AWS Lambda Layers for Code Reusability

A Course Project Report Submitted in partial fulfillment of the course requirements for the award of grades in the subject of

CLOUD BASED AIML SPECIALITY
(22SDCS07A)

by

Evani VSS Lalitha

(2210030372)

Under the esteemed guidance of

Ms. P. Sree Lakshmi
Assistant Professor,
Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K L Deemed to be UNIVERSITY

Aziznagar, Moinabad, Hyderabad,
Telangana, Pincode: 500075

April 2025

K L Deemed to be UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DEEMED TO BE UNIVERSITY)

Certificate

This is Certified that the project entitled **"AWS Lambda Layers for Code Reusability"** which is a experimental &/ Simulation work carried out by Evani VSS Lalitha (2210030372), in partial fulfillment of the course requirements for the award of grades in the subject of **CLOUD BASED AIML SPECIALITY**, during the year **2024-2025**. The project has been approved as it satisfies the academic requirements.

Ms.P.Sree Lakshmi	**Dr. Arpita Gupta**
Course Coordinator	**Head of the Department**

Ms. P. Sree Lakshmi

Course Instructor

CONTENTS

1. INTRODUCTION

AWS Lambda Layers are a powerful feature that helps improve code reusability, modularity, and maintainability in serverless applications. Instead of packaging the same libraries and dependencies with every Lambda function, Layers allow you to store shared code separately and reference it across multiple functions [3].

By using Lambda Layers, deployment size is significantly reduced, leading to faster function execution and easier updates. When a common dependency, such as a logging library or machine learning model, is needed in multiple functions, it can be placed in a Layer. This avoids redundancy and ensures consistency across all functions using the shared code [5].

AWS allows up to five Layers per Lambda function, providing flexibility in managing dependencies. When an update is needed, you simply modify the Layer without changing every individual function. This approach streamlines the development process and reduces operational overhead [6].

Lambda Layers are particularly useful in microservices-based architectures, where different functions often require the same utilities or configurations. They also support third-party dependencies, making integration with external tools seamless [7].

By leveraging AWS Lambda Layers, developers can create efficient, scalable, and maintainable serverless applications while simplifying dependency management and deployment [4].

2. AWS SERVICES USED AS PART OF THE PROJECT

To implement AWS Lambda Layers for code reusability, several AWS services have been utilized to ensure a seamless, efficient, and scalable solution. Below are the key AWS services used in the project:

AWS Lambda

AWS Lambda is a serverless computing service that allows users to run code in response to events without managing servers. In this project, Lambda functions are used to perform essential tasks such as:

- Creating an account: A function is triggered to create new user accounts when needed.
- Getting account balance: A function retrieves and displays the current balance associated with an account.
- Updating account balance: Another function is responsible for modifying the balance based on transactions or inputs.

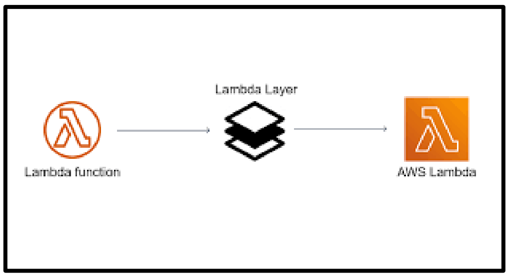
These Lambda functions leverage reusable code through Lambda Layers, reducing redundancy and improving modularity [3].

AWS Lambda Layers

Lambda Layers provide a mechanism for sharing common code and dependencies among multiple Lambda functions. This helps to:

- Reduce deployment package size by keeping shared libraries separate.
- Improve maintainability by centralizing code that multiple functions use.

- Enhance execution efficiency as functions can load dependencies from layers instead of bundling them within the function code [5].



AWS Management Console

The AWS Management Console is used as a web-based interface to create, configure, and manage AWS Lambda functions and layers. It provides a graphical user interface (GUI) to:

- Create and edit Lambda functions.
- Deploy Lambda Layers.
- Monitor function performance and logs.
- Manage AWS resources efficiently [6].

Amazon S3 (Simple Storage Service)

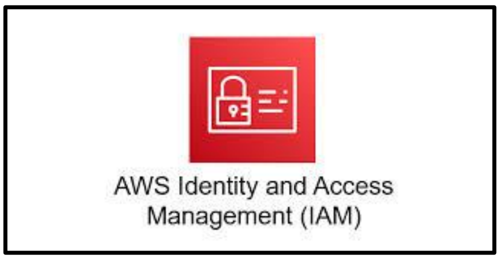
Amazon S3 is used for storing and managing resources such as:

- Lambda Layer dependencies: Large libraries and dependencies are stored in an S3 bucket and retrieved when needed.
- Function logs and outputs: Intermediate outputs or logs can be stored and accessed later [2].

AWS IAM (Identity and Access Management)

IAM is used to define permissions and security policies to control access to AWS services. IAM roles are assigned to:

- Lambda functions to grant permissions to access other AWS resources.
- Users to manage and deploy Lambda functions securely [7].



Amazon CloudWatch

CloudWatch is a monitoring and logging service used to:

- Track Lambda function executions and performance.
- Collect logs and error messages for debugging.
- Set up alarms and notifications in case of failures or performance issues [5].

By integrating these AWS services, the project ensures a highly efficient, modular, and scalable architecture for managing multiple serverless functions with reusable code.

3. STEPS INVOLVED IN SOLVING PROJECT PROBLEM STATEMENT

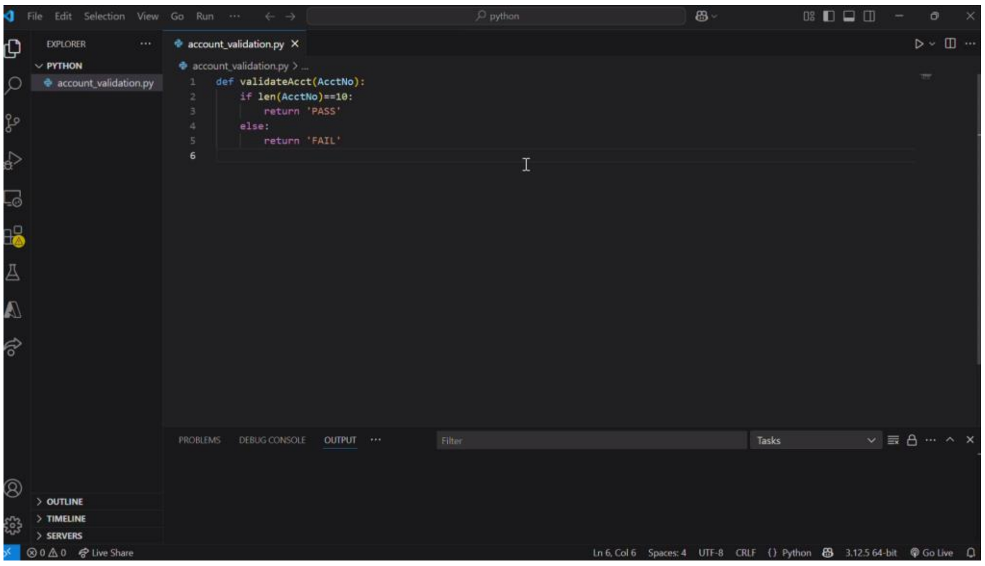
To set up AWS Lambda Layers for code reusability:

