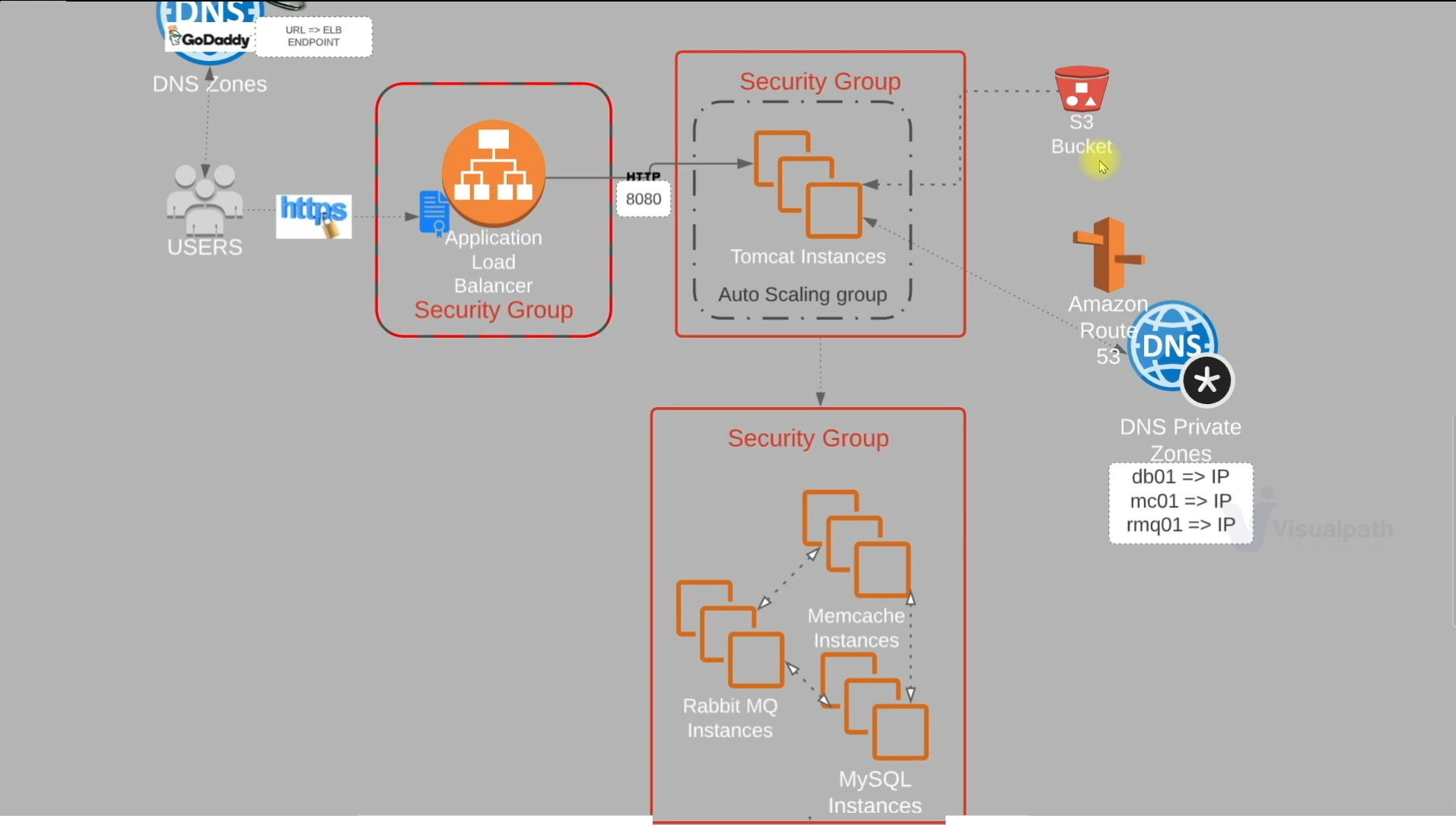
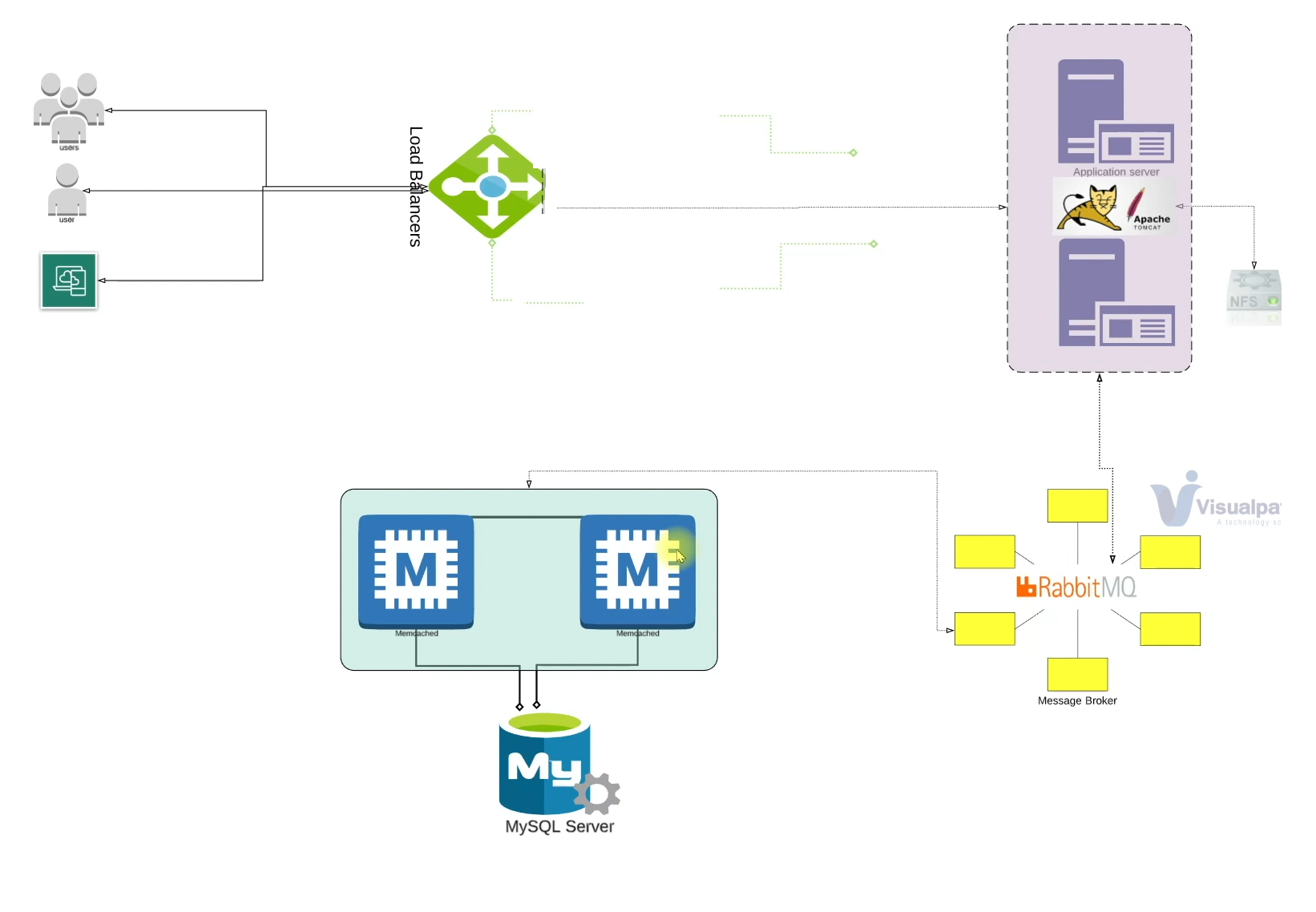
AWS Multi-tier web application hosting

