Amazon - Lambda

What is Amazon Lambda?

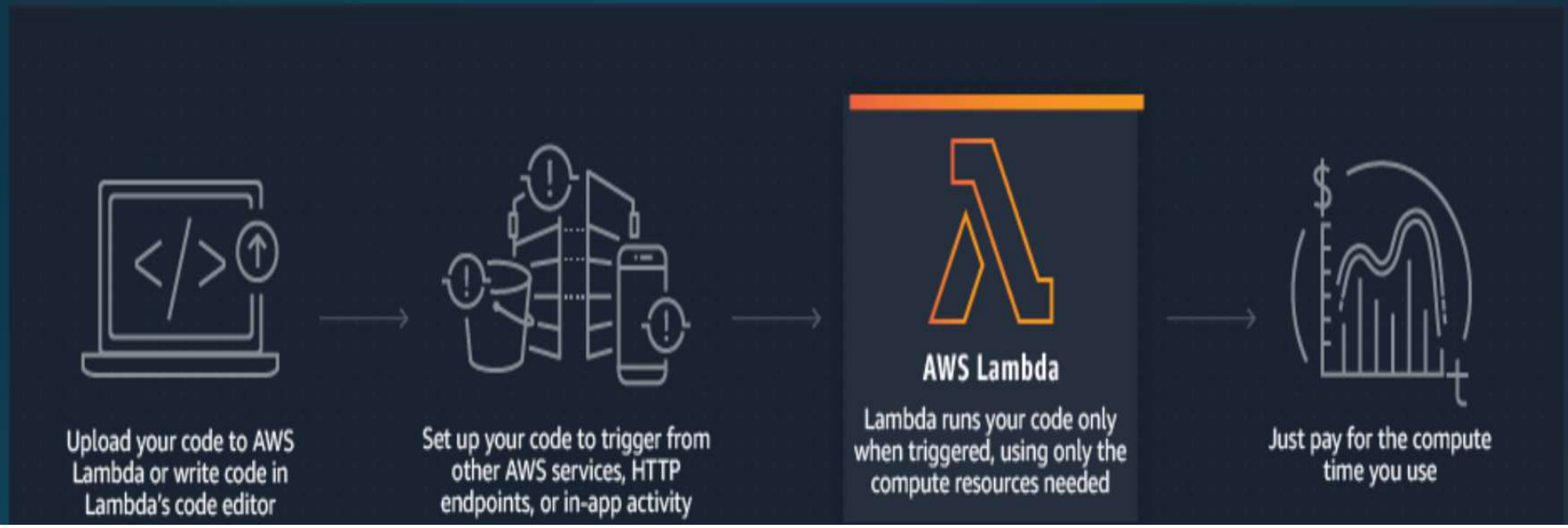
AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration.

Just upload your code and Lambda takes care of everything required to run and scale your code with high availability.

You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

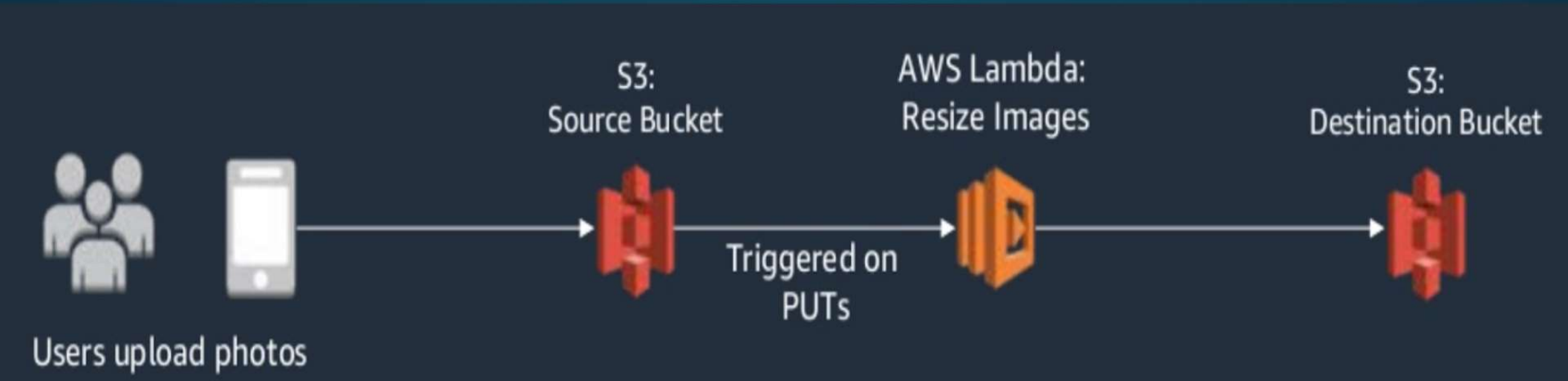
Amazon Lambda

How it works?



