Module: 2

Cloud Deployment Models

Cloud deployment models define how cloud services are deployed, managed, and accessed based on organizational needs and infrastructure ownership.

The four main models are **Public Cloud**, **Private Cloud**, **Hybrid Cloud**, and **Community Cloud**.

Types of CDM

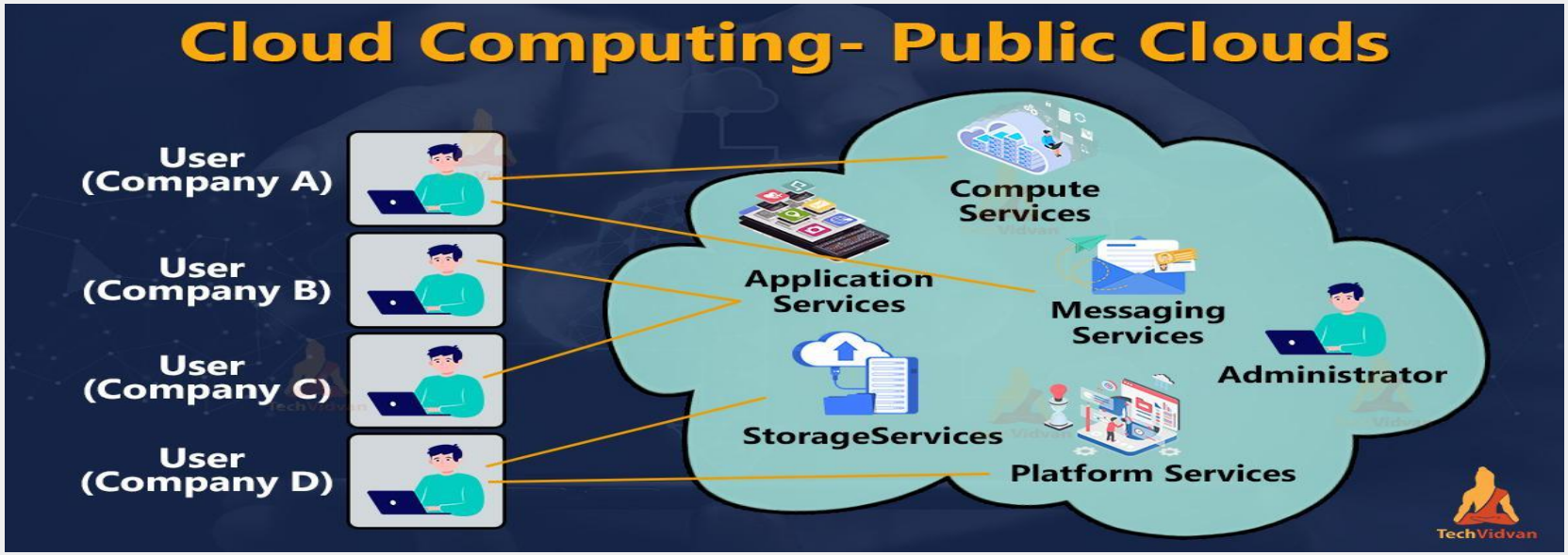
Public Cloud

The public cloud is a cloud computing model where **services and infrastructure are hosted by third-party** providers and shared among multiple organizations.

Public cloud services are accessed over the internet and offer scalability, cost-efficiency, and minimal management.

Characteristics:

- Shared resources across multiple tenants.
- Providers manage infrastructure, maintenance, and upgrades.
- Accessible over the internet.
- Pay-as-you-go pricing.



Examples:

- **Netflix:** Uses Amazon Web Services **(AWS)** for streaming, enabling scalability during peak hours.
- **Adobe Creative Cloud:** Runs on **Microsoft Azure**, providing design software globally with high availability.
- **Spotify:** Leverages Google Cloud Platform **(GCP)** for data analytics and real-time processing of user preferences.
- **Dropbox:** Migrated from private infrastructure to **AWS** for cost savings and enhanced collaboration features.

