

## Experiment No. 11

**Aim:** To explain AWS Lambda, its workflow, various functions and create your first Lambda functions using Python/Java/Nodejs. (LO1, LO6)

### Theory:

#### What is AWS Lambda?

AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of Amazon Web Services. Therefore, you don't need to worry about which AWS resources to launch, or how will you manage them. Instead, you need to put the code on Lambda, and it runs. In AWS Lambda the code is executed based on the response of events in AWS services such as add/delete files in S3 bucket, HTTP request from Amazon API gateway, etc. AWS Lambda function helps you to focus on your core product and business logic instead of managing operating system (OS) access control, etc.

#### How does AWS Lambda work?

The following AWS Lambda example with block diagram explains the working of AWS Lambda in a few easy steps:

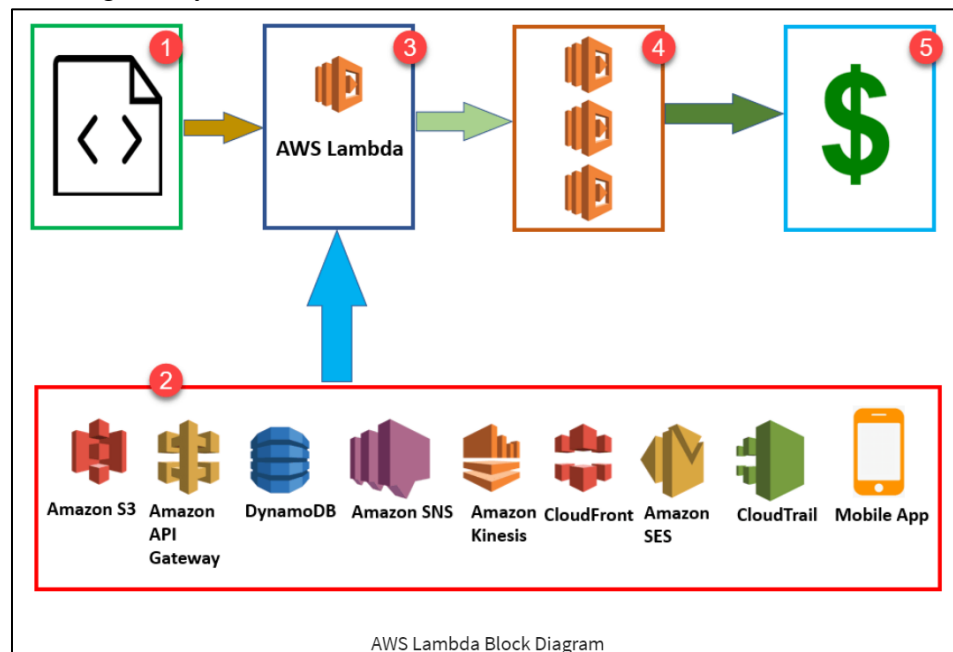
**Step 1:** First upload your AWS Lambda code in any language supported by AWS Lambda. Java, Python, Go, and C# are some of the languages that are supported by AWS Lambda function.

**Step 2:** These are some AWS services which allow you to trigger AWS Lambda.

**Step 3:** AWS Lambda helps you to upload code and the event details on which it should be triggered.

**Step 4:** Executes AWS Lambda Code when it is triggered by AWS services:

**Step 5:** AWS charges only when the AWS lambda code executes, and not otherwise.



### **AWS Lambda Concepts:**

- **Function:** A function is a program or a script which runs in AWS Lambda. Lambda passes invocation events into your function, which processes an event and returns its response.
- **Runtimes:** Runtime allows functions in various languages which runs on the same base execution environment. This helps you to configure your function in runtime. It also matches your selected programming language.
- **Event source:** An event source is an AWS service, such as Amazon SNS, or a custom service. This triggers function helps you to executes its logic.
- **Lambda Layers:** Lambda layers are an important distribution mechanism for libraries, custom runtimes, and other important function dependencies. This AWS component also helps you to manage your development function code separately from the unchanging code and resources that it uses.
- **Log streams:** Log stream allows you to annotate your function code with custom logging statements which helps you to analyse the execution flow and performance of your AWS Lambda functions.

