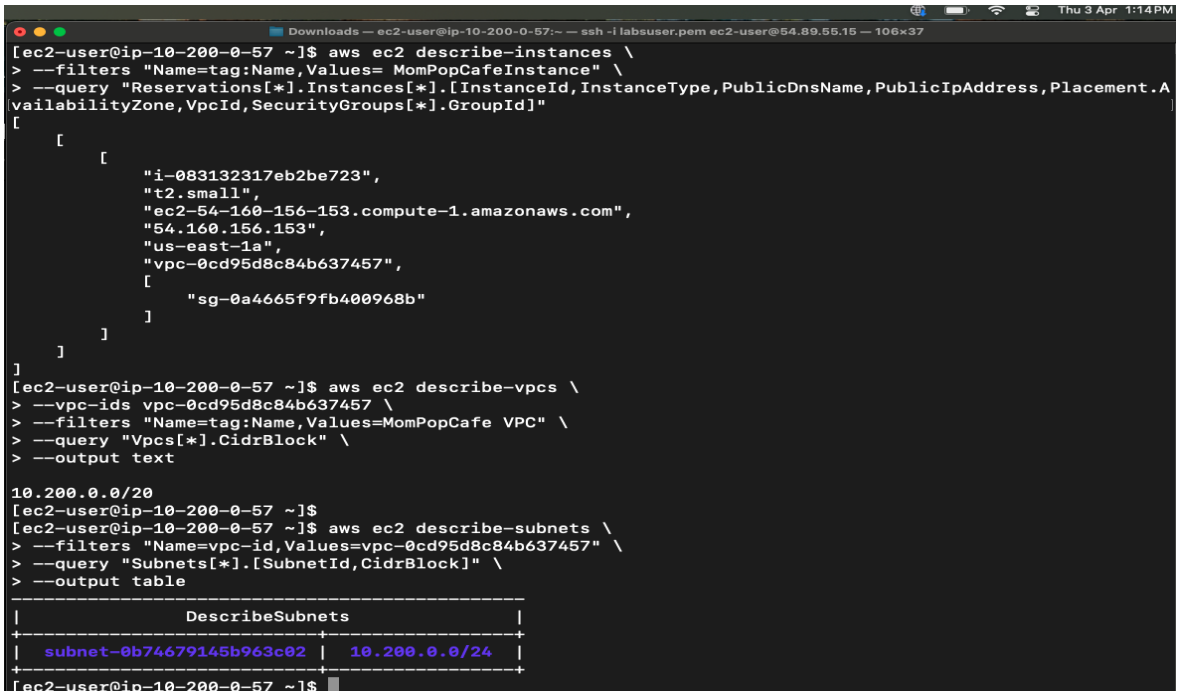
Module 6: Activity 6 - Migrate to Amazon RDS

Task 1.1: Connect to the CLI Host instance by using SSH

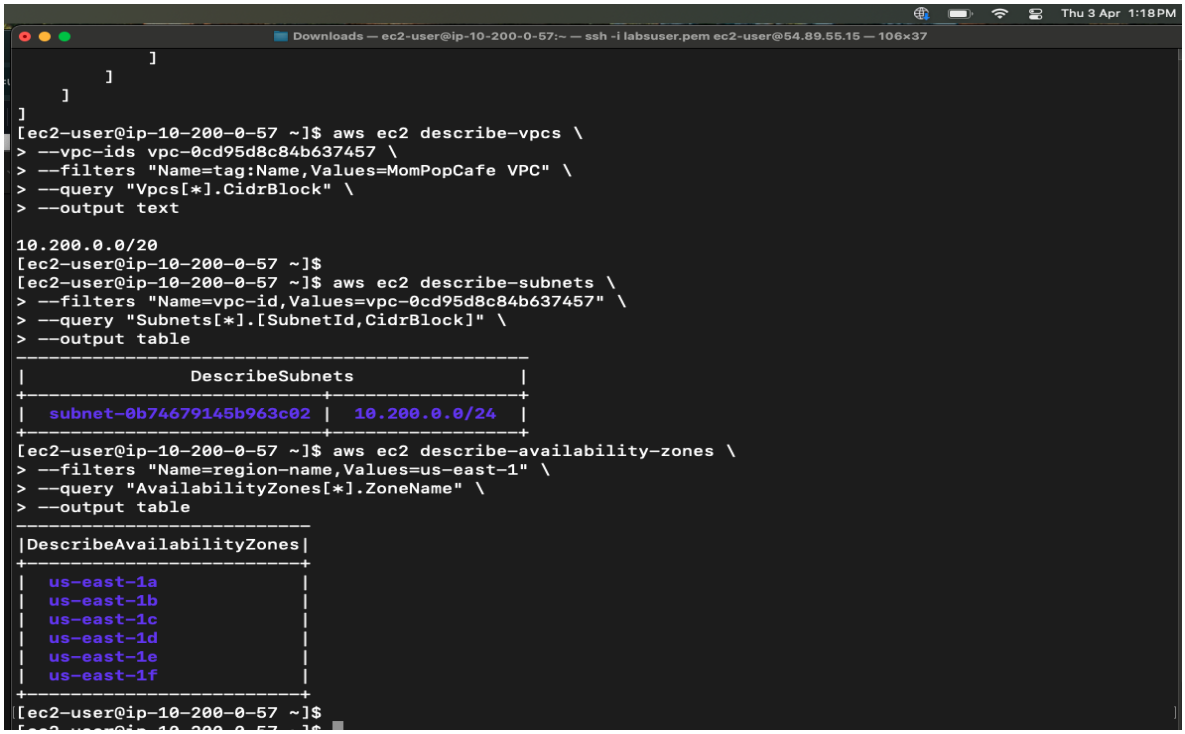
Fig 6.1: Running AWS CLI to Describe EC2 Instances



Note: This image shows the command used to describe the EC2 instances associated with the "MomPopCafeInstance" tag, returning key information about the instance like Instance ID, Type, DNS, IP, Availability Zone, and VPC ID.

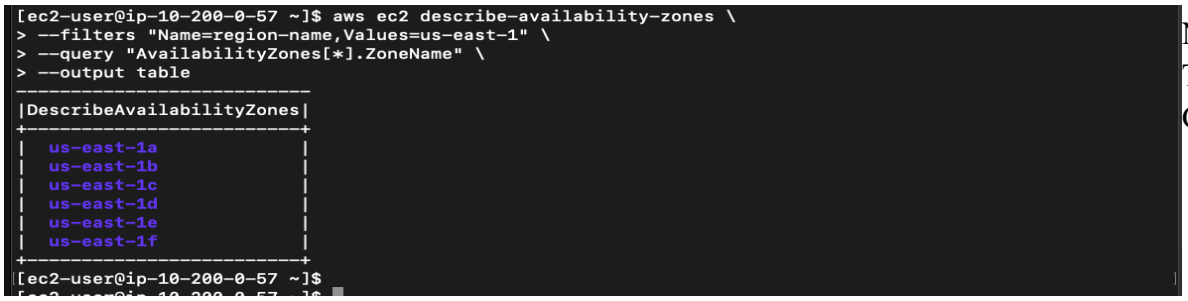
Task 1.2: Gather current environment configuration information

Fig 6.2: Retrieving Subnet CIDR Block



Note: The command retrieves the CIDR block of the subnet associated with the "MomPopCafe" VPC to ensure proper networking and resource placement.

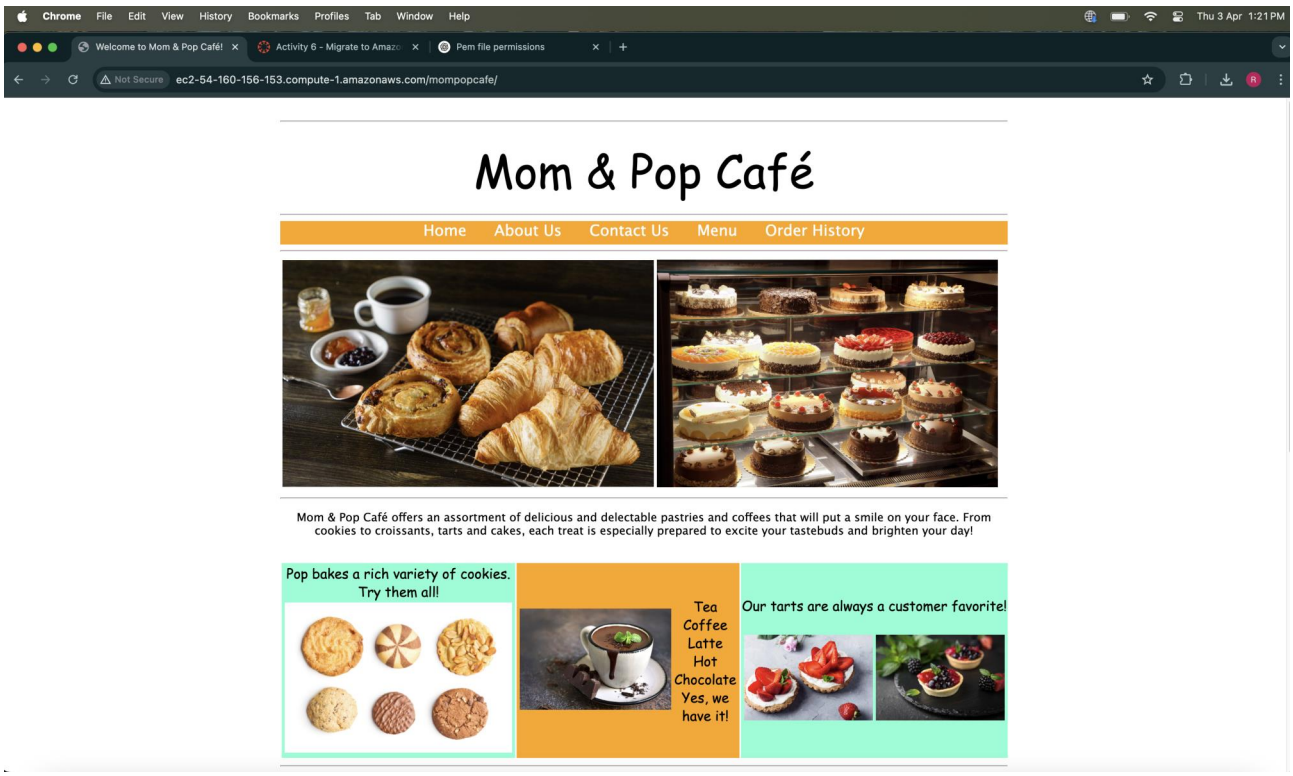
Fig 6.3: Retrieving Availability Zone Information



Note: The CLI command retrieves the list of Availability Zones in the "us-east-1" region, which is important for subnet distribution in the migration process.

Task 1.3: Create prerequisite components

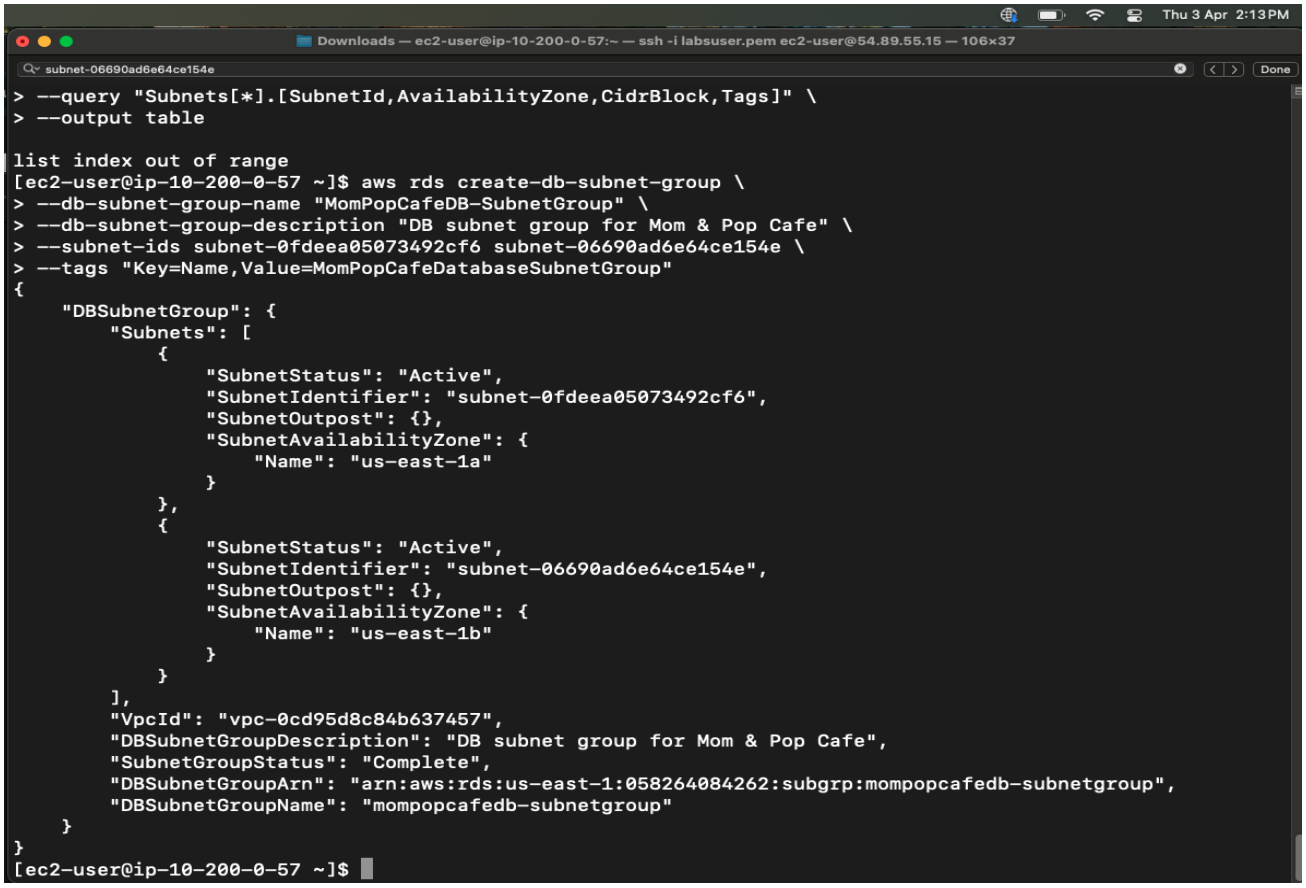
Fig 6.4: Web Server Test Page



Note: This image shows the initial test page of the Mom & Pop Café website hosted on the EC2 instance, verifying the site's availability before migration.

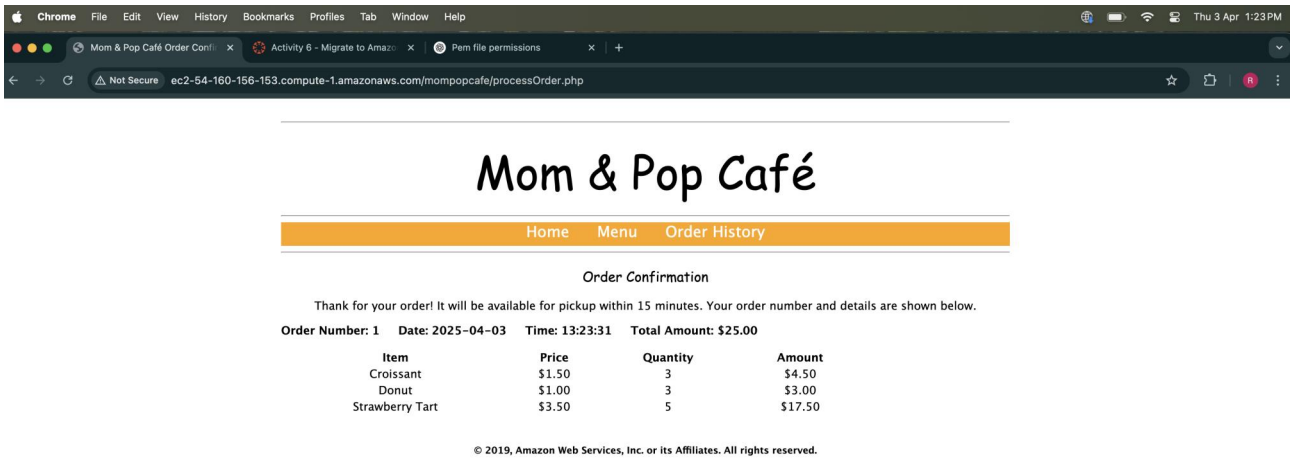
Task 1.4: Create the Amazon RDS MariaDB instance

Fig 6.5: Create the Amazon RDS MariaDB instance



Note: The order confirmation page shows successful order processing, confirming that the application functionality is intact after the database migration to RDS. Configuring the AWS CLI environment to define the AWS account credentials, Region name, and output format to use with the command aws configure and enter the values which are asked to configure.

Fig 6.6: Order Confirmation Page



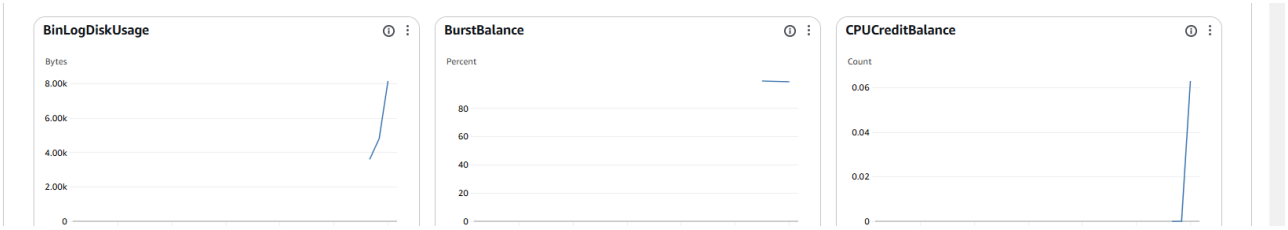
Note: The screenshot displays the order confirmation page for the "Mom & Pop Café" website. It shows a successful order with itemized details, demonstrating the continued functionality of the website post-migration.

Fig 6.7: SQL statement to retrieve the data in the product table

```
MariaDB [mom_pop_db]> select * from product;
+----+--------------------------+-------------------------------------------------------------------------+-------
+---------------+-----------------------------------------+
| id | product_name             | description                                                             | price
| product_group | image_url                               |
+----+--------------------------+-------------------------------------------------------------------------+-------
+---------------+-----------------------------------------+
|  1 | Croissant                | Fresh, buttery and fluffy... Simply delicious!                          |  1.50
|             1 | images/Croissants.jpg                   |
|  2 | Donut                    | We have more than half-a-dozen flavors!                                 |  1.00
|             1 | images/Donuts.jpg                       |
|  3 | Chocolate Chip Cookie    | Made with Swiss chocolate with a touch of Madagascar vanilla            |  2.50
|             1 | images/Chocolate-Chip-Cookies.jpg       |
|  4 | Muffin                   | Banana bread, blueberry, cranberry or apple                             |  3.00
|             1 | images/Muffins.jpg                      |
|  5 | Strawberry Blueberry Tart | Bursting with the taste and aroma of fresh fruit                       |  3.50
|             1 | images/Strawberry-&-Blueberry-Tarts.jpg |
|  6 | Strawberry Tart          | Made with fresh ripe strawberries and a delicious whipped cream         |  3.50
```

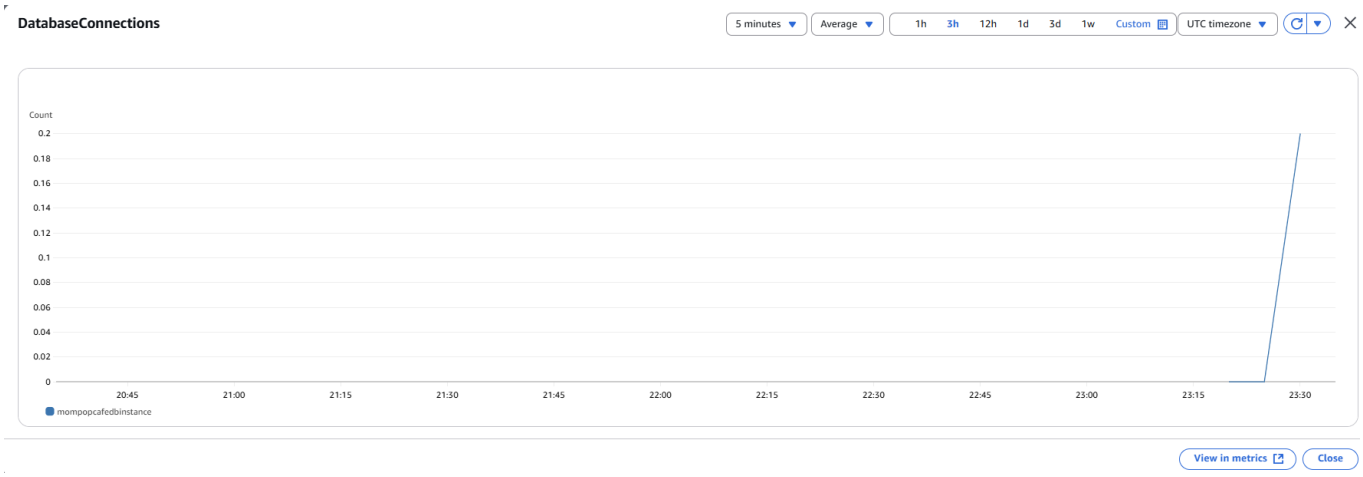
Note: checking whether migrated application data to Amazon RDS by first establishing an SSH connection to the application server. Since, creating a backup of the local database using `mysqldump`. Finally, restore this backup file to the newly created Amazon RDS instance, ensuring successful data transfer.

Fig 6.8: Monitor the Amazon RDS database



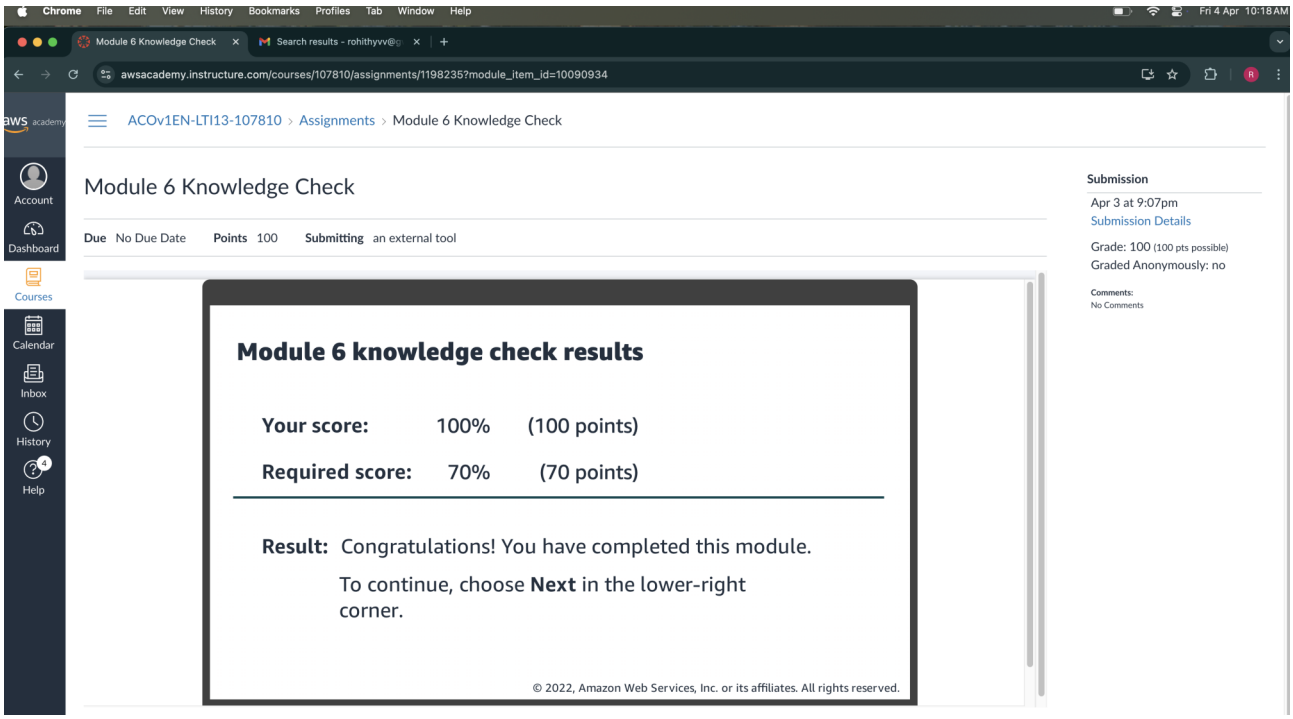
Note: In the above image, we can see the performance metrics of our DB, One of the benefits of using Amazon RDS is the ability to monitor the performance of a database instance. Amazon RDS automatically sends metrics to Amazon CloudWatch every minute for each active database.

Fig 6.9: Database connections monitoring



Note: DB Connections graph shows a line that indicates that 1 connection is in use. This connection was established by the interactive SQL session from the MomPopCafeInstance.

Fig 6.10: Module 6 Knowledge Check Results



Note: This screenshot shows the successful completion of the Module 6 Knowledge Check, confirming the understanding of the migration process and Amazon RDS concepts.

