

EXPERIMENT - 12

Aim: To create a Lambda function which will log "An Image has been added" once you add an object to a specific bucket in S3.

THEORY:

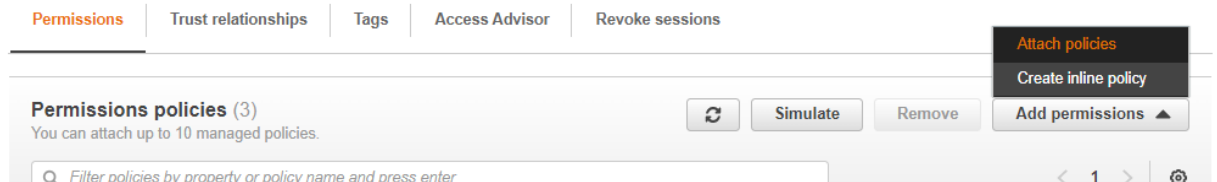
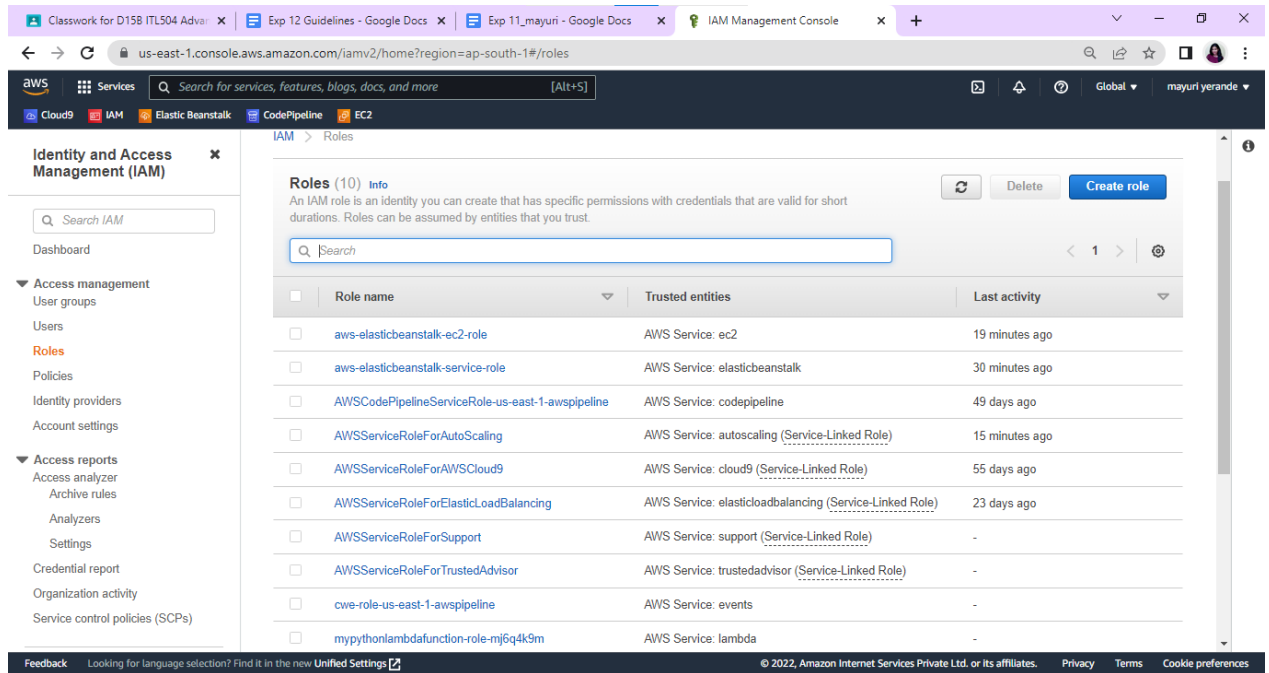
- ⇒ AWS Lambda is a serverless computing service provided by Amazon web services. Users of AWS Lambda create functions, self contained application written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes these functions in an efficient and flexible manner.
- ⇒ The Lambda functions can perform any kind of computing task, from cleaning web pages and processing streams of data to calling APIs and integrating with other AWS services.
- ⇒ The concept of "serverless" computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is fully managed service that takes care of all infrastructure for you. And so "serverless" just means that servers, the operating systems, the network layer and rest of infrastructure have already taken care so that we can focus on writing application code.

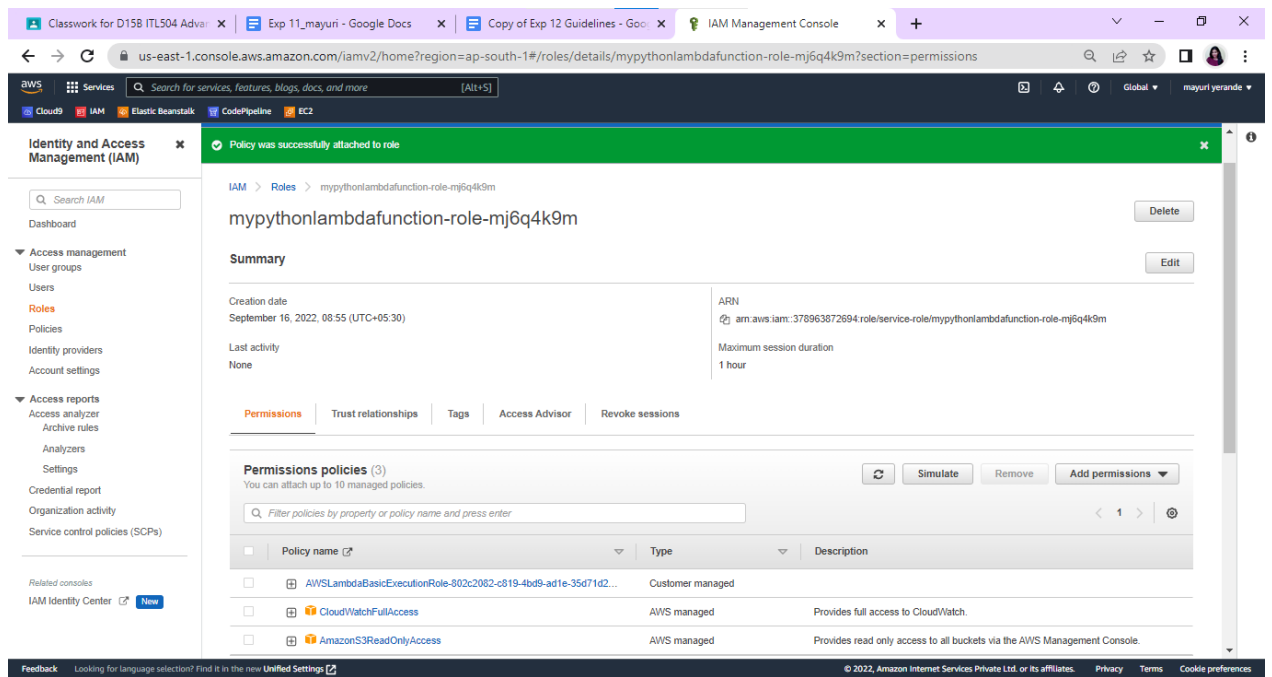
⇒ Features of Lambda :-

- AWS Lambda easily scales the infrastructure without any additional configuration.
- It offers multiple options like AWS S3, Kinesis, CodeCommit and many more to trigger an event.
- You don't need to invest upfront. You pay only for memory used by lambda function and minimal cost on number of requests hence cost efficient.
- AWS Lambda is secure. It uses AWS IAM to define all roles and security policies.
- It offers fault tolerance for both external running code and the function.

Steps to create a Lambda function that reacts to uploads in an S3 Bucket

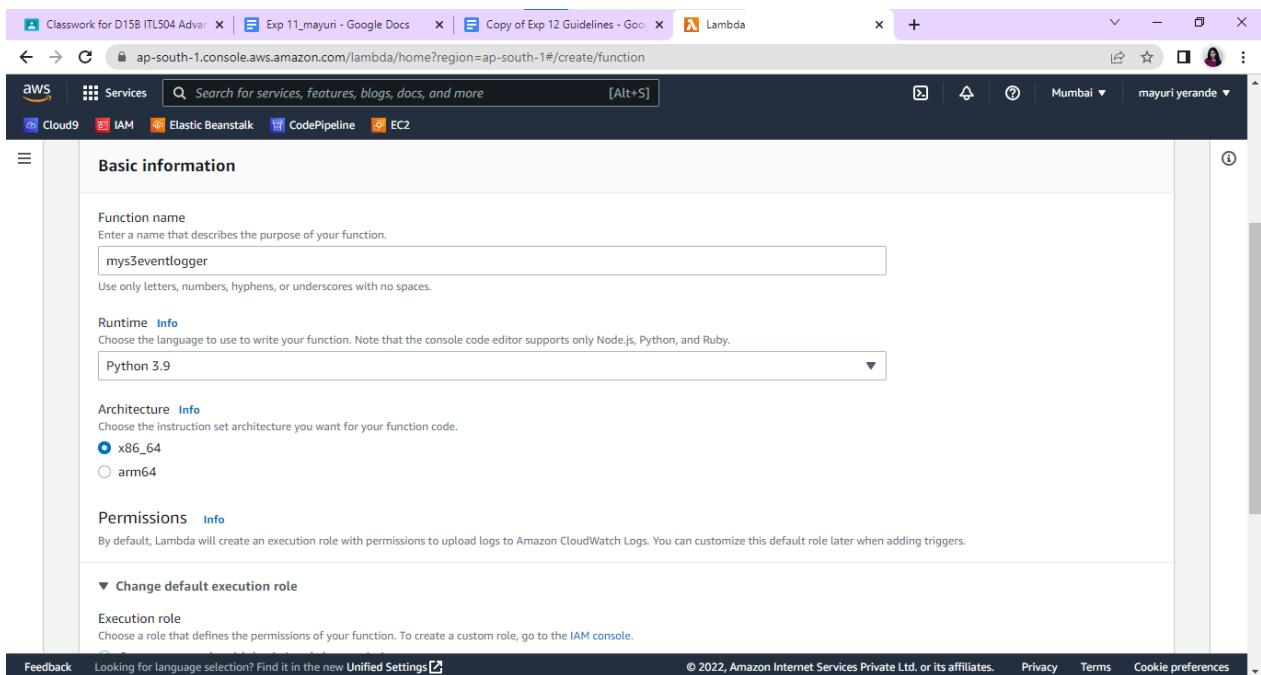
1. Open up the IAM Console and under Roles, choose the Role we previously created for the Python Lambda Function.