1. Create Your Code: Write the common code (e.g., account_validation.py) and ensure its structured properly [1].
2. Zip Your Code: Compress the folder containing your Python code into a .zip file [2].
3. Access AWS Lambda Console:
    - Go to the AWS Management Console.
    - Navigate to the Lambda section.
    - Click on Layers and then Create Layer [3].
4. Upload the Layer:
    - Choose the option to upload as a zip file.
    - Select the zip file you created earlier.
    - Configure compatible runtimes (e.g., Python 3.7, 3.8, 3.9).
    - Click Create to create the layer [2].
5. Create a New Lambda Function:
    - Click on Functions and then Create Function [3].
    - Choose a runtime (e.g., Python 3.9) and set permissions as needed.
6. Add the Layer to the Function:
    - In your Lambda function configuration, find the Layers section.
    - Click on Add a layer [5].
    - Select your custom layer from the list and choose the appropriate version.
7. Reference the Layer in Your Code: Update your Lambda function code to reference the functions defined in the layer [1].
8. Test and Monitor the Function:
    - Invoke the Lambda function via AWS CLI or test events in the AWS console.
    - Check the logs in Amazon CloudWatch for debugging and monitoring [5].

4. STEPWISE SCREENSHOTS WITH BRIEF DESCRIPTION

Step 1: Creating account_validation.py for Lambda Layer

- Open VS Code → Create account_validation.py under "python" folder.
- Zip the "python" folder to upload as a Lambda Layer.



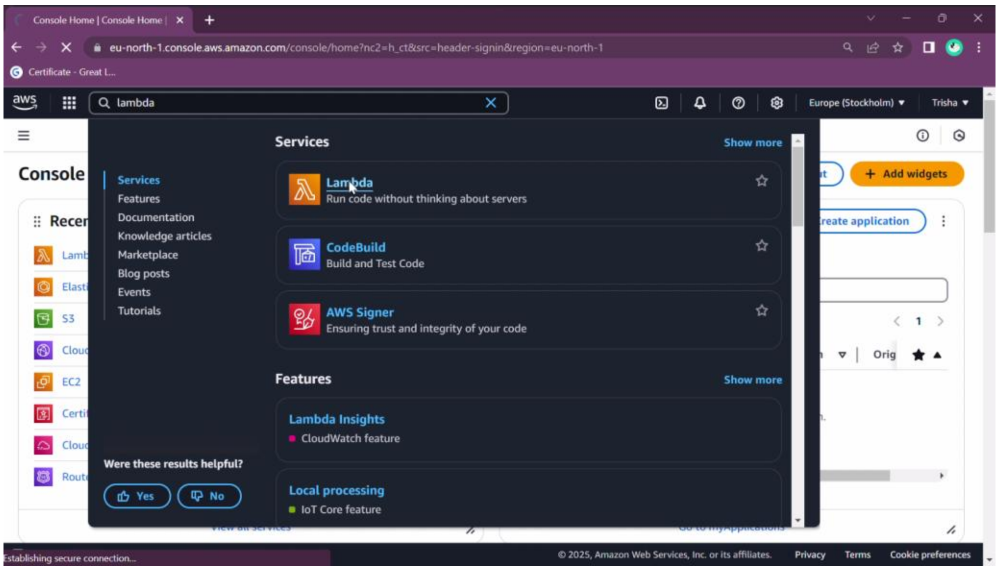
```python
def validateAcct(AcctNo):
    if len(AcctNo)==10:
        return 'PASS'
    else:
        return 'FAIL'
```

File Edit Selection View Go Run
python
EXPLORER
PYTHON
account_validation.py
PROBLEMS DEBUG CONSOLE OUTPUT
OUTLINE
TIMELINE
SERVERS
Ln 6, Col 6 Spaces: 4 UTF-8 CRLF Python 3.12.5 64-bit Go Live

Fig 4.1 Creating account_validation.py for Lambda Layer

Step 2: Accessing AWS Lambda Service

- Log in to AWS Management Console.
- Search for "Lambda" and open the service.

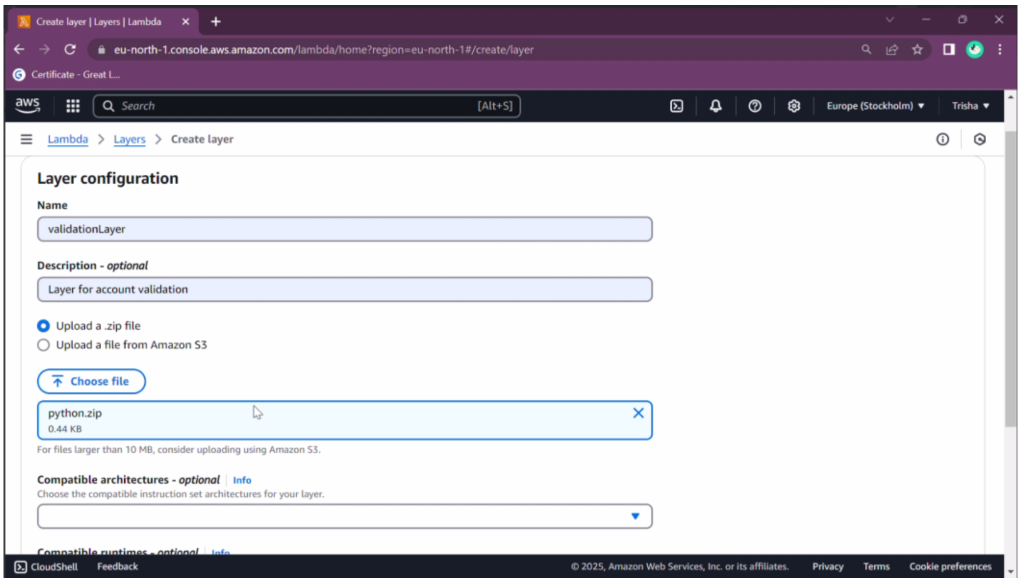


Console Home | Console Home
eu-north-1.console.aws.amazon.com/console/home?nc2=h_ct&src=header-signin&region=eu-north-1
lambda
Europe (Stockholm)
Trisha
Services
Show more
Services
Features
Documentation
Knowledge articles
Marketplace
Blog posts
Events
Tutorials
Lambda
Run code without thinking about servers
CodeBuild
Build and Test Code
AWS Signer
Ensuring trust and integrity of your code
Features
Lambda Insights
CloudWatch feature
Local processing
IoT Core feature
Were these results helpful?
Yes
No
Add widgets
Console
Establishing secure connection...
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Fig 4.2 Accessing AWS Lambda Service

Step 3: Creating a Lambda Layer

- Go to AWS Lambda → Click "Layers" in the sidebar.
- Click "Create layer".
- Enter Layer name and Description.
- Click "Upload" → Select the .zip file containing your Python code.
- Choose compatible runtimes (e.g., Python 3.x, Node.js).
- Click "Create" to finalize the layer.


