# COS20019

## Cloud Computing Architechture

### Week 10 – ACA Module 10:
### Challenge Lab – Creating a Scalable and Highly Available Environment for the Cafe

Truong Ngoc Gia Hieu
105565520

# Challenge Lab: Creating a Scalable and Highly Available Environment for the Café

## A. Scenario

The café will soon be featured on a famous TV food show. When the show airs, Sofía and Nikhil anticipate that the café's web server will experience a temporary spike in the number of users—perhaps even up to tens of thousands of users. Currently, the café's web server is deployed in one Availability Zone, and Sofía and Nikhil are worried that the website won't be able to handle the expected increase in traffic. They want to ensure that their customers have a great experience when they visit the website and that they don't experience any issues, such as lags or delays in placing orders.

To ensure this experience, the website must be responsive, able to scale both up and down to meet fluctuating customer demand, and be highly available. Instead of overloading a single server, the architecture must distribute customer order requests across multiple application servers so it can handle the increase in demand.

In this lab, you take on the role of Sofía to implement a scalable and highly available architecture for the café's web application.

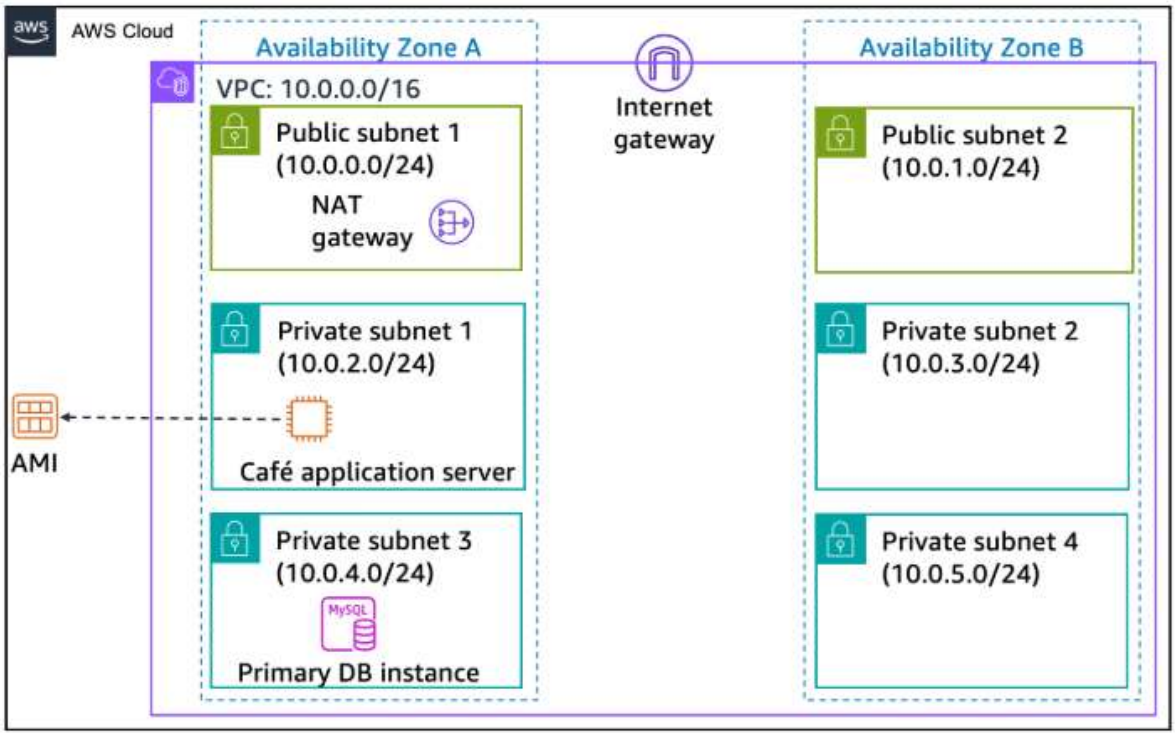
## B. Lab overview and objectives

In this lab, you use Elastic Load Balancing (ELB) and Amazon EC2 Auto Scaling to create a scalable and highly available environment on Amazon Web Services (AWS).

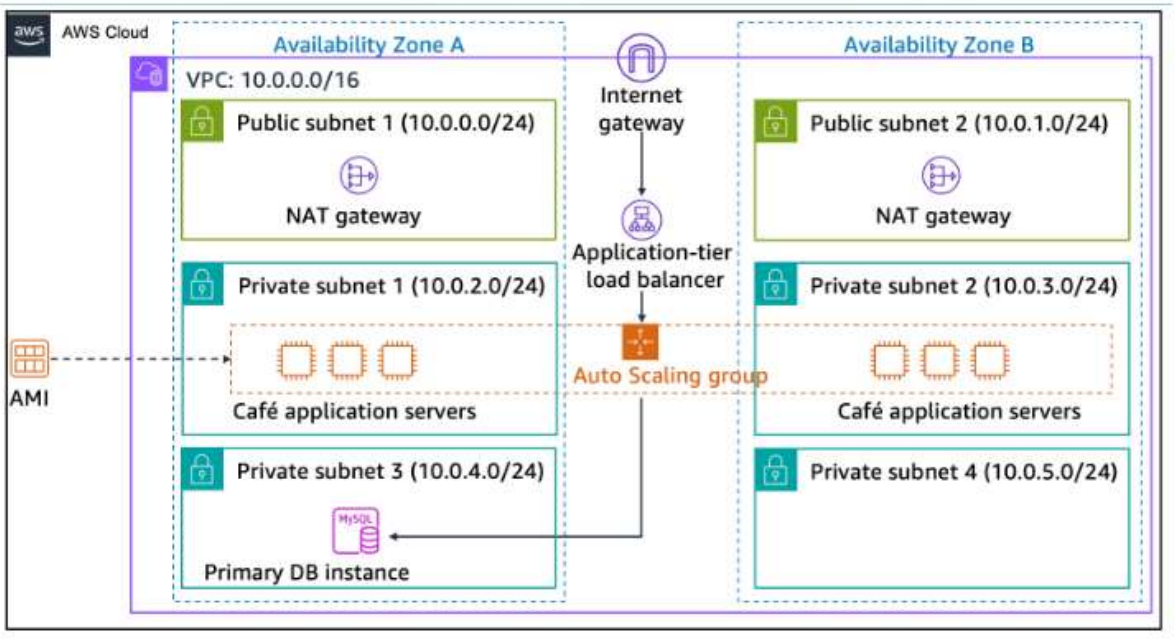
After completing this lab, you should be able to do the following:

- Inspect a virtual private cloud (VPC).
- Update a network to work across multiple Availability Zones.
- Create an Application Load Balancer.
- Create a launch template.
- Create an Auto Scaling group.
- Set an alarm on a system metric to initiate auto scaling.
- Test load balancing and automatic scaling.

When you start the lab, your architecture will look like the following example:




At the end of this lab, your architecture should look like the following example:




**Note:** In this challenge lab, step-by-step instructions are not provided for most tasks. You must figure out how to complete the tasks on your own.

## Duration

This lab requires approximately **90 minutes** to complete..

## AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

# C. Accessing the AWS Management Console

To access the AWS Management, click on the button Start Lab and wait the circle next to the text AWS turns from yellow to green.


***Figure 1: ÀWS Activated***

## A business request for the café: Implementing a scalable and highly available environment (challenge)

Sofía understands that she must complete some tasks to implement high availability and scalability for the café's web application. However, before changing the café's application architecture, Sofía must evaluate its current state.

In the next several tasks, you work as Sofía to create and configure the resources that you need to implement a scalable and highly available application.

