Sriganesh Pera

AWS Lambda is a serverless and event-driven compute service. It allows you to upload a piece of source code to execute against a valid event.

That uploaded piece of code is called as lambda function.

What is a lambda function?

A lambda function includes a source code along with all dependencies. Each Lambda function has its own configuration information such as:

1. Runtime
2. Environment variables
3. Handler
4. IAM role
5. VPC and memory

The amount of memory that can be allocated to a lambda function ranges from 128MB – 1536MB.Based on requirement, you can allot more memory in increments of 64MB and AWS automatically allots proportional CPU power and resources for executing the specific lambda function.

Lambda function can be configured to execute between 1 to 300 seconds. Lambda function execution time is called timeout and default timeout is 3 seconds.

why Lambda is called Function as a service.

So you write a Lambda function and configure it to respond to an event.

Lambda function is stateless, which means you don't need to worry about underlying infrastructure such as provisioning servers ,high-availability and fault tolerance. Lambada has all those services built-in. It also has built-in logging and monitoring through cloud watch.

So, **Lambda allocates:**

1.Cpu power

2.Disk I/O

3.Network Bandwidth

You allocate:

1.Memory

2.Execution timeout

AWS Lambda supports followings languages:

1.Java 2.Node.js 3.Python 4.C#

Since Lambda is an event driven function, to **execute a Lambda function** 3 things are required:

1. **Event source:** Event sources publish events that invoke Lambda functions, which in turn passes event itself as an argument to **Handler function**.

Examples of event sources: S3, Kinesis Streams, DynamoDB, SNS, SES and Cognito.

2. **Function Code:** AWS lambda function

3. **Permissions:** Successful Code executions requires sufficient privileges. It can be given through IAM.

AWS Lambda scheduled events:

You can create a Lambda function and direct AWS Lambda to execute it on a regular schedule. You can specify :

1. **Rate expression :** fixed rate at which lambda function is executed (for every 1 hour or 15 mins etc)

   e.g; rate(Value Unit) in which Value can be a positive integer. Unit can be minute(s), hour(s), or day(s).

For example:

Example	Cron expression
Invoke Lambda function every 5 minutes	rate(5 minutes)
Invoke Lambda function every hour	rate(1 hour)
Invoke Lambda function every seven days	rate(7 days)

2. **Cron expression:**

cron(Minutes Hours Day-of-month Month Day-of-week Year)

The following table lists common examples of cron expressions.

Example	Cron expression
Invoke a Lambda function at 10:00am (UTC) everyday	cron(0 10 * * ? *)
Invoke a Lambda function 12:15pm (UTC) everyday	cron(15 12 * * ? *)
Invoke a Lambda function at 06:00pm (UTC) every Mon-Fri	cron(0 18 ? * MON-FRI *)
Invoke a Lambda function at 8:00am (UTC) every first day of the month	cron(0 8 1 * ? *)
Invoke a Lambda function every 10 min Mon-Fri	cron(0/10 * ? * MON-FRI *)

For both rate expression and cron expression the default minimum is 5 minutes. You can't schedule the function to execute below the minimum time.

Programming model of AWS Lambda:

Code written for AWS Lambda follows a pattern. There are 4 concepts that form the core of the programming model:

1. Handler:

When a Lambda function is invoked, AWS Lambda starts executing the code by calling the **handler function.**

AWS Lambda passes any event data to this handler as the first parameter. Then this handler process the incoming event data and may invoke any other functions/methods in the code.

Handler function syntax:

```
exports.myHandler = function(event, context, callback) {
  // Your code goes here.
  callback();
}
```

myHandler is the handler function to which event data is passed. As you can see,Handler function has three parameters : event,context,callback

2. **Event:** Lambda uses this parameter to pass any event related data back to the handler.

3. **Context**: Context data consist of Lambda function's runtime information such as function name ,log name, stream name and how much time is remaining before Lambda terminates the function. This context object data is passed to handler.

Some commonly used context object methods are:

- `getRemainingTimeInMillis()`
- `functionName`
- `functionVersion`

- `memoryLimitInMB`:
- `logGroupName`:

4. **callback**: This parameter is used to return any data back to its caller. If not specified, AWS Lambda will call it implicitly and return the value as null.

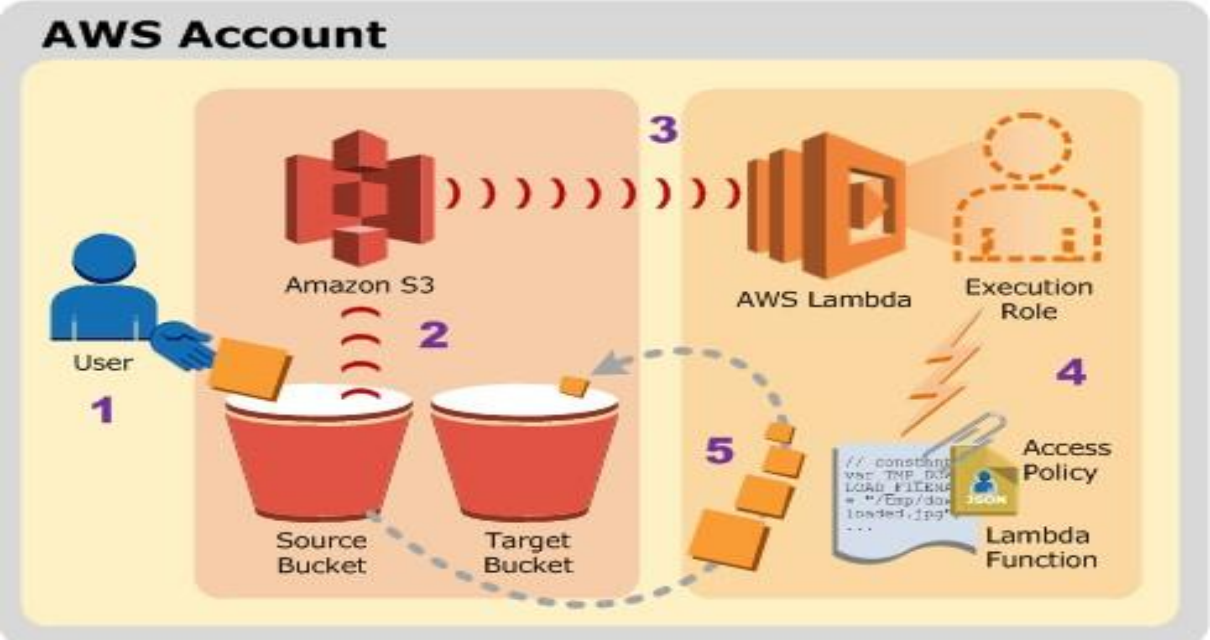
The callback parameter also supports two optional parameters: error and result; error will return any of the function's error information back to the caller, while result will return any result of the function's successful execution.

:**Still not getting it?**

Let's do a simple lab then. Before, we do the lab let me explain the workflow of what we going to implement using AWS Lambda.

Example workflow:

Lab objective: Sending a resized image to an other from a bucket that stores original sized image.



Algorithmic steps:

1.A user uploads an object in S3

2. Amazon S3 detects the object-created event.

3.Amazon S3 detects the object-created event to AWS Lambda by invoking Lambda function and passes event data as a parameter.

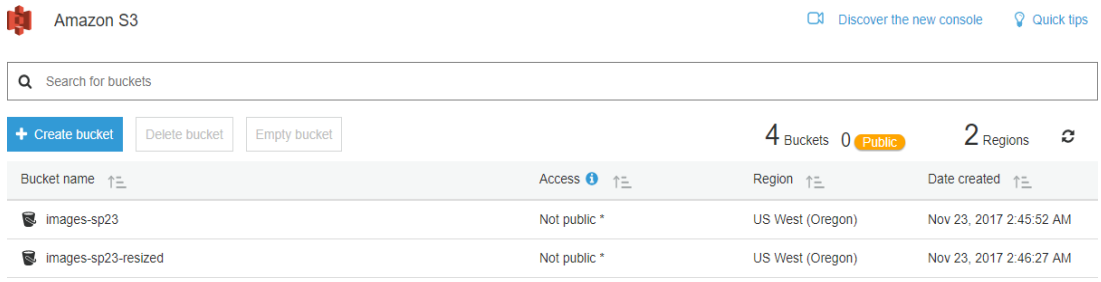
4.AWS Lambda executes the function.

5.From the event data it receives, the Lambda function knows the source bucket name and object key name. The lambda function reads the object and creates a thumbnail to the target bucket.