Create layer | Layers | Lambda
eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/create/layer
Certificate - Great L...
Search
[Alt+S]
Europe (Stockholm)
Trisha
Lambda > Layers > Create layer
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Layer configuration

Name

validationLayer

Description - *optional*

Layer for account validation

● Upload a .zip file
○ Upload a file from Amazon S3

Choose file

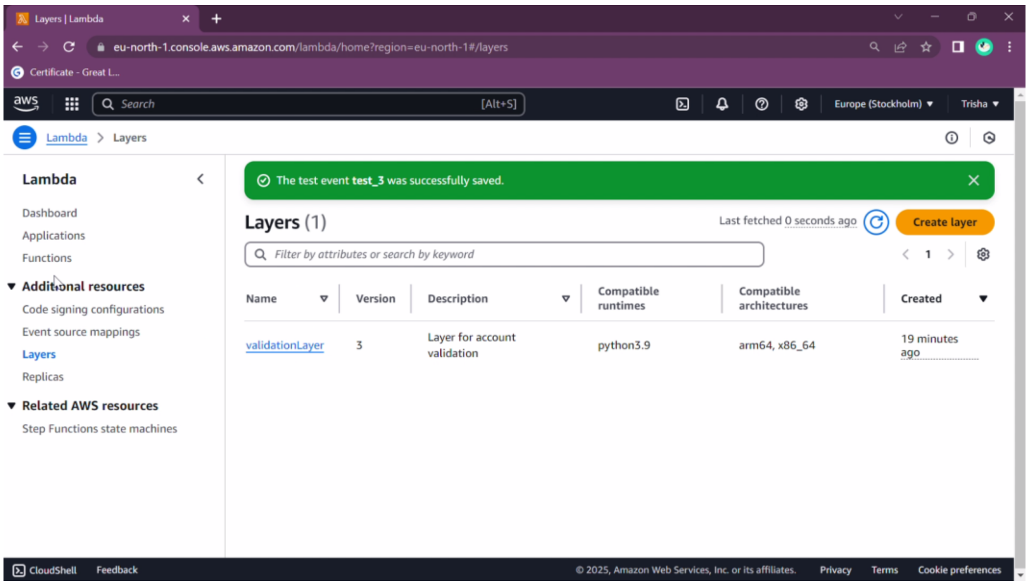
python.zip
0.44 KB

For files larger than 10 MB, consider uploading using Amazon S3.

Compatible architectures - *optional* Info
Choose the compatible instruction set architectures for your layer.

Compatible runtimes - *optional* Info

Fig 4.3.1 Uploading python.zip to Create Lambda Layer



The test event test_3 was successfully saved.

Layers (1)

Last fetched 0 seconds ago

Create layer

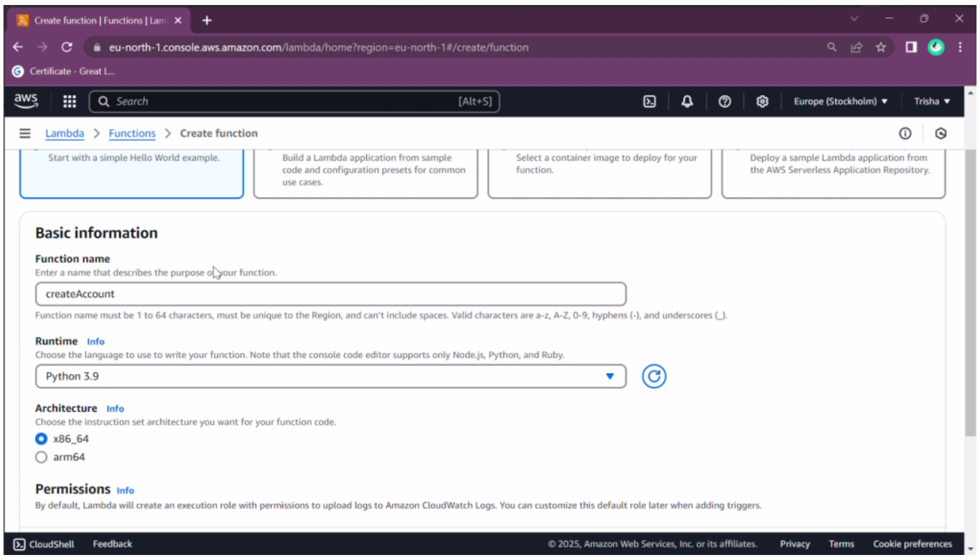
Filter by attributes or search by keyword

Name	Version	Description	Compatible runtimes	Compatible architectures	Created
validationLayer	3	Layer for account validation	python3.9	arm64, x86_64	19 minutes ago

Fig 4.3.2 Successfully Created Lambda Layer for Validation

Step 4: Creating New Lambda Function

- Go to AWS Lambda → Click "Functions" in the sidebar → Click "Create Function".
- Enter Function name and Runtime.
- Click "Create" to finalize the function.


Create function | Functions | Lam
eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/create/function
Certificate - Great L...
Search
[Alt+S]
Europe (Stockholm)
Trisha
Lambda > Functions > Create function
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Start with a simple Hello World example.

Build a Lambda application from sample code and configuration presets for common use cases.

Select a container image to deploy for your function.

Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

Function name
Enter a name that describes the purpose of your function.
createAccount
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9

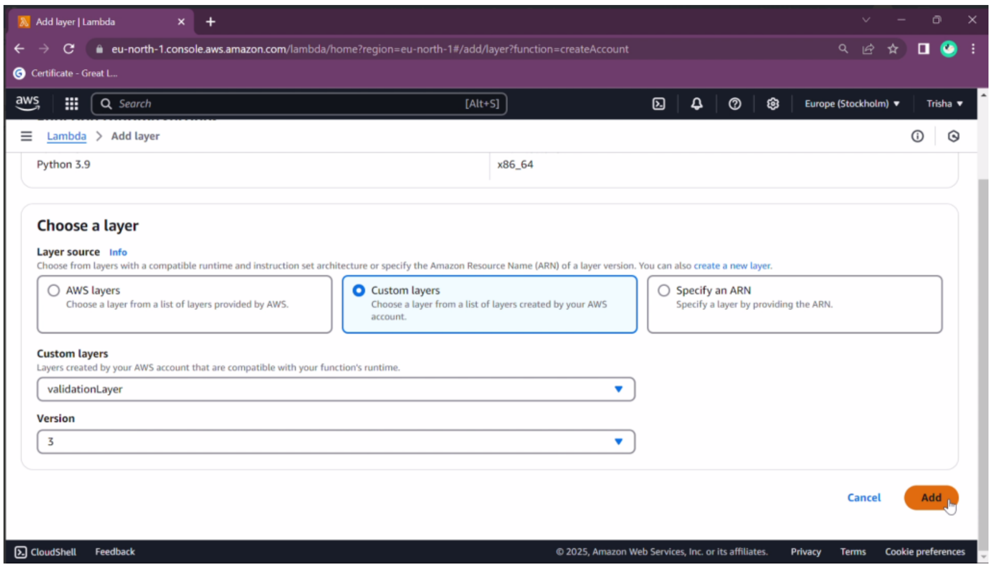
Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64]
arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Fig 4.4 Entering Basic Information to Create New Lambda Function

Step 5: Attaching Custom Lambda Layer to Function

- Go to the "Layers" section in your function.
- Click "Add a layer", choose the created layer.



Python 3.9 x86_64

Choose a layer

Layer source Info
Choose from layers with a compatible runtime and instruction set architecture or specify the Amazon Resource Name (ARN) of a layer version. You can also create a new layer.

AWS layers
Choose a layer from a list of layers provided by AWS.

