**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, all you need to do is supply your code in one of the language runtimes that Lambda supports.

You organize your code into Lambda functions. The Lambda service runs your function only when needed and scales automatically. You only pay for the compute time that you consume—there is no charge when your code is not running.

**When to use Lambda**

Lambda is an ideal compute service for application scenarios that need to scale up rapidly, and scale down to zero when not in demand. For example, you can use Lambda for:

**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 with your iOS, Android, Web, and React Native frontends.

When using Lambda, you are responsible only for your code. Lambda manages the compute fleet that offers a balance of memory, CPU, network, and other resources to run your code. Because Lambda manages these resources, you cannot log in to compute instances or customize the operating system on provided runtimes.

Lambda performs operational and administrative activities on your behalf, including managing capacity, monitoring, and logging your Lambda functions.

If you do need to manage your compute resources, AWS has other compute services to consider, such as:

AWS App Runner builds and deploys containerized web applications automatically, load balances traffic with encryption, scales to meet your traffic needs, and allows for the configuration of how services are accessed and communicate with other AWS applications in a private Amazon VPC.

AWS Fargate with Amazon ECS runs containers without having to provision, configure, or scale clusters of virtual machines.

Amazon EC2 lets you customize operating system, network and security settings, and the entire software stack. You are responsible for provisioning capacity, monitoring fleet health and performance, and using Availability Zones for fault tolerance.

**Key features**

The following key features help you develop Lambda applications that are scalable, secure, and easily extensible:

**Environment variables**

Use environment variables to adjust your function's behavior without updating code.

**Versions**

Manage the deployment of your functions with versions, so that, for example, a new function can be used for beta testing without affecting users of the stable production version.

**Container images**

Create a container image for a Lambda function by using an AWS provided base image or an alternative base image so that you can reuse your existing container tooling or deploy larger workloads that rely on sizable dependencies, such as machine learning.

**Layers**

Package libraries and other dependencies to reduce the size of deployment archives and makes it faster to deploy your code.

**Lambda extensions**

Augment your Lambda functions with tools for monitoring, observability, security, and governance.

**Function URLs**

Add a dedicated HTTP(S) endpoint to your Lambda function.

**Response streaming**

Configure your Lambda function URLs to stream response payloads back to clients from Node.js functions, to improve time to first byte (TTFB) performance or to return larger payloads.

**Concurrency and scaling controls**

Apply fine-grained control over the scaling and responsiveness of your production applications.

**Code signing**

Verify that only approved developers publish unaltered, trusted code in your Lambda functions

**Private networking**

Create a private network for resources such as databases, cache instances, or internal services.

**File system access**

Configure a function to mount an Amazon Elastic File System (Amazon EFS) to a local directory, so that your function code can access and modify shared resources safely and at high concurrency.

**Lambda SnapStart for Java**

Improve startup performance for Java runtimes by up to 10x at no extra cost, typically with no changes to your function code.

**Amazon Lex**

**Overview:**

Amazon Lex is a fully managed artificial intelligence (AI) service with advanced natural language models to design, build, test, and deploy voice and text conversational interfaces in applications. Lex integrates with AWS Lambda, used to easily trigger functions for execution of your back-end business logic for data retrieval and updates. Once built, your bot can be deployed directly to contact centers, chat and text platforms, and IoT devices. Lex provides rich insights and pre-built dashboards to track metrics for your.

**Amazon Lex + Generative AI**

Amazon Lex leverages the power of Generative AI and Large Language Models (LLMs) to enhance the builder and customer experience. As the demand for conversational AI continues to grow, developers are seeking ways to enhance their chatbot with human-like interactions. Large language models can be highly useful in this regard by providing automated responses to frequently asked questions, analyzing customer sentiment and intents to route calls appropriately, generating summaries of conversations to help agents, and even automatically generating emails or chat responses to common customer inquiries. This new generation of AI-powered assistants provide seamless self-service experiences that delight customers.

Amazon Lex is committed to infusing Generative AI into all parts of the builder and end-user experiences to help increase containment while resolving increasingly complex use cases with confidence. Amazon Lex has launched the below features to empower developers and users alike:

**How Lex works**

Powered by the same technology as Alexa, Amazon Lex provides you with the tools to tackle challenging deep learning problems, such as speech recognition and language understanding, through an easy-to-use fully managed service. Amazon Lex integrates with AWS Lambda which you can use to easily trigger functions for execution of your back-end business logic for data retrieval and updates. Once built, your bot can be deployed directly to chat platforms, mobile clients, and IoT devices. You can also use the reports provided to track metrics for your bot. Amazon Lex provides a scalable, secure, easy to use, end-to-end solution to build, publish and monitor your bots.

**GCP PCB/SUB**

**Exactly-once delivery**

This page explains how to receive and acknowledge messages using exactly-once semantics. Only the pull subscription type supports exactly-once delivery, including subscribers that use the StreamingPull API.

Push and export subscriptions don't support exactly-once delivery.

**Exactly-once delivery**

Pub/Sub supports exactly-once delivery, within a cloud region, based on a Pub/Sub-defined unique message ID.

When the feature is enabled, Pub/Sub provides the following:

No redelivery occurs after the message is successfully acknowledged.

No redelivery occurs while a message is outstanding. A message is considered outstanding until the acknowledgment deadline expires or the message is acknowledged.

In case of multiple valid deliveries, due to acknowledgment deadline expiration or client-initiated negative acknowledgment, only the latest acknowledgment ID can be used to acknowledge the message. Any requests with a previous acknowledgment ID fail.

**Redelivery versus duplicate**

It is important to understand the difference between expected and unexpected redeliveries.

A redelivery can happen either because of client-initiated negative acknowledgment of a message or when the client doesn't extend the acknowledgment deadline of the message before the acknowledgment deadline expires. Redeliveries are considered valid and system working as intended.

To troubleshoot redeliveries, see Dealing with duplicates and forcing retries.

A duplicate is when a message is re-sent after a successful acknowledgment or before acknowledgment deadline expiration.

A redelivered message retains the same message ID between redelivery attempts.

Subscriptions with exactly-once delivery enabled don't receive duplicate deliveries.

Exactly-once delivery support in client libraries

Supported client libraries have an interface for acknowledgment with response (example: Go). You can use this interface to check if the acknowledgment request succeeded. If the acknowledgment request succeeds, the clients are guaranteed to not receive a re-delivery. If the acknowledgment request fails, the clients can expect a re-delivery.

Clients can also use the supported client libraries without the acknowledgment interface. However, in such cases, the acknowledgment failures can lead to silent re-deliveries of messages.

Supported client libraries have interfaces for setting the minimum lease extension time (example: Go). You must set the value for the minimum lease extension to a high number to avoid any network-related acknowledgment expirations. The maximum value is set at 600 seconds.

The default values and range for the variables related to exactly-once delivery and the names of the variables might differ across client libraries. For example, in the Java client library, the following variables control exactly-once delivery.