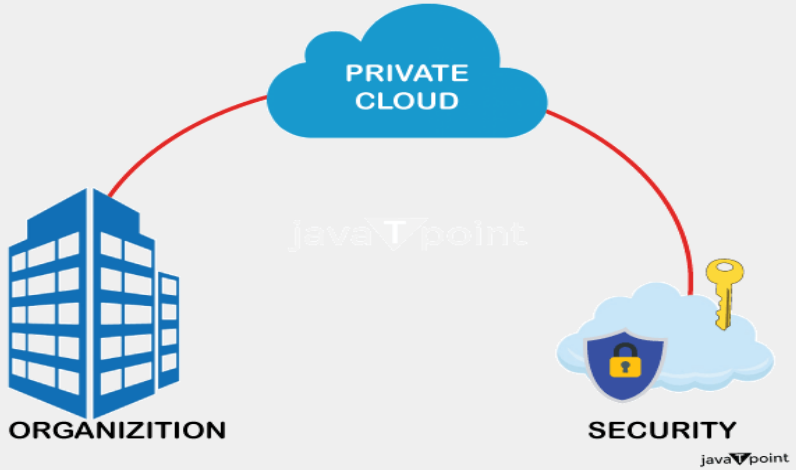
Use Cases:

- Hosting websites and blogs.
- High-traffic applications requiring scalability.

Development and testing environments.

- **Private Cloud**

The private cloud is dedicated to a **single organization**, offering enhanced ***security, control, and customization***. It can be hosted on-premises or by third-party providers.



Characteristics:

- Exclusive use by one organization.
- Enhanced privacy and compliance.

Customizable architecture tailored to organizational needs.

High initial costs but reduced ongoing risks.

Examples:

- **Walmart:** Operates a private cloud using **OpenStack** for internal operations and inventory management.
- **Capital One:** Uses **VMware**-based private cloud infrastructure for secure and compliant financial services.
- **NASA:** Developed its private cloud called **Nebula** to support research data and internal projects.
- **BMW:** Runs its applications on a private cloud for production and supply chain optimization.

Use Cases:

- Banking and financial institutions with strict regulatory needs.
- Healthcare organizations handling sensitive patient data.
- Large enterprises requiring control over data and applications.

- **Hybrid Cloud**

The hybrid cloud ***integrates public and private clouds***, allowing organizations to use both depending on their needs. For example, sensitive data can reside in a private cloud, while public cloud services handle high-volume workloads.

Characteristics:

- Combines public and private cloud capabilities.
- Complex setup and management.

Provides scalability while retaining data control.

Ideal for dynamic or unpredictable workloads.

Examples:

- **Johnson Controls:** Uses ***Microsoft Azure Hybrid Cloud*** for IoT solutions, blending on-premises control with cloud scalability.
- **Philips Healthcare:** Employs ***AWS Outposts*** for a hybrid cloud setup to process medical imaging data securely.
- **HSBC:** Uses ***Google Anthos*** to migrate workloads between on-premises systems and public cloud environments.
- **General Electric (GE):** Combines private cloud systems with **AWS** to support industrial IoT and analytics.

Use Cases:

- Disaster recovery and business continuity. Applications with fluctuating demand. Seamless integration of legacy systems with modern cloud services.

- **Community Cloud**

A community cloud is a shared infrastructure tailored to the needs of a specific group of organizations with common objectives, such as compliance or collaborative projects.

Characteristics:

- Shared by organizations with similar requirements. Cost-effective through shared resources.
- Focus on compliance and security for specific industries. Limited scalability compared to public clouds.

Examples:

- **G-Cloud UK:** A cloud framework for public sector organizations in the UK to access shared services securely.
- **NASA Nebula:** A community cloud developed for U.S. federal agencies to store and share research data.
- **Healthcare Community Cloud:** Used by hospitals for secure patient data sharing and collaborative research.
- **European Cloud Infrastructure (GAIA-X):** Designed to create a community cloud for European businesses, focusing on sovereignty and data privacy.

Use Cases:

- Collaborative research in science and academia. Government agencies sharing secure data.
- Healthcare organizations managing shared patient records.

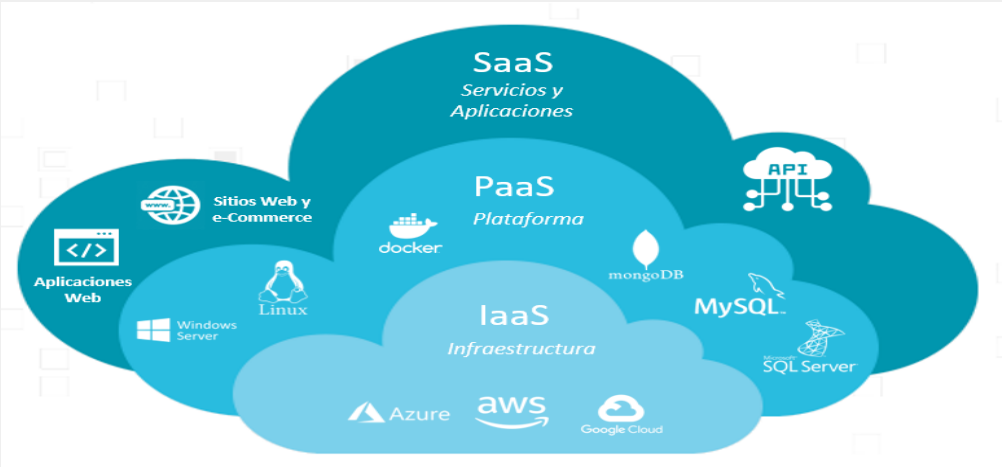
- **As Per Studies Cloud Deployment Model can be Chosen Based on Factors Such as:**
  - **Compliance Requirements:** Private or community clouds for sensitive data.
  - **Budget Constraints:** Public clouds for startups and small businesses.
  - **Scalability Needs:** Hybrid or public clouds for dynamic workloads.
  - **Data Control:** Private clouds for maximum security and customization.

