

Welcome to AWS Academy Cloud Developing

Objective

- To understand the course structure, objectives, the business case (Cafe scenario) and different roles in cloud computing that will be referenced throughout the AWS Academy Cloud Developing Course.

Theory

- This module provides an overview of cloud computing and how AWS supports developers in building scalable applications. It introduces the Cafe business case - a fictional company used throughout the course to simulate real world cloud development challenges. The module also explains key roles in cloud environments, such as cloud developers, solution architects, DevOps engineers and cloud administrators.

Tools / Services used:

- AWS Academy Learner Lab Platform
- Student Guide (AWS Academy)
- AWS Academy Learning Portal
- External Learning Tools & Videos

Procedure / Steps

1. Accessed AWS Academy Cloud Developing course through the Learner Lab.

2. Reviewed the course objectives and learning roadmap in Part 1 & Part 2.
3. Studied the Cafe Business Case introduction, learning how AWS services will be applied to solve business problems.
4. Explored the Roles in cloud computing section to understand responsibilities of different cloud professionals.
5. Watched the Module 1 wrapup video summarizing key learnings.

Output / Observations

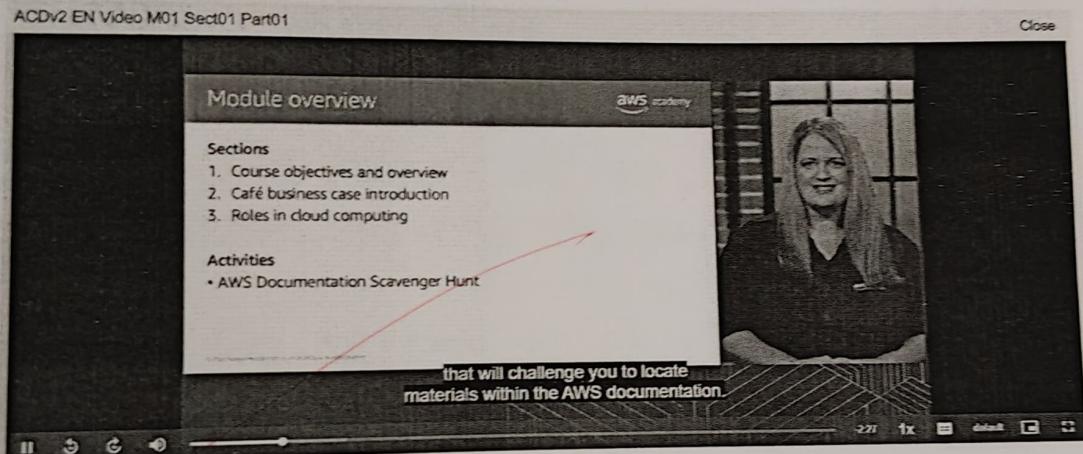
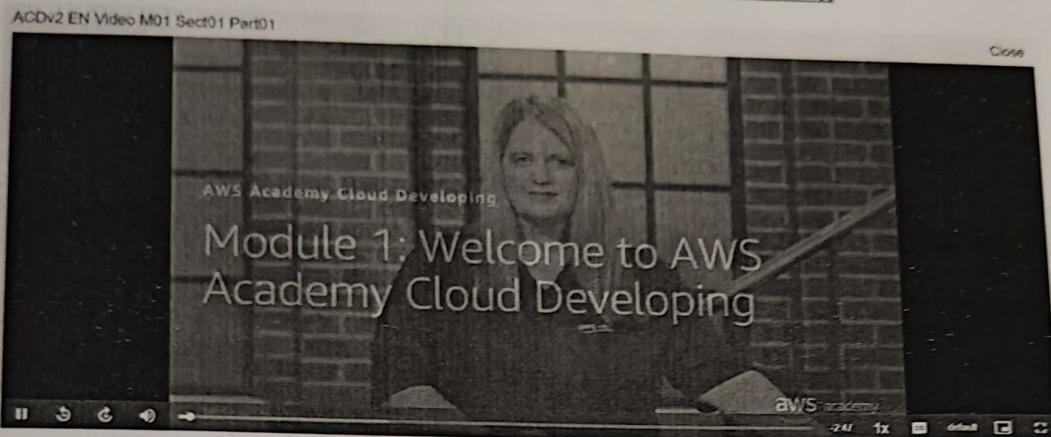
- Understand the main focus areas of the AWS Academy cloud developing course.
- Learned about fictional Cafe Company and how AWS services can modernize business applications.
- Identified core roles in the cloud industry and how they collaborate in real-world cloud projects.

Conclusion

→ Module 1 provided a foundational understanding of what to expect from the AWS cloud developing course. It helped me recognize how cloud developers, architects and operations teams work together to build scalable, secure and efficient applications using AWS services.

18
12/11/25

Module 1: Welcome to AWS Academy Cloud Developing



Exploring AWS CloudShell and an IDE

Objective

To understand and explore the use of AWS CloudShell and an Integrated Development Environment (IDE) for developing, testing and managing AWS resources efficiently through the command line and code editor.

Theory

AWS CloudShell is a browser-based shell that makes it easy to securely manage, interact and automate AWS resources. It provides the AWS Command Line Interface (CLI) pre-installed, removing the need for local configuration.

An IDE (Like Vs Code or cloud9) helps developers write, debug and deploy applications faster by integrating AWS SDKs and extensions for easier cloud development.

Tools / Services Used

- AWS CloudShell
- AWS CLI
- AWS SDKs (Software Development Kit)
- Visual Studio Code
- Amazon Q Developer

Procedure / Steps

1. Logged into AWS Academy Leaves Lab
2. Launched AWS CloudShell from the Management Console
3. Explored build in AWS CLI (commands) (aws s3 ls, aws ec2)

- describe instance etc)
4. Set up AWS SDK environment for Python or Node.js within CloudShell.
 5. Installed and configure Visual Studio Code with AWS Toolkit extension.
 6. Connected the IDE to AWS account to run simple Lambda or S3 operations.
 7. Explored Amazon & Developers

Output / Results.

- Successfully executed AWS CLI commands inside CloudShell.
- Verified SDK functionality by listing AWS resources through code.
- Connected IDE to AWS and performed basic cloud operations.
- Screenshots of CloudShells and IDE Configuration attached.
- Knowledge Check completed with a score of

Conclusion

→ I learned how to use AWS CloudShell for command-line access to AWS services and how to integrate an IDE for development and testing. This lab enhanced my understanding of cloud-based development workflows and the convenience of managing AWS resources without local setup.

