

# **Assignment-1**

**Problem Statement:** Install and Configure the AWS CLI on a Red Hat EC2 Instance.

## **Procedure:**

- 1) **Connect to the Red Hat EC2 instance by using SSH**
  - a. Download the labsuser.pem file from the Details panel.
  - b. Open a terminal, set the file permissions, and connect to the instance using the ssh command.
- 2) **Install the AWS CLI on Red Hat Linux**
  - a) **Ensure Python is installed by running:**  
`python3 --version`
  - b) **Install the unzip utility:**  
`sudo yum install -y unzip`
  - c) **Download and install the AWS CLI:**  
`curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`  
`unzip awscliv2.zip`  
`sudo ./aws/install`
  - d) **Verify the installation:**  
`aws help`
- 3) **Observe IAM configuration details in the AWS Management Console**
  - ☐ Navigate to the IAM service and review the awsstudent user details, noting the policy document and security credentials.
- 4) **Configure the AWS CLI to connect to your AWS Account**
  - ☐ Run the configuration command:  
`aws configure`  
  
Enter the AWS Access Key ID and Secret Access Key from the Details panel, along with the default region (us-east-1) and output format (json).
- 5) **Observe IAM configuration details using the AWS CLI**
  - ☐ List IAM users to verify configuration:  
`aws iam list-users`

## Activity 1 Challenge: Download the lab\_policy JSON-formatted IAM policy document using AWS CLI

### 1) List policies:

- ❑ `aws iam list-policies --scope Local`  
Identify the ARN of the lab\_policy.

### 2) Get policy version:

- ❑ `aws iam get-policy --policy-arn <lab_policy_arn>`  
Note the default version ID.

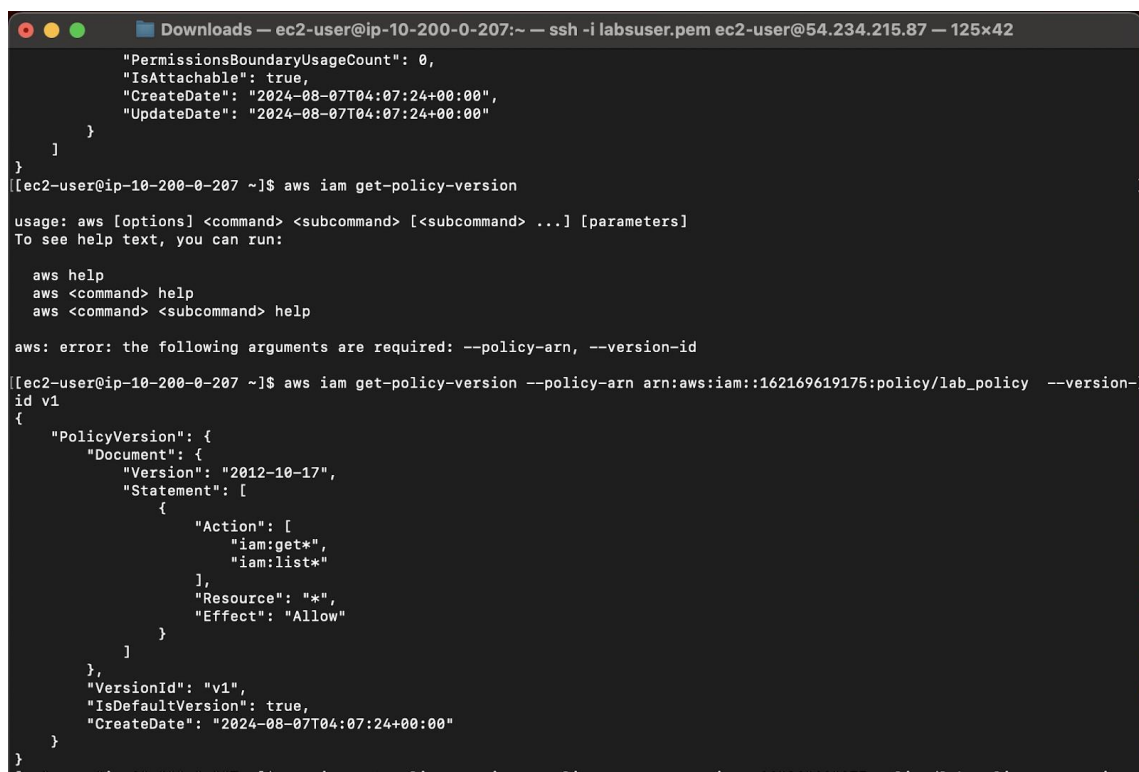
### 3) Get policy document:

- ❑ `aws iam get-policy-version --policy-arn <lab_policy_arn> --version-id <version_id>`

### 4) Save policy document to a file:

- ❑ `aws iam get-policy-version --policy-arn <lab_policy_arn> --version-id <version_id> > lab_policy.json`

## Screenshots:



```
Downloads — ec2-user@ip-10-200-0-207:~ — ssh -i labsuser.pem ec2-user@54.234.215.87 — 125x42
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2024-08-07T04:07:24+00:00",
    "UpdateDate": "2024-08-07T04:07:24+00:00"
  }
}
[ec2-user@ip-10-200-0-207 ~]$ aws iam get-policy-version
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

    aws help
    aws <command> help
    aws <command> <subcommand> help

aws: error: the following arguments are required: --policy-arn, --version-id
[ec2-user@ip-10-200-0-207 ~]$ aws iam get-policy-version --policy-arn arn:aws:iam::162169619175:policy/lab_policy --version-id v1
{
  "PolicyVersion": {
    "Document": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Action": [
            "iam:get*",
            "iam:list*"
          ],
          "Resource": "*",
          "Effect": "Allow"
        }
      ]
    },
    "VersionId": "v1",
    "IsDefaultVersion": true,
    "CreateDate": "2024-08-07T04:07:24+00:00"
  }
}
```

```

inflating: aws/dist/docutils/writers/s5_html/themes/big-white/pretty.css
inflating: aws/dist/docutils/writers/s5_html/themes/big-black/pretty.css
inflating: aws/dist/docutils/writers/s5_html/themes/big-black/_base_
inflating: aws/dist/docutils/writers/s5_html/themes/big-black/framing.css
inflating: aws/dist/docutils/writers/s5_html/themes/small-black/pretty.css
inflating: aws/dist/docutils/writers/s5_html/themes/small-black/_base_
inflating: aws/dist/docutils/writers/s5_html/themes/small-white/pretty.css
inflating: aws/dist/docutils/writers/s5_html/themes/small-white/framing.css
inflating: aws/dist/docutils/writers/html4css1/html4css1.css
inflating: aws/dist/docutils/writers/html4css1/template.txt
inflating: aws/dist/docutils/writers/html5_polyglot/plain.css
inflating: aws/dist/docutils/writers/html5_polyglot/math.css
inflating: aws/dist/docutils/writers/html5_polyglot/minimal.css
inflating: aws/dist/docutils/writers/html5_polyglot/template.txt
inflating: aws/dist/docutils/writers/html5_polyglot/tuftig.css
inflating: aws/dist/docutils/writers/html5_polyglot/responsive.css
inflating: aws/dist/docutils/writers/latex2e/xelatex.tex
inflating: aws/dist/docutils/writers/latex2e/titlingpage.tex
inflating: aws/dist/docutils/writers/latex2e/docutils.sty
inflating: aws/dist/docutils/writers/latex2e/default.tex
inflating: aws/dist/docutils/writers/latex2e/titlepage.tex
inflating: aws/dist/docutils/writers/odf_odt/styles.odt
inflating: aws/dist/docutils/writers/pep_html/template.txt
inflating: aws/dist/docutils/writers/pep_html/pep.css
[ec2-user@ip-10-200-0-207 ~]$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
[ec2-user@ip-10-200-0-207 ~]$ aws help
[ec2-user@ip-10-200-0-207 ~]$ clear

[ec2-user@ip-10-200-0-207 ~]$ aws configure
AWS Access Key ID [None]: AKIASLQQALTSAWI3KHP
AWS Secret Access Key [None]: v91f8ei5lXDibjVRcaJY96nK7GY7u+JUPjxEovyP
Default region name [None]: us-east-1
Default output format [None]: json
[ec2-user@ip-10-200-0-207 ~]$ aws iam list-users
{
  "Users": [
    {
      "Path": "/",
      "UserName": "awsstudent",
      "UserId": "AIDASLQQQALTUJ7RBLUM5",
      "Arn": "arn:aws:iam::162169619175:user/awsstudent",
      "CreateDate": "2024-08-07T04:06:47+00:00"
    }
  ]
}
[ec2-user@ip-10-200-0-207 ~]$ aws iam list-policies
{
  "Policies": [
    {
      "PolicyName": "lab_policy",
      "PolicyId": "ANPASLQQQALTW6ROHKL04",
      "Arn": "arn:aws:iam::162169619175:policy/lab_policy",
      "Path": "/",
      "DefaultVersionId": "v1",
      "AttachmentCount": 1,
      "PermissionsBoundaryUsageCount": 0,
      "IsAttachable": true,
      "CreateDate": "2024-08-07T04:07:24+00:00",
      "UpdateDate": "2024-08-07T04:07:24+00:00"
    },
    {
      "PolicyName": "AdministratorAccess",
      "PolicyId": "ANPAIWMBCSKTIE64ZLYK",
      "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",
      "Path": "/",
      "DefaultVersionId": "v1",

```

In this activity, we successfully installed the AWS CLI on a Red Hat EC2 instance and configured it to connect to an AWS account. Practiced using the AWS CLI to interact with IAM and retrieved a policy document using multiple AWS CLI commands. This hands-on exercise reinforced the concept that any operation available in the AWS Management Console can also be performed using the AWS CLI, demonstrating the flexibility and power of CLI-based AWS management.

## Assignment-2

### Problem Statement: Creating Amazon EC2 Instances

### Procedure:

#### **TASK - 1 : Launch an Amazon EC2 Instance using the Management Console**

1. **Launch EC2 Instance:** Go to the EC2 Console, select "Launch instance," and choose "Bastion Server" as the name.
2. **Set AMI and OS:** Select the default Amazon Linux AMI (Amazon Linux 2023) for the operating system.
3. **Choose Instance Type:** Keep the default t2.micro instance type, suitable for development and testing with 1 CPU and 1 GiB memory.
4. **Assign Key Pair:** Choose the "vockey" key pair to enable SSH access to the instance.
5. **Configure Network Settings:** Set the network to Lab VPC, select the public subnet, enable Auto-assign public IP, and create a security group allowing SSH access on port 22.
6. **Configure Storage:** Use default storage settings in the "Configure storage" section.
7. **Assign IAM Role and Launch:** Set the IAM role to "Bastion-Role" for EC2 access, launch the instance, and verify that it reaches "Running" status with 2/2 checks passed.

