

Introduction to Cloud Computing

Understanding the Foundations of
Modern IT Infrastructure

About Me

Md Shahriar Rahman

- Software engineer @ OrangeBD
- 2 years+ industry experience
- AWS certified Cloud Practitioner
- bdapps National Hackathon Winner

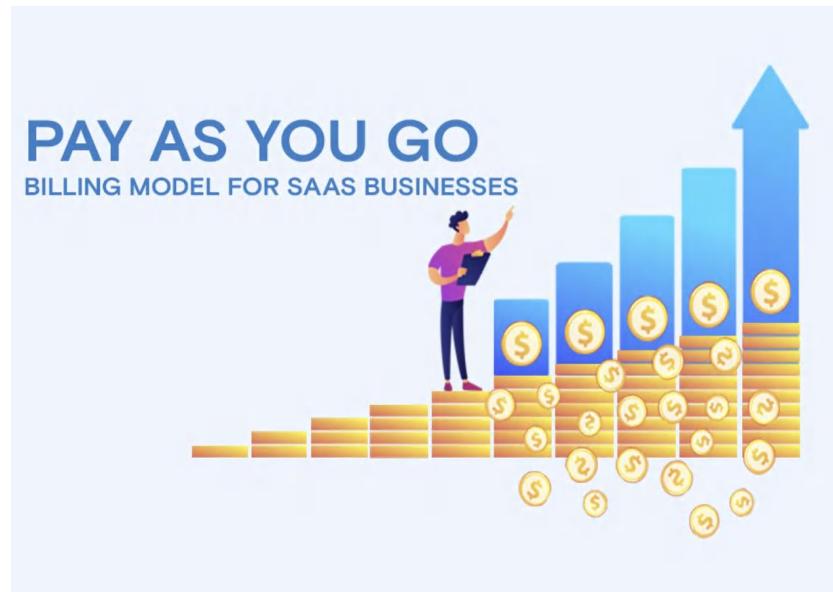
- 2022



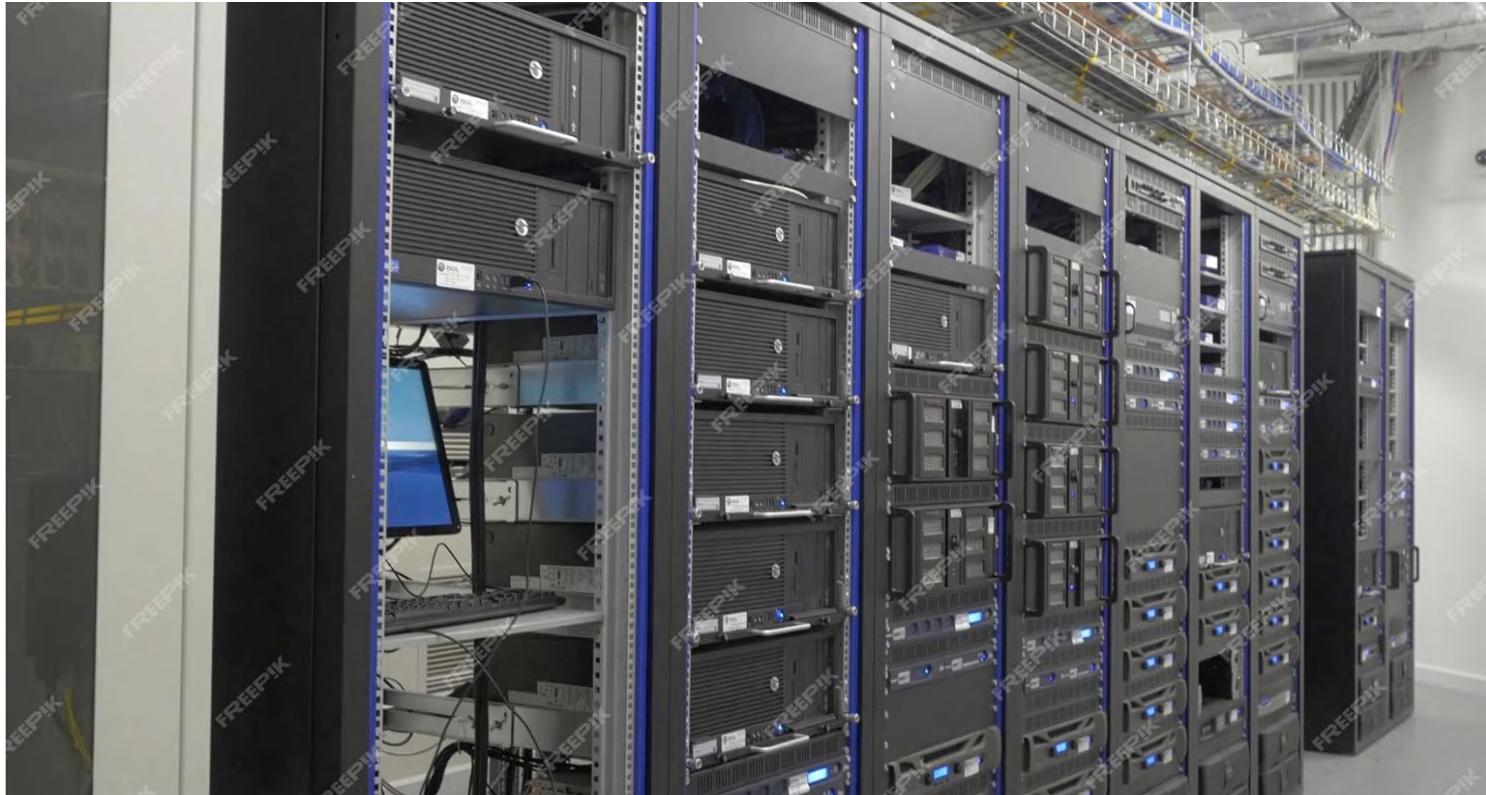
1. Introduction to Cloud Computing

Cloud computing is the delivery of on-demand computing services—such as servers, storage, databases, networking and more—over the internet ('the cloud').

Compare cloud computing to utility services like electricity or water on - demand, scalable, and pay -as-you-go



Server in Server Rack



2. Evolution of Computing

Mainframe Era (1950s-1970s): Centralized computing, limited to large organizations.

Client -Server Model (1980s-1990s): Decentralized computing with personal computers and local servers.

Cloud Era (2000s-Present): On-demand, scalable resources available globally, thanks to advances in virtualization, internet speed, and storage.



Mainframe Era



Client-Server Model



Cloud Era

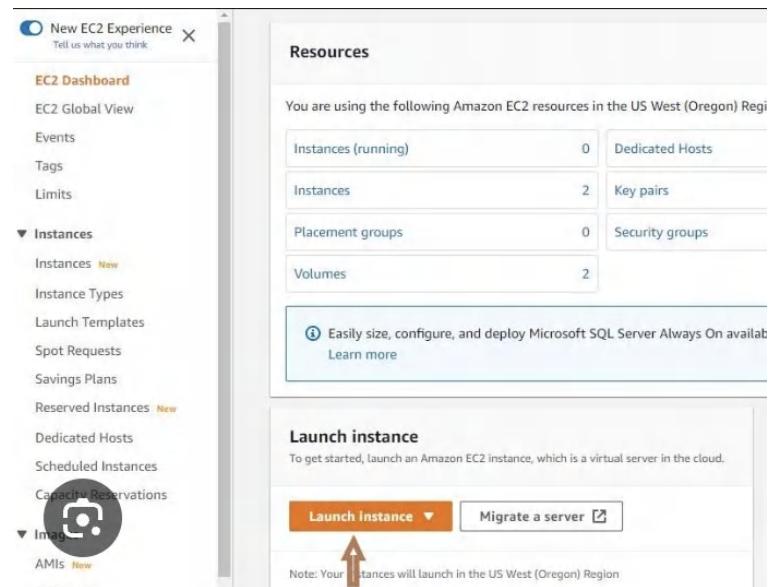
3. Key Characteristics of Cloud Computing

- 1. On-Demand Self -Service** : Users can provision resources automatically.
- 2. Broad Network Access** : Available over the network, accessible from various devices.
- 3. Resource Pooling** : Multi -tenant model with dynamic allocation.
- 4. Rapid Elasticity** : Scale up/down based on demand.
- 5. Measured Service** : Pay only for what you use.

OnDemand Service:

Users can access computing resources like servers and storage automatically, without human interaction with service providers.

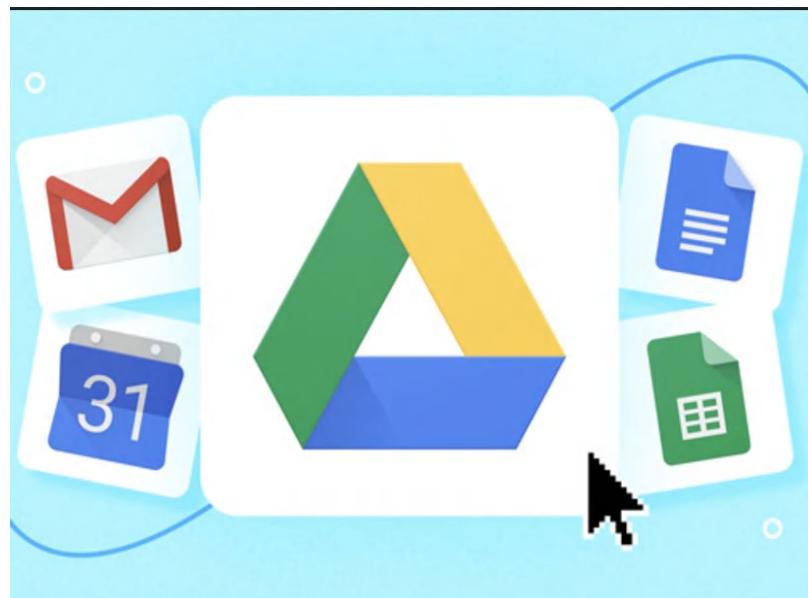
Example : Launching a virtual machine (VM) on AWS.



Broad Network Access

Services are accessible over the network via standard devices such as laptops, smartphones, or tablets .

Example : Accessing Google Drive from any device.

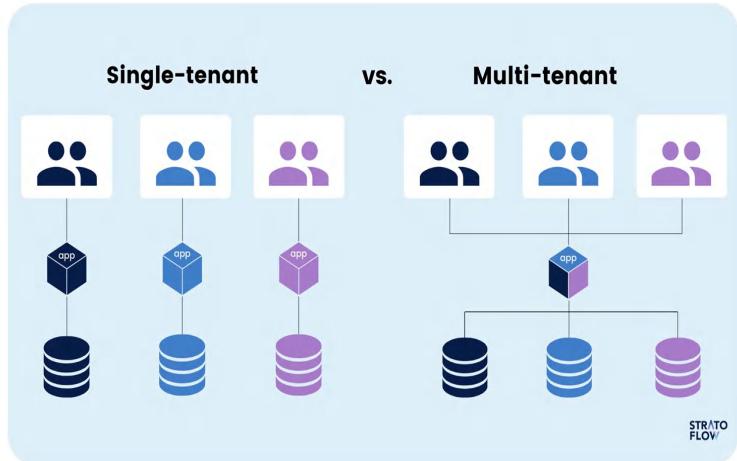


Resource Pooling:

Providers pool resources to serve multiple users using a multi-tenant model.

Resources are dynamically assigned.

Example: AWS uses data centers globally to serve customers seamlessly.

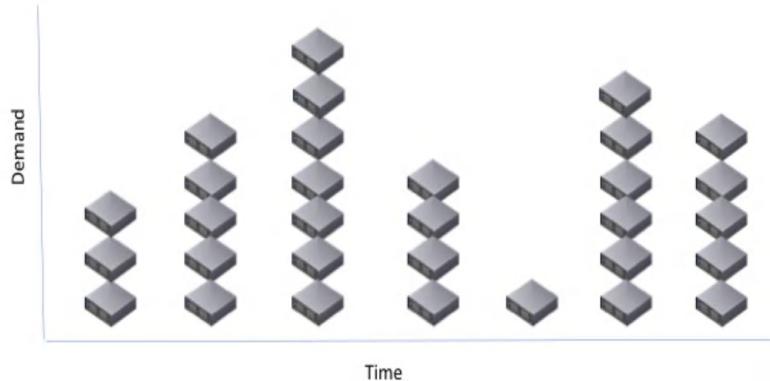


Rapid Elasticity:

Resources can be scaled up or down automatically or manually.

Example : E-commerce platforms scaling up during Eid sales.

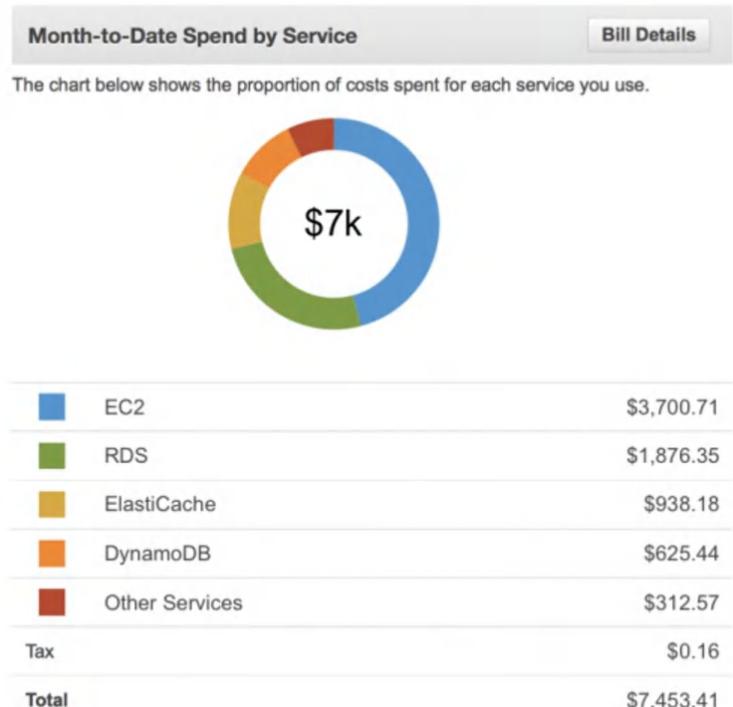
Elasticity in Cloud Computing



Measured Service:

Cloud systems automatically control and optimize resource usage, providing a metering capability.

Example : AWS bills based on the number of compute hours or storage used.



4. Cloud Service Models

- IaaS (Infrastructure as a Service) :

Example : AWS EC2, Google Compute Engine

Provides virtualized computing resources (VMs, storage, etc.)

- PaaS (Platform as a Service) :

Example : AWS Elastic Beanstalk, Google App Engine

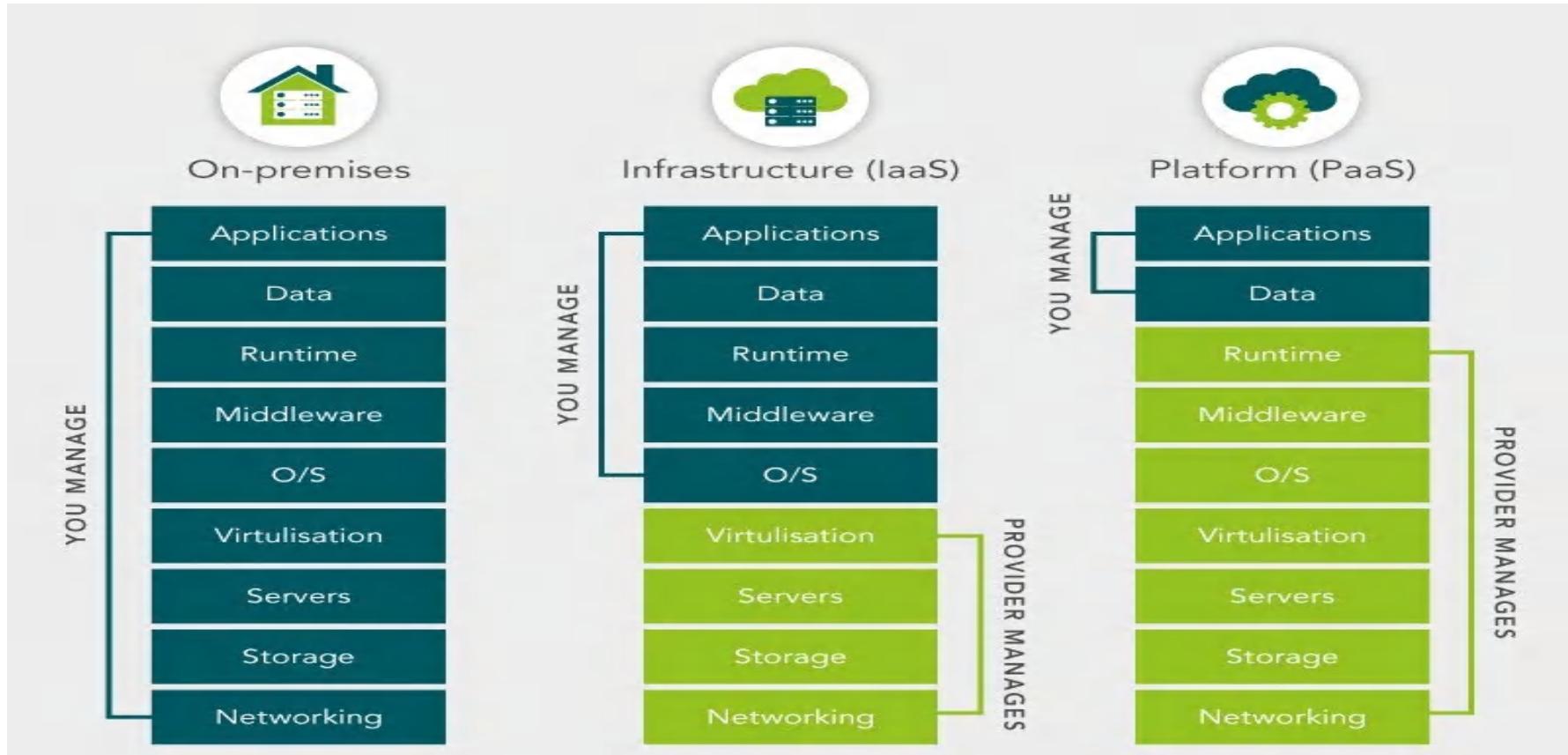
Provides a platform for developing, testing, and deploying applications.

- SaaS (Software as a Service) :

Example : Gmail, Dropbox, Salesforce

Users access software over the internet without managing infrastructure.

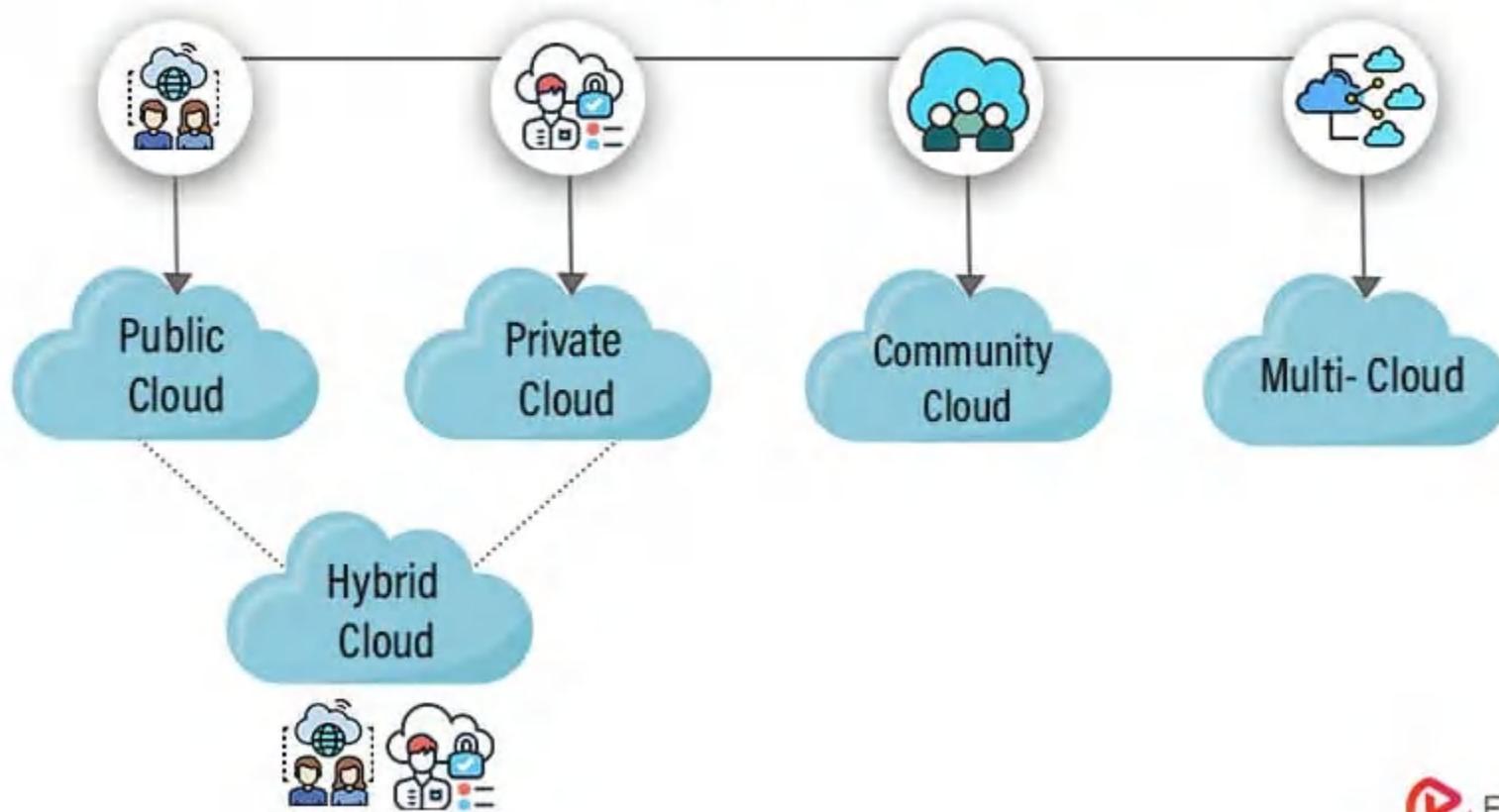
Responsibility Sharing



IaaS vs. PaaS vs. SaaS Examples



Cloud Deployment Models



Cloud Computing Benefits



01 FASTER AND
FLEXIBLE
SCALABILITY

02 REDUCED IT
COST

03 IMPROVED BUSINESS
INTERACTION AND
COLLABORATION

04 ENHANCED
BUSINESS VALUE

05 EASY DATA
BACKUP AND
RESTORE

06 COMPETITIVE
EDGE AND
SUSTAINABILITY

07 FLEXIBLE
STORAGE

08 MOBILITY AND
REMOTE WORK

09 BUSINESS
CONTINUITY WITH
ANYWHERE ACCESS

7. Challenges in Cloud Computing

1. Data security and privacy (Cloud did not used in govt project)
2. Vendor lock -in
3. Downtime risks

Summary

- Cloud computing delivers on-demand IT services over the internet.
- Three service models: IaaS, PaaS, SaaS.
- Four deployment models: Public, Private, Hybrid, Community.
- Benefits include cost efficiency, scalability, and innovation, but challenges like security and vendor lock-in remain.

Thank You

Open for Questions

Connect with me in LinkedIn



Resource Sharing in the Cloud

Characteristics: On -Demand,
Pay-as-You-Go, Multi -Tenancy

Objectives for Today's Class

By the end of this class, students will be able to:

- Understand the concept of resource sharing in cloud computing.
- Explain the key characteristics of cloud computing: on-demand, pay-as-you-go, and multi-tenancy.
- Discuss the advantages and limitations of resource sharing in the cloud.

What is resource in the cloud?

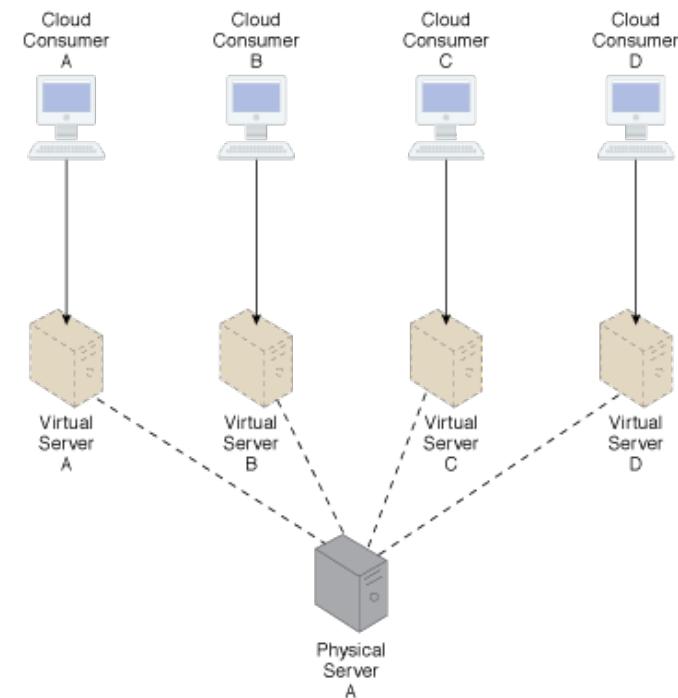
Cloud resources are essential components provided by cloud platforms to build, deploy, and manage applications. They include:

- **Compute Resources** : Virtual Machines (VMs), Containers, Serverless (e.g., AWS Lambda, Azure VMs).
- **Storage Resources** : Object, Block, and File Storage (e.g., AWS S3, Azure Blob).
- **Networking Resources** : Virtual Private Cloud (VPC), Load Balancers, DNS.
- **Database Resources** : Relational (SQL) and NoSQL Databases (e.g., AWS RDS, DynamoDB).
- **Security & IAM** : Access control, Encryption (e.g., AWS IAM, Azure Key Vault).
- **Monitoring & Logging** : Performance tracking, log analysis (e.g., AWS CloudWatch).
- **Development Tools** : CI/CD, API Gateways (e.g., AWS CodePipeline, Azure API Gateway).

What is Resource Sharing in the Cloud?

Resource sharing in cloud computing refers to the **ability to allocate and manage computing resources dynamically among multiple users**, ensuring optimal usage.

Example : Multiple users sharing the same physical server but each running their own isolated virtual machine.



Key Characteristics of Cloud Resource Sharing

- 1. On-Demand Self -Service** : Users can provision resources automatically.
- 2. Broad Network Access** : Available over the network, accessible from various devices.
- 3. Resource Pooling** : Providers pool computing resources to serve multiple consumers using a multi -tenant model

PayasYouGo Model

Cloud users are charged based on their usage of resources rather than a fixed upfront cost.

Benefits :

- Cost efficiency: Pay only for what you use.
- Flexibility: Scale resources up or down according to need.

Example : AWS charges for storage (S3) based on the amount of data stored and transferred.



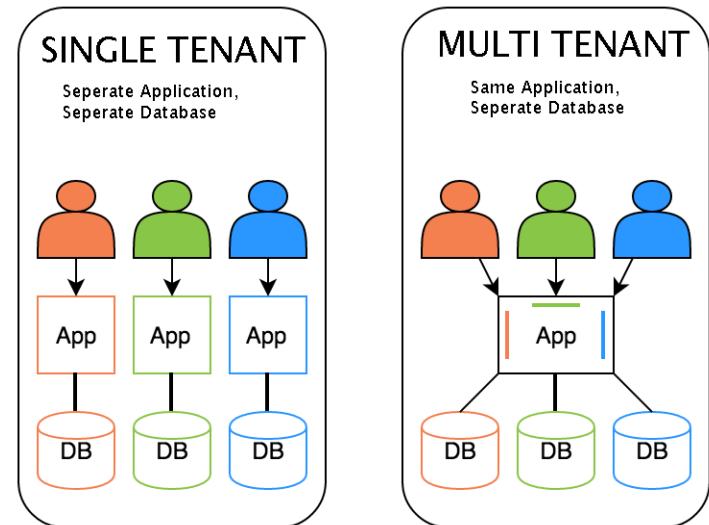
MultiTenancy in Cloud Computing

Multi-tenancy allows multiple customers (tenants) to share the same infrastructure while maintaining data isolation.

Key Benefits :

- Efficient resource utilization.
- Reduced cost per user.

Example : Salesforce provides a single instance of its **CRM** application to multiple companies.



Advantages of Resource Sharing

- 1. Cost Savings:** Reduces the need for physical hardware and maintenance.
- 2. Scalability:** Easily adjust resource allocation based on demand.
- 3. High Availability:** Redundant (অপ্রয়োজনীয়) resources ensure system uptime and reliability.
- 4. Increased Collaboration:** Teams can access shared resources and work seamlessly.

Challenges of Resource Sharing

- 1. Data Security Risks** : Potential exposure of data in shared environments.
- 2. Performance Issues** : Resource contention can occur if many users need resources simultaneously.
- 3. Compliance Issues** : Ensuring regulatory compliance (e.g., GDPR) in shared environments.
- 4. Vendor Lock -in** : Dependence on a single cloud provider can limit flexibility.

Case Study: Netflix on AWS

Scenario : Netflix uses AWS to stream content globally to millions of users.

Resource Sharing : Netflix dynamically scales its resources during peak viewing times.

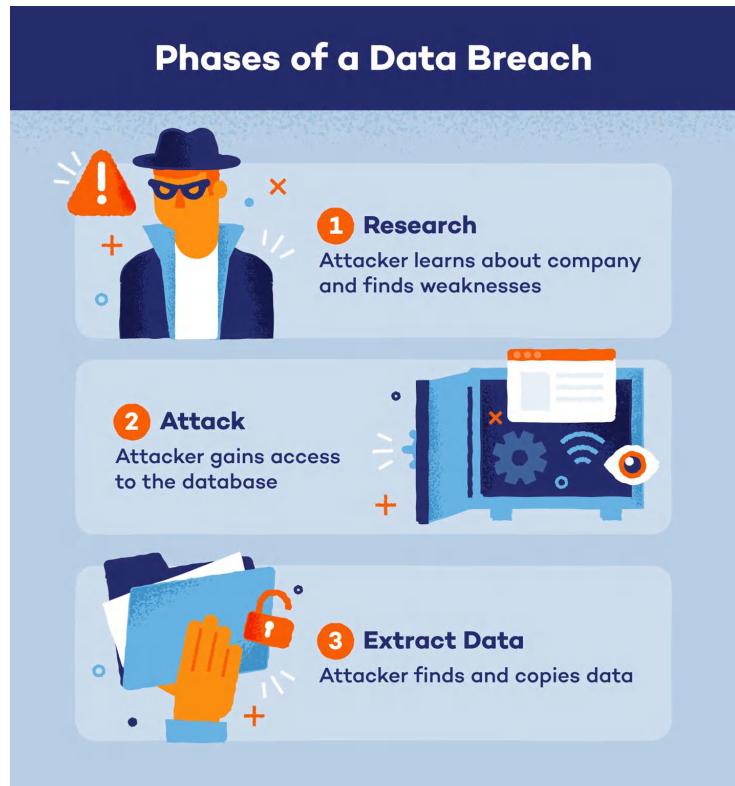
Outcome : Efficient resource allocation ensures high availability and uninterrupted streaming.

Summary

- Resource sharing in cloud computing enables efficient use of computing resources by pooling them across multiple users.
- Key characteristics include on-demand self-service, pay-as-you-go, and multi-tenancy.
- While it offers scalability and cost efficiency, it also poses challenges related to security and performance.

Next Class: Challenges and Risks in Cloud Comp

Discussion on data breaches, vendor lock -in, and regulatory compliance.



Challenges and Risks in Cloud Computing

— Vendor Lock-in, Downtime, Data Privacy —

Objectives

By the end of this class, we will be able to:

1. Identify the main challenges and risks associated with cloud computing.
2. Explain key issues like vendor lock -in, downtime, and data privacy.
3. Discuss strategies to mitigate these risks in cloud environments.

Overview of Challenges and Risks

Cloud Challenges:

1. Vendor Lock -in
2. Downtime and Availability
3. Data Privacy and Security
4. Compliance and Regulatory Issues
5. Performance and Latency
6. Cost Management

1. Vendor Lock

Definition : Difficulty in migrating from one cloud provider to another due to proprietary services, APIs, or data formats.

Impact: Limited flexibility and higher switching costs.

Example : AWS Lambda's proprietary functions can make it difficult to migrate to Azure Functions.

Mitigation Strategies:

- Use open standards and interoperable services.
- Implement multi-cloud strategies to avoid dependence on a single vendor.

2. Downtime and Availability

Definition : Periods when cloud services are unavailable, leading to disruptions in business operations.

Real-World Example : AWS outage in 2020 impacted major platforms like Netflix, Slack, and Twitch.

Mitigation Strategies:

- Utilize multiple availability zones or regions.
- Implement failover and disaster recovery solutions.
- Use Service Level Agreements (SLAs) to ensure uptime guarantees.

3. Data Privacy and Security Risks

Key Issues:

- Unauthorized access to sensitive data.
- Data breaches due to misconfigured storage (e.g., S3 buckets).
- Insider threats and vulnerabilities in multi -tenant environments.

Mitigation Strategies:

- Implement robust encryption for data at rest and in transit.
- Use multi -factor authentication (MFA) and Identity and Access Management (IAM).
- Regularly audit and monitor for unusual activities.

4. Compliance and Regulatory Challenges

Overview : Ensuring cloud services comply with legal regulations like GDPR, HIPAA, and PCI-DSS.

Challenges :

- Data sovereignty: Where data is stored and processed.
- Meeting industry-specific compliance requirements.

Mitigation Strategies:

- Choose cloud providers that offer compliance certifications.
- Understand the shared responsibility model for cloud compliance.

5. Performance and Latency Issues

Definition : Delays in data processing and delivery due to geographical distances and network limitations.

Impact : Affects user experience in latency -sensitive applications like video streaming and gaming.

Mitigation Strategies:

- Use Content Delivery Networks (CDNs) to cache data closer to users.
- Optimize application architecture for cloud environments.

