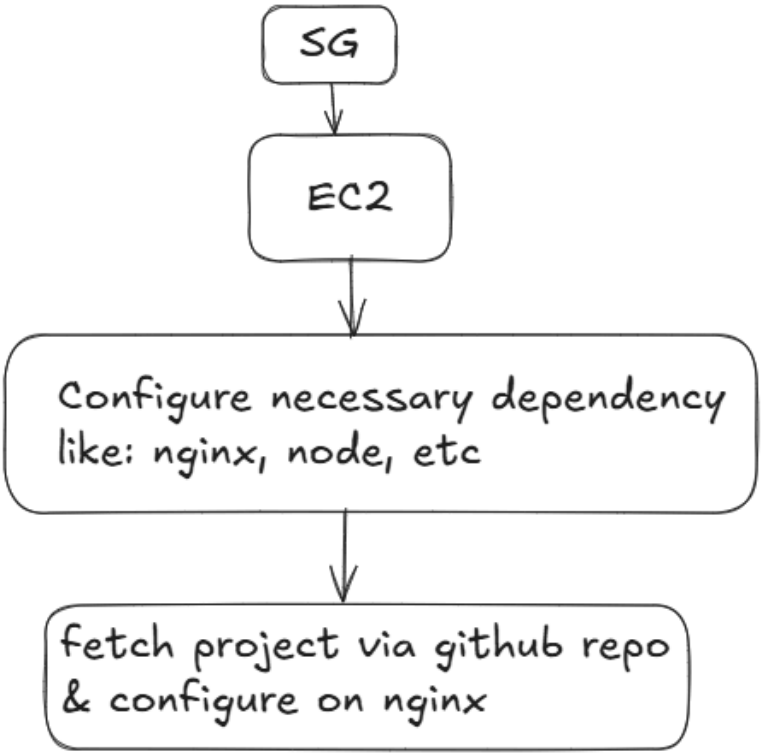
Project #01

AWS

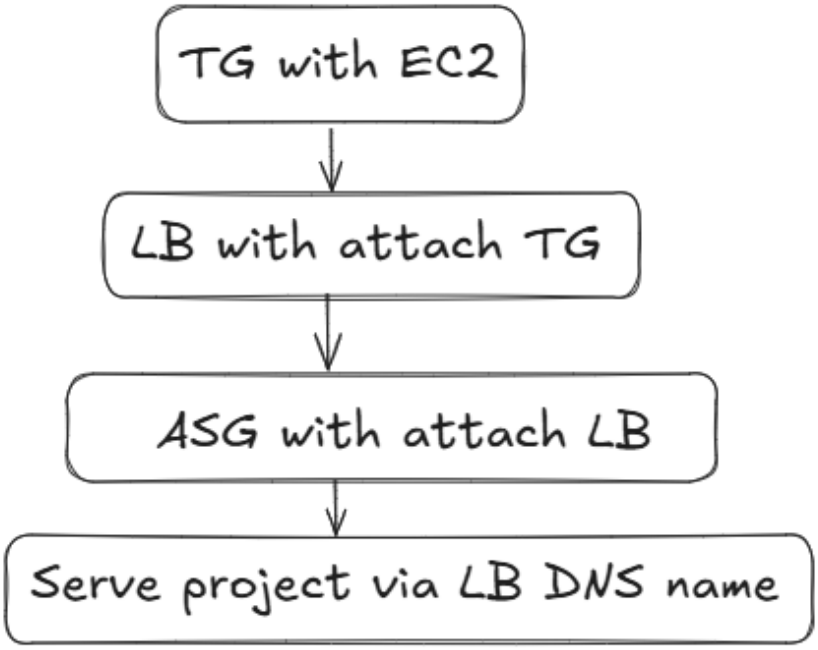
Deploy a website on AWS

Project Overview:

Step 1: Deploy a Vue js project on AWS EC2.



Step 2: Add load balancer and auto scaling on this project.



Project Overview:

Cover AWS service name lists:

1. SG → Security Groups
2. EC2 → Elastic Compute Cloud
3. TG → Target Groups
4. LB → Load Balancer (use ALB)
5. ASG → Auto Scaling Groups

Configuration Dependency:

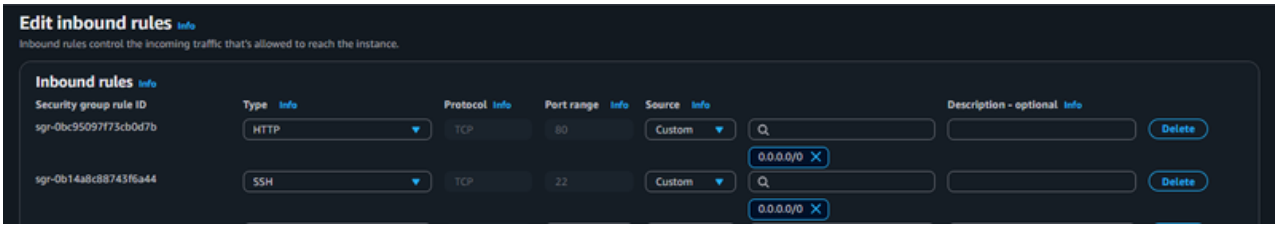
1. Nginx
2. Node js
3. npm → package manager

Step 2 is not mandatory for deploying a website on AWS EC2.

