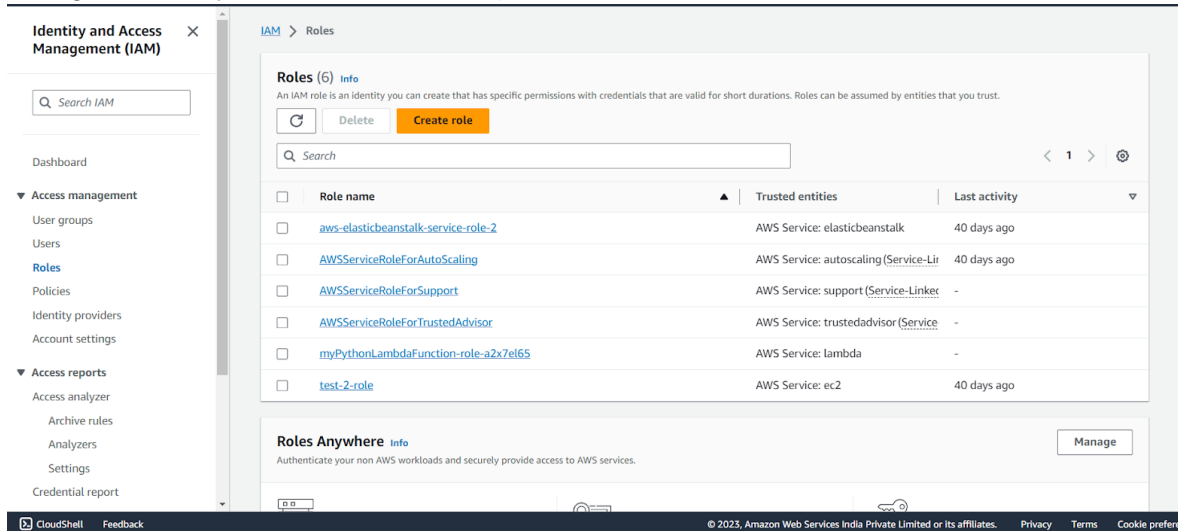


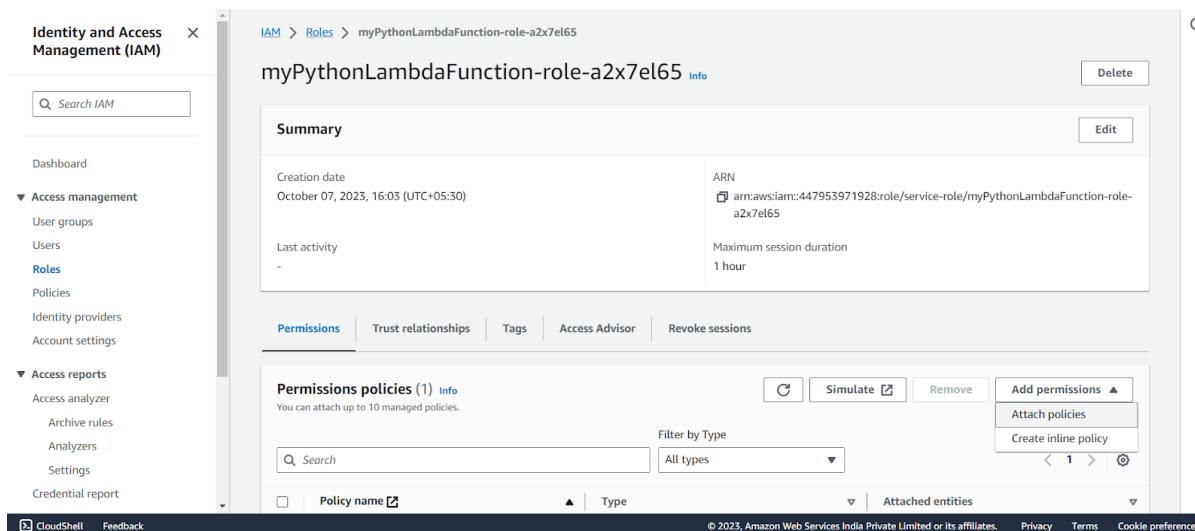
Experiment No 12

Vedant Sanap
D15A 48
Batch C

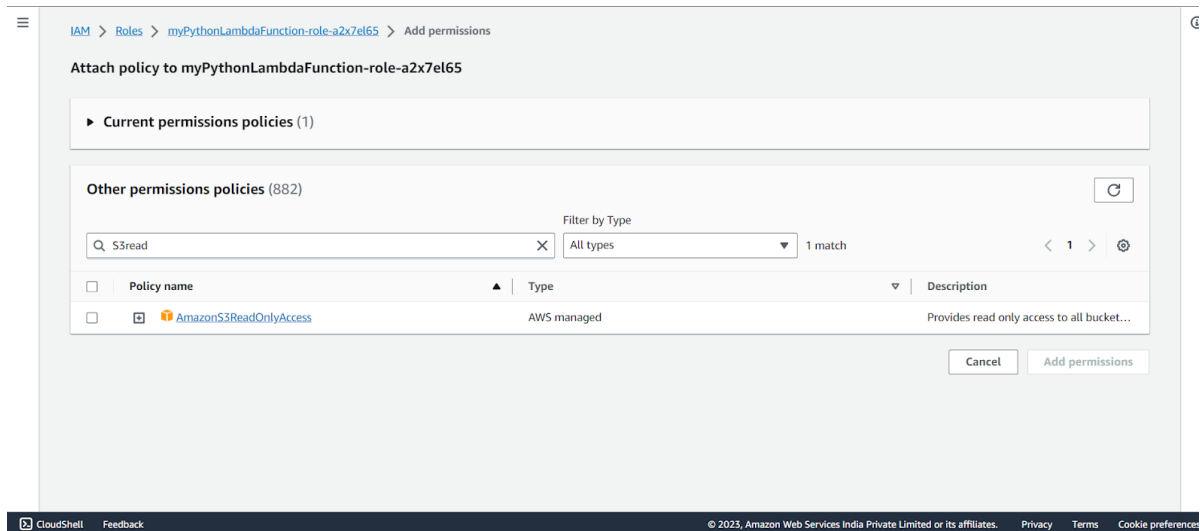
Step 1: Open up the IAM Console and under Roles, choose the Role we previously created for the Python Lambda Function (You can find your role name configuration of your Lambda function).



