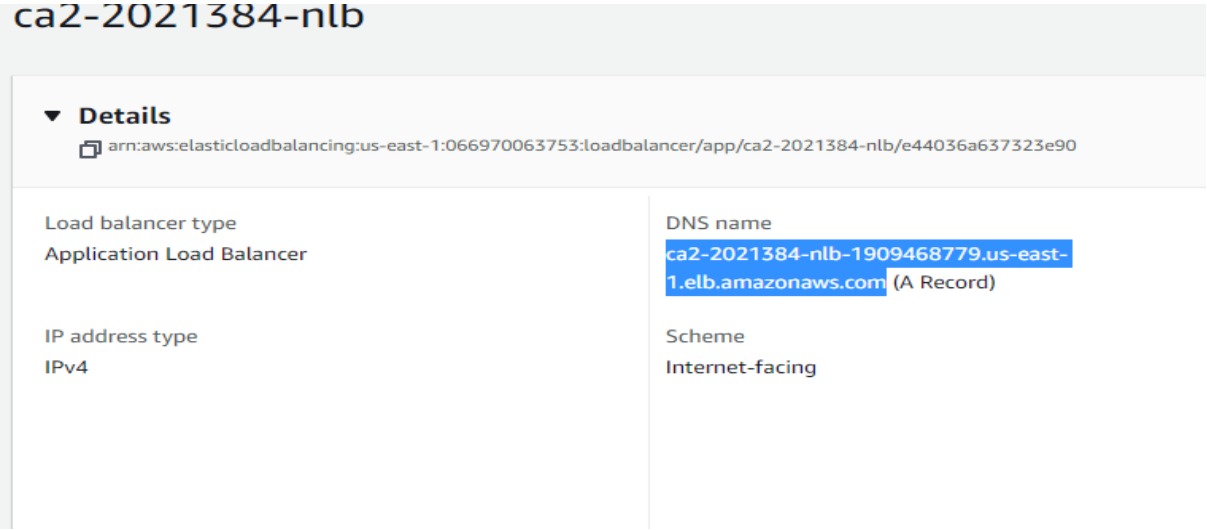
- I **selected** the new **security group** named **ca2-security-grp** which I **made before** and also **selected ca2-target-group** in **Default action** of **Listeners and routing**.



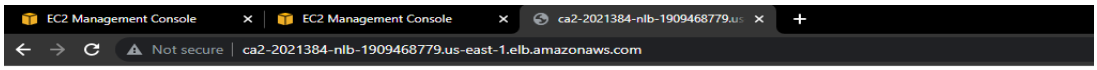
- After **clicking** on **create**, it can be seen in shot below that **ca2-2021384-nlb** has **created** successfully.



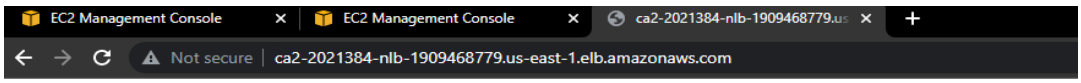
- **Copy** the **DNS name** of **ca2-2021384-nlb** and **search** it into **google**.



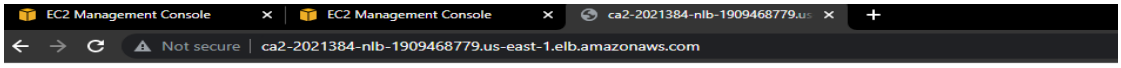
- It can be **seen** in screenshots below that the **load balancer is working properly.**
- It **showed** the **page** of **every instance** by **refreshing** the **web page** again and again.



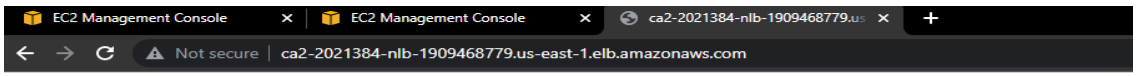
Hello World! This is Web server 1 ip-172-31-90-254.ec2.internal



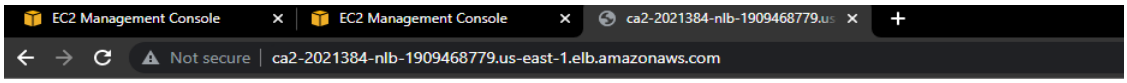
Hello World! This is web server 2 ip-172-31-82-89.ec2.internal



Hello World! This web server 03 ip-172-31-90-122.ec2.internal



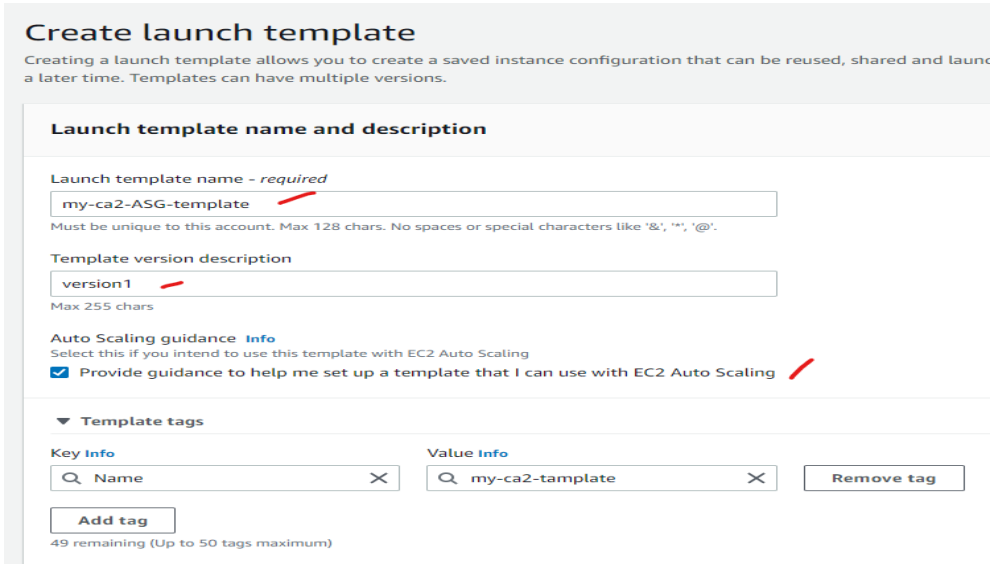
Hello World! This is web server 5 ip-172-31-81-154.ec2.internal



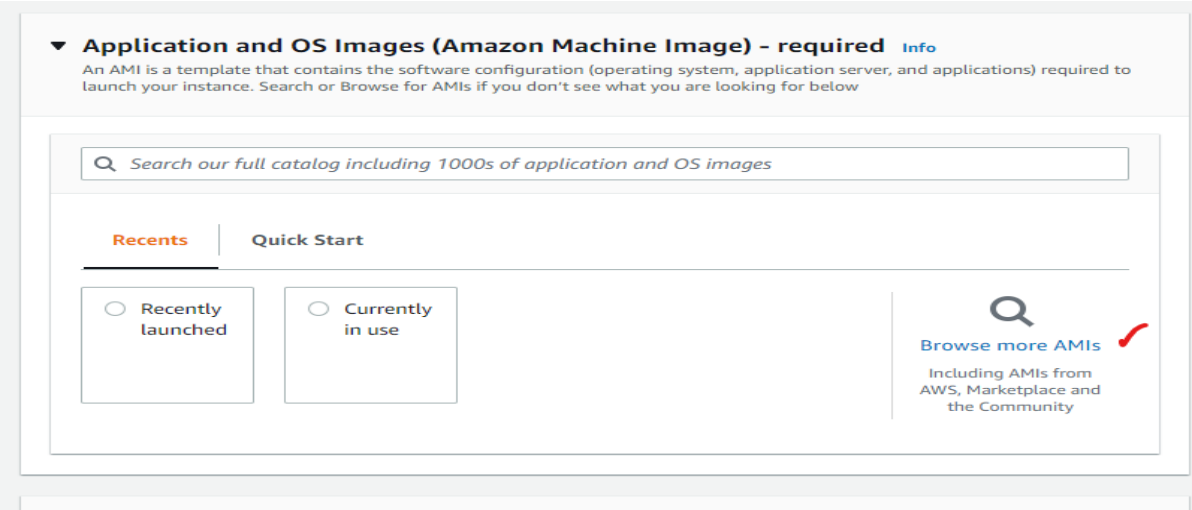
Hello World! This is web server 04 ip-172-31-88-68.ec2.internal

Part 5: Auto Scaling Group

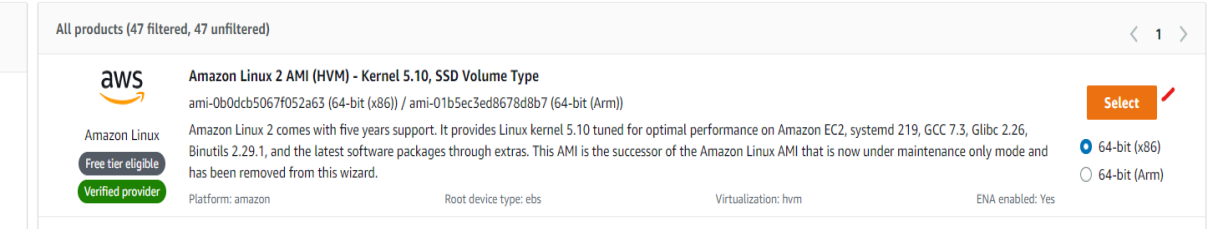
- For **Auto scaling**, Firstly I made **template** for it by navigating **Launch Template** from left menu.
- **Click** on **Launch template**.
- I gave it to **name** as **my-ca2-ASG-template** and write **version1** in **template version** option.
- Make sure to **tick** the **Auto Scaling guidance**.
- I gave **tag** value as **my-ca2-tamplate**.

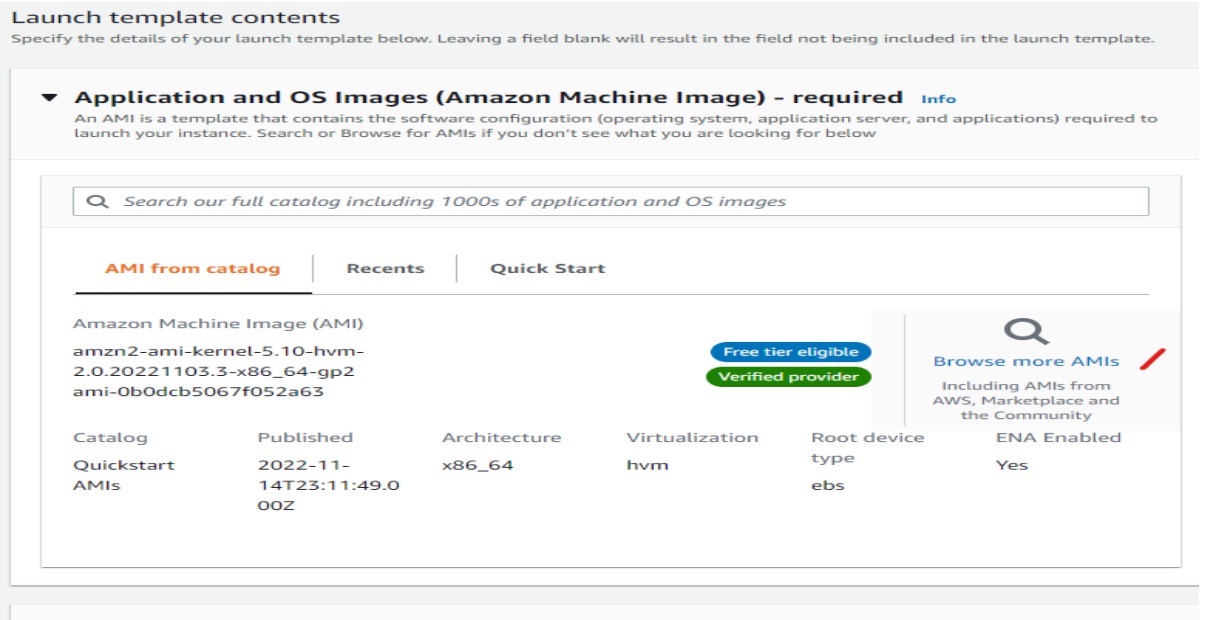


- **Click** on **Browse more AMIs.**

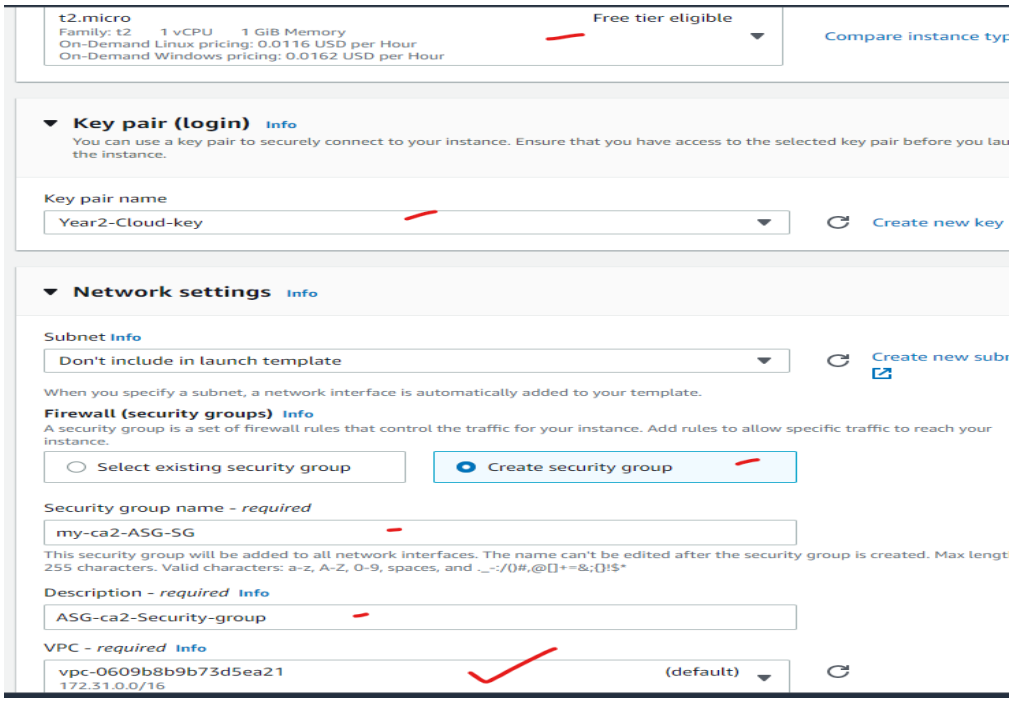


- I **selected** the **very first Amazon Linux 2 AMI**.
- And it **showed** as **selected** below.

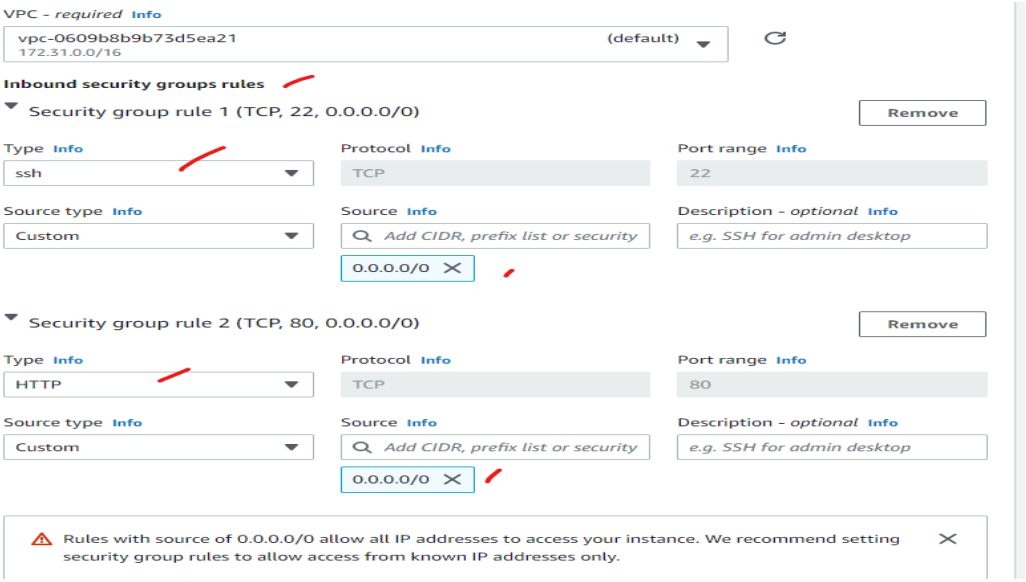




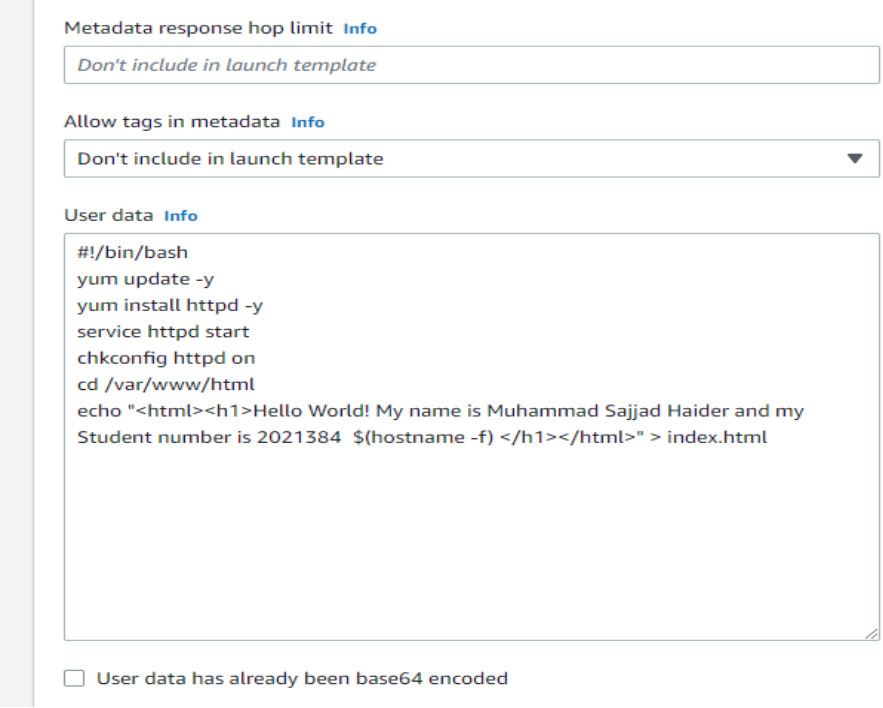
- I used **t2.micro** free tier eligible for this instance.
- Selected **Year2-Cloud-key** for key pair.
- In **Firewall**, I **created new security group** with name **my-ca2-ASG-SG** and gave **description** as **ASG-ca2-Security-group**.
- I **selected default VPC** as shown below in screenshot.



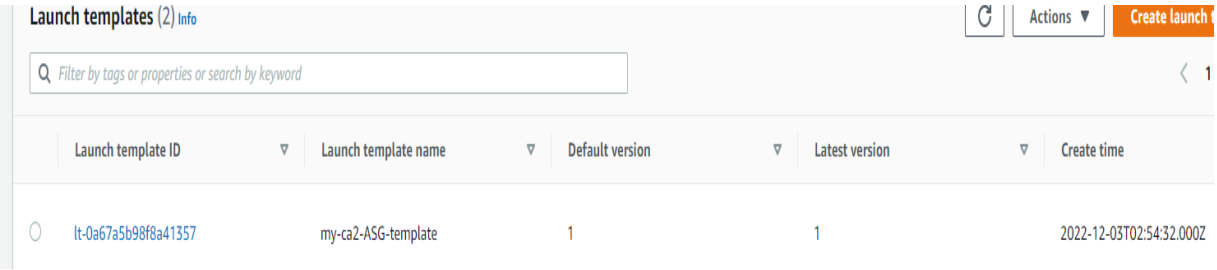
- I have **created two inbound security groups rules**, **one** with the type on **SSH** and the **other** with the type off **HTTP**.
- I chose **CIDR 0.0.0.0/0** for **both** of them.



