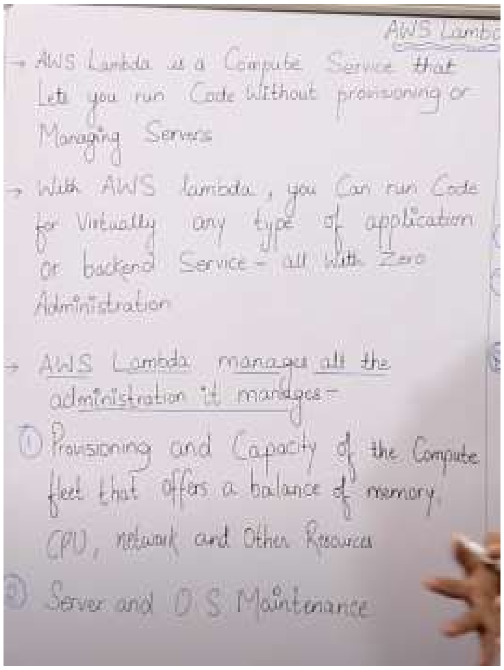
1.Serverless compute on AWS | What is AWS Lambda



AWS Lambo

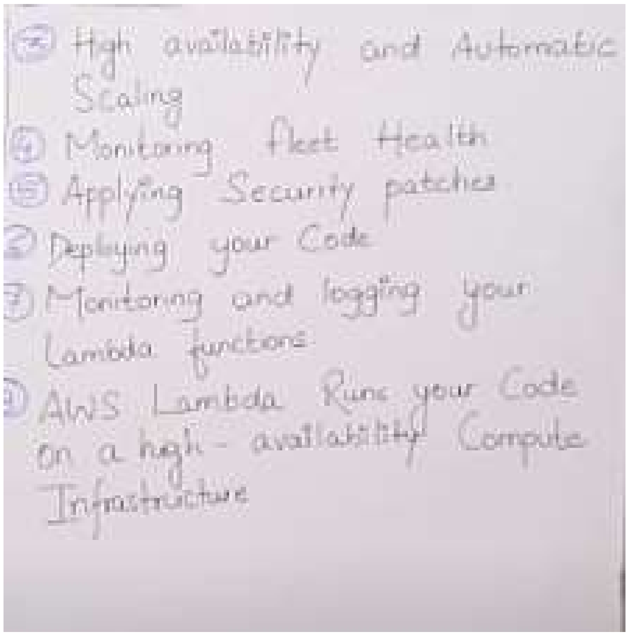
→ AWS Lambda is a Compute Service that Lets you run Code without provisioning or Managing Servers

→ With AWS lambda, you Can run Code for Virtually any type of application or backend Service - all with Zero Administration

→ AWS Lambda manages all the administration it manages -

① Provisioning and Capacity of the Compute fleet that offers a balance of memory, CPU, network and Other Resources

② Server and O S Maintenance



③ High availability and Automatic Scaling

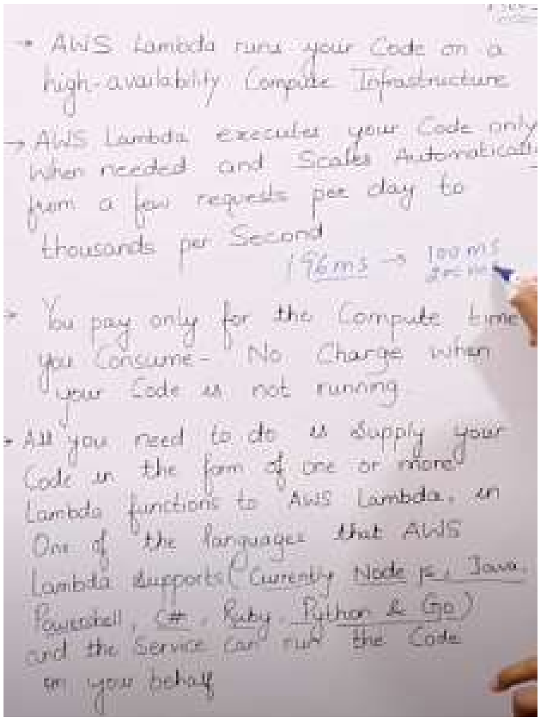
④ Monitoring fleet Health

⑤ Applying Security patches

⑥ Deploying your Code

⑦ Monitoring and logging your Lambda functions

⑧ AWS Lambda Runs your Code on a high-availability Compute Infrastructure



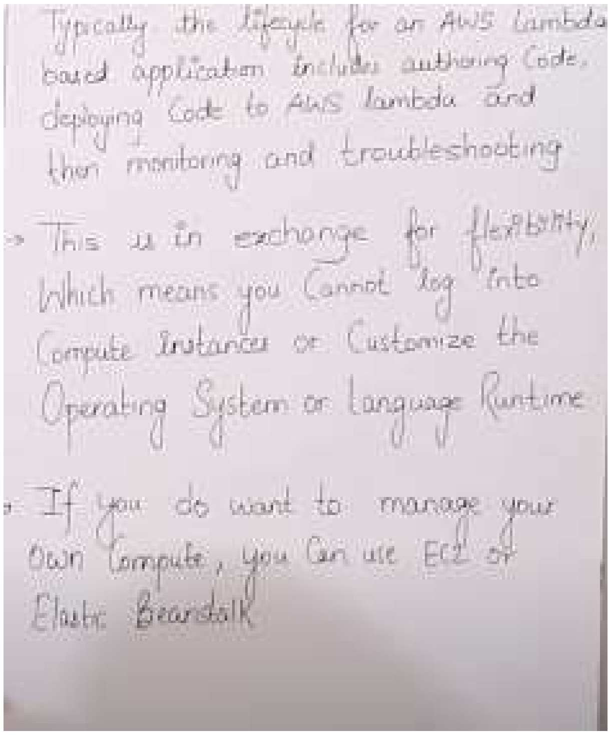
- AWS Lambda runs your Code on a high-availability Compute Infrastructure

→ AWS Lambda executes your Code only when needed and Scales Automatically from a few requests per day to thousands per Second

196ms → 100 ms [illegible]

→ You pay only for the Compute time you Consume - No Charge when your Code is not running

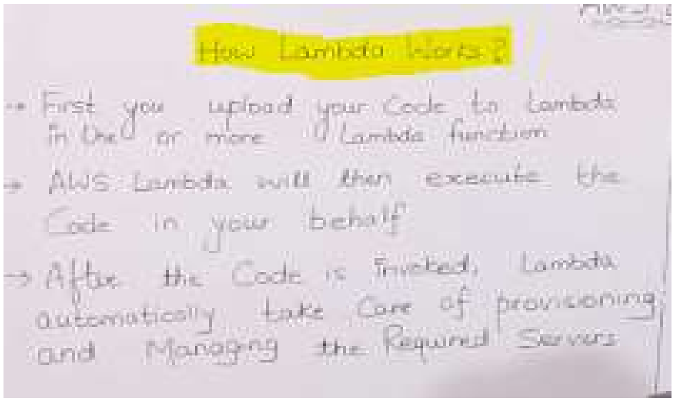
→ All you need to do is Supply your Code in the form of one or more Lambda functions to AWS Lambda, in One of the languages that AWS Lambda supports (Currently Node js, Java, Powershell, C#, Ruby, Python & Go) and the Service Can run the Code on your behalf



Typically the lifecycle for an AWS Lambda based application includes authoring Code, deploying Code to AWS lambda and then monitoring and troubleshooting

→ This is in exchange for flexibility, Which means you Cannot log into Compute Instances or Customize the Operating System or language Runtime

→ If you do want to manage your own Compute, you Can use EC2 or Elastic Beanstalk



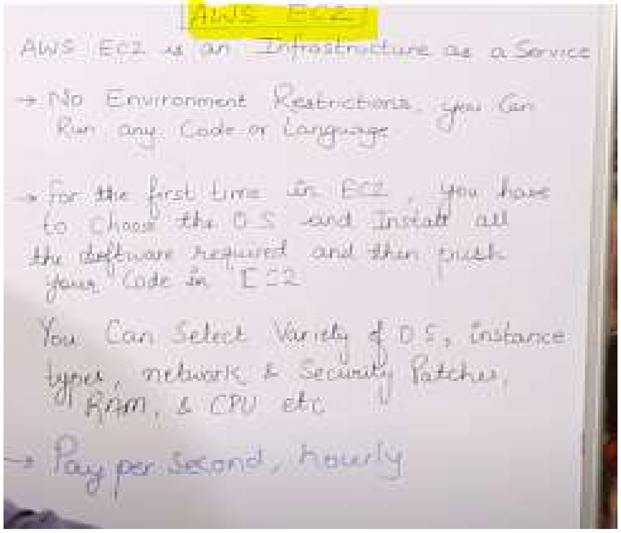
How Lambda Works ?

→ First you upload your Code to Lambda in One or more Lambda function

→ AWS Lambda will then execute the Code in your behalf

→ After the Code is invoked, Lambda automatically take Care of provisioning and Managing the Required Servers

Difference between AWS LAMBDA and AWS EC2 :



AWS EC2

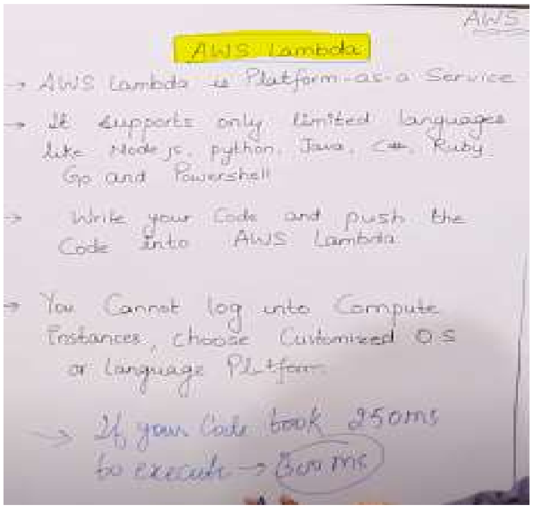
AWS EC2 is an Infrastructure as a Service

→ No Environment Restrictions, you Can Run any Code or Language

→ For the first time in EC2, you have to Choose the O.S and Install all the software required and then push your Code in EC2

You Can Select Variety of O.S, instance types, network & Security Patches, RAM, & CPU etc

→ Pay per Second, hourly



AWS Lambda

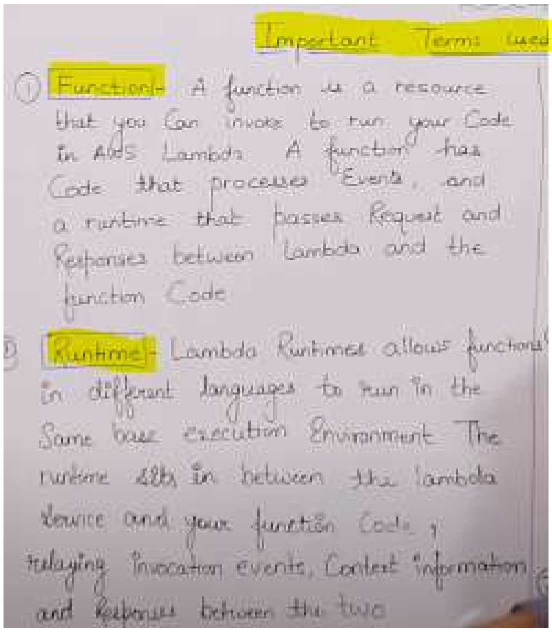
→ AWS Lambda is Platform-as-a Service

→ It supports only limited languages like Node js, python, Java, C#, Ruby Go and Powershell

→ Write your Code and push the Code into AWS Lambda

→ You Cannot log into Compute Instances, choose Customised O.S or language Platform

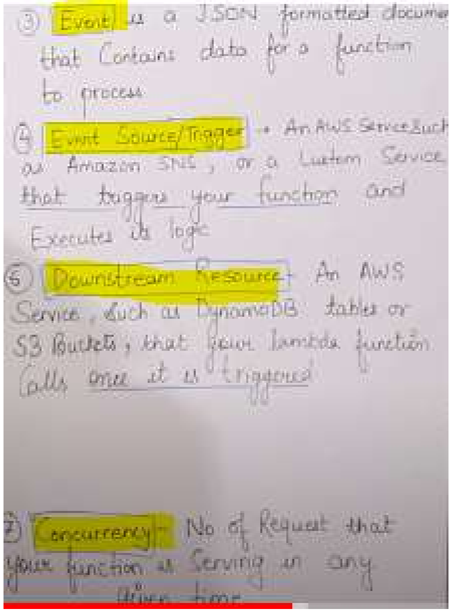
→ If your Code took 250ms to execute → 300 ms



Important Terms used

① Function:- A function is a resource that you Can invoke to run your Code in AWS Lambda. A function has Code that processes Events, and a runtime that passes Request and Responses between Lambda and the function Code