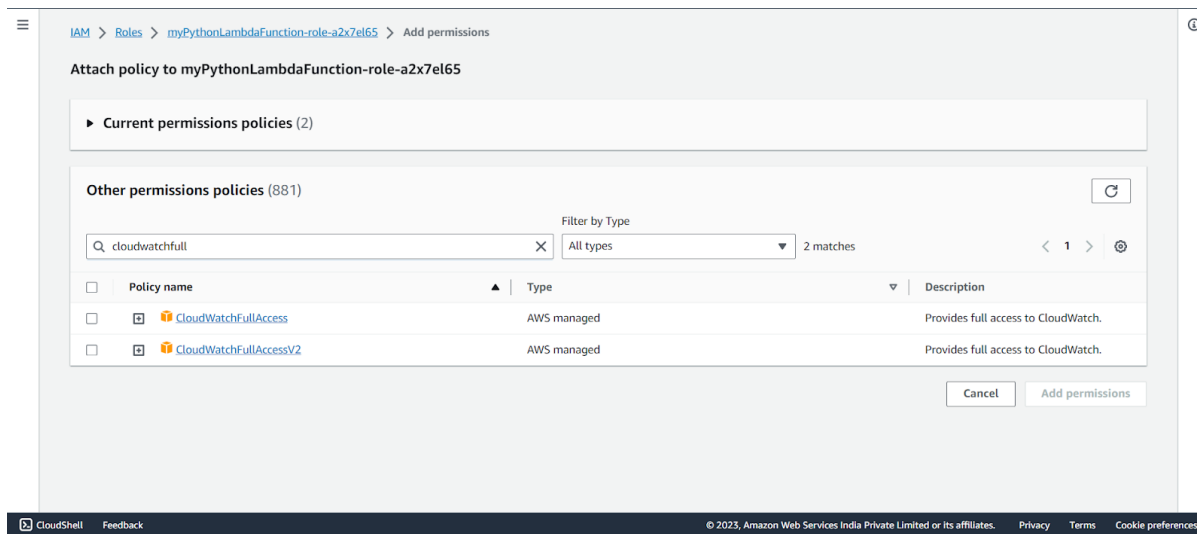
Step 2: Under Attach Policies, add S3-ReadOnly and CloudWatchFull permissions to this role.



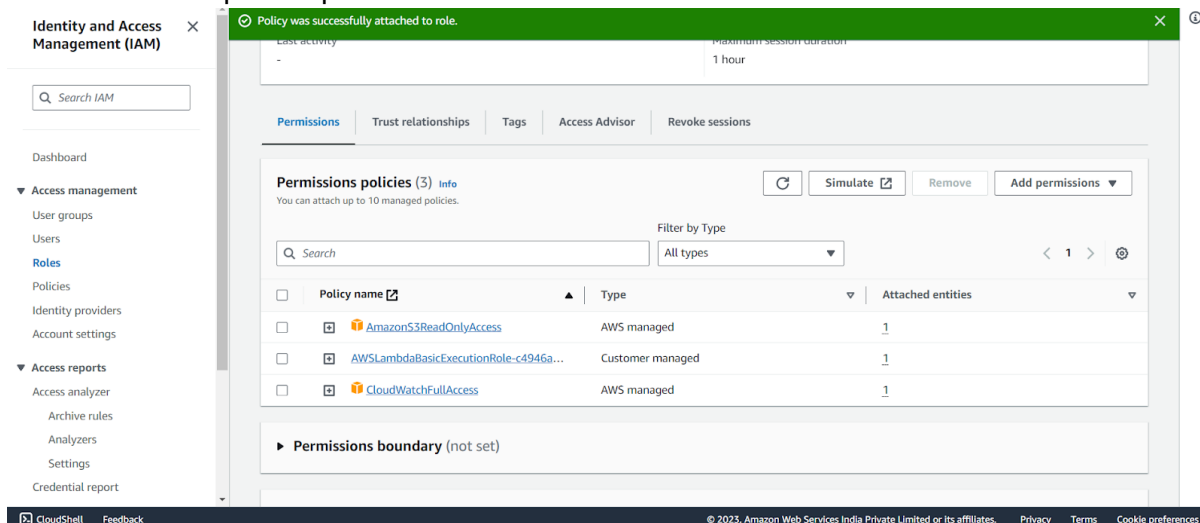
S3-ReadOnly



CloudWatchFull



After successful attachment of policy you will see something like this you will be able to see the updated policies.



