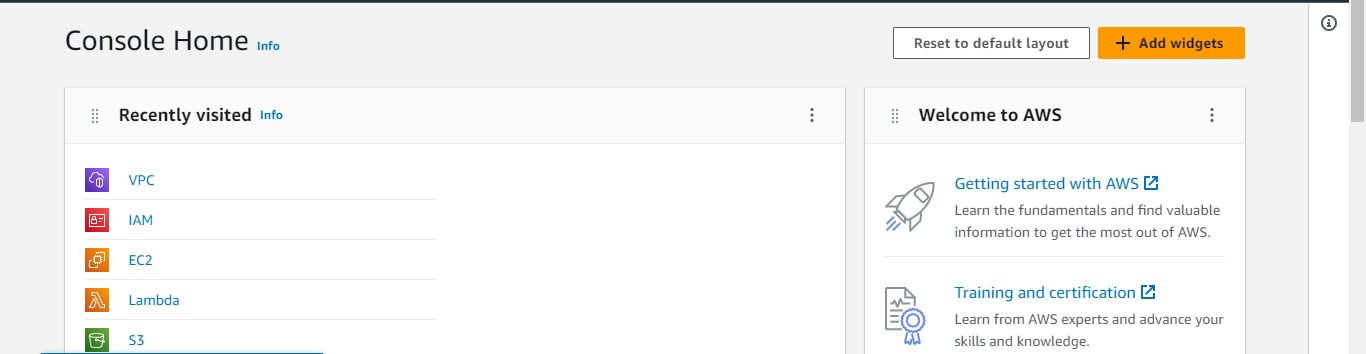
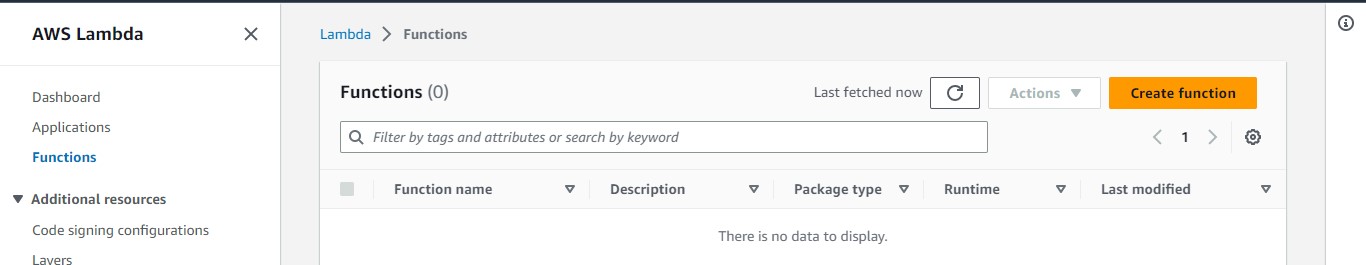
# **Experiment – 10**

**Creating, starting and stopping EC2 instances using AWS Lambda.**

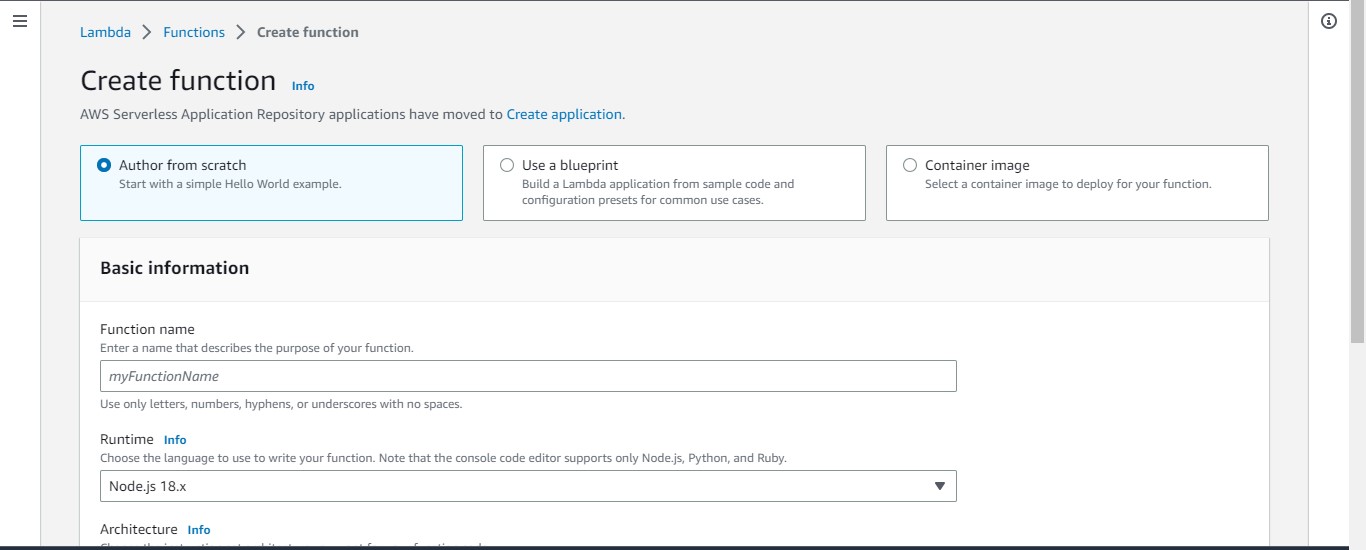
**Step 1:** Login to AWS Console and goto Lambda