Now let's deep dive into Aws Lambda

Step 1: Create two S3 Buckets one with named images-(your initials)23 and other with images-(your initials)23-resized

In my case it is sp so the bucket name is images-sp23 and images-sp23-resized



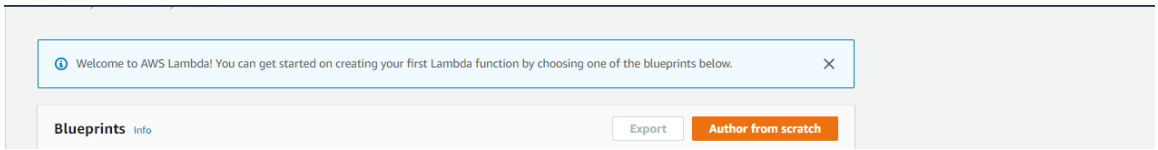
Now upload the image in the given link inside images-sp-23

https://s3-us-west-2.amazonaws.com/imgaes/HappyFace.jpg

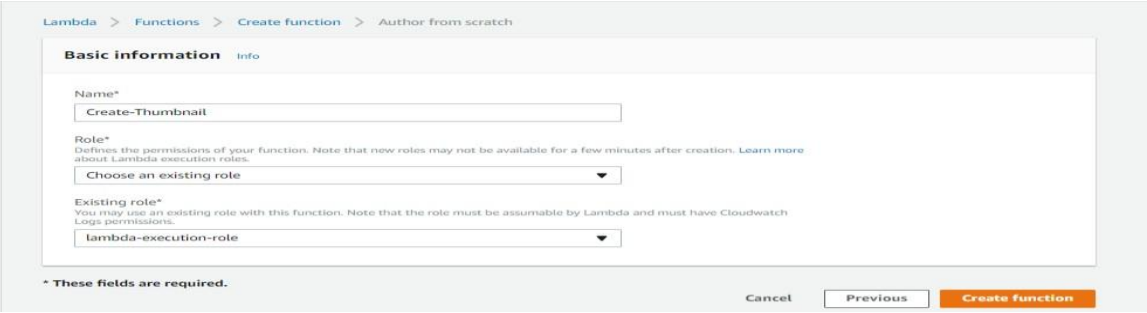
Now before you go to step 2 go to IAM and create a role named 'lambda-execution-role' and attach lambda full access policy to it.

Step 2 : Create Lambda function:

Now go to AWS Lambda and click create function. Now on the blue prints page, Click author from scratch-the orange button.



Now set the details as following and click create function.



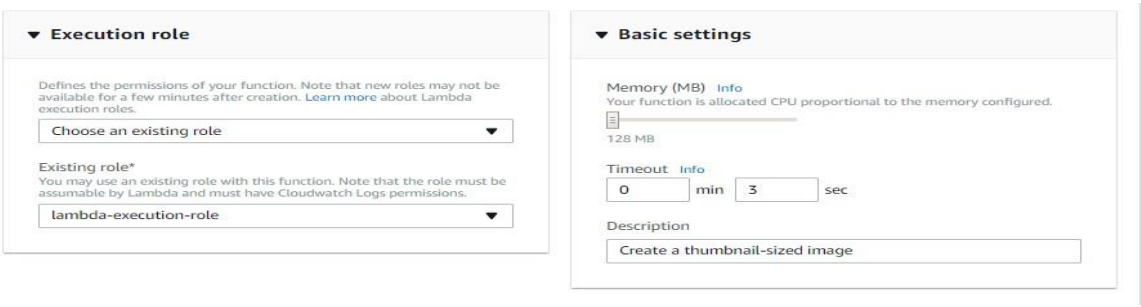
Now in the function code set the following:

Copy and paste this S3 link on the URL: https://s3-us-west-2.amazonaws.com/us-west-2-aws-training/awsu-spl/spl-88/scripts/CreateThumbnail.zip

Now this zip folder contains the following code. Let's analyze the code:

1. Receive an Event, which contains the name of the incoming object(Bucket,Key)
2. Download the image to local storage
3. Resize the image using Pillow library
4. Upload the resized image to -resized bucket.

Now expand the basic settings at the bottom of the page and enter the following description as shown:

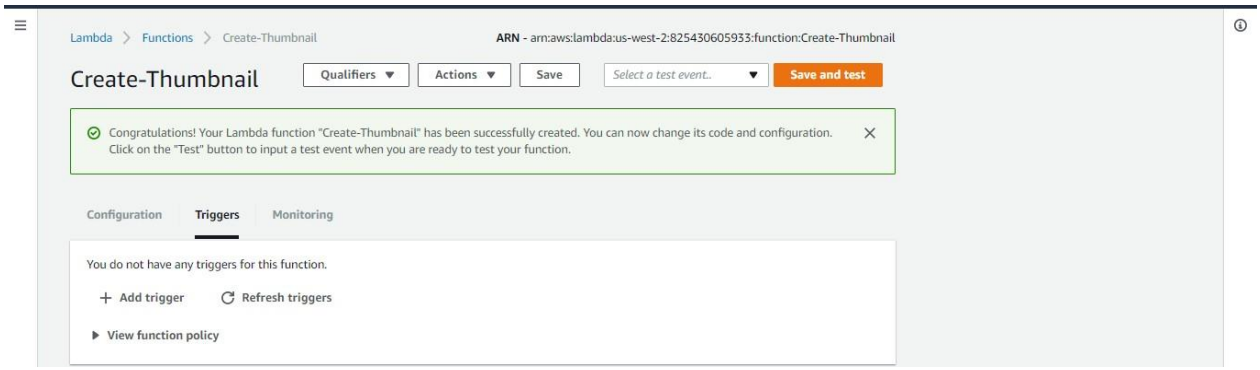


Now leave other settings as default:

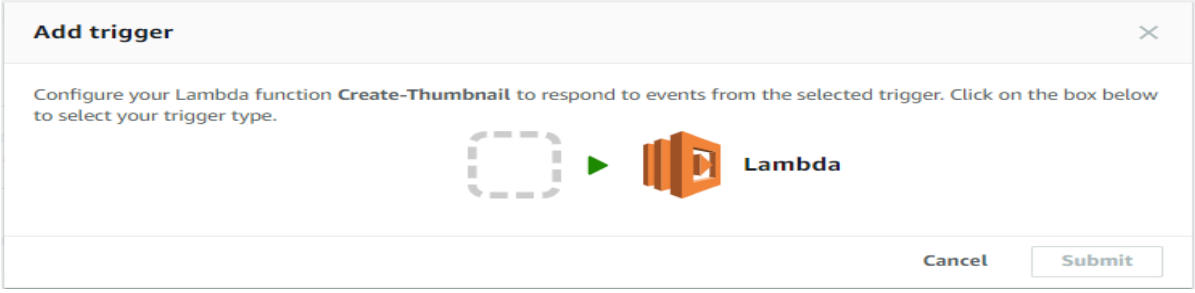
Brief explanation of other settings:

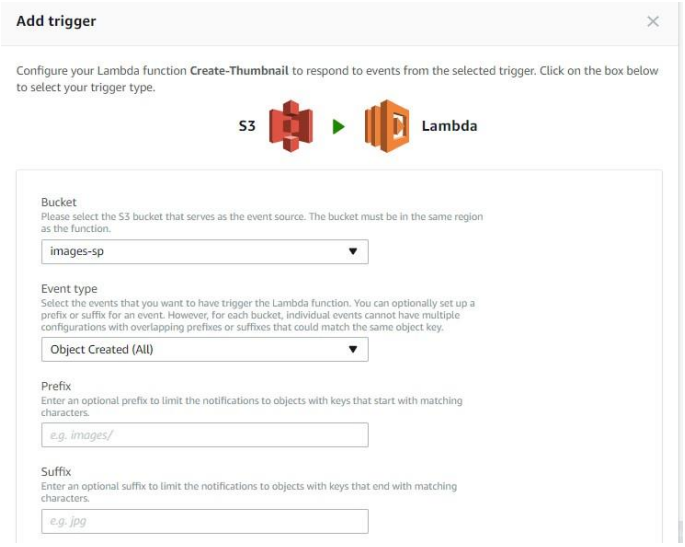
1. **Memory** : defines resources allocated to your function. Increasing memory also increases CPU allocated to the function

Now we are going to create a trigger that Lambda function gets executed if a new image is stored in S3.So,go to the top part of the page and click on triggers tab and click + Add trigger

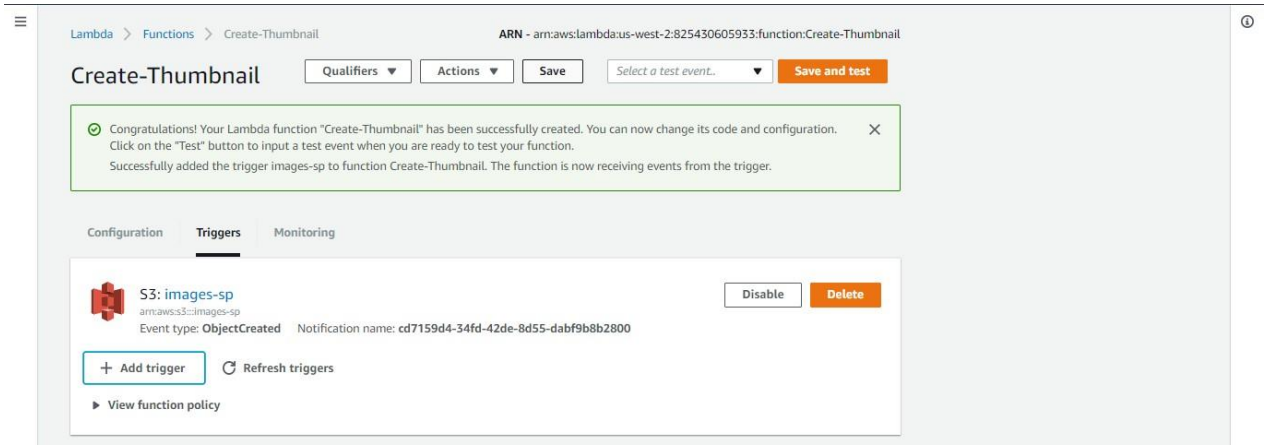


Now it will open a window .Now you click on the empty square and then from the drop-down menu select S3 and configure following settings