This is a multi-tier web application hosting project in AWS cloud. We used four EC2 Virtual machines, First VM is to host Apache tomcat server with application code in it, second VM is to place a database server, third VM will have Memcached running in it and the last VM will have Rabbit MQ which is the queuing system in between App and DB/Cache layer. A domain satzwebio was registered with the domain provider(Godaddy), a load balancer is used to get traffic from users and point to app servers.

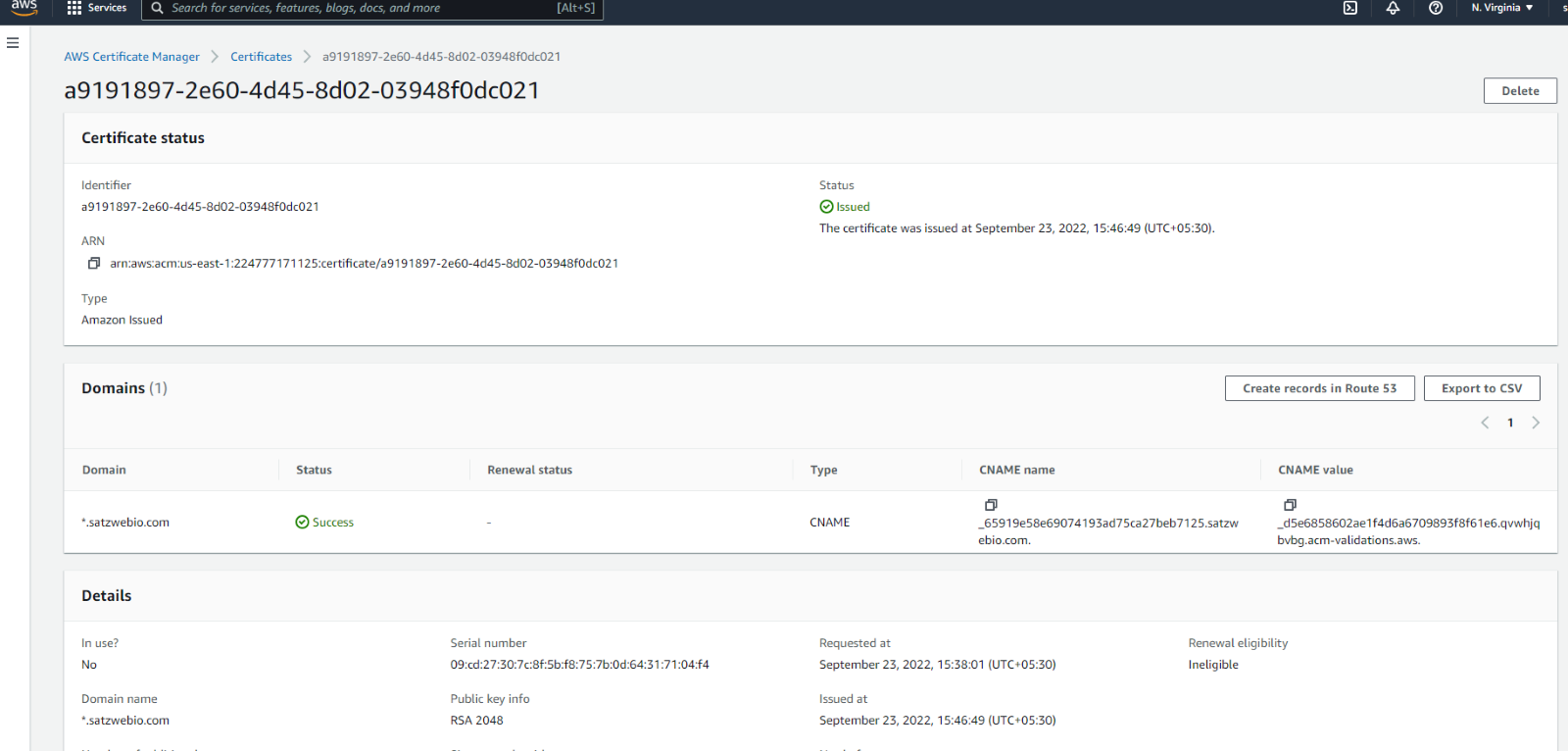
A S3 bucket is created to store artifacts of Application code.





1. Request a public certificate;

Create a public certificate with DNS validation,



1. Create 3 security groups

1. Load balancer security group -

Allow Http traffic in port 80 and Https traffic in port 443 for both ipv4 and ipv6.

