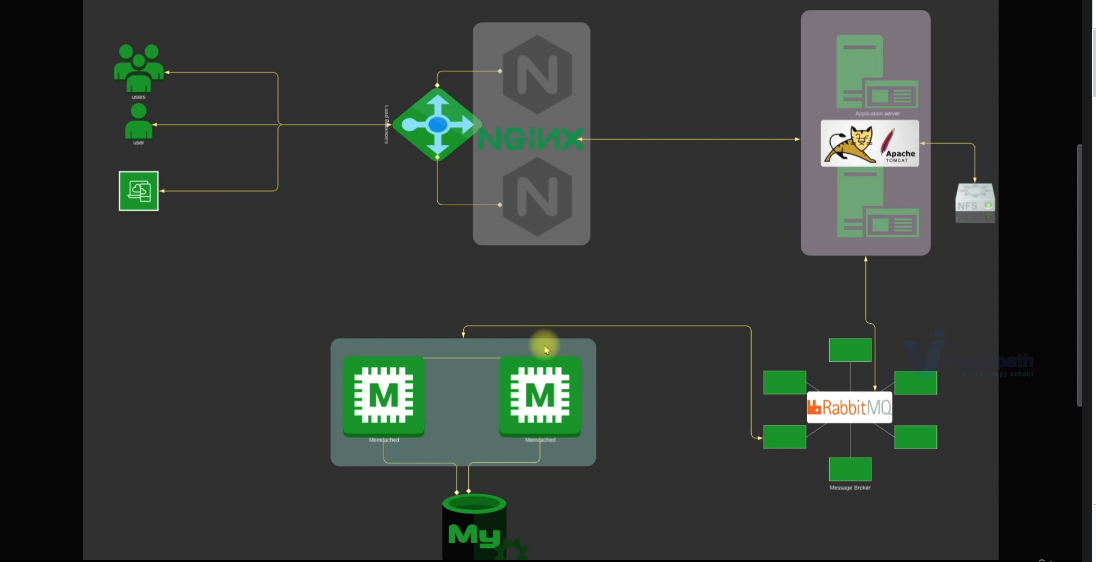
**Project-3:Lift&Shift Application Workflow to AWS**

Project Source : <https://www.udemy.com/course/devopsprojects/?src=sac&kw=devops+projects>

## Prerequisites:

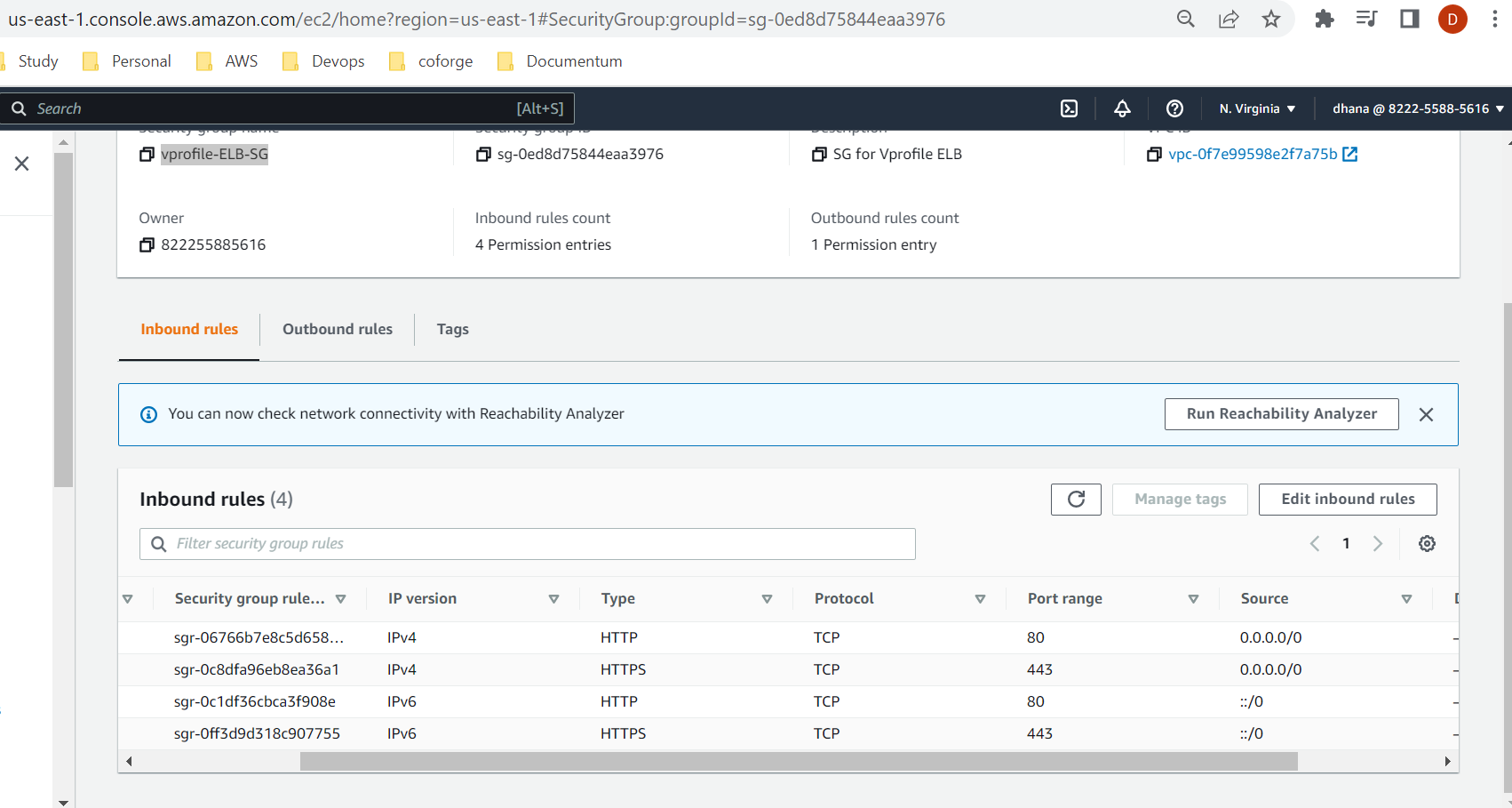
* AWS Account
* Registered DNS Name
* Maven
* JDK8
* AWS CLI

