AWS Academy Cloud Operations [107810]

Title Page:

- Report Title: AWS Academy Cloud Foundations Report
- Name: Rohith Yellambalase Vijayakumar
- Course: AWS Academy Cloud Foundations
- Date: Mar 1, 2025
  GWID: G26286080

Table of Contents:

Introduction:
Abstract

This report documents the progress made in the AWS Academy Cloud Operations course, specifically covering Modules 1 through 5. It details the completion of guided labs, activities, and knowledge checks, highlighting key concepts learned and challenges encountered. The report adheres to APA 7th edition formatting guidelines, including proper citation and figure presentation. The focus is on demonstrating understanding of fundamental cloud operations principles within the AWS ecosystem, including EC2, VPC, CloudWatch, and Fargate.

Module 1: Understanding Systems Operations on AWS

1. **Summary of Lessons Learned**

After going through this module, I now understand:

- **AWS CLI Functionality:** I learned the AWS CLI is a command-line tool allowing interaction with AWS services, vital for automation and scripting.
- **EC2-Based CLI Installation:** I practiced installing the AWS CLI on an EC2 Red Hat Linux instance, mimicking real-world cloud-based management.
- **CLI Configuration:** I understood how to configure the AWS CLI with access keys, enabling secure connection to an AWS account.
- **IAM Interaction via CLI:** I gained experience using the AWS CLI to query IAM details, demonstrating practical usage for identity management.

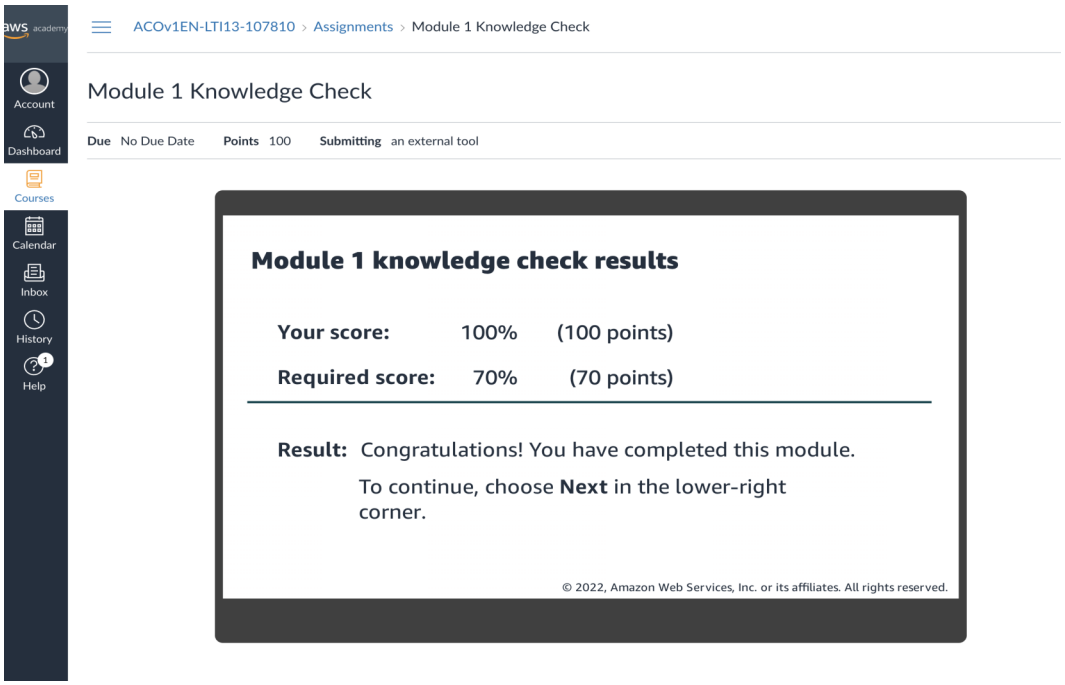
2. **Key Learnings**

Some of the important takeaways for me from this module are:

- **Remote CLI Management:** I learned that managing the AWS CLI from an EC2 instance standardizes environments and enhances remote administration.
- **Secure Credential Management:** I understood the importance of securely configuring access keys for CLI access, vital for AWS account security.
- **Practical IAM Querying:** I gained firsthand experience using the CLI to interact with IAM, enabling efficient retrieval of user and role information.
- **Business Case Application:** I applied the learning to a business scenario simulating Sophie's role, demonstrating the CLI's real-world relevance for tasks like Mom & Pop Café AWS account management.

Knowledge Check Completion 1

Fig. 1. Successful completion of knowledge test for Module 1



Note: AWS Academy Cloud Foundations Module 1 Knowledge Check passed with a perfect score of 100%.

Module 1: Activity 1 - Install and Configure the AWS CLI

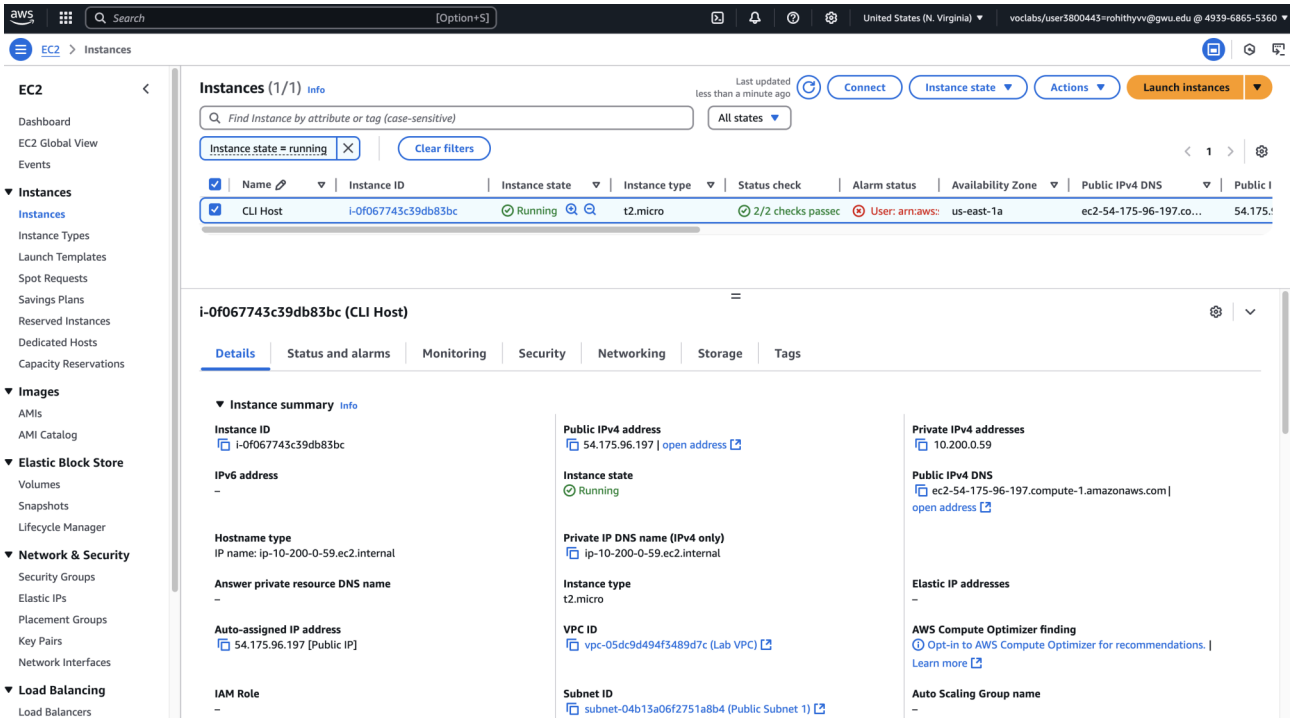
Summary of Lessons Learned

This activity provided hands-on experience in installing and configuring the AWS CLI on an EC2 instance. I learned to establish an SSH connection to the instance, install the AWS CLI, and configure it with an access key to connect to an AWS account. The activity emphasized the importance of the AWS CLI as a tool for interacting with AWS services and automating tasks.

Key Learnings

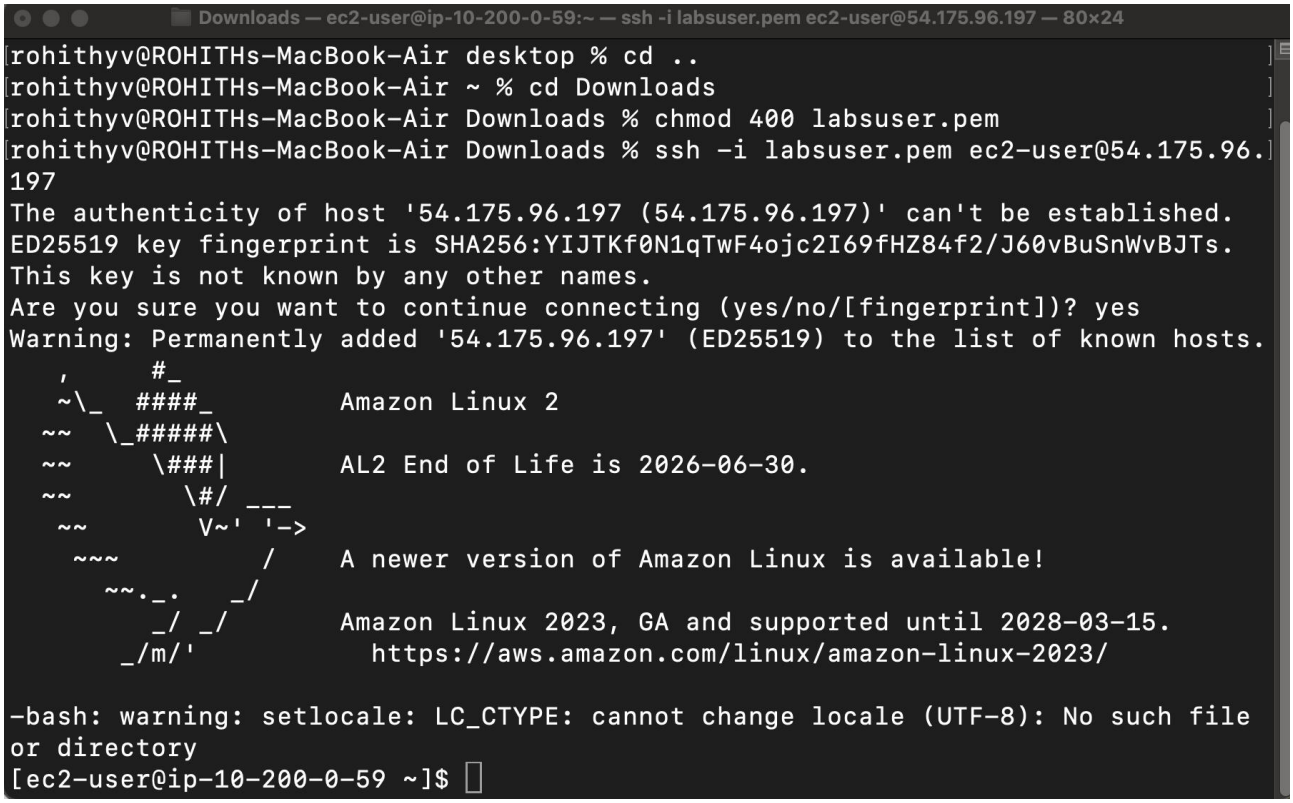
- **AWS CLI Installation:** I gained practical experience in installing the AWS CLI on a Red Hat Linux instance, following the installation instructions and verifying the installation.
- **SSH Connection:** I learned to establish an SSH connection to an EC2 instance, enabling remote access and command-line interaction.
- **AWS CLI Configuration:** I understood how to configure the AWS CLI with an access key, connecting it to an AWS account for secure access to services.
- **IAM Interaction:** I practiced using the AWS CLI to query IAM details, demonstrating its practical application for managing users and roles.
- **Real-World Relevance:** The activity simulated a real-world scenario of setting up AWS CLI access for the Mom & Pop Café, highlighting the importance of this tool for managing AWS resources in a business context.

Fig. 1.1 Activity 1: AWS EC2 Instance Details



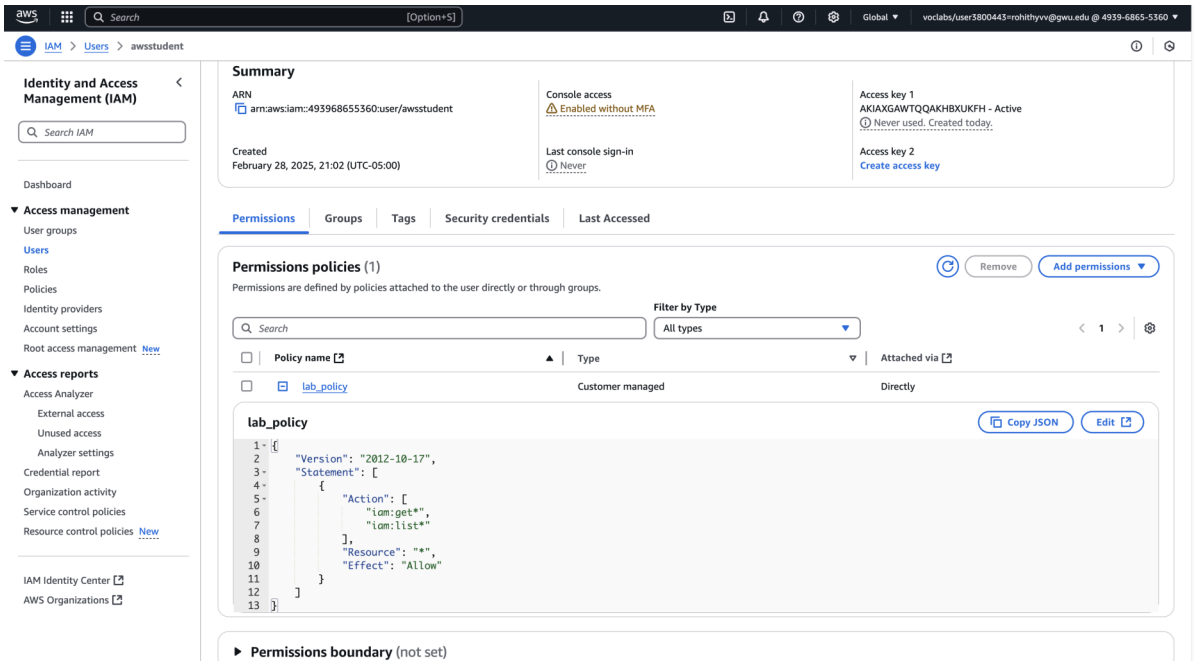
Note: Screenshot of an AWS EC2 instance dashboard showing a running t2.micro instance with its public IP, DNS, and other key details.