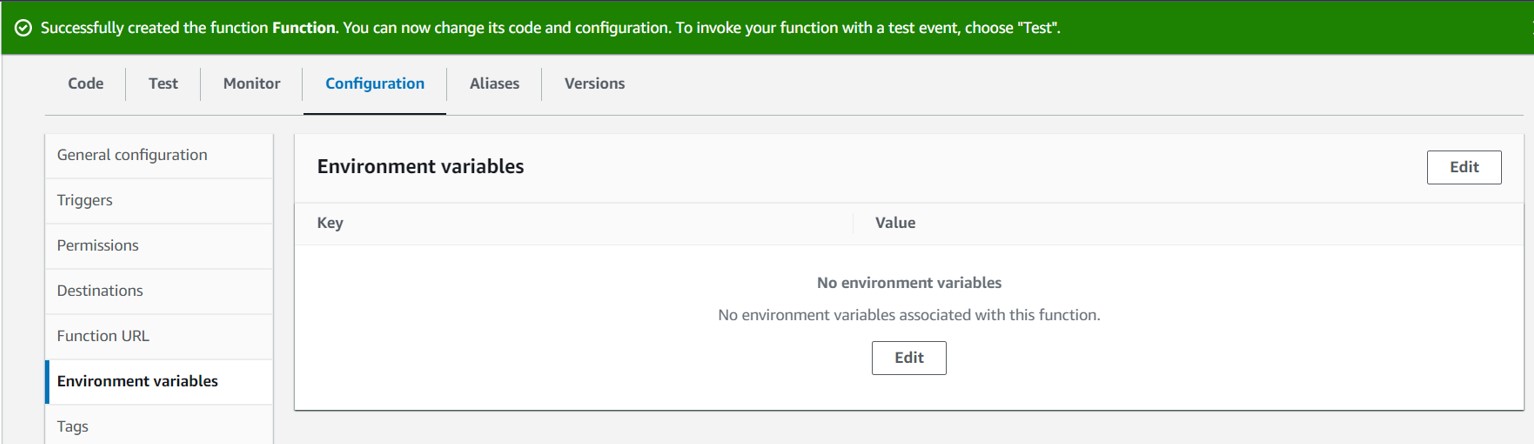
**Step 2:** In Lambda create a new function.



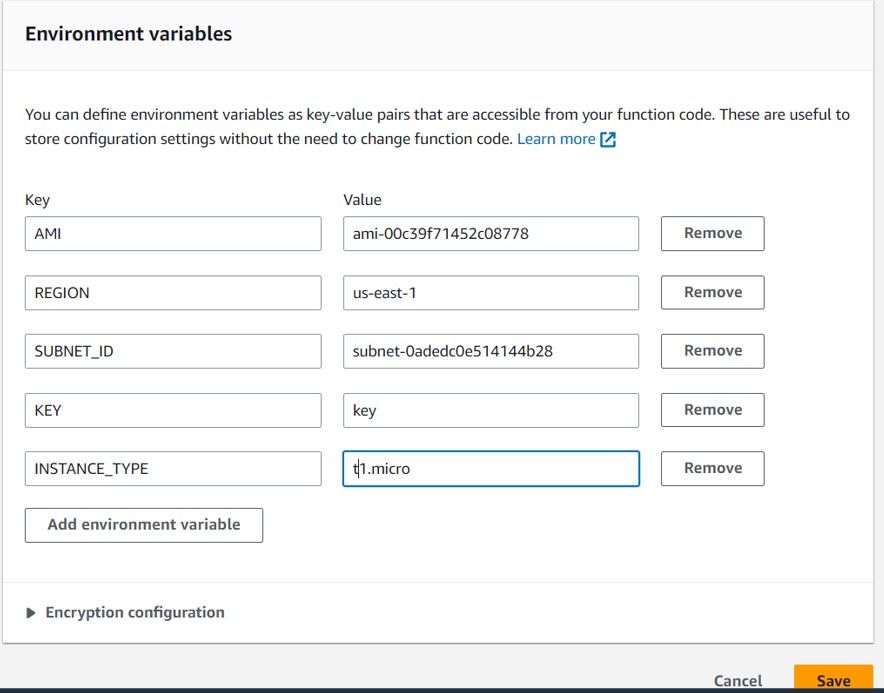
**Step 3:** Give a name to the function & assign a name, runtime to the function



