

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

Aim :- Orchestrating serverless functions with step functions.

Lab overview and objectives

In this lab, you will use AWS Step Functions to coordinate the actions necessary to generate and deliver a report upon request. The report will contain data from a database.

After completing this lab, you should be able to:

- Create an asynchronous state machine by using Step Functions
- Configure an Amazon Simple Notification Service (Amazon SNS) topic to deliver email alerts
- Configure AWS Lambda functions to be invoked from a Step Functions state machine
- Use a parallel state flow object in the design of a Step Functions state machine
- Invoke a state machine to start when a REST API endpoint is invoked
- Generate a presigned URL for an object stored in an Amazon Simple Storage Service (Amazon S3) bucket

AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

Scenario

Thanks to Sofia, the coffee suppliers inventory now automatically updates on the café website. But the work is not finished! Frank asks if it would be possible to log in to the website to request a report with the latest inventory information.

Yesterday, Mateo, who is an AWS consultant and Sofia's friend, came into the café for a macchiato, his favorite espresso drink. While he was enjoying his beverage, Sofia told him about how she needs to build a reporting mechanism for Frank. After discussing the high-level business requirements, Mateo suggested that she use AWS Step Functions to coordinate the steps to generate the report. He described how Step Functions can help a developer to automate business processes by creating workflows, and providing parallelization and service integrations, among other features.

X`In this lab, Sofia will build the functionality to create and deliver the report. Then, in the next lab, she will improve the design further and implement authentication on the website to limit who can request and access reports.

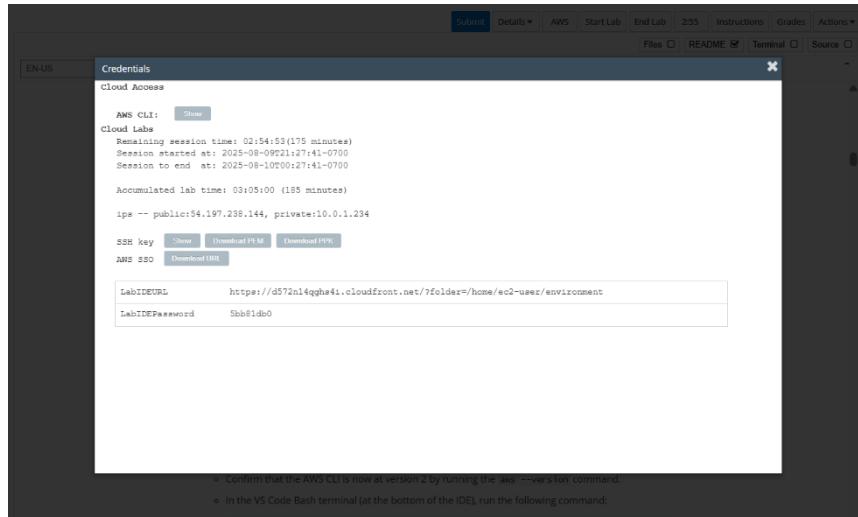
Task 1: Preparing the lab

Connect to the VS Code IDE.

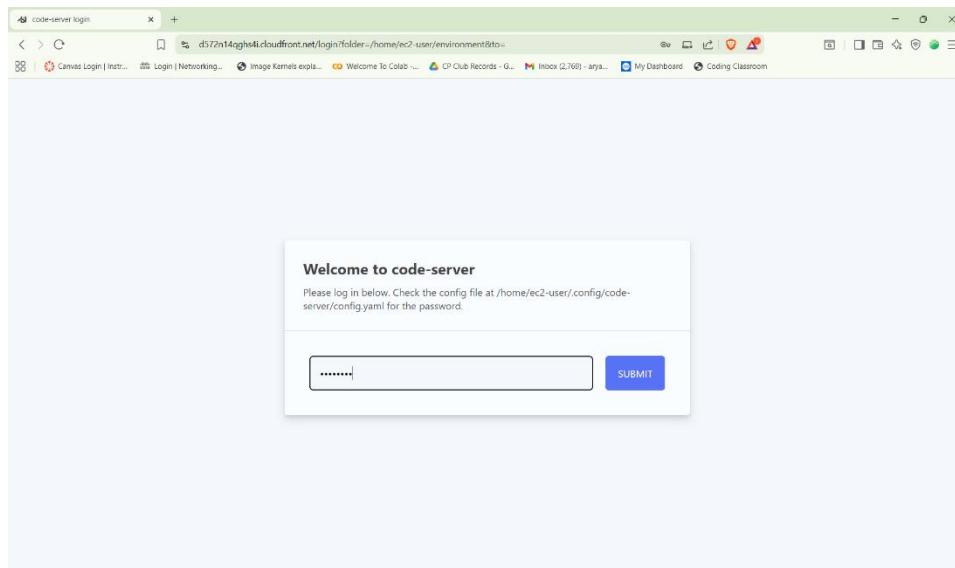
1. At the top of these instructions, choose **Details** followed by **AWS: Show**
2. Copy values from the table **similar** to the following and paste it into an editor of your choice for use later.
 - **LabIDEURL**

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.
Experiment No: 12	Date:

- **LabIDEPASSWORD**



3. In a new browser tab, paste the value for **LabIDEURL** to open the VS Code IDE.
4. On the prompt window **Welcome to code-server**, enter the value for **LabIDEPASSWORD** you copied to the editor earlier, choose **Submit** to open the VS Code IDE.



5. Download and extract the files that you need for this lab.
 - In the VS Code bash terminal (located at the bottom of the IDE), run the following commands:

```
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/11-lab-step/code.zip -P /home/ec2-user/environment
```



Subject: Cloud Developing (01CT0720)

Aim: Orchestrating serverless functions with step functions.

Experiment No: 12

Date:

Enrolment No: 92200133030

The screenshot shows the VS Code interface. In the left sidebar, under 'ENVIRONMENT', there is a folder named 'code.zip'. In the bottom right corner of the main editor area, a progress bar indicates the download of 'code.zip' from an AWS Lambda environment. The terminal tab shows the command used to download the file.

```
[ec2-user@ip-10-0-1-125 environment]$ wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACODEV-2-91558/05-lab-lambda/code.zip -P /home/ec2-user/environment  
--2025-08-10 11:15:29 -- https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACODEV-2-91558/05-lab-lambda/code.zip  
Resolving aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)... 52.92.206.170, 52.92.153.26, 52.92.236.34,  
...  
Connecting to aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)|52.92.206.170|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 5827466 (5.0M) [application/zip]  
Saving to: '/home/ec2-user/environment/code.zip'  
  
code.zip 100%[=====] 5.56M 9.97MB/s in 0.6s  
2025-08-10 11:15:30 (9.97 MB/s) - '/home/ec2-user/environment/code.zip' saved [5827466/5827466]  
[ec2-user@ip-10-0-1-125 environment]$
```

6. You should see that the **code.zip** file was downloaded to the VS Code IDE and is now in the left navigation pane.

- Extract the file by running the following command:
`unzip code.zip`

The screenshot shows the VS Code interface after extracting the 'code.zip' file. The left sidebar now shows the extracted contents: 'python_3', 'resources', and a newly created folder 'code'. The terminal tab shows the command used to extract the zip file, and the output shows the individual files being extracted from the archive.

```
[ec2-user@ip-10-0-1-125 environment]$ unzip code.zip  
Archive: code.zip  
extracting: python_3/create_report_code.py  
extracting: python_3/get_all_products_code.py  
extracting: python_3/get_all_products_wrapper.py  
extracting: python_3/create_report_wrapper.py  
extracting: python_3/update_config.py  
extracting: resources/seed.py  
extracting: resources/setup.sh  
extracting: resources/public_policy.json  
extracting: resources/permissions.py  
extracting: resources/website/all_products.html  
extracting: resources/website/all_products.json  
extracting: resources/website/all_products_on_offer.json  
[ec2-user@ip-10-0-1-125 environment]$
```

7. Run a script that upgrades the version of the AWS CLI installed on the VS Code IDE.

- To set permissions on the script and then run it, run the following commands in the Bash



Subject: Cloud Developing (01CT0720)

Experiment No: 12

Date:

Enrolment No: 92200133030

terminal:

```
chmod +x ./resources/setup.sh && ./resources/setup.sh
```

The script will prompt you for the **IP address** by which your computer is known to the internet. Use www.whatismyip.com to discover this address and then paste the IPv4 address into the command prompt and finish running the script.

```
● [ec2-user@ip-10-0-1-150 environment]$ chmod +x resources/setup.sh && resources/setup.sh
Please enter a valid IP address:
152.58.63.192
IP address:152.58.63.192
upload: resources/website/all_products_on_offer.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/all_products_on_offer.json
upload: resources/website/callback.html to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/callback.html
upload: resources/website/all_products.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/all_products.json
upload: resources/website/beans.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/beans.json
upload: resources/website/images/beans/excelsa.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/beans/excelsa.png
upload: resources/website/config.js to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/config.js
upload: resources/website/images/items/blueberry_bagel.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/blueberry_bagel.png
upload: resources/website/images/items/apple_pie.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/apple_pie.jpeg
upload: resources/website/images/items/blueberry_jelly_doughnut.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/blueberry_jelly_doughnut.jpeg
upload: resources/website/images/beans/robusta.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/beans/robusta.png
upload: resources/website/images/items/boston_cream_doughnut.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/boston_cream_doughnut.jpeg
upload: resources/website/images/expanded.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/expanded.png
upload: resources/website/images/beans/liberica.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/beans/liberica.png
upload: resources/website/images/items/apple_pie_slice.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/apple_pie_slice.png
upload: resources/website/images/items/apple_pie.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/apple_pie.png
upload: resources/website/images/beans/arabica.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/beans/arabica.png
upload: resources/website/images/items/boston_cream_doughnut.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/boston_cream_doughnut.png
upload: resources/website/favicon.ico to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/favicon.ico
upload: resources/website/images/items/apple_pie_slice.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/apple_pie_slice.jpeg
upload: resources/website/images/items/cherry_pie.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/cherry_pie.png
upload: resources/website/images/items/blueberry_bagel.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/blueberry_bagel.jpeg
upload: resources/website/images/items/blueberry_jelly_doughnut.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/blueberry_jelly_doughnut.png
upload: resources/website/images/items/cherry_pie_slice.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/cherry_pie_slice.png
upload: resources/website/images/items/cherry_pie.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/cherry_pie.jpeg
upload: resources/website/images/items/cherry_pie_slice.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/cherry_pie_slice.jpeg
upload: resources/website/images/items/chocolate_chip_cupcake.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvveyyvh51/images/items/chocolate_chip_cupcake.jpeg
```

Layout: US

8. Verify the AWS CLI version and also verify that the SDK for Python is installed.

- Confirm that the AWS CLI is now at version 2 by running the **aws --version** command.
- In the VS Code Bash terminal (at the bottom of the IDE), run the following command:
pip3 show boto3

```
● [ec2-user@ip-10-0-1-234 environment]$ aws --version
aws-cli/2.28.6 Python/3.13.4 Linux/6.1.147-172.266.amzn2023.x86_64 exe/x86_64.amzn.2023
● [ec2-user@ip-10-0-1-234 environment]$ pip3 show boto3
Name: boto3
Version: 1.40.6
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
Author: Amazon Web Services
Author-email:
License: Apache License 2.0
Location: /usr/local/lib/python3.11/site-packages
Requires: botocore, jmespath, s3transfer
Required-by:
○ [ec2-user@ip-10-0-1-234 environment]$
```

Subject: Cloud Developing (01CT0720)

Aim: Orchestrating serverless functions with step functions.

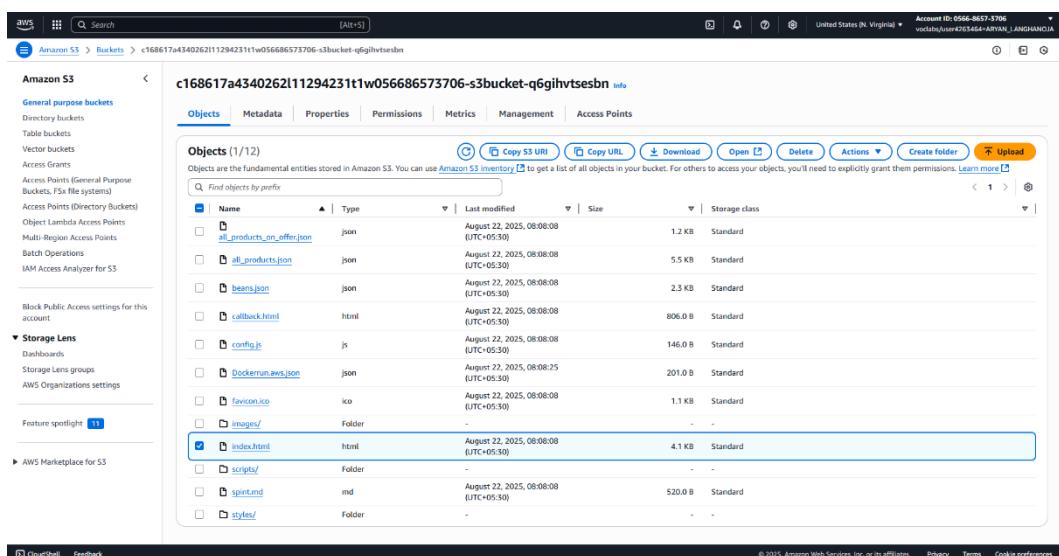
Experiment No: 12

Date:

Enrolment No: 92200133030

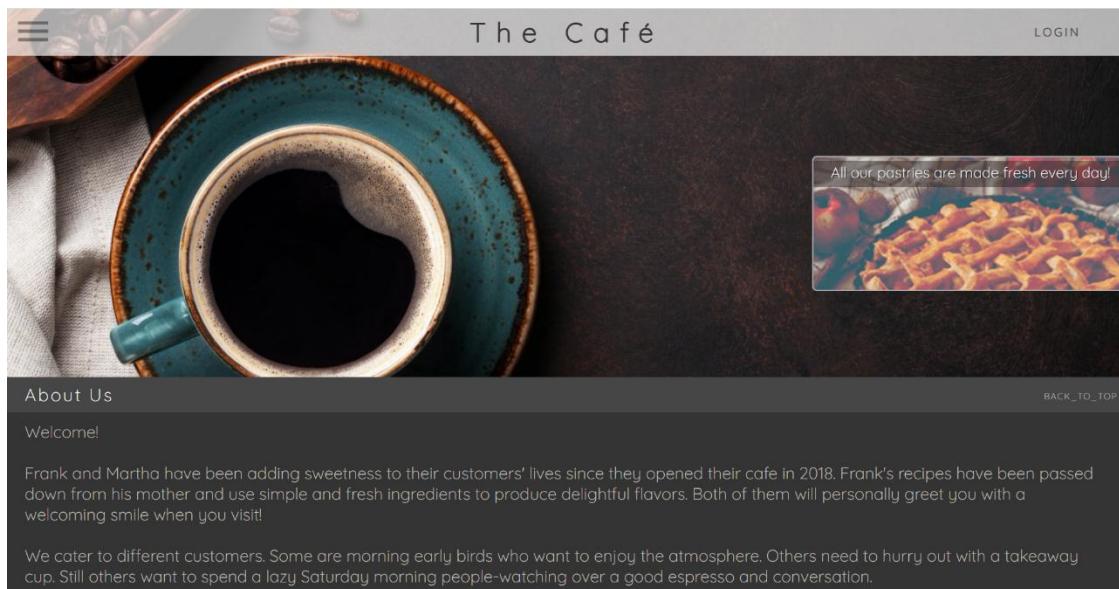
9. Verify that you can access the café website.

- Navigate to the Amazon S3 console.
- Choose the link for the bucket that has *-s3bucket* in the name.
- Open the **index.html** file.
- In the **Object overview** section, open the **Object URL** in a new browser tab.