# D. Task 1: Inspecting your environment

To answer required questions, click on the button **AWS Details**. Then, choose the **Access the multiple choice questions link** to load following questions and submit after finishing each question


***Figure 2: Question 1***

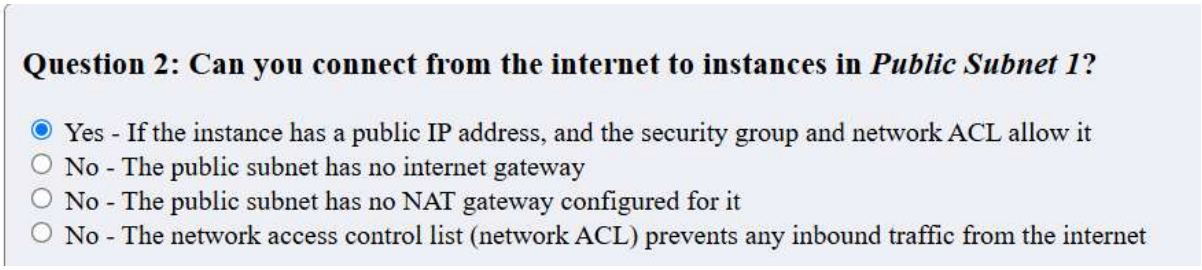



*Figure 3: Question 2*

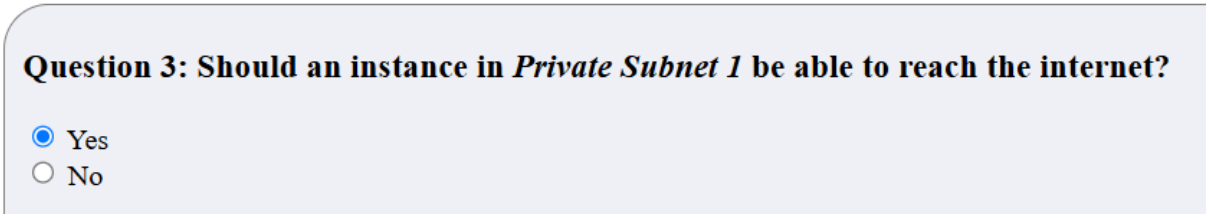



*Figure 4: Question 3*

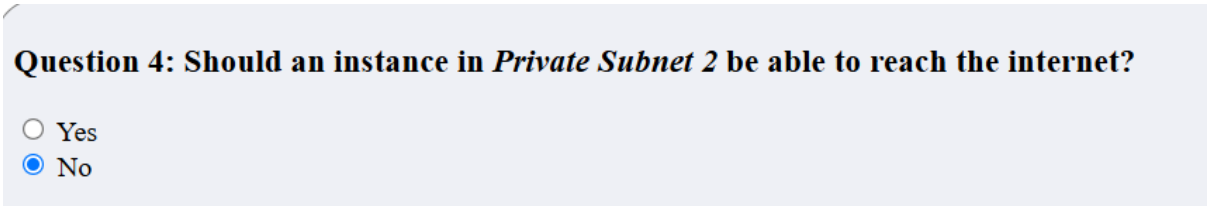



*Figure 5: Question 4*


*Figure 6: Question 5*

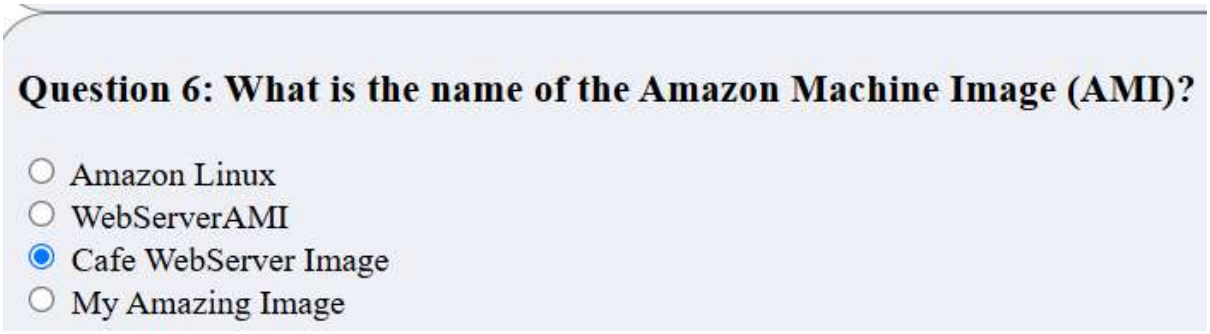



*Figure 7: Question 6*

# E. Task 2: Updating a network to work across multiple Availability Zones (Creating a NAT gateway for the second Availability Zone)

***Step 2.1***: Search and select the **NAT gateways** service

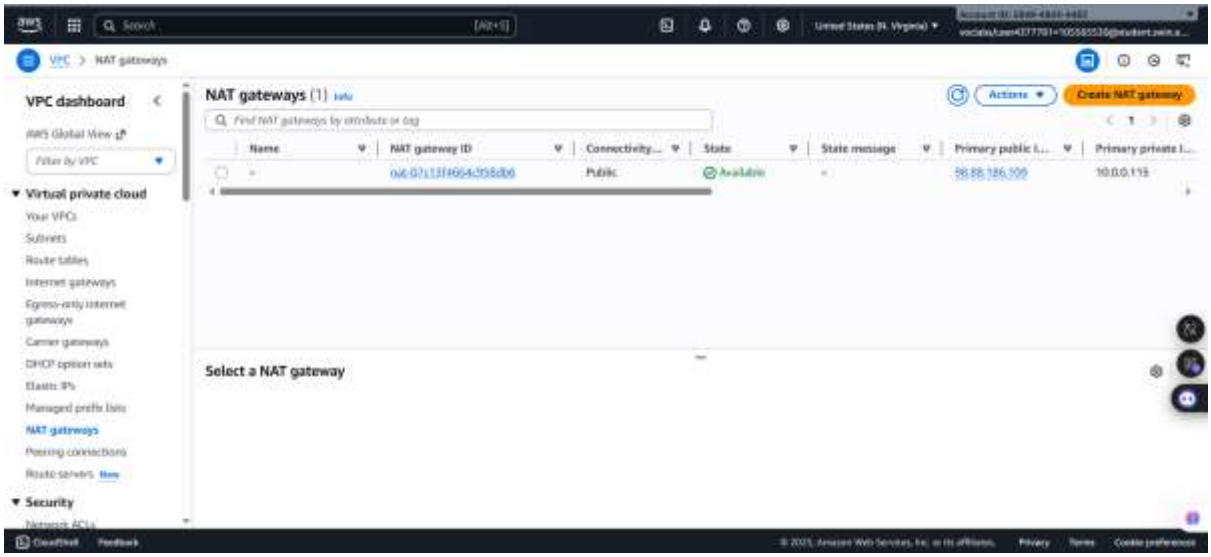



***Figure 8: NAT Gateway page***

***Step 2.2***: After clicking button "Create NAT gateway", configure following settings:

- **Name – *optional:*** lab-natgateway
- **Subnet:** Public Subnet 2
- **Elastic IP allocation ID:** Choose **Allocate Elastic IP** button

