1**. AWS Lambda**

2. **AWS API Gateway**

3. **AWS KMS (Key Management Service)**

4. AWS SQS, Amazon MQ

5. **AWS Secrets Manager**

6. **AWS Web Application Firewall (WAF)**

7. AWS Systems Manager Parameter Store

8. AWS Cost Optimization

1.AWS Lambda

Theory

https://www .youtube.com/watch?v=5fTtmeCpSRw&list=PLdpzxOOAlwvLNOxX0RfndiYSt1Le9azze&index=19

Lambda function task

<https://www.youtube.com/watch?v=OKYJCHHSWb4>

2. AWS API Gateway

<https://www.youtube.com/watch?v=jcibXVFiFek>

3. AWS KMS (Key Management Service)

<https://www.youtube.com/watch?v=f3APF1dP8w0>

<https://www.youtube.com/watch?v=SmrvRYg-ETQ>

<https://www.youtube.com/watch?v=zSUUBAxjIbk>

4. AWS SQS, Amazon MQ

<https://www.youtube.com/watch?v=CyYZ3adwboc&t=15s>

<https://www.youtube.com/watch?v=MH01PNZLR98&t=373s>

<https://www.youtube.com/watch?v=1WGtEGzQOyo>

5. AWS Secrets Manager

<https://www.youtube.com/watch?v=FllcHYsBm78>

<https://www.youtube.com/watch?v=Y3Gn_iP3FlE>

<https://www.youtube.com/watch?v=DjvlhrBRtXM>

6. AWS Web Application Firewall (WAF)

<https://www.youtube.com/watch?v=FHRXXrQ765M>

7. AWS Systems Manager Parameter Store

<https://www.youtube.com/watch?v=zxi3M_rIE2k>

8. AWS Cost Optimization

LAMBDA:

As of now we learned to run any application or website we were using server either it may be windows server or EC2 server. Means we were using virtual server/machine that we call EC2 instance. And this EC2 instance we find in compute. So in compute we have lots of options in that one option is LAMBDA.

Difference between LAMBDA an EC2 instance:

If you want to run any your application or website for we needed one server, and on that server we run that website. Because without server website can not run. But with the help of cloud powerful features now you don’t need any server to run you application or code.

Suppose I you have a such option just put your code/application here and this application run automatically whenever it’s need that option called LAMBDA.

LAMBDA: LAMBDA can defined as a server less computing. Means without server also LAMBDA can run/perform your application or your any task with the help of LAMBDA function/option. Just we need to select this instead of any server. And just pay the bill for that.

One most important thing is, If you take server the charge of this server occur based on per second, per minute or per hour and that we need to pay if our server is running even though your application is using 1 hour only in a day. And we were also very happy that we don’t need put hardware in real-time.

In another side LAMBDA has flexibility that you get the charges when application run there for that particular time only. And no need to give any server charges. So this cost effective benefit of LAMBDA.

. AWS lambda is a compute service that lets your own code/application without provisionong or managing servers. (In LAMBDA you can run your application without purchasing any server).

.With AWS LAMBDA, you can run code for virtually any type of application or backend service – all with zero Administartion. Means we don’t bother how LAMBDA is working, who is doing lambda maintenance, who is doing update etc., this all AWS will take care. Just we put code in lambda.

. **AWS lambda manages all the administration. What it manages**

1. Provisioning and capacity of the compute fleet that offers a balance of memory, CPU, network, and other resources. We don’t get any option to run application, we select this instance or any particular instance. this AWS only decide because at backend we don’t know AWS running which instance this all are hidden. (see server less computing is for us only but at backend somewhere it is running on hardware or instances)
2. Server and OS maintenance (AWS only manage we don’t need to bother on which server is running or OS is running at backend.
3. High availability and Auto Scaling (Means you lambda function fails or for any reason it’s not working then AWS provide high availability means somewhere that application run.
4. Monitoring fleet health ( means you have one lambda function or many lambda functions you created through Auto Scaling, their all health check monitoring done by AWS.
5. Applying security patches (If any security issue happens we don’t need to bother or keep firewall or SG, Everything take care by AWS)
6. Deploying your code for your application. (just we need to give code only, AWS only responsible to run this code).
7. Monitoring or logging your lambda functions
8. All you need to do is supply your code in the form of one or more lambda functions (code) to AWS lambda, in one of the languages that AWS lambda supports (At presents it supports 7 languages. Node JS. JAVA. POWERSHELL, C# sharp, Ruby, Python, Go language of google) and the service can run the code on your behalf.
9. Typically the lifestyle for an AWS LAMBDA based application includes, Authoring code, Deploying code to AWS lambda and then monitoring and troubleshooting
10. This is in exchange for flexibility, which means you can’t log into compute instances or customize the OS or language runtime.
11. If you do want to manage your own compute, you can use EC2 or Elastic Beanstalk

**How Lambda works?**

1. First you upload your code to lambda on one or more lambda function.
2. AWS lambda will then execute the code in your behalf.
3. After the code is invoked, lambda automatically take care of provisioning or managing the required servers.

**Difference between AWS LAMBDA and AWS EC2**

### **AWS Lambda**

* AWS Lambda is Platform-as-a-Service. (Means code runs on platform only like python language ect., on which we develop the code. But you don’t bother about infrastructure which is handled by AWS lambda at backend.
* 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 customized OS or language platform.

### **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 OS and install all the software required and then push your code in EC2.
* You can select variety of OS, instance types, network & security patterns, RAM, & CPU etc.

### **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 requests 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 document 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.

**6) Concurrency** – Number of requests that your function is serving in any given time.

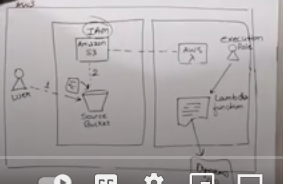
**When lambda triggers**

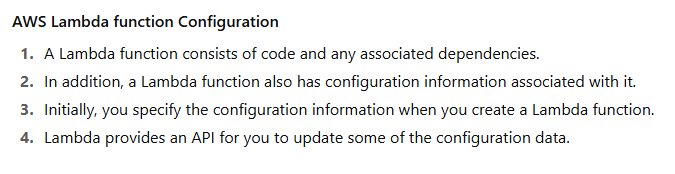
**When Lambda Triggers**

* 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 an HTTP request using Amazon API Gateway
* With these capabilities, you can use Lambda to easily build data processing triggers for AWS devices 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 creates 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





Even if you created a lambda function later through EPI you can edit or change configuration of your lambda.

**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 assumes when it executes the Lambda function on your behalf.

**AWS Lambda function – Services it can access**

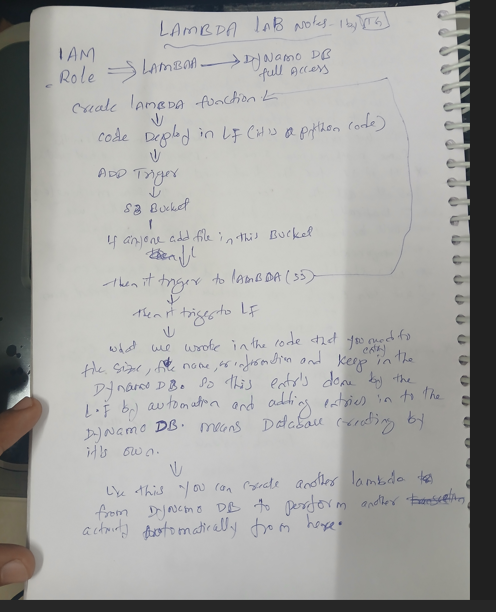
→ Lambda functions can access –

* AWS services or non-AWS services
* AWS services running in AWS VPC (e.g., Redshift, ElastiCache, RDS instance)
* Non-AWS services running on EC2 instances in an AWS VPC

AWS Lambda runs 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.

