AWS LAMBDA

AWS Lambda is a powerful service that lets you run code without provisioning or managing servers. A few features that I loved the most are Auto Scaling, Pay-as-you-go, Event-driven Execution and Integrated Monitoring.

AWS Lambda is serverless, event-driven compute service that lets us run code for virtually any type of application or backend services with provision or managing the servers. The beauty of Lambda is it's the simple of run the code in the cloud as it is abstract away from everything except for function interface which you get to fill in the code you want to run.

AWS Lambda is very cost effective if you used it in a right way. It is most useful for small snippet of code that rarely change and one of the most effective methods in the Lambda can be used to consider it as a plugin system for other AWS services.

Its support event driven applications triggered by events such as HTTP request request's Dynamodb table updates or state transactions. You simply upload you code and Lambda handles everything from provisioning to scaling and maintenance. It will automatically scale applications based on traffic, handling server management auto scaling, security patching and monitoring.

AWS Lambda is ideal for developers who want to focus on writing code without worrying about infrastructure management.

AWS Lambda Components

In the case where you have to multiple simultaneous events Lambda simply spin up multiple copies of the functions to handle the events. In other words Lambda can be describe as a type of function as service (Faas).

There are three components comprise AWS Lambda.

- **A Function:** This is the actual code that performs the task.
- **A configuration:** This specify how your function is executed**.**
- **An event source:** This is an event that triggers the function. You can trigger with several AWS services or a third-party service**.**

Key Concepts of Lambda Functions.

Lambda function: Key concepts	
Code package	• Essentially your actual code, with all assets and binaries needed for that code to run within the runtime environment configured. • The maximum size is 50MB compressed or 250MB extracted. • You can specify an existing code package in S3 bucket or upload the code package directly when you create the function. Lambda will then store your code package in an S3 bucket managed by the service. Either way, AWS Lambda pulls the code from the S3 bucket every time the function is invoked.
Handler	• The starting point from the time AWS Lambda is invoked. • The handler is the specific code included in the code package (e.g., a Python or Node.JS function). • If the handler is successfully invoked, the code can do what it wants with the runtime environment.
Event object	• One of the parameters provided to the handler when it is invoked. The information in the object is needed by the function to perform the logic it was created to perform. • Depending on the event source, this object info varies; e.g., an S3 bucket event source will have all information about the bucket itself.

Context object	• Allows your function code to interact with the Lambda execution environment.
	• Regardless of your runtime environment, there are three key pieces of data available to context objects:
	  • **AWS RequestID** to track specific invocations of a Lambda function
	  • **Remaining time used** to tell the milliseconds remaining before your function timeout occurs
	  • **Logging** to stream log statements to Amazon CloudWatch

Advantages of AWS Lambda

1. Serverless Architecture: No need to manage servers.
2. Cost-Effective: Pay only for execution time.
3. Automatic Scaling: Seamlessly scales with the workload.
4. Event-Driven Execution: Efficient handling of events from multiple sources.
5. Faster Deployment: Focus on code, not infrastructure.