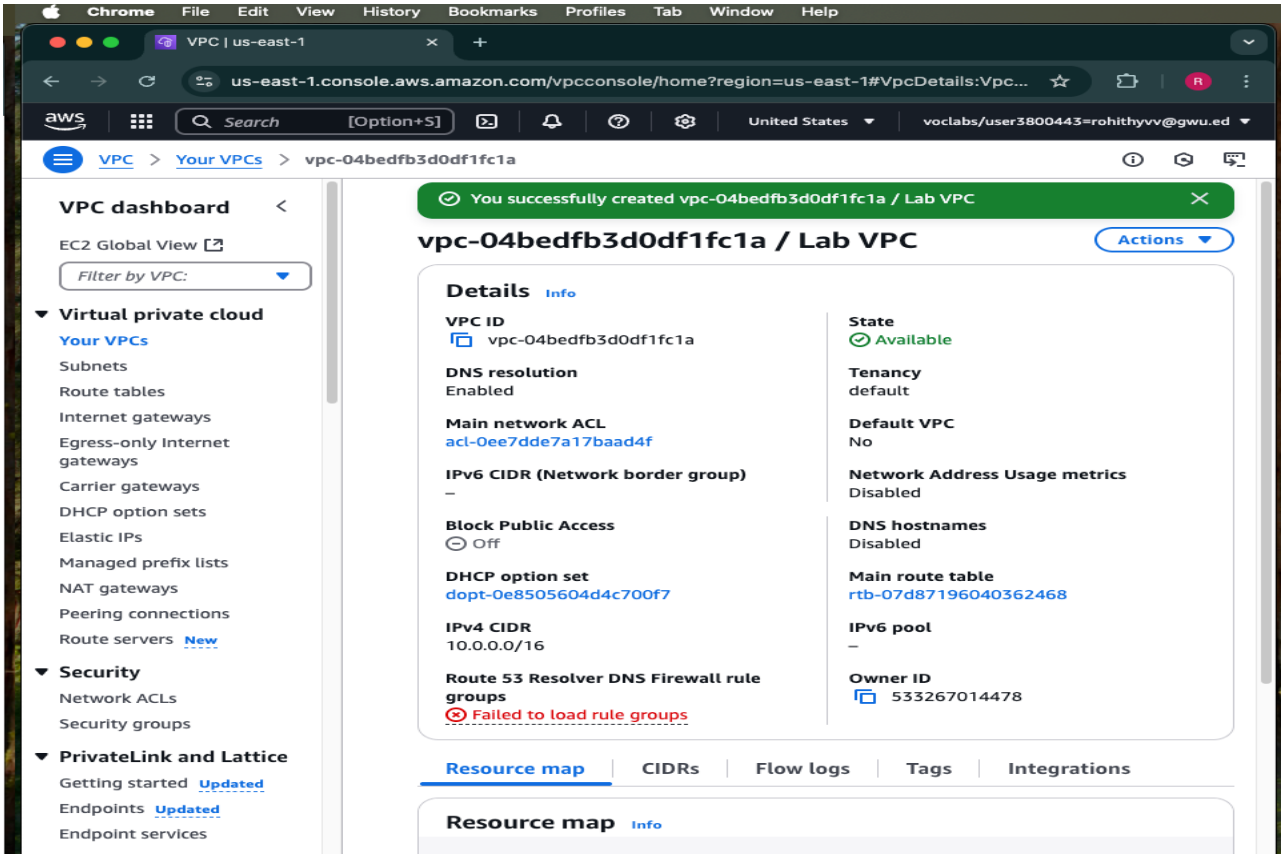
Summary:

- Amazon Aurora ensures high availability by replicating data across three Availability Zones, supporting automatic failover, read replicas, and using a distributed storage system.
- AWS Database Migration Service (DMS) helps migrate data securely and efficiently between databases.
- AWS Schema Conversion Tool (SCT) converts the source database schema to be compatible with the target database.
- Migrating a database involves converting the schema with SCT and then replicating the data using DMS.

Module 7: Activity 7 – Networking

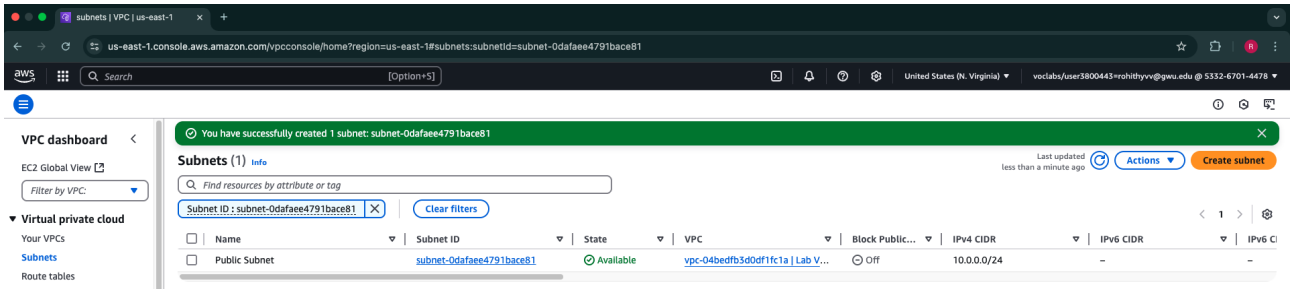
Lab 4 : Configure VPC

Fig 7.1 : Created Lab VPC for the application



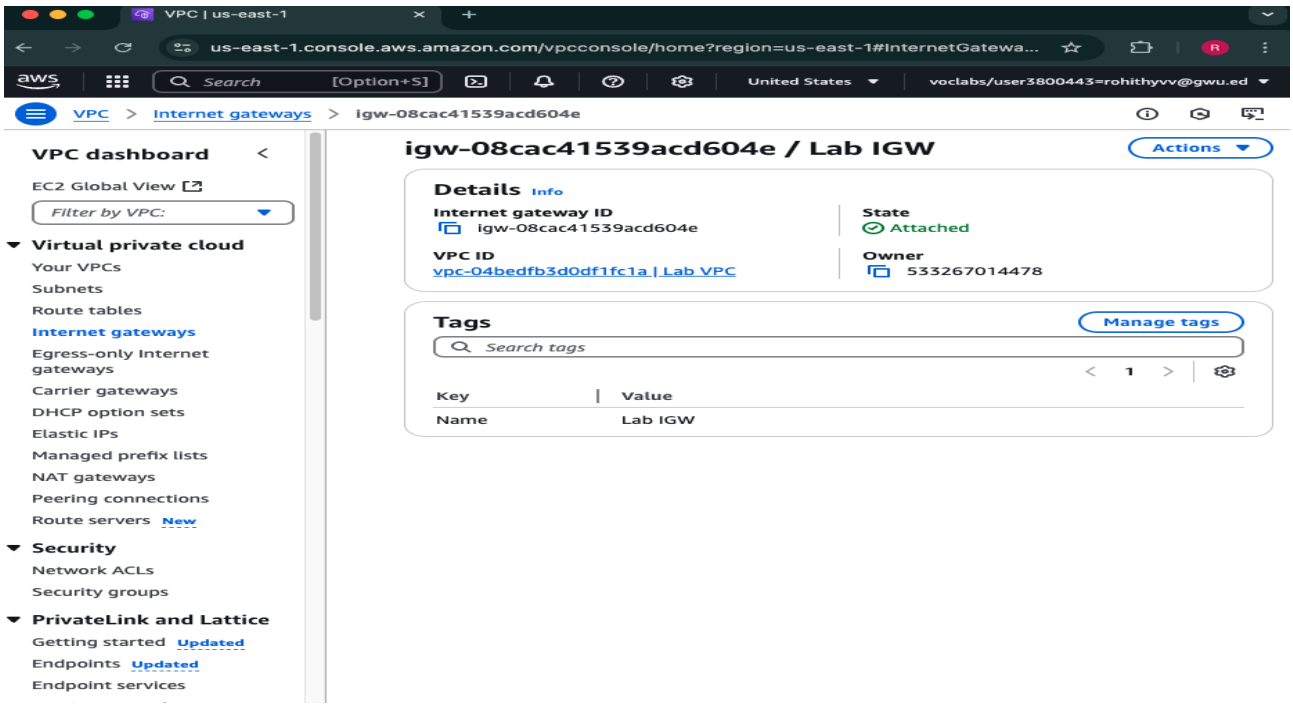
Note: This image corresponds to the successful creation of a VPC in the AWS console, showing the details of the created Lab VPC, which is the starting step of the lab.

Fig 7.2 : Create Subnets



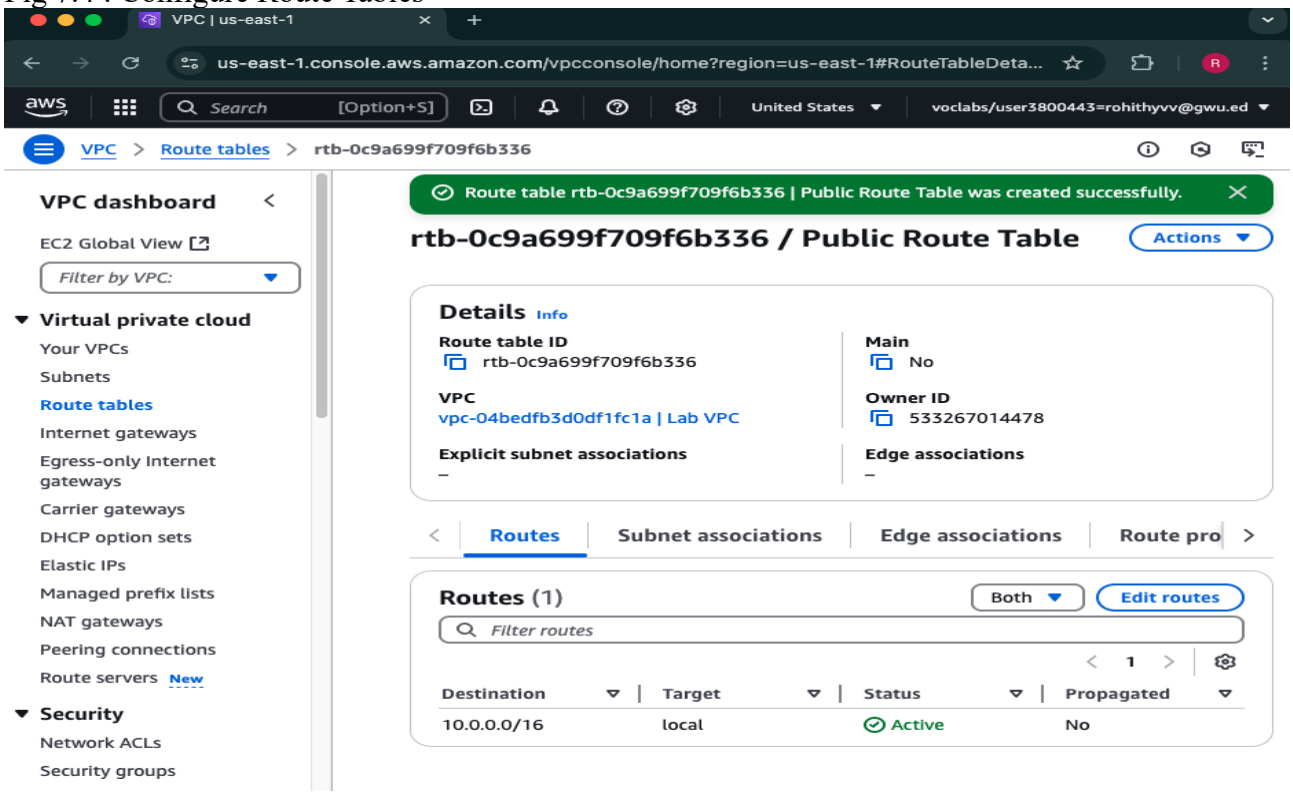
Note: This image reflects the creation of a public subnet within the "Lab VPC." It shows the subnet details like the ID and CIDR block (10.0.0.0/24). The subnet is marked as "Available," indicating its readiness for further configuration.

Fig 7.3 : Create an Internet Gateway



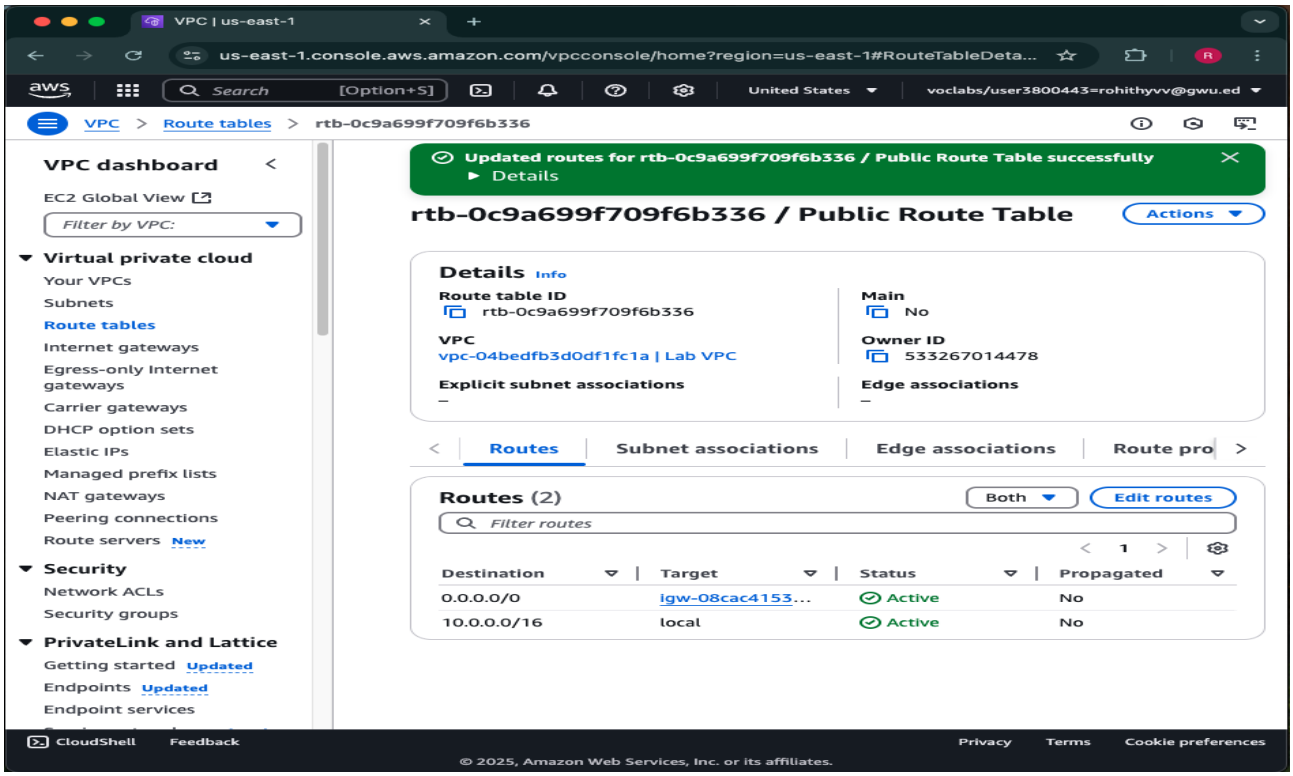
Note: This image shows the successful creation of the Internet Gateway (IGW) for the Lab VPC. It is attached to the VPC, enabling internet connectivity for resources in the public subnet, which is crucial for enabling internet-bound traffic.

Fig 7.4 : Configure Route Tables



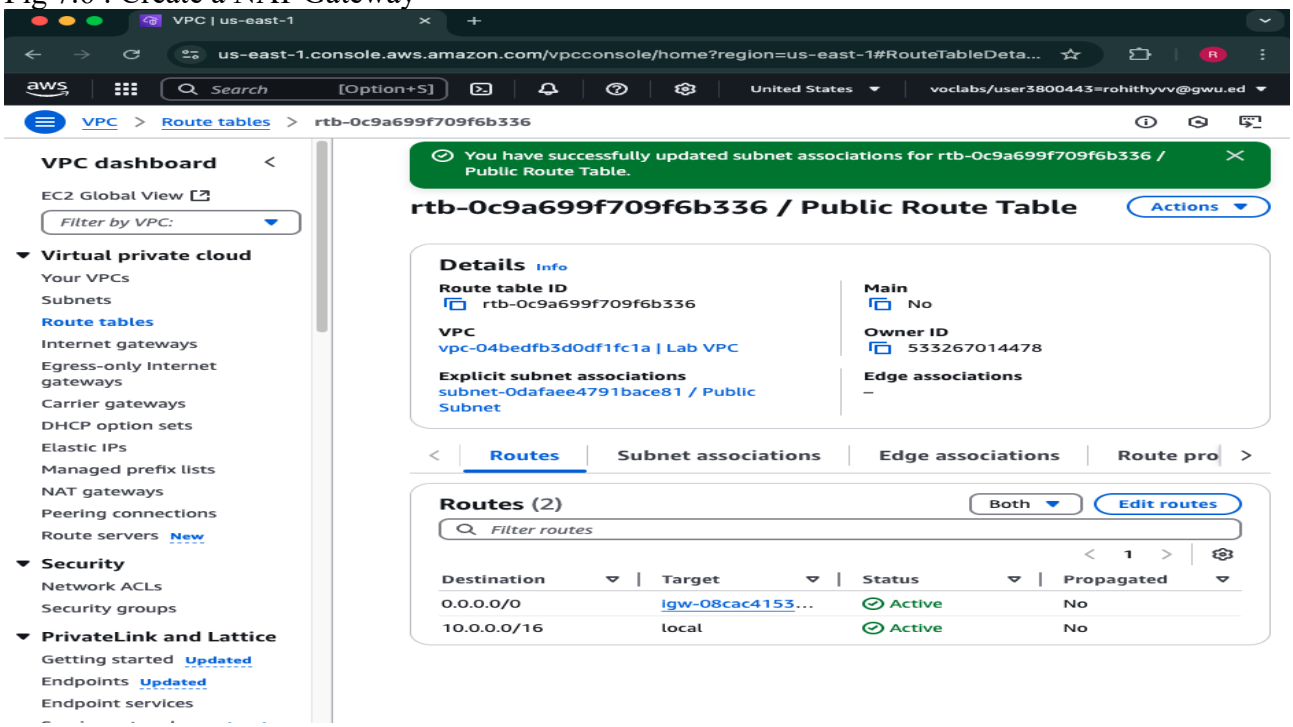
Note: This image illustrates the creation of a public route table for the Lab VPC, which includes routes to direct internet-bound traffic (0.0.0.0/0) to the internet gateway. The route table setup is an essential step for configuring the public subnet.

Fig 7.5 : Launch a Bastion Server in the Public Subnet



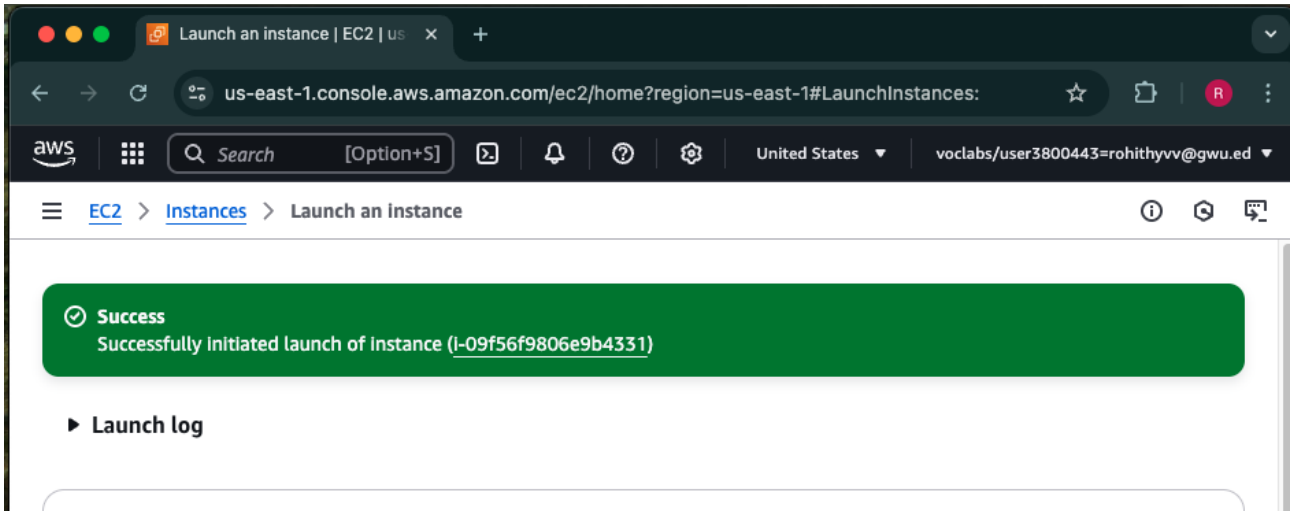
Note: This image shows the EC2 instance (Bastion Server) launched in the public subnet. It serves as a jump host for securely accessing instances in the private subnet. The instance has a public IP for direct SSH access.

Fig 7.6 : Create a NAT Gateway



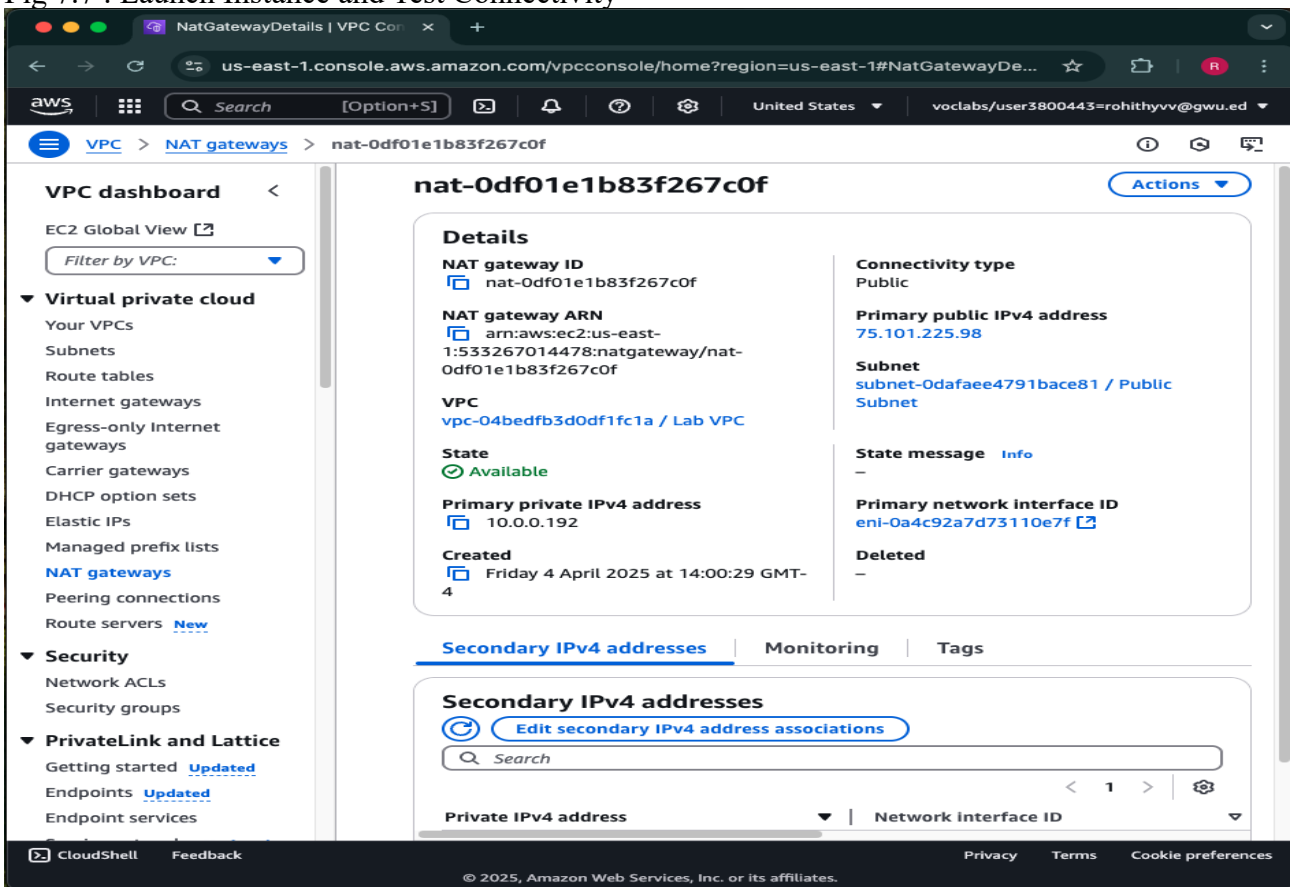
Note: This image depicts the creation of the NAT Gateway, which is placed in the public subnet. The NAT Gateway enables instances in the private subnet to access the internet securely while preventing inbound access from the internet.

Fig 7.7 : Testing the Private Subnet



Note: This image shows the update to the private route table, configuring it to route outbound traffic to the NAT Gateway. It confirms that private subnet resources can reach the internet through the NAT Gateway.