Lab:

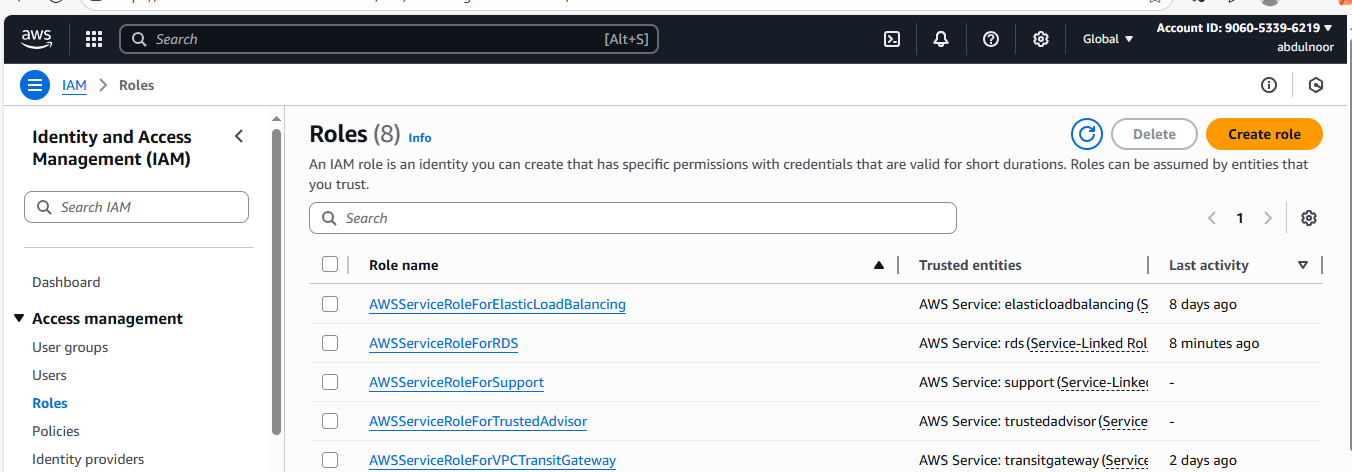


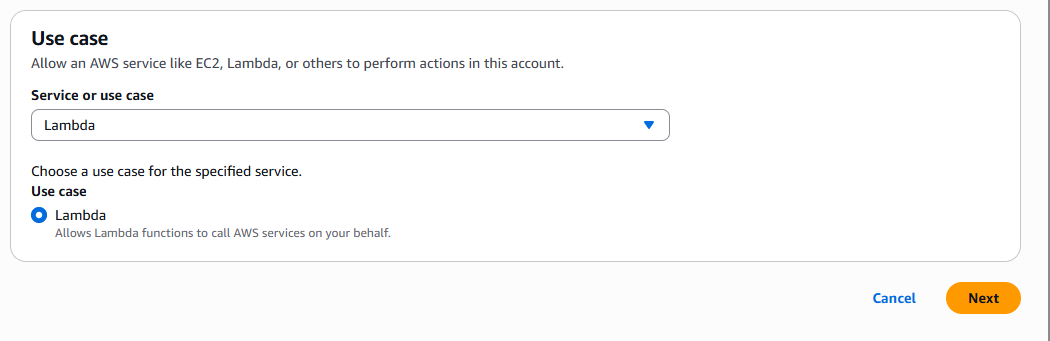
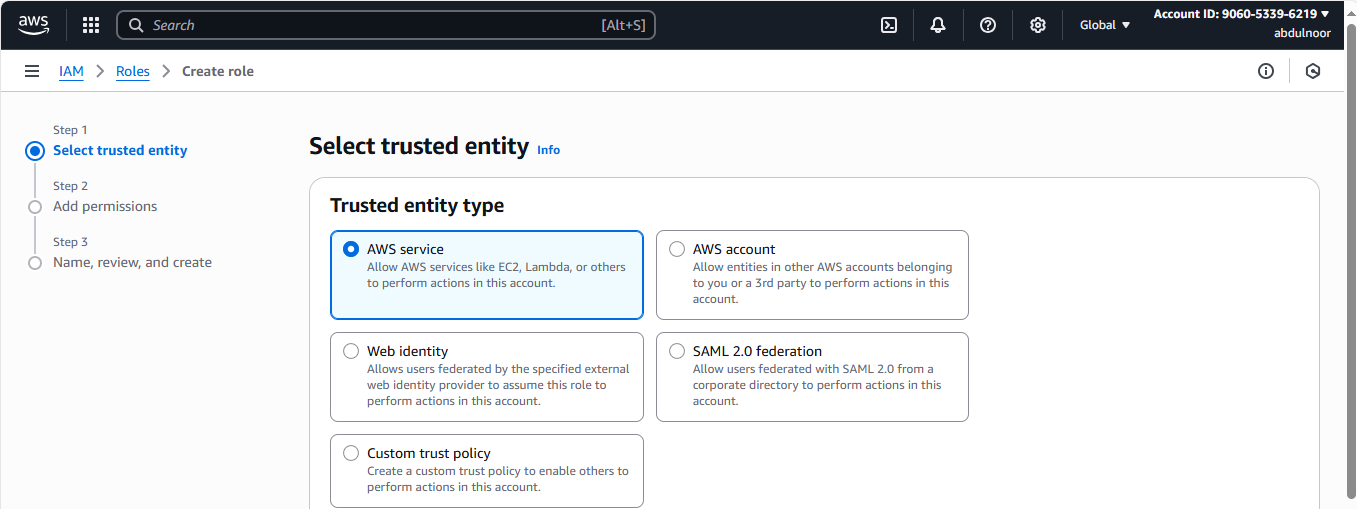
Go to AWS Console:

And click on 3 lines left side – click on all services – under compute – click on LAMBDA.

