SERVERLESS COMPARISON GUIDE

ABSTRACT

Serverless allows developers to build and run applications and services without thinking about the servers actually running the code. Serverless services, or FaaS (Functions-as-a-Service) providers, instrument this concept by allowing developers to upload the code while taking care of deploying running and scaling it.

AWS Lambda was the first one in the market to offer this kind.

Serverless can help create an environment that allows DevOps teams to focus on improving code, processes, and upgrade procedures, instead of on provisioning, scaling, and maintaining servers.

Shanmugam Karthikeyan

Upnxtblog.com

Serverless Comparison Guide

www.upnxtblog.com

Contents

Introduction

Serverless allows developers to build and run applications and services without thinking about the servers actually running the code. Serverless services, or FaaS (Functions-as-a-Service) providers, instrument this concept by allowing developers to upload the code while taking care of deploying running and scaling it.AWS Lambda was the first one in the market to offer this kind.

Serverless can help create an environment that allows DevOps teams to focus on improving code, processes, and upgrade procedures, instead of on provisioning, scaling, and maintaining servers.

Popular cloud providers that supports Function As A Service (FaaS) as follows:

- AWS via Lamdba Service
- Azure via Azure Functions
- Google via Google Cloud Functions

We will compare these cloud provider's support for Serverless in next section but first lets check how it works & its advantages/disadvantages.

How it Works

Serverless services or FaaS lets you run code without provisioning or managing servers (but still servers are needed). You pay only for the compute time you consume there is no charge when your code is not running. You can run code for virtually any type of application or backend service all with zero administration. Just upload your code and FaaS provider would take care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other services or call it directly from any web or mobile app.

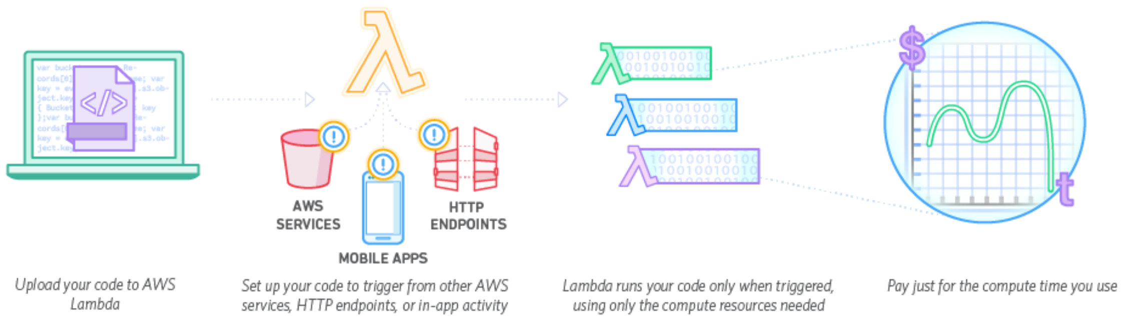
AWS SERVICES
MOBILE APPS
HTTP ENDPOINTS
Upload your code to AWS Lambda
Set up your code to trigger from other AWS services, HTTP endpoints, or in-app activity
Lambda runs your code only when triggered, using only the compute resources needed
Pay just for the compute time you use

Image - AWS Lambda / How it works

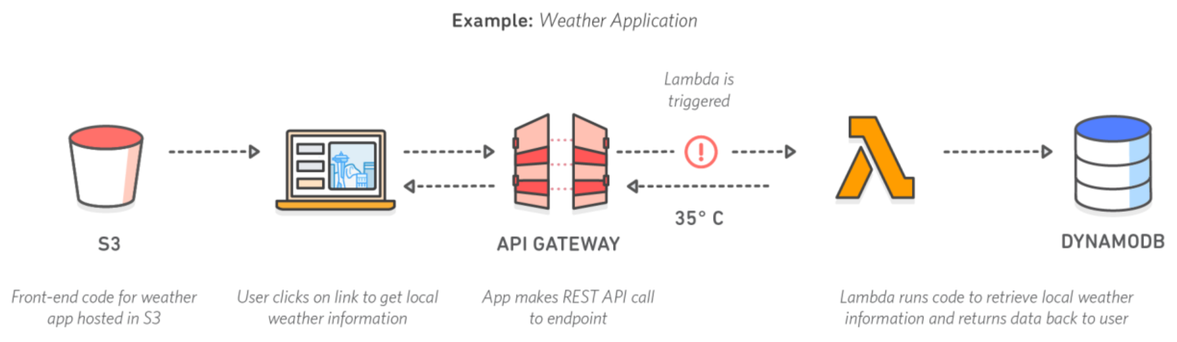
Example: Weather Application
Lambda is triggered
35° C
S3
API GATEWAY
DYNAMODB
Front-end code for weather app hosted in S3
User clicks on link to get local weather information
App makes REST API call to endpoint
Lambda runs code to retrieve local weather information and returns data back to user