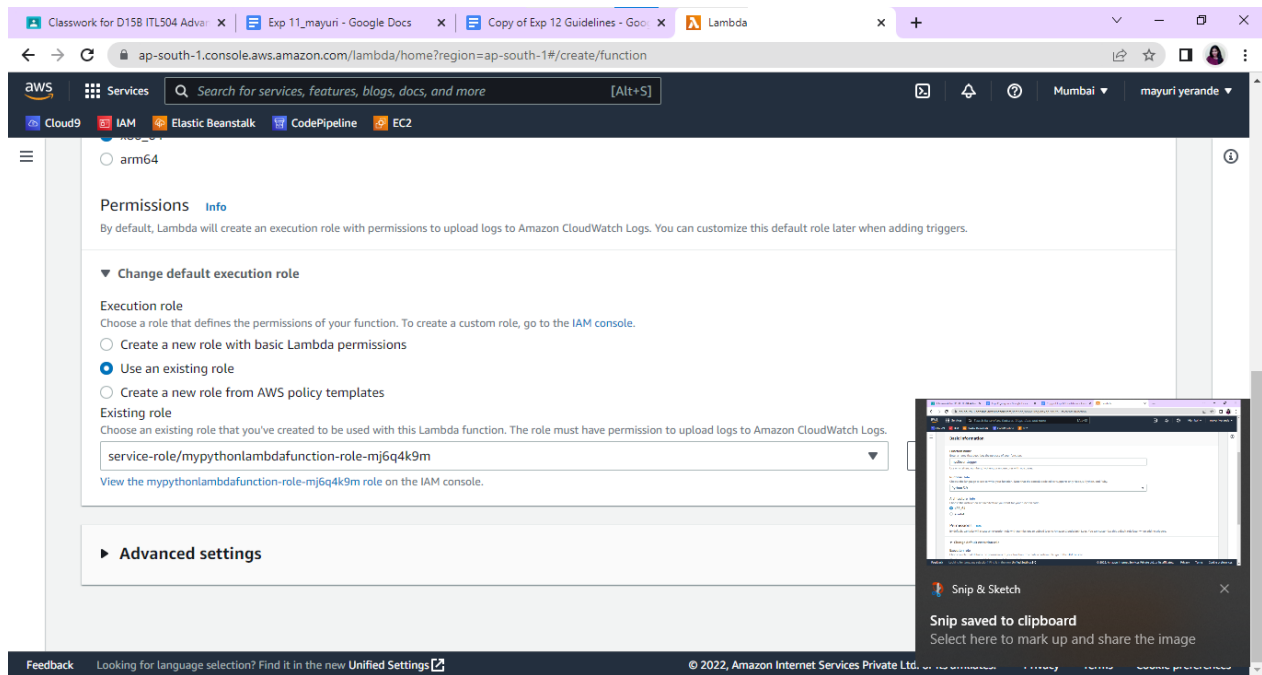




Under Attach Policies, add S3-ReadOnly and CloudWatchFull permissions to this role.

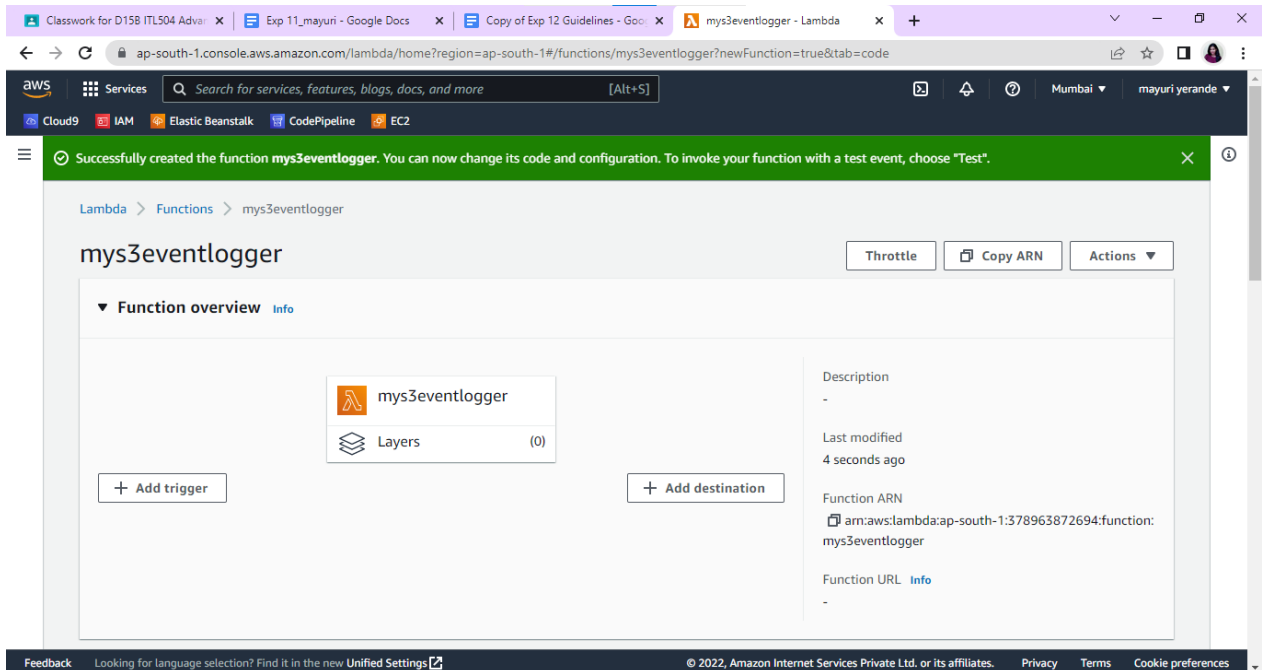
2. Open up AWS Lambda and create a new Python function





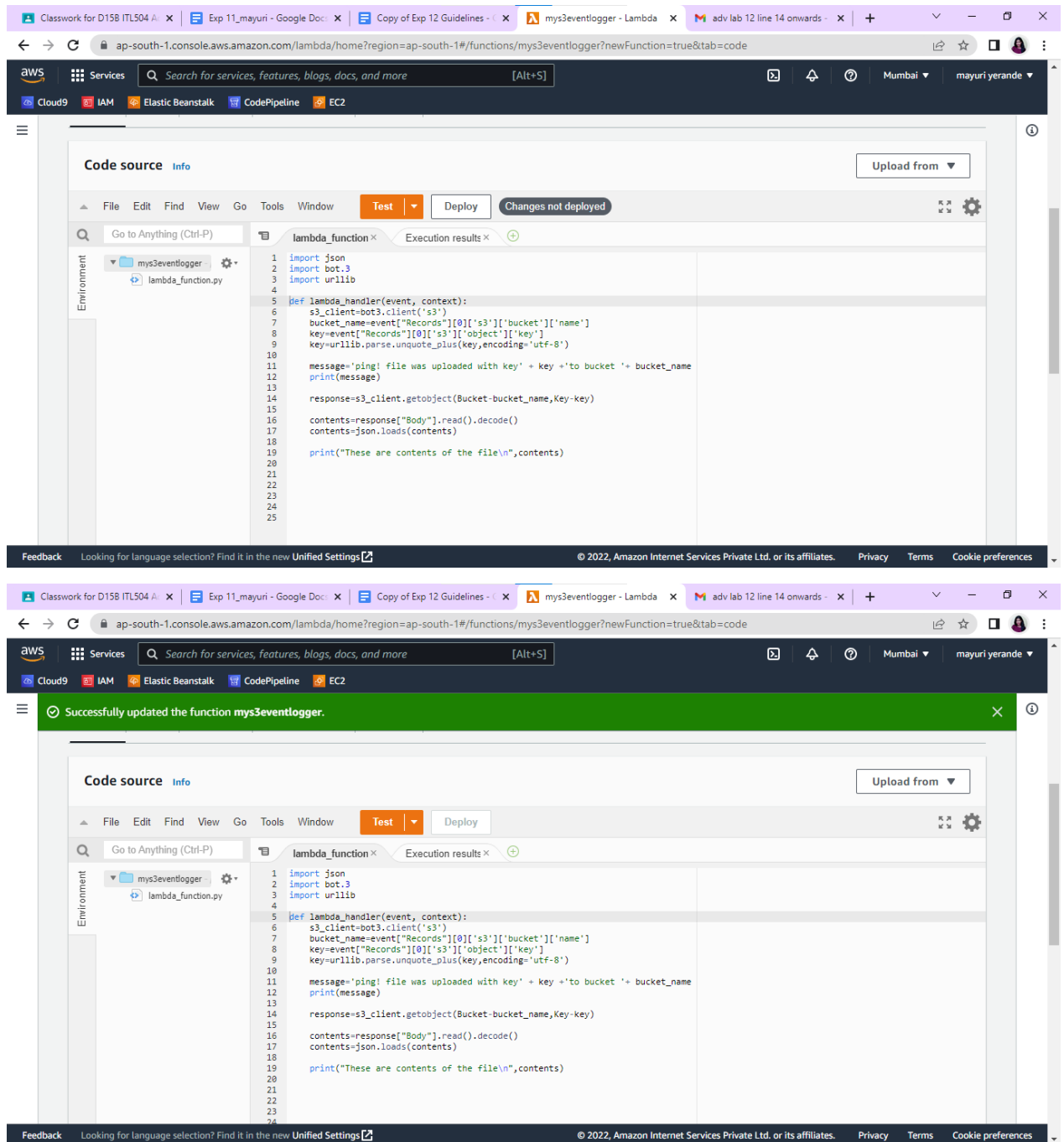
Under Execution Role, choose the existing role, the one which was previously created and to which we just added permissions.

3. The function is up and running.



4. 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.



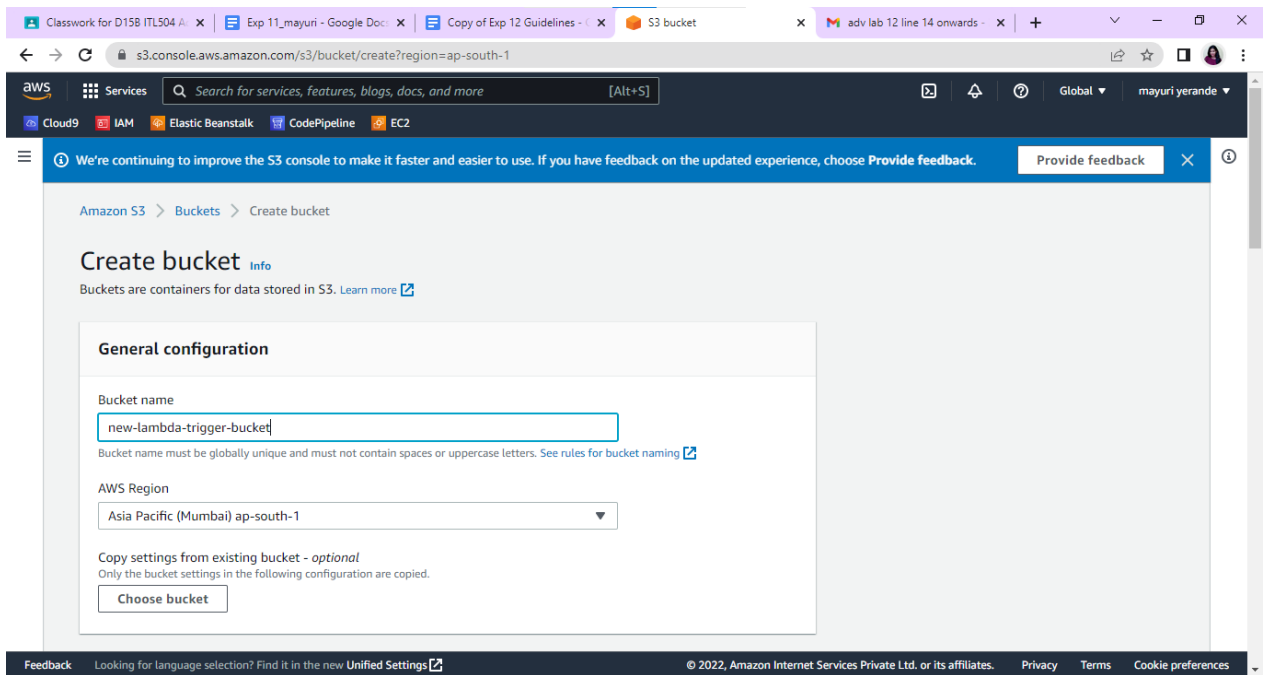
The screenshot shows the AWS Lambda console interface for the function `mys3eventlogger`. The code source is displayed in a text editor with the following Python code:

```
1 import json
2 import bot3
3 import urllib
4
5 def lambda_handler(event, context):
6     s3_client=bot3.client('s3')
7     bucket_name=event["Records"][0]['s3']['bucket']['name']
8     key=event["Records"][0]['s3']['object']['key']
9     key=urllib.parse.unquote_plus(key,encoding='utf-8')
10
11     message='ping! file was uploaded with key ' + key +'to bucket ' + bucket_name
12     print(message)
13
14     response=s3_client.getObject(Bucket=bucket_name,Key=key)
15
16     contents=response["Body"].read().decode()
17     contents=json.loads(contents)
18
19     print("These are contents of the file\n",contents)
20
21
22
23
24
25
```

