AWS Serverless Architecture.

What is serverless computing?

With serverless computing, there is no virtual infrastructure for the user to manage and the user is only billed for when their code is running.

Everything from the scaling to the fault tolerance and redundancy of the underlying infrastructure is taken care of. There are servers involved, of course, it is just the user does not have to worry about any aspect of looking after them, all of that is handled by the cloud service provider.

Serverless computing services are available on all the major cloud platforms, both to individuals and businesses.

Firms from many sectors are using AWS Lambda. Other options for serverless services include Google Cloud Functions, which only supports Node.js JavaScript, Python, and Go but allows for unlimited execution time for functions. Microsoft's Azure Functions, which supports a wider range of languages including Bash, Batch, C#, F#, Java, JavaScript (Node.js), PHP, PowerShell, Python, and TypeScript, and that has similar pricing to Lambda. There's also IBM Cloud Functions, whose language support includes JavaScript (Node.js) and Swift.

AWS Lambda

AWS Lambda is a compute service that lets you run code without provisioning or managing servers.

When using AWS Lambda, you are responsible only for your code. AWS Lambda manages the compute fleet that offers a balance of memory, CPU, network, and other resources. This is in exchange for flexibility, which means you cannot log in to compute instances, or customize the operating system on provided runtimes.

- A serverless compute service.
- Lambda executes your code only when needed and scales automatically.
- Lambda functions are stateless – no affinity to the underlying infrastructure.

- You choose the amount of memory you want to allocate to your functions and AWS Lambda allocates proportional CPU power, network bandwidth, and disk I/O.
- AWS Lambda is SOC, HIPAA, PCI, ISO compliant.
- You can also provide your own custom runtime.

Natively supports the following languages:

  - Node.js
  - Python
  - Java
  - C #
  - Go
  - Ruby

Components of a Lambda Application

- **Function** – a script or program that runs in Lambda. Lambda passes invocation events to your function. The function processes an event and returns a response.
- **Runtimes** – Lambda runtimes allow functions in different languages to run in the same base execution environment. The runtime sits in-between the Lambda service and your function code, relaying invocation events, context information, and responses between the two.
- **Layers** – Lambda layers are a distribution mechanism for libraries, custom runtimes, and other function dependencies. Layers let you manage your in-development function code independently from the unchanging code and resources that it uses.
- **Event source** – an AWS service or a custom service that triggers your function and executes its logic. An **event source** is the entity that publishes events, and a Lambda function is the custom code that processes the events.
- **Downstream resources** – an AWS service that your Lambda function calls once it is triggered.
- **Log streams** – While Lambda automatically monitors your function invocations and reports metrics to CloudWatch, you can annotate your function code with custom logging statements that allow you to analyze the execution flow and performance of your Lambda function.

So now, what is the difference between Docker (ECS – Amazon EC2 Container Services) and Lambda?

EC2 Container Services is highly Scalable container management service that supports Docker containers. Allows you to run the application on managed cluster of Amazon Ec2 Instances. Also, you don’t have to worry about the need for Install, Operate and Scale your infrastructure.

Whereas AWS Lambda is a function-as-a-service, offering to run your code in response to the events. you are responsible only for your code. AWS Lambda manages the compute fleet that offers a balance of memory, CPU, network, and other resources.