Then, click button **Create NAT gateway** at the bottom of the page

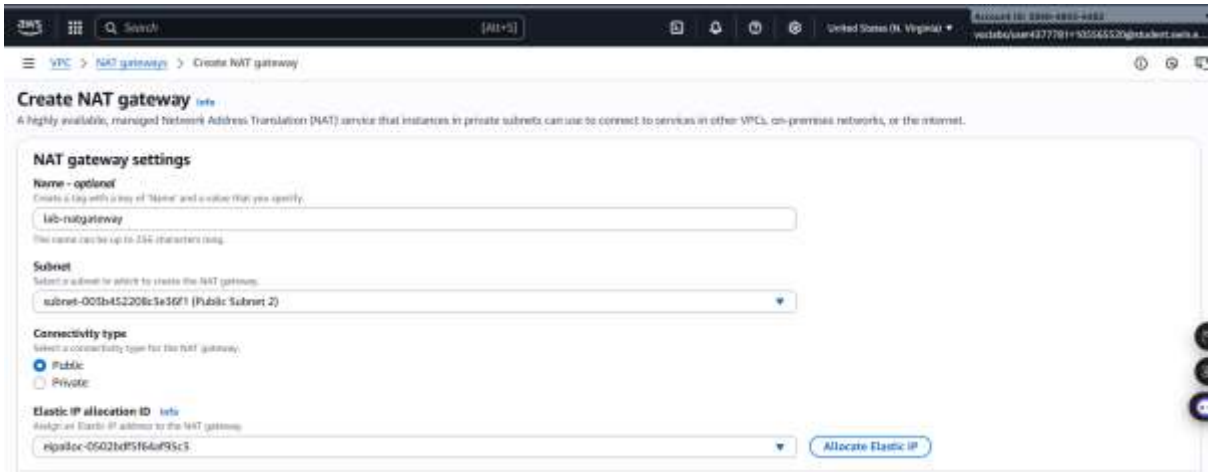



***Figure 9: NAT Gateway configuration***

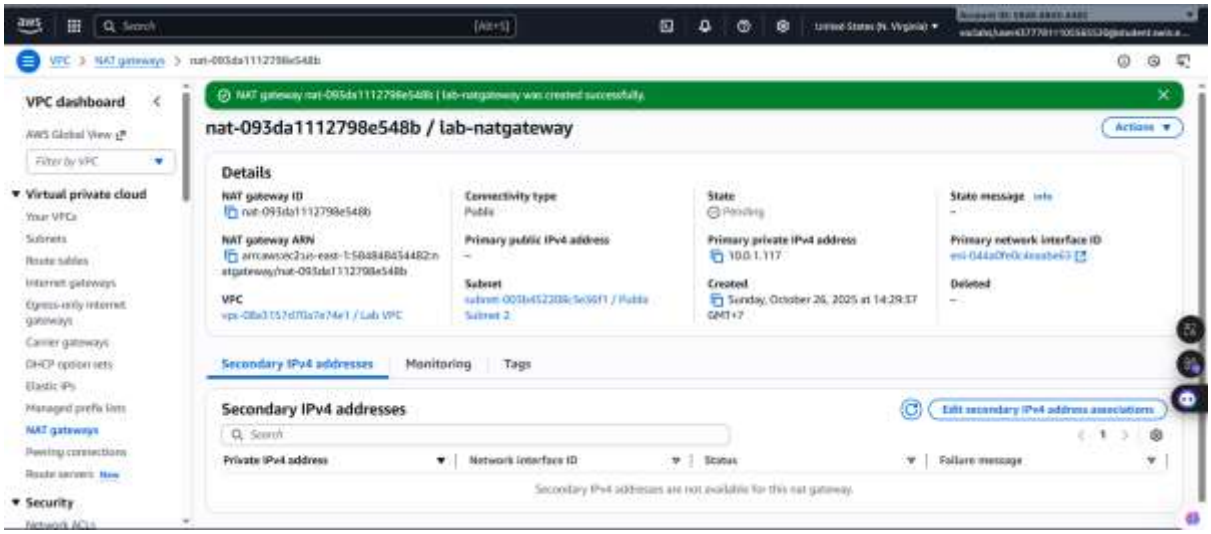



*Figure 10: NAT Gateway created successfully*

***Step 2.3:*** In the left navigation pane, choose the **Route tables** and tick the box next to the **Private Route Table 2**

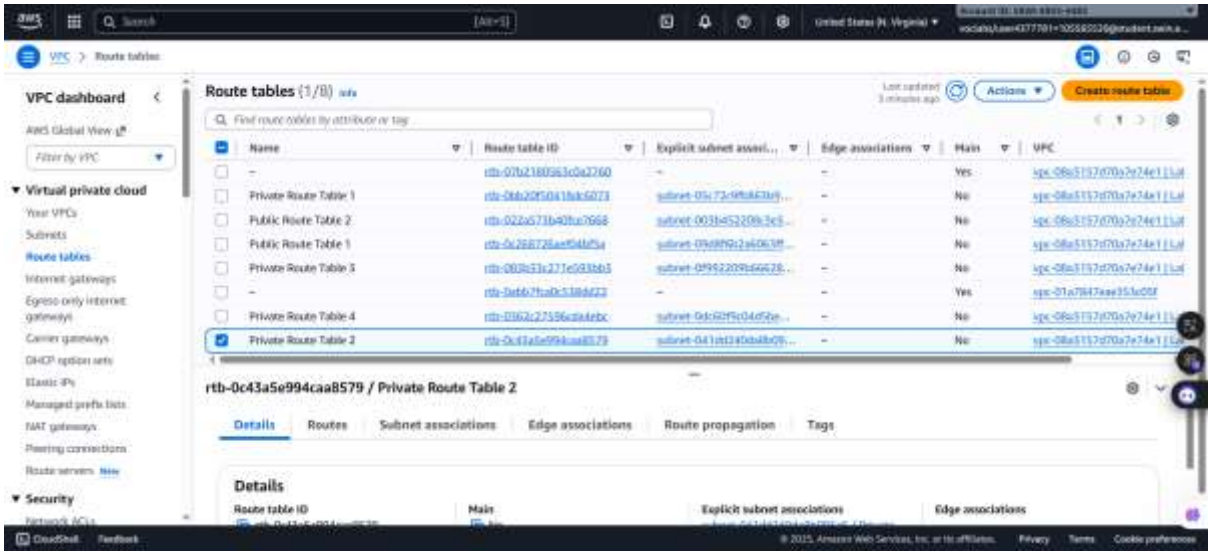



*Figure 11: Private Route Table 2 selected*

***Step 2.4:*** In the **Routes** tab, choose **Edit routes**, click **Add route**, and then configure below settings:

+ **Destination**: 0.0.0.0/0
+ **Target**: NAT Gateway => lab-natgateway

Then, click **Save changes**

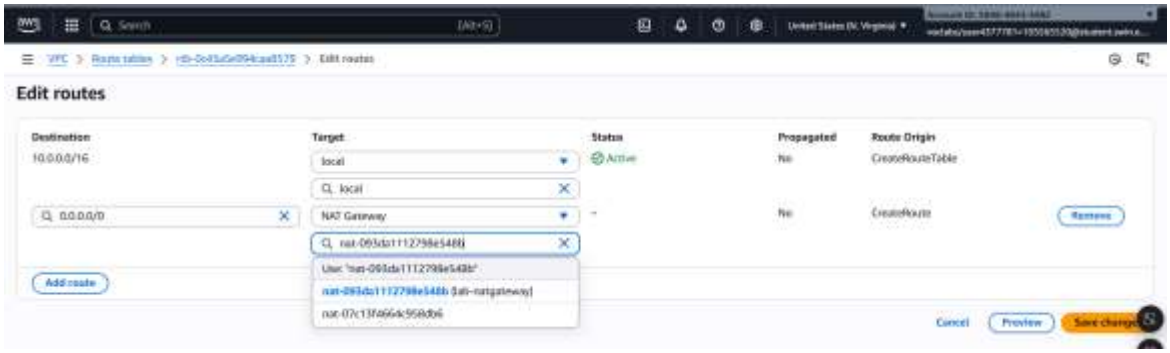



*Figure 12: Edit route configuration*

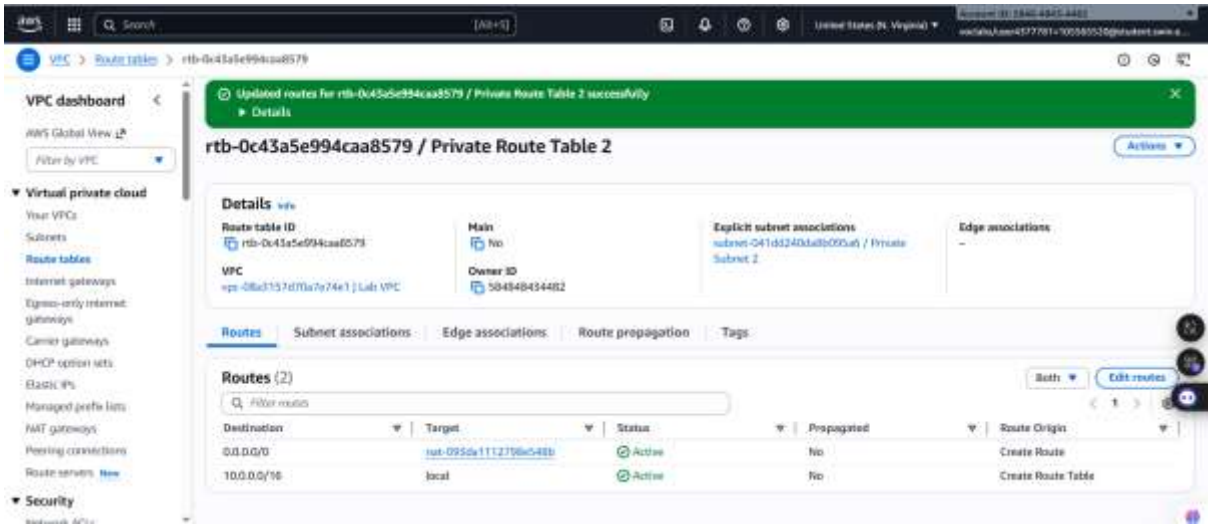



*Figure 13: Edit route successfully*

# F. Task 3: Creating a launch template

During the lab setup, an AMI was created from the CafeWebAppServer instance. In this task, you create a launch template by using this AMI.