##### Architecture on DataCenter: ( Before AWS)

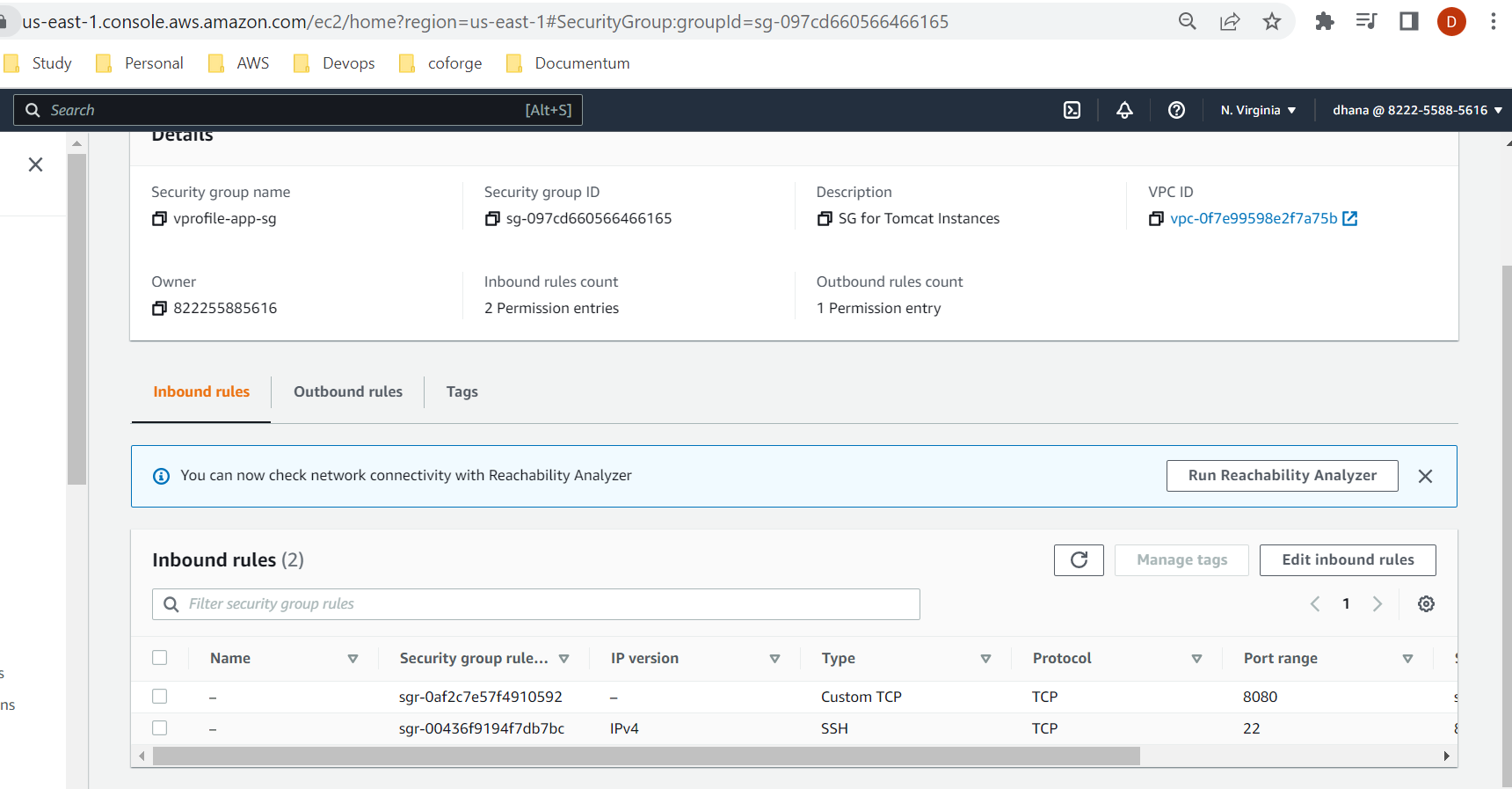


### Step-1 Create Security Groups for Services

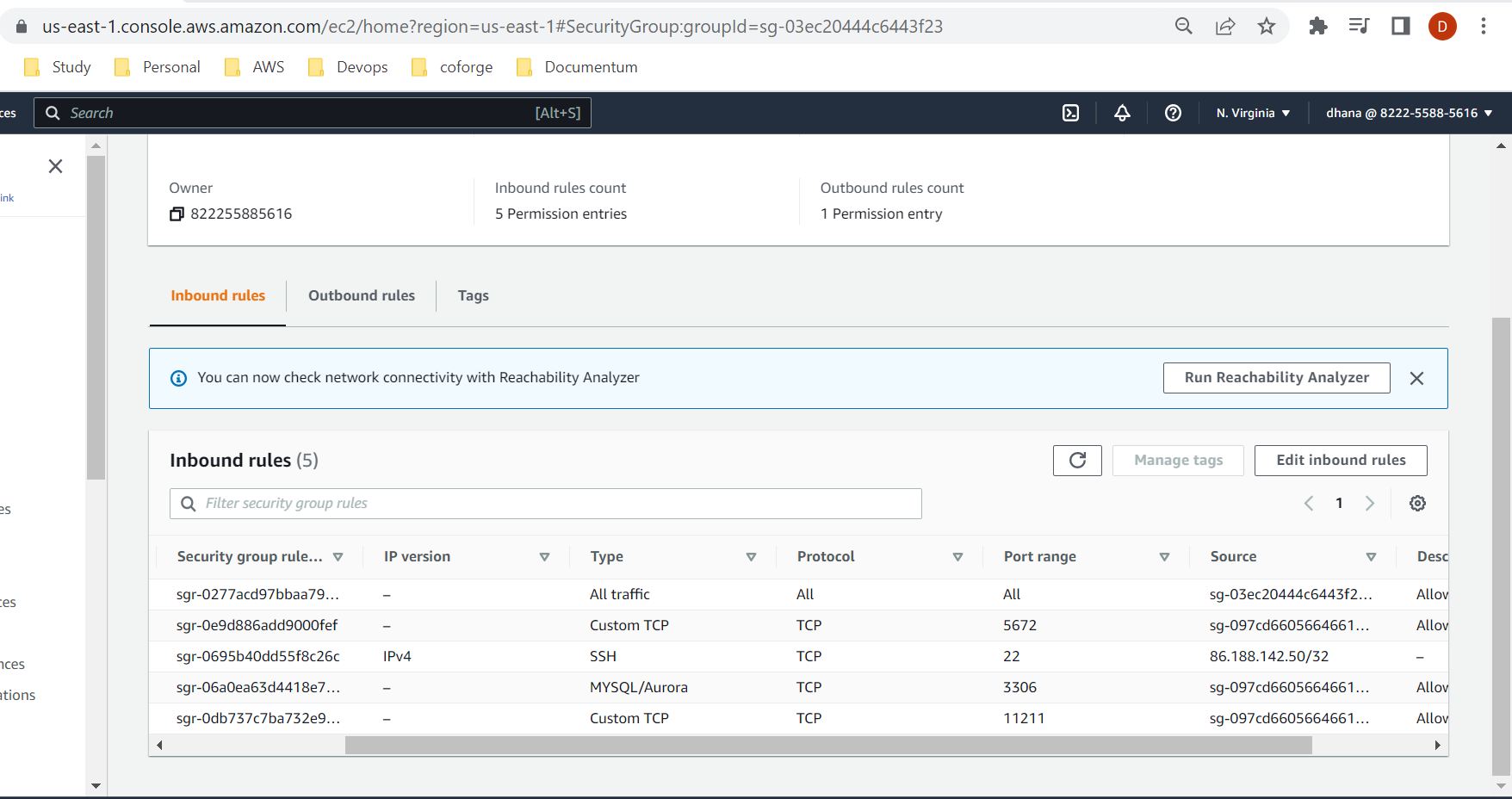
* We will create vprofile-ELB-SG first. We will configure Inbound rules to Allow both HTTP and HTTPS on port 80 and 443 respectively from Anywhere IPv4 and IPv6.