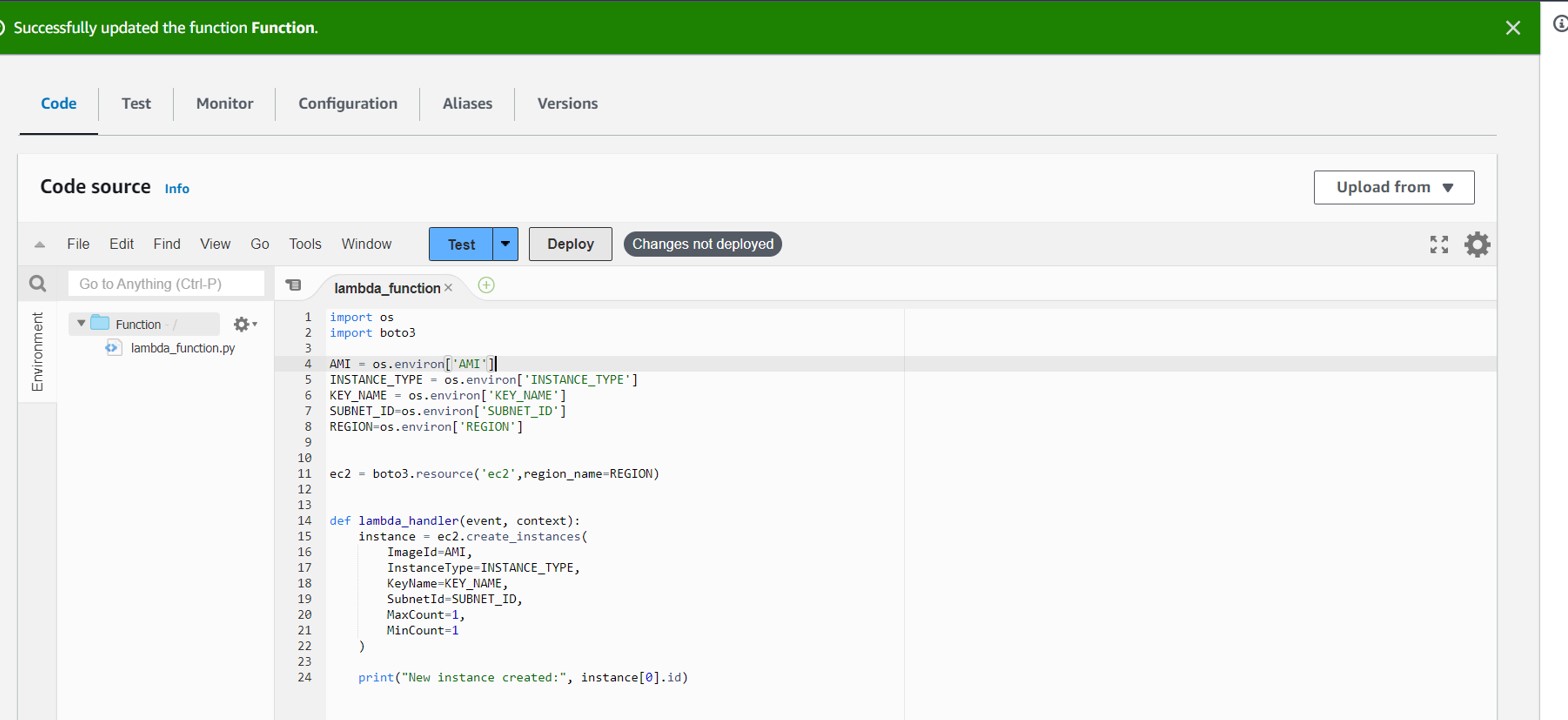
**Step 4:** Click on create function and goto configration



**Step 5:** Add environment variable by providing AMI,InstanceType,Region,key



**Step 6:** In the code section write a python script that creates a EC2 instance



#Python Script to create an EC2 instance import os

import boto3

AMI = os.environ['AMI']

INSTANCE\_TYPE = os.environ['INSTANCE\_TYPE']

KEY\_NAME = os.environ['KEY\_NAME']

SUBNET\_ID=os.environ['SUBNET\_ID']

REGION=os.environ['REGION']

ec2 = boto3.resource('ec2',region\_name=REGION) def lambda\_handler(event, context): instance = ec2.create\_instances(

ImageId=AMI,

InstanceType=INSTANCE\_TYPE,

KeyName=KEY\_NAME,

SubnetId=SUBNET\_ID,MaxCount=1, MinCount=1

)

print("New instance created:", instance[0].id)

**Step 7:** Now goto IAM and create a custom policy with the JSON Script-

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents"

],

"Resource": "arn:aws:logs:\*:\*:\*"

