
Unit 1: Introduction

1. Definition, Historical Developments, Computing Platforms and Technologies

- **Definition: Cloud Computing** is the on-demand delivery of IT resources and applications over the internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, from a cloud provider.
- **Historical Developments:**
    - **1960s:** Concept of **utility computing** (like an electricity utility) emerged, where resources could be shared.
    - **1990s:** The internet and widespread use of the World Wide Web provided the necessary **network infrastructure**.
    - **2000s: Virtualization** technology matured, making resource sharing more efficient. **Amazon Web Services (AWS)** launched its S3 storage service in 2006, marking the beginning of the modern cloud era.
- **Computing Platforms and Technologies:** These include the underlying technologies that make the cloud possible:
    - **Virtualization** (covered in Unit 2).
    - **Service-Oriented Architecture (SOA)** for designing services.
    - **Web Services** and **APIs** (like REST, SOAP) for accessing services.
    - **Distributed Systems** (covered below).

2. Building Cloud Computing Environments

Building a cloud environment involves:

1. **Infrastructure Setup:** Establishing the physical hardware (servers, storage, network gear).
2. **Virtualization Layer:** Implementing a **Hypervisor** (or Virtual Machine Monitor) to create and manage virtual resources.
3. **Resource Management:** Deploying software to dynamically allocate, monitor, and scale resources (often called a **Cloud Operating System** or **Cloud Management Platform**).
4. **Service Delivery:** Creating interfaces (web portals, APIs) for users to consume the resources as services.

3. Principles of Parallel and Distributed Computing

Feature	**Parallel Computing**	**Distributed Computing**
Definition	Multiple processors working simultaneously on different parts	Multiple independent computers (nodes) networked together,

Feature	Parallel Computing	Distributed Computing
	of the **same problem** in a single machine (or tightly coupled cluster).	working on different tasks/data, communicating via message passing.
Goal	To **speed up** a single computation (High Performance Computing).	To handle a large amount of data or tasks, ensuring **fault tolerance** and **scalability**.
Communication	Fast, shared memory access (low latency).	Message passing over a network (higher latency).
Example	Multi-core CPU processing an image filter.	A web server farm handling concurrent user requests.

- **Elements of Parallel Computing:**
    - **Synchronization:** Coordinating tasks to ensure correct execution order.
    - **Load Balancing:** Distributing the workload evenly among processors.
    - **Shared Memory vs. Distributed Memory:** How processors access data.
- **Elements of Distributed Computing:**
    - **Concurrency:** Multiple tasks executing over the same period.
    - **Fault Tolerance:** The system continues to operate even if some nodes fail.
    - **Transparency:** Users view the system as a single, coherent resource.
    - **Scalability:** Ability to easily add more nodes/resources.
- **Technologies for Distributed Computing:**
    - **Networking Protocols** (TCP/IP).
    - **Middleware** (e.g., Message Queuing systems, Remote Procedure Calls - RPC).
    - **Distributed File Systems** (e.g., Hadoop Distributed File System - HDFS).