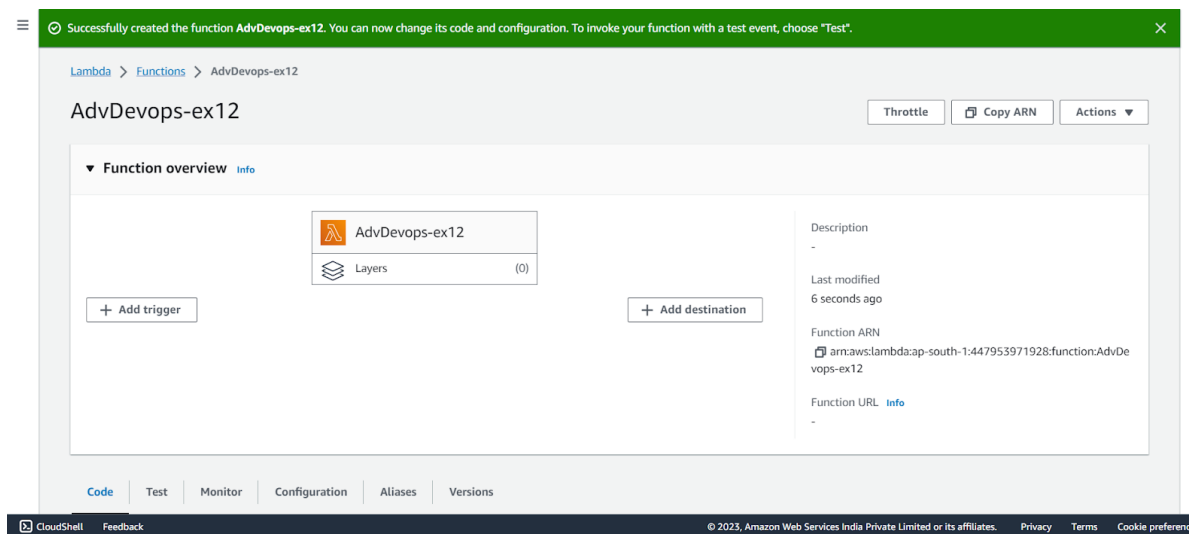
Step 3: Open up AWS Lambda and create a new Python function.

The screenshot shows the 'Create function' page in the AWS Lambda console. At the top, there are three tabs: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. Below these, the 'Basic information' section is visible. It includes a 'Function name' field with the value 'AdvDevOps-ex12', a 'Runtime' dropdown set to 'Python 3.11', and an 'Architecture' dropdown set to 'x86_64'. The 'Permissions' section is partially visible at the bottom.

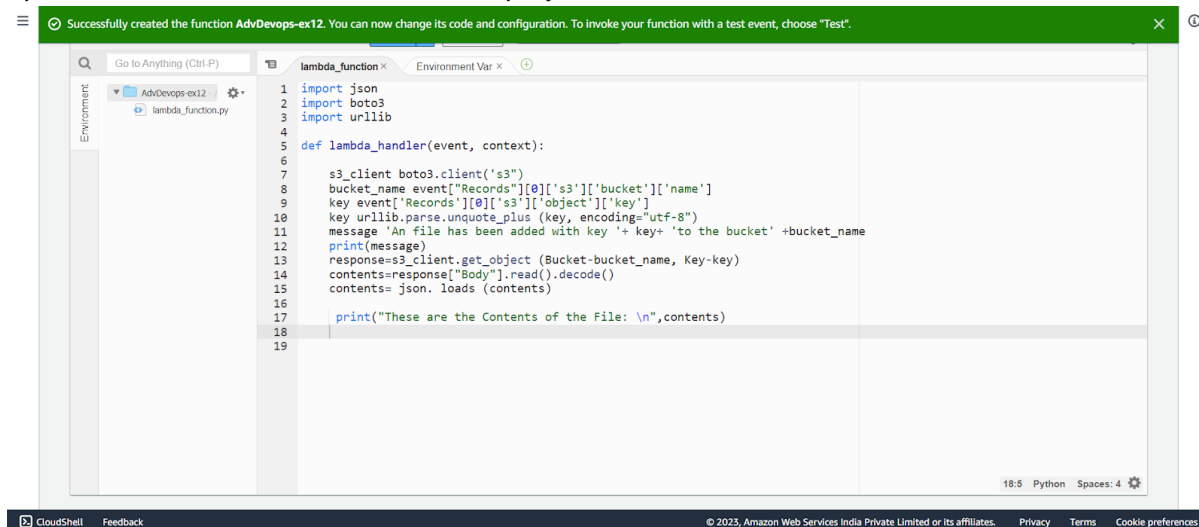
Under Execution Role, choose the existing role, then select the one which was previously created and to which we just added permissions.

