



UNIVERSITY OF PETROLEUM AND ENERGY
STUDIES
DEHRADUN

DISTRIBUTED SYSTEMS

Serverless Computing with AWS Lambda

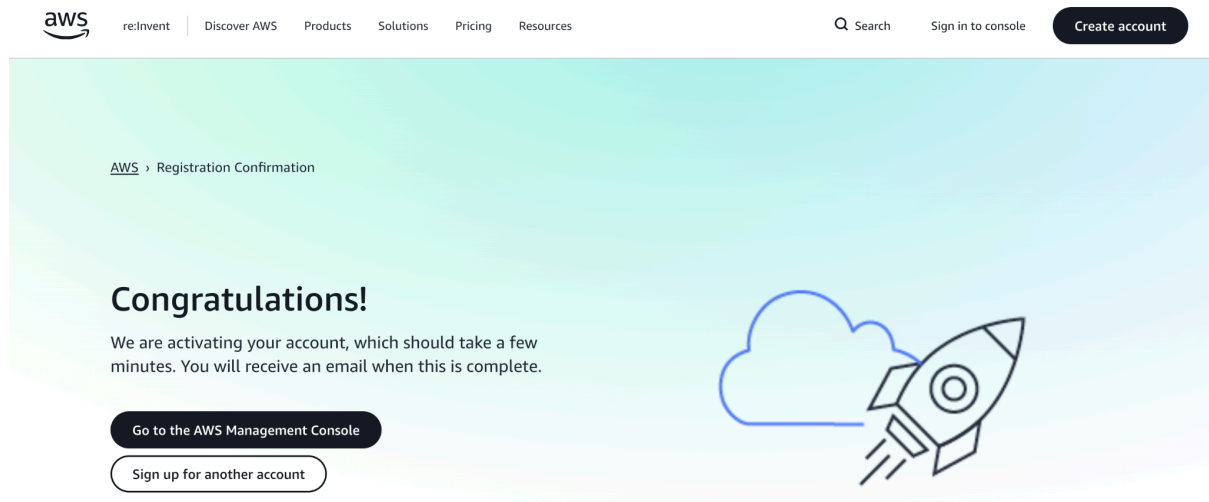
MTECH-COMPUTER SCIENCE
ENGINEERING
CYBER SECURITY AND FORENSICS

Name: Jigesh Sheoran
SAP ID: 590025428

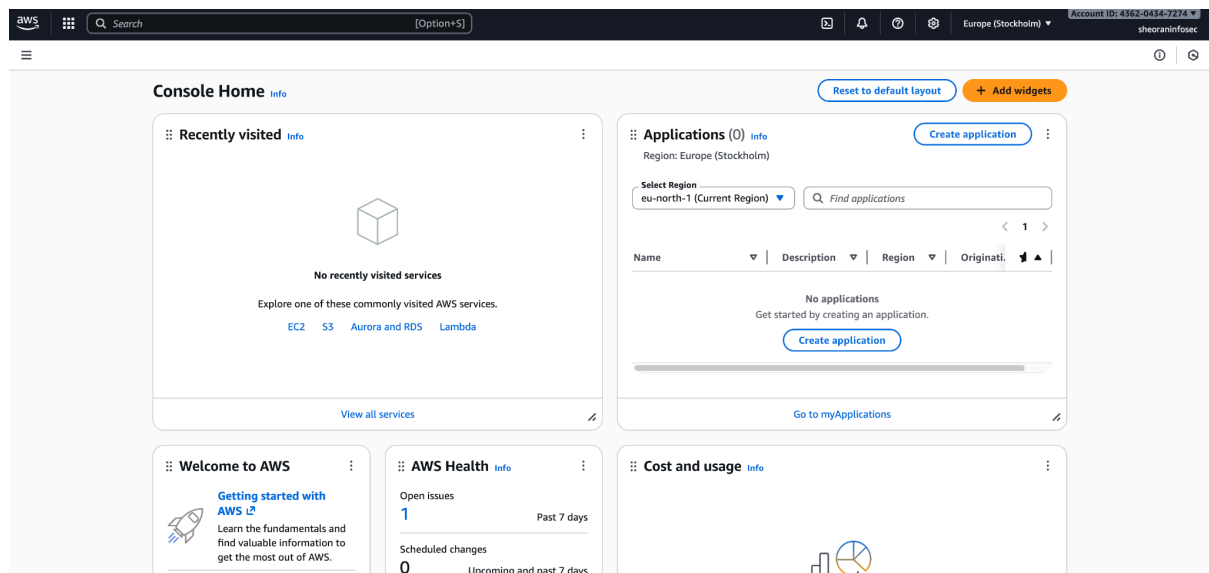
Objective — Develop and deploy a serverless function using AWS Lambda. Write a simple function triggered by an event (e.g., HTTP request) to understand the principles of serverless computing and its use in cloud-based applications.

