Individual Cloud Computing Journey - Week 5
Serverless computing with AWS Lambda

Aniketh Satyanarayana

1)Getting into the Lambda Console:
Using the navigation bar on the top, search 'Lambda' and open up the Lambda Console.

What is a Lambda function like?

- It has the code, the dependencies and also the config.
- The compute resources like memory, timeout for execution and the IAM role assumed by Lambda go into the configuration.

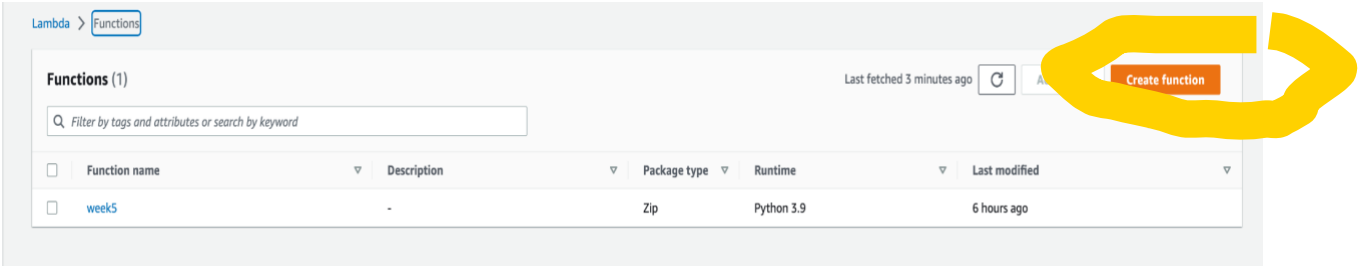
Create a function with the help of a Lambda blueprint.

- While creating a function in the Lambda console, you can choose to start from scratch, use a blueprint, use a container image, or deploy an application from the AWS Serverless Application Repository.
- Also,from within Lambda one can easily switch between languages and frameworks.

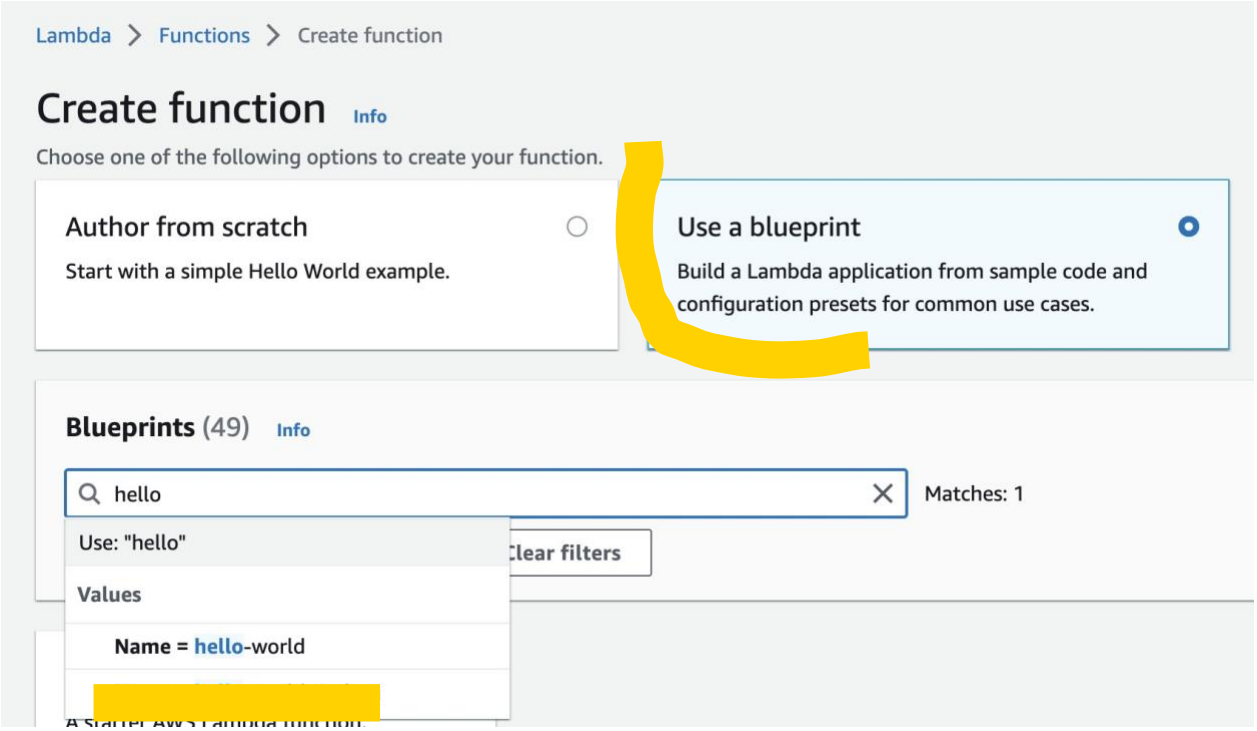
How does a Lambda function blueprint help here?

- A blueprint provides sample code that shows how to use Lambda with an AWS service or a popular third-party application.
- Blueprints include sample code and function configuration presets for Node.js and Python runtimes.

