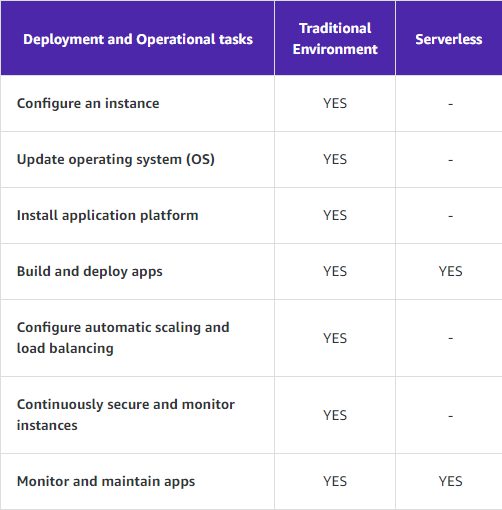
AWS Lambda



**Introduction to Serverless**

One of the major benefits of cloud computing is its ability to abstract (hide) the infrastructure layer. This ability eliminates the need to manually manage the underlying physical hardware. In a serverless environment, this abstraction allows you to focus on the code for your applications without spending time building and maintaining the underlying infrastructure. With serverless applications, there are never instances, operating systems, or servers to manage. AWS handles everything required to run and scale your application. By building serverless applications, your developers can focus on the code that makes your business unique.

# Serverless operational tasks



***What is AWS Lambda?***

AWS Lambda is a compute service. You can use it to run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure. It operates and maintains all of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring, and logging. With Lambda, you can run code for almost any type of application or backend service.

# Benefits of AWS Lambda

* You can run code without provisioning or managing servers
* It initiates functions for you in response to events
* It scales automatically
* It provides built-in code monitoring and logging via Amazon CloudWatch

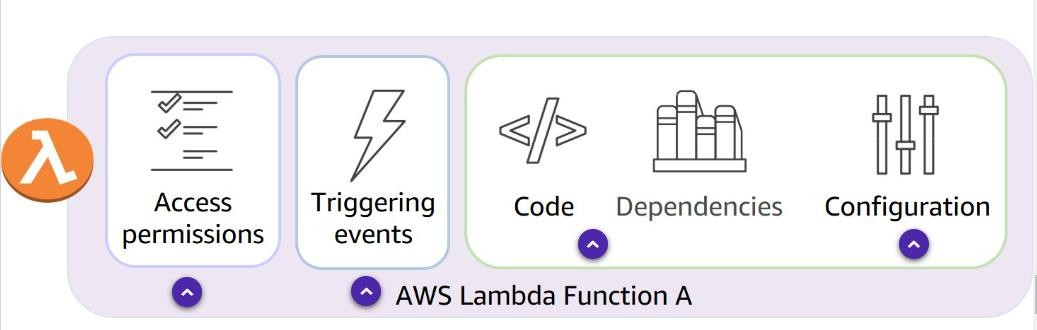
# Six main features of AWS Lambda

1. Bring your own code
2. Integrates with and brings other AWS services
3. Flexible resource and concurrency model
4. Flexible permissions model
5. Availability and fault tolerance are built-in
6. Pay for value

# What is a Lambda function?

The code you run on AWS Lambda is called a *Lambda function*. Think of a function as a small, self-contained application. After you create your Lambda function, it is ready to run as soon as it is initiated. Each function includes your code as well as some associated configuration information, including the function name and resource requirements. Lambda can rapidly launch as many copies of the function as needed to scale the rate of incoming events.

After you upload your code to AWS Lambda, you can configure an event source, such as an Amazon Simple Storage Service (Amazon S3) event, Amazon DynamoDB stream, Amazon Kinesis stream, or Amazon Simple Notification Service (Amazon SNS) notification. When the resource changes and an event is initiated, Lambda will run your function and manage the compute resources as needed to keep up with incoming requests.



Access Permissions define what services Lambda is permitted to interact with. Triggering events specify which events or event sources can initiate the function. Provide the code and any dependencies or libraries necessary to run your code.

Define the execution parameters, such as memory, timeout, and concurrency.

# Types of invocation methods in AWS Lambda

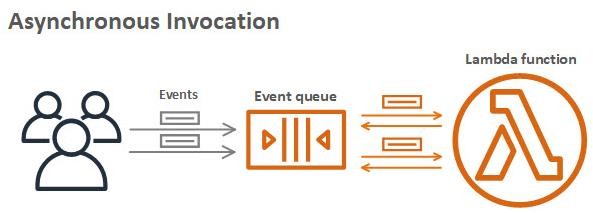
1. Synchronous Invocation

When you invoke a function synchronously, Lambda runs the function and waits for a response. When the function completes, Lambda returns the response from the function's code with additional data, such as the version of the function that was invoked. Synchronous events expect an immediate response from the function invocation.



1. Asynchronous Invocation

When you invoke a function asynchronously, events are queued and the requestor doesn't wait for the function to complete. This model is appropriate when the client doesn't need an immediate response.



1. Polling Invocation

This invocation model is designed to integrate with AWS streaming and queuing based services with no code or server management. Lambda will poll (or watch) these services, retrieve any matching events, and invoke your functions. This invocation model supports the following services:

* Amazon Kinesis
* Amazon SQS
* Amazon DynamoDB Streams

With this type of integration, AWS will manage the poll on your behalf and perform synchronous invocations of your function.

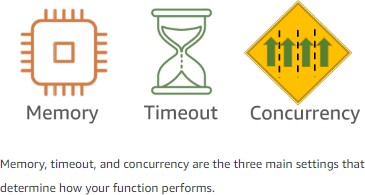
# Lambda execution environment

Lambda invokes (call on) your function in an execution environment, which is a secure and isolated environment. The execution environment manages the resources required to run your function. The execution environment also provides lifecycle support for the function's runtime and any external extensions associated with your function.



# Configuring Lambda functions

When building and testing a function, you must specify three primary configuration settings: memory, timeout, and concurrency. These settings are important in defining how each function performs.



You can allocate up to 10 GB of memory to a Lambda function. Lambda allocates CPU and other resources linearly in proportion to the amount of memory configured.

The AWS Lambda timeout value dictates how long a function can run before Lambda terminates the Lambda function. At the time of this publication, the maximum timeout for a Lambda function is 900 seconds. This limit means that a single invocation of a Lambda function cannot run longer than 900 seconds (which is 15 minutes).

