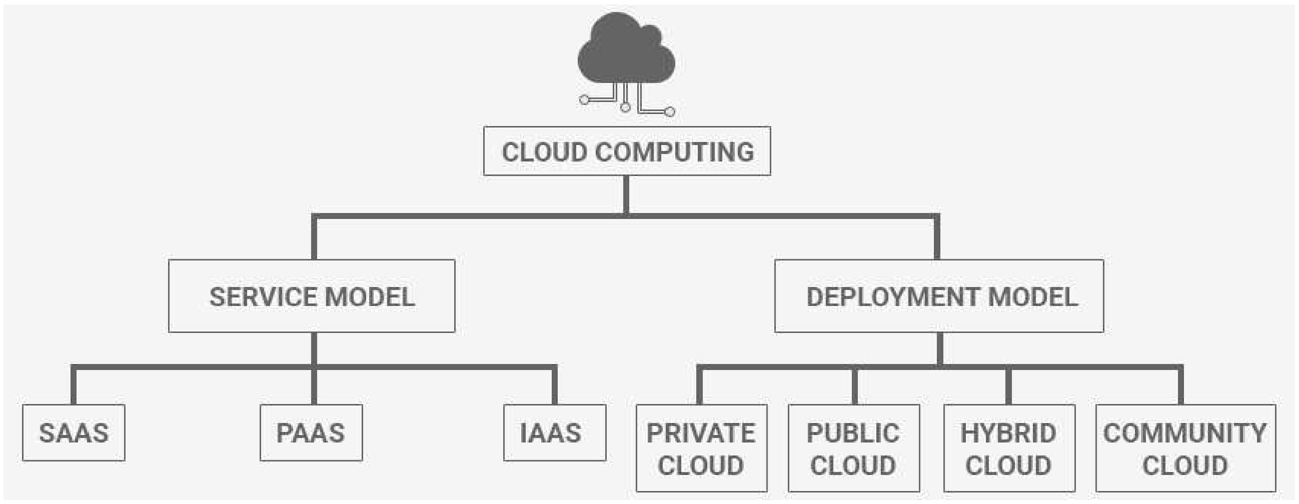
# CLOUD COMPUTING

Cloud computing is the delivery of computing services over the internet ("the cloud"), enabling faster innovation, flexible resources, and economies of scale. These services typically include computing power, storage, databases, networking, software, analytics, and intelligence.

## Key Characteristics of Cloud Computing

- **On-Demand Self-Service**: Users can provision computing resources automatically without human interaction with the provider.
- **Broad Network Access**: Resources are accessible over the network via standard mechanisms, usable by various client platforms.
- **Resource Pooling**: Providers' resources are pooled to serve multiple consumers, dynamically assigned and reassigned as needed.
- **Rapid Elasticity**: Capabilities can be automatically scaled in or out quickly to meet demand.
- **Measured Service**: Resource use is automatically controlled and optimized through metering.



## Types of Cloud Computing Services

**Main Types of Cloud Computing**

### 1. Software as a Service (SaaS)

- **Purpose**: Provides software applications over the internet on a subscription basis.
- **Characteristics**: Users access applications via web browsers or APIs, with no need for infrastructure management.
- **Examples**: Gmail, Slack, Salesforce, Dropbox, Microsoft Office 365, Zoom, Atlassian Jira.

### 2. Platform as a Service (PaaS)

- **Purpose**: Offers a platform to develop, deploy, and manage applications without managing the underlying infrastructure.
- **Characteristics**: Provides tools and services for application development, including databases and middleware.
- **Examples**: Microsoft Azure App Services, Google App Engine, Heroku, Red Hat OpenShift, IBM Cloud Foundry, Salesforce App Cloud.

### 3. Infrastructure as a Service (IaaS)

- **Purpose**: Provides virtualized computing resources over the internet, including virtual machines, storage, and networking.
- **Characteristics**: Users manage the operating systems and applications, while the provider manages the hardware.
- **Examples**: AWS EC2, Microsoft Azure Virtual Machines, Google Compute Engine, IBM Cloud Infrastructure, DigitalOcean Droplets, Linode.