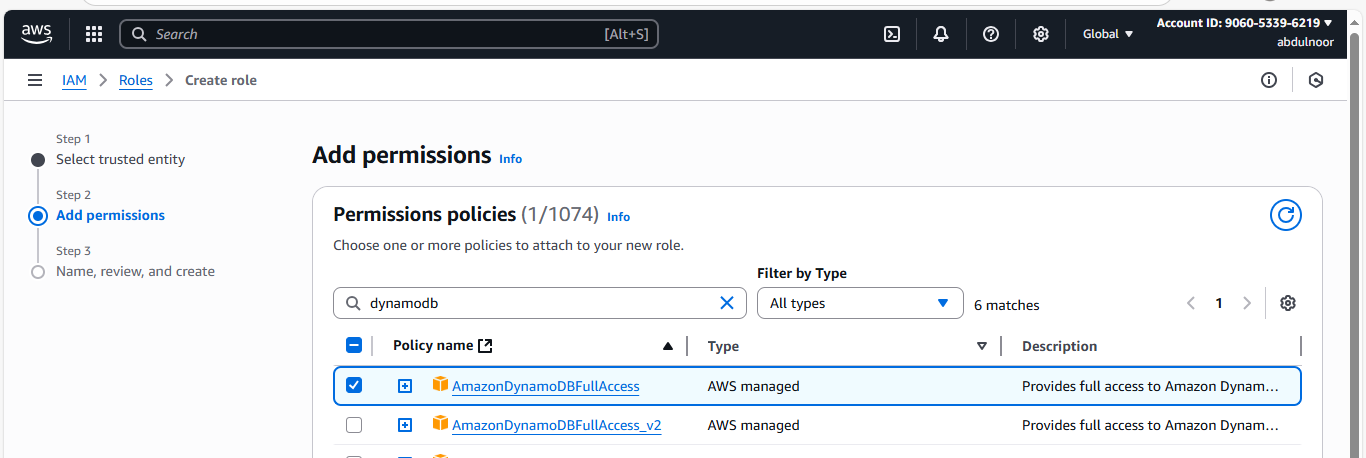
Go to IAM

Click on ROLES let side – click on Create Role

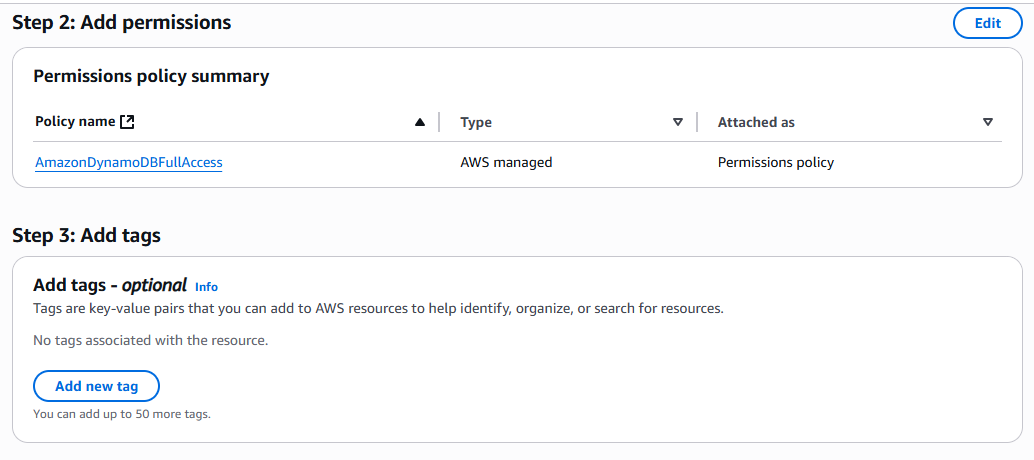
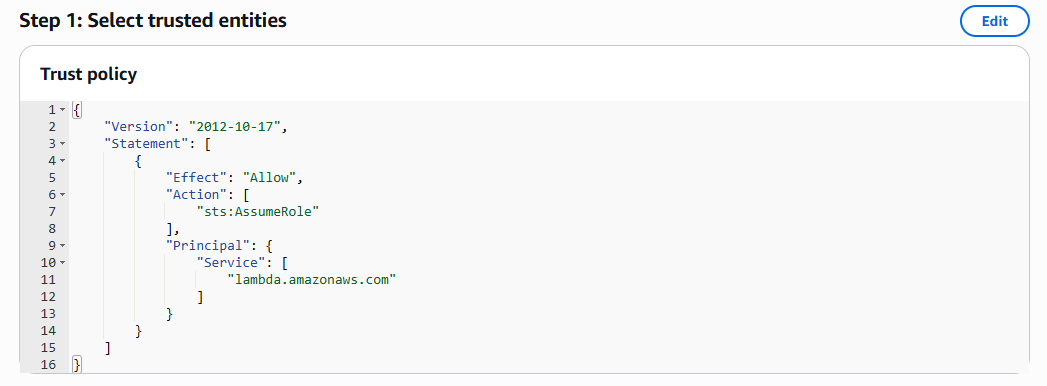
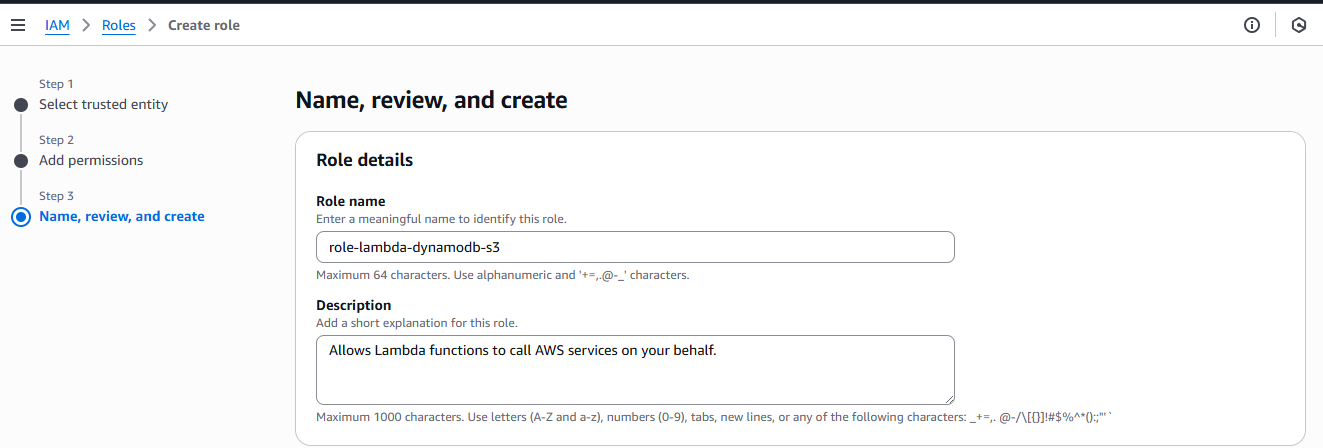




Click on Next



Click on Next

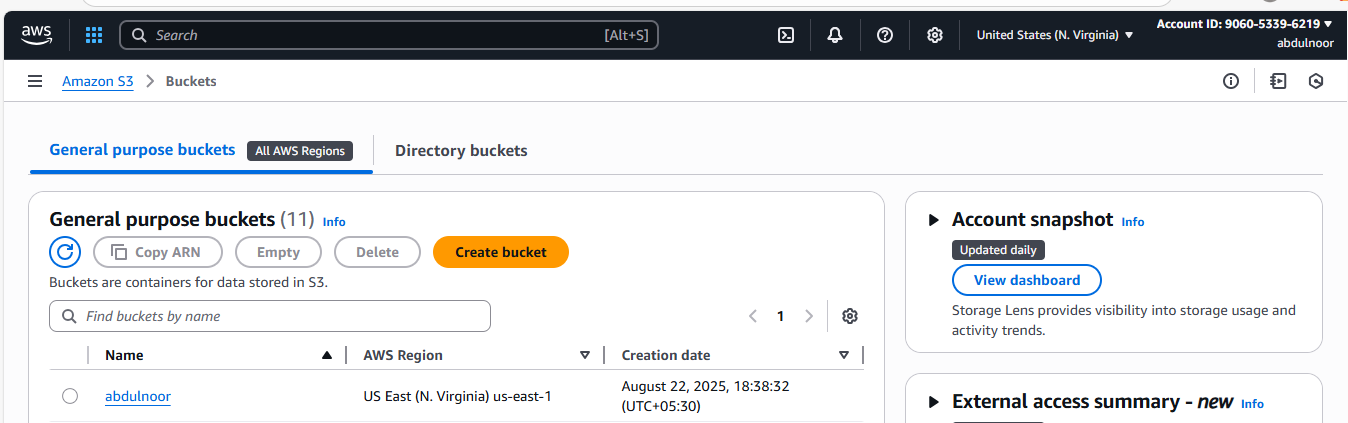


No need to do anything in the above step 1 and 2. This automatically AWS generate script for role .

Click on create role.

Now Role has ready.

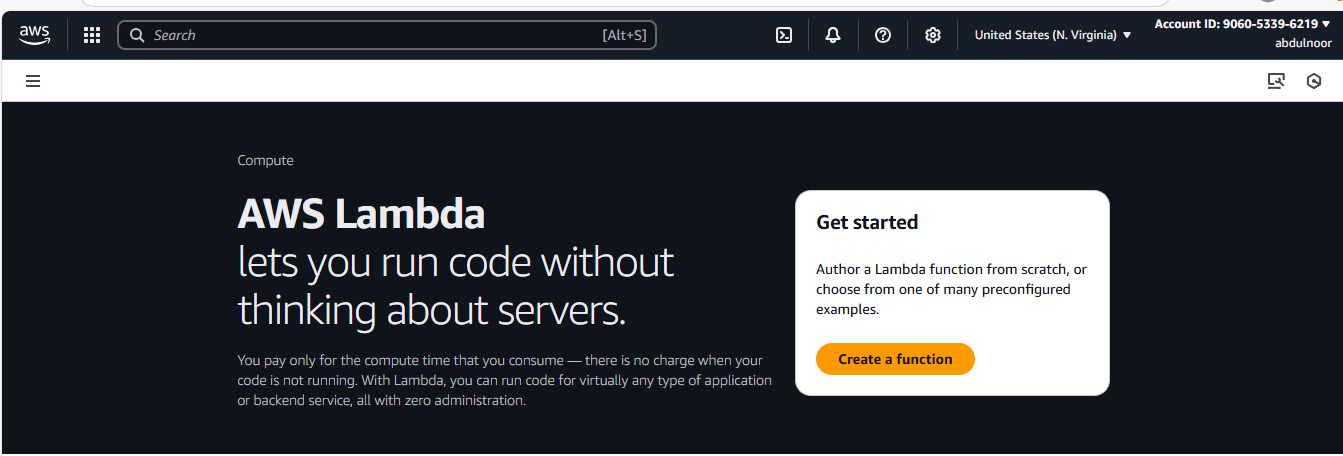
2) create a S3 bucket



S3 bucket can create with any name because code doesn’t any work with it’s name.

Because we attach to S3 bucket in trigger and there we select the bucket.

