Name: Manav Jawrani

Roll No.: 19

Subject: Advanced DevOps

Experiment No.: 12

Experiment 12

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

Theory:

What are the features of Lambda function?
The following key features help you develop Lambda applications that are scalable, secure and easily extensible:

1. Concurrency and scaling controls:
Using concurrency settings to ensure that your production applications are highly available and highly responsive. Lambda manages the infrastructure that runs that code, and scales automatically in response to incoming requests.

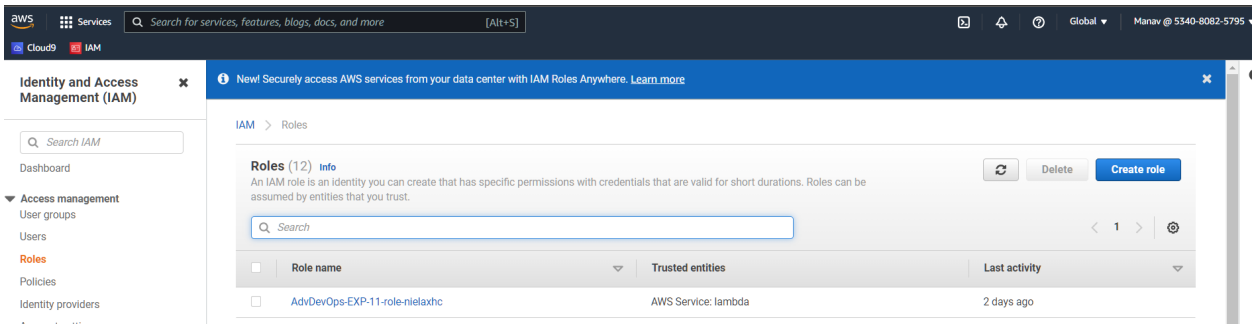
2. Function URLs:
Lambda offers built-in HTTP(S) endpoint support through function URLs. With function URLs, you can assign a dedicated HTTP endpoint to your Lambda function. When your function URL is configured you can use it to invoke your function through a web browser, cURL, postman or any HTTP client. You can also add a function URL to an existing function.

3. Event source mappings:
Event source mappings maintain a local queue of unprocessed items and handle retries if the function returns an error or is throttled. You can configure an event source mapping to customize batching behavior and error handling or to send a record of items that fail processing to a destination.

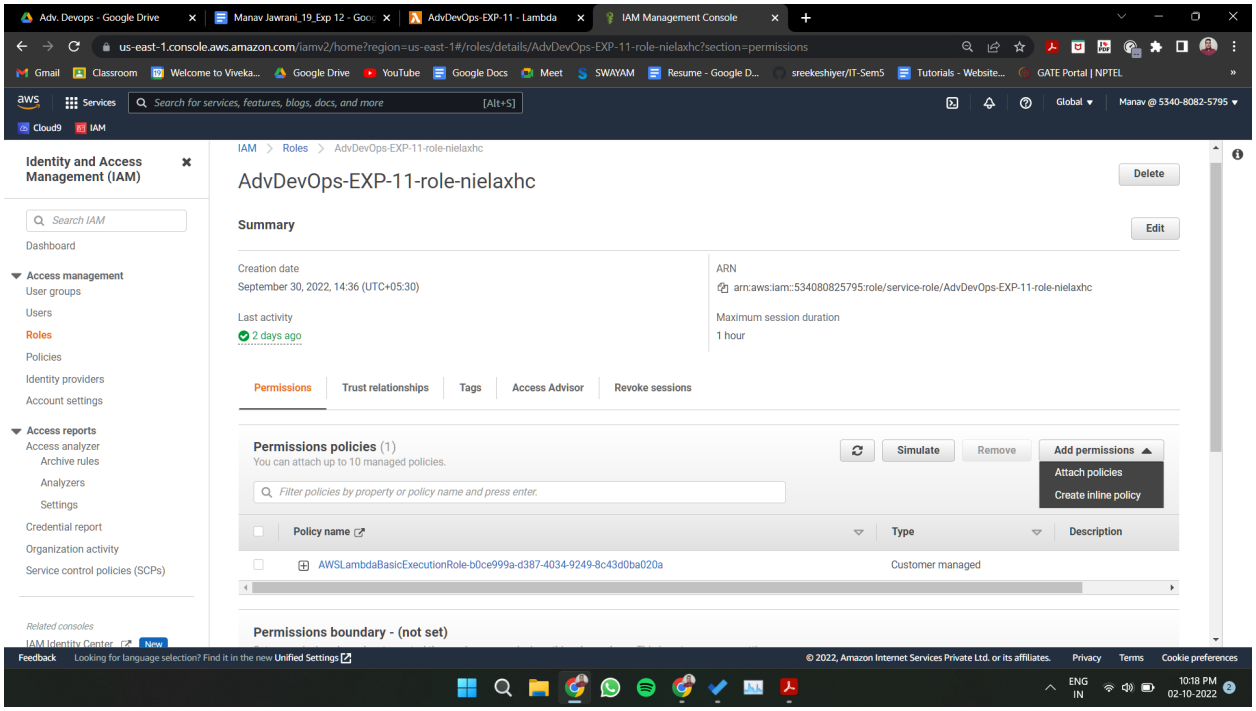
4. Testing and deployment tools:
Lambda supports deploying code as is or as container images. You can use a rich tools ecosystem for authoring, building and deploying your Lambda functions using AWS and popular community tools like the Docker Command Line Interface (CLI).

Implementation:

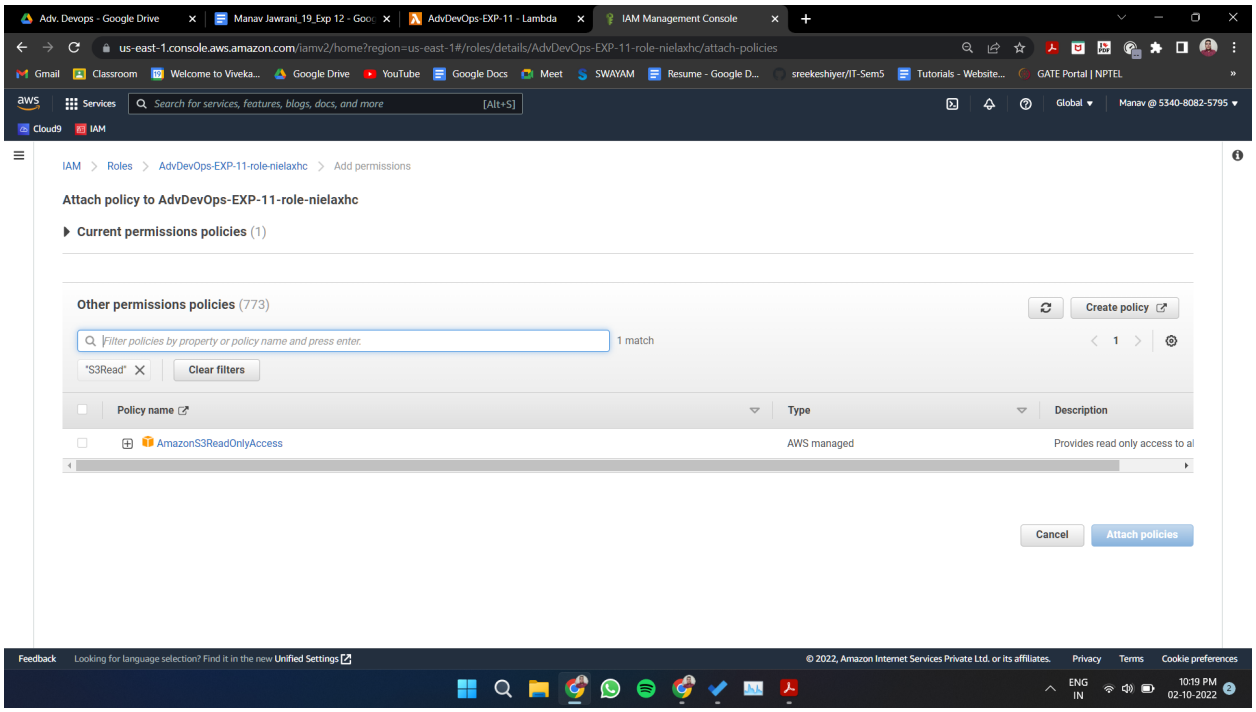
Step 1: Open up the IAM Console and under Roles, choose the Role we previously created for the Python Lambda Function (You can find your role name configuration of your Lambda function).