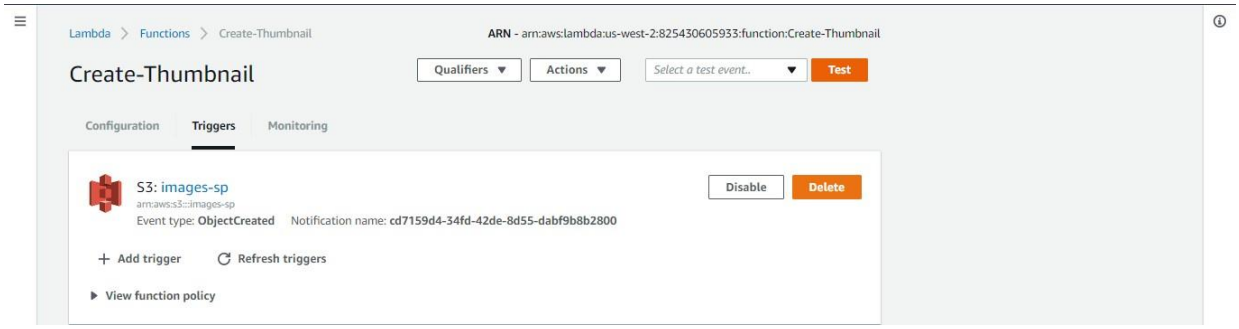
Click submit and click save at the top of the window



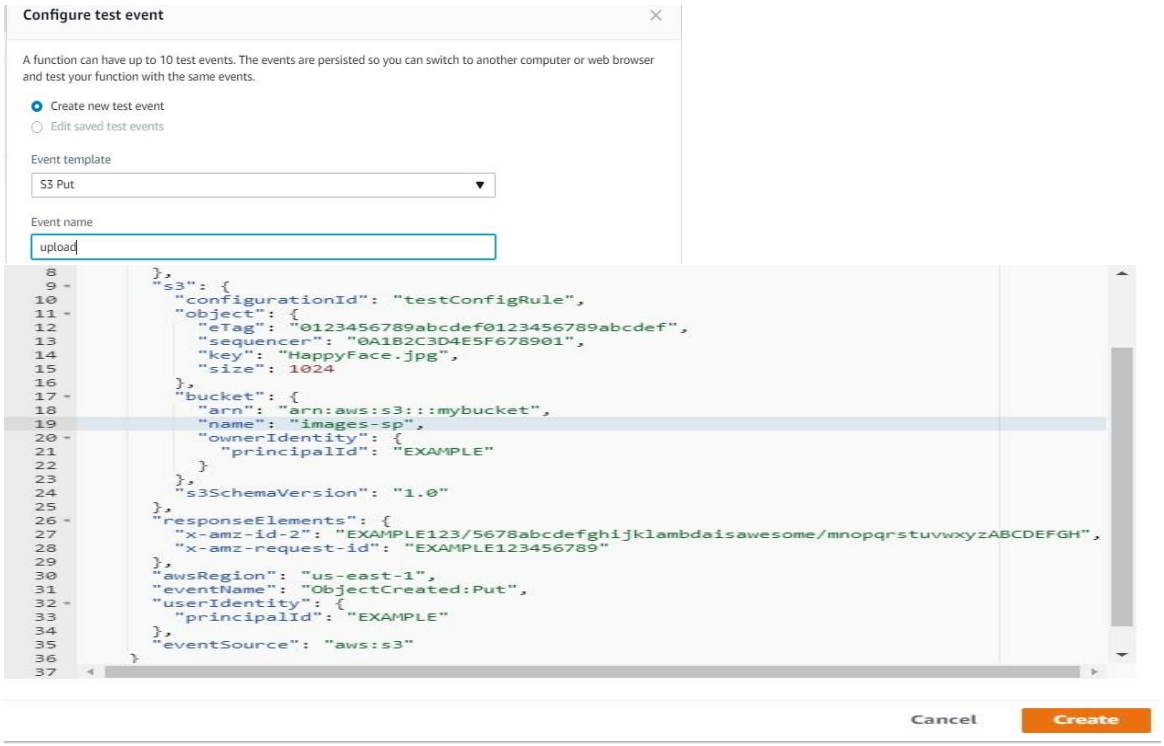
So now the lambda function is configured

Now test your function:

Click on test



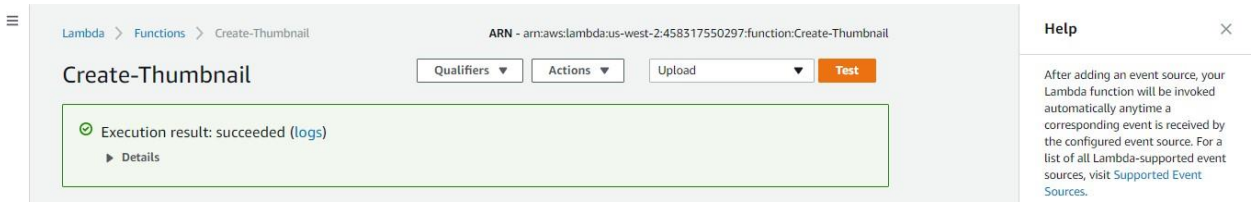
Now a configure test event window is opened. Now select the following



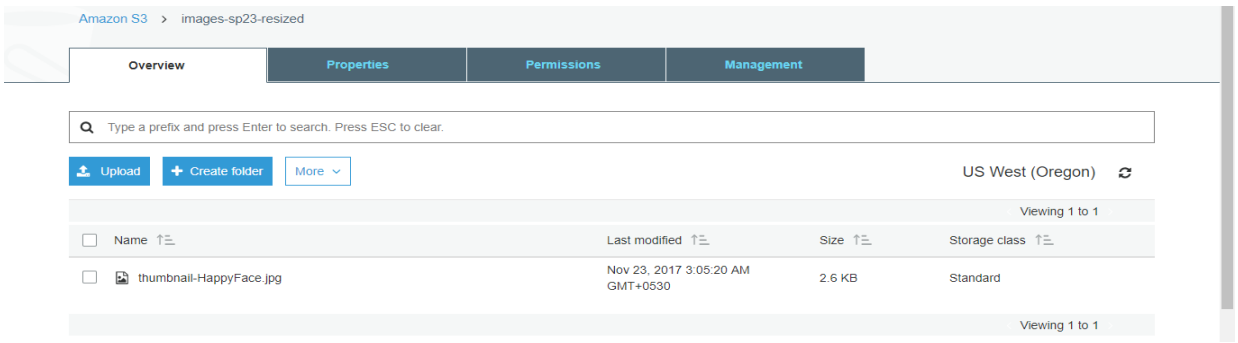

```
      },
      "s3": {
        "configurationId": "testConfigRule",
        "object": {
          "eTag": "0123456789abcdef0123456789abcdef",
          "sequencer": "0A1B2C3D4E5F678901",
          "key": "HappyFace.jpg",
          "size": 1024
        },
        "bucket": {
          "arn": "arn:aws:s3:::mybucket",
          "name": "images-sp",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          }
        },
        "s3SchemaVersion": "1.0"
      },
      "responseElements": {
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH",
        "x-amz-request-id": "EXAMPLE123456789"
      },
      "awsRegion": "us-east-1",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "eventSource": "aws:s3"
    }
```

And edit the code and insert the name of your bucket, in my case it is:

Now click create and then test



Now go to s3 and check your resized bucket to find the resized thumbnail



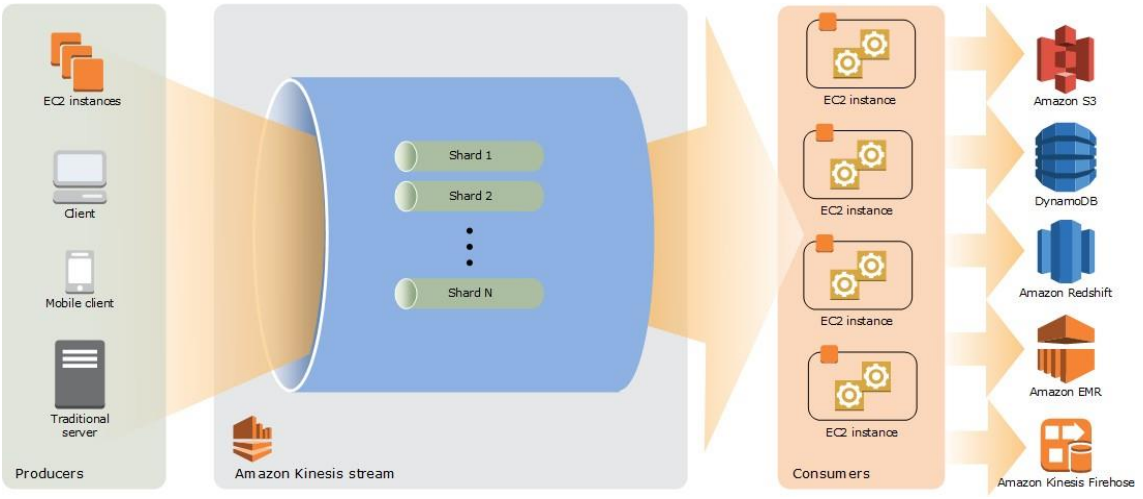
Amazon Kinesis:

Amazon Kinesis is a fully managed service for real-time processing of streaming data at a massive scale. Amazon Kinesis can collect and process hundreds of terabytes of data per hour, from hundreds of thousands of sources and can feed it to many applications at a time.