6. Cost Management Challenges

Key Issues:

- Unpredictable costs due to dynamic resource scaling.
- Overspending on unused or underutilized resources.

Mitigation Strategies:

- Implement cost monitoring tools like AWS Cost Explorer or Azure Cost Management.
- Use reserved instances or savings plans for predictable workloads.
- Regularly review and optimize resource allocation.

RealWorld Case Study

Case: Capital One Data Breach (2019)

Incident : Misconfigured firewall exposed sensitive customer data on AWS.

Impact : Data breach affected over 100 million customers.

Lessons Learned:

- Importance of proper configuration and continuous monitoring.
- Need for comprehensive security policies in cloud environments.

Summary of Key Points

- Cloud Challenges and Risks: Vendor lock -in, downtime, data privacy, compliance, and cost management.
- Effective mitigation strategies include multi -cloud adoption, robust security measures, cost control, and using open standards.
- Understanding and planning for these challenges are essential for successful cloud adoption.

Next Class Preview

- Next Topic: Cloud Service Models (IaaS, PaaS, SaaS)
 - Deep dive into infrastructure, platform, and software services.

Cloud Service Models

— Understanding IaaS, PaaS, and SaaS —

Introduction to Cloud Service Models

Definition:

Service models define how cloud services are delivered to users, based on the level of control and management responsibility.

Key Service Models:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

1. Infrastructure as a Service (IaaS)

Provides virtualized computing resources over the internet.

Users manage operating systems, applications, and data while the cloud provider manages hardware and virtualization.

Examples:

- Amazon Web Services (AWS) EC2
- Google Compute Engine
- Microsoft Azure Virtual Machines

Key Features:

- Virtual machines, storage, and networks.
- High flexibility and scalability.



2. Platform as a Service (PaaS)

Provides a platform that allows developers to build, test, deploy, and manage applications without worrying about the underlying infrastructure.

The cloud provider manages OS, runtime, and middleware.

Examples:

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

Key Features:

- Simplifies development and deployment.
- Ideal for application development without infrastructure management.



3. Software as a Service (SaaS)

Delivers software applications over the internet on a subscription basis.

Users access applications via a web browser while the cloud provider manages everything .

Examples :

- Gmail
- Microsoft Office 365
- Salesforce

Key Features:

- No installation required.
- Accessible from anywhere with an internet connection.



Key Differences Between Service Models

Aspect	IaaS	PaaS	SaaS
User Responsibility	Apps, data, OS	Apps, data	Only usage
Provider Responsibility	Hardware, virtualization	Hardware, OS, runtime	Everything
Use Case	Full Control and flexibility	Development platforms	End-user applications

Advantages and Disadvantages

IaaS:

- Advantages: High control and scalability.
- Disadvantages: Complex management required.

PaaS:

- Advantages: Simplifies app development.
- Disadvantages: Limited control over the environment.

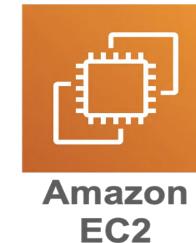
SaaS:

- Advantages: Easy to use and maintain.
- Disadvantages: Limited customization.

Case Study Comparing Real World Examples

IaaS Example:

- Use AWS EC2 to host a website where you manage the server, storage, and OS.



PaaS Example:

- Use Heroku to develop and deploy a web application.



SaaS Example:

- Use Gmail for email management.



Identify the Service Model

Example 1: Hosting a virtual machine on Azure.

Example 2: Building a web app on AWS Elastic Beanstalk.

Example 3: Using Dropbox to store files.

Identify the Service Model

Example 1: Hosting a virtual machine on Azure. (Answer: IaaS)

Example 2: Building a web app on AWS Elastic Beanstalk. (Answer: PaaS)

Example 3: Using Dropbox to store files. (Answer: SaaS)

Summary

Cloud service models offer varying levels of control and abstraction.

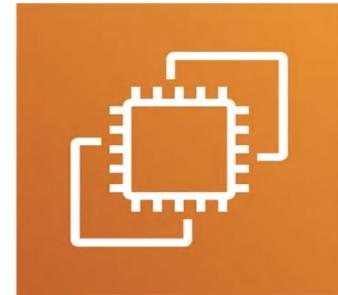
- IaaS: High control over infrastructure.
- PaaS: Simplifies app development.
- SaaS: End-user applications delivered over the internet.

IaaS (Amazon) EC

— Unlocking Scalable Cloud Computing —

What is Amazon EC2?

- **Amazon Elastic Compute Cloud (EC2)** : A web service offering scalable virtual servers.
- **Primary Goal** : Simplify the process of provisioning and managing computing resources.
- **Key Capabilities** : Flexibility, scalability, and cost-effectiveness for deploying applications.



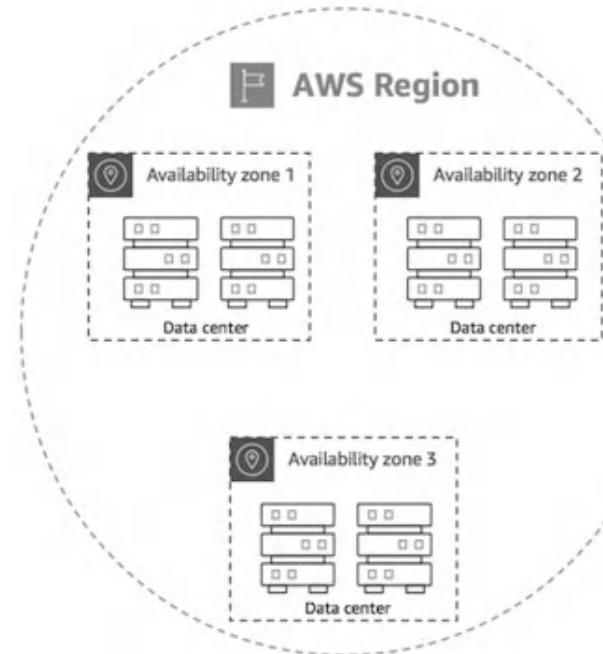
EC2 Architecture Overview

1. Regions and Availability Zones :

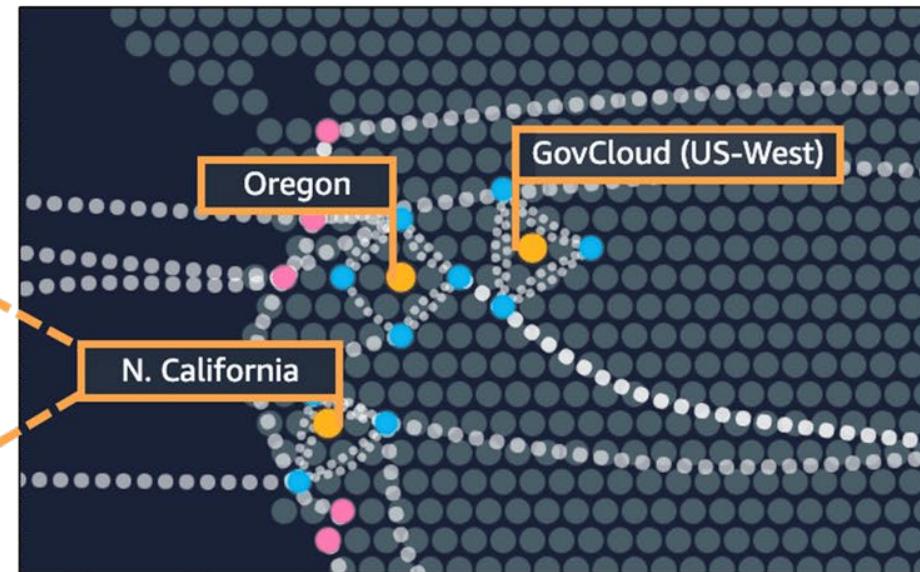
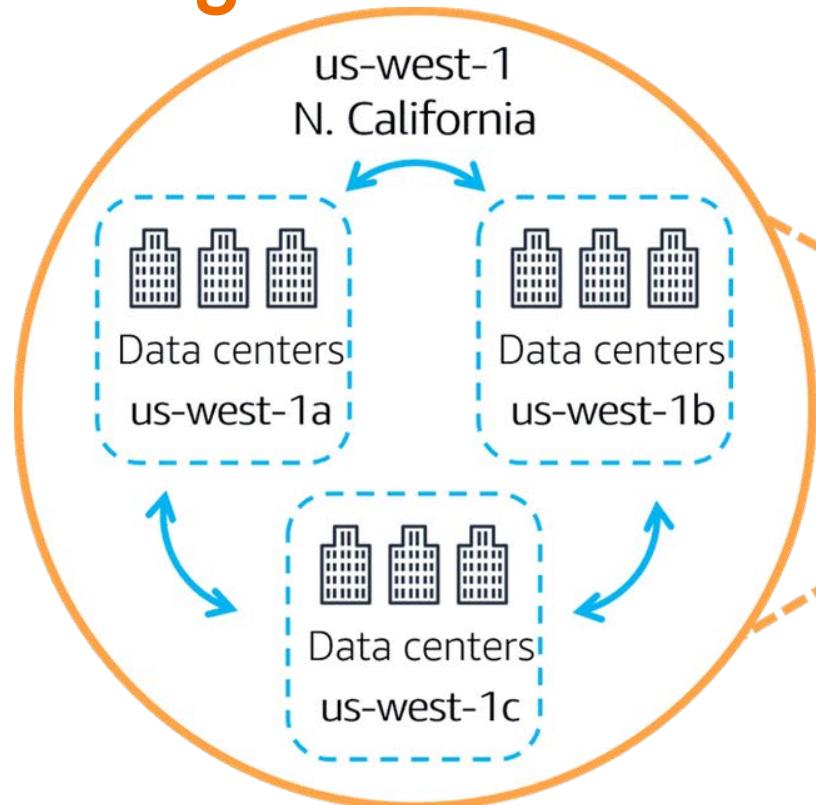
- **Regions** : Geographically separate areas (e.g., US-East-1, AP-South-1).
- **Availability Zones (AZs)** : Isolated data centers within regions for fault tolerance.

2. **Virtual Machines (Instances)** : EC2 provides virtualized servers.

3. **Elastic Block Store (EBS)** : Persistent storage for EC2 instances.



Regions vs Availability Zone vs Data Center

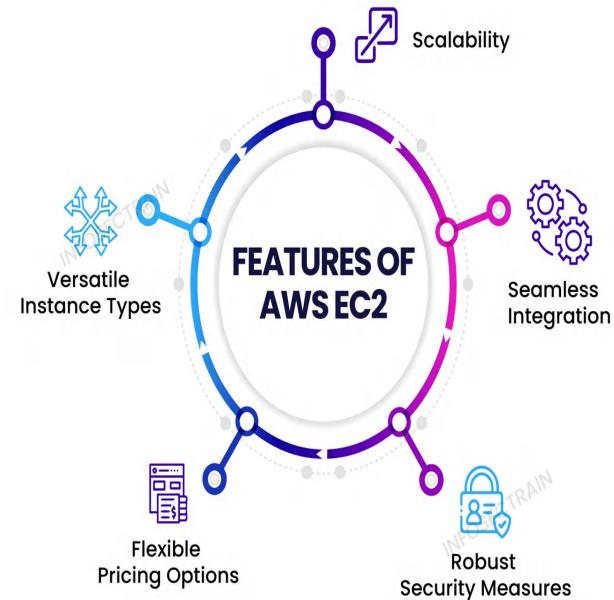


Orange circle: Regions

Blue circle: Availability Zones

Key Features of Amazon EC

- Elasticity** : Scale instances up/down based on workload.
- Customizable Configurations** : Wide selection of OS, instance types, and storage options.
- Cost Management** : Multiple pricing models to optimize expenses.
- Networking** : Secure, isolated environments via Virtual Private Cloud (VPC).
- Integration** : Works seamlessly with AWS services like S3, RDS, Lambda.



EC2 Instance Types

1. General Purpose : Balanced performance (e.g., t2.micro).
2. Compute Optimized : For compute -intensive workloads (e.g., c5.large).
3. Memory Optimized : For memory -intensive applications (e.g., r5.large).
4. Storage Optimized : For high-performance storage needs (e.g., i3.large).
5. Accelerated Computing : For GPU-based workloads (e.g., p3.large).

Different AWS EC2 Instance Types



General
Purpose



Compute-
Optimized



Memory-
Optimized



Storage-
Optimized



Accelerated
Computing

M and T
families



C-type
family



R and X
families



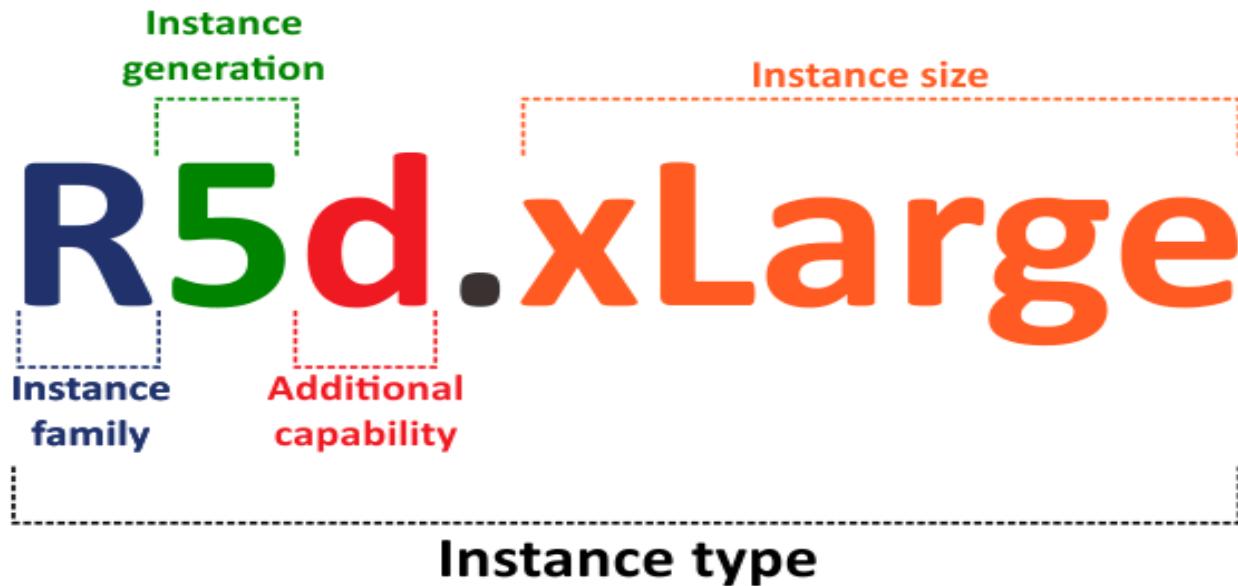
I, D, and H
families



P, G, D, F, and
V families



AWS EC2 instance naming



Use Cases of EC

1. **Web Hosting:** Host websites and applications.
2. **Big Data Analytics:** Analyze large datasets efficiently.
3. **Machine Learning:** Train and deploy ML models.
4. **Gaming:** Power online multiplayer games.
5. **DevOps :** Test and deploy code in scalable environments.



EC2 – Pricing Model

On-Demand Least Commitment

- low cost and flexible
- only pay per hour
- short-term, spiky, unpredictable workloads
- cannot be interrupted
- For first time apps

Spot upto 90% Biggest Savings

- request spare computing capacity
- flexible start and end times
- Can handle interruptions (server randomly stopping and starting)
- For non-critical background jobs

Reserved upto 75% off Best Long-term

- steady state or predictable usage
- commit to EC2 over a 1 or 3 year term
- Can resell unused reserved instances

Dedicated Most Expensive

- Dedicated servers
- Can be on-demand or reserved (upto 70% off)
- When you need a guarantee of isolate hardware (enterprise requirements)

Security Features of EC2

- **IAM Integration:** Fine-grained access control.
- **Security Groups:** Virtual firewalls to control instance traffic.
- **Key Pairs:** Encrypt connections via SSH.
- **VPC:** Isolated networking environment.
- **Data Encryption:** Encrypt data at rest (EBS, S3) and in transit (TLS).

Storage Options for EC2

1. **Elastic Block Store (EBS)** : Persistent storage for single instances.
2. **Instance Store** : Temporary storage that persists only during instance lifecycle.
3. **Elastic File System (EFS)** : Scalable file storage for multiple instances.
4. **S3 Integration** : Object storage for backups and archives.

E2 Auto Scaling

Purpose : Automatically adjust compute resources to meet demand.

Benefits :

- Reduces costs by scaling down during low demand.
- Ensures high availability during traffic spikes.

Components :

- **Scaling Groups** : Define instance pools.
- **Scaling Policies** : Rules for scaling up/down.

Amazon EC2 and Networking

1. **Elastic IP Addresses** : Static IP addresses for EC2 instances.
2. **Load Balancers** :
 - Application Load Balancer (ALB): For HTTP/HTTPS traffic.
 - Network Load Balancer (NLB): For TCP traffic.
1. **VPC Peering** : Connect multiple VPCs for seamless communication.

Getting Started with EC2

1. Log in to the AWS Management Console.
2. Launch an instance:
 - Choose AMI (e.g., Ubuntu, Windows Server).
 - Select instance type (e.g., t2.micro).
1. Configure storage and networking.
2. Connect to the instance using SSH or RDP.

Advantages of EC2

- Flexibility to choose from various configurations.
- Global scalability with low latency.
- Integration with other AWS services (e.g., S3, RDS).
- Cost-effectiveness with multiple pricing models.

Summary

- Amazon EC2 provides scalable, flexible, and cost-effective compute power.
- Suitable for diverse use cases, from startups to large enterprises.
- Key to leveraging cloud computing benefits in modern applications.

Amazon S3

— The Backbone of Scalable Cloud Storage —

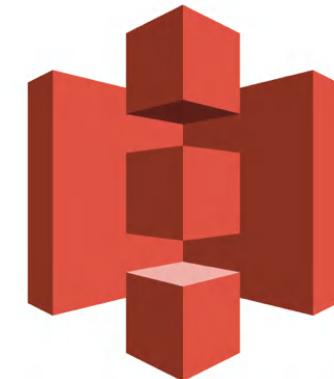
What is Amazon S3?

Amazon Simple Storage Service (S3): A scalable object storage service by AWS.

Purpose : Store and retrieve any amount of data from anywhere.

Use Cases:

- Backup and restore
- Data archiving
- Big data analytics
- Hosting static websites



Amazon S3

Core Concepts

Bucket:

- A container for storing objects.
- Each bucket has a unique name within the AWS ecosystem.
- Supports versioning, encryption, and access controls.

Object:

- Files stored in buckets, accompanied by metadata.
- Each object is identified by a unique key.

Key:

- A string used to identify an object within a bucket.

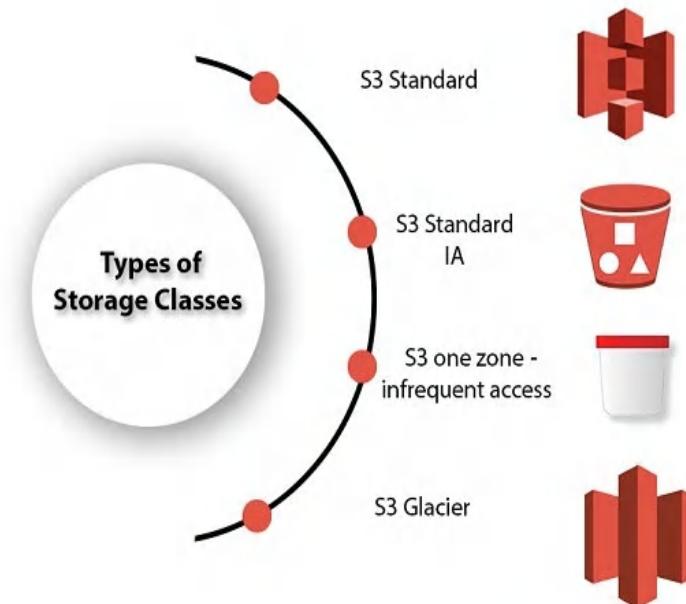
Key Features of Amazon S

- **Scalability** : Automatically scales storage capacity.
- **Durability** : 99.999999999 % (11 9's) durability.
- **Availability** : High uptime and regional redundancy.
- **Security** : Encryption and fine -grained access control.
- **Performance** : Low-latency and high - throughput storage.



S3 Storage Classes

- **Standard** : High durability, availability, and performance for frequently accessed data.
- **Standard -IA (Infrequent Access)**: Cheaper storage for less frequently accessed data.
- **One Zone -IA**: Cost-effective for infrequently accessed data stored in a single Availability Zone.
- **Glacier** : Low-cost storage for long -term archival.
- **Intelligent -Tiering**: Automatically moves data between cost -effective tiers.



AWS S3 Lifecycle Management



S3 Lifecycle Configuration Example

How Amazon S3 Works

1. Create a Bucket:

- Use the AWS Console, CLI, or SDK to create a bucket.
- Name it uniquely within the AWS global namespace.

1. Upload Objects:

- Files are stored in the bucket as objects with metadata.

1. Access Objects:

- Retrieve files using REST API, SDKs, or presigned URLs.

Benefits of Amazon S

- **Reliability** : Built on AWS global infrastructure.
- **Cost-Effectiveness:** Pay only for storage used; no upfront costs.
- **Flexibility** : Supports a wide range of use cases, from backups to web hosting.
- **Ease of Use** : Easy to manage and configure with AWS tools.

Security Features

Data Encryption:

- Server-Side Encryption (SSE): Encrypts data at rest.
- Client-Side Encryption: Data encrypted before uploading.

Access Management:

- IAM Policies: Control who can access what.
- Bucket Policies: Define access rules for buckets.
- S3 Access Points: Simplify managing access to shared buckets.

Compliance :

- Meets regulatory standards like GDPR, HIPAA, and PCI DSS.

RealWorld Use Cases

1. Data Backup and Archival:

Reliable storage for disaster recovery.

1. Static Website Hosting:

Cost-effective for hosting HTML, CSS, and JavaScript files.

1. Big Data Analytics:

Stores large datasets for tools like AWS Athena or Amazon EMR.

1. Media Storage:

Host and distribute video, images, and other media files globally.

Cost and Pricing Model

- **Storage Cost** : Based on GB/month.
- **Data Transfer** : Outbound transfer incurs costs, inbound is free.
- **Request Pricing** :
 - Costs for GET, PUT, DELETE operations.
- **Glacier Retrieval Costs** : Vary by retrieval speed (Instant, Expedited, Standard).

S3 Best Practices

- **Enable Versioning:** Protect against accidental deletion.
- **Use Lifecycle Policies:** Optimize storage costs.
- **Secure Your Data:** Enforce encryption and IAM best practices.
- **Monitor Usage:** Use CloudWatch for monitoring and cost tracking.

Summary

- Amazon S3 is a foundational service for AWS cloud computing.
- **Key Takeaways** : Scalable, durable, secure, and cost-effective storage.
- **Encouragement** : Explore S3 by creating a free-tier AWS account.

AWS IAM (Identity and Access Management)

— Securing AWS Resources with Roles, Policies, and Permissions —

Objectives

- Understand the purpose of IAM in AWS
- Learn key IAM components: Users, Groups, Roles, and Policies
- Explore IAM security best practices
- Hands-on: Creating and managing IAM entities

What is IAM?

Definition : IAM allows secure management of access to AWS services and resources

Key Features:

- Manage users and their access
- Granular permissions
- Identity federation for external access
- Free service (pay only for used AWS resources)

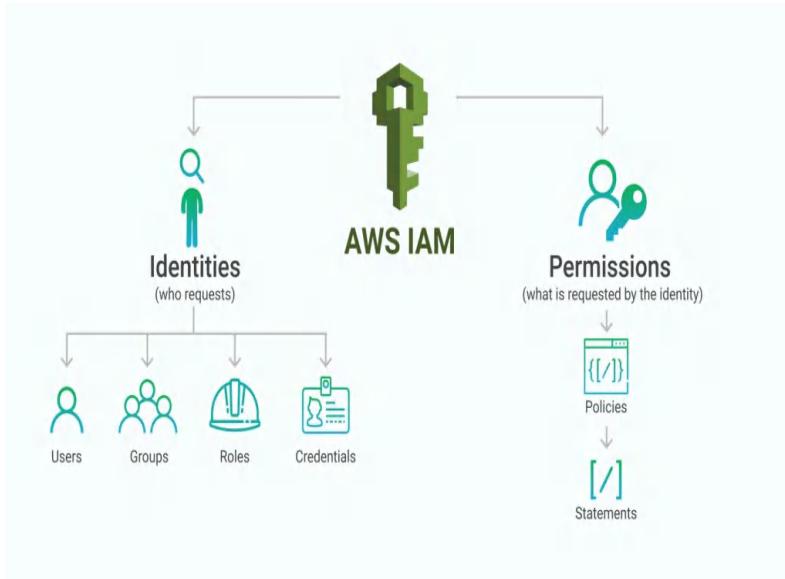
Key IAM Concepts

Users : Individual identities with credentials to access AWS

Groups : Collections of users sharing the same permissions

Roles : Temporary access to AWS resources, used by applications or services

Policies : JSON documents defining permissions



IAM Policies Explained

Types of Policies:

- AWS Managed Policies (predefined)
- Customer Managed Policies (customizable)
- Inline Policies (attached directly to an entity)

Structure:

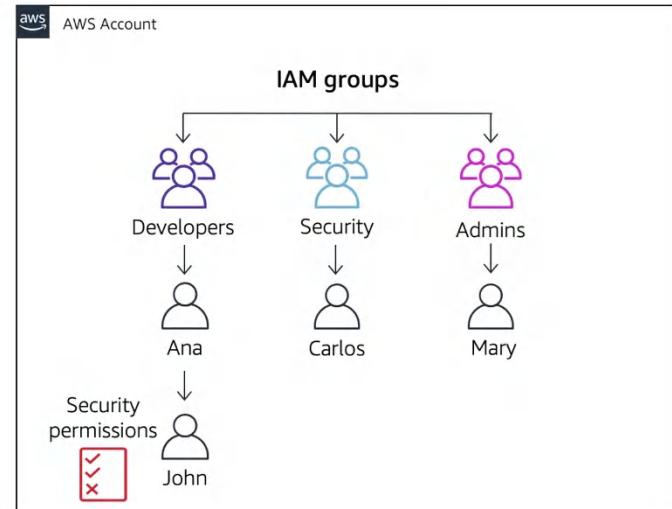
- Actions: What can be done (e.g., s3>ListBucket)
- Resources: Where the action is applied (e.g., arn:aws:s3:::bucket -name)
- Effect: Allow or Deny