- **Comparison of Deployment Models** (Public/Private/Hybrid/Community Cloud)

Model	Ownership	Scalability	Security	Cost	Examples
Public Cloud	Third-party	High	Moderate	Pay-as-you-go	AWS, Azure, GCP,Dropbox
Private Cloud	Single organization	Moderate to high	High	High upfront costs	Walmart, NASA, BMW
Hybrid Cloud	Mix of public/private	High	High	Depends on usage	HSBC, Philips, Johnson Controls
Community Cloud	Shared organizations	Moderate	High	Shared costs	NASA Nebula, G-Cloud UK, GAIA-X

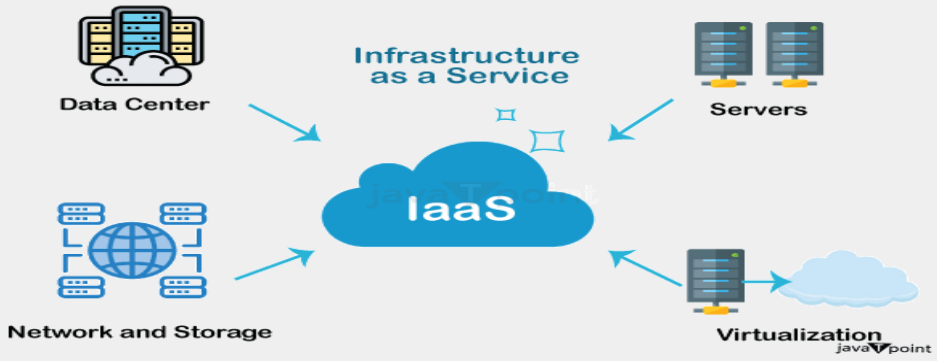
Cloud Service Models

Cloud service models are the different ways that cloud computing services are offered to customers. They define how services are delivered, managed, and offered. Some examples of cloud service models include: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and Function as a Service (FaaS)



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

IaaS provides **virtualized computing resources** like *servers, storage, and networking* on a *pay-as-you-go* basis. It eliminates the need for physical hardware, offering flexibility and scalability.



Key Features:

- Virtualized hardware resources. Scalable infrastructure for varying workloads.
- Pay-as-you-go pricing. High availability and disaster recovery options.

Advantages:

- No hardware investment or maintenance. Highly scalable and flexible.
- Full control over the operating system and applications. Supports multiple platforms and development tools.

Disadvantages:

- Requires IT expertise for management.
- Security and compliance responsibilities are partly on the user.

Real-World Examples:

- **Amazon Web Services (AWS EC2)**: Provides scalable virtual machines for computing needs.
- **Microsoft Azure Virtual Machines**: Enables deployment of Linux and Windows servers on-demand.
- **Google Compute Engine**: Offers customizable virtual machines in Google Cloud.

Use Cases:

- Hosting websites and applications.
- Development and testing environments.
- Backup and disaster recovery.
- High-performance computing (e.g., scientific simulations).

Platform as a Service (PaaS)

PaaS provides a **platform** for developers to *build, test, and deploy applications* without worrying about the underlying infrastructure. It includes tools, libraries, and frameworks for application development.

Key Features:

- Integrated development environment (IDE).
- Simplifies deployment and scaling.
- Pre-configured operating systems and middleware.

Advantages:

- Reduces development time and costs.
- No need to manage servers or storage.

- Built-in tools for monitoring, security, and load balancing.

Disadvantages:

- Limited control over the underlying infrastructure. May face vendor lock-in.
- Dependency on provider uptime.

Real-World Examples:

- **Heroku**: A PaaS solution for deploying and scaling web applications in various programming languages.
- **AWS Lambda**: A serverless compute service that runs code in response to events or triggers.
- **Amazon Elastic Beanstalk**: A service that automatically creates websites.
- **Google App Engine**: Simplifies app development with built-in scalability.
- **Microsoft Azure App Services**: A platform for hosting web apps, RESTful APIs, and mobile backends.