8
12/11/26

Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

Lab 2.1: Exploring AWS CloudShell and an IDE

Due No Due Date Points 100 Submitting an external tool

Submission
Sep 10 at 10:28am
Submission Details
Grade: 100 (100 pts possible)
Graded Anonymously: no
Comments:
No Comments

Submit Details AWS Start Lab End Lab Actions Instructions Grades

Files README Terminal Source

EN-US

```
~ $ aws --version
aws-cli/2.28.23 Python/3.13.7 Linux/6.1.247-172.266.mzn2023.x86_64 exec-env/CloudShell exe/x86_64.mzn.2023
```

```
~ $ aws --help
usage: aws [options] <command> [<subcommand> [<subcommand> ...]] [parameters]
To see help text, you can run:
aws <command> help
aws <command> <subcommand> help
aws: error: argument subcommand: Invalid choice, valid choices are:
ls
cp
rm
m
presign
- S
```

```
~ $ cat list-buckets.py
import boto3
session = boto3.Session()
s3_client = session.client('s3')
b = s3_client.list_buckets()
for item in b['Buckets']:
    print(item['Name'])- $
```

Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

The screenshot shows a software development interface with several tabs at the top: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, displaying the following command-line session:

```
git checkout my-branch
# list-buckets.py
def list_buckets():
    import boto3
    session = boto3.Session()
    client = session.client('s3')
    buckets = client.list_buckets()
    for item in buckets['Buckets']:
        print(item['Name'])

buckets.py
download: s3://c171585a4451303111526042t1w8913772498-samplebucket-gomovsqsnwkm/list-buckets.py to ./list-buckets.py
[c171585a4451303111526042t1w8913772498-samplebucket-gomovsqsnwkm] $ aws s3 cp index.html s3://c171585a4451303111526042t1w8913772498-samplebucket-gomovsqsnwkm/index.html
upload: ./index.html to s3://c171585a4451303111526042t1w8913772498-samplebucket-gomovsqsnwkm/index.html
[c171585a4451303111526042t1w8913772498-samplebucket-gomovsqsnwkm] $
```

Below the terminal, there's a red circle highlighting the "AWS" section of the grade details. The grade details show:

- Last submitted: Sep-09-2025 9:58:49 pm PDT
- Submission count: 1
- Due date: None
- AWS: SHOW

A red arrow points from the "AWS" link in the grade details back to the "AWS" section of the terminal output.

Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

```
+ bash
eee_W_5015312@runweb1877
eee_W_5015312@runweb1877
eee_W_5015312@runweb1877
Submission Report:
[Executed at: Tue Sep 10 10:29:43 2025]
Testing report - The
Testing report - The
Back in submit.sh...
end

eee_W_5015312@runweb1877
43:-$ 
eee_W_5015312@runweb1877
43:-$ 
```

Total score 10/10

[Task 1] list-buckets.py was copied

[Task 3A] Index.html was copied to

The screenshot shows a web-based lab submission interface. At the top, there's a toolbar with various icons. Below it, the URL bar shows "ENG IN". The main area has a "Due" button, "No Due Date", "Points 100", and a "Submitting" status message. To the right, there's a progress bar with a "100%" label. A large central window is titled "End Lab" and contains the following text:
Region: us-east-1
Lab ID: arn:aws:cloudformation:us-east-1:691377249095:stack/c171e85a44513031115260421lw891377249095/a43435a0-8dfe-11f0-ad21-0aifcd50382f
Creation Time: 2025-09-09T21:20:46-0700
You may close this message box now. Lab resources are terminating ...

The screenshot shows a "Knowledge check results" page. On the left, there's a sidebar with navigation links: Home, Modules, Announcements, Discussions, Grades, and Lucid (Whiteboard). The main content area displays the following information:
Knowledge check results
Your score: 100% (100 points)
Required score: 70% (70 points)
Result: Congratulations! You have completed this module.
To continue, choose **Next** in the lower-right corner.

Working with Amazon S3

Objective

- To learn how to create, configure and manage Amazon S3 (Simple Storage Service) buckets and objects, and to understand how to control access, store data efficiently and ensure durability in cloud-based storage.

Theory

→ Amazon S3 is an object storage service that provides industry-leading scalability, data availability, security and performance. It allows developers to store and retrieve any amount of data from anywhere on the web. S3 stores data as objects within buckets each having unique access permissions and configurations. S3 also integrates with IAM for access control and supports Versioning, encryption and lifecycle policies for cost optimization and security.

Tools / Services used

- Amazon S3
- AWS Management Console
- AWS CLI
- IAM (Identity and Access Management)
- Bucket Policy / Access Control List (ACL)

Procedure / Steps

- Logged into the AWS Academy Learner Lab environment.

- Navigated to the Amazon S3 service from the AWS Management console.
- Created a new S3 bucket with a unique global name.
- Configure Bucket Versioning and Public Access settings.
- Uploaded multiple files (text/images) to the bucket.
- Used the AWS CLI (aws s3 ct, aws s3 ls) to interact with the bucket.
- Modified bucket policies and access permissions to allow deny public access.
- Tested file retrieval using the object URL.
- Verified bucket configuration and ensure secure access settings.

Output / Result

- Successfully created and configured an Amazon S3 bucket.
- Uploaded and accessed multiple objects using the AWS Console and CLI.
- Applied and tested access control policies.
- Completed knowledge check.

Conclusion

→ This lab enhanced my understanding of cloud storage solutions using Amazon S3. I learned how to create and manage buckets, control permissions and securely store data in the cloud. The lab is also provided practical exposure to managing S3 through both AWS Management Console and AWS CLI.

JG
12/1/25

Lab 3.1: Working with Amazon S3

The screenshot shows a software interface for working with Amazon S3. On the left, there's a sidebar with icons for Account, Dashboard, Courses, Calendar, and Inbox. The main area has a 'Launch Terminal' button at the top. Below it is a terminal window showing a command-line interface. To the right of the terminal is a 'Submission Report' window. The report header says 'Submission Report' and 'EN-US'. It includes a timestamp: 'Executed at: Tue Sep 16 22:08:04 PDT 2025'. The report lists three tasks:

- [Task 2] The S3 bucket was created: Status: Success
- [Task 3] Policy applied to bucket: Status: Problem
- [Task 4] index.html was copied to S3: Status: Success

The report also contains some terminal output and a note about choosing 'Submit'.

Module 3 knowledge check results

Your score: 90% (90 points)

Required score: 70% (70 points)

Result: Congratulations! You have completed this module.

To continue, choose **Next** in the lower-right corner.

Securing Access to Cloud Resources

Aims

To understand how to secure access to AWS resources by implementing Identity and Access Management (IAM) principles and following the AWS Shared Responsibility Model.

Theory

In cloud environment, security is a shared responsibility between AWS and the customer.

- AWS manages the security of the cloud (infrastructure, hardware, regions, data centers)
- Customers manage the security in the cloud (data encryption, user access, permissions)

AWS Identity and Access Management (IAM) is a key service that helps organizations securely control access to AWS resources. IAM allows you to:

- Create and manage users, groups and roles.
- Define policies to control permissions.
- Implement multi-factor authentication (MFA).
- Enable cross-account access and fine-grained authorization.