Next we will create vprofile-app-SG. We will open port 8080 to accept connections from vprofile-ELb-SG.

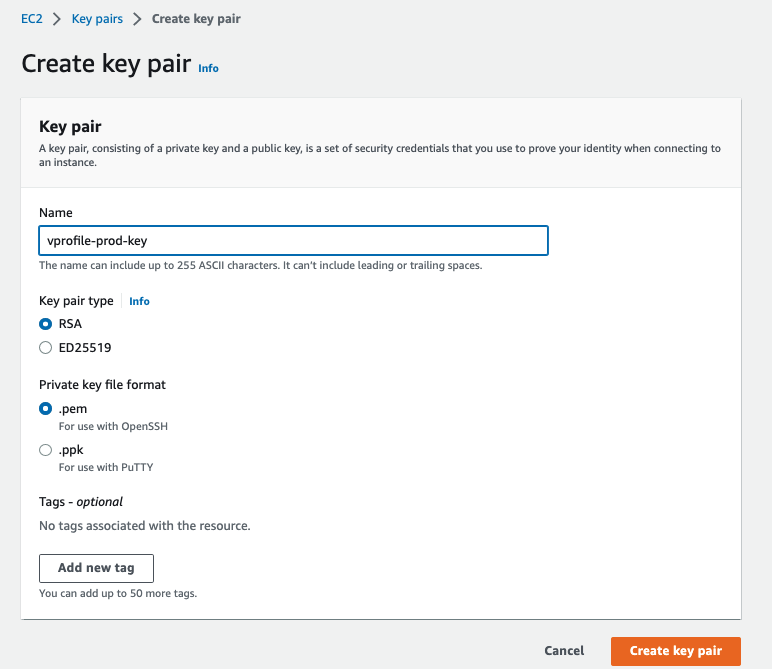


Finally, we will create vprofile-backend-SG. WE need to open port 3306 for MySQL, 11211 for Memcached and 5672 for RabbitMQ server. We can check which ports needed for application services to communicate each other from application.properties file under src/main/resources directory.We also need to open commucation AllTraffic from own SecGrp for backend services to communicate with each other.



### Step-2 Create KeyPair to Connect EC2 instances

Keypair to connect our instances via SSH.



### Step-3

### Provision Backend EC2 instances with UserData script

##### DB Instance:

* Create DB instance with below details.We will also add Inbound rule to vprofile-backend-SG for SSH on port 22 from MyIP to be able to connect our db instance via SSH.

Name: vprofile-db01

Project: vprofile

AMI: Centos 7

InstanceType: t2.micro

SecGrp: vprofile-backend-SG

UserData: mysql.sh

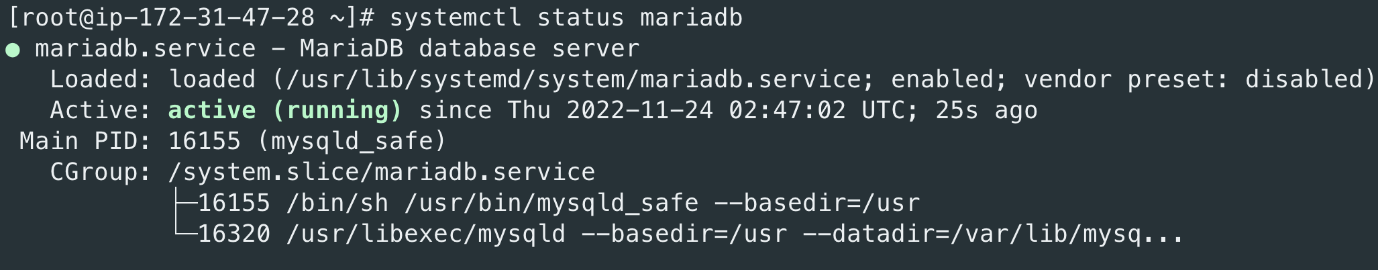
* Once our instance is ready, we can SSH into the server and check if userdata script is executed.We can also check status of mariadb.

ssh -i vprofile-prod-key.pem centos@<public\_ip\_of\_instance>

sudo su -

curl http://169.254.169.254/latest/user-data

systemctl status mariadb

[](https://github.com/rumeysakdogan/AWS_DevOps_Projects/blob/main/Project-3%3A%20Lift%20and%20Shift%20Application%20Workload%20to%20AWS/images/mariadb-running.png)

##### Memcached Instance:

* Create Memcached instance with below details.

Name: vprofile-mc01

Project: vprofile

AMI: Centos 7

InstanceType: t2.micro

SecGrp: vprofile-backend-SG

UserData: memcache.sh

* Once our instance is ready, we can SSH into the server and check if userdata script is executed.We can also check status of memcache service and if it is listening on port 11211.

