Name : Ganesh Gupta **RollNo**./Div:13/D15C

Experiment 11

Aim: To understand **AWS Lambda**, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Theory:

AWS Lambda

A fully managed, serverless computing service where you run code without provisioning or managing servers. Lambda automatically scales your application based on the number of incoming requests or events, ensuring efficient resource utilization. You are only charged for the time your code is running, with no upfront cost, making it cost-effective for on-demand workloads.

Lambda Workflow

- **Create a Function**: Write the function code and define its handler (entry point). You can use the AWS Console, CLI, or upload a deployment package.
- **Set Event Sources**: Define how the function is triggered (e.g., when an object is uploaded to S3 or a DynamoDB table is updated).
- **Execution**: When triggered, Lambda runs your function, executes the logic, and automatically scales to handle the incoming event volume.
- **Scaling and Concurrency**: Lambda scales automatically by launching more instances of the function to handle simultaneous invocations. There are also options for configuring **reserved concurrency** to manage traffic.
- **Monitoring and Logging**: Lambda integrates with Amazon CloudWatch for logging and monitoring. Logs for each invocation are sent to CloudWatch, allowing you to track performance and troubleshoot errors.

AWS Lambda Functions

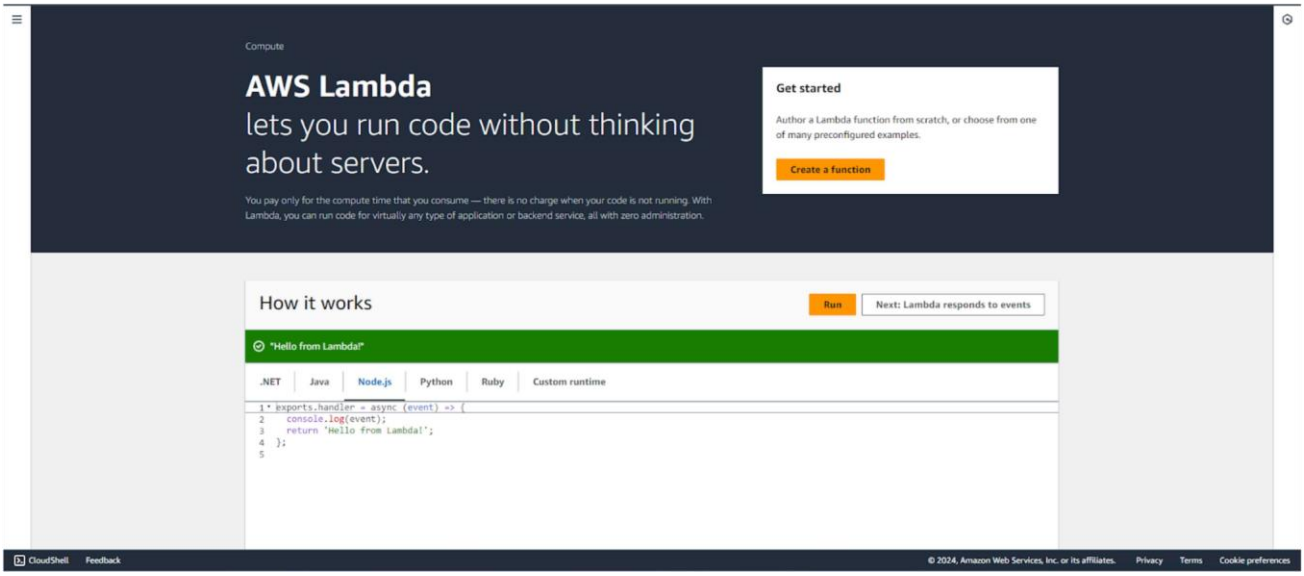
- **Python**: Great for quick development with its rich standard library and support for lightweight tasks.
- **Java**: Typically used for more complex, compute-intensive tasks. While it's robust, cold start times can be higher.
- **Node.js**: Excellent for I/O-bound tasks like handling APIs or streaming data, with fast startup times and efficient memory usage.