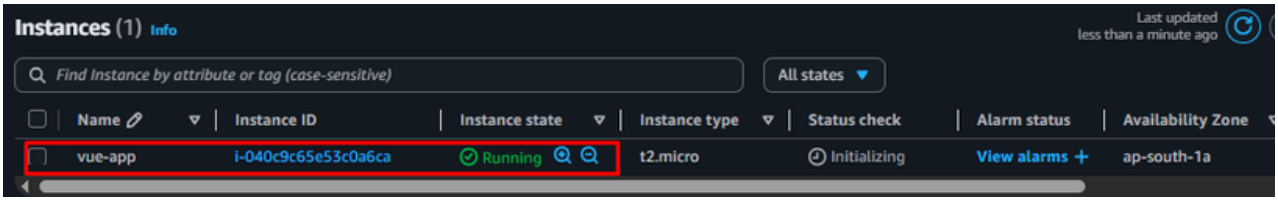
I have previously learned the AWS services mentioned in Step 2. I included Step 2 for my practice purposes. You don't need to follow this if configuring a Load Balancer is not required for your project.

Step 1: Deploy on AWS EC2

First, create a Security Groups (SG) that allows SSH and HTTP access. If you plan to use SSL, allow HTTPS as well. In my case, I didn't configure SSL, so I ignored HTTPS.



Now, launch an EC2 instance using this SG. If you want access through SSH, don't forget to generate a key pair key.



I used the following repository for this project.

https://github.com/alamgirweb11/meals-app

Step 1: Deploy on AWS EC2

These are the configuration file I set up.

```
# update packages
sudo apt update && sudo apt upgrade -y

# install nginx & check status
sudo apt install nginx -y
systemctl status nginx

# Install Node.js & npm (latest LTS)
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E
bash -

# this command installs Node.js & npm both
sudo apt install -y nodejs build-essential

# check version
node -v
npm -v

# clone GitHub repo
git clone https://github.com/alamgirweb11/meals-app.git

# rename file, for meaningful based on EC2 name
mv meals-app vue-app

# locate vue app
cd vue-app

# create Environment file for api setup
nano .env
# attach this end point and save .env file
VITE_API_BASE_URL =
'https://www.themealdb.com/api/json/v1/1/'
```