Fig. 1.2 Activity 1: AWS EC2 Instance Details



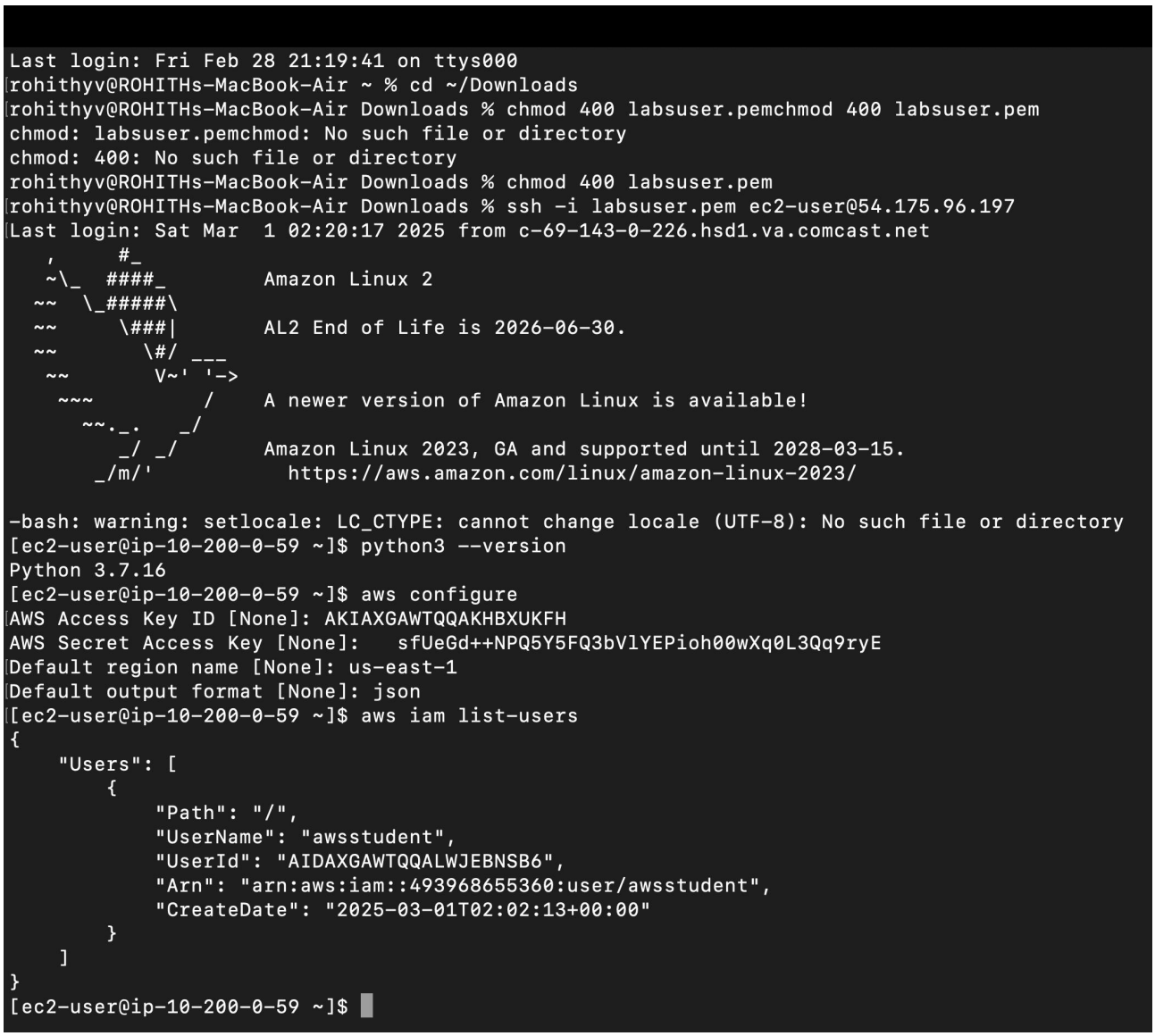
Note: A user connects to an Amazon Linux 2 EC2 instance via SSH, accepting the host key and receiving a warning about the locale setting.

Fig. 1.3 AWS IAM User Permissions



Note: Screenshot of AWS IAM user "awsstudent" details, showing an attached custom policy "lab_policy" with specific permissions defined in JSON format.

Fig. 1.4 SSH Connection, AWS Configuration, and IAM User Listing on EC2



Note: An SSH session to an EC2 instance showing AWS configuration, IAM user listing ("awsstudent"), locale warning, and Python version.

Module 2: Tooling and Automation

1. **Summary of Lesson Learned**

This module provided me a practical experience in automating AWS infrastructure through scripting and tooling, emphasizing efficiency and consistency. I learned to leverage tools for infrastructure as code, streamlining deployment and management processes.

2. **Key Learnings :**
   Some key takeaways from this module include:

- **Centralized Resource Management:** I learned how AWS Systems Manager provides a unified interface for managing EC2 instances and other resources at scale, simplifying operational tasks.
- **Inventory and Compliance:** I gained practical experience using Systems Manager Inventory to verify instance configurations and permissions, ensuring compliance and consistency.
- **Remote Task Execution:** I understood how Systems Manager Run Command enables remote execution of tasks on multiple servers, automating routine operations and maintenance.
- **Parameterized Configuration Management:** I learned to use Systems Manager Parameter Store to securely store and manage application settings and configurations, facilitating dynamic updates.
- **Secure Instance Access:** I experienced using Systems Manager Session Manager for secure, browser-based command-line access to instances, eliminating the need for SSH keys.

Knowledge Check Completion 2

Fig. 2. Successful completion of knowledge test for Module 2

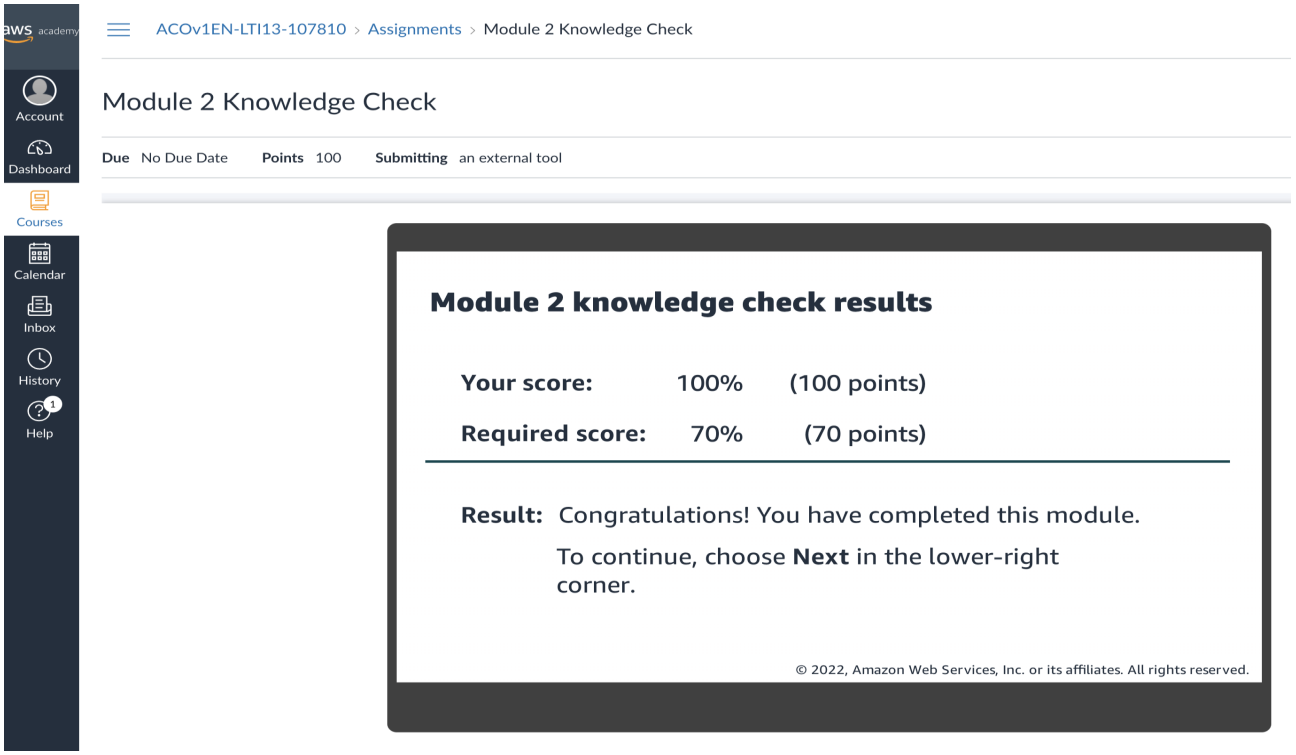
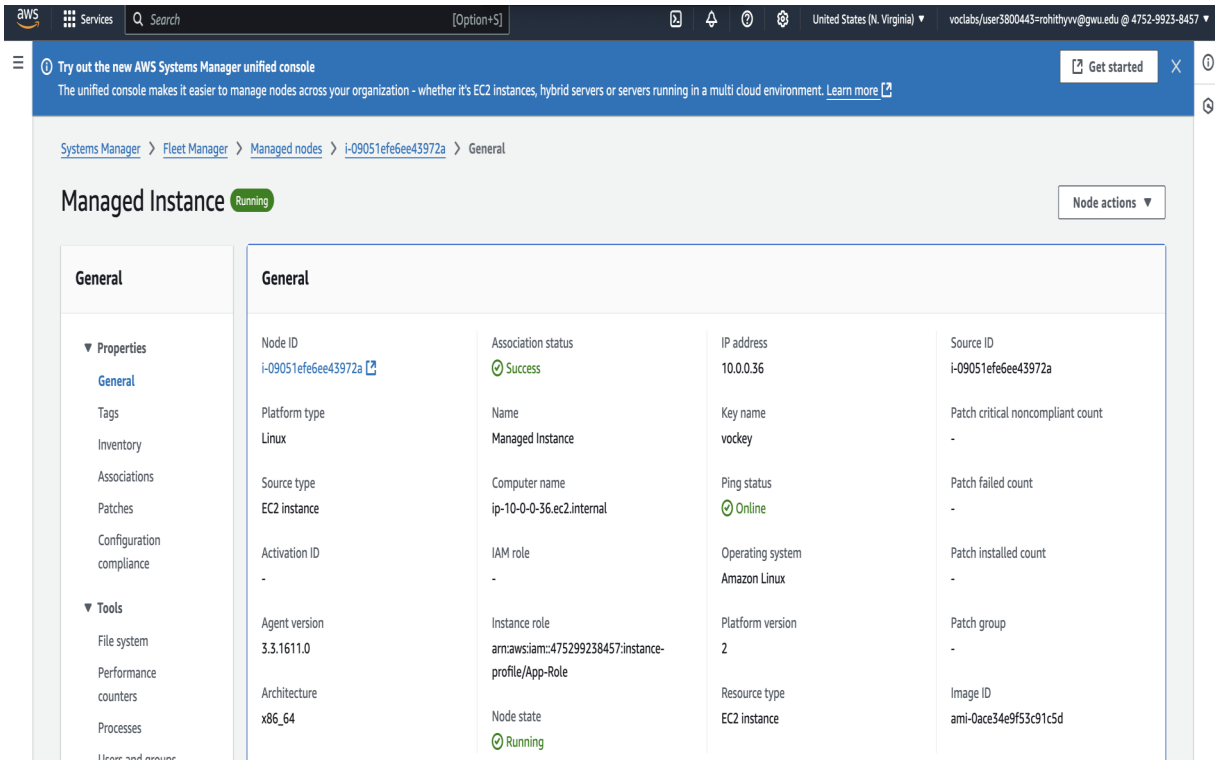
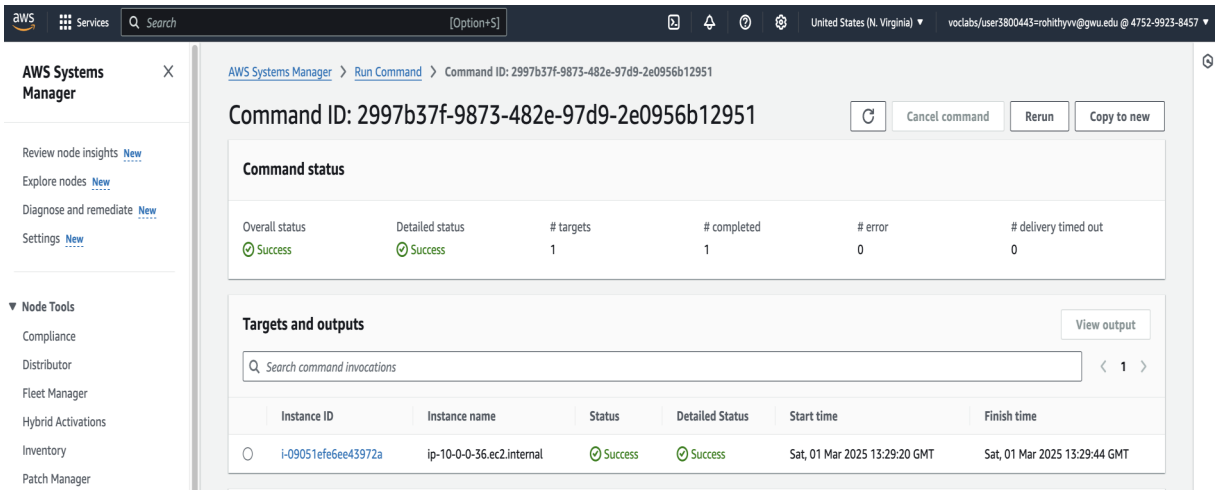


Fig 2.1 **Lab 1 - Using AWS Systems Manager**



Note: This the AWS Systems Manager Fleet Manager console, displaying details of a managed EC2 instance. The General tab shows key information like instance ID, IP address, platform, and IAM role. The instance is running Amazon Linux and is accessible via SSH using the "vockey" key pair.

Fig. 2.2 AWS Systems Manager Run Command - Successful Execution



Note: Created the AWS Systems Manager Run Command execution. Command ID is displayed, with a successful status for one target instance.

