# Automating Infrastructure Deployment

## Problem Statement

Modern cloud environments require automated, repeatable, and version-controlled infrastructure deployment. Managing networking and application resources manually across multiple AWS services and Regions is time-consuming and prone to configuration errors. There is a need for a centralized, automated approach to deploy, update, and replicate infrastructure using Infrastructure-as-Code (IaC) mechanisms.

## Objectives

After completing this lab, learners will be able to:

1. Deploy a VPC networking layer using AWS CloudFormation templates.
2. Deploy an application layer through CloudFormation to create and manage application resources.
3. Use Git and AWS CodeCommit to version-control CloudFormation templates and trigger stack creation and updates.
4. Automate stack deployments and updates using AWS CodePipeline.
5. Replicate networking and application infrastructure across multiple AWS Regions using CloudFormation for consistent multi-Region deployments.

## Business Goal

Automate the deployment and management of the cafe’s static and dynamic website using AWS CloudFormation, while adopting version control (CodeCommit) and continuous delivery (CodePipeline). Later, duplicate the same infrastructure in another AWS Region.

# Phase 1 - Static Website Hosting with CloudFormation

## Task 1 : Connect to VS Code IDE

1. Copy LabIDEURL and LabIDEPassword from AWS Details.
2. Open the VS Code IDE in a new browser tab and log in.

## Task 2 : Create CloudFormation Template for S3 Bucket

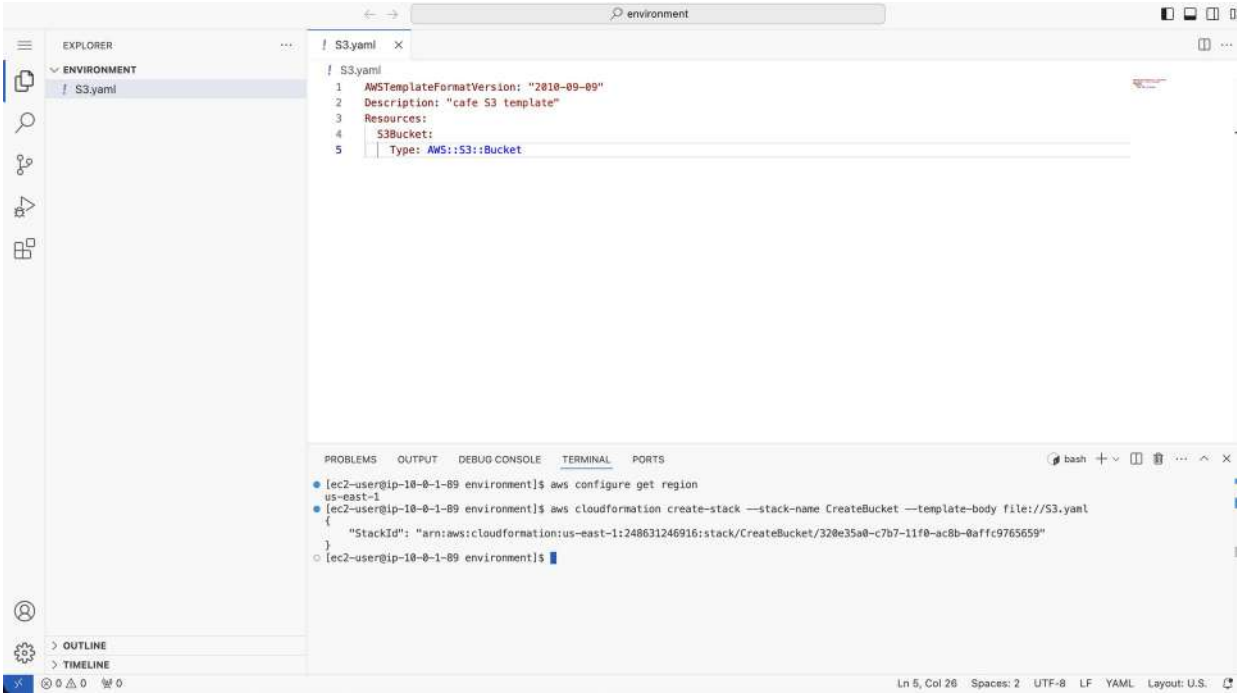
1. Create file S3.yaml.
2. Add the following yaml code snippet

```
AWSTemplateFormatVersion: "2010-09-09"
Description: cafe S3 template
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
```

3. Deploy stack using AWS CLI:

```
aws cloudformation create-stack --stack-name CreateBucket --template-body file://S3.yaml
```

4. Verify stack and bucket creation in CloudFormation + S3 console




## Task 3 : Configure Bucket as Static Website

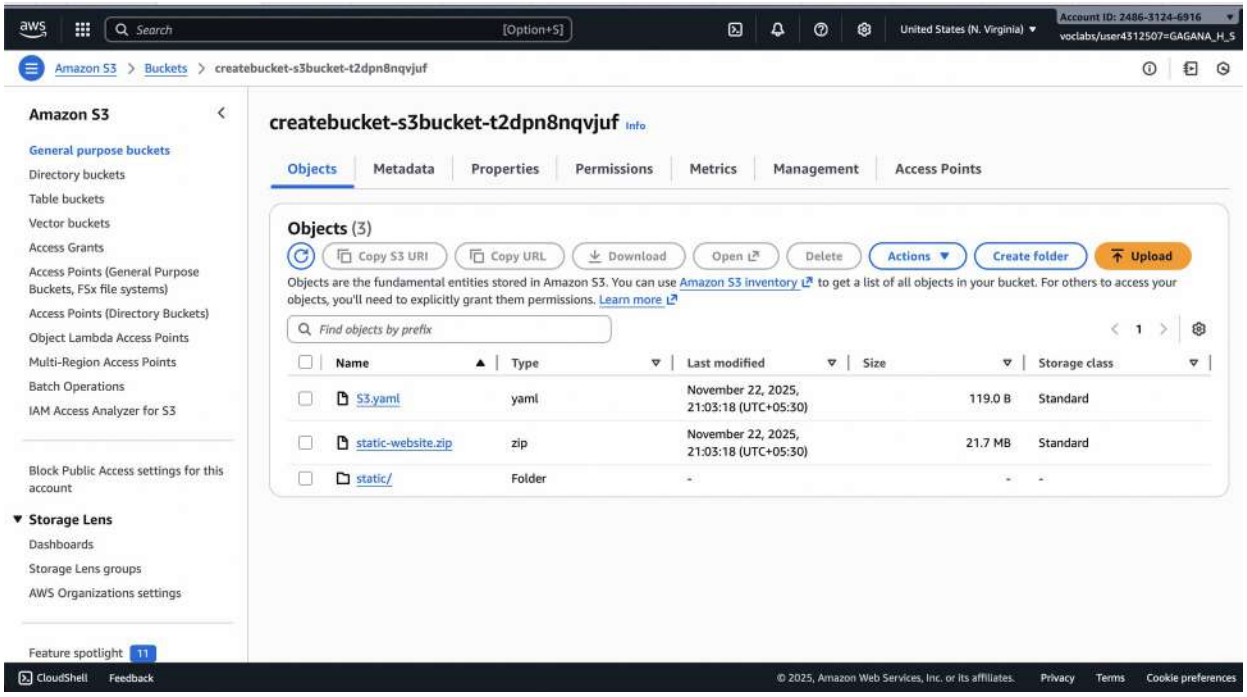
1. Download website assets → set S3 ownership & access → upload files.

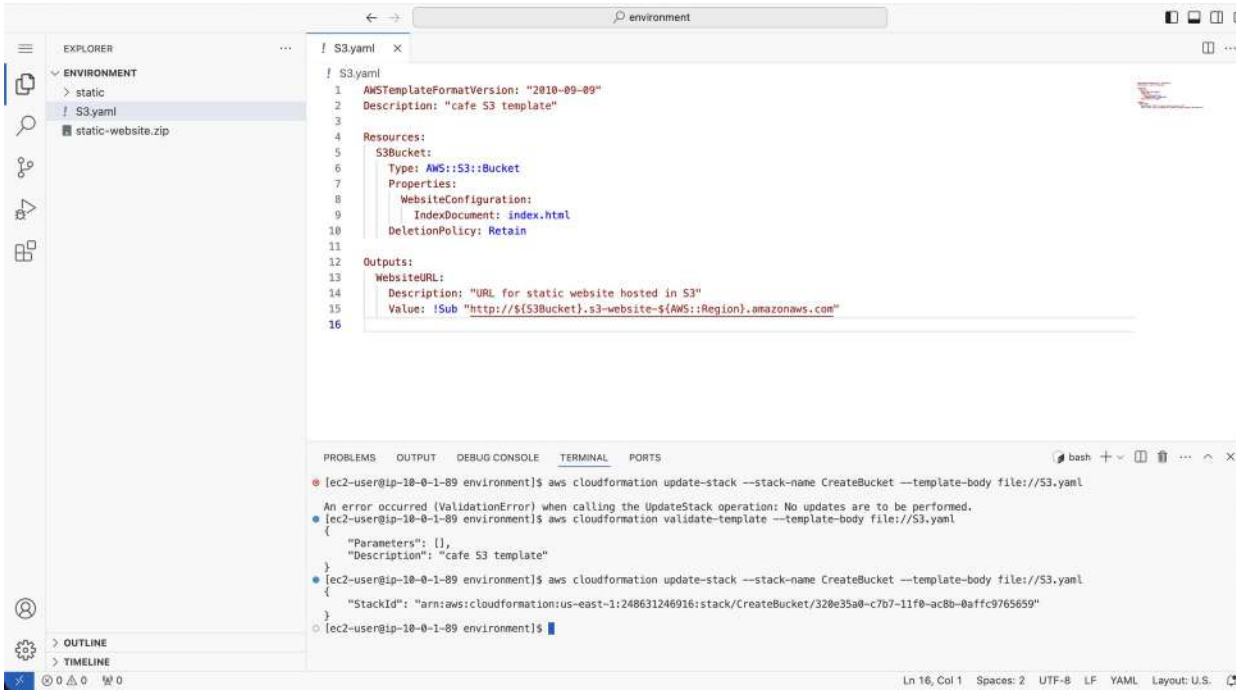
2. Modify template to:
    a. Add DeletionPolicy: Retain
    b. Enable static website hosting (index.html)
    c. Add Output for WebsiteURL

3. Validate and update the stack:

aws cloudformation validate-template --template-body file://S3.yaml
aws cloudformation update-stack --stack-name CreateBucket --template-body file://S3.yaml

4. Test website URL from Outputs tab




# Phase 2 - Version Control with CodeCommit

### Task 4 : Clone CodeCommit Repository

1. Identify repository: CFTemplatesRepo
2. Clone using HTTPS (GRC) URL:

   git clone <CodeCommit-URL>
3. Check repo state: git status

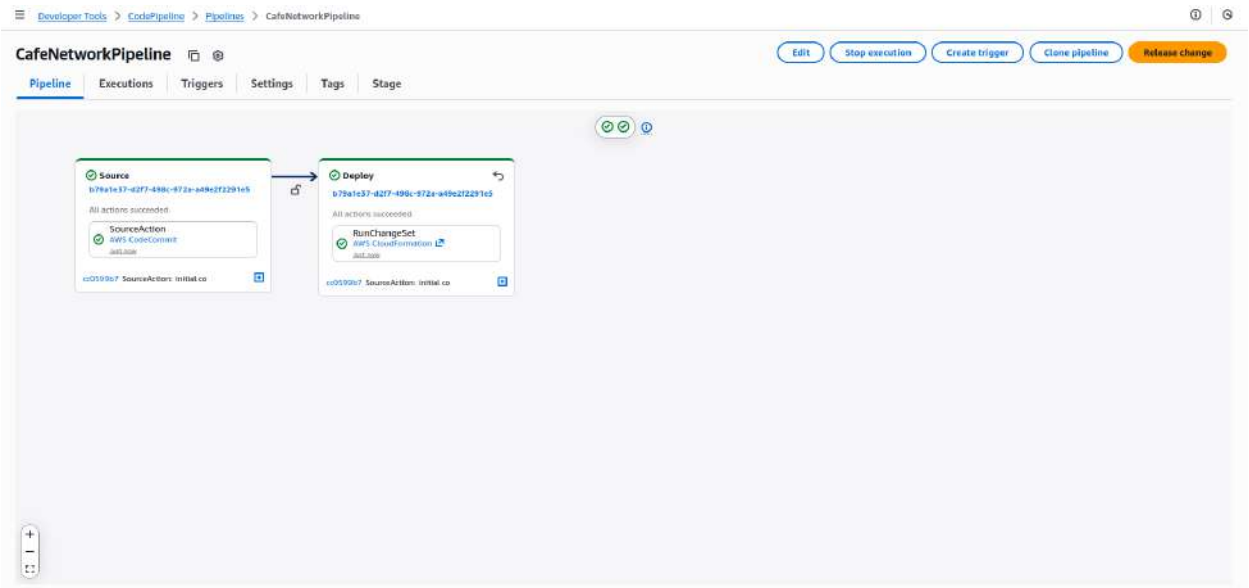
# Phase 3 - CI/CD Deployment of Network & Application

### Task 5 : Create Network Layer via CloudFormation + CodePipeline

1. Duplicate template → rename cafe-network.yaml
2. Edit description: *Network layer for the cafe*
3. Commit and push:

   ```
   git add templates/cafe-network.yaml
   git commit -m "initial commit of network template"
   git push
   ```

4. CafeNetworkPipeline triggers automatically → creates update-cafe-network stack
5. Verify VPC & Subnet in VPC console




## Task 6 : Add Outputs to Network Template

1. Add VpcId and PublicSubnet Outputs including Export Names.
2. Commit and push → Pipeline updates stack.
3. Verify Outputs in CloudFormation.

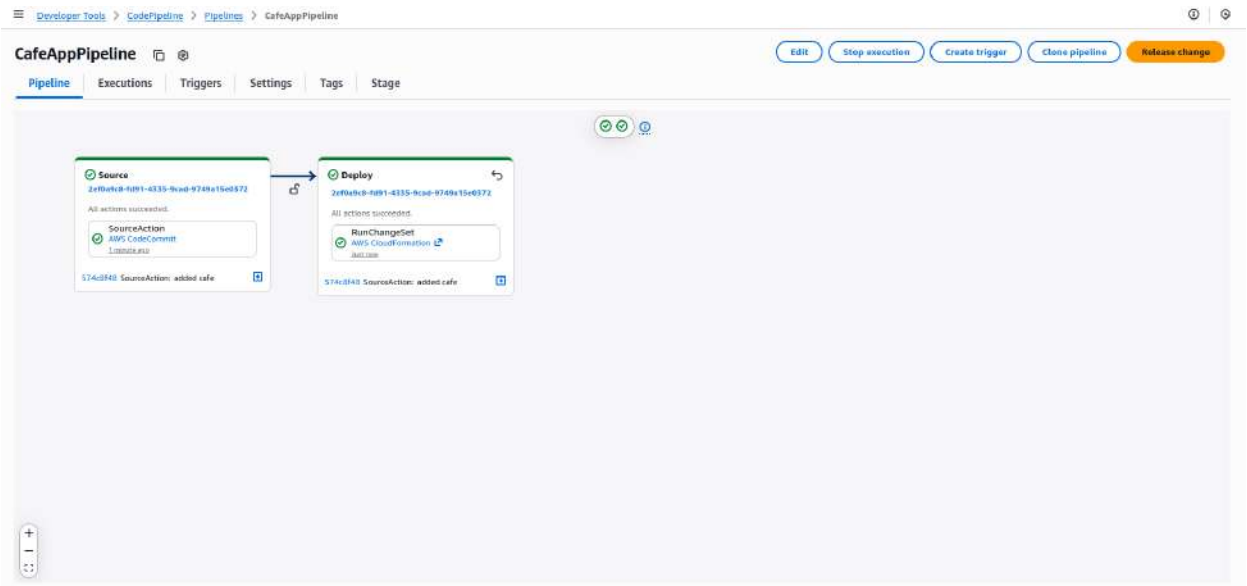
## Task 7 : Create Application EC2 Stack via CI/CD

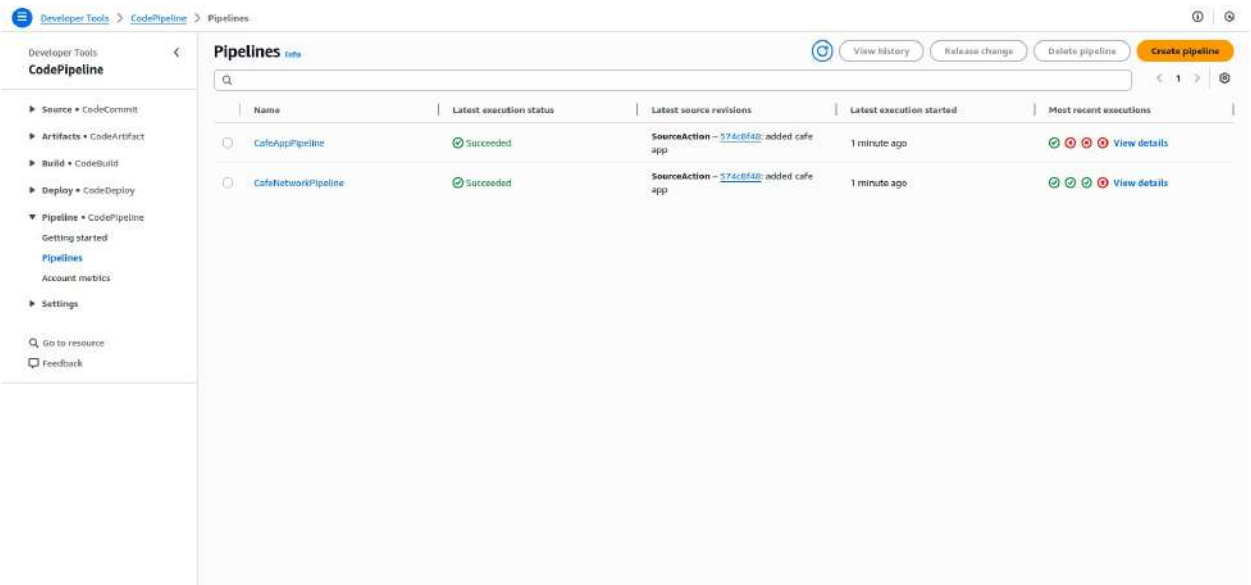
1. Duplicate template2.yaml → rename cafe-app.yaml
2. Add new parameter for InstanceType
3. Add EC2 instance resource using:
    a. LatestAmiId
    b. InstanceType parameter
    c. IAM Role: CafeRole
    d. Mapping-based KeyName
    e. Tag: Name: Cafe Web Server
    f. Complete UserData installation script
4. Validate → git add → commit → push

5. CafeAppPipeline runs → creates update-cafe-app stack

6. Verify EC2 instance and test site:

   http://<public-ip-address>/cafe




# Phase 4 - Replicate Infrastructure in a Second Region

## Task 8 : Duplicate Network + Application in us-west-2

1. Create network stack in new region:

aws cloudformation create-stack --stack-name update-cafe-network --template-body file://.../cafe-network.yaml --region us-west-2

2. Create key pair: cafe-oregon.

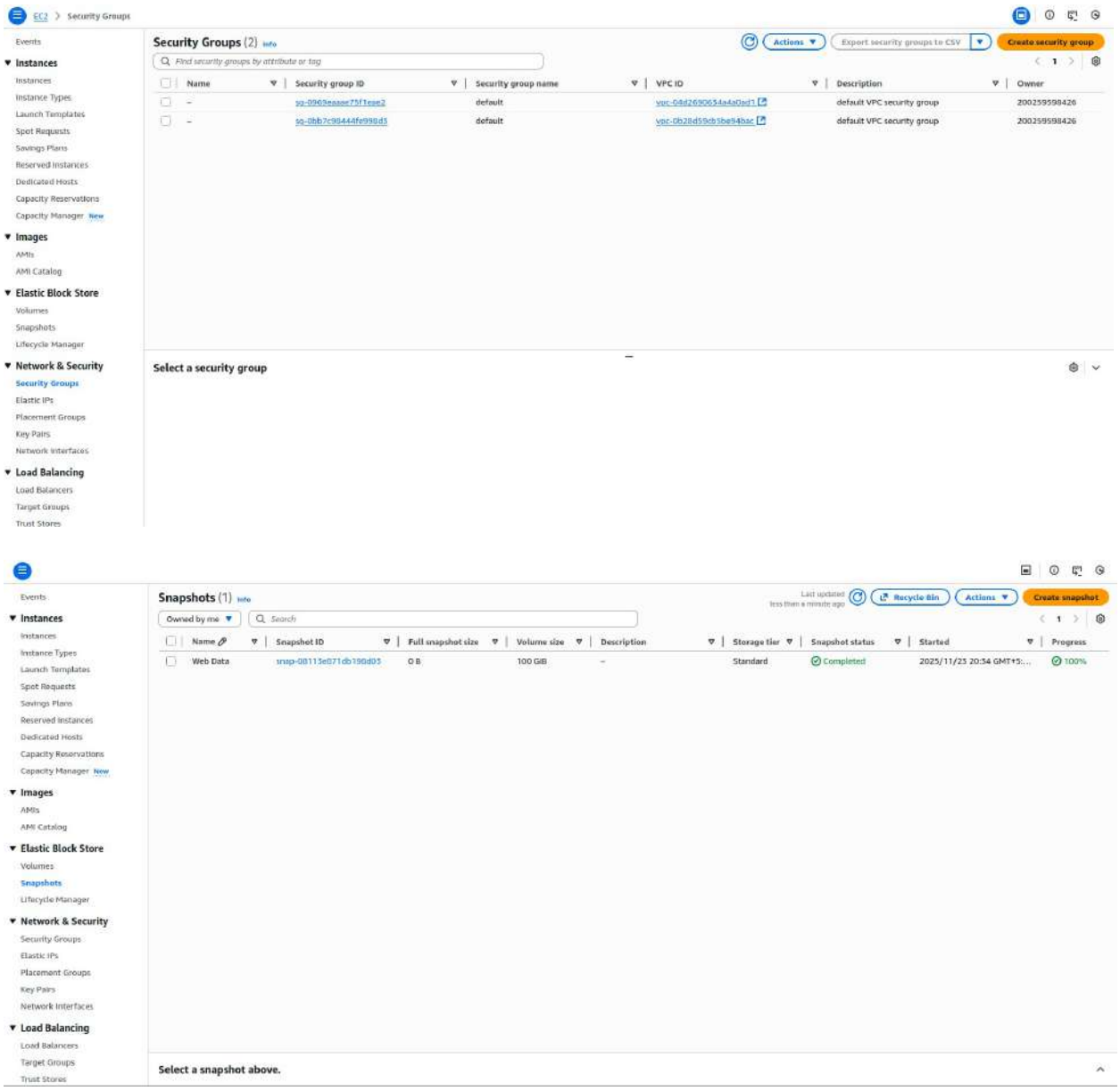
3. Upload cafe-app.yaml to S3 and copy Object URL.

4. In CloudFormation (us-west-2) → create stack using S3 URL.

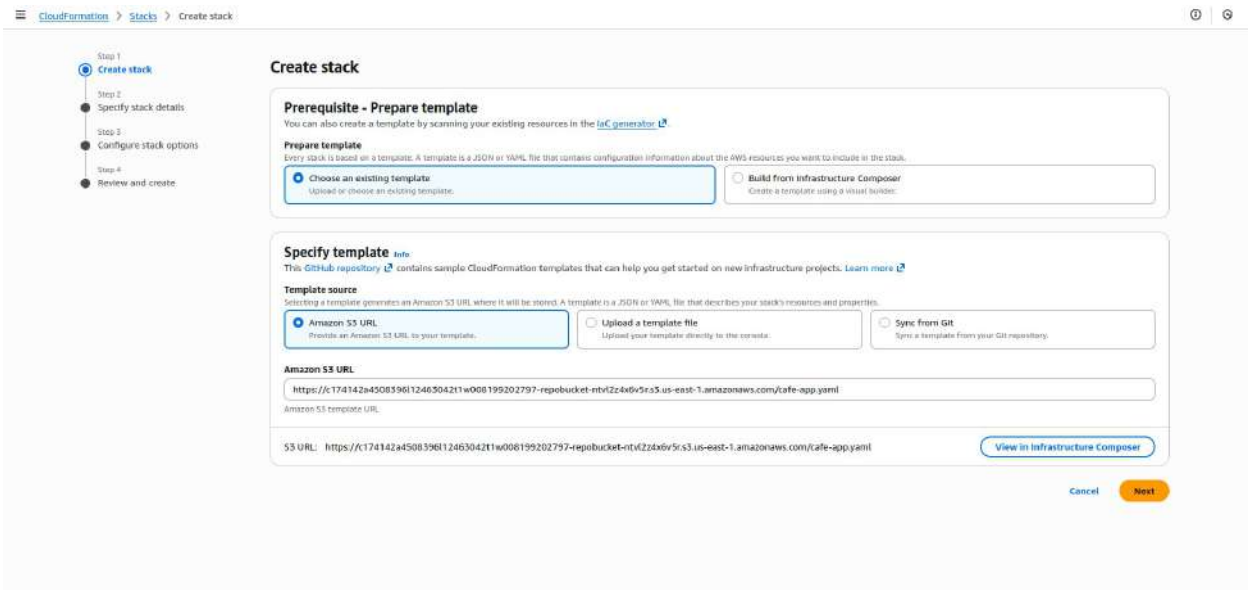
5. Choose InstanceType: t3.micro (via parameter)

6. Verify EC2 instance, wait for bootstrap → Access website

   http://<public-ip-address>/cafe


EC2 > Security Groups
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Capacity Manager New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores
Security Groups (2) Info
Find security groups by attribute or tag
Actions
Export security groups to CSV
Create security group
Name
Security group ID
Security group name
VPC ID
Description
Owner
sg-0969eaaae75f1eee2
default
vpc-04d2690654a4a0ad1
default VPC security group
200259598426
sg-0bb7c98444fe998d3
default
vpc-0b28d59cb5be84bac
default VPC security group
200259598426
Select a security group


Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Capacity Manager New
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
Load Balancing
Load Balancers
Target Groups
Trust Stores
Snapshots (1) Info
Last updated less than a minute ago
Recycle Bin
Actions
Create snapshot
Owned by me
Search
Name
Snapshot ID
Full snapshot size
Volume size
Description
Storage tier
Snapshot status
Started
Progress
Web Data
snap-08113e071db196d03
0 B
100 GiB
–
Standard
Completed
2025/11/25 20:34 GMT+5:...
100%
Select a snapshot above.


CloudFormation > Stacks > Create stack
Step 1
Create stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review and create
Create stack
Prerequisite - Prepare template
You can also create a template by scanning your existing resources in the IaC generator.
Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.
Choose an existing template
Upload or choose an existing template.
Build from Infrastructure Composer
Create a template using a visual builder.
Specify template Info
This GitHub repository contains sample CloudFormation templates that can help you get started on new infrastructure projects. Learn more
Template source
Selecting a template generates an Amazon S3 URL where it will be stored. A template is a JSON or YAML file that describes your stack's resources and properties.
Amazon S3 URL
Provide an Amazon S3 URL to your template.
Upload a template file
Upload your template directly to the console.
Sync from Git
Sync a template from your Git repository.
Amazon S3 URL
https://c174142a4508396l1246304211w008199202797-repobucket-ntvl2z4x6v5r.s3.us-east-1.amazonaws.com/cafe-app.yaml
Amazon S3 template URL
S3 URL: https://c174142a4508396l1246304211w008199202797-repobucket-ntvl2z4x6v5r.s3.us-east-1.amazonaws.com/cafe-app.yaml
View in Infrastructure Composer
Cancel
Next

# Automating Infrastructure Deployment with AWS CloudFormation

## Objectives:

Deploying infrastructure in a consistent, reliable manner is difficult. It requires people to follow documented procedures without taking any undocumented shortcuts. It can also be difficult to deploy infrastructure after hours when fewer staff are available. AWS CloudFormation changes this situation by defining infrastructure in a template that can be automatically deployed—even on an automated schedule.

### Task 1: Deploying a networking layer

At the top of the AWS Management Console, in the search box, search for and choose CloudFormation.

Choose Create stack > With new resources (standard) and configure these settings:

**Step 1:** Create stack

Prepare template: Choose  Template is ready.

Template source: Choose Upload a template file > Choose file, and then choose the lab-network.yaml file that you downloaded.

Choose Next.

**Step 2:** Specify stack details

Stack name: lab-network

Choose Next.

**Step 3:** Configure stack options

In the Tags section, choose Add new tag and configure the following:

Key: application

Value: inventory

Choose Next.

**Step 4:** Review and create

Choose Submit.

Choose the Stack info tab.

Wait for the Status to change to CREATE_COMPLETE.

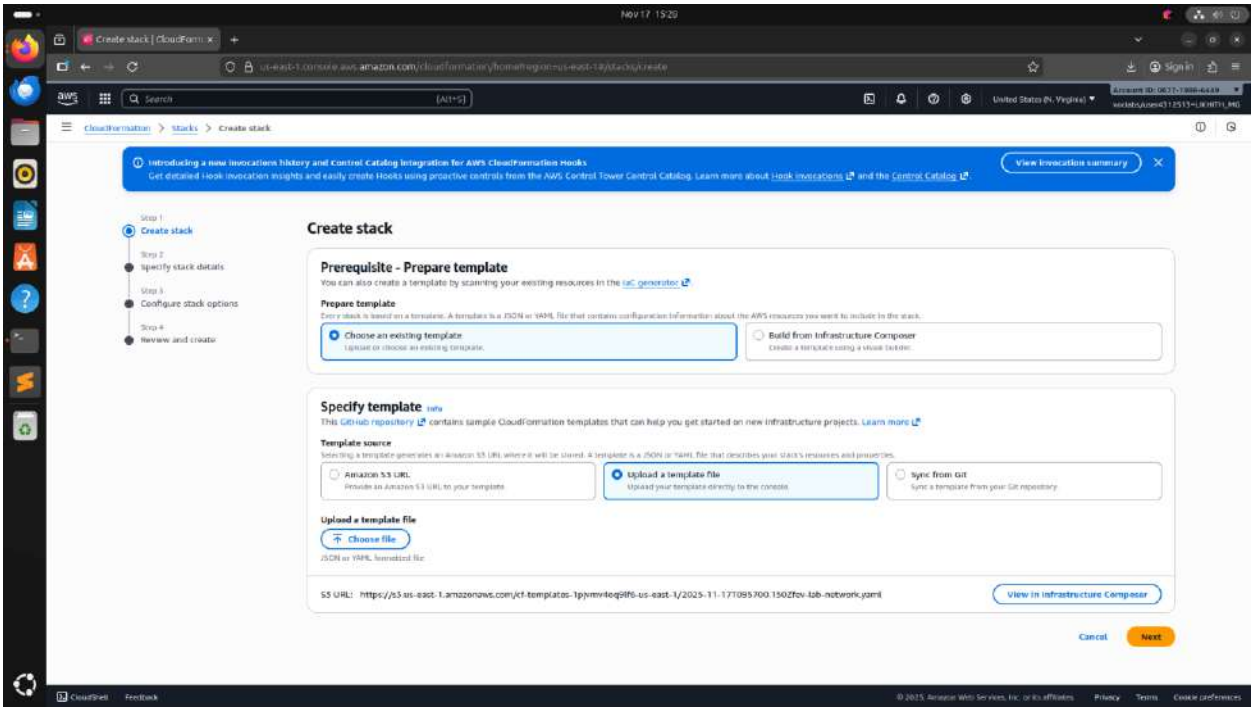
Choose Refresh every 15 seconds to update the display, if necessary.

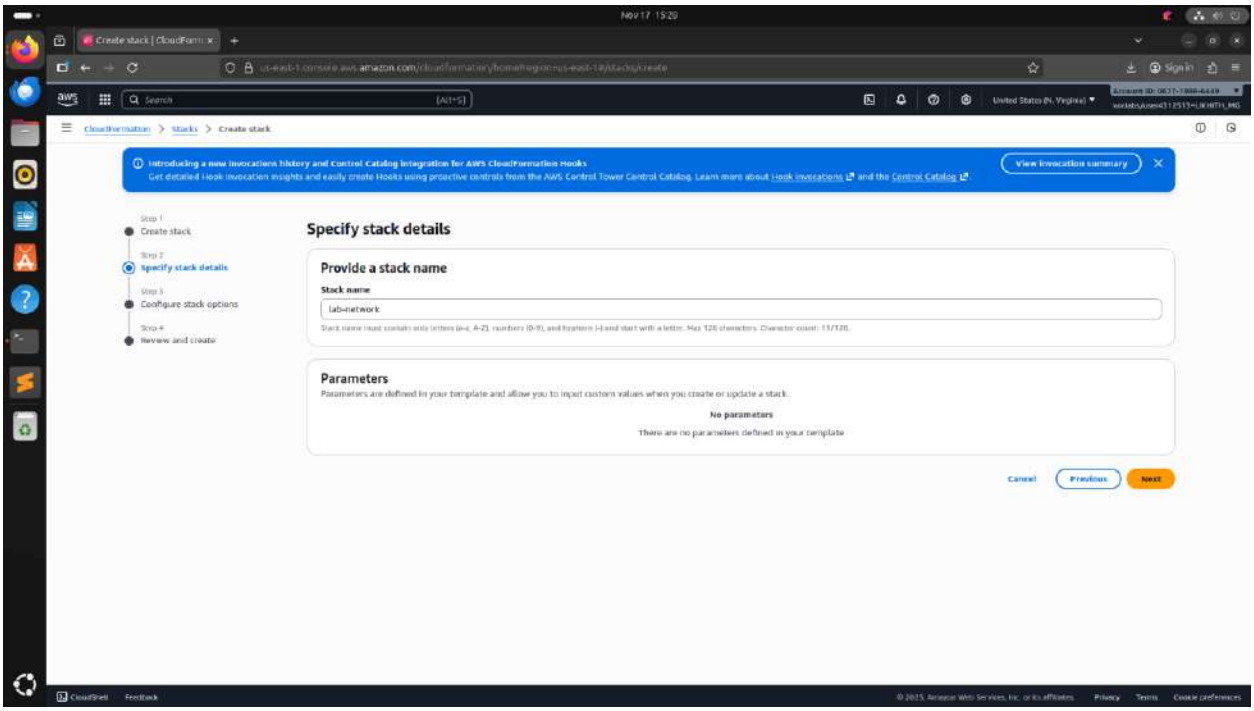
You can now examine the resources that were created.

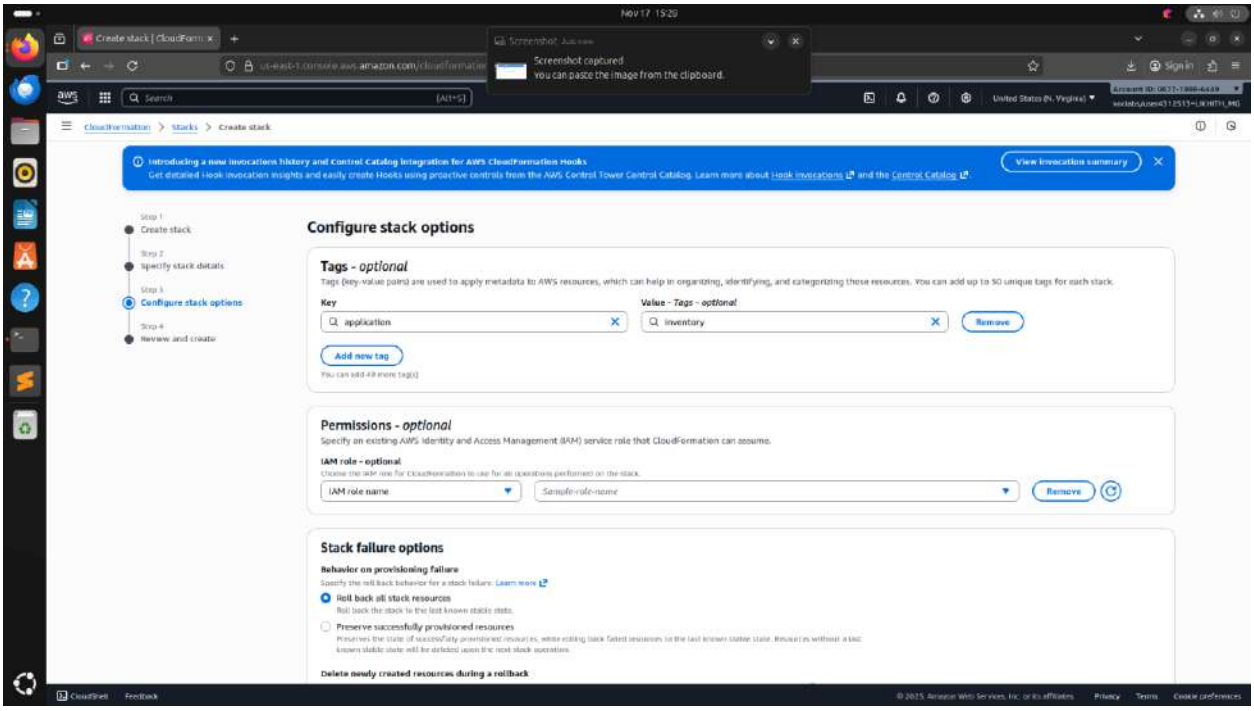
Choose the Resources tab.

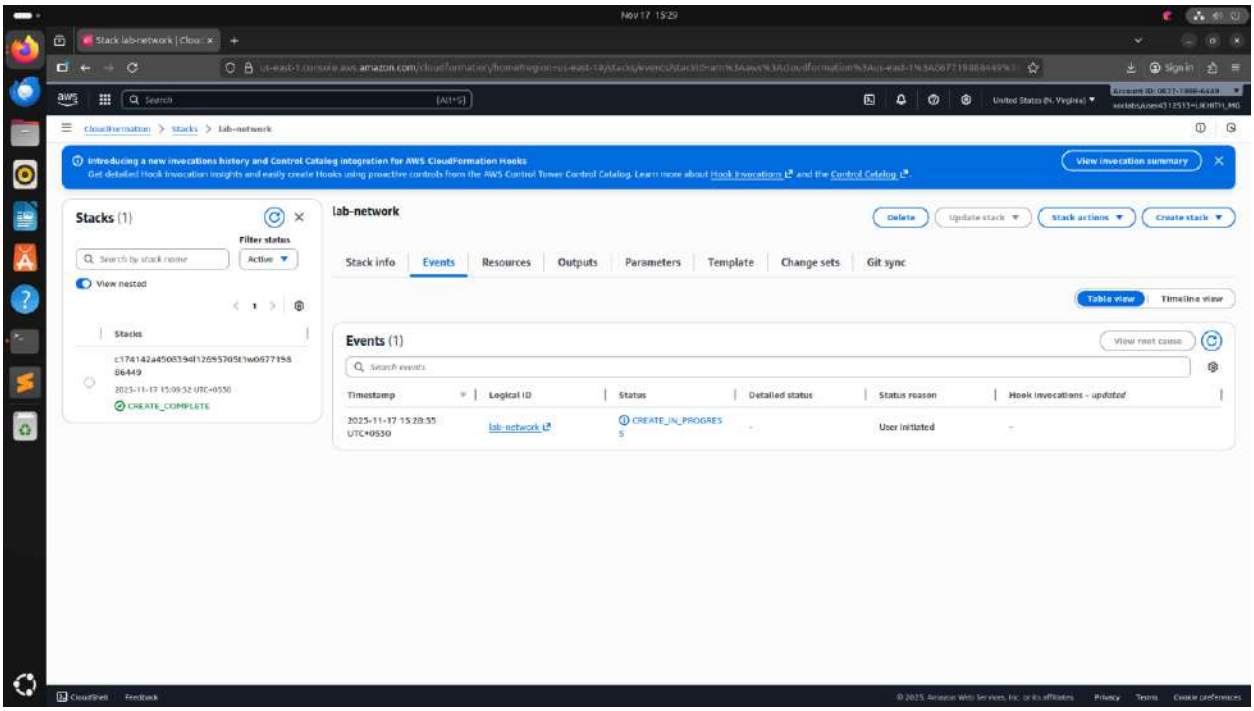
You see a list of the resources that were created by the template.

If the list is empty, update the list by choosing Refresh .




Nov 17 15:29
Create stack | CloudForm
Search
[Alt+S]
United States (N. Virginia)
CloudFormation > Stacks > Create stack
Introducing a new invocations history and Control Catalog integration for AWS CloudFormation Hooks
Get detailed Hook invocation insights and easily create Hooks using proactive controls from the AWS Control Tower Control Catalog. Learn more about Hook invocations and the Control Catalog.
View invocation summary
Step 1
Create stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review and create
Specify stack details
Provide a stack name
Stack name
lab-network
Stack name must contain only letters (a-z, A-Z), numbers (0-9), and hyphens (-) and start with a letter. Max 128 characters. Character count: 11/128.
Parameters
Parameters are defined in your template and allow you to input custom values when you create or update a stack.
No parameters
There are no parameters defined in your template
Cancel
Previous
Next
CloudShell
Feedback
Privacy
Terms
Cookie preferences


Nov 17 15:29
Screenshot
Screenshot captured
You can paste the image from the clipboard.
Create stack | CloudForm
Search
[Alt+S]
United States (N. Virginia)
CloudFormation > Stacks > Create stack
Introducing a new invocations history and Control Catalog integration for AWS CloudFormation Hooks
Get detailed Hook invocation insights and easily create Hooks using proactive controls from the AWS Control Tower Control Catalog. Learn more about Hook invocations and the Control Catalog.
View invocation summary
Step 1
Create stack
Step 2
Specify stack details
Step 3
Configure stack options
Step 4
Review and create
Configure stack options
Tags - optional
Tags (key-value pairs) are used to apply metadata to AWS resources, which can help in organizing, identifying, and categorizing those resources. You can add up to 50 unique tags for each stack.
Key
Value - Tags - optional
application
inventory
Remove
Add new tag
You can add 49 more tag(s)
Permissions - optional
Specify an existing AWS Identity and Access Management (IAM) service role that CloudFormation can assume.
IAM role - optional
IAM role name
Sample-role-name
Remove
Stack failure options
Behavior on provisioning failure
Roll back all stack resources
Roll back the stack to the last known stable state.
Preserve successfully provisioned resources
Delete newly created resources during a rollback
CloudShell
Feedback
Privacy
Terms
Cookie preferences




## Task 2: Deploying an application layer

Create stack > With new resources (standard), and then configure these settings:

**Step 1:** Create Stack

Prepare template: Choose  Template is ready.

Template source: Choose Upload a template file > Choose file, and then choose the lab-application.yaml file that you downloaded.

Choose Next.

**Step 2:** Specify stack details

Stack name: lab-application

Notice the NetworkStackName: lab-network

Choose Next.

The Network Stack Name parameter tells the template the name of the first stack that you created (lab-network), so it can retrieve values from the outputs.

**Step 3:** Configure stack options

In the Tags section, choose Add new tag and configure the following:

Key: application

Value: inventory

Choose Next.

**Step 4:** Review and create

Choose Submit.

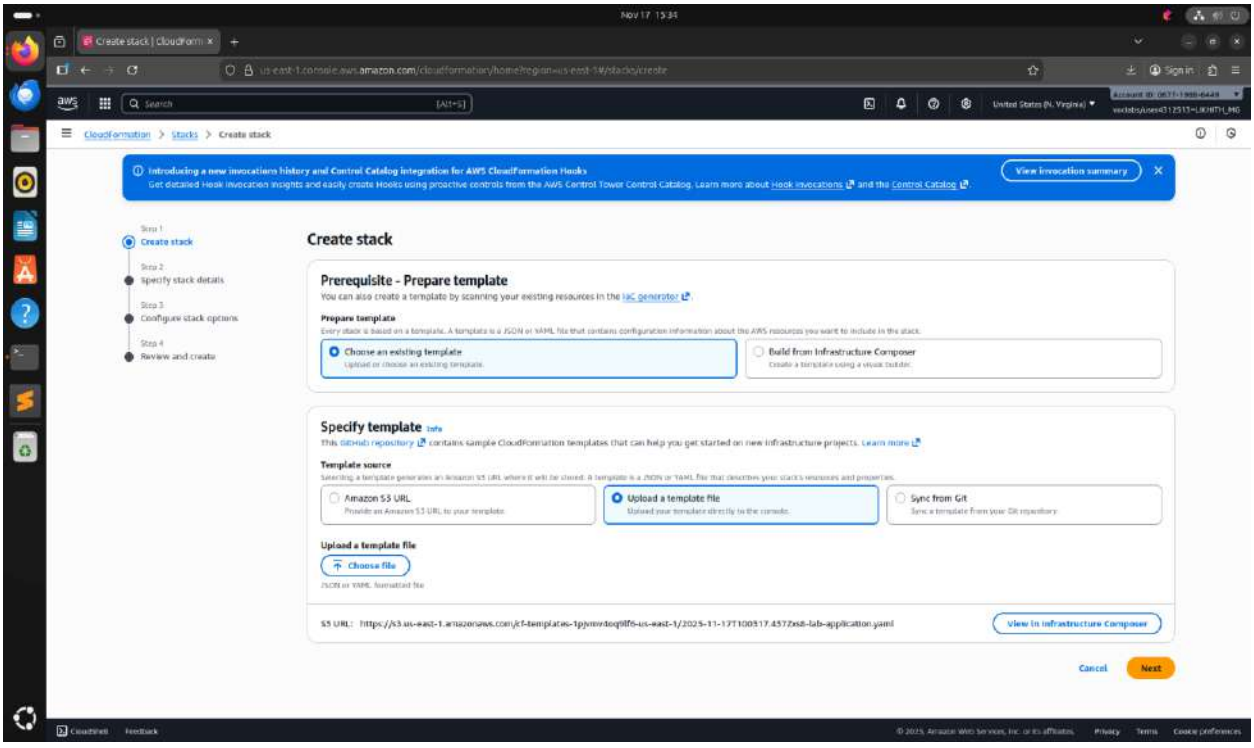
While the stack is being created, examine the details in the Events tab and the Resources tab. You can monitor the progress of the resource-creation process and the resource status.

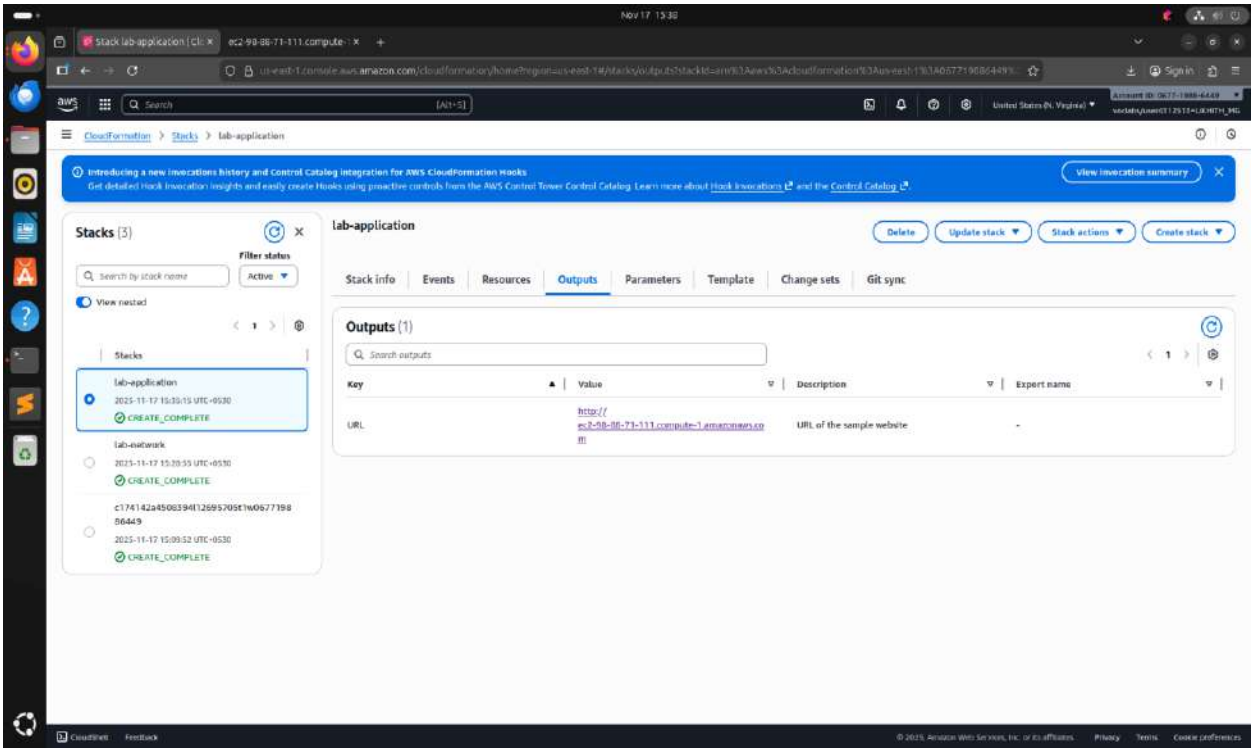
In the Stack info tab, wait for the Status to change to CREATE_COMPLETE.

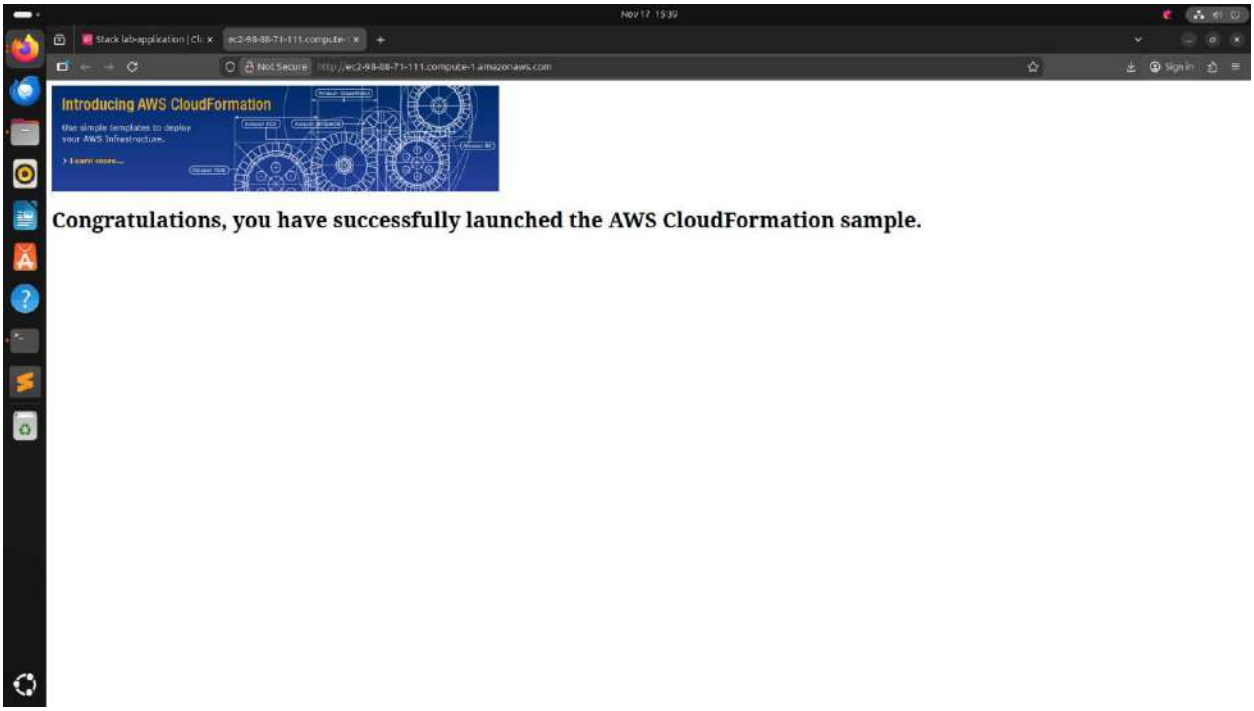
Your application is now ready!

Choose the Outputs tab.

Copy the URL that is displayed, open a new web browser tab, paste the URL, and press ENTER.






## Task 3: Updating a Stack

At the top of the AWS Management Console, in the search box, search for and choose EC2.

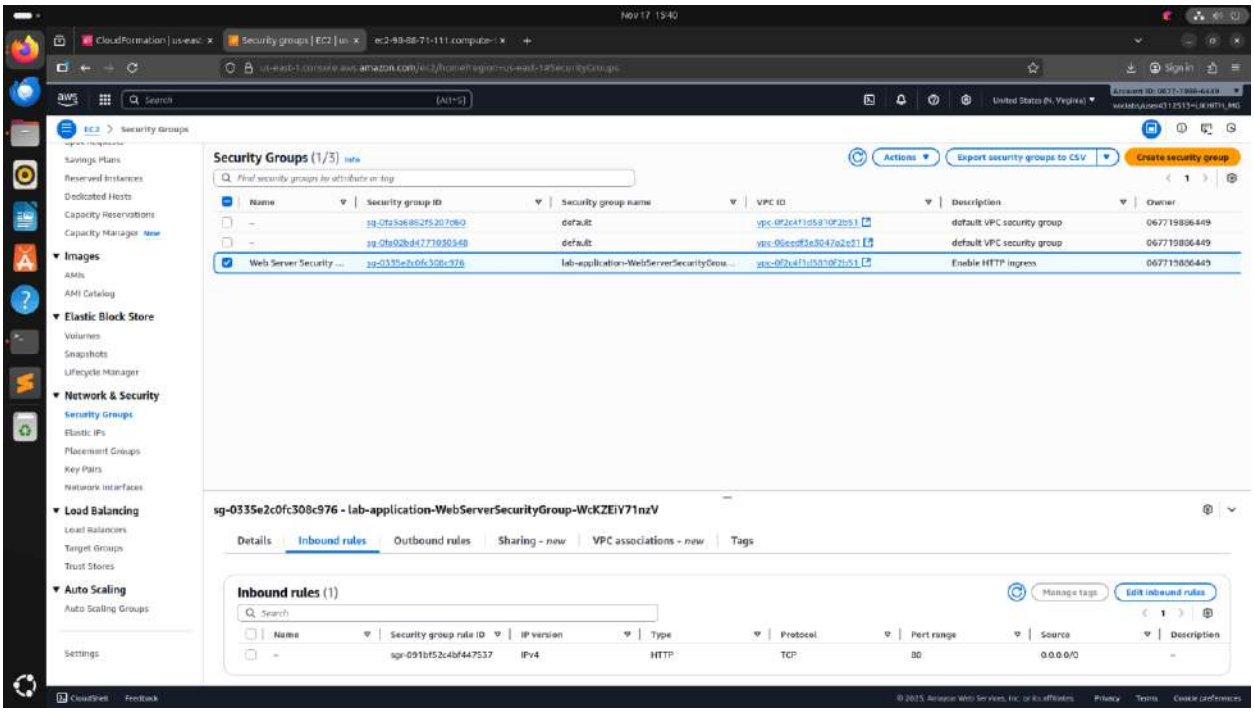
In the left navigation pane, in the Network & Security section, choose Security Groups.

Select the check box for lab-application-WebServerSecurityGroup.

Choose the Inbound rules tab.

Currently, only one rule is in the security group. The rule permits HTTP traffic.

You now return to AWS CloudFormation to update the stack.




From the Services menu at the top, choose CloudFormation.

Open the context (right-click) menu for the following link and download the updated template to your computer: lab-application2.yaml

From the Stacks list of the AWS CloudFormation console, choose lab-application.

Choose Update and configure these settings:

Prepare template: Choose Replace current template.

Template source: Choose Upload a template file.

Upload a template file: Choose file, and then choose the lab-application2.yaml file that you downloaded.

Choose Next on each of the next three screens to go to the Review lab-application page.

In the Change set preview section at the bottom of the page, AWS CloudFormation displays the following resources that will be updated:

Choose Submit.

In the Stack info tab, wait for the Status to change to UPDATE_COMPLETE.

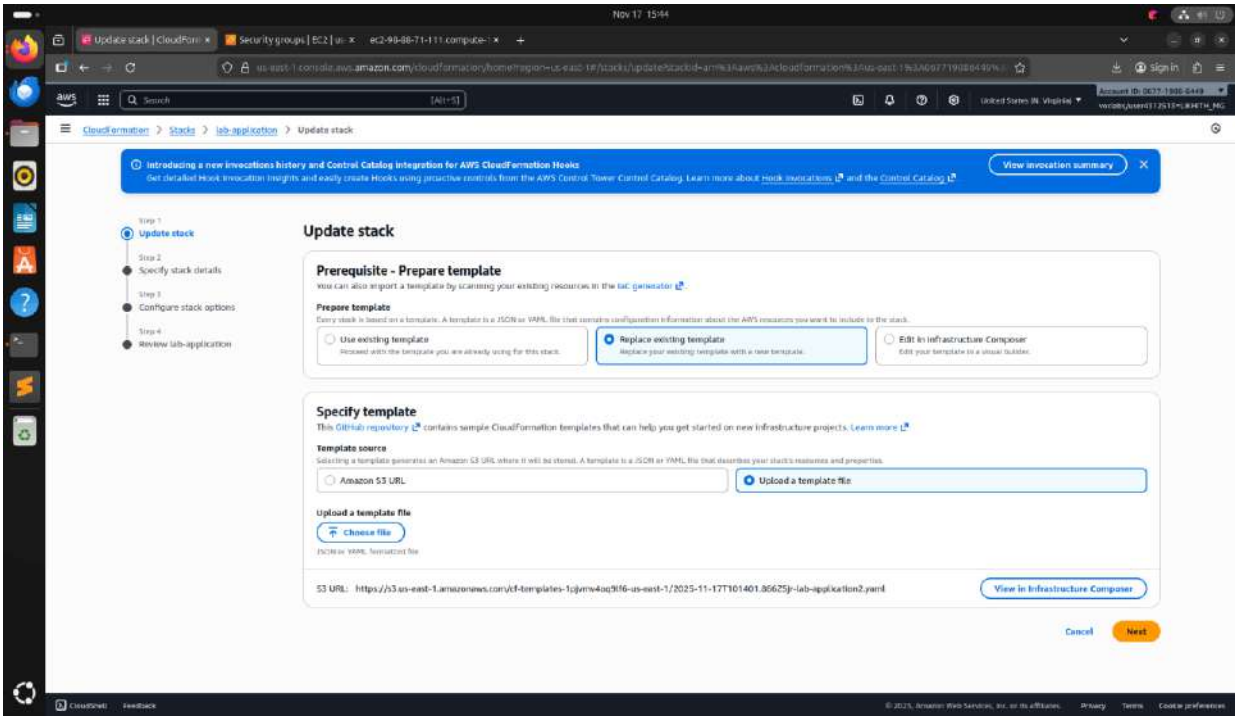
Update the status by choosing Refresh every 15 seconds, if necessary.

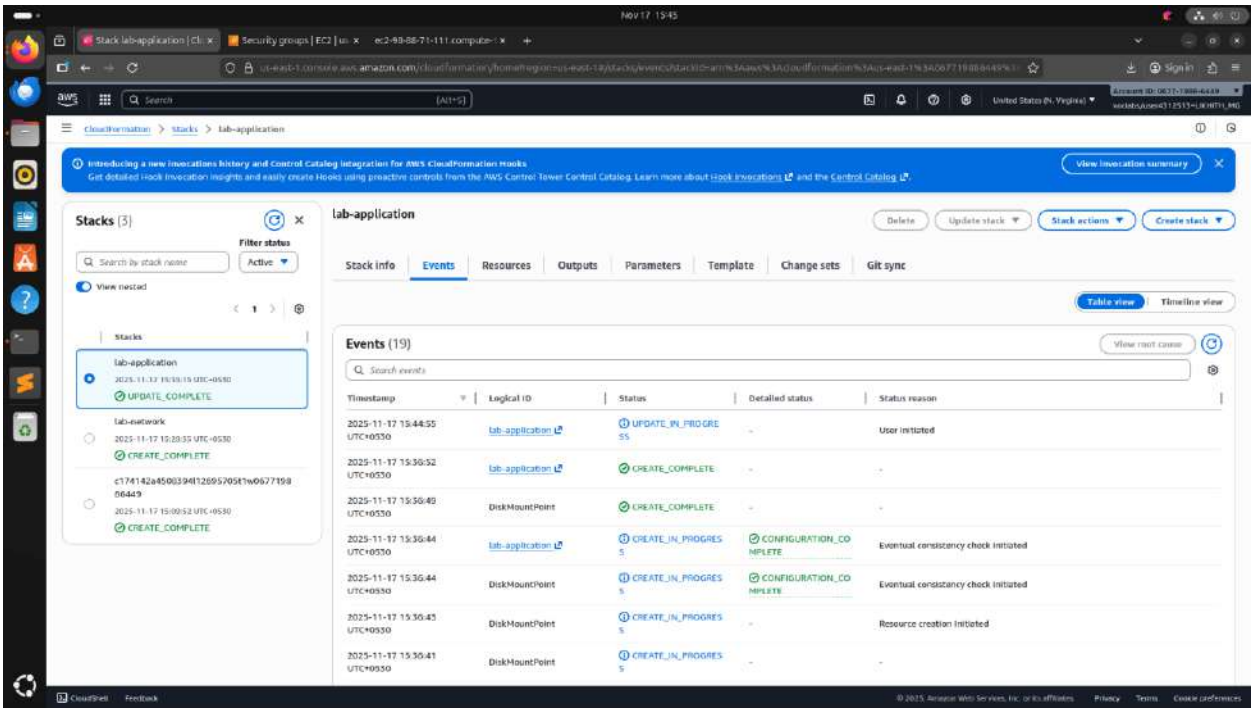
You can now verify the change.

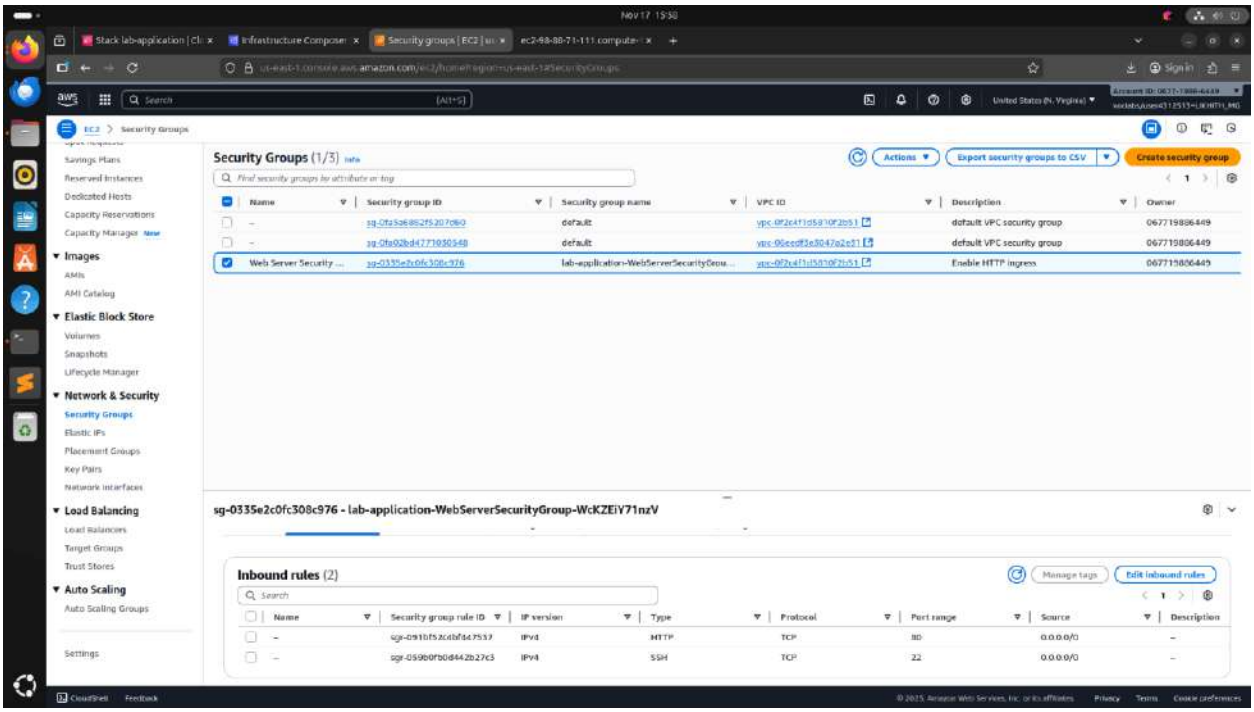
Return to the Amazon EC2 console, and from the left navigation pane, choose Security Groups.

In the Security Groups list, choose lab-application-WebServerSecurityGroup.

The Inbound rules tab should display an additional rule that allows HTTPS traffic over TCP port 443.

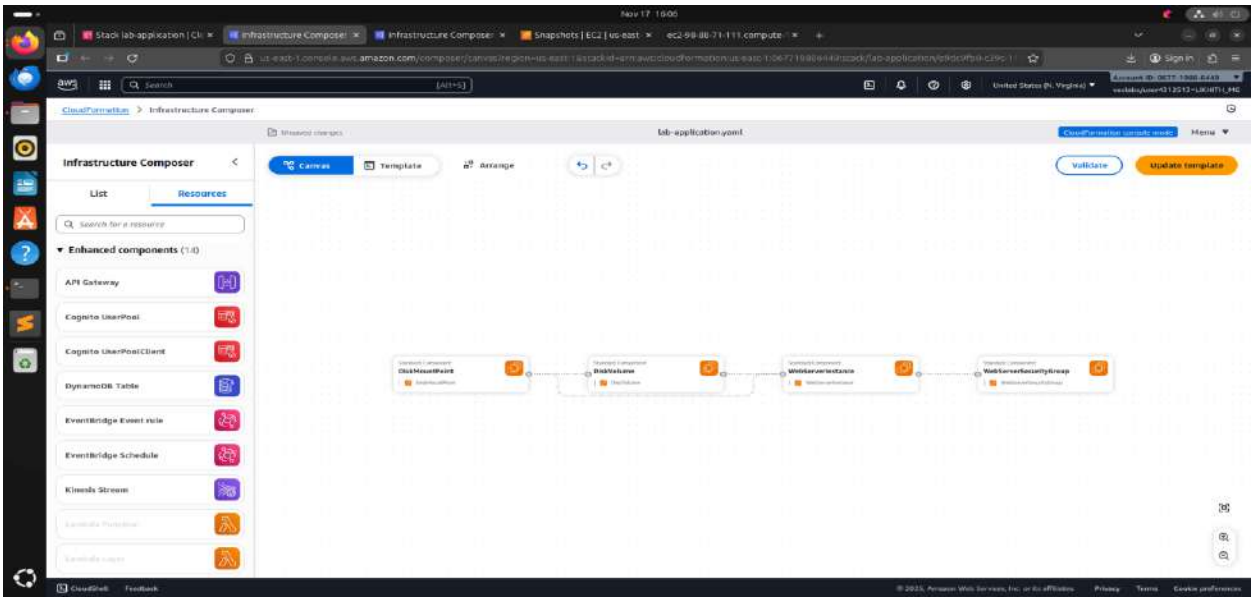





# Task 4: Exploring templates with AWS CloudFormation Designer

Cloud formation -> SELECT Stack ->  Template -> SELECT View Infrastructure Composer




## Task 5: Deleting the stack

AWS CloudFormation console by choosing Close at the top of the Designer page (choose Leave page if prompted).

In the list of stacks, choose the lab-application link.

Choose Delete.

On the Delete stack? dialog box, choose Delete.

You can monitor the deletion process in the Events tab and update the screen by choosing Refresh occasionally. You might also see an events log entry that indicates that the EBS snapshot is being created.

Wait for the stack to be deleted. It will disappear from the stacks list.

