

UNIT-2

Cloud Computing Architecture

1. Cloud Reference Model

The **Cloud Reference Model** is a conceptual framework that describes how cloud computing services are structured and delivered. It provides a clear way to understand and categorize cloud services based on their level of abstraction and user responsibility. This model is typically divided into **three main layers**: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Each layer provides a different level of control, flexibility, and management for users and service providers.

a. Infrastructure Layer (IaaS) – *Infrastructure as a Service*

Function:

This layer provides the foundational IT resources such as **virtual machines (VMs), storage, networking, and other infrastructure components** over the internet. It essentially virtualizes physical hardware and makes it available as a service.

Key Features:

- Virtualization of servers, storage, and networks
- On-demand resource provisioning
- Elastic scalability (scale up/down based on demand)
- Pay-as-you-go pricing

Users:

- **System administrators**
- **IT operations teams**
- **DevOps engineers**

These users typically need full control over the computing environment to deploy and manage operating systems, middleware, and applications.

Example Services:

- **Amazon EC2 (Elastic Compute Cloud)**
- **Google Compute Engine**
- **Microsoft Azure Virtual Machines**

Key Control:

- **Users manage:**

- Operating systems
- Storage allocation
- Deployed applications
- Security and firewall settings
- **Cloud provider manages:**
 - Physical hardware
 - Network infrastructure
 - Virtualization technology (e.g., hypervisors)

Use Cases:

- Hosting custom applications
- Data storage and backup solutions
- Disaster recovery setups
- High-performance computing

b. Platform Layer (PaaS) – *Platform as a Service*

Function:

PaaS provides a **platform and environment** to allow developers to build applications and services over the internet. It abstracts away the underlying infrastructure, letting developers focus solely on application logic and data.

Key Features:

- Integrated development environment (IDE)
- Application hosting
- Database management systems
- Built-in scalability and load balancing
- Middleware support (e.g., application servers, messaging systems)

Users:

- **Software developers**
- **Application engineers**
- **DevOps (in build/deploy phases)**

These users want to **develop, test, and deploy applications** without managing OS-level or hardware configurations.

Example Services:

- **Google App Engine**
- **AWS Elastic Beanstalk**
- **Microsoft Azure App Service**
- **Heroku**

Key Control:

- **Users manage:**
 - Application code
 - Application configuration
 - Data used by the application
- **Cloud provider manages:**
 - OS and runtime environment
 - Middleware (e.g., database connectors, app servers)
 - Infrastructure (VMs, storage, networking)

Use Cases:

- Web application development
 - API development and hosting
 - Mobile backend services
 - Continuous integration/continuous deployment (CI/CD) pipelines
-

c. Application Layer (SaaS) – *Software as a Service***Function:**

SaaS delivers **fully functional software applications** over the internet. Users can access these applications through a web browser or API without worrying about installation, infrastructure, or updates.

Key Features:

- Web-based access to software
- Automatic updates and patching
- Multi-tenancy (shared infrastructure for multiple users)
- Subscription-based billing

Users:

- **End-users**
- **Business professionals**
- **Corporate employees**
- **Students, educators, general public**

Users interact directly with the software and do not need technical knowledge of the underlying platform or infrastructure.

Example Services:

- **Google Workspace (Gmail, Docs, Drive)**
- **Salesforce (CRM software)**
- **Microsoft 365 (Word, Excel, Teams)**

- **Dropbox, Zoom, Slack**

Key Control:

- **Users manage:**
 - Usage of the application
 - Basic user settings
 - Content/data they input
- **Cloud provider manages:**
 - Application logic and updates
 - Data storage and security
 - All underlying infrastructure and platform

Use Cases:

- Email and collaboration tools
- Customer Relationship Management (CRM)
- Enterprise Resource Planning (ERP)
- Project management and communication apps

Summary Table: Cloud Reference Model

Layer	Service Model	Managed by User	Managed by Provider	Users	Examples
Application	SaaS	App usage and data input	Everything else (app, OS, infrastructure)	End-users, business professionals	Google Workspace, Salesforce
Platform	PaaS	Applications, data	OS, middleware, runtime, hardware	Developers, engineers	Google App Engine, AWS Beanstalk
Infrastructure	IaaS	OS, apps, storage, configurations	Virtualization, physical hardware, networking	Sysadmins, IT operations	Amazon EC2, Google Compute Engine

2. Types of Clouds (Deployment Models)

Cloud **deployment models** define how cloud infrastructure is provisioned, where it is hosted, and who has control over it. These models directly affect how cloud services are accessed, secured, and managed. There are **four main deployment models**: Public Cloud, Private Cloud, Community Cloud, and Hybrid Cloud. Each has unique characteristics, benefits, and use cases depending on the organization's needs for **control, security, cost, and scalability**.

a. Public Cloud

Definition:

Public cloud refers to cloud services that are **delivered over the internet and shared across multiple organizations** (called tenants). These resources—like servers, storage, and networking—are owned and managed by a third-party cloud provider.