Tools / Services Used

- AWS Identity and Access Management (IAM)
- AWS Management Console

Policy JSON Editor Cross-Account Access Demonstration

Activities / steps

- Reviewed the Shared Responsibility Model and Identified AWS vs Customer responsibilities.
- Studied IAM Components - users, grants, roles and policies.
- Observed the demonstration - Configuration Cross Account Access.
- Viewed the demonstration.
- Explored the authorization using IAM Policies (Part 1 and Part 2)
- Completed the knowledge check

Output / Results

- Gained hands on understanding of how IAM secures AWS environments.
- Completed the knowledge check.

Conclusion

The module helped me understand the importance of Access Control and Identity Management in cloud environment. I learned how AWS IAM ensures that only authorized users can access specific resources and how policies define granular permissions for secure cloud operations.

8/11/25

Module 4: Securing Access to Cloud Resources

The screenshot shows a web-based interface for managing AWS CloudFormation stacks. The left sidebar contains navigation links: Home, Modules (which is selected), Announcements, Discussions, Grades, and Lucid (Whiteboard). The main content area is titled 'ACDv2-EN-M04-Demo-Cross-Account'. It shows the 'Outputs' section of the stack details, listing various outputs such as 'AWSRegion', 'AWSAccountID', and 'CrossAccountRoleARN'. A tooltip or callout box highlights the 'CrossAccountRoleARN' output with the text 'that this policy will allow'.

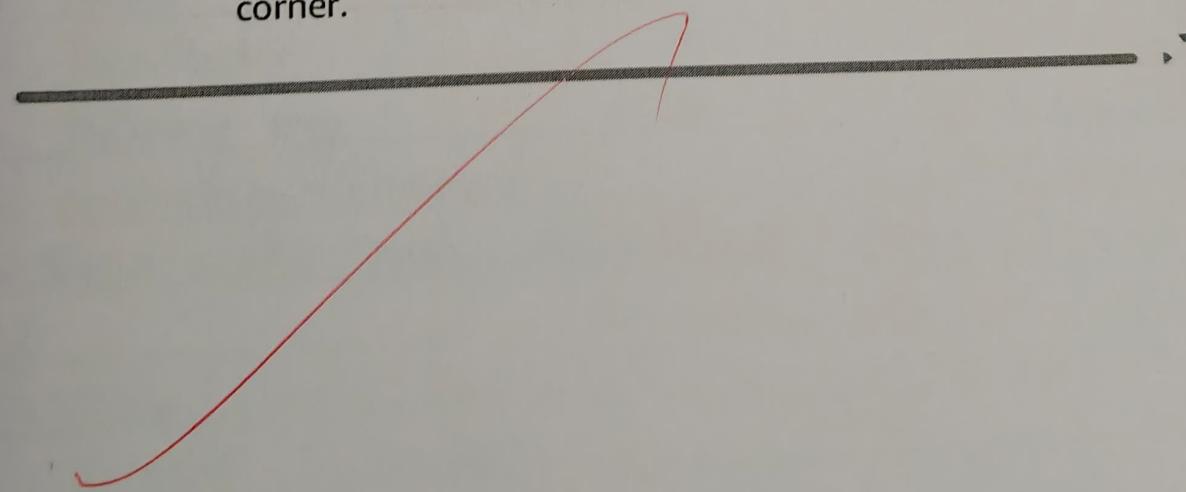
Module 4 knowledge check results

Your score: 80% (80 points)

Required score: 70% (70 points)

Result: Congratulations! You have completed this module.

To continue, choose **Next** in the lower-right corner.



Working with Amazon DynamoDB

Objective

To learn how to create, configure and manage Amazon DynamoDB tables for building scalable, flexible and high-performance NoSQL database solutions in AWS.

Theory

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.

It stores data in key-value and document formats, making it ideal for applications requiring high availability and low latency.

Key DynamoDB Concepts include:

- Tables
- Items
- Attributes
- Primary Key
- Secondary Indexes
- Read/Write Capacity Units (RCUs, WCUs)

Tools / Services Used

- Amazon DynamoDB
- AWS Management Console
- AWS CLI
- DAM for Access Management

Procedure / steps

Logged into the AWS Academy Learner Lab environment.

Opened the Amazon DynamoDB from the AWS Console Management.

Created a new table named StudentData with a partition key:

StudentID

Configure ReadWrite Capacity

Added sample data

Queries and scanned the table to retrieve items

Created a Global Secondary Index.

Exported backup and restore.

Output

- Successfully created and configured a DynamoDB table.
- Performed CRUD operation
- Implemented Secondary Index
- Verified table performance.
- Completed knowledge check

Conclusion

- This lab enhanced my understanding of how to develop NoSQL database solution on AWS using DynamoDB. I learned to design data models, manage indexes, configure capacity and use advanced features like streams and backups.

88
12/11/26

Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

Module 5: Developing Flexible NoSQL Solutions
Lab 5.1: Working with DynamoDB

Submission Report

```
[Executed at: Thu Oct 30 23:47:58 PDT 2025]
Testing report - The DynamoDB table was created!
Testing report - The batch load completed successfully!
Testing report - The GSI was added to the table!
Back in submit.sh...
end

Submitted, choose Yes
```

Lab 5.1: Working with DynamoDB

Due No Due Date Points 100 Submitting an external tool

Submit Details AWS Start Lab End Lab 2:05 Instructions Grades Actions

EN-US Files README Terminal Source

Submitting your work

65. At the top of these instructions, choose **Submit** to record your progress and when prompted, choose **Yes**.

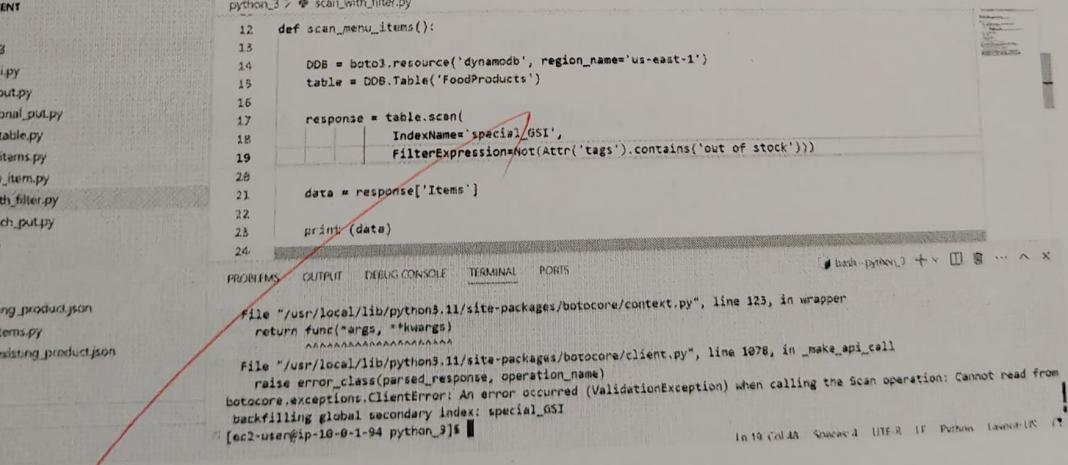
66. If the results don't display after a couple of minutes, return to the top of these instructions