### **Use Cases of AWS Lambda:**

AWS Lambda used for a wide range of applications like:

1. Helps you for ETL process
2. Allows you to perform real-time file processing and real-time stream processing
3. Use for creating web applications
4. Use in Amazon products like Alexa Chatbots and Amazon Echo/Alexa
5. Data processing (real-time streaming analytics)
6. Automated Backups of everyday tasks
7. Scalable back ends (mobile apps, IoT devices)
8. Helps you to execute server-side backend logic
9. Allows you to filter and Transform data

### **Advantages of using AWS Lambda:**

Here, are pros/benefits of using AWS lambda:

1. You can use it as a plugin for Eclipse and Visual Studio.
2. As it is serverless architecture, you don't need to worry about managing or provisioning servers.
3. Helps developers to run and execute the code's response to events without building any infrastructure.
4. You just need to pay for the compute time taken, only when your code runs.
5. You can monitor your code performance in real time through CloudWatch.
6. It allows you to run your code without provisioning or to manage any other server
7. You can scale it automatically to handle a few requests per day and even support more than thousands of requests per second.
8. AWS Lambda can be configured with the help of external event timers to perform scheduled tasks.
9. Lambda function in AWS should be configured with external event and timers so; it can be used for scheduling.
10. Lambda functions are stateless so that it can be scaled quickly.

### **Limitations of AWS Lambda:**

Here are the cons/disadvantages of using AWS Lambda:

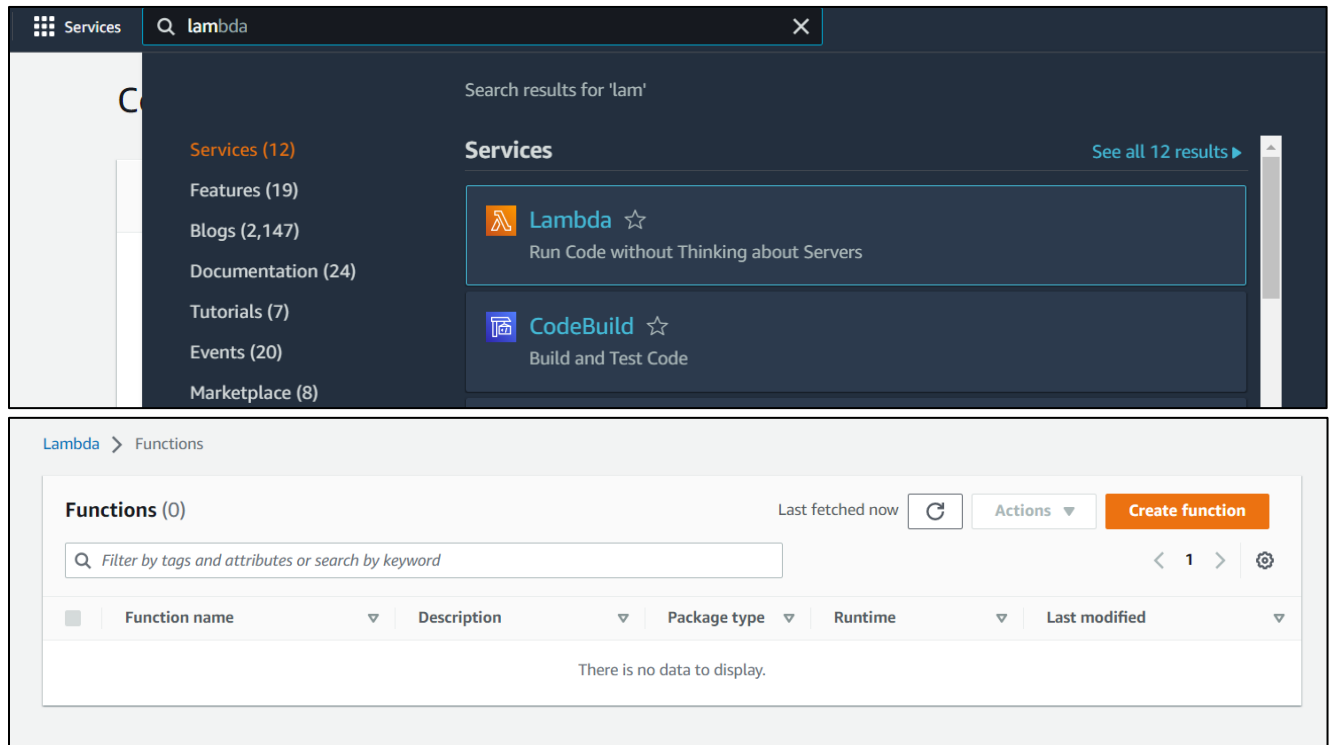
1. AWS Lambda tool is not suitable for small projects.
2. AWS Lambda entirely relies on AWS for the infrastructure, so you can't install any additional software if your code demands it.
3. Concurrent execution is limited to 100
4. AWS Lambda completely depended on AWS for the infrastructure; you cannot install anything additional software if your code demands it.
5. Its memory volume can vary between 128 to 1536 MB.
6. Event request should not exceed 128 KB.
7. Lambda functions help you to write their logs only in CloudWatch. This is the only tool that allows you to monitor or troubleshoot your functions.
8. Its code execution timeout is just 5 minutes.