***Step 3.1:*** Search and select **EC2.** Next , choose **Launch Templates** in left navigation pane and select button **Create launch templte**

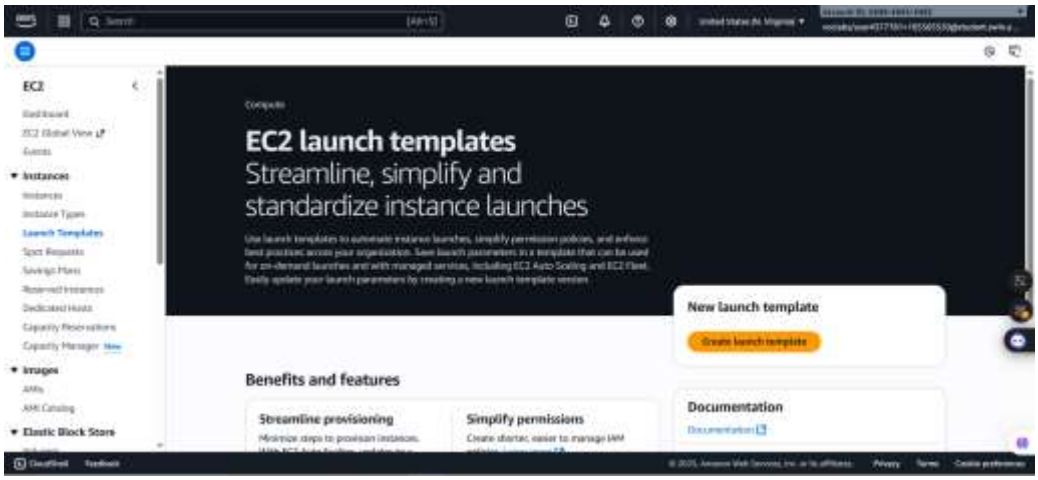



*Figure 14: Launch template homepage*

***Step 3.2:*** Set the name for the templte in the field **Launch template name – required** is *cafechallengetemplate*

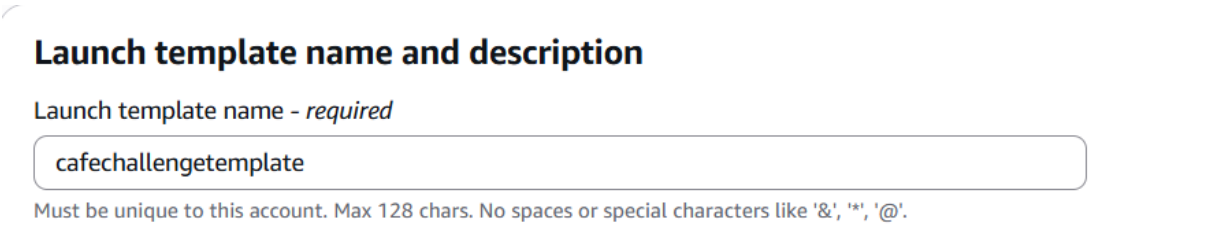



***Figure 15: Template name configuration***

***Step 3.3:*** **Application and OS Images (Amazon Machine Image):** From **My AMIs,** choose **Cafe WebServer Image.**

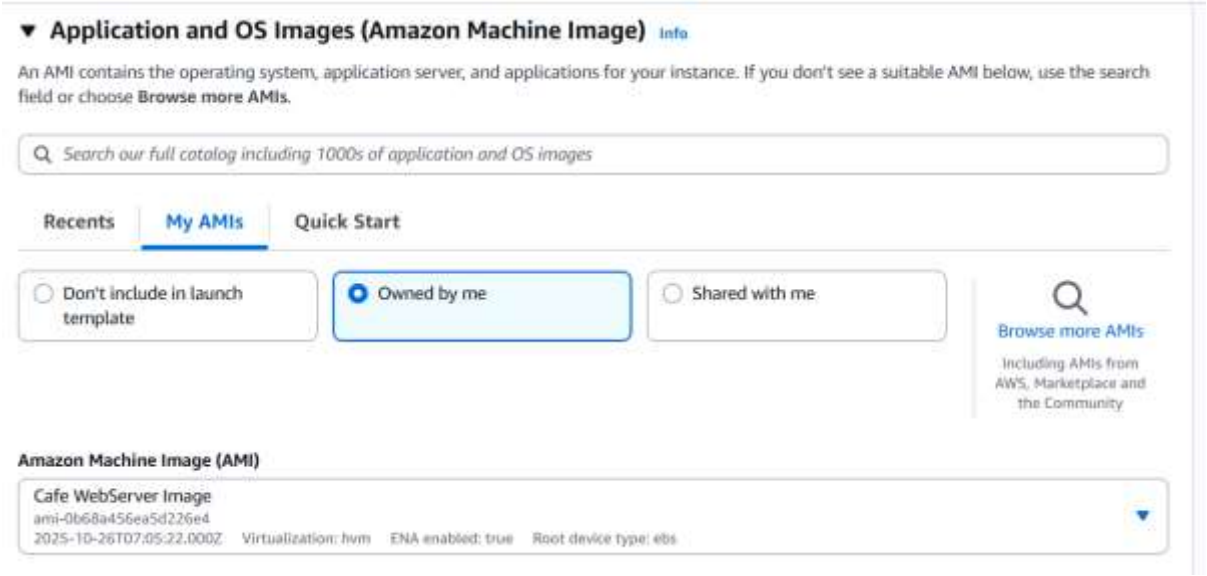



***Figure 16: AMI Configuration***

***Step 3.4:*** **Instance type**: Choose **t2.micro.**

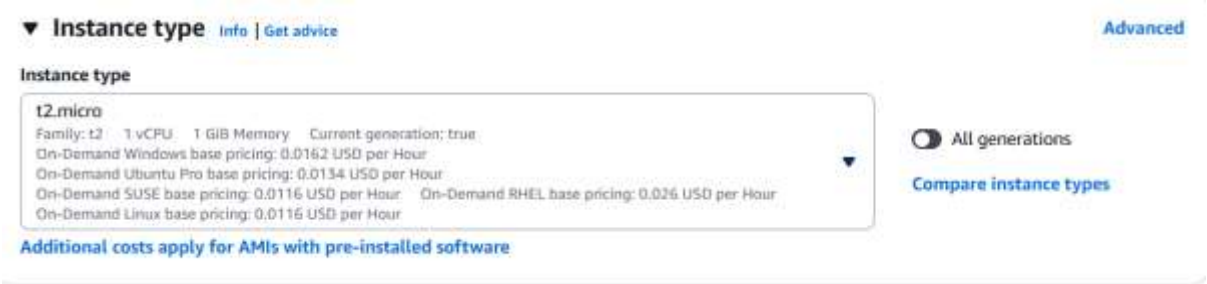



***Figure 17: Instance configuration***

***Step 3.5:*** **Key pair (login):** Choose **Create new key pair.**


*Figure 18: Key pair configuration*

***Step 3.6:*** **Security groups:** Enter **CafeSG.**

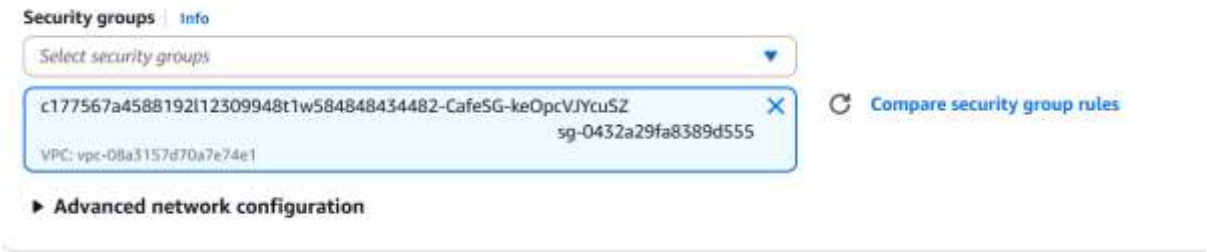



*Figure 19: Security group configuration*

***Step 3.7:*** **Resource tags**: Choose **Add new tag**, and configure the following options:

- Key: Enter Name.
- Value: Enter webserver.
- Resource types: Choose Instances.

