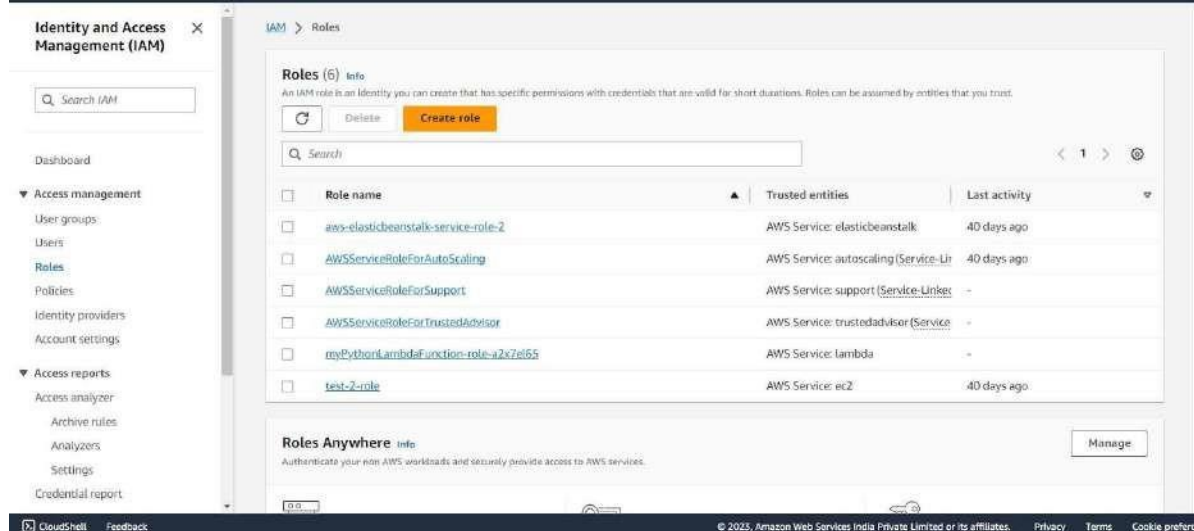


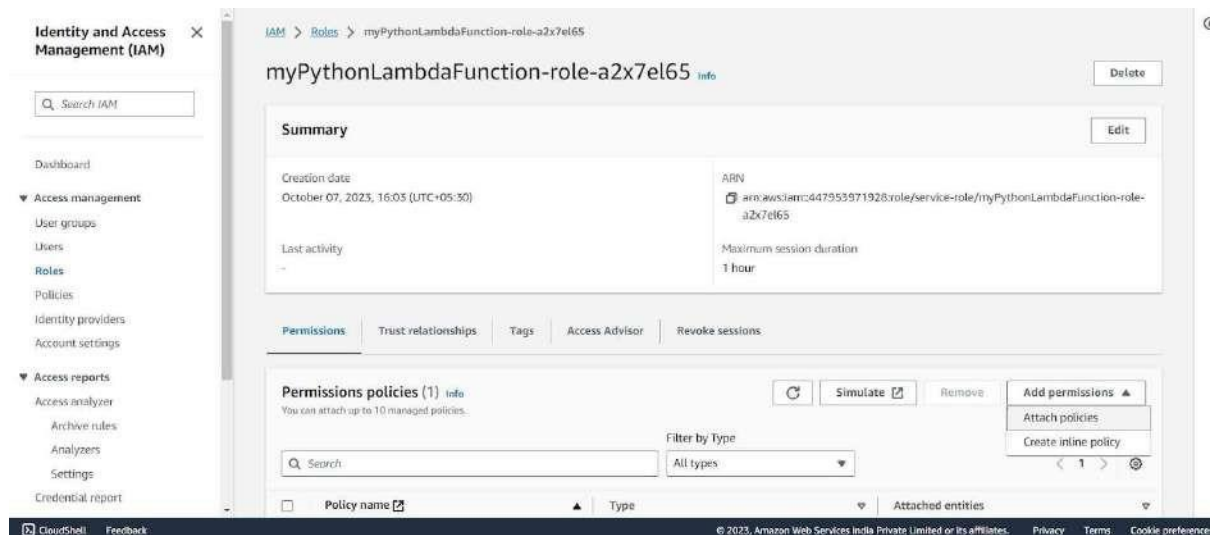
Adv. DevOps Exp. 12

Prajyot Shinde D15A - 57

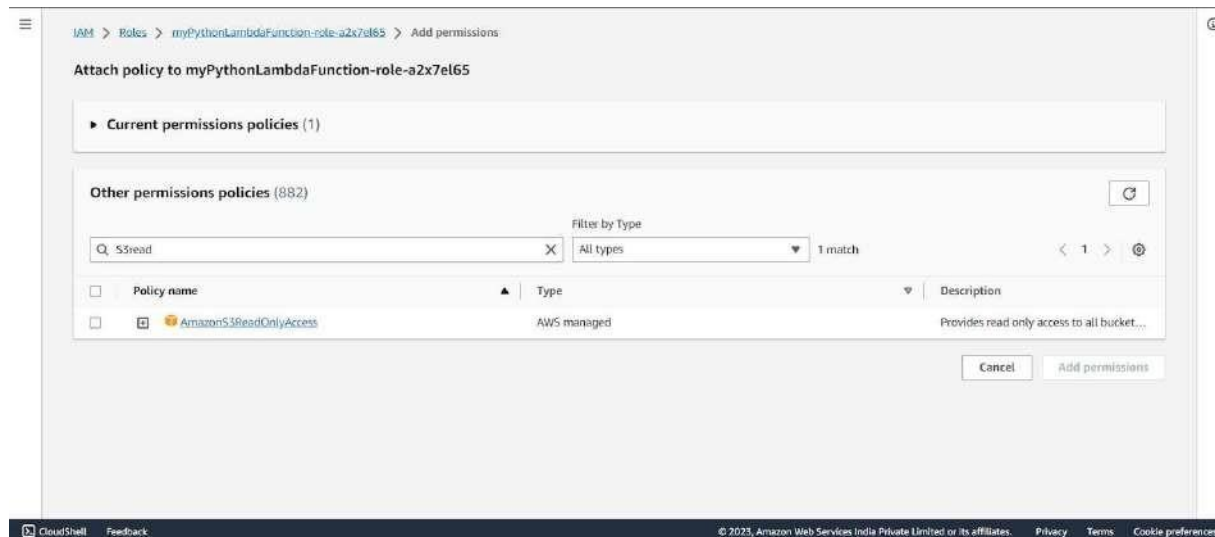
Step 1: Open the IAM (user)



