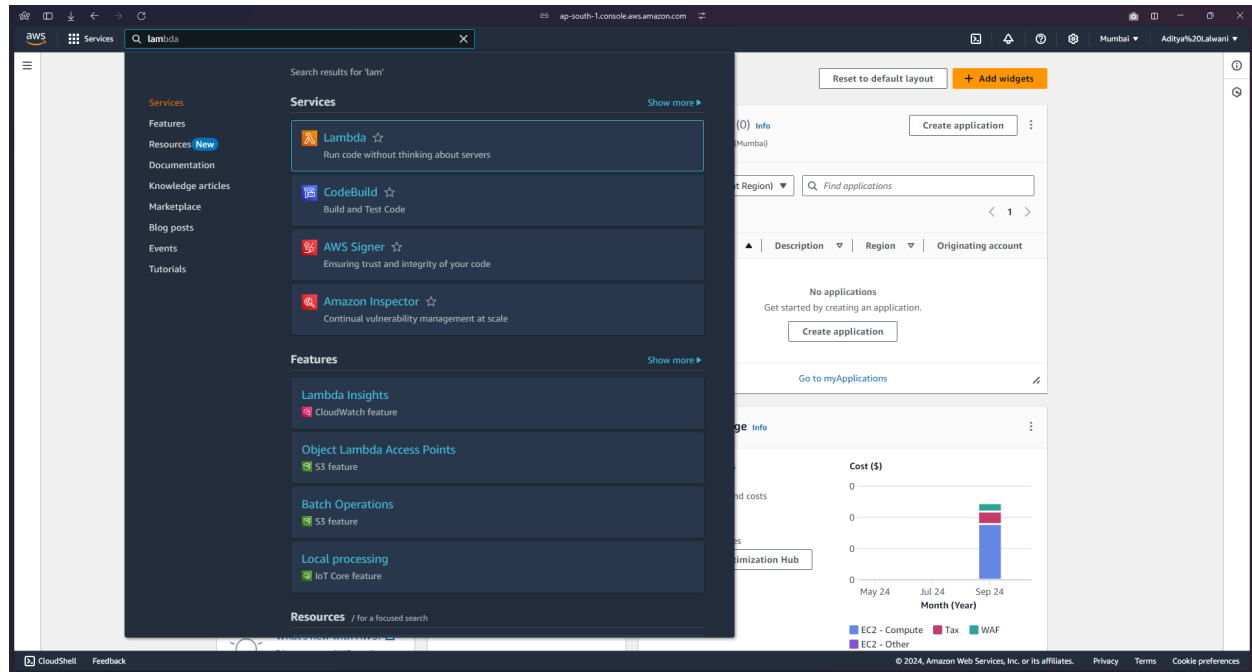


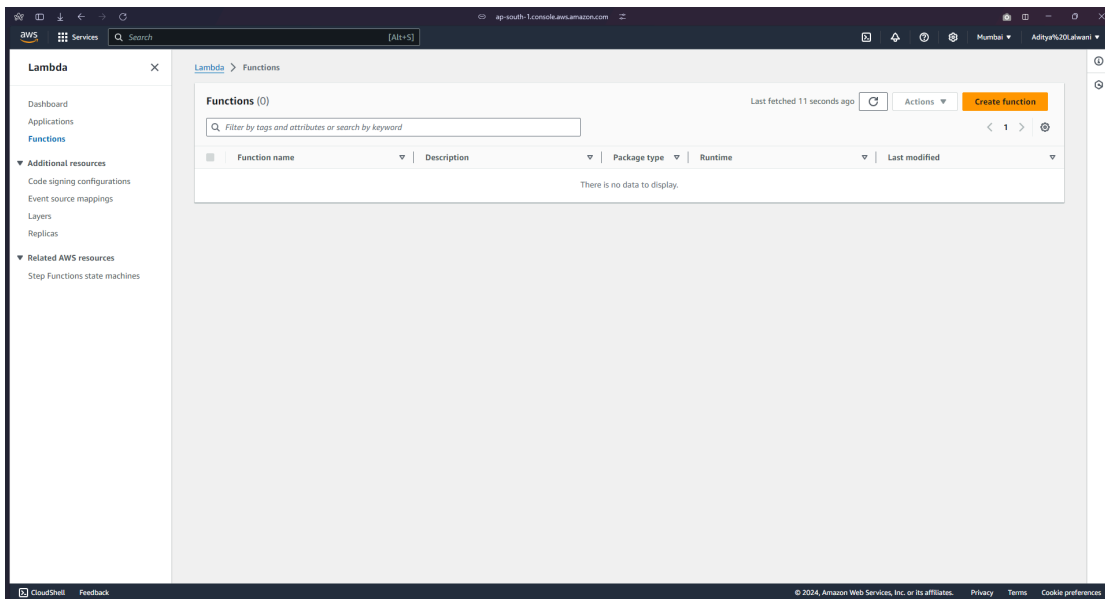
Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

## Set up AWS Lambda Function

Search for Lambda in the services tab. Click on it once found.