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "FirstStatement",  
      "Effect": "Allow",  
      "Action": ["iam:ChangePassword"],  
      "Resource": "*"  
    },  
    {  
      "Sid": "SecondStatement",  
      "Effect": "Allow",  
      "Action": "s3>ListAllMyBuckets",  
      "Resource": "*"  
    }  
  ]  
}
```

IAM Components Relationship

Diagram showing how users, groups, roles, and policies interact:

- Users belong to groups
- Groups have attached policies
- Roles are assumed by users or AWS services



HandsOn: Creating an IAM User

Objective : Create a user with specific permissions

Steps :

- Go to the IAM Dashboard
- Click "Add User"
- Assign access type (console or programmatic)
- Attach a policy (e.g., AmazonS3ReadOnlyAccess)
- Download the access credentials



HandsOn: Creating a Group

Objective : Create a group with shared permissions

Steps :

- Create a group in IAM
- Attach a policy (e.g., AmazonEC2FullAccess)
- Add users to the group

IAM: Groups



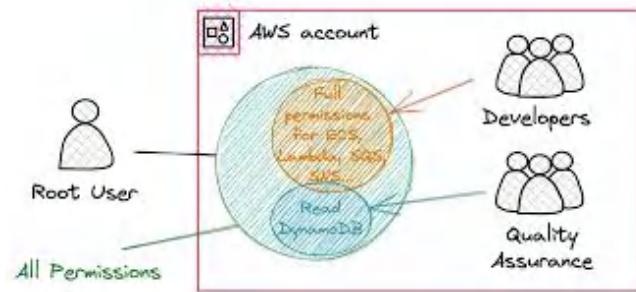
HandsOn: Creating a Role

Objective : Grant temporary access to a service
(e.g., EC2 accessing S3)

Steps:

Create a role in IAM

- Select a use case (e.g., AWS Service > EC2)
- Attach a policy (e.g., AmazonS3FullAccess)
- Assign the role to an EC2 instance



IAM Security Best Practices

- Enable Multi -Factor Authentication (MFA)
- Follow the principle of least privilege
- Rotate access keys regularly
- Use IAM roles instead of sharing credentials
- Monitor activity with **CloudTrail**

RealWorld Use Cases

Developers : Programmatic access to deploy applications

Administrators : Manage accounts, services, and users

Applications : Access S3, DynamoDB, or other resources using roles

Summary

- IAM is essential for securing AWS resources
- Key components: Users, Groups, Roles, and Policies
- Always implement security best practices

AWS VPC (Virtual Private Cloud)

Building Secure and Scalable
Cloud Networks

What is Amazon VPC?

Amazon Virtual Private Cloud (VPC) is a **logically isolated section** of the AWS Cloud where you can define your virtual network.

Purpose :

- Launch AWS resources in a secure and controlled environment.
- Customize networking to meet specific requirements.

Use Cases:

- Hosting web applications securely.
- Extending on-premises data centers to the cloud.
- Creating multi-tier architectures.



Key Benefits of Amazon VPC

Isolation : Fully isolate resources for enhanced security.

Flexibility : Customize IP addressing, subnets, and routing.

Security : Leverage security groups and network ACLs.

Scalability : Expand resources as needed without manual reconfiguration.



Core Components of a VPC

1. **Subnets** : Public and private sub -networks within a VPC.
2. **Route Tables**: Rules for directing traffic within the network.
3. **Internet Gateway (IGW)**: Access the internet from your VPC.
4. **NAT Gateway**: Allow private instances to initiate internet connections.
5. **Elastic IP (EIP)**: Static public IP for resources.

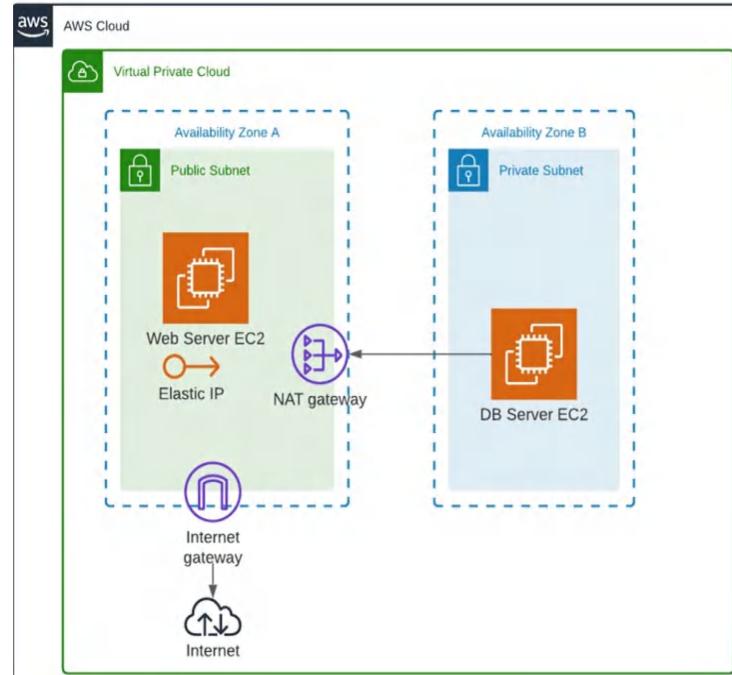
Public and Private Subnets

Public Subnet:

- Direct internet access via the Internet Gateway (IGW).
- Ideal for web servers and other public-facing resources.

Private Subnet:

- No direct internet access.
- Used for sensitive resources like databases and backend systems.



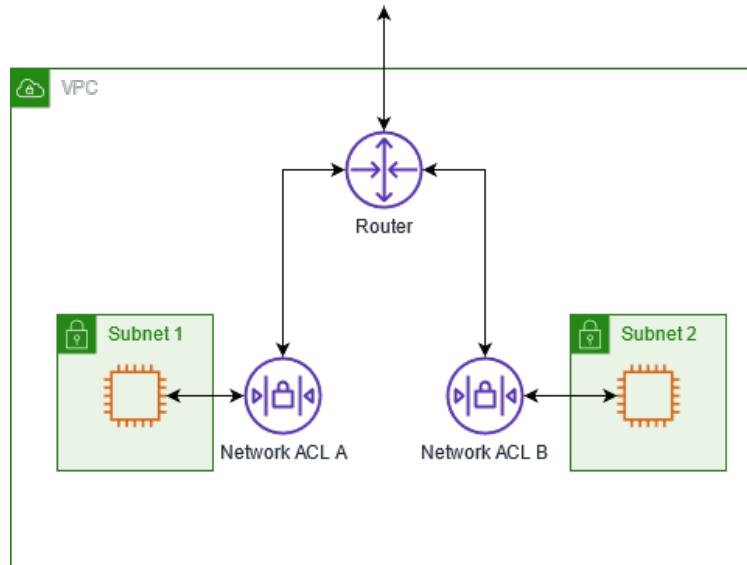
Security Features in VPC

Security Groups:

- Instance-level firewall.
- Controls inbound/outbound traffic.

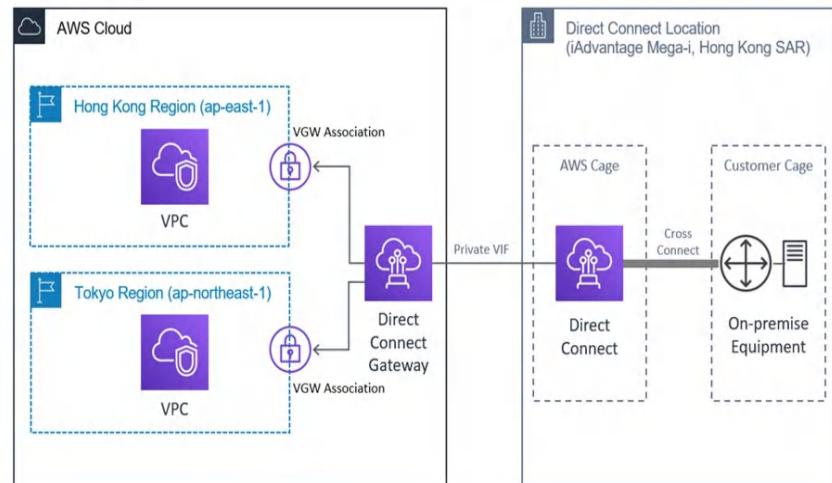
Network ACLs (Access Control Lists):

- Subnet-level firewall.
- Allows/denies traffic at the subnet level.



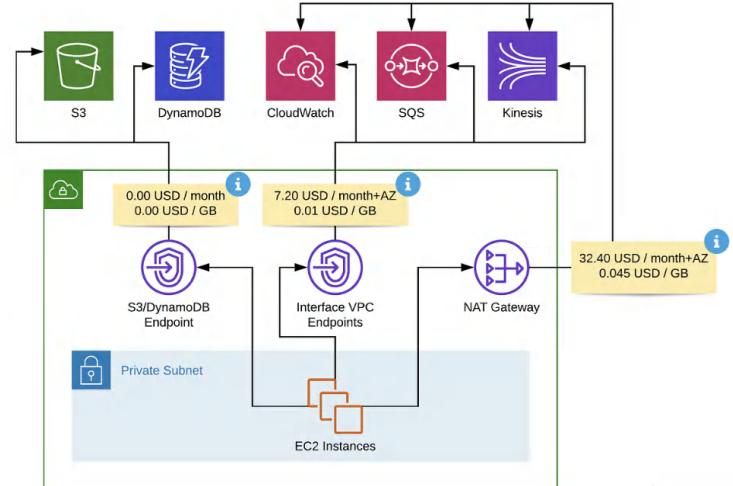
Connecting to VPC

1. Internet Gateway (IGW): Enables internet access for public subnets.
2. VPN and Direct Connect: Securely connect your on-premises network to AWS.
3. Peering Connections: Connect multiple VPCs for resource sharing.



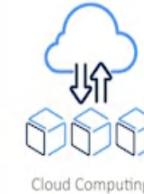
Advanced Networking in VPC

1. **NAT Gateway:** Allow instances in private subnets to access the internet without exposing them.
2. **Elastic Load Balancer (ELB):** Distribute traffic across instances in public and private subnets.
3. **VPC Endpoints:** Privately connect VPC to AWS services without using IGW or NAT.



VPC Use Cases

1. **Web Hosting:** Deploy scalable, secure web applications.
2. **Hybrid Cloud:** Extend on-premises networks using VPN or Direct Connect.
3. **Data Analytics:** Run secure data analysis workflows.



VPC Best Practices

1. Design subnets for specific use cases (public, private, isolated).
2. Use least privilege principles with security groups and ACLs.
3. Monitor traffic using VPC Flow Logs.
4. Leverage NAT Gateways for secure internet access from private subnets.

Summary

- VPC provides isolated, secure, and customizable networking in AWS.
- Plan and design the VPC architecture based on application needs.
- Utilize security and monitoring tools to safeguard resources.

Introduction to Linux

— Exploring the Foundation of
Open Source Operating Systems —

What is Linux?

- Linux is an **open -source** and **Unix** *-like operating system.
- Created by Linus Torvalds in 1991.
- Used in diverse fields: Servers, desktops, embedded systems, and more.
- Key Point: Linux powers over **90% of servers** on the internet.

*UNIX is a multi-user, multitasking operating system (OS) that was developed in the late 1960s by Bell Laboratories at AT&T

Brief History of Linux

- Unix (1969): Inspiration for Linux.
- GNU Project (1983): Free software foundation started by Richard Stallman.
- 1991: Linus Torvalds announced Linux Kernel.
- Open Source Revolution: Collaboration among developers worldwide.

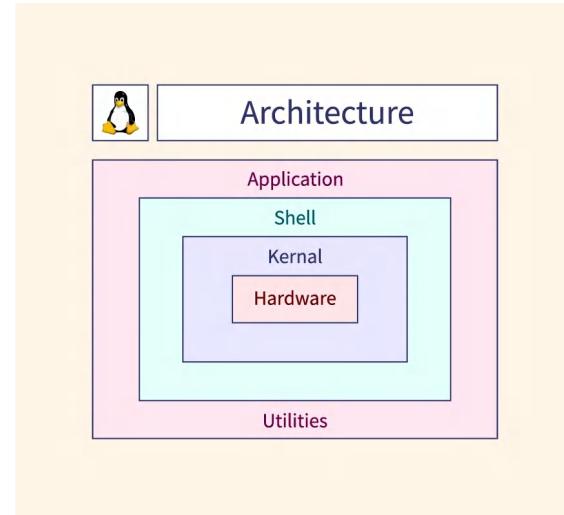
Core Features of Linux

- **Open Source** : Freely available and customizable.
- **Multiuser Support** : Many users can work simultaneously.
- **Portability:** Runs on a wide range of hardware.
- **Security:** Built -in user and process isolation.
- **Stability:** Reliable for long -term use.



Linux Architecture

- 1. Hardware** : Physical components of the computer.
- 2. Kernel** : The core that interacts with hardware.
- 3. System Libraries** : Provide essential functionality.
- 4. Shell** : Command -line interface for user interaction.
- 5. Applications** : Software running on top of the OS.



Linux Distributions

What is a Distribution?

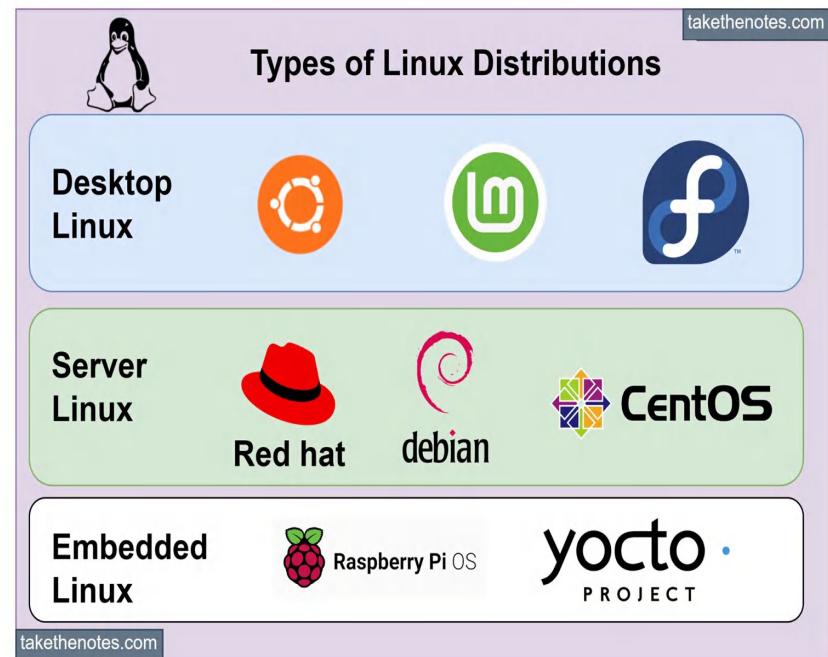
A collection of the Linux kernel, software packages, and tools.

Popular Distributions :

- **Ubuntu** : User-friendly.
- **Fedora** : Cutting-edge software.
- **Debian** : Stability and reliability.
- **Arch Linux** : Customizable and minimalist.

Specialized lightweight distros :

- Tiny Core Linux (~16 MB)
- Puppy Linux (~300 MB)



Why Choose Lightweight Linux Distros?

Designed for systems with low RAM and storage.

Suitable for:

- Old computers
- Embedded systems
- Minimalistic environments

Examples:

- Tiny Core Linux: ~16 MB
- Damn Small Linux: ~50 MB
- AntiX: ~400 MB
- Bodhi Linux: ~700 MB

Basic Linux Commands

Navigating the File System:

- ls: List files.
- cd: Change directory.
- pwd: Print working directory.

File Management:

- mkdir: Create a directory.
- rm: Remove a file or directory.
- cp: Copy files.

System Monitoring:

- top: View running processes.
- df -h: Check disk usage.
- free -m: Check memory usage.

RealWorld Applications of Linux

- **Web Servers:** Powering Google, Facebook, and Twitter.
- **Supercomputers:** Runs on 100% of the world's top 500 supercomputers.
- **IoT Devices:** Used in smart devices and embedded systems.
- **Programming:** Preferred platform for developers due to robust tools.
- **Gaming:** Platforms like SteamOS bring gaming to Linux.

Linux Desktop Environments

- **GNOME:** Modern and simple.
- **KDE Plasma:** Highly customizable.
- **XFCE:** Lightweight and fast.
- **LXQt:** Minimal resource usage.
- **Mate:** Traditional desktop feel.

Myths and Misconceptions

- "**Linux is hard to use**" → Modern distros are beginner-friendly.
- "**No software is available**" → Vast libraries of applications.
- "**Only experts use Linux**" → Distros like Ubuntu and Linux Mint are for everyone.

Advantages of Linux

- **Cost:** Free to use.
- **Customizability :** Tailor it to your needs.
- **Performance :** Runs efficiently on all hardware.
- **Security :** Resilient against malware and viruses.
- **Community :** Active global support.

Challenges of Linux

- Learning curve for beginners.
- Software compatibility: Some proprietary software may not run.
- Hardware drivers: Rare cases of unsupported hardware.

Conclusion

- Linux is a versatile, secure, and open -source OS.
- Powers much of today's technology and infrastructure.
- Encouragement: Start small, experiment with a lightweight distro.
- “The Linux community is always there to support you.”

AWS Auto Scaling and Load Balancing

Optimizing Scalability and Availability in the Cloud

Agenda

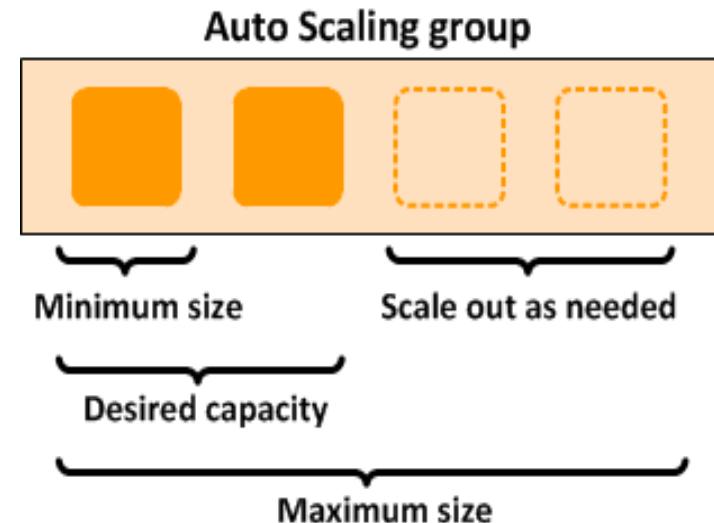
- Auto Scaling
- Load Balancing
- Key Features and Benefits
- How They Work Together
- Use Cases
- Real-World Examples
- Demo (Optional)
- Summary

Auto Scaling

Automatically adjusts the number of compute resources to meet demand.

Key Benefits:

- Ensures application availability.
- Reduces costs by scaling down during low traffic.
- Improves performance during traffic spikes.

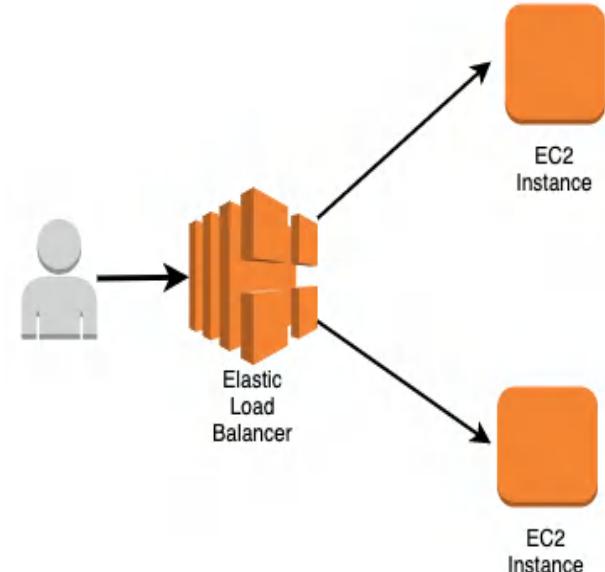


Load Balancing

Distributes incoming traffic across multiple resources to prevent overloading any single resource.

Types of Load Balancers in AWS:

1. **Application Load Balancer (ALB):** Layer 7 (HTTP/HTTPS) routing.
2. **Network Load Balancer (NLB):** Layer 4 (TCP/UDP) routing.
3. **Classic Load Balancer:** Legacy support for Layer 4 and 7.



Key Features and Benefits

Auto Scaling Features:

- **Dynamic Scaling:** Automatically adjusts resources based on traffic.
- **Predictive Scaling:** Uses machine learning to predict demand patterns.
- **Scheduled Scaling:** Pre-defined scaling based on known traffic times.

Load Balancing Features:

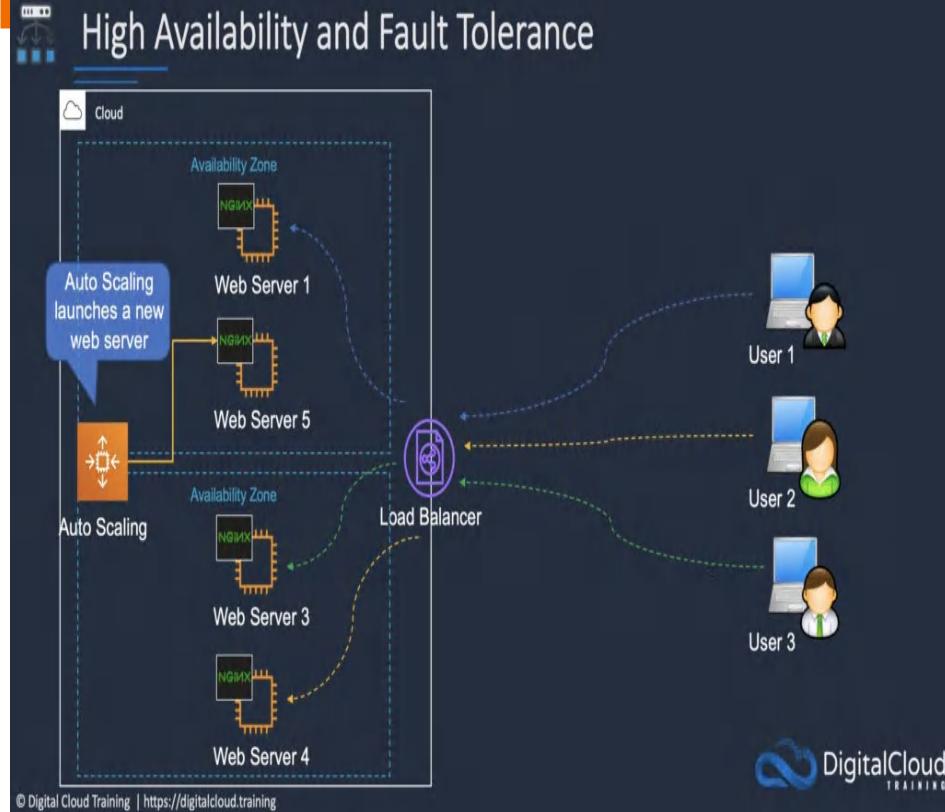
- **Health Checks:** Ensures traffic is sent only to healthy instances.
- **Sticky Sessions:** Keeps user sessions connected to the same instance.
- **SSL Termination:** Manages SSL decryption to offload instance workload.
- **Cross-Zone Load Balancing:** Distributes traffic evenly across zones.

How They Work Together

Scenario : High traffic to a web application.

- Load Balancer distributes traffic to multiple instances.
- Auto Scaling adds more instances as traffic increases.
- When traffic decreases, Auto Scaling reduces instances to save costs.

Outcome : Improved performance, reduced costs, and no downtime.



Use Case

E-commerce Platforms: Handle seasonal traffic spikes during sales events.

Media Streaming: Maintain smooth streaming by scaling resources based on viewer demand.

Web Applications: Ensure high availability and seamless user experiences.

RealWorld Example

Case Study:

- A streaming service like Netflix uses **Auto Scaling** to dynamically allocate resources during new show launches.
- **Load Balancers** ensure traffic is evenly distributed among servers to prevent crashes.

Metrics:

- 99.99% uptime.
- Significant cost savings during off-peak hours.

Demo Step

Step 1: Create an Auto Scaling group in the AWS Management Console.

Step 2: Attach an Application Load Balancer.

Step 3: Configure **scaling policies** and **health checks** .

Step 4: Monitor scaling activities in **CloudWatch** .

Summary

- Auto Scaling ensures cost -efficient scalability by adjusting resources dynamically.
- Load Balancing enhances application availability and distributes traffic efficiently.
- Together, they form the backbone of resilient cloud architectures.

Amazon SQS & SNS

Reliable Messaging and Notifications
for Modern Applications

Amazon SQS (Simple Queue Service)

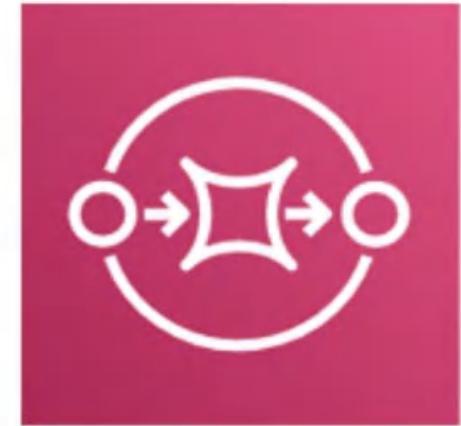
What is Amazon SQS?

- Fully managed message queuing service .
- Decouples* and scales microservices and serverless applications.

Why Use SQS?

- Reliable, scalable, and cost-effective message delivery .
- Handles large workloads and asynchronous communication .

*Services don't need to directly talk to each other. One sends a message to the queue, and the other picks it up when ready



Amazon SQS

Key Features of SQS

- Fully managed and serverless.
- Two queue types: **Standard** (high throughput) and **FIFO** (strict order).
- Secure with **AWS KMS** encryption.
- Supports message retention from 1 minute to 14 days.

Types of Queues

Standard Queue:

- High throughput.
- At-least-once delivery (potential duplication).
- Best-effort ordering (not strict).

FIFO Queue:

- Limited throughput.
- Exactly-once delivery.
- Strict message order.

How SQS Works

1. Message Producers **send messages** to the queue.
2. Messages are **temporarily stored** in the SQS Queue.
3. Message Consumers **poll and process** the messages.
4. Processed messages are **deleted** explicitly.

SQS Use Cases

- Decoupling microservices.
- Asynchronous task execution.
- Buffering for burst traffic.
- Background jobs.

Security and Monitoring

IAM Policies: Control access to SQS.

Encryption: Server-side with AWS KMS.

Monitoring: Amazon CloudWatch for metrics like message count and age.

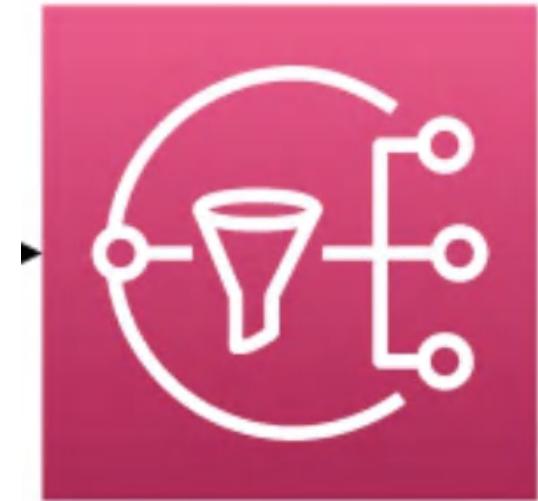
Amazon SNS (Simple Notification Service)

What is Amazon SNS?

- Fully managed publish/subscribe messaging service.
- Enables event -driven and real -time communication.

Why Use SNS?

- High fan -out for messages.
- Supports multiple protocols (HTTP, Lambda, SMS, etc.).



Key Features of SNS

- Topic-based publish/subscribe model.
- Message delivery to multiple subscribers.
- Integration with SQS, Lambda, and email/SMS.
- Message filtering and attributes.

How SNS Works

1. Publishers **send messages** to a topic.
2. SNS distributes the message to **subscribers** .
3. Supports **multiple delivery** protocols (HTTP, SQS, Lambda, SMS, etc.).

SNS Use Cases

- Real-time event notifications.
- Application alerts (e.g., system failures).
- Distributing updates to multiple systems.

Security and Monitoring

IAM Policies: Restrict who can publish/subscribe to topics.

Encryption: Data at rest and in transit.

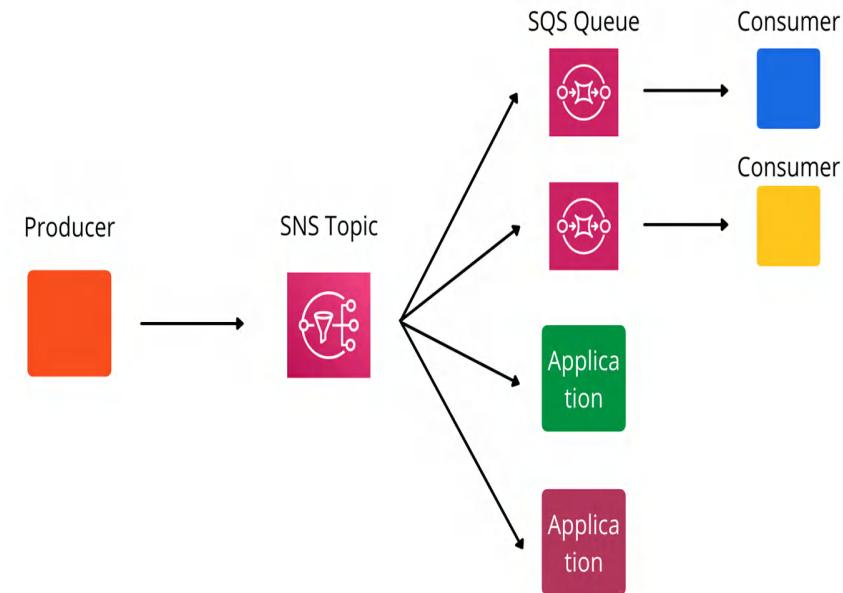
Monitoring: CloudWatch metrics for delivery success and failures.

Using SQS and SNS Together

SNS can fan-out messages to multiple SQS queues.

Benefits:

- Decouple publishers from consumers.
- Reliable delivery with SQS buffering.



Practical Example

Scenario:

- SNS topic publishes **system updates** .
- Multiple SQS queues process updates for different microservices.

Advantages of Integration:

- Scalable and fault -tolerant architecture.
- Combines the strengths of both services (real -time and reliable messaging).

Summary

Amazon SQS: Reliable queuing for asynchronous messaging.

Amazon SNS: Real-time notifications with publish/subscribe.

Integration allows decoupling and scaling distributed systems.

AWS Route 53 (DNS)

— Enhancing Domain Management
and Global Traffic Optimization —

Agenda

- Introduction to Route 53
- Key Features and Benefits
- Routing Policies
- Hosted Zones and Record Types
- Integration with AWS Services
- Use Cases
- Real-World Examples
- Demo (Optional)
- Summary

Amazon Route 53

A highly available and scalable Domain Name System (DNS) web service from AWS.

Primary Functions:

- Domain registration.
- DNS routing.
- Health checks and traffic management.

Why Route 53?

- Global availability.
- Seamless integration with AWS services.
- Highly reliable and secure.



Key Features and Benefits

Features:

- **Domain Registration:** Register and manage domain names directly from AWS.
- **DNS Management:** Create and manage DNS records for domains.
- **Health Checks:** Monitor the health of resources and reroute traffic if needed.
- **Traffic Flow:** Optimize traffic globally with routing policies.
- **Integration:** Works seamlessly with other AWS services.

Benefits:

- **High Availability:** Built-in redundancy for reliable DNS service.
- **Scalability:** Supports traffic for large-scale applications.
- **Security:** DNSSEC support for secure DNS management.

Routing Policies

1. **Simple Routing:** Basic DNS resolution for a single resource.
2. **Weighted Routing:** Distribute traffic based on assigned weights.
3. **Latency Routing:** Directs traffic to the region with the lowest latency.
4. **Geolocation Routing:** Routes traffic based on user location.
5. **Failover Routing:** Ensures high availability by redirecting traffic to a backup resource.
6. **Multi -Value Answer:** Returns multiple IP addresses for redundancy.

Hosted Zones and Record Types

Hosted Zones:

- **Public Hosted Zone** : Manages DNS records for domains accessible on the internet.
- **Private Hosted Zone** : Manages DNS records for internal AWS resources.

Record Types:

- **A (Address Record)** : Maps domain to an IPv4 address.
- **AAAA**: Maps domain to an IPv6 address.
- **CNAME**: Alias for another domain name.
- **MX**: Mail exchange server.
