



Vivekanand Education Society's

Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai,, Approved by AICTE & Recognized by Govt. of Maharashtra

Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

Department of Information Technology

A.Y. 2024-25

Advance DevOps Lab

Experiment 11

Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Roll No.	43
Name	Harsh Pramod Padyal
Class	D15B
Subject	Advance DevOps Lab
LO Mapped	LO1: To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements. LO6: To engineer a composition of nano services using AWS Lambda and Step Functions with the Serverless Framework.
Grade:	

AIM : To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

THEORY :

AWS Lambda is a serverless computing service by AWS that allows you to run code without provisioning or managing servers. You create functions in supported languages like Python, Java, and Node.js, and these functions are executed in response to specific events such as API calls, file uploads to S3, or data changes in DynamoDB.

Key Features

- **Automatic Scaling:** Lambda automatically scales the infrastructure to handle incoming requests, reducing operational complexity.
- **Cost-Efficiency:** You only pay for the compute time you consume, with no upfront costs or server management fees.
- **Security:** Lambda integrates with AWS Identity and Access Management (IAM) to define roles and policies, ensuring secure execution.
- **Fault Tolerance:** AWS Lambda is designed to provide high availability and fault tolerance, handling server failures and maintaining continuous operation.

Execution Model

Lambda functions run in stateless containers fully managed by AWS. When an event triggers a function, AWS initiates a container to execute the function. If subsequent requests come in, additional containers are spun up to handle them. AWS may keep containers warm for a short period to reduce cold start latency.

Stateless Functions

Due to the stateless nature of Lambda, each function invocation is independent, running in a fresh environment. Code outside the main handler function runs once per container lifecycle, while the handler itself runs on every invocation.

Common Use Cases

- **Scalable APIs:** Lambda is ideal for building APIs that need to scale according to demand. Each API request can be routed to a specific Lambda function, and the service automatically adjusts to handle varying workloads.
- **Event-Driven Data Processing:** Lambda excels in scenarios like real-time data processing, where functions are triggered by events from sources like S3 or DynamoDB, making it suitable for tasks like data transformation, analytics, and notifications.

Packaging and Deployment

Lambda functions, along with their dependencies, are packaged and uploaded to AWS, often using an S3 bucket. AWS Lambda then uses this package to execute the function when an event occurs. Tools like the Serverless Stack Framework (SST) can simplify this process.

The image shows two screenshots of the AWS Lambda console. The top screenshot is the 'Create function' page, and the bottom screenshot is the 'Function overview' page for a function named 'harsh'.

Top Screenshot: Create function

The 'Create function' page has three options to create a 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.
- ☐ **Container image**
Select a container image to deploy for your function.

Basic information

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

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
 [Refresh](#)

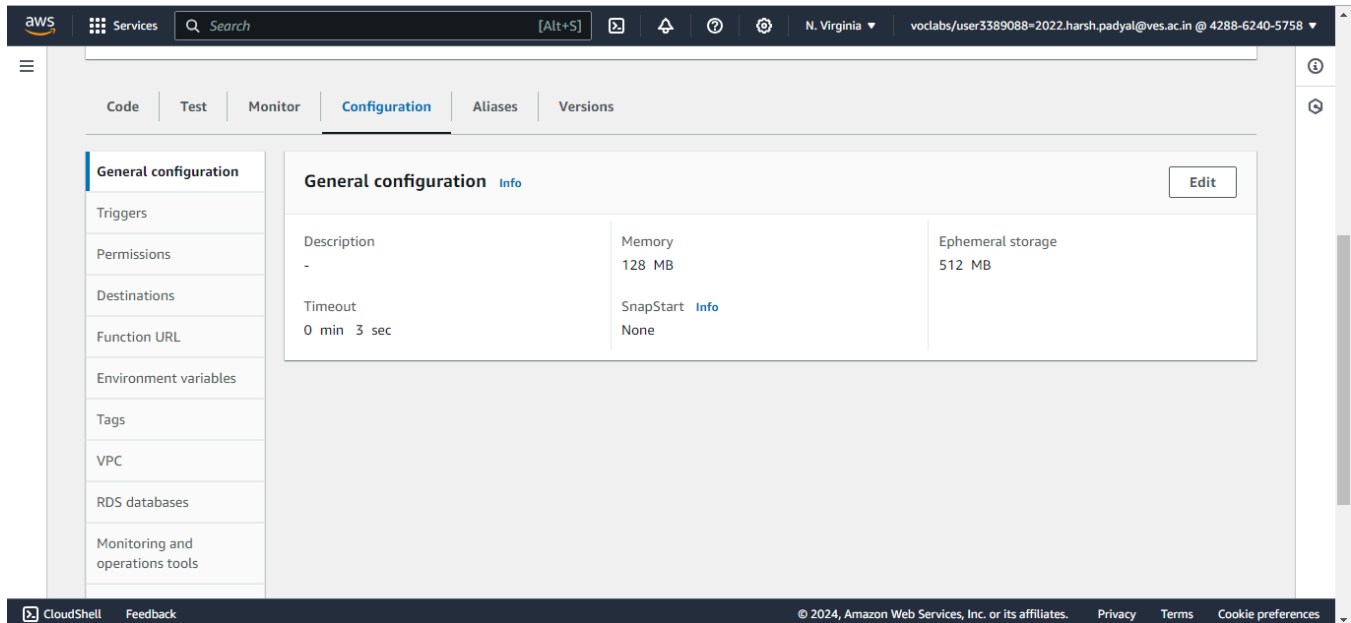
Architecture [Info](#)
Choose the architecture to use for your function.