Name: Harsh Dalvi  
Roll No: 13

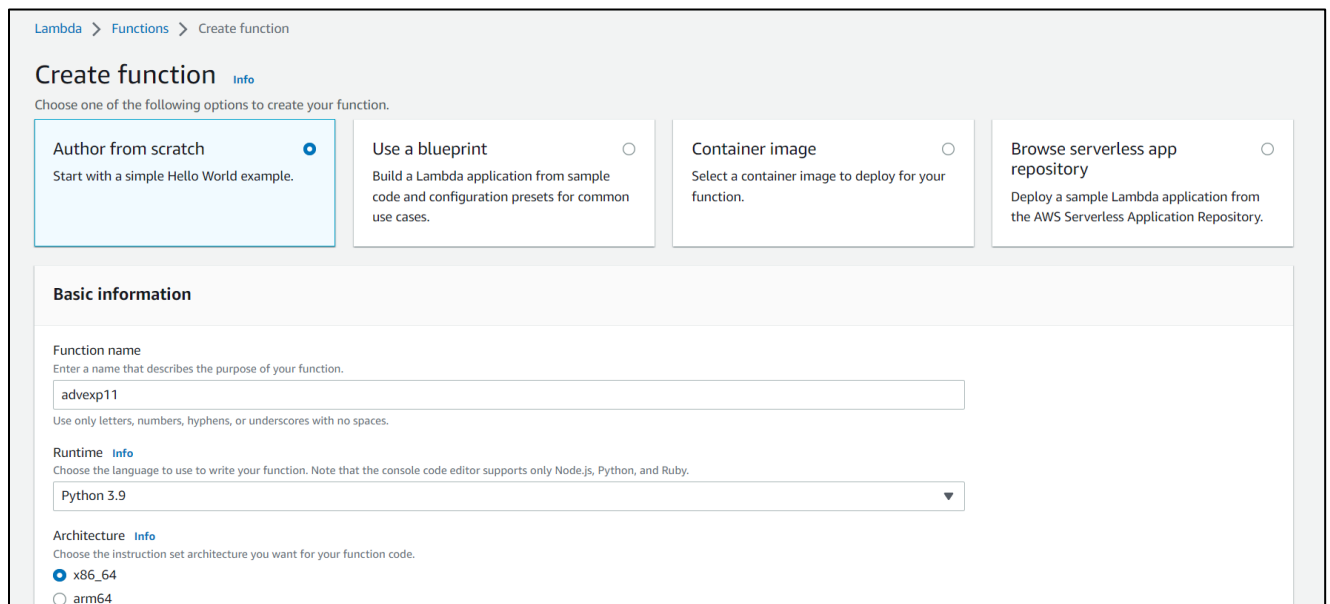
Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

## Steps to perform the Experiment:

**Step1:** Sign in to your AWS account and search Lambda and click on Lambda.



**Step 2:** Choose Create function.



Name: Harsh Dalvi  
Roll No: 13

Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

### Step 3: Go to IAM Roles and create a new role.

IAM > Roles

**Roles (16)** [Info](#)

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

[Refresh](#) [Delete](#) [Create role](#)

< 1 > [Settings](#)

<input type="checkbox"/>	Role name	Trusted entities	Last activity
<input type="checkbox"/>	<a href="#">adve12-role-5bb7wucj</a>	AWS Service: lambda	1 hour ago
<input type="checkbox"/>	<a href="#">advexp11-role-wlvye7o</a>	AWS Service: lambda	-
<input type="checkbox"/>	<a href="#">aws-elasticbeanstalk-ec2-role</a>	AWS Service: ec2	78 days ago
<input type="checkbox"/>	<a href="#">aws-elasticbeanstalk-service-role</a>	AWS Service: elasticbeanstalk	78 days ago

IAM > Roles > Create role

Step 1  
**Select trusted entity**

Step 2  
Add permissions

Step 3  
Name, review, and create

### Select trusted entity [Info](#)

**Trusted entity type**

☒ **AWS service**  
Allow AWS services like EC2, Lambda, or others to perform actions in this account.