The screenshot shows the 'Permissions' section of the AWS Lambda console. It features a 'Change default execution role' section with three radio buttons: 'Create a new role with basic Lambda permissions', 'Use an existing role' (selected), and 'Create a new role from AWS policy templates'. Below this, there is a dropdown menu for 'Existing role' with the value 'service-role/myPythonLambdaFunction-role-a2x7el65'. At the bottom right, there are 'Cancel' and 'Create function' buttons.

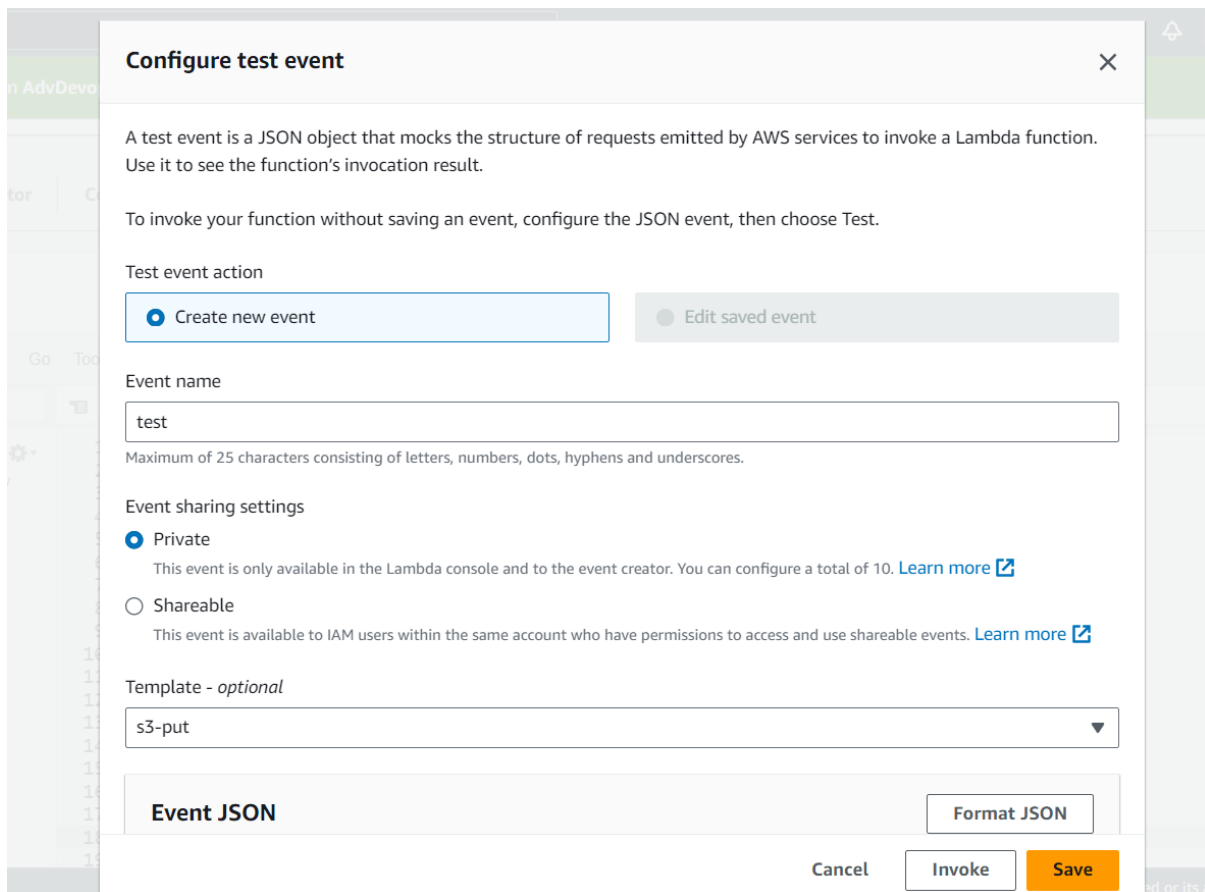
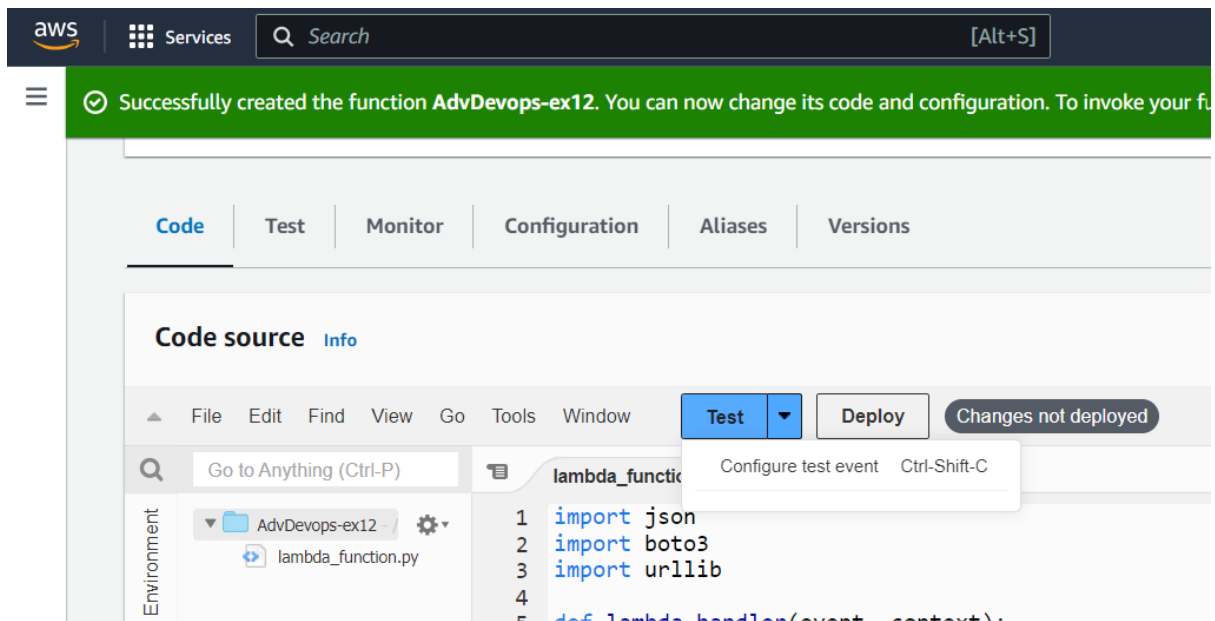
Step 4: The function is up and running.



