

# 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

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

```
[ec2-user@ip-10-0-1-89 environment]$ aws configure get region
us-east-1
[ec2-user@ip-10-0-1-89 environment]$ aws cloudformation create-stack --stack-name CreateBucket --template-body file://S3.yaml
{
  "StackId": "arn:aws:cloudformation:us-east-1:248631246916:stack/CreateBucket/320e35a0-c7b7-11f0-ac8b-0affc9765659"
}
[ec2-user@ip-10-0-1-89 environment]$
```

## 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

Amazon S3

General purpose buckets

- Directory buckets
- Table buckets
- Vector buckets
- Access Grants
- Access Points (General Purpose Buckets, FSx file systems)
- Access Points (Directory Buckets)
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

▼ Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

Feature spotlight [ 1 ]

CloudShell Feedback

Objects (3)

Name	Type	Last modified	Size	Storage class
S3.yaml	yaml	November 22, 2025, 21:03:18 (UTC+05:30)	119.0 B	Standard
static-website.zip	zip	November 22, 2025, 21:03:18 (UTC+05:30)	21.7 MB	Standard
static/	Folder	-	-	-

Actions ▾ Create folder Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

EXPLORER

ENVIRONMENT

S3.yaml

```

! S3.yaml
 1 AWSTemplateFormatVersion: "2010-09-09"
 2 Description: "cafe S3 template"
 3
 4 Resources:
 5   S3Bucket:
 6     Type: AWS::S3::Bucket
 7     Properties:
 8       WebsiteConfiguration:
 9         IndexDocument: index.html
10         DeletionPolicy: Retain
11
12 Outputs:
13   WebsiteURL:
14     Description: "URL for static website hosted in S3"
15     Value: !Sub "http://${S3Bucket}.s3-website-${AWS::Region}.amazonaws.com"
16

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

[ec2-user@ip-10-0-1-89 environment]$ aws cloudformation update-stack --stack-name CreateBucket --template-body file://S3.yaml
An error occurred (ValidationError) when calling the UpdateStack operation: No updates are to be performed.
[ec2-user@ip-10-0-1-89 environment]$ aws cloudformation validate-template --template-body file://S3.yaml
{
  "Parameters": [],
  "Description": "cafe S3 template"
}
[ec2-user@ip-10-0-1-89 environment]$ aws cloudformation update-stack --stack-name CreateBucket --template-body file://S3.yaml
{
  "StackId": "arn:aws:cloudformation:us-east-1:248631246916:stack/CreateBucket/320e35a0-c7b7-11f0-ac8b-0affc9765659"
}
[ec2-user@ip-10-0-1-89 environment]$ 

```

Ln 16, Col 1 Spaces: 2 UTF-B LF YAML Layout: U.S.

## 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

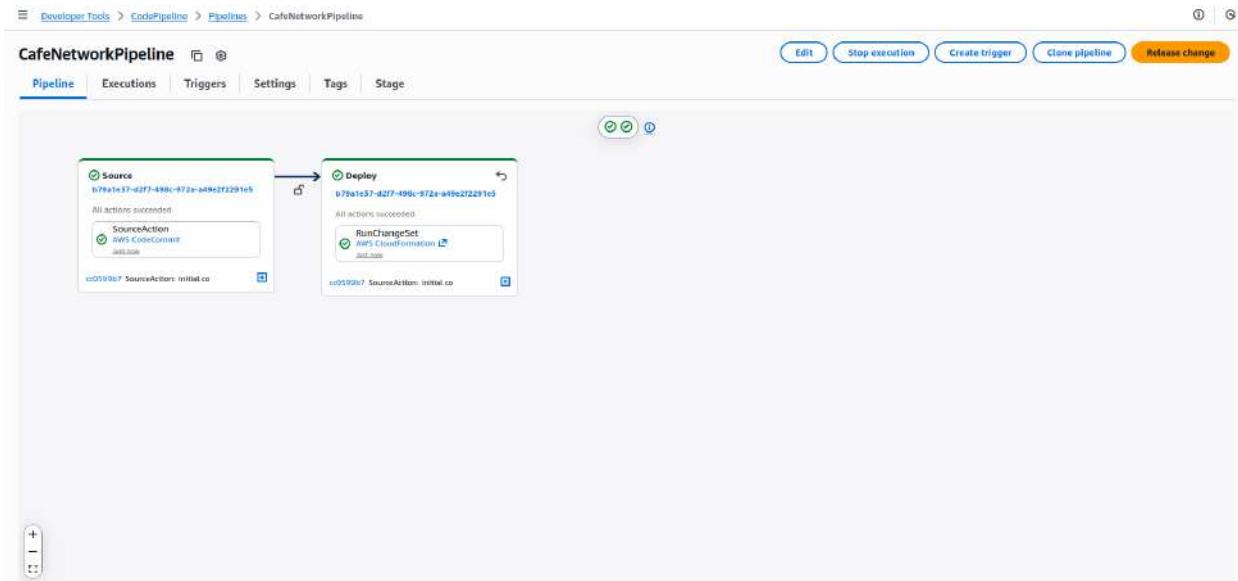
2. Edit description: *Network layer for the cafe*

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.

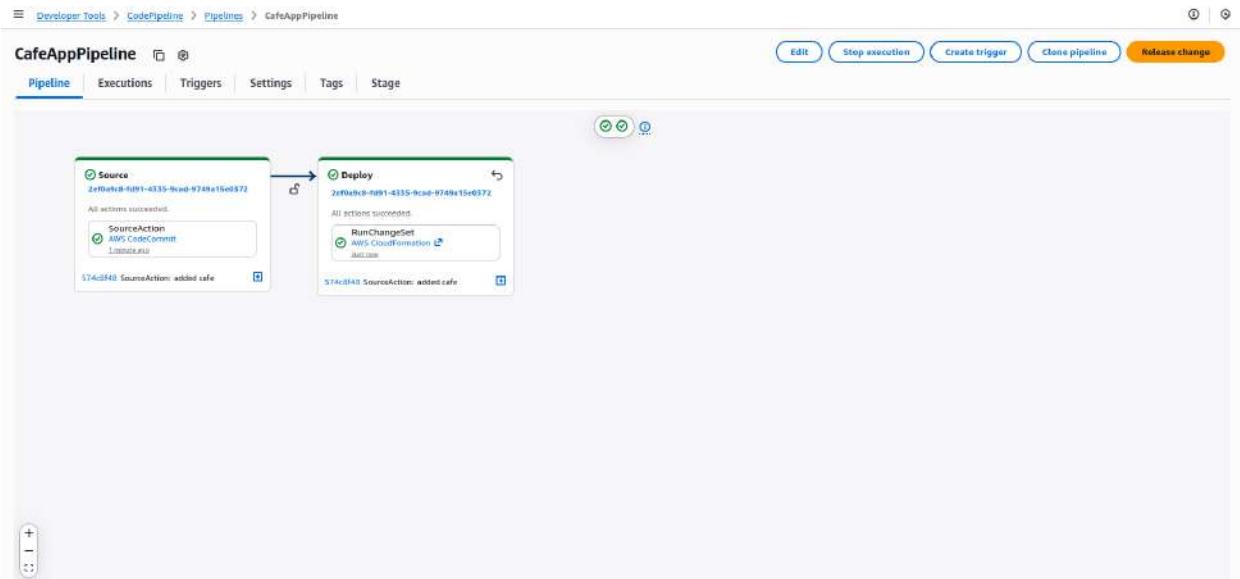
## Task 7 : Create Application EC2 Stack via CI/CD

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

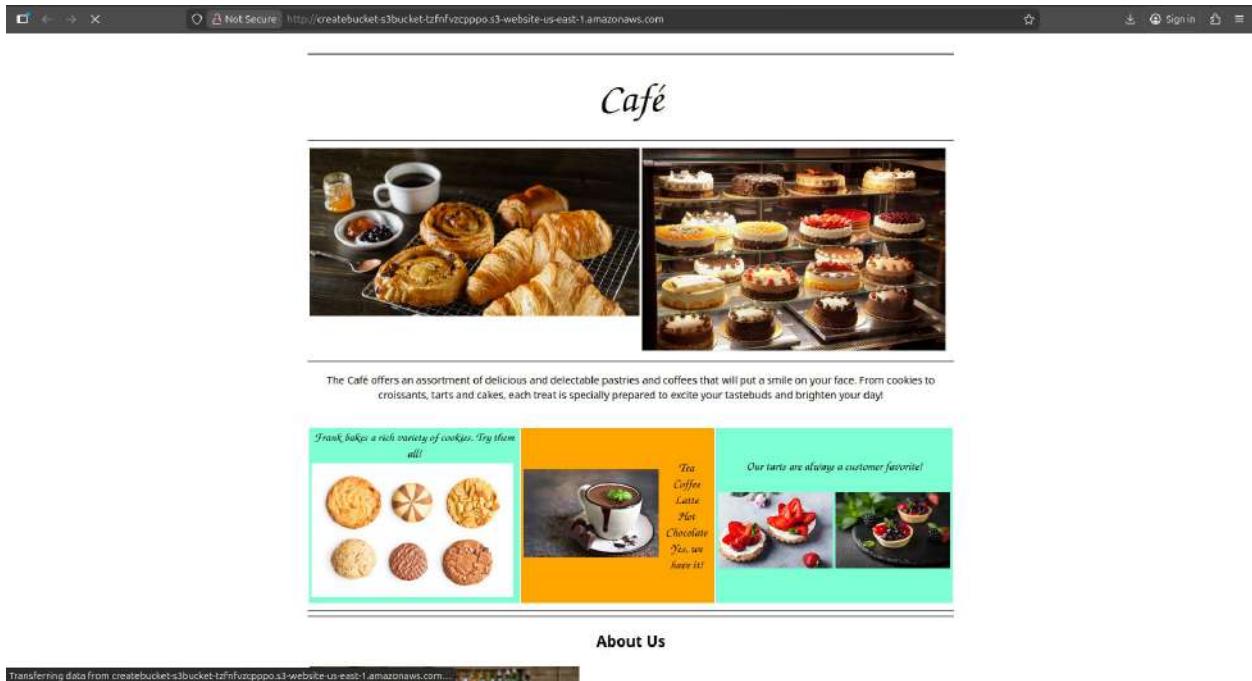
6. Verify EC2 instance and test site:

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



The screenshot shows the AWS CodePipeline console with the 'Pipelines' section. It lists two pipelines: 'CafeAppPipeline' and 'CafeNetworkPipeline'. Both pipelines are in a 'Succeeded' state. The 'CafeAppPipeline' was triggered by 'SourceAction - 574c0f40: added cafe app' and started 1 minute ago. The 'CafeNetworkPipeline' was triggered by 'SourceAction - 574c0f40: added cafe app' and also started 1 minute ago. The left sidebar provides navigation links for various AWS services like CodeCommit, CodeArtifact, CodeBuild, CodeDeploy, and CodePipeline.

Name	Latest execution status	Latest source revisions	Latest execution started	Most recent executions
CafeAppPipeline	Succeeded	SourceAction - 574c0f40: added cafe app	1 minute ago	<span>Success</span> <span>Failure</span> <span>Failure</span> <span>Failure</span> View details
CafeNetworkPipeline	Succeeded	SourceAction - 574c0f40: added cafe app	1 minute ago	<span>Success</span> <span>Success</span> <span>Failure</span> View details



## 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**

**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

Name	Security group ID	Security group name	VPC ID	Description	Owner
-	sg-0965eae07f5f1ee2	default	vpc-0ad269065da4a0ed1	default VPC security group	200259598426
-	sg-0bb7c9844fe998d5	default	vpc-0b28d59cbfb914bac	default VPC security group	200259598426

**Select a security group**

**Snapshots (1) [Info](#)**

Last updated less than a minute ago

Owned by me  Search

Name	Snapshot ID	Full snapshot size	Volume size	Description	Storage tier	Snapshot status	Started	Progress
Web Data	snsp-08113e071db190d05	0 B	100 GB	-	Standard	Completed	2025/11/23 20:34 GMT+5...	100%

**Select a snapshot above.**

**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

CloudFormation > Stacks > Create stack

Step 1 **Create stack**

Step 2

Step 3

Step 4

Step 5 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://c174142a450859611246304211w008199202797-repobucket-ntv1zz4x6v5r5.us-east-1.amazonaws.com/cafe-app.yaml>

S3 URL: <https://c174142a450859611246304211w008199202797-repobucket-ntv1zz4x6v5r5.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.

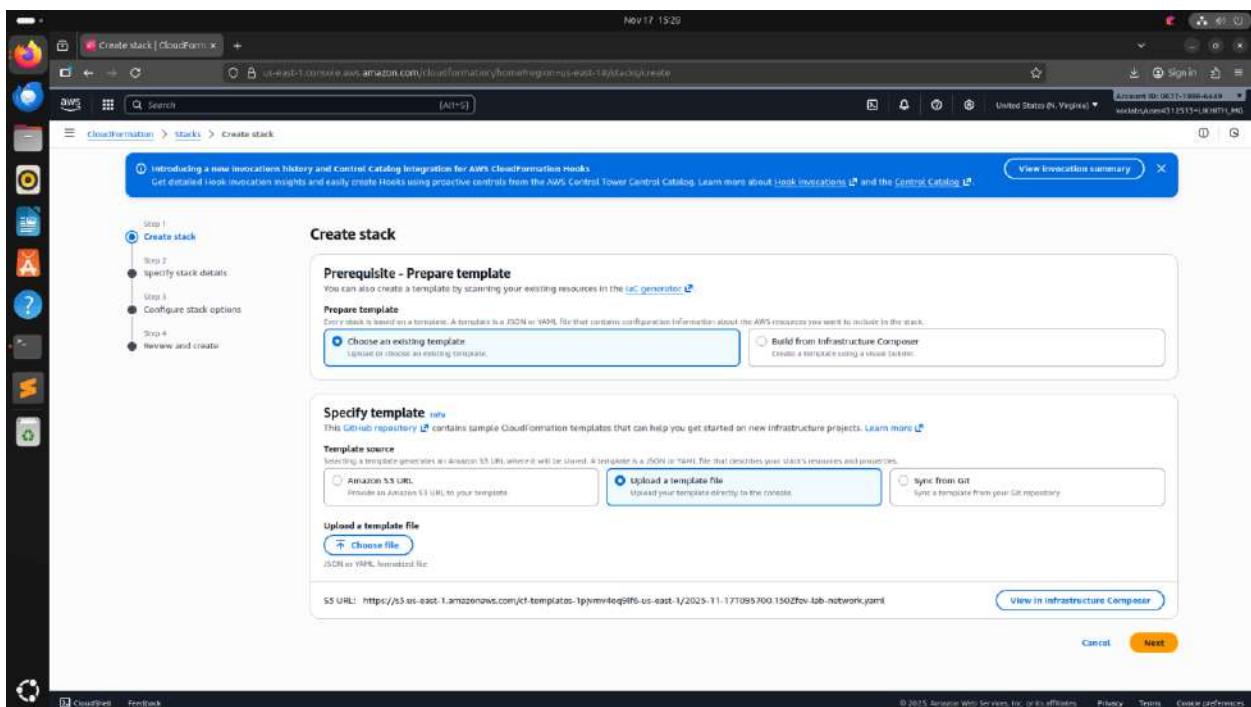
Choose Refresh every 15 seconds to update the display, if necessary.

You can now examine the resources that were created.

Choose the Resources tab.

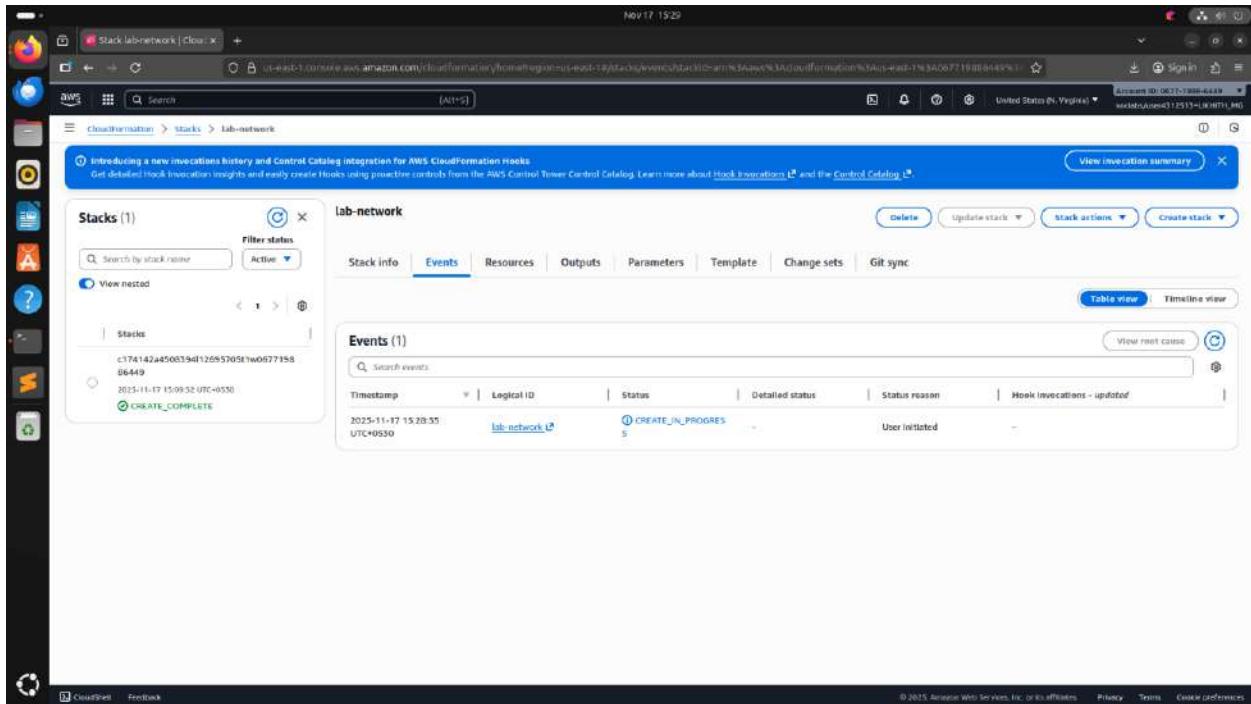
You see a list of the resources that were created by the template.

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



The screenshot shows the 'Specify stack details' step of the CloudFormation wizard. The left sidebar lists steps: Step 1 (Create stack), Step 2 (Specify stack details, which is selected and highlighted in blue), Step 3 (Configure stack options), and Step 4 (Review and create). The main area has two sections: 'Provide a stack name' and 'Parameters'. In 'Provide a stack name', the stack name is set to 'lab-network'. A note below says: 'Stack name must contain valid letters (a-z, A-Z), numbers (0-9), and hyphens (-) and start with a letter. Max 128 characters. Character count: 11/128.' Below this is a 'Parameters' section with a note: 'Parameters are defined in your template and allow you to input custom values when you create or update a stack.' It shows 'No parameters' and 'There are no parameters defined in your template.' At the bottom are 'Cancel', 'Previous', and 'Next' buttons.

The screenshot shows the 'Configure stack options' step of the CloudFormation wizard. The left sidebar lists steps: Step 1 (Create stack), Step 2 (Specify stack details), Step 3 (Configure stack options, which is selected and highlighted in blue), and Step 4 (Review and create). A screenshot overlay titled 'Screenshot - Just now' is displayed, showing a message: 'Screenshot captured. You can paste the image from the clipboard.' The main area has three sections: 'Tags - optional', 'Permissions - optional', and 'Stack failure options'. In 'Tags - optional', there is a key-value pair: 'Key' is 'application' and 'Value' is 'inventory'. In 'Permissions - optional', there is an 'IAM role - optional' section with a note: 'Specify an existing AWS Identity and Access Management (IAM) service role that CloudFormation can assume.' It shows an IAM role named 'Samurai-role-name'. In 'Stack failure options', there is a 'Behavior on provisioning failure' section with notes for 'Roll back all stack resources' (selected) and 'Preserve successfully provisioned resources'. At the bottom is a 'Delete newly created resources during a rollback' section. The bottom of the screen includes standard AWS navigation links: CloudFront, Feedback, Sign in, Account ID: 062718884449, United States (N. Virginia), and Copyright information: © 2015 Amazon Web Services, Inc. or its affiliates. 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.

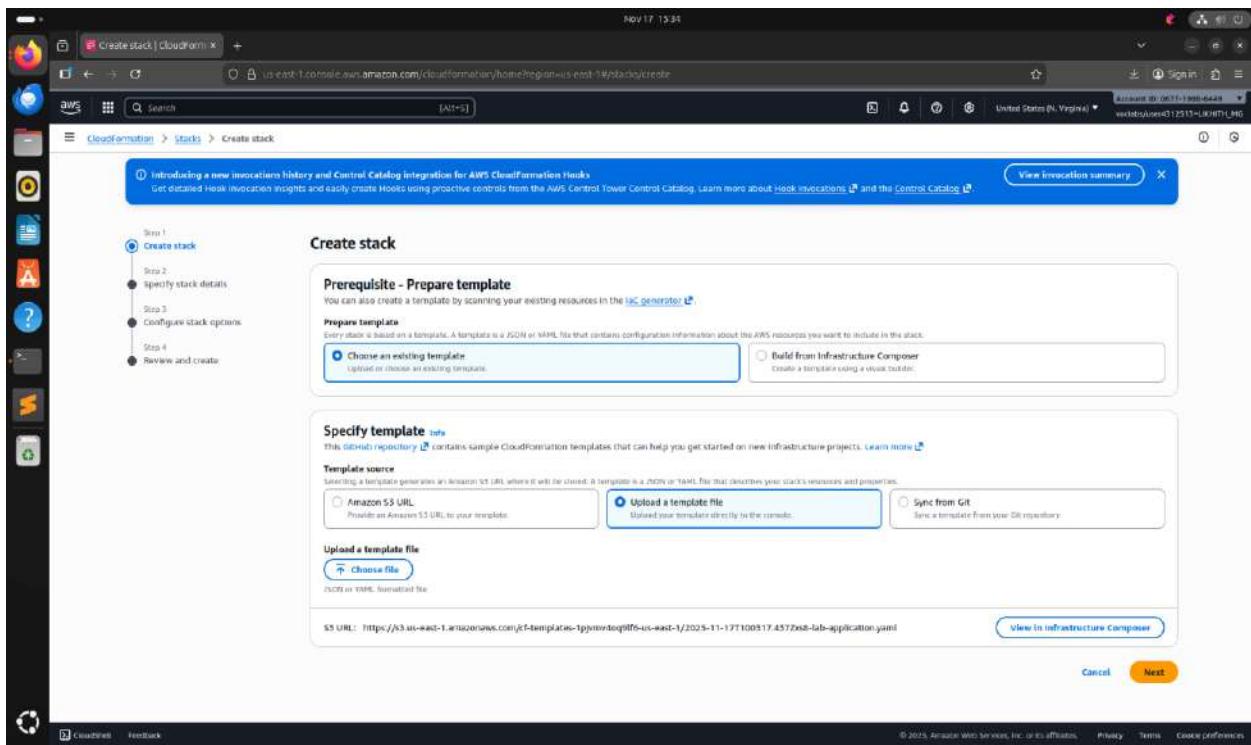
While the stack is being created, examine the details in the Events tab and the Resources tab. You can monitor the progress of the resource-creation process and the resource status.

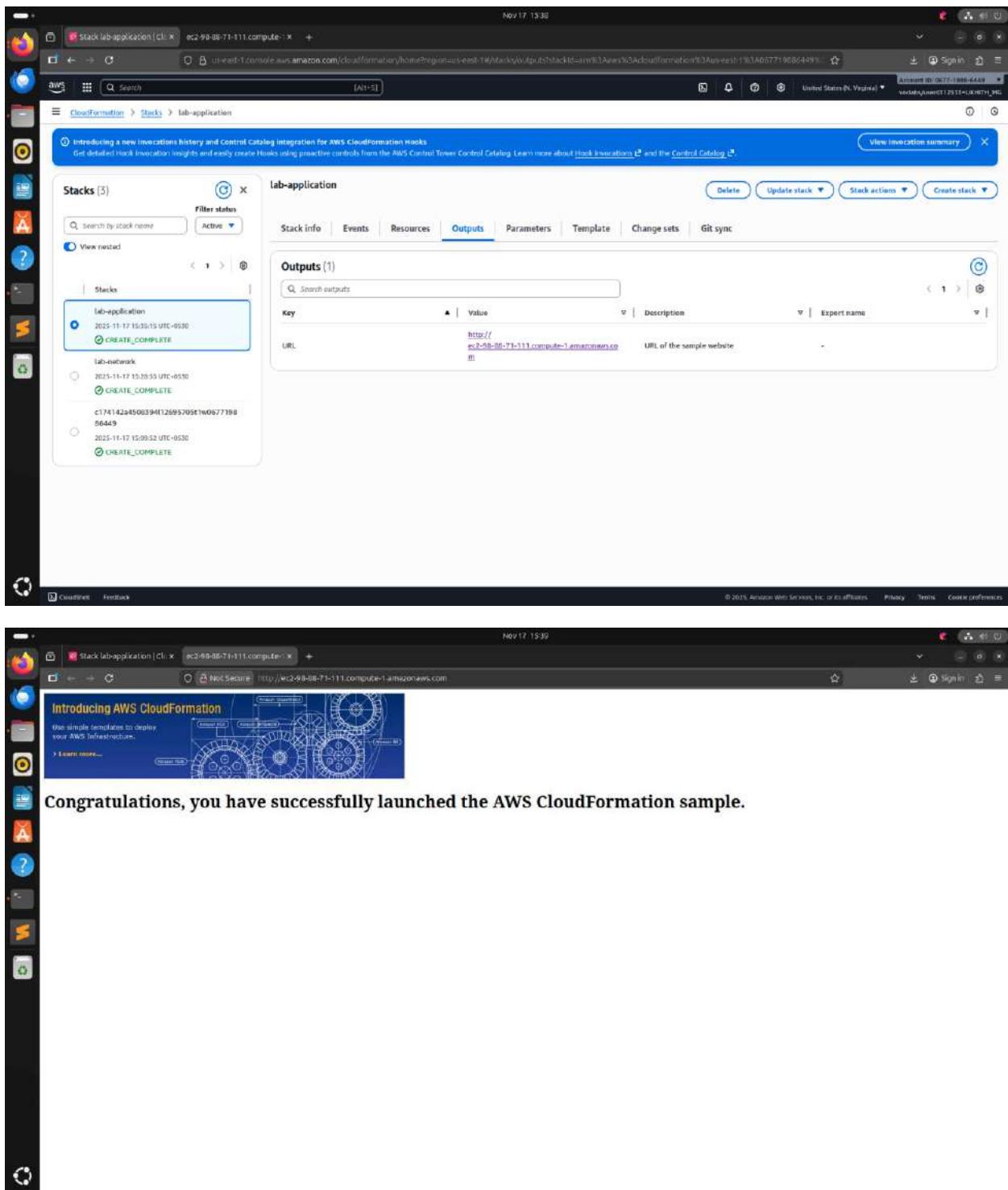
In the Stack info tab, wait for the Status to change to CREATE\_COMPLETE.

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.





The screenshot shows the AWS CloudFormation console with the 'lab-application' stack selected. The 'Outputs' tab is active, displaying a single output named 'URL' with the value 'http://ec2-98-88-71-111.compute-1.amazonaws.com'. The browser address bar shows the URL 'http://ec2-98-88-71-111.compute-1.amazonaws.com'.

## 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.

The screenshot shows the AWS CloudFormation console with the following details:

**Left Navigation Pane:** Shows the AWS Services menu with the "Network & Security" section expanded, specifically the "Security Groups" option.

**Top Bar:** Shows the URL as "Security groups | EC2 | us-east-1 | CloudFormation | us-east-1 | sg-0335e2c0fc308c976 - lab-application-WebServerSecurityGroup-WkZElY71nzV".

**Table View:** A table titled "Security Groups (1/3) [info]" showing three entries:

Name	Security group ID	Security group name	VPC ID	Description	Owner
sg-0f2a52688f2f52079b0	default	vpc-0f2e4f1d5910f2b93	vpc-0f2e4f1d5910f2b93	default VPC security group	067719896449
10-0fb92bd47710f30468	default	vpc-0f2e4f1d5910f2b93	vpc-0f2e4f1d5910f2b93	default VPC security group	067719896449
<b>Web Server Security...</b>	<b>sg-0335e2c0fc308c976</b>	<b>lab-application-WebServerSecurityGroup...</b>	<b>vpc-0f2e4f1d5910f2b93</b>	<b>Enable HTTP ingress</b>	<b>067719896449</b>

**Selected Security Group:** sg-0335e2c0fc308c976 - lab-application-WebServerSecurityGroup-WkZElY71nzV

**Inbound Rules Tab:** Shows the "Inbound rules" tab with one rule listed:

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
	sgn-091bf52c4bf447537	IPv4	HTTP	TCP	80	0.0.0.0/0	

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.

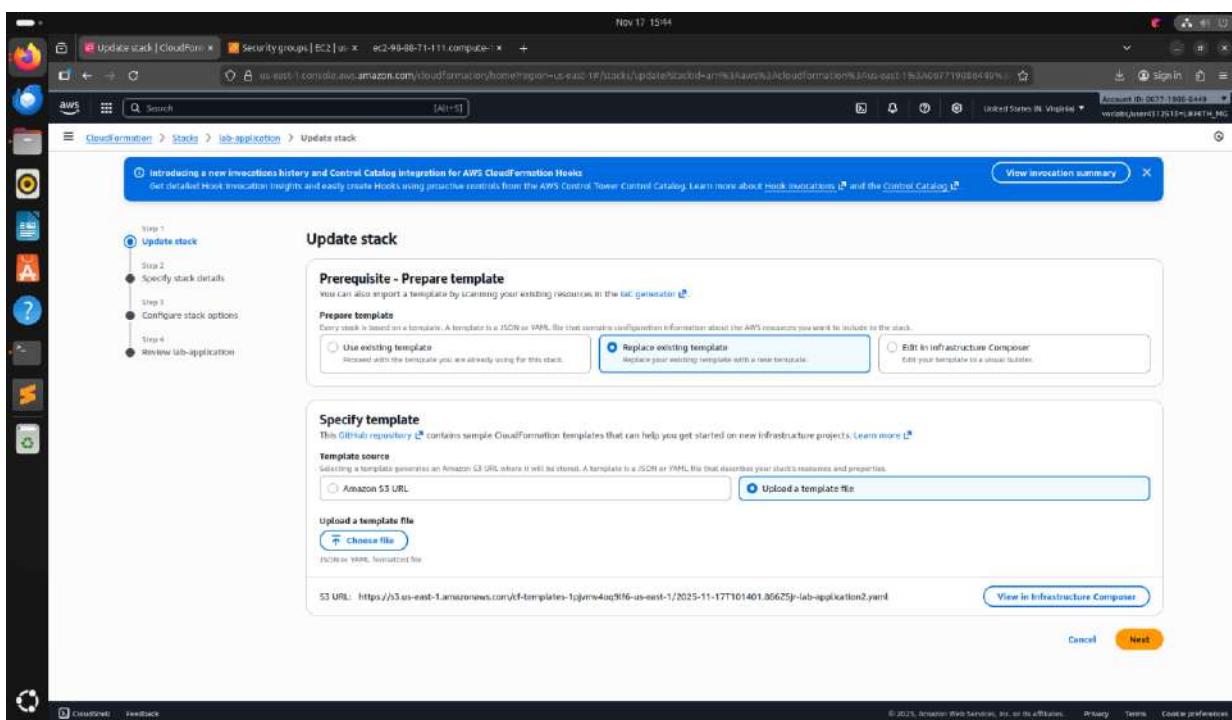
Update the status by choosing Refresh every 15 seconds, if necessary.

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.