Click on create functions.



Give a name to your Lambda function. Select the runtime as Node.js 20.x

Basic information

**Function name**  
Enter a name that describes the purpose of your function.  
  
Use only letters, numbers, hyphens, or underscores with no spaces.

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

**Architecture** info  
Choose the instruction set architecture you want for your function code.  
☒ x86\_64  
☐ arm64

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

▼ Change default execution role

**Execution role**  
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console [link](#).

☐ Create a new role with basic Lambda permissions  
☒ Use an existing role  
☐ Create a new role from AWS policy templates

**Existing role**  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  
   
[View the LabRole role](#) on the IAM console.

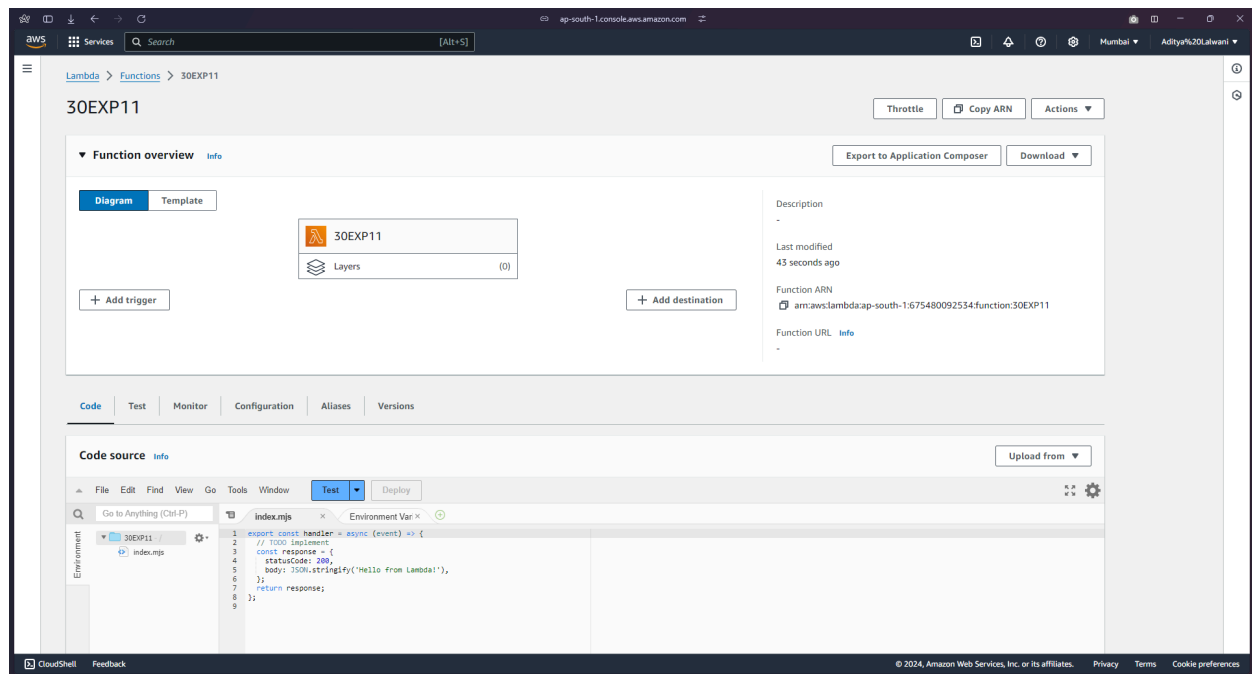
Once the function is created, click on the name of the function