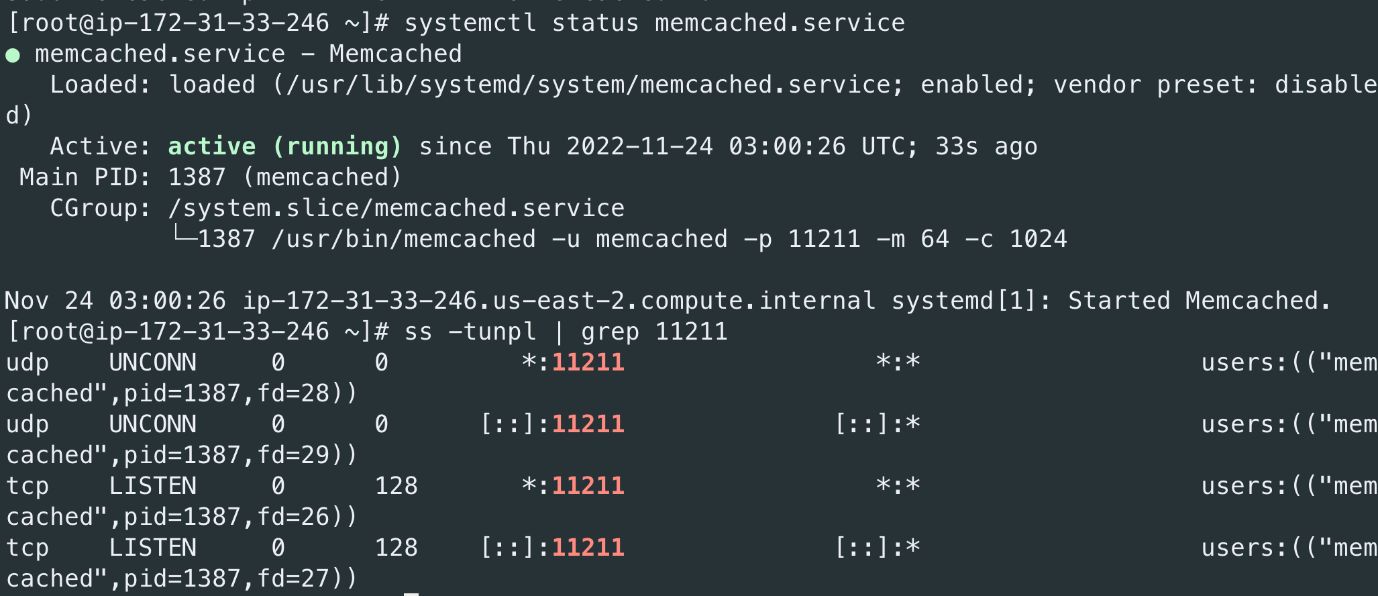
ssh -i vprofile-prod-key.pem centos@<public\_ip\_of\_instance>

sudo su -

curl http://169.254.169.254/latest/user-data

systemctl status memcached.service

ss -tunpl | grep 11211

[](https://github.com/rumeysakdogan/AWS_DevOps_Projects/blob/main/Project-3%3A%20Lift%20and%20Shift%20Application%20Workload%20to%20AWS/images/memcached-running.png)

##### RabbitMQ Instance:

* Create RabbitMQ instance with below details.

Name: vprofile-rmq01

Project: vprofile

AMI: Centos 7

InstanceType: t2.micro

SecGrp: vprofile-backend-SG

UserData: rabbitmq.sh

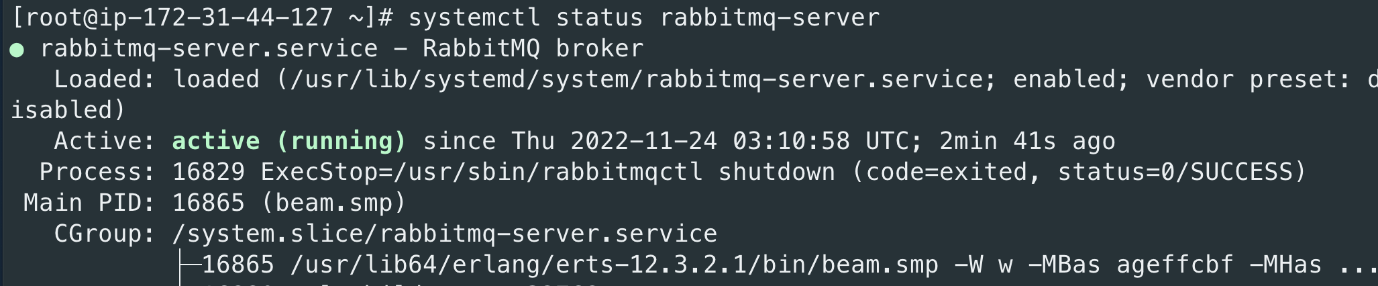
* Once our instance is ready, we can SSH into the server and check if userdata script is executed.We can also check status of rabbitmq service.

ssh -i vprofile-prod-key.pem centos@<public\_ip\_of\_instance>

sudo su -

curl http://169.254.169.254/latest/user-data

systemctl status rabbitmq-server

[](https://github.com/rumeysakdogan/AWS_DevOps_Projects/blob/main/Project-3%3A%20Lift%20and%20Shift%20Application%20Workload%20to%20AWS/images/rabbitmq-running.png)

Note: It may take some time to run userdata script after you connect to server. You can check the process *ps -ef* to see if the process start for service. If not wait sometime and check with *systemctl status <service\_name>* command again.

### Create Private Hosted Zone in Route53

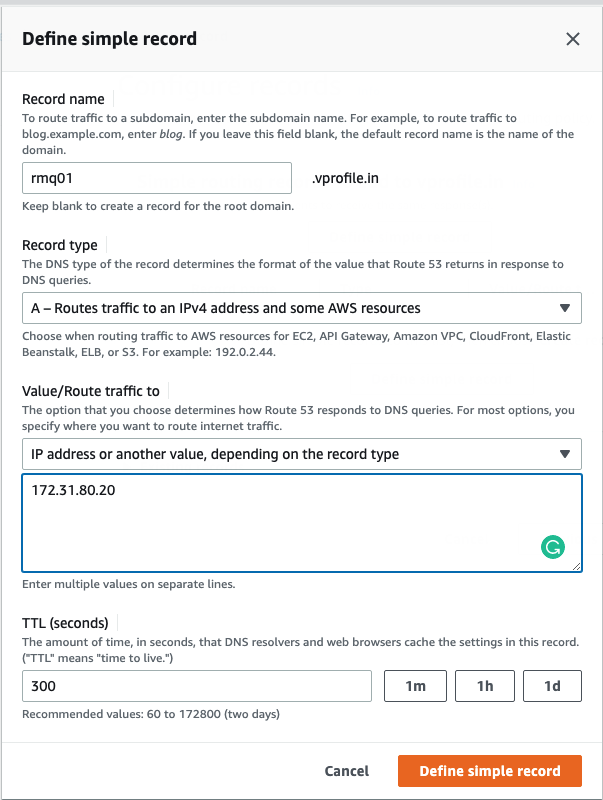
* Our backend stack is running. Next we will update Private IP of our backend services in Route53 Private DNS Zone.Lets note down Private IP addresses.