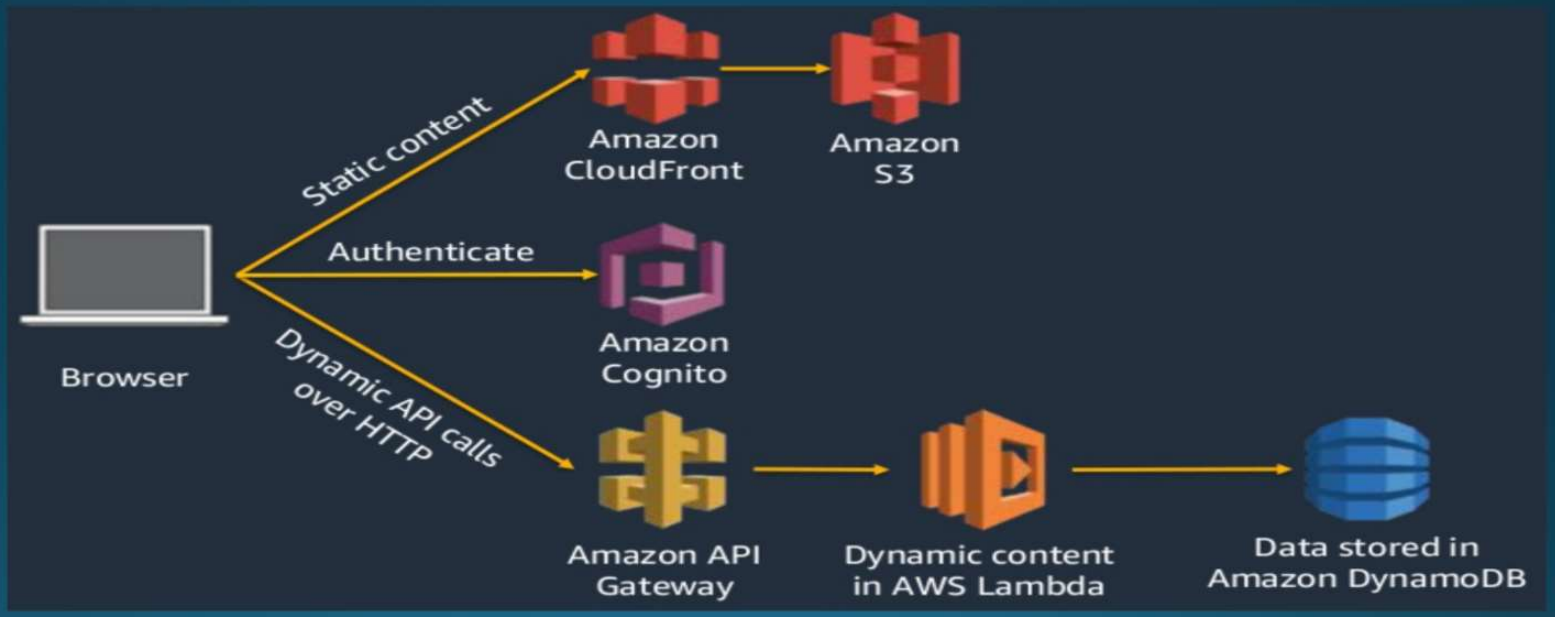
Amazon - Lambda

Image Thumbnail Creation from S3



Amazon - Lambda

Web Application



Amazon - Lambda

Benefits

NO SERVERS TO MANAGE:

AWS Lambda automatically runs your code without requiring you to provision or manage servers. Just write the code and upload it to Lambda.

CONTINUOUS SCALING:

AWS Lambda automatically scales your application by running code in response to each trigger. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload.

SUBSECOND METERING:

With AWS Lambda, you are charged for every **100ms** your code executes and the number of times your code is triggered.

Amazon - Lambda

LAMBDA FUNCTION

- The code you run on AWS Lambda is called a "**Lambda function**".
- After you create your Lambda function it is always ready to run as soon as it is triggered, like a formula in a spreadsheet.
- Each function includes your code as well as some associated configuration information, including the function name and resource requirements.
- Lambda functions are "stateless," with no affinity to the underlying infrastructure, so that Lambda can rapidly launch as many copies of the function as needed to scale to the rate of incoming events.
- After you upload your code to AWS Lambda, you can associate your function with specific AWS resources (e.g. **Amazon S3 bucket, Amazon DynamoDB table, Amazon Kinesis stream, or Amazon**

Amazon - Lambda

Features

Bring your own code:

- There are no new languages, tools, or frameworks to learn. User can use any third party library, even native ones.
- AWS Lambda supports Java, Node.js, C#, and Python code, with support for other languages coming in the future.

Flexible resource model:

- User can choose the amount of memory they want to allocate to the functions and AWS Lambda allocates proportional CPU power, network bandwidth, and disk I/O.

Amazon - Lambda

Features

Built-in fault tolerance:

- Lambda has built-in fault tolerance. AWS Lambda maintains compute capacity across multiple Availability Zones in each region to help protect the code against individual machine or data center facility failures.

- AWS Lambda is designed to provide high availability for both the service itself and for the functions it operates. There are no maintenance windows or scheduled downtimes.

Amazon - Lambda

Features

Automatic scaling:

- AWS Lambda invokes the code automatically and scales to support the rate of incoming requests without requiring to configurations.
- There is no limit to the number of requests the code can handle.
- AWS Lambda typically starts running the code within milliseconds of an event, The performance remains consistently high as the frequency of events increases.

Amazon - Lambda

Limitations

Resources Limit:

Resource	Default Limit
Ephemeral disk capacity ("/tmp" space)	512 MB
Number of processes and threads (combined total)	1,024
Maximum execution duration per request	900 seconds
Number of unique Lambda functions you can connect to each Scheduled Event	5

Amazon - Lambda

Limitations

Throttle Limit:

- The throttle limit is 1000 concurrent Lambda function executions per region.
- The throttle limit can be increased by contacting AWS support center.

Web Services

Amazon - Lambda

Limitations

Service Limit:

Items	Default Limit
Lambda function deployment package size (.zip)	50 MB
Size of code/dependencies that you can zip into a deployment package (uncompressed zip/jar size)	250 MB
Elastic Network Interface Per VPC	250
Memory Allocation	128 MB to 10,240 MB, in 1-MB increments.