Module 2: Activity 2 - Create a Website on S3

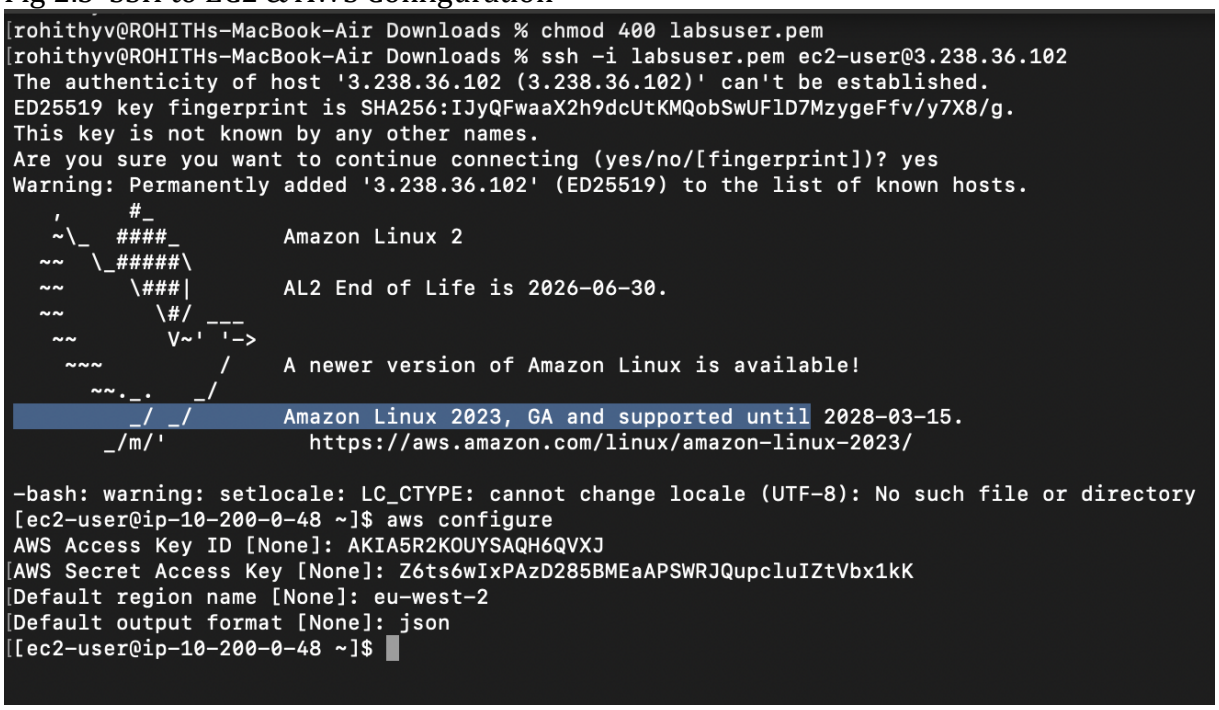
Summary of Lessons Learned

This activity provided hands-on experience in using the AWS CLI for managing S3 buckets and IAM users, culminating in the deployment of a static website. I learned to create and configure S3 buckets, upload website files, and manage access permissions using IAM policies. Additionally, I gained experience in creating a batch script to automate website updates, highlighting the efficiency of using the AWS CLI for managing cloud resources.

Key Learnings

- **S3 Bucket Creation and Configuration:** I learned to create S3 buckets using the AWS CLI, configuring them for website hosting by enabling static website hosting and setting index and error documents.
- **IAM User Observation:** I gained insights into IAM user configuration by observing the details of a user with full S3 access, understanding the permissions and policies associated with managing S3 resources.
- **Website File Upload:** I practiced uploading website files to an S3 bucket using the AWS CLI, learning how to structure and organize files for proper website deployment.
- **Batch Scripting for Automation:** I learned to create a batch script that uses the AWS CLI to copy files from a local directory to an S3 bucket, automating the process of updating the static website.
- **Website Deployment and Access:** I successfully deployed a static website to an S3 bucket, making it accessible to clients via a public URL, demonstrating the practical application of S3 for website hosting.

Fig 2.3 SSH to EC2 & AWS Configuration



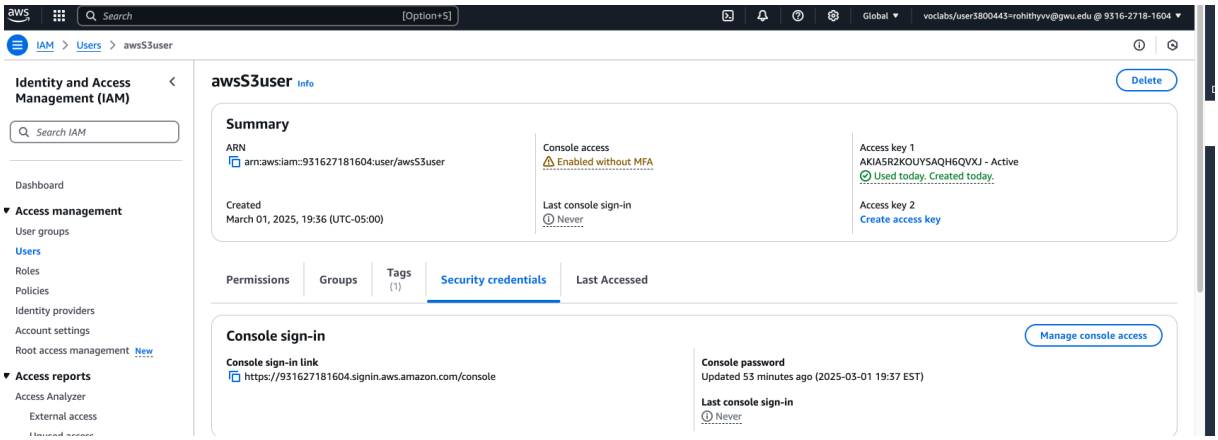
Note: Note: Terminal shows SSH connection to an EC2 instance, accepting the host key, and configuring AWS credentials with access keys and region.

Fig 2.4 EC2 Setup and S3 Bucket Creation using AWS CLI

```
ED25519 key fingerprint is SHA256:IJyQFwaaX2h9dcUtKMQobSwUFlD7MzygeFfv/y7X8/g.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.238.36.102' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory
[ec2-user@ip-10-200-0-48 ~]$ aws configure
AWS Access Key ID [None]: AKIA5R2KOUYSAQH6QVXJ
AWS Secret Access Key [None]: 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
Default region name [None]: eu-west-2
Default output format [None]: json
[ec2-user@ip-10-200-0-48 ~]$ aws s3api create-bucket

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: --bucket

[ec2-user@ip-10-200-0-48 ~]$
[ec2-user@ip-10-200-0-48 ~]$
[ec2-user@ip-10-200-0-48 ~]$
[ec2-user@ip-10-200-0-48 ~]$
[ec2-user@ip-10-200-0-48 ~]$
[ec2-user@ip-10-200-0-48 ~]$
[ec2-user@ip-10-200-0-48 ~]$ aws s3api create-bucket --bucket rohith-yv-bucket --region us-east-1
{
    "Location": "/rohith-yv-bucket"
}
```

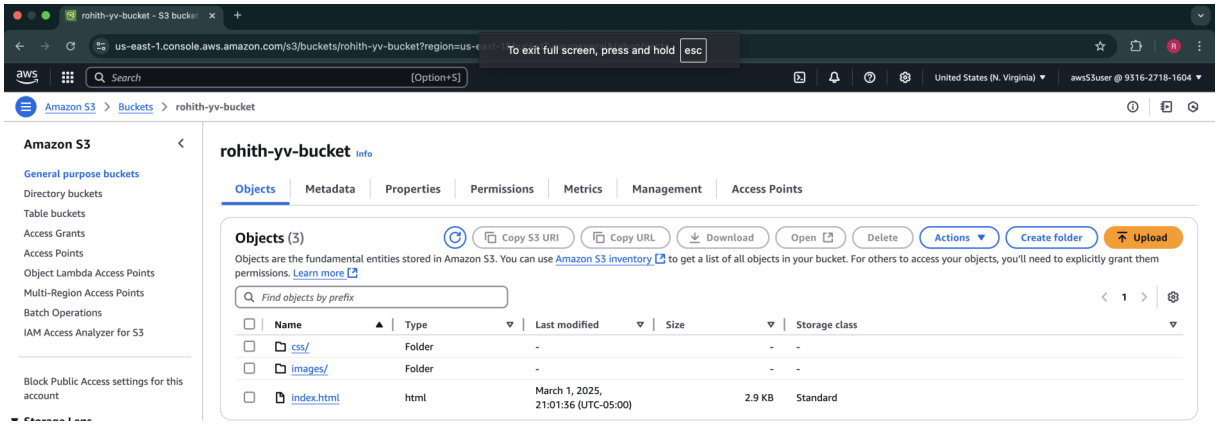
Note: User SSHs into EC2, configures AWS credentials, attempts to create an S3 bucket (initially missing the bucket name), and successfully creates it.

Fig 2.5 AWS IAM User "awsS3user" Details



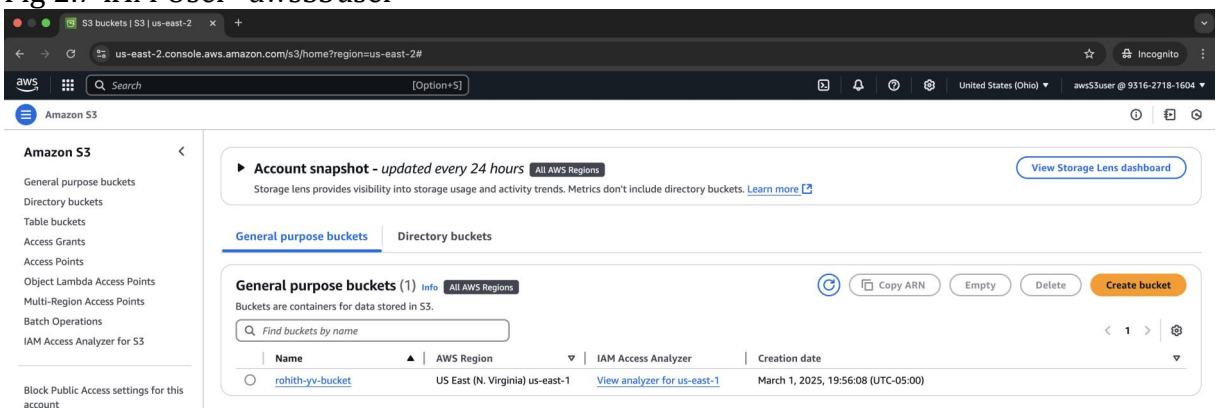
Note: Screenshot shows the AWS IAM user "awsS3user" with details including ARN, console access, access keys, and console sign-in information.

Fig 2.6 AWS S3 Bucket: rohith-yv-bucket



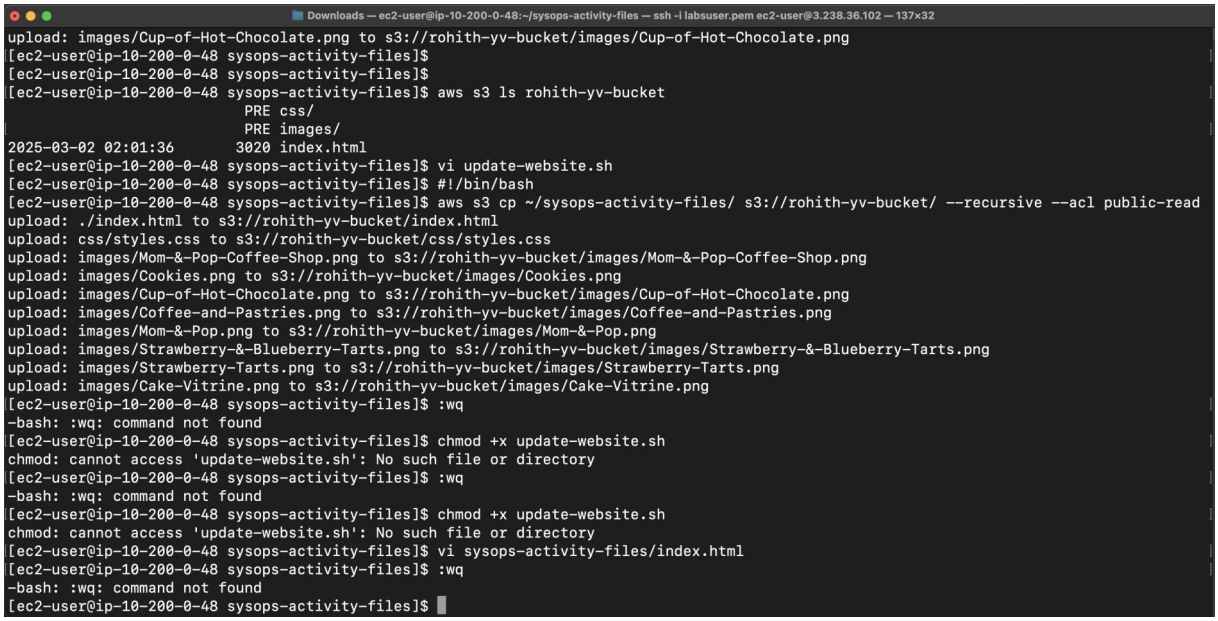
This shows the AWS S3 console displaying the "rohith-yv-bucket" contents. It lists folders and an HTML file, with options to manage bucket objects. It also shows the current user and AWS region.

Fig 2.7 IAM User "awsS3user"



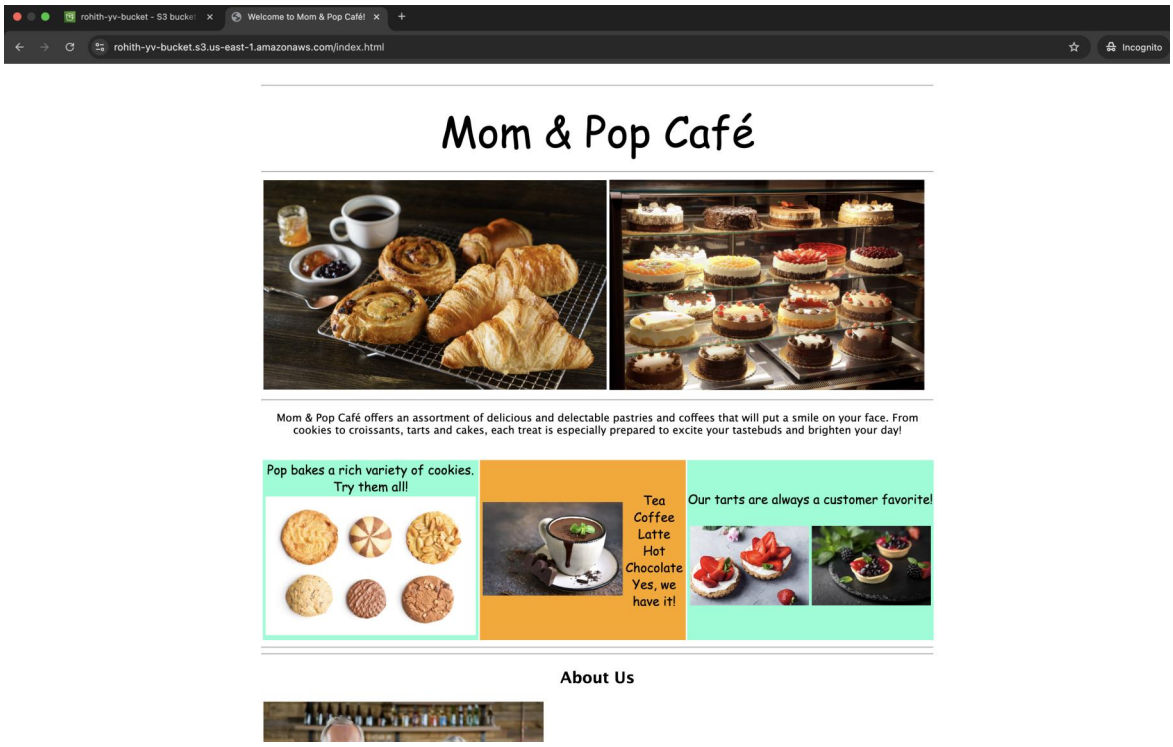
Note: Screenshot of AWS IAM user "awsS3user" showing access key, console access, and other details. Created on March 1, 2025.