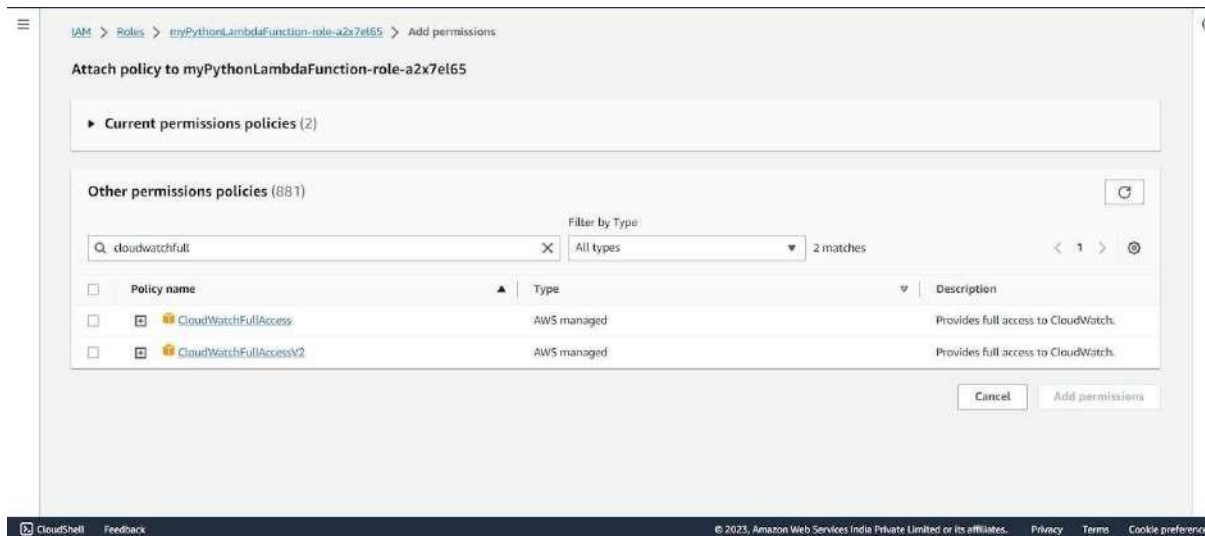
Step 2: Under Attach Policies, add S3-ReadOnly and CloudWatchFull permissions to this role.