The screenshot shows the AWS CloudFormation console with the 'lab-application' stack selected. The 'Events' tab is active, showing a list of 19 events. The events include:

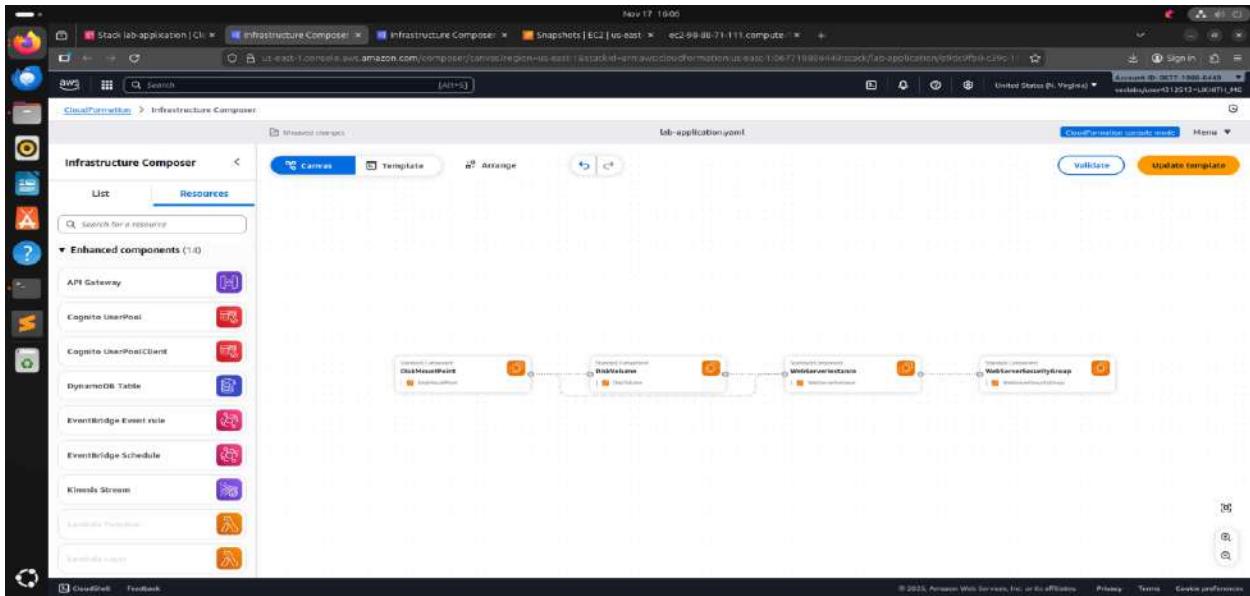
- 2025-11-17 19:44:55 UTC+0530: lab-application UPDATE\_IN\_PROGRESS
- 2025-11-17 19:50:52 UTC+0530: lab-application CREATE\_COMPLETE
- c174142a5400394f126957091w067719806449: DiskMountPoint CREATE\_COMPLETE
- 2025-11-17 19:50:49 UTC+0530: lab-application CREATE\_IN\_PROGRESS
- 2025-11-17 19:50:44 UTC+0530: DiskMountPoint CREATE\_IN\_PROGRESS
- 2025-11-17 19:50:44 UTC+0530: DiskMountPoint CONFIGURATION\_COMPLETE
- 2025-11-17 19:50:43 UTC+0530: DiskMountPoint CREATE\_IN\_PROGRESS
- 2025-11-17 19:50:43 UTC+0530: DiskMountPoint CONFIGURATION\_COMPLETE
- 2025-11-17 19:50:41 UTC+0530: DiskMountPoint CREATE\_IN\_PROGRESS

The screenshot shows the AWS Infrastructure Composer console under the 'Security Groups' section. The 'Web Server Security' group is selected, showing its inbound rules:

Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
-	sgr-091bfb2c0bf87517	IPv4	HTTP	TCP	80	0.0.0.0/0	-
-	sgr-099eb0bd442b27c3	IPv4	SSH	TCP	22	0.0.0.0/0	-

## 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.

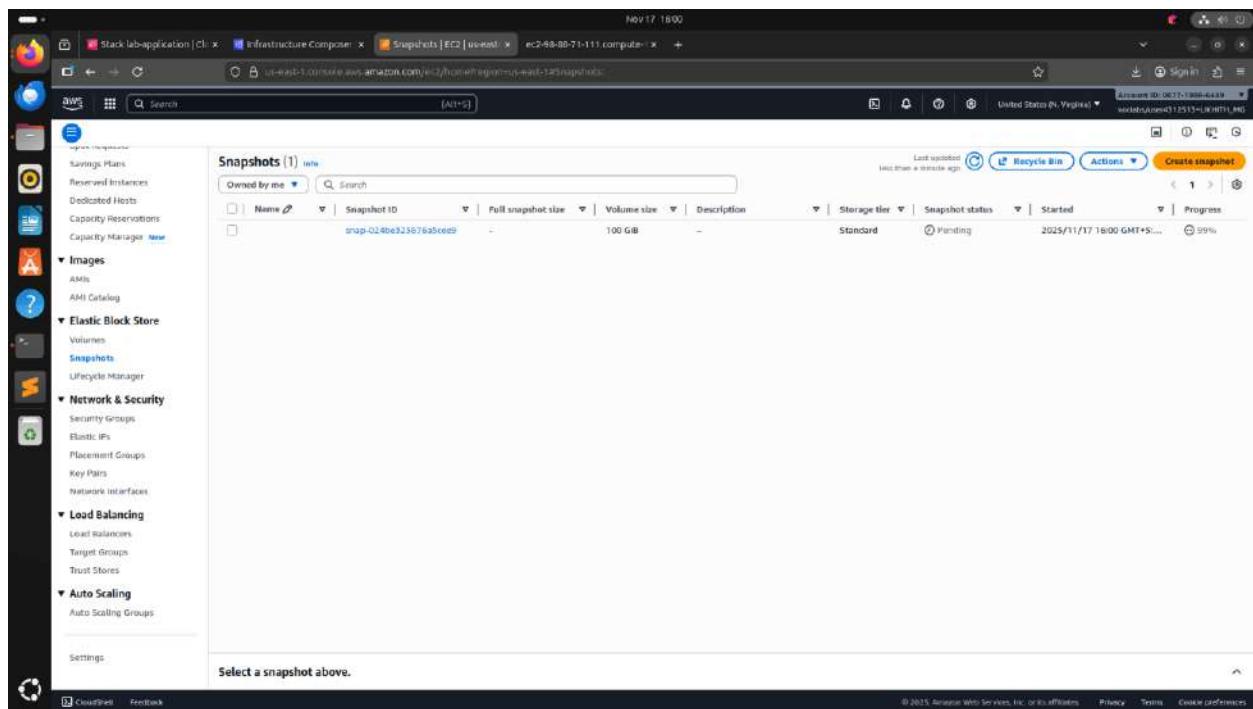
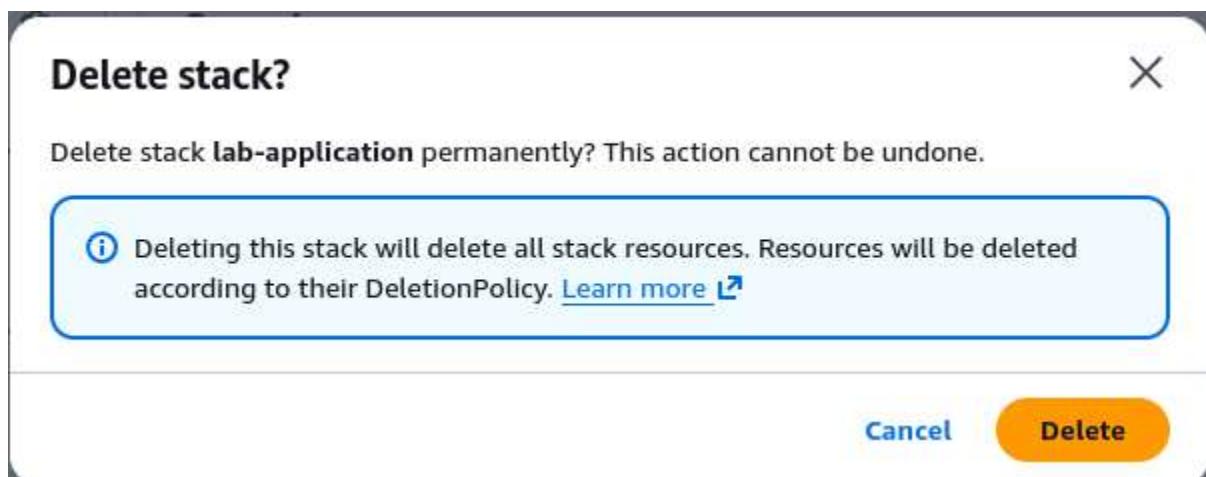
The application stack was removed, but the network stack remained untouched. This scenario reinforces the idea that different teams (for example, the network team or the application team) can manage their own stacks.

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.



# 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**.

### Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

#### Name and tags Info

##### Name

[Add additional tags](#)

#### ▼ Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

[Recents](#)[Quick Start](#)[Browse more AMIs](#)

Including AMIs from AWS, Marketplace and the Community

#### Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type  
ami-0ecb62995f68bb549 (64-bit (x86)) / ami-01b9f1e7dc427266e (64-bit (Arm))  
Virtualization: hvm ENA enabled: true Root device type: ebs

[Free tier eligible](#)

#### Description

Ubuntu Server 24.04 LTS (HVM) EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Canonical, Ubuntu, 24.04, amd64 noble Image

#### Architecture

#### AMI ID

#### Publish Date

#### Username

ubuntu

Verified provider

[64-bit \(x86\)](#)[ami-0ecb62995f68bb549](#)[2025-10-22](#)

## ▼ 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



### 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

**Inbound Security Group Rules**

- ▼ Security group rule 1 (TCP, 22, 45.112.148.130/32)
 

[Remove](#)

Type   <a href="#">Info</a>	Protocol   <a href="#">Info</a>	Port range   <a href="#">Info</a>
ssh	TCP	22
Source type   <a href="#">Info</a>	Name   <a href="#">Info</a>	Description - optional   <a href="#">Info</a>
My IP	<input type="text" value="Add CIDR, prefix list or security group"/> 45.112.148.130/32	e.g. SSH for admin desktop
  
- ▼ Security group rule 2 (TCP, 80, 0.0.0.0/0)
 

[Remove](#)

Type   <a href="#">Info</a>	Protocol   <a href="#">Info</a>	Port range   <a href="#">Info</a>
HTTP	TCP	80
Source type   <a href="#">Info</a>	Source   <a href="#">Info</a>	Description - optional   <a href="#">Info</a>
Anywhere	<input type="text" value="Add CIDR, prefix list or security group"/> 0.0.0.0/0	e.g. SSH for admin desktop

**⚠️** Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. X

[Add security group rule](#)

► Advanced network configuration

## 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
    // 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

---

# Mom & Pop Café

---

Home   About Us   Contact Us   Menu   Order History



Mom & Pop Café offers an assortment of delicious and delectable pastries and coffees that will put a smile on your face. From cookies to croissants, tarts and cakes, each treat is especially prepared to excite your tastebuds and brighten your day!

*Pop bakes a rich variety of cookies. Try them all!*



*Our tarts are always a customer favorite!*





*Tea  
Coffee  
Latte  
Hot  
Chocolate  
Yes, we have it!*

---

---

# Mom & Pop Café

---

Home   **Menu**   Order History

*Pastries*



**Croissant**  
\$1.50  
Fresh, buttery and fluffy... simply delicious!  
Quantity:



**Donut**  
\$1.00  
We have more than half-a-dozen flavors!  
Quantity:



**Chocolate Chip Cookie**  
\$2.50  
Made with Swiss chocolate with a touch of Madagascar vanilla  
Quantity:



**Muffin**  
\$3.00  
Banana bread, blueberry, cranberry or apple  
Quantity:



**Strawberry Blueberry Tart**  
\$3.50  
Bursting with the taste and aroma of fresh fruit  
Quantity:



**Strawberry Tart**  
\$3.50  
Made with fresh ripe strawberries and a delicious whipped cream  
Quantity:

# 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:** IAM 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.

User name	Path	Groups	Last activity	MFA	Console last sign-in	Access key ID	Active key age
user-1	/tmp/66/	0	~	-	15 minutes	Active - AKIAZUCNLSZ...	15 minutes
user-2	/tmp/66/	0	~	-	15 minutes	Active - AKIAZUCNLSZ...	15 minutes
user-3	/tmp/66/	0	~	-	15 minutes	Active - AKIAZUCNLSZ...	15 minutes

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

Group name	Users	Permissions	Creation time
EC2-Admin	0	Defined	16 minutes ago
EC2-Support	0	Defined	16 minutes ago
S3-Support	0	Defined	16 minutes ago

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

**S3-Support**

**Summary**

User group name: S3-Support

Creation time: November 18, 2023, 13:51 (UTC+05:30)

ARN: arn:aws:iam::661582816049:group/api/66/S3-Support

**Users** | Permissions | Access Advisor

**Users in this group (0)**

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

No resources to display

**Groups** | Last activity | Creation time

**Add users**

**Remove**

**1 user added to this group:**

**S3-Support**

**Summary**

User group name: S3-Support

Creation time: November 18, 2023, 13:51 (UTC+05:30)

ARN: arn:aws:iam::661582816049:group/api/66/S3-Support

**Users** | Permissions | Access Advisor

**Users in this group (1)**

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

user-1

No resources to display

**Groups** | Last

**Add users**

**Remove**

- Added user-2 to EC2-Support

**EC2-Support**

**Summary**

User group name: EC2-Support

Creation time: November 18, 2023, 13:51 (UTC+05:30)

ARN: arn:aws:iam::661582816049:group/api/66/EC2-Support

**Users** | Permissions | Access Advisor

**Users in this group (0)**

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

No resources to display

**Groups** | Last activity | Creation time

**EC2-Support**

**Summary**

User group name: EC2-Support

Creation time: November 18, 2025, 13:51 (UTC+05:30)

ARN: arn:aws:iam::661587616049:group/spl06/EC2-Support

**Users (1)**

user-3

- Added user-3 to EC2-Admin

**EC2-Admin**

**Summary**

User group name: EC2-Admin

Creation time: November 18, 2025, 13:51 (UTC+05:30)

ARN: arn:aws:iam::661587616049:group/spl06/EC2-Admin

**Users (0)**

No resources to display

**EC2-Admin**

**Summary**

User group name: ec2-admin

Creation time: November 18, 2025, 13:51 (UTC+05:30)

ARN: arn:aws:iam::661587616049:group/spl06/ec2-admin

**Users (1)**

user-3

Finally verified that each group displayed 1 assigned user.

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

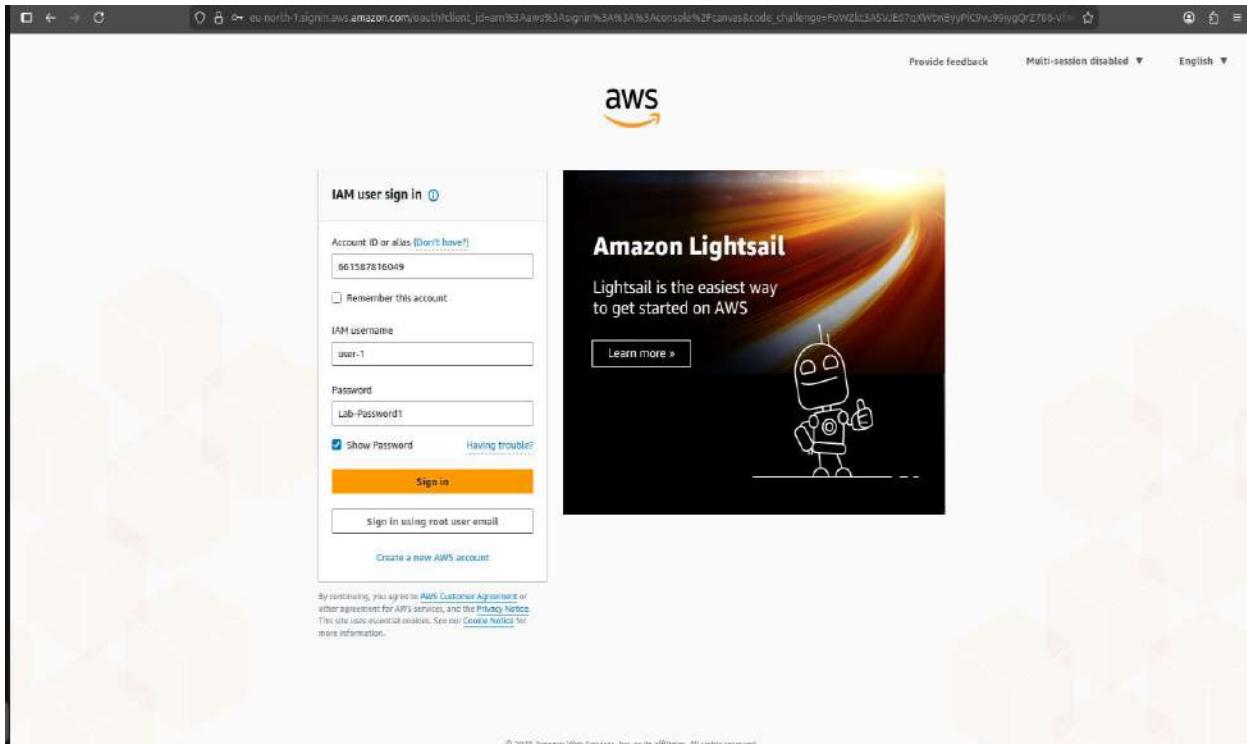
- In the navigation pane on the left, choose the Dashboard. On the right side, a Sign-in URL for IAM users in this account is displayed. The sign-in link shown in the lab environment was: <https://661587816049.signin.aws.amazon.com/console>

The screenshot shows the AWS IAM Dashboard. On the left, there's a navigation sidebar with options like 'Identity and Access Management (IAM)', 'Access management', 'Access reports', and 'AWS Organizations'. The main area is titled 'IAM Dashboard' and contains sections for 'IAM resources' (User groups: 3, Users: 4, Roles: 13, Policies: 1, Identity providers: 0), 'What's new' (with a list of recent changes), and 'AWS Account' (Account ID: 661587816049, Account Alias: Create, Sign-in URL for IAM users in this account: https://661587816049.signin.aws.amazon.com/console). There are also sections for 'Tools' (Policy simulator) and 'Additional information' (Security best practices in IAM).

- 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.

The screenshot shows a Firefox browser window. The address bar displays the URL: https://661587816049.signin.aws.amazon.com/console. A modal dialog box is open, titled 'Next-level privacy on mobile', featuring an image of a person using a smartphone. It explains that Firefox Focus clears history every time while blocking ads and trackers, and provides a 'Download Firefox Focus' button. Below the modal, a smaller text note says: 'Firefox clears your search and browsing history when you close all private windows, but this doesn't make you anonymous.' with a 'Learn more' link.

Logged in as user-1



Successfully accessed S3 and viewed bucket list (read-only)

A screenshot of the Amazon S3 console showing the bucket list. The URL is <https://us-east-1.console.aws.amazon.com/s3/home?region=us-east-1#>. The page has a header with the AWS logo and the account ID 661587816049. On the left, there's a sidebar with links for General purpose buckets, Directory buckets, Table buckets, Vector buckets, Access Grants, Access Points (General Purpose Buckets, FSx file systems), Access Points (Directory Buckets), Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and IAM Access Analyzer for S3. Below that is a section for Storage Lens with links for Dashboards, Storage Lens groups, and AWS Organizations settings. There's also a Feature spotlight section and a link to the AWS Marketplace for S3. The main content area shows a table for "General purpose buckets (1/1)". The table has columns for Name, AWS Region, and Creation date. One row is shown: samplebucket-7a82d9c0, US East (N. Virginia) us-east-1, November 18, 2025, 13:50:51 (UTC+05:30). To the right of the table are two boxes: "Account snapshot" (updated daily) which provides visibility into storage usage and activity trends, and "External access summary - new" (updated daily) which helps identify bucket permissions allowing public access or access from other AWS accounts.

Tried accessing EC2 but received "not authorized" error

Screenshot of the AWS EC2 Instances page. The user is trying to perform an action on an instance, but a red error message box appears stating: "You are not authorized to perform this operation. User: arn:aws:iam::661567816049:user/user-1 is not authorized to perform: ec2:DescribeInstances because no identity-based policy allows the ec2:DescribeInstances action".

Logged in as user-2

Screenshot of the AWS IAM user sign-in page for user-2. The page shows the sign-in form with fields for Account ID or alias, IAM username, and Password. To the right of the form is a promotional banner for Amazon Lightsail.

Successfully viewed EC2 instance (read-only)

Screenshot of the AWS EC2 Instances page showing two running t2.micro instances: LabHost and Bastion Host.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
LabHost	i-0678558a9257a3d57	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-100-24-119-163.compute-1.amazonaws.com	100.24.119.163	-
Bastion Host	i-071050fe0e597a8a	Running	t2.micro	2/2 checks passed	View alarms	us-east-1a	ec2-44-200-160-192.compute-1.amazonaws.com	44.200.160.192	-

## Attempted to stop EC2 instance but got permission denied

Screenshot of the AWS EC2 Instances page showing a failed attempt to stop instance i-0678558a9257a3d57.

**Failed to stop the instance i-0678558a9257a3d57**

You are not authorized to perform the operation. User: arn:aws:iam::661387810049:root [user-2] is not authorized to perform: ec2:StopInstances on resource:arn:aws:ec2:us-east-1:661387810049:instance/i-0678558a9257a3d57 because no identity-based policy allows the ec2:StopInstances action. Encoded authorization failure message: w6WqjgphRk8kE-2v-09Tb-4tC3Q9glL1JfC0r4g46s1TAfSm/wjSm/2fBMrk94X-HvvwrtLuvic5f7FkOy96e9H1AVdVWe5Pw-R-96880Xcvvzxa-Xpjd\_QjwYLuqBUzrnQjQcOsQsgqgkPf7stgQubqM4YJLmAdmpWqg2Qgk8d7qZDfOtg2zalMn7ZAShQybfaz-CHGK9jCCPrPcPHVO\_Smgqsd8tDXk8kE-hrrxxMgjgzK9Rsm2Phk5NODhWQ4GAS5sos5Mc84vzcatunIfkdlPofYV028Uvg4Yc4h4hs4D-gtYF8HgQDjh8EpzTSF-OkaCje-FWYj7tIdy53z06rgq7\_20x452m12-UWujoUbStjrh-gevveWVXfCv24RkWc3ke859Cw3567luuvA\*AbdyfctVvhej20CTsUoE0dg97\_wiacuCYwifgD8-jpYc4pjk3aee4gkUsD\_0h750e7r6AGHgKqHesqjG3WKLzbAo8sJtIMAvUjnnAPms\_L\_GsQ\_BTEOpavdvhm459hWfricarbjf1sIbuqjUniqkAEVNa\_d0ShwQ7QyWj-OlpmrkHomzCbdCA00L2-xOVVB0dMgVwgPj9shtw-27NLBHTV2MVJLNEx2vQbV7hRvVjhq5TAERfB02X4nW9kwUdzax0eqOidm/2d6pIsbyf\_UUT9NReabsUxb09E3pGh5h58C9Gh6gUcB3X40-Nkzb8YDj7bs9d-4qntuifCOj7g-dRbw44ur-2056Lsh1w-N\_Cd6Qulxkt\_Wbzfr7A0-PryTnsALMwK-iztL5\_e\_r5utv5j0k453Uvz2D-05mCvqjcfusfAgnm/voVQSVs\_UPEtgAv/hqfHMeuk7Rf2ALh1chukkj5zz7j/ghvZoy\_xWdAK-94qLs5z7bc1JmPKYU8YbNko2AdStasZkNmAgElf2/p7H9503cbr

**Instances (1/2) Info**

Details	Status and alarms	Monitoring	Security	Networking	Storage	Tags
<b>i-0678558a9257a3d57 (LabHost)</b>						
<b>Details</b>						
<b>Instance summary</b>						
Instance ID	i-0678558a9257a3d57	Public IPv4 address	100.24.119.163   open address	Private IPv4 addresses	10.1.11.152	
IPv6 address	-	Instance state	Running	Public DNS	ec2-100-24-119-163.compute-1.amazonaws.com   open address	
Hostname type	IP name: ip-10-1-11-152.ec2.internal	Private IP DNS name (IPv4 only)	ip-10-1-11-152.ec2.internal	Instance type	t2.micro	
Answer private resource DNS name	-			Elastic IP addresses	-	

## Tried accessing S3 but was denied

Screenshot of the AWS S3 Bucket creation page:

The page shows the configuration for creating a new bucket named "user-3".

**Default encryption**: Info  
Server-side encryption is automatically applied to new objects stored in this bucket.

**Encryption type**: Info  
Secure your objects with two separate layers of encryption. For details on pricing, see [DSSE-KMS pricing](#) on the Storage tab of the [Amazon S3 pricing page](#).

Server-side encryption with Amazon S3 managed keys (SSE-S3)

Server-side encryption with AWS Key Management Service keys (SSE-KMS)

Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)

**Bucket Key**  
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering costs to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

Disable

Enable

**Advanced settings**

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

**Failed to create bucket**  
To create a bucket, the `s3:CreateBucket` permission is required.  
View your permissions in the [IAM console](#). [Identity and Access Management in Amazon S3](#)

[Diagnose with Amazon Q](#)

[Cancel](#) [Create bucket](#)

Screenshot of the AWS IAM User Sign In page:

The user is logged in as "user-3".

**IAM user sign in**

Account ID or alias (Don't have?)  
661587816049

Remember this account

IAM username  
user-3

Password  
Lab-Password3

Show Password [Having trouble?](#)

[Sign in](#)

[Sign in using root user email](#)

[Create a new AWS account](#)

By continuing, you agree to [AWS Customer Agreement](#) or other agreement for AWS services, and the [Privacy Notice](#). This site uses essential cookies. See our [Cookie Notice](#) for more information.

**Amazon Lightsail**  
Lightsail is the easiest way to get started on AWS

[Learn more](#)

© 2025 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Accessed EC2 and successfully stopped the LabHost instance

**Instances (1/2)**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4	Elastic IP
LabHost	i-0670558a9257a3d57	Stopping	t2.micro	2/2 checks passed	User: amarnath	us-east-1a	ec2-100-24-119-163.co...	100.24.119.163	-
Boston Host	i-0716504dd8907a8a	Running	t2.micro	2/2 checks passed	User: amarnath	us-east-1a	ec2-44-200-160-192.co...	44.200.160.192	-

**i-0670558a9257a3d57 (LabHost)**

**Details** Status and alarms Monitoring Security Networking Storage Tags

**Instance summary**

Instance ID	i-0670558a9257a3d57	Public IPv4 address	100.24.119.165   open address
IPv6 address	-	Instance state	Stopping
Hostname type	IP name: ip-10-1-11-152.ec2.internal	Private IP DNS name (IPv6 only)	(ip-10-1-11-152.ec2.internal)
Answer private resource DNS name	-	Instance type	t2.micro
		Private IPv4 addresses	10.1.11.152
		Public DNS	ec2-100-24-119-163.compute-1.amazonaws.com   open address
		Elastic IP addresses	-

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.

The screenshot shows a web browser window with the URL [http://lab.vocareum.com/main/main.php?mode=lab&mode\\_id=4508331&step\\_id=4508332&hidden\\_bar=1](http://lab.vocareum.com/main/main.php?mode=lab&mode_id=4508331&step_id=4508332&hidden_bar=1). The page title is "Lab overview". It contains a flow diagram with three icons: a sun icon labeled "EventBridge event", a Lambda function icon labeled "Lambda function triggered", and a red octagonal stop sign icon labeled "EC2 instance stopped". Below the diagram is a descriptive text block: "In this hands-on activity, you will create an AWS Lambda function. You will also create an Amazon EventBridge event to trigger the function every minute. The function uses an AWS Identity and Access Management (IAM) role. This IAM role allows the function to stop an Amazon Elastic Compute Cloud (Amazon EC2) instance that is running in the Amazon Web Services (AWS) account." There are sections for "Duration" (approx. 30 minutes), "AWS service restrictions" (mentioning restrictions on other services), and "Accessing the AWS Management Console".

## 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:

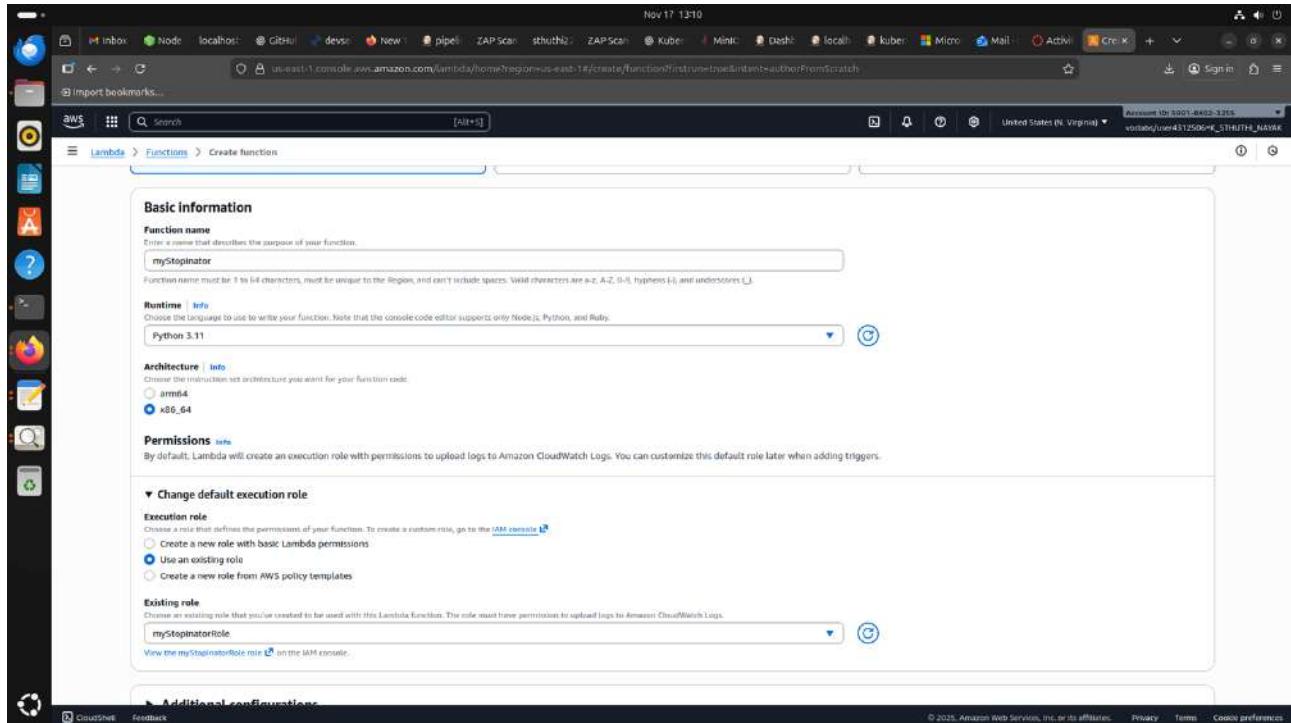
- **Function name:** myStopinator
- **Runtime:** Python 3.11

4. Configured the execution role:

- **Execution role:** Use an existing role
- **Existing role:** myStopinatorRole

5. Clicked **Create function**.

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



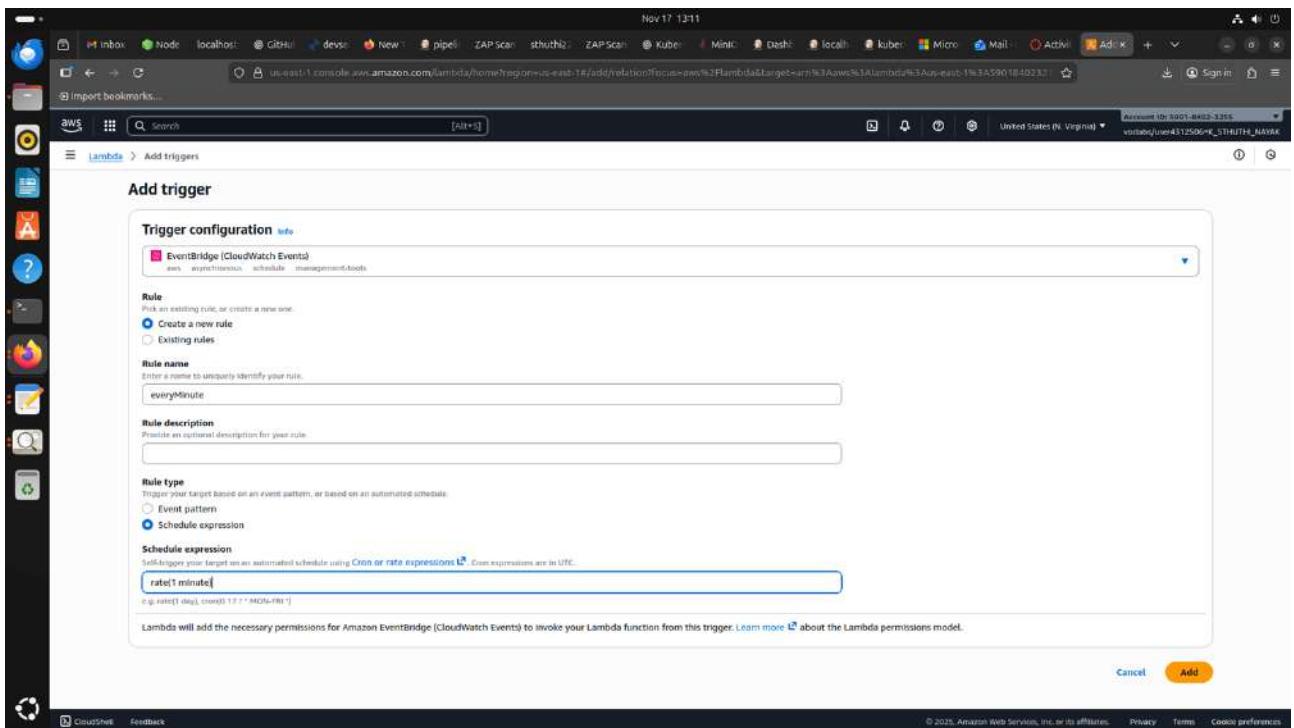
## Task 2: Configure the Trigger

**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

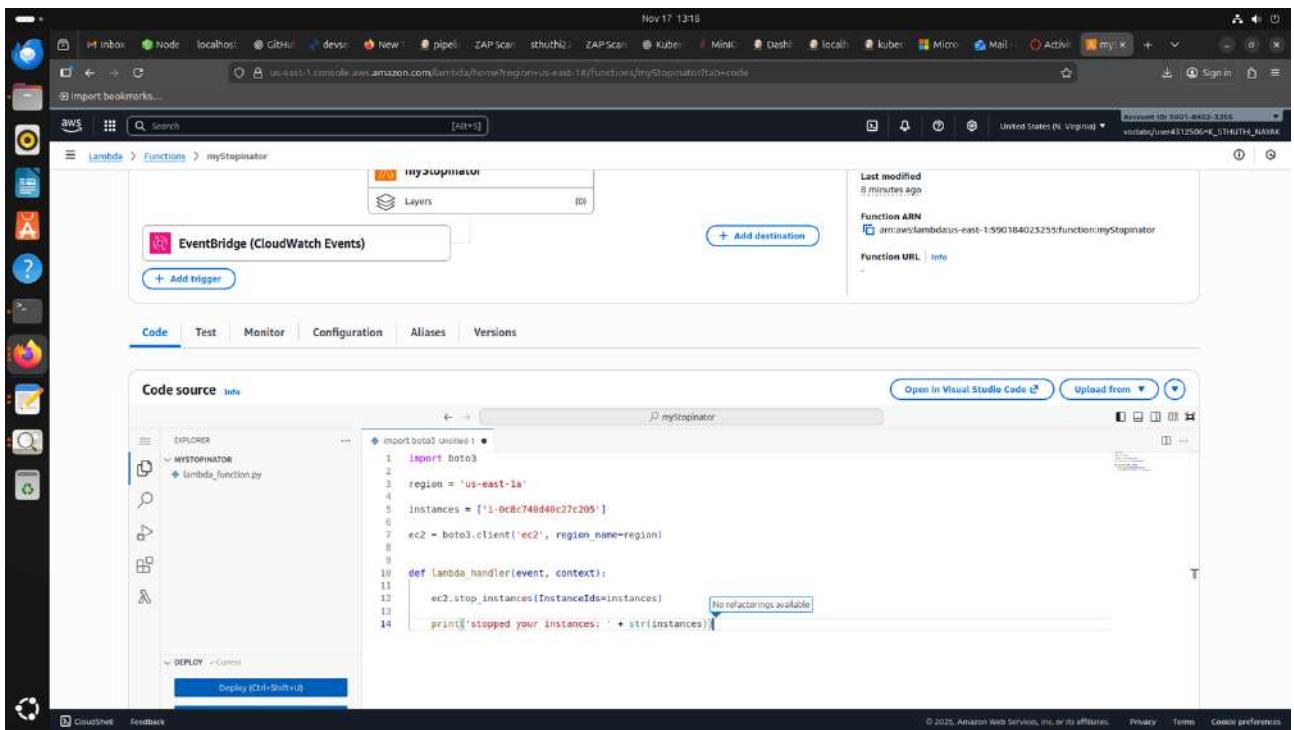
region = '<REPLACE_WITH_REGION>'
instances = ['<REPLACE_WITH_INSTANCE_ID>']

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.