Bottom Screenshot: Function overview

The 'Function overview' page for the function 'harsh' shows the following details:

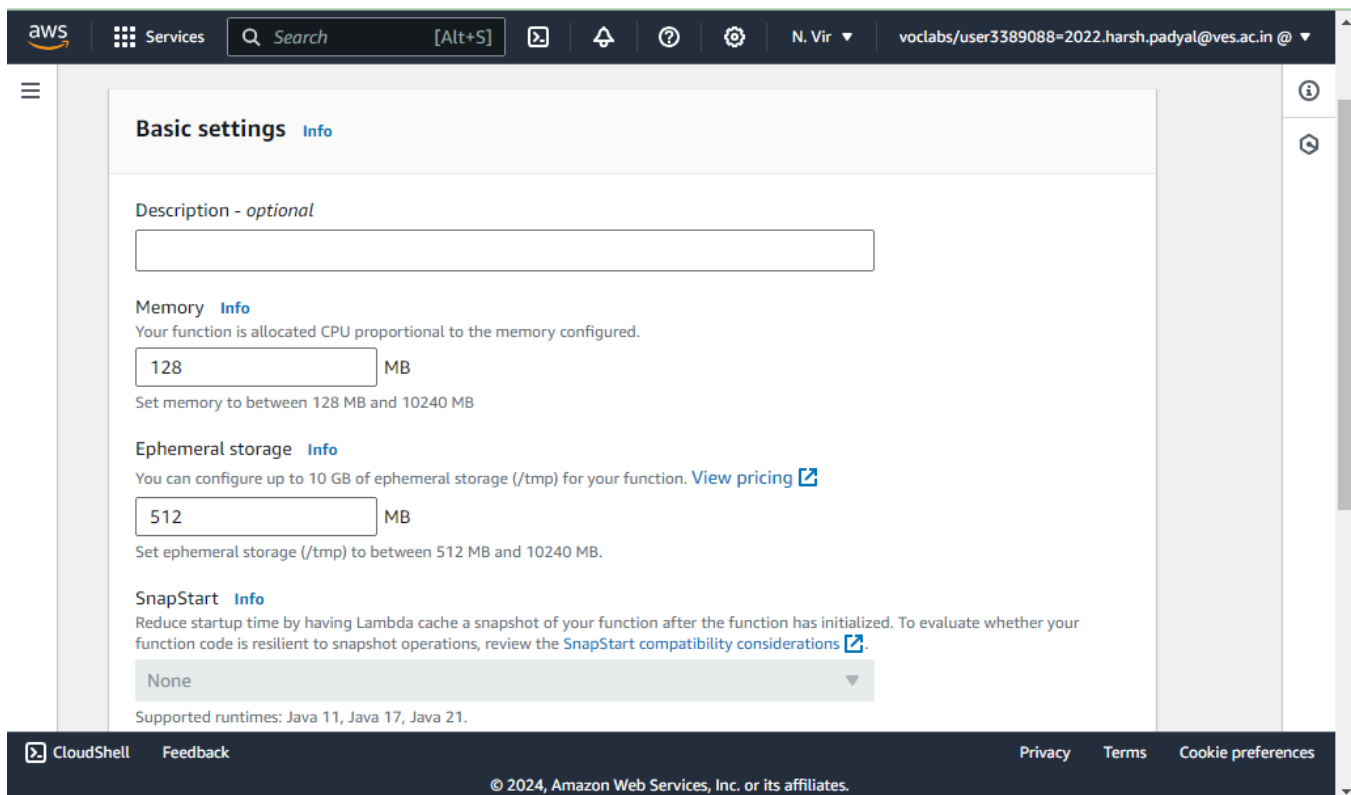
- Function overview** [Info](#)
- Diagram** [Template](#)
- Layers** (0)
- Description**
-
- Last modified**
13 seconds ago
- Function ARN**
[arn:aws:lambda:us-east-1:428862405758:function:harsh](#)
- Function URL** [Info](#)
-