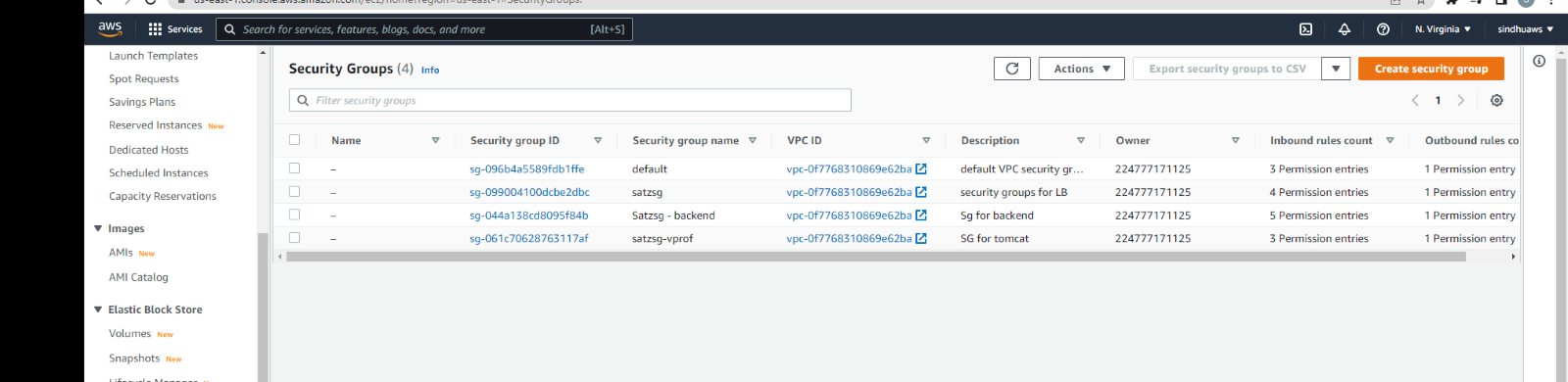
2. App server security group -

Allow traffic only from LB security group on port 8080.

3. Backend security group

Allow traffic from app security group only on ports 3306, 5672, 11211;

Allow trafiic on all ports for the own security group itself.

Allow ssh(22) for MyIP to login via putty.  


1. Create Key pairs

Create a EC2 key pair in pem format.

1. Launch instances with user data

1. Create a EC2 instance for Database. (centos 7, t2.micro, add backedn sec group)

Add user data from mydbsql file and launch instance.

login to DB instance "ssh -i vprofile-prof-key.pem [centos@54.196.151.252](mailto:centos@54.196.151.252)"

Check Mariadb is up "systemctl status mariadb"

mysql -u root -p

show databases;

use accounts

show tables;

2. Create a EC2 instance for Memcahced

Login into it

systemctl status memcached

check the ports - "ss -tunpl | grep 11211"

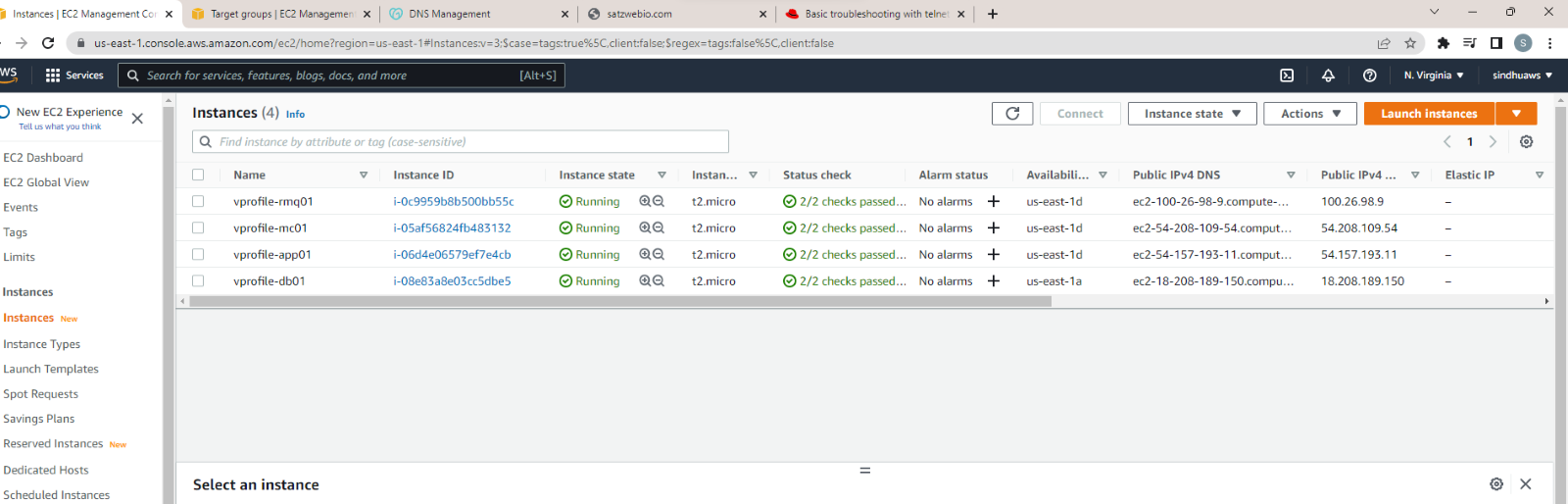
3. Create a EC2 instance for rabbitmq-server