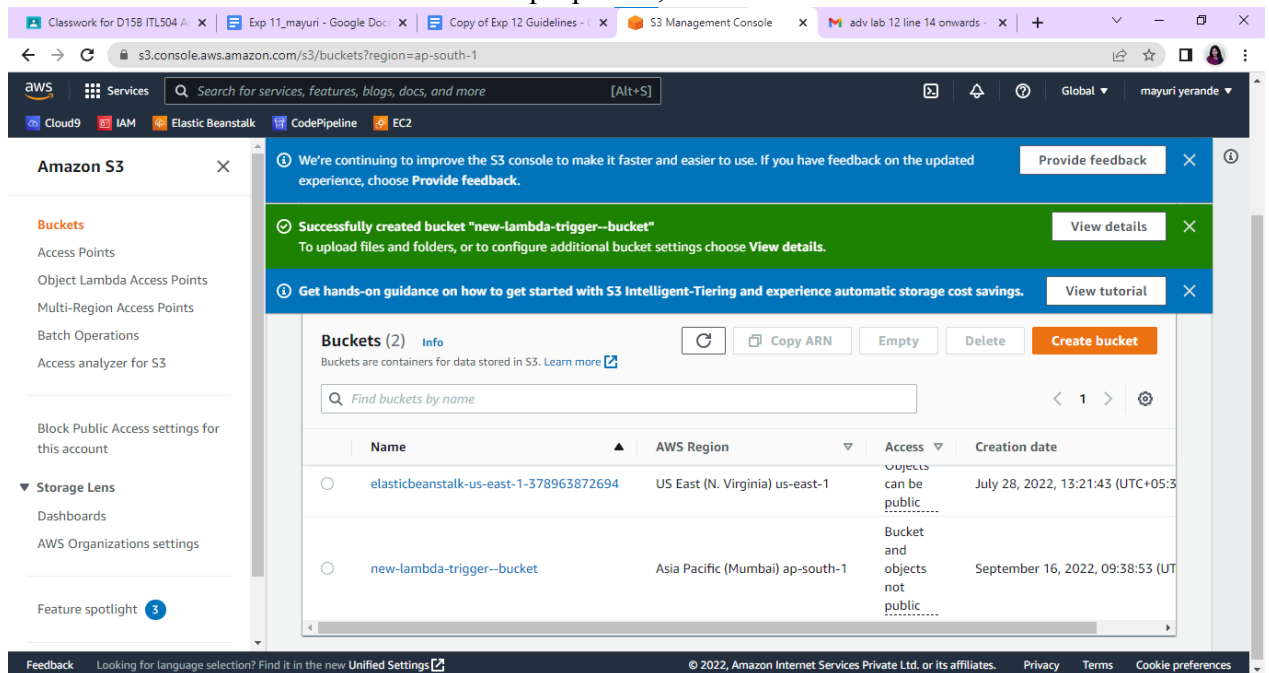
The console shows a green notification bar at the top stating "Successfully updated the function `mys3eventlogger`." The interface includes a search bar, a navigation menu, and a footer with the text "© 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences".

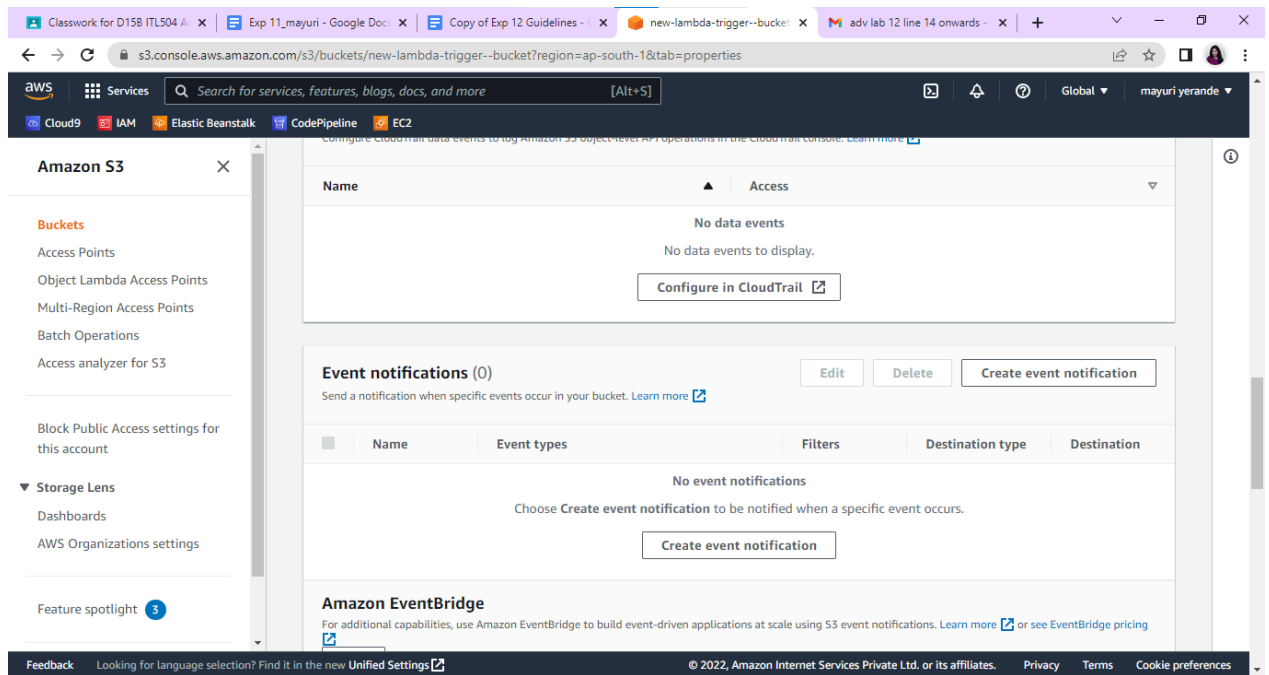
5. Open up the S3 Console and create a new bucket

6. With all general settings, create the bucket in the same region as the function.



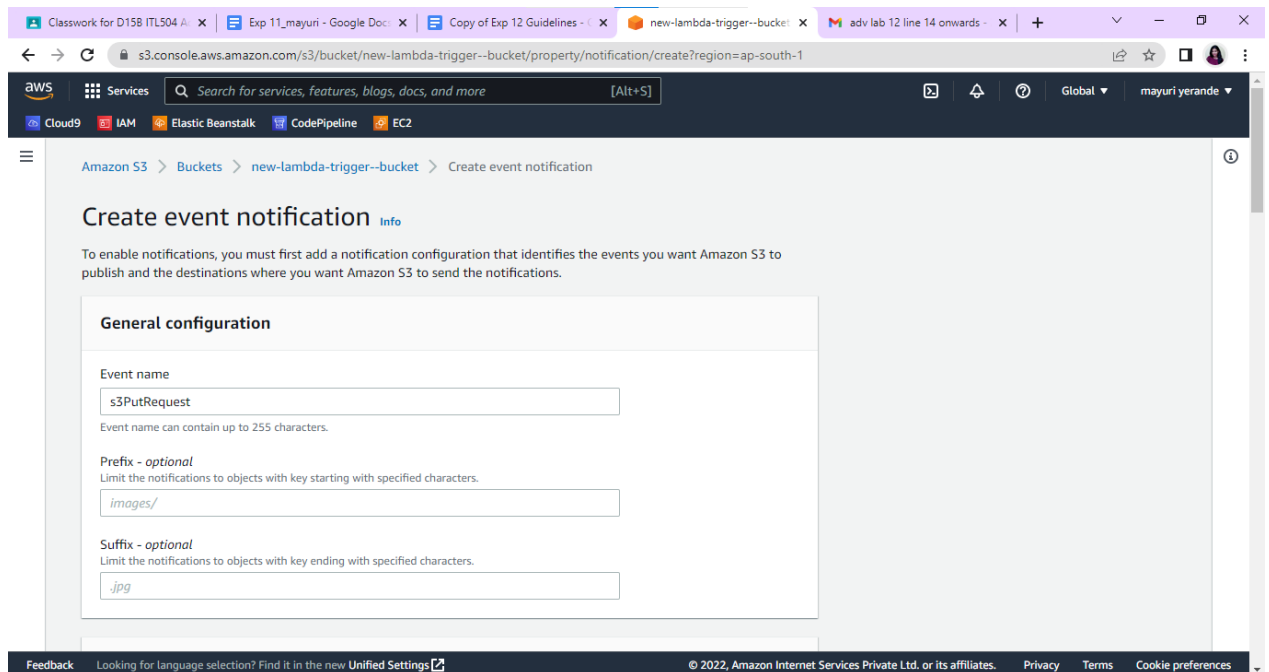
7. Click on the created bucket and under properties, look for events.