Step 1: Set up a free tier AWS Account.

Step 2: Go to: <https://aws.amazon.com> / Select Create AWS Account / Choose Personal Account / Select Free Tier



Step 3: Login in to your AWS Console once you receive confirmation email.



Step 3: Create a IAM User (to gain CLI Access)

IAM → Users → Create User (Username: **lambda-student)**

Step 4: Then generate Access Keys under

(IAM → Users → **lambda-student → Security Credentials → Create Access Key)**

Step 5: Save the access keys to somewhere safe. Will need them later to access your aws account through Command Line Interface.

Review and create
Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name lambda-student	Console password type None	Require password reset No
------------------------------------	--------------------------------------	-------------------------------------

Permissions summary

Name	Type	Used as
AmazonAPIGatewayAdministrator	AWS managed	Permissions policy
AWSLambda_FullAccess	AWS managed	Permissions policy
CloudWatchLogsFullAccess	AWS managed	Permissions policy
IAMFullAccess	AWS managed	Permissions policy

Tags - optional
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.
No tags associated with the resource.

[Add new tag](#)
You can add up to 50 more tags.

[Cancel](#) [Previous](#) [Create user](#)

Step 6: Install AWS CLI on macOS.

```
sheoraninfosec — zsh — 95x32

Last login: Tue Nov 25 20:47:22 on ttys000
[sheoraninfosec@Jigeshs-MacBook-Air ~ % curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
 100 46.4M  100 46.4M    0     0 5534k      0  0:00:08  0:00:08 --:--:-- 5598k
sheoraninfosec@Jigeshs-MacBook-Air ~ %
```

Step 7: Verify if it installed correctly by checking the version.

```
sheoraninfosec — zsh — 95x32

[sheoraninfosec@Jigeshs-MacBook-Air ~ % sudo installer -pkg AWSCLIV2.pkg -target /
Password:
installer: Package name is AWS Command Line Interface
installer: Installing at base path /
installer: The install was successful.
[sheoraninfosec@Jigeshs-MacBook-Air ~ % aws --version
aws-cli/2.32.6 Python/3.13.9 Darwin/25.0.0 exe/arm64
sheoraninfosec@Jigeshs-MacBook-Air ~ %
```

Step 8: Configure AWS CLI with your IAM credentials.

```
sheoraninfosec — zsh — 95x32

[sheoraninfosec@Jigeshs-MacBook-Air ~ % aws configure
AWS Access Key ID [*****2W5Y]: AKIAWL60D6FGCC32W5Y
AWS Secret Access Key [*****K2M1]: tz9dC0eDSZZSC47HQpIIMJPmQtip1/4zOR5EK2M1

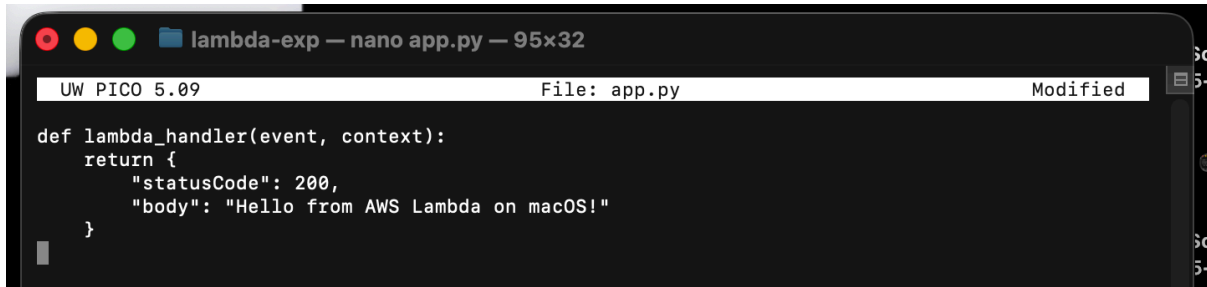
Default output format [None]: json
sheoraninfosec@Jigeshs-MacBook-Air ~ %
```

Step 9: Set the default Region to : ap-south-1 (Mumbai) & output format : json.