Lambda > Functions

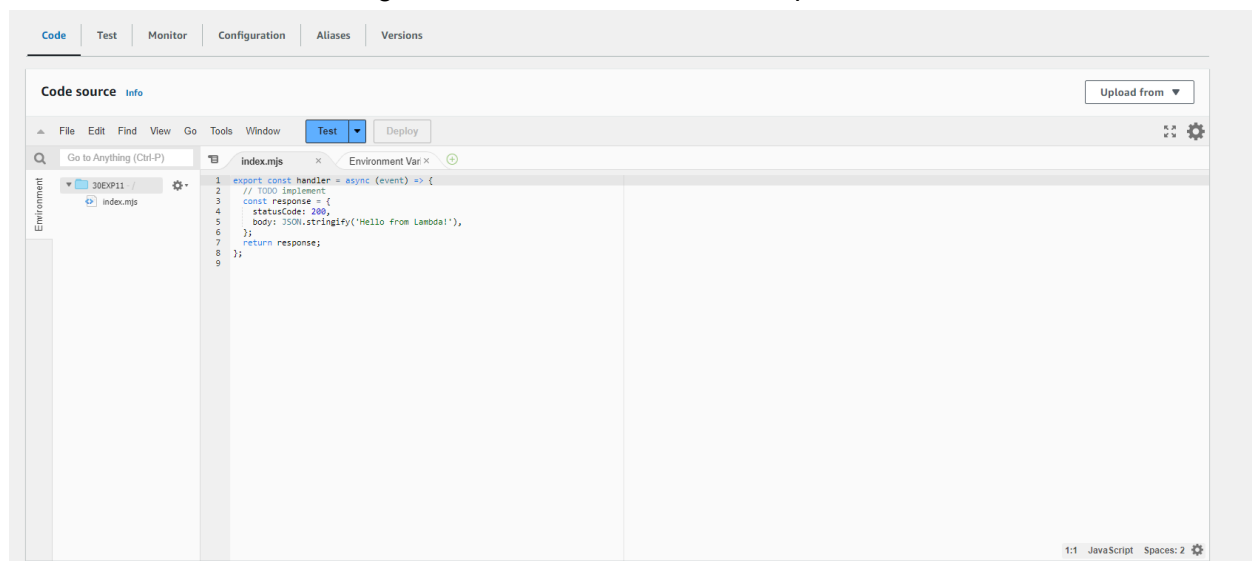
Functions (1) Last fetched 3 minutes ago Actions Create function

<input type="checkbox"/>	Function name	Description	Package type	Runtime	Last modified
<input type="checkbox"/>	<a href="#">30EXP11</a>	-	Zip	Node.js 20.x	9 seconds ago

This is the dashboard of our lambda function.

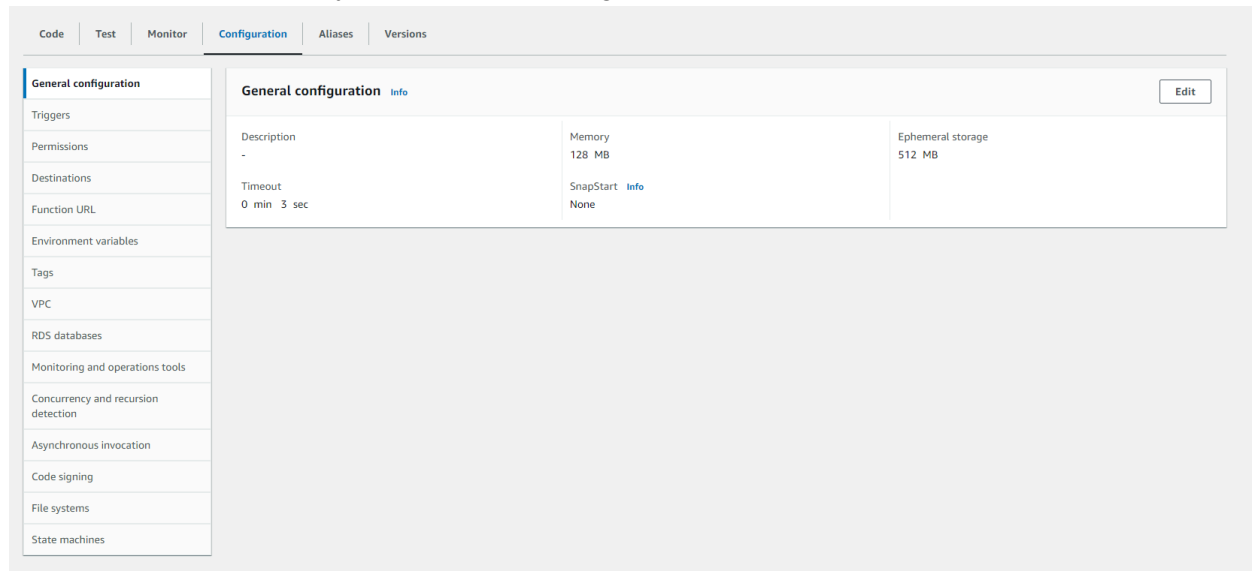


This function has the following default code, which is used to print “Hello form Lambda!”.