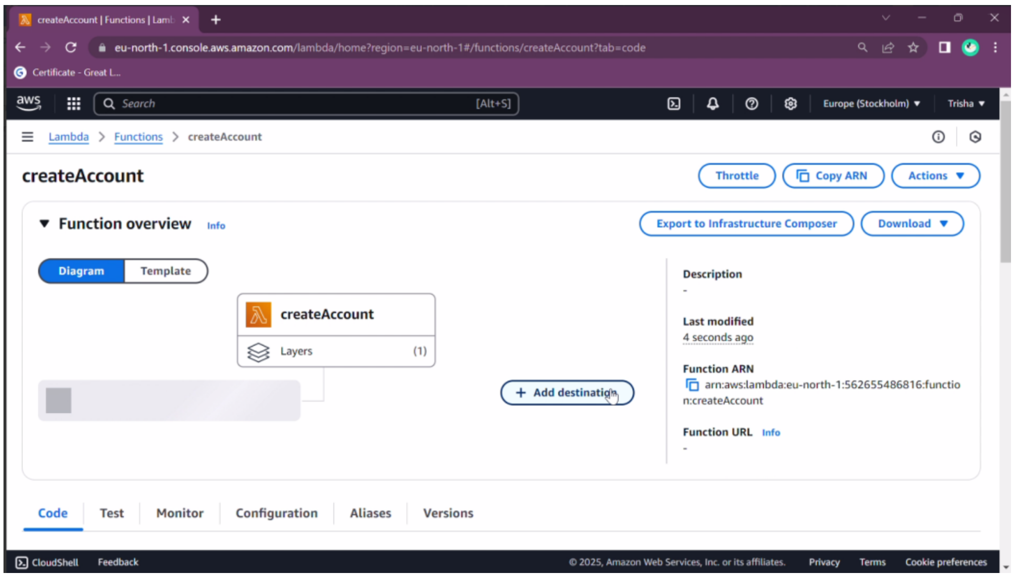
[✓ Custom layers]
Choose a layer from a list of layers created by your AWS account.

Specify an ARN
Specify a layer by providing the ARN.

Custom layers
Layers created by your AWS account that are compatible with your function's runtime.
validationLayer

Version
3

Fig 4.5.1 Attaching Custom Lambda Layer to Function


createAccount

Function overview

Description
-

Last modified
4 seconds ago

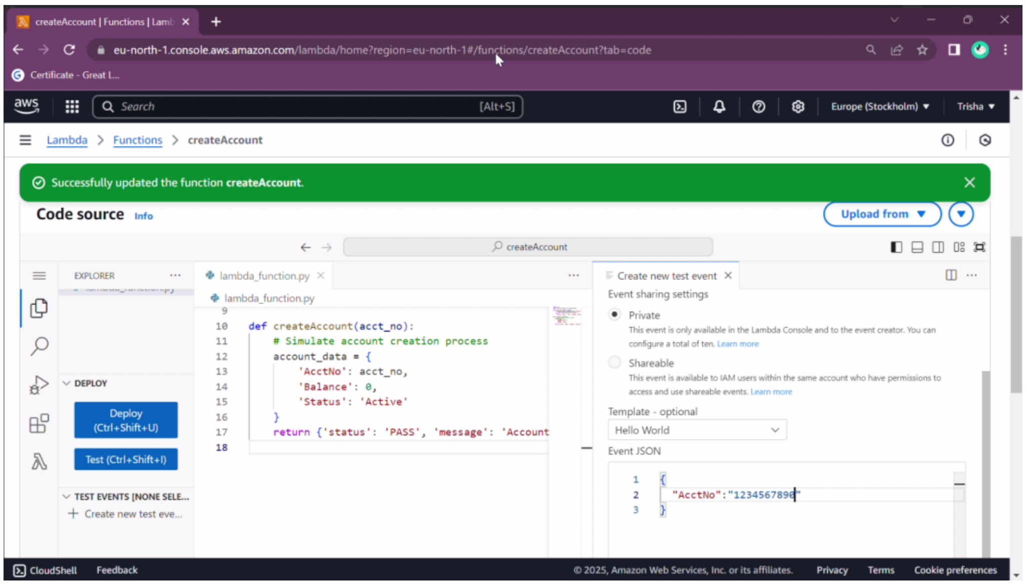
Function ARN
arn:aws:lambda:eu-north-1:562655486816:function:createAccount

Function URL
-

Fig 4.5.2 Lambda Function 'createAccount' Successfully Created

Step 6: Execution of CreateAccount Lambda Function

- Click "Deploy" to update the Lambda function with the new code and layer.
- Click "Test".
- Configure a test event (e.g., a sample JSON request) → Put AcctNo=”1234567890”
- Run the test and check the Execution results to confirm the function is successfully using the layer.


Successfully updated the function createAccount.

```python
def createAccount(acct_no):
    # Simulate account creation process
    account_data = {
        'AcctNo': acct_no,
        'Balance': 0,
        'Status': 'Active'
    }
    return {'status': 'PASS', 'message': 'Account
```

Create new test event

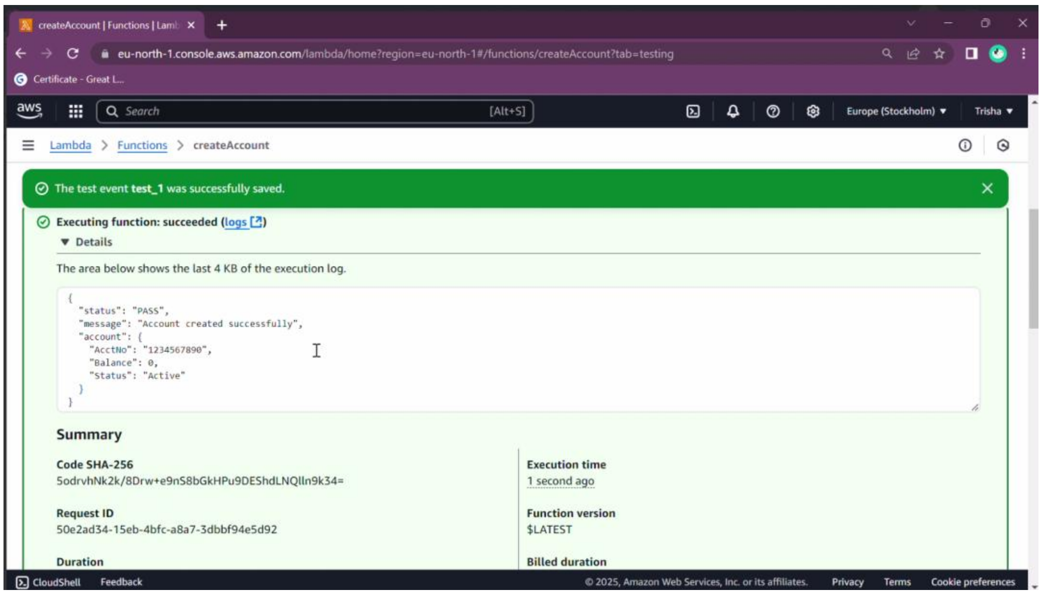
Event sharing settings: Private

Template - optional: Hello World

Event JSON

```json
{
 "AcctNo":"1234567890"
}
```

Fig 4.6.1 Writing and Saving Lambda Function Code



eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/functions/createAccount?tab=testing

Lambda > Functions > createAccount

The test event **test_1** was successfully saved.