☐ **AWS account**  
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

☐ **Web identity**  
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

☐ **SAML 2.0 federation**  
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

☐ **Custom trust policy**  
Create a custom trust policy to enable others to perform actions in this account.

**Use case**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

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.

Use cases for other AWS services:

[Cancel](#) [Next](#)

<input type="checkbox"/>	<a href="#">AWSLambdaVPCLayerRole</a>	AWS m...	Provides minimum permissions for a Lambda function to execu...
<input type="checkbox"/>	<a href="#">AWSLambdaRole</a>	AWS m...	Default policy for AWS Lambda service role.
<input type="checkbox"/>	<a href="#">AWSLambdaENIMa...</a>	AWS m...	Provides minimum permissions for a Lambda function to mana...
<input type="checkbox"/>	<a href="#">AWSLambdaMSKE...</a>	AWS m...	Provides permissions required to access MSK Cluster within a ...
<input type="checkbox"/>	<a href="#">AWSLambda_Read...</a>	AWS m...	Grants read-only access to AWS Lambda service, AWS Lambd...
<input checked="" type="checkbox"/>	<a href="#">AWSLambda_FullA...</a>	AWS m...	Grants full access to AWS Lambda service, AWS Lambda cons...

► **Set permissions boundary - optional** [Info](#)

Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting, but you can use it to delegate permission management to others.

[Cancel](#) [Previous](#) [Next](#)

Name: Harsh Dalvi  
Roll No: 13

Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

[IAM](#) > [Roles](#) > Create role

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

## Name, review, and create

### Role details

**Role name**  
Enter a meaningful name to identify this role.

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

**Description**  
Add a short explanation for this role.

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

Step 1: Select trusted entities Edit

```
1- {
2-   "Version": "2012-10-17",
3-   "Statement": [
4-     {
5-       "Effect": "Allow",
6-       "Action": [
7-         "sts:AssumeRole"
8-       ],
9-       "Principal": {
10-        "Service": [
11-          "lambda.amazonaws.com"
12-        ]
13-      }
14-    }
15-  ]
16- }
```

Step 2: Add permissions Edit

Permissions policy summary

Policy name <a href="#">?</a>	Type	Attached as
AWSLambda_FullAccess	AWS managed	Permissions policy

Tags

**Add tags - optional** [Info](#)  
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add tag  
You can add up to 50 more tags.

Cancel Previous Create role

✓ Role Lambda\_XIE created. View role ✕

[IAM](#) > Roles

**Roles** (Selected 1/17) [Info](#)

Refresh Delete Create role

X 1 match < 1 > [Settings](#)

<input checked="" type="checkbox"/>	Role name	Trusted entities	Last activity
<input checked="" type="checkbox"/>	Lambda_XIE	AWS Service: lambda	-

Name: Harsh Dalvi  
Roll No: 13

Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

#### Step 4: Add IAM role to Lambda function.

▼ Change default 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\_XIE

[View the Lambda\\_XIE role on the IAM console.](#)

► Advanced settings

Cancel **Create function**

And click on Create function.


☑ Successfully created the function advexp11. You can now change its code and configuration. To invoke your function with a test event, choose "Test".


Lambda > Functions > advexp11

advexp11

Throttle Copy ARN Actions ▼

▼ Function overview Info

 advexp11

 Layers (0)

+ Add trigger

+ Add destination


Description

-

Last modified

4 seconds ago

Function ARN

 arn:aws:lambda:ap-south-1:052585512669:function:advexp11

Function URL [Info](#)

-

#### Step 5: Go to the Code section and print something.

To configure a test event, choose Test.