},

{

"Action": [

"ec2:RunInstances"

],

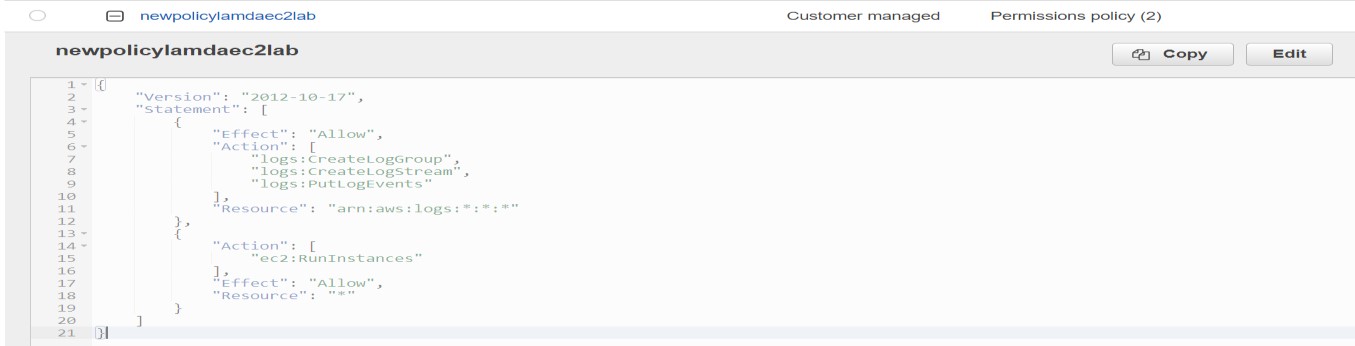
"Effect": "Allow",

"Resource": "\*"