Total score 15/15

[Task 2] DynamoDB table created

[Task 5] Batch load completed

[Task 7] GSI added to table



```
scanning_with_filter.py
python_3 > scan_with_filter.py
12 def scan_menu_items():
13
14     DDB = boto3.resource('dynamodb', region_name='us-east-1')
15     table = DDB.Table('FoodProducts')
16
17     response = table.scan(
18         IndexName='special_GSI',
19         FilterExpressionNot(Attr('tags').contains('out of stock')))
20
21     data = response['Items']
22
23     print (data)
24
File "/usr/local/lib/python3.11/site-packages/botocore/context.py", line 123, in wrapper
    return func(*args, **kwargs)
           ^^^^^^^^^^^^^^
File "/usr/local/lib/python3.11/site-packages/botocore/client.py", line 1078, in _make_api_call
    raise error_class(parsed_response, operation_name)
botocore.exceptions.ClientError: An error occurred (ValidationException) when calling the Scan operation: Cannot read from backfilling global secondary index: special_GSI
[ec2-user@ip-10-0-1-94 python_3]$
```

✓ Completed · Items returned: 27 · Items scanned: 27 · Efficiency: 100% · RCUs consumed: 0.5 X

Table: FoodProducts - Items returned (27)



Actions ▾

Create item

Scan started on October 31, 2025, 12:15:32

	product_name (String)	description	price_in_cents
<input type="checkbox"/>	<u>blueberry jelly doughnut</u>	A doughnut w...	295
<input type="checkbox"/>	<u>vanilla glazed doughnut</u>	so good!	295
<input type="checkbox"/>	<u>boston cream doughnut</u>	Boston's favor...	295

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

Module 5 Knowledge Check

Due No Due Date Points 100 Submitting an external tool

Module 5 knowledge check results

Your score: 90% (90 points)

Required score: 70% (70 points)

Result: Congratulations! You have completed this module.

To continue, choose **Next** in the lower-right corner.

Developing REST APIs with API Gateway

Objective

To learn how to create, configure, deploy and secure REST APIs using Amazon API Gateway integrated with AWS backend services.

Theory

Amazon API Gateway is a fully managed service that allows developers to create, publish, maintain, monitor and secure RESTful APIs at scale. It acts as a bridge between clients and backend services such as AWS Lambda, DynamoDB or EC2.

Tools / Services Used:

- Amazon API Gateway
- AWS Lambda
- AWS IAM
- Cloudwatch

Procedure / Steps:

1. Logged into the AWS Academy Learner Lab
2. Created a REST API using API Gateway
3. Defined resources and methods.
4. Integrated API with AWS Lambda function
5. Deployed the API to a stage
6. Tested API endpoint via the invoke URL
7. Configured IAM roles and policies for access control.

Output / Result

- Successfully built and deployed a REST API using API Gateway and Lambda
- Verified working endpoints and access permissions.
- Completed knowledge check

Conclusion

Learned how to develop and deploy secure, scalable REST APIs using API Gateway, integrate with AWS services and monitor performance through CloudWatch.

✓
12/11/26

California Engineering Lab
Keweenaw Research, 1-74447-723466, AGC1 Div 5, Section 3

Architects: Gropius, Breuer, Mies van der Rohe

Table 6.1: Developing WEST Africa with DSD: Government

Best practices from Project XRF: Implementing an external role

Submitting your work

At the top of these instructions, choose **Submit** to report your progress and when prompted, choose **File**.

If you previously hid the terminal in the browser page, expose it again by selecting the **Terminal** tab.

Run the following command:

```
git add .
```

Then run the command:

```
git commit -m "Initial commit"
```

Finally, run the command:

```
git push
```

Check the status of your repository:

```
git status
```

Check the history of your repository:

```
git log
```

Check the current branch:

```
git branch
```

Switch to another branch:

```
git checkout [branch_name]
```

Push changes to a different branch:

```
git push -u [branch_name]
```

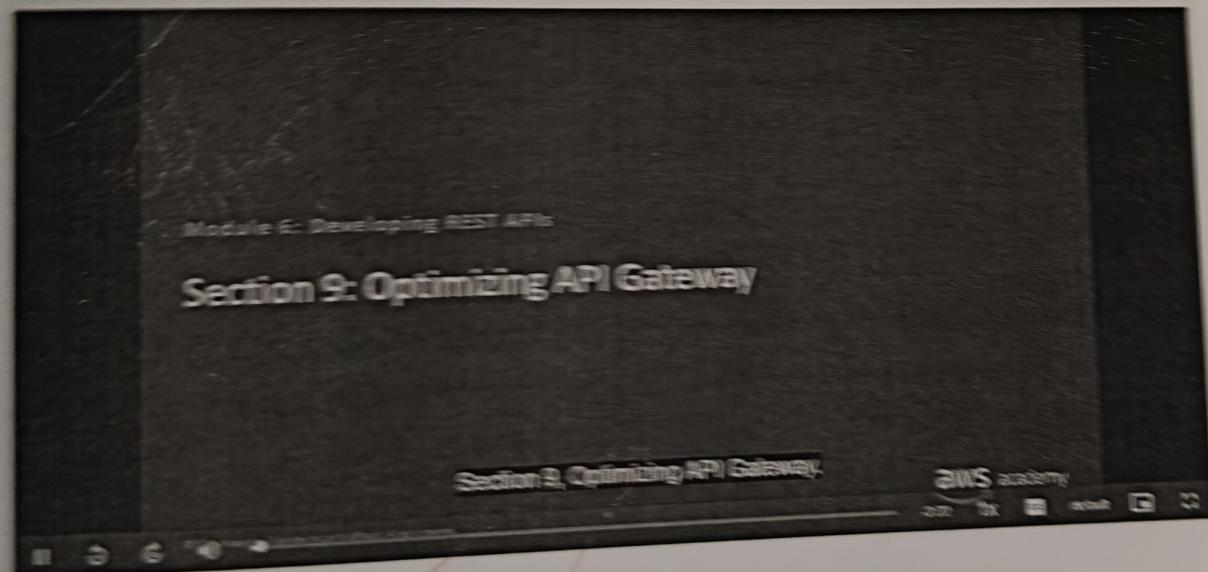
Get help on a command:

```
git help [command]
```

Get help on all commands:

```
git help
```

ACQUA BOTTLED WATER



Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

Due: No Due Date Points: 100 Submitting an external tool

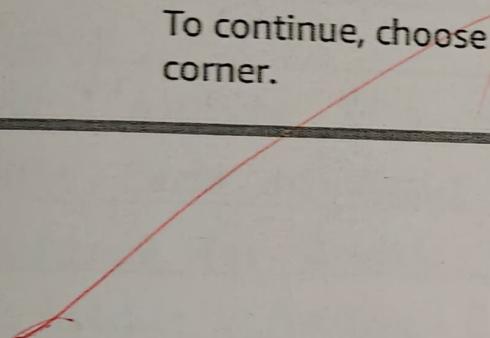
Module 6 knowledge check results

Your score: 90% (90 points)

Required score: 70% (70 points)

Result: Congratulations! You have completed this module.