Sources can be: Website click streams, marketing and financial information, manufacturing instrumentation and social media, and operational logs and metering data.

example: As shown in the diagram sensors in tractors detect the need for a spare part and send the real-time sensor data to kinesis for analysis, which triggers Lambda function ,which in turn runs the code to analyze the trends in data to identify whether a spare part of faulty part replacement is needed or not.

Kinesis Architecture:



The producers continually push data to Kinesis Data Streams and the consumers process the data in real time. Consumers (such as a custom application running on Amazon EC2, or an Amazon Kinesis Data Firehose delivery stream) can store their results using an AWS service such as Amazon DynamoDB, Amazon Redshift, or Amazon S3.

(i) Data Records:

A data record is the unit of data stored in a Kinesis stream. Data records are composed of a sequence number, partition key, and data blob, which is an immutable sequence of bytes.

(ii) Kinesis stream:

Ordered sequence of such data records are called **Kinesis streams**.The data records in the stream are distributed into **shards.**

(iii) Shards:

A group of data records is called a shard. A stream is composed of one or more shards, each of which provides a fixed storage.

Capacity provisioning for each shard:

(a) **for reads:** Clients can read upto 5 transactions per second to a maximum total data read rate of 2 MB per second

(b) **for write:** Clients can write 1,000 records per second for writes, up to a maximum total data write rate of 1 MB per second (including partition keys).

(iv) Partition keys:

Partition keys are used to uniquely identify each shard. The maximum length limit of of a partition key is 256 bytes. A partition key is specified by the applications putting the data into a stream.

(v) Producers and consumers:

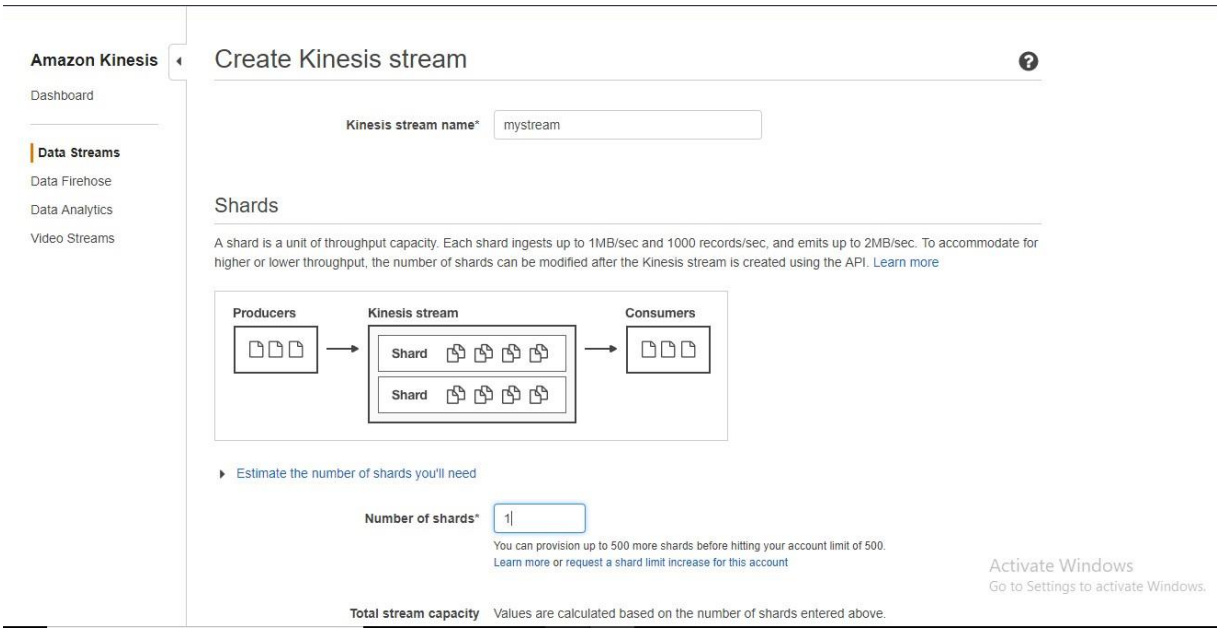
Producers put records into Amazon Kinesis Data Streams. For example, a web server sending log data to a stream is a producer.

Consumers get records from Amazon Kinesis Data Streams and process them. These consumers are known as Amazon Kinesis Data Streams Applications, which are custom applications that run on EC2 servers.

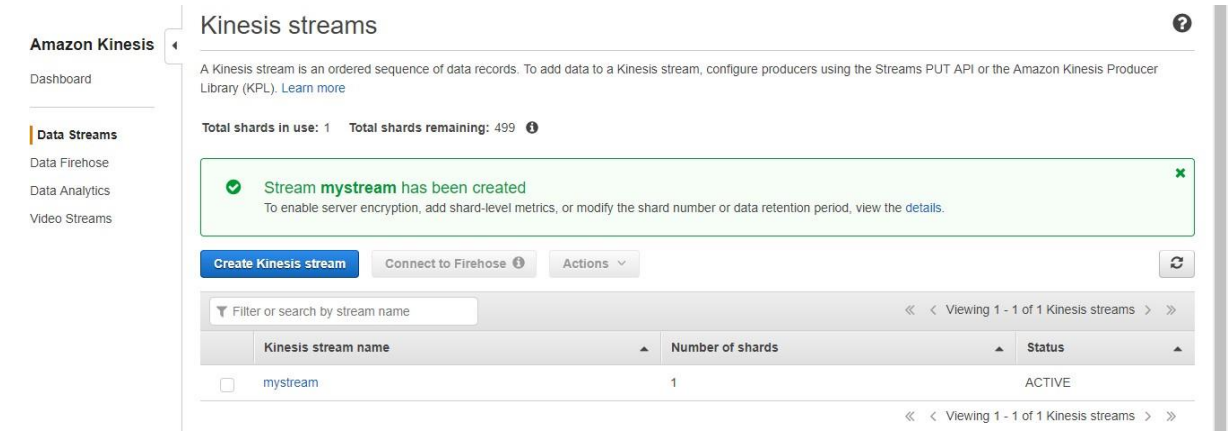
Lab:

Step 1: Create Kinesis data stream

Go to AWS and click Create data stream. Provide your own name and set the number of shards to 1.Then click create kinesis stream.

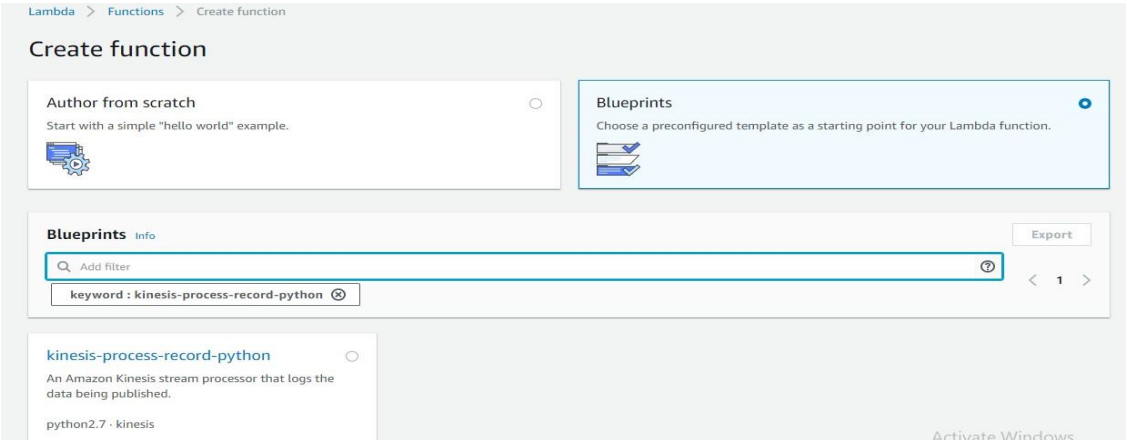


As you can see I create a Kinesis stream with the name 'mystream'