rmq01 172.31.80.20

db01 172.31.22.178

mc01 172.31.87.132

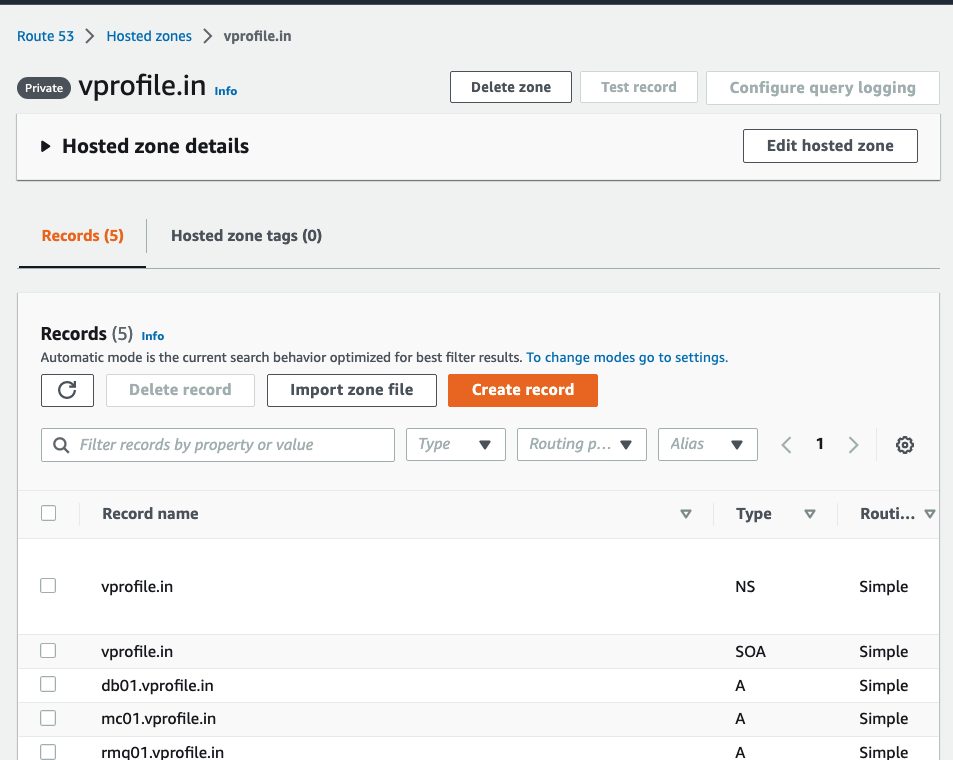
* Create vprofile.in Private Hosted zone in Route53. we will pick Default VPC in N.Virginia region.

[](https://github.com/rumeysakdogan/AWS_DevOps_Projects/blob/main/Project-3%3A%20Lift%20and%20Shift%20Application%20Workload%20to%20AWS/images/route53-records.png)

* Now we will create records for our backend services. The purpose of this activity is we will use these record names in our application.properties file. Even if IP address of the services, our application won't need to change the config file.

Simple Routing -> Define Simple Record

Value/Route traffic to: IP address or another value

[](https://github.com/rumeysakdogan/AWS_DevOps_Projects/blob/main/Project-3%3A%20Lift%20and%20Shift%20Application%20Workload%20to%20AWS/images/backend-records.png)

### Step-4: Provision Application EC2 instances with UserData script

### Provision Application EC2 instances with UserData script

* Create Tomcat instance with below details.We will also add Inbound rule to vprofile-app-SG for SSH on port 22 from MyIP to be able to connect our db instance via SSH.

Name: vprofile-app01

Project: vprofile

AMI: Ubuntu 18.04

InstanceType: t2.micro

SecGrp: vprofile-app-SG

UserData: tomcat\_ubuntu.sh

### Step-5: Create Artifact Locally with MAVEN

* Clone the repository.

git clone https://github.com/rumeysakdogan/vprofile-project.git

* Before we create our artifact, we need to do changes to our application.properties file under /src/main/resources directory for below lines.

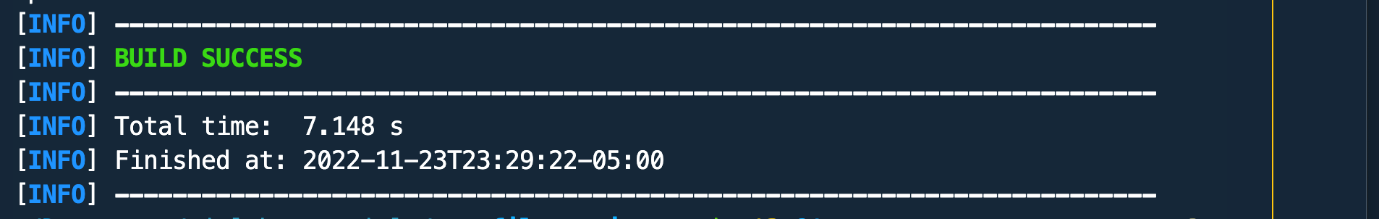
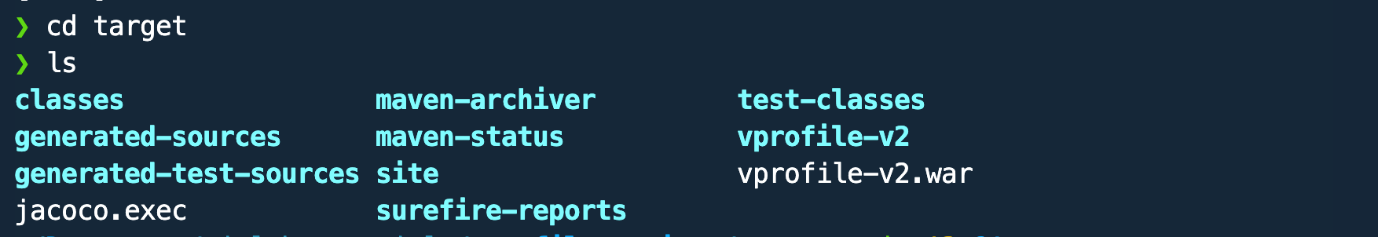
jdbc.url=jdbc:mysql://db01.vprofile.in:3306/accounts?useUnicode=true&