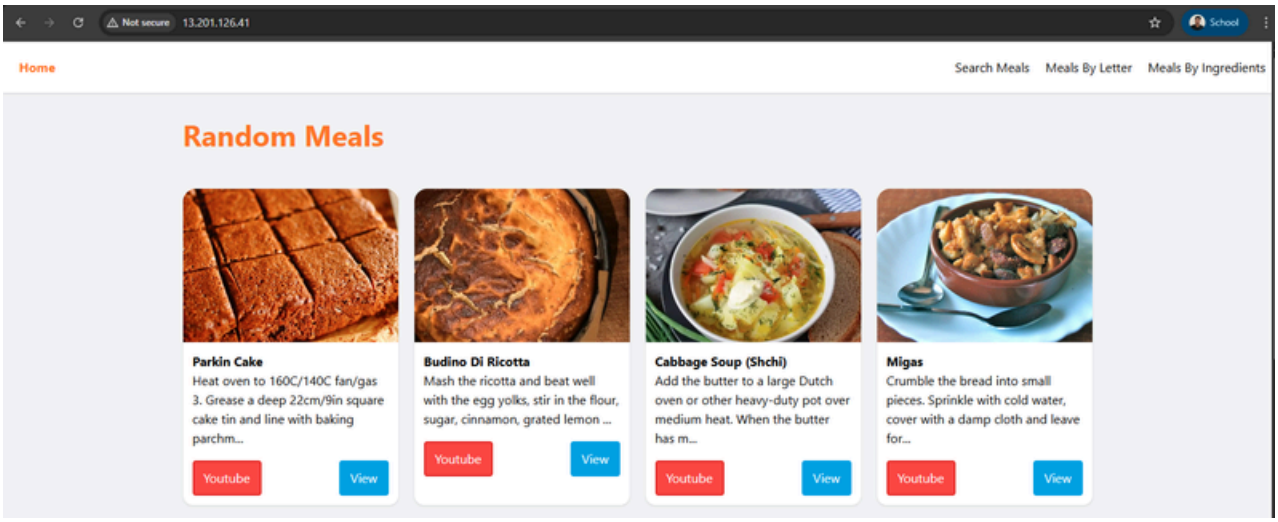
Step 1: Deploy on AWS EC2

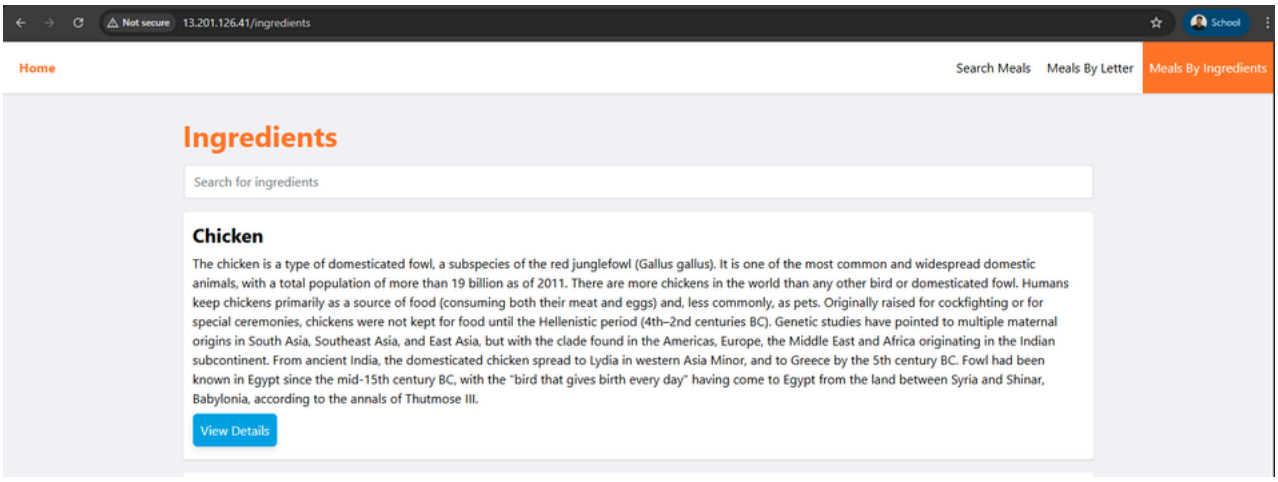
These are the configuration file I set up.

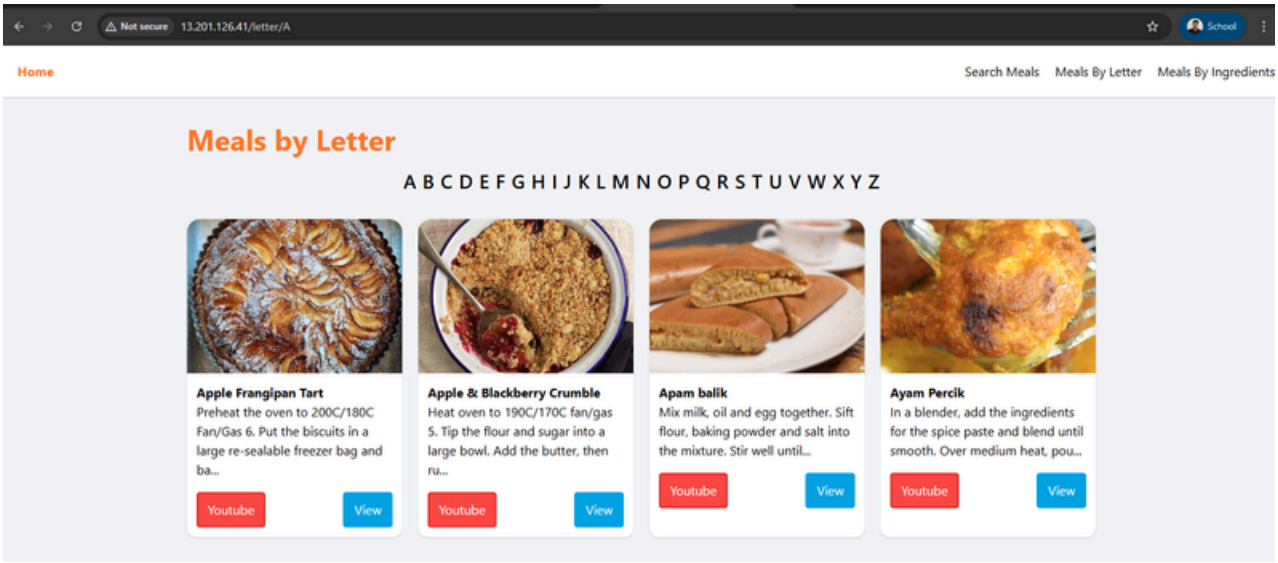
```
# now install npm and build
# --legacy-peer-deps use for ignoring old version package installations
npm install --legacy-peer-deps
npm run build

# configure app in nginx
sudo nano /etc/nginx/sites-available/vue-app

# add config file & save
server {
  listen 80;
  server_name _;

  root /var/www/vue-app/dist;
  index index.html;

  location / {
    try_files $uri $uri/ /index.html;
  }
}

# remove default config file
sudo rm /etc/nginx/sites-enabled/default

# create folder & copy build files
sudo mkdir -p /var/www/vue-app
sudo cp -r ~/vue-app/dist /var/www/vue-app/
sudo cp -r ~/vue-app/dist /var/www/vue-app/

# enable config
sudo ln -s /etc/nginx/sites-available/vue-app /etc/nginx/sites-enabled/

# test nginx config
sudo nginx -t

# restart nginx
sudo systemctl restart nginx
```

Step 1: Deploy on AWS EC2

Successfully deploy our site on AWS EC2.