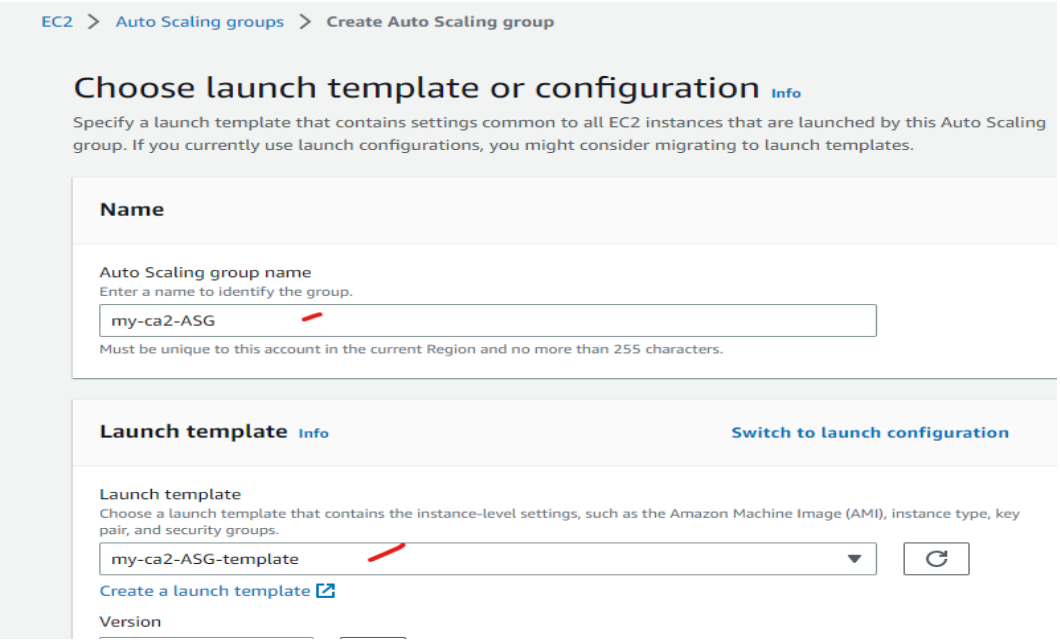
- I have set **Bootstrap** in **additional information** and wrote my **name** and **student number** in it.
- Click on **create**.



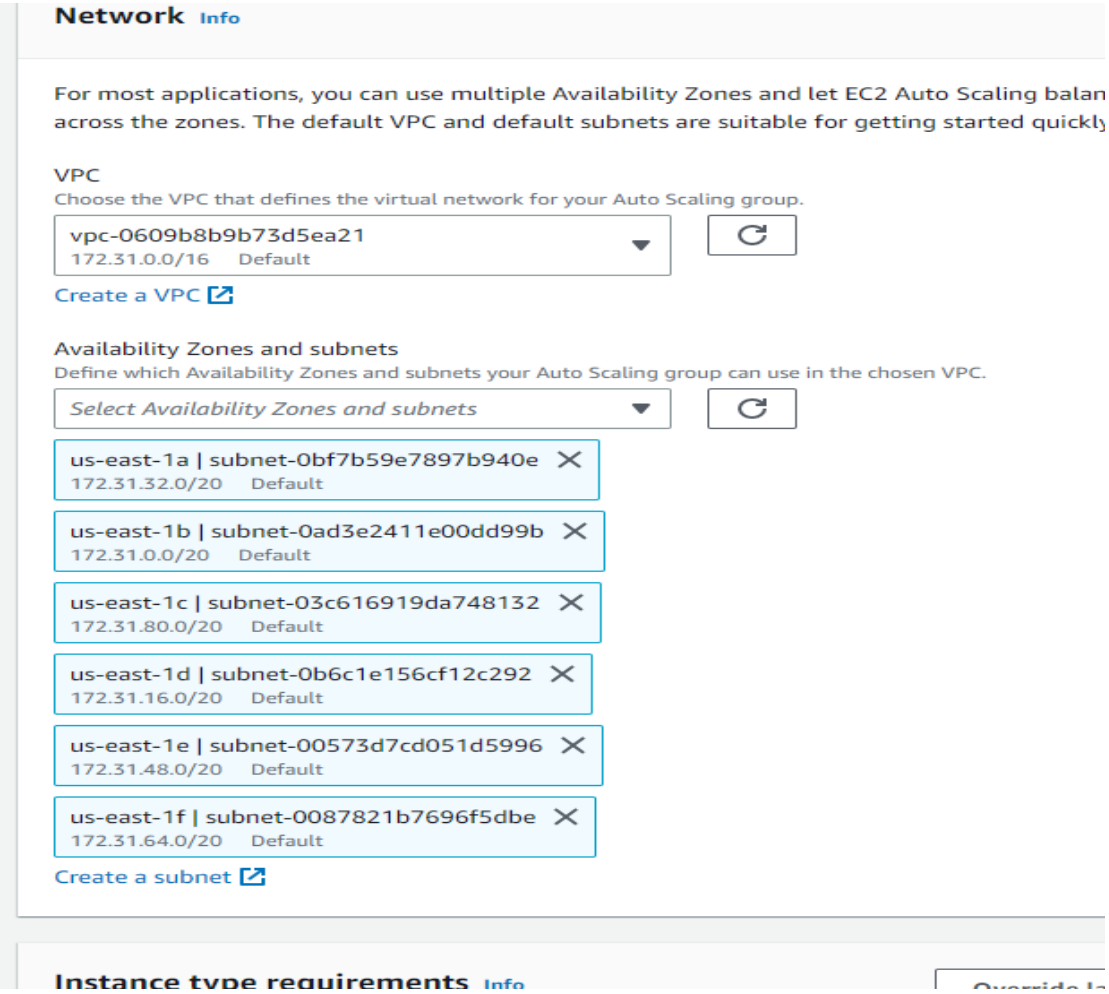
- It can be seen that **my-ca2-ASG-template** has **created**.



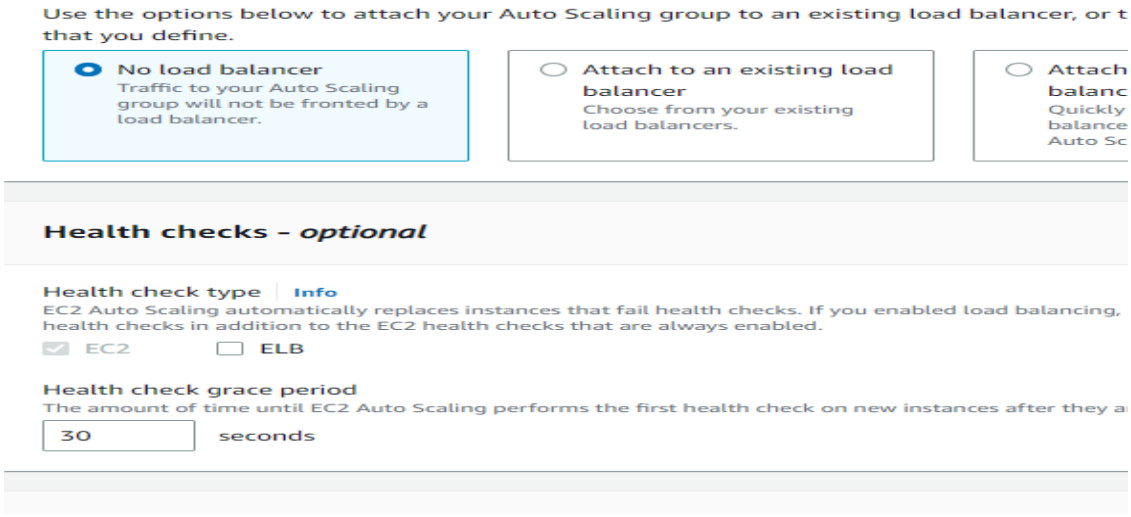
- In **EC2**, navigate **Auto scaling groups**.
- Click on **create Auto Scaling group**.
- I gave it to **name** as **my-ca2-ASG**.
- In **Launch template**, I **selected** the **template** which I have just created with the name **my-ca2-ASG-template**.
- Click **next**.



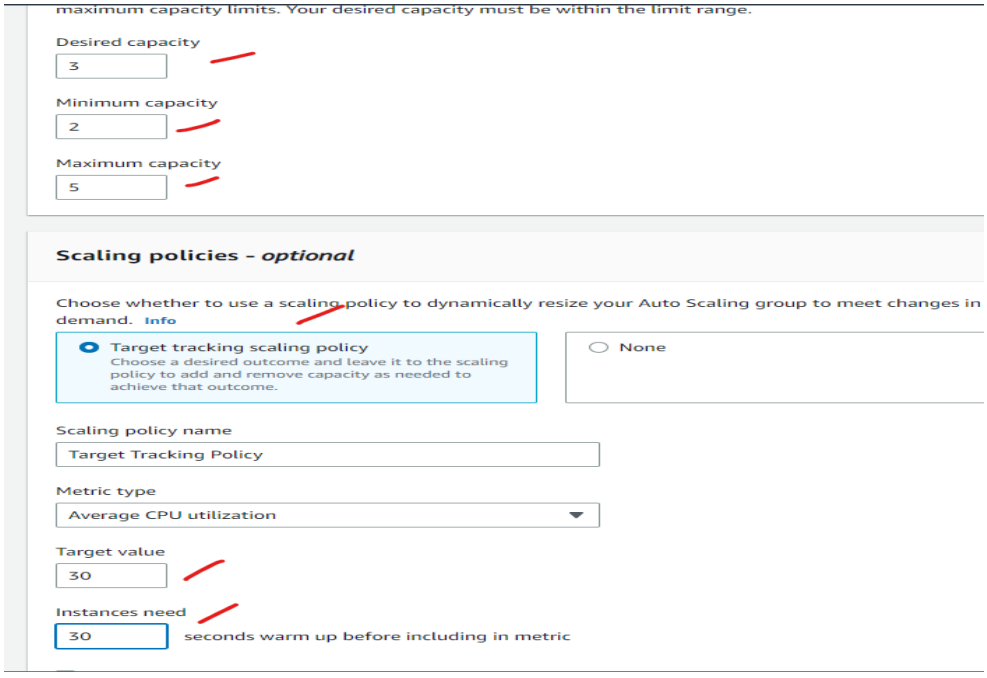
- In **Network**, I selected **default VPC** and all **default subnets**.
- Click **Next**.



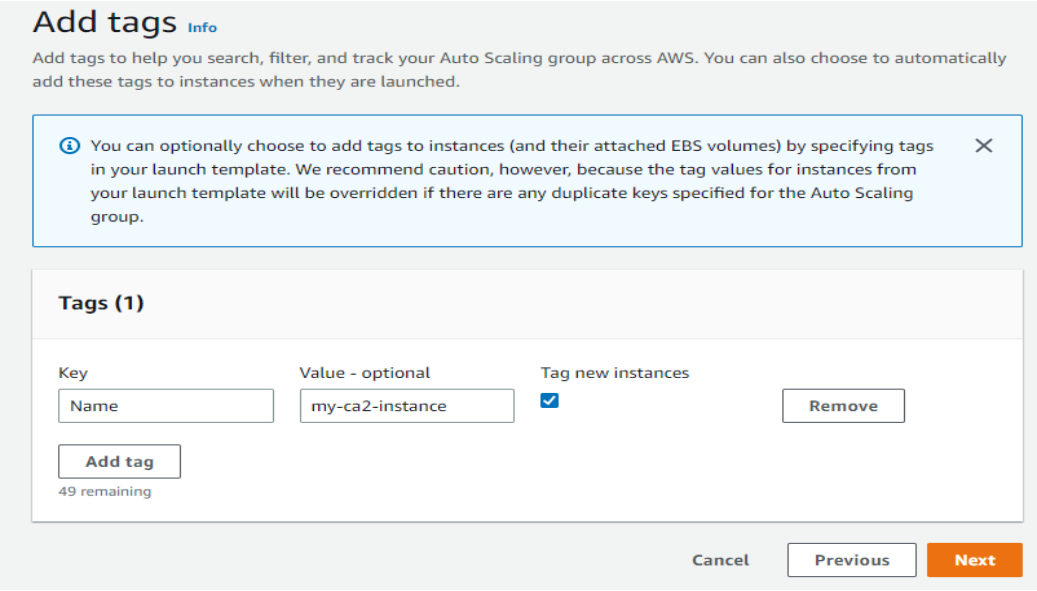
- **Stick** with **NO load balancer** and set **30 seconds** to **Health checks** (Optional).
- Click **Next**.



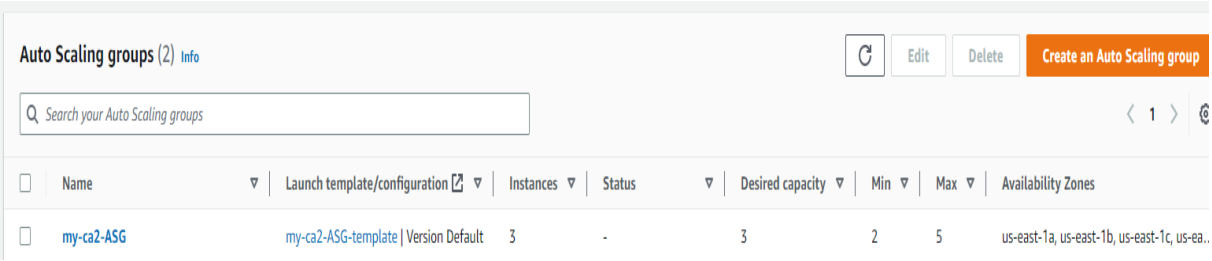
- I have set **3 Desired capacity**, **2 Minimum capacity** and **5 Maximum capacity**.
- In **Scaling policies** (Optional), I selected **Target tracking scaling policy**.
- I have set **target value 30**.
- I have set **warm up** time for **instances 30 seconds**.



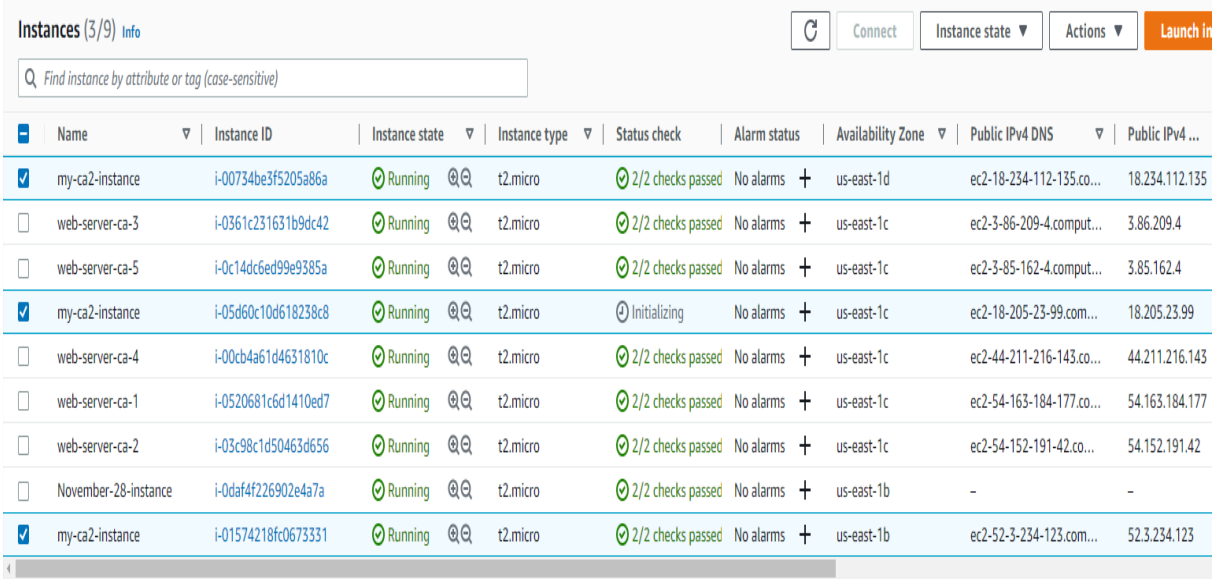
- I gave **tags value** (optional) as **my-ca2-instance** and click **Next**.



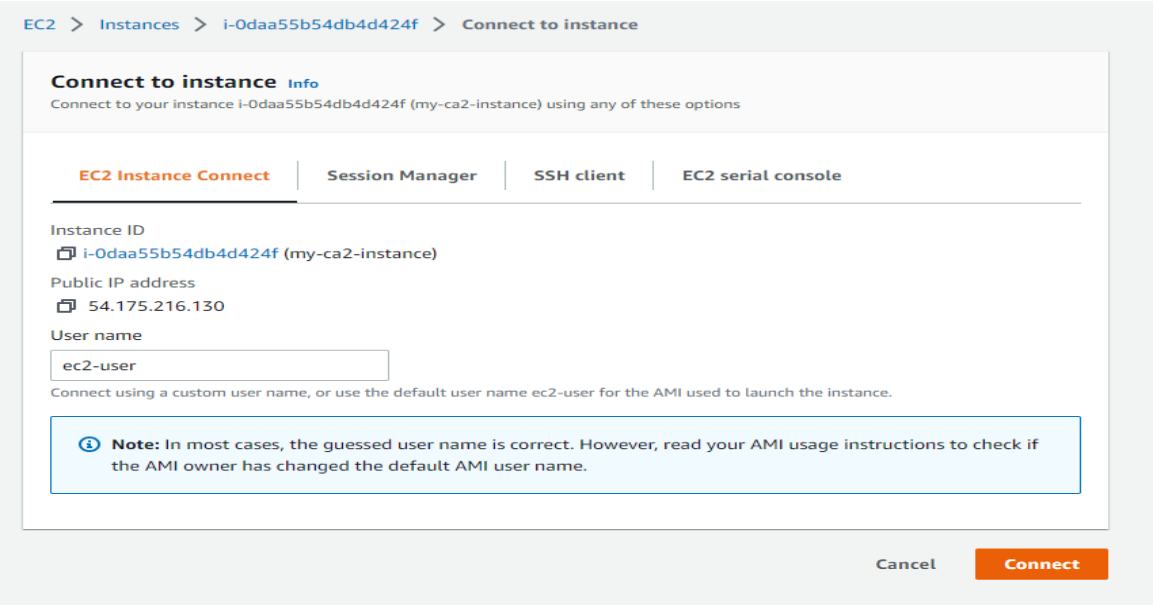
- It can be seen that **my-ca2-ASG** has **created** perfectly.



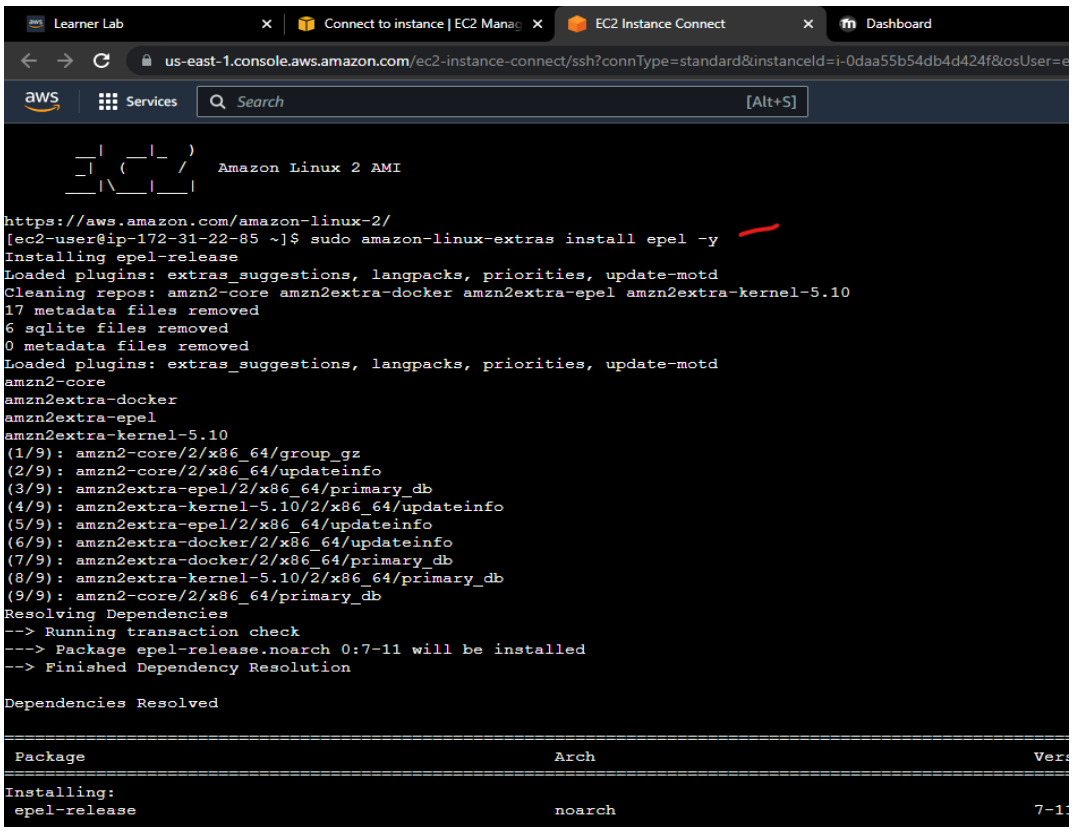
- There are **3 instances** are **running** with the **name** of **my-ca2-instance** which I gave as a **tag** above while **making my-ca2-ASG**.
- All three has **come again** after **deleting** as a **warm up**.