Ownership:

- Fully owned and operated by **cloud service providers** such as:
 - **Amazon Web Services (AWS)**
 - **Microsoft Azure**
 - **Google Cloud Platform (GCP)**

□ Benefits:

- **Cost-effective:** Pay-as-you-go pricing with no capital investment in infrastructure.
- **Scalable:** Instantly scale resources up or down based on demand.
- **Easy setup:** No need to purchase or maintain physical hardware.
- **Global reach:** Infrastructure available across many geographic regions.
- **High availability:** Built-in redundancy and disaster recovery features.

Use Case Examples:

- **Startups** launching apps with limited budgets.
 - **E-commerce sites** expecting variable traffic.
 - **Dev/test environments** that need quick provisioning and teardown.
 - Businesses that don't require full control over infrastructure.
-

b. Private Cloud

Definition:

A private cloud is a cloud environment **exclusively dedicated to one organization**. It can be **hosted on-premises** (within the company's own data center) or by a third-party provider, but it is **not shared** with other customers.

Ownership:

- Owned and managed either:
 - **Internally** by the organization (on-premises)
 - Or **externally** by a vendor (off-premises), but used exclusively by that organization.

□ Benefits:

- **Greater control** over hardware, software, and data.
- **Enhanced security and privacy:** Ideal for sensitive data or strict regulatory environments.
- **Customization:** Tailored infrastructure and performance settings.
- **Compliance:** Easier to meet industry-specific standards (HIPAA, PCI DSS, etc.).

Use Case Examples:

- **Financial institutions** requiring strict data control and audit trails.
- **Government agencies** handling classified information.
- **Healthcare providers** storing patient records.
- **Large enterprises** that need to integrate with legacy systems securely.

c. Community Cloud

Definition:

A community cloud is a **shared infrastructure** that serves a **specific group of organizations** with common concerns (such as security, compliance, or mission objectives). It's a collaborative model where several organizations share the cost and governance.

Ownership:

- Jointly owned and operated by:
 - A **group of organizations**
 - Or a **third-party provider** managing it for the group

□ Benefits:

- **Cost-sharing:** Reduces individual organization costs by pooling resources.
- **Custom governance:** Policies can be tailored to the group's shared needs.
- **Improved compliance:** Designed with industry-specific security controls.
- **Collaboration:** Promotes sharing of data and tools across the community.

Use Case Examples:

- **Healthcare consortia** sharing medical research data while maintaining HIPAA compliance.
- **Educational institutions** collaborating on cloud-hosted learning platforms.
- **Government departments** needing shared infrastructure with legal jurisdiction controls.
- **Scientific research alliances** with shared computing requirements.

d. Hybrid Cloud

Definition:

Hybrid cloud is a **combination of public and private cloud environments**, integrated in a way that allows **data and applications to move seamlessly** between them. It helps organizations balance scalability with control.

Ownership:

- **Mixed ownership:**
 - Public cloud components are owned by a cloud provider.
 - Private components are managed by the organization or a partner.

□ Benefits:

- **Flexibility:** Run sensitive workloads in private cloud and less critical ones in public cloud.
- **Cost optimization:** Use public cloud for burst capacity (e.g., peak demand) instead of overprovisioning.
- **Improved disaster recovery:** Back up private cloud data in the public cloud.
- **Seamless integration:** Unified systems across both environments with tools like hybrid cloud platforms.

Use Case Examples:

- **Retail companies:** Run e-commerce sites on public cloud but store customer data in private cloud.
- **Enterprises:** Use public cloud for development and testing, and private cloud for production.
- **Regulated industries:** Process sensitive data privately while analyzing anonymized data publicly.
- **Disaster recovery:** Public cloud as a backup for on-prem private infrastructure.

Comparison Summary Table

Deployment Model	Ownership	Access	Benefits	Ideal For
Public Cloud	Third-party providers	Internet (shared)	Low cost, scalable, fast setup	Startups, growing businesses, general-purpose workloads
Private Cloud	Single organization	Internal/private	High security, full control	Enterprises with strict security/compliance needs

Deployment Model	Ownership	Access	Benefits	Ideal For
Community Cloud	Group of organizations	Shared group	Shared cost/control, tailored compliance	Healthcare, education, government collaborations
Hybrid Cloud	Mixed (public + private)	Integrated	Flexibility, cost-efficiency, better DR	Large orgs with mixed workload and compliance needs

3. Cloud service models

Cloud **service models** define **what kind of resources** and **level of abstraction** are offered to customers in a cloud environment. These models determine how much **responsibility and control** the user has versus the cloud provider. There are three primary service models:

1. Software as a Service (SaaS)

What It Provides:

SaaS delivers **fully functional, ready-to-use software applications** over the internet. These applications are managed entirely by the cloud provider, including updates, security, and infrastructure. Users simply log in and use the software through a web browser or app — no installation or maintenance needed.

