

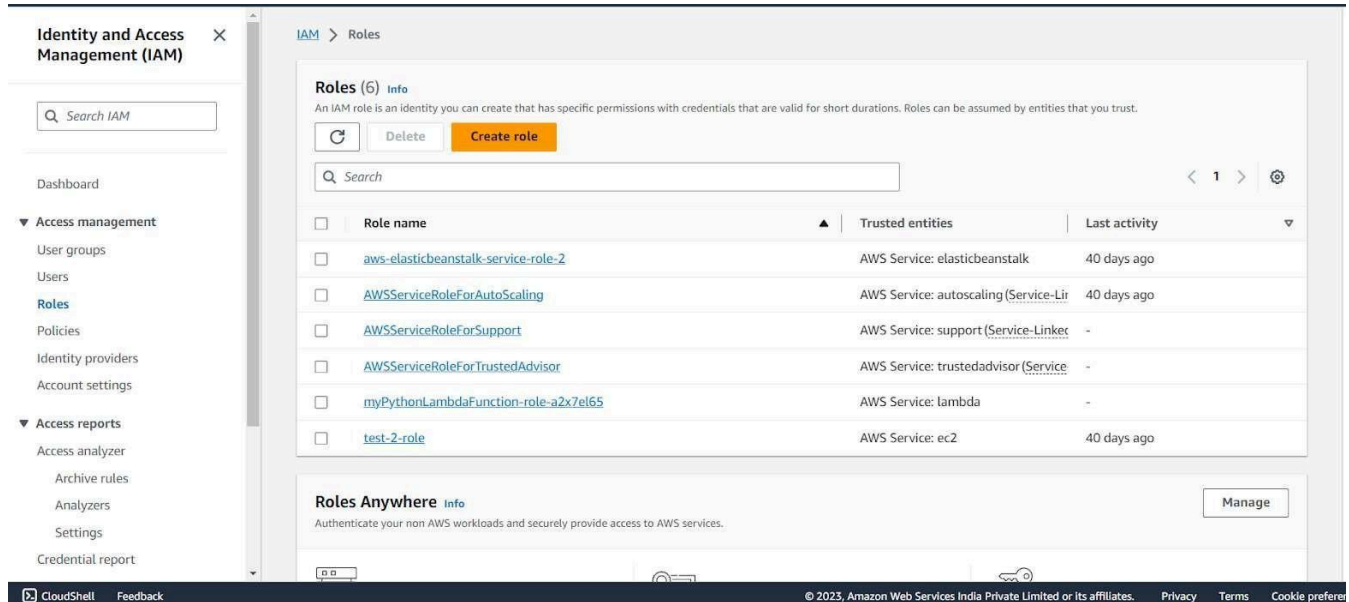
## Adv. DevOps Exp. 12

Name-Aryan Anil Patankar

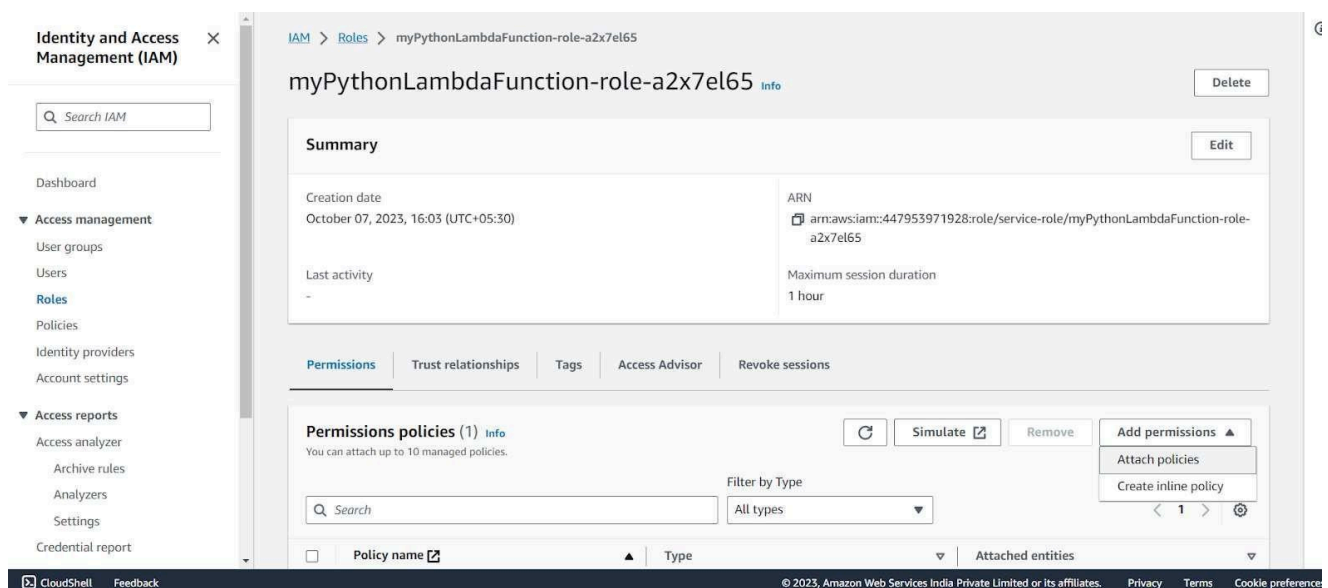
Class-D15A

Roll No-34

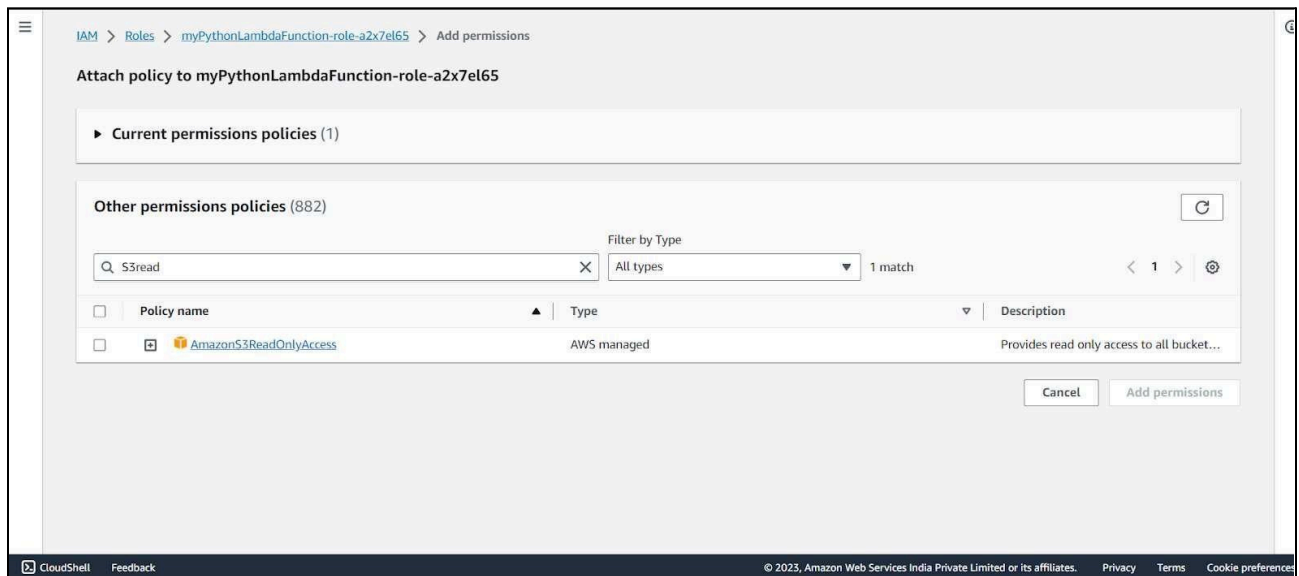
Step 1: Open up the IAM Console and under Roles, choose the Role