Cloud computing refers to the delivery of computing (use or operation of computers) services.

Instead of owning and maintaining physical servers or infrastructure, users can access and use these resources on a pay-as-you-go basis. Cloud computing provides flexibility, scalability, and cost-effectiveness, allowing individuals and businesses to focus on their applications and data without the need to manage the underlying hardware.

**Cloud Service Models**

**Infrastructure as a Service (IaaS):**

* Provides virtualized computing resources over the internet.
* Users manage operating systems, applications, and runtime environments.
* Example: Amazon EC2.

**Platform as a Service (PaaS):**

* Offers a development and deployment environment in the cloud.
* Users manage applications and data, while the provider handles runtime, middleware, OS, and virtualization.
* Example: Google App Engine.

**Software as a Service (SaaS):**

* Delivers software applications over the internet, on a subscription basis.
* Users do not manage or control the underlying infrastructure but can configure application-specific settings.
* Example: Salesforce and ALM Saas.

**Key Differences**

Scope of Service: IaaS offers the building blocks for cloud IT, PaaS adds a layer of tools and software, and SaaS delivers completed applications.

Management and Control: As you move from IaaS to PaaS to SaaS, the user's control over the infrastructure decreases while the ease of use increases.

Target Audience: IaaS is aimed at IT professionals, PaaS at developers, and SaaS at end-users.

Cloud computing deployment models covered in the lesson:

**Private Cloud (Deployment Model),** A cloud environment exclusively used by a single organization. It can be hosted on-premises or by a third-party provider but is not shared with other organizations.

**Control and Privacy:** Offers greater control and privacy.

**Usage:** Ideal for businesses with high regulatory compliance needs or those needing specialized configurations.

**Public Cloud,** Services hosted and delivered by third-party providers, accessed over the Internet.

Examples include AWS, Microsoft Azure, and Google Cloud Platform.

Benefits include operational expenditure (OpEx) over capital expenditure (CapEx), economies of scale, and massive elasticity.

Users consume services like compute, storage, networking, and database.

**Hybrid Cloud,** Combines on-premises, private cloud, and public cloud services.

Ideal for companies requiring a mix of cloud solutions.

Allows critical applications and sensitive data to remain on-premises or in a private cloud.

Facilitates portability of data and applications and offers more deployment choices.

Helps in gradual migration to public cloud.

**Multi-Cloud,** Involves using multiple clouds from multiple providers, both public and private.

Allows companies to choose different clouds for different workloads based on suitability.

Examples include using AWS for some workloads, Azure for others, and on-premises private clouds for additional needs.

**Scalability and Elasticity**

Scalability: Ability to increase resources to meet rising demand.

Elasticity: Automatic scaling up and down as demand fluctuates.

Types of Scaling

**Scaling Up (Vertical Scaling):**

Adding more resources like CPU, RAM, and storage to an existing server.

Suitable for stateful applications where user data or session information is important.

Example: A user's experience on Netflix, where the system remembers past interactions.

**Scaling Out (Horizontal Scaling):**

Adding more servers to handle the load.

Ideal for stateless applications where the user experience is consistent and doesn't depend on past interactions.

Example: A news site where each user sees the same content regardless of the server they connect to.

Application Examples

Stateless Application: User interaction doesn't affect the viewed content (e.g., browsing a news site without an account).

Stateful Application: User interaction is remembered and affects future experiences (e.g., Netflix account with personalized recommendations).

E-commerce Application Example

Browsing Phase (Stateless): Users add items to a cart, with information stored in cookies, not on the server.

Purchase Phase (Stateful): User purchases are processed and recorded in a database, creating stateful data.

Implementation

Stateful Applications: Preferable to scale up to avoid inconsistencies across different servers.

Stateless Applications: Can scale out, as it doesn't matter which server the user connects to.

**Load Balancing:**

Example: A website like example.com is hosted on several servers. When a user wants to access the site, their request goes to a load balancer.

The load balancer acts like a traffic cop, directing the request to one of the cloud servers. If one server is busy or down, it directs to another.

Example Scenario: During a sale, an e-commerce site receives high traffic. The load balancer efficiently distributes incoming user requests among multiple servers to prevent any single server from being overwhelmed.

**High Availability:**

This ensures the application is available most of the time.

Example: In a streaming service like Netflix, if one server fails, the load balancer quickly shifts the load to another server, so users experience minimal disruption.

There may be short service interruptions, but the overall system remains accessible.

**Fault Tolerance:**

This involves designing systems with zero downtime, even if individual components fail.

Example: A financial transaction system uses multiple redundant hard drives. If one hard drive fails, the system continues to operate seamlessly using the other drives.

Redundant network adapters in critical servers ensure that even if one adapter fails, the server stays connected to the network.

**Combining High Availability and Fault Tolerance:**

Use of multiple availability zones can protect against data center-level failures.

Example: A cloud-based service operates across different geographical zones. If an entire data center in one zone is compromised (due to natural disaster, for instance), the service remains operational using the other zones.

Auto-scaling ensures that if a server in one zone fails, a new one is automatically launched, possibly in a different zone, to maintain service levels.

Example: An online gaming service automatically scales up its server capacity during peak gaming hours to handle the increased load, ensuring consistent gaming experience.

**Application in Cloud Services:**

Emphasis on using cloud services like AWS (Amazon Web Services) to build applications that are highly available and fault tolerant.

Example: An application hosted on AWS can leverage AWS services like Elastic Load Balancing, Auto Scaling, and multi-zone deployment to ensure that the application remains operational and responsive, even under heavy load or partial system failures.

Sign up requirements: Credit card and unique Email address.

Note: Most of us can’t afford the credit card. However, virtual credit card also is enough to sign up. For instance, use the Paytm Virtual credit card with refundable charges.

Once the sign-up process is complete.

- Root user (Email address) to sign in, the root user has full control over the account.

- Identity and Access Management (IAM), is a service provided to securely control access to AWS resources. It enables us to manage users, groups, roles, and permissions, and ensures secure access to AWS services and resources in your AWS account. Here are the key components and features of IAM:

**Users:** A user in IAM is an individual or service that interacts with AWS. Each user can have specific permissions determining what they can and cannot do in AWS.

**Groups:** A group is a collection of IAM users. You can specify permissions for a group, and those permissions are then applied to all users in the group. This makes it easier to manage permissions for multiple users with similar responsibilities.

**Roles:** Roles in IAM are used to delegate permissions to AWS services or users from other AWS accounts. Unlike users, roles do not have long-term credentials; instead, they assume temporary credentials when needed.

**Policies:** Policies are objects in AWS that, when attached to an identity or resource, define their permissions. Policies are written in JSON format and specify the actions allowed or denied by users, groups, or roles.

**Multi-Factor Authentication (MFA):** IAM supports multi-factor authentication, adding an extra layer of security by requiring users to provide additional verification beyond just a password (such as a code from a hardware token or SMS).

**Identity Federation:** IAM allows the integration of external identity providers (like corporate directories) to grant external user identities permissions to AWS resources. This is useful in scenarios where you want users to log in with their existing credentials and not have separate AWS credentials.

**Access Control:** IAM allows you to control who can access which resources in AWS in a granular fashion. This includes access to AWS services, specific resources within a service, and even the ability to perform specific actions within a service.