Scenario :-

Application Running on Physical/Virtual Machines

Work load in your Datacentre.

Problem:-

Complex Management

Scale up & scale down

ost

Manual Process

Time Consuming

Solution:-

Cloud Setup –

Pay-As-You-go, IAAS, Flexibility, Ease of Infra Management

AWS Service:-

EC2 Instances

ELB –

Autoscaling

S3

Route 53

Objective:-

Flexible Infra

No Upfront Cost

Modernize Effectively

IAAC

Architecture of AWS Setup:-

EC2 Instances, ELB, Autoscaling, EFS/S3 for shared storage, Amazon Certificate Manager, Route 53

Users -> ELB -> Autoscaling Group -> Route 53

Flow of Execution:-

1. Login to AWS Account

2. Create Key Pair

3. Create Security Groups

4. Launch Instances with user data (Bash Scripts)

5. Update IP to name mapping in route 53

6. Build Application from source code

7. Upload to S3 bucket

8. Download Artifact to Tomcat Ec2 instance

9. Setup ELB with HTTPS[Cert from Amazon Certificate Manager]

10. Build Autoscaling Group for Tomcat Instances.

**1. Login to AWS Account**

Come to North Virginia Region

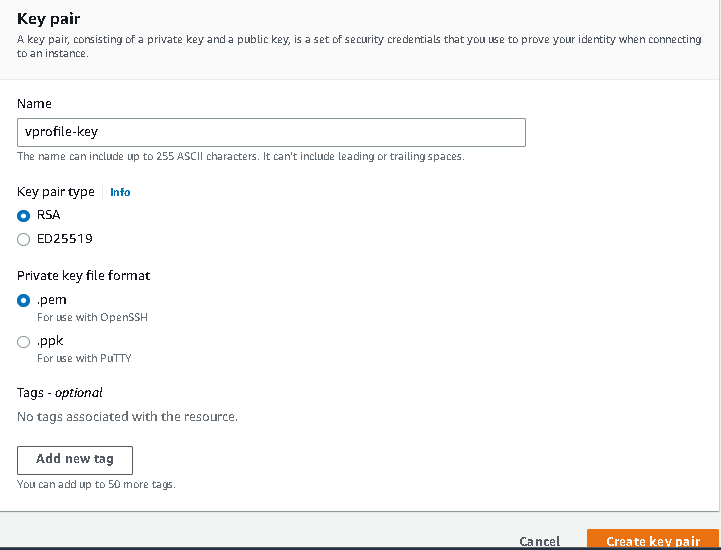
**2. Create Key Pair:-**

Goto Key pair > click create key pair >

Name : vprofile-key

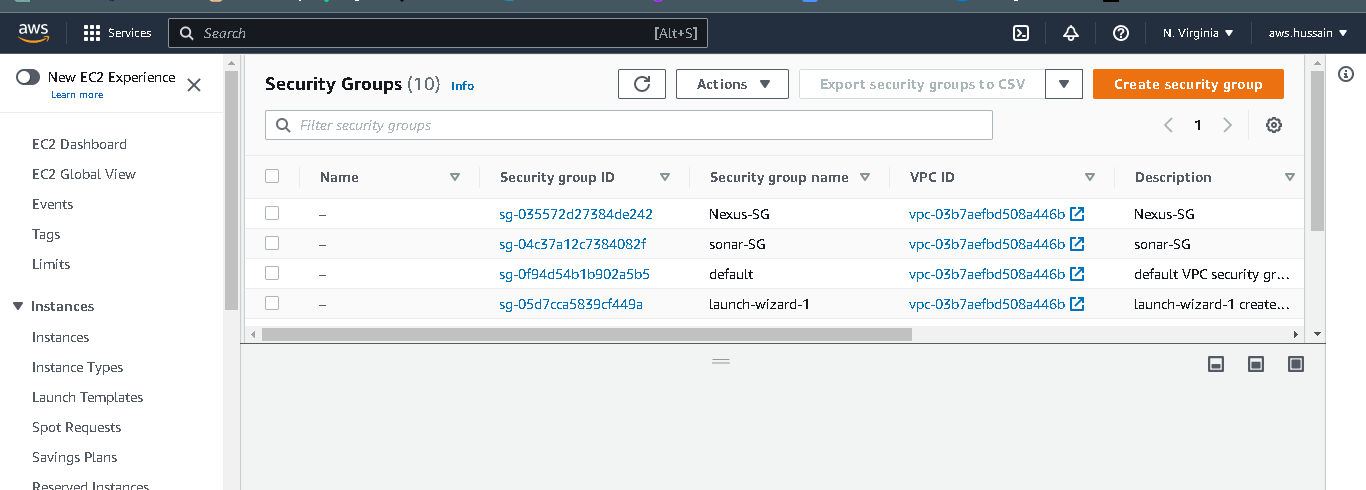
Key format : .pem

Click Create key Pair.



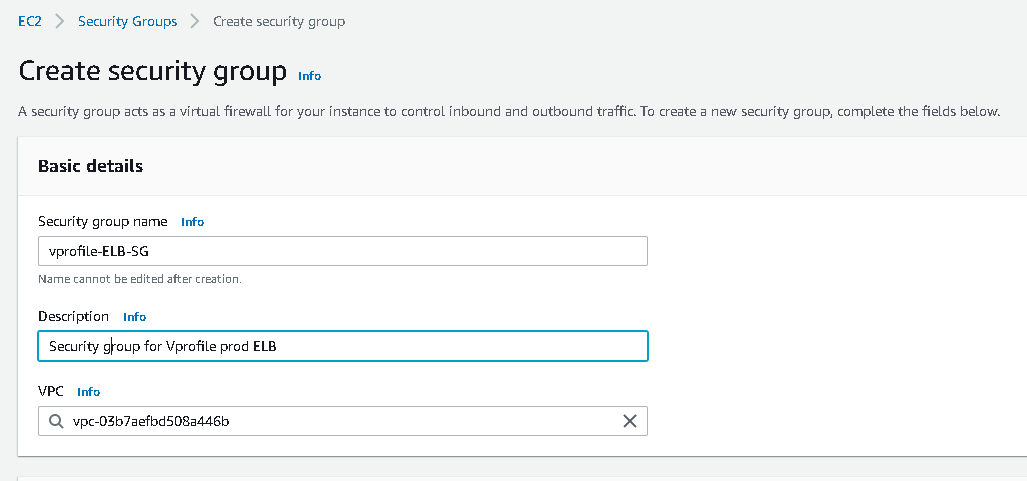
**3. Create Security Groups**

**Go to Security group > create security group for ELB >**



Click Create Security Group >

Name : vprofile-ELB-SG



Allow the Security group rules as follows:-

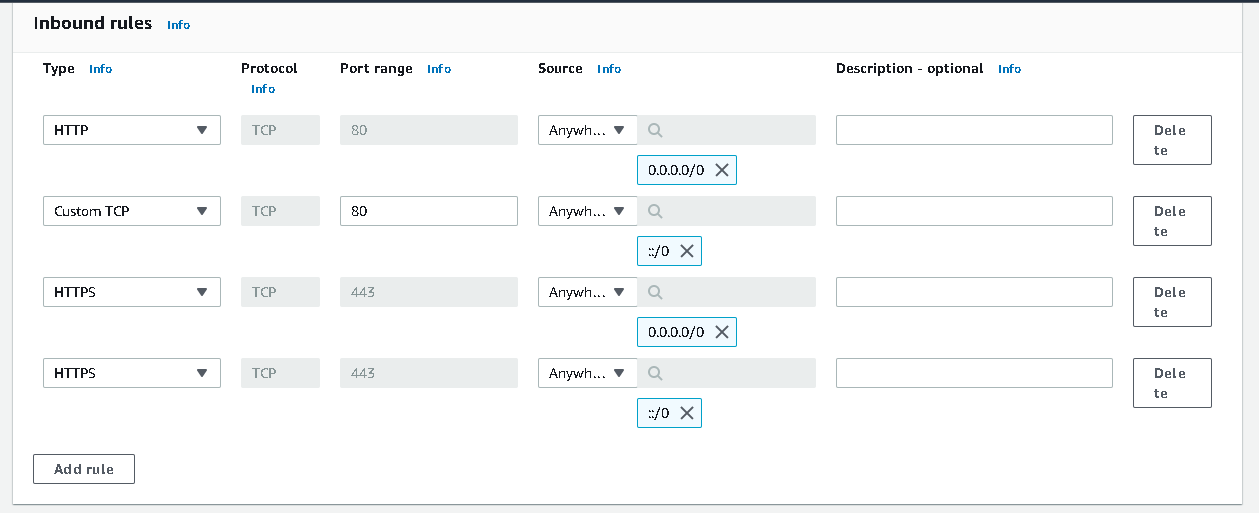
HTTP : 80 : Anywhere Ipv4

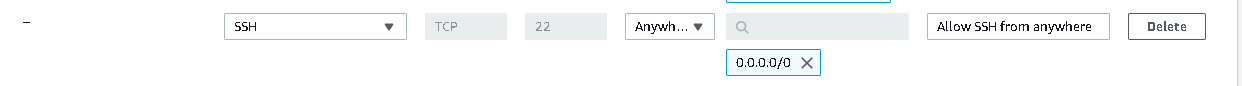
Custom TCp : 80 : anywhere Ipv6

HTTPS : 443 : Anywhere Ipv4

HTTPS : 443 : Anywhere Ipv6

Ssh : 80 : Anywhere from ipv4





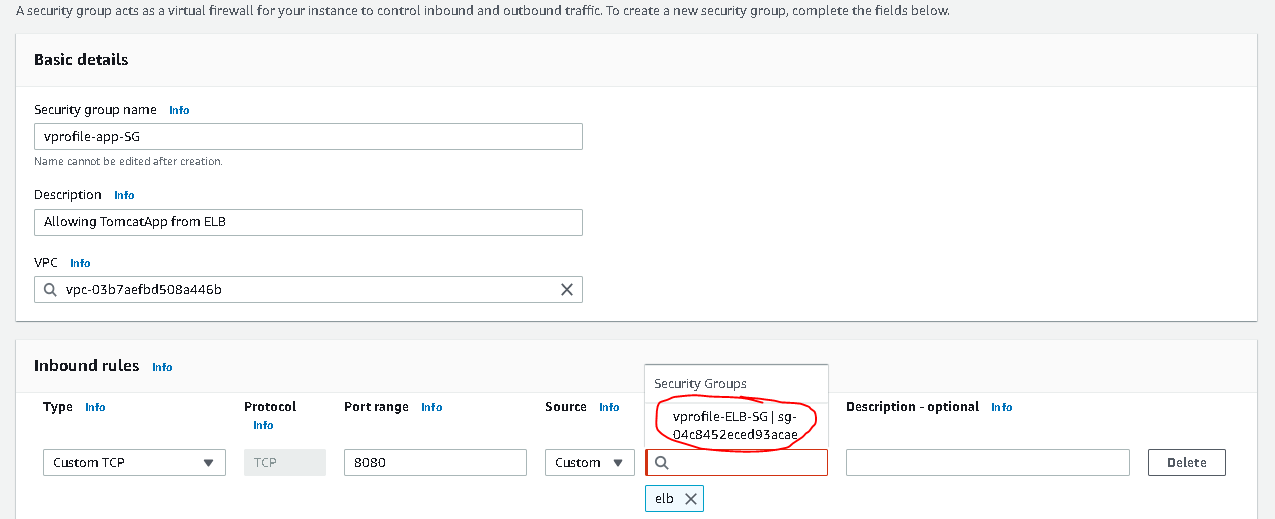
Next we are going to create security group for Tomcat App:-

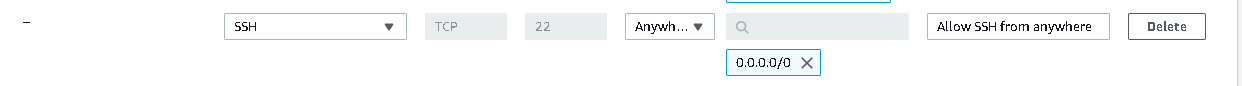
Click Create Security group > Name : vprofile-app-sg

Allow these rules :-

Cusom TCP : 8080 : Allow from ELB SG (type ELB in search bar, you will get the SG id)