Prerequisites: AWS Personal/Academy Account

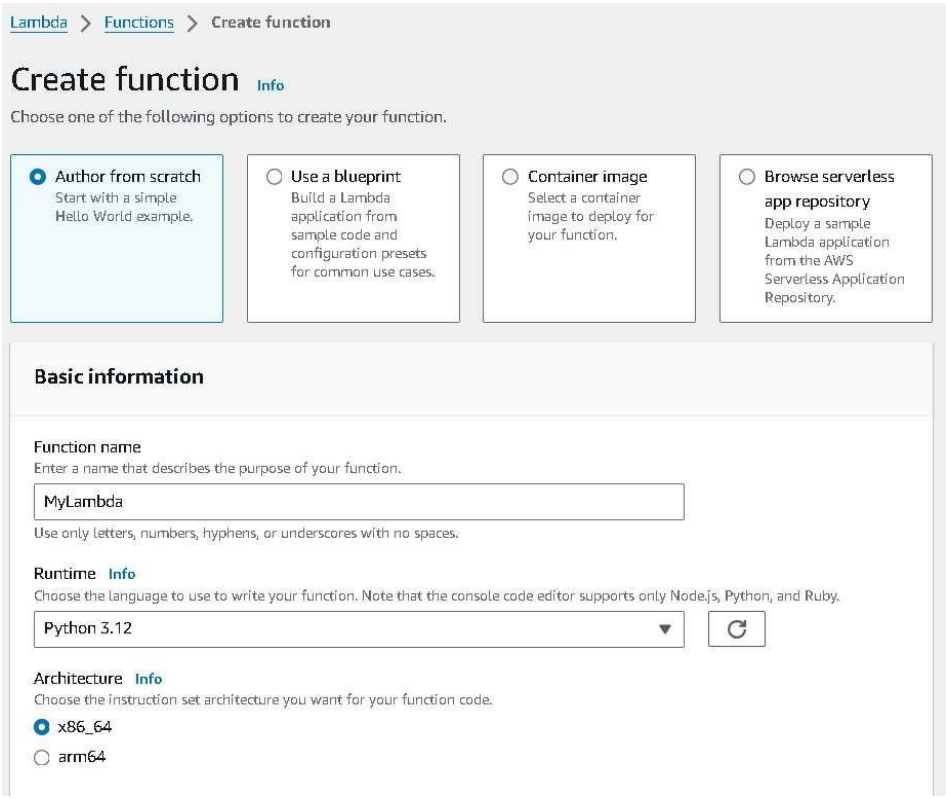
Name : Ganesh Gupta **RollNo**./Div:13/D15C

Steps To create the lambda function:

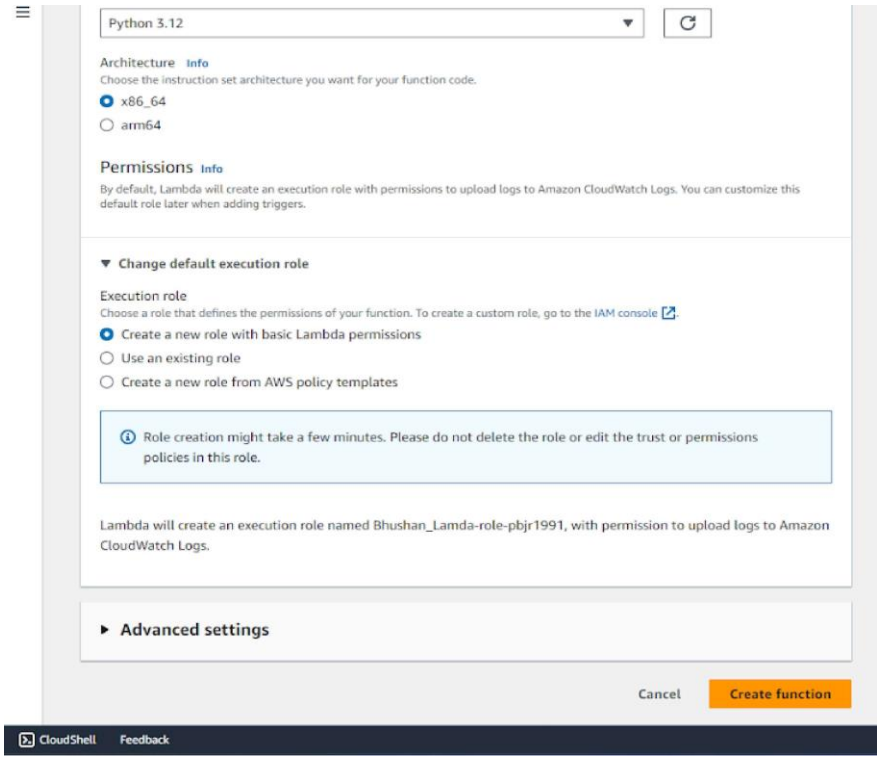
Step 1: Login to your AWS Personal/Academy Accout.Open lambda and click on create function button.