The screenshot shows the AWS Management Console with the EC2 service selected. The left sidebar shows navigation links for EC2, including Dashboard, Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, and Key Pairs. The main content area displays the 'Instances (2) info' section with a table of instance details. The table has columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4 IP, and Elastic IP. Two instances are listed: 'Bastion Host' (running, t2.micro, 2/2 checks passed, us-east-1a, ec2-54-235-6-84.com, 54.235.6.84, -) and 'Instance1' (stopped, t2.micro, -). A search bar at the top allows filtering by instance name or tag. Buttons for 'Connect', 'Instance state', 'Actions', and 'Launch Instances' are available at the top right. A note at the bottom indicates the instances were created via a Lambda function.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
Bastion Host	i-040455a06c32498bd	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1a	ec2-54-235-6-84.com...	54.235.6.84	-
Instance1	i-08fc740d49c27205	Stopped	t2.micro	-	View alarms +	us-east-1a	-	98.81.89.141	-

## 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

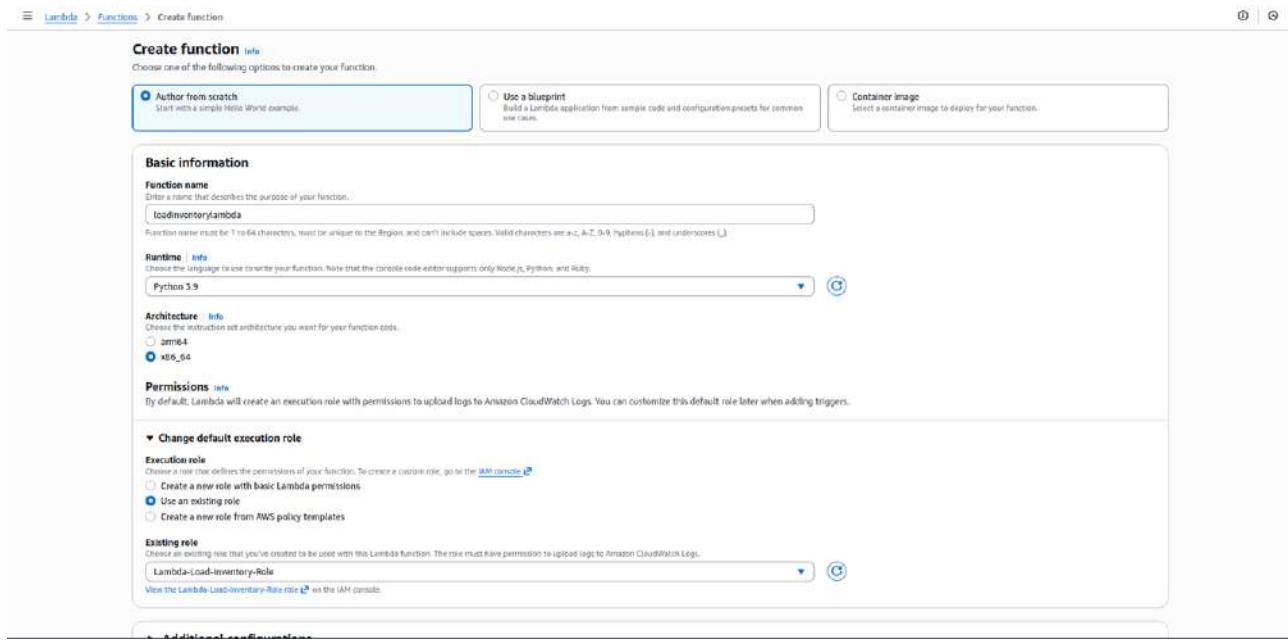
## STEP 1: CREATE A LAMBDA FUNCTION

Function name: loadinventorylamda

Runtime: python 3.9

Change execution role: use an existing 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

The screenshot shows the AWS S3 'Upload' interface for the 'myinventory12321' bucket. At the top, there's a header with navigation links: 'Amazon S3 > Buckets > myinventory12321 > Upload'. Below the header is a large 'Upload' section with a 'Drag and drop files and folders here, or choose Add files or Add folder.' area. Underneath is a 'Files and folders' table showing six CSV files:

Name	Folder	Type	Size
inventory-berlin.csv	-	text/csv	140.0 B
inventory-calcutta.csv	-	text/csv	150.0 B
inventory-karachi.csv	-	text/csv	145.0 B
inventory-guiana.csv	-	text/csv	134.0 B
inventory-shanghai.csv	-	text/csv	152.0 B
inventory-springfield.csv	-	text/csv	168.0 B

Below the table is a 'Destination' section with a 'Destination' dropdown set to 's3://myinventory12321'. It includes sections for 'Destination details' (Bucket settings), 'Permissions' (Grant public access and access to other AWS accounts), and 'Properties' (Specify storage class, encryption settings, tags, and more). At the bottom right are 'Cancel' and 'Upload' buttons.

## STEP 4: DYNAMO DB

-Explore the items.

-Items will be displayed.

The screenshot shows the Dyanamlk interface with the following details:

- Left Sidebar:** Includes sections for Home, Projects, Reports, Dashboards, and Help.
- Top Bar:** Shows the title "Inventory" and navigation icons for Refresh, Save, and Print.
- Table List:** A sidebar titled "Tables (1)" showing one item: "Inventory".
- Inventory View:** The main area displays the "Inventory" table with the following data:

Item	Item Description	Item Count
Shoe Tag	Shoe Tag	14
Shoe Laces	Shoe Laces	8
Shoe Sole	Shoe Sole	2
Shoe Box	Shoe Box	0
Shoe Pad	Shoe Pad	10
Shoe Shoe	Shoe Shoe	12
Shoe Bag	Shoe Bag	9
Shoe Box (red)	Shoe Box (red)	11
Shoe Box	Shoe Box	14
Shoe Laces	Shoe Laces	22
Shoe Pad	Shoe Pad	10
Shoe Shoe	Shoe Shoe	22
Shoe Tag	Shoe Tag	1
Shoe Laces (red)	Shoe Laces (red)	2
Shoe Box	Shoe Box	0
Shoe Laces	Shoe Laces	0
Shoe Pad	Shoe Pad	10
Shoe Shoe	Shoe Shoe	0
Shoe Tag	Shoe Tag	13
Shoe Laces (red)	Shoe Laces (red)	14
Shoe Box	Shoe Box	0
Shoe Laces	Shoe Laces	14
Shoe Pad	Shoe Pad	0
Shoe Shoe	Shoe Shoe	14
Shoe Tag	Shoe Tag	1

Below the table, there are buttons for "Save" and "Print".

## STEP 5: DASHBOARD

[Go to AWS I details](#)

Cloud Access	
AWS CLI:	<a href="#">Show</a>
<b>Cloud Labs</b>	
Remaining session time: 02:19:16(140 minutes)	
Session started at: 2025-11-16T23:48:06-0800	
Session to end at: 2025-11-17T02:48:06-0800	
Accumulated lab time: 11:41:00 (701 minutes)	
No running instance	
SSH key	<a href="#">Show</a>
	<a href="#">Download PEM</a>
	<a href="#">Download PPK</a>
AWS SSO	<a href="#">Download URL</a>
<b>Dashboard</b> <a href="https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-AACACAD-3-89090/18-lab-mod14-guided-Lambda/s3/web/inventory.html?region=us-east-1&amp;poolId=us-east-1:df383338-2b3c-4379-9cd3-7bafb21c5307">https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-AACACAD-3-89090/18-lab-mod14-guided-Lambda/s3/web/inventory.html?region=us-east-1&amp;poolId=us-east-1:df383338-2b3c-4379-9cd3-7bafb21c5307</a>	
IdentityPoolId	us-east-1:df383338-2b3c-4379-9cd3-7bafb21c5307

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

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The first step, 'Basic information', is selected. In the 'Function name' field, 'stocknotification' is entered. The 'Runtime' is set to 'Python 3.9'. Under 'Architecture', 'x86\_64' is chosen. In the 'Permissions' section, 'Lambda-Check-Stock-Role' is selected from the dropdown. At the bottom right, there are 'Cancel' and 'Create function' buttons.

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'][0])
            if count == 0:
                store = record['dynamodb']['NewImage']['Store'][0]
                item = record['dynamodb']['NewImage']['Item'][0]
                # 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

**Create subscription**

<b>Details</b>
Topic ARN C: arn:aws:sns:us-east-1:267597929468:no_stock
Protocol The type of endpoint to subscribe Email
Endpoint An email address that can receive notifications from Amazon SNS. sudheerkumar04@gmail.com
After your subscription is created, you must confirm it. <a href="#">Info</a>
<b>Subscription filter policy - optional</b> <small>This policy filters the messages that a subscriber receives.</small>
<b>Redrive policy (dead-letter queue) - optional</b> <small>Send undeliverable messages to a dead-letter queue.</small>
<a href="#">Cancel</a> <a href="#">Create subscription</a>

Go mail and confirm subscription.

## Step 8:Trigger function

Go to lambda

Add trigger :dynamodb

Dynamo table:inventory

**Add trigger**

**Trigger configuration** [Info](#)

**DynamoDB** [aws](#) [database](#) [event-source-mapping](#) [mysql](#) [peeling](#)

**DynamoDB table**  
Choose or enter the ARN of a DynamoDB table.

arn:aws:dynamodb:us-east-1:767397929468:table/inventory [X](#) [?](#)

**Event poller configuration**

**Activate trigger** Select to activate the trigger now. Keep unchecked to create the trigger in a deactivated state for testing (recommended).

**Enable EventCount metrics** Track the number of events polled, filtered, invoked, and dropped by your event source mapping. CloudWatch charges apply.

**Batch size** [Info](#) The maximum number of records in each batch to send to the functions.

100 [?](#)

**Starting position** [Info](#) The position in the stream to start reading from.

Latest [?](#)

**Batch window - optional** The maximum amount of time to gather records before invoking the functions, in seconds.

10 [?](#)

**Additional settings**

In order to read from the DynamoDB trigger, your execution role must have proper permissions.

[Cancel](#) [Add](#)

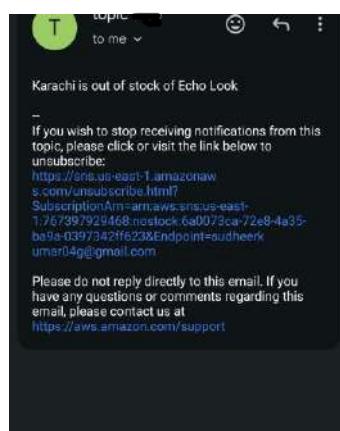
## STEP 9: upload files

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

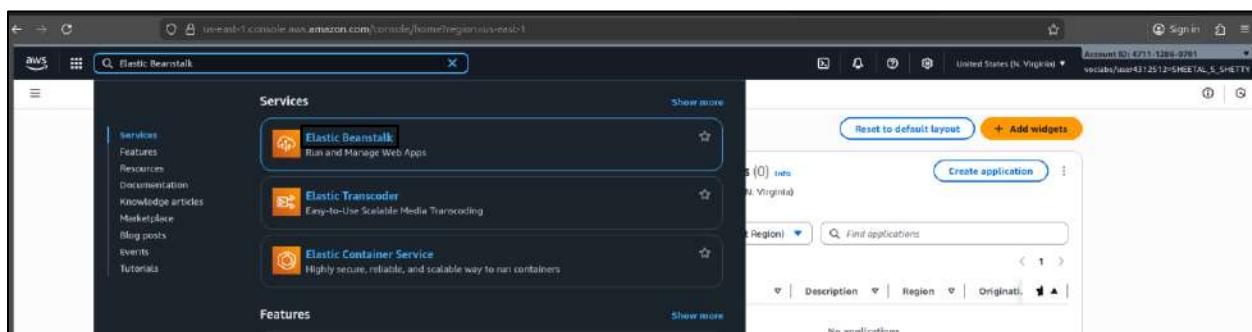
1. You upload your application code (Java, .NET, Node.js, Python, etc.).
2. Elastic Beanstalk automatically handles:
  - a. Provisioning and configuring servers (EC2 instances)
  - b. Setting up load balancers and Auto Scaling groups
  - c. Managing the application environment
  - d. Monitoring, logging,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

The screenshot shows the AWS Elastic Beanstalk console. In the left navigation menu, 'Environments' is selected. The main area displays a table titled 'Environments (1)'. The single environment listed is 'c1741-samp-0M9mzq0CP13', which is marked as 'Ok'. Other columns in the table include 'Health', 'Application', 'Platform', 'Domain', 'Running...', 'Tier name', 'Date created...', and 'Last modified...'. A 'Create environment' button is located at the top right of the table.

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

The screenshot shows the AWS EC2 console. In the left navigation menu, 'Load Balancing' is selected. The main area displays a table titled 'Load balancers (1/1)'. The single load balancer listed is 'awseb-e-4-AWSEBLba-16BFSLHRKYY7', which is of type 'classic'. Other columns include 'Name', 'State', 'Type', 'Scheme', 'IP address type', 'VPC ID', 'Availability Zones', and 'Security groups'. Below the table, there is a section for the specific load balancer 'awseb-e-4-AWSEBLba-16BFSLHRKYY7', showing its DNS name and distribution of targets by availability zone.

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.

The screenshot shows a web browser displaying a 404 Not Found error page. The URL in the address bar is 'http://awseb-e-4-awsebba-16BFSLHRKYY7-1100723373.us-east-1.elb.amazonaws.com'. The error message 'HTTP Status 404 – Not Found' is prominently displayed at the top of the page.

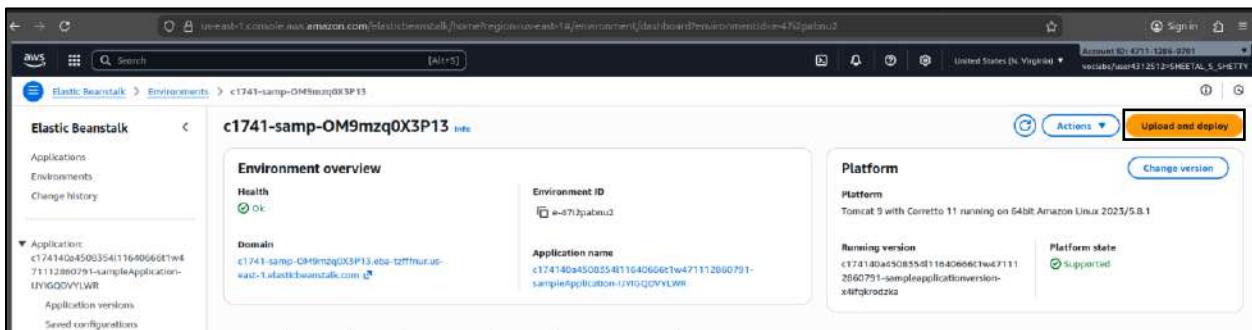
Step2: Deploying a sample application to Elastic Beanstalk.

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.

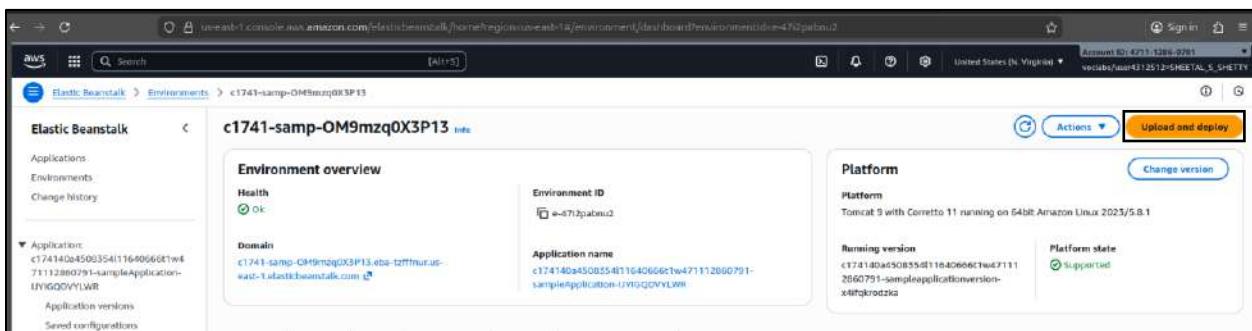


The screenshot shows a browser window with the AWS Current Lab instruction page. The title is "Task 2: Deploy a sample application to Elastic Beanstalk". Step 8 is highlighted with a red box around the text "To download a sample application, choose this link" and the URL "http://docs.aws.amazon.com/elastictbeanstalk/latest/dg/samples/tomcat.zip". Below it, steps 9 and 10 are listed: "Back in the Elastic Beanstalk Dashboard, choose Upload and Deploy." and "Choose File, then navigate to and open the tomcat.zip file that you just downloaded."

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

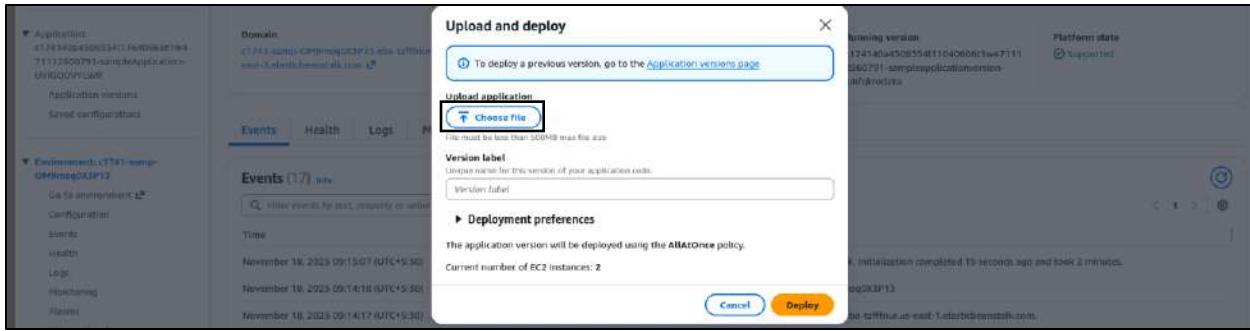


The screenshot shows the AWS Elastic Beanstalk Environment Overview page for environment "c1741-samp-OM9mzq0X3P13". The "Actions" button is highlighted with a red box. The page displays the environment overview, platform details (Tomcat 9 with Corretto 11 running on 64bit Amazon Linux 2023/5.8.1), and the application name (c174140a4500354111640666t1w47112660791-sampleApplication-UvIQGQVYLWR).

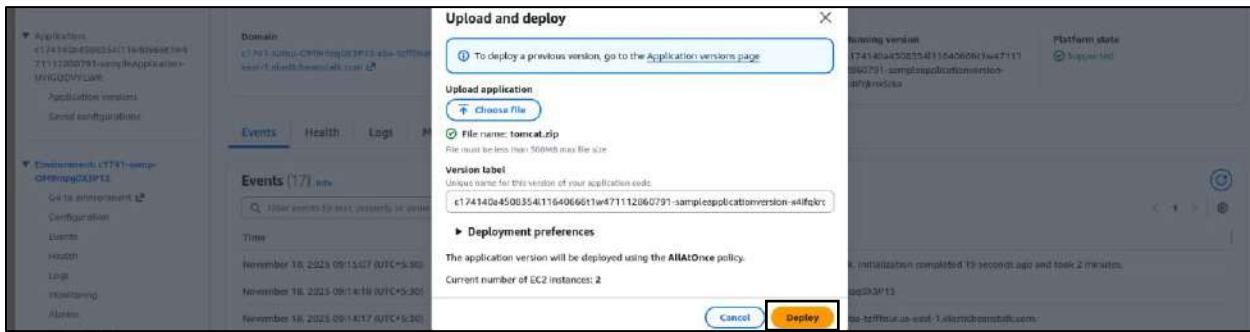


The screenshot shows the AWS Elastic Beanstalk Environment Overview page for environment "c1741-samp-OM9mzq0X3P13". The "Actions" button is highlighted with a red box. The page displays the environment overview, platform details (Tomcat 9 with Corretto 11 running on 64bit Amazon Linux 2023/5.8.1), and the application name (c174140a4500354111640666t1w47112660791-sampleApplication-UvIQGQVYLWR).

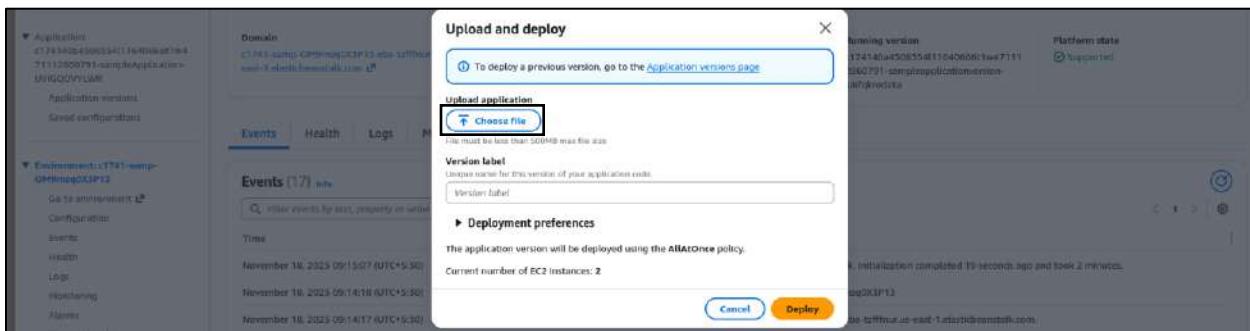
Select **Choose File**, then navigate to the **tomcat.zip** file you downloaded and open it.



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 screenshot shows the AWS Elastic Load Balancing (ELB) console. On the left, there's a navigation sidebar with various services like Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, 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 (which is selected), Target Groups, Trust Stores, and Auto Scaling.

The main content area displays a table titled "Load balancers (1/1)". The table has columns for Name, State, Type, Scheme, IP address type, VPC ID, Availability Zones, and Security groups. One row is visible, showing "awseb-e-4-AWSEBLba-16BFSLHRKVYV7" as the Name, classic as the Type, and vpc-0cc573ff79050edeb as the VPC ID. The Availability Zones listed are us-east-1b (use1-az2) and us-east-1c (use1-az3). The Security group is sg-0fb0544d2e50db4b.

Below the table, a section titled "Load balancer: awseb-e-4-AWSEBLba-16BFSLHRKVYV7" provides more details. It shows the DNS name as awseb-e-4-AWSEBLba-16BFSLHRKVYV7-1100723373.us-east-1.elb.amazonaws.com (A Record). A note says, "This Classic Load Balancer can be migrated to a next generation load balancer. Migration wizard uses your load balancer's current configurations to create a new load balancer. Learn more." There's also a "Launch migration wizard" button. At the bottom, there's a section titled "Distribution of targets by Availability Zone (AZ)" with a note about viewing registered instances and their health states.

The deployed web application will now be displayed.

The screenshot shows a web browser window with the URL http://awseb-e-4-awsebba-16bfslhrkvyyv7-1100723373.us-east-1.elb.amazonaws.com. The page has a blue header with the text "Congratulations" and "Your first AWS Elastic Beanstalk Application is now running on your own dedicated environment in the AWS Cloud". To the right, there's a "What's Next?" section with links to learn how to build, deploy, and manage your own applications using AWS Elastic Beanstalk, AWS Elastic Beanstalk concepts, learn how to create new application versions, and learn how to manage your application environments. Below that is a "Download the AWS Reference Application" section with a link to download a fully-featured reference application written in AWS SDK for Java. At the bottom, there's an "AWS Toolkit for Eclipse" section with links to download, get started, and view all AWS Elastic Beanstalk documentation.

# Amazon Elastic File System (Amazon EFS) Report

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

## Objectives

- ⑩ Access the AWS console, create an EFS file system, and launch an Amazon Linux EC2 instance.
- ⑩ Connect to the EC2 instance and mount the EFS file system.
- ⑩ Review and monitor the file system's performance.

## Lab Environment Setup

- ⑩ This lab introduces you to Amazon Elastic File System (Amazon EFS) by using the AWS Management Console.
- ⑩ A timer-based lab session was started using the “Start Lab” button.
- ⑩ Pop-up windows were allowed in the browser to open the AWS Management Console in a new tab.
- ⑩ Resources were named exactly as specified in the instructions to ensure the lab scoring script works properly.

The screenshot shows a web browser window for AWS Academy. The URL is [awsacademy.instructure.com/courses/131613/assignments/1511959/module\\_item\\_id/12598718](https://awsacademy.instructure.com/courses/131613/assignments/1511959/module_item_id/12598718). The page title is "ACAv3EN-US-LT13-131613 > Assignments > Guided lab: Introducing Amazon Elastic File System (Amazon EFS) > Guided lab: Introducing Amazon Elastic File System (Amazon EFS)".

The main content area displays the assignment details:

- Due: No Due Date
- Points: 15
- Status: Submitting an external tool

Below this, there is a "AWS" button and a dropdown menu set to "EN\_US". To the right, there is a "Submission" section showing the submission date as "Oct 28 at 1:41pm", "Submission Details", "Grade: 5 (15 pts possible)", and "Graded Anonymously: no". It also shows "Comments: No Comments".

The central content area contains the following text:

### Guided Lab: Introducing Amazon Elastic File System (Amazon EFS)

#### Lab overview and objectives

This lab introduces you to Amazon Elastic File System (Amazon EFS) by using the AWS Management Console.

After completing this lab, you should be able to:

- Log in to the AWS Management Console
- Create an Amazon EFS file system
- Log in to an Amazon Elastic Compute Cloud (Amazon EC2) instance that runs Amazon Linux
- Mount your file system to your EC2 instance
- Examine and monitor the performance of your file system

At the bottom, there are navigation links: "Previous" and "Next".

# 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*.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation pane with sections like Dashboard, EC2 Global View, Events, Instances (selected), Images, Elastic Block Store, and Network & Security. The main area shows a table of instances. One instance is selected: "EFS Client" (i-09dfd0f7bcbba0af3). The instance is running on a t2.micro type. Below the table, the details for the selected instance are shown. The "Security" tab is active. Under "Security details", it shows the IAM Role and the attached Security Groups. A callout box highlights the "Security group ID copied" message next to the sg-0a9e53857194be273 entry.

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

- ⑩ **Security group name:** EFS Mount Target
- ⑩ **Description:** Inbound NFS access from EFS clients
- ⑩ **VPC:** Lab VPC

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

⑩ Type: NFS

⑩ Source:

⑩ Custom

- ⑩ In the *Custom* box, paste the security group's **Security group ID** that you copied to your text editor

⑩ Choose **Create security group**.

The screenshot shows the AWS EC2 console with the 'Security Groups' page selected. On the left, there is a navigation sidebar with sections for Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups (which is currently selected), and Elastic IPs. The main area displays a table titled 'Security Groups (3) info'. The columns are: Name, Security group ID, Security group name, VPC ID, Description, and Owner. The data is as follows:

Name	Security group ID	Security group name	VPC ID	Description	Owner
EFSClient	sg-0a9e53857194be273	EFSClient	vpc-096da48e30394258d	EFS Client	53851797227
-	sg-0c9b4204eaahhh6fc7	default	vpc-075h774dec577f345	default VPC security group	53851797227
-	sg-0c3377404fb65b35b	default	vpc-096da48e30394258d	default VPC security group	53851797227

The screenshot shows the 'Create security group' wizard. At the top, it says 'Create security group' with a 'Info' link. Below that, a note states: 'A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.' The form is divided into sections:

- Basic details**:
  - Security group name: EFS Mount Target
  - Description: EFS Mount Target
  - VPC: vpc-096da48e30394258d (Lab VPC)
- Inbound rules**:
  - Type: NFS
  - Protocol: TCP
  - Port range: 2049
  - Source: Custom (dropdown menu shows 'sg-0a9e53857194be273')
  - Description - optional: Use: 'sg-0a9e53857194be273'
- Outbound rules**:
  - Type: All traffic
  - Protocol: All
  - Port range: All
  - Destination: Custom (dropdown menu shows 'EFSClient | sg-0a9e53857194be273')
  - Description - optional: 0.0.0.0/0

## 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**:

⑩ Uncheck Enable Automatic backups.

⑩ **Lifecycle management**:

⑩ for **Transition into IA** Select *None*.

⑩ In the **Tags optional** section, configure:

⑩ **Key:** Name

⑩ **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.  
  
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](#)

<b>Transition into Infrequent Access (IA)</b> Transition files to IA based on the time since they were last accessed in Standard storage. <input type="button" value="30 day(s) since last access"/>	<b>Transition into Archive</b> Transition files to Archive based on the time since they were last accessed in Standard storage. <input type="button" value="90 day(s) since last access"/>	<b>Transition into Standard</b> Transition files back to Standard storage based on when they are first accessed in IA or Archive storage. <input type="button" value="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](#)

<input checked="" type="radio"/> <b>Enhanced</b> Provides more flexibility and higher throughput levels for workloads with a range of performance requirements.	<input type="radio"/> <b>Bursting</b> 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 provided.