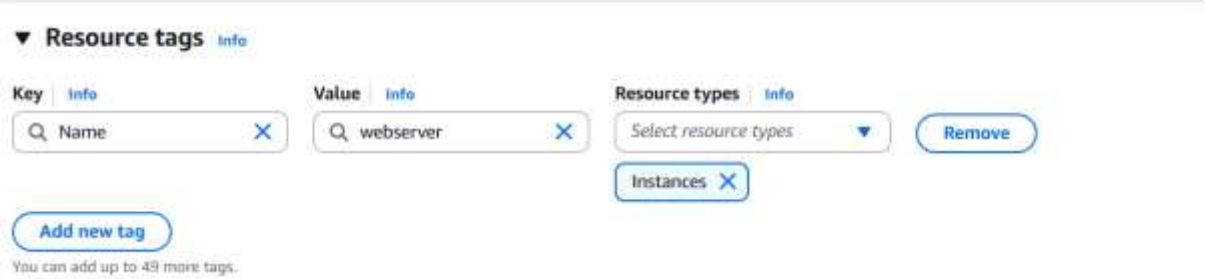



*Figure 20: Resource Tag configuration*

***Step 3.8:*** **IAM instance profile:** Choose **CafeRole.**


*Figure 21: IAM instance profile configuration*

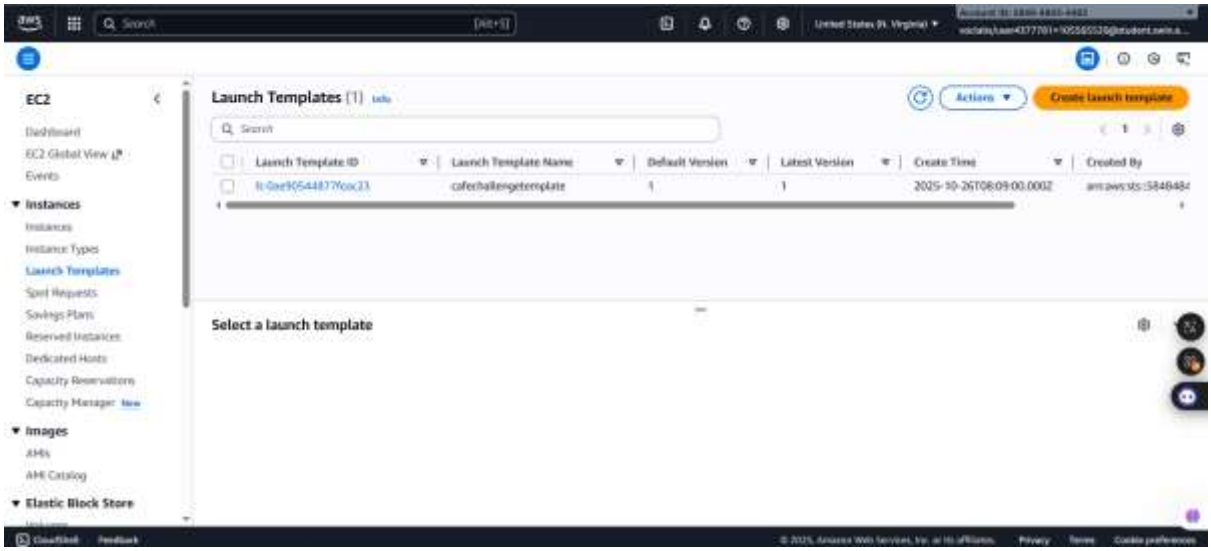



*Figure 22: Teemplate created successfully*

# G. Task 4: Creating an Auto Scaling group

Now that the launch template is defined, you create an Auto Scaling group for the instances. In this task, do not create a load balancer when you create the Auto Scaling group. You create a load balancer in the next task.

While creating an Auto Scaling group, you set a metric (CPU utilization) to initiate auto scaling events.\