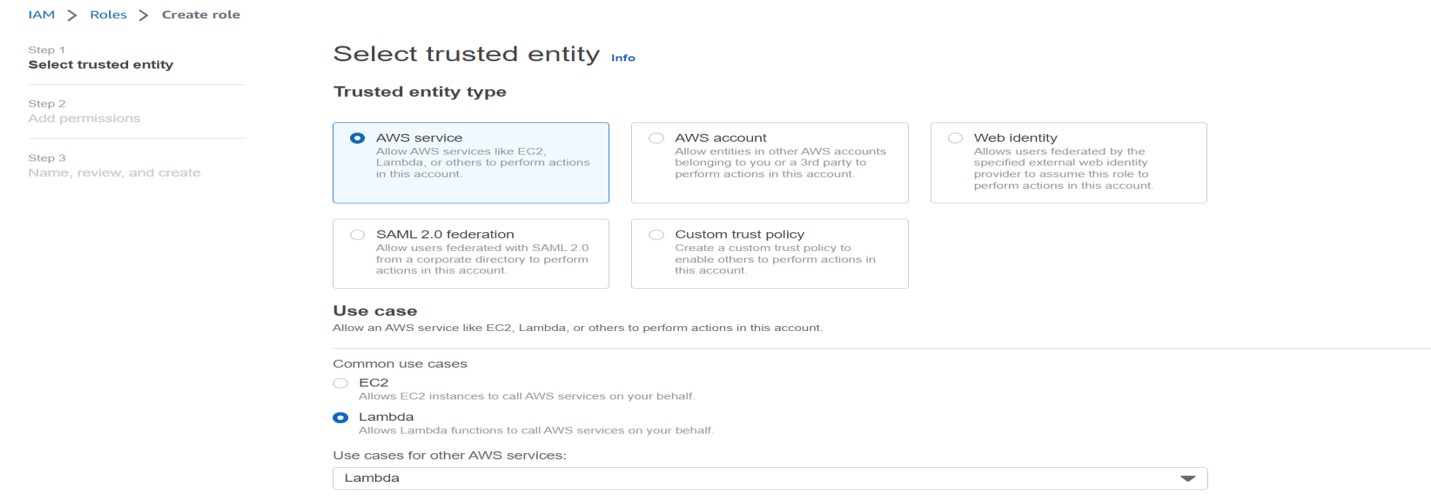
}

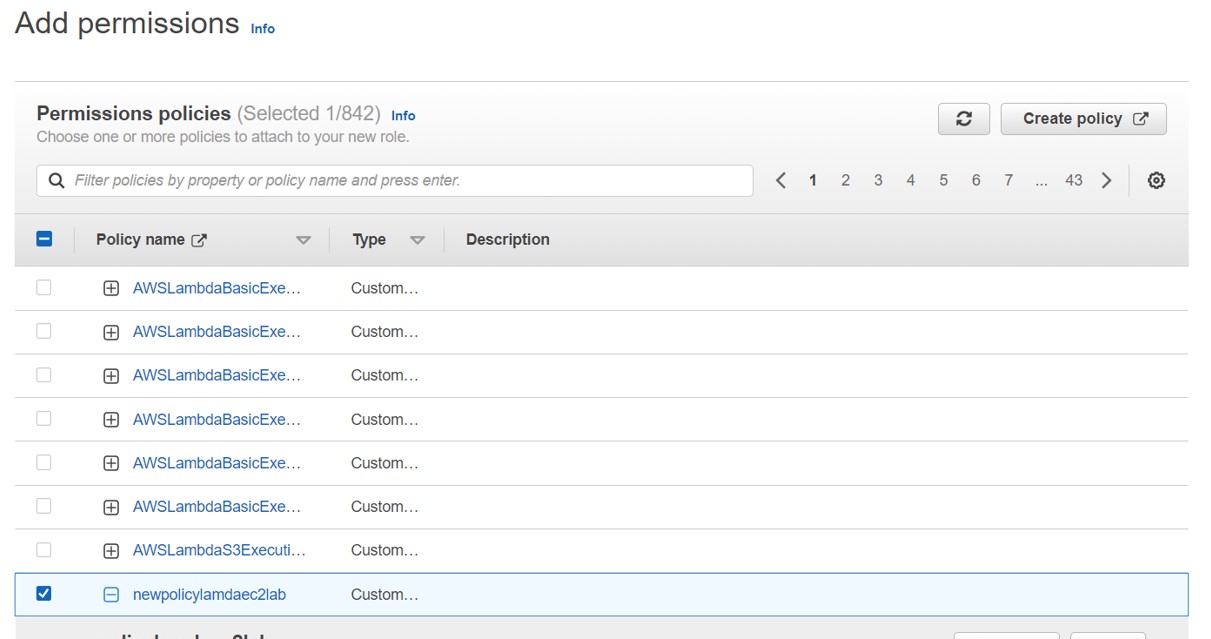
]

}



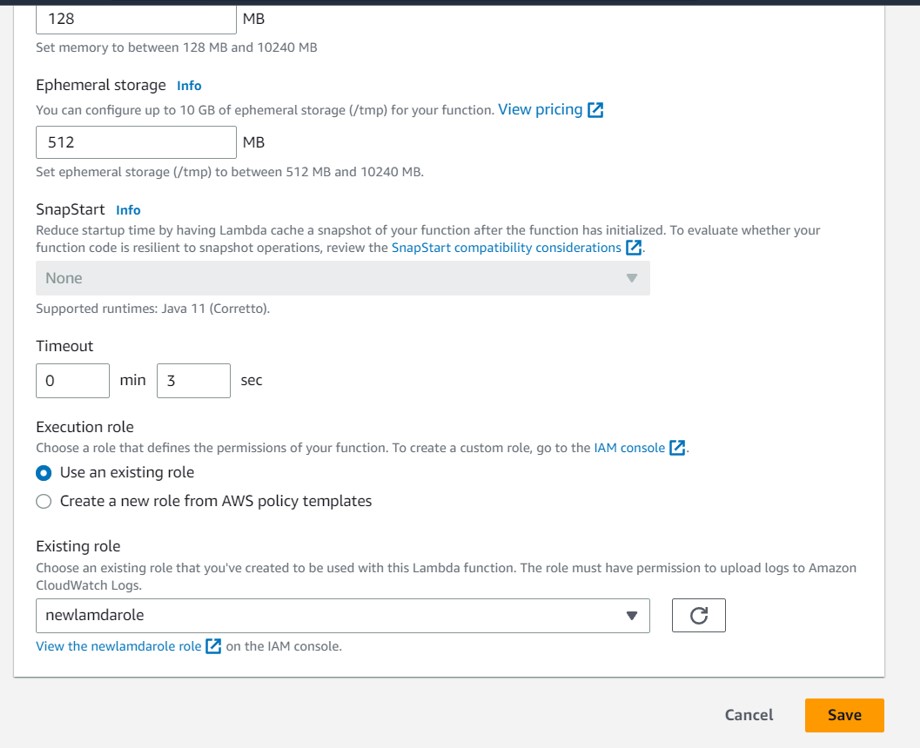
**Step 8:** Create an IAM role for Lambda service and attach the above policy to it