systemctl status rabbitmq-server

4. Create a EC2 instance for App01

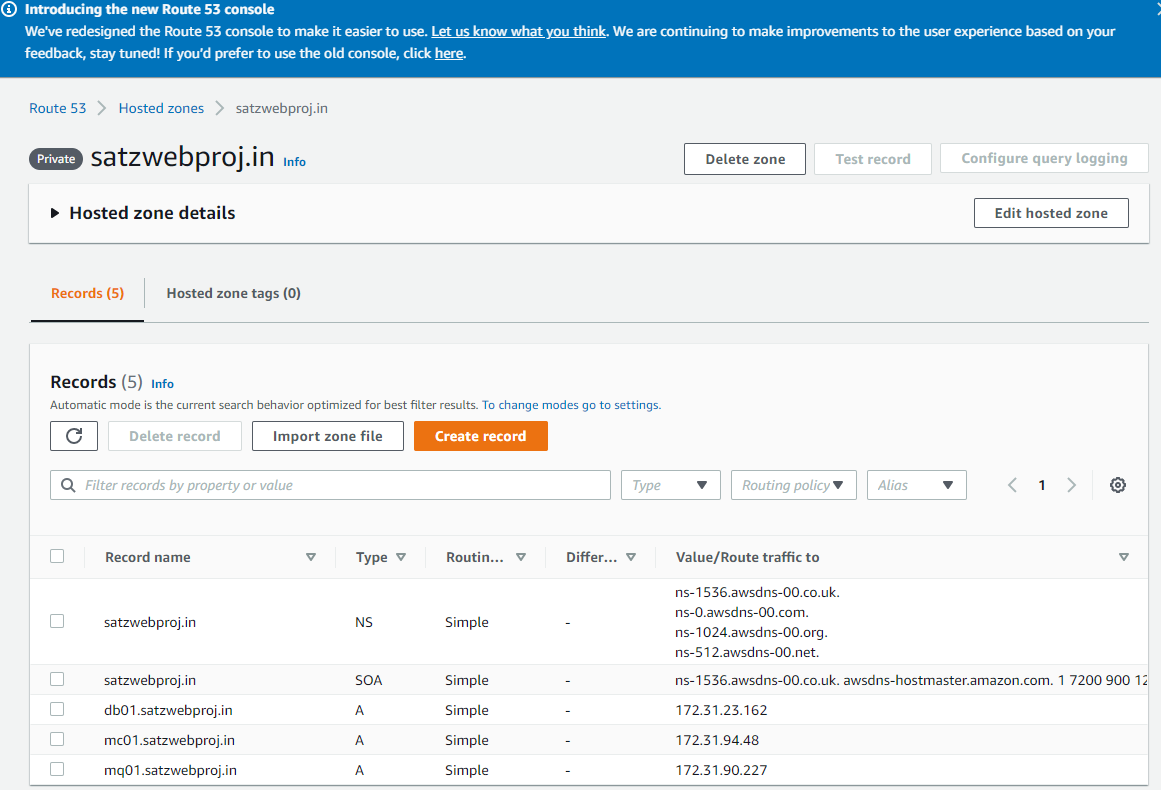
OS: Ubuntu 18 free tier

Update user data with the content present in tomcat\_ubuntu.sh file not tomcat.sh



1. Update IP to name mapping in route 53

Create route 53 hosted zone for the selected VPC.

Create 3 simple records like, db01, mc01 and mq01 and provide the corresponding private Ip Addresses.  


1. Build application from source code

Go to the source code and update application.properties file in src/main/resources.

update db01 with db01.satzwebproj.in; Do this similar to mc01 and mq01.

Cd to the repository which has pom file and run "mvn install" it will generate artifacts

Make sure below:

[root@ip-172-31-23-162 etc]# java -version

openjdk version "1.8.0\_345"

OpenJDK Runtime Environment (build 1.8.0\_345-b01)

OpenJDK 64-Bit Server VM (build 25.345-b01, mixed mode)

[root@ip-172-31-23-162 etc]# mvn -version

Apache Maven 3.0.5 (Red Hat 3.0.5-17)

Maven home: /usr/share/maven

Java version: 1.8.0\_345, vendor: Red Hat, Inc.

Java home: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.345.b01-1.el7\_9.x86\_64/jre

Default locale: en\_US, platform encoding: UTF-8

OS name: "linux", version: "3.10.0-1160.76.1.el7.x86\_64", arch: "amd64", family: "unix"

[root@ip-172-31-23-162 etc]#

1. Upload to S3 bucket

aws s3 mb s3://satzweb-storage

Go to the target directory(Which is the one maven generated) and perform below

aws s3 cp vprofile-v2.war s3://satzweb-storage/vprofile-v2.war

PS E:\devops\aws\liftshift\target> aws s3 ls s3://satzweb-storage

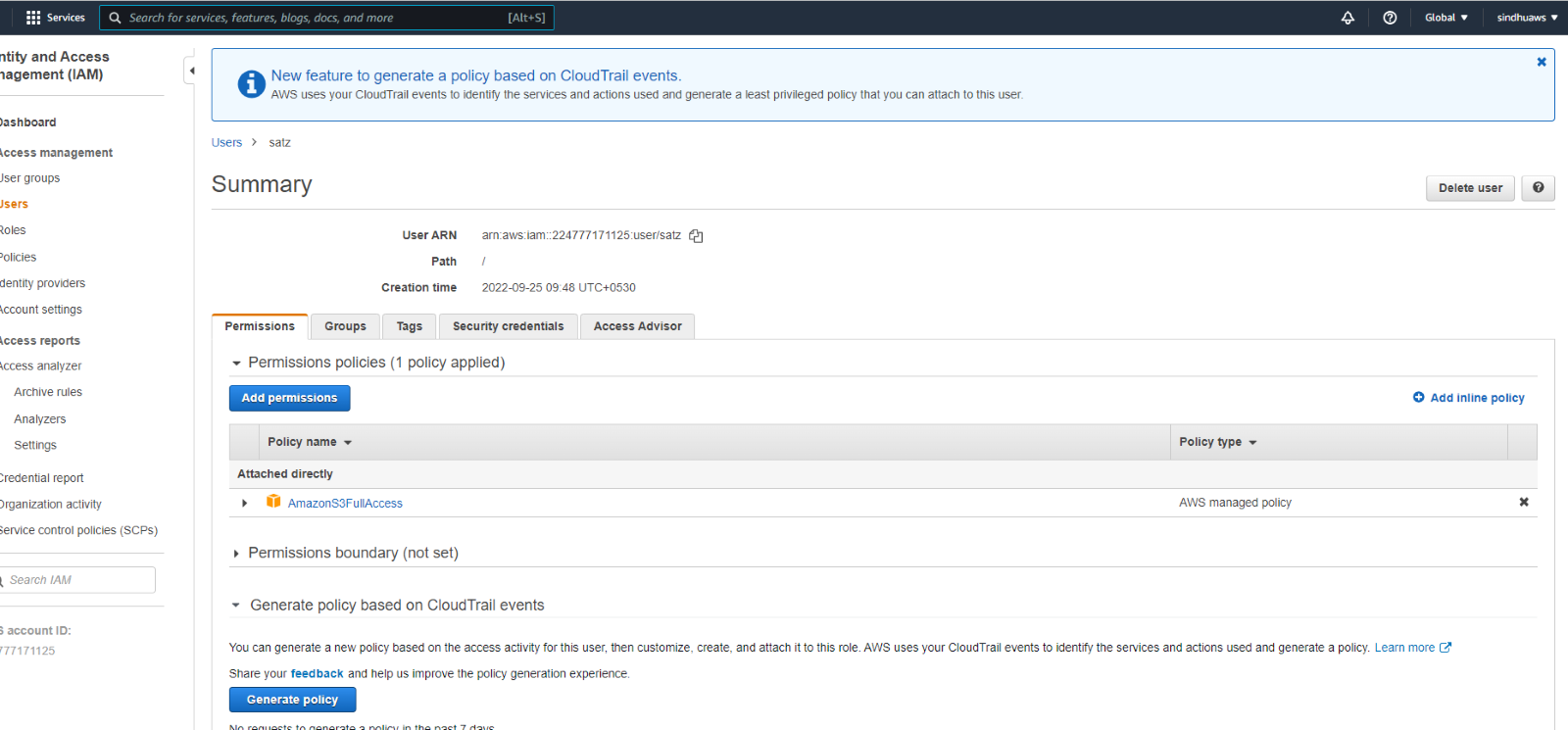
2022-09-25 16:18:45 48451801 vprofile-v2.war