- **Open** one **instance**, Go to **Connect** inside it.
- Click **Connect**.



- I **used** the **stress parameters script** which has been given in Moodle.
- I **applied first command** of this **script** in this **Linux instance**.



- After that, I applied **second command**.

```
 66  php8.1                    available     [ =stable ]
[ec2-user@ip-172-31-22-85 ~]$ sudo yum install stress -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
228 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
---> Package stress.x86_64 0:1.0.4-16.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                                                 Arch
================================================================================
Installing:
 stress                                                  x86_64

Transaction Summary
================================================================================
Install  1 Package

Total download size: 39 k
Installed size: 94 k
Downloading packages:
warning: /var/cache/yum/x86_64/2/epel/packages/stress-1.0.4-16.el7.x86_64.rp
Public key for stress-1.0.4-16.el7.x86_64.rpm is not installed
stress-1.0.4-16.el7.x86_64.rpm
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
Importing GPG key 0x352C64E5:
 Userid     : "Fedora EPEL (7) <epel@fedoraproject.org>"
 Fingerprint: 91e9 7d7c 4a5e 96f1 7f3e 888f 6a2f aea2 352c 64e5
 Package    : epel-release-7-11.noarch (@amzn2extra-epel)
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : stress-1.0.4-16.el7.x86_64
  Verifying  : stress-1.0.4-16.el7.x86_64

Installed:
```

- At last, I gave the **stress command** with **30-cpu** and **timeout-60**.

```
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-7
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : stress-1.0.4-16.el7.x86_64
  Verifying  : stress-1.0.4-16.el7.x86_64

Installed:
  stress.x86_64 0:1.0.4-16.el7

Complete!
[ec2-user@ip-172-31-22-85 ~]$
[ec2-user@ip-172-31-22-85 ~]$ sudo stress --cpu  30 --timeout 60
stress: info: [9022] dispatching hogs: 30 cpu, 0 io, 0 vm, 0 hdd
```

i-0daa55b54db4d424f (my-ca2-instance)

- It **wasn't showing** me any **stress**.

- So that I **applied** the **stress command** with **60-cpu** and **timeout-120**.

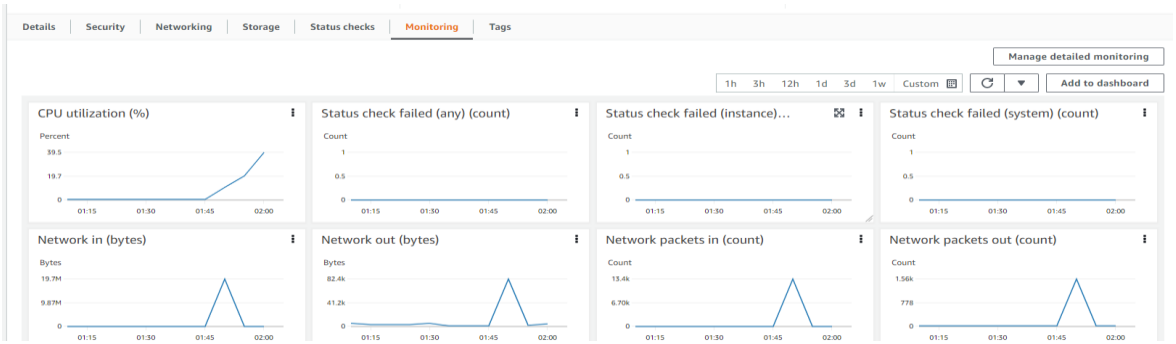
```
[ec2-user@ip-172-31-22-85 ~]$
[ec2-user@ip-172-31-22-85 ~]$
[ec2-user@ip-172-31-22-85 ~]$
[ec2-user@ip-172-31-22-85 ~]$ sudo stress --cpu  30 --timeout 60
stress: info: [9125] dispatching hogs: 30 cpu, 0 io, 0 vm, 0 hdd


stress: info: [9125] successful run completed in 60s
[ec2-user@ip-172-31-22-85 ~]$
[ec2-user@ip-172-31-22-85 ~]$
[ec2-user@ip-172-31-22-85 ~]$ sudo stress --cpu  60 --timeout 120
```

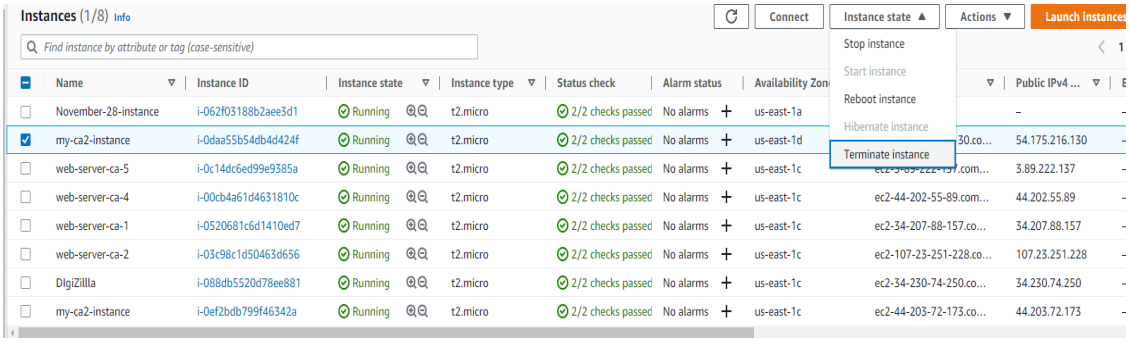
i-0daa55b54db4d424f (my-ca2-instance)

PublicIPs: 54.175.216.130 PrivateIPs: 172.31.22.85

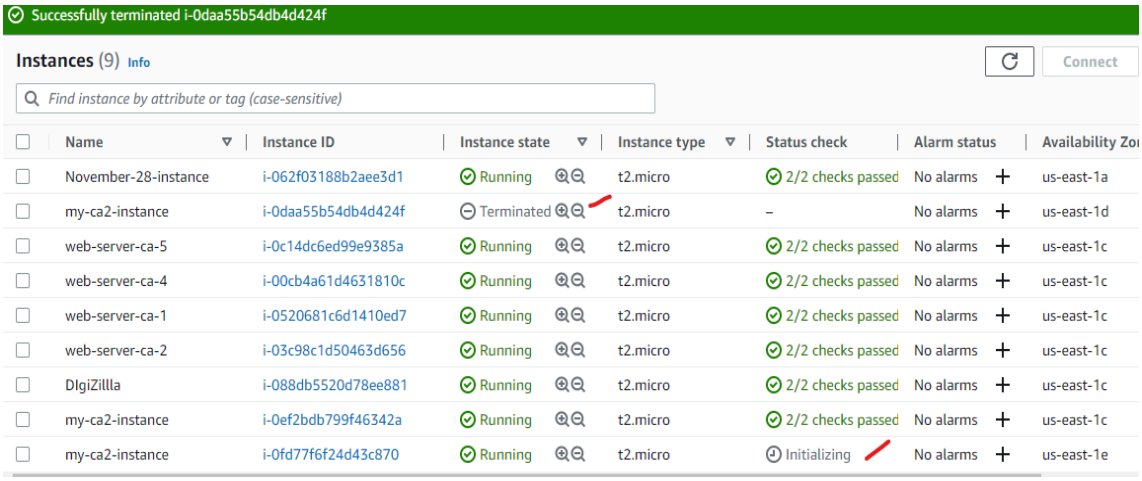
- Go to **inside instance** again and **click** on **Monitoring**.
- You can see the monitoring below:



- It can be seen that I have **two instances** here with the **name my-ca2-instance**.
- **Terminated** the **one instance** from them.

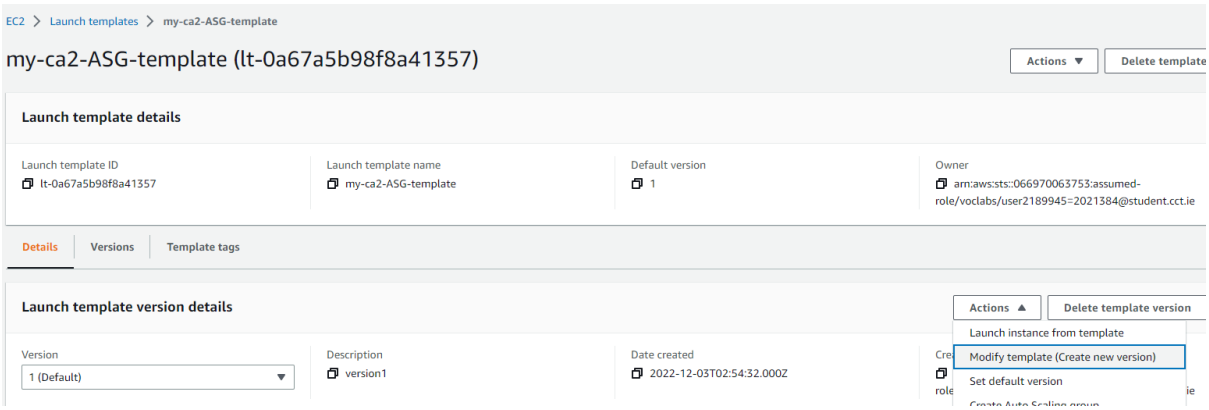


- But it can be seen that the **terminated instance** has **come again.**
- That's mean, **Auto scaling** is **working properly**.

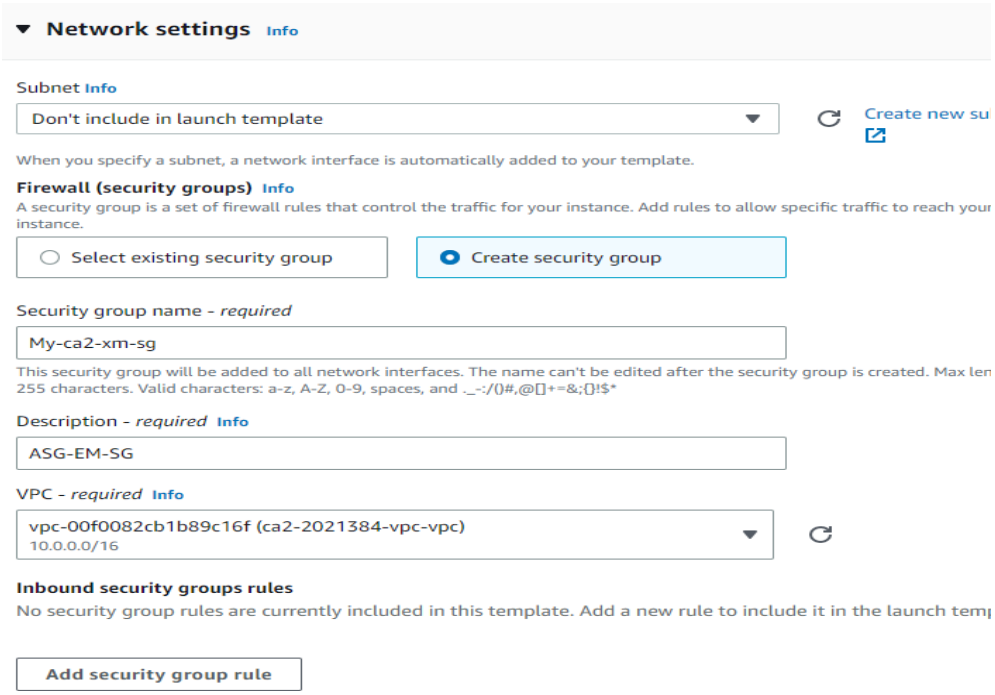


For Extra Marks

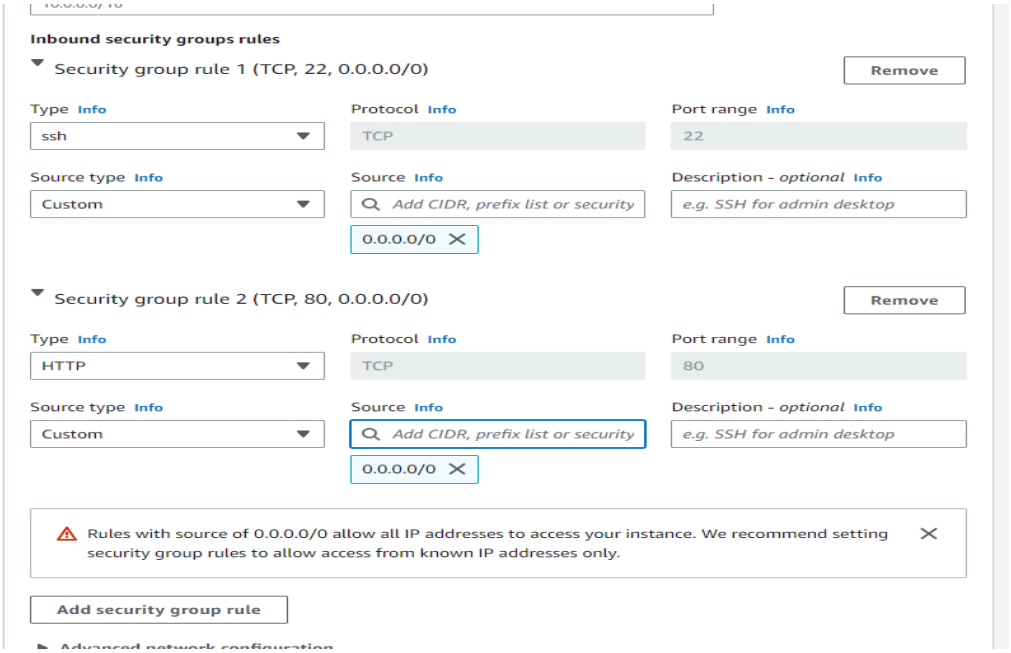
- For **Auto scaling** group **working** in **VPC** that I **created** in **Part 2** above as **ca2-2021384-vpc**.
- I did **some modifications** in **ASG template**.
- First, **Go** to **inside** the **my-ca2-ASG-template**.
- Click on **Actions** and select **Modify template**.



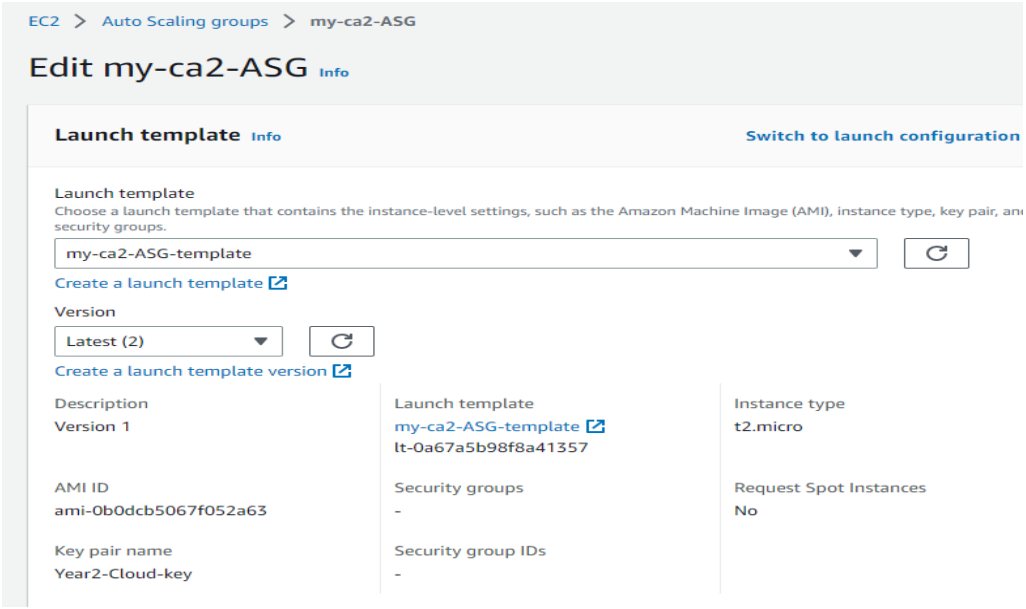
- **Template name** was **same**.
- But I **created** new **security group** with the name **My-ca2-xm-sg**.
- I gave **description** as **ASG-EM-SG**.
- I **selected** my **ca2-2021384-vpc**.



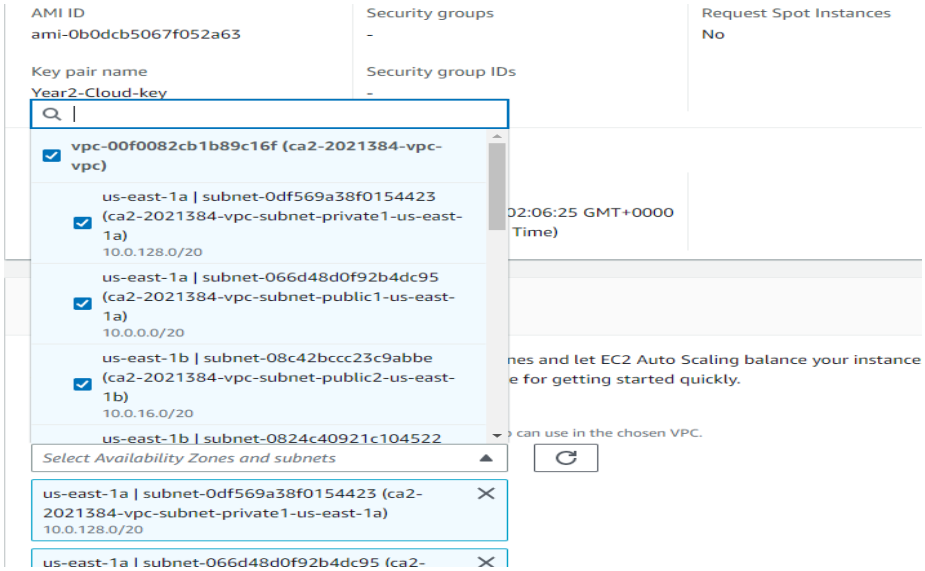
- Then I **added two inbound rules first** with the **type** of **ssh** and **other** one with **HTTP**.
- And set their **CIDR's** as **0.0.0.0/0**.
- Click on create.



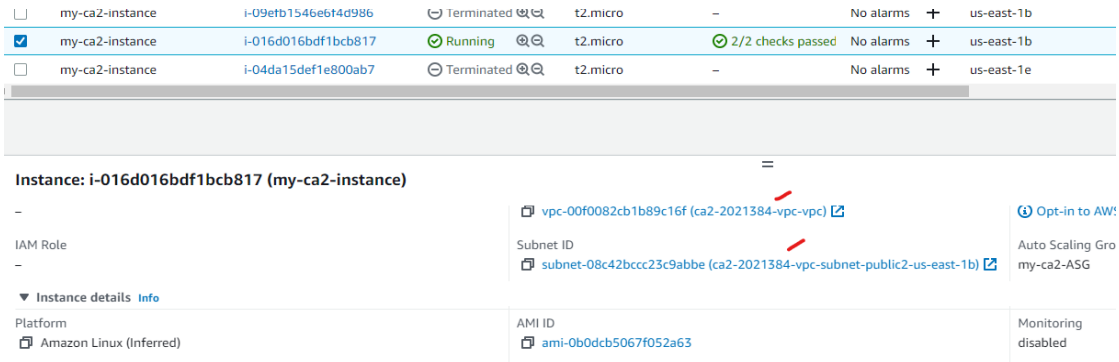
- After that **go inside** the **my-ca2-ASG** auto scaling group.
- **Edit** the **Launch template**.
- **Select Version** as **Latest (2)** which I just have **set above**.