▶ **Additional settings**

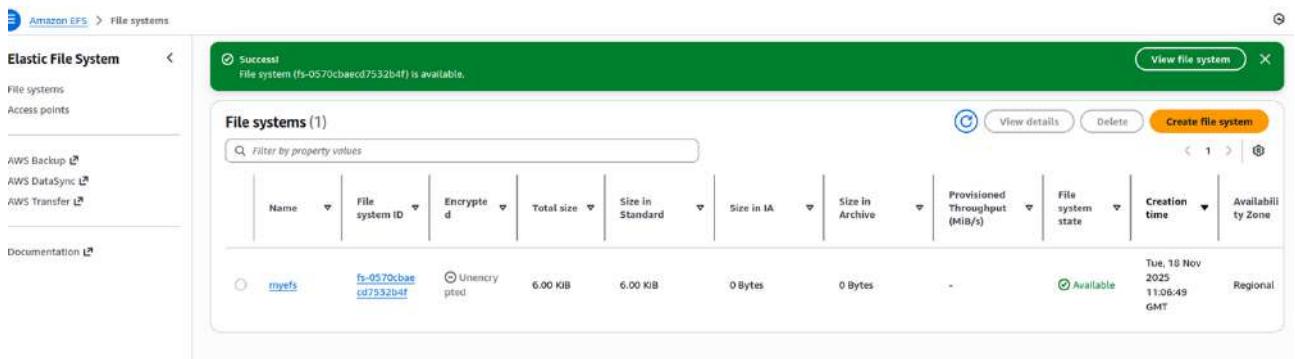
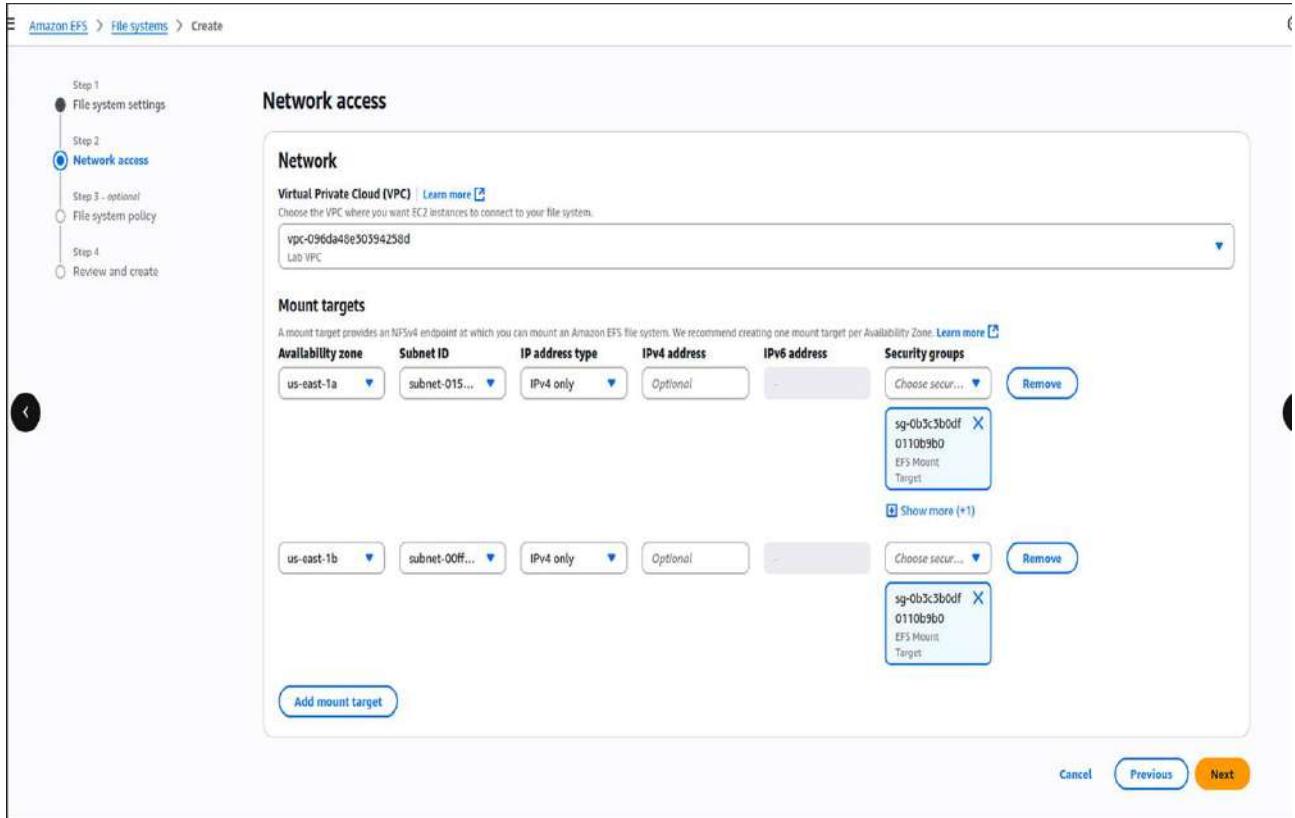
▼ **Tags optional**

Add tags to associate key-value pairs to your resource. [Learn more](#)

<b>Tag key</b>	<input type="text" value="Name"/> <span style="border: 1px solid #ccc; padding: 2px 5px;">X</span>	<input type="text" value="myefs"/> <span style="border: 1px solid #ccc; padding: 2px 5px;">X</span>
<b>Add tag</b>		

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.

The screenshot shows a Cloud Access interface with the following details:

- AWS CLI:** Show
- Cloud Labs:**
  - Remaining session time: 01:50:33(111 minutes)
  - Session started at: 2025-11-18T06:07:59-0800
  - Session to end at: 2025-11-18T08:07:59-0800
- Accumulated lab time:** 05:24:00 (324 minutes)
- IPs:** public:52.1.203.163, private:10.0.1.118
- SSH key:** Show, Download PEM, Download PPK
- AWS SSO:** Download URL
- InstanceSessionURL:** <https://us-east-1.console.aws.amazon.com/systems-manager/session-1-0b459763e87d1273e>

## 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**.

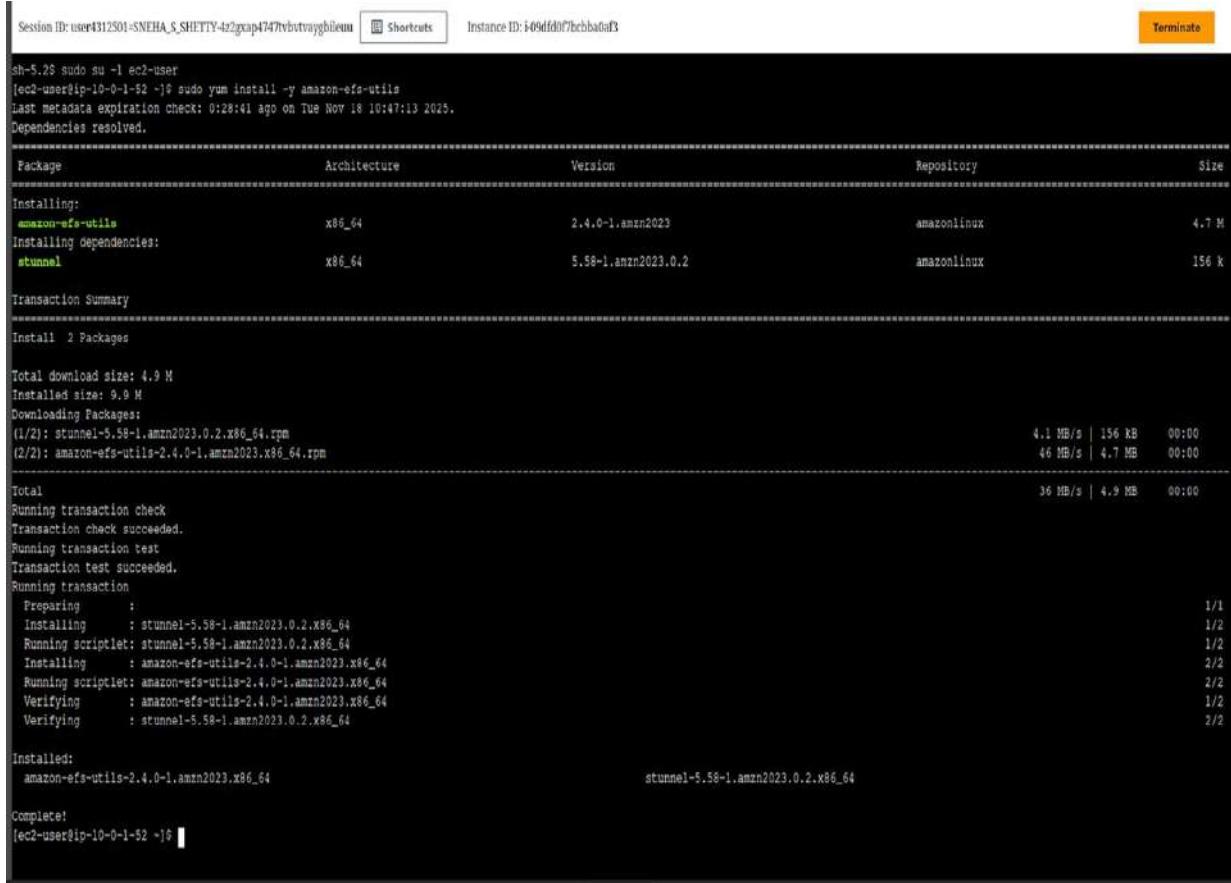
5. In the **Amazon EFS Console**, on the top right corner of the page, choose **Attach** to open the Amazon EC2 mount instructions.

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,wszie=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
```



The screenshot shows a terminal window with the following details at the top:

- Session ID: user4312501-SNEHA\_S\_SHETTY-4z2gcap4747vhvtvaygbilemu
- Shortcuts
- Instance ID: i-09dffd0f7bcbbafaf3
- Terminate

The terminal output shows the execution of a `sudo yum install -y amazon-efs-utils` command. The output includes dependency resolution, package installation details, transaction summary, and a detailed log of the yum process.

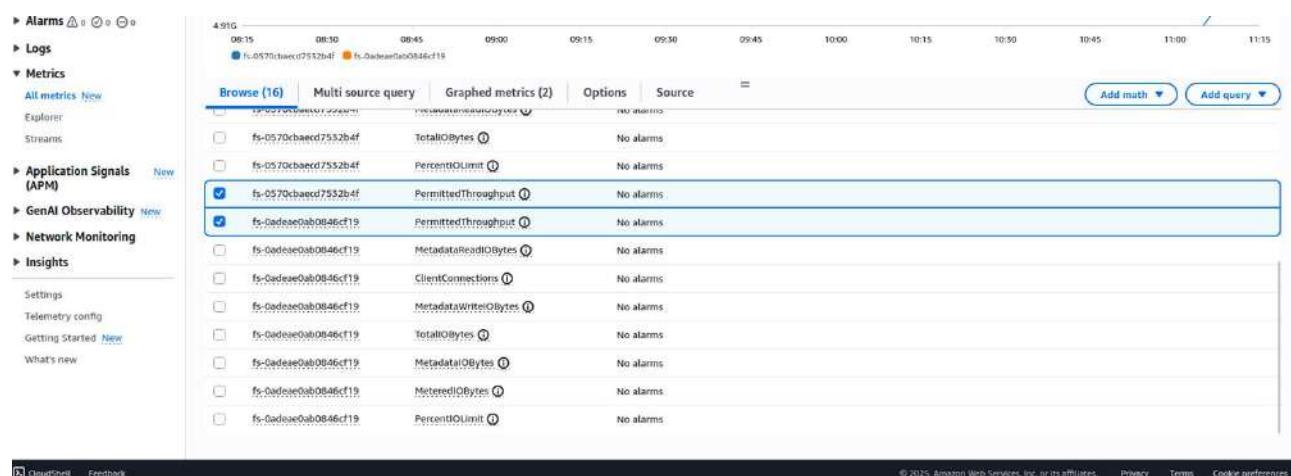
```
sh-4.2$ sudo su -l ec2-user  
[ec2-user@ip-10-0-1-52 ~]$ sudo yum install -y amazon-efs-utils  
Last metadata expiration check: 0:28:41 ago on Tue Nov 18 10:47:13 2025.  
Dependencies resolved.  
=====  
Package           Architecture      Version       Repository     Size  
=====  
Installing:  
amazon-efs-utils          x86_64        2.4.0-1.amzn2023  amazonlinux   4.7 M  
Installing dependencies:  
stunnel                  x86_64        5.58-1.amzn2023.0.2  amazonlinux  156 K  
=====  
Transaction Summary  
=====  
Install 2 Packages  
  
Total download size: 4.9 M  
Installed size: 9.9 M  
Downloading Packages:  
(1/2): stunnel-5.58-1.amzn2023.0.2.x86_64.rpm           4.1 MB/s | 156 KB    00:00  
(2/2): amazon-efs-utils-2.4.0-1.amzn2023.x86_64.rpm       46 MB/s | 4.7 MB    00:00  
  
Total                                         36 MB/s | 4.9 MB    00:00  
  
Running transaction check  
Transaction check succeeded.  
Running transaction test  
Transaction test succeeded.  
Running transaction  
  Preparing :                                                 1/1  
  Installing : stunnel-5.58-1.amzn2023.0.2.x86_64           1/2  
  Running scriptlet: stunnel-5.58-1.amzn2023.0.2.x86_64      1/2  
  Installing : amazon-efs-utils-2.4.0-1.amzn2023.x86_64       2/2  
  Running scriptlet: amazon-efs-utils-2.4.0-1.amzn2023.x86_64  2/2  
  Verifying   : amazon-efs-utils-2.4.0-1.amzn2023.x86_64       1/2  
  Verifying   : stunnel-5.58-1.amzn2023.0.2.x86_64            2/2  
  
Installed:  
amazon-efs-utils-2.4.0-1.amzn2023.x86_64                      stunnel-5.58-1.amzn2023.0.2.x86_64  
  
Complete!  
[ec2-user@ip-10-0-1-52 ~]$
```

```
[ec2-user@ip-10-0-1-52 ~]$ sudo mkdir efs  
[ec2-user@ip-10-0-1-52 ~]$ sudo mount -t efs -o tls fs-0570cbaecd7532b4f:/ efs  
[ec2-user@ip-10-0-1-52 ~]$ sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wszie=1048576,hard,timeo=600,retrans=2,noresvport fs-0570cbaecd7532b4f.efs.us-east-1.amazonaws.com:/ efs  
[ec2-user@ip-10-0-1-52 ~]$
```

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

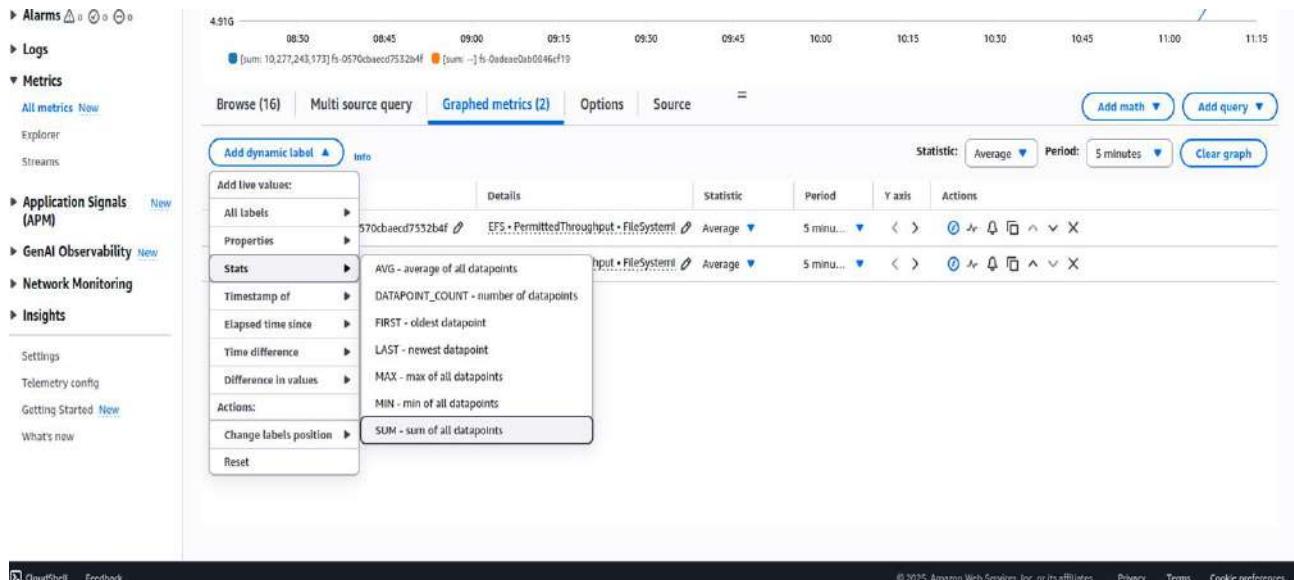
## Monitoring performance by using Amazon CloudWatch

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.

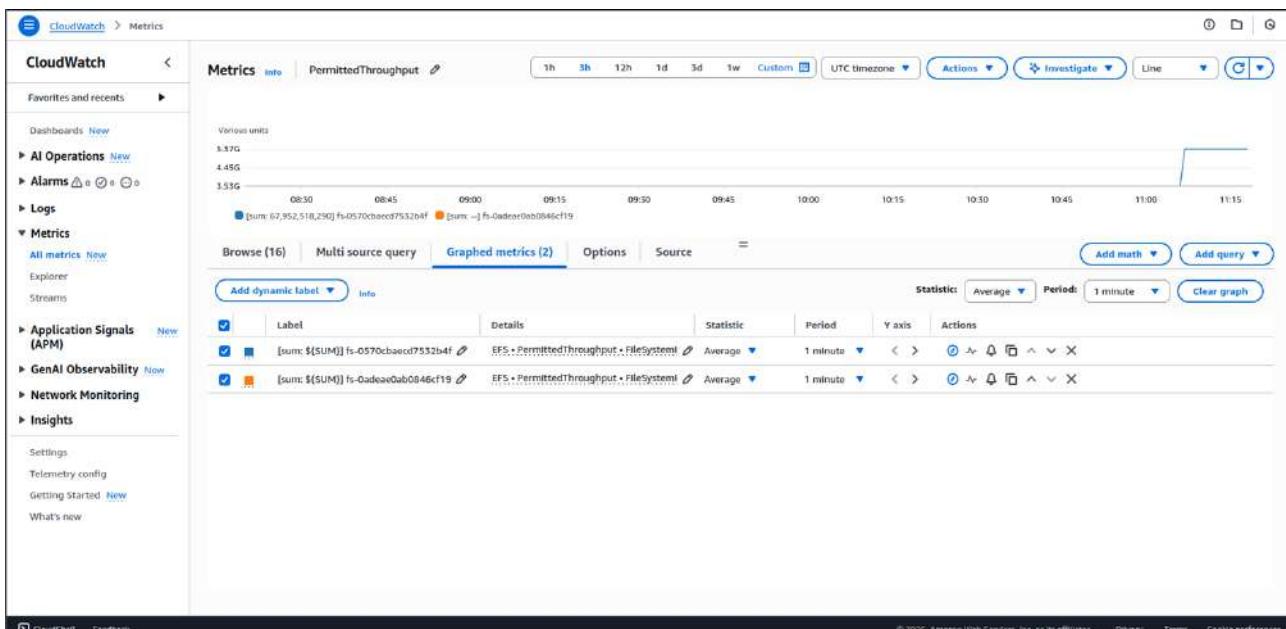


The screenshot shows the AWS CloudWatch Metrics interface. On the left, a navigation pane includes links for Alarms, Logs, Metrics, Application Signals (APM), GenAI Observability, Network Monitoring, and Insights. Under Metrics, the 'All metrics' tab is selected. The main area displays a list of metrics for two file systems: 'fs-0570cbacd7532b4f' and 'fs-0adee0dab0846cf19'. For each file system, there are several metrics listed: TotalIOBytes, PercentOLimit, PermittedThroughput (which is checked for both), MetadataReadIOBytes, ClientConnections, MetadataWriteIOBytes, TotalIOBytes (checked for the second file system), MetadataIOBytes, MeteredIOBytes, and PercentOLimit. The 'PermittedThroughput' metrics for both file systems are highlighted with a blue border. The top of the interface features a timeline from 08:15 to 11:35 and buttons for 'Add math' and 'Add query'.

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

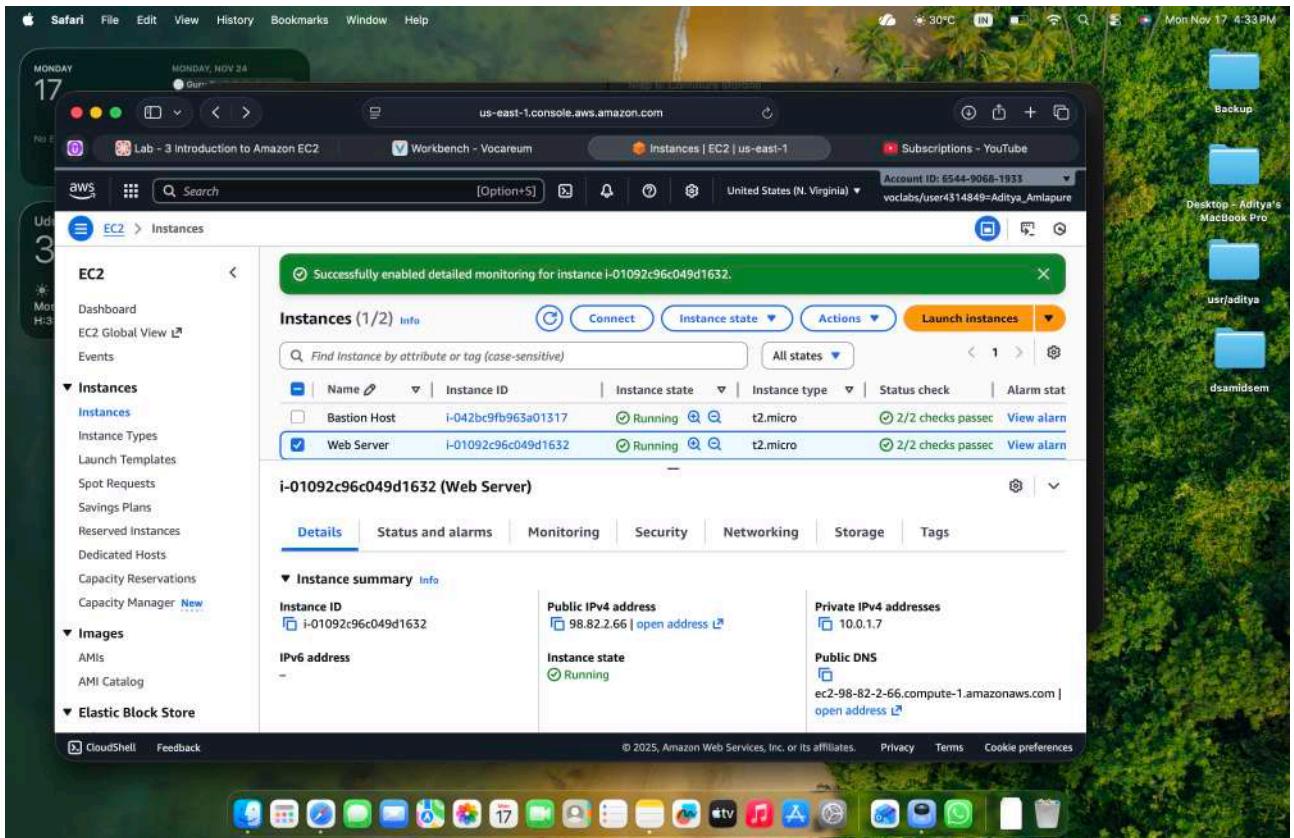
- user data box

```
#!/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

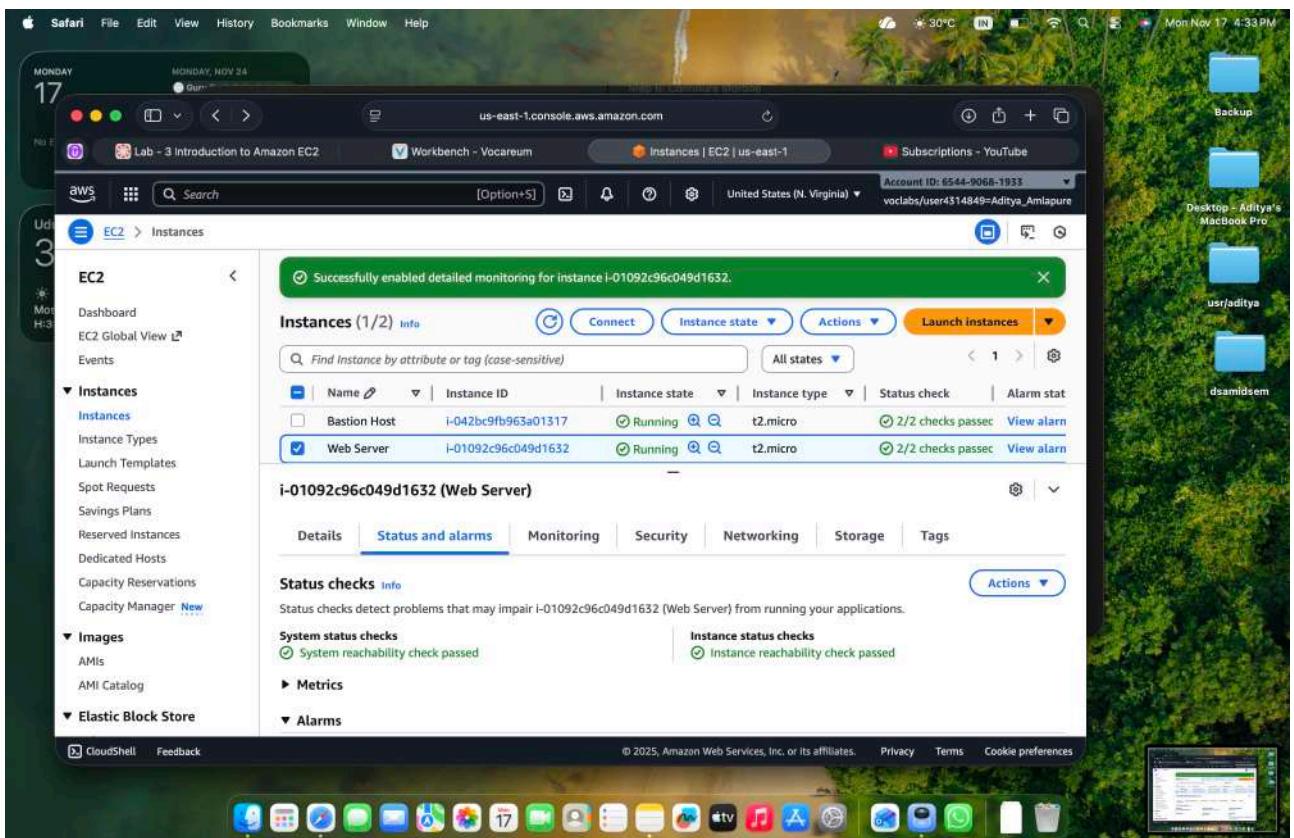
Step 8: Launch the instance



## Task 2: Monitoring Your Instance

Status checks tab

System reachability and Instance reachability



## Monitor and troubleshoot Get system log.

The screenshot shows the AWS CloudWatch Logs interface for an EC2 instance. The top navigation bar includes tabs for 'CloudTrail events', 'SSM command history', 'Reachability Analyzer - new', 'Instance events', 'Instance screenshot', and 'System log'. The 'System log' tab is selected. A sub-header reads: 'Logs are only retrieved after an instance lifecycle event (launch/reboot/terminate/hibernate)'. Below this, there are three status indicators: 'Instance state' (Running), 'System status check' (Check passed), and 'Instance status check' (Check passed). The main content area displays the system log output, which includes logs from cloud-init, SSH host key fingerprints, and the end of the SSH host key keys process. The log concludes with 'Cloud-init v. 22.2.2 finished at Mon, 17 Nov 2025 10:56:32 +0000. Datasource DataSourceEc2. Up 31.33 seconds'.

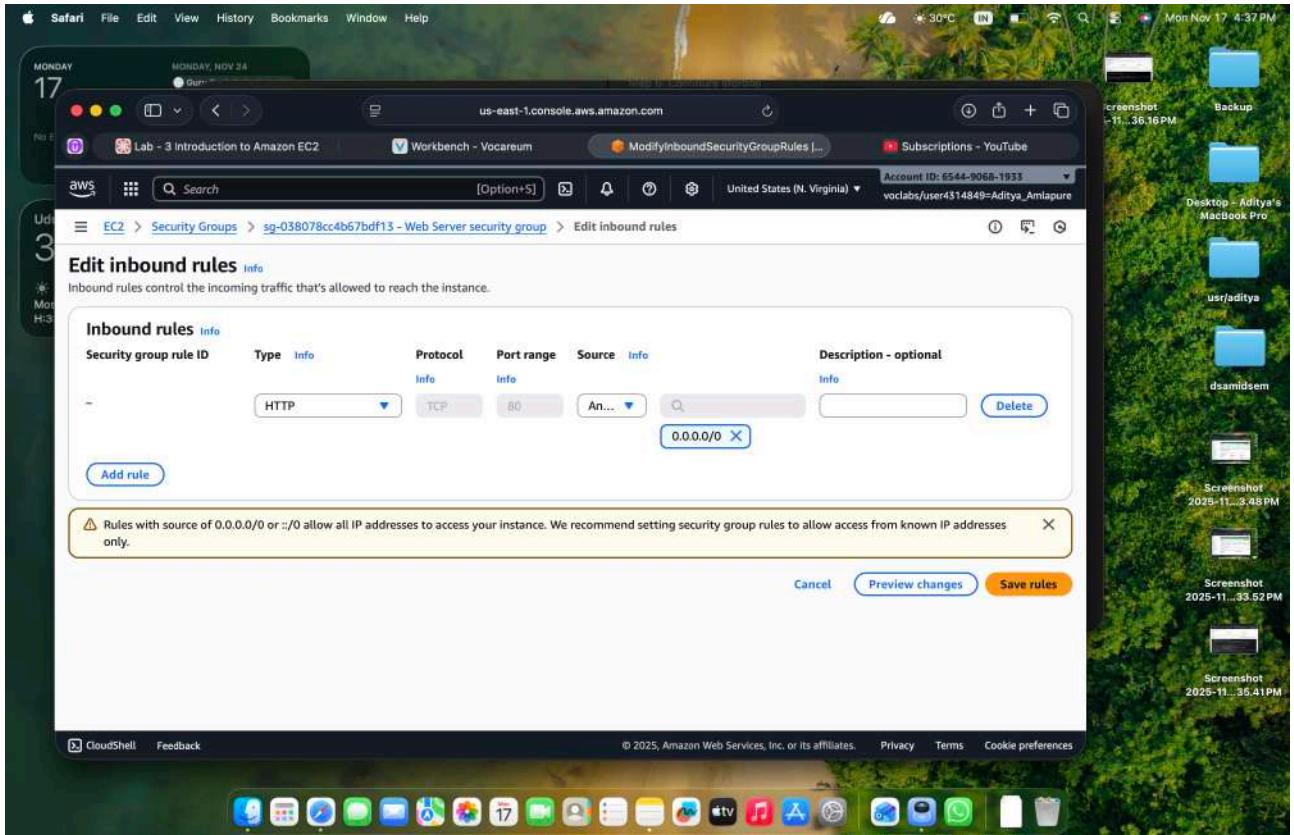
## Monitor and troubleshoot Get instance screenshot.

The screenshot shows the AWS Instance Diagnostics interface for an EC2 instance. The top navigation bar includes tabs for 'CloudTrail events', 'SSM command history', 'Reachability Analyzer - new', 'Instance events', 'Instance screenshot', and 'System log'. The 'Instance screenshot' tab is selected. A sub-header reads: 'Last updated November 17, 2025, 16:36 (UTC+05:30)'. Below this, there are three status indicators: 'Instance state' (Running), 'System status check' (Check passed), and 'Instance status check' (Check passed). The main content area displays the instance screenshot, which is a terminal window showing the boot logs of the Amazon Linux 2023.9.20251110 kernel. The logs indicate issues with zram memory limits and successful logins.