Fig 7.7 : Launch Instance and Test Connectivity



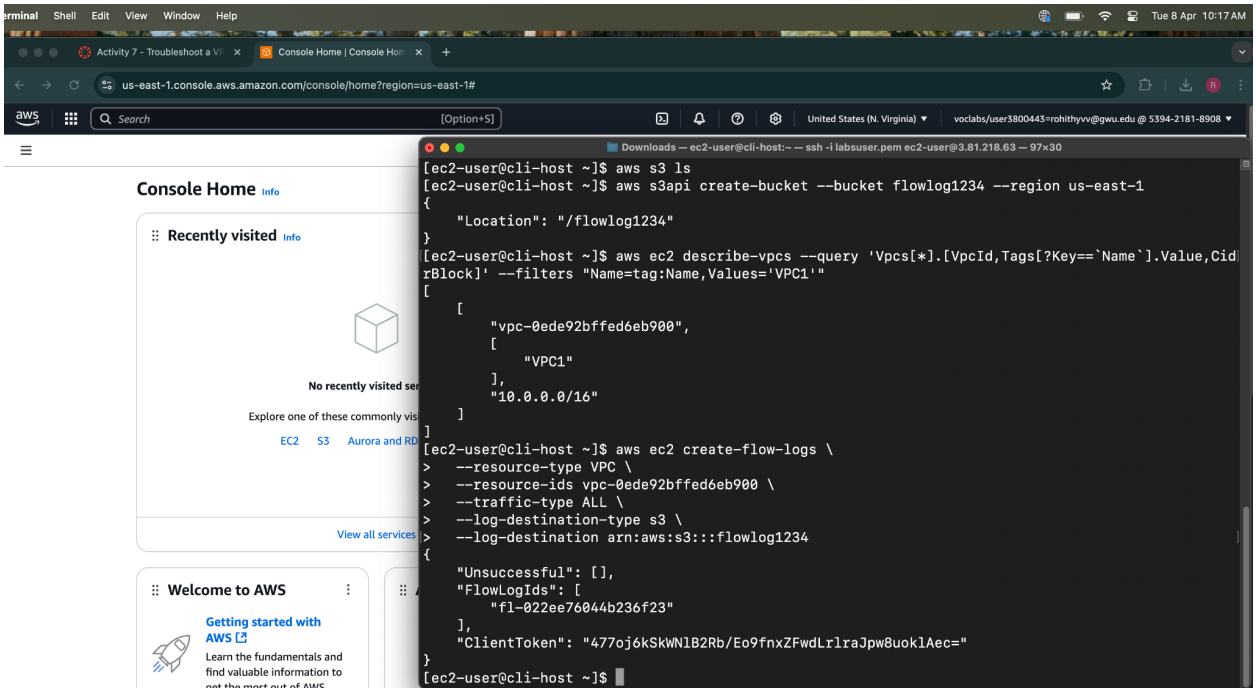
Note: This image illustrates the successful launch of an EC2 instance within the Lab VPC. The next steps include connecting to this instance and testing connectivity, validating that the network configuration works as expected.

Activity 7 : Troubleshoot a VPC

Fig 7.8 : Configure the AWS CLI on the CLI Host EC2 Instance

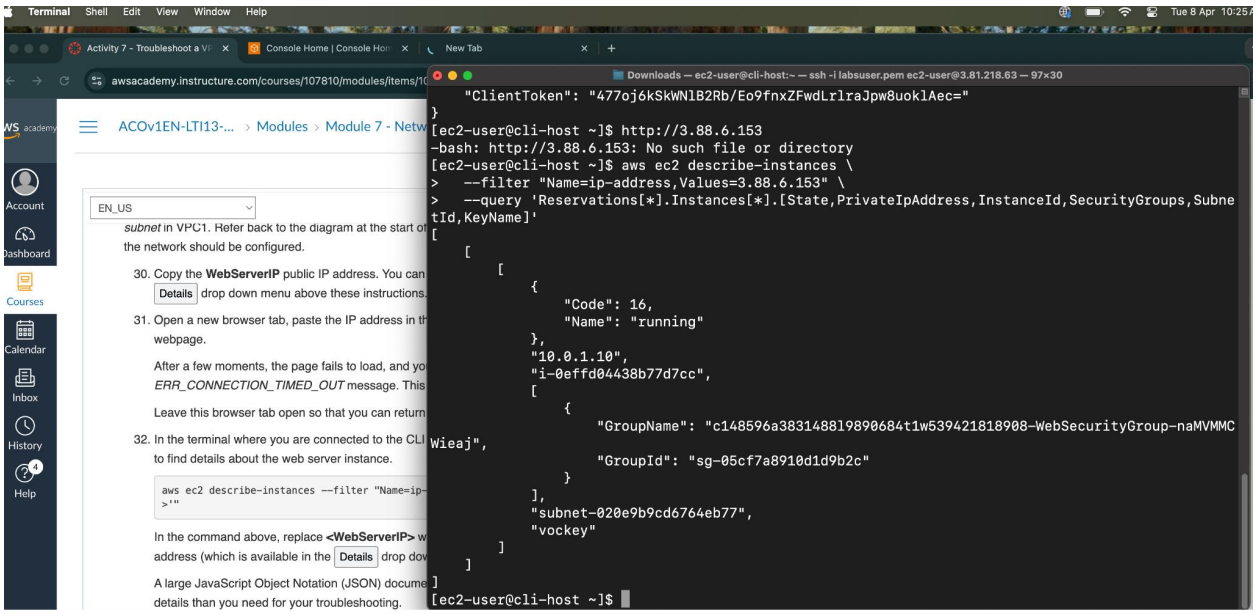
Note : In the above image we discover the region in which the CLI Host instance is running and Update the AWS CLI software with the credentials.

Fig 7.8 : Confirming VPC Flow Logs Settings



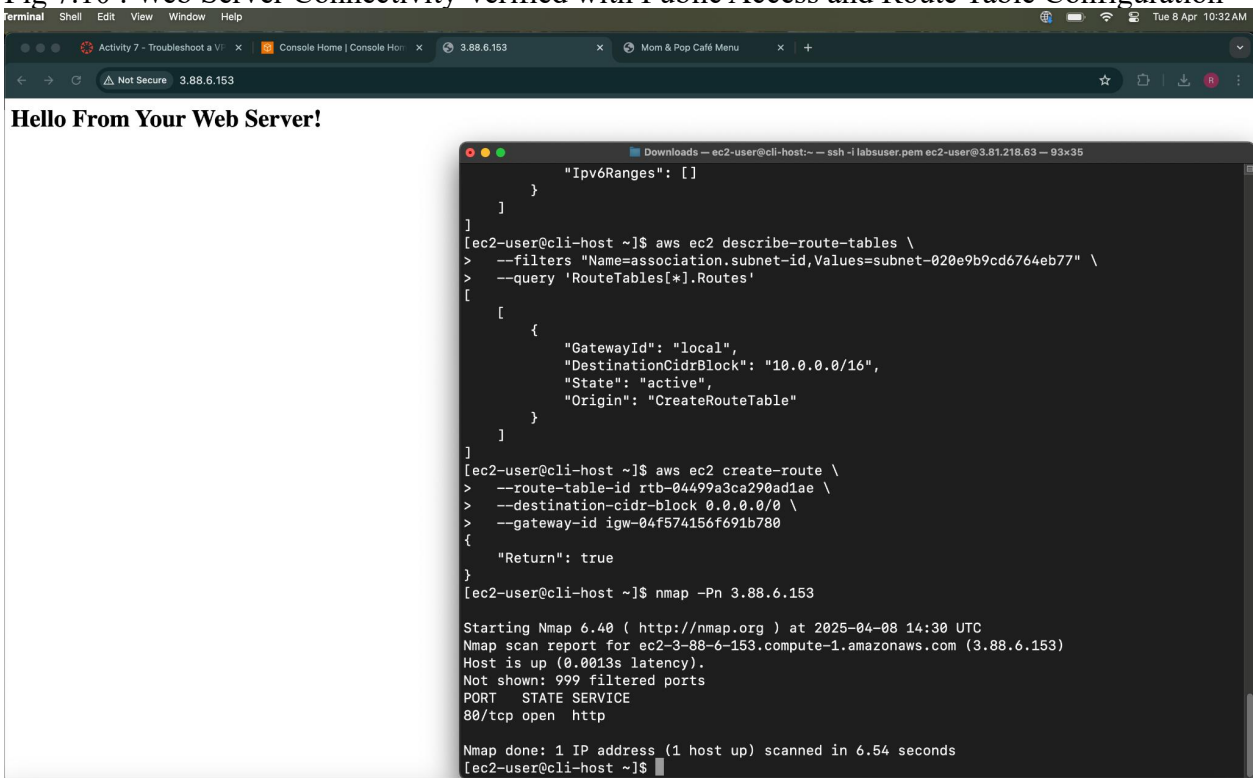
Note: This screenshot captures the configuration interface for enabling VPC Flow Logs. It helps monitor and troubleshoot network traffic by logging information about IP traffic going to and from network interfaces in the specified VPC.

Fig 7.9 : Investigating EC2 Web Server Instance with AWS CLI



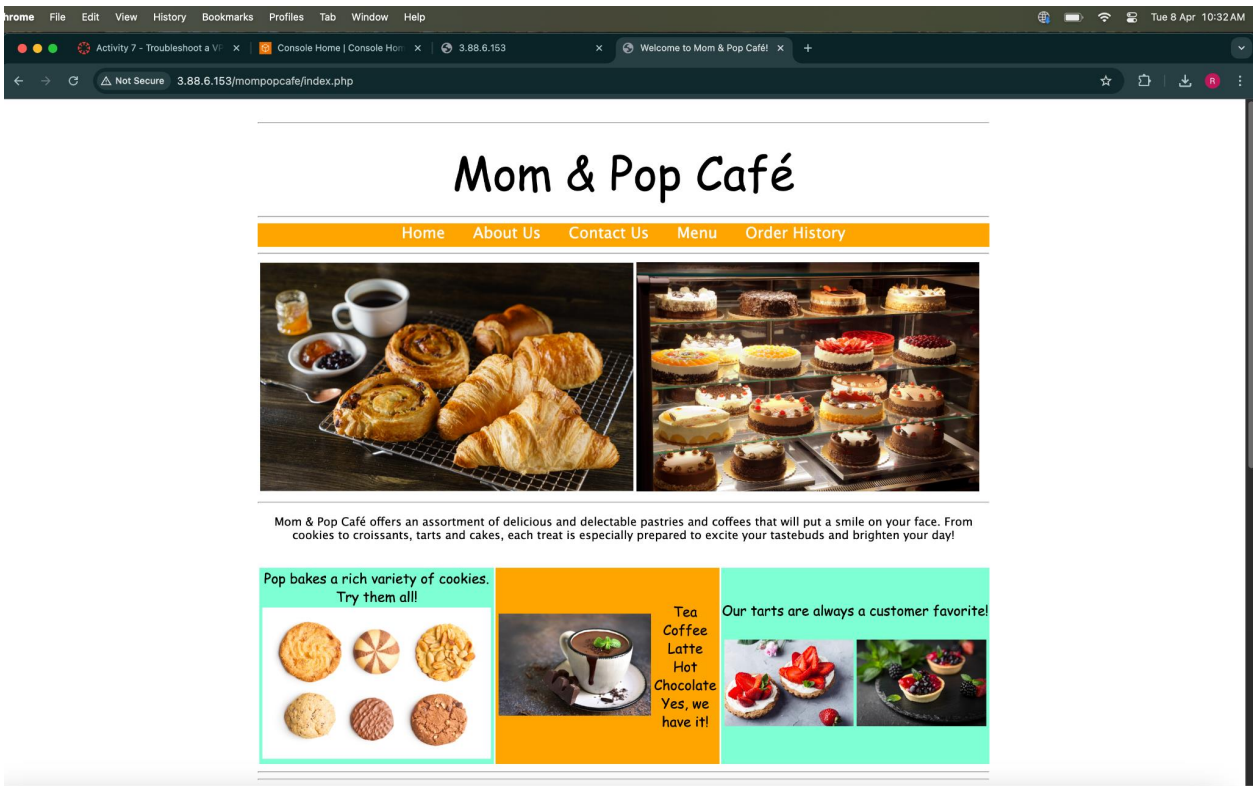
Note: This screenshot shows the use of the aws ec2 describe-instances command to fetch instance metadata for troubleshooting a connection issue to a public IP (3.88.6.153). The instance is confirmed as “running” in subnet vockey with security group details provided.

Fig 7.10 : Web Server Connectivity Verified with Public Access and Route Table Configuration



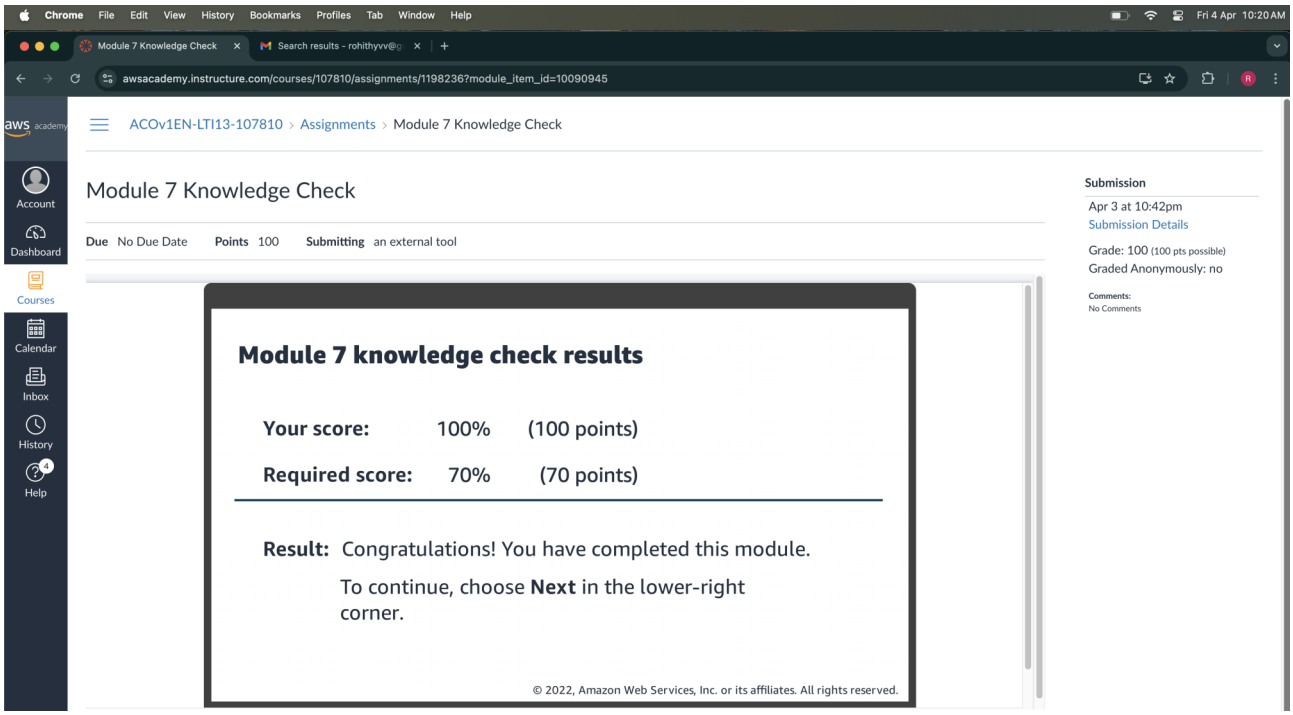
Note:This screenshot confirms successful access to a web server at IP 3.88.6.153, displaying the message “Hello From Your Web Server!” in a browser. Terminal commands show that a route to the internet was created via igw-0457f4516f691b780, and port 80 is open, verified using nmap.

Fig 7.11 : Successful Deployment of the Mom & Pop Café Web Application



Note: This screenshot confirms the successful launch of the full Mom & Pop Café website hosted at 3.88.6.153/mompopcafe/index.php. The page displays a complete frontend with navigation, pastries and cake visuals, and engaging marketing content—verifying that the EC2 web server and PHP application are functioning correctly.

Fig 7.12 : Module 7 Knowledge Check – Perfect Score Achievement

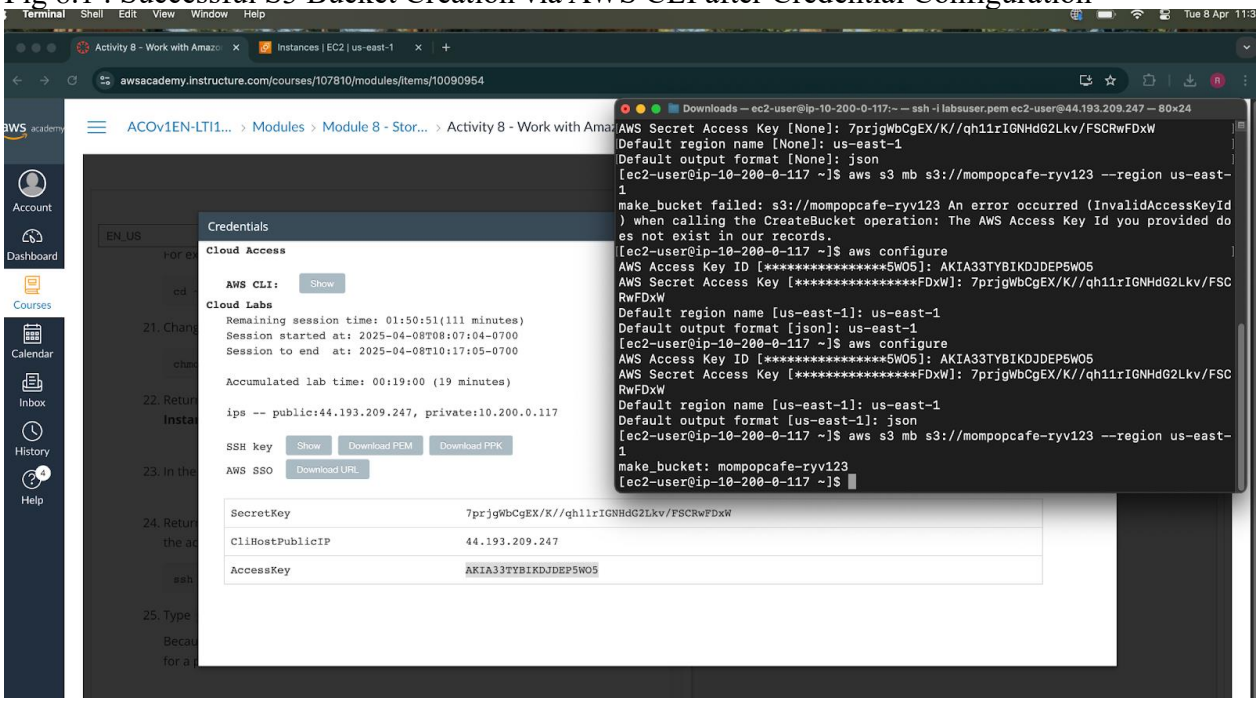


Note: This screenshot shows successful completion of the "Module 7 Knowledge Check" on AWS Academy with a perfect score of 100%. The required passing score was 70%. This confirms mastery of VPC configuration concepts, including subnets, route tables, and internet gateway setup.
Summary:

- CIDR Blocks Define IP Ranges in VPC: When creating a VPC, a CIDR block determines the range of IP addresses available for the network and its subnets.
- Subnets Enable Segmentation and Isolation: VPC CIDR blocks are divided into smaller subnets for logical grouping, isolation, and traffic control, supporting both public and private subnets.
- NAT Devices Enable Internet Access for Private Subnets: NAT gateways allow instances in private subnets to access the internet securely while blocking incoming traffic.
- VPC Peering and VPN Enable Cross-Network Communication: VPC peering allows private IP routing between VPCs, while Site-to-Site VPN connects on-premises data centers to AWS securely over the internet.
- Route Tables and Security Groups Control Traffic: Route tables manage traffic flow at the subnet level, while security groups manage access permissions at the instance level.

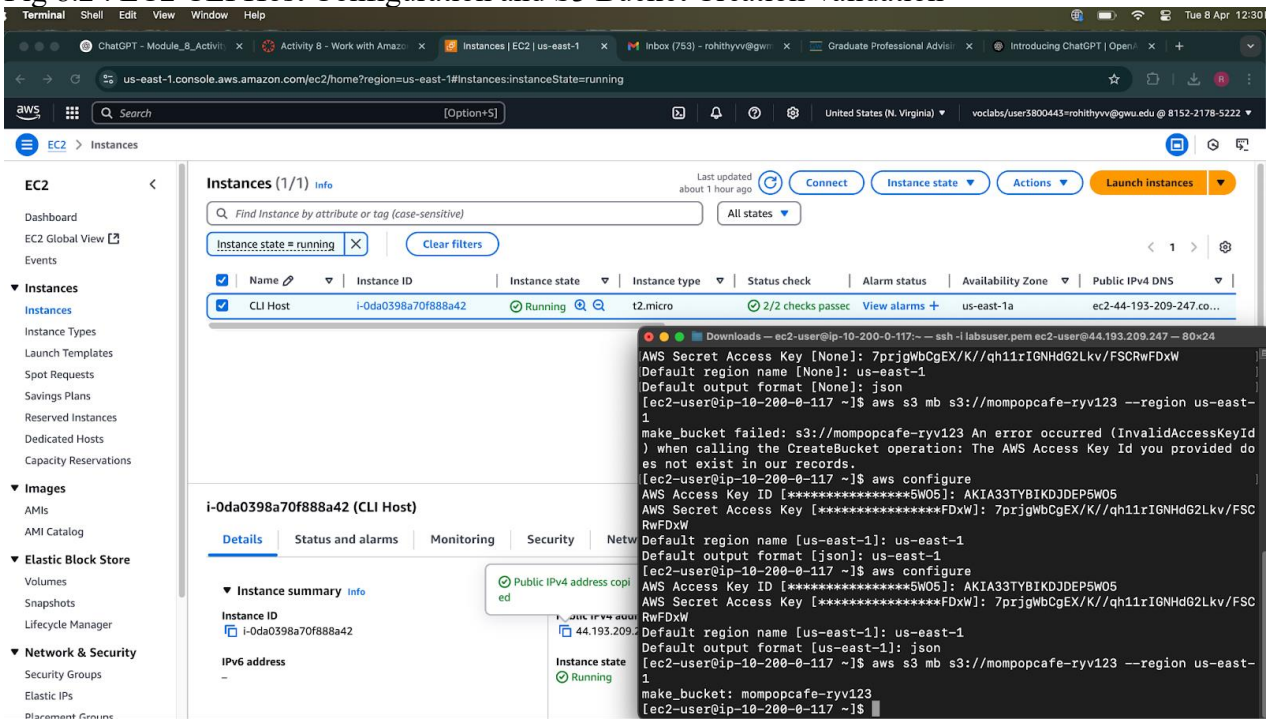
Module 8: Activity 8 –

Fig 8.1 : Successful S3 Bucket Creation via AWS CLI after Credential Configuration



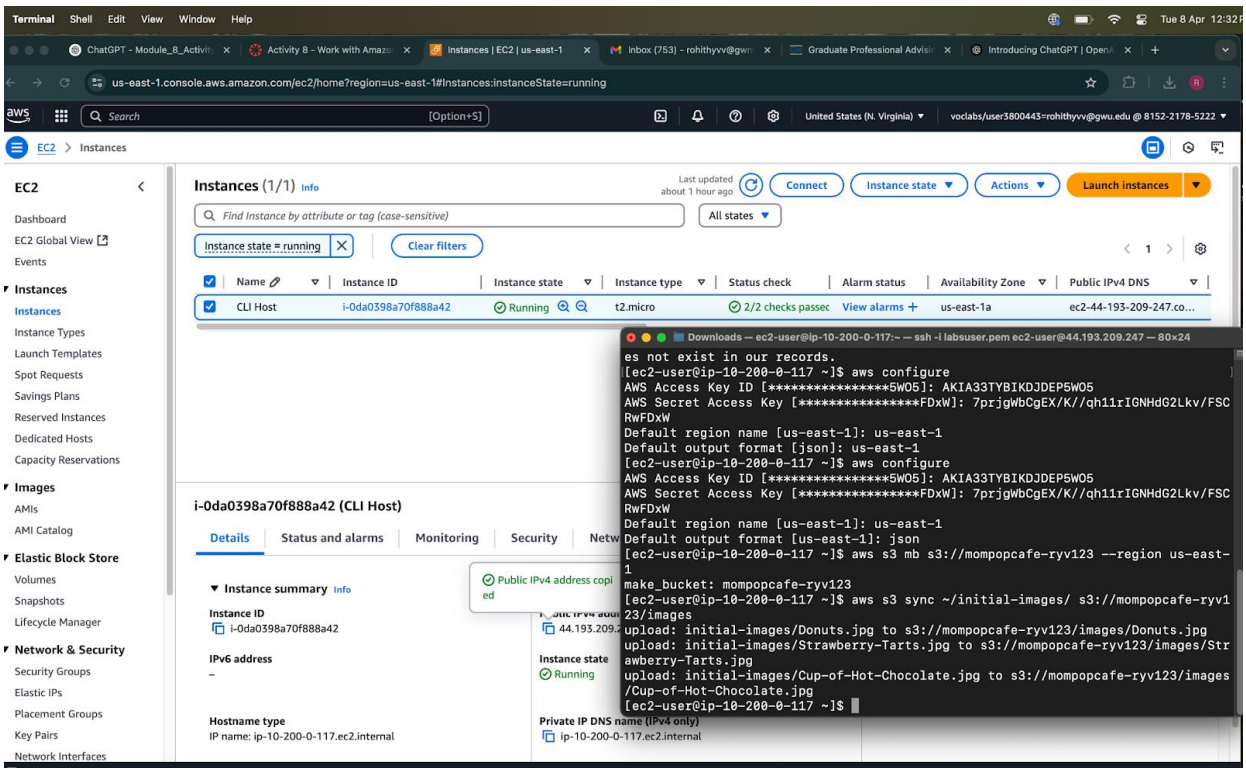
Note: This image shows a successful S3 bucket creation (mompopcafe-ryv123) using the AWS CLI after resolving an initial "InvalidAccessKeyId" error. The user corrected the AWS access credentials via aws configure, enabling proper authentication for the operation.