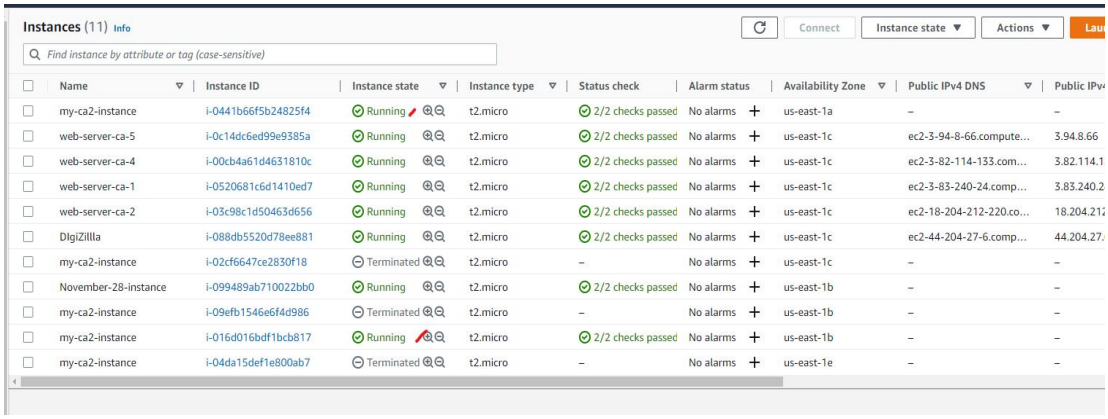
- Go down and **edit** the **networks** option **in this** launch template **edit page**.
- I **selected all** the **subnets** of **ca2-2021384-vpc**.



- It **can be seen** that my **new instances with** the personal VPC **ca2-2021384-vpc are running**.

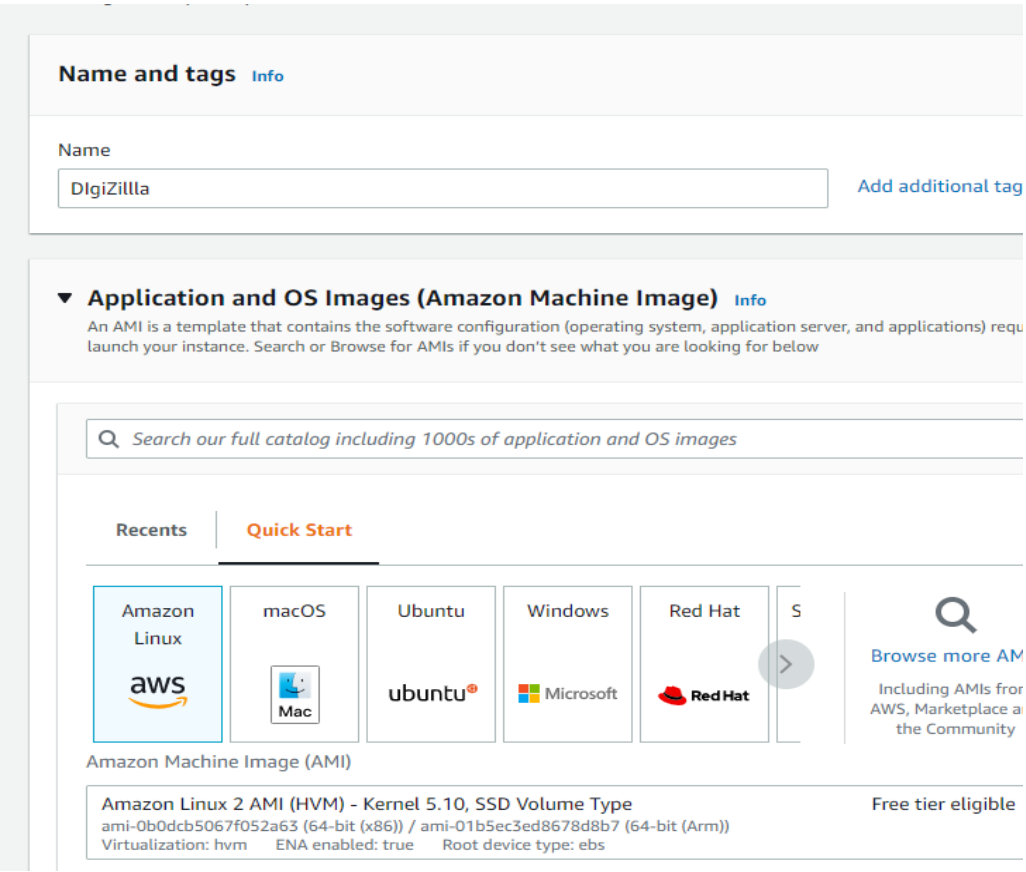


- These have also **come back after terminated**.
- That's mean **Auto scaling is working properly**.

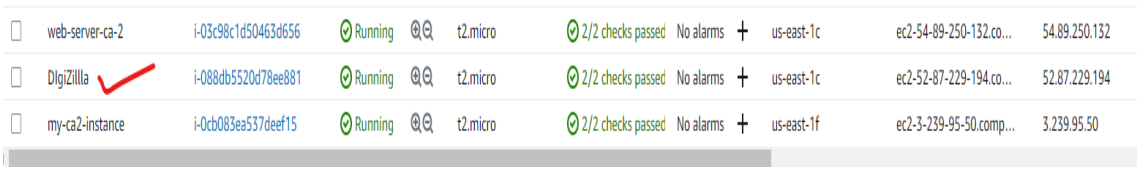


Part 6: Amazon Linux VM instance

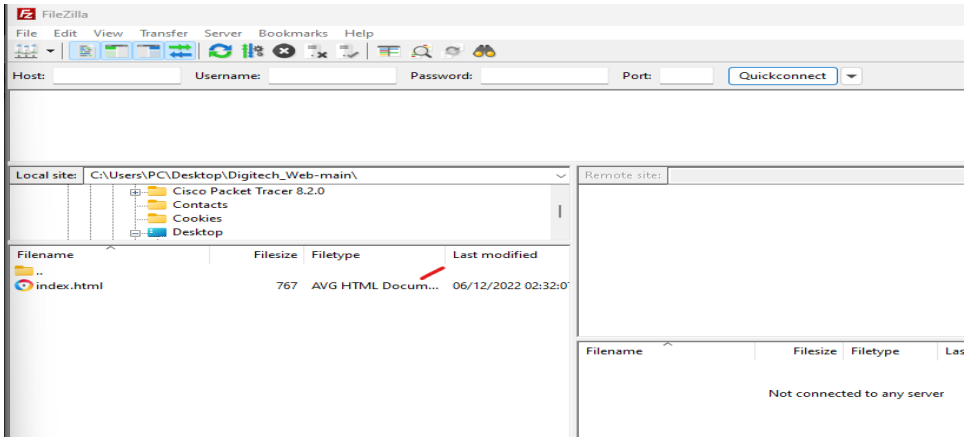
- In **EC2**, navigate **instances** and **click** on **launch instance**.
- I have created a new **Amazon Linux instance**.
- I gave it to name **DigiZilla**.
- I **made** it by **following** the **same procedure** which I **used** in **part 4** above for making 5 instances.



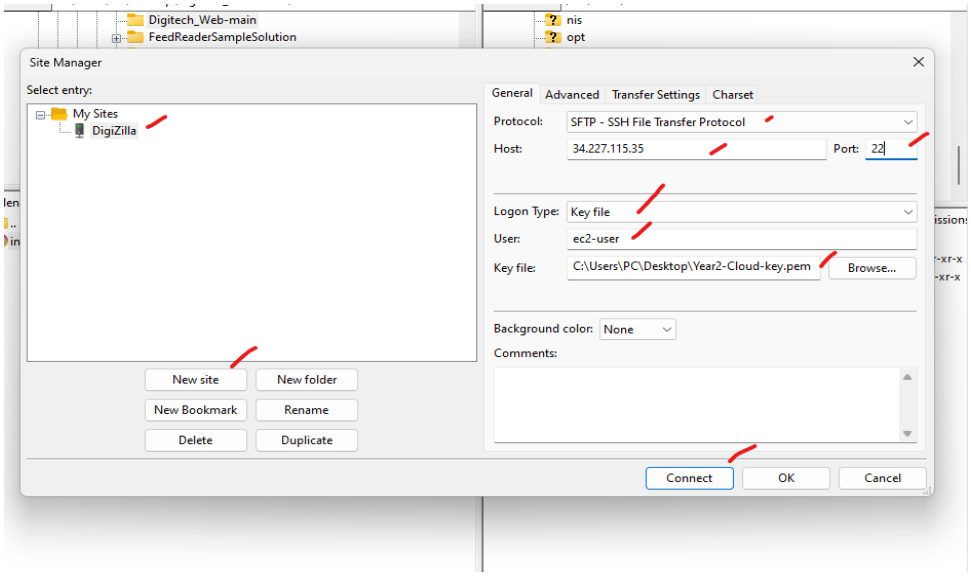
- **Amazon Linux DigiZilla** has **created**.



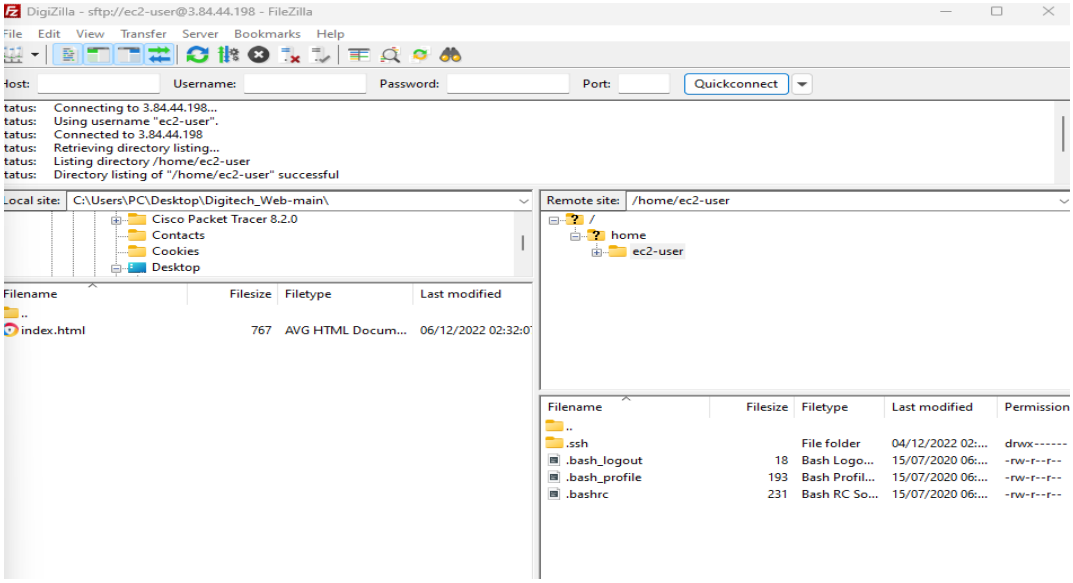
- I selected the **Digitech** website **inside** from my **device** in **FileZilla**.



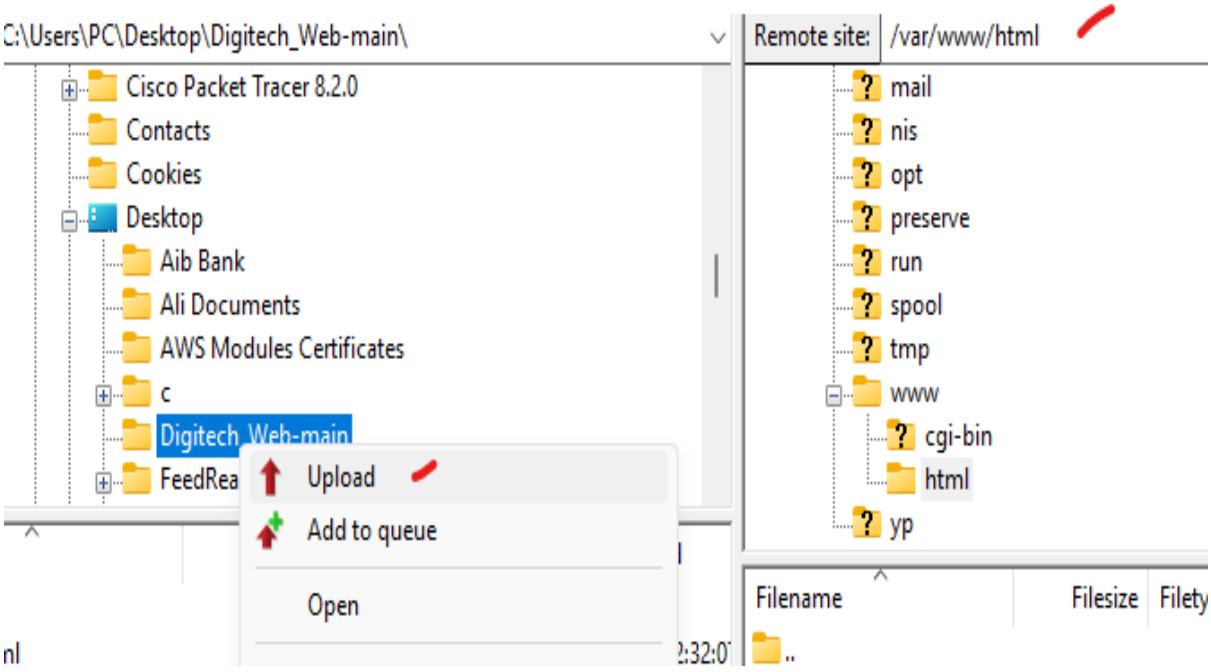
- In **top left** of FileZillla **software**, **Click** on **File**.
- Go to **Site manager** and Click on **new site**.
- I gave **name** as **DigiZilla**.
- Select **secure ftp** in **Protocol** drop down menu.
- In **Host**, paste the **IP address** of **DigiZilla instance**, which I made in AWS cloud.
- In **Logon type**, select **Key file**.
- I wrote **ec2-user**.
- Browse and gave the device **location** of my **Year2-cloud-key** and Click **Connect**.



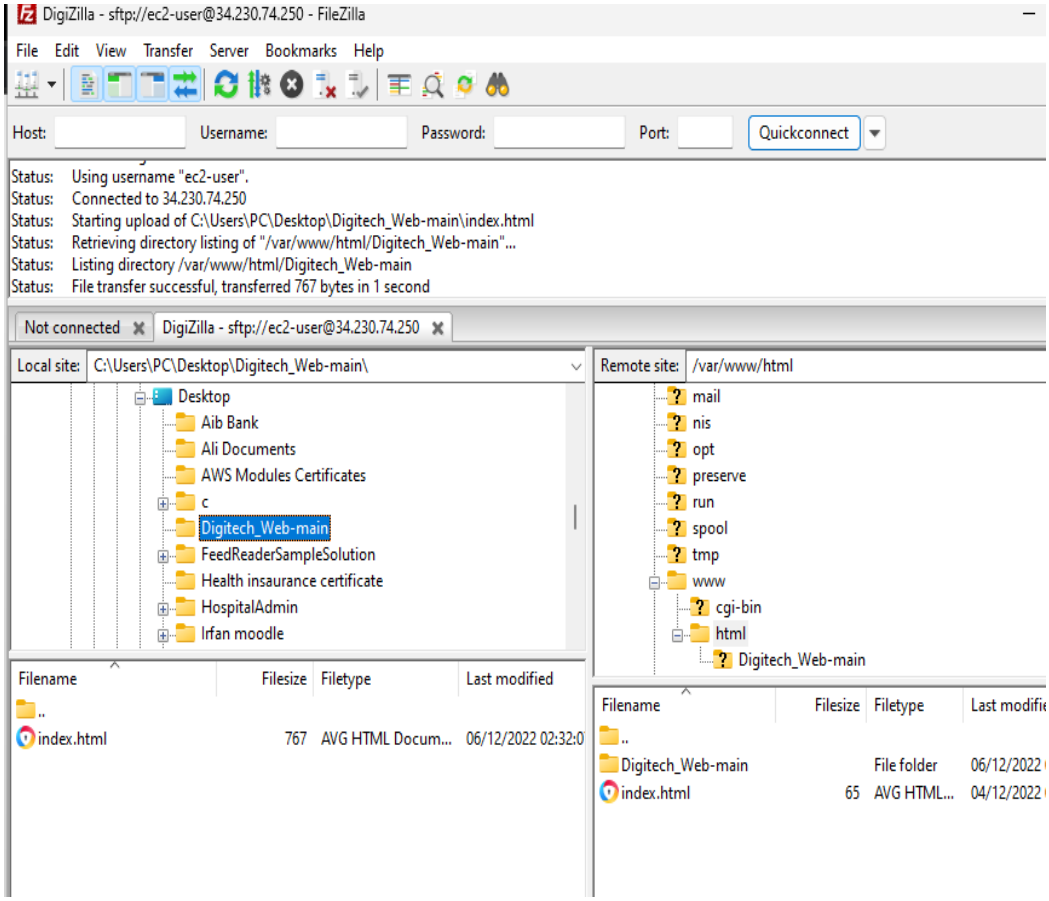
- It can be seen that, DigiZilla is **successfully connected**.



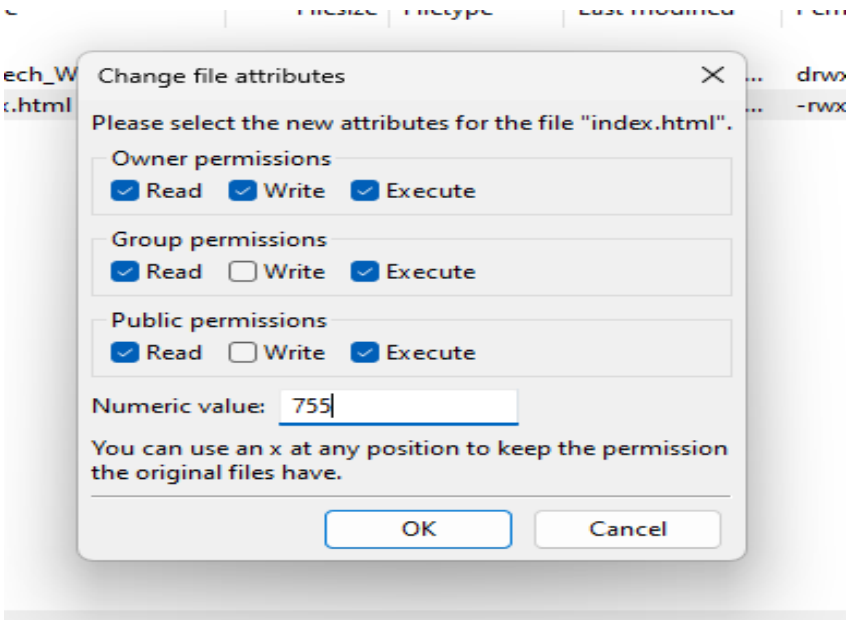
- In **Remote site** section, **expand** the **root**.
- Then **expand** the folder **var**, inside var **expand www**.
- Inside www, **expand html folder**.
- **Right click** on **Digitech web** and **click** on **Upload**.



