AWS CLOUD PRATITIONER (CLF-C02) NOTES

Module 1: Cloud Computing Basics

1. What is Cloud Computing?

- **Definition:** Delivery of computing services (servers, storage, databases, networking, software) over the internet ("the cloud").
- **Benefits:**
  - On-demand access
  - Broad Network Access
  - Multi-tenancy and Resource Pooling
  - Rapid Elasticity and Scalability
  - Pay-as-you-go pricing
  - Scalability and flexibility
  - No need to manage physical infrastructure
- **Example:** Hosting a website on AWS instead of buying and managing your own servers.

---

2. The Different Types of Cloud Computing

- **Deployment Models:**
  1. **Public Cloud –** Services offered over the internet (e.g., AWS, Azure, GCP)
  2. **Private Cloud –** Cloud infrastructure operated solely for one organization (e.g., Rackspace)
  3. **Hybrid Cloud –** Combination of public and private clouds

     Example: A company uses AWS for web hosting (public) and keeps sensitive data on-premises (private).

---

3. AWS Cloud Overview

- **Amazon Web Services (AWS):** Leading cloud provider offering 200+ services.
- **Core Services:**
  - Compute (EC2, Lambda)
  - Storage (S3, EBS)
  - Databases (RDS, DynamoDB)
  - Networking (VPC, Route 53)
- **Global Infrastructure:**

1. Regions

- **Definition**: A **Region** is a geographical area that contains multiple **Availability Zones**.
- **Purpose**: Allows you to deploy applications close to your users for lower latency and compliance.
- **Example**: us-east-1 (N. Virginia), ap-south-1 (Mumbai)

---

2. Availability Zones (AZs)

- **Definition**: A **physically separate data center** within a region, each with independent power, cooling, and networking.
- **Purpose**: Provide high availability and fault tolerance.
- **Example**: us-east-1 has AZs like us-east-1a, us-east-1b, etc.

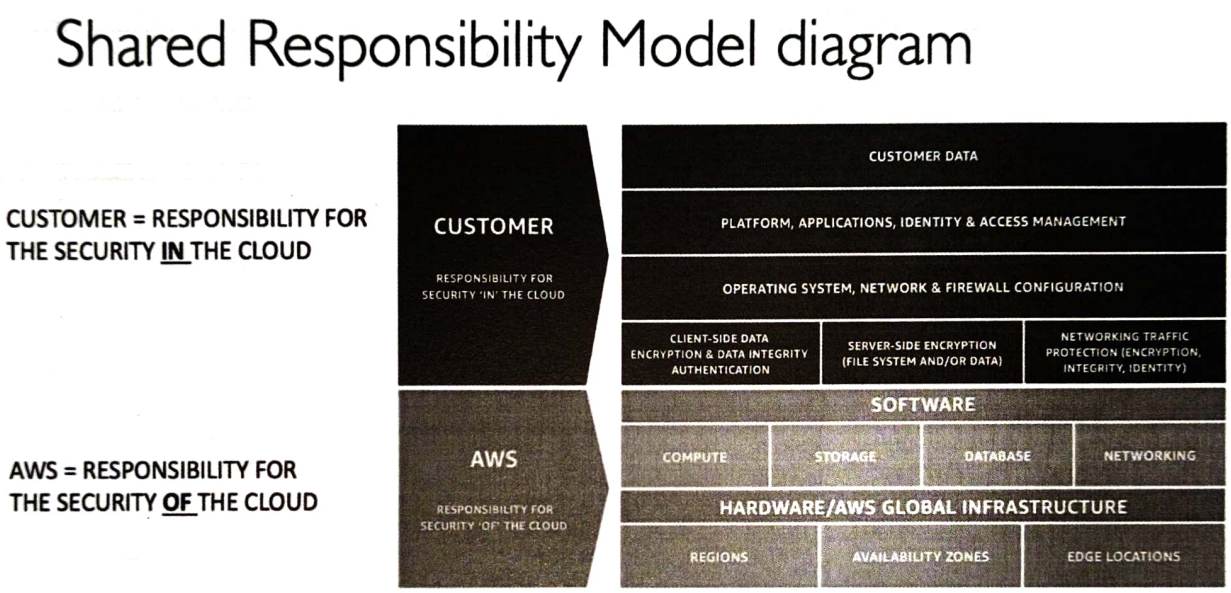
---

3. Edge Locations

- **Definition**: An **Edge Location** is an AWS **data center** used by services like **CloudFront** to **cache and deliver** content and services **closer to end users**.
- **Purpose**: Reduce latency for content delivery and DNS resolution.
- **Example**: A user in Delhi accesses a cached video from a nearby Edge Location instead of the origin server in Mumbai.
- **Use Case:** Startups to enterprises use AWS for hosting, analytics, ML, and more.

4. Shared Responsibility Model & AWS Acceptable Use Policy

- **Shared Responsibility Model:**
  - **AWS:** Responsible for security *of* the cloud (hardware, software, networking).
  - **Customer:** Responsible for security *in* the cloud (data, IAM, configurations).



- **Acceptable Use Policy:**
  - Prohibits illegal, harmful, or abusive use of AWS services.
  - Includes restrictions on spam, malware, and unauthorized access.

Module 2: IAM (Identity and Access Management)

1. Users

- **Definition**: Represents a single person or application.
- **Purpose**: Used to sign in to the AWS Management Console or access AWS services via CLI/SDK.
- **Example**:
    - User: john.doe
    - Permissions: Can access S3 and EC2
    - Login: Uses a username and password

---

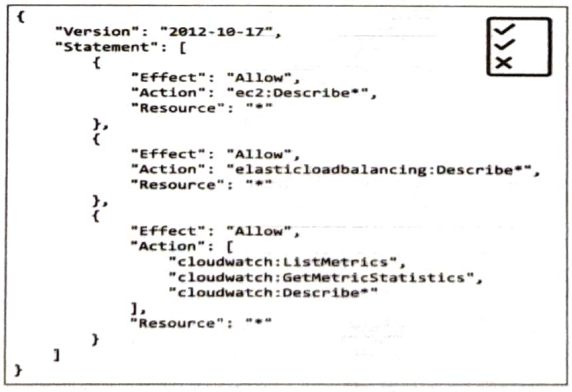
2. Groups

- **Definition**: A collection of IAM users.
- **Purpose**: Assign permissions to multiple users at once.
- **Example**:
    - **Group:** Developers
    - **Members:** john.doe, jane.smith
    - **Permissions:** Full access to EC2 and Lambda

---

3. Policies

- **Definition**: JSON documents that define permissions.
- **Purpose**: Attach to users, groups, or roles to grant access.
- **Example**:



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:Describe*"
            ],
            "Resource": "*"
        }
    ]
}
```

---

4. Roles

- **Definition:** IAM Roles are AWS identities with specific permissions that can be assumed by trusted entities such as AWS services or external users.
- **Purpose:**
  - Allow AWS services to perform actions on your behalf.
  - Enable secure access without using long-term credentials.
- **Common Use Cases:**
  - **EC2 Instance Roles** – Allow EC2 instances to access other AWS services.
  - **Lambda Function Roles** – Grant Lambda functions permissions to interact with AWS resources.
  - **Roles for CloudFormation** – Enable CloudFormation to manage resources during stack operations.
- **Example:**
  - **Role Name:** EC2S3AccessRole
  - **Trusted Entity:** EC2 Instance
  - **Permissions:** Read/write access to Amazon S3
- **Key Concept:** Instead of embedding credentials in your application, assign a role to the service (like EC2), and it will assume the role to access AWS resources securely.

---

5. Security

- **Components**:
  - **MFA (Multi-Factor Authentication)**: Adds an extra layer of security.
  - **Password Policy**: Enforces rules like minimum length, complexity.
- **Example**:
  - **MFA:** Enabled for john.doe using an authenticator app
  - **Password Policy:** Minimum 12 characters, must include symbols

---

6. AWS CLI

- **Definition**: The **AWS Command Line Interface (CLI)** is a tool that allows you to interact with AWS services using commands in your terminal or command-line shell.
- **Example**:

```
➜  ~ aws s3 cp myfile.txt s3://ccp-mybucket/myfile.txt
upload: ./myfile.txt to s3://ccp-mybucket/myfile.txt
➜  ~ aws s3 ls s3://ccp-mybucket
2021-05-14 03:22:52          0 myfile.txt
➜  ~
```

  - Copies a file from your local machine to an S3 bucket.
  - Lists the objects stored in the specified S3 bucket.

---

7. AWS SDK

- **Definition**: Software Development Kit for accessing AWS using programming languages.
- **Example** (Python using Boto3):

```
import boto3
s3 = boto3.client('s3')
s3.list_buckets()
```

---

8. Access Keys

- **Definition**: Used for programmatic access via CLI or SDK.
- **Components**: Access Key ID + Secret Access Key
- **Example**:
  - **Access Key ID:** AKIAIOSFODNN7EXAMPLE
  - **Secret Access Key:** wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

---

9. Audit

- **Tools:**
  - **IAM Credential Reports**: Lists all users and the status of their credentials.
  - **IAM Access Advisor**: Shows permissions granted and last accessed time.
- **Example:**
  - **Credential Report:** Reveals john.doe hasn't rotated access keys in 90 days
  - **Access Advisor:** Shows jane.smith hasn't used EC2 in 6 months

---

Module 3: EC2 (Elastic Compute Cloud)

1. EC2 Instance

- **Definition**: A virtual server in AWS. (Infrastructure as a Service)

  (Allows users to run Virtual Servers on demand)

- **Components**:
  - **AMI (Amazon Machine Image)**:
    - Defines the OS (e.g., Ubuntu, Amazon Linux)
    - Templates for launching EC2 instances, custom machine images
  - **Instance Size**: Determines CPU and RAM (e.g., t2.micro, m5.large)
  - **Storage**: EBS volumes attached (e.g., 30 GB gp2)
  - **Security Groups**: Acts as a virtual firewall
  - **EC2 User Data**: Script that runs at first boot
- **Example**:
  - **AMI:** Ubuntu 22.04
  - **Instance Type:** t2.micro
  - **Storage:** 30 GB
  - **Security Group:** Allows HTTP (80), HTTPS (443), SSH (22)
  - **User Data:** Installs Apache on launch

---

2. Security Groups

- **Definition**: Virtual firewalls for EC2 instances.
- **Purpose**: Control inbound and outbound traffic.
- **Example**:
  - **Inbound:** Allow TCP on port 22 (SSH), 80 (HTTP)
  - **Outbound:** Allow all traffic

---

3. EC2 User Data

- **Definition**: Script executed at the first boot of the instance.
- **Purpose**: Automate setup tasks like installing software.
- **Example** (Bash script):

```
#!/bin/bash
apt update
apt install -y apache2
systemctl start apache2
```

---

4. SSH (Secure Shell)

- **Definition**: Securely connect to EC2 instances.
- **Port**: 22
- **Example**:

```
ssh -i my-key.pem ubuntu@ec2-3-91-123-45.compute-1.amazonaws.com
```

---

5. EC2 Instance Role

- **Definition**: IAM role attached to an EC2 instance.
- **Purpose**: Grant permissions to the instance to access AWS services.
- **Example**:
  - **Role:** S3ReadOnlyRole
  - **Permissions:** Read access to S3
  - **Use Case:** EC2 instance pulls data from an S3 bucket

---

6. Purchasing Options

- **On-Demand**: Pay per hour/second, no commitment
    - ***Use Case*:** Short-term, unpredictable workloads
- **Spot Instances**: Up to 90% cheaper, can be interrupted
    - ***Use Case*:** Batch jobs, fault-tolerant workloads
- **Reserved Instances**: 1 or 3-year commitment
    - ***Standard*:** Up to 75% savings
    - ***Convertible*:** Change instance type
- **Dedicated Host**: Physical server fully dedicated to you
    - ***Use Case*:** Licensing or compliance needs
- **Dedicated Instance**: Runs on hardware dedicated to you, but not isolated from other instances

Module 4: EC2 Instance Storage

1. EBS Volumes (Elastic Block Store)

- **Definition**: Network-attached storage that can be attached to one EC2 instance at a time.
- **Key Features**:
  - Persistent storage (data survives instance stop/start)
  - Can be resized and backed up
  - Mapped to a specific **Availability Zone**
- **Use Case:**
  - Preserve Root volume when instance is terminated
- **Example**:
  - **Volume Type:** gp3 (General Purpose SSD)
  - **Size:** 100 GB
  - **Attached to:** i-0abc123def456ghij in us-east-1a

---

2. EBS Snapshots

- **Definition**: Backups of EBS volumes stored in S3.
- **Use Cases**:
  - Backup and restore
  - Create new volumes in different AZs
- **Example**:
  - **Snapshot ID:** snap-0123456789abcdef0
  - Used to create a new volume in us-east-1b

---

3. AMI (Amazon Machine Image)

- **Definition**: Pre-configured template for launching EC2 instances.
- **Includes**:
  - OS + application code + configurations

- **Example**:
  - Custom AMI with Ubuntu + NGINX + app code
  - Used to launch multiple identical EC2 instances

---

4. EC2 Image Builder

- **Definition**: Automates the creation, testing, and distribution of AMIs.
- **Use Case**:
  - Regularly update AMIs with latest patches and software
- **Example**:
  - **Pipeline:** Build → Test → Approve → Distribute AMI to multiple regions

---

5. EC2 Instance Store

- **Definition**: Temporary, high-speed storage physically attached to the host.
- **Key Points**:
  - **Ephemeral**: Data is lost if the instance is stopped or terminated
  - **High IOPS**: Great for temporary data like caches or buffers
- **Example**:
  - **Instance Type:** i3.large with 475 GB NVMe SSD
  - **Use Case:** Temporary storage for processing logs

---

6. EFS (Elastic File System)

- **Definition**: Managed network file system (NFS) that can be mounted by multiple EC2 instances.
- **Key Features**:
  - Scalable and shared
  - Regional service (not tied to a single AZ)
- **Example**:
  - Mounted on 10 EC2 instances for shared access to app data

---

7. EFS-IA (Infrequent Access)

- **Definition**: Storage class for EFS with lower cost for files accessed less frequently.
- **Use Case**:
  - Archive or backup data
- **Example**:
  - Store logs older than 30 days in EFS-IA to reduce costs

---

8. FSx for Windows File Server

- **Definition**: Fully managed Windows file system with SMB protocol support.
- **Use Case**:
  - Windows-based applications needing shared file storage
- **Example**:
  - Used by a .NET application running on Windows EC2 instances

---

9. FSx for Lustre

- **Definition**: High-performance file system for compute-intensive workloads.
- **Use Case**:
  - Machine learning, big data, HPC
- **Example**:
  - Used with Amazon S3 to process large datasets for analytics

Module 5: ELB & ASG (Elastic Load Balancing and Auto Scaling Group)

1. Cloud Concepts Overview

Concept	Description	Example Use Case
High Availability	System remains operational even if some components fail.	Deploy EC2 instances in multiple AZs.
Scalability	Ability to handle increased load.	Add more EC2 instances **(Horizontal)** or upgrade instance type **(Vertical).**
Elasticity	Automatically adjust resources based on demand.	Auto Scaling Group adds/removes EC2s based on traffic.
Agility	Speed and flexibility in deploying and managing resources.	Quickly launch new environments using CloudFormation.

---

2. Elastic Load Balancers (ELB)

- **Purpose**: Distribute incoming traffic across multiple EC2 instances.
- **Supports**:
  - Multi-AZ deployments
  - Health checks to route traffic only to healthy instances
- **Types**:
  - **Classic Load Balancer (CLB)** – Legacy, supports HTTP/HTTPS & TCP
  - **Application Load Balancer (ALB)** – Layer 7 (HTTP/HTTPS), supports path-based routing
  - **Network Load Balancer (NLB)** – Layer 4 (TCP/UDP), ultra-low latency

- **Gateway Load Balancer (GWLB)** – Layer 3, used for third-party virtual appliances

- **Example**:
  - ALB routes /api/* to backend service A and /web/* to service B
  - NLB handles millions of TCP connections for a gaming app

---

3. Auto Scaling Groups (ASG)

- **Purpose**: Automatically manage the number of EC2 instances based on demand.
- **Key Features**:
  - Launches instances across multiple AZs
  - Replaces unhealthy instances automatically
  - Integrated with ELB for traffic distribution
- **Scaling Policies**:
  - **Target Tracking**: Maintain a metric (e.g., CPU at 60%)
  - **Step Scaling**: Add/remove instances based on thresholds
  - **Scheduled Scaling**: Scale based on time (e.g., business hours)
  - **Predictive Scaling**:
    - Uses machine learning to predict future traffic ahead of time
    - Useful when our load has predictive time-based pattern
- **Example**:
  - ASG with min=2, max=10 instances
  - Scales out when CPU > 70%, scales in when CPU < 30%
  - Integrated with ALB to distribute traffic

Module 6: Amazon S3

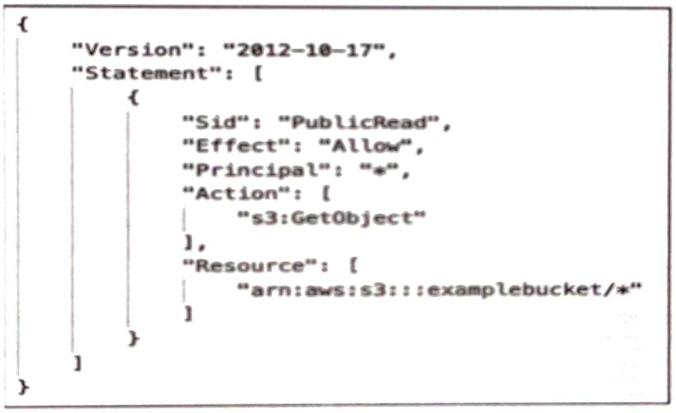
1. Buckets vs Objects

- **Bucket**: Container for storing objects; must have a globally unique name.
- **Object**: File stored in a bucket, includes metadata.
- **Region**: Buckets are tied to a specific AWS region.
- **Example**:
    - **Bucket:** my-website-assets in us-east-1
    - **Object:** index.html, logo.png

---

2. S3 Security

- **IAM Policies**: Control access at the user or role level.
- **Bucket Policies**: JSON-based rules for public or cross-account access.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket/*"
            ]
        }
    ]
}
```