Step 10: Create Lambda Function Folder

```
[Default output format [None]: json
sheoraninfosec@Jigeshs-MacBook-Air ~ % mkdir lambda-exp
sheoraninfosec@Jigeshs-MacBook-Air ~ % cd lambda-exp
sheoraninfosec@Jigeshs-MacBook-Air lambda-exp % pwd
/Users/sheoraninfosec/lambda-exp
sheoraninfosec@Jigeshs-MacBook-Air lambda-exp % nano app.py
sheoraninfosec@Jigeshs-MacBook-Air lambda-exp %
```

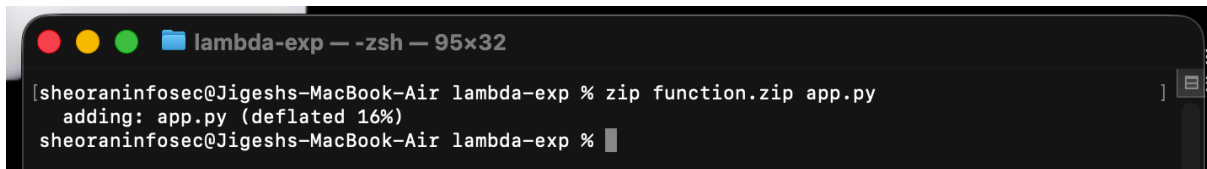
Step 11: Create lambda_handler file.



The screenshot shows a terminal window titled "lambda-exp — nano app.py — 95x32". The editor displays the following Python code:

```
def lambda_handler(event, context):
    return {
        "statusCode": 200,
        "body": "Hello from AWS Lambda on macOS!"
    }
```

Step 12: Zip the Lambda Code



The screenshot shows a terminal window titled "lambda-exp — -zsh — 95x32". The following command and output are visible:

```
sheoraninfosec@Jigeshs-MacBook-Air lambda-exp % zip function.zip app.py
adding: app.py (deflated 16%)
sheoraninfosec@Jigeshs-MacBook-Air lambda-exp %
```

Step 13: Write trust policy, recommended but not necessary.



The screenshot shows a terminal window titled "lambda-exp — nano trust.json — 95x32". The editor displays the following JSON code:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Step 14: Create IAM Execution Role (for Lambda)

```
sheoraniinfosec@Jigeshs-MacBook-Air lambda-exp % nano trust.json
sheoraniinfosec@Jigeshs-MacBook-Air lambda-exp % aws iam create-role \
  --role-name lambda-execution-role-exp \
  --assume-role-policy-document file://trust.json
{
  "Role": {
    "Path": "/",
    "RoleName": "lambda-execution-role-exp",
    "RoleId": "AROAWLD6OD6FOBBHWN6UV",
    "Arn": "arn:aws:iam::436204347274:role/lambda-execution-role-exp",
    "CreateDate": "2025-11-27T06:43:55+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "lambda.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}
```

Step 15: Also attach the basic execution policy.

```
sheoraniinfosec@Jigeshs-MacBook-Air lambda-exp % aws iam attach-role-policy \
  --role-name lambda-execution-role-exp \
  --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
sheoraniinfosec@Jigeshs-MacBook-Air lambda-exp %
```

Step 16: Create Lambda Function (from CLI)

```
sheoraniinfosec@Jigeshs-MacBook-Air lambda-exp % aws iam get-role --role-name lambda-execution-role-exp
{
  "Role": {
    "Path": "/",
    "RoleName": "lambda-execution-role-exp",
    "RoleId": "AROAWLD6OD6FOBBHWN6UV",
    "Arn": "arn:aws:iam::436204347274:role/lambda-execution-role-exp",
    "CreateDate": "2025-11-27T06:43:55+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "lambda.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    },
    "MaxSessionDuration": 3600,
    "RoleLastUsed": {}
  }
}
```