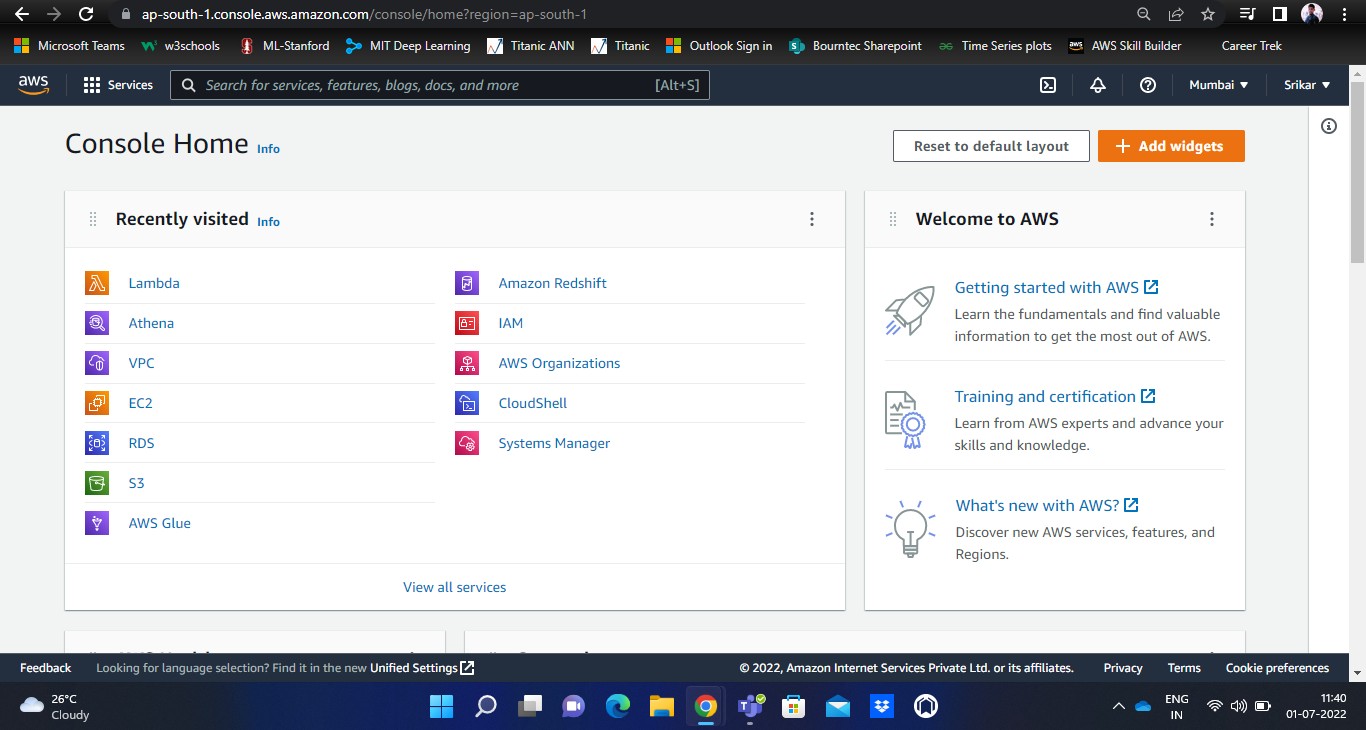
Concurrency is the number of invocations your function runs at any given moment. When your function is invoked, Lambda launches an instance of the function to process the event. When the function code finishes running, it can handle another request. If the function is invoked again while the first request is still being processed, another instance is allocated. Having more than one invocation running at the same time is the function's concurrency.

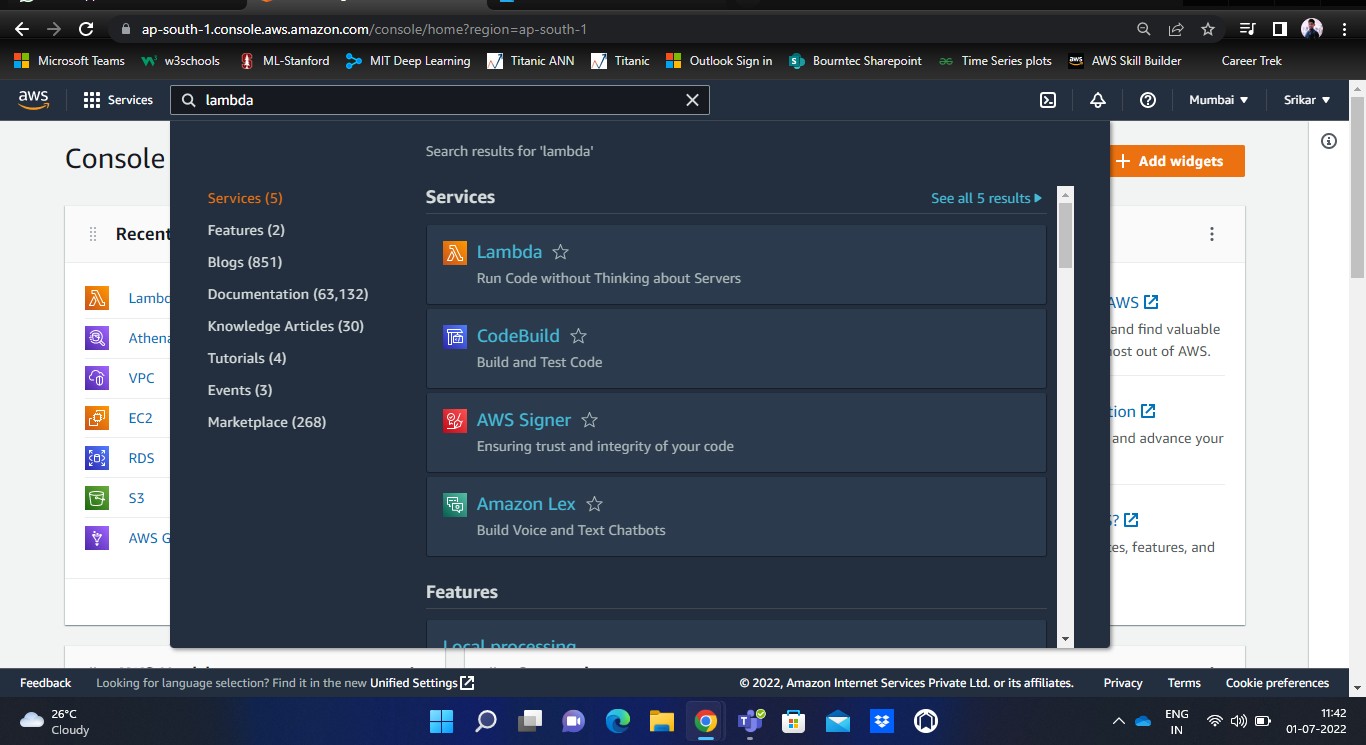
**Working with AWS Lambda**

***Step – 1*** Login process

Login to your AWS Management Console. Set up the respective region that you are in.

Then, type ‘Lambda’ in the console’s search bar. Finally, you will be navigated to the AWS Lambda page.