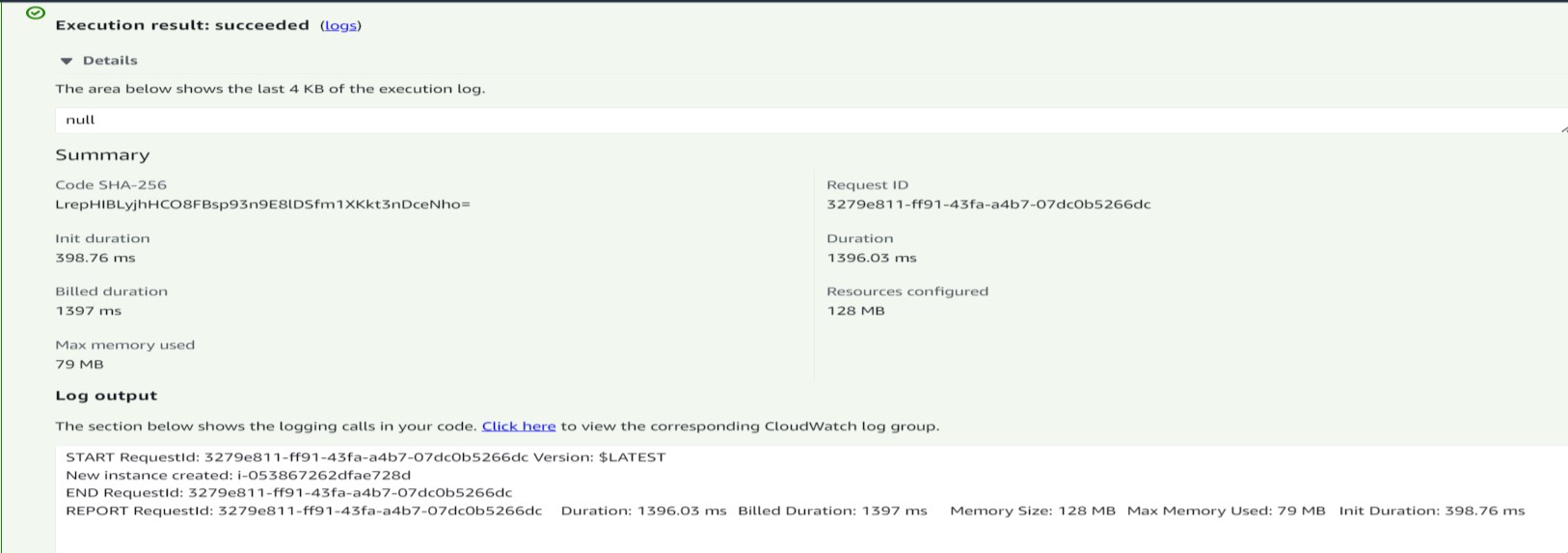


**Step 9:** Goto Lambda-> Function-> Configration->Exection Role & add created role

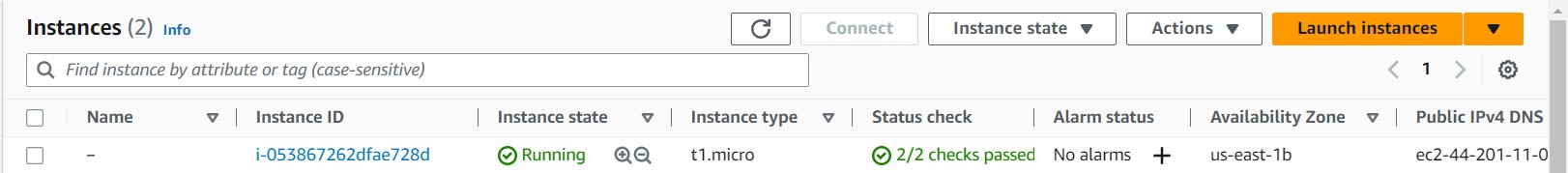




**Step 10:** Now goto Function -> Test

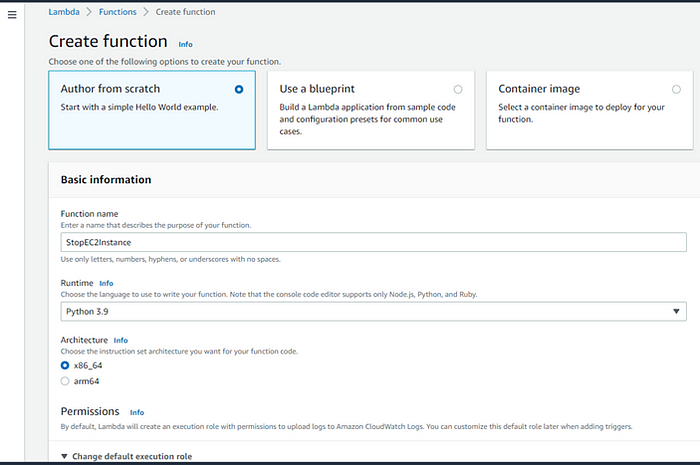


**Step 11:** Goto EC2 instances and the new instance would be present. Now so far we have created and started the EC2 instance.