***Step 4.1:*** In left navigation pane, select **Auto Scalin Group** and choose button **Create Auto Scaling group**

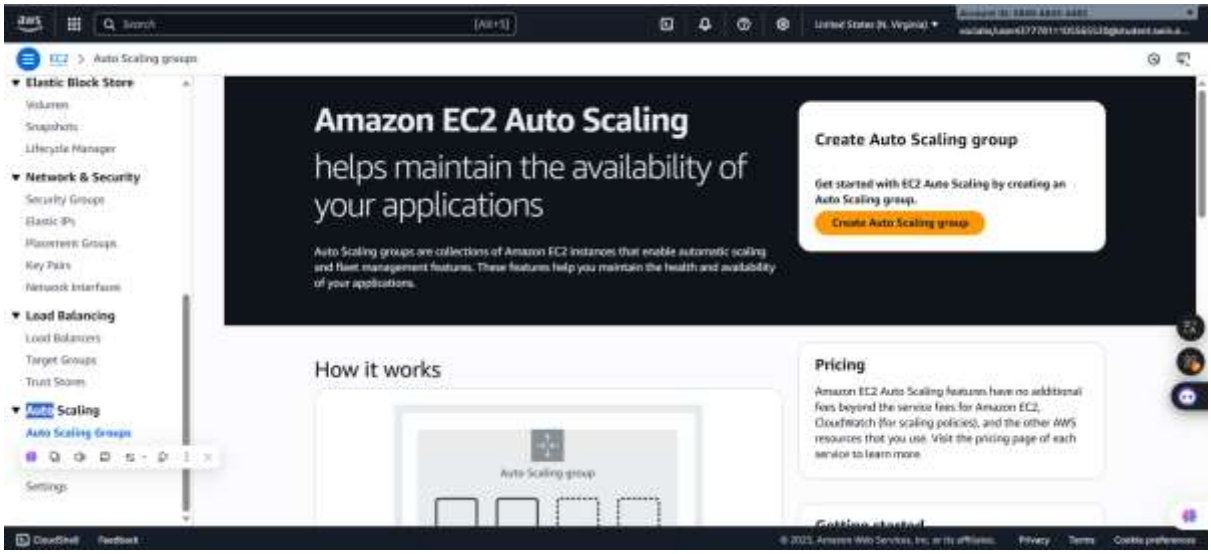



*Figure 23: Auto Scaling Group homepage*

***Step 4.2:*** Set the **name** and **template** in **Step 1: Choose launch template or configuration**

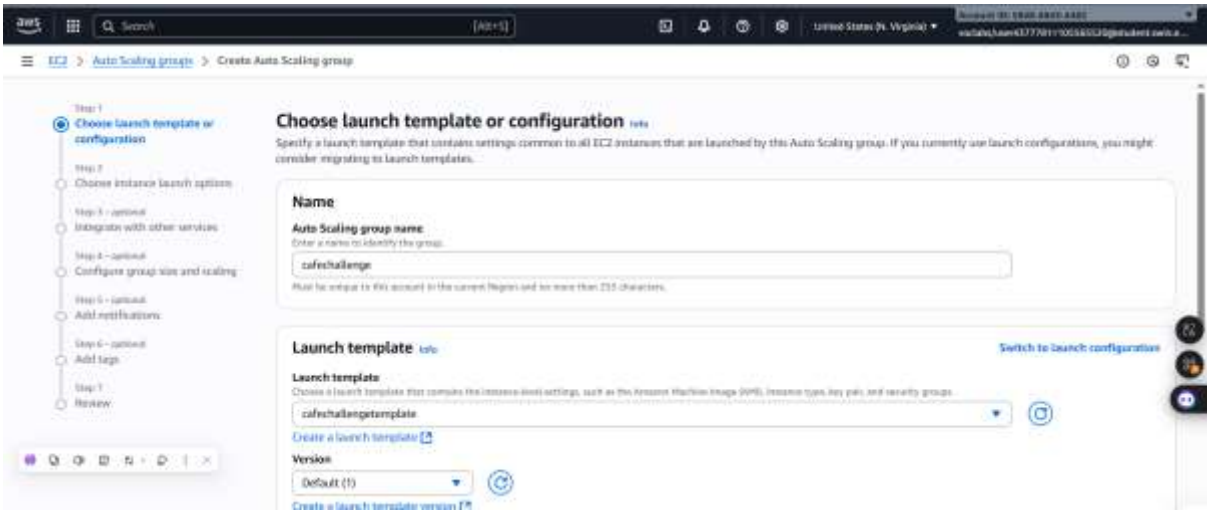
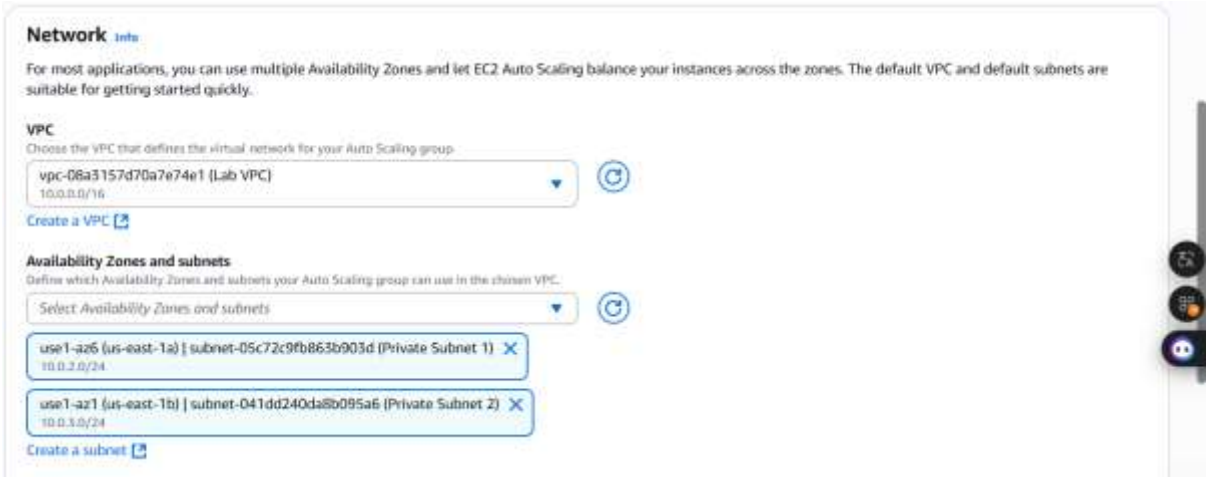



*Figure 24: Name and template configuration*

***Step 4.3:*** Congiure following settings in **Step 2: Create instance launch options**

**** VPC: Use the VPC that was configured for this lab.***

*** Availability Zones and subnets:** Choose **Private Subnet 1** and **Private Subnet**




**2.**

*Figure 25: Network configuration*

***Step 4.4:*** In **step 4 – Optional**, configure:

- **Desired capacity**: Enter 2.
- **Min desired capacity**: Enter 2.
- **Max desired capacity**: Enter 6.

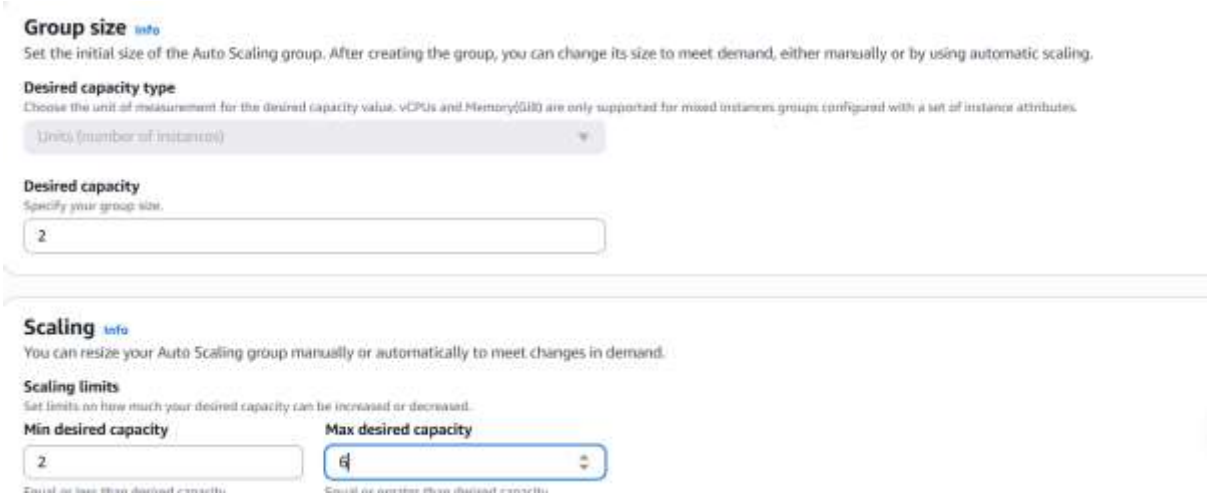



*Figure 26: Group Size and Scaling configuration*

***Step 4.5:*** **Automatic scaling - optional**: Choose **Target tracking scaling policy**, and configure the following options:

- **Metric type**: Choose Average CPU utilization.
- **Target value**: Enter 25.
- **Instance warmup**: Enter 60.