## Additional Service Models

☐ **Desktop as a Service (DaaS)**

- **Purpose**: Delivers virtual desktop environments over the internet.
- **Characteristics**: Provides access to a desktop interface from any device, managed by the provider.
- **Examples**: Amazon WorkSpaces, VMware Horizon Cloud, Citrix Virtual Apps and Desktops, Microsoft Windows Virtual Desktop, Cloudalize.

☐ **Identity as a Service (IDaaS)**

- **Purpose**: Provides cloud-based identity and access management, including authentication and user management.
- **Characteristics**: Manages and secures user identities, often integrating with various applications and systems.
- **Examples**: Okta, Azure Active Directory, OneLogin, Auth0, IBM Security Verify, Ping Identity.

☐ **Function as a Service (FaaS)**

- **Purpose**: Executes individual functions in response to events without managing server infrastructure.
- **Characteristics**: Automatically scales based on the number of requests, with users only paying for the execution time.
- **Examples**: AWS Lambda, Google Cloud Functions, Azure Functions, IBM Cloud Functions, Alibaba Cloud Function Compute, Oracle Functions.

## Deployment Models

### Public Cloud

- **Description**: Cloud services owned and operated by third-party vendors that deliver computing resources like storage and databases over the internet.
- **Examples**: Amazon EC2 (virtual servers), Amazon S3 (storage), Amazon RDS (relational database service), AWS, Microsoft Azure, Google Cloud Platform.

### Private Cloud

- **Description**: Cloud services dedicated to a single organization, often hosted on-premises or through a third-party provider with dedicated hardware.
- **Examples**: AWS Outposts (private cloud extension of AWS), VMware vSphere (virtualization platform for private clouds), OpenStack, Microsoft Azure Stack.

### Hybrid Cloud

- **Description**: A combination of public and private clouds, allowing data and applications to be shared between them.
- **Examples**: AWS Direct Connect (connects on-premises environments to AWS), Microsoft Azure Arc (manage and extend Azure services to on-premises and other clouds),AWS Outposts, Google Anthos.

### Community Cloud

- **Description**: Cloud infrastructure shared by several organizations with common concerns, such as security or compliance, often managed by a third party.

- **Examples**: Government cloud initiatives, healthcare consortium clouds.

**Multi-Cloud**

- **Description**: The use of multiple cloud computing services from different providers to avoid vendor lock-in and leverage the best services from each.
- **Examples**: Using AWS for infrastructure and Google Cloud for machine learning, or combining Microsoft Azure with IBM Cloud for different services.

## Benefits of Cloud Computing

1. **Cost Efficiency**: Reduces capital expenditure (CapEx) and converts it to operational expenditure (OpEx), lowering the cost of IT infrastructure.
2. **Scalability and Flexibility**: Easily scales resources up or down based on demand.
3. **Accessibility**: Provides access to data and applications from anywhere with an internet connection.
4. **Disaster Recovery and Business Continuity**: Offers robust backup and recovery solutions.
5. **Collaboration**: Enhances collaboration through shared access to documents and applications.
6. **Automatic Updates**: Ensures that users have the latest versions of applications and security patches.

# AWS SERVICES

## Compute

**Amazon EC2 (Elastic Compute Cloud)**

- **Description**: Provides resizable compute capacity in the cloud using virtual servers called instances.
- **Use Case**: Running applications, hosting web servers, or performing computational tasks.
- **Example**: Hosting a website or web application on a virtual server that can scale based on demand.
- **Example**: A startup runs its e-commerce website on EC2 instances to handle traffic spikes during sales events. They use auto-scaling to increase the number of instances during high traffic periods and reduce them during low traffic.

**AWS Lambda**

- **Description**: A serverless compute service that runs code in response to events without provisioning or managing servers.
- **Use Case**: Automating tasks, such as processing files or running backend services without managing infrastructure.
- **Example**: Automatically processing images uploaded to S3 by triggering a Lambda function.

## Storage

**Amazon S3 (Simple Storage Service)**

- **Description**: Scalable object storage service for storing and retrieving any amount of data.