Use Cases:

- Developing and testing software applications. Deploying microservices-based architectures.
- Rapid prototyping and development.

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

SaaS delivers fully **functional software applications** over the internet. Users access these applications via a web browser *without installing* or managing software.

Key Features:

- Fully managed by the provider.
- Subscription-based pricing.
- Accessible from any device with an internet connection.

Advantages:

- No hardware or software maintenance.
- Regular updates and new features managed by the provider.
- Easy scalability and accessibility.

Disadvantages:

- Limited customization options. Data security concerns in multi-tenant environments.
- Dependency on internet connectivity.

Real-World Examples:

- **Google Workspace (formerly G Suite)**: Cloud-based productivity apps like Gmail, Docs, and Sheets.
- **Salesforce**: A leading CRM platform for sales, marketing, and customer support.
- **Zoom**: Cloud-based video conferencing software.
- **AWS:** AWS RDS Aurora, DynamoDB, SNS, SQS, Lambda, and S3.

Use Cases:

- Email and communication tools. Customer Relationship Management (CRM) platforms.
- Collaboration tools like Microsoft Teams or Slack.

● Function as a Service (FaaS)

FaaS, also known as **serverless computing**, allows developers to write and deploy code in small units (functions) *without managing servers*. The provider automatically handles infrastructure scaling.

Key Features:

- Event-driven execution. Pay-per-use pricing. Simplified scaling and resource allocation.

Advantages:

- Reduces operational overhead. Highly cost-effective for infrequent workloads.
- Supports rapid application development.

Disadvantages:

- Not suitable for long-running processes. Cold starts can cause delays.
- Debugging and monitoring may be complex.

Real-World Examples:

- **AWS Lambda**: Executes code in response to events like HTTP requests or database updates.
- **Google Cloud Functions**: Runs lightweight functions triggered by cloud events.
- **Azure Functions**: Integrates with other Azure services for event-driven workflows.

Use Cases:

- Running scheduled tasks. Real-time data processing.
- Backend processing for mobile or IoT applications.

Choosing the Right Service Model Based on Organisations Specific Needs:

- **IaaS:**

  Best for IT teams needing complete control over infrastructure for hosting, backups, or running custom environments.

- **PaaS:**

  Ideal for developers focusing on application development without managing infrastructure.

- **SaaS:**

  Suitable for businesses requiring ready-to-use applications like email, CRM, or collaboration tools.

- **FaaS:**

  Perfect for developers creating lightweight, event-driven apps without worrying about servers.

- **Comparison of Cloud Service Models**

Aspect	IaaS	PaaS	SaaS	FaaS
Definition	Virtualized computing resources	Application development platform	Fully functional software	Serverless code execution

User Control	Full control over resources	Application development and deployment	Minimal user control	Focused on functions only
Scalability	High	High	High	High
Examples	AWS EC2, Azure VM, Google Compute	Heroku, Google App Engine, Azure App Services	Google Workspace, Salesforce, Zoom	AWS Lambda, Google Cloud Functions
Use Cases	Hosting, backup, testing	App development, microservices	CRM, collaboration tools	Event-driven apps, real-time data

Cloud Data Lifecycle Phases

The **cloud data lifecycle** refers to the stages data undergoes ***from its creation or collection in a cloud environment to its deletion***. Understanding these phases **ensures proper data management, security, and compliance in cloud computing**. The lifecycle generally comprises the following phases:

- **Data Creation/Collection**

This phase involves generating or collecting data through various sources such as *users, applications, IoT devices, or sensors.*

- **Key Activities**:
    - Data input or generation (e.g., customer information, transaction records).
    - Data collection from external sources (e.g., APIs, web scraping).
    - Defining metadata for data organization.
- **Challenges**:
    - Ensuring data integrity and accuracy. Avoiding duplication or incomplete data collection.
- **Best Practices**:
    - Use secure channels for data transmission during collection.
    - Validate data upon creation to ensure quality.

- **<u>Data Storage</u>**

Once created, data is stored in cloud environments such as *databases, data warehouses, or object storage systems*.

- **Key Activities**:
    - Organizing data in structured, semi-structured, or unstructured formats.
    - Encrypting data at rest.
    - Implementing storage tiering (hot, warm, or cold storage) based on data access frequency.

- **Challenges**:
  - Selecting the appropriate storage solution. Managing storage costs and scalability.
- **Best Practices**:
  - Use redundant storage for fault tolerance (e.g., RAID, cloud provider replication).
  - Ensure data access policies align with compliance requirements.