3) click on LAMBDA under compute



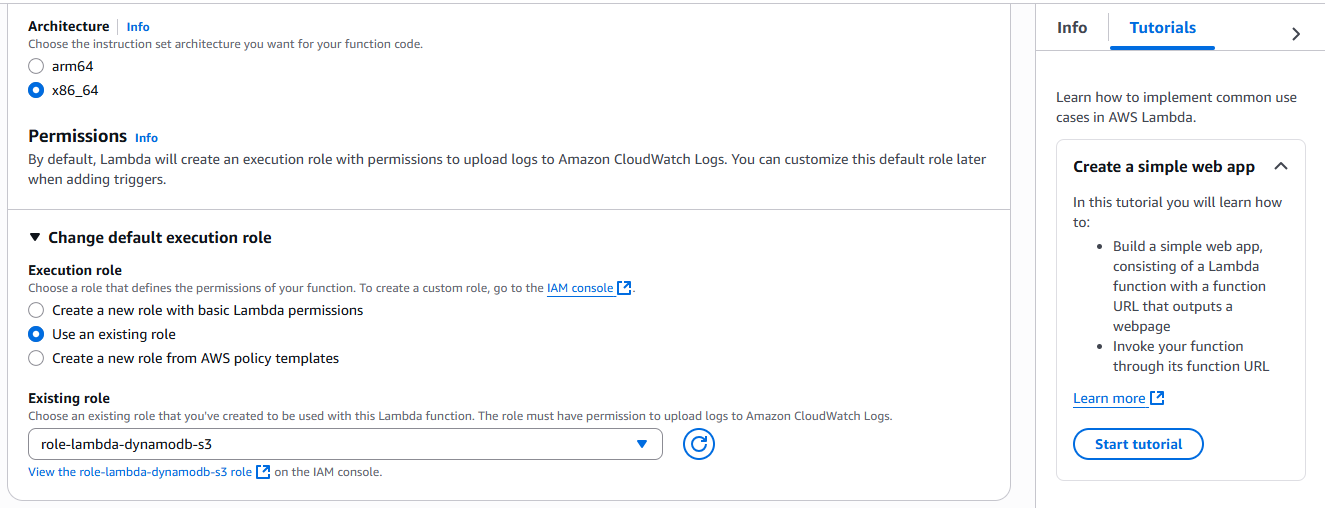
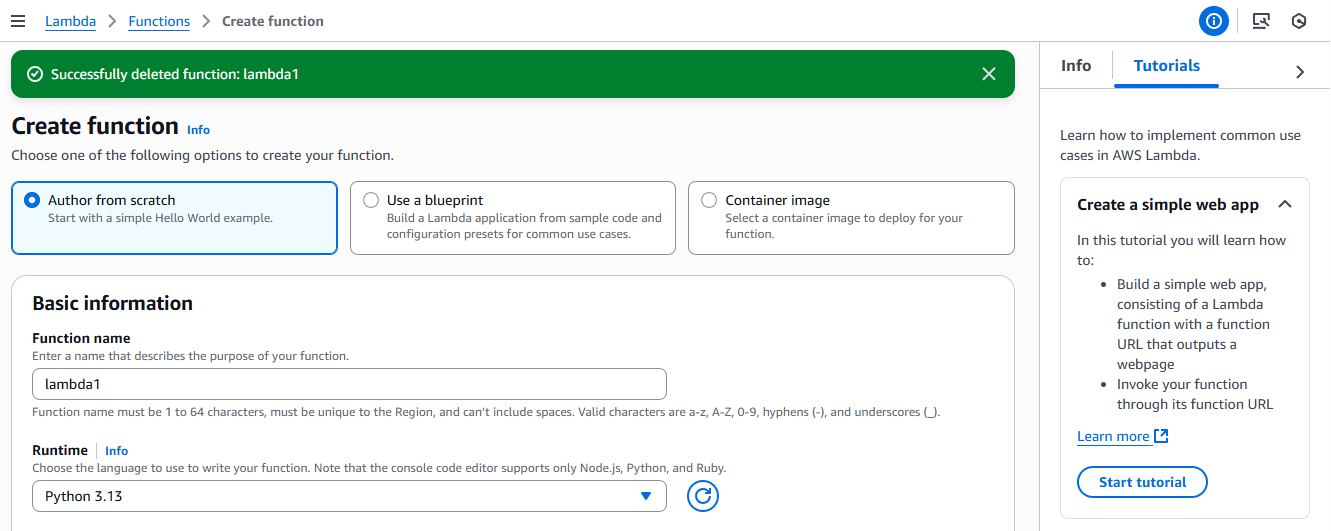
Click on create a function

See here we don’t have earlier any function

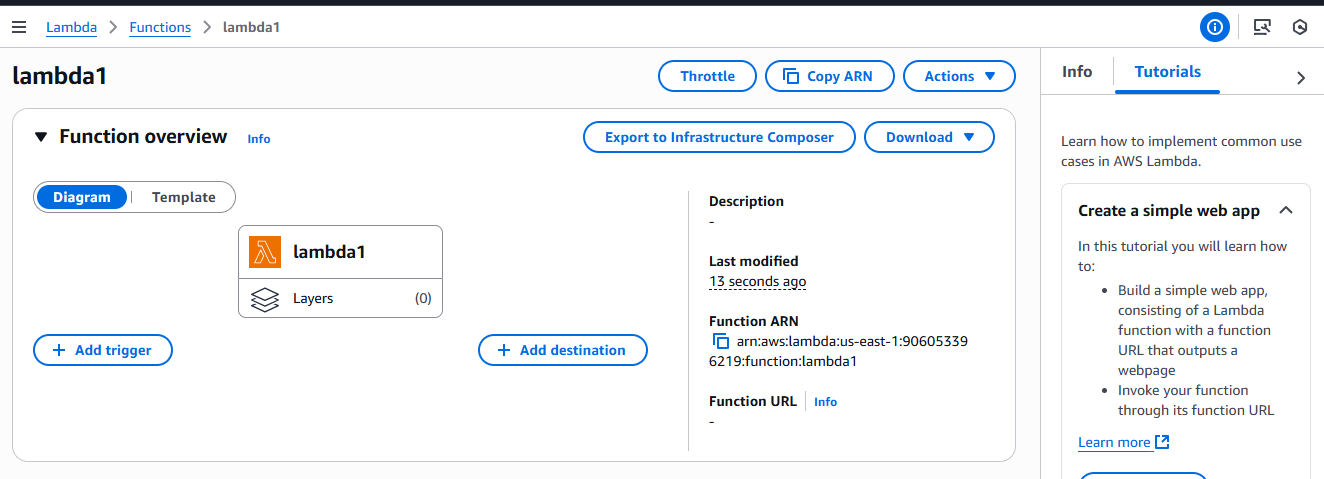
Points:

Author From Scratch : to put/attach our code we use this option.

Runtime: In which programing language you want to run your code (like python)



Click on create

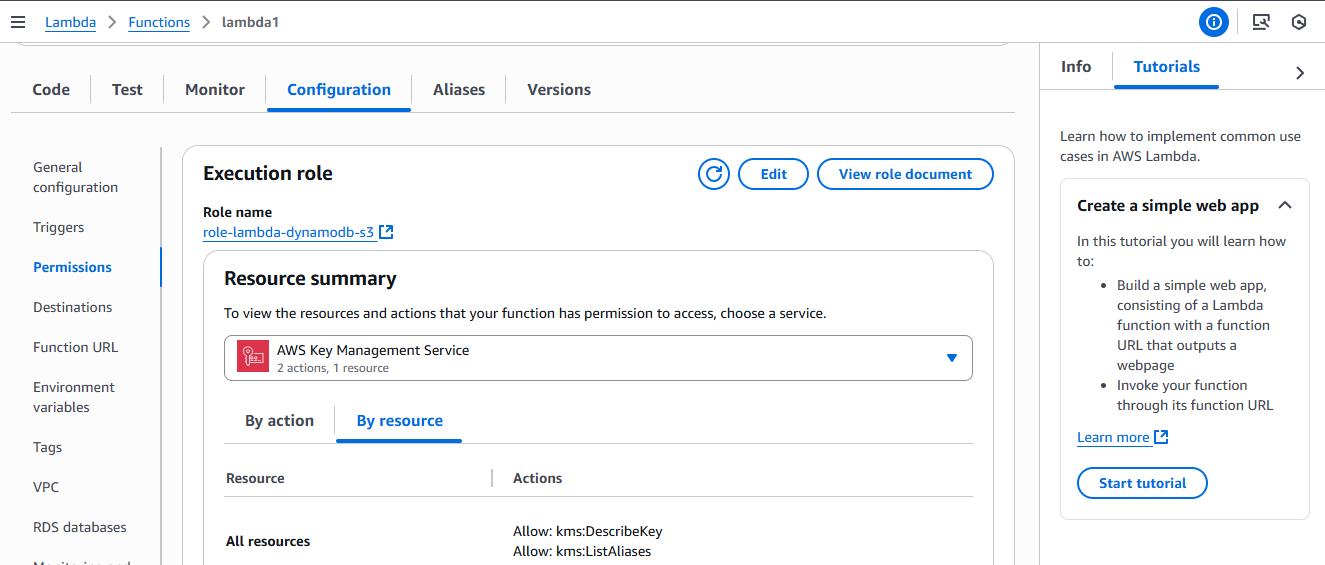


Now I have created Lambda function, name as lambda1

In let side I need add trigger (S3 bucket)