## Set up configurations and test events

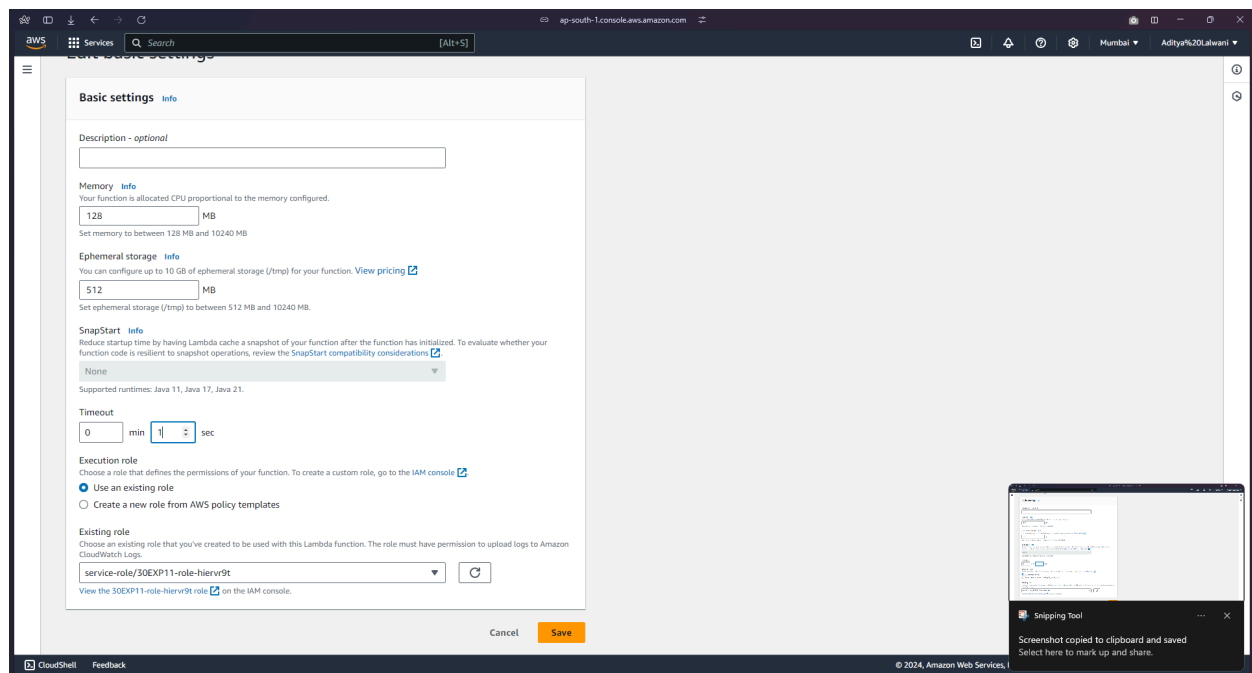
Just above the test code, you would find Configuration, click on it. Then click on Edit.



The screenshot shows the AWS Lambda console's Configuration tab. On the left is a sidebar with various configuration categories: General configuration, Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, VPC, RDS databases, Monitoring and operations tools, Concurrency and recursion detection, Asynchronous invocation, Code signing, File systems, and State machines. The main area displays the 'General configuration' settings. It includes a description field (empty), a memory setting of 128 MB, and an ephemeral storage setting of 512 MB. The timeout is currently set to 0 min 3 sec. There is an 'Edit' button in the top right corner of the configuration panel.

Configuration	Value
Description	-
Memory	128 MB
Ephemeral storage	512 MB
Timeout	0 min 3 sec
SnapStart	None

Here, change the Timeout to 1 sec. This is the time for which the function can be running before it is forcibly terminated.



This screenshot shows the 'Basic settings' dialog box for an AWS Lambda function. The 'Description' field is optional and empty. The 'Memory' is set to 128 MB. The 'Ephemeral storage' is set to 512 MB. The 'SnapStart' is set to 'None'. The 'Timeout' is now set to 0 min 1 sec. Under 'Execution role', the option 'Use an existing role' is selected, and the role 'service-role/30EXP11-role-hieruv9t' is chosen from the dropdown. At the bottom, there are 'Cancel' and 'Save' buttons. A small 'Snipping tool' window is visible in the bottom right corner, indicating a screenshot was taken.