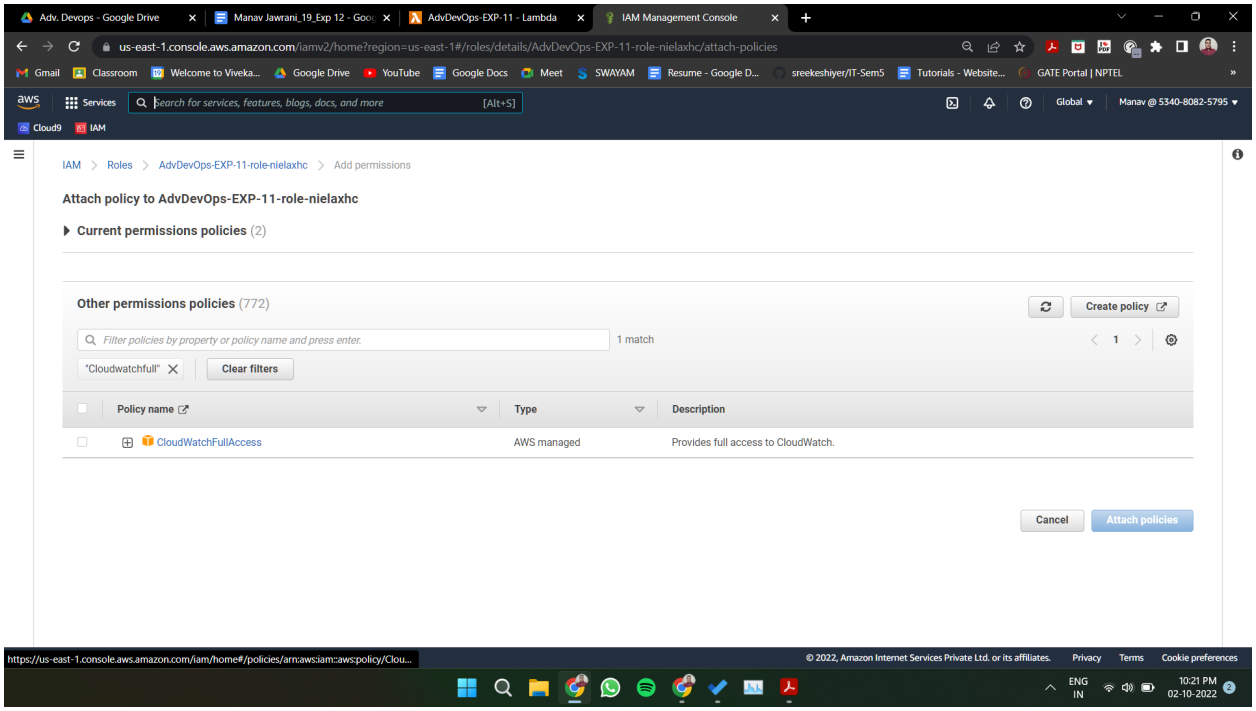
Step 2: Under Attach Policies, add S3-ReadOnly and CloudWatchFull permissions to this role.