Fig 8.2 : EC2 CLI Host Configuration and S3 Bucket Creation Validation



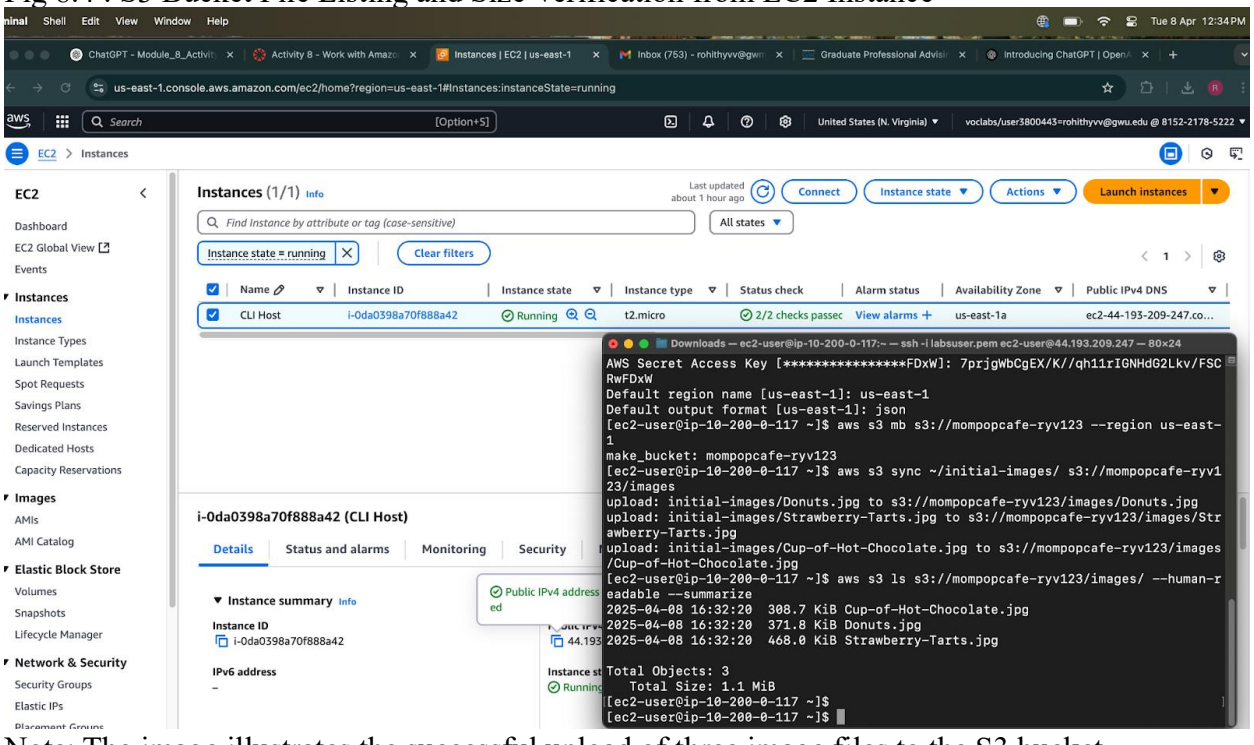
Note: The screenshot displays the AWS EC2 dashboard with a running CLI Host instance and the terminal where AWS CLI commands were executed. After resolving an InvalidAccessKeyId error by configuring credentials, the user successfully created the S3 bucket mompopcafe-ryv123 in the us-east-1 region.

Fig 8.3 : Image Upload to S3 Bucket from EC2 CLI Host via AWS CLI Sync



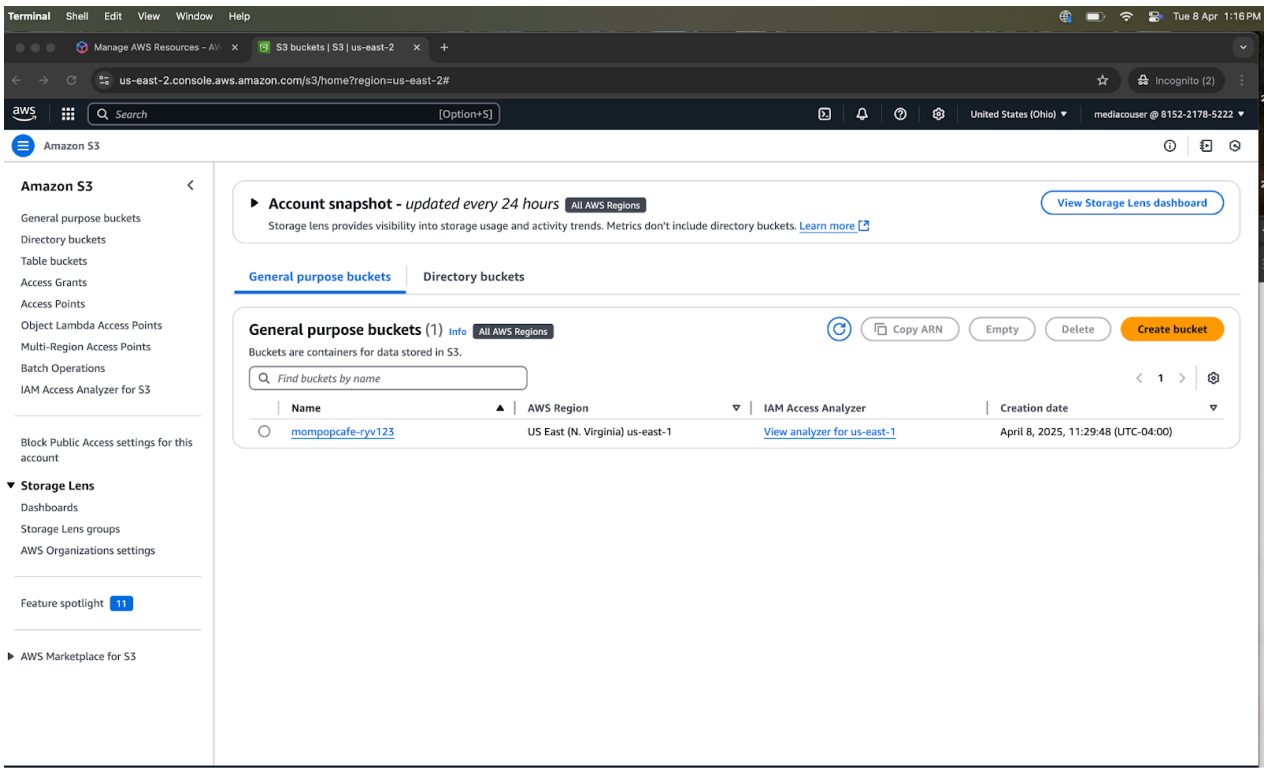
Note: This screenshot shows successful synchronization of local images from the EC2 instance to the S3 bucket mompopcafe-ryv123. The user used aws s3 sync to upload images such as Donuts.jpg, Strawberry-Tarts.jpg, and Cup-of-Hot-Chocolate.jpg under the images/ directory.

Fig 8.4 : S3 Bucket File Listing and Size Verification from EC2 Instance



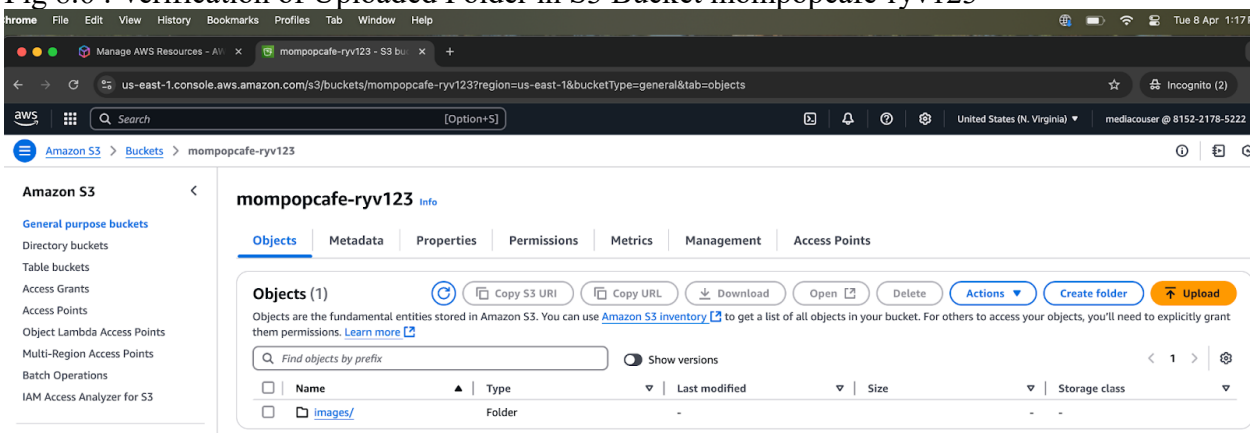
Note: The image illustrates the successful upload of three image files to the S3 bucket mompopcafe-ryv123/images/, followed by listing them using the aws s3 ls command. The total size of the uploaded objects is 1.1 MiB, confirming transfer integrity.

Fig 8.5 : Test the mediacouser permissions



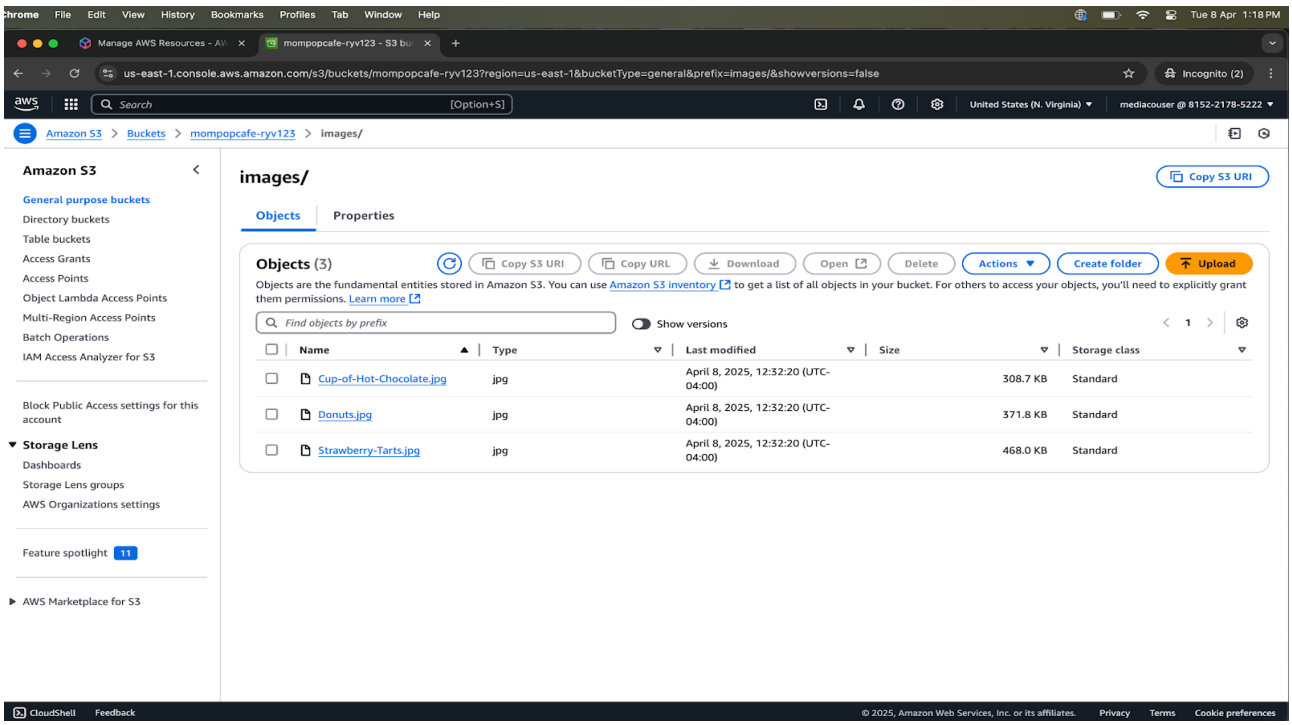
Note: This screenshot confirms the successful creation of the S3 bucket mompopcafe-ryv123 in the AWS Management Console. The bucket is listed under the US East (N. Virginia) region (us-east-1), with the creation timestamp recorded as April 8, 2025.

Fig 8.6 : Verification of Uploaded Folder in S3 Bucket mompopcafe-ryv123



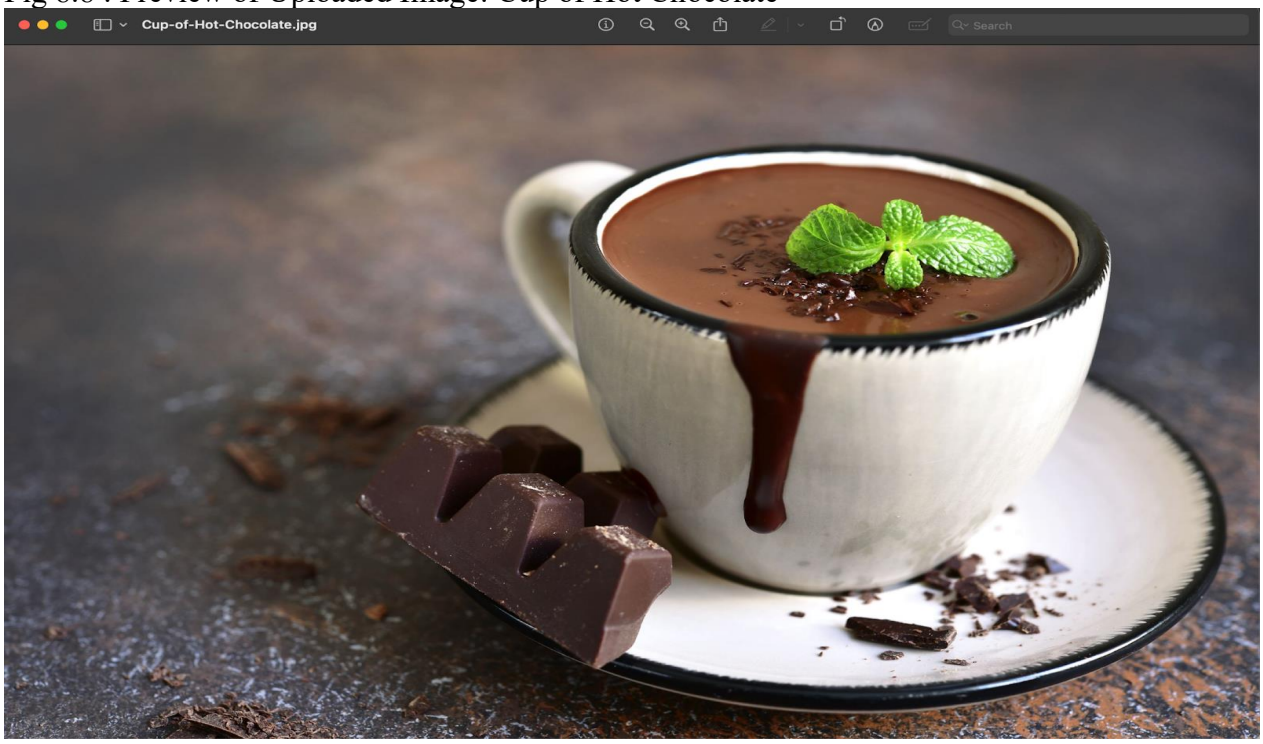
Note: The image shows the S3 console view of the mompopcafe-ryv123 bucket. It confirms that the images/ folder was successfully uploaded, indicating that the mediacouser has sufficient permissions to list and access objects within the bucket.

Fig 8.7 : Confirmation of Image Uploads in S3 Bucket Folder images/



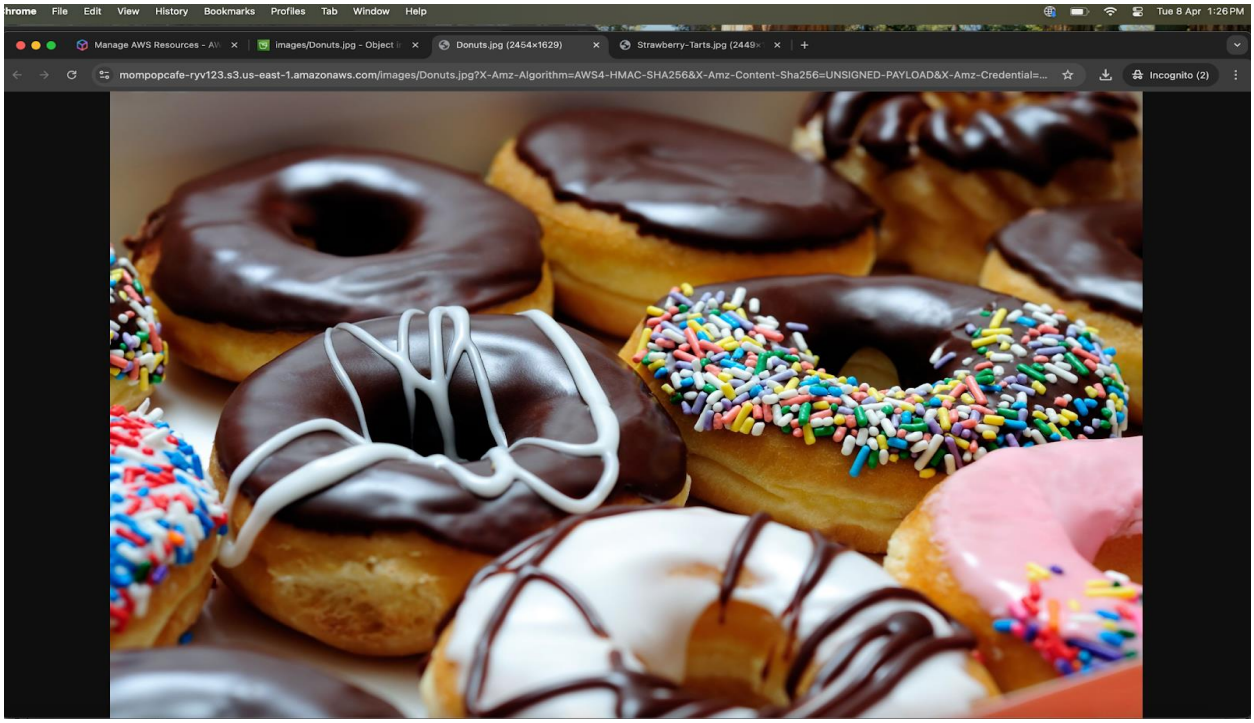
Note: This screenshot verifies that three image files—Cup-of-Hot-Chocolate.jpg, Donuts.jpg, and Strawberry-Tarts.jpg—have been successfully uploaded to the images/ folder in the S3 bucket mompopcafe-ryv123, with accurate timestamps and storage class settings.

Fig 8.8 : Preview of Uploaded Image: Cup of Hot Chocolate



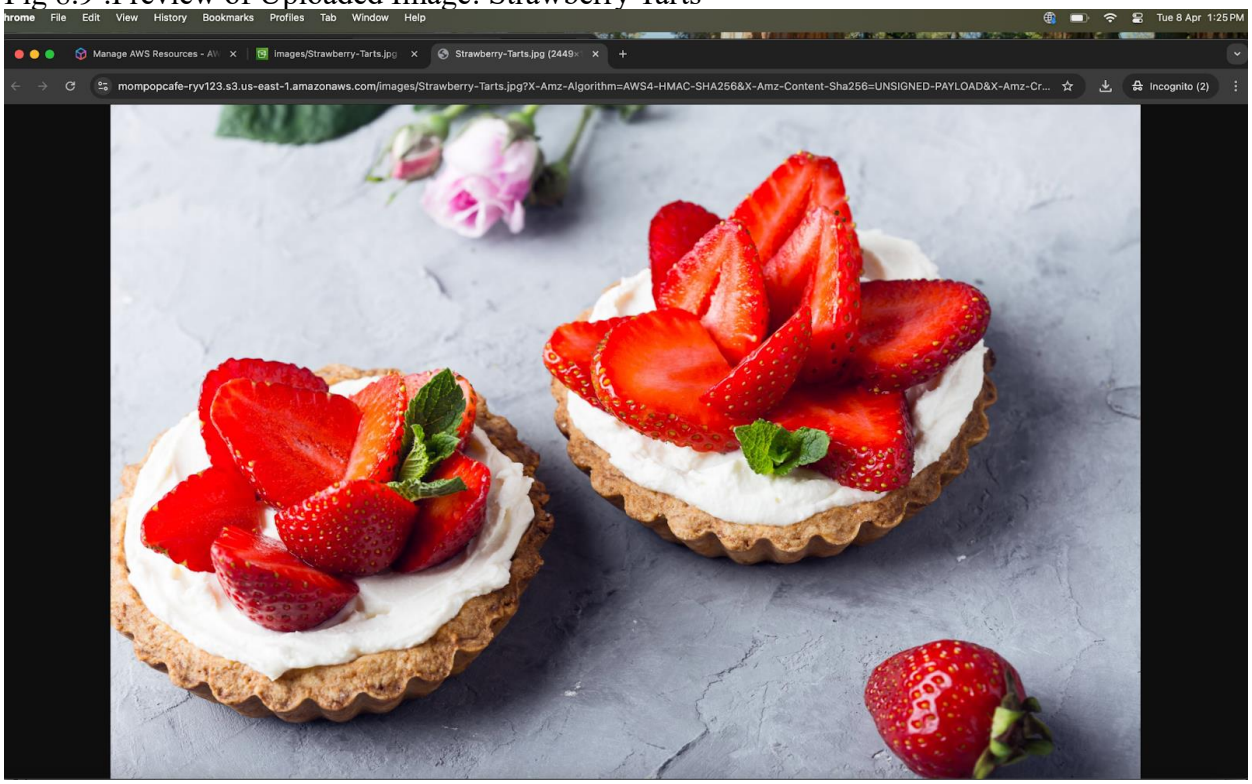
Note: This image, titled Cup-of-Hot-Chocolate.jpg, was successfully uploaded to the S3 bucket mompopcafe-ryv123. It visually confirms the content integrity post-upload, showcasing a rich cup of hot chocolate garnished with mint and chocolate shavings.

Fig 8.8 : Preview of Uploaded Image: Donuts Assortment



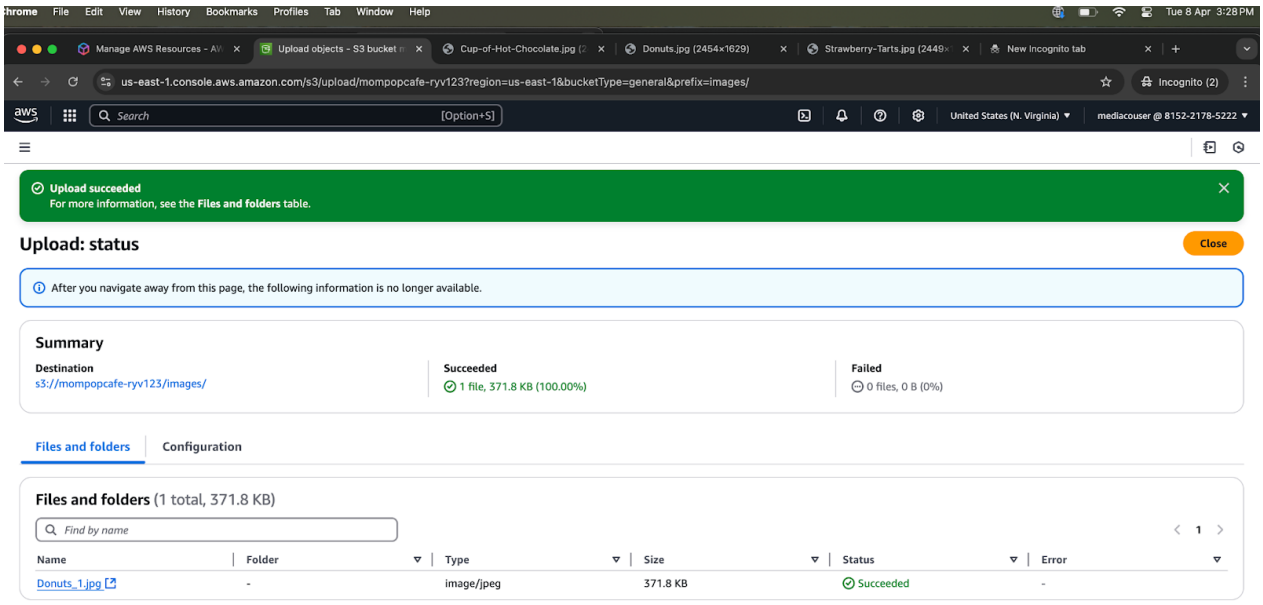
NNote: This image titled Donuts.jpg was uploaded to the S3 bucket mompopcafe-ryv123/images/. It displays a colorful assortment of donuts topped with chocolate glaze, sprinkles, and icing, confirming successful file accessibility and visual integrity after upload.