#### **TASK - 2 : Launch an Amazon EC2 Instance using the Management Console**

1. **Copy Public IP:** Copy the IPv4 Public IP address of the Bastion Server from the EC2 Console.
2. **Download Credentials:**  
For Windows: Download `labsuser.ppk`.  
For Mac/Linux: Download `labsuser.pem`.
3. **Set Up Connection:**  
For Windows: Install PuTTY if not already installed.  
For Mac/Linux: Open a terminal.
4. **Configure SSH Settings:**  
Windows: Open PuTTY, set the hostname as the Bastion Server's Public IP, and under "Auth," browse and select `labsuser.ppk`.

Mac/Linux: In the terminal, navigate to the directory with `labsuser.pem` and set its permissions with `chmod 400 labsuser.pem`.

5. **Connect to Server:**  
Windows: In PuTTY, choose "Open" and log in as `ec2-user`.  
Mac/Linux: Use the SSH command `ssh -i labsuser.pem ec2-user@<public-ip>`.

## 6. Confirm Connection:

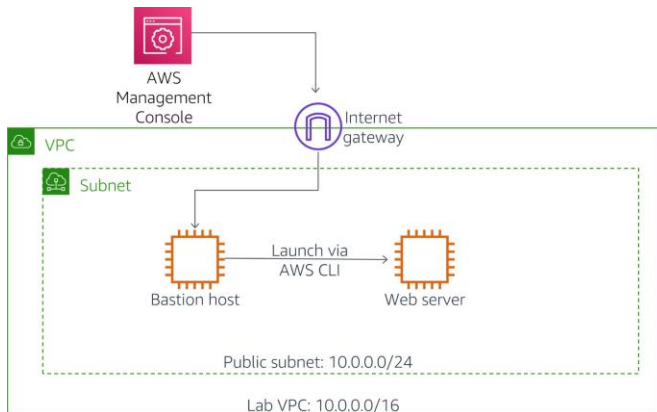
When prompted, type "yes" to allow the first connection.

7. **Access AWS CLI:** You're now connected to the Bastion Server and can use the AWS CLI to interact with AWS services.

## TASK - 3 : Launch an Instance using the AWS CLI

1. **Obtain AMI ID:** Use AWS Systems Manager Parameter Store to get the latest Amazon Linux 2023 AMI ID and store it in the **AMI** environment variable.
2. **Get Subnet and Security Group IDs:** Retrieve and store the Public Subnet ID in **SUBNET** and Web Security Group ID in **SG**.
3. **Download User Data Script:** Download the **UserData.txt** script, which configures the instance as a web server.
4. **Launch the EC2 Instance:** Use the AWS CLI **run-instances** command with the AMI, subnet, security group, user data script, and **t2.micro** instance type. Tag the instance as "Web Server" and save the instance ID in **INSTANCE**.
5. **Check Instance Status:** Run **describe-instances** with **INSTANCE** to monitor the status until it shows "running."
6. **Get Public DNS:** Retrieve the instance's Public DNS name using **describe-instances** and copy it.
7. **Test the Web Server:** Paste the DNS name into a browser to verify that the web server is running.

The final architecture will be:



## Challenge 1: Connect to EC2 Instance

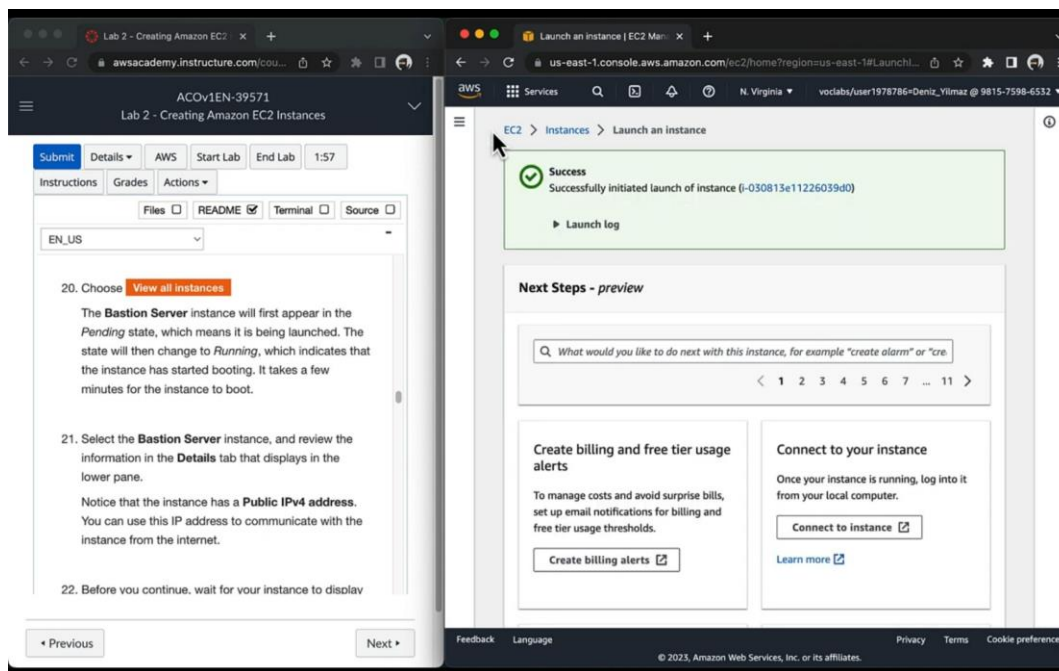
- Obtain the DNS name of the Misconfigured Web Server and attempt SSH access.
- Diagnose and resolve the connection issue.

- Be prepared to explain the problem and solution.

## Challenge 2: Fix Web Server Installation

- Access the Misconfigured Web Server's public IP in a browser to check website availability.
- Diagnose and fix the issue, or launch a new instance with correct settings.
- Be ready to discuss the issue and solution.

### Screenshots :-



Lab 2 - Creating Amazon EC2

ACOV1EN-39571  
Lab 2 - Creating Amazon EC2 Instances

SubmitDetailsAWSStart LabEnd Lab1:54

InstructionsGradesActions

FilesREADMETerminalSource

EN\_US

This will display a status of **pending** or **running**.  
Repeat the above command until it returns a status of **running**.

Test the Web Server

You can now test that the web server is working. You can retrieve a URL to the instance via the AWS CLI.

45. Paste this command:

```
aws ec2 describe-instances --instance-ids $INSTANCE --query 'Reservations[].Instances[].PublicDnsName' --output text
```

This returns the **DNS Name** of the instance.

PreviousNext

Instances | EC2 Management

us-east-1 console.aws.amazon.com/ec2/home?region=us-east-1#instances:~:topo=tree

Services

N. Virginia

vociabs/user1978786-Deniz\_Yilmaz @ 9815-7598-6532

Instances (1/2) Info

ConnectInstance stateActionsLaunch instances

Find instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type
Misconfigured Web Server	i-0831d943b132d2c6d	Running	t2.micro

```
"HibernationOptions": {
  "Configured": false
},
"MetadataOptions": {
  "State": "pending",
  "HttpEndpoint": "enabled",
  "HttpTokens": "optional",
  "HttpPutResponseHopLimit": 1
},
"AmiLaunchIndex": 0
},
"ReservationId": "r-0ca725190dd541234",
"Groups": [],
"OwnerId": "981575986532"
}
[ec2-user@ip-10-0-0-189 ~]$ aws ec2 describe-instances --instance-ids $INSTANCE --query 'Reservations[].Instances[].State.Name' --output text
running
[ec2-user@ip-10-0-0-189 ~]$ aws ec2 describe-instances --instance-ids $INSTANCE --query 'Reservations[].Instances[].PublicDnsName' --output text
ip-10-0-0-189.ec2.internal
```

FeedbackLanguagePrivacyTermsCookie preferences

© 2023, Amazon Web Services, Inc. or its affiliates.

Lab 2 - Creating Amazon EC2

ACOV1EN-39571  
Lab 2 - Creating Amazon EC2 Instances

SubmitDetailsAWSStart LabEnd Lab1:53

InstructionsGradesActions

FilesREADMETerminalSource

EN\_US

This returns the **DNS Name** of the instance.

46. Copy the DNS name that is displayed.  
It should look similar to: `ec2-35-11-22-33.compute-1.amazonaws.com`

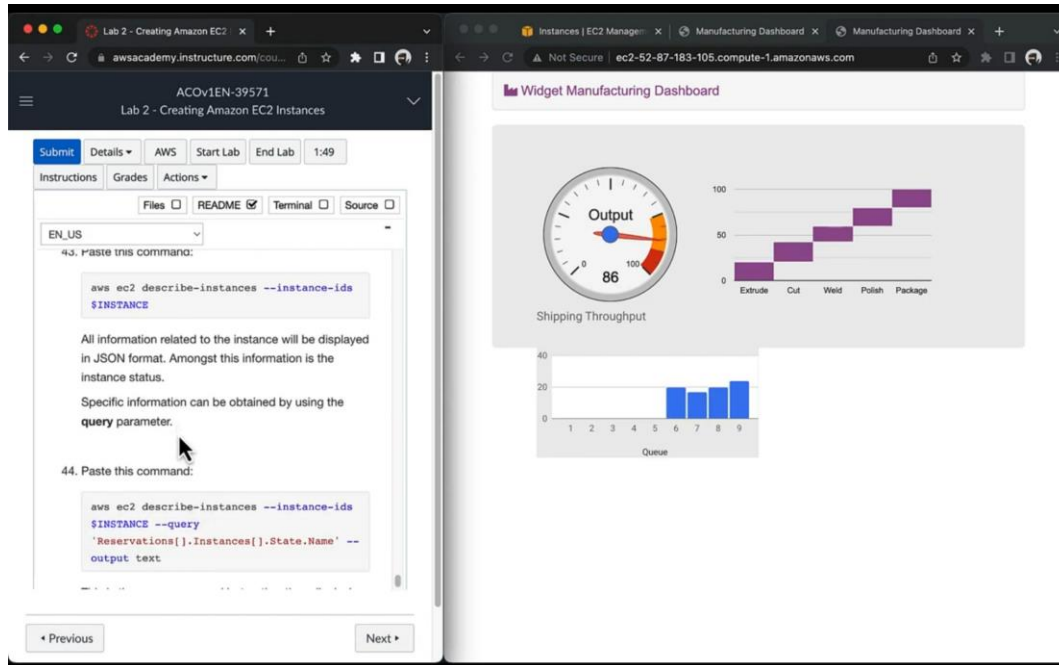
47. Paste the DNS name into a new web browser tab, then press Enter.  
A web page should be displayed, demonstrating that the web server was successfully launched and configured.  
You can also see the instance in the EC2 management console.