Executing function: succeeded (logs)

Details

The area below shows the last 4 KB of the execution log.

```
{
  "status": "PASS",
  "message": "Account created successfully",
  "account": {
    "AcctNo": "1234567890",
    "Balance": 0,
    "Status": "Active"
  }
}
```

Summary

Code SHA-256
5odrvhNk2k/8Drw+e9nS8bGkHPu9DEShdLNQlln9k34=

Execution time
1 second ago

Request ID
50e2ad34-15eb-4bfc-a8a7-3dbbf94e5d92

Function version
$LATEST

Duration

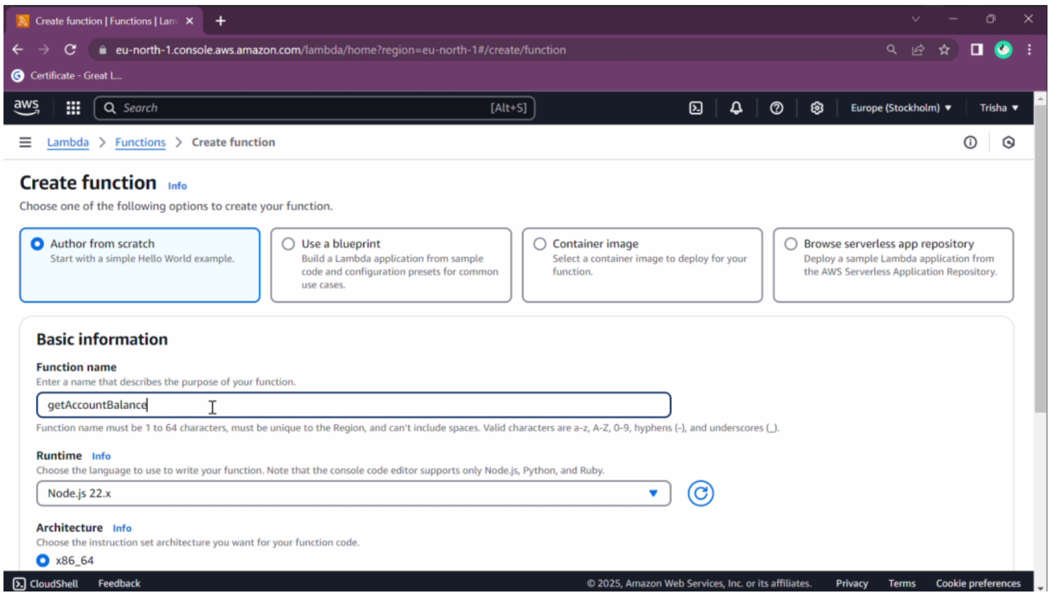
Billed duration

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

Fig 4.6.2 Successful Execution of Lambda Function with Output

Step 7: Creating a Function for getAccountBalance

- Again, create function for getAccountBalance .
- Click "Add a layer".



eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/create/function

Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

[✓ Author from scratch] Start with a simple Hello World example.

Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.

Container image: Select a container image to deploy for your function.

Browse serverless app repository: Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

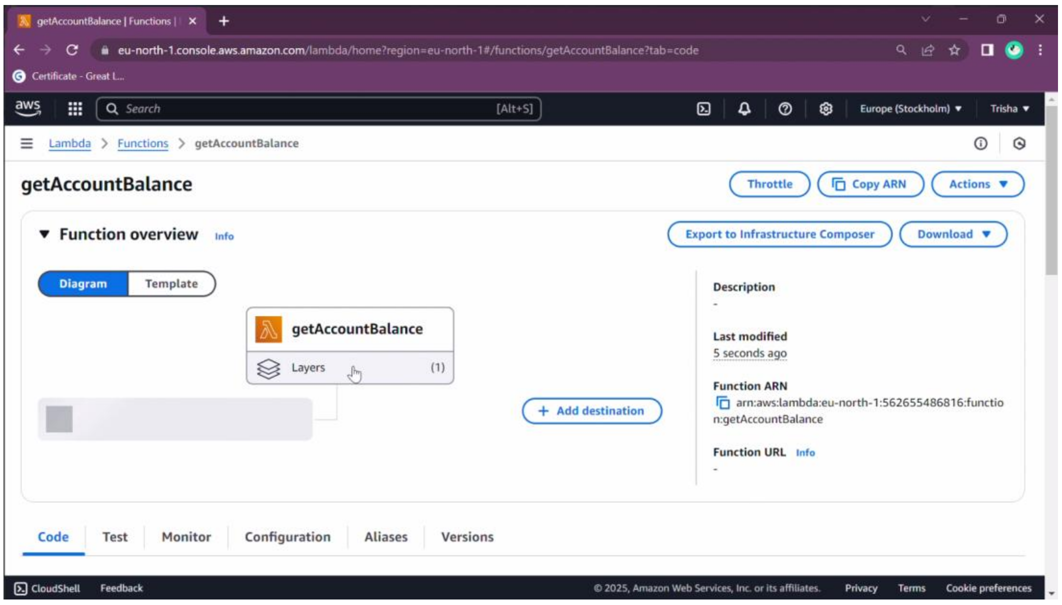
Function name
Enter a name that describes the purpose of your function.
getAccountBalance
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Node.js 22.x

Architecture Info
Choose the instruction set architecture you want for your function code.
[✓ x86_64]

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

Fig 4.7.1 Creating a Function for ‘getAccountBalance’


getAccountBalance

Description
-

Last modified
5 seconds ago

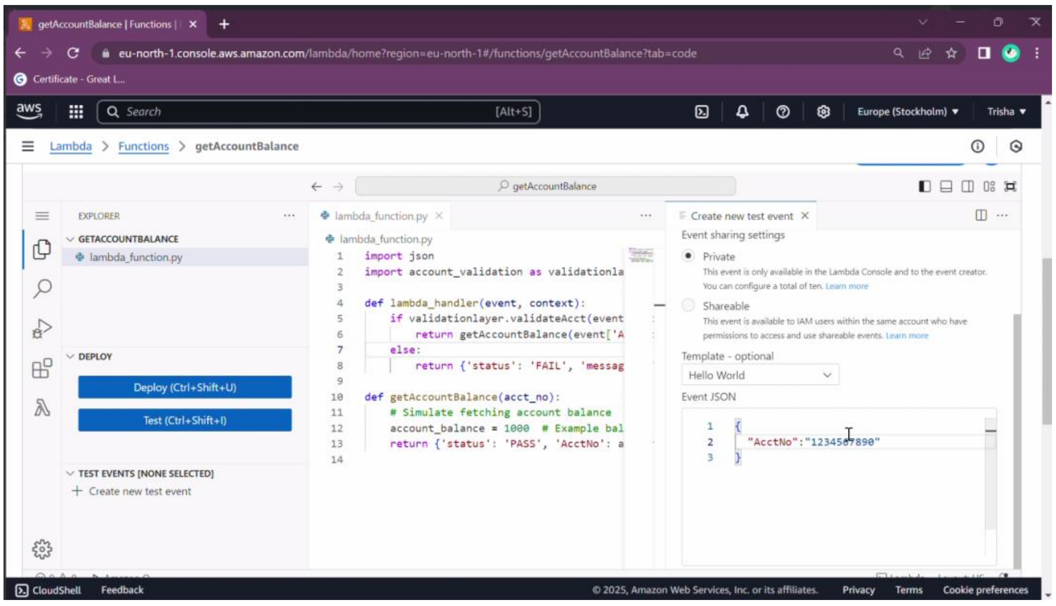
Function ARN
arn:aws:lambda:eu-north-1:562655486816:function:getAccountBalance