In the AWS Lambda console, choose **‘Create function’.**



Select **‘Use a blueprint’.** In the dropdown that appears, choose **‘hello-world-python’.**



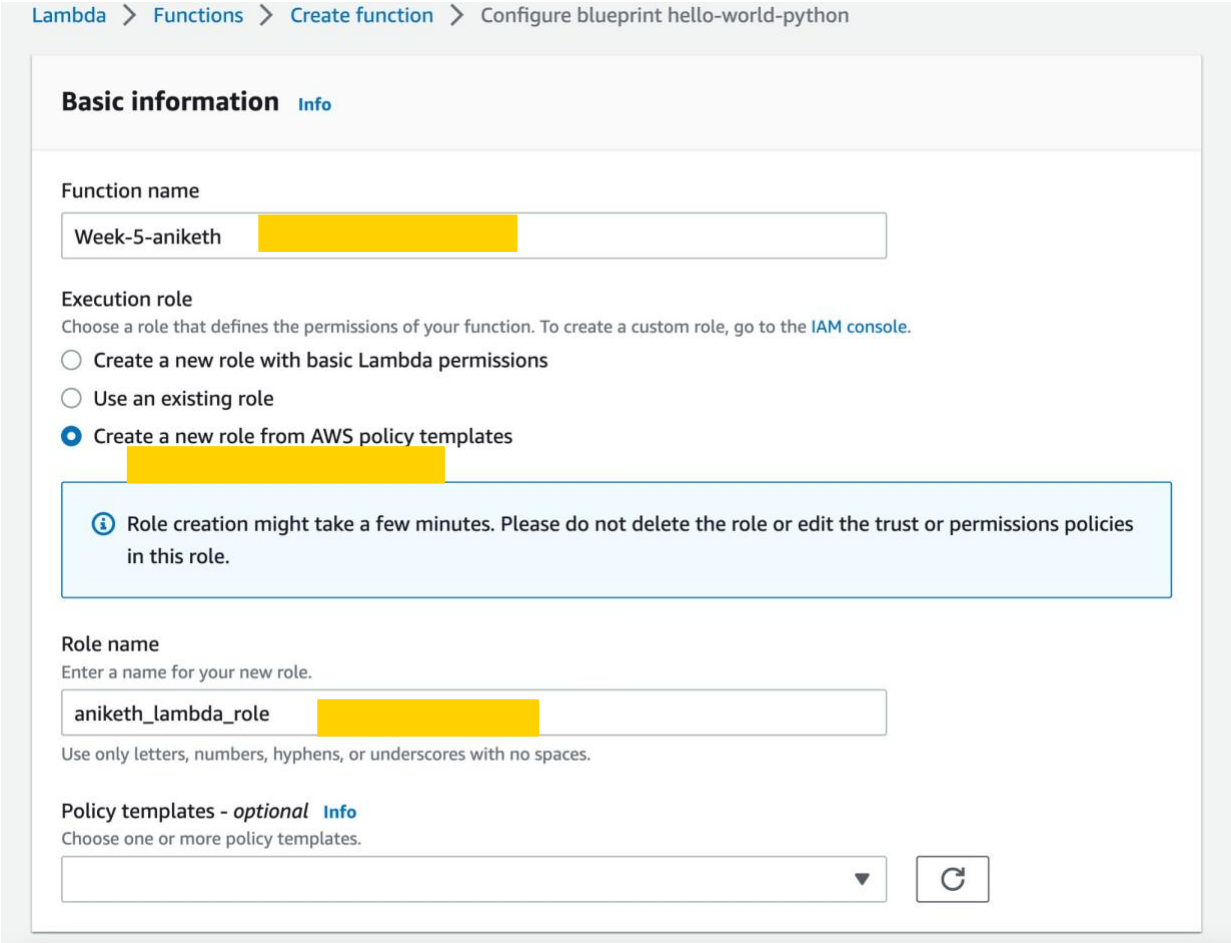
The blueprint ‘hello-world-function’ looks like this.

The following basic information are to be provided:

Function name : I am naming this as 'Week-5-aniketh'

Execution role : 'Create a new role from AWS policy templates'