**Basic settings**

Description - optional

Memory: 128 MB

Ephemeral storage: 512 MB

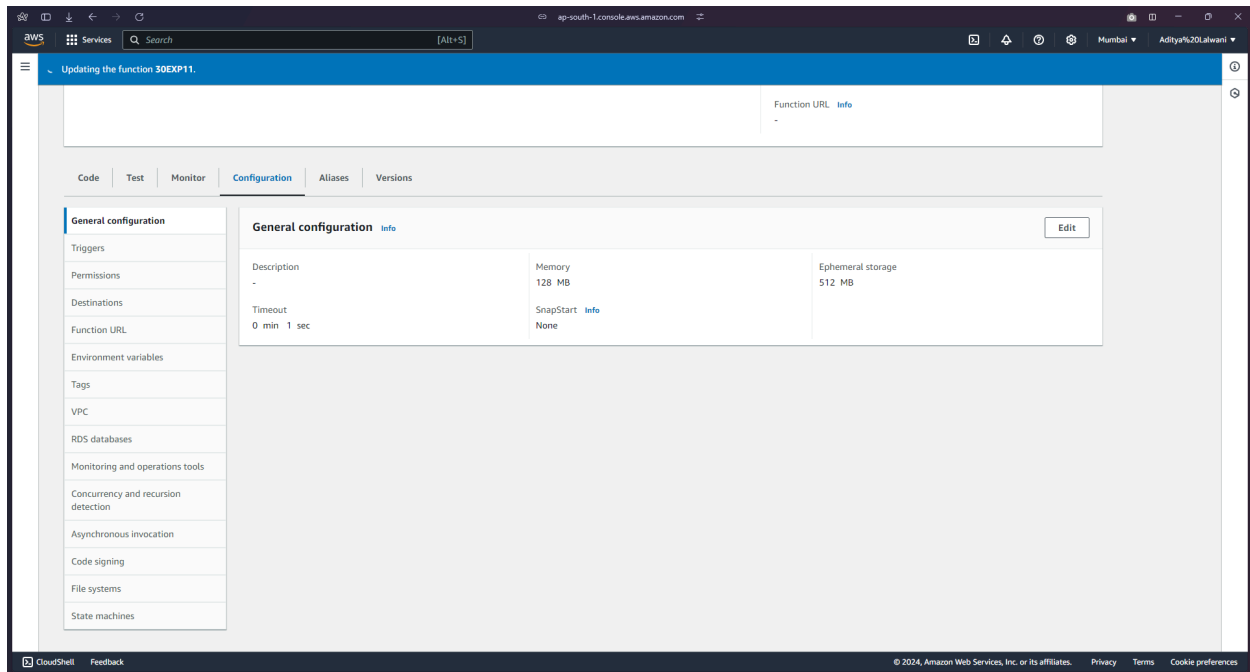
SnapStart: None

Timeout: 0 min 1 sec

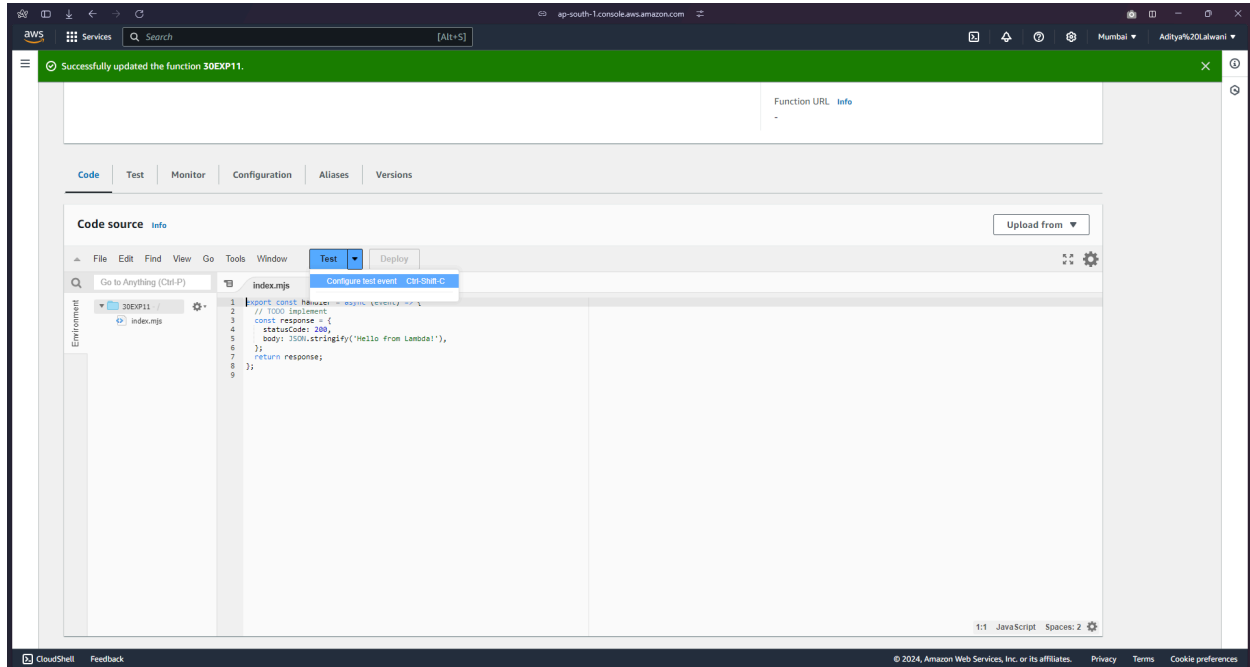
Execution role: ☒ Use an existing role