    - **Resources:** buckets and objects
    - **Effect:** Allow / Deny
    - **Actions:** Set of API to Allow or Deny
    - **Principal:** The account or user to
- **Public Access Settings**: Block public access by default.
- **Encryption Options**:
    - SSE-S3 (AWS-managed keys)
    - SSE-KMS (Customer-managed keys)
    - SSE-C (Customer-provided keys)

- **Example**:
  - Bucket policy allows public read access to index.html
  - SSE-KMS used for encrypting sensitive documents

---

3. S3 Static Website Hosting

- **Purpose**: Host static HTML/CSS/JS websites directly from S3.
- **Example**:
  - **Bucket:** my-portfolio-site
  - **Website URL:** http://my-portfolio-site.s3-website-us-east-1.amazonaws.com

---

4. S3 Versioning

- **Definition**: Keeps multiple versions of an object.
- **Benefits**:
  - Recover from accidental deletes or overwrites
- **Example**:
  - resume.pdf has versions: v1, v2, v3

---

5. S3 Replication

- **Definition: S3 Replication** is a feature that automatically copies objects from one S3 bucket to another. It helps with compliance, lower latency, and disaster recovery.
- **Types**:
  - **Same-Region Replication (SRR) –** Log Replication, Live Replication between production and Test accounts
  - **Cross-Region Replication (CRR) –** Compliance, Lower latency access, Replication across accounts
- **Requirements**: Versioning must be enabled.
- **Example**:
  - Replicate data from us-east-1 to eu-west-1 for disaster recovery.

6. S3 Storage Classes

Class	Use Case
Standard	Frequent access, low latency
Infrequent Access (IA)	Less frequent access, lower cost
Intelligent-Tiering	Automatically moves data between tiers
One Zone-IA (Z-IA)	IA in a single AZ, lower cost
Glacier Instant Retrieval	Archive with milliseconds access
Glacier Flexible Retrieval	Archive with minutes to hours access
Glacier Deep Archive	Lowest cost, access in 12+ hours

---

7. Snow Family

- **Purpose**: Transfer large amounts of data to AWS using physical devices.
- **Devices**:
  - **Snowcone**: Small, portable
  - **Snowball**: Larger capacity
  - **Snowmobile**: Exabyte-scale data transfer
- **Use Case**: Data migration, edge computing

---

8. OpsHub

- **Definition**: Desktop application to manage Snow Family devices.
- **Features**:
  - Device setup
  - Data transfer and Monitoring

---

9. AWS Storage Gateway

- **Definition**: Hybrid cloud storage service.
- **Purpose**: Extend on-premises storage to AWS.
- **Types**:
  - File Gateway
  - Volume Gateway
  - Tape Gateway
- **Example**:
  - Backup on-prem files to S3 using File Gateway

Module 7: Databases & Analytics

1. Relational Databases – OLTP (Online Transaction Processing)

- **Amazon RDS**: Managed relational databases (MySQL, PostgreSQL, Oracle, SQL Server, MariaDB)
- **Amazon Aurora**: High-performance, MySQL/PostgreSQL-compatible engine
- **Use Case**: E-commerce, financial apps

Multi-AZ vs Read Replicas vs Multi-Region

Feature	Purpose	Example Use Case
Multi-AZ	High availability (failover support)	Automatic failover during DB failure
Read Replica	Read scalability	Offload read traffic from primary DB
Multi-Region	Disaster recovery, global apps	Serve users from multiple continents

---

2. In-Memory Database

- **Amazon ElastiCache**:
  - Managed Coaching Service for **Redis** and **Memcached**
  - Low-latency caching layer
- **Use Case**: Session storage, leaderboard, real-time analytics