PreviousNext

Widget Manufacturing Dashboard

The dashboard displays two main visualizations:

- Shipping Throughput:** A gauge chart on the left shows a value of 77. To its right is a bar chart showing throughput for five stages: Extrude, Cut, Weld, Polish, and Package. The throughput increases progressively from Extrude to Package.
- Queue:** A bar chart at the bottom shows a single bar with a value of 9.



**Conclusion** :- Deploying secure infrastructure is faster and more efficient. As a Systems Operator, you can leverage the AWS Command-Line Interface to automate these tasks, simplifying processes. By completing this lab, you now have the skills to launch Amazon EC2 instances both through the management console and via the AWS CLI.



## Assignment-3

**Problem Statement:** Create a new Amazon Machine Image (AMI) from an existing Amazon Elastic Compute Cloud (Amazon EC2) instance.

### **Procedure:**

#### **Task 1: Access the AWS Management Console**

1. Start the lab by selecting "Start Lab" at the top of the instructions.
2. Wait for the lab status to change to "ready," then close the Start Lab panel.
3. Select "AWS" to open the AWS Management Console in a new tab, ensuring you can see both the console and the lab instructions simultaneously.
4. Do not change the AWS Region during the lab.

#### **Task 2: Create a New AMI for Amazon EC2 Auto Scaling**

##### **1. Log in to the Command Host Instance:**

- For Windows Users:
  - Access the EC2 service and select the Command Host instance.
  - Copy the Public IPv4 address and use PuTTY to SSH into the instance with the downloaded .ppk key.
- For Mac/Linux Users:
  - Access the EC2 service and select the Command Host instance.
  - Copy the Public IPv4 address and use the terminal to SSH into the instance with the downloaded .pem key.

##### **2. Create a New EC2 Instance:**

- Use the AWS CLI on the Command Host to create a new EC2 instance:
  - Retrieve the necessary values (AMI ID, Security Group, Subnet ID) from the details of the Command Host instance.

Execute the following command, replacing the placeholders with the retrieved values:

bash

Copy code

```
aws ec2 run-instances --key-name vockey --instance-type t2.micro
--image-id <AmiId> --user-data
file:///home/ec2-user/UserData.txt --security-group-ids
<HTTPAccess> --subnet-id <SubnetId>
--associate-public-ip-address --tag-specifications
'ResourceType=instance,Tags=[{Key=Name,Value=WebServerBaseImage}
]' --output text --query 'Instances[*].InstanceId'
```

### 3. Monitor the Instance Status:

Use the following command to wait until the new instance is running:

bash

Copy code

```
aws ec2 wait instance-running --instance-ids <NEW-INSTANCE-ID>
```

Obtain the public DNS name of the instance to verify the web server installation:

bash

Copy code

```
aws ec2 describe-instances --instance-id <NEW-INSTANCE-ID>
--query
'Reservations[0].Instances[0].NetworkInterfaces[0].Association.P
ublicDnsName'
```

- Open a web browser and navigate to `http://<PUBLIC-DNS-ADDRESS>/index.php` to check if the web server is running.

### 4. Create a Custom AMI:

Execute the following command to create a new AMI from the created EC2 instance:

bash

Copy code

```
aws ec2 create-image --name WebServer --instance-id
<NEW-INSTANCE-ID>
```

## Task 3: Create an Auto Scaling Environment

### 1. Create an Application Load Balancer:

- In the AWS Management Console, navigate to EC2 and then to Load Balancers.
- Choose "Create Load Balancer" and select "Application Load Balancer."
- Configure the following settings:
  - Load balancer name: `webserverloadbalancer`
  - VPC: Select Lab VPC
  - Subnets: Select both public subnets
  - Security groups: Select `HTTPAccess`
- Create a target group named `webserver-app` and configure the health check settings:
  - Health check path: `/index.php`
  - Healthy threshold: `2`
  - Interval: `10` seconds
- Complete the load balancer creation process.

### 2. Create a Launch Template:

- Navigate to Launch Templates in the EC2 dashboard and select "Create launch template."

- Set the following parameters:
  - Launch template name: `WebServerLaunchTemplate`
  - AMI: Search and select `WebServer`
  - Instance type: `t2.micro`
  - Security groups: Select `HTTPAccess`
  - Enable detailed CloudWatch monitoring
- Save the launch template.
- 3. **Create an Auto Scaling Group:**
  - Navigate to Auto Scaling Groups and select "Create Auto Scaling group."
  - Configure the settings:
    - Auto Scaling group name: `WebServersASGroup`
    - Launch template: Choose `WebServerLaunchTemplate`
    - VPC: Select `Lab VPC`
    - Subnets: Choose Private Subnet 1 and Private Subnet 2
    - Attach to an existing load balancer: Choose `webserver-app`
    - Enable group metrics collection in CloudWatch
    - Set desired capacity: `2`, minimum capacity: `2`, maximum capacity: `4`
    - Configure scaling policies for average CPU utilization targeting `45%`
  - Create the Auto Scaling group.

#### Task 4: Verify the Auto Scaling Configuration

1. Navigate to Instances in the EC2 dashboard to ensure that two new instances are created in your Auto Scaling group.
2. Check the Target Groups section to confirm that the status of these instances is healthy.
3. Obtain the DNS name of the load balancer and use it to access the web application.
4. Initiate the stress test by selecting "Start Stress" on the web page to increase CPU utilization.
5. Monitor the Auto Scaling group activity to verify that new EC2 instances are launched in response to the increased load.

#### Challenge 1: Create a New AMI for Amazon EC2 Auto Scaling

- Launch a new EC2 instance and create an AMI from it using the AWS CLI.

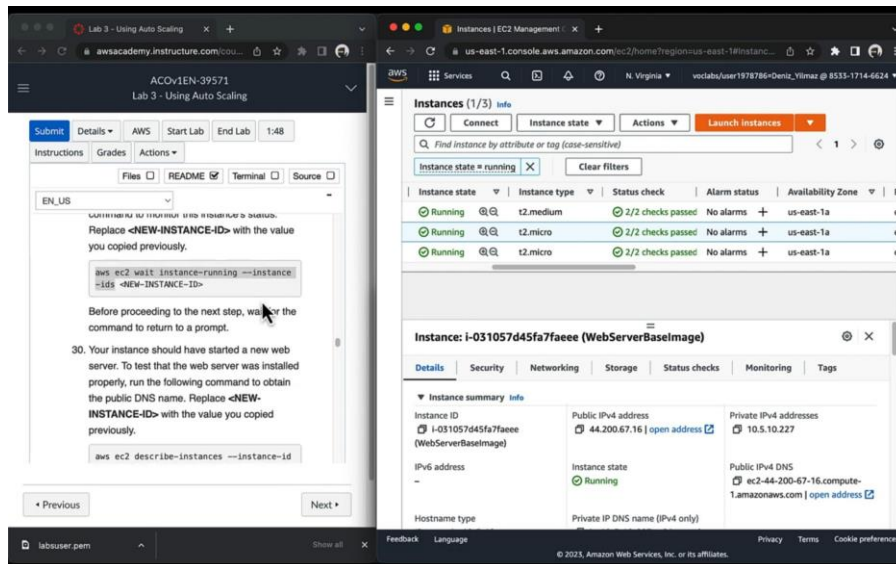
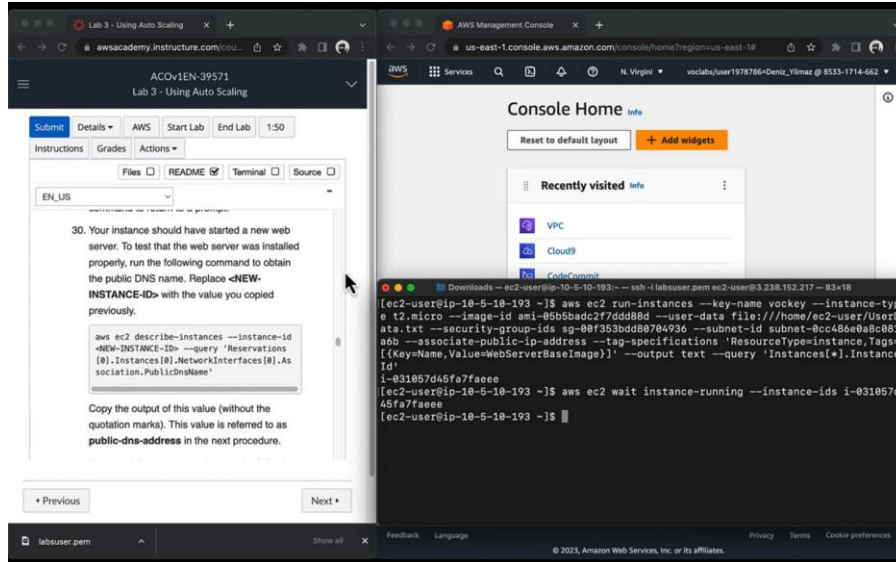
#### Challenge 2: Create an Auto Scaling Environment