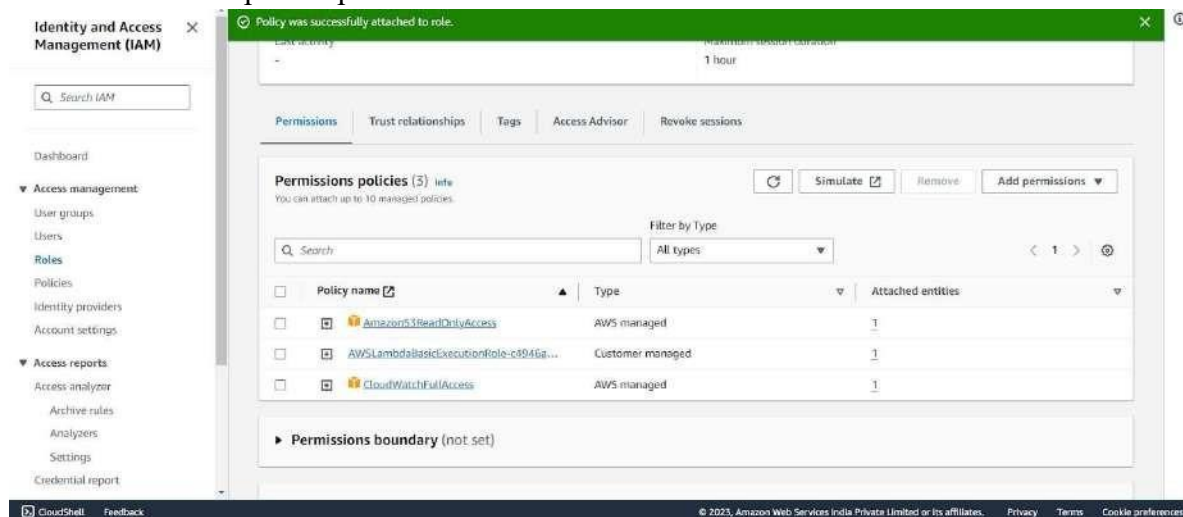
S3-ReadOnly



CloudWatchFull



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



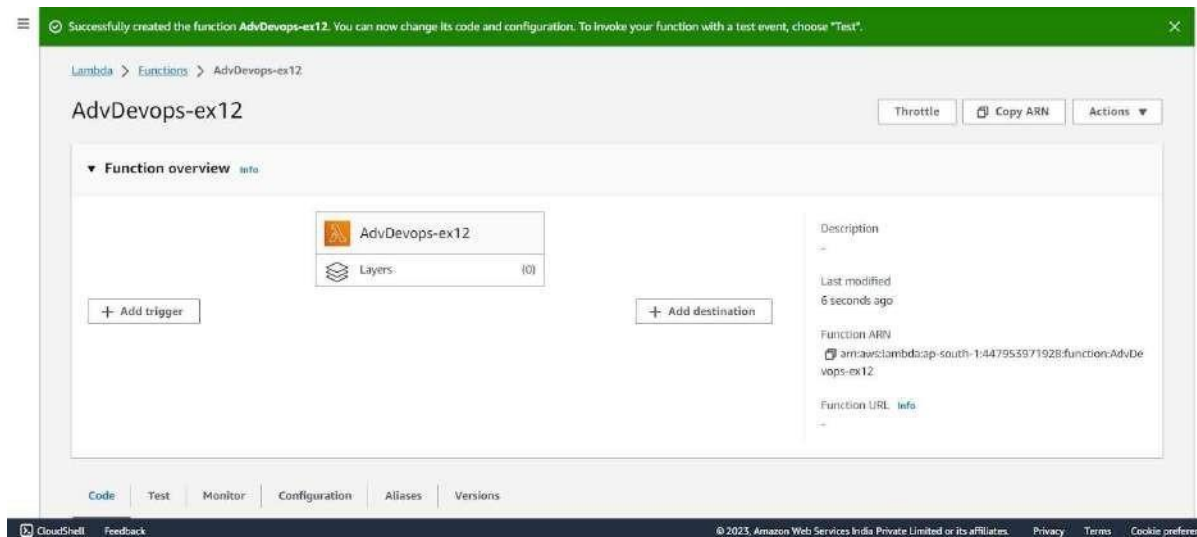
Step 3: Open up AWS Lambda and create a new Python function.

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-ec12', 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.

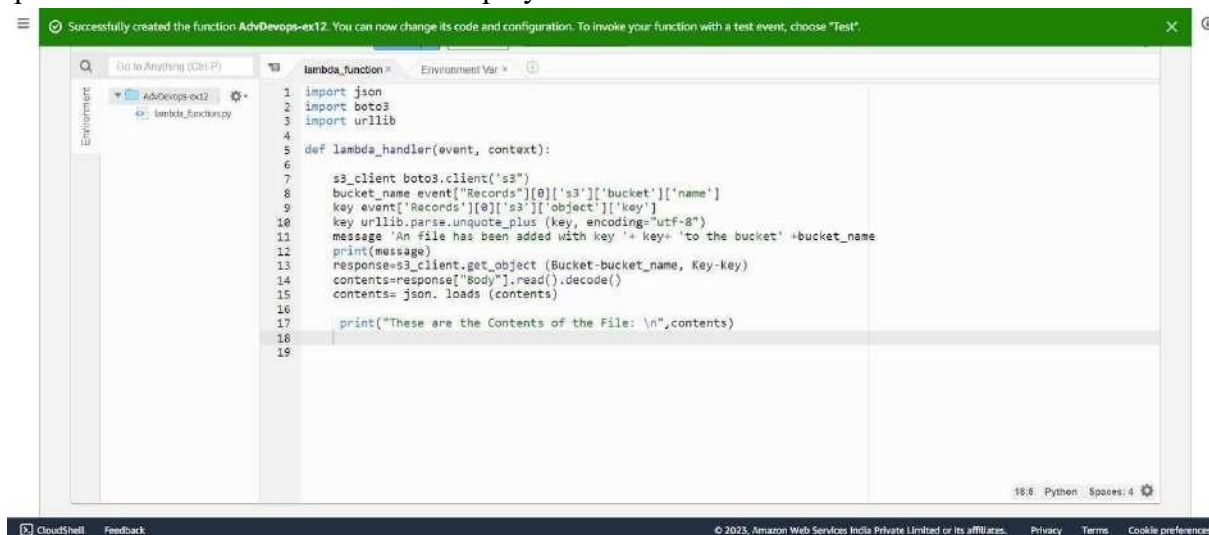
Under Execution Role, choose the existing role, then select the one which was previously created and to which we just added permissions.

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, the 'Existing role' section shows a dropdown menu with the selected role 'service-role/myPythonLambdaFunction-role-a2x7el65'. At the bottom right, there are 'Cancel' and 'Create function' buttons.