memcached.active.host=mc01.vprofile.in

rabbitmq.address=rmq01.vprofile.in

* We will go to vprofile-project root directory to the same level pom.xml exists. Then we will execute below command to create our artifact vprofile-v2.war:

mvn install

[](https://github.com/rumeysakdogan/AWS_DevOps_Projects/blob/main/Project-3%3A%20Lift%20and%20Shift%20Application%20Workload%20to%20AWS/images/mvn-build-success.png) [](https://github.com/rumeysakdogan/AWS_DevOps_Projects/blob/main/Project-3%3A%20Lift%20and%20Shift%20Application%20Workload%20to%20AWS/images/artifact-created.png)

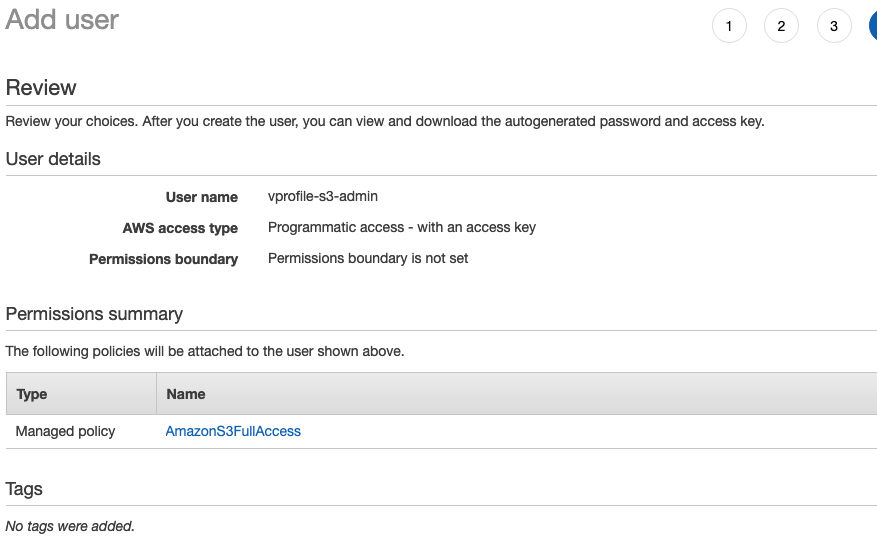
### Step-6: Create S3 bucket using AWS CLI, copy artifact

* We will upload our artifact to s3 bucket from AWS CLI and our Tomcat server will get the same artifact from s3 bucket.
* We will create an IAM user for authentication to be used from AWS CLI.

name: vprofile-s3-admin

Access key - Programmatic access

Policy: s3FullAccess

[](https://github.com/rumeysakdogan/AWS_DevOps_Projects/blob/main/Project-3%3A%20Lift%20and%20Shift%20Application%20Workload%20to%20AWS/images/iam-user.png)

* Next we will configure our aws cli to use iam user credentials.

aws configure

AccessKeyID:

SecretAccessKey:

region: us-east-1

format: json

* Create bucket. Note: S3 buckets are global so the naming must be UNIQUE!

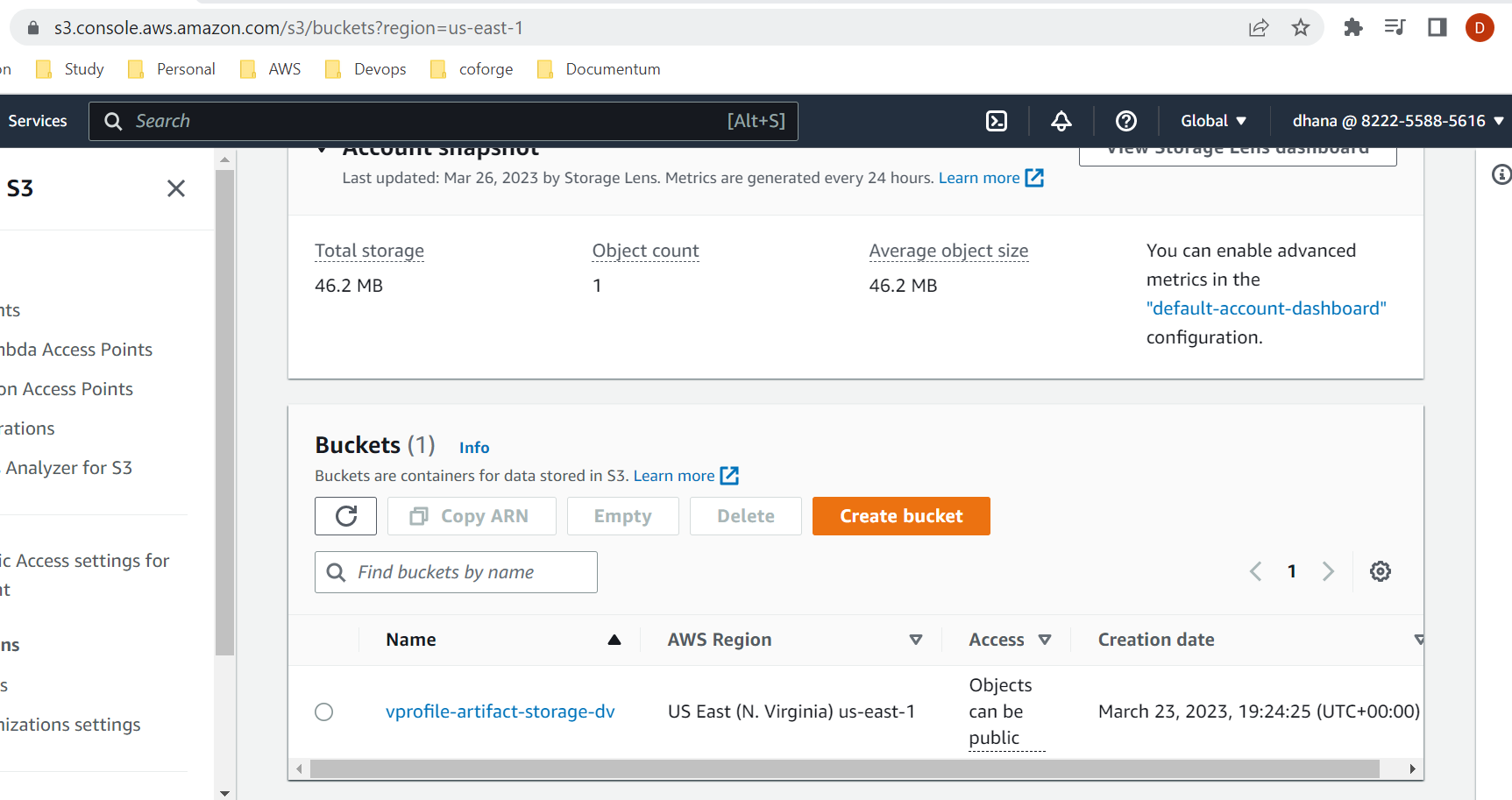
aws s3 mb s3://vprofile-artifact-storage-dv