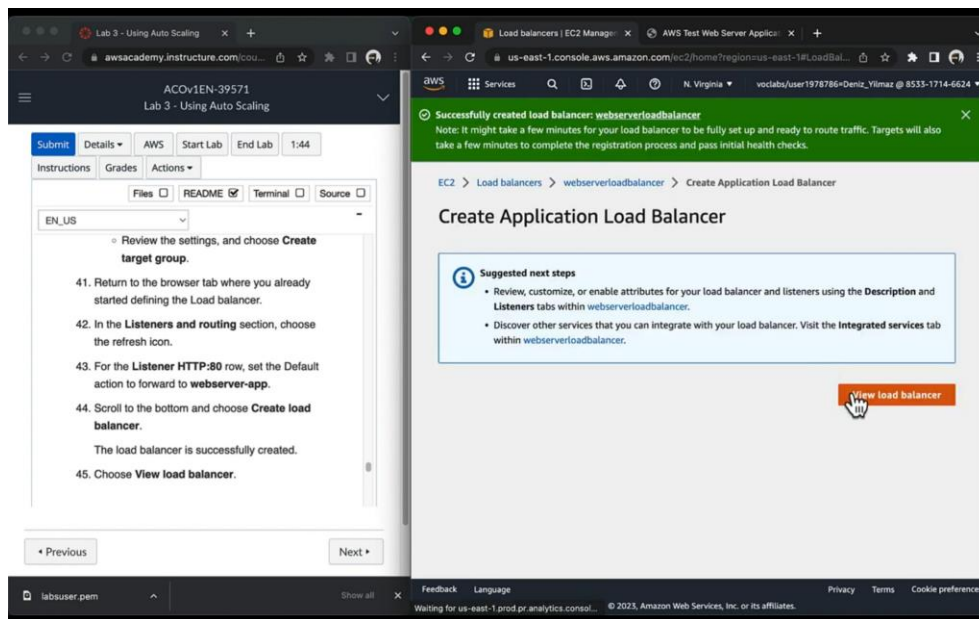
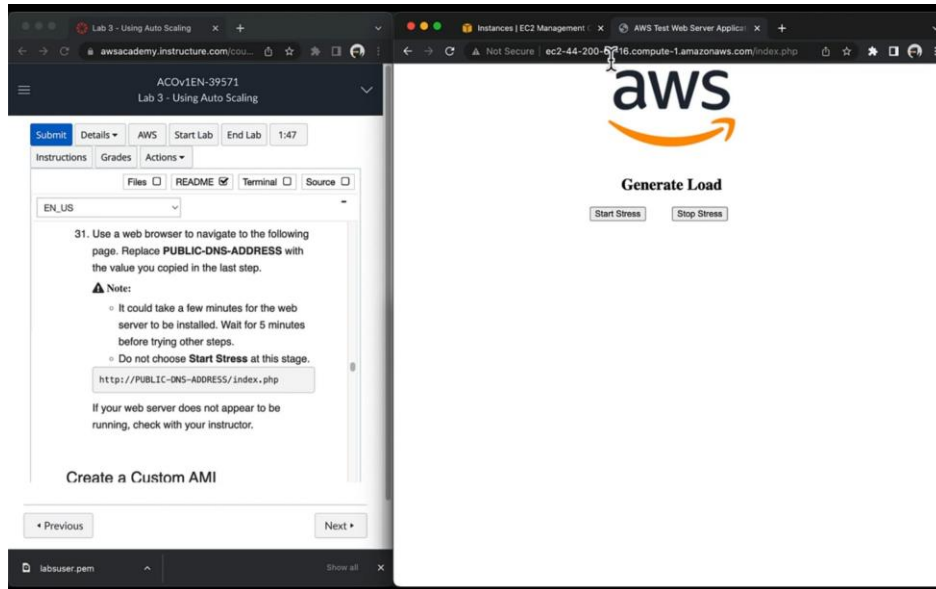
- Create an Application Load Balancer and configure subnets and security groups.
- Set up a Launch Template with the created AMI for Auto Scaling.
- Configure an Auto Scaling Group with desired capacity, scaling policies, and CPU utilization thresholds.

#### Challenge 3: Verify the Auto Scaling Configuration

- Test the Auto Scaling setup by initiating a stress test on the web server to trigger scale-up.

## Screenshots:

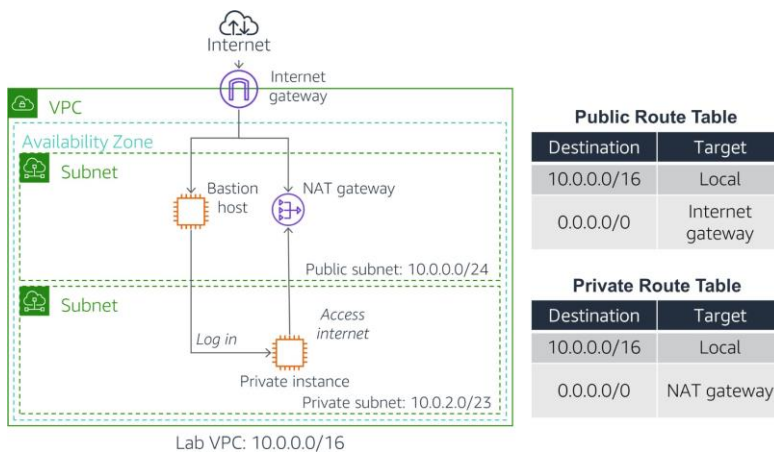




**Conclusion :** The lab demonstrated how to create and configure a scalable web server system with AWS Auto Scaling. By using AMIs, load balancing, and scaling policies, the setup ensures automatic response to increased loads, maintaining performance and availability.

## Assignment-4

**Problem Statement:** Configure a Virtual Private Cloud (VPC) in AWS that includes both public and private subnets, an Internet Gateway (IGW), a NAT Gateway, and a Bastion Host. This setup will enable secure communication between resources in a private subnet and the internet, while maintaining isolation from direct external access.



## Procedure:

### TASK- 1 : Create a Virtual Private Cloud (VPC)

- Open the AWS Management Console and navigate to the VPC service.
- Ensure the "New VPC Experience" is enabled.
- Select "Your VPCs" and click on "Create VPC."
- Configure the VPC with the following settings:
  - Name tag: **Lab VPC**
  - IPv4 CIDR block: **10.0.0.0/16**
- After creation, enable DNS hostnames for the VPC.

### TASK- 2 : Create Subnets

- In the VPC console, navigate to "Subnets" and click "Create subnet."
- Create a Public Subnet:
  - Configure with:
    - VPC ID: **Lab VPC**
    - Subnet name: **Public Subnet**
    - Availability Zone: Select the first AZ
    - IPv4 CIDR block: **10.0.0.0/24**
  - Enable auto-assign public IPv4 address.

- Create a Private Subnet:
  - Configure with:
    - VPC ID: `Lab VPC`
    - Subnet name: `Private Subnet`
    - Availability Zone: Select the first AZ
    - IPv4 CIDR block: `10.0.2.0/23`

### **TASK- 3 Create an Internet Gateway**

- In the left navigation pane, choose "Internet gateways" and select "Create internet gateway."
- Name the gateway `Lab IGW` and attach it to `Lab VPC`.

### **TASK- 4 :Configure Route Tables**

- Navigate to "Route Tables" and select the route table associated with `Lab VPC`.
- Rename it to `Private Route Table`.
- Create a new route table named `Public Route Table`.
- Edit the routes to add a route that directs `0.0.0.0/0` to the `Lab IGW`.
- Associate the public subnet with the `Public Route Table`.

### **TASK- 5 : Launch a Bastion Server in the Public Subnet**

- Go to the EC2 console and select "Launch instance."
- Configure the instance:
  - Name: `Bastion Server`
  - AMI: Amazon Linux 2023
  - Instance Type: `t2.micro`
  - Key Pair: `vockey`
  - Network settings: Set to `Public Subnet` with auto-assign public IP enabled.
  - Security Group: Configure to allow SSH access.

### **TASK- 6 : Create a NAT Gateway**

- In the VPC console, navigate to "NAT gateways" and select "Create NAT gateway."
- Choose the `Public Subnet` and allocate an Elastic IP.
- Configure the `Private Route Table` to route `0.0.0.0/0` traffic to the NAT Gateway.

### **TASK- 7 : (Optional) Test the Private Subnet**

- Launch an Amazon EC2 instance in the Private Subnet with:
  - Name: `Private Instance`
  - AMI: Amazon Linux 2023
  - Instance Type: `t2.micro`

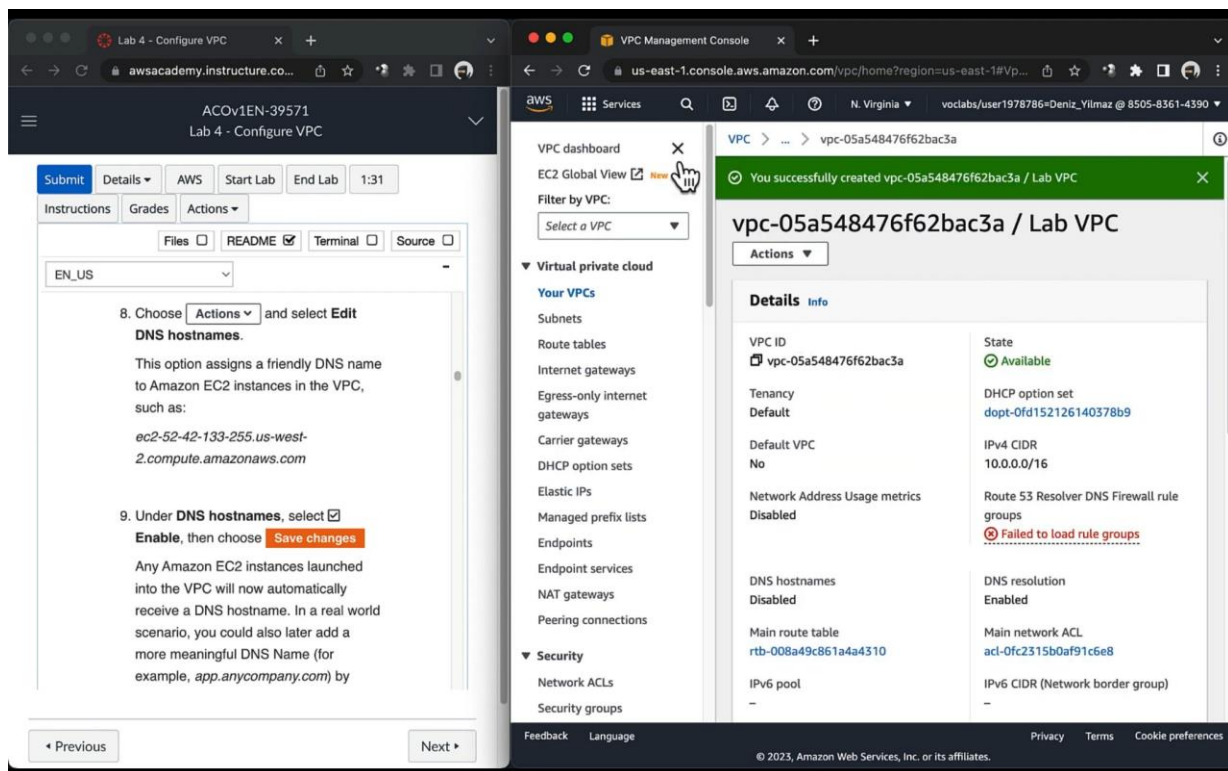


- Key Pair: `vockey`
- Security Group: Allow SSH access, but disable public IP assignment.
- Log into the Bastion Server and then SSH into the Private Instance using its private IP.