- **Use Case**: Backup, archiving, and serving static assets like images, videos, and website files.
- **Example**: Storing and serving images for a website or storing backups of databases.
- **Example**: A media company stores and serves thousands of high-resolution images for their online photo gallery using Amazon S3. They use S3 to handle the storage and retrieval of images with low latency and high durability.

#### Amazon EBS (Elastic Block Store)

- **Description**: Provides block storage volumes for use with EC2 instances.
- **Use Case**: Storing data for applications that require consistent, low-latency access to block storage.
- **Example**: Attaching EBS volumes to EC2 instances for application data storage and database use.

## Networking

#### Amazon VPC (Virtual Private Cloud)

- **Description**: Creates a private network within the AWS cloud to launch AWS resources in a virtual network.
- **Use Case**: Isolating resources, controlling network access, and ensuring secure communication between resources.
- **Example**: Setting up a private subnet for database instances and a public subnet for web servers.
- **Example**: A healthcare organization uses Amazon VPC to create a private network within AWS for its electronic health record (EHR) system. They isolate their database instances in a private subnet while exposing their application servers in a public subnet for secure access.

## Monitoring and Management

**Amazon CloudWatch**

- **Description**: Monitors and manages AWS resources and applications, providing metrics, logs, and alarms.
- **Use Case**: Tracking performance metrics, setting up alarms for resource utilization, and logging application activities.
- **Example**: Monitoring EC2 instance CPU utilization and setting up alarms for high usage.
- **Example**: An application development team uses Amazon CloudWatch to monitor the performance of their web application. They set up alarms to notify them when CPU utilization exceeds 80% on their EC2 instances, ensuring they can address performance issues proactively.

## Migration and Transfer

**AWS Snowball**

- **Description**: A data transport solution that uses physical devices to transfer large amounts of data to and from AWS.
- **Use Case**: Migrating large datasets to AWS when network transfer is impractical or time-consuming.
- **Example**: Using Snowball to transfer terabytes of data from an on-premises data center to Amazon S3.
- **Example**: A large research institution uses AWS Snowball to transfer petabytes of data from their on-premises data center to Amazon S3. This is done to facilitate the analysis of large datasets for scientific research, which would be too slow over the internet.

**AWS Snowcone**

- **Description**: A small, rugged, and portable edge computing and data transfer device.
- **Use Case**: Collecting and processing data at the edge and transferring it to AWS when network connectivity is limited.
- **Example**: Using Snowcone to collect and process data from IoT devices in a remote location before transferring it to AWS.

#### AWS Data Migration Service (DMS)

- **Description**: Facilitates the migration of databases to AWS with minimal downtime.
- **Use Case**: Moving data from on-premises databases or other cloud databases to AWS.
- **Example**: Migrating a SQL Server database to Amazon RDS with minimal disruption to ongoing operations.

## Media Services

#### AWS Elastic Transcoder

- **Description**: Converts media files from their source format into formats suitable for playback on various devices.
- **Use Case**: Preparing video content for streaming on different devices and formats.
- **Example**: Transcoding video files for compatibility with mobile devices and web browsers.
- **Example**: A video streaming service uses AWS Elastic Transcoder to convert user-uploaded videos into multiple formats and resolutions for streaming on different devices, including smartphones, tablets, and smart TVs.

## Database

#### Amazon RDS (Relational Database Service)

- **Description**: Managed relational database service that supports various database engines.
- **Use Case**: Hosting and managing relational databases for applications with automated backups, patching, and scaling.
- **Example**: Running a MySQL database for a web application with automatic backups and scaling capabilities.

#### Amazon DynamoDB

- **Description**: Managed NoSQL database service offering fast and flexible performance with seamless scalability.
- **Use Case**: Storing and retrieving large amounts of data with low-latency access for web and mobile applications.
- **Example**: Using DynamoDB to store user session data and application state for a high-traffic website.

## Security and Identity

#### AWS IAM (Identity and Access Management)

- **Description**: Manages access to AWS services and resources securely.
- **Use Case**: Controlling user permissions and managing access to AWS resources.
- **Example**: Creating IAM roles for different teams with specific permissions for accessing AWS services.

## Analytics

#### Amazon Redshift

- **Description**: Data warehousing service designed for large-scale data analysis and reporting.
- **Use Case**: Performing complex queries and data analysis on large datasets.