Existing role: service-role/30EXP11-role-hieruv9t

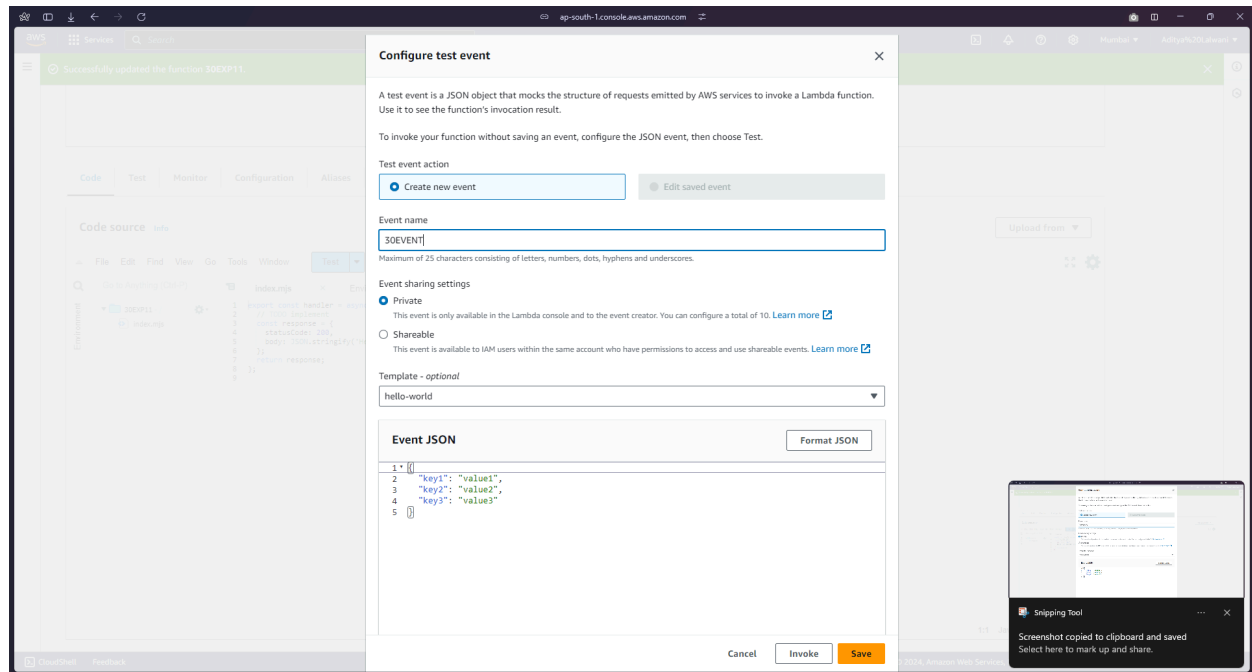
We can see the executed changes.



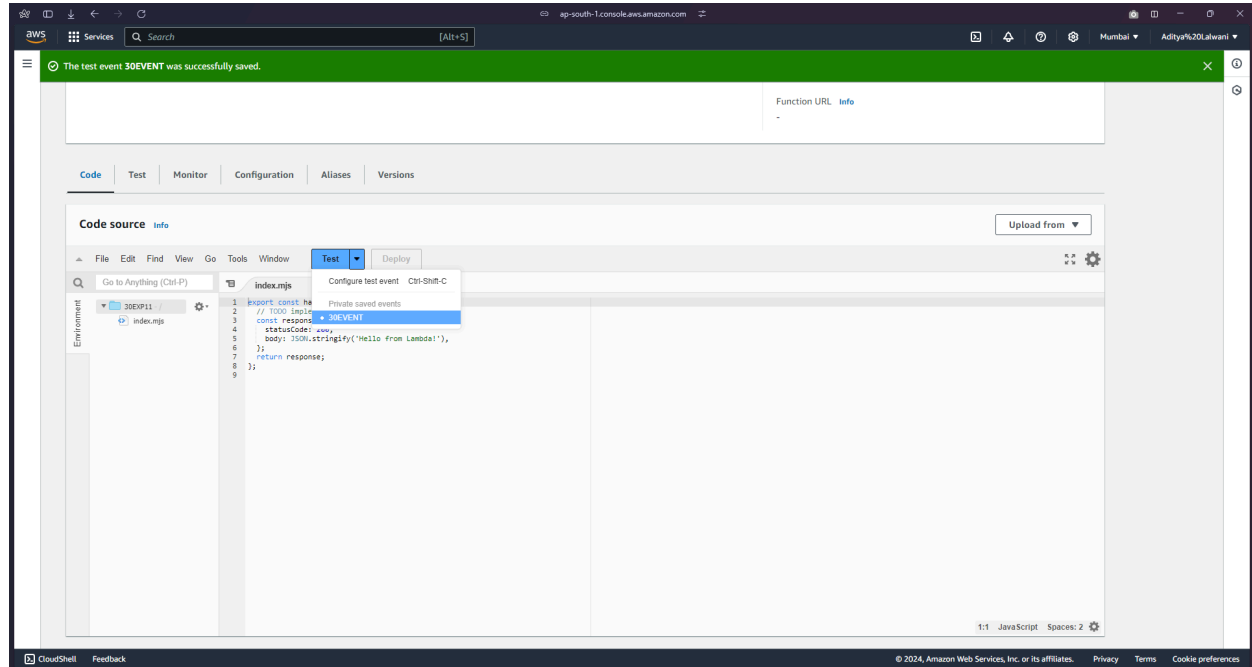
Switch back to the code tab. Click on the dropdown arrow near test. Then select configure test event.