② Runtime:- Lambda Runtimes allow functions in different languages to run in the Same base execution Environment. The runtime sits in between the lambda service and your function Code, relaying Invocation events, Context information and Responses between the two



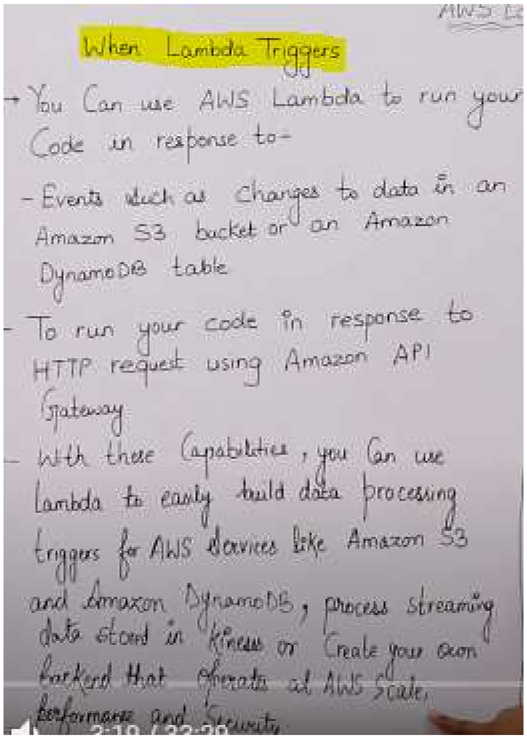
③ Event is a JSON formatted documen that Contains data for a function to process

④ Event Source/Trigger - An AWS Service such as Amazon SNS, or a Custom Service that triggers your function and Executes its logic

⑤ Downstream Resource - An AWS Service, such as DynamoDB tables or S3 Buckets, that your lambda function Calls once it is triggered

⑦ Concurrency:- No of Request that your function is Serving in any given time

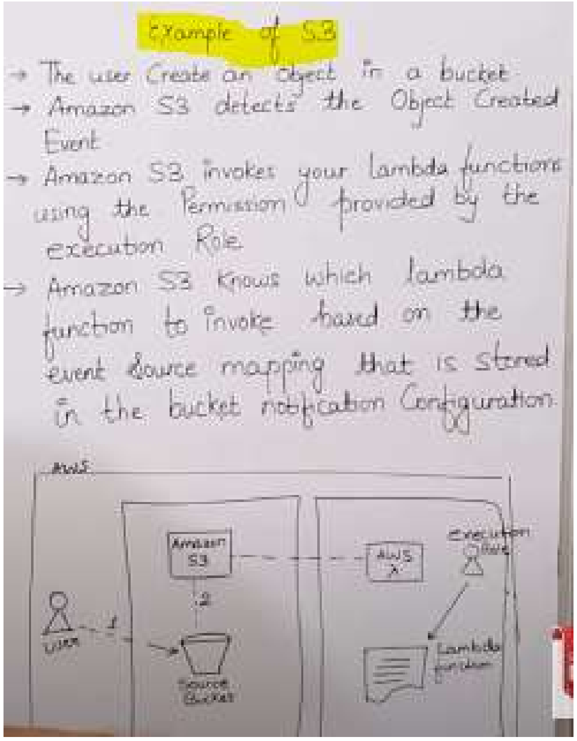
Invocation types | AWS Lambda Trigger | AWS Serverless



When Lambda Triggers

→ You Can use AWS Lambda to run your Code in response to-

- Events such as Changes to data in an Amazon S3 bucket or an Amazon DynamoDB table
- To run your code in response to HTTP request using Amazon API Gateway
- With these Capabilities, you Can use lambda to easily build data processing triggers for AWS Services like Amazon S3 and Amazon DynamoDB, process streaming data stored in Kinesis or Create your own backend that operates at AWS Scale, performance and Security



Example of S3

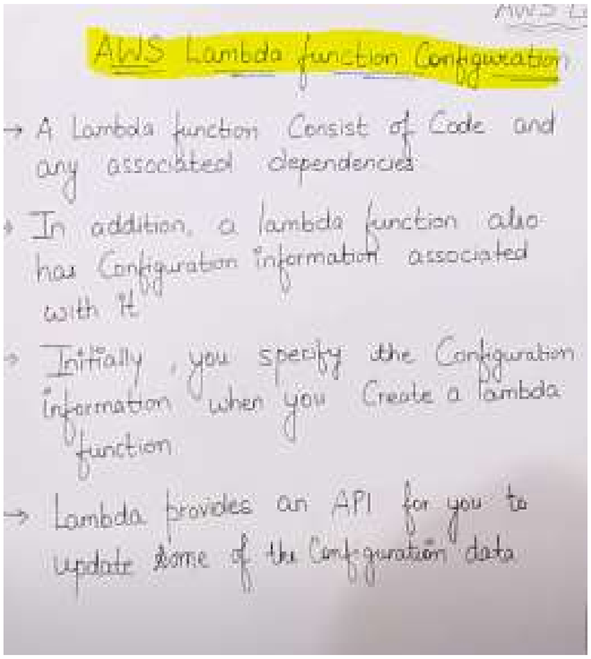
→ The user Create an object in a bucket

→ Amazon S3 detects the Object Created Event

→ Amazon S3 invokes your lambda functions using the Permission provided by the execution Role

→ Amazon S3 Knows which lambda function to invoke based on the event source mapping that is stored in the bucket notification Configuration.

AWS
Amazon S3
AWS λ
execution Role
2
1
User
Source Bucket
Lambda function



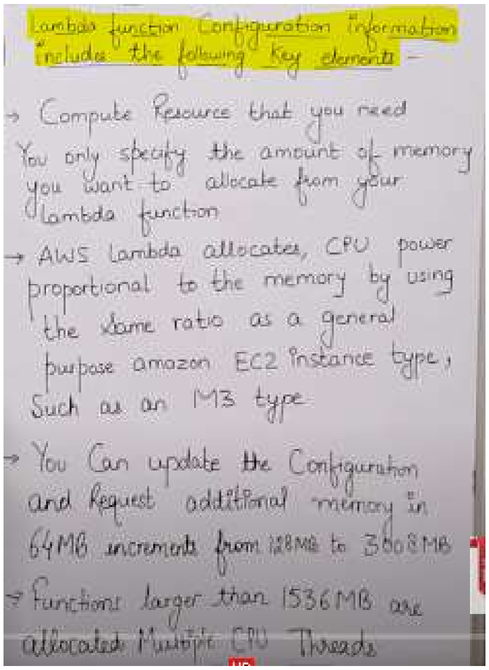
AWS Lambda function Configuration

→ A Lambda function Consist of Code and any associated dependencies

→ In addition, a lambda function also has Configuration information associated with it

→ Initially, you specify the Configuration information when you Create a lambda function

→ Lambda provides an API for you to update some of the Configuration data



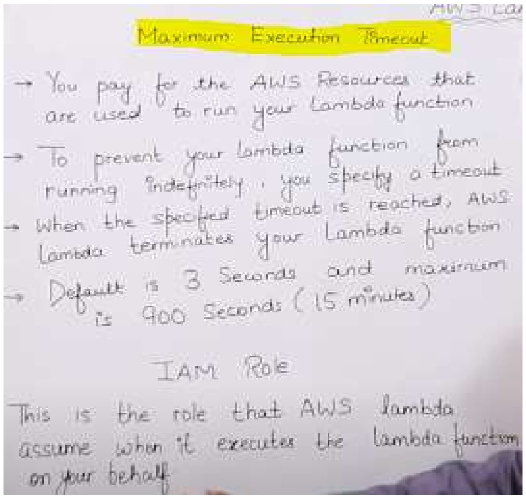
Lambda function Configuration information includes the following Key elements -

→ Compute Resource that you need You only specify the amount of memory you want to allocate from your lambda function

→ AWS lambda allocates, CPU power proportional to the memory by using the same ratio as a general purpose amazon EC2 instance type, Such as an M3 type

→ You Can update the Configuration and Request additional memory in 64MB increments from 128MB to 3008MB

→ Functions larger than 1536MB are allocated Multiple CPU Threads



Maximum Execution Timeout

→ You pay for the AWS Resources that are used to run your Lambda function

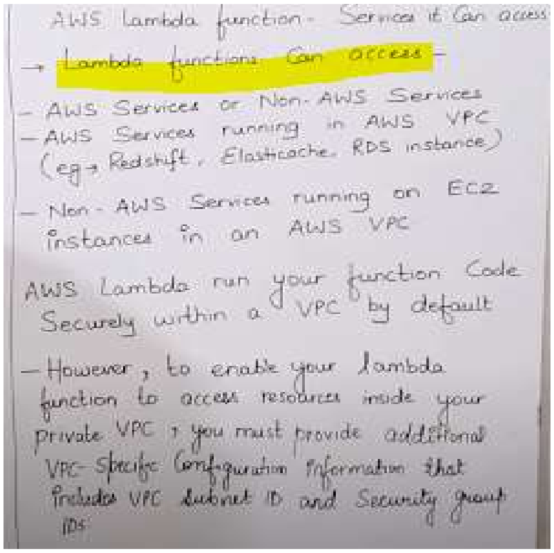
→ To prevent your lambda function from running indefinitely, you specify a timeout

→ When the specified timeout is reached, AWS Lambda terminates your Lambda function

→ Default is 3 Seconds and maximum is 900 Seconds (15 minutes)

IAM Role

This is the role that AWS lambda assume when it executes the lambda function on your behalf



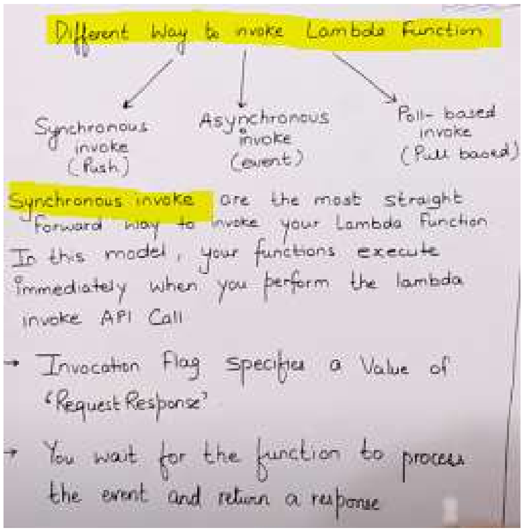
AWS lambda function - Services it Can access

→ Lambda functions Can access -

- AWS Services or Non-AWS Services
- AWS Services running in AWS VPC (eg → Redshift, Elasticache, RDS instance)
- Non-AWS Services running on EC2 instances in an AWS VPC

AWS Lambda run your function Code Securely within a VPC by default

- However, to enable your lambda function to access resources inside your private VPC, you must provide additional VPC-Specific Configuration information that includes VPC subnet ID and Security group IDs



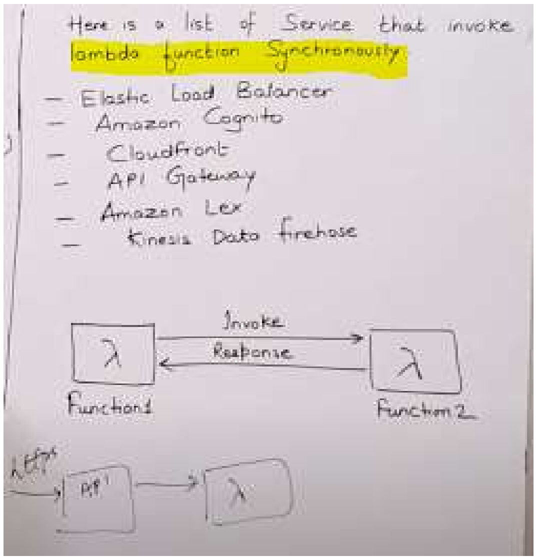
Different Way to invoke Lambda Function

Synchronous invoke (Push)
Asynchronous invoke (event)
Poll-based invoke (Pull based)

Synchronous invoke are the most straight forward way to invoke your Lambda Function. In this model, your functions execute immediately when you perform the lambda invoke API Call

→ Invocation Flag specifies a Value of 'Request Response'

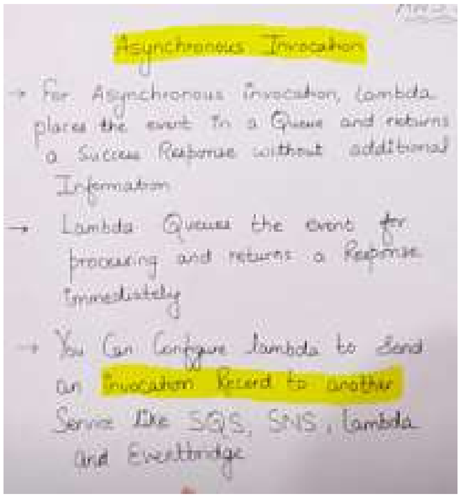
→ You wait for the function to process the event and return a response