Fig 2.8 S3 Upload, Script Creation & Errors



Note: User uploads files to S3, creates a script (update-website.sh), encounters :wq and chmod errors, and attempts to edit index.html again.

Fig 2.9 Mom & Pop Café Website



Note: This webpage displays the content of the Mom & Pop Café website. The website features a header with the café's name and logo, a menu section showcasing various pastries and drinks with images and descriptions, and an "About Us" section.

Module 3: Computing Servers

In Module 3, 'Computing Services' in the AWS Academy curriculum explores fundamental concepts of server management within the AWS ecosystem. Students delve into EC2 instance types, deployment strategies, and lifecycle management. Practical exercises cover instance configuration, scaling, and troubleshooting, emphasizing operational efficiency and cost optimization. The module aims to equip learners with the skills to effectively manage compute resources in a cloud environment.

1. **Summary of Lesson Learned**

In this module, I learned about the diverse range of AWS computing services, particularly EC2, enabling me to understand instance types, launch configurations, and scaling options. I learned to optimize compute resources for performance and cost efficiency, and grasp the importance of automation in server management within the AWS cloud.

2. **Key Learnings**

Some key takeaways from this module include:

- **EC2 Instance Variety:** I gained a deep understanding of the diverse EC2 instance types, their specific use cases, and how to select the optimal instance for varying workloads.
- **Launch Configurations and Templates:** I learned to create and manage launch configurations and templates, enabling consistent and repeatable instance deployments.
- **Auto Scaling and Elasticity:** I grasped the principles of Auto Scaling and how to implement it to dynamically adjust compute capacity based on demand, ensuring application availability and performance.
- **Instance Lifecycle Management:** I understood the EC2 instance lifecycle, including launching, stopping, terminating, and managing instance states effectively.
- **Cost Optimization Strategies:** I learned to implement cost optimization strategies, such as reserved instances, spot instances, and right-sizing, to minimize compute costs.

3. **Knowledge Check Module 3**

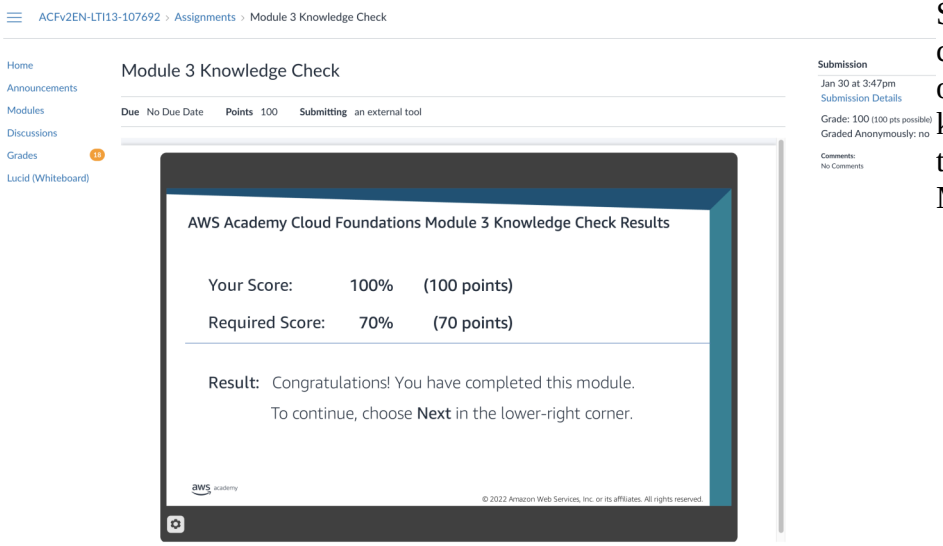


Fig. 3. Successful completion of knowledge test for Module

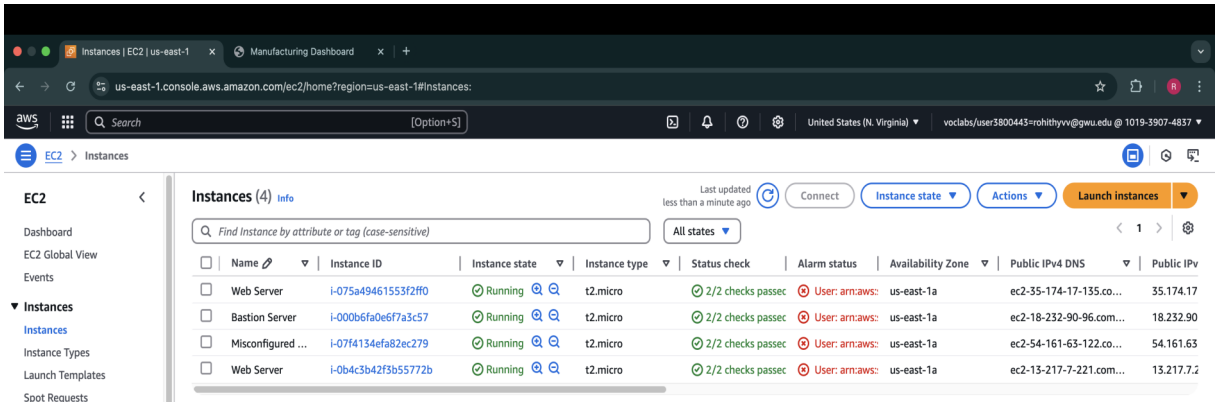
Note: AWS Academy Cloud Foundations Module 3 Knowledge Check passed with a perfect score of 100%.

Module 3: Lab 2 - Creating Amazon EC2 Instances
Summary of Lesson Learned
This lab demonstrated the efficiency of cloud-based infrastructure deployment compared to traditional methods. I learned to launch EC2 instances both through the AWS Management Console and via the AWS Command-Line Interface (CLI), gaining practical experience in automating server deployment.

Key Learnings
Console-Based EC2 Launch: I gained hands-on experience launching EC2 instances through the AWS Management Console, understanding the visual interface and configuration options.

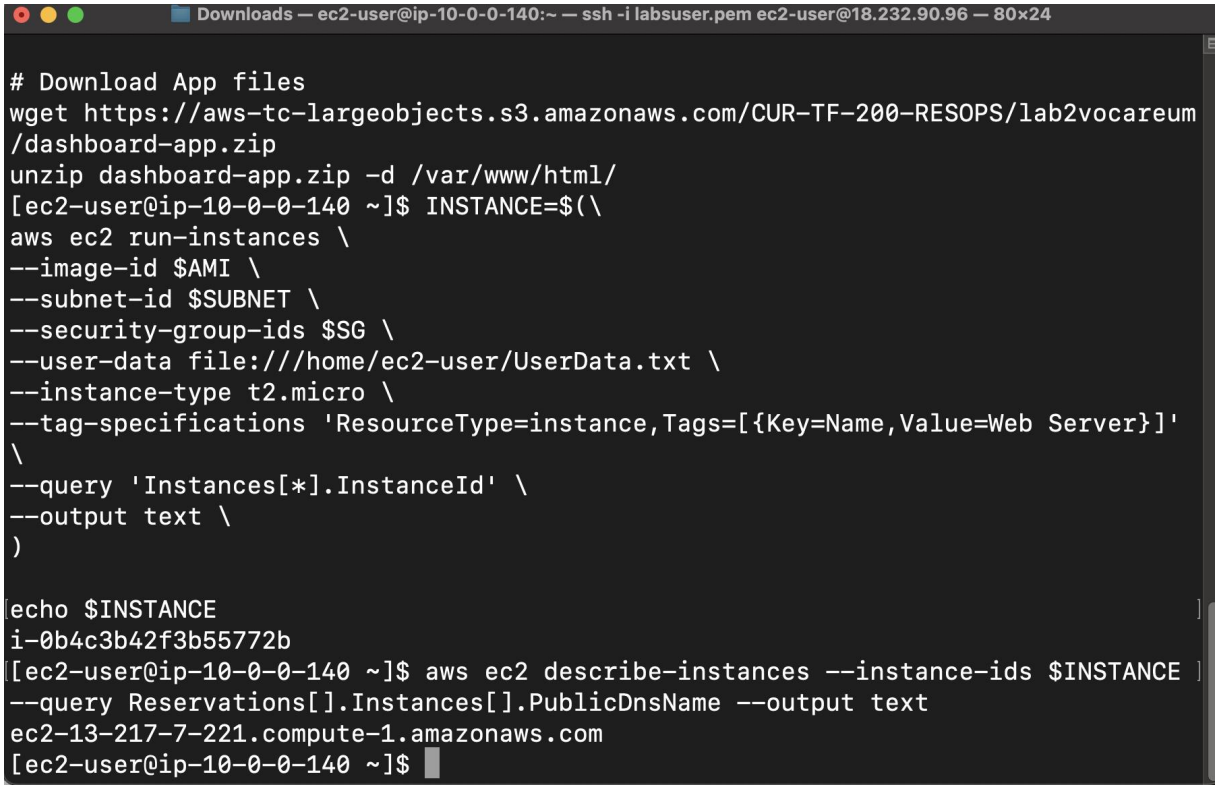
- **CLI-Based EC2 Launch:** I learned to automate EC2 instance deployment using the AWS CLI, understanding how to write and execute commands for infrastructure provisioning.
- **Automation Benefits:** I understood the speed and efficiency gains achieved by using the AWS CLI for infrastructure deployment, highlighting the value of automation in cloud operations.
- **Infrastructure as Code Fundamentals:** This lab introduced the foundational concepts of infrastructure as code through CLI usage, demonstrating how to define and deploy infrastructure programmatically.
- **Troubleshooting EC2 Instances (Optional):** I had the opportunity to practice troubleshooting EC2 instance issues, enhancing my problem-solving skills in a cloud environment.

Fig 3.1 AWS EC2 Management Console: Instances Dashboard in us-east-1 Region



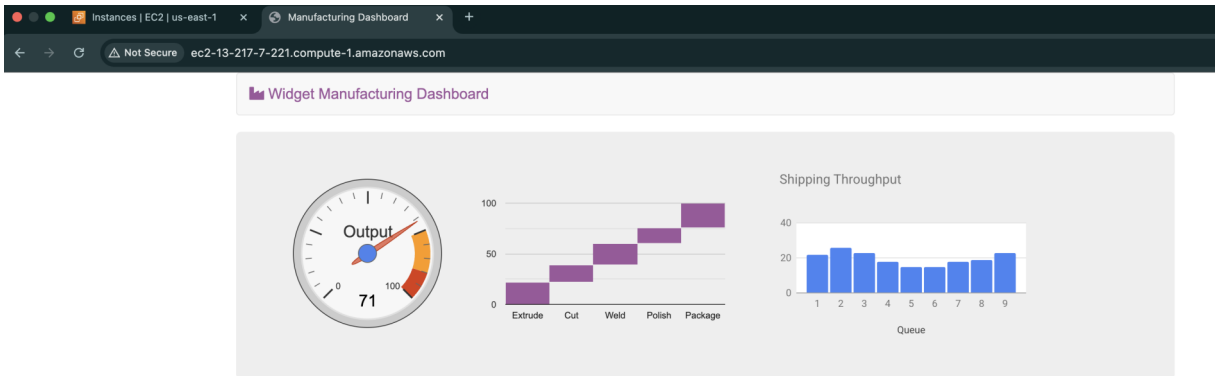
Note: This the AWS Systems Manager Fleet Manager console, displaying details of a managed EC2 instance. The General tab shows key information like instance ID, IP address, platform, and IAM role. The instance is running Amazon Linux and is accessible via SSH using the "vockey" key pair.

Fig. 3.2 EC2 Instance Deployment and Configuration via AWS CLI



Note: AWS CLI commands for launching an EC2 instance, downloading application files, and retrieving the instance's public DNS.

Fig. 3.3 Widget Manufacturing Dashboard Deployed on EC2 Instance



Note: This screenshot displays a web-based manufacturing dashboard hosted on an Amazon EC2 instance, demonstrating real-time output and shipping throughput metrics. The dashboard, accessible via the instance's public DNS, provides a visual representation of operational data, highlighting the application's functionality and performance.

Module 3: Activity 3 - Troubleshoot Creating an EC2 Instance Using the AWS CLI

Summary of Lessons Learned

This activity reinforced the practical application of the AWS CLI for deploying and configuring EC2 instances. It highlighted the importance of troubleshooting skills in resolving issues related to CLI commands and EC2 service settings, ensuring successful instance launch and application deployment.

Key Learnings

- **EC2 Launch with User Data:** I learned to launch an EC2 instance using the AWS CLI while specifying a `user-data` file for automated software installation and configuration.
- **LAMP Stack Deployment:** I gained experience in deploying a LAMP stack (Linux, Apache, MySQL, PHP) on an EC2 instance, understanding its role in web application hosting.
- **Website and Database Deployment:** I practiced deploying website files and executing SQL scripts on the instance to create a functional web application with a database backend.
- **Troubleshooting CLI Commands:** I developed skills in troubleshooting errors encountered during AWS CLI commands, ensuring accurate syntax and parameter usage.
- **Troubleshooting EC2 Service Settings:** I learned to identify and resolve issues related to EC2 service settings, such as security group configurations and instance type selection.
- **Real-World Application:** I applied these skills in a business scenario simulating the deployment of an online ordering system for the Mom & Pop Café, demonstrating the practical relevance of cloud operations.

Fig. 3.4 Activity 3: AWS EC2 Instances Dashboard: Command Host Details

```
Last login: Sat Mar  1 08:56:02 on ttys000
rohithyv@ROHITHs-MacBook-Air ~ % cd downloads
rohithyv@ROHITHs-MacBook-Air downloads % chmod 400 labsuser.pem
rohithyv@ROHITHs-MacBook-Air downloads % ssh -i labsuser.pem ec2-user@54.88.100.41
Last login: Sat Mar  1 13:58:16 2025 from c-69-143-0-226.hsd1.va.comcast.net
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2026-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory
[ec2-user@cli-host ~]$ cat create-lamp-instance-userdata.txt
cat: create-lamp-instance-userdata.txt: No such file or directory
[ec2-user@cli-host ~]$ aws configure
AWS Access Key ID [****************AIW5]:       AKIARZ6NQ33YANFLAIW5
AWS Secret Access Key [****************rSce]:   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
Default region name [us-east-1]: us-east-1
Default output format [json]: json
[ec2-user@cli-host ~]$ cd ~/sysops-activity-files/starters
[ec2-user@cli-host starters]$ cp create-lamp-instance.sh create-lamp-instance.backup
[ec2-user@cli-host starters]$ cat create-lamp-instance-userdata.txt
#!/bin/bash
yum -y update
amazon-linux-extras install -y lamp-mariadb10.2-php7.2 php7.2
yum -y install httpd mariadb-server

systemctl enable httpd
systemctl start httpd

systemctl enable mariadb
systemctl start mariadb

echo '<html><h1>Hello From Your Web Server!</h1></html>' > /var/www/html/index.html
find /var/www -type d -exec chmod 2775 {} \;
find /var/www -type f -exec chmod 0664 {} \;
echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php

usermod -a -G apache ec2-user
chown -R ec2-user:apache /var/www
chmod 2775 /var/www

#Check /var/log/cloud-init-output.log after this runs to see errors, if any.

#
# Download and unzip the Mom & Pop Cafe application files.
#

# Database scripts
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/ILT-TF-200-ACSOPS-1/activity-3/momPopDb.tar.gz
```

Note: AWS CLI Commands for Creating and Launching an EC2 Instance
This code snippet demonstrates the use of AWS CLI commands to create and launch an EC2 instance. It includes steps for configuring the instance with user data and a script to install and configure a LAMP stack (Linux, Apache, MySQL, PHP). The code also demonstrates the use of the aws ec2 run-instances command to launch the instance with specified parameters.