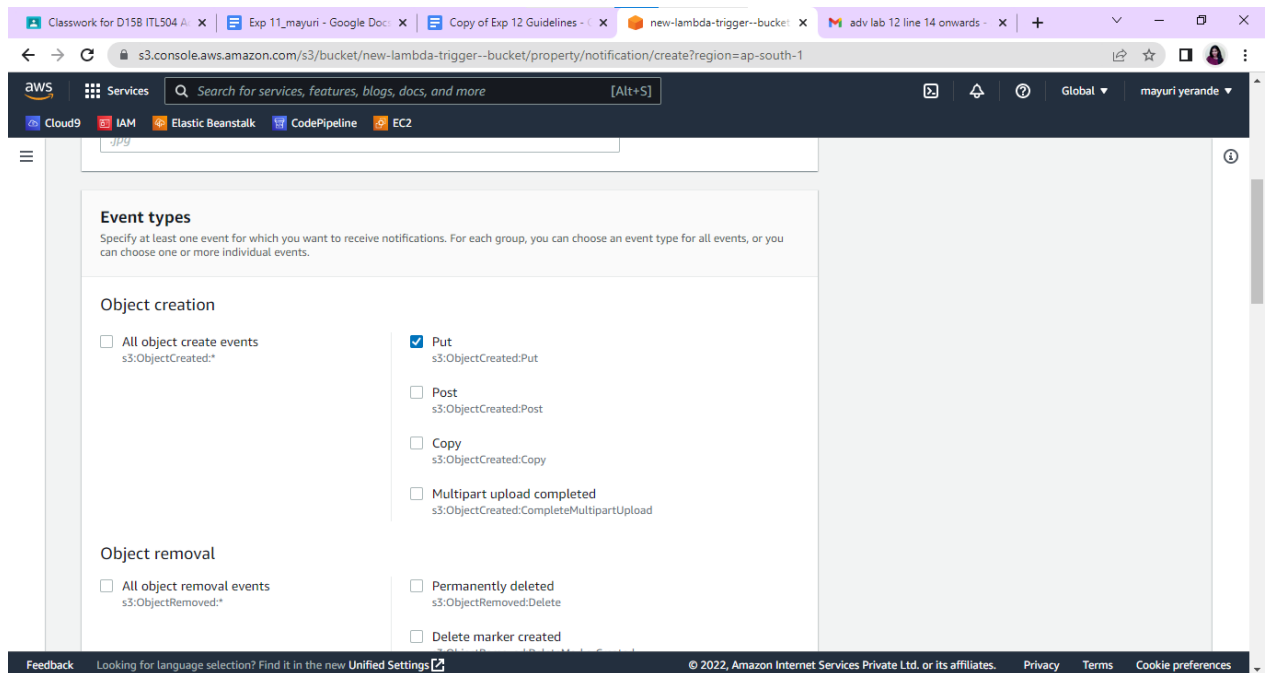




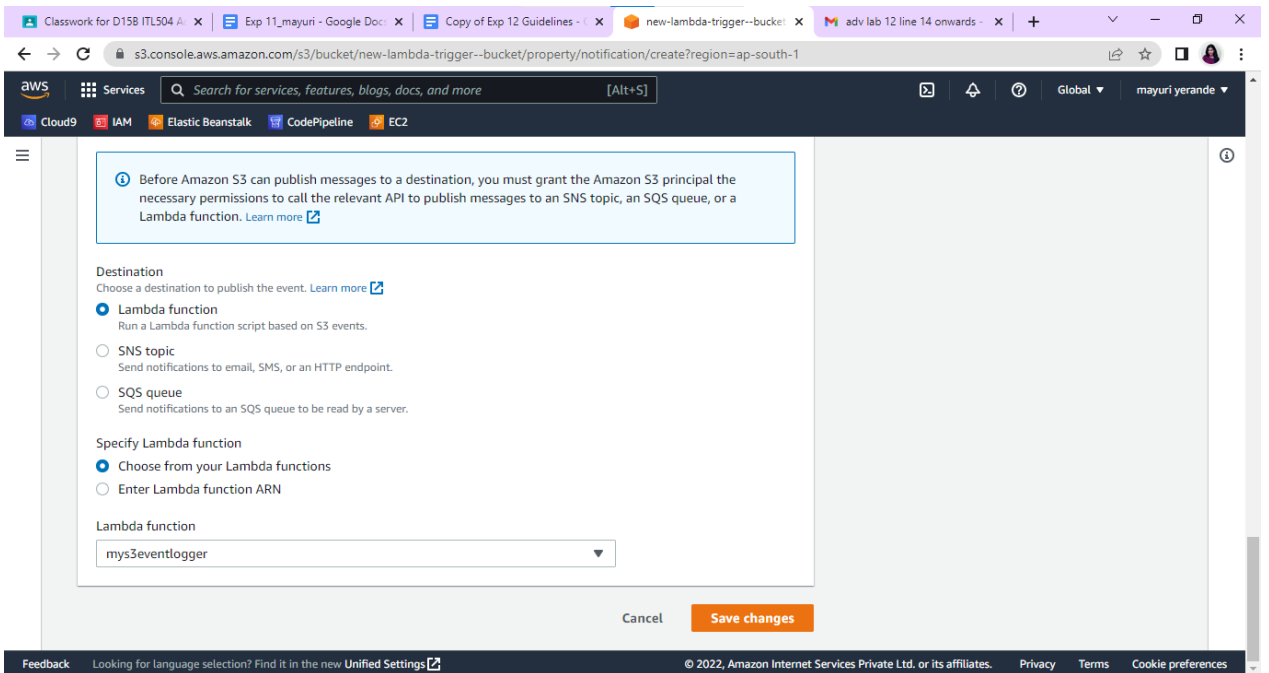
Click on Create Event Notification.

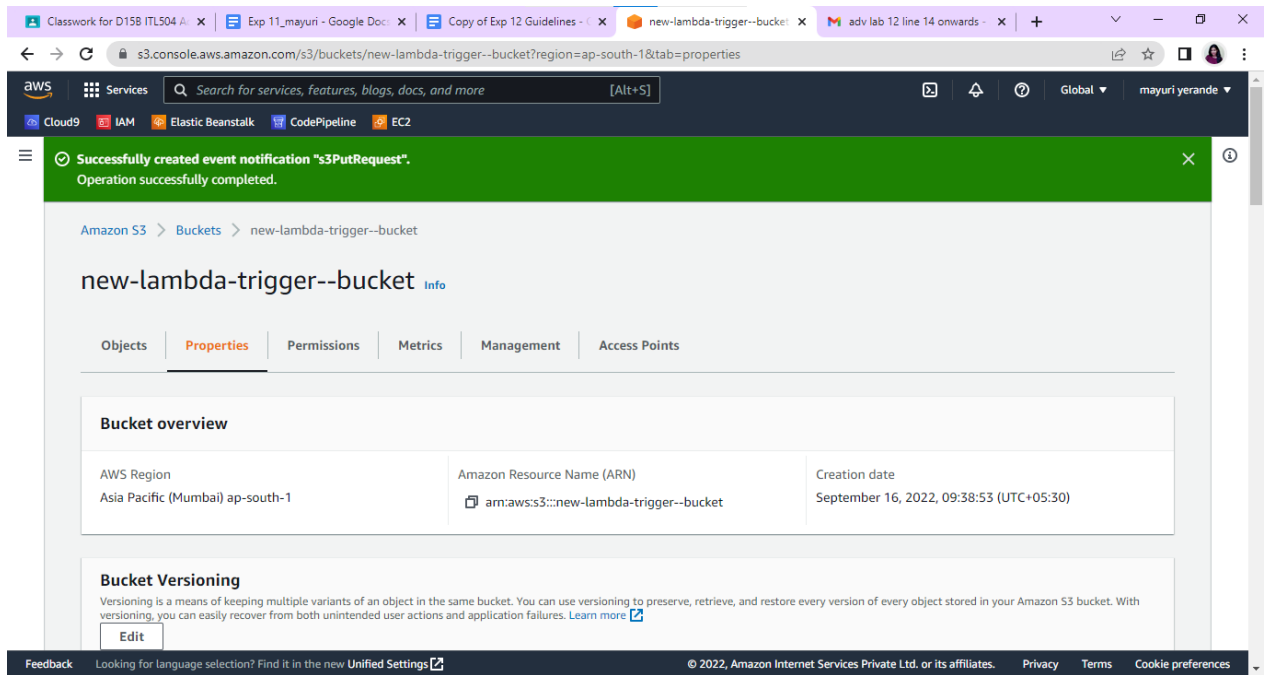
8. Mention an event name and check Put under event types.





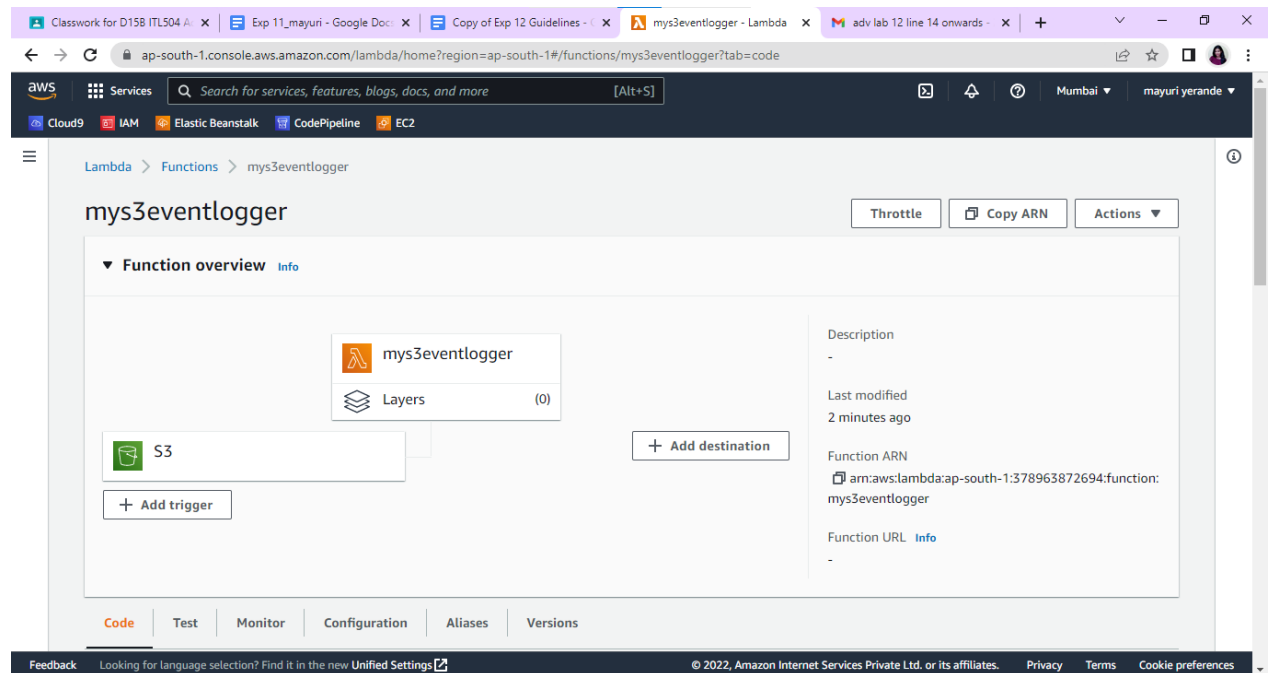
You can optionally choose .json under the suffix since the code only accepts JSON.