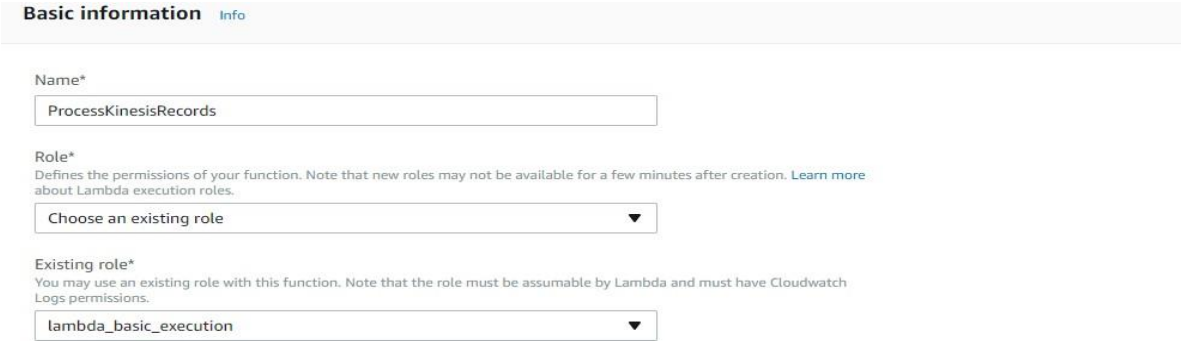
Step 2:Create a Lambda function

Now go back to AWS Dashboard and click on Lambda. Then, click create function. Now select blueprints and in the search bar type 'kinesis-process-record-python'

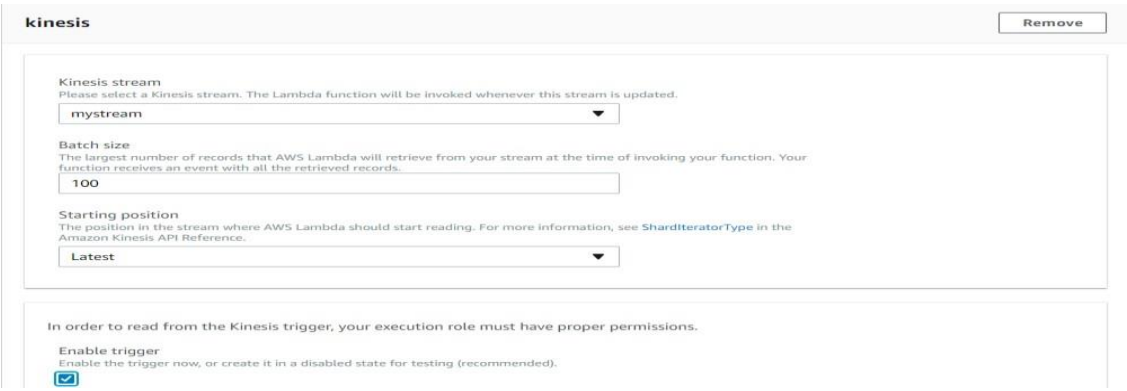


Now click the 'kinesis-process-record-python' blueprint.

Now in the basic information. Fill out the information as shown in screenshot.



Now in the Kinesis stream choose the stream name that you've created in the step 1.



Scroll down and examine the Lambda function before you click create function

Runtime
Python 2.7

```python
from __future__ import print_function

import base64
import json

print('Loading function')


def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))
    for record in event['Records']:
        # Kinesis data is base64 encoded so decode here
        payload = base64.b64decode(record['kinesis']['data'])
        print("Decoded payload: " + payload)
    return 'Successfully processed {} records.'.format(len(event['Records']))
```

This function will get triggered whenever data is sent to Kinesis.

<u>Step 4: Test the Lambda function</u>

Now click Test and which opens configure test event dialogue box. In the event name give 'stream'. Then click create and click Test.

You'll see Execution result: succeeded

Lambda > Functions > ProcessKinesisRecords

ARN - arn:aws:lambda:us-west-2:562705734084:function:ProcessKinesisRecords

ProcessKinesisRecords

Qualifiers ▼ | Actions ▼ | stream ▼ | Test | Save

Congratulations! Your Lambda function "ProcessKinesisRecords" has been successfully created and configured with mystream as a trigger in a disabled state. We recommend testing the function behavior before enabling the trigger.

Execution result: succeeded (logs)

▶ **Details**

Now expand details button to see the Summary and output.

The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
"Successfully processed 1 records."
```

Summary

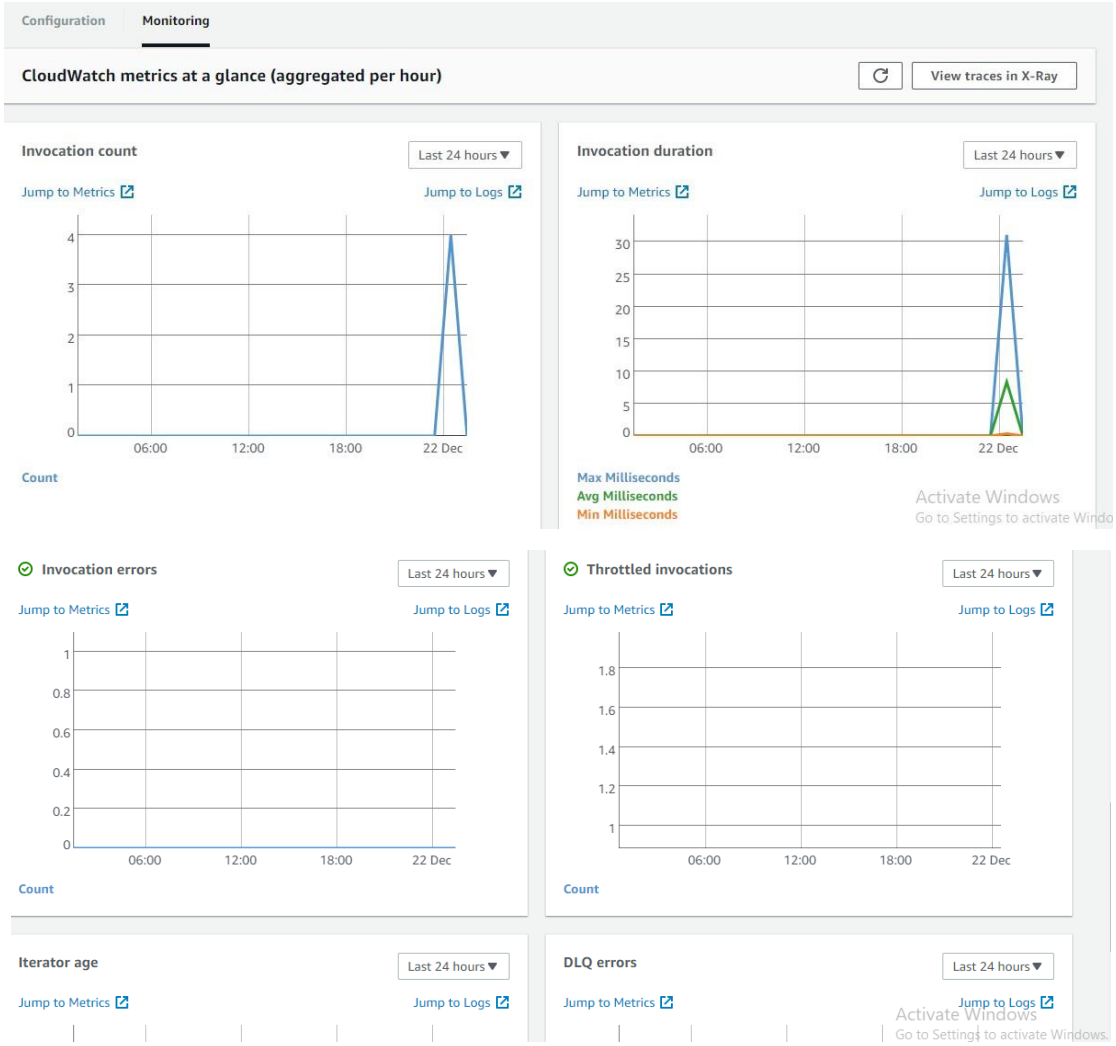
Code SHA-256	9OKEqV0MBZbUecqih4+mr4OOC9asu5qQjC0I41uMuQ4=	**Request ID**	fecd2c5e-e684-11e7-b592-b9bf6b3fcf72
Duration	31.09 ms	**Billed duration**	100 ms
Resources configured	128 MB	**Max memory used**	19 MB

Log output

The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. Click here to view the CloudWatch log group.