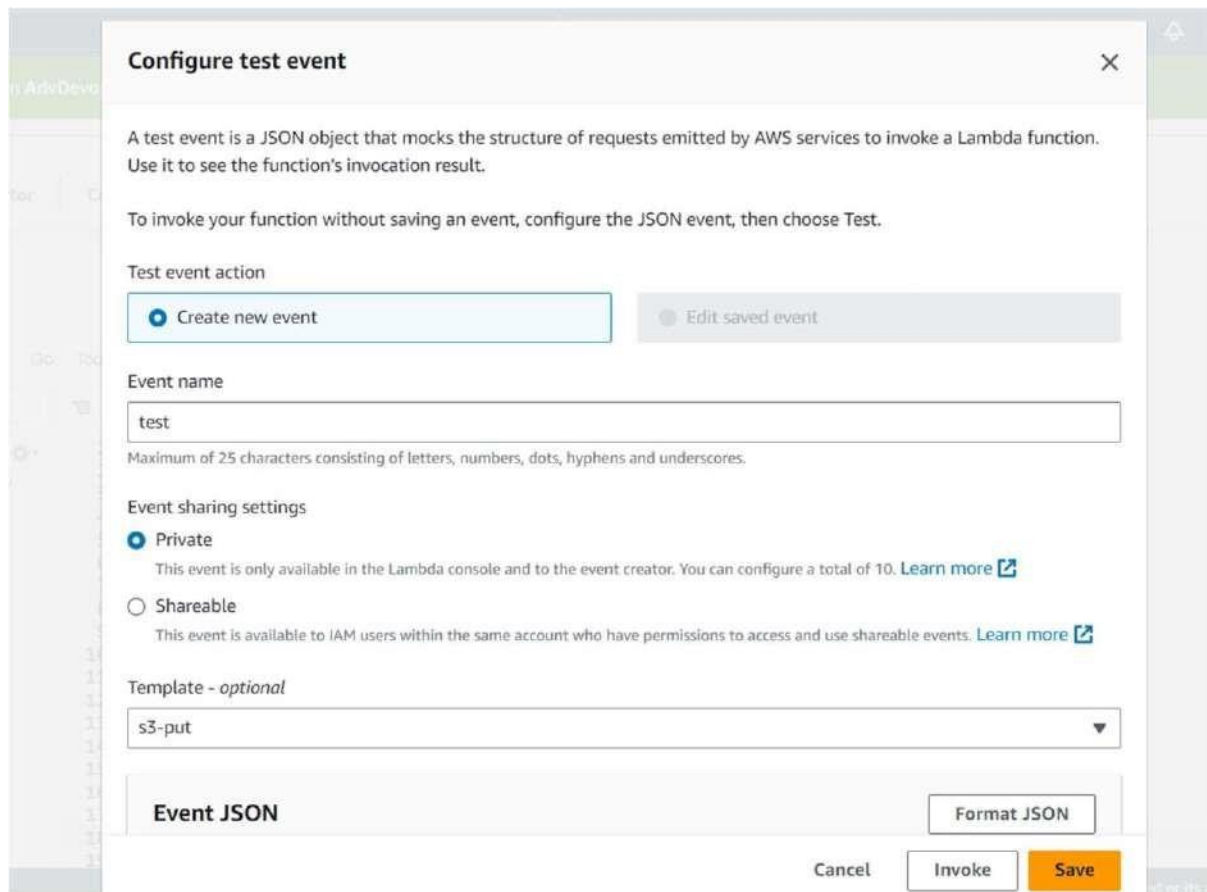
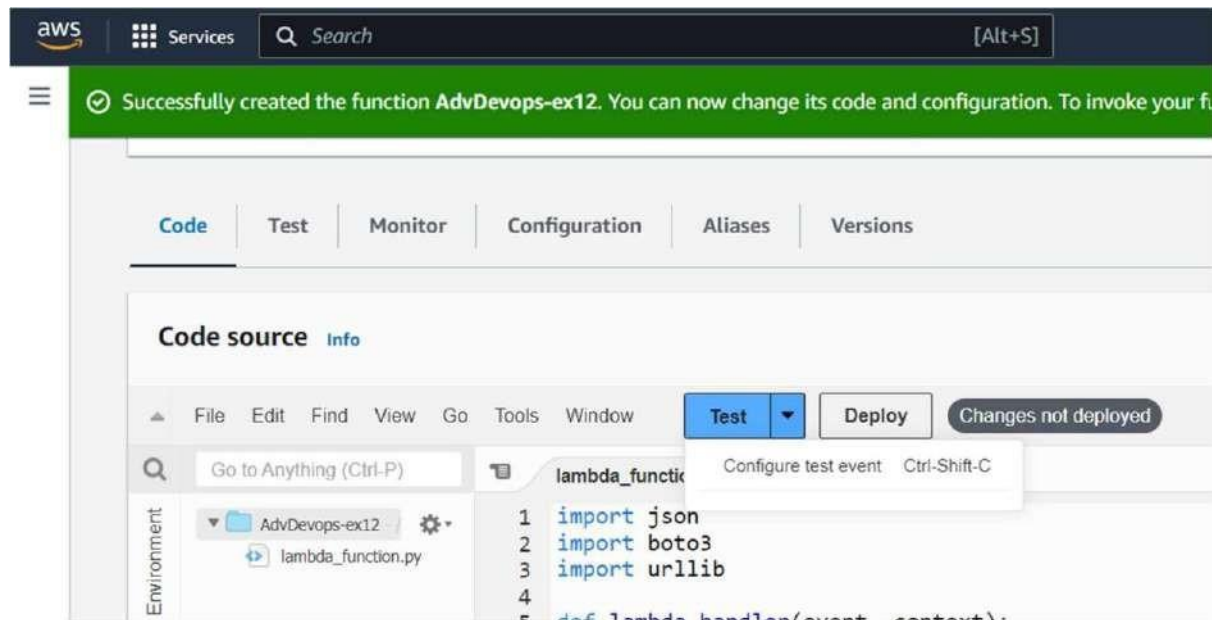
Step 4: The function is up and running.



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.

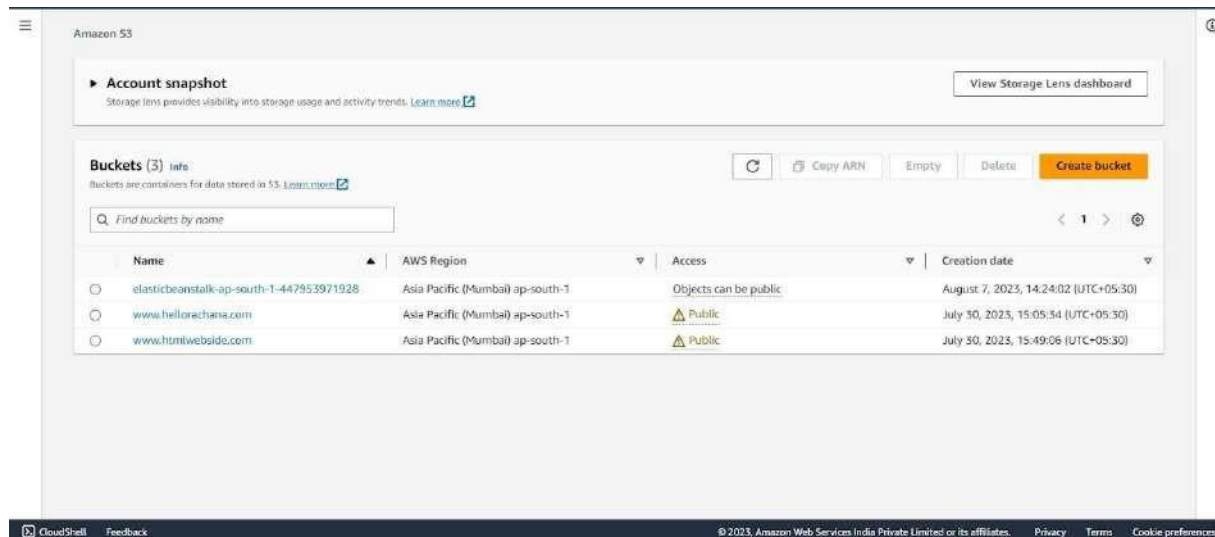


Step 6: Click on Test and choose the 'S3 Put' Template.