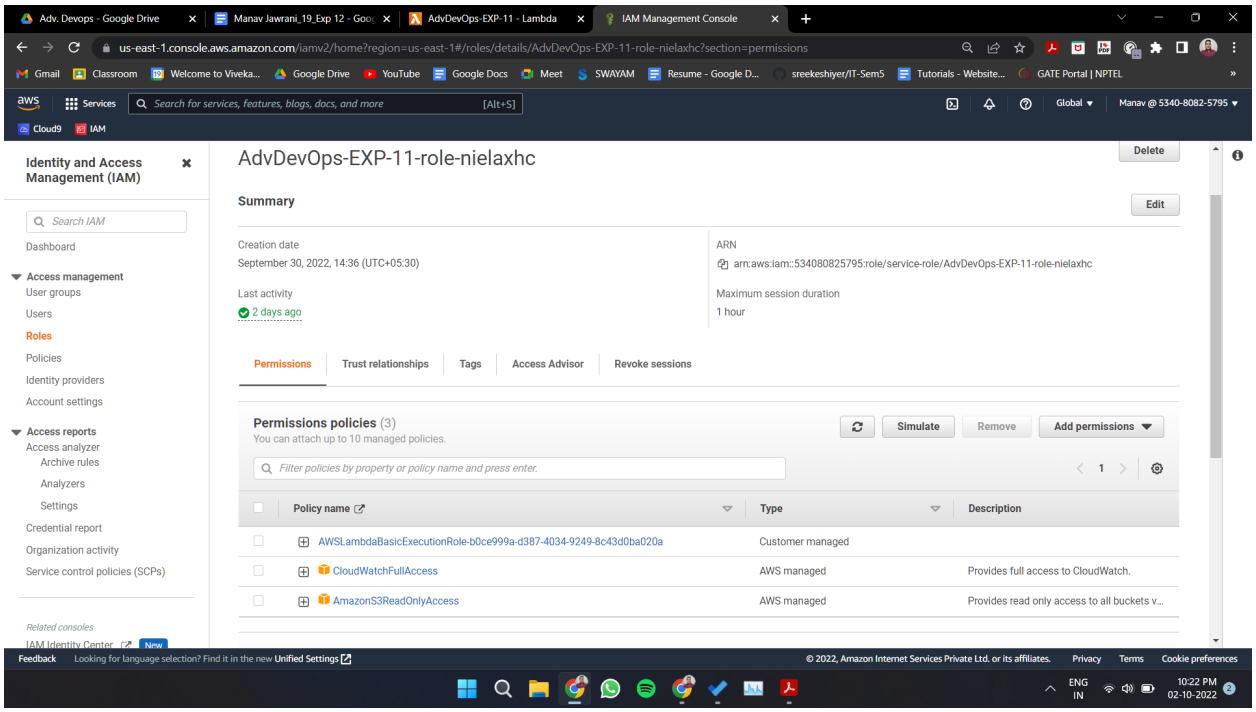
S3-ReadOnly



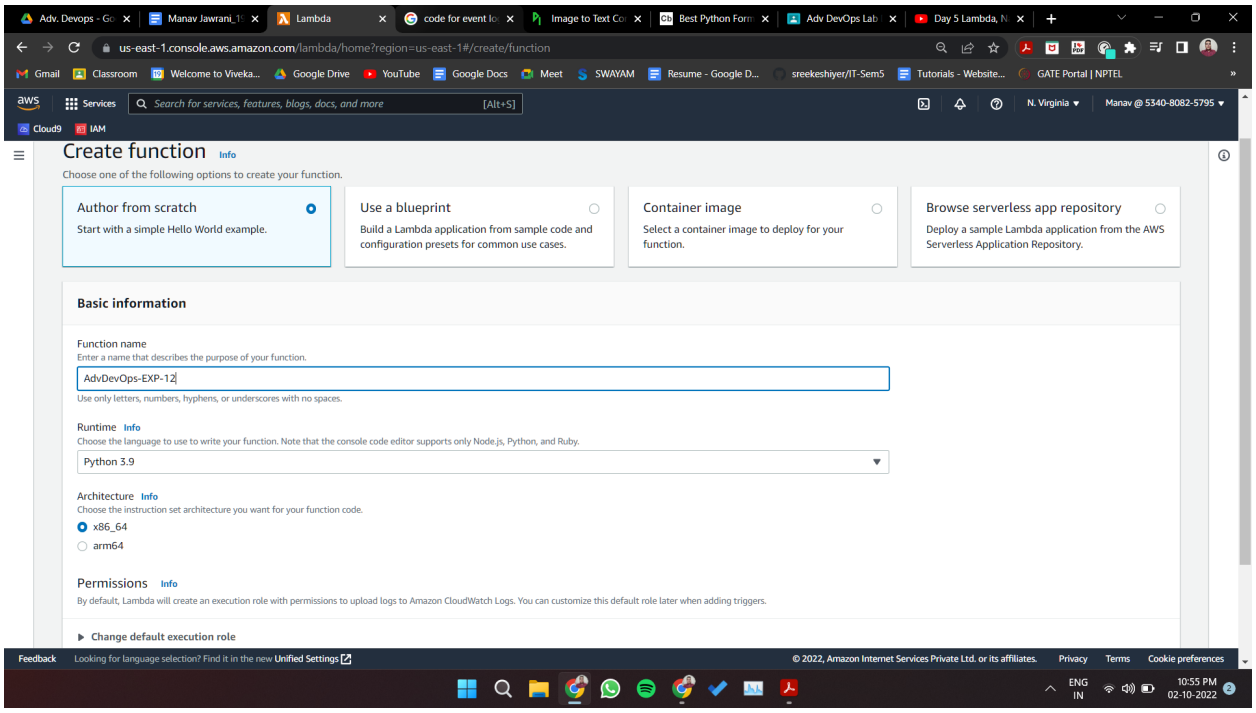
CloudWatchFull



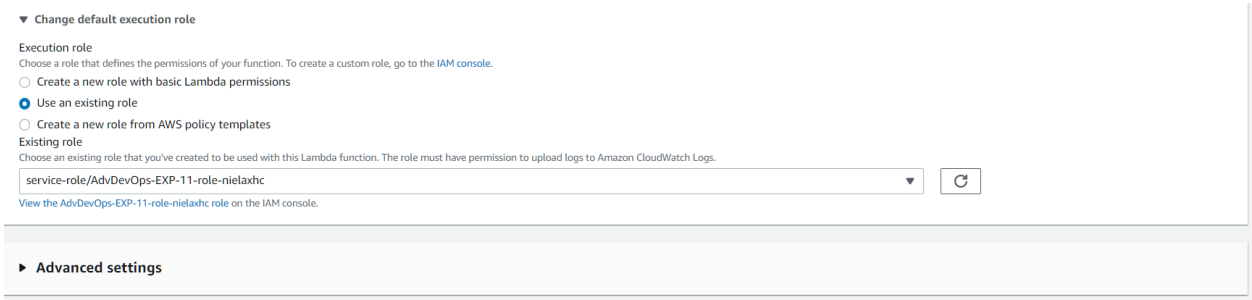
After successful attachment of policy you will see something like this you will be able to see the updated policies.



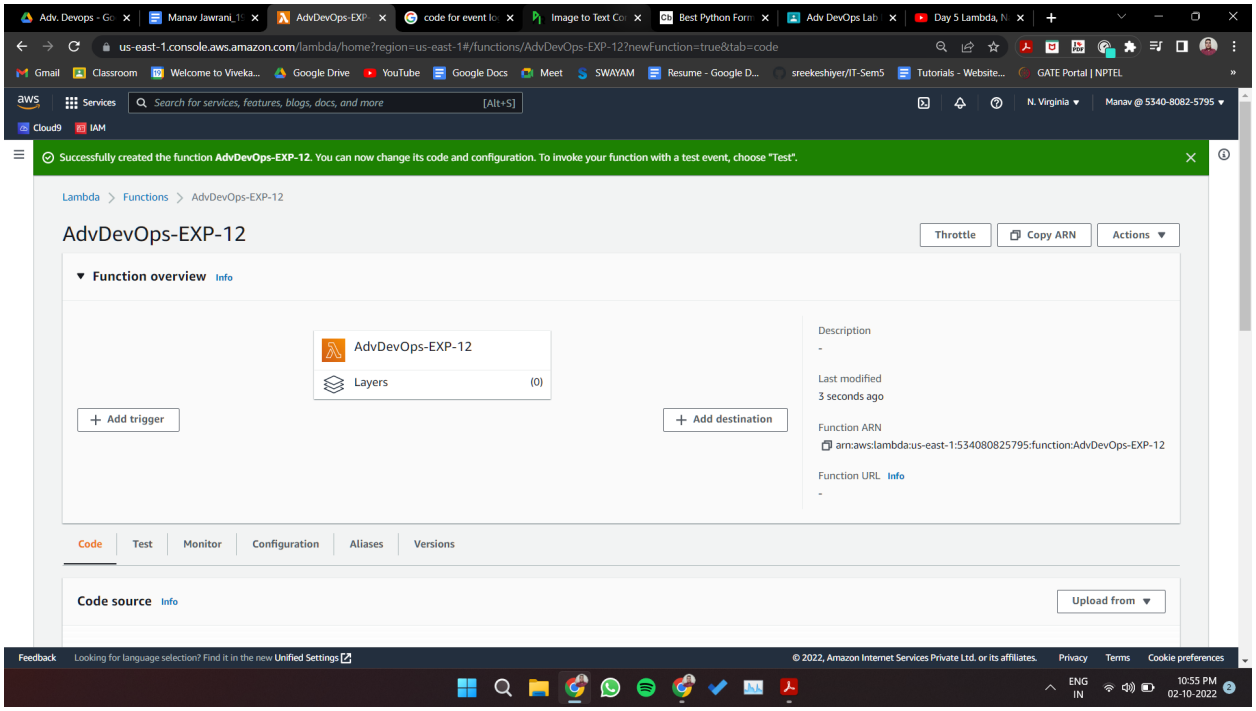
Step 3: Open up AWS Lambda and create a new Python function.



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



Step 4: The function is up and running.