User Responsibility:

- **Minimal.** Users are only responsible for:
 - Accessing the application
 - Managing their own data (e.g., documents, emails, settings)
 - Using features provided by the software

The **cloud provider handles**:

- Hosting
- Application updates and patches
- Security
- Operating system and hardware maintenance

Examples:

Application	Use
Gmail	Web-based email service
Microsoft 365	Cloud-based productivity tools like Word, Excel, Outlook
Slack	Cloud messaging and collaboration platform
Zoom	Video conferencing and webinar software
Salesforce	Customer Relationship Management (CRM)

Use Cases:

- Businesses needing communication tools without server setup
- Teams collaborating on shared documents
- Companies automating sales and customer support workflows

Benefits:

- Easy to use and deploy
- No maintenance or infrastructure required
- Subscription-based pricing (predictable costs)
- Automatic software updates
- Accessible from anywhere via browser

2. Platform as a Service (PaaS)

What It Provides:

PaaS delivers a **development and deployment environment in the cloud**. It offers tools, programming languages, libraries, and infrastructure needed to build, test, and deploy applications. Developers can focus on writing code without worrying about hardware or system software.

User Responsibility:

- Responsible for:
 - Writing and managing application **code**
 - Configuring app settings
 - Managing app **data**
- **Cloud provider handles:**
 - Operating system

- Middleware (e.g., Java, .NET runtimes)
- Runtime environments
- Networking and storage
- Server provisioning

Examples:

Platform	Purpose
Heroku	Simple app deployment for web/mobile apps
Google App Engine	Scalable PaaS for Python, Java, Node.js apps
AWS Elastic Beanstalk	Automatically handles deployment, load balancing, scaling
Microsoft Azure App Service	Build and host web apps with .NET, Java, PHP, etc.

Use Cases:

- Rapid development and deployment of web/mobile applications
- Hosting microservices and APIs
- Automating CI/CD pipelines
- Building scalable apps without infrastructure management

Benefits:

- Speeds up development with pre-built tools
- Abstracts infrastructure complexity
- Auto-scaling and load balancing
- Supports team collaboration
- Integrated development, testing, and deployment environments

3. Infrastructure as a Service (IaaS)

What It Provides:

IaaS offers **virtualized computing resources**—including **virtual machines, storage, and networks**—over the internet. It is the most basic cloud service model and provides maximum control over the computing environment.

User Responsibility:

- Users are responsible for:
 - Installing and managing operating systems
 - Deploying and maintaining applications

- Configuring storage, security, and networking
- **Cloud provider manages:**
 - Physical infrastructure (servers, storage devices, networking)
 - Hypervisors and virtualization layers
 - Data center facilities

Examples:

Service	Function
Amazon EC2	Scalable virtual servers
Microsoft Azure Virtual Machines	Windows/Linux VMs in the cloud
Google Compute Engine	High-performance VMs with customizable resources
IBM Cloud Infrastructure	Bare metal and virtual server offerings

Use Cases:

- Migrating legacy apps to the cloud
- Hosting custom enterprise applications
- Creating test and development environments
- Big data analytics and high-performance computing
- Backup and disaster recovery solutions

Benefits:

- High flexibility and customization
- Full control over the software stack
- Scalable on-demand resources
- Pay only for what you use (compute, storage, bandwidth)
- No need to buy or maintain physical hardware

Service Model Comparison Table

Feature	SaaS	PaaS	IaaS
User Focus	End-users	Developers	System admins, IT teams
Managed by User	App data, usage settings	App code, logic, user data	OS, middleware, apps, data
Managed by	Everything else (app,	Runtime, OS, infra	Hardware,

Feature	SaaS	PaaS	IaaS
Provider	OS, infra)		virtualization
Level of Control	Lowest	Medium	Highest
Customization	Minimal	Medium	Full
Examples	Gmail, Microsoft 365, Salesforce	Heroku, Google App Engine, Beanstalk	AWS EC2, Azure VMs, GCE

Final Summary:

Model	You Want To...	Best For...
SaaS	Use a complete app without managing anything	Businesses needing ready-made solutions (e.g., email, CRM)
PaaS	Build and deploy apps quickly without worrying about infrastructure	Developers focused on coding and innovation
IaaS	Have full control over your environment	IT teams running legacy apps or custom systems