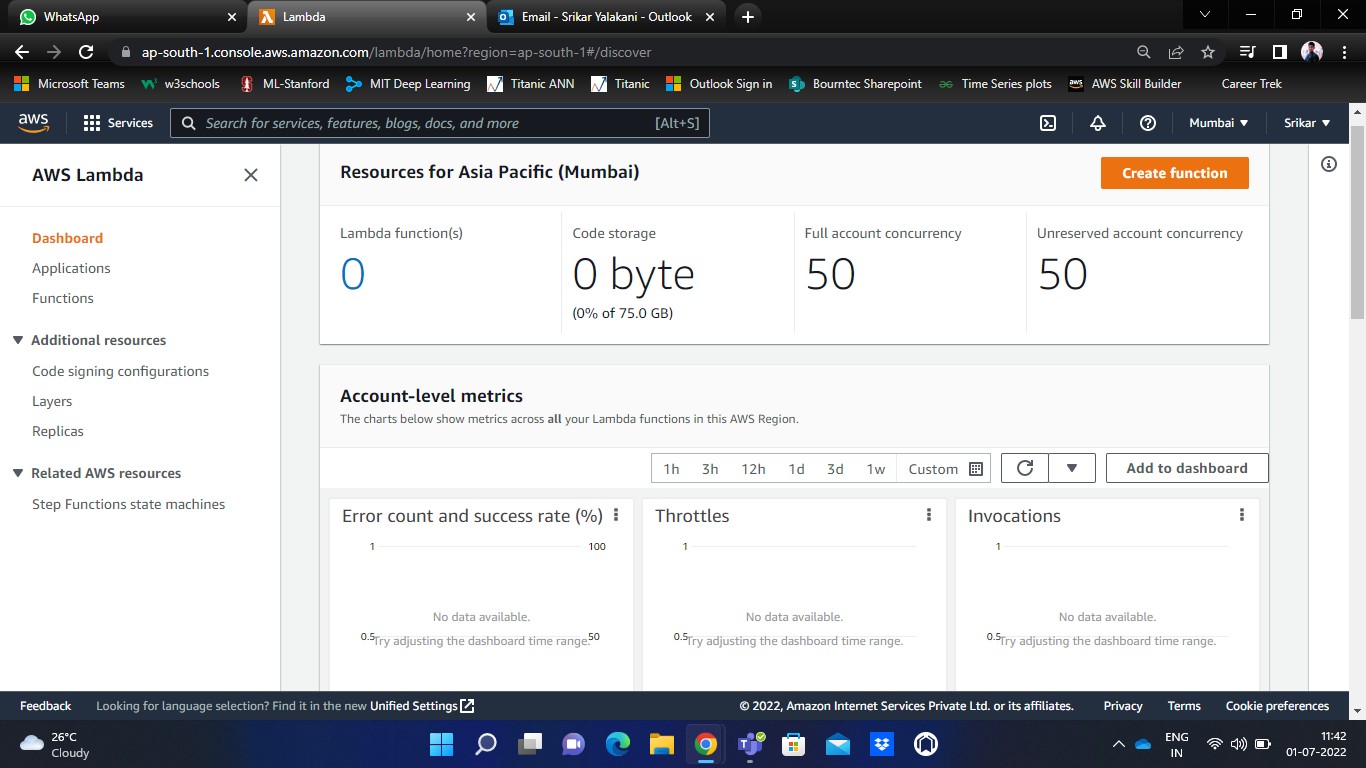




On the left navigation page of Lambda, you will find a few options like Dashboard, Applications, Functions, additional resources, etc.

***Step – 2*** Overview of the Lambda dashboard

Now, let’s look at the overview of the Lambda dashboard.

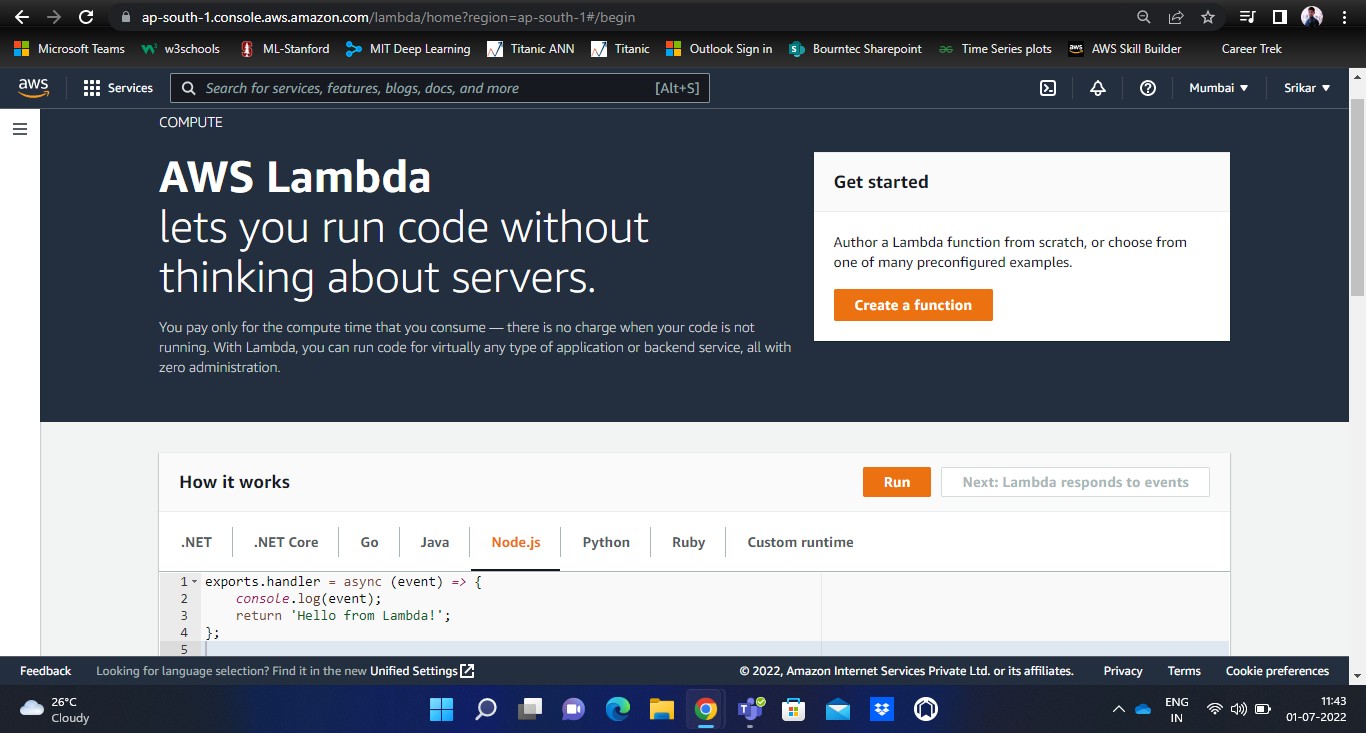


You can find various performance metrics of your Lambda here.

If you click on Applications, you can find your running applications in Lambda.