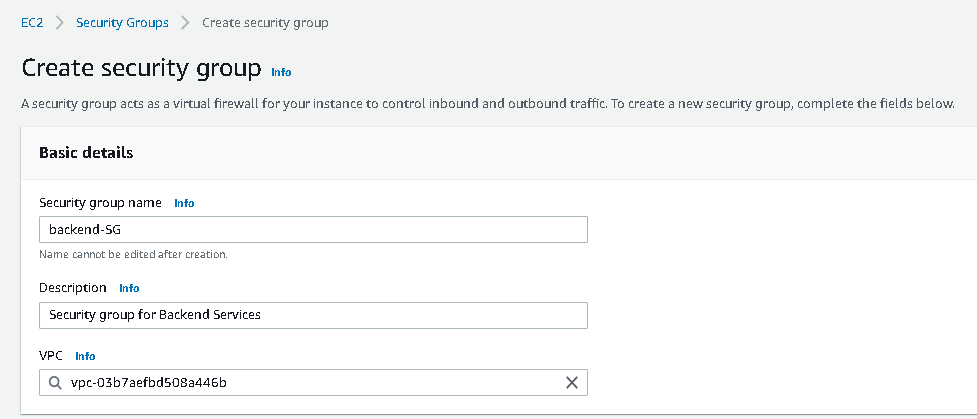
Ssh : 80 : Anywhere from ipv4





Now Create a another security group for backend services like RabbitMQ, MemcacheD, 7 MySQL

Click New Security Group > Name : backend services



In inbound Rules > Give these below rules

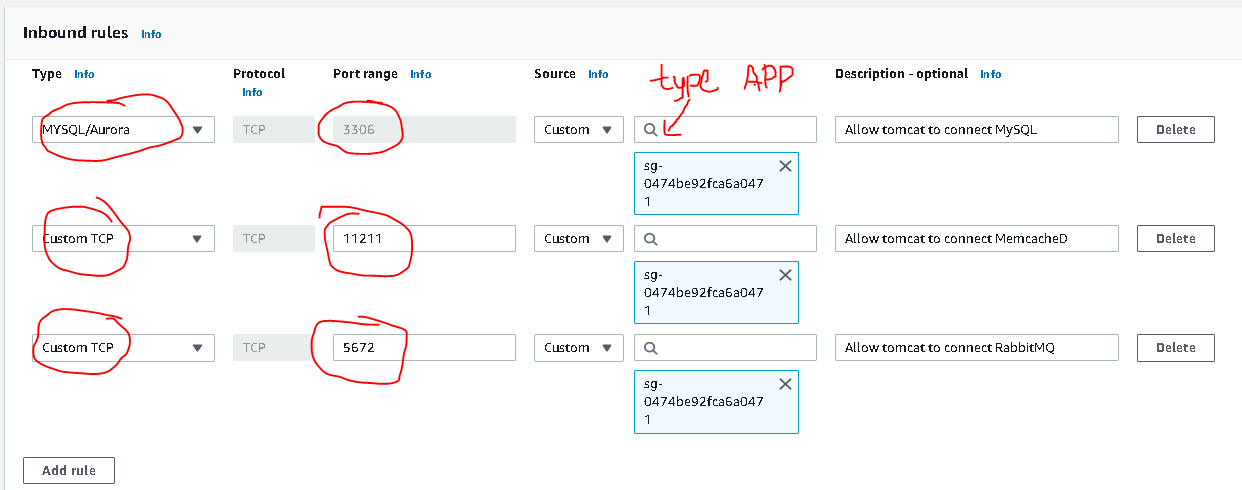
**Note :- MySQL Works on port 3306. MemcacheD works on port 11211. RabbitMQ works on 5672**

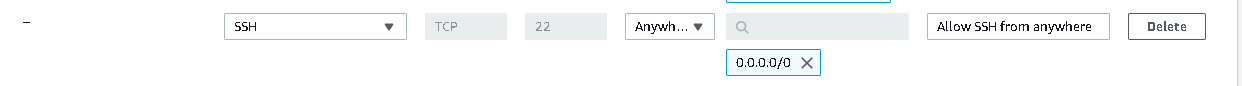
MySQL : 3306 : type app in search bar, you will get tomcat app SG, Then select it.

Custom TCP : 11211 : select tomcat app SG

Custom TCP : 5672 : selct tomcat app SG

Ssh : 80 : Anywhere from ipv4

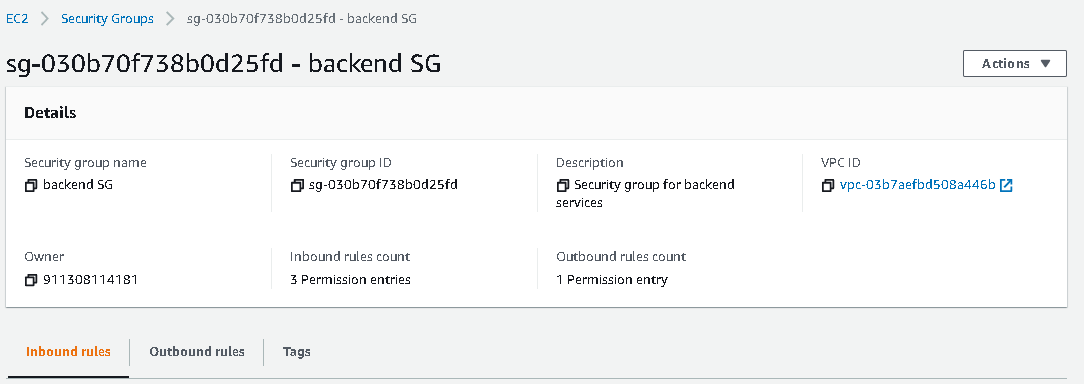




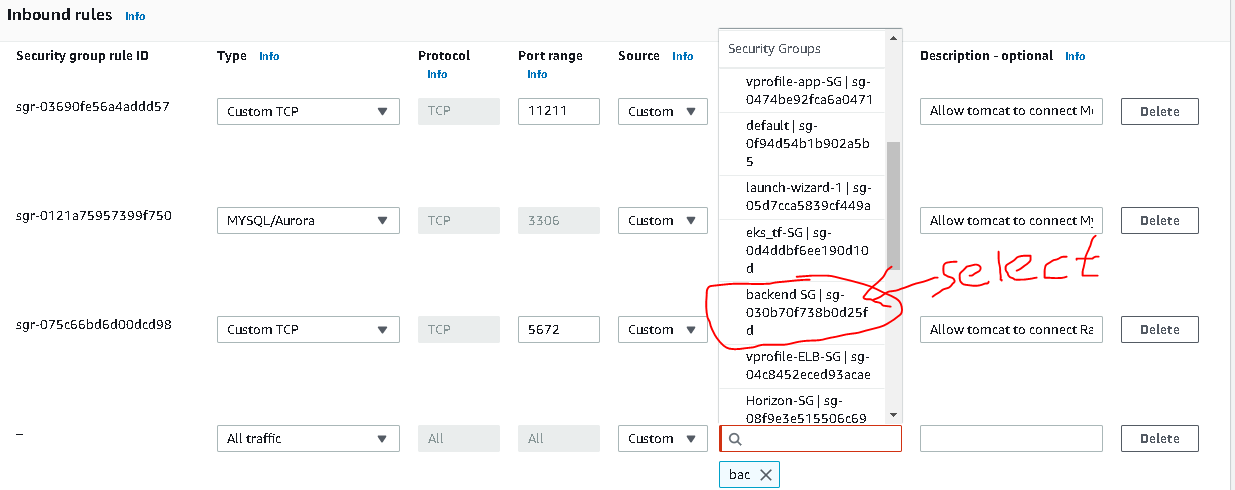
Click Create

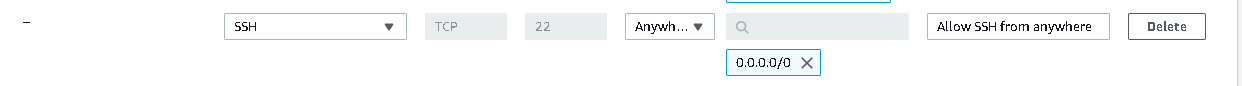
Again select our backend security group >

Add another rule which is backend SG itself, so that our backend services will communicate internally with each other.



Edit Inbound Rules >





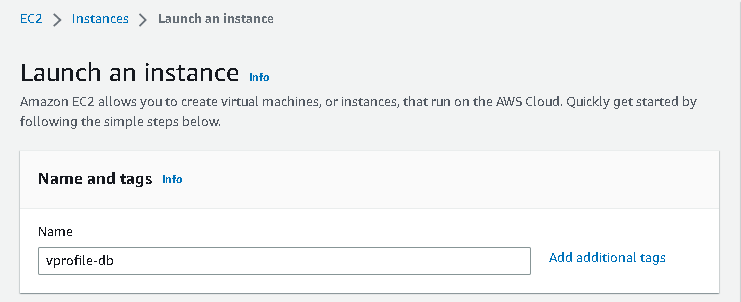
**4. Launch Instances with user data (Bash Scripts):-**

Now Launch Instance as mentioned below :-

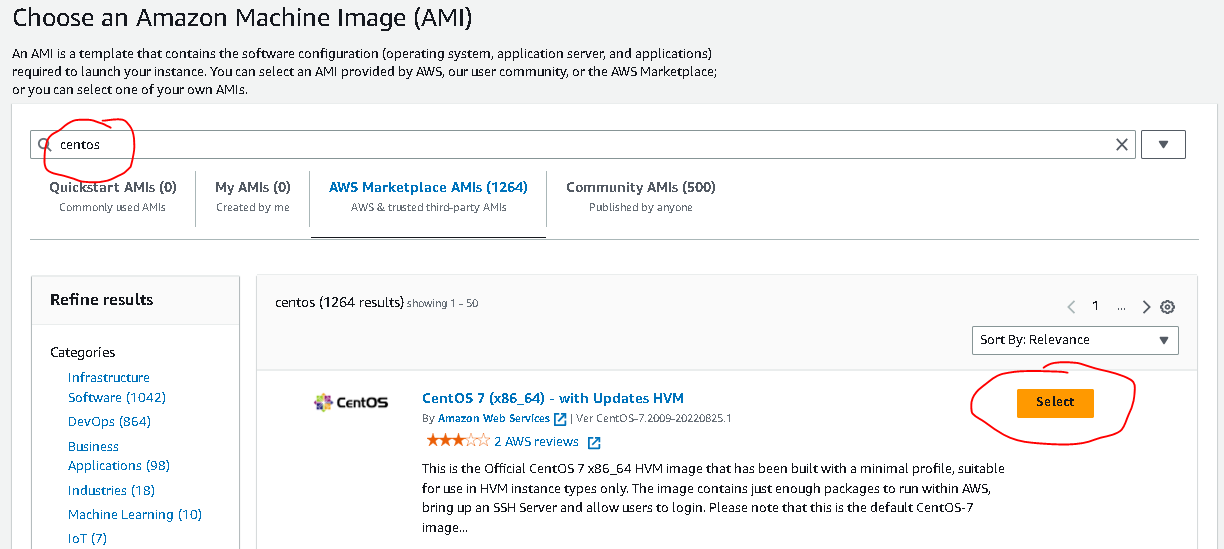
Launch CentOS Instance for MySQL

Launch CentOS Instance for MySQL

Launch an instance for Database-> Name : vprofile-db

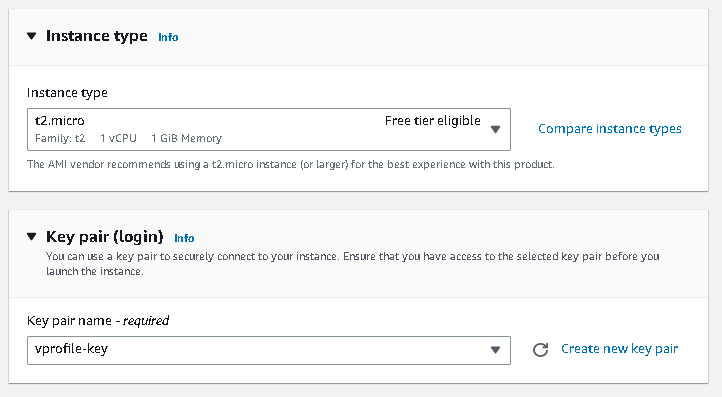


Choose AMI CentOS 7

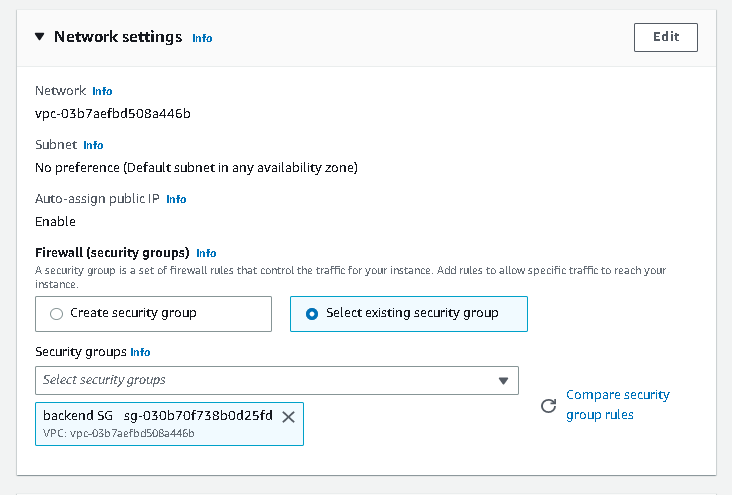


Select instance type : t2 micro

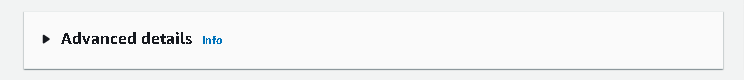
Select Key pair : vprofile-key



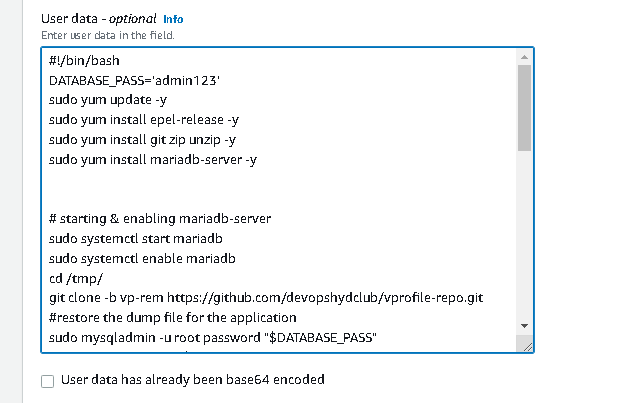
Select Existing Security Group : backend SG