1. Create a role and assign it to Ec2 App instance:

Go to IAM and create a EC2 role with Amazon S3 Full access.

Update the new role in EC2 app instance



1. Download artifact to Tomcat EC2 instance

ssh into app server

ssh -i vprofile-prof-key.pem [ubuntu@54.157.193.11](mailto:ubuntu@54.157.193.11)

switch to root and go to /var/lib/tomcat8/webapps

stop tomcat service - systemctl stop tomcat8

rm -rf ROOT/

apt install awscli

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

cd /tmp/

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

systemctl start tomcat8

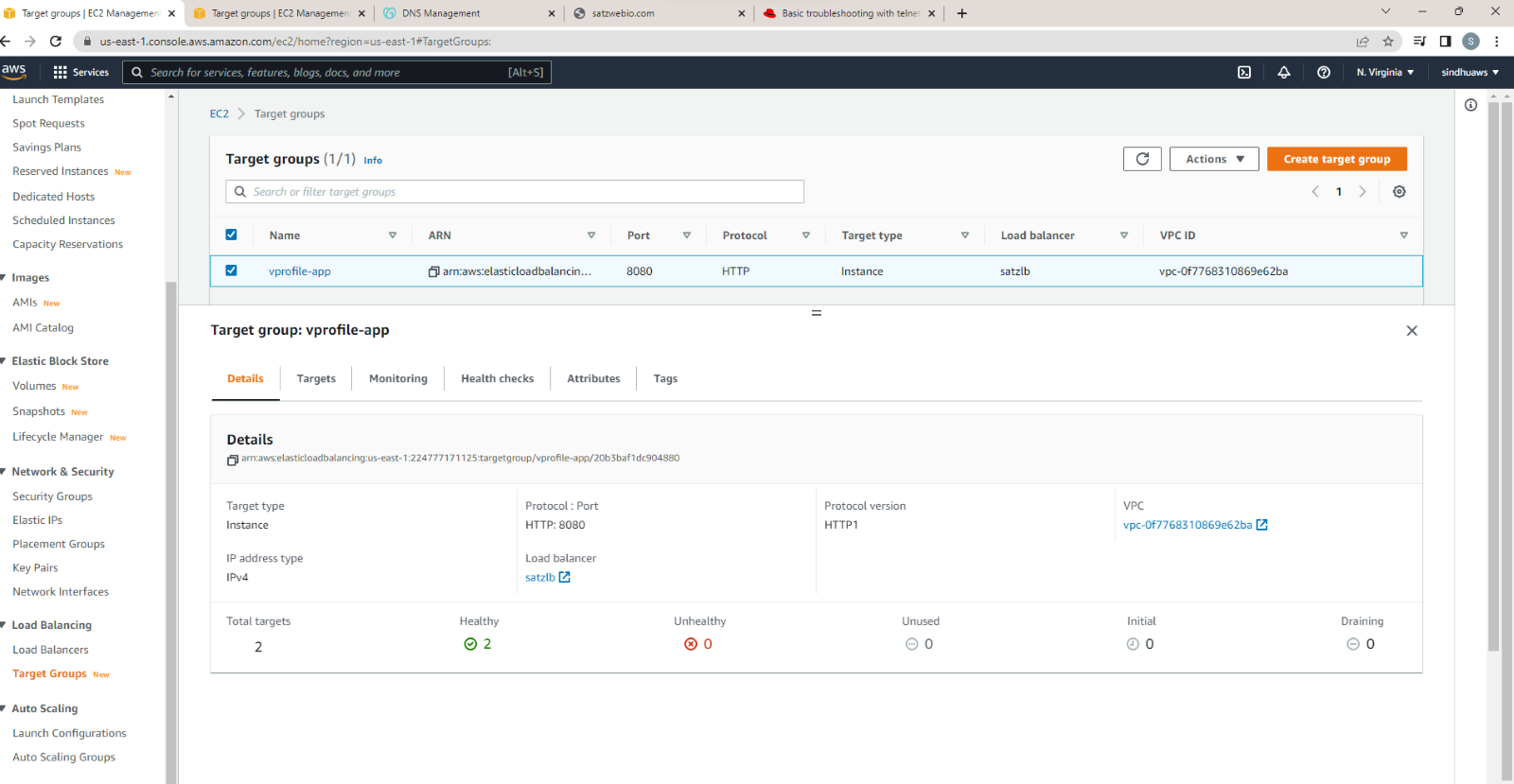
cd /var/lib/tomcat8/webapps/ROOT/WEB-INF/classes#

telnet db01.satzwebproj.in 3306

(Just checking if app is able to connect to db, try similar for all hosts)