This exercise provided hands-on experience in using the AWS CLI to create and launch EC2 instances. I learned how to configure the instance with user data using a script, which allows for the installation and configuration of software post-launch. The use of the aws ec2 run-

instances command demonstrated the flexibility and power of the AWS CLI for automating infrastructure deployment.

Fig. 3.5 Activity 3: AWS EC2 Instances Dashboard: Command Host Details

```
# Database scripts
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/ILT-TF-200-ACSOPS-1/activity-3/momPopDb.tar.gz
tar -zxvf momPopDb.tar.gz

# Web application files
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/ILT-TF-200-ACSOPS-1/activity-3/mompopcafe.tar.gz
tar -zxvf mompopcafe.tar.gz -C /var/www/html/

#
# Run the scripts to set the database root password, and create and populate the application database.
# Check the following logs to make sure there are no errors:
#
#       /momPopDb/set-root-password.log
#       /momPopDb/create-db.log
#
cd momPopDb
./set-root-password.sh
./create-db.sh
hostnamectl set-hostname web-server
[ec2-user@cli-host starters]$ ./create-lamp-instance.sh

Running create-instance.sh on 2025-03-01 14:11:50

Region: eu-west-2
Instance Type: t2.small
Profile: default

Looking up account values...
VPC:
Subnet Id:
Key: vockey
AMI ID: ami-0eebf19cec0b40d10

Creating a new security group...
Note: AWS CLI version 2, the latest major version of the AWS CLI, is now stable and recommended for general use. For more information, see the AWS CLI version 2 installation instructions
at: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html

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: argument —vpc-id: expected one argument
Security Group:

Opening port 22 in the new security group
Note: AWS CLI version 2, the latest major version of the AWS CLI, is now stable and recommended for general use. For more information, see the AWS CLI version 2 installation instructions
at: https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2.html

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

  aws help
```

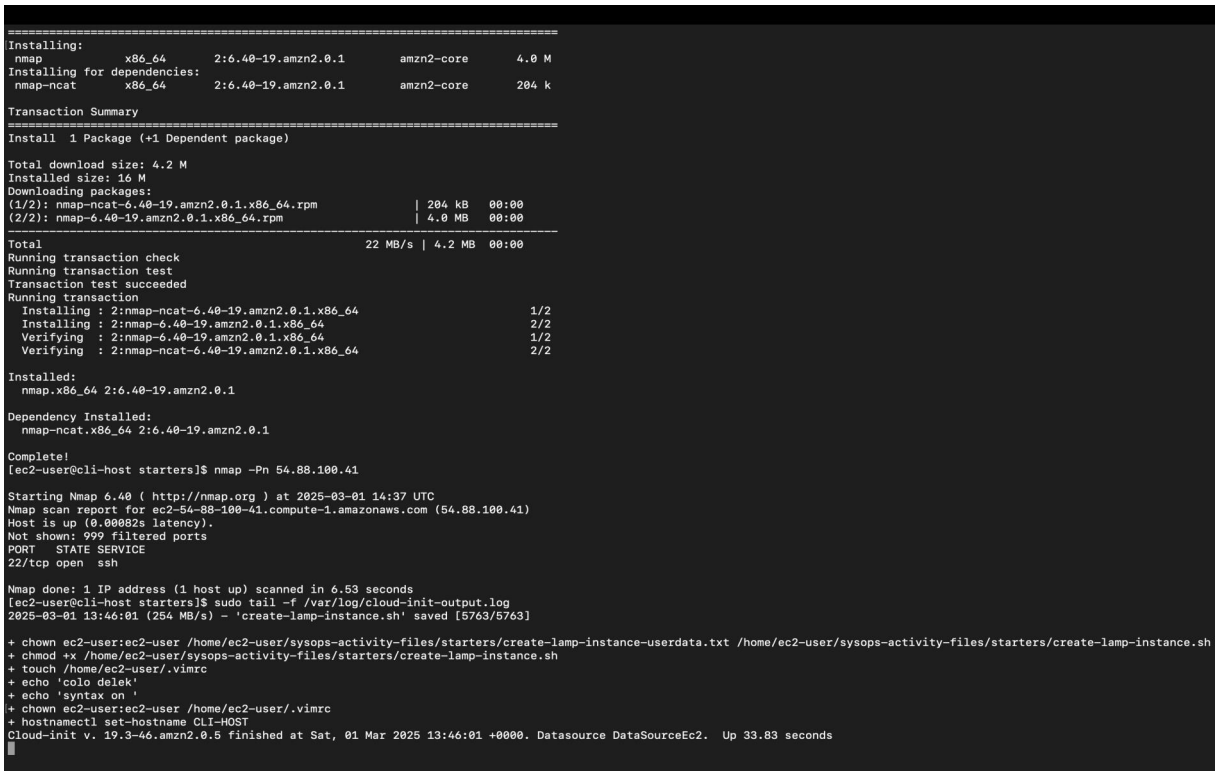
Note: This code snippet demonstrates the use of AWS CLI commands to create and launch an EC2 instance. It includes steps for configuring the instance with user data and a script to install and configure a LAMP stack (Linux, Apache, MySQL, PHP). The code also demonstrates the use of the aws ec2 run-instances command to launch the instance with specified parameters.

Fig. 3.6 Activity 3: AWS EC2 Instance Creation Script (Bash)

```bash
#!/bin/bash
DATE=`date '+%Y-%m-%d %H:%M:%S'`
echo
echo "Running create-instance.sh on "$DATE
echo

# Hard coded values
region="eu-west-2"
echo "Region: "$region
instanceType="t2.small"
echo "Instance Type: "$instanceType
profile="default"
echo "Profile: "$profile

echo
echo "Looking up account values..."

# get vpcId
vpc=$(aws ec2 describe-vpcs \
--filters "Name=tag:Name,Values='MomPopCafe VPC'" \
--region $region \
--profile $profile | grep VpcId | cut -d '"' -f4 | sed -n 1p)
echo "VPC: "$vpc

# get subnetId
subnetId=$(aws ec2 describe-subnets \
--filters "Name=tag:Name,Values='MomPopCafe Public Subnet 1'" \
--region $region \
--profile $profile \
--query "Subnets[?contains(AvailabilityZone, 'eu-west-2a')]" | grep SubnetId | cut -d '"' -f4 | sed -n 1p)
echo "Subnet Id: "$subnetId

# Get keypair name
key=$(aws ec2 describe-key-pairs \
--profile $profile | grep KeyName | cut -d '"' -f4 )
echo "Key: "$key

# Get AMI ID
imageId=$(aws ssm get-parameters \
--names '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2' \
--profile $profile \
--region $region | grep ami- | cut -d '"' -f4 | sed -n 2p)
echo "AMI ID: "$imageId

#check for existing mompopcafe instance
existingEc2Instance=$(aws ec2 describe-instances \
--region $region \
--profile $profile \
--filters "Name=tag:Name,Values=mompopcafeserver" "Name=instance-state-name,Values=running" \
| grep InstanceId | cut -d '"' -f4)
if [[ "$existingEc2Instance" != "" ]]; then
  echo
  echo "WARNING: Found existing running EC2 instance with instance ID "$existingEc2Instance"."
  echo "This script will not succeed if it already exists. "
```

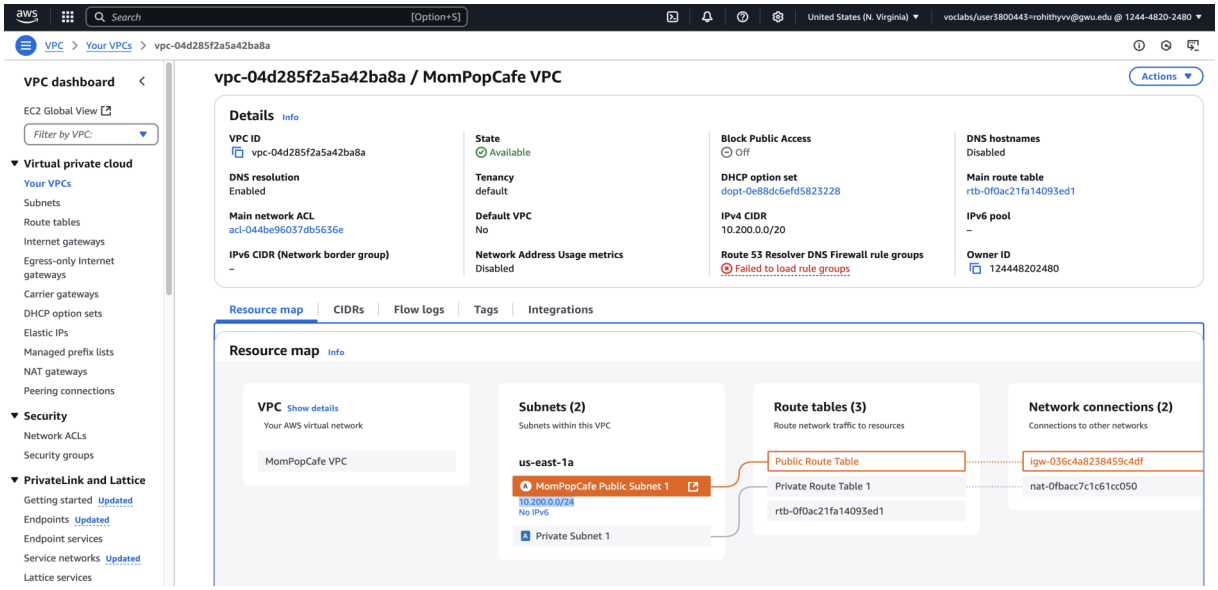
Note: This image shows a Bash script designed to create an AWS EC2 instance. It includes hardcoded values for region, instance type, and profile, and uses AWS CLI commands to look up account values like VPC ID, subnet ID, key pair name, and AMI ID. The script also checks for an existing instance with the same name before proceeding.

Fig. 3.7 Activity 3: Nmap Installation and Scan on AWS EC2 Instance



Note: The image shows the installation of the nmap and nmap-ncat packages on an Amazon Linux 2 instance. It then shows an nmap scan being run against an IP address, followed by the end of the cloud-init output log.

Fig. 3.8 Activity 3: Nmap Installation and Scan on AWS EC2 Instance



Note: This shows the AWS VPC dashboard with a resource map of the "MomPopCafe VPC," highlighting its subnets, route tables, and network connections.

Module 4: Computing, Scaling, and Name Resolution

In Module 4, 'AWS Cloud Security,' I explored advanced computing concepts, focusing on scaling techniques and name resolution within the cloud environment. I gained an understanding of load balancing, Auto Scaling groups, and Amazon Route 53 for managing traffic and ensuring application availability. The module emphasized the importance of these technologies in building scalable and resilient cloud infrastructure.

1. Summary of Lesson Learned

This module expanded my knowledge of advanced computing concepts within AWS. I learned about different scaling techniques, including load balancing and Auto Scaling groups, to manage traffic and ensure application availability. Additionally, I gained an understanding of Amazon Route 53 for reliable name resolution and DNS management.

2. Key Learnings

Some key takeaways from this module include:

- **Load Balancing:** I understood the importance of load balancers in distributing traffic across multiple EC2 instances, enhancing application performance and fault tolerance.
- **Auto Scaling Groups:** I learned to create and manage Auto Scaling groups to dynamically adjust the number of EC2 instances based on demand, ensuring scalability and cost efficiency.
- **Amazon Route 53:** I gained experience with Amazon Route 53 for configuring DNS records, routing traffic to various resources, and implementing health checks for improved availability.
- **Scaling Strategies:** I explored different scaling strategies, such as horizontal and vertical scaling, and their respective benefits in different scenarios.
- **DNS Management:** I learned about different DNS record types, their purposes, and how to manage them effectively using Route 53.

- **High Availability and Fault Tolerance:** I understood how load balancing, Auto Scaling, and Route 53 contribute to building highly available and fault-tolerant applications in the AWS cloud.

3. **Knowledge Check Completion**

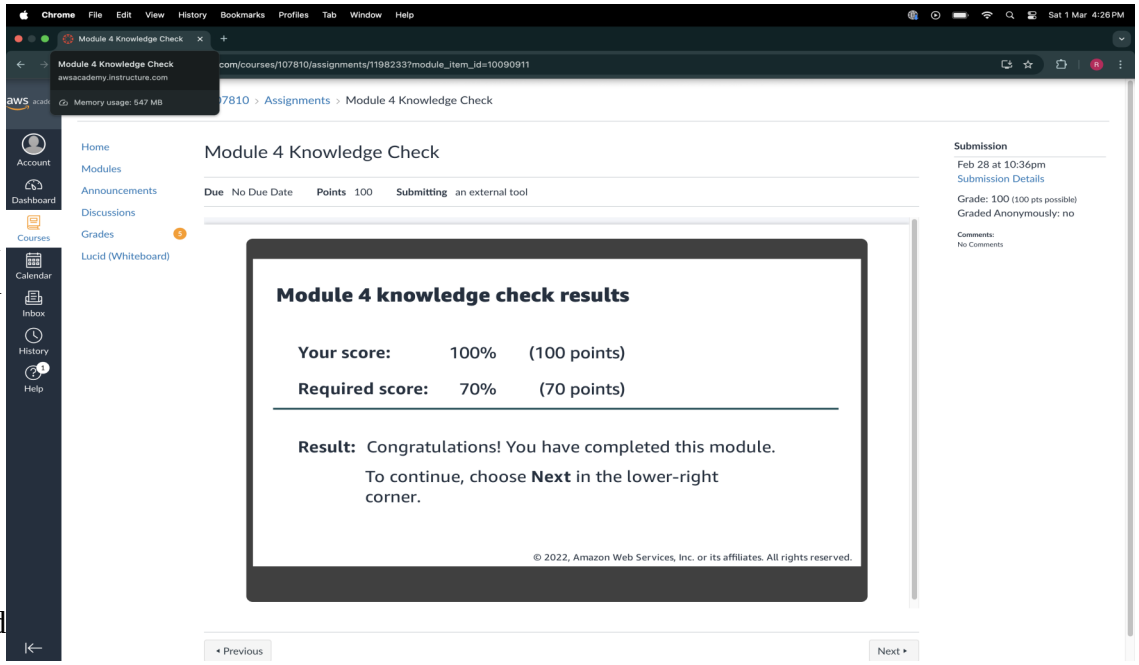


Fig. 4. Successful completion of knowledge test for Module 4

Note: AWS Academy Cloud Foundations Module 4 Knowledge Check passed with a perfect score of 100%.