we previously created for the Python Lambda Function (You can find your role name configuration of your Lambda function).



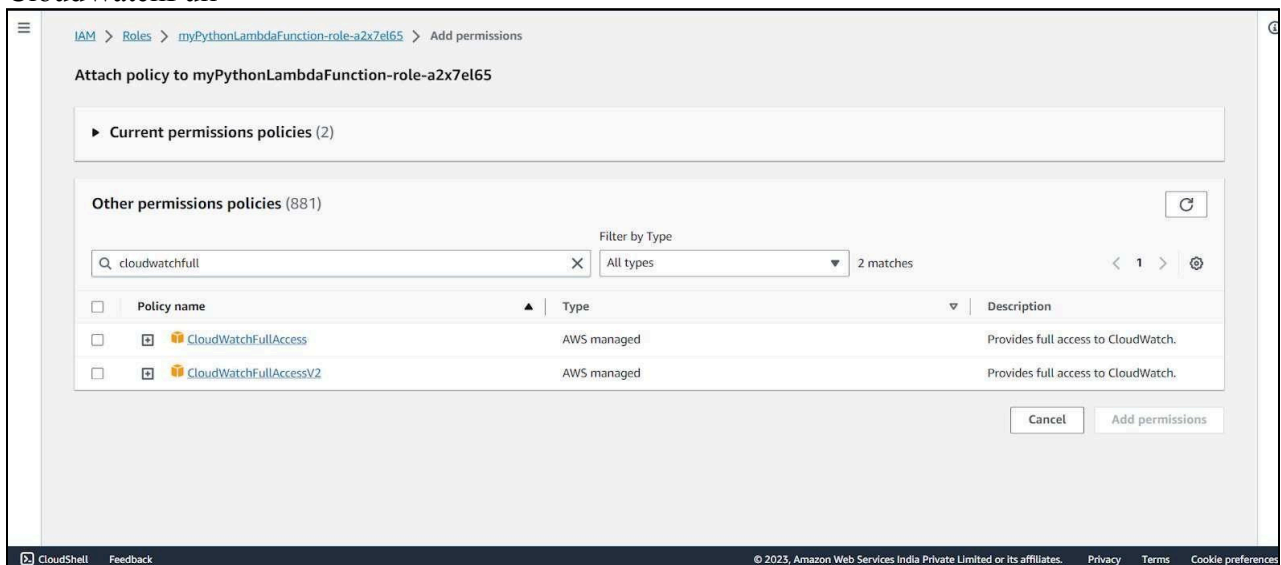
Step 2: Under Attach Policies, add S3-ReadOnly and CloudWatchFull permissions to this role.