Challenges of AWS Lambda

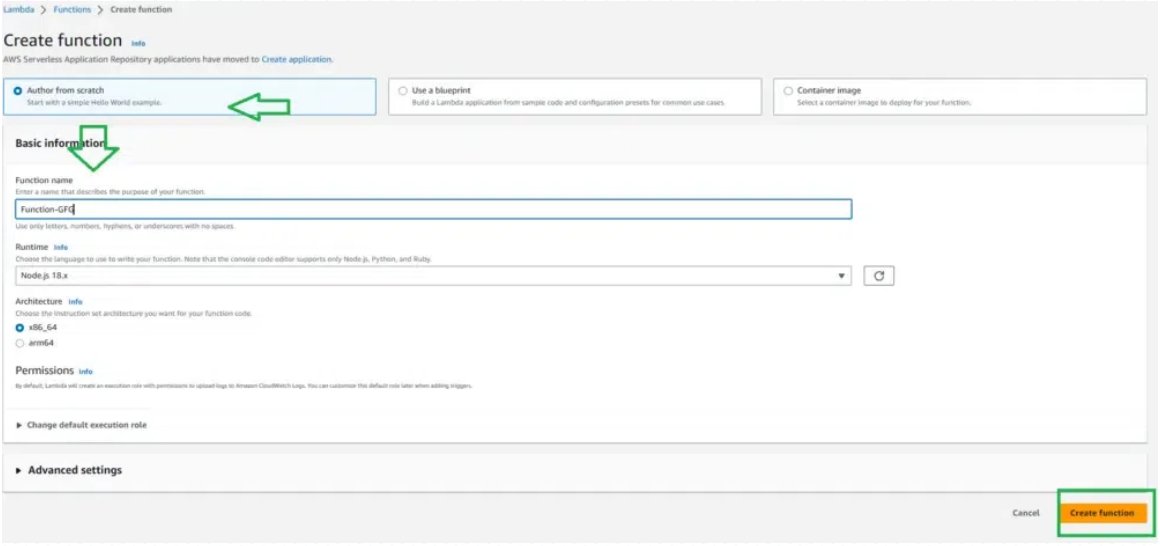
1 Execution Time Limit: Maximum function runtime is 15 minutes.
2 Limited Runtime Environments: Supports specific runtimes like Node.js, Python, Java, and Go.
3 Resource Constraints: Maximum memory of 10 GB, storage of 512 MB (temporary), and limited vCPUs.

Steps for Creating AWS Lambda Functions Using AWS Console

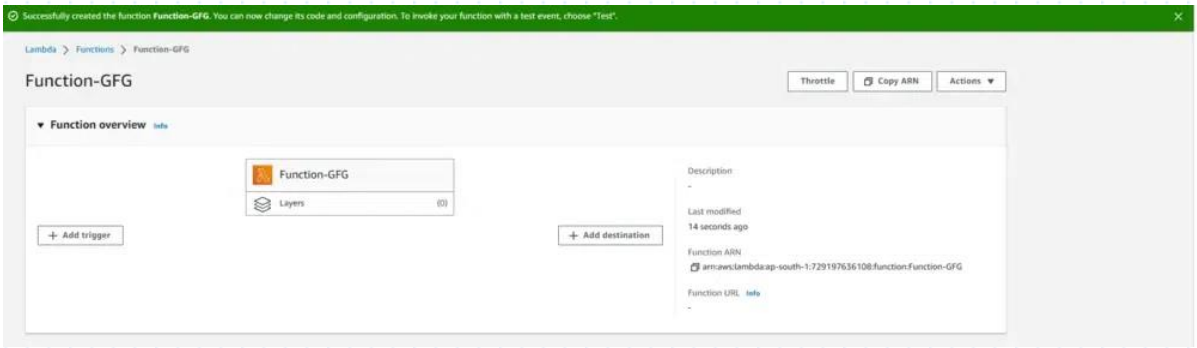
Step 1: Log in to your AWS console and search for Lambda. As shown in the following image.

Step 2: Click on Create function**.**

Step 3: Here we are going to use a sample hello world program by using Author from scratch and configure the details according to your requirement.



Step 4: Successfully our function is created.



Pricing of AWS Lambda Function

AWS lambda pricing is based on the no.of requests made to your function and the time it has taken to execute the function. The following are the pricing factors to be considered for AWS Lambda. As previously mentioned, with AWS Lambda user only pays for what he uses, factoring in the number of requests and duration of the execution of the code. No charges are taken for creating lambda functions. Lambda considers a request to be each time it starts executing in response to a trigger such as an event notification or an invocation volume. The duration of the code is calculated from the moment the code begins executing until it returns or is terminated.

Pricing Factor	Details
Requests	It will charge per request. Each request is counted individually. Pricing varies by region.
Duration	The charges are based on execution time from start to termination. Memory allocation affects cost.
Free Tier	The 1 million free requests per month. 400,000 GB-seconds of compute time per month.

Conclusion

AWS Lambda, a serverless compute service, executes your code in response to events, handling compute resources for you. Discover how AWS's comprehensive set of infrastructure capabilities and services enables rapid and cost-effective modern applications development.