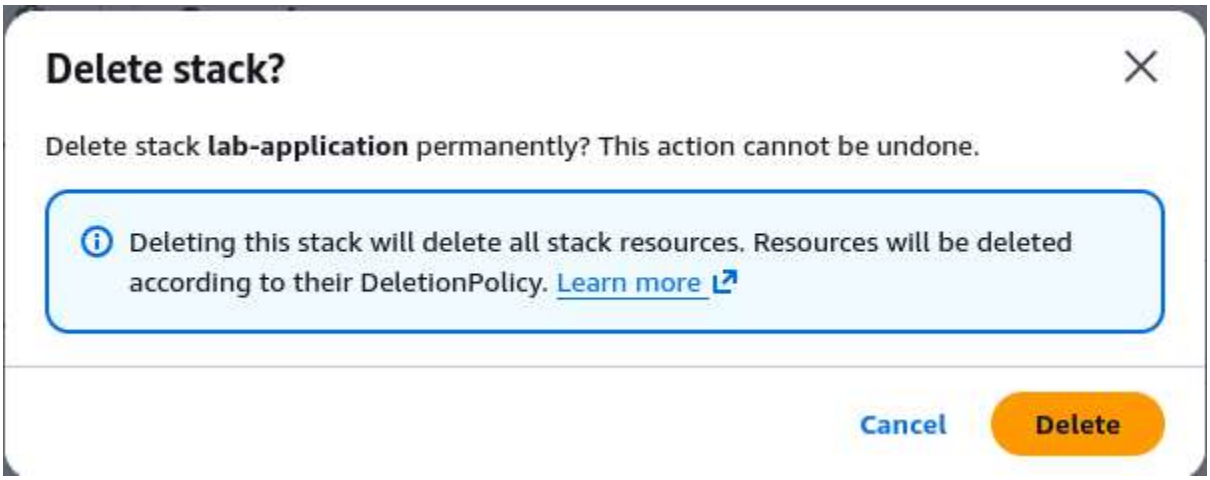
The application stack was removed, but the network stack remained untouched. This scenario reinforces the idea that different teams (for example, the network team or the application team) can manage their own stacks.

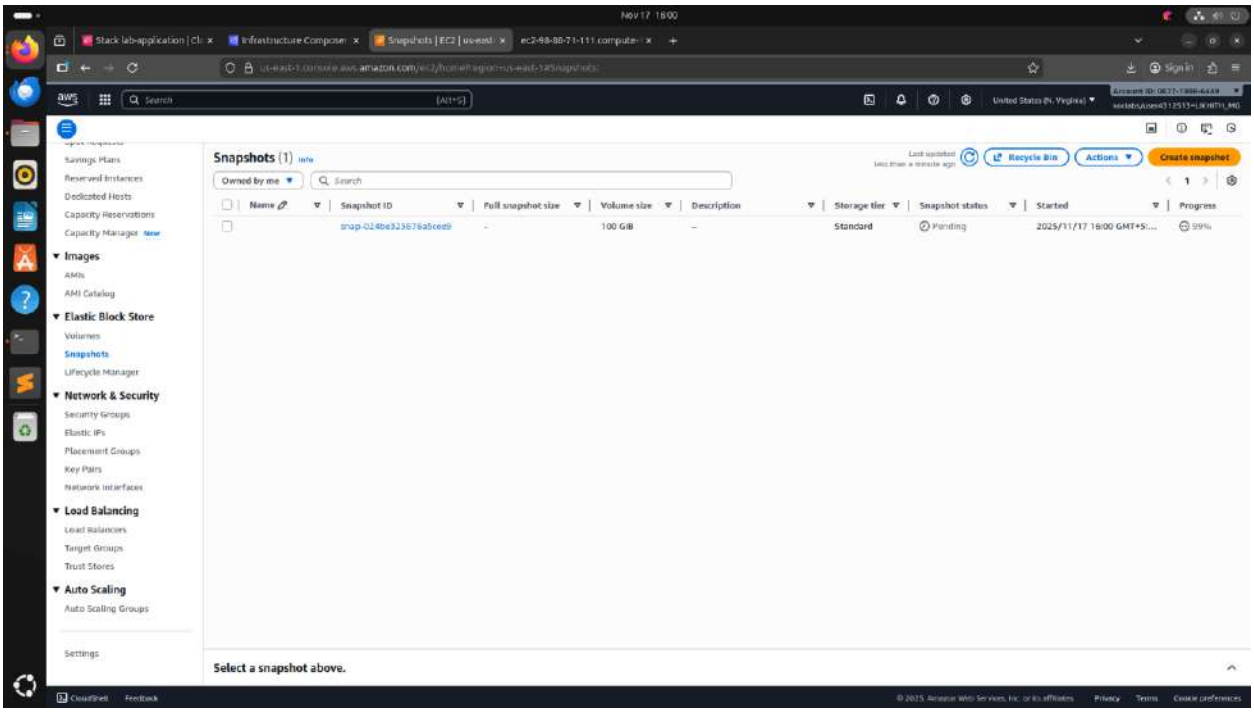
You will now verify that a snapshot of the EBS volume was created before the EBS volume was deleted.

From the Services menu, choose EC2.

In the left navigation pane, in the Elastic Block Store section, choose Snapshots.

You see a snapshot Web Data with a Started time in the last few minutes, and it changes to Completed soon.


Delete stack?
Delete stack lab-application permanently? This action cannot be undone.
Deleting this stack will delete all stack resources. Resources will be deleted according to their DeletionPolicy. Learn more
Cancel
Delete


Snapshots (1)
Recycle Bin
Actions
Create snapshot
Owned by me
Search
Name
Snapshot ID
Full snapshot size
Volume size
Description
Storage tier
Snapshot status
Started
Progress
100 GiB
Standard
Pending
Select a snapshot above.

# Amazon Elastic Compute Cloud (Amazon EC2)  Report

**Lab:** Introducing Amazon Elastic Compute Cloud(EC2)

## Objectives

- Launch a web server with termination protection enabled
- Monitor Your EC2 instance
- Modify the security group that your web server is using to allow HTTP access
- Resize your Amazon EC2 instance to scale and enable stop protection
- Explore EC2 limits
- Test stop protection
- Stop your EC2 instance

## Lab Environment Setup

- This lab provides you with a basic overview of launching, resizing, managing, and monitoring an Amazon EC2 instance.
- **Amazon Elastic Compute Cloud (Amazon EC2)** is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.
- Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change.
- Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios.

# Lab 3: Introduction to Amazon EC2

## Lab overview and objectives


# Task 1: Launch Your Amazon EC2 Instance

- In the **AWS Management Console** choose **Services**, choose **Compute** and then choose **EC2**.

- Choose the Launch instance  menu and select **Launch instance**.

**Step 1: Name and tags**

- Name the instance **Web Server**, which creates a tag with key **Name** and value **Web Server**.

**Step 2: AMI selection**

- Keep the default **Amazon Linux 2023** AMI selected from the Quick Start list.

**Step 3: Instance type**

- Keep the default **t2.micro** instance type with 1 vCPU and 1 GiB memory.

**Step 4: Key pair**

- Select the **vockey** key pair for secure login authentication.

**Step 5: Network settings**

- Select **Lab VPC**, use **PublicSubnet1**, create a security group named **Web Server security group**, and remove the default inbound rule.

**Step 6: Configure storage**

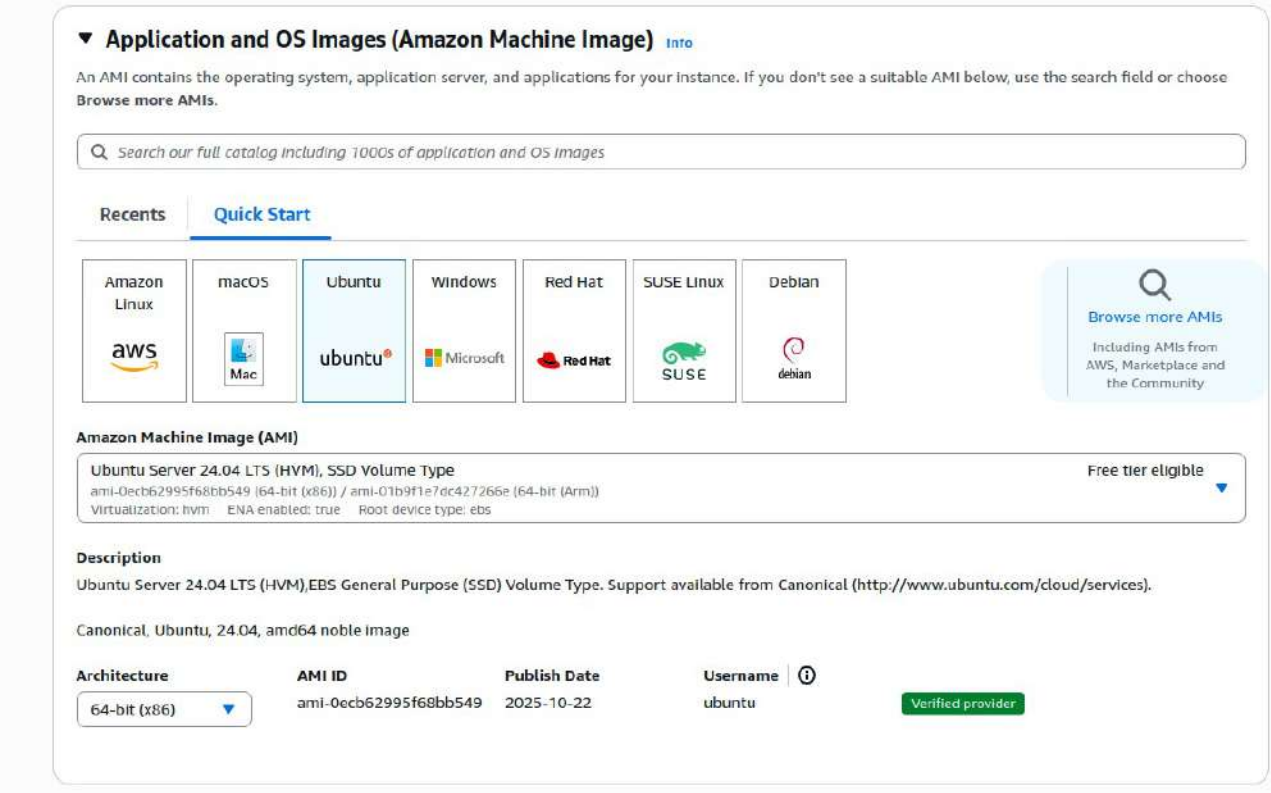
- Keep the default **8 GiB EBS root volume** unchanged.

**Step 7: Advanced details**

- Enable **termination protection** and add the provided user-data script to install and start the Apache web server.

**Step 8: Launch the instance**

- Choose **Launch instance**, then view your instance and wait until it reaches **Running** with **2/2 status checks passed**.

## Instance type Info | Get advice

**Instance type**

t2.micro
Family: t2 1 vCPU 1 GiB Memory Current generation: true On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour On-Demand Linux base pricing: 0.0116 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

## Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

**Key pair name - *required***

vockey

Create new key pair

## Network settings Info

**VPC - *required*** Info

vpc-0146cdfc86b20f546 (Lab VPC)
10.0.0.0/16

**Subnet** Info

subnet-00ca9a38f9763dc47 PublicSubnet1
VPC: vpc-0146cdfc86b20f546 Owner: 975050223293 Availability Zone: us-east-1a (use1-az1)
Zone type: Availability Zone IP addresses available: 1 CIDR: 10.0.1.0/28)

Create new subnet

**Auto-assign public IP** Info

Enable

Additional charges apply when outside of free tier allowance

**Firewall (security groups)** Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

- (•) Create security group
- ( ) Select existing security group

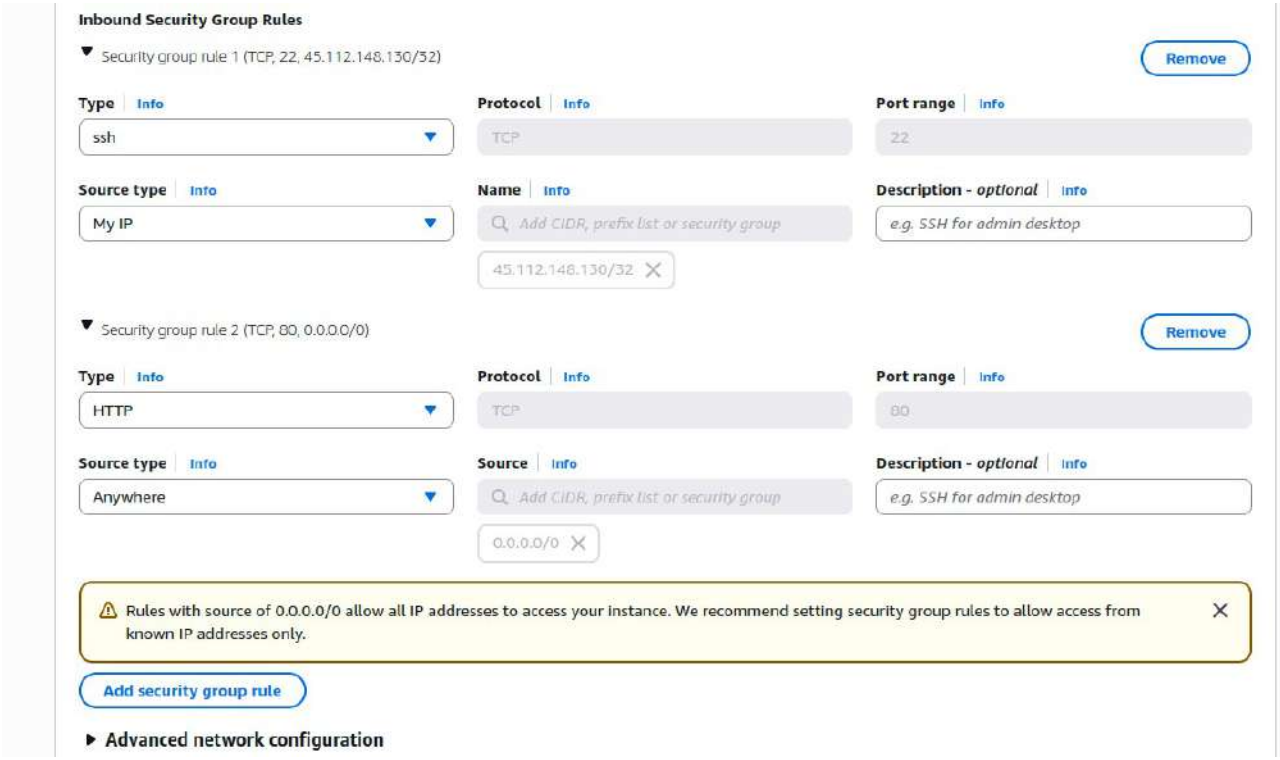
Security group name - *required*

websg1

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

**Description - *required*** Info

launch-wizard-1 created 2025-11-18T07:12:50.161Z




**In Terminal Execution:**

ubuntu@ip-10-0-1-8:~$ git clone https://github.com/sreepathysois/Cafe_Dynamic_Website.git

ubuntu@ip-10-0-1-8:~$ sudo apt-get install apache2 php php-mysql mysql-server mysql-client libapache2-mod-php

ubuntu@ip-10-0-1-8:~/Cafe_Dynamic_Website/mompopcafe$ sudo apt-get update

ubuntu@ip-10-0-1-8:~/Cafe_Dynamic_Website/mompopcafe$ sudo cp -rf * /var/www/html/

ubuntu@ip-10-0-1-8:~/Cafe_Dynamic_Website/mompopcafe$ sudo rm -rf /var/www/html/index.html

ubuntu@ip-10-0-1-8:~/Cafe_Dynamic_Website/mompopcafe$ sudo systemctl restart apache2

ubuntu@ip-10-0-1-8:~/Cafe_Dynamic_Website/mompopcafe$ sudo mysql -u root -p

mysql> create database cafedb;

mysql> create user 'msis'@'localhost' identified by 'msis@123';

mysql> grant all privileges on cafedb .* to 'msis'@'localhost';

mysql> exit

ubuntu@ip-10-0-1-8:~/Cafe_Dynamic_Website$ cd mompopdb/

ubuntu@ip-10-0-1-8:~/Cafe_Dynamic_Website/mompopdb$ mysql -u msis -p

mysql> use cafedb;

mysql> source create-db.sql

mysql> exit

ubuntu@ip-10-0-1-8:~/Cafe_Dynamic_Website/mompopdb$ cd

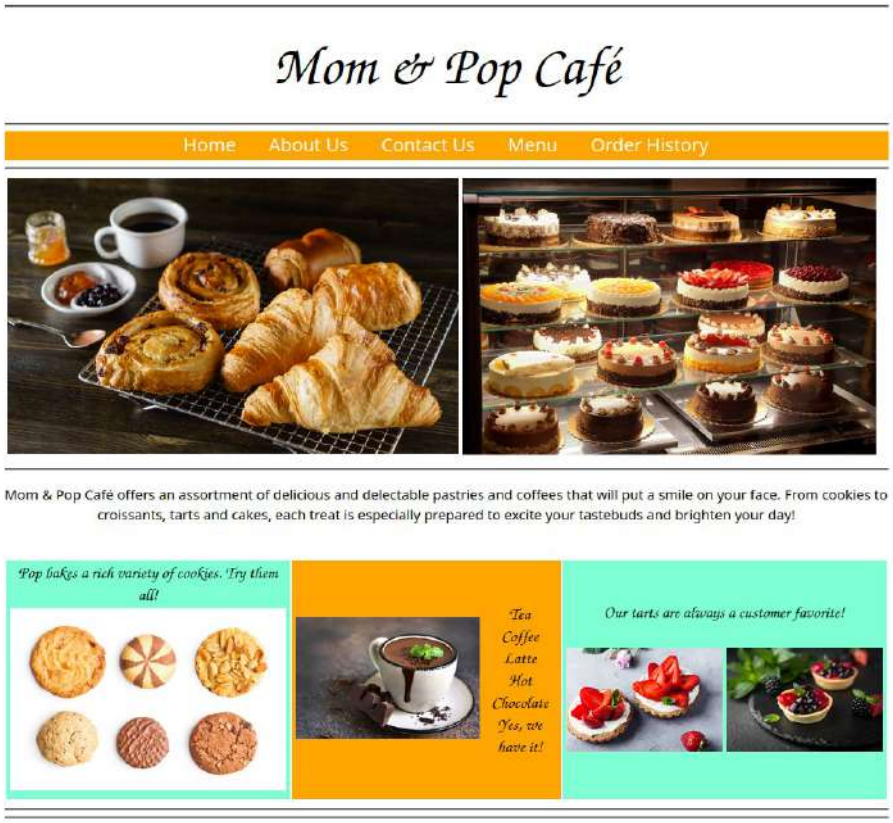
ubuntu@ip-10-0-1-8:~$ sudo nano /var/www/html/getAppParameters.php

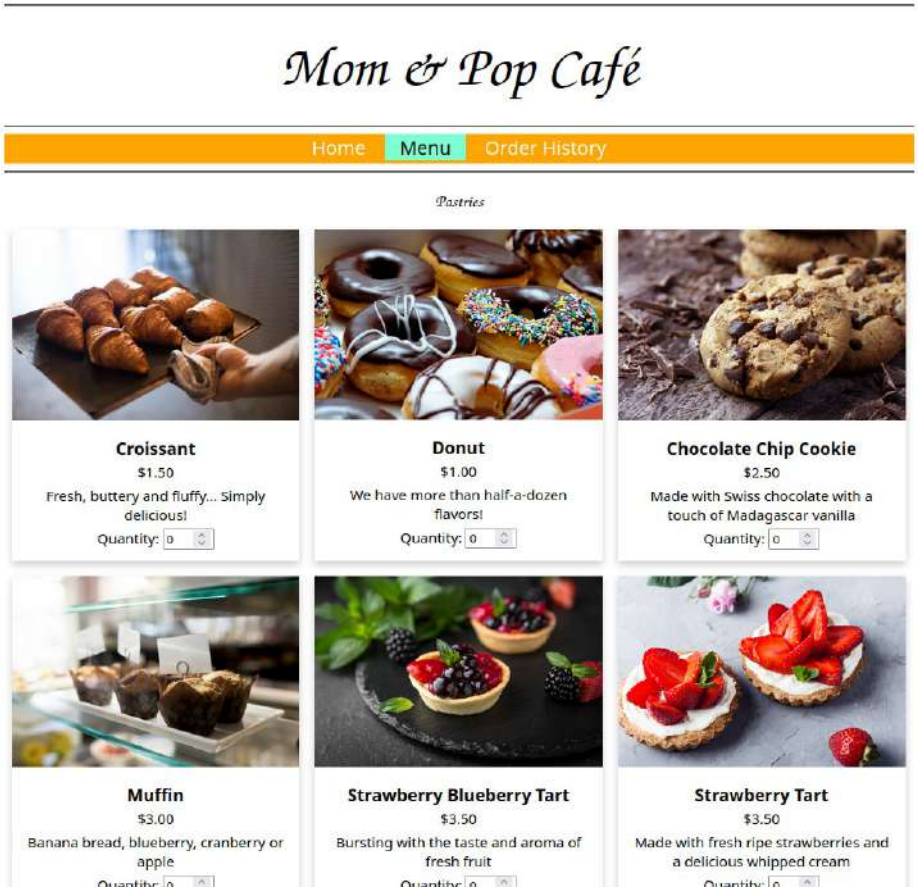
```php
<?php
  // Get the application environment parameters from the Parameter Store.
  //include ('getAppParameters.php');
  $showServerInfo = "false";
  $timeZone = "America/New_York";
  $currency = "$";
  $db_url = "localhost";
  $db_name = "cafedb";
  $db_user = "msis";
  $db_password = "msis@123";

  // Display the server metadata information if the showServerInfo parameter is>
  //include('serverInfo.php');
?>
```

ubuntu@ip-10-0-1-8:~$ sudo systemctl restart apache2

ubuntu@ip-10-0-1-8:~$ sudo systemctl restart mysq

**In Web Browser using EC2 Public ip**



# AWS Identity and Access Management (IAM)

## 1. Introduction

AWS Identity and Access Management (IAM) is a critical security service in AWS that allows organizations to control access to AWS resources securely. With IAM, administrators can create users, groups, policies, and roles, assigning permissions based on the principle of least privilege, ensuring users can only access what is necessary for their job.

In this lab, we explored pre-created IAM users and groups, examined the attached managed and inline policies, assigned users to groups according to job roles, and tested their access through the IAM sign-in URL. This provides hands-on understanding of how IAM manages secure access in real AWS environments.

## 2. Lab Objectives

- To understand IAM user and group management.
- To explore managed and inline IAM policies.
- To assign users to appropriate groups based on business roles.
- To verify access restrictions through real-time testing using login URLs.
- To observe how permission boundaries affect AWS service access.

## 3. AWS IAM Key Concepts (Explanation)

**a) IAM Users:** IAM Users represent individual identities that can authenticate into AWS using passwords or access keys.

**b) IAM Groups: I**AM Groups are collections of users, allowing permissions to be assigned collectively instead of individually.

**c) IAM Policies:** Policies contain permission rules in JSON format which define:

- **Effect** – Allow or Deny
- **Action** – What operations are permitted (example: s3:ListBucket)
- **Resource** – Which AWS resource(s) the policy applies to (* or ARN)

Policies are of two types:

- **Managed Policy** – Prebuilt by AWS or admins, reusable across users and groups.
- **Inline Policy** – Attached directly to a single user/group for unique permission use cases.

**d) IAM Roles:** A temporary access identity used by services, applications, or federated users (not used in this lab but important conceptually).

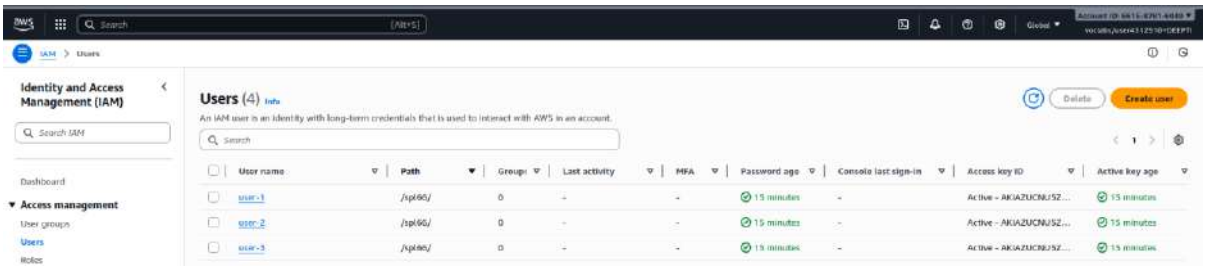
## 4. Business Scenario Summary

| User | Assigned Group | Permissions Access |
|---|---|---|
| user-1 | S3-Support | Read-Only access to Amazon S3 |
| user-2 | EC2-Support | Read-Only access to Amazon EC2 |
| user-3 | EC2-Admin | View, Start, Stop Amazon EC2 instances |

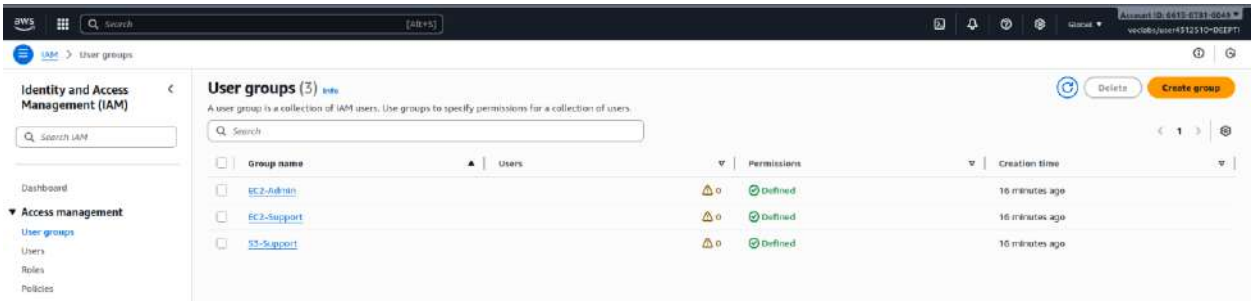
**5. Task-Wise Lab Explanation**

**Task 1: Explore Users and Groups**

- Open IAM from AWS console.
- Observed three pre-created users: user-1, user-2, user-3.




- Verified that user-1 initially had no permissions and no group membership.
- Observed pre-created groups: EC2-Admin, EC2-Support, S3-Support.



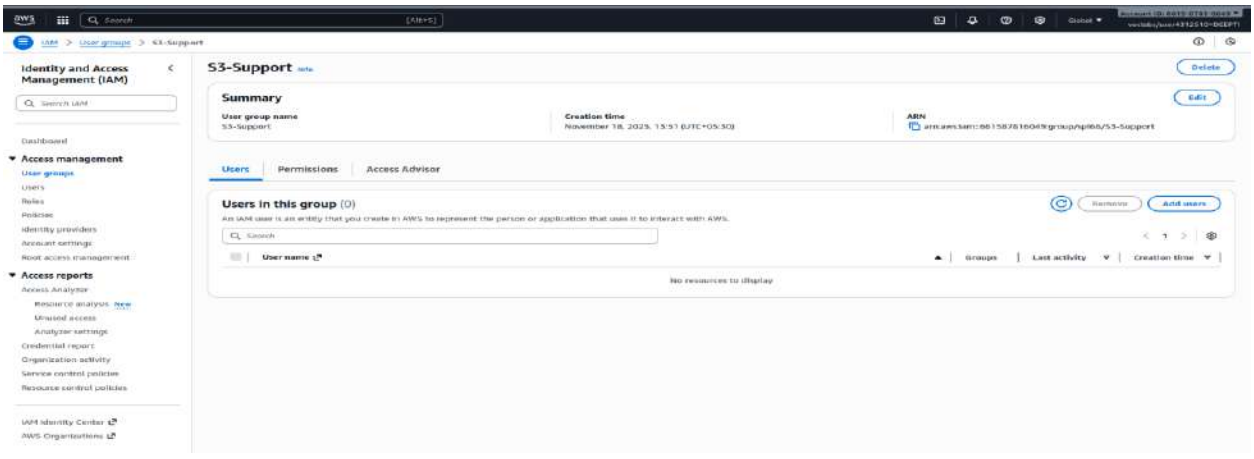

Checked permissions for each group:

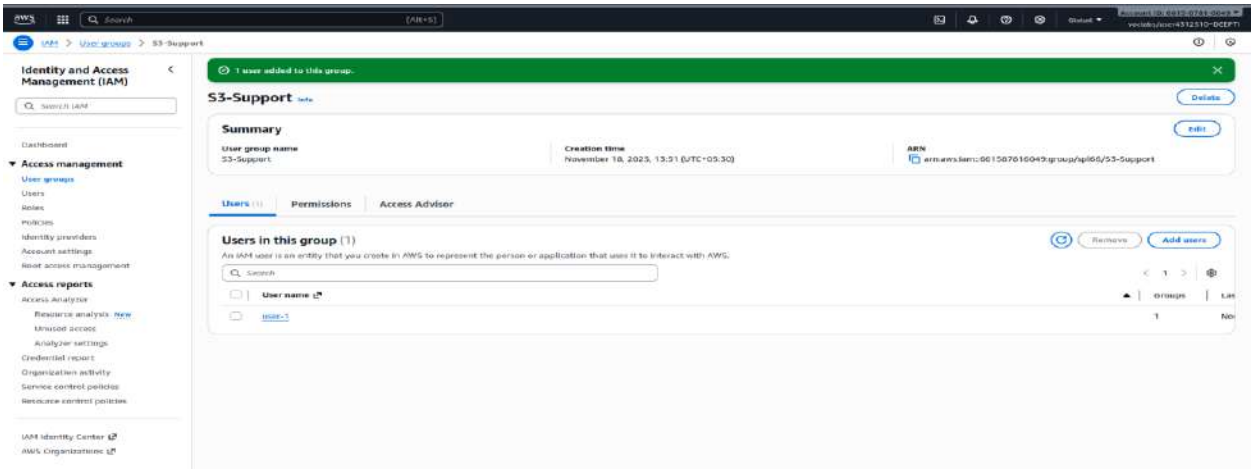
- EC2-Support → AmazonEC2ReadOnlyAccess (Managed Policy)
- S3-Support → AmazonS3ReadOnlyAccess (Managed Policy)
- EC2-Admin → Inline Policy allowing Describe, Start, Stop EC2 instances

**Task 2: Add Users to Groups**

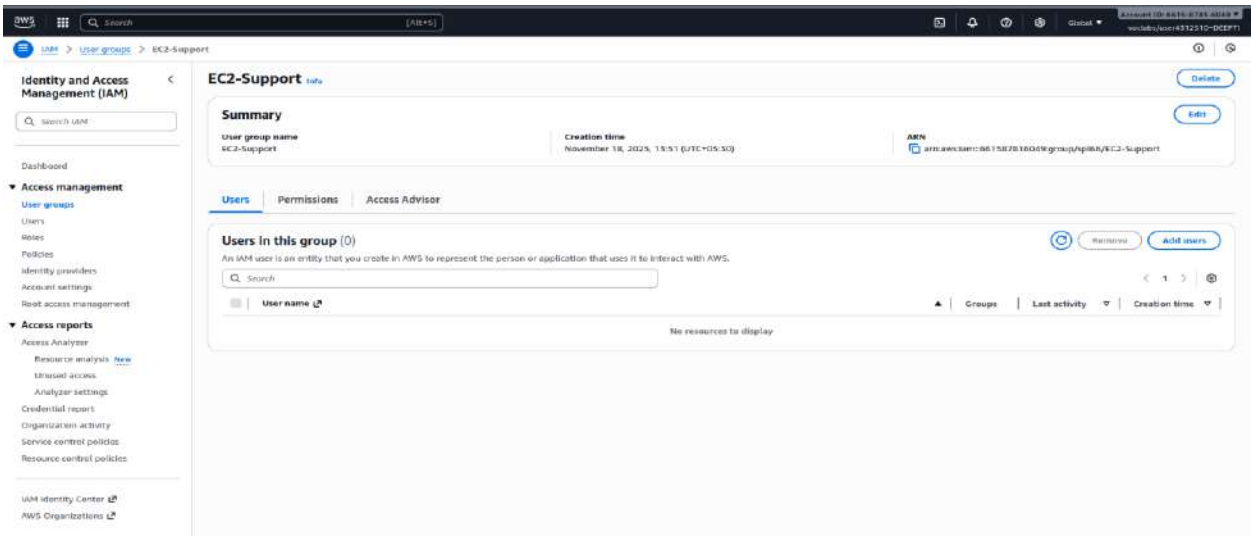
Based on business requirements:

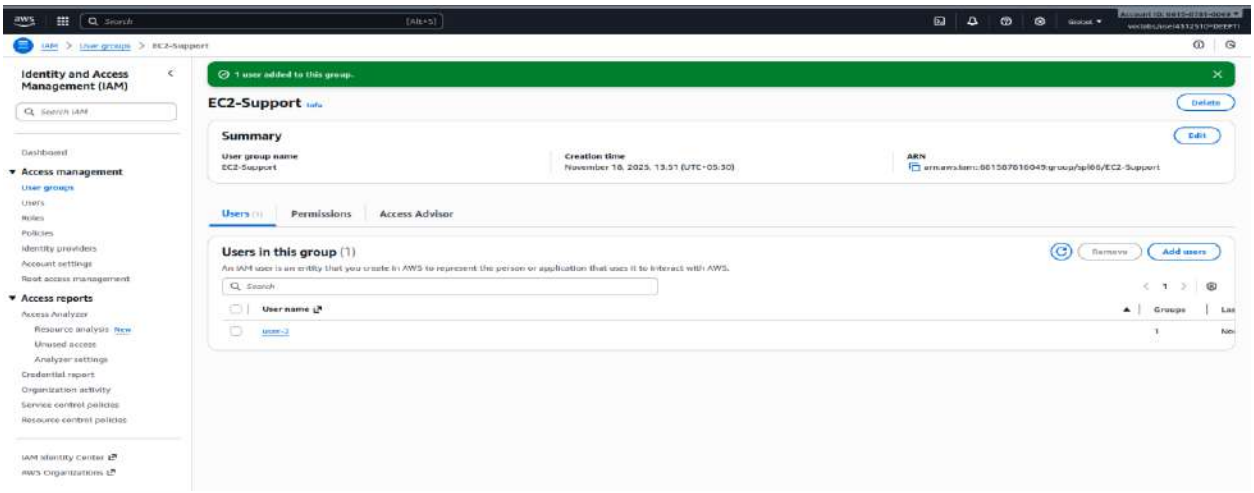
- Added user-1 to S3-Support



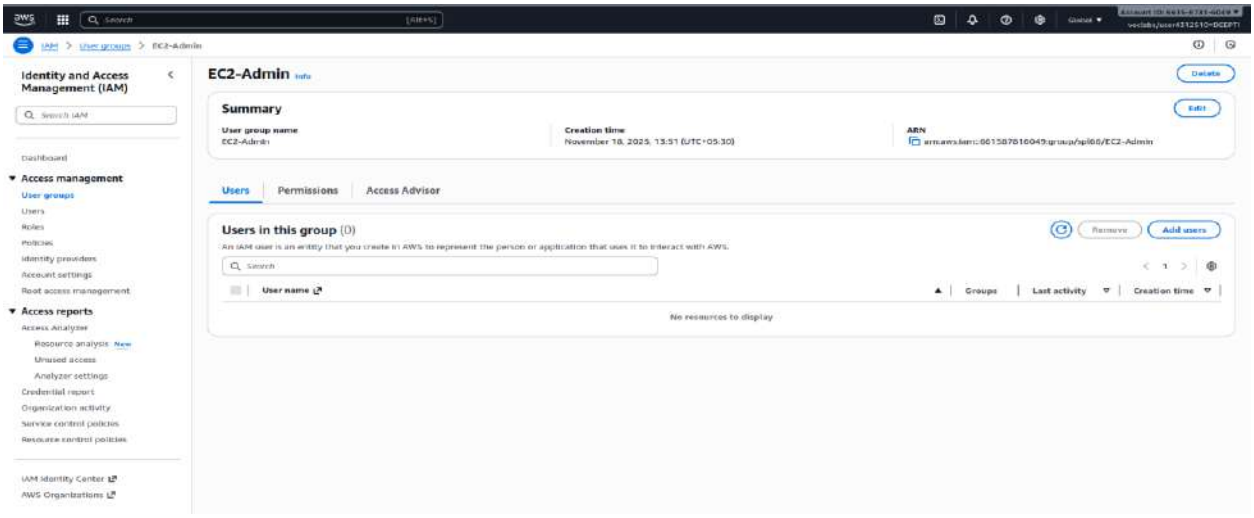



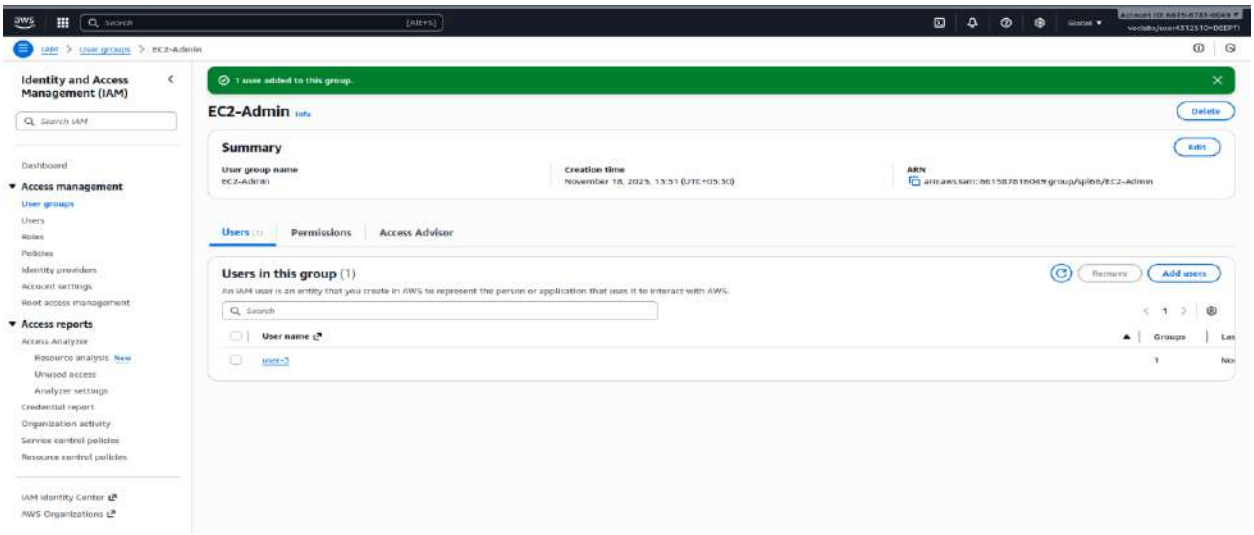

- Added user-2 to EC2-Support






- Added user-3 to EC2-Admin

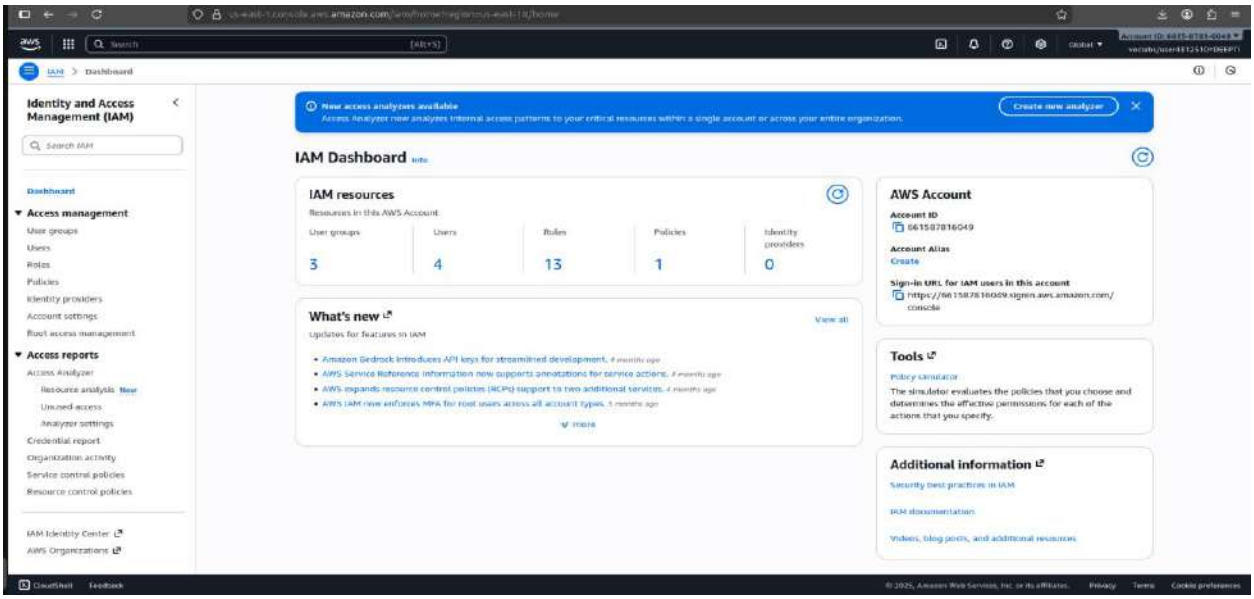


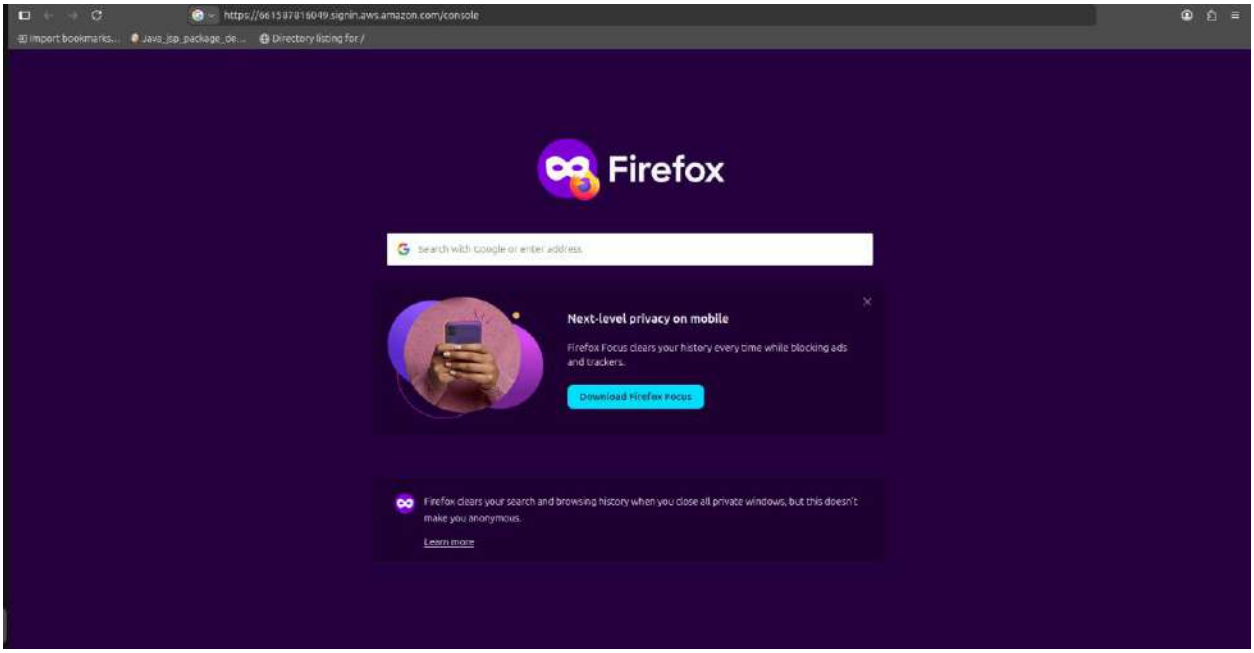

Finally verified that each group displayed 1 assigned user.

**Task 3: Sign-In and Permissions Testing**

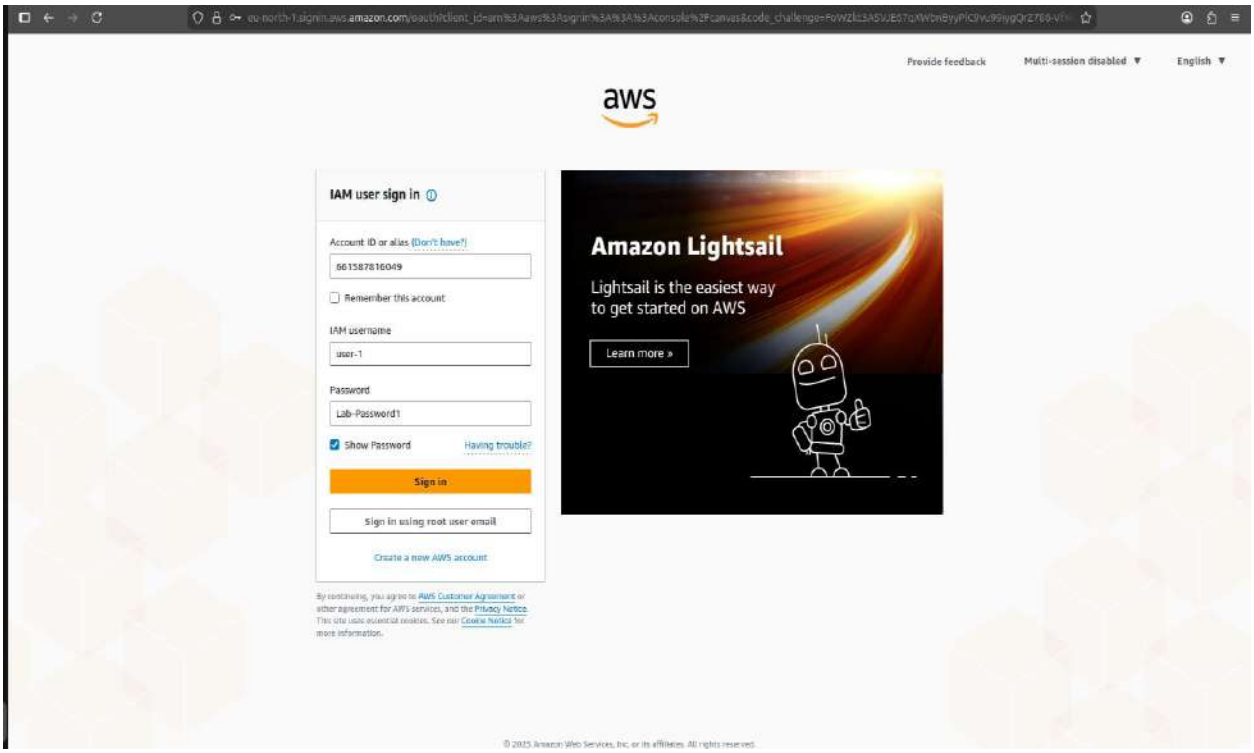
- In the navigation pane on the left, choose the Dashboard. On the right side, a Sign-in URL for IAM users in this account is displayed. The sign-in link shown in the lab environment was: https://661587816049.signin.aws.amazon.com/console



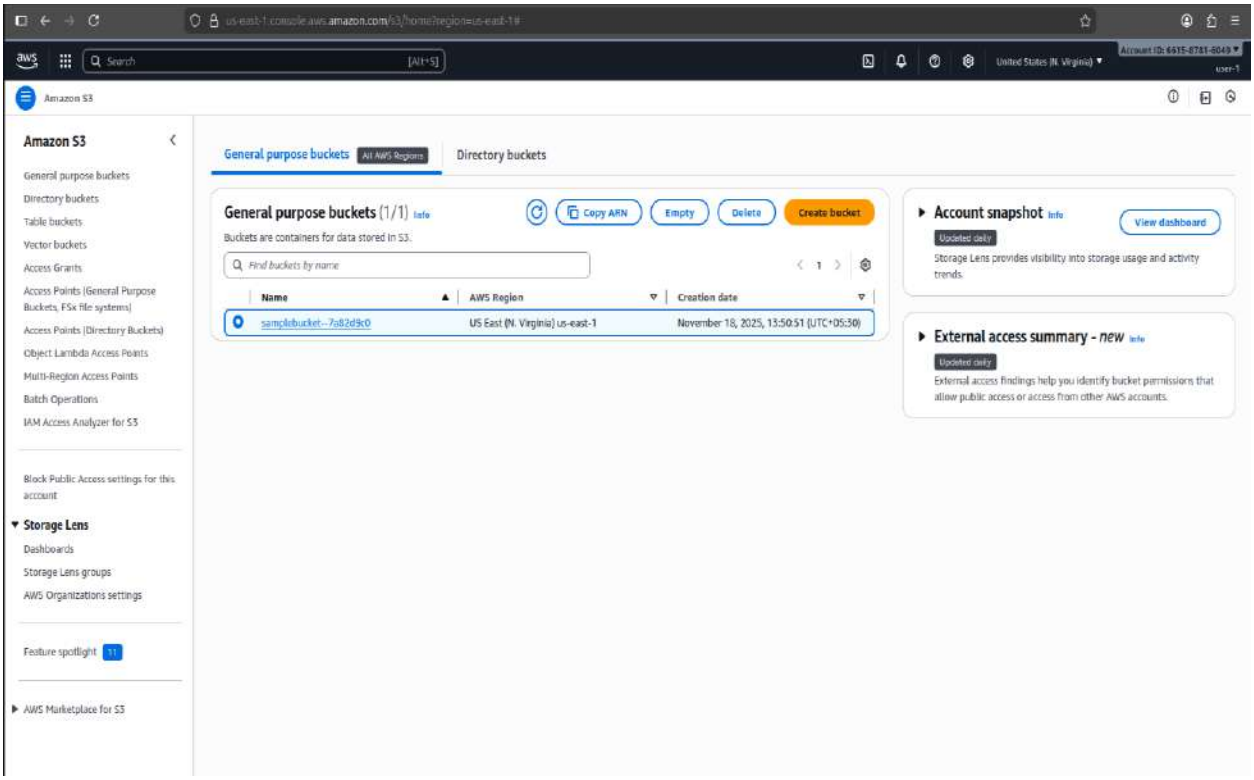

- This link can be used to sign-in to the AWS Account you are currently using. Copy the Sign-in URL for IAM users in this account to a text editor. Open a private (Incognito) window.



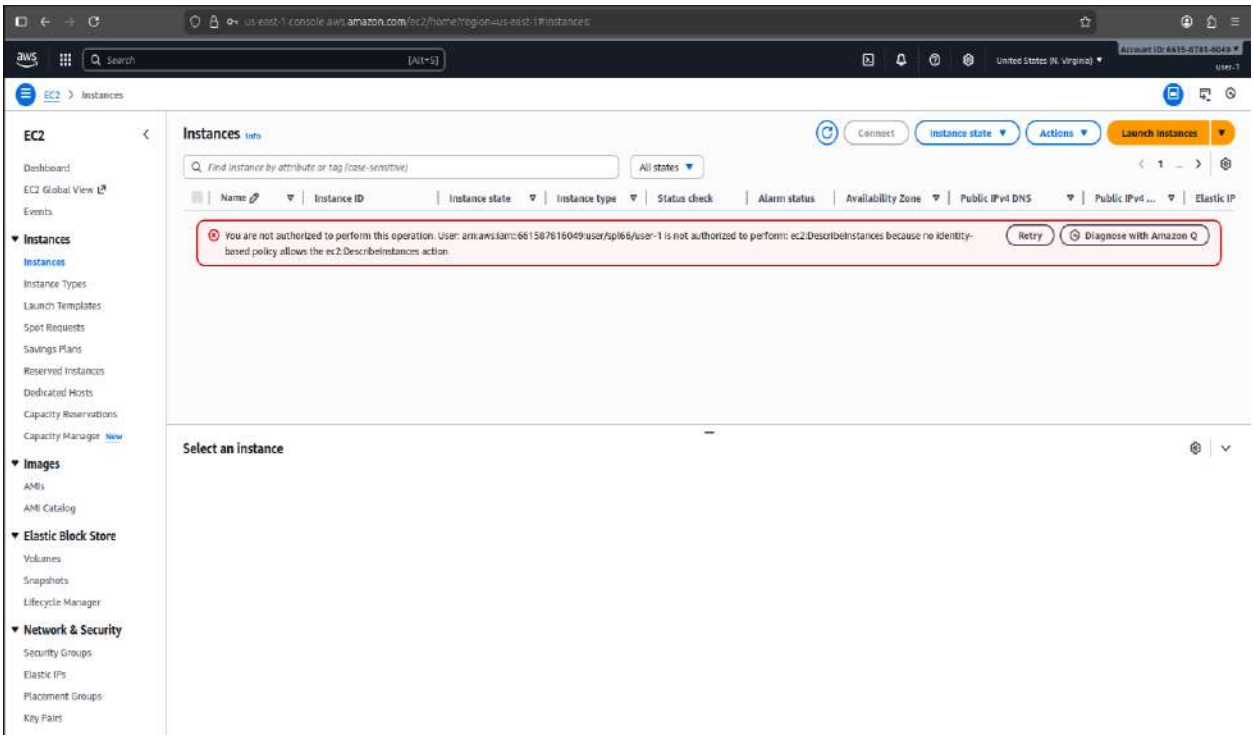

Logged in as user-1



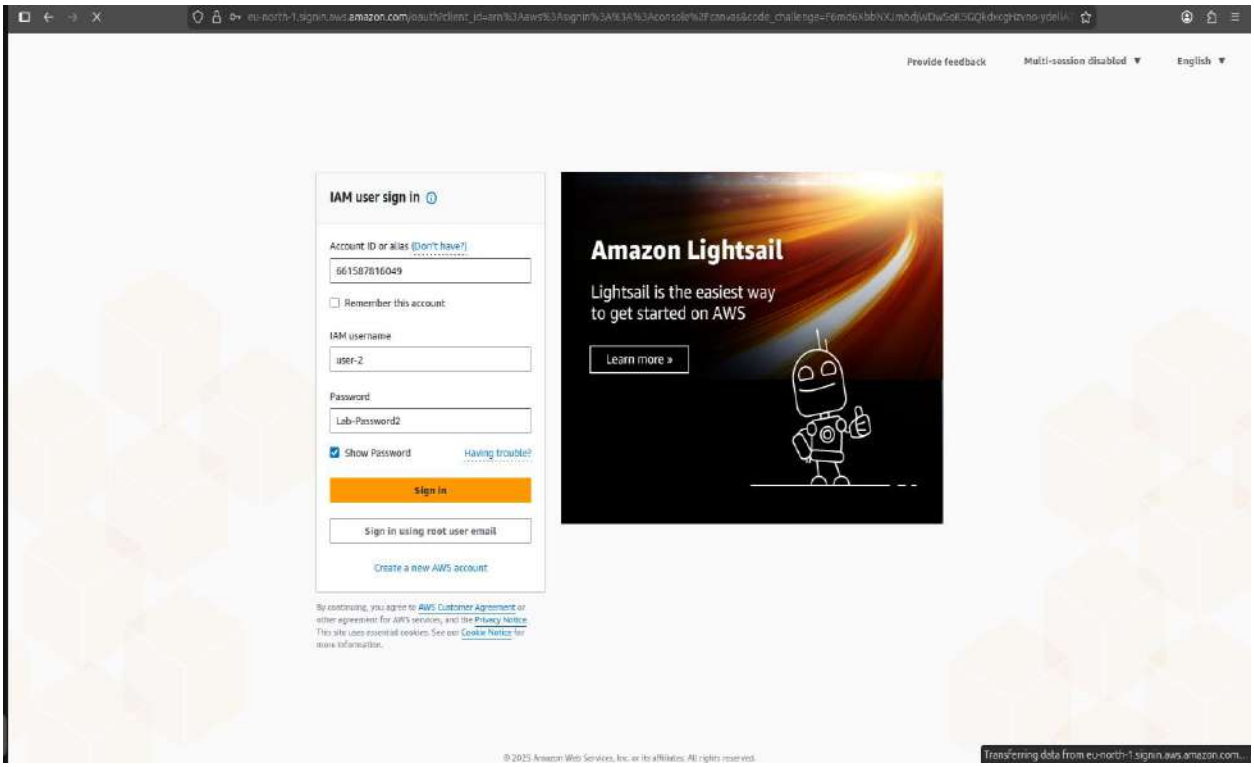

Successfully accessed S3 and viewed bucket list (read-only



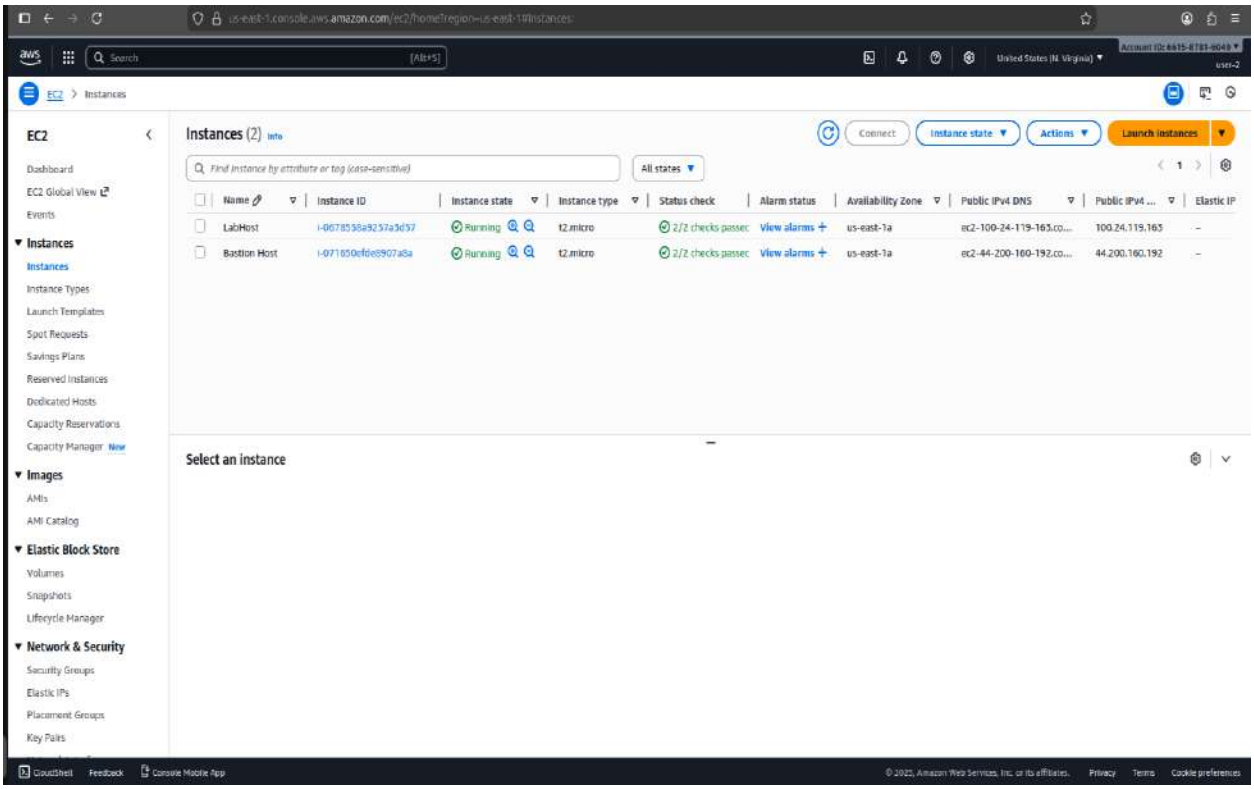

Tried accessing EC2 but received "not authorized" error



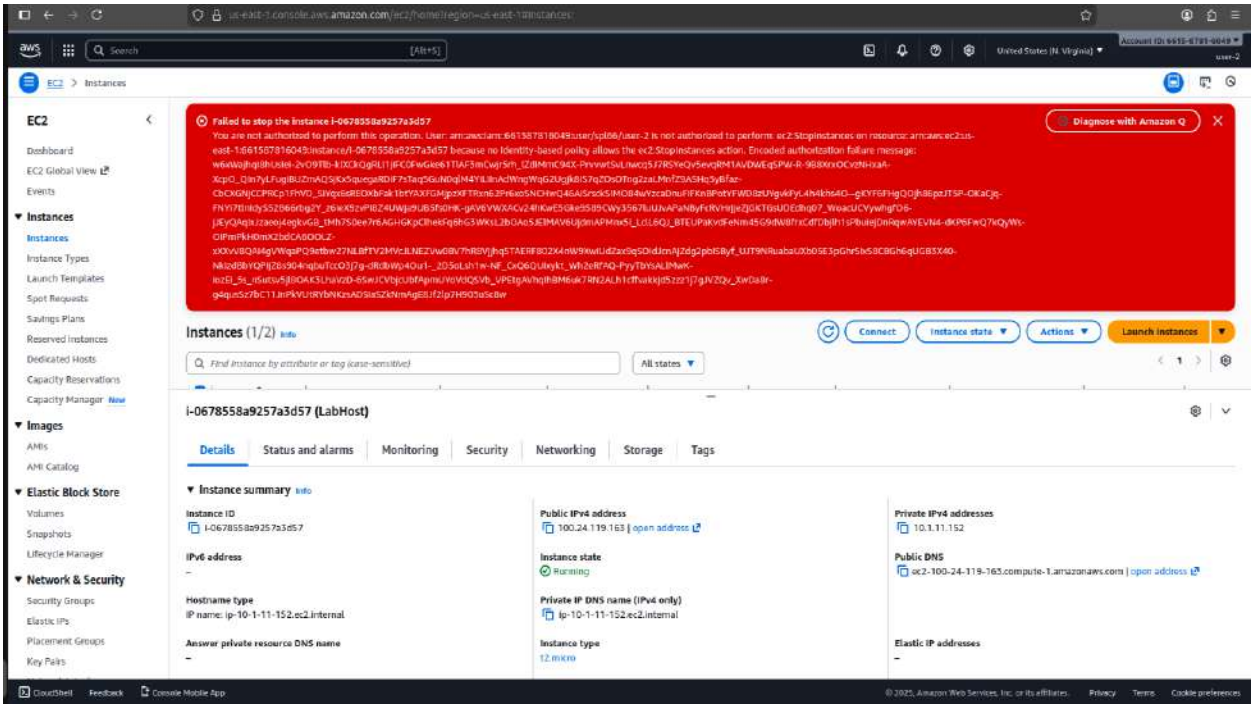

Logged in as user-2



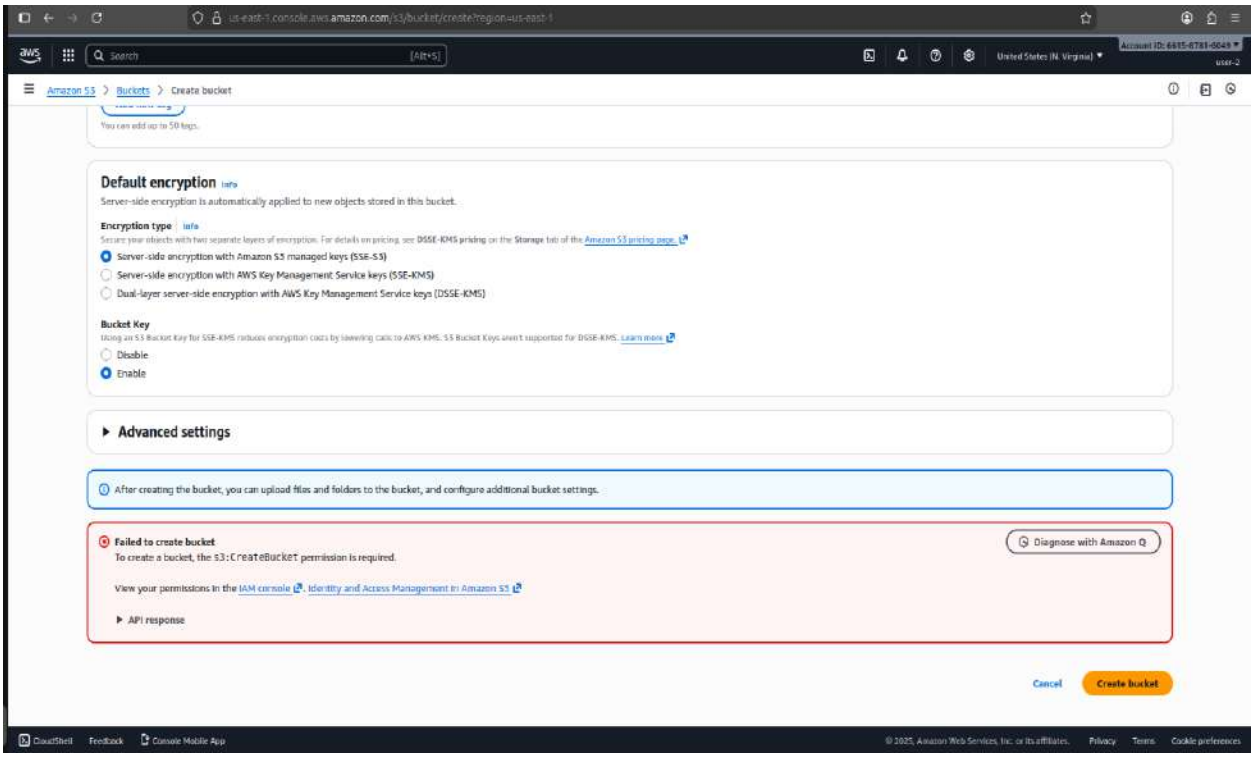

Successfully viewed EC2 instance (read-only)



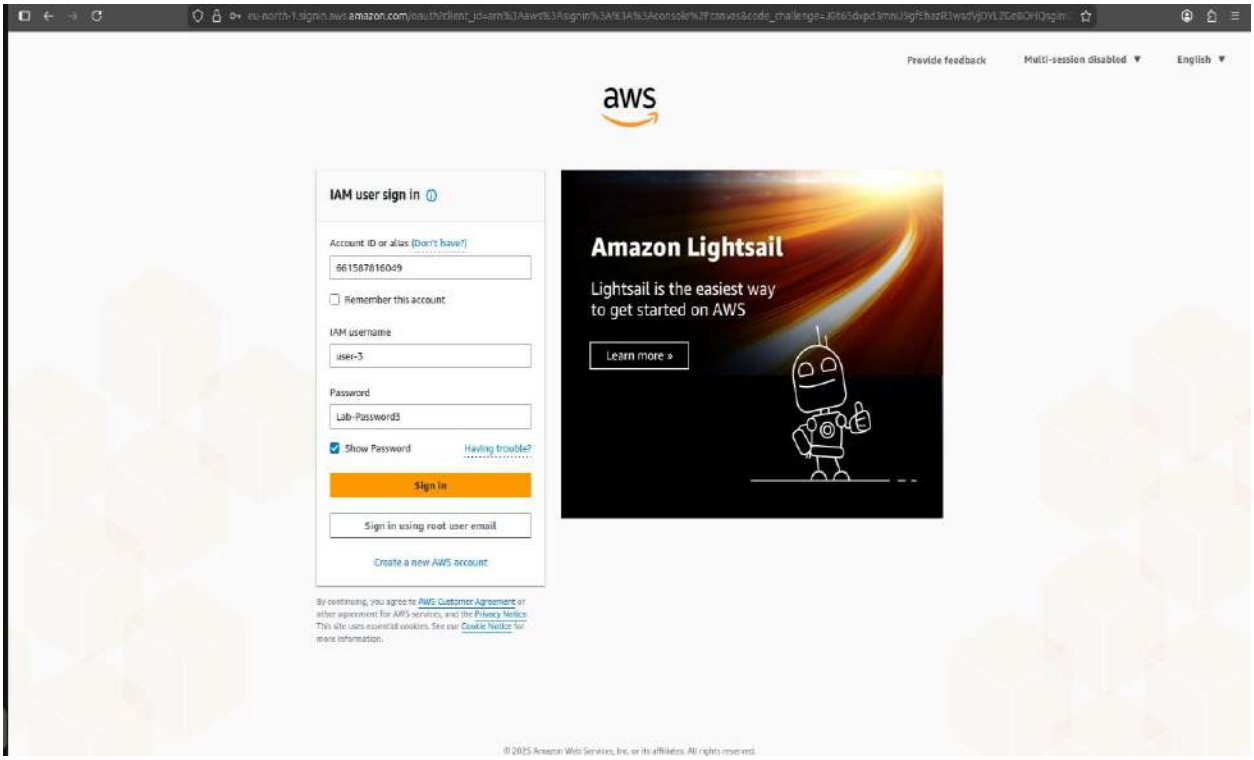

Attempted to stop EC2 instance but got permission denied



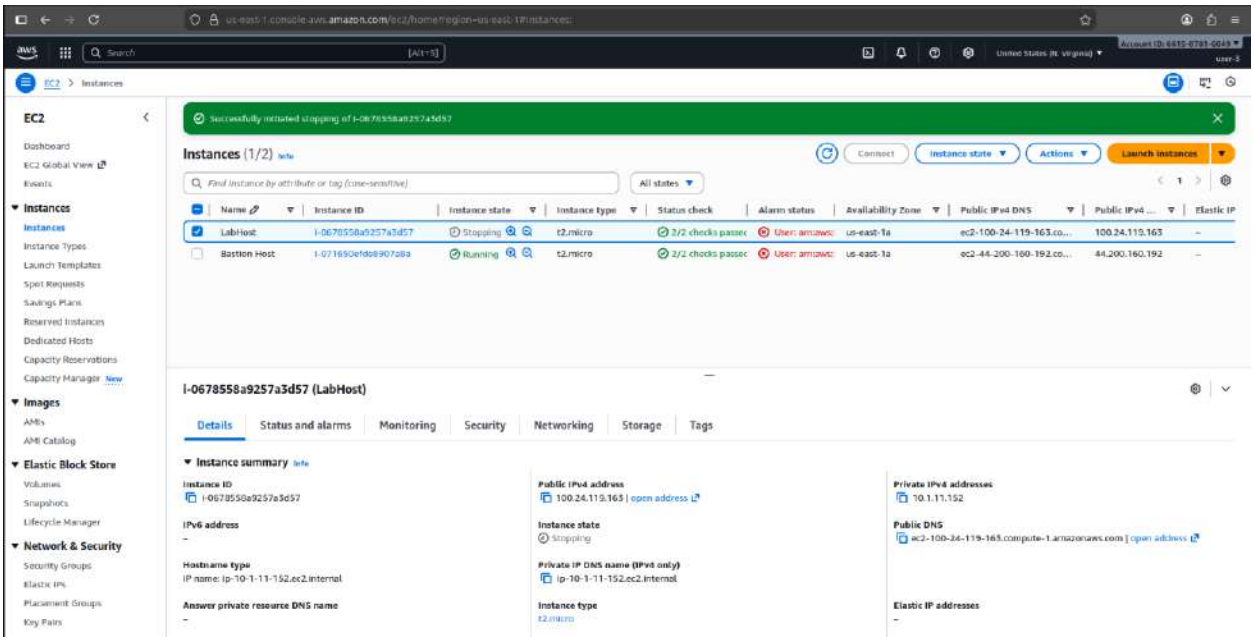

Tried accessing S3 but was denied




Logged in as user-3




Accessed EC2 and successfully stopped the LabHost instance




Permissions worked as per admin role design

## 6. Conclusion

Through this lab, we successfully learned how IAM enables secure access control in AWS environments. We explored and analyzed IAM users, groups, and policies, assigned users to groups based on job roles, and verified permission restrictions by testing actions in AWS Management Console.