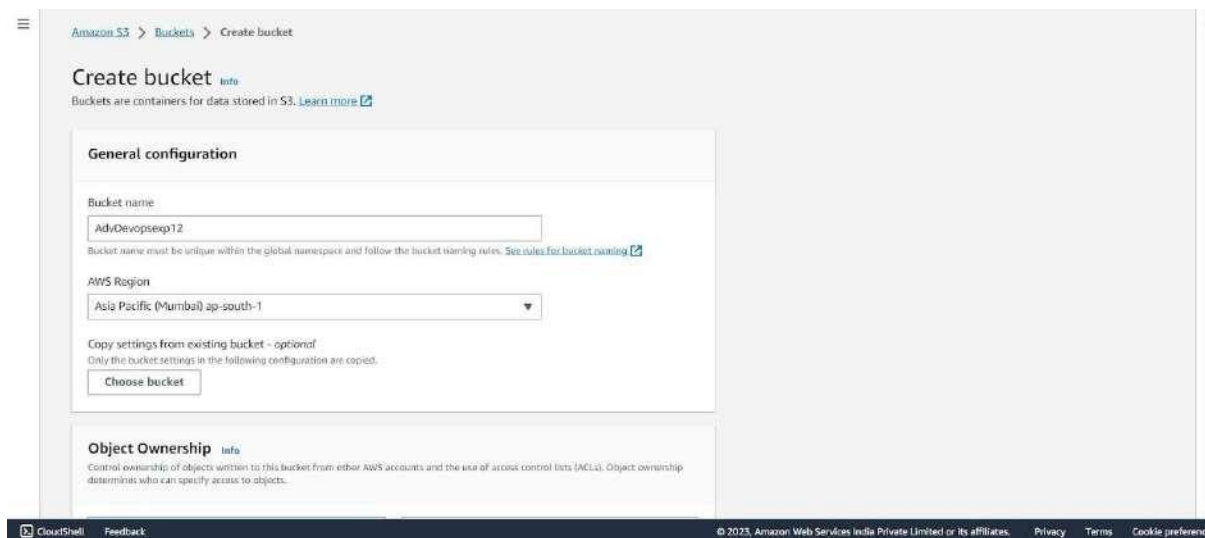


And Save it.

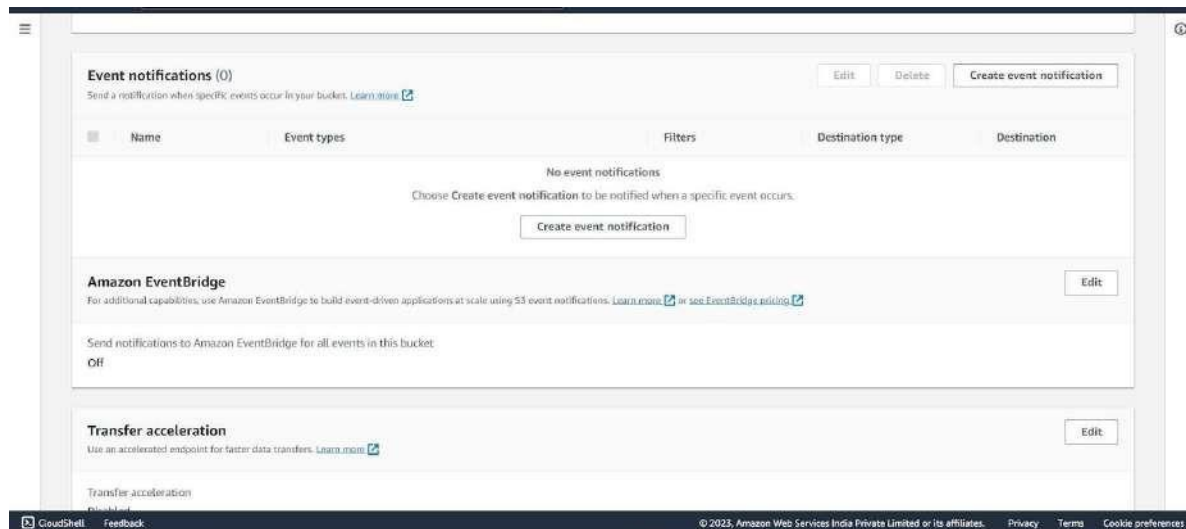
Step 7: Open up the S3 Console and create a new bucket.