The screenshot shows the Amazon S3 console with the following details:

- Bucket Name:** c168617a4340262111294231t1w056686573706-s3bucket-q6gihvtsesbn
- Region:** United States (N. Virginia)
- Account ID:** 0566-8657-3706
- Objects (1/12):**
 - Name:** index.html
 - Type:** HTML
 - Last modified:** August 22, 2025, 08:08:08 (UTC+05:30)
 - Size:** 4.1 KB
 - Storage class:** Standard
- Actions:** Copy S3 URI, Copy URL, Download, Open, Delete, Actions, Create folder, Upload



The screenshot shows the homepage of The Café website:

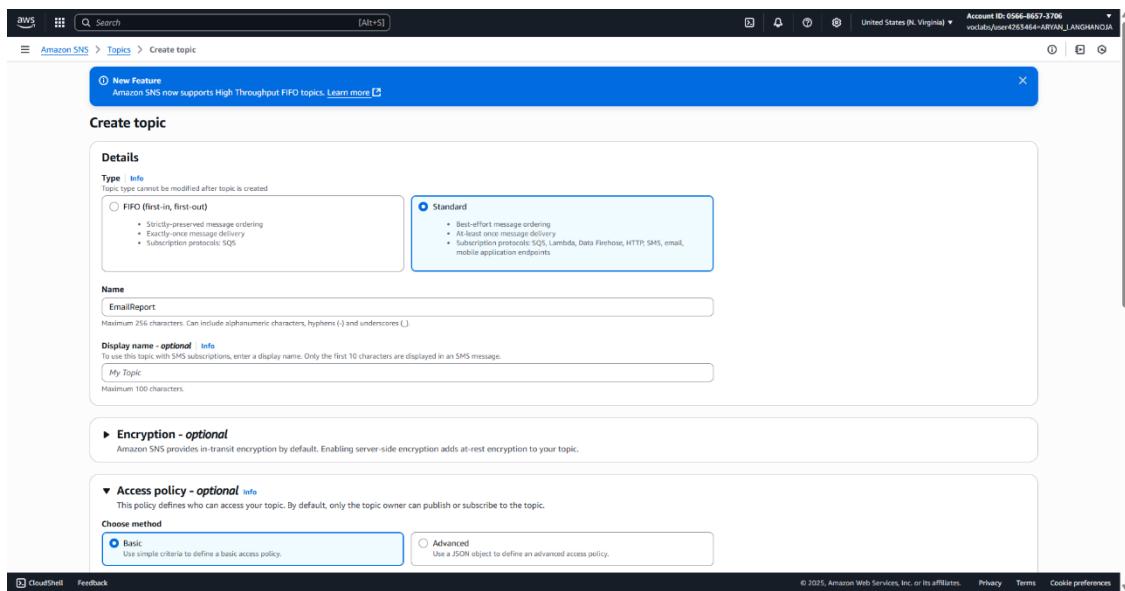
- Header:** The Café
- Image:** A large image of a cup of coffee.
- Text:** All our pastries are made fresh every day!
- Navigation:** About Us, Welcome!, Frank and Martha have been adding sweetness to their customers' lives since they opened their cafe in 2018. Frank's recipes have been passed down from his mother and use simple and fresh ingredients to produce delightful flavors. Both of them will personally greet you with a welcoming smile when you visit!
- Text:** We cater to different customers. Some are morning early birds who want to enjoy the atmosphere. Others need to hurry out with a takeaway cup. Still others want to spend a lazy Saturday morning people-watching over a good espresso and conversation.
- Buttons:** BACK_TO_TOP

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

Task 2: Configuring an SNS topic and subscribing to it

10. Create an SNS topic for email notification.

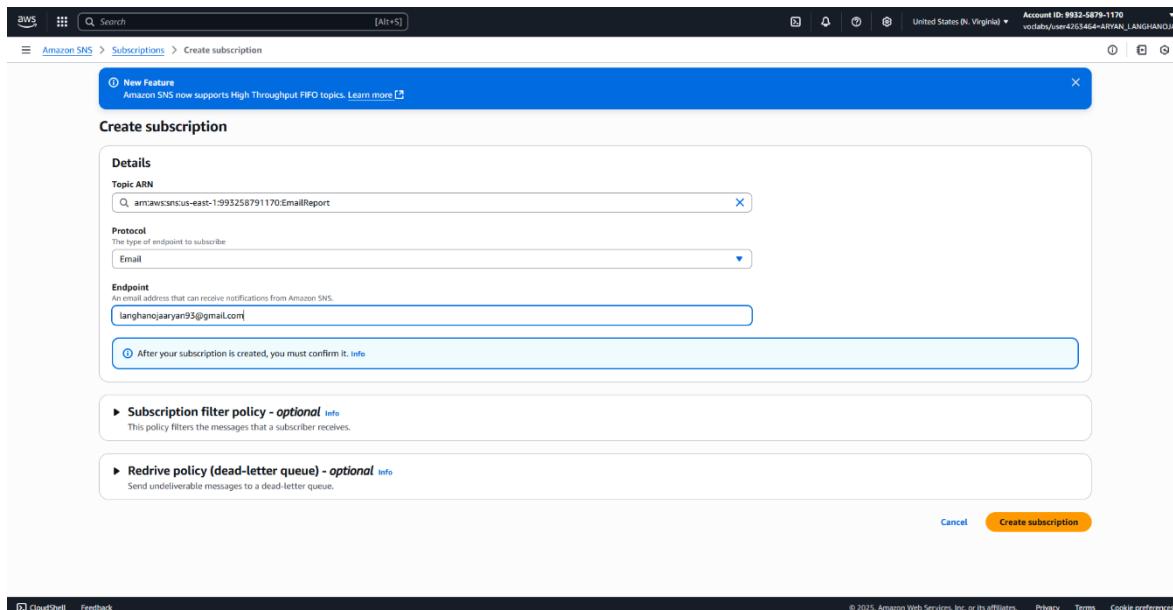
- Navigate to the Amazon SNS console.
- In the left navigation pane, choose **Topics**.
- Choose **Create topic** and configure the following:
 - i. **Type:** Choose **Standard**.
 - ii. **Name:** Enter `EmailReport`
 - iii. Expand the **Access policy - optional** section.
 - iv. **Define who can publish messages to the topic:** Choose **Everyone**.
 - v. **Define who can subscribe to this topic:** Choose **Everyone**.
- At the bottom of the page, choose **Create topic**.



11. Create an email subscription to the SNS topic.

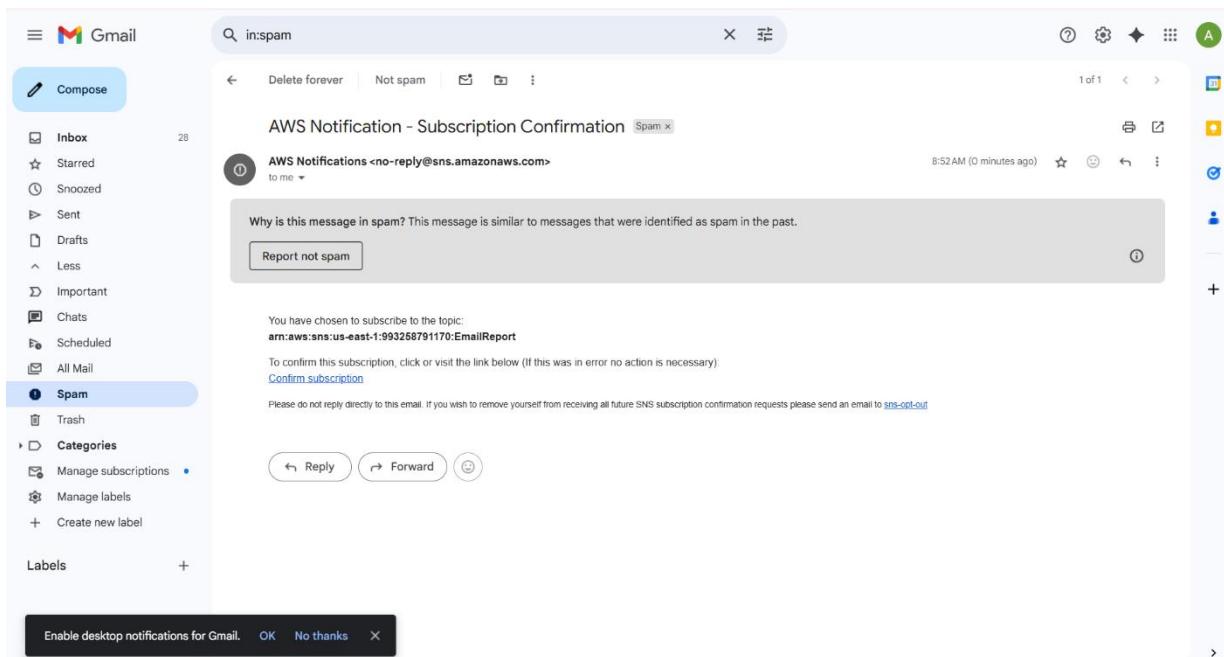
- Choose **Create subscription** and configure the following:
 - i. **Topic ARN:** Notice that the Amazon Resource Number (ARN) of the topic that you just created is already filled in.
 - ii. **Protocol:** Choose **Email**.
 - iii. **Endpoint:** Enter an email address where you can receive emails during this lab. (In the café story, Sofia would enter Frank's email address.)
- Choose **Create subscription**.

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.
Experiment No: 12	Date:



12. Check your email and confirm the subscription.

- Check your email for a message from AWS Notifications.
- In the email body, choose the Confirm subscription link.
- A webpage opens and displays a message that the subscription was successfully confirmed.



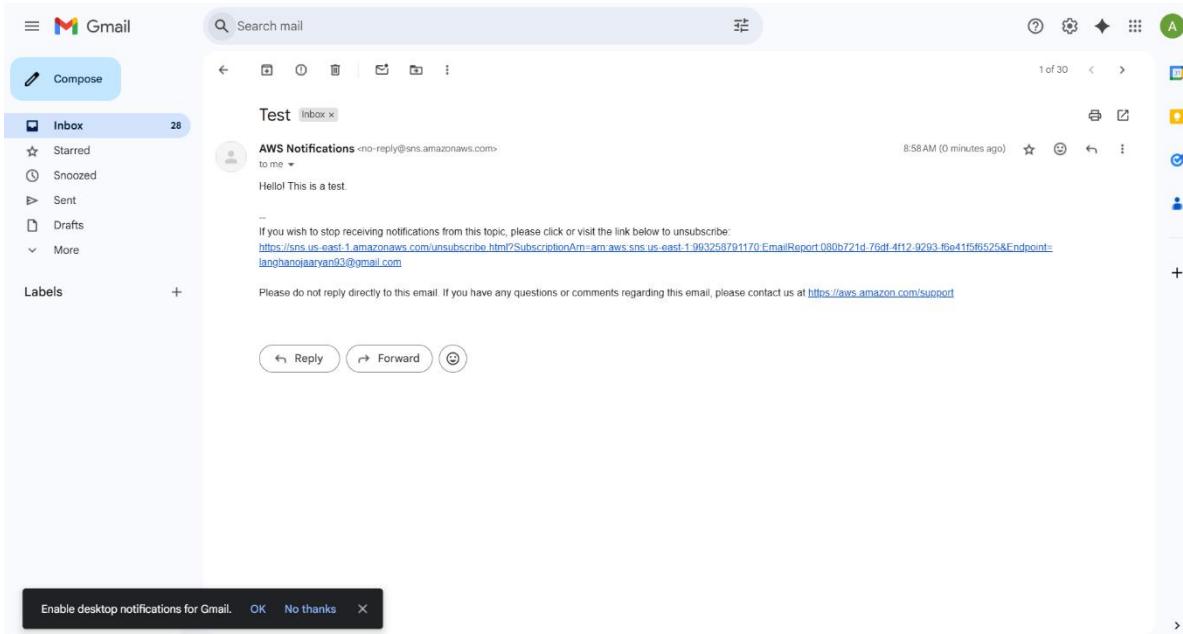
 Marwadi University Marwadi Chandarana Group	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030



13. Publish a test message to confirm that messages can be published to the *EmailReport* SNS topic.

- Return to the Amazon SNS console, and return to the *EmailReport* topic page.
 - Tip:** In the breadcrumbs at the top of the page, choose **EmailReport**.
- Choose **Publish message** and configure the following:
 - Subject - optional:** Enter **Test**
 - Message body:** Enter **Hello! This is a test.**
- At the bottom of the page, choose **Publish message**.

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030



The screenshot shows a Gmail inbox with an incoming email from 'AWS Notifications <no-reply@sns.amazonaws.com>' to the user. The subject of the email is 'Test'. The body of the email contains the text 'Hello! This is a test.' and a note about unsubscribing with a provided link. Below the inbox, a modal dialog box is displayed, asking the user if they want to enable desktop notifications for Gmail, with 'OK' and 'No thanks' buttons.

Task 3: Creating a Step Functions state machine

14. Analyze the details of the IAM role that Step Functions will use.

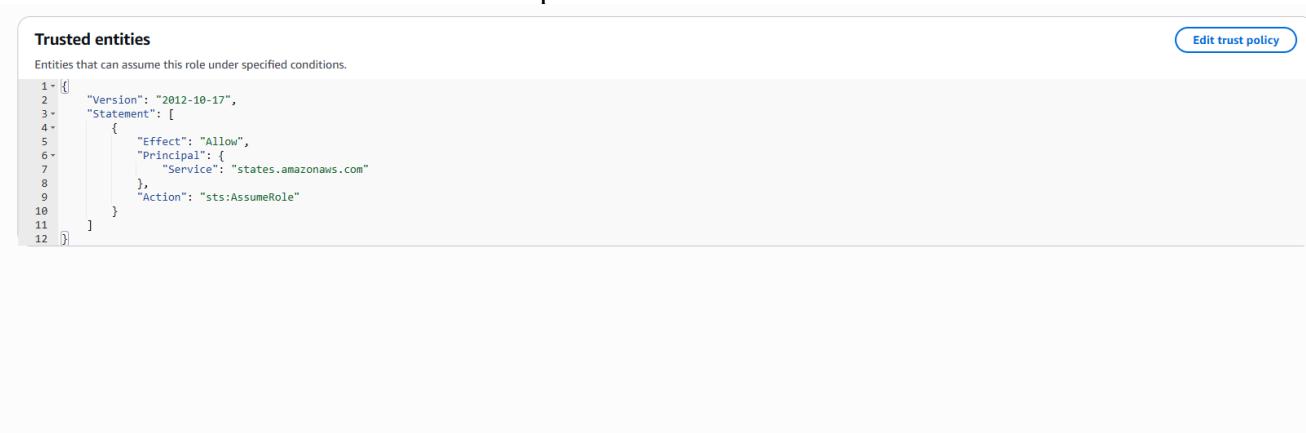
- Navigate to the IAM console.
- Review the details of the *RoleForStepToCreateAReport* IAM role.
 - In the left navigation pane, choose Roles.
 - In the search box under Roles, search for and choose the *RoleForStepToCreateAReport* role.
 - On the Permissions tab, expand the *stepPolicyForCreateReport* policy, and choose {} JSON.

Notice that this role allows all Lambda and log actions on all resources.

- Expand the *AWSLambdaRole* managed policy, which is also attached to the role.

Notice that this role allows the *lambda:InvokeFunction* action on all resources. This will allow you to test the function from the Lambda console.

- Choose the Trust relationships tab.



The screenshot shows the 'Trust relationships' tab for the *RoleForStepToCreateAReport* IAM role. It displays a JSON policy document with line numbers 1 through 12. The policy grants the *sts:AssumeRole* action to the *states.amazonaws.com* service.

```

1 - [
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Principal": {
7         "Service": "states.amazonaws.com"
8       },
9       "Action": "sts:AssumeRole"
10    }
11  ]
12 ]
  
```



Subject: Cloud Developing (01CT0720)

Aim: Orchestrating serverless functions with step functions.