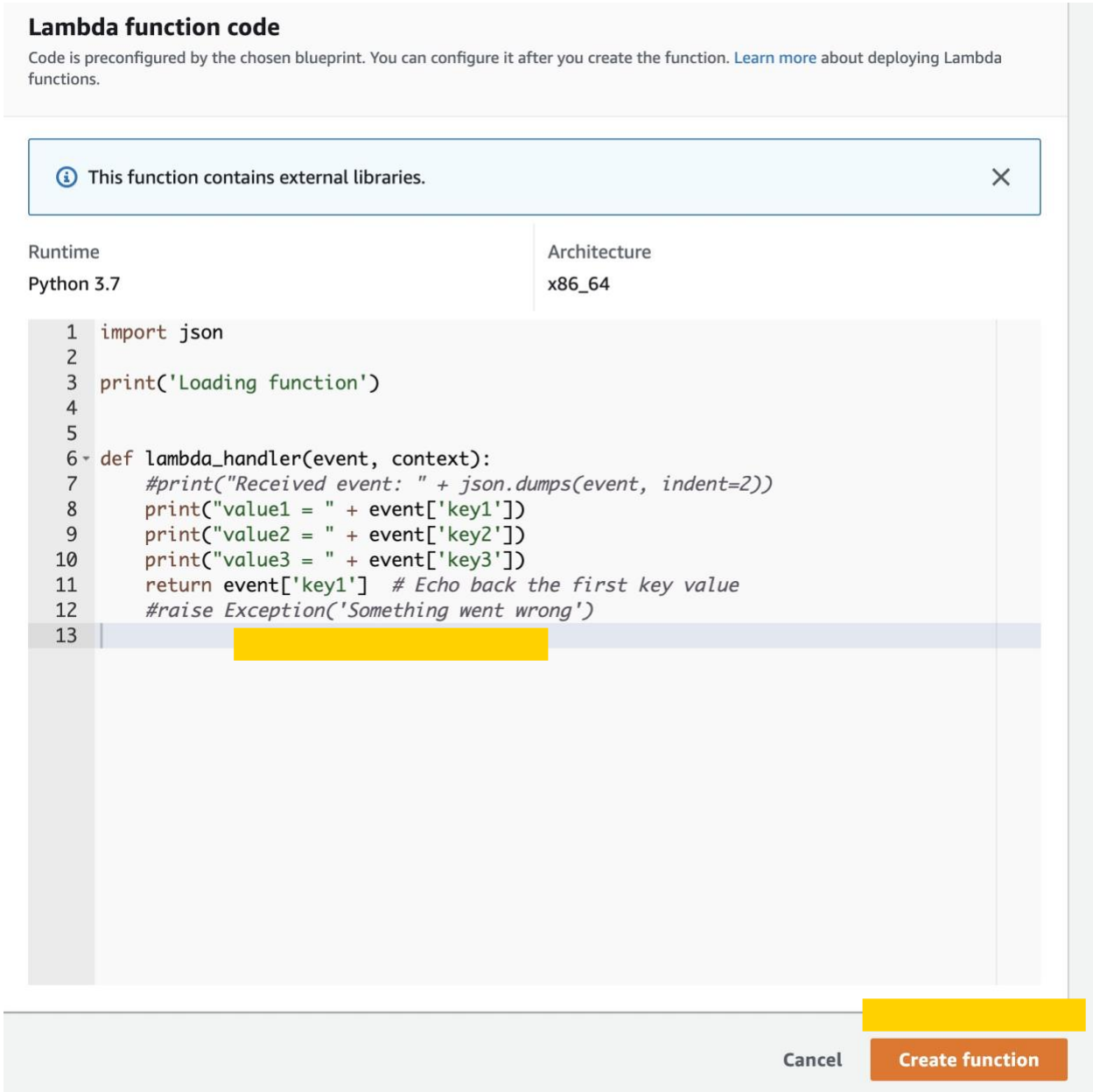
Role name : I am assigning 'aniketh_lambda_role'



Create a lambda function and configure it:

For the lambda function code, I am using the predefined function,which is,

```
import json

print('Loading function')

def lambda_handler(event, context):

    #print("Received event: " + json.dumps(event, indent=2))

    print("value1 = " + event['key1'])

    print("value2 = " + event['key2'])

    print("value3 = " + event['key3'])

    return event['key1']  # Echo back the first key value

    #raise Exception('Something went wrong')
```



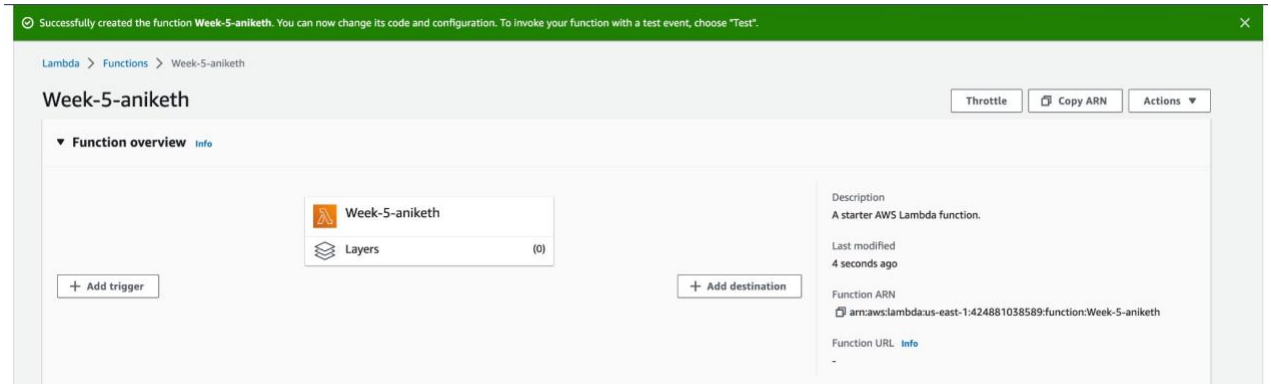
Review the lambda function code and choose '**Create Function'.**

Here **def** is for defining a function with a handler name as lambda_handler, it can be anything depending on your use.

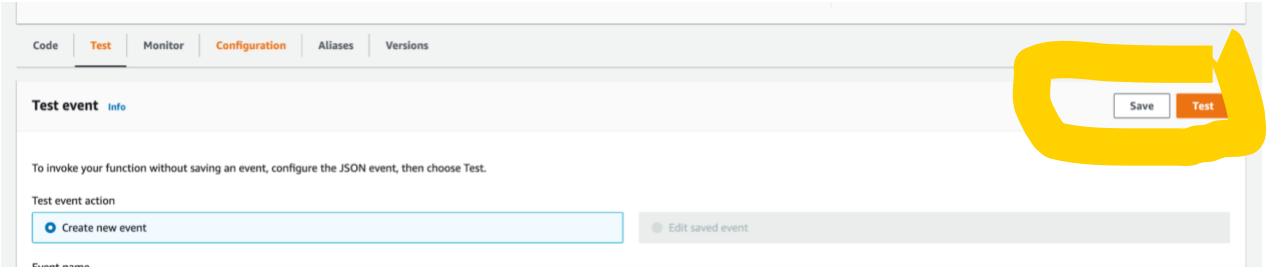
event :- this is the type of data that is being passed to your handler. Generally it is of python dict type but again it can be of any type.

context:- this is used to provide runtime information to your handler. This is of LambdaContext type.

