AWS - Lambda

---

1. Introduction

Welcome to this module on **Serverless Computing with AWS**. In this session, we will explore what serverless computing means, how AWS implements it, and specifically, how **AWS Lambda**, the core compute service in the AWS Serverless ecosystem, works.

---

2. What is Serverless Computing?

Serverless computing is a cloud model where developers can build and deploy applications without having to manage servers or infrastructure. This doesn't mean there are no servers—it means the provisioning, scaling, and maintenance of those servers are handled by the cloud provider (in this case, AWS).

Key Characteristics:

- **No server management**: You don't need to set up or manage servers.
- **Automatic scaling**: Applications scale automatically based on demand.
- **Event-driven**: Code is triggered by specific events.
- **Micro-billing**: You are billed only for actual compute used (no idle charges).
- **Faster development**: Developers can focus on business logic, not infrastructure.

---

3. AWS Serverless Ecosystem

While AWS Lambda is the central compute service, the AWS serverless ecosystem includes many services that work together:

Key Components:

- **AWS Lambda**: Runs backend code in response to events.

- **Amazon API Gateway**: Manages REST or WebSocket APIs.
- **Amazon DynamoDB**: NoSQL database with seamless scaling.
- **AWS Step Functions**: Coordinates workflows of multiple Lambda functions.
- **Amazon S3, SNS, SQS, Kinesis**: Integrate with Lambda to build full serverless apps.

---

4. AWS Lambda Overview

AWS Lambda is a compute service where you upload your code, and AWS runs it in response to events.

Key Features:

- **Zero server management**: AWS handles all infrastructure.
- **Flexible triggers**: S3 uploads, API Gateway, IoT, DynamoDB, and more.
- **Micro-billing model**: You're charged per request and compute time.
- **Rapid development**: Simply write the code and define the trigger.

---

5. AWS Lambda vs Amazon EC2

Here's a comparison between **AWS Lambda** and **Amazon EC2**, another AWS compute option:

Feature	Amazon EC2	AWS Lambda
Infrastructure	Manually provisioned virtual machines	Fully managed by AWS
Scaling	Manual or auto-scaling (pre-configured)	Automatic, event-driven scaling
Billing	Billed per hour (or second) while running	Billed per request and execution time

Management Overhead	OS patching, security, monitoring, etc.	No infrastructure maintenance needed
Startup Time	Minutes (boot VM)	Milliseconds to seconds (cold start)
Resource Usage	Constant even when idle	No cost when idle

---

6. AWS Lambda Benefits

- **Event-driven Execution**: Triggers from over 200 AWS and external services.
- **Multiple Language Support**: Python, JavaScript (Node.js), Java, Go, Ruby, PowerShell, and .NET (C#).
- **Pay-as-you-go**: Only pay when your function runs.
- **Automatic Scaling**: Based on number of concurrent events.
- **No Server Management**: Just upload your code and define triggers.
- **Seamless AWS Integration**: Easily connects with S3, DynamoDB, API Gateway, Kinesis, etc.

---

7. AWS Lambda Pricing

Pricing is based on two factors:

1. **Number of Requests**:
    - First 1 million requests/month: **Free**
    - After that: **$0.20 per 1 million requests**
2. **Duration (GB-seconds)**:
    - Calculated as: **Execution time × Allocated memory**

    - First 400,000 GB-seconds/month: **Free**
    - After that: **$1 per 600,000 GB-seconds**

Memory Allocation:

- Ranges from **128 MB to 10 GB**
- Higher memory means more CPU and network performance

Example:

If a function runs for 100ms with 512MB memory:

- **Compute charge** = 0.1s × 0.5 GB = 0.05 GB-seconds
- Multiply by number of invocations for total monthly cost

---

8. How AWS Lambda Works

Key Concept: Event-driven

- Your Lambda function is triggered by **events**.
- Events can come from:
    - **API Gateway** (HTTP request)
    - **S3** (file upload)
    - **DynamoDB Streams**
    - **Kinesis**
    - **IoT Devices**

    - **CloudWatch Events**

Execution Flow Example:

1. A user uploads a file to an S3 bucket.
2. The upload triggers a Lambda function.
3. The function processes the file (e.g., resizes an image).
4. The result is stored or sent to another service (e.g., database or email).

Visual Demo (imagined):

- More clicks on mobile or API = more invocations
- AWS automatically scales out Lambda functions
- **Cost increases** with invocation count and execution duration

---

9. Common Use Cases

1. Real-Time Data Processing

- Process logs, metrics, or files in real-time from S3, Kinesis, or DynamoDB Streams.

2. Serverless APIs

- Combine API Gateway with Lambda to create backend services.

3. Image and Video Processing

- Resize, compress, or transcode media files on upload to S3.

4. Scheduled Tasks

- Run cron jobs or periodic scripts using CloudWatch Events.

5. Chatbots & Messaging

- Use SNS or SQS to build bots or message processors.

6. ETL Pipelines

- Extract, transform, and load data between storage or database layers.

---

10. AWS Lambda Limits and Quotas

Key Quotas:

Category	Limit
Concurrent Executions	1,000 (default, can request increase)
Memory Allocation	128 MB to 10,240 MB (in 64 MB increments)
Execution Timeout	Up to 15 minutes (900 seconds)
Deployment Package Size	50 MB (zipped), 250 MB (unzipped in /tmp)
Environment Variables	Max combined size: 4 KB
Payload Size (Sync)	Max 6 MB (request + response)
Total Account Limit	1,000 concurrent executions (can be increased)

> 💡 Always check the AWS Lambda Limits Page for up-to-date info.

---

11. Summary

- **AWS Lambda** enables true serverless computing by abstracting away infrastructure.
- It’s **cost-efficient**, **scales automatically**, and is perfect for **event-driven workloads**.

- It integrates seamlessly with other AWS services, making it ideal for **microservices**, **APIs**, **real-time data processing**, and more.
- You only pay for what you use—**no idle charges**.