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

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



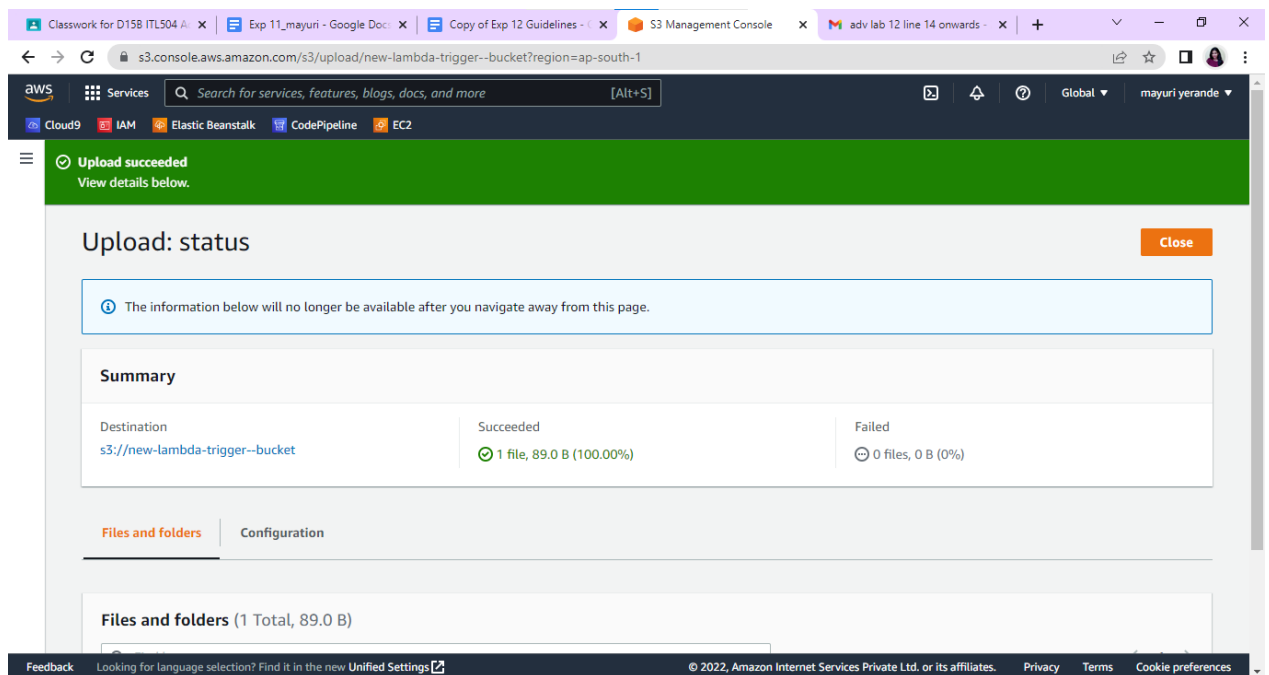
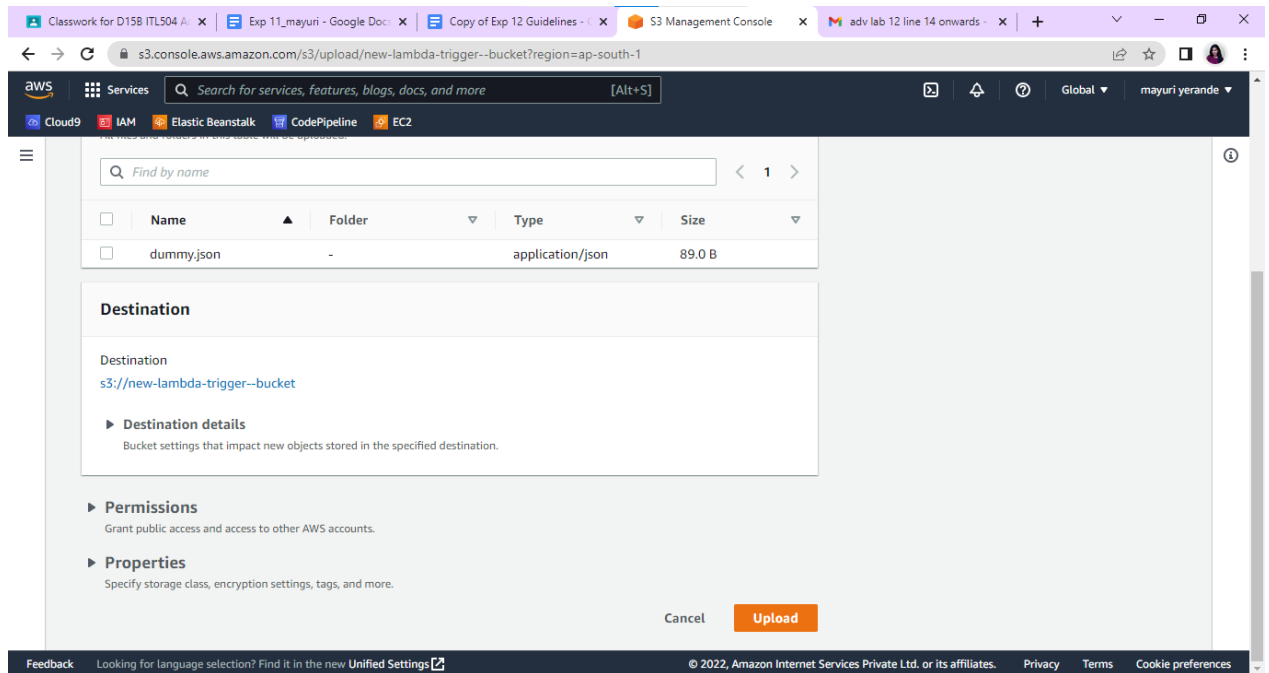
10. Now, create a dummy JSON file locally.

Dummy.json

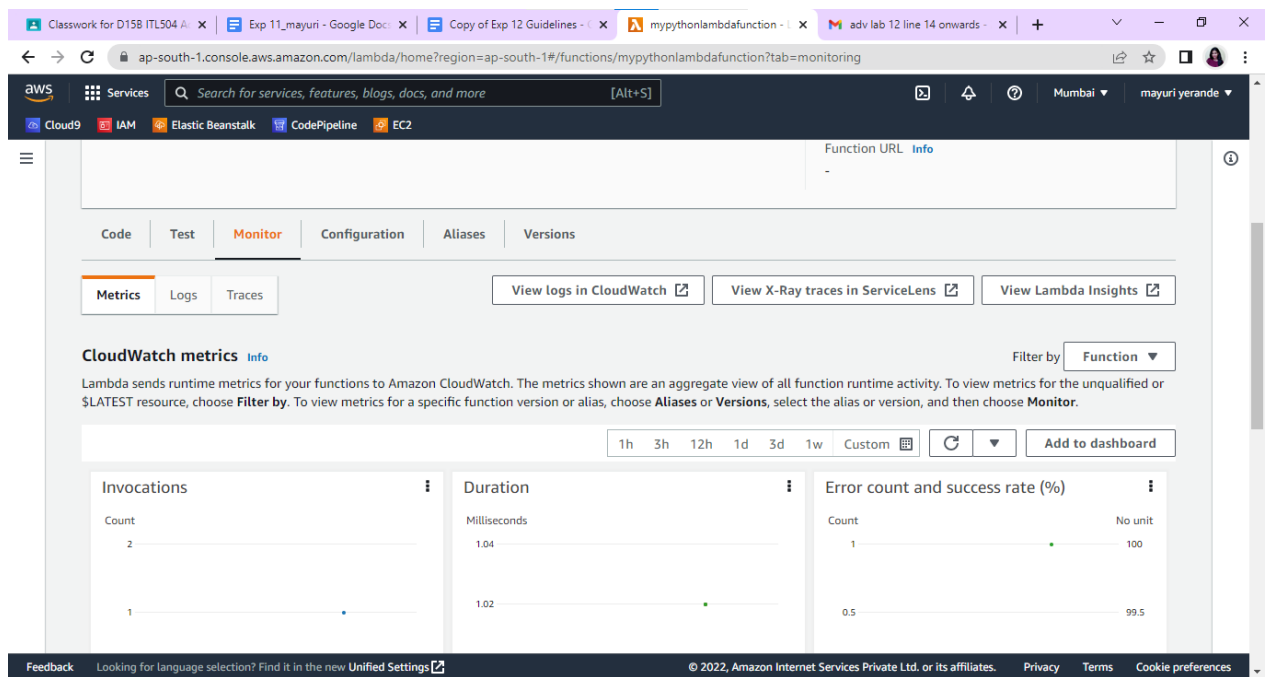
```
{  
  
  "id":70,  
  
  "name":"Mayuri Yerande",  
  
  "Designation":"Student",  
  
  "Class" :D15B  
  
}
```

11. Go back to your S3 Bucket and click on Add Files to upload a newfile.

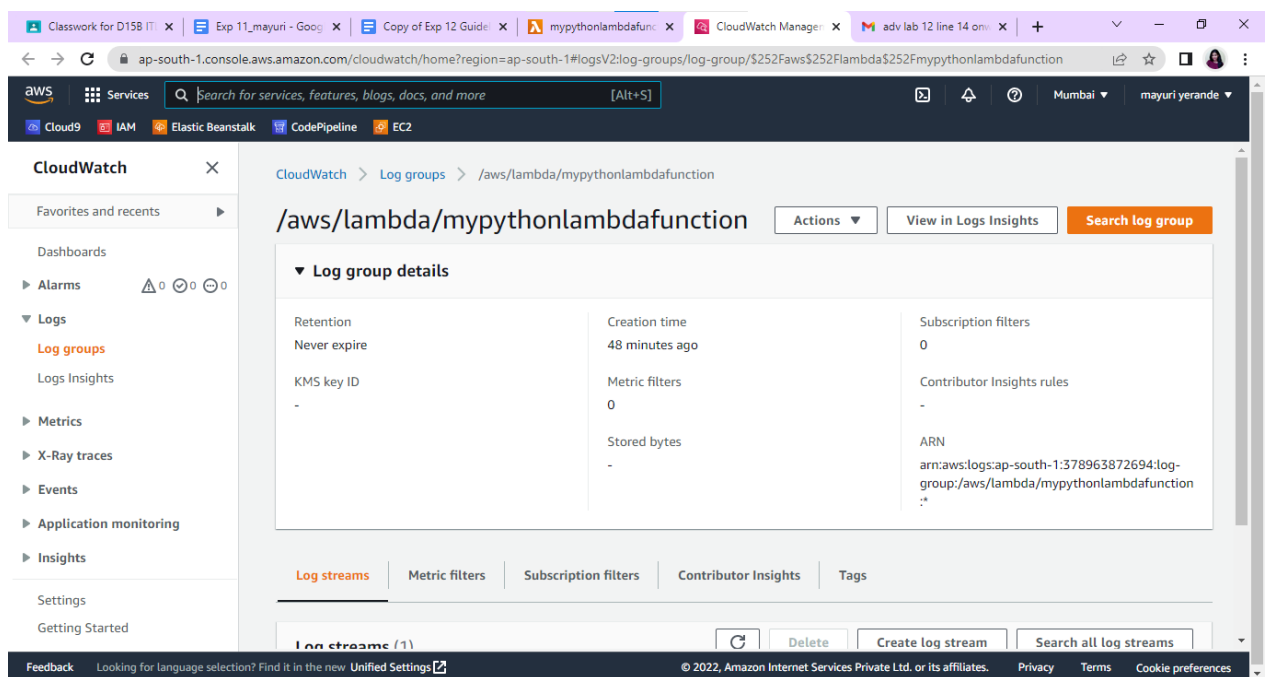
12. Select the dummy data file from your computer and click Upload.