Step 5: Make the following changes to the function and click on the deploy button. This code basically logs a message and logs the contents of a JSON file which is uploaded to an S3 Bucket and then deploy the code.

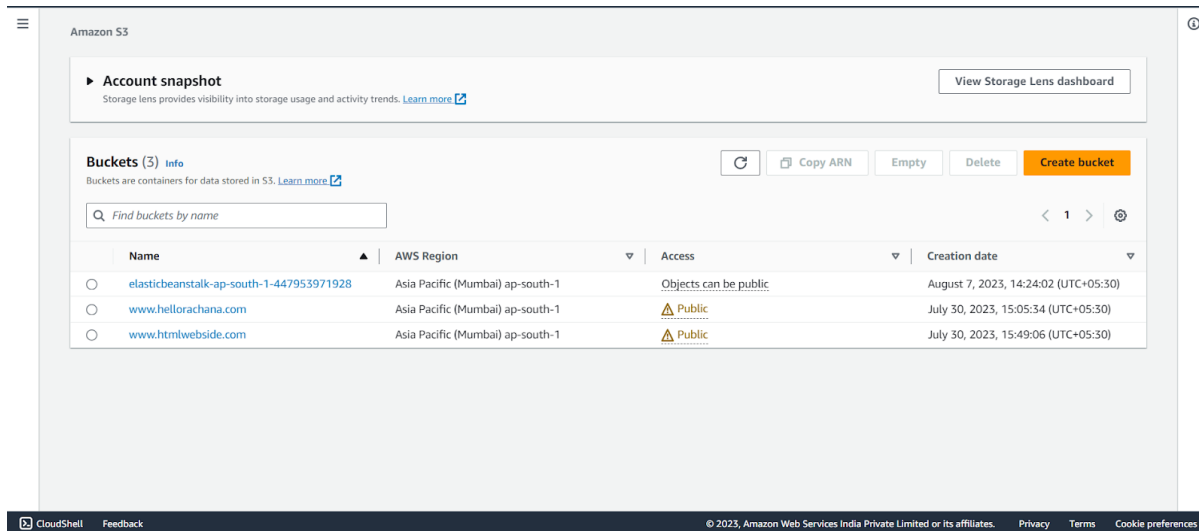


Step 6: Click on Test and choose the 'S3 Put' Template.