Here is a list of Service that invoke lambda function Synchronously

- Elastic Load Balancer
- Amazon Cognito
- CloudFront
- API Gateway
- Amazon Lex
- Kinesis Data firehose

Invoke
Response
λ
λ
Function1
Function2
Let's
API
λ

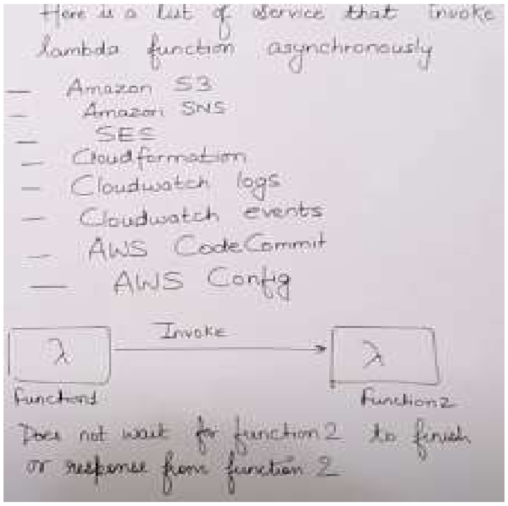


Asynchronous Invocation

→ For Asynchronous Invocation, Lambda places the event in a Queue and returns a Success Response without additional Information

→ Lambda Queues the event for processing and returns a Response immediately

→ You Can Configure lambda to send an Invocation Record to another Service like SQS, SNS, Lambda and EventBridge

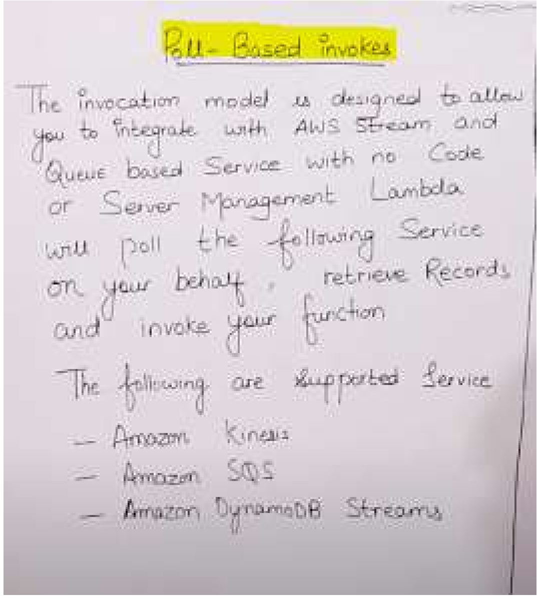


Here is a list of Service that invoke lambda function asynchronously

- Amazon S3
- Amazon SNS
- SES
- Cloudformation
- Cloudwatch logs
- Cloudwatch events
- AWS CodeCommit
- AWS Config

Invoke
λ
λ
Function1
Function2

Does not wait for function2 to finish or response from function 2

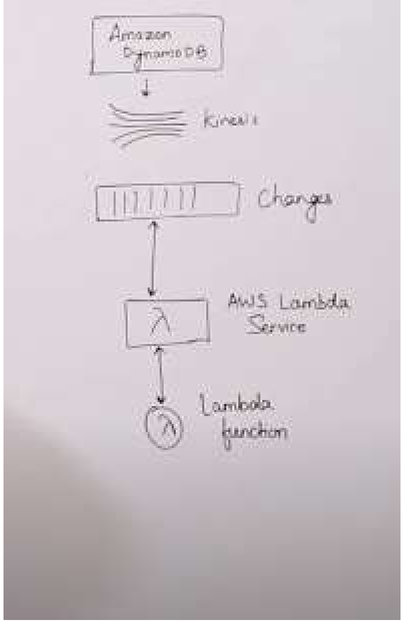


Poll- Based invokes

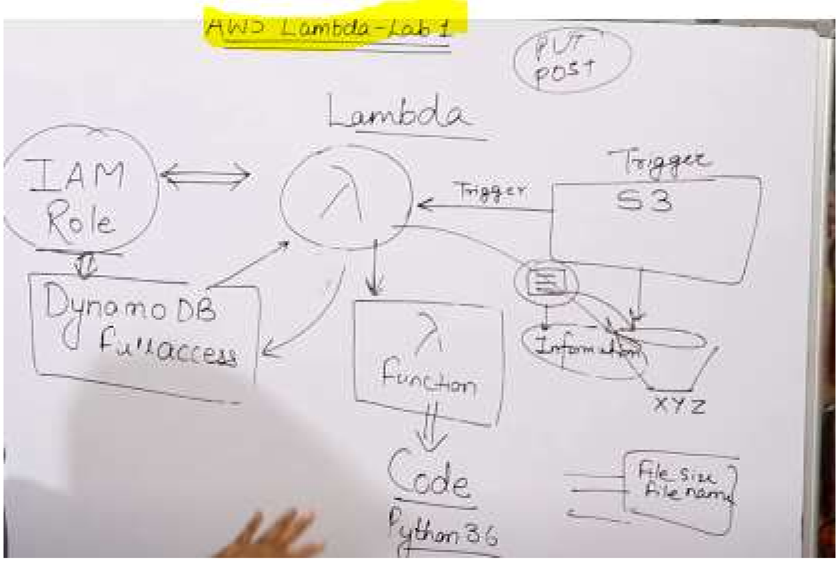
The invocation model is designed to allow you to integrate with AWS Stream and Queue based Service with no Code or Server Management Lambda will poll the following Service on your behalf , retrieve Records and invoke your function

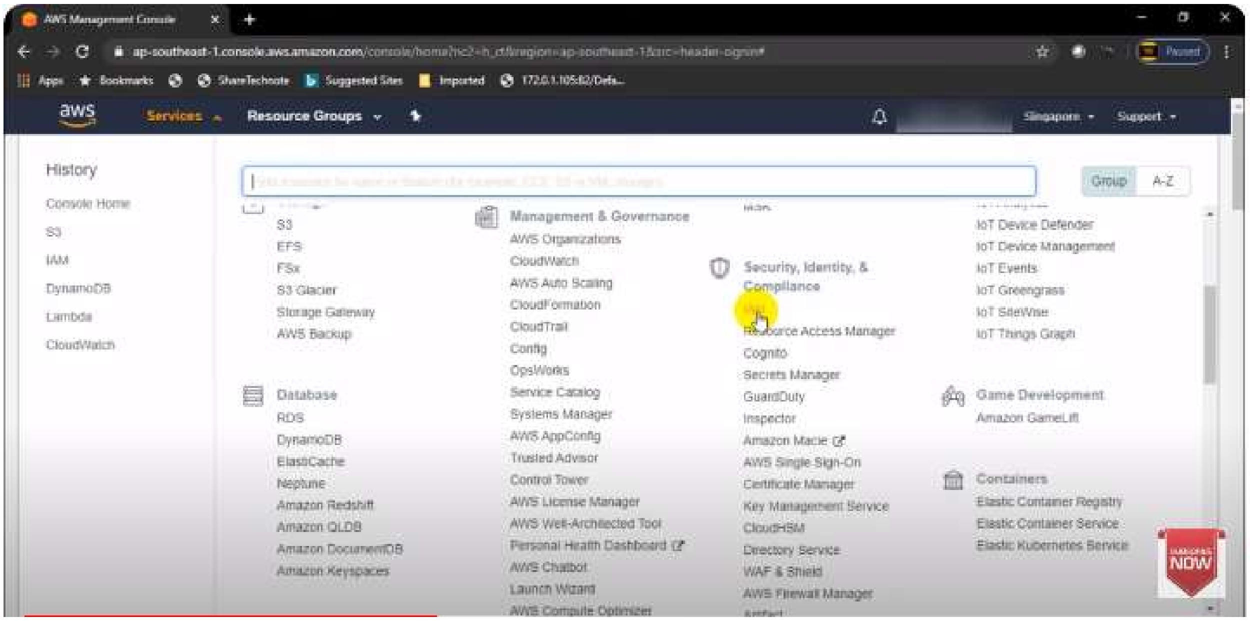
The following are supported Service

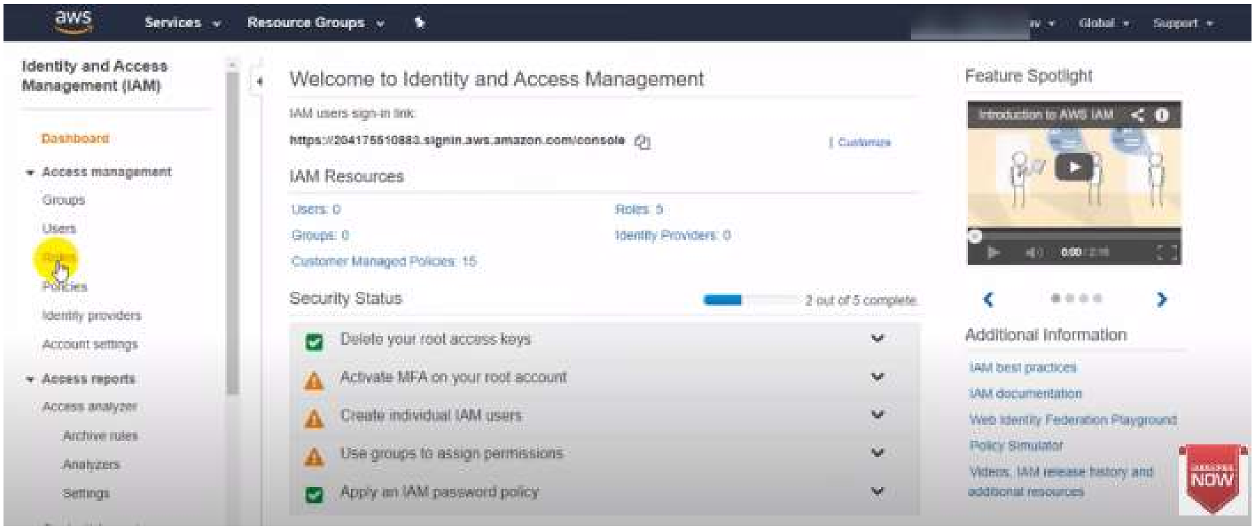
- Amazon Kinesis
- Amazon SQS
- Amazon DynamoDB Streams



Setup S3 Trigger with Lambda and dynamo DB







Welcome to Identity and Access Management

IAM users sign-in link:

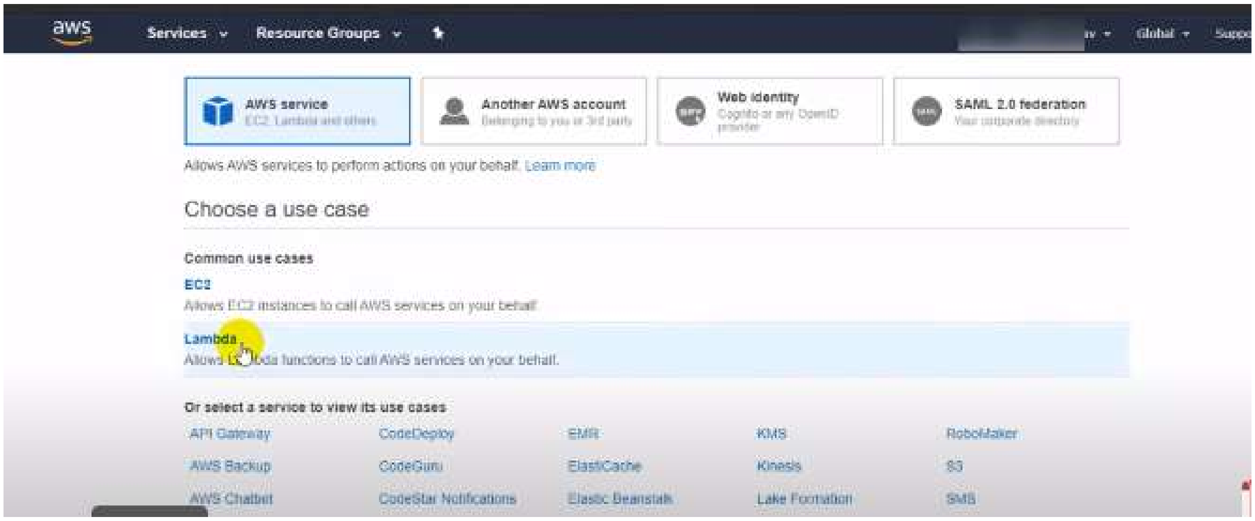
https://204175510880.signin.aws.amazon.com/console

IAM Resources

Users: 0
Roles: 5
Groups: 0
Identity Providers: 0
Customer Managed Policies: 15

Security Status 2 out of 5 complete.