Fig 8.9 :Preview of Uploaded Image: Strawberry Tarts



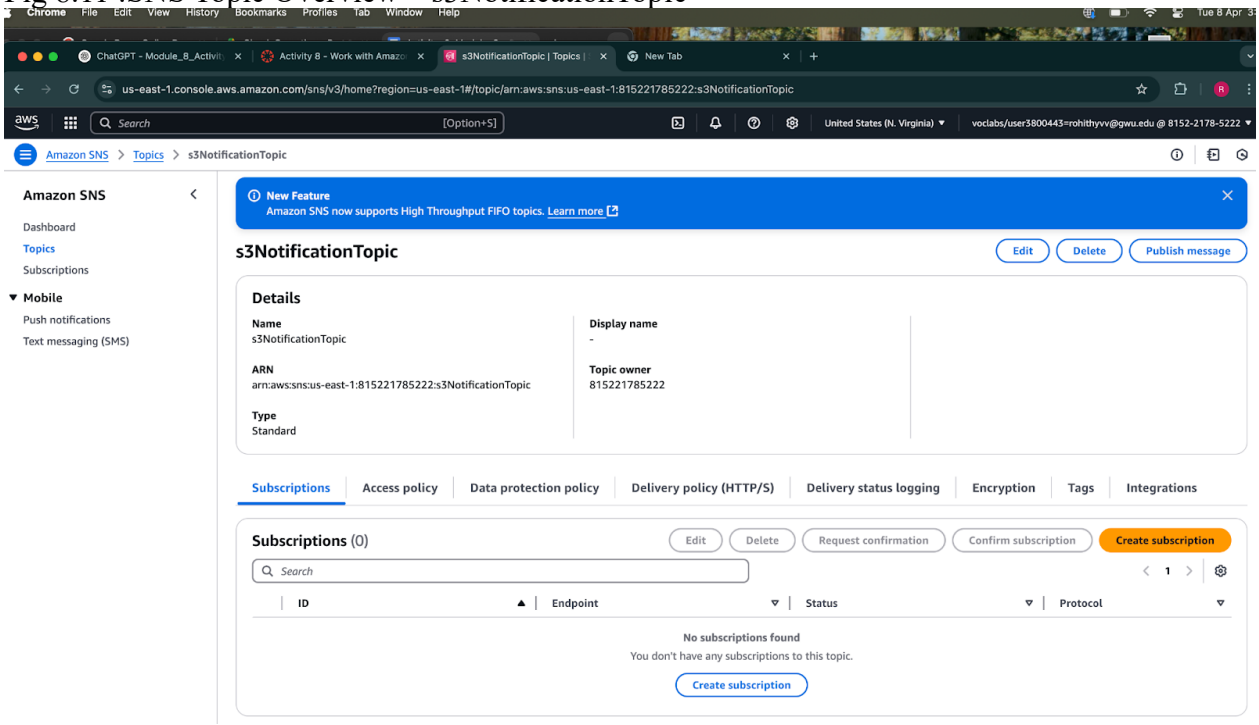
Note: This image, named Strawberry-Tarts.jpg, showcases two freshly prepared tarts topped with whipped cream and sliced strawberries. Successfully uploaded to the mompopcafe-ryv123/images/ S3 folder, it verifies image accessibility and visual quality after upload to the cloud.

Fig 8.10 :Successful Upload Confirmation of Donuts_1.jpg



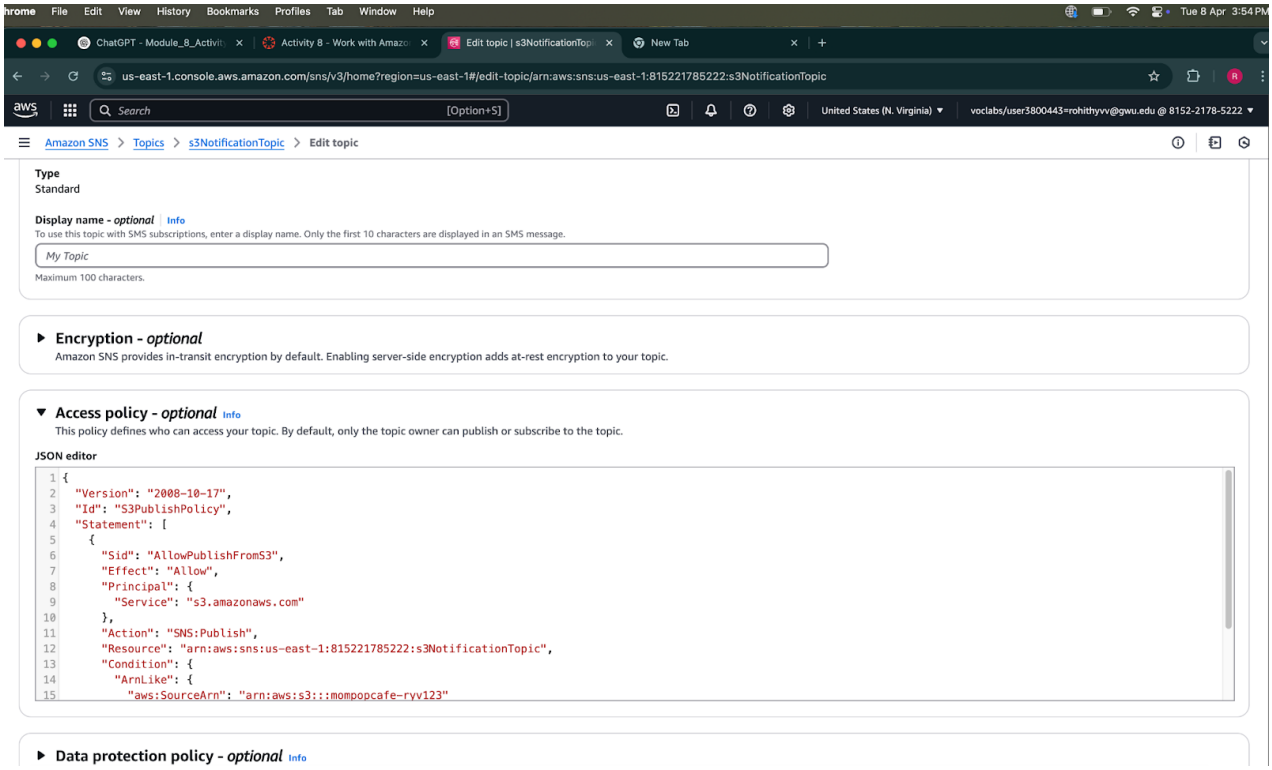
Note: This screenshot confirms the successful upload of Donuts_1.jpg to the S3 bucket mompopcafe-ryv123/images/. The file uploaded without errors, achieving 100% success. The image size is 371.8 KB, and the confirmation assures data integrity post-upload.

Fig 8.11 :SNS Topic Overview – s3NotificationTopic



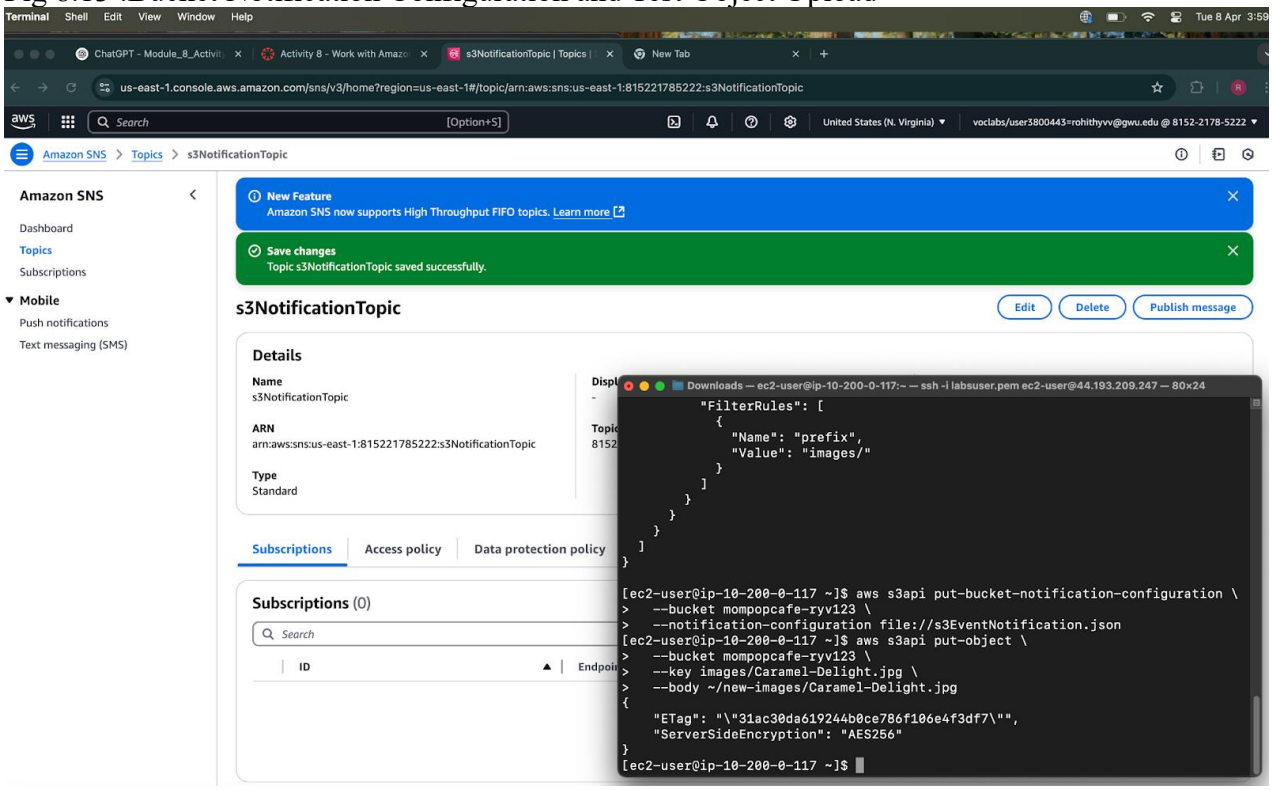
Note: This image shows the SNS topic s3NotificationTopic created to receive S3 event notifications. It includes details like ARN, topic type, and owner ID, and confirms that no subscriptions have yet been added to the topic.

Fig 8.12 :SNS Topic Access Policy for S3 Event Publishing



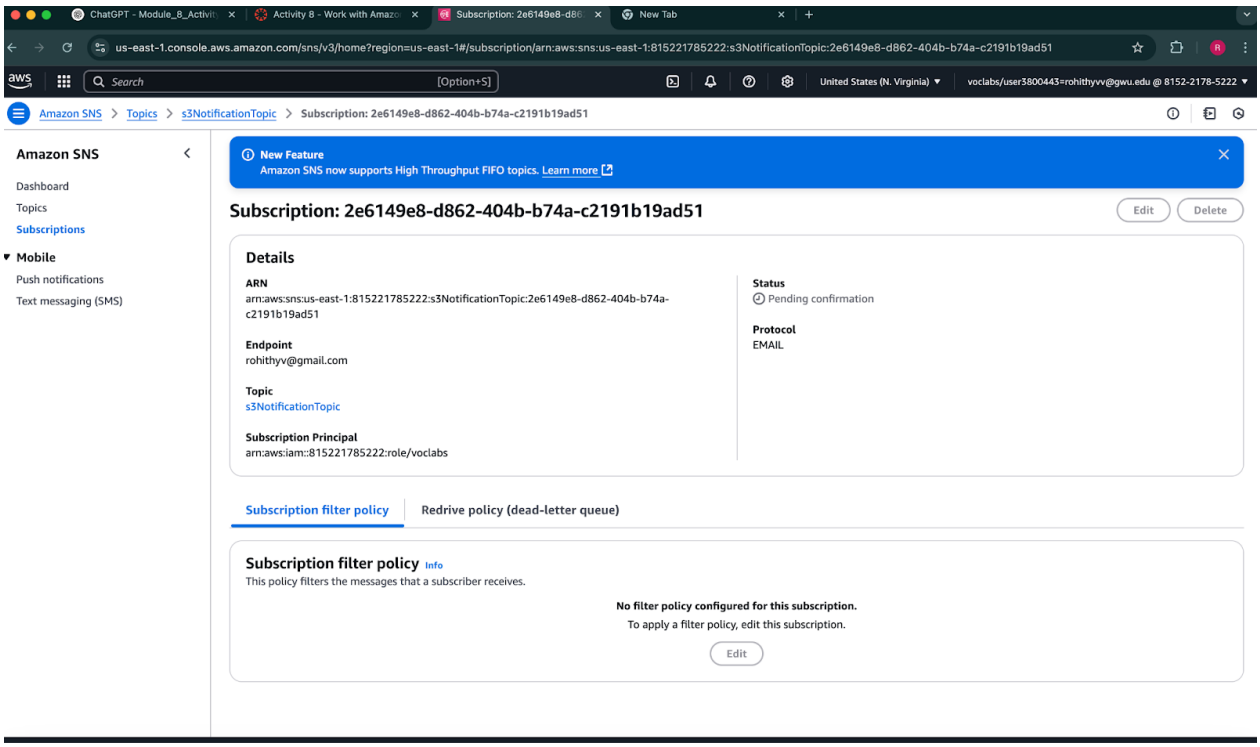
Note: This screen captures the custom access policy configured to allow the S3 service to publish notifications to the SNS topic. The policy grants S3:Publish permission specifically from the mompopcafe-ryv123 bucket using aws:SourceArn

Fig 8.13 :Bucket Notification Configuration and Test Object Upload



Note: This composite image confirms the setup of an S3 bucket notification for prefix images/, followed by a successful object upload (Caramel-Delight.jpg) using the aws s3api put-object command, verifying notification trigger behavior with encryption metadata.

Fig 8.14 :Pending Email Subscription Confirmation for SNS Topic

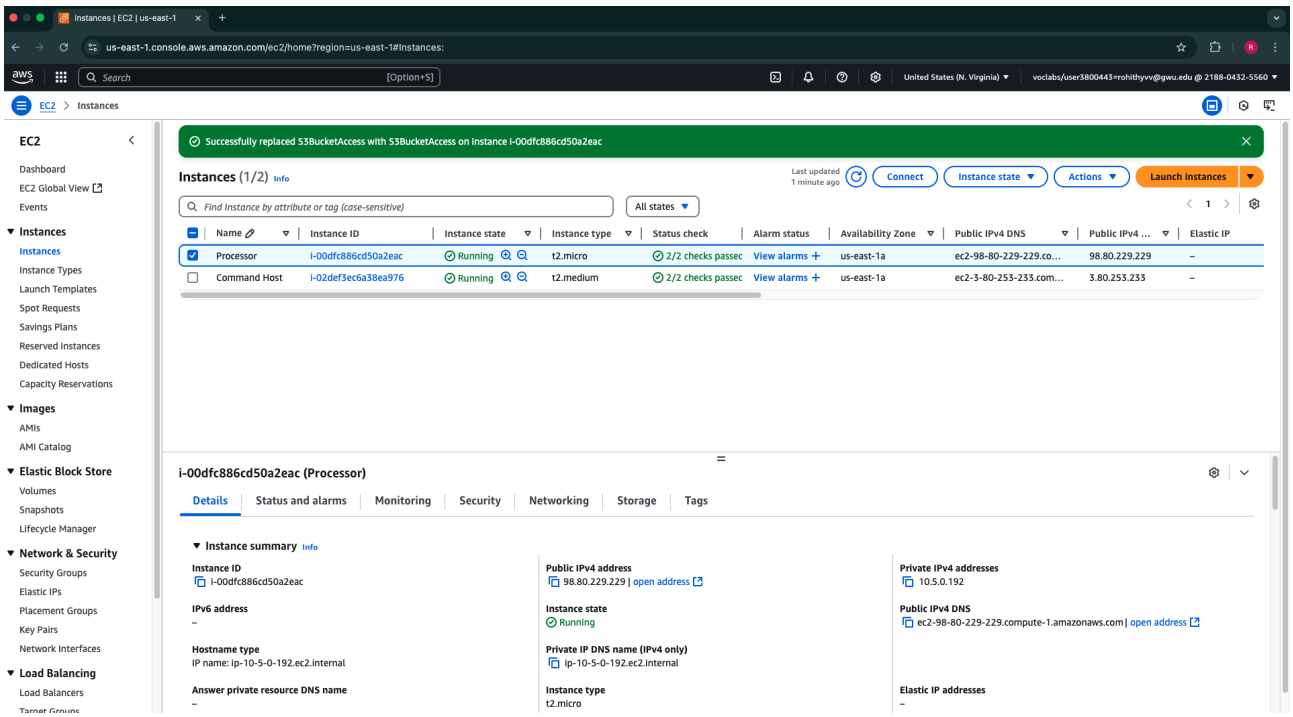


Note: This image shows a pending SNS subscription created using the email protocol. The endpoint rohithyv@gmail.com awaits confirmation to begin receiving notifications from the s3NotificationTopic. The subscription is linked to the IAM role voclabs.

Module 8: Storing and Archiving

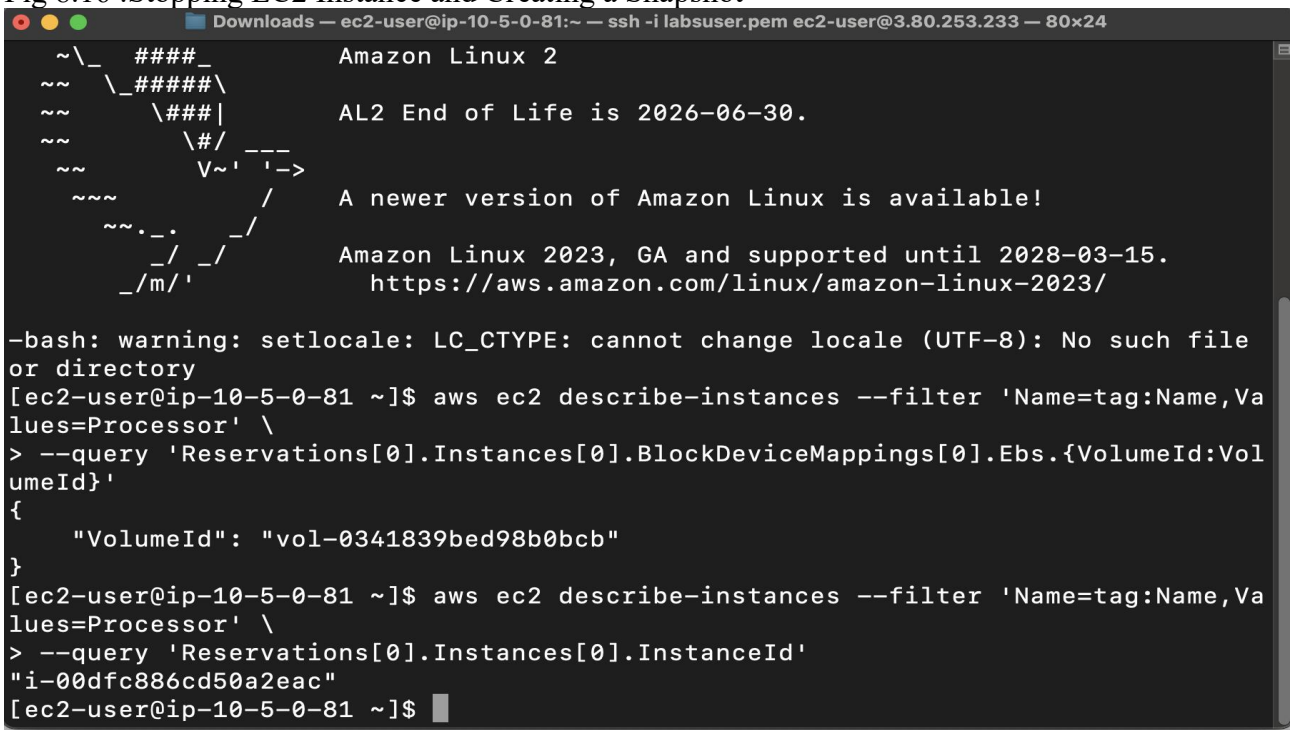
Lab 5 Managing Storage

Fig 8.15 :Identifying EC2 Volume and Instance for Backup



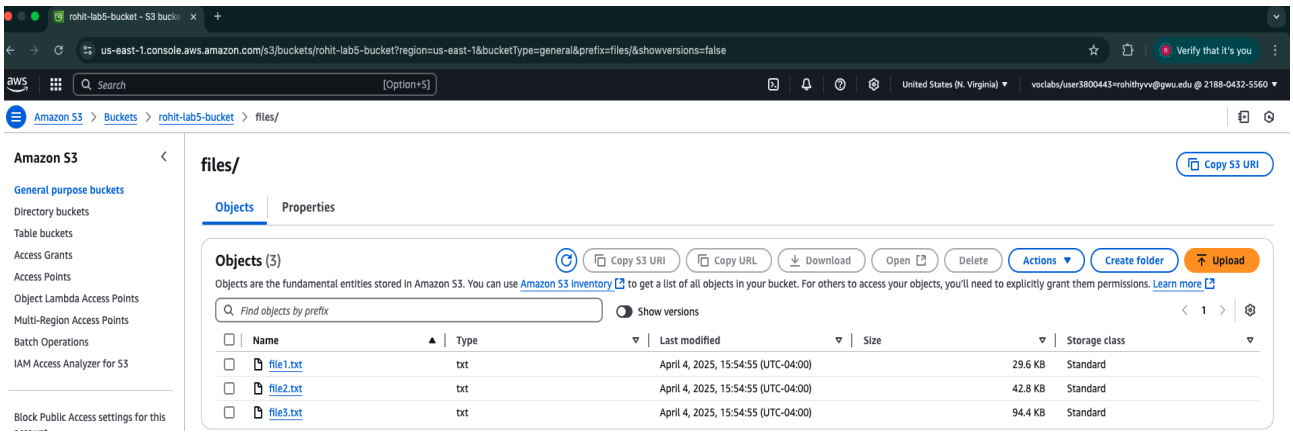
Note. This screenshot shows the use of AWS CLI to identify the Volume ID and Instance ID of an EC2 instance tagged as "Processor." These identifiers are essential for accurate snapshot creation.

Fig 8.16 :Stopping EC2 Instance and Creating a Snapshot



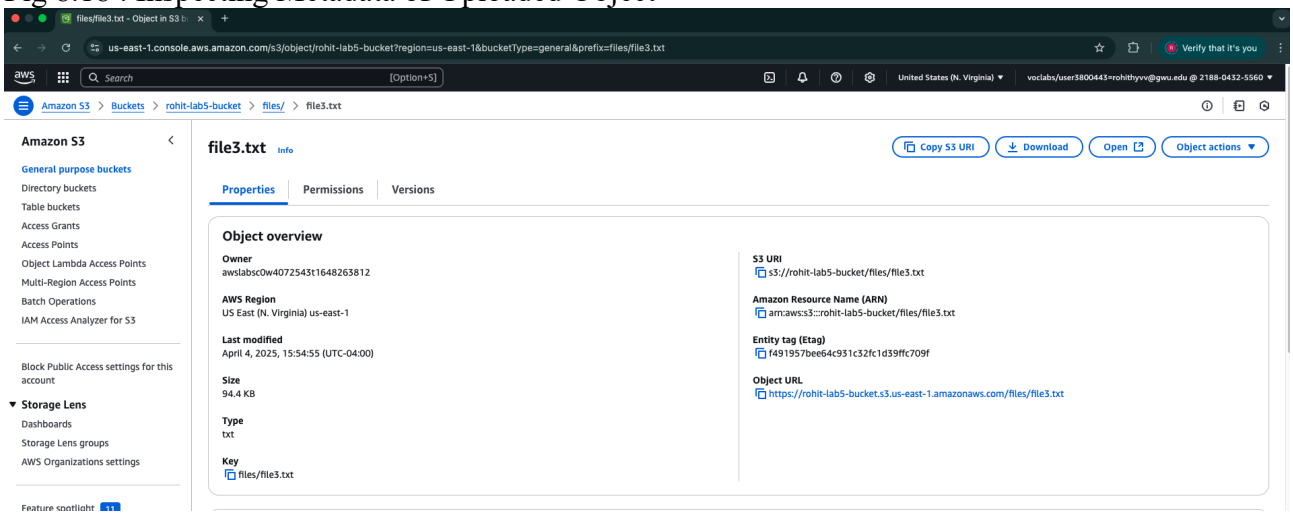
Note: Note. After the snapshot is confirmed, the EC2 instance is restarted to resume service. This image reflects how AWS CLI confirms the instance's transition from stopped to running state.

Fig 8.17 :Listing All S3 Buckets in Console



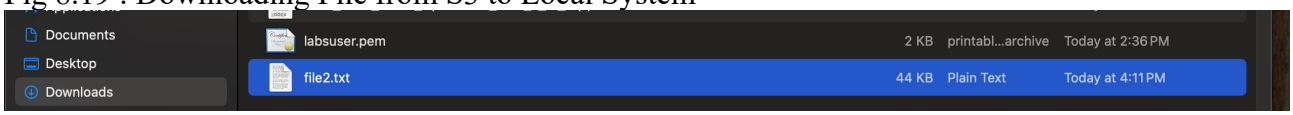
Note. The AWS S3 Console lists all created buckets. This confirms the presence and region of the lab bucket used for file operations and version control.

Fig 8.18 : Inspecting Metadata of Uploaded Object



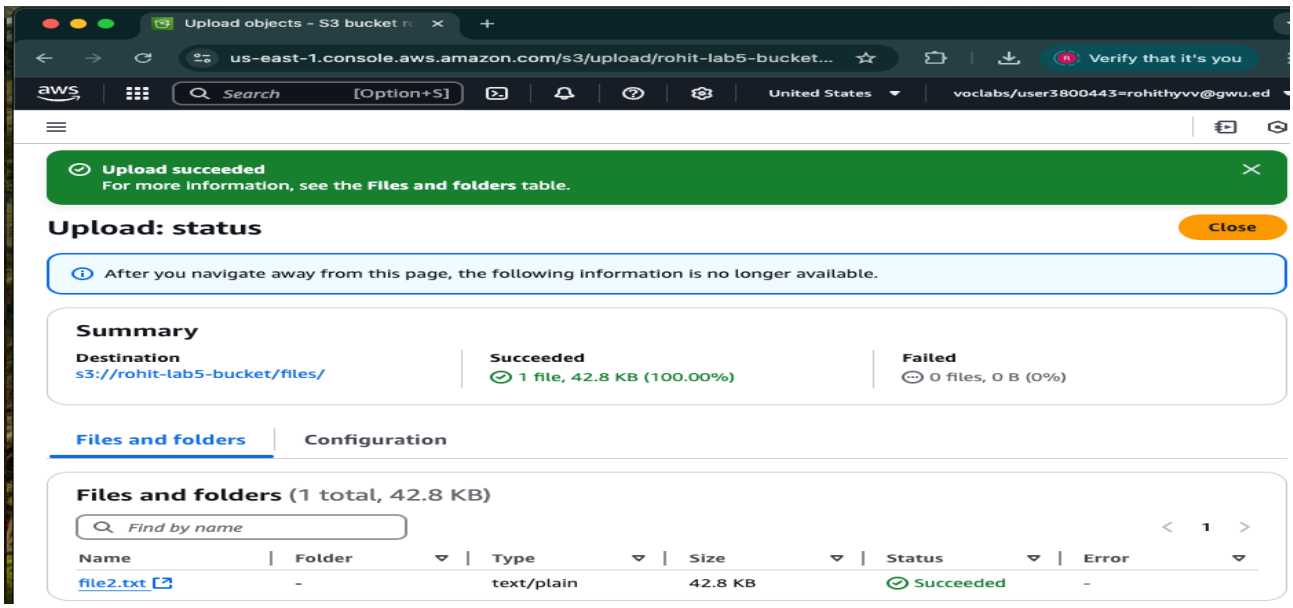
Note. Metadata view of file3.txt in the AWS Console. Shows S3 URI, object URL, ETag, and modification timestamps for referencing and API access.