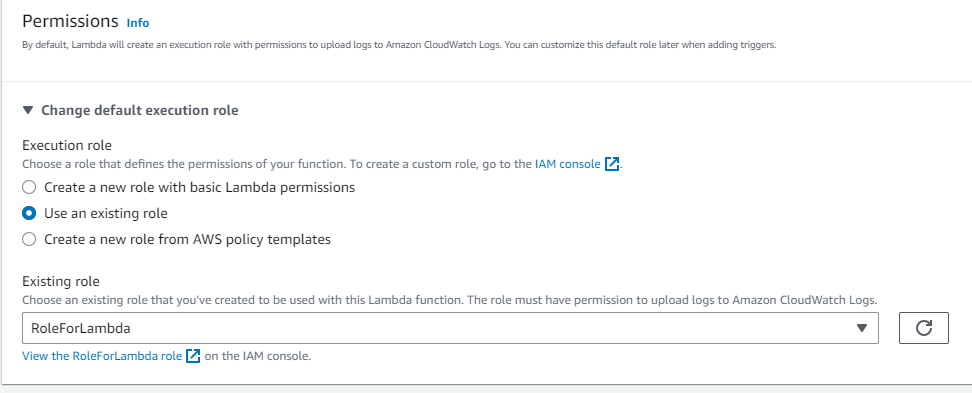


**Now let’s stop the EC2 instance using AWS Lambda**

**Step 1:** Navigate to the ‘Lambda’ module and click create function.



**Step 2:** Navigate to the **Permissions** section, select **use an existing role**, in **existing role** add role which we have create above and click on **Create function**

****

**Step 3:** Run that Code.

import boto3

region = ‘ap-south-1’// region in which the instance is made

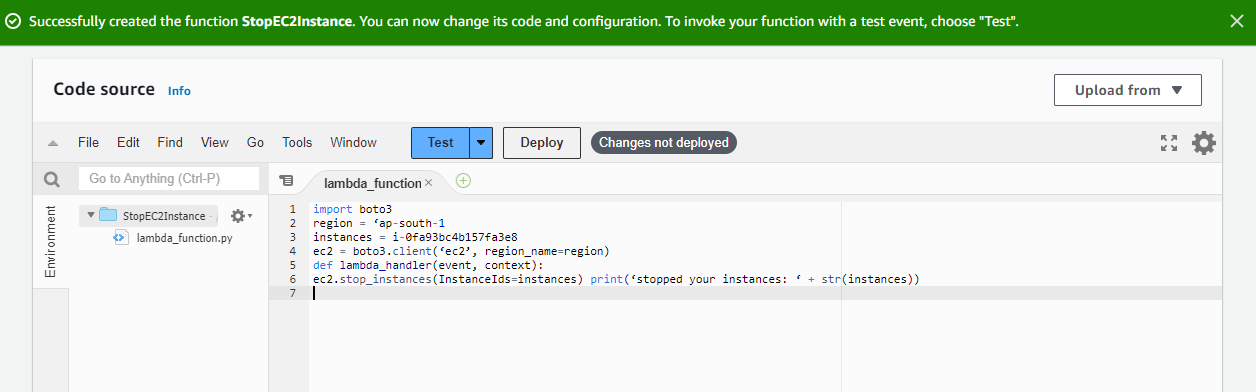
instances = [i-0fa93bc4b157fa3e8](https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#InstanceDetails:instanceId=i-0fa93bc4b157fa3e8) // our instance id

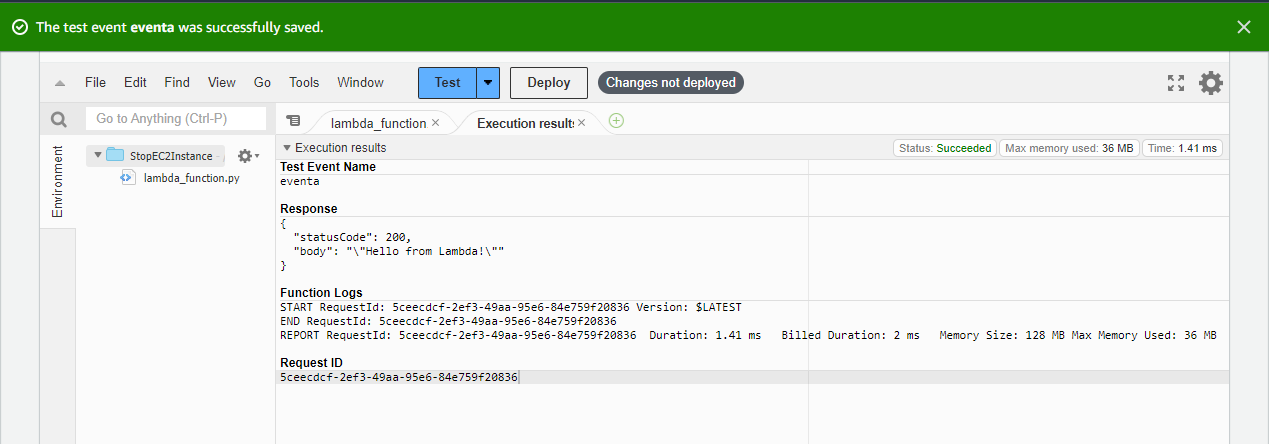
ec2 = boto3.client(‘ec2’, region\_name=region)

def lambda\_handler(event, context):