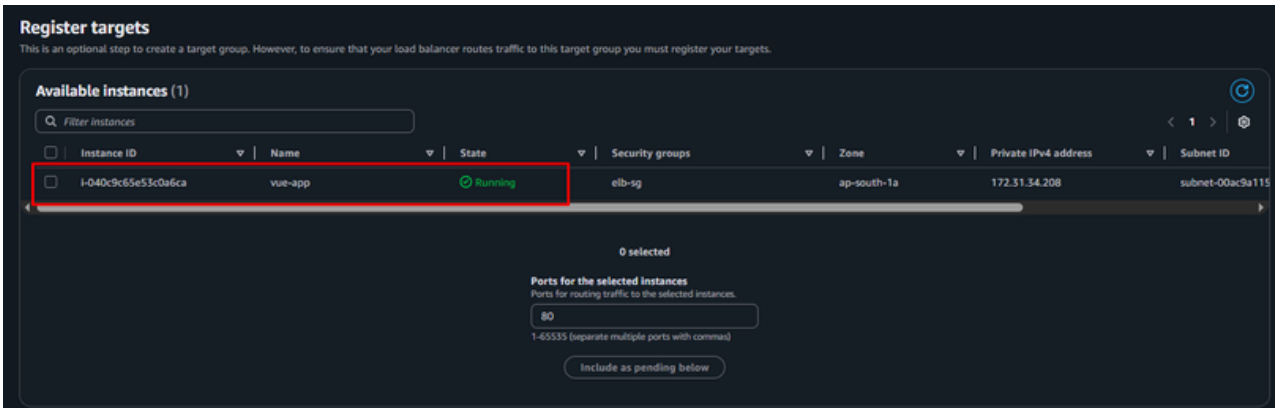


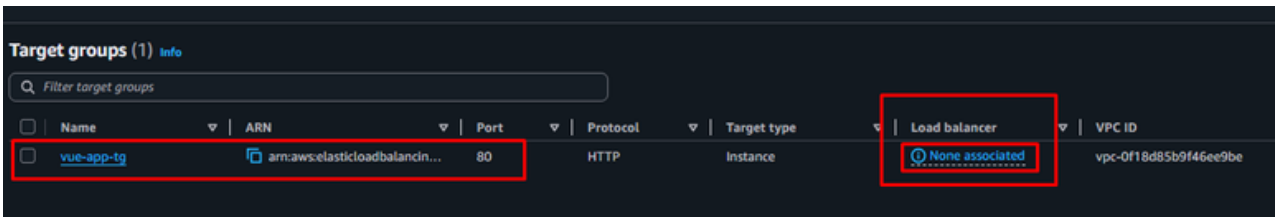


Step 1 Completed

Step 2: Load Blancer & Auto Scaling

For load balance & Auto scaling, first you need to create a Target Groups. Register targets here i select my EC2 instance. Here it is:

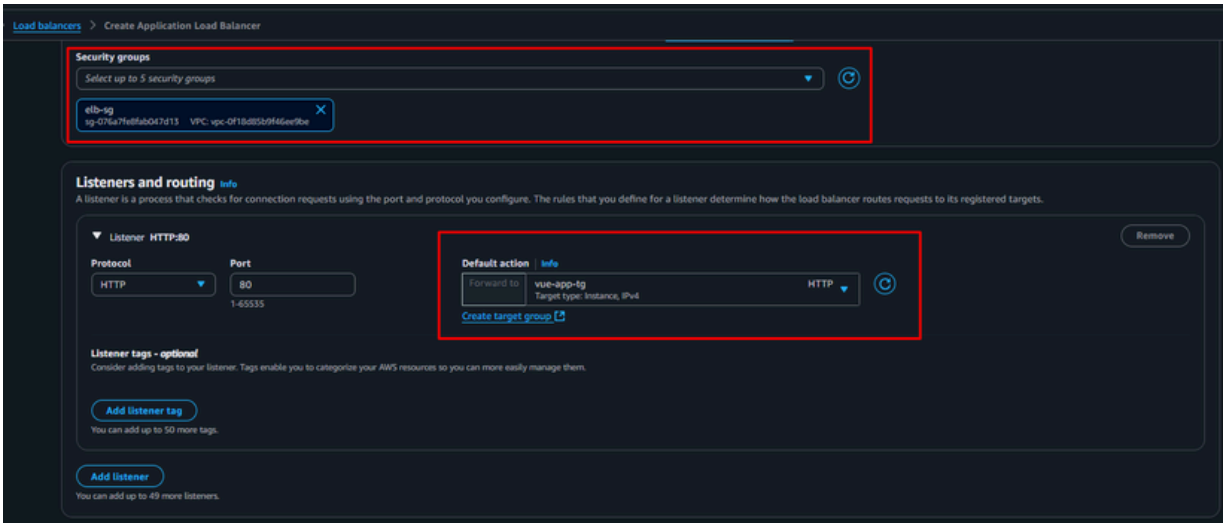




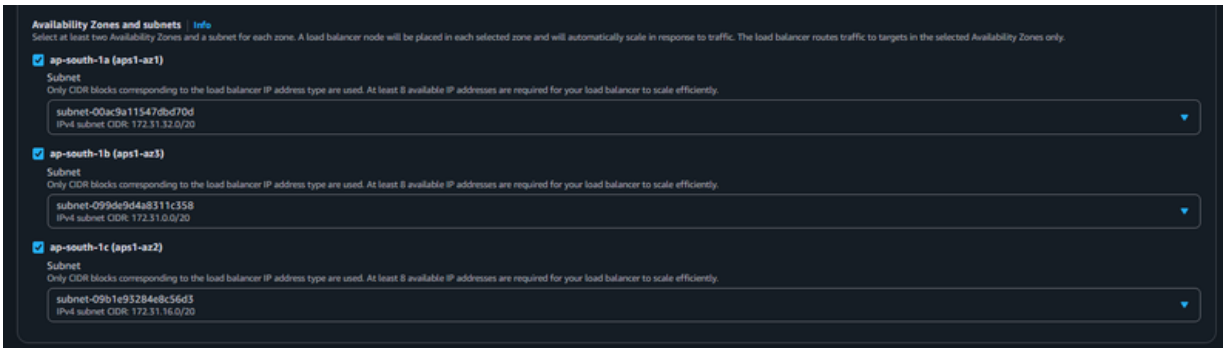
The target groups created successfully, but the load balancer is not yet configured. That's why it shows as not associated with a load balancer.