Code Test Monitor Configuration Aliases Versions

Code source Info

Upload from ▼

File Edit Find View Go Tools Window Test ▼ Deploy Changes not deployed

Go to Anything (Ctrl-P)

Environment

advexp11 - /

lambda\_function.py

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     print("Hello XIE, Welcome to TE-IT")
6     print(event)
7     return {
8         'statusCode': 200,
9         'body': json.dumps('Hello from Lambda!')}
10
11
```

Name: Harsh Dalvi  
Roll No: 13

Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

**Step 6:** For Event name, enter event\_exp11.

### Configure test event

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

☒ Create new event

☐ Edit saved event

Event name

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☐ Private  
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☒ Shareable  
This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

#### Event JSON

Format JSON

```
1 {  
2   "key1": "value1",  
3   "key2": "value2",  
4   "key3": "value3"  
5 }
```

The test event event\_exp11 was successfully saved.

Function URL [Info](#)

-

Code | Test | Monitor | Configuration | Aliases | Versions

Code source [Info](#)

File Edit Find View Go Tools Window 

Test

Deploy

Changes not deployed

Go to Anything (Ctrl-P)

Environment

advexp11 /

lambda\_function.py

lambda\_function

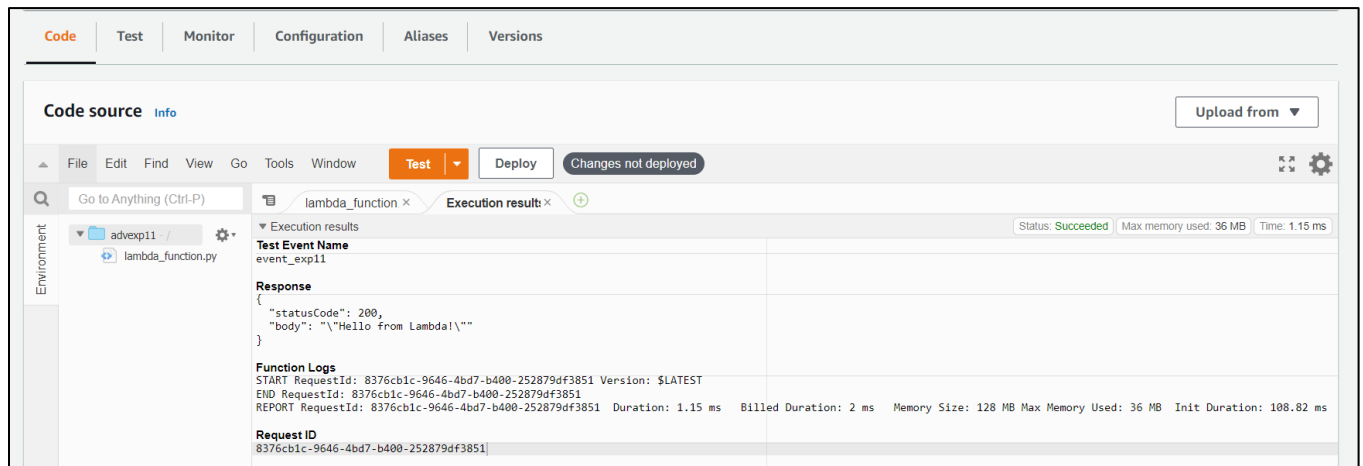
```
1 import json  
2  
3 def lambda_handler(event, context):  
4     # TODO implement  
5     print("Hello XIE, Welcome to TE-IT")  
6     print(event)  
7     return {  
8         'statusCode': 200,  
9         'body': json.dumps('Hello from Lambda!')  
10    }  
11
```



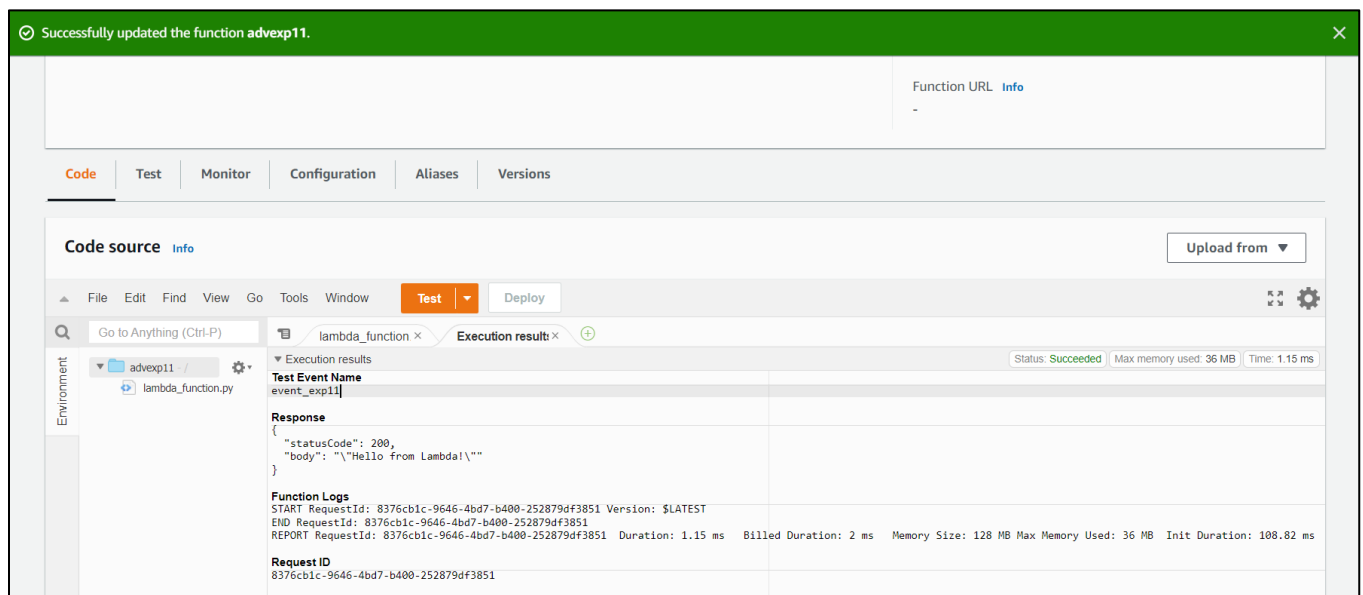
Name: Harsh Dalvi  
Roll No: 13

Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

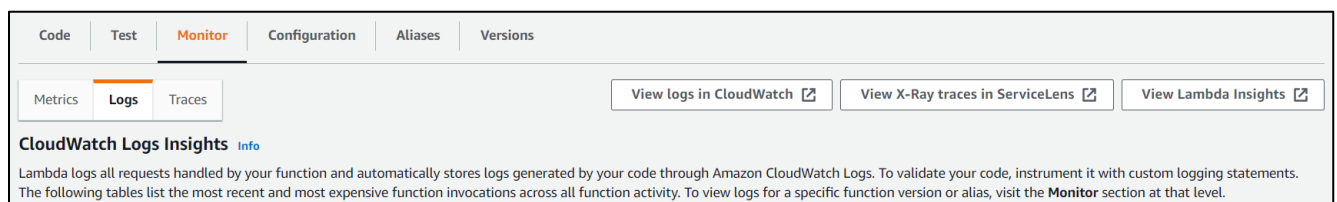
**Step 7:** To invoke the function, choose Test.



Click on Deploy.