Module 4: Lab 3 - Using Auto Scaling

Summary of Lessons Learned

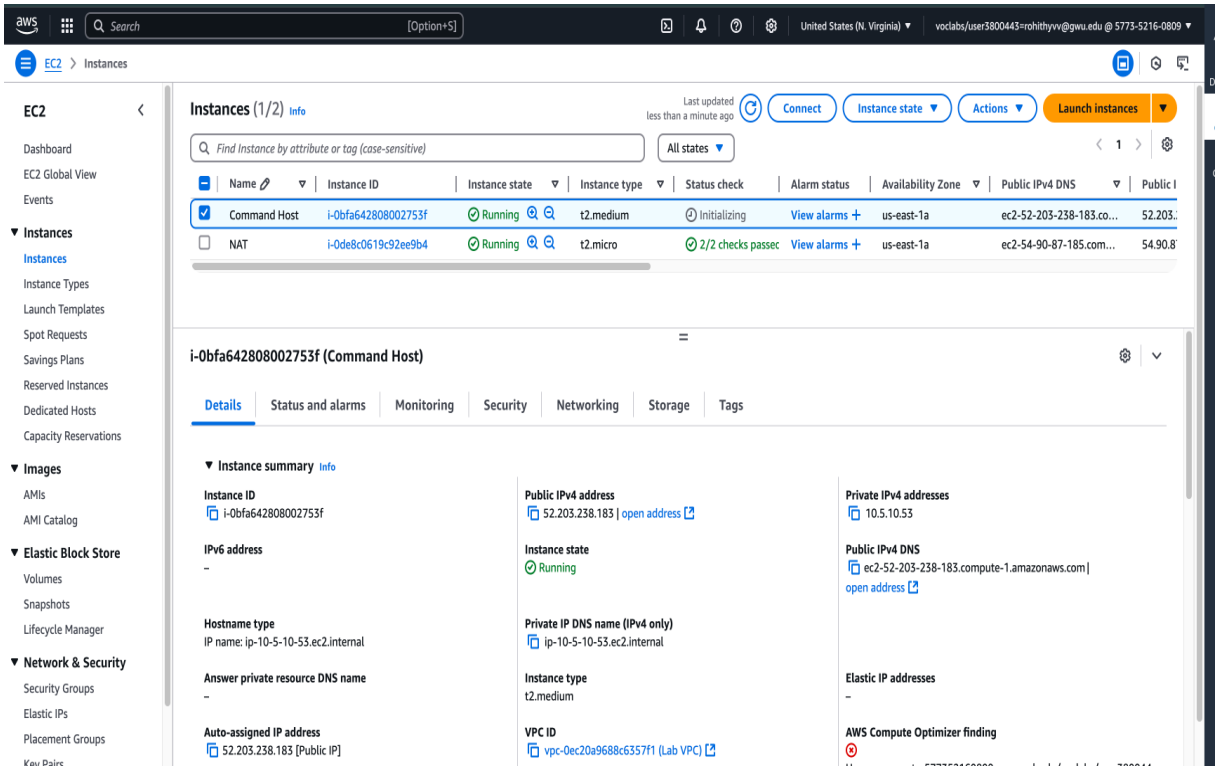
This lab provided hands-on experience in creating and managing Auto Scaling resources within AWS. I learned to create a new Amazon Machine Image (AMI) and use it to define an Auto Scaling group that automatically adjusts the number of EC2 instances based on demand. This lab reinforced the importance of Auto Scaling in ensuring application scalability and high availability.

Key Learnings

- **AMI Creation with AWS CLI:** I learned to create a new AMI from an existing EC2 instance using the AWS CLI, enabling the reuse of server configurations for Auto Scaling.

- **Auto Scaling Group Configuration:** I gained experience in configuring an Auto Scaling group, defining parameters such as desired capacity, minimum size, and maximum size to control instance scaling.
- **Launch Template Utilization:** I learned to use a launch template to specify instance configurations for the Auto Scaling group, ensuring consistency and simplifying instance deployment.
- **Auto Scaling Policies Implementation:** I practiced implementing Auto Scaling policies, such as scaling based on CPU utilization, to dynamically adjust the number of instances based on real-time demand.
- **Application Load Balancer Integration:** I learned to integrate an Application Load Balancer with the Auto Scaling group, distributing traffic across instances and ensuring high availability.
- **Scalability and High Availability:** This lab reinforced the importance of Auto Scaling in achieving application scalability and high availability, ensuring that applications can handle varying workloads and remain resilient to failures.

Fig. 4.1 Lab 3: AWS EC2 Instances Dashboard: Command Host Details



Note: This shows the AWS EC2 Instances Dashboard, highlighting the details of the "Command Host" instance in the us-east-1 region. It shows the instance's state, type, public and private IP addresses, DNS names, and VPC information, illustrating key aspects of EC2 instance management and configuration. The "Initializing" status check indicates the instance is still starting up.

From this lab, I learned how to manage AWS EC2 Instances dashboard, providing an overview of active EC2 instances, including their states, types, and network details. It highlights the "Command Host" instance, currently in the "Running" state and "Initializing," along with its public and private IP addresses and associated VPC information. The user is operating within the "us-east-1" region and has access to various EC2-related services through the navigation pane. Overall, the screenshot captures a snapshot of EC2 instance management within the AWS Management Console.

Activity 4: Configure failover routing with Amazon Route 53

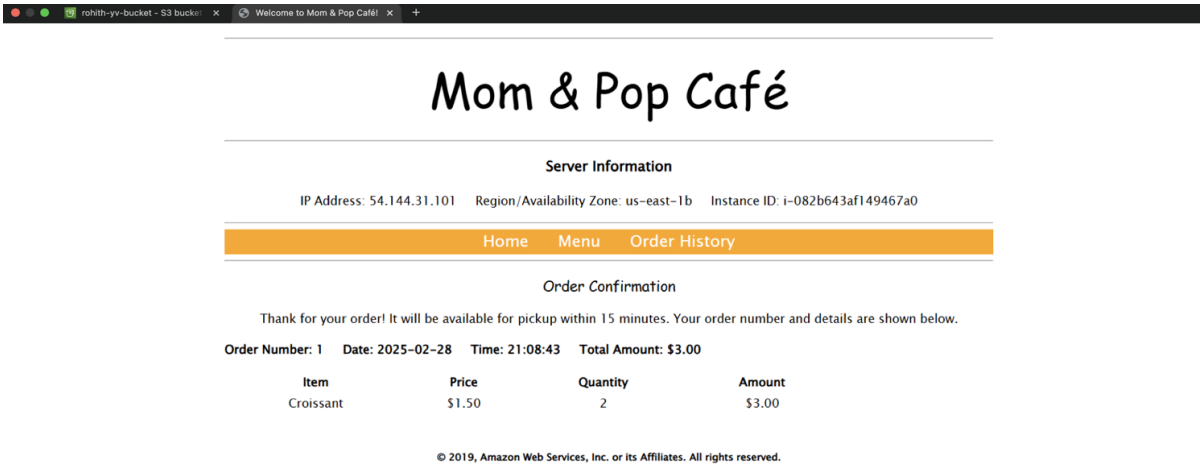
Summary of Lessons Learned

This activity provided hands-on experience in configuring failover routing using Amazon Route 53 to enhance website availability. I learned to create health checks to monitor the primary website's health and configure DNS records to automatically redirect traffic to a secondary instance in case of failure. The activity emphasized the importance of high availability and disaster recovery in maintaining business continuity.

Key Learnings

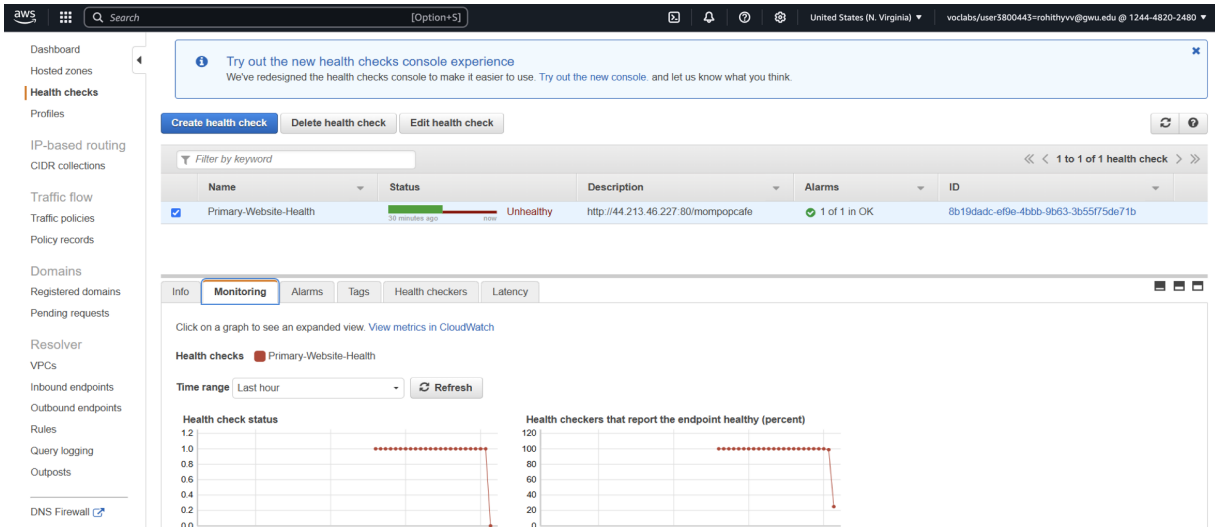
- **Route 53 Health Checks:** I gained practical experience in creating Route 53 health checks to monitor the health of an HTTP endpoint, configuring parameters like the protocol, port, and path.
- **Failover Routing Configuration:** I learned to configure failover routing in Route 53, associating the health check with a DNS record to automatically redirect traffic to a secondary instance when the primary one becomes unhealthy.
- **Email Notifications:** I configured email notifications for the health check, ensuring that administrators are alerted when the primary website's health deteriorates.
- **High Availability and Disaster Recovery:** The activity reinforced the importance of high availability and disaster recovery planning, demonstrating how to implement failover mechanisms to minimize downtime and ensure business continuity.
- **Real-World Application:** The activity simulated a real-world scenario of implementing failover routing for the Mom & Pop Café website, highlighting the practical application of Route 53 for maintaining website availability.

Fig. 4.2 Activity 4: Confirm the Websites and Localize Them



Note: The primary deployment of the Mom & Pop Café website open and refresh the page. We should see that a “Server Information” section now displays on the web page, just under the Mom & Pop Café banner, the Order Confirmation page reflects the time the order was placed, in the time zone where the web server is running.

Fig. 4.3 Activity 4: Verify the Failover Functionality



Note: The Route 53 health check you configured will notice that the application is not responding, and the record set entries you configured will cause DNS traffic to failover to the secondary EC2 instance.

Module 5: Computing, Containers, and Serverless

This module I focused on serverless computing with AWS Lambda, highlighting its benefits and practical applications. I gained an understanding of Lambda functions, their execution model, and how they interact with other AWS services. The hands-on experience reinforced my knowledge of serverless architecture and its role in building scalable and cost-effective applications.

- **Summary of Lesson Learned**

This module immersed me in the world of serverless computing with AWS Lambda. I gained hands-on experience building and deploying Lambda functions, understanding their integration with other AWS services like API Gateway and databases. The focus on practical application solidified my grasp of serverless architecture and its advantages for creating scalable and cost-efficient solutions.

2. **Key Learnings**

Some key takeaways from this module include:

- **AWS Lambda Fundamentals:** I learned about Lambda functions, their event-driven execution model, and how to configure them for various use cases.
- **Serverless Application Development:** I gained insights into developing serverless applications, including writing code for Lambda functions and integrating them with other services.
- **Lambda Layers and Dependencies:** I explored the use of Lambda layers to manage dependencies and streamline function deployments.
- **API Gateway Integration:** I learned how to use API Gateway to create RESTful APIs that trigger Lambda functions, enabling serverless backend development.
- **Database Interaction:** I understood how Lambda functions can interact with databases like MariaDB, facilitating data processing and storage in serverless applications.
- **Scalability and Cost-Efficiency:** I learned how Lambda's automatic scaling and pay-per-use model contribute to building scalable and cost-effective solutions.

3. **Knowledge Check Completion**

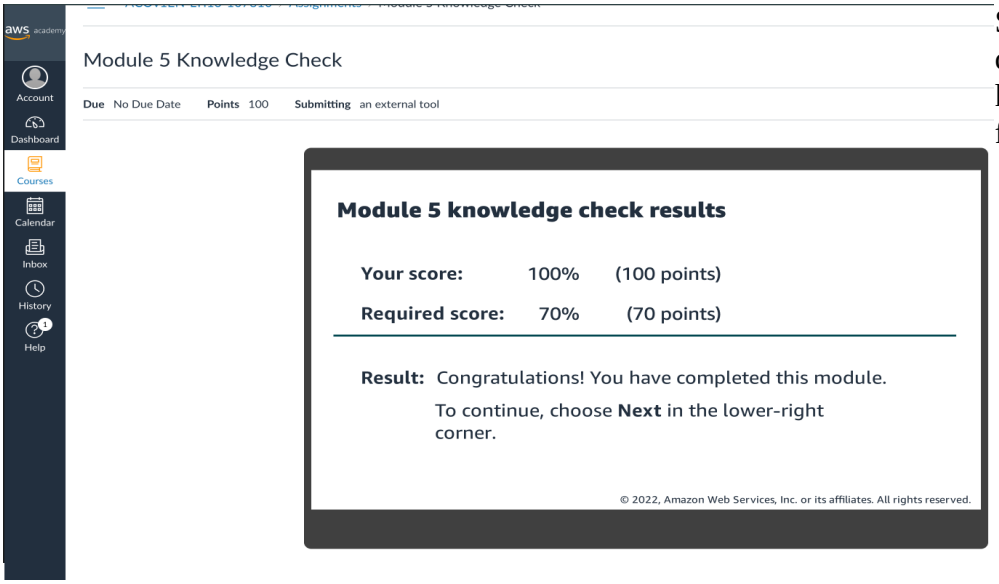


Fig. 5. Successful completion of knowledge test for Module 5

Note: AWS Academy Cloud Foundations Module 5 Knowledge Check passed with a perfect score of 100%.