- ✓ Delete your root access keys
- ⚠ Activate MFA on your root account
- ⚠ Create individual IAM users
- ⚠ Use groups to assign permissions
- ✓ Apply an IAM password policy



Allows AWS services to perform actions on your behalf. Learn more

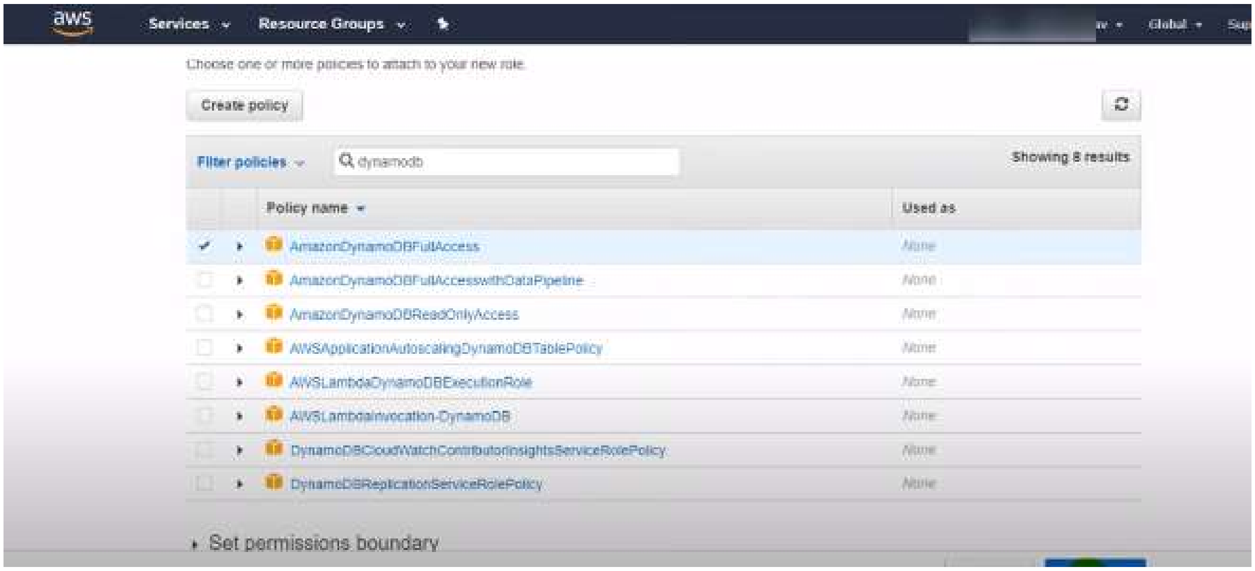
Choose a use case

Common use cases

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

Or select a service to view its use cases

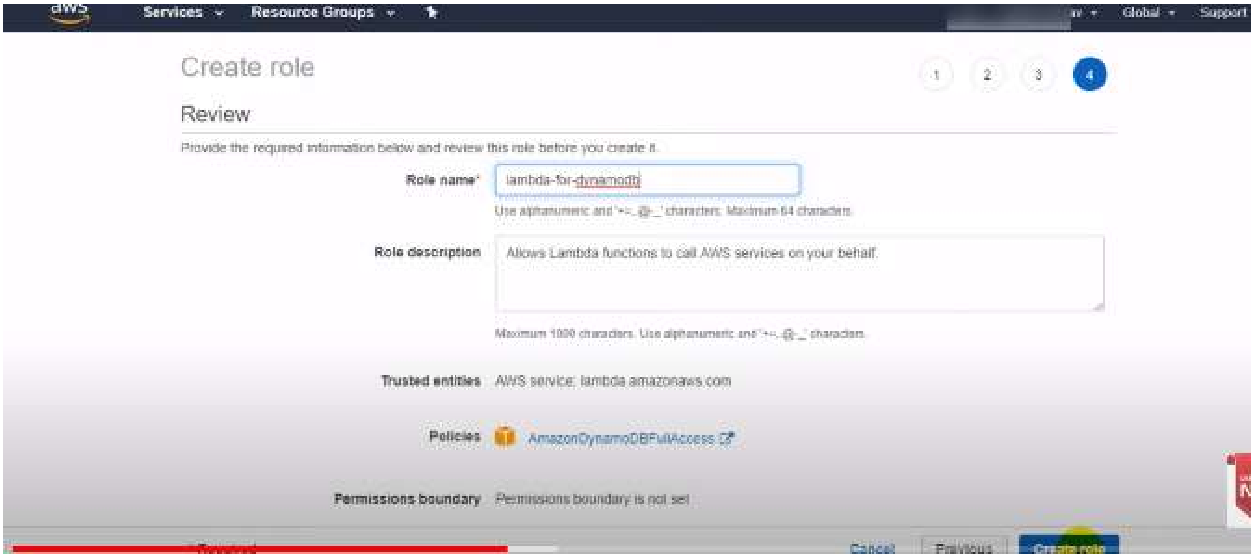


Choose one or more policies to attach to your new role.

🔍 dynamodb

		Policy name	Used as
✓	▸	AmazonDynamoDBFullAccess	None
□	▸	AmazonDynamoDBFullAccesswithDataPipeline	None
□	▸	AmazonDynamoDBReadOnlyAccess	None
□	▸	AWSApplicationAutoscalingDynamoDBTablePolicy	None
□	▸	AWSLambdaDynamoDBExecutionRole	None
□	▸	AWSLambdaInvocation-DynamoDB	None
□	▸	DynamoDBCloudWatchContributorInsightsServiceRolePolicy	None
□	▸	DynamoDBReplicationServiceRolePolicy	None

▸ Set permissions boundary



Create role

Review

Provide the required information below and review this role before you create it.

Role name* lambda-for-dynamodb

Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.

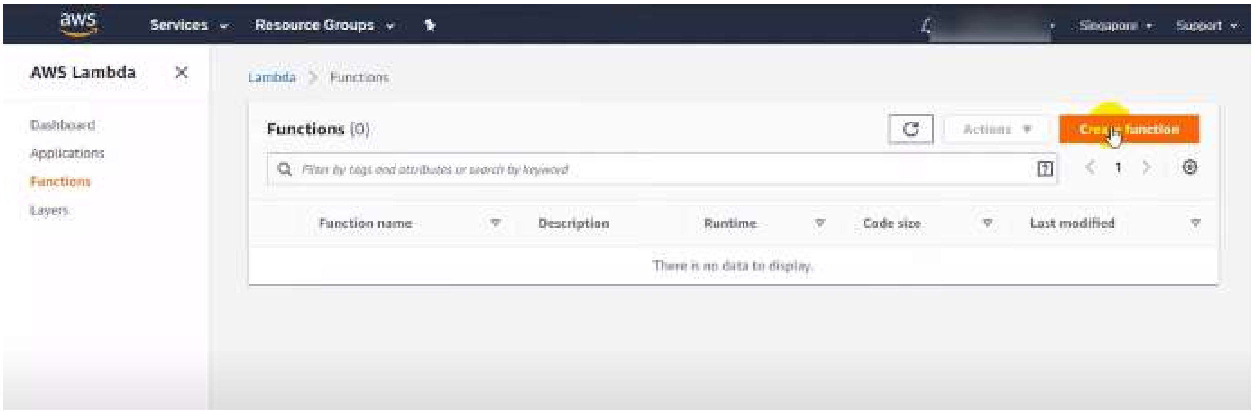
Role description Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Trusted entities AWS service: lambda.amazonaws.com

Policies AmazonDynamoDBFullAccess

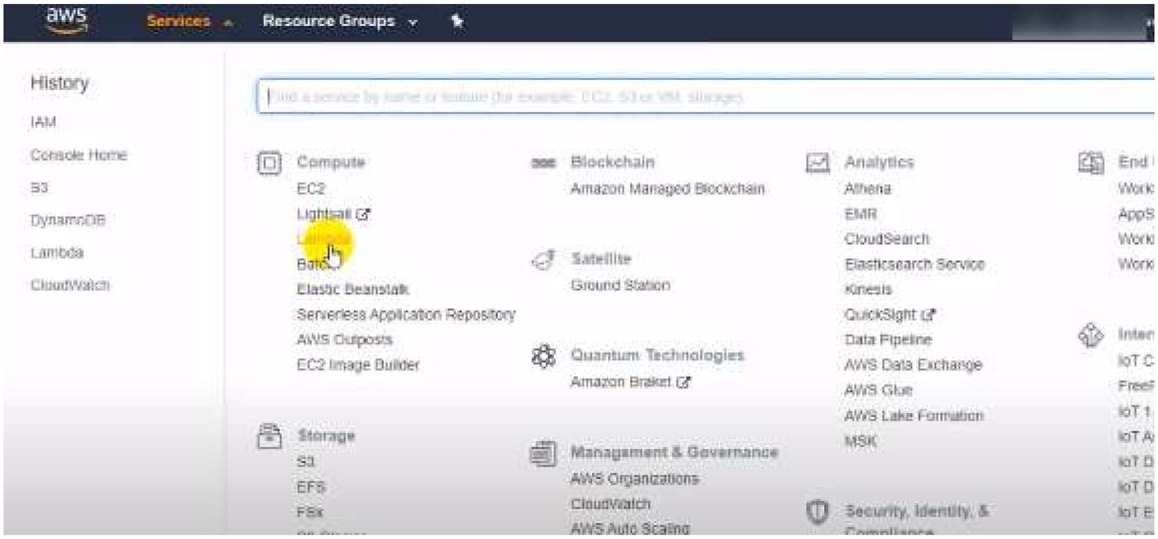
Permissions boundary Permissions boundary is not set

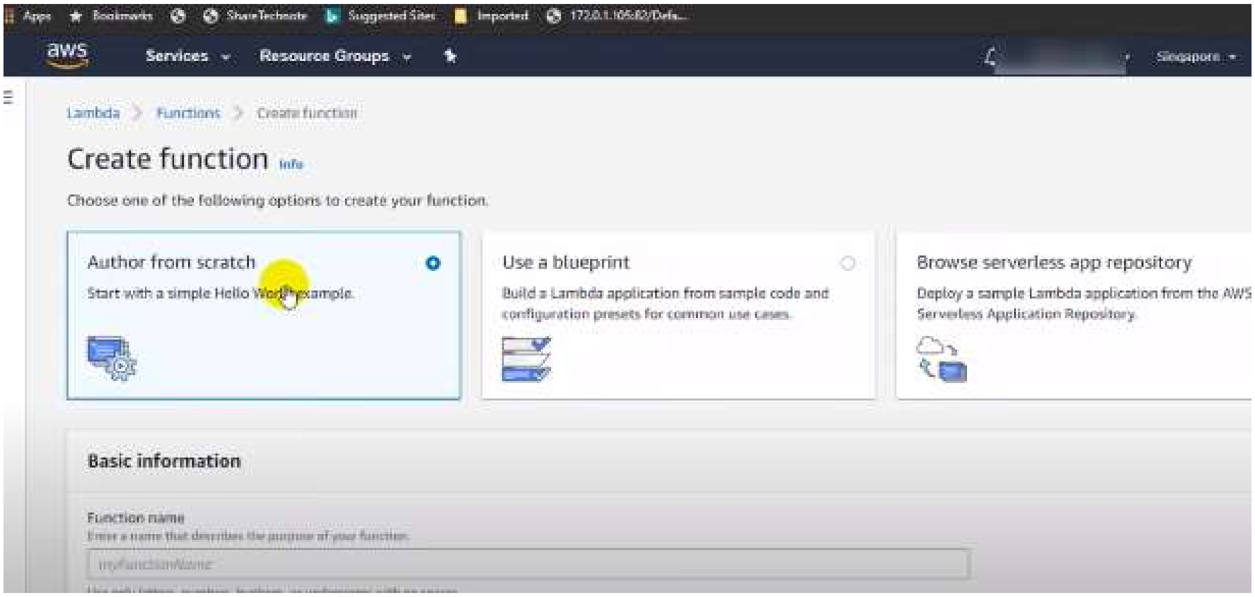


Functions (0)

Filter by tags and attributes or search by keyword

Function name	Description	Runtime	Code size	Last modified
There is no data to display.				





Lambda > Functions > Create function

Create function Info

Choose one of the following options to create your function.

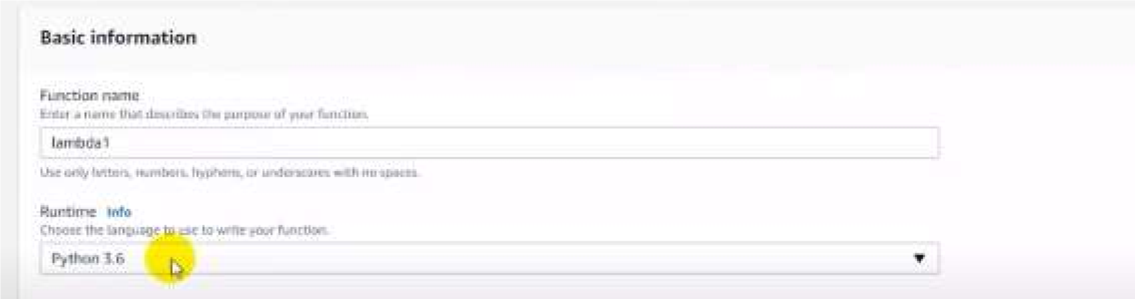
● Author from scratch
Start with a simple Hello World example.

○ Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.

Browse serverless app repository
Deploy a sample Lambda application from the AWS Serverless Application Repository.

Basic information

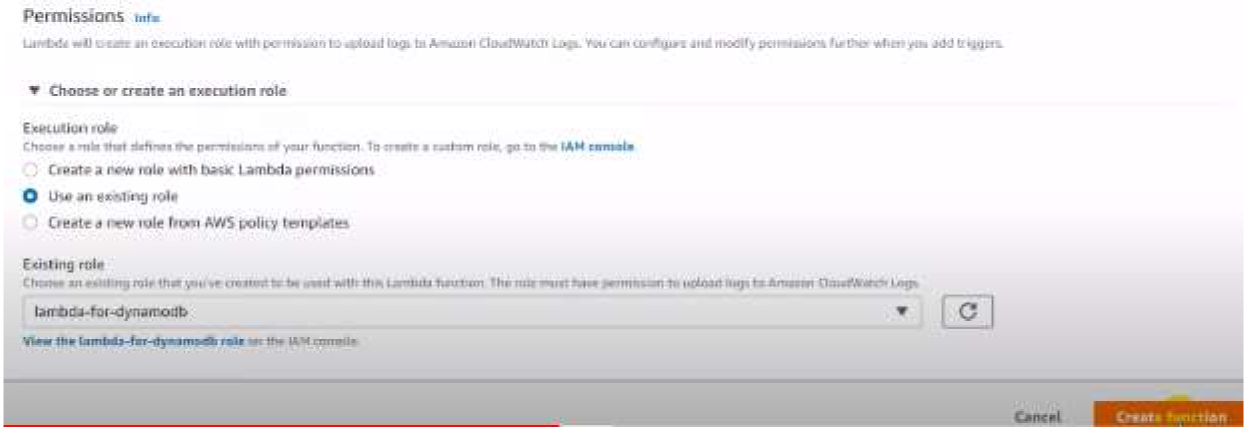
Function name
Enter a name that describes the purpose of your function.
myFunctionName



Basic information

Function name
Enter a name that describes the purpose of your function.
lambda1
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function.
Python 3.6 ▼



Permissions Info

Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ Choose or create an execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console

○ Create a new role with basic Lambda permissions

● Use an existing role

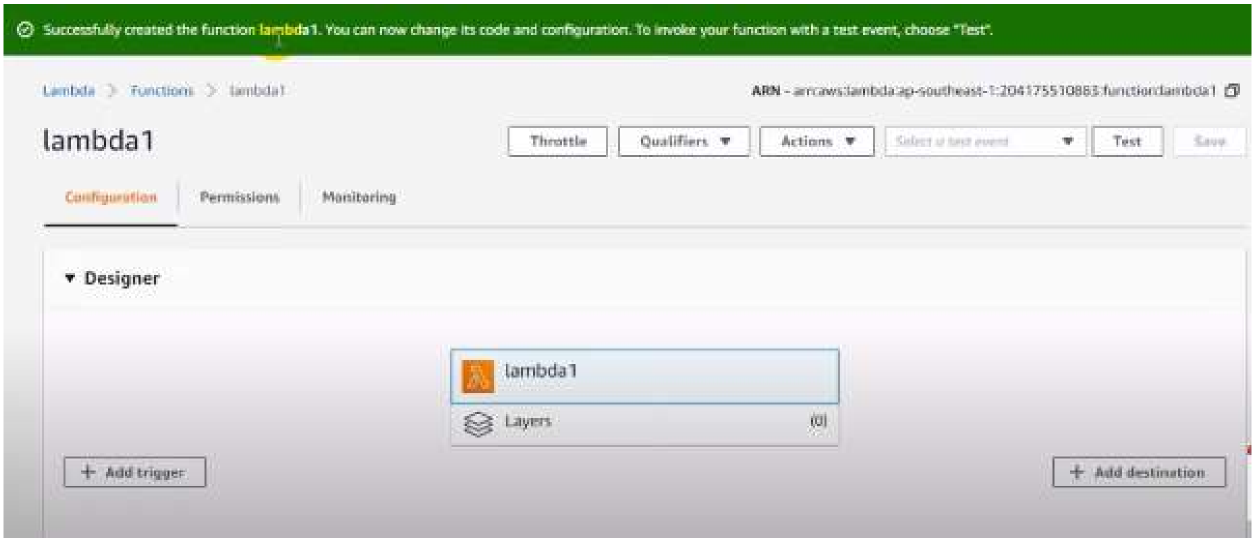
○ Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

lambda-for-dynamodb ▼

View the lambda-for-dynamodb role on the IAM console

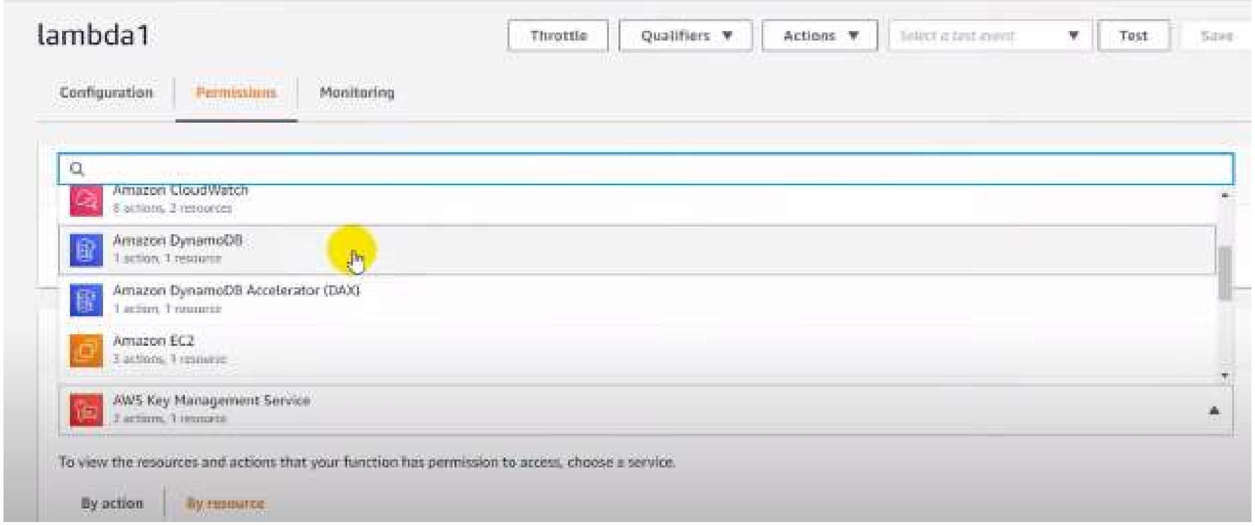


Successfully created the function lambda1. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > lambda1

ARN - arn:aws:lambda:ap-southeast-1:204175510863:function:lambda1

lambda1



lambda1

Amazon CloudWatch
6 actions, 2 resources

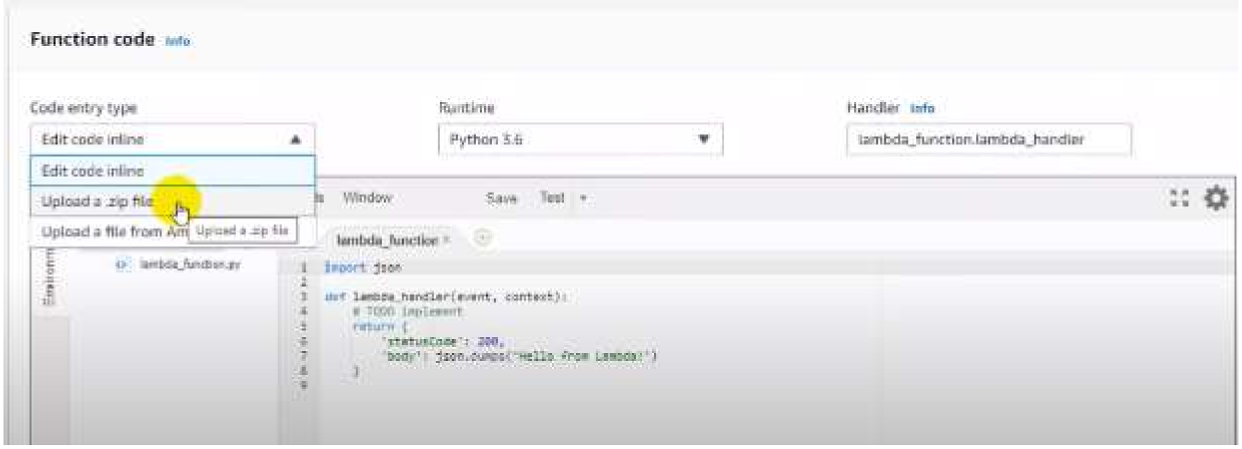
Amazon DynamoDB
1 action, 1 resource

Amazon DynamoDB Accelerator (DAX)
1 action, 1 resource

Amazon EC2
3 actions, 1 resource

AWS Key Management Service
2 actions, 1 resource

To view the resources and actions that your function has permission to access, choose a service.



