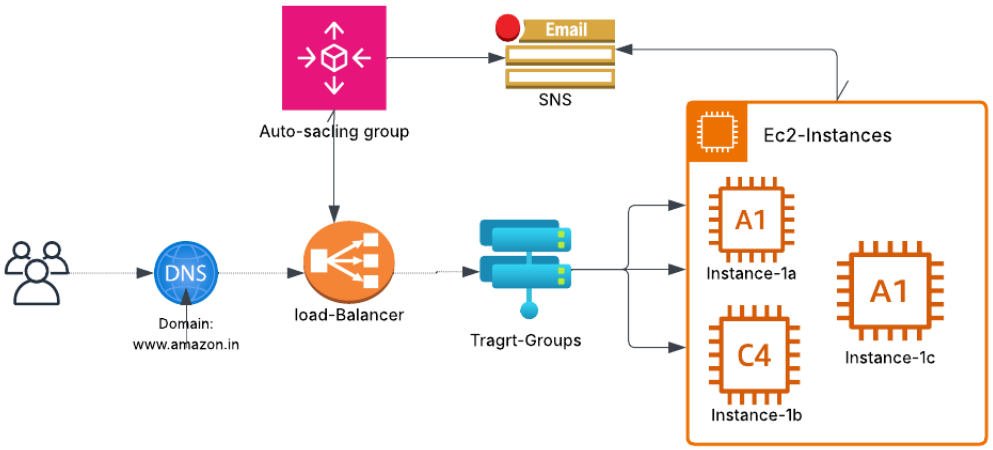
www.linkedin.com/in/arun-kumar-akula

Deploying a Scalable Food Delivery App on AWS

Overview :

This guide walks through deploying a Food Delivery Web Application (similar to Swiggy or Zomato) on AWS using EC2, **Application Load Balancer** (ALB), **Auto Scaling Group** (ASG), and **SNS** for alerts. This architecture ensures the application is always available, highly responsive under heavy load, and can automatically scale to handle customer traffic. ideal for real-time food ordering platforms.

Architecture Overview:



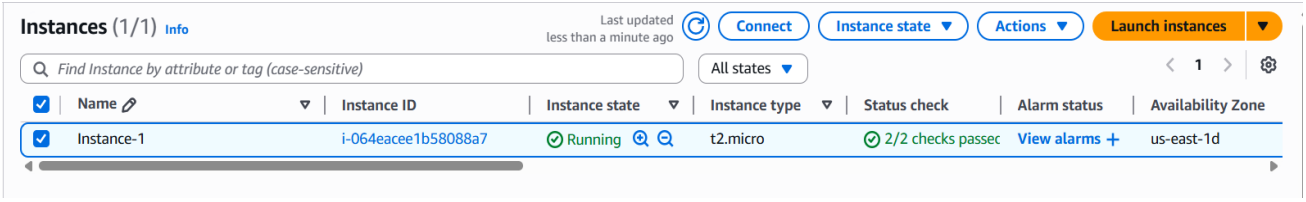
- Users access the app using a domain name (e.g., www.amazon.in).
- Requests go to the Application Load Balancer (ALB), which evenly distributes traffic.
- The Target Group monitors and routes traffic only to healthy EC2 instances.
- Multiple EC2 Instances (e.g., Instance-1a, 1b, 1c) run in different Availability Zones for high availability.
- The Auto Scaling Group (ASG) adds or removes EC2 instances automatically based on CPU load.
- SNS (Simple Notification Service) sends email alerts during scale-in/scale-out events.

Steps to Deploy the Food App:

Step 1: Launch EC2 Instance & Deploy the App

- Navigate to EC2 > Instance > Launch Instances

- Create instance named: Instance-1(in us-east-1a).
- AMI: Amazon Linux 2 (or) ubuntu
- Type: t2.micro
- Key pair: create (or) select existing key-pair
- Security Group: SSH & HTTP



SSH into Each Instance and Deploy the App :

- Commands to Deploy the App

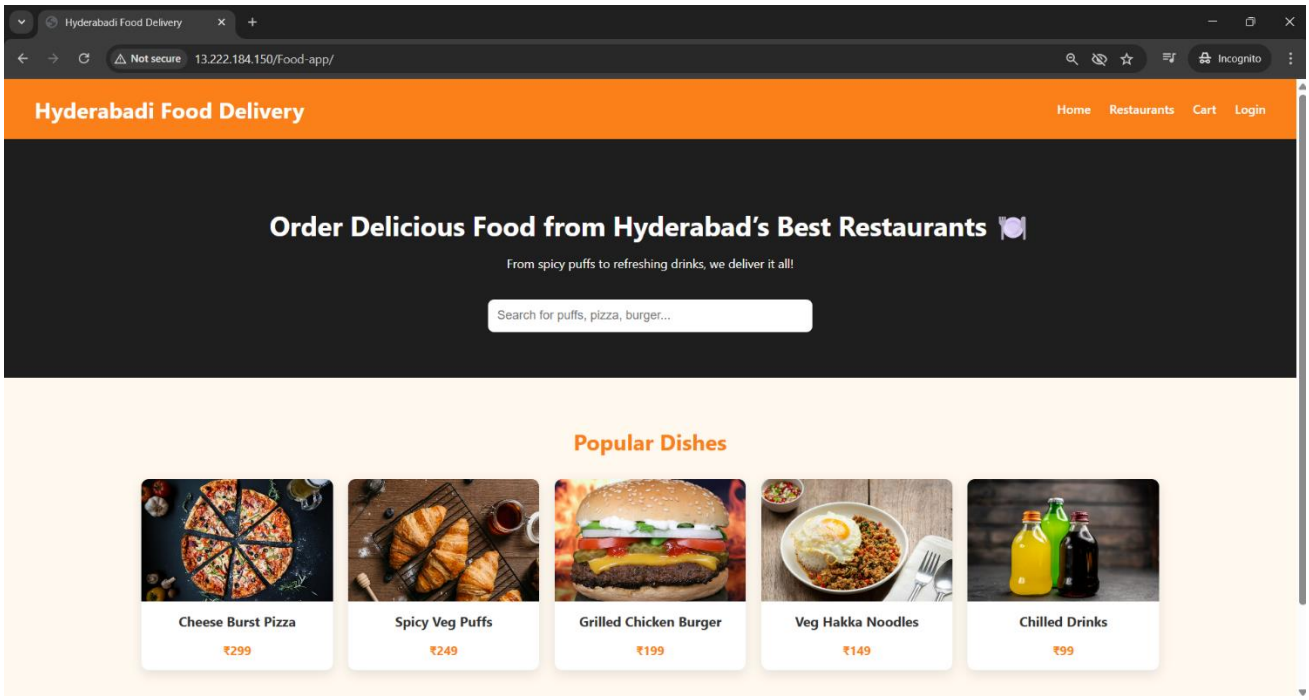
```
sudo yum update -y

sudo yum install httpd -y

nginx -v

sudo systemctl start httpd

sudo systemctl enable httpd

sudo git clone https://github.com/Arunkumarakula/Food-app.git /var/www/html/
```

```
root@ip-172-31-22-126:~#
root@ip-172-31-22-126:~# nginx -v
nginx version: nginx/1.24.0 (Ubuntu)
root@ip-172-31-22-126:~#
```

```
root@ip-172-31-22-126:~#
root@ip-172-31-22-126:~# cd /var/www/html/
root@ip-172-31-22-126:/var/www/html# ls
index.nginx-debian.html
root@ip-172-31-22-126:/var/www/html# rm -rf index.nginx-debian.html
root@ip-172-31-22-126:/var/www/html# ls
root@ip-172-31-22-126:/var/www/html#
root@ip-172-31-22-126:/var/www/html# git clone https://github.com/Arunkumarakula/Food-app.git
Cloning into 'Food-app'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (4/4), done.
root@ip-172-31-22-126:/var/www/html# ls
Food-app
root@ip-172-31-22-126:/var/www/html# cd Food-app/
root@ip-172-31-22-126:/var/www/html/Food-app# ls
index.html  styles.css
root@ip-172-31-22-126:/var/www/html/Food-app#
```