---

3. Key/Value Database

- **Amazon DynamoDB**:
  - **Fully managed NoSQL database** with high availability across 3 AZs
  - Serverless, NoSQL
  - Millisecond performance at scale
  - DyanoDB is a **Key/Value database**

- **Amazon DAX**:
    - In-memory cache for DynamoDB
- **Use Case**: Shopping carts, IoT data, mobile apps

---

4. Data Warehouse – OLAP (Online Analytical Processing)

- **Amazon Redshift**:
    - Based on PostgreSQL and designed for OLAP (Online Analytical Processing)
    - Columnar storage, SQL-based
    - Optimized for complex queries on large datasets
- **Use Case**: Business intelligence, reporting
- **NOTE**:
    - **Redshift Serverless**: Automatically provisions and scales data warehouse underlying capacity.
    - **Use Case**: Reporting, Dashboarding applications, Real time analytics

---

5. Big Data Processing

- **Amazon EMR (Elastic MapReduce)**:
    - EMR helps creating Hadoop Clusters (Big Data) to analyze and process vast amounts of data
    - Managed Hadoop, Spark, Hive, Presto
    - Scalable data processing
- **Use Case**: Log analysis, machine learning pipelines

---

6. Serverless Query & Visualization

- **Amazon Athena**:
    - Analyze data in S3 using Serverless SQL
    - Query S3 data using SQL
    - Serverless, pay-per-query

- **Amazon QuickSight**:
    - BI tool for dashboards and visualizations
    - Serverless, integrates with Redshift, Athena, S3

---

7. Document Database

- **Amazon DocumentDB**:
    - MongoDB-compatible
    - Stores JSON-like documents
    - **Use Case**: Content management systems, catalogs

---

8. Ledger Database

- **Amazon QLDB (Quantum Ledger Database)**:
    - Immutable, cryptographically verifiable
    - Transparent history of changes
- **Use Case**: Financial transactions, supply chain

---

9. Blockchain

- **Amazon Managed Blockchain**:
    - Supports **Hyperledger Fabric** and **Ethereum**
- **Use Case**: Decentralized apps, consortium networks

---

10. ETL & Data Catalog

- **AWS Glue**:
    - Serverless ETL (Extract, Transform, Load)
    - Includes **Glue Data Catalog**
- **Use Case**: Prepare data for analytics in Redshift or Athena

---

11. Database Migration

- **AWS DMS (Database Migration Service)**:
  - Migrate databases to AWS with minimal downtime
- **Use Case**: Move from on-prem SQL Server to RDS PostgreSQL

---

12. Graph Database

- **Amazon Neptune**:
  - Fully managed **graph database** for highly connected datasets
  - Supports **Gremlin** and **SPARQL**
  - Optimized for relationships and graph queries
  - **Use Case:** Ideal for Social Networks, Knowledge Graphs, Fraud Detection and Recommendation Engine

---

13. Time-Series Database

- **Amazon Timestream**:
  - Purpose-built for time-series data
  - Serverless, scalable
- **Use Case**: IoT sensor data, application metrics

Module 8: Other Compute Services

1. Docker

- **Definition**: Container technology to package and run applications with all dependencies.
- **Use Case**: Run microservices in isolated environments.
- **Example**: A Node.js app containerized and deployed on AWS.

---

2. Amazon ECS (Elastic Container Service)

- **Definition**:
  - Launch Docker containers on EC2 instances.
  - **You must provision & maintain the infrastructure (the EC2 instances)**
- **Use Case**: Run and scale containerized applications.
- **Example**: Deploy a microservices architecture using ECS on EC2.

---

3. AWS Fargate

- **Definition**:
  - Serverless compute engine for containers.
  - **You do not provision the infrastructure (no EC2 instances to manage)**
- **Key Features**:
  - No need to manage EC2 instances
  - Pay-per-use model
- **Use Case**: Run containers without provisioning infrastructure.
- **Example**: Deploy a containerized API backend using Fargate.

---

4. Amazon ECR (Elastic Container Registry)

- **Definition**: Private Docker image repository.
- **Use Case**: Store and manage container images securely.
- **Example**: Push a Docker image to ECR and deploy it via ECS

5.Amazon EKS

- **Definition:** Amazon EKS is a **managed Kubernetes service** that allows you to run Kubernetes clusters on AWS without needing to install or operate your own Kubernetes control plane or nodes.
- **Use Case**
  - Deploy, manage, and scale **containerized applications** using Kubernetes.
  - Run containers on **EC2 instances** or **AWS Fargate (serverless)**.
  - Ideal for teams that want to use Kubernetes but offload infrastructure management to AWS.
  - **Example**
    A company uses **Amazon EKS** to deploy a microservices-based application. The containers are orchestrated using Kubernetes and run on **Fargate**, allowing the team to focus on development without managing servers.

---

6. AWS Batch

- **Definition**: Run batch computing workloads on AWS.
- **Use Case**: Large-scale parallel jobs like video rendering or data processing.
- **Example**: Process thousands of images using AWS Batch and EC2 Spot Instances.

---

7. Amazon Lightsail

- **Definition**: Simplified cloud platform with predictable pricing.
- **Use Case**: Host simple web apps, blogs, or dev environments.
- **Example**: Launch a WordPress site with Lightsail for a fixed monthly cost.

---

8. AWS Lambda

8.1. Overview

- **Definition**: Serverless, event-driven compute service.
- **Model**: Function-as-a-Service (FaaS)
- **Scaling**: Automatic and seamless
- **Invocation Time**: Up to 15 minutes

8.2. Billing

- **Based On**:
  - **Execution Time × Memory Allocated**
  - **Number of Invocations**
- **Example**:
  - 1 million invocations/month with 512 MB RAM = low cost

8.3. Language Support

- **Supported**: Python, Node.js, Java, Go, .NET, Ruby, etc.
- **Not Supported**: Arbitrary Docker containers (use ECS/Fargate instead)

8.4. Use Cases

- **Image Processing**: Generate thumbnails when images are uploaded to S3
- **Scheduled Tasks**: Run serverless cron jobs using EventBridge
- **APIs**: Expose Lambda functions via **API Gateway**

Module 9: Deployments and Managing Infrastructure as Scale

1. AWS CloudFormation

- **Definition:** AWS CloudFormation is an Infrastructure as Code (IaC) service that lets you define and provision AWS resources using templates.
- **Use Case:** Automate and standardize the deployment of AWS infrastructure across environments.
- **Key Features:** Supports templates, stacks, change sets, and drift detection for efficient infrastructure management.
- **Example:** Use a YAML template to launch an EC2 instance, create an S3 bucket, and configure IAM roles in one deployment.

2.AWS Elastic Beanstalk

- **Definition**: AWS Elastic Beanstalk is a Platform as a Service (PaaS) that lets you deploy and manage applications without worrying about the underlying infrastructure.
- **Use Case**: Ideal for developers who want to quickly deploy web applications and services using familiar languages and frameworks.
- **Key Features**: Supports automatic provisioning, load balancing, scaling, and monitoring of applications.
- **Example**: Deploy a Python Flask app by uploading your code, and Elastic Beanstalk handles the rest—like EC2, ELB, and Auto Scaling.

---

3. AWS CodeCommit

- **Definition**: Fully managed source control service that hosts private Git repositories.
- **Features**:
  - Version control
  - Encrypted at rest and in transit
- **Use Case**: Store and manage application source code securely.
- **Example**: Push your app code to a CodeCommit repo instead of GitHub.