**Step 8:** Now go to the Monitor section and click on View logs in CloudWatch



Name: Harsh Dalvi  
Roll No: 13

Subject: Advance DevOps , Sem: SEM V  
Class / Batch: TE-IT / Batch B

The screenshot displays the AWS CloudWatch console interface for a log group. The breadcrumb navigation at the top shows 'CloudWatch > Log groups > /aws/lambda/advexp11'. The main header area includes the log group name '/aws/lambda/advexp11' and three buttons: 'Actions', 'View in Logs Insights', and 'Search log group'. Below this is a 'Log group details' section with a table of properties:

Log group details		
Retention	Creation time	Subscription filters
Never expire	3 minutes ago	0
KMS key ID	Metric filters	Contributor Insights rules
-	0	-
	Stored bytes	ARN
	-	arn:aws:logs:us-east-1:553534940533:log-group:/aws/lambda/advexp11:*

Below the details section is a horizontal menu with tabs: 'Log streams', 'Metric filters', 'Subscription filters', 'Contributor Insights', and 'Tags'. The 'Log streams' tab is active. It shows 'Log streams (1)' with a search bar containing 'Filter log streams or try prefix search', a 'Delete' button, a 'Create log stream' button, and a 'Search all log streams' button. There is also an 'Exact match' checkbox and pagination controls showing '1' of 1 items. A table lists the log streams:

Log stream	Last event time
2022/10/22/[\$LATEST]00278159c1eb4233b97cc5b25080a997	2022-10-23 01:01:38 (UTC+05:30)

**Conclusion:** From this experiment, it is concluded that we have understood the concepts of AWS Lambda, its workflow and various functions. In this experiment, we have created Lambda functions using Python. AWS Lambda function helps us to focus on our core product and business logic instead of managing the operating system (OS) access control, OS patching, right-sizing, provisioning, scaling, etc. Hence, we have successfully achieved the Lab Outcome One and Lab Outcome Six (LO1 and LO6). Also, we have achieved PO1, PO2, PO3, PO4, PO5, PO9, PO10 and PO12 from this experiment.