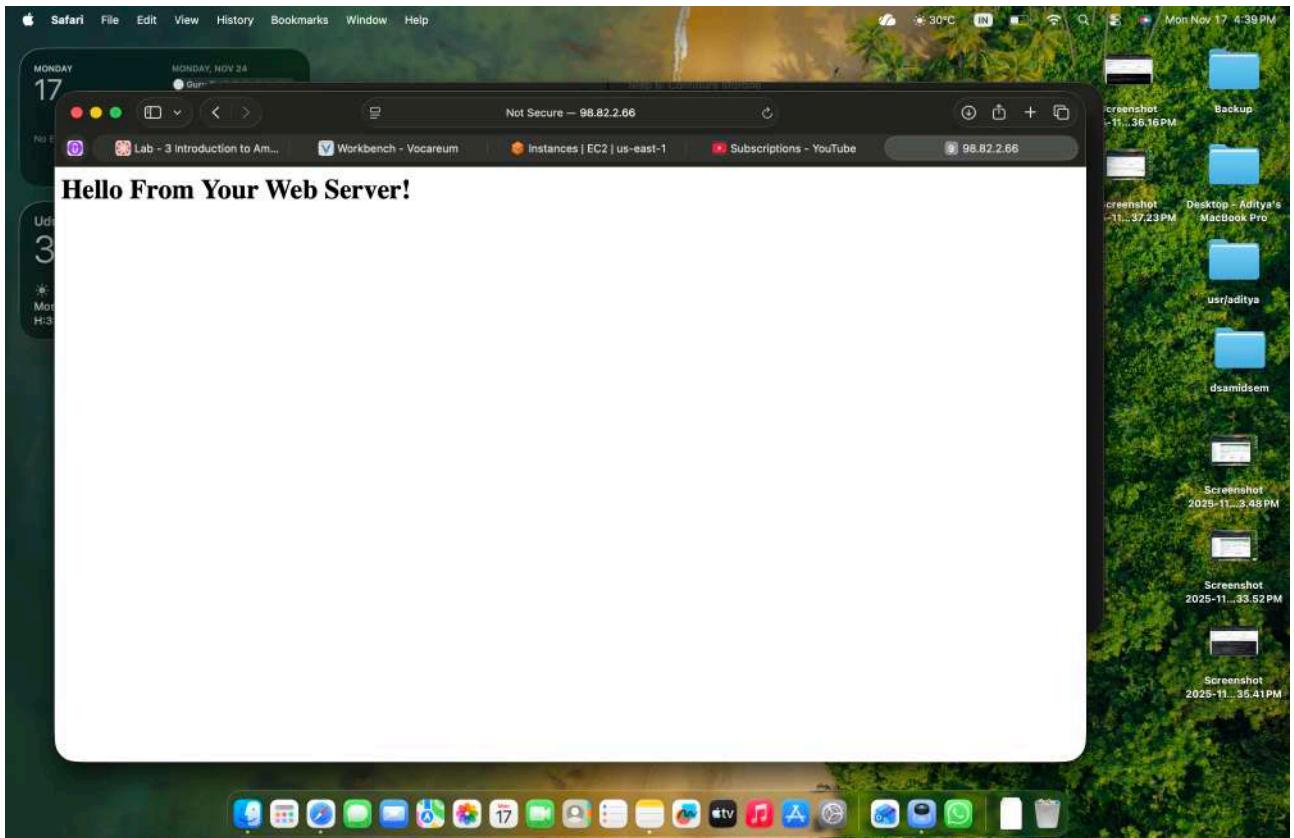
## Task 3: Update Your Security Group and Access the Web Server

## Web Server security group - Inbound rules - edit

- Type: HTTP
- Source: Anywhere-IPv4



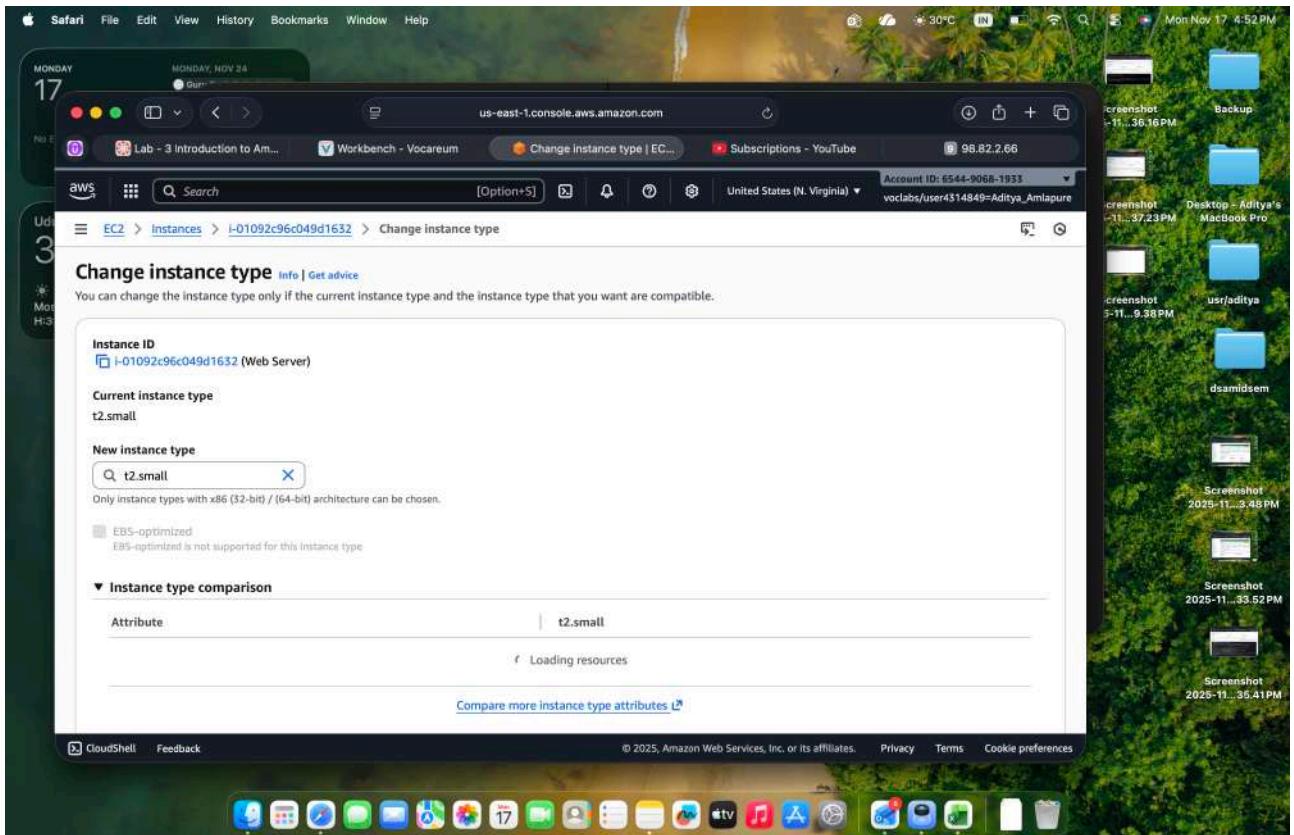
Return to the web server tab that you previously opened and refresh the page.  
You should see the message Hello From Your Web Server!

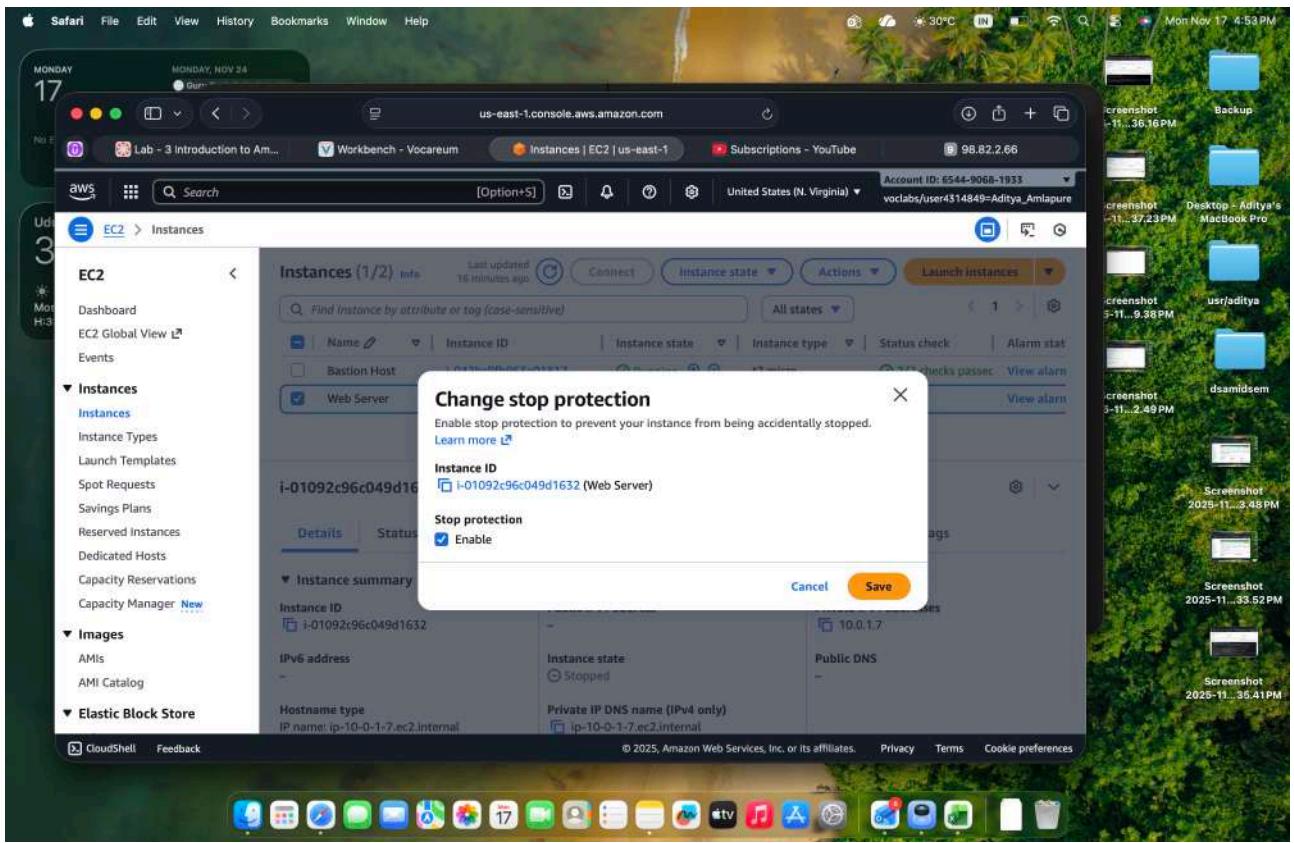


## Task 4: Resize Your Instance: Instance Type and EBS Volume

## Stop Your Instance

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



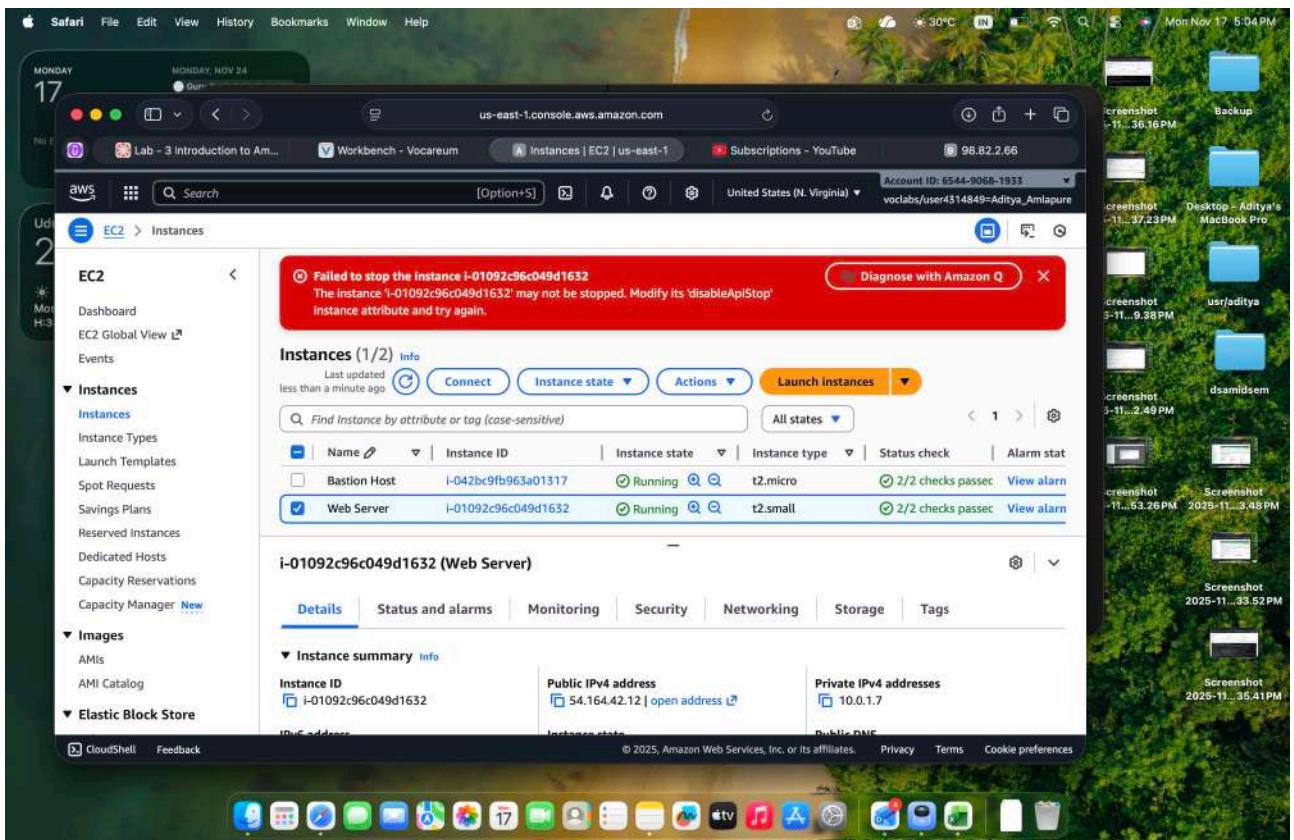


Resize the EBS Volume. 10

Start the stopped instance

Task 5: Explore EC2 Limits  
Service Quotas

Task 6: Test Stop Protection



# 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**.

**Create NAT gateway** Info

A highly available, managed Network Address Translation (NAT) service that instances in private subnets can use to connect to services in other VPCs, on-premises networks, or the internet.

**NAT gateway settings**

**Name - optional**  
Create a tag with a key of 'Name' and a value that you specify.  
  
The name can be up to 256 characters long.

**Subnet**  
Select a subnet in which to create the NAT gateway.

**Connectivity type**  
Select a connectivity type for the NAT gateway.  
 Public  
 Private

**Elastic IP allocation ID** Info  
Assign an Elastic IP address to the 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**.

The screenshot shows the 'Edit routes' page in the AWS Route53 console. It lists two routes for the destination 10.0.0.0/16. The first route has a target of 'local' and is active. The second route has a target of 'NAT Gateway' and is also active. Both routes are not propagated and have their route origin set to 'CreateRouteTable'. There is a 'Remove' button for the second route. At the bottom, there are buttons for 'Add route', 'Cancel', 'Preview', and '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**.

The screenshot shows the AWS EC2 Launch Templates page. On the left, there's a sidebar with navigation links like Dashboard, EC2 Global View, Instances, Launch Templates (selected), and others. The main area has a table titled "Launch Templates (1/1)" with one item listed:

Launch Template ID	Launch Template Name	Default Version	Latest Version	Create Time	Created By
lt-0fd12de1a72db7cc9	CafeWebServerTemplate	1	1	2025-11-18T07:10:38.000Z	arn:aws:sts::6374232...

Below the table, there's a detailed view for the "CafeWebServerTemplate" with tabs for Details, Versions, and Template tags. The Details tab shows the launch template ID, name, and version information. The Owner section indicates the template was created by a user with ARN arn:aws:sts::6374232... assumed role/vclabs/user4312508-VIKRAM\_TJM\_MANNA\_MADHYASTA.

## 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

The screenshot shows the AWS EC2 Load Balancers console. On the left, there's a navigation sidebar with options like Volumes, Snapshots, Lifecycle Manager, Network & Security, Load Balancing, Auto Scaling, and Settings. The main area displays a table titled 'Load balancers (1/1)'. The table has columns for Name, State, Type, Scheme, IP address type, VPC ID, and Availability Zones. One row is selected, showing 'CafeALB' as the name, 'Active' as the state, 'application' as the type, 'Internet-facing' as the scheme, 'IPv4' as the IP address type, 'vpc-091f9a3f317680236' as the VPC ID, and '2 Availability Zones'.

## 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**.

The screenshot shows the AWS CloudWatch Metrics interface. A line graph tracks CPU utilization across multiple instances over a period of approximately 24 hours. The Y-axis represents CPU utilization from 0% to 100%, and the X-axis shows time intervals. The data series fluctuates between 50% and 100% utilization.

## Task 5 — Testing and Validation

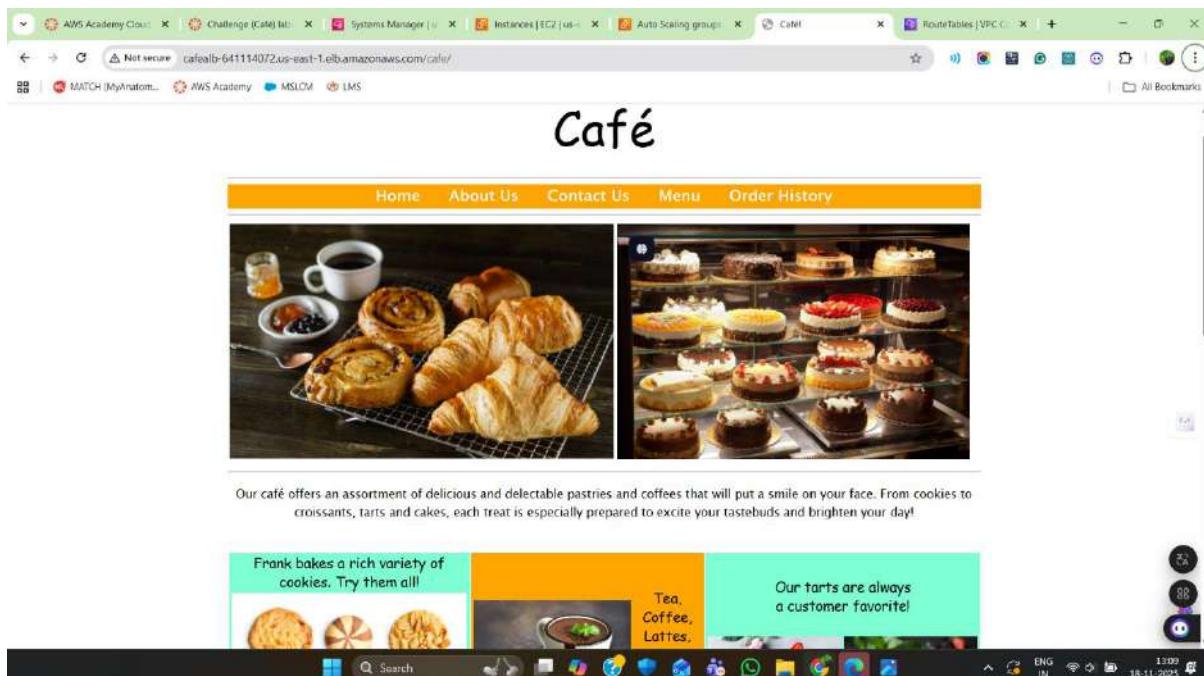
### 5.1 Application Reachability Test

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

```
installing:
stress                               x86_64                         1.0.4-16.el7
Transaction Summary
Install 1 Package
Total download size: 39 k
Installed size: 94 k
Downloading packages:
warning: /var/cache/yum/x86_64/2/epel/packages/stress-1.0.4-16.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID 352c64e5: NOKEY
Public key for stress-1.0.4-16.el7.x86_64.rpm is not installed
stress-1.0.4-16.el7.x86_64.rpm
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KRY-EPEL-7
Importing GPG key 0x352C64E5:
  Userid : "Fedora EPEL (7) <epel@fedoraproject.org>"
  Fingerprint: 91e9 7d7c 435e 9ef1 7f3e 88ff 6a2f aea2 352c 64e5
  Package : epel-release-7-11.noarch (#amzn2extra-epel)
  From    : /etc/pki/rpm-gpg/RPM-GPG-KRY-EPEL-7
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : stress-1.0.4-16.el7.x86_64
  Verifying   : stress-1.0.4-16.el7.x86_64
1/1
1/1
Installed:
  stress.x86_64 0:1.0.4-16.el7
Complete!
sh: 4:29 stress --cpu 1 --timeout 600
stress: info: [3367] dispatching hogs: 1 cpu, 0 io, 0 vm, 0 hdd
```

The screenshot shows the AWS CloudWatch Metrics Insights interface. A query is being run against the AWS CloudWatch Metrics service. The query is as follows:

```
CloudWatchMetricsQuery {
    MetricFilterName: "CPUUtilizationFilter"
    Metrics: [
        {
            MetricName: "CPUUtilization",
            Namespace: "AWS/EC2"
        }
    ],
    StartTime: "2018-06-01T00:00:00Z",
    EndTime: "2018-06-01T01:00:00Z",
    Statistics: [
        "Average"
    ],
    Period: 300
}
```

The results show a single data series named "CPUUtilization" with a single data point. The value is 1.0, which corresponds to the 25% CPU utilization threshold mentioned in the text above.

**Instances (4) Info**

Instances (4) Info									
		Last updated less than a minute ago		Actions		Launch Instances			
Name		Instance ID		Instance state		Status check		Alarm status	
		i-045869f27fb393563	i-045869f27fb393563	Running	Q Q	t2.micro	2/2 checks passed	View alarms +	us-east-1a
	webserver	i-0cdc809a1a99aa173	i-0cdc809a1a99aa173	Shutting-down	Q C	t2.micro	-	View alarms +	us-east-1a
	webserver	i-0f94678aeba34169a	i-0f94678aeba34169a	Running	Q Q	t2.micro	2/2 checks passed	View alarms +	us-east-1b
	CafeWebAppS...	i-002df20ccf01c176d	i-002df20ccf01c176d	Running	Q Q	t2.micro	2/2 checks passed	View alarms +	us-east-1a

**Auto Scaling groups**

Auto Scaling groups (1) Info									
		Last updated less than a minute ago		Actions		Create Auto Scaling group			
Name		Launch template/configuration		Instances		Status		Desired capacity	
	CafeWebServerASG	CafeWebServerTemplate	Version Default	6		Updating capacity...	3	2	6

**Instances (9) Info**

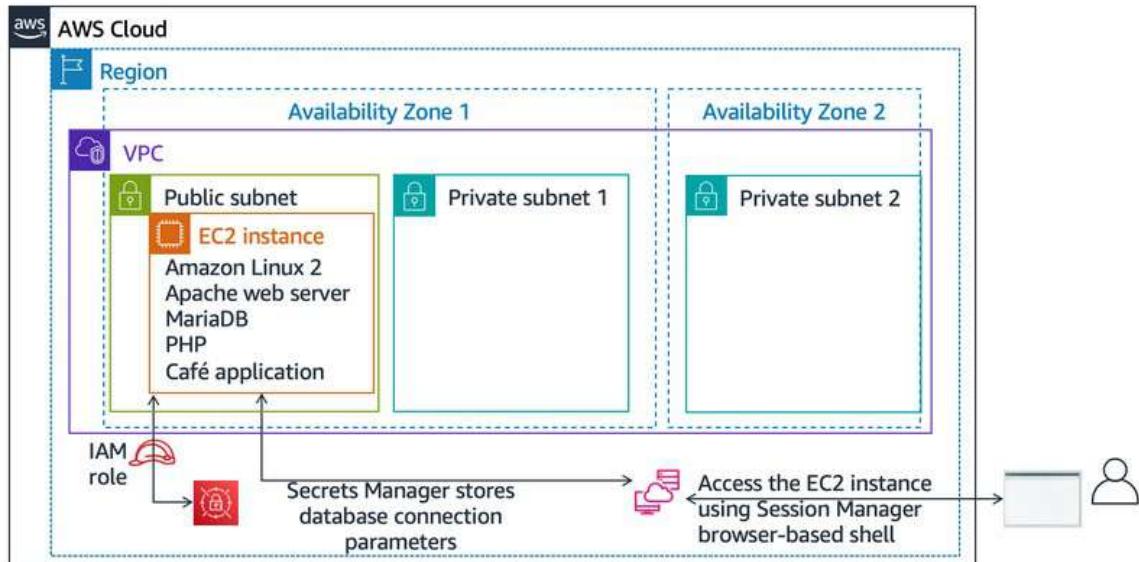
Instances (9) Info									
		Last updated less than a minute ago		Actions		Launch instances			
Name		Instance ID		Instance state		Status check		Alarm status	
		i-045869f27fb393563	i-045869f27fb393563	Running	Q Q	t2.micro	2/2 checks passed	View alarms +	us-east-1a
	webserver	i-0cdc809a1a99aa173	i-0cdc809a1a99aa173	Terminated	Q Q	t2.micro	-	View alarms +	us-east-1a
	webserver	i-0f94678aeba34169a	i-0f94678aeba34169a	Terminated	Q Q	t2.micro	-	View alarms +	us-east-1b
	webserver	i-0763a15c42aeed3fd	i-0763a15c42aeed3fd	Running	Q Q	t2.micro	2/2 checks passed	View alarms +	us-east-1b
	webserver	i-0b36ce2b70e23b2d2	i-0b36ce2b70e23b2d2	Running	Q Q	t2.micro	Initializing	View alarms +	us-east-1b
	webserver	i-0ac8d48636c050975	i-0ac8d48636c050975	Running	Q Q	t2.micro	Initializing	View alarms +	us-east-1b
	CafeWebAppS...	i-002df20ccf01c176d	i-002df20ccf01c176d	Running	Q Q	t2.micro	2/2 checks passed	View alarms +	us-east-1a
	webserver	i-0fe6bb908feb56ff	i-0fe6bb908feb56ff	Running	Q Q	t2.micro	Initializing	View alarms +	us-east-1a
	webserver	i-078dd64e1b4e6eae0	i-078dd64e1b4e6eae0	Running	Q Q	t2.micro	Initializing	View alarms +	us-east-1a

## Conclusion

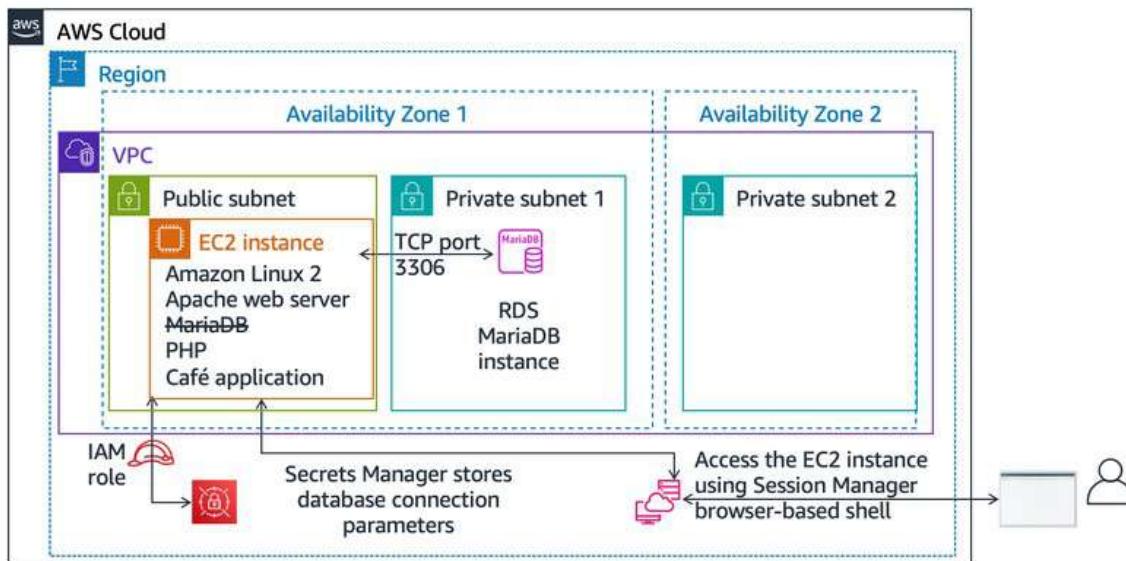
This lab successfully demonstrated the deployment of a highly available, fault tolerant, and scalable web application architecture using core AWS services. A multi-AZ design combined with NAT Gateways, private subnets, an ALB, and an Auto Scaling Group ensures robust performance and resiliency.

# 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**.

The screenshot shows the Aurora and RDS Dashboard. On the left sidebar, there are sections for Aurora and RDS, including Databases, Query editor, Performance insights, Snapshots, Exports in Amazon S3, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Custom engine versions, Zero-ETL integrations, Events, Event subscriptions, Recommendations (0), and Certificate update.

The main content area is titled "Resources" and displays usage statistics for the US East (N. Virginia) region:

- DB Instances (0/40): Allocated storage (0 TB/100 TB)
- DB Clusters (0/40): Instances and storage include Neptune and DocumentDB. [Increase DB instances limit](#)
- Reserved Instances (0/40)
- Snapshots (0)
- Manual: DB Cluster (0/100), DB Instance (0/100)
- Automated: DB Cluster (0), DB Instance (0)
- Recent events (0)
- Event subscriptions (0/20)

Parameter groups (1): Default (1), Custom (0/100)

Option groups (1): Default (1), Custom (0/20)

Subnet groups (1/50): Default network vpc-0b7ef057d9d053d25

Supported platforms VPC

A "Create a database" section is present with a "Create a database" button and a note: "Note: your DB instances will launch in the US East (N. Virginia) region". It also mentions: "You can use a backup from Amazon S3 to restore and create a new Aurora MySQL and MySQL database." A "Restore from S3" button is available.

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** (unchecked)

#### **10. Monitoring: Uncheck **Enable Enhanced monitoring****

## 11. Click **Create database**.

Aurora and RDS > Databases > Create database

### Create database Info

Choose a database creation method

Standard create  
Use all of the configuration options, including ones for availability, security, backups, and maintenance.

Easy create  
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

**Engine options**

Engine type Info

Aurora (MySQL Compatible) 

Aurora (PostgreSQL Compatible) 

MySQL 

PostgreSQL 

MariaDB 

Oracle 

Microsoft SQL Server 

IBM DB2 

**Edition**

MySQL Community

Engine version Info

View the engine versions that support the following database features.

Hide filters

Show only versions that support the Multi-AZ DB cluster Info  
Create a Multi-AZ DB cluster with one primary DB instance and two read-only standby DB instances. Multi-AZ DB clusters provide up to 2x better transaction commit latency and automatic failover in (typically under 35 seconds).

Show only versions that support the Amazon RDS Optimized Writes Info  
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Engine version

MySQL 5.6.45

**Engine version**

MySQL 8.0.43 [View details](#)

Enable RDS Extended Support [Info](#)  
Amazon RDS Extended Support is a paid offering. By selecting this option, you consent to being charged for this offering if you are running your database major release and the RDS-end-of-standard support date for that version. Check the end of standard support date for your version in the EOL for MySQL documentation.

---

**Templates**  
Choose a sample template to insert your own name.

**Production**  
Use defaults for high availability and fast, consistent performance.

**Dev/Test**  
This instance is intended for development use outside of a production environment.

**Free Tier**  
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.

---

**Availability and durability**

**Deployment options** [Info](#)  
Choose the deployment option that provides the availability and durability needed for your use case. AWS is committed to a certain level of uptime depending on the deployment option you chose. Learn more in the Amazon RDS service level agreement (SLA).

**Single-AZ DB instance deployment (1 instance)**  
Optimal for a single DB instance without durability increases. This setup provides:

- 99.99% uptime
- 1 Availability Zone



**Multi-AZ DB instance deployment (2 instances)**  
Optimal for a single DB instance with durability increases. This setup provides:

- 99.999999999% uptime
- 2 Availability Zones (AZ-1 and AZ-2)
- Read-only secondary instance
- Automatic failover



**Multi-AZ DB cluster deployment (3 instances)**  
Optimal for a multi-client application that needs cross-region failover and durability in multiple Availability Zones. This setup provides:

- 99.999999999% uptime
- 3 Availability Zones (AZ-1, AZ-2, AZ-3)
- Read-only secondary instances
- Improved fault tolerance
- Automatic failover



---

**Settings**

**DB instance identifier** [Info](#)  
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current RDS Region.  
  
The DB instance identifier is case-insensitive, but it stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 12 alphanumeric characters or hyphens; first character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

**Credential Settings**

**Master username** [Info](#)  
Type a login ID for the master user of your DB instance.  
  
1 to 16 alphanumeric characters. The first character must be a letter.

**Credentials management**  
You can use AWS Secrets Manager or manage your master user credentials.

**Managed in AWS Secrets Manager - root user**  
AWS generates a password for your root manager in three phases (its lifecycle) using AWS Secrets Manager.

**Auto generate password**  
Amazon RDS can generate a password for you, or you can specify your own password.

**Master password** [Info](#)  
  
Minimum 8 characters. Must contain at least 3 of the following categories: Uppercase letters, Lowercase letters, Numbers, Special characters. Can't contain any of the following symbols: / " @

**Confirm master password** [Info](#)

## Instance configuration

The DB Instance configuration options below are limited to those supported.

#### ▼ Hide filters

#### Show instance classes that support Amazon RDS

**Avanza: PDS Optimizado** (Mejor rendimiento en la fotografía)

### Include previous generation classes

**Storage**

**Storage type:** Info  
Recommended: IOPS SSD (io2) storage volumes are now available.

**Allocated storage:** Info  
20 GB  
Present: 20 GB, Available: 6,144 GB

**Provisioned IOPS:** Info  
1000  
Default IOPS of 2,000 IOPS is included for allocated storage less than 400 GiB.

**Storage throughput:** Info  
125 MiBps  
Default storage throughput of 125 MiBps is included for allocated storage less than 400 GiB.

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  
 Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

**Virtual private cloud (VPC):** Info  
 Choose the VPC. The VPC defines the virtual networking environment for this DB instance.  
 Let VPC (vpc-05190d5e72f0e199)  
 3 Subnets, 3 Availability Zones.

**DB subnet group:** Info  
 Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instances can use in the VPC that you selected.  
 db-subnet-group  
 3 Subnets, 3 Availability Zones.

**Public access:** Info  
 No  
 RDS doesn't assign a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

**VPC security group (Network):** Info  
 Choose one or more VPC security groups to allow access to your database. Make sure that the security groups allow the appropriate incoming traffic.  
 Choose existing  
 Choose existing VPC security groups.

**Existing VPC security groups:**  
 Choose one or more options  
 db05 X

**Availability Zone:** Info  
 No preference

**RDS Proxy:**  
 RDS Proxy is a fully managed, highly available database proxy that improves application availability, reliability, and security.  
 Create an RDS Proxy  
 RDS automatically creates an IAM user and a Lambda function for the proxy. RDS Proxy has additional costs. For more information, see Amazon RDS Proxy pricing.

**Certificate authority - optional:** Info  
 Using a server certificate provides an extra layer of security by validating that the connection is being made to an Amazon database. It does so by checking the server certificate that is automatically installed in all databases that you provision.

**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 10 more tags.

**Database authentication:**

**Database authentication options:** Info  
 Password authentication  
 Authorizes using database password.  
 Password and IAM database authentication  
 Authorizes using the database password and credentials through AWS IAM users and roles.  
 Password and Kerberos authentication  
 Choose a directory in which you want to allow root users to authenticate with their old password using 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.

**Additional monitoring settings:**  
 Enhanced Monitoring, CloudWatch Logs and DevOps Guru.

**Enhanced Monitoring:**  
 Enable Enhanced monitoring  
 Enabling Enhanced Monitoring metrics are useful when you want to see how different processes or threads use the CPU.