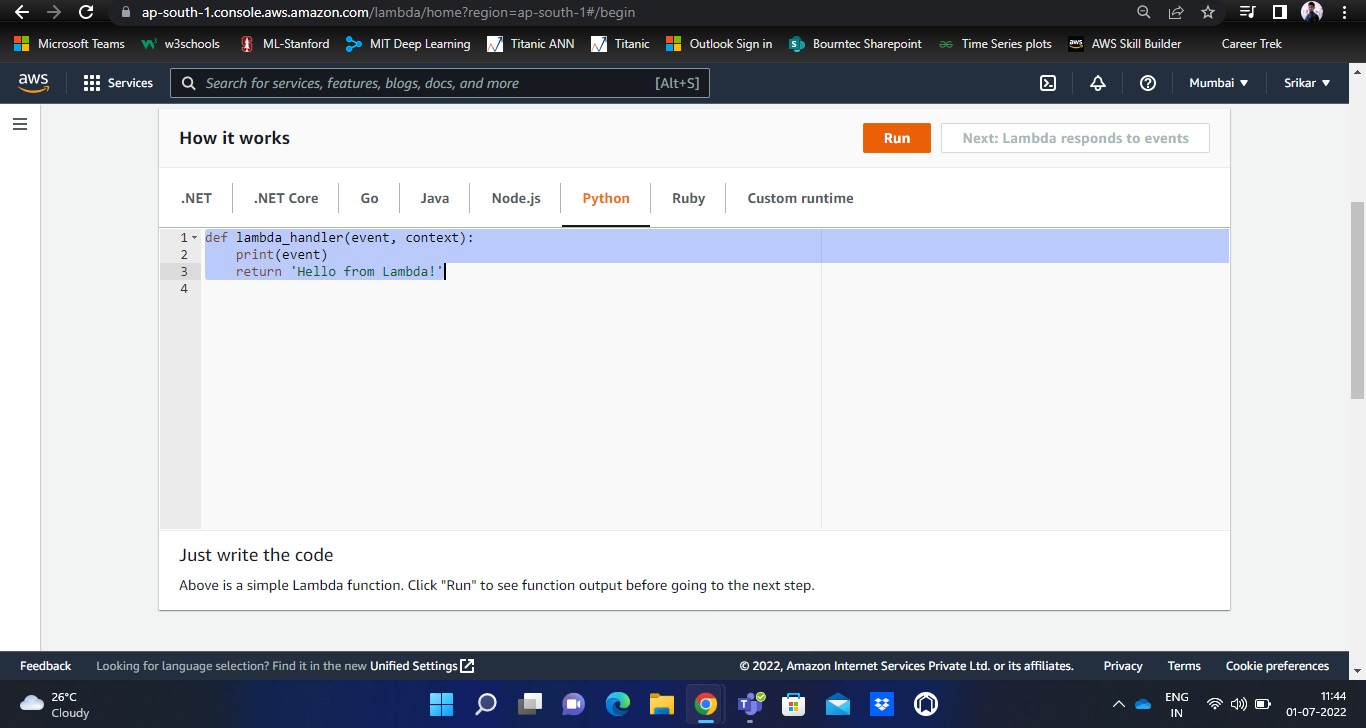
If you click on Functions, you can find your lambda functions that exist along with some details like package details, runtime details, etc.

***Step – 3*** Getting started with Lambda

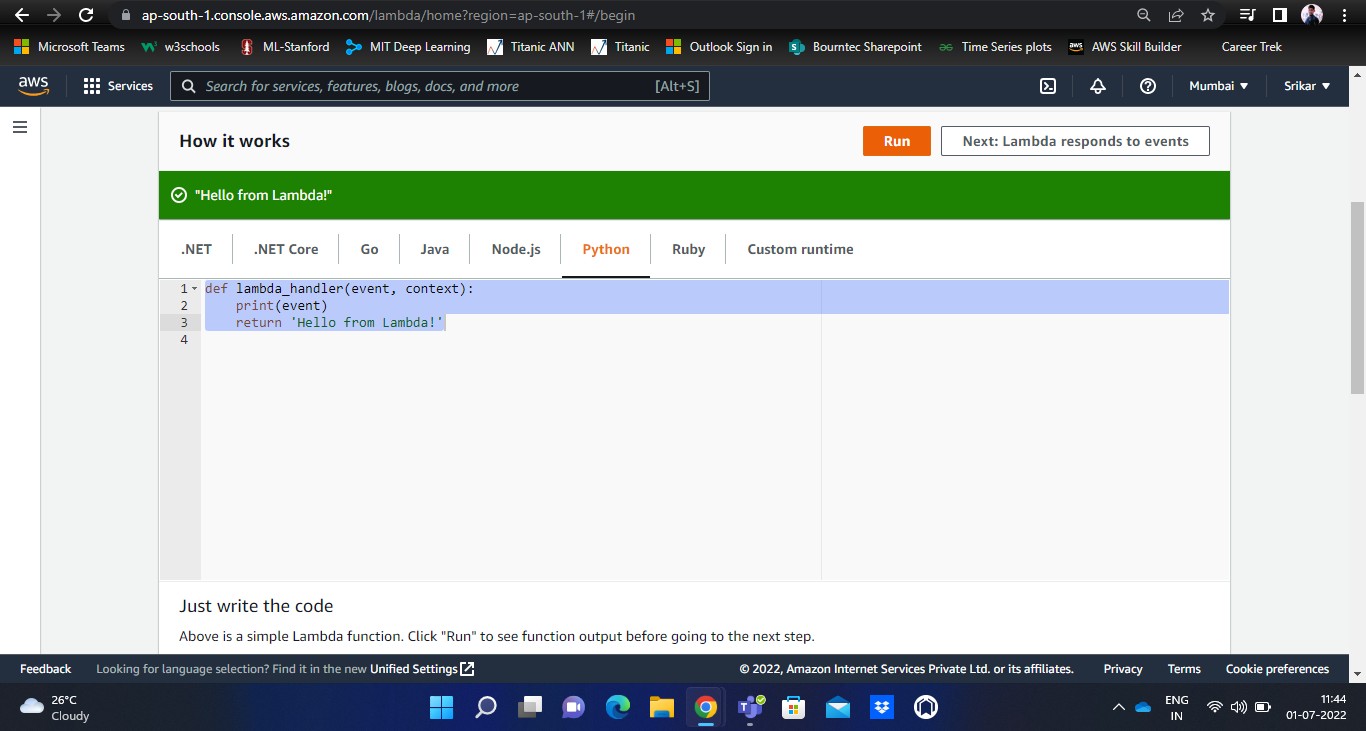


You can notice various programming languages that are supported by Lambda.

There is a sample code that can be run with any programming language. Let’s run the sample code using Python. Select the entire code, and click on Run to process the code.



After your code is successfully run, you will receive a pop-up regarding the same.