➢ Test the application using the public IP address in your browser:

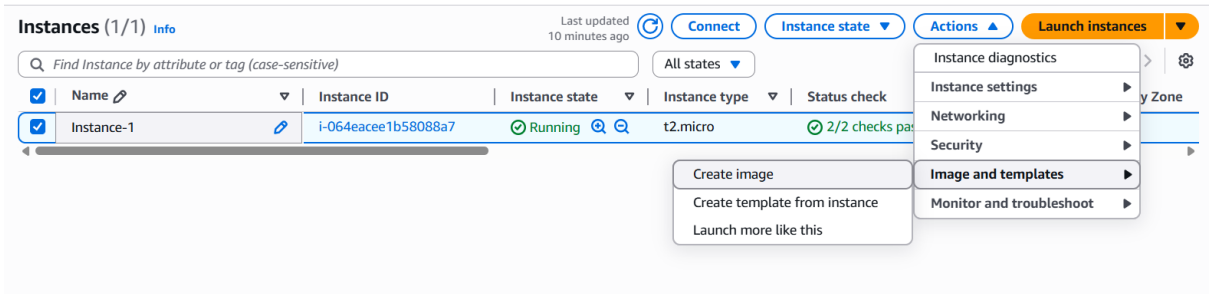
```
http://<your-ec2-public-ip>
```



Step 2: Create an EC2 Instance Using AMI For Load Balancer High Availability.

Steps to Create a New AMI from Your Existing EC2 Server

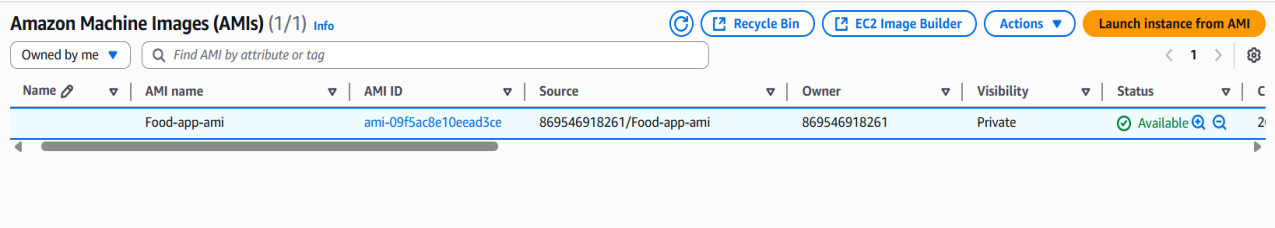
- Go to EC2 > Instances
- Select your running app instance (e.g: Server-1)
- Click Actions > Image and templates > Create image



Fill in:

- Image name: FoodApp-AMI
- Click Create Image

Wait a few minutes , go to **EC2 > AMIs** and make sure the AMI status shows **available**



Launch a New EC2 Instance from the New AMI

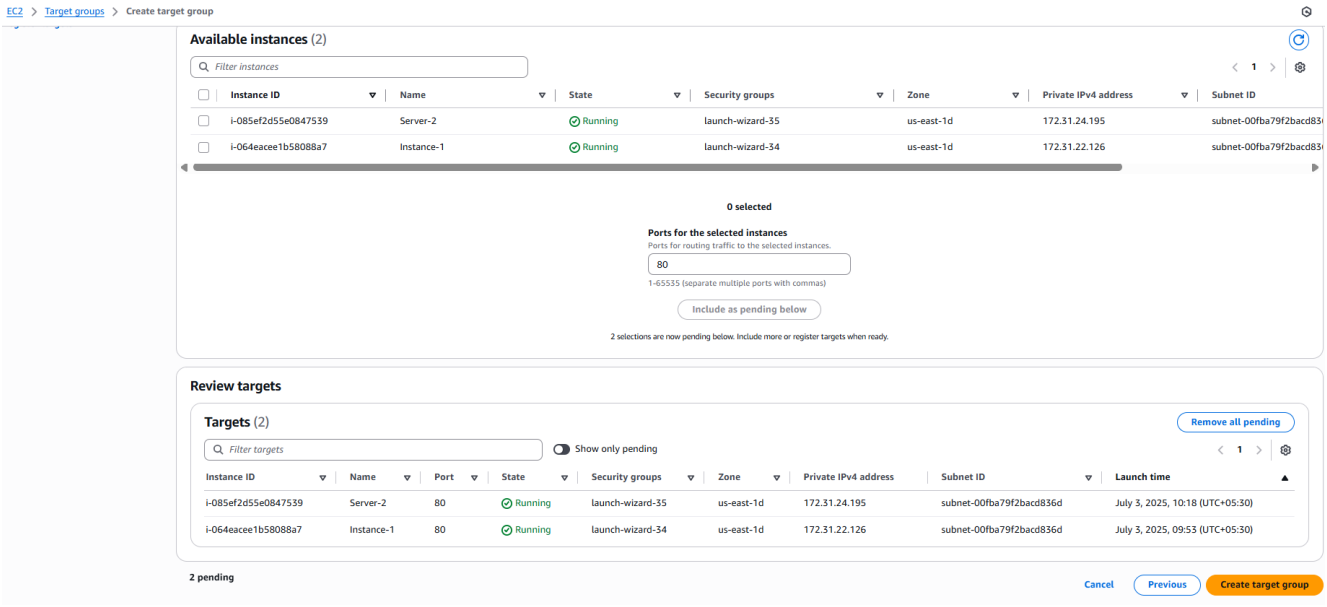
- Select your **FoodApp-AMI** → Click Launch instance from image
- Configure:
  - **Name:** FoodApp-Instance
  - **Instance type:** t2.micro
  - **Key pair:** my-keypair
  - **Network:** Default VPC
  - **Subnet:** Choose a different AZ (e.g., us-east-1a)
  - **Security Group:** SSH & HTTP
  - Click **Launch Instance**
- Test the application using the public IP address in your browser.(The ouput will be Food-app)