Fig 8.19 : Downloading File from S3 to Local System



Note. Shows successful upload of file2.txt back into the bucket. This reflects how overwriting a file creates a new version in versioned S3 buckets.

Fig 8.20 : Uploading File from Local System



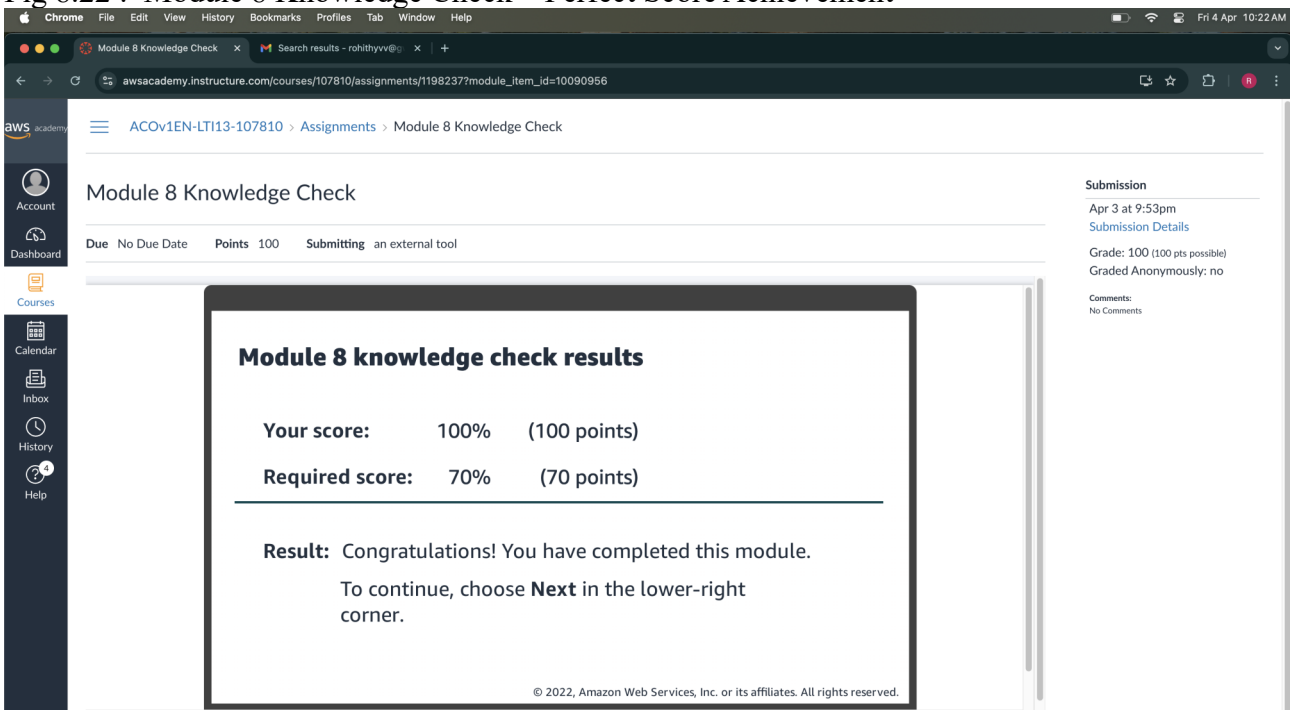
Note. Shows successful upload of file2.txt back into the bucket. This reflects how overwriting a file creates a new version in versioned S3 buckets.

Fig 8.21 : Listing S3 Objects Using AWS CLI

```
[ec2-user@ip-10-5-0-81 ~]$ aws s3 ls s3://rohit-lab5-bucket/files/
2025-04-04 20:14:27      43784 file2.txt
2025-04-04 19:54:55      96675 file3.txt
[ec2-user@ip-10-5-0-81 ~]$
```

Note. AWS CLI lists current files in the S3 bucket. Verifies programmatic access matches UI state and confirms the synchronization of object management.

Fig 8.22 : Module 8 Knowledge Check – Perfect Score Achievement



Note: The image confirms successful completion of the Module 8 Knowledge Check with a perfect score of 100%. The user exceeded the passing requirement of 70%, demonstrating full mastery of the content.

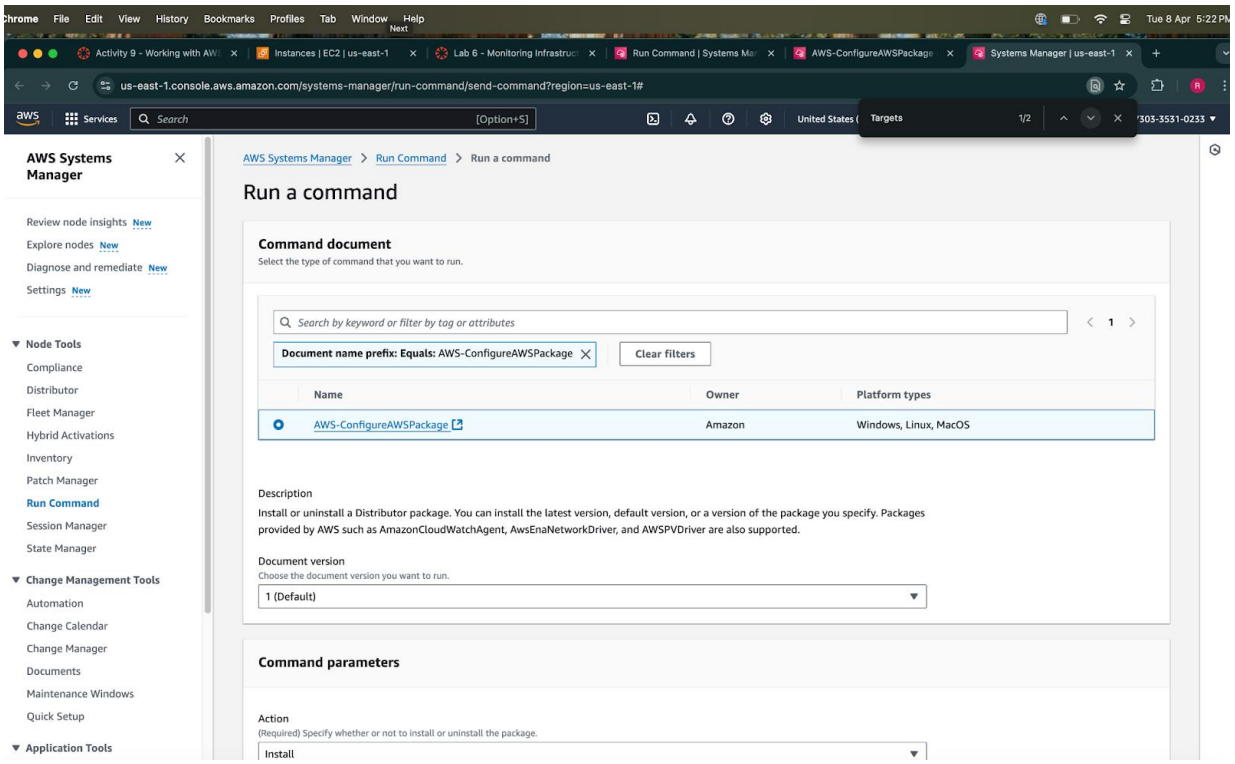
Summary:

- Amazon EBS Snapshots are incremental backups of EBS volumes, stored in Amazon S3, and only save changes after the first snapshot, helping reduce storage costs and improving backup efficiency.
- Amazon S3 is a scalable object storage service used for storing and accessing data online, offering different storage classes like Standard, Intelligent-Tiering, Infrequent Access, and Glacier for varying access needs.
- Instance Store provides temporary block-level storage physically attached to the host, ideal for fast-changing data like caches, buffers, and scratch space, but is not persistent.
- AWS Data Migration Tools such as AWS DataSync, Snowball, and Transfer for SFTP help move large volumes of data securely and efficiently between on-premises systems and AWS storage services.

Module 9: Monitoring and Security

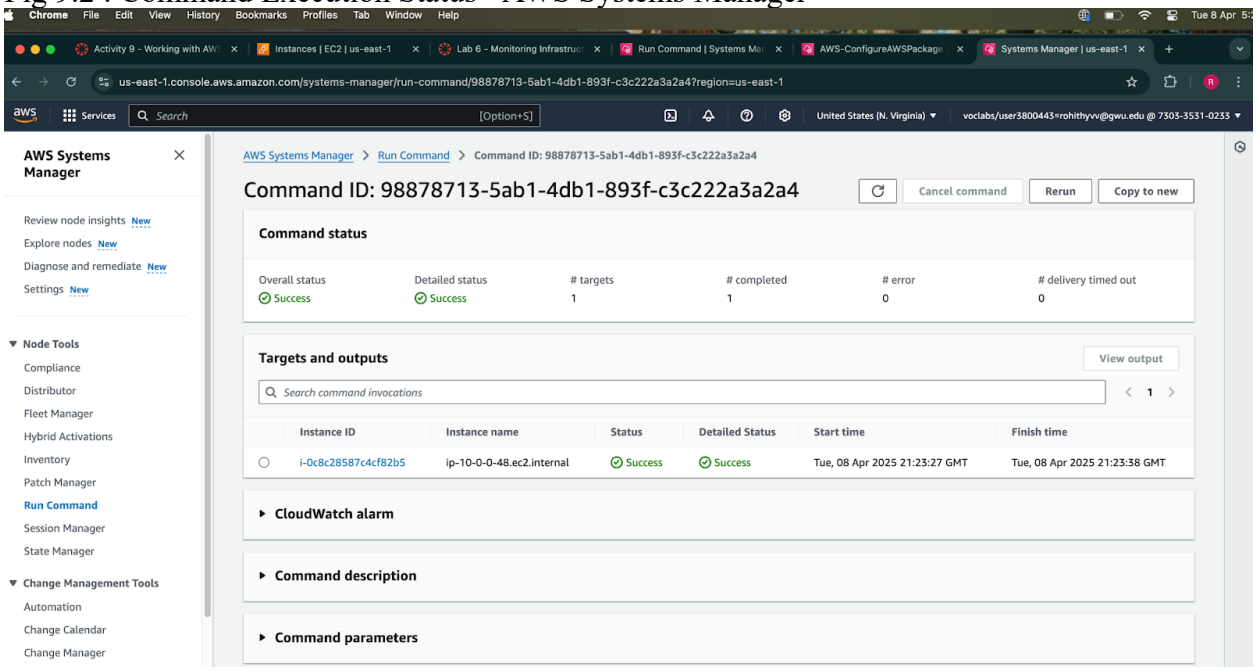
Lab 6 : Monitoring Infrastructure

Fig 9.1 : CloudWatch Agent on the Web Server



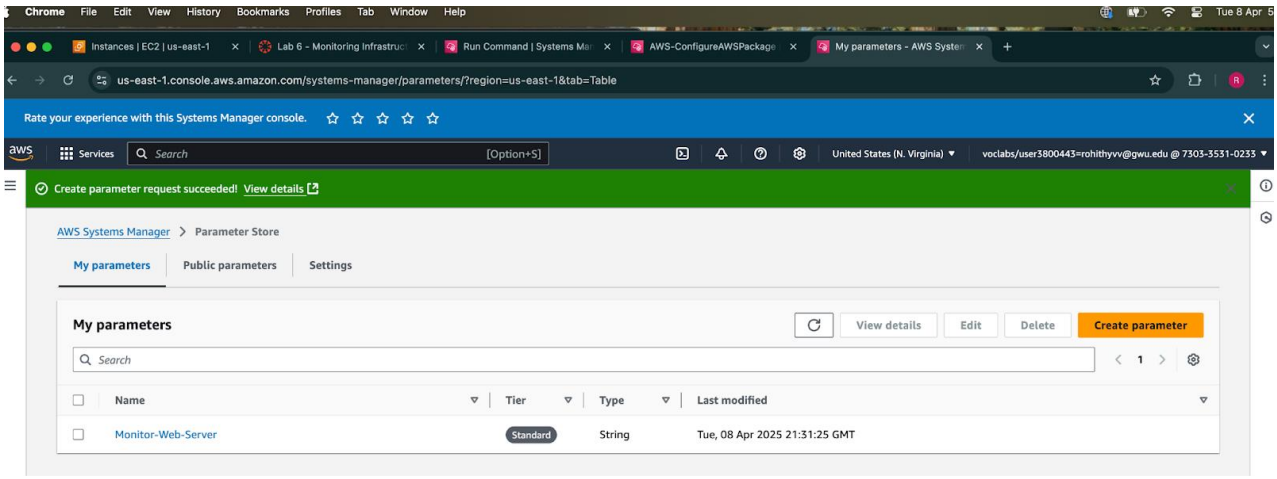
This screenshot shows the initiation of a Run Command using AWS-ConfigureAWSPackage to install or uninstall a Distributor package. The document targets EC2 instances to configure packages such as CloudWatch Agent, helping streamline monitoring or software configuration tasks.

Fig 9.2 : Command Execution Status - AWS Systems Manager



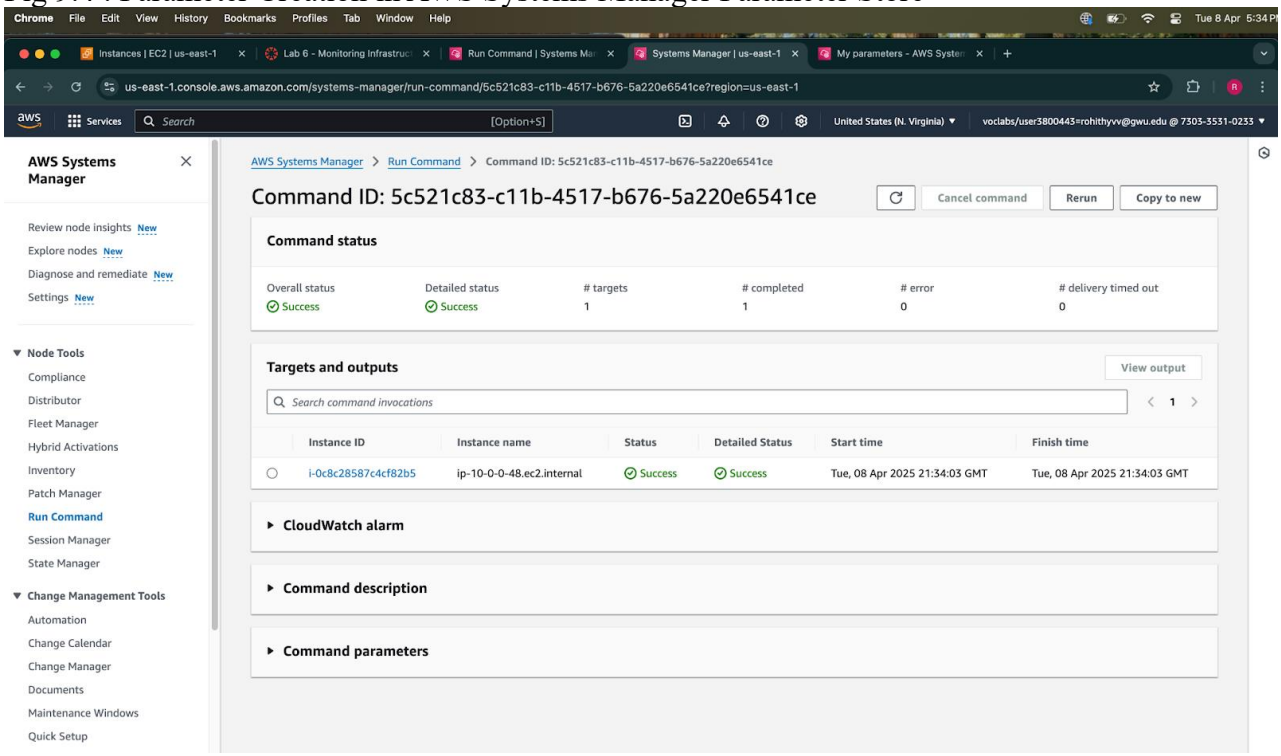
Note: The command AWS-ConfigureAWSPackage was successfully executed on the target EC2 instance. Both the overall and detailed status show success, confirming that the package (likely CloudWatch Agent) was correctly installed without any errors or delivery timeouts.

Fig 9.3 : Command Execution Status - AWS Systems Manager



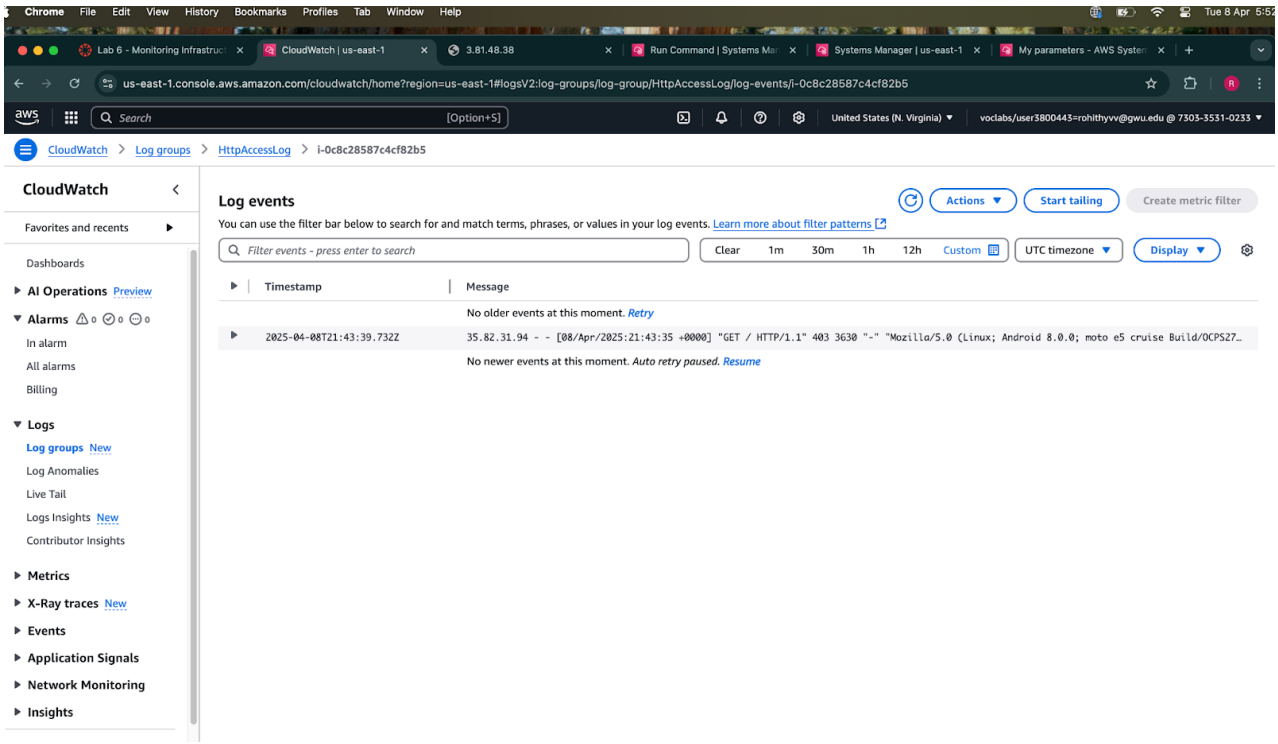
Note: The parameter Monitor-Web-Server was created successfully under the Parameter Store. This standard string parameter can be used for storing configuration values securely and centrally to manage and reference in automation tasks like EC2 monitoring or script execution.

Fig 9.4 : Parameter Creation in AWS Systems Manager Parameter Store



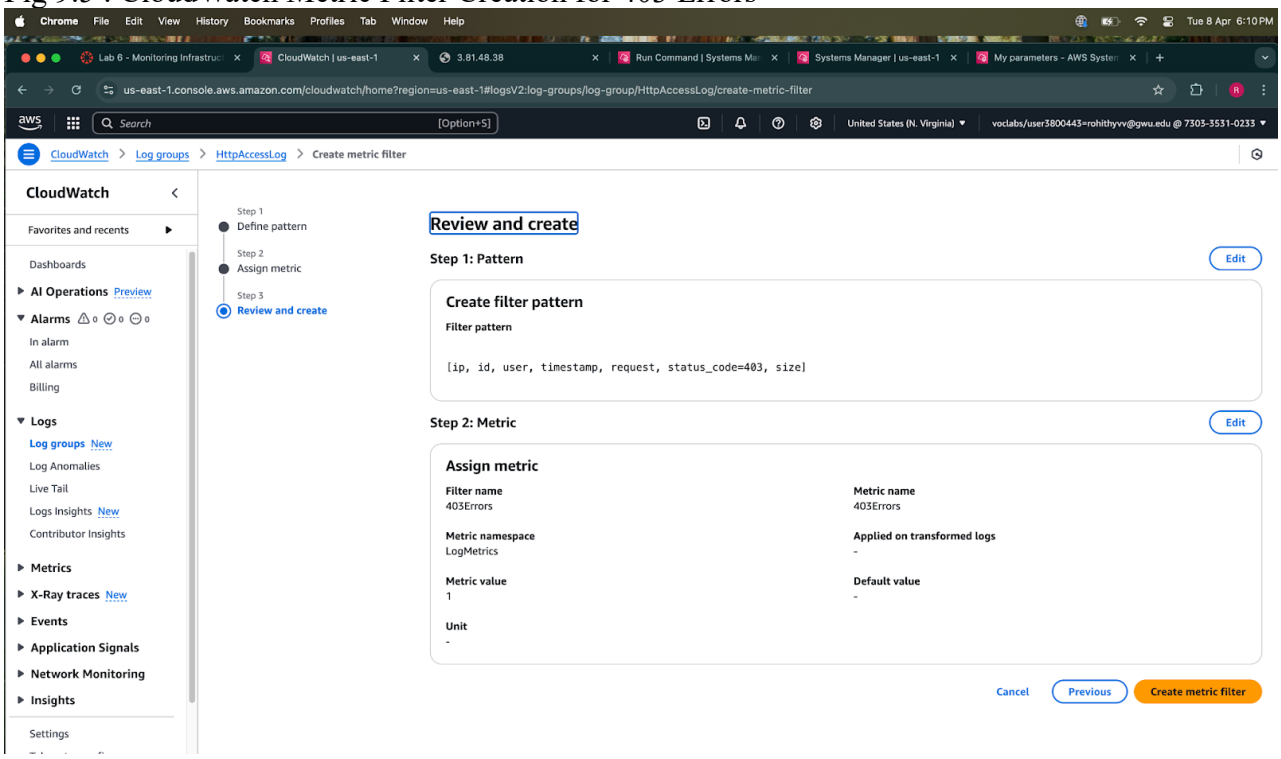
Note: The parameter Monitor-Web-Server was created successfully under the Parameter Store. This standard string parameter can be used for storing configuration values securely and centrally to manage and reference in automation tasks like EC2 monitoring or script execution.

Fig 9.5 :CloudWatch Log Event Showing 403 Error



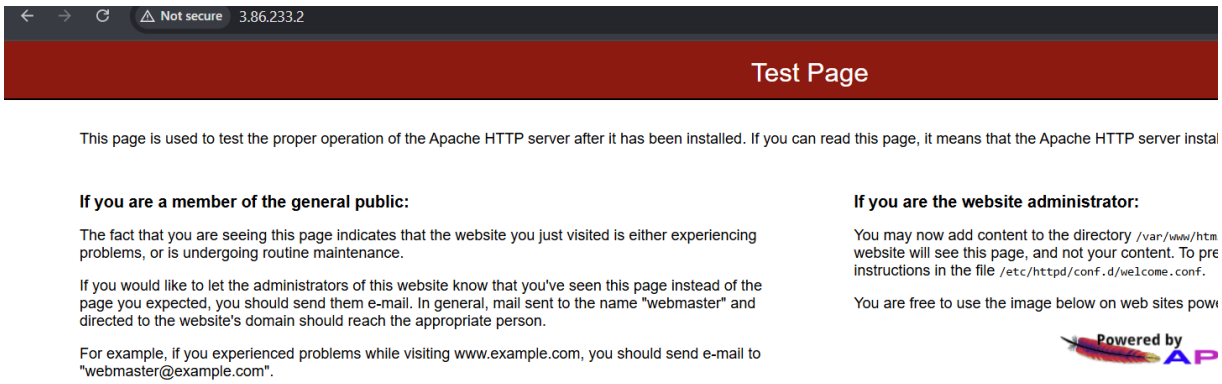
Note:This CloudWatch log entry shows a 403 Forbidden error response captured in the log group HttpAccessLog. The request was denied, possibly due to permission issues, and the log confirms monitoring is active and capturing HTTP access errors.