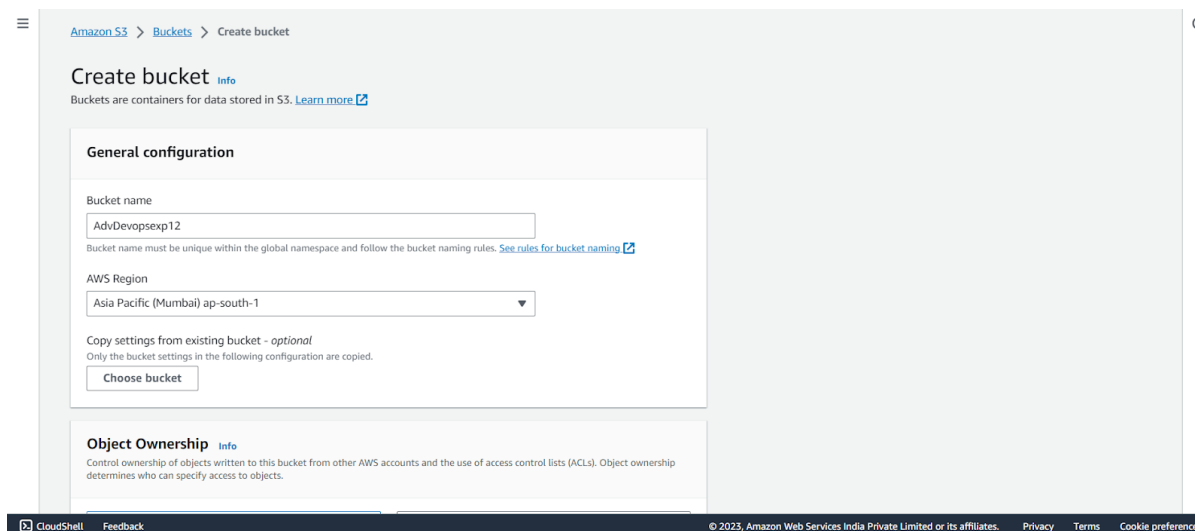


And Save it.

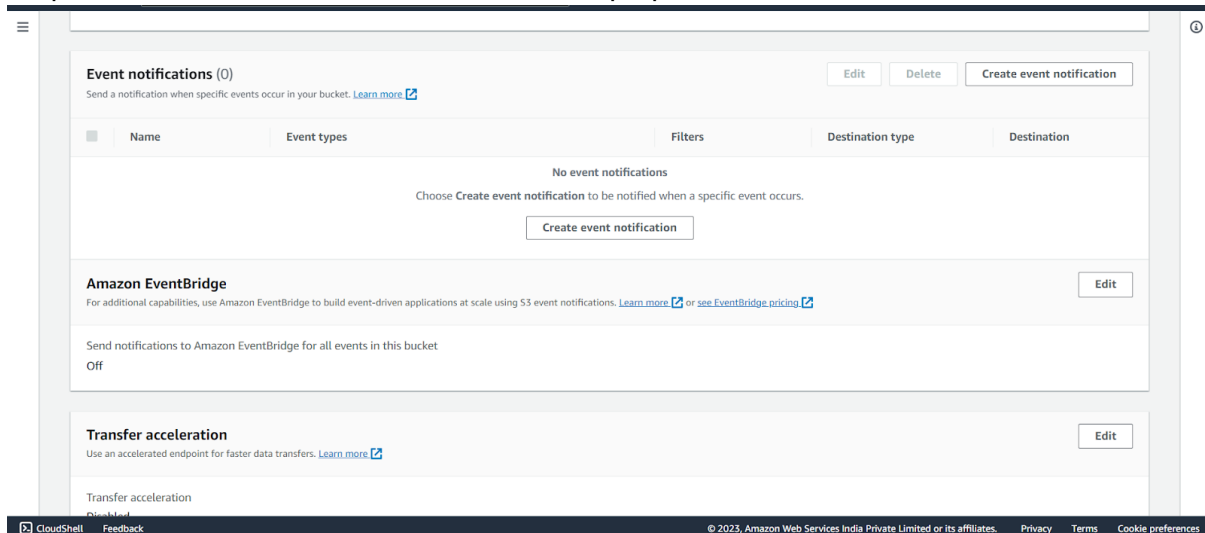
Step 7: Open up the S3 Console and create a new bucket.



Step 8: With all general settings, create the bucket in the same region as the function.



Step 9: Click on the created bucket and under properties, look for events.



Click on Create Event Notification.

Step 10: Mention an event name and check Put under event types.

aws

Services

Search

[Alt+S]

General configuration

Event name

Event name can contain up to 255 characters.

Prefix - optional

Limit the notifications to objects with key starting with specified characters.

Suffix - optional

Limit the notifications to objects with key ending with specified characters.

Event types

Specify at least one event for which you want to receive notifications. For each group, you can choose an event type for all events, or you can choose one or more individual events.

Object creation

☐ All object create events
s3:ObjectCreated:*

☒ Put
s3:ObjectCreated:Put

☐ Post
s3:ObjectCreated:Post

CloudShell

Feedback

© 2023, Amazon Web Services India Pr

Choose Lambda function as destination and choose your lambda function and save the changes.

aws

Services

Search

[Alt+S]

Destination

Before Amazon S3 can publish messages to a destination, you must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function. [Learn more](#)

Destination

Choose a destination to publish the event. [Learn more](#)

☒ Lambda function
Run a Lambda function script based on S3 events.

☐ SNS topic
Fanout messages to systems for parallel processing or directly to people.

☐ SQS queue
Send notifications to an SQS queue to be read by a server.

