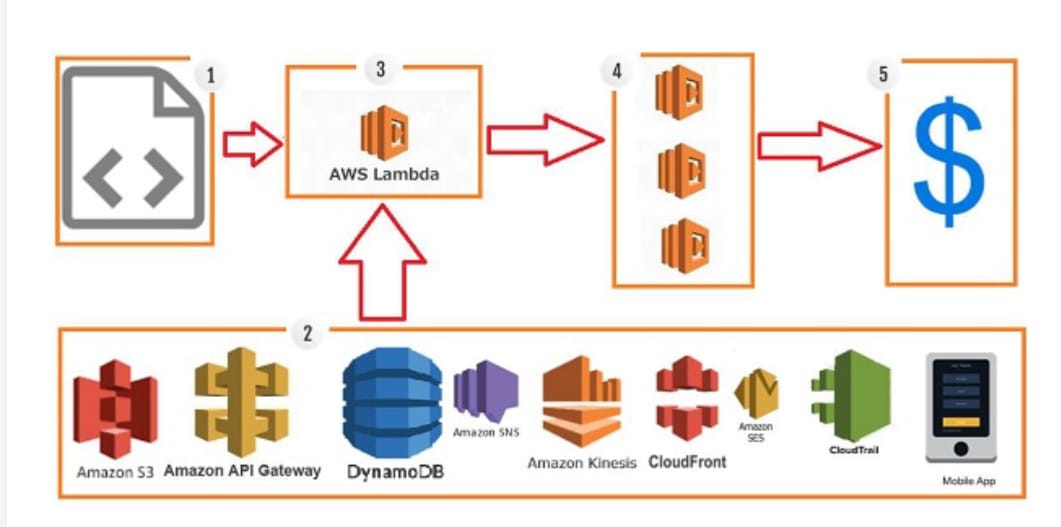
**What is AWS lambda**

* AWS Lambda is a server less compute service that executes code in response to events and automatically manages the underlying compute resources
* Introduced in November 2014, it is part of the Function as a Service (FaaS) model, where developers write functions (small, self-contained pieces of code) that are triggered by specific events, such as an HTTP request, a file upload to Amazon S3, or a database update. Lambda abstracts server management, allowing developers to focus on writing business logic rather than handling infrastructure tasks like provisioning servers, scaling, or patching operating systems
* **Key characteristics of AWS Lambda**
* Server less: No need to manage servers, operating systems, or infrastructure. AWS handles all underlying resources
* Event-Driven: Functions are executed in response to events from AWS services (e.g., S3, Dynamo DB, API Gateway) or custom sources
* Automatic Scaling: Lambda scales automatically to handle varying workloads, from a few requests per day to thousands per second
* Pay-Per-Use: You are charged only for the compute time your functions consume, billed in milliseconds, with no cost when idle.
* **How AWS Lambda works?**

The block diagram that explains the working of AWS Lambda in five easy steps is shown below −

 **Block Diagram**

**Step 1** − Upload AWS lambda code in any of languages AWS lambda supports, that is Node JS Java, Python, C# and Go.

**Step 2** − These are few AWS services on which AWS lambda can be triggered.

**Step 3** − AWS Lambda which has the upload code and the event details on which the trigger has occurred. For example, event from Amazon S3, Amazon API Gateway, Dynamo dB, Amazon SNS, Amazon Kinesis, Cloud Front, Amazon SES, Cloud-trail, mobile app etc.

**Step 4** − Executes AWS Lambda Code only when triggered by AWS services under the scenarios such as −

* User uploads files in S3 bucket
* http get/post endpoint URL is hit
* data is added/updated/deleted in dynamo dB tables
* push notification
* data streams collection
* hosting of website
* email sending
* mobile app, etc.

**Step 5** − Remember that AWS charges only when the AWS lambda code executes, and not otherwise

* **AWS Lambda Architecture**
* AWS Lambda fits into the broader server less architecture, often used with services like API Gateway, Dynamo DB, S3, and Cloud Watch to build scalable applications. Here’s how it integrates
* **Front end Integration**: Combine Lambda with Amazon API Gateway or Function URLs to handle HTTP requests for web or mobile applications. AWS Amplify simplifies Front end integration for React, iOS, or Android apps
* **Data Processing**: Use Lambda with Amazon Kinesis or SQS for real-time stream processing or with S3 for file processing (e.g., resizing images)
* **Workflow Orchestration**: Use AWS Step Functions to coordinate multiple Lambda functions for complex workflows.
* **Storage**: Store persistent data in S3, Dynamo DB, or Amazon EFS for state full operations.
* **Advantages of using AWS Lambda**

AWS Lambda offers multiple benefits when you are working on it. This section discusses them in detail −

* **Ease of working with code**

AWS Lambda gives you the infrastructure to upload your code. It takes care of maintaining the code and triggers the code whenever the required event happens. It allows you to choose the memory and the timeout required for the code

* **Log Provision**

AWS Lambda gives the details of number of times a code was executed and time taken for execution, the memory consumed etc. AWS Cloud Watch collects all the logs, which helps in understanding the execution flow and in the debugging of the code.

* **Multi Language Support**

AWS Lambda support**s** popular languages such as Node. js, Python, Java, C# and Go. These are widely used languages and any developer will find it easy to write code for AWS Lambda.

* **Disadvantages of using AWS Lambda**
* In spite of many advantages, AWS Lambda possesses the following disadvantages −
* It is not suitable for small projects.
* You need to carefully analyze your code and decide the memory and timeout. In case if your function needs more time than what is allocated, it will get terminated as per the timeout specified on it and the code will not be fully executed.
* Since AWS Lambda relies completely on AWS for the infrastructure, you cannot install anything additional software if your code demands it.
* **Events that Trigger AWS Lambda**
* The events can trigger AWS Lambda are as follows −
* Entry into a S3 object
* Insertion, updation and deletion of data in Dynamo DB table
* Push notifications from SNS
* GET/POST calls to API Gateway
* Headers modification at viewer or origin request/response in Cloud Front
* Log entries in AWS Kinesis data stream
* Log history in Cloud Trail