## Challenges:

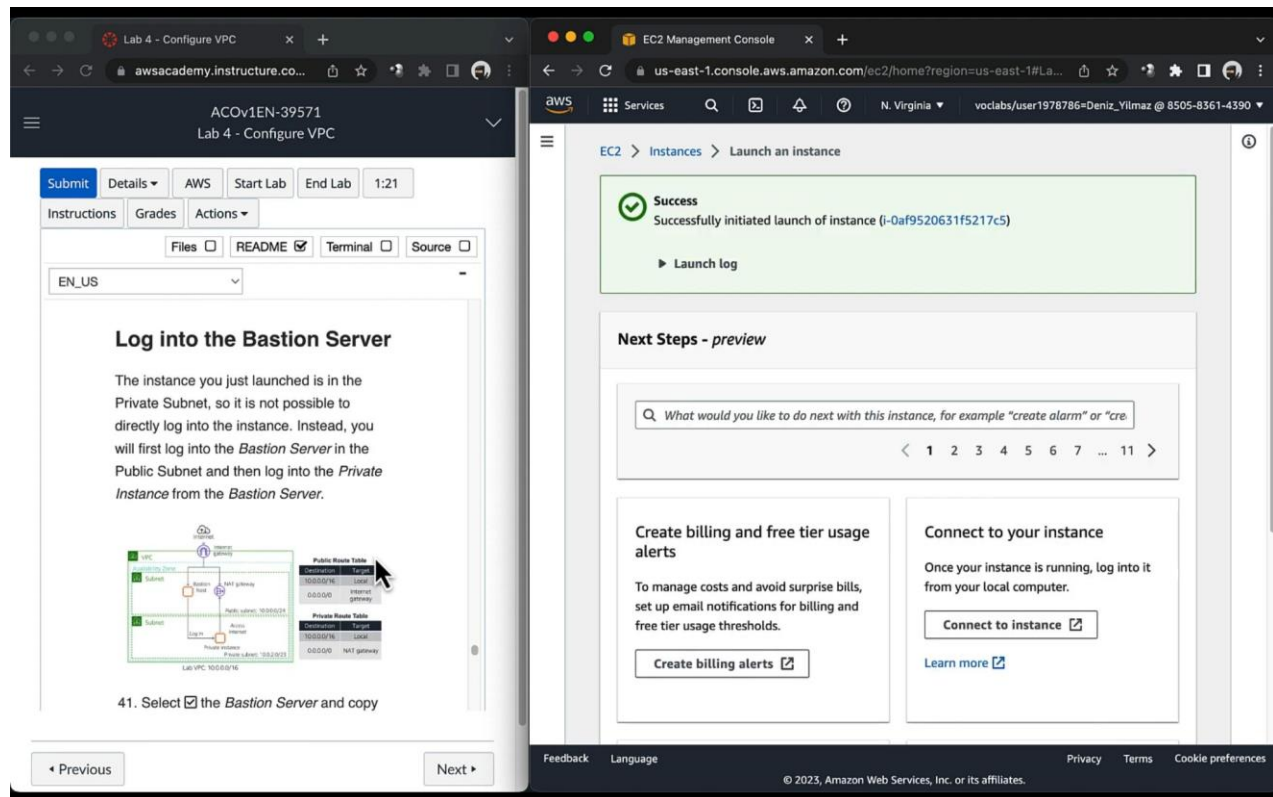
- **Configuration Complexity:** Setting up the VPC, subnets, and gateways required careful attention to IP address ranges and routing configurations to ensure proper communication and isolation.
- **Security Group Management:** Ensuring that security groups allowed necessary access while maintaining security was crucial, particularly for the Bastion Server and the instances in the private subnet.
- **NAT Gateway Setup:** Configuring the NAT Gateway and ensuring that the route tables were correctly directing traffic was a challenging aspect of maintaining secure outbound access from the private subnet.
- **Troubleshooting Connectivity:** If the "Private Instance" could not connect to the internet, it required troubleshooting of the NAT Gateway setup and route table configurations.

## Screenshots:





[illegible]



**Conclusion:** By completing this lab, participants gain practical experience in setting up a secure VPC architecture in AWS. They learn to create and configure VPC components, including subnets, Internet Gateways, and NAT Gateways. The lab provides insight into AWS networking principles and practices, including maintaining security through isolation while allowing necessary external communication for resources within the private subnet.

## **Assignment-5**

**Problem Statement:** To learn how to manage storage on AWS, including creating EC2 instances, configuring AWS CLI, managing Amazon EBS snapshots, and synchronizing files with Amazon S3.

### **Procedure:**

#### **Task 1: Create and Configure AWS Resources**

1. Create Amazon EC2 Instance: Set up an EC2 instance that will be used to administer AWS resources.
2. Set Up AWS CLI: Install and configure the AWS Command Line Interface (CLI) on the EC2 instance for managing resources.
3. Create Amazon S3 Bucket: Access the S3 service in the AWS Management Console and create a unique bucket for storing log files.
4. Attach IAM Role: Attach an instance profile (S3BucketAccess role) to allow the EC2 instance permissions to access the S3 bucket.

#### **Task 2: Create EBS Snapshots**

1. Connect to EC2 Instance: Use SSH to connect to the EC2 instance (instructions vary for Windows and Mac/Linux).
2. Identify Volume ID: Retrieve the volume ID of the Amazon EBS volume attached to the EC2 instance.
3. Stop EC2 Instance: Shut down the instance to ensure a consistent snapshot of the EBS volume.
4. Create Initial Snapshot: Use the AWS CLI to create a snapshot of the EBS volume.
5. Restart EC2 Instance: Start the instance after taking the snapshot.
6. Automate Snapshot Creation with Cron: Schedule recurring snapshots of the EBS volume every minute using a cron job.

#### **Task 3: Retain Only the Last Two Snapshots**

1. Run Python Script: Use a pre-written Python script to maintain only the last two snapshots for the EBS volume, removing older snapshots to manage storage and costs.

#### **Task 4: Challenge – Synchronize Local Directory with Amazon S3**

1. Download Sample Files: Download and unzip a sample set of files on the EC2 instance.
2. Enable Versioning on S3 Bucket: Configure versioning on the S3 bucket to retain different versions of files.
3. Sync Local Directory with S3: Use AWS CLI commands to synchronize the local directory with the S3 bucket.
4. Enable Automatic File Deletion: Configure the sync command to remove files from S3 when they are deleted locally.
5. Restore Deleted Files: Use versioning to retrieve and restore deleted files from S3.

## Challenges:

### **Challenge 1: Synchronize Contents to S3**

Description: Synchronize files from a local directory on the EC2 instance to an Amazon S3 bucket, ensuring all contents remain up-to-date between the local system and S3.

### **Challenge 2: Automate Snapshot Creation with Consistency**

Description: Configure cron jobs to automate snapshot creation for the EC2 instance without requiring manual intervention, aiming to maintain consistent, scheduled backups.

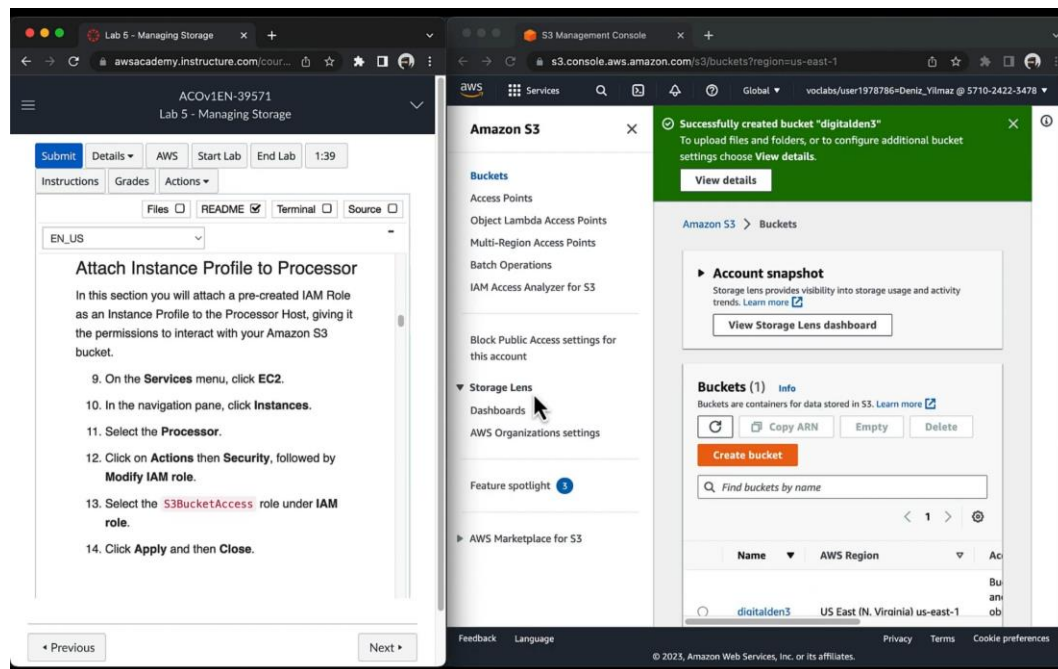
### **Challenge 3: Manage Snapshot Retention**