Image- Another Sample

Some of the prominent use-cases are

- **Using FaaS with Cloud services as event sources** – Whenever any of the *Event sources* publish events that cause the function to be invoked.
- **On-demand Lambda function invocation over HTTPS (Amazon API Gateway)** – In addition to invoking functions using event sources, you can also invoke respective functions over HTTPS. This can be done by defining a custom REST API and endpoint using API Gateway.
- **On-demand function invocation (build your own event sources using custom apps) –** User applications such as client, mobile, or web applications can publish events and invoke Lambda functions using the Mobile SDKs, such as the AWS Mobile SDK for Android.
- **Scheduled events** – You can also set up AWS Lambda to invoke your code on a regular, scheduled basis using the AWS Lambda console. You can specify a fixed rate (number of hours, days, or weeks) or you can specify a cron expression.

Advantages

Cost

Serverless computing is more cost-effective than renting or purchasing a fixed quantity of servers, which generally involves significant periods of under utilization or idle time. It can even be more cost-efficient than provisioning an autoscaling group, due to more efficient utilization of the underlying machine resources.

Operations

Serverless architecture means that developers and operators do not need to spend time setting up and tuning autoscaling policies or systems; the cloud provider is responsible for ensuring that the capacity always meets the demand.

Productivity

Units of code exposed to the outside world are simple functions. This means that typically, the programmer does not have to worry about multithreading or directly handling HTTP requests in their code, simplifying the task of back-end software development.

Disadvantages

Performance

Infrequently-used serverless code may suffer from greater response latency than code that is continuously running on a dedicated server, virtual machine, or container. This is because, unlike with autoscaling, the cloud provider typically "spins down" the serverless code completely when not in use. This means that if the runtime (for example, the Java runtime) requires a significant amount of time to start up, it will create additional latency.

Resource limits

Serverless computing is not suited to some computing workloads, such as high-performance computing, because of the resource limits imposed by cloud providers, and also because it would likely be cheaper to bulk-provision the number of servers believed to be required at any given point in time.

Monitoring and debugging

Diagnosing performance or excessive resource usage problems with serverless code may be more difficult than with traditional server code, because although entire functions can be timed, there is typically no ability to dig into more detail by attaching profilers, debuggers or APM tools. Furthermore, the environment in which the code runs is typically not open source, so its performance characteristics cannot be precisely replicated in a local environment.

Security

Serverless is sometimes mistakenly considered as more secure than traditional architectures. While this is true to some extent because OS vulnerabilities are taken care of by the cloud provider, the total attack surface is significantly larger as there are many more components to the application compared to traditional architectures and each component is an entry point to the serverless application.

Serverless Comparison : AWS Lambda vs. Azure Functions vs. Google Functions

	AWS Lambda	Google Cloud Functions	Azure Functions
Scalability & Availability	Automatic scaling	Automatic scaling	Manual or metered scaling (App Service Plan), or sub-second automatic scaling (Consumption Plan)
Max # of functions	Unlimited functions	*1000* functions per project	Unlimited functions
Concurrent executions	1000 parallel executions per account, per region (soft limit)	No limit	No limit
Max execution time	5 mins	9 mins	5 mins
Supported languages	JavaScript, Java, C#, PHP, Go, Python and more	JavaScript Only	C#, JavaScript, F#, Python, Batch, PHP, PowerShell
Dependencies	Deployment Package per language	npm package.json	Functions written in the dedicated editor do not require dependencies
Deployment	ZIP uploads only	ZIP upload, Cloud Storage or Cloud Source Repositories	Web Editor Visual Studio Team Services, OneDrive, Local Git repository, GitHub, Bitbucket, Dropbox, External repository
Event-driven	S3, SNS, SES, DynamoDB, Kinesis, CloudWatch, Cognito, API Gateway, CodeCommit, etc.	Cloud Pub/Sub or Cloud Storage Object Change Notifications	Blob, EventHub, Generic WebHook, GitHub WebHook, Queue, Http, ServiceBus Queue, Service Bus Topic, Timer triggers
Trigger types	Wide range of Amazon services + API Gateway	Cloud Pub/Sub and Cloud Storage + HTTP trigger	Microsoft cloud services + HTTP trigger
Orchestration	AWS Step Functions	Not yet	Azure Logic Apps
Logging	CloudWatch Logs	Stackdriver Logging	App Services monitoring
Monitoring	CloudWatch & X-Ray	Stackdriver Monitoring	Application Insights
In-browser code editor	Yes	Cloud Source Repositories only	Functions environment, App Service editor
Pricing	1M free requests per month and 400,000 GB-seconds of compute time per month for free, then $0.20/1M invocations	2M requests for free, then $0.40/2M invocations	1 million requests for free, then $0.20/1M invocations, plus $0.000016/GB-s

Resources

1. Serverless Architectures
2. Curated list of awesome services, solutions and resources for serverless / nobackend applications.