- **Data Use/Processing**

This phase involves accessing, processing, and analyzing stored data to derive value and insights.

- **Key Activities**:
  - Querying and retrieving data for applications or analytics.
  - Transforming data for analysis (e.g., ETL – Extract, Transform, Load).
  - Using data in real-time applications or decision-making systems.
- **Challenges**:
  - Balancing performance with resource usage during data processing.
  - Ensuring real-time data availability.
- **Best Practices**:
  - Optimize database queries for efficient data retrieval.
  - Leverage cloud-native services like AWS Lambda for real-time processing.

- **Data Sharing/Distribution**

  Sharing data across *systems, users, or organizations for collaboration, reporting, or integration purposes.*

  - **Key Activities**:
    - Creating APIs or data pipelines for distribution.
    - Sharing data securely across geographic regions or teams.
    - Ensuring data access controls (e.g., role-based access).
  - **Challenges**:
    - Preventing unauthorized access during sharing.
    - Managing data consistency across distributed environments.
  - **Best Practices**:
    - Use encrypted communication protocols (e.g., HTTPS, TLS).
    - Employ data governance policies to regulate sharing.

- **Data Archival**

  Data that is **no longer frequently accessed** is moved to *long-term storage or archived* for compliance and historical purposes.

  - **Key Activities**:

    - Identifying data suitable for archival based on usage patterns.
    - Moving data to cost-effective, long-term storage solutions.
    - Applying retention policies and ensuring data immutability (if required).
- **Challenges**:
    - Determining the appropriate archival period.
    - Retrieving archived data efficiently if needed.
- **Best Practices**:
    - Use archival services like AWS Glacier or Azure Blob Archive.
    - Regularly review and adjust archival policies to align with compliance requirements.

- **Data Deletion/Destruction**

  When data is no longer needed or has reached the end of its retention period, it is securely deleted or destroyed.

  - **Key Activities**:
    - Identifying data eligible for deletion based on retention policies.
    - Securely erasing data to prevent recovery.
    - Verifying deletion to meet compliance standards.
  - **Challenges**:
    - Ensuring data is completely unrecoverable.

- Adhering to legal and regulatory requirements for data destruction.

- **Best Practices**:
  - Use cloud-native tools for secure deletion (e.g., AWS KMS for cryptographic deletion).
  - Document and audit the deletion process for compliance purposes.

Content Beyond the Syllabus (Optional)

Grid computing, fog computing, and edge computing are all distributed computing paradigms designed to optimize performance, reduce latency, and improve efficiency. Here's an overview of each:

- **Grid Computing**

  Grid computing is a distributed computing model where multiple computers, often geographically dispersed, work together to solve complex problems. Resources such as processing power, storage, and memory are pooled to execute tasks collaboratively.

  - **Key Features**:
    - High-performance computing across distributed nodes.
    - Used for resource-intensive tasks like scientific simulations, financial modeling, or data analysis.
    - Relies on middleware to manage and coordinate the distributed resources.
  - **Example Use Cases**:

- Weather forecasting. Genomic research.
- Large-scale simulations (e.g., CERN particle physics).

- **Benefits**:
  - Cost-effective: Utilizes existing resources. Scalability: Adds more nodes for higher capacity.
- **Challenges**:
  - Network dependency and latency. Security concerns due to resource sharing.

- **Fog Computing**

Fog computing extends cloud computing by bringing computational, storage, and networking resources closer to the devices at the network edge. It acts as a bridge between edge devices and the cloud.

- **Key Features**:
  - Decentralized processing. Low latency due to proximity to end-users.
  - Supports real-time processing.
- **Example Use Cases**:
  - Smart cities (traffic management, surveillance). Industrial IoT (monitoring and control systems).
  - Healthcare (real-time monitoring of patients).
- **Benefits**:
  - Reduces data travel distance, lowering latency. Minimizes bandwidth usage by processing data locally.

- **Challenges**:
    - Complexity in deployment and management. Ensuring interoperability among devices.

- **Edge Computing**

  Edge computing focuses on processing data directly at or near the source of data generation (e.g., sensors, IoT devices) rather than sending it to centralized servers or the cloud.

    - **Key Features**:
        - Ultra-low latency. High autonomy (less reliance on central servers).
        - Suitable for highly localized data processing.
    - **Example Use Cases**:
        - Autonomous vehicles (real-time decision-making).
        - Smart devices (voice assistants, AR/VR applications). Remote monitoring (oil rigs, agriculture).
    - **Benefits**:
        - Instant response times. Enhanced privacy (data stays close to its origin).
    - **Challenges**:
        - Limited resources compared to cloud systems. Maintenance of distributed devices.