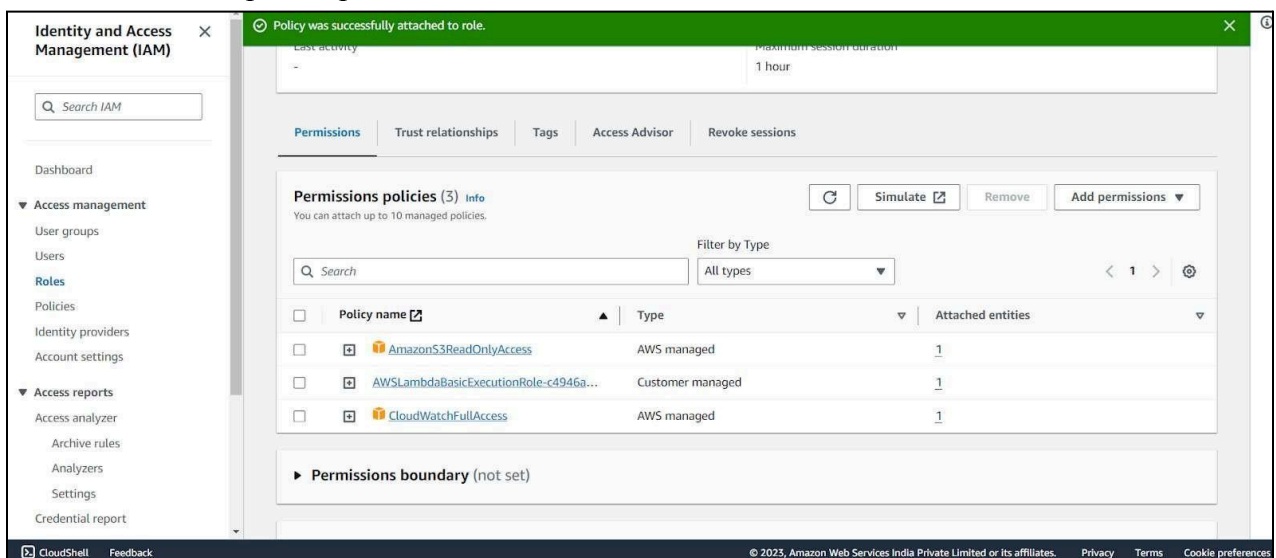
S3-ReadOnly



## CloudWatchFull



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



[Lambda](#) > [Functions](#) > Create function

## Create function Info

AWS Serverless Application Repository applications have moved to [Create application](#).

☒ **Author from scratch**  
Start with a simple Hello World example.

☐ **Use a blueprint**  
Build a Lambda application from sample code and configuration presets for common use cases.

☐ **Container image**  
Select a container image to deploy for your function.

---

### 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.

[CloudShell](#) [Feedback](#) © 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

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

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

**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](#).

☐ Create a new role with basic Lambda permissions

☒ **Use an existing role**

☐ Create a new role from AWS policy templates

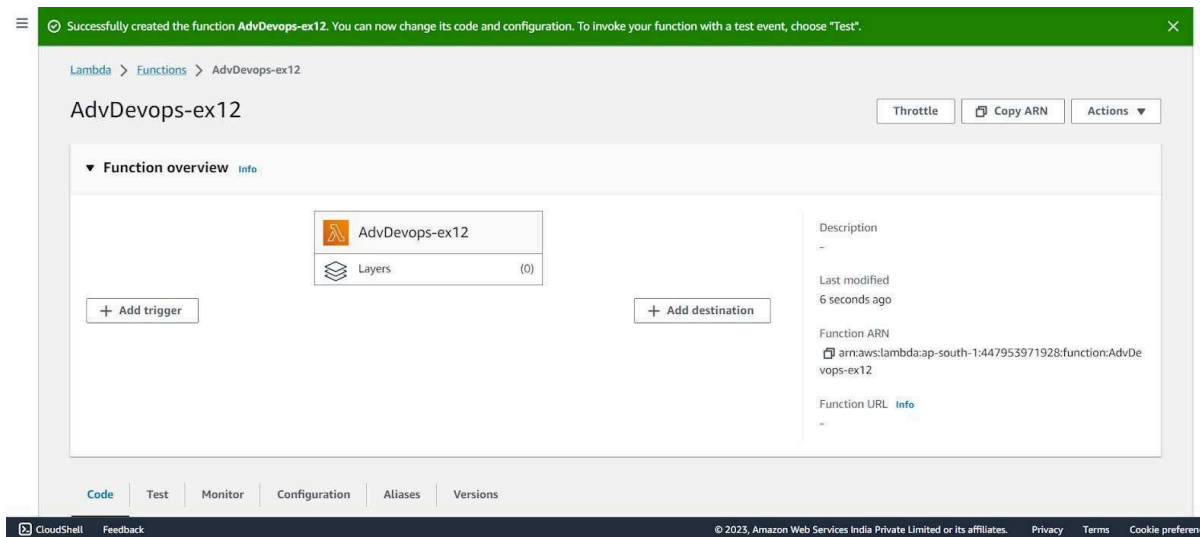
**Existing role**  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  
   
[View the myPythonLambdaFunction-role-a2x7el65 role](#) on the IAM console.

► **Advanced settings**

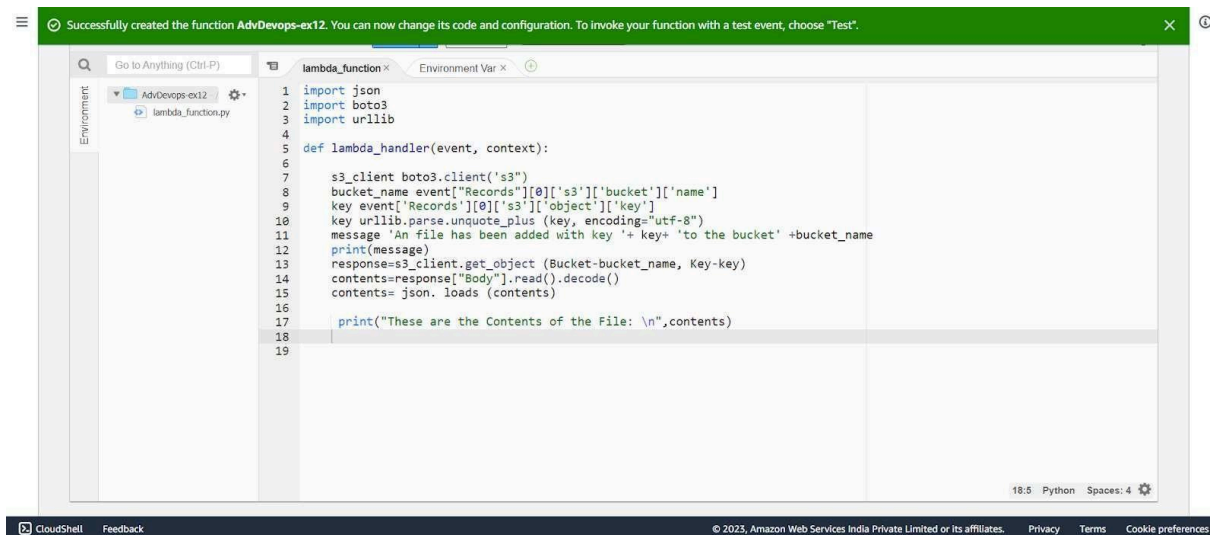
[Cancel](#) [Create function](#)

