AWS Lambda : Serverless Compute Service

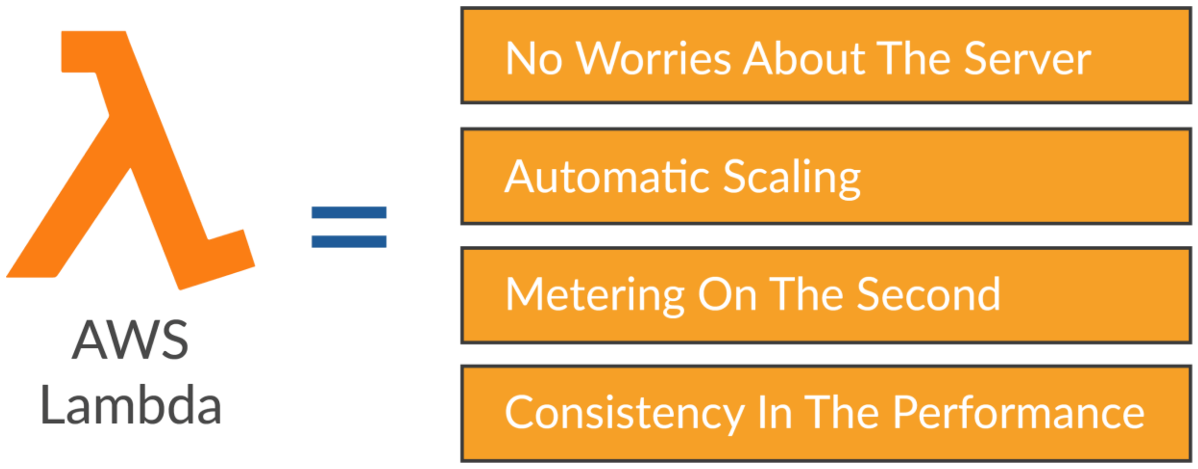
AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources, making it easy to build applications that respond quickly to new information.

AWS Lambda

AWS Lambda is one of the services that fall under the **compute domain** of services that AWS provides. Lambda allows us to execute code or any type of obligation. With lambda, we can run code virtually for any type of obligation or back-end services. All we need to do is supply our code in one of the languages that AWS Lambda supports. The Lambda code called **function** is written in any language of your choice **Node.js, Python, Go, Java, etc** and you can also use serverless and container tools (e.g AWS SAM) for building, testing, and deploying functions.

Features

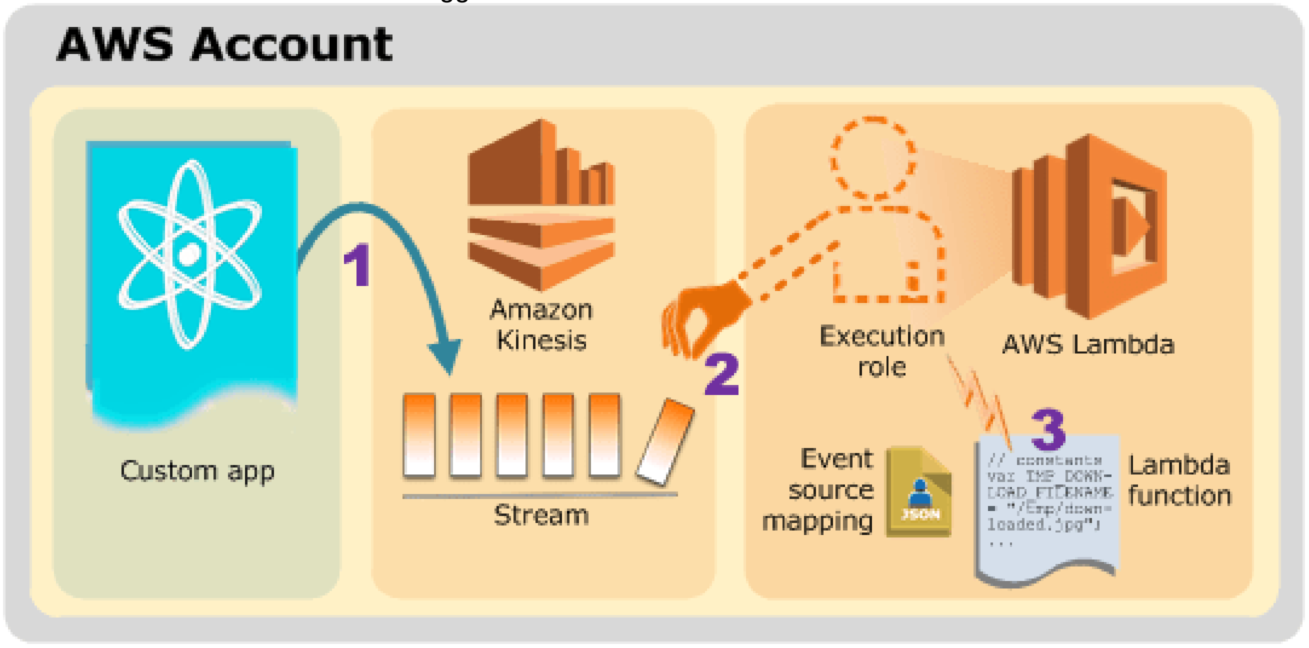
- **No worries about the server:** Lambda automatically runs our code without requiring us to provision or manage the servers. All you need to is write the code or upload it to Lambda as a **ZIP file** or **container image** and Lambda will take care of the rest. So stop worrying about provisioning and managing servers. The only thing Lambda expects is a code that is working.
- **Automatic Scaling:** Scaling of applications in AWS lambda automatically takes place in response to a trigger. Our code runs in parallel and processes each trigger individually and the scaling depends on the size of the workload.
- **Metering on the second:** Billing in Lambda is based on **milliseconds**. We only pay for the amount of time for which our code is running which means that we are not charged for any of the servers. The only payment required is for the amount of time the code is computed. You don’t have to pay anything when the code is not running.
- **Consistency in performance:** With AWS Lambda the consistency in performance is not compromised at all as we can enhance the effectiveness of the code execution by selecting the right memory size for our function.