Step 2: Load Blancer & Auto Scaling

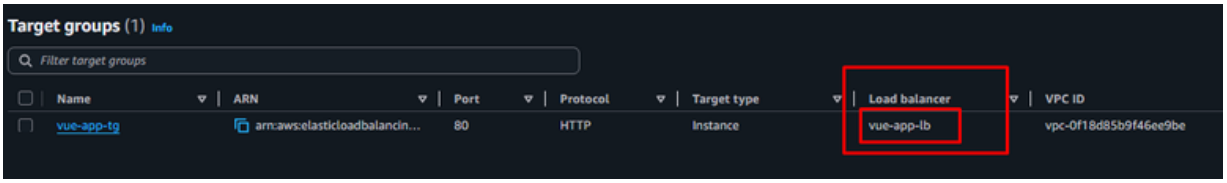
Now, created a load balancer with this TG.
When create a LB please ensure the SG are same as EC2 and select TG which you attach with TG.



Make sure to choose the Availability Zone (AZ).



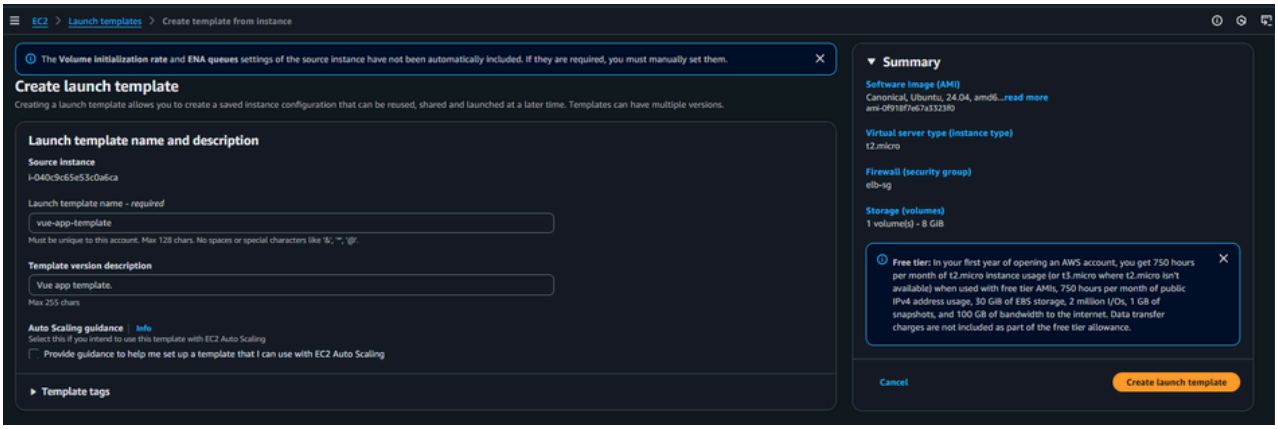
Now go to TG section and see the LB is located.



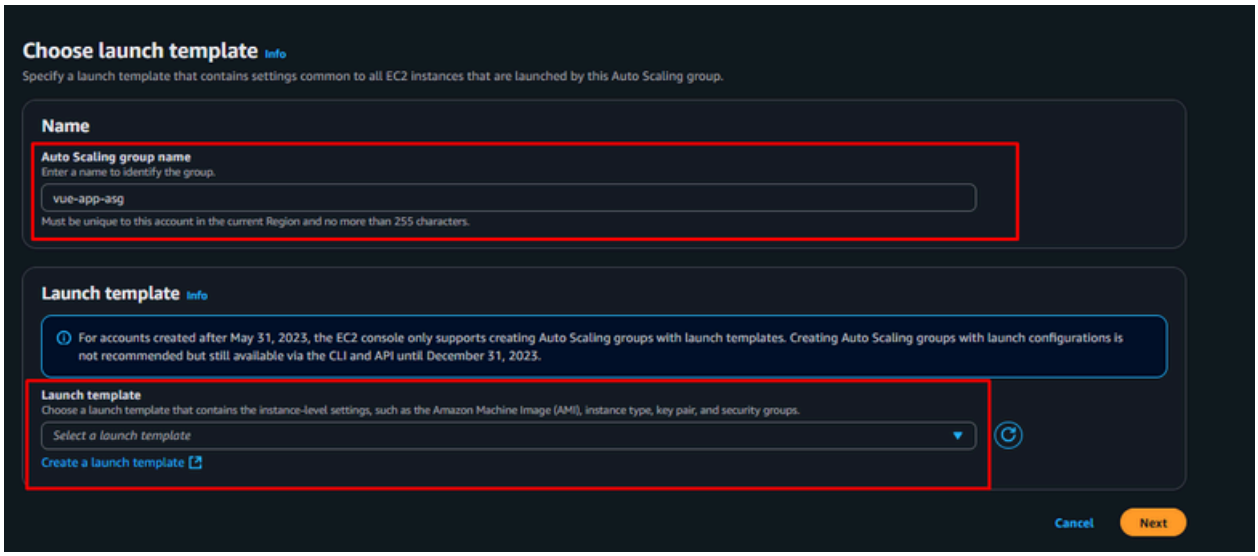
Step 2: Load Blancer & Auto Scaling

Now it’s time to create an Auto Scaling Group (ASG). To do this, go to the Auto Scaling Groups section and configure it based on your needs.