Unit 2: Virtualization 🧱

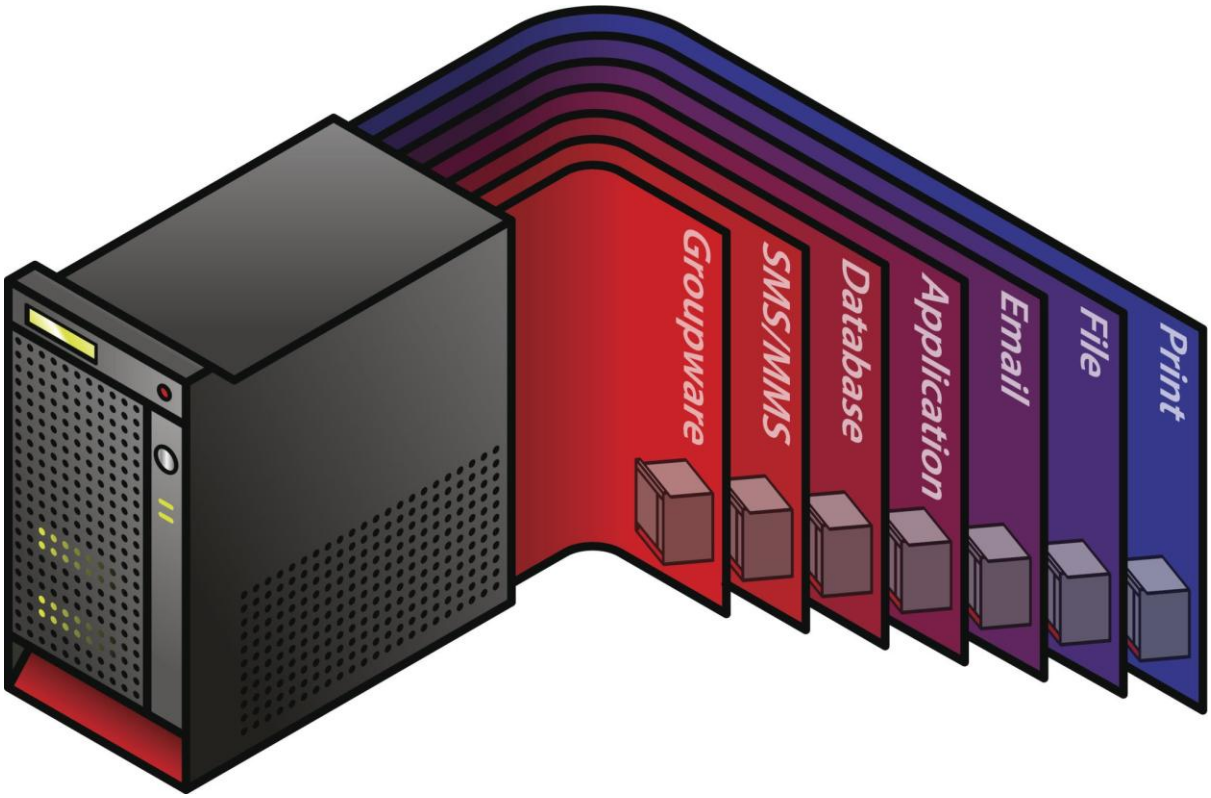
1. Characteristics and Techniques

Virtualization is the process of creating a software-based (virtual) representation of something, such as an application, server, storage device, or network.

- **Characteristics:**

- **Resource Isolation:** Virtual Machines (VMs) are isolated from each other and the host machine.
- **Hardware Independence:** VMs can run on different underlying physical hardware.
- **Encapsulation:** The entire state of a VM is saved in a few files, making it easy to move.
- **Partitioning:** Dividing a single physical server's resources (CPU, RAM, storage) among multiple VMs.

- **Virtualization Techniques:**
  - **Hardware Virtualization (Server Virtualization):**
    - **Type 1 (Bare-Metal) Hypervisor:** Runs directly on the host hardware (e.g., VMware ESXi, Xen). Highly efficient.
    - **Type 2 (Hosted) Hypervisor:** Runs on top of a conventional operating system (e.g., VirtualBox, VMware Workstation). Less efficient but easier to set up.
  - **Operating System (OS) Level Virtualization (Containerization):** Shares the host OS kernel instead of virtualizing the hardware (e.g., Docker, Linux Containers). Much lighter and faster (covered in Unit 5).
  - **Other Types:** Network Virtualization (SDN), Storage Virtualization, Application Virtualization.



Getty Images

2. Virtualization and Cloud Computing

Virtualization is the **fundamental enabling technology** for cloud computing.

- It allows cloud providers to run multiple, isolated, customer environments (VMs) on a single physical server.
- This achieves **multitenancy** (sharing the same infrastructure among multiple users) and dramatically improves **resource utilization**.
- It facilitates **dynamic resource allocation** (spinning up or down VMs quickly) and **VM migration** (moving a running VM between physical servers for maintenance).