The newly created function shows up like this.



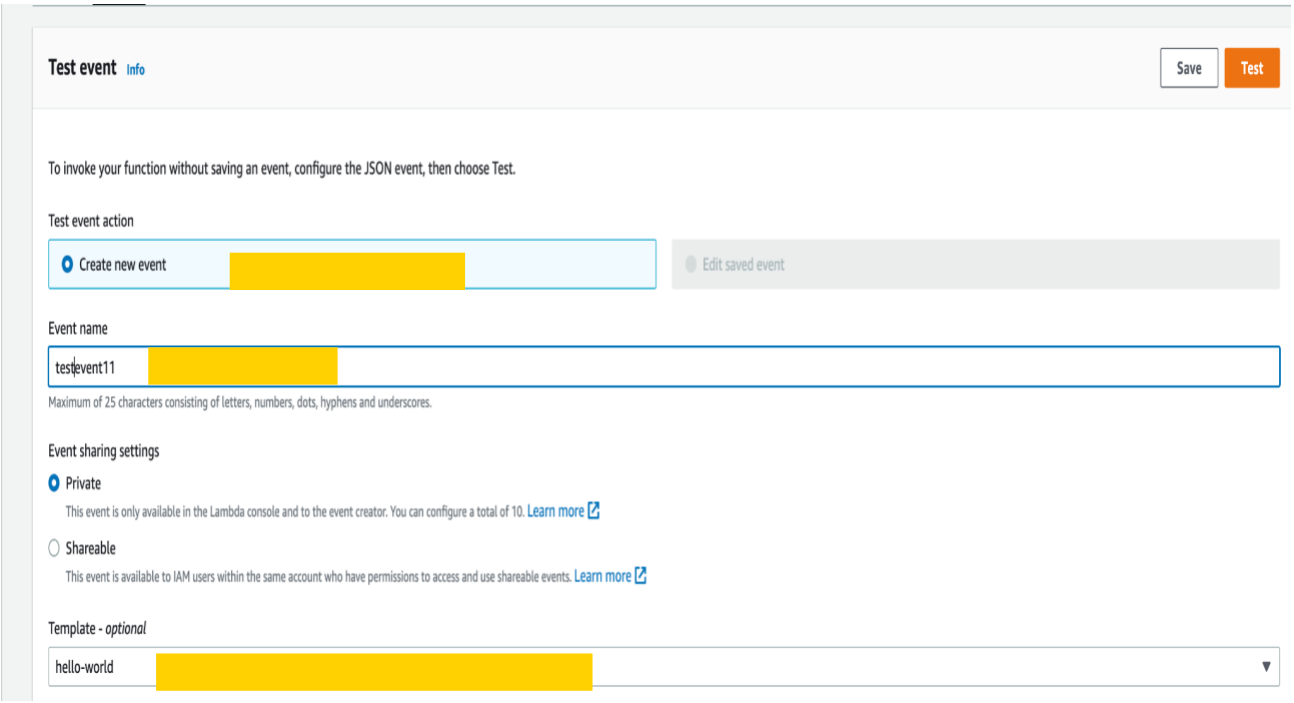
We need to test the function to make sure that it delivers.



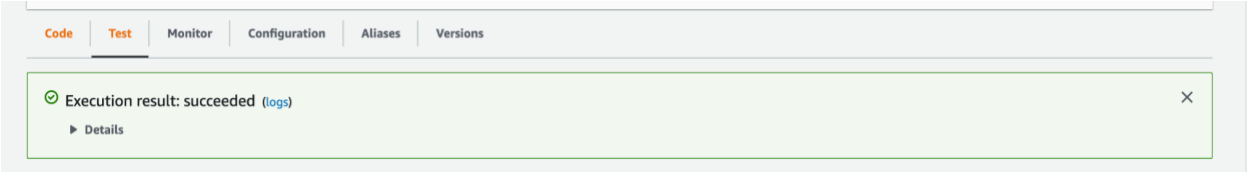
Select **Configure Test Event** from the drop-down menu called Test.

What is a Test event?
Test events provide developers the ability to define a sample event in the Lambda console, and then invoke a Lambda function using that event to test their code. One can **'Create a new event'** or **choose from a 'saved event'.**

The event settings can be set to Private or Shareable. **Private** Previously, test events were only available to the developers who created them. **Shareable -**Developers can make test events available to other team members in their AWS account using granular IAM permissions. This capability makes it easier for developers to collaborate and streamline testing workflows. It also allows developers to use a consistent set of test events across their entire team.



A success prompt like this appears after the creation of the test.



Further, upon expanding the dropdown, we can see the logs generated upon running the function. (Function Logs and the Log output.).

Successfully created the function **Week-5-aniketh**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Execution result: succeeded (logs)

▼ Details

The area below shows the last 4 KB of the execution log.