To continue, choose **Next** in the lower-right corner.



Creating Lambda functions Using the AWS SDK for Python

Objective

To learn how to create, configure and invoke AWS Lambda functions using the AWS SDK for Python in a serverless environment.

Theory

AWS Lambda is a serverless compute service that automatically runs code without managing servers. It executes code in response to events such as API calls, file uploads or database changes. Using Boto3, developers can programmatically create and manage Lambda functions from Python applications.

Tools / Services Used

- AWS Lambda
- AWS SDK for Python (Boto3)
- IAM Roles and Policies
- Amazon CloudWatch

Procedure / Steps

1. Logged into the AWS Academy Learner Lab
2. Created an IAM role with Lambda execution permissions.
3. Authored a Python function using Boto3 to print a custom message.
4. Deployed the function to AWS Lambda
5. Invoked the function manually and verified logs.

Modified function configuration and redeployed updates.
Tested error handling and event sources.

Output / Result

- Successfully created and executed a Lambda function using Python SDK.
- Observed function logs and performance metrics in CloudWatch.
- Completed knowledge check.

Conclusion

- Learned how to build serverless applications using AWS Lambda and Python SDK. This module improved my understanding of event-driven architecture, function deployment and monitoring in AWS.

✓ 80
12/11/26

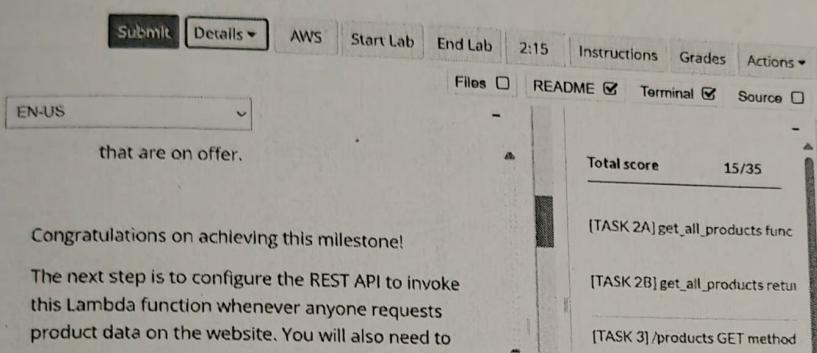
Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

Lab 7.1: Creating Lambda Functions Using the AWS SDK for Python

Lab 7.1: Creating Lambda Functions Using the AWS SDK for Python

Due No Due Date Points 100 Submitting an external tool

Submission
Nov 4 at 7:57 am
Submission Details
Grade: 0 (100 pts possible)
Graded Anonymously, no
Comments: No Comments



The screenshot shows a software interface with a toolbar at the top containing 'Submit', 'Details', 'AWS', 'Start Lab', 'End Lab', '2:15', 'Instructions', 'Grades', and 'Actions'. Below the toolbar is a dropdown menu set to 'EN-US'. The main area has tabs for 'Files' (unchecked), 'README' (checked), 'Terminal' (checked), and 'Source' (unchecked). The 'README' tab displays the following text:
that are on offer.

Congratulations on achieving this milestone!
The next step is to configure the REST API to invoke this Lambda function whenever anyone requests product data on the website. You will also need to

Total score 15/35

[TASK 2A] get_all_products func

[TASK 2B] get_all_products return

[TASK 3] /products GET method

Module 7 knowledge check results

Your score: 100% (100 points)

Required score: 70% (70 points)

Result: Congratulations! You have completed this module.

To continue, choose **Next** in the lower-right corner.



Introducing Containers and Container Services (Lab-8.1 & 8.2)

Objective

- To understand Containerization using Docker and Lambda on AWS using Elastic Beanstalk and other managed Container services.

Theory

Containers provide a lightweight and portable way to package applications and their dependencies, ensuring consistency across environments.

Docker is the most widely used container platform that allows developers to build, ship and run applications quickly.

AWS supports containers through services like

- Amazon Elastic Container Service (ECS)
- Elastic Kubernetes Service (EKS)
- Elastic Beanstalk, which automates deployment and scaling of containerized apps.

Tools / Services Used

- Docker
- AWS Elastic Beanstalk
- Amazon ECS (Elastic Container Service)
- AWS Management Console / CloudShell

Procedure / steps

Logged into the AWS Academy Learner Lab environment.

Created a Dockerfile for a sample web application.

Built and tested the Docker Image locally to verify functionality.

Deployed the containerized application on Elastic Beanstalk as a managed service.

Observed automatic environment provisioning and scaling.

Explored ECS dashboard for understanding container orchestration and management.

Monitored deployment status and application performance metrics.

Completed the module knowledge check.

Output

- Successfully created and ran a Docker Container for a web application.
- Deployed the containerized app using AWS Elastic Beanstalk.
- Understood how to manage container scaling, deployment and monitoring.

Conclusion

- This module provided hands-on experience in containerization and cloud deployment. I learned how to migrate applications to Docker, deploy them on AWS managed services like Elastic Beanstalk and manage their lifecycle efficiently. This knowledge is essential for modern DevOps and Microservices based architecture.

Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

Lab 8.1: Migrating a Web Application to Docker Containers

Lab 8.1: Migrating a Web Application to Docker Containers

Due No Due Date Points 100 Submitting an external tool

Submission Report

[Executed at: Mon Nov 10 5:29:48 EST 2025]

Testing report - The suppliers database contains at least one supplier!
Testing report - The security group was configured correctly!
Testing report - The Dockerfile for MySQL was found!
Testing report - Evidence was found of a successful launch of a MySQL contain
Testing report - The two containers are running as expected.
Testing report - The image node-app was pushed to Amazon ECR!