This demonstrates how IAM enforces secure, role-based access, ensuring every individual has only the required level of access, maintaining security and operational integrity.

# AWS Lambda Stopinator Lab Report

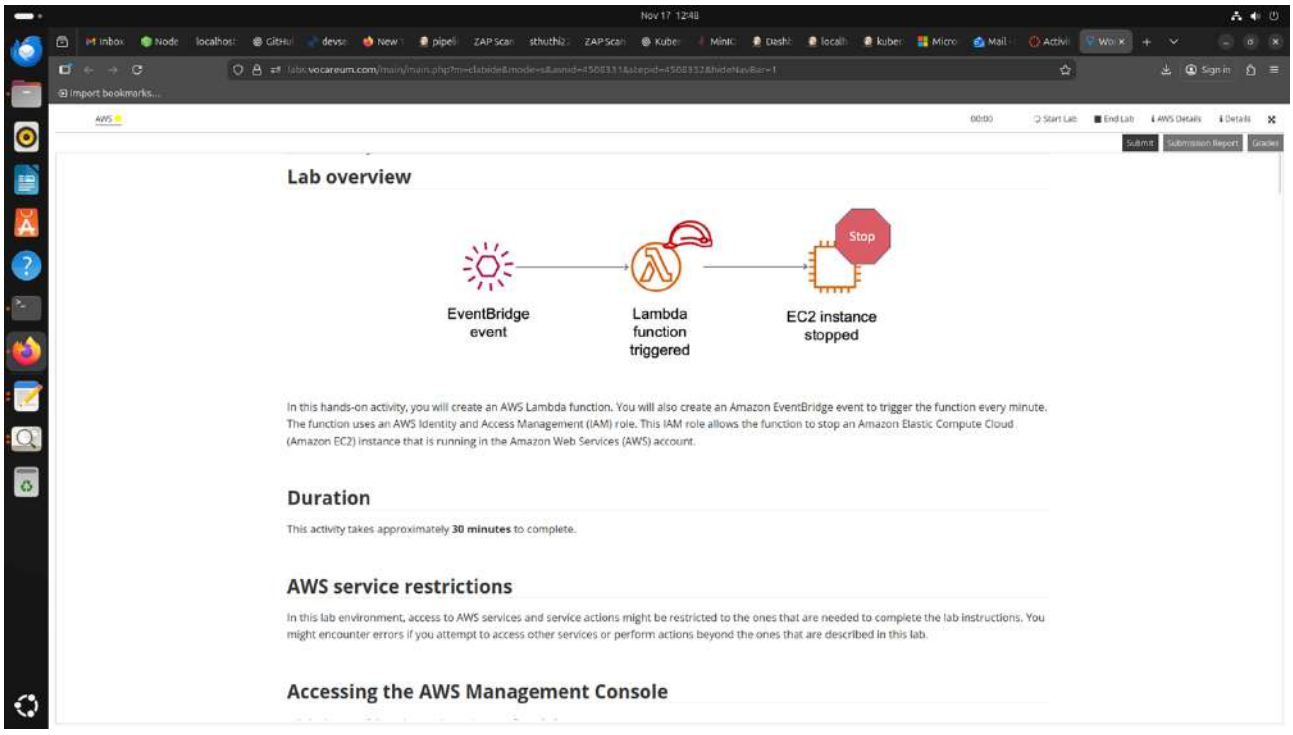
**Lab:** AWS Lambda – Scheduled EC2 Stopinator

## Objective

The objective of this lab was to create an AWS Lambda function that automatically stops a specified EC2 instance at a scheduled interval using Amazon EventBridge (CloudWatch Events) as the trigger. This demonstrates serverless automation in AWS.

## Lab Environment Setup

- AWS Management Console was used for this lab.
- A timer-based lab session was started using the “Start Lab” button.
- Pop-up windows were allowed in the browser to open the AWS Management Console in a new tab.
- Resources were named exactly as specified in the instructions to ensure the lab scoring script works properly.



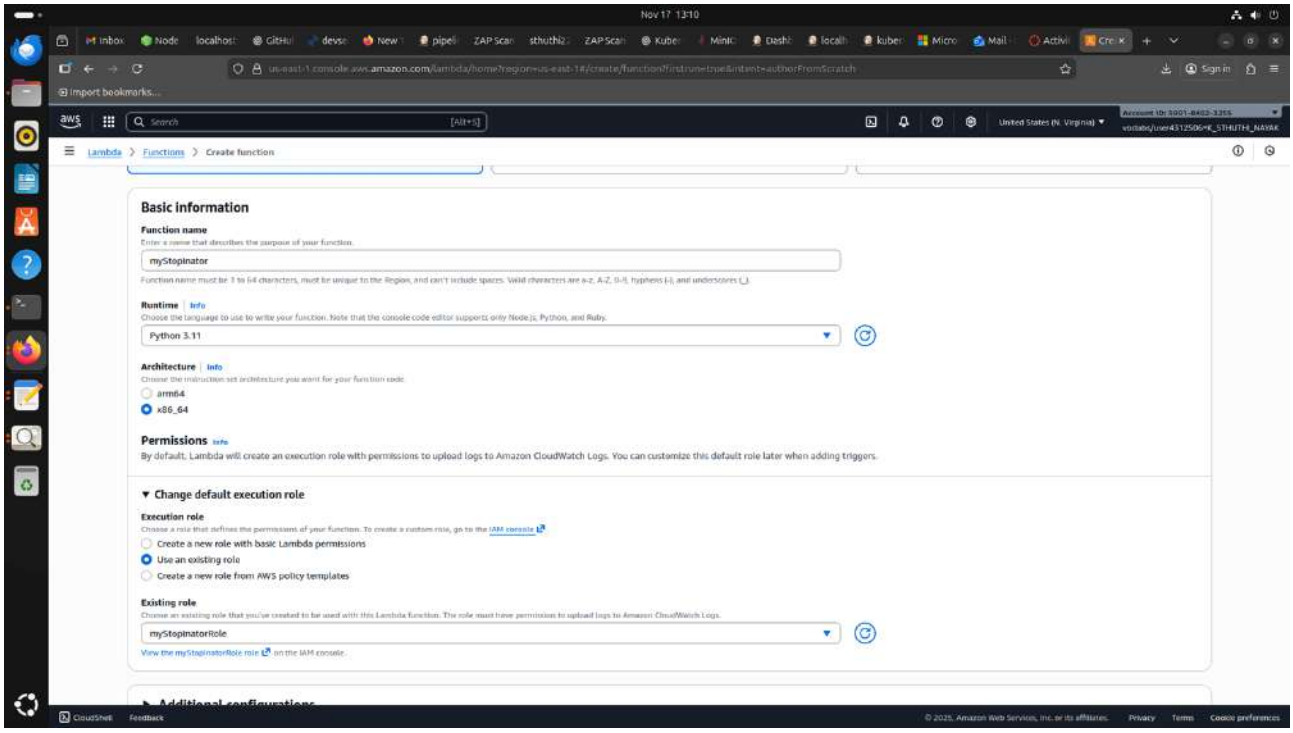

## Task 1: Create a Lambda Function

**Steps:**

1. Navigated to **AWS Lambda** in the AWS Console.
2. Chose **Create a function → Author from scratch**.

3. Configured the function:
    - **Function name:** myStopinator
    - **Runtime:** Python 3.11
4. Configured the execution role:
    - **Execution role:** Use an existing role
    - **Existing role:** myStopinatorRole
5. Clicked **Create function**.

**Outcome:** Lambda function myStopinator successfully created.

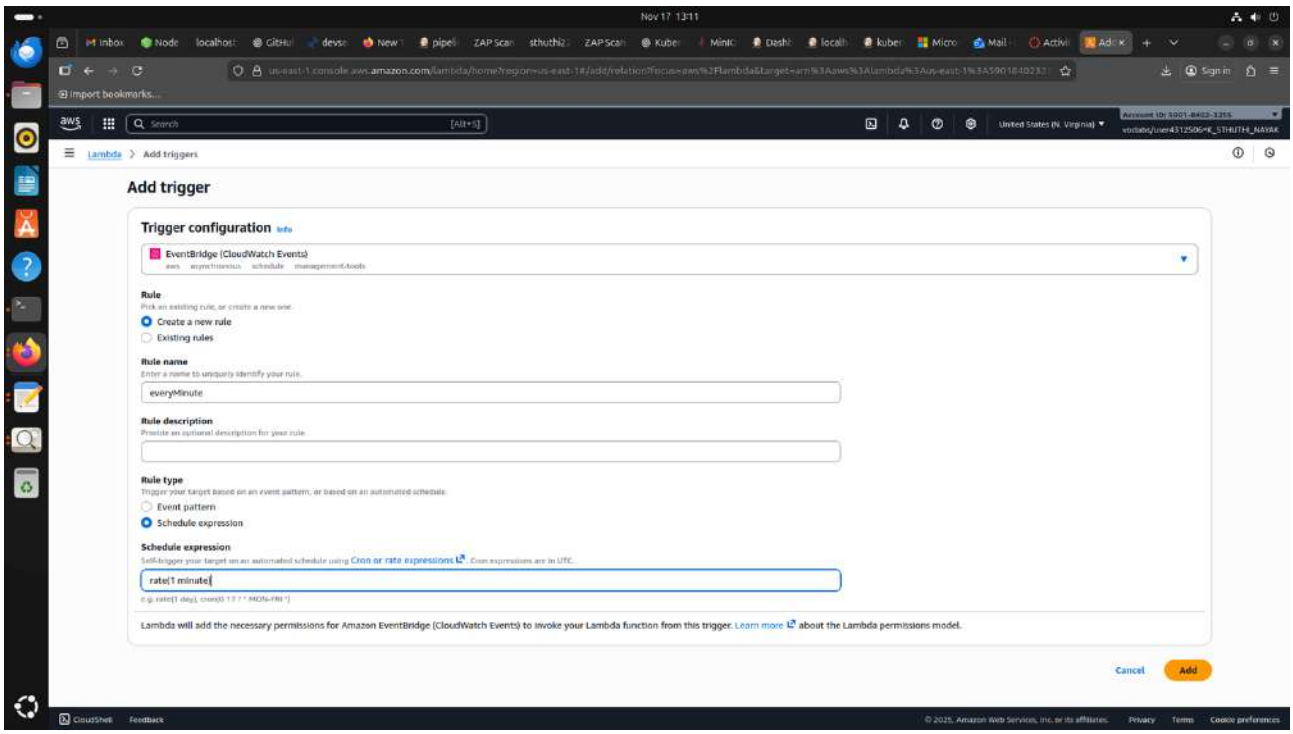



# Task 2: Configure the Trigger

**Steps:**

1. In the Lambda console, clicked **Add trigger**.
2. Selected **EventBridge (CloudWatch Events)**.
3. Created a new rule:
    - **Rule name:** everyMinute
    - **Rule type:** Schedule expression
    - **Schedule expression:** rate(1 minute)
4. Clicked **Add** to attach the trigger to the Lambda function.

**Outcome:** Lambda function is now scheduled to run every minute.




# Task 3: Configure the Lambda Function Code

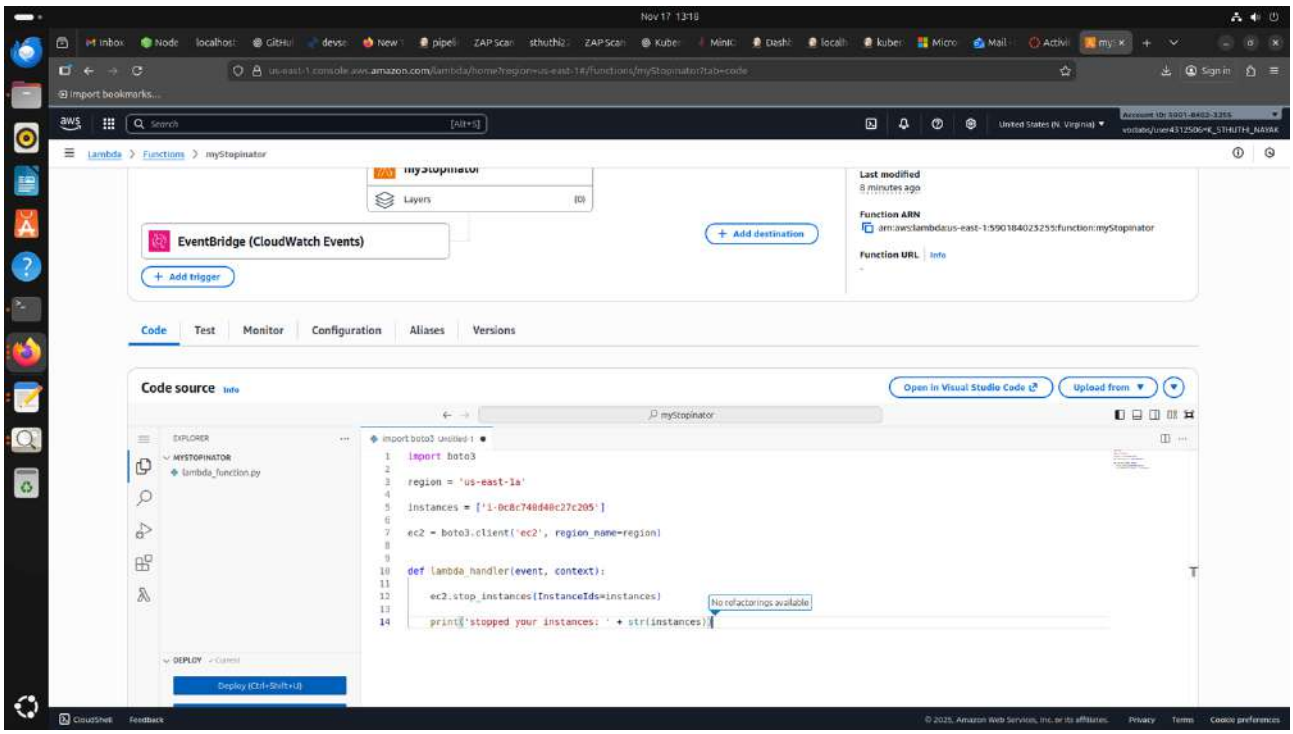
**Steps:**

1. Opened lambda_function.py under the **Code** tab.
2. Replaced the default code with the following Python script:

```
import boto3

region = '<REPLACE_WITH_REGION>'
instances = ['<REPLACE_WITH_INSTANCE_ID>']

ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    ec2.stop_instances(InstanceIds=instances)
    print('stopped your instances: ' + str(instances))
```

3. Replaced placeholders:
   - <REPLACE_WITH_REGION> → e.g., 'us-east-1'
   - <REPLACE_WITH_INSTANCE_ID> → Actual EC2 instance ID of instance1
4. Saved the changes and clicked **Deploy**.

**Outcome:** Lambda function code is configured to stop the specified EC2 instance every minute.



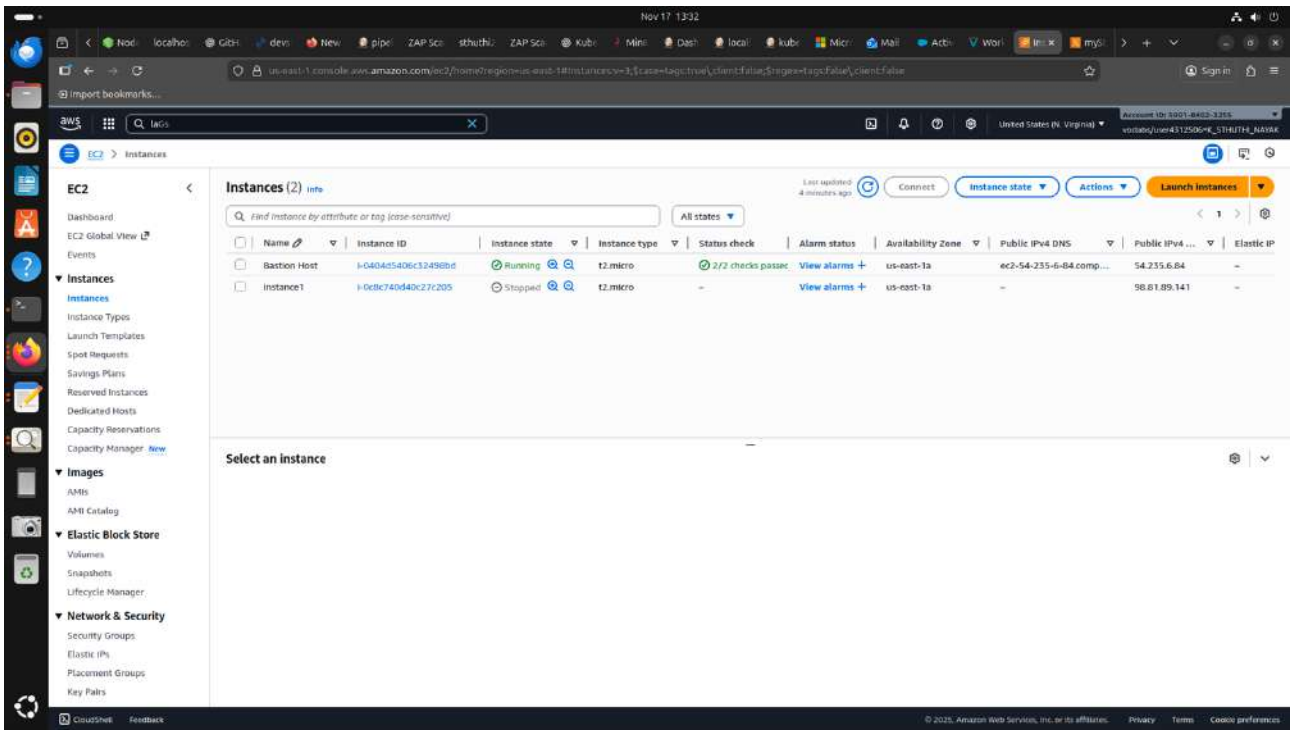

# Task 4: Verify Lambda Function Execution

**Steps:**

1. Navigated to **Monitor** tab in Lambda to check invocation metrics.
2. Observed the function invocation count and success rate.
3. Checked the EC2 console to verify the instance was stopped.
4. Attempted to restart the instance.

**Observation:**

- The instance stops automatically again within 1 minute due to the scheduled Lambda trigger.




# Conclusion

- The lab successfully demonstrated automation of EC2 instance management using AWS Lambda and EventBridge.
- Serverless functions eliminate the need for a dedicated server to run scheduled tasks.
- Proper naming conventions and role permissions are essential for automation scripts to function correctly.

# IMPLEMENTING SERVERLESS ARCHITECTURE ON AWS

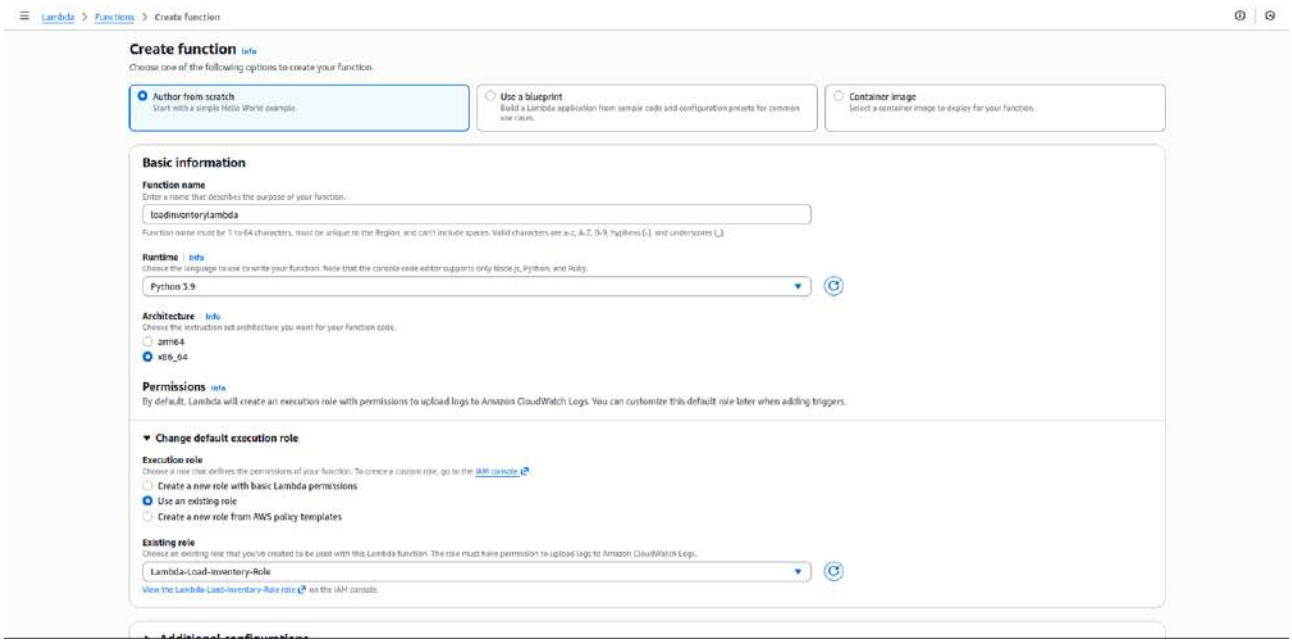
## STEP 1: CREATE A LAMBDA FUNCTION

Function name: loadinventorylamda

Runtime: python 3.9

Change execution role: use an exixting role

Choose: lambda_load_inventory_role




In the **Code source** section, in the **Environment** pane, choose **lambda_function.py**.

In the code editor for the **lambda_function.py** file, delete all the default code.

In the **Code source** editor, copy and paste the following code:

```
 Load-Inventory Lambda function
# This function is invoked by an object being created in an Amazon S3 bucket.
# The file Is downloaded and each line is inserted into a DynamoDB table.
import json, urllib, boto3, csv
# Connect to S3 and DynamoDB
s3 = boto3.resource('s3')
dynamodb = boto3.resource('dynamodb')
# Connect to the DynamoDB tables
inventoryTable = dynamodb.Table('Inventory');
# This handler is run every time the Lambda function is invoked
def lambda_handler(event, context):
```

```
 # Show the incoming event in the debug log
print("Event received by Lambda function: " + json.dumps(event, indent=2))
# Get the bucket and object key from the Event
bucket = event['Records'][0]['s3']['bucket']['name']
 key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'])
localFilename = '/tmp/inventory.txt'
# Download the file from S3 to the local filesystem
try:
s3.meta.client.download_file(bucket, key, localFilename)
except Exception as e:
print(e)
print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.'.format(key, bucket))
raise e
# Read the Inventory CSV file
with open(localFilename) as csvfile:
reader = csv.DictReader(csvfile, delimiter=',')
# Read each row in the file
rowCount = 0
for row in reader:
rowCount += 1
# Show the row in the debug log
print(row['store'], row['item'], row['count'])
try:
# Insert Store, Item and Count into the Inventory table
inventoryTable.put_item(
Item={
'Store':  row['store'],
'Item':   row['item'],
'Count':  int(row['count'])})
except Exception as e:
print(e)
print("Unable to insert data into DynamoDB table".format(e))
# Finished!
return "%d counts inserted" % rowCount
```

# STEP 2: CREATE S3 BUCKET

Name: myinventory

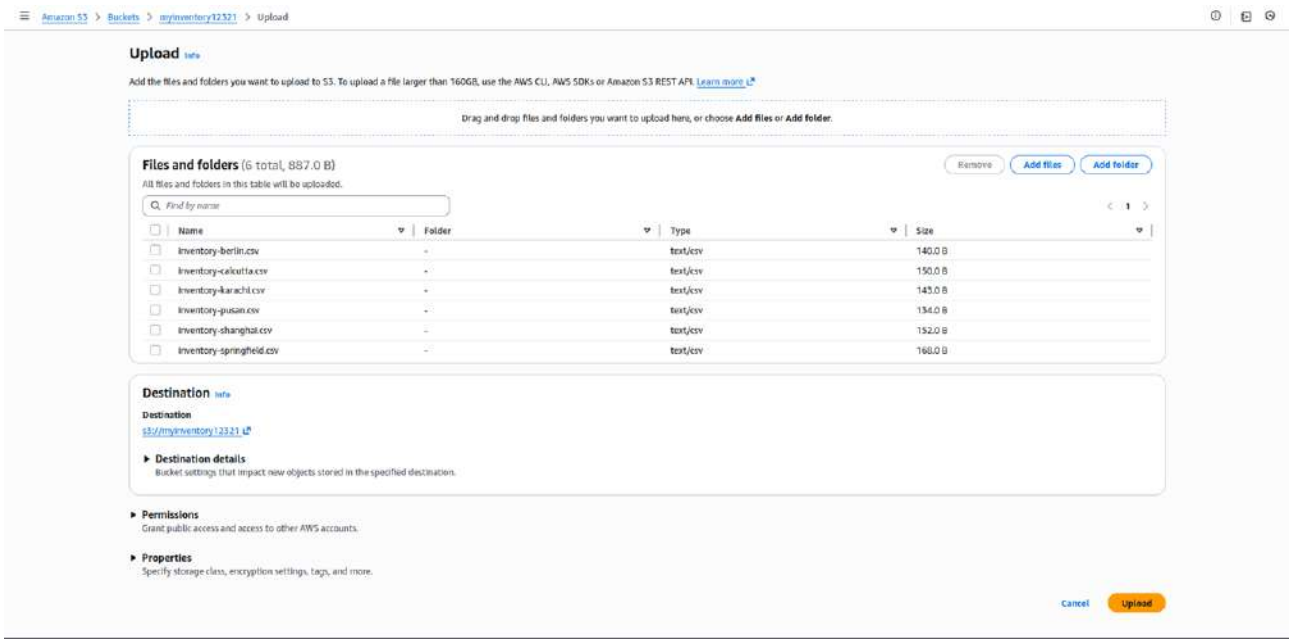
Navigate to Properties→Event notification→Create event notification

Event Name:Inventory load

Event Types:Object creation

Destination:Lambda Function

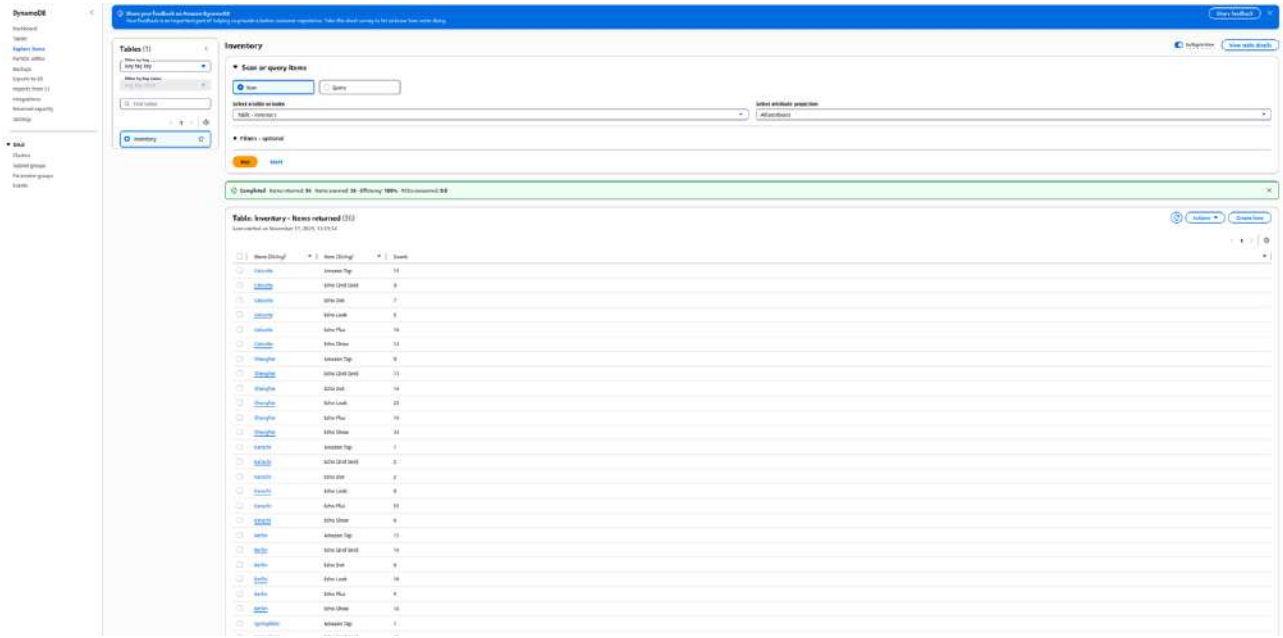
## STEP 3:UPLOAD THE CSV FILES INTO S3




## STEP 4: DYNAMO DB

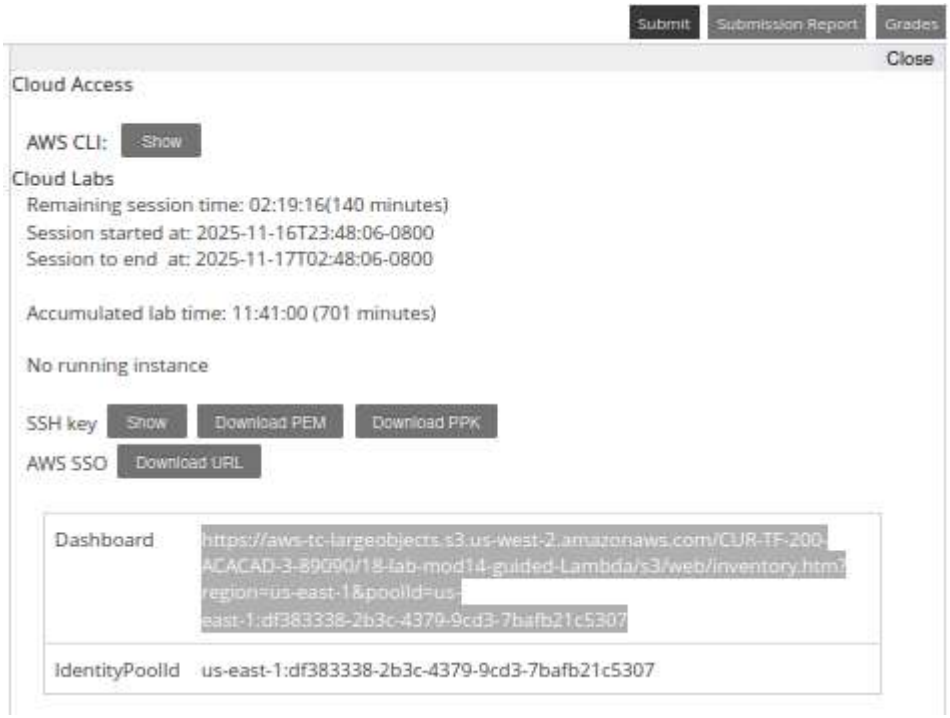
-Explore the items.

-Items will be displayed.



STEP 5: DASHBOARD

Go to AWS I details




Access the dashboard link.

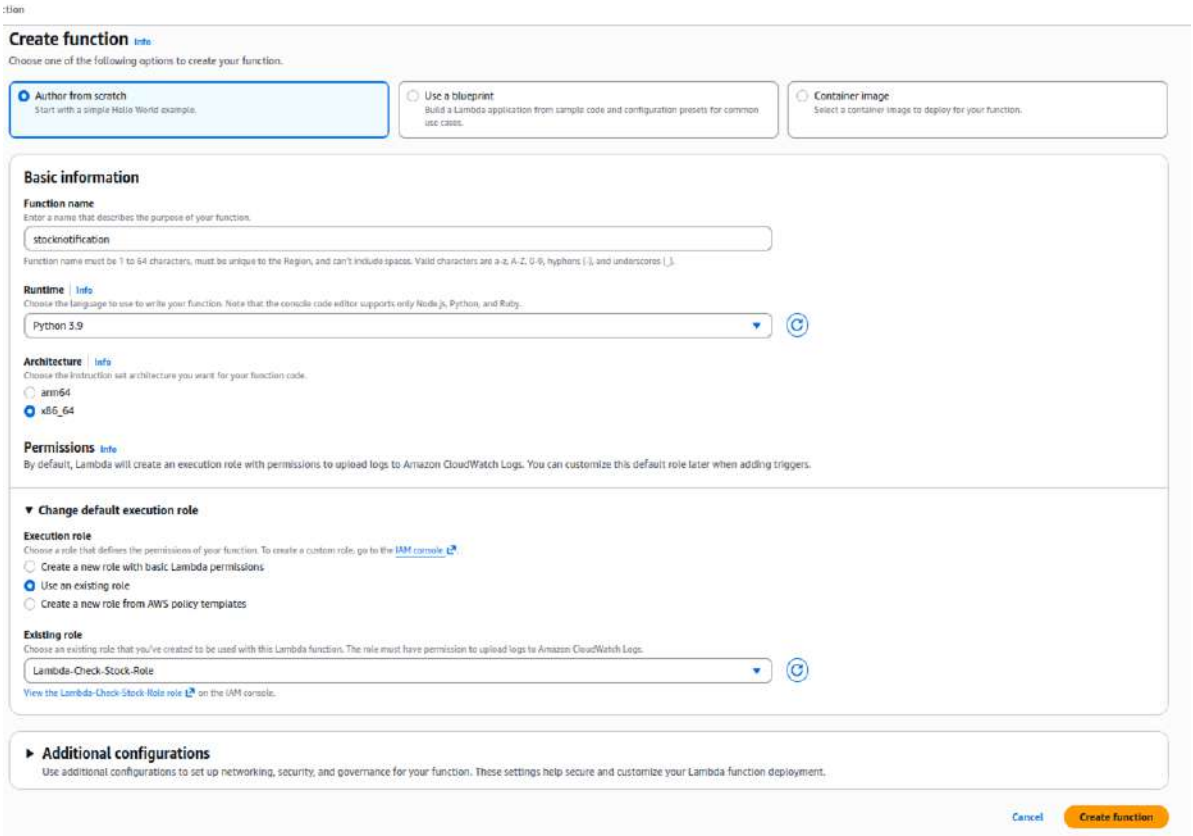
STEP 6: CREATE ANOTHER LAMBDA FUNCTION

Create function: Storenotification

Runtime:python 3.9

use existing role: lambda_check_stock_role

create function




In the **Code source** section, in the **Environment** pane, choose **lambda_function.py**.

In the code editor for the **lambda_function.py** file, delete all the default code.

In the **Code source** editor, copy and paste the following code:

```
# Stock Check Lambda function
#
# This function is invoked when values are inserted into the Inventory DynamoDB table.
# Inventory counts are checked and if an item is out of stock, a notification is sent to an SNS Topic.
import json, boto3
# This handler is run every time the Lambda function is invoked
def lambda_handler(event, context):
  # Show the incoming event in the debug log
  print("Event received by Lambda function: " + json.dumps(event, indent=2))
  # For each inventory item added, check if the count is zero
  for record in event['Records']:
    newImage = record['dynamodb'].get('NewImage', None)
    if newImage:
      count = int(record['dynamodb']['NewImage']['Count']['N'])
      if count == 0:
        store = record['dynamodb']['NewImage']['Store']['S']
        item  = record['dynamodb']['NewImage']['Item']['S']
        # Construct message to be sent
        message = store + ' is out of stock of ' + item
        print(message)
        # Connect to SNS
        sns = boto3.client('sns')
        alertTopic = 'NoStock'
        snsTopicArn = [t['TopicArn'] for t in sns.list_topics()['Topics']
                       if t['TopicArn'].lower().endswith(':' + alertTopic.lower())][0]
        # Send message to SNS
        sns.publish(
          TopicArn=snsTopicArn,
          Message=message,
          Subject='Inventory Alert!',
          MessageStructure='raw'
        )
  # Finished!
  return 'Successfully processed {} records.'.format(len(event['Records']))
```

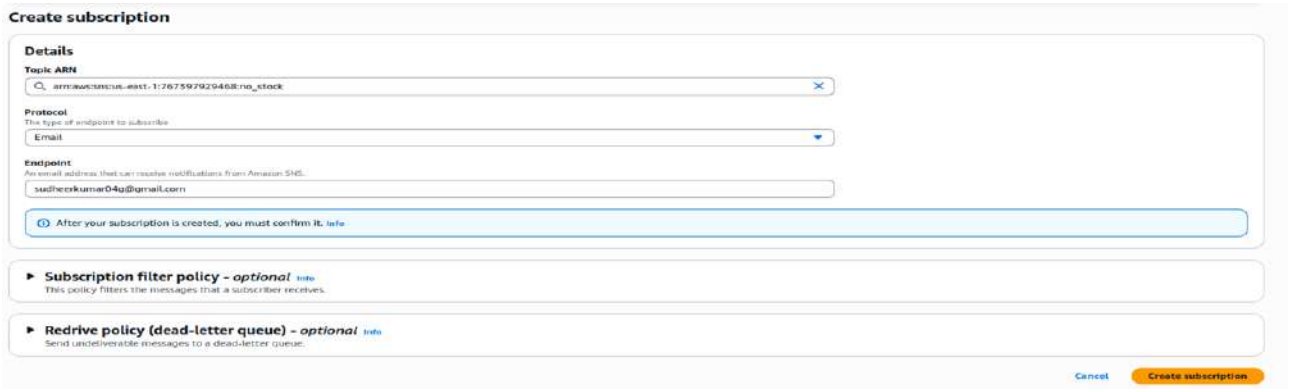
## STEP 7: SIMPLE NOTIFICATION SERVICE

Create topic: standard

Name: stock

Create topic.

Create Subscription



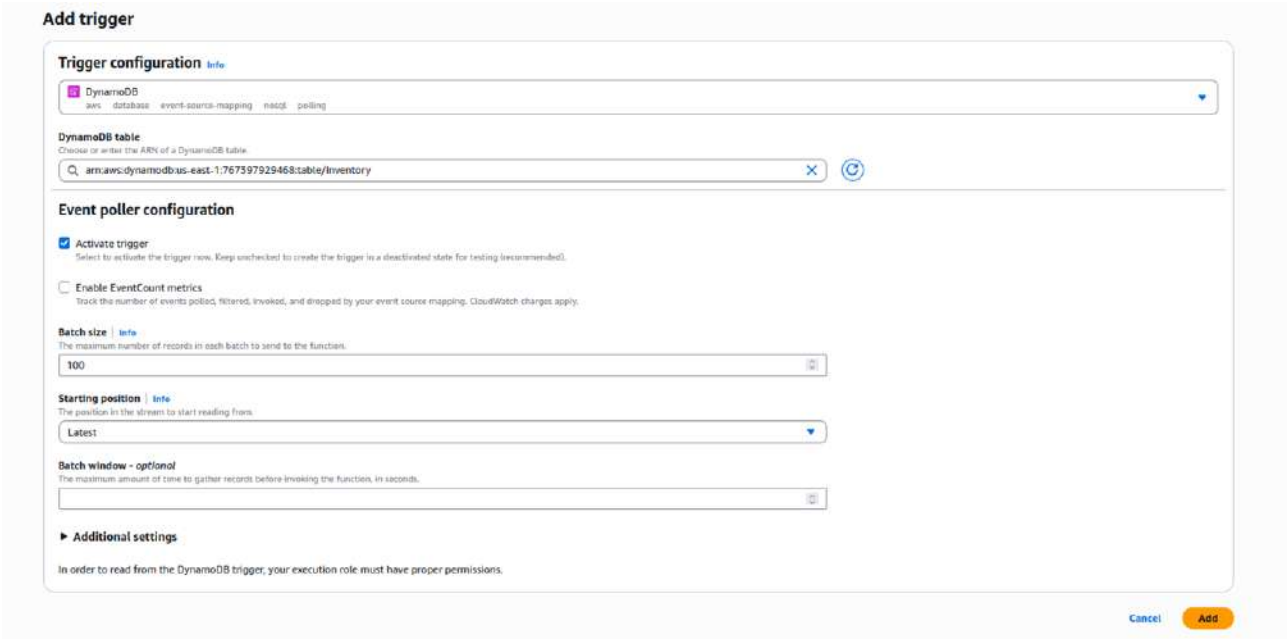
Go mail and confirm subscription.

Step 8:Trigger function

Go to lambda

Add trigger :dynamodb

Dnamo table:inventory



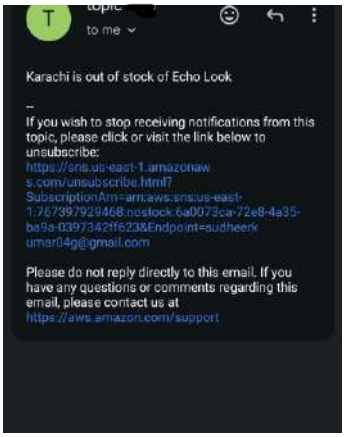

STEP 9: upload files

Go to S3 bucket and upload the csv files.

STEP 10: Notification

Go and check the email.

Email will report should be received.



AWS Elastic Beanstalk

**Elastic Beanstalk** is a fully managed service provided by AWS that makes it easy to deploy and manage applications in the cloud without worrying about the underlying infrastructure. It is a **Platform as a Service (PaaS)** that simplifies deploying, managing, and scaling web applications and services.

## How it works

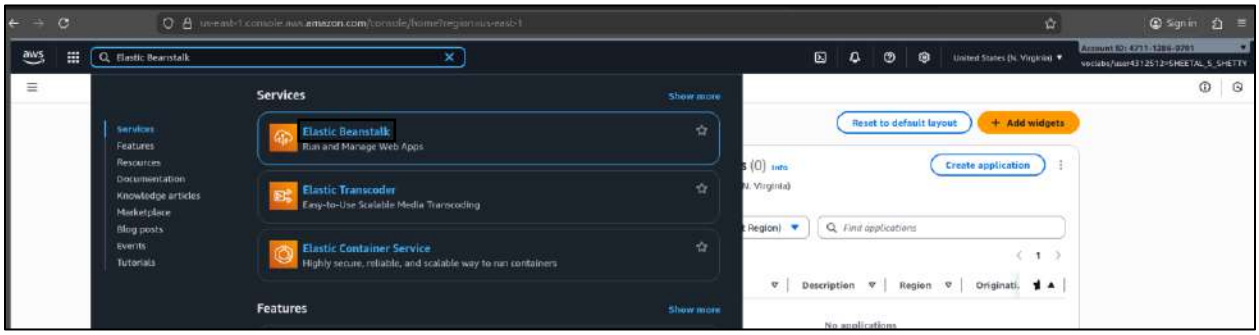
1. You upload your application code (Java, .NET, Node.js, Python, etc.).
2. Elastic Beanstalk automatically handles:
    a. Provisioning and configuring servers (EC2 instances)
    b. Setting up load balancers and Auto Scaling groups
    c. Managing the application environment
    d. Monitoring, logging, and health checks

## Who is it for

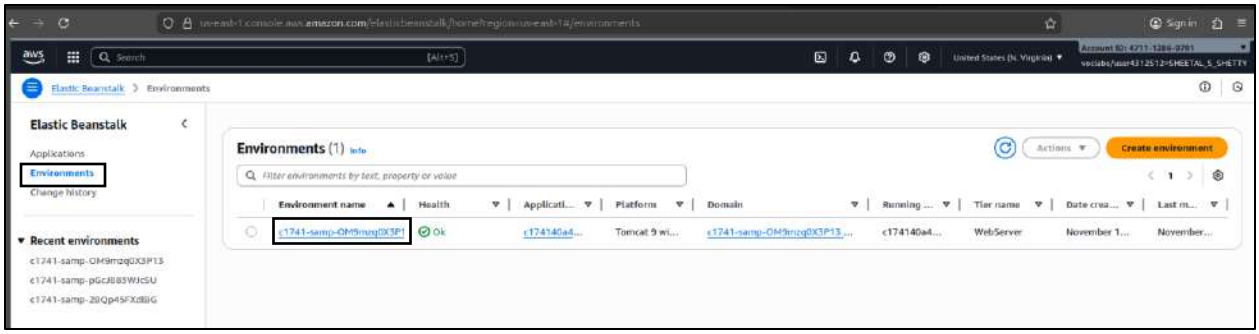
- Developers who want to focus on writing code rather than managing infrastructure
- Small to medium-sized applications that need quick and easy deployment
- Teams that want a managed environment but still need some level of control when required

Lab:

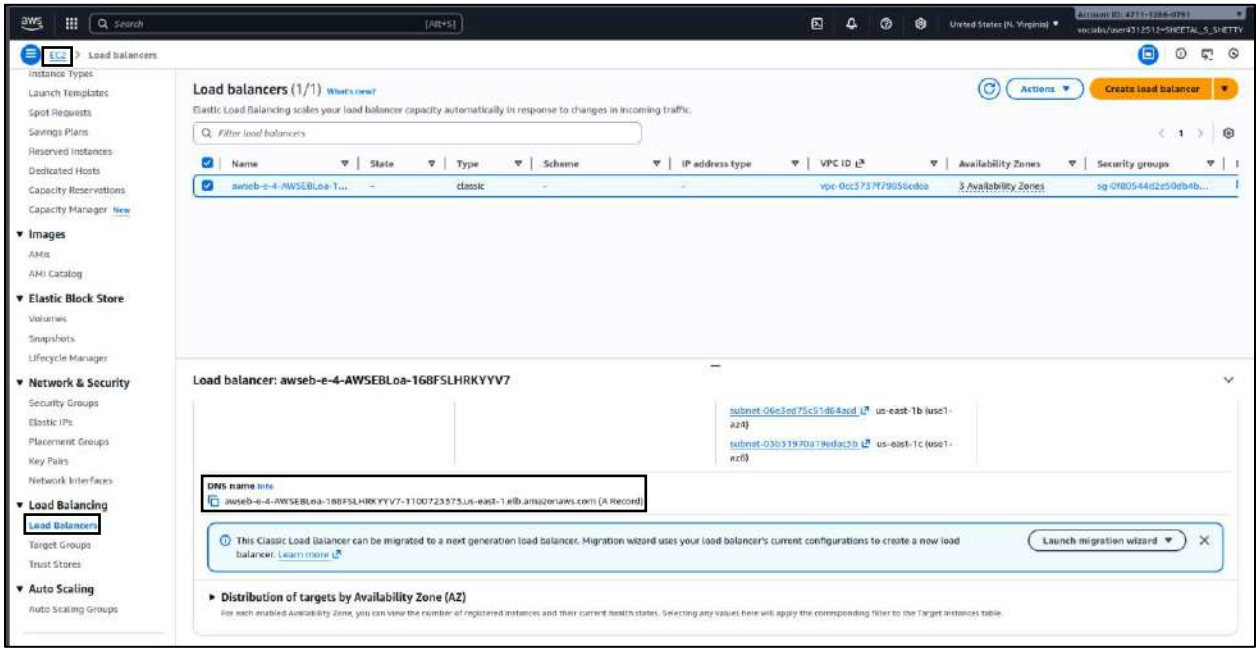
Step1: In the console, in the search box, search and choose **Elastic Beanstalk.**




Under the **Environment,** click on the name of the environment. This opens the Dashboard page for the Elastic Beanstalk environment.




Go to EC2-> Click **Load Balancer** from the left navigation menu and access the **Domain** Name.

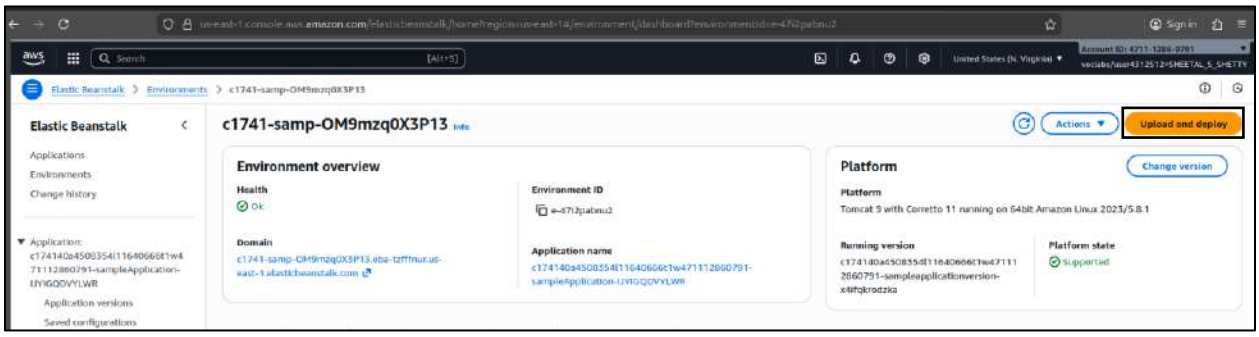


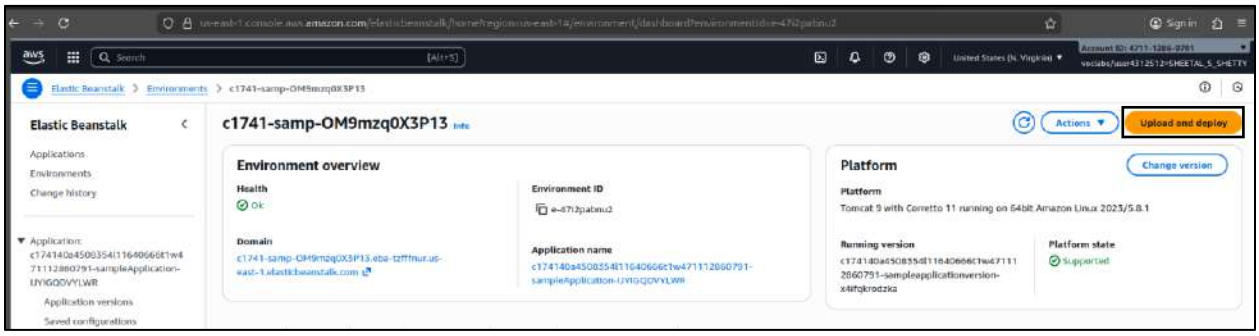

When you open the domain link in browser, AWS launches the EC2 instance with an application server. However, since no application code has been deployed to the Beanstalk environment yet, the page will not display any content. Instead, it will show an **HTTP Status 404 – Not Found** message. This is expected behavior until the application is deployed.


Step2: Deploying a sample application to Elastic Beanstalk.

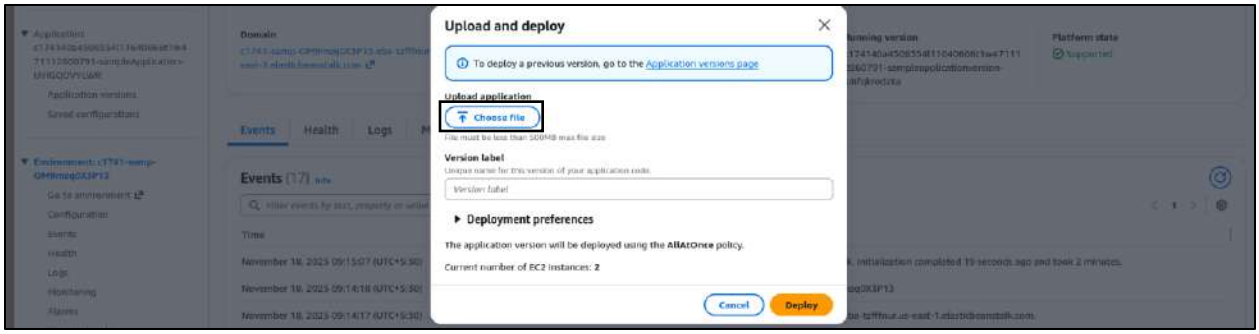
Go to the AWS current lab instruction page. There, you will find a link to download the sample **tomcat.zip** file. Click on the link, and the **tomcat.zip** file will be downloaded to your system.


After downloading, return to the Elastic Beanstalk dashboard and continue with the **Upload and Deploy** steps.



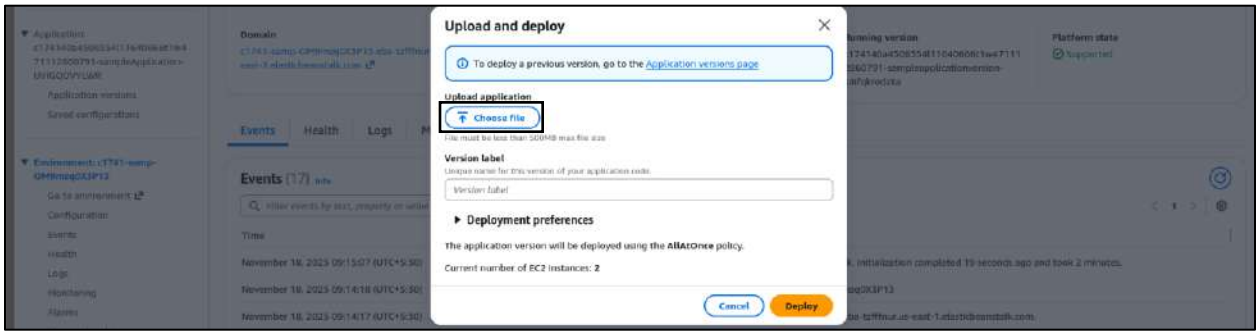




Select **Choose File**, then navigate to the **tomcat.zip** file you downloaded and open it.

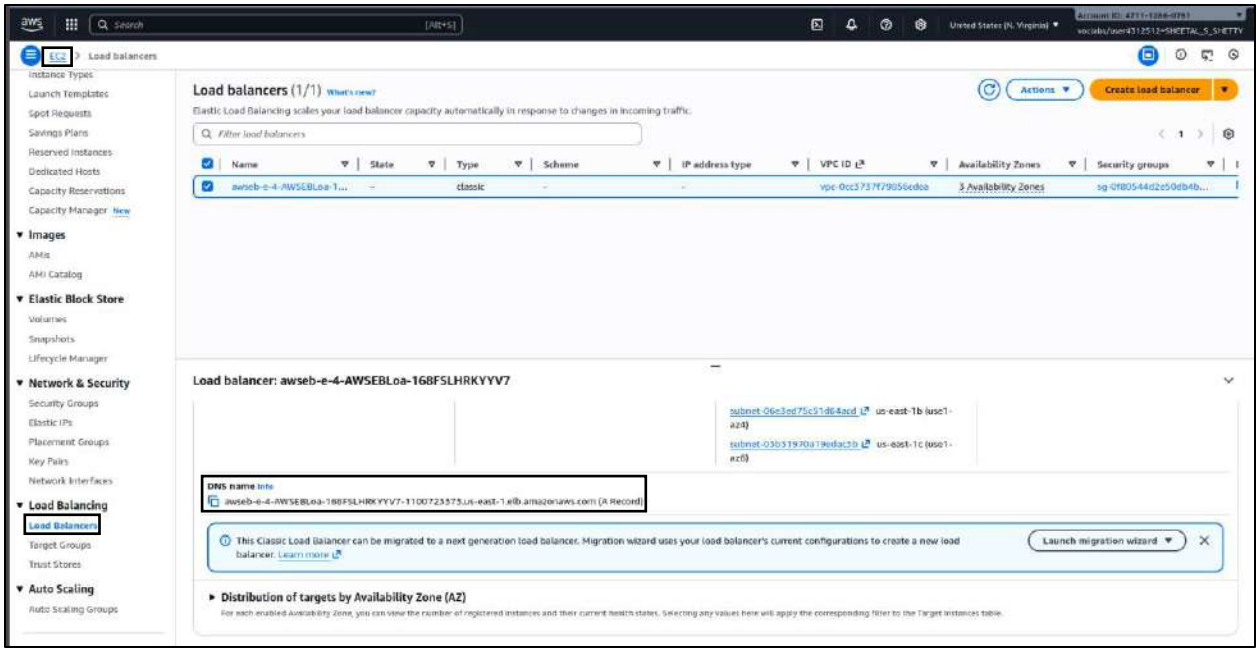



Next, choose **Deploy**.

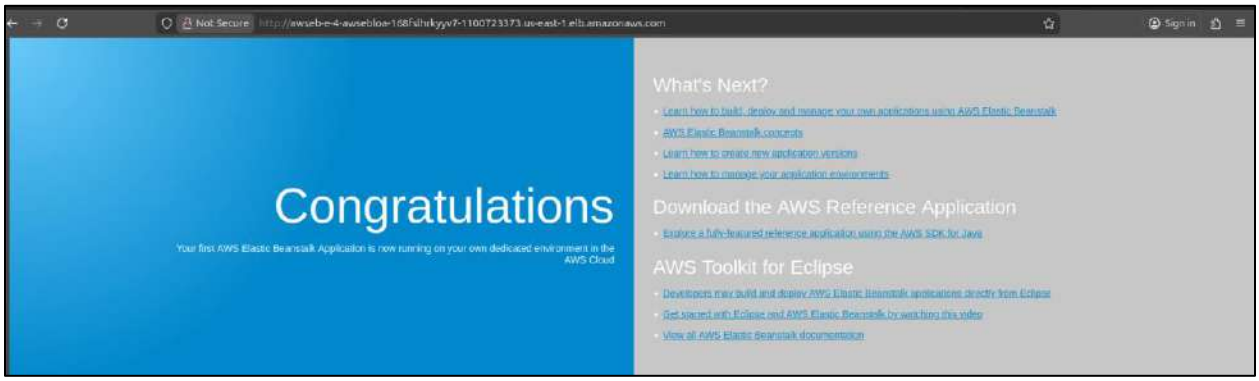

Elastic Beanstalk will take a minute or two to update your environment and deploy the application.




Step3: Once the deployment is complete, go to EC2-> Click **Load Balancer** from the left navigation menu and access the **Domain** Name.




The deployed **web application** will now be displayed.



# **Amazon Elastic File System (Amazon EFS)** Report

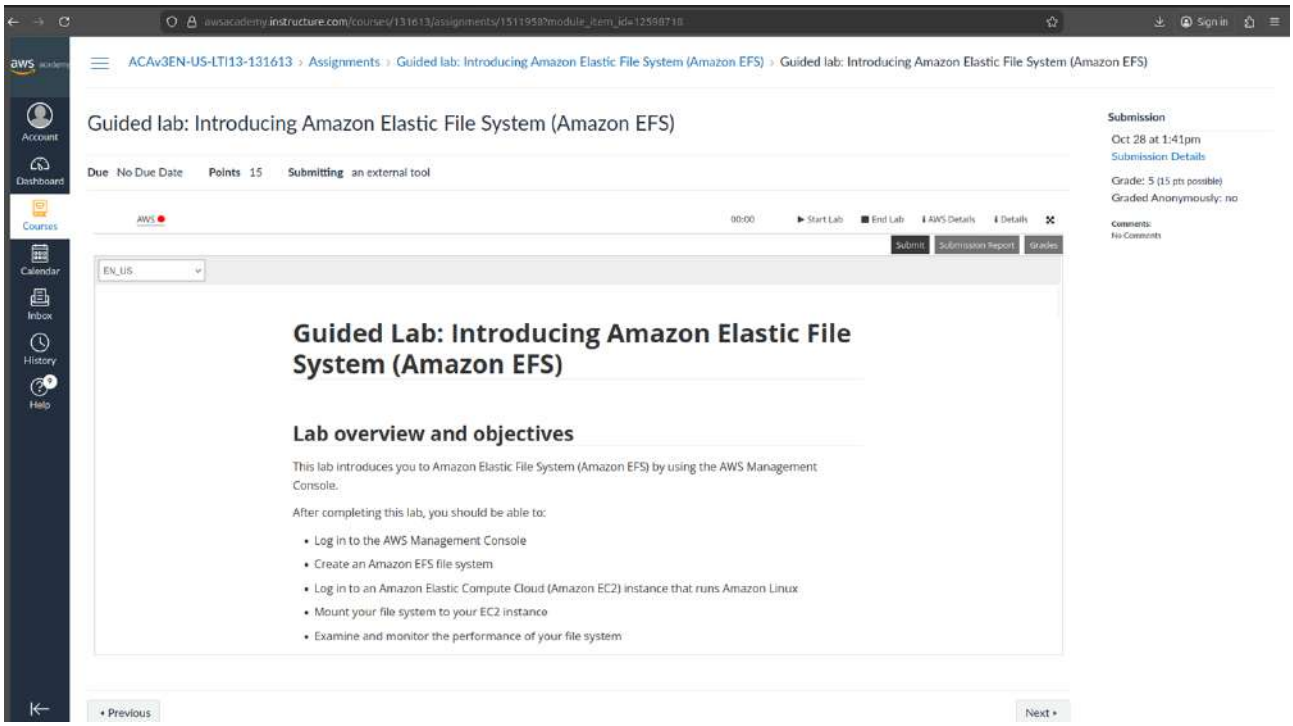
**Lab:** Introducing Amazon Elastic File System (Amazon EFS)

## Objectives

- Access the AWS console, create an EFS file system, and launch an Amazon Linux EC2 instance.
- Connect to the EC2 instance and mount the EFS file system.
- Review and monitor the file system's performance.

## Lab Environment Setup

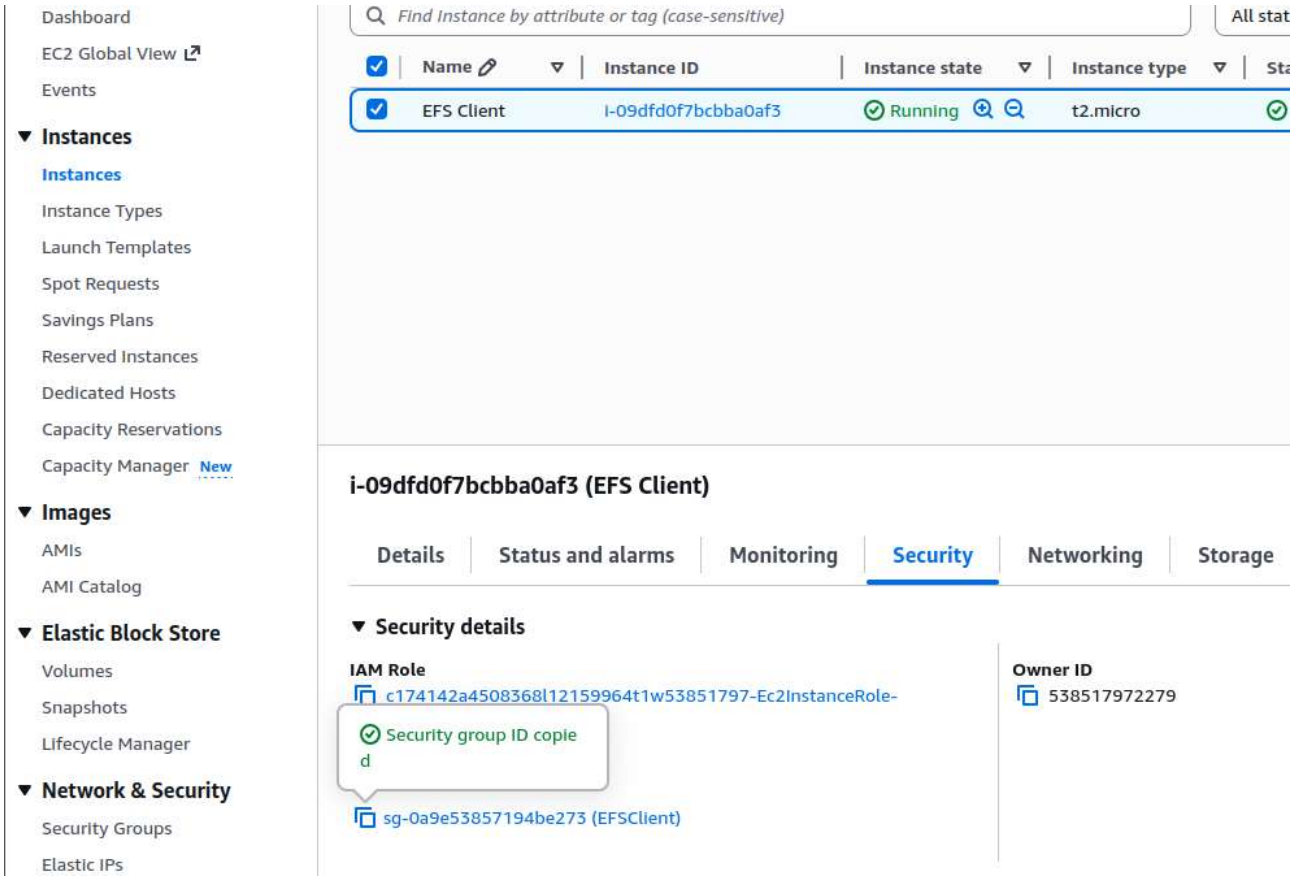
- This lab introduces you to Amazon Elastic File System (Amazon EFS) by using the AWS Management Console.
- A timer-based lab session was started using the "Start Lab" button.
- Pop-up windows were allowed in the browser to open the AWS Management Console in a new tab.
- Resources were named exactly as specified in the instructions to ensure the lab scoring script works properly.



# Task 1: Creating a security group to access your EFS file system

**Steps:**

1. At the top of the AWS Management Console, in the search box, search for and choose EC2.
2. In the navigation pane on the left, choose **Security Groups**.
3. Copy the **Security group ID** of the *EFSClient* security group to your text editor.The Group ID should look similar to *sg-03727965651b6659b*.