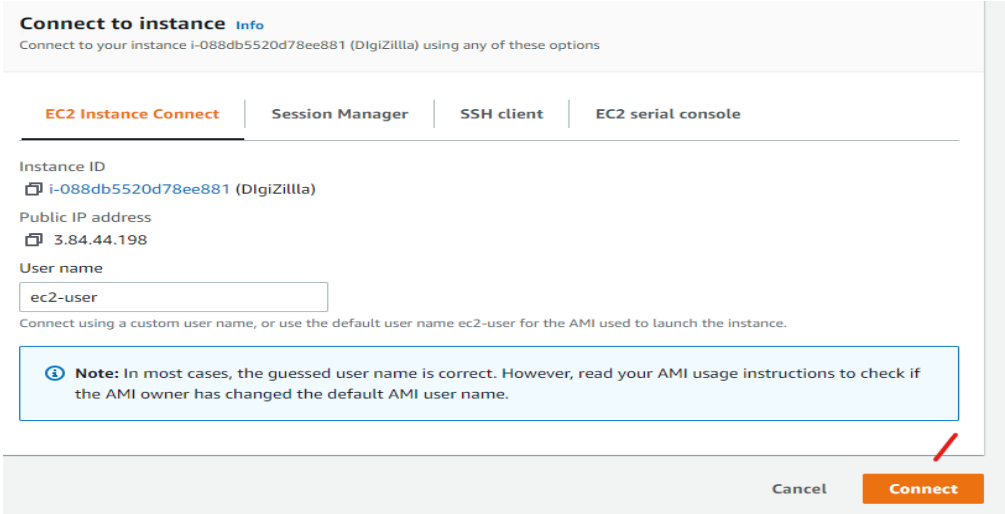
- It can be seen in screenshot below that **All files are transferred successful**.



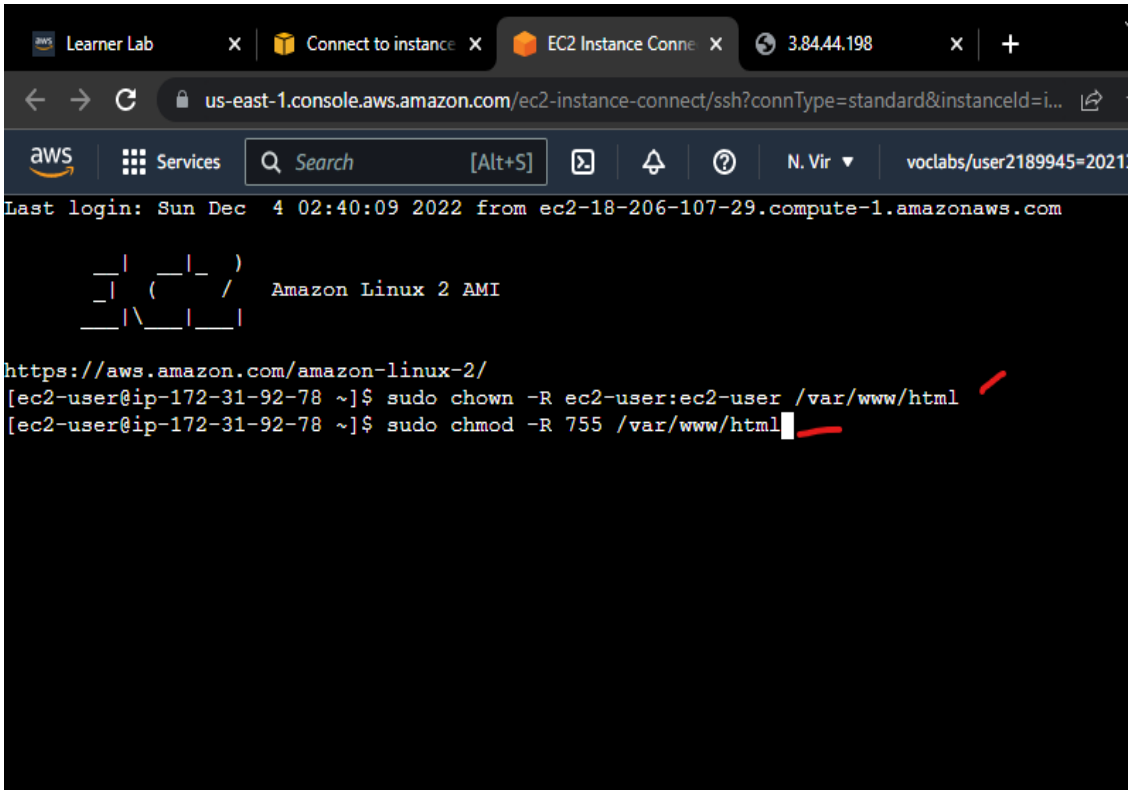
- **Right click** on **index.html** file in **remote site section** and **click** on **File permission**.
- Make sure the **Numerical value** is **755**.



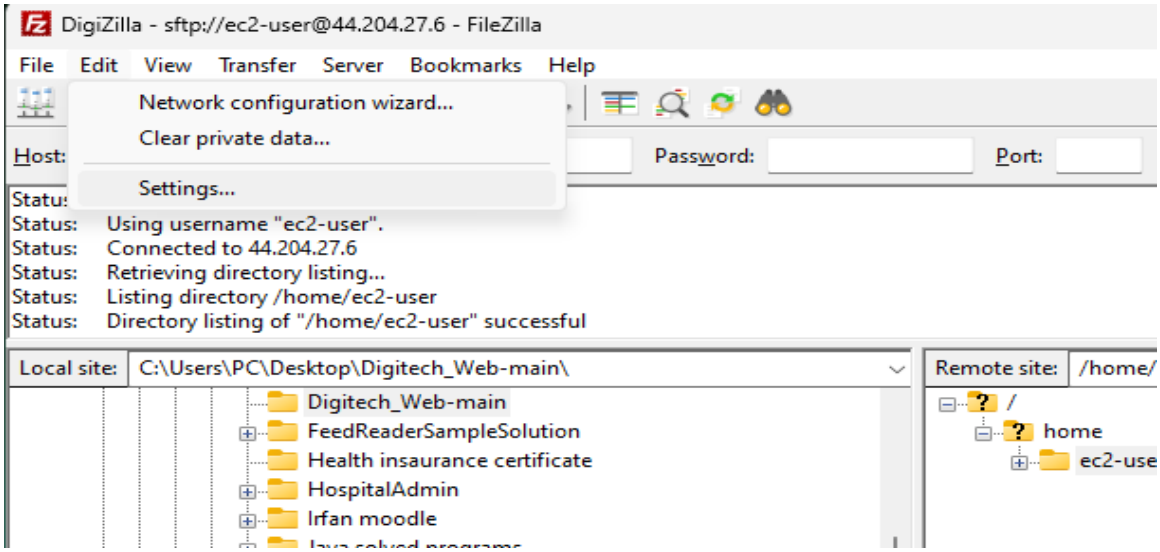
- Go **inside** the **DigiZilla** instance.
- Click **Connect**.



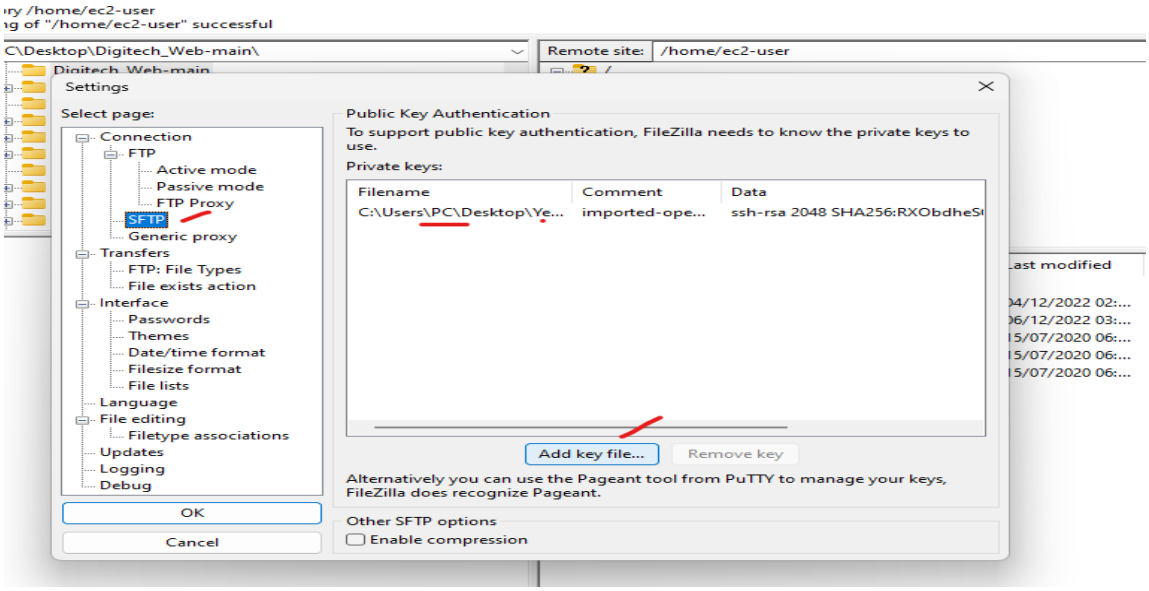
- I gave **Web permissions** which has been given on Moodle.
- **First**, I gave command as **sudo chown -R ec2-user:ec2-user /var/www/html**.
- **Second** command as **sudo chmod -R 755 /var/www/html**.



- I **copied** the **Public IP address** and **paste into Google** and **searched** it.
- **But** it **showed** me **only bootstrap script** while searching on google.
- Then**,** went to **Edit** on **FileZilla** and navigate **Settings.**



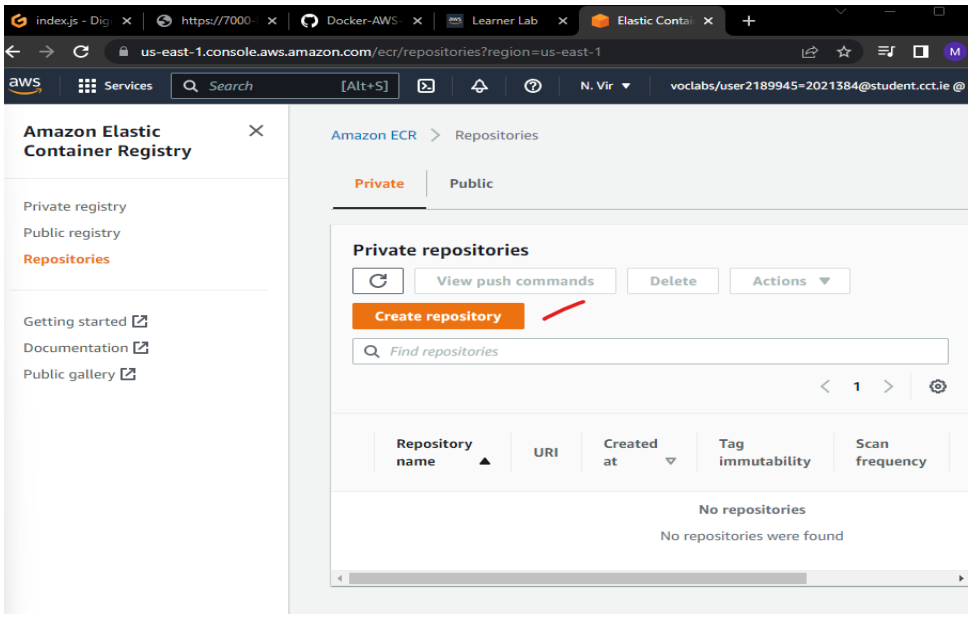
- Click **SFTP** option on left menu.
- Click on **Add key file** option.
- Navigate the **same key pair** from **device** which I **used** for **making** the **DigiZilla** instance.
- And **added it**.



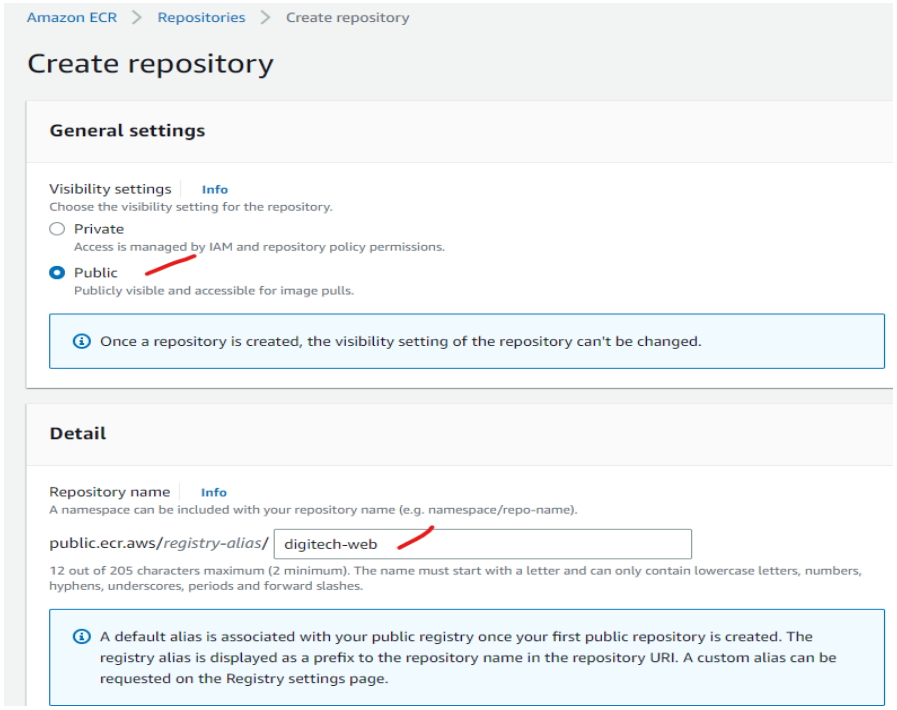
- I **uploaded** the **Digitech** web **again** and **followed** the **whole procedure again**.
- And it **can be seen** in the screenshot below that the **Digitech web** is **working successfully**.

Challenge Task 2:

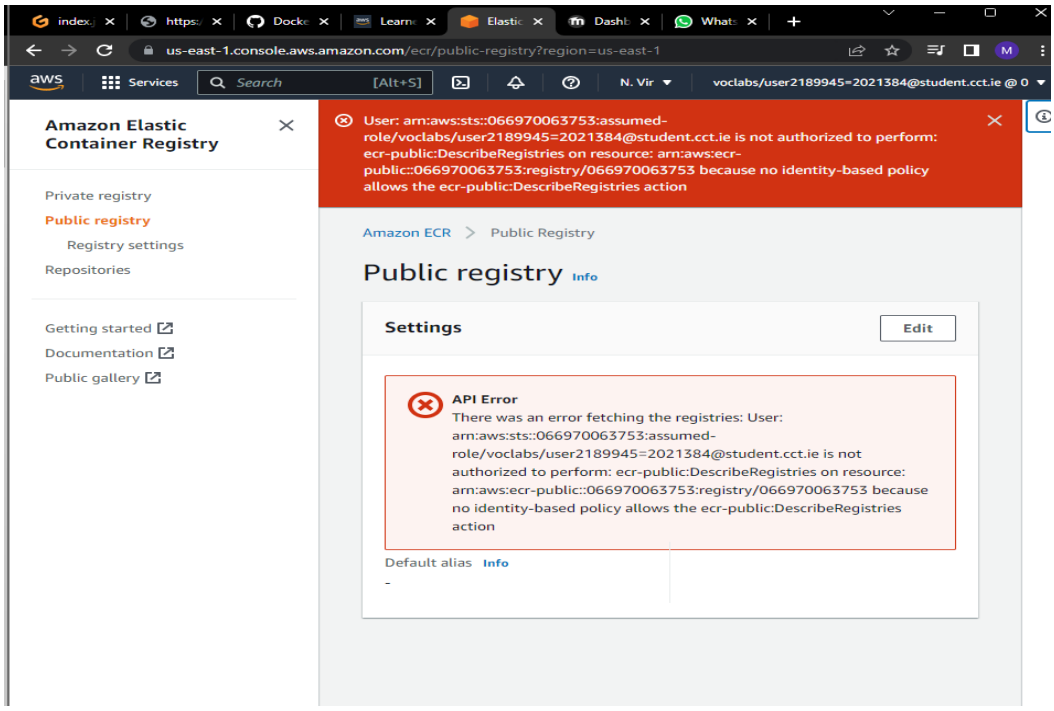
- I typed **ecs** for **Elastic container** registry in **search bar** of **services**.
- Click on **Repositories**.
- Click on **create repository**.



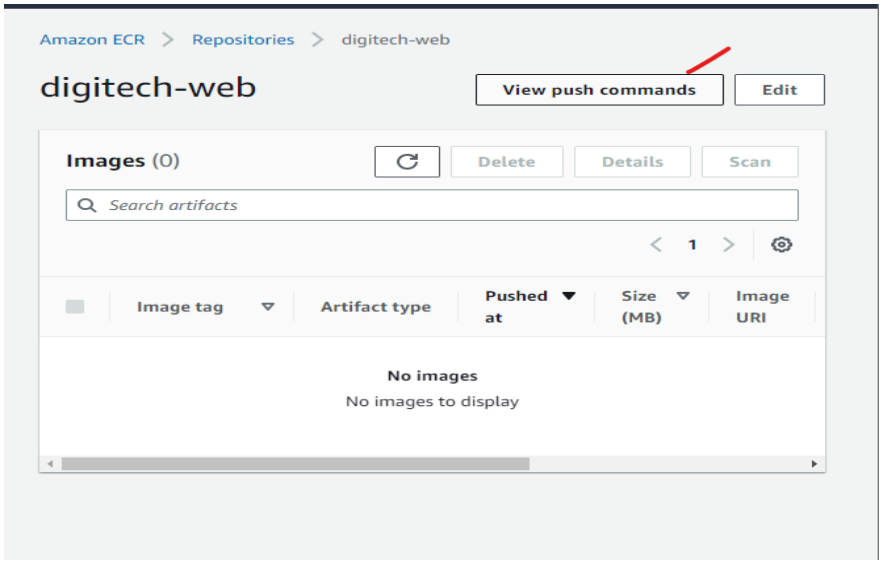
- I **made** it **Public**.
- I gave **repository name** as **digitech-web**.



- I **tried many times** but it **gave** me **error** while making it.
- **According to it** I **am not authorized** to **perform** it as **Public on** a **student account.**



- So, I **made** it as **private** with the **same name** as **digitech-web**.
- It was **created successfully**.
- Then **went inside** the digitech-web.
- Click on **View Push Commands**.



- I **needed** to **copy** all **these commands** and **run into** my **online visual studio where I created my Docker file.**
- **I tried very hard** and **install AWS tools** also **in** my **online visual studio**.
- **But it didn't work**.
- It **again gave** me **error as no access**.

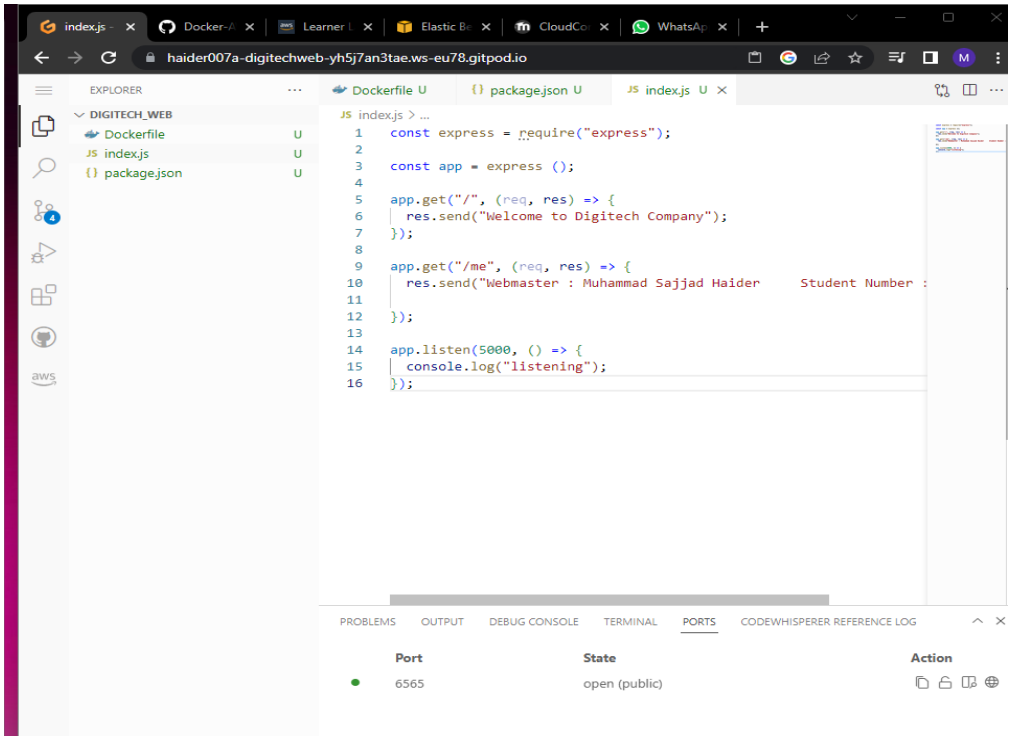
Make sure that you have the latest version of the AWS Tools for PowerShell and Docker installed. For more information, see Getting Started with Amazon ECR.