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

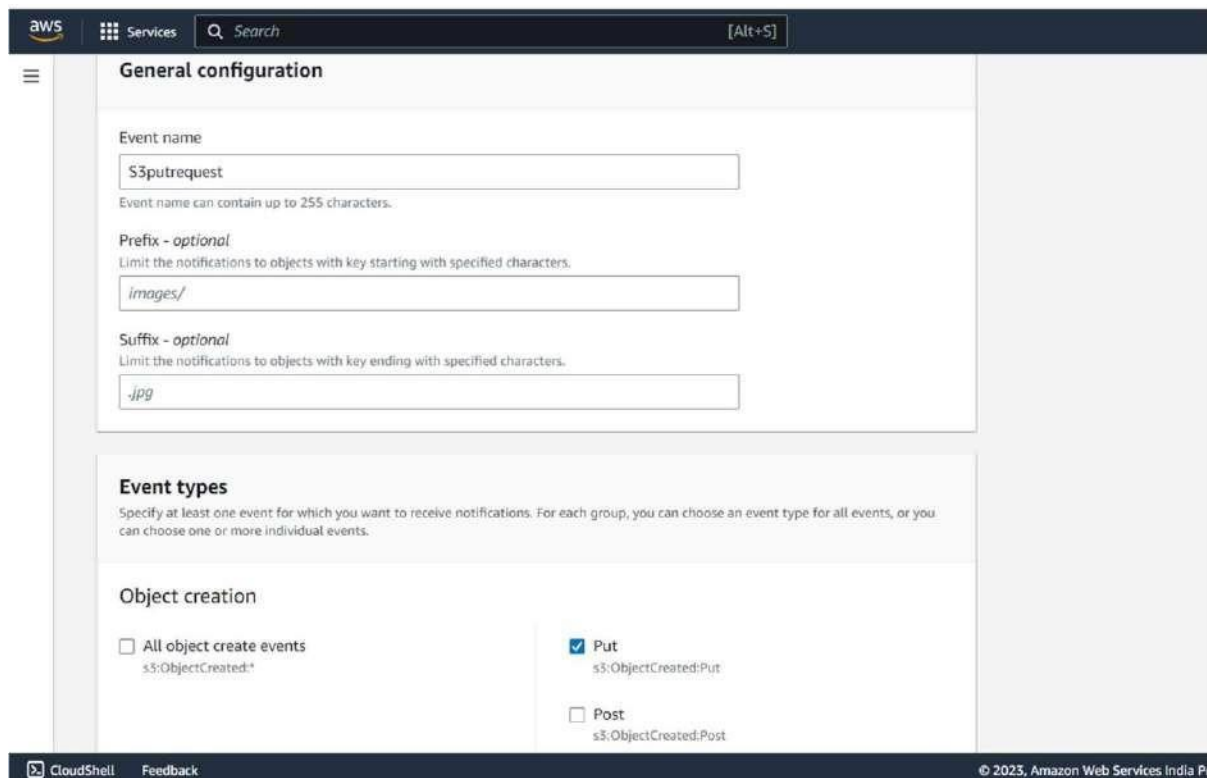


Step 9: Click on the created bucket and under properties, look for events.

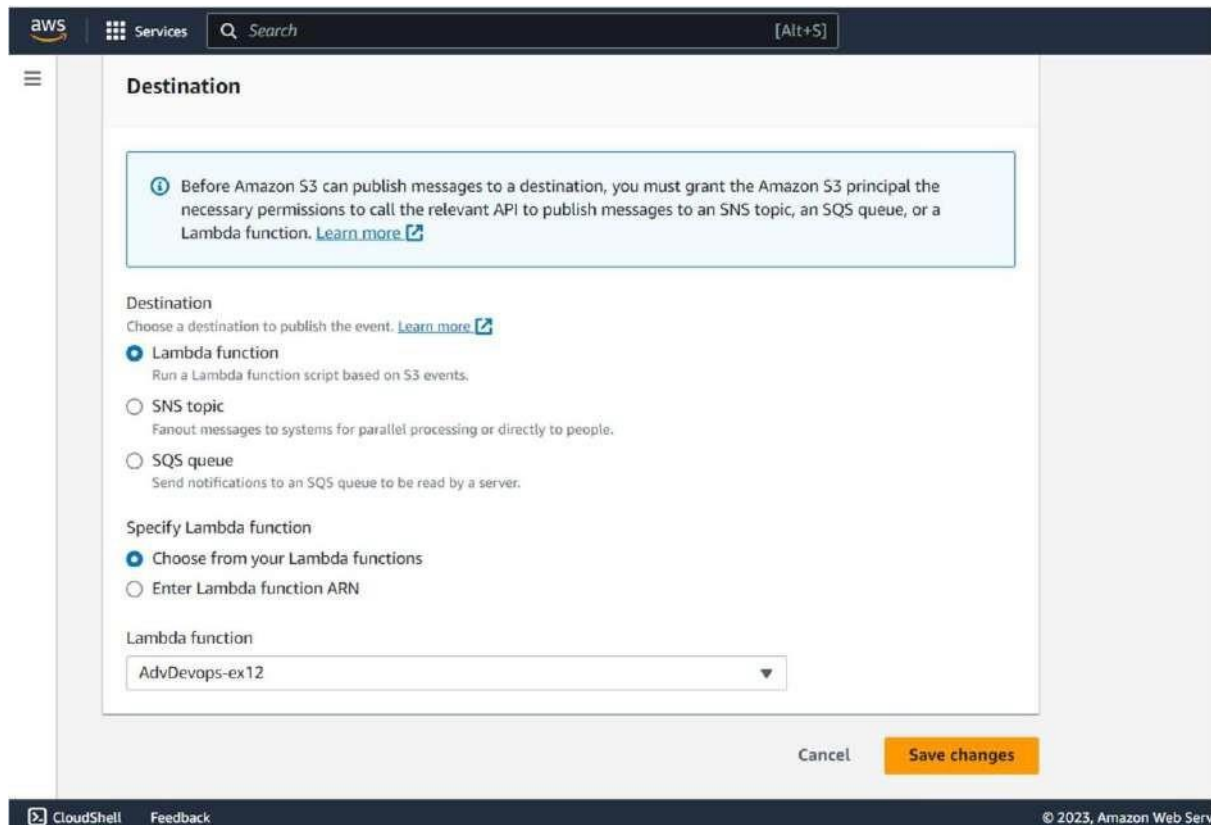


Click on Create Event Notification.