---

4. AWS CodeBuild

- **Definition**: Fully managed build service that compiles source code, runs tests, and produces artifacts.
- **Features**:
  - Continuous integration
  - Pay-as-you-go
- **Use Case**: Build and test code after every commit.
- **Example**: Compile a Java app and run unit tests automatically.

---

5. AWS CodeDeploy

- **Definition**: Automates code deployments to EC2, Lambda, or on-prem servers.
- **Features**:
  - Blue/Green and Rolling deployments
  - Monitors deployment health
- **Use Case**: Deploy new versions of your app with zero downtime.
- **Example**: Deploy a new version of a Node.js app to EC2 instances.

---

6. AWS CodePipeline

- **Definition**: CI/CD orchestration service that automates the build, test, and deploy phases.
- **Features**:
  - Integrates with CodeCommit, CodeBuild, CodeDeploy
  - Visual workflow
- **Use Case**: Automate the entire software release process.
- **Example**: CodePipeline triggers on Git push → builds with CodeBuild → deploys with CodeDeploy.

---

7. AWS CodeArtifact

- **Definition**: Managed artifact repository for software packages.
- **Supports**: npm, Maven, PyPI, NuGet
- **Use Case**: Store and share internal libraries and dependencies.
- **Example**: Host private npm packages for internal use.

8. AWS CDK (Cloud Development Kit)

- **Definition**: Define cloud infrastructure using familiar programming languages (TypeScript, Python, Java, etc.).
- **Use Case**: Infrastructure as Code (IaC) with logic and reuse.
- **Example**: Use Python to define an S3 bucket and Lambda function in a reusable stack.

9. AWS Systems Manager (SSM)

- **Definition**: AWS Systems Manager is a unified interface that helps you manage your AWS resources and automate operational tasks across AWS services.
- **Use Case**: Ideal for securely managing EC2 instances, automating patching, running commands remotely, and storing configuration data.
- **Key Features**: Includes Session Manager, Run Command, Patch Manager, Parameter Store, and Automation workflows.
- **Example**: Use SSM Run Command to remotely install software on a fleet of EC2 instances without SSH access.

10. SSM Session Manager

- **Definition**: Session Manager is a feature of AWS Systems Manager that allows secure, browser-based or CLI access to EC2 instances without using SSH or opening inbound ports.
- **Use Case**:
  Ideal for managing EC2 instances in private subnets or restricted environments without needing bastion hosts or SSH keys.
- **Key Features**:
  Provides audit logging, role-based access control, and encrypted sessions for secure instance management.
- **Example**: Use Session Manager to connect to a Linux EC2 instance from the AWS Console or CLI without needing an SSH key or public IP.

Module 10: Leveraging the AWS Global Infrastructure

1. Amazon Route 53

- **Definition**: Scalable and highly available **Domain Name System (DNS)** web service.
- **Key Features**:
  - Global DNS resolution
  - Latency-based routing
  - Health checks and failover
- **Use Cases**:
  - Route users to the **nearest AWS region** for low latency
  - Implement **disaster recovery** strategies
- **Example**: Route users in Europe to eu-west-1 and users in Asia to ap-southeast-1

---

2. Amazon CloudFront

- **Definition**:
  - Global **Content Delivery Network (CDN)**
  - Improves read performance, content cached at the edge
- **Key Features**:
  - Caches content at **Edge Locations**
  - Reduces latency and improves performance
- **Use Cases**:
  - Serve static assets (images, videos, JS/CSS)
  - Protect origin with signed URLs and WAF
  - DDoS Protection (because worldwide), integration with Shield, AWS Web Application Firewall
- **Example**: Deliver a website's images from the nearest edge location to the user

---

3. S3 Transfer Acceleration

- **Definition**: Speeds up uploads and downloads to S3 using AWS edge locations.
- **Use Case**: Improve performance for global users uploading large files to S3.
- **Example**: A user in India uploads a video to an S3 bucket in the US via the nearest edge location.

---

4. AWS Global Accelerator

- **Definition**: Uses the AWS global network to improve **availability and performance** of global applications.
- **Key Features**:
  - Static IP addresses
  - Intelligent routing to healthy endpoints
- **Use Case**: Real-time multiplayer games, global APIs
- **Example**: Route users to the fastest regional endpoint for a global API

---

5. AWS Outposts

- **Definition**: Extend AWS infrastructure and services to **on-premises data centers**.
- **Use Case**: Low-latency apps, data residency requirements
- **Example**: Run EC2 and RDS locally in your own data center using Outposts

---

6. AWS Wavelength

- **Definition**: Brings AWS services to the **edge of 5G networks**.
- **Use Case**: Ultra-low latency applications like AR/VR, gaming, real-time analytics
- **Example**: A mobile game hosted on Wavelength for sub-10ms latency

---

7. AWS Local Zones

- **Definition**: Extend AWS infrastructure closer to users in metro areas.
- **Use Case**: Latency-sensitive applications like video editing, gaming, ML inference
- **Example**: Deploy EC2 and EBS in a Local Zone in Los Angeles for a video production company

Module 11: Cloud Integration

1. Amazon SQS (Simple Queue Service)

- **Definition**: Fully managed message queuing service.
- **Key Features**:
  - Decouples microservices, distributed systems, and serverless apps
  - Messages retained for up to **14 days**
  - **Multiple producers** can send messages
  - **Multiple consumers** can read and delete messages
- **Use Case**: Buffering orders between a web app and a payment processor
- **Example**:
  - **Producer:** Web app sends order data to SQS
  - **Consumer:** Lambda function processes orders from the queue

---

2. Amazon SNS (Simple Notification Service)

- **Definition**: Fully managed **pub/sub** messaging service.
- **Key Features**:
  - **Multiple subscribers** receive all messages
  - No message retention (messages are pushed instantly)
  - Supports multiple protocols: Email, SMS, Lambda, SQS, HTTP/S
- **Use Case**: Send alerts to multiple systems when a new user signs up
- **Example**:
  - **Topic:** NewUserSignup
  - **Subscribers:** Email (admin), Lambda (analytics), SQS (logging)

---

3. Amazon Kinesis

- **Definition**:
  - Real-time big data streaming platform.
  - Managed services to collect, process, and analyze real-time streaming data at any scale
- **Key Features**:
  - Ingest, process, and analyze streaming data
  - Supports real-time dashboards, analytics, and ML
- **Use Case**: Monitor website clickstream data in real time
- **Example**:
  - **Stream:** ClickEvents
  - **Consumers:** Lambda for processing, Redshift for analytics

---

4. Amazon MQ

- **Definition**: Managed message broker service for **ActiveMQ** and **RabbitMQ**.
- **Key Features**:
  - Supports standard messaging protocols: **AMQP**, **MQTT**, **STOMP**, etc.
  - Ideal for migrating legacy applications to AWS
- **Use Case**: Enterprise apps using traditional message brokers
- **Example**:
  - On-prem app using RabbitMQ migrates to Amazon MQ with minimal changes

Module 12: Cloud Monitoring

1. Amazon CloudWatch Metrics

- **Definition**: Collects and monitors performance data from AWS services and custom applications.
- **Examples**:
  - EC2 CPU utilization
  - S3 bucket size
  - Billing metrics (e.g., estimated charges)

1.1. Alarms

- **Definition**: Trigger actions based on metric thresholds.
- **Use Cases**:
  - Send notifications via SNS
  - Auto-reboot or stop EC2 instances
- **Example**: Alarm triggers if CPU > 80% for 5 minutes