Step 17: Copy the Arn from the output of this created lambda function and paste it to “ aws lambda create-function “.

```
sheoraninfosec@Jigeshs-MacBook-Air lambda-exp % aws lambda create-function \
--function-name lambda-exp \
--runtime python3.12 \
--role arn:aws:iam::436204347274:role/lambda-execution-role-exp \
--handler app.lambda_handler \
--zip-file fileb://function.zip

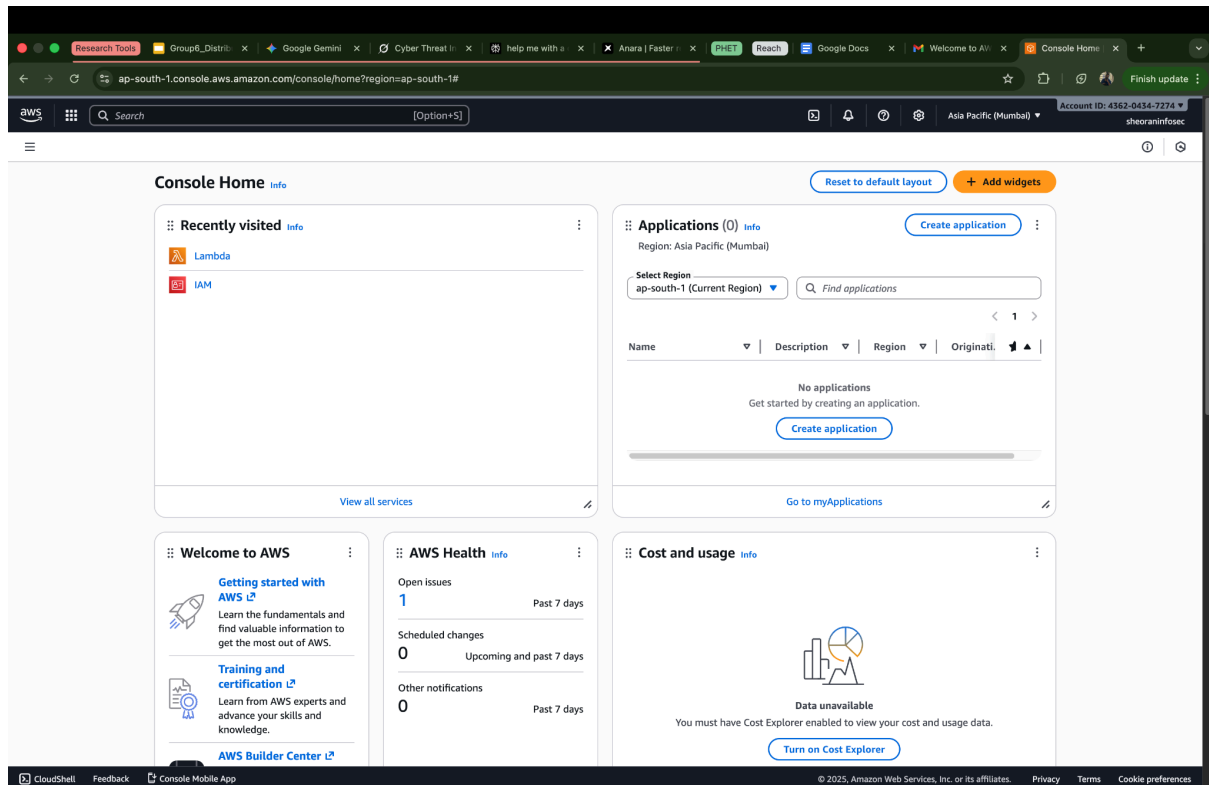
{
  "FunctionName": "lambda-exp",
  "FunctionArn": "arn:aws:lambda:ap-south-1:436204347274:function:lambda-exp",
  "Runtime": "python3.12",
  "Role": "arn:aws:iam::436204347274:role/lambda-execution-role-exp",
  "Handler": "app.lambda_handler",
  "CodeSize": 274,
  "Description": "",
  "Timeout": 3,
  "MemorySize": 128,
  "LastModified": "2025-11-27T06:49:32.872+0000",
  "CodeSha256": "BRA7ur/Q3VKi8HpPd1IF/PoiY+CFZHeWeMAqUj3uJ5E=",
  "Version": "$LATEST",
  "TracingConfig": {
    "Mode": "PassThrough"
  },
  "RevisionId": "49354e78-e114-4f8d-816b-b3795d14509b",
  "State": "Pending",
  "StateReason": "The function is being created.",
  "StateReasonCode": "Creating",
  "PackageType": "Zip",
  "Architectures": [
    "x86_64"
  ],
}
```

Step 18: Check if the lambda function is deployed correctly or not.