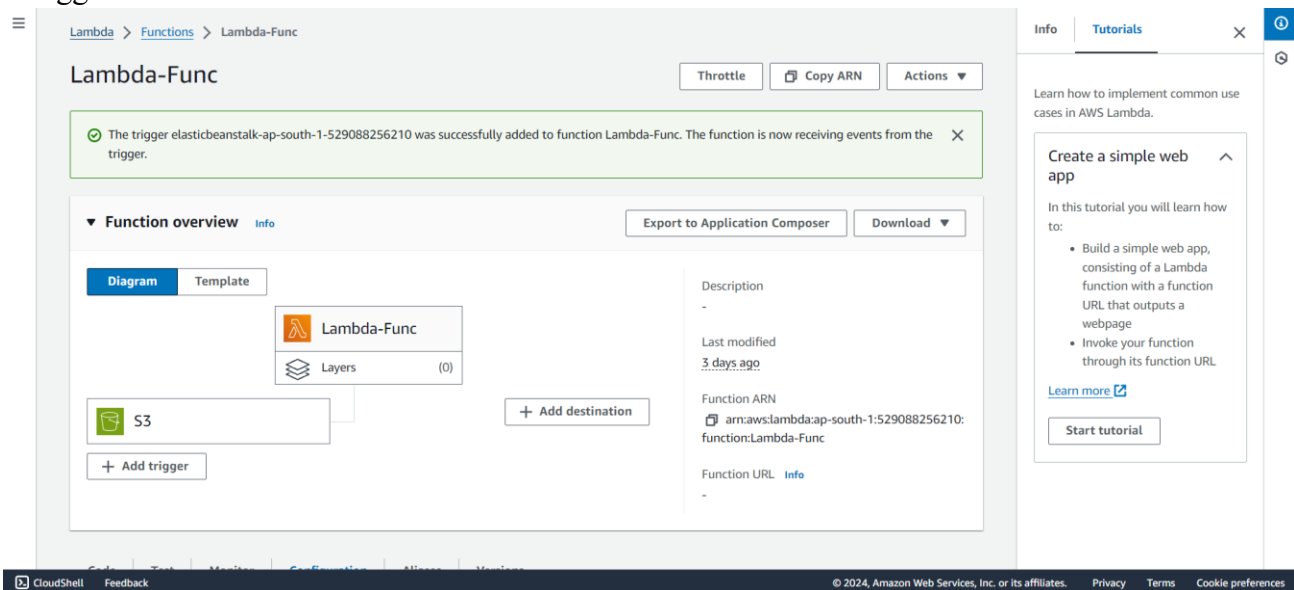
Step 10: Mention an event name and check Put under event types.



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



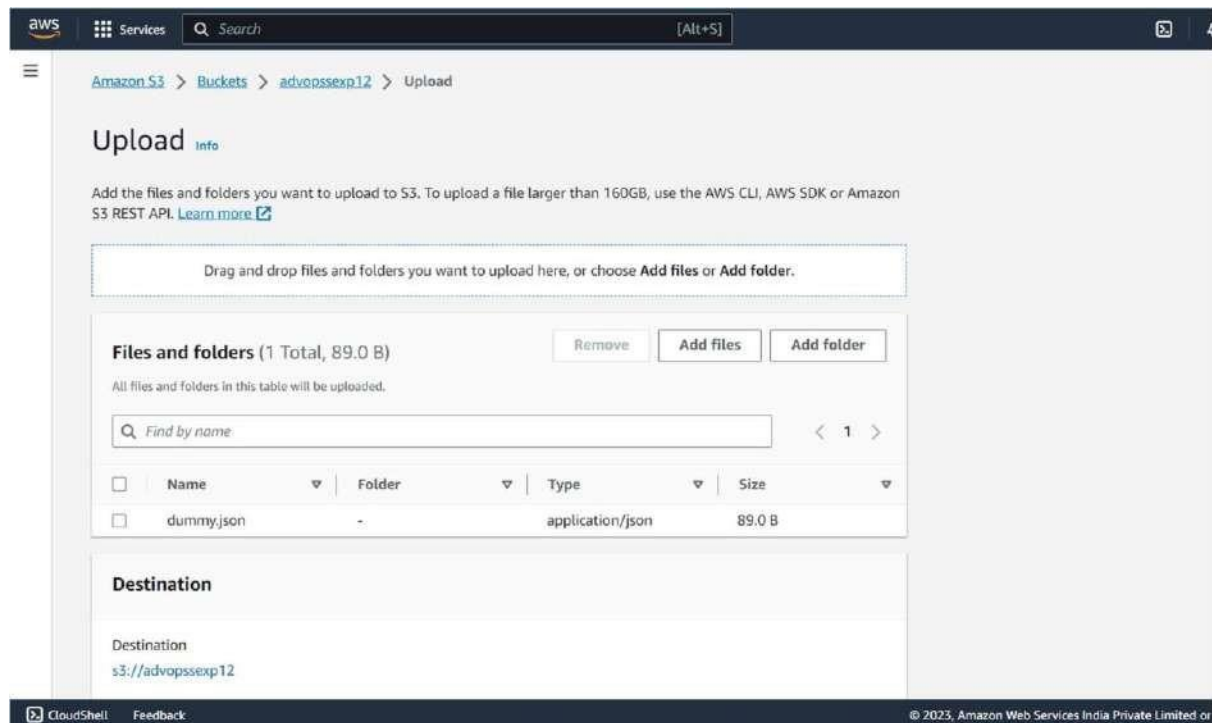
Step 11: Refresh the Lambda function console and you should be able to see an S3 Trigger in the overview.



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

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

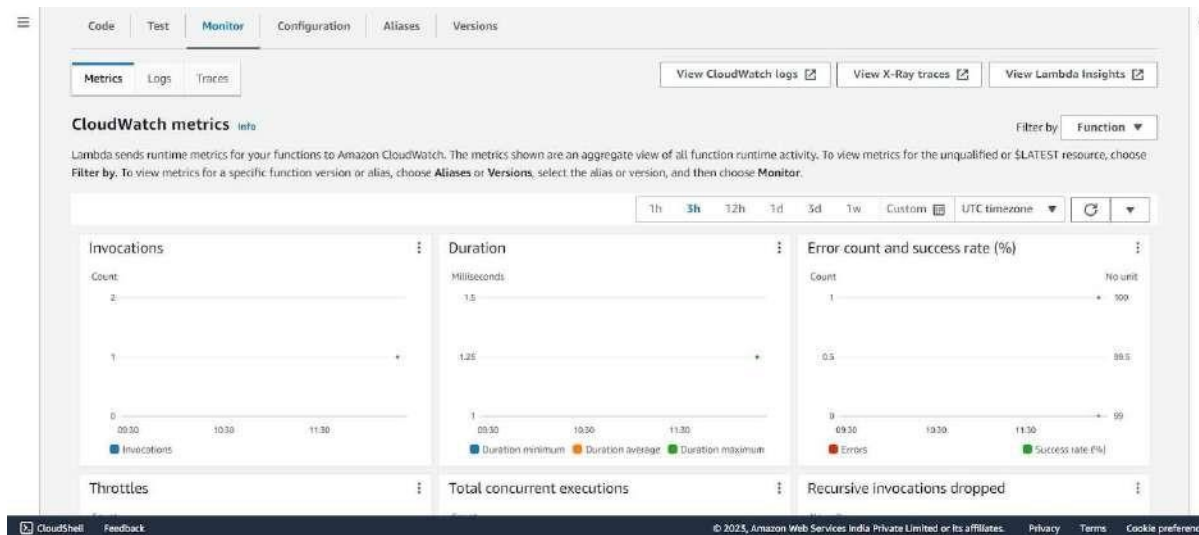
Step 14: Select the dummy data file from your computer and click Upload.