Before creating an ASG, you first need to create a launch template. In my case, I used my EC2 instance as the launch template.

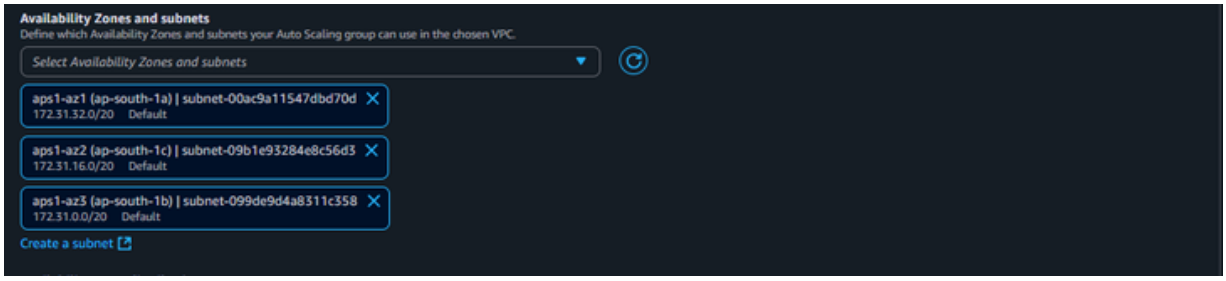


If you want to use a previous launch template, just choose it from the dropdown. Or, you can create a new one from here, as this option is also provided.

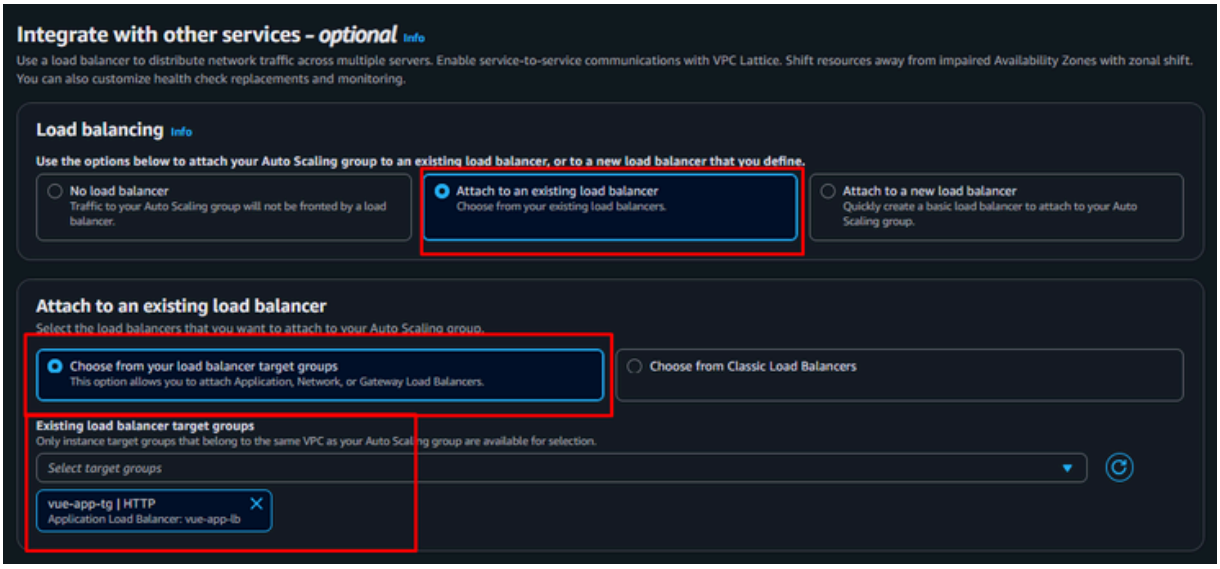


Step 2: Load Blancer & Auto Scaling

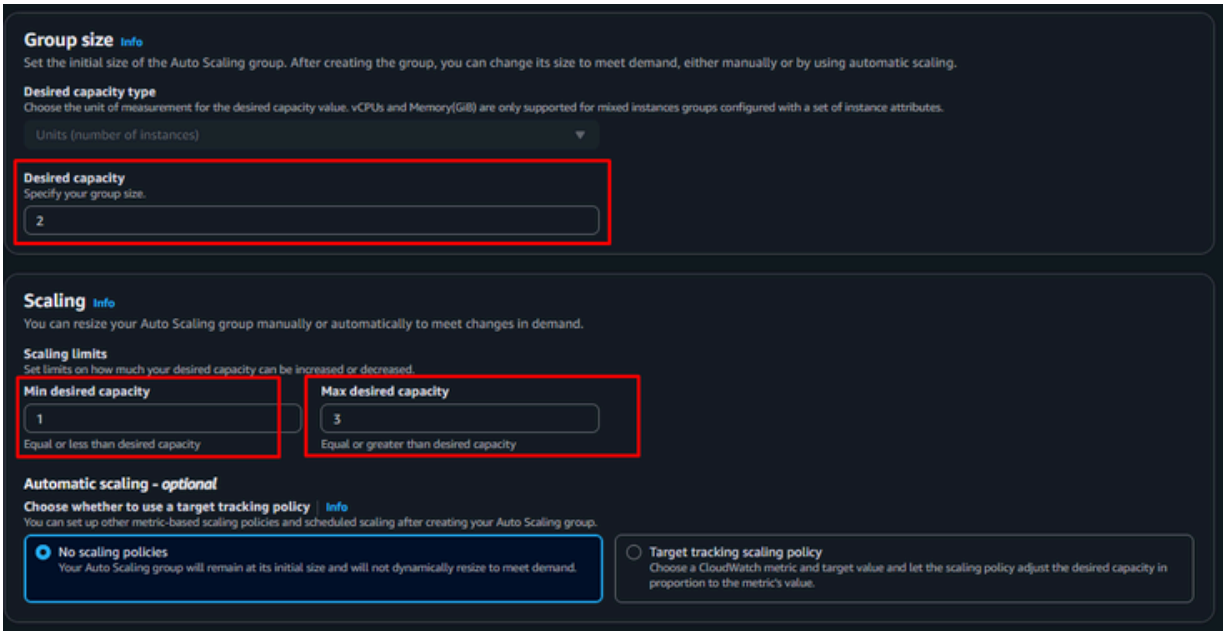
Don’t forget to select AZ when creating an ASG.