"value1"

Summary

Code SHA-256 PT7Yw4PDjiA4lGwvKIhvAD0SVhf8BOeHcgRfXW/ogVo=	Request ID 62fd3b28-6cf4-4917-a248-d1eca732814b
Init duration 114.17 ms	Duration 2.89 ms
Billed duration 3 ms	Resources configured 128 MB
Max memory used 36 MB	

Log output

The section below shows the logging calls in your code. Click here to view the corresponding CloudWatch log group.

```
Loading function
START RequestId: 62fd3b28-6cf4-4917-a248-d1eca732814b Version: $LATEST
value1 = value1
value2 = value2
value3 = value3
END RequestId: 62fd3b28-6cf4-4917-a248-d1eca732814b
REPORT RequestId: 62fd3b28-6cf4-4917-a248-d1eca732814b   Duration: 2.89 ms Billed Duration: 3 ms   Memory Size: 128 MB   Max Memory Used: 36 MB   Init Duration: 114.17 ms
```

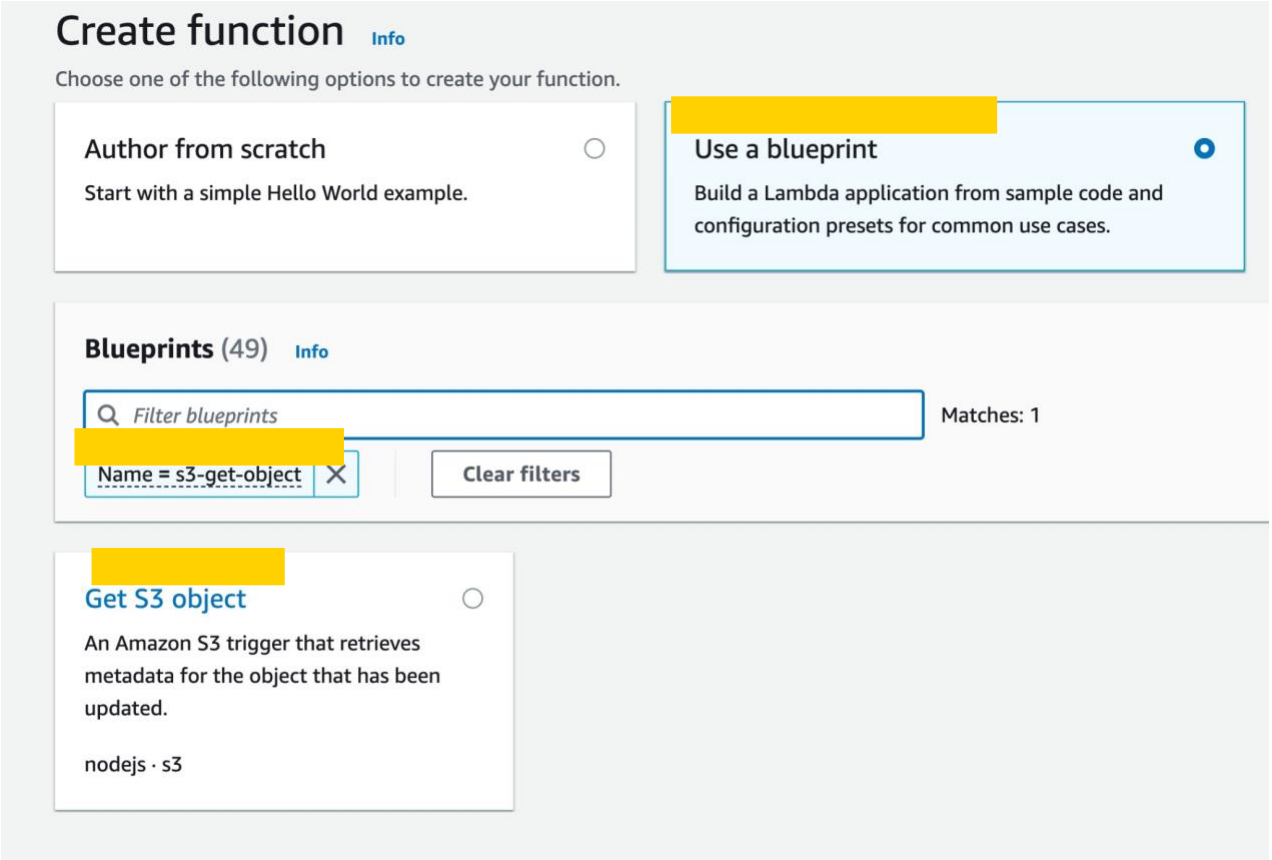
Going a step ahead,

To invoke a lambda function on S3 Bucket trigger:

The overall Steps: some of which are rudimentary and/or explained previously.

1) Create a bucket and upload a sample object
2) Create the Lambda function 3) Review the function code 4) Test in the console