1. Create Target group:

For port 8080 and add app instance

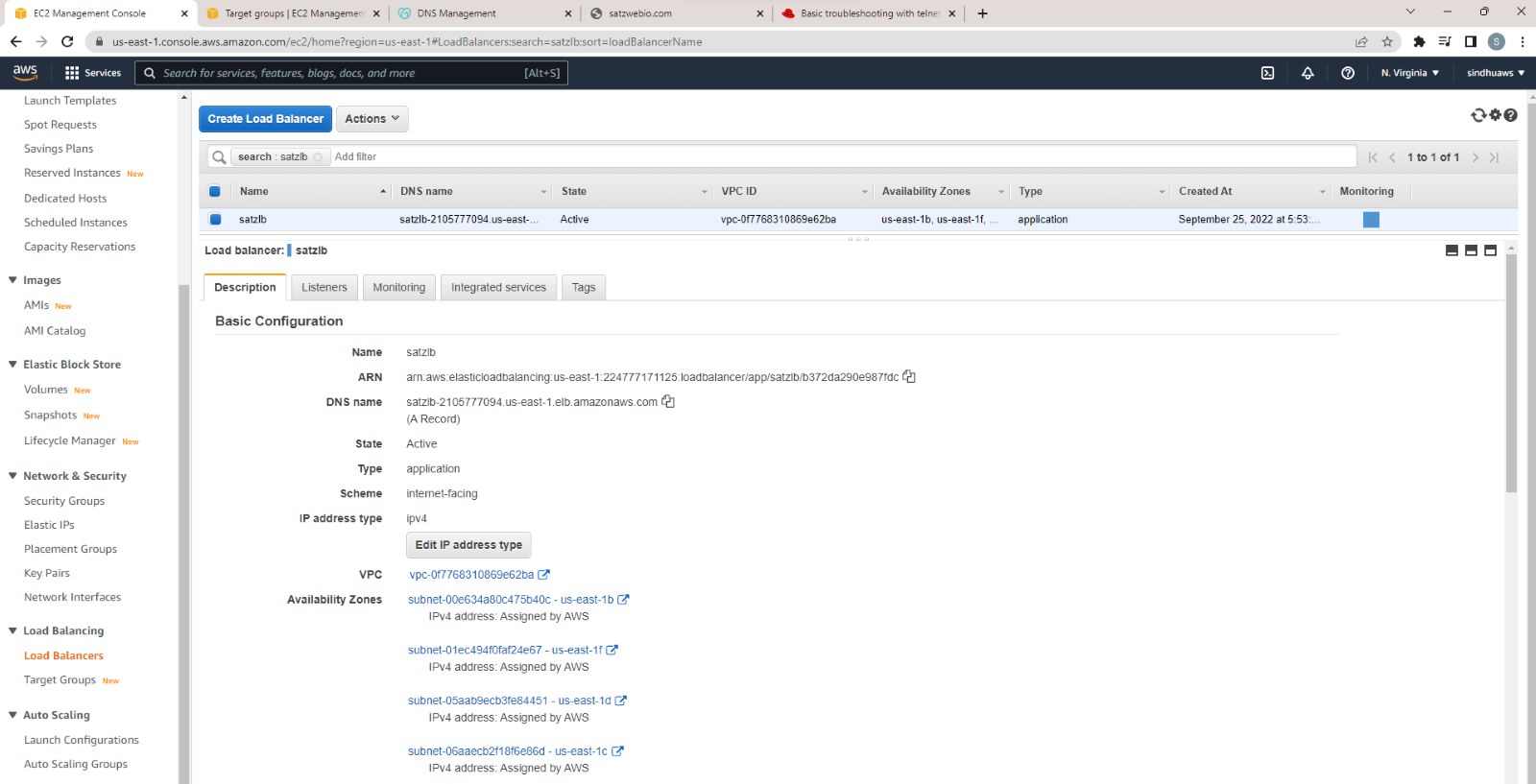


1. Setup ELB with HTTPS [Cert from Amazon certificate Manager]

Create a load balancer internet facing and IPV4, select all the regions, add the target group.

Choose the LB security group, add both http and https as listeners.

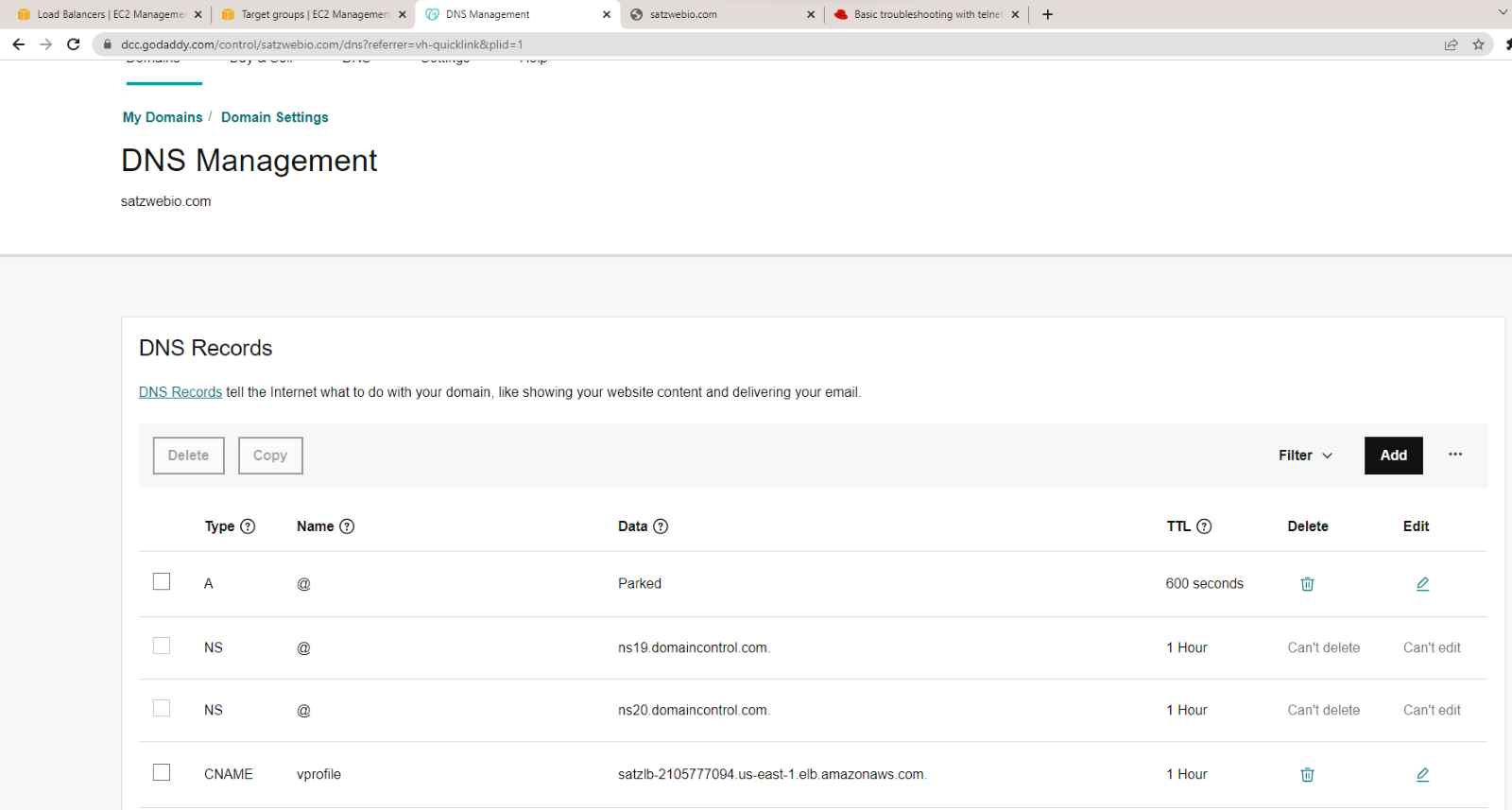
Add the cert that we created.