Fig 9.5 : CloudWatch Metric Filter Creation for 403 Errors



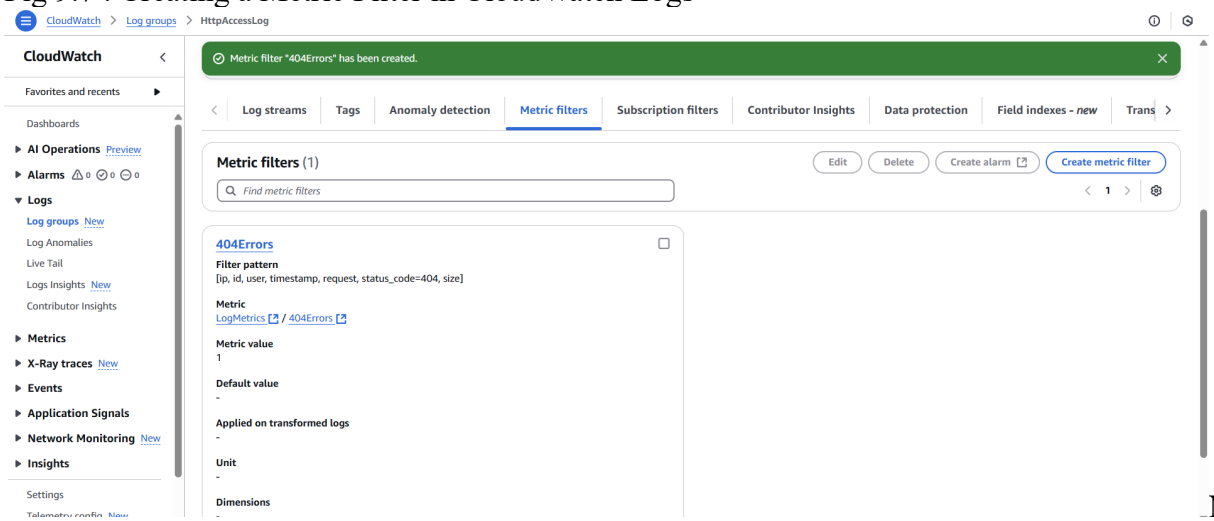
Note: This screen finalizes the creation of a CloudWatch metric filter for detecting HTTP 403 errors. The filter pattern extracts logs with status_code=403, assigning them to the metric 403Errors in namespace LogMetrics, enabling alarm triggers or monitoring dashboards.

Fig 9.6: Web Server Test Page



Note: we paste the WebServerIP copied in the browser to see a web server Test Page, now will generate log data by attempting to access a page that does not exist.

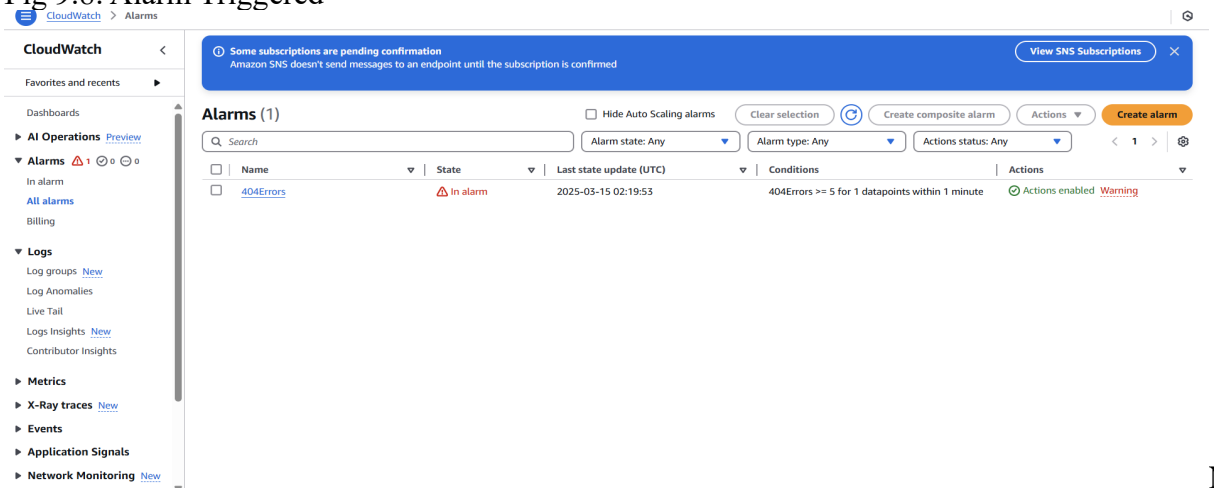
Fig 9.7 : Creating a Metric Filter in CloudWatch Logs



Note :

Note: We will configure a Filter to identify 404 Errors in the log file. This would normally be an indication that the web server is generating invalid links that users are clicking.

Fig 9.8: Alarm Triggered

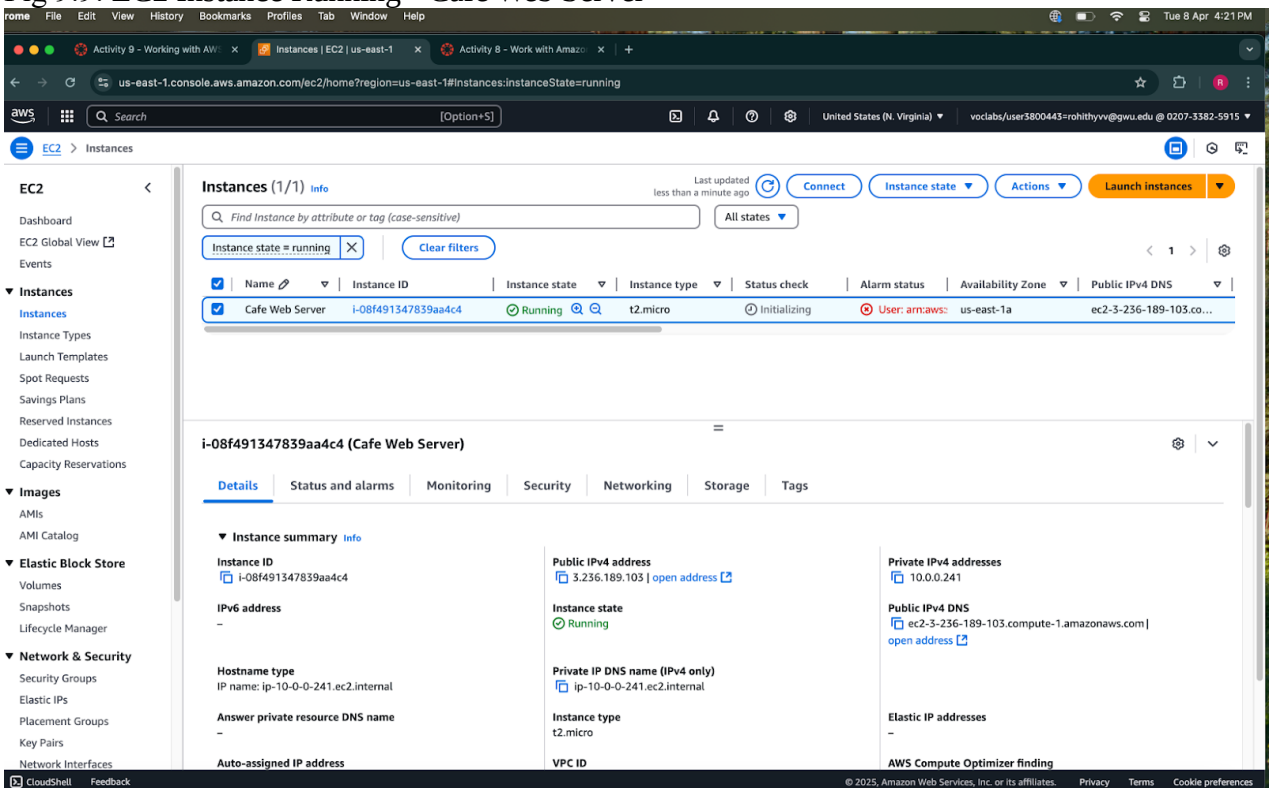


Note:

Note; Attempting to go to pages that do not exist by adding a page name after the IP address, graph shown on the CloudWatch page should turn red to indicate that it is in the ALARM state.

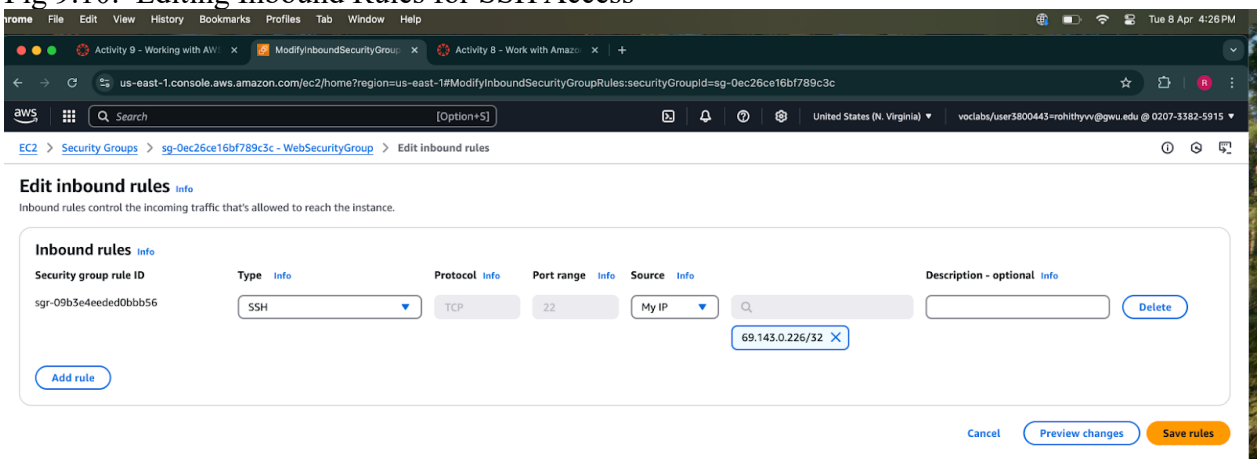
Activity 9 : Working with AWS CloudTrail

Fig 9.9: EC2 Instance Running - Cafe Web Server



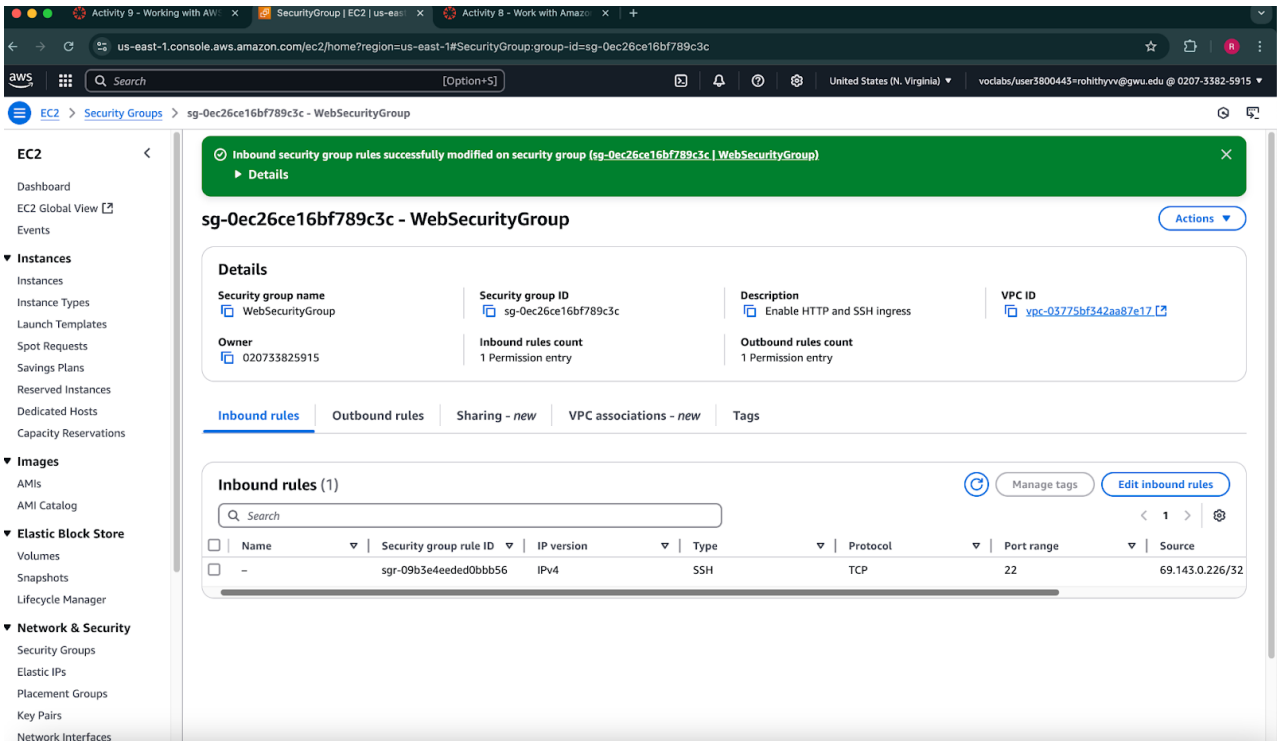
Note: The EC2 instance named "Cafe Web Server" is in a running state in the `us-east-1a` availability zone. It's a t2.micro instance with a public IPv4 address, allowing you to host web services accessible via the internet.

Fig 9.10: Editing Inbound Rules for SSH Access



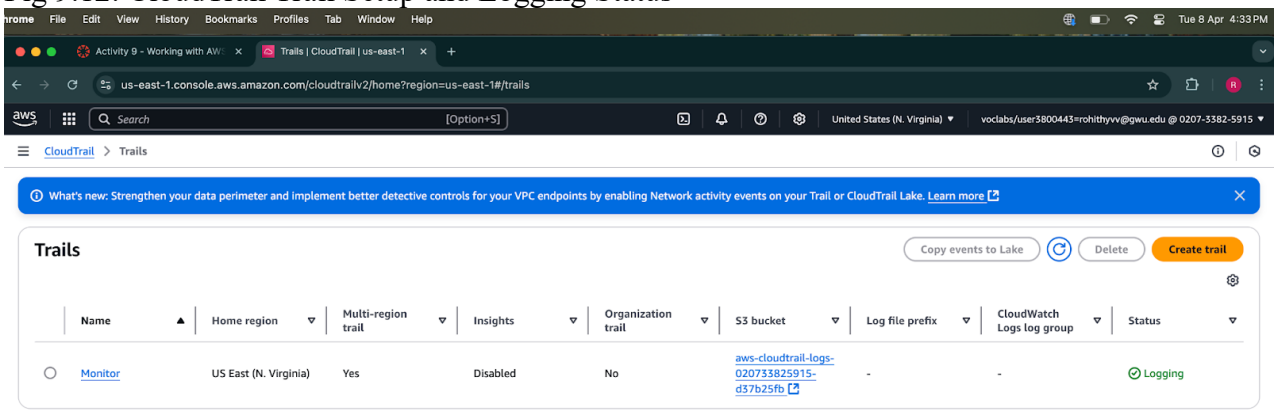
Note: This screen shows modification of a security group to allow SSH access. The rule allows TCP traffic on port 22 from a specific IP address (69.143.0.226/32), securing the instance by limiting remote access to only the user's machine.

Fig 9.11: Security Group Rule Successfully Modified



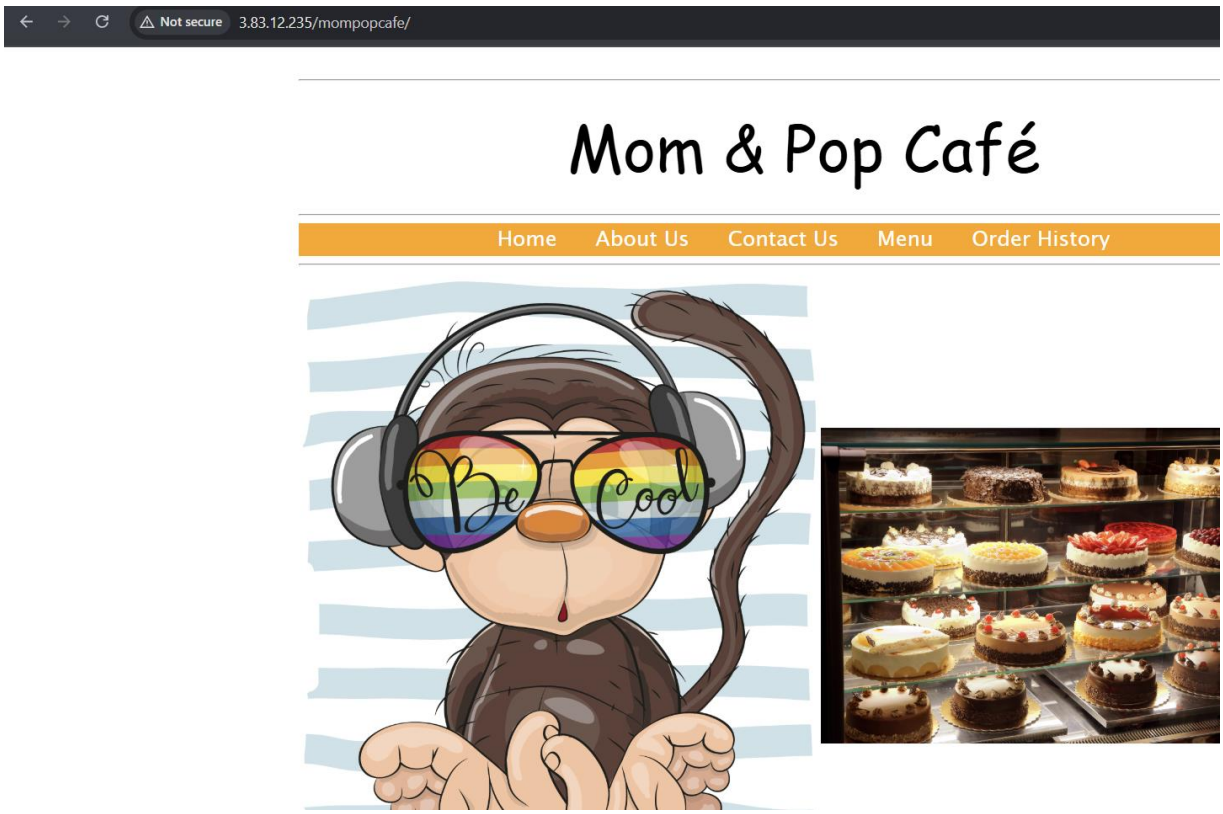
Note: Inbound rules for the security group `WebSecurityGroup` were successfully updated. SSH access (TCP port 22) is now restricted to a single IP, enhancing security by preventing unrestricted remote login attempts from unauthorized networks.

Fig 9.12: CloudTrail Trail Setup and Logging Status



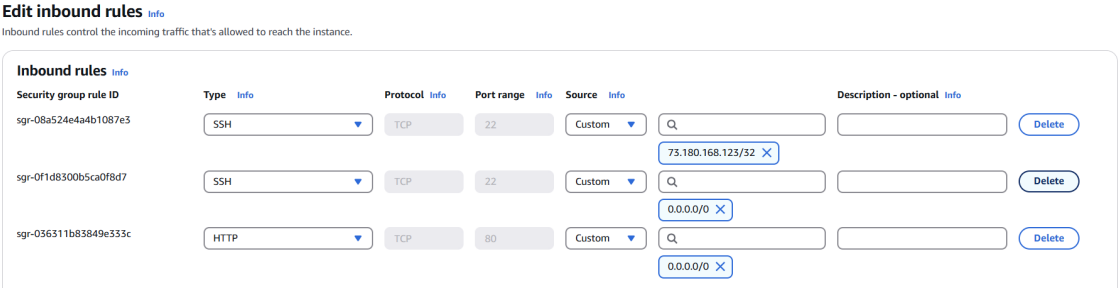
Note: A CloudTrail trail named "Monitor" is configured to capture account activity in the US East (N. Virginia) region. The trail is logging successfully and is set up as a multi-region trail, storing logs in the specified S3 bucket.

Fig 9.13: Website has been hacked with an random image is uploaded in the webpage of mompopcafe



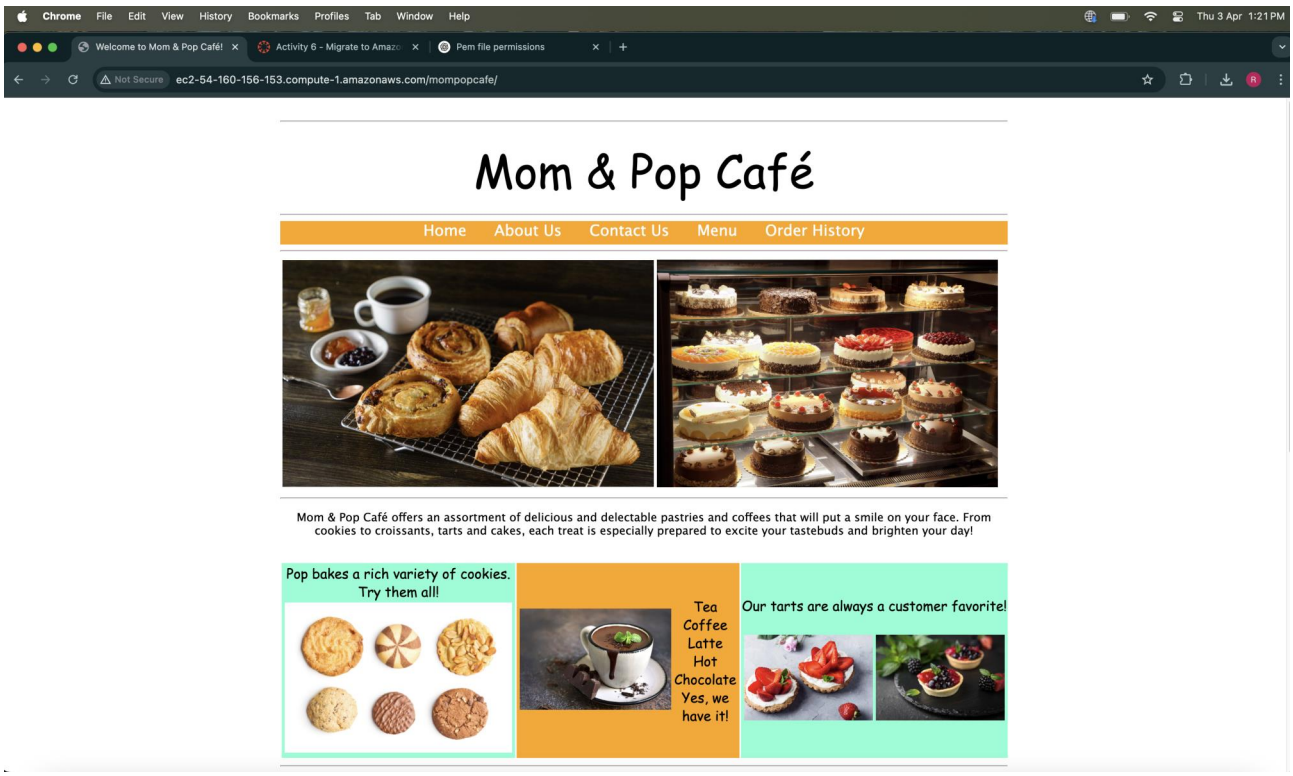
Note: Notice that the website has been hacked. As we enabled CloudTrail before this happened, CloudTrail can give valuable information about what users have been doing in my account.

Fig 9.14 : Deleting inbound rule which hacker created



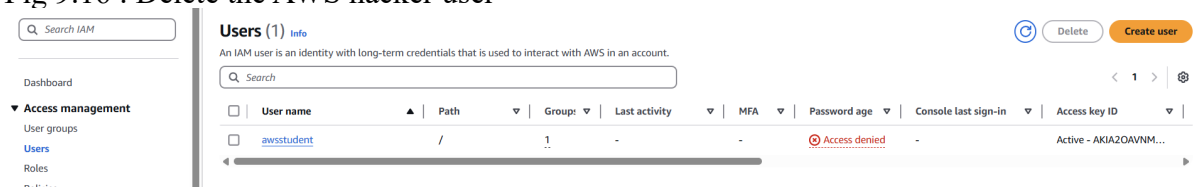
Note: With the WebSecurityGroup selected, in **Inbound rules** should delete the inbound rule that allows port 22 access from 0.0.0.0/0 (the one the hacker created)

Fig 9.15 : After deleting the image and fixing the website



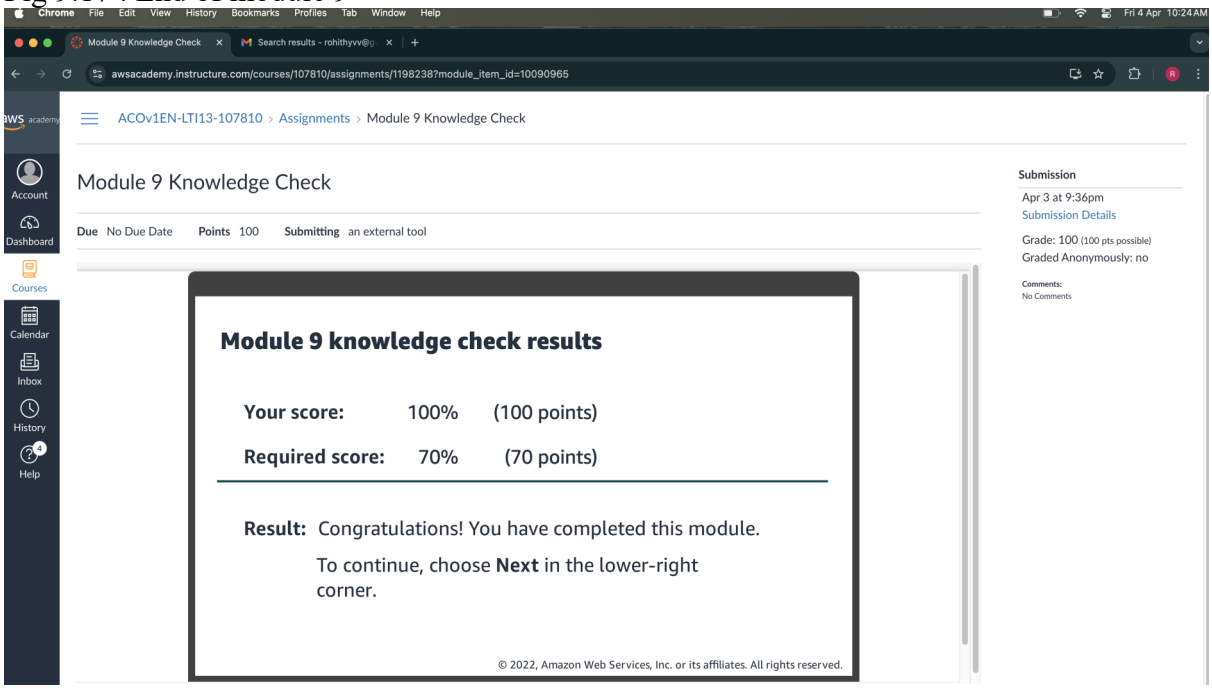
Note: After deleting the inbound rule rule which hacker created and corrupt files uploaded by him, when we refresh our mompopcafe website it should look fine.