.

Components of a Lambda Application

- **Function:** The piece of code you upload on Lamda is called a function.
- **Runtimes:** Runtimes basically allow the functions written in different languages to be executed in the same base environment
- **Layers:** Layers allow you to manage the in-development function code independently from the unchanging code and the resources that it uses.
- **Event Source:** An event is an AWS service or any custom service that triggers your function.
- **Downstream Resources:** Downstream Resource is an AWS service that is called by your Lambda function once it is triggered.

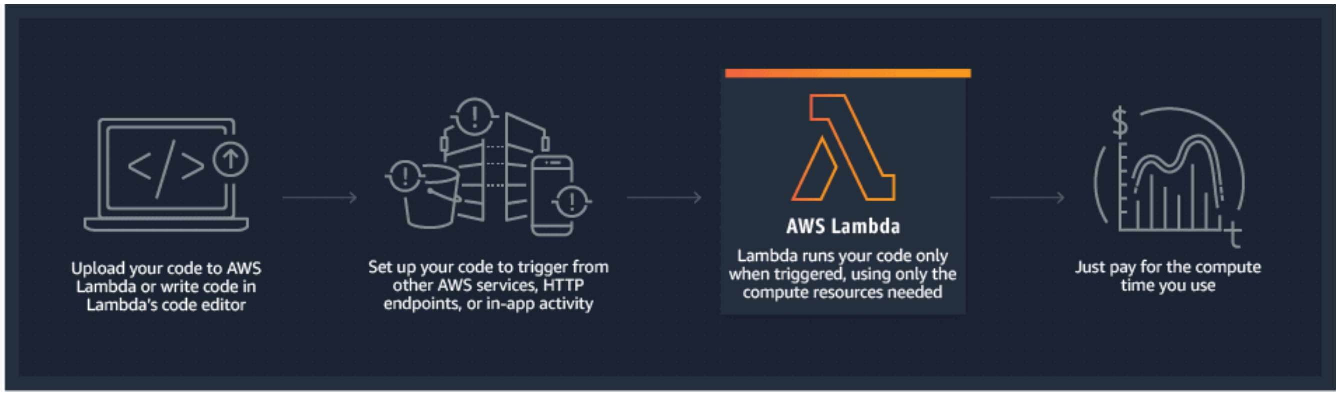


```
// constants
var IMG_DOWN-
LOAD_FILENAME
= "/tmp/down-
loaded.jpg";
...
```