In attaching load balancer section, don’t forget to attach your created LB.

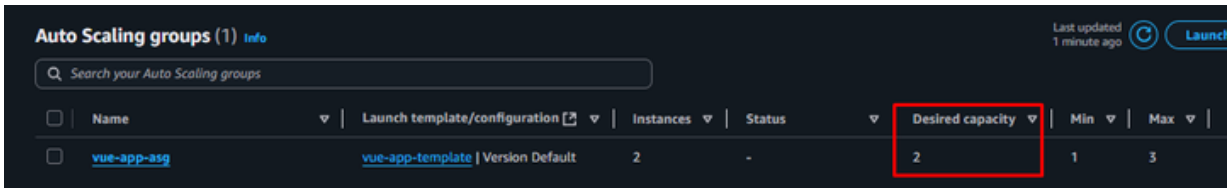


Select desired, min, max capacity it’s important.

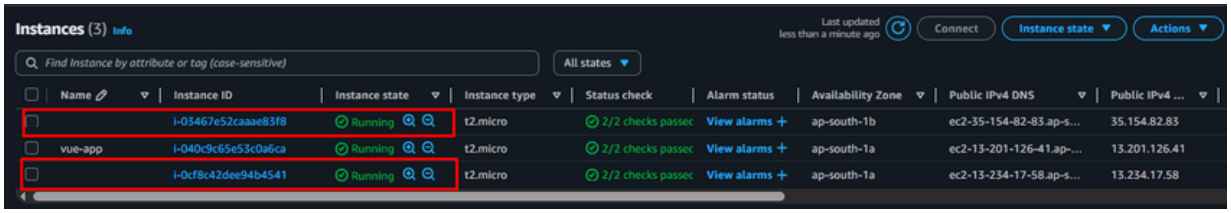


Step 2: Load Blancer & Auto Scaling

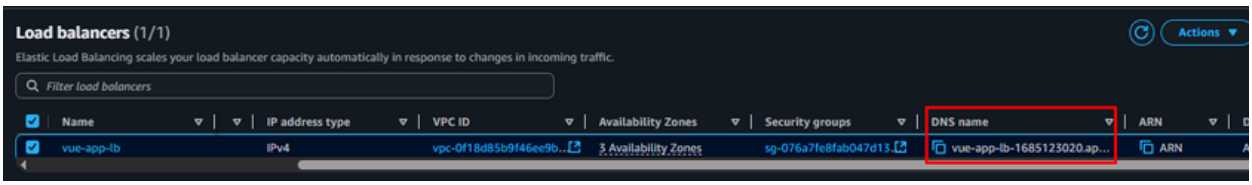
Our ASG successfully created, and instances were launched based on the desired capacity. Since I set it to 2, it launched 2 instances.



EC2 instances:



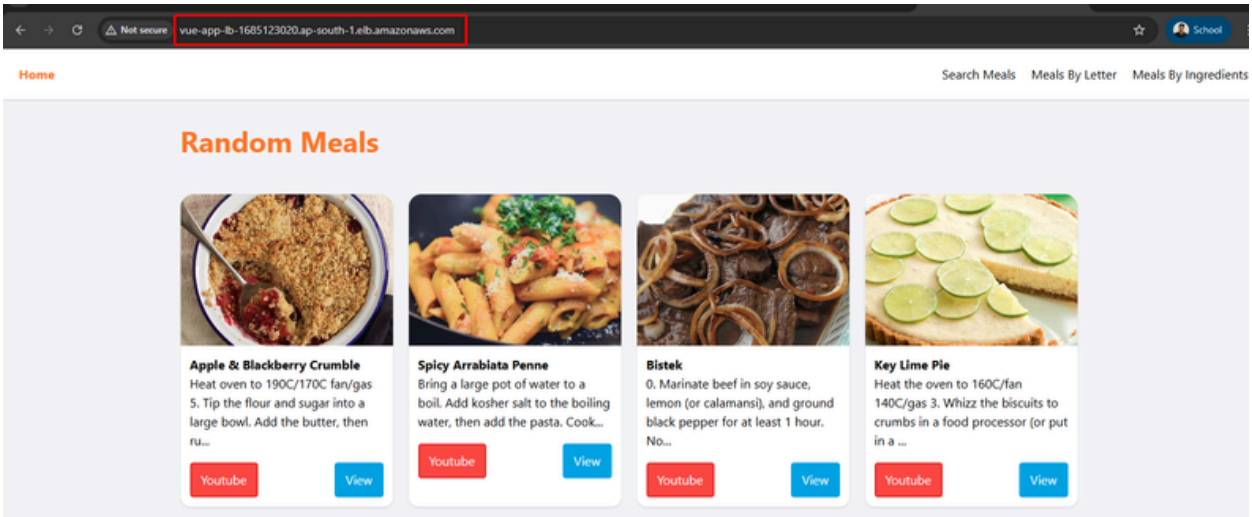
Here the LB now copy the DNS name and browse.