- **TXT**: Holds text information for verification or other purposes.

Integration with AWS Services

Works seamlessly with services like:

- Elastic Load Balancer (ELB) for distributing traffic.
- CloudFront for content delivery.
- S3 for static website hosting.

Enables advanced configurations:

- Automatically updates DNS records for Auto Scaling groups.
- Secure domain connections with ACM (provides SSL).

Use Cases

E-commerce Platforms:

- Efficiently handle global traffic with **geolocation routing**.

Global Applications:

- Ensure low-latency access using **latency-based routing**.

Disaster Recovery:

- Maintain availability using **failover routing**.

RealWorld Example

A global SaaS provider uses Route 53 for:

- Managing domains across regions.
- Ensuring high availability with failover routing.
- Optimizing user experience with latency -based routing.

Metrics:

- Improved global accessibility.
- 99.99% uptime for critical applications.

Demo

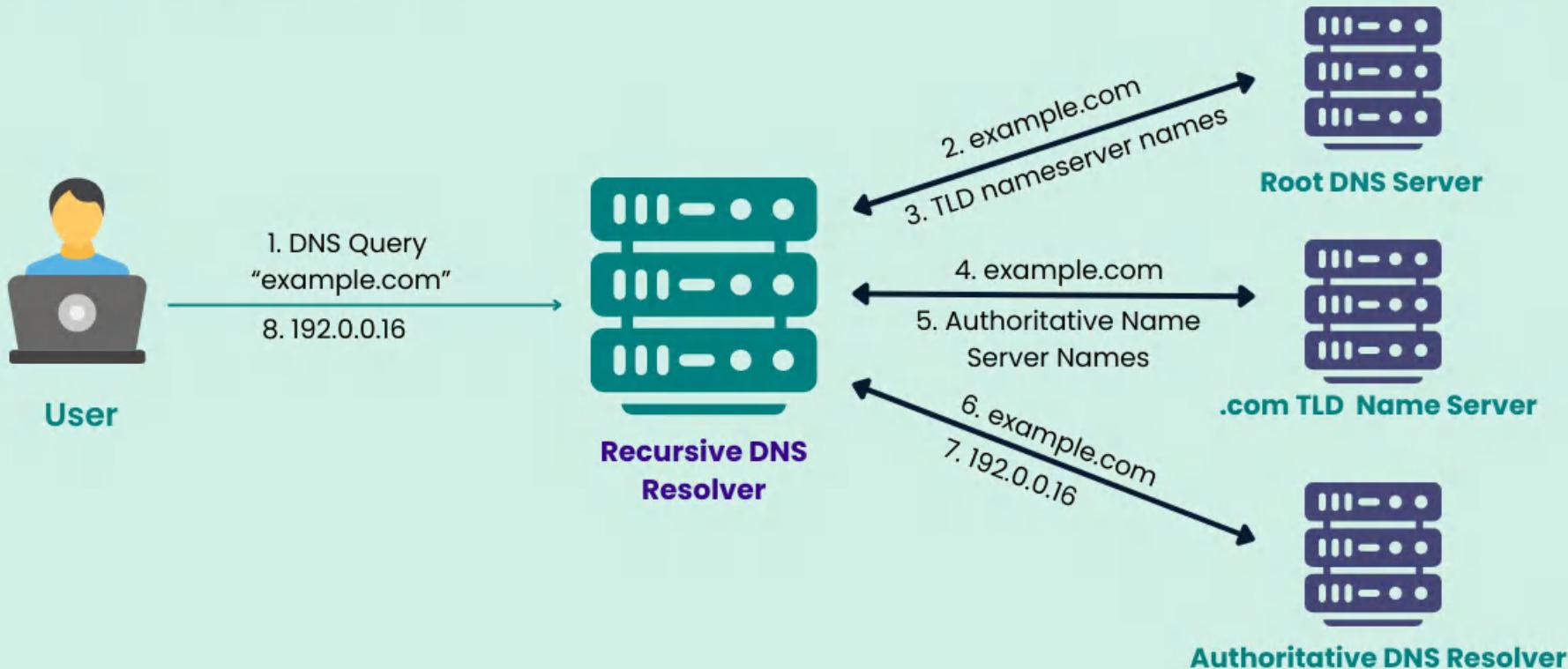
Step 1: Create a hosted zone in the AWS Management Console.

Step 2: Add DNS records (A, CNAME, etc.).

Step 3: Configure routing policies.

Step 4: Test DNS resolution using tools like nslookup.

DNS Lookup Process

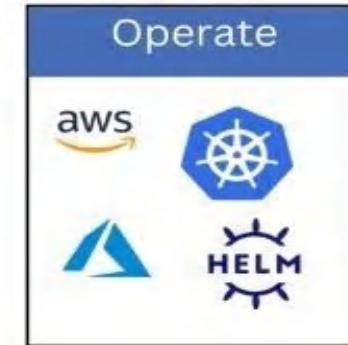
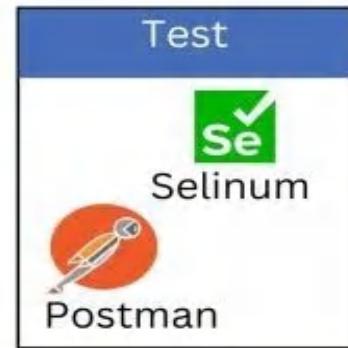
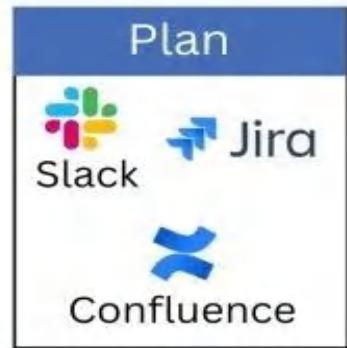


Summary

- Route 53 offers a scalable and reliable DNS service.
- Supports advanced traffic management with various routing policies.
- Seamless integration with AWS services ensures global optimization.

AWS SES (Simple Email Service)

— Email Sending Simplified with AWS —



What is Amazon SES?

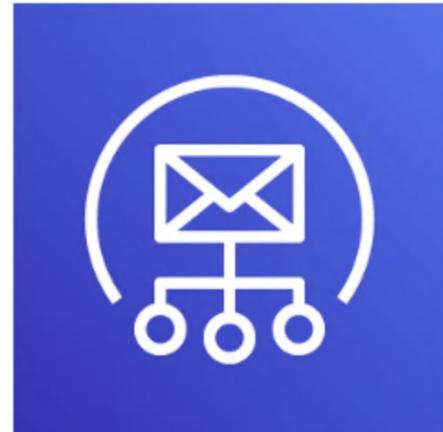
A cloud-based **email sending service** for sending transactional and marketing emails.

Features:

- High deliverability
- Scalable
- Reliable

Use Cases:

- Notifications
- Marketing campaigns
- Transactional emails (e.g., OTPs)



Key Features of Amazon SES

- Highly scalable and cost -effective.
- Support for bulk and transactional emails.
- Built -in analytics (open, click -through, and bounce rates).
- Integration with other AWS services like Lambda and SNS.

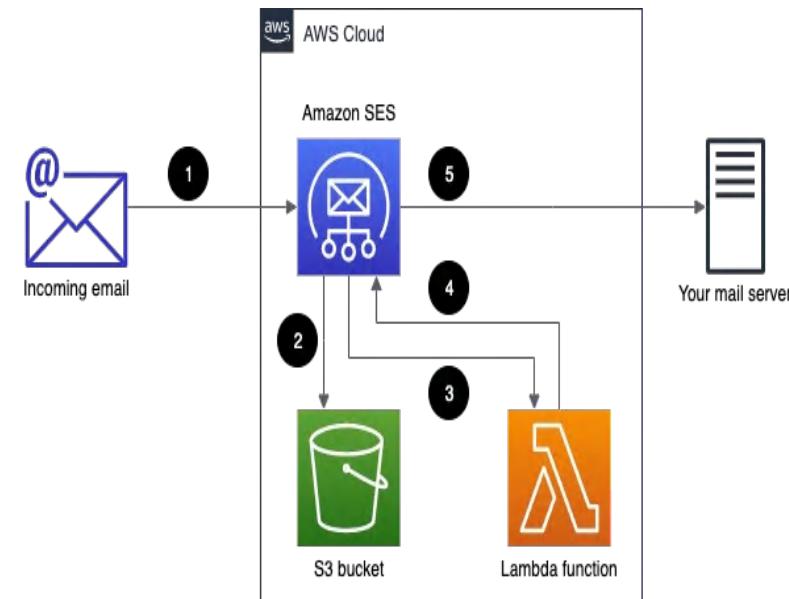


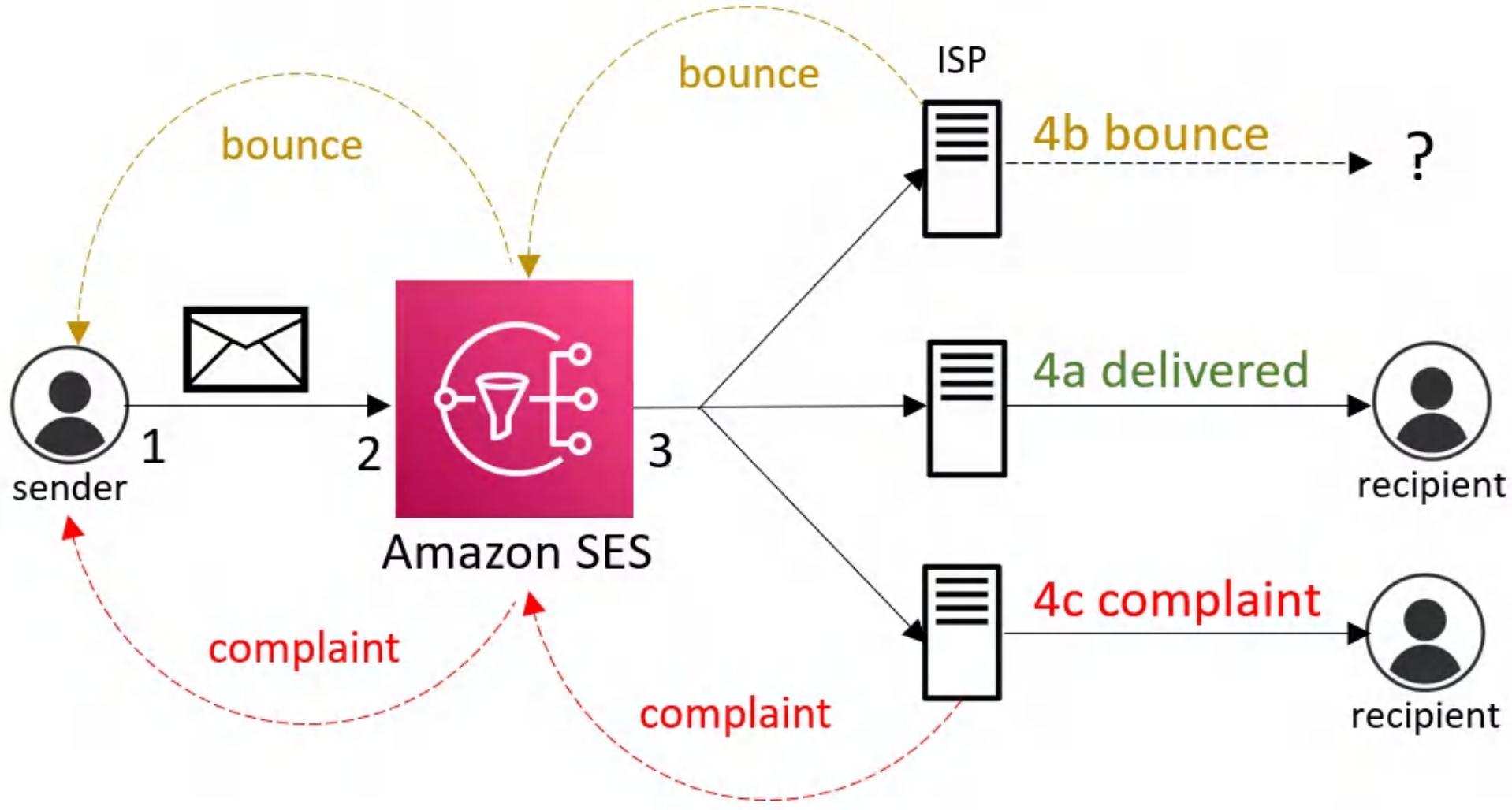
How Amazon SES Works

1. Basic flow: Verify domain/email → Create SMTP credentials → Send email.
2. Identity types: Domain vs. Email Address.
3. Authentication methods: SPF, DKIM, and DMARC.
 1. **DKIM** ensures that an **email has not been altered** in transit and verifies the sender's authenticity by attaching a cryptographic signature.
 2. **SPF** allows the **domain owner to specify which mail servers are authorized** to send emails on behalf of their domain.
 3. **DMARC** (Domain-based Message Authentication, Reporting, and Conformance) is an **email authentication protocol designed to protect your domain from spoofing and phishing attacks**. It builds on SPF and DKIM to provide a policy framework for email authentication and reporting.

Sending Emails with AWS SES

- Sending emails via SMTP Interface.
- Using AWS SDKs (e.g., Boto3 for Python).
- Limits and quotas.





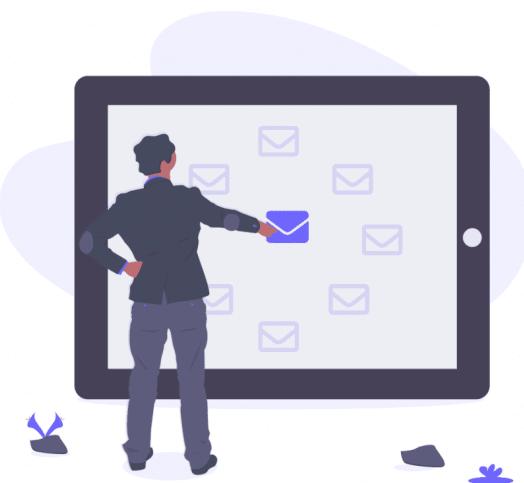
SES Pricing

- Emails sent: \$0.10 per 1,000 emails
- Attachments: Cost for data transfer applies
- **Free Tier:** 62,000 emails per month when sent from an EC2 instance

Email service	Cost per 10,000 emails
MailChimp	\$200
Campaign Monitor	\$105
Amazon SES	\$1

SES Best Practices

- Use DKIM and SPF for email authentication.
- Monitor bounce and complaint rates.
- Regularly review email metrics.
- Using IAM for access control.



Question: Sending Bulk Emails to Customers

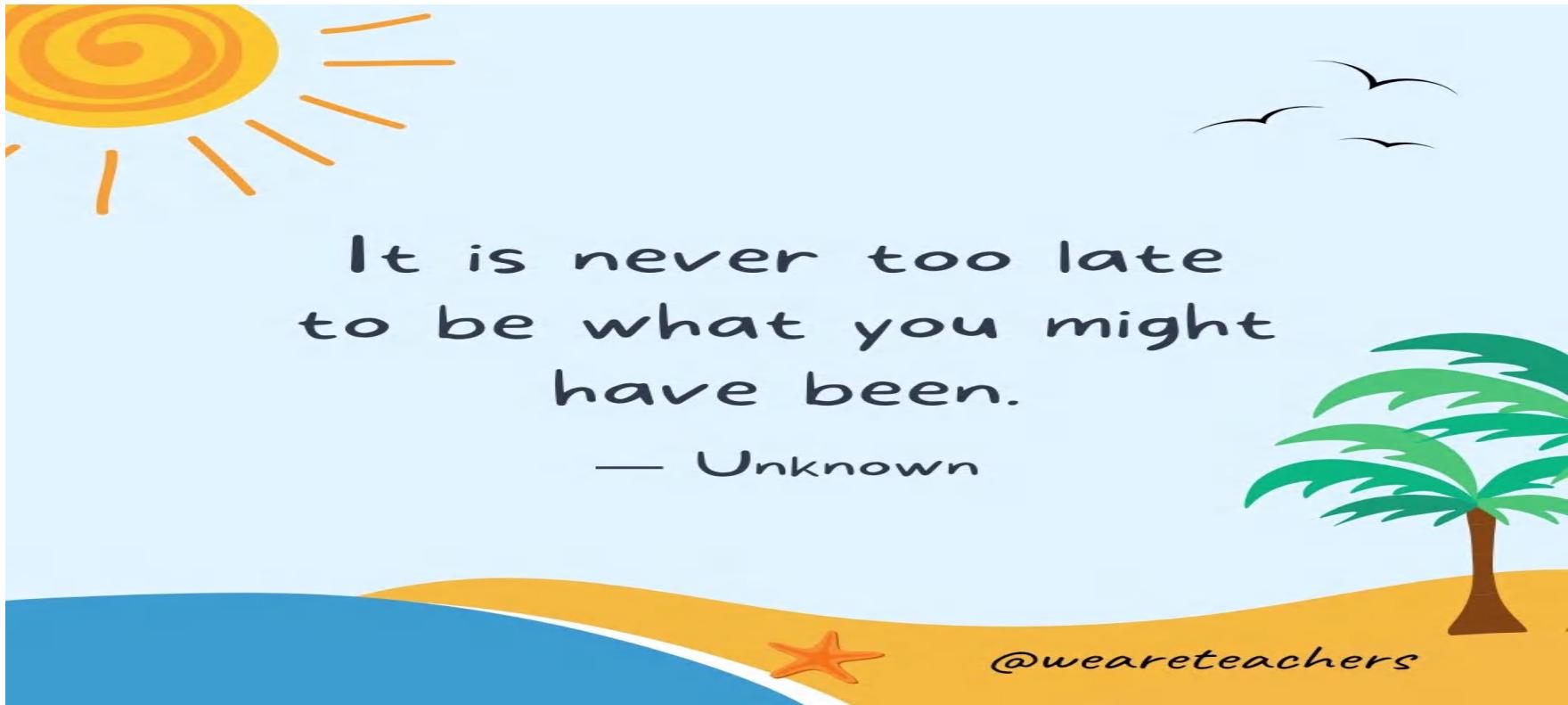
You have an e-commerce website and need to send order confirmation emails to thousands of customers every day. You've decided to use AWS SES for this purpose.

1. What steps would you take to configure SES to handle these bulk emails effectively?
2. How would you ensure that your emails are successfully delivered and avoid being marked as spam?
3. How would you monitor and handle bounce rates or complaints using AWS SES?

Steps to handle these bulk emails effectively

1. Setup Sender Identity
2. Request Production Access
3. Use IAM for Access Control
4. Manage Quotas and Limits
5. Configure Sending
6. Leverage SNS for Feedback
7. Monitor Performance

Thank You



It is never too late
to be what you might
have been.

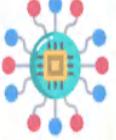
— Unknown

@weareteachers

Data Centers

Design, Architecture, and Tier
Levels



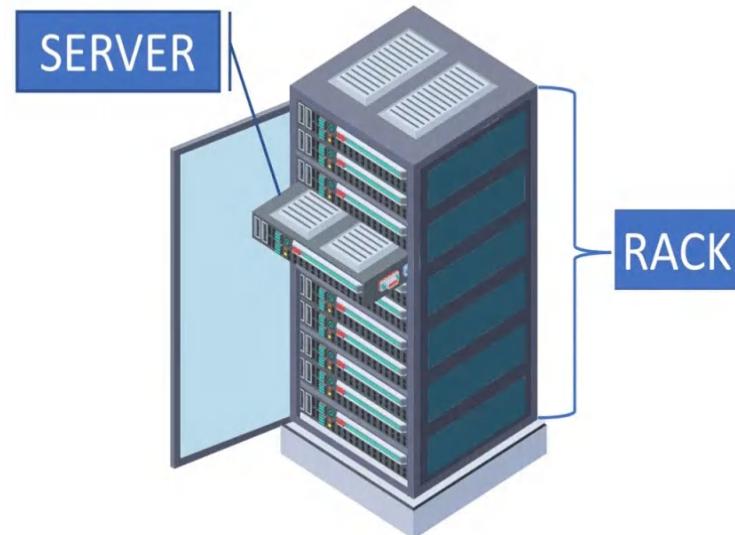
		
 Elastic Compute Cloud (EC2)	 Virtual Machines	 Compute Engine
 Amazon S3	 Azure Blob Storage	 Cloud Storage
 Amazon VPC	 Azure Virtual Network	 Cloud Virtual Network

Data Centers

Data Center: A data center is a facility used to house computer systems and associated components, such as telecommunications and storage systems.

Purpose:

- Centralized management of IT resources
- Support for critical applications
- High availability and redundancy

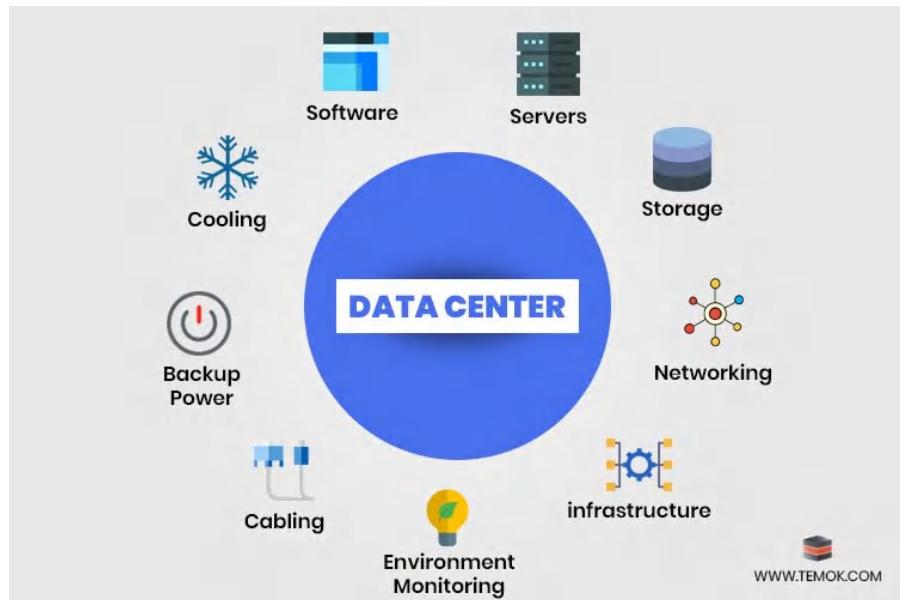


Key Components:

Key Components:

- Servers
- Storage Systems
- Networking Equipment
- Cooling Systems
- Power Supply Units
- Security Systems

Interdependence : Efficient operation relies on seamless integration of these components.



Data Center Design Principles

Scalability : Design for future growth.

Redundancy : Minimize single points of failure.

Efficiency : Optimize power and cooling.

Security : Physical and virtual protection.

Sustainability : Eco-friendly solutions.

Data Center Architecture

Layers :

- Physical Layer: Hardware and infrastructure
- Virtual Layer: Virtualized servers and networks
- Application Layer: Software and services

Topology:

- Star
- Mesh
- Leaf-Spine

Power Efficiency in Data Centers

Key Metrics:

- PUE (Power Usage Effectiveness): Ratio of total facility energy to IT equipment energy.
- DCiE (Data Center Infrastructure Efficiency): Inverse of PUE.

Techniques to Improve Efficiency:

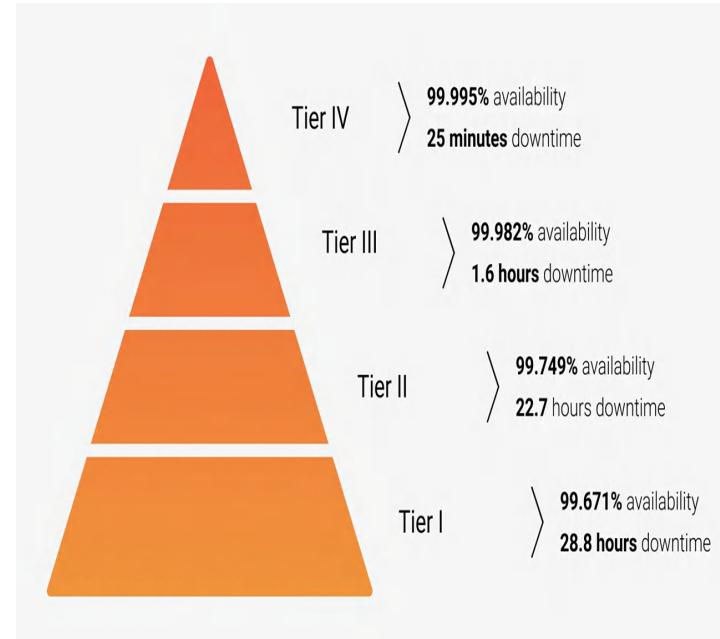
- Energy-efficient hardware
- Liquid cooling systems
- Renewable energy sources

Tier Levels of Data Centers

Classification system based on uptime and redundancy.

Tiers:

- **Tier I:** Basic Capacity (No redundancy)
- **Tier II:** Redundant Capacity
- **Tier III:** Concurrently Maintainable
- **Tier IV:** Fault-Tolerant (Fully redundant systems)



Data Center Security

Aspects:

- **Physical Security:** CCTV, access control
- **Cybersecurity:** Firewalls, encryption
- **Disaster Recovery:** Backup and replication

Importance : Protect data integrity and availability.

Sustainability in Data Centers

Challenges : High energy consumption.

Solutions:

- Use of renewable energy.
- Efficient cooling systems.
- Green certifications (LEED, Energy Star).

Case Study: Efficient Data Center Design

Google's Data Centers

Techniques : AI-based cooling, renewable energy use.

Outcome : Significant energy savings.

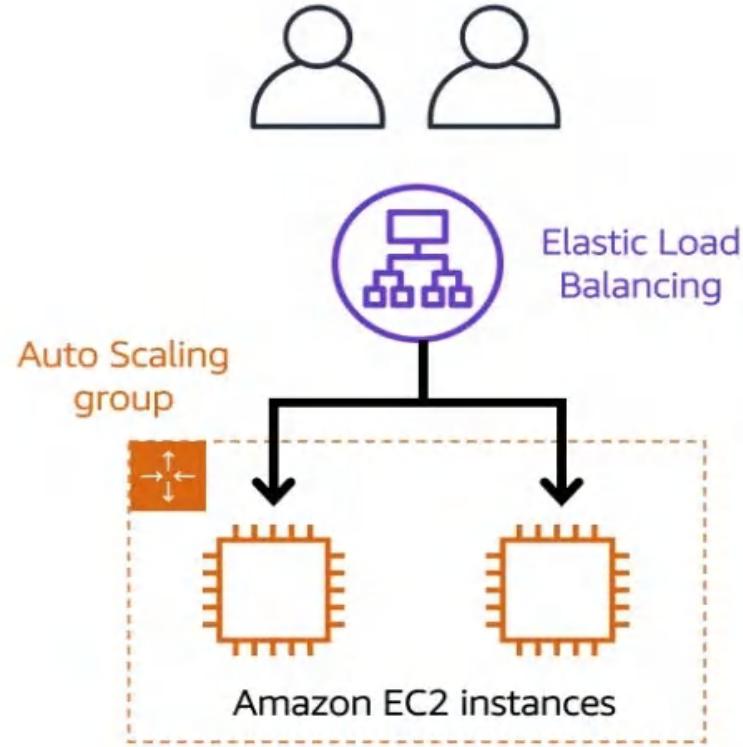


Summary

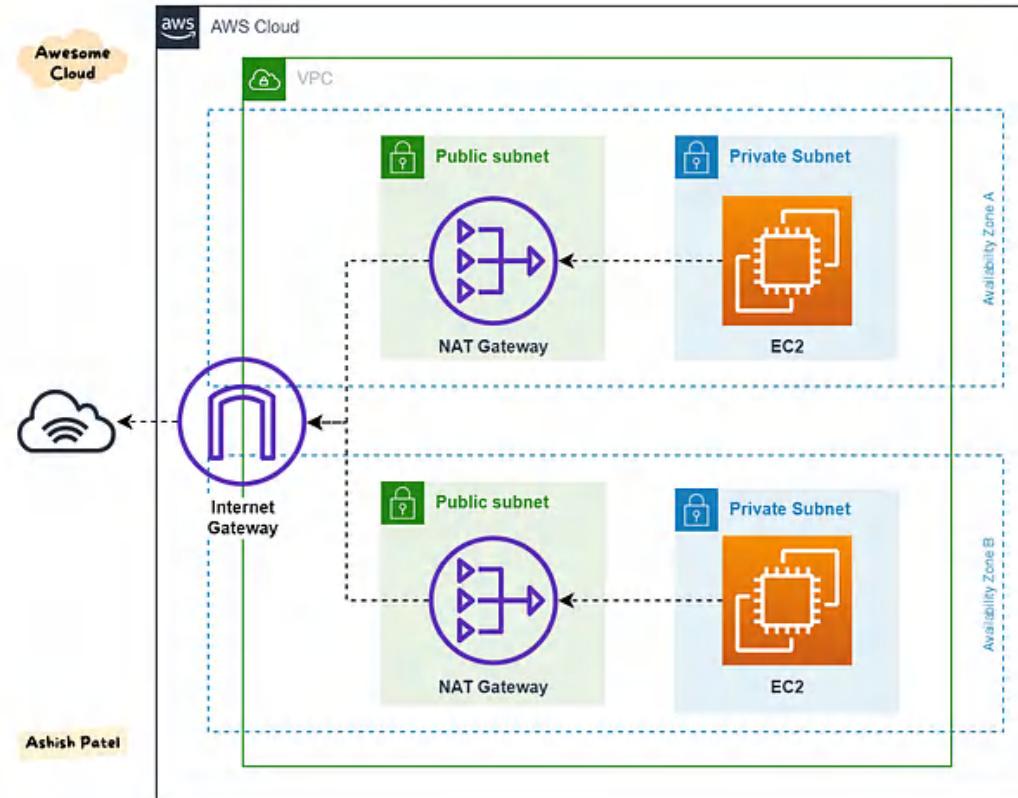
- Data centers are critical for modern IT infrastructure.
- Design and efficiency directly impact performance and sustainability.
- Understanding tier levels helps in planning for required uptime and redundancy.

Previous Class Test Diagram

Auto Scaling



VPC



Virtualization: Future of Computer

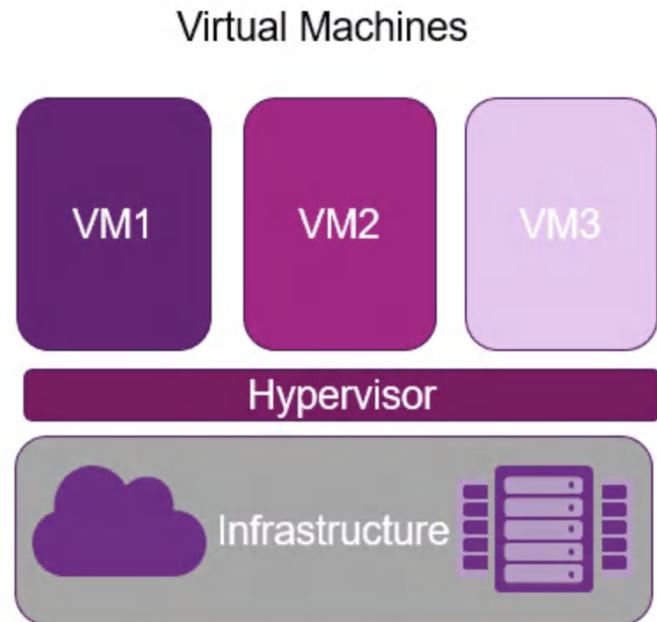
A deep dive into the concepts, types,
and applications of virtualization.

Introduction to Virtualization

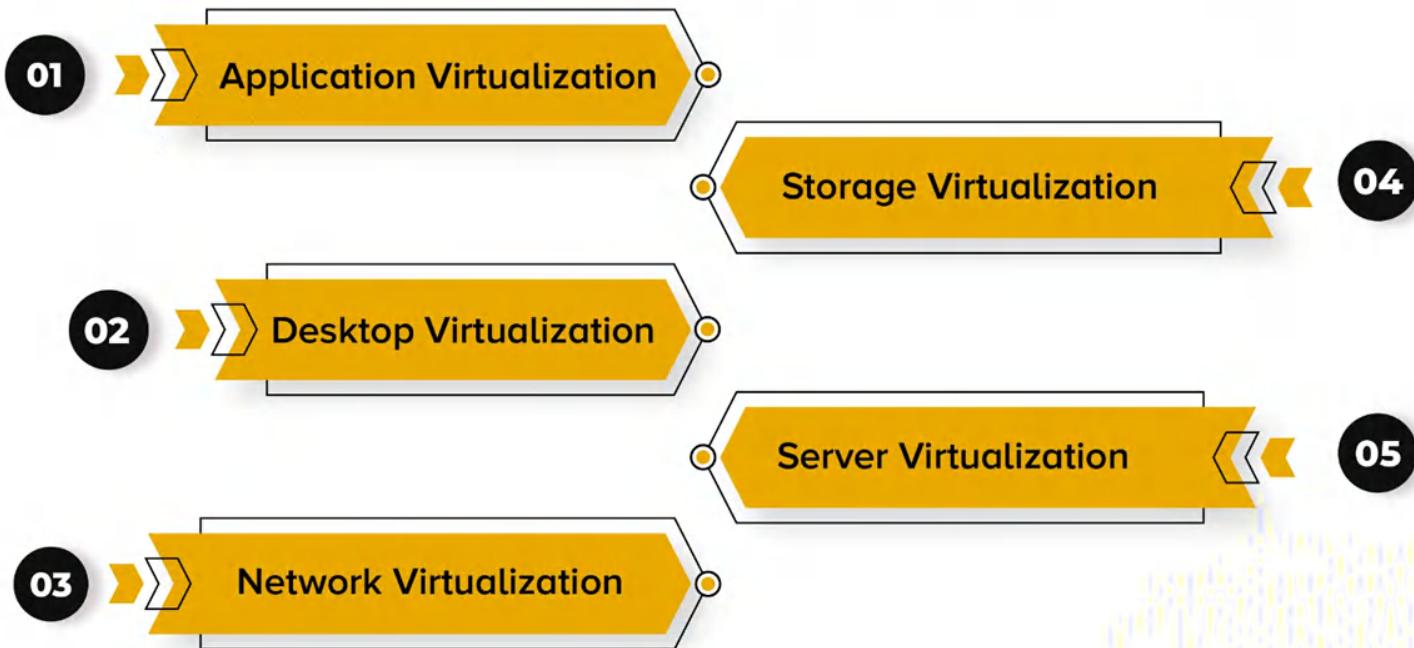
Virtualization is the process of creating a virtual version of computing resources such as servers, storage devices, networks, or even operating systems.

Key Benefits:

- Improved resource utilization.
- Reduced IT costs.
- Enhanced flexibility and scalability.



Types of Virtualization solutions in Cloud computing

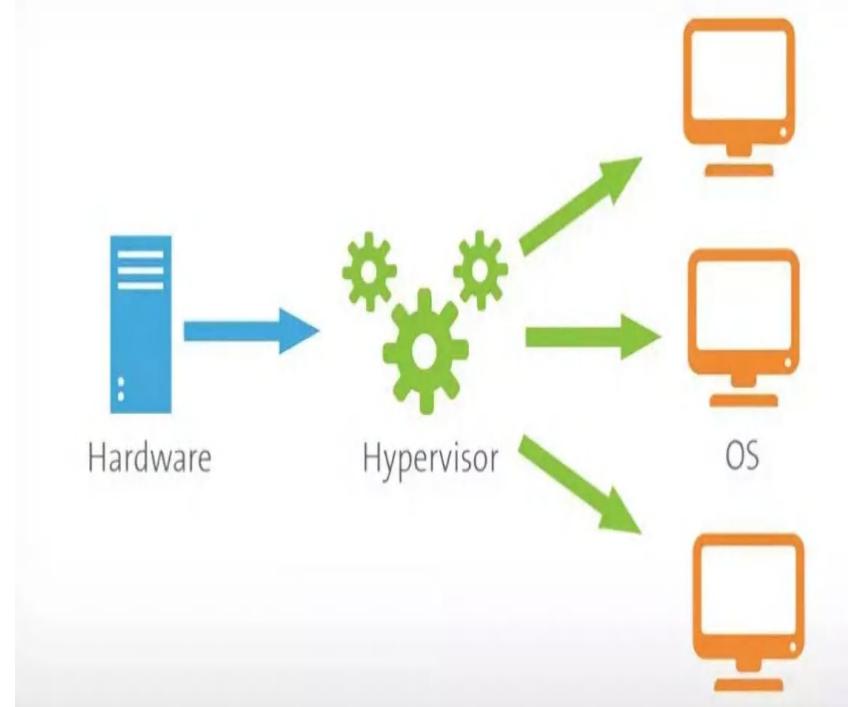


Types of Virtualization

- 1. Server Virtualization:** Partitioning a physical server into multiple virtual servers.
- 2. Desktop Virtualization:** Allows users to access desktops remotely.
- 3. Network Virtualization:** Combines hardware and software network resources into a single, software-based entity.
- 4. Storage Virtualization:** Pools physical storage resources into a single virtual resource.

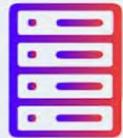
What is Hypervisor?

A **hypervisor**, also known as a **virtual machine monitor** (VMM) or virtualizer, is a type of computer software, firmware or hardware that creates and runs virtual machines.



Hypervisor types

Type 1 Hypervisor



Hardware

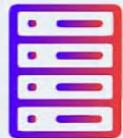


Hypervisor

Guest 1

Guest 2

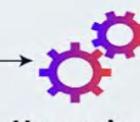
Type 2 Hypervisor



Hardware



Host OS



Hypervisor

Guest 1

Guest 2

How Virtualization Works

Core Components:

1. Hypervisor : Software that creates and manages virtual machines (VMs).
2. Guest OS: The operating system running inside the VM.
3. Host OS: The operating system managing the physical hardware.

Types of Hypervisors:

- Type 1: Runs directly on hardware (e.g., VMware ESXi).
- Type 2: Runs on a host operating system (e.g., VirtualBox).

Advantages of Virtualization

Cost Savings:

- Lower hardware requirements.
- Reduced energy consumption.

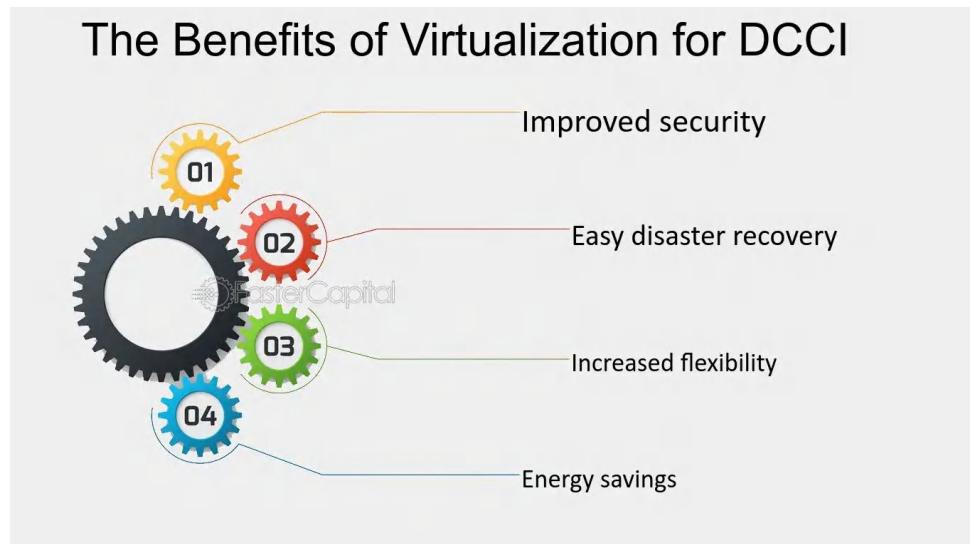
Flexibility:

- Easy migration of workloads.
- Scalability.

Improved Disaster Recovery:

- Backups and snapshots.

Data Center Consolidation initiative



Challenges of Virtualization

- **Initial Costs:** Higher upfront costs for setup.
- **Complexity:** Requires expertise for configuration and maintenance.
- **Performance Overhead:** Potential decrease in performance due to resource sharing.
- **Security Concerns:** Risks like VM escape and hypervisor attacks .

Applications of Virtualization

Cloud Computing: Foundation for services like IaaS, PaaS, and SaaS.

Development and Testing: Isolated environments for software development.

Disaster Recovery: Quick restoration of systems.

Education and Training: Virtual labs and simulations.

Future of Virtualization

Emerging Trends:

- Containerization (e.g., Docker, Kubernetes).
- Edge Computing.
- Virtual Reality (VR) and Augmented Reality (AR).

Impact : Greater focus on scalability, speed, and efficiency.

Summary

- Virtualization transforms IT infrastructure by optimizing resource utilization, reducing costs, and improving scalability.
- Key types include server, storage, network, and desktop virtualization.
- Challenges like security risks and complexity need to be managed effectively.

Containerization: Application Deployment

A comprehensive look at containerization technologies, benefits, and applications.

Developer:

It works on my
computer

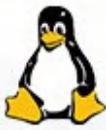
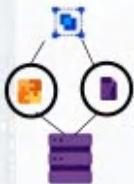


Product Manager:

Yes, but we are NOT going
to give your computer to
the customer



The Benefits of Using Containers

Any OS	Any where	Any app	Any languages
 Linux	 On-Premises	 Monolith	 Java
 Windows	 Cloud	 Microservices	 .Net  Python  Node

Introduction to Containerization

Containerization is a lightweight virtualization method where applications and their dependencies are packaged together into containers, ensuring consistency across environments.

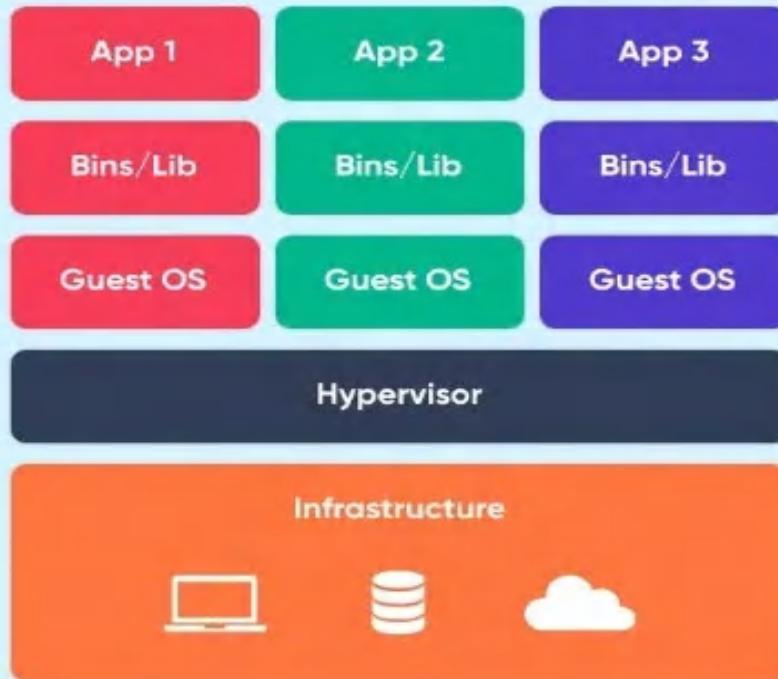
Key Features:

- Lightweight compared to virtual machines.
- Portable and consistent.
- Isolated environments for applications.

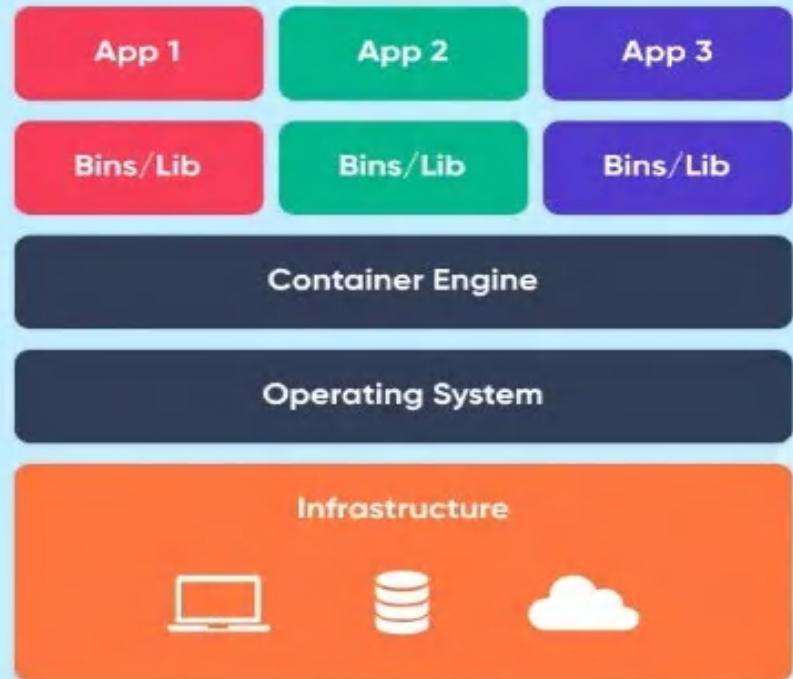




Virtual Machines



Containers



Virtualization and Containerization

Virtualization	Containerization
Hardware level process isolation	OS level process isolation
Each VM has separate OS	Each container can share OS resources
Boots in minutes	Boots in seconds
More resource usage	Less resource usage
Typically bigger in size as they contain whole OS underneath	Smaller in size with only container runtime over the host OS
VMs can easily be moved to a new host	Containers are destroyed and recreated rather than moving

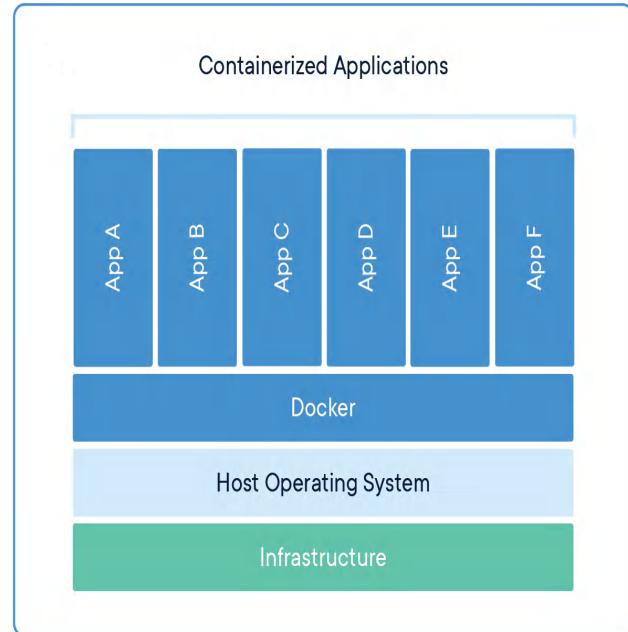
How Containers Work

Core Components:

- **Container Engine:** Manages and runs containers (e.g., Docker Engine).
- **Container Image:** A lightweight, standalone executable package containing everything needed to run an application.
- **Orchestration Tools:** Manage large -scale container deployments (e.g., Kubernetes).

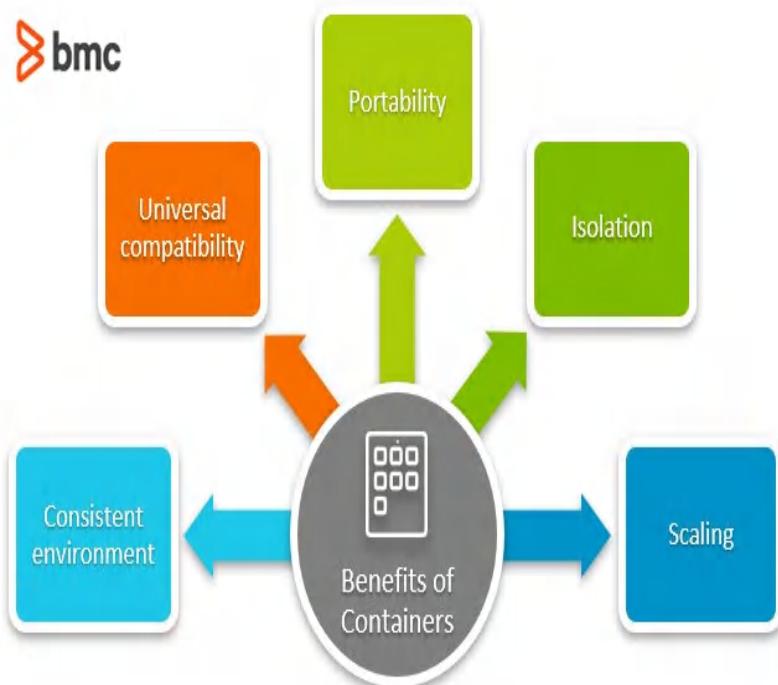
Comparison with Virtual Machines:

- Containers **share the host OS kernel**, while VMs include a full OS.
- Containers are faster and more efficient.



Benefits of Containerization

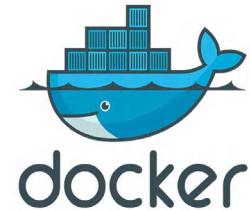
- 1. Portability:** Consistent behavior across development, testing, and production environments.
- 2. Efficiency:** Uses fewer resources than traditional VMs.
- 3. Scalability:** Easy to scale applications horizontally.
- 4. Isolation:** Secure and independent runtime environments.
- 5. Rapid Deployment:** Faster builds, testing, and deployments.



Popular Containerization Tools

Docker:

- Most widely used containerization platform.
- Easy image creation and management.



Kubernetes:

- Orchestrates containerized applications.
- Manages scaling, networking, and distribution.



Podman:

- Docker-compatible but daemonless.(No BGP)
- Focuses on security.



RealWorld Use Cases

1. **Microservices Architecture:** Breaking monolithic apps into smaller, manageable services.
2. **CI/CD Pipelines:** Containerized builds for consistency.
3. **Hybrid Cloud Deployments:** Seamlessly migrate workloads between cloud providers.

Challenges of Containerization

1. **Security Concerns:** Vulnerabilities in shared host OS.
2. **Learning Curve:** Requires understanding of tools like Docker/Kubernetes.
3. **Resource Management:** Monitoring and limiting resource use.
4. **Compatibility Issues:** Legacy systems may not support containers.

Future of Containerization

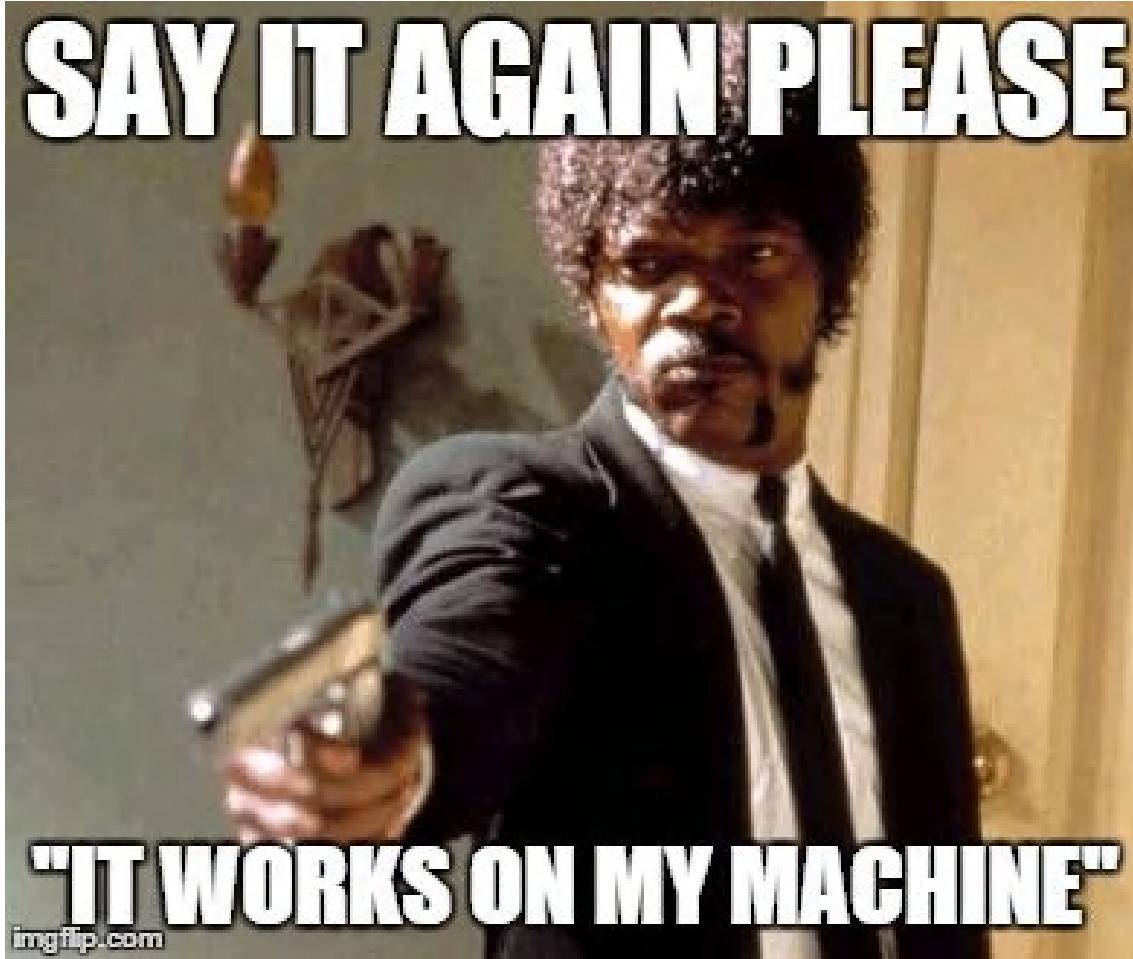
Emerging Tools: Alternatives like Podman, CRI -O.

Trends:

1. Serverless computing leveraging containers.
2. Edge computing with containerized apps.

Predictions : Greater adoption in IoT and AI workloads.

SAY IT AGAIN PLEASE



"IT WORKS ON MY MACHINE"

Summary

Recap of key points:

- Lightweight and portable applications.
- Key tools: Docker and Kubernetes.
- Real-world applications in microservices and CI/CD.

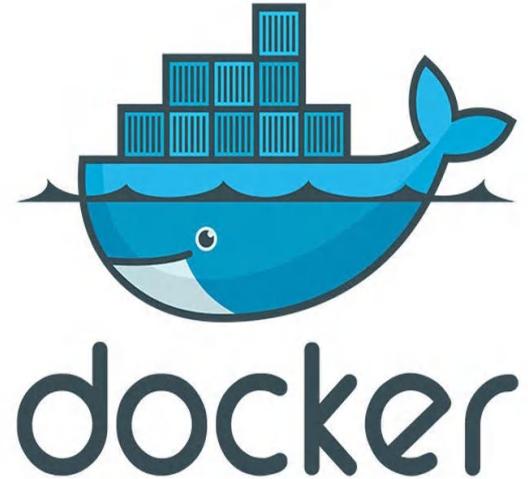
Docker: Containerization Simplified

— Understanding Docker's Role in Modern DevOps —

What is Docker?

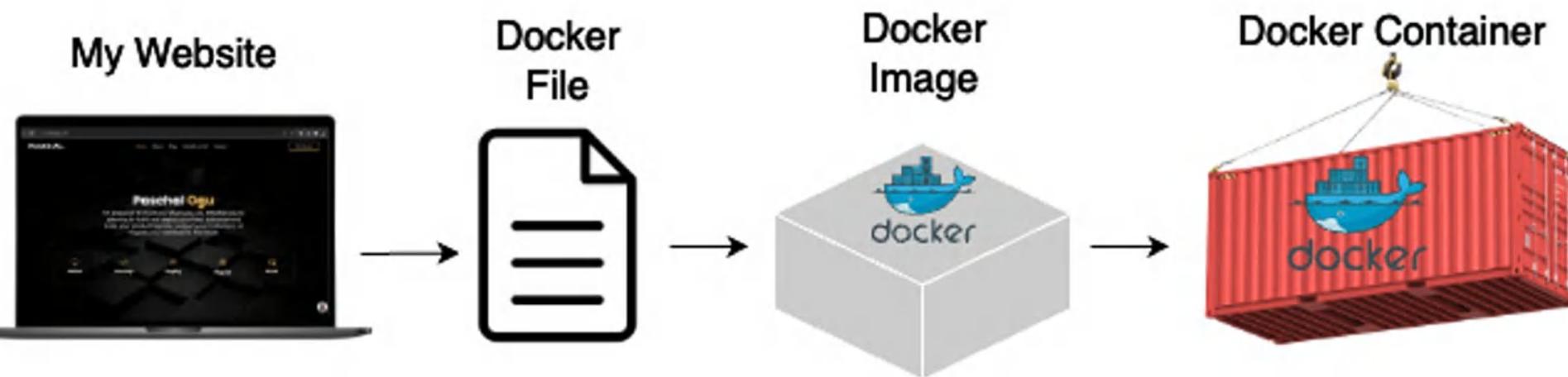
Docker is a platform for developing, shipping, and running applications inside lightweight, portable containers.

- Containers **package an application and its dependencies into a single unit** that can run anywhere.
- Unlike VMs, Docker containers **share the host OS kernel**, making them more lightweight and efficient.



Key Components of Docker

1. **Docker Engine:** The core component that runs Docker containers.
2. **Images:** Read-only templates used to create containers.
3. **Containers:** The running instances of Docker images that can interact with other services.
4. **Docker Hub:** A registry of Docker images, where you can download and upload images.
5. **Docker Compose:** A tool for defining and running multi -container Docker applications.



Docker Architecture

1.Client -Server Architecture:

- **Docker Client:** Users interact with the Docker client to issue commands (e.g., docker run).
- **Docker Daemon:** The Docker Daemon runs in the background, managing containers and images.

2.Docker Registries: Docker images are stored in registries such as Docker Hub or private registries.

3.Communication: The Docker client communicates with the Docker daemon via REST API or UNIX sockets.



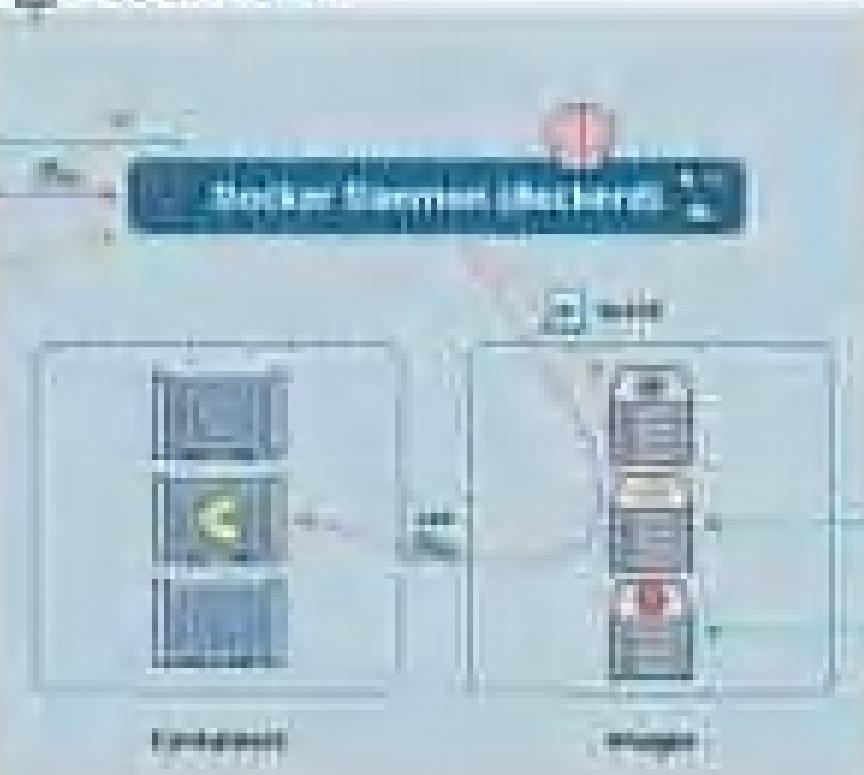
Docker Architecture



Docker Client



Docker Host

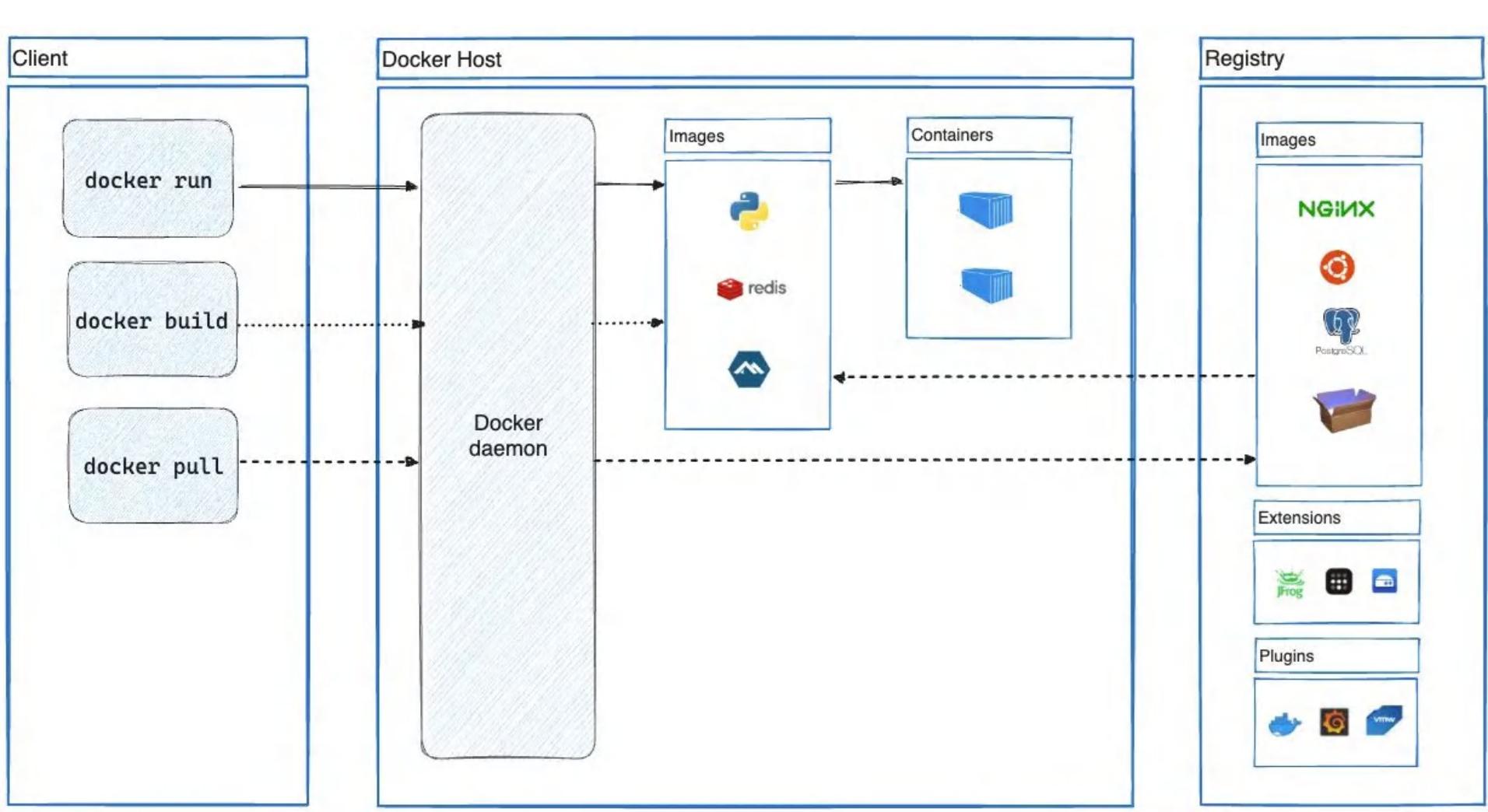


Registry (Hub)

Provides storage
service for images
(e.g., Docker Hub)

Provides storage
service for containers





How Docker Works

1. **Build Process:** Docker uses Dockerfiles to automate the process of building images.
2. **Run Process:** Containers are run from images, which include the application and its dependencies.
3. **Isolation:** Containers isolate processes from each other and from the host system, ensuring consistency across environments.
4. **Networking:** Docker provides networking features to link containers together and manage their communication.

Docker Commands

- `docker build`: Build an image from a Dockerfile
- `docker run`: Run a container from an image
- `docker ps`: List running containers
- `docker stop`: Stop a running container
- `docker pull`: Download an image from Docker Hub
- `docker exec`: Execute a command inside a running container

Dockerfile Basics

Dockerfile: A text file containing instructions on how to build a Docker image.

Basic Syntax:

- **FROM:** Specifies the base image
- **RUN:** Executes commands inside the container
- **COPY:** Copies files into the container
- **CMD:** Specifies the command to run **when the container starts**