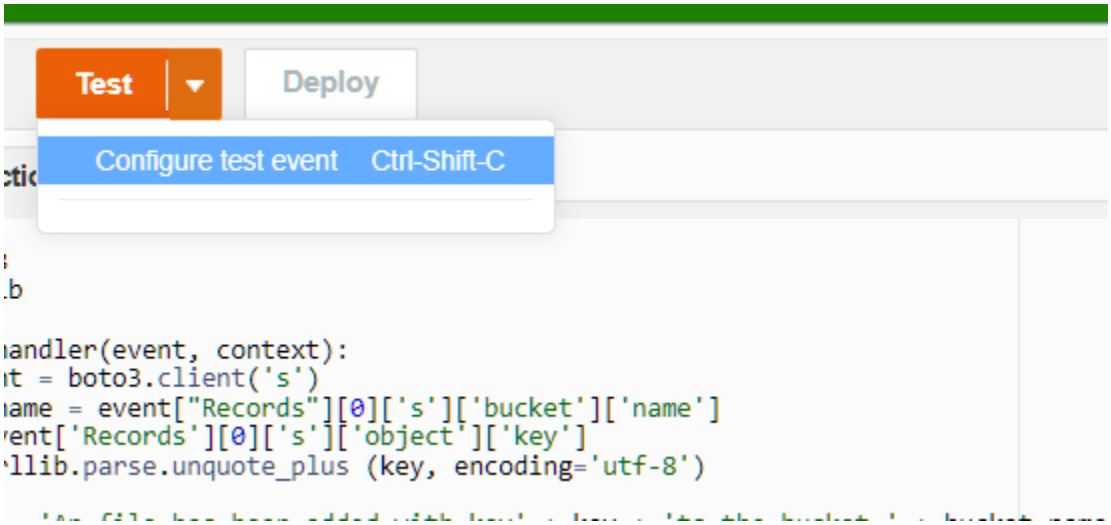


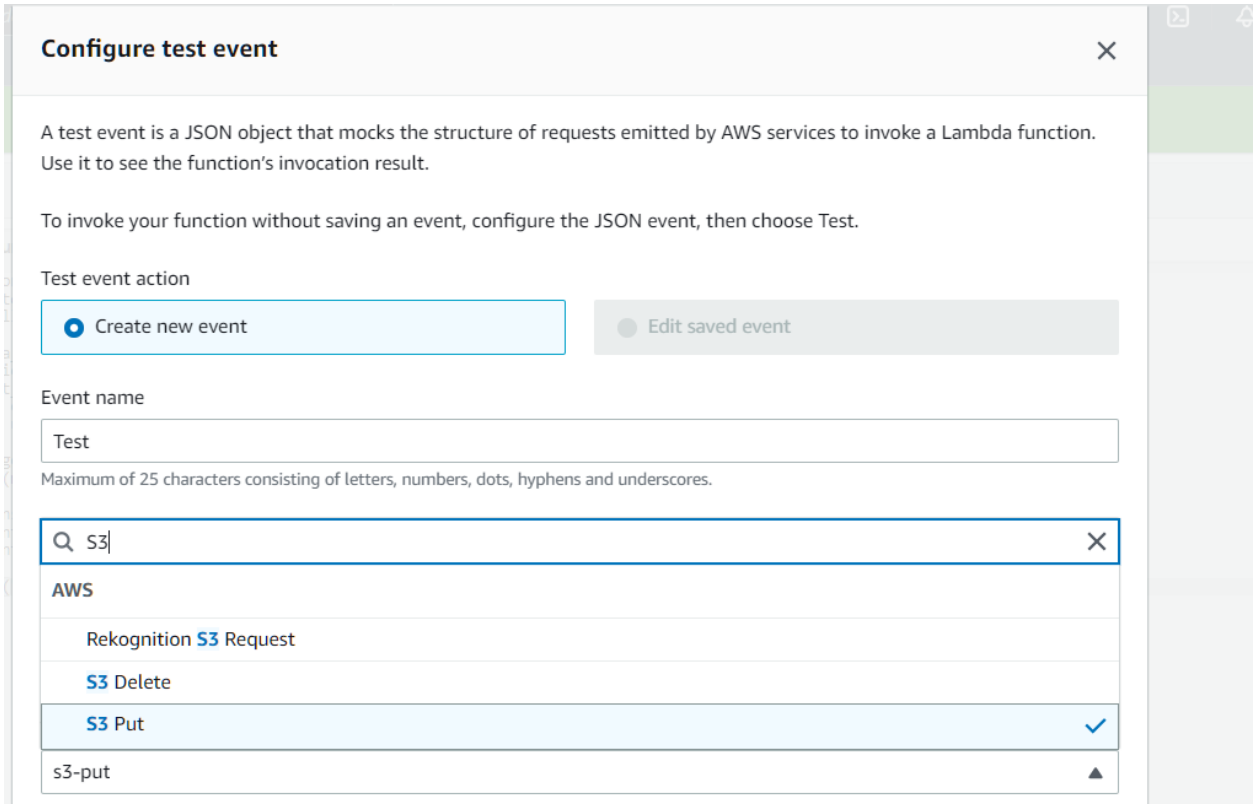
Step 5: Make the following changes to the function and click on the deploy button. This code basically logs a message and logs the contents of a JSON file which is uploaded to an S3 Bucket and then deploy the code.

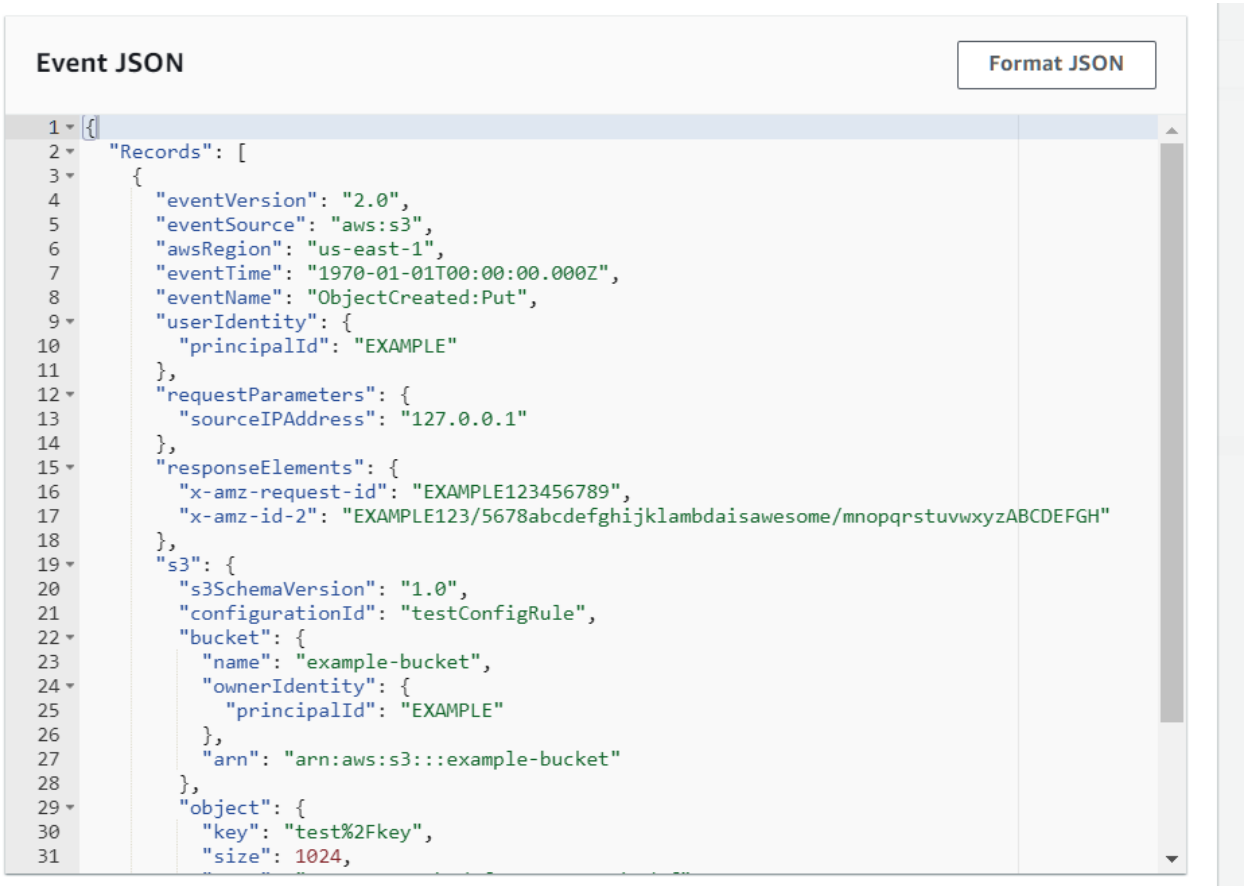
lambda_function

```
import json
import boto3
import urllib

def lambda_handler(event, context):

    s3_client = boto3.client('s3')
    bucket_name = event["Records"][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']
    key = urllib.parse.unquote_plus(key, encoding='utf-8')

    message = 'An file has been added with key  ' + key + 'to the bucket ' + bucket_name
    print(message)

    response = s3_client.get_object(Bucket=bucket_name, Key=key)
    contents = response["Body"].read().decode()
    contents = json.loads(contents)

    print("These are the Contents of the File: \n", contents)
```

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

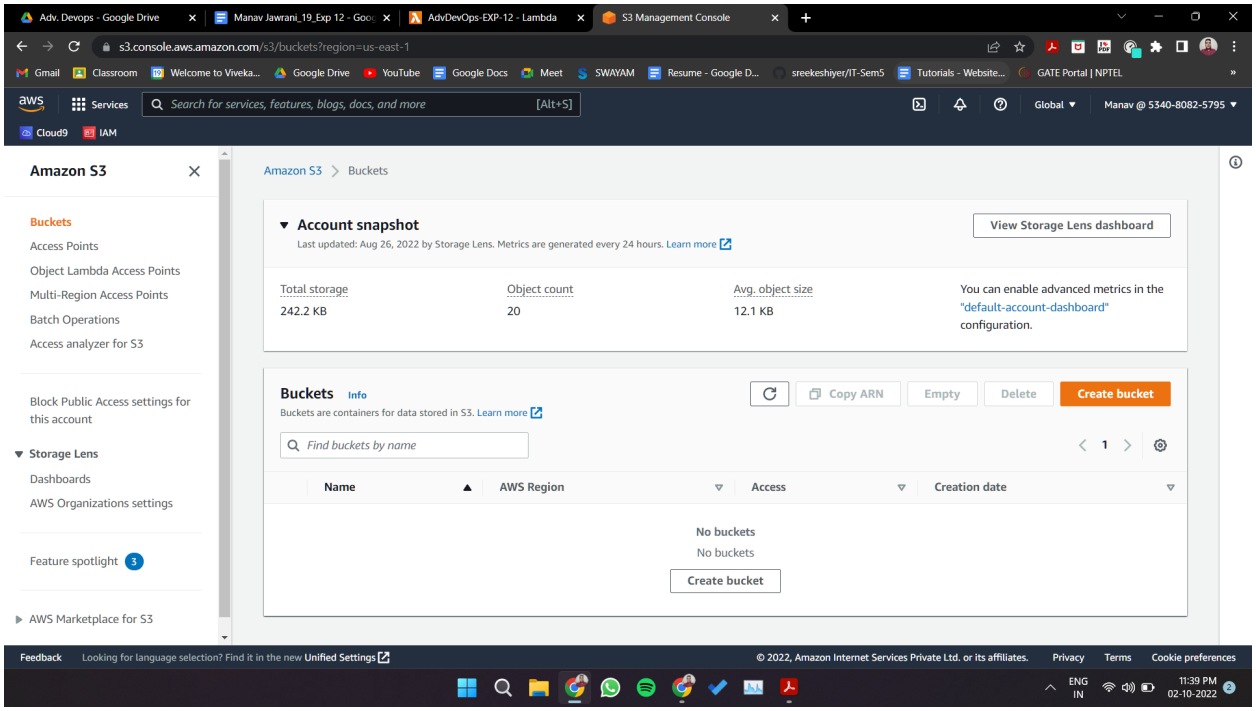



Configure test event
A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.
To invoke your function without saving an event, configure the JSON event, then choose Test.
Test event action
Create new event
Edit saved event
Event name
Test
Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.
S3
AWS
Rekognition S3 Request
S3 Delete
S3 Put
s3-put