Step 3: Create a Target group to Load Balancer

- Navigate to EC2 > Target Groups
- Click Create target group
- Configure basic settings:
  - Target type: Instances
  - Name: e.g., FoodApp-TG
  - Protocol: HTTP
  - Port: 80
  - VPC: Choose your default or specific VPC
  - Click Next.

➢ **Register Targets (EC2 Instances):**

- Under Available instances, select one or more EC2 instances to attach
- Click **Include as pending below**

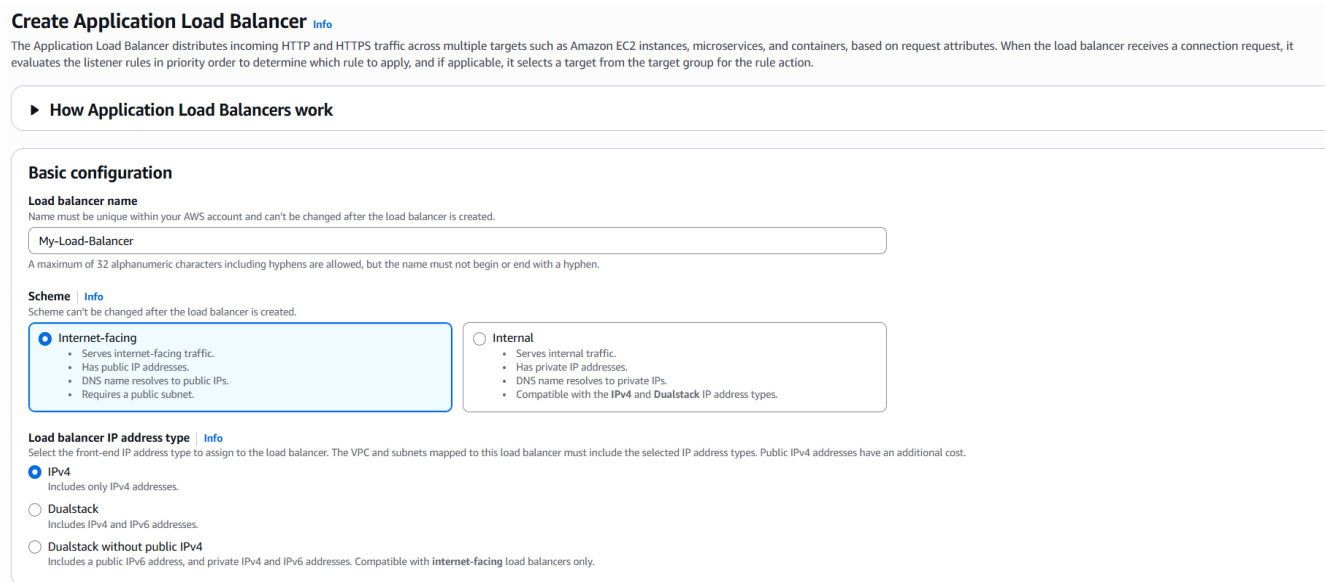


- Click **Create target group**

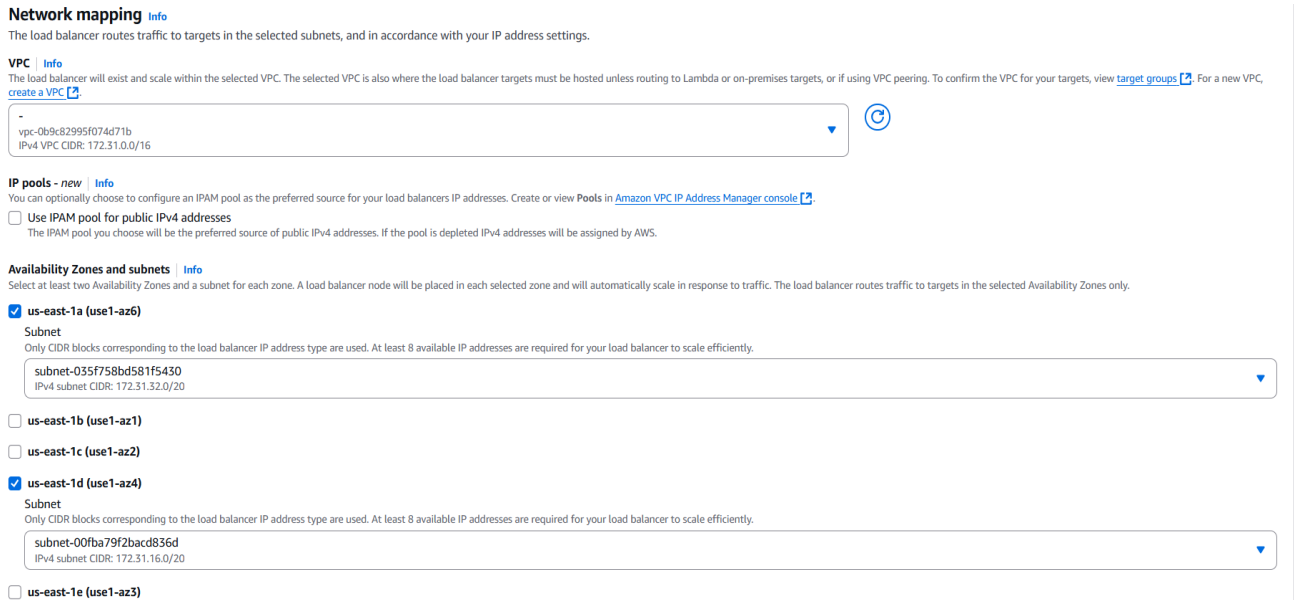
Your target group is now ready (This group will be used when you create or modify a **Load Balancer** to forward traffic to these registered EC2 instances.)