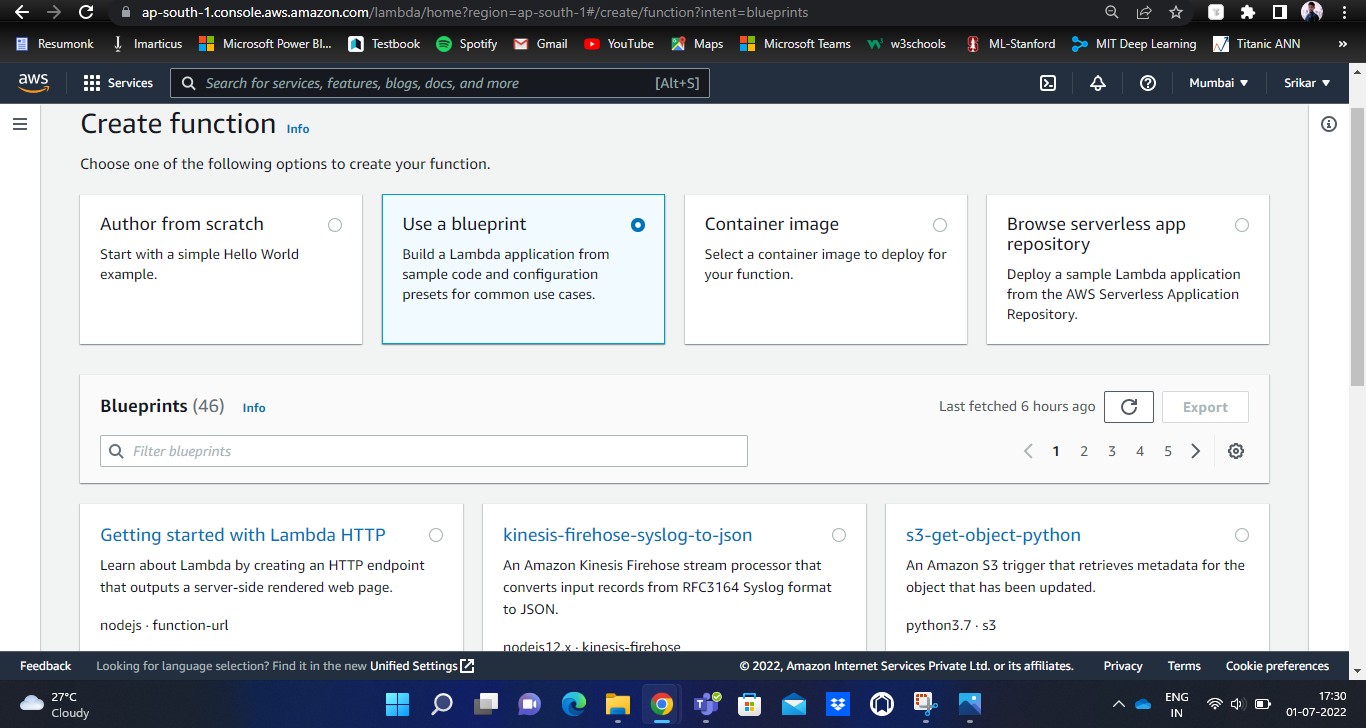


***Step – 4*** Creating a sample Lambda function

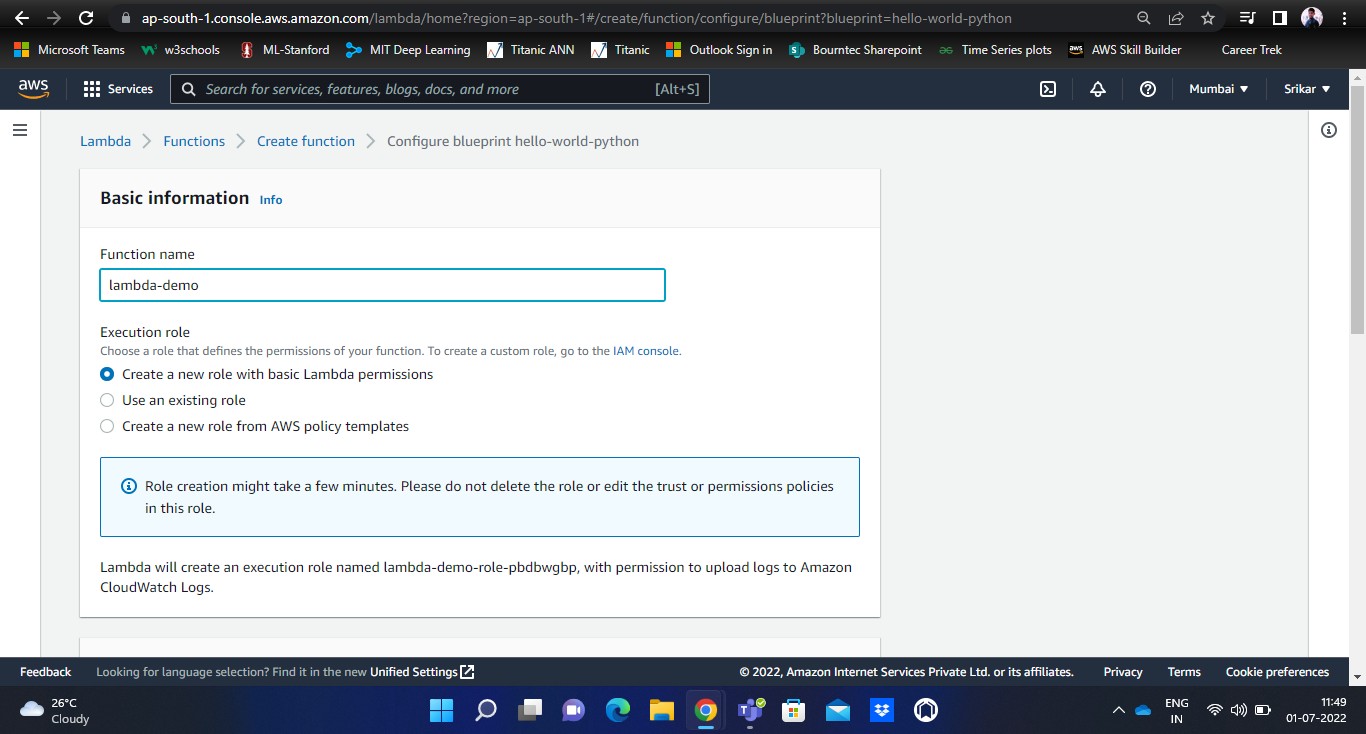
On your Lambda home page, click on Create function.

You will have four options - Author from scratch, Use a blueprint, Container image, etc.

Choose ‘Use a blueprint’. This will help us to choose an already existing blueprint of a few lambda functions.