Activity 5 - Working with AWS Lambda

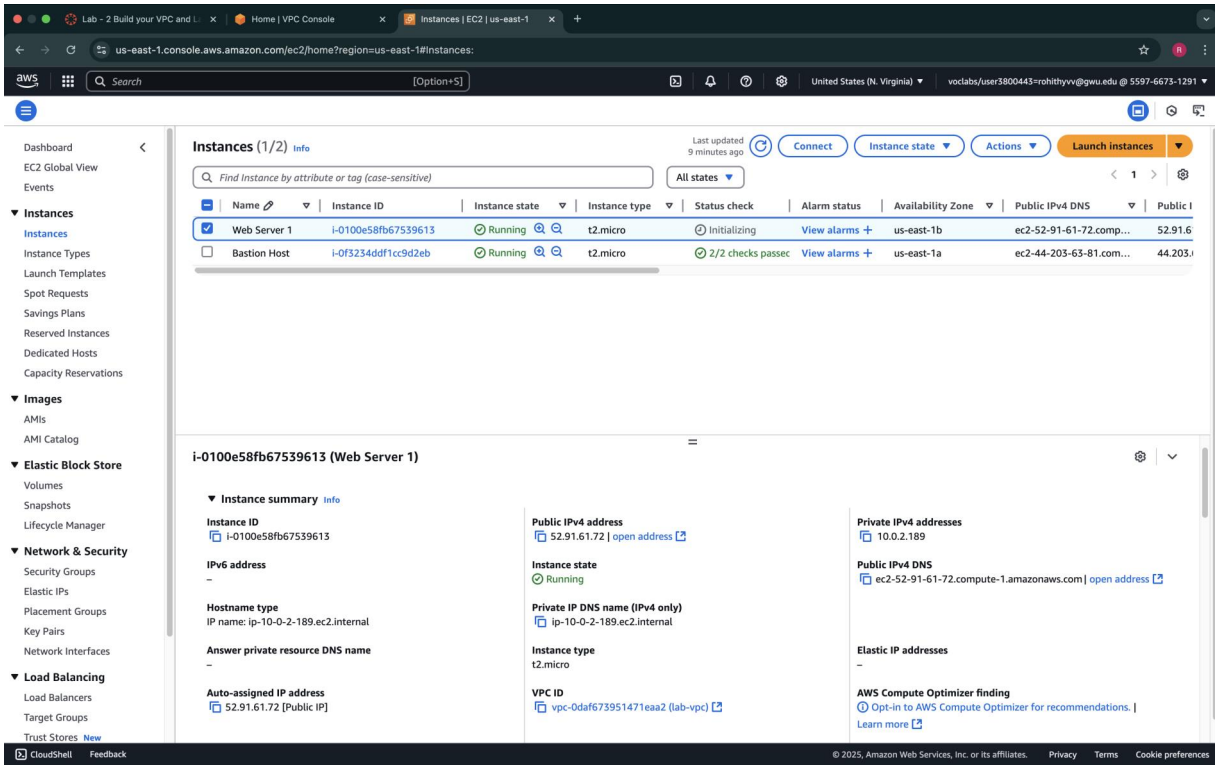
Summary of Lessons Learned

This activity provided hands-on experience in deploying and configuring a serverless computing solution using AWS Lambda. I learned to create and configure Lambda functions, manage dependencies with layers, and integrate with other AWS services like CloudWatch Events, SSM Parameter Store, and SNS for scheduling, data retrieval, and notification. The activity reinforced the practical application of Lambda functions for automating tasks and building scalable, cost-effective solutions.

Key Learnings

- **Lambda Function Deployment:** I gained practical experience in deploying Lambda functions, configuring their execution roles, and setting up environment variables.
- **Lambda Layers for Dependencies:** I learned to create and use Lambda layers to manage external library dependencies, streamlining function deployment and reducing package size.
- **CloudWatch Events for Scheduling:** I learned to use CloudWatch Events to trigger Lambda functions on a schedule, enabling automated execution of tasks.
- **SSM Parameter Store Integration:** I understood how to use SSM Parameter Store to securely store and retrieve sensitive information, such as database credentials, for use in Lambda functions.
- **SNS for Notifications:** I learned to integrate Lambda functions with SNS to send notifications, such as email reports, based on function execution results.
- **Troubleshooting with CloudWatch Logs:** I gained experience in using CloudWatch Logs to monitor Lambda function execution and troubleshoot any issues that may arise.
- **Real-World Application:** I applied these skills in a business scenario simulating the automation of a sales analysis report for the Mom & Pop Café, demonstrating the practical relevance of serverless computing.

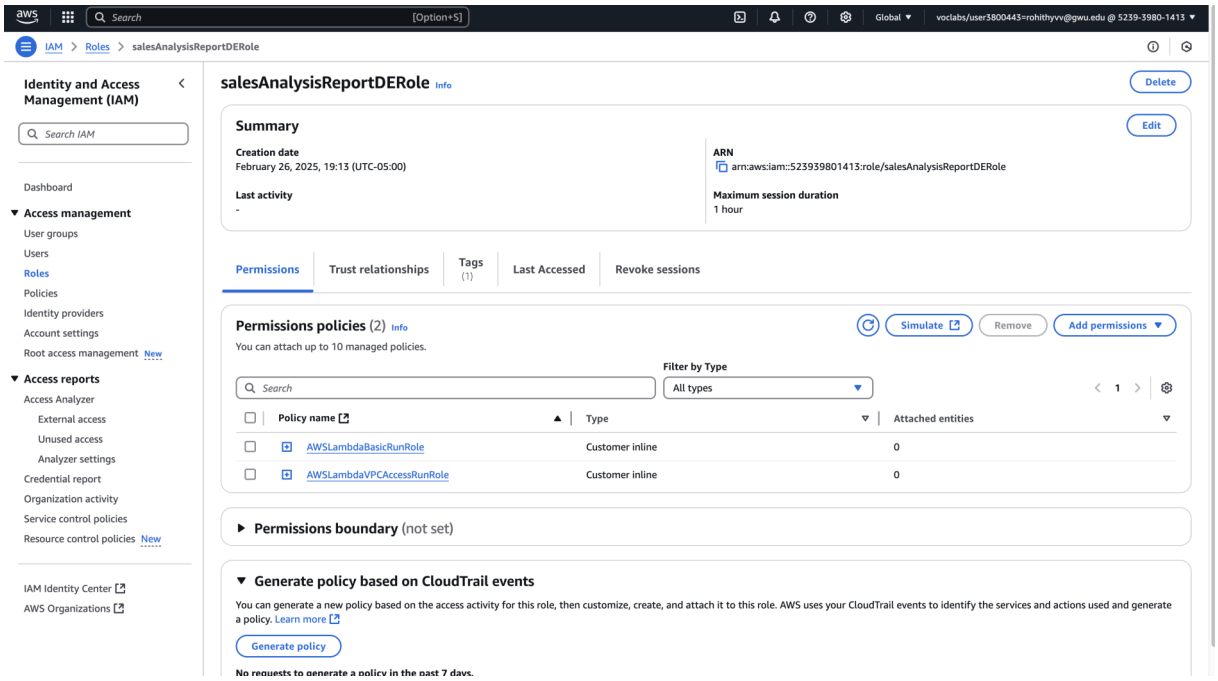
Fig. 5.1 . Activity 5 - AWS IAM Role: salesAnalysisReportDERole Trust Relationship



Note: This screenshot shows the "salesAnalysisReportDERole" IAM role configured to trust AWS Lambda. This allows Lambda functions to assume the role and access necessary resources securely, following the principle of least privilege.

In this lab, I deepened my understanding of IAM roles and trust relationships within AWS. I learned how to configure an IAM role specifically for a Lambda function, enabling it to access other AWS resources securely. The visual representation of the trust policy document, with its JSON format and explicit permission grant to the "lambda.amazonaws.com" service principal, clarified how to control which services can assume a role. This reinforced the importance of least privilege and secure configuration in serverless applications.

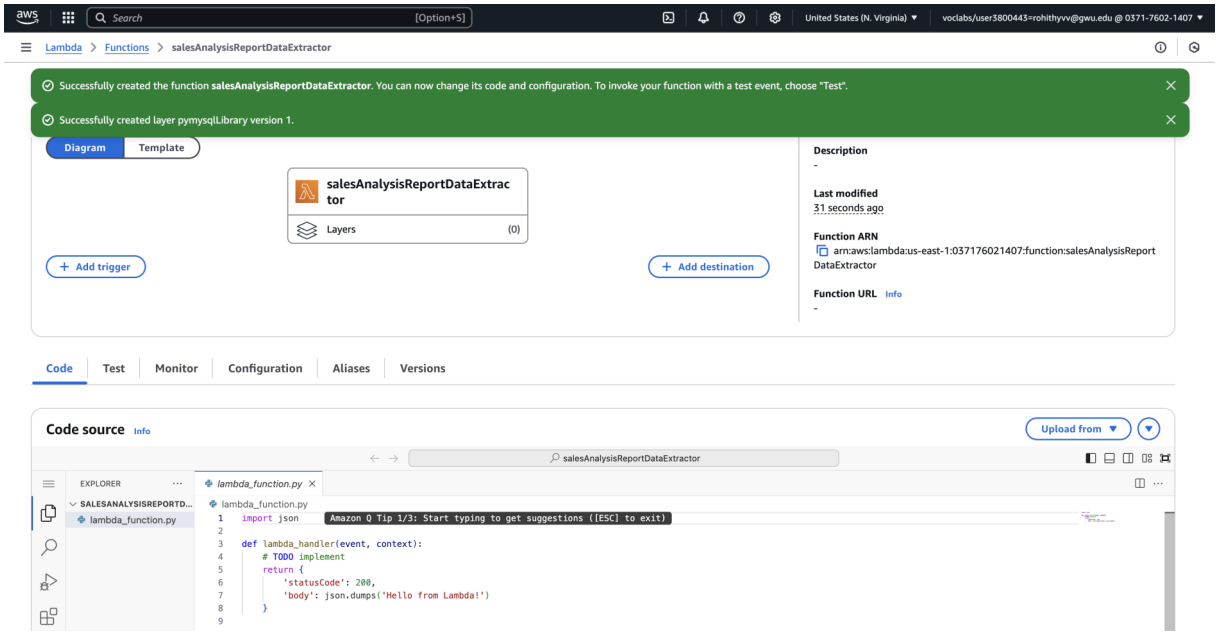
Fig. 5.2 **AWS IAM Role: salesAnalysisReportDERole Permissions Policies**



Note: This screenshot shows the permissions policies attached to the "salesAnalysisReportDERole," including "AWSLambdaBasicRunRole" and "AWSLambdaVPCAccessRunRole" for Lambda function execution and VPC access.

In this lab, I learned to manage permissions for IAM roles in AWS. Seeing the "salesAnalysisReportDERole" with its attached policies, like "AWSLambdaBasicRunRole" and "AWSLambdaVPCAccessRunRole," helped me understand how to grant specific permissions to Lambda functions. The options for simulating, adding, or removing permissions, along with generating policies and setting boundaries, showed me the flexibility and control AWS provides for managing access to resources, which is crucial for maintaining security and adhering to least privilege.

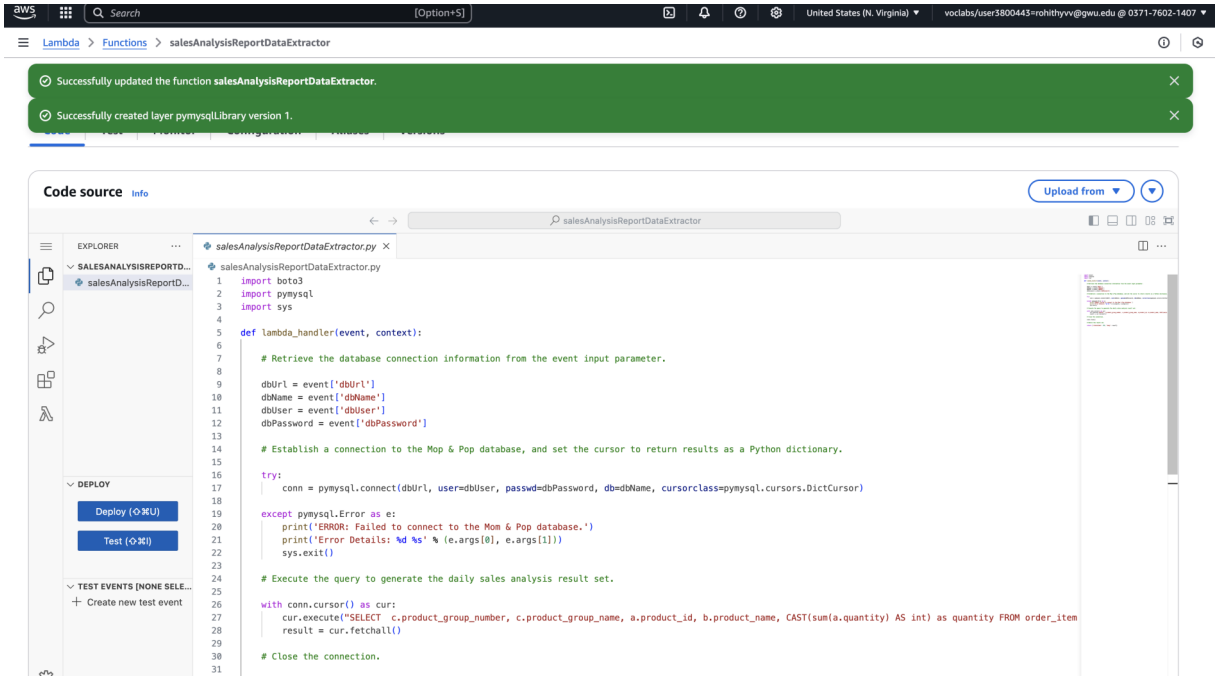
Fig. 5.3 **AWS Lambda Function: salesAnalysisReportDataExtractor - Creation and Code View**



Note: This shows the "salesAnalysisReportDataExtractor" Lambda function successfully created with the "pymysqlLibrary" layer. The function's Python code, currently a simple "Hello from Lambda!" message, is visible in the code view section.

This screenshot solidified my understanding of creating and configuring AWS Lambda functions. I learned how to navigate the Lambda console, define function settings, and add layers like the "pymysqlLibrary" for external dependencies. Seeing the Python code directly within the console helped me visualize how the function works, even with a simple "Hello from Lambda!" example. The confirmation messages reassured me that I had successfully created both the function and the layer, giving me confidence in my ability to deploy serverless functions in AWS.