- **Example**: Running analytics queries on customer data to generate business intelligence reports.

## Developer Tools

#### AWS CodeBuild

- **Description**: Continuous integration service that compiles source code, runs tests, and produces software packages.
- **Use Case**: Automating the build and test process in a continuous integration/continuous deployment (CI/CD) pipeline.
- **Example**: Building and testing application code every time changes are pushed to a source code repository.

## Machine Learning Services

### Amazon SageMaker

- **Use Case**: Building, training, and deploying machine learning models.
- **Example**: Developing a recommendation engine for an e-commerce platform.

### AWS Rekognition

- **Use Case**: Image and video analysis.
- **Example**: Detecting objects, people, text, scenes, and activities in images and videos uploaded by users.

## AWS CLI

The **Command Line Interface (CLI)** allows users to interact with software or operating systems using text commands. AWS CLI is a tool provided by

Amazon Web Services (AWS) that enables users to manage AWS services through the command line.

**AWS CLI Overview**

**Description**

- **AWS CLI**: A unified tool to manage AWS services from the command line. It provides a consistent interface for interacting with AWS services and automating tasks.

**Use Case**

- **Automation**: Automate repetitive tasks such as launching EC2 instances, creating S3 buckets, or managing IAM users.
- **Scripting**: Write scripts to manage AWS resources and perform batch operations.
- **Remote Management**: Manage AWS resources from any terminal without needing the AWS Management Console.

## INSTANCES

**Description**:

- Instances are virtual servers in the cloud provided by Amazon EC2 (Elastic Compute Cloud). They can run various operating systems and applications, and their configurations can be customized based on the required performance, storage, and other specifications.

**Use Case**:

- **Web Hosting**: Running web servers to host websites and applications.
- **Application Hosting**: Deploying and running enterprise applications, databases, and other services.

- **Development and Testing**: Providing environments for development and testing without needing physical hardware.

**Examples**:

- **Web Application**: Hosting a website or web application on EC2 instances.
- **Data Processing**: Running data processing jobs on compute-optimized EC2 instances.
- **Database Hosting**: Setting up a relational or NoSQL database on EC2 instances for an application.

## KEY PAIRS

**Description**:

- Key pairs are used for secure SSH access to EC2 instances. Each key pair consists of a public key and a private key. The public key is placed on the EC2 instance, and the private key is kept by the user. The private key is used to securely connect to the instance via SSH.

**Use Case**:

- **Secure Access**: Provides secure, encrypted access to EC2 instances for management and administration.
- **Authentication**: Ensures that only users with the corresponding private key can connect to the EC2 instance.

**Examples**:

- **Connecting to an EC2 Instance**: Using the private key to establish an SSH connection to an EC2 instance.
- **Deploying Applications**: Logging into an EC2 instance to deploy and manage applications.

## AWS AUTO SCALING

**Description**:

- AWS Auto Scaling automatically adjusts the number of Amazon EC2 instances or other AWS resources in response to changing demand. It ensures that you have the right amount of resources available to handle your application load efficiently.

**Key Components**:

1. **Auto Scaling Groups**
    - **Description**: Groups of EC2 instances that are managed collectively. An Auto Scaling group ensures that a specified number of instances are running and can automatically adjust the number of instances based on scaling policies or scheduled actions.
    - **Use Case**: Ensuring that the application maintains a consistent performance level during varying traffic loads.
    - **Example**: Automatically increasing the number of EC2 instances during peak hours and reducing them during off-peak times.
2. **Scaling Policies**
    - **Description**: Rules that define how and when to scale the number of instances up or down in an Auto Scaling group. There are two main types of policies:
        - **Simple Scaling Policies**: Trigger scaling actions based on a single CloudWatch alarm.
        - **Target Tracking Scaling Policies**: Adjust the number of instances to maintain a specified metric, such as average CPU utilization.
    - **Use Case**: Automatically adding or removing instances based on predefined thresholds or performance metrics.
    - **Example**: Scaling out when the average CPU utilization exceeds 70% and scaling in when it drops below 30%.
3. **Launch Configurations**

- **Description**: Templates that define the configuration of EC2 instances launched by the Auto Scaling group, including the instance type, AMI, key pair, and security groups.
- **Use Case**: Providing a standardized configuration for all instances launched by an Auto Scaling group.
- **Example**: Setting up a launch configuration to deploy EC2 instances with a specific instance type, AMI, and security settings.