Was mysql_1 container found running? (1=yes, 0=no): 1
Was node_app_1 container found running? (1=yes, 0=no): 1
Excellent! Found evidence of mysql container and node app container

[TASK 4] Created MySQL Docker

Submit Details AWS Start Lab End Lab 1:13 Instructions Grades Actions

Files README Terminal Source

bash

```
+ * Found key for Cloud9.
Result of query for mysql dockerfile: CompletedProcess(args=['ssh',
'-i', 'c9key.pem', '-o StrictHostKeyChecking=no', 'ec2-user@98.95.134.
71', 'find /home/ec2-user/environment/containers/mysql -name Dockerfile
|wc -l'], returncode=0, stdout=b'1\n', stderr=b"Warning: Permanently a
dded '98.95.134.71' (ECDSA) to the list of known hosts.\r\n")
Does mysql Dockerfile exist? (1 = yes, 0 = no): 1
Excellent! Found the mysql dockerfile.
Does Docker log show that mysql started in container? (1+ = yes, 0
= no): 2
Excellent! Found evidence in logs of mysql container starting succe
ssfully.
***** TASK 5 *****
Was mysql_1 container found running? (1=yes, 0=no): 1
Was node_app_1 container found running? (1=yes, 0=no): 1
Excellent! Found evidence of mysql container and node app container
both running.
***** TASK 6 *****
Back in submit.sh...
```

Totalscore 30/30

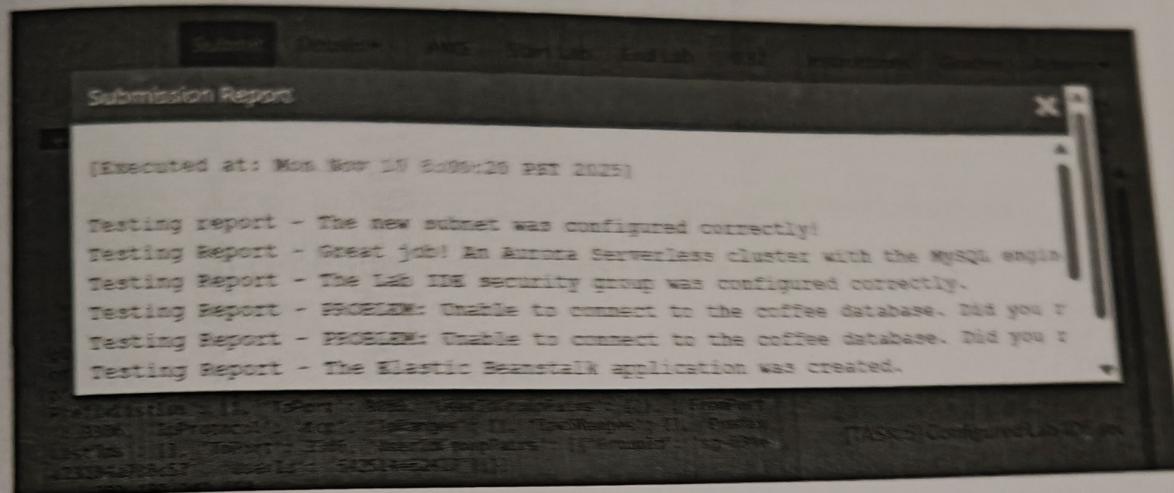
[TASK 2] Created new supplier

[TASK 3] Configured the security

[TASK 4] Created MySQL Docker

Lab 8.2: Running Containers on a Managed Service

Due No Due Date Points 100 Submitting an external tool



Lab 8.2: Running Containers on a Managed Service

Due No Due Date Points 100 Submitting an external tool

The screenshot shows a lab interface with the following details:

- Submit button
- Details dropdown
- AWS tab
- Start Lab button
- End Lab button
- 0:34 timer
- Instructions tab
- Grades tab (highlighted)
- Actions dropdown

Task details:

- File icon
- README.md
- Terminal icon
- Source icon

Task 2: Configured new subnet

Task 3: Created database

Code editor content (partially visible):

```
i-01d6e736bd93f8d1
  ("GroupName": "Lab IDE SG", "GroupId": "sg-034ea23104a780c57")
    sg-034ea23104a780c57
      [{"FromPort": 80, "IpProtocol": "tcp", "IpRanges": [], "Ipv6Ranges": []}, {"PrefixListIds": [{"Description": "Allow HTTP from com.amazonaws.global.cloudfront.origin-facing", "PrefixListId": "pl-3b927c52"}]}, {"ToPort": 80, "UserIdGroupPairs": []}], {"FromPort": 8000, "IpProtocol": "tcp", "IpRanges": [{"CidrIp": "263.192.242.194/32"}], "Ipv6Ranges": []}]
```

Module 8 knowledge check results

Your score: 100% (100 points)

Required score: 70% (70 points)

Result: Congratulations! You have completed this module.

To continue, choose **Next** in the lower-right corner.

Caching Information for Scalability (Lab. 9.1 & 9.2)

Objective

To understand how Caching improves applications and scalability using Amazon ElastiCache and Amazon CloudFront, and to implement caching strategies for faster data delivery and reduced latency.

Theory

Caching stores frequently accessed data in temporary storage to reduce database load and response time.

- Amazon ElastiCache is a managed in-memory caching service supporting Redis and Memcached engines, ideal for application level Caching.
- Amazon CloudFront is a global Content Delivery Network (CDN) that caches content closer to users for low-latency access and enhanced security. Together, these services improve application scalability, performance and user experience.

Tools / service used

- Amazon ElastiCache (Redis / Memcached)
- Amazon CloudFront
- Amazon EC2 (for application testing)
- AWS Management Console

- Procedure / steps
- logged into the AWS Academy learner lab.
 - created an ElastiCache cluster and connected it with an application hosted on EC2.
 - Configured Caching for frequently queried data and tested response time improvements.
 - Set up a CloudFront distribution to cache web content from an S3 bucket or web server.
 - Configured Origin settings.
 - Tested Content delivery through CloudFront URL.
 - Verified faster response time.
 - Completed knowledge check.

Output

- Successfully implemented Caching with ElastiCache for database queries.
- Configured CloudFront as a CDN for global content delivery.
- Verified reduced latency and improved application performance.
-

Conclusion

→ This module provided practical understanding of Caching strategies using AWS services. I learned how ElastiCache accelerates data retrieval at the application layer and how CloudFront improves content delivery speed globally. These Caching techniques are essential for building scalable, high-performance cloud applications.

✓
88
12/11/25

Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

Lab 9.1: Caching Application Data with ElastiCache

Lab 9.1: Caching Application Data with ElastiCache

Due No Due Date Points 100 Submitting an external tool

```
[Executed at: Sun Nov 9 11:57:18 PST 2025]

Testing report - The security group was configured correctly!
Testing report - The subnet group was configured correctly!
Testing Report - The ElastiCache for Memcached cluster was created. Great job!
Testing Report - The update_item.py script was run, Great job!
Testing Report - The create_item.py script was run, Well done!
Testing Report - The delete_item.py script was run, Well done!
Testing Report - The MEMC_HOST configuration was added to myenv.
```

Lab 9.2: Implementing CloudFront for Caching and Application Security

Due No Due Date Points 100 Submitting an external tool

Details

Last submitted: Nov-09-2025 3:27:50 am PST

Submission count: 1

Due date: None

AWS: Show

View Submission Report

New report ready

Files README Terminal Source

Total score 50/60

[Task 2A] Created CloudFront distribution

[Task 2B] Configured S3 bucket

[Task 2C] Updated S3 bucket policy

[Task 3A] Created a global AWS Lambda function

Module 9 knowledge check results

Your score: 90% (90 points)

Required score: 70% (70 points)

Result: Congratulations! You have completed this module.
To continue, choose Next in the lower-right corner.