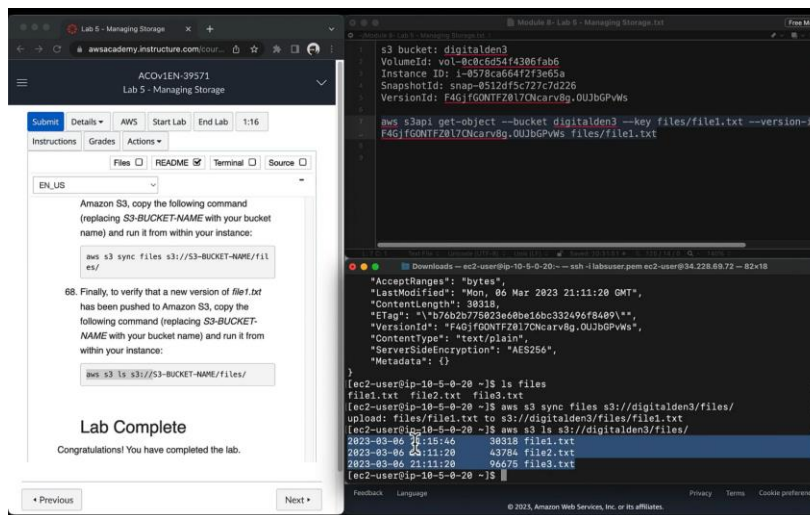
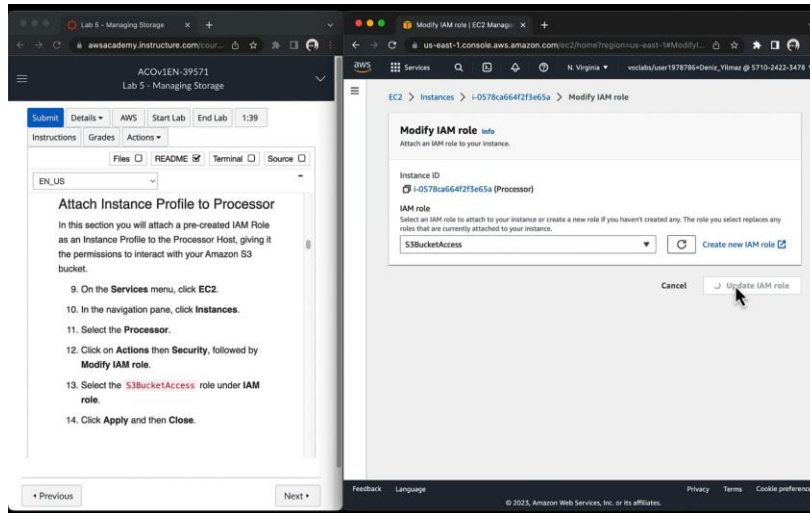
Description: Implement a Python script that automatically deletes older snapshots, ensuring only the last two snapshots are retained. This prevents storage costs from increasing unnecessarily.

### **Challenge 4: File Recovery from S3 Versioning**

Description: Test S3 versioning by deleting and restoring a specific file, confirming versioning's capability to recover previous versions.

## Screenshot:





**Conclusion:** Upon completing this lab, the user will have acquired skills in managing storage resources on AWS, including taking and maintaining snapshots, configuring cron jobs for automation, using AWS CLI for efficient data management, and synchronizing files with S3 for data backup and recovery.

## **Assignment-6**

**Problem Statement:** Demonstrate how to monitor applications and infrastructure using Amazon CloudWatch Metrics, Logs, Events, and AWS Config.

### **Procedure:**

#### **Task 1: Install and Configure CloudWatch Agent**

1. Install the Agent: Use AWS Systems Manager Run Command to deploy the CloudWatch Agent to target EC2 instances.
2. Configure Log Collection: Specify the log files to be monitored, including system logs (e.g., `/var/log/messages`) and application logs (e.g., web server access logs).
3. Configure Metric Collection: Define the system and application metrics to be collected, such as CPU utilization, memory usage, disk I/O, and custom application metrics.

#### **Task 2: Monitor Application Logs with CloudWatch Logs**

1. Create Log Groups: Define log groups to organize and store log data.
2. Configure Log Filters: Create filters to extract specific patterns or keywords from log data (e.g., error messages, specific HTTP status codes).
3. Set Up Alarms: Create alarms to trigger notifications or automated actions based on log data, such as:
  - High error rates
  - Unusual traffic patterns
  - Specific error messages

#### **Task 3: Monitor Instance Metrics with CloudWatch Metrics**

1. Identify Key Metrics: Determine the critical metrics for your applications, such as CPU utilization, memory usage, disk I/O, and network traffic.
2. Create Dashboards: Visualize metrics using CloudWatch dashboards to gain insights into system performance.
3. Set Up Alarms: Configure alarms to notify you of abnormal metric values, such as:
  - High CPU utilization
  - Low disk space
  - Network errors

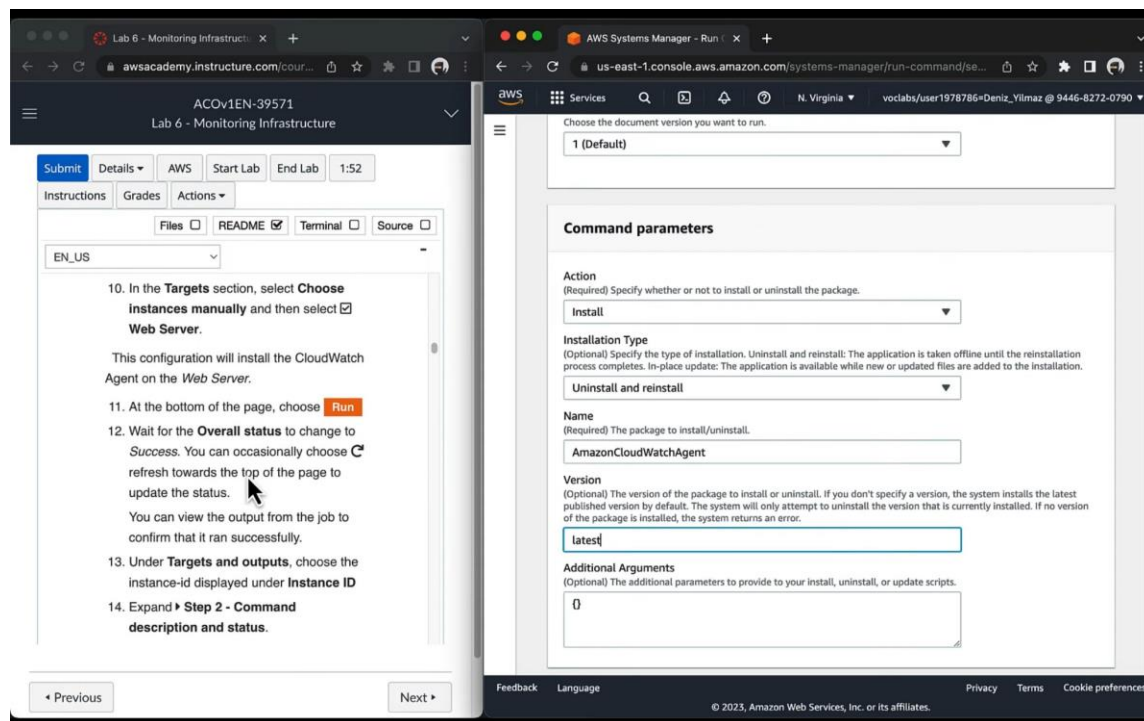
#### **Task 4: Create Real-Time Notifications with CloudWatch Events**

1. Define Event Patterns: Specify the events that should trigger notifications, such as EC2 instance state changes, Lambda function errors, or S3 object creation.
2. Configure Targets: Determine the target recipients for notifications, such as SNS topics, Lambda functions, or other AWS services.
3. Test Notifications: Verify that notifications are delivered to the intended recipients and contain relevant information.

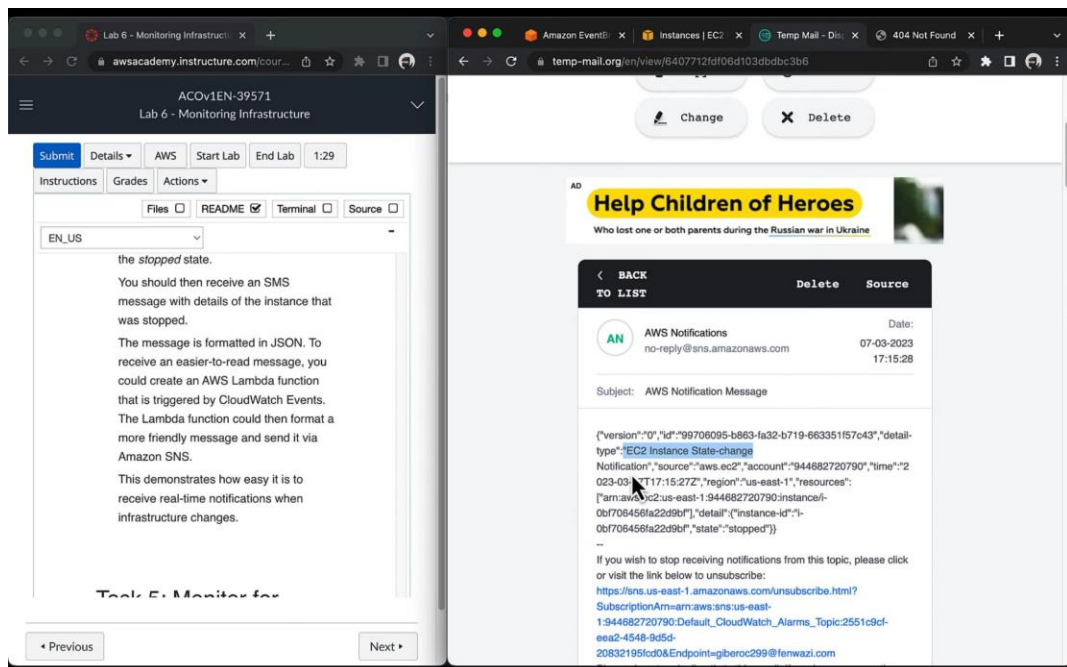
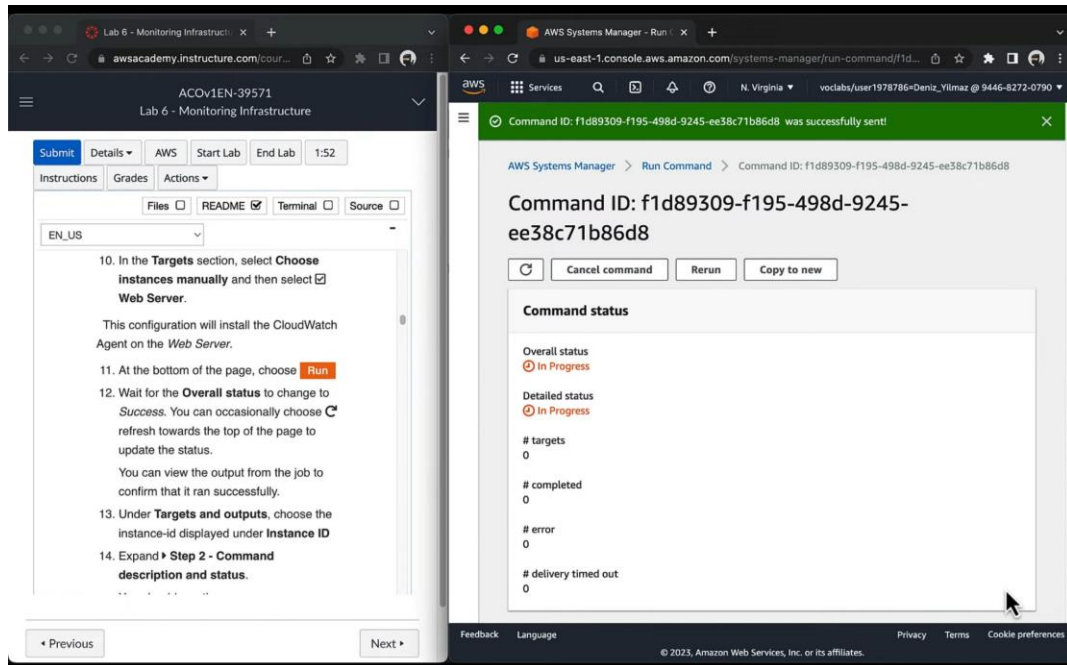
## Task 5: Monitor for Infrastructure Compliance with AWS Config

