AWS LAMBDA DOCUMENTAION

Naman Moolri **CEQ-525**

➢ AWS Lambda

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, and logging. With Lambda, you can run code for virtually any type of application or backend service. All you need to do is supply your code in one of the languages that Lambda support.

➢ Use Cases of AWS Lambda:

File processing: Use Amazon Simple Storage Service (Amazon S3) to trigger Lambda data processing in real time after an upload.

Stream processing: Use Lambda and Amazon Kinesis to process real-time streaming data for application activity tracking, transaction order processing, clickstream analysis, data cleansing, log filtering, indexing, social media analysis, Internet of Things (IoT) device data telemetry, and metering.

Web applications: Combine Lambda with other AWS services to build powerful web applications that automatically scale up and down and run in a highly available configuration across multiple data centers.

IoT backends: Build serverless backends using Lambda to handle web, mobile, IoT, and third-party API requests.

Mobile backends: Build backends using Lambda and Amazon API Gateway to authenticate and process API requests. Use AWS Amplify to easily integrate your backend with your iOS, Android, Web, and React Native frontends.

➢ Lambda Features:

Concurrency and scaling controls: Concurrency and scaling controls such as concurrency limits and provisioned concurrency give you fine-grained control over the scaling and responsiveness of your production applications.

Functions defined as container images: Use your preferred container image tooling, workflows, and dependencies to build, test, and deploy your Lambda functions.

Code signing: Code Signing for Lambda provides trust and integrity controls that let you verify that only unaltered code that approved developers have published is deployed in your Lambda functions.

Function blueprints: A function blueprint provides sample code that shows how to use Lambda with other AWS services or third-party applications. Blueprints include sample code and function configuration presets for Node.js and Python runtimes.

Database access: A database proxy manages a pool of database connections and relays queries from a function. This enables a function to reach high concurrency levels without exhausting database connections.

File systems access: You can configure a function to mount an Amazon Elastic File System to a local directory. With Amazon EFS, your function code can access and modify shared resources safely and at high concurrency.

➢ Creating AWS Lambda Function

To get started with Lambda, use the Lambda console to create a function:

Create a Lambda function using a blueprint. A blueprint provides sample code to do some minimal processing. Most blueprints process events from specific event sources, such as Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB, or a custom application.

Steps to create a Lambda function with the console:

1. Open the Function page of the Lambda console.
2. Choose Create function.
3. Select Author from Scratch.
4. Select Runtime for Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
5. Enter a Function name.
6. For Execution role, choose Create a new role with basic Lambda permissions. Lambda creates an execution role that grants the function permission to upload logs to Amazon CloudWatch. The Lambda function assumes the execution role when you invoke your

function and uses the execution role to create credentials for the AWS SDK and to read data from event sources.

7. In advance settings, enable Tags (Name, Owner, Purpose).

8. Click on create function.

Our function is successfully created, and we can change its code and configurations. Now, we can run our lambda code without adding any trigger. We just need to add Test Event for lambda function.

Steps to Configure Test Event:

1. Give Event name.
2. Select Event sharing settings as Private.
3. Provide Event Json. (As key and value Pair)

➢ CloudWatch

AWS Lambda automatically monitors Lambda functions on your behalf, pushing logs to Amazon CloudWatch. To help you troubleshoot failures in a function, after you set up permissions, Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs.

Log Groups:

Log groups are a standard part of CloudWatch and used to organize all logging. Any log generated by a Lambda function uses the naming convention /aws/lambda/function-name. A log group is a logical collection of log streams, which you can explore in the CloudWatch console.

The log stream contains messages from that execution environment and also any output from your Lambda function's code. Every message is timestamped, including your custom logs, which means you do not need to output timestamps. Even if your function does not log any output from your code, there are three minimal log statements generated per invocation (START, END and REPORT).

Lambda functions use an execution role to get permission to write logs to Amazon CloudWatch Logs, and to access other services and resources. If you don't already have an execution role for function development, create one.

To create an execution role

1. Open the roles page in the IAM console.
2. Choose Create role.
3. Create a role with the following properties.
    - Trusted entity – Lambda.
    - Permissions – AWS Lambda Basic Execution Role.
    - Role name – lambda-role.