Function URL
-

Fig 4.7.2 Lambda Function 'getAccountBalance' Successfully Created

Step 8: Execution of getAccountBalnce Lambda Function

- Choose the created layer. → Deploy → Test
- Configure a test event (e.g., a sample JSON request) → Put AcctNo=”1234567890”
- Run the test and check the Execution results to confirm the function is successfully using the layer.


```python
import json
import account_validation as validationla

def lambda_handler(event, context):
    if validationlayer.validateAcct(event
        return getAccountBalance(event['A
    else:
        return {'status': 'FAIL', 'messag

def getAccountBalance(acct_no):
    # Simulate fetching account balance
    account_balance = 1000  # Example bal
    return {'status': 'PASS', 'AcctNo': a
```

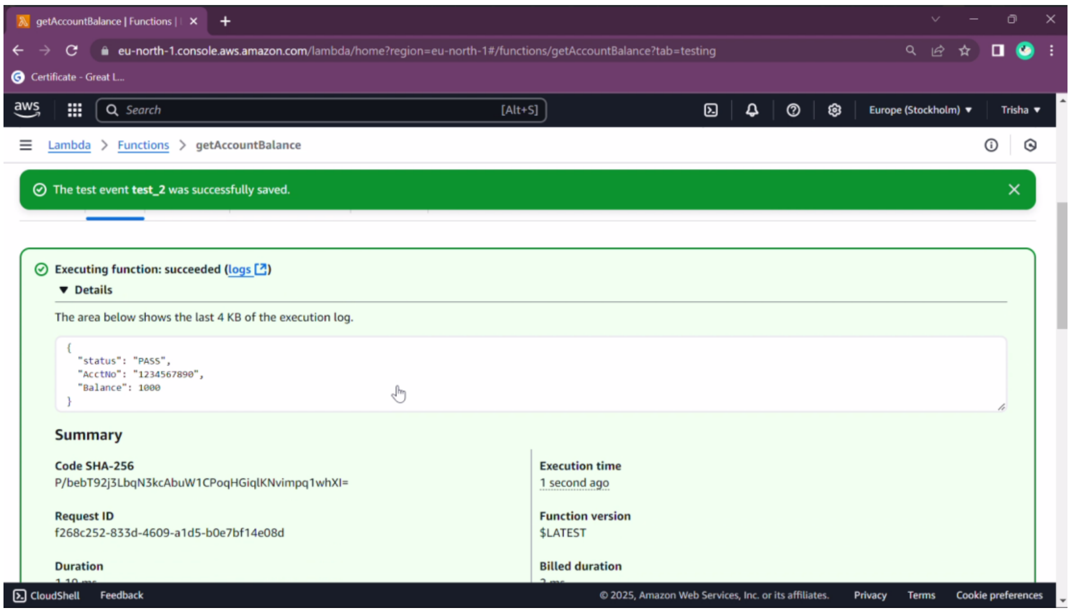
Event sharing settings: Private

Template - optional: Hello World

Event JSON

```json
{
  "AcctNo":"1234567890"
}
```

Fig 4.8.1 Creating a Function for ’updateAccountBalance’



The test event **test_2** was successfully saved.

Executing function: succeeded (logs)

Details

The area below shows the last 4 KB of the execution log.

```
{
  "status": "PASS",
  "AcctNo": "1234567890",
  "Balance": 1000
}
```

Summary

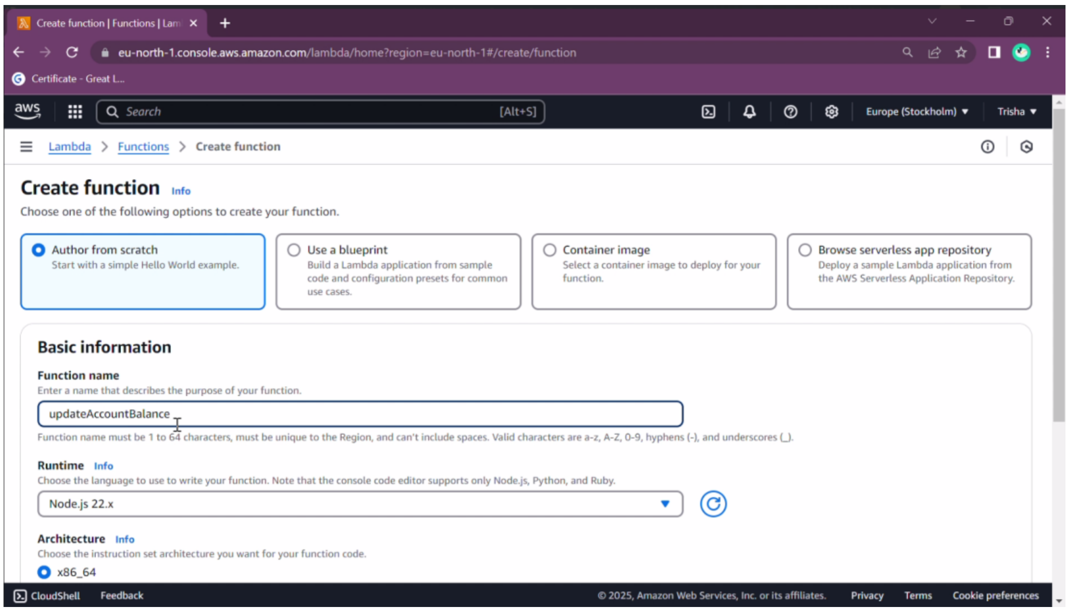
Code SHA-256 P/bebT92j3LbqN3kcAbuW1CPoqHGiqlKNvimpq1whXI=	**Execution time** 1 second ago
Request ID f268c252-833d-4609-a1d5-b0e7bf14e08d	**Function version** $LATEST
Duration	**Billed duration**

getAccountBalance | Functions
eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/functions/getAccountBalance?tab=testing
Certificate - Great L...
Search
[Alt+S]
Europe (Stockholm)
Trisha
Lambda > Functions > getAccountBalance
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Fig 4.8.2 Lambda Function 'updateAccountBalance' Successfully Created

Step 9: Creating a new Lambda Function for UpdateAccountBalance

- Again, create function for updateAccountBalance
- Click "Add a layer".



Create function

Choose one of the following options to create your function.