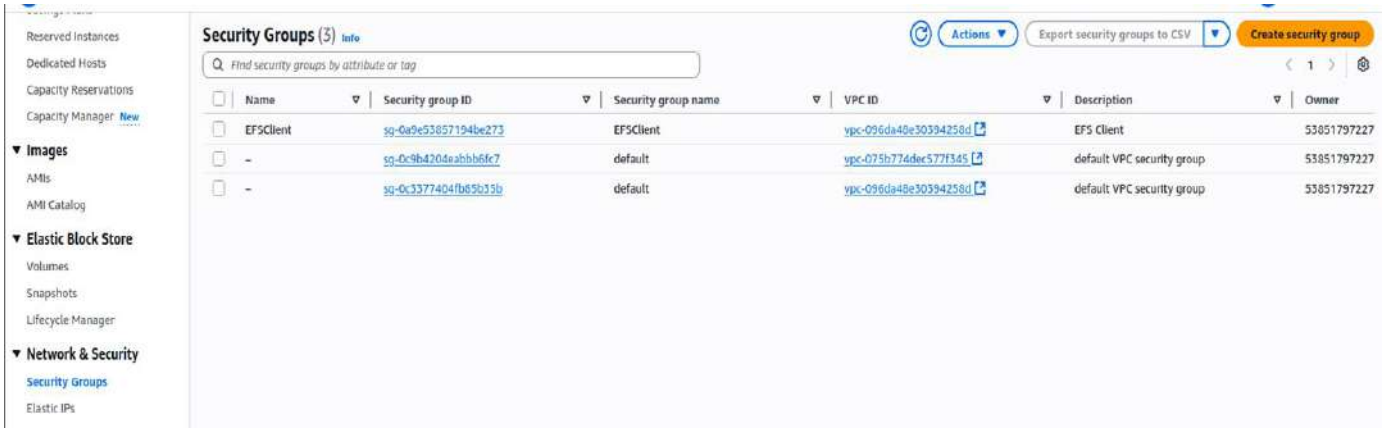


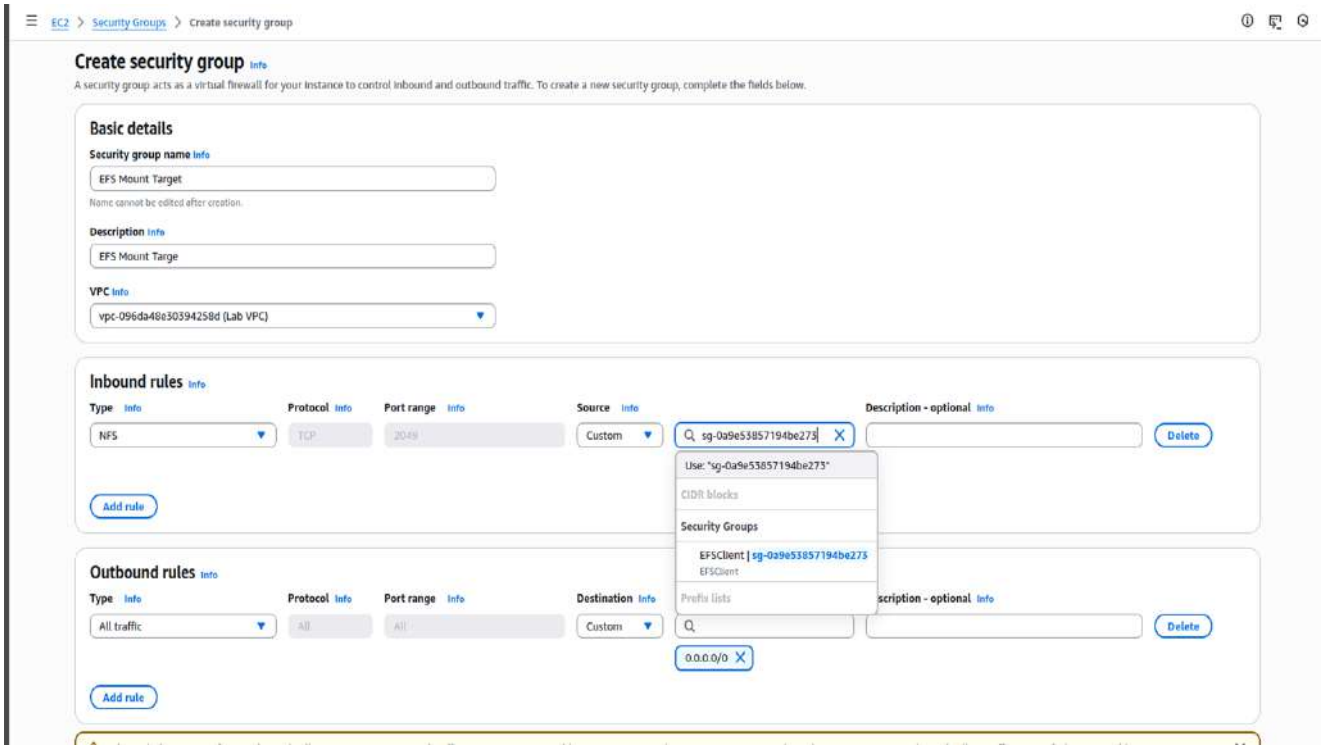

4. Choose **Create security group** then configure:

- **Security group name:** EFS Mount Target
- **Description:** Inbound NFS access from EFS clients
- **VPC:** *Lab VPC*

5. Under the **Inbound rules** section, choose **Add rule** then configure:

- **Type:** *NFS*
- **Source:**
  - *Custom*
  - In the *Custom* box, paste the security group's **Security group ID** that you copied to your text editor
- Choose **Create security group**.




# Task 2: Creating an EFS file system

**Steps:**

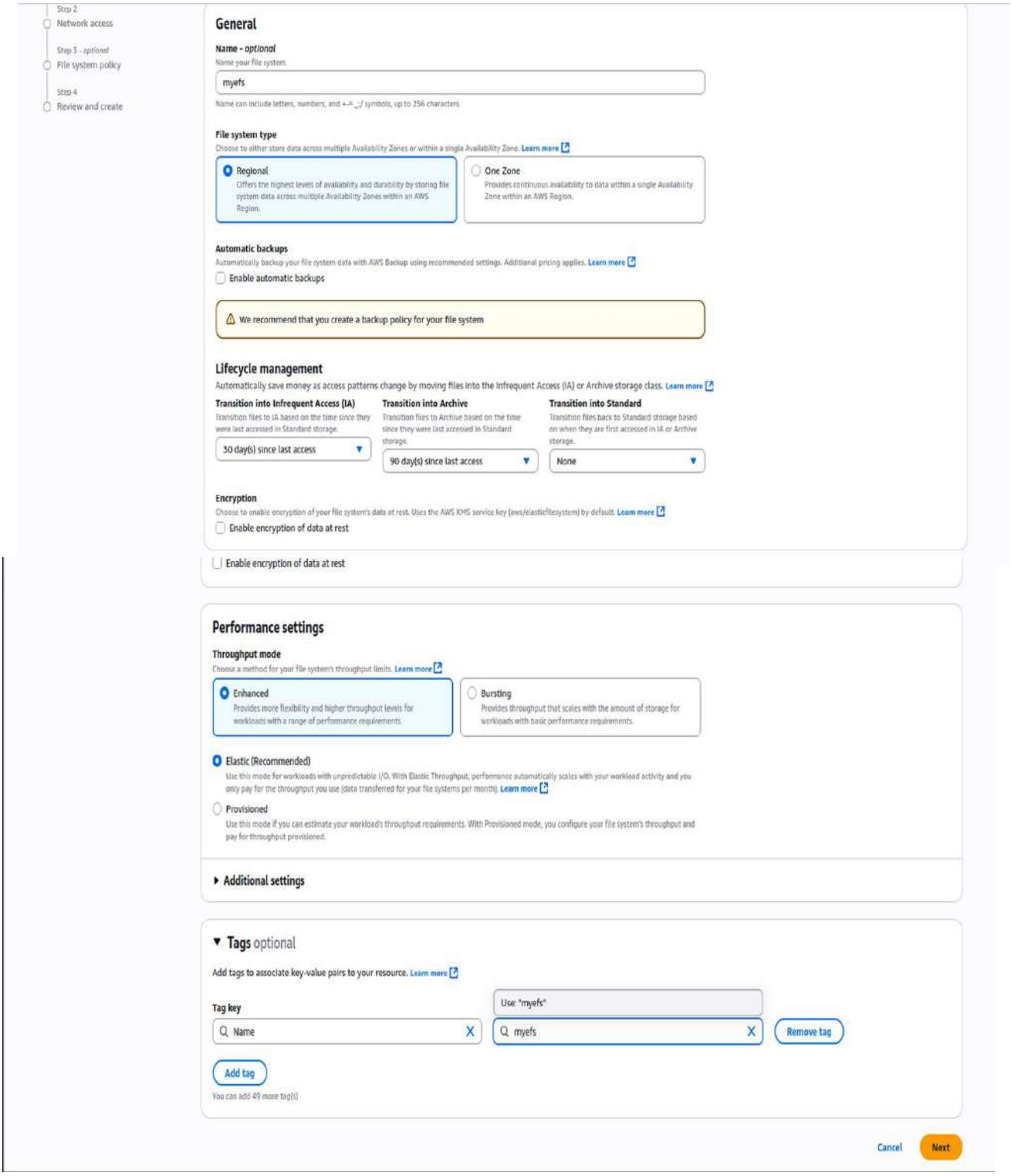
1. At the top of the AWS Management Console, in the search box, search for and choose EFS.
2. Choose **Create file system**.
3. In the **Create file system** window, choose **Customize**.
4. On **Step 1**:

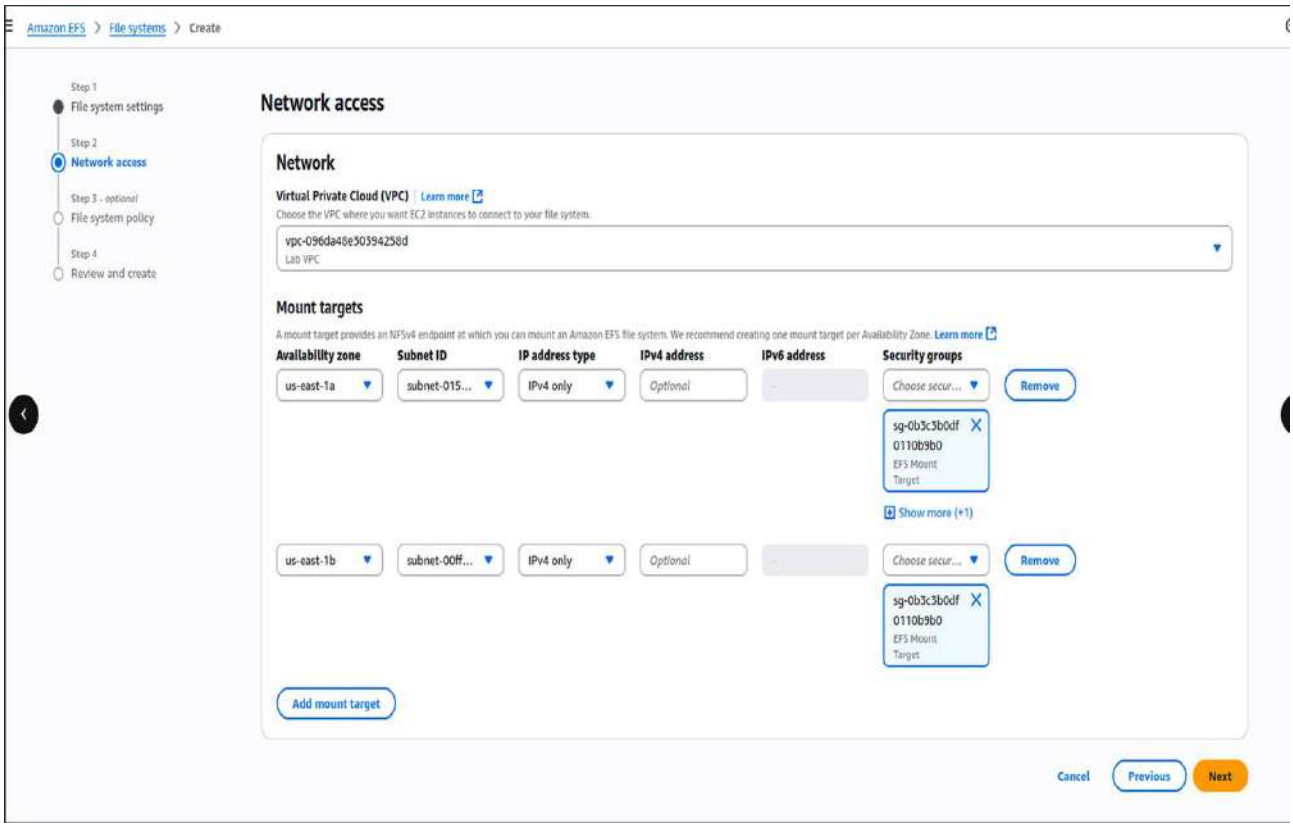
- Uncheck Enable Automatic backups.
- **Lifecycle management**:
- for **Transition into IA** Select *None*.
- In the **Tags** *optional* section, configure:
- **Key:** Name
- **Value:** myefs

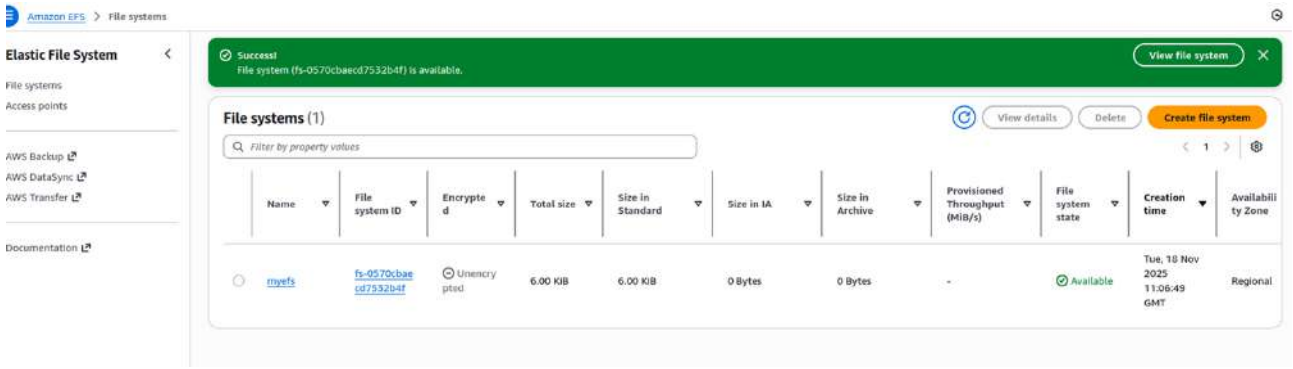
5. Choose Next.
6. For **VPC**, select *Lab VPC*.
7. Detach the default security group from each *Availability Zone* mount target by choosing the check box on each default security group.
8. Attach the **EFS Mount Target** security group to each *Availability Zone* mount target by choosing **EFS Mount Target** for each Availability Zone.
9. Choose **Next**.
10. On **Step 3**, choose Next.
11. On **Step 4:**

    Review your configuration.

    Choose Create.


Step 2
Network access
Step 3 - optional
File system policy
Step 4
Review and create
General
Name - optional
Name your file system.
myefs
Name can include letters, numbers, and +-=._:/ symbols, up to 256 characters.
File system type
Choose to either store data across multiple Availability Zones or within a single Availability Zone. Learn more
Regional
Offers the highest levels of availability and durability by storing file system data across multiple Availability Zones within an AWS Region.
One Zone
Provides continuous availability to data within a single Availability Zone within an AWS Region.
Automatic backups
Automatically backup your file system data with AWS Backup using recommended settings. Additional pricing applies. Learn more
Enable automatic backups
We recommend that you create a backup policy for your file system
Lifecycle management
Automatically save money as access patterns change by moving files into the Infrequent Access (IA) or Archive storage class. Learn more
Transition into Infrequent Access (IA)
Transition files to IA based on the time since they were last accessed in Standard storage.
30 day(s) since last access
Transition into Archive
Transition files to Archive based on the time since they were last accessed in Standard storage.
90 day(s) since last access
Transition into Standard
Transition files back to Standard storage based on when they are first accessed in IA or Archive storage.
None
Encryption
Choose to enable encryption of your file system's data at rest. Uses the AWS KMS service key (aws/elasticfilesystem) by default. Learn more
Enable encryption of data at rest
Enable encryption of data at rest
Performance settings
Throughput mode
Choose a method for your file system's throughput limits. Learn more
Enhanced
Provides more flexibility and higher throughput levels for workloads with a range of performance requirements.
Bursting
Provides throughput that scales with the amount of storage for workloads with basic performance requirements.
Elastic (Recommended)
Use this mode for workloads with unpredictable I/O. With Elastic Throughput, performance automatically scales with your workload activity and you only pay for the throughput you use (data transferred for your file systems per month). Learn more
Provisioned
Use this mode if you can estimate your workload's throughput requirements. With Provisioned mode, you configure your file system's throughput and pay for throughput provisioned.
Additional settings
Tags optional
Add tags to associate key-value pairs to your resource. Learn more
Tag key
Use: "myefs"
Name
myefs
Remove tag
Add tag
You can add 49 more tag(s)
Cancel
Next

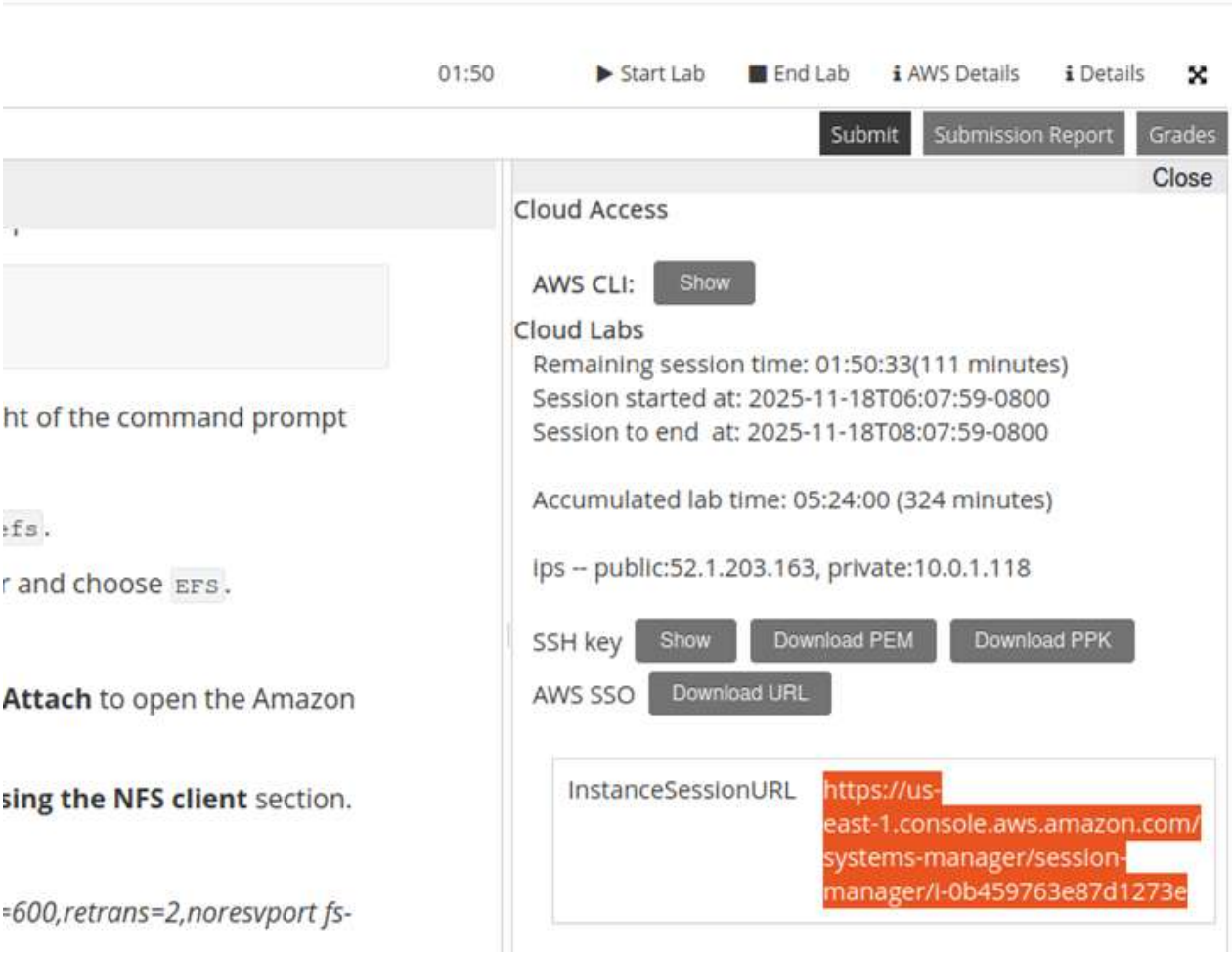



# Task 3: Connecting to your EC2 instance

1. To connect to the **EC2 instance**, from the top of this page, choose **i AWS Details** and copy the value for *InstanceSessionURL* .
2. Paste it into the new browser tab or window to connect to the EC2 instance using AWS Systems Manager Session Manager.

   You should now be connected to the instance.




# Task 4: Creating a new directory and mounting the EFS file system

1. In your EC2 terminal session, run the following command to install the required utilities:

   sudo su -l ec2-user
   sudo yum install -y amazon-efs-utils

2. Run the following command to create directory for mount: `sudo mkdir efs`.

3. At the top of the AWS Management Console, in the search box, search for and choose EFS.

4. Choose **myefs.**

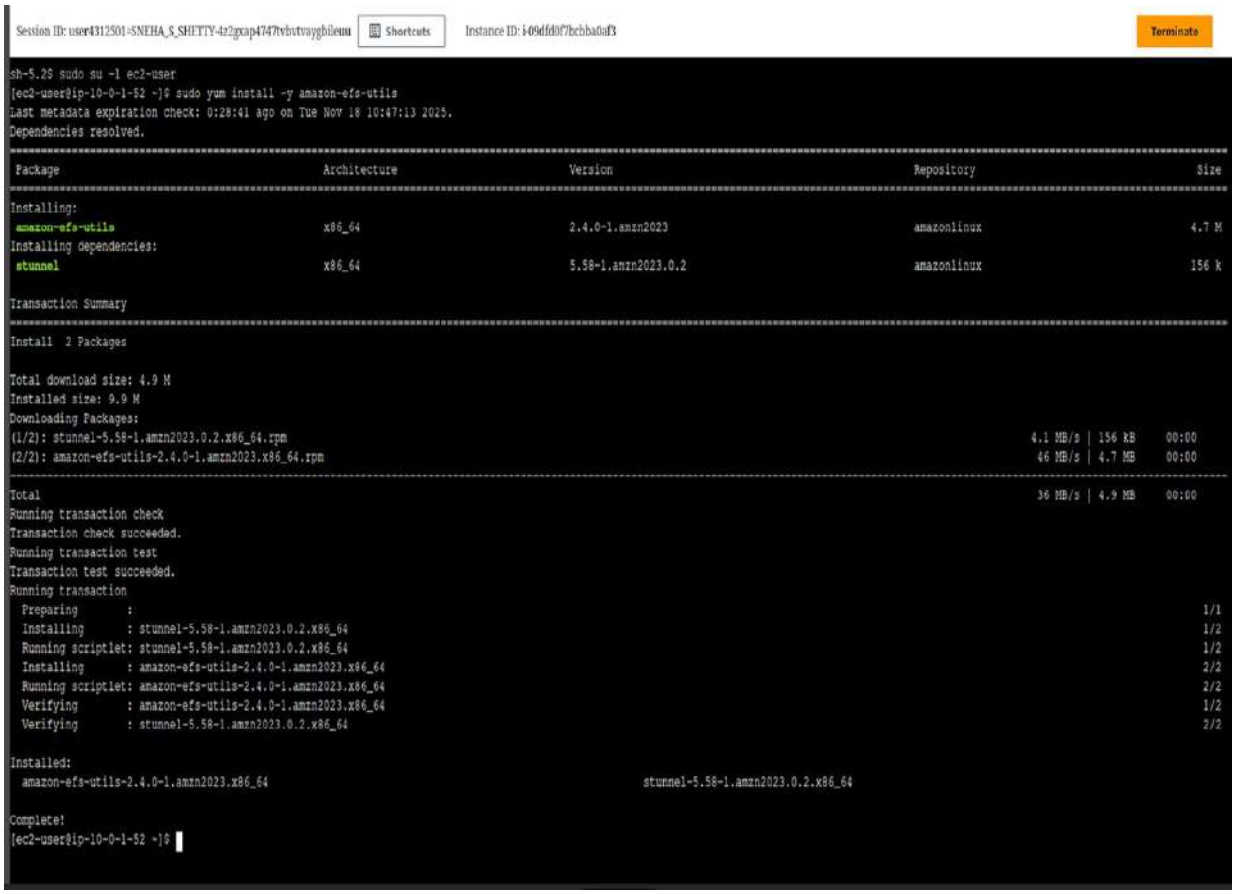
5. In the **Amazon EFS Console**, on the top right corner of the page, choose **Attach** to open the Amazon EC2 mount instructions.

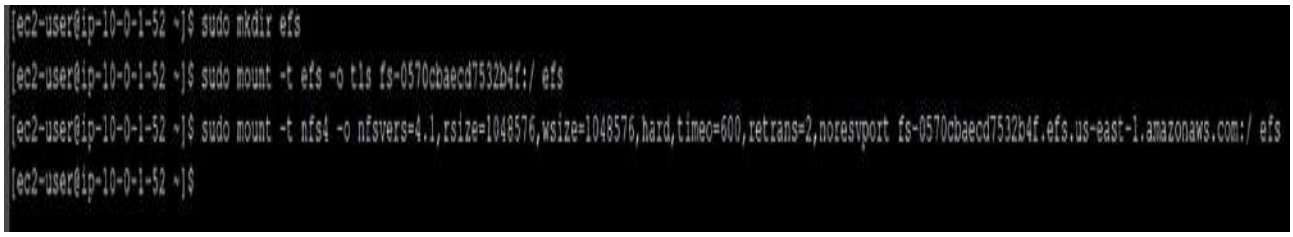
6. In your EC2 terminal session, Copy and run the entire command in the **Using the NFS client section.**

sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-bce57914.efs.us-west-2.amazonaws.com:/ efs

7. Get a full summary of the available and used disk space usage by entering:

```
sudo df -hT
```




# Task 5: Examining the performance behavior of your new EFS file system

## Monitoring performance by using Amazon CloudWatch

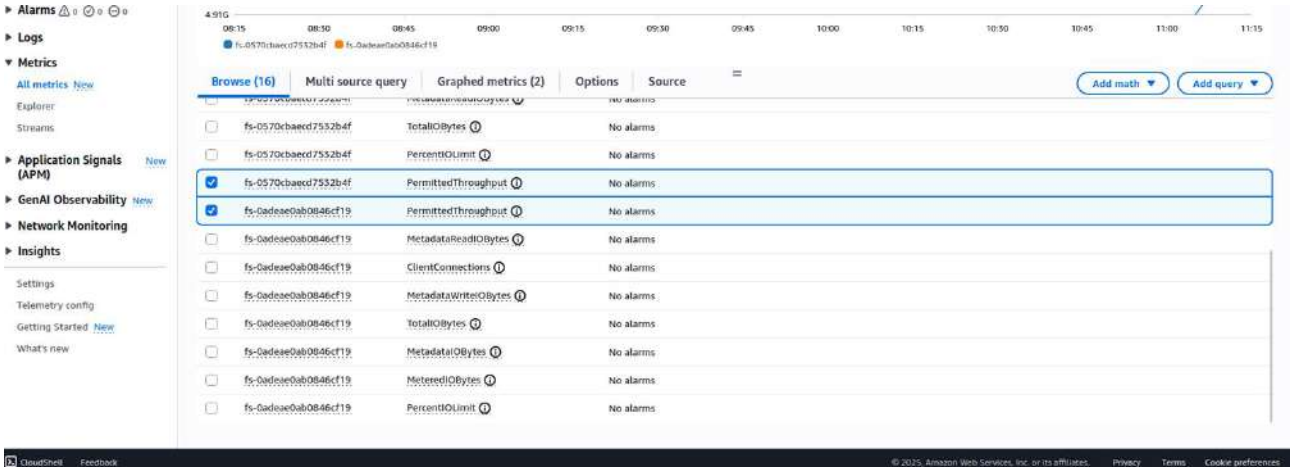
1. At the top of the AWS Management Console, in the search box, search for and choose `CloudWatch`.

2. In the navigation pane on the left, choose **All Metrics**.

3. In the **All metrics** tab, choose **EFS**.

4. Choose **File System Metrics**.

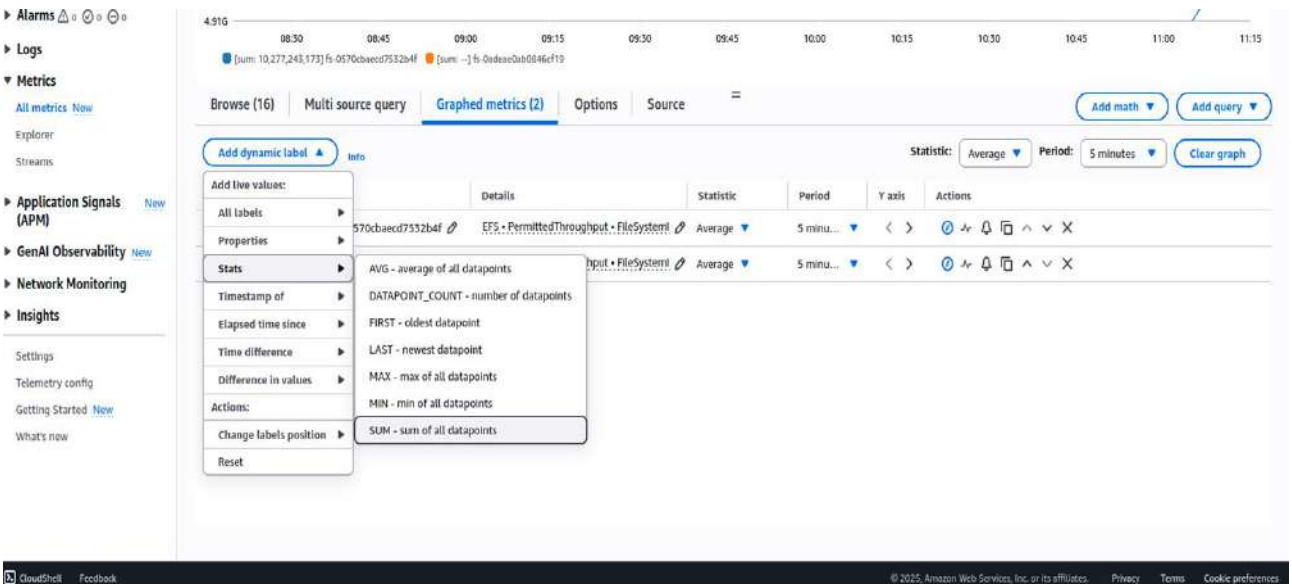
5. Select all the options that has the **PermittedThroughput** Metric Name.



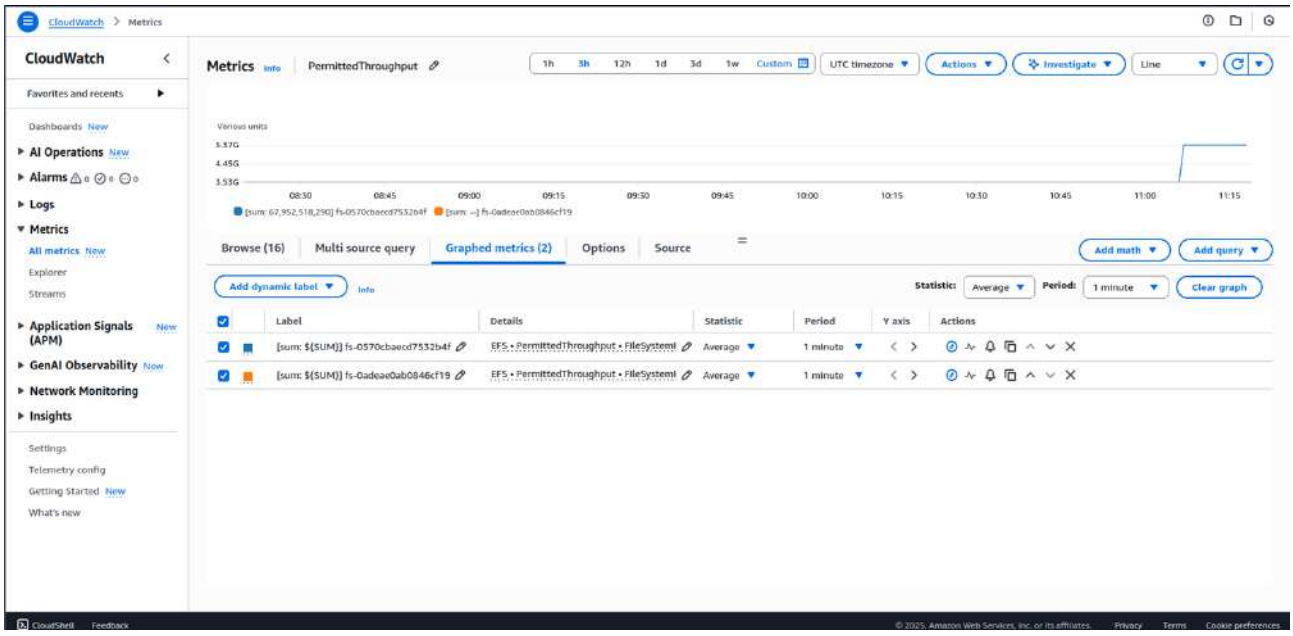

6. Choose the **Graphed metrics** tab.

7. On the **Statistics** column, select **Sum**.

8. On the **Period** column, select **1 Minute**.




9. Note the the peak value, which is around 7.6G. Take this number (in bytes) and divide it by the duration in seconds (60 seconds). The result gives you the write throughput (B/s) of your file system during your test.



# Elastic ( user data) Introduction to EC2

Cloud foundations Module 6  Lab3

## Task 1: Launch Your Amazon EC2 Instance

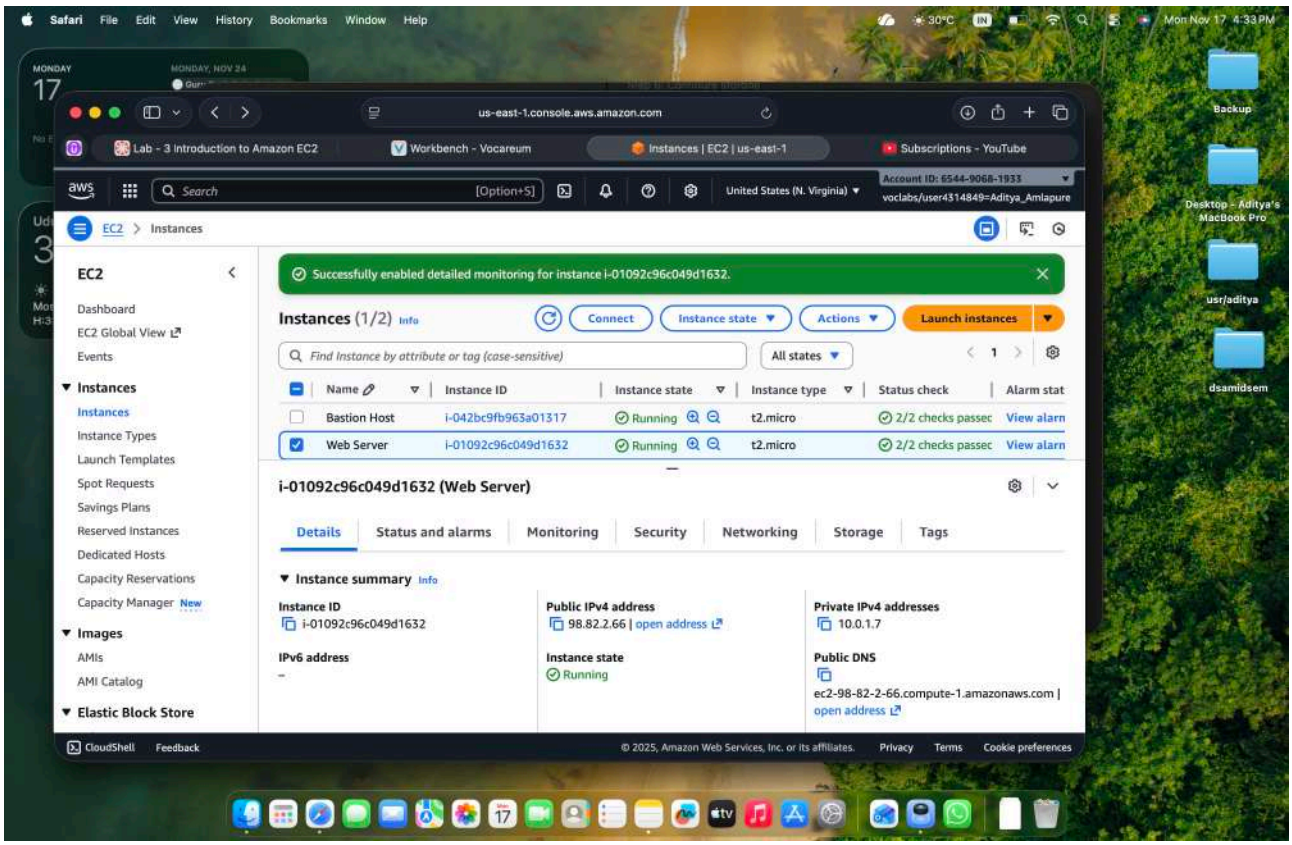
Step 1: Name and tags - Web Server
Step 2: Application and OS Images (Amazon Machine Image) -Amazon Linux AMI

Step 3: Instance type - t2.micro
Step 4: Key pair (login) - vockey
Step 5: Network settings - Lab VPC - PublicSubnet1
Firewall (security groups) - Create security group
Security group name: Web Server security group
Description: Security group for my web server
Inbound security group rules- remove existing rules.
Step 6: Configure storage
Step 7: Advanced details
- Termination protection, select Enable
- user data box

```
#!/bin/bash
dnf install -y httpd
systemctl enable httpd
systemctl start httpd
echo '<html><h1>Hello From Your Web Server!</h1></html>' > /var/www/html/index.html
```

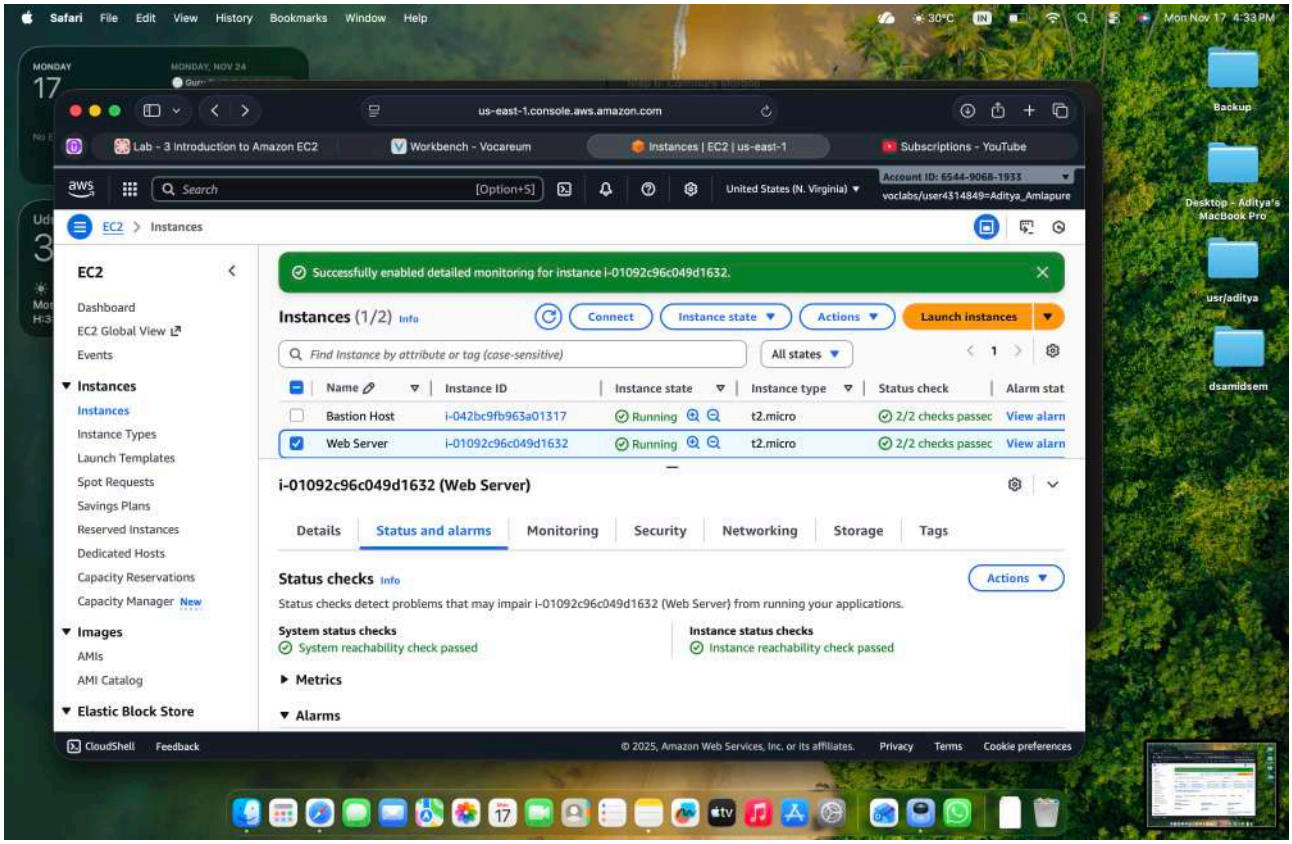
The script will:
- Install an Apache web server (httpd)
- Configure the web server to automatically start on boot
- Run the Web server once it has finished installing
- Create a simple web page

Step 8: Launch the instance

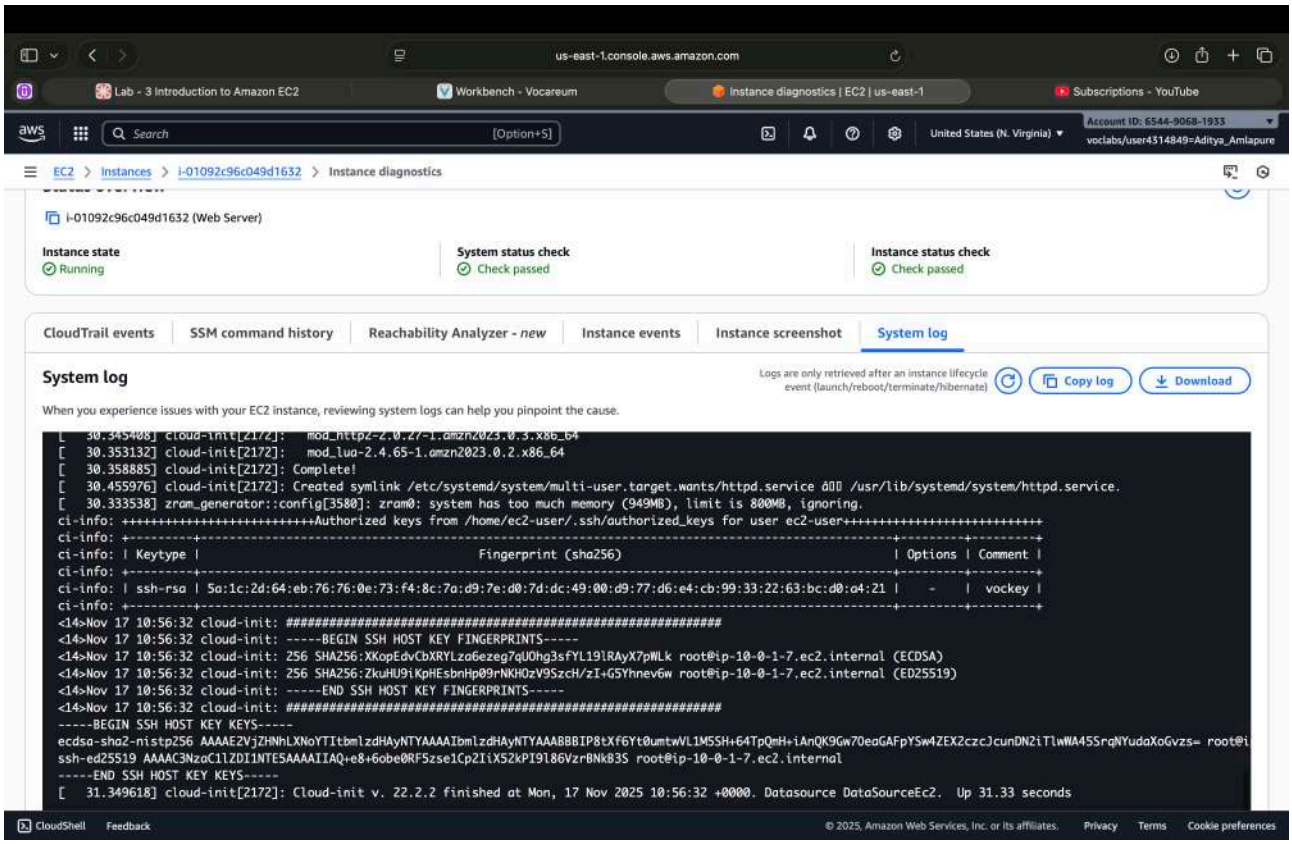



# Task 2: Monitoring  Your Instance

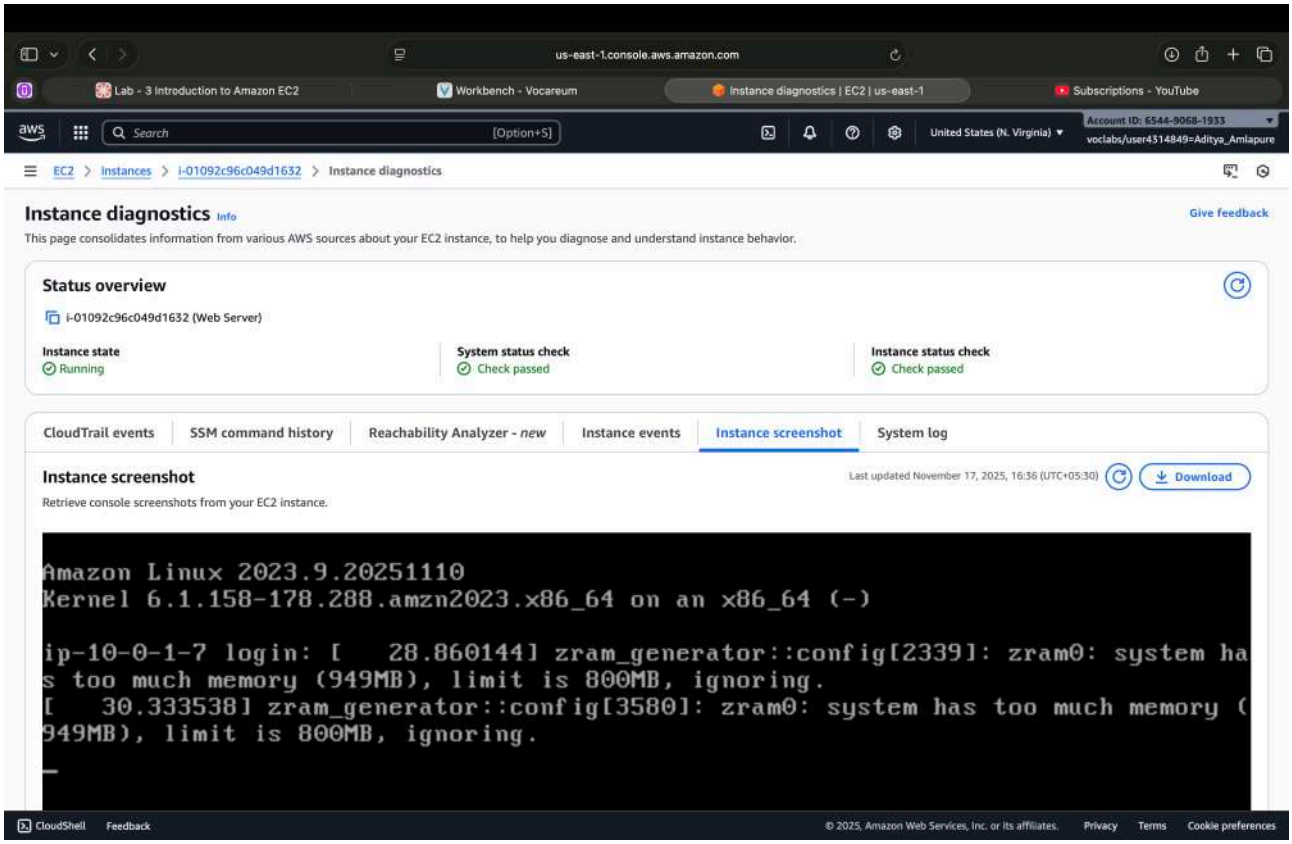
Status checks tab
System reachability and Instance reachability



Monitor and troubleshoot  Get system log.

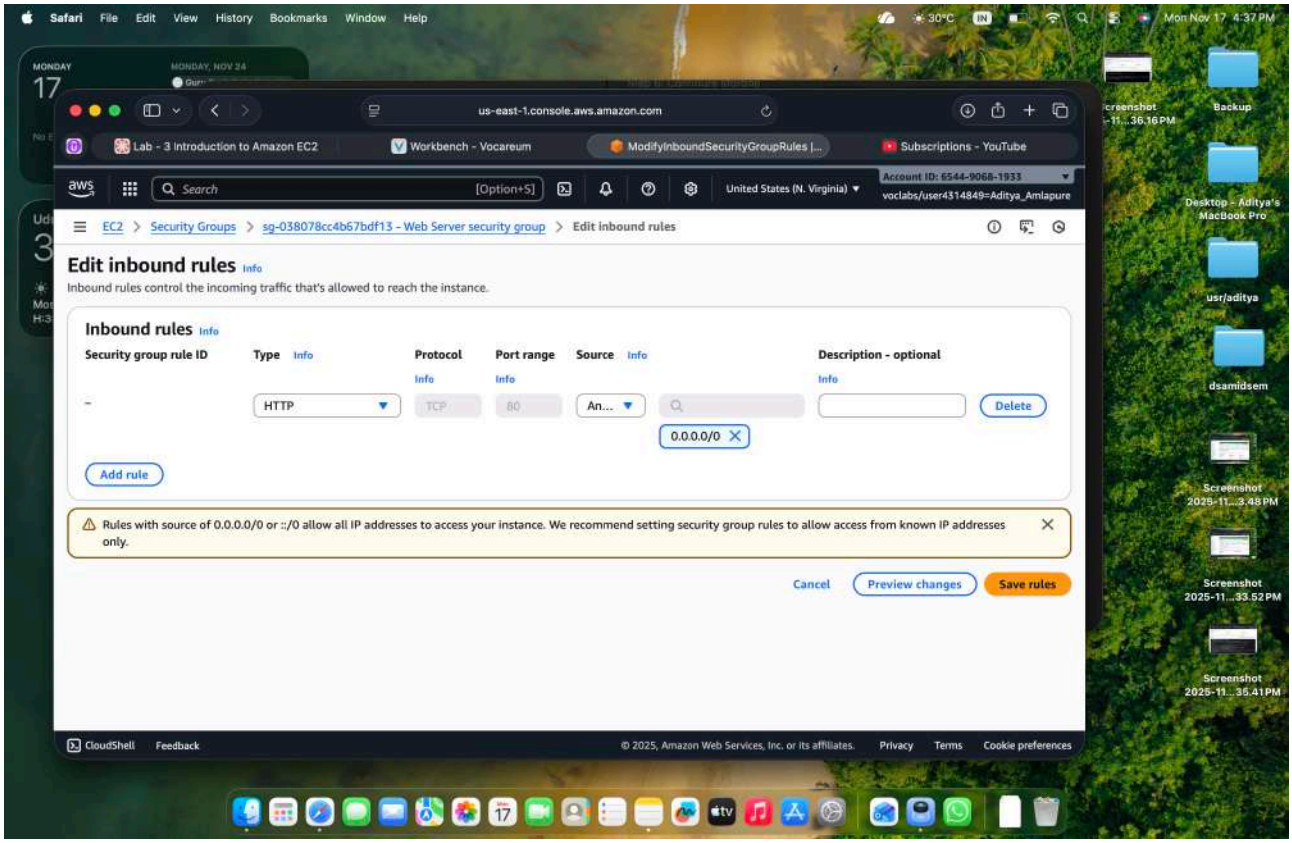



Monitor and troubleshoot  Get instance screenshot.

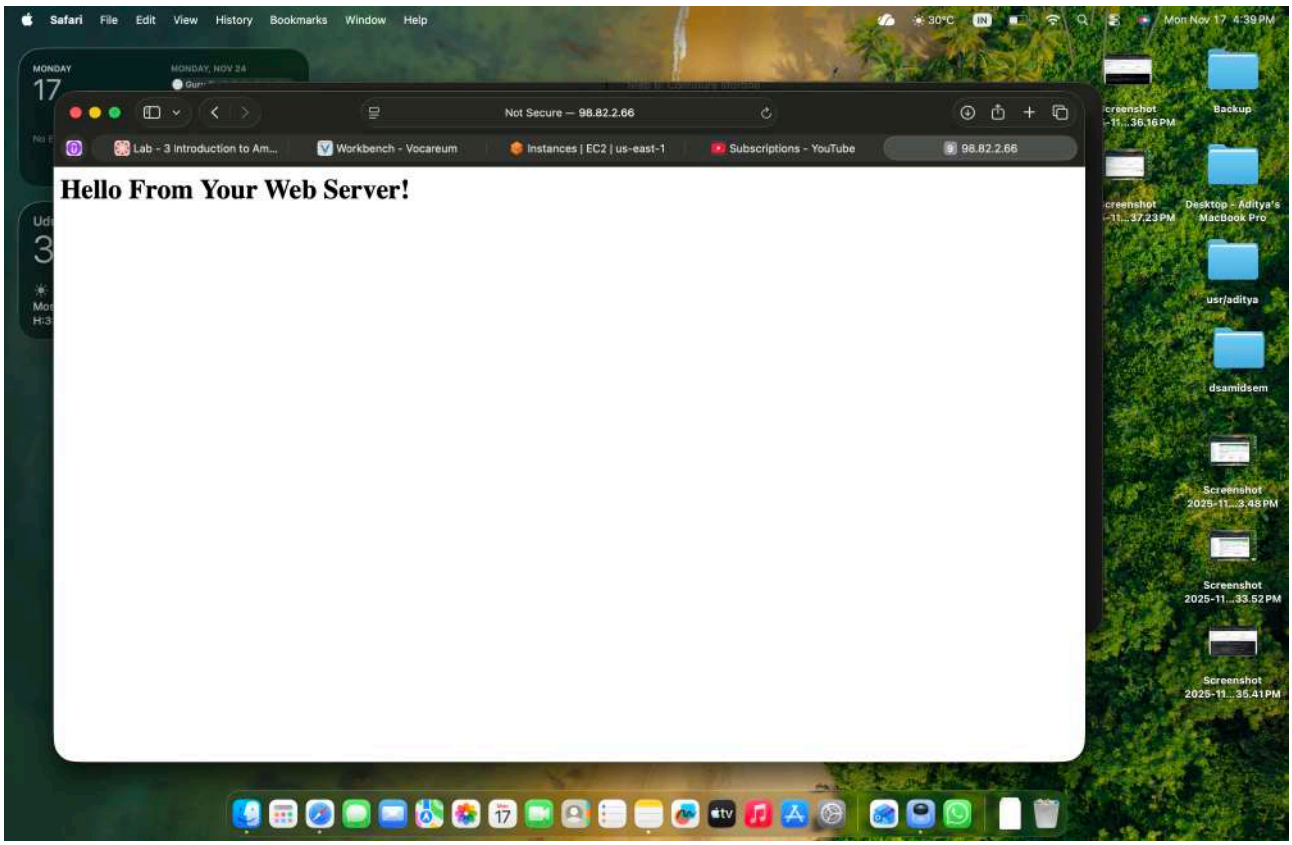



Task 3: Update Your Security Group and Access the Web Server

Web Server security group - Inbound rules -  edit
- Type: HTTP
- Source: Anywhere-IPv4



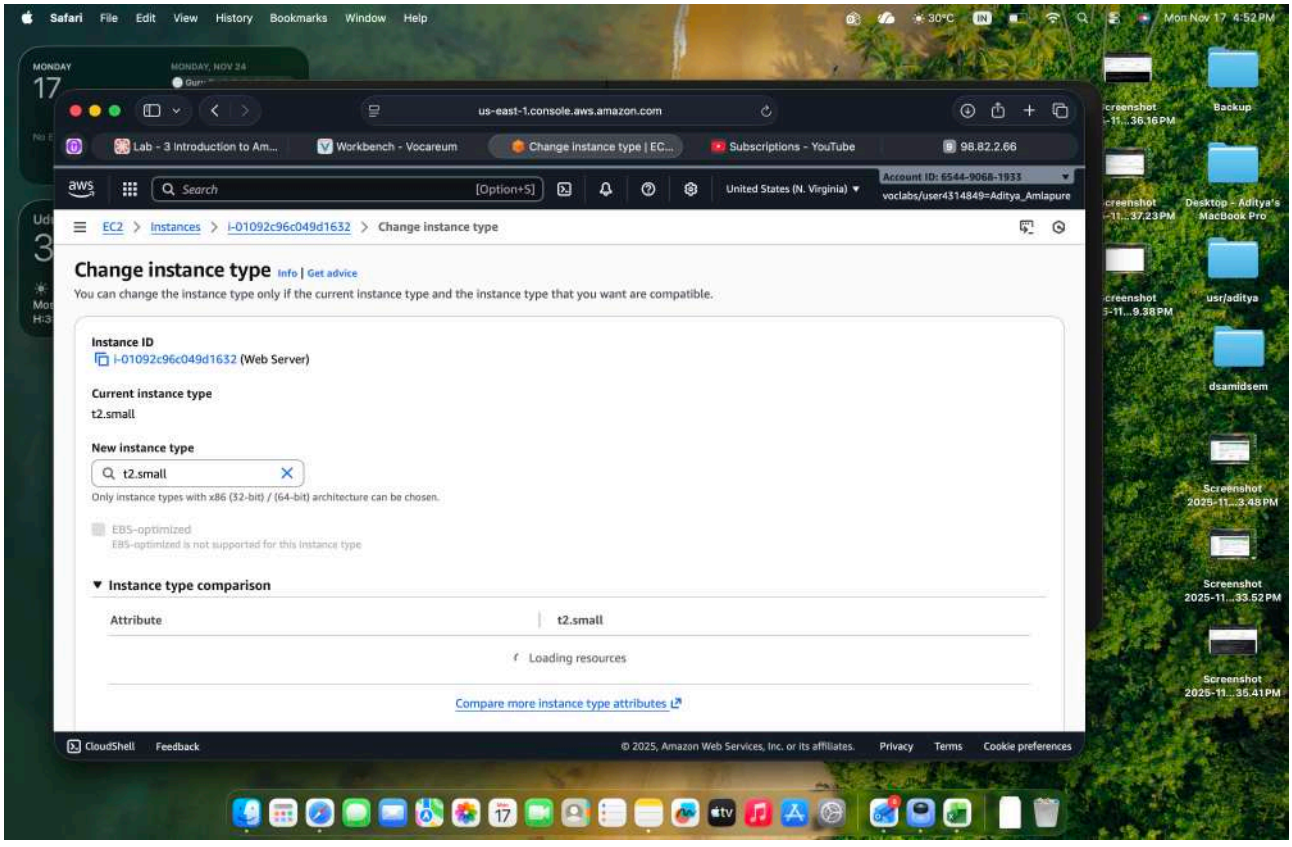

Return to the web server tab that you previously opened and refresh  the page.
You should see the message Hello From Your Web Server!

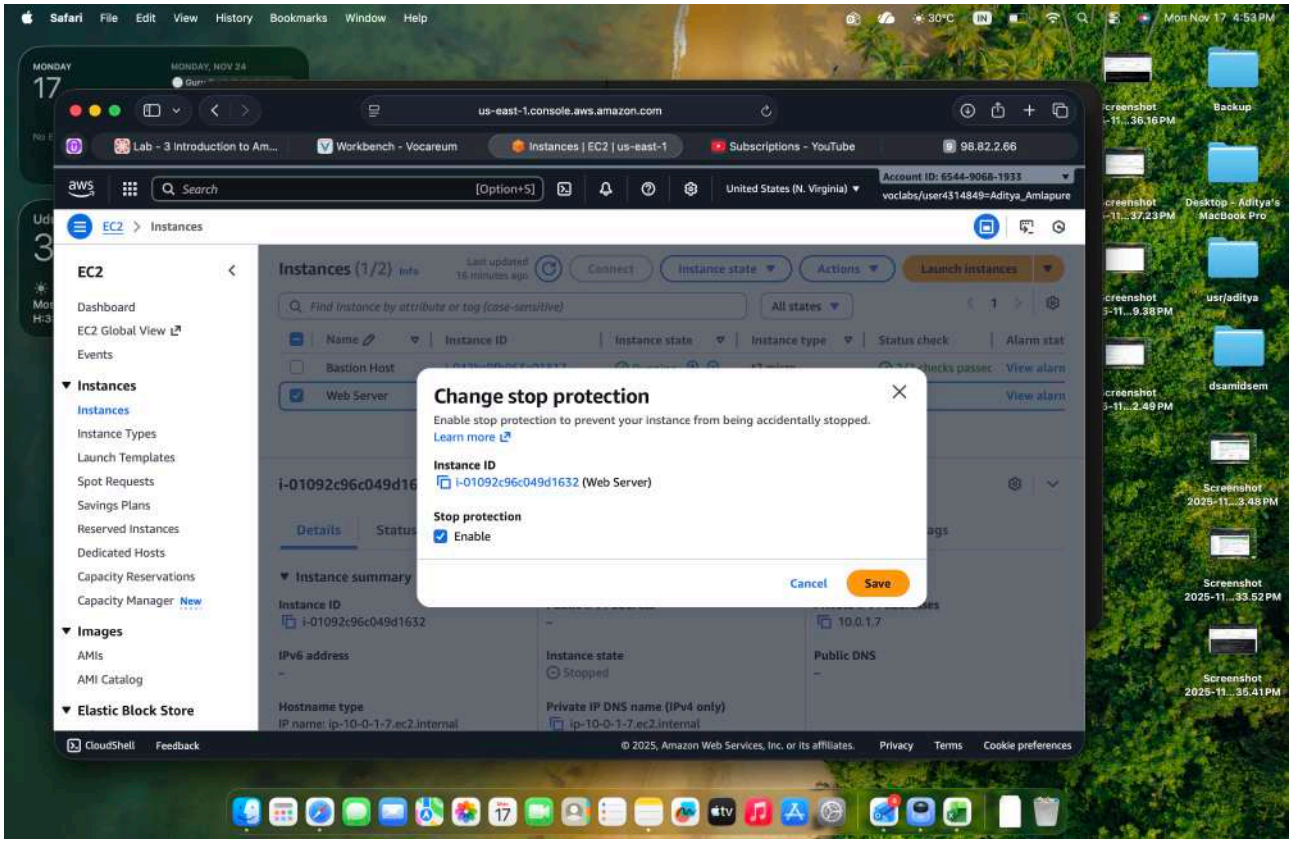



Task 4: Resize Your Instance: Instance Type and EBS Volume

Stop Your Instance

Change The Instance Type  ( t2.small) and enable stop protection



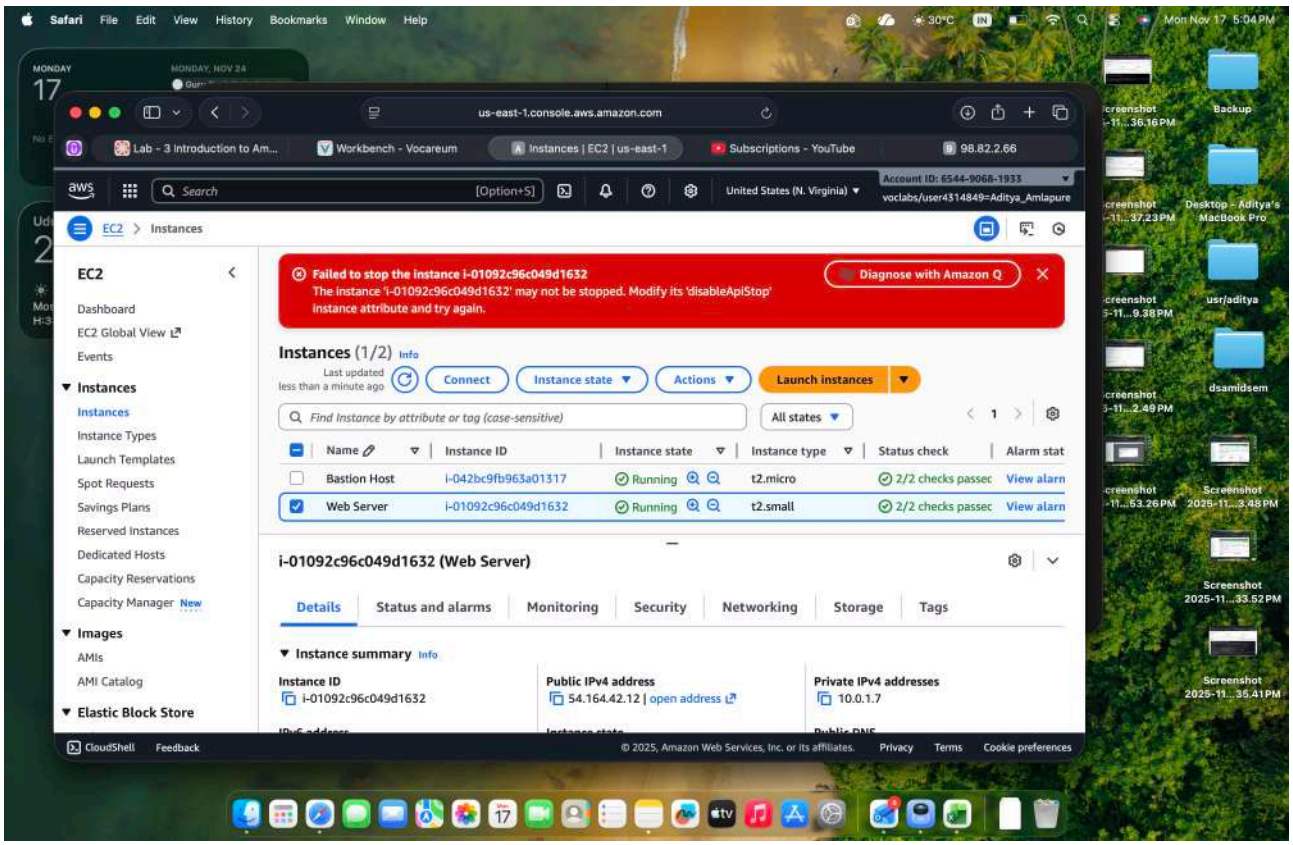



Resize the EBS Volume.  10

Start the stopped instance

Task 5: Explore EC2 Limits
Service Quotas

Task 6: Test Stop Protection


us-east-1.console.aws.amazon.com
Lab - 3 Introduction to Am...
Workbench - Vocareum
Instances | EC2 | us-east-1
Subscriptions - YouTube
98.82.2.66
Search
[Option+S]
United States (N. Virginia)
Account ID: 6544-9068-1933
voclabs/user4314849=Aditya_Amlapure
EC2 > Instances
EC2
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Capacity Manager New
Images
AMIs
AMI Catalog
Elastic Block Store
Failed to stop the instance i-01092c96c049d1632
The instance 'i-01092c96c049d1632' may not be stopped. Modify its 'disableApiStop' instance attribute and try again.
Diagnose with Amazon Q
Instances (1/2) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm stat
Bastion Host
i-042bc9fb963a01317
Running
t2.micro
2/2 checks passec
View alarn
Web Server
i-01092c96c049d1632
Running
t2.small
2/2 checks passec
View alarn
i-01092c96c049d1632 (Web Server)
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
Instance summary Info
Instance ID
i-01092c96c049d1632
Public IPv4 address
54.164.42.12 | open address
Private IPv4 addresses
10.0.1.7
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

# Creating a Scalable and Highly Available Environment for the Café

## TASK 1 — Update Network for Multi-AZ High Availability

### Task 1.1 — Create NAT Gateway in Public Subnet 2

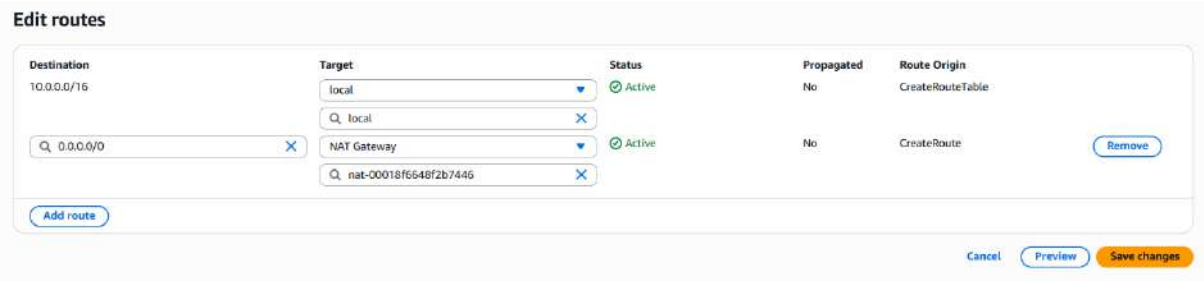
**Steps**

1. Open **VPC Console**.
2. Left menu → **NAT Gateways**.
3. Click **Create NAT Gateway**.
4. Configure:
    - **Subnet:** Public Subnet 2
    - **Elastic IP:** Allocate Elastic IP
    - **Name:** NAT-GW-AZ2
5. Click **Create NAT Gateway**.