And you can see your permissions which added in the IAM role that is Dynamodb in the permission tab let side because we are using first time this function.

Go to configuration tab – click on Permission tab let side here you can your permissions in the resource summary.



Click on code tab and remove the existing code and paste your code according to the python version. Code starts as below

import boto3

from uuid import uuid4

from typing import Any, Dict

def lambda\_handler(event: Dict[str, Any], context: Any) -> Dict[str, Any]:

s3 = boto3.client("s3")

dynamodb = boto3.resource("dynamodb")

dynamo\_table = dynamodb.Table("newtable")

for record in event.get("Records", []):

bucket\_name = record["s3"]["bucket"]["name"]

object\_key = record["s3"]["object"]["key"]

size = record["s3"]["object"].get("size", -1)

event\_name = record.get("eventName")

event\_time = record.get("eventTime")

# Insert item into DynamoDB

dynamo\_table.put\_item(

Item={

"unique": str(uuid4()),

"Bucket": bucket\_name,

"Object": object\_key,

"Size": size,

"Event": event\_name,

"EventTime": event\_time

}

)

return {

"statusCode": 200,

"body": "Data inserted successfully into DynamoDB"

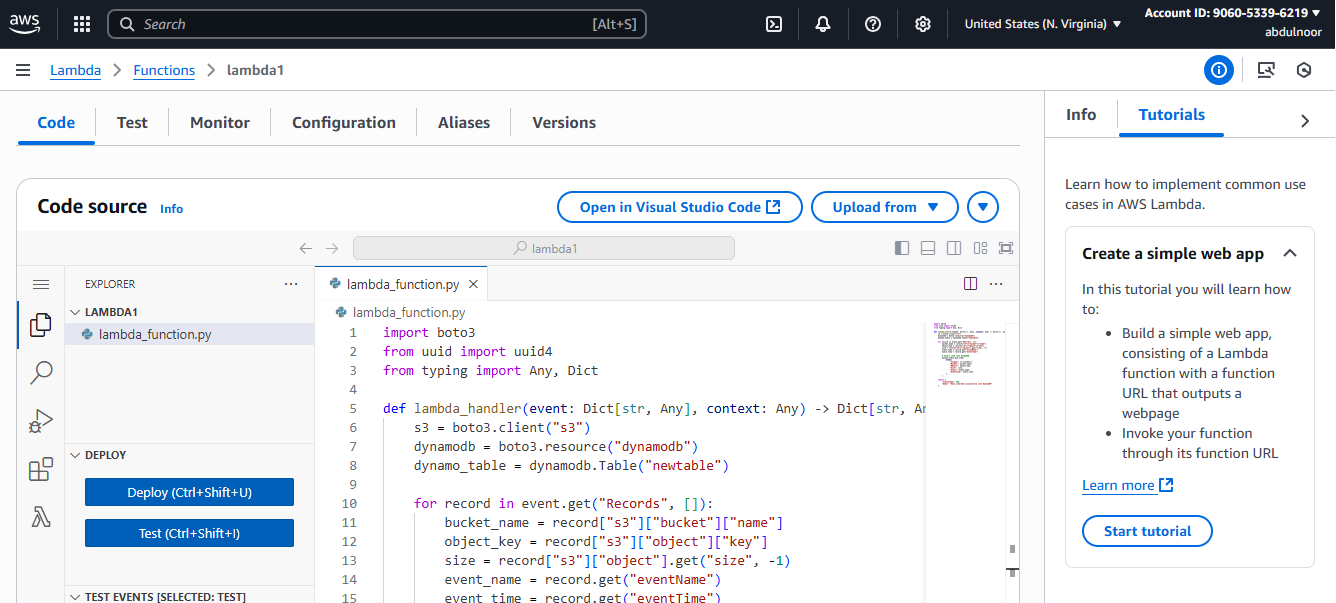
}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

here table name is newtable

and the id which keep in that is unique , that is partition key name

you can change according to your table name

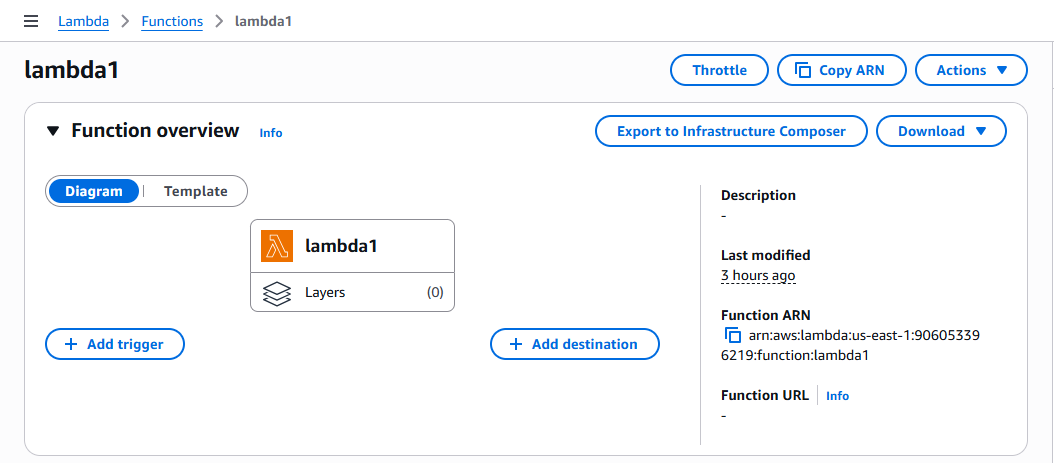


click on DEPLOY (ctrl+shift+u)

if you scroll down there is basic settings in that there is timeout setting which is maximum 15 minutes to run your code.