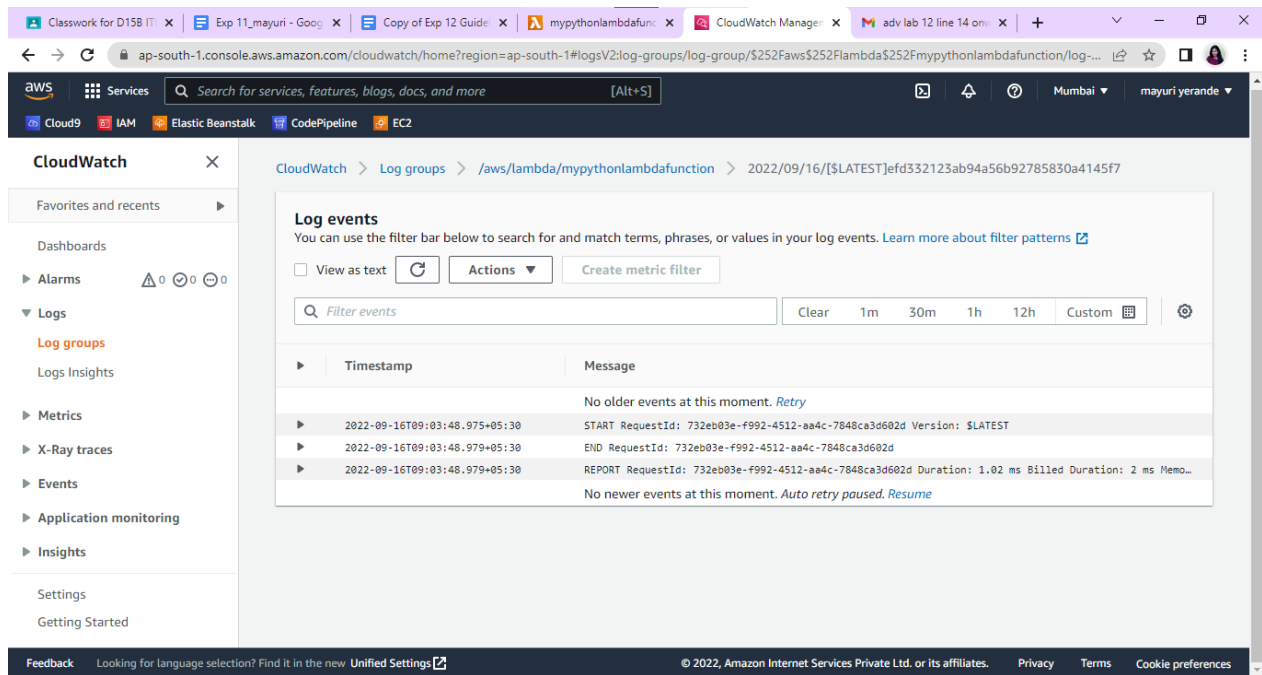
13. Go back to your Lambda function and check the Monitor tab.



Under Metrics, click on View logs in Cloudwatch to check the Function logs



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



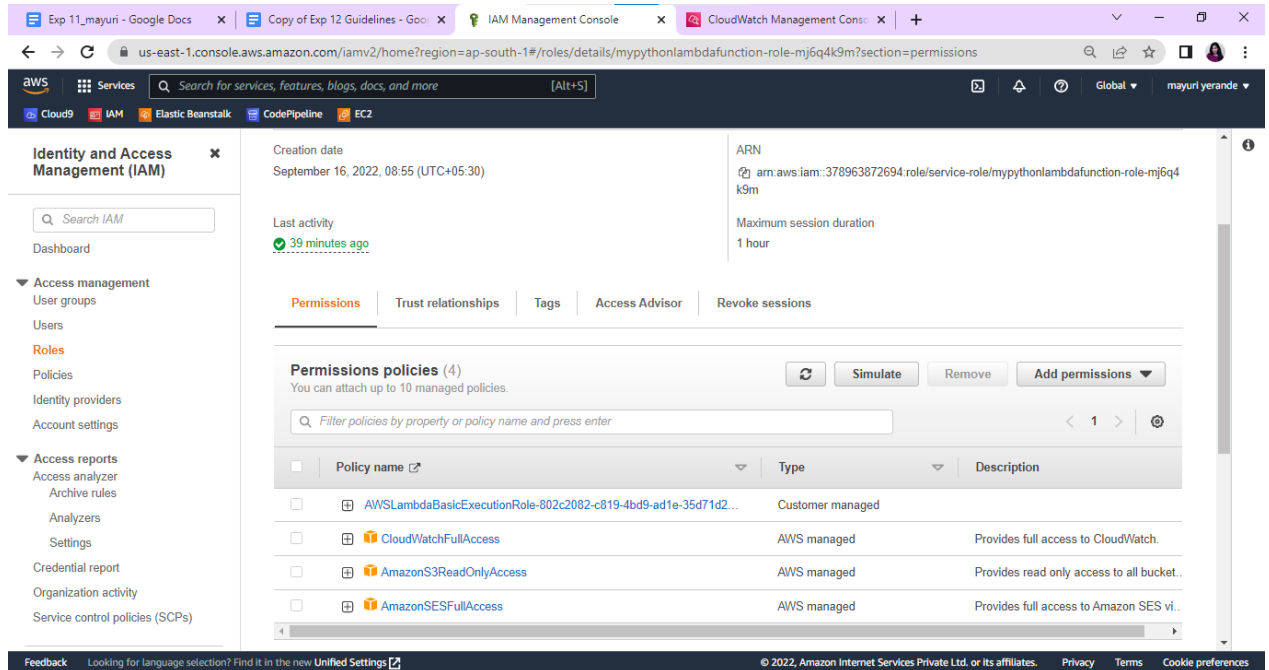
As you can see, our function logged that a file was uploaded with its file name and the bucket to which it was uploaded. It also mentions the contents inside the file as our function was defined to.

Hence, we have successfully created a Python function inside AWS Lambda which logs every time an object is uploaded to an S3 Bucket.

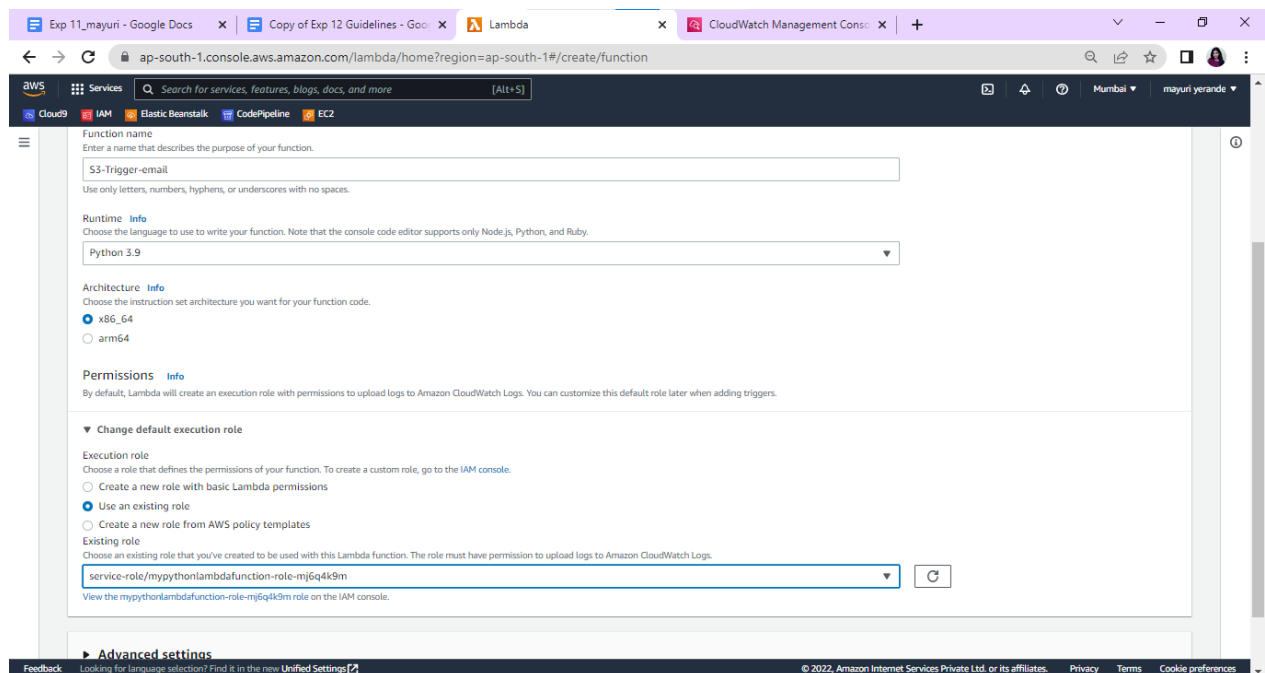
Part 2

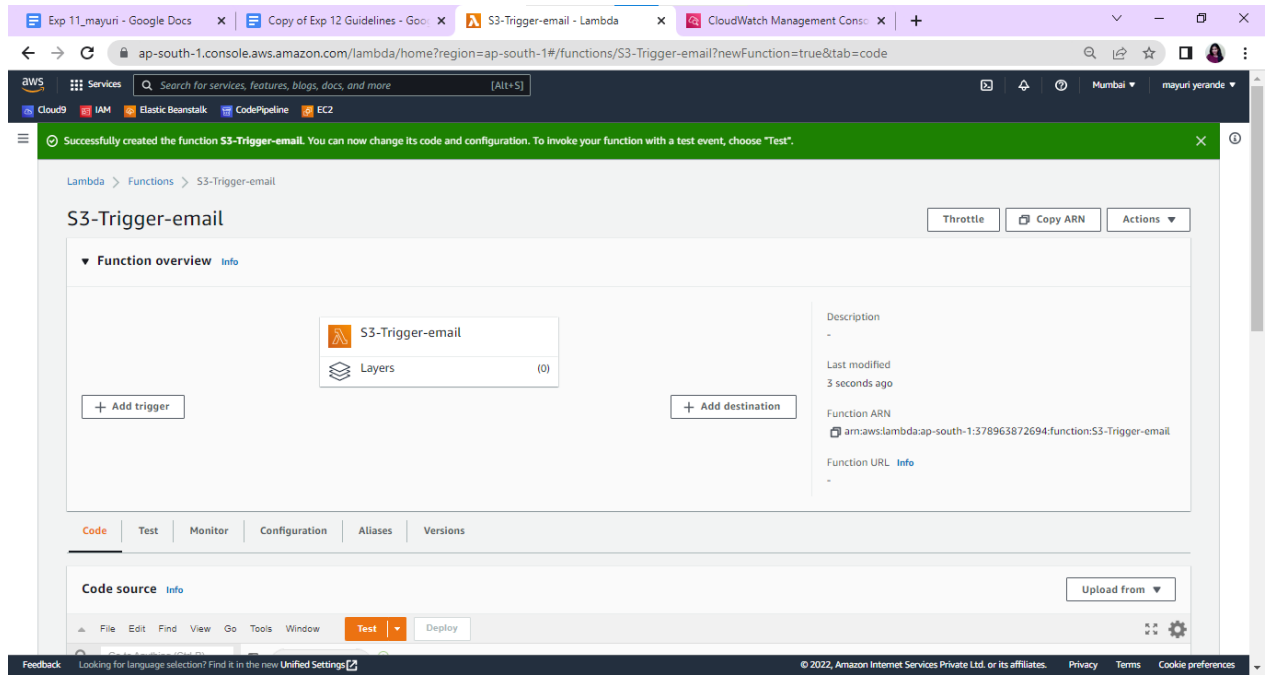
Sending an Email on Bucket additions to Bucket