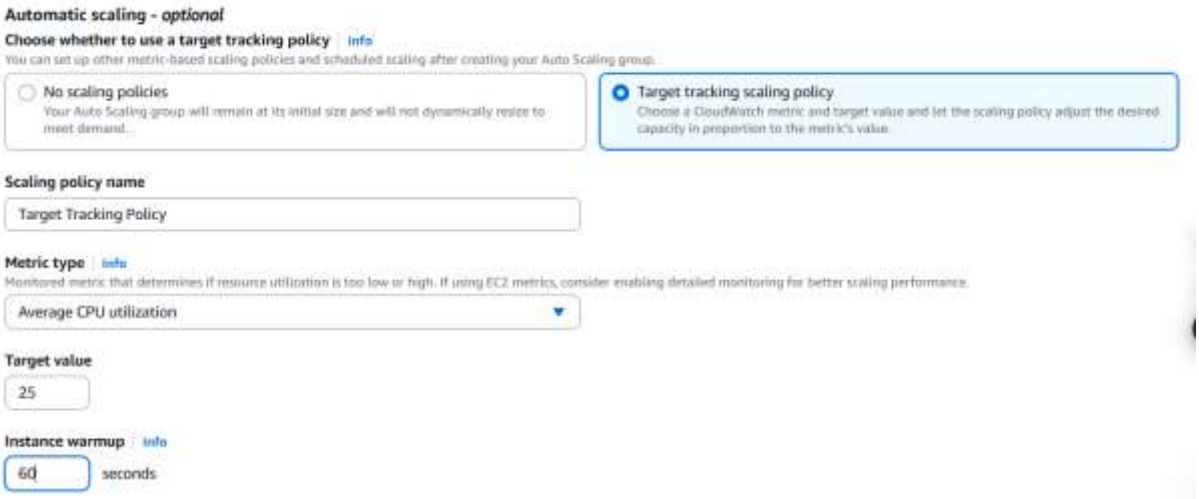



*Figure 27: Automatic scaling configuration*

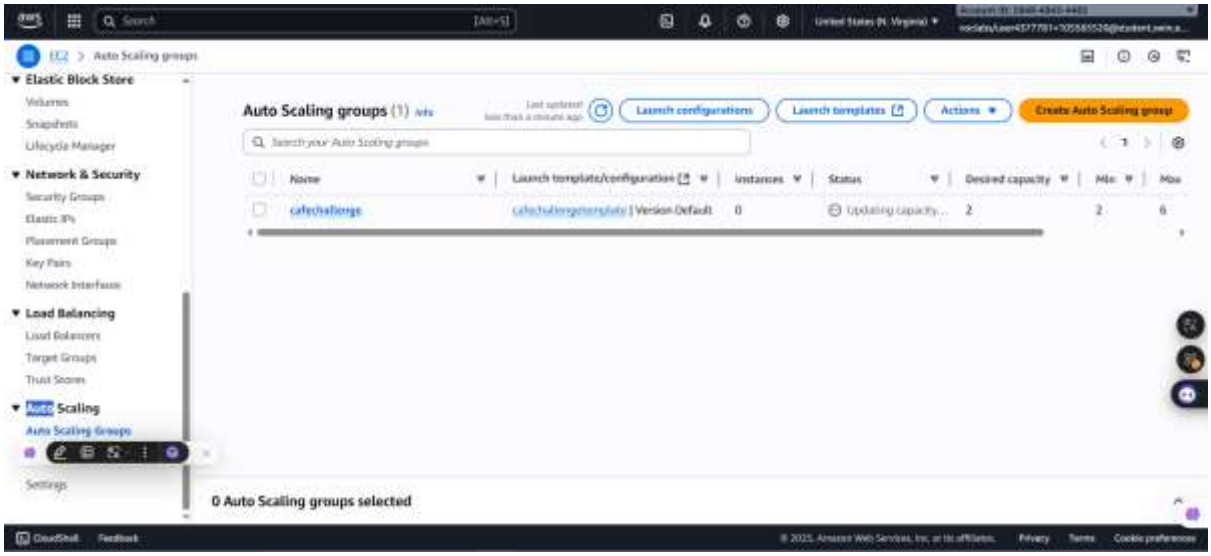



*Figure 28: Auto Scaling Group created successfully*

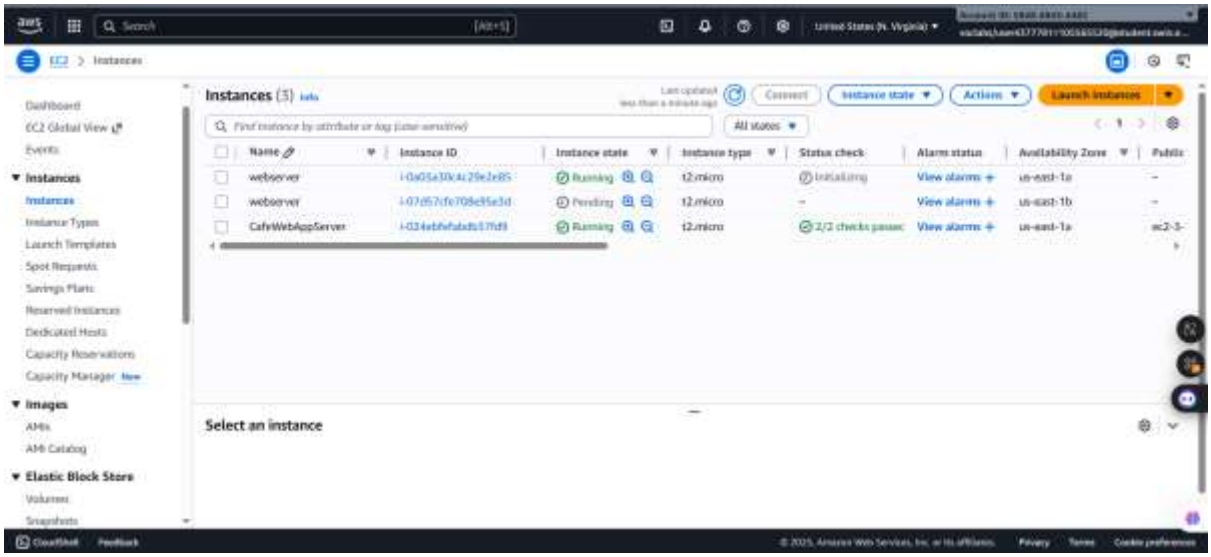



*Figure 29: EC2 Confirmation*

# H. Task 5: Creating a load balancer

Now that your web application server instances are deployed in private subnets, you need a way for the outside world to connect to them. In this task, you create a load balancer to distribute traffic across your private instances.

***Step 5.1:*** In left navigation pane, choose **Load balancer** and select **Create Load Balancer** button. Then, choose button **Create** in Application Load Balancer type