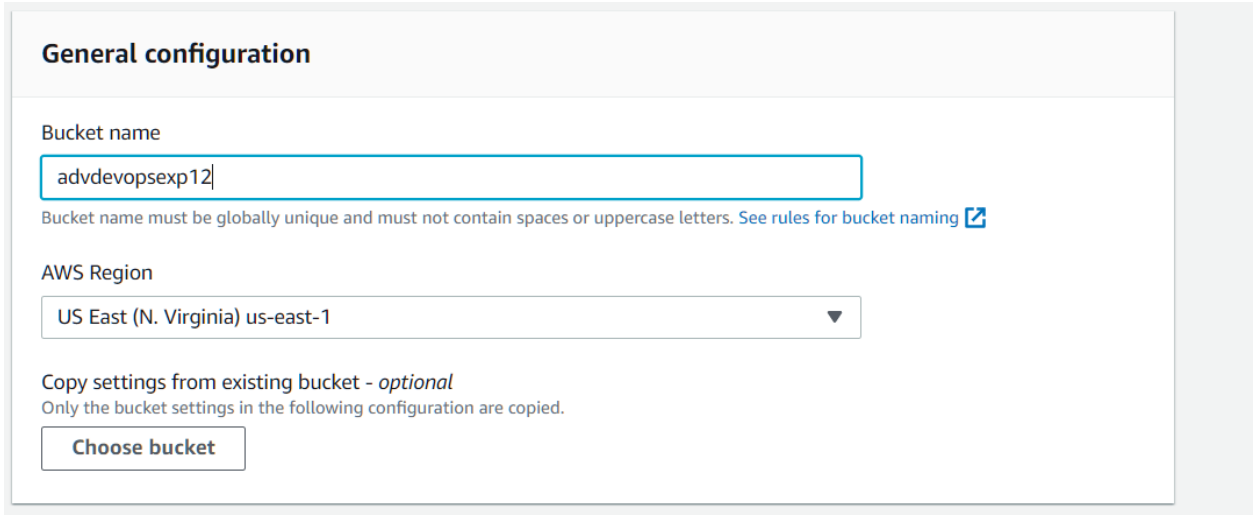

Event JSON
Format JSON
{
"Records": [
{
"eventVersion": "2.0",
"eventSource": "aws:s3",
"awsRegion": "us-east-1",
"eventTime": "1970-01-01T00:00:00.000Z",
"eventName": "ObjectCreated:Put",
"userIdentity": {
"principalId": "EXAMPLE"
},
"requestParameters": {
"sourceIPAddress": "127.0.0.1"
},
"responseElements": {
"x-amz-request-id": "EXAMPLE123456789",
"x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
},
"s3": {
"s3SchemaVersion": "1.0",
"configurationId": "testConfigRule",
"bucket": {
"name": "example-bucket",
"ownerIdentity": {
"principalId": "EXAMPLE"
},
"arn": "arn:aws:s3:::example-bucket"
},
"object": {
"key": "test%2Fkey",
"size": 1024,

And Save it.

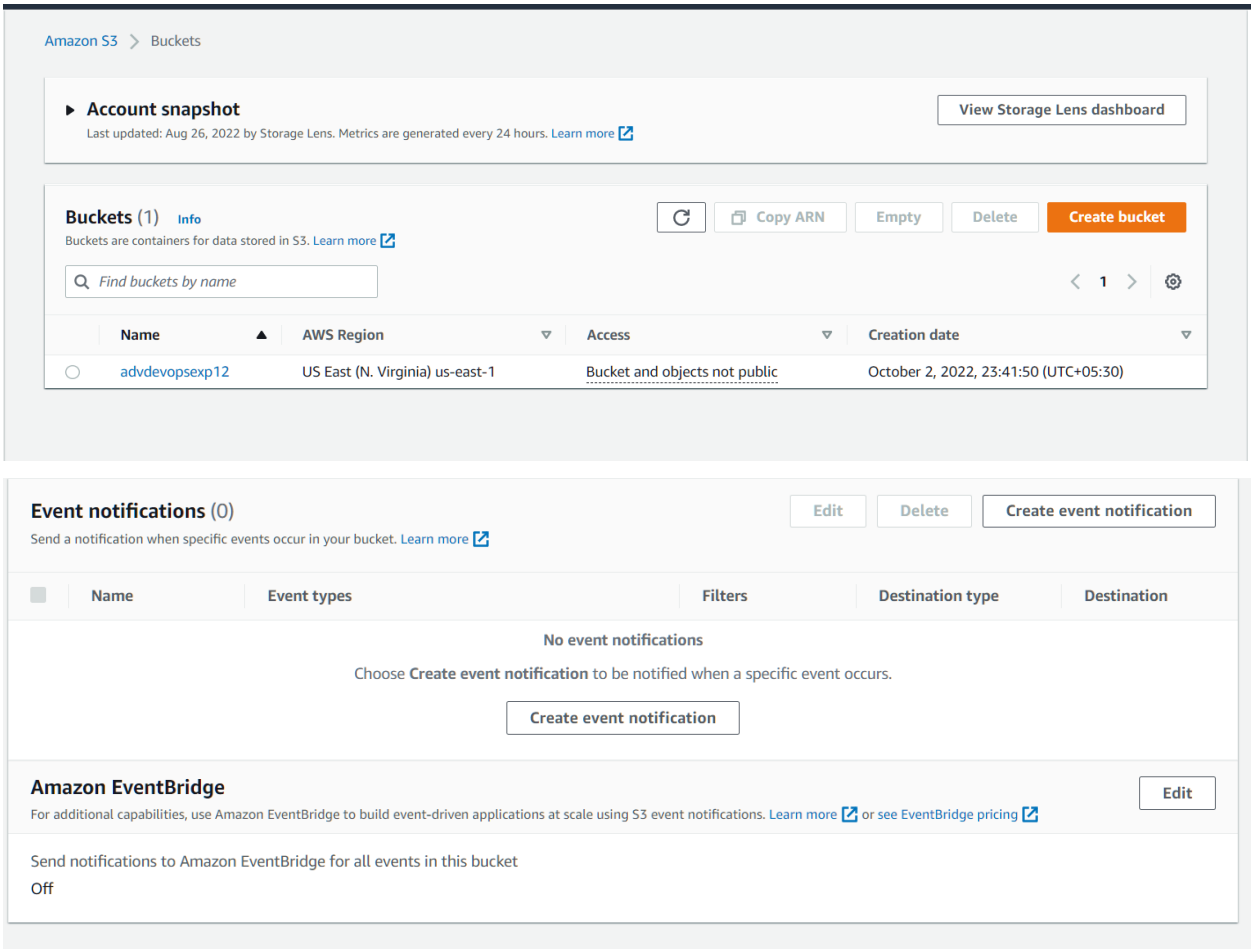
Step 7: Open up the S3 Console and create a new bucket.