Function code Info

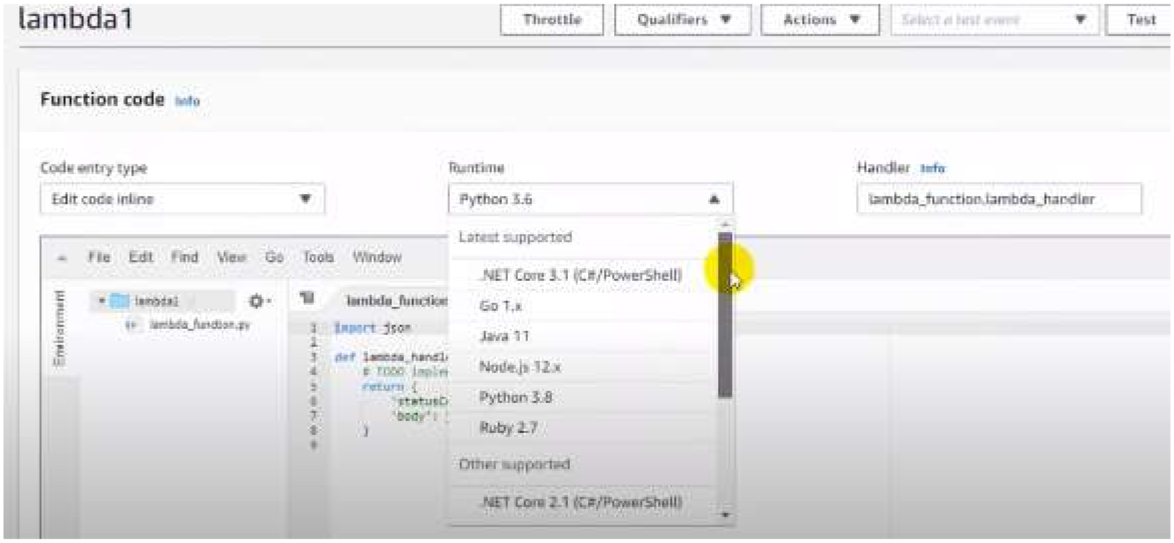
Code entry type: Edit code inline

Runtime: Python 3.6

Handler: lambda_function.lambda_handler

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```



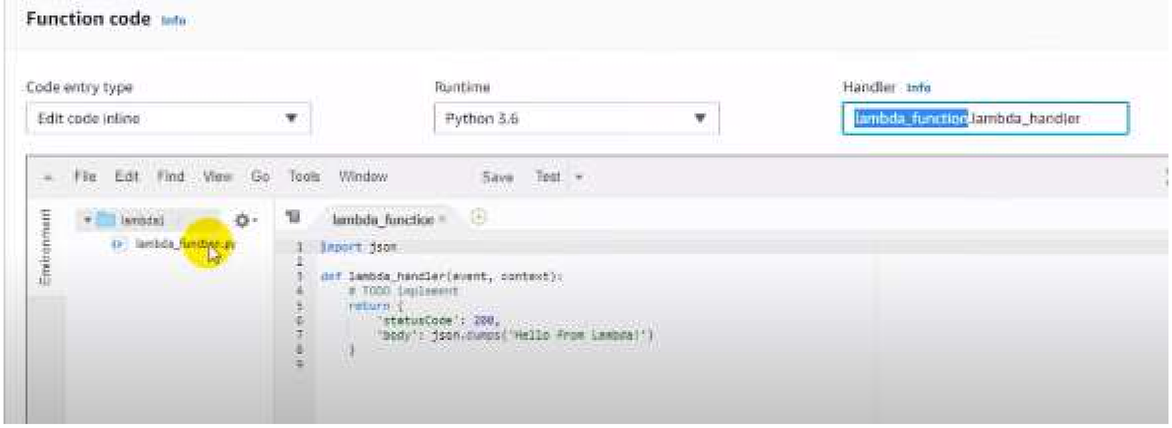
lambda1

Function code Info

Code entry type: Edit code inline

Runtime: Python 3.6

Handler: lambda_function.lambda_handler



Function code Info

Code entry type: Edit code inline

Runtime: Python 3.6

Handler: lambda_function.lambda_handler

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

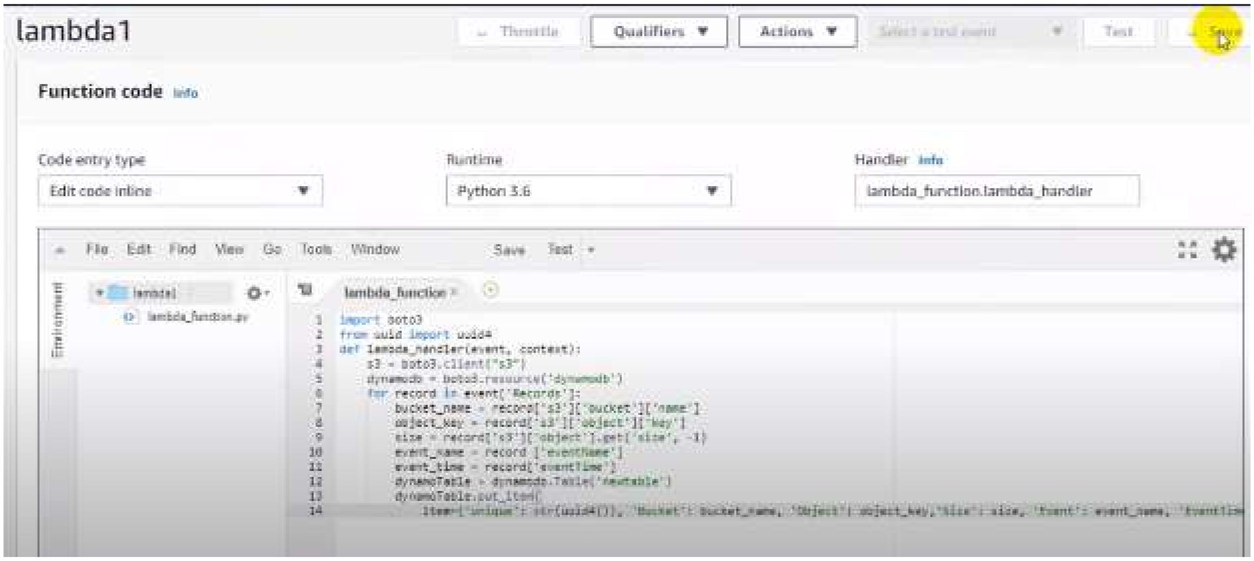
Code used in this Lab**********************

import boto3

from uuid import uuid4

```
def lambda_handler(event, context):
 s3 = boto3.client("s3")
dynamodb = boto3.resource('dynamodb')
for record in event['Records']:
bucket_name = record['s3']['bucket']['name']
object_key = record['s3']['object']['key']
size = record['s3']['object'].get('size', -1)
event_name = record ['eventName']
 event_time = record['eventTime']
dynamoTable = dynamodb.Table('newtable')
dynamoTable.put_item(
Item={'unique': str(uuid4()), 'Bucket': bucket_name, 'Object': object_key,'Size': size, 'Event':
event_name, 'EventTime': event_time}) ************************************
```

/* Only table name (new table) and partition(unique) name should be change*/

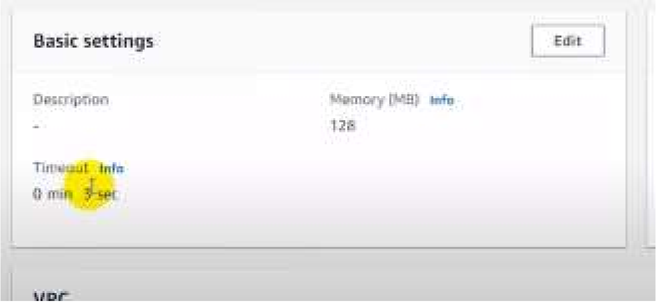


Code entry type: Edit code inline

Runtime: Python 3.6

Handler: lambda_function.lambda_handler

```
import boto3
from uuid import uuid4
def lambda_handler(event, context):
    s3 = boto3.client("s3")
    dynamodb = boto3.resource('dynamodb')
    for record in event['Records']:
        bucket_name = record['s3']['bucket']['name']
        object_key = record['s3']['object']['key']
        size = record['s3']['object'].get('size', -1)
        event_name = record ['eventName']
        event_time = record['eventTime']
        dynamoTable = dynamodb.Table('newtable')
        dynamoTable.put_item(
            Item={'unique': str(uuid4()), 'Bucket': bucket_name, 'Object': object_key,'Size': size, 'Event': event_name, 'EventTim
```

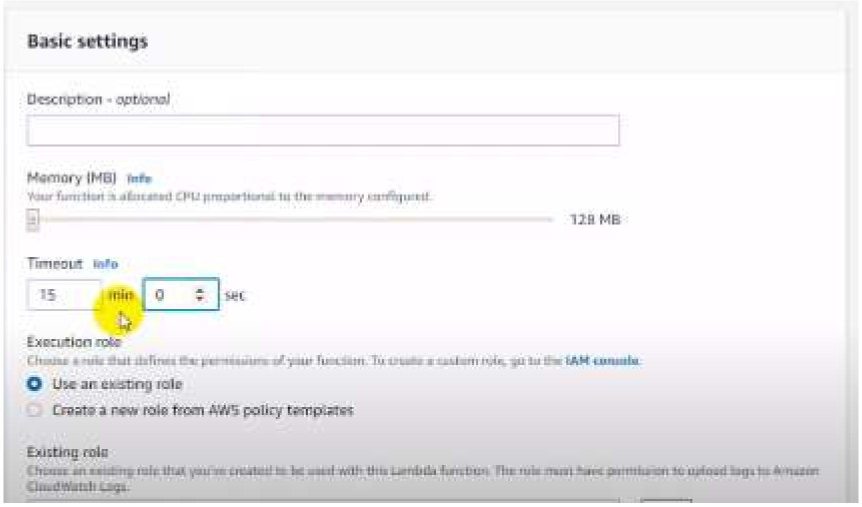


Description: -

Memory (MB): 128

Timeout: 0 min 3 sec

default 3 sec and max 15mins



Basic settings

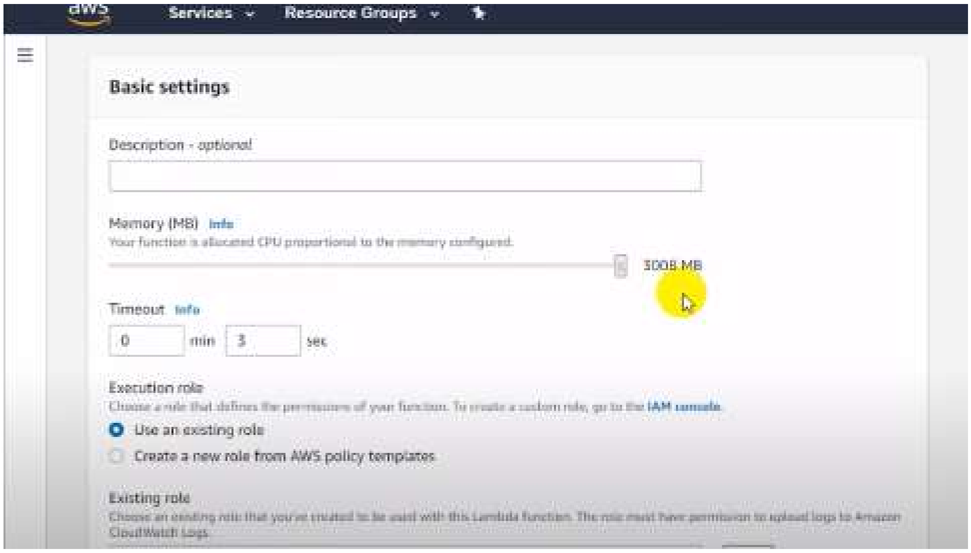
Description - *optional*

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

Timeout Info
15 min 0 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
● Use an existing role
○ Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.



Basic settings

Description - *optional*

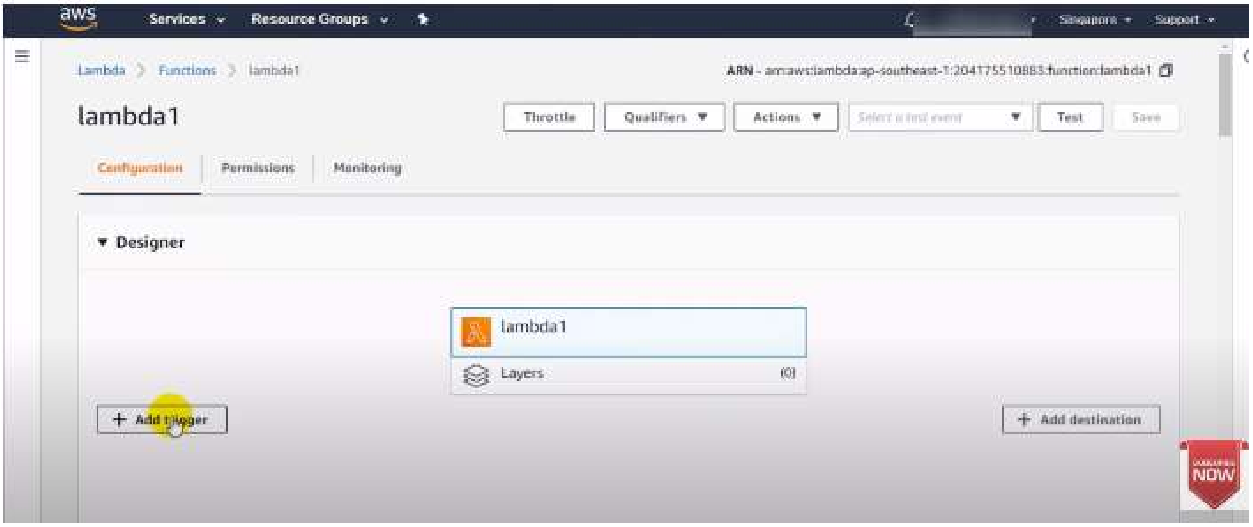
Memory (MB) Info
Your function is allocated CPU proportional to the memory configured.
3008 MB

Timeout Info
0 min 3 sec

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
● Use an existing role
○ Create a new role from AWS policy templates

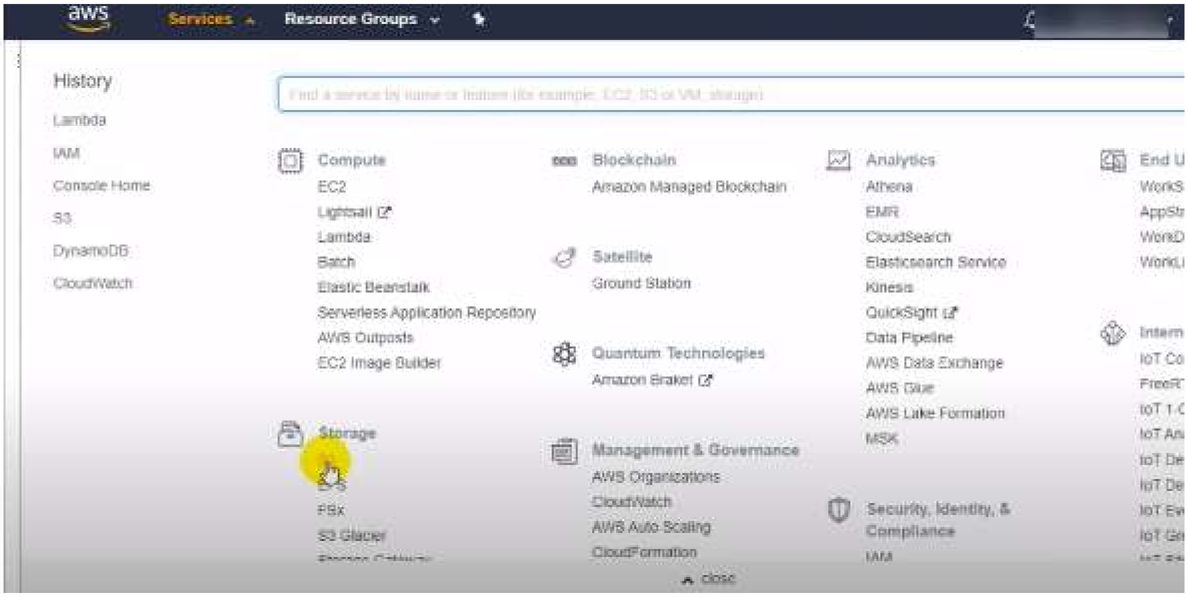
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