Experiment No: 12

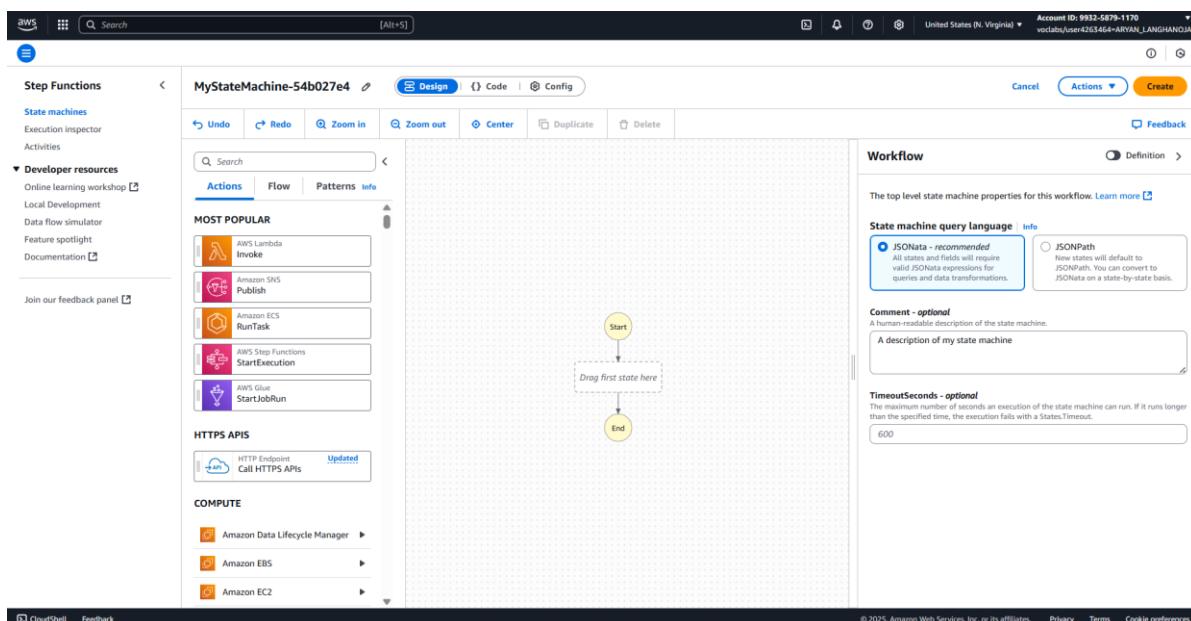
Date:

Enrolment No: 92200133030

15. Begin to create a Step Functions state machine.

- Navigate to the Step Functions console.
- In the left navigation pane, choose State machines.
- Choose Create state machine and configure the following for step 1:
 - Choose a template: Choose Blank.
 - Choose Select.

The Workflow Studio appears for step 2.



16. Design the workflow.

- In the search box under Step 2: Design workflow, enter SNS
- Drag the Amazon SNS Publish object onto the canvas, to the box labeled *Drag first state here*.
- In the SNS Publish pane that displays, configure the following:
 - Topic: Choose the ARN of the EmailReport SNS topic that you created earlier.
 - Notice that Message is set to Use state input as message, as shown in the following image.
- Choose {} Code in the upper part of the screen.
- The generated Amazon States Language (ASL) code for the state machine displays, as shown in the following image.
- Choose {} Code in the upper part of the screen.
- The generated Amazon States Language (ASL) code for the state machine displays, as shown in the following image.

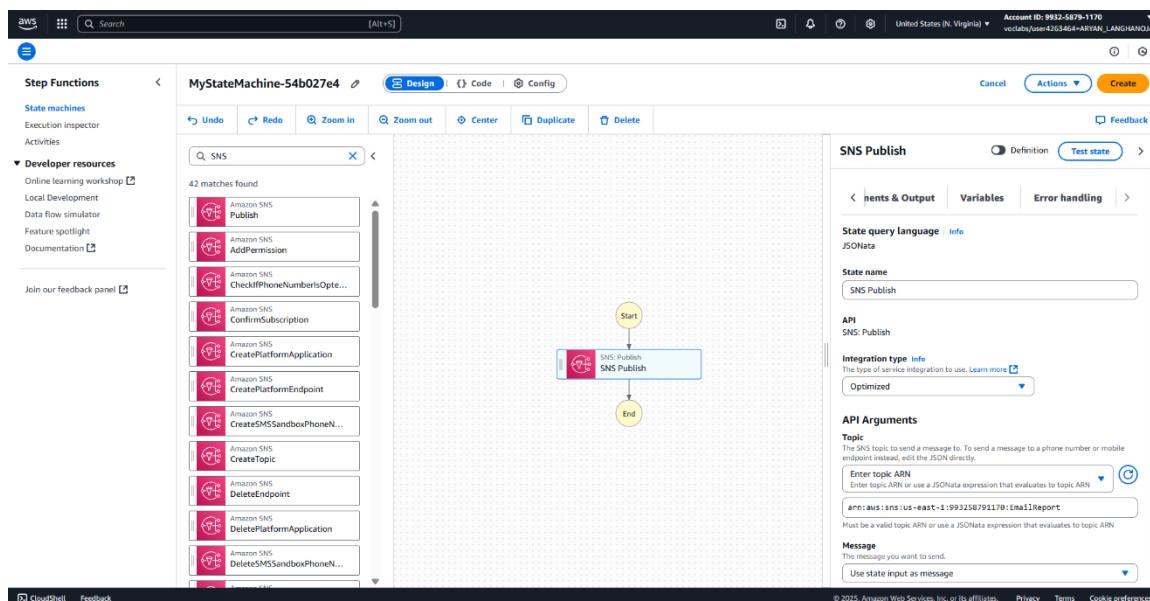
Subject: Cloud Developing (01CT0720)

Aim: Orchestrating serverless functions with step functions.

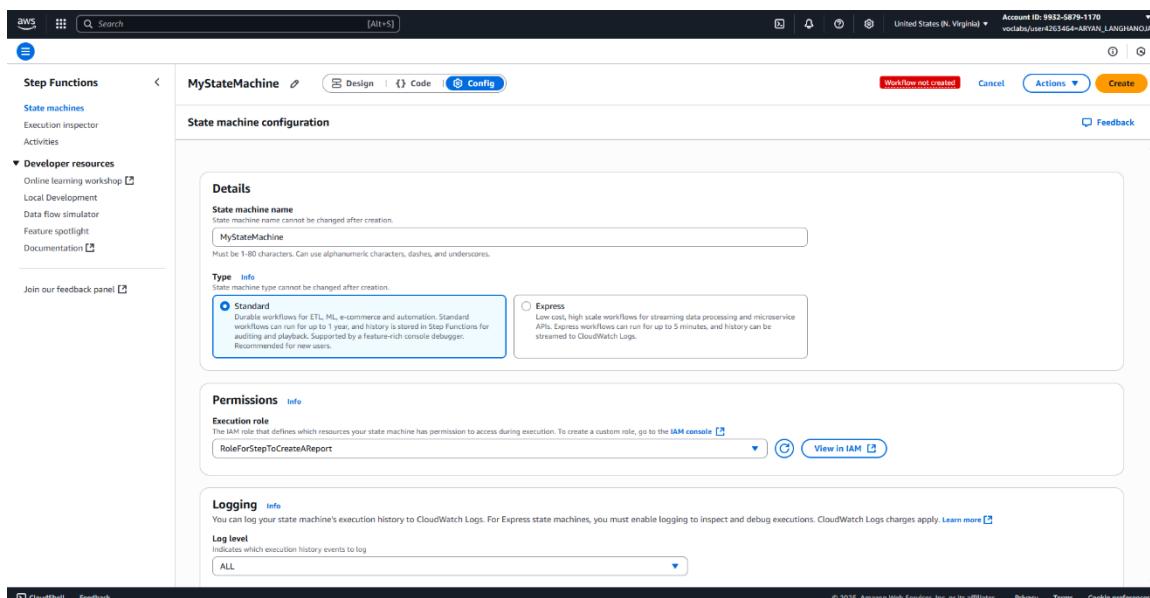
Experiment No: 12

Date:

Enrolment No: 92200133030



- Choose {} Code in the upper part of the screen.
- The generated Amazon States Language (ASL) code for the state machine displays, as shown in the following image.
- Choose Config and configure the following:
 - State machine name:** Enter MyStateMachine
 - Execution role:** Choose Choose an existing role.
 - Existing roles:** Choose RoleForStepToCreateAReport.
 - Log level:** Choose ALL.
- At the top of the page, choose Create.



 Marwadi University Marwadi Chandarana Group	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

17. Test the state machine.

- Choose **Execute** button at the top and configure the following:
- In the code editor, replace the existing JSON code with the following.


```
{
  "presigned_url_str": "Testing that my email message works"
}
```
- Choose **Start execution**.
- A message displays and indicates that the execution started successfully.
- To see the details, choose **Execution input and output**.
- Details about the execution are displayed.

Start execution

Name
c618668d-85b4-49d7-bf55-968db8600c35
Must be 1-80 characters. Can use alphanumeric characters, dashes, or underscores.

Input - optional
Enter input values for this execution in JSON format

[Format JSON](#) [Export](#) [Import](#)

```
1 {
2   "presigned_url_str": "Testing that my email message works"
3 }
```

[Start execution with latest revision](#)

Open in a new browser tab

[Cancel](#) [Start execution](#)

Step Functions > **Execution: c618668d-85b4-49d7-bf55-968db8600c35**

Execution started successfully

Execution: c618668d-85b4-49d7-bf55-968db8600c35

[Edit state machine](#) [New execution](#) [Actions](#)

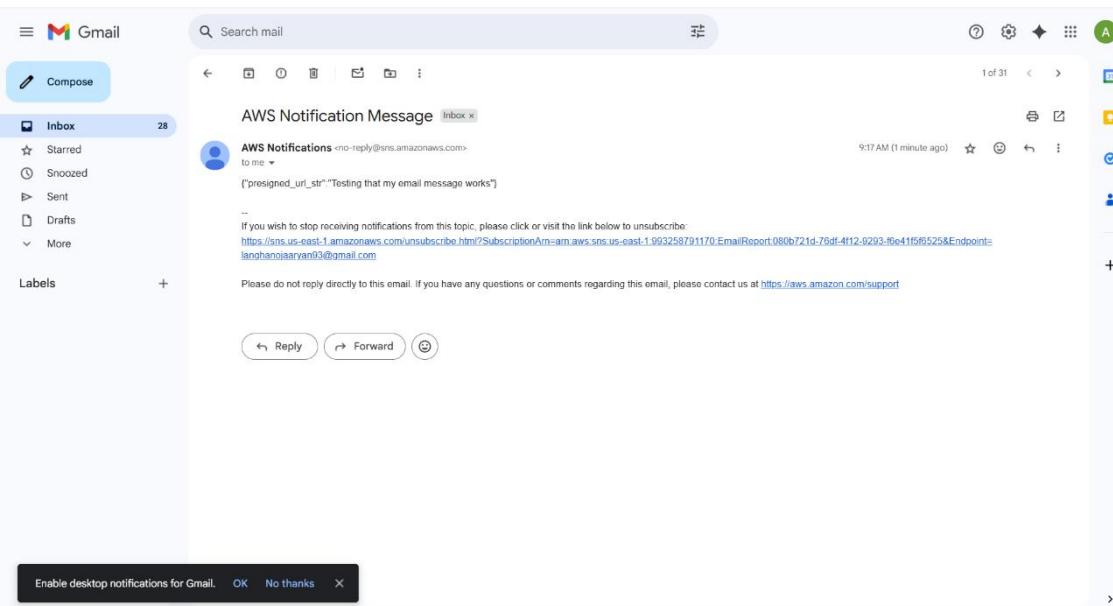
Details	Execution input and output	Definition
Execution status: Succeeded Execution type: Standard Execution ARN: arn:aws:states:us-east-1:993258791170:execution:MyStateMachine:c618668d-85b4-49d7-bf55-968db8600c35 IAM role ARN: arn:aws:iam::993258791170:role/RoleForStepToCreateARole State transitions: Learn more Execution Logs: Learn more CloudWatch Logs	Start time: Aug 22, 2025, 09:17:08.799 (UTC+05:30) End time: Aug 22, 2025, 09:17:09.016 (UTC+05:30) Duration: 00:00:00.217 Alias: - Version: -	Graph view Table view

Graph view



Step details
Choose a step to view its details.

 Marwadi University Marwadi Chandarana Group	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030



Task 4: Creating a Lambda function to generate a presigned URL

18. Create a sample *report.html* page.
- Return to the VS Code IDE.
 - In the Environment window, choose File > New Text File.
 - In the new file that opens, paste in the following code.
 - <output>Hello! This is some sample HTML.</output>
 - Choose File > Save.
 - Save the file under /home/ec2-user/environment directory.
 - Name the file report.html and choose Save.

```

environment
report.html
report.html
report.html

[ec2-user@ip-10-0-1-186 environment]$ pip3 show boto3
Name: boto3
Version: 1.40.15
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
Author: Amazon Web Services
Author-email:
License: Apache License 2.0
Location: /usr/local/lib/python3.11/site-packages
Requires: botocore, jmespath, s3transfer
Required-by:
[ec2-user@ip-10-0-1-186 environment]$

```



Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

19. Upload the sample report file to your S3 bucket, and observe the bucket policy settings.

- To upload the file to the S3 bucket and set the *cache-control max-age* value on the file to 0, run the following two commands in the VS Code IDE terminal:

```
bucket=$(aws s3api list-buckets --query "Buckets[].Name" | grep s3bucket | tr -d ',' | sed -e 's://"g' | xargs)

aws s3 cp report.html s3://$bucket/ --cache-control "max-age=0"
```

```
● [ec2-user@ip-10-0-1-186 environment]$ bucket=$(aws s3api list-buckets --query "Buckets[].Name" | grep s3bucket | tr -d ',' | sed -e 's://"g' | xargs)
aws s3 cp report.html s3://$bucket/ --cache-control "max-age=0"
upload: ./report.html to s3://c168617a434026211294231t1w993258791170-s3bucket-hhospq9qatsj/report.html
○ [ec2-user@ip-10-0-1-186 environment]$
```

- In a separate browser tab, navigate to the Amazon S3 console.
- Choose the link for the bucket that has *-s3bucket* in the name.
- Verify that the *report.html* file is now in the bucket.
- Choose the Permissions tab.
- Review the information in the Bucket policy section.

20. Test direct access to the sample report.

- In the Amazon S3 console, choose the **Objects** tab, and then choose **report.html**.
- Copy the **Object URL** and paste it into a new browser tab.
- You receive an *AccessDenied* error, as shown in the following image. This is expected because of the bucket policy details that you just observed.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied</Message>
  <RequestId>EKEPVK4N1YRCGBBB</RequestId>
  <HostId>ih/01fEy2lvmuH2r8kYVk+X4KGm0GBzuithr3d02NYIZt07PMz9gGn+KzyPHVRJ6nN6JwQ0DF0c=</HostId>
</Error>
```



Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

21. Create a presigned URL to access the report and test it.

- To generate a presigned URL that is valid for 30 seconds, run the following command:
`aws s3 presign s3://$bucket/report.html --expires-in 30`

Hello! This is some sample HTML.

22. First, analyze the details of the IAM role that the Lambda functions in this lab will use.

- Navigate to the IAM console.
 - Review the details of the *RoleForAllLambdas* IAM role.
 - In the left navigation pane, choose **Roles**.
 - In the search box under **Roles**, search for and choose the *RoleForAllLambdas* role.
 - On the **Permissions** tab, expand the **lambdaPolicyForAllLambdaSteps** policy, and choose **{ JSON**.
 - Notice that this role allows Amazon S3 and Amazon EC2 actions on all resources.
 - Choose the **Trust relationships** tab.
 - Notice that this role allows the Lambda service to assume this role (as indicated by the *lambda.amazonaws.com* service endpoint).