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

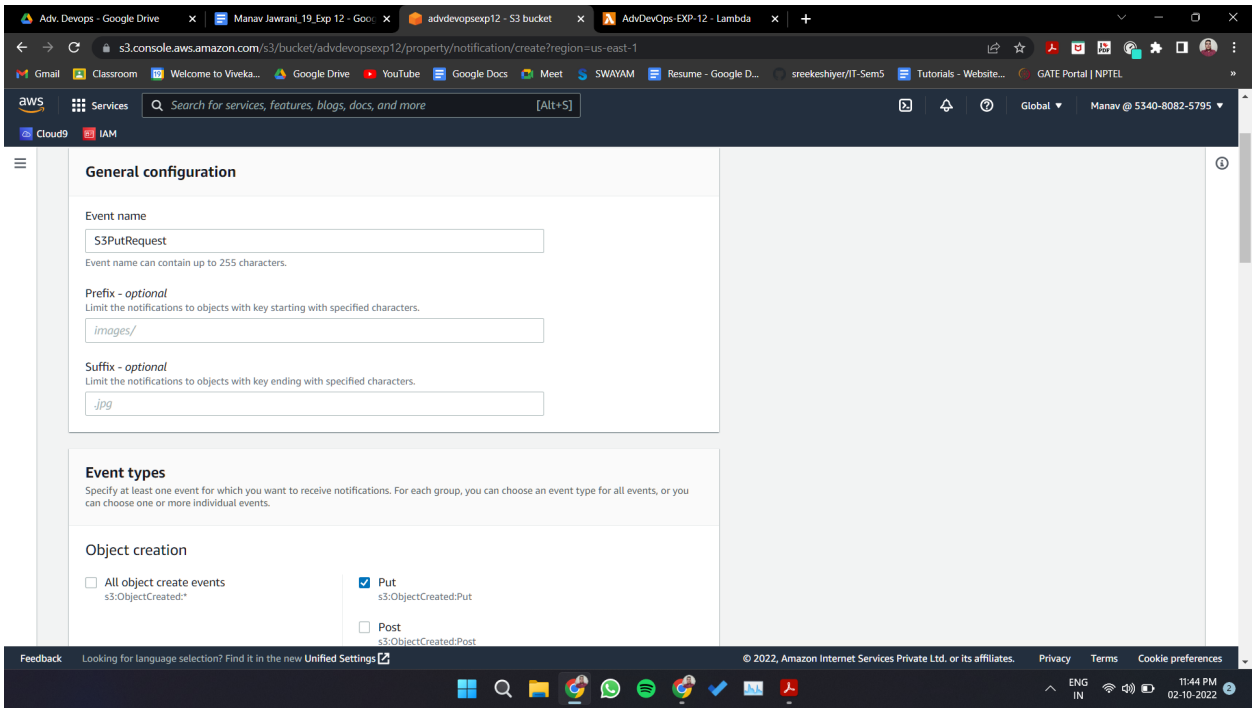


Step 9: Click on the created bucket and under properties, look for events.

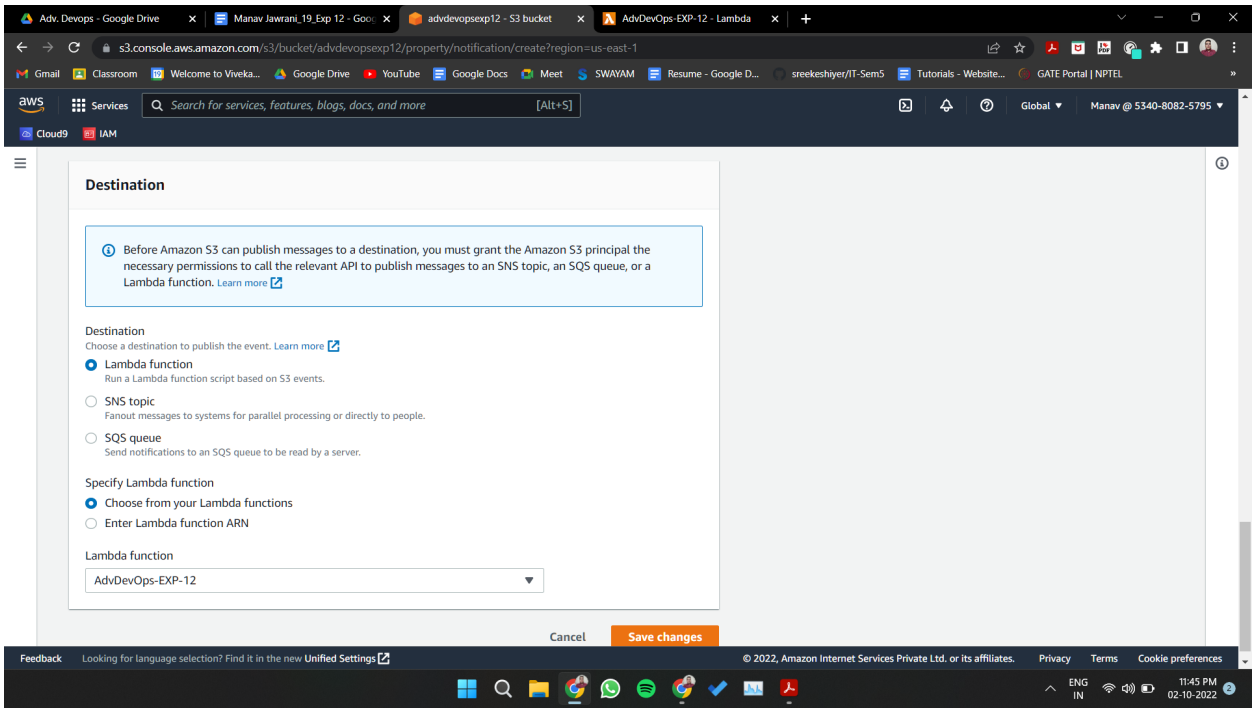


Click on Create Event Notification.

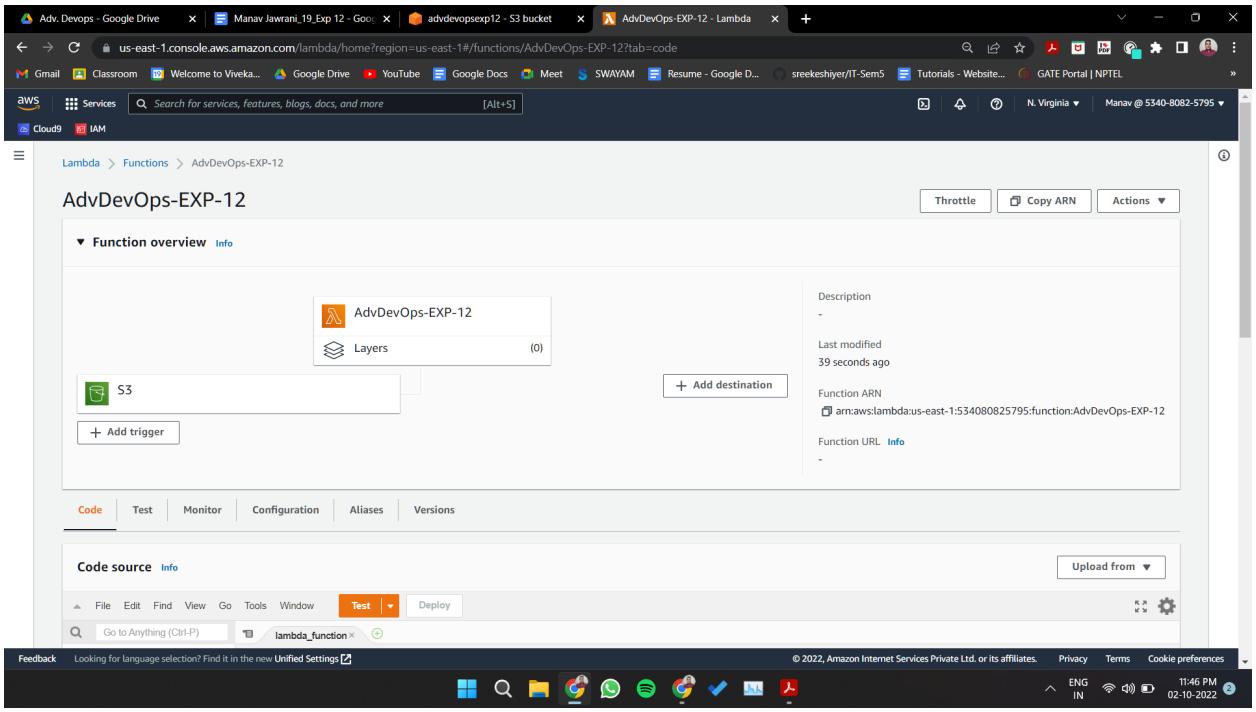
Step 10: Mention an event name and check Put under event types.