Specify Lambda function

☒ Choose from your Lambda functions

☐ Enter Lambda function ARN

Lambda function

Cancel

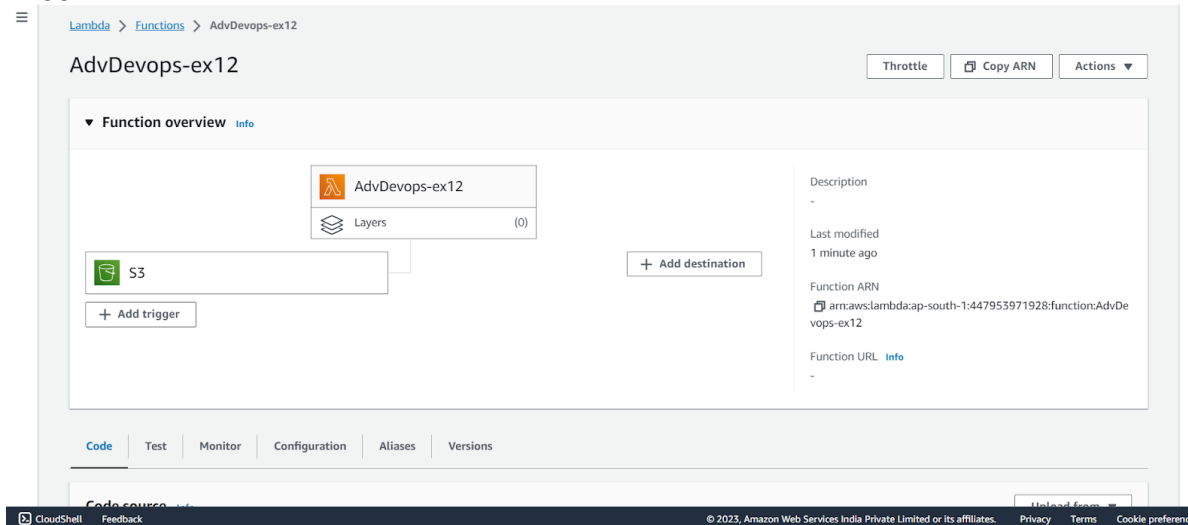
Save changes

CloudShell

Feedback

© 2023, Amazon Web Serv

Step 11: Refresh the Lambda function console and you should be able to see an S3 Trigger in the overview.



Step 12: Now, create a dummy JSON file locally.

```
{ } dummy.json X
{ } dummy.json > ...
1  {
2    "firstname" : "Shashwat",
3    "lastname"  : "Tripathi",
4    "gender"   : "Male",
5    "age": 19
6  }
```

Step 13: Go back to your S3 Bucket and click on Add Files to upload a new file.

Step 14: Select the dummy data file from your computer and click Upload.

aws Services Search [Alt+S]

Amazon S3 > Buckets > advopssexp12 > Upload

Upload [Info](#)

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (1 Total, 89.0 B) Remove Add files Add folder

All files and folders in this table will be uploaded.

Find by name

<input type="checkbox"/>	Name	Folder	Type	Size
<input type="checkbox"/>	dummy.json	-	application/json	89.0 B

Destination

Destination

s3://advopssexp12

CloudShell Feedback © 2023, Amazon Web Services India Private Limited or

Step 15: After this make the necessary changes in the Test configuration file which we created it previously by replacing the Bucket Name and the ARN of Bucket.