* Go to target directory and copy the artifact to bucket with below command. Then verify by listing objects in the bucket.

aws s3 cp vprofile-v2.war s3://vprofile-artifact-storage-dv

aws s3 ls vprofile-artifact-storage-dv

* We can verify the same from AWS Console.



### Step-7: Download Artifact to Tomcat server from S3

* In order to download our artifact onto Tomcat server, we need to create IAM role for Tomcat. Once role is created we will attach it to our app01 server.

Type: EC2

Name: vprofile-artifact-storage-role

Policy: s3FullAccess

* Before we login to our server, we need to add SSH access on port 22 to our vprofile-app-SG.
* Then connect to app011 Ubuntu server.

ssh -i "vprofile-prod-key.pem" ubuntu@<public\_ip\_of\_server>

sudo su -

systemctl status tomcat8

* We will delete ROOT (where default tomcat app files stored) directory under /var/lib/tomcat8/webapps/. Before deleting it we need to stop Tomcat server.

cd /var/lib/tomcat8/webapps/

systemctl stop tomcat8

rm -rf ROOT

* Next we will download our artifact from s3 using aws cli commands. First we need to install aws cli. We will initially download our artifact to /tmp directory, then we will copy it under /var/lib/tomcat8/webapps/ directory as ROOT.war. Since this is the default app directory, Tomcat will extract the compressed file.

apt install awscli -y

aws s3 ls s3://vprofile-artifact-storage-rd

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

cd /tmp

cp vprofile-v2.war /var/lib/tomcat8/webapps/ROOT.war

systemctl start tomcat8

* We can also verify application.properties file has the latest changes.

cat /var/lib/tomcat8/webapps/ROOT/WEB-INF/classes/application.properties

* We can validate network connectivity from server using telnet.

apt install telnet

telnet db01.vprofile.in 3306

### Step-8: Setup LoadBalancer

* Before creating LoadBalancer , first we need to create Target Group.

Intances

Target Grp Name: vprofile-elb-TG

protocol-port: HTTP:8080

healtcheck path : /login

Advanced health check settings

Override: 8080

Healthy threshold: 3

available instance: app01 (Include as pending below)

* Now we will create our Load Balancer.