Buttons at the top right of the 'Function overview' page include: Throttle, Copy ARN, Actions, Export to Application Composer, and Download.



The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and a user profile dropdown. Below this, a horizontal menu contains tabs for Code, Test, Monitor, Configuration (which is selected), Aliases, and Versions. On the left side, a sidebar lists various configuration options: General configuration, Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, VPC, RDS databases, and Monitoring and operations tools. The main content area displays the 'General configuration' details for a function. It includes fields for Description (set to '-'), Memory (128 MB), Ephemeral storage (512 MB), Timeout (0 min 3 sec), and SnapStart (None). An 'Edit' button is located in the top right corner of the configuration panel. The footer of the console shows 'CloudShell', 'Feedback', and copyright information for Amazon Web Services, Inc.

General configuration		
Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart	
0 min 3 sec	None	



This screenshot displays the 'Basic settings' tab in the AWS Lambda console. The interface is similar to the previous one, with the same top navigation and left sidebar. The 'Basic settings' section includes a 'Description - optional' text input field. Below this, the 'Memory' section shows a value of 128 MB, with a note that the function is allocated CPU proportional to the memory configured and a range of 128 MB to 10240 MB. The 'Ephemeral storage' section shows 512 MB, with a note that up to 10 GB can be configured and a range of 512 MB to 10240 MB. The 'SnapStart' section is set to 'None' and includes a link to 'SnapStart compatibility considerations'. At the bottom, it lists supported runtimes: Java 11, Java 17, and Java 21. The footer contains 'CloudShell', 'Feedback', and copyright information.