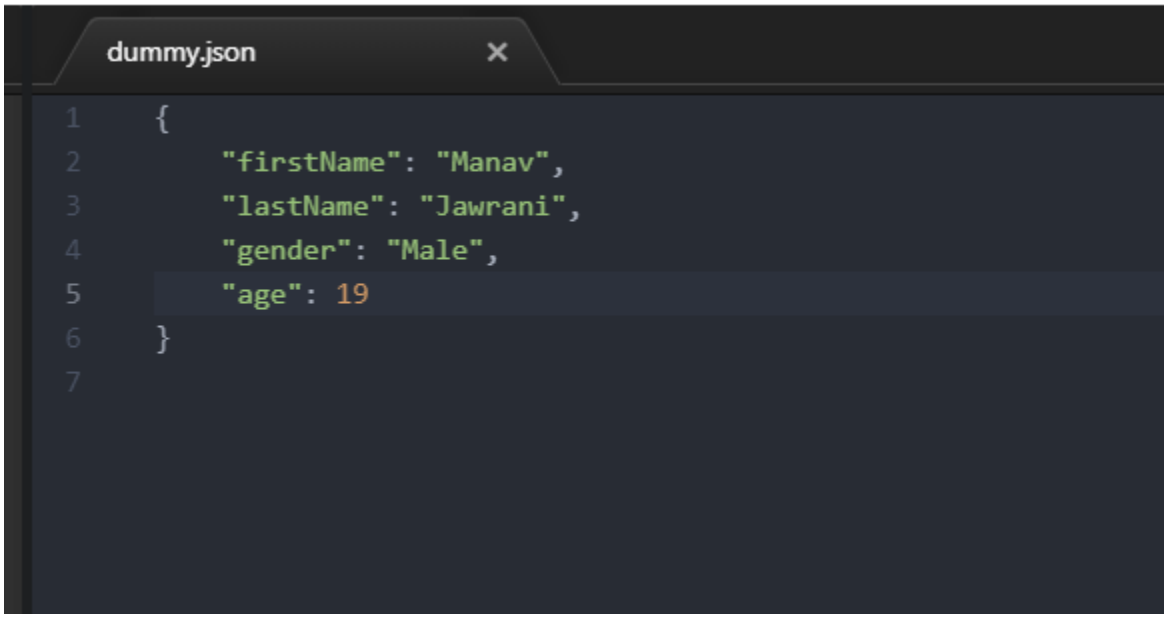
Choose Lambda function as destination and choose your lambda function and save the changes.



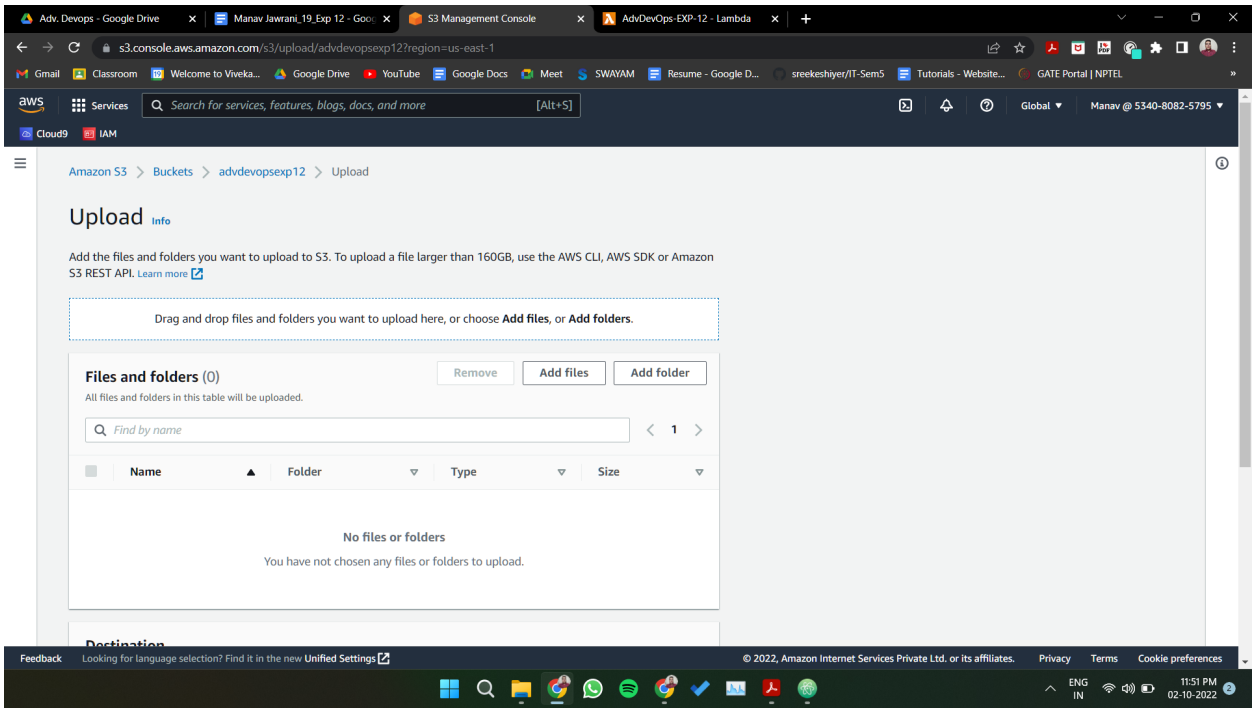
Step 11: Refresh the Lambda function console and you should be able to see an S3 Trigger in the overview.



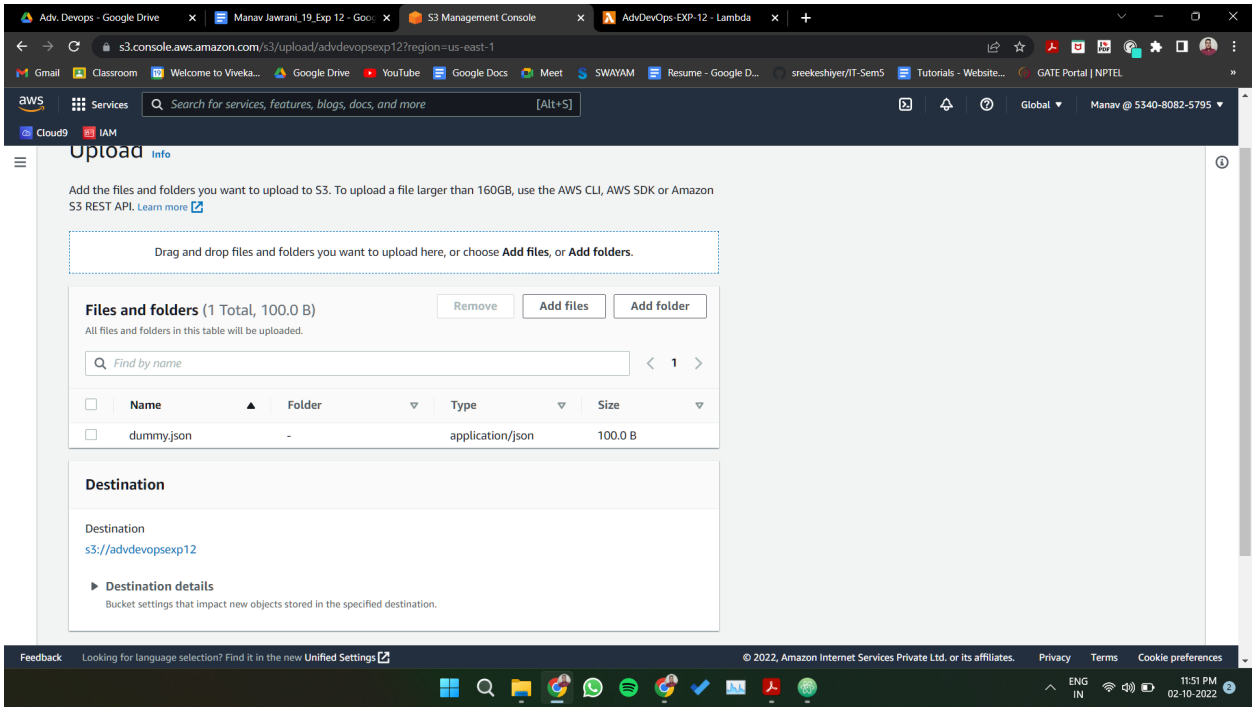
Step 12: Now, create a dummy JSON file locally.