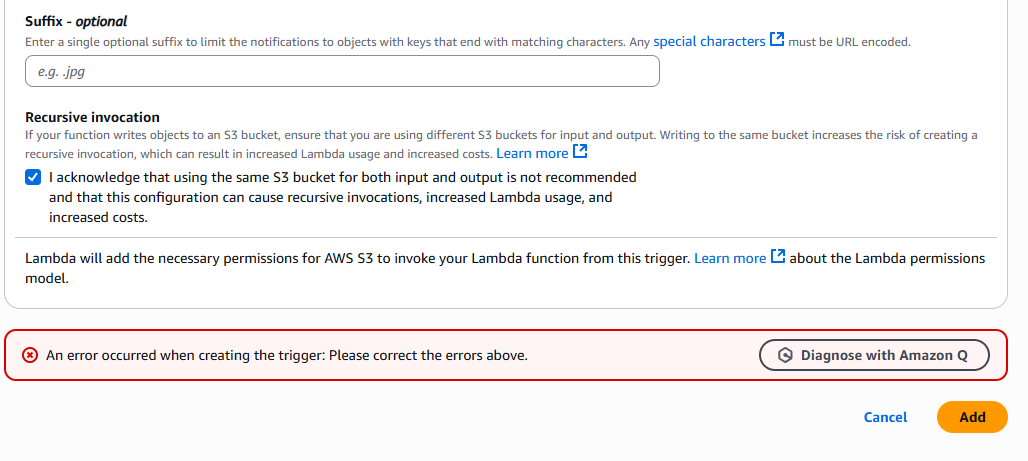
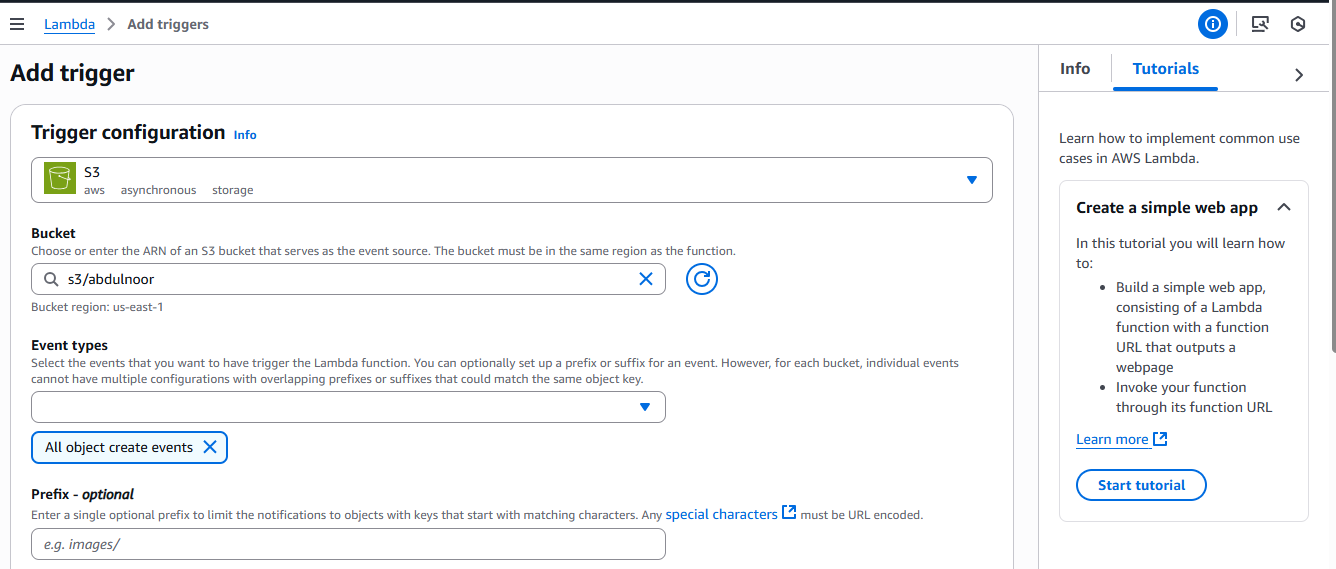
you can check by giving 16 minutes it wont take.

Now I need to add trigger in my lambda function

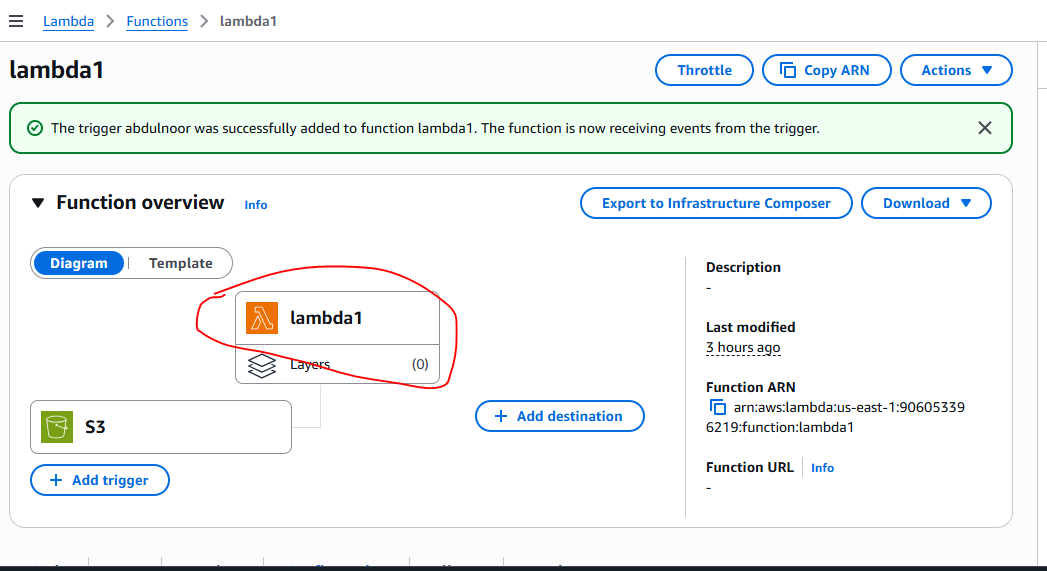


Trigger happened when event/activity occur then s3 bucket need to be trigger then S3 informed to lambda and lambda updates the all the information in dynamodb.

Ex: any data come into this bucket or Ex: object created



Click on add

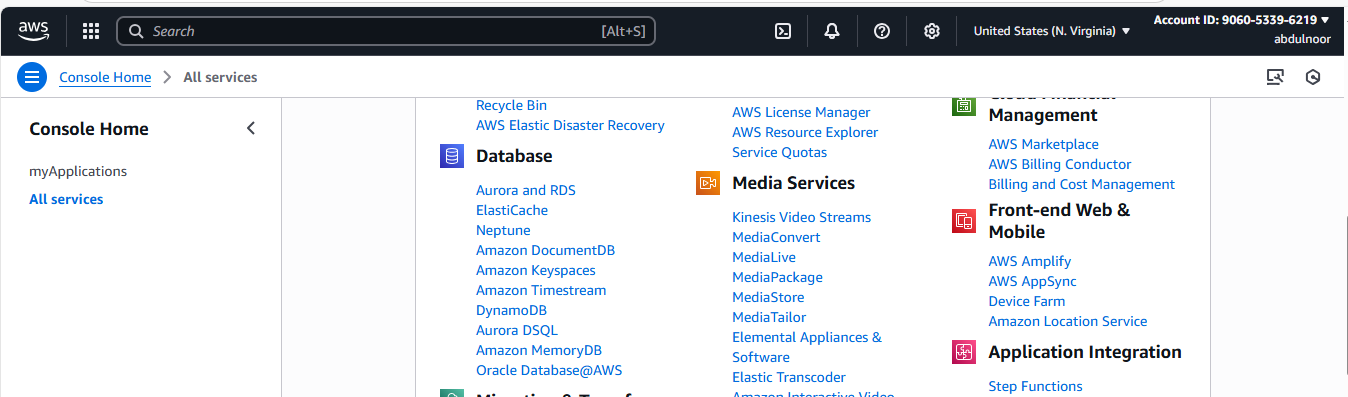


Now Lambda need to inform to Dynamodb that objects came in S3 bucket that information need to put in DDB according to my written code.

Click on lamda1 function so that below you can see here itself your written code.