### Update Private Subnet 2 Route Table

1. VPC console → **Route Tables**.
2. Find the route table for **Private Subnet 2**
3. Select it → **Routes** tab → **Edit routes**.
4. Add route:
    - **Destination:** 0.0.0.0/0
    - **Target:** NAT Gateway → NAT-GW-AZ2

5. Click **Save changes**.




## TASK 2 — Create a Launch Template

**1. Open Launch Templates**

EC2 Console → left menu → **Launch Templates** → **Create launch template**

**2. Template Information**

- Name: CafeWebServerTemplate

**3. AMI**

- Application & OS Images → **My AMIs**
- Select: **Cafe WebServer Image**

**4. Instance Type**

- Choose: **t2.micro**

**5. Key Pair**

- vokey

**6. Network Settings**

- Security Group: **CafeSG**

**7. Tags**

Add tag:

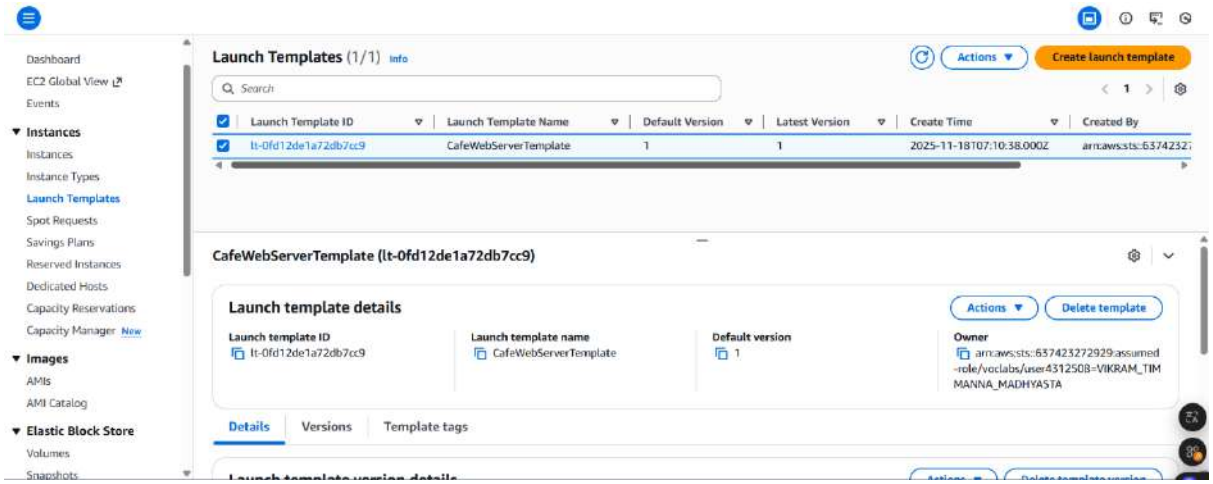
- **Key:** Name
- **Value:** webserver
- **Resource:** Instances

**8. IAM Role**

- Under Advanced details:
    - **IAM Instance Profile:** CafeRole

**9. Create**

Click **Create launch template**.




## TASK 3 — Create Application Load Balancer

### 1. Open Load Balancer Console

EC2 → Load Balancers → **Create Load Balancer** → Application Load Balancer

### 2. ALB Configuration

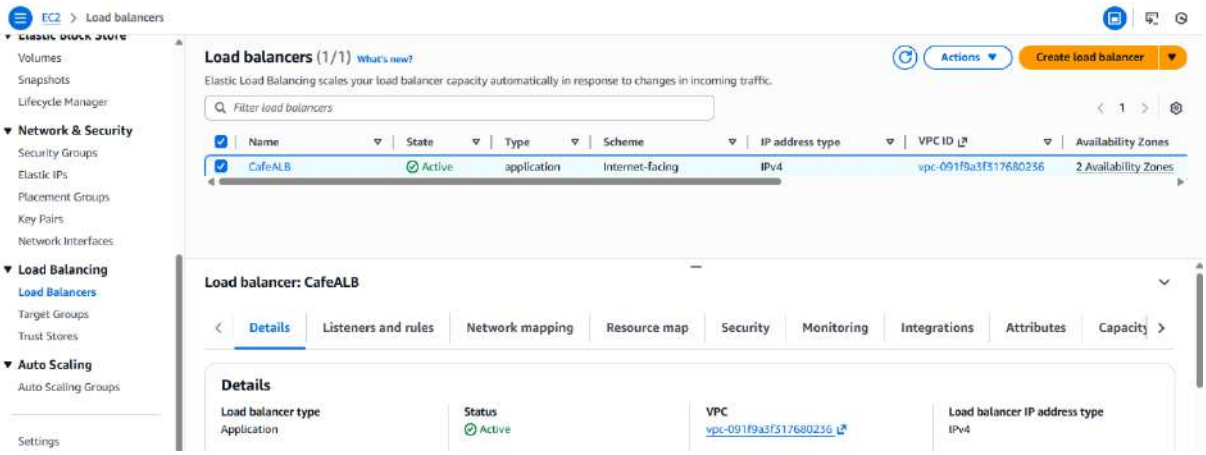
- Name: CafeALB
- Scheme: **Internet-facing**
- IP type: IPv4
- VPC: *lab VPC*
- Subnets:
  Public Subnet 1
  Public Subnet 2
- Select security group

### 3. Create Target Group

- Target type: **Instances**
- Name: CafeTargetGroup
- Protocol: **HTTP**
- Health check path: /cafe
- Click **Create target group**

### 4.Create

**Create Load Balancer**




## TASK 4 — Create Auto Scaling Group

**1. Open ASG Console**

EC2 → **Auto Scaling Groups** → **Create Auto Scaling group**

**2. Basic Details**

- ASG name: CafeWebServerASG
- Launch template: CafeWebServerTemplate

**3. Network**

- VPC: *your lab VPC*
- Subnets (select both):
  Private Subnet 1
  Private Subnet 2

**4. Group Size**

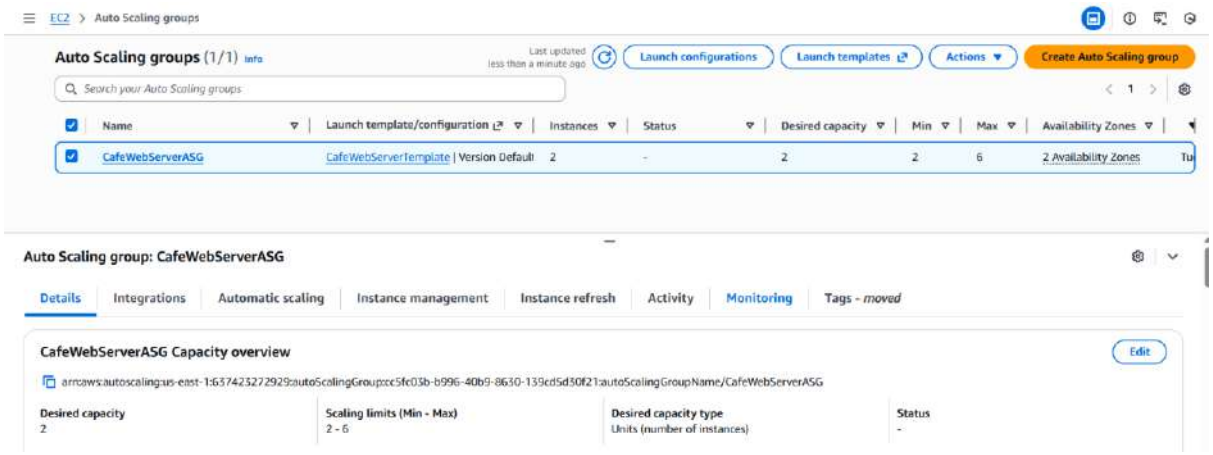
- **Desired:** 2
- **Min:** 2
- **Max:** 6

**5. Scaling Policy**

- Select **Target tracking scaling policy**
    - Metric: **Average CPU Utilization**
    - Target: **25%**
    - Instance warmup: **60 sec**

**6. Create ASG**

Click **Create Auto Scaling group**.




## Task 5 — Testing and Validation

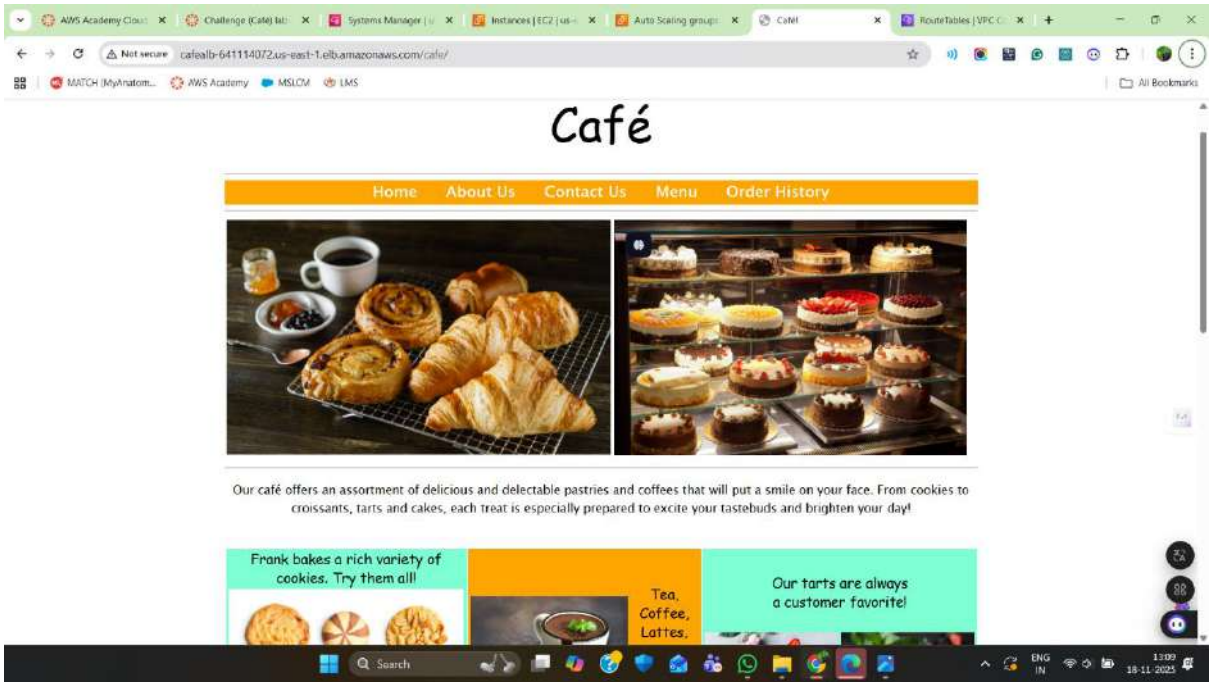
### 5.1 Application Reachability Test

The ALB DNS name was used to verify that the café application was accessible:

http://cafealb-641114072.us-east-1.elb.amazonaws.com/cafe/

This confirmed:

- Instances were healthy
- Target group was functioning
- ALB routing was operational



## 5.2 Auto Scaling Stress Test

To validate autoscaling behavior, a stress test was performed using **AWS Systems Manager Session Manager**.

**Commands to Run:**

sudo amazon-linux-extras install epel

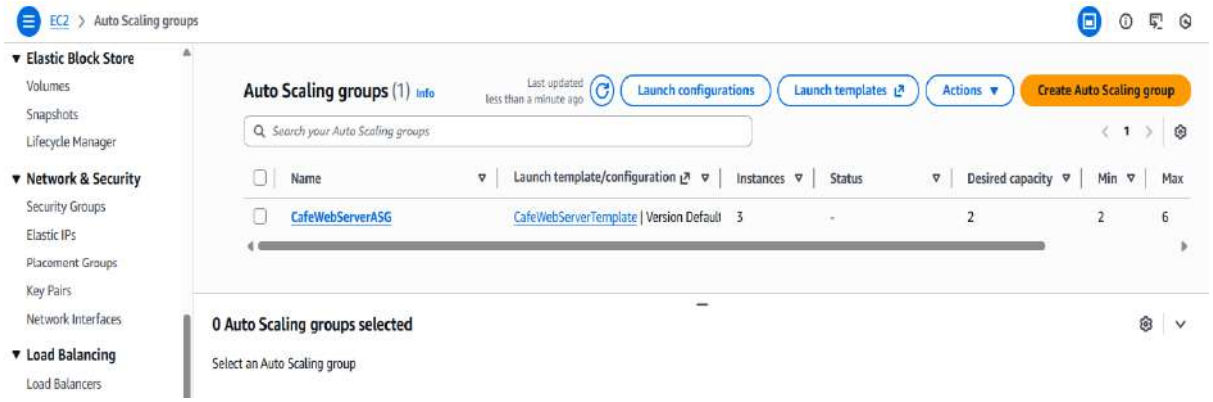
sudo yum install stress -y

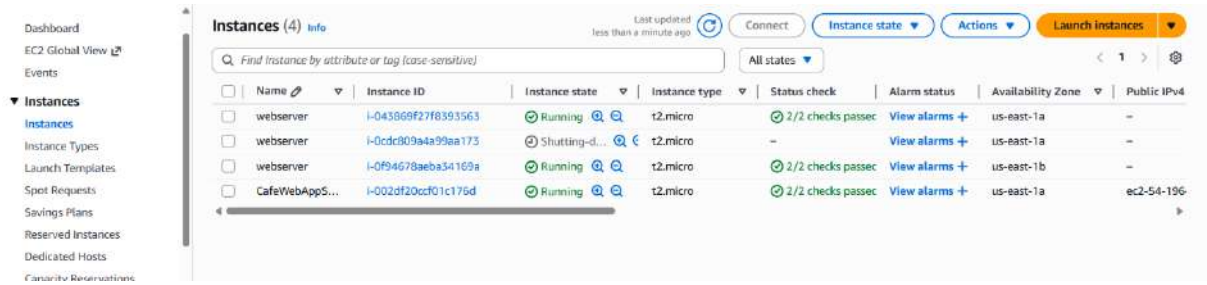
stress --cpu 1 --timeout 600

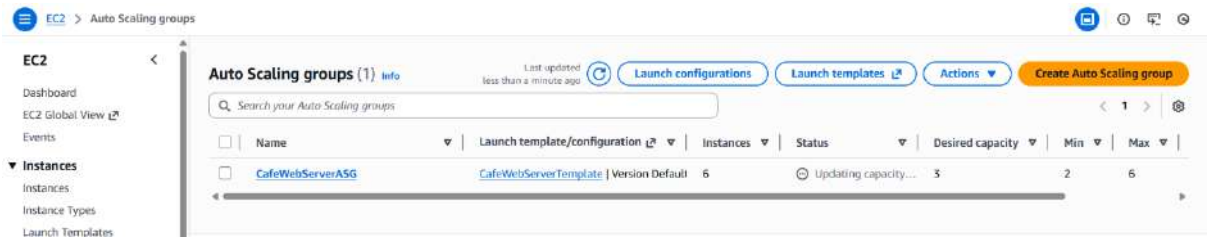
This generated artificial CPU load.
The ASG detected CPU >25% and initiated **scale-out**, launching additional EC2 instances.

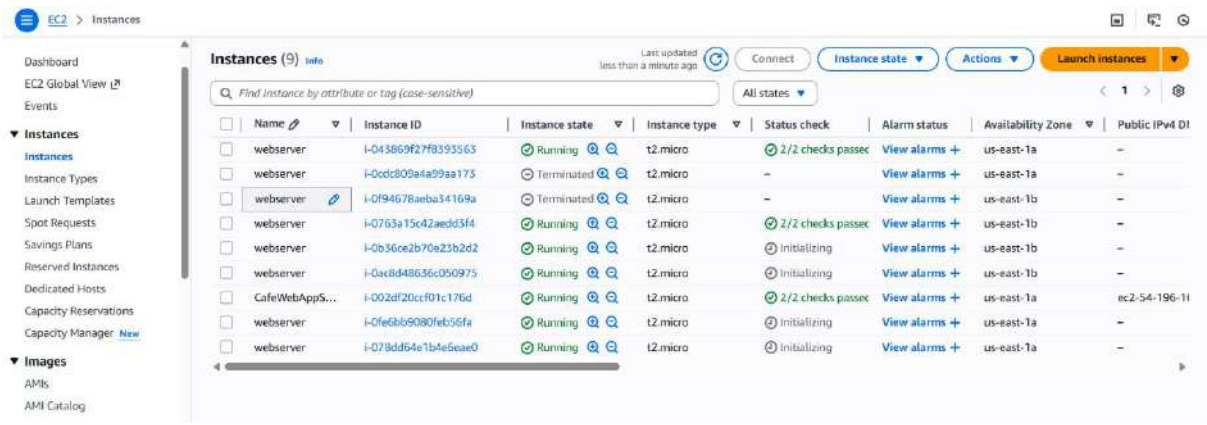
Monitoring the ASG dashboard confirmed:

- Scaling policies were applied correctly
- New instances launched in both private subnets
- Load gradually balanced across instances



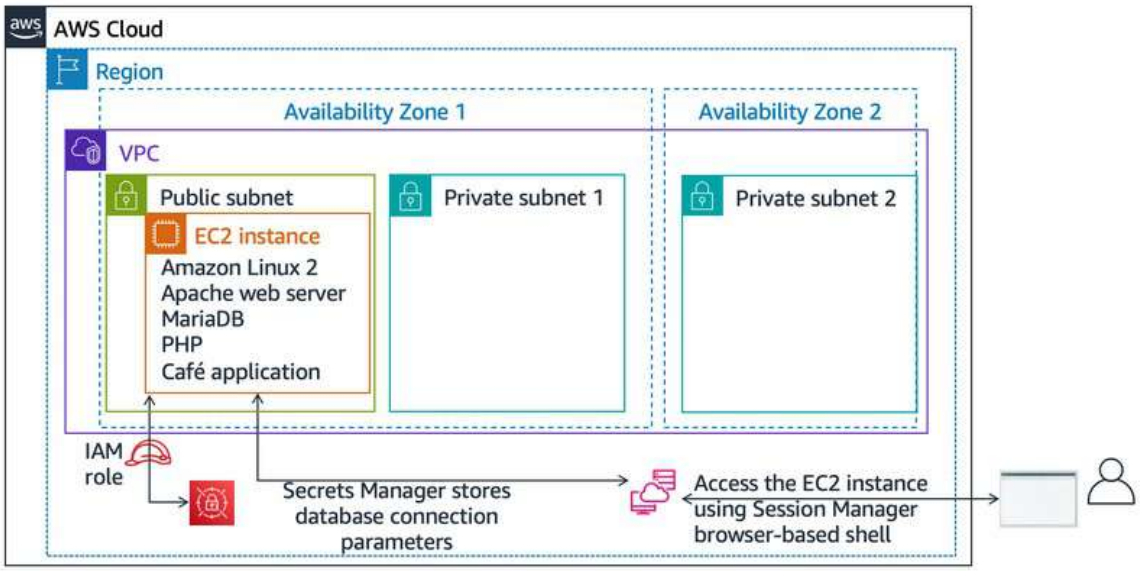




## Conclusion

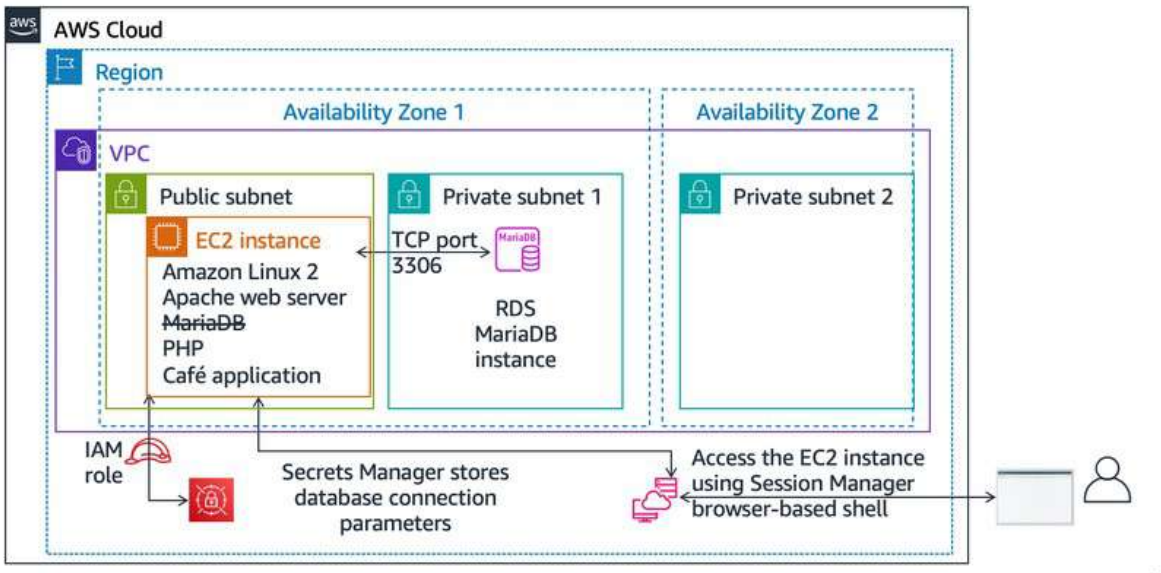
This lab successfully demonstrated the deployment of a highly available, fault tolerant, and scalable web application architecture using core AWS services. A multi-AZ design combined with NAT Gateways, private subnets, an ALB, and an Auto Scaling Group ensures robust performance and resiliency.

# CA, Mod4; Challenge (Cafe) lab: Migrating a Database to Amazon RDS

Initially :

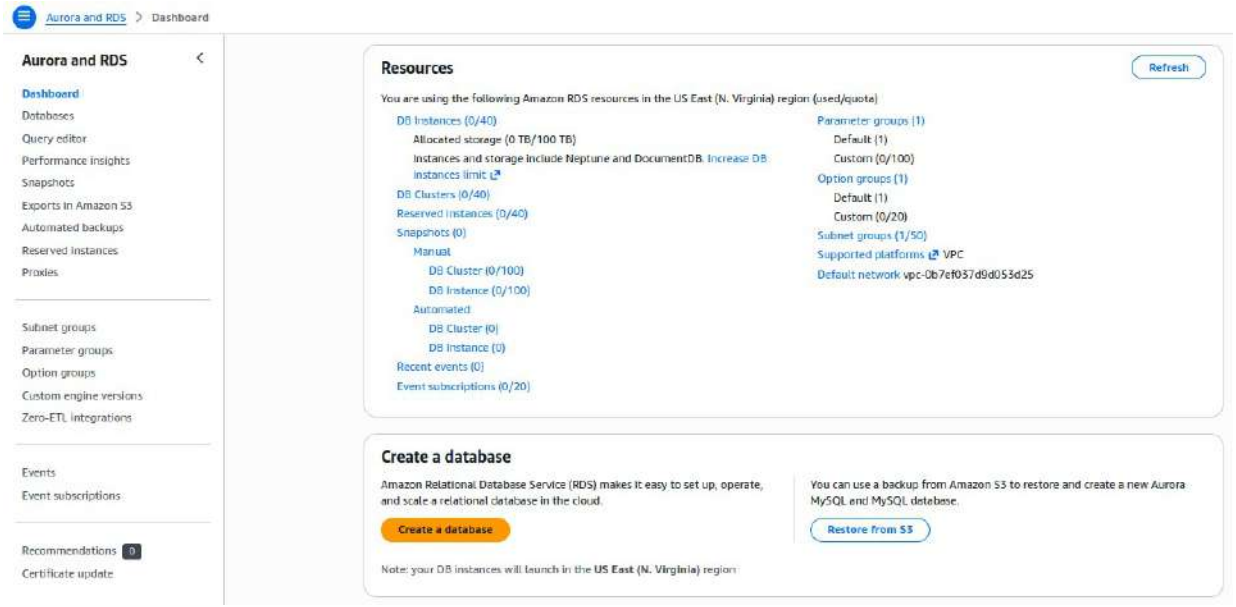



Final output :




**Step 1: Creating an RDS Instance**

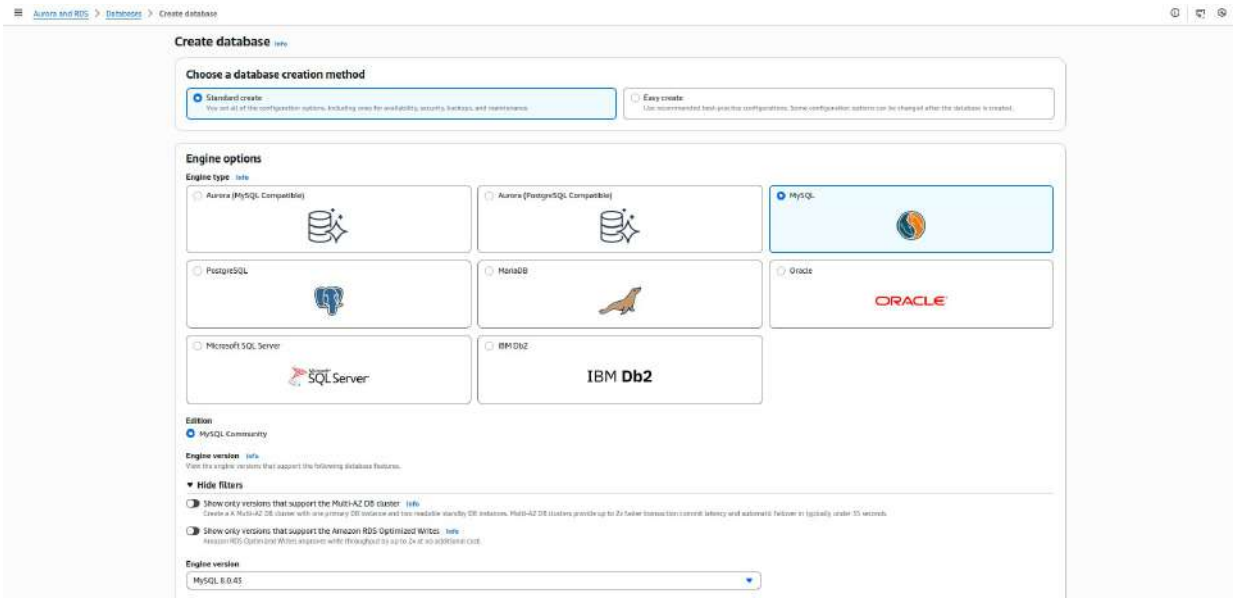
1. Go to the **Amazon RDS console → Create database**.

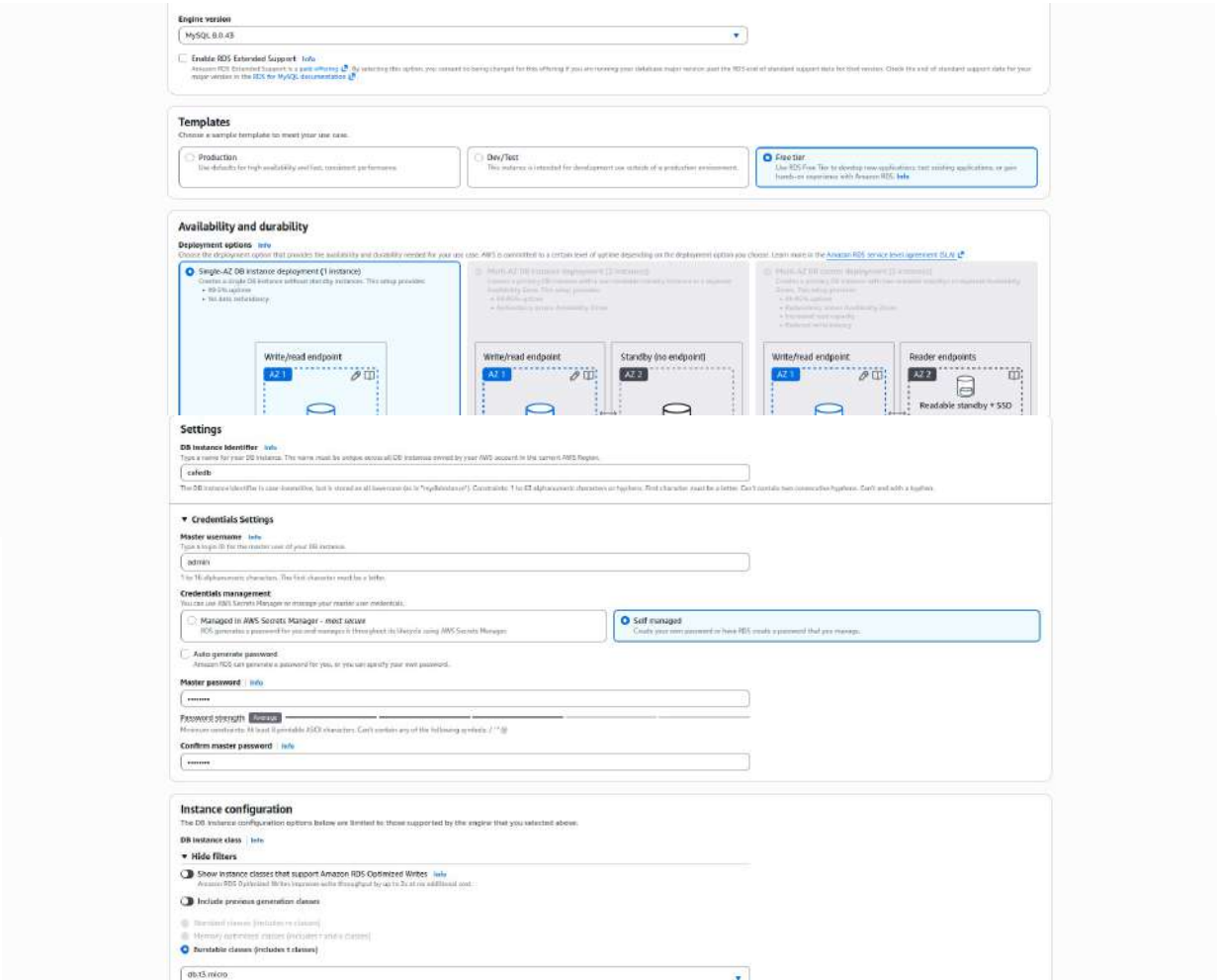


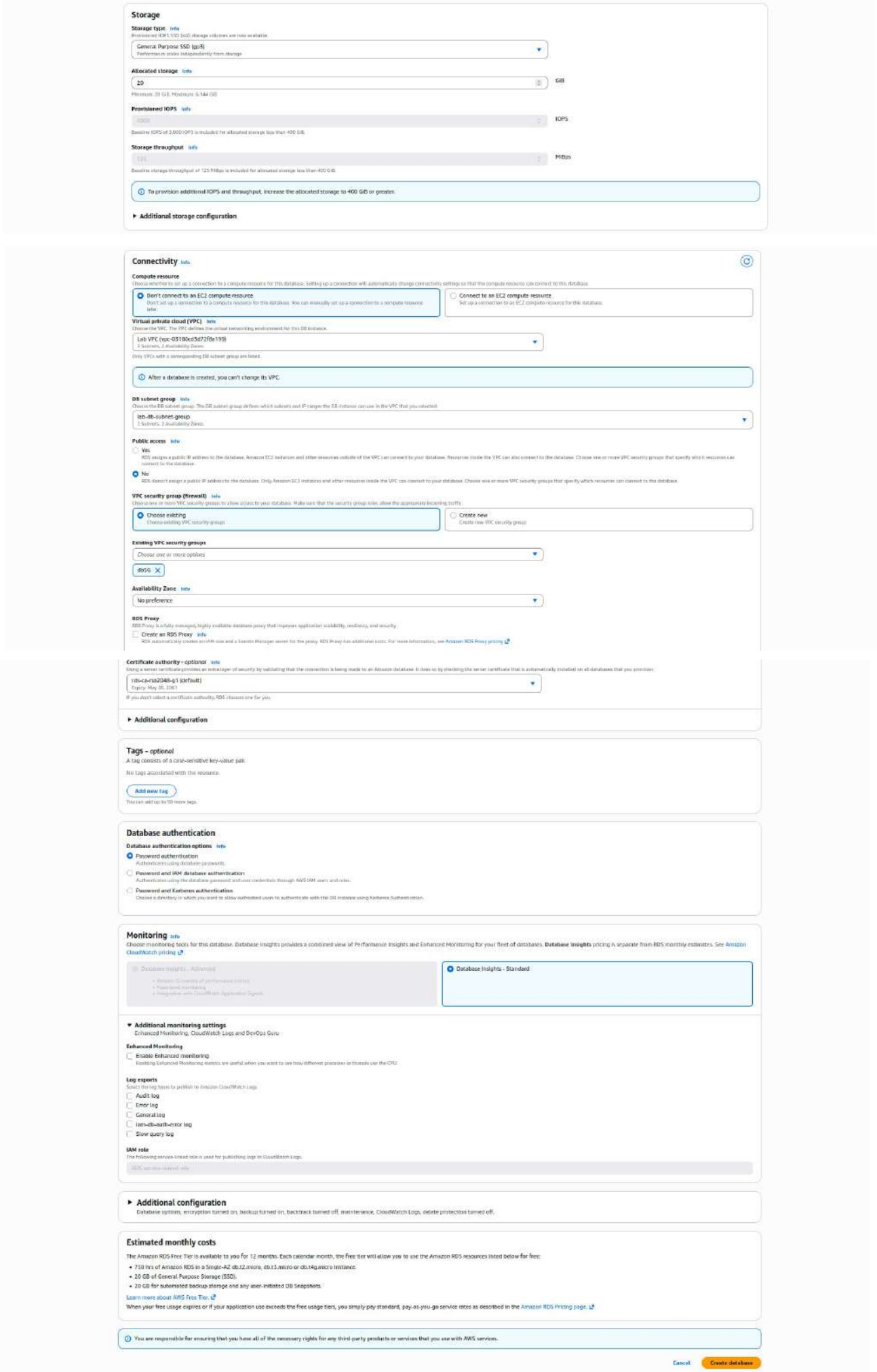

2. Choose a database creation method: **Standard create**
3. Engine type: **MariaDB**.
4. Template: **Free tier**.
5. Availability & durability: **Single-AZ DB instance deployment (1 instance)**
6. Settings:
    - DB instance identifier: **`Cafedb`**
    - Master username: **`admin`**
    - Credentials management: **`self managed`**
    - Master password: **`Msis1234`**
7. Instance configuration:
    - DB instance class: **db.t3.micro** (Burstable classes)
8. Storage:
    - Storage type: **General Purpose SSD (gp3)**

- Allocated storage: **`20 GiB`**

9. Connectivity:
    - Virtual private cloud (VPC): select **Lab VPC**
    - DB subnet group: **lab-db-subnet-group**
    - Existing VPC security groups: **dbSG** (uncheck default)
10. Monitoring: Uncheck **Enable Enhanced monitoring**
11. Click **Create database**.




Storage
Storage type Info
General Purpose SSD (gp3)
Allocated storage Info
20
GiB
Provisioned IOPS Info
IOPS
Storage throughput Info
MiBps
To provision additional IOPS and throughput, increase the allocated storage to 400 GiB or greater.
Additional storage configuration
Connectivity Info
Compute resource
Don't connect to an EC2 compute resource
Connect to an EC2 compute resource
Virtual private cloud (VPC) Info
Lab VPC (vpc-05180cd5d72f0e199)
After a database is created, you can't change its VPC.
DB subnet group Info
lab-db-subnet-group
Public access Info
Yes
No
VPC security group (firewall) Info
Choose existing
Create new
Existing VPC security groups
Choose one or more options
dbSG
Availability Zone Info
No preference
RDS Proxy
Create an RDS Proxy Info
Certificate authority - optional Info
rds-ca-rsa2048-g1 (default)
Additional configuration
Tags - optional
A tag consists of a case-sensitive key-value pair.
No tags associated with the resource.
Add new tag
You can add up to 50 more tags.
Database authentication
Database authentication options Info
Password authentication
Password and IAM database authentication
Password and Kerberos authentication
Monitoring Info
Choose monitoring tools for this database. Database Insights provides a combined view of Performance Insights and Enhanced Monitoring for your fleet of databases. Database Insights pricing is separate from RDS monthly estimates. See Amazon CloudWatch pricing.
Database Insights - Advanced
Database Insights - Standard
Additional monitoring settings
Enhanced Monitoring, CloudWatch Logs and DevOps Guru
Enhanced Monitoring
Enable Enhanced monitoring
Log exports
Audit log
Error log
General log
iam-db-auth-error log
Slow query log
IAM role
Additional configuration
Database options, encryption turned on, backup turned on, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.
Estimated monthly costs
The Amazon RDS Free Tier is available to you for 12 months. Each calendar month, the free tier will allow you to use the Amazon RDS resources listed below for free:
750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro instance.
20 GB of General Purpose Storage (SSD).
20 GB for automated backup storage and any user-initiated DB Snapshots.
Learn more about AWS Free Tier.
When your free usage expires or if your application use exceeds the free usage tiers, you simply pay standard, pay-as-you-go service rates as described in the Amazon RDS Pricing page.
You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.
Cancel
Create database

## Step 3: In Session Manager

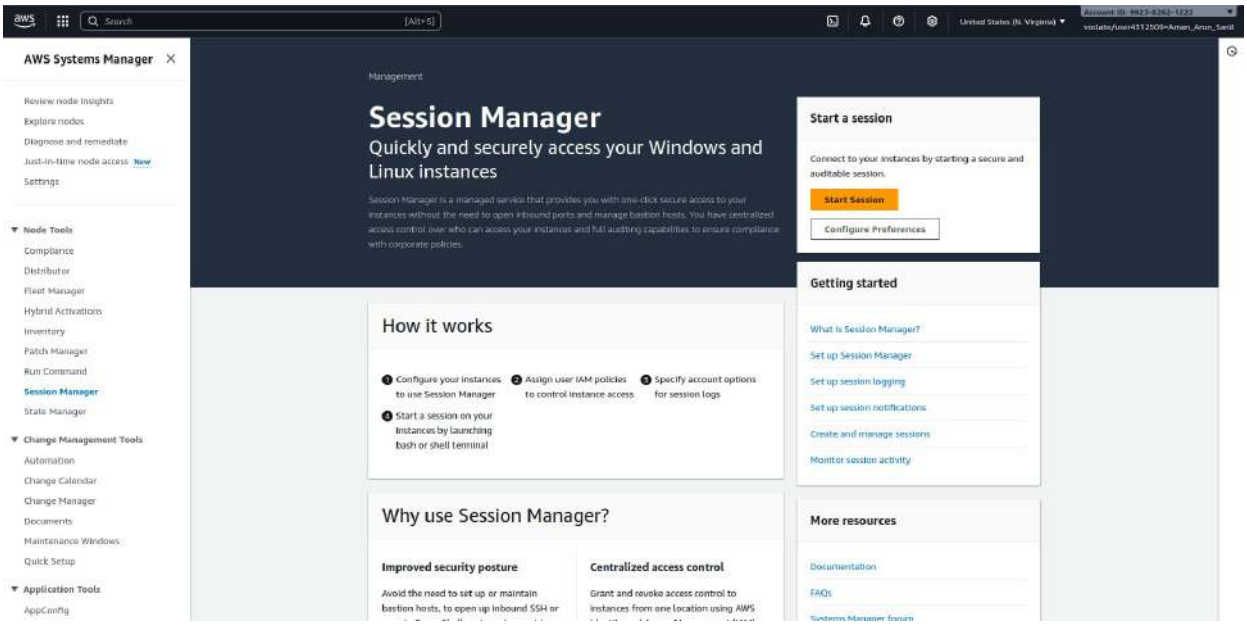
1. **Create session** → select **cafe server** → **Start session**.
2. Run the following commands:

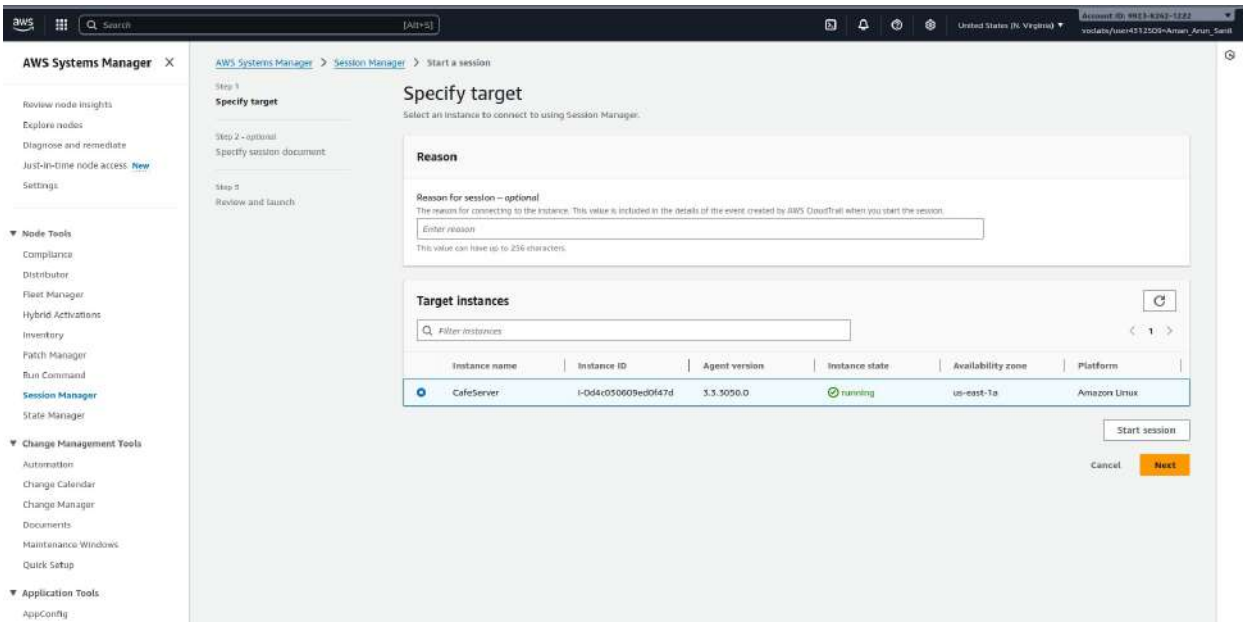
   sudo su

   su ec2-user

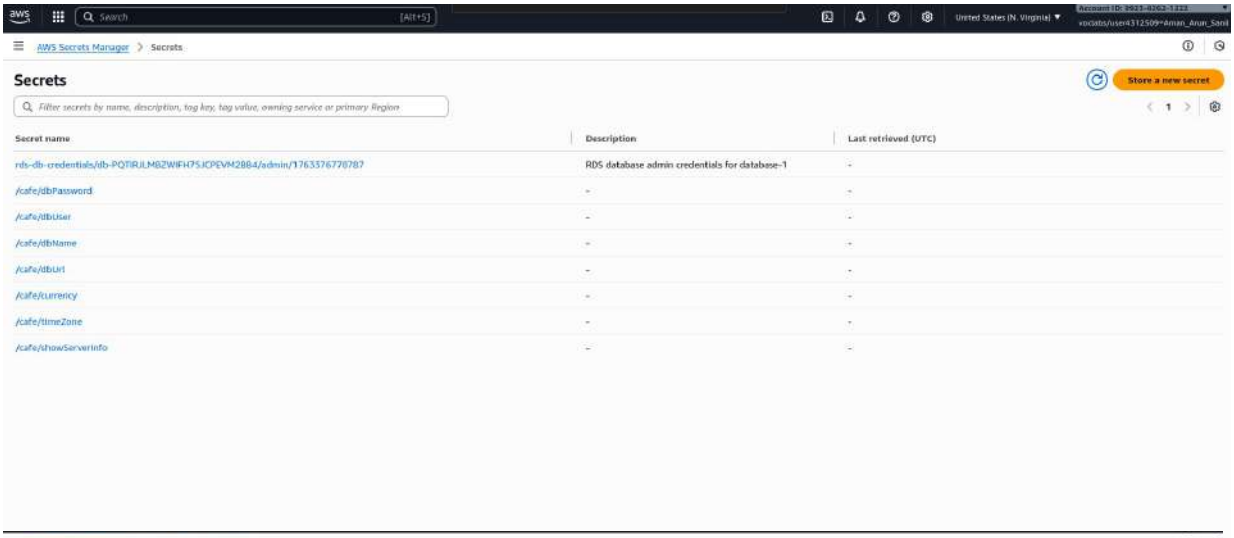
   cd /var/www/html/cafe/

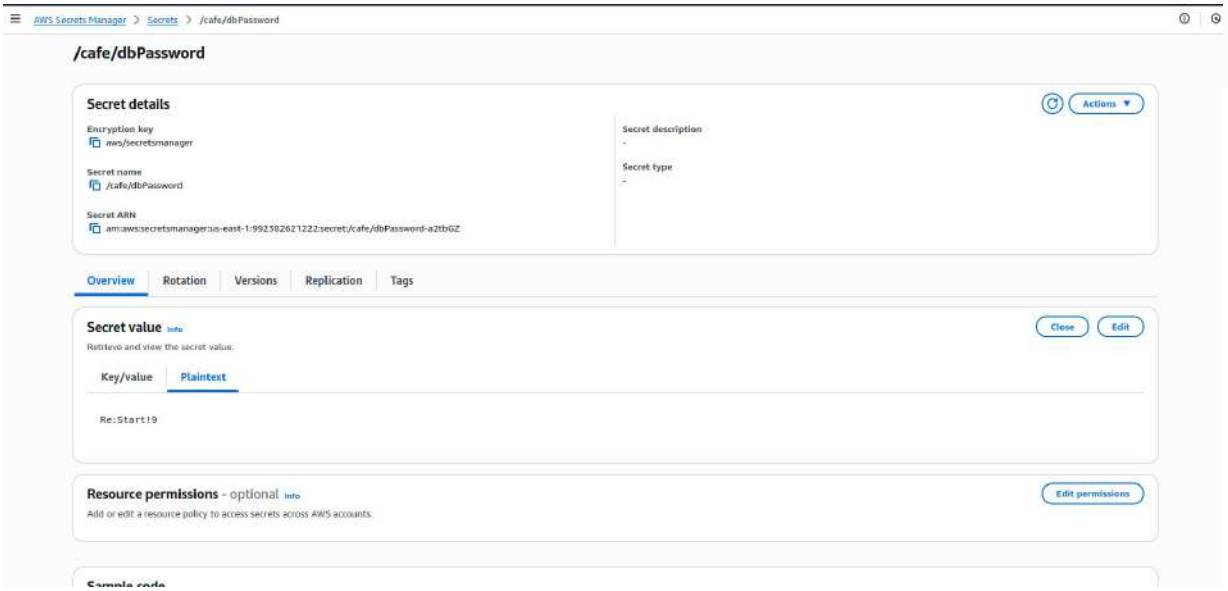
   sudo mysql -u root -p  //paste the secretes password from Secretes Manager



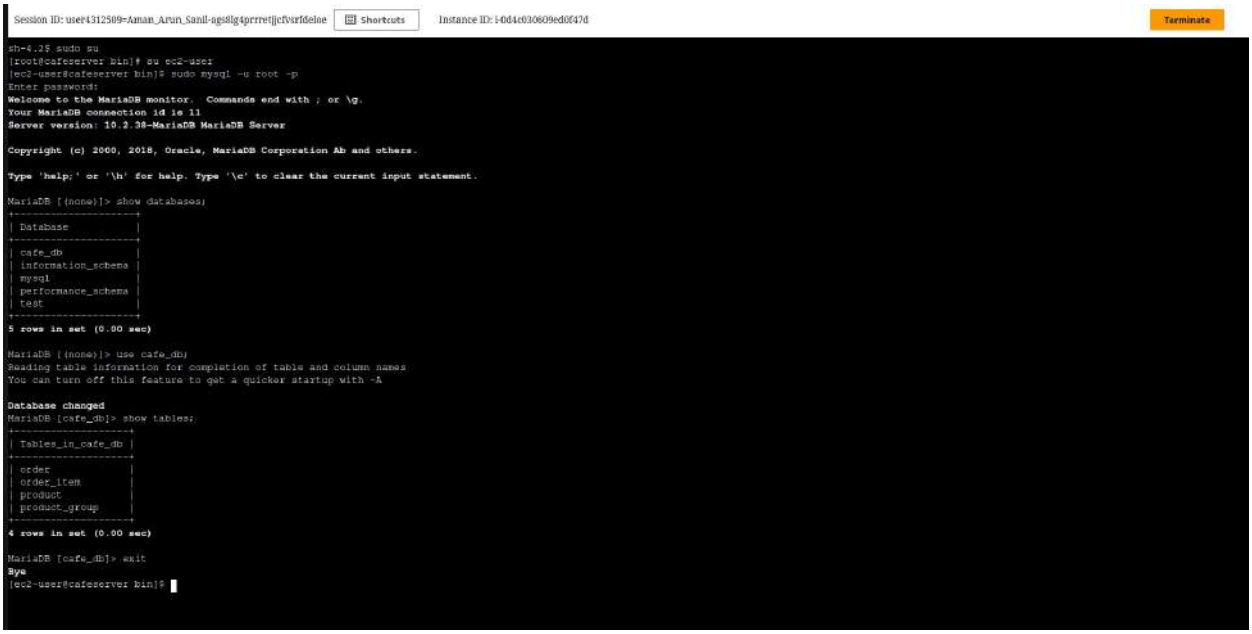

## To retrieve password from Secretes Manager :



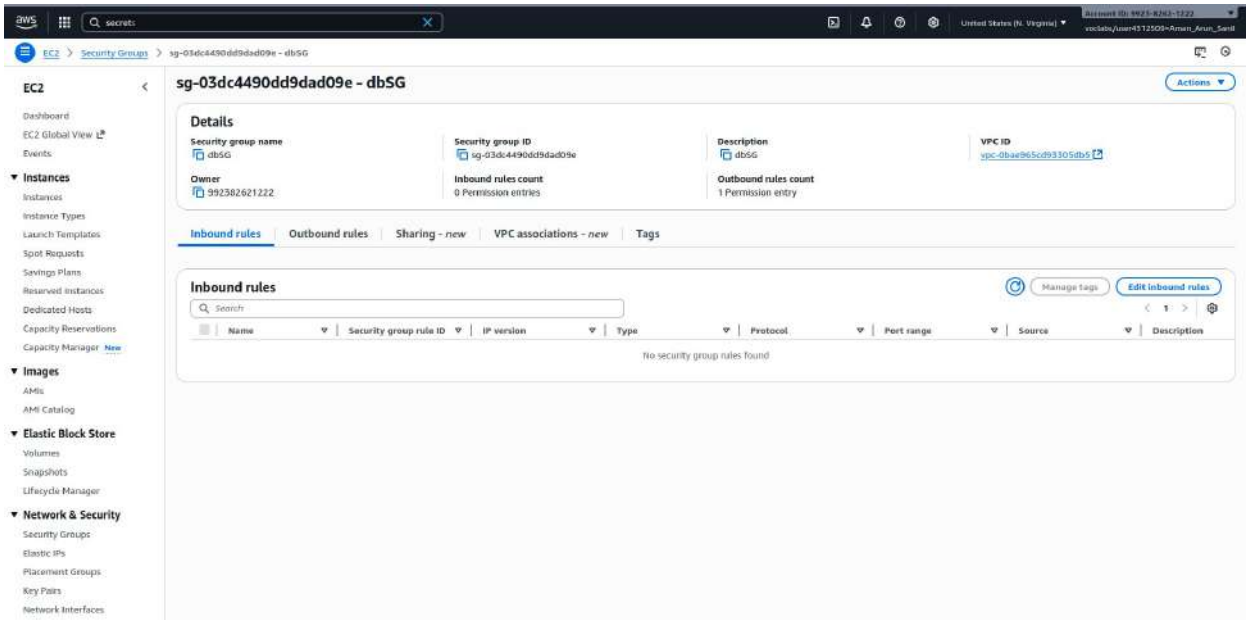



## In Session Manager after entering into mysql :

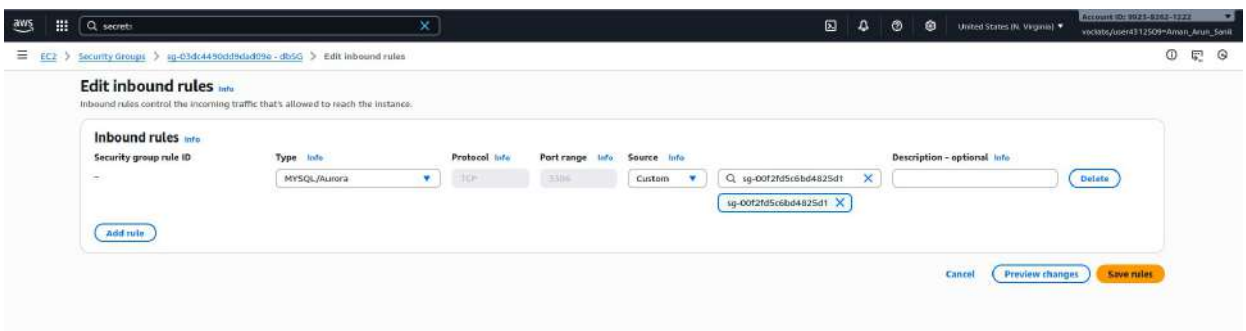
> show Databases;
> use cafe_db;
> show tables;

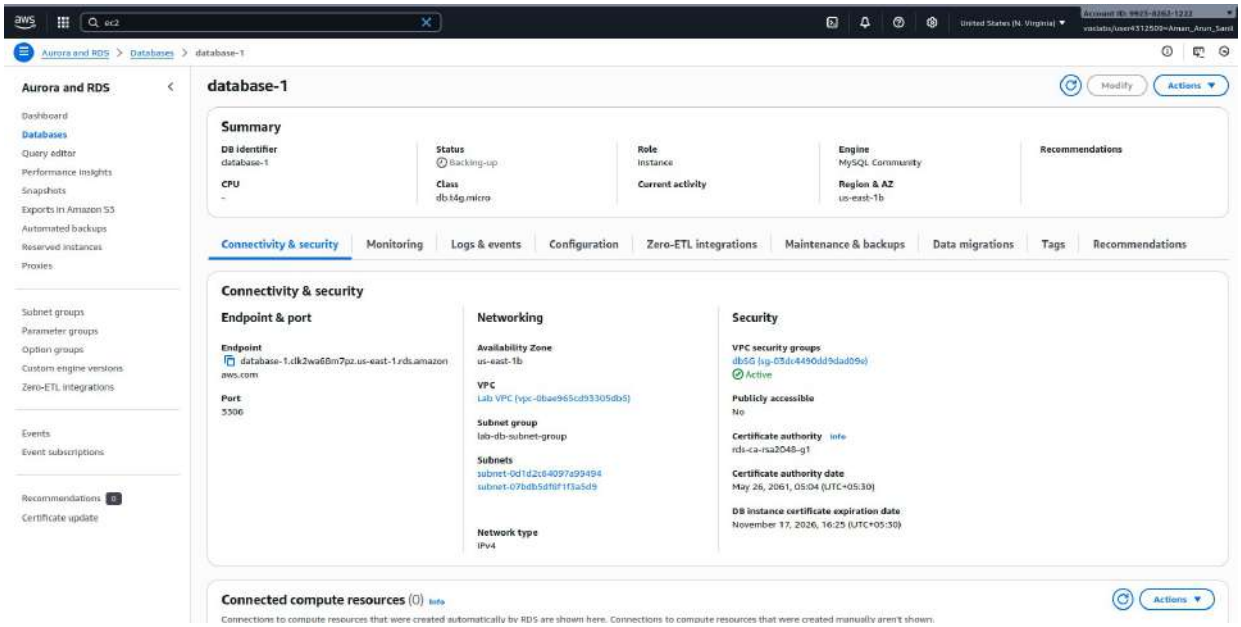



## In EC2, security groups > Edit inbound Rules of DB security group:




## select MYSQL/AURORA → CUSTOM → CAFESG

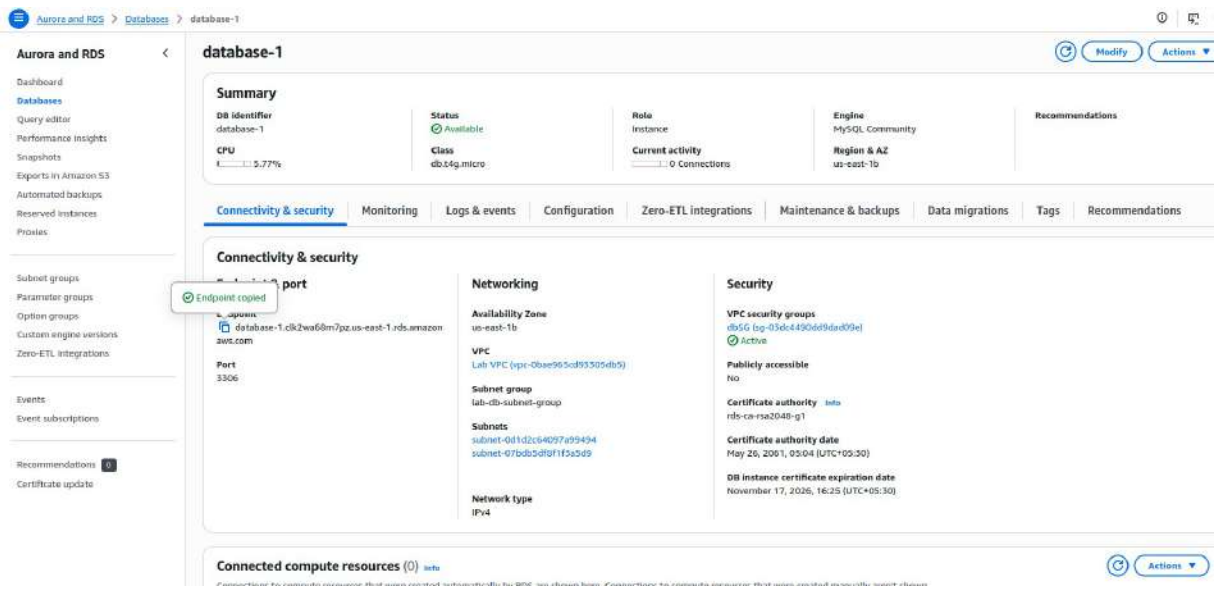


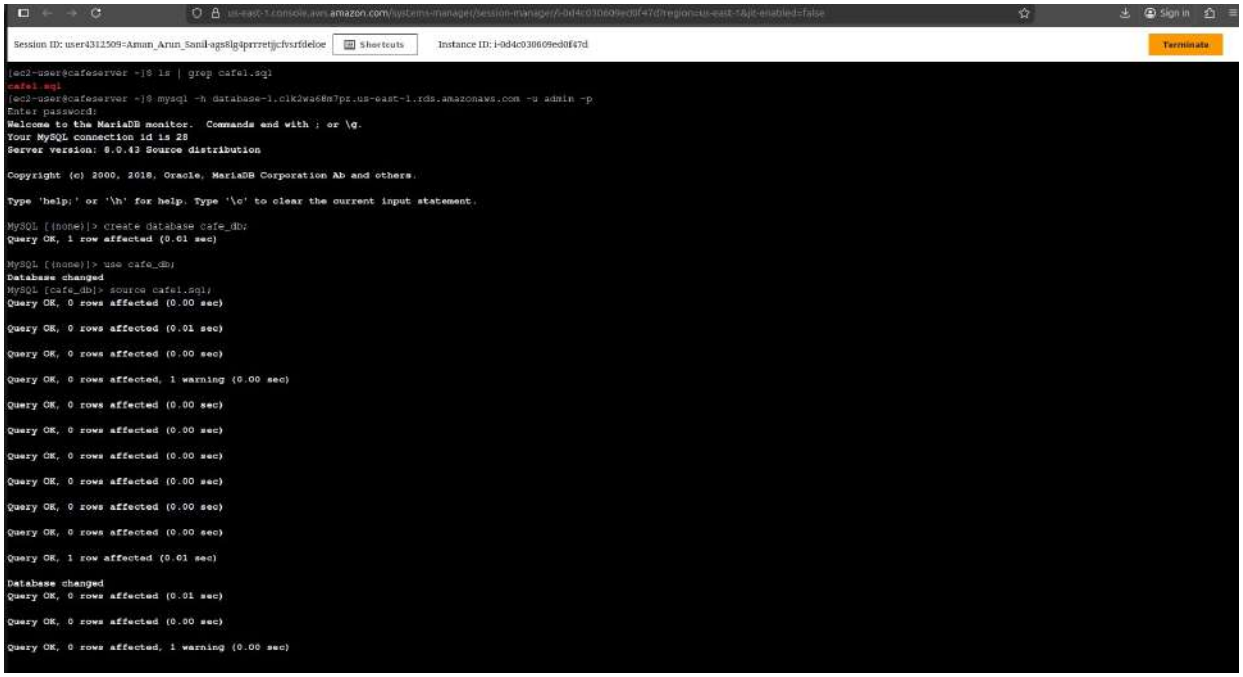




# In RDS, Copy the RDS Endpoint :


// here paste the rds endpoint and password as Msis1234





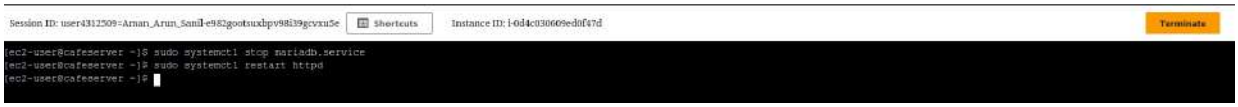


in mysql run the follwing commands;

> create database cafe_db;
> use cafe_db;
> show tables;
> source cafe1.sql;
> show tables;

> exit;

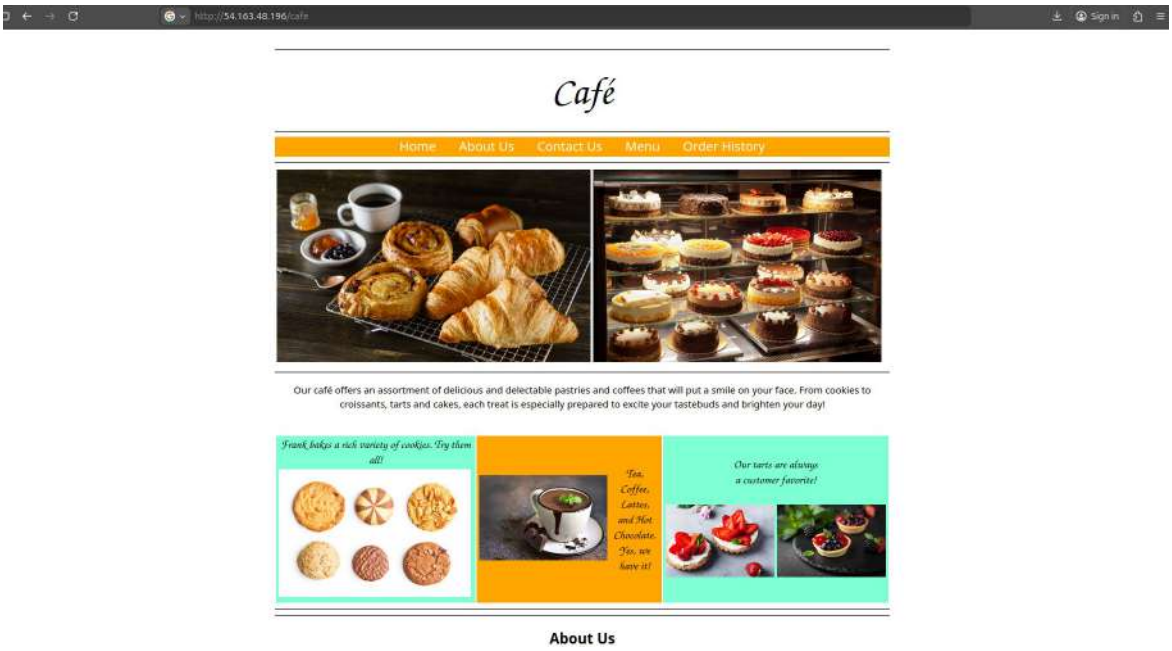
sudo systemctl stop mariadb.service
sudo systemctl status mariadb.service  // shows inactive, dead
sudo systemctl restart httpd




### Test the Cafe Application on the EC2 Instance

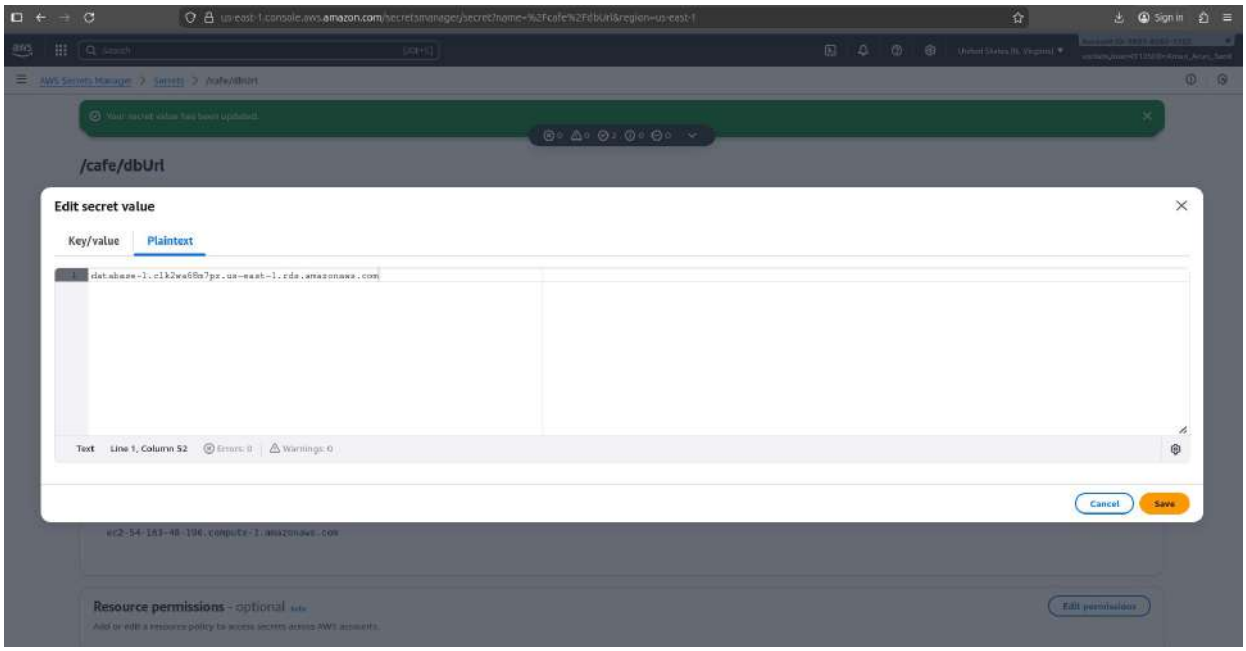
1. Go to **EC2 console → Instances → CafeServer**.
2. Copy the public address.