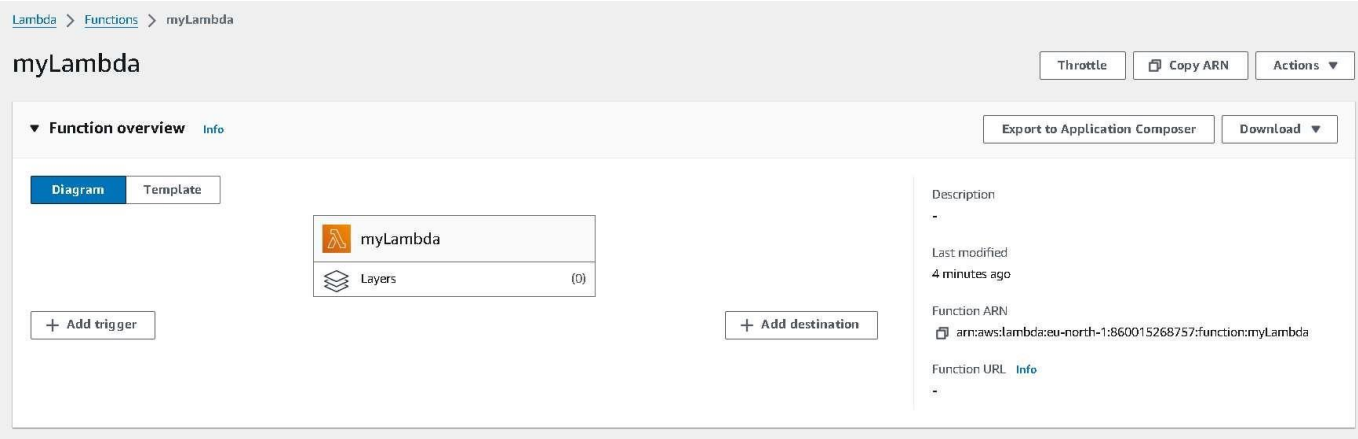


Step 2: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12, Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.



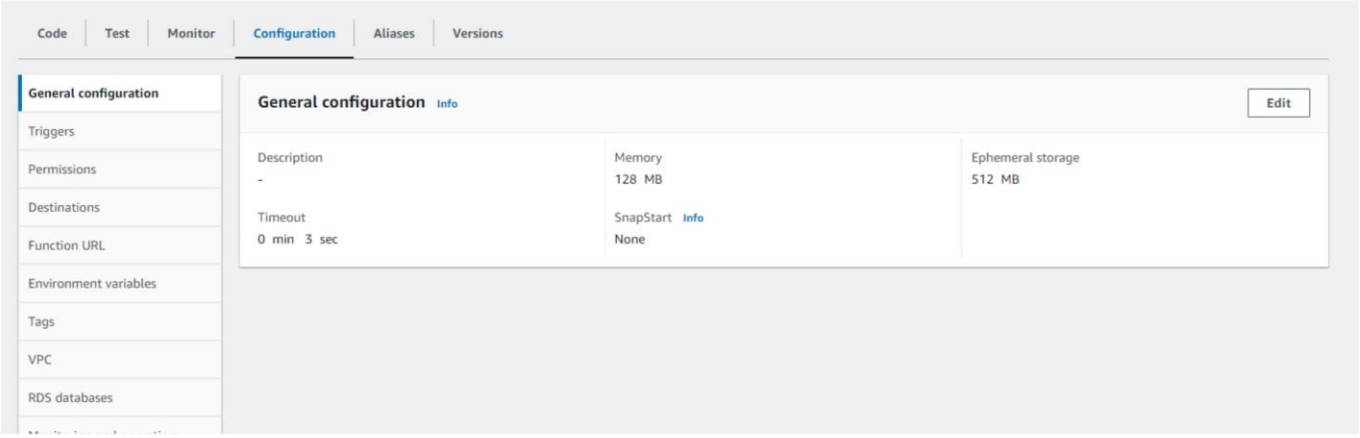
Name : Ganesh Gupta **RollNo**./Div:13/D15C



