**AWS Lambda Service Presentation**

Introduction to AWS LAMBDA

* AWS Lambda is famous among the modern cloud architecture because of its ability to run code in a fully managed, ***serverless environment***.
* AWS Lambda handles all the administrative tasks for us including
  + - Server
    - OS management
    - Resource allocation
    - Automatic Scaling
    - Code Monitoring
    - Logging

All you need to do is provide code in a language in which AWS lambda is comfortable.

Key Reasons:

* **Cost Efficiency**
  + - * Pay-per-Execution: Only pay for actual compute time consumed.
      * No charges for idle resources, unlike hourly EC2 billing.
      * Highly cost-effective based on usage.
* **Speed**
  + - * Rapid Execution: Functions start and execute quickly.
      * Ideal for applications demanding low latency and rapid response
      * Scalability
* **Automatic Scaling:** 
  + - * Handles vast concurrent requests automatically.
      * Supports 1,000-3,000 concurrent executions per region.
      * Limits are adjustable via AWS Support for higher demands.
* **Event-Driven Execution:**
  + - * Functions triggered by a wide array of AWS services (S3, DynamoDB, Kinesis, API Gateway).
      * Enables highly reactive and integrated workflows.
* **Flexible Resource Allocation:**
  + - * Optimize performance by configuring memory and timeout settings per function.
      * Tailor resources precisely to your application's specific needs.
* **Multi-Language Support:**
  + - * Supports popular languages including Node.js, Python, Java, C#, and Go.
      * Accessible to a diverse range of development teams.
* **Built-in Monitoring:**
  + - * Seamless integration with Amazon CloudWatch.
      * Provides real-time performance tracking and comprehensive logging capabilities.

**Core Advantages of AWS Lambda**

**Optimized Costing**

* **Pay-Per-Use:** Only pay for the compute time consumed.
* Eliminates upfront infrastructure investments.
* Significantly reduces overall operational expenses.

**Reduced Operational Burden**

* **Zero Server Management:** AWS handles all underlying infrastructure.
* Developers focus purely on writing and deploying code.
* Abstracts away server provisioning and maintenance.

**Accelerated Development**

* **Rapid Iteration:** Serverless architecture enables faster development cycles.
* Teams can quickly iterate and deploy updates with ease.
* Boosts agility and time-to-market.

**Built-in High Availability**

* **Automated Reliability:** Automatically manages application availability and fault tolerance.
* Ensures your code runs reliably without manual intervention.
* Inherently designed for resilience.

**Common AWS Lambda Use Cases**

**Real-time Data Processing**

* Transform and analyse data streams instantly.
* Ideal for tasks like image resizing, file conversions, and analytics pipelines.

**Serverless Web Applications**

* Build scalable and cost-effective backends using Lambda with API Gateway.
* Perfect for APIs, microservices, and dynamic web content.

**IoT Data Ingestion & Analytics**

* Process data from connected devices in real-time.
* Enable immediate insights and actions based on IoT device telemetry.

**Automation & Scheduled Tasks**

* Automate routine operations like backups and data synchronization.
* Execute scheduled reports and other recurring jobs with event-driven triggers.

How to use Lambda in AWS?

* **Prerequisites:** 
  + AWS Account
  + Basic AWS Console familiarity
  + Programming knowledge (Python, Node.js, Java, etc.)
* **Creating Your First Lambda Function**

Authoring Your Function

1. **Navigate:** Go to AWS Management Console -> Lambda.
2. **Create Function:** Click "Create function."
3. **Choose Method:** Select "Author from scratch."
4. **Define Basics:**
   * **Function name:** MyHelloWorldFunction
   * **Runtime:** Choose your language (e.g., Python 3.9).
   * **Execution Role:** "Create a new role with basic Lambda permissions."
5. **Write Code:** Use the inline editor.
   * Your code runs within the handler function.
   * **Example (Python):**

Python:

{

def lambda\_handler(event, context):

return {'statusCode': 200, 'body': 'Hello from Lambda!'}

}

1. **Deploy:** Click the "Deploy" button to save.

**Testing & Configuration**

**Headline:** Testing & Optimizing Your Function

* **Testing Your Code:**
  + Go to "Test" tab.
  + "Configure test event" (e.g., hello-world template).
  + Click "Test" to view results and logs.
* **Key Configurations:**
  + **Memory:** Allocate MB (influences CPU & cost).
  + **Timeout:** Set max execution time (e.g., 30 seconds).
  + **Environment Variables:** Store dynamic configurations (API keys, DB strings).
  + *Find under Configuration tab.*

🡪**Making It Event-Driven (Adding Triggers)**

**Headline:** Responding to Events

* **What are Triggers?**
  + AWS services or applications that invoke your Lambda.
  + Your function executes only when a specific event occurs.
* **How to Add:**
  + On Function Overview, click **"+ Add trigger"**.
  + **Select Source:**
    - **API Gateway:** For serverless APIs (HTTP requests).
    - **S3:** For file uploads/deletions.
    - **DynamoDB:** For database changes.
    - **CloudWatch Events/ EventBridge:** For scheduled tasks (cron) or service events.
    - **SQS:** For message queues.
  + **Configure:** Follow specific trigger settings.

**🡪 Monitoring & Next Steps**

**Headline:** Observe, Debug, and Grow

* **Monitoring with CloudWatch:**
  + "Monitor" tab on your function page.
  + View **Invocations, Duration, Errors, Throttles**.
  + "View logs in CloudWatch" for detailed debugging.