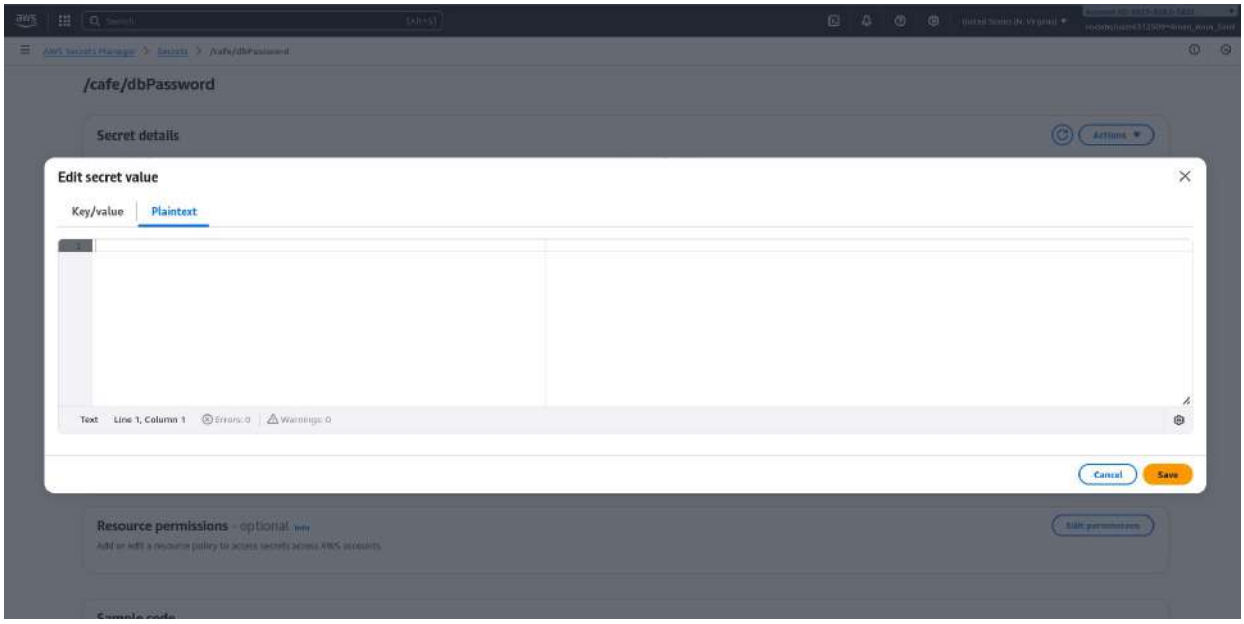
3. Access the application:
`http://<public-ip>/cafe`

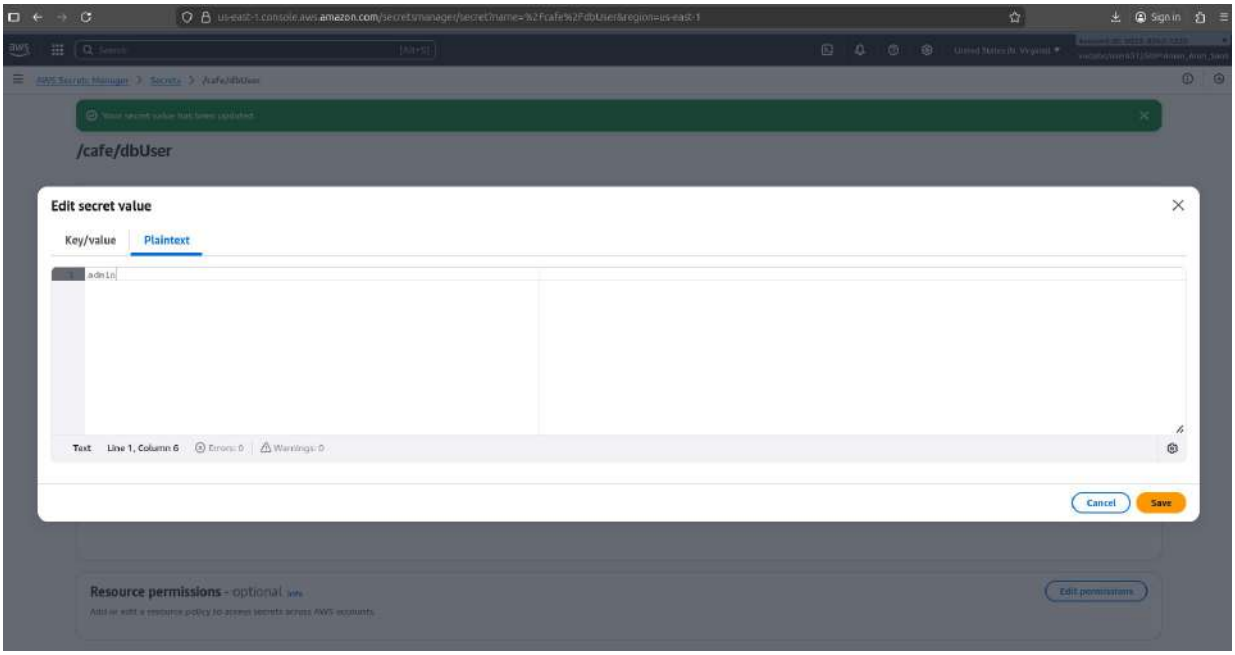



For the Menu and Order to work do the below changes in **Secrets Manager**

**In Secrets Manager change:**
dbUrl → paste Endpoint
dbPassword → Msis1234
dbUser → admin

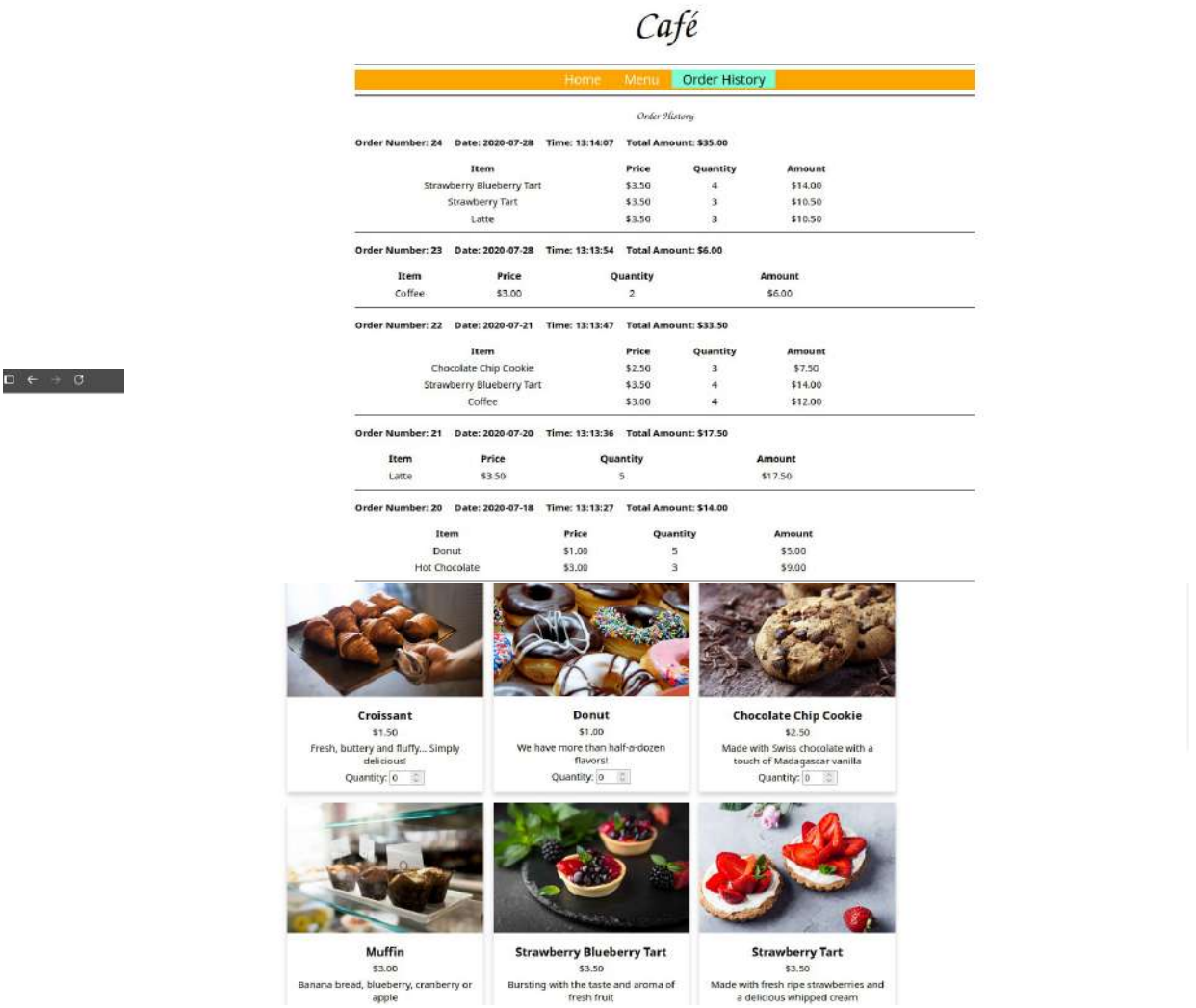



/cafe/dbPassword
Secret details
Actions
Edit secret value
Key/value
Plaintext
Text
Line 1, Column 1
Cancel
Save
Resource permissions - optional


/cafe/dbUser
Edit secret value
Key/value
Plaintext
Text
Line 1, Column 6
Cancel
Save
Resource permissions - optional
Edit permissions

http://<public-ip>/cafe

//check the Menu and Order up and running



# CFA LAB

## Challenge (Cafe) lab: Creating a Static Website for the Cafe

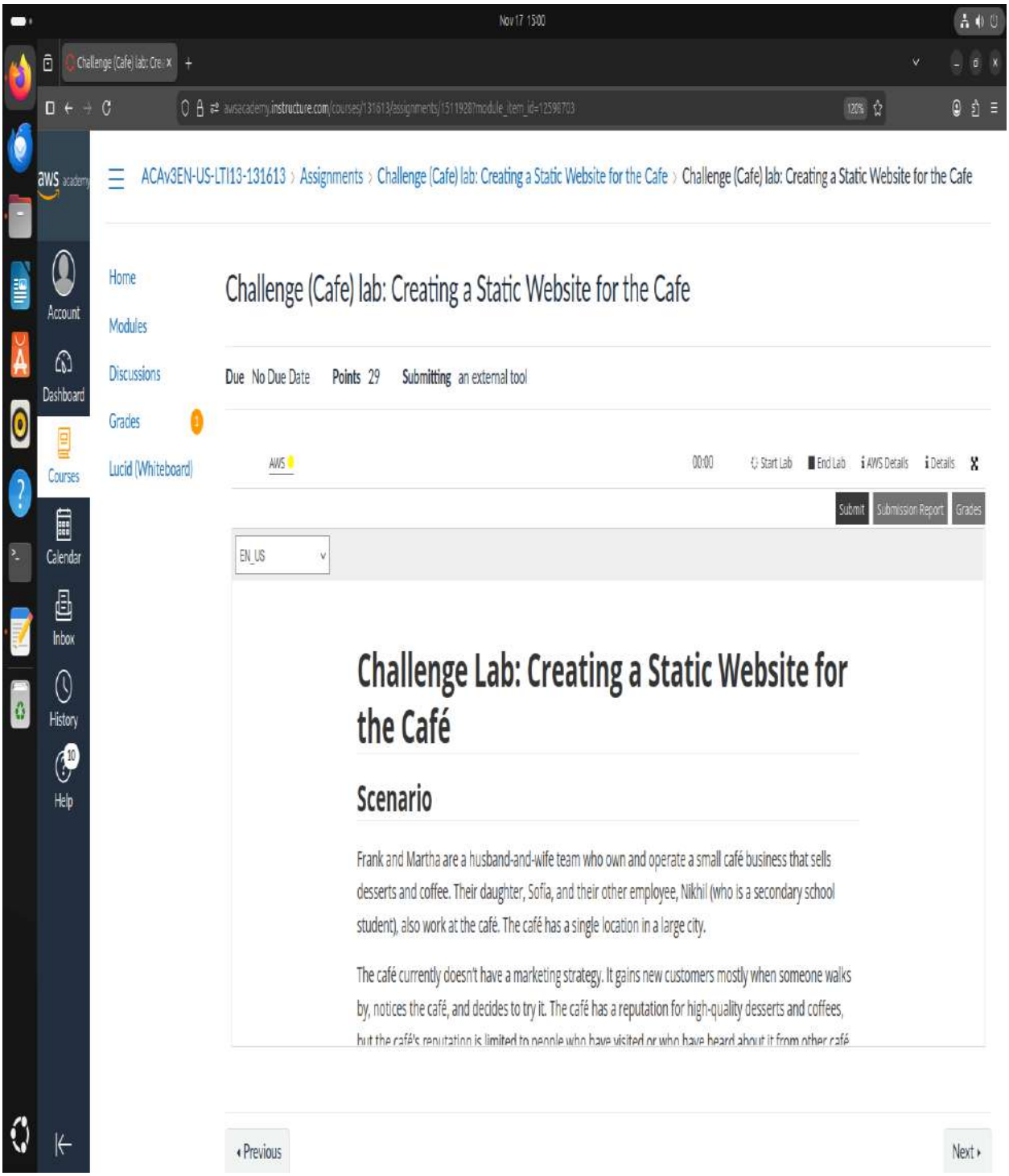
1. At the top of these instructions, choose **Start Lab**.
    a. The lab session starts.
    b. A timer displays at the top of the page and shows the time remaining in the session.

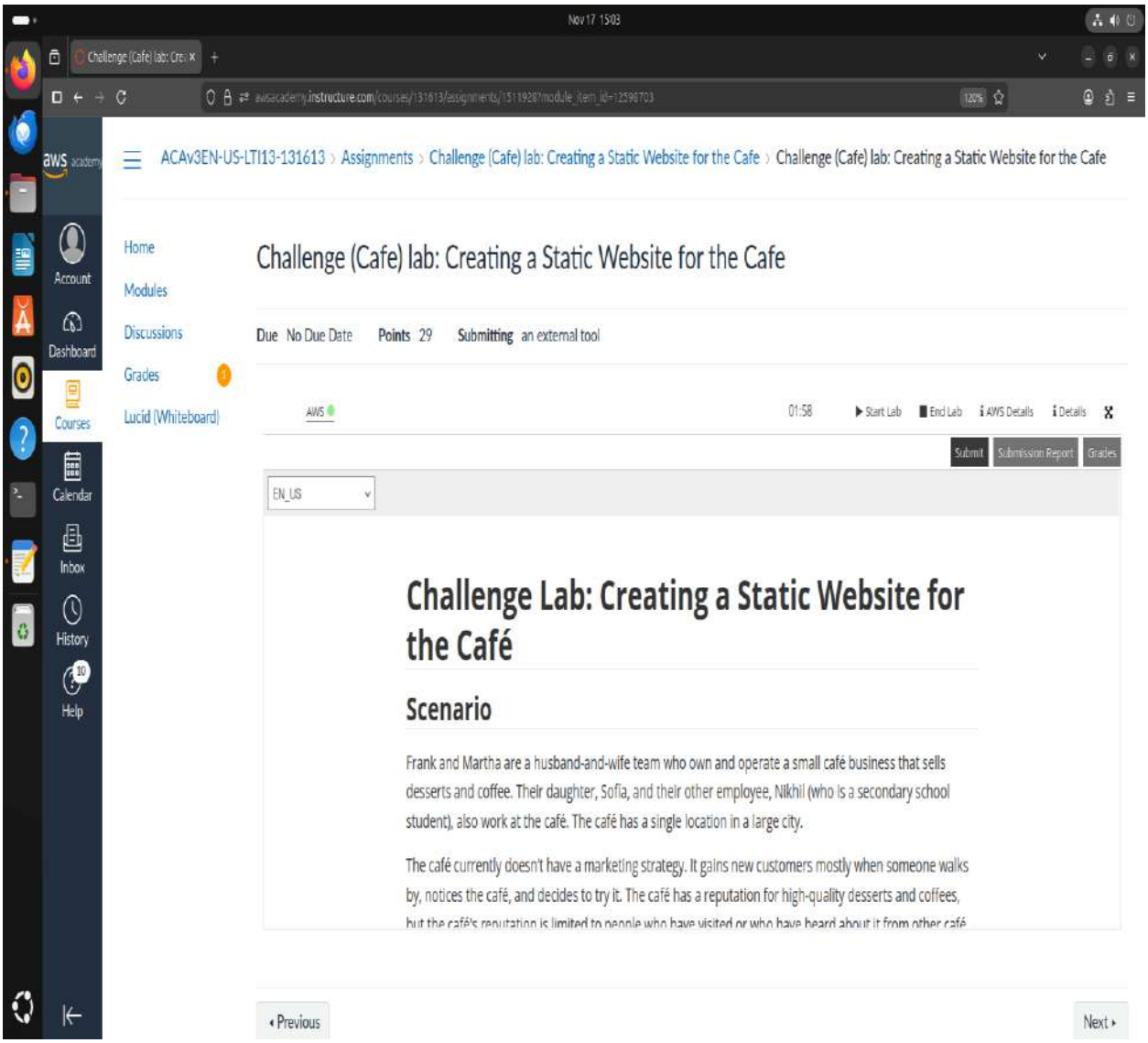
**Tip:** To refresh the session length at any time, choose **Start Lab** again before the timer reaches 0:00.

    c. Before you continue, wait until the circle icon to the right of the AWS link in the upper-left corner turns green. When the lab environment is ready, the **AWS Details** panel displays.
2. To connect to the AWS Management Console, choose the **AWS** link in the upper-left corner above the terminal window.
    a. A new browser tab opens and connects you to the console.

**Tip:** If a new browser tab does not open, a banner or icon is usually at the top of your browser with a message that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and then choose **Allow pop-ups**.

3. Arrange the AWS Management Console tab so that it displays alongside these instructions. Ideally, you have both browser tabs open at the same time so that you can follow the lab steps.


Nov 17 15:00
Challenge (Cafe) lab: Cre
awsacademy.instructure.com/courses/131613/assignments/1511928?module_item_id=12598703
ACAv3EN-US-LTI13-131613 › Assignments › Challenge (Cafe) lab: Creating a Static Website for the Cafe › Challenge (Cafe) lab: Creating a Static Website for the Cafe
Account
Dashboard
Courses
Calendar
Inbox
History
Help
Home
Modules
Discussions
Grades
Lucid (Whiteboard)
Challenge (Cafe) lab: Creating a Static Website for the Cafe
Due No Due Date
Points 29
Submitting an external tool
AWS
00:00
Start Lab
End Lab
AWS Details
Details
Submit
Submission Report
Grades
EN_US
Challenge Lab: Creating a Static Website for the Café
Scenario
Frank and Martha are a husband-and-wife team who own and operate a small café business that sells desserts and coffee. Their daughter, Sofia, and their other employee, Nikhil (who is a secondary school student), also work at the café. The café has a single location in a large city.
The café currently doesn't have a marketing strategy. It gains new customers mostly when someone walks by, notices the café, and decides to try it. The café has a reputation for high-quality desserts and coffees,
‹ Previous
Next ›




## Task 2: Creating an S3 bucket to host your static website

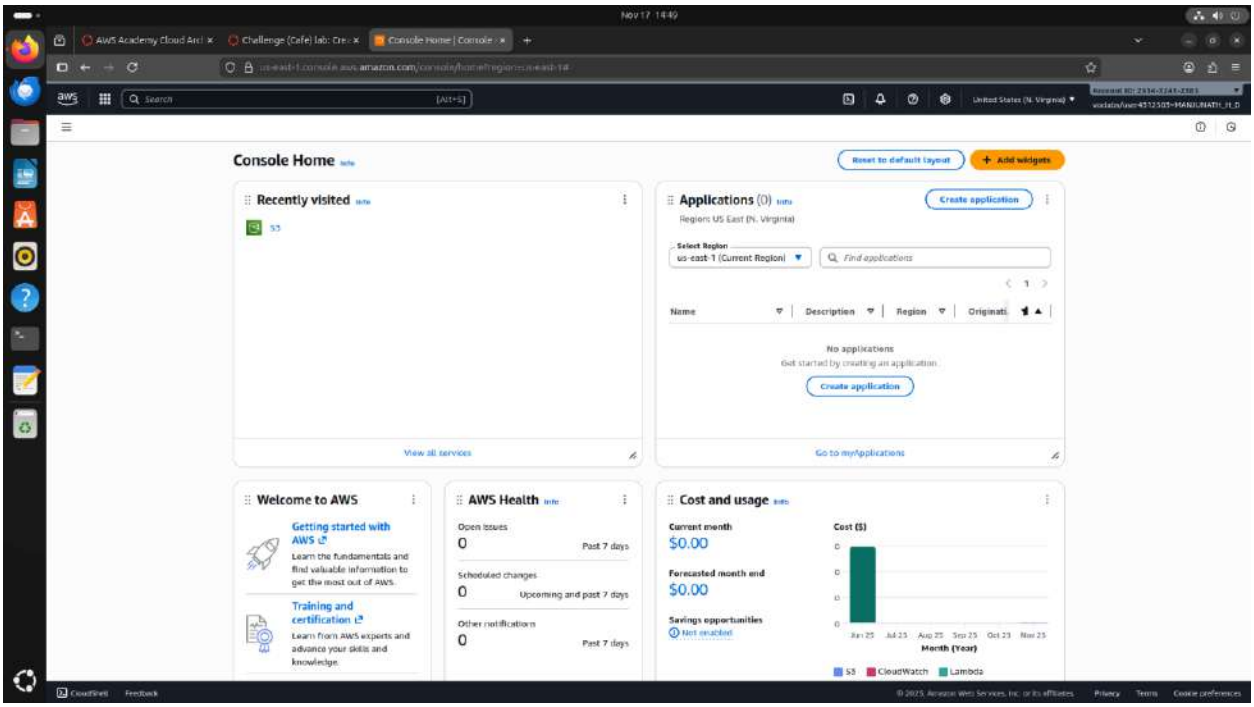
In this task, you create an S3 bucket and configure it to host your static website.

6. Open the Amazon S3 console.
7. Create a bucket in the **US East (N. Virginia) us-east-1** AWS Region to host your static website.

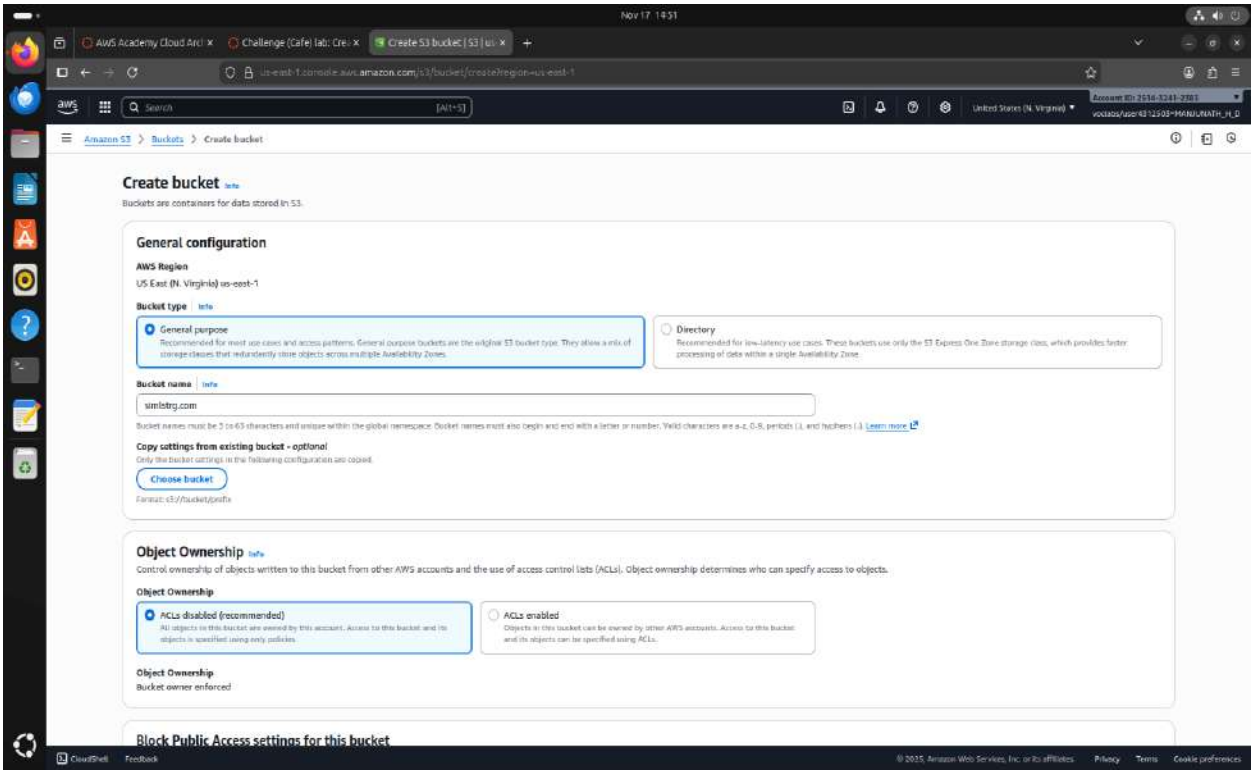
**Tip**: You must clear **Block *all* public access** and enable **ACLs**.

8. Enable static website hosting on your bucket.

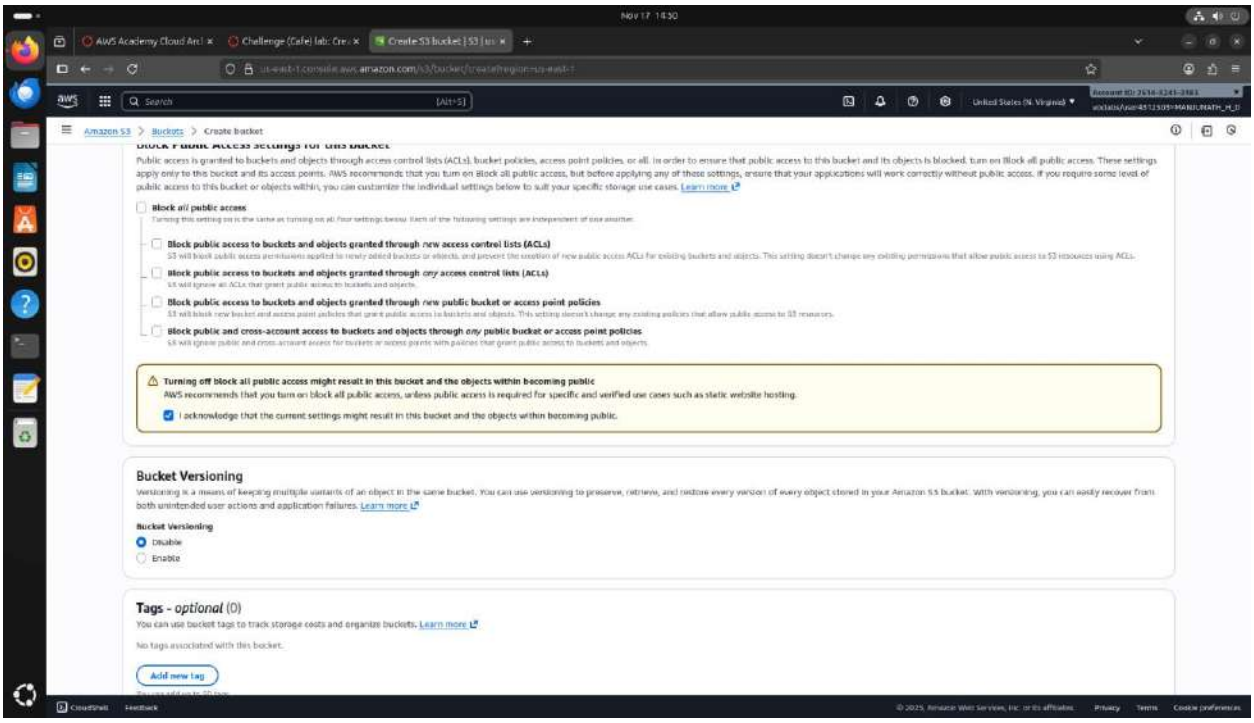
**Tip**: You use the index.html file for your index document.



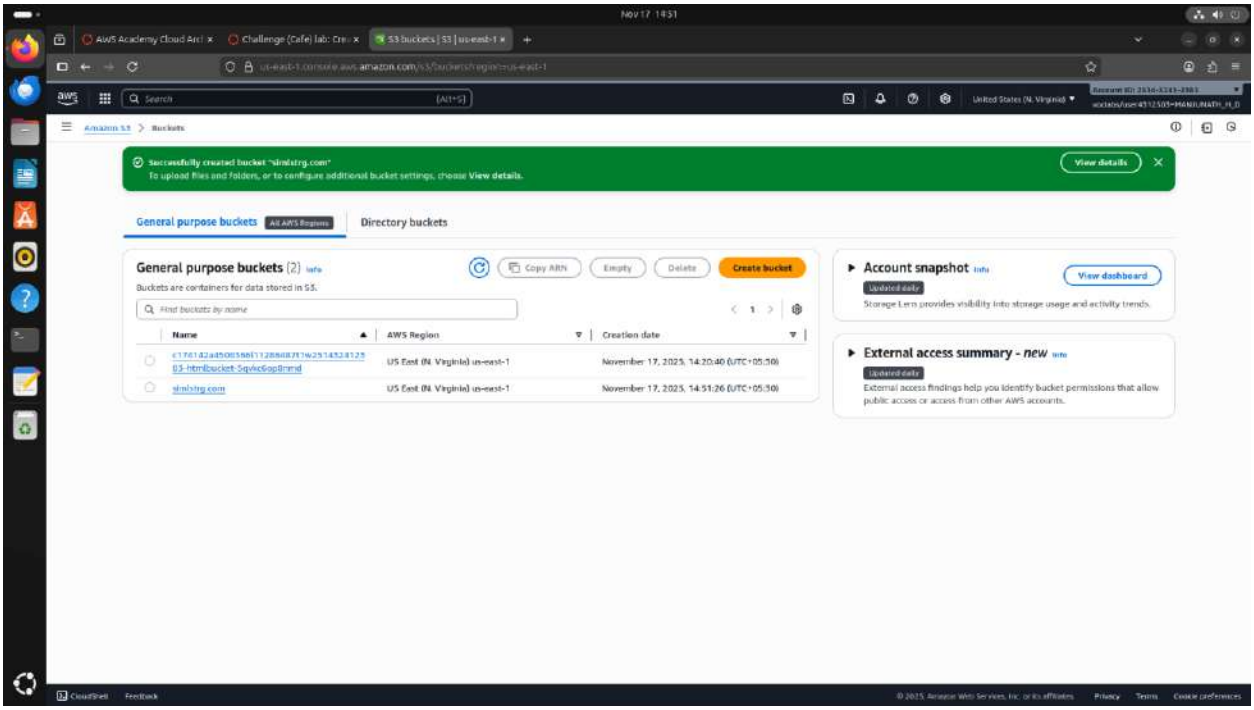

STEP 3 : Select the S3 bucket in the AWS Management Console.



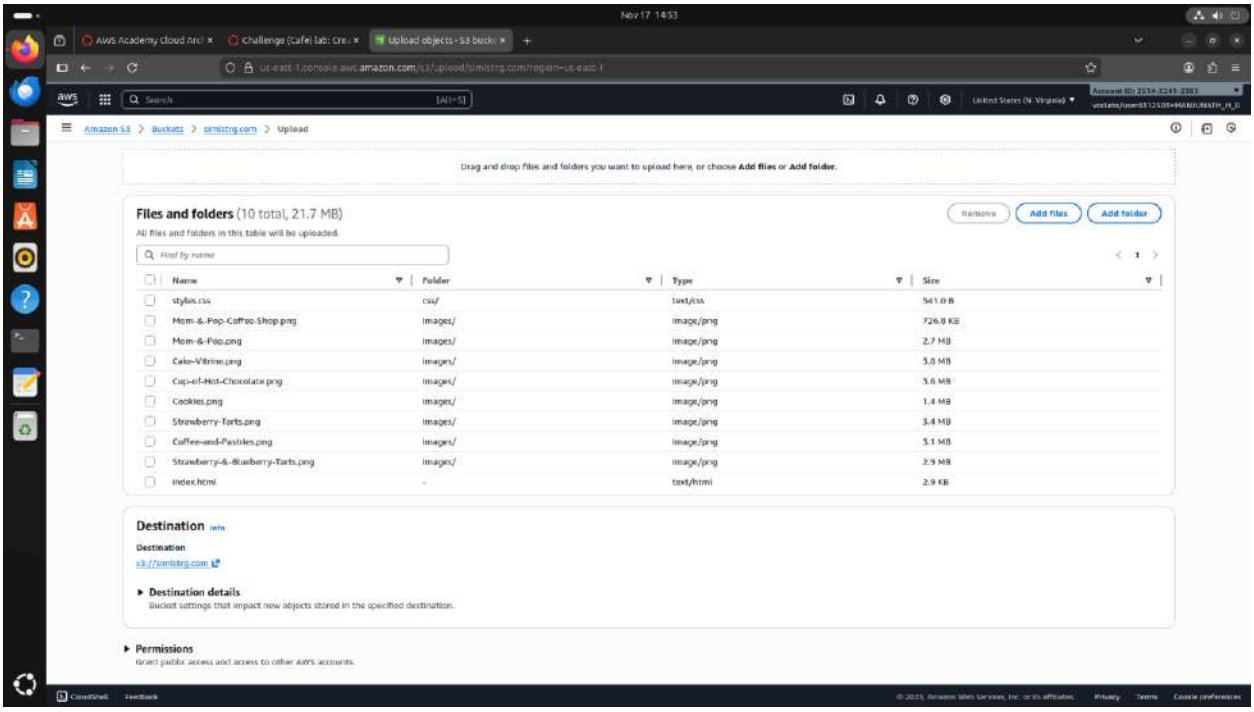

STEP 4 : I created an S3 bucket called *simlstrg*, and its ACL is not enabled yet.



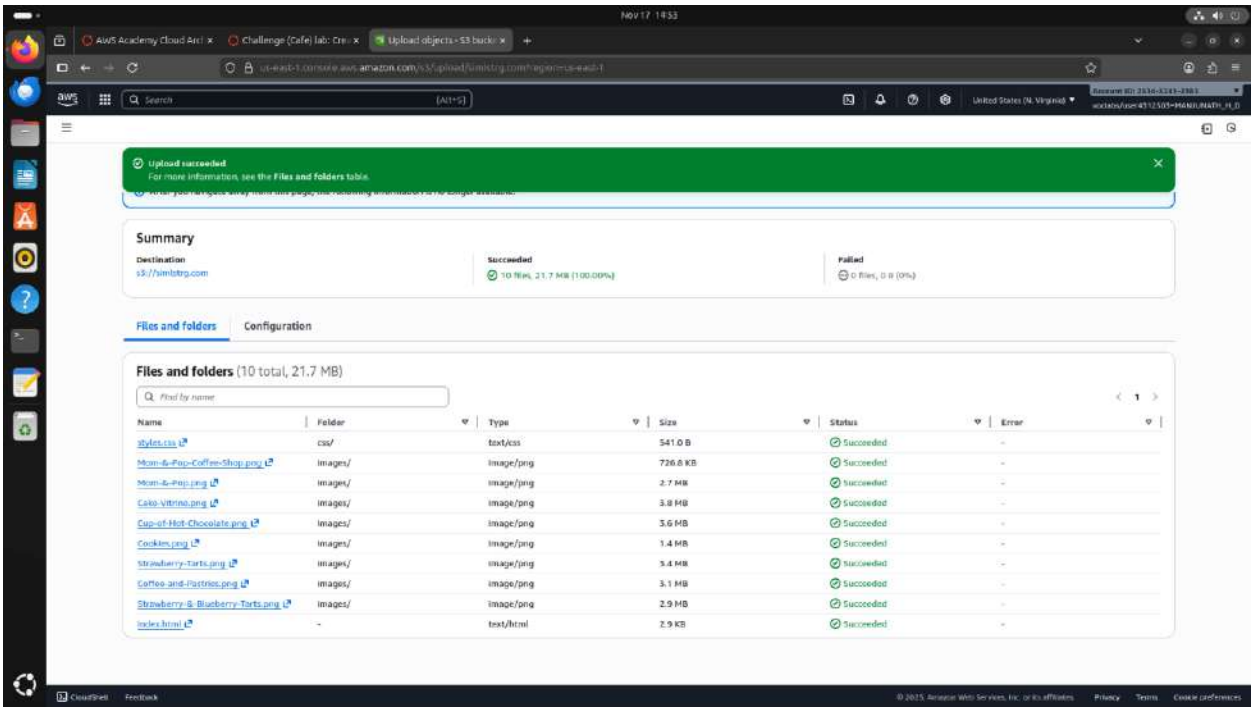

Step 5: Uncheck Block all public access and check the box for 'I acknowledge that I want to block all public access.'



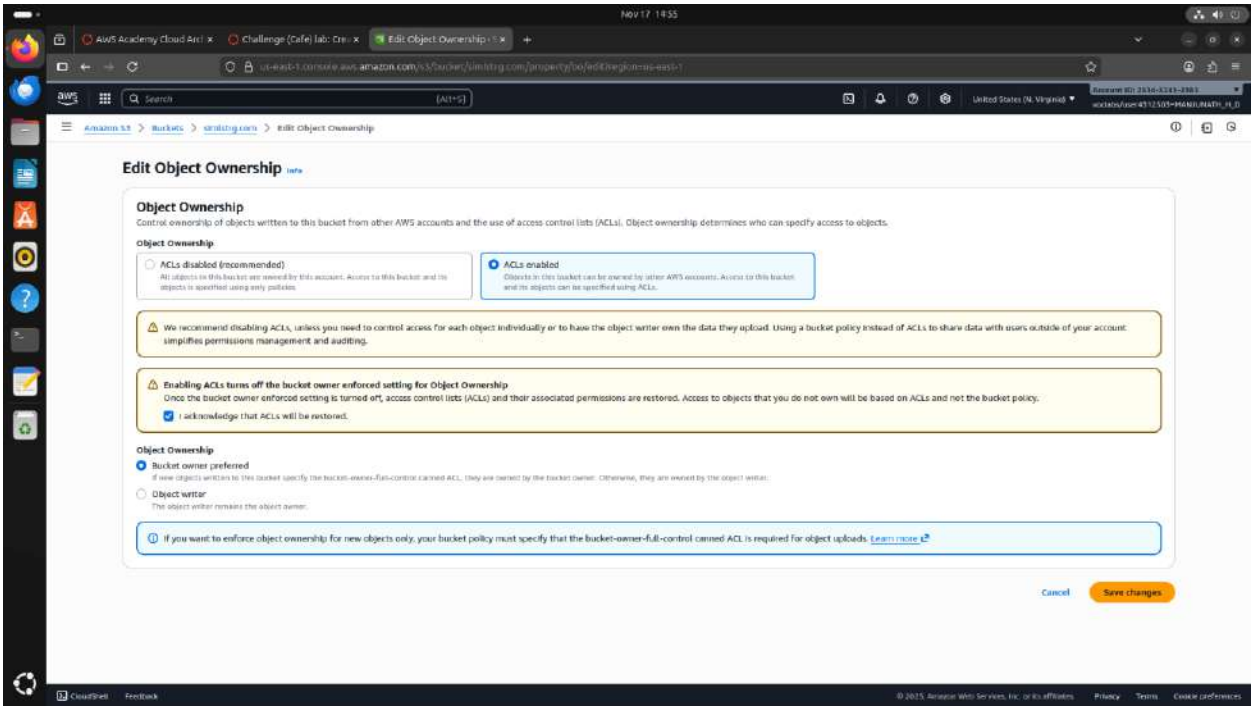

> Now the S3 bucket with the name simlstrg has sucessfully created.



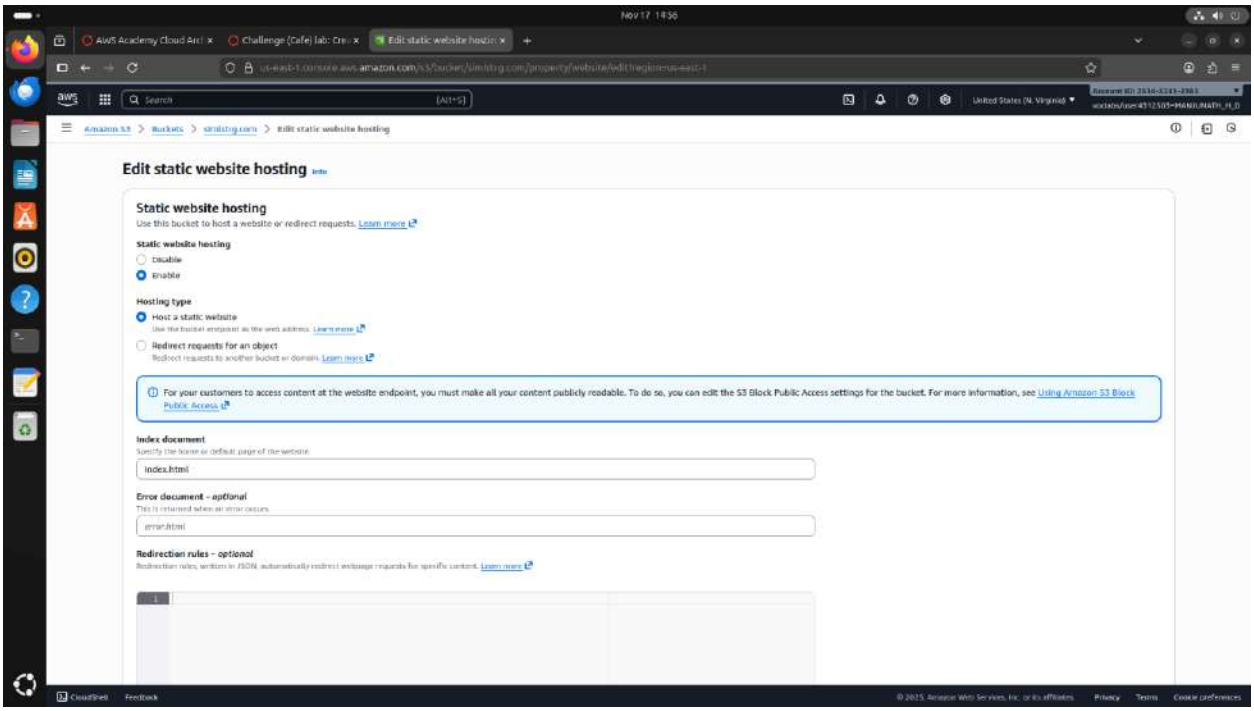

STEP 6 : Upload the index.html file and the folders css and images.



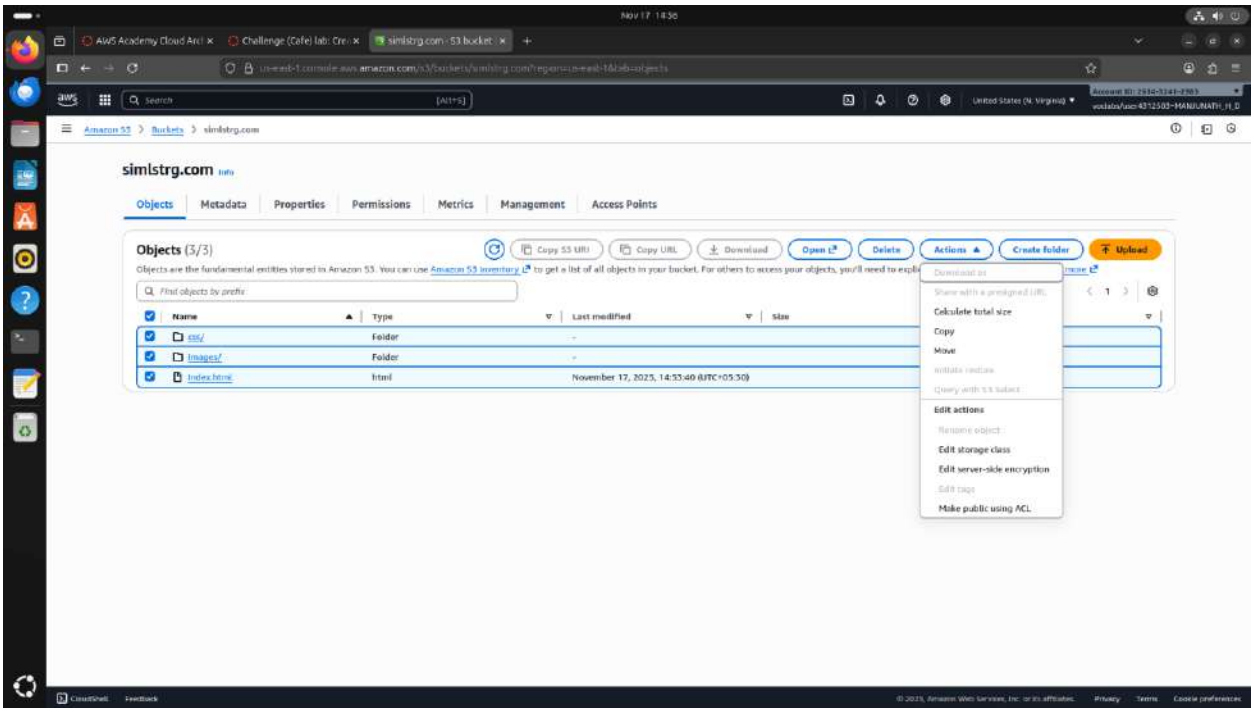

> Uploaded sucessfully



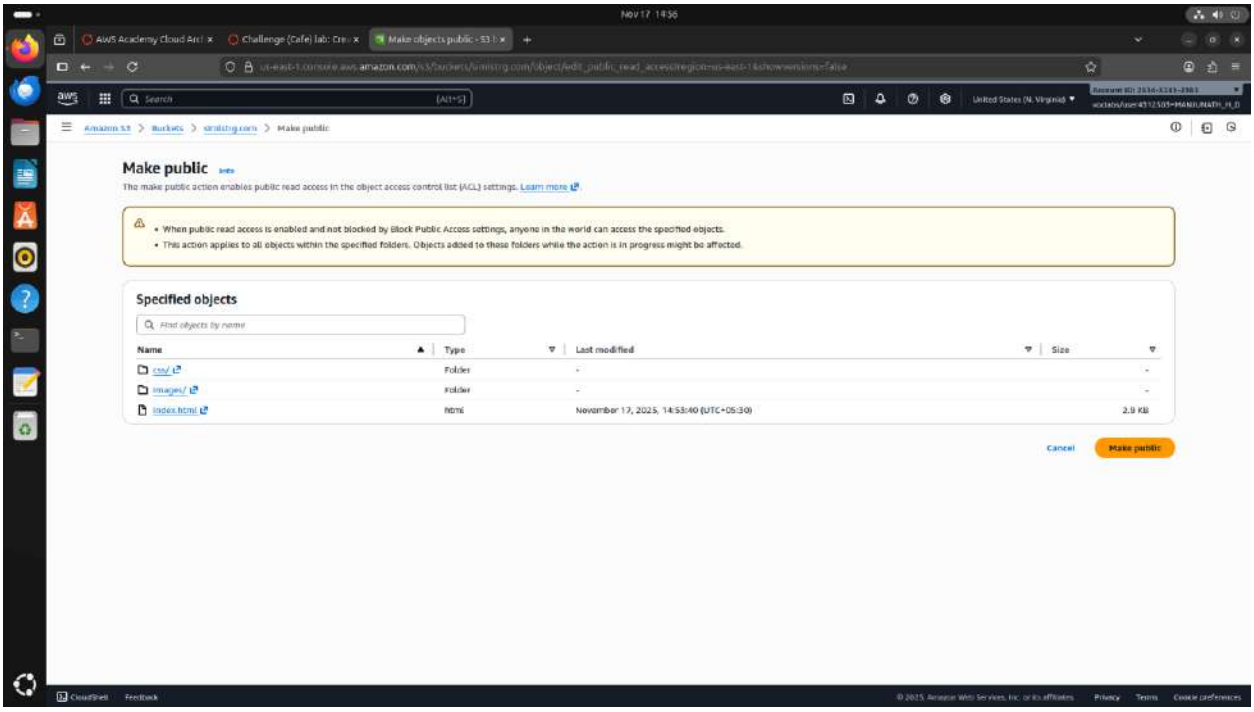

STEP 7 : Go to permissions and enable the ACL and give check mark to the i acknowledge to enable ACL.



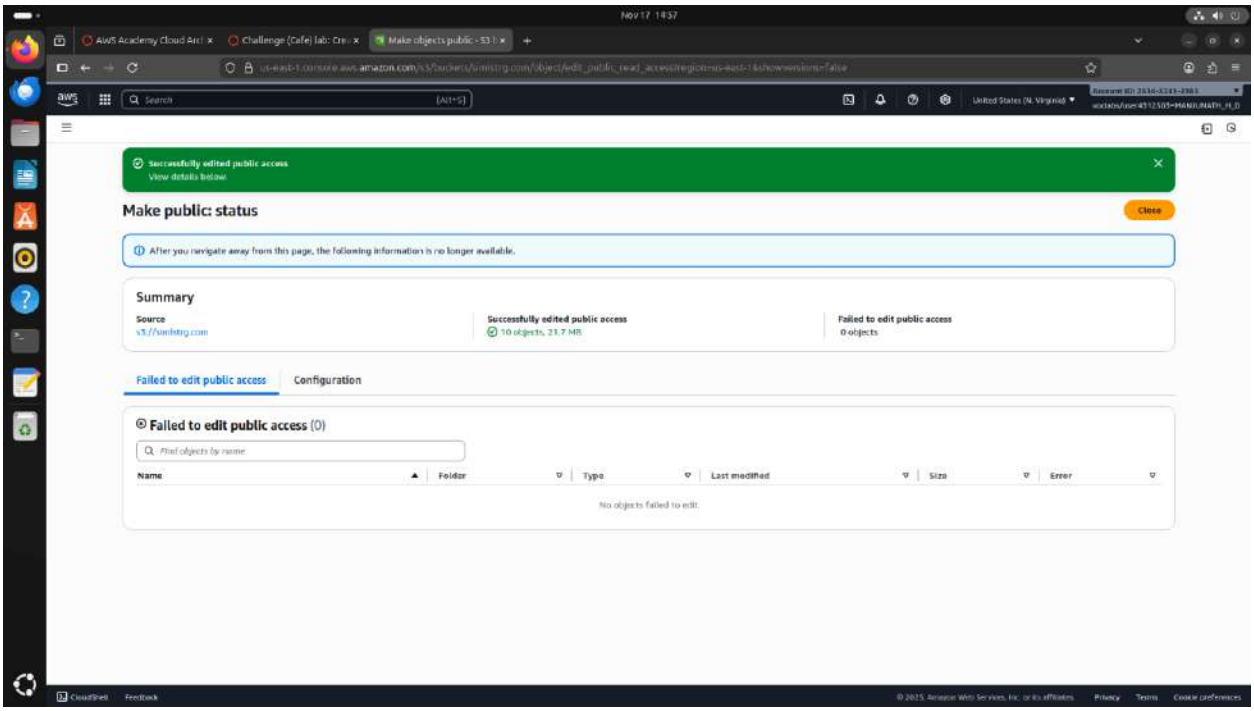

STEP 8: Go to properties and enable static website hosting and give the name index.html in the place of index document.



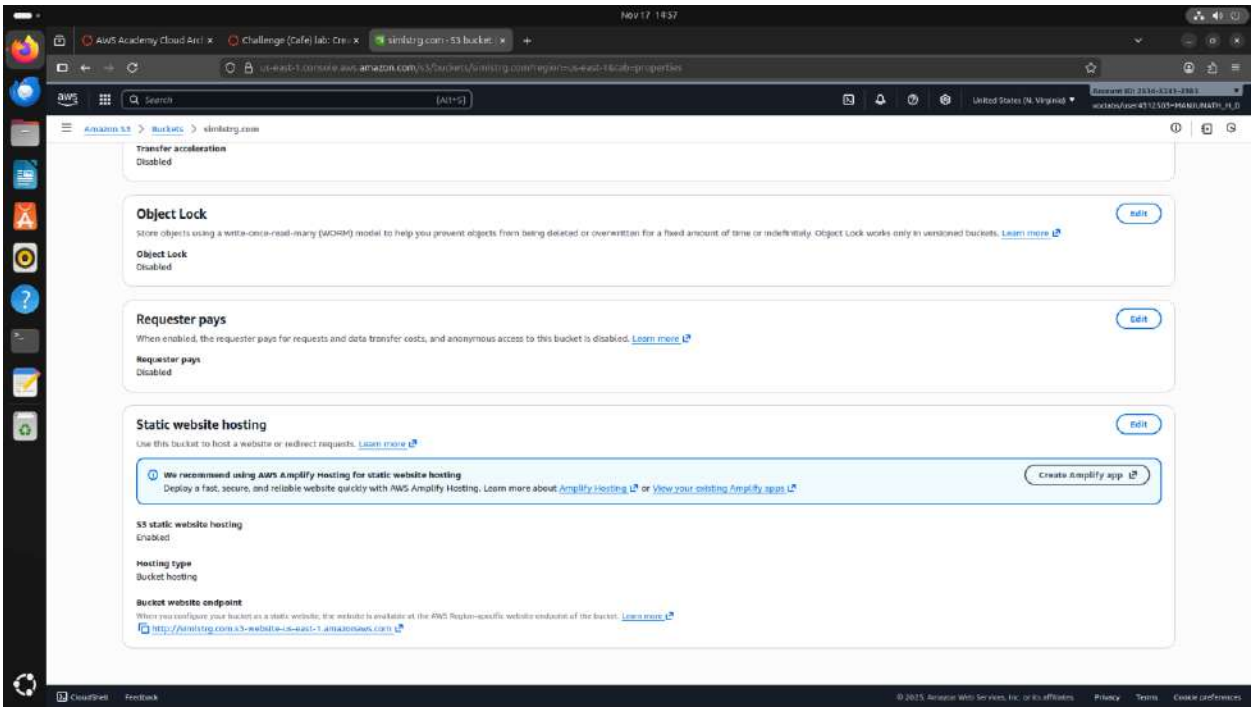

STEP 9 : Go to objects and check all the boxes which contains the uploaded files.

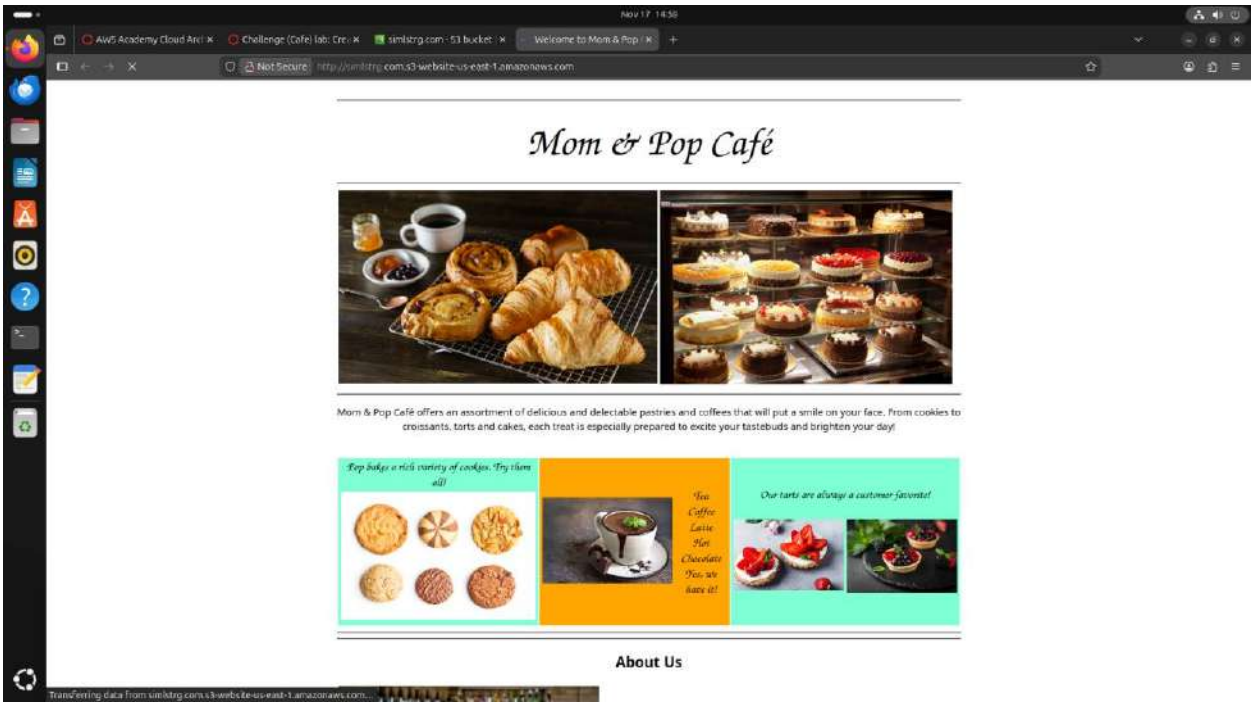



>and go to actions->make public using ACL and then press make public.




> Now the static website is ready for the public access.

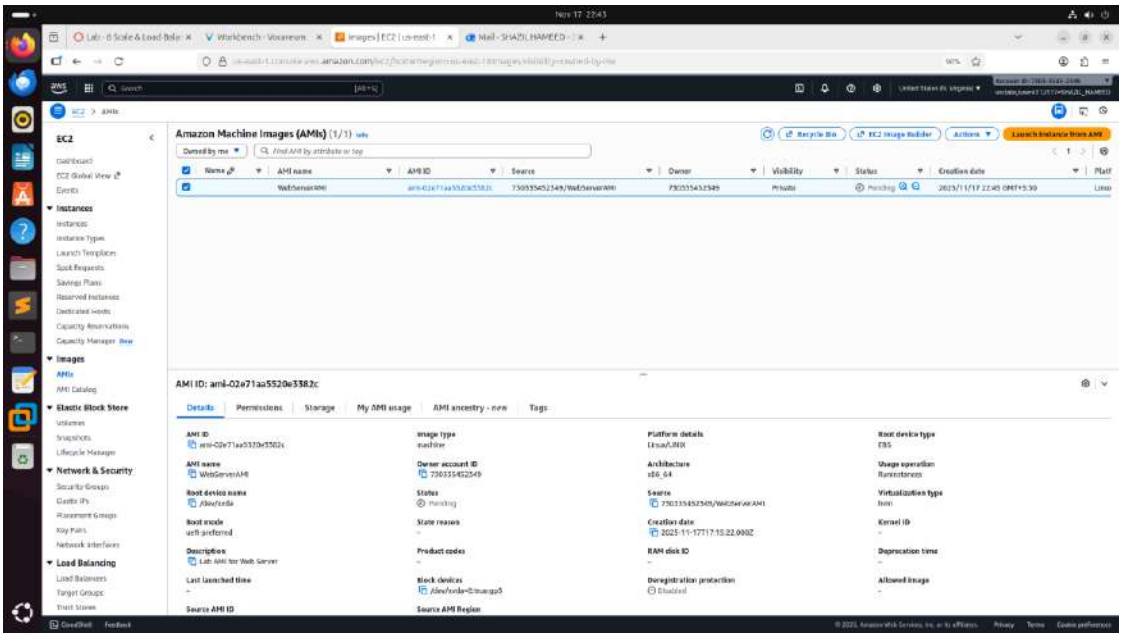


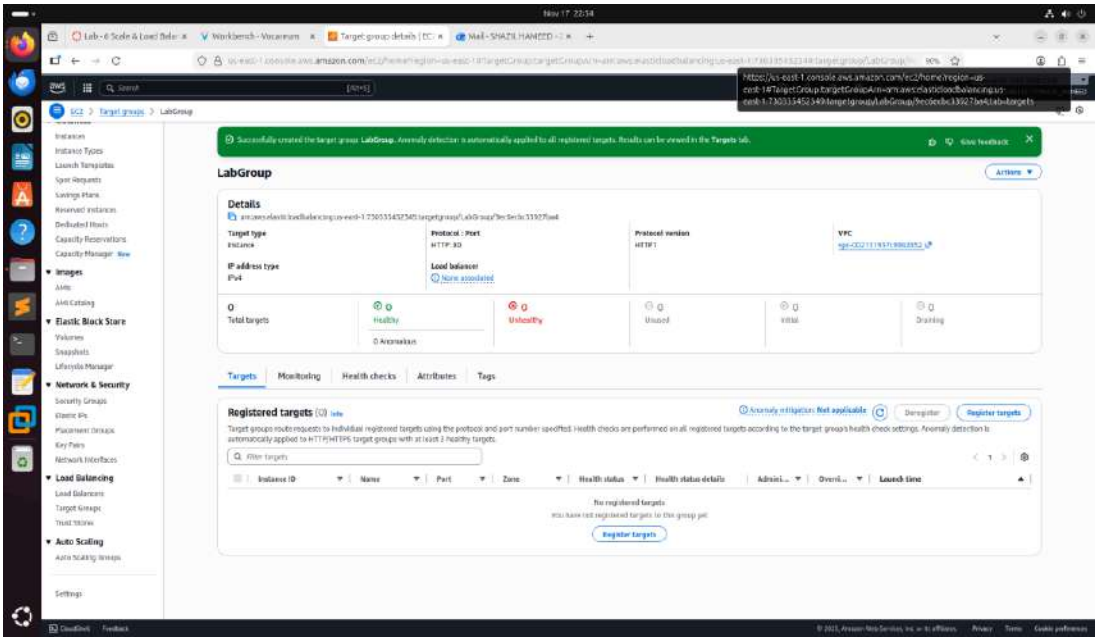

> Now go the properties and scroll down, there you get an link-> then the cafe-static-website will open on a new window.

# Scale and Load Balance Your Architecture

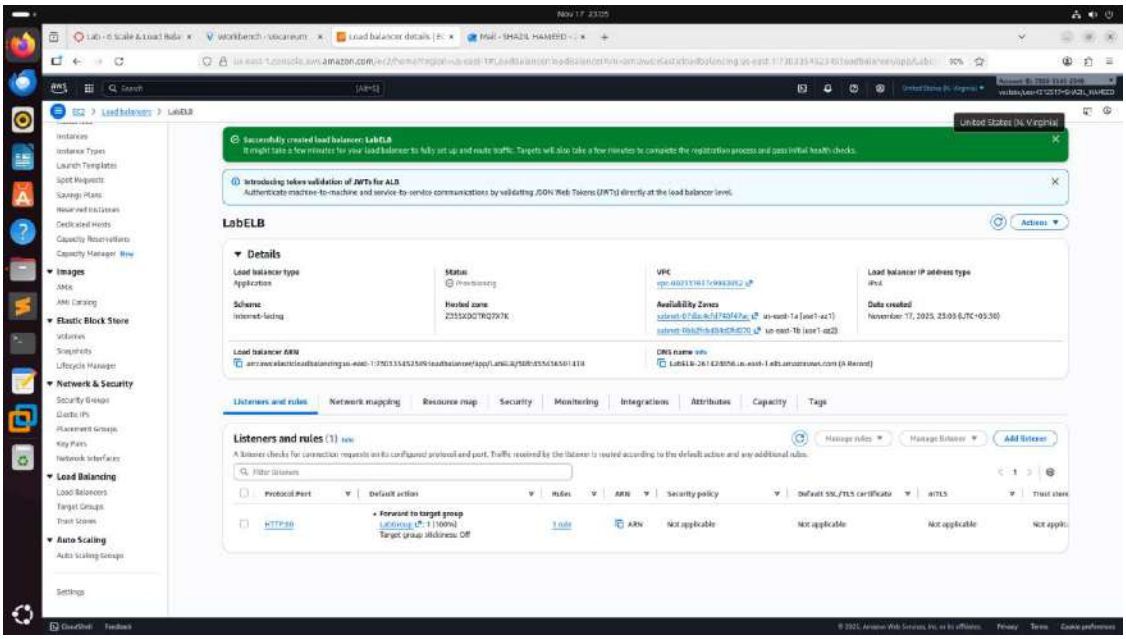
## 1.Create an Amazon Machine Image (AMI) from a running instance




## 2. Create a Load Balancer

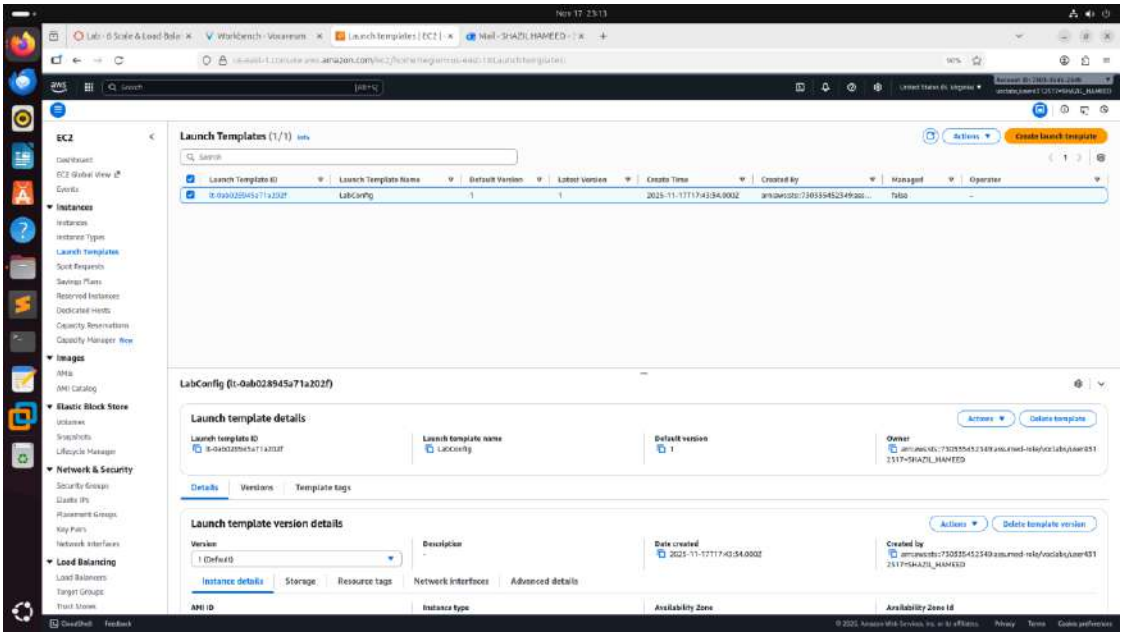



Created target group

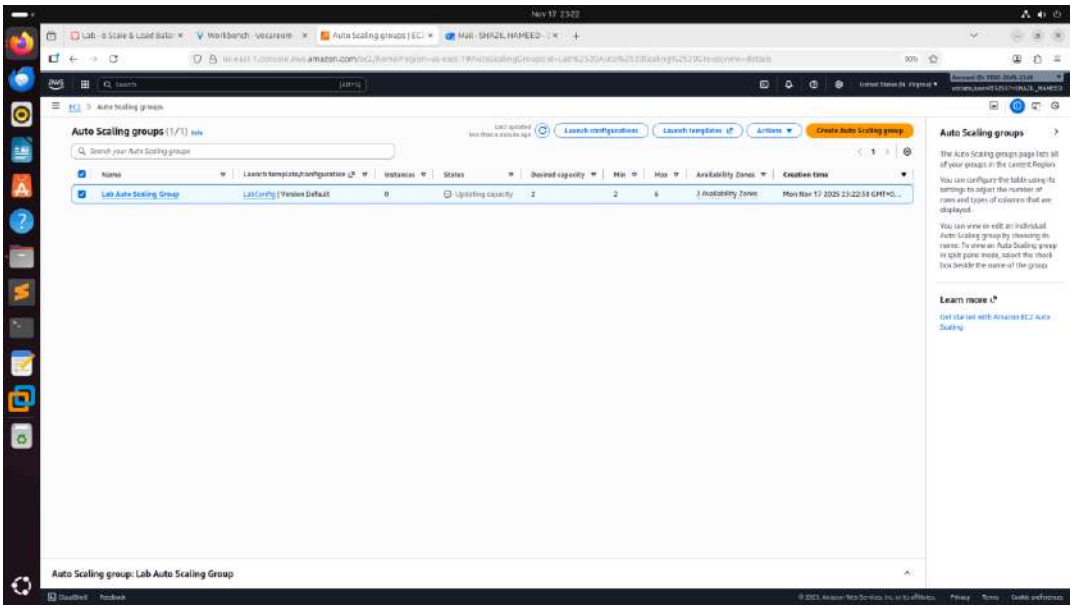



**Created  Load Balancer**

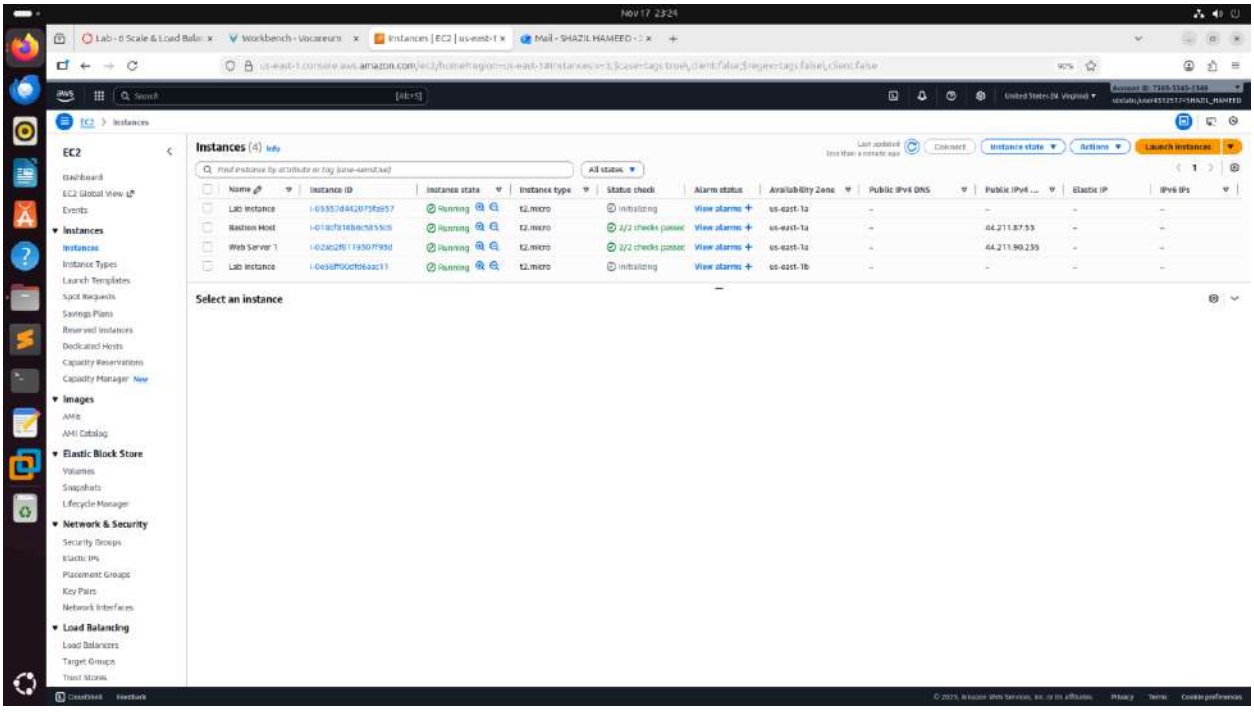
## 3 Create a Launch Template and an Auto Scaling Group




**Created launch template**




**Created auto scaling group**




Verified  Load Balancing is Working

# VPC Environment Creation and Configuration Lab Report

## Objective

The objective of this lab was to create and configure a custom Amazon Virtual Private Cloud (VPC) named lab-vpc in the us-east-1 region, complete with public and private subnets, and corresponding route tables.