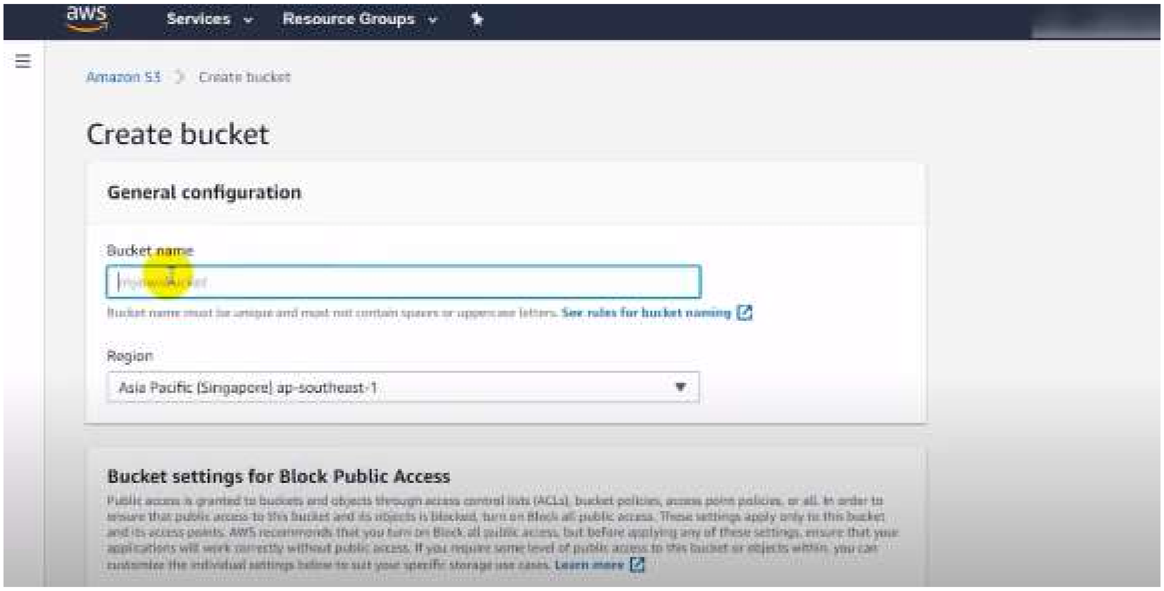
Memory min -128 max -3008



ARN - arn:aws:lambda:ap-southeast-1:204175510883:function:lambda1

lambda1





Create bucket

General configuration

Bucket name

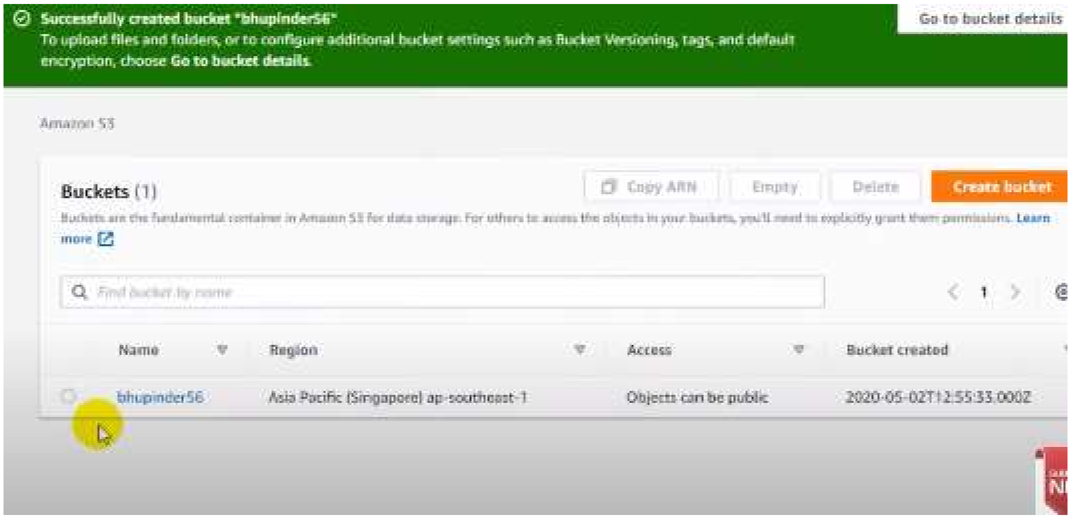
Bucket name must be unique and must not contain spaces or uppercase letters. See rules for bucket naming

Region

Asia Pacific (Singapore) ap-southeast-1

Bucket settings for Block Public Access

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

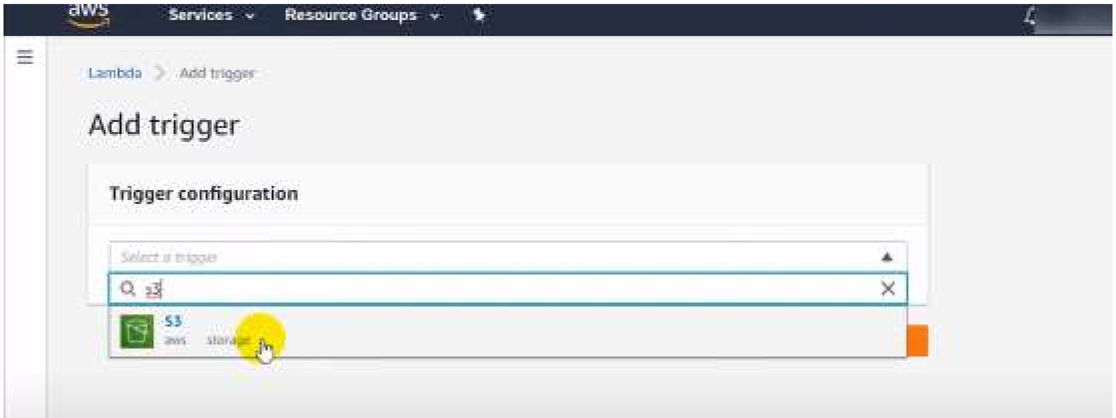


Successfully created bucket "bhupinder56"
To upload files and folders, or to configure additional bucket settings such as Bucket Versioning, tags, and default encryption, choose **Go to bucket details**.

Buckets (1)

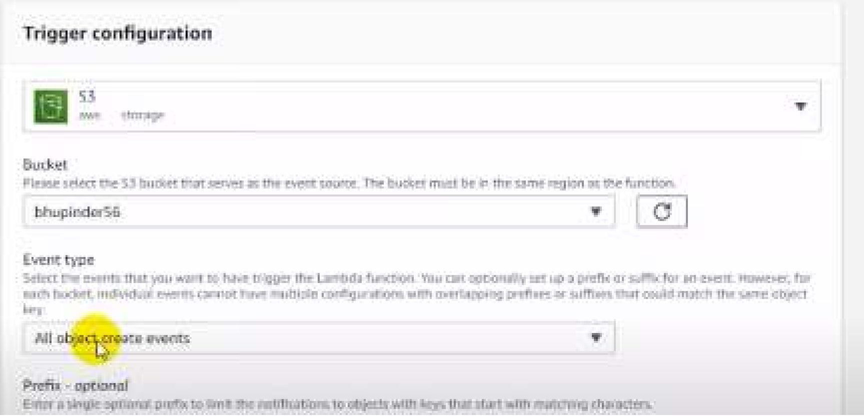
Buckets are the fundamental container in Amazon S3 for data storage. For others to access the objects in your buckets, you'll need to explicitly grant them permissions. Learn more

	Name	Region	Access	Bucket created
○	bhupinder56	Asia Pacific (Singapore) ap-southeast-1	Objects can be public	2020-05-02T12:55:33.000Z



Add trigger

Trigger configuration



Trigger configuration

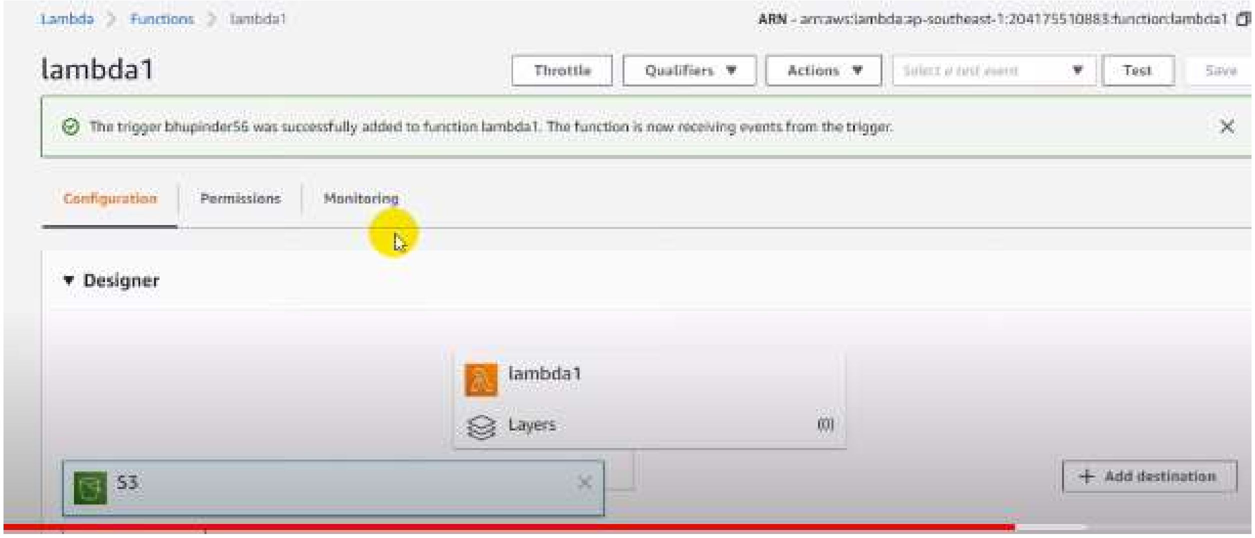
Bucket
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

bhupinder56

Event type
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

All object create events

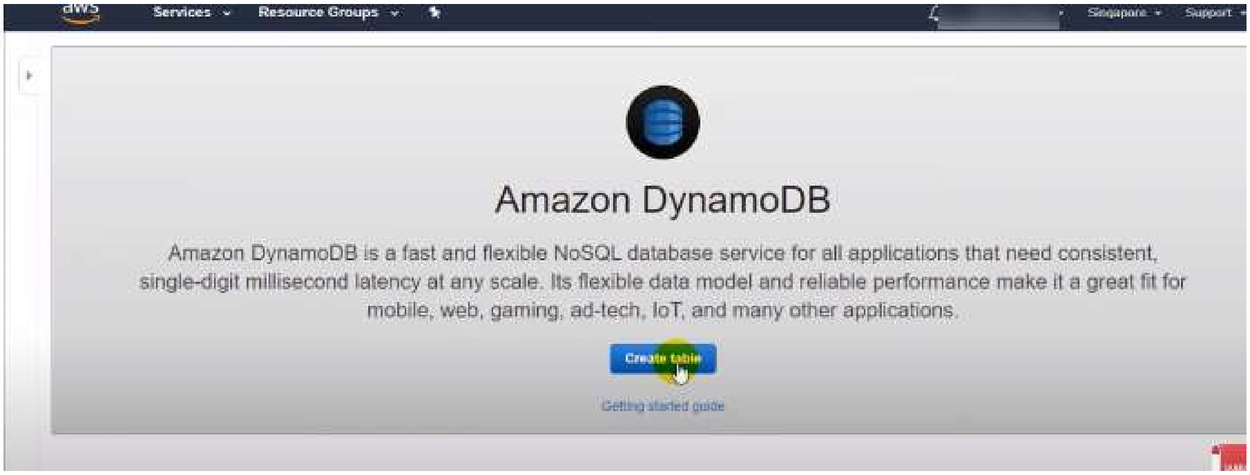
Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.