**Log exports:**  
 Audit log  
 Error log  
 General log  
 transaction-error log  
 Slow query log

**IAM role:**  
 The following service-linked role is used for publishing logs to CloudWatch Logs.  
 AWS/rds-db-logs

**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.

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.

**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
```

The screenshot shows the AWS Systems Manager Session Manager landing page. The left sidebar contains navigation links for Review node insights, Explore nodes, Diagnose and remediate, Just-in-time node access, Settings, Node Tools (Compliance, Distributor, Fleet Manager, Hybrid Activations, Inventory, Patch Manager, Run Command, Session Manager, State Manager), Change Management Tools (Automation, Change Calendar, Change Manager, Documents, Maintenance Windows, Quick Setup), and Application Tools (AppConfig). The main content area features a "Session Manager" section with the sub-headline "Quickly and securely access your Windows and Linux instances". It includes a "How it works" section with three steps: 1. Configure your Instances to use Session Manager, 2. Assign user IAM policies to control instance access, 3. Specify account options for session logs. Below this is a "Why use Session Manager?" section with two sub-sections: "Improved security posture" (Avoid the need to set up or maintain bastion hosts, to open up inbound SSH or) and "Centralized access control" (Grant and revoke access control to instances from one location using AWS). To the right, there's a "Start a session" box with "Start Session" and "Configure Preferences" buttons, and a "Getting started" sidebar with links like "What is Session Manager?", "Set up Session Manager", "Set up session logging", "Set up session notifications", "Create and manage sessions", and "Monitor session activity". A "More resources" sidebar includes links for Documentation, FAQs, and Systems Manager forums.

The screenshot shows the "Specify target" step of the Session Manager wizard. The left sidebar is identical to the previous screenshot. The main content area has a breadcrumb trail: AWS Systems Manager > Session Manager > Start a session. The "Specify target" step asks to "Select an instance to connect to using Session Manager". It has three tabs: Step 1 - Specify target (selected), Step 2 - optional (Specify session document), and Step 3 - Review and launch. Under "Reason", there is a text input field "Enter reason" with the placeholder "This value can have up to 256 characters.". The "Target instances" section shows a table with one row selected: Instance name (CafeServer), Instance ID (i-0da4050609ed0f47d), Agent version (3.3.3050.0), Instance state (Running), Availability zone (us-east-1a), Platform (Amazon Linux). There are "Start session" and "Next" buttons at the bottom right.

Session ID: user4312509-Aman\_Arun\_Saini-ag8lgiprretjcfvrtdeoeo    [ ] shortcuts    Instance ID: i-0d4c030609ed0f47d

```
sh-4.2# sudo su
[root@cafeinstance bin]# su ec2-user
[ec2-user@cafeinstance bin]$ sudo mysql -u root -p
Enter password: 
```

## To retrieve password from Secrets Manager :

[aws](#) Search [Alt+F] United States (N. Virginia) [View details for this instance](#) [Terminate](#)

AWS Secrets Manager > Secrets

**Secrets**

Filter secrets by name, description, tag key, tag value, owning service or primary Region

Secret name	Description	Last retrieved (UTC)
rds-db-credentials/db-PQTIRULM82WIFH75JCPVNM2B84/admin/1763376770782	RDS database admin credentials for database-1	-
/cafe/dbPassword	-	-
/cafe/dbUser	-	-
/cafe/dbName	-	-
/cafe/dbUrl	-	-
/cafe/currency	-	-
/cafe/timeZone	-	-
/cafe/showServerInfo	-	-

[Store a new secret](#)

[AWS Secrets Manager](#) > [Secrets](#) > /cafe/dbPassword

**/cafe/dbPassword**

**Secret details**

Encryption key: aws/secretsmanager

Secret name: /cafe/dbPassword

Secret ARN: arn:aws:secretsmanager:us-east-1:199230262122:secret:/cafe/dbPassword-a2tbGZ

Secret description: -

Secret type: -

**Actions**

**Overview** **Rotation** **Versions** **Replication** **Tags**

**Secret value** info Rehydrate and view the secret value.

Key/value **Plaintext**

Re:Start!9

**Resource permissions - optional** info Add or edit a resource policy to access secrets across AWS accounts.

**Edit permissions**

**Sample code**

## In Session Manager after entering into mysql :

- > show Databases;
- > use cafe\_db;
- > show tables;

```
Session ID: use4312509=Aman_Arun_Sanil-ag8lg4prretjcfvrdeloe  Instance ID: i-0d4c030609ed0f47d
sh-4.2$ sudo su
[root@cafereserver bin]# su ec2-user
[ec2-user@cafereserver bin]# sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11
Server version: 10.2.30-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\e' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| cafe_db |
| information_schema |
| mysql |
| performance_schema |
| test |
+-----+
5 rows in set (0.00 sec)

MariaDB [(none)]> use cafe_db;
Reading table information for compilation of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [cafe_db]> show tables;
+-----+
| Tables_in_cafe_db |
+-----+
| order |
| order_item |
| product |
| product_group |
+-----+
4 rows in set (0.00 sec)

MariaDB [cafe_db]> exit
Bye
[ec2-user@cafereserver bin]#
```

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

The screenshot shows the AWS EC2 Security Groups page. On the left, there's a navigation sidebar with links like Dashboard, Instances, Images, and Network & Security. The main area shows a security group named "sg-03dc4490dd9dad09e - dbSG". It has a "Details" section with fields for Security group name (dbSG), Owner (992382621222), Security group ID (sg-03dc4490dd9dad09e), Description (dbSG), and VPC ID (vpc-dbae965cd93305dbe). Below this is an "Inbound rules" section with a table header for Name, Security group rule ID, IP version, Type, Protocol, Port range, Source, and Description. A note at the bottom says "No security group rules found".

## select MYSQL/AURORA → CUSTOM → CAFESG

This screenshot shows the "Edit inbound rules" dialog for the "sg-03dc4490dd9dad09e - dbSG" security group. It has tabs for Inbound rules, Outbound rules, Sharing - new, VPC associations - new, and Tags. Under Inbound rules, there's a table with columns for Name, Security group rule ID, IP version, Type, Protocol, Port range, Source, and Description. A note at the top says "Inbound rules control the incoming traffic that's allowed to reach the instance." At the bottom, there are buttons for Add rule, Cancel, Preview changes, and Save rules.

```
(ec2-user@cafereserver ~]$ cd
(ec2-user@cafereserver ~)$ mysql -u root -p > cafe.sql
InnoDB: Using unique option prefix 'database' is error-prone and can break in the future. Please use the full name 'databases' instead.
Enter password:
```

## In RDS, Copy the RDS Endpoint :

```
[ec2-user@cafereserver ~]$ ls | grep cafe1.sql
cafe1.sql
[ec2-user@cafereserver ~]$ mysql -h database-1.ck2wa6m7pz.us-east-1.rds.amazonaws.com -u admin -p
```

// here paste the rds endpoint and password as Msis1234

```

Session ID: user4312509-Arun_Soum_agn8lg4przrtjchvsrlde0e Instance ID: i-0d4c030609ed0f47d
[ec2-user@cafeserver ~]$ ls | grep cafe1.sql
cafe1.sql
[ec2-user@cafeserver ~]$ mysql -h database-1.clikwa68n7px.us-east-1.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \q.
Your MySQL connection id is 28
Server version: 8.0.43 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> Create database cafe_db;
Query OK, 1 row affected (0.01 sec)

MySQL [(none)]> use cafe_db;
Database changed
MySQL [cafe_db]> source cafe1.sql;
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected, 1 warning (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 1 row affected (0.01 sec)
Database changed
Query OK, 0 rows affected (0.01 sec)
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected, 1 warning (0.00 sec)

[ec2-user@cafeserver ~]$

```

```

MySQL [cafe_db]> show tables;
+-----+
| Tables_in_cafe_db |
+-----+
| order_item          |
| product             |
| product_group       |
+-----+
4 rows in set (0.00 sec)

MySQL [cafe_db]> exit
Bye
[ec2-user@cafeserver ~]$

```

in mysql run the following 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

```

```

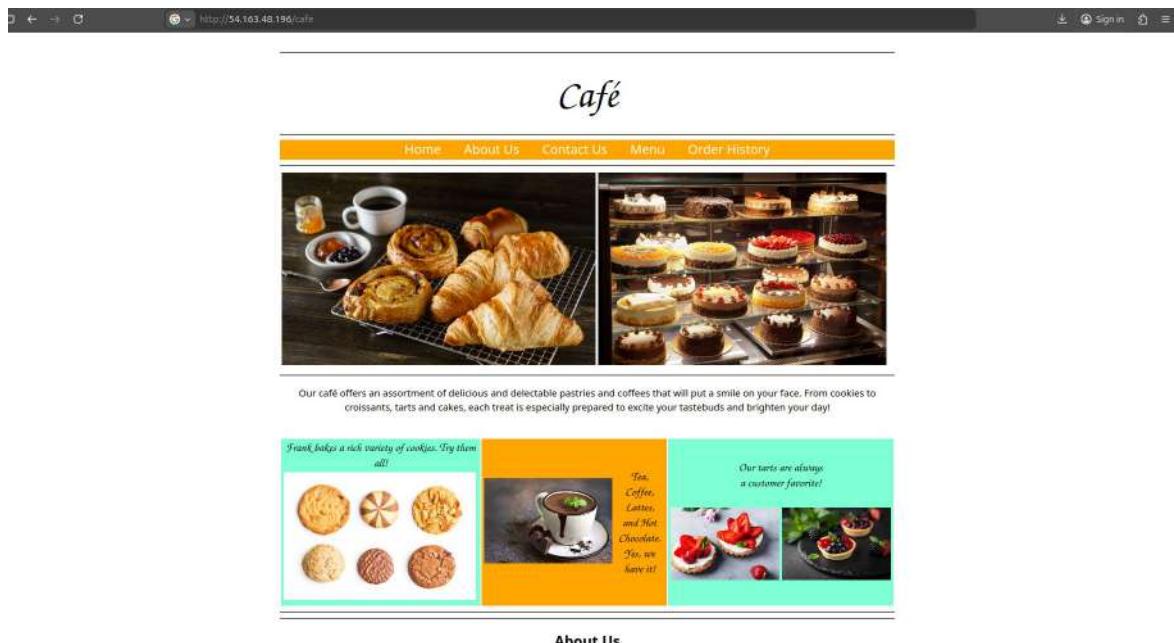
Session ID: user4312509-Arun_Soum_agn8lg4przrtjchvsrlde0e Instance ID: i-0d4c030609ed0f47d
[ec2-user@cafeserver ~]$ sudo systemctl stop mariadb.service
[ec2-user@cafeserver ~]$ sudo systemctl restart httpd
[ec2-user@cafeserver ~]$ exit

```

## 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

A screenshot of the AWS Secrets Manager 'Edit secret value' dialog. The URL is /cafe/dbUrl. The 'Plaintext' tab is selected. The value entered is database-1.clk2ws68s7px.us-east-1.rds.amazonaws.com. At the bottom right are 'Cancel' and 'Save' buttons. Below the dialog, there is a note: 'Your secret value has been updated.' and a timestamp: 'e2c-54-183-48-196.compute-1.amazonaws.com - [IP] - [Time]'.

aws Actions

Amazon Secrets Manager > Secrets > /cafe/dbPassword

### /cafe/dbPassword

Secret details

Edit secret value

Key/value Plaintext

Text	Line 1, Column 1	Errors: 0	Warnings: 0
<span style="border: 1px solid #0070C0; border-radius: 5px; padding: 2px 10px; color: #0070C0;">Cancel</span> <span style="background-color: #0070C0; color: white; border-radius: 5px; padding: 2px 10px; border: none;">Save</span>			

Resource permissions - optional [View](#)  
Add or edit a resource policy to access secrets across AWS accounts.

[Edit permissions](#)

Sample code

aws Actions

Amazon Secrets Manager > Secrets > /cafe/dbUser

### /cafe/dbUser

Your secret value has been updated

Edit secret value

Key/value Plaintext

Text	admin	
<span style="border: 1px solid #0070C0; border-radius: 5px; padding: 2px 10px; color: #0070C0;">Cancel</span> <span style="background-color: #0070C0; color: white; border-radius: 5px; padding: 2px 10px; border: none;">Save</span>		

Resource permissions - optional [View](#)  
Add or edit a resource policy to access secrets across AWS accounts.

[Edit permissions](#)

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

//check the Menu and Order up and running

*Café*

[Home](#) [Menu](#) [Order History](#)

*Order History*

Order Number: 24 Date: 2020-07-28 Time: 13:14:07 Total Amount: \$35.00			
Item	Price	Quantity	Amount
Strawberry Blueberry Tart	\$3.50	4	\$14.00
Strawberry Tart	\$3.50	3	\$10.50
Latte	\$3.50	3	\$10.50

Order Number: 23 Date: 2020-07-28 Time: 13:12:54 Total Amount: \$6.00			
Item	Price	Quantity	Amount
Coffee	\$3.00	2	\$6.00

Order Number: 22 Date: 2020-07-21 Time: 13:13:47 Total Amount: \$33.50			
Item	Price	Quantity	Amount
Chocolate Chip Cookie	\$2.50	3	\$7.50
Strawberry Blueberry Tart	\$3.50	4	\$14.00
Coffee	\$3.00	4	\$12.00

Order Number: 21 Date: 2020-07-20 Time: 13:13:36 Total Amount: \$17.50			
Item	Price	Quantity	Amount
Latte	\$3.50	5	\$17.50

Order Number: 20 Date: 2020-07-18 Time: 13:13:27 Total Amount: \$14.00			
Item	Price	Quantity	Amount
Donut	\$1.00	5	\$5.00
Hot Chocolate	\$3.00	3	\$9.00



**Croissant**  
\$1.50  
Fresh, buttery and fluffy... Simply delicious!  
Quantity:



**Donut**  
\$1.00  
We have more than half-a-dozen flavors!  
Quantity:



**Chocolate Chip Cookie**  
\$2.50  
Made with Swiss chocolate with a touch of Madagascar vanilla  
Quantity:



**Muffin**  
\$3.00  
Banana bread, blueberry, cranberry or apple  
Quantity:



**Strawberry Blueberry Tart**  
\$3.50  
Bursting with the taste and aroma of fresh fruit  
Quantity:



**Strawberry Tart**  
\$3.50  
Made with fresh ripe strawberries and a delicious whipped cream  
Quantity:

# 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: Creating a Static Website for the Café

ACAv3EN-US-LT13-131613 > Assignments > Challenge (Cafe) lab: Creating a Static Website for the Café > Challenge (Cafe) lab: Creating a Static Website for the Café

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

Modules      Due No Due Date Points 29 Submitting an external tool

Discussions      Grades      Lucid (Whiteboard)

AWS

EN\_US

Submit      Submission Report      Grades

# 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, but the café's reputation is limited to people who have visited or who have heard about it from other café

◀ Previous      Next ▶

The screenshot shows a web browser window with the URL [https://awsacademy.instructure.com/courses/131613/assignments/151928/module\\_item\\_id=12596703](https://awsacademy.instructure.com/courses/131613/assignments/151928/module_item_id=12596703). The page title is "Challenge (Café) lab: Creating a Static Website for the Café". The assignment details are: Due: No Due Date, Points: 29, Status: Submitting an external tool. The assignment content includes a title "Challenge Lab: Creating a Static Website for the Café" and a scenario describing a family-owned café. Navigation buttons at the bottom are "Previous" and "Next".

Challenge (Café) lab: Creating a Static Website for the Café

Home      Modules      Discussions      Grades      Lucid (Whiteboard)

Due: No Due Date      Points: 29      Submitting an external tool

AWS      EN\_US

Submit      Submission Report      Grades

# 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, but the café's reputation is limited to people who have visited or who have heard about it from other café

◀ Previous      Next ▶

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

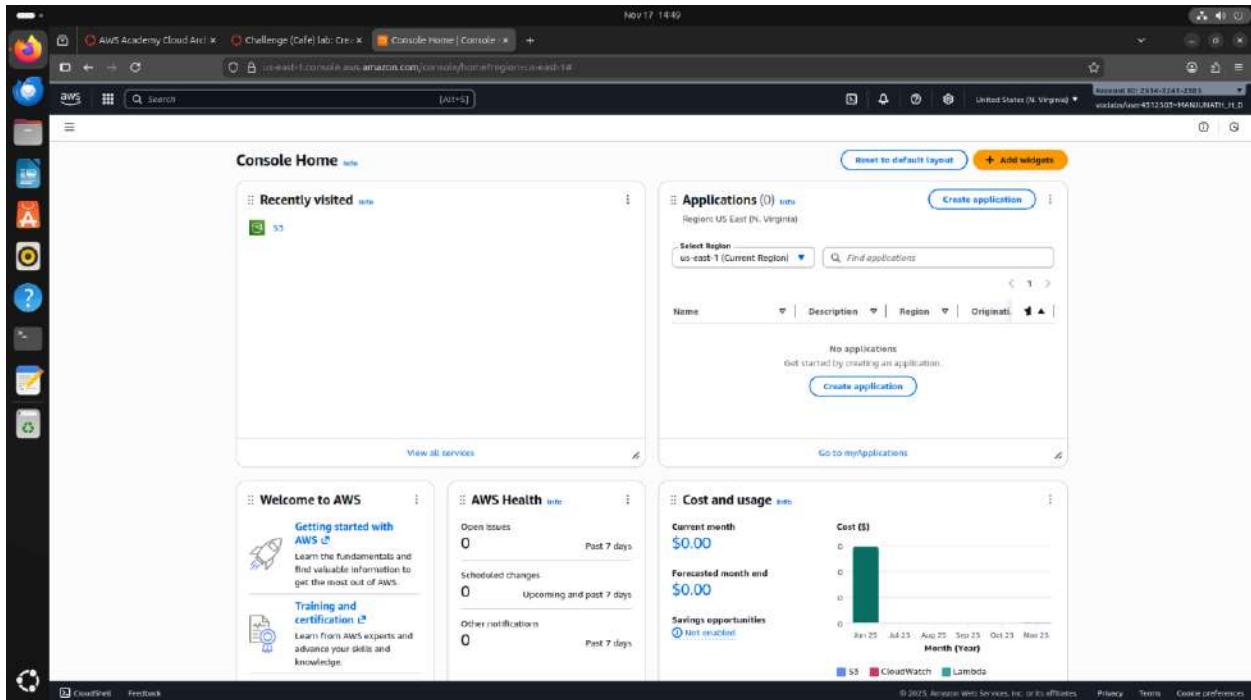
In this task, you create an S3 bucket and configure it to host your static website.

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.

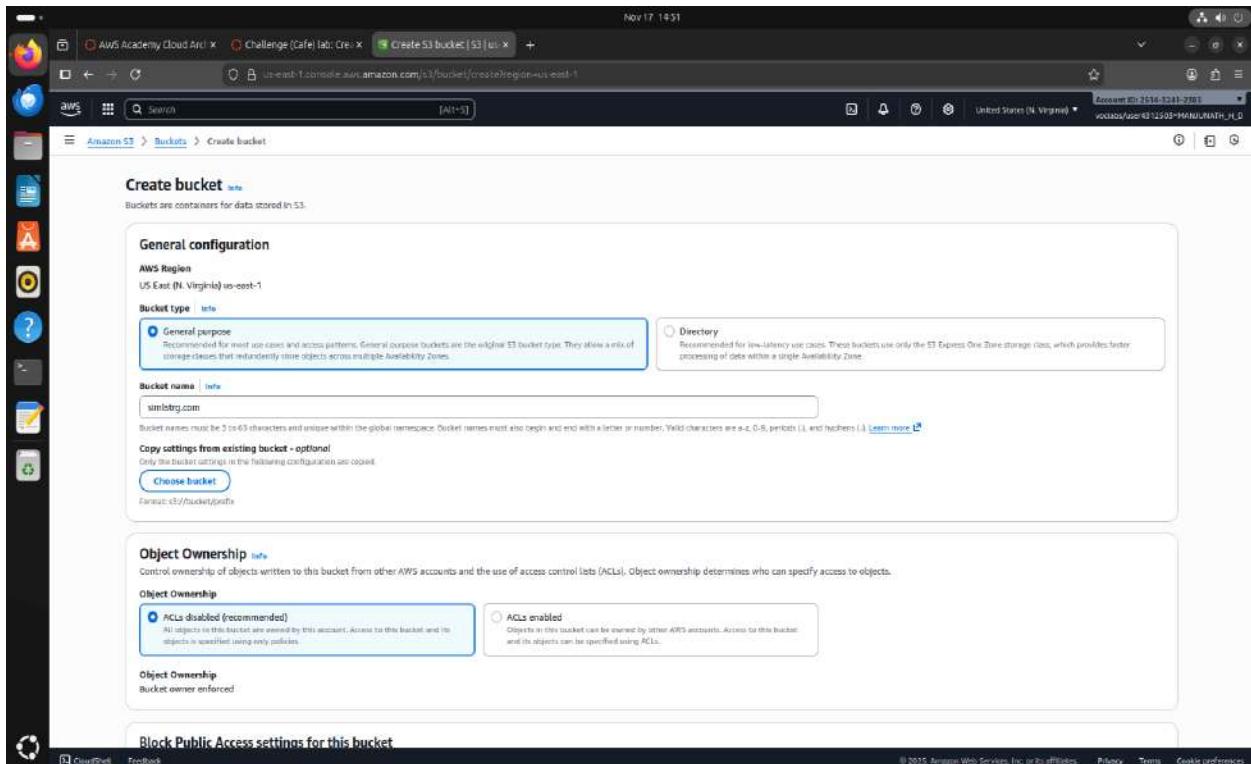
**Tip:** You must clear **Block all public access** and enable **ACLs**.

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.

The screenshot shows the 'Create S3 bucket' configuration page. At the top, there's a note about 'BLOCK ALL PUBLIC ACCESS SETTINGS FOR THIS BUCKET'. It explains that public access is granted through ACLs, bucket policies, or access point policies. To block all public access, you must turn on 'Block all public access'. There are four options under this setting:

- Block public access to buckets and objects granted through new access control lists (ACLs)
- Block public access to buckets and objects granted through any access control lists (ACLs)
- Block public access to buckets and objects granted through new public bucket or access point policies
- Block public and cross-account access to buckets and objects through any public bucket or access point policies

A warning message states: "Turning off block all public access might result in this bucket and the objects within becoming public. AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting." A checkbox labeled "I acknowledge that the current settings might result in this bucket and the objects within becoming public." is checked.

Below this, there's a section for 'Bucket Versioning' with a note: "Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures." A 'Disable' button is shown.

The 'Tags - optional (0)' section has a note: "You can use bucket tags to track storage costs and organize buckets." An 'Add new tag' button is available.

At the bottom right, there are links for 'Privacy', 'Terms', and 'Cookie preferences'.

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

The screenshot shows the 'Amazon S3 > Buckets' page. A green success message at the top says: "Successfully created bucket 'simlstrg.com'. To upload files and folders, or to configure additional bucket settings, choose View details." Below this, there are two tabs: 'General purpose buckets' (selected) and 'Directory buckets'. The 'General purpose buckets' tab shows a table with two entries:

Name	AWS Region	Creation date
e17f01d2a45058e1172880871762518324125	US East (N. Virginia) us-east-1	November 17, 2025, 14:20:40 (UTC+05:30)
simlstrg.com	US East (N. Virginia) us-east-1	November 17, 2025, 14:51:26 (UTC+05:30)

To the right of the table, there are sections for 'Account snapshot' (with a link to 'View dashboard') and 'External access summary - new' (with a link to 'View details').

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

The screenshot shows the AWS S3 'Upload' interface. At the top, there's a header with tabs for 'AWS Academy cloud href' and 'challenge (Cafe) lab: Create'. Below the header, the URL is <https://us-east-1.console.aws.amazon.com/s3/upload/s3mistrig.com?region=us-east-1>. The sidebar on the left shows various AWS services like CloudWatch, Lambda, and CloudFront. The main area is titled 'Upload' and shows a table of files and folders being uploaded to the 's3mistrig.com' bucket. The table includes columns for Name, Folder, Type, and Size. The files listed are:

Name	Folder	Type	Size
style.css	css/	text/css	541.0 B
Mom-&-Pop-Coffee-Shopping.png	images/	image/png	726.9 KB
Mom-&-Pop.png	images/	image/png	2.7 MB
Cake-Vitrine.png	images/	image/png	3.0 MB
Cup-of-Hot-Chocolate.png	images/	image/png	3.6 MB
Cookies.png	images/	image/png	1.4 MB
Strawberry-Tarts.png	images/	image/png	3.4 MB
Coffee-and-Pastries.png	images/	image/png	3.1 MB
Strawberry-&-Blueberry-Tarts.png	images/	image/png	2.9 MB
index.html	-	text/html	2.9 KB

Below the table, there's a section for 'Destination' with a dropdown set to 's3://s3mistrig.com'. There are sections for 'Destination details' (which says 'Bucket settings that impact new objects stored in the specified destination') and 'Permissions' (which says 'Grant public access and access to other AWS accounts'). At the bottom, there are buttons for 'Remove', 'Add files', and 'Add folder'.

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

Upload succeeded

For more information, see the Files and folders table.

View your progress today from the homepage, or reviewing information for a longer duration.

Summary	
Destination	Succeeded
s3://imlitrq.com	10 files, 21.7 MB [100.00%]
Pailed 0 files, 0 B (0%)	

**Files and folders (10 total, 21.7 MB)**

Name	Folder	Type	Size	Status	Error
style.css	css/	text/css	541.0 B	Successed	-
Mon-de-Pap-Coffee-Shop.png	images/	image/png	726.8 KB	Successed	-
Mon-de-Pap.png	images/	image/png	2.7 MB	Successed	-
Cake-Vitrine.png	images/	image/png	3.8 MB	Successed	-
Cup-of-Hot-Chocolate.png	images/	image/png	3.6 MB	Successed	-
Cookies.png	images/	image/png	1.4 MB	Successed	-
Strawberry-Tarts.png	images/	image/png	3.4 MB	Successed	-
Coffee-and-Pastries.png	images/	image/png	3.1 MB	Successed	-
Strawberry-&-Blueberry-Tarts.png	images/	image/png	2.9 MB	Successed	-
index.html	-	text/html	2.9 KB	Successed	-

> Uploaded sucessfully

Edit Object Ownership

**Object Ownership**

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

**ACLs disabled (recommended)**  
All objects in this bucket are owned by this account. Access to this bucket and its objects is governed using only policies.

**ACLs enabled**  
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

**We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.**

**Enabling ACLs turns off the bucket owner enforced setting for Object Ownership**  
Once the bucket owner enforced setting is turned off, access control lists (ACLs) and their associated permissions are restored. Access to objects that you do not own will be based on ACLs and not the bucket policy.

I acknowledge that ACLs will be restored.

**Object Ownership**

**Bucket owner preferred**  
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.

**Object writer**  
The object writer remains the object owner.

If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads. [Learn more](#)

[Cancel](#) [Save changes](#)

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

**Edit static website hosting**

**Static website hosting**

Use this bucket to host a website or redirect requests. [Learn more](#)

Disable

Enable

**Hosting type**

Host a static website

Use the bucket endpoint as the web address. [Learn more](#)

Redirect requests for an object

Redirect requests to another bucket or domain. [Learn more](#)

**For your customers to access content at the website endpoint, you must make all your content publicly readable.** To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#).

**Index document**

Specify the home or default page of the website.

index.html

**Error document - optional**

This is returned when an error occurs.

error.html

**Redirection rules - optional**

Redirection rules, written in JSON, automatically redirect requests for specific content. [Learn more](#)

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

**simlstrg.com**

**Objects (3/3)**

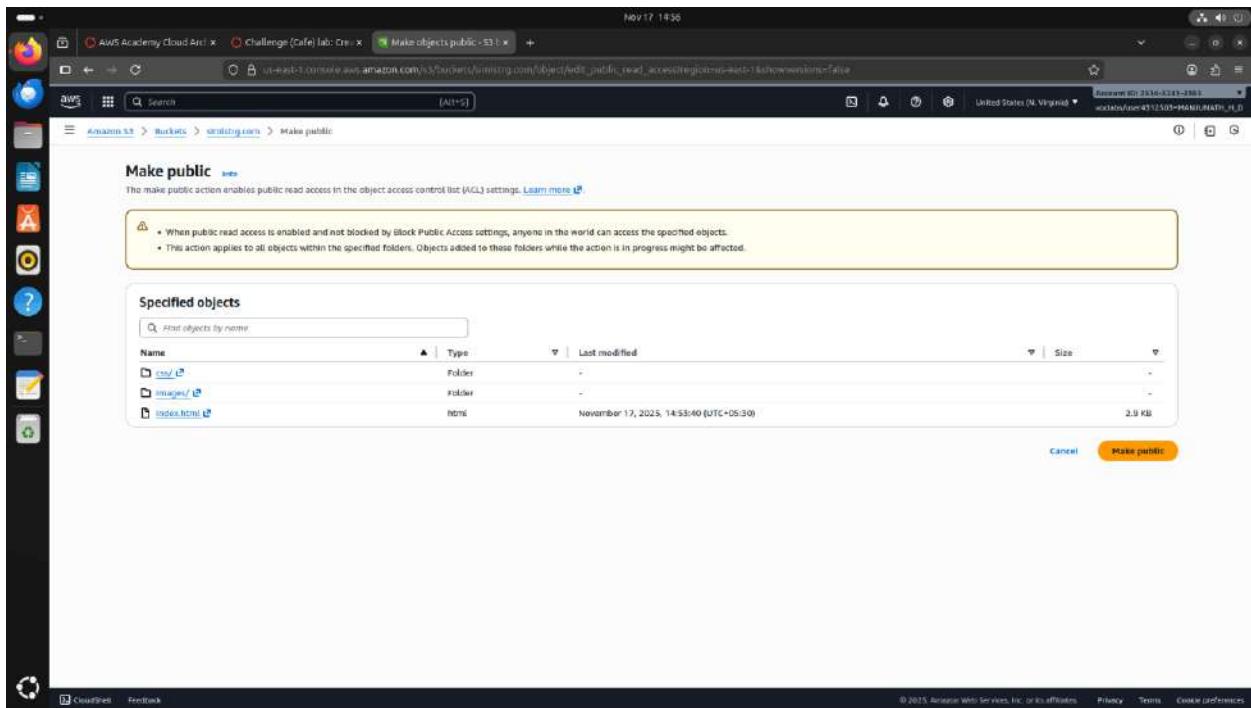
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to enable [public access](#).

Name	Type	Last modified	Size
static/	Folder		
images/	Folder		
index.html	HTML	November 17, 2025, 14:53:40 UTC+05:30	

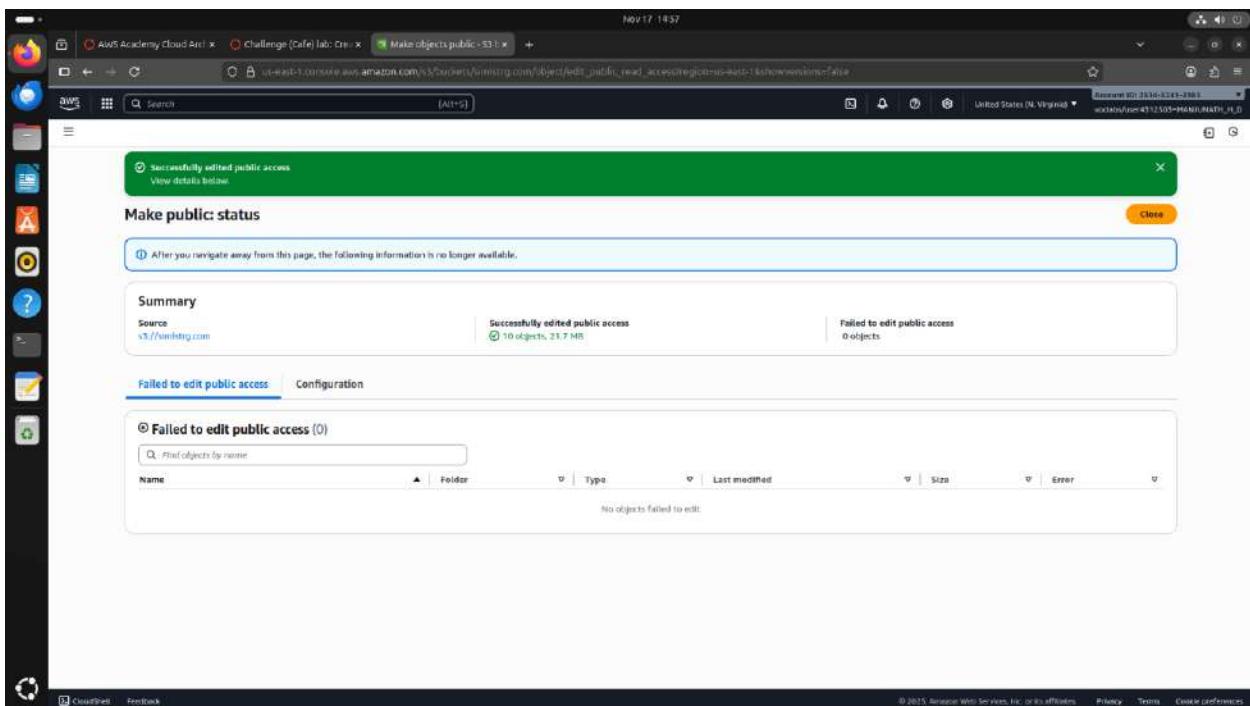
**Actions**

- Download as
- Create folder
- Upload
- Copy
- Move
- Metadata
- Query with S3 Select
- Edit actions
- Restore object
- Edit storage class
- Edit server-side encryption
- Set tags
- Make public using ACL

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.