Implementing a Messaging System Using Amazon SNS and Amazon SQS

Objective

To learn how to design and implement asynchronous communication between distributed systems using Amazon Simple Queue Service (SQS) and Amazon Simple Notification Service (SNS)

Theory

Asynchronous Messaging allows application to communicate and process tasks independently without waiting for immediate responses.

Amazon SQS provides a reliable, scalable queue system for storing and processing messages asynchronously.

Amazon SNS is a fully managed pub/sub (publish-subscribe) messaging service that sends notifications to multiple subscribers or endpoints.

→ Combined these services enable decoupled, scalable and fault-tolerant application architectures.

Tools / services used

- Amazon SQS (Simple Queue Service)
- Amazon SNS (Simple Notification Service)
- AWS Management Console
- AWS CLI

Procedure / Steps

Logged into the AWS Academy learner lab.
 Created a standard SQS queue for message storage.
 Created an SNS topic and subscribed an SQS queue to it.
 Configured permissions and access policies for SNS-SQS integration.
 Published sample messages to the SNS topic.
 Verified message delivery to the SQS queue and tested message retrieval.
 Completed knowledge check.

Output

Successfully implemented an SNS-SQS integrated messaging system.

- Verified message delivery and queue processing
- Demonstrated asynchronous communication between components

Conclusion

This module helped me understand event-driven and decoupled application design using AWS messaging services. I learned how SQS manages message queues reliably and how SNS distributes notifications efficiently across multiple subscribers.

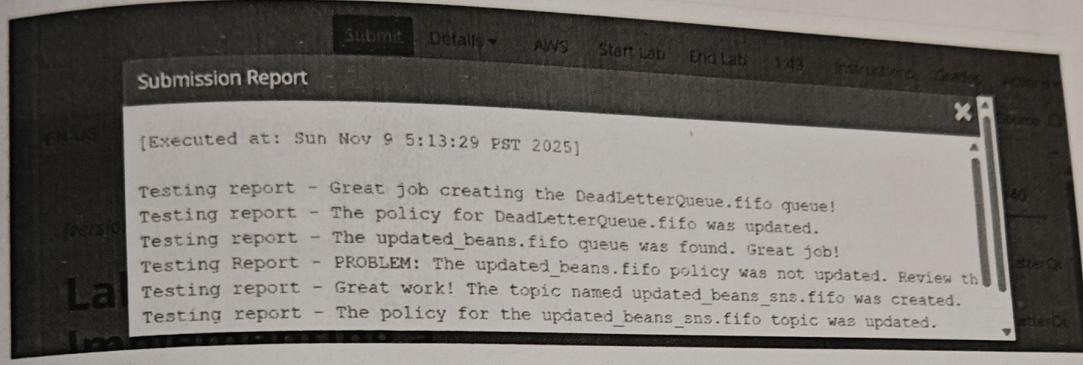
JF
12/11/25

Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

Lab 10.1: Implementing a Messaging System Using Amazon SNS and Amazon SQS

Lab 10.1: Implementing a Messaging System Using Amazon SNS and Amazon SQS

Due No Due Date Points 100 Submitting an external tool



The interface includes tabs for "Files", "README", "Terminal", and "Source". The "Terminal" tab shows a bash session with the user "eee_W_5015312@runweb1950". The "Score" section displays a total score of 35/40.

Total score 35/40

[Task 2A] Created DeadLetterQl

[Task 2B] Secured DeadLetterQl

Module 10 knowledge check results

Your score: 80% (80 points)

Required score: 70% (70 points)

Result: Congratulations! You have completed this module.
To continue, choose Next in the lower-right corner.

Orchestrating serverless functions with step functions

Objective

To learn how to coordinate and manage multiple serverless functions using AWS Step Functions to build reliable automated workflows.

Theory

- AWS Step Functions is a serverless Orchestration service that helps automate multi-step processes by combining AWS services like Lambda, SNS or SQS into a single, visual workflow.

It uses State Machine to define tasks, sequences and conditions. Each state represents a step - such as invoking a Lambda function or making a decision - allowing developers to manage complex application logic with reliability and error handling.

Tools / Services Used

- AWS Step Functions
- AWS Lambda
- Amazon CloudWatch
- AWS Management Console

Procedure / Steps

- Logged into the AWS Academy Learner Lab
- Opened AWS Step Functions and created a new state machine.
- Defined workflow states and transitions using the Amazon States Language (ASL) - JSON based language

Integrated Lambda functions to perform tasks within the workflow.

Executed and tested the workflow to verify state transitions.
Monitored workflow execution and logs in CloudWatch.
Completed the knowledge check.

Output

Successfully created and executed a Step Functions state machine.

- Verified function coordination and error handling.
- Observed workflow execution flow and logs.

Conclusion

This module provided practical experience in workflow orchestration using AWS Step Functions. I learned how to link multiple serverless functions into a coordinated sequence, improving automation, reliability and visibility in distributed systems.

80
12/11/25

Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

Lab 11.1: Orchestrating Serverless Functions with Step Functions

Due No Due Date

Points 100

Submitting an external tool

Lab 11.1: Orchestrating Serverless Functions with Step Functions

Submission Report

[Executed at: Sun Nov 9 7:49:09 PST 2025]

Testing report - Great job creating the EmailReport SNS topic!

Testing report - You created the state machine. Well done!

Testing Report - The SNS Publish action was added to the state machine.

Testing Report - The Lambda function named GeneratePresignedURL was created.

Testing Report - The GeneratePresignedURL action was added to the state machine.

Testing Report - The API Gateway create_report method was configured to integrate

The screenshot shows a terminal window with a command prompt and a browser-based interface for grading. The terminal window shows a user session with a bash prompt. The browser interface has tabs for 'Files' (unchecked), 'README' (checked), 'Terminal' (checked), and 'Source' (unchecked). It displays a 'Total score' of 50/50. Below the score, three tasks are listed: '[Task 2] Created SNS topic', '[Task 3A] Created state machine', and '[Task 3B] Added SNS Publish act'. The 'Terminal' tab shows the command 'tee_W_5015312@runweb1948'.

Module 11 knowledge check results

Your score: 90% (90 points)

Required score: 70% (70 points)

Result: Congratulations! You have completed this module.

To continue, choose **Next** in the lower-right corner.

Implementing Application Authentication using Amazon Cognito

Objective

- To understand and implement secure user authentication in cloud based application using Amazon Cognito and AWS Security Token Service (STS)

Theory

- Building secure applications in the cloud involves managing user identities, authentication and access control.
- Amazon Cognito simplifies adding user sign-up, sign-in and access control to web and mobile apps. It integrates with IAM and supports social and enterprise identity providers.
- AWS Security Token Service (STS) issues temporary security credentials for controlled, short-term access to AWS resources.

These services ensure applications remain secure, scalable and compliant.