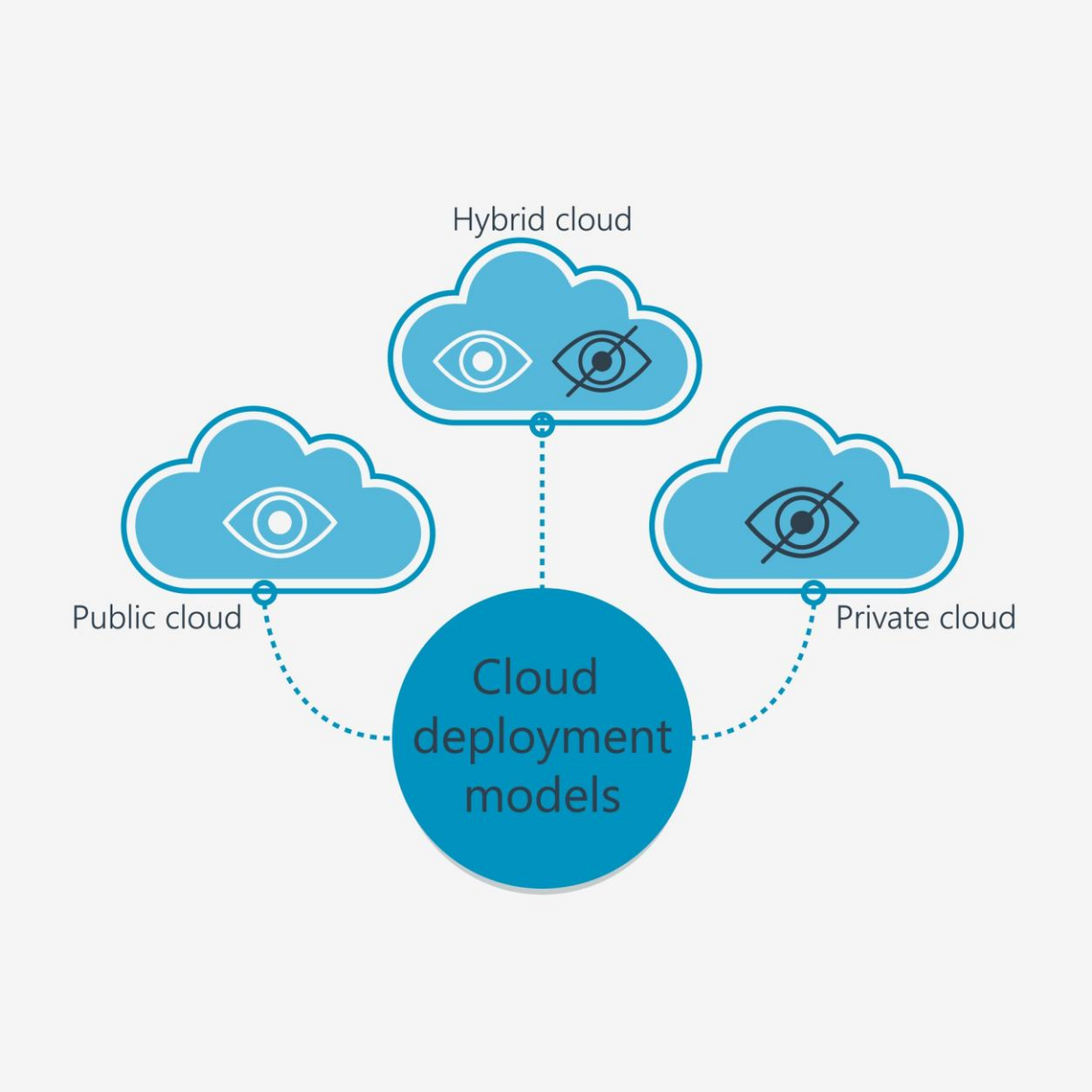
3. Pros and Cons of Virtualization

Pros (Advantages)	**Cons (Disadvantages)**
Cost Savings: Fewer physical servers needed (less power, cooling, space).	**Performance Overhead:** The hypervisor layer introduces some performance loss, especially for I/O-intensive tasks.
Better Resource Utilization: High server capacity usage.	**Security Risks (Hypervisor):** If the hypervisor is compromised, all hosted VMs could be affected.
Disaster Recovery/Backup: VMs can be easily backed up, restored, or moved.	**Sprawl:** The easy creation of VMs can lead to "VM Sprawl," where unmanaged VMs proliferate, wasting resources.
Rapid Provisioning: VMs can be created in minutes.	**Licensing Complexity:** Software licensing can be complicated in a virtual environment.

Unit 3: Cloud Computing Architecture 🏛️

1. Cloud Reference Model

The cloud architecture is typically described by its **deployment models** (where the cloud is located) and **service models** (what the cloud offers).