Step 4: Ceate a Load Balancer

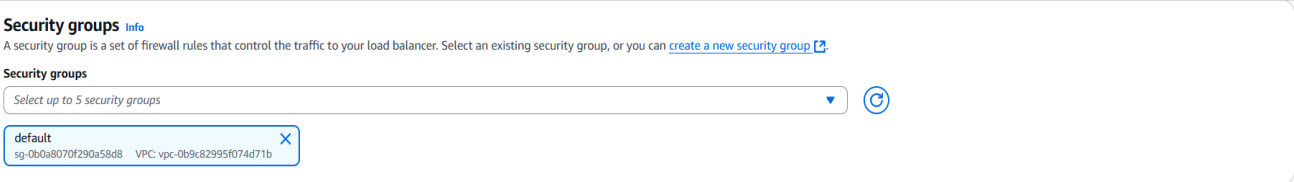
- Go to AWS Console → **EC2 > Load Balancers**
- Click **Create Load Balancer**
- Choose **Application Load Balancer**
- Configure Load Balancer :
  - **Name:** FoodApp-ALB
  - **Scheme:** Internet-facing

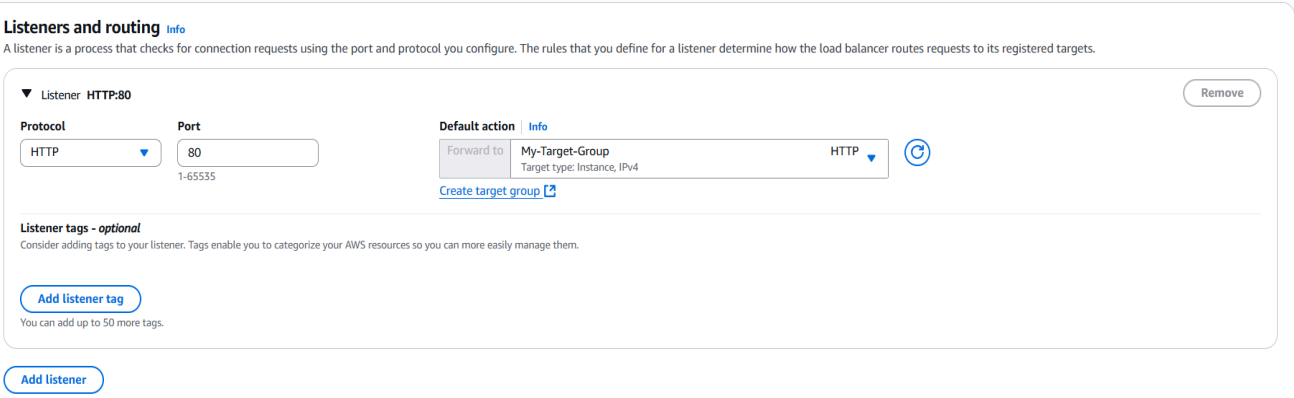


- **IP address type:** IPv4
- **Network mapping :** Default & Select at least two availability zones the actuall app server’s are running.



- **Listeners:** Add listener for HTTP on port 80
- Click **Next**
- **Configure Security Group :** Choose an existing Security Group or default *(Ensure it allows HTTP - port 80).*