vprofile-prod-elb

Internet Facing

Select all AZs

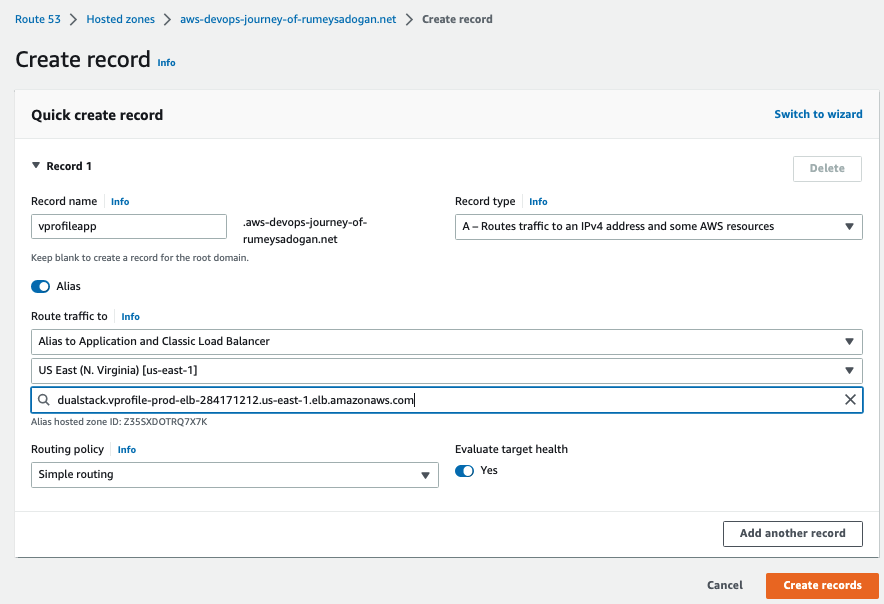
SecGrp: vprofile-elb-secGrp

Listeners: HTTP, HTTPS

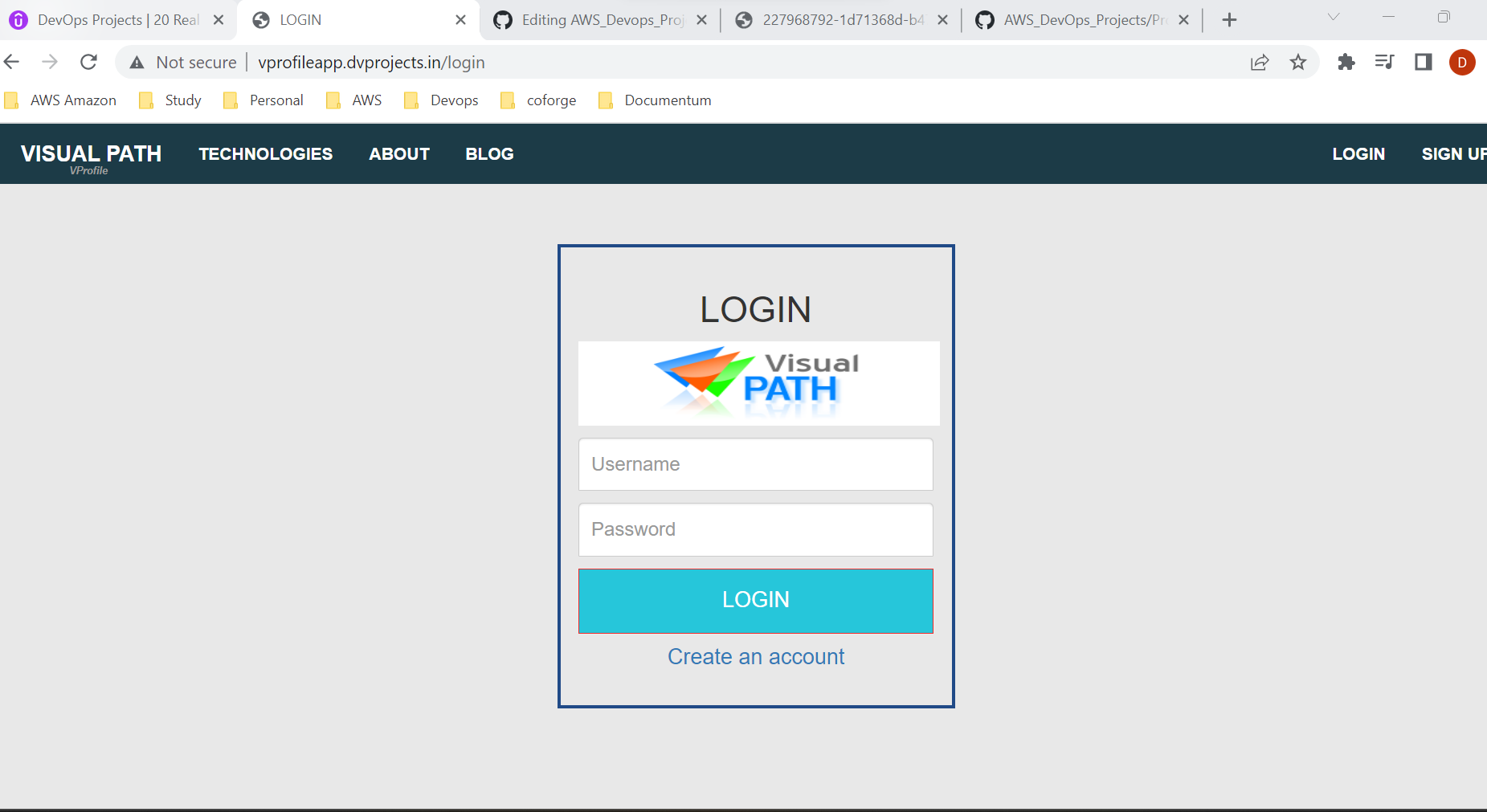
Select the certificate for HTTPS

### Step-9: Create Route53 record for ELB endpoint

* We will create an A record with alias to ALB so that we can use our domain name to reach our application.

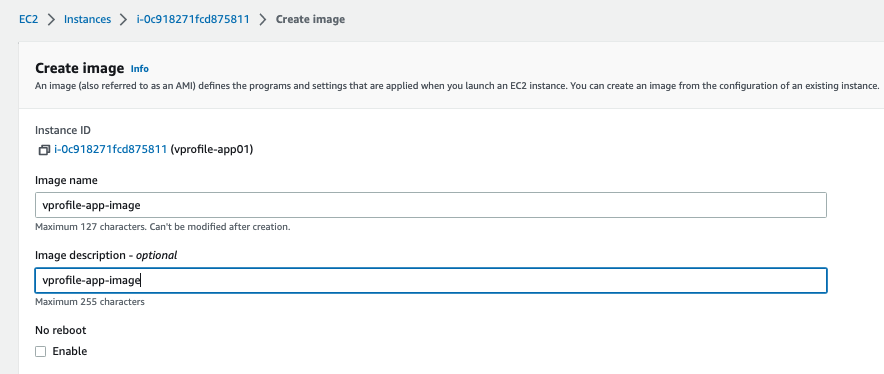
[](https://github.com/rumeysakdogan/AWS_DevOps_Projects/blob/main/Project-3%3A%20Lift%20and%20Shift%20Application%20Workload%20to%20AWS/images/record-for-elb.png)

* Lets check our application using our DNS. We can securely connect to our application!



### Step-10: Configure AutoScaling Group for Application Instances

* We will create an AMI from our App Instance.

[](https://github.com/rumeysakdogan/AWS_DevOps_Projects/blob/main/Project-3%3A%20Lift%20and%20Shift%20Application%20Workload%20to%20AWS/images/vprofile-ami.png)

* Next we will create a Launch template using the AMI created in above step for our ASG.

Name: vprofile-app-LT

AMI: vprofile-app-image

InstanceType: t2.micro

IAM Profile: vprofile-artifact-storage-role

SecGrp: vprofile-app-SG

KeyPair: vprofile-prod-key

* Our Launch template is ready, now we can create our ASG.

Name: vprofile-app-ASG

ELB healthcheck

Add ELB

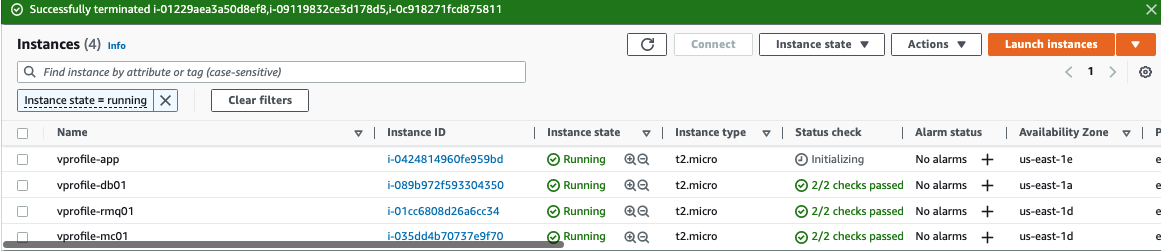
Min:1

Desired:2

Max:4

Target Tracking-CPU Utilization 50

* If we terminate any instances we will see ASG will create a new one using LT that we created.

[](https://github.com/rumeysakdogan/AWS_DevOps_Projects/blob/main/Project-3%3A%20Lift%20and%20Shift%20Application%20Workload%20to%20AWS/images/asg-provisioning-instance.png)

### Step-11: Clean-up

* Delete all resources we created to avoid any charges from AWS.