```
FROM node:14
COPY . /app
WORKDIR /app
RUN npm install
CMD ["npm", "start"]
```

Docker Images and Layers

Image Layers: Docker images are built in layers. Each layer represents a change made to the image.

Layer Caching: Docker caches layers to speed up image builds by reusing layers that haven't changed.

Image Size: Smaller images lead to faster deployments and less overhead.

Optimizing Dockerfiles: Combining commands and minimizing layers helps reduce image size.

Docker Networking

Network Types:

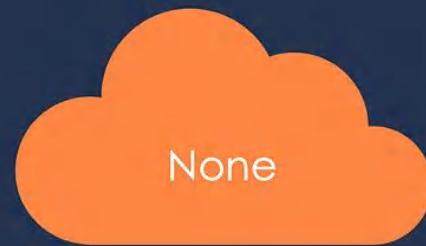
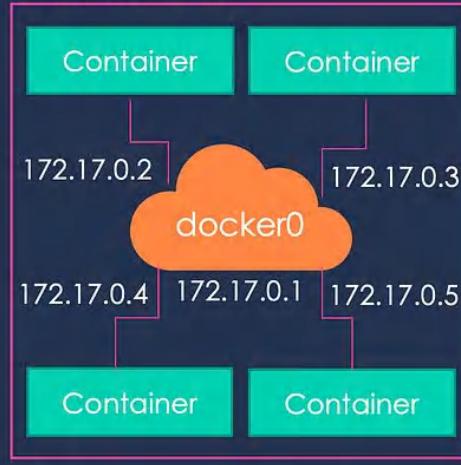
- **Bridge Network:** Default network for containers on the same host.
- **Host Network:** The container shares the host's network stack.
- **None Network:** No networking for the container.
- **Overlay Network:** For multi -host networking in Docker Swarm or Kubernetes.

Port Mapping: Expose container ports to communicate with external applications.



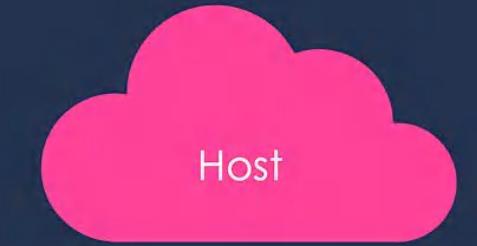
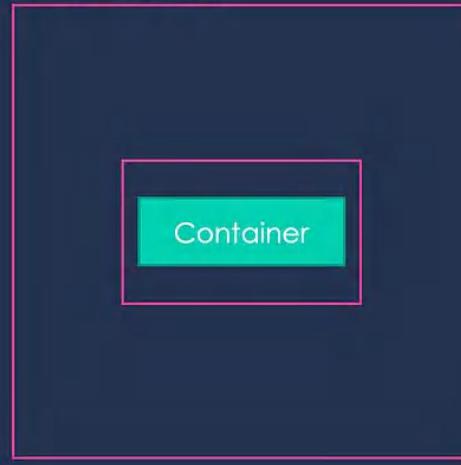
```
docker run ubuntu
```

Docker Host