Screenshot of the AWS S3 Bucket Properties page for 'simlstry.com-ss3-bucket'. The 'Static website hosting' section is highlighted.

**Object Lock**  
Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock works only in versioned buckets. [Learn more](#)

**Requester pays**  
When enabled, the requester pays for requests and data transfer costs, and anonymous access to this bucket is disabled. [Learn more](#)

**Static website hosting**  
Use this bucket to host a website or redirect requests. [Learn more](#)

We recommend using AWS Amplify Hosting for static website hosting  
Deploy a fast, secure, and reliable website quickly with AWS Amplify Hosting. Learn more about [Amplify Hosting](#) or [View your existing Amplify apps](#).

**SS static website hosting**  
Enabled

**Hosting type**  
Bucket hosting

**Bucket website endpoint**  
Once you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)

<http://simlstry.com.s3-website-us-east-1.amazonaws.com>

Screenshot of the static website hosted at <http://simlstry.com.s3-website-us-east-1.amazonaws.com>. The page title is 'Welcome to Mom & Pop Café'.

# Mom & Pop Café

Mom & Pop Café offers an assortment of delicious and delectable pastries and coffees that will put a smile on your face. From cookies to croissants, tarts and cakes, each treat is especially prepared to excite your tastebuds and brighten your day!

Pop bakes a rich variety of cookies. Try them all!

Tea  
Coffee  
Latte  
Flat  
Chocolate  
You may have it!

Our tarts are always a customer favorite!

[About Us](#)

> 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

The screenshot shows the AWS Management Console with the 'EC2' service selected. In the main pane, a table lists 'Amazon Machine Images (AMIs) (1/1)'. A single row is selected for 'WebServerAMI'. At the top right of the table, there is a yellow button labeled 'Launch Instance from AMI'.

## 2. Create a Load Balancer

The screenshot shows the AWS Management Console with the 'Load Balancing' service selected. In the main pane, a table lists 'Target groups (0/1)'. A green success message at the top states 'Successfully created the target group LabGroup. Anomaly detection is automatically applied to all registered targets. Results can be viewed in the Targets tab.' Below the message, the 'LabGroup' details are shown, including the protocol (HTTP), port (80), and VPC settings. The 'Targets' tab is selected, showing a table with one row: 'No registered targets. You have not registered targets for this group yet.' A red button labeled 'Register targets' is visible at the bottom of the table.

Created target group

Screenshot of the AWS CloudFront console showing the creation of a new distribution named 'Lab'. The distribution is set to 'Active' and has a single origin endpoint 'http://172.0.0.1:443'. The 'Actions' dropdown shows options like 'Edit', 'Delete', 'Create Alias', 'Create Invalidation', and 'Create Cache Policy'.

## Created Load Balancer

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

Screenshot of the AWS EC2 console showing the creation of a new launch template named 'LabConfig'. The launch template details include a launch template ID 'lt-090025945711x20f', a launch template name 'LabConfig', and a default version '1'. The launch template version details show a version ID '1 (Default)', a description '...', a date created '2023-11-17T17:43:34.000Z', and an owner 'arn:aws:ec2:us-east-1:73055452349:root'. The 'Actions' dropdown shows options like 'Edit', 'Delete', 'Create New Version', and 'Delete Version'.

## Created launch template

The screenshot shows the AWS Auto Scaling Groups page. At the top, there are tabs for 'Auto Scaling groups (1/1)', 'Launch configurations', 'Launch templates', and 'Actions'. A prominent orange button says 'Create Auto Scaling group'. Below the tabs, there's a search bar and a table with columns: Name, Launch template/Configuration, Instances, Status, Desired capacity, Min, Max, Availability zones, and Creation time. One row is visible for 'Lab Auto Scaling Group'. On the right side, there's a sidebar titled 'Auto Scaling groups' with instructions on how to manage the table and a link to 'Learn more'.

## Created auto scaling group

The screenshot shows the AWS EC2 Instances page. The left sidebar has sections for 'EC2', 'Instances', 'Launch Templates', 'Images', 'Elastic Block Store', 'Network & Security', and 'Load Balancing'. The main area is titled 'Instances (4) info' and lists four instances: 'Lab Instance' (t2.micro), 'Radmin Host' (t2.micro), 'Web Server 1' (t2.micro), and another 'Lab Instance' (t2.micro). Each instance row includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, Public IPv4 IP, Elastic IP, and IPv6 IPs. The 'Launch instances' button is located at the bottom right of the table.

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

**Create VPC**

**Tags for all resources in the VPC:**

- Auto-generate
- lab

**IPv4 CIDR block:** [Info](#)  
Determine the starting IP and the size of your VPC using CIDR notation.  
10.0.0.0/16 [\(Edit\)](#) (IPv4)  
CIDR block size must be between /16 and /28.

**IPv6 CIDR block:** [Info](#)  
 No IPv6 CIDR block  
 Amazon-provided IPv6 CIDR block.

**Tenancy:** [Info](#)  
Default

**Number of Availability Zones (AZs):** [Info](#)  
Choose the number of AZs in which to provision subnets. We recommend at least three AZs for high availability.  
1 | 2 | 3 [\(Edit\)](#)

**Number of public subnets:** [Info](#)  
The number of public subnets to add to your VPC. Use public subnets for web applications that need to be publicly accessible over the internet.  
0 | 1 | 2 [\(Edit\)](#)

**Number of private subnets:** [Info](#)  
The number of private subnets to add to your VPC. Use private subnets to secure backend resources that don't need public access.  
0 | 1 | 2 [\(Edit\)](#)

**Customize subnets CIDR blocks:**  
Public subnet CIDR block in us-east-1a  
10.0.0.0/24 [\(Edit\)](#)

**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

© 2025, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Nov 18 14:25

**VPC dashboard**

**Details:** [Info](#)

<b>VPC ID:</b> vpc-04c508510705089f1	<b>State:</b> <a href="#">Available</a>	<b>Block Public Access:</b> <input type="checkbox"/> Off	<b>DNS hostnames:</b> Enabled
<b>DNS resolution:</b> Enabled	<b>Tenancy:</b> default	<b>DHCP option set:</b> dopt-05376328c9c3634c3	<b>Main route table:</b> rtb-0634428d77d5a697
<b>Main network ACL:</b> acl-0bf2145ea74055409	<b>Default VPC:</b> No	<b>IPv4 CIDR:</b> 10.0.0.0/16	<b>IPv6 pool:</b> -
<b>IPv6 CIDR (Network border group):</b> -	<b>Network Address Usage metrics:</b> Disabled	<b>Route 53 Resolver DNS Firewall rule groups:</b> -	<b>Owner ID:</b> 782040769740

**Resource map:** [Info](#)

**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  
lab-rtb-public  
lab-rtb-private1-us-east-1a  
lab-rtb-public1-us-east-1a

**Network Connections (2):** Connections to other networks  
lab-lgw  
lab-nat-public1-us-east-1a

© 2025, Amazon Web Services, Inc. or its affiliates. [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 and lab-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".

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR	IPv6 CIDR
lab-subnet-public1-us-east-1a	subnet-01f58aa5e4a29c42	Available	vpc-04c508510705089f1   lab...	Off	10.0.0.0/24	-
lab-subnet-public2	subnet-092fe7ebd859847f4	Available	vpc-04c508510705089f1   lab...	Off	10.0.2.0/24	-
-	subnet-05e107119d9338a33	Available	vpc-0c32ccb30bea76f5a	Off	172.31.64.0/20	-
lab-subnet-private2	subnet-03e0b679faaf7e4b	Available	vpc-04c508510705089f1   lab...	Off	10.0.3.0/24	-
-	subnet-0008097ed011a73c0	Available	vpc-0c32ccb30bea76f5a	Off	172.31.0.0/20	-
-	subnet-0105d46fb8a510cbf2	Available	vpc-0c32ccb30bea76f5a	Off	172.31.48.0/20	-
-	subnet-00282ffee2e7ea4f1	Available	vpc-0c32ccb30bea76f5a	Off	172.31.32.0/20	-
-	subnet-05e81627933aa4ed3b	Available	vpc-0c32ccb30bea76f5a	Off	172.31.80.0/20	-
lab-subnet-private1-us-east-1a	subnet-091916c14e4b7ae0	Available	vpc-04c508510705089f1   lab...	Off	10.0.1.0/24	-
Work Public Subnet	subnet-0754515abb4820aa0a	Available	vpc-04581655fb07986   WOR...	Off	10.0.0.0/24	-
-	subnet-0a1fc1a97b79543a	Available	vpc-0c32ccb30bea76f5a	Off	172.31.16.0/20	-

## 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

You have successfully updated subnet associations for rtb-06c6423ccb610a88b4 / lab-rtb-private1-us-east-1a.

**Route tables (1/6) Info**

Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC	Owner ID
rtb-022f8fd96ab3f5750	-	-	-	Yes	vpc-0-32ccb610a88b4	702840769740
rtb-0f17ba4413f7f5	-	-	-	Yes	vpc-0-43386b55fbef7986	702840769740
Work Public Route Table	rtb-0d21c1ba96a6500f8	subnet-05945d5eb8420...	-	No	vpc-0-43386b55fbef7986   Work...	702840769740
rtb-0baa72bd77d5e657	-	-	-	Yes	vpc-0-4c508510705089f1	702840769740
<b>lab-rtb-public</b>	<b>rtb-0ea5bdcc10b3cbb21</b>	<b>subnet-01ff5baac5e6a29c...</b>	-	No	vpc-0-4c508510705089f1   lab...	<b>702840769740</b>
lab-rtb-private1-us-east-1a	rtb-06c6423ccb610a88b4	2 subnets	-	No	vpc-0-4c508510705089f1   lab...	702840769740

**rtb-0ea5bdcc10b3cbb21 / lab-rtb-public**

Details Routes Subnet associations Edge associations Route propagation Tags

**Routes (2)**

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	ipw-0a20952d7f75624ce	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Nov 18 14:28

You have successfully updated subnet associations for rtb-06c6423ccb610a88b4 / lab-rtb-private1-us-east-1a.

**Route tables (1/6) Info**

Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC	Owner ID
rtb-022f8fd96ab3f5750	-	-	-	Yes	vpc-0-32ccb610a88b4	702840769740
rtb-0f17ba4413f7f5	-	-	-	Yes	vpc-0-43386b55fbef7986	702840769740
Work Public Route Table	rtb-0d21c1ba96a6500f8	subnet-05945d5eb8420...	-	No	vpc-0-43386b55fbef7986   Work...	702840769740
rtb-0baa72bd77d5e657	-	-	-	Yes	vpc-0-4c508510705089f1	702840769740
<b>lab-rtb-public</b>	<b>rtb-0ea5bdcc10b3cbb21</b>	<b>subnet-01ff5baac5e6a29c...</b>	-	No	vpc-0-4c508510705089f1   lab...	<b>702840769740</b>
<b>lab-rtb-private1-us-east-1a</b>	<b>rtb-06c6423ccb610a88b4</b>	<b>subnet-0918916e14e4b7...</b>	-	No	vpc-0-4c508510705089f1   lab...	<b>702840769740</b>

**rtb-06c6423ccb610a88b4 / lab-rtb-private1-us-east-1a**

Details Routes Subnet associations Edge associations Route propagation Tags

**Routes (2)**

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	ipw-0c1f72fe24a90a660	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table

© 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

The screenshot shows the AWS EC2 Volumes page. A green success message at the top center reads "Successfully created volume vol-089daec5c5ce48031." Below this, the "Volumes" table lists three volumes:

Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Source volume ID	Created
My Volume	vol-089daec5c5ce48031	gp2	1 GiB	100	-	-	-	2025/11/20
	vol-09bf1205113368745	gp3	9 GiB	3000	125	snap-0881c81...	-	2025/11/20
	vol-0ede54c865b16ff94	gp3	8 GiB	3000	125	snap-0881c81...	-	2025/11/20

Below the table, a section titled "Fault tolerance for all volumes in this Region" shows "0 / 2" volumes.

## Task 2: Attach the Volume to an Instance

The screenshot shows the "Attach volume" dialog for the volume "vol-089daec5c5ce48031".

**Basic details**

Volume ID: vol-089daec5c5ce48031 (My Volume)

Availability Zone: us-east-2a (us-east-1a)

Instance: i-0c9e45266c84198ce (Lab) (running)

Only instances in the same Availability Zone as the selected volume are displayed.

Device name: /dev/sdf

Recommended device names for Linux: /dev/xvda for root volume, /dev/sd[f-p] for data volumes.

Info: Newer Linux kernels may rename your devices to /dev/xvdf through /dev/xvdz internally, even when the device name entered here (and shown in the details) is /dev/sdf through /dev/sdp.

Buttons: Cancel and Attach volume

## Task 3: Connect to Your Amazon EC2 Instance

The screenshot shows the AWS EC2 Instance Connect interface. At the top, the instance ID is listed as i-0c9e45266c84198ce (Lab). The 'Public IPv4 address' is 34.227.13.110. The 'Username' field contains 'ec2-user'. A note at the bottom states: 'Note: In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.' Below the form are 'Cancel' and 'Connect' buttons.

## Task 4: Create and Configure Your File System

```
Amazon Linux 2023
https://aws.amazon.com/linux/amazon-linux-2023

[ec2-user@ip-10-1-11-236 ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        4.0M   0M  4.0M  0% /dev
tmpfs          475M   0M  475M  0% /dev/shm
tmpfs          190M  452K 190M  1% /run
/dev/xvda1     8.0G  1.6G  6.4G  21% /
tmpfs          475M   0M  475M  0% /tmp
/dev/xvda128    10M  1.3M  8.7M  13% /boot/efi
tmpfs          95M   0M  95M  0% /run/user/1000
[ec2-user@ip-10-1-11-236 ~]$ sudo mkfs -t ext3 /dev/sdf
mke2fs 1.46.5 (30-Dec-2021)
Creating filesystem with 262144 4k blocks and 65536 inodes
filesystem UUID: b719f563-4cda-43da-a549-3d6faea2a6a2
Superblock backups stored on blocks:
            32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

[ec2-user@ip-10-1-11-236 ~]$ sudo mkdir /mnt/data-store
```

## Task 5: Create an Amazon EBS Snapshot

The screenshot shows two consecutive pages from the AWS EC2 Volumes interface.

**Page 1: Snapshot details**

- Description:** A text input field with placeholder "Add a description for your snapshot". Below it is a note: "255 characters maximum."
- Encryption:** A dropdown menu showing "Not encrypted".
- Tags:** A section titled "Tags Info" with a note: "A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs." It includes a table for adding tags:

Key	Value - optional
<input type="text" value="Name"/>	<input type="text" value="My Snapshot"/>

A note below says "You can add 49 more tags." and there are "Add tag" and "Cancel" buttons.
- Create snapshot** button (orange)

**Page 2: Snapshots (1) Info**

- Snapshots (1) Info:** A table listing the created snapshot:

Name	Snapshot ID	Full snapshot size	Volume size	Description	Storage tier	Snapshot status
My Snapshot	snap-0c1671a8c83d1d4a0	-	1 GiB	-	Standard	Pending
- Actions:** Buttons for "Recycle Bin" and "Create snapshot".
- Left sidebar:** Includes sections for Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), Network & Security (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces).
- Message:** "Select a snapshot above."

The screenshot shows the AWS Cloud9 interface. On the left, there's a sidebar with navigation links: Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, and Network Interfaces. The 'Snapshots' link under 'Elastic Block Store' is highlighted.

The main content area displays a table titled 'Snapshots (1/1)'. It shows one entry: 'My Snapshot' with Snapshot ID 'snap-0c1671a8c83d1d4a0', Full snapshot size '53 MiB', Volume size '1 GiB', and a status of 'Completed'. A context menu is open over this entry, with 'Create volume from snapshot' highlighted.

At the bottom of the page, there's a footer with links: © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

## Create a Volume Using Your Snapshot

The screenshot shows the AWS Cloud9 interface. The sidebar and main content area are identical to the previous screenshot, but there's a green success message at the top: 'Successfully created volume vol-0f4415c04689e9d79.'

The main content area displays a table titled 'Snapshots (1)'. It shows one entry: 'My Snapshot' with Snapshot ID 'snap-0c1671a8c83d1d4a0', Full snapshot size '53 MiB', Volume size '1 GiB', Storage tier 'Standard', and a status of 'Completed'. Below the table, a message says 'Select a snapshot above.'

At the bottom of the page, there's a footer with links: © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, and Cookie preferences.

## Task 6: Restore the Amazon EBS Snapshot

### Create a Volume Using Your Snapshot

The screenshot shows the AWS Management Console with the Volumes page open. On the left, there's a navigation sidebar with links like Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, Elastic Block Store, and Network & Security. The main area displays a table of volumes, including 'My Volume' and 'Restored Vol...', which is selected. A context menu is open over the 'Restored Vol...' row, with 'Attach volume' being the selected option. The table columns include Name, Volume ID, Type, Size, IOPS, and Throughput.

### Attach the Restored Volume to Your EC2 Instance

The screenshot shows the 'Attach volume' step in the AWS Management Console. It starts with a 'Basic details' section where you can select a volume (vol-0f4415c04689e9d79), an availability zone (use1-az2), and an instance (i-0c9e45266c84198ce). Below that is a 'Device name' field set to '/dev/sdg'. A note at the bottom states: 'Newer Linux kernels may rename your devices to /dev/xvdf through /dev/xvdp internally, even when the device name entered here (and shown in the details) is /dev/sdf through /dev/sdp.' At the bottom right are 'Cancel' and 'Attach volume' buttons.

## Mount the Restored Volume

The screenshot shows a terminal session in an AWS CloudShell window. The user has mounted a restored volume at /mnt/data-store and is demonstrating basic file operations on it.

```
Superblock backups stored on blocks:
 32768, 98304, 163840, 229376

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

[ec2-user@ip-10-1-11-236 ~]$ sudo mkdir /mnt/data-store
[ec2-user@ip-10-1-11-236 ~]$ sudo mount /dev/sdf /mnt/data-store
[ec2-user@ip-10-1-11-236 ~]$ echo "/dev/sdf /mnt/data-store ext3 defaults,noatime 1 2" | sudo tee -a /etc/fstab
/dev/sdf /mnt/data-store ext3 defaults,noatime 1 2
[ec2-user@ip-10-1-11-236 ~]$ cat /etc/fstab
#
UUID=9a57ac2f-bfe7-4e29-bf89-56caddc22c97      /          xfs  defaults,noatime 1  1
UUID=9682-52BE        /boot/efi    vfat  defaults,noatime,uid=0,gid=0,umask=0077,shortname=winnt,x-systemd.automount 0 2
/dev/sdf  /mnt/data-store ext3 defaults,noatime 1 2
[ec2-user@ip-10-1-11-236 ~]$ df -h
Filesystem      Size   Used  Avail Use% Mounted on
/devtmpfs       4.0M   0  4.0M  0% /dev
tmpfs          475M   0  475M  0% /dev/shm
tmpfs          190M  448K  190M  1% /run
/dev/xvda1     8.0G  1.6G  6.4G  21% /
tmpfs          475M   0  475M  0% /tmp
/dev/xvda128   10M   1.3M  8.7M  13% /boot/efi
tmpfs          95M   0  95M  0% /run/user/1000
/dev/xvdf      975M  60K  924M  1% /mnt/data-store
[ec2-user@ip-10-1-11-236 ~]$ sudo sh -c "echo some text has been written > /mnt/data-store/file.txt"
[ec2-user@ip-10-1-11-236 ~]$ cat /mnt/data-store/file.txt
some text has been written
[ec2-user@ip-10-1-11-236 ~]$ sudo rm /mnt/data-store/file.txt
[ec2-user@ip-10-1-11-236 ~]$ ls /mnt/data-store/

```

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates

The screenshot shows a terminal session in an AWS CloudShell window. The user has mounted a restored volume at /mnt/data-store and created a new directory /mnt/data-store2, then mounted another volume there.

```
Writing superblocks and filesystem accounting information: done

[ec2-user@ip-10-1-11-236 ~]$ sudo mkdir /mnt/data-store
[ec2-user@ip-10-1-11-236 ~]$ sudo mount /dev/sdf /mnt/data-store
[ec2-user@ip-10-1-11-236 ~]$ echo "/dev/sdf /mnt/data-store ext3 defaults,noatime 1 2" | sudo tee -a /etc/fstab
/dev/sdf /mnt/data-store ext3 defaults,noatime 1 2
[ec2-user@ip-10-1-11-236 ~]$ cat /etc/fstab
#
UUID=9a57ac2f-bfe7-4e29-bf89-56caddc22c97      /          xfs  defaults,noatime 1  1
UUID=9682-52BE        /boot/efi    vfat  defaults,noatime,uid=0,gid=0,umask=0077,shortname=winnt,x-systemd.automount 0 2
/dev/sdf  /mnt/data-store ext3 defaults,noatime 1 2
[ec2-user@ip-10-1-11-236 ~]$ df -h
Filesystem      Size   Used  Avail Use% Mounted on
/devtmpfs       4.0M   0  4.0M  0% /dev
tmpfs          475M   0  475M  0% /dev/shm
tmpfs          190M  448K  190M  1% /run
/dev/xvda1     8.0G  1.6G  6.4G  21% /
tmpfs          475M   0  475M  0% /tmp
/dev/xvda128   10M   1.3M  8.7M  13% /boot/efi
tmpfs          95M   0  95M  0% /run/user/1000
/dev/xvdf      975M  60K  924M  1% /mnt/data-store
[ec2-user@ip-10-1-11-236 ~]$ sudo sh -c "echo some text has been written > /mnt/data-store/file.txt"
[ec2-user@ip-10-1-11-236 ~]$ cat /mnt/data-store/file.txt
some text has been written
[ec2-user@ip-10-1-11-236 ~]$ sudo rm /mnt/data-store/file.txt
[ec2-user@ip-10-1-11-236 ~]$ ls /mnt/data-store/
lost+found
[ec2-user@ip-10-1-11-236 ~]$ sudo mkdir /mnt/data-store2
[ec2-user@ip-10-1-11-236 ~]$ sudo mount /dev/sdg /mnt/data-store2
[ec2-user@ip-10-1-11-236 ~]$ ls /mnt/data-store2/
file.txt  lost+found
[ec2-user@ip-10-1-11-236 ~]$ ss
```

© 2025, Amazon Web Services, Inc. or its affiliates

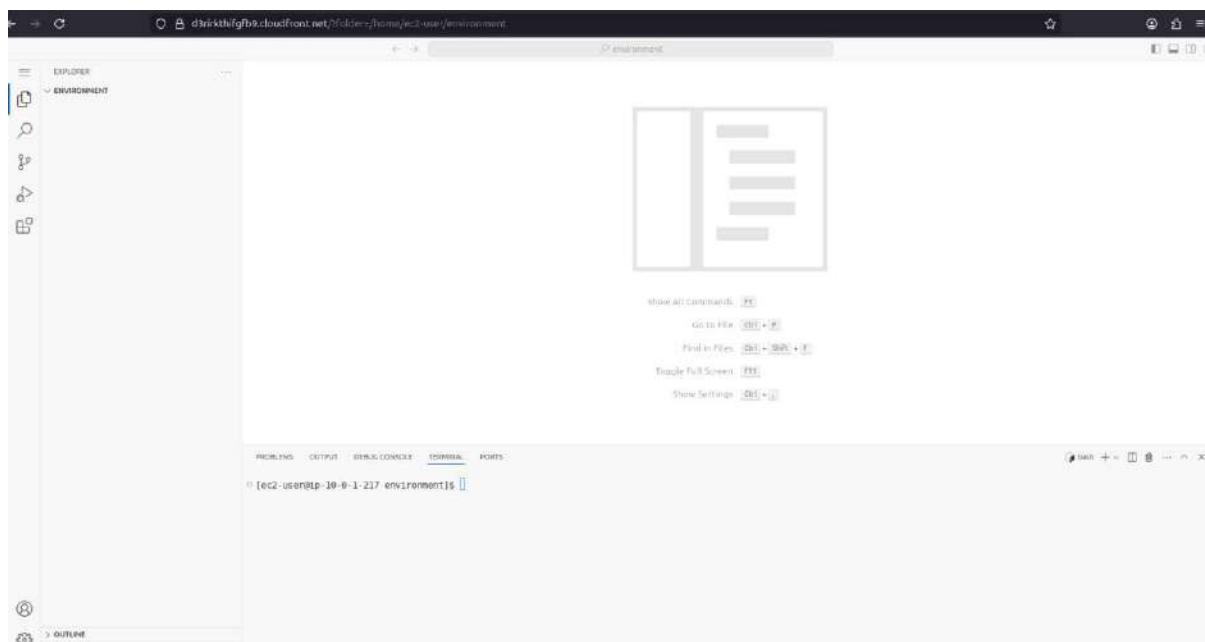
# Breaking a Monolithic Node.js Application into Microservices

## 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:

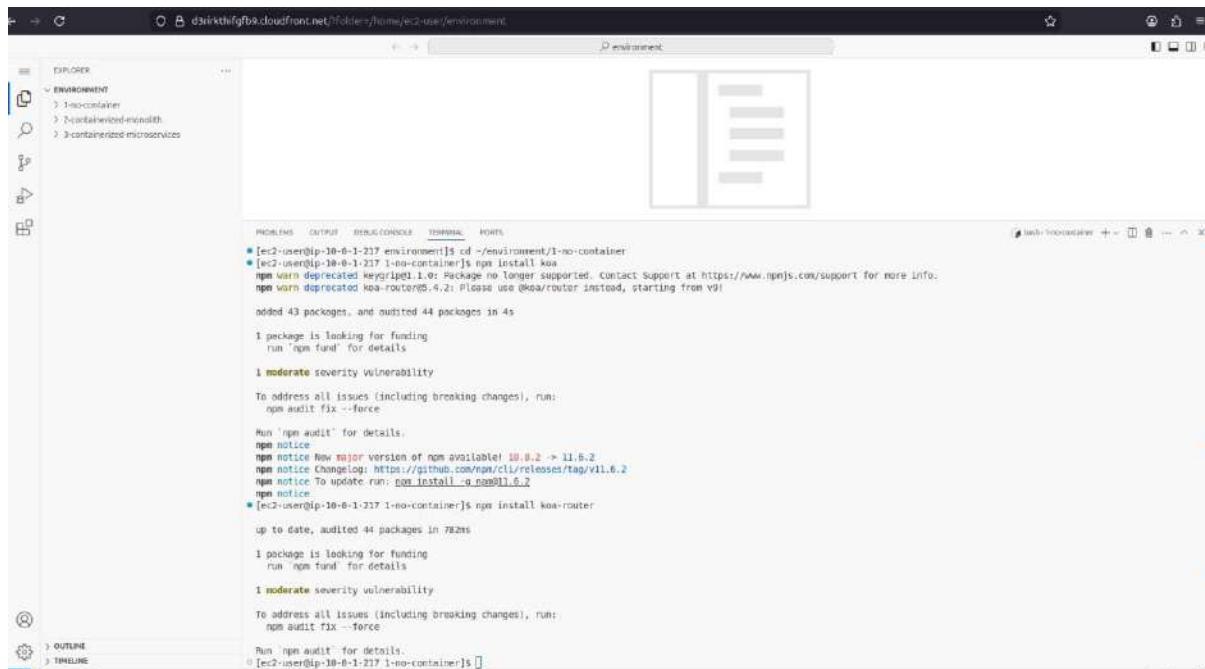
```
curl -s https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACACAD-3-113230/19-lab-mod14-guided-ECS/s3/lab-files-ms-node-js.tar.gz | tar -zv
```

## 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`



The screenshot shows the Microsoft Visual Studio Code interface. The terminal tab is active, displaying the following command-line session:

```
[ec2-user@ip-10-0-1-237 environment]$ cd ~/environment/1-no-container
[ec2-user@ip-10-0-1-237 1-no-container]$ npm install koa
npm warn deprecated keyv@1.0: Package no longer supported. Contact Support at https://www.npmjs.com/support for more info.
npm warn deprecated koa-router@0.4.2: Please use @koa/router instead, starting from v9!
added 43 packages, and audited 44 packages in 4s
1 package is looking for funding
  run `npm fund` for details
  1 moderate severity vulnerability

  To address all issues (including breaking changes), run:
    npm audit fix -f --force

  Run 'npm audit' for details.
npm notice
npm notice New major version of npm available! 10.0.2 => 11.6.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.6.2
npm notice To update run: npm install -g npm@11.6.2
npm notice
[ec2-user@ip-10-0-1-237 1-no-container]$ npm install koa-router
up to date, audited 44 packages in 78ms
1 package is looking for funding
  run `npm fund` for details
  1 moderate severity vulnerability

  To address all issues (including breaking changes), run:
    npm audit fix -f --force

  Run 'npm audit' for details.
[ec2-user@ip-10-0-1-237 1-no-container]$
```

### 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`

```

d202z-imcncl.cloudfront.net:~/code/home/ec2-user/environment
PROBLEMS OUTPUT TERMINAL PORTS ⚡
db.json index.js package.json server.js
● [ec2-user@ip-10-0-1-204 ~] $ npm install koa
npm install koa-router
npm warn deprecated keygrip@0.1.0: Package no longer supported. Contact Support at https://www.npmjs.com/support for more info.
npm warn deprecated koa-router@0.4.1: Please use @koa/router instead, starting from v9!
added 43 packages, and audited 44 packages in 4s

1 package is looking for funding
  run 'npm fund' for details

1 moderate severity vulnerability

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.
npm notice New major version of npm available! 10.0.2 -> 11.6.2
npm notice Check out: https://github.com/npm/cli/releases/tag/v11.6.2
npm notice To update run: npm install -g npm@11.6.2
npm notice

up to date, audited 44 packages in 821ms

1 package is looking for funding
  run 'npm fund' for details

1 moderate severity vulnerability

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.
● [ec2-user@ip-10-0-1-204 ~] $ npm start
> start
> node index.js
Leader: 32953 is running
Worker: 32971 started
Worker: 32970 started

```

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
```



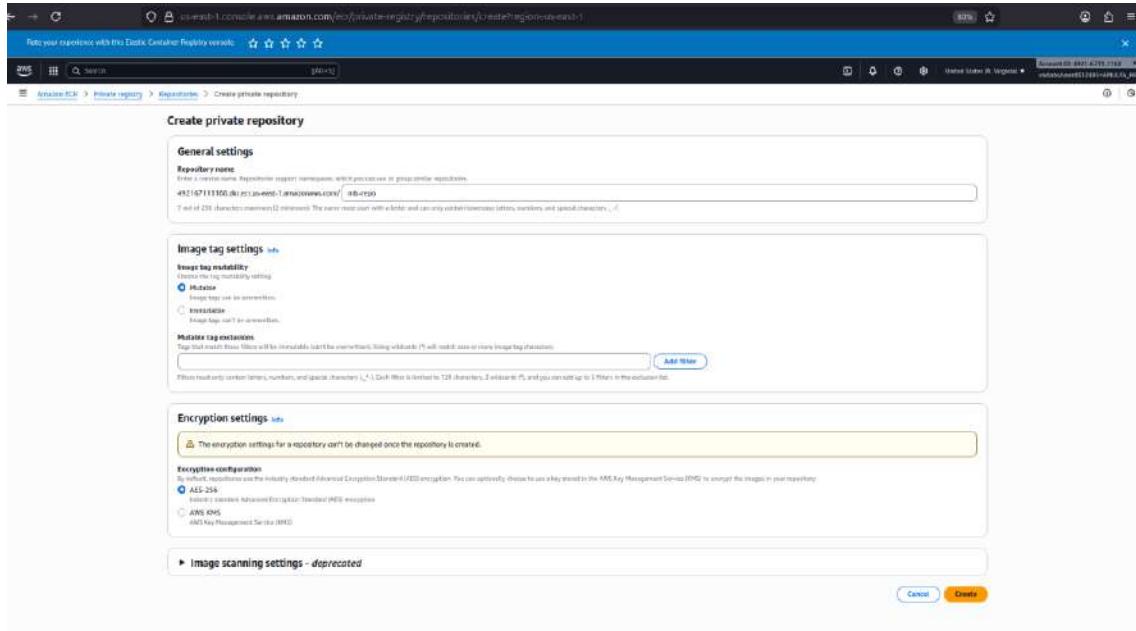
7. Now you stop the Node.js server.

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

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



- 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)
```