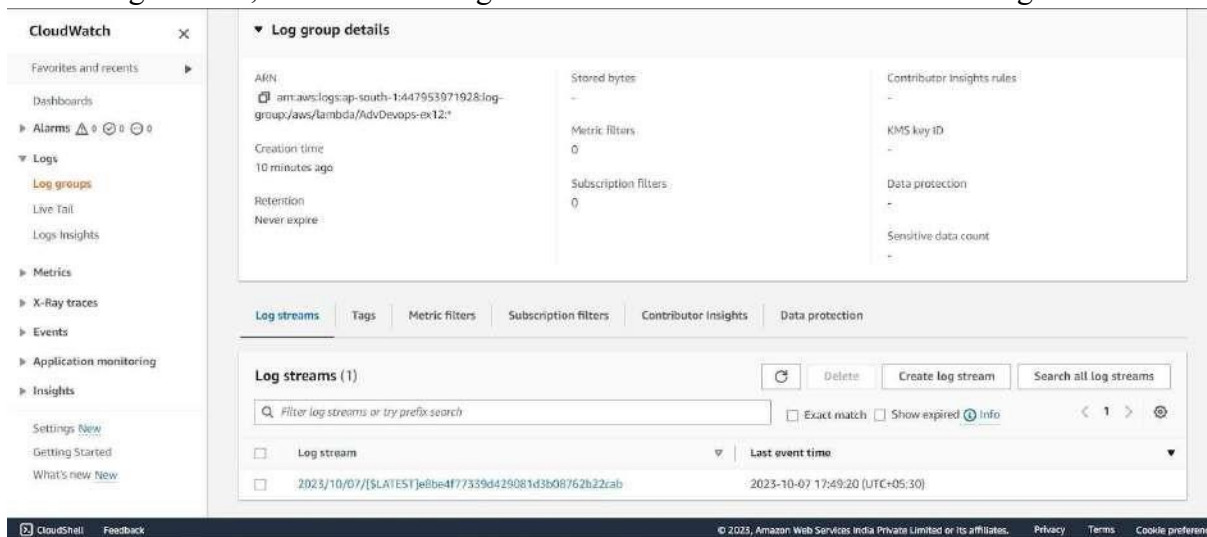
Step 15: After this make the necessary changes in the Test configuration file which we created it previously by replacing the Bucket Name and the ARN of Bucket.



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.

The screenshot displays the AWS CloudWatch console interface. On the left is a navigation sidebar with sections for 'Favorites and recents', 'Dashboards', 'Alarms', 'Logs', 'Metrics', 'X-Ray traces', 'Events', 'Application Signals', 'Network monitoring', and 'Insights'. The 'Logs' section is expanded, showing 'Log groups', 'Log Anomalies', 'Live Tail', 'Logs Insights', and 'Contributor Insights'. The main panel shows the breadcrumb path: 'CloudWatch > Log groups > /aws/lambda/Lambda-Func > 2024/10/11/[\${LATEST}]aff54e3f606143548ec12e1f2f25a4df'. Below the path, there's a 'Log events' header with a refresh button, an 'Actions' dropdown, 'Start tailing', and 'Create metric filter' buttons. A search bar contains the text 'Filter events - press enter to search'. Below the search bar, a table lists log events with columns for 'Timestamp' and 'Message'. The table shows several events, including 'INIT_START', 'START', 'END', and 'REPORT' messages, each with associated request IDs and durations. At the bottom of the table, it says 'No newer events at this moment. Auto retry paused. Resume'. The footer of the console shows 'CloudShell', 'Feedback', and copyright information for Amazon Web Services, Inc.

Timestamp	Message
No older events at this moment. Retry	
2024-10-11T05:23:33.473Z	INIT_START Runtime Version: python:3.12.v36 Runtime Version ARN: arn:aws:lambda:ap-south-1::runtime:188d9ca2e2714ff5637bd2bb...
2024-10-11T05:23:33.568Z	START RequestId: 4662c213-1c33-4099-a4ac-ef71af92f36a Version: \$LATEST
2024-10-11T05:23:33.587Z	END RequestId: 4662c213-1c33-4099-a4ac-ef71af92f36a
2024-10-11T05:23:33.587Z	REPORT RequestId: 4662c213-1c33-4099-a4ac-ef71af92f36a Duration: 1.93 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memor...
2024-10-11T05:23:55.230Z	START RequestId: fb1457e5-f5cd-4787-bc2b-141c11b7271 Version: \$LATEST
2024-10-11T05:23:55.247Z	END RequestId: fb1457e5-f5cd-4787-bc2b-141c11b7271
2024-10-11T05:23:55.247Z	REPORT RequestId: fb1457e5-f5cd-4787-bc2b-141c11b7271 Duration: 1.55 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memor...
2024-10-11T05:25:23.104Z	START RequestId: 1bbbb172-9698-4cf7-912b-d7f9d38c7748 Version: \$LATEST
2024-10-11T05:25:23.106Z	END RequestId: 1bbbb172-9698-4cf7-912b-d7f9d38c7748
2024-10-11T05:25:23.106Z	REPORT RequestId: 1bbbb172-9698-4cf7-912b-d7f9d38c7748 Duration: 1.63 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memor...
No newer events at this moment. Auto retry paused. Resume	

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.