[CloudShell](#) [Feedback](#) © 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

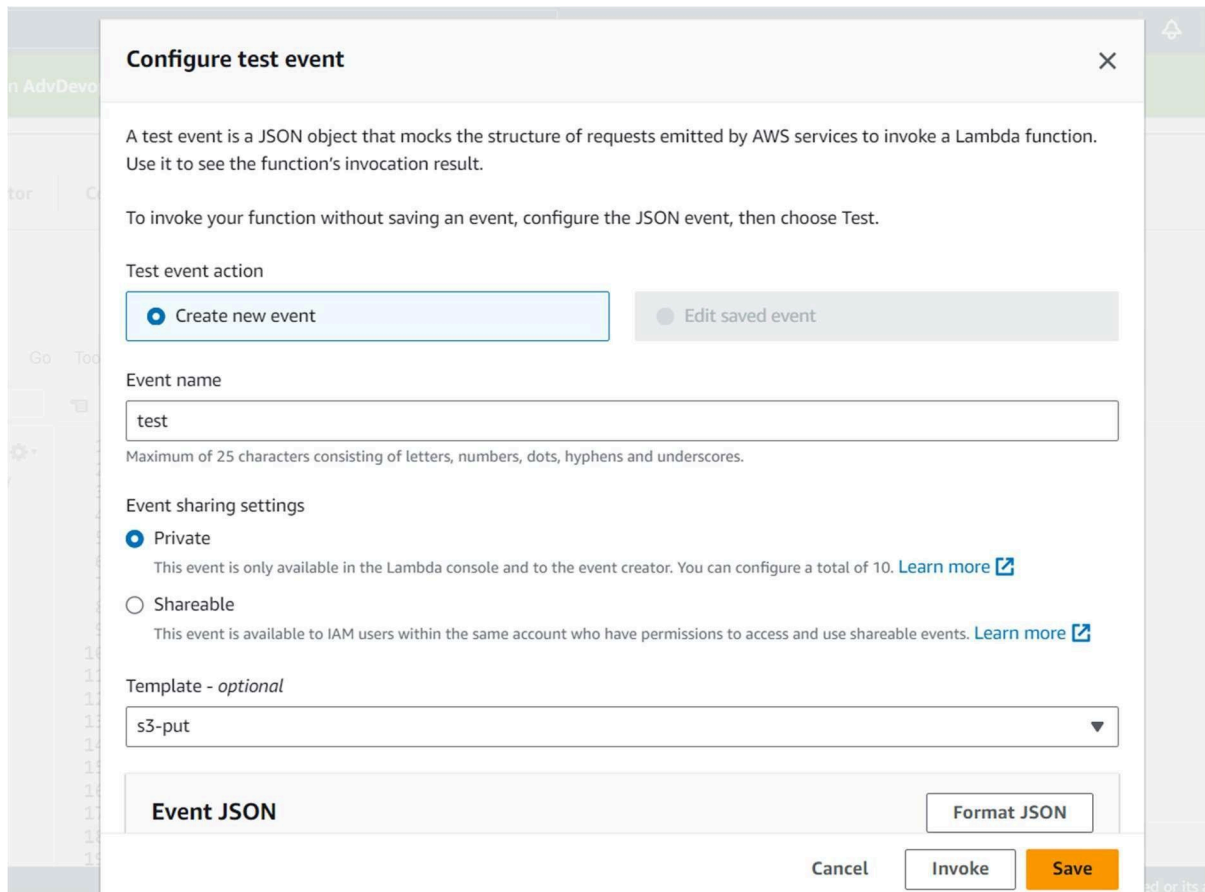
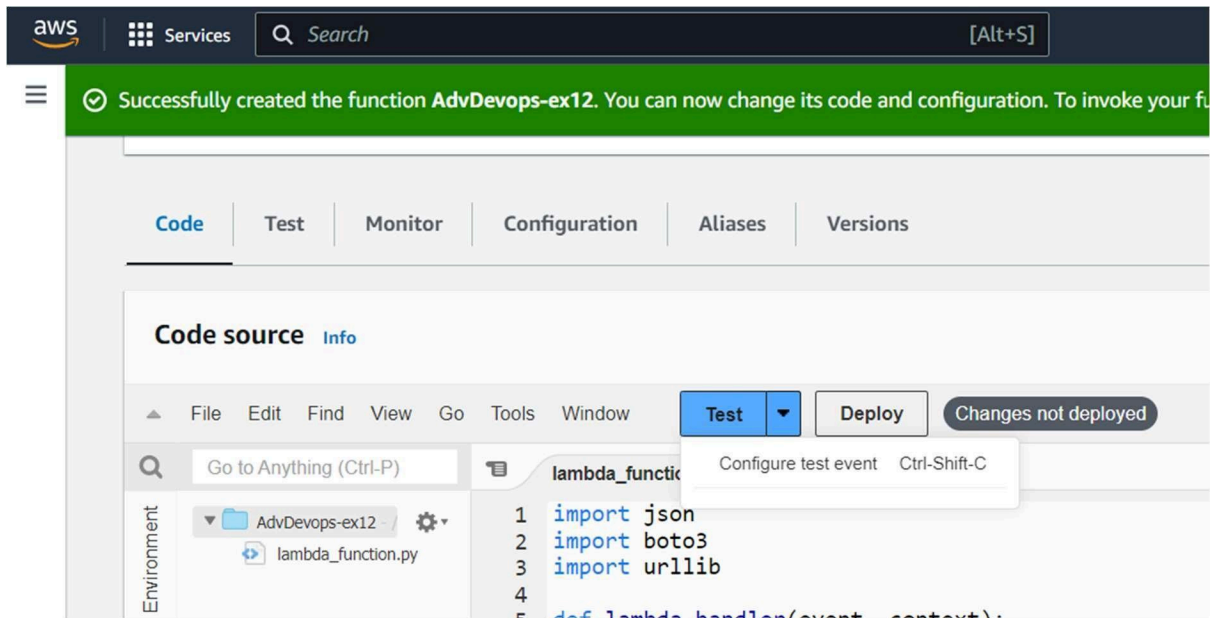
Step 4: The function is up and running.