```
START RequestId: fecd2c5e-e684-11e7-b592-b9bf6b3fcf72 Version: $LATEST
Decoded payload: Hello, this is a test 123.
END RequestId: fecd2c5e-e684-11e7-b592-b9bf6b3fcf72
REPORT RequestId: fecd2c5e-e684-11e7-b592-b9bf6b3fcf72  Duration: 31.09 ms      Billed Duration: 100 ms         Memory Size: 128 MB     Max Memory Used: 19 MB
```

Now click Test button 3 more times waiting few seconds after each test,which will generate test data in cloudwatch


Configuration
Monitoring
CloudWatch metrics at a glance (aggregated per hour)
View traces in X-Ray
Invocation count
Last 24 hours
Jump to Metrics
Jump to Logs
4
3
2
1
0
06:00
12:00
18:00
22 Dec
Count
Invocation duration
Last 24 hours
Jump to Metrics
Jump to Logs
30
25
20
15
10
5
0
06:00
12:00
18:00
22 Dec
Max Milliseconds
Avg Milliseconds
Min Milliseconds
Invocation errors
Last 24 hours
Jump to Metrics
Jump to Logs
1
0.8
0.6
0.4
0.2
0
06:00
12:00
18:00
22 Dec
Count
Throttled invocations
Last 24 hours
Jump to Metrics
Jump to Logs
1.8
1.6
1.4
1.2
1
06:00
12:00
18:00
22 Dec
Count
Iterator age
Last 24 hours
Jump to Metrics
Jump to Logs
DLQ errors
Last 24 hours
Jump to Metrics
Jump to Logs
Activate Windows
Go to Settings to activate Windows.

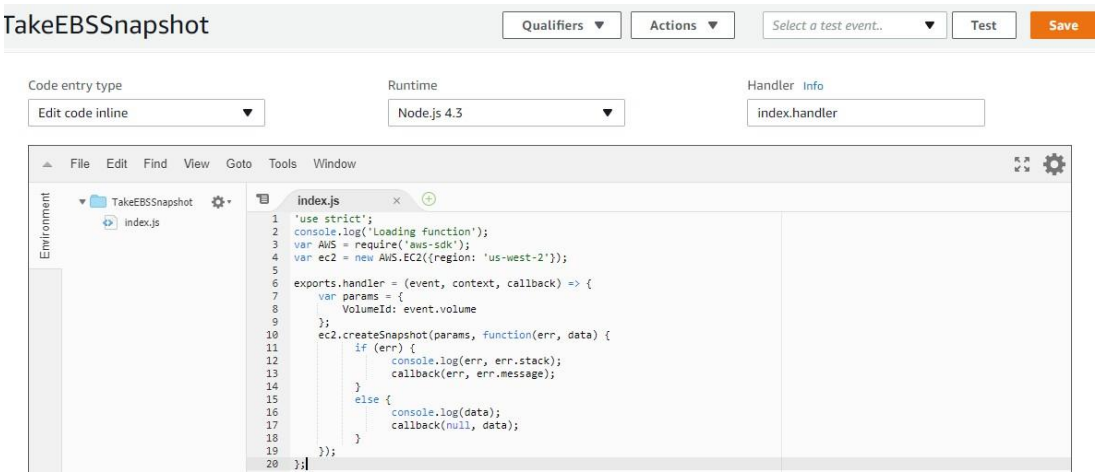
Automate capturing EBS snapshot using AWS Lambda

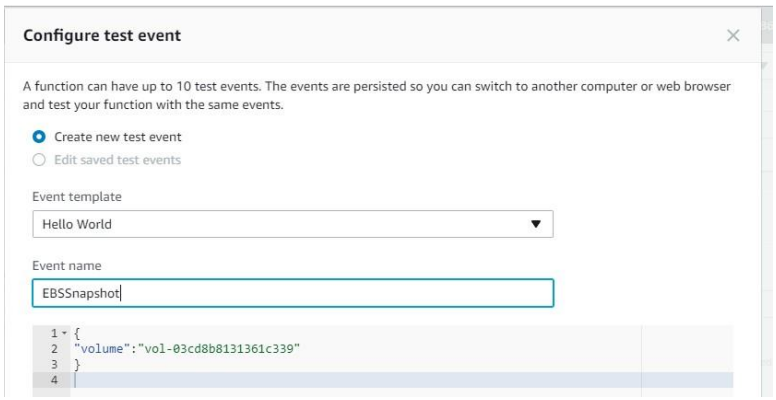
Step 1: Create an IAM role

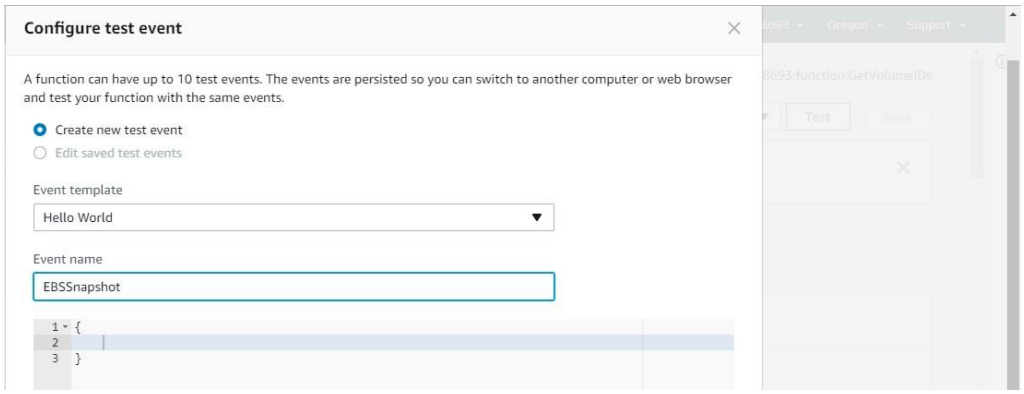
Create an IAM Role(Lambda service role) and attach EC2FullAccess and CloudWatchFullAccess policies and save it

Step 2: Create a Lambda Function

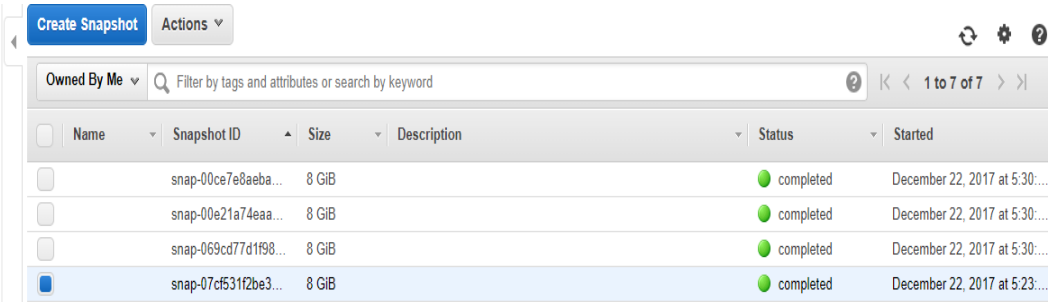
Click on AWS Lambda then click create function and click 'Author Function from scratch'







Step 3: Check for EBS snapshot



Amazon API Gateway

What is an API?

API is a set of rules which one software talks with other software ,just like client/server interaction.

REST API/RESTful API:

REST means State Transfer, which means to transfer state from client to server in every request.

REST is a web standards based architecture and uses HTTP Protocol for data communication.

In REST architecture, a REST Server simply provides access to resources and the REST client access those resources. Here each resource is identified by URIs.

REST uses various representations to represent a resource like Text, JSON and XML. JSON is very popular.

HTTP Methods

The following HTTP methods are most commonly used in a REST based architecture.

- **GET** – Provides a read only access to a resource.
- **PUT** – Used to create a new resource.
- **DELETE** – Used to remove a resource.
- **POST** – Used to update an existing resource or create a new resource.
- **OPTIONS** – Used to get the supported operations on a resource.

REST Constraints:

REST constraints are design rules that are applied to establish the distinct characteristics of the REST architectural style.

1. Uniform interface