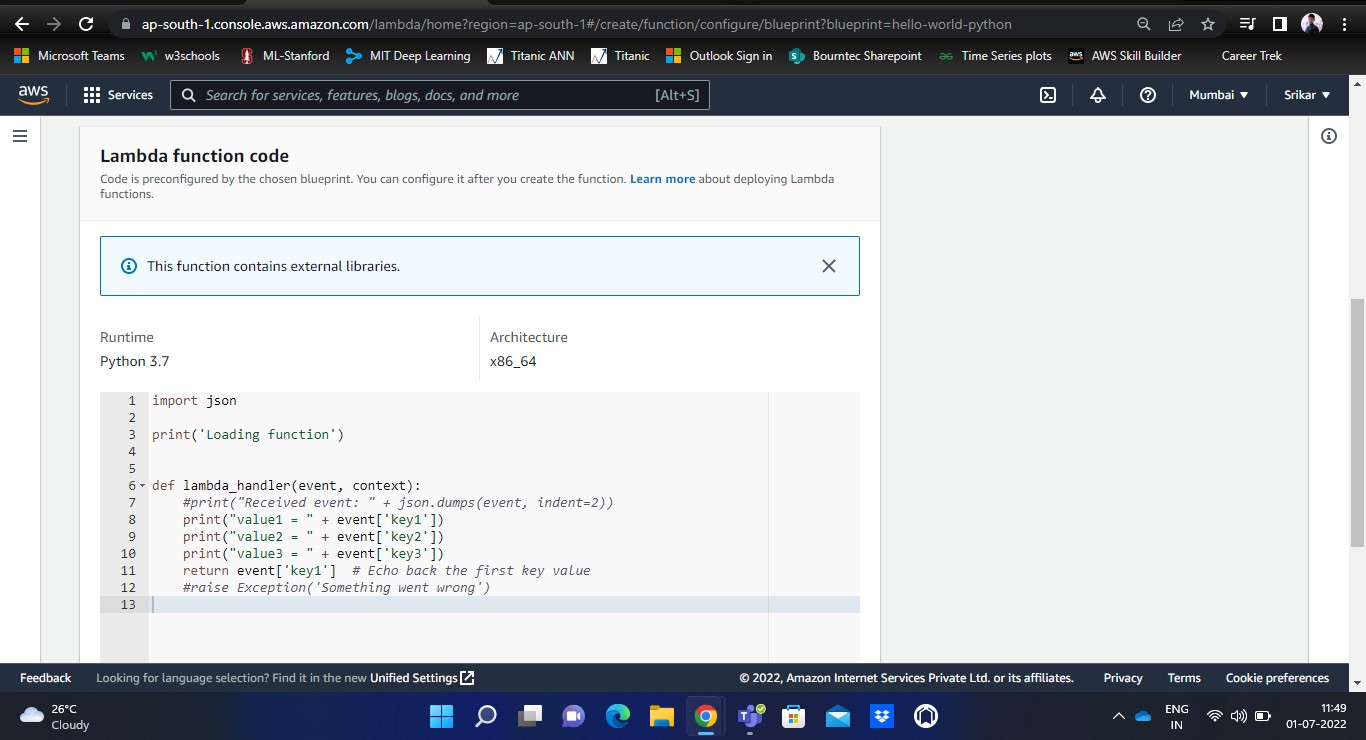


Choose any blueprint. Then, you have to provide a few details as shown below.

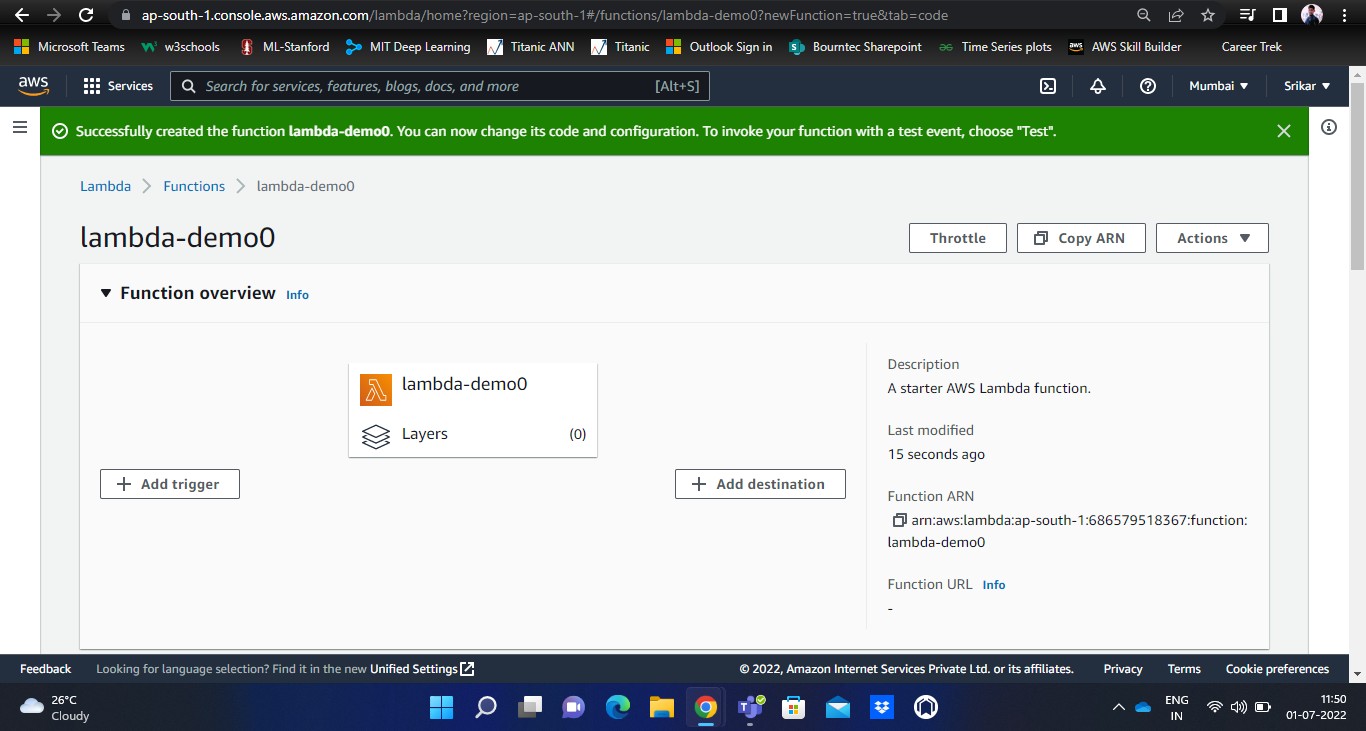


Provide the Function name, and set the other options to default.

When you scroll down, you will find the Lambda function code as shown below.



You can edit your code as per your requirement, and click on Create function option in the bottom to run your code.