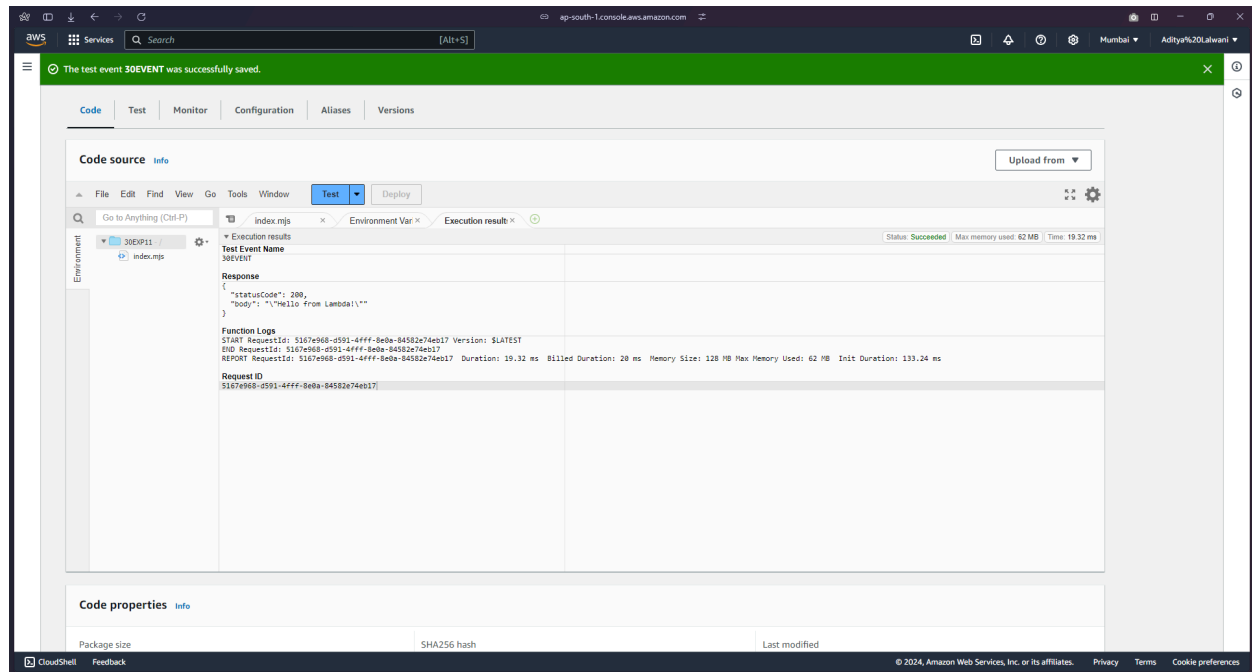
Here, create a new event, keep the other options default and save the event.