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.

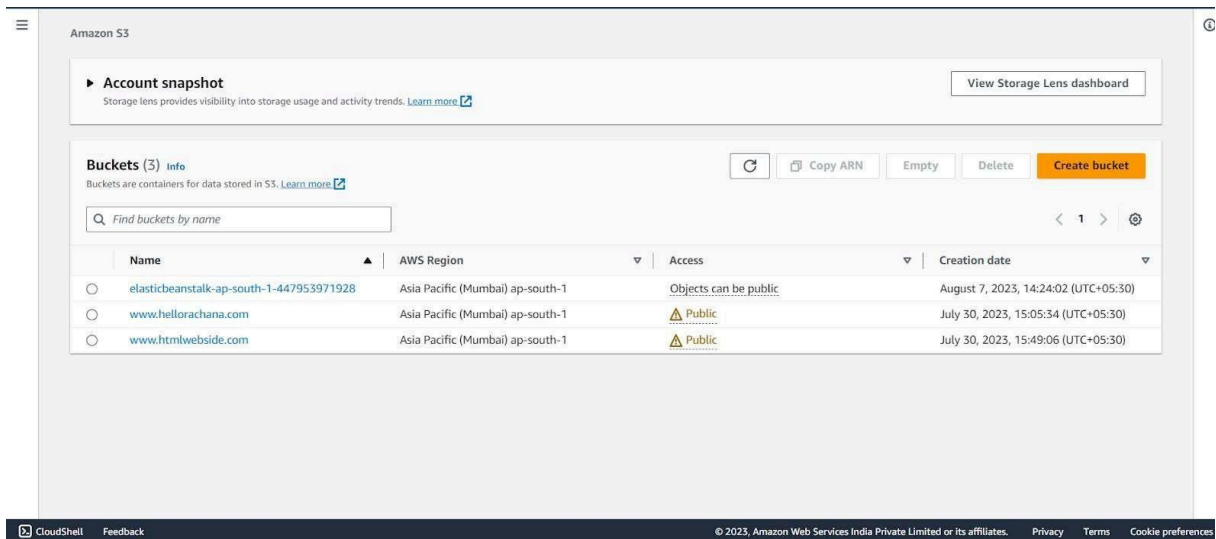


Step 6: Click on Test and choose the 'S3 Put' Template.

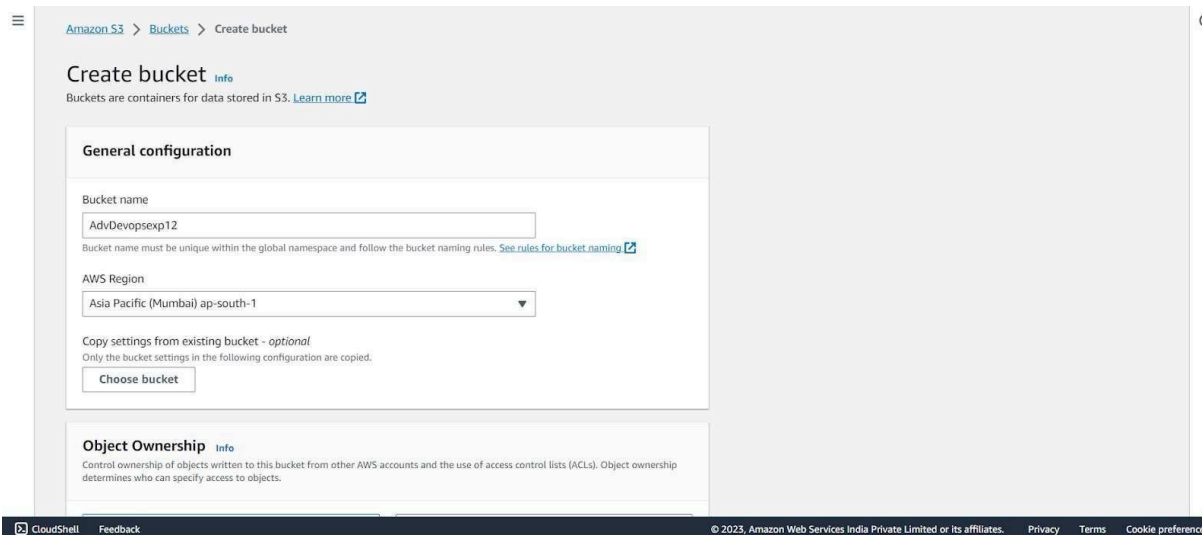


And Save it.