```
lambdaPolicyForAllLambdaSteps
1- [ {
2-     "Version": "2012-10-17",
3-     "Statement": [
4-         {
5-             "Action": [
6-                 "ec2:*",
7-                 "s3:*"
8-             ],
9-             "Resource": "*",
10            "Effect": "Allow"
11        }
12    ]
13 }]
```

[Copy JSON](#) [Edit](#)

Trusted entities

Entities that assume this role under specified conditions.

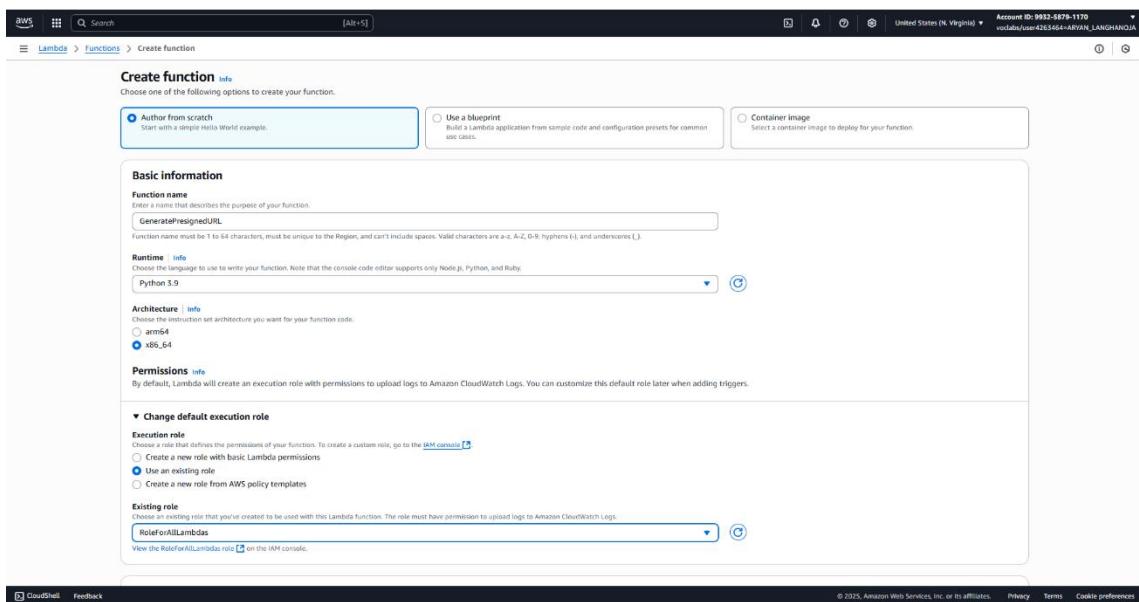
[Edit trust policy](#)

```
1 - [{}  
2 -   "Version": "2012-10-17",  
3 -     "Statement": [  
4 -       {  
5 -         "Effect": "Allow",  
6 -           "Principal": {  
7 -             "Service": "lambda.amazonaws.com"  
8 -           },  
9 -           "Action": "sts:AssumeRole"  
10 -       }  
11 -     ]  
12 - ]
```

Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

23. Create a Lambda function that will generate a presigned URL for the report.

- Navigate to the Lambda console.
- Verify that you are in the N. Virginia (us-east-1) Region.
- Choose **Create function** and configure the following:
 - Choose **Author from scratch**.
 - **Function name:** Enter `GeneratePresignedURL`
 - **Runtime:** Choose **Python 3.9**.
 - Expand the **Change default execution role** section.
 - **Execution role:** Choose **Use an existing role**.
 - **Existing role:** Choose **RoleForAllLambdas**.
- At the bottom of the page, choose **Create function**.



Code source [Info](#)

```

EXPLORER    lambda_function.py
GENERATEPRESIGNEDURL
lambda_function.py
1 import logging
2 import boto3
3 from botocore.exceptions import ClientError
4
5 s3_client = boto3.client('s3')
6
7 def lambda_handler(event, context):
8     bucket_name = "clue8617a41402621112942311w993258791170-s3bucket-vmeq1yrtmqbv"
9     object_name = "report.html"
10    expiration_in_seconds = 120
11
12
13    presigned_url_stm = s3_client.generate_presigned_url('get_object',Params={
14        'Bucket': bucket_name,'Key': object_name,'ExpiresIn':expiration_in_seconds}
15    )
16    response["presigned_url_stm"] = presigned_url_stm
17 except ClientError as e:
18     logging.error(e)
19     return None
20
21 # The response contains the presigned URL
22 return response
  
```

Open in Visual Studio Code [Upload from](#)

Deploy (Ctrl+Shift+I) Test (Ctrl+Shift+T)

ENVIRONMENT VARIABLES

L15, Col 57 Spaces: 4 UTF-8 LF Python Lambda Layout: US



Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

24. Test the *GeneratePresignedURL* Lambda function.

 - Choose Deploy.
 - Choose Test, and for Event name, enter test1
 - Choose Save, and then choose Test again.

The response includes a presigned URL, as shown in the following image.

- Copy the presigned URL and paste it in a new browser tab to confirm that it works.
 - If you load the page within 2 minutes, it will return the sample report contents, as shown in the following image.

Hello! This is some sample HTML.

25. Add the *GeneratePresignedURL* Lambda function to the Step Functions state machine.

- Navigate to the Step Functions console.
 - Select **MyStateMachine**, and then choose **Edit**.
 - Search for Lambda
 - Drag the **AWS Lambda Invoke** object onto the canvas to the arrow just *above* the **SNS Publish** object, as shown in the following image

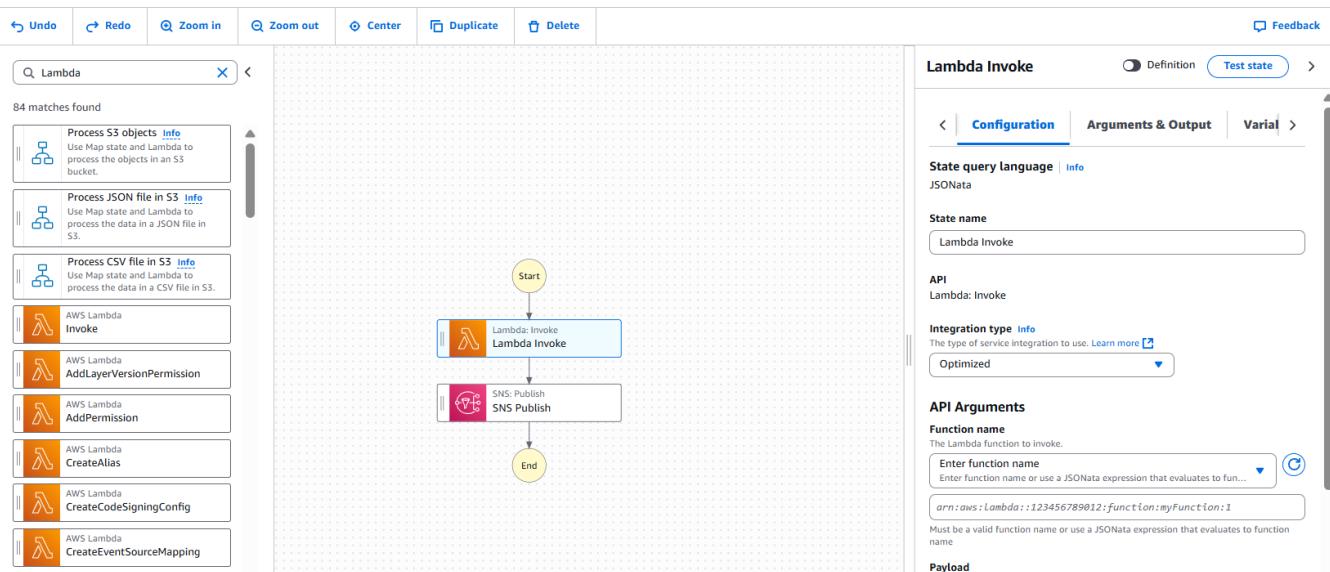


Subject: Cloud Developing (01CT0720)

Experiment No: 12

Date:

Enrolment No: 92200133030



26. Configure the function details in the Lambda Invoke pane.

- State name: Enter GeneratePresignedURL
- Function name: Choose GeneratePresignedURL:\$LATEST.
- Payload: Choose No payload.
- Next state: Choose SNS Publish.
- Choose Save.

The screenshot shows the configuration pane for the 'GeneratePresignedURL' Lambda Invoke state. At the top, it says 'Definition' and 'Test state'. The 'API Arguments' section shows the 'Function name' as 'arn:aws:lambda:us-east-1:993258791170:function:GeneratePresignedURL:\$LATEST'. The 'Payload' section shows 'No payload'. Under 'Additional configuration', there are two checkboxes: 'Wait for callback - optional' (unchecked) and 'Enable cross-account access - optional' (unchecked). The 'Next state' dropdown is set to 'SNS Publish'. The 'Comment - optional' section has an empty text area.



Marwadi University
Faculty of Engineering and Technology
Department of Information and Communication Technology

Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

27. Test the update to the state machine.

- Choose **Execute**.
 - In the code editor, delete the name-value pair that appears on line 2. The input code should only include the brackets, as shown in the following code block.

10

- Choose **Start execution**.
 - To see the details, choose **Execution input and output**
 - Check your email for a notification, and choose the presigned URL in the message to verify that you can load the report.

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030



Task 5: Configuring the REST API to invoke the state machine

28. Configure API Gateway to invoke the Step Functions state machine.

- Navigate to the API Gateway console.
- Choose the link for ProductsApi.
- In the Resources pane, under /create_report, choose POST.
- Choose Integration Request and Edit. Configure the following:
 - Integration type: Choose AWS Service.
 - AWS Region: Choose us-east-1.
 - AWS Service: Choose Step Functions.
 - HTTP method: Choose POST.
 - Action Type: Choose Use action name.
 - Action: Enter StartExecution
 - Execution role: Paste the ARN that you copied in the previous step.
 - The ARN format is arn:aws:iam::account-number:role/RoleForAPIGWTToTriggerStep, where account-number is your AWS account number.
 - Content Handling: Choose Passthrough.
- Request body passthrough: When there are no templates defined(recommended).
- Expand the Mapping Templates section.
- In the Content-Type field, enter application/json and then choose the check mark icon to confirm the entry.
- Choose Method request passthrough
- Replace the contents in Template body with the following code.

```
{
  "input": "$util.escapeJavaScript($input.json('$'))",
  "stateMachineArn": "arn:aws:states:us-east-1:account-
number:stateMachine:MyStateMachine"
}
```

- In the code that you pasted in, replace account-number with your AWS account number.

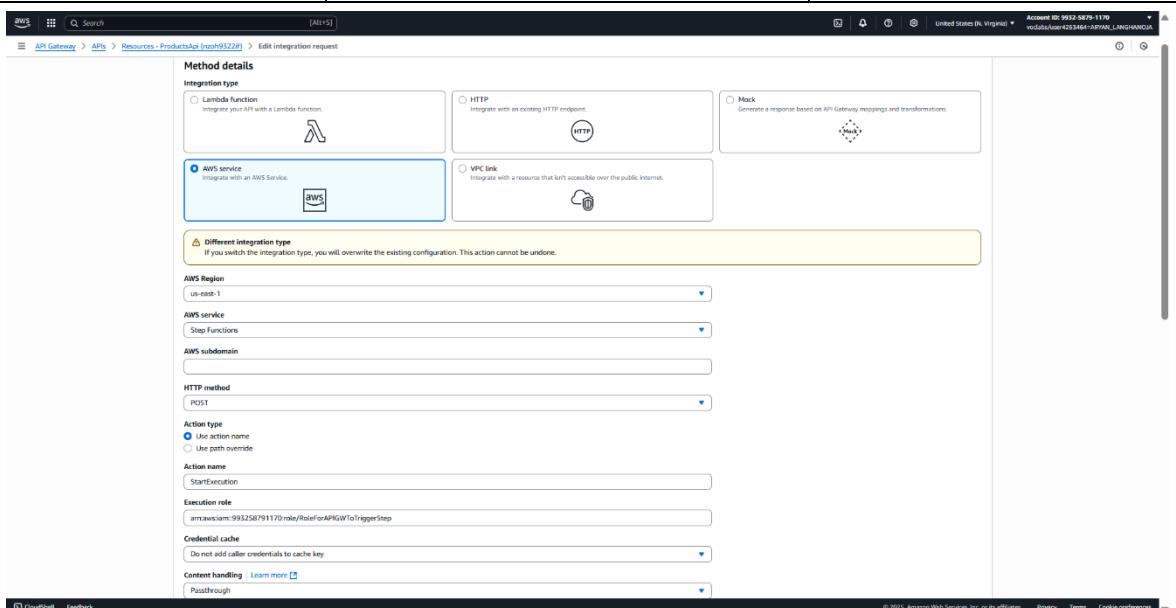
Subject: Cloud Developing (01CT0720)

Aim: Orchestrating serverless functions with step functions.

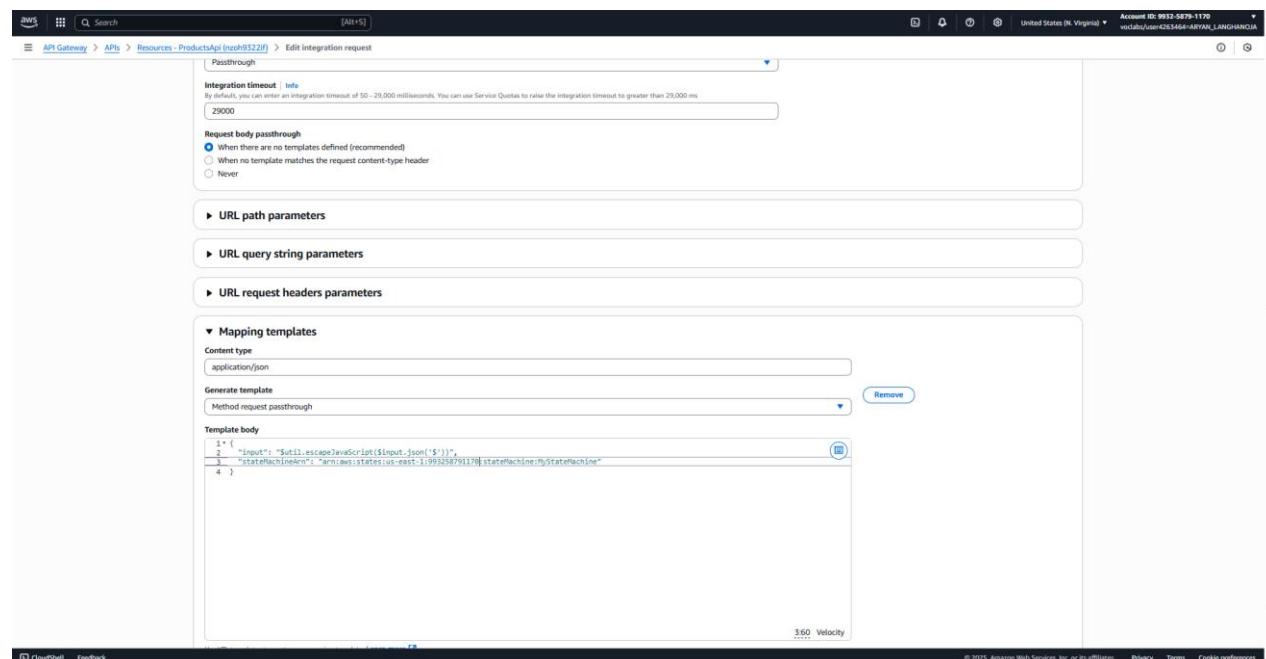
Experiment No: 12

Date:

Enrolment No: 92200133030



The screenshot shows the 'Edit integration request' page for an API endpoint. The 'Integration type' section is set to 'Lambda Function'. Other options like 'HTTP', 'VPC link', and 'AWS service' are also visible. Below this, the 'AWS Region' is set to 'us-east-1', 'AWS service' to 'Step Functions', and 'HTTP method' to 'POST'. The 'Action type' is set to 'Use action name' with 'Action name' as 'StartExecution'. The 'Execution role' is 'arn:aws:iam::993258791170:role/RoleForAPIGWToTriggerStep'. Under 'Content handling', 'Passthrough' is selected. A note at the bottom states: 'If you switch the integration type, you will overwrite the existing configuration. This action cannot be undone.'



The screenshot shows the 'Edit integration request' page for a pass-through integration. The 'Integration type' is set to 'Passthrough'. The 'Integration timeout' is set to '29000'. Under 'Request body passthrough', the option 'When there are no templates defined (recommended)' is selected. Below this, sections for 'URL path parameters', 'URL query string parameters', and 'URL request headers parameters' are shown. The 'Mapping templates' section is expanded, showing 'Content type' as 'application/json', 'Generate template' as 'Method request passthrough', and a 'Template body' containing the following JSON template:

```

1  "input": "util.escapeJavaScript($input.json('$'))",
2  "stateMachineArn": "arn:aws:states:us-east-1:993258791170::stateMachine:myStateMachine"
3
4

```

29. Test the modification to the API.

- Choose the Test tab.
- Keep all the default settings, and at the bottom of the page, choose Test.
- The Response Body section displays a JSON message similar to the following.



Subject: Cloud Developing (01CT0720)

Date:

Enrolment No: 92200133030

```

○ /create_report - POST method test results
Request
  /create_report
    Latency ms
      105
    Status
      200

Response body
{
  "executionArn": "arn:aws:states:us-east-1:993258791170:execution:MyStateMachine-8b2934e1:7f49b470-6fb1-4e7e-ba29-37bf7e89db00",
  "startDate": "1.755879568556E9"
}

Response headers
{
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-68a89890-92ec304140e456eb95f04a0e"
}

Logs
Execution log for request bd9c241d-56e8-448c-9b2b-cb87998e9f7c
Fri Aug 22 16:19:28 UTC 2025 : Starting execution for request: bd9c241d-56e8-448c-9b2b-cb87998e9f7c
Fri Aug 22 16:19:28 UTC 2025 : HTTP Method: POST, Resource Path: /create_report
Fri Aug 22 16:19:28 UTC 2025 : Method request path: {}
Fri Aug 22 16:19:28 UTC 2025 : Method request query string: {}
Fri Aug 22 16:19:28 UTC 2025 : Method request headers: {}
Fri Aug 22 16:19:28 UTC 2025 : Method request body before transformations:
Fri Aug 22 16:19:28 UTC 2025 : Endpoint request URI: https://states.us-east-1.amazonaws.com/?Action=StartExecution
Fri Aug 22 16:19:28 UTC 2025 : Endpoint request headers:
  {Authorization=*****}
  *****e234b
  X-Amz-Date=20250822T161928Z, X-Amz-Endpoint=ap-1-id=nzo93221f, Accept=application/json, User-Agent=AmazonAPIGateway_nzo93221f, X-Amz-SecurityToken=IQoJb3JpZ2luXVjEHM//////////wEAcvZvLWhc3Qm5GNEQC1HD+bUFUKsWkVjY2Awhvc4yCpvN1dDAEYtPf4tcmSA1anv4Z44EDMeoHPV7wSz33CSRDwU9PzX9CqI0xTrOsSrIaggZEAA0DK5Mz110c5MTE3MC1MdX199gbx2zYS5Ks1Mb4r102zFkr7oZLdGltFHkAx5kquCR7EPJ0u4kbwTDnw9RnHe/hruJ5196xTHmEH1j1KXTp/REG9+dK8E7zK8hsx19uLxApFE791wJ3P14zAcmb2cHALUqhMsCj0p0hnh2xdkYxf14m3gr7sdtca03eCYqrUN6jEdy1lztISL7mt4dopSP9M93z5tkP6p1jOUkgVhugYz61+7Q4+7c9ajFOPRna7oju9LpamPhRL47y2dWeKG1LkNuoyNg3v4fcFaly/CHG0uBhTTHi3j5p7xLjQ0P7cjQ0E877/UmR903pbA+gkKSHMTkCasGRnw9LqvB7woIU7FQRpvb [TRUNCATED]
Fri Aug 22 16:19:28 UTC 2025 : Endpoint request body after transformations:
  {
    "input": "{}",
    "stateMachineArn": "arn:aws:states:us-east-1:993258791170:stateMachine:MyStateMachine-8b2934e1"
  }
Fri Aug 22 16:19:28 UTC 2025 : Sending request to https://states.us-east-1.amazonaws.com/?Action=StartExecution
Fri Aug 22 16:19:28 UTC 2025 : Received response Status: 200 Integration latency: 67 ms

```

The screenshot shows a Gmail inbox with the following details:

- Inbox (28 messages):**
 - AWS Notifications**: ["presigned_url_str"] - Testing that my email message works" - If you wish to stop receiving notifications from this topic, please click or visit the link below
 - AWS Notifications**: <https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:993258791170:EmailReport:cBCda2c4-2835-4d41-af66-635c7eac27d7&EndpointLabel=AWS%20Notification>
 - AWS Notifications**: ["presigned_url_str"] - <https://c168617a43402621112942311w993258791170-s3bucket-vmeq1yrmqbv.s3.amazonaws.com/report.html?AWSAccessKeyId=ASIA6OQWWMEB1414CLKA&Sig>
 - AWS Notifications**: to me - ["presigned_url_str"] - <https://c168617a43402621112942311w993258791170-s3bucket-vmeq1yrmqbv.s3.amazonaws.com/report.html?AWSAccessKeyId=ASIA6OQWWMEB1414CLKA&Sig>
- Compose** button
- Search mail** input field
- Message list header: AWS Notification Message
- Message 1: From AWS Notifications, sent at 9:17 AM (12 hours ago). Body: ["presigned_url_str"] - Testing that my email message works" - If you wish to stop receiving notifications from this topic, please click or visit the link below
- Message 2: From AWS Notifications, sent at 9:02 PM (47 minutes ago). Body: <https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:993258791170:EmailReport:cBCda2c4-2835-4d41-af66-635c7eac27d7&EndpointLabel=AWS%20Notification>
- Message 3: From AWS Notifications, sent at 9:31 PM (19 minutes ago). Body: ["presigned_url_str"] - <https://c168617a43402621112942311w993258791170-s3bucket-vmeq1yrmqbv.s3.amazonaws.com/report.html?AWSAccessKeyId=ASIA6OQWWMEB1414CLKA&Sig>
- Message 4: From AWS Notifications, sent at 9:49 PM (0 minutes ago). Body: to me - ["presigned_url_str"] - <https://c168617a43402621112942311w993258791170-s3bucket-vmeq1yrmqbv.s3.amazonaws.com/report.html?AWSAccessKeyId=ASIA6OQWWMEB1414CLKA&Sig>

 <p>Marwadi University Marwadi Chandarana Group</p>	<p>Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology</p>	
<p>Subject: Cloud Developing (01CT0720)</p>	<p>Aim: Orchestrating serverless functions with step functions.</p>	
<p>Experiment No: 12</p>	<p>Date:</p>	<p>Enrolment No: 92200133030</p>

Hello! This is some sample HTML.

Hello! This is some sample HTML.

30. Test generating the presigned URL by using the Invoke URL endpoint.
 - Return to the VS Code IDE.
 - To generate a new presigned URL, run the following command. Replace *invoke-url* with the Invoke URL that you just copied.
curl -X POST invoke-url/create_report
 - The full command looks similar to the following:
curl -X POST https://nzoh9322if.execute-api.us-east-1.amazonaws.com/prod/create_report
 - The result includes a JSON response that contains executionArn and startDate name-value pairs.

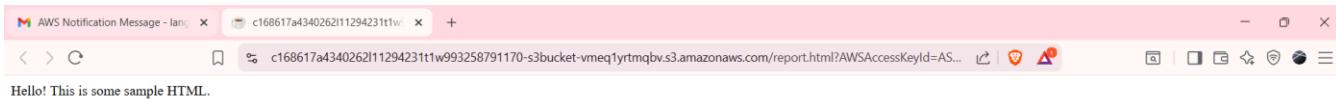
```
[ec2-user@ip-10-0-1-152 environment]$ curl -X POST https://nzo9322if.execute-api.us-east-1.amazonaws.com/prod/create_report  
{"executionArn":"arn:aws:states:us-east-1:993258791170:execution:MyStateMachine-8b2934e1:4a407bc9-53c0-41b4-9e32-b3a9713b17a1","startDate":1.755879938266E9}[ec2-user@ip-10-0-1-152 environment]$
```

The screenshot shows a Gmail inbox with the following details:

- Compose** button.
- Inbox** label with 28 messages.
- Starred**, **Snoozed**, **Sent**, **Drafts**, and **More** buttons.
- Labels** section with a plus sign.
- AWS Notification Message** from **AWS Notifications** at 9:17 AM (12 hours ago). The message body is: {"presigned_url_str": "Testing that my email message works"} - If you wish to stop receiving notifications from this topic, please click or visit the link below.
- AWS Notifications** message at 9:49 PM (7 minutes ago) with the URL: https://c168617a434026211294231t1w993258791170-s3bucket.vmeq1yrtmqbv.s3.amazonaws.com/report.html?AWSAccessKeyId=ASIA6OQWWMEBEEPTOCC&Sig=915f3... The message body is: {"presigned_url_str": "https://c168617a434026211294231t1w993258791170-s3bucket.vmeq1yrtmqbv.s3.amazonaws.com/report.html?AWSAccessKeyId=ASIA6OQWWMEBEEPTOCC&Sig=915f3...".
- AWS Notifications** message at 9:55 PM (1 minute ago) with the URL: https://c168617a434026211294231t1w993258791170-s3bucket.vmeq1yrtmqbv.s3.amazonaws.com/report.html?AWSAccessKeyId=ASIA6OQWWMEBEEPTOCC&Sig=915f3... The message body is: {"presigned_url_str": "https://c168617a434026211294231t1w993258791170-s3bucket.vmeq1yrtmqbv.s3.amazonaws.com/report.html?AWSAccessKeyId=ASIA6OQWWMEBEEPTOCC&Sig=915f3...".

At the bottom, there are buttons for **Reply**, **Forward**, and a smiley face icon. A prompt to "Enable desktop notifications for Gmail." with "OK" and "No thanks" buttons is also visible.

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030



Task 6: Configuring a Lambda function to generate an HTML report

31. Create a Lambda function that will generate an HTML report with data extracted from a JSON object.
 - Navigate to the Lambda console.
 - Verify that you are in the N. Virginia (us-east-1) Region.
 - Choose **Create function** and configure the following:
 - Choose **Author from scratch**.
 - **Function name:** Enter `generateHTML`
 - **Runtime:** Choose **Node.js 20.x**.
 - Expand the **Change default execution role** section.
 - **Execution role:** Choose **Use an existing role**.
 - **Existing role:** Choose **RoleForAllLambdas**.
 - At the bottom of the page, choose **Create function**.
 - Choose the **Configuration** tab, and then choose **Edit** in the **General configuration** section.
 - Adjust the timeout to **2** minutes, and then choose **Save**.
 - Choose the **Code** tab, and in the **Code source** section, replace the existing code with the following.



Subject: Cloud Developing (01CT0720)

Experiment No: 12

Date:

Enrolment No: 92200133030

The screenshot shows the AWS Lambda 'Create function' wizard. At the top, a green success message says 'Successfully updated the function GeneratePresignedURL.' Below it, the 'Basic information' section shows the function name 'generateHTML' and runtime 'Node.js 20.x'. Under 'Architecture', 'x86_64' is selected. In the 'Permissions' section, 'RoleForAllLambdas' is chosen. The 'Additional configurations' section is collapsed.

32. Test the new *generateHTML* Lambda function.

- Choose Deploy.
- Choose the down arrow next to Test, and choose Configure test event.
- For Event name, enter test2
- Paste the following code in as the input (replace the existing lines):

```
{
"my_json_arr": [
{
  "suppliers_id_int": 1,
  "supplier_name_str": "AnyCompany coffee suppliers",
  "supplier_address_str": "123 Any Street",
  "bean_info_obj_arr": [
    {
      "type_str": "Arabica",
      "product_name_str": "Best bean",
      "quantity_int": "300",
      "price_int_str": "18.00"
    },
    {
      "type_str": "Robusta",
      "product_name_str": "Great bean",
      "quantity_int": "100",
      "price_int_str": "12.00"
    }
  ]
},
{
  "type_str": "Arabica",
  "product_name_str": "Best bean",
  "quantity_int": "300",
  "price_int_str": "18.00"
}
],
}
},
```

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