- [✓ Author from scratch] Start with a simple Hello World example.
- Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.
- Container image: Select a container image to deploy for your function.
- Browse serverless app repository: Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

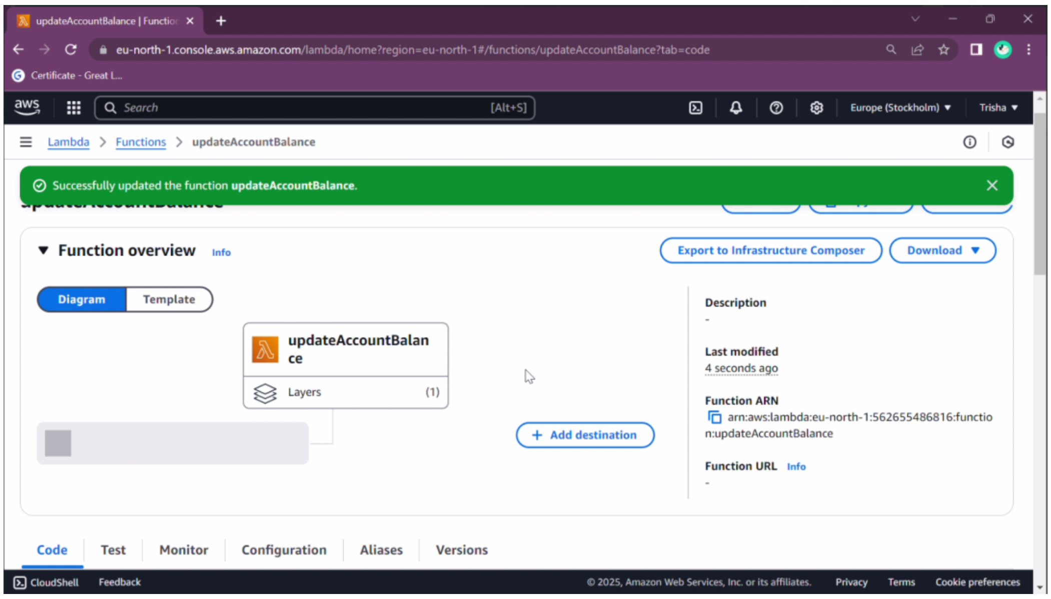
Function name: Enter a name that describes the purpose of your function.
updateAccountBalance
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime: Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Node.js 22.x

Architecture: Choose the instruction set architecture you want for your function code.
[✓ x86_64]

Create function | Functions | Lam
eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/create/function
Certificate - Great L...
Search
[Alt+S]
Europe (Stockholm)
Trisha
Lambda > Functions > Create function
Info
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Fig 4.9.1 Creating a New AWS Lambda Function Named 'updateAccountBalance'


Successfully updated the function updateAccountBalance.

Function overview

Description
-

Last modified
4 seconds ago

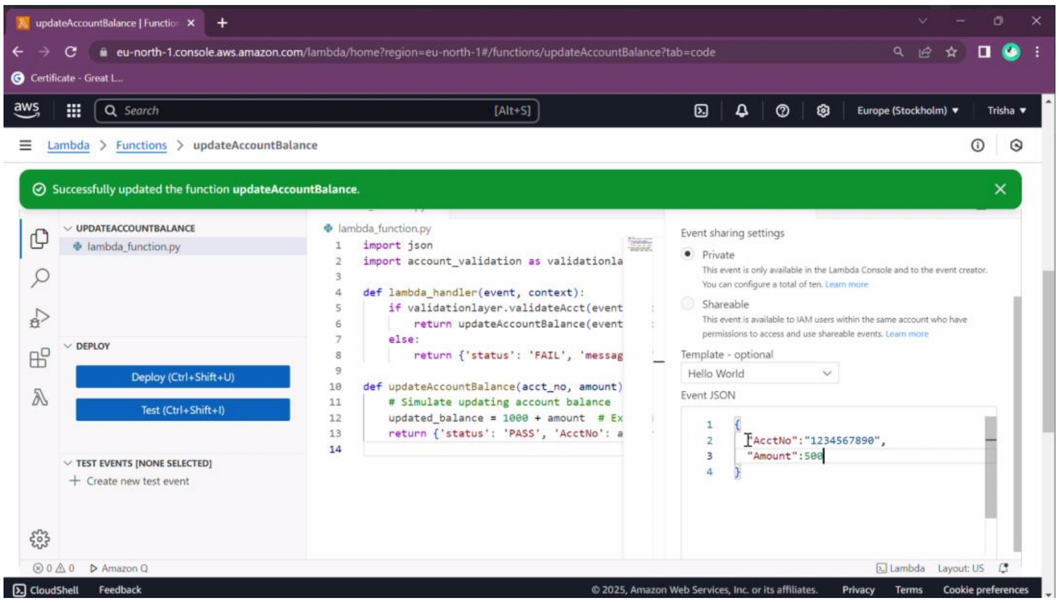
Function ARN
arn:aws:lambda:eu-north-1:562655486816:function:updateAccountBalance

Function URL
-

Fig 4.9.2 AWS Lambda Function 'updateAccountBalance' Created Successfully

Step 10: Execution Result of UpdateAccountBalance Function

- Choose the created layer→ Deploy → Test
- Configure a test event (e.g., a sample JSON request) → Put AcctNo="1234567890", Amount=500
- Run the test and check the Execution results to confirm the function is successfully using the layer.


Successfully updated the function updateAccountBalance.

lambda_function.py

```
import json
import account_validation as validationla

def lambda_handler(event, context):
    if validationlayer.validateAcct(event
        return updateAccountBalance(event
    else:
        return {'status': 'FAIL', 'messag

def updateAccountBalance(acct_no, amount)
    # Simulate updating account balance
    updated_balance = 1000 + amount  # Ex
    return {'status': 'PASS', 'AcctNo': a
```

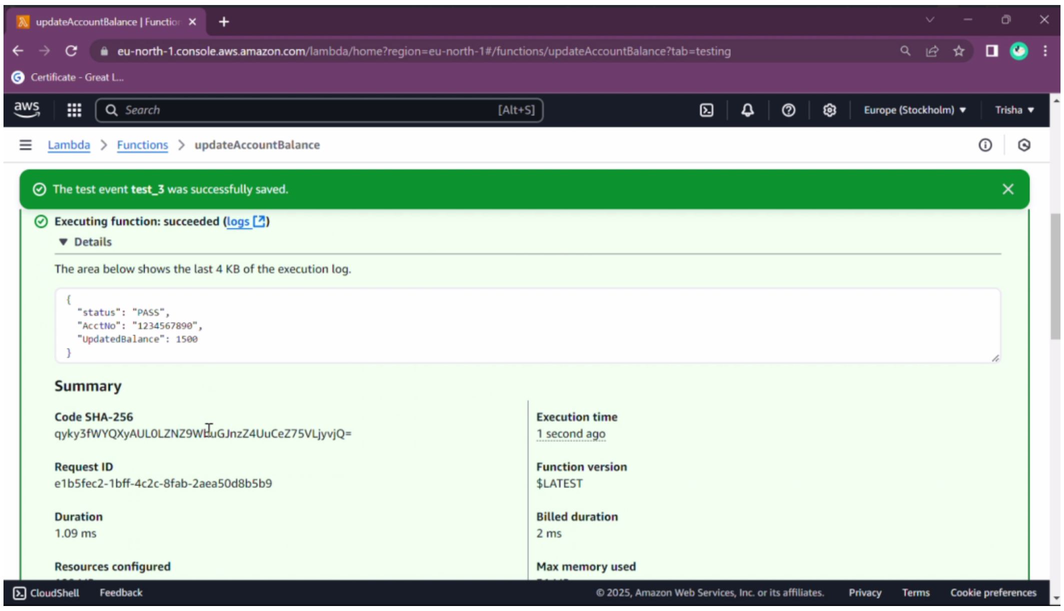
Event sharing settings: Private

Template - optional: Hello World

Event JSON

```
{
 "AcctNo":"1234567890",
 "Amount":500
}
```

Fig 4.10.1 Lambda Function Code for Validation and Test Input



The test event **test_3** was successfully saved.