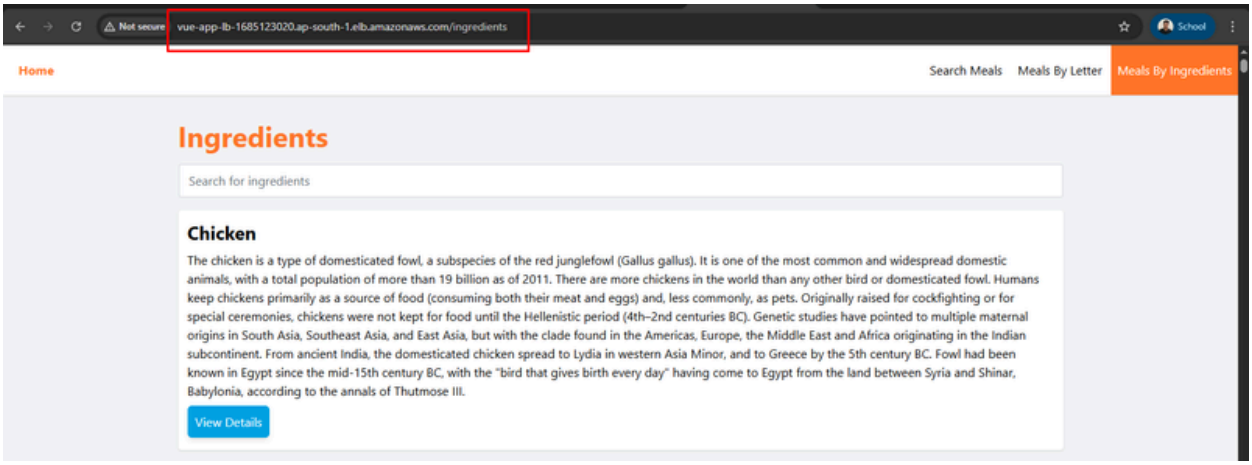


Step 2 Completed

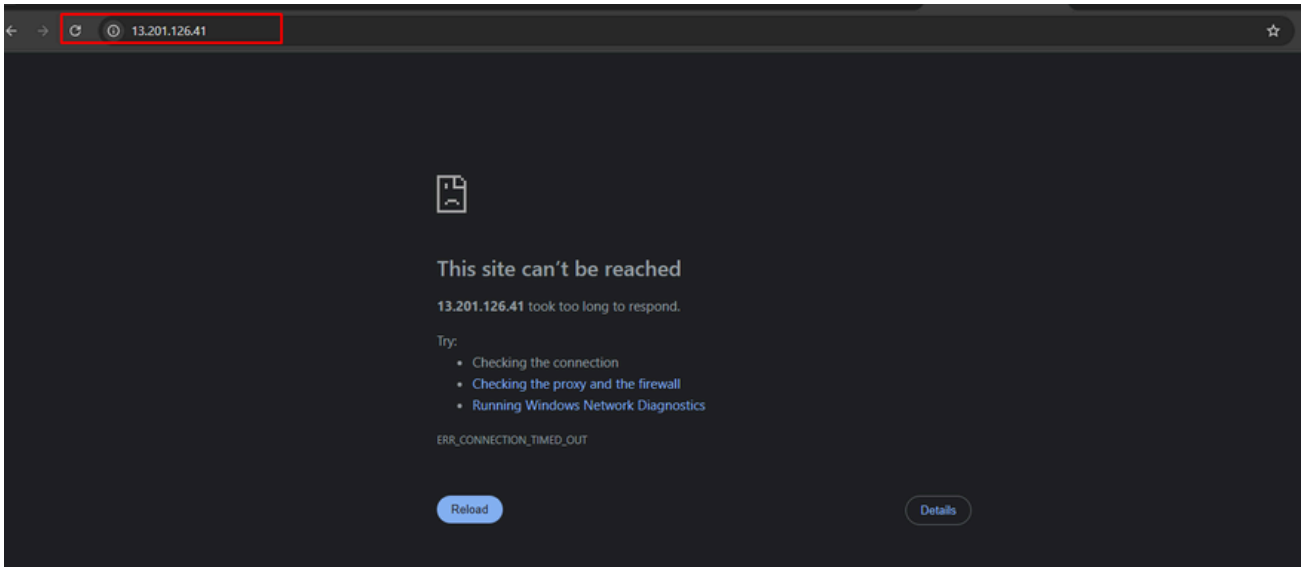
Step 2: Load Blancer & Auto Scaling

Browse the site using load balancer DNS name.





This is the EC2 public IP, but it is not accessible from the browser.



Thank You

Stay Connect:

/in/alamgirweb11

/alamgirweb11