4. **Scaling Plans**
   - **Description**: Provides a way to define and manage scaling policies for resources beyond EC2, such as Amazon ECS and Amazon DynamoDB. Scaling plans are part of the AWS Auto Scaling service.
   - **Use Case**: Managing scaling for multiple resources across AWS services using a unified plan.
   - **Example**: Setting up scaling plans to adjust the number of EC2 instances and DynamoDB read/write capacity based on demand.
5. **CloudWatch Alarms**
   - **Description**: Monitors metrics and sends notifications or triggers actions based on specific conditions. CloudWatch alarms are used in scaling policies to determine when to scale in or out.
   - **Use Case**: Monitoring performance metrics and initiating scaling actions when thresholds are breached.
   - **Example**: Creating an alarm to trigger scaling actions when the CPU utilization of instances exceeds 80%.
6. **Scheduled Actions**
   - **Description**: Allows you to define scaling actions based on a schedule, such as scaling up during business hours and scaling down after hours.
   - **Use Case**: Predictably managing resource levels based on known usage patterns.
   - **Example**: Scheduling an Auto Scaling group to add extra instances every weekday at 9 AM and remove them at 6 PM.

## AWS LOAD BALANCERS

**Description**:

- AWS Load Balancers distribute incoming traffic across multiple targets, such as EC2 instances, in one or more Availability Zones. This improves the availability and fault tolerance of your applications by ensuring that no single instance bears too much load.

### Types of AWS Load Balancers

1. **Application Load Balancer (ALB)**
    - **Description**: Operates at the application layer (Layer 7) and is designed to handle HTTP and HTTPS traffic. It supports advanced routing capabilities, such as path-based and host-based routing, and can route requests to different target groups based on URL paths or hostnames.
    - **Use Case**: Ideal for web applications that require advanced request routing, SSL termination, or WebSocket support.
    - **Example**: Routing traffic to different microservices based on the URL path, such as /api for API requests and /web for static web content.
2. **Network Load Balancer (NLB)**
    - **Description**: Operates at the transport layer (Layer 4) and is designed for handling TCP and UDP traffic. It provides extremely low latency and is capable of handling millions of requests per second while maintaining high throughput.
    - **Use Case**: Best suited for applications that require high performance, such as real-time applications, gaming, and network traffic.
    - **Example**: Distributing traffic for a high-performance application or game server that relies on TCP or UDP protocols.
3. **Classic Load Balancer (CLB)**
    - **Description**: Operates at both the application layer (Layer 7) and transport layer (Layer 4) but is an older generation of load

    balancers. It supports basic load balancing features for both HTTP/HTTPS and TCP traffic.
    - **Use Case**: Suitable for applications that do not require the advanced features of ALB or NLB and are already using CLB.
    - **Example**: Basic load balancing for legacy applications that do not need advanced routing or high performance.
4. **Gateway Load Balancer (GWLB)**
    - **Description**: Operates at Layer 3 (network layer) and integrates with third-party virtual appliances, such as firewalls or intrusion detection systems. It provides a way to deploy and manage these appliances in a scalable and redundant manner.
    - **Use Case**: Useful for integrating and scaling network appliances, such as firewalls or deep packet inspection systems, within your network architecture.
    - **Example**: Routing traffic through a virtual firewall to inspect and filter network traffic before it reaches the backend instances.

**Key Features**

- **Health Checks**: Automatically checks the health of registered targets and routes traffic only to healthy instances.
- **SSL Termination**: Offloads SSL/TLS encryption and decryption from instances to the load balancer, improving performance and simplifying certificate management (ALB).
- **Sticky Sessions**: Maintains user session affinity by routing requests from the same client to the same target (ALB).
- **Cross-Zone Load Balancing**: Distributes traffic evenly across all registered targets in different Availability Zones (ALB, NLB).
- **Scaling**: Automatically adjusts the number of targets based on the load (ALB, NLB).