Basic settings

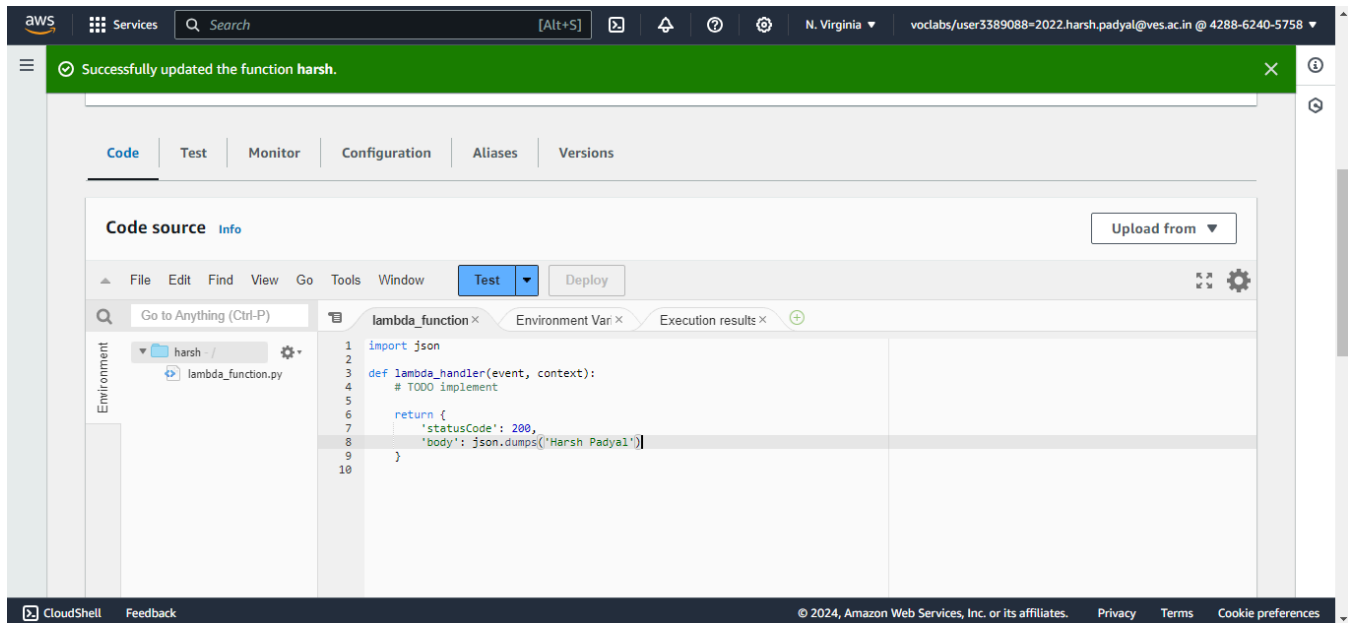
Description - optional

Memory 128 MB
Your function is allocated CPU proportional to the memory configured.
Set memory to between 128 MB and 10240 MB

Ephemeral storage 512 MB
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart None
Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the [SnapStart compatibility considerations](#).

Supported runtimes: Java 11, Java 17, Java 21.



Test event action

☒ Create new event ☐ Edit saved event

Event name

HarshPadyal

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

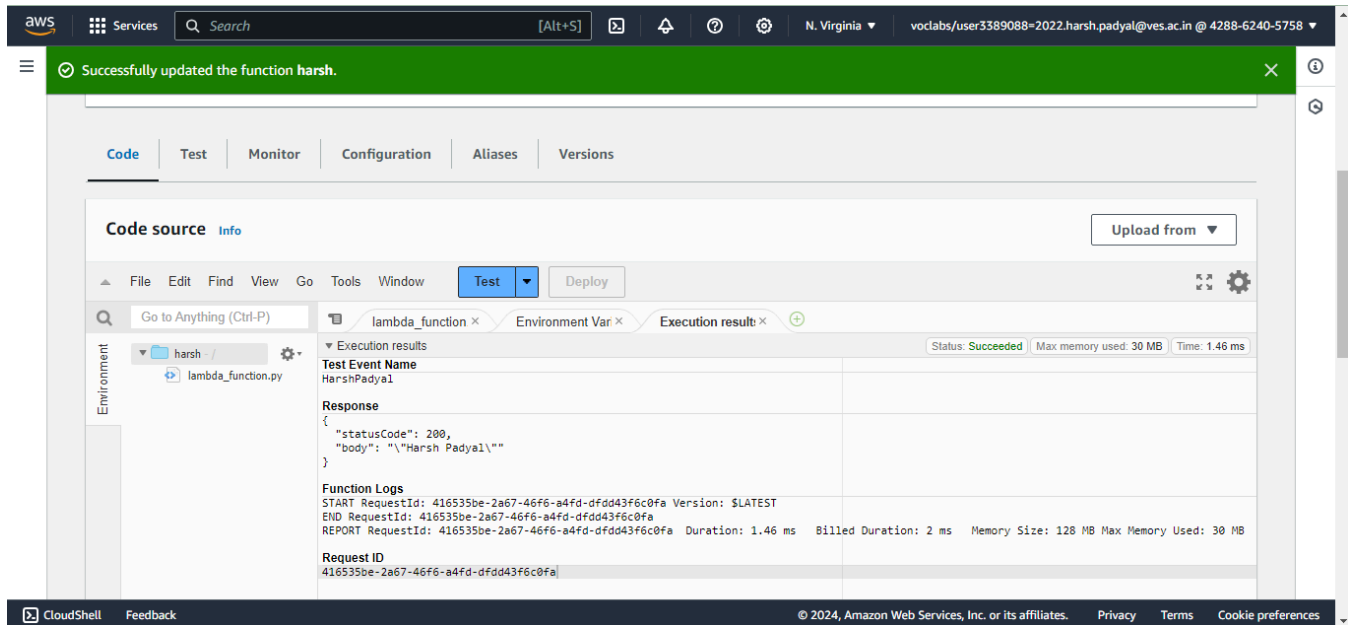
hello-world

Event JSON

Format JSON

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

Cancel Invoke Save



CONCLUSION :

AWS Lambda simplifies the process of running code in the cloud by handling server management, scaling, and security for you. Its flexibility and cost-efficiency make it an ideal choice for a wide range of applications, from scalable APIs to real-time data processing. By leveraging AWS Lambda, you can focus on writing code while AWS takes care of the rest.