1.2. Logs

- **Definition**: Collects and stores logs from EC2, Lambda, and other sources.
- **Use Case**: Centralized log management and analysis
- **Example**: View application logs from a Lambda function

1.3. Events / EventBridge

- **Definition**: Respond to AWS events or schedule tasks.
- **Use Case**:
  - Trigger Lambda on EC2 state change
  - Run a job every day at midnight
- **Example**: Automatically start EC2 instance at 8 AM daily

---

2. AWS CloudTrail

- **Definition**: Records all API calls made in your AWS account.
- **Use Case**: Security auditing, compliance, and troubleshooting
- **Example**: Track who deleted an S3 bucket

CloudTrail Insights

- **Definition**: Detects unusual activity in API usage.
- **Use Case**: Identify spikes in failed login attempts or access anomalies
- **Example**: Alert when there's a sudden increase in DescribeInstances calls

---

3. AWS X-Ray

- **Definition**: Traces requests through distributed applications.
- **Use Case**: Debug latency issues and visualize service maps
- **Example**: Trace a user request through API Gateway → Lambda → DynamoDB

---

4. AWS Health Dashboard

AWS Health Dashboard

- **Definition**: Shows the status of AWS services across all regions.
- **Use Case**: Check for ongoing outages or service disruptions
- **Example**: View if S3 is experiencing issues in us-east-1

AWS Account Health Dashboard

- **Definition**: Personalized view of AWS events affecting your resources.
- **Use Case**: Monitor events like EC2 maintenance or service limits
- **Example**: Notification about upcoming RDS maintenance in your account

---

5. Amazon CodeGuru

- **Definition**: Uses ML to provide code reviews and performance recommendations.
- **Use Case**: Improve code quality and application efficiency

- **Example**: Detects inefficient loops or recommends better database access patterns
- **CodeGuru Reviewer**: Automatic code reviews for static code analysis **(Development)**
- **CodeGuru Profiler**:
  Visibility/recommendation about application performance during runtime **(Production)**

Module 13: VPC & Networking

1. VPC Endpoints

- **Definition**: Enable private connectivity to AWS services **without using the public internet.**
- **Types**:
  - **Interface Endpoints**: ENIs for services like S3, DynamoDB
  - **Gateway Endpoints**: For S3 and DynamoDB only
- **Use Case**: Access S3 privately from within your VPC
- **Example**: EC2 instance in a private subnet accesses S3 via a Gateway Endpoint

---

2. AWS PrivateLink

- **Definition**: Securely connect to services **hosted in another VPC** (even third-party or SaaS) using private IPs.
- **Use Case**: Access a partner's service without exposing traffic to the internet
- **Example**: Connect to a payment gateway provider's VPC privately

---

3. VPC Flow Logs

- **Definition**: Capture information about **IP traffic** going to and from network interfaces in your VPC.
- **Use Case**: Network monitoring, troubleshooting, and security analysis
- **Example**: Analyze traffic patterns or detect suspicious activity

---

4. Site-to-Site VPN

- **Definition**: Secure **IPSec VPN connection** between your on-premises data center and AWS VPC over the public internet.
- **Use Case**: Extend your data center into the cloud
- **Example**: Connect your corporate network to AWS for hybrid cloud setup

---

5. Client VPN

- **Definition**: **OpenVPN-based** secure connection from a user's device to AWS.
- **Use Case**: Remote employees securely access AWS resources
- **Example**: Developer connects to a private EC2 instance from home

---

6. AWS Direct Connect

- **Definition**: Dedicated **private network connection** from your premises to AWS.
- **Use Case**: High-throughput, low-latency, and secure workloads
- **Example**: Financial firm transfers large datasets to AWS daily

---

7. AWS Transit Gateway

- **Definition**: Central hub to **connect multiple VPCs and on-prem networks**.
- **Use Case**: Simplify and scale network architecture
- **Example**: Connect 50+ VPCs and a data center using a single Transit Gateway

Module 14: Security & Compliance

1. Shared Responsibility Model

- **AWS**: Secures the infrastructure (hardware, software, networking, facilities).
- **Customer**: Secures data, identity, applications, and configurations.
- **Example**: AWS secures the physical servers; you configure IAM roles and S3 permissions.

---

2. DDoS Protection

- **Definition:** A **DDoS attack** when multiple systems flood a target (like website or server) with traffic to overwhelm it and make it unavailable to user
- **AWS Shield**:
  - **Standard**: Automatic protection against common DDoS attacks
  - **Advanced**: 24/7 support, real-time metrics, and financial protections
- **Use Case**: Protect public-facing apps like websites or APIs

---

3. Web Application Firewall (WAF)

- **Definition**: Protects web apps from common exploits (SQL injection, XSS).
- **Use Case**: Filter HTTP requests based on rules
- **Example**: Block requests from specific IPs or countries

---

4. Encryption Services

- **AWS KMS (Key Management Service)**: Manage encryption keys (AWS or customer-managed)
- **AWS CloudHSM**: Hardware-based key storage, customer controls keys
- **AWS Certificate Manager**: Provision and manage SSL/TLS certificates
- **Use Case**: Encrypt S3 data, secure HTTPS websites

---

5. Compliance & Auditing

- **AWS Artifact**: Access compliance reports (e.g., PCI, ISO, SOC)
- **AWS Config**: Track configuration changes and evaluate compliance
- **AWS CloudTrail**: Log and monitor all API calls in your account
- **CloudTrail Insights**: Detect unusual API activity
- **Use Case**: Prove compliance during audits, detect unauthorized changes

---

6. Threat Detection & Analysis

- **Amazon GuardDuty**: Detects threats using VPC, DNS, and CloudTrail logs
- **Amazon Inspector**: Scans EC2, Lambda, and ECR for vulnerabilities
- **Amazon Detective**: Investigate and visualize security issues
- **AWS Security Hub**: Centralized view of security alerts across AWS accounts
- **Use Case**: Detect malware, misconfigurations, and suspicious behavior

---

7. Network Protection

- **AWS Network Firewall**: Protects VPCs with customizable rules
- **AWS Firewall Manager**: Centralized management of WAF, Shield, and security groups across accounts
- **Use Case**: Enforce consistent firewall rules across an organization

---

8. Data Protection & Discovery

- **Amazon Macie**: Uses ML to discover and protect sensitive data (e.g., PII) in S3
- **Use Case**: Identify and secure exposed personal data

---

9. Root User Privileges

- **High-risk actions only root can perform**:
  - Change account settings
  - Close AWS account

- Modify support plans
- Register as a Reserved Instance seller

- **Best Practice**: Avoid using root user for daily tasks

---

10. IAM Access Analyzer

- **Definition**: Identifies resources shared externally (e.g., S3 buckets, IAM roles)
- **Use Case**: Audit and reduce unintended public or cross-account access

---

11. AWS Abuse Reporting

- **Purpose**: Report AWS resources being used for malicious or illegal activity
- **Example**: Report phishing or DDoS attacks originating from AWS IPs

Module 15: AWS Machine Learning

1. Amazon Rekognition

- **Definition**: Image and video analysis service.
- **Features**:
  - Face detection and comparison
  - Object and scene labeling
  - Celebrity recognition
- **Use Case**: Tagging photos, verifying identities
- **Example**: Detect faces in a photo and label objects like "car", "tree", "person"