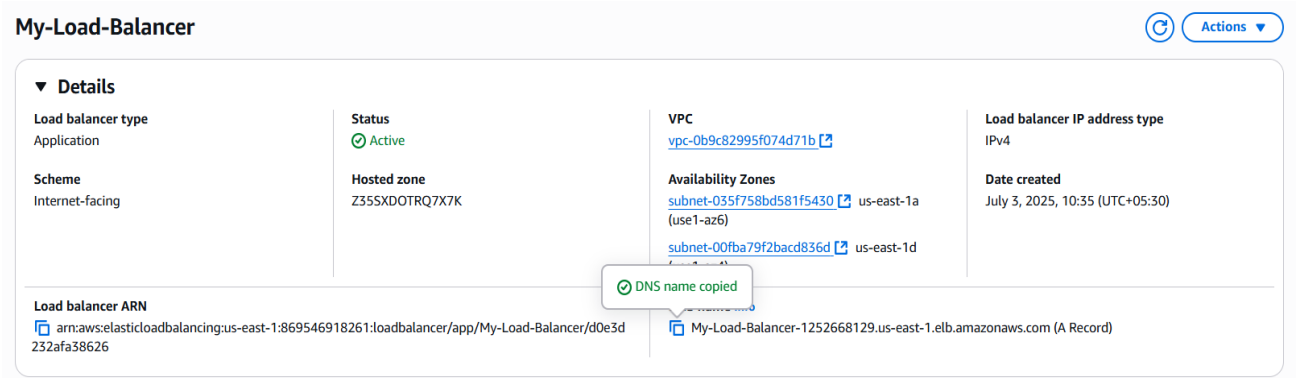
Review and Create

- Review settings
- Click Create Load Balance

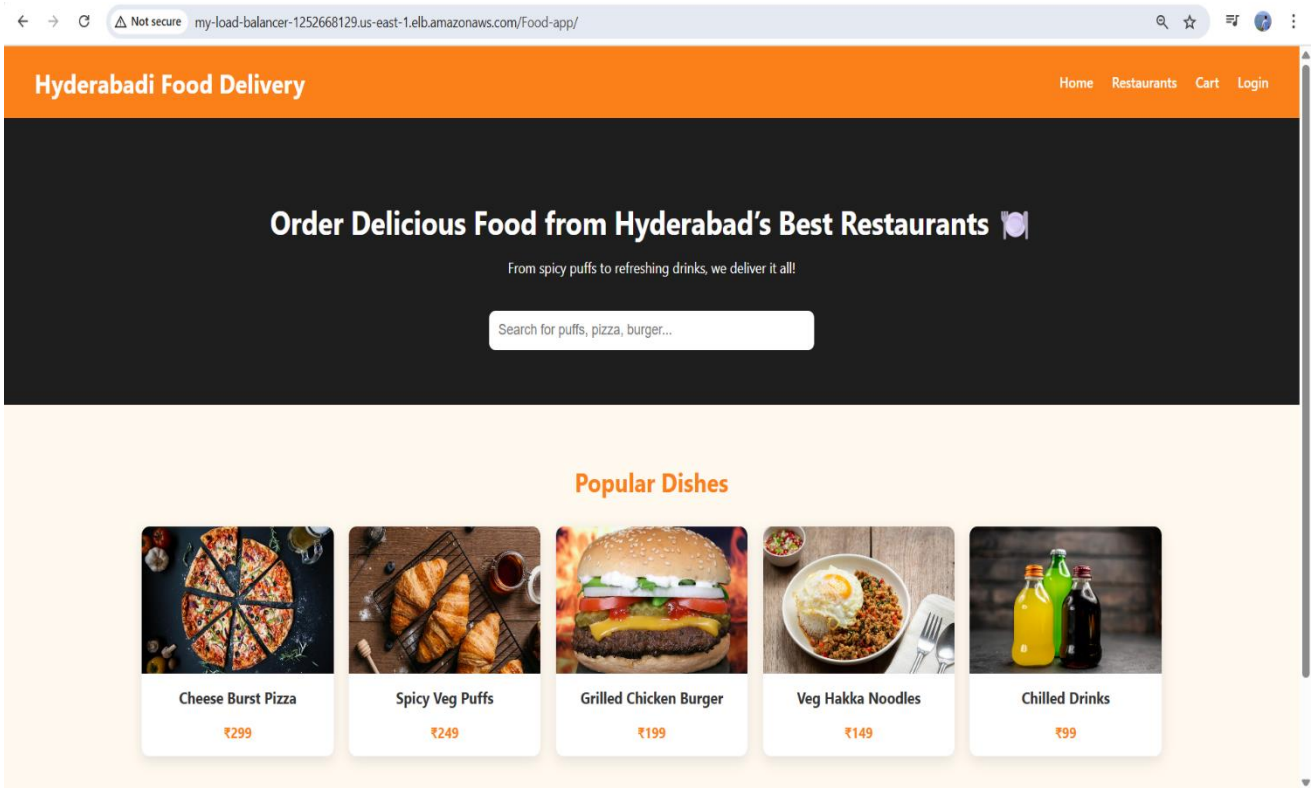
➢ Test Your Load Balancer

- Once status becomes **active**
- Open in your browser

http://<your-load-balancer-DNS-name>



➢ You should see your food app served from any healthy instance in the target group.



Step 5: Create Auto Scaling Group with SNS Notifications

Prerequisites :

- A Launch Template or AMI ready (e.g., FoodApp-LTEMP)

- A Target Group (e.g., FoodApp-TG)
- A Load Balancer created (e.g., FoodApp-ALB)
- A Security Group – default or existing
- A Key Pair

Create Auto Scaling Group:

- Go to EC2 > Auto Scaling Groups
- Click Create Auto Scaling group

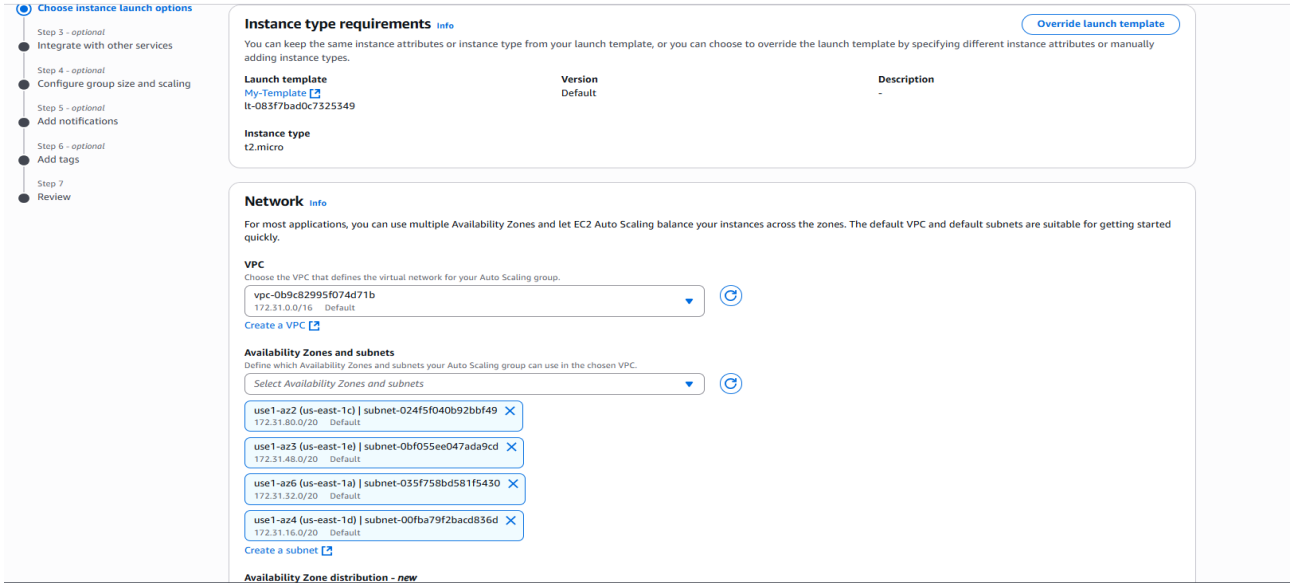
Fill in:

Name: FoodApp-ASG

Launch Template: Select FoodApp-LTEMP or your AMI-based launch template

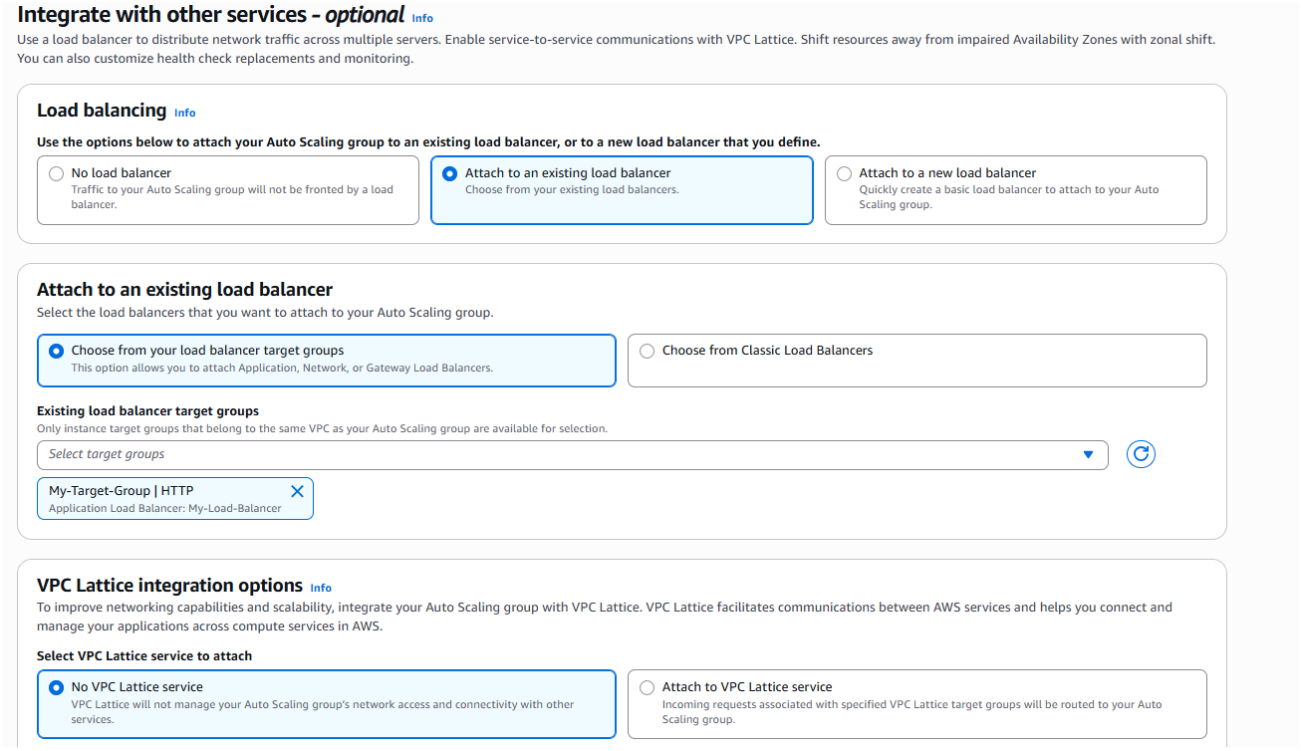
Click Next

- Configure Network
    - **VPC:** Choose default or custom VPC
    - **Availability Zones:** Select at least 2 (e.g., us-east-1a, 1b)
    - Click **Next**



- Attach Load Balancer
    - Choose **Attach to an existing load balancer**
    - **Attach to existing load balancer :** choose from your exsisting load balancer target groups

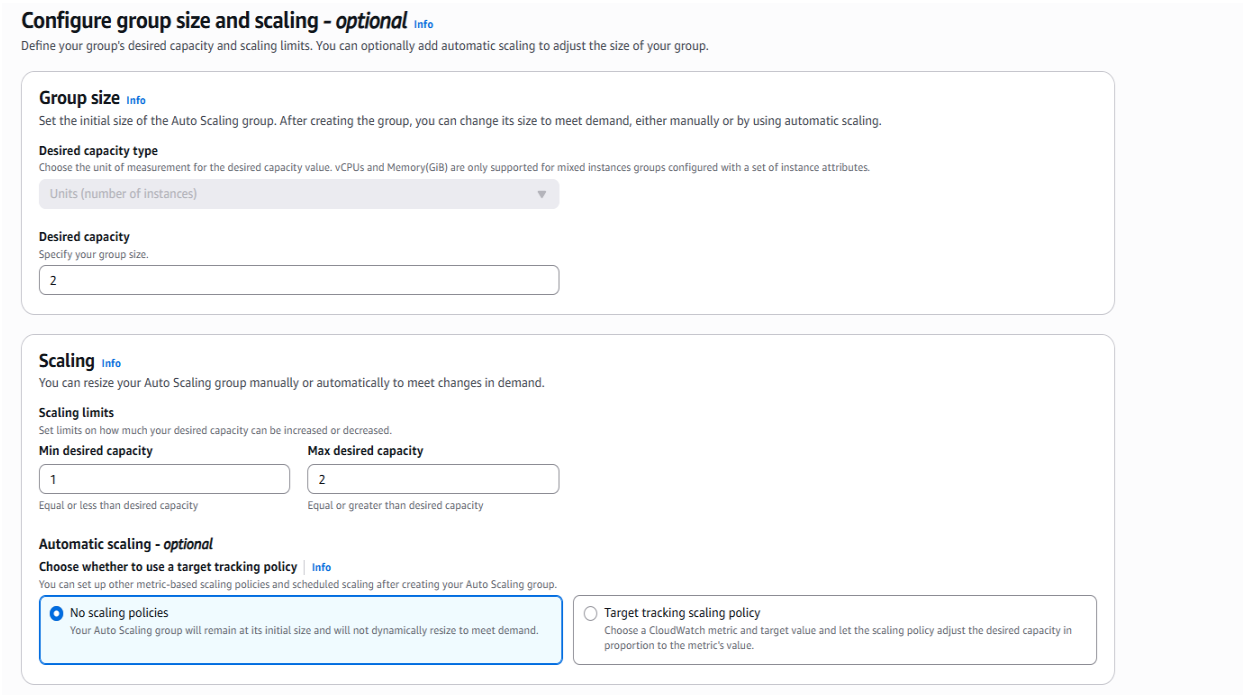
    - **Target Group:** FoodApp-TG
    - Click **Next**



- Set Desired Capacity :