***Step 5.2:*** For **Load balancer name**, type cafechallenge


*Figure 30: Name configuration*

***Step 5.3:*** In **Network mapping** configuration:

*** VPC**: Use the VPC configured for this lab.

*** Subnets**: Use the two public subnets.

***Figure 30: Name configuration***

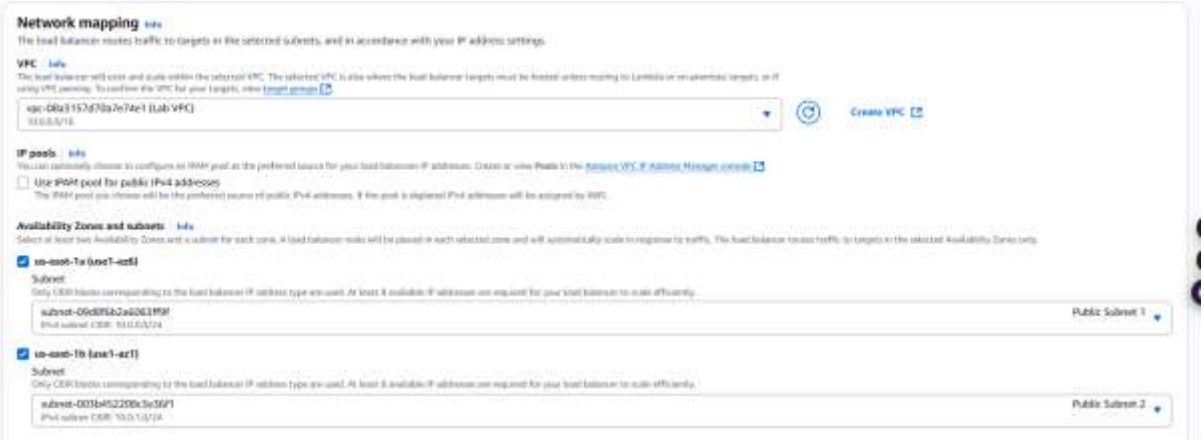



***Figure 31: Network mapping configuration***

***Step 5.4:*** **Security groups**: Create a new security group that allows HTTP traffic from anywhere.


***Figure 32: Security group configuration***

***Step 5.5:*** Target group: Create a new target group.


***Figure 33: Target group configuration***

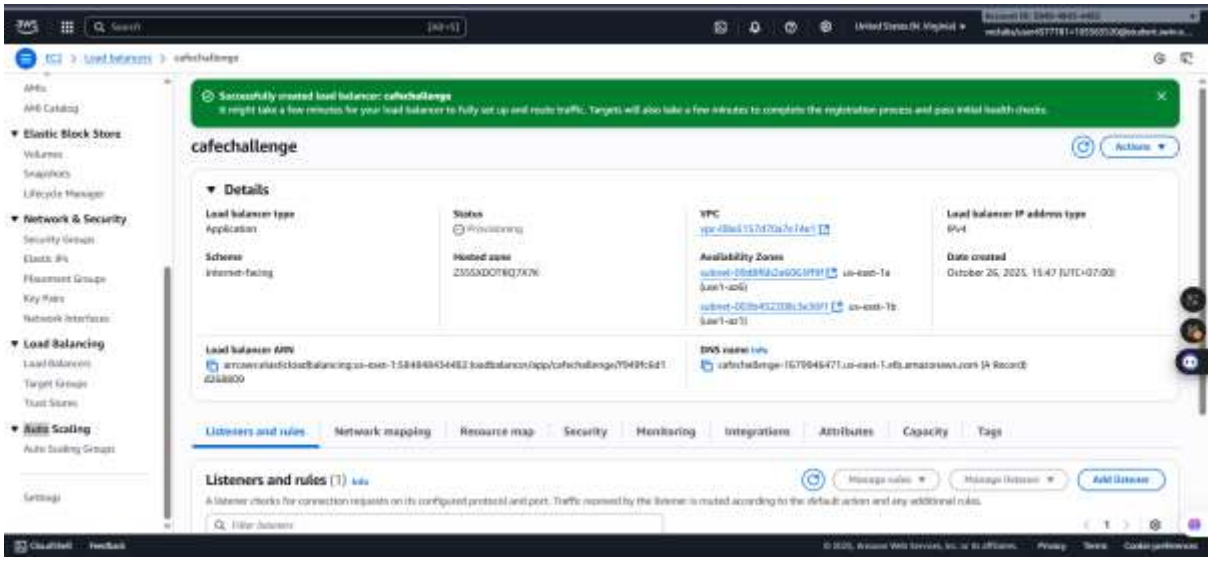



*Figure 34: Created successfully*

# I. Task 6: Testing load balancing and automatic scaling

## I.I Task 6.1: Testing the web application without load

***Step 6.1***: Paste the link **http://cafechallenge-1679946471.us-east-1.elb.amazonaws.com/cafe/** to new browser tab

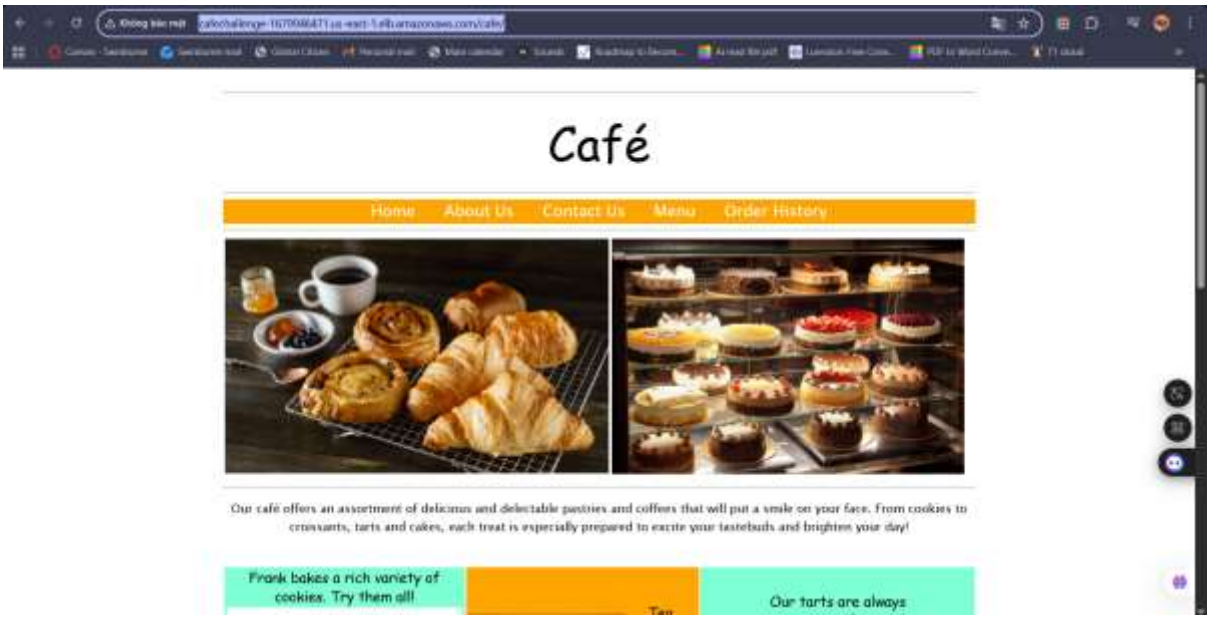



*Figure 35: Load successfully*

## I.II Task 6.2: Testing automatic scaling under load

Step 6.2: Paste the command lines into EC2 console

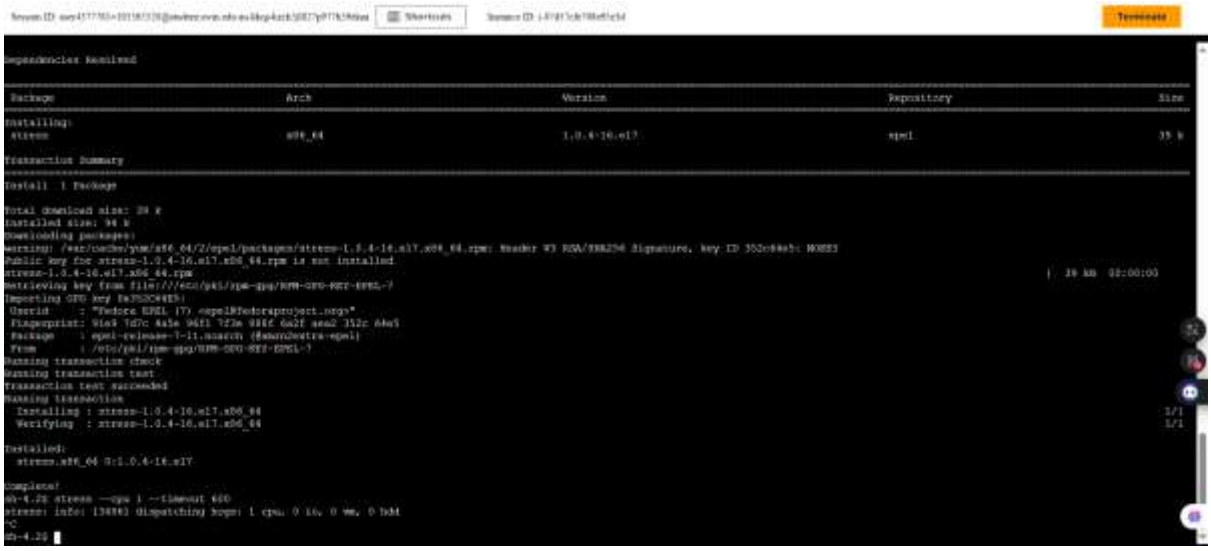


*Figure 36: Command lines*

# J. Completed task

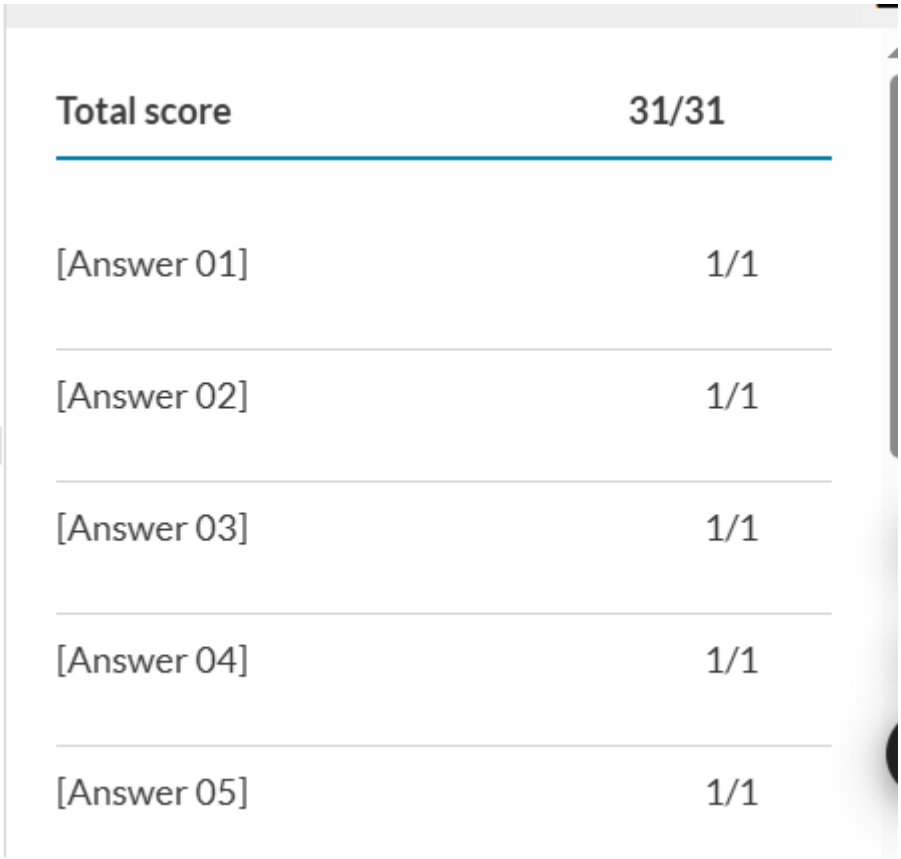



*Figure 37: Completed task*

# K. Conclusion

During this challenge lab, I learned how to transform a single-server, single-Availability Zone (AZ) web application architecture into a highly available and scalable one on AWS. I gained hands-on experience by extending the network across multiple AZs through the creation and configuration of a new NAT Gateway and updating its associated route table.

The core learning involved implementing elasticity and high availability by creating an EC2 Launch Template and integrating it with an Auto Scaling group. Finally, I learned how to distribute traffic to the application servers in private subnets by creating an Application Load Balancer (ALB), linking it to the Auto Scaling group via a target group, and successfully validating the system's ability to scale based on CPU utilization to handle a massive traffic spike.