🌩Module 3: AWS Compute Services

◆ Introduction

AWS Compute Services provide the **processing power** required to run applications in the cloud. Instead of buying and maintaining your own physical servers, AWS allows you to **rent virtual servers**, **run code without servers**, and **automatically scale** your resources as needed.

◆ 1. Amazon EC2 (Elastic Compute Cloud)

Definition

Amazon EC2 is the core compute service in AWS. It provides **resizable virtual servers**, called **instances**, that you can use to run applications securely and efficiently in the cloud.

EC2 allows you to:

- Choose your own operating system (Linux, Windows, etc.).
- Configure hardware (CPU, RAM, Storage, Network).
- Scale up or down easily.
- Pay only for what you use (per second/minute).

⚙ Key Features of EC2

- **Elasticity:** Scale capacity up or down within minutes.
- **Security:** Use VPC, IAM roles, and Security Groups for protection.
- **Storage Options:** EBS, Instance Store, or EFS.
- **Customizable:** Choose machine images (AMI) with preinstalled software.
- **Monitoring:** Use CloudWatch to monitor performance metrics.

💡 EC2 Instance Types

Each instance type is optimized for different workloads:

Type	Description	Use Case
General Purpose (t3, m5)	Balanced CPU, Memory, Network	Web servers, Dev environments
Compute Optimized (c5)	High CPU performance	Batch processing, gaming servers

Type	Description	Use Case
Memory Optimized (r5)	Large memory capacity	Databases, in-memory caches
Storage Optimized (i3, d2)	High disk throughput	Big Data, data warehousing
GPU Instances (p3, g4)	Graphics or ML acceleration	Machine learning, rendering

---

🖥 Instance Configuration Steps

1. **Choose AMI (Amazon Machine Image):**
   OS + preconfigured software (e.g., Ubuntu with Nginx).
2. **Select Instance Type:**
   Example: `t2.micro` (Free tier eligible).
3. **Configure Instance Details:**
   Number of instances, VPC, subnet, IAM role.
4. **Add Storage:**
   Choose EBS volume size (default: 8 GB).
5. **Add Tags:**
   Label your instance (e.g., Name = WebServer1).
6. **Configure Security Group:**
   Set firewall rules (allow HTTP/SSH).
7. **Review and Launch.**

EC2 Management Tools

- **AWS Management Console:** GUI to manage instances.
- **AWS CLI:** Command-line tool for automation.
- **AWS SDKs:** Programmatically manage EC2 using Python (boto3), Java, etc.

◆ 2. Elastic Load Balancing (ELB)

Definition

ELB automatically distributes **incoming network traffic** across multiple EC2 instances to ensure no single instance becomes overloaded. This improves **fault tolerance** and **availability**.

⚙ Types of Load Balancers

Type	Layer	Description	Use Case
Application Load Balancer (ALB)	Layer 7 (HTTP/HTTPS)	Routes based on URL path or host	Web apps, microservices
Network Load Balancer (NLB)	Layer 4 (TCP/UDP)	High-performance and low latency	Gaming, real-time apps
Gateway Load Balancer (GLB)	Layer 3 (Network)	Integrates 3rd-party security appliances	Firewalls, intrusion detection

---

How ELB Works

- Client sends request → Load Balancer receives it.
- ELB checks instance health.
- ELB routes request to a **healthy EC2 instance**.
- If one instance fails, traffic is automatically redirected to others.

🌟 Benefits

- High availability
- Fault tolerance
- Automatic scaling with Auto Scaling Group
- Secure (supports SSL/TLS termination)
- Health monitoring of instances

◆ 3. Auto Scaling and Elasticity

Definition

Auto Scaling automatically adjusts the number of EC2 instances based on **demand** or **performance metrics** (CPU usage, request count, etc.).
This ensures your application always has the **right number of instances** running.

⚙ Components of Auto Scaling

1. **Launch Template / Launch Configuration:**
   Defines what instance type and AMI to launch.
2. **Auto Scaling Group (ASG):**
   Logical group of EC2 instances managed together.

3. **Scaling Policies:**
   Define rules for scaling (e.g., add instance if CPU > 70%).

💡 Example Scenario

- During peak hours (9 AM–6 PM): Traffic increases → Auto Scaling adds 3 instances.
- During low usage (night): Traffic decreases → Auto Scaling removes 2 instances.

This ensures **cost efficiency** and **performance stability**.

🌟 Benefits of Auto Scaling

- **Elasticity:** Automatically matches capacity with demand.
- **Cost-effective:** Only pay for used resources.
- **Resilience:** Replaces unhealthy instances automatically.

◆ 4. AWS Lambda (Serverless Computing)

Definition

AWS Lambda is a **serverless compute service** that runs your code automatically **without managing servers**. You just upload your function — AWS handles scaling and execution.

⚙ How Lambda Works

1. You upload code or write it directly in AWS Console.
2. Set a **trigger** (like API Gateway, S3, or CloudWatch event).
3. Lambda executes your code whenever the trigger happens.
4. You pay only for the milliseconds your code runs.

Key Concepts

- **Event-driven:** Executes in response to triggers.
- **Stateless:** Each function invocation is independent.
- **Scalable:** Automatically handles thousands of requests per second.

Supported Languages

Python, Node.js, Java, C#, Ruby, Go, PowerShell.

💡 Example Use Case

📸 A photo is uploaded to S3 → Lambda triggers → resizes image → saves new image in another S3 bucket.

This happens automatically, without any EC2 server running 24/7.

🌟 Benefits

- No infrastructure management.
- Highly scalable.
- Pay only for execution time.
- Integrated with many AWS services.

◆ 5. AWS Elastic Beanstalk

Definition

Elastic Beanstalk is a **Platform as a Service (PaaS)** that helps developers **deploy and manage applications** without needing to configure infrastructure manually.

⚙ How Elastic Beanstalk Works

1. Developer uploads the application (ZIP or Git repo).
2. Beanstalk automatically:
    - Creates EC2 instances.
    - Sets up Load Balancer and Auto Scaling.
    - Deploys your app.
    - Monitors performance with CloudWatch.
3. You can manage and update the environment easily.

Supported Platforms

- Python (Flask, Django)

- Java (Spring Boot)
- .NET
- PHP
- Node.js
- Ruby
- Go

💡 Example

You upload a Django web app to Beanstalk.
Beanstalk automatically:

- Creates EC2 and Load Balancer.
- Sets up Auto Scaling.
- Deploys your app and monitors it.

You only focus on code — AWS handles infrastructure.

🌟 Benefits

- Simplifies deployment and management.
- Automatic scaling and health monitoring.
- Cost-efficient and customizable.
- Integrated with EC2, S3, RDS, and CloudWatch.

◆ 6. Summary Table

AWS Service	Type	Purpose	Key Benefit
EC2	IaaS	Virtual servers on demand	Full control of OS & environment
ELB	IaaS	Distribute traffic	High availability & fault tolerance
Auto Scaling	IaaS	Scale EC2 automatically	Cost optimization & elasticity
Lambda	Serverless	Run code automatically	Pay only for use, no servers
Elastic Beanstalk	PaaS	App deployment & management	Simplified setup, scaling, monitoring