1. Create Config Rules: Define rules to assess resource configurations against compliance standards.
2. Evaluate Resource Configurations: Regularly evaluate resource configurations to identify deviations from policies.
3. Remediate Non-Compliant Resources: Implement automated remediation actions or manual intervention to address compliance issues.

### Screenshot:







**Conclusion:** By effectively utilizing Amazon CloudWatch Metrics, Logs, Events, and AWS Config, organizations can gain valuable insights into their infrastructure's health and performance. This lab demonstrated how to set up and configure these services to monitor applications, track system metrics, and ensure compliance with organizational standards.



## **Assignment-7**

**Problem Statement:** Lab teaches you how to leverage tags for managing and automating actions on AWS resources, specifically Amazon EC2 instances. Identify resources based on tags using the AWS CLI. Use tags to control stop and start actions for EC2 instances.

### **Procedure:**

#### **Task 1: Using Tags to Find Resources**

- Connect to the Command Host instance using SSH.
- Use AWS CLI commands with JMESPath to find instances based on specific tags (Project and Environment).
- Modify the command to display additional information like Availability Zone and Version tags.
- Use multiple tag filters to identify instances associated with a specific project and environment.
- Run a script to automatically change the Version tag for development instances.

#### **Task 2: Stop and Start Resources by Tag**

- Examine the stopinator.php script that utilizes the AWS SDK for PHP to stop and start instances based on tags.
- Use the script to stop and then restart EC2 instances tagged as part of a development environment for a specific project.

#### **Task 3: Challenge: Terminate Non-Compliant Instances**

- (Optional) Develop a solution to automatically terminate instances that lack a specific tag (e.g., Environment tag) using either AWS CLI commands or the PHP SDK.

### **Challenges:**

- Complexity: Scripting and managing tag-based automation can be complex, especially for large-scale deployments.
- Accidental Termination: Incorrect termination rules could lead to unintentional resource loss.
- Over-reliance: Excessive dependence on tags can limit flexibility in managing resources.

## Screenshot:

The screenshot displays two browser windows. The left window, titled 'Lab 7 - Managing Resources', shows the AWS Cloud Labs interface for session ACOv1EN-39571. It lists lab resources including IP addresses and provides buttons for downloading the SSH key, PEM file, and SSO URL. The right window shows the AWS Management Console 'Console Home' for the us-east-1 region. Below the console home, a terminal window is open, showing commands to navigate to the 'downloads' directory, change permissions on '400 labsuser.pem', and execute an SSH command to connect to an EC2 instance.

The screenshot displays two browser windows. The left window, titled 'Lab 7 - Managing Resources', shows the AWS Cloud Labs interface for session ACOv1EN-39571. It displays the 'Instructions' tab, which includes a script for creating EC2 instances and tags. The right window shows the AWS Management Console 'Instances | EC2 Management' page. It lists three EC2 instances: 'Command Host', 'app server', and 'web server', all in a 'Running' state. Below the instances list, a terminal window is open, showing the command to run 'change-resource-tags.sh' and the output of the 'aws ec2 describe-instances' command.

Lab 7 - Managing Resources

ACOV1EN-39571

Lab 7 - Managing Resources

Submit Details AWS Start Lab End Lab 1:44

Instructions Grades Actions

Files README Terminal Source

EN\_US

40. On the **Services** menu, choose **EC2**.

41. In the navigation pane, choose **Instances**.

42. Verify that two instances are stopping or have already been stopped.

43. Return to the SSH session for Command Host, and from the Linux prompt, restart your instances with the following command:

```
./stopinator.php -t"Project=ERPSystem;Env=ironseent=development" -s
```

44. Return to the EC2 Management Console window and verify that the two instances that were previously shut down are now restarting.

Task 3: Challenge:

Previous Next

Instances | EC2 Management

us-east-1 console.aws.amazon.com/ec2/home?region=us-east-1#instan...

Services

N. Virginia

voclabs/user1978786-Deniz\_Yilmaz @ 4829-1946-1763

Instances (2/9) info

Connect Instance state Actions Launch instances

Find instance by attribute or tag (case-sensitive)

Name	Instance ID	Instance state	Instance type
app server	i-09f55a120649ea840	Running	t2.micro
web server	i-0c2a678506e7b0e25	Running	t2.micro
web server	i-0d97b0430adf54984	Running	t2.micro
web server	i-0970c4b8aa9811462	Stopped	t2.micro
app server	i-03ba534914172ac04	Stopping	t2.micro

Instances: i-0970c4b8aa9811462 (web server), i-03ba534914172ac04 (app server)

Monitoring

1h 3h 12h 1d 3d 1w Custom Add to dashboard

CPU utilization (%)

Status check fail...

Status check fail...

Status check fail...

Feedback Language

© 2023, Amazon Web Services, Inc. or its affiliates.

Lab 7 - Managing Resources

ACOV1EN-39571

Lab 7 - Managing Resources

Submit Details AWS Start Lab End Lab 1:38

Instructions Grades Actions

Files README Terminal Source

EN\_US

```
./terminate-instances.php --region <region> --subnet-id <subnet-id>
```

You should see something similar to the following results:

```

Checking i-dd3a90d1
Checking i-a4248ea8
Checking i-793a9075
Checking i-a9248ea5
Checking i-aa248ea6
Checking i-da3a90c0
Checking i-a13b91a0
Checking i-a23b91ae
Checking i-ab248ea7
Terminating instances...
Instances terminated.

```

Previous Next

Module 10 - Lab 7 - Managing Resources with Tagging.txt

Free Mode

us-east-1 subnet-071e955d54caa6076

Module 10 - Lab 7 - Managing Resources with Tagging.txt

Downloads - ec2-user@ip-10-5-0-139:~/aws-tools - ssh -i labuser.pem ec2-user@35.174.171.117 - 75x18

[ec2-user@ip-10-5-0-139 aws-tools]\$ nano terminate-instances.php

[ec2-user@ip-10-5-0-139 aws-tools]\$ ./terminate-instances.php --region us-east-1 --subnet-id subnet-071e955d54caa6076

```

Checking i-08a1c50fd2b737358
Checking i-09f73f52e3cf95148
Checking i-09f55a120649ea840
Checking i-03ba534914172ac04
Checking i-0c2a678506e7b0e25
Checking i-0970c4b8aa9811462
Checking i-0d97b0430adf54984
Terminating instances...
Instances terminated.

```

[ec2-user@ip-10-5-0-139 aws-tools]\$

Feedback Language

© 2023, Amazon Web Services, Inc. or its affiliates.

**Conclusion:** By effectively utilizing tags, you can gain significant control over your AWS resources. Tagging allows you to Perform automated actions based on resource attributes. Simplify resource grouping and discovery. Enforce compliance policies by identifying non-compliant resources. However, remember to implement tagging strategies thoughtfully and with proper safeguards to minimize risks and ensure efficient resource management.

## **Assignment-8**

**Problem Statement:** To learn how to automate infrastructure provisioning using AWS CloudFormation. Define infrastructure as code using CloudFormation templates. Deploy, update, and delete stacks. Understand the basic structure of CloudFormation templates.

### **Procedure:**

#### **Task 1: Deploy a Basic VPC**

1. Create a CloudFormation template defining a VPC and security group.
2. Deploy the template to create the specified resources.
3. Monitor the stack creation process.

#### **Task 2: Add an S3 Bucket**

1. Modify the template to include an S3 bucket resource.
2. Update the stack to deploy the S3 bucket.

#### **Task 3: Add an EC2 Instance**

1. Add an EC2 instance resource to the template, referencing existing resources and using a Systems Manager Parameter Store parameter for the AMI ID.
2. Update the stack to deploy the EC2 instance.