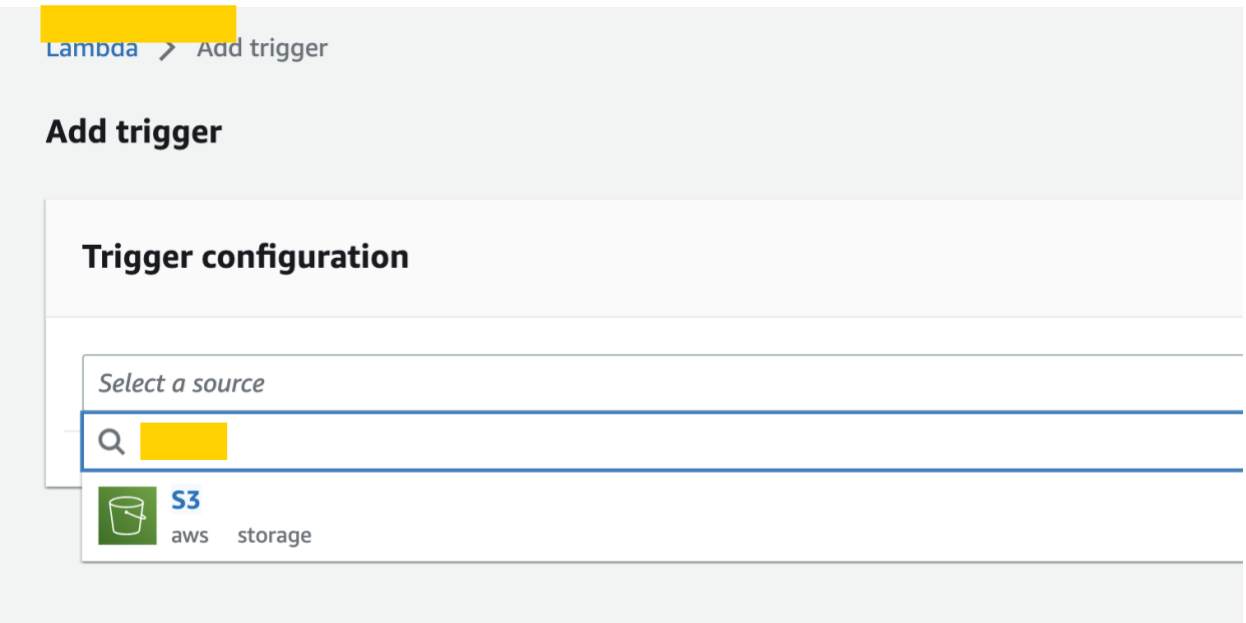
Lambda function creation:



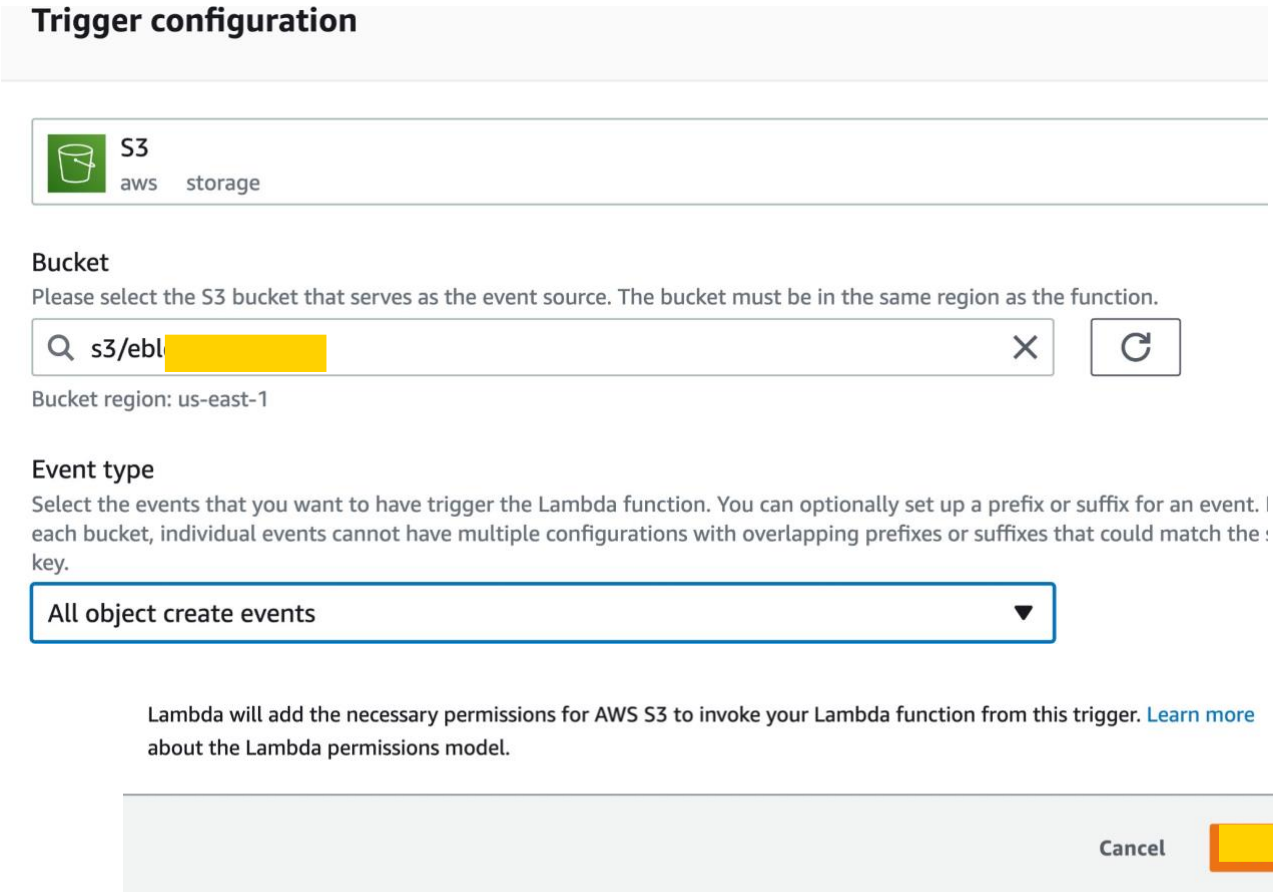
In the search results tab, after selecting a Python function, I choose **s3-get-object-python**, then I choose Configure. **And then in the following image,create trigger.**

- Under Basic information, I do the following:

  Function name : I am naming this as **my-s3-function**.
  Execution role : 'Create a new role from AWS policy templates'
  Role name : I am assigning '**my-s3-function-role**.'
  (Screenshot shown previously)



Configure the trigger as follows:



For the S3 trigger, I choose the S3 bucket that I had created previously. Press the Add button.