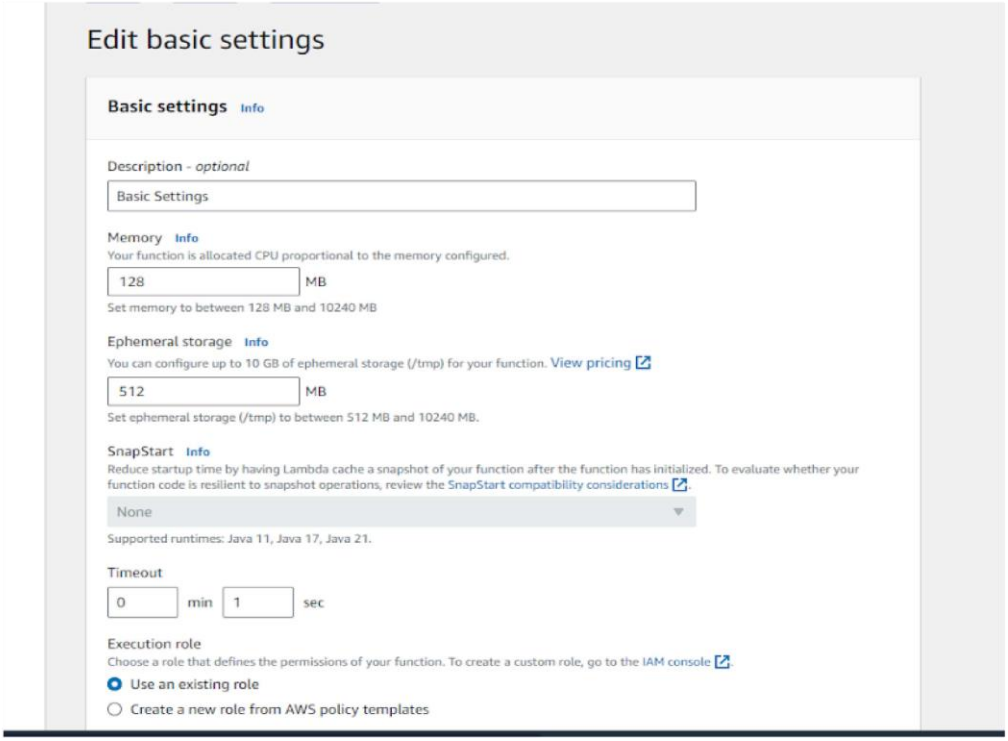


Name : Ganesh Gupta **RollNo**./Div:13/D15C

So See or Edit the basic settings go to configuration then click on edit general setting.