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see Registry Authentication.

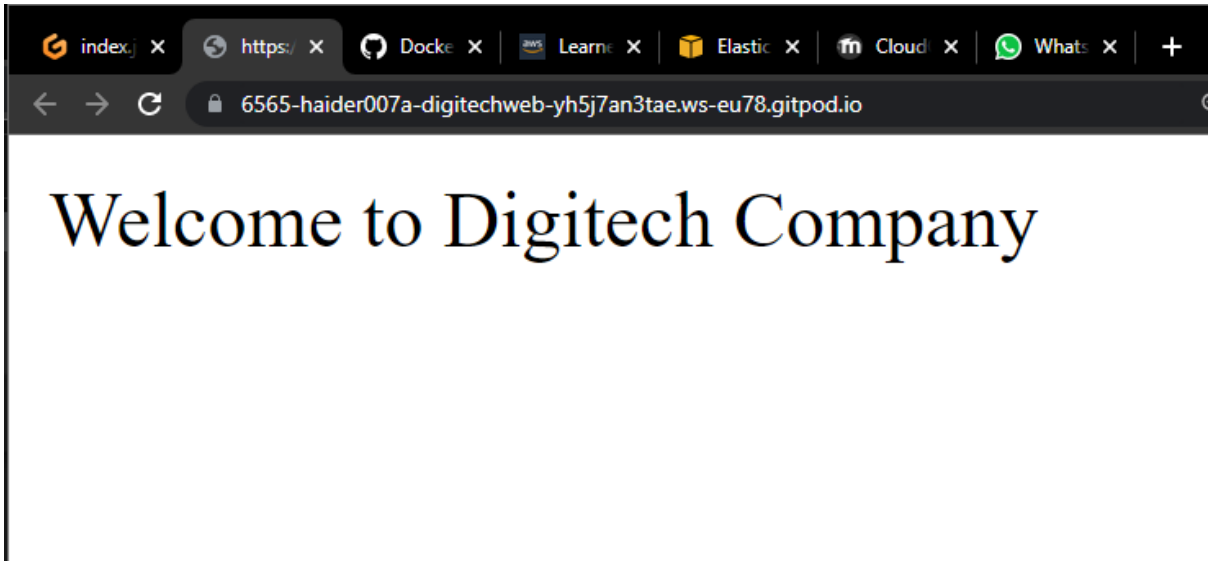
1. Retrieve an authentication token and authenticate your Docker client to your registry.
   Use AWS Tools for PowerShell:
   ```
   (Get-ECRLoginCommand).Password | docker login --username AWS --password-stdin 066970063753.dkr.ecr.
   east-1.amazonaws.com
   ```
2. Build your Docker image using the following command. For information on building a Docker file fr
   scratch see the instructions here. You can skip this step if your image is already built:
   ```
   docker build -t digitech-web .
   ```
3. After the build completes, tag your image so you can push the image to this repository:
   ```
   docker tag digitech-web:latest 066970063753.dkr.ecr.us-east-1.amazonaws.com/digitech-web:latest
   ```
4. Run the following command to push this image to your newly created AWS repository:
   ```
   docker push 066970063753.dkr.ecr.us-east-1.amazonaws.com/digitech-web:latest
   ```

Cl

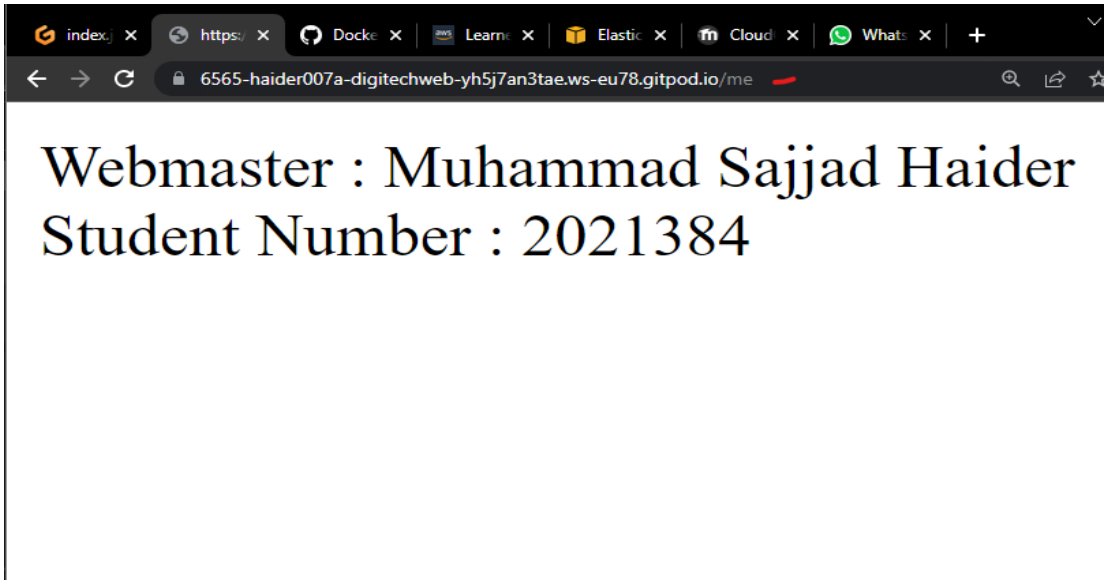
- Here is my **online visual studio** where I **created my docker file**.
- This **web app** is **running** on **port 6565 successfully**.



- It **can be seen** that **Docker file** is also **running properly** in **web browser**.



- If I **type /me next to link**, it **shows** me the **name and student number**.
- I **gave** that **shots as proof** that I **didn't have access** that's why I **couldn't complete.**
- Anyhow, **now I learnt it properly** and **know how to work** on it.

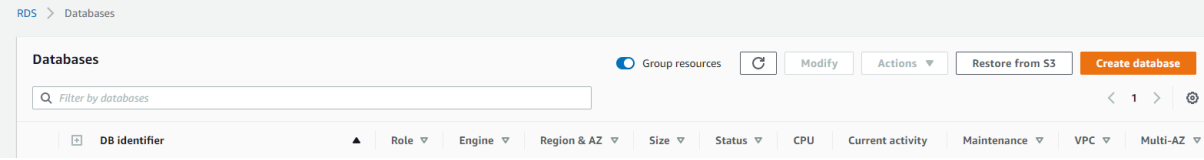


Reference

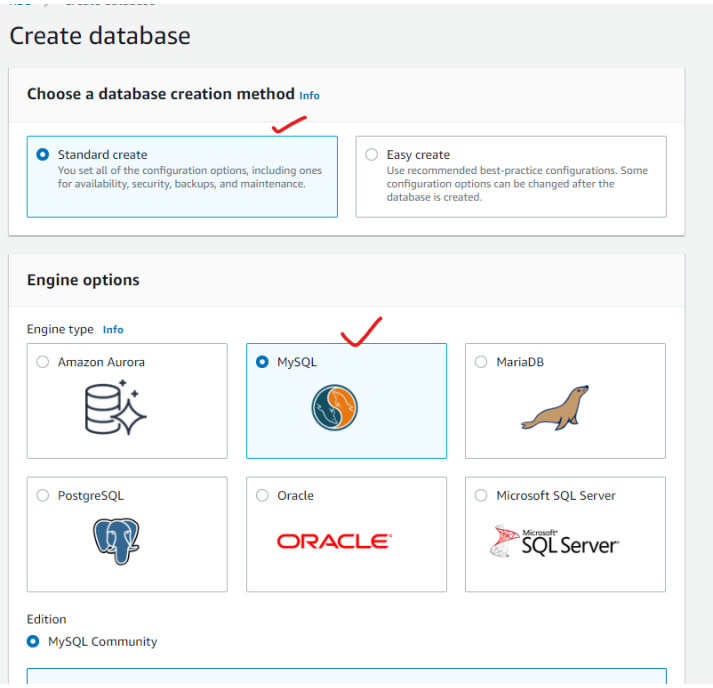
www.youtube.com. (n.d.). *How to Deploy a Docker App to AWS ECS*. [online] Available at: https://www.youtube.com/watch?v=YDNSItBN15w [Accessed 11 Dec. 2022].

Challenge Task 3:

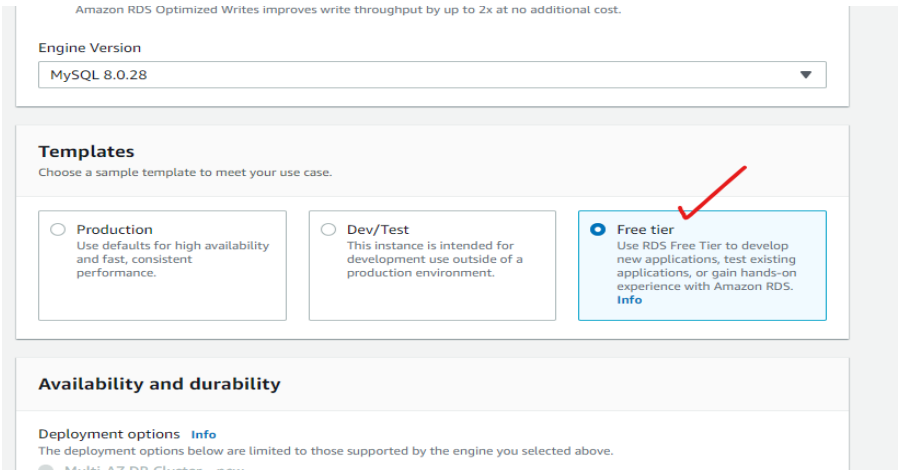
- Navigate **RDS** in **services** and select **Databases** from left menu.
- Click on **Create database**.



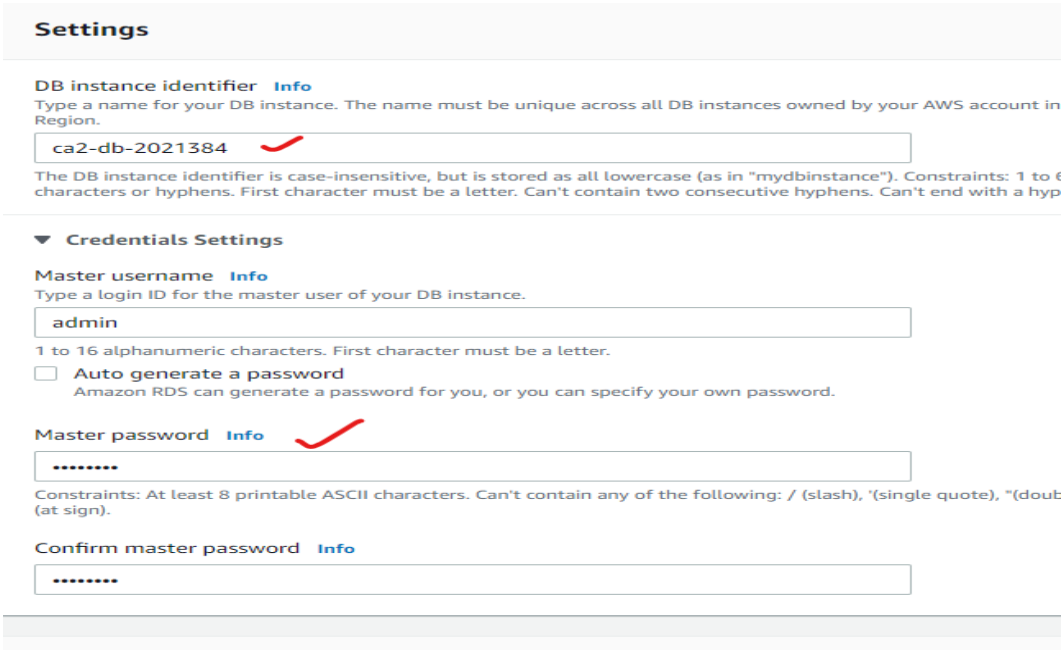
- Stick with **Standard create**.
- In **Engine options**, select **MySQL**.



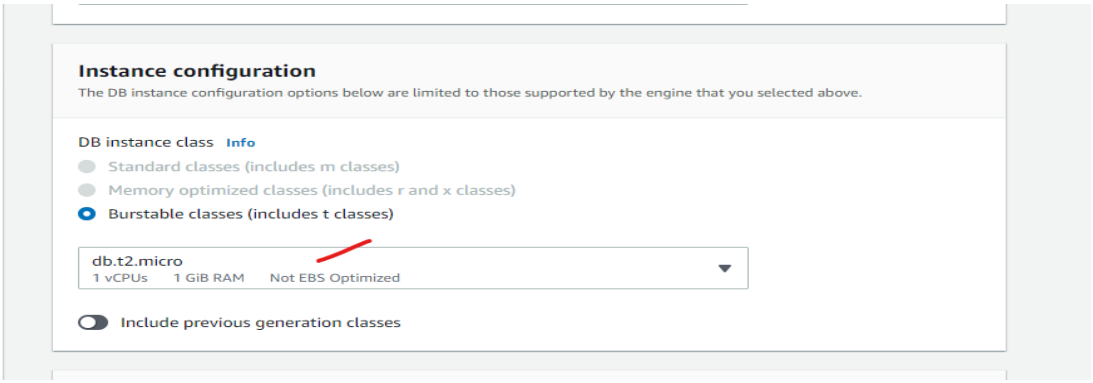
- In **Templates**, select **Free tier**.



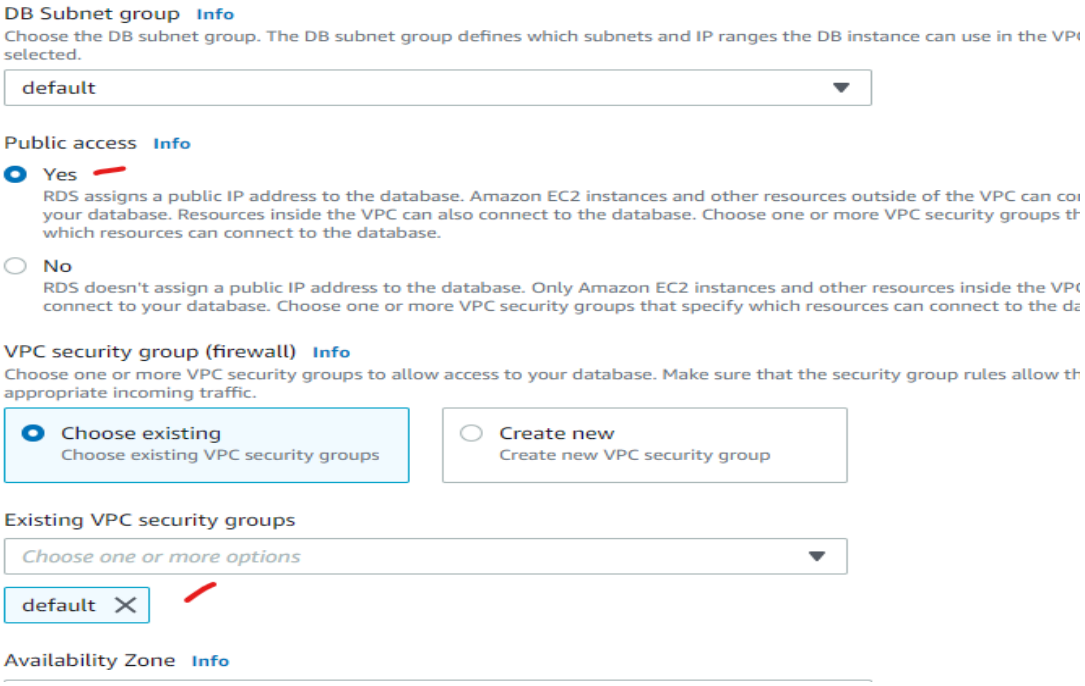
- In **Settings**, I gave **ca2-db-2021384** as **identifier**.
- I wrote **admin** in **Master username** and **root1234** in **Master password**.



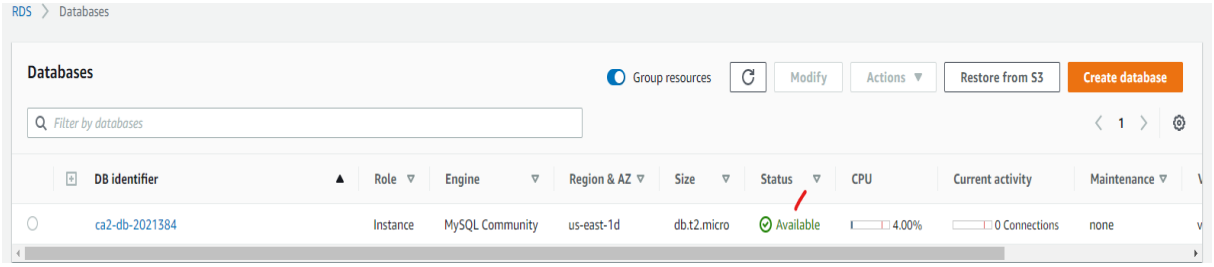
- I selected **db.t2.micro** in **DB instance class**.



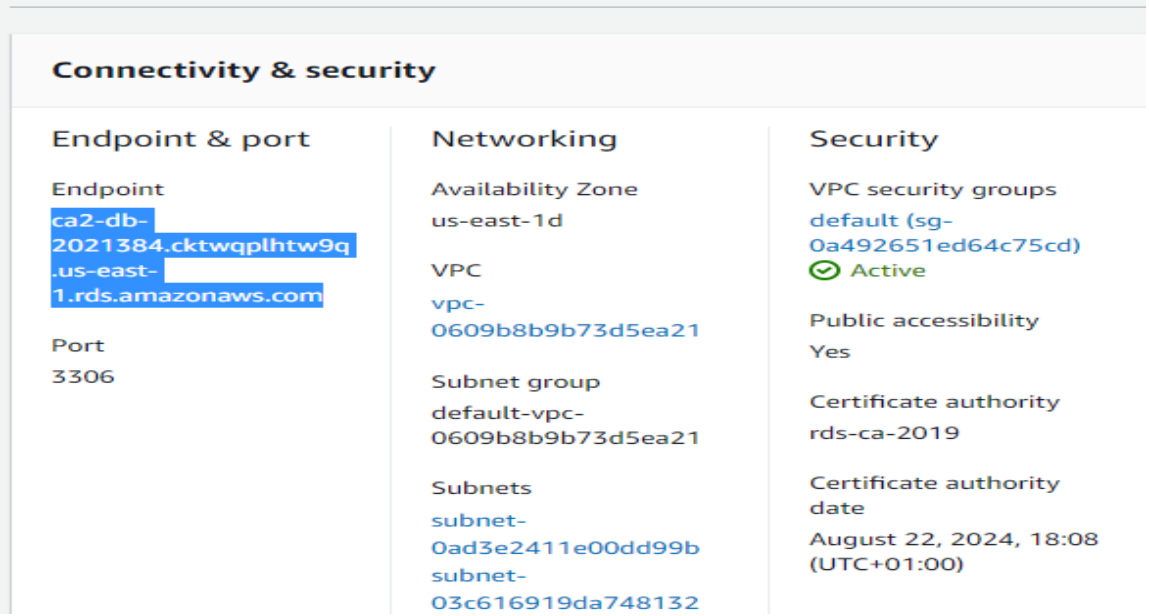
- In **Connectivity**, **Public access** should be **Yes**.
- I selected **default VPC** security group.
- Click **create**.