Executing function: succeeded (logs)

Details

The area below shows the last 4 KB of the execution log.

```
{
  "status": "PASS",
  "AcctNo": "1234567890",
  "UpdatedBalance": 1500
}
```

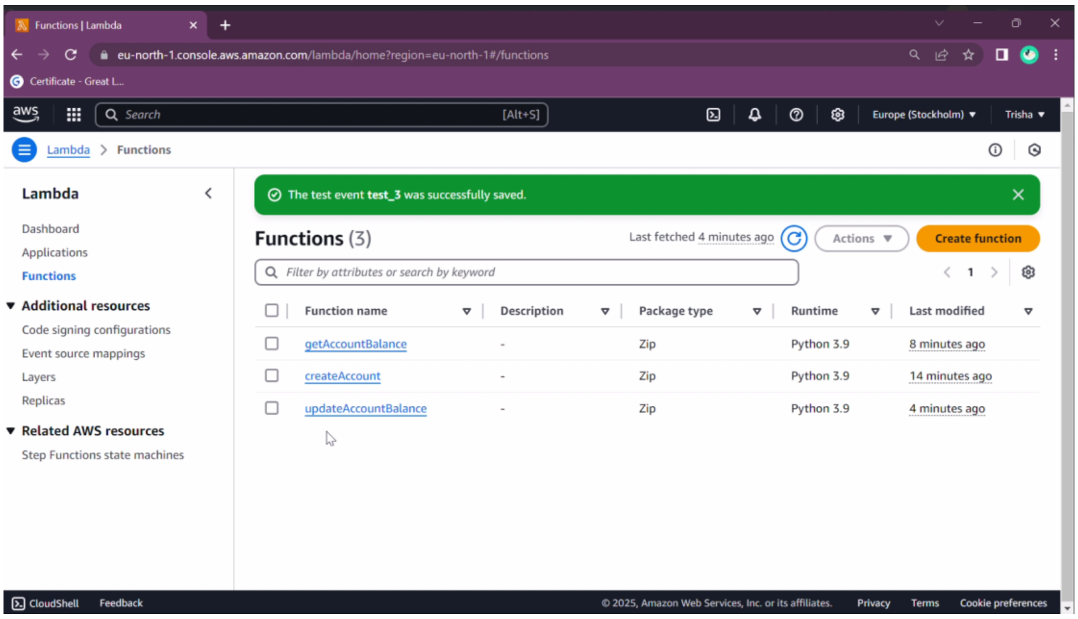
Summary

Code SHA-256 qyky3fWYQXyAUL0LZNZ9WEuGJnzZ4UuCeZ75VLjyvjQ=	Execution time 1 second ago
Request ID e1b5fec2-1bff-4c2c-8fab-2aea50d8b5b9	Function version $LATEST
Duration 1.09 ms	Billed duration 2 ms
Resources configured	Max memory used

Fig 4.10.2 Execution Result of Validation Function

Step 11: List of Created Lambda Functions in AWS Console

- Three Functions are created namely ‘getAccountBalance’,’createAccount’ and ‘updateAccountBalance’.
- These Functions are added to the Lambda Layer.



The test event **test_3** was successfully saved.

Functions (3)

Last fetched 4 minutes ago

Function name	Description	Package type	Runtime	Last modified
getAccountBalance	-	Zip	Python 3.9	8 minutes ago
createAccount	-	Zip	Python 3.9	14 minutes ago
updateAccountBalance	-	Zip	Python 3.9	4 minutes ago

Fig 4.11 List of Created Lambda Functions in AWS Console

5. LEARNING OUTCOMES

Understanding AWS Lambda Layers:

Gained insights into how Lambda Layers improve code reuse, deployment efficiency, and modularity in serverless applications [1].

Practical Implementation of Code Reusability:

Learned how to separate common dependencies into reusable Lambda Layers to optimize function development [2].

Efficient Dependency Management:

Understood how storing dependencies in layers minimizes Lambda deployment package sizes and speeds up execution [7].

Enhancing Security with IAM:

Gained knowledge on managing permissions using AWS IAM to securely access AWS services from Lambda functions [5].

Debugging and Monitoring with CloudWatch:

Learned how to track function execution, identify errors, and optimize performance using Amazon CloudWatch logs and metrics [2,5,7].

Using AWS Cloud9 for Development:

Explored Cloud9 as an IDE for writing, testing, and debugging AWS Lambda functions efficiently [3].

Integration of AWS Services:

Understood how different AWS services like S3, IAM, and CloudWatch work together to create an optimized serverless workflow [2,5].

Improved Development Workflow:

Gained experience in using AWS Management Console and AWS CLI for efficient serverless function management [3].

6. CONCLUSION

AWS Lambda Layers play a crucial role in enhancing serverless application development by enabling code reusability, reducing redundancy, and improving execution efficiency. By separating shared dependencies from individual Lambda functions, developers can minimize deployment package sizes, leading to faster execution and reduced cold-start times. The integration of AWS services like IAM for secure access control, CloudWatch for monitoring and logging, and Amazon S3 for efficient storage management further strengthens Lambda-based applications, making them more scalable and manageable. Additionally, AWS Cloud9 provides a seamless development environment for writing, testing, and debugging Lambda functions, while the AWS Management Console offers an intuitive interface for resource monitoring and configuration. By adopting Lambda Layers, organizations can lower maintenance costs, enhance application performance, and achieve greater agility in deploying and managing serverless workloads. As serverless computing continues to evolve, AWS Lambda Layers will remain an essential component in building highly scalable, cost-effective, and maintainable cloud-native applications.

7. FUTURE ENHANCEMENTS

In the future, this project can be improved by adding automatic updates for Lambda layers using CI/CD tools, so developers don’t have to update them manually. A shared library of commonly used layers can also be created to save time and effort. Testing can be added to check if the layers work correctly before using them in Lambda functions. We can also track how often layers are used and how they affect performance and cost. These changes will make the project more useful, easier to manage, and helpful for building bigger serverless applications.

7. REFERENCES

[1] Amazon Web Services (AWS). "Building Intelligent Chatbots with AWS Lambda and Amazon Lex." Available at: https://aws.amazon.com/lex/

[2] Amazon Web Services (AWS). "Tutorial: Using an Amazon S3 trigger to create thumbnail images." Available at: https://docs.aws.amazon.com/lambda/latest/dg/with-s3-tutorial.html

[3] Amazon Web Services (AWS). "Using API Gateway and AWS Lambda for Serverless Applications." Available at: https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

[4] Amazon Web Services (AWS). "Building serverless applications with streaming data." Available at: https://aws.amazon.com/blogs/compute/building-serverless-applications-with-streaming-data-part-4/

[5] Amazon Web Services (AWS). "Monitoring AWS Lambda Functions with Amazon CloudWatch." Available at: https://docs.aws.amazon.com/lambda/latest/dg/monitoring-cloudwatch.html

[6] Amazon Web Services (AWS). "Automating Database Tasks Using AWS Lambda and Amazon RDS." Available at: https://aws.amazon.com/rds/features/

[7] Amazon Web Services (AWS). "Event-Driven Architecture with Amazon SNS and SQS." Available at: https://docs.aws.amazon.com/sns/latest/dg/welcome.html

[8] Amazon Web Services (AWS). "Building IoT Applications Using AWS IoT Core and Lambda." Available at: https://aws.amazon.com/iot-core/