The AWS Lambda Basic Execution Role policy has the permissions that the function needs to write logs to CloudWatch Logs. You can add permissions to the role later or swap it out for a different role that is specific to a single function.

Your Lambda function comes with a CloudWatch Logs log group. The function runtime sends details about each invocation to CloudWatch Logs. It relays any logs that your function outputs during invocation. If your function returns an error, Lambda formats the error and returns it to the invoker.

➢ Adding Triggers

a) API Gateway: You can create a web API with an HTTP endpoint for your Lambda function by using Amazon API Gateway. API Gateway provides tools for creating and documenting web APIs that route HTTP requests to Lambda functions. You can secure access to your API with authentication and authorization controls. Your APIs can serve traffic over the internet or can be accessible only within your VPC.

Amazon API Gateway invokes your function synchronously with an event that contains a JSON representation of the HTTP request.

API Gateway waits for a response from your function and relays the result to the caller. For a custom integration, you define an integration response and a method response to convert the output from the function to an HTTP response. For a proxy integration, the function must respond with a representation of the response in a specific format.

Choosing an API type

HTTP API – A lightweight, low-latency RESTful API.

REST API – A customizable, feature-rich RESTful API.

HTTP API features

- **Automatic deployments** – When you modify routes or integrations, changes deploy automatically to stages that have automatic deployment enabled.
- **Default stage** – You can create a default stage ($default) to serve requests at the root path of your API's URL. For named stages, you must include the stage name at the beginning of the path.
- **CORS configuration** – You can configure your API to add CORS headers to outgoing responses, instead of adding them manually in your function code.

REST API features

- **Integration types** – REST APIs support custom Lambda integrations. With a custom integration, you can send just the body of the request to the function or apply a transform template to the request body before sending it to the function.
- **Access control** – REST APIs support more options for authentication and authorization.
- **Monitoring and tracing** – REST APIs support AWS X-Ray tracing and additional logging options.

1. Open the Functions of the Lambda console.
2. Choose a function.
3. Under **Function overview**, choose **Add trigger**.
4. Select **API Gateway**.
5. Choose **Create an API** or **Use an existing API**.

a. **New API:** For **API type**, choose **HTTP API** or REST API.

b. **Existing API:** Select the API from the dropdown menu or enter the API ID (for example, r3pmxmplak).

6. For **Security**, choose **Open**.
7. Choose **Add**.

Inside Configuration option tab, check your API Gateway inside Triggers options. Now see it created endpoints.

Click on drop down details. It include some information about API Gateway such as : APT Type, Authorization, Method type, Resource Path, service principal, Stage, Statement Id.

You can use a Lambda function to monitor and analyze logs from an Amazon CloudWatch Logs log stream. CloudWatch Logs invokes your function asynchronously with an event that contains log data. The value of the data field is a Base64-encoded .gzip file archive.

b) S3 Bucket in Lambda:

You can use Lambda to process event notifications from Amazon Simple Storage Service. Amazon S3 can send an event to a Lambda function when an object is created or deleted. You configure notification settings on a bucket, and grant Amazon S3 permission to invoke a function on the function's resource-based permissions policy.

Add S3 Bucket Trigger:

Steps to Create S3 Bucket:

1. Click on Create Bucket

2, Give Bucket name(try1-s3)

3. Object Ownership (ACLs)

4. Uncheck block all public access

5. Check acknowledge option

6. Bucket versioning (disable)

7. Add tags (Owner, Purpose, Name)

8. Click on Create bucket button.

Our bucket has been created successfully. Now we can add this bucket as a trigger in your lambda function.

Add Files to Bucket:

1. Click on upload button

2. Click on add files. (Select file on your PC)

3. Click on add button. Now, file is uploaded in your S3 bucket.

Steps to Add S3 bucket to Lambda function:

1. Click on add trigger button

2. Select S3 bucket

3. Give Bucket Name

4. Select all objects and then create events

5. Check the recursive invocation button

6. Finally click on add button