Shutterstock

2. Types of Clouds (Deployment Models)

Type	Description	Key Feature
Public Cloud	Services offered over the public internet and sold to the general public. Owned and operated by a third-party provider (e.g., AWS, GCP, Azure).	**Shared Infrastructure, Pay-as-you-go.**
Private Cloud	Computing services offered either over the Internet or a private internal network and dedicated to a single organization.	**Exclusive Use, High Security/Control.**

Type	Description	Key Feature
Hybrid Cloud	A combination of two or more distinct cloud infrastructures (private and public) that remain unique entities but are bound together by standardized technology.	**Flexibility, Workload Portability.**
Community Cloud	Infrastructure shared by several organizations with common concerns (e.g., security, compliance, jurisdiction).	**Shared Governance, Sector-Specific.**

3. Types of Services (Service Models)

These models define what the cloud provider manages and what the user manages.

- **IaaS (Infrastructure as a Service):**
    - **What it is:** The building blocks for cloud IT. Access to networking features, computers (virtual machines), and data storage.
    - **User Manages:** Operating System, Middleware, Applications, and Data.
    - **Provider Manages:** Servers, Storage, Networking, and Virtualization.
    - **Example:** AWS EC2, Azure VMs, Google Compute Engine.
- **PaaS (Platform as a Service):**
    - **What it is:** A complete development and deployment environment in the cloud, with resources that enable users to deliver everything from simple cloud-based apps to sophisticated enterprise applications.
    - **User Manages:** Applications and Data.
    - **Provider Manages:** OS, Middleware, Runtime, and everything below.
    - **Example:** AWS Elastic Beanstalk, Azure App Service, Google App Engine.
- **SaaS (Software as a Service):**
    - **What it is:** Complete application delivered over the internet, typically on a subscription basis.
    - **User Manages:** Nothing but consumption (and perhaps configuration).
    - **Provider Manages:** All aspects of the application stack.
    - **Example:** Gmail, Salesforce, Microsoft 365.

4. Economics of Clouds

- **Capital Expenditure (CapEx) $\rightarrow$ Operational Expenditure (OpEx):** Cloud shifts costs from upfront capital investment in hardware (CapEx) to monthly service fees (OpEx).

- **Pay-as-you-go Pricing:** Customers only pay for the exact resources they consume.
- **Economies of Scale:** Cloud providers buy and manage hardware at massive scale, passing the cost savings to customers.
- **Reduced Total Cost of Ownership (TCO):** Less cost associated with maintenance, cooling, and space.

5. Open Challenges

- **Security & Compliance:** Data breach risks, meeting regulatory requirements across various jurisdictions.
- **Vendor Lock-in:** Difficulty and cost of moving data and applications from one cloud provider to another.
- **Interoperability:** Ensuring different cloud services and on-premises systems can work together seamlessly.
- **Management Complexity:** Governing resources across multi-cloud or hybrid environments.

6. Case Study: Amazon Web Services (AWS)

AWS is the world's leading public cloud platform.

- **IaaS Example: Amazon EC2 (Elastic Compute Cloud):** Provides resizable compute capacity (VMs).
- **PaaS Example: AWS Elastic Beanstalk:** An easy-to-use service for deploying and scaling web applications.
- **Storage Example: Amazon S3 (Simple Storage Service):** Object storage service offering industry-leading scalability, data availability, security, and performance.
- **Key Feature:** Massive global infrastructure and a vast ecosystem of integrated services (databases, AI/ML, serverless, etc.).

Unit 4: Migration into Cloud and Virtual Machine Provisioning 🚀

1. Broad Approaches to Migrating into the Cloud

Often summarized as the **"6 R's"** of migration:

1. **Rehost (Lift and Shift):** Moving an application as-is to a cloud VM. Quickest, but least optimized.
2. **Replatform (Lift, Tinker, and Shift):** Moving an application while making minor, cloud-friendly changes (e.g., replacing on-prem database with a managed cloud database).
3. **Refactor/Rearchitect:** Modifying an application's architecture to fully utilize cloud-native features (e.g., converting to microservices, using serverless functions). Most work, greatest long-term benefit.
4. **Repurchase (Drop and Shop):** Moving from a custom/on-prem application to a SaaS offering (e.g., replacing an internal email server with Microsoft 365).