Lambda function code

Code is preconfigured by the chosen blueprint. You can configure it after you create the function. Learn more about deploying Lambda functions.

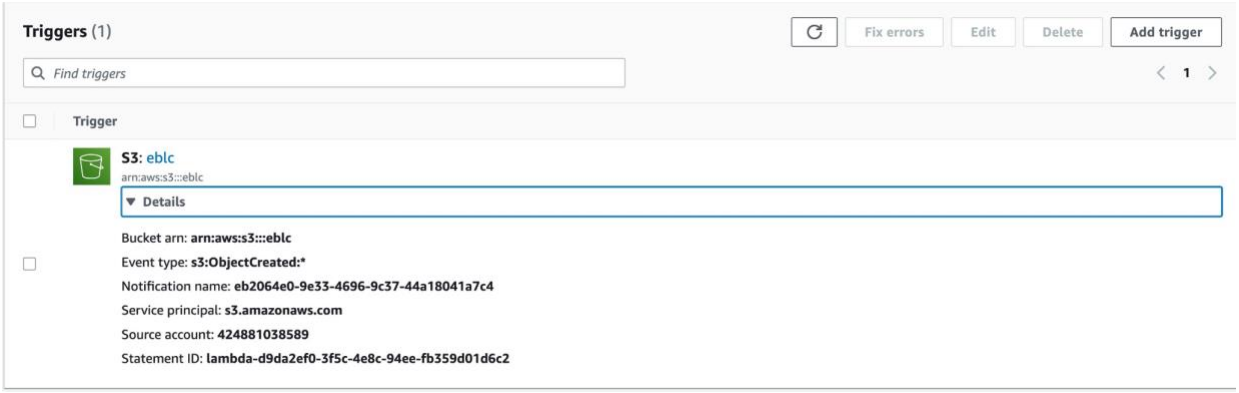
Runtime	Architecture
Python 3.7	x86_64

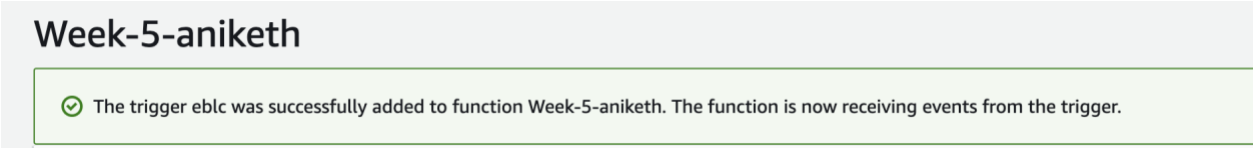
```
mport json
mport urllib.parse
mport boto3

rint('Loading function')

3 = boto3.client('s3')


ef lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encodi
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and your b
        raise e

```

Cancel | Create function

Press the ‘Create Function’ button after reviewing the ‘Lambda function code’.

When an S3 trigger is configured using the Lambda console, the console modifies that function's resource-based policy to allow Amazon S3 to invoke the function.





1)Review the function code

The Lambda function retrieves the source S3 bucket name and the key name of the uploaded object from the event parameter that it receives. The function uses the Amazon S3 `getObject` API to retrieve the content type of the object.

Therefore,in the below code, I replace the parameters "arn":"arn:aws:s3:::sourcebucket" with the bucket name for which the trigger is created. – like
"arn":"arn:aws:s3:::sourcebucket"
To

"arn":"arn:aws:s3:::eblc"

"key":"HappyFace.jpg",
To
"key":"dbscreenshot.jpg",

{
  "Records":[
    {
      "eventVersion":"2.0",
      "eventSource":"aws:s3",
      "awsRegion":"us-west-2",
      "eventTime":"1970-01-01T00:00:00.000Z",
      "eventName":"ObjectCreated:Put",
      "userIdentity":{
      "principalId":"AIDAJDPLRKLG7UEXAMPLE"
      },
      "requestParameters":{
        "sourceIPAddress":"127.0.0.1"
      },
      "responseElements":{
        "x-amz-request-id":"C3D13FE58DE4C810",

"x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/JRWeUWerMUE5JgHvANOjpD"
      },

```
"s3":{
        "s3SchemaVersion":"1.0",
        "configurationId":"testConfigRule",
        "bucket":{
          "name":"sourcebucket",
          "ownerIdentity":{
          "principalId":"A3NL1KOZZKExample"
          },
          "arn":"arn:aws:s3:::sourcebucket"
        },
        "object":{
          "key":"HappyFace.jpg",
          "size":1024,
          "eTag":"d41d8cd98f00b204e9800998ecf8427e",
          "versionId":"096fKKXTRTtl3on89fVO.nfljtsv6qko"
        }
      }
    }
  ]
}
```