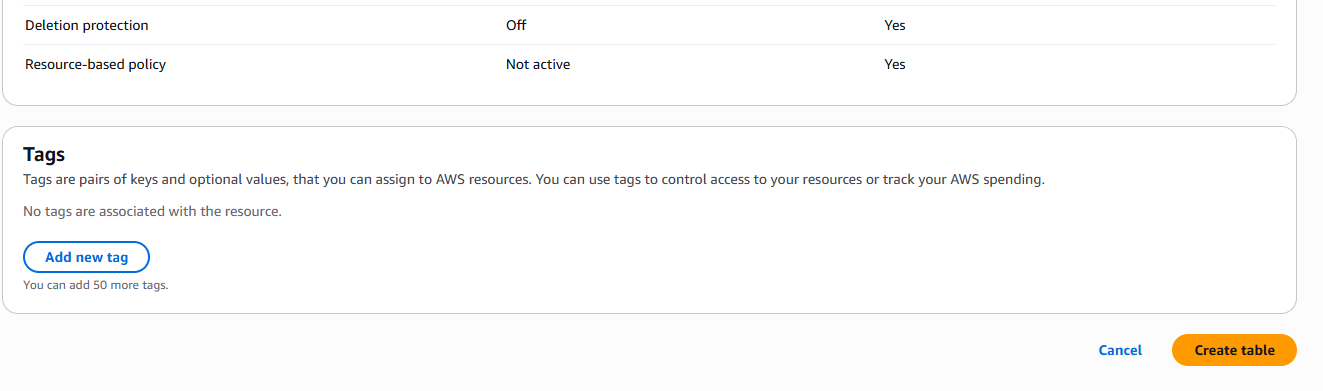
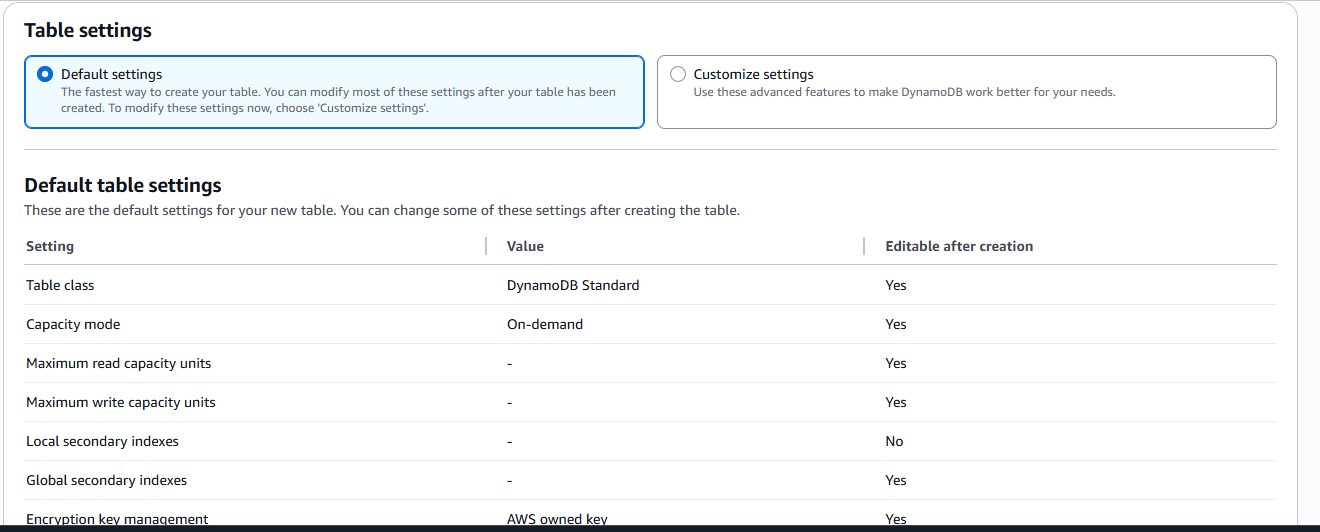
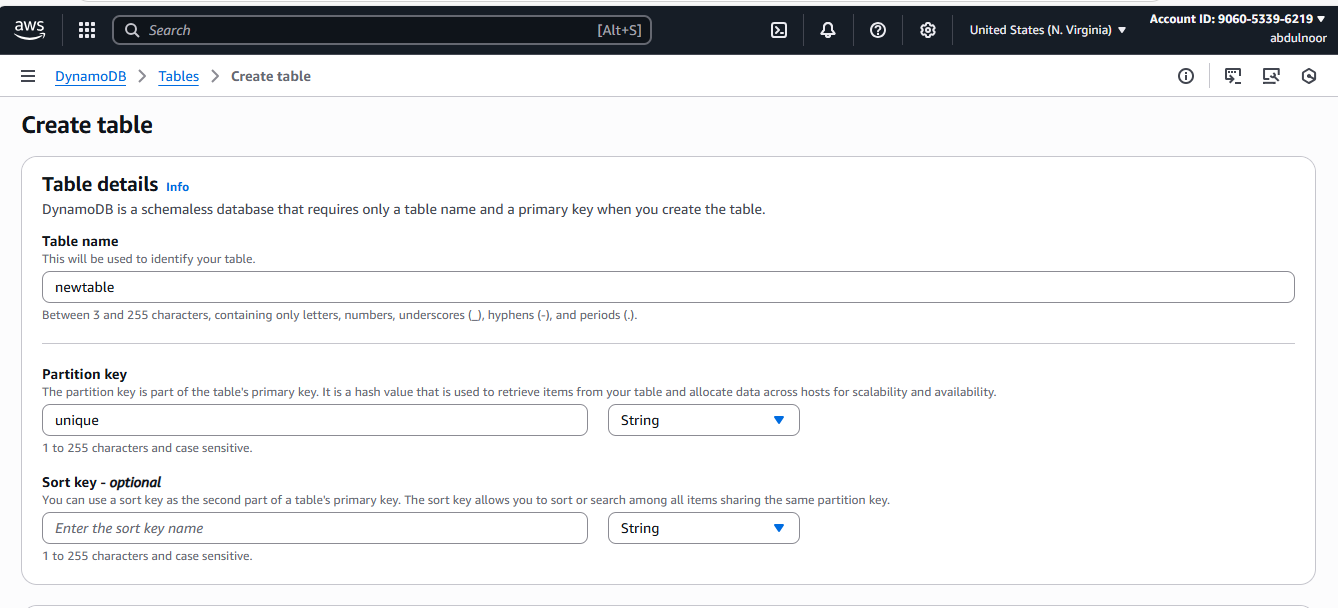
Now for this we need to create dynamodb.

Now go to aws console all services – under databases – click on DynamoDB

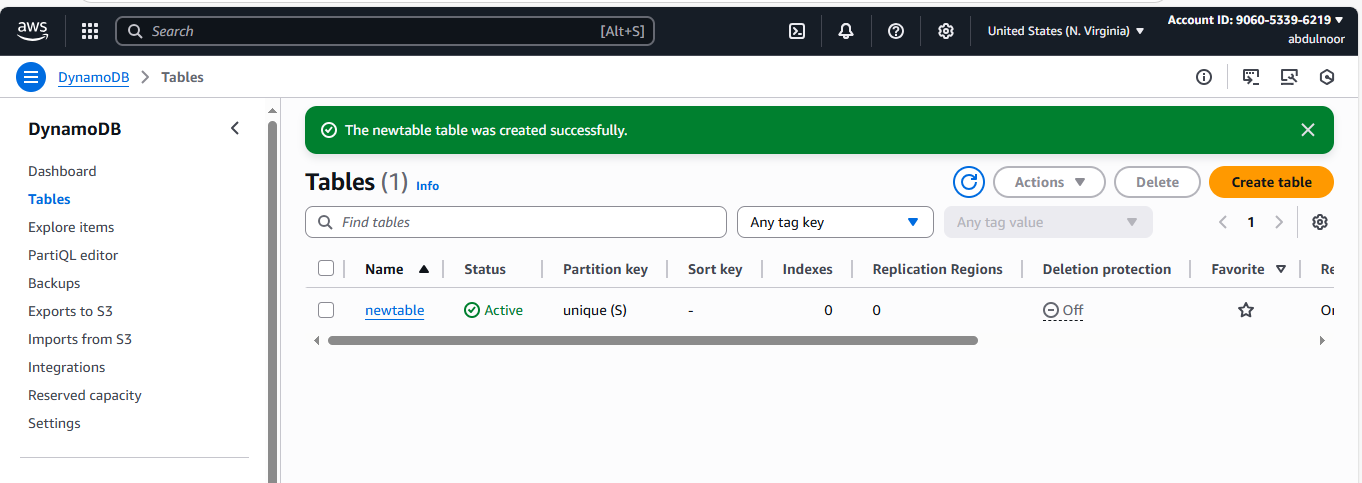


Click on create table

Give table whatever you kept in your code that is I kept newtable and item is unique