## Lab Environment Setup

- The **AWS Management Console** was used for this lab.
- The lab focused on creating and configuring core VPC components: VPC, Subnets, and Route Tables.

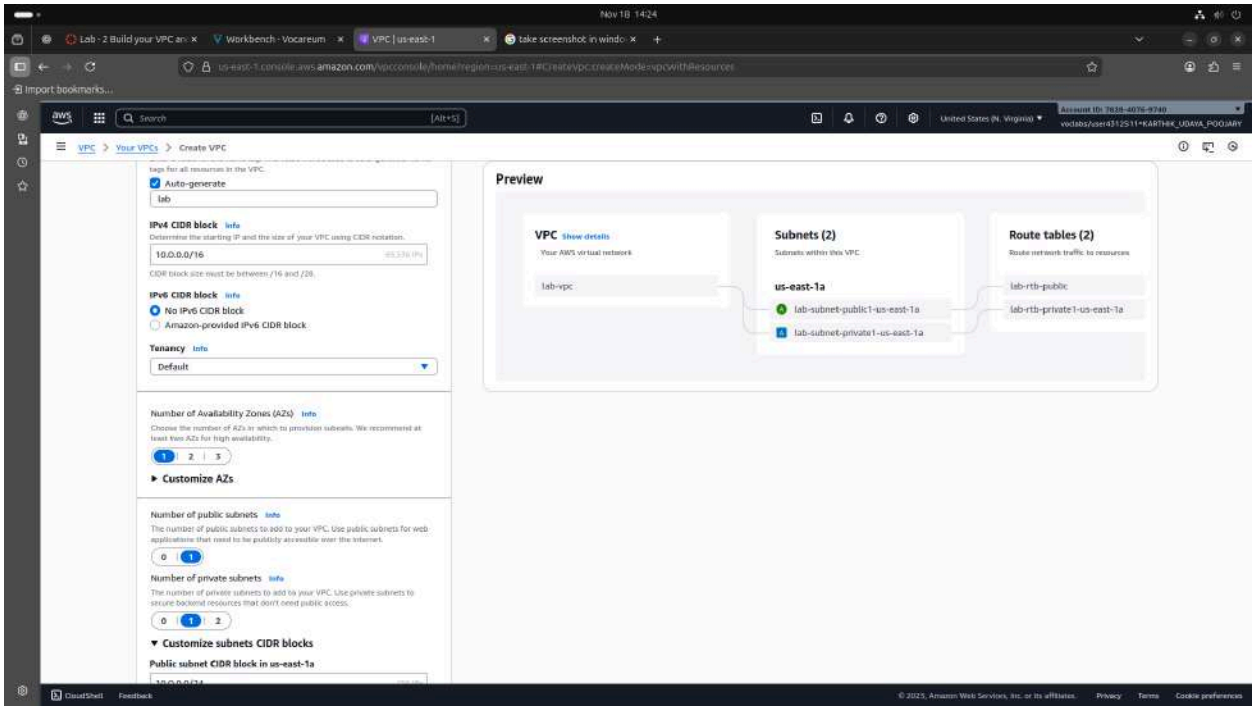
## Task 1: Create the Custom VPC and Subnets

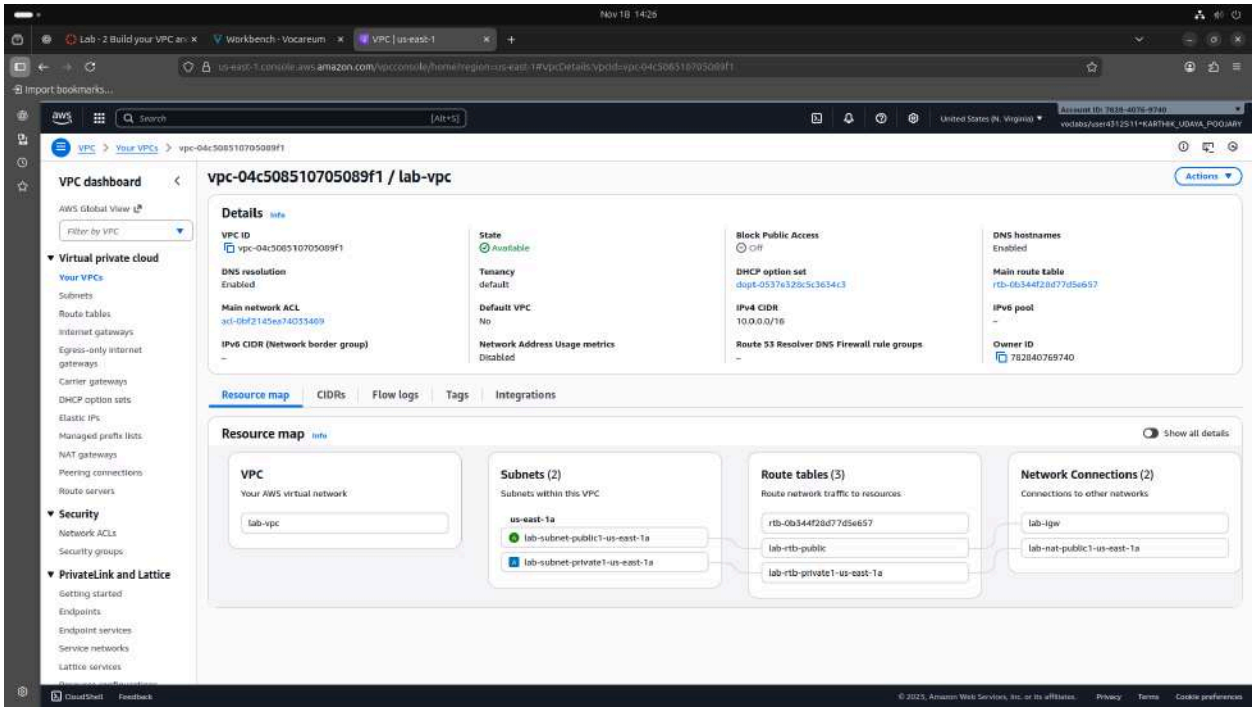
The VPC creation process involved the following steps:

1. **Initiation:** The VPC dashboard was accessed, and the "Create VPC" workflow was started.
2. **Configuration:**
    - The IPv4 CIDR block was set to `10.0.0.0/16`.
    - The VPC was configured to span a single Availability Zone (AZ).
    - One public and one private subnet were specified for creation within the VPC.

**Result**

The operation successfully created the VPC named `lab-vpc` (ID: `vpc-04c508510705089f1`). The resulting infrastructure included two subnets (`lab-subnet-public1-us-east-1a` and `lab-subnet-private1-us-east-1a`) and an associated Internet Gateway (IGW).


Nov 18 14:24
Lab - 2 Build your VPC an
Workbench - Vocareum
VPC | us-east-1
take screenshot in windo
Import bookmarks...
Search
[Alt+S]
United States (N. Virginia)
VPC > Your VPCs > Create VPC
Auto-generate
lab
IPv4 CIDR block Info
10.0.0.0/16
IPv6 CIDR block Info
No IPv6 CIDR block
Amazon-provided IPv6 CIDR block
Tenancy Info
Default
Number of Availability Zones (AZs) Info
1 2 3
Customize AZs
Number of public subnets Info
0 1
Number of private subnets Info
0 1 2
Customize subnets CIDR blocks
Public subnet CIDR block in us-east-1a
Preview
VPC Show details
Your AWS virtual network
lab-vpc
Subnets (2)
Subnets within this VPC
us-east-1a
lab-subnet-public1-us-east-1a
lab-subnet-private1-us-east-1a
Route tables (2)
Route network traffic to resources
lab-rtb-public
lab-rtb-private1-us-east-1a
CloudShell
Feedback
Privacy
Terms
Cookie preferences


Nov 18 14:26
VPC dashboard
AWS Global View
Filter by VPC
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
NAT gateways
Peering connections
Route servers
Security
Network ACLs
Security groups
PrivateLink and Lattice
Getting started
Endpoints
Endpoint services
Service networks
Lattice services
VPC > Your VPCs > vpc-04c508510705089f1
vpc-04c508510705089f1 / lab-vpc
Actions
Details Info
VPC ID
vpc-04c508510705089f1
State
Available
Block Public Access
Off
DNS hostnames
Enabled
DNS resolution
Enabled
Tenancy
default
DHCP option set
dopt-0537e328c5c3634c3
Main route table
rtb-06344f28d77d5e657
Main network ACL
acl-0bf2145ea74053409
Default VPC
No
IPv4 CIDR
10.0.0.0/16
IPv6 pool
IPv6 CIDR (Network border group)
Network Address Usage metrics
Disabled
Route 53 Resolver DNS Firewall rule groups
Owner ID
782840769740
Resource map
CIDRs
Flow logs
Tags
Integrations
Resource map Info
Show all details
VPC
Your AWS virtual network
lab-vpc
Subnets (2)
Subnets within this VPC
us-east-1a
lab-subnet-public1-us-east-1a
lab-subnet-private1-us-east-1a
Route tables (3)
Route network traffic to resources
rtb-0b344f28d77d5e657
lab-rtb-public
lab-rtb-private1-us-east-1a
Network Connections (2)
Connections to other networks
lab-igw
lab-nat-public1-us-east-1a
CloudShell
Feedback
Privacy
Terms
Cookie preferences

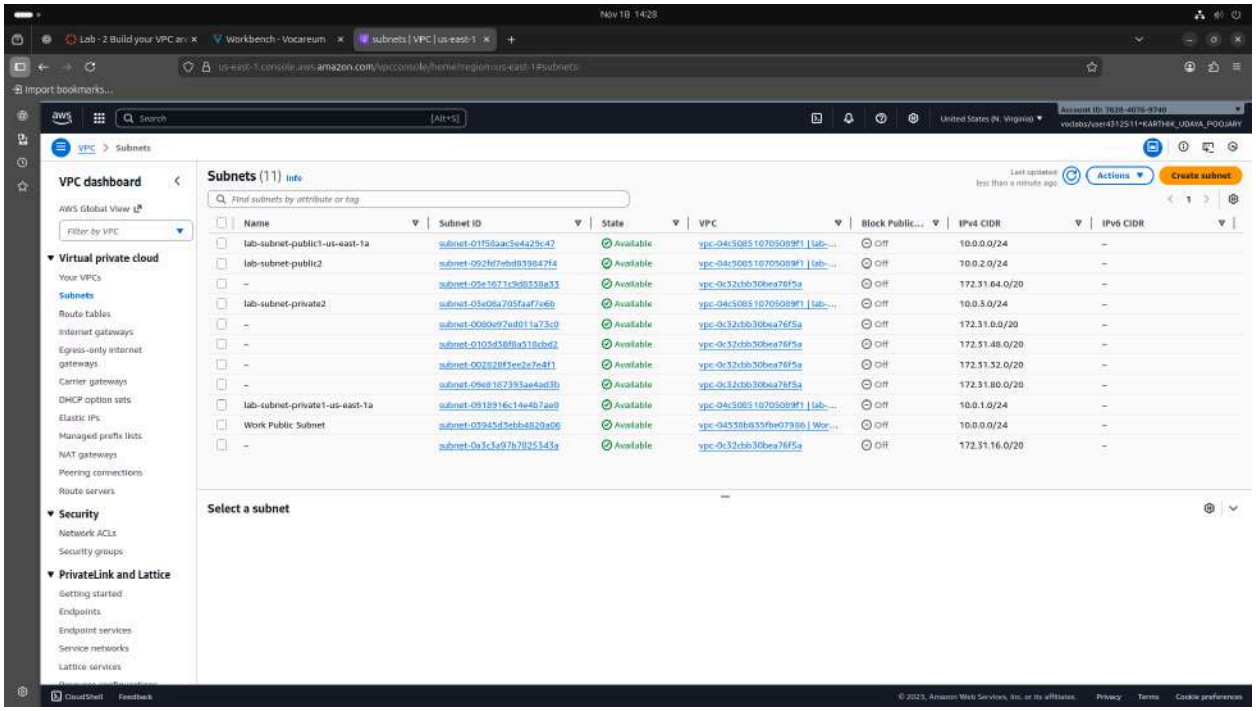
W

## Task 2: Verify Subnet Details

Steps:

1. Navigated to the Subnets section in the VPC console.
2. Verified the creation and status of the new subnets:lab-subnet-public1-us-east-1a andlab-subnet-private1-us-east-1a.

Outcome: The creation of the two specific subnets within the lab-vpc was confirmed, and their availability status was noted as "Available".



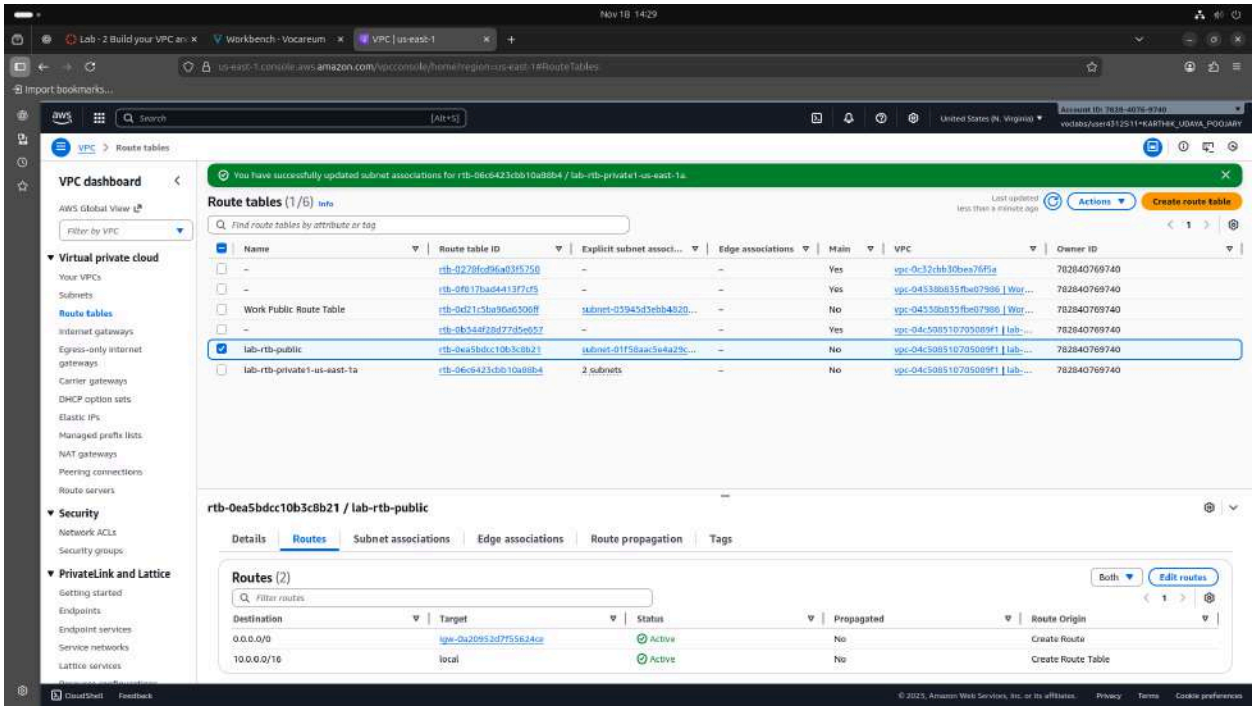

---

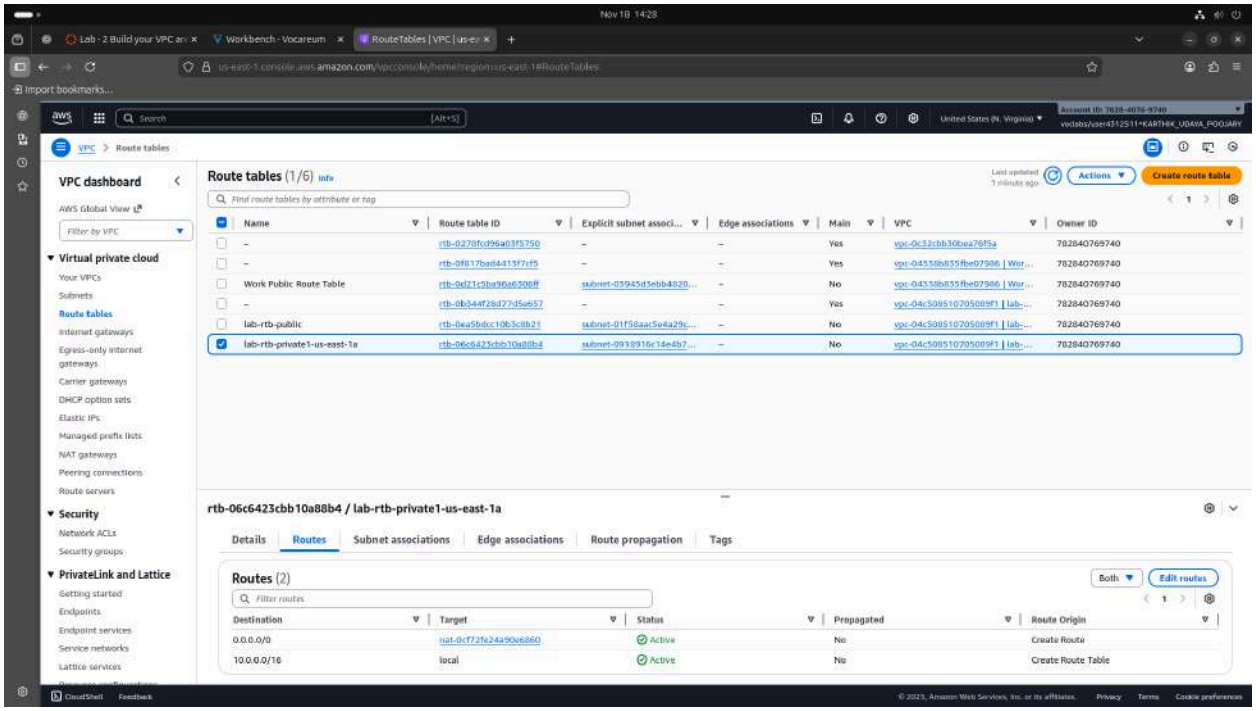
## Task 3: Configure and Verify Route Tables

Steps:

1. Navigated to the Route tables section.
2. Selected the public route table lab-rtb-public and verified the default route http://0.0.0.0/0 pointed to the Internet Gateway.
3. Selected the private route table ( and verified the default route http://0.0.0.0/0 pointed to the NAT Gateway.

Outcome: Both the public and private route tables were correctly configured and associated with their respective subnets, ensuring proper routing for external and internal traffic.


Nov 18 14:29
Lab - 2 Build your VPC an
Workbench - Vocareum
VPC | us-east-1
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#RouteTables
Import bookmarks...
Search
[Alt+S]
United States (N. Virginia)
VPC > Route tables
VPC dashboard
AWS Global View
Filter by VPC
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
NAT gateways
Peering connections
Route servers
Security
Network ACLs
Security groups
PrivateLink and Lattice
Getting started
Endpoints
Endpoint services
Service networks
Lattice services
You have successfully updated subnet associations for rtb-06c6423cbb10a88b4 / lab-rtb-private1-us-east-1a.
Route tables (1/6) Info
Find route tables by attribute or tag
Actions
Create route table
Name
Route table ID
Explicit subnet associ...
Edge associations
Main
VPC
Owner ID
rtb-0278fcd96a03f5750
rtb-0f617bad4413f7cf5
Work Public Route Table
rtb-0d21c5ba96a6506ff
subnet-05945d3ebb4820...
rtb-0b344f28d77d5e657
lab-rtb-public
rtb-0ea5bdcc10b3c8b21
subnet-01f58aac5e4a29c...
lab-rtb-private1-us-east-1a
rtb-06c6423cbb10a88b4
2 subnets
vpc-0c32cbb30bea76f5a
vpc-04538b835fbe07986 | Wor...
vpc-04c508510705089f1 | lab-...
782840769740
rtb-0ea5bdcc10b3c8b21 / lab-rtb-public
Details
Routes
Subnet associations
Edge associations
Route propagation
Tags
Routes (2)
Filter routes
Both
Edit routes
Destination
Target
Status
Propagated
Route Origin
0.0.0.0/0
igw-0a20952d7f55624ce
Active
No
Create Route
10.0.0.0/16
local
Active
No
Create Route Table
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

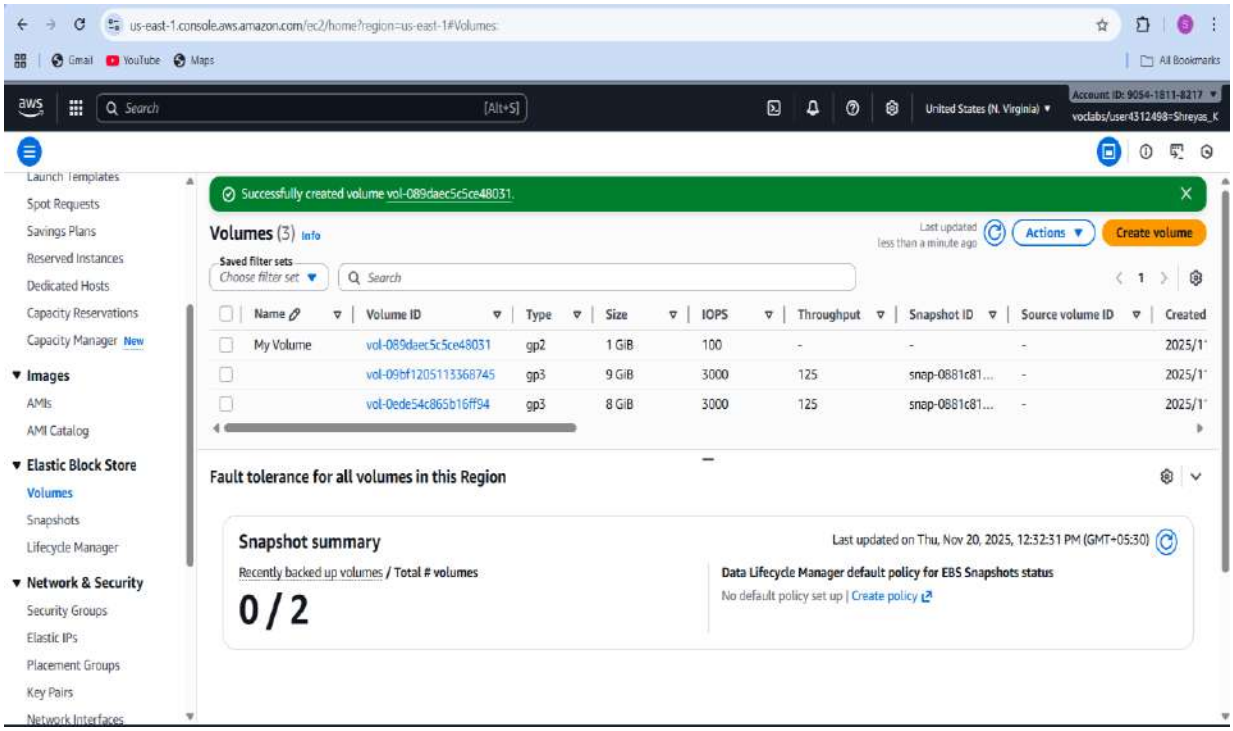

Nov 18 14:28
Lab - 2 Build your VPC an
Workbench - Vocareum
RouteTables | VPC | us-ea
us-east-1.console.aws.amazon.com/vpcconsole/home?region=us-east-1#RouteTables
Import bookmarks...
Search
[Alt+S]
United States (N. Virginia)
VPC > Route tables
VPC dashboard
AWS Global View
Filter by VPC
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
Carrier gateways
DHCP option sets
Elastic IPs
Managed prefix lists
NAT gateways
Peering connections
Route servers
Security
Network ACLs
Security groups
PrivateLink and Lattice
Getting started
Endpoints
Endpoint services
Service networks
Lattice services
Route tables (1/6) Info
Find route tables by attribute or tag
Actions
Create route table
Name
Route table ID
Explicit subnet associ...
Edge associations
Main
VPC
Owner ID
rtb-0278fcd96a03f5750
rtb-0f617bad4413f7cf5
Work Public Route Table
rtb-0d21c5ba96a6506ff
subnet-05945d3ebb4820...
rtb-0b344f28d77d5e657
lab-rtb-public
rtb-0ea5bdcc10b3c8b21
subnet-01f58aac5e4a29c...
lab-rtb-private1-us-east-1a
rtb-06c6423cbb10a88b4
subnet-0918916c14e4b7...
vpc-0c32cbb30bea76f5a
vpc-04538b835fbe07986 | Wor...
vpc-04c508510705089f1 | lab-...
782840769740
rtb-06c6423cbb10a88b4 / lab-rtb-private1-us-east-1a
Details
Routes
Subnet associations
Edge associations
Route propagation
Tags
Routes (2)
Filter routes
Both
Edit routes
Destination
Target
Status
Propagated
Route Origin
0.0.0.0/0
nat-0cf72fe24a90a6860
Active
No
Create Route
10.0.0.0/16
local
Active
No
Create Route Table
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

## Conclusion

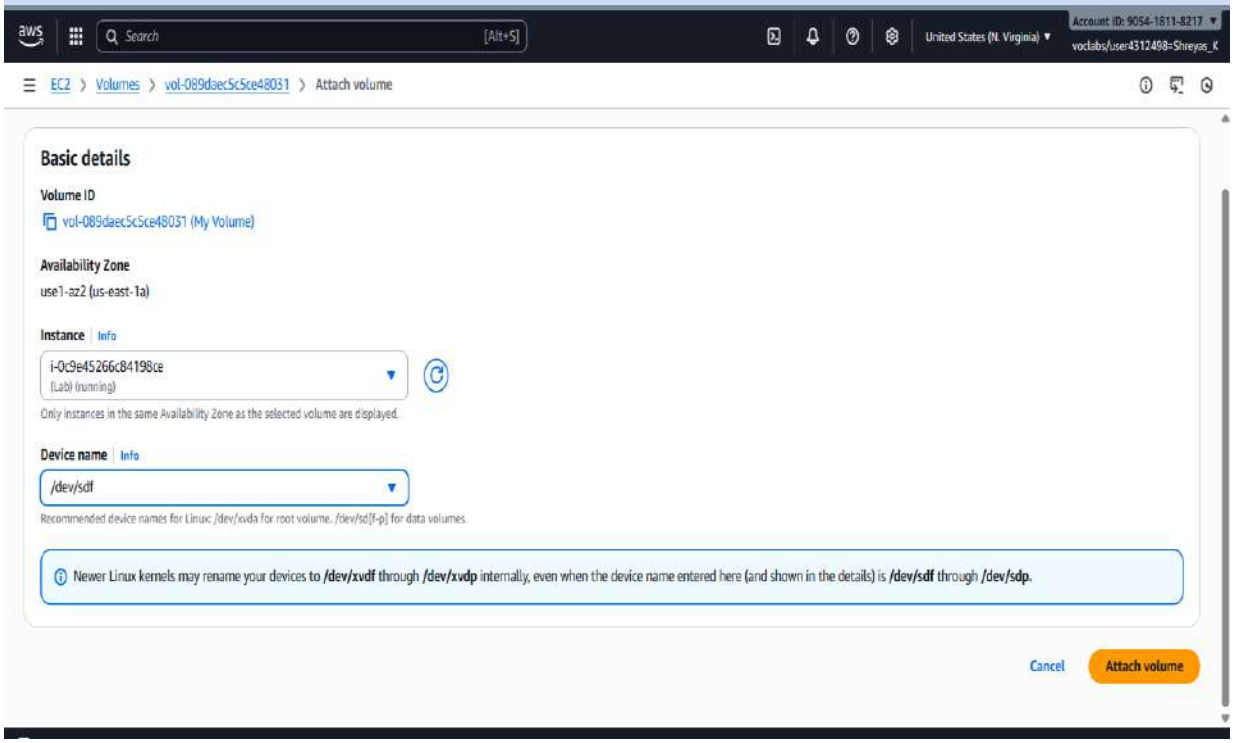
The lab successfully demonstrated the creation of a custom VPC environment with a CIDR block of 10.0.0.0/16, properly segregated into one public and one private subnet. The correct routing rules were applied to the associated route tables, establishing the foundational network for hosting applications with both public and private tiers on AWS.

# Lab 4: Working with EBS

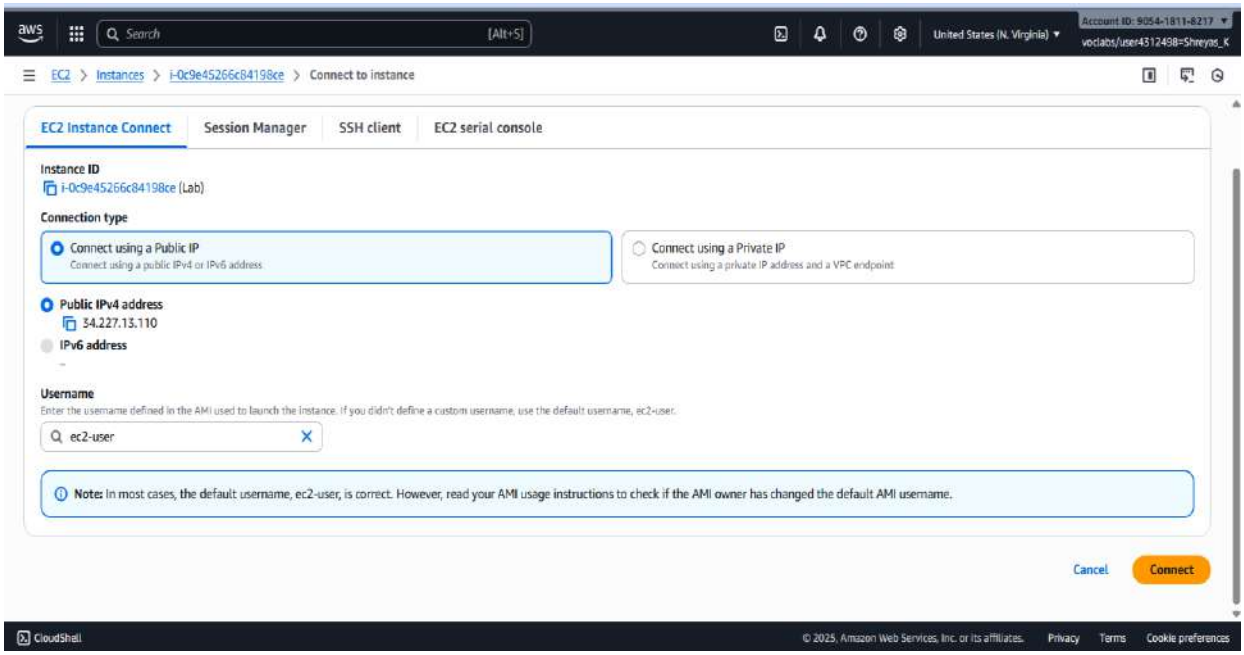
## Task 1: Create a New EBS Volume



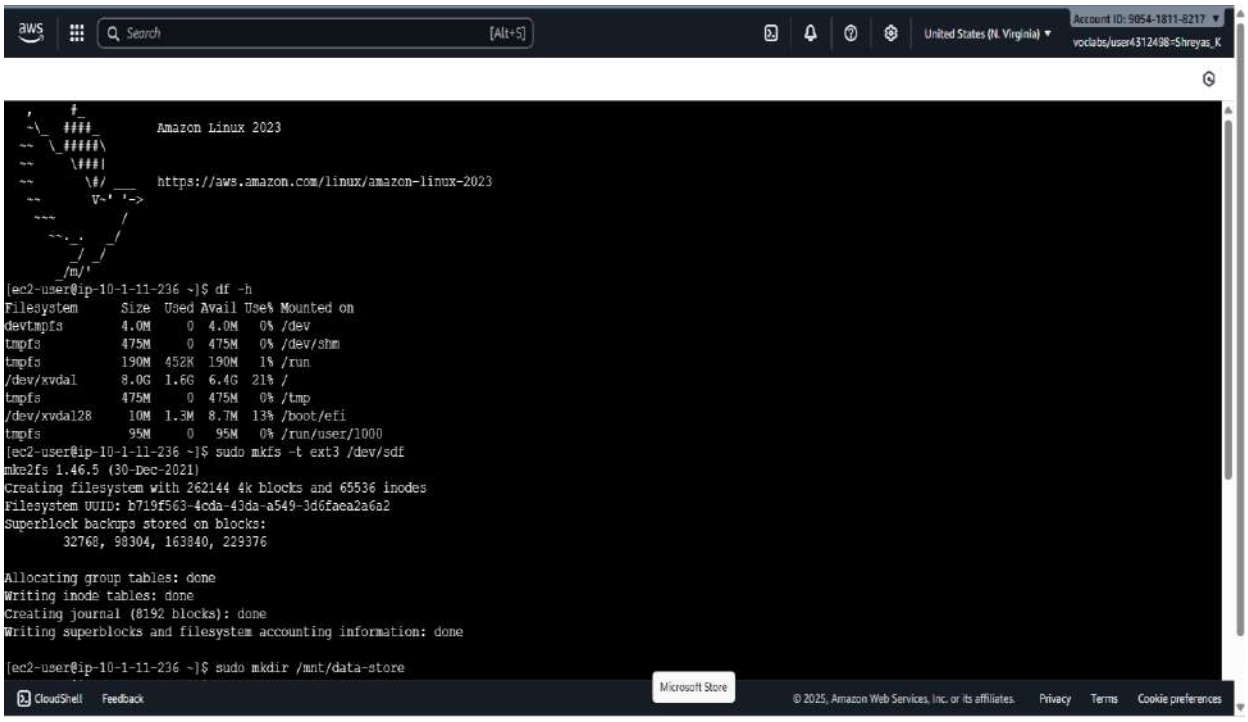

## Task 2: Attach the Volume to an Instance



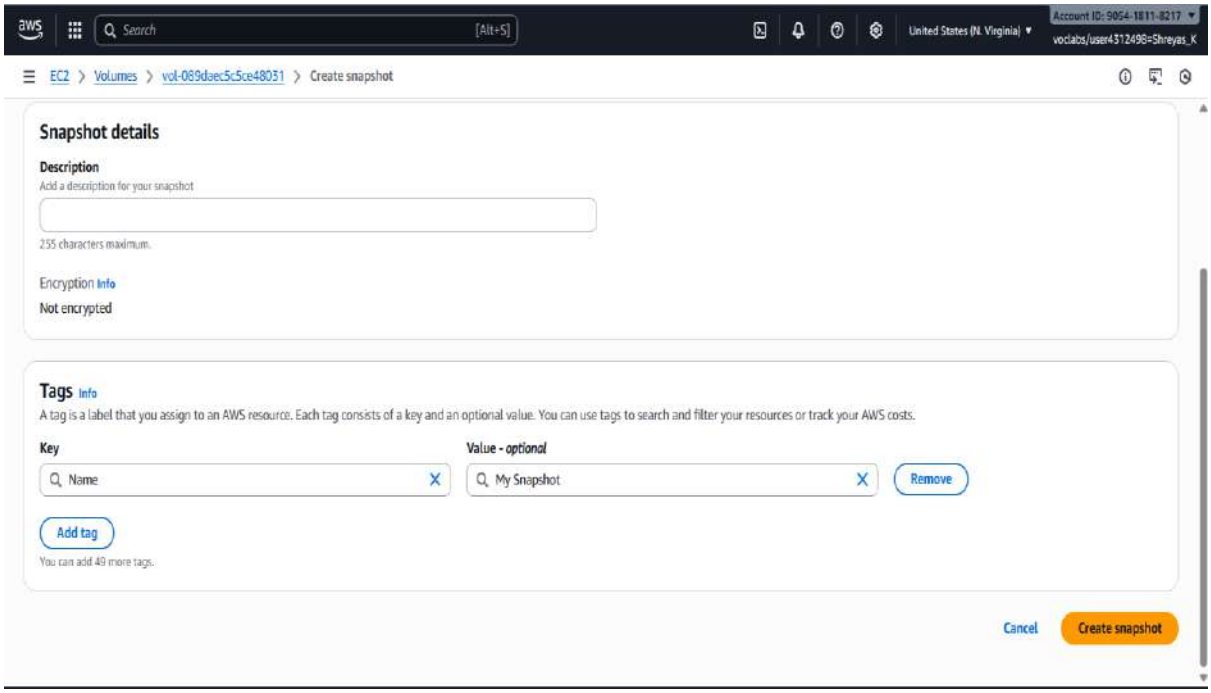
## Task 3: Connect to Your Amazon EC2 Instance

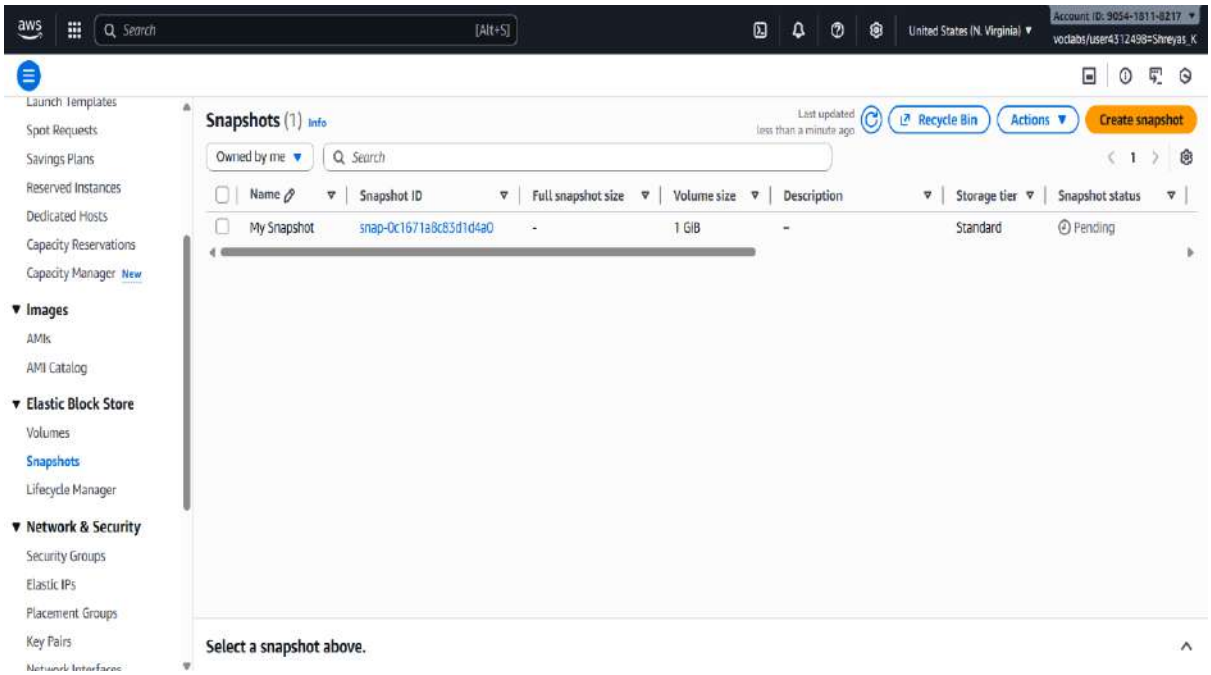


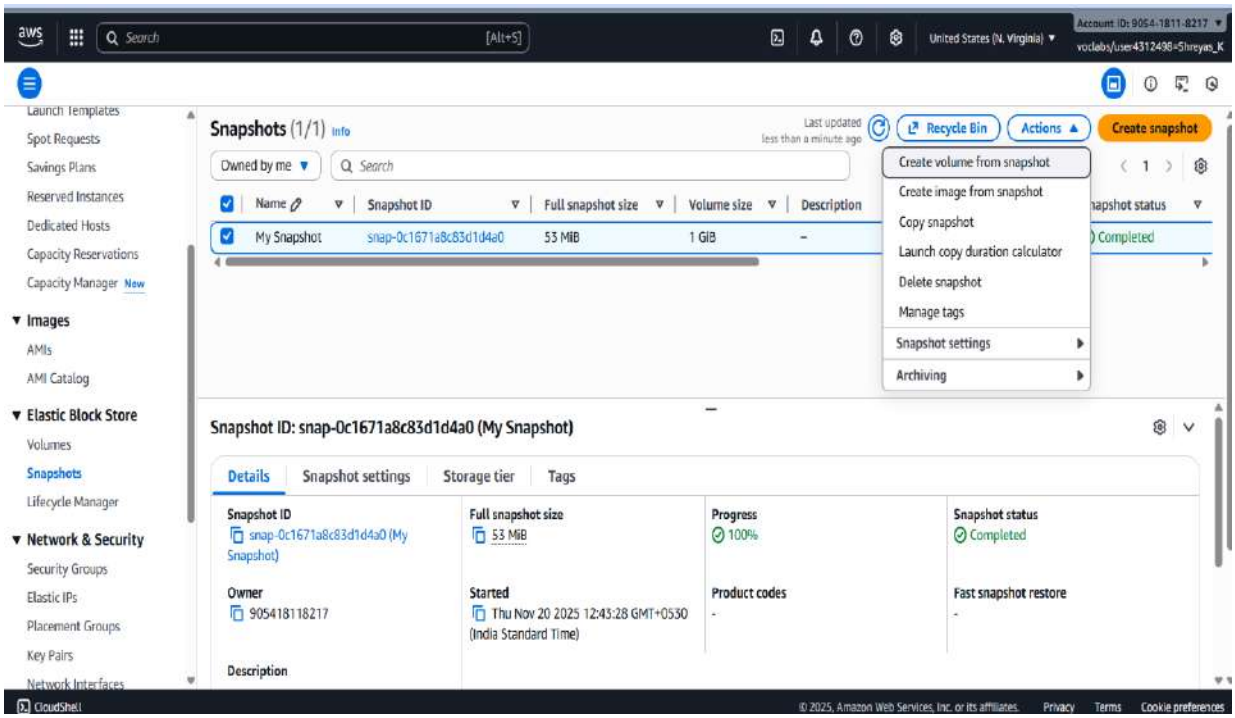

## Task 4: Create and Configure Your File System



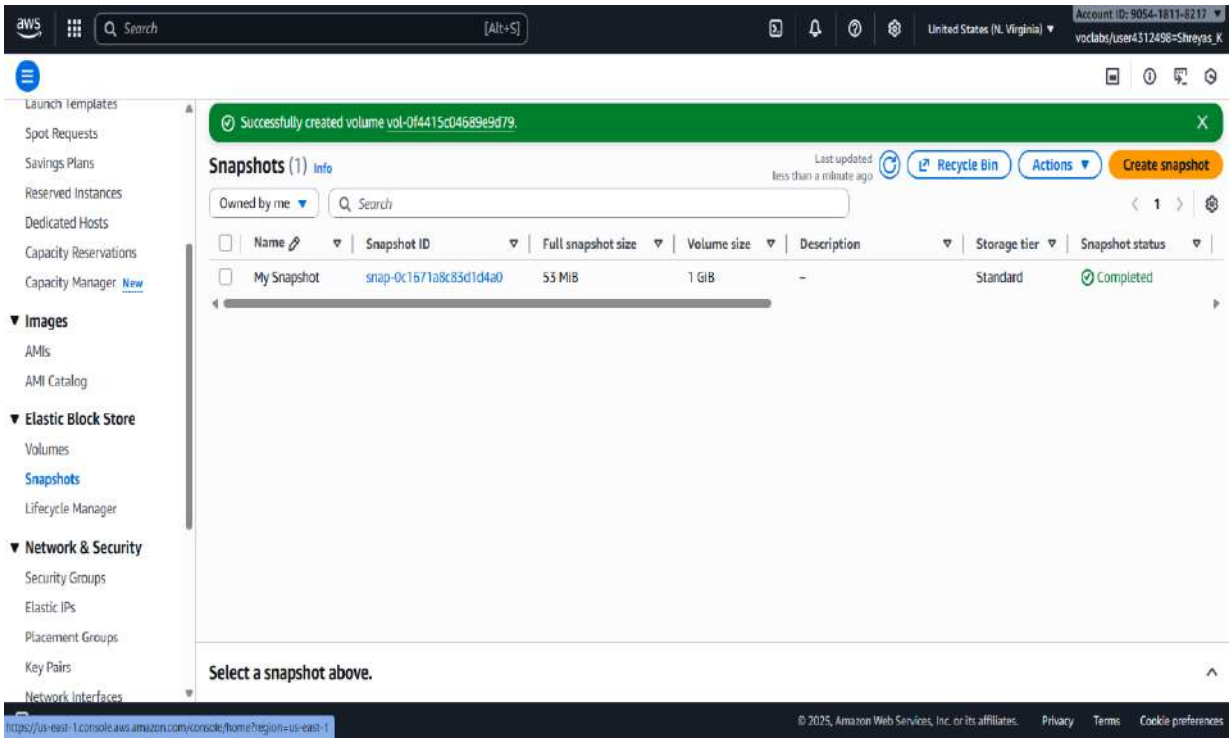
# Task 5: Create an Amazon EBS Snapshot



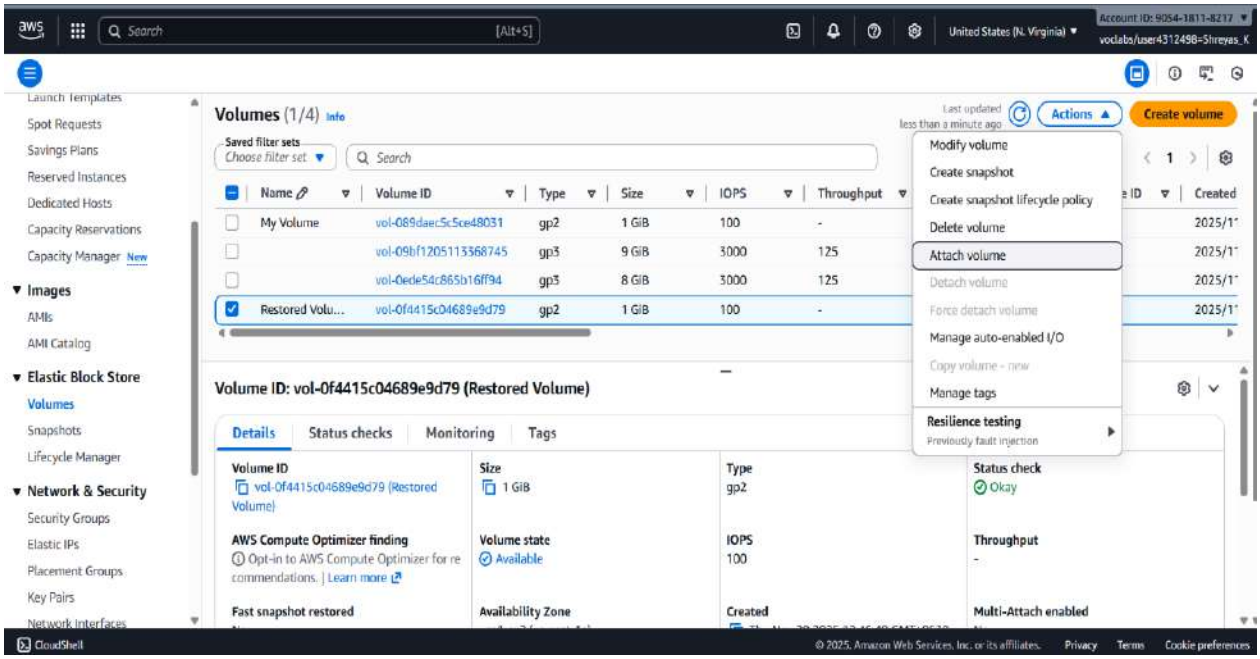



## Create a Volume Using Your Snapshot

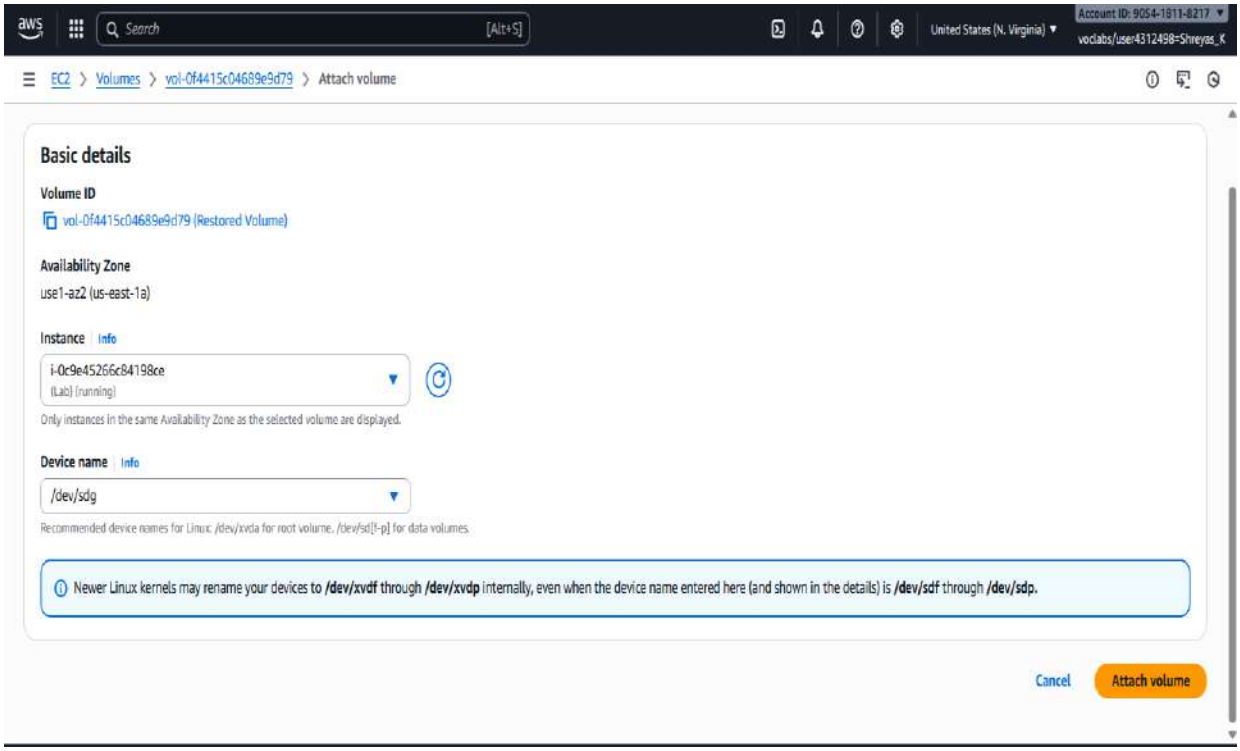


## Task 6: Restore the Amazon EBS Snapshot

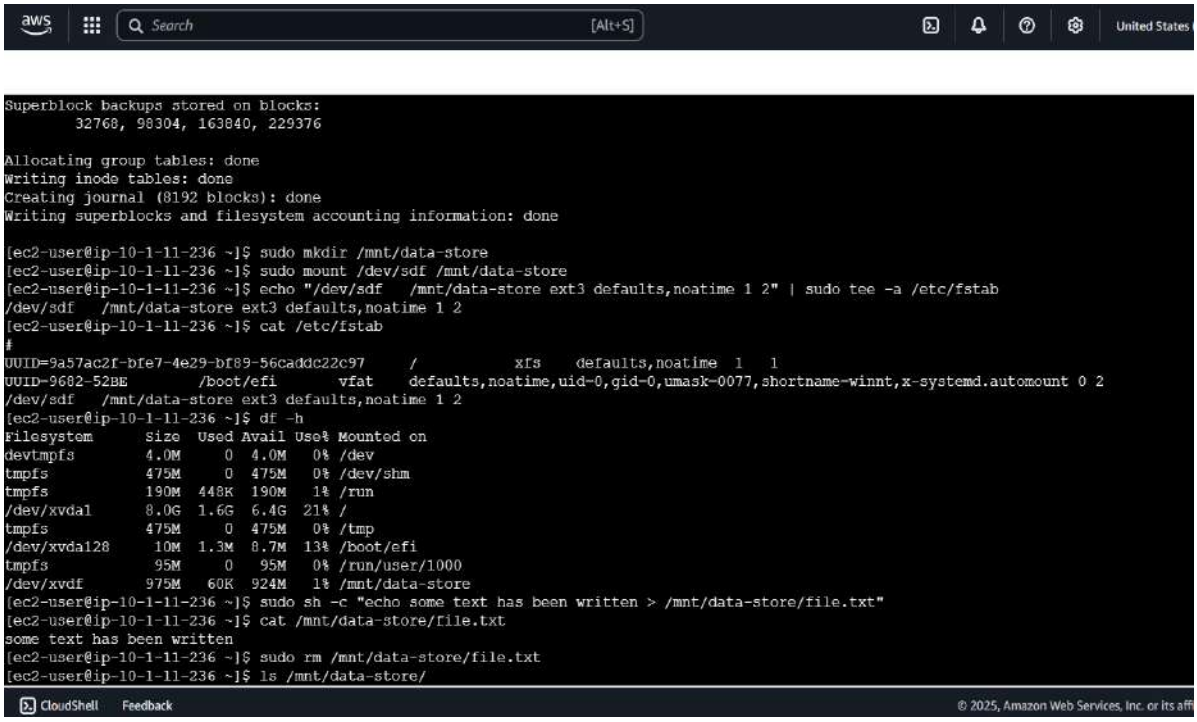
## Create a Volume Using Your Snapshot

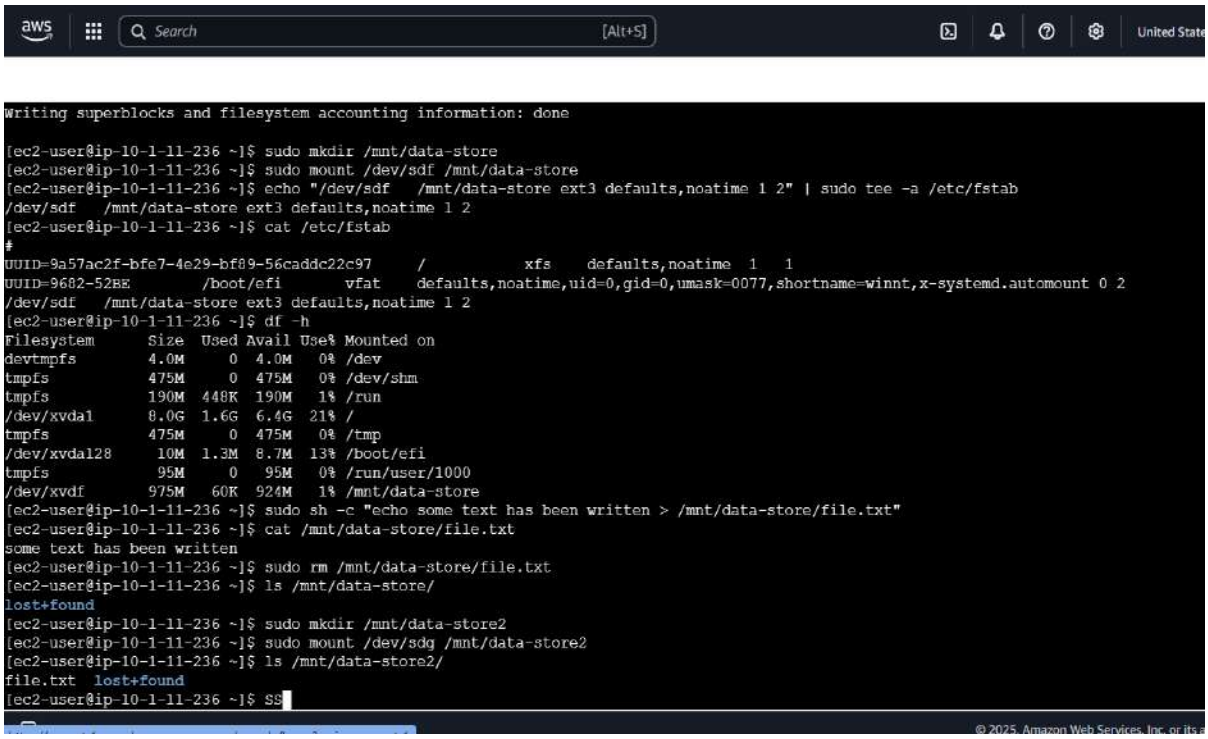



## Attach the Restored Volume to Your EC2 Instance



# Mount the Restored Volume



# Breaking a Monolithic Node.js Application into Microservices

**Objective:**

- Migrate a monolithic Node.js application to run in a Docker container.
- Refactor a Node.js application from a monolithic design to a microservices architecture.
- Deploy a containerized Node.js microservices application to Amazon ECS

**Task 1**: Connecting to the IDE and preparing the development environment on the EC2 instance.

1. At top of the instructions choose AWS Details
2. Copy values from the table

    a.LabIDEURL

    b.LabIDEPassword

3. In a new browser tab, paste the LabIDEURL On the prompt window Welcome to code-server: Enter the value for LabIDEPassword.
4. Choose submit

5. In the bottom pane of the IDE, in the terminal tab, run the following command:

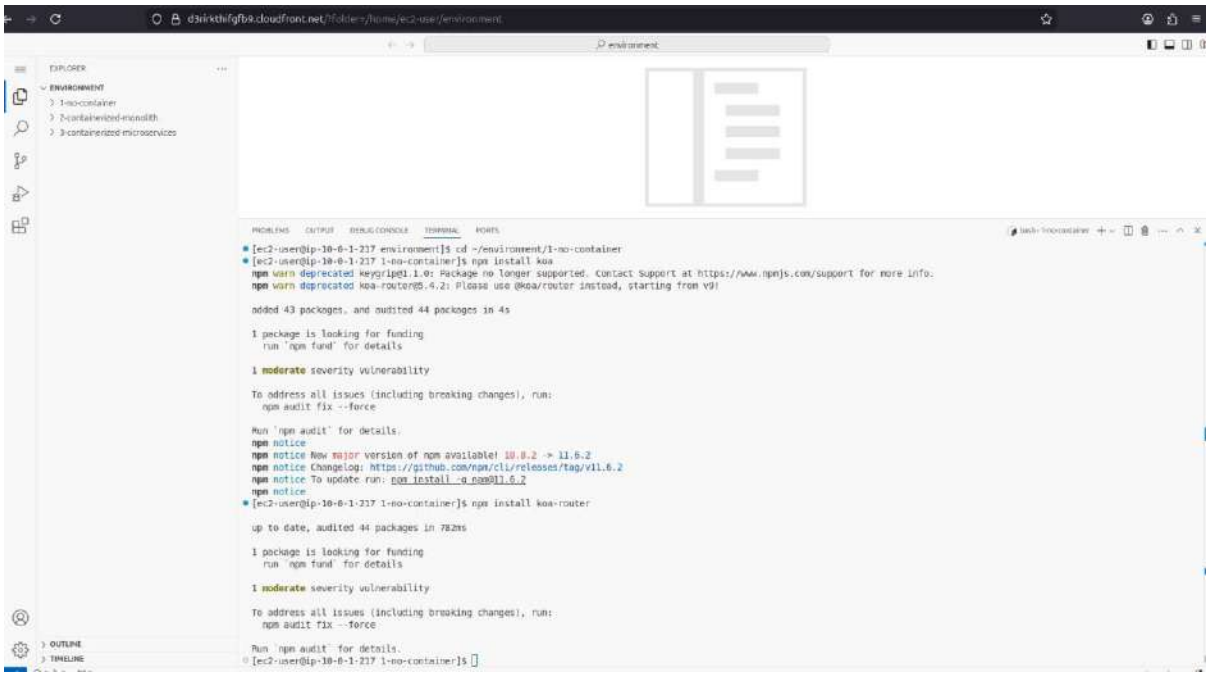
curl -s https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACACAD-3-113230/19-lab-mod14-guided-ECS/s3/lab-files-ms-node-js.tar.gz | tar –zxv

## Task 2: Running the application on a basic Node.js server

**Step 1**: Installing the required Node.js modules

1.In the Lab IDE terminal, to install the koa and koa-router modules, enter the following commands

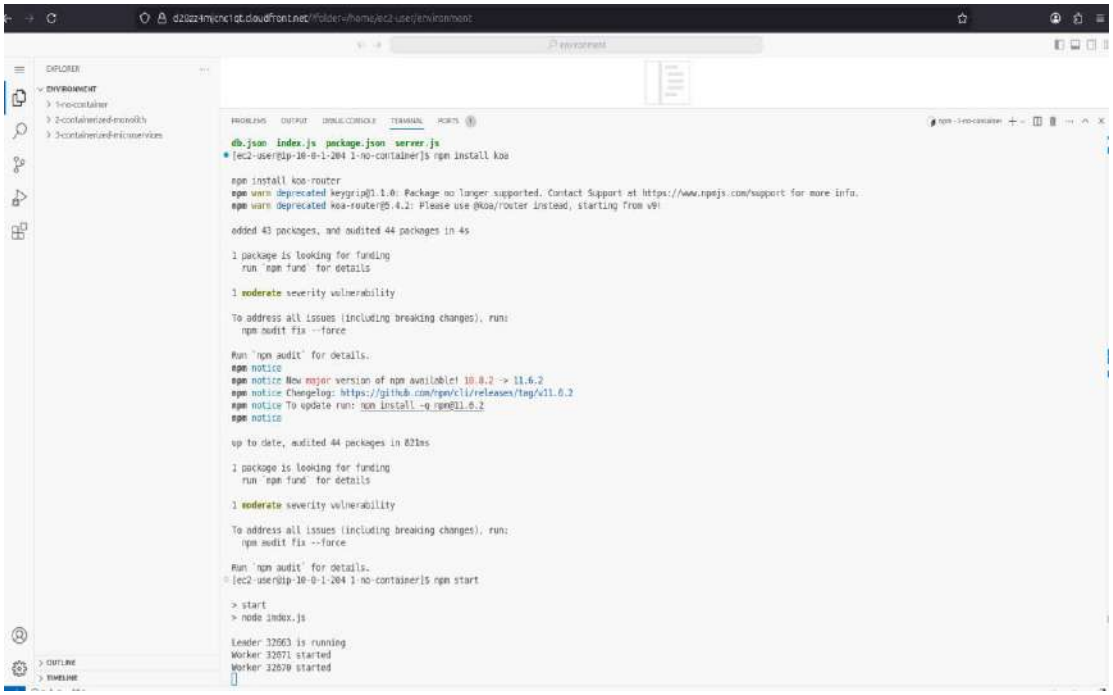
- cd ~/environment/1-no-container
- npm install koa
- npm install koa-router




**Step 2:** Running the application

1.In the terminal tab, to start the Node.js server and the application, enter the following command:

- Npm start




2.In the bottom pane, choose (+), and choose New Terminal to open a new terminal tab.

3.In the right terminal tab, to retrieve the /api/users resource, enter the following command:

```
curl localhost:3000/api/users
```

4. Retrieve information about 4th user: In the right terminal tab, enter the following command:

```
curl localhost:3000/api/users/4
```

5. Retrieve threads: In the right terminal tab, enter the following command:

```
curl localhost:3000/api/threads
```

6. Retrieve thread 1: In the right terminal tab, enter the following command:

```
curl localhost:3000/api/posts/in-thread/1
```

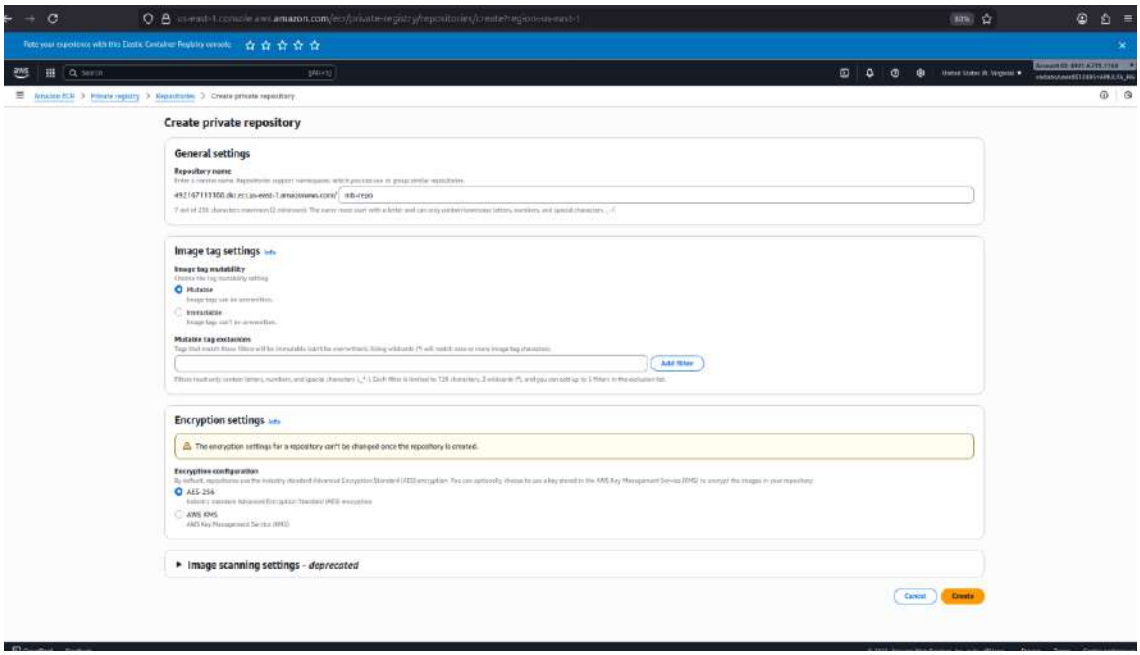
7.Now you stop the Node.js server.