1. Go to the IAM console and edit the same Lambda Role. This time, add SESFullAccess Permission to the role.



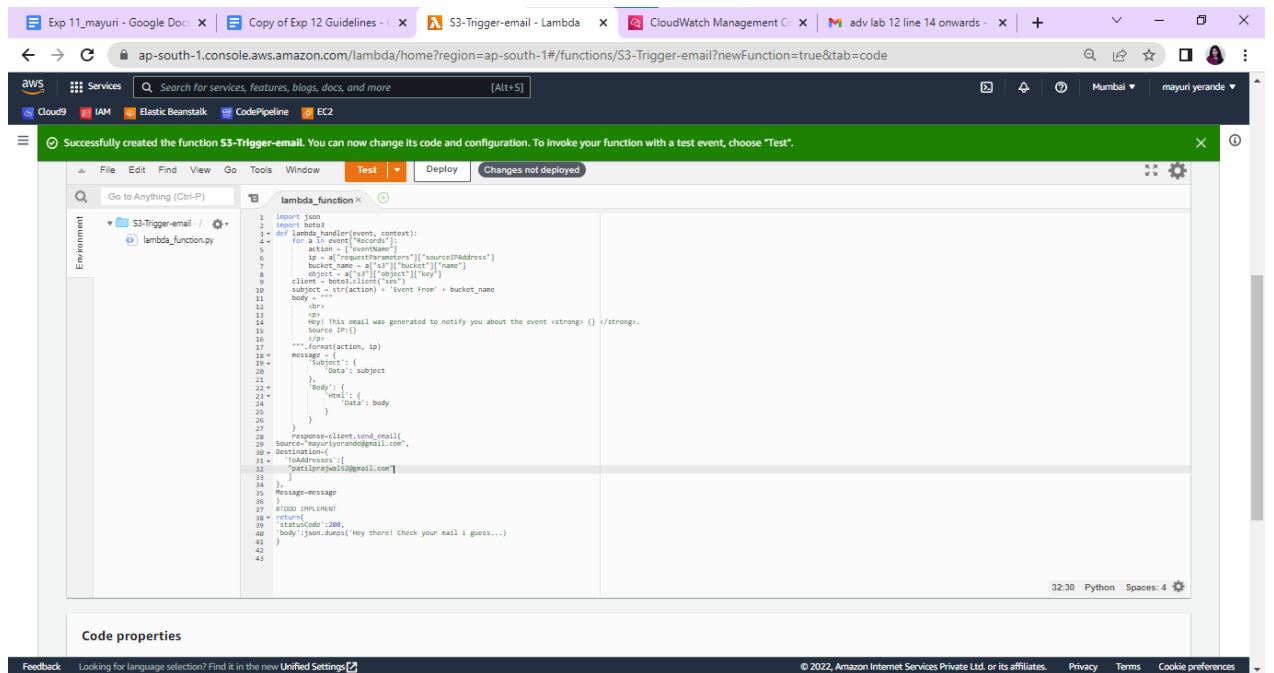
2. Create a new Lambda function in a Python environment. Use the existing role which was previously created.



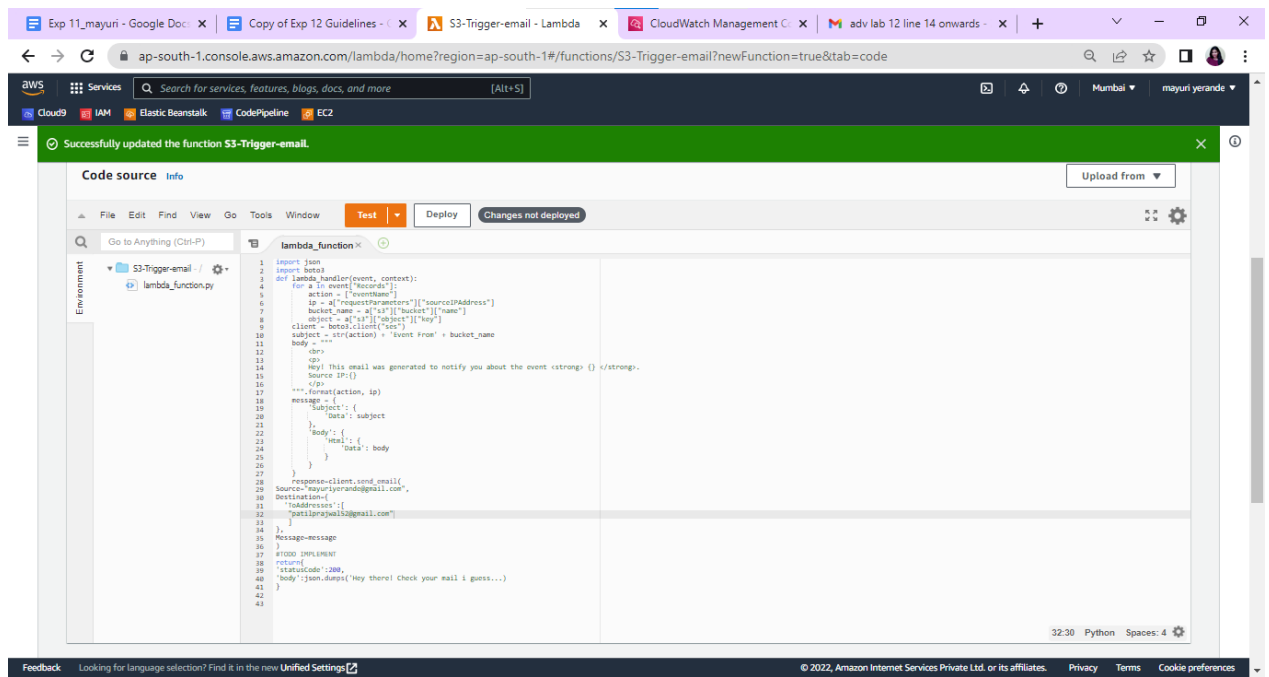


3. In this function, above the default hello-world TODO, add the following code.

This code is basically to send an email on the creation of an object in the attached S3 Bucket. It sends the bucket name, event and source IP address.



In this code, modify the **Source** and **Destination ToAddresses** to your sender and receiver email addresses. Once done, deploy the function.



The screenshot shows the AWS Lambda console interface for the function 'S3-Trigger-email'. The code is written in Python and uses the boto3 library to send an email via SMTP. The code is as follows:

```
1 import json
2 import boto3
3 def lambda_handler(event, context):
4     for a in event['Records']:
5         action = a['eventName']
6         ip = a['requestParameters']['sourceIPAddress']
7         bucket_name = a['s3']['bucket']['name']
8         object = a['s3']['object']['key']
9         client = boto3.client('ses')
10        subject = str(action) + 'Event from' + bucket_name
11        body = ""
12        cbr
13        cbr
14        cbr
15        Source IP:()
16        cbr
17        cbr
18        cbr
19        cbr
20        cbr
21        cbr
22        cbr
23        cbr
24        cbr
25        cbr
26        cbr
27        cbr
28        response=client.send_email(
29            Source='mayuri.yerande@gmail.com',
30            Destination={
31                'ToAddresses':(
32                    "patilprajwal2@gmail.com"
33                )
34            },
35            Message={
36                'Subject': subject
37            },
38            Body={
39                'Text': {
40                    'Data': body
41                }
42            }
43        )
44        return {
45            'statusCode': 200,
46            'body': json.dumps('Hey there! Check your mail i guess...')
47        }
```

The code is displayed in the 'Code source' tab of the AWS Lambda console. The console also shows a green notification bar at the top stating 'Successfully updated the function S3-Trigger-email.' and a 'Deploy' button. The bottom of the console shows the footer with '© 2022, Amazon Internet Services Private Ltd. or its affiliates.' and links for 'Privacy', 'Terms', and 'Cookie preferences'.

4. Open up the SES Console and click on Manage Email Addresses.
5. Choose Verify Email Address and verify both sender and receiver email addresses
Click on the verification links you are sent and verify the emails.

The screenshot shows the Amazon SES console interface. The left sidebar contains navigation links: Account dashboard, Reputation metrics, SMTP settings, Configuration, Verified identities (selected), Configuration sets, Dedicated IPs, Email templates, Suppression list, and Cross-account notifications. The main content area displays the verification status for the email address **mayuriyerande@gmail.com**. At the top right of the main area are buttons for 'Delete' and 'Send test email'. Below this is a 'Legacy TXT records' section with an information icon and text explaining DKIM. A 'Summary for mayuriyerande@gmail.com' section follows, containing a table with verification details.

Identity status	Amazon Resource Name (ARN)	AWS Region
Verified	arn:aws:ses:ap-south-1:378963872694:identity/mayuriyerande@gmail.com	Asia Pacific (Mumbai)