Click on create table

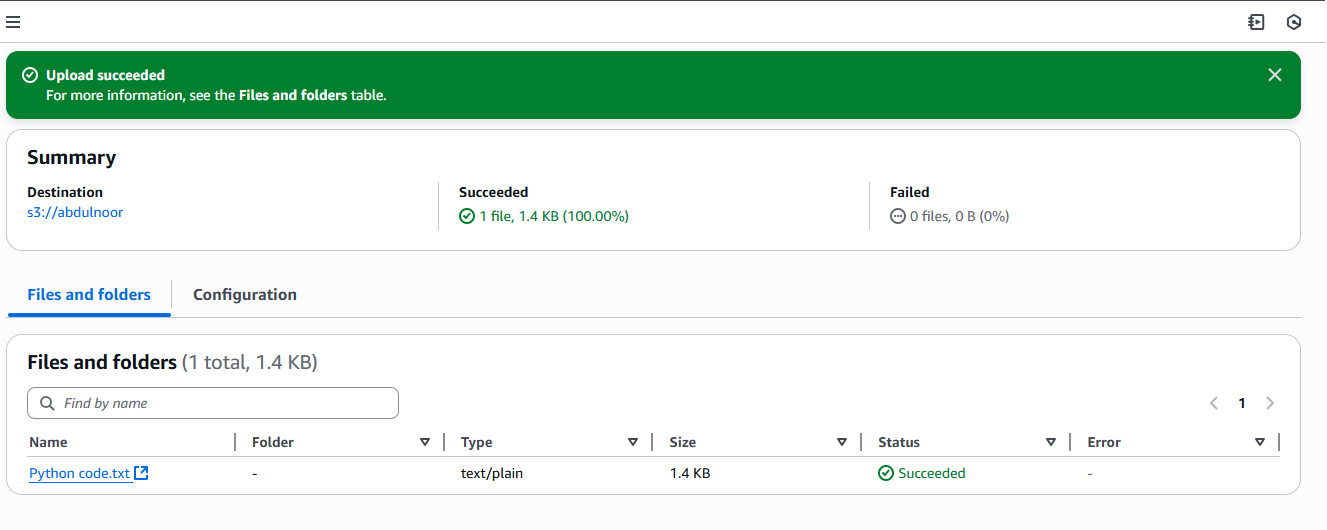


Selct your table and click Actions – click on explore items

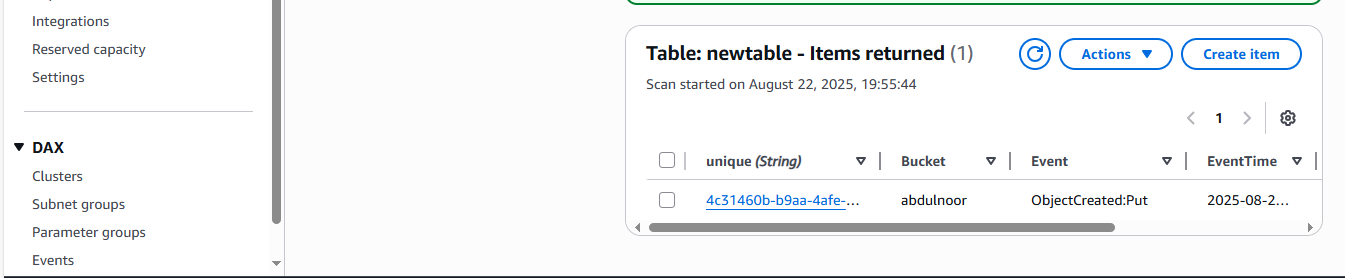
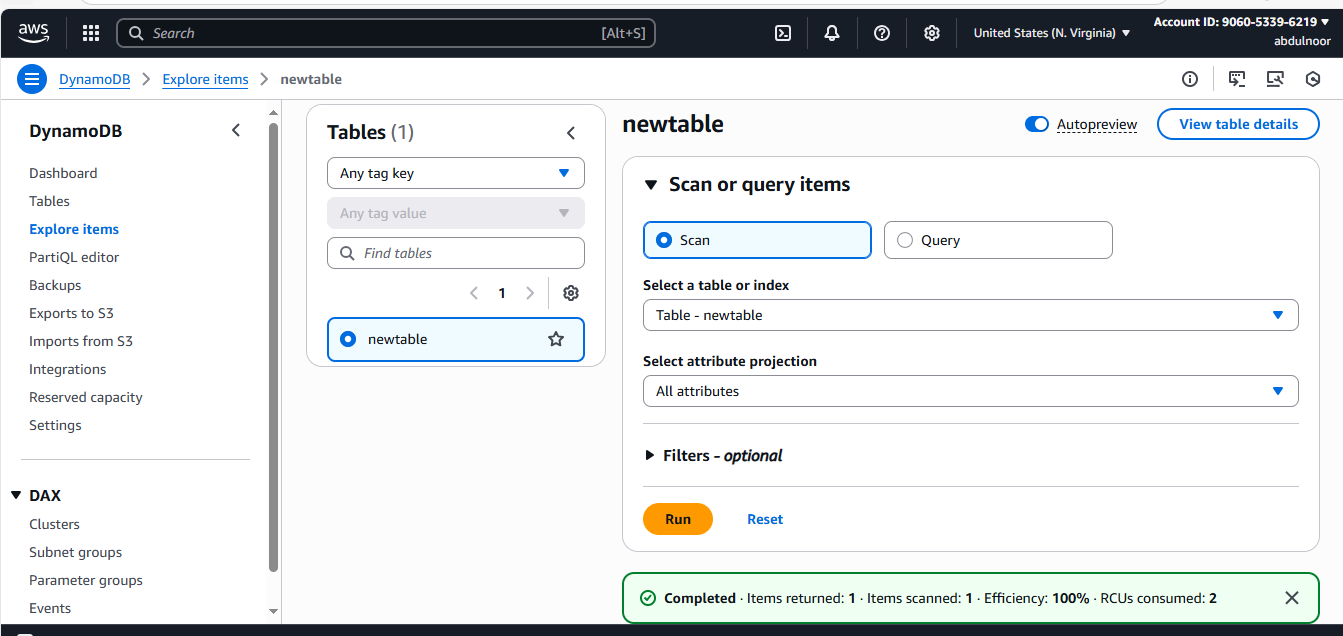
Here you can see there is no any items in your table

Now if I keep any items in the S3 bucket then will it be any entry in DDB automatedly

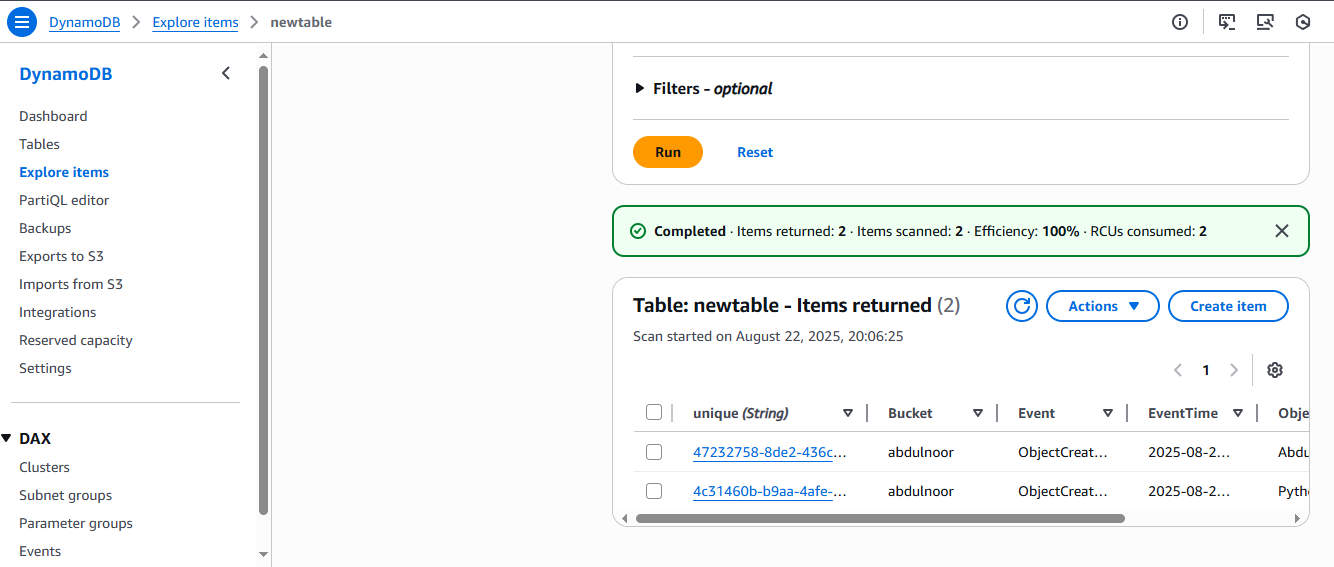
Go to S3 bucket and upload item/object



Now check in DDB any item automatically added or not



Add one more object



See previously it was 1 now 2 items are appearing

.