Resource interaction between client and servers shouldn't be binding on the type of machine(both server and client)

2. Client–server:

Client is the one which requests Server for the resource,

Server is the one which sends data in response to the request it recieves.

3. Stateless:

Each request from client to server must contain all of the information necessary to understand the request, and server shouldn't rely on previous requests.

By stateless it means that the server does not store any state about the client session on the server side. The client session is stored on the client.

4. Cacheability:

The client will cache data to improve network effecieincy.

5. Layered System:

If you deploy the APIs on server A, and store data on server B and authenticate requests in Server C, for example. A client cannot ordinarily tell whether it is connected directly to the end server, or to an intermediary .In sum, Client should know only resource URIs and that's all.

6.Code on demand(optional):

Code on demand is any technology that sends executable software code from a server computer to a client computer upon request from the client's software.

Scenario:

You're signing into an APP and that APP requests your info through the following API's:

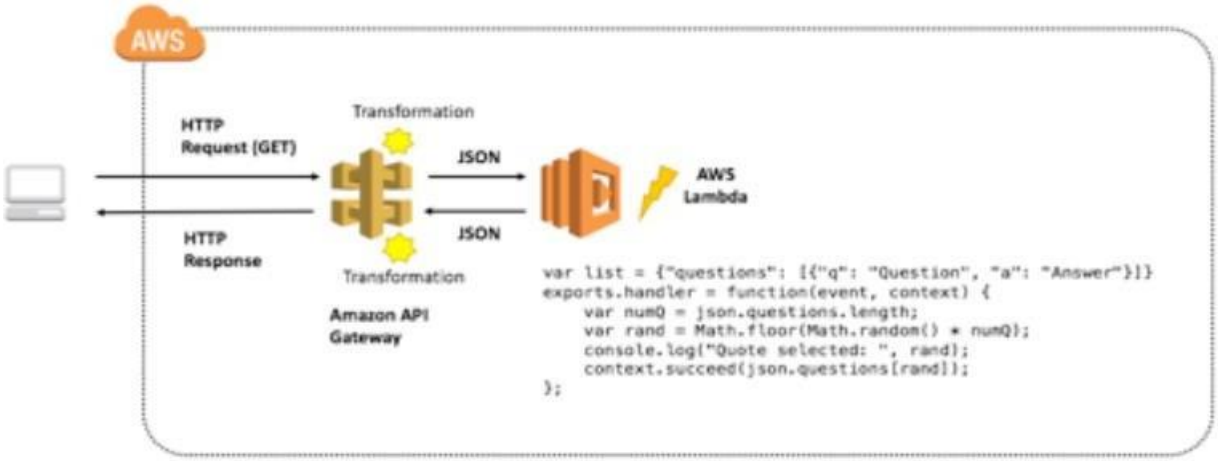
POST: Now you used Facbook API to log in,which POST your credentials upon log in

GET: That app the GET's your data from Facebook API when you click Allow

Amazon API Gateway is a fully managed service that it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. With a few clicks in the AWS Management Console, you can create an API that acts as a "front door" for applications to access data, business logic, or functionality from your back-end services, such as makes workloads running on Amazon Elastic Compute Cloud (Amazon EC2), code running on AWS Lambda, or any Web application.

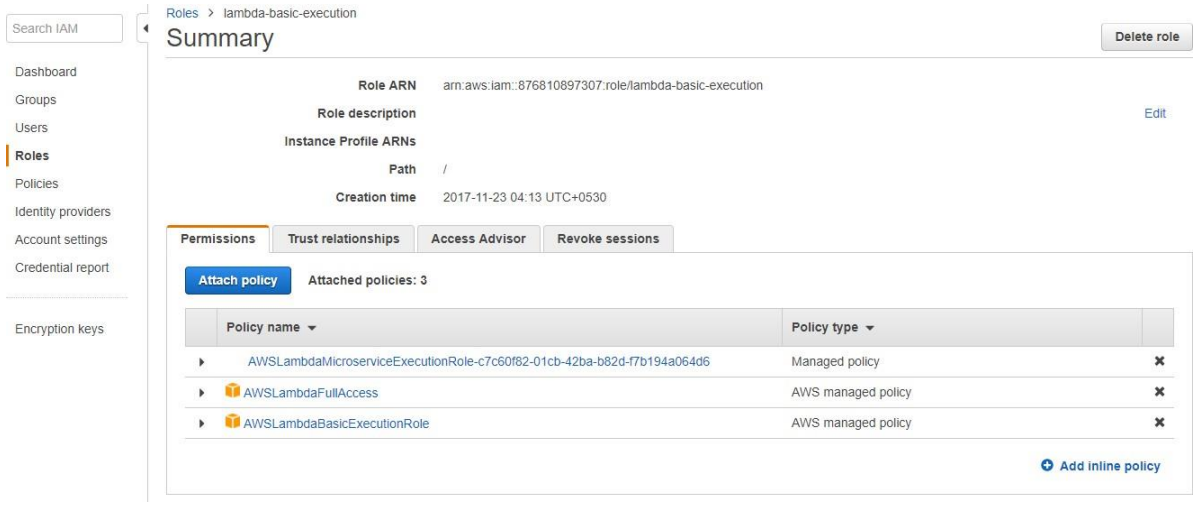
It provides a web-based interface for building, deploying, managing and monitoring your APIs. The Amazon API Gateway console provides a comprehensive UI to guide you through the process of creating an API.

Lab 2: In this lab we'll create a simple FAQ microservice. The micro-service will return a JSON object containing a random question and answer pair using API gateway endpoint that invokes lambda function.

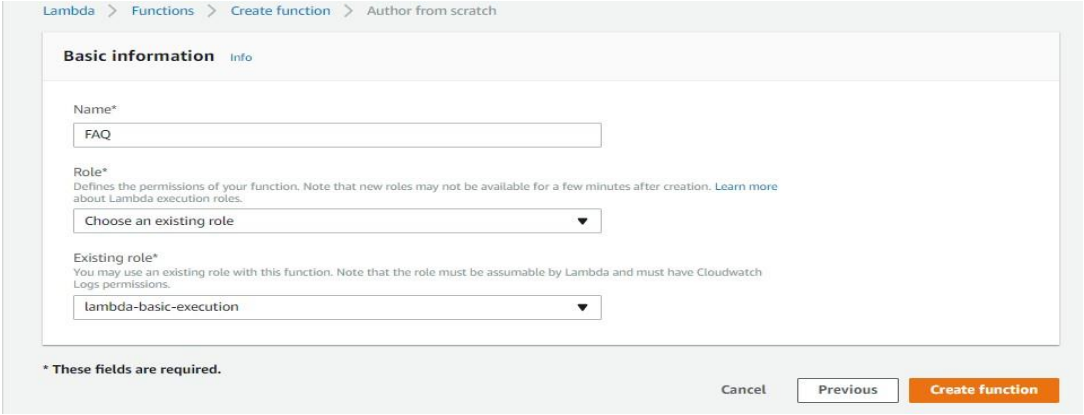


Step 1: Create an IAM role and attach following policies.

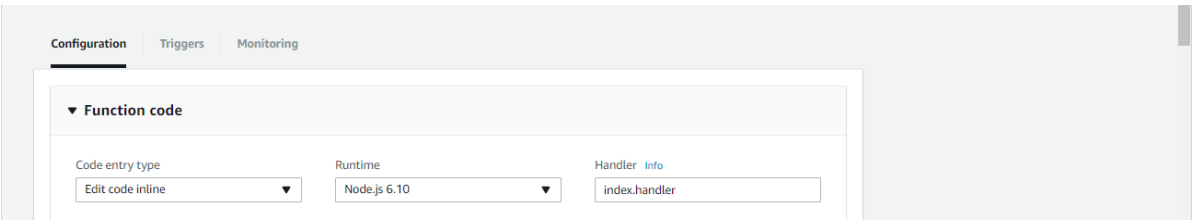
Name the role as lambda-basic-execution



Step 2: Now go to AWS Lambda and Click create function and then click **'Author from Scratch'**



Step 3: In the function code. Set the following settings:



Now copy the following json code and paste it in the code area as show:

You can download the code from here:

https://s3-us-west-2.amazonaws.com/jsoncode/json+code.txt

Dashboard

Functions

index.js