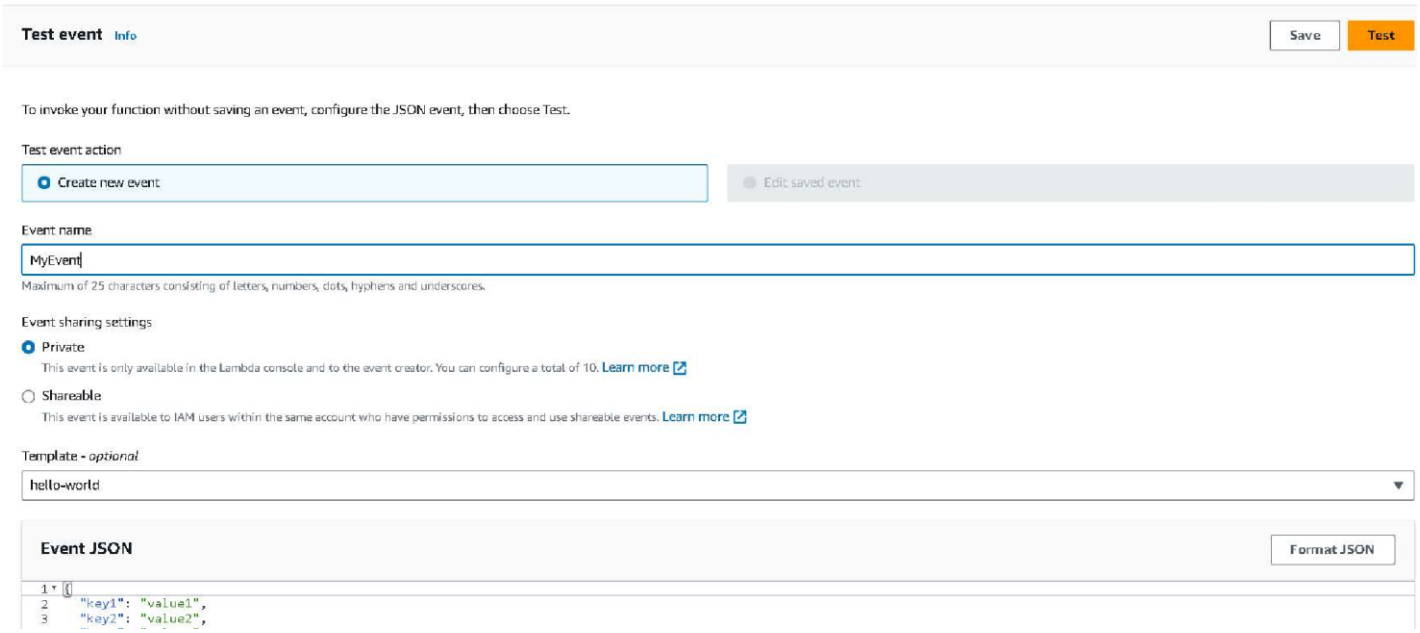


Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

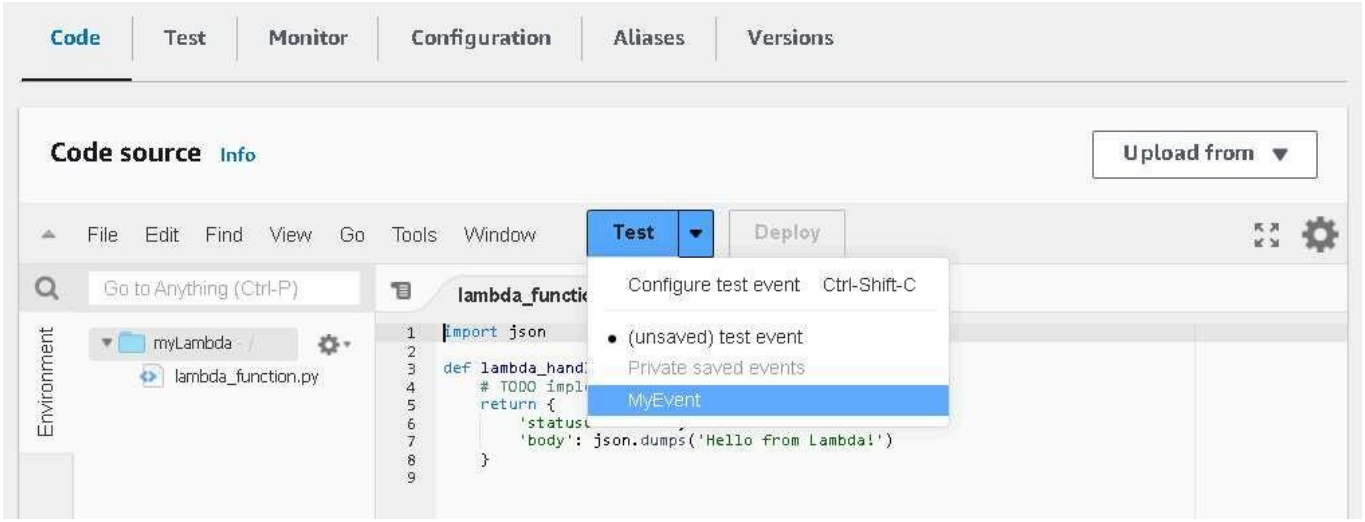


Name : Ganesh Gupta **RollNo**./Div:13/D15C

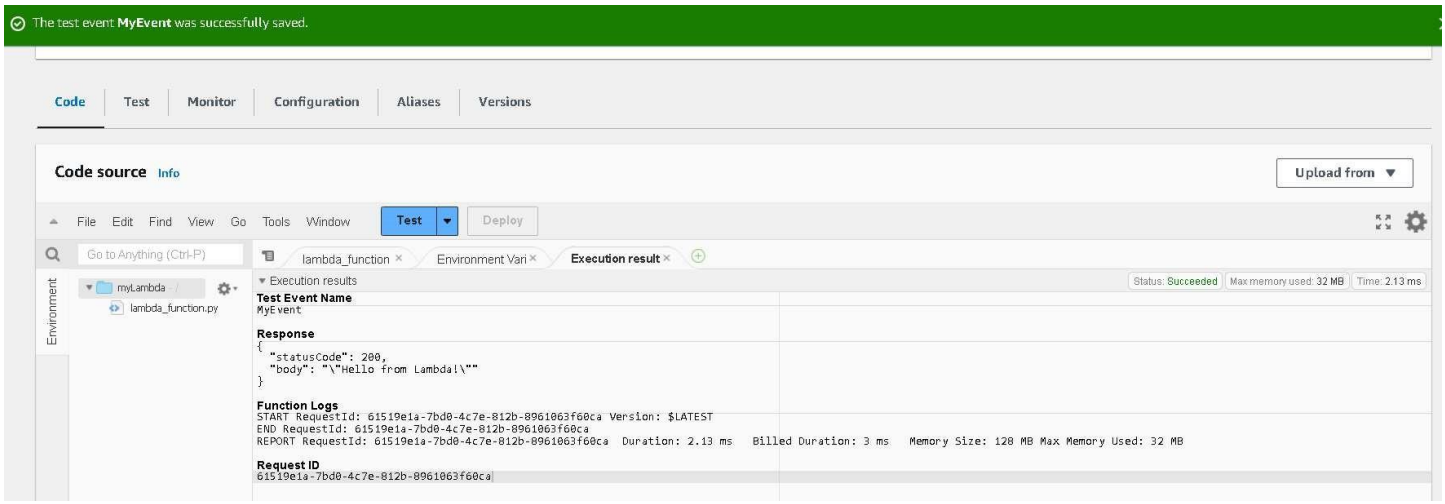
Step 3: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select hello-world template.



Step 4: Now In the Code section select the created event from the dropdown of test then click on test . You will see the below output.

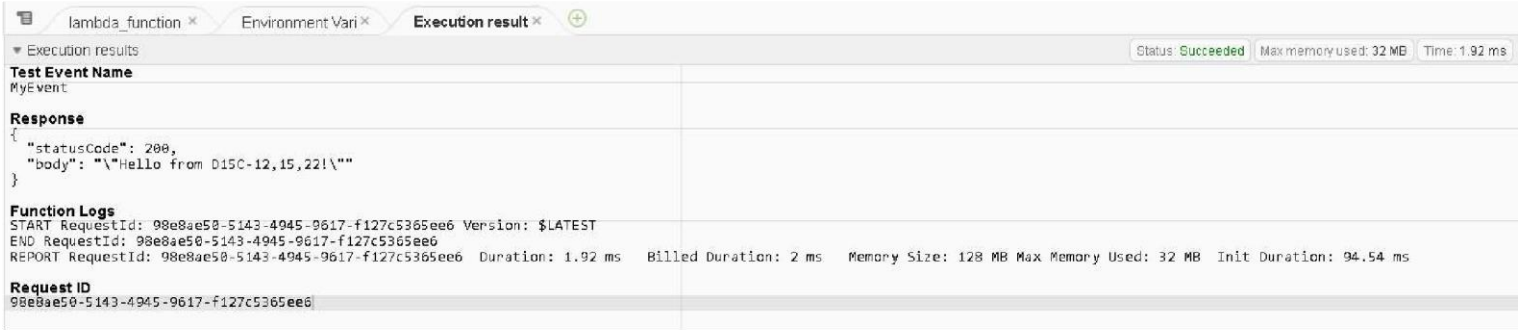


Name : Ganesh Gupta **RollNo**./Div:13/D15C



Step 5: You can edit your lambda function code. I have changed the code to display the new String.

Step 6: Now click on the test and observe the output. We can see the status code 200 and your string output and function logs. On successful deployment.



Conclusion:

In this experiment, we successfully implemented an AWS Lambda function, covering all the key steps involved. Starting with the function's setup in Python, we configured essential settings such as adjusting the timeout to 1 second. A test event was then created, followed by deploying the function and verifying its output. We also made code modifications to the Lambda function, redeployed it, and observed the real-time effects of these changes. This hands-on experience highlighted the ease and adaptability of AWS Lambda for building serverless applications, enabling developers to concentrate on writing code while AWS handles infrastructure and scalability.