How Lambda Works

Now let’s get to know how **AWS Lambda works** or in other words, let’s look at how complicated functions behind the scenes work in a simple and seamless way
The clients send data to AWS Lambda and the client could be anyone who’s sending requests to AWS Lambda. It could be an application or other AWS services Lambda receives the request and

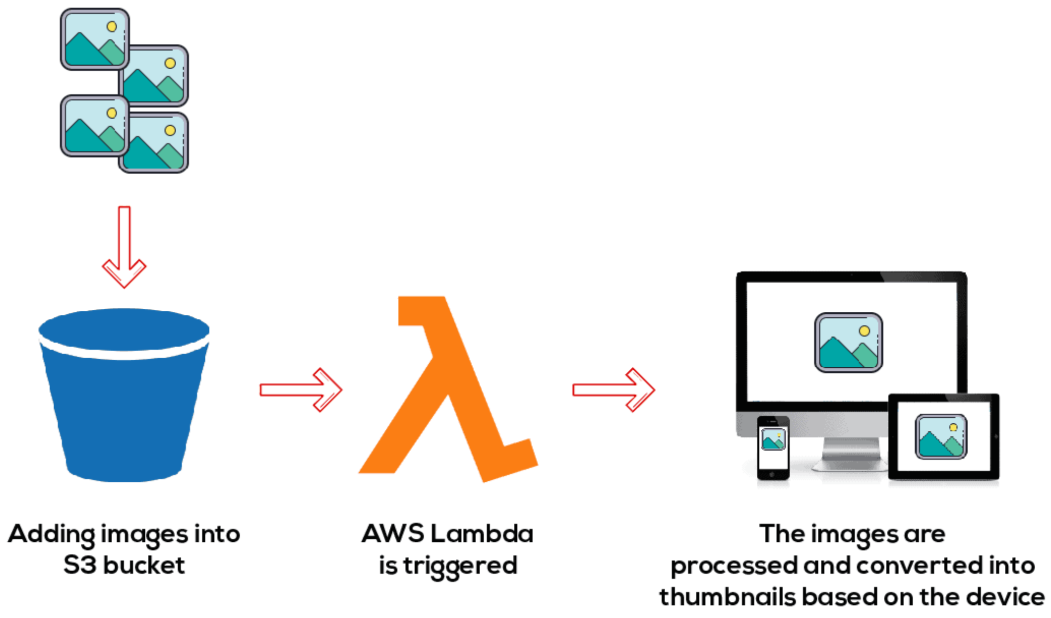
depending on the size of data or the volume of the data, it runs on a defined number of containers. The requests are given to the container to handle and the container which contains the code the user has provided to satisfy the query would run. With an increasing number of requests, an increasing number of containers are created, and when the requests reduce the number of containers reduces as well which also helps in saving cost.


Upload your code to AWS Lambda or write code in Lambda's code editor
Set up your code to trigger from other AWS services, HTTP endpoints, or in-app activity
AWS Lambda
Lambda runs your code only when triggered, using only the compute resources needed
Just pay for the compute time you use

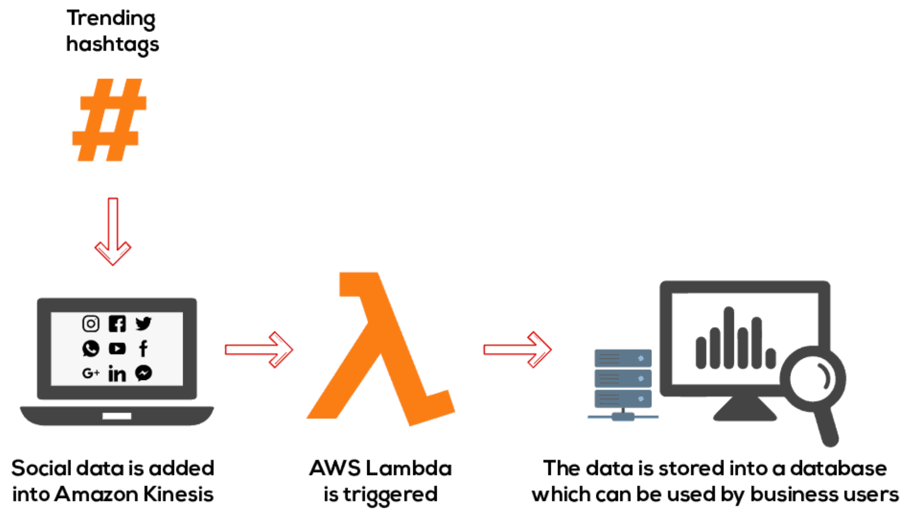
Use Cases

There are a huge number of ways that AWS Lambda is specifically used in the business world and some of them are listed below.