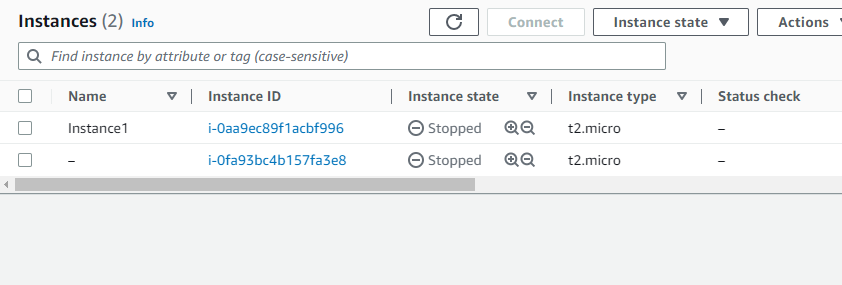
ec2.stop\_instances(InstanceIds=instances) print(‘stopped your instances: ‘ + str(instances))

**Step 4:** Now Click on Lambda function, Go to **Test**, click on **Configure test Events**.





**Step-5:** Now we can see that our instance with ID ‘[i-0fa93bc4b157fa3e8](https://ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#InstanceDetails:instanceId=i-0fa93bc4b157fa3e8)’ has stopped.



**QUESTIONS**

**1.What is AWS Lambda? Explain the working and benefits of AWS lambda in detail.**

**Ans:-** AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). It allows developers to run their code without managing servers, which can greatly reduce the operational overhead of deploying and scaling applications.

AWS Lambda works by allowing developers to upload their code to AWS Lambda and then configuring an event source, such as an API Gateway or S3 bucket, to trigger the execution of the code. When an event occurs, AWS Lambda automatically runs the code in response to the event, and then shuts down the execution environment once the code has completed running. This means that you only pay for the time your code is actually running, and there is no need to pay for idle time or to provision and manage servers.

One of the primary benefits of AWS Lambda is its scalability. As more events occur, AWS Lambda will automatically scale the execution environment to handle the increased load, ensuring that your code can handle any amount of traffic without needing to manually scale your infrastructure. Additionally, AWS Lambda integrates with a wide variety of AWS services, such as Amazon S3, DynamoDB, and API Gateway, making it easy to build serverless applications that can handle a wide range of use cases.

Another benefit of AWS Lambda is its cost-effectiveness. Because you only pay for the time your code is actually running, AWS Lambda can be much cheaper than traditional server-based computing models, especially for applications with variable or sporadic traffic patterns. Additionally, AWS Lambda offers a free tier that includes 1 million free requests per month and 400,000 GB-seconds of compute time per month, making it easy to get started with the service without incurring any costs.

Overall, AWS Lambda is a powerful tool that can greatly simplify the process of building and deploying applications in the cloud. Its scalability, flexibility, and cost-effectiveness make it an attractive choice for a wide range of use cases, from simple data processing tasks to complex, multi-tier applications.

**2.Explain in detail the use cases of AWS Lambda?**

**Ans:-** AWS Lambda is a serverless computing platform that provides a wide range of use cases across various industries. Here are some examples of how AWS Lambda can be used:

* **Event-driven computing**: AWS Lambda can be used to create eventdriven architectures that can respond to real-time events. This makes it ideal for building chatbots, real-time data processing applications, and event-driven automation.
* **Web applications:** AWS Lambda can be used to build backend services for web applications. For example, it can be used to process form data, authenticate users, or perform other backend tasks for web applications.
* **Data processing:** AWS Lambda can be used to process data from various sources. For example, it can be used to process log files, create thumbnails for images, or perform data transformation tasks.
* **Internet of Things (IoT):** AWS Lambda can be used to create serverless applications that interact with IoT devices. This makes it ideal for building smart homes, industrial automation, and other IoT applications.
* **Mobile applications**: AWS Lambda can be used to build backend services for mobile applications. For example, it can be used to process user data, perform authentication, or push notifications.
* **Machine learning**: AWS Lambda can be used to build machine learning models and perform real-time predictions. It can also be used to trigger training jobs and perform model evaluation tasks.
* **Batch processing**: AWS Lambda can be used to perform batch processing tasks that require large-scale data processing. For example, it can be used to process data for financial reports, inventory management, or customer analytics.
* **Serverless architectures**: AWS Lambda can be used to build serverless architectures that do not require the management of servers. This makes it ideal for building applications that require automatic scaling, high availability, and cost-effectiveness.

In summary, AWS Lambda provides a wide range of use cases that can simplify the development and deployment of applications in the cloud. Its flexibility, scalability, and cost-effectiveness make it an ideal platform for a wide range of applications across various industries.