---

2. Amazon Transcribe

- **Definition**: Converts **speech to text**.
- **Use Case**: Generate subtitles, transcribe customer service calls
- **Example**: Convert a podcast episode into a text transcript

---

3. Amazon Polly

- **Definition**: Converts **text to lifelike speech**.
- **Use Case**: Voice assistants, audiobooks, accessibility tools
- **Example**: Read out news articles using natural-sounding voices

---

4. Amazon Translate

- **Definition**: Real-time **language translation**.
- **Use Case**: Multilingual websites, customer support
- **Example**: Translate product descriptions from English to Spanish

---

5. Amazon Lex

- **Definition**: Build **conversational chatbots** using voice and text.
- **Use Case**: Customer service bots, virtual assistants

- **Example**: Create a chatbot that books appointments via chat or voice

---

6. Amazon Connect

- **Definition**: Cloud-based **contact center** service.
- **Use Case**: Handle customer support calls with AI-powered routing
- **Example**: Set up a call center with Lex-powered voice bots

---

7. Amazon Comprehend

- **Definition**: **Natural Language Processing (NLP)** service.
- **Features**:
  - Sentiment analysis
  - Entity recognition
  - Language detection
- **Use Case**: Analyze customer feedback or social media posts
- **Example**: Detect positive or negative sentiment in product reviews

---

8. Amazon SageMaker

- **Definition**: End-to-end platform for building, training, and deploying ML models.
- **Use Case**: Custom ML solutions for developers and data scientists
- **Example**: Train a fraud detection model using tabular data

---

9. Amazon Kendra

- **Definition**: **Intelligent search engine** powered by ML.
- **Use Case**: Enterprise document search
- **Example**: Search across PDFs, Word docs, and wikis with natural language queries

---

10. Amazon Personalize

- **Definition**: Real-time **recommendation engine**.
- **Use Case**: Product or content recommendations
- **Example**: Suggest movies or products based on user behavior

---

11. Amazon Textract

- **Definition**: Extracts **text and structured data** from scanned documents.
- **Use Case**: Digitize forms, invoices, and receipts
- **Example**: Extract names and totals from scanned invoices

Module 16: Account Managament, Billing & Support

1. AWS Compute Optimizer

- **Definition**: Recommends optimal AWS resource configurations to reduce cost and improve performance.
- **Use Case**: Identify underutilized EC2 instances or over-provisioned EBS volumes.
- **Example**: Suggests changing an EC2 instance from m5.large to t3.medium.

---

2. AWS Pricing Calculator

- **Definition**: Estimate the cost of AWS services before using them.
- **Use Case**: Plan budgets for new projects or compare service costs.
- **Example**: Calculate monthly cost for 3 EC2 instances, 1 RDS database, and 100 GB S3 storage.

---

3. Billing Dashboard

- **Definition**: High-level overview of your AWS spending.
- **Features**:
  - Free tier usage tracking
  - Monthly spend summary
- **Example**: View current month's spend and remaining free tier usage.

---

4. Cost Allocation Tags

- **Definition**: Tag AWS resources to track costs by project, team, or environment.
- **Use Case**: Generate cost reports by department or application.
- **Example**: Tag EC2 instances with Project=Marketing and Environment=Dev.

---

5. Cost and Usage Reports (CUR)

- **Definition**: Most detailed billing dataset available in AWS.
- **Use Case**: Deep dive into usage patterns and cost breakdowns.

- **Example**: Analyze hourly EC2 usage and associated costs across accounts.

---

6. AWS Cost Explorer

- **Definition**: Visualize and analyze AWS spending over time.
- **Features**:
  - Forecast future costs
  - Filter by service, region, or tag
- **Example**: View daily S3 costs and forecast next month's bill.

---

7. Billing Alarms (CloudWatch)

- **Definition**: Set alarms for billing thresholds (only in us-east-1).
- **Use Case**: Get notified when spending exceeds a set amount.
- **Example**: Alert when monthly bill exceeds $100.

---

8. AWS Budgets

- **Definition**: Advanced tool to track usage, costs, and Reserved Instance (RI) utilization.
- **Features**:
  - Alerts via email or SNS
  - Supports cost, usage, and RI coverage tracking
- **Example**: Notify finance team if EC2 usage exceeds budgeted amount.

---

9. Savings Plans

- **Definition**: Flexible pricing model to save up to 72% over On-Demand pricing.
- **Types**:
  - **Compute Savings Plans**: Flexible across instance families and regions
  - **EC2 Instance Savings Plans**: Specific to instance family and region
- **Example**: Commit to $100/month usage for 1 year to save on EC2 and Fargate.

---

10. Cost Anomaly Detection

- **Definition**: Uses ML to detect unusual spending patterns.
- **Use Case**: Identify unexpected spikes in usage or cost.
- **Example**: Alert when Lambda usage suddenly triples in a day.

---

11. Service Quotas

- **Definition**: Manage and monitor AWS service limits.
- **Use Case**: Prevent service disruptions due to quota limits.
- **Example**: Get notified when EC2 instance limit is 90% used.

Module 17: Advanced Identity

1. IAM (Identity and Access Management)

- **Definition**: Manage access to AWS services and resources securely.
- **Use Case**: Grant permissions to users, groups, and roles within your AWS account.
- **Example**: Create an IAM user devops-admin with full access to EC2 and S3.

---

2. AWS Organizations

- **Definition**: Manage and govern multiple AWS accounts centrally.
- **Features**:
  - Consolidated billing
  - Service control policies (SCPs)
- **Use Case**: Separate accounts for dev, test, and prod environments.
- **Example**: Apply a policy to restrict S3 access in all dev accounts.

---

3. AWS STS (Security Token Service)

- **Definition**: Issue **temporary, limited-privilege credentials** for AWS access.
- **Use Case**: Grant short-term access to third-party users or federated identities.
- **Example**: Provide a contractor with 1-hour access to a specific S3 bucket.

---

4. Amazon Cognito

- **Definition**: User identity and authentication service for web and mobile apps.
- **Features**:
  - User pools (sign-up/sign-in)
  - Federated identities (Google, Facebook, etc.)
- **Use Case**: Add user login to a mobile app.
- **Example**: Authenticate users via Google and store profiles in Cognito.

---

5. AWS Directory Service

- **Definition**: Integrate **Microsoft Active Directory** with AWS.
- **Use Case**: Enable domain join for EC2 Windows instances or use AD for authentication.
- **Example**: Use AD credentials to log into EC2 Windows servers.
- **Directory Services**:
    - **AWS Managed Microsoft AD:** Create your own AD in AWS, manage users locally supports MFA
    - **AD Connector:** Directory Gateway (Proxy) to redirect to on-premise AD, supports MFA
    - **Simple AD:** AD-Compatible managed directory on AWS

---

6. IAM Identity Center (formerly AWS SSO)

- **Definition**: Centralized access management for multiple AWS accounts and applications.
- **Features**:
    - Single sign-on (SSO)
    - Integrates with external identity providers (e.g., Okta, Azure AD)
- **Use Case**: One login for developers to access multiple AWS accounts.
- **Example**: Assign a user access to dev-account and prod-account with different roles.

Module 18: Other AWS Services

1. Amazon Workspaces

- **Definition**: Managed, secure **Desktop-as-a-Service (DaaS)**.
- **Use Case**: Provide remote desktops to employees.
- **Example**: Deploy Windows desktops for a remote development team.