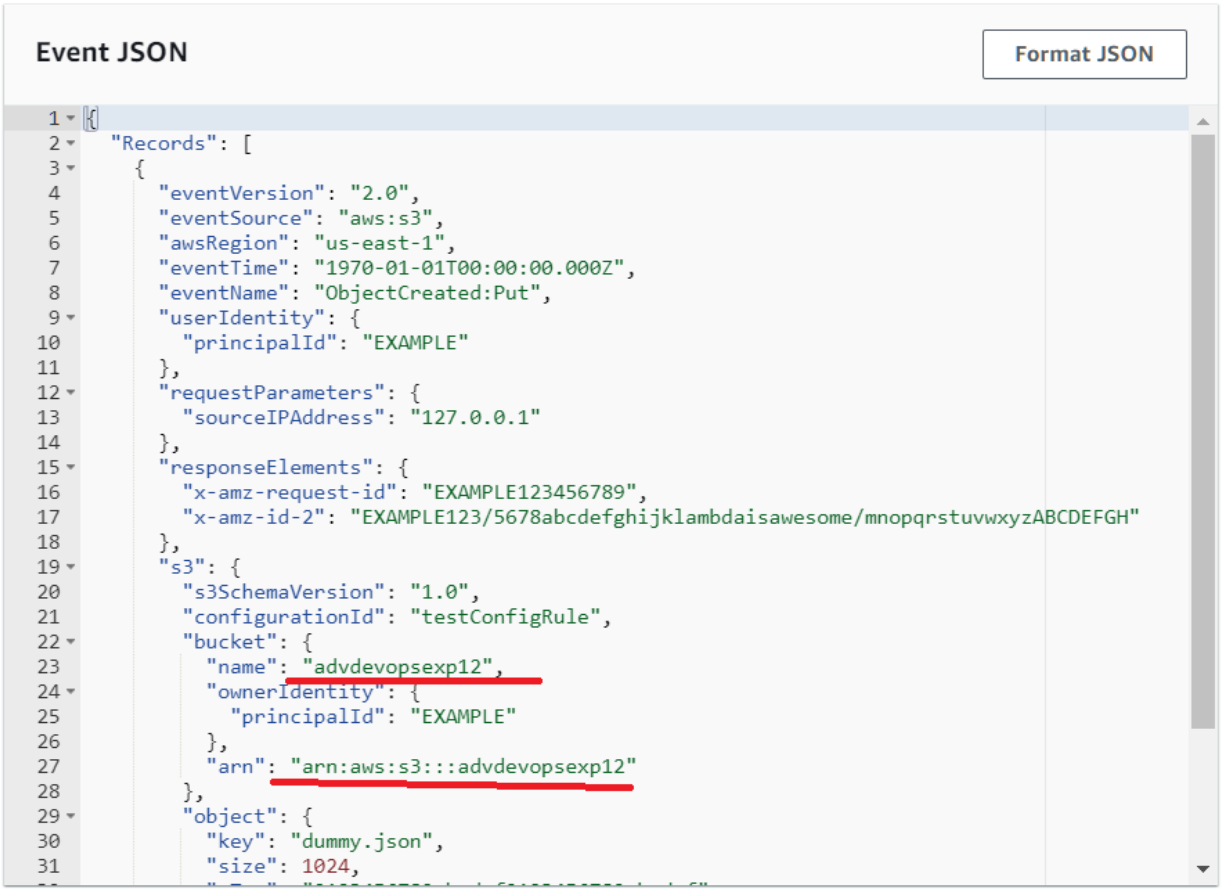
Step 13: Go back to your S3 Bucket and click on Add Files to upload a new file.



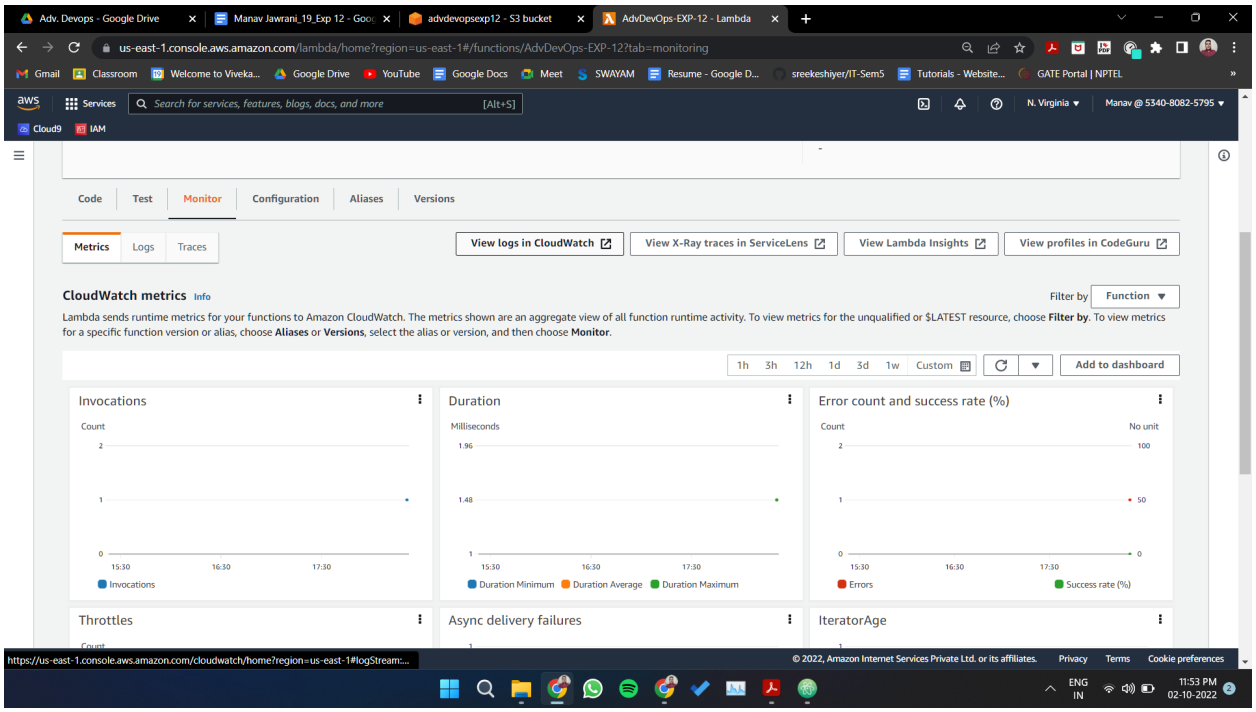
Step 14: Select the dummy data file from your computer and click Upload.



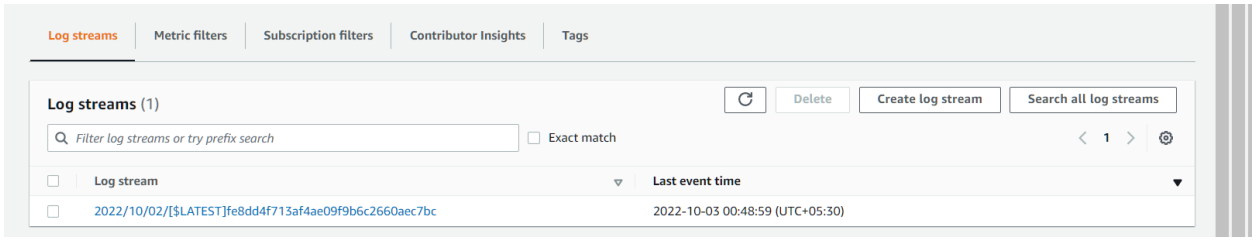
Step 15: After this make the necessary changes in the Test configuration file which we created previously by replacing the **Bucket Name** and the **ARN of Bucket.**



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.

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.

Conclusion:

1. Here, we have successfully created a python function, inside AWS Lambda which logs everytime when an object is uploaded to an AWS S3 bucket.
2. Along with this we got to know how to create ~~a~~ logs for any of the Lambda function in future so that we can see the logs everytime whenever there is a change.