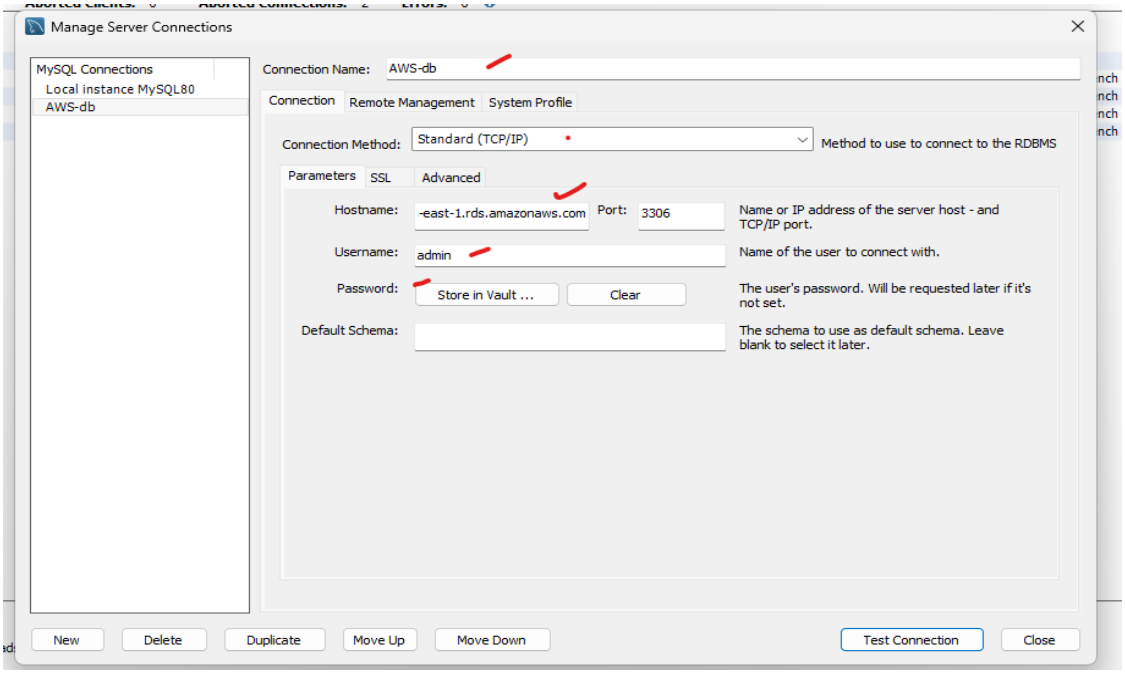
- It took time in **creating** and **back-up**, finally, **created** successfully.



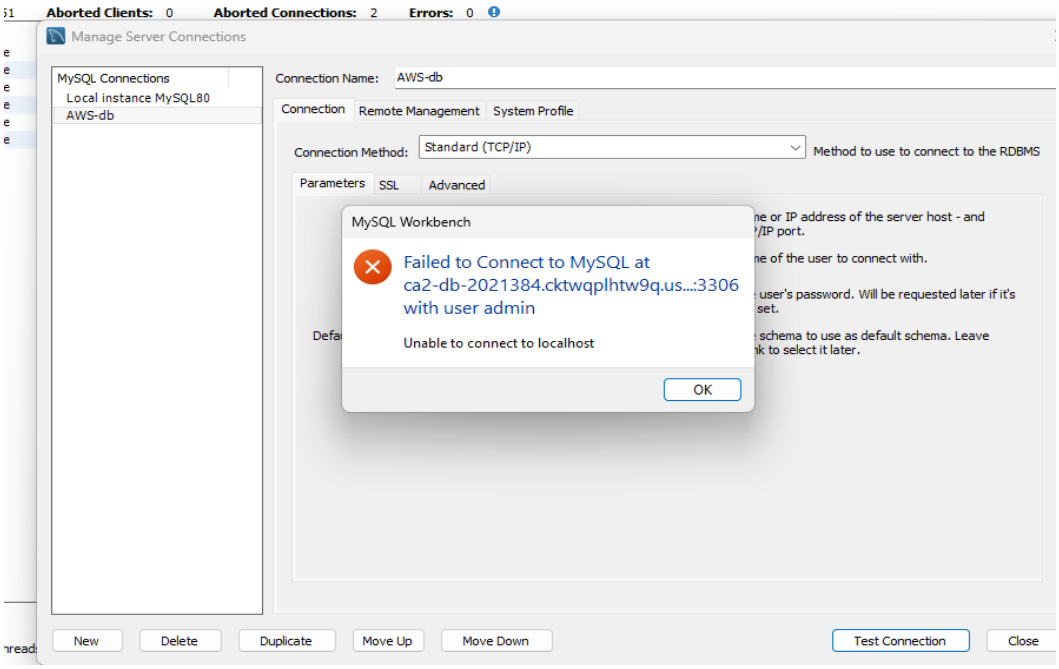
- Open the ca2-db-2021384.
- Copy the highlighted Endpoint.



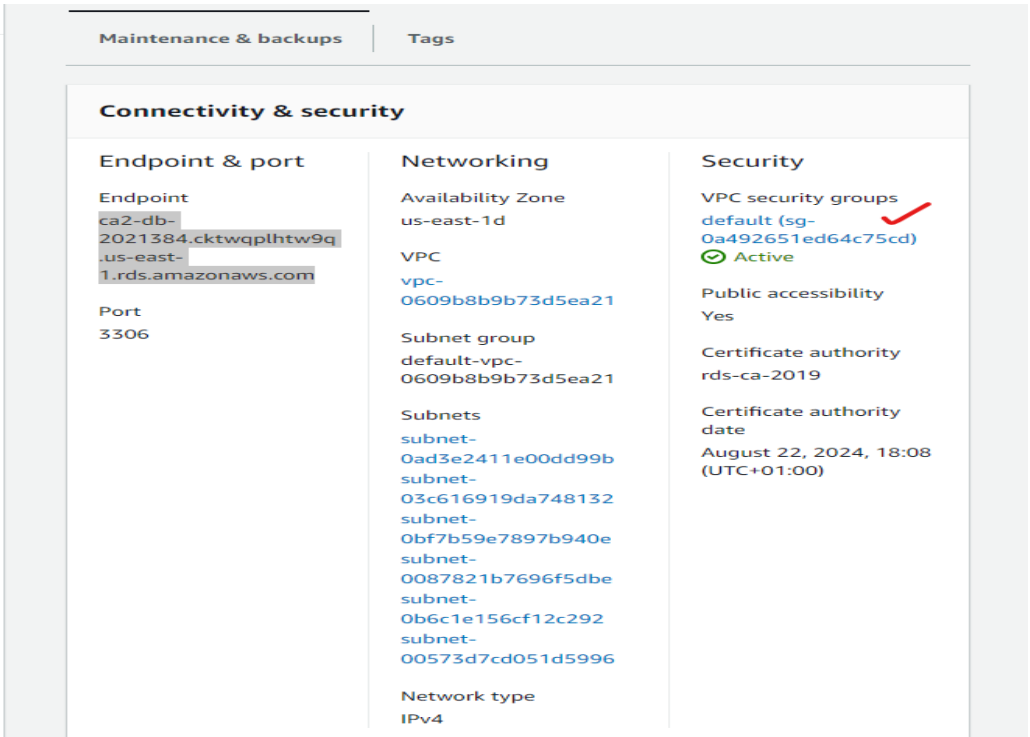
- I used **MySQL workbench** for **connecting** the database **ca2-db-2021384**.
- I gave **connection** name as **AWS-db** while **configuring** the **connection** in **MySQL**.
- **Paste** the copied **Endpoint** in **Hostname**.
- Wrote username **admin** and gave **password** which I have set **while making ca2-db-2021384** above.
- **Test connection**.



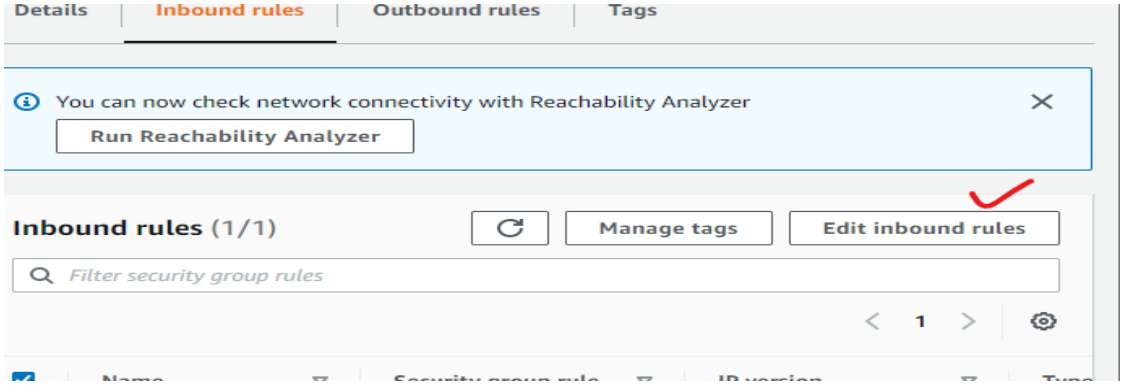
- It can be seen that **connection** is **failed**.



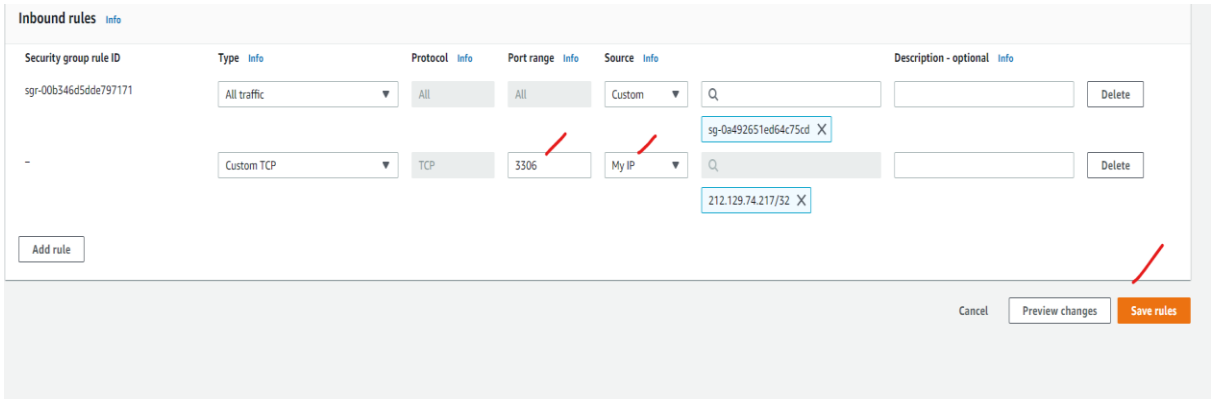
- For **troubleshooting**, **go to default VPC security group** in Security.



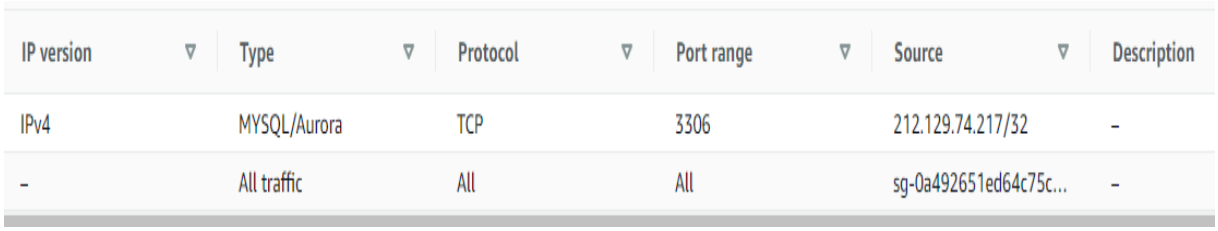
- Click **edit inbound rules**.



- **Add rule.**
- Gave the **port range 3306**.
- Set **Source** to **MyIP**.
- Click **Save rule**.

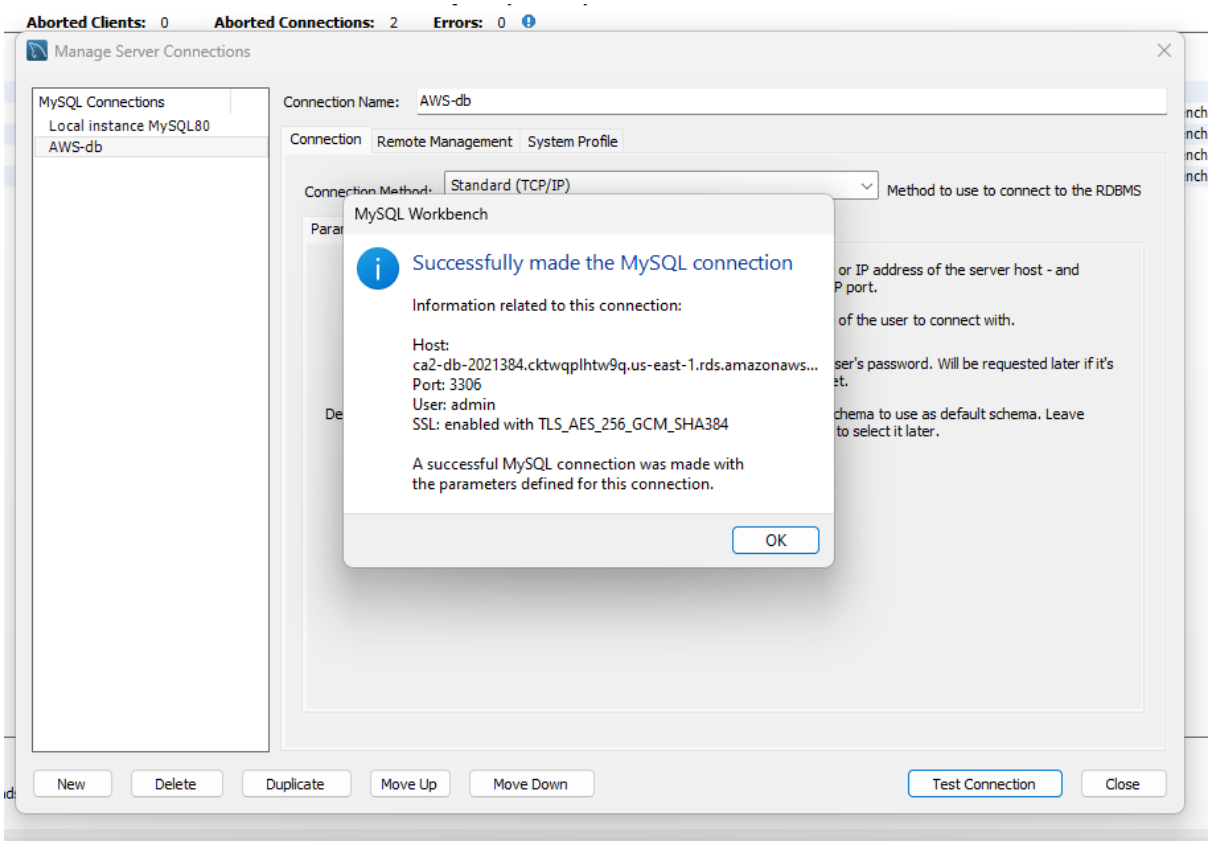


- **MYSQL/Aurora** rule has **updated**.



IP version	Type	Protocol	Port range	Source	Description
IPv4	MYSQL/Aurora	TCP	3306	212.129.74.217/32	-
-	All traffic	All	All	sg-0a492651ed64c75c...	-

- **Tested connection** again.
- It can be seen that the **MySQL connection** has **successfully made**.

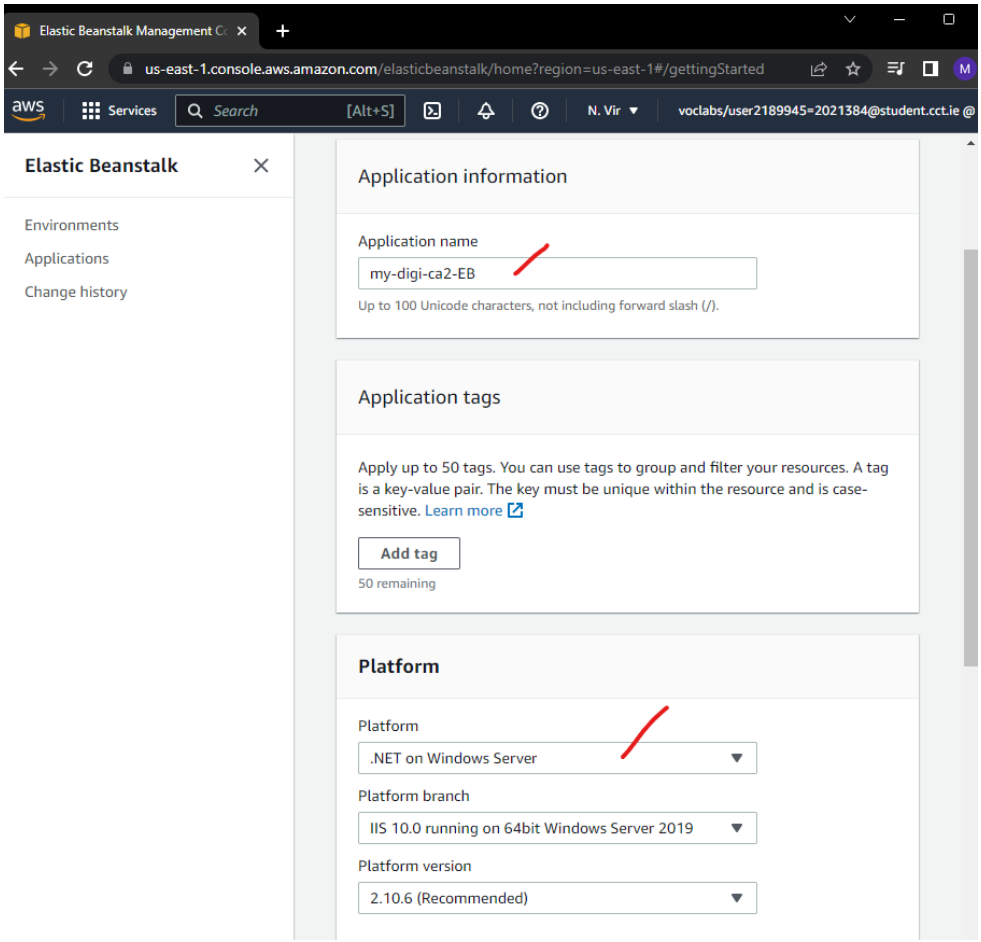


Reference

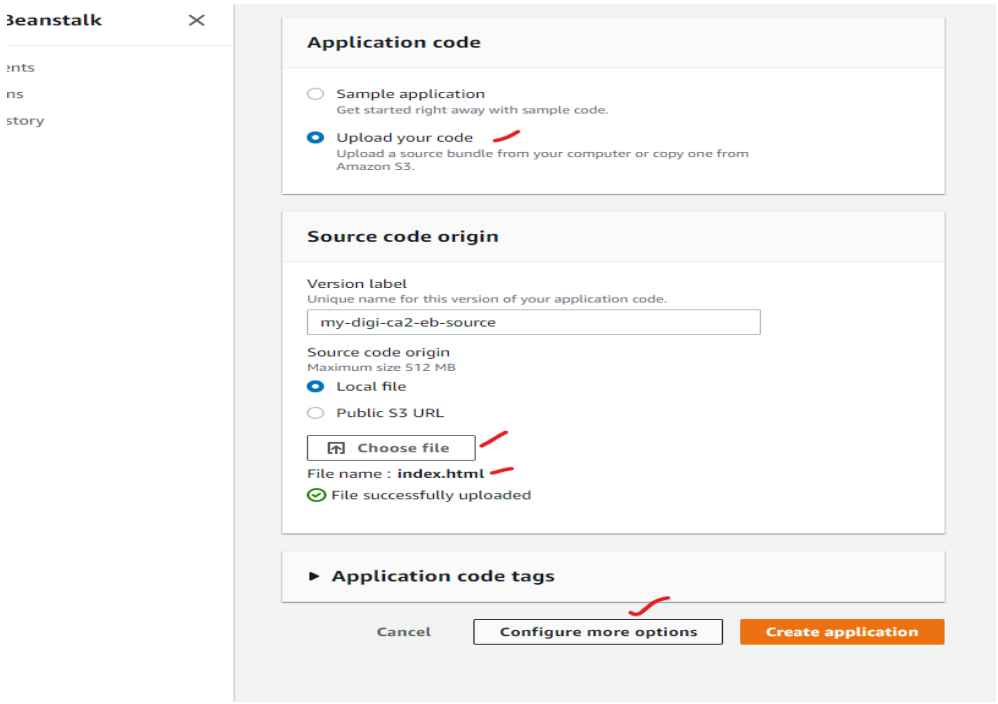
www.youtube.com. (n.d.). *Create a MySQL database on AWS!* [online] Available at: https://www.youtube.com/watch?v=v3jH1YxJqaY [Accessed 11 Dec. 2022].