12. Map ELB endpoint to website name in Godaddy DNS.

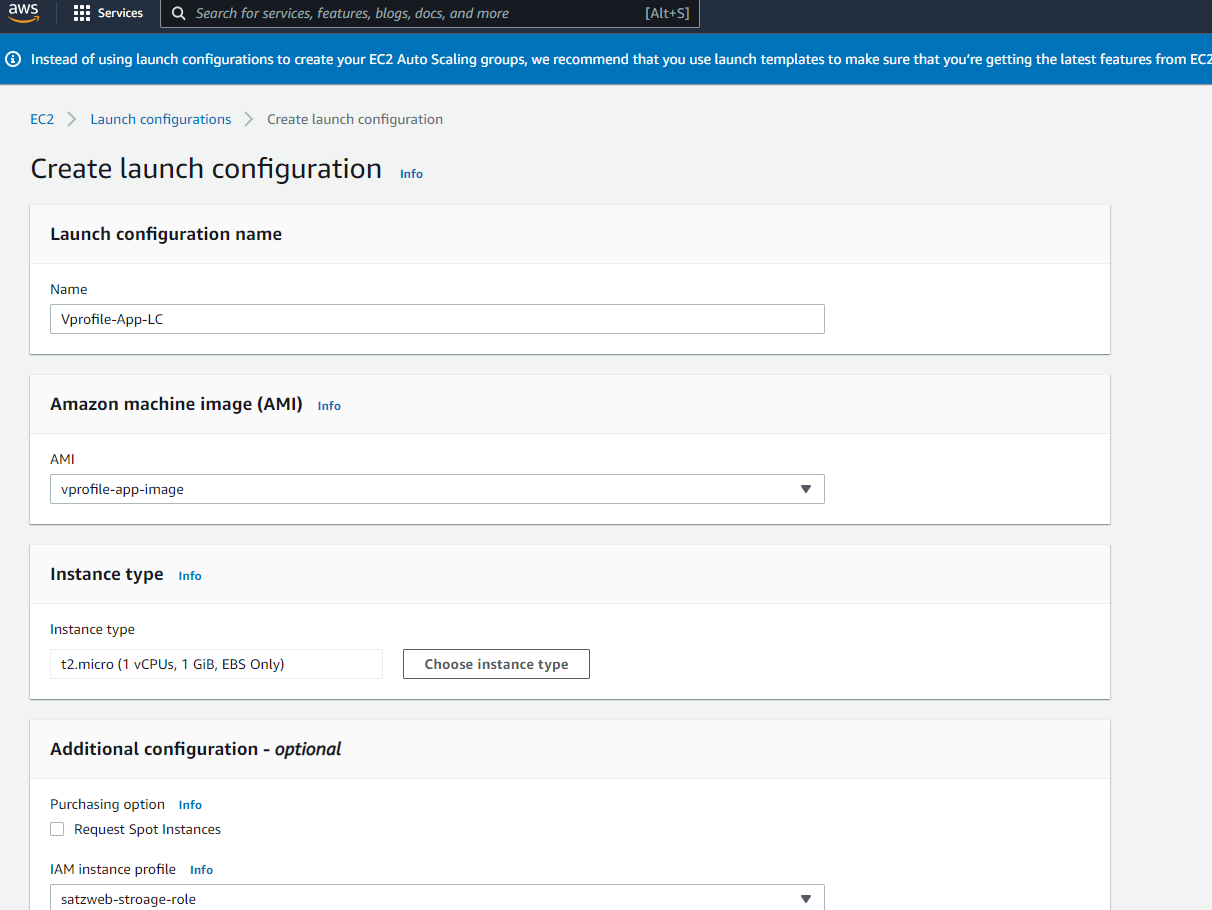
Copy DNS name from LB and go to godaddy, add a new Cname records like xyz.