```
docker run \  
--network=none  
ubuntu
```

Docker Host



```
docker run \  
--network=host  
ubuntu
```

Docker Host



Docker Compose

What is Docker Compose?: A tool for defining and running multi -container Docker applications.

docker -compose.yml: The configuration file where you define services, networks, and volumes.

Common Commands:

- **docker-compose up:** Starts all services defined in the Compose file
- **docker-compose down:** Stops all services and removes containers

```
version: '3'  
services:  
  web:  
    image: nginx  
    ports:  
      - "8080:80"  
  db:  
    image: postgres
```

Docker in DevOps and CI/CD

DevOps and Docker: Docker helps streamline the development pipeline by providing consistent environments from development to production.

Continuous Integration (CI): Build, test, and deploy Docker images as part of the CI pipeline.

Continuous Deployment (CD): Automate the deployment of Docker containers to production environments.

Popular CI/CD Tools: Jenkins, GitLab CI, CircleCI, and GitHub Actions.

Best Practices for Docker

Keep Images Small: Use minimal base images to reduce overhead.

Use Multi -stage Builds: Separate the build and runtime environments to optimize image size.

Limit Container Privileges: Use Docker's security features to limit access and reduce vulnerabilities.

Monitor and Log: Use tools like Docker stats and logging drivers to monitor containers.

Introduction to Kubernetes

— Understanding the Basics of Container Orchestration —

What is Kubernetes?

- Kubernetes (K8s) is an open-source platform designed to automate deploying, scaling, and operating containerized applications.
- Originally developed by Google, now maintained by the Cloud Native Computing Foundation (CNCF).
- Helps manage clusters of containers.



kubernetes

Good reasons for using Kubernetes

Hassle-free
load balancing



Enables data
storage using disc



Automates deploying
containerized applications

Have
availability

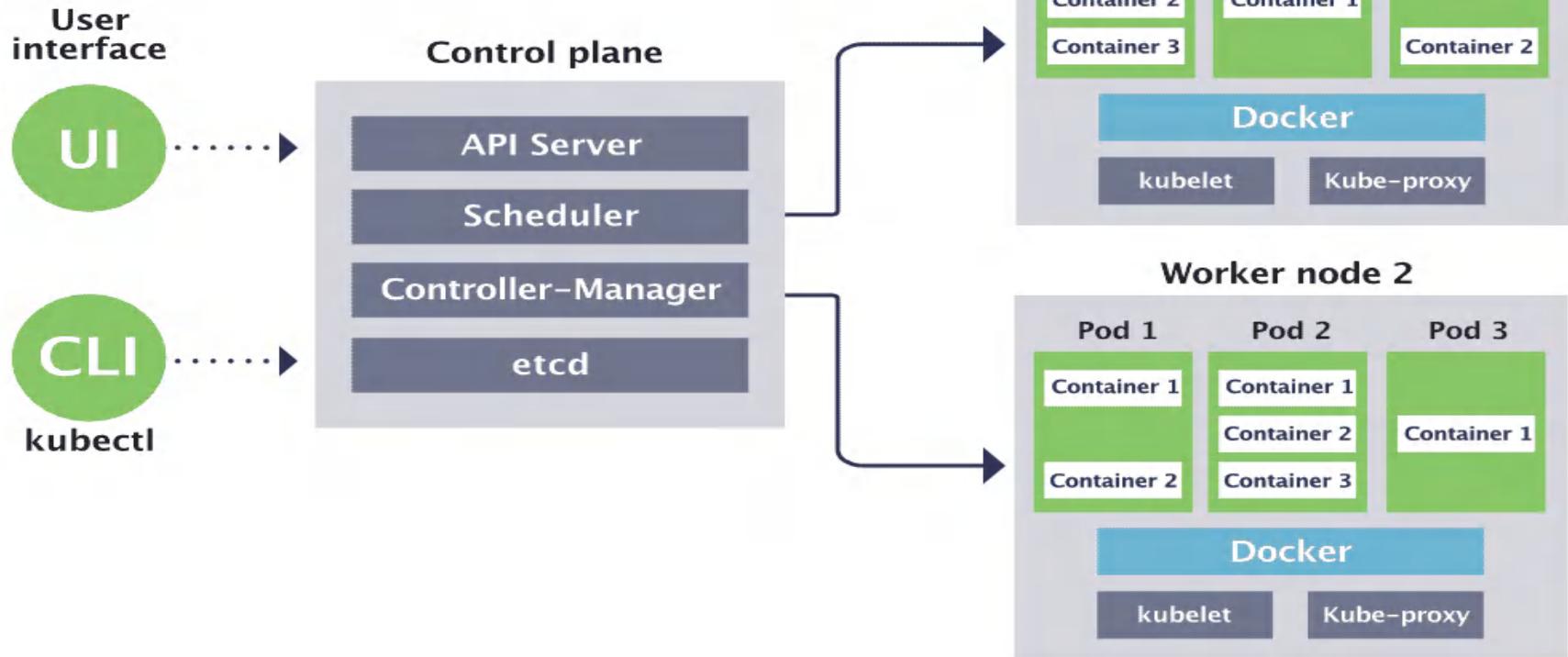


Easy to monitor
and scale application
services

Kubernetes Architecture

1. Master node and worker nodes architecture.
2. **Control Plane:** API Server, Scheduler, Controller Manager.
3. **Worker Plane:** Kubelet, Kube Proxy.
4. Etcd as a distributed key-value store.

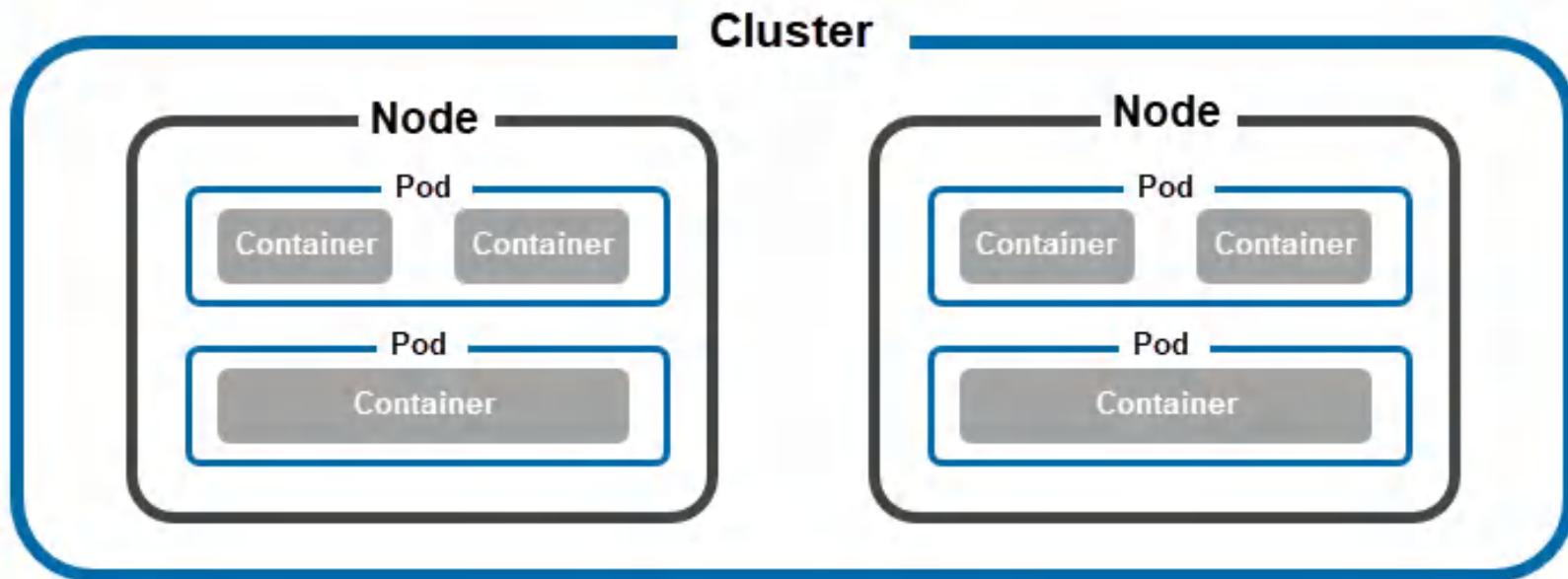
Kubernetes architecture



Architecture Details

- **Control Plane:** Manages the cluster's overall operations through components like API Server, Scheduler, and Controller Manager.
 - **API Server:** Central communication hub for the Kubernetes cluster, handling REST API requests.
 - **Scheduler:** Assigns Pods to nodes based on resource availability and constraints.
 - **Controller Manager:** Ensures the cluster's desired state by running controllers like Node and Replication Controllers.
 - **etcd :** A distributed, highly available key -value store used by Kubernetes to store all cluster data, including configuration, state, and metadata. It ensures consistency across the cluster.
- **Worker Plane:** Executes workloads on nodes using Kubelet (container management) and Kube Proxy (networking).
 - **Kubelet:** Manages containers on a worker node, ensuring they are running as instructed.
 - **Kube Proxy:** Handles networking, enabling communication between Pods and Services.

Cluster



How Kubernetes Works

- Uses declarative configuration and desired state.
- Schedules workloads on worker nodes.
- Monitors application health.
- Self-heals by restarting failed containers.

Key Kubernetes Objects

Pod: Group of containers.

Service : Exposes application as a network service.

ConfigMap : External configuration for Pods.

Ingress : HTTP and HTTPS routing.

Deployment : Manages replicas of Pods.

Benefits of Kubernetes

- High availability.
- Scalability and performance.
- Portability across environments.
- Cost optimization with resource management.

Popular Use Cases

- Microservices architecture.
- Batch processing and CI/CD pipelines.
- Edge computing and IoT.
- Hybrid and multi -cloud setups.

Getting Started with Kubernetes

1. Install Minikube or use a managed Kubernetes service (e.g., GKE, EKS, AKS).
2. Write configuration files in YAML.
3. Use `kubectl` for cluster management.

Challenges of Kubernetes

1. Complexity in setup and management.
2. Steep learning curve.
3. Security and monitoring concerns.
4. Resource overhead.

Conclusion

- Kubernetes is a powerful tool for managing containerized applications.
- Its robust features and scalability make it ideal for modern application deployment.
- Embrace Kubernetes to simplify and optimize your infrastructure.

What is your goal after graduation?

And after that I will try to complete phd from abroad

I want to be a banker

knowledge

higher study in abroad

sector

Join the Industry.

Don't

abilities

Usa

company

higher study

gain
join

Business

job

Want to do a job

Understanding API Gateways

— Managing Communication in Modern Architectures —

What is an API Gateway?

An **API Gateway** is a server that acts as an intermediary between clients and backend services, managing API calls, authentication, and traffic.

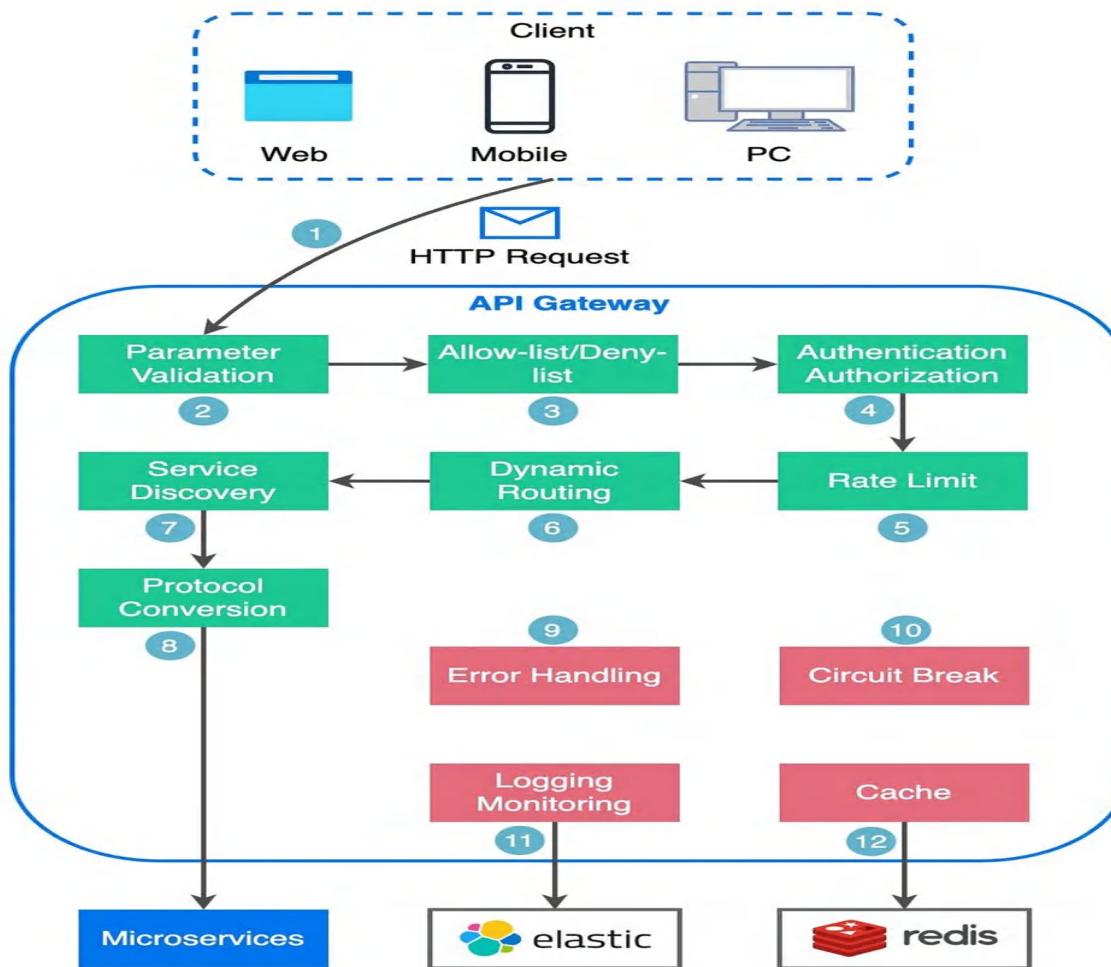
Core Functions:

- Request routing
- Authentication and authorization
- Rate limiting
- Caching
- Monitoring and analytics

Real-World Examples: AWS API Gateway, Kong, Apigee, NGINX

What does API Gateway do?

 blog.bytebytogo.com



Why Use an API Gateway?

Simplified Client Interaction: Combines multiple APIs into a single interface.

Enhanced Security: Centralized authentication and authorization mechanisms.

Scalability: Distributes traffic across backend services.

Monitoring: Tracks usage patterns and performance metrics.

Performance Optimization: Provides caching to reduce latency.

Architecture of API Gateway

Client: Sends requests to the API Gateway.

API Gateway: Processes the request, applies security checks, and routes it to the appropriate backend service.

Backend Services: Handle the actual functionality, such as database queries or business logic.

Response Flow: Backend -> API Gateway -> Client

Key Features of an API Gateway

1. Authentication and Authorization Ensures secure access to APIs.
2. Rate Limiting Prevents abuse by limiting the number of requests.
3. Load Balancing Distributes incoming requests across multiple back services.
4. Caching Reduces response times for repeated requests.
5. Traffic Shaping Prioritizes specific API requests based on policies.
6. Monitoring and Analytics Tracks performance and identifies bottlenecks.

API Gateway vs Reverse Proxy

- **API Gateway:**
 - Manages multiple APIs.
 - Provides additional features like authentication, monitoring, and caching.
 - Specifically designed for API ecosystems.
- **Reverse Proxy:**
 - Handles requests and forwards them to servers.
 - Primarily used for load balancing and SSL termination.
- **Key Difference:** API Gateways are specialized for API management, while reverse proxies are general-purpose tools.

Benefits of Using API Gateways

- **Centralized API Management:** Easier to manage and monitor multiple APIs.
- **Improved Security:** Single point for implementing security protocols.
- **High Availability:** Reduces downtime with load balancing.
- **Enhanced User Experience:** Faster response times with caching.
- **Seamless Integration:** Supports diverse backend services.

Common Use Cases for API Gateways

1. **Microservices:** Manages communication between services.
2. **Serverless Applications:** Routes traffic to serverless functions.
3. **Mobile Applications:** Simplifies backend interactions for mobile apps.
4. **IoT Devices:** Handles high traffic from IoT devices efficiently.
5. **Third-Party Integrations:** Provides a unified interface for external APIs.

Best Practices for API Gateways

- **Secure API Endpoints:** Implement strong authentication mechanisms.
- **Enable Logging and Monitoring:** Track API usage and performance.
- **Optimize Caching:** Reduce latency and improve performance.
- **Set Rate Limits:** Prevent abuse and ensure fairness.
- **Plan for Scalability:** Design the gateway to handle traffic spikes.
- **Documentation:** Provide clear guidelines for developers.

RealWorld API Gateway Providers

1. **AWS API Gateway:** Fully managed service for building and deploying APIs.
2. **Kong:** Open-source gateway with advanced plugins.
3. **Apigee:** Google's API management solution.
4. **NGINX:** Combines reverse proxy with API management features.
5. **Azure API Management:** Integrates seamlessly with Azure services.

Conclusion

- **Key Points:**
 - API Gateways centralize and secure API traffic management.
 - They play a crucial role in modern architectures like microservices.
- **Action Item:** Explore API Gateway solutions for your next project.

CloudFront: Your Content Delivery Sup

Accelerating Content Delivery on AWS (CDN
Strategies, Security, and Scalability)

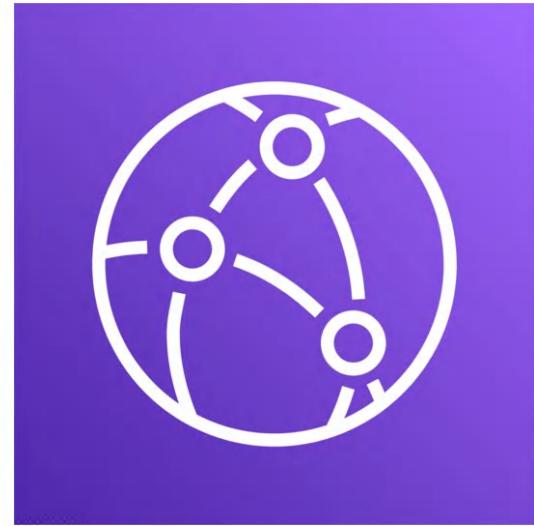
Introduction to CDN & CloudFront

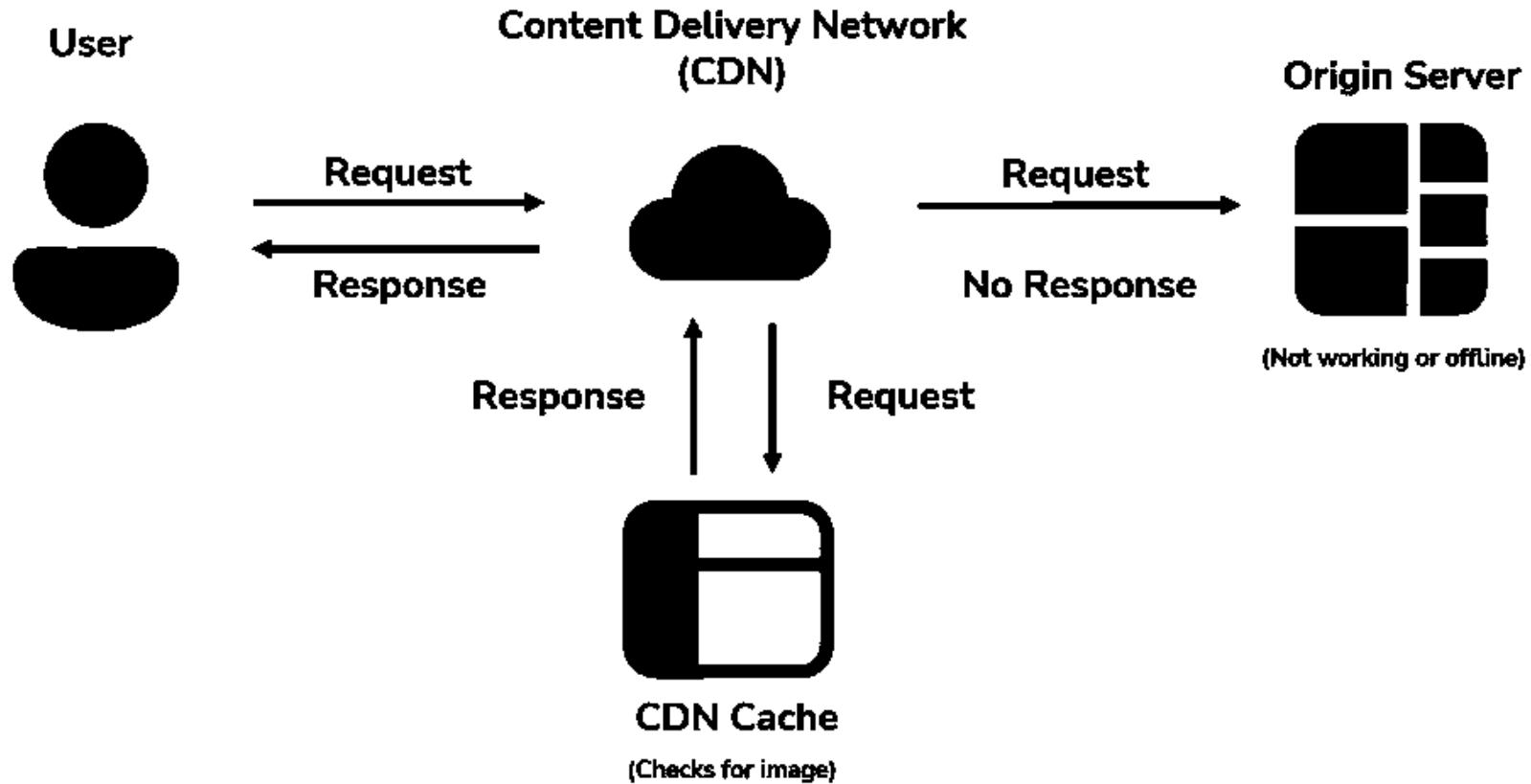
- What is a CDN?

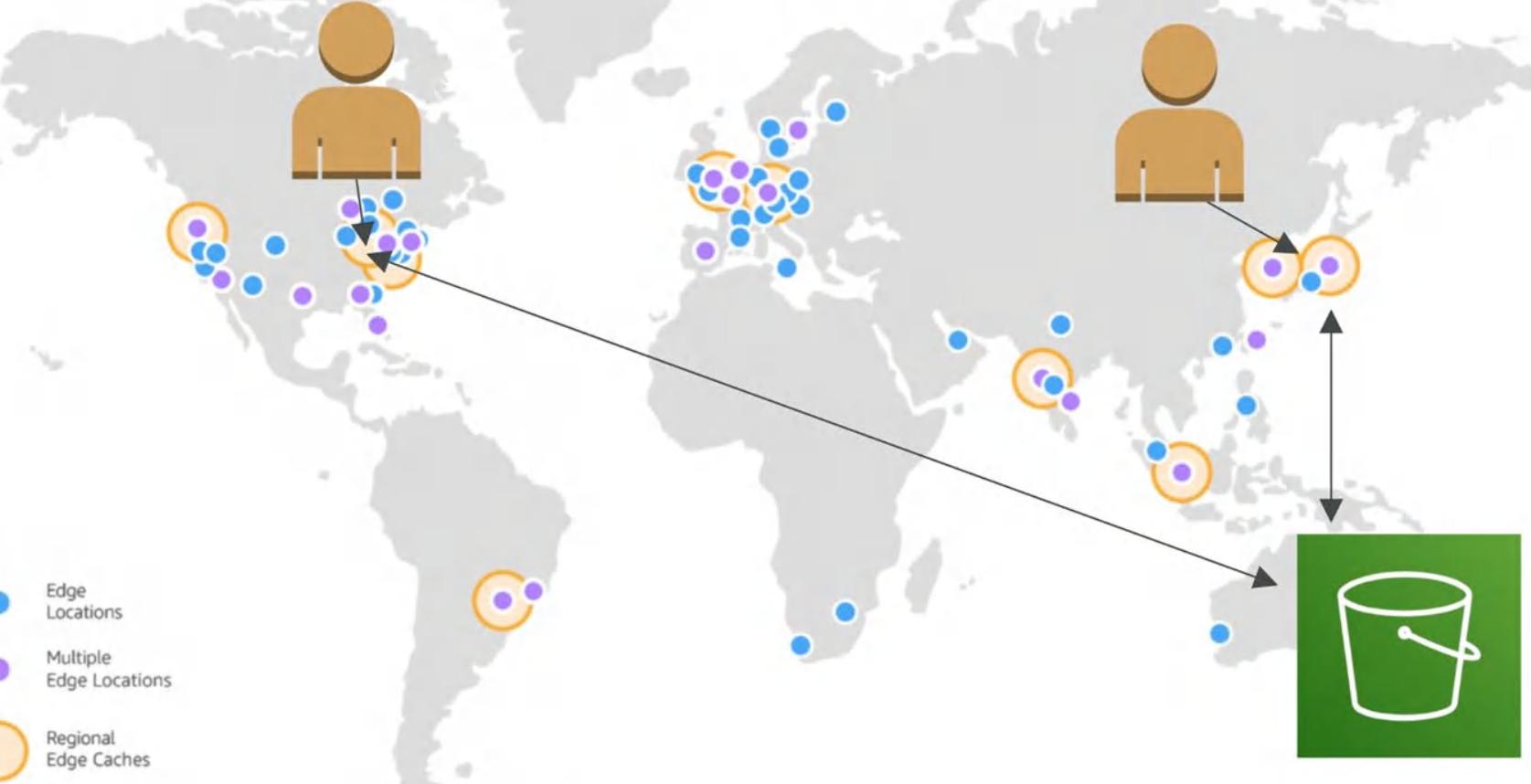
A Content Delivery Network (CDN) is a globally distributed network of servers that delivers web content (images, videos, scripts) to users based on their geographic location.

- What is Amazon CloudFront?

AWS's CDN service that integrates with other AWS services (S3, EC2, Lambda) to accelerate content delivery with low latency and high transfer speeds.







Key Features of CloudFront

- **Global Edge Locations** 1000+ edge locations worldwide.
- **Caching** Stores content closer to users for faster delivery.
- **Security** Integration with AWS Shield (DDoS protection), SSL/TLS encryption.
- **Dynamic Content** Supports both static and dynamic content.
- **Lambda@Edge** Run serverless code at edge locations.

CloudFront Use Cases

1. Accelerate Static Websites: Hosted on S3.
2. Live/On -Demand Video Streaming: Integrates with AWS Media Services.
3. APIs and Dynamic Content: Caching API responses.
4. Security: Block malicious traffic with AWS WAF.

Security with CloudFront

- **SSL/TLS:** Encrypts data in transit.
- **AWS WAF:** Block SQL injection, XSS, etc.
- **Signed URLs/Cookies:** Restrict access to private content.
- **DDoS Protection:** Via AWS Shield Standard/Advanced.

Pricing Model

- **Data Transfer Out:** Cost per GB based on region.
- **HTTP/HTTPS Requests:** Cost per 10,000 requests.
- **Lambda@Edge:** Cost based on compute time and requests.
- **Free Tier:** 1 TB/month data transfer + 10 million HTTP requests.

Best Practices

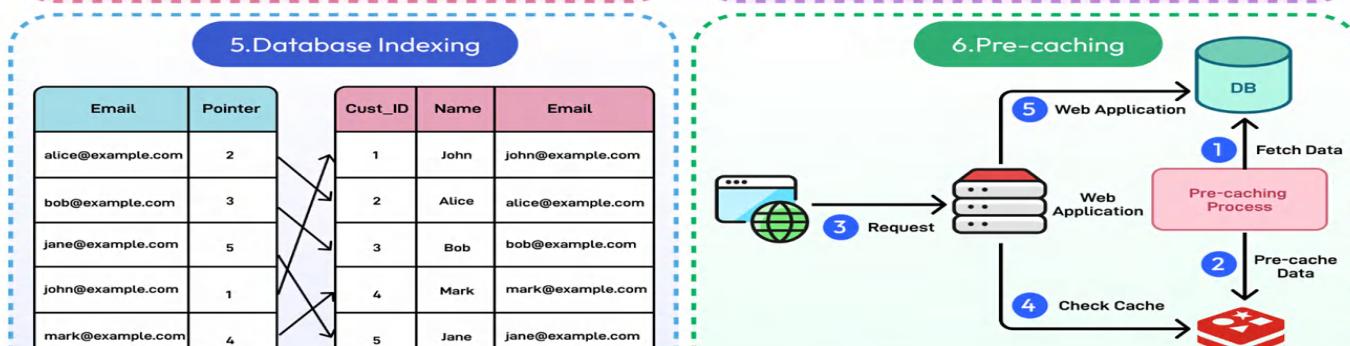
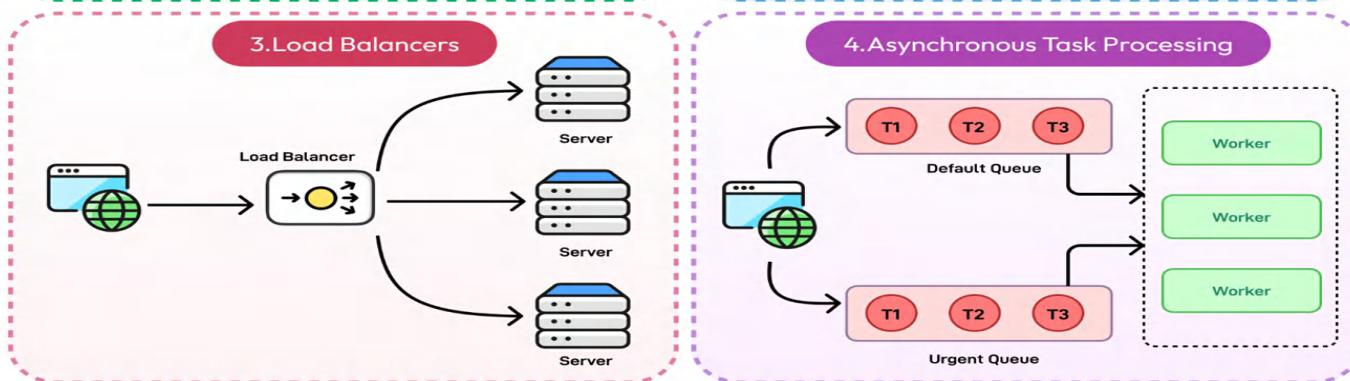
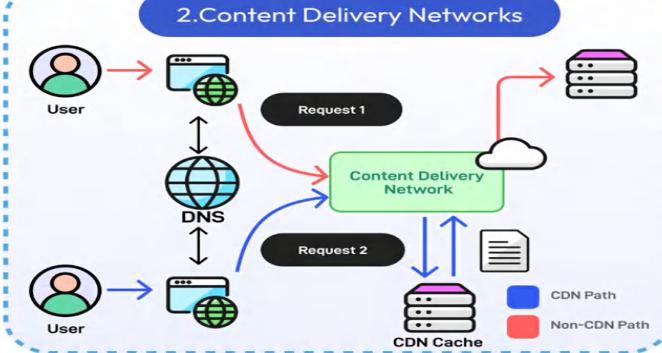
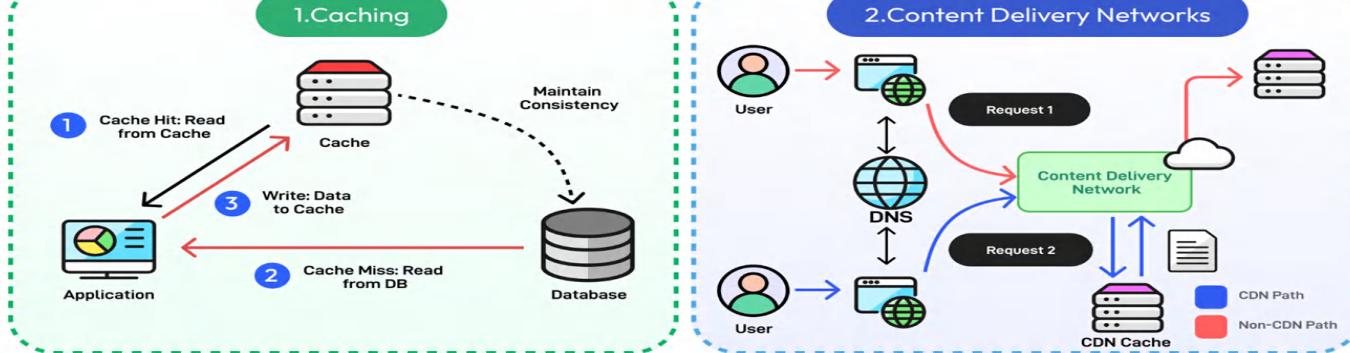
- Use Cache-Control Headers to define **TTL** (Time to Live).
- Enable **Compression** (gzip, Brotli) for faster transfers.
- Monitor with **CloudWatch Metrics**.
- Use **Origin Shield** to reduce origin load.

Setting Up CloudFront

- Steps to create a CloudFront distribution.
- Configuring origin, behaviors, and caching settings.
- Deploying and monitoring CloudFront with AWS tools.

Summary

- Amazon CloudFront is a powerful CDN solution.
- Enhances performance, security, and scalability.
- Seamlessly integrates with AWS services for a robust cloud architecture.



Amazon RDS

— Managed Relational Database Service —

Relational Databases

Students

Student ID	Dept ID	Name	Email
1	M01	Joe Miller	joe@abc.com
2	B01	Sarah T	sarah@abc.com

Subjects

Student ID	Subject
1	Physics
1	Chemistry
1	Math
2	History
2	Geography
2	Economics

Departments

Dept ID	SPOC	Email	Phone
M01	Kelly Jones	kelly@abc.com	+1234567890
B01	Satish Kumar	satish@abc.com	+1234567891

NoSQL

- NoSQL = non-SQL = non relational databases
- NoSQL databases are purpose built for specific data models and have flexible schemas for building modern applications.

```
{  
  "name": "John",  
  "age": 30,  
  "cars": [  
    "Ford",  
    "BMW",  
    "Fiat"  
  ],  
  "address": {  
    "type": "house",  
    "number": 23,  
    "street": "Dream Road"  
  }  
}
```

Students

Practice

- Find the username with the highest marks from the table.
- Find the second highest marks from the table.

SL	Username	Marks	time
1	20200204093_tanjuma	10/10	1:31
2	Hasin Daiyan_20200204055	10/10	1:40
3	Rehenuma Tabassum_104	10/10	1:52
4	Protiva-28	10/10	2:06
5	Sadia Sabrin Neha_20200204100	10/10	3:17
6	Tanvir Raiyan 20200204034	10/10	3:34
7	Ramisha54	9/10	1:26
8	20200204042_SabrinaTabassum	9/10	1:52
9	Ajrin khanam orpa_053	9/10	2:09
10	arijit067	9/10	2:13

1. Find the username with the highest marks from the Students table

```
SELECT Username, Marks
```

```
FROM Students
```

```
WHERE Marks = (SELECT MAX(Marks) FROM Students);
```

Find the second highest marks from the table.