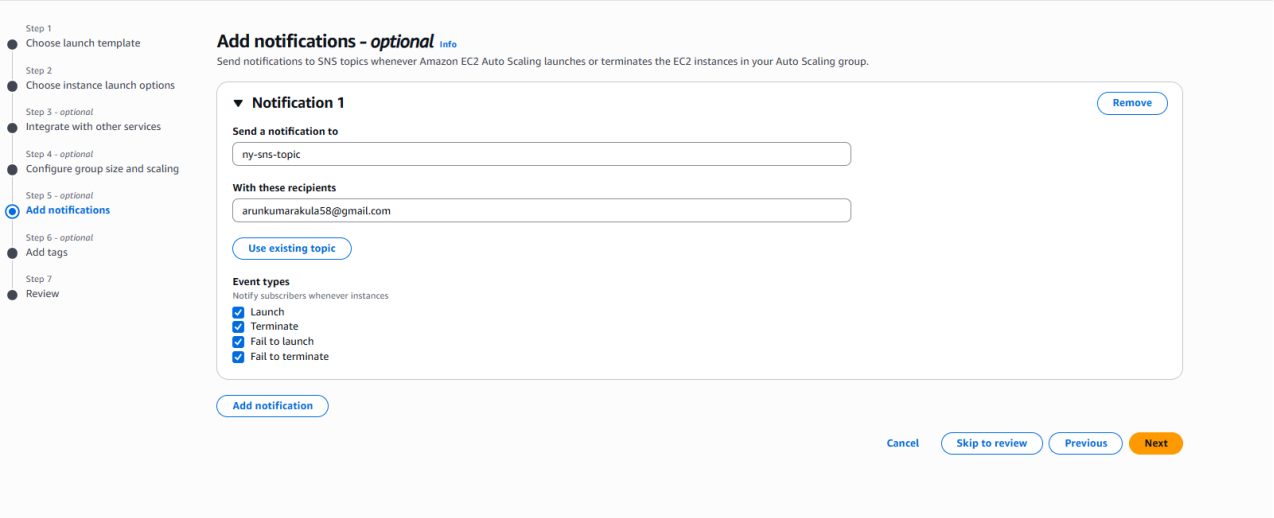
  Group size:

  - Minimum: 2
  - Desired: 2
  - Maximum: 4
  - Click **Next**

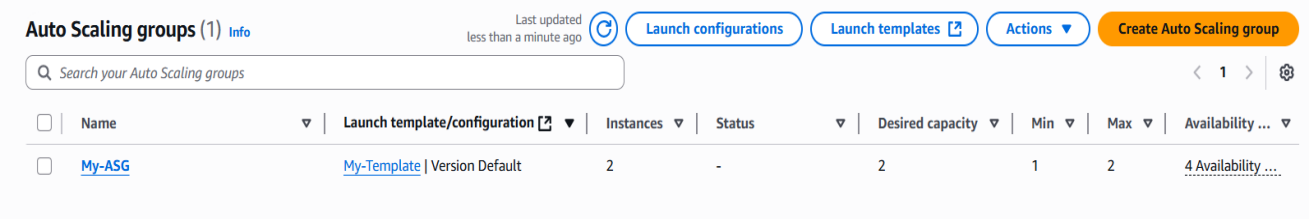


- Configure Scaling Policies
    - Choose **No scaling policies**
    - Metric type: **Average CPU utilization**
    - Target value: `70%`
    - Click **Next**

- Set-up SNS Toipc
    - Click **Create topic**
    - Name: FoodApp-SNS
    - With these recipients : youemail@gmail.com
    - Click **Next**

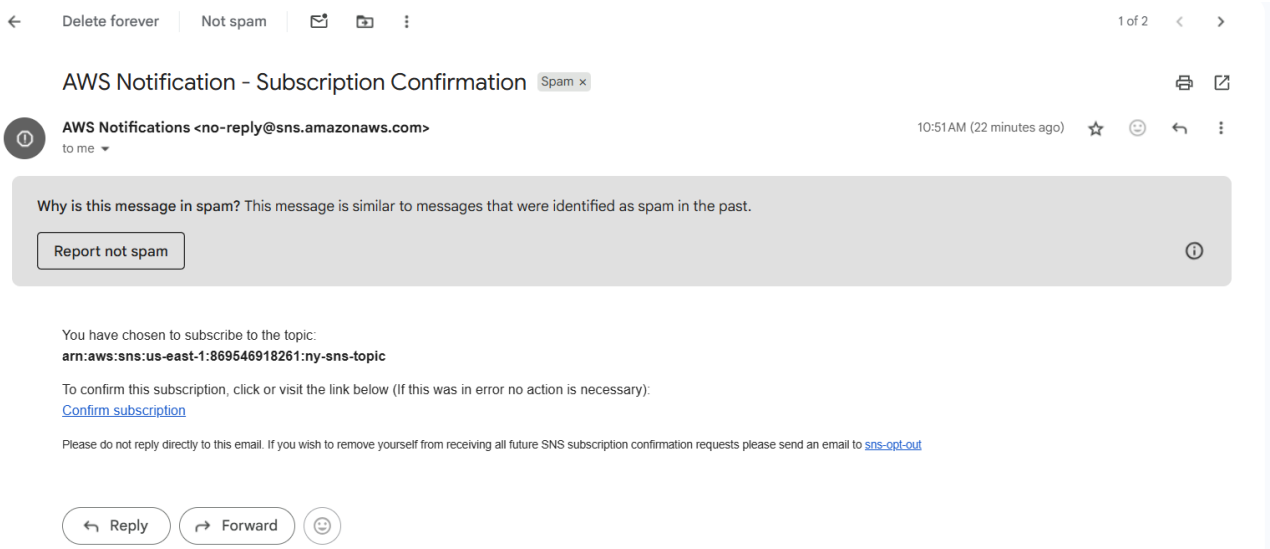


- **Review and Create**
    - Review all configurations
    - Click **Create Auto Scaling Group**