---

2. Amazon AppStream 2.0

- **Definition**: Stream desktop applications to users via a browser.
- **Use Case**: Deliver high-performance apps without local installs.
- **Example**: Stream CAD software to students in a virtual lab.

---

3. AWS IoT Core

- **Definition**: Connect and manage **IoT devices** securely.
- **Use Case**: Collect sensor data from smart devices.
- **Example**: Monitor temperature sensors in a smart building.

---

4. Elastic Transcoder

- **Definition**: Convert media files into different formats.
- **Use Case**: Prepare videos for mobile, web, and TV playback.
- **Example**: Convert uploaded videos to MP4 and WebM formats.

---

5. AWS AppSync

- **Definition**: Managed **GraphQL API** service.
- **Use Case**: Build real-time, offline-capable mobile/web apps.
- **Example**: Sync chat messages across devices in real time.

---

6. AWS Amplify

- **Definition**: Full-stack development platform for web and mobile apps.
- **Use Case**: Quickly build and deploy serverless apps.
- **Example**: Deploy a React app with authentication and GraphQL backend.

---

7. AWS Infrastructure Composer

- **Definition**: Visual tool to design and deploy cloud infrastructure.
- **Use Case**: Drag-and-drop interface for building AWS architectures.
- **Example**: Design a serverless app with Lambda, API Gateway, and DynamoDB.

---

8. AWS Device Farm

- **Definition**: Test mobile and web apps on real devices in the cloud.
- **Use Case**: Ensure app compatibility across devices.
- **Example**: Run automated UI tests on Android and iOS devices.

---

9. AWS Backup

- **Definition**: Centralized backup service for AWS resources.
- **Use Case**: Automate backups for EC2, RDS, EFS, DynamoDB, etc.
- **Example**: Daily backup of RDS databases with 30-day retention.

---

10. Disaster Recovery Strategies

- **Types**:
  - Backup & Restore
  - Pilot Light
  - Warm Standby
  - Multi-Site Active-Active
- **Use Case**: Ensure business continuity during outages.

---

11. AWS DataSync

- **Definition**: Transfer data between on-prem and AWS storage.
- **Use Case**: Migrate large datasets to S3 or EFS.
- **Example**: Sync 10 TB of logs from on-prem NAS to S3.

---

12. Cloud Migration Strategies – 7Rs

- **Strategies**:
  - **Rehost ("Lift and Shift")**:
    - Move applications to the cloud **without changes**.
    - Fastest method, often used for large-scale migrations.
  - **Replatform ("Lift, Tinker, and Shift")**
    - Make **minor optimizations** (e.g., change database engine) without changing the core architecture.
  - **Repurchase**
    - Move to a **different product**, typically a SaaS solution (e.g., replacing a self-hosted CRM with Salesforce).
  - **Refactor / Re-architect**
    - **Redesign the application** to take full advantage of cloud-native features (e.g., microservices, serverless).
  - **Retire**
    - **Decommission** applications that are no longer needed or used.
  - **Retain**
    - Keep some applications **on-premises** due to compliance, complexity, or low ROI for migration.
  - **Relocate**
    - Move applications **without purchasing new hardware**, often used in VMware Cloud on AWS scenarios.
- **Use Case**: Plan cloud migration based on app needs.

---

13. AWS Migration Evaluator

- **Definition:**
  - Assess on-prem workloads and estimate AWS costs.
  - Provides insights and recommendations to help you build your migration and modernization business case.
- **Use Case**: Build a business case for migration.
- **Example**: Evaluate 100 VMs for migration to EC2.

---

14. AWS Migration Hub

- **Definition**: Central dashboard to track migration progress.
- **Use Case**: Monitor server and database migrations.
- **Example**: Track status of 20 database migrations to RDS.

---

15. AWS Fault Injection Simulator (FIS)

- **Definition**: Run **chaos engineering** experiments.
- **Use Case**: Test app resilience by simulating failures.
- **Example**: Simulate EC2 instance termination to test failover.

---

16. AWS Step Functions

- **Definition**: Serverless orchestration for workflows.
- **Use Case**: Coordinate Lambda functions and services.
- **Example**: Order processing workflow with payment, shipping, and notifications.

---

17. AWS Ground Station

- **Definition**: Ingest satellite data directly into AWS.
- **Use Case**: Process satellite imagery in real time.
- **Example**: Receive weather satellite data and analyze with SageMaker.

---

18. Amazon Pinpoint

- **Definition**: Targeted **user engagement** service.
- **Use Case**: Send emails, SMS, push notifications.
- **Example**: Send promotional emails to app users based on behavior.

Module 19: AWS Architecting Ecosystem

1. AWS Well-Architected Framework

A set of best practices to help design and operate reliable, secure, efficient, and cost-effective systems in the cloud.

The 6 Pillars:

1. **Operational Excellence**
    - **Focus:** Monitoring, incident response, and continuous improvement
    - **Example:** Use CloudWatch and AWS Config for proactive operations
2. **Security**
    - **Focus:** Protecting data, systems, and assets
    - **Example:** Use IAM, KMS, and GuardDuty for layered security
3. **Reliability**
    - **Focus:** Recover from failures and meet customer demands
    - **Example:** Use Multi-AZ deployments and Route 53 failover
4. **Performance Efficiency**
    - **Focus:** Use resources efficiently as demand changes
    - **Example:** Use Auto Scaling and AWS Lambda for dynamic workloads
5. **Cost Optimization**
    - **Focus:** Avoid unnecessary costs
    - **Example:** Use Compute Optimizer and Savings Plans
6. **Sustainability**
    - **Focus:** Minimize environmental impact
    - **Example:** Use Graviton processors and monitor carbon footprint

---

2. AWS Well-Architected Tool

- **Definition**: Evaluate workloads against the 6 pillars. (Free tool)
- **Use Case**: Identify risks and get improvement plans.
- **Example**: Run a review for your production workload and get actionable insights.

---

3. Customer Carbon Footprint Tool

- **Definition**: Helps you estimate the carbon emissions associated with your AWS usage and identify ways to reduce your carbon footprint.
- **Use Case**: Support sustainability goals.
- **Example**: View monthly emissions by service and region.

---

4. AWS CAF (Cloud Adoption Framework)

- **Definition**: Provides guidance for cloud transformation across six perspectives:
  - Business, People, Governance, Platform, Security, Operations
- **Use Case**: Plan and execute cloud migration and modernization.

---

5. Right Sizing

- **Definition**: Match instance types and sizes to workload needs.
- **Use Case**: Reduce cost and improve performance.
- **Example**: Replace underutilized m5.large with t3.medium.

---

6. AWS Ecosystem

- **Definition**: The broad set of AWS services, tools, partners, and community resources that help customers build, deploy and manage applications in the cloud.
- **Use Case**: Build, deploy, and manage cloud-native applications.

---

7. AWS IQ & re:Post

- **AWS IQ**: A Service that connects you with AWS-Certified experts for on-demand project work
- **AWS re:Post**: Community-driven Q&A platform for AWS topics.

---

8. AWS Knowledge Center

- **Definition**: Official AWS FAQs and troubleshooting guides.
- **Use Case**: Quickly resolve common issues.

---

9. AWS Managed Services (AMS)

- **Definition**: Operate AWS infrastructure on behalf of enterprise customers.
- **Use Case**:
  - Offload day-to-day operations like patching, monitoring, and backups.
  - It helps to reduce operational overhead and risk.