8.In the left terminal tab, press Ctrl+C to shut down the server process.

## Task 3: Containerizing the monolith for Amazon ECS

**Step 1**: Provisioning the repository

1. On the AWS Management Console, in the search box, enter and choose Elastic Container Registry.

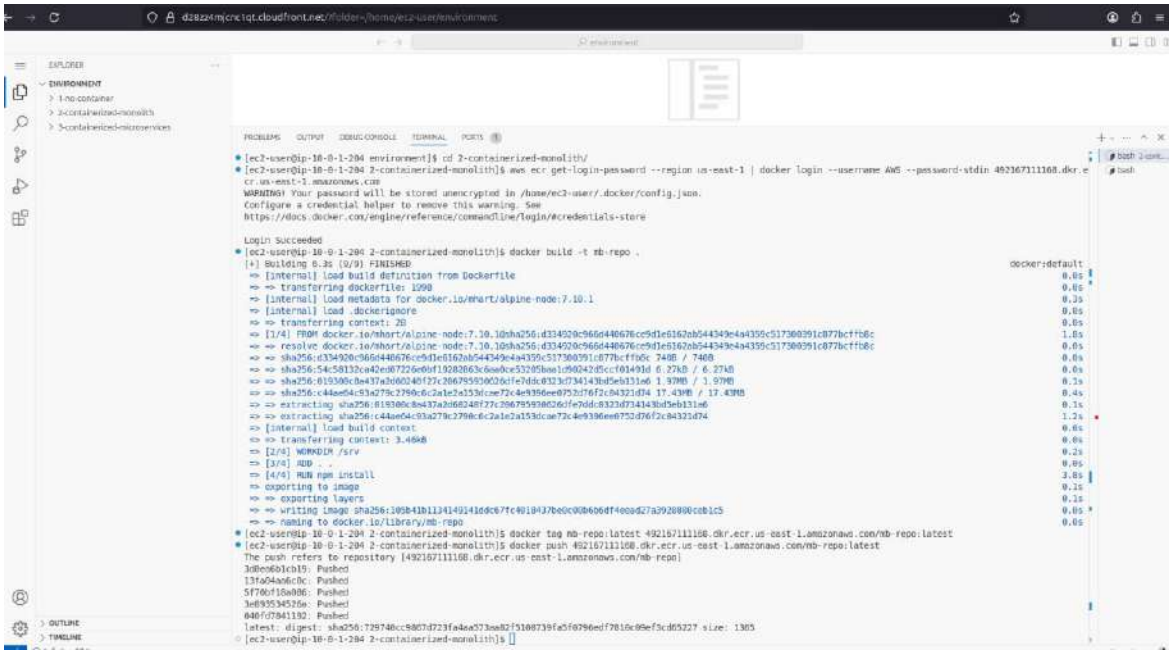



2. In the Create a repository section, choose Create

3. Under Create Private repositories
4. For the Repository name enter mb-repo
5. Choose Create.

**Step 2:** Building and pushing the Docker image

1. From the Private repositories listed, choose the mb-repo you created and then choose View push commands.
2. A pop-up window titled Push commands for mb-repo opens.



3. Make sure that the macOS/Linux tab is selected.
4. In the pop-up window, for the first command, choose the Copy icon to copy it to the clipboard.

   $ (aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin NNNNNNNNNNNN.dkr.ecr.us-east-1.amazonaws.com)

5. In the terminal tab, to change the directory to the 2-containerized-monolith folder, enter the following command:

   ```
   cd ~/environment/2-containerized-monolith
   ```

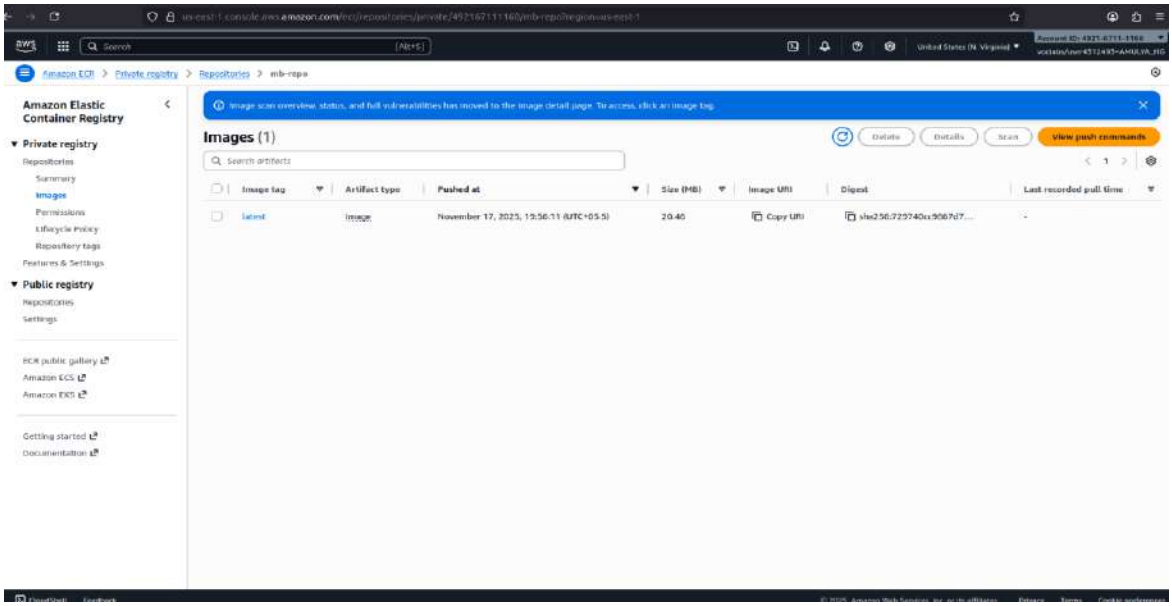
6. Switch to the Elastic Container Registry browser tab.
7. Copy the following commands:

   a. docker build -t mb-repo

b. docker tag mb-repo:latest  NNNNNNNNNNN.dkr.ecr.us- east-1.amazonaws.com/mb-repo:latest

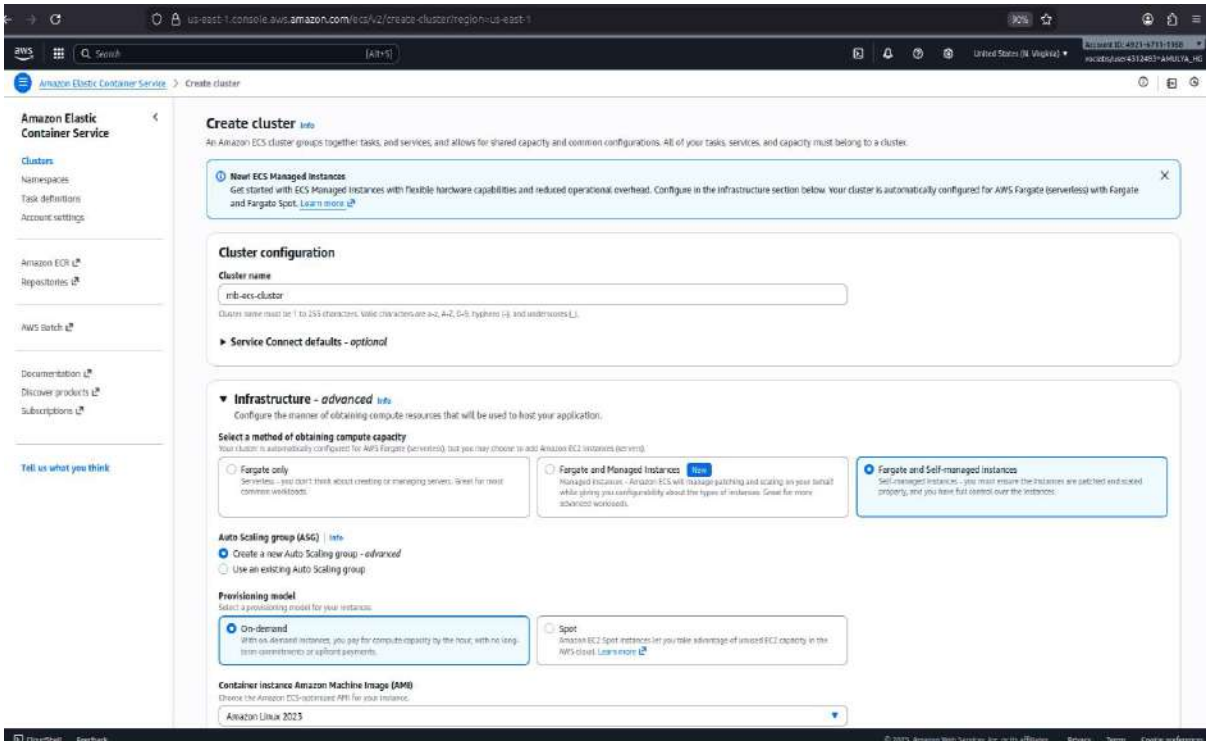
c.  docker push NNNNNNNNNNNN.dkr.ecr.us-east-1.amazonaws.com/mb-repo:latest

8. In the Repositories list, choose mb-repo
9. In the Images list, next to Copy URI, choose the Copy icon. Paste the value in a text editor.

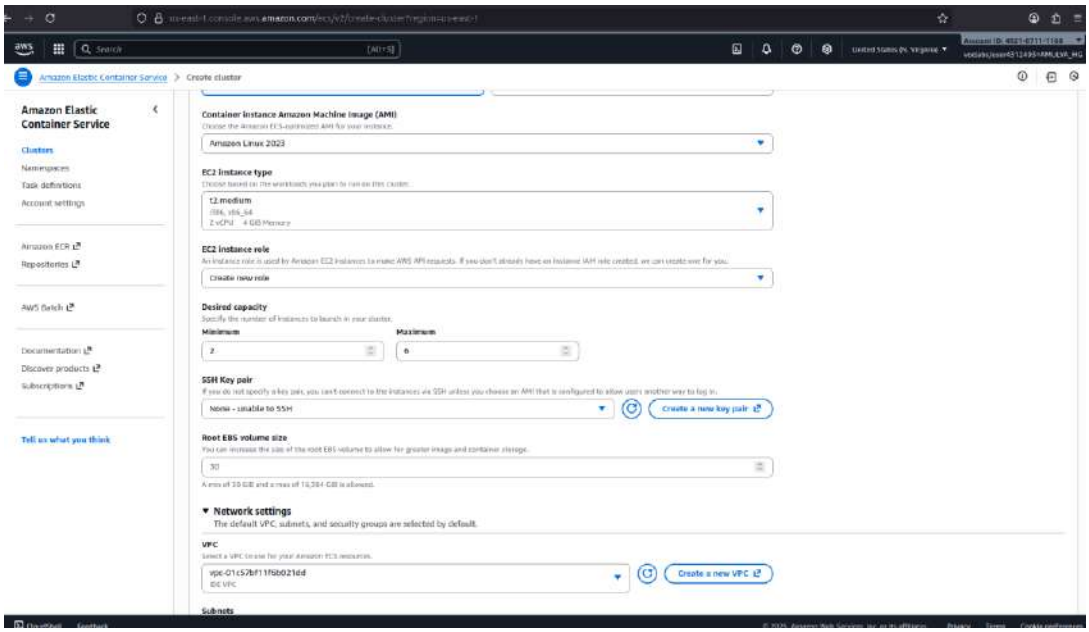



## Task 4: Deploying the monolith to Amazon ECS

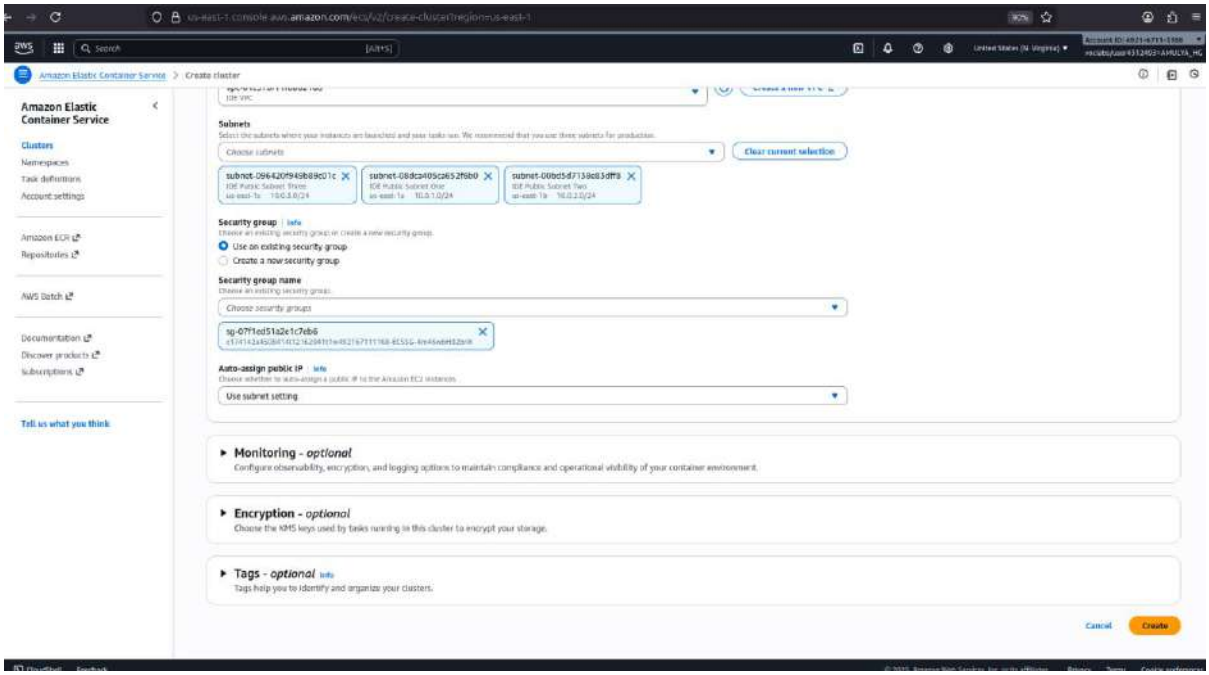
**Step 1:** Creating an Amazon ECS cluster



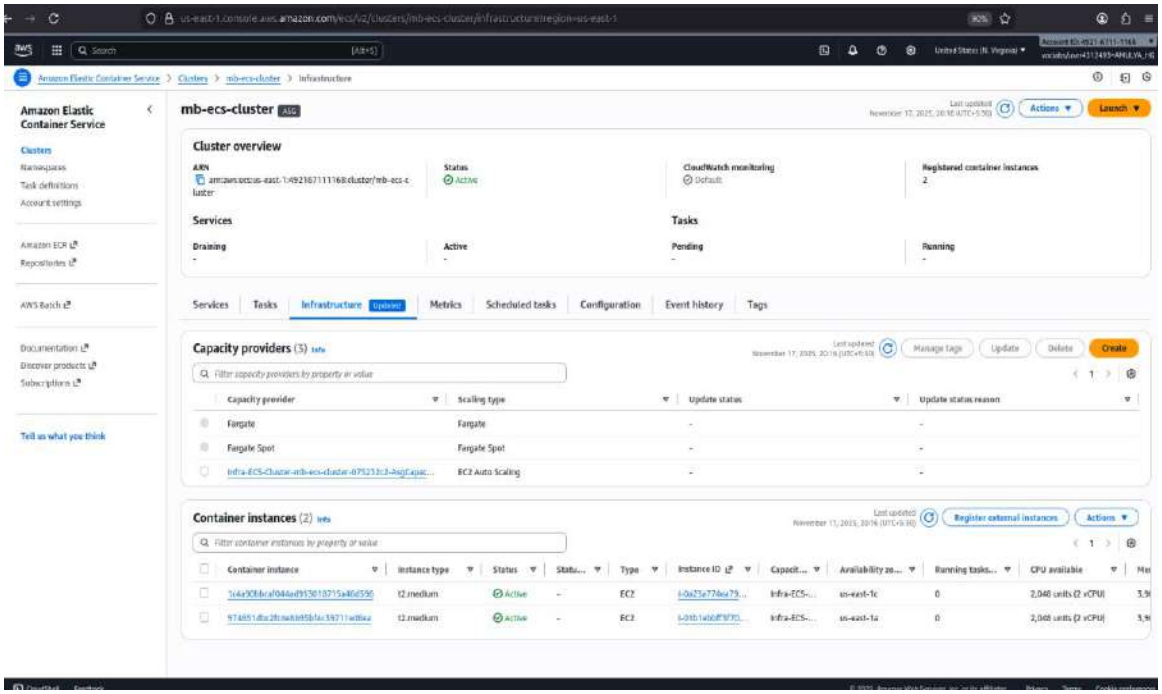

1. On the AWS Management Console**,** in the search box, enter and select Elastic Container Services
2. From the left menu, choose Clusters.
3. Choose Create cluster, and configure the following options:
4. In the Cluster configuration section, for the Cluster name, enter mb-ecs-cluster.
5. Uncheck *AWS Fargate (serverless)*
6. Check Amazon EC2 instances,
7. Leave the Auto Scaling group (ASG) settings at defaults.



8. Provisioning model: On-demand.
9. For EC2 instance type, choose t2.medium.
10. For Container instance Amazon Machine Image (AMI), choose Amazon Linux 2023.
11. EC2 instance role: *Create new role*.
12. For Desired capacity, for Minimum 2 and Maximum, enter 6
13. VPC: choose IDE VPC.
14. Subnets: Reconfirm that, all public subnets chosen.
15. Security group: Choose Use an existing security group.
16. Security group name dropdown list, select the security group that has ECSSG in the name.
17. Clear the default security group
18. Choose Create.
19. Choose mb-ecs-cluster




20. Choose infrastructure tab
21. The Container instances pane shows that two EC2 instances for the cluster were created.




**Step 2:** Creating a task definition for the application container image

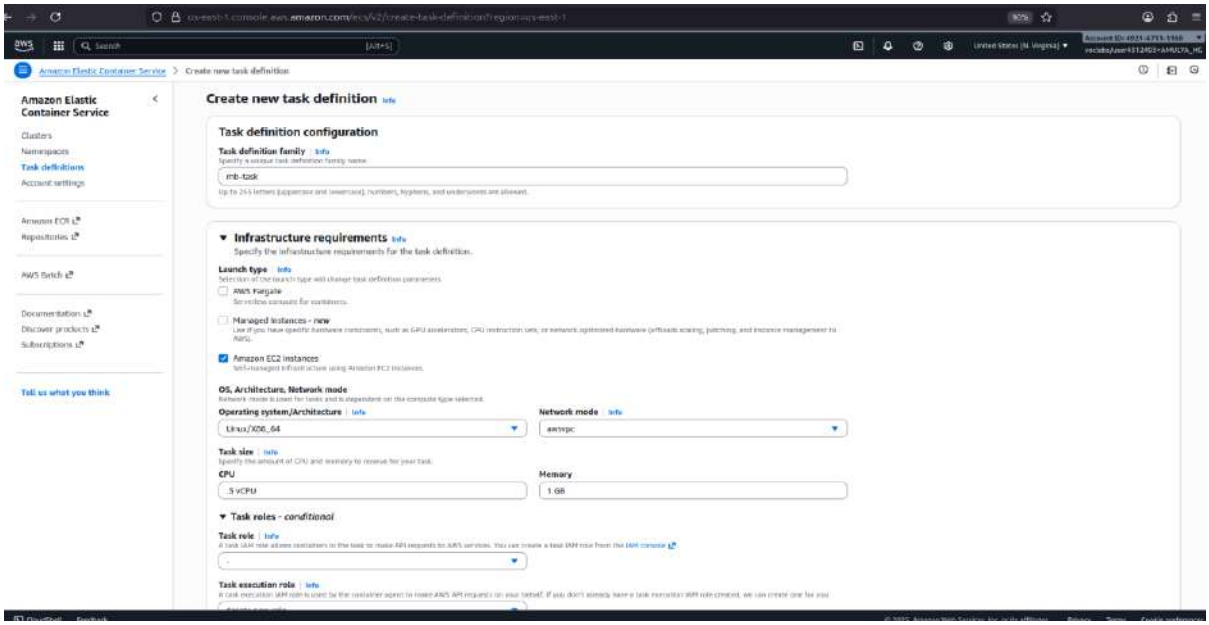
1.On the Amazon ECS console, from the left menu, choose Task definitions.

2. Choose Create new task definition

3. Task definition configuration section, for Task definition family, enter mb-task

4. In the Infrastructure requirements, select Amazon EC2 instances, and clear *AWS Fargate*.

5. For the Task size, enter CPU: .5vCPU, Memory: 1GB

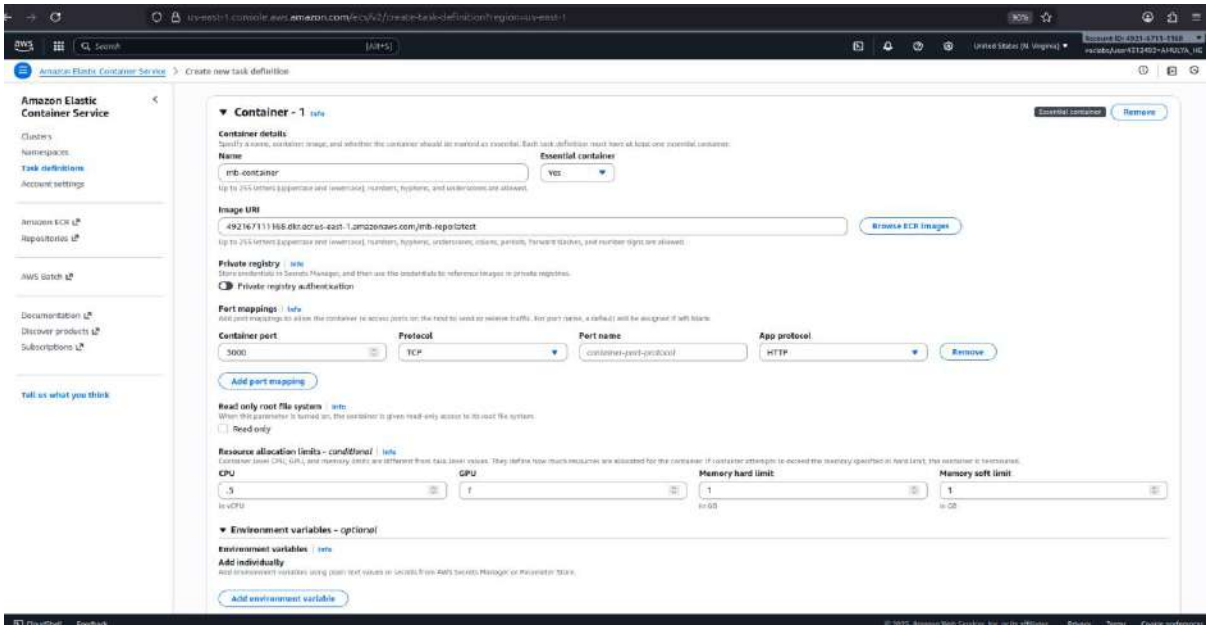



6. For Container details, for Name, enter mb-container

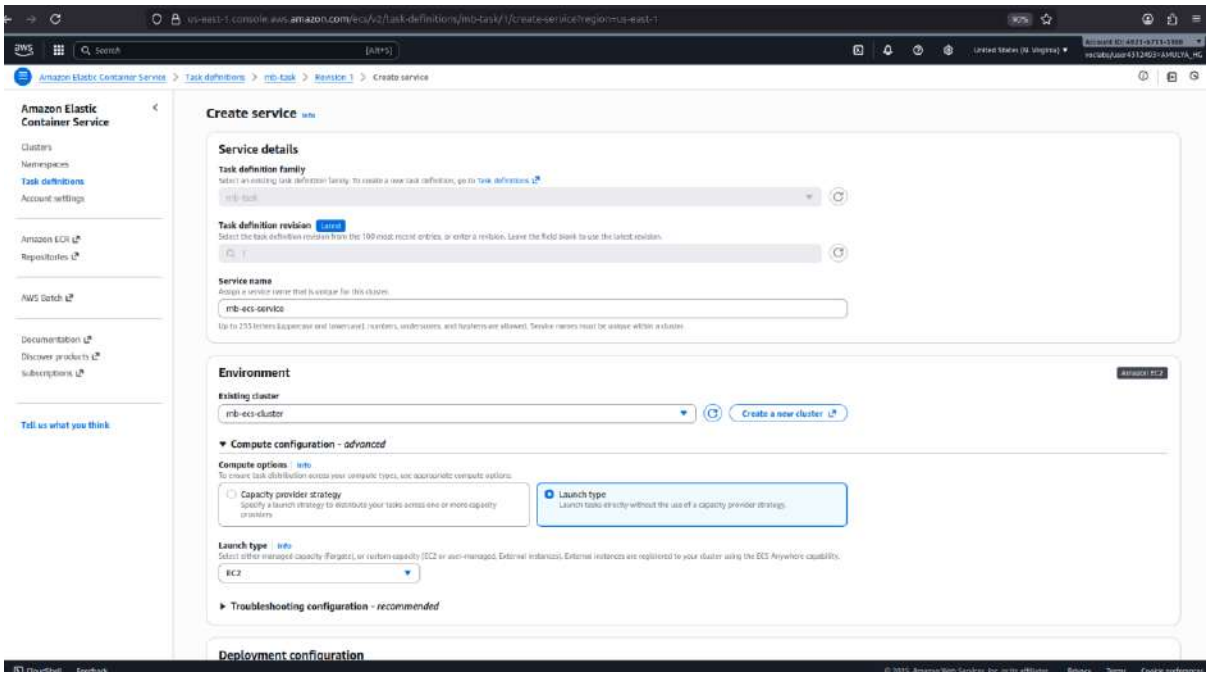
7. For Image URI, paste the URI of the user's container image that you copied to a text editor earlier.

8. For Port mappings, for Container port, enter 3000

9. Choose Create.

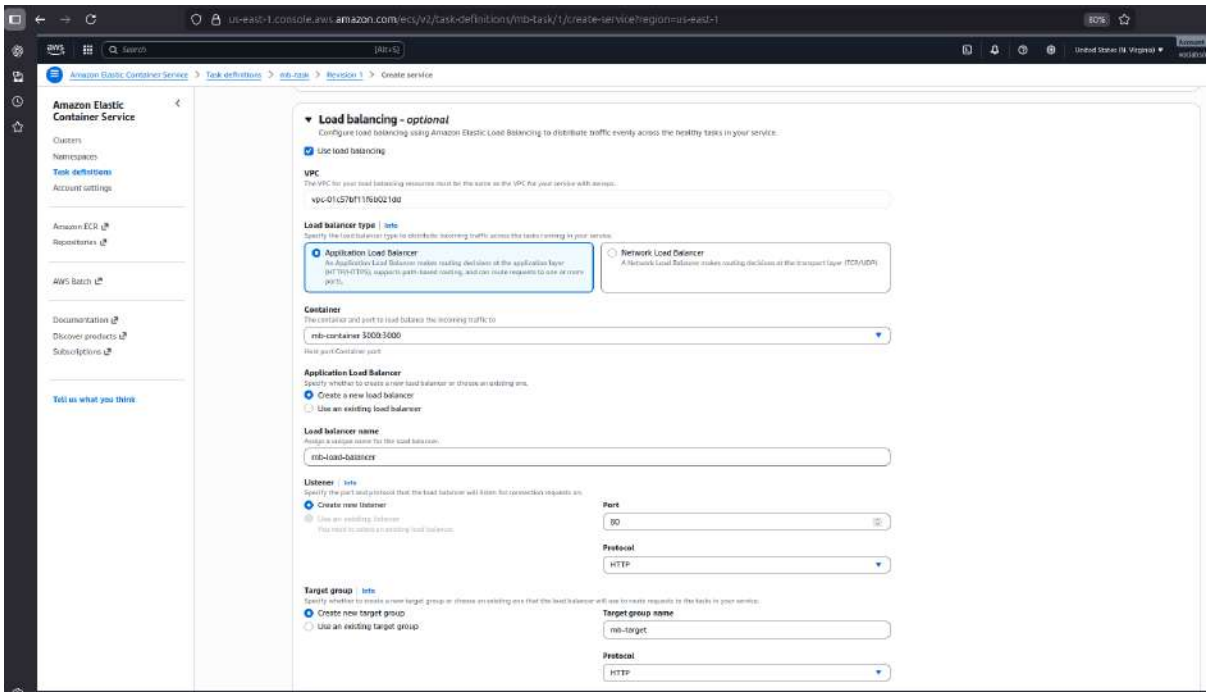



Step 3: Creating and deploying the service

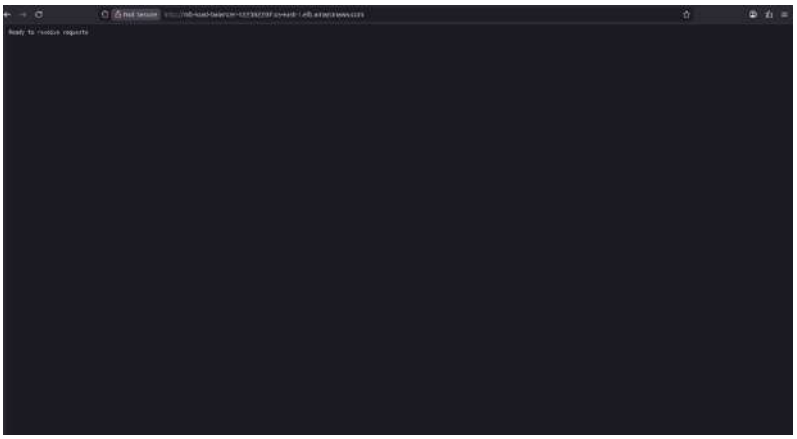



1. Select mb-task, choose Deploy then choose Create Service.
2. For Compute options, choose Launch type.
3. For Launch type, choose EC2.
4. For Service name choose mb-ecs-service.
5. For the Security group, choose Use an existing security group.

4. From the Security group name dropdown list, select the security group that has ECSSG in the name
5. Clear the default security group.
6. Check Use load balancing.
7. For Load balancer type, choose Application Load Balancer.
8. For Application Load Balancer, choose Create a new load balancer.
9. For Load balancer name, enter mb-load-balancer
10. For Target group name, enter mb-target.
11. Choose Create



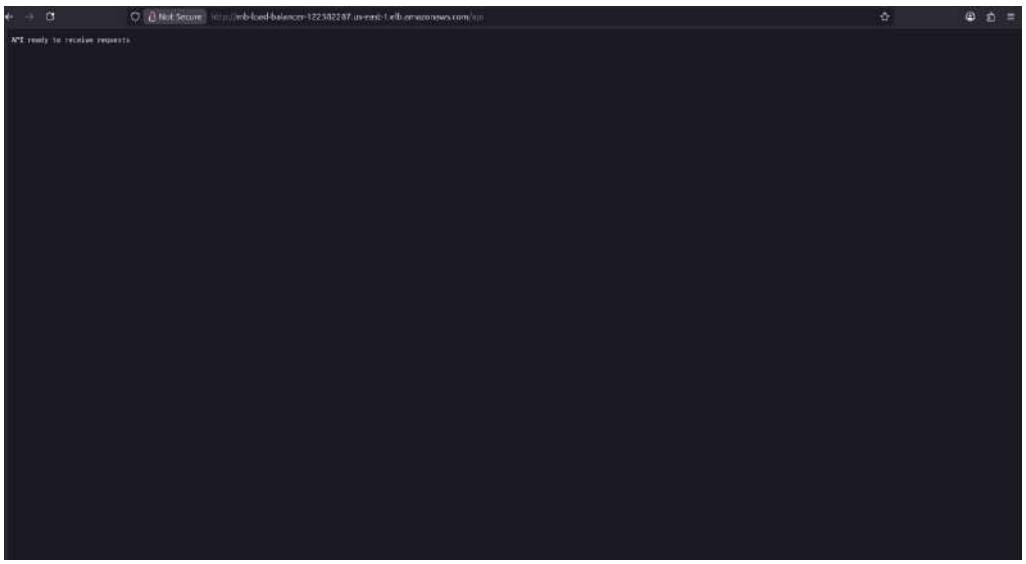

12. From the services list, choose mb-ecs-service
13. It should show a Status of *Active* with one Task that's Running
14. In the Load balancer health section choose load balancer.
15. Copy the DNS name for the load balancer, and paste it into a new browser tab

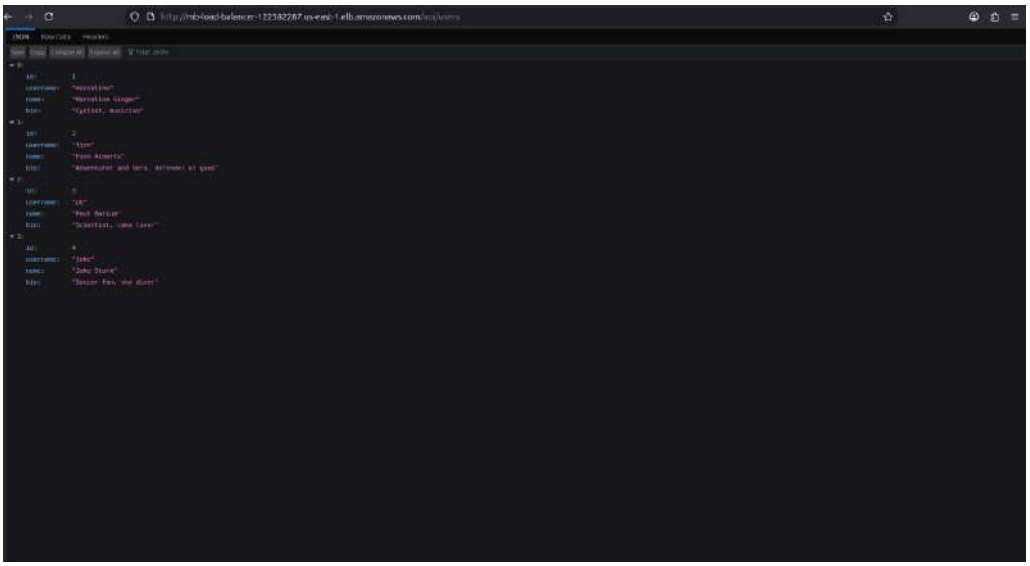


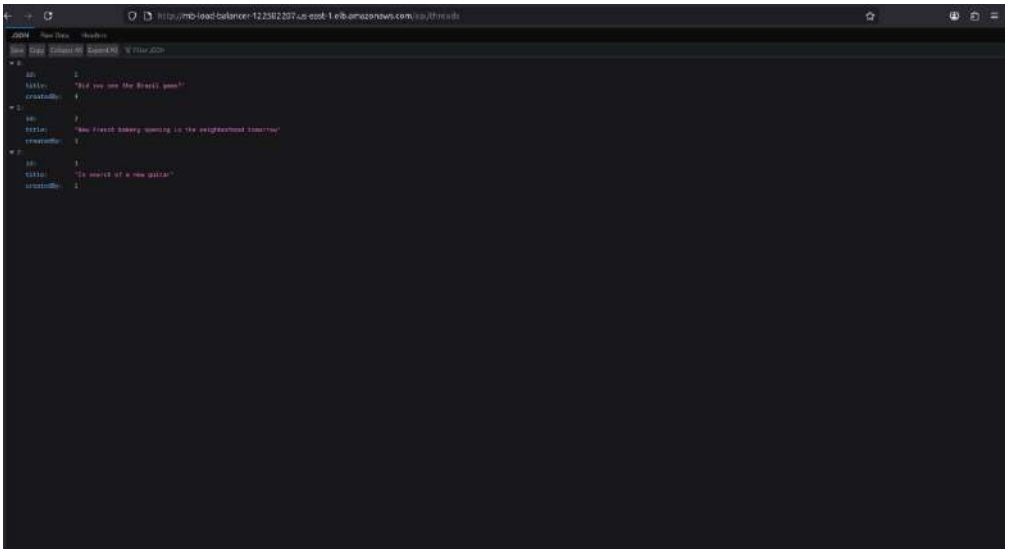
**Step 4:** Testing the containerized monolith

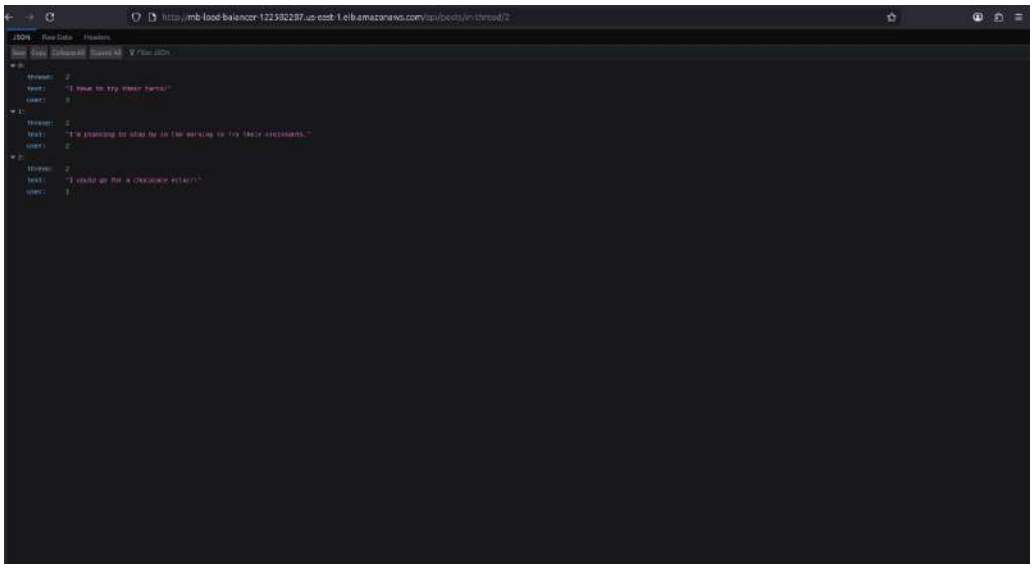
1. You need the DNS name of the load balancer that you used in the previous steps.
2. Open a new browser tab, paste the DNS name into the address field, and press Enter.
3. Enter the following addresses in the browser tab and examine the results. For each address, replace *DNS name* with the DNS name from the previous steps

- *DNS name*/api
- *DNS name*/api/users
- *DNS name*/api/threads
- *DNS name*/api/posts/in-thread/2





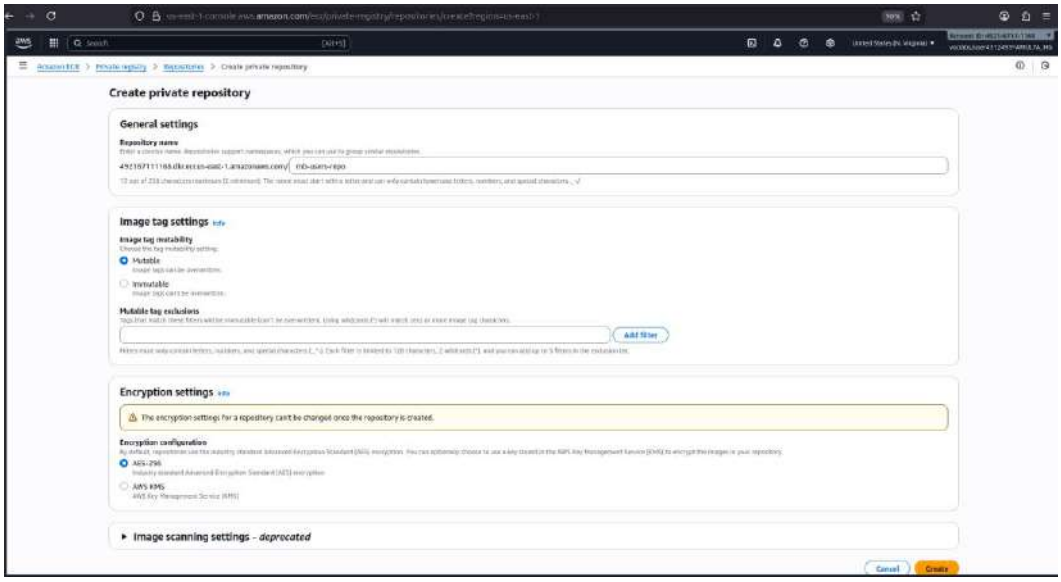




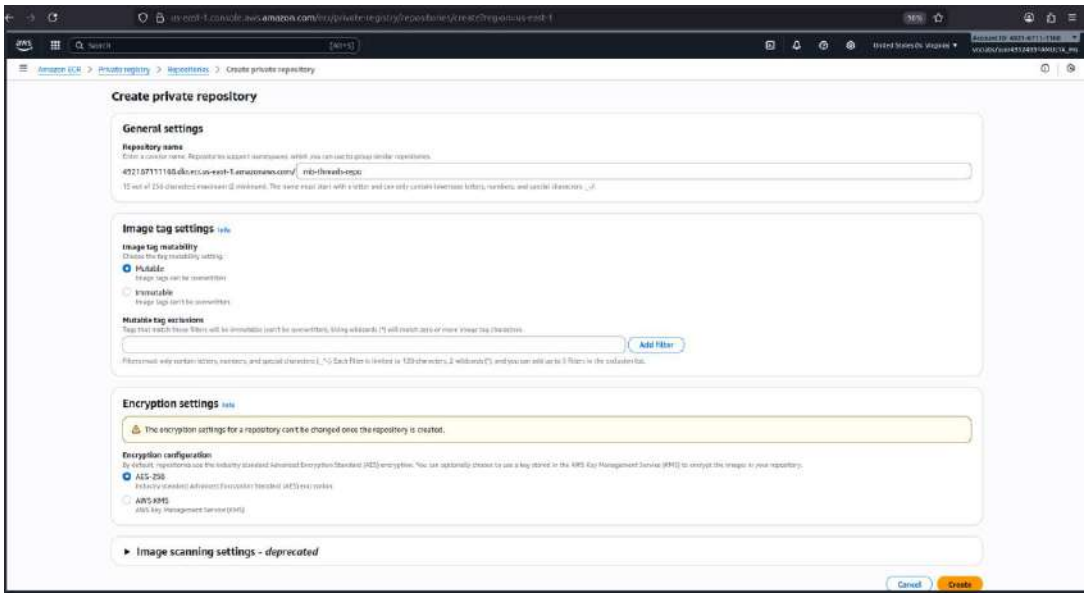
**Task 5: Refactoring the monolith**

**Step1:** Provisioning an ECR repository for each microservice

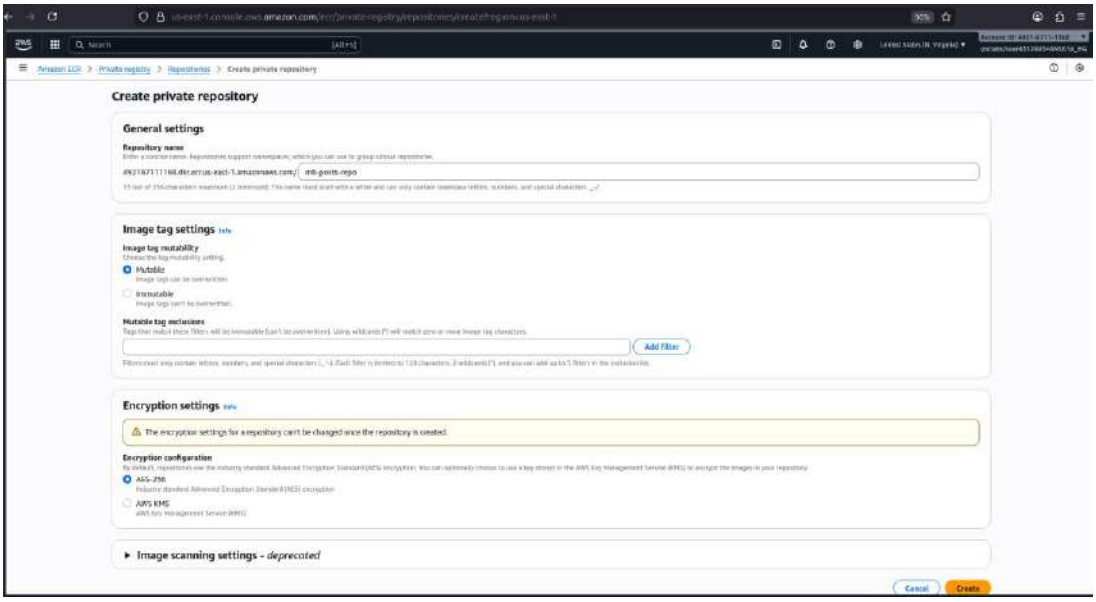
1. On the AWS Management Console, in the search box, enter and choose Elastic Container Registry
2. Choose Repositories, then choose Create repository.
3. For Repository name, enter mb-users-repo
4. Choose Create

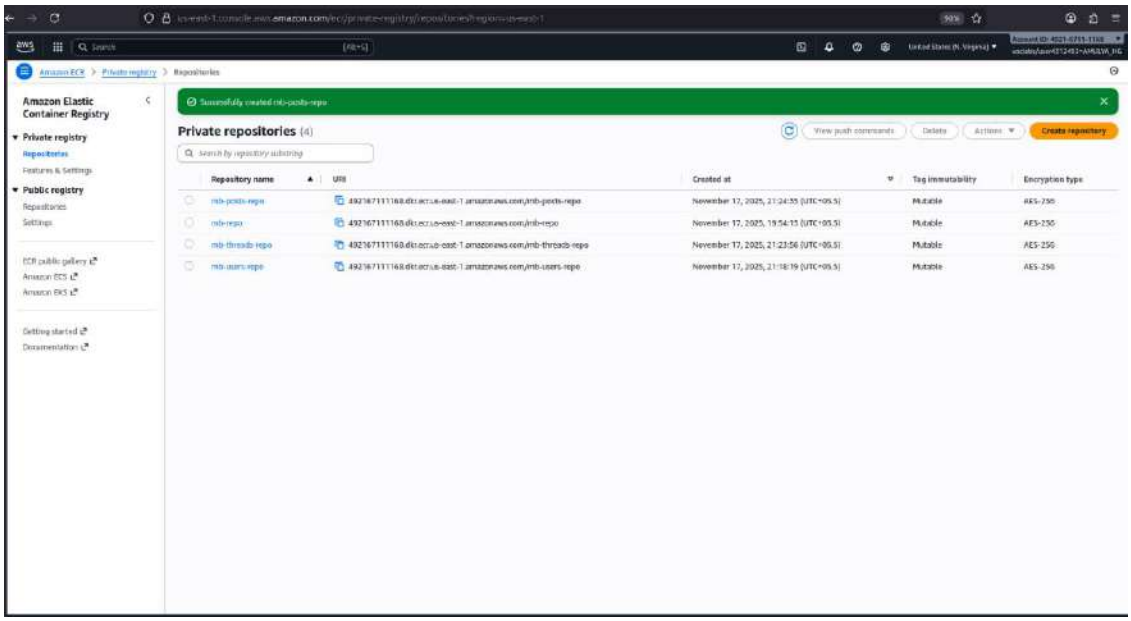



5. Repeat the steps in this sub-task to create a repository named mb-threads-repo




6. Repeat the steps in this sub-task to create a repository named mb-posts-repo




**Step 2:** Building and pushing the image for the users microservice

1. Switch to the Lab IDE browser tab
2. In the terminal tab, to change directory to the 3-containerized-microservices/ users' folder

```
cd ~/environment/3-containerized-microservices/users
```

3.Switch to the Elastic Container Registry browser tab.

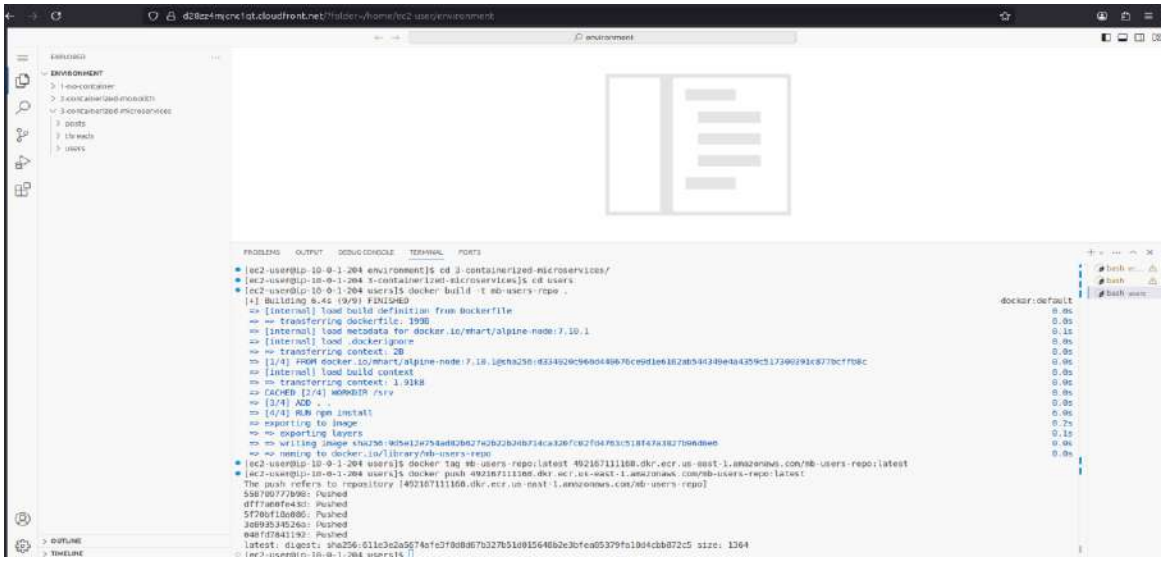
4.From the Private repositories list, choose mb-users-repo.

5.At the top of the page, choose View push commands

6.A pop-up window titled Push commands for mb-users-repo opens.

7. Enter the following commands

- docker build -t mb-users-repo
- docker tag mb-users-repo:latest 1234567890.dkr.ecr.us-east-1.amazonaws.com/mb-users-repo:latest
- docker push 1234567890.dkr.ecr.us-east-1.amazonaws.com/mb-users-repo:latest



8. Switch to Elastic Container Registry select mb-users-repo

9. In the Image list you can see the container image that you pushed.

10. In the image list, choose copy URI for the image URI.

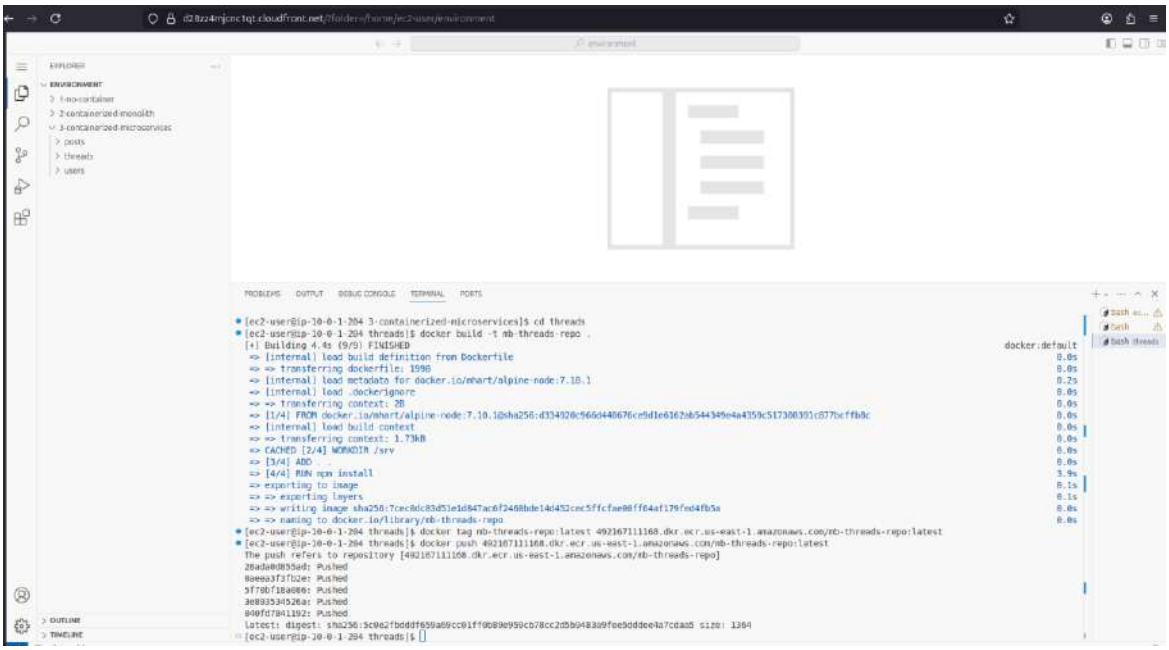
**Step 3:** Building and pushing the image for the threads microservice

1.In the terminal tab, change directory to the 3-containerized-microservices/threads folder.

cd ~/environment/3-containerized-microservices/threads

2. Switch to the Elastic Container Registry.
3. Choose Repositories, and choose mb-threads-repo
4. At the top of the page, choose View push commands
5. A pop-up window titled Push commands for mb-threads-repo opens.
6. Repeat the steps from the previous task to do the following

- Build the Docker image for the microservice
- Tag the image with the repository URI so that it can be pushed to the repository
- Push the container image to the microservice's repository



7.Switch to Elastic Container Registry select mb-threads-repo
8.In the image list you can see the container image that you pushed
9.In the image list, choose copy URI for the image URI

**Step 3:** Building and pushing the image for the posts microservice

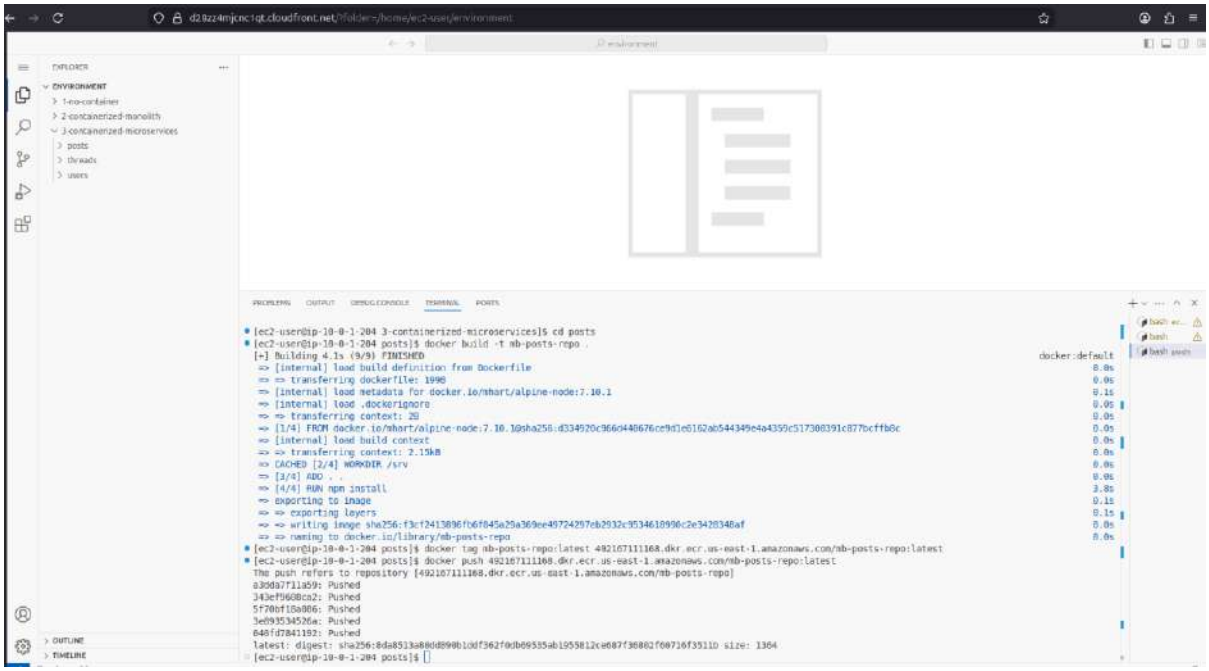
1. In the terminal tab, change directory to the 3-containerized-microservices/posts folder.

cd ~/environment/3-containerized-microservices/posts

2. Switch to the Elastic Container Registry.
3. Choose Repositories, and choose mb-posts-repo
4. At the top of the page, choose View push commands
5. A pop-up window titled Push commands for mb-posts-repo opens.
6. Repeat the steps from the previous task to do the following

- Build the Docker image for the microservice

- Tag the image with the repository URI so that it can be pushed to the repository
- Push the container image to the microservice's repository



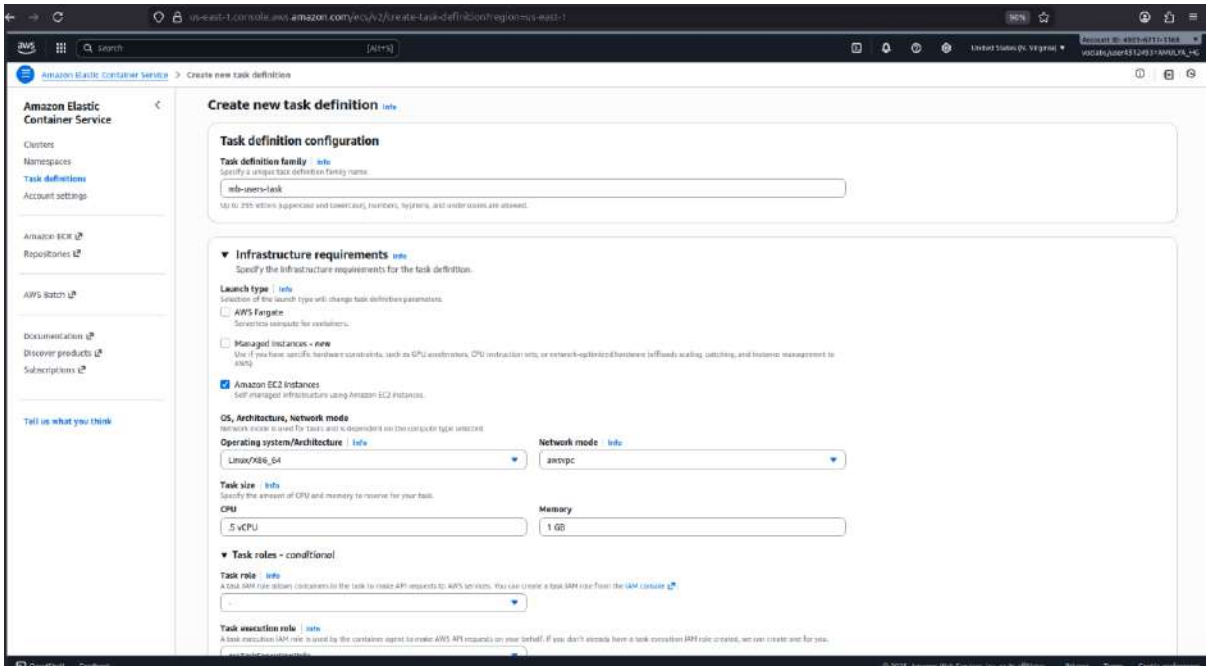
7. Switch to Elastic Container Registry select mb-posts-repo
8. In the image list you can see the container image that you pushed
9. In the image list, choose copy URI for the image URI

Task 6: Deploying the containerized microservices

Step 1: Creating a task definition for the users microservice

1. On the AWS Management Console, in the search box, enter and select Elastic Container Service.
2. Choose Task Definition
3. Choose Create new Task definition
4. In the Task definition configuration section, for Task definition family, enter mb-users-task
5. In the Infrastructure requirements, select Amazon EC2 instances, and clear AWS Fargate
6. For the Task size, choose CPU: .5 vCPU, Memory: 1GB
7. In the Container - 1 section, configure the following options:
8. For Container details, for Name, enter mb-users-container
9. For Image URI, paste the URI of the users container image that you copied to a text editor earlier.
10. For Port mappings, for Container port, enter 3000.

11. Choose Create.




Step 2: Creating a task definition for the posts microservice

1. In the left navigation pane, choose Task definitions.
2. Choose Create new task definition, and configure the following options:
3. In the Task definition configuration section, for Task definition family, enter mb-posts-task
    a. In the Infrastructure requirements, select Amazon EC2 instances, and clear AWS Fargate.
    b. For the Task size, choose CPU: .5 vCPU, Memory: 1GB
    c. For Container - 1, configure the following options:
    d. For Container details, for Name, enter mb-posts-container
    e. For Image URI, paste the URI of the posts container image that you copied to a text editor earlier.
    f. For Port mappings, for Container port, enter 3000. This option specifies the port on which the container receives requests.
4. Choose Create.
5. A message is displayed at the top that says, "Task definition successfully created."

Step 3: Creating a task definition for the threads microservice

1. In the left navigation pane, choose Task definitions.

2. Choose Create new task definition, and configure the following options:

3. In the Task definition configuration section, for Task definition family, enter mb-threads-task

- In the Infrastructure requirements, select Amazon EC2 instances, and clear AWS Fargate.
- For the Task size, choose CPU: .5 vCPU, Memory: 1GB
- For Container - 1, configure the following options:
- For Container details, for Name, enter mb-threads-container
- For Image URI, paste the URI of the threads container image that you copied to a text editor earlier.
- For Port mappings, for Container port, enter 3000.

4. Choose Create.

Task 7: Creating a task definition for each microservice

Step1: Creating a task definition for the users microservice

1. On the AWS Management Console, in the search box, enter and select Elastic Container Service
2. In the left navigation pane, choose Task definitions.
3. Choose Create new task definition, and configure the following options:
    a. In the Task definition configuration section, for Task definition family, enter mb-users-task
    b. In the Infrastructure requirements, select Amazon EC2 instances, and clear AWS Fargate.
    c. For the Task size, choose CPU: .5 vCPU, Memory: 1GB
    d. In the Container - 1 section, configure the following options:
    e. For Container details, for Name, enter mb-users-container
    f. For Image URI, paste the URI of the users container image that you copied to a text editor earlier.
    g. For Port mappings, for Container port, enter 3000. This option specifies the port on which the container receives requests.
4. Choose Create.

Step 2: Creating a task definition for the posts microservice

1. In the left navigation pane, choose Task definitions.
2. Choose Create new task definition, and configure the following options:
   In the Task definition configuration section, for Task definition family, enter mb-posts-task
    - In the Infrastructure requirements, select Amazon EC2 instances, and clear AWS Fargate.
    - For the Task size, choose CPU: .5 vCPU, Memory: 1GB
    - For Container - 1, configure the following options:

- For Container details, for Name, enter mb-posts-container
- For Image URI, paste the URI of the posts container image that you copied to a text editor earlier.
- For Port mappings, for Container port, enter 3000. This option specifies the port on which the container receives requests.

3. Choose Create.

Step 3: Creating a task definition for the threads microservice

1. In the left navigation pane, choose Task definitions.
2. Choose Create new task definition, and configure the following options:

In the Task definition configuration section, for Task definition family, enter mb-threads-task

- In the Infrastructure requirements, select Amazon EC2 instances, and clear AWS Fargate.
- For the Task size, choose CPU: .5 vCPU, Memory: 1GB
- For Container - 1, configure the following options:
- For Container details, for Name, enter mb-threads-container
- For Image URI, paste the URI of the threads container image that you copied to a text editor earlier.
- For Port mappings, for Container port, enter 3000. This option specifies the port on which the container receives requests.

3. Choose Create.

Task 8: Creating and deploying the services

Step 1:

1. In the left navigation pane, choose Clusters, and choose your mb-ecs-cluster cluster.
2. On the Services tab, choose Create, and configure the following options:

- In the Environment section, configure the following options:
  - For Compute options, choose Launch type.
  - For Launch type, choose EC2.
- In the Deployment configuration section, configure the following options:
  - For Application type, choose Service.
  - For Family, chose mb-users-task.
  - For Service name, enter mb-users-service
- Expand the Networking section, and configure the following options:
  - For Security group, choose Use an existing security group.
  - From the Security group name dropdown list, select the security group that has ECSSG in the name.
  - Clear the default security group.
- Expand the Load balancing - optional section, and configure the following options:

- Check Use load balancing.
- For Load balancer type, choose Application Load Balancer.
- For Application Load Balancer, choose Use an existing load balancer.
- For Load balancer, choose mb-load-balancer.
- For Listener, choose Use an existing listener, and then choose 80:HTTP from the dropdown list.
- For Target group, choose Create new target group.
- For Target group name, enter mb-users-target
- For Path pattern, enter /api/users*
- For Evaluation order, enter 1

3. Choose Create.

Step 2:

1. Return to your mb-ecs-cluster cluster. On the Services tab, choose Create, and configure the following options:

- In the Environment section, configure the following options:
  - For Compute options, choose Launch type.
  - For Launch type, choose EC2.
- In the Deployment configuration section, configure the following options:
  - For Application type, choose Service.
  - For Family, choose mb-posts-task.
  - For Service Name, enter mb-posts-service
- Expand the Networking section, and configure the following options:
  - For Security group, choose Use an existing security group.
  - From the Security group name dropdown list, select the security group that has ECSSG in the name.
  - Clear the default security group.
- Expand the Load balancing section, and configure the following options:
  - Check Use load balancing.
  - For Load balancer type, choose Application Load Balancer.
  - For Application Load Balancer, choose Use an existing load balancer.
  - For Load balancer, choose mb-load-balancer.
  - For Listener, choose Use an existing listener, and then choose 80:HTTP from the dropdown list.
  - For Target group, choose Create new target group.
  - For Target group name, enter mb-posts-target
  - For Path pattern, enter /api/posts*
  - For Evaluation order, enter 2

2. Choose Create.

Step 3:

1. Return to your mb-ecs-cluster cluster. On the Services tab, choose Create, and configure the following options:

- In the Environment section, configure the following options:
    - For Compute options, choose Launch type.
    - For Launch type, choose EC2.
- In the Deployment configuration section, configure the following options:
    - For Application type, choose Service.
    - For Family, choose mb-threads-task.
    - For Service Name, enter mb-threads-service
- Expand the Networking section, and configure the following options:
    - For Security group, choose Use an existing security group.
    - From the Security group name dropdown list, select the security group that has ECSSG in the name.
    - Clear the default security group.
- Expand the Load balancing section, and configure the following options:
    - Check Use load balancing.
    - For Load balancer type, choose Application Load Balancer.
    - For Application Load Balancer, choose Use an existing load balancer.
    - For Load balancer, choose mb-load-balancer.
    - For Listener, choose Use an existing listener, and then choose 80:HTTP from the dropdown list.
    - For Target group, choose Create new target group.
    - For Target group name, enter mb-threads-target
    - For Path pattern, enter /api/threads*
    - For Evaluation order, enter 3

2. Choose Create.

Task 9: Validating the deployment

In the browser address bar, enter the following addresses in the browser tab, and examine the results.

- *DNS name*/api/users

Expected output: List of users

- *DNS name*/api/users/2

Expected output: Details of user 2

- *DNS name*/api/threads

Expected output: List of threads

- *DNS name*/api/posts/in-thread/2

Expected output: Details of thread 2

- *DNS name*/xxx

Expected output: Invalid request