- **AWS Lambda is used to process an image when it's uploaded**
  Let's say the object gets uploaded in the S3 bucket in a raw format or in the format which we don't expect to get uploaded. AWS Lambda is triggered anytime a new object is added to the bucket and the images are processed and converted into thumbnails based on the devices that would be reading the data. It can be a PC, mobiles, tablets, etc. So based on the different device formats Lambda can get triggered and convert the video or image into different formats as per requirement.

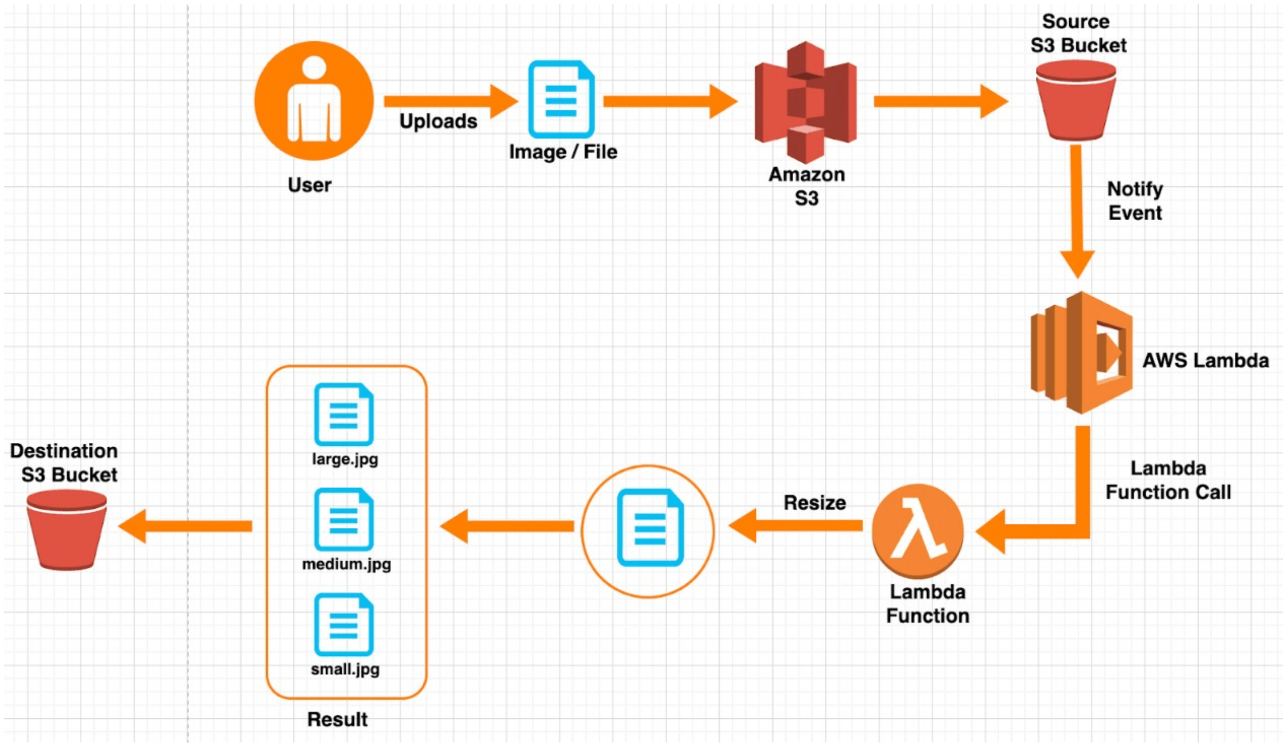

Adding images into S3 bucket
AWS Lambda is triggered
The images are processed and converted into thumbnails based on the device

- **AWS Lambda is used to analyze social media data**
  Let's say we are collecting the hashtag trending data and the data is received and is added to the Kinesis stream to feed into the Amazon environment and Lamda actions get triggered. It receives the data and the data is stored in the database which can be stored by businesses for later analysis.