```
SELECT Name, Marks
```

```
FROM Students
```

```
WHERE Marks = (
```

```
    SELECT MAX(Marks)
```

```
    FROM Students
```

```
    WHERE Marks < (SELECT MAX(Marks) FROM Students)
```

```
);
```

Databases & Shared Responsibility on AWS

Many databases technologies could be run on EC2,
but you must handle yourself

- The resiliency,
- Backup,
- Patching,
- High availability,
- Fault tolerance,
- Scaling

AWS RDS Overview

What is Amazon RDS?

- A **fully managed relational database service** supporting MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, and Amazon Aurora.
- Handles database administration tasks (backups, patching, scaling).

Key Benefits

- High availability with Multi-AZ deployments.
- Automated backups and point-in-time recovery.
- Scalable compute and storage.



Supported Database Engines

Available Engines

1. MySQL : Open-source, widely used for web apps.
2. PostgreSQL : Advanced features, JSON support.
3. MariaDB : MySQL fork with enhanced performance.
4. Oracle : Enterprise -grade, licensed.
5. SQL Server : Microsoft's relational database.
6. Amazon Aurora : AWS-optimized MySQL/PostgreSQL-compatible DB.



Key Features

1. Automated Backups

- Daily backups + transaction logs (retained for up to 35 days).
- Restore to any point within the retention period.

2. Multi-AZ Deployments

- Synchronous replication to a standby instance in another AZ.
- Automatic failover during outages.

3. Read Replicas

- Asynchronous replication for read-heavy workloads (up to 15 replicas).
- Available for MySQL, PostgreSQL, MariaDB, and Aurora.

Use Cases

1. Web Applications

- Host dynamic websites (e.g., WordPress, e-commerce).

2. Enterprise Applications

- Run ERPs, CRMs, or financial systems.

3. Analytics

- Use read replicas for reporting without impacting production.

Scaling RDS

Vertical Scaling

- Increase instance size (e.g., from db.t3.small to db.r5.large).
- Requires downtime for some engines.

Horizontal Scaling

- Add read replicas to distribute read traffic.

Storage Scaling

- Automatically scales up (no downtime) for most engines.

Best Practices

1. Backups

- Enable automated backups and test restores.

2. Monitoring

- Use CloudWatch for CPU, memory, and storage metrics.

3. Cost Optimization

- Use Reserved Instances for long -term workloads.
- Delete unused instances.

AWS ElastiCache

- The same way RDS is to get managed Relational Databases ...
- ElastiCache is to get managed Redis or Memcached
- **Caches are in -memory databases** with high performance, low latency
- Helps reduce load off databases for read intensive workloads





Thank You

Feeling gratitude and not expressing it is like
wrapping a present and not giving it.

AWS Security: KMS, WAF, and S

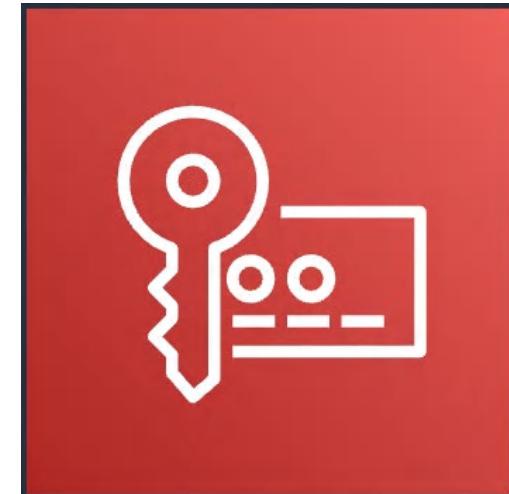
Advanced Cloud Security

Introduction to AWS Security

- AWS Security is based on the **Shared Responsibility Model** .
- AWS provides built-in security tools like **KMS, WAF, and Shield** to help protect applications and data.
- Security services help in
 - a. encryption,
 - b. access control, and
 - c. DDoS protection .

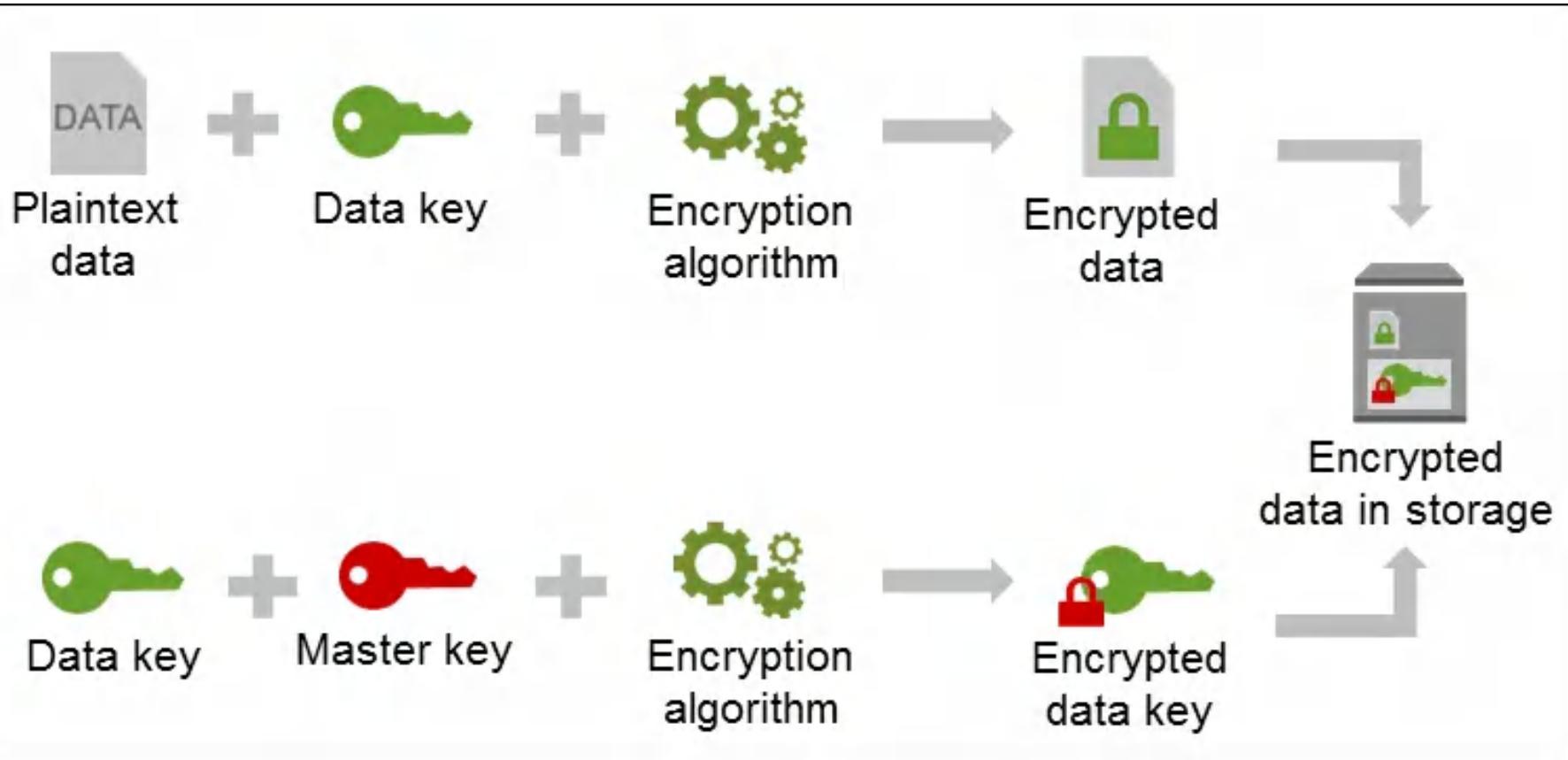
AWS Key Management Service (KMS)

- AWS KMS is a managed service for creating and managing encryption keys.
- It supports AWS services like S3, EBS, and RDS for data encryption.
- Key policies and IAM permissions control access to keys.
- Integration with CloudTrail for logging key usage.

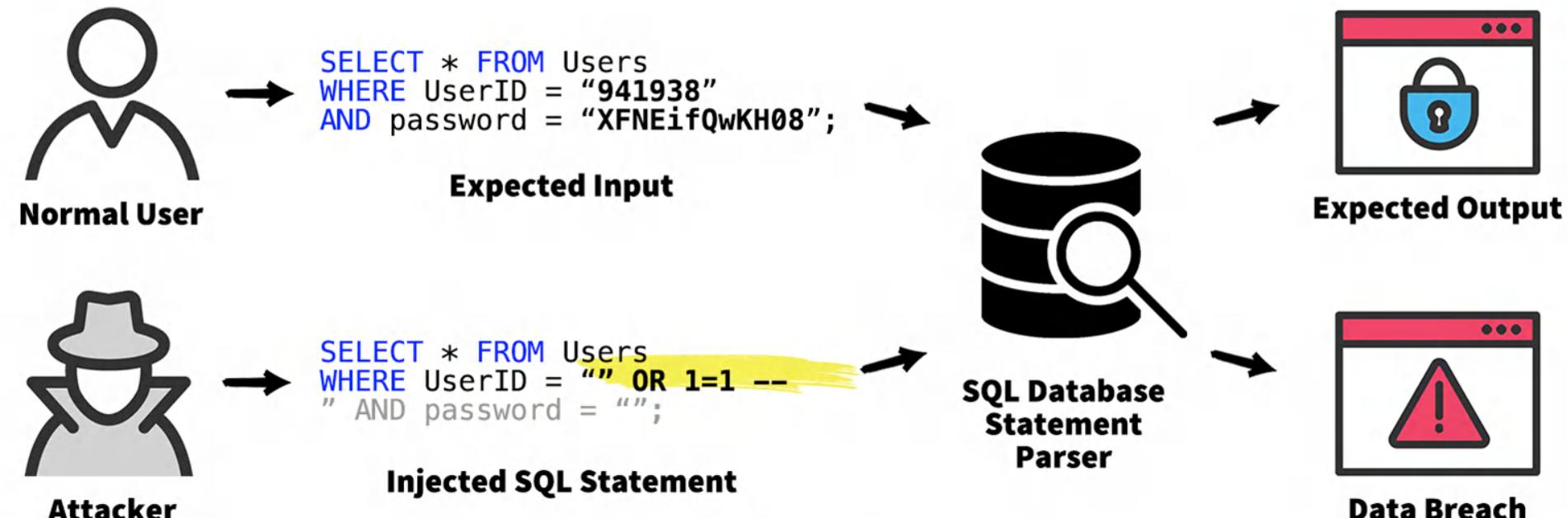


How AWS KMS Works

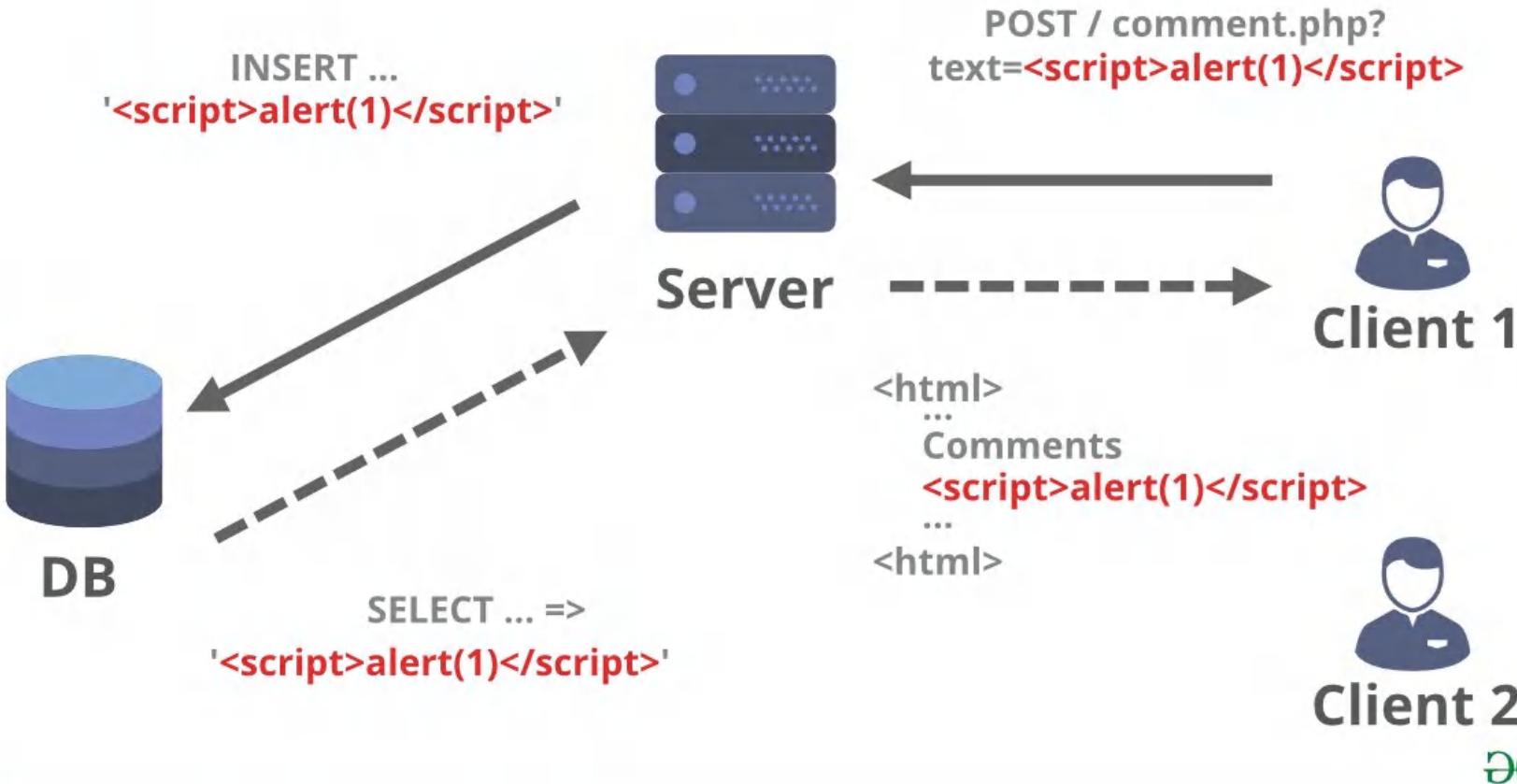
- Uses **Customer Master Keys (CMKs)** to encrypt and decrypt data.
- Supports **Automatic Key Rotation** for security.
- **Envelope Encryption** : KMS encrypts a data key, which then encrypts the data.
- Integrated with AWS services like **Lambda, S3, RDS, and CloudTrail** .



What is SQL Injection?

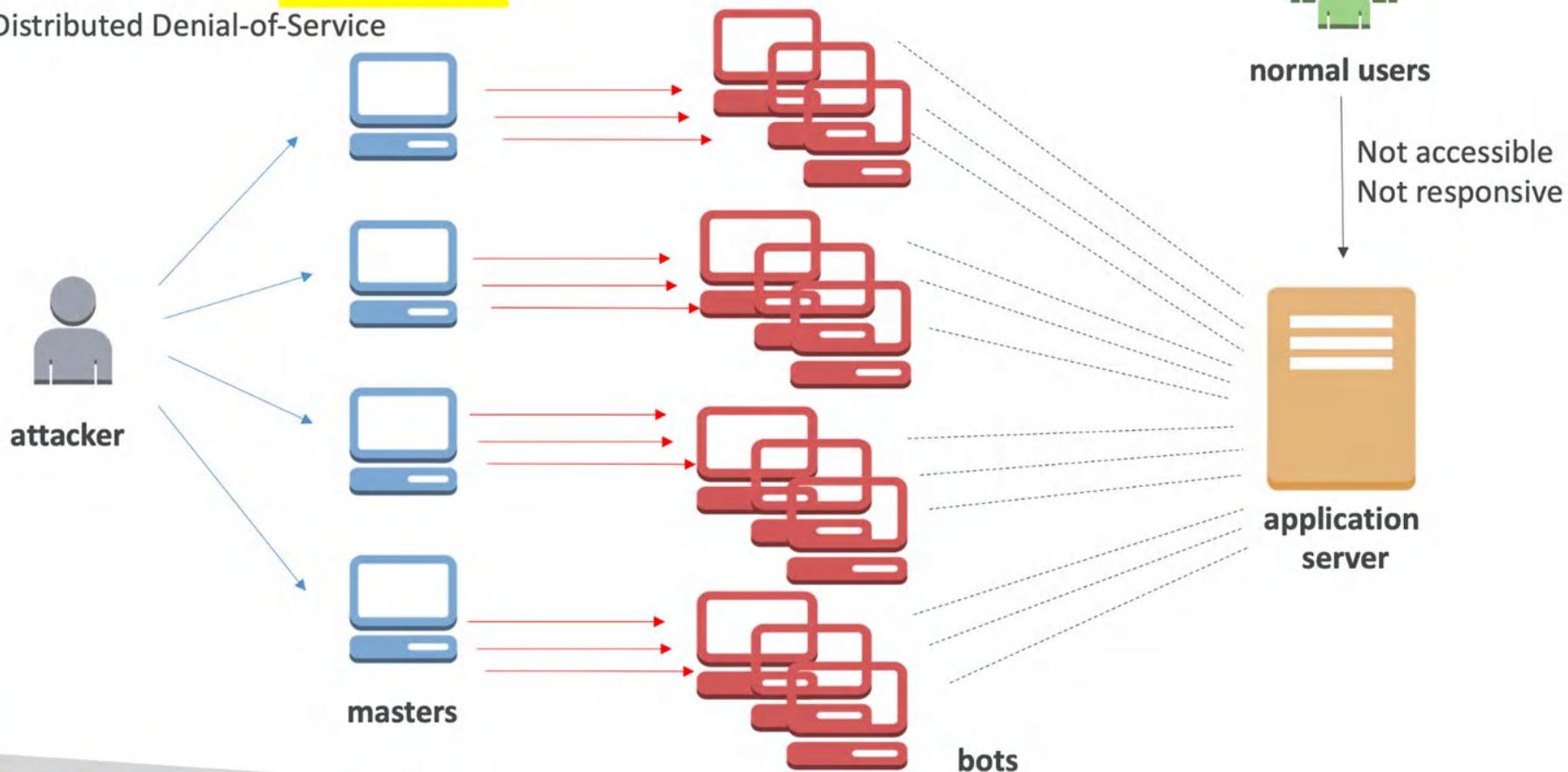


Cross Site Scripting(XSS)



What's a DDoS* Attack?

*Distributed Denial-of-Service



AWS Web Application Firewall (WAF)

- AWS WAF protects applications from SQL Injection, XSS, and DDoS attacks .
- Works with ALB, API Gateway, and CloudFront .
- Uses WebACLs (Web Access Control Lists) for filtering requests.
- Can block, allow, or monitor requests based on rules.

How AWS WAF Works

- WebACL Rules define what traffic is allowed or blocked.
- Managed Rules help block common threats.
- Rate-based rules protect against brute force and DDoS attacks .
- Works with Shield for enhanced DDoS protection .

AWS Shield for DDoS Protection

- AWS Shield provides DDoS protection for applications.
- Two types:
 1. AWS Shield Standard (free, automatic protection)
 2. AWS Shield Advanced (paid, 24/7 response, insurance)
- Integrated with CloudFront, Route 53, and WAF .

AWS Security Best Practices

- Enable AWS Shield and WAF for web applications.
- Use KMS for encrypting sensitive data .
- Implement IAM policies with least privilege access .
- Monitor security events using CloudWatch and CloudTrail.
- Regularly update WAF rules to prevent new threats.

Summary

- AWS KMS encrypts and manages keys securely.
- AWS WAF protects applications from common web threats.
- AWS Shield provides DDoS mitigation .
- Security best practices involve **encryption, monitoring, and access control** .



Thank You

Feeling gratitude and not expressing it is like
wrapping a present and not giving it.

Capacity Planning & Cloud Broker

Resource Allocation, Cost Optimization,
and the Cloud Broker Role

Overview

Key Sections:

1. What is Capacity Planning?
2. Resource Allocation in Cloud Environments
3. Role of Cloud Brokers
4. Cost Optimization Strategies
5. Integration of Capacity Planning & Cloud Brokers
6. Case Study/Examples
7. Challenges & Best Practices

Capacity Planning

Capacity planning is a strategic process that helps businesses ensure they have the IT resources to meet customer demand

Objectives:

- Avoid over/under-provisioning.
- Ensure scalability.
- Minimize costs.



CLOUD CAPACITY

Tonex Training

Key Components:

- Demand forecasting.
- Resource scalability planning.
- Performance monitoring.

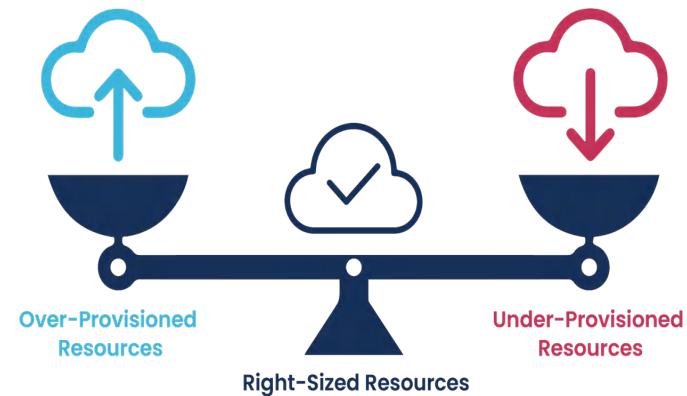
Resource Allocation in Cloud Environments

Definition : Assigning cloud resources to workloads efficiently.

Static vs. Dynamic allocation:

- **Static** : Predefined resource allocation.
- **Dynamic** : Adjusts based on real-time needs.

Autoscaling : AWS Auto Scaling, Kubernetes Horizontal Pod Autoscaler.



Challenges:

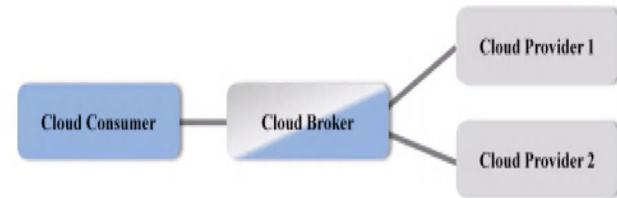
- Over-provisioning (wasted costs).
- Under -provisioning (performance issues).

Role of Cloud Brokers

Cloud Brokers: A third -party intermediary that manages cloud services across multiple providers.

Responsibilities:

- Vendor selection and negotiation.
- Cost optimization through pricing analysis.
- Centralized monitoring and governance.



Benefits:

- Simplified multi -cloud management.
- Reduced administrative overhead.

Cost Optimization Strategies

- **Reserved Instances:** Pre-purchase discounted capacity.
- **Spot Instances:** Use unused cloud capacity at lower rates.
- **Right -Sizing:** Match resources to actual workload needs.
- **Automated Tools:** Leverage AI/ML for cost analytics.



Challenges:

- Hidden costs (e.g., data transfer fees).
- Complex pricing models.

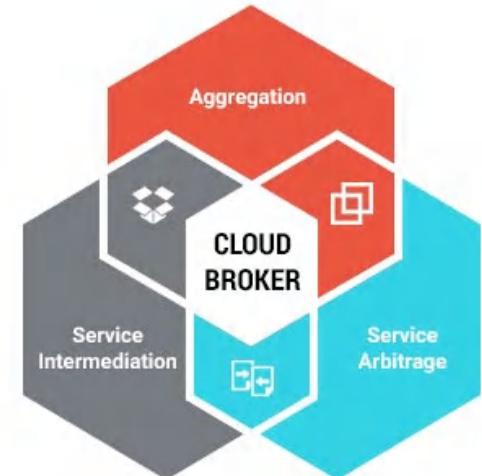
Integrating Capacity Planning & Cloud Brokers

How Brokers Assist:

- Analyze historical data for accurate forecasting.
- Automate resource scaling based on policies.
- Negotiate better pricing with providers.

Benefits:

- Reduced manual intervention.
- Improved cost -performance balance.



Case Study/Examples

A company uses AWS EC2 with a cloud broker tool to:

- Forecast peak holiday traffic.
- Automatically scale resources.
- Reduce costs by 30% using reserved instances.

Outcome:

- Zero downtime during traffic spikes.
- 25% lower annual cloud spend.

Challenges & Best Practices

Challenges:

- Unpredictable demand spikes.
- Vendor lock -in risks.

Best Practices:

- Regularly review capacity plans.
- Use multi -cloud strategies to avoid lock -in.
- Monitor costs with dashboards.

Conclusion

- Capacity planning ensures efficient resource use.
- Cloud brokers simplify multi -cloud management.
- Cost optimization requires strategic tools and planning.

Service Level Agreements (SLAs) in Cloud Computing

Ensuring Accountability and
Performance

What is an SLA?

- Formal agreement between a cloud service provider (CSP) and a customer.
- Defines measurable metrics like **uptime** , **response time** , and **support** .
- Legally binding document to ensure accountability.



Key Components of an SLA

- 1. Service Scope** : What services are covered?
- 2. Performance Metrics** : Uptime (e.g., 99.9%), latency, etc.
- 3. Responsibilities** : Roles of CSP vs. customer.
- 4. Penalties** : Credits/refunds for unmet metrics.
- 5. Exclusions** : Situations where SLA doesn't apply (e.g., outages due to customer error).

Types of Cloud SLAs

- **Standard SLA** : Predefined terms for all customers (e.g., public cloud providers).
- **Custom SLA** : Tailored for enterprise clients with specific needs.
- **Multi -tier SLA** : Different levels for different services (e.g., storage vs. compute).

Importance of SLAs in Cloud Computing

- Ensures transparency between CSP and customer.
- Minimizes downtime risks with uptime guarantees.
- Provides legal recourse for breaches.
- Builds trust in cloud adoption.

Common SLA Metrics

- **Uptime** : e.g., 99.9% ("three nines") = ~8.76 hours downtime/year.
- **Response Time** : Time to address support tickets.
- **Data Availability** : Redundancy and backup guarantees.
- **Security** : Compliance certifications (e.g., ISO 27001).

Calculate the allowable downtime

Determine Total Time in Desired Period

For a year:

$365 \text{ days/year} \times 24 \text{ hours/day} = 8760 \text{ hours/year.}$

- Convert Uptime Percentage to Downtime Fraction

Example for 99.9% uptime:

$$1 - 0.999 = 0.001 \text{ (or } 0.1\%)$$

- Calculate Downtime Duration

Multiply downtime fraction by total time:

$$0.001 \times 8760 \text{ hours} = 8.76 \text{ hours/year.}$$

General Formula:

$$\text{Downtime} = (1 - \text{Uptime Percentage}) \times \text{Total Time in Period}$$

Examples for Common Uptime SLAs:

Uptime (%)	Downtime per Year	Downtime per Month (30 days)
99.9%	~8.76 hours	~43.8 minutes
99.99%	~52.56 minutes	~4.38 minutes
99.999%	~5.26 minutes	~26.3 seconds

Challenges with SLAs

- **Complexity** : Hard to measure metrics like performance.
- **Vagueness** : Ambiguous terms (e.g., "best effort").
- **Multi -tenancy** : Shared resources may impact performance.
- **Enforcement** : Difficulty claiming penalties.

Best Practices for SLAs

- Negotiate clear, measurable metrics.
- Understand exclusions (e.g., maintenance windows).
- Monitor compliance with third-party tools.
- Review and update SLAs as needs evolve.

RealWorld Example

1. AWS EC2 SLA: 99.99% uptime for Enterprise Support.
2. Azure: Credits for downtime exceeding SLA thresholds.
3. Google Cloud: Tiered SLAs for different services.

Sample Question

Kloud 360, a company uses a cloud provider with a 99.95% uptime SLA. Last year, the service was down for 4 hours due to a server failure and 1 hour during scheduled maintenance.

- I. What is SLA? Difference between a standard SLA and a custom SLA?
- II. Did the provider breach the SLA?
- III. What recourse does Company Kloud 360 have if the SLA was breached?

What is SLA (Service Level Agreement)?

[Create an AWS Account](#)

What is a Service Level Agreement (SLA)?

What are the types of service level agreements?

What are the common elements of a service level agreement?

What are some examples of metrics that service level agreements cover?

What factors should I consider when setting metrics for a service level agreement?

How can a client monitor vendor performance against the service level agreement?

What kind of penalties can service providers incur?

What is a Service Level Agreement (SLA)?

A service level agreement (SLA) is an outsourcing and technology vendor contract that outlines a level of service that a supplier promises to deliver to the customer. It outlines metrics such as uptime, delivery time, response time, and resolution time. An SLA also details the course of action when requirements are not met, such as additional support or pricing discounts. SLAs are typically agreed upon between a client and a service provider, although business units within the same company can also make SLAs with each other.

What are the types of service level agreements?

Here are some common types of service level agreements (SLAs).

Customer-level SLA

A customer-based SLA is an agreement that covers all of the services used by a customer. A customer service level agreement covers specific details of services, provisions of service availability, an outline of responsibilities, escalation procedures, and terms for cancellation.

Service-level SLA

A service-based SLA is an agreement that defines the quality of service provided by a vendor to a customer.



Summary

- SLAs are critical for defining expectations in cloud services.
- Focus on metrics, penalties, and clarity.
- Always review and monitor SLAs actively.

CI/CD: Streamlining Software Del

From Code to Production with
Speed and Confidence

DIFFERENCE BETWEEN GIT AND GITHUB



<https://git-scm.com/>

GIT

- Version Control system
- Manage source code and History



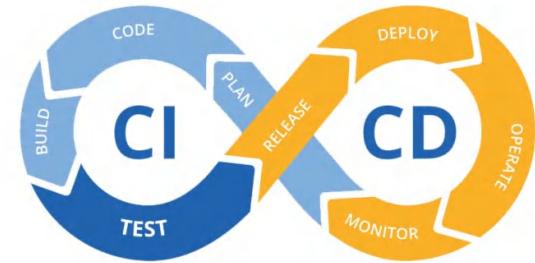
<https://github.com/>

Github

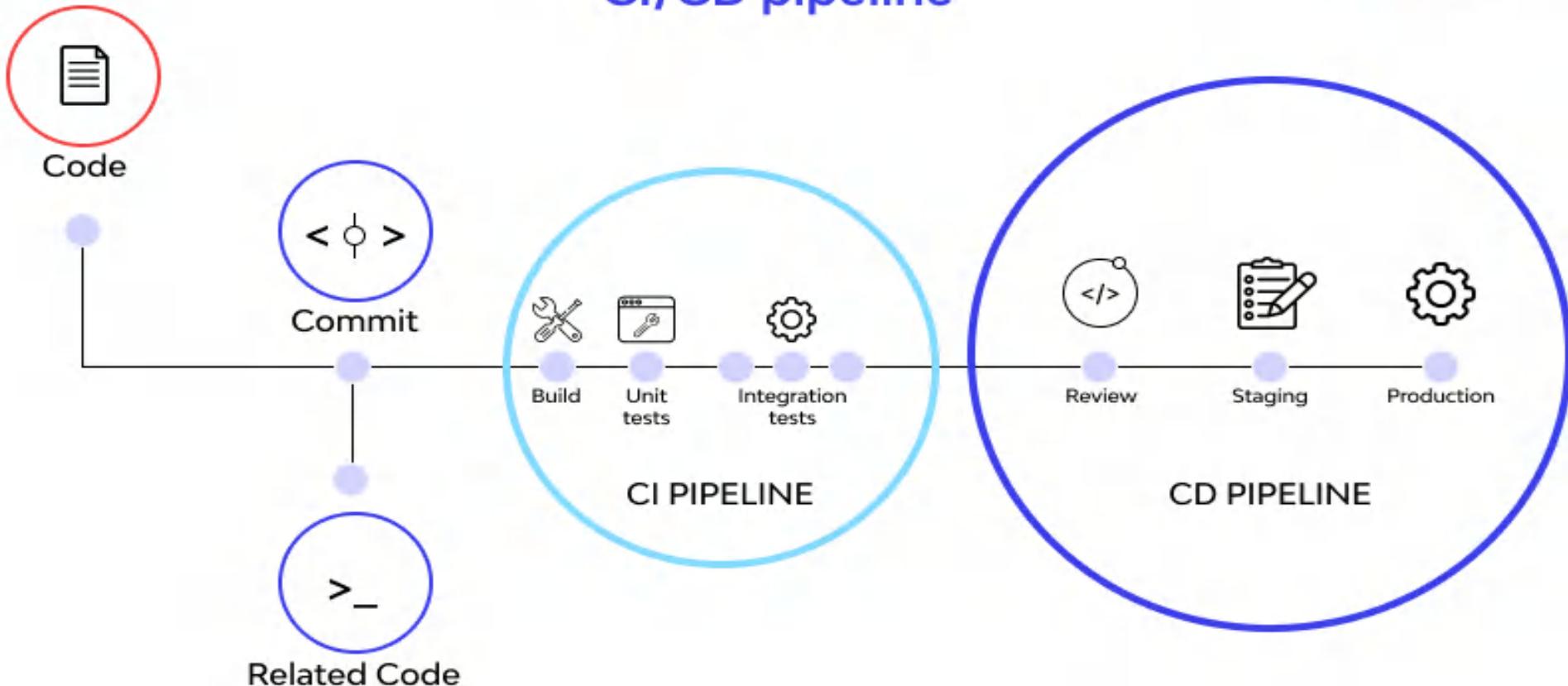
- Website to host code for projects
- Allows creation of project repositories

What is CI/CD? (Overview)

- **CI = Continuous Integration** : Frequent code merges + automated testing.
- **CD = Continuous Delivery** (auto-release to staging) or **Deployment** (auto-release to production).
- Goal: Faster, reliable software releases.



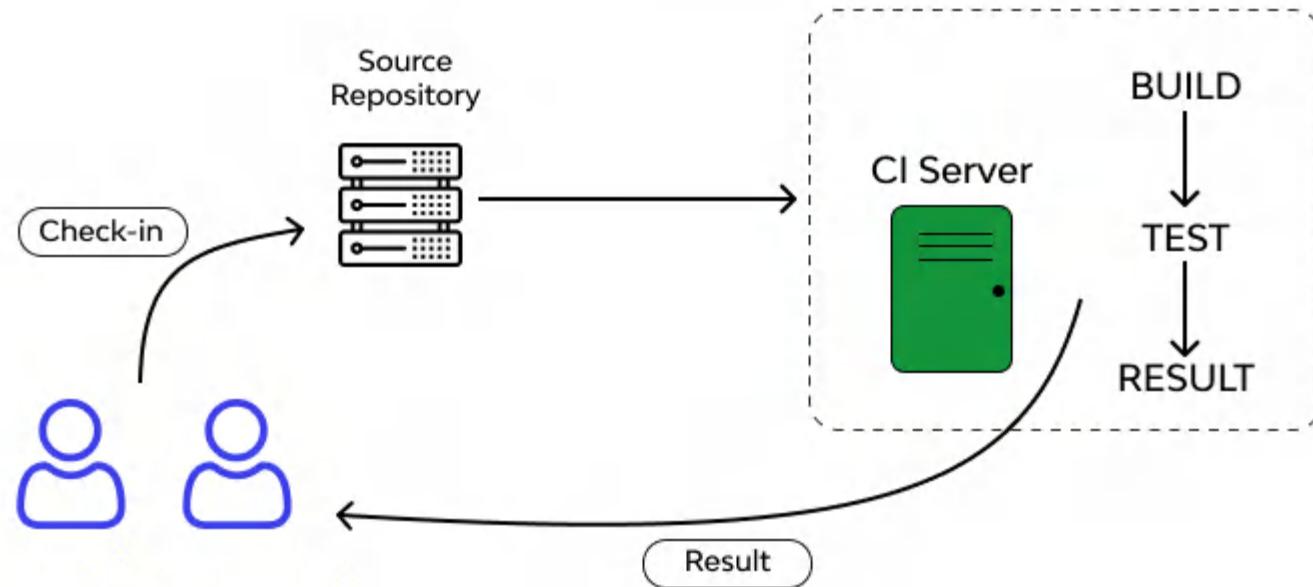
CI/CD pipeline



Continuous Integration (CI)

- Developers merge code to a shared repo **multiple times a day**.
- Automated build and test workflows catch bugs early.
- Tools: Jenkins, GitLab CI, CircleCI.

Continuous Integration (CI)



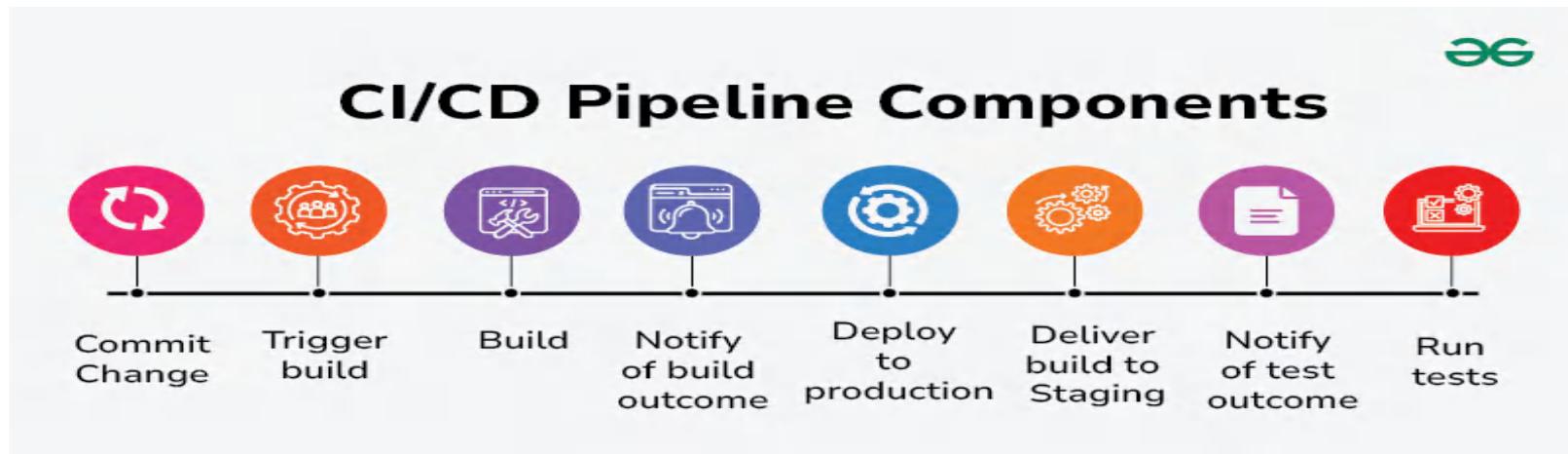
Continuous Delivery vs. Deployment (CD)

- **Delivery** : Automatically deploys to staging for manual approval.
- **Deployment** : Automatically deploys to production (no human intervention).
- Example: Netflix uses CD to deploy 1,000+ times/day.

Key Components of CI/CD

- Source Code Repo (e.g., GitHub).
- Build Server (e.g., Jenkins).
- Testing Frameworks (e.g., Selenium).
- Deployment Tools (e.g., Kubernetes).

Code commit → 2. Build → 3. Test → 4. Deploy to Staging → 5. Deploy to Production.



Benefits of CI/CD

- Faster releases.
- Fewer bugs in production.
- Better team collaboration.
- Scalability for large projects.

Challenges & Pitfalls

- Complexity in setup.
- Cultural resistance to automation.
- Security risks in automated pipelines.

Popular CI/CD Tools

- CI/CD Platforms: GitLab, GitHub Actions.
- Automation: Jenkins, Travis CI.
- Deployment: Docker, AWS CodeDeploy.

Best Practices

1. Automate everything.
2. Test in production -like environments.
3. Monitor pipelines for bottlenecks.

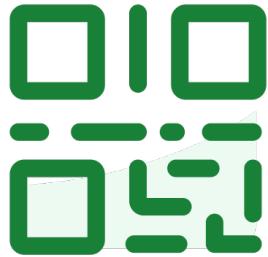
Future of CI/CD

- AI-driven test optimization.
- GitOps (infrastructure as code).
- Serverless CI/CD pipelines.

Conclusion

- CI/CD = Faster, safer, collaborative software delivery.
- Start small, iterate, and invest in automation.

Do not edit
How to change the design

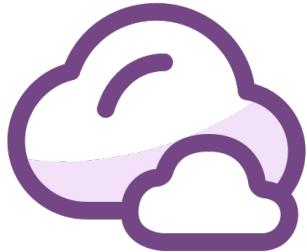


Join at slido.com
#2958053

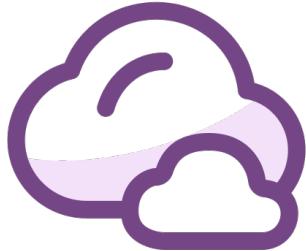


Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

slido



How many hours have you been code?



What is your expected salary?



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)



Why do we need CI CD?



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

Serverless Computing with AWS Lambda

Run applications without managing
Infrastructure

What is Serverless Computing?

- Serverless computing allows you to **build and run applications without managing servers**.
- The cloud provider handles the infrastructure, scaling, and maintenance.
- Benefits: Reduced operational cost, automatic scaling, and faster time to market.





What is the main advantage of serverless computing?

Key Characteristics of Serverless

- No server management
- Flexible scaling
- Pay-as-you-go pricing
- High availability and fault tolerance



Which of the following is NOT a characteristic of serverless computing?

Q1

What is the primary characteristic of serverless computing?

1. No servers are used at all
2. Developers manage the infrastructure manually
3. Automatic scaling and event -driven execution
4. It only supports AWS services

Q1 Answer

What is the primary characteristic of serverless computing?

1. No servers are used at all
2. Developers manage the infrastructure manually
3. Automatic scaling and event -driven execution *(Correct)*
4. It only supports AWS services

What is AWS Lambda?

- AWS Lambda is a compute service that runs code without provisioning or managing servers.
- Functions execute automatically in response to events. (like image upload)
- Supported languages include Python, Node.js, Java, Go, Ruby, and C#.





AWS Lambda runs code in response to:

How AWS Lambda Works

1. Developer writes a function and deploys it.
2. An event (such as an API call, S3 file upload) triggers the function.
3. Lambda executes the function in **an isolated environment**.
4. The function runs and then scales automatically based on demand.



Which service can trigger AWS Lambda?

AWS Lambda Use Cases

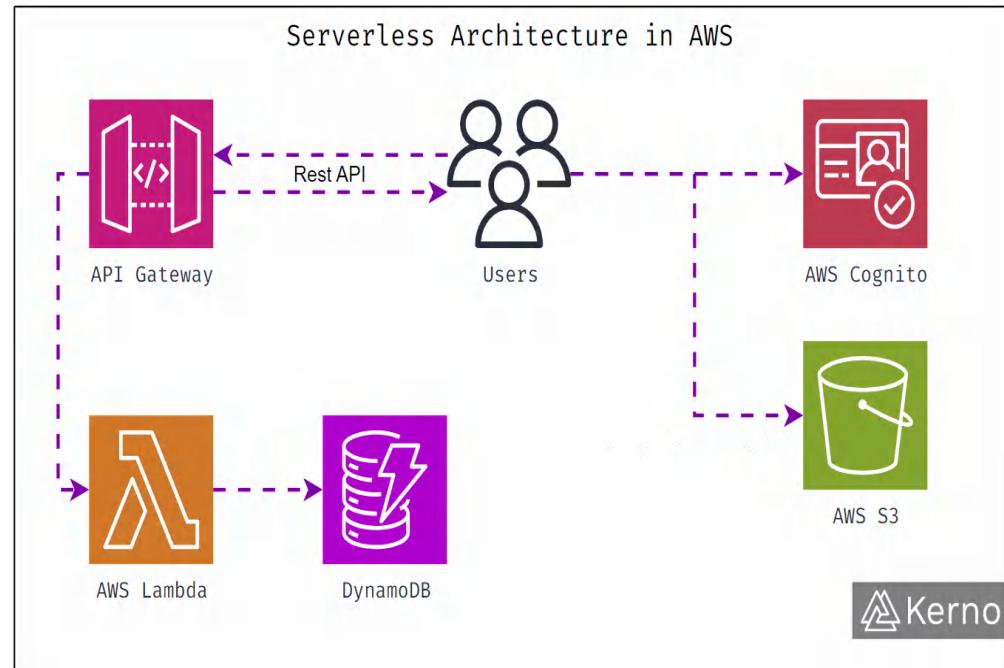
- Real-time file processing (e.g., image or video uploads)
- Data transformation and ETL (Extract, transform, and load) workflows
- Backend for web and mobile applications
- IoT backends and automated backups



Which is NOT a common use case for AWS Lambda?

Serverless Architecture with AWS Lambda

- AWS Lambda integrates with API Gateway for RESTful APIs.
- DynamoDB for data storage.
- Amazon S3 for object storage.
- Amazon SNS for notifications.





Which service is commonly used with AWS Lambda for data storage?

AWS Lambda Pricing

- Pricing is based on the number of requests and duration of code execution.
- First 1M requests per month are free.
- Duration is calculated from code execution start to end.

Pricing (US)

First 1M requests / month	Free
First 400K GB-sec / month	Free
Requests / month	\$0.20 per 1M
GB-sec / month	\$16.67 per 1M

AWS Lambda Pricing

Region:

US East (Ohio)



Architecture	Duration	Requests
x86 Price		
First 6 Billion GB-seconds / month	\$0.0000166667 for every GB-second	\$0.20 per 1M requests
Next 9 Billion GB-seconds / month	\$0.000015 for every GB-second	\$0.20 per 1M requests
Over 15 Billion GB-seconds / month	\$0.0000133334 for every GB-second	\$0.20 per 1M requests
Arm Price		
First 7.5 Billion GB-seconds / month	\$0.0000133334 for every GB-second	\$0.20 per 1M requests
Next 11.25 Billion GB-seconds / month	\$0.0000120001 for every GB-second	\$0.20 per 1M requests
Over 18.75 Billion GB-seconds / month	\$0.0000106667 for every GB-second	\$0.20 per 1M requests



AWS Lambda pricing depends on:

Exam Questions (SceBased)

A company wants to build a serverless image processing pipeline that automatically resizes images when uploaded to an S3 bucket. Which AWS services should they use, and why?

Exam Questions (SceBased)

A company wants to build a serverless image processing pipeline that automatically resizes images when uploaded to an S 3 bucket. Which AWS services should they use, and why?

They should use Amazon S 3 for file storage and AWS Lambda to process and resize images when a new file is uploaded. Lambda can be triggered by S 3 events, enabling real -time processing without server management