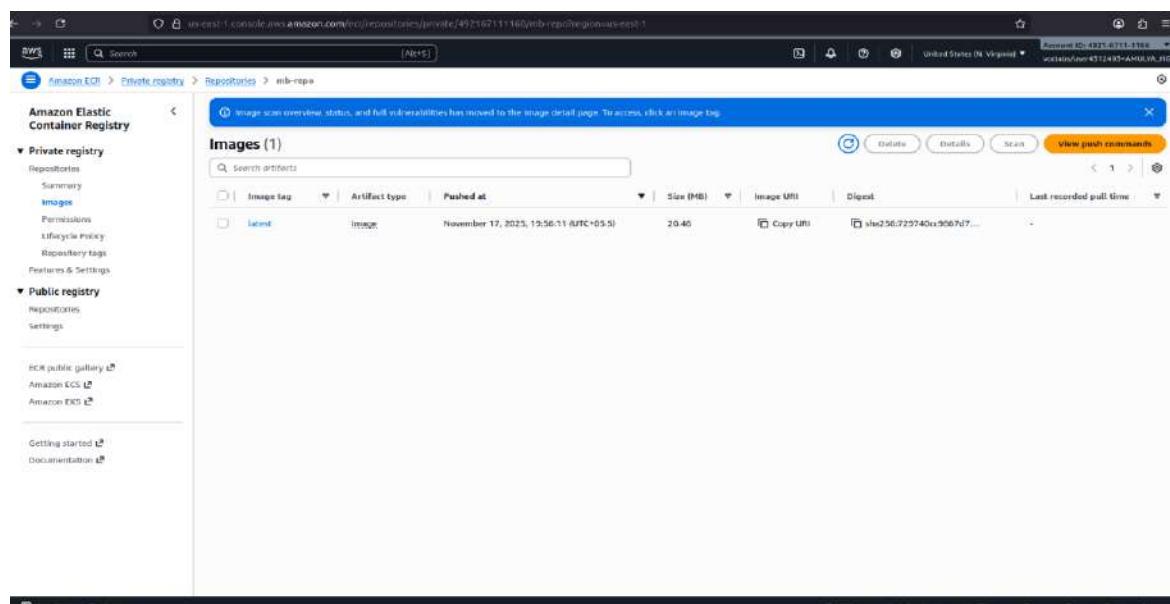
5. In the terminal tab, to change the directory to the 2-containerized-monolith folder, enter the following command:

```
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 NNNNNNNNNNNN.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

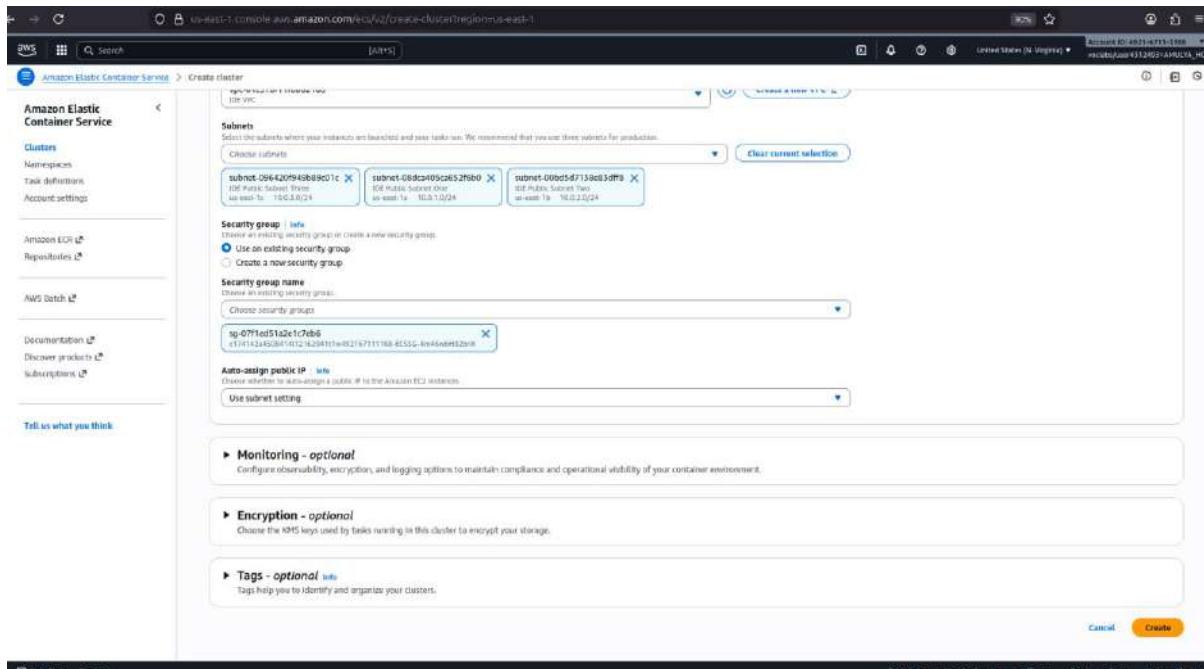
### Step 1: Creating an Amazon ECS cluster

The screenshot shows the 'Create cluster' wizard in the AWS Elastic Container Service console. The 'Cluster configuration' step is active. A note at the top says 'Now! ECS Managed Instances' and 'Get started with ECS Managed Instances with flexible hardware capabilities and reduced operational overhead. Configure in the infrastructure section below. Your cluster is automatically configured for AWS Fargate (serverless) with Fargate and Fargate Spot. Learn more.' Below this, the 'Cluster name' field contains 'mb-ecs-cluster'. Under 'Service Connect defaults - optional', there's a note: 'Your cluster is automatically configured for AWS Fargate (serverless), but you may choose to add Amazon EC2 instances (server).'. The 'Infrastructure - advanced' section has a note: 'Configure the manner of obtaining compute resources that will be used to host your application. Your cluster is automatically configured for AWS Fargate (serverless), but you may choose to add Amazon EC2 instances (server).'. It lists three options: 'Fargate only' (selected), 'Fargate and Managed Instances' (disabled), and 'Fargate and Self-managed instances' (disabled).

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.

The screenshot shows the 'Create cluster' wizard in the AWS Elastic Container Service console. The 'Container instance Amazon Machine Image (AMI)' step is active. It shows the selected AMI as 'Amazon Linux 2023'. The 'EC2 instance type' dropdown shows 't2.medium (8GiB, 1vCPU - 4 GiB Memory)'. The 'Desired capacity' section shows 'Minimum' set to '2' and 'Maximum' set to '6'. The 'Root EBS volume size' dropdown shows '30'. The 'Network settings' section shows the VPC 'vpc-01c57bf1f5b60216d' and Subnets. Other sections visible include 'Container instance Amazon Machine Image (AMI)', 'EC2 instance type', 'EC2 instance role', 'Container instance type', 'Desired capacity', 'Root EBS volume size', 'Network settings', and 'Subnets'.

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.

The screenshot shows the AWS ECS console interface. On the left, there's a navigation sidebar for 'Amazon Elastic Container Service' with options like 'Clusters', 'Namespaces', 'Task definitions', and 'Documentation'. The main content area is titled 'mb-ecs-cluster' (ASG). It has tabs for 'Services', 'Tasks', 'Infrastructure' (which is active), 'Updates', 'Metrics', 'Scheduled tasks', 'Configuration', 'Event history', and 'Tags'. Under 'Infrastructure', there are sections for 'Capacity providers' (listing Fargate, Fargate Spot, and Infra-EC2 Auto Scaling) and 'Container instances' (listing two instances: 'in-ext-1c' and 'in-ext-1a').

## 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

The screenshot shows the 'Create new task definition' wizard. Step 1: 'Task definition configuration' shows 'Task definition family' set to 'mb-task'. Step 2: 'Infrastructure requirements' shows 'Launch type' set to 'AWS Fargate' (unchecked), 'Managed instances - new' (unchecked), and 'Amazon EC2 instances' (checked). Step 3: 'Task size' shows 'CPU' set to '.5vCPU' and 'Memory' set to '1 GB'. Step 4: 'Task roles - conditional' shows 'Task role' set to 'mb-task-role'. Step 5: 'Task execution role' shows 'Task execution role' set to 'mb-task-execution-role'.

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.

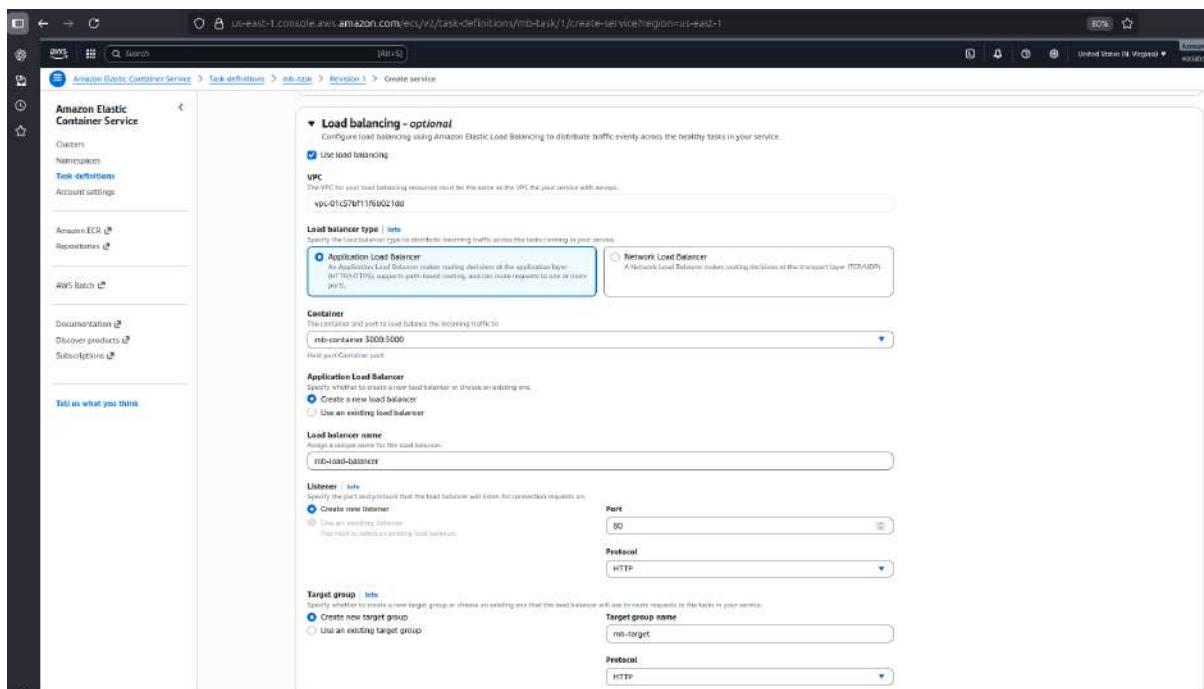
Tell us what you think:

### Step 3: Creating and deploying the service

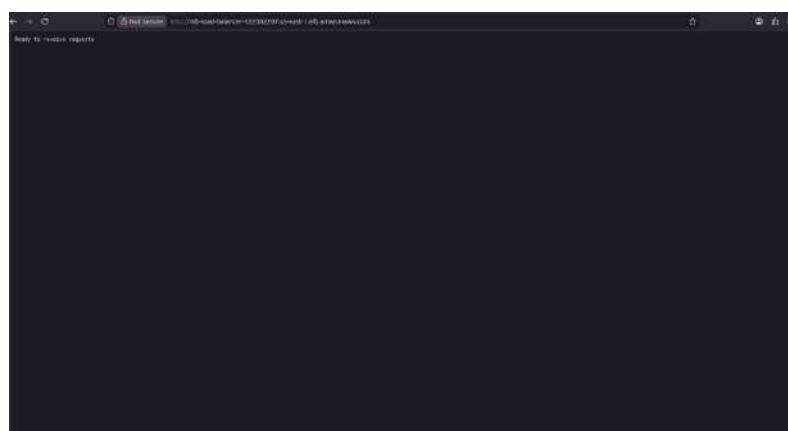
Tell us what you think:

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



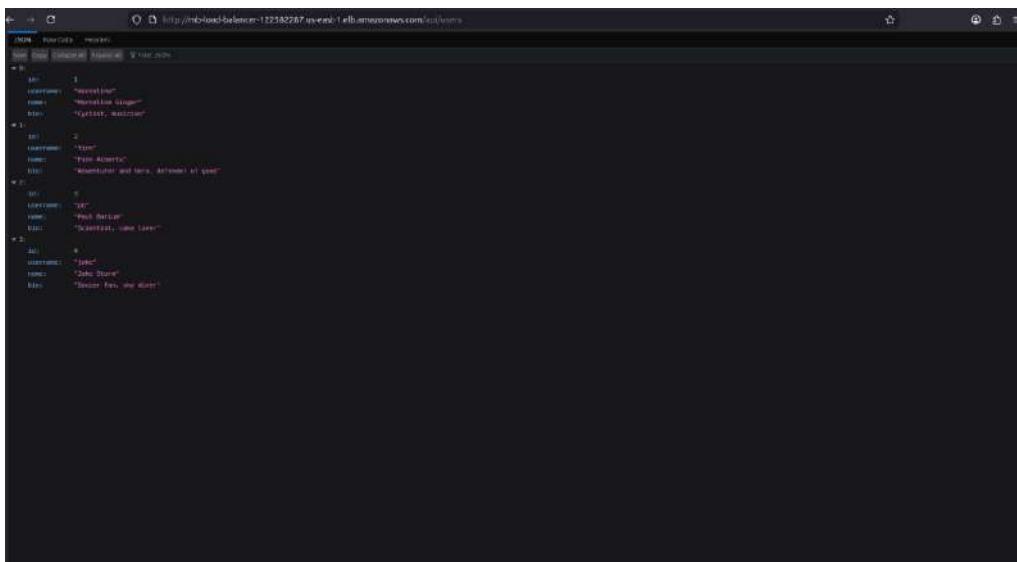
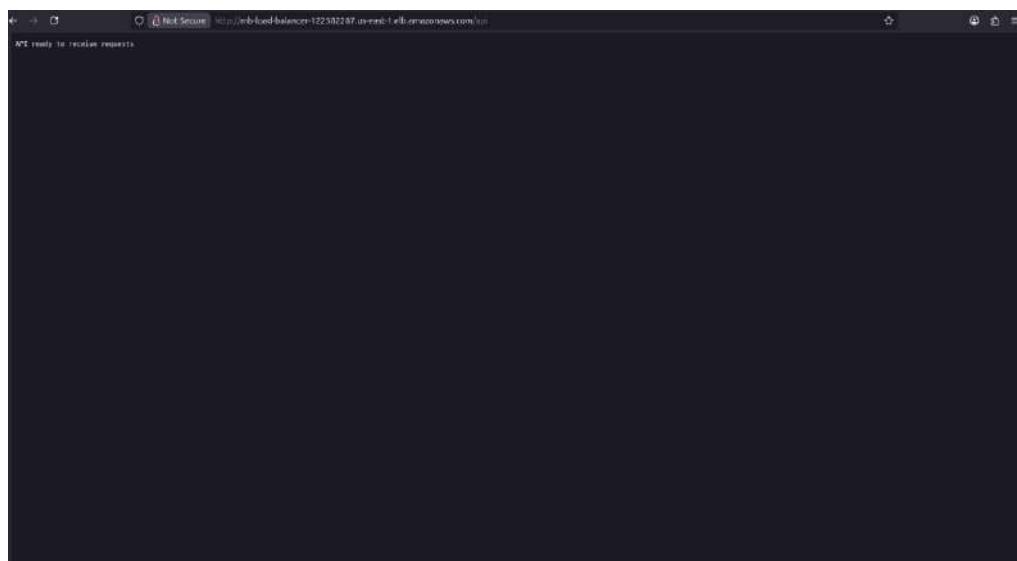
12. From the services list, choose mb-ecs-service
13. It should show a Status of *Active* with one Task that's Running
14. In the Load balancer health section choose load balancer.
15. Copy the DNS name for the load balancer, and paste it into a new browser tab



**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

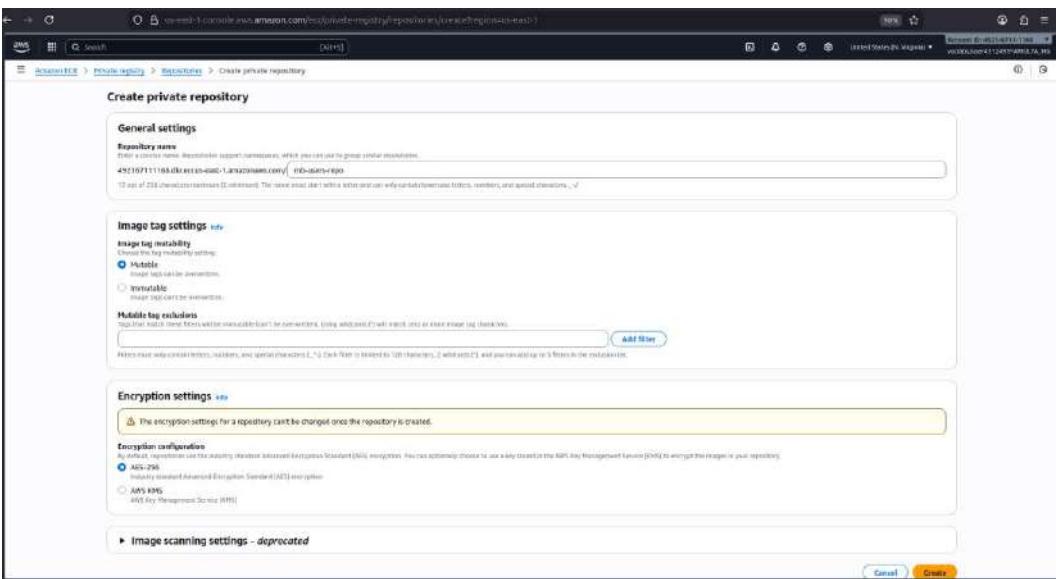
- *DNS name/api*
- *DNS name/api/users*
- *DNS name/api/threads*
- *DNS name/api/posts/in-thread/2*



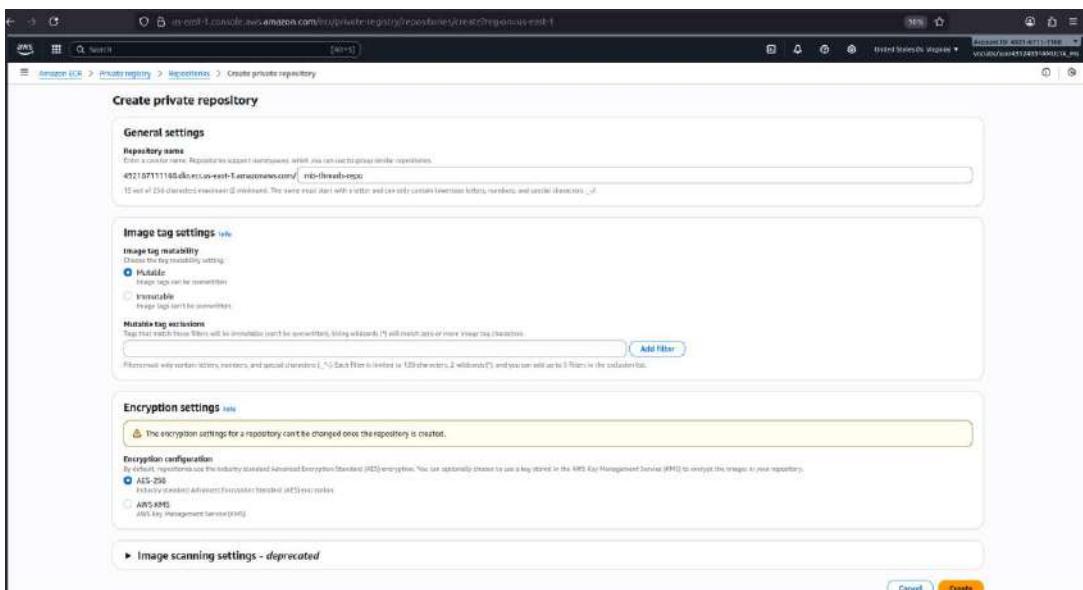
## Task 5: Refactoring the monolith

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

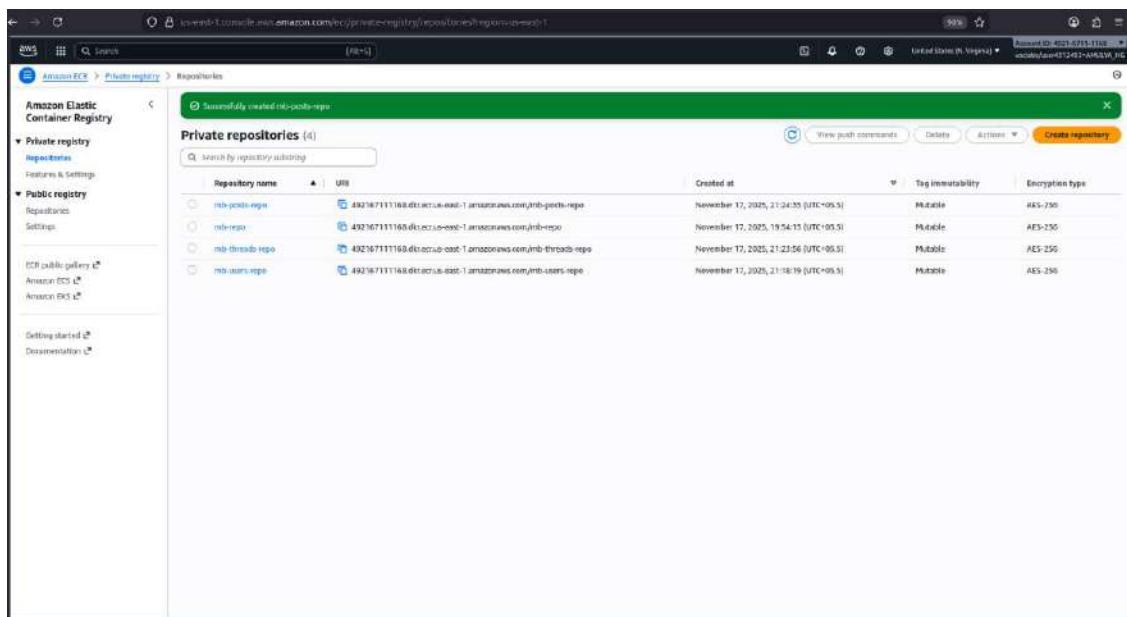
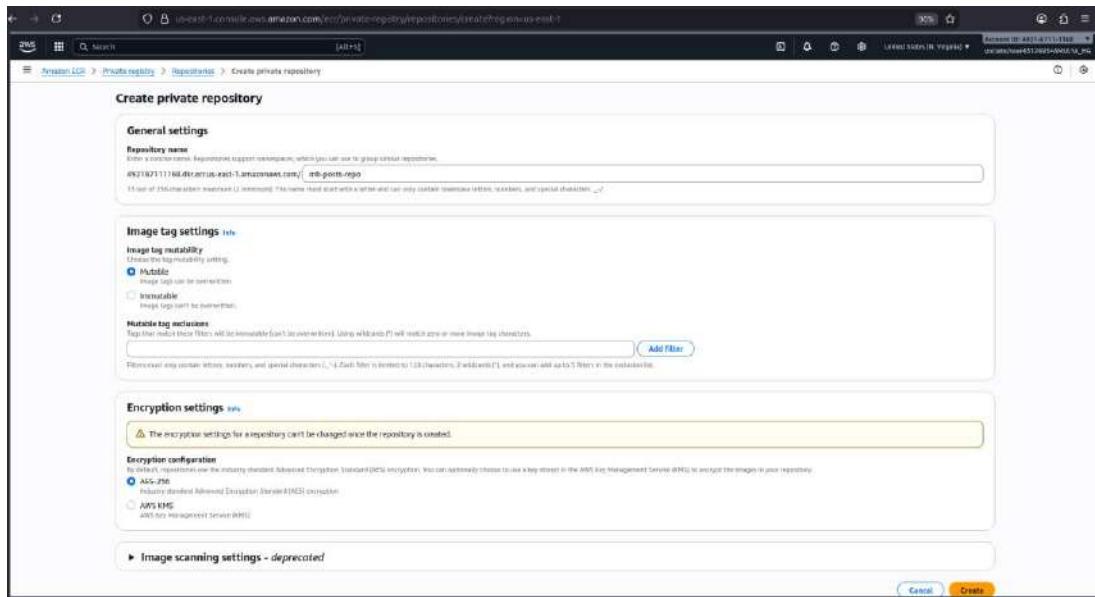
1. On the AWS Management Console, in the search box, enter and choose Elastic Container Registry
  2. Choose Repositories, then choose Create repository.
  3. For Repository name, enter mb-users-repo
  4. Choose Create



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](https://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](https://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

```

ENVIRONMENT
  1 container
  2 containerized-microservices
  3-containerized-microservices
    > posts
    > threads
    > users

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

[ec2-user@ip-10-0-1-204 ~]$ cd threads
[ec2-user@ip-10-0-1-204 threads]$ docker build -t mb-threads-repo .
[+] Building 4.4s FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] transfering dockerfile: 199B
=> [internal] load metadata for docker.io/mhart/alpine-node:7.10.1-1
=> [internal] load .dockerignore
=> [internal] transfering context: 2B
=> [1/4] FROM docker.io/mhart/alpine-node:7.10.10sha256:d334920c4660440676ce9d1e6162ab544349e4a4359c517300391c877cffbdc
=> [internal] load build context
=> [internal] transfering context: 1.73kB
=> [internal] transfering manifest: 1.73kB
=> [3/4] ADD .
=> [4/4] RUN npm install
=> exporting to image
=> exporting layers
=> exporting manifest
=> exporting metadata
=> pushing to docker.io/library/mb-threads-repo
[ec2-user@ip-10-0-1-204 threads]$ docker tag mb-threads-repo:latest 492167111168.dkr.ecr.us-east-1.amazonaws.com/mb-threads-repo:latest
[ec2-user@ip-10-0-1-204 threads]$ docker push 492167111168.dkr.ecr.us-east-1.amazonaws.com/mb-threads-repo:latest
The push refers to repository [492167111168.dkr.ecr.us-east-1.amazonaws.com/mb-threads-repo]
28ad031935d6: Pushed
3f79b718a866: Pushed
308835345264: Pushed
809fd7841192: Pushed
latest: digest: sha256:sc9e2fbdddf659ad9cc01ff0b99e999cb78cc2db0483a9fee5ddde1a7cdad size: 1364
[ec2-user@ip-10-0-1-204 threads]$ 

```

- 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

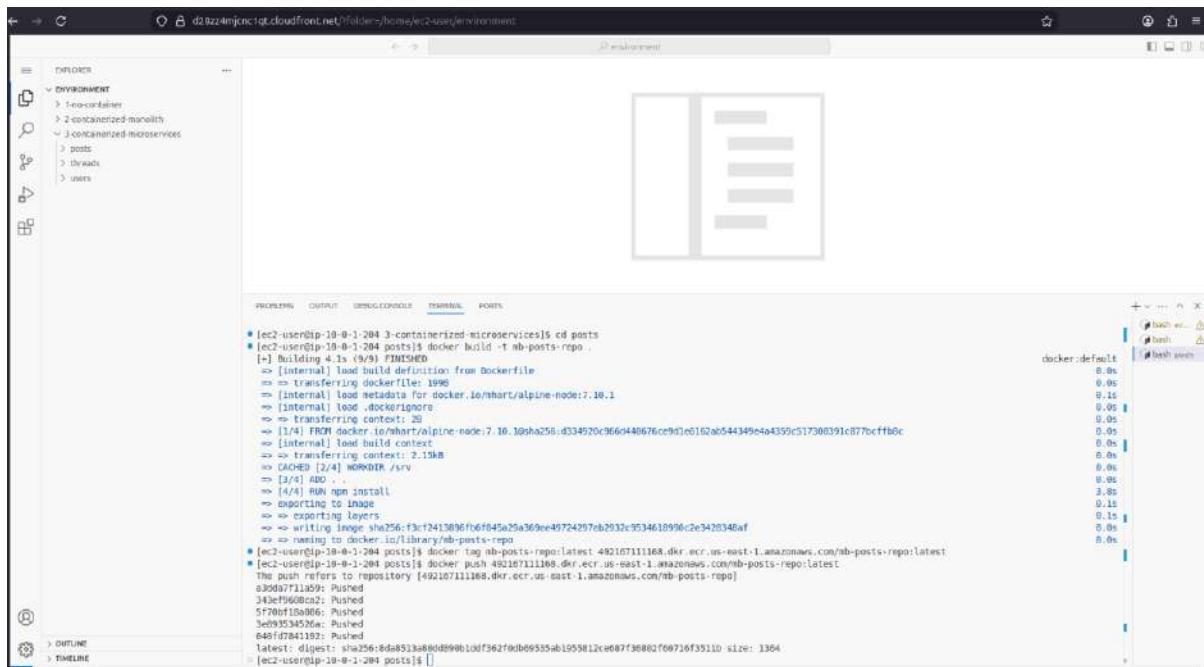
### Step 3: Building and pushing the image for the posts microservice

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



```

d2bz24mjrc1gt.cloudfront.net?file=/home/ec2-user/environment

* [ec2-user@ip-19-0-1-284 ~] cd posts
* [ec2-user@ip-19-0-1-284 posts]$ docker build -t mb-posts-repo .
[+] Building 4.1s (9/9) FINISHED
   => [internal] load build definition from Dockerfile
   => [internal] load metadata for docker.io/mhart/alpine-node:7.10.1
   [internal] load .dockerignore
   [internal] load build context
   => [7/7] FROM docker.io/mhart/alpine-node:7.10.1@sha256:d334920c9660440676ce9f1e0162ab544349e4a4359c517308391c877bcff80c
   [internal] load context
   => [internal] load build context
   => [internal] transfer context: 2.19kB
   => [internal] ADD .
   => [3/4] RUN npm install
   => [4/4] RUN npm start
   => [internal] export build context
   => [internal] exporting layers
   => [internal] writing image sha256:f3cf2413096fb67845a29a369ee48724297cb2932e9534610999c2e342034af
   => [internal] naming to docker.io/library(mb-posts-repo)
* [ec2-user@ip-19-0-1-284 posts]$ docker tag mb-posts-repo:latest 492167111168.dkr.ecr.us-east-1.amazonaws.com/mb-posts-repo:latest
* [ec2-user@ip-19-0-1-284 posts]$ docker push 492167111168.dkr.ecr.us-east-1.amazonaws.com/mb-posts-repo:latest
492167111168.dkr.ecr.us-east-1.amazonaws.com/mb-posts-repo:latest
492167111168.dkr.ecr.us-east-1.amazonaws.com/mb-posts-repo:latest
492167111168.dkr.ecr.us-east-1.amazonaws.com/mb-posts-repo:latest
492167111168.dkr.ecr.us-east-1.amazonaws.com/mb-posts-repo:latest
57f70ff10a806: Pushed
57f70ff10a806: Pushed
3e9b3534526a: Pushed
648fd7d841192: Pushed
latest: digest: sha256:8da8513aa80dd899b10df362f0db69535ab1955812c687f38881f00716f35110 size: 1364
* [ec2-user@ip-19-0-1-284 posts]$

```

7. Switch to Elastic Container Registry select mb-posts-repo

8. In the image list you can see the container image that you pushed

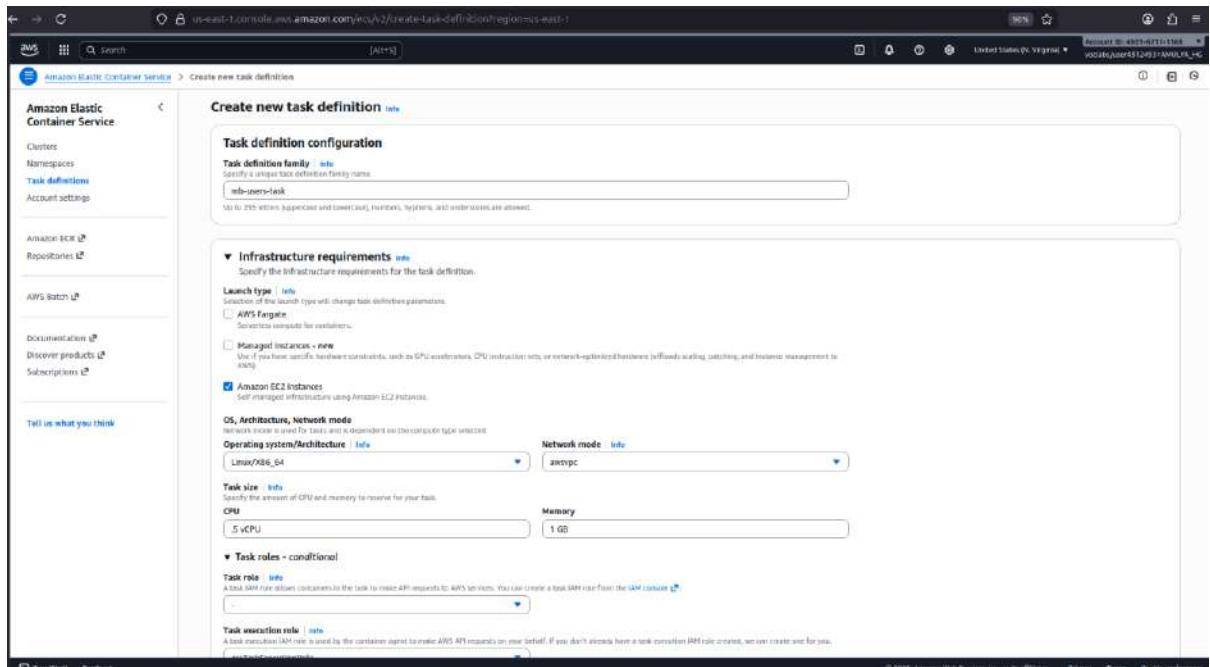
9. In the image list, choose copy URI for the image URI

## 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