Trending hashtags
Social data is added into Amazon Kinesis
AWS Lambda is triggered
The data is stored into a database which can be used by business users

- **Backing Up Data**
  Consider a situation where you have to set up a temporary storage system to back up data as soon as it is uploaded as real-time manual backups are not possible and efficient too, looking

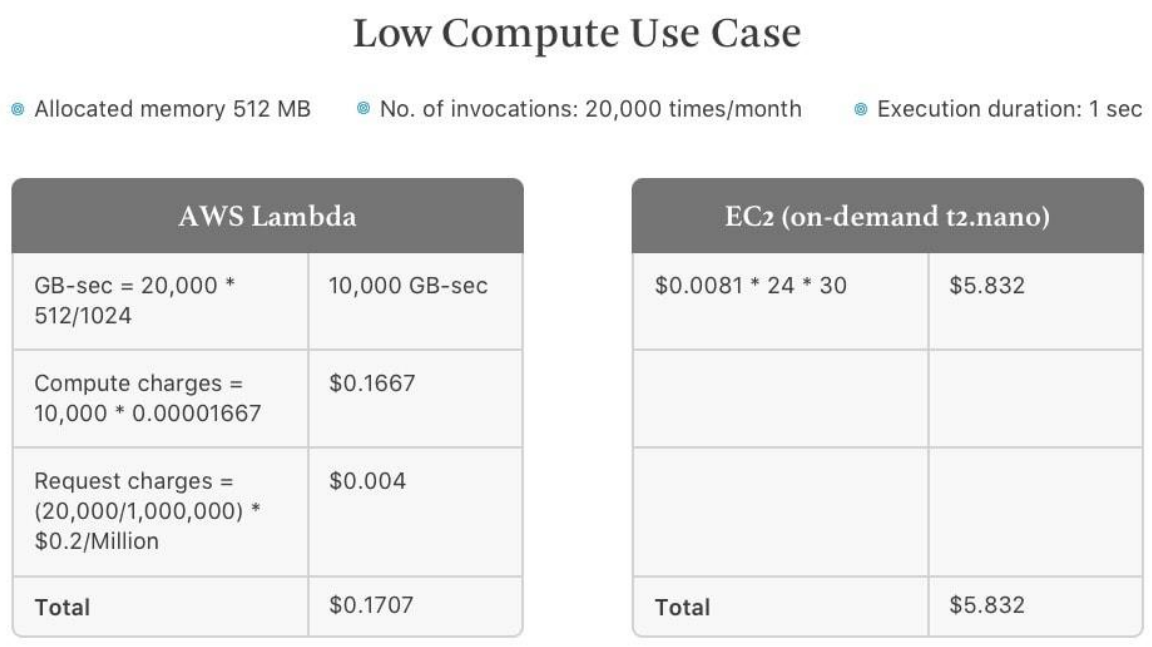
at the size of the data and the random times it will be put in the bucket. AWS Lambda comes to the rescue. This is what can be done to achieve this.



1) Create two buckets i.e. source and destination bucket. One for uploading and the other for backing up the data
2) Create IAM role and policies
3) Create a Lambda function to copy files between buckets. The Lambda function is triggered every time there is an upload into the bucket. This data is then uploaded to the backup bucket.
4) Testing out

AWS Lambda Pricing

With AWS Lambda you only pay for what you use. You will be charged as per the number of **requests** for your **functions** and the amount of **time** for the execution of your code.
The **request** is counted, each time the response to an event is executed by Lambda. Similarly, the **amount of time** taken is calculated from the time your code begins executing till it returns a response or gets terminated rounded up to the nearest **1ms,** and also how much memory size you assign to your function. For the free usage tier, **1M free requests per month and 400,000 GB-seconds of compute time per month.**



Low Compute Use Case

- Allocated memory 512 MB
- No. of invocations: 20,000 times/month
- Execution duration: 1 sec

AWS Lambda	
GB-sec = 20,000 * 512/1024	10,000 GB-sec
Compute charges = 10,000 * 0.00001667	$0.1667
Request charges = (20,000/1,000,000) * $0.2/Million	$0.004
Total	$0.1707

EC2 (on-demand t2.nano)	
$0.0081 * 24 * 30	$5.832
Total	$5.832

Limitations of AWS Lambda

- You have a limited disk space of **512 MB**
- By default, the deployment package size is **50 MB**
- Memory ranges from **128 to 3008 MB**
- The maximum execution time for a function is **15 minutes**
- Body payload size for request and response is up to **6 MB**
- Event request body can be up to **128 KB**