2) Test in the console

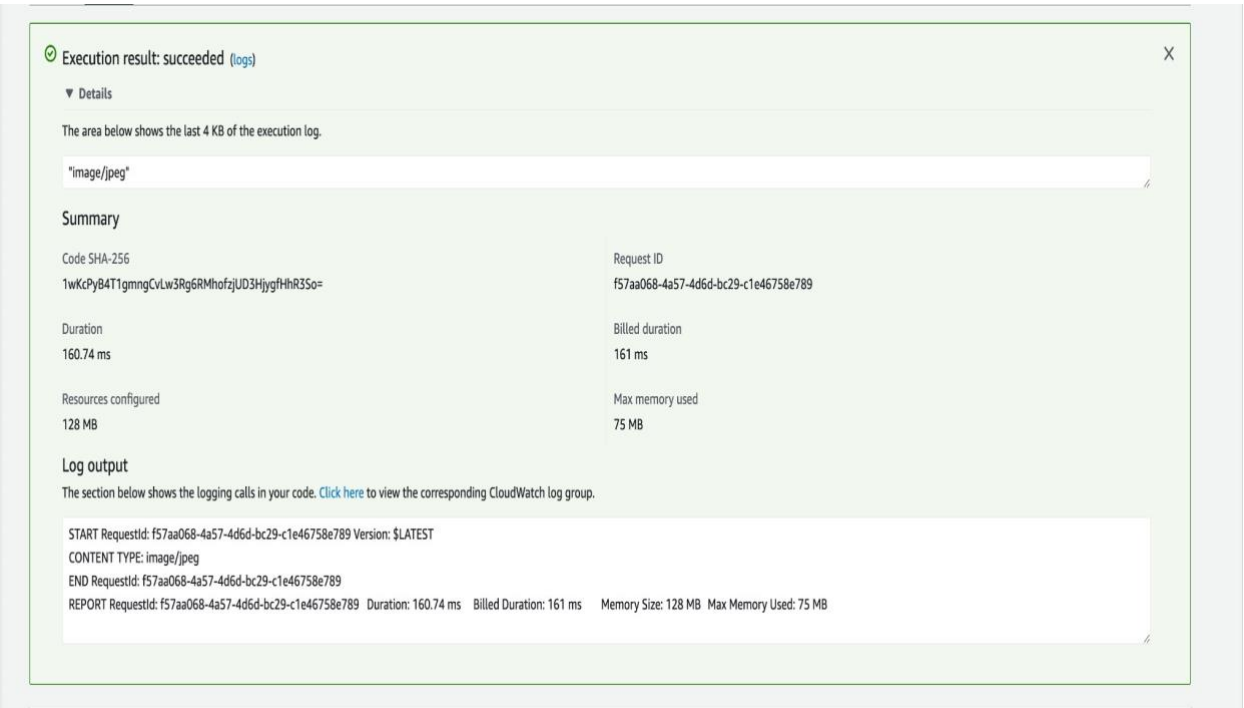
To invoke the function with your test event, under **Code source**, choose **Test**.

The **Execution results** tab displays the response, function logs, and request ID.

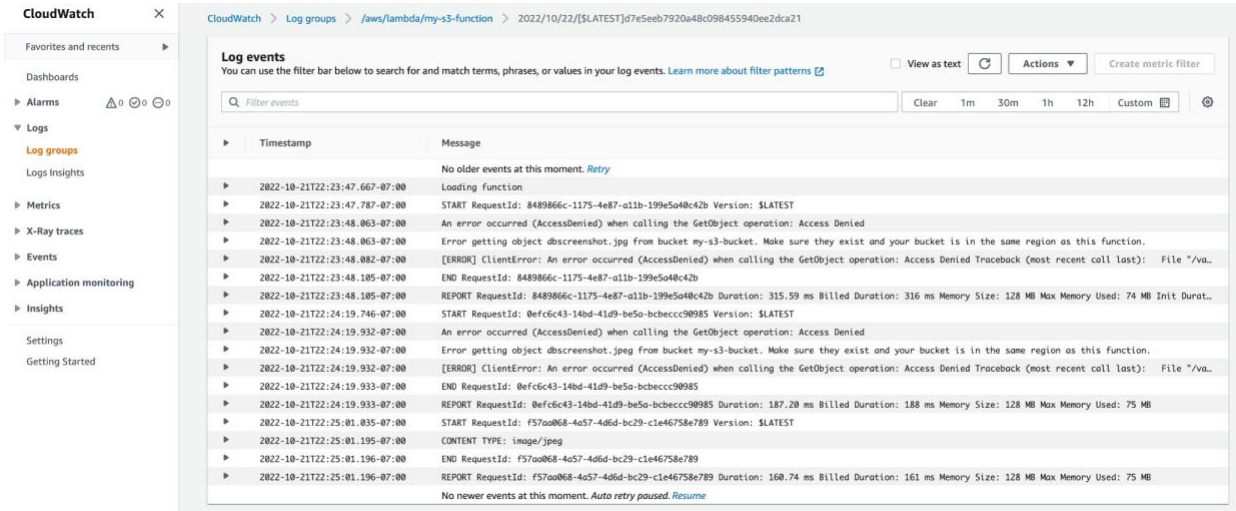
The success query shows as below

With this, we can make sure that whenever a new object is inserted into a S3 bucket, a trigger and an event are generated.

Referring to the screenshot below, we can say that the Lambda function for the s3 bucket trigger is invoked at the said time and the billable duration is displayed.



The logs can be seen like this:



References:

1)Lambda-event-handler context
https://www.edureka.co/community/34245/what-is-the-meaning-of-def-lambda-handler-event-context

3)https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html

3)Test-events-
https://aws.amazon.com/about-aws/whats-new/2022/03/aws-lambda-console-test-events/#:~:text=Test%20events%20provide%20developers%20the,the%20developers%20who%20created%20them.