Q2

Your team is developing a mobile application that needs a serverless backend for user authentication, data storage, and sending notifications. Recommend AWS services to fulfill these requirements.

Q2

Your team is developing a mobile application that needs a serverless backend for user authentication, data storage, and sending notifications. Recommend AWS services to fulfill these requirements.

- AWS Lambda for backend logic
- Amazon DynamoDB for data storage
- Amazon Cognito for user authentication
- Amazon SNS for sending notifications

Q3

A client wants a cost -effective solution that sends customized emails to users based on new entries in a database. What serverless approach would you recommend?



A client wants a cost -effective solution that sends customized emails to users based on new entries in a database. What serverless approach would you recommend?

- Amazon DynamoDB for database entries
- AWS Lambda triggered by database changes
- Amazon SES (Simple Email Service) for sending customized emails

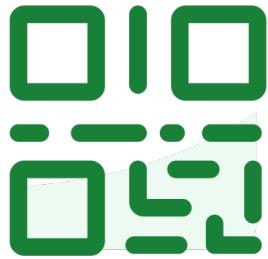


Audience Q&A



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

Do not edit
How to change the design



Join at slido.com
#4345678



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

slido



What are the use case of serverless computing?

Monitoring & Logging in AWS (CloudWatch, CloudTrail, Logging)

— Understanding application behavior and performance —

Introduction to AWS CloudFormation

AWS CloudFormation allows you to model and provision AWS resources using Infrastructure as Code (IaC).

- Automates the creation and management of AWS resources.
- Supports YAML and JSON templates.





What is the primary purpose of AWS CloudFormation?

Key Features of AWS CloudFormation

- Template -driven resource management.
- Supports rollback on failure.
- Stack creation, update, and deletion.
- Integrates with CI/CD pipelines.



Which feature allows AWS CloudFormation to revert changes when stack creation fails?

CloudFormation Stacks and Templates

- **Stack** : Collection of AWS resources managed as a single unit.
- **Template** : JSON or YAML file defining AWS resources and configurations.
- Parameters, resources, mappings, outputs, and conditions are key template components.



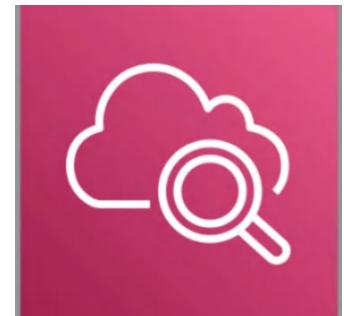
What is an AWS CloudFormation stack?

Introduction to Monitoring & Logging in AWS

Monitoring and logging are essential for understanding application behavior and performance in the cloud.



AWS provides services like CloudWatch, CloudTrail, and various logging options for complete visibility.





Which AWS services are primarily used for monitoring and logging?

Introduction to Amazon CloudWatch

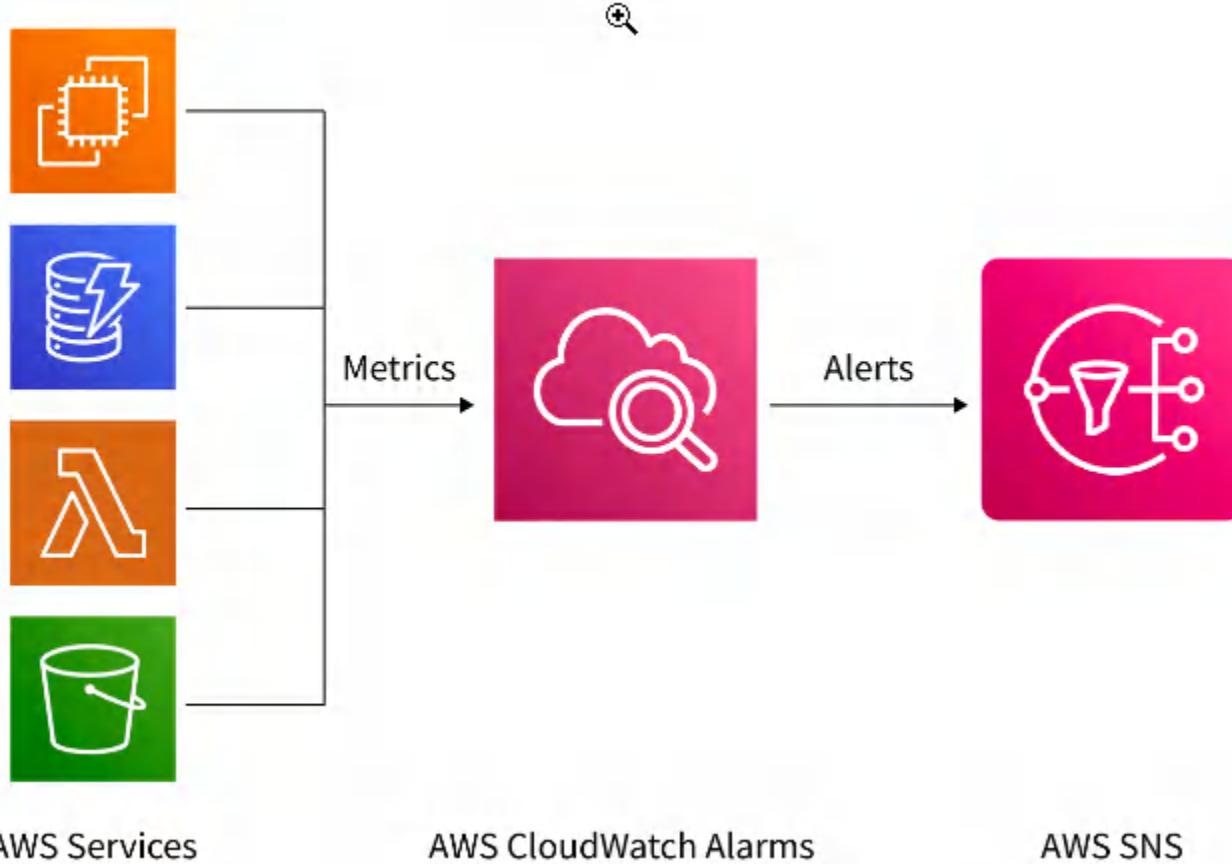
CloudWatch provides monitoring for AWS resources and applications.

Key features: metrics collection, logs, dashboards, and alarms.

Enables real-time observability and operational insights.



What is a key feature of Amazon CloudWatch?



CloudWatch Logs and Metrics

CloudWatch Logs: Store, monitor, and access log files from AWS services and applications.

CloudWatch Metrics: Provide performance data for AWS resources.

Supports real -time application and infrastructure insights.



Which component of CloudWatch stores log files?

Introduction to AWS CloudTrail

- CloudTrail records AWS API calls for account activity.
- Provides event history for auditing and compliance.
- Integrates with CloudWatch for real -time monitoring.



Which AWS service records API call history for auditing purposes?

CloudTrail Event Categories

Management Events: Track control plane operations (e.g., creating resources).

Data Events: Provide insights into data plane operations (e.g., S3 object access).

Insight Events: Detect unusual operational activities.



Which CloudTrail event category tracks S3 object access?

Logging Best Practices

1. Enable multi -region CloudTrail for comprehensive tracking.
2. Use CloudWatch Alarms to detect anomalies.
3. Centralize logs using S 3 and analyze with AWS Athena.
4. Regularly review logs for security and operational health.



Which AWS service is commonly used to store centralized logs?

Functionality	Cloud Watch	Cloud Trail
Definition	A service to monitor AWS cloud resources and the applications which are run on AWS	A service that enables governance, compliance, operational auditing, and risk auditing of your AWS account
Users question	Whats happening in AWS	Who and what is performed on AWS
Objective	Monitoring service	Monitoring service
Period	Delivers metric data within 15 min for basic monitoring and 1 min for detailed one	Delivers event within 15 min of API call

Question 1:

A company wants to track all API activities across multiple AWS regions for security auditing. Which AWS service and configuration would you recommend?

Question 1

A company wants to track all API activities across multiple AWS regions for security auditing. Which AWS service and configuration would you recommend?

- Use AWS CloudTrail with multi-region trails to record all API activities.
- Store logs in Amazon S3 for durability and use AWS Athena for querying.

Question 2:

Your application experiences performance issues during peak hours. How can you monitor and troubleshoot the root cause using AWS services?

Question 2:

Your application experiences performance issues during peak hours. How can you monitor and troubleshoot the root cause using AWS services?

- Utilize Amazon CloudWatch to set up dashboards for real-time metrics.
- Analyze CloudWatch Logs for application errors.
- Set CloudWatch Alarms to trigger notifications for performance anomalies.

Q3

A client needs to detect unusual API activities that might indicate a security threat. What AWS service and feature would you suggest?

Q3

A client needs to detect unusual API activities that might indicate a security threat. What AWS service and feature would you suggest?

- Recommend AWS CloudTrail Insights, which automatically detects unusual patterns in API activity.
- Integrate with Amazon CloudWatch for real-time alerts.

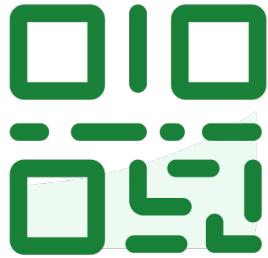


Audience Q&A



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

Do not edit
How to change the design



Join at [slido.com](https://www.slido.com)
#4178747



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

slido



What is Infrastructure as a Code (IaC)? Give an example.

Quiz 3 Topics

- 22. AWS Security: KMS, WAF, and Shield
- 23. Capacity Planning & Cloud Brokers
- 24. Service-Level Agreements (SLAs) in Cloud Computing
- 25. CI_CD_ Streamlining Software Delivery
- 26. Serverless Computing
- 27. IAC(Cloudformation), Monitoring & Logging



What is your Quiz 2 experience?



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

AWS Budget and Cost Optimisation

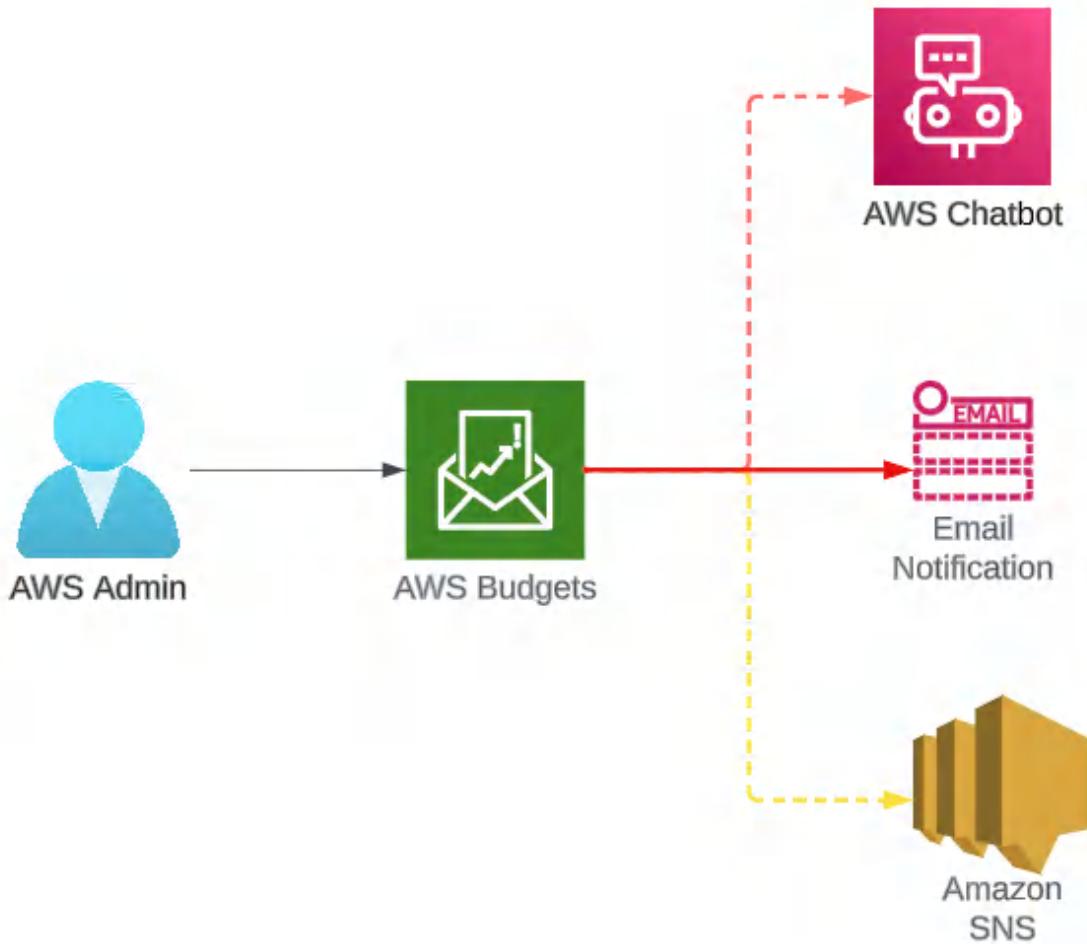
— Cost Optimization & Budgeting
Tools - Pricing Calculator —

Introduction to Cost Optimization & Budgeting Tools

- Cost Optimization in AWS focuses on minimizing expenses while maintaining performance and security.
- Key Factors:
 - Right-Sizing Resources
 - Choosing the Right Pricing Model (On-demand, Reserved Instances, Spot Instances)
 - Monitoring and Automation



AWS Budgets





Which of the following is NOT a key factor in AWS cost optimization?

AWS Pricing Models Overview

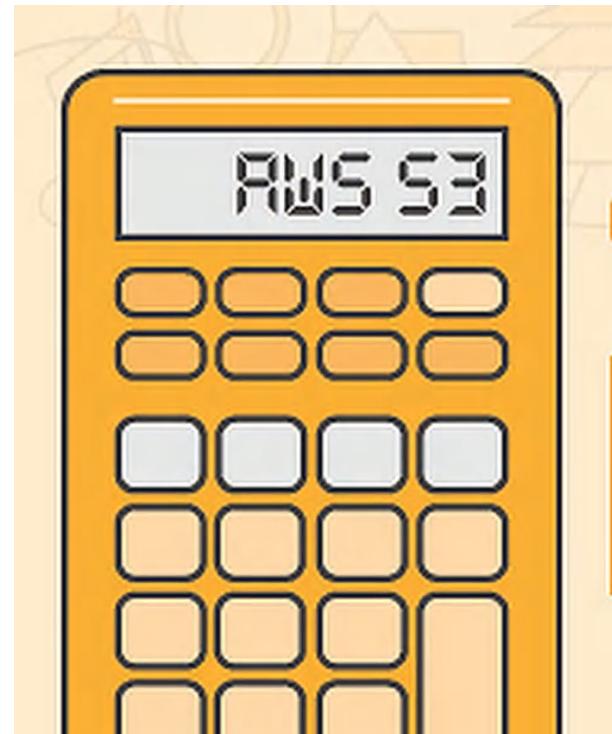
- **On-Demand Instances:** Pay-as-you-go.
- **Reserved Instances:** 1 to 3-year commitment for discounted pricing.
- **Spot Instances:** Up to 90% discount but can be interrupted.
- **Savings Plans:** Flexible pricing for consistent usage.



Which AWS pricing model provides the highest discount but with the risk of interruption?

What is AWS Pricing Calculator?

- A tool to estimate AWS service costs.
- Helps in planning and forecasting.
- Supports all major AWS services.
- URL:
 - <https://calculator.aws/>



Find the QPS and Storage Requirement

Example: Estimate Twitter QPS and storage requirements

Please note the following numbers are for this exercise only as they are not real numbers from Twitter.

Assumptions:

- 300 million monthly active users.
- 50% of users use Twitter daily.
- Users post 2 tweets per day on average.
- 10% of tweets contain media.
- Data is stored for 5 years.

Solution:

Estimations:

Query per second (QPS) estimate:

- Daily active users (DAU) = 300 million * 50% = 150 million
- Tweets QPS = 150 million * 2 tweets / 24 hour / 3600 seconds = ~3500
- Peak QPS = 2 * QPS = ~7000

We will only estimate media storage here.

- Average tweet size:
 - tweet_id 64 bytes
 - text 140 bytes
 - media 1 MB
- Media storage: 150 million * 2 * 10% * 1 MB = 30 TB per day
- 5-year media storage: 30 TB * 365 * 5 = ~55 PB



What is the primary purpose of AWS Pricing Calculator?

Key Features of AWS Pricing Calculator

- **Detailed Estimates:** Resource-level cost breakdown.
- Export and Share Estimates.
- **Customization:** Customize pricing based on regions, storage, data transfer, etc.
- **Savings Estimates:** Compare On-Demand vs Reserved pricing.



Which of the following is NOT a feature of AWS Pricing Calculator?

Using AWS Pricing Calculator Step by Step

1. Choose Services – Select EC2, S3, RDS, etc.
2. Configure Requirements – Instance type, storage, region.
3. Estimate Costs – Check pricing breakdown.
4. Compare Pricing Models – On-demand vs Reserved.
5. Export & Share Report.



Which is the first step when using AWS Pricing Calculator?

AWS Budgets Tool

What is AWS Budgets?

- Set custom cost and usage budgets.
- Receive alerts via email or SNS.
- Track actual vs forecasted spend.

Supports : Service, Linked Accounts, Tags.



AWS Budgets

Billing & Cost Management Dashboard

[Cost Management](#)[Cost Explorer](#)[Budgets](#)[Budgets Reports](#)[Savings Plans](#)[Cost & Usage Reports](#)[Cost Categories \(beta\)](#)[Cost allocation tags](#)[Billing](#)[Bills](#)[Orders and invoices](#)[Credits](#)[Preferences](#)[Billing preferences](#)[Payment methods](#)[Consolidated billing](#)[Tax settings](#)

Getting Started with AWS Billing & Cost Management

- Manage your costs and usage using [AWS Budgets](#)
- Visualize your cost drivers and usage trends via [Cost Explorer](#)
- Dive deeper into your costs using the Cost and Usage Reports with Athena integration
- [Learn more:](#) Check out the [AWS What's New webpage](#)

Do you have Reserved Instances (RIs)?

- Access the RI Utilization & Coverage reports—and RI purchase recommendations—via [Cost Explorer](#).

[Bill Details](#)

Month-to-Date Spend by Service

The chart below shows the proportion of costs spent for each service you use.



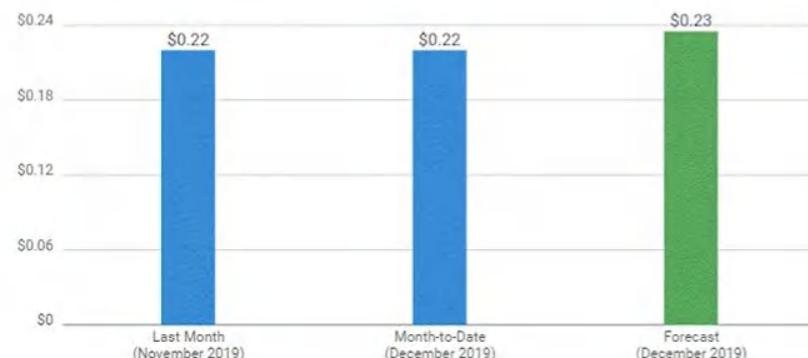
Spend Summary

[Cost Explorer](#)

Welcome to the AWS Billing & Cost Management console. Your last month, month-to-date, and month-end forecasted costs appear below.

Current month-to-date balance for December 2019

\$0.22



GuardDuty	\$0.15
S3	\$0.07
CloudTrail	\$0.00
CloudWatch	\$0.00
Other Services	\$0.00
Tax	\$0.00
Total	\$0.22

New cost and usage report

Recent reports ▾

Save to report library

Cost and usage graph Info

Total cost

\$148,547.11

Average monthly cost

\$24,757.85

Service count

44

Costs (\$)

100K

75K

50K

25K

0

May 2023

Jun 2023

Jul 2023

Aug 2023

Sep 2023

Oct 2023

█ Savings Plans for Compute usage
 █ Relational Database Service
 █ EC2-Instances
 █ EC2-Other
 █ QuickSight
 █ OpenSearch Service
█ ElastiCache
 █ Elastic Load Balancing
 █ Elastic Container Service for Kubernetes
 █ Others

Report parameters

Time

Date Range - New capability

2023-05-01 — 2023-10-31

Displaying last 6 months

Granularity

Monthly

Group by

Dimension - New capability

Clear

Service

Filters - New capability

Info

Applied filters (0)

Clear all

Service

Clear

Choose services

▼

Linked account

Clear

Choose linked accounts

▼

Region

Clear

Choose regions

▼

Instance type

Clear

Choose instance types

▼

Usage type

Clear

Choose usage types

▼



What can AWS Budgets NOT alert you about?

AWS Cost Explorer

What is AWS Cost Explorer?

- Visualize and analyze AWS spending.
- Track spend trends over time.
- Forecast future costs.

Granular Filtering: By service, linked account, region, etc.



What does AWS Cost Explorer primarily help with?

Best Practices for Cost Optimization

1. Enable AWS Budgets & Alerts.
2. Use Reserved & Spot Instances when possible.
3. Right-size Instances & Storage.
4. Use Auto Scaling.
5. Monitor with Cost Explorer.



Which of the following is NOT a cost optimization best practice?

Real Life scenario

A startup is building a web application on AWS. Initially, they deployed all EC2 instances as On-Demand. Their monthly AWS bill is higher than expected. They want to reduce costs without compromising performance for predictable workloads.

Real Life scenario

A startup is building a web application on AWS. Initially, they deployed all EC2 instances as On-Demand. Their monthly AWS bill is higher than expected. They want to reduce costs without compromising performance for predictable workloads.

- Use Reserved Instances for predictable traffic.
- Use Spot Instances for flexible or batch workloads.
- Set up AWS Budgets for monthly cost alerts.
- Use AWS Pricing Calculator to estimate and compare costs.

Questions 2

A company has moved its internal CRM system to AWS. They want to predict their monthly cloud bill and monitor if spending exceeds 80% of their budget.

Question:

Which AWS tools should they use to estimate and monitor costs? Explain how these tools will help.

Questions

A company has moved its internal CRM system to AWS. They want to predict their monthly cloud bill and monitor if spending exceeds 80% of their budget.

Question:

Which AWS tools should they use to estimate and monitor costs? Explain how these tools will help.

- Use AWS Pricing Calculator to estimate costs based on their infrastructure.
- Use AWS Budgets to set a monthly budget and get alerts when spending exceeds 80%.

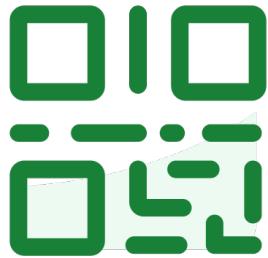


Audience Q&A



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

Do not edit
How to change the design



Join at [slido.com](https://www.slido.com)
#2619759



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

slido



How can you prevent yourself from cloud over billing?



What is your first OS?



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

Cloud Operating Systems and Services

The way to manage hardware and software resources in a cloud environment

Introduction to Cloud Operating Systems

- Cloud Operating System (Cloud OS) manages hardware and software resources in a cloud environment.
- It acts as a platform for cloud applications to run.
- Enables scalability, virtualization, and multi-tenancy.
- Examples: OpenStack, Windows Azure, Google Cloud OS.



Which of the following best defines a Cloud Operating System?

Role of Cloud Operating System

1. **Manages Virtual Machines (VMs):** Provision, monitor, and destroy VMs.
2. **Resource Management:** Allocates CPU, memory, and storage dynamically.
3. **User & Access Management:** Multi -user, multi -tenant support.
4. **Scalability & Elasticity:** Supports automatic scaling.
5. **Automation:** Automated deployment, monitoring, and fault tolerance.



Which of the following is NOT a key function of a Cloud OS?



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

Types of Cloud Operating Systems

- Open Source Cloud OS : OpenStack, CloudStack.
- Proprietary Cloud OS: Microsoft Azure, Google Cloud OS.
- Hybrid Solutions: Combining private and public features.



Which of these is an open-source Cloud OS?

Cloud Servers

Introduction

- Cloud servers are virtual servers running in a cloud environment.
- Hosted by cloud providers like AWS, Azure, GCP.
- Highly scalable and can be deployed globally.
- Types: Compute, Database, Storage, App Servers.



What is a key benefit of using cloud servers?

Types of Cloud Servers

1. Compute Servers (EC2, VM Instances).
2. Database Servers (RDS, Cloud SQL).
3. Storage Servers (S3, Blob Storage).
4. Application Servers (App Engine, Elastic Beanstalk).



Which type of server is used primarily for computing in the cloud?

Key Features of Cloud Operating System

1. **Elastic Scaling:** Automatically scales resources up/down.
2. **API Access:** Provides programmatic access to all services.
3. **Resource Pooling:** Combines resources for multi-tenancy.
4. **Self-service Portals:** Web interfaces for users to manage services.
5. **Monitoring & Alerts:** Tracks usage and health metrics.



Which feature allows developers to manage cloud resources using scripts or programs?

Cloud Operating System vs Traditional Operating

Feature	Traditional OS	Cloud OS
Hardware	Local device	Distributed
cloud hardware		
Users	Single or few users	Multi -tenant
Scalability	Limited	Elastic & dynamic
automated		Automation Minimal Highly
Deployment	Manual	
& API-based		Self-service



Which of the following is a major difference between traditional OS and Cloud OS?

Security in Cloud Operating Systems

1. Data Encryption (at rest and in transit).
2. Identity & Access Management (IAM).
3. Firewall & Network Security Groups.
4. Monitoring & Intrusion Detection.
5. Compliance with standards (GDPR, HIPAA, SOC).



What is a critical security feature in Cloud OS?

Future Trends in Cloud Operating Systems

1. AI-driven management (predictive scaling and self-healing).
2. Serverless integration (managing functions instead of VMs).
3. Hybrid cloud support (seamless on-prem and cloud operations).
4. Increased security automation.
5. More compliance-ready templates.



Which future trend will improve Cloud OS automation?

What is Compliance in Cloud OS?

- Compliance refers to adhering to *laws, regulations, and standards* to protect **data, privacy, and system integrity** in cloud environments.
- Cloud Operating Systems (Cloud OS) must provide features and configurations to help organizations meet these requirements.

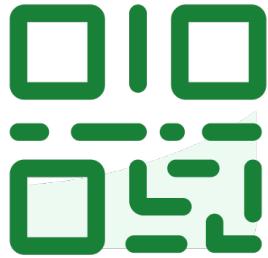
Key Standards Explained

Standard	Description	Applicable Sectors
GDPR (General Data Protection Regulation)	Protects personal data of EU citizens ; focuses on data privacy, consent, and rights to data access/erasure .	All industries handling EU data
HIPAA (Health Insurance Portability and Accountability Act)	Ensures health data protection (PHI) ; mandates security controls, audits, and breach notifications .	Healthcare providers, insurers
SOC (Service Organization Control - SOC 1, SOC 2, SOC 3)	Provides guidelines for security, availability, processing integrity, confidentiality, and privacy in cloud services.	Cloud providers, financial services



Which of the following standards focuses specifically on protecting healthcare data?

Do not edit
How to change the design



Join at [slido.com](https://www.slido.com)
#2249136



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

slido



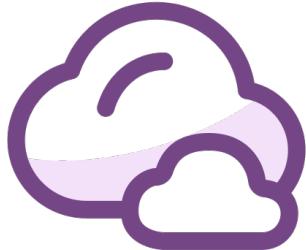
How was your experience at AUST?



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)



What would you do if you got the chance to be readmitted in the first semester?



If you were on the management committee, what changes would you make?

REVISION CLASS

— CSE4267: Cloud Computing —

1. What is Cloud Computing ?

Cloud computing is the delivery of on-demand computing services—such as servers , storage , databases , networking and more—over the internet ('the cloud') .

- Compare cloud computing to utility services like electricity or water on-demand, scalable, and pay-as-you-go



Deployment vs Service Model

Deployment Cloud Model

Public

Private

Hybrid

Service Cloud Model

IaaS

PaaS

SaaS

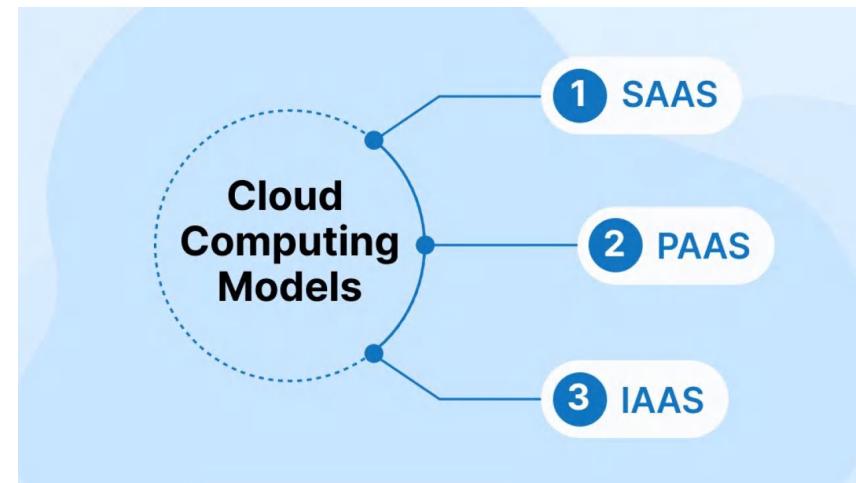
Cloud Service Models

On Premise = you buy everything and make the pizza at home