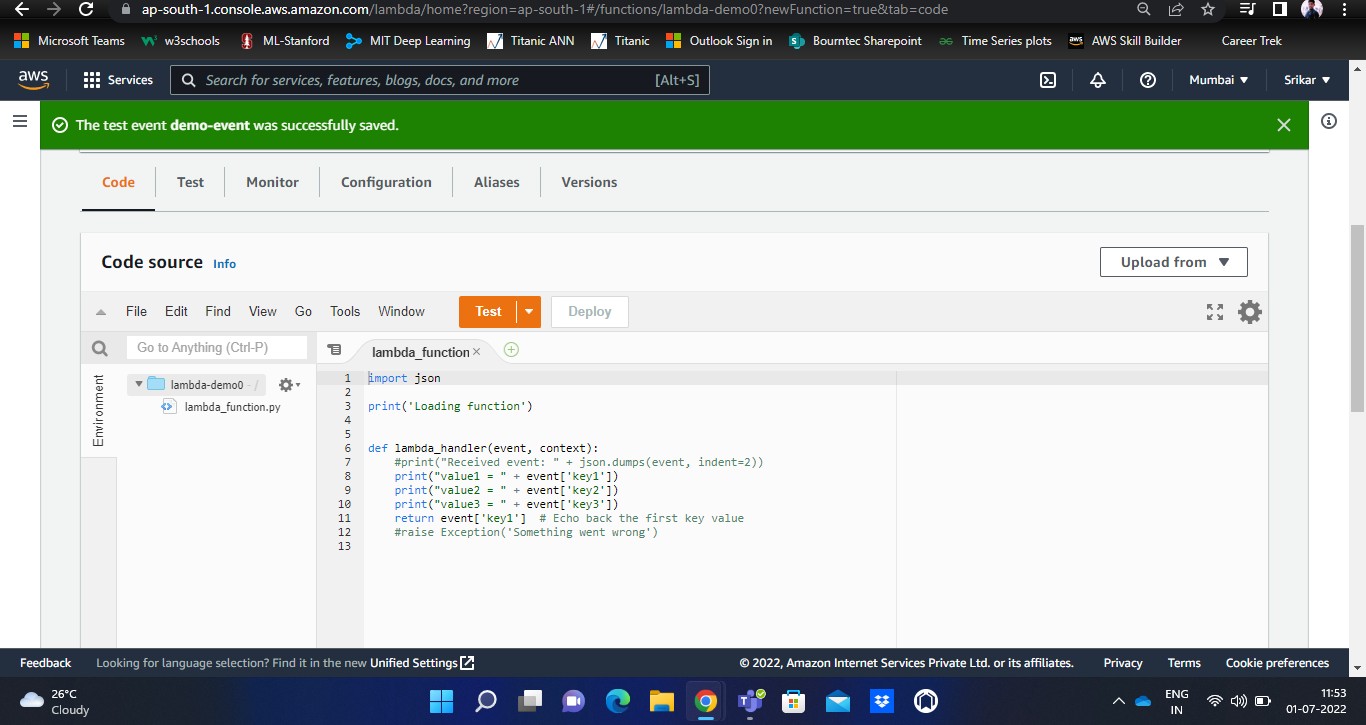


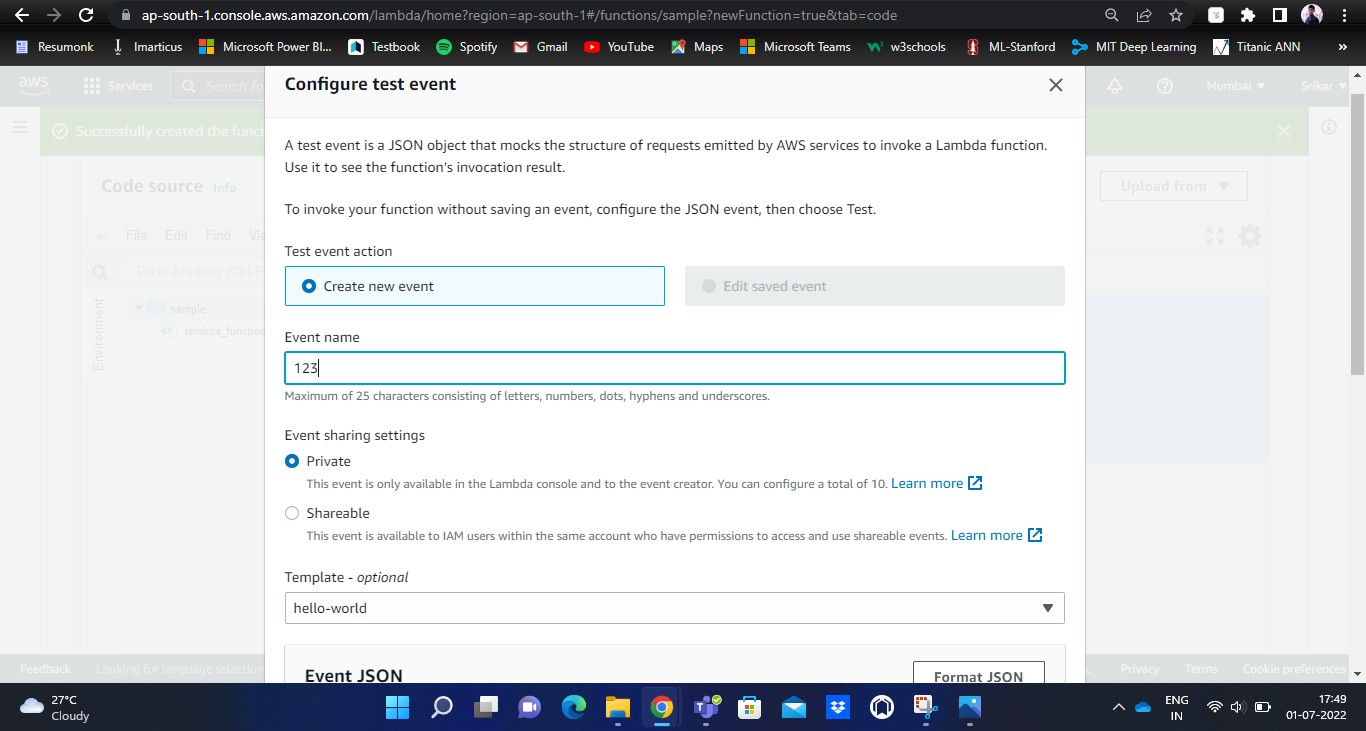
***Step – 5*** Testing & Deploying the code

Once your code is successfully run, you will be shown a source code along with two important options like – Test and Deploy.

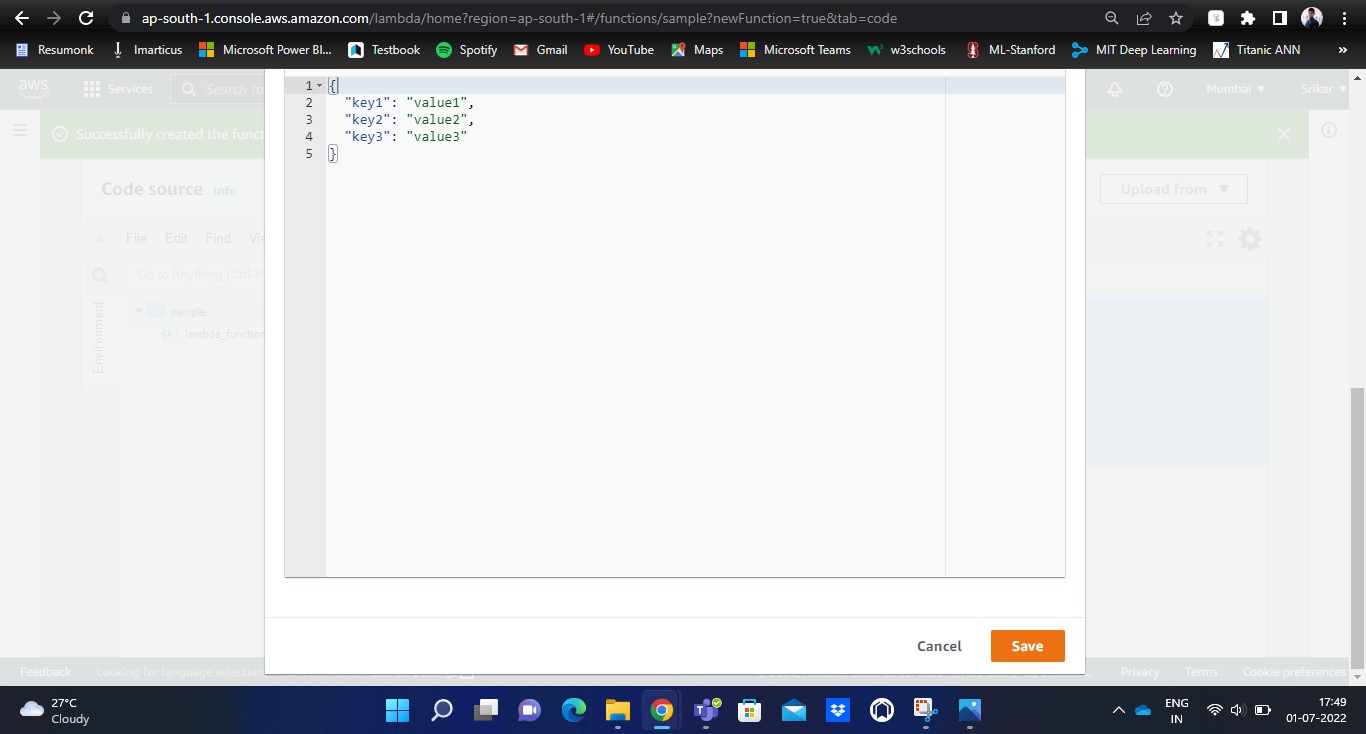
Test is used to test the code after making necessary changes to the original code.