ARN - arn:aws:lambda:ap-southeast-1:204175510883:function:lambda1

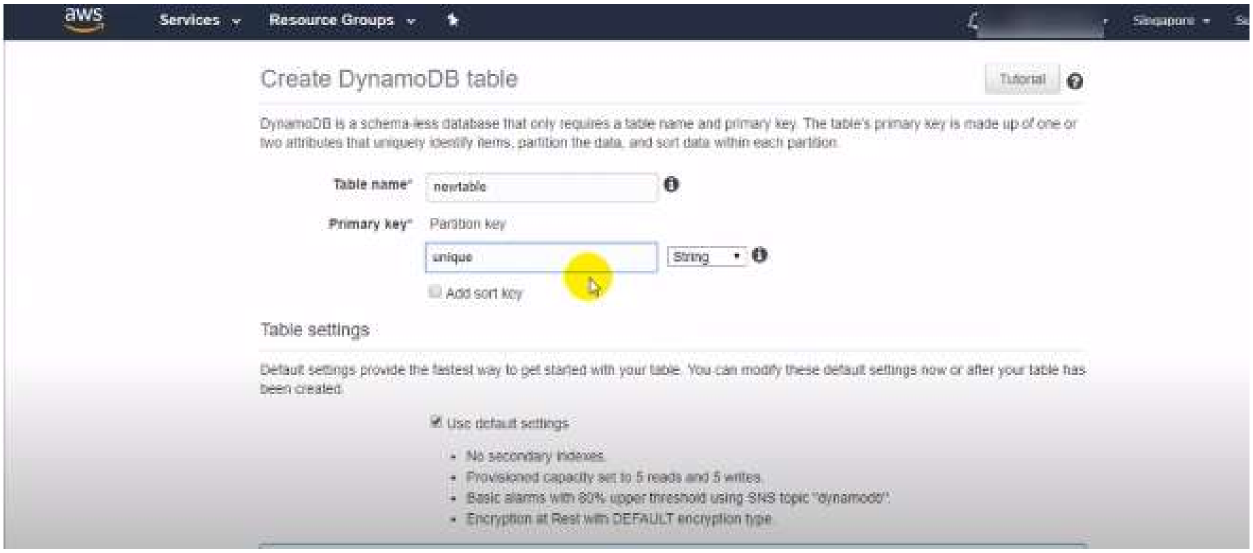
lambda1

The trigger bhupinder56 was successfully added to function lambda1. The function is now receiving events from the trigger.



Amazon DynamoDB

Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, IoT, and many other applications.



Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* newtable

Primary key* Partition key

unique String

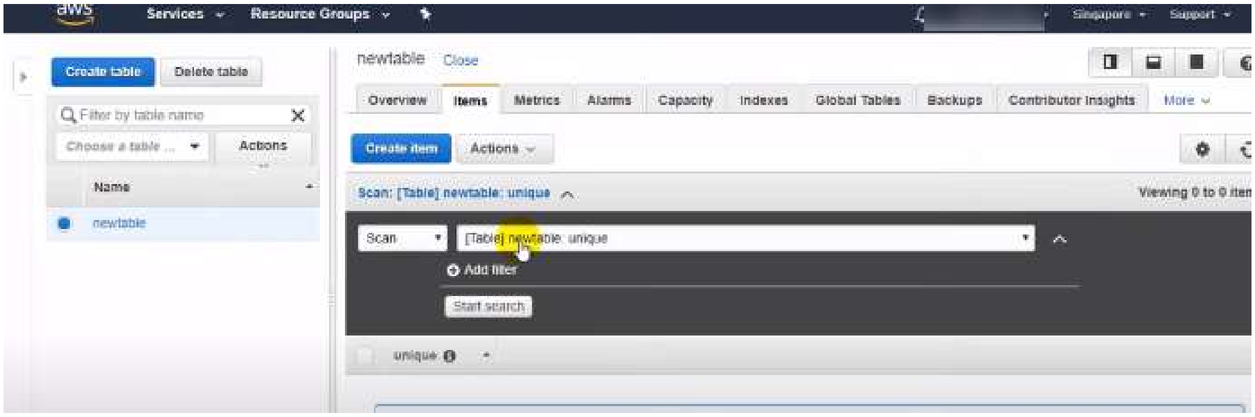
□ Add sort key

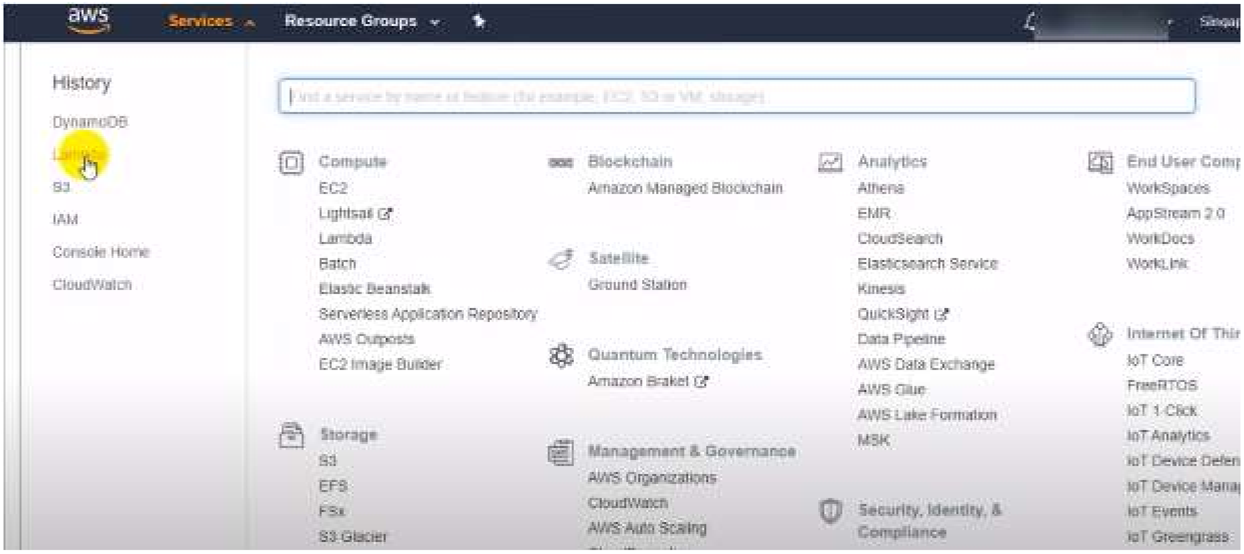
Table settings

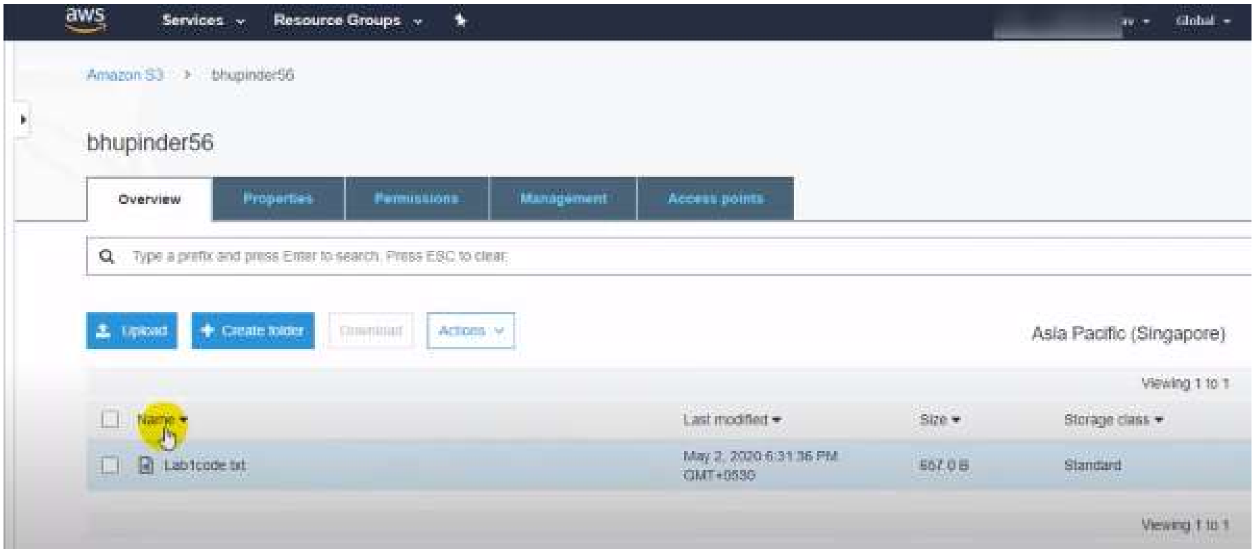
Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

☑ Use default settings

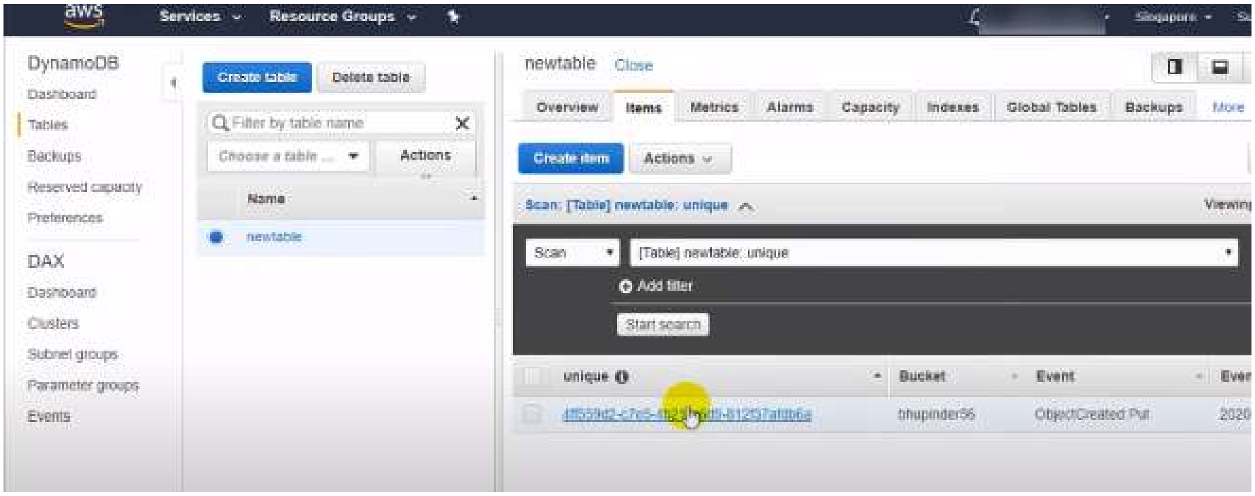
- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
- Encryption at Rest with DEFAULT encryption type.



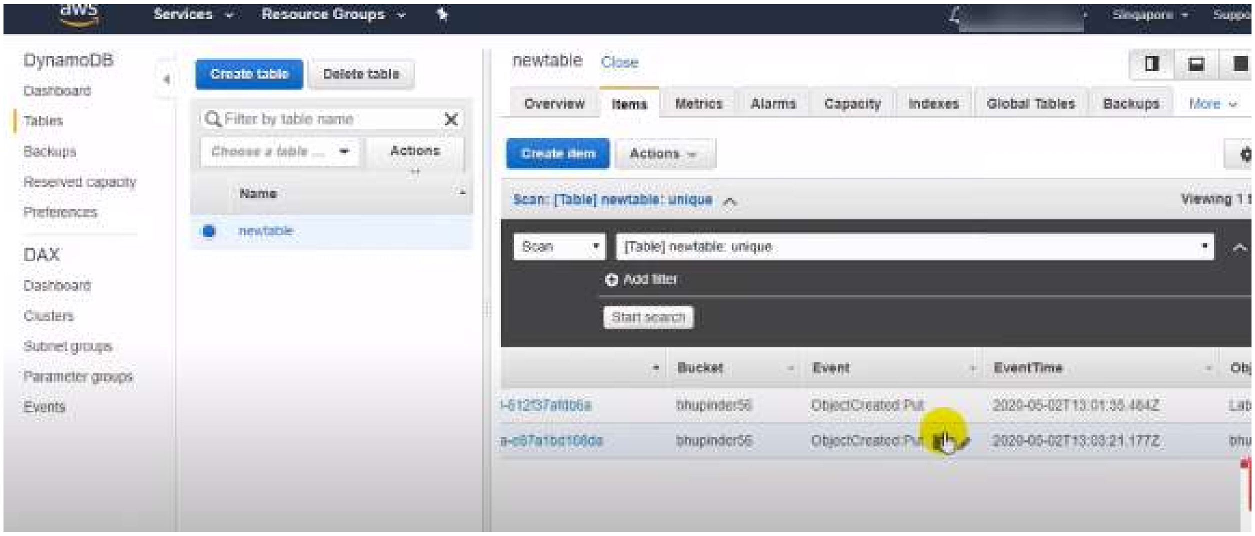




□	Name	Last modified	Size	Storage class
□	Lab1code.txt	May 2, 2020 6:31:36 PM GMT+0530	667.0 B	Standard



	unique	Bucket	Event	Ever
□	4ff559d2-c7e5-4[illegible]-81257afdb6a	bhupinder56	ObjectCreated:Put	2020



	Bucket	Event	EventTime	Obj
l-612f37afdb6a	bhupinder56	ObjectCreated:Put	2020-05-02T13:01:35.484Z	Lab
a-e67a1bd108da	bhupinder56	ObjectCreated:Put	2020-05-02T13:03:21.177Z	bhu