5. **Retain (Revisit):** Keeping some applications on-premises because they are not cloud-ready or are too costly/complex to migrate.
6. **Retire:** Decommissioning applications that are no longer needed.

2. The Seven-Step Model of Migration into a Cloud

1. **Define Strategy:** Determine business goals, scope, and migration approach (6 R's).
2. **Assess Readiness:** Evaluate the current environment (applications, infrastructure, skills) and identify gaps.
3. **Choose Landing Zone:** Set up the cloud environment (accounts, networking, security controls).
4. **Pilot Migration:** Migrate a small, non-critical application as a test case to validate processes.
5. **Replicate Environment:** Build the target cloud environment for the production workload.
6. **Migrate Application:** Execute the large-scale migration, usually in waves.
7. **Operate and Optimize:** Cut over traffic, decommission old infrastructure, and continuously optimize cloud usage and cost.

3. Virtual Machines Provisioning and Manageability

VM Provisioning is the process of creating and deploying a VM.

- **Static Provisioning:** Pre-creating and dedicating VMs to users, which can lead to wasted resources.
- **Dynamic Provisioning:** Creating and allocating VMs only when requested (the standard cloud model).
- **Manageability:** Tools and APIs provided by the cloud vendor/hypervisor (e.g., AWS CloudFormation, API calls) for tasks like starting, stopping, patching, and monitoring the VM lifecycle.

4. Virtual Machine Migration Services

This involves moving a running VM from one physical host to another.

- **Cold Migration:** VM is shut down before the move (zero downtime).
- **Warm Migration:** VM is suspended, moved, and then resumed (brief downtime).
- **Live Migration (Hot Migration):** The VM remains running throughout the move, transferring memory and state while the user is connected (near-zero downtime). This is critical for maintenance and load balancing in cloud data centers.

5. VM Provisioning and Migration in Action & Provisioning in the Cloud Context

- In a cloud context, provisioning is heavily **automated** and **API-driven**. A user request (e.g., a CLI command) triggers the cloud's management plane to locate a suitable physical host, deploy a copy of the OS/Image, allocate resources, and power it up.
- Cloud provisioning focuses on **scalability** (auto-scaling groups can launch hundreds of VMs) and **elasticity** (resources can be scaled up or down instantly).

Unit 5: Advanced Concepts – Docker, Container and Kubernetes 📦

1. Introduction to CaaS and Why Containers?

- **CaaS (Container as a Service):** A cloud service model where the provider manages the underlying infrastructure and container orchestration, allowing users to simply upload, run, and manage their containers.
- **Why Containers?**
    - **Portability:** A container image runs the same way on any environment (developer's laptop, on-prem server, public cloud).
    - **Consistency:** Eliminates "works on my machine" issues by packaging the application and all its dependencies.
    - **Efficiency:** They are lightweight and start up instantly, requiring far less resource overhead than VMs.

2. Difference between Virtualization and Containers

Feature	Virtual Machine (VM)	Container
Isolation	High (Hypervisor separates OS and Hardware).	Medium (Separate user space, but shares the Host OS kernel).
Size	Gigabytes (includes a full Guest OS).	Megabytes (only includes application code and libraries).
Startup	Minutes (needs to boot an entire OS).	Seconds or milliseconds.
Resource Usage	High (requires dedicated OS/resources).	Low (shares Host OS kernel and resources dynamically).
Technology	Hypervisor (Type 1 or 2).	Docker/Container Runtime and shared Host OS kernel.

3. Introduction to Containers, Docker and its Architecture

- **Container:** A lightweight, executable software package that includes everything needed to run an application: code, runtime, system tools, system libraries, and settings. They are isolated processes running on the host OS.
- **Docker:** The most popular open-source platform for building, shipping, and running containers.

- **Docker Architecture:**
    - **Docker Daemon (Server):** The background service running on the host machine that manages all the containers, images, volumes, and networks.
    - **Docker Client:** The command-line tool (docker) that communicates with the Daemon via an API.
    - **Docker Images:** Read-only templates used to create containers (built from a Dockerfile).
    - **Docker Registries:** Stores Docker images (e.g., Docker Hub).