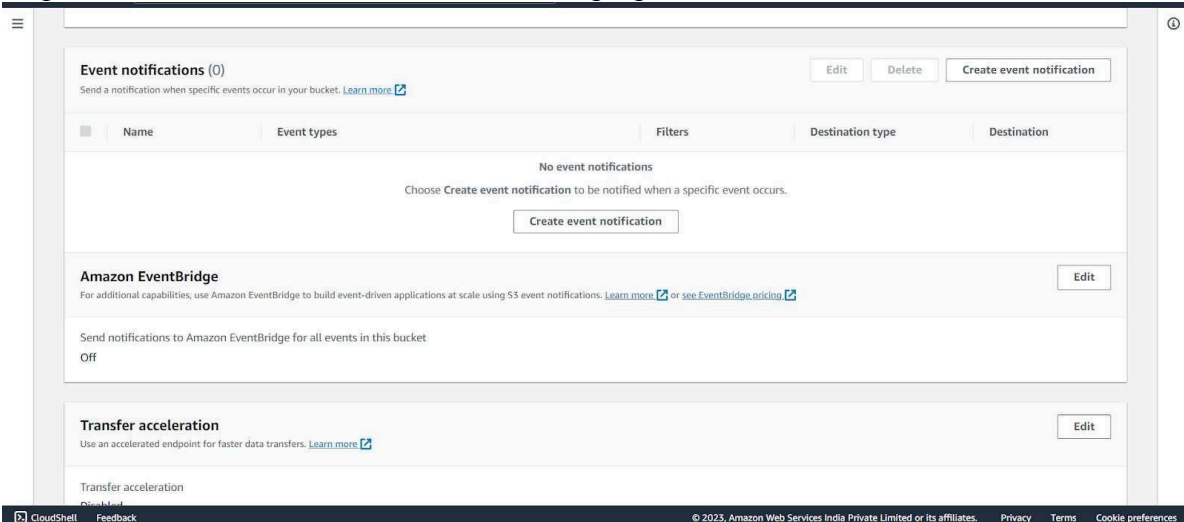
Step 7: Open up the S3 Console and create a new bucket.



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.

The screenshot shows the 'General configuration' page in the AWS Event Notifications console. The 'Event name' field is set to 'S3putrequest'. The 'Prefix - optional' field is set to 'images/'. The 'Suffix - optional' field is set to '.jpg'. Under the 'Event types' section, 'Object creation' is expanded, and the 'Put' checkbox is checked, with 's3:ObjectCreated:Put' selected. The 'Post' checkbox is unchecked. The footer shows 'CloudShell', 'Feedback', and '© 2023, Amazon Web Services India Pvt'.

aws Services Search [Alt+S]

### General configuration

Event name  
S3putrequest  
Event name can contain up to 255 characters.

Prefix - optional  
Limit the notifications to objects with key starting with specified characters.  
images/

Suffix - optional  
Limit the notifications to objects with key ending with specified characters.  
.jpg

### 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 Pvt

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

The screenshot shows the 'Destination' page in the AWS Event Notifications console. A blue information box at the top states: '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'. Under the 'Destination' section, 'Lambda function' is selected. Under 'Specify Lambda function', 'Choose from your Lambda functions' is selected. The 'Lambda function' dropdown menu shows 'AdvDevops-ex12'. At the bottom right, there are 'Cancel' and 'Save changes' buttons. The footer shows 'CloudShell', 'Feedback', and '© 2023, Amazon Web Serv'.

aws Services Search [Alt+S]

### Destination

**i** 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

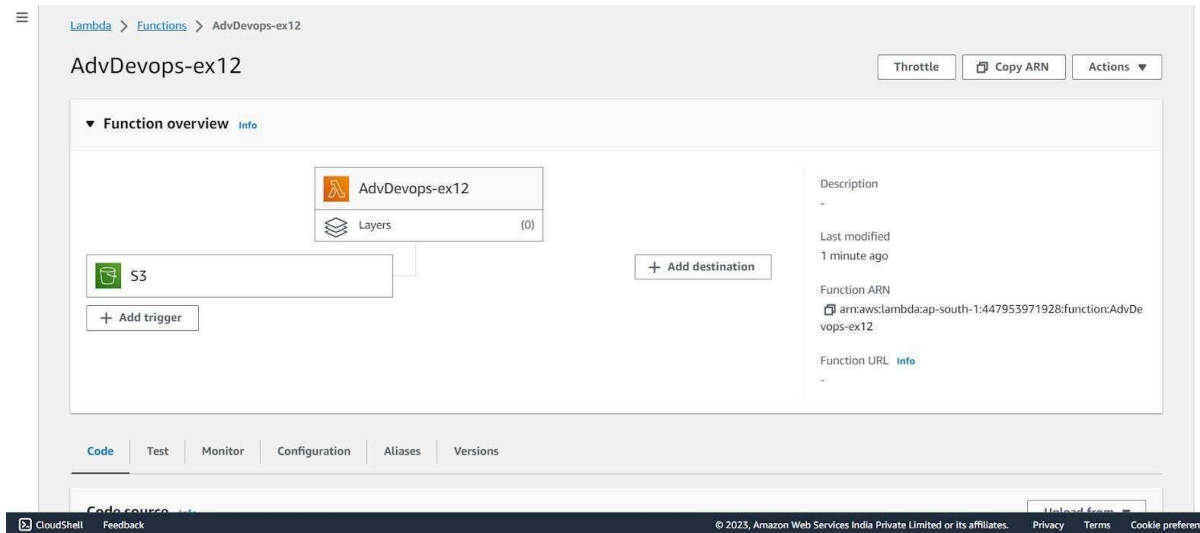
☐ Enter Lambda function ARN

Lambda function  
AdvDevops-ex12

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.