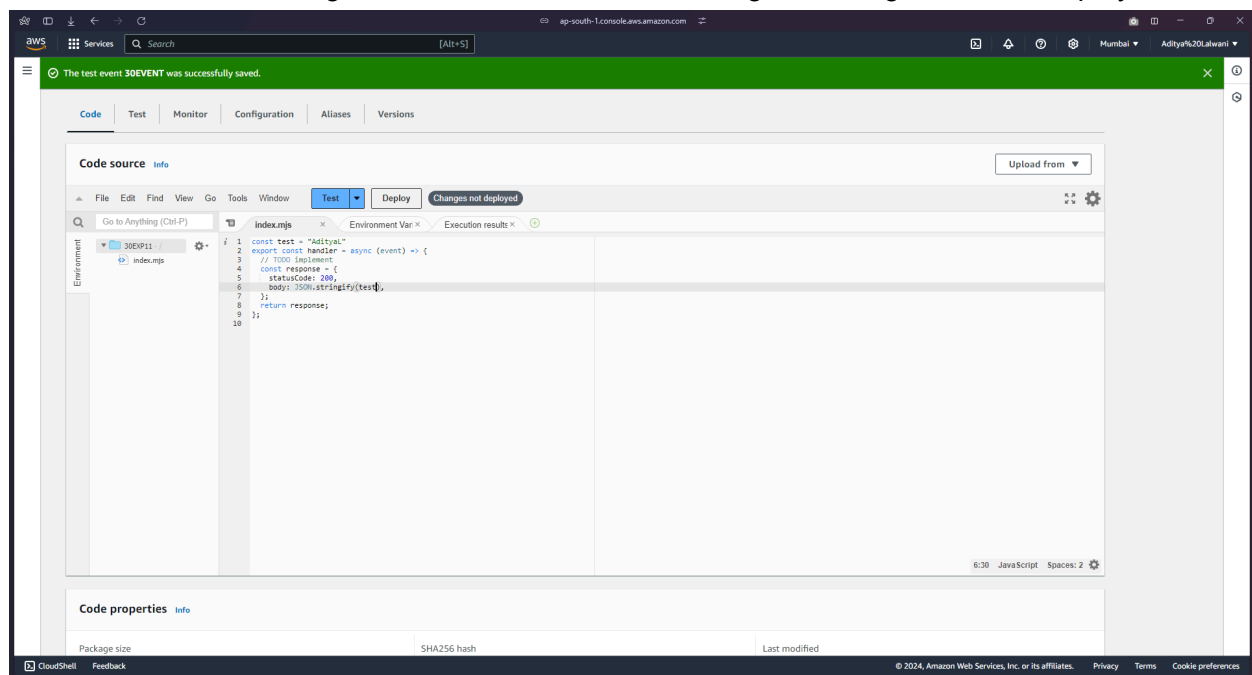
Now, again click on the dropdown. This time, select the event you have created. Then, click on TEST.



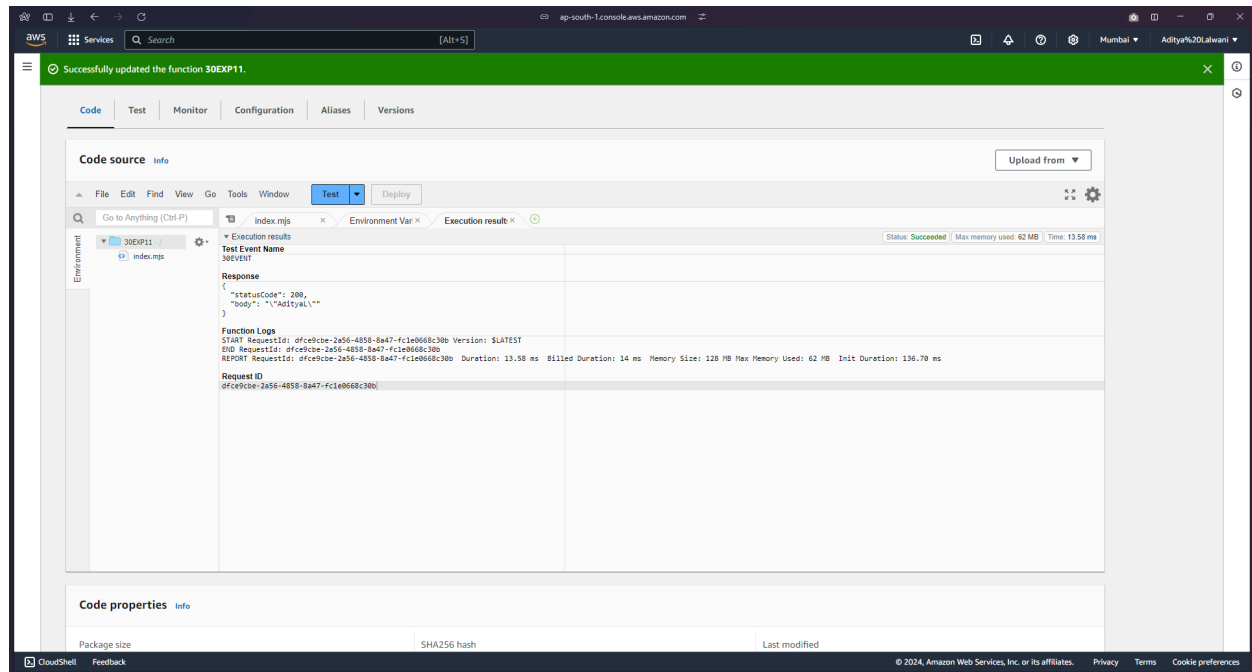
We can see the expected output for the sample code.



For a test, declare a string and call it in line 6. After making the changes click on deploy.



Run the test. We can see that the string we declared has come in the output.



In this experiment, we successfully used the AWS Lambda service by creating and configuring Lambda functions using Node.js. We learned how to set up a Lambda function, modify its configuration (such as adjusting the timeout), and test the function with custom events. Through this process, we observed how Lambda handles executions, including managing timeouts and returning expected outputs based on the code changes.