```
{
  "suppliers_id_int": 2,
  "supplier_name_str": "Central Example Corp. coffee",
  "supplier_address_str": "100 Main Street",
  "bean_info_obj_arr": [
    {
      "type_str": "Robusta",
      "product_name_str": "Top bean",
      "quantity_int": "200",
      "price_int_str": "10.00"
    },
    {
      "type_str": "Liberica",
      "product_name_str": "Better bean",
      "quantity_int": "150",
      "price_int_str": "14.00"
    }
  ]
}
• Choose Save, and then choose Test again.
• The response includes the following JSON message.
{
  "msg_str": "Report published to S3"
}
```

Task 7: Adding the GenerateHTML function to the state machine

33. Edit the Step Functions state machine to add a parallel state.
- Navigate to the Step Functions console.
 - In the left navigation pane, choose State machines.
 - Select MyStateMachine, and then choose Edit.
 - Choose the Flow tab in the left pane.
 - Drag the Parallel object onto the canvas, just above the Lambda Invoke object.
 - Drag the GeneratePresignedURL object to the box labeled *Drop state here* on the right, under the Parallel state object, as shown in the following image.

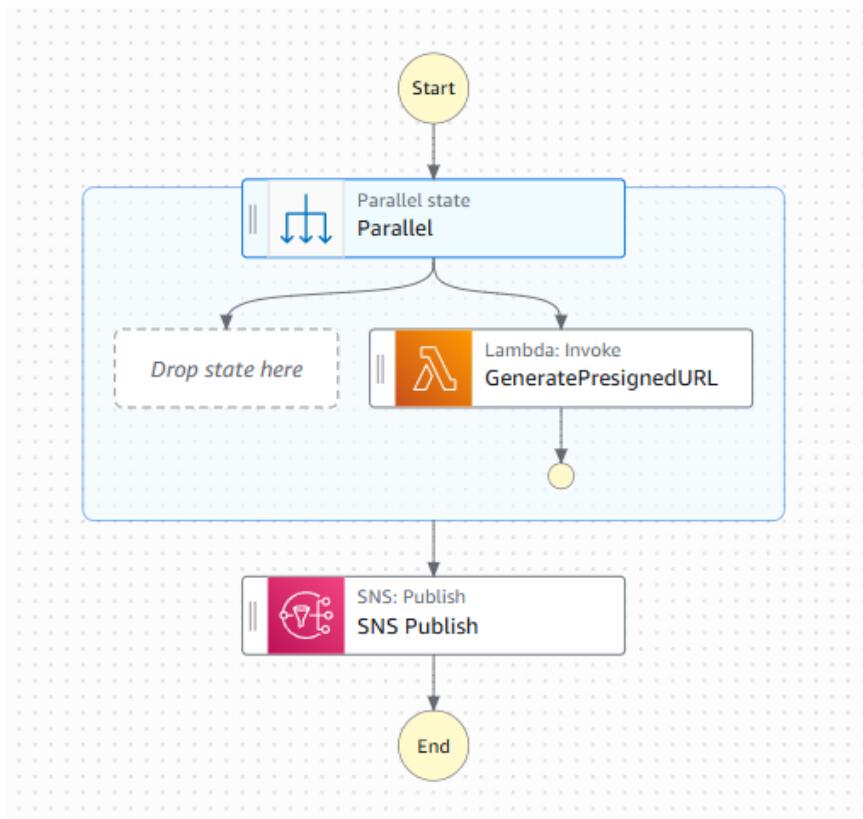


Subject: Cloud Developing (01CT0720) **Aim: Orchestrating serverless functions with step functions.**

Experiment No: 12

Date:

Enrolment No: 92200133030



34. Add the *GenerateHTML* function to the state machine, and configure it.

- Choose the Actions tab in the left pane.
- Search for Lambda
- Drag the AWS Lambda Invoke object onto the canvas to the box labeled *Drop state here* on the left, under the Parallel state object.
- Choose the new Lambda Invoke object and configure the following function details in the Lambda Invoke pane.
 - State name: Enter generateHTML
 - Function name: Choose generateHTML:\$LATEST.
 - Payload: Choose Use state input as payload.
 - Next state: Choose Go to end.



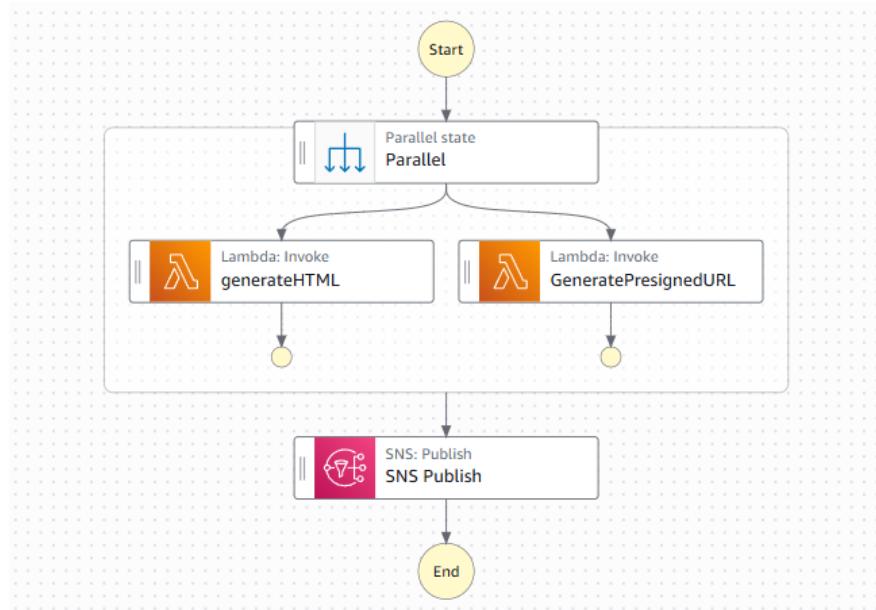
Subject: Cloud Developing (01CT0720)

Aim: Orchestrating serverless functions with step functions.

Experiment No: 12

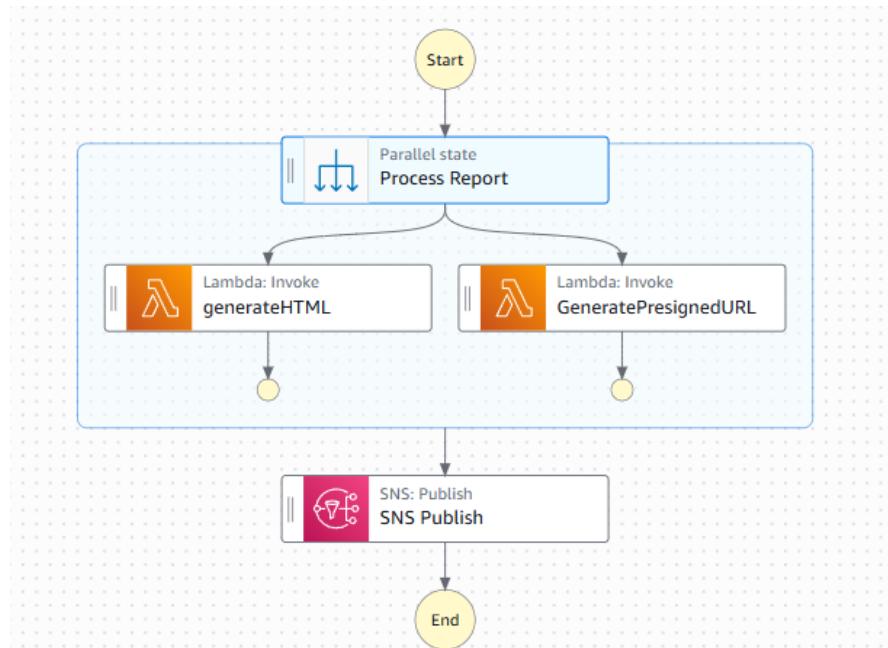
Date:

Enrolment No: 92200133030



35. Configure the Parallel state card details, and then save the changes.

- On the canvas, choose the Parallel state object.
- For **State name**, enter **Process Report**
- Choose **Save**.



36. Test the updated state machine.



Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

37. Choose Execute and configure the following:

In the code editor, replace the existing JSON code with the following.

{

```
"my_json_arr": [
  {
    "suppliers_id_int": 1,
    "supplier_name_str": "Dave coffee suppliers",
    "supplier_address_str": "123 Any Street",
    "bean_info_obj_arr": [
      {
        "type_str": "Arabica",
        "product_name_str": "Best bean",
        "quantity_int": "300",
        "price_int_str": "18.00"
      },
      {
        "type_str": "Robusta",
        "product_name_str": "Great bean",
        "quantity_int": "100",
        "price_int_str": "12.00"
      }
    ]
  },
  {
    "suppliers_id_int": 2,
    "supplier_name_str": "Central Example Corp. coffee",
    "supplier_address_str": "100 Main Street",
    "bean_info_obj_arr": [
      {
        "type_str": "Robusta",
        "product_name_str": "Top bean",
        "quantity_int": "200",
        "price_int_str": "10.00"
      },
      {
        "type_str": "Liberica",
        "product_name_str": "Better bean",
        "quantity_int": "150",
        "price_int_str": "14.00"
      }
    ]
  }
]
```



Marwadi University
Faculty of Engineering and Technology
Department of Information and Communication Technology

Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

Notice that the input now consists of an array.

- Choose Start execution.
 - To see the details, choose Execution input and output.
 - Check your email for a new notification with a presigned URL.

The email now includes two name-value pairs, in the following format, which is truncated for brevity:

```
[{"msg_str": "Report published to S3"}, {"presigned url str": "https://...&Expires=..."}]
```

www.EasyEngineering.net

 Search [Alt+S]

aws Search [Alt+Z] United States (N. Virginia) Account Id: 983-8376-4564-AIRAN_ANGHANIA

Step Functions > State machines > MyStateMachine-Blk2054r1 > Execution: 8f7fdbba-ecc7-431e-8de1-7a466bdbf5a2

Execution started successfully.

Execution: 8f7fdbba-ecc7-431e-8de1-7a466bdbf5a2

Edit state machine New execution Actions

Details Execution input and output Definition

State input

```
1 * {  
2 *   "my_json_arr": [  
3 *     {  
4 *       "supplier_id_int": 1,  
5 *       "supplier_name_str": "Dave coffee suppliers",  
6 *       "supplier_address_str": "123 Any Street",  
7 *       "bean_info_obj_arr": [  
8 *         {  
9 *           "type_str": "Arabica",  
10 *          "product_name_str": "Best bean",  
11 *          "quantity_int": "800",  
12 *          "price_int_str": "10.00"  
13 *        },  
14 *        {  
15 *          "type_str": "Robusta",  
16 *          "product_name_str": "Great bean",  
17 *          "quantity_int": "1000",  
18 *          "price_int_str": "12.00"  
19 *        }  
20 *      ],  
21 *    },  
22 *    {  
23 *      "suppliers_id_int": 2,  
24 *      "supplier_name_str": "Central Express Corp.",  
25 *      "supplier_address_str": "100 Main Street",  
26 *      "bean_info_obj_arr": [  
27 *        {  
28 *          "type_str": "Robusta",  
29 *          "product_name_str": "Top bean",  
30 *          "quantity_int": "2000"  
31 *        }  
32 *      ]  
33 *    }  
34 *  ]  
35 * }  
36 *
```

State output

```
1 * {  
2 *   "MessageId": "20200707T0133-5140-8014-d65e431ea9f0b",  
3 *   "AllAttachments": {},  
4 *   "AllHeaders": {},  
5 *   "X-Amzn-RequestID": "  
6 *     "Adder398d-367c-46c1-8259-c0798fdadd35"  
7 *   ",  
8 *   "Connection": [  
9 *     "keep-alive"  
10 *   ],  
11 *   "Content-Length": [  
12 *     "204"  
13 *   ],  
14 *   "Date": [  
15 *     "Fri, 22 Aug 2025 16:50:44 GMT"  
16 *   ],  
17 *   "Content-Type": [  
18 *     "text/xml"  
19 *   ]  
20 * },  
21 *   "HttpHeaders": [  
22 *     "connection": "keep-alive"  
23 *     "Content-Length": "204"  
24 *     "Content-Type": "text/xml"  
25 *   ],  
26 *   "Date": "Fri, 22 Aug 2025 16:50:44 GMT",  
27 *   "X-Amzn-RequestID": "Adder398d-367c-46c1-8259-c0798fdadd35"  
28 * },  
29 *   "HttpStatusCode": 200  
30 * },  
31 *   "SdkResponseMetadata": {}  
32 * }  
33 * }  
34 * }  
35 * }  
36 *
```

Graph view Table view

CloudShell Feedback

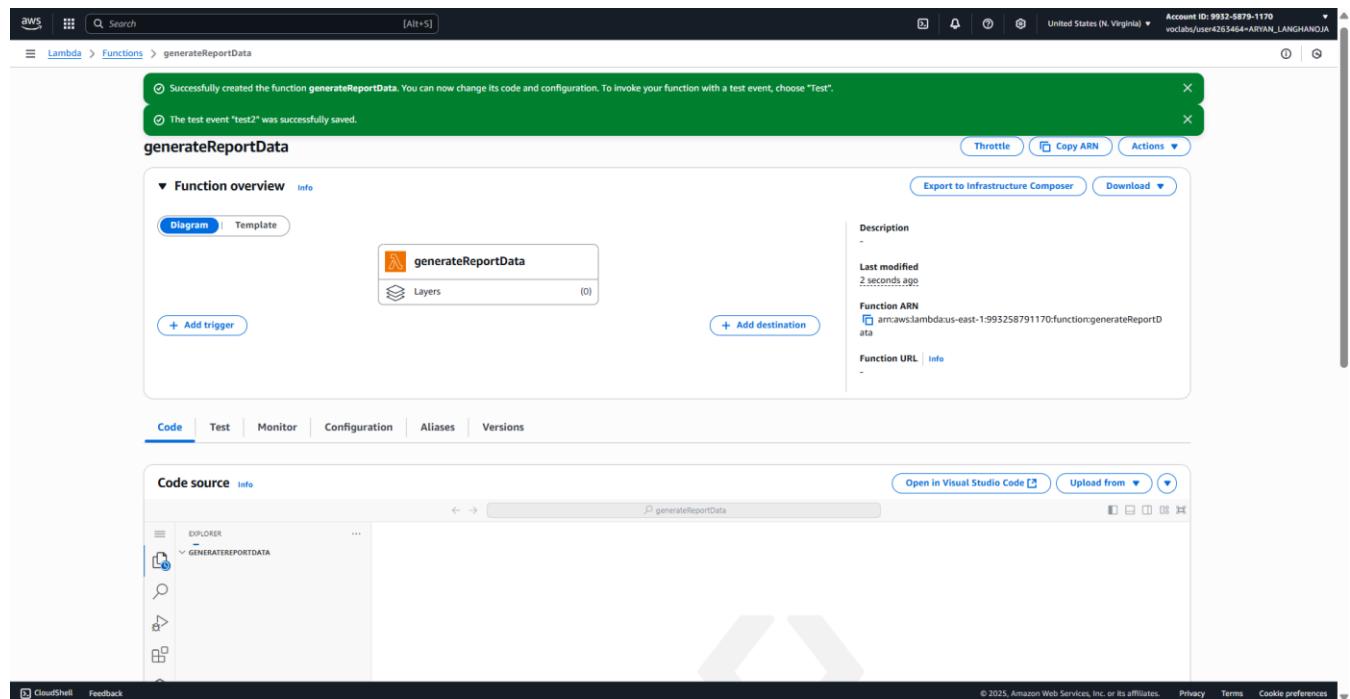
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

Task 8: Creating a Lambda function to retrieve supplier records

38. Create a Lambda function to query the database.

- Navigate to the Lambda console.
- Verify that you are in the N. Virginia (us-east-1) Region.
- Choose **Create function** and configure the following:
 - Choose **Author from scratch**.
 - **Function name:** Enter `generateReportData`
 - **Runtime:** Choose **Node.js 20.x**.
 - Expand the **Change default execution role** section.
 - **Execution role:** Choose **Use an existing role**.
 - **Existing role:** Choose **RoleForAllLambdas**.
- At the bottom of the page, choose **Create function**.
- Choose the **Configuration** tab, and then choose **Edit**.
- Set the timeout for this function to **1** minute and save the change.





Subject: Cloud Developing (01CT0720)

Aim: Orchestrating serverless functions with step functions.

Experiment No: 12

Date:

Enrolment No: 92200133030

39. Create an index.js file.

- Return to the VS Code IDE.
- In the **Environment** window, open the context (right-click) menu for the root directory, and then choose **New Folder**.
- Name the folder `lambda`
- Inside the **lambda** folder, create a new file named `index.js`
- Paste the following code into the file.

```
import { S3Client } from '@aws-sdk/client-s3';
import mysql from 'mysql2/promise';

const s3Client = new S3Client({ region: "us-east-1" });

export const handler = async (event, context) => {
    try {
        // Establish MySQL connection
        const connection = await mysql.createConnection({
            host: 'DB-ENDPOINT',
            user: 'nodeapp',
            password: 'coffee',
            database: 'COFFEE'
        });

        // Query suppliers
        const [suppliers] = await connection.query('SELECT * FROM suppliers');

        // Query beans
        const [beans] = await connection.query('SELECT * FROM beans');

        // Close connection
        await connection.end();

        // Merge data
        const mergedData = mergeData(suppliers, beans);

        // Return the merged result
        return {
            statusCode: 200,
            body: JSON.stringify({
                my_json_arr: mergedData
            })
        };
    } catch (err) {
        console.error("Error executing query:", err);
        return {
            statusCode: 500,
            body: JSON.stringify({
                error: 'Failed to retrieve data from MySQL',
                details: err.message
            })
        };
    }
}
```



Subject: Cloud Developing (01CT0720)

Aim: Orchestrating serverless functions with step functions.

Experiment No: 12

Date:

Enrolment No: 92200133030

```
};