#### **Task 4: Delete the Stack**

1. Delete the CloudFormation stack to remove all associated resources

### **Challenges:**

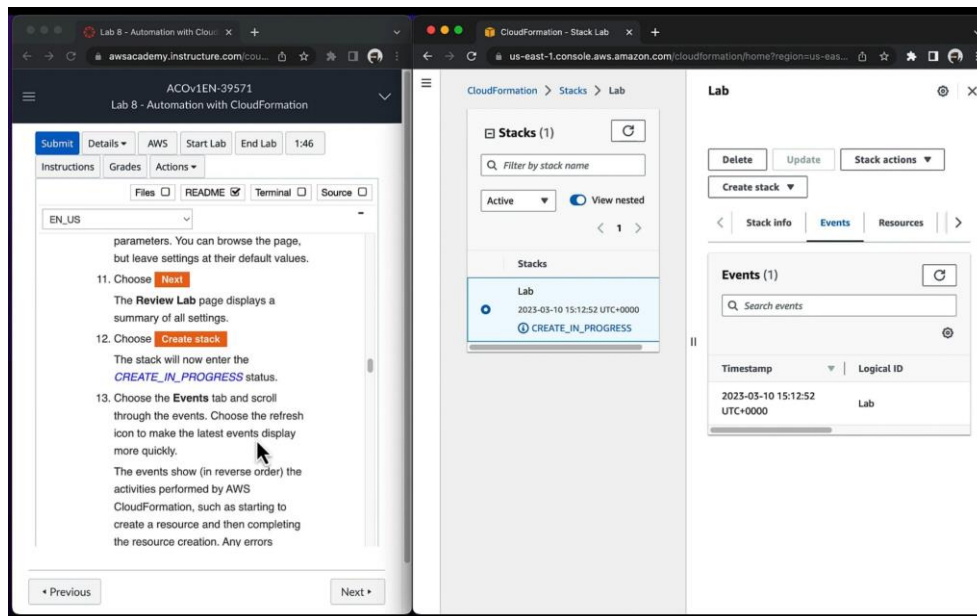
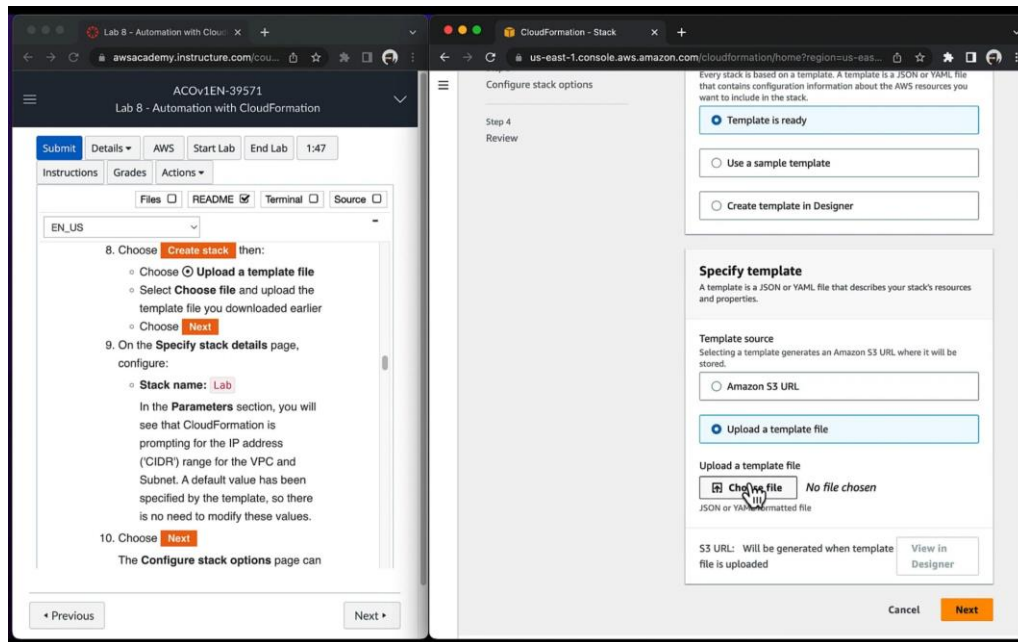
Template Complexity: Creating complex templates can be challenging.

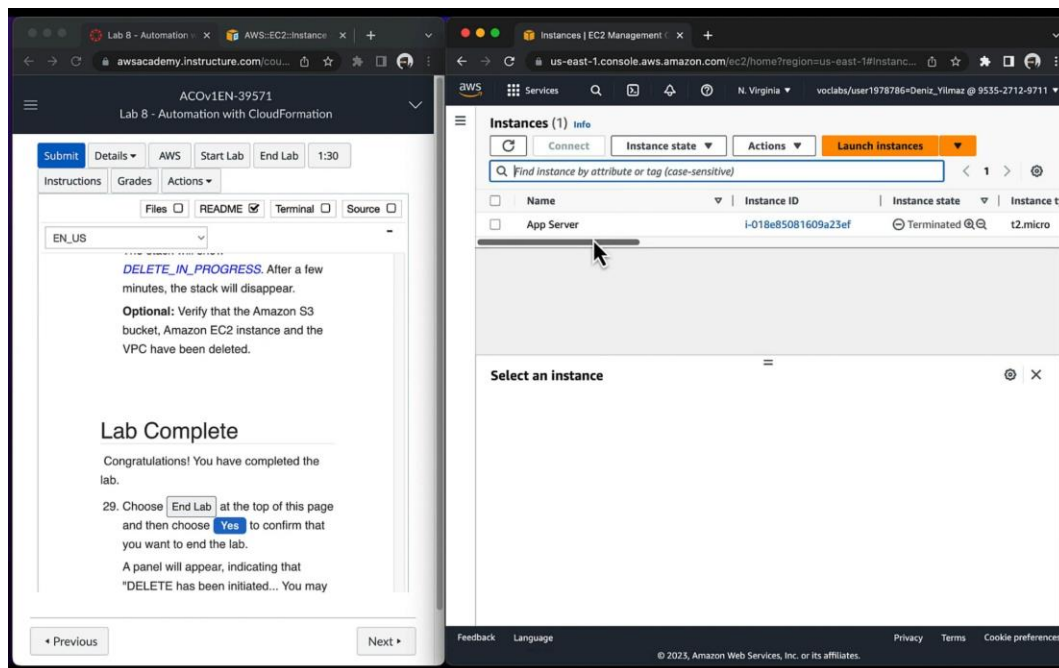
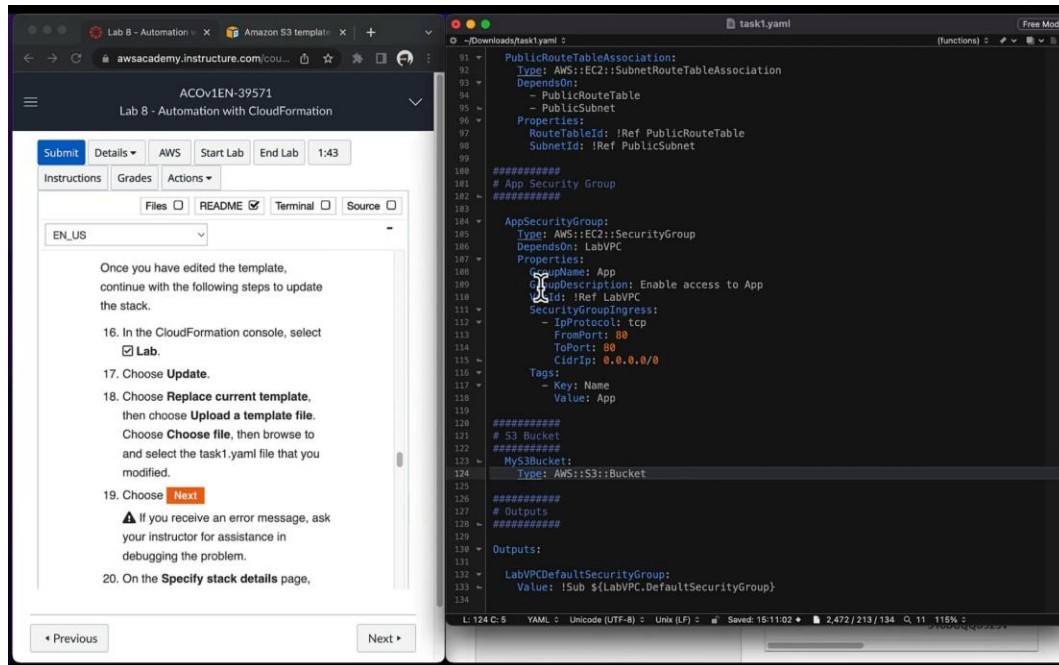
Debugging Errors: Identifying and resolving errors in templates can be time-consuming.

Dependency Management: Managing dependencies between resources can be complex.

IAM Permissions: Ensuring CloudFormation has necessary permissions can be complex.

## Screenshot:





**Conclusion:** CloudFormation is a powerful tool for automating infrastructure provisioning and management. By understanding the basics of CloudFormation templates and leveraging best practices, you can streamline your deployment processes and improve the consistency and reliability of your infrastructure.



Aditya Tamane

**Certificate of Completion for**

AWS Academy Graduate - AWS Academy Cloud Operations

**Course hours completed**

40 hours

**Issued on**

11/07/2024

**Digital badge**

<https://www.credly.com/go/MDuZZBG1>



## Grades for aditya.tamane1@gmail.com

 Print Grades (javascript:window.print())

Course

Arrange By

AWS Academy Cloud Ope ▾

Due Date ▾

Apply

Name	Due	Submitted	Status	Score
<a href="#">Module 1 Knowledge Check</a> Knowledge Checks		Jul 29 at 9:36am		80 / 100 
<a href="#">Module 2 Knowledge Check</a> Knowledge Checks		Nov 7 at 2:39pm		90 / 100 
<a href="#">Module 3 Knowledge Check</a> Knowledge Checks		Nov 7 at 2:49pm		100 / 100 
<a href="#">Module 4 Knowledge Check</a> Knowledge Checks		Nov 7 at 3pm		80 / 100 
<a href="#">Module 5 Knowledge Check</a> Knowledge Checks		Nov 7 at 3:26pm		70 / 100 
<a href="#">Module 6 Knowledge Check</a> Knowledge Checks		Nov 7 at 3:37pm		80 / 100 
<a href="#">Module 7 Knowledge Check</a> Knowledge Checks		Nov 7 at 3:49pm		80 / 100 
<a href="#">Module 8 Knowledge Check</a> Knowledge Checks		Nov 7 at 3:51pm		100 / 100 
<a href="#">Module 9 Knowledge Check</a> Knowledge Checks		Nov 7 at 3:53pm		100 / 100 
<a href="#">Module 10 Knowledge Check</a> Knowledge Checks		Nov 7 at 3:55pm		100 / 100 
<a href="#">Module 11 Knowledge Check</a> Knowledge Checks		Nov 7 at 4:02pm		80 / 100 



Name	Due	Submitted	Status	Score
<a href="#">Practice Exam</a>				- / 100
Exam Questions				
Assignments			N/A	0.00 / 0.00
Knowledge Checks			87.27%	960.00 / 1,100.00
Exam Questions			N/A	0.00 / 0.00
Total			87.27%	960.00 / 1,100.00