Tools / Services used

- Amazon Cognito (User Pools & Identity Pools)
- AWS Security Token Service (STS)
- AWS Management Console
- IAM Roles and Policies

Procedure / Steps

- Logged into the AWS Academy Learner Lab.

created an Amazon Cognito User Pool to manage user registration and login.

- Configured app clients and enabled authentication flow.
- Integrated AWS STS to issue temporary credentials for authorized users.
- Linked Cognito authentication with an application
- Tested sign up, login and token validation processes
- Completed knowledge check

Output / Result

- Successfully implemented user authentication with Amazon Cognito.
- Verified secure sign-up and sign-in functionality
- Observed temporary credential generation via AWS STS.

Conclusion

- This module provided practical knowledge of securing AWS applications using Cognito and STS. I learned how to manage user identities, enable secure access and integrate authentication features into real-world cloud applications.

DB
DB

Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

Lab 12.1: Implementing Application Authentication Using Amazon Cognito

Lab 12.1: Implementing Application Authentication Using Amazon Cognito

Due No Due Date Points 100 Submitting an external tool

31. At the top of these instructions, choose **Submit** to record your progress and when prompted, choose **Yes**.

Tip: If you previously hid the terminal in the browser panel, expose it again by selecting the **Terminal** checkbox. This action will ensure that the lab instructions remain visible after you choose **Submit**.

```
g the frank user!
bash
Testing report - Great job creating the API Gateway authorizer!
/usr/lib/python3.7/site-packages/boto3/compat.py:82: PythonDeprecationWarning: Boto3 will no longer support Python 3.7 starting December 13, 2023. To continue receiving service updates, bug fixes, and security updates please upgrade to Python 3.8 or later. More information can be found here: https://aws.amazon.com/blogs/developer/python-support-policy-updates-for-aws-sdks-and-tools/
warnings.warn(warning, PythonDeprecationWarning)
Back in submit.sh...
```

Total score 25/25

[Task 1] Confirmed SNS subscription
[Task 2] Created the Cognito user pool
[Task 3] Created Resource Server

Due No Due Date Points 100 Submitting an external tool

```
Submission Report
[Executed at: Sun Nov 9 23:03:49 PST 2025]

Testing report - Great job confirming the SNS subscription!
Testing report - Great job creating the Cognito user pool!
Testing report - Great job creating the resource server!
Testing report - Great job creating the frank user!
Testing report - Great job creating the API Gateway authorizer!

[CONFIRMATION] Confirmation of the task that this lab instructions remain visible after you choose Submit.
```

Source 25/25

SNS subscription
Cognito user pool
Resource server
API Gateway authorizer

[Task 3] Created Resource Server

Module 12 knowledge check results

Your score: 100% (100 points)

Required score: 70% (70 points)

Result: Congratulations! You have completed this module.
To continue, choose **Next** in the lower-right corner.

Automating Application Deployment Using a CI/CD Pipeline.

Objective

- To learn how to automate application deployment using AWS Continuous Integration and Continuous Delivery (CI/CD) services, ensuring faster and more reliable software delivery.

Theory

- DevOps Combine development and operations to improve collaboration, automation and delivery speed. AWS provides Managed DevOps tools such as
 - AWS CodeCommit - for source code management.
 - AWS CodeBuild - for building and testing applications.
 - AWS CodeDeploy - for automated deployment.
 - AWS CodePipeline - for ~~orchestration~~ orchestrating the entire CI/CD workflow

Together, they help implement a complete CI/CD lifecycle for modern cloud applications

Tools / Services Used

- AWS CodePipeline
- AWS CodeCommit
- AWS CodeBuild
- AWS CodeDeploy
- AWS CloudFormation / AWS SAM

Automating Application Deployment Using a CI/CD Pipeline.

Objective

- > To learn how to automate application deployment using AWS Continuous Integration and Continuous Delivery (CI/CD) services, ensuring faster and more reliable software delivery.

Theory

- > DevOps Combine development and operations to improve collaboration, automation and delivery speed. AWS provides managed DevOps tools such as
 - AWS CodeCommit - for source code management.
 - AWS CodeBuild - for building and testing applications.
 - AWS CodeDeploy - for automated deployment.
 - AWS CodePipeline - for ~~orchestration~~ orchestrating the entire CI/CD workflow

Together, they help implement a complete CI/CD lifecycle for modern cloud applications

Tools / Services Used

- > AWS CodePipeline
- > AWS CodeCommit
- > AWS CodeBuild
- > AWS CodeDeploy
- > AWS CloudFormation / AWS SAM

Procedure / Steps

Logged into the AWS Academy Learner Lab.

Created a CodeCommit repository and pushed sample application code.

Configured a CodeBuild project to compile and test the code.

Created a CodeDeploy application to automate deployment to EC2 or Lambda.

Set up a CodePipeline to integrate all stages (Source → Build → Deploy).

Deploy the application automatically through the pipeline.

Monitored build and deployment status in the AWS console.

Completed the module knowledge check.

Output

Successfully built and deployed an automated CI/CD pipeline using AWS DevOps tools.

Verified end-to-end automation from code commit to deployed.

Conclusion

~~This module provided hands-on experience in implementing DevOps automation using AWS CI/CD services. I learned how to streamline software delivery, improve deployment reliability, and maintain continuous integration practices for cloud-based applications.~~

80
12/11/26

Software Engineering Lab
Khurram Rashid, A70405223016, ASET Div- C, Sem- 5

Lab 13.1: Automating Application Deployment Using a CI/CD Pipeline

Lab 13.1: Automating Application Deployment Using a CI/CD Pipeline

Due No Due Date Points 100 Submitting an external tool

Submission Report

(Executed at: Mon Nov 16 1:21:48 PST 2025)

```
Testing report Task 1 - PROBLEM: The email subscription for the SMS topic has not been confirmed. Check your inbox for a message from AWS Notifications.
Testing report Task 2 - SUCCESS: Great job creating the CodeCommit repo!
Testing report Task 3 - SUCCESS: Great job creating the pipeline!
Testing report Task 6a - SUCCESS: Great job committing the website code to the repo!
Testing report Task 6b - SUCCESS: Great job pushing the website code to the repo!
```

Lab 13.1: Automating Application Deployment Using a CI/CD Pipeline

Due No Due Date Points 100 Submitting an external tool

Submission Report:

(Executed at: Mon Nov 16 1:21:48 PST 2025)

```
eee_W_5815312@runweb195438:~$ 
eee_W_5815312@runweb195438:~$ 
eee_W_5815312@runweb195438:~$
```

Total score 20/25

[Task 1] Confirmed SMS subscription
[Task 2] Created front_end_website

Module 13 knowledge check results

Your score: 80% (80 points)

Required score: 70% (70 points)

Result: Congratulations! You have completed this module.

To continue, choose **Next** in the lower-right corner.



Khurram Rashid

Certificate of Completion for

AWS Academy Graduate - Cloud Developing - Training Badge

Course hours completed

40 hours

Issued on

11/09/2025

Digital badge

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