Challenge Task 4:

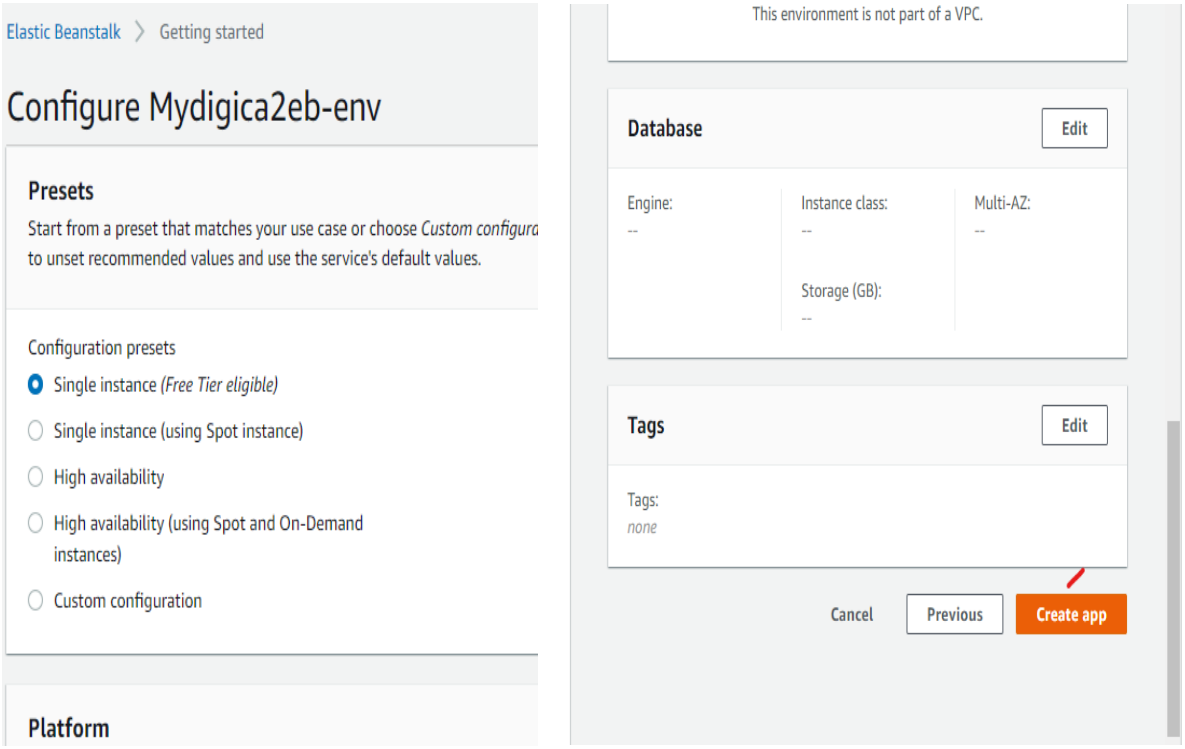
- Firstly, **Go to compute**.
- **Open Elastic beanstalk** and **click** on **create application**.
- I gave **name** of **application** as **my-digi-ca2-EB**.
- I chose **platform** as **.NET**.



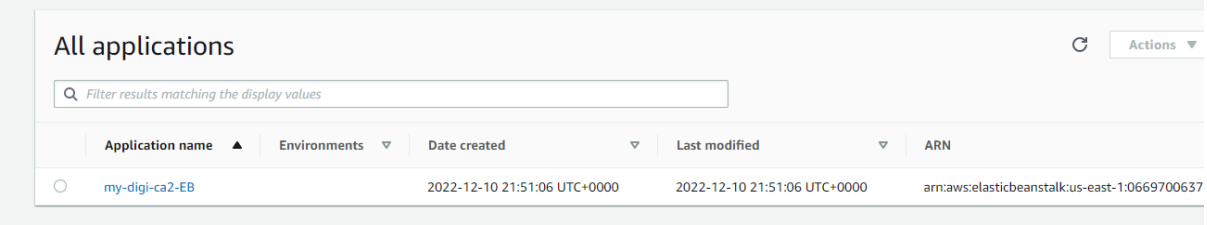
- In **application code**, **click** on **your code**.
- In **source code origin** click on **Local file**.
- Select **choose file option** and **browse** the **digitech web page**.
- **Index.html** page has **added.**
- Click on **Configure more options.**



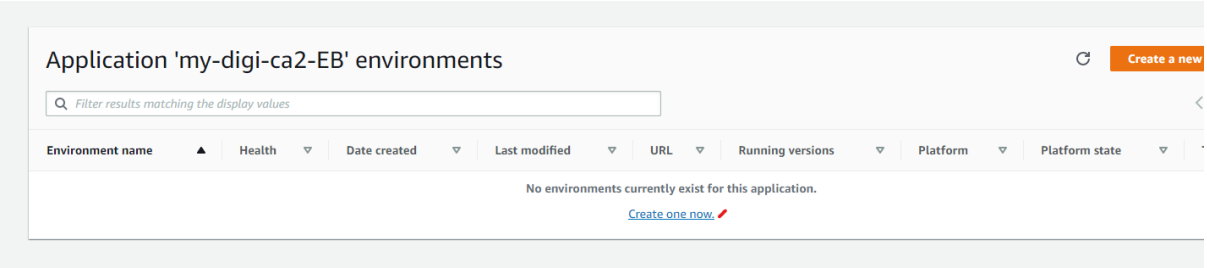
- **Stick** with **Single instance (Free tier)** in **Presets**.
- **Click** on **create app**.



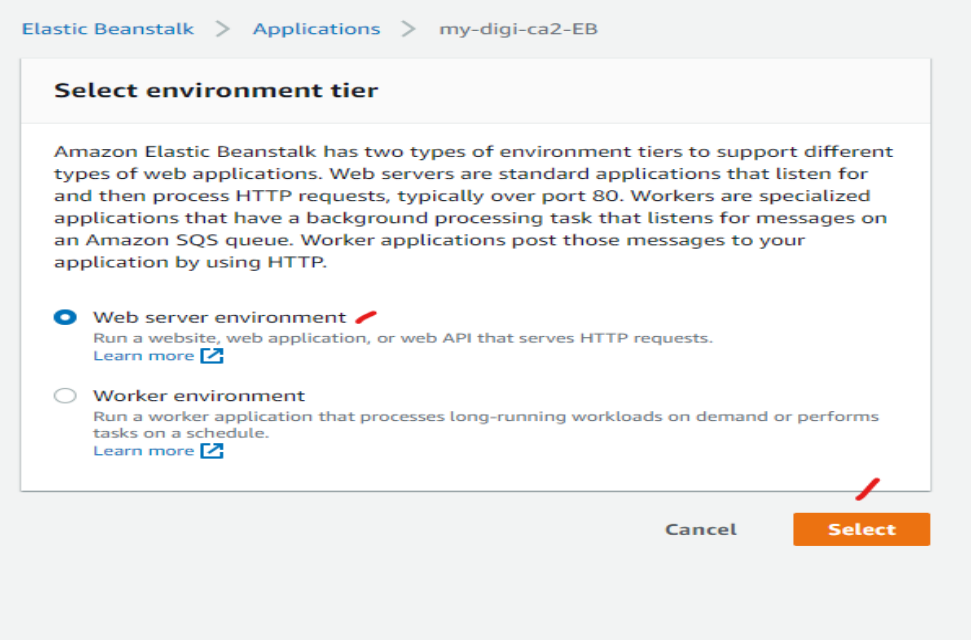
- It **can be seen** that **my-digi-ca2-EB** has **created**.



- Go **Inside** this application **my-digi-ca2-EB.**
- **Click** on **Create one now** for **creating** the **Environment**.

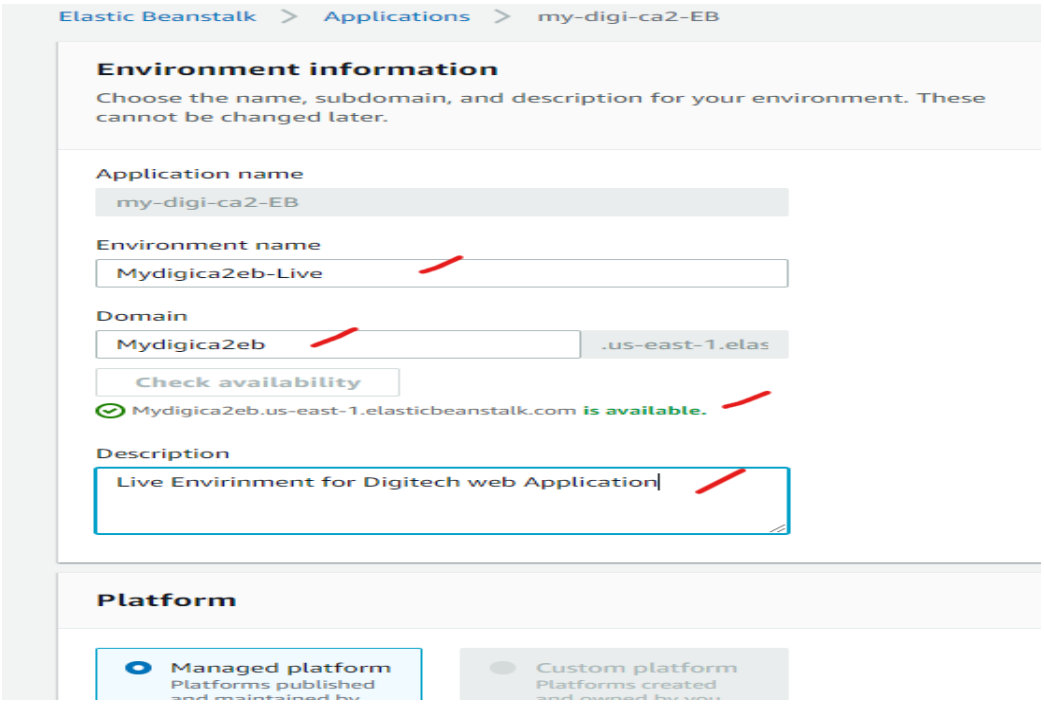


- In **select environment tier** section, **Stick** with **Web Server Environment**.
- **Click** on **Select**.

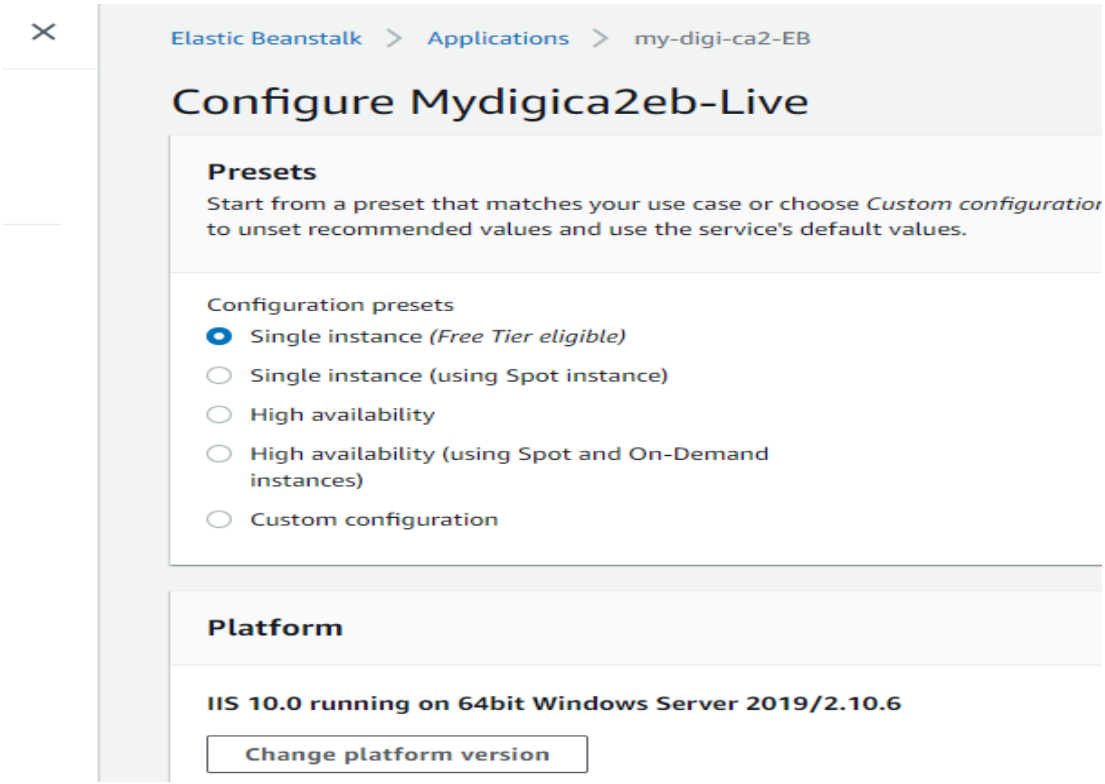


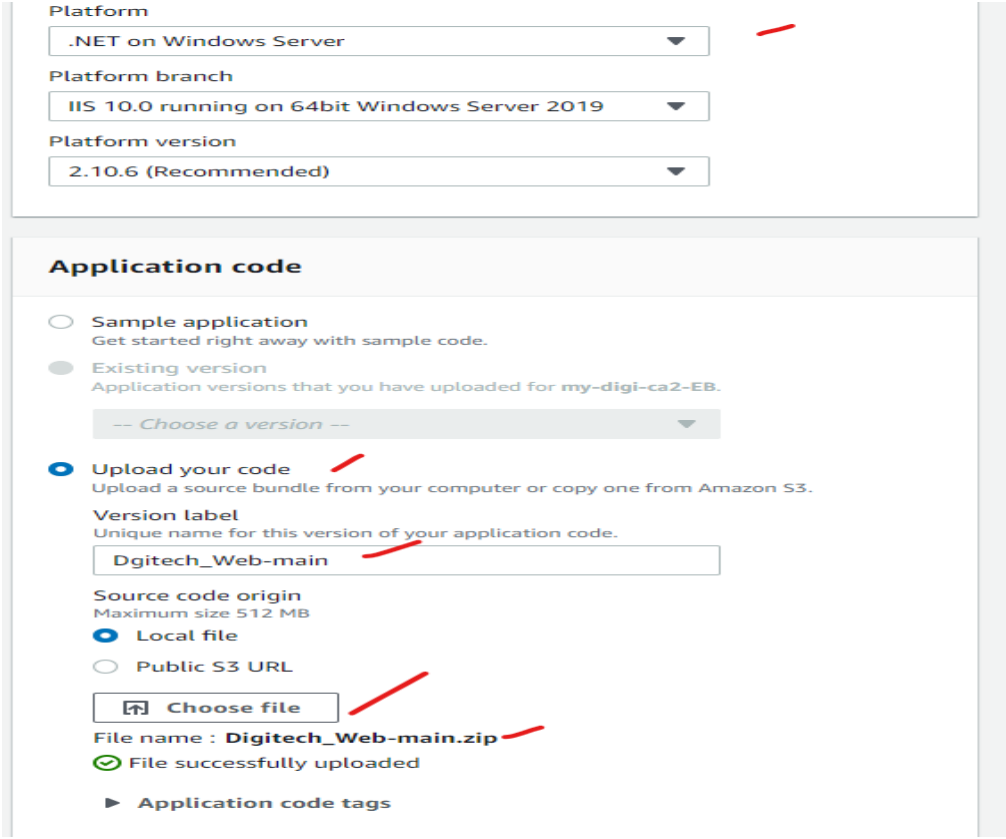
- I gave **Environment name** as **Mydigica2eb-Live.**

- **Domain** as **Mydigica2eb.**
- Click on **Configure more options.**

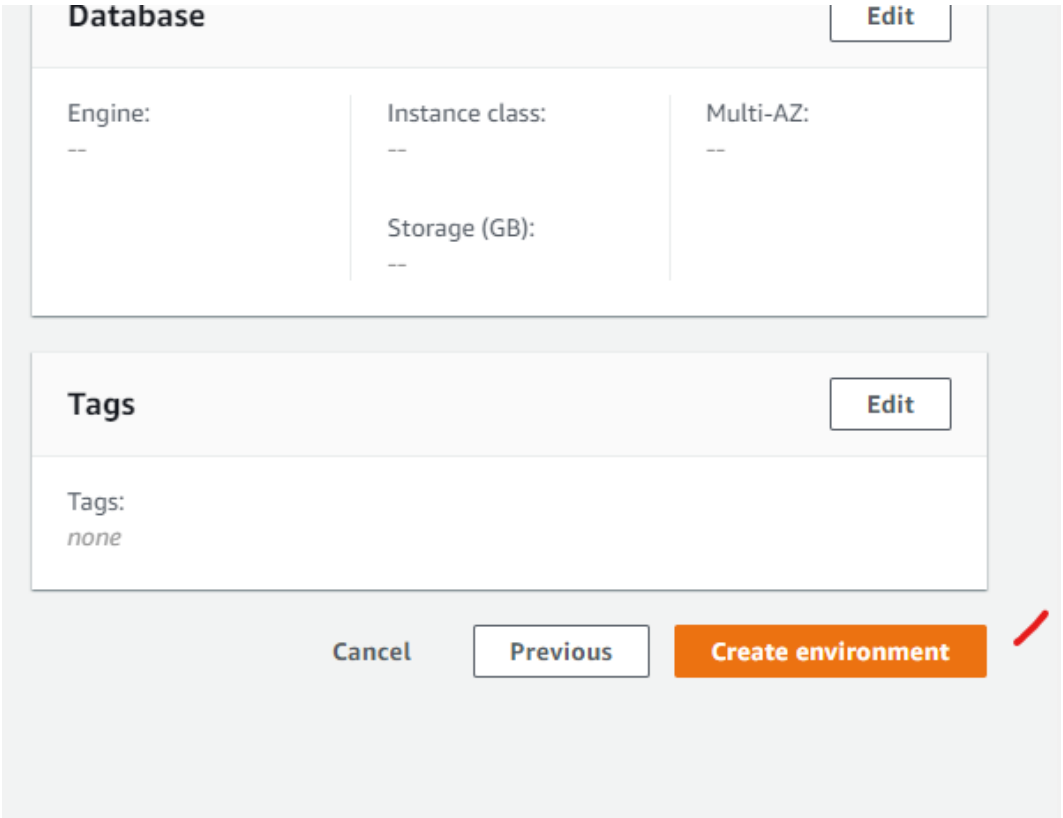


- I **followed the whole procedure** again which **I did** while **making application** above.





- **Click** on **create Environment**.



- I **tried many times** but it **gave** me **error** while making it.
- **According to it** I **am not authorized** to **perform** it as **Public on** a **student account.**
- I **gave** that **shots as proof** that I **didn't have access** that's why I **couldn't complete.**
- Anyhow, **now I learnt it properly** and **know how to work** on it.

Reference

www.youtube.com. (n.d.). *How To Create And Launch An Application With AWS Elastic Beanstalk*. [online] Available at: https://www.youtube.com/watch?v=Mh9Qx-K4UEo [Accessed 11 Dec. 2022].