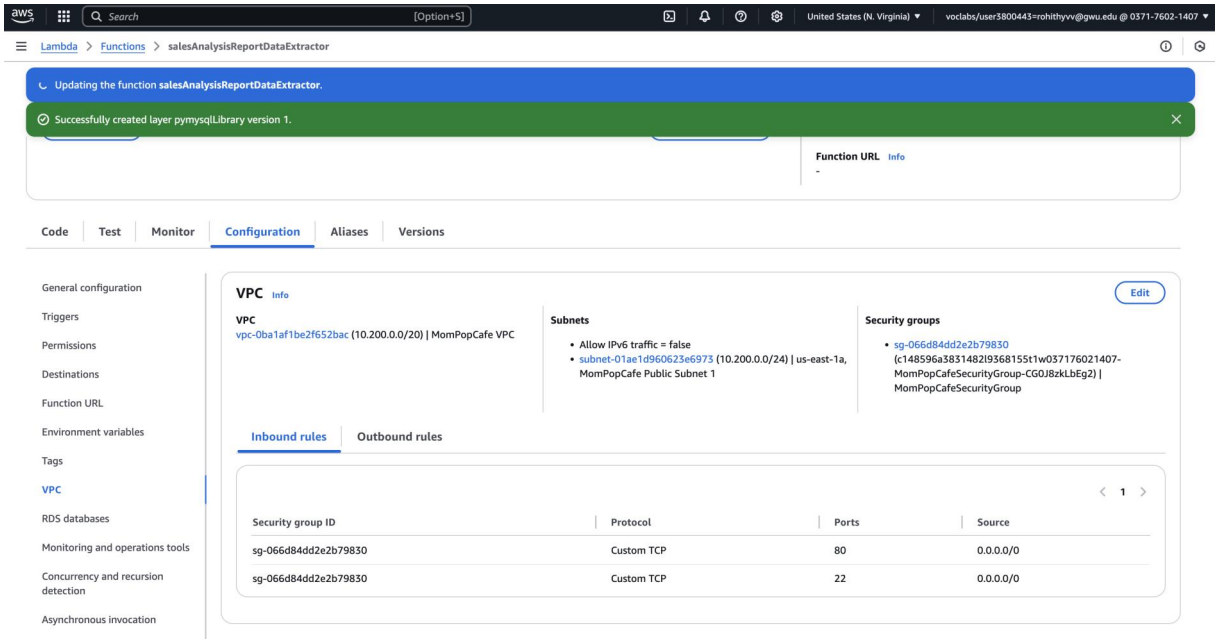
Fig. 5.4 AWS IAM Role: salesAnalysisReportDERole Configuration



Note: This screenshot displays the "salesAnalysisReportDERole" IAM role, configured to allow Lambda functions access for secure resource interaction. The trust policy documents grants "sts:AssumeRole" permission to the "" service principal.

I learned how to connect to a database, retrieve data with SQL queries, and handle potential errors within a Lambda function. This hands-on experience solidified my understanding of using Lambda for real-world data processing tasks.

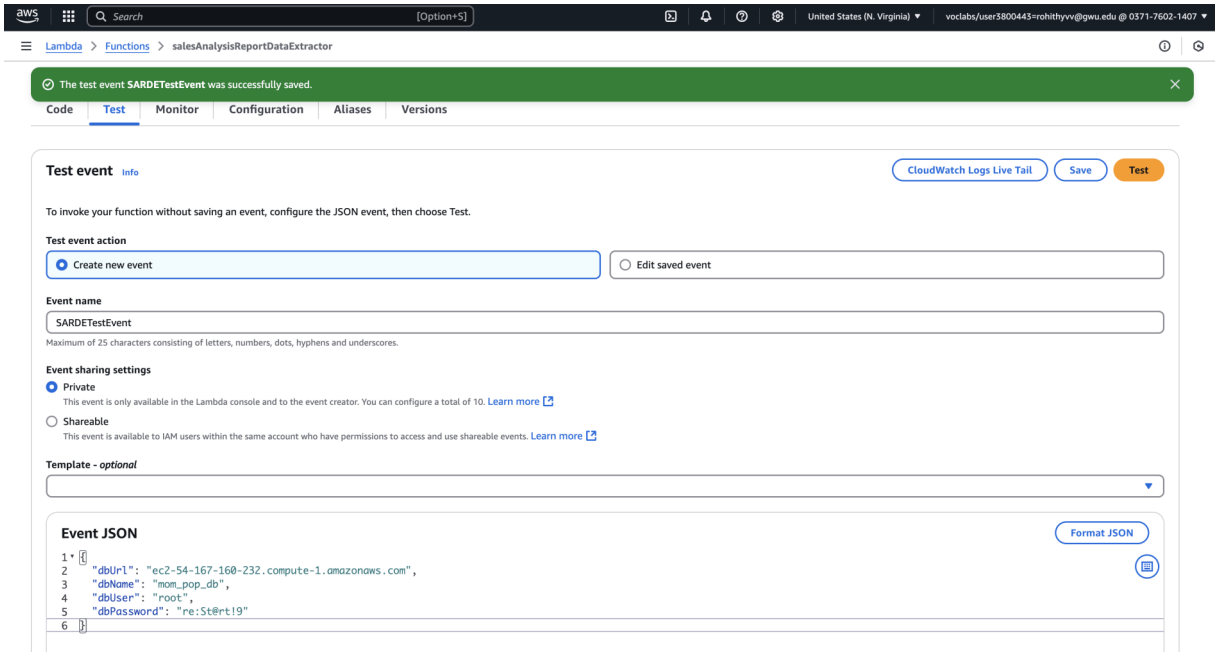
Fig. 5.5 AWS Lambda Function: salesAnalysisReportDataExtractor - VPC Configuration



Note: Screenshot showing the VPC configuration of the "salesAnalysisReportDataExtractor" Lambda function, including VPC ID, subnet, security groups, and inbound rules.

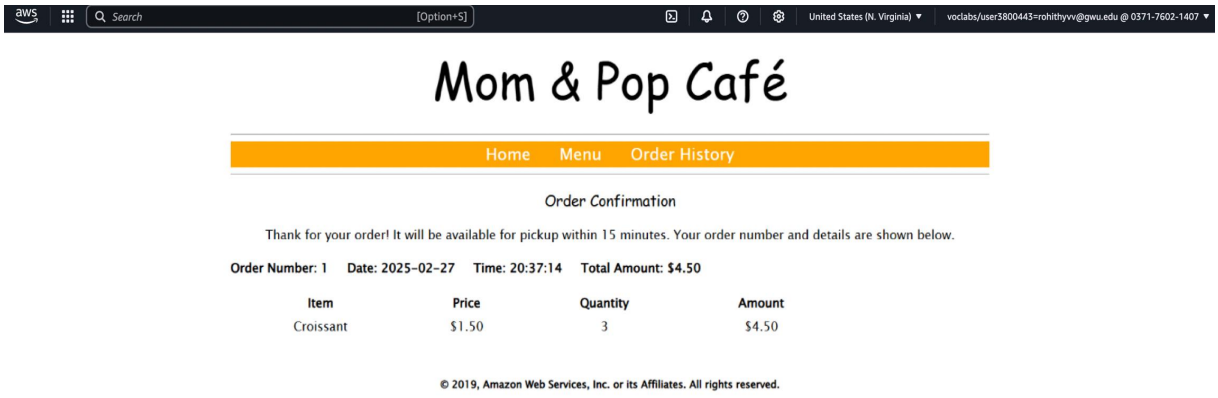
I learned, the importance of network configuration for Lambda functions that need to access resources within a VPC. I learned how to associate a Lambda function with a specific VPC, subnet, and security group, enabling it to connect to resources like databases or EC2 instances. The inbound rules shown for the security group, allowing traffic on ports 80 and 22, demonstrated how to control network access to the function. Seeing the VPC ID and subnet details reinforced the concept of deploying Lambda functions within a private network for enhanced security and connectivity.

Fig. 5.6 AWS Lambda Function: salesAnalysisReportDataExtractor - Test EventConfiguration



Note: Screenshot showing the "Test" tab of the "salesAnalysisReportDataExtractor" Lambda function in the AWS console, displaying the configuration of a test event named "SARDETestEvent" with database connection details in JSON format.

Fig. 5.7 Order Confirmation - Mom & Pop Cafe



Note: This order confirmation page shows a successful purchase of three croissants for **$4.50**. The order will be ready for pickup within **15 minutes**, ensuring quick service.

I learned, how to create and configure test events for AWS Lambda functions. I learned how to define the event structure in JSON format, including parameters like database connection

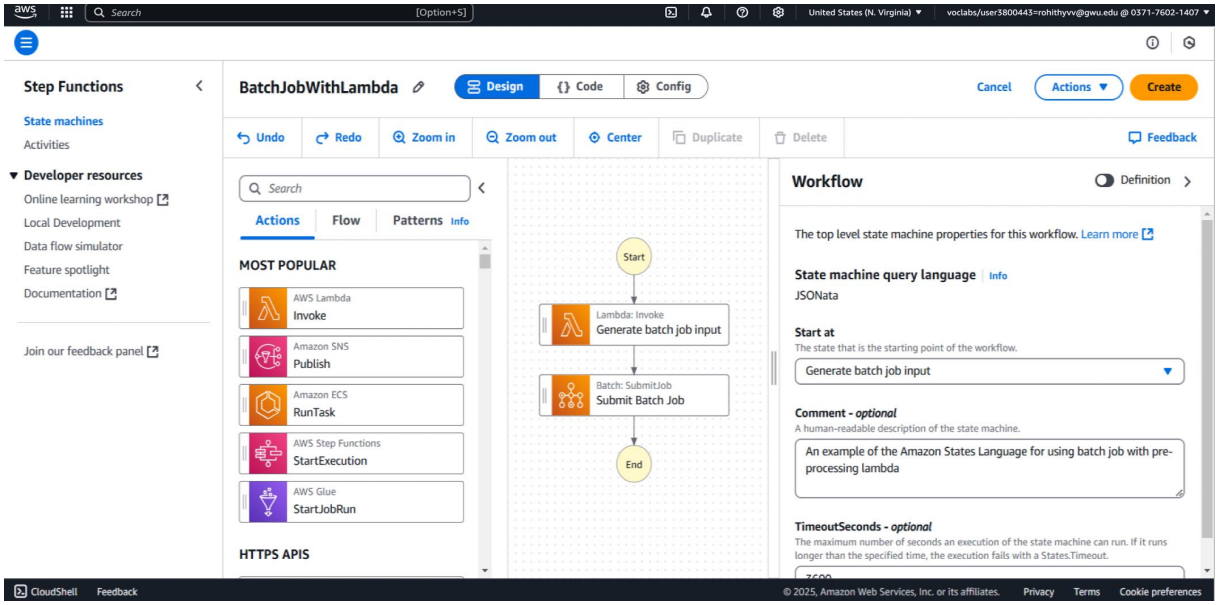
details, which are used to simulate real-world events that trigger the Lambda function. The ability to create and save test events allows for thorough testing of Lambda functions before deployment, ensuring they handle different input scenarios correctly. The "Private" event sharing setting also highlighted the option to control event visibility, which is important for security and collaboration.

Module 5 focused on serverless computing and containerization, highlighting AWS services that enhance application scalability, efficiency, and management. I explored AWS Lambda, which allows running code without the need for server provisioning or management, making it well-suited for event-driven applications. Additionally, I gained insights into APIs, particularly RESTful APIs, and how Amazon API Gateway streamlines API creation, management, and security.

The module also covered containerization and AWS services that facilitate it, including Amazon Elastic Container Service (ECS), Elastic Kubernetes Service (EKS), and AWS Fargate, which offer scalable and efficient container management solutions. Lastly, I delved into AWS Step Functions, which orchestrate workflows by integrating various AWS services to enable automated and scalable application development. This module deepened my understanding of serverless computing and container orchestration, both of which are crucial for building modern cloud-native applications.

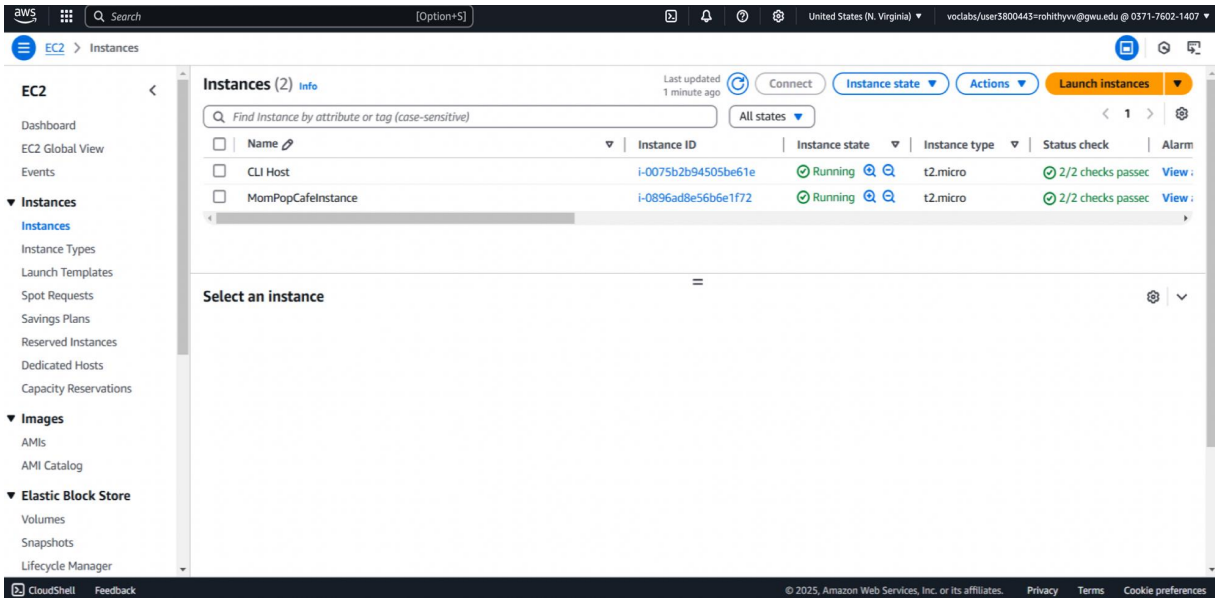
Module 5 - Demo

Fig. 5.8 Demo1 - Introduction to AWS Step Functions



Note: This AWS Step Functions workflow automates batch job submission using AWS Lambda for preprocessing. It demonstrates state machine execution, integrating Lambda and AWS Batch to streamline job processing efficiently.

Fig. 5.9 Demo2 - Deploy Docker Container on Amazon Elastic Container Service (ECS)



Note: In this activity, I have deployed docker container on AWS Elastic Container Service
This AWS EC2 dashboard displays two running t2.micro instances, CLI Host and MomPopCafeInstance. Both instances have passed status checks, ensuring operational stability in the US East (N. Virginia) region.

Conclusion:

These modules provided a comprehensive journey through fundamental cloud operations concepts and techniques in AWS. Starting with an overview of cloud computing and the AWS ecosystem, I progressed to hands-on experience with essential services like EC2, Systems Manager, and Auto Scaling. The exploration of containerization and serverless computing with, ECS, and Lambda expanded my understanding of modern application deployment methods. Through practical labs and activities, I gained valuable skills in managing, automating, and troubleshooting cloud resources, reinforcing the importance of scalability, efficiency, and security in cloud operations. This learning path has equipped me with a solid foundation for navigating the evolving landscape of cloud computing and effectively managing applications in the AWS cloud.

References:

Include citations in APA format for any external references used.

Amazon Web Services. (2022). AWS Academy Cloud Foundations. Retrieved from https://aws.amazon.com/training/awsacademy/

Bradley University. (2022). AWS Academy Cloud Foundations. Retrieved from https://www.bradley.edu/academic/continue/programs/aws-cloud-foundations/

Studocu. (2022). AWS Academy Cloud Foundations M0. Retrieved from https://www.studocu.com/row/document/innovation-academy/aws-academy-cloud-foundations/100-acclfo-20-en-m0sg-aws-academy-cloud-foundations-m0/116513417