4. Understanding Docker Container, Networking

- **Docker Container:** A running instance of a Docker image. It is a process running on the host, isolated by Linux kernel features like **cgroups** (for resource limiting) and **namespaces** (for isolation).
- **Networking:** Docker creates virtual networks to allow containers to communicate.
    - **Bridge Network (Default):** A private network for containers on a single host to communicate.
    - **Host Network:** Container shares the host's networking stack, no isolation.
    - **Overlay Network:** Used for allowing containers on different hosts (multi-host communication) to communicate, typically used with orchestrators like Kubernetes.

5. Kubernetes – Introduction, Architecture

- **Introduction: Kubernetes (K8s)** is an open-source system for automating deployment, scaling, and management of containerized applications. It acts as an orchestrator for containers, managing their lifecycle across a cluster of hosts.
- **Architecture:** Kubernetes follows a **Master-Slave (Control Plane - Worker Node)** architecture.

Component	Role
Control Plane (Master Node)	Manages the cluster state and decision-making. Includes: **API Server** (front-end), **etcd** (cluster configuration/state storage), **Scheduler** (assigns pods to nodes), **Controller Manager** (runs control loops).
Worker Node (Slave Node)	Runs the actual containerized applications (Pods). Includes: **Kubelet** (agent for node communication), **Kube-Proxy** (networking and service discovery), and a **Container Runtime** (e.g., Docker).
Pod	The smallest deployable unit in Kubernetes; a group of one or more containers that share storage and network resources.

Unit 6: Cloud Security & Management 🛡️

1. Fundamental Cloud Security

- **Basic Terms and Concepts:**
    - **Shared Responsibility Model:** A key concept where security is a partnership: the **Cloud Provider** is responsible for **Security of the Cloud** (physical infrastructure, networking, hypervisor), and the **Customer** is responsible for **Security in the Cloud** (data, applications, operating system patches, identity).
    - **Compliance:** Adhering to standards and regulations (e.g., GDPR, HIPAA, PCI-DSS).
- **Threat Agents:** The entities that pose a threat:
    - **Malicious Insiders:** Employees of the cloud provider or customer.
    - **Hackers/External Attackers:** Individuals or groups targeting cloud resources.
    - **Government/State-Sponsored Actors:** Targeting high-value, sensitive data.
- **Cloud Security Threats (Top Examples):**
    - **Data Breaches:** Unauthorized exposure of sensitive data.
    - **Insecure APIs/Interfaces:** Vulnerabilities in the primary means of cloud interaction.
    - **Misconfiguration and Inadequate Change Control:** The #1 cause of cloud breaches (e.g., leaving a storage bucket publicly accessible).
    - **Lack of Cloud Security Architecture and Strategy:** Failure to design security into the cloud environment from the start.
- **Case Study Example (Misconfiguration):** A major company left an Amazon S3 bucket storing customer data publicly readable because of a single, mistaken configuration setting, leading to a massive data leak. This highlights the customer's responsibility in the Shared Responsibility Model.

2. Cloud Security Mechanisms

- **PKI (Public Key Infrastructure):**
    - A system of digital certificates, Certificate Authorities (CAs), and other registration authorities that verify and authenticate the validity of each party involved in an electronic transaction.
    - **Role in Cloud:** Used for securing communication (TLS/SSL encryption for web traffic), ensuring the identity of servers and clients.
- **IAM (Identity and Access Management):**
    - The framework of policies and technologies to ensure that the right users (human or machine) have the appropriate access to technology resources.
    - **Key Concepts:**
        - **Identity:** Who the user is (e.g., email, account name).

        - **Authentication:** Verifying the identity (e.g., password, MFA).
        - **Authorization:** What the authenticated user is allowed to do (defined by roles/policies).
    - **Case Study (AWS IAM):** AWS IAM allows users to create specific users, groups, and roles, and attach fine-grained **JSON policies** to them, dictating exactly which AWS services and resources they can access (e.g., "Allow read-only access to S3 bucket 'my-sensitive-data'").
- **SSO (Single Sign-On):**
    - An authentication scheme that allows a user to log in with a single ID and password to gain access to multiple related, yet independent, software systems.
    - **Role in Cloud:** Essential for large organizations, enabling users to log into their cloud console (e.g., AWS, Azure) using the same credentials they use for their corporate network (e.g., via a standard like **SAML**). This centralizes identity management and simplifies the user experience.