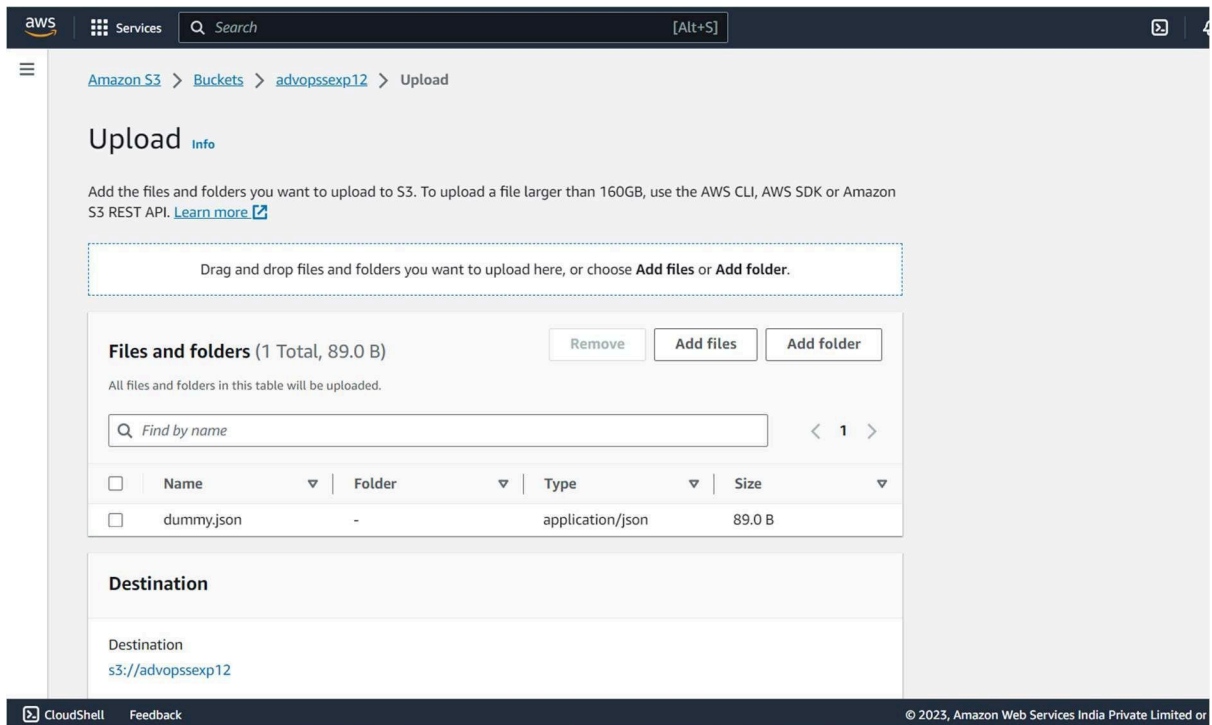
Step 12: Now, create a dummy JSON file locally.

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

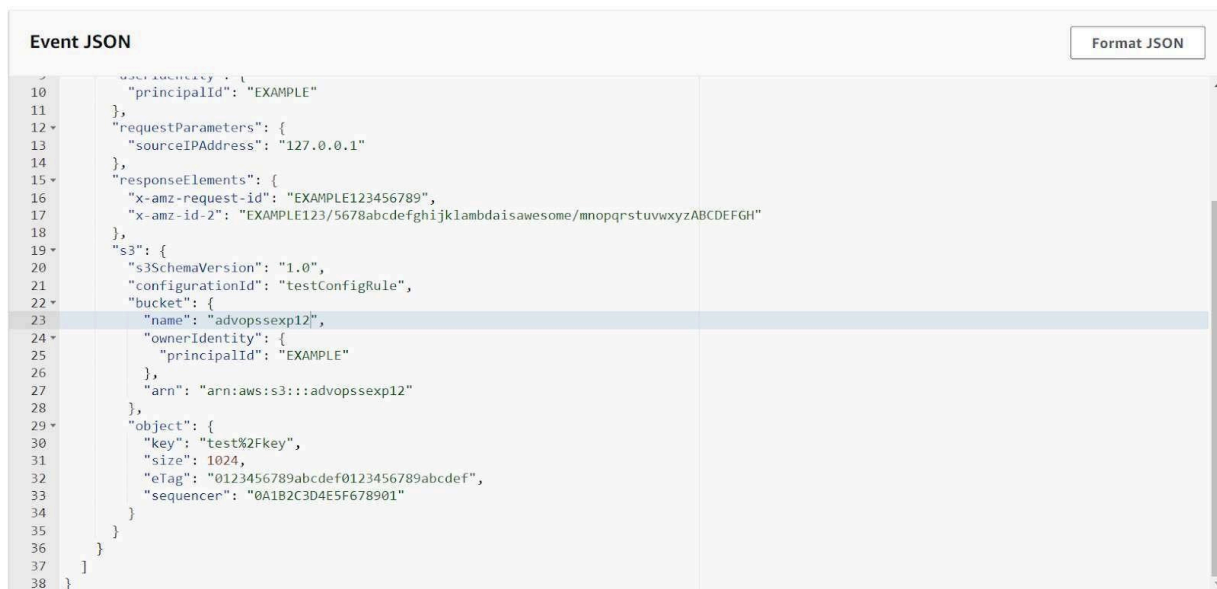
Step 13: Go back to your S3 Bucket and click on Add Files to upload a new file.