so xyz.satzwebio.com

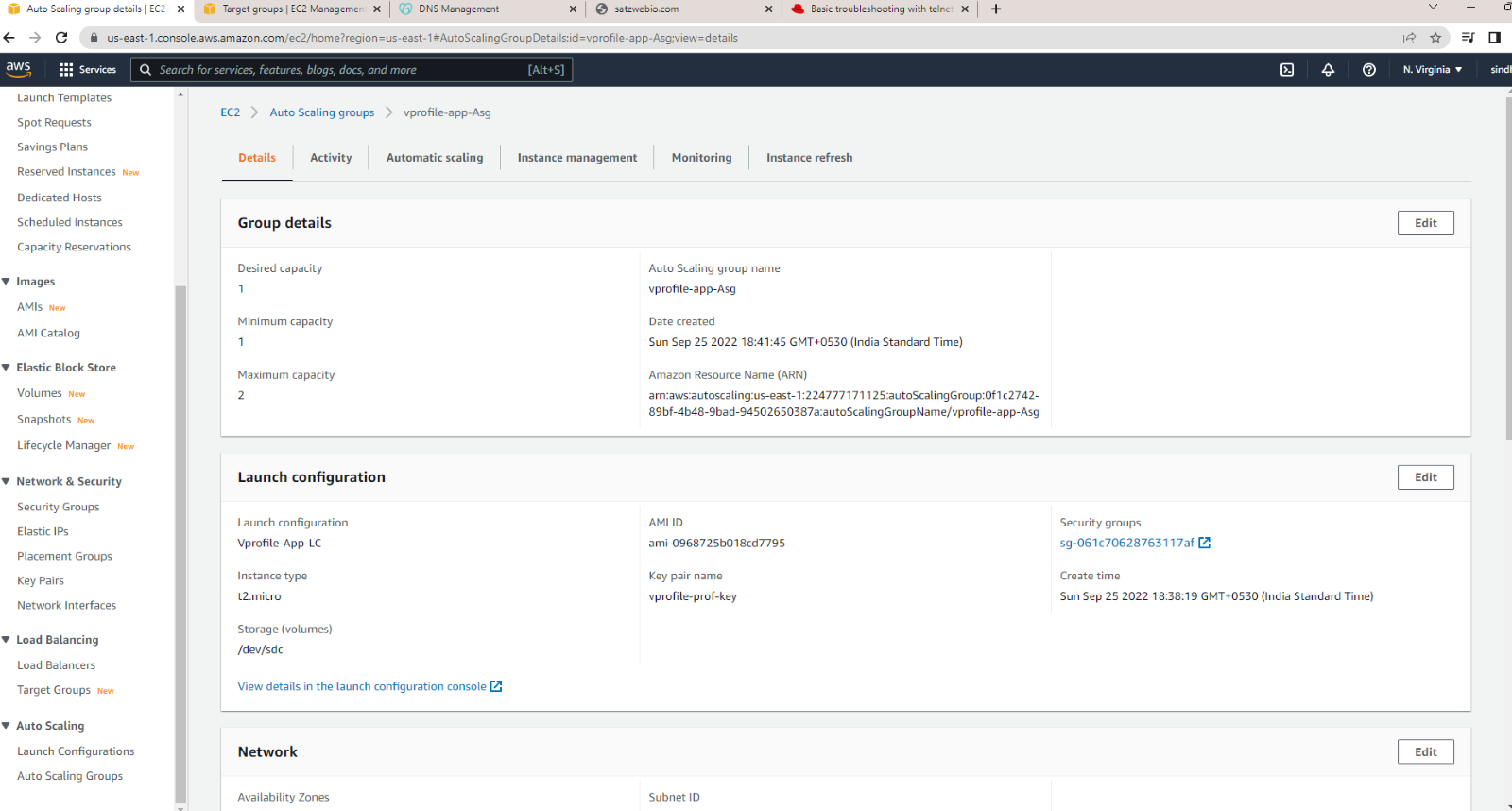


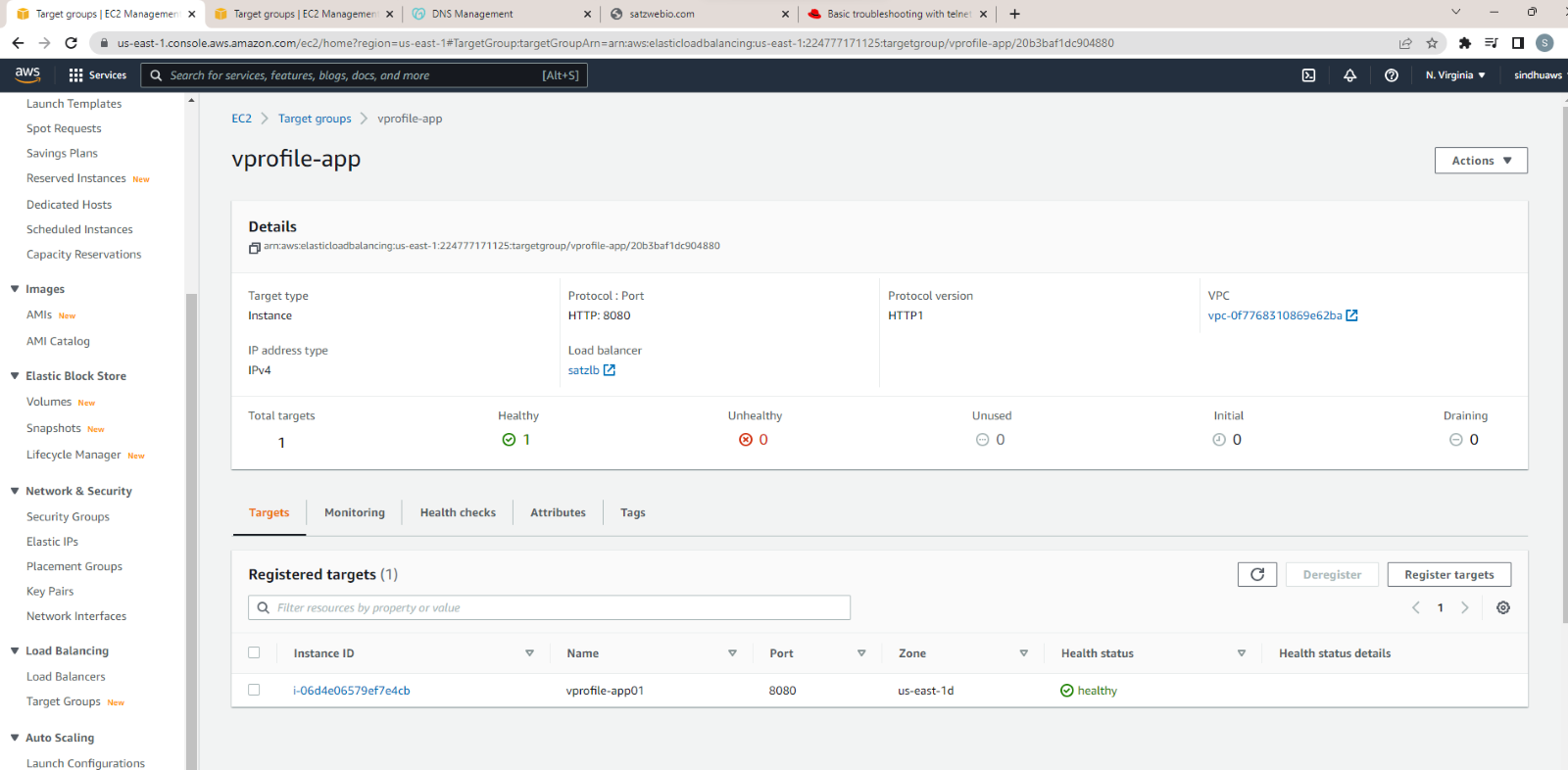
13. Create Image for the App instance

14. Create a launch configuration with the AMI, existing key pair.



15. Create a Autoscaling group and setup target policy,





16. Verify the application

