Parul® University

AMAZON WEB SERVICES

By – Mr. Ram Rawat

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-6

AWS Security , Monitoring, Scaling, and Billing

What is IAM?

- IAM (Identity and Access Management) is an AWS service that helps you securely manage access to AWS resources.
- It controls who can sign in (authentication) and what they can do (authorization).
- Provides users, groups, roles, and policies to assign permissions.

Example:

- A developer gets access to only EC2 instances.
- An admin gets full access to all AWS services.

Key IAM Concepts:

- **Users** → Individual identities with credentials (e.g., *John – Developer*).
- **Groups** → Collection of users with same permissions (e.g., *Admins, Developers*).
- **Roles** → Temporary access given to AWS services or users (e.g., *EC2 role to access S3*).
- **Policies** → JSON documents that define permissions (e.g., *Allow read-only access to S3*).

Example:
User: Alice (Developer)
Group: Developers → Policy: Can launch and manage EC2 instances only.

Why IAM is Important?

- Ensures secure access control to AWS resources.
- Follows least privilege principle (users get only required permissions).
- Provides centralized management of users, roles, and policies.
- Helps with compliance, auditing, and monitoring.
- Protects against unauthorized access.

AWS IAM: Implementing IAM Policies

IAM Policies → JSON documents that define what actions are allowed or denied on AWS resources.
Policies are attached to Users, Groups, or Roles.
Each policy has:

- Effect → Allow or Deny
- Action → What can be done (e.g., s3:GetObject)
- Resource → On which AWS resource (e.g., mybucket/*)

Example : Policy (Read-only S3 Access):

```
{
"Effect": "Allow",
"Action": "s3:GetObject",
"Resource": "arn:aws:s3:::mybucket/*“
}
```

This allows the user/role to read objects from S3 but not delete them.

Parul®
University

WHERE TO WRITE THIS POLICY

Parul®
University

Monitoring & Scaling in AWS

AWS CloudWatch

- A monitoring and observability service in AWS
- Collects metrics, logs, and events from AWS resources and applications.
- Helps you set alarms, visualize performance, and take automated actions.

Examples:

- Monitor **EC2 instance CPU usage** → Trigger alarm if CPU > 80%.
- Track **S3 bucket storage size** → Send alert if it crosses threshold.
- Automatically restart an EC2 instance if it becomes **unhealthy**.

Example

•An **online shopping website** runs on multiple **EC2 instances**.
•During **festival sales**, traffic increases sharply.
•**CloudWatch** monitors:
- **CPU usage** of EC2 instances
- **Number of incoming requests**

•If **CPU > 80%**, CloudWatch triggers an **alarm** → notifies the admin and triggers **Auto Scaling** to launch new EC2 instances.
•This keeps the website **fast, available, and stable** for all customers.

Elastic Load Balancing (ELB)

•Automatically distributes **incoming traffic** across multiple targets (EC2, containers, IPs).

•Ensures **high availability**, **fault tolerance**, and better performance.

•Supports health checks → routes traffic only to **healthy instances**.

Example:

•A company's website runs on **3 EC2 servers**.

•ELB distributes user requests among all servers → prevents one server from overloading.

•If one EC2 fails, ELB redirects traffic to the remaining healthy servers → **zero downtime**.

Auto Scaling

- Automatically adjusts the number of EC2 instances based on demand.
- Ensures applications are always available, reliable, and cost-efficient.
- Works with CloudWatch alarms to scale up or down.

Example:

•An e-commerce site experiences heavy traffic during a **sale**.
•Auto Scaling **adds more EC2 instances** to handle the load.
•When traffic drops at night, it **removes extra instances** to save cost.

Types of Auto Scaling

Dynamic Scaling:

- Reacts to changes in demand using CloudWatch alarms.
- Example: Add instance when CPU > 80%.

Predictive Scaling:

- Uses machine learning to forecast traffic and scale in advance.
- Example: Before Black Friday, AWS predicts high load and adds servers early.

Scheduled Scaling:

- Scales based on a set time schedule.
- Example: Add 2 instances every day at 9 AM, remove them at 9 PM.

Example

- During festival sales → **Dynamic scaling** adds servers instantly.
- At night → **Scheduled scaling** reduces servers.
- Before a planned marketing campaign → **Predictive scaling** adds capacity in advance.

Parul® University

DIGITAL LEARNING CONTENT

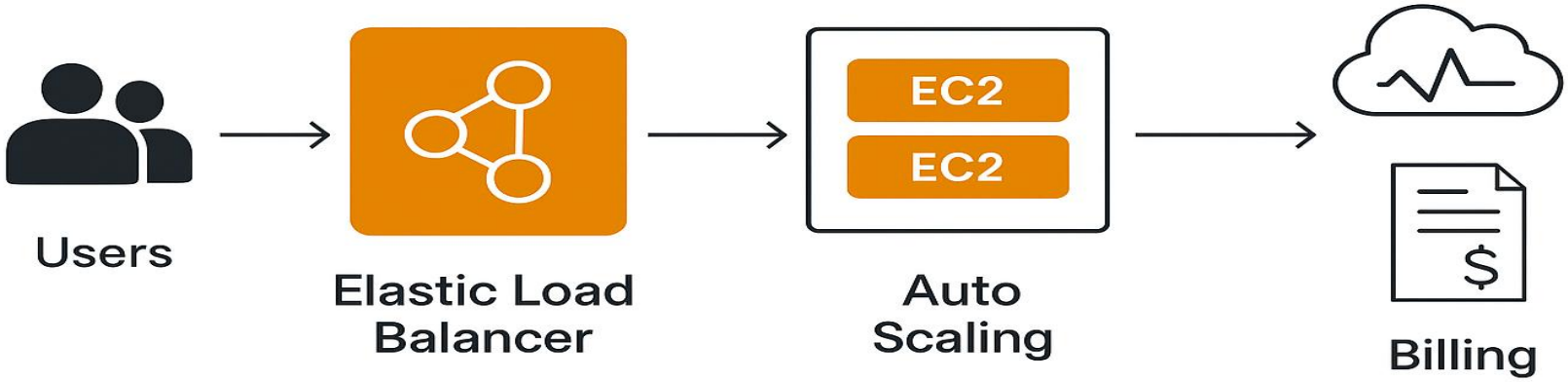
Cloud Economics & Billing

- **Pay-as-you-go pricing** → Pay only for what you use, no upfront cost.
- Use **Cost Explorer & Billing Dashboard** to analyze and optimize spend.
- Set **budgets & alerts** to control unexpected costs.

Example:

- Budget = **$100/month**
- At **80% usage**, AWS sends an alert → helps prevent overspending.

Scale & Load Balance Your Architecture



Scale & Load Balance Your Architecture

- Combines **Auto Scaling + Elastic Load Balancer + CloudWatch**.
- Ensures **high availability, fault tolerance, and cost efficiency**.

Example (End-to-End):

- **CloudWatch** monitors website traffic and triggers alarms.
- **Auto Scaling** launches new EC2 servers when demand increases.
- **ELB** distributes user requests across all servers.
- **Billing Dashboard** tracks monthly usage and cost.

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in