```
2   "service": "lambda",
3   "reference": "https://aws.amazon.com/lambda/faqs/",
4   "questions": [{
5     "q": "What is AWS Lambda?",
6     "a": "AWS Lambda lets you run code without provisioning or managing servers. You pay only t
7   },{
8    "q":"What events can trigger an AWS Lambda function?",
9    "a":"You can use AWS Lambda to respond to table updates in Amazon DynamoDB, modifications to
10  },{
11   "q":"When should I use AWS Lambda versus Amazon EC2?",
12   "a":"Amazon Web Services offers a set of compute services to meet a range of needs. Amazon E
13  },{
14    "q":"What kind of code can run on AWS Lambda?",
15    "a":"AWS Lambda offers an easy way to accomplish many activities in the cloud. For example,
16  },{
17    "q":"What languages does AWS Lambda support?",
18    "a":"AWS Lambda supports code written in Node.js (JavaScript), Python, and Java (Java 8 cor
19  },{
20    "q":"Can I access the infrastructure that AWS Lambda runs on?",
21    "a":"No. AWS Lambda operates the compute infrastructure on your behalf, allowing it to perf
22  },{
23    "q":"How does AWS Lambda isolate my code?",
24    "a":"Each AWS Lambda function runs in its own isolated environment, with its own resources
25  },{
26    "q":"How does AWS Lambda secure my code?",
27    "a":"AWS Lambda stores code in Amazon S3 and encrypts it at rest. AWS Lambda performs addit
28  },{
29    "q":"What is an AWS Lambda function?",
30    "a":"The code you run on AWS Lambda is uploaded as a Lambda function. Each function has as:
31
```

Now expand the basic settings and in the description, give this text

Basic settings

Memory (MB) Info
Your function is allocated CPU proportional to the memory configured.

128 MB

Timeout Info

0 min 3 sec

Description

Provide a random FAQ

Now click on triggers and + trigger

Add trigger

Configure your Lambda function **FAQ** to respond to events from the selected trigger. Click on the box below to select your trigger type.

Lambda

Cancel Submit

Now from the drop-down menu select API gateway and fill the following info as shown:

Warning: Your API endpoint will be publicly available and can be invoked by all users.

We'll set up an API Gateway endpoint with a proxy integration type (learn more about the input and output format for your function). Any method (GET, POST, etc.) will trigger your Lambda function. To set up more advanced method mappings or subpath routes, visit Amazon API Gateway console.

API name
The name used to identify your API.
LambdaMicroservice — Enter value

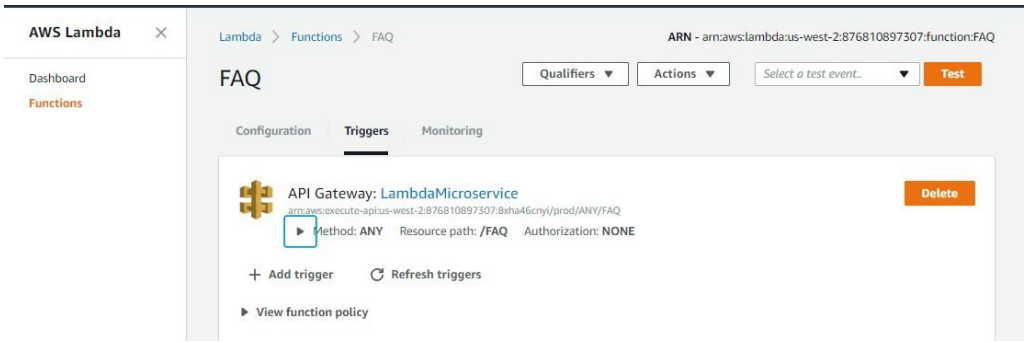
Deployment stage
The name of your API's deployment stage.
prod — Enter value

Security
Configure the security mechanism for your API endpoint.
Open

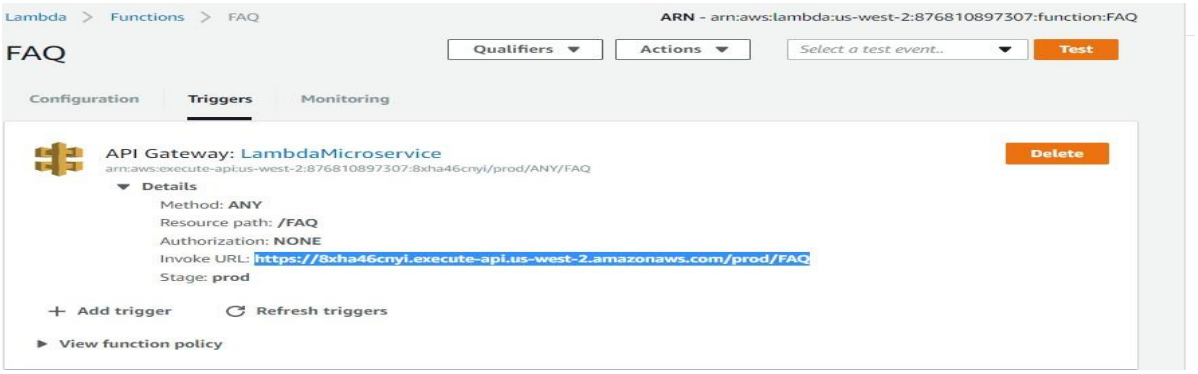
Lambda will add the necessary permissions for Amazon API Gateway to invoke your Lambda function from this trigger. Learn more about the Lambda permissions model.

Cancel Submit

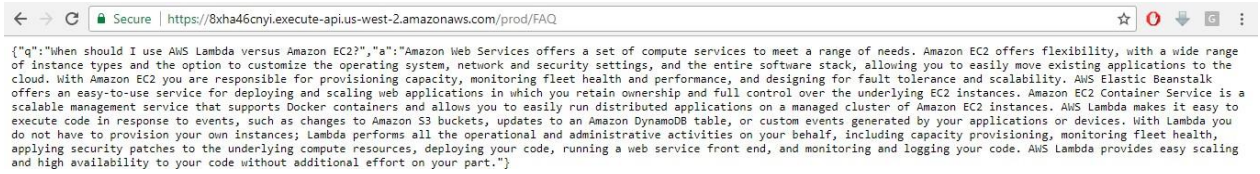
Now under the API gateway. Click on Method: Any



Now copy the invole URL and paste it in a different browser



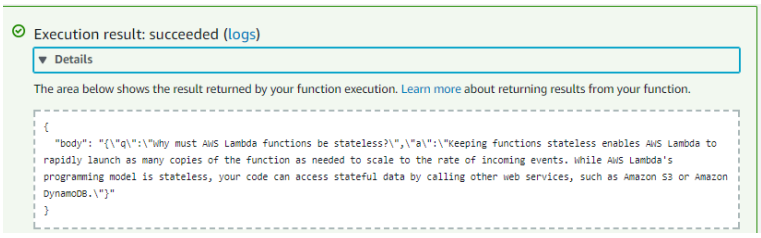
You should be able to see a random entry like this one for ever refresh.



Now click on Test button. For the event name, type BasicTest and in the code editor, delete all the code and insert only.{}

Now save it and click Test

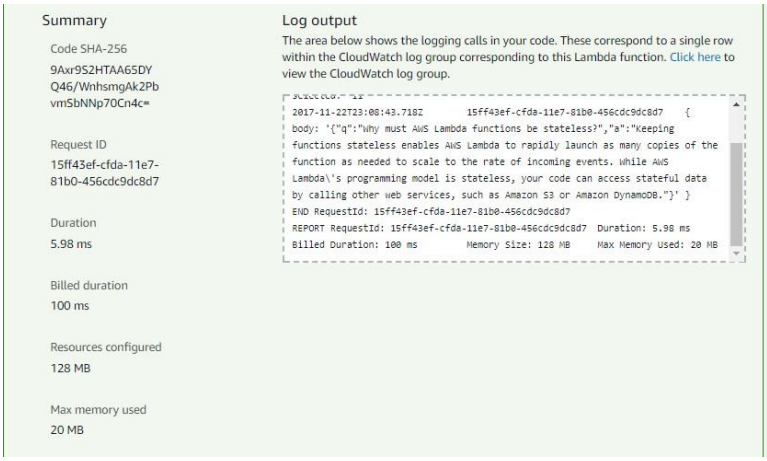
Now expand Details button You will see the FAQ entry wrapped inside



Now below that you find 2 things:

1.Summary display

2.Log output



Now click on blue link click here which will take you to Cloudwatch logs

Click on the event stream and you can see the streaming data captured for logs

You can find the event data aka logs.

Filter events

all 30s 5m 1h 6h 1d 1w custom

	Time (UTC +00:00)	Message
	2017-11-22	
		No older events found at the moment. Retry.
▸	23:01:54	START RequestId: 22271a05-cfd9-11e7-a64d-0554014bb2d5 Version: $LATEST
▸	23:01:54	2017-11-22T23:01:54.920Z 22271a05-cfd9-11e7-a64d-0554014bb2d5 Quote selected: 16
▸	23:01:54	2017-11-22T23:01:54.920Z 22271a05-cfd9-11e7-a64d-0554014bb2d5 { body: '{"q":"Which versions of Python are supported?","a":"Lambda provides a Python 2.7-compat
▸	23:01:54	END RequestId: 22271a05-cfd9-11e7-a64d-0554014bb2d5
▸	23:01:54	REPORT RequestId: 22271a05-cfd9-11e7-a64d-0554014bb2d5 Duration: 80.59 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 20 MB
▸	23:01:58	START RequestId: 24679405-cfd9-11e7-a2dd-f3b942c6732e Version: $LATEST
▸	23:01:58	2017-11-22T23:01:58.379Z 24679405-cfd9-11e7-a2dd-f3b942c6732e Quote selected: 9
▸	23:01:58	2017-11-22T23:01:58.379Z 24679405-cfd9-11e7-a2dd-f3b942c6732e { body: '{"q":"Will AWS Lambda reuse function instances?","a":"To improve performance, AWS Lamb
▸	23:01:58	END RequestId: 24679405-cfd9-11e7-a2dd-f3b942c6732e
▸	23:01:58	REPORT RequestId: 24679405-cfd9-11e7-a2dd-f3b942c6732e Duration: 0.88 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 20 MB
▸	23:04:47	START RequestId: 89471787-cfd9-11e7-85a1-7994d9d9bb0f Version: $LATEST
▸	23:04:47	2017-11-22T23:04:47.638Z 89471787-cfd9-11e7-85a1-7994d9d9bb0f Quote selected: 2
▸	23:04:47	2017-11-22T23:04:47.638Z 89471787-cfd9-11e7-85a1-7994d9d9bb0f { body: '{"q":"When should I use AWS Lambda versus Amazon EC2?","a":"Amazon Web Services o
▸	23:04:47	END RequestId: 89471787-cfd9-11e7-85a1-7994d9d9bb0f
▸	23:04:47	REPORT RequestId: 89471787-cfd9-11e7-85a1-7994d9d9bb0f Duration: 10.17 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 20 MB
▸	23:08:43	START RequestId: 15ff43ef-cfda-11e7-81b0-456cdc9dc8d7 Version: $LATEST
▸	23:08:43	2017-11-22T23:08:43.712Z 15ff43ef-cfda-11e7-81b0-456cdc9dc8d7 Quote selected: 11
▸	23:08:43	2017-11-22T23:08:43.718Z 15ff43ef-cfda-11e7-81b0-456cdc9dc8d7 { body: '{"q":"Why must AWS Lambda functions be stateless?","a":"Keeping functions stateless enabl
▸	23:08:43	END RequestId: 15ff43ef-cfda-11e7-81b0-456cdc9dc8d7
▸	23:08:43	REPORT RequestId: 15ff43ef-cfda-11e7-81b0-456cdc9dc8d7 Duration: 5.98 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 20 MB
		No newer events found at the moment. Retry.

Conclusion:

A Lambda function is created and it will be triggered when the client invokes it using an API gateway.