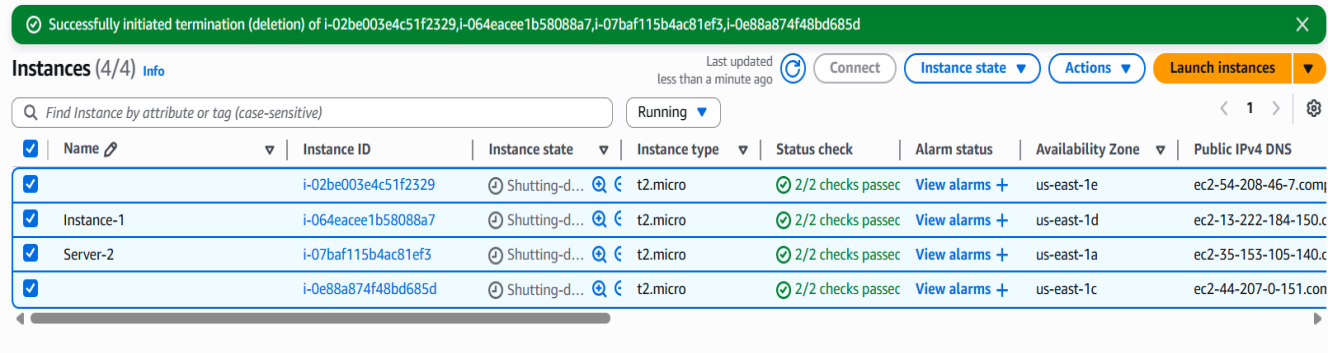


- **Subscribe SNS – Topic via Email:**
- Go to the email (check the email which is shared by aws)
- **Confirm** the subscription from your inbox

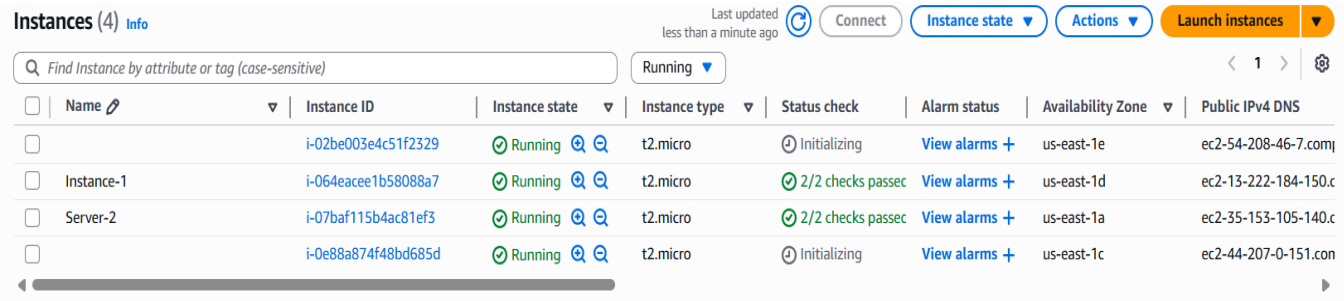
www.linkedin.com/in/arun-kumar-akula



- Test the application using the load balancer DNS-ip or Pulic-Ip of your server.(you will see your food-app)
- Now Terminate or install stress on your linux machine.

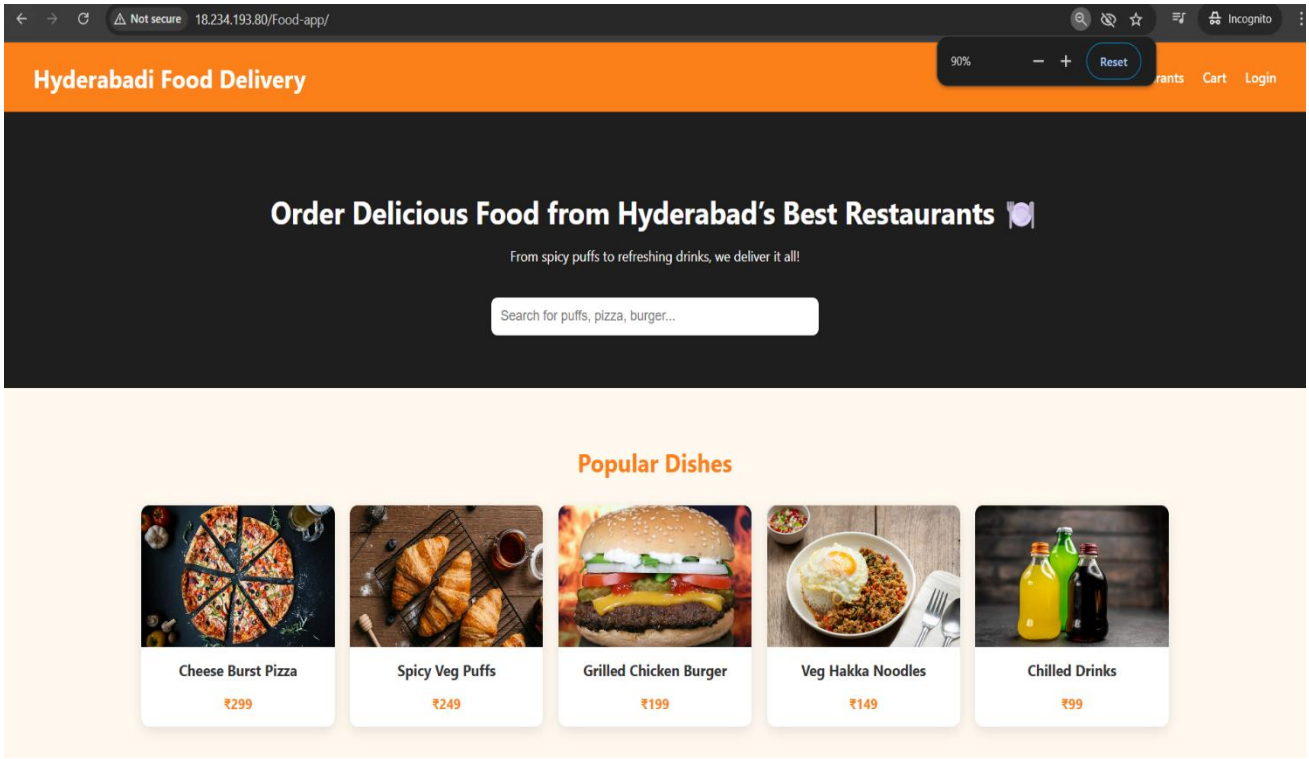


- Now the instances are shutting down, and when our application goes down, the Auto Scaling Group automatically detects the issue and replaces them with new instances.



- As shown in the above image, the Auto Scaling Group launches new instances to ensure high availability.

- Access the application via the Load Balancer DNS or public IP. Even if some EC2 instances are terminated, the Auto Scaling Group ensures the service remains available.



- Whenever our servers go down or get terminated, SNS will send a notification via email.