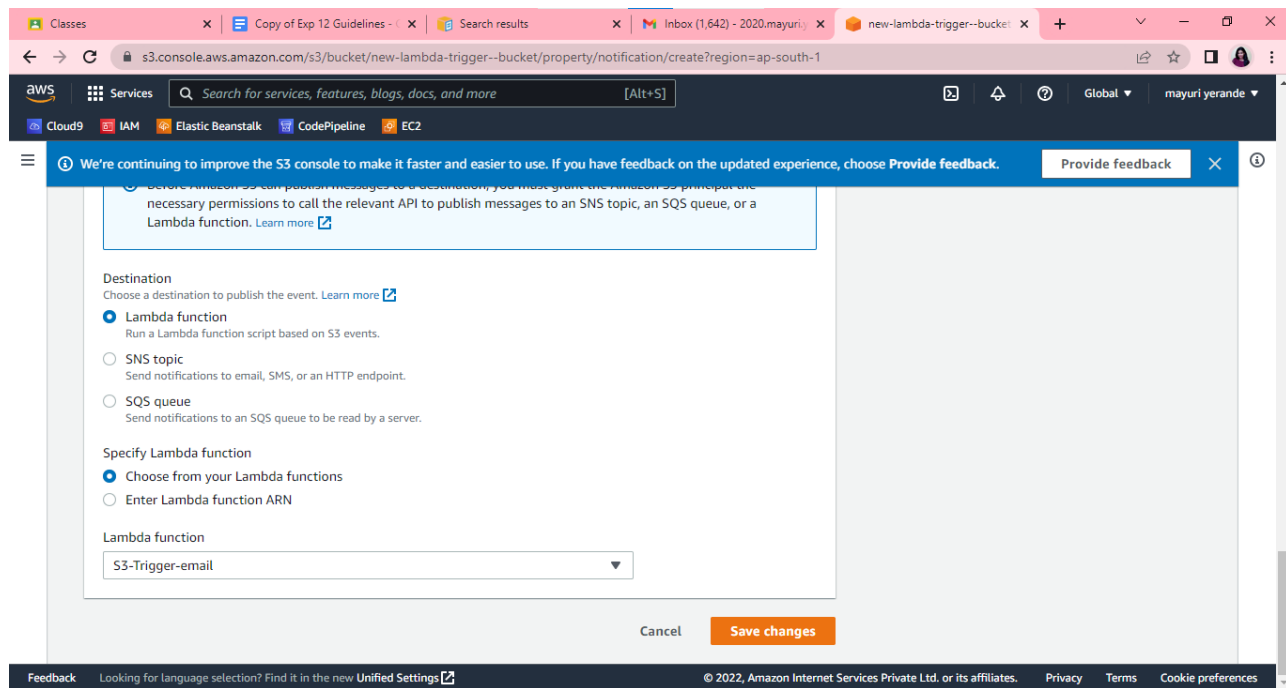
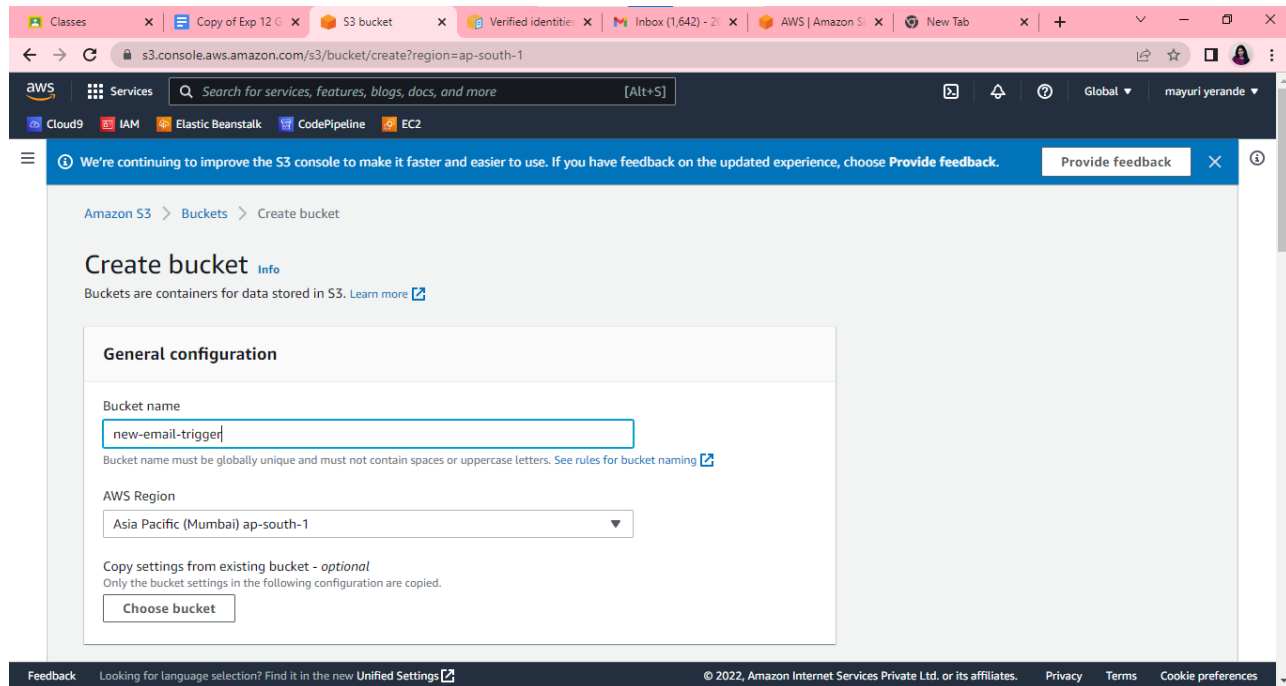
The footer of the console shows a feedback link, a language selection notice, a copyright notice for 2022, and links for Privacy, Terms, and Cookie preferences.

This screenshot shows the Amazon SES console for the email address **patilprajwal52@gmail.com**. The interface is similar to the previous one, with the 'Verified identities' section selected in the sidebar. The main content area shows the verification status for this email address, including a 'Legacy TXT records' section and a 'Summary for patilprajwal52@gmail.com' table.

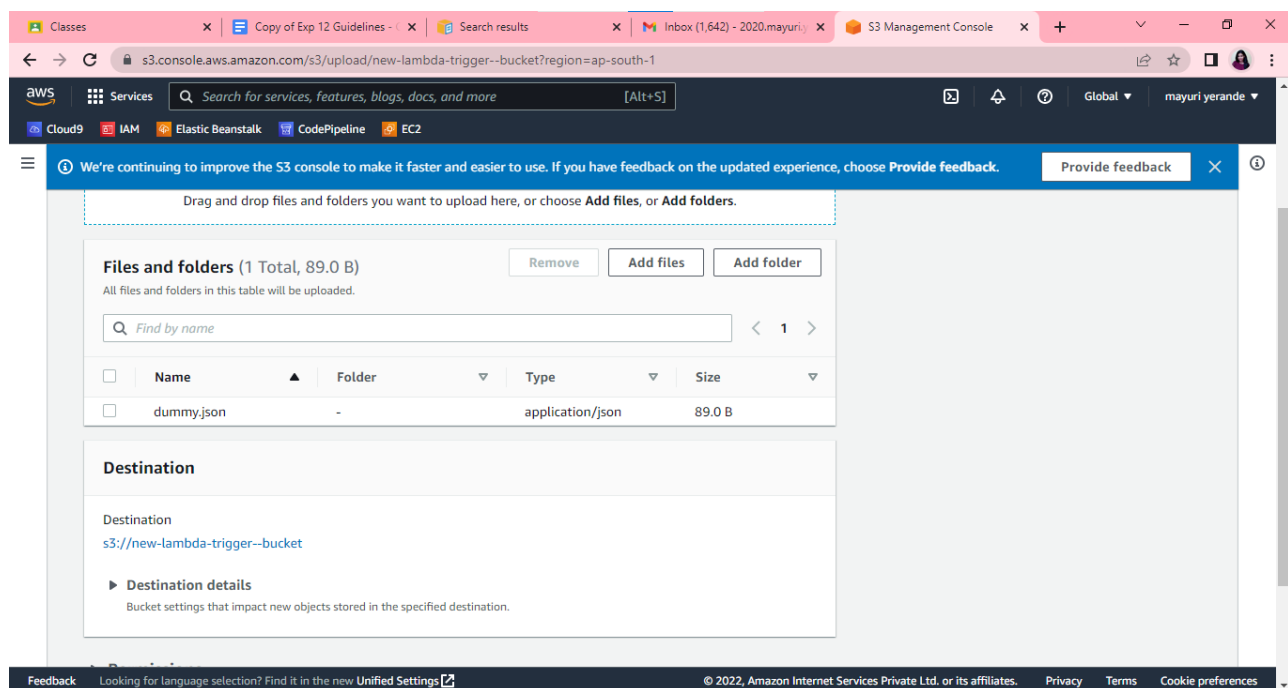
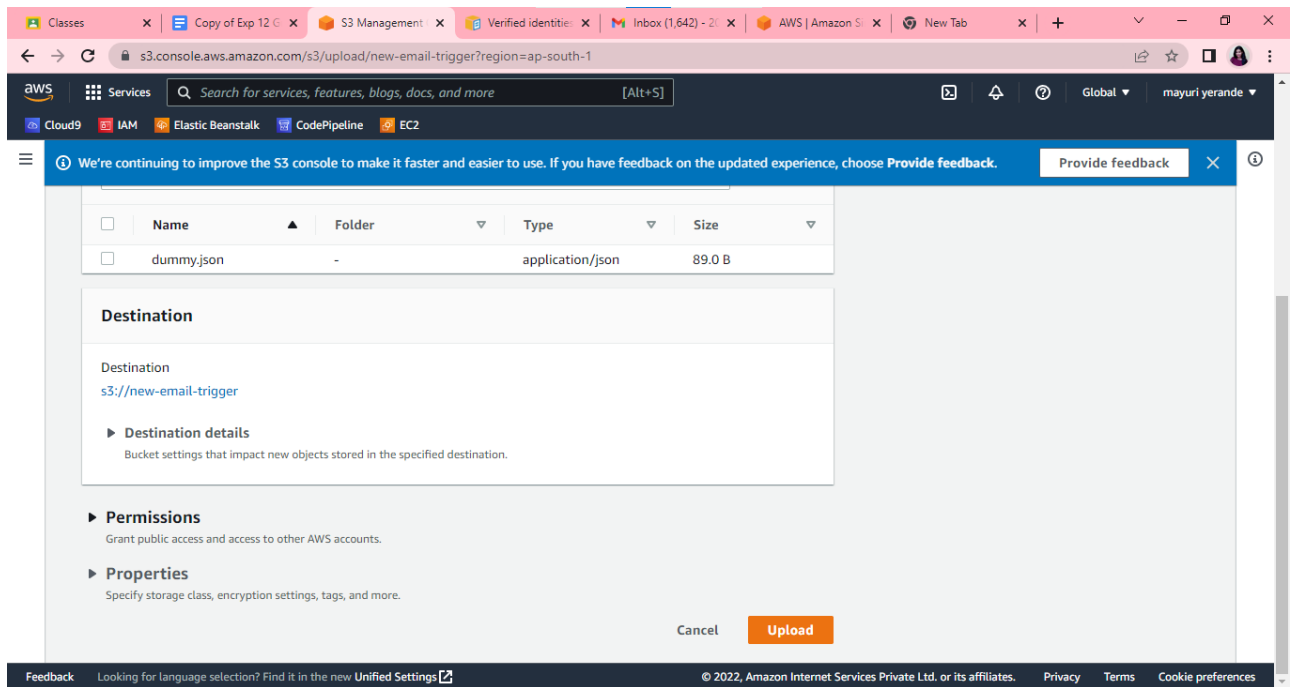
Identity status	Amazon Resource Name (ARN)	AWS Region
Verified	arn:aws:ses:ap-south-1:378963872694:identity/patilprajwal52@gmail.com	Asia Pacific (Mumbai)

The footer of the console is identical to the first screenshot, showing feedback, language selection, copyright, and policy links.

- Now, open up the S3 Console, create a new bucket as you did previously and add an event notification inside events and attach it to your Lambda function.



7. Once that's done, upload any file to your S3 Bucket. I'll upload the same dummy JSON file again



Check your **ToAddress** email. You'll receive an email from the Source Address via Amazon SES. In this way, we successfully created a function in AWS Lambda that sends an email on uploading an object to an S3 Bucket using Amazon SES.

Conclusions:

In this experiment, we learned how to create AWS Lambda function to log every time an object is added to S3 bucket.