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

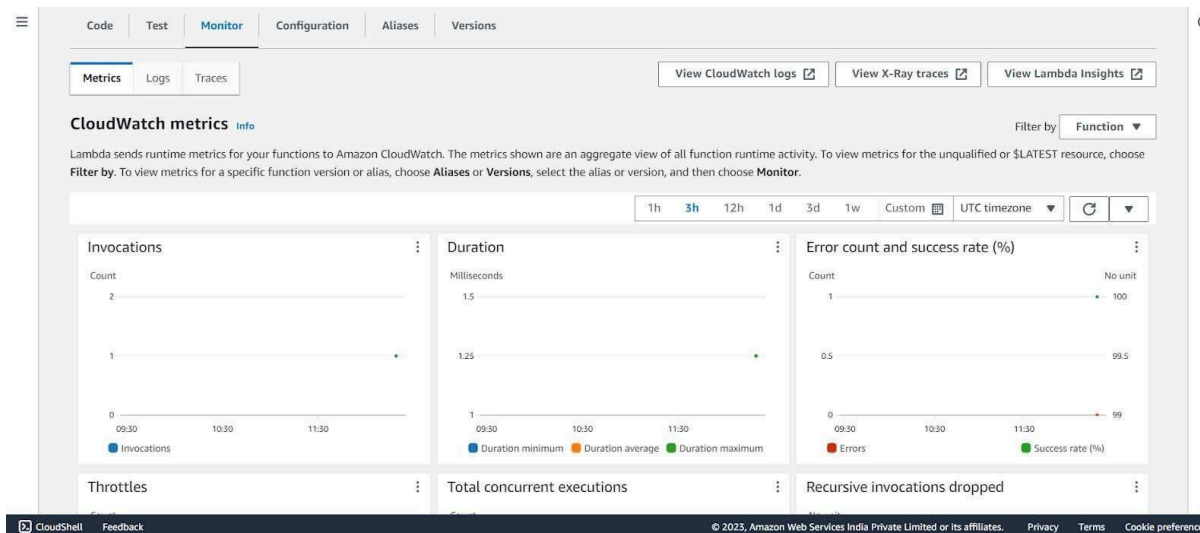




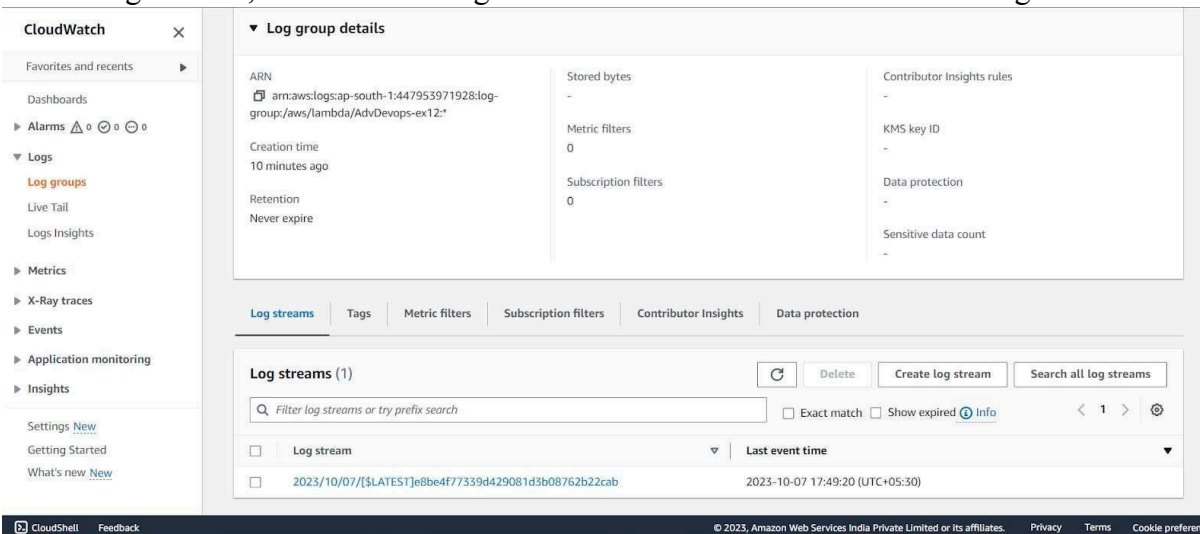
Step 15: After this make the necessary changes in the Test configuration file which we created it previously by replacing the Bucket Name and the ARN of Bucket.



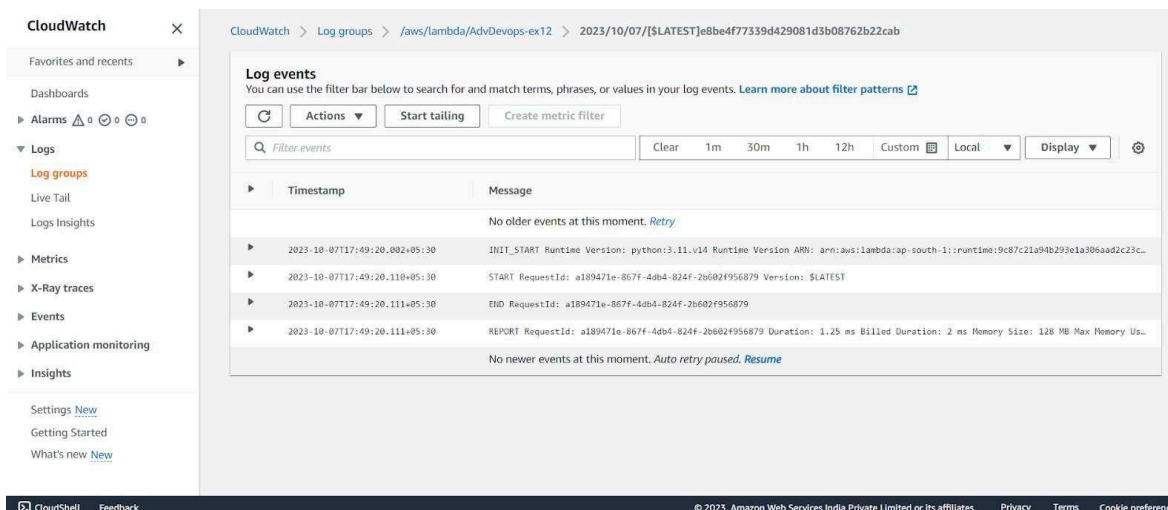
Step 16: Go back to your Lambda function , Refresh it and check the Monitor tab.



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.