Fig 9.16 : Delete the AWS hacker user



Note: The hacker not only accessed the EC2 instance hosting the website, but they also managed to run an AWS CLI command that opened port 22 in the security group to the entire internet. In this step we removed the chaos IAM user from the account.

Fig 9.17 : End of module 9



Note: This image shows the completion of module 9

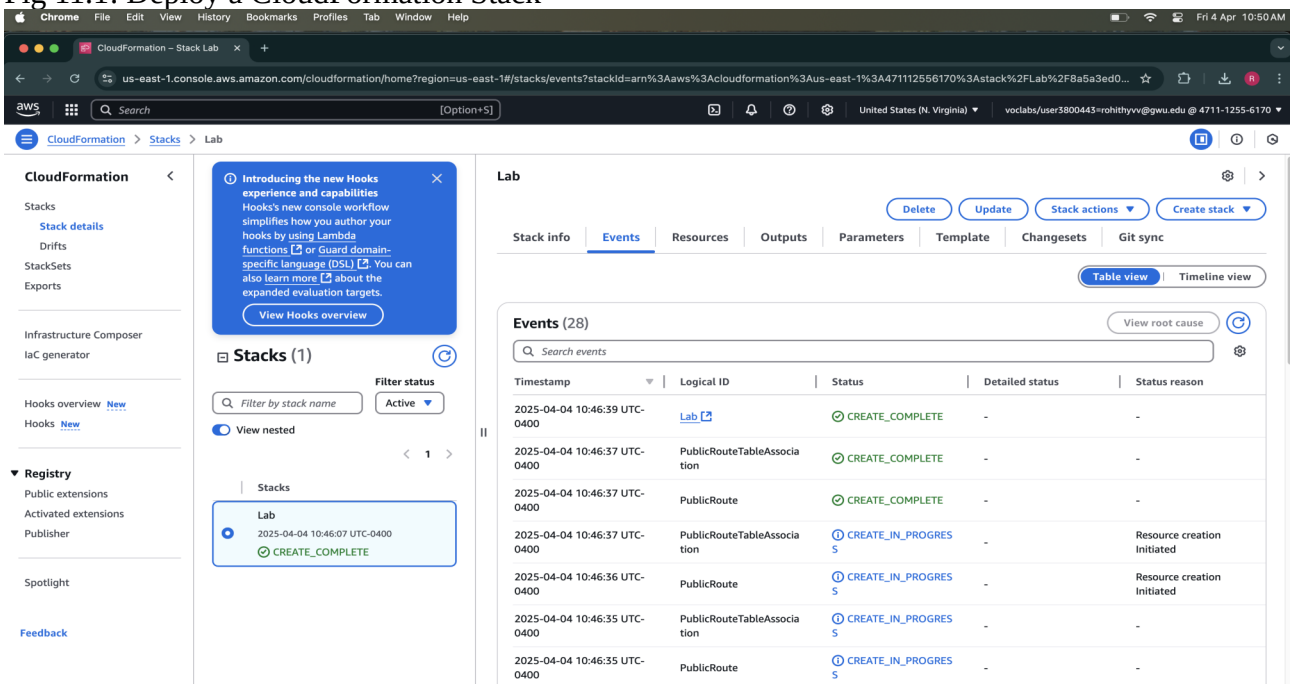
Summary:

- Amazon CloudWatch provides real-time monitoring of AWS resources by collecting metrics, logs, and events. It supports alarms and automated actions based on thresholds or scheduled rules.
- CloudWatch Logs help collect and store log data from AWS services, EC2 instances, and custom sources, allowing for search, alerting, and troubleshooting.
- AWS CloudTrail records all account activity related to actions taken through the AWS Management Console, SDKs, CLI, and other services, storing the logs in S3 for auditing and compliance purposes.
- Amazon Athena allows you to run SQL queries directly on data stored in Amazon S3, making it useful for analyzing service logs like CloudTrail and VPC Flow Logs without data movement.
- AWS Config continuously records resource configurations and evaluates them against desired states, supporting compliance auditing, change tracking, and security analysis.

Module 11 : Creating Automated and Repeatable Deployments

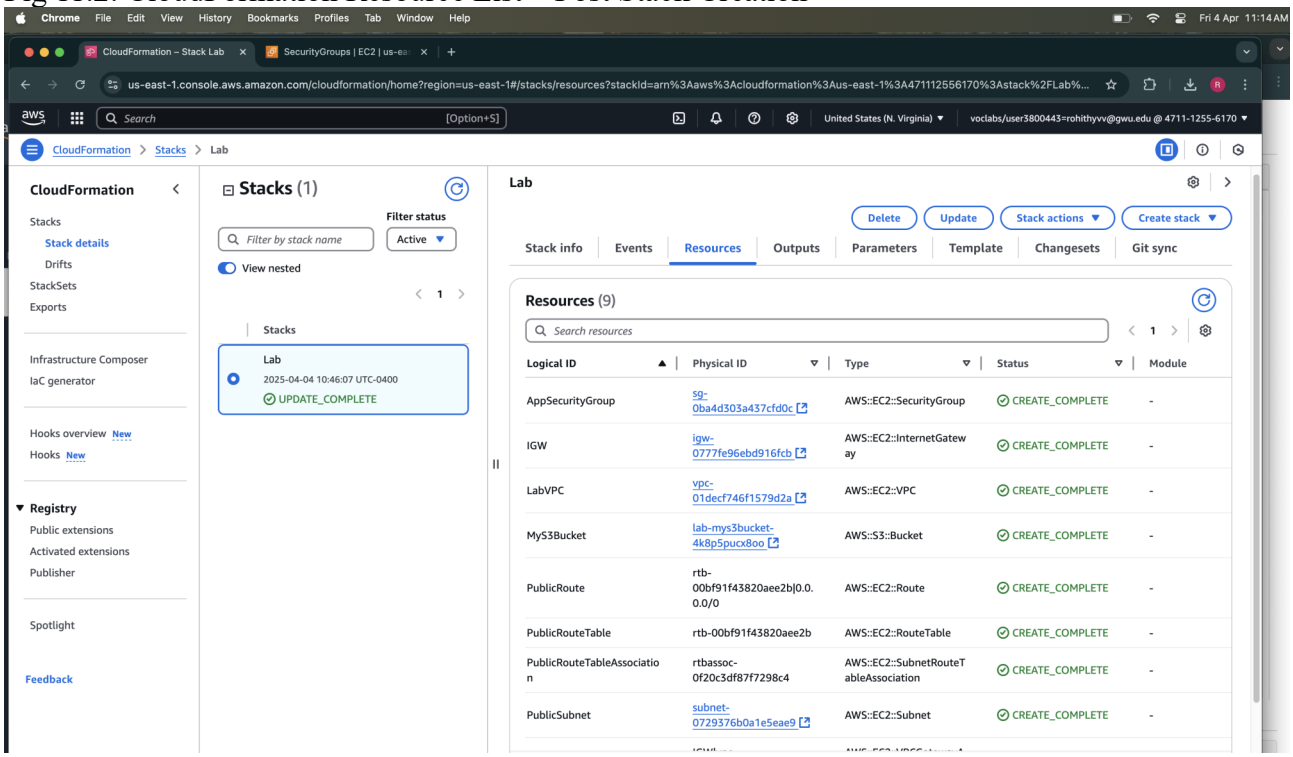
Lab 8 : Automation with CloudFormation

Fig 11.1: Deploy a CloudFormation Stack



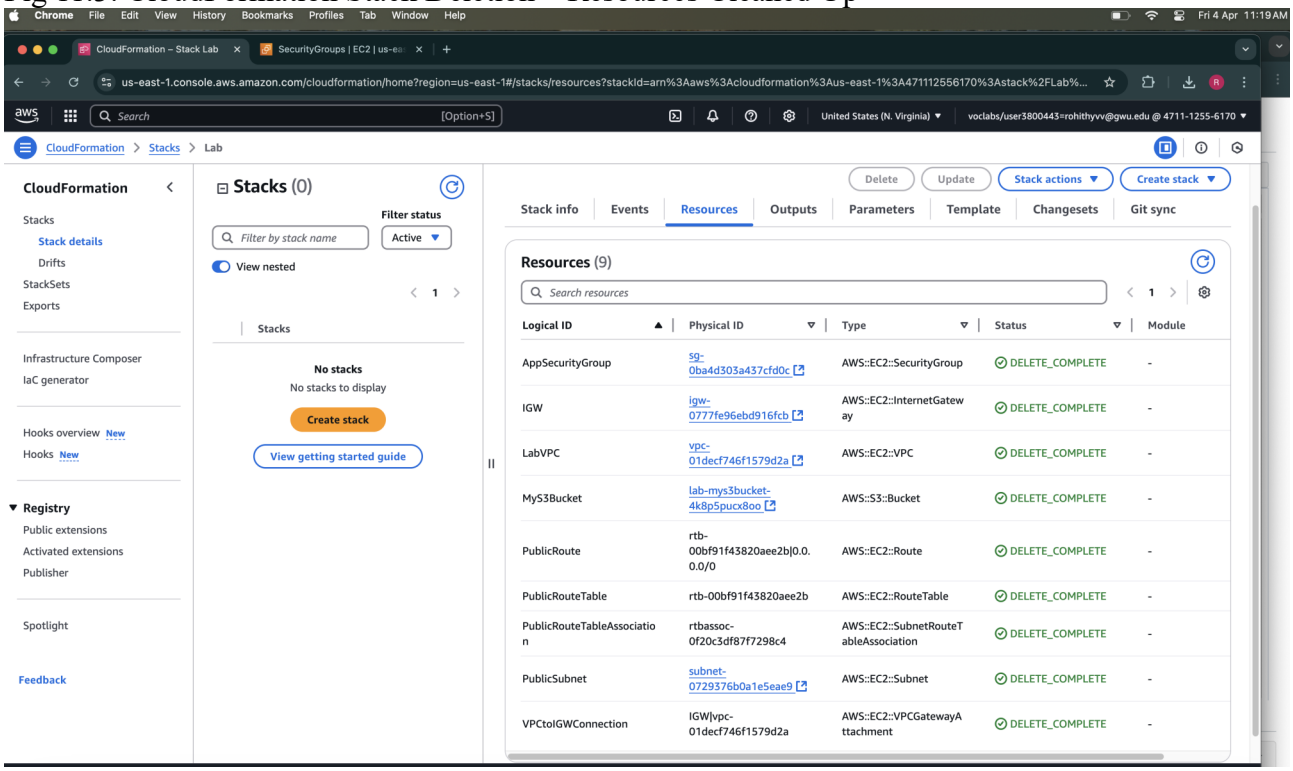
Note: This screenshot shows the successful creation of a CloudFormation stack named "Lab." All stack components, including public route tables and VPC resources, have reached the `CREATE_COMPLETE` status, confirming that infrastructure deployment was successful and error-free.

Fig 11.2: CloudFormation Resource List – Post Stack Creation



Note: This screen displays the resources created as part of the "Lab" stack. Items include a security group, internet gateway, VPC, S3 bucket, route table, subnet, and associations—all with `CREATE_COMPLETE` status, indicating a successful provisioning of AWS infrastructure.

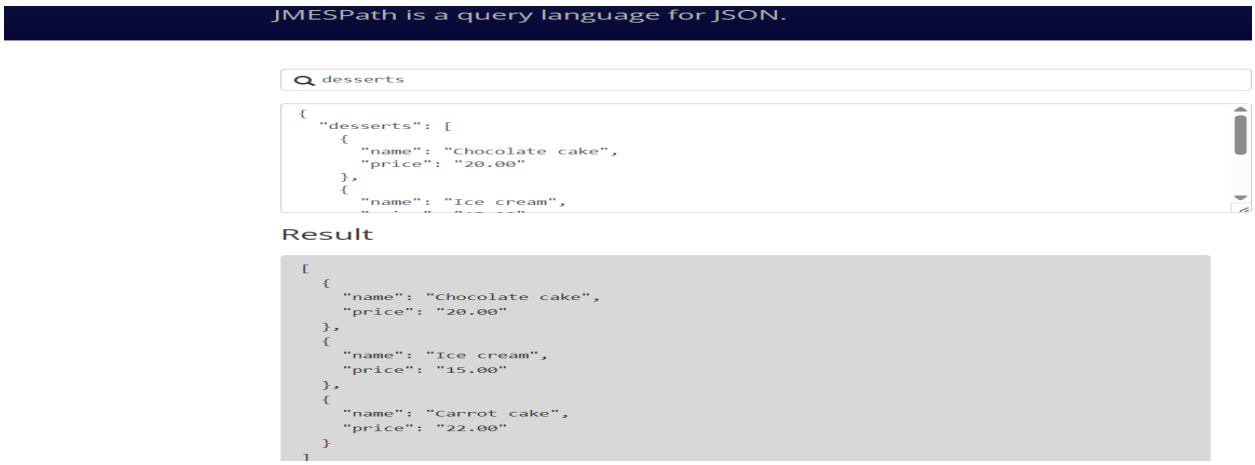
Fig 11.3: CloudFormation Stack Deletion – Resources Cleaned Up



Note: The confirms that all resources from the "Lab" CloudFormation stack, including VPC, S3 bucket, and internet gateway, have been deleted successfully (`DELETE_COMPLETE` status). The stack is no longer listed, ensuring clean deprovisioning of infrastructure.

Activity 11: Troubleshoot CloudFormation

Fig 11.5 : Practice querying JSON-formatted data by using JMESPath

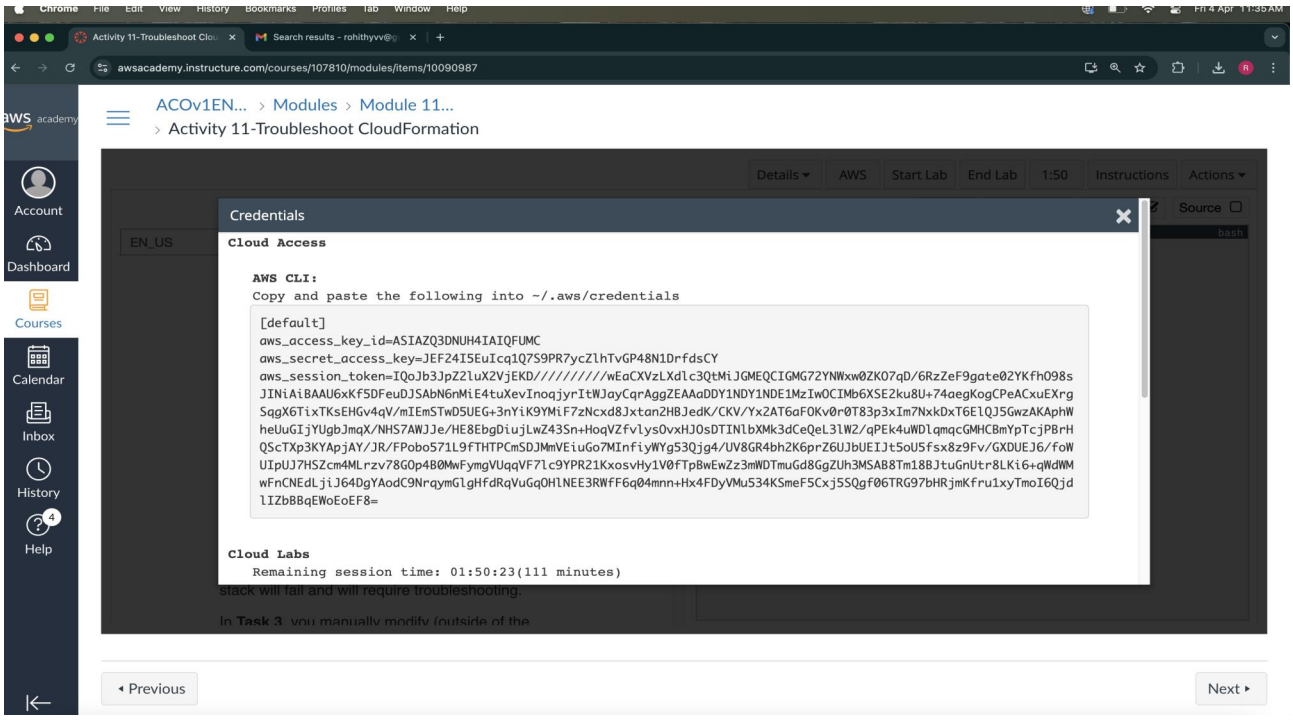


Note: The image shows the JMESPath website, in the document window that currently displays the locations JSON document, copied from the JSON document provided in the lab

Fig 11.6: To retrieve the LogicalResourceId of the EC2 instance resource

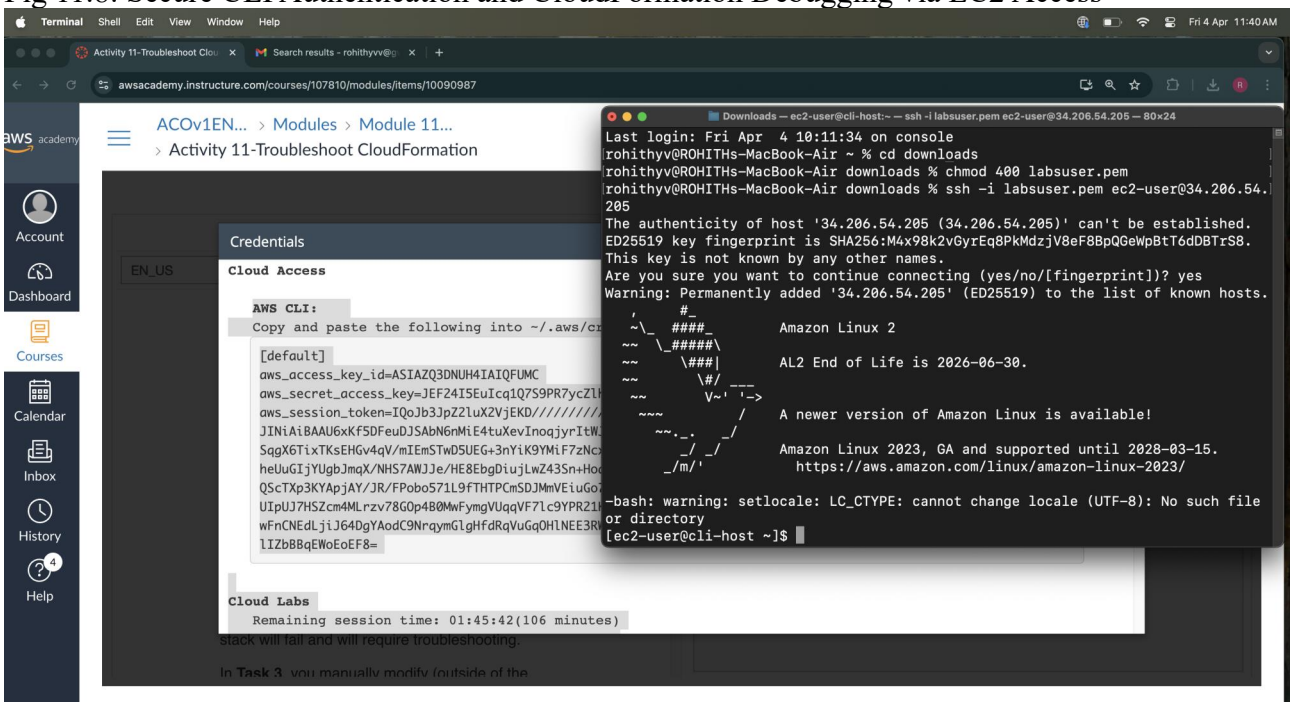
Note: The image shows “StackResources[?ResourceType == 'AWS::EC2::Instance'].LogicalResourceId“ is executed to retrieve the LogicalResourceId of the EC2 instance resource. The instructions often have AWS CLI commands that include --query or --filter parameters. These parameters use JMSEPath expressions to filter the output that is returned by an AWS CLI command.

Fig 11.7: CloudFormation CLI Setup and EC2 SSH Session for Troubleshooting



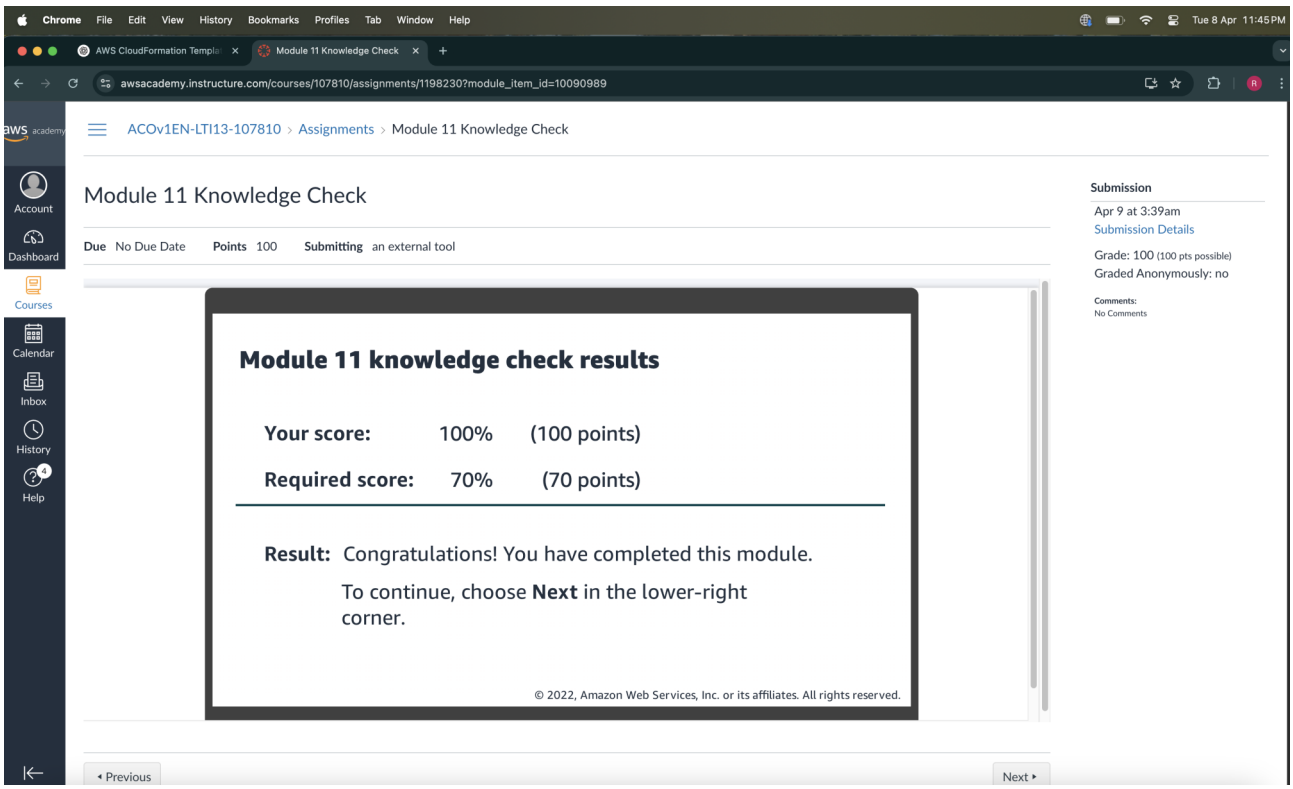
Note: This step captures configuring AWS CLI credentials and initiating an SSH session into the EC2 CLI host. By accessing the instance terminal, users can manually investigate and resolve CloudFormation stack failures directly from the command line interface.

Fig 11.8: Secure CLI Authentication and CloudFormation Debugging via EC2 Access



Note: The screenshot demonstrates secure authentication using temporary AWS credentials and SSH key-based login into an EC2 instance. This is a preparatory step to perform hands-on debugging of AWS CloudFormation issues within a lab environment using the CLI tools.

Fig 11.13: End of module 11



Note: This image shows the completion of module 11

Summary:

- User data allows you to run custom scripts at instance launch, offering a simple way to automate configuration, though it becomes harder to manage at scale in enterprise environments.
- Amazon Machine Images (AMIs) help reduce deployment time by pre-configuring EC2 instances with software and settings specific to your organization's needs.
- Configuration tools like Chef, Puppet, and Ansible let you apply reusable templates to manage and update instance configurations dynamically.
- AWS offers managed services like OpsWorks for Chef/Puppet and CloudFormation for defining infrastructure as code, enabling consistent, repeatable deployments.