// Function to merge suppliers and beans data
function mergeData(suppliers_arr, beans_arr) {
    const fine_tunes_data_arr = [];

    for (const supplier of suppliers_arr) {
        const o = {
            suppliers_id_int: supplier.id,
            supplier_name_str: supplier.name,
            supplier_address_str: supplier.address,
            supplier_phone_str: supplier.phone,
            bean_info_obj_arr: []
        };

        for (const bean of beans_arr) {
            if (supplier.id === bean.supplier_id) {
                const b = {
                    type_str: bean.type,
                    product_name_str: bean.product_name,
                    quantity_int: bean.quantity
                };
                o.bean_info_obj_arr.push(b);
            }
        }

        fine_tunes_data_arr.push(o);
    }

    return fine_tunes_data_arr;
}
```

- Retrieve the value for the database endpoint:
 - i. In a new browser tab, navigate to the Amazon RDS console.
 - ii. Choose **Databases**, and then choose the **supplierdb** link.
 - iii. Copy the **Writer - Endpoint** value for the database. The value displays in the **Connectivity & security** section.
- Return to the VS Code IDE.
- In the *index.js* file, replace the **DB-ENDPOINT** placeholder, which appears on line 9, with the database endpoint that you just copied. Be sure to keep the single quotes around it.
- Save the file.

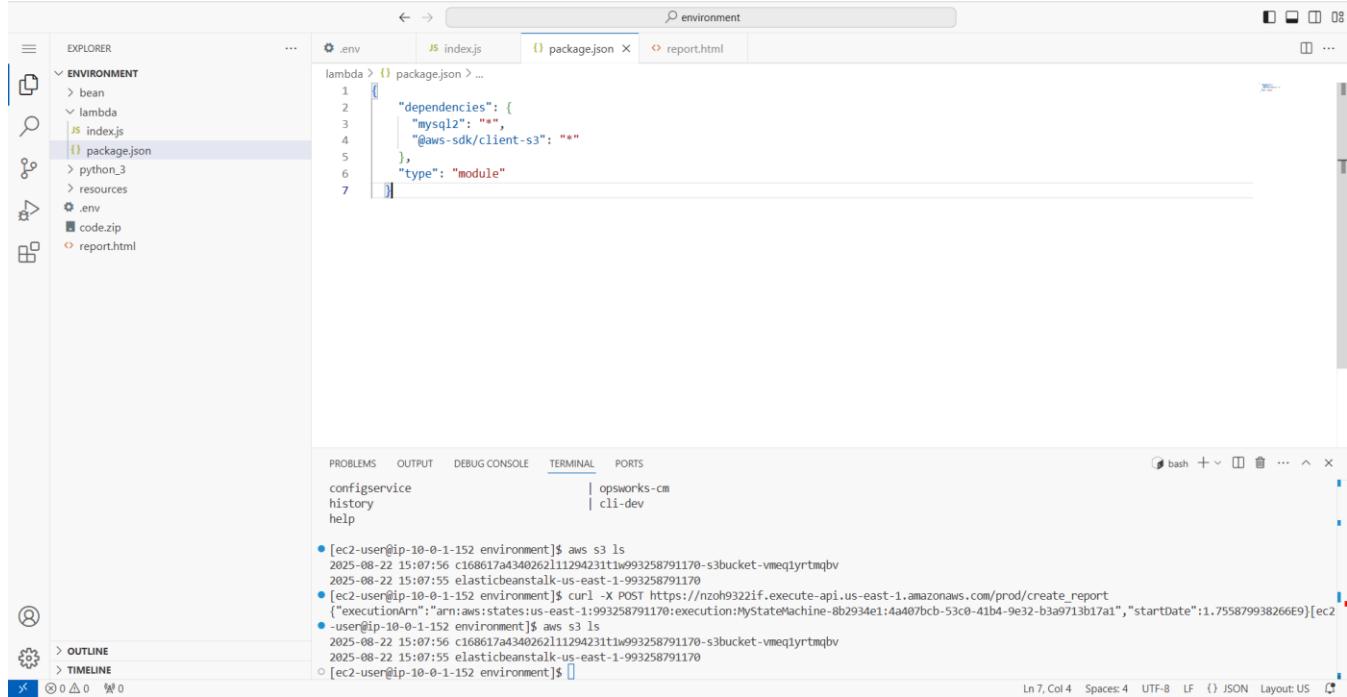
40. Create a package.json file.

- In the VS Code IDE, create a new file named **package.json** in the **lambda** folder.
- Paste the following code into the new file, and save the changes.

```
{
  "dependencies": {
    "mysql12": "*",
    "@aws-sdk/client-s3": "*"
  },
  "type": "module"
```

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.
Experiment No: 12	Date: Enrolment No: 92200133030

}



The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows a project structure with files like .env, index.js, package.json, report.html, and resources. The package.json file is open in the center editor, displaying the following code:

```

1 "dependencies": {
2   "mysql": "*",
3   "@aws-sdk/client-s3": "*"
4 },
5 "type": "module"
6
7

```

Below the editor is the Terminal panel, which displays a series of AWS CLI commands and their outputs. The commands include aws s3 ls, curl -X POST, and aws s3 cp. The output shows the creation of an S3 bucket and the upload of files to it.

41. Generate the .zip package.

- To install npm and then create a .zip archive file named *lambda.zip* that contains the two files that you just created, as well as the node modules, run the following commands in the VS Code IDE terminal.

```

cd ~/environment/lambda
npm install
zip -r ..../lambda.zip *

```

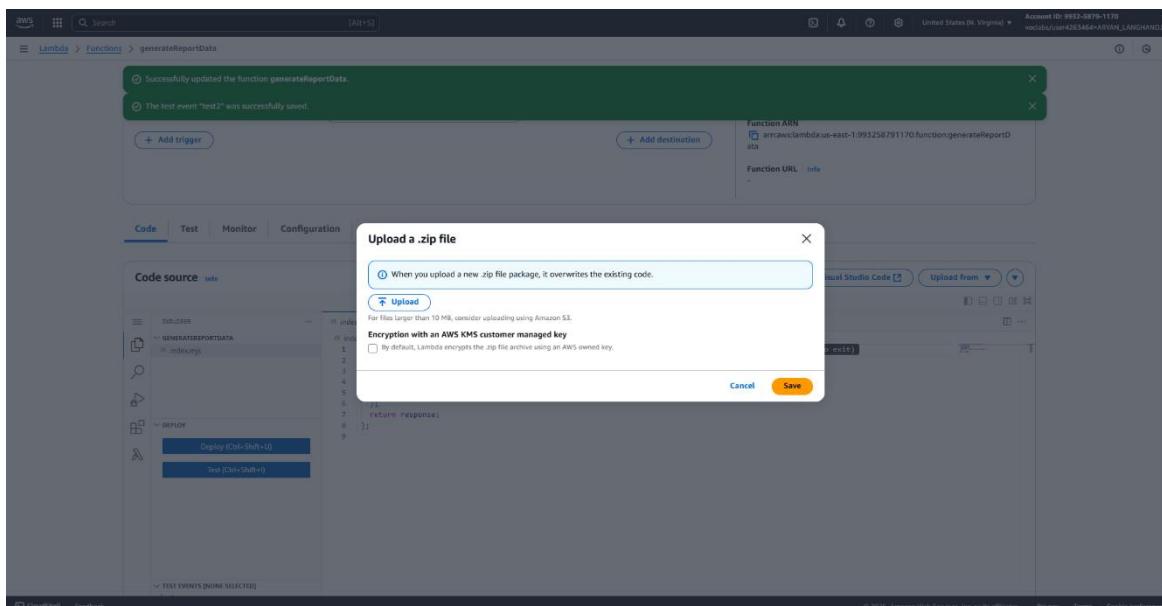
- Download the resulting *lambda.zip* file from the VS Code IDE to your computer:
 - In the **Environment** window, open the context (right-click) menu for the **lambda.zip** file, and then choose **Download**.
 - When prompted, save the file.

 <p>Marwadi University Marwadi Chandarana Group</p>	<p>Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology</p>	
<p>Subject: Cloud Developing (01CT0720)</p>	<p>Aim: Orchestrating serverless functions with step functions.</p>	
<p>Experiment No: 12</p>	<p>Date:</p>	<p>Enrolment No: 92200133030</p>

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash - lambda + × ⓘ
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/getFlexibleChecksumsPlugin.d.ts (deflated 68%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/header.d.ts (deflated 29%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/headerWithPrefix.d.ts (deflated 32%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/index.d.ts (deflated 51%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/isChecksumWithPartNumber.d.ts (deflated 4%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/listStreaming.d.ts (deflated 18%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/NODE_REQUEST_CHECKSUM_CALCULATION_CONFIG_OPTIONS.d.ts (deflated 54%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/NODE_RESPONSE_CHECKSUM_VALIDATION_CONFIG_OPTIONS.d.ts (deflated 55%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/resolveFlexibleChecksumsConfig.d.ts (deflated 60%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/selectChecksumAlgorithmFunction.d.ts (deflated 49%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/stringLasher.d.ts (deflated 34%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/stringToSelector.d.ts (deflated 48%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/types.d.ts (deflated 41%)
adding: node_modules/@aws-sdk/middleware-flexible-checksums/dist-types/validateChecksumFromResponse.d.ts (deflated 49%)
adding: node_modules/@aws-sdk/middleware-host-header/ (stored 0%)
adding: node_modules/@aws-sdk/middleware-host-header/LICENSE (deflated 65%)
adding: node_modules/@aws-sdk/middleware-host-header/dist-cjs/ (stored 0%)
adding: node_modules/@aws-sdk/middleware-host-header/dist-cjs/index.js (deflated 63%)
adding: node_modules/@aws-sdk/middleware-host-header/dist-es/ (stored 0%)
adding: node_modules/@aws-sdk/middleware-host-header/dist-es/index.js (deflated 58%)
adding: node_modules/@aws-sdk/middleware-host-header/package.json (deflated 60%)
adding: node_modules/@aws-sdk/middleware-host-header/README.md (deflated 61%)
adding: node_modules/@aws-sdk/middleware-host-header/dist-types/ (stored 0%)
adding: node_modules/@aws-sdk/middleware-host-header/dist-types/index.d.ts (deflated 63%)
adding: node_modules/@aws-sdk/middleware-host-header/dist-types/tss4/ (stored 0%)
adding: node_modules/@aws-sdk/middleware-host-header/dist-types/tss4/index.d.ts (deflated 63%)
adding: node_modules/@aws-sdk/middleware-location-constraint/ (stored 0%)
adding: node_modules/@aws-sdk/middleware-location-constraint/dist-es/ (stored 0%)
adding: node_modules/@aws-sdk/middleware-location-constraint/dist-es/configuration.js (deflated 8%)
adding: node_modules/@aws-sdk/middleware-location-constraint/dist-es/index.js (deflated 60%)
adding: node_modules/@aws-sdk/middleware-location-constraint/dist-cjs/ (stored 0%)
adding: node_modules/@aws-sdk/middleware-location-constraint/dist-cjs/index.js (deflated 64%)
adding: node_modules/@aws-sdk/middleware-location-constraint/package.json (deflated 60%)
```

42. Finish configuring the `generateReportData` Lambda function.

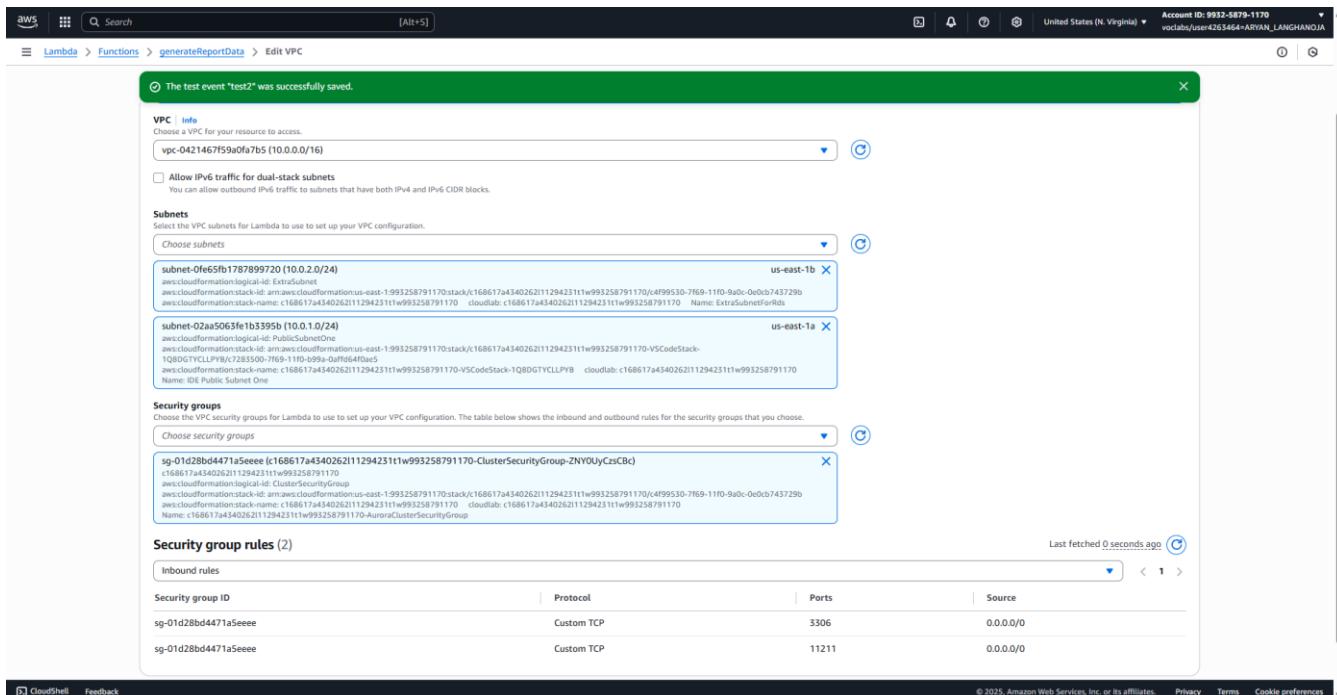
- Return to the Lambda console, where you were configuring the *generateReportData* function.
 - Choose the **Code** tab.
 - In the **Code source** section, choose **Upload from**, and then choose **.zip file**.
 - Choose **Upload**, and then browse to and open the *lambda.zip* file that you just saved to your computer.
 - Choose **Save**.
 - Verify that the **Environment** panel in the **Code source** section now shows that the files contained in the .zip file were uploaded and extracted successfully, as shown in the following image.



 Marwadi University Marwadi Chandarana Group	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

43. Configure the *generateReportData* Lambda function to access resources in the virtual private cloud (VPC).

- In the Lambda console, choose the Configuration tab, and then choose VPC.
- In the VPC section, choose Edit.
- Choose the VPC named Lab IDE VPC.
- For Subnets, choose *both* us-east-1a and us-east-1b.
- For Security groups, choose the security group that has ClusterSecurityGroup in the name.
- When you are satisfied that your settings are correct, choose Save.
- A message that says *Updating the function generateReportData* appears in a blue bar at the top of the console.



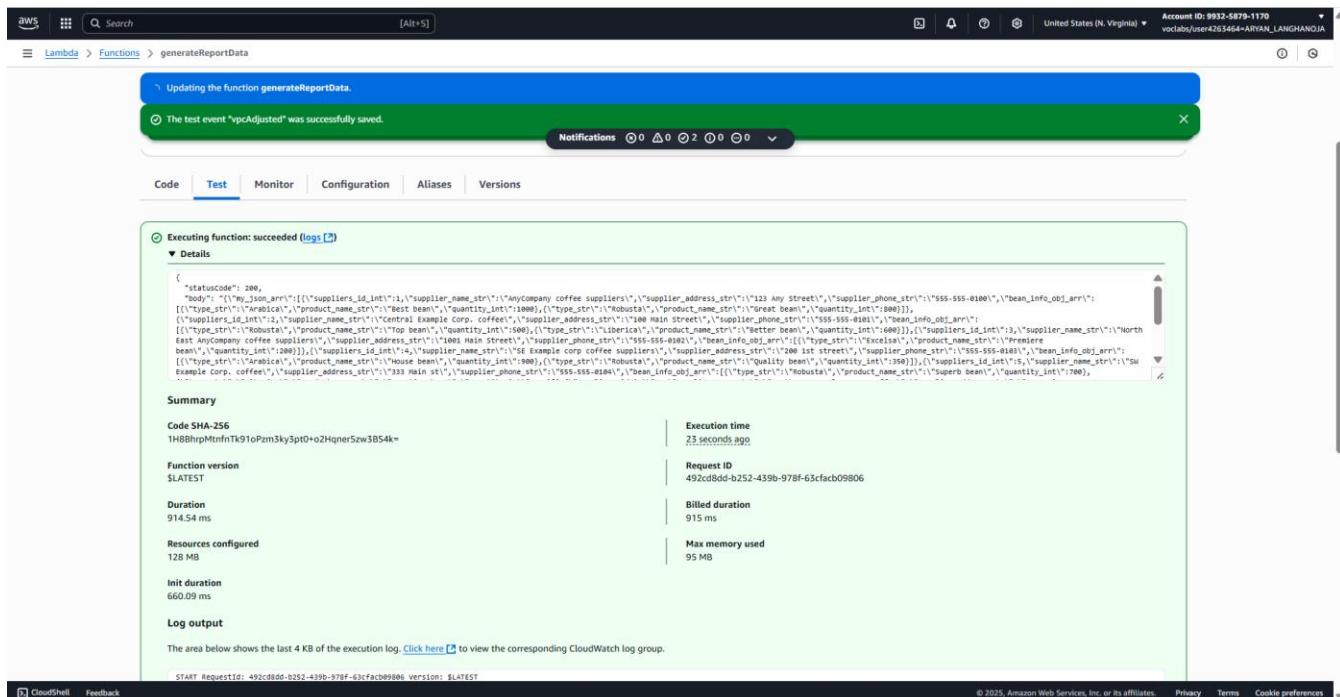
The screenshot shows the AWS Lambda VPC configuration page. At the top, a green banner indicates that the test event "test2" was successfully saved. Below this, the "VPC" tab is selected. Under "Choose a VPC for your resource to access," the VPC "vpc-0421467f59a0fa7b5 (10.0.0.0/16)" is chosen. The "Subnets" section shows two subnets: "us-east-1b" and "us-east-1a". The "Security groups" section shows a single security group "sg-01d28bd4471a5eeee". The bottom part of the screenshot shows a table of "Security group rules (2)".