Deploy is used to save the newly made changes to the original code, before testing it. Deploying the code is mandatory after changes without which the newly updated changes will not be saved.





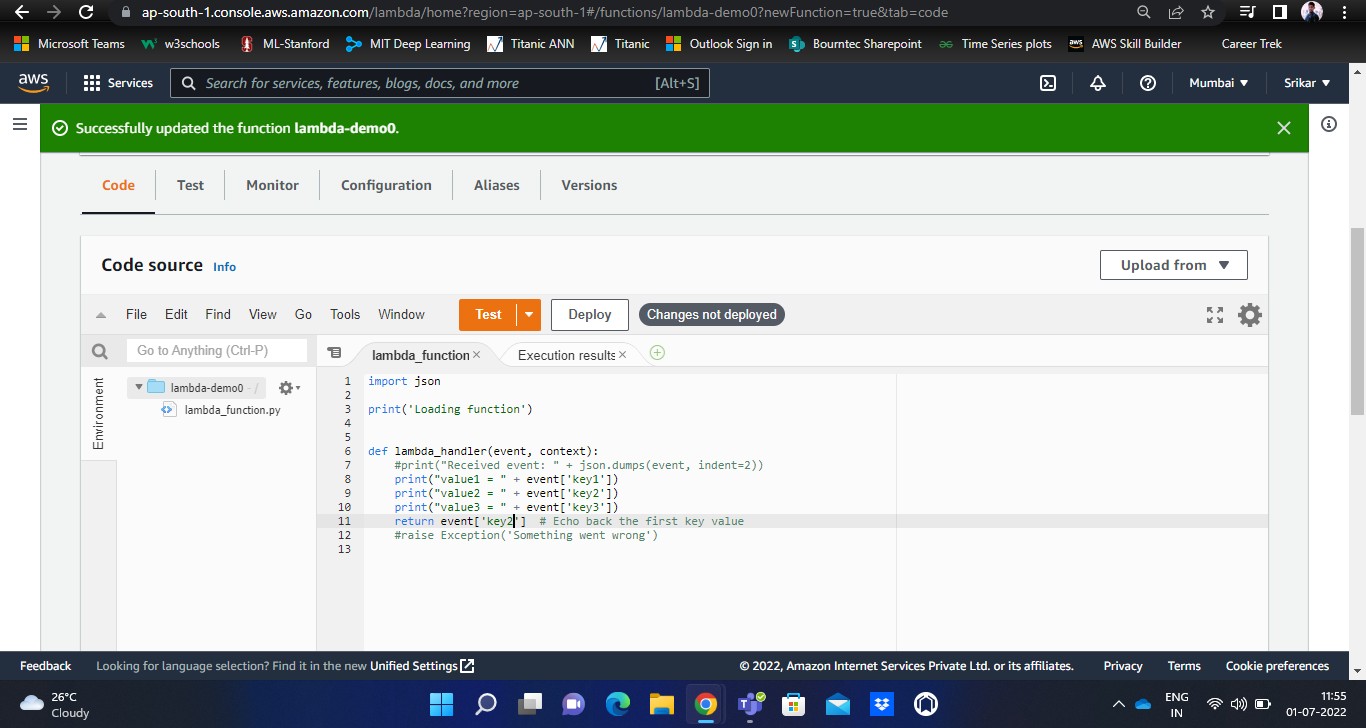
This is a sample JSON script of our source code.

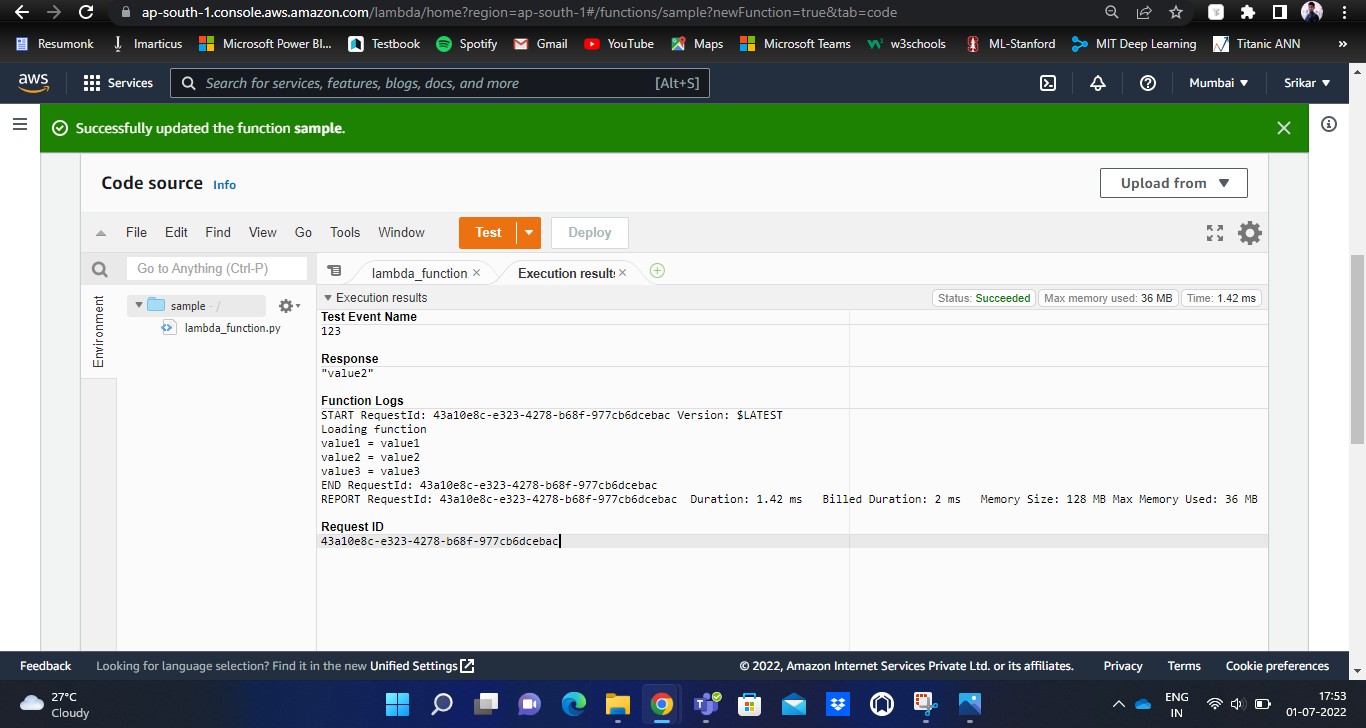


Click on Save.

***Step – 6*** Viewing the results

After making necessary changes and deploying the code, you need to test it to see the execution results.





Here, you can find the execution result.

You can also notice various parameters like the amount of time elapsed, memory used, billed duration, etc.