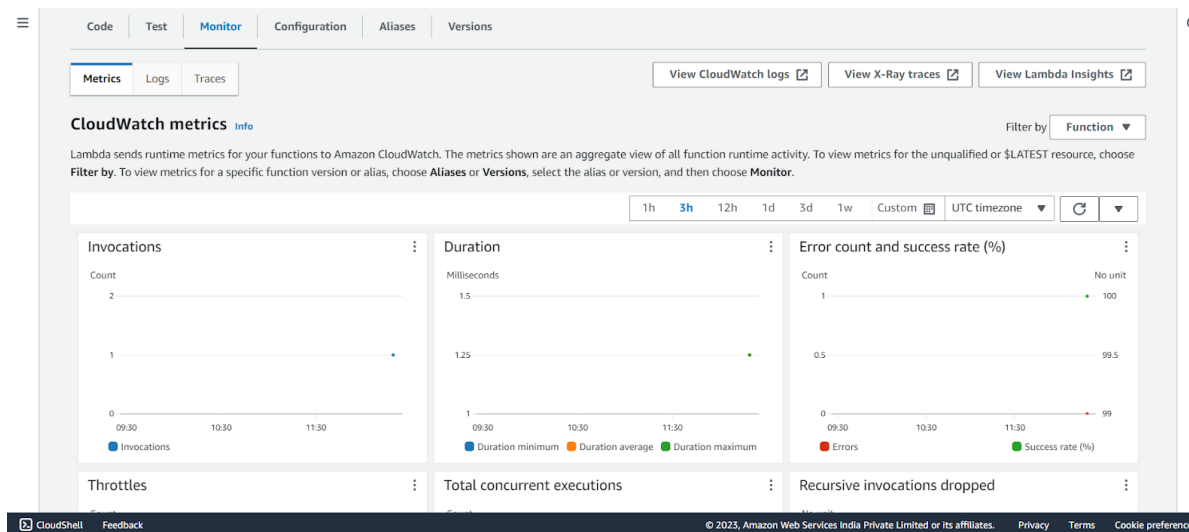
Event JSON Format JSON

```

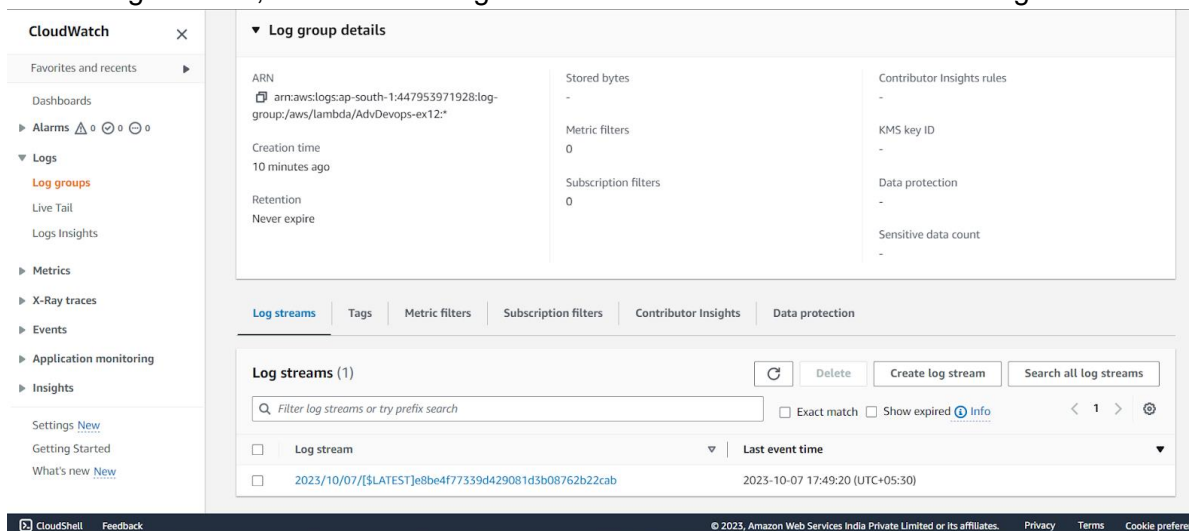
10      "principalId": "EXAMPLE"
11    },
12    "requestParameters": {
13      "sourceIPAddress": "127.0.0.1"
14    },
15    "responseElements": {
16      "x-amz-request-id": "EXAMPLE123456789",
17      "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
18    },
19    "s3": {
20      "s3SchemaVersion": "1.0",
21      "configurationId": "testConfigRule",
22      "bucket": {
23        "name": "advopssexp12",
24        "ownerIdentity": {
25          "principalId": "EXAMPLE"
26        },
27        "arn": "arn:aws:s3:::advopssexp12"
28      },
29      "object": {
30        "key": "test%2Fkey",
31        "size": 1024,
32        "eTag": "0123456789abcdef0123456789abcdef",
33        "sequencer": "0A1B2C3D4E5F678901"
34      }
35    }
36  }
37 }
38 }

```

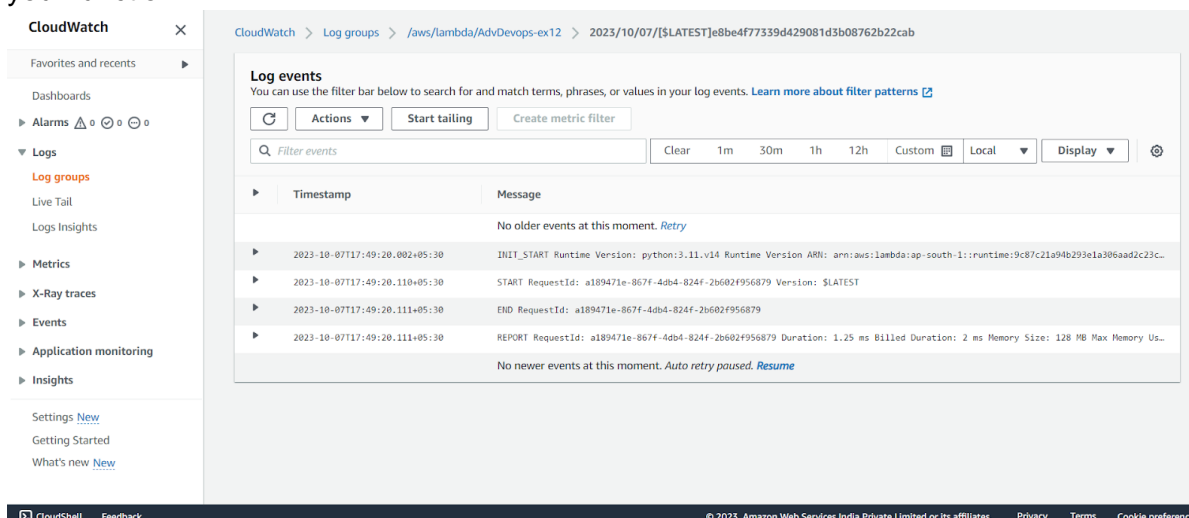
Step 16: Go back to your Lambda function , Refresh it and check the Monitor tab.



Under Log streams, click on View logs in Cloudwatch to check the Function logs.



Step 17: Click on this log Stream that was created to view what was logged by your function.



Conclusion: Thus, we have created a Lambda function which logs “An Image has been added” once you add an object to a specific bucket in S3.