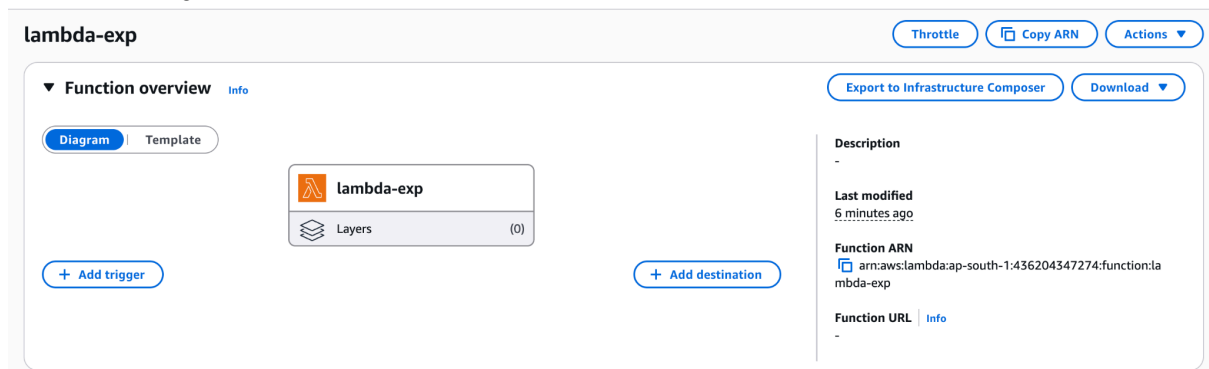
```
sheoraninfosec@Jigeshs-MacBook-Air lambda-exp % aws lambda list-functions

{
  "Functions": [
    {
      "FunctionName": "lambda-exp",
      "FunctionArn": "arn:aws:lambda:ap-south-1:436204347274:function:lambda-exp",
      "Runtime": "python3.12",
      "Role": "arn:aws:iam::436204347274:role/lambda-execution-role-exp",
      "Handler": "app.lambda_handler",
      "CodeSize": 274,
      "Description": "",
      "Timeout": 3,
      "MemorySize": 128,
      "LastModified": "2025-11-27T06:49:32.872+0000",
      "CodeSha256": "BRA7ur/Q3VKi8HpPd1IF/PoiY+CFZHeWeMAqUj3uJ5E=",
      "Version": "$LATEST",
      "TracingConfig": {
        "Mode": "PassThrough"
      },
      "RevisionId": "cb828a45-cf10-4d72-a219-39d7cf156052",
      "PackageType": "Zip",
      "Architectures": [
        "x86_64"
      ],
      "EphemeralStorage": {
        "Size": 512
      },
      "SnapStart": {
        "ApplyOn": "None",
        "OptimizationStatus": "Off"
      },
    }
  ],
}
```

Step 19: Create API Gateway Trigger (via AWS Console)



AWS Console → Lambda → Open **lambda-exp** → Click Add Trigger → Select API Gateway



Step 20: Choose -
Create a new API
API Type: HTTP API
Security: Open
Click Add

Add trigger

Trigger configuration [Info](#)

API Gateway
aws api application-services backend HTTP REST serverless

Add an API to your Lambda function to create an HTTP endpoint that invokes your function. API Gateway supports REST, HTTP, and WebSocket APIs. [Learn more](#)

Intent
Use an existing api or have us create one for you.

☒ Create a new API
☐ Use existing API

API type

☒ **HTTP API**
Build low-latency and cost-effective REST APIs with built-in features such as OIDC and OAuth2, and native CORS support.

☐ **WebSocket API**
Build a WebSocket API using persistent connections for real-time use cases such as chat applications or dashboards.

☐ **REST API**
Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Security
Configure the security mechanism for your API endpoint.

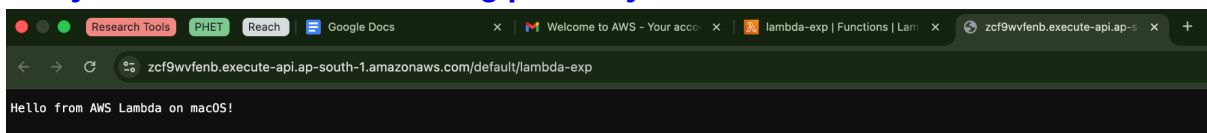
Open

► **Additional settings**

Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

[Cancel](#) [Add](#)

Step 21: You'll get a public HTTPS URL. Paste it into a new tab and you will see that you AWS Lambda is running perfectly.



Step 22: Not necessary / but recommended.

API Gateway cleanup:

AWS Console → API Gateway → Your API → Delete

If you don't need CLI access after performing this experiment it is advised to delete API Keys to avoid unnecessary free access.

-----END OF DOCUMENT -----