Inbound rules	Protocol	Ports	Source
sg-01d28bd4471a5eeee	Custom TCP	3306	0.0.0.0/0
sg-01d28bd4471a5eeee	Custom TCP	11211	0.0.0.0/0

44. Test the *generateReportData* function with the new VPC settings applied.

- In the **Lambda** console, return to the **Code** tab.
- Choose the down arrow next to **Test**, and choose **Configure test event**.
- Choose **Create new test event**.
- For **Event name**, enter `vpcAdjusted`
- Choose **Save**, and then choose **Test** again.
- The response includes all eight records from the Amazon RDS database.

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.
Experiment No: 12	Date:



The screenshot shows the AWS Lambda Test interface. A blue header bar indicates "Updating the function generateReportData." Below it, a message says "The test event "vpcAdjusted" was successfully saved." The main area has tabs for "Code", "Test" (which is selected), "Monitor", "Configuration", "Aliases", and "Versions". The "Logs" tab is open, showing the log output of the successful execution. The log output is a large JSON object representing the generated report data. The "Summary" section provides execution details: Code SHA-256 (1HB8hrpMtnTkh91oPzm3ky3pt0+o2HqnerSz3BS4k=), Function version (\$LATEST), Duration (914.54 ms), Resources configured (128 MB), Init duration (660.09 ms), and Log output (area below shows the last 4 KB of the execution log). The bottom status bar includes CloudShell, Feedback, © 2025, Amazon Web Services, Inc. or its affiliates, Privacy, Terms, and Cookie preferences.

Task 9: Adding the generateReportData function to the state machine

45. Add the *generateReportData* function to the state machine.

- Navigate to the Step Functions console.
- Select MyStateMachine, and then choose Edit.
- Search for Lambda
- Drag an AWS Lambda Invoke object onto the canvas, *above* the Process Report parallel state object.

46. Configure the function details in the Lambda Invoke pane.

- State name: Enter *getRealData*
- Function name: Choose *generateReportData:\$LATEST*.
- Payload: Choose No payload.
- Choose Save.



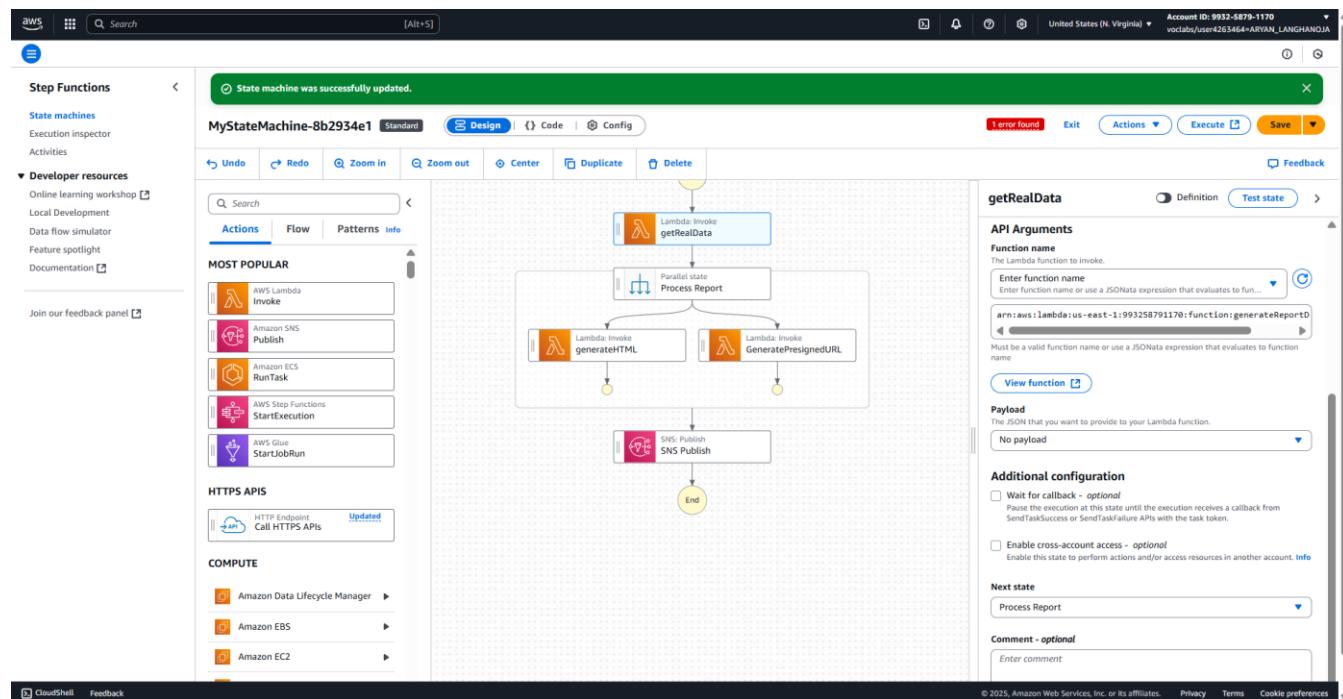
Subject: Cloud Developing (01CT0720)

Aim: Orchestrating serverless functions with step functions.

Experiment No: 12

Date:

Enrolment No: 92200133030



47. Test the completed state machine by using the Step Functions console.

- Choose Execute and configure the following:
 - In the code editor, delete the name-value pair that appears on line 2. The input code should only include the brackets, as shown in the following code block.

{
}

- Choose Start execution.
- Check your email for a notification, and choose the presigned URL in the message to verify that you can load the report.

Start execution

Name
ab584178-7f00-41df-87ef-034b51b23efc
Must be 1-80 characters. Can use alphanumeric characters, dashes, or underscores.

Input - optional
Enter input values for this execution in JSON format.
Format JSON Export Import

```
1: {  
2: }
```

Start execution with latest revision
 Open in a new browser tab

 <p>Marwadi University Marwadi Chandarana Group</p>	<p>Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology</p>	
<p>Subject: Cloud Developing (01CT0720)</p>	<p>Aim: Orchestrating serverless functions with step functions.</p>	
<p>Experiment No: 12</p>	<p>Date:</p>	<p>Enrolment No: 92200133030</p>

48. Test the *create report* REST API endpoint.

- Navigate to the API Gateway console.
 - Choose the link for **ProductsApi**.
 - In the navigation pane, choose **Stages**.
 - In the **Resources** pane, choose **prod**.
 - Copy the **Invoke URL**.
 - Return to the VS Code IDE.
 - In the terminal, enter `curl -X POST` but *do not run the command yet*.
 - Add a space and then paste in the invoke URL that you just copied. *Do not run the command yet*.
 - Add `/create_report` to the end of the invoke URL.
 - The resulting command looks similar to the following, where *unique-string* is some unique string:
`curl -X POST https://unique-string.execute-api.us-east-1.amazonaws.com/prod/create_report`
 - Run the command.
 - The response looks similar to the following.

```
curl -X POST https://unique-string.execute-api.us-east-1.amazonaws.com/prod/create_report
```

- Run the command.
 - The response looks similar to the following.

```
{
  "executionArn": "arn:aws:states:us-east-1:221264991720:express:MyStateMachine:453b6f3d-ab5a-4ac3-a2d2-0d74f728a740:243fc35-ff80-4f8d-bc2a-b0580391277e", "startDate": 1.632788164722E9
}
```



Marwadi
University
Marwadi Chandarana Group

Marwadi University
Faculty of Engineering and Technology
Department of Information and Communication Technology

Subject: Cloud Developing (01CT0720)

Aim: Orchestrating serverless functions with step functions.

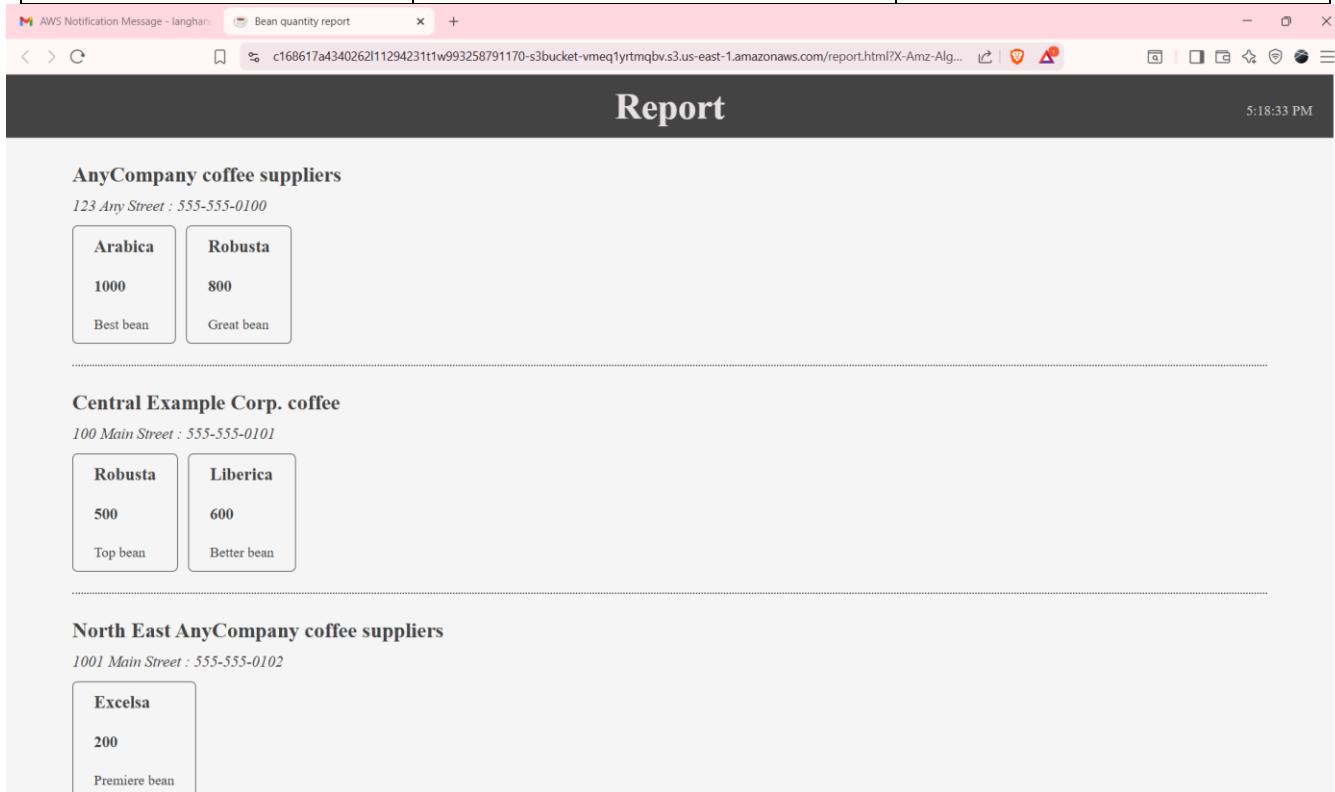
Experiment No: 12

Date:

Enrolment No: 92200133030

- Check your email for a notification, and load the report using the presigned URL contained in the email.

 Marwadi University <small>Marwadi Chandarana Group</small>	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030



AnyCompany coffee suppliers
123 Any Street : 555-555-0100

Arabica	Robusta
1000	800
Best bean	Great bean

Central Example Corp. coffee
100 Main Street : 555-555-0101

Robusta	Liberica
500	600
Top bean	Better bean

North East AnyCompany coffee suppliers
1001 Main Street : 555-555-0102

Excelsa
200
Premiere bean

Conclusion:-

- I connected to the VS Code IDE using the provided LabIDEURL and LabIDEPASSWORD.
- I downloaded the lab resources into the environment using:

```
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/11-lab-step/code.zip -P /home/ec2-user/environment
```

```
unzip code.zip
```

- I upgraded the AWS CLI by giving permissions and executing the setup script:

```
chmod +x ./resources/setup.sh && ./resources/setup.sh
```

- I verified the AWS CLI version and Python SDK installation:

```
aws --version
```

```
pip3 show boto3
```

- I checked the café website by opening index.html from the S3 bucket in the browser.
- I configured an SNS topic named EmailReport and subscribed via email.
- I confirmed the subscription and tested notifications by publishing a message.
- I created an IAM role (RoleForStepToCreateAReport) for Step Functions.
- I designed a state machine in Step Functions with SNS Publish as the first state and executed it with test JSON input:

```
{
```



Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

```
"presigned_url_str": "Testing that my email message works"
```

```
}
```

- I created a sample HTML report (report.html), uploaded it to the S3 bucket, and tested object access:

```
bucket=$(aws s3api list-buckets --query "Buckets[].Name" | grep
s3bucket | tr -d ',' | sed -e 's/"//g' | xargs)
aws s3 cp report.html s3://$bucket/ --cache-control "max-age=0"
aws s3 presign s3://$bucket/report.html --expires-in 30
```

- I created and tested the GeneratePresignedURL Lambda function in Python 3.9, verified its execution, and integrated it into the Step Functions state machine.
- I created another Lambda function (generateHTML) in Node.js 20.x to generate an HTML report from a JSON object and tested it with sample supplier/bean JSON data.
- I updated the Step Functions state machine to include a Parallel State with both GeneratePresignedURL and generateHTML Lambda functions running in parallel.
- I created another Lambda function (generateReportData) in Node.js 20.x to fetch supplier records from the RDS database using mysql2.
- I packaged the function:

```
cd ~/environment/lambda
npm install
zip -r ./lambda.zip *
```

- I uploaded lambda.zip to AWS Lambda, configured it with VPC and subnets, and successfully tested database queries.
- I integrated generateReportData into the state machine before the parallel tasks.
- I tested the final state machine execution with empty {} input and successfully received a report email containing both generated HTML and a presigned URL.
- I configured API Gateway with /create_report endpoint mapped to Step Functions StartExecution action.
- I tested the REST API with:

```
curl -X POST https://nzoh9322if.execute-api.us-east-
1.amazonaws.com/prod/create_report
```

- I received execution details in JSON and verified the email notification with a working report link.
- Successfully orchestrated **serverless functions** with **AWS Step Functions, Lambda, SNS, S3, RDS, and API Gateway**.
- Automated a workflow where a report is generated dynamically, stored securely, and shared via presigned URL through an email notification.



Subject: Cloud Developing (01CT0720)	Aim: Orchestrating serverless functions with step functions.	
Experiment No: 12	Date:	Enrolment No: 92200133030

Result :-

Total score	50/50
[Task 2] Created SNS topic	5/5
[Task 3A] Created state machine	5/5
[Task 3B] Added SNS Publish action to state machine	5/5
[Task 4A] Created GeneratePresignedURL Lambda function	5/5
[Task 4B] Added GeneratePresignedURL action to state machine	5/5
[Task 5] Configured create_report integration	5/5
[Task 6] Created generateHTML Lambda function	5/5
[Task 7] Added generateHTML action to state machine	5/5
[Task 8] Created generateReportData Lambda function	5/5
[Task 9] Added getRealData action to state machine	5/5