Click on Advanced dateils > Go to User Data



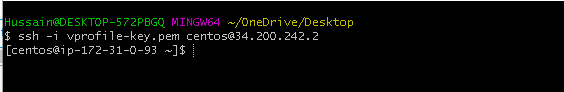
Paste the user data from the given link :



Then Click Launch Instance

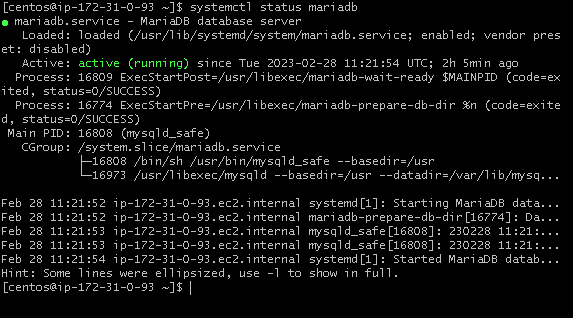
Do SSH to the launched instance >

ssh –i vprofile-key.pem centos@ipv4 public address



Now, check the status of MySQL whether it is active or not:-

systemctl status mariadb

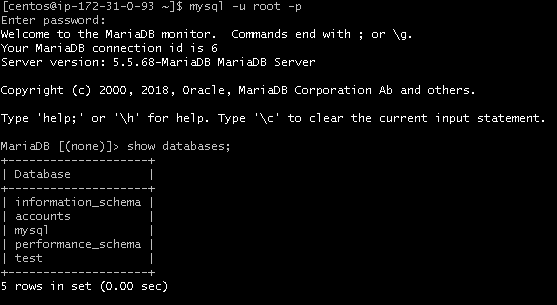


Validate the Database ->

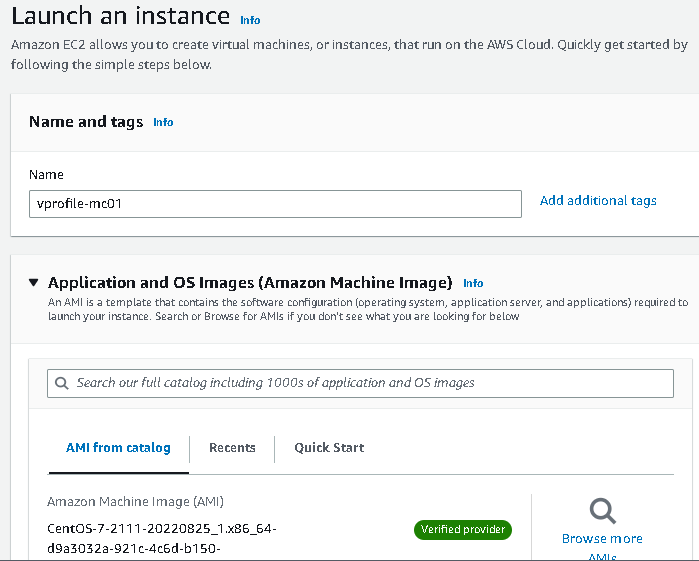
Mysql –u root –p

It will ask password -> Password : admin

show databases;

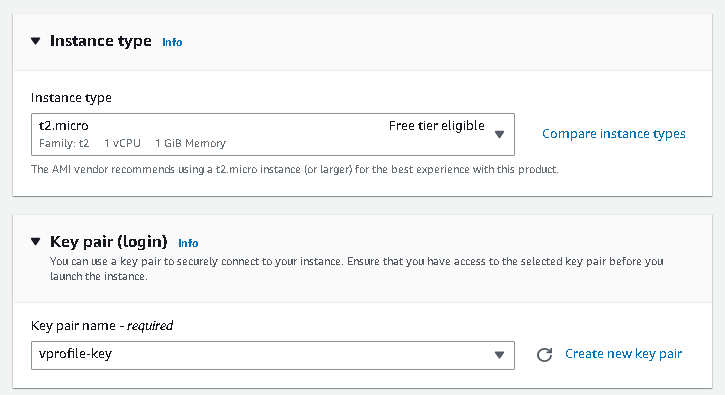


Now Go to AWS EC2 console > launch another instance for MemcacheD with CentOs

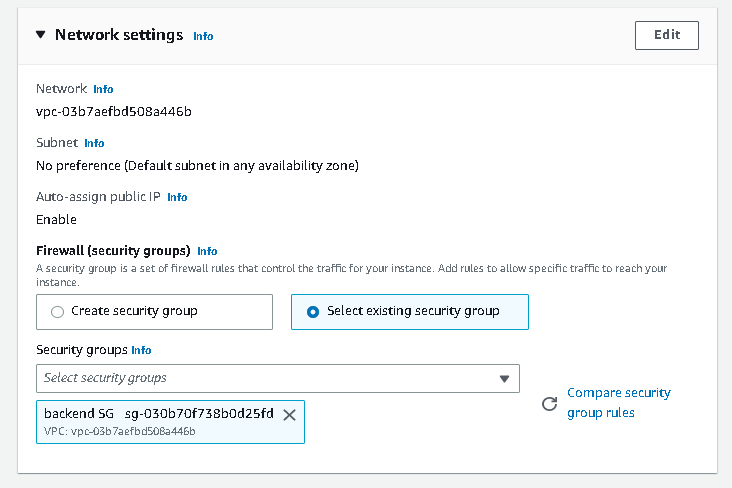


Instance Type : t2 micro

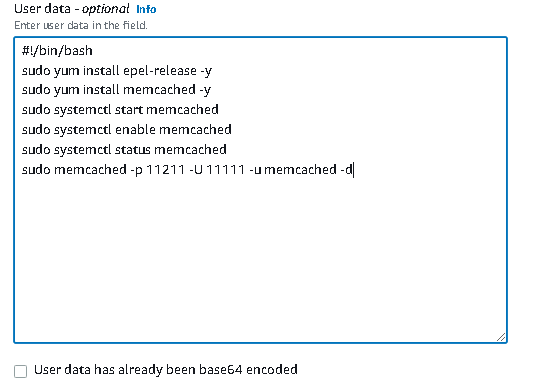
Key Pair : vprofile-key



Security Group: backend SG



In Userdata, copy the script from given link & paste in the userdata



**Click Launch Instance**

Now, do ssh :-

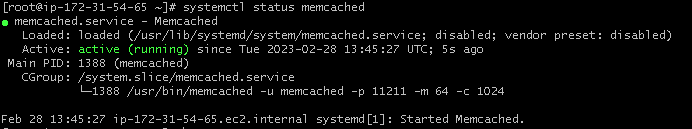
ssh –I vprofile-key.pem centos@ipv4 pub address



Wait for sometime to start our memcached service(it takes 2-5 minutes to start)

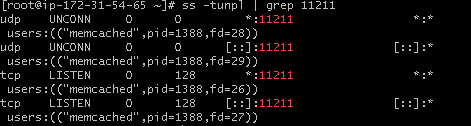
Now, Check the status of memcached:-

systemctl status memcached



Check whether it is running on right port:

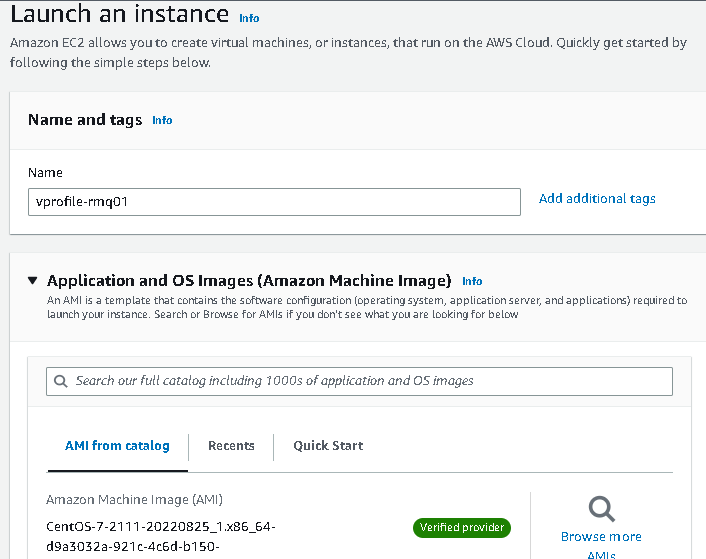
ss –tunpl | grep 11211



Now Go to AWS EC2 Console & launch another instance for RabbitMQ :

Name: vprofile-rmq01

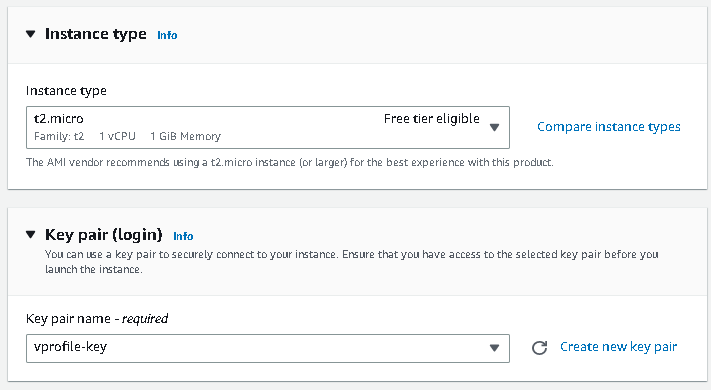
OS; CentOS

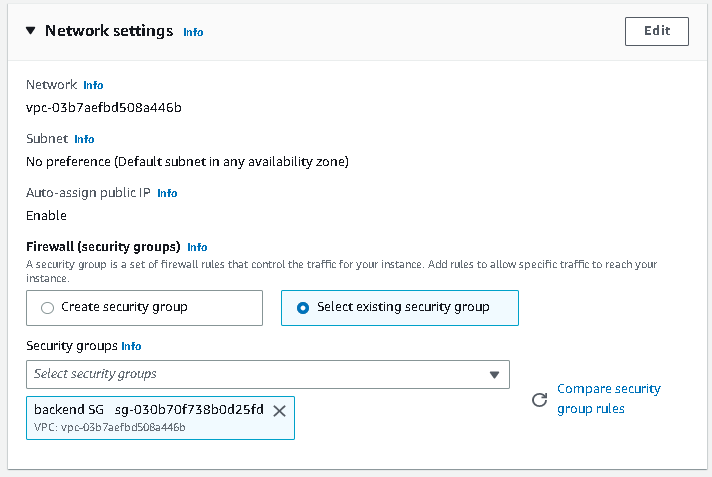


Instance Type : t2 micro

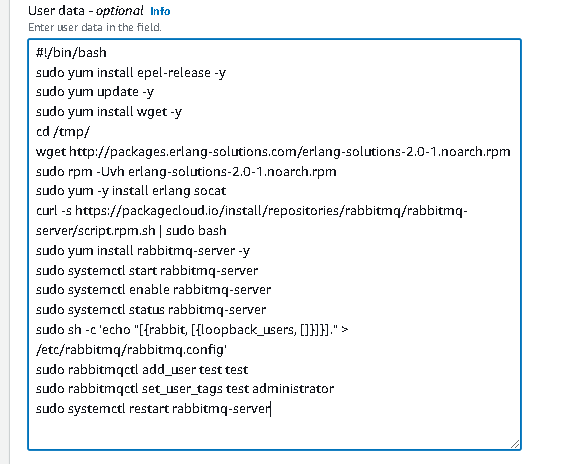
Key Pair: vprofile-key

Security Group : backend SG





Now, Paste the Script In UserData from link given:-

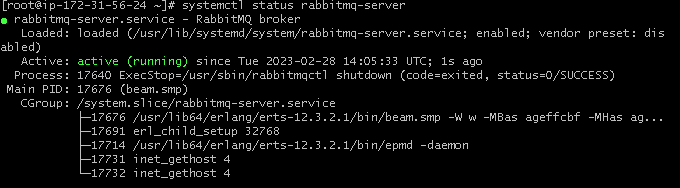


**Click Launch Instance**

Connect with ssh:

ssh –i vprofile-key.pem centos@ipv4 pub address

systemctl status rabbitmq-server

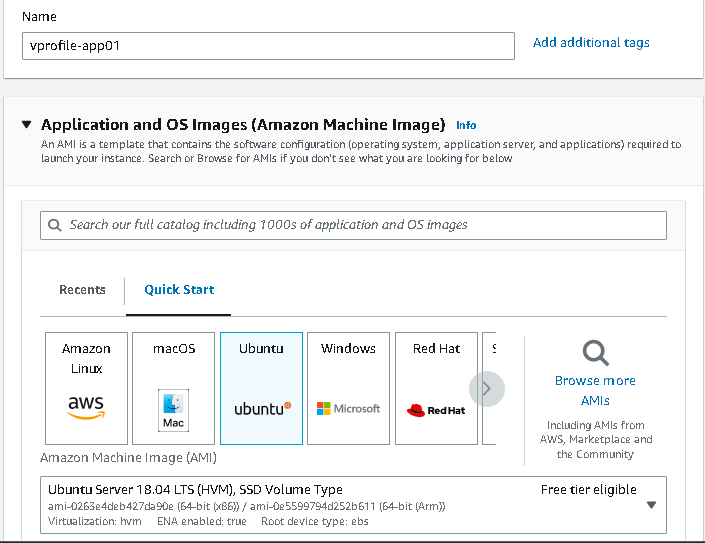


Now, Launch another server for Tomcat Application:

Go to AWS EC2 Console > Launch instance:

Name: vprofile-app01

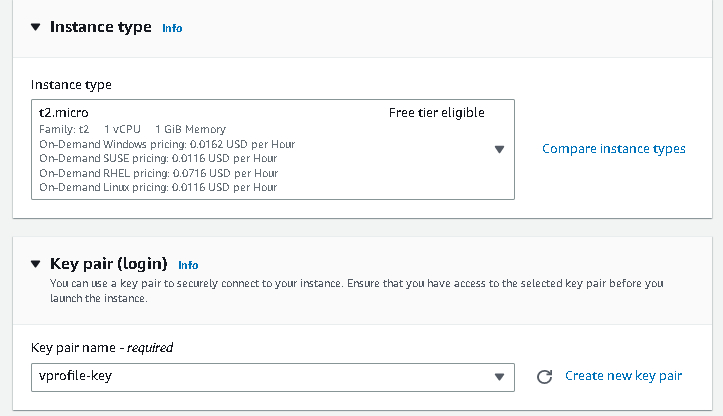
OS: Ubuntu Server 18.04 LTS

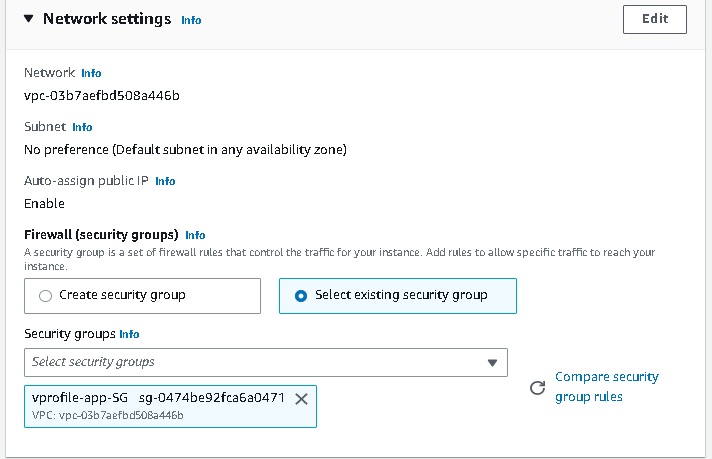


Instance Type: t2 micro

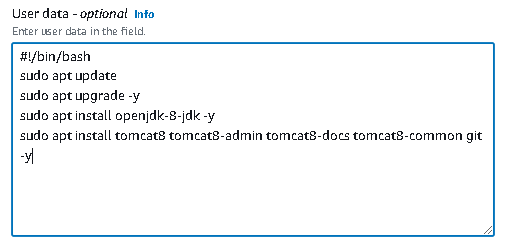
Key Pair: vprofile-key

Security Group: vprofile-app-SG





In userdata : paste the script from the given link:



**Click Launch Instance**

**5. Update IP to name mapping in route 53**

Copy all the instances private ip’s

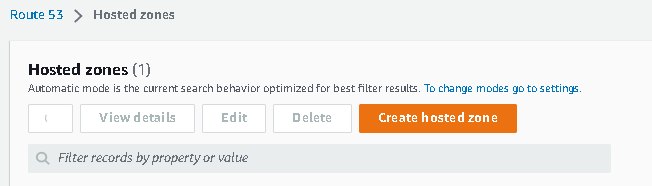
dbserver :- Private Ip addr

vprofile-mc01 :- Private Ip addr

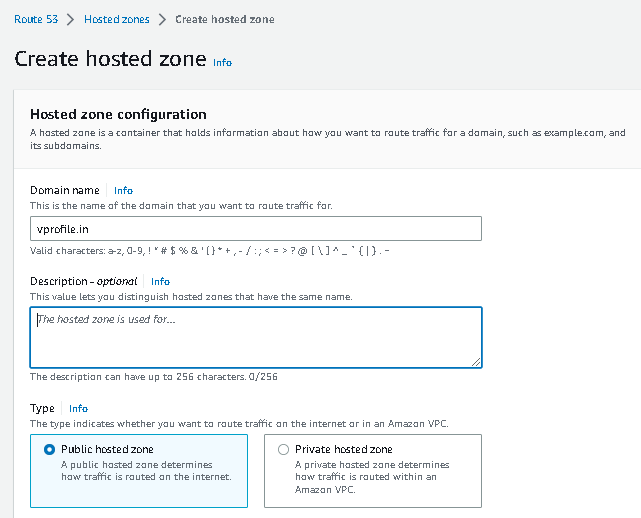
vprofile-rmq01 :- Private Ip addr

Now Goto Route 53 >

Click on Create Hosted Zone

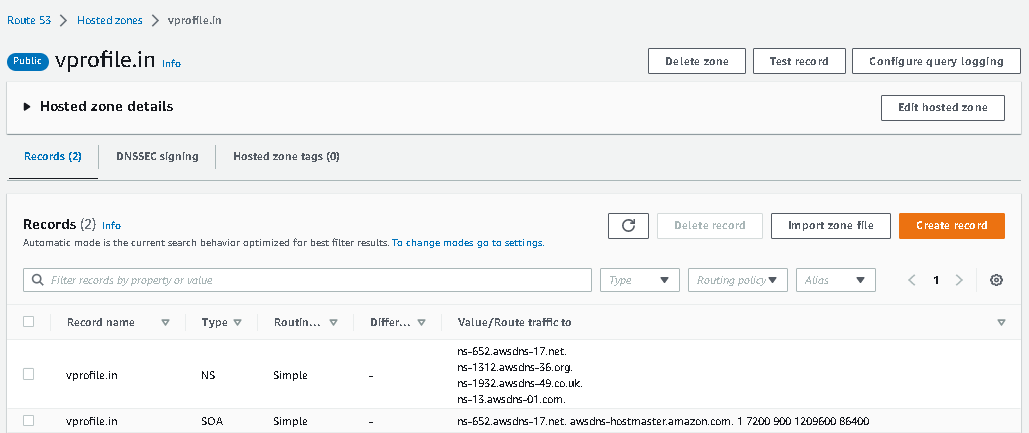


Domain Name: vprofile.in



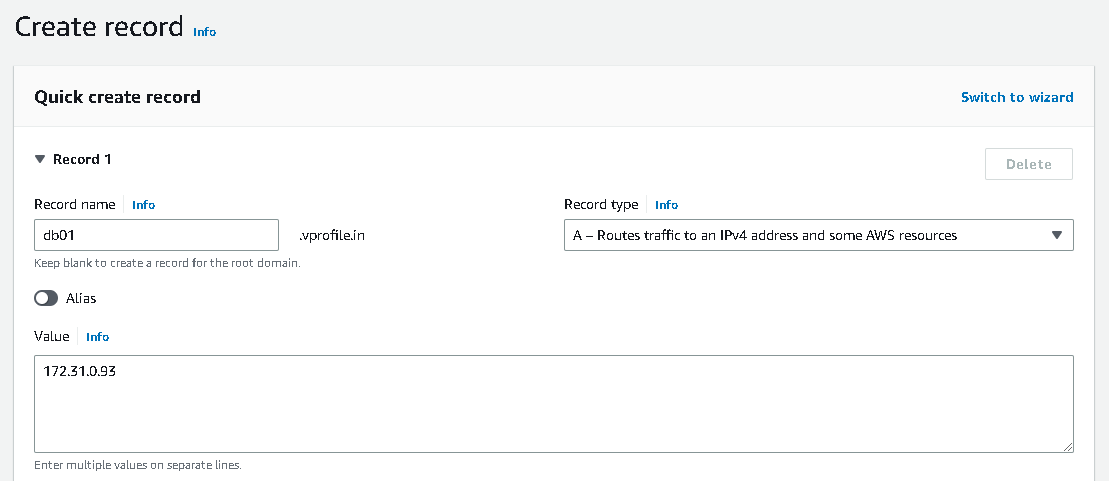
Click **Create Hosted Zone**

**Now, Click create Record**



Record Name: - db01

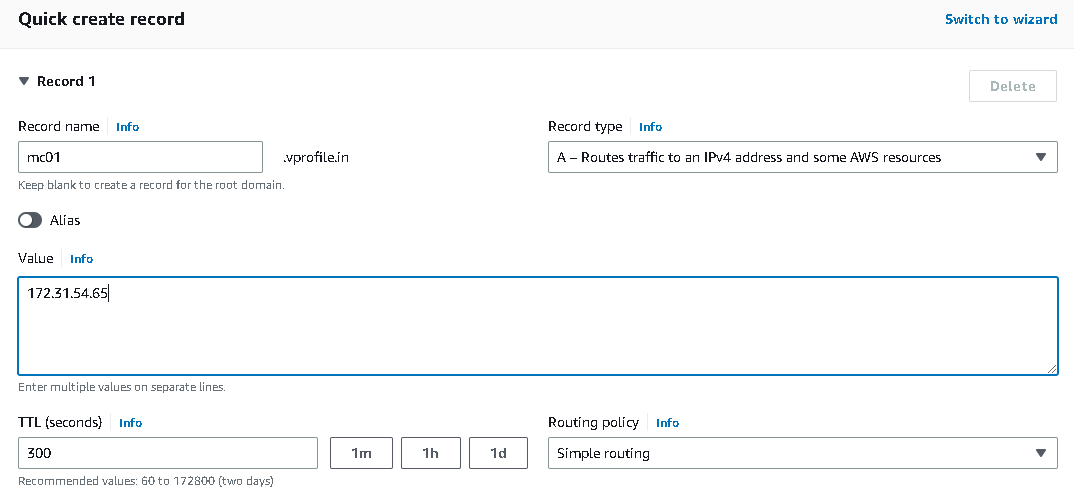
Value :- <db01 private ip>



Click Create Record

Record Name: - mc01

Value :- <mc01 private ip>

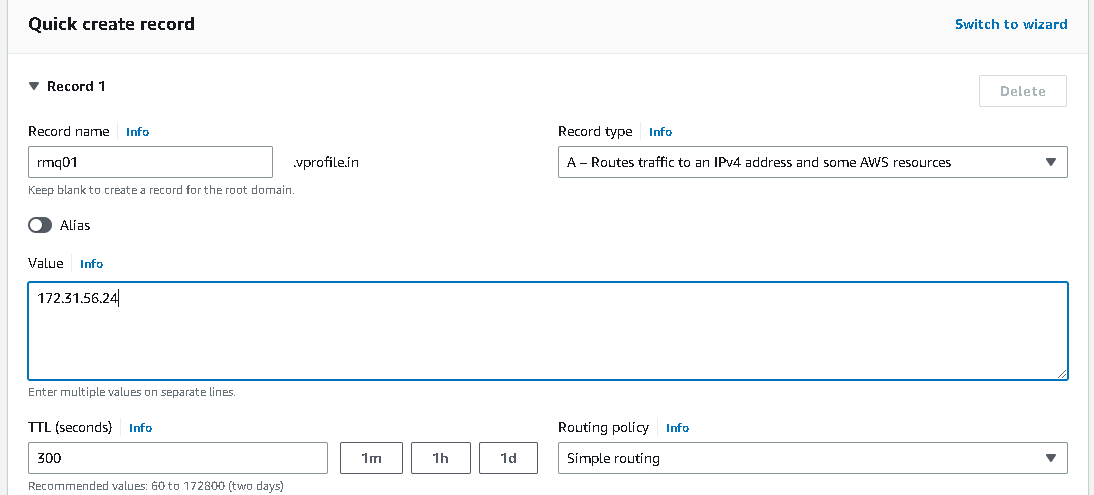


Click Create Record

-------

Record Name: - rmq01

Value :- <rmq01 private ip>



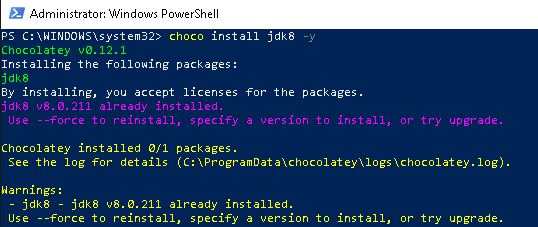
Click **Create Record**

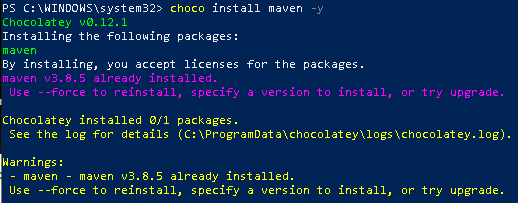
Now come to local machine > open powershell > install jdk 8 & maven

Note :- you need to install chocolatey in powershell before installing the jdk8 & maven

choco install jdk8 –y

choco install mvn -y





Now go to src > main > resources

You will have application.properties file

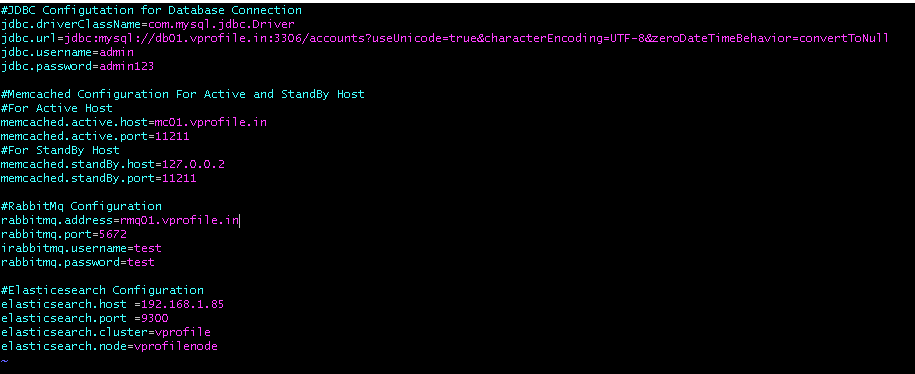
Edit the properties file

vi application.properties







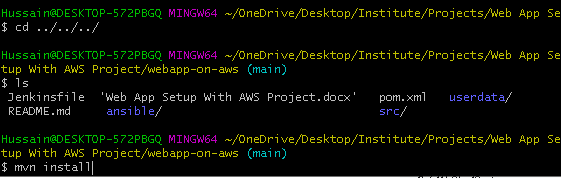


Save and exit

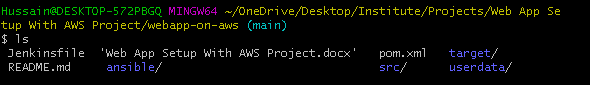
Now come to our project directory where we src directory

Type mvn install to build our artifact

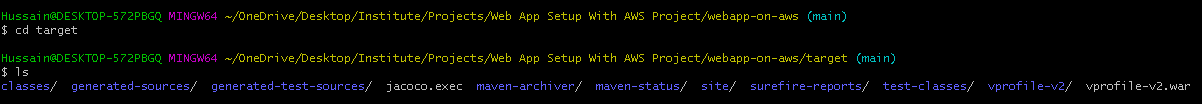
mvn install (this will take couple of minutes to build an artifact)



Type ls > you will get Target directory



Goto Target Directory >type ls > You will get vprofile-v2.war(It is built by us)



Now, we need to send this artifact(vprofile-v2.war) to the S3 bucket by using AWSCLI

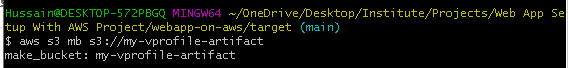
Note:- You need to install awscli in local machine & configured



Create a S3 bucket:-

aws s3 mb s3://my-vprofile-artifact

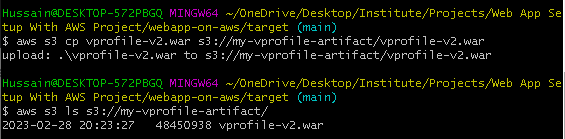
Note:- You need to take unique bucket name(S3 bucket name should be different)



Now Copy our artifact to our S3 bucket & validate :-

aws s3 cp vprofile-v2.war s3://my-vprofile-artifact/vprofile-v2.war

aws s3 ls s3://my-vprofile-artifact/

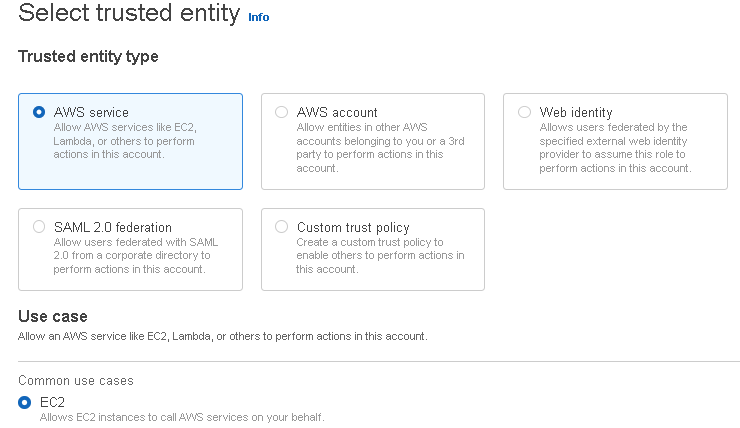


Now we need to create a Role to store our artifact to the tomcat server

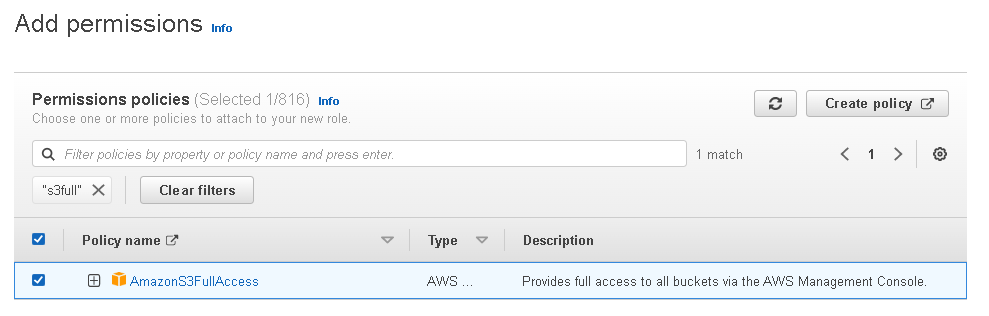
Go to IAM > Role > Create Role :

Select AWS Service >

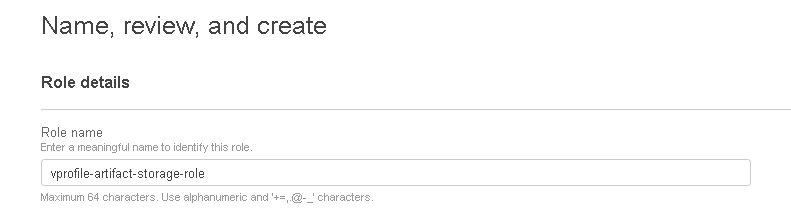
Use Case: EC2



**Add permissions > s3fullaccess**



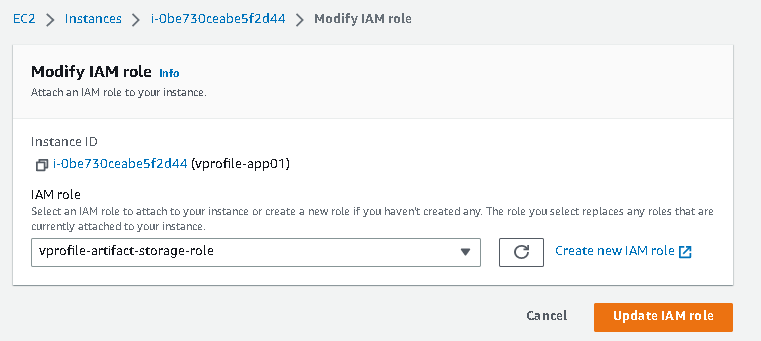
Role Name: vprofile-artifact-storage-role



Click **Create Role**

Now go to EC2 > select app01 server > Actions > Security > Modify IAM Role >

Select our Role (vprofile-artifact-storage-role)



**Click Update IAM Role**

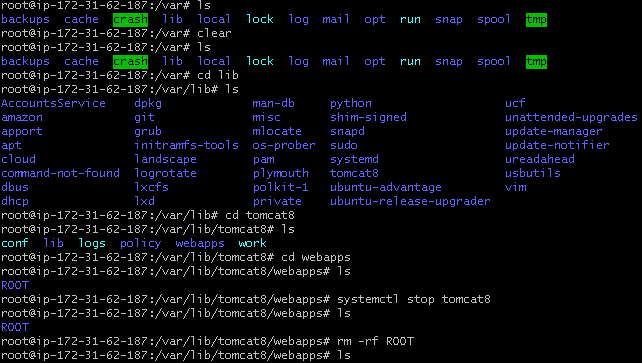
Now connect app01 server with ssh >



Now go to /var/lib/tomcat8/webapps

Stop the tomcat server: - systemctl stop tomcat

Remove ROOT directory :- rm –rf ROOT



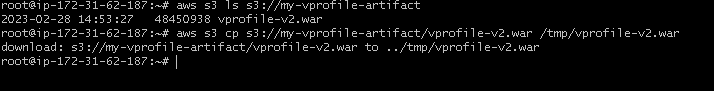
Now Install AWSCLI , (No need to Configure)



aws s3 ls s3://my-vprofile-artifact

**Download the artifact from S3 to our tomcat server in /tmp directory**

aws s3 cp s3://my-vprofile-artifact/vprofile-v2.war /tmp/vprofile-v2.war



**Now copy the artifact to** /var/lib/tomcat8/ROOT.war



Start Tomcat8 :- systemctl start tomcat8

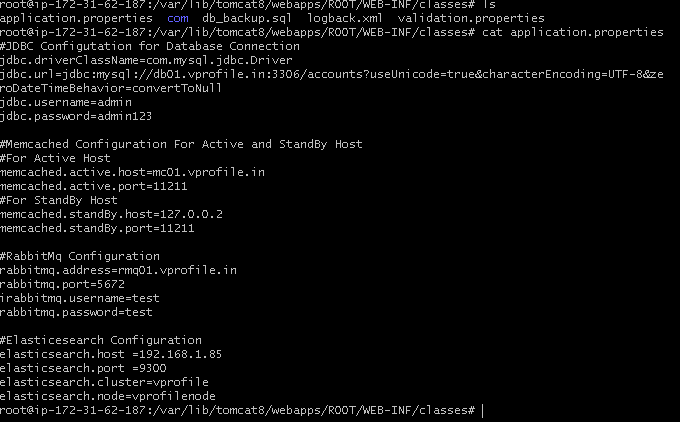


Now go to the path as given below:-

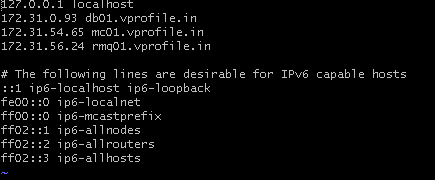
cd /var/lib/tomcat8/webapps/ROOT/WEB-INF/classes

ls

cat application.properties

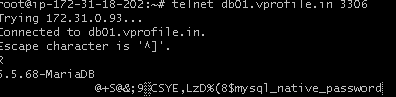


Then type vi /etc/hosts > add the private IP’s of backend servers(db01, mc01, rmq01)



Save & Exit

Now, type telnet db01.vprofile.in 3306 -> to confirm that our backend servers are connecting to tomcat server



**9. Setup ELB:-**

Go to Load Balancer > Target Groups > Create Target Groups >

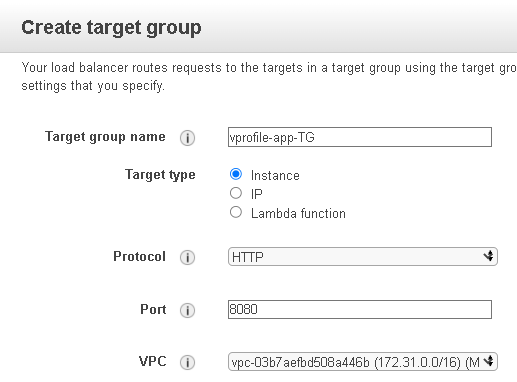
Target Group Name : vprofile-app-TG

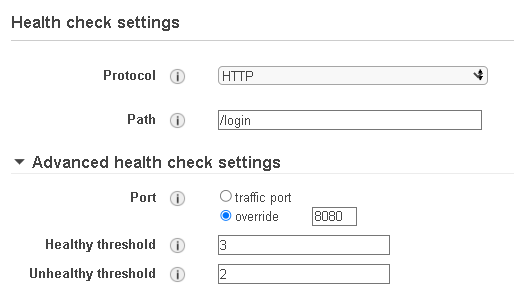
Target Type: Instance

Protocol : HTTP

Port: 8080

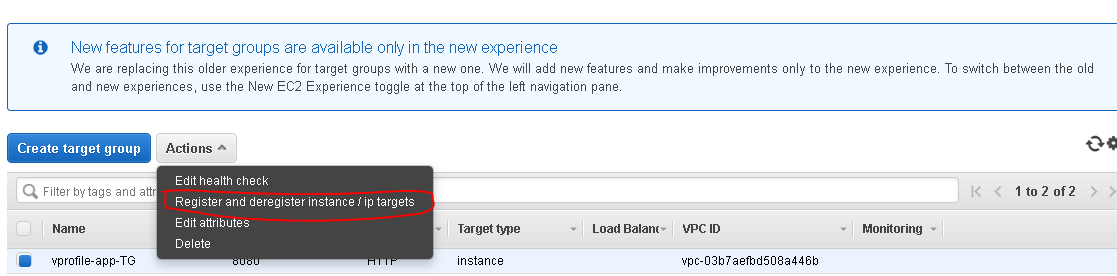
Path : /login



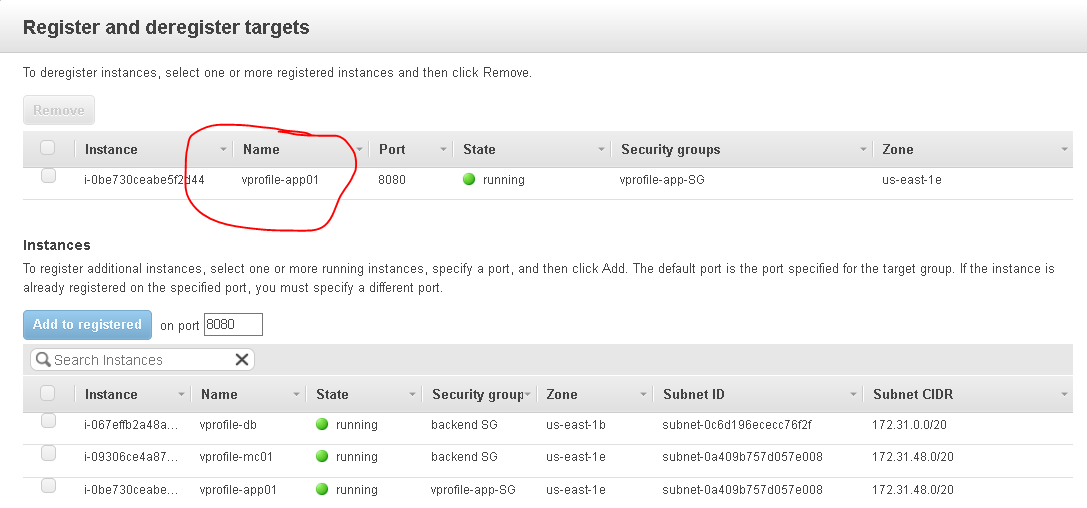


Click **Create**

Now, Select our Target Group: vprofile-app-TG >



Select the Target Group > Register Instances > Register Instances > Select our Tomcat App Instance > Add > Save Changes

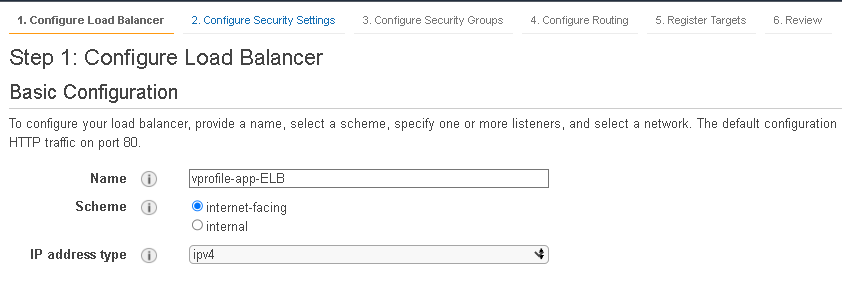


Now Go to Load Balancer > Create Load Balancer > Select Application Load Balancer > Create

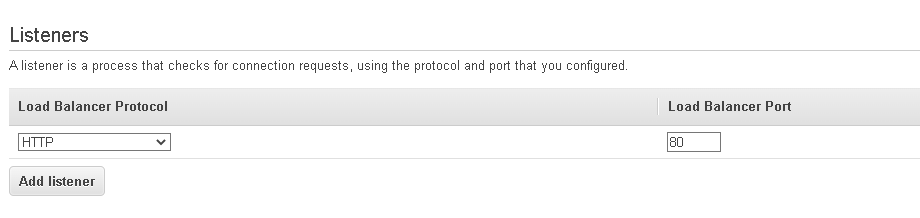
Name: vprofile-app-ELB

Scheme : Internet Facing

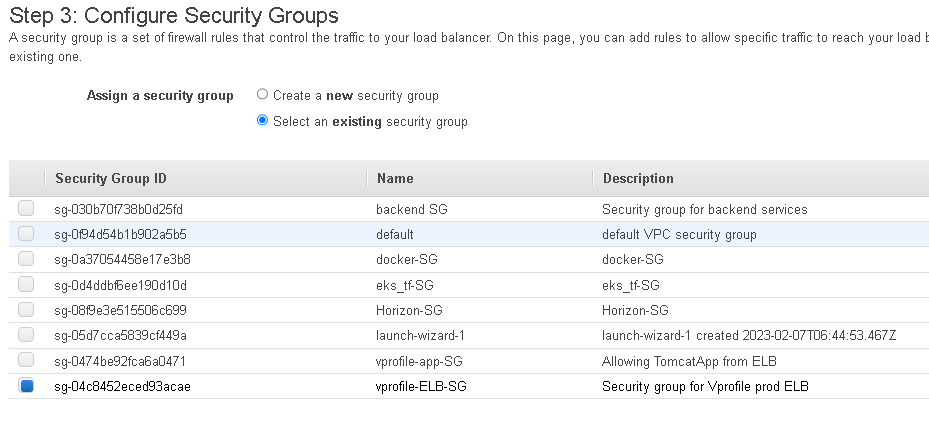
Ip type: ipv4



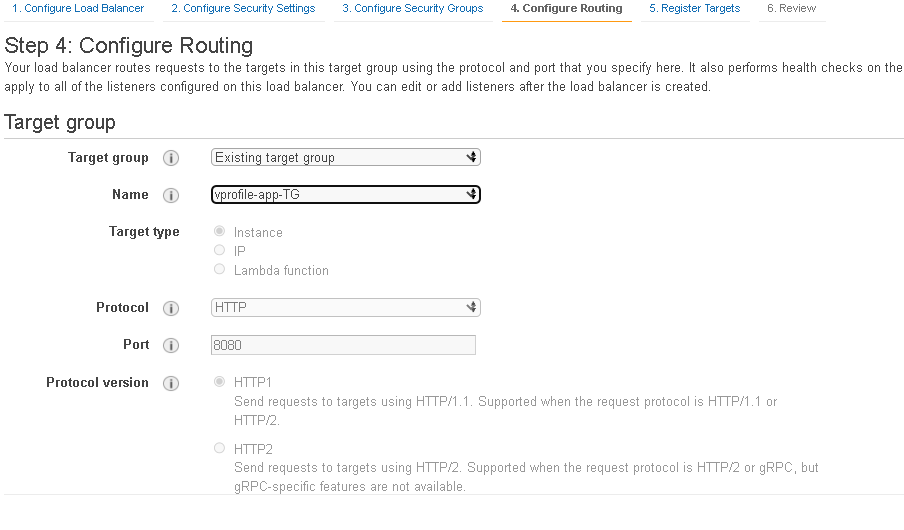
Listener: HTTP : 80



Select ELB Security Group



Select our Target Group : vprofile-app-TG

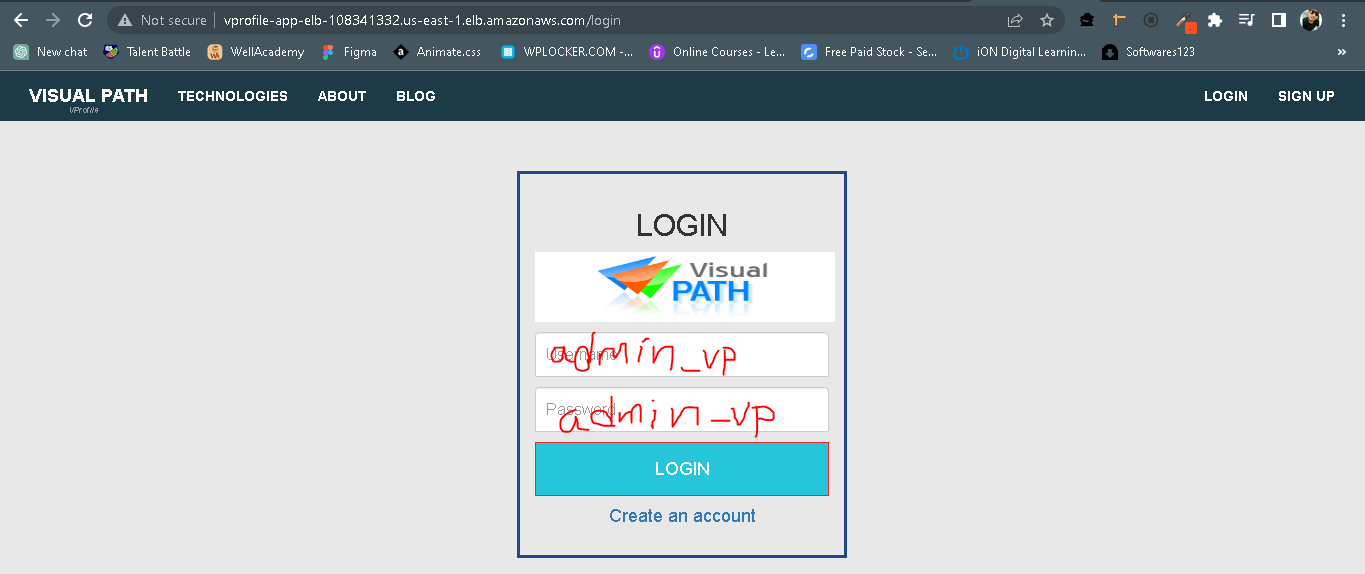


Click **Register Targets** > Click **Review**> **Next** > Click on **Create**

Then Copy the DNS Name & paste it in the URL address

Username: admin\_vp

Password: admin\_vp

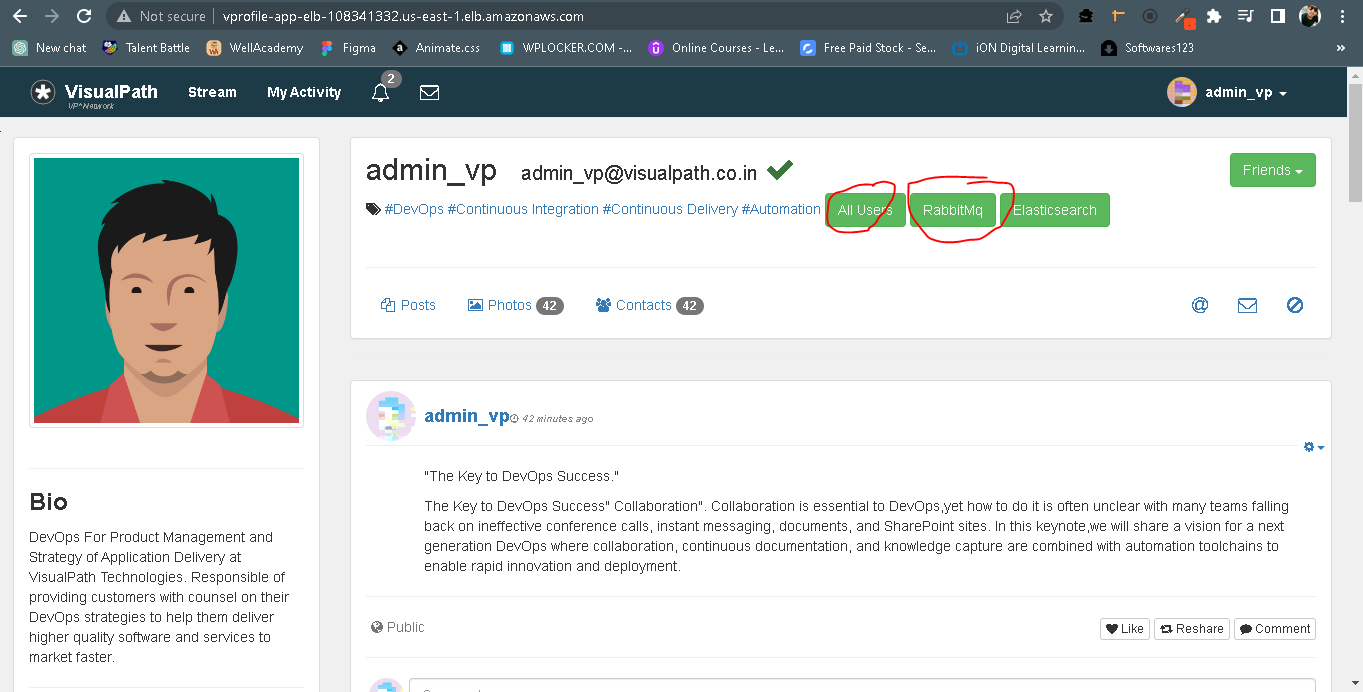


Click > Login

Once you successfully logged in, check for backend servers are working are not.

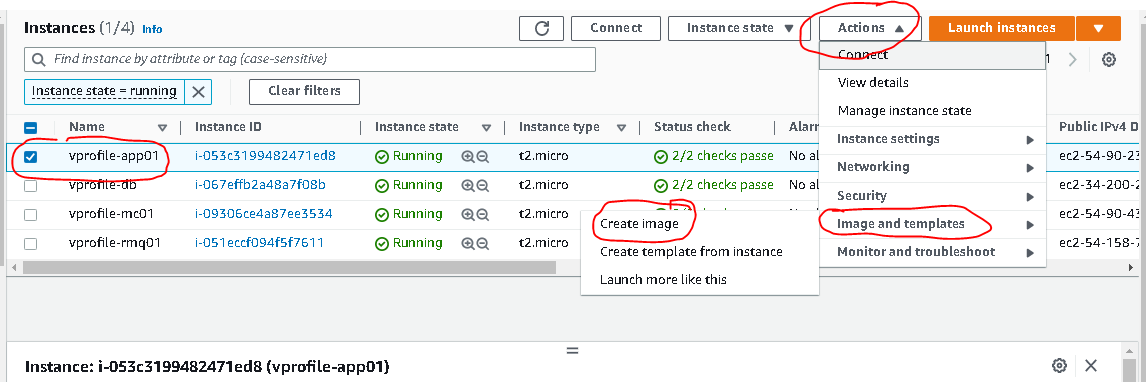
Click on All Users – If it opens a new webpage then our memcache is working

Click on RabbitMq – If it opens a new webpage then our rabbitmq is working

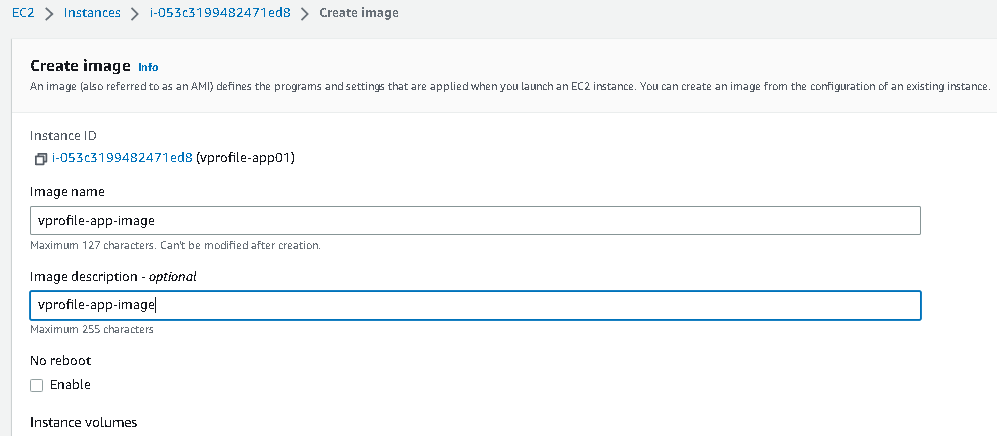


**10. Build Autoscaling Group for Tomcat Instances:-**

G0 to EC2 > select our app01 server > Actions > Image and templates > Create Image

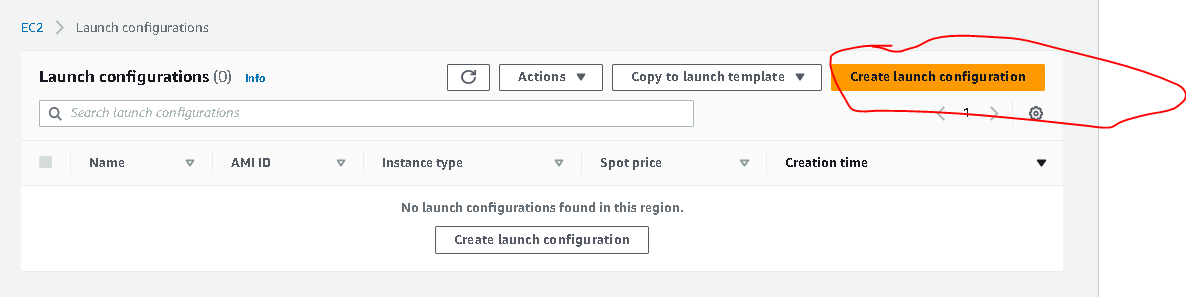


**Name : vprofle-app-image**

****

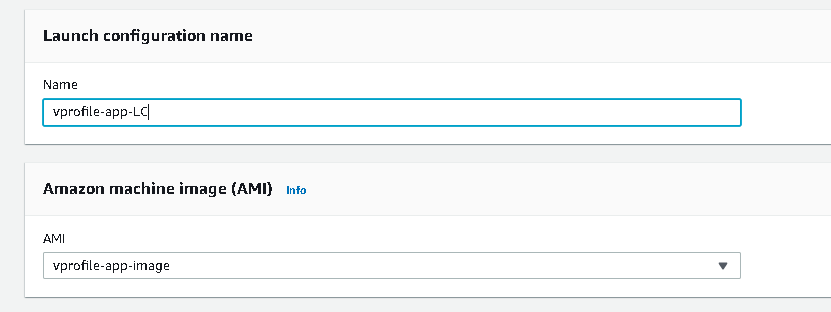
Click create

Then Go to Launch Configurations > Create Launch Configuration



Name : vprofile-app-LC

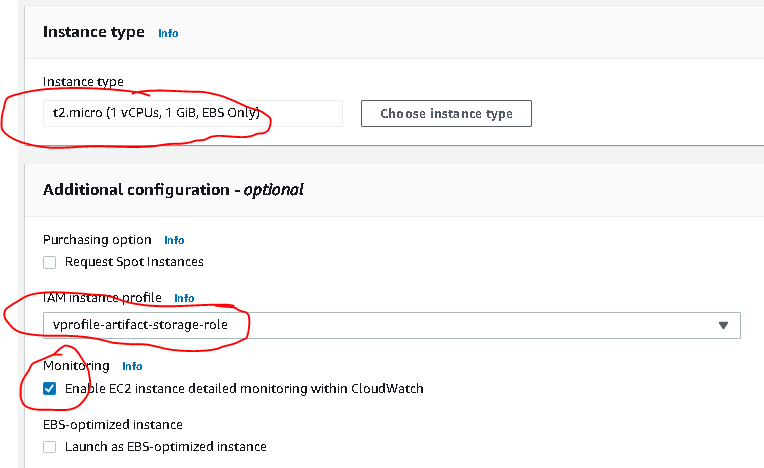
AMI : select our image (vprofile-app-image)



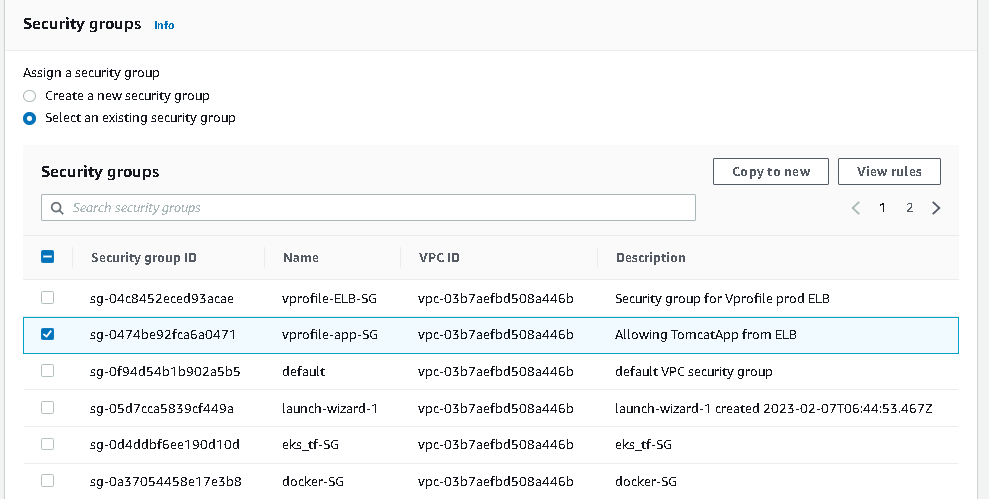
Insatnce Type : t2 micro (it is free tier)

IAM Instance Profile : select our Role (vprofile-artifact-storage-role)

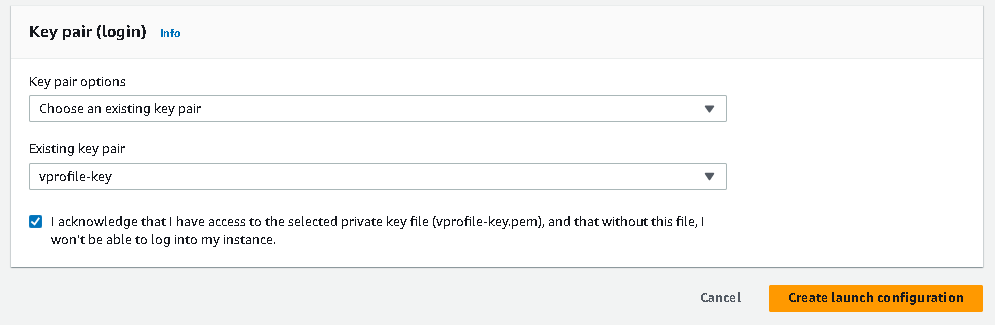
Monitoring : enable to get the logs in cloudwatch



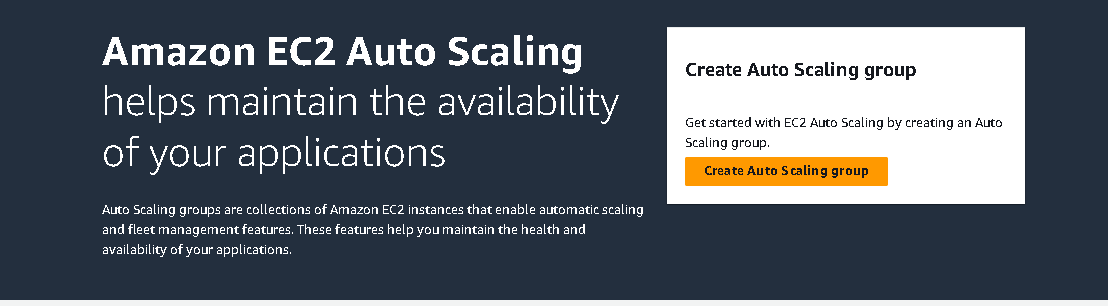
Security Group : select the **app-SG**



Keypair : vprofile-key > click **create Launch Configuration**



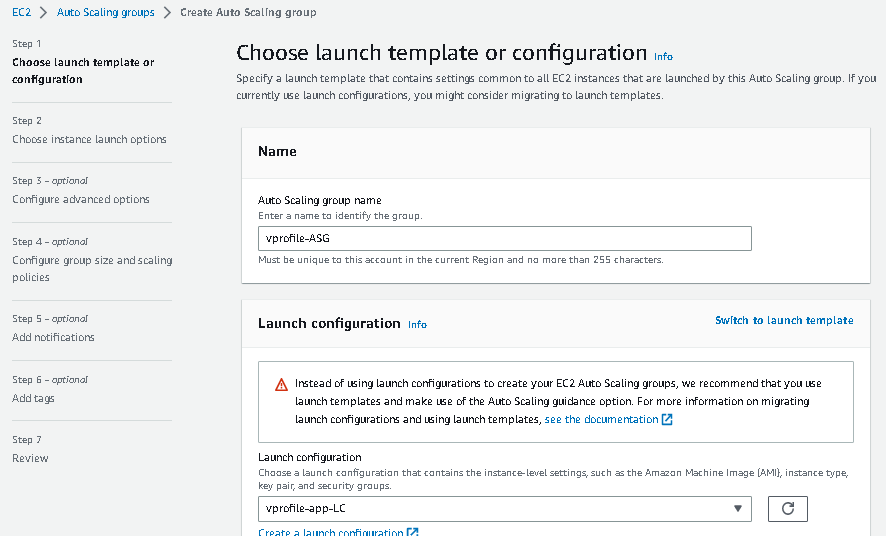
Now go to Autoscaling Groups > click on create Autoscaling Group



Name : vprofile-ASG

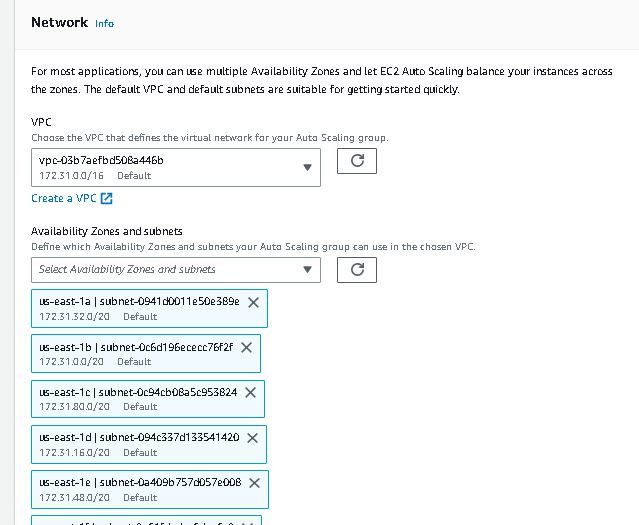
Click on **Switch to launch configuration**

Launch Configuration : vprofile-app-LC



Click Next

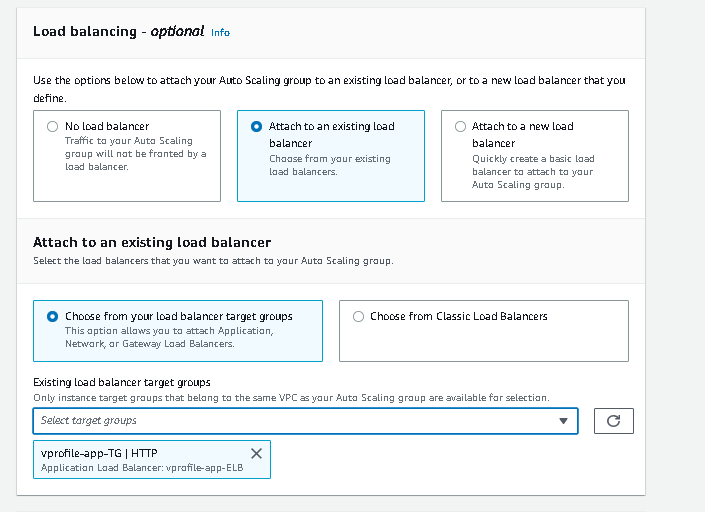
Select the VPC(in my case I’m using default VPC) & select all the Availabilty Zones



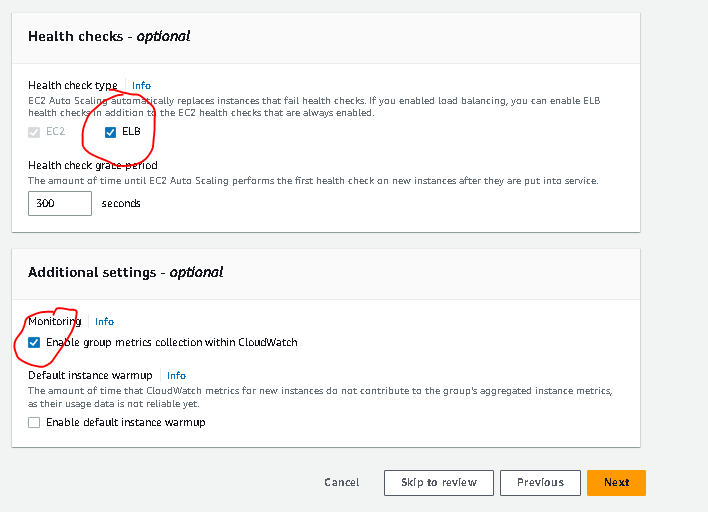
Click **Next**

Select our existing Load Balancer

Select vprofile-app-TG



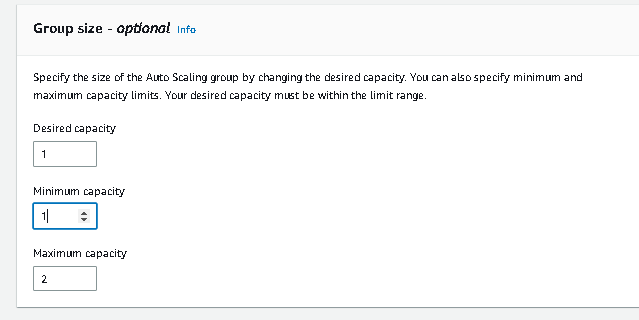
Enable **ELB** & **Monitoring**

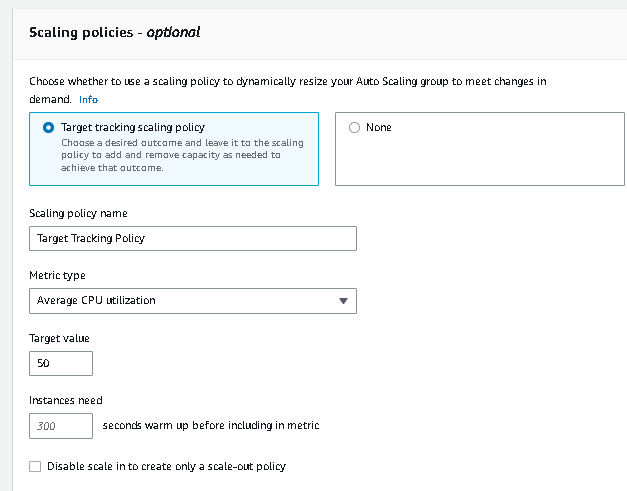


Click **Next**

Select Desired Capacity : I choose 1

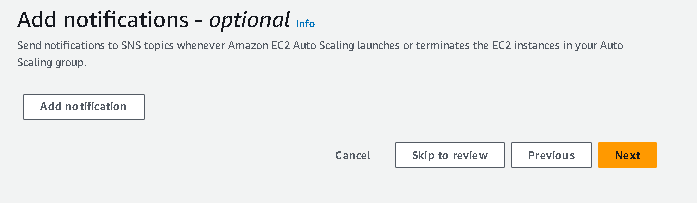
Minimum & maximum as per the requirement





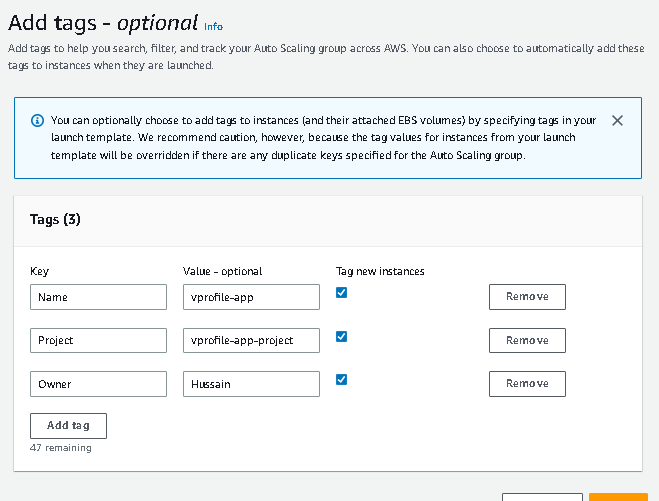
Click **Next**

Add Notification by creating SNS Topic



Click **Next**

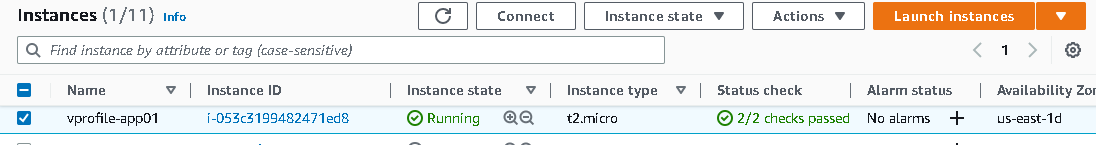
Give Some Tags as per your choice

****

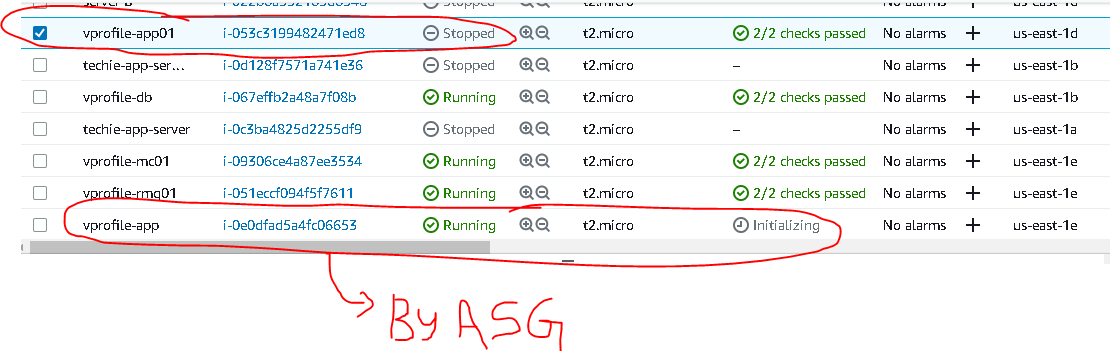
Click **Next**

Click **Create Auto Scaling Group ->**

**Now if your app01 get terminated or go down then your autoscaling group will scale up the instance with your application**

****

**I have stopped the app01 wantedly > see our ASG is launching app server automatically**

****

**Note: If you want to delete the entire project, then first delete autoscaling group > Launch Configurations > Target Groups > Load Balancer > Instances > Keypairs & Security Groups**

**If You don’t delete autoscaling first, then it will go on launching new instances when you terminate the instance**

**Congratulations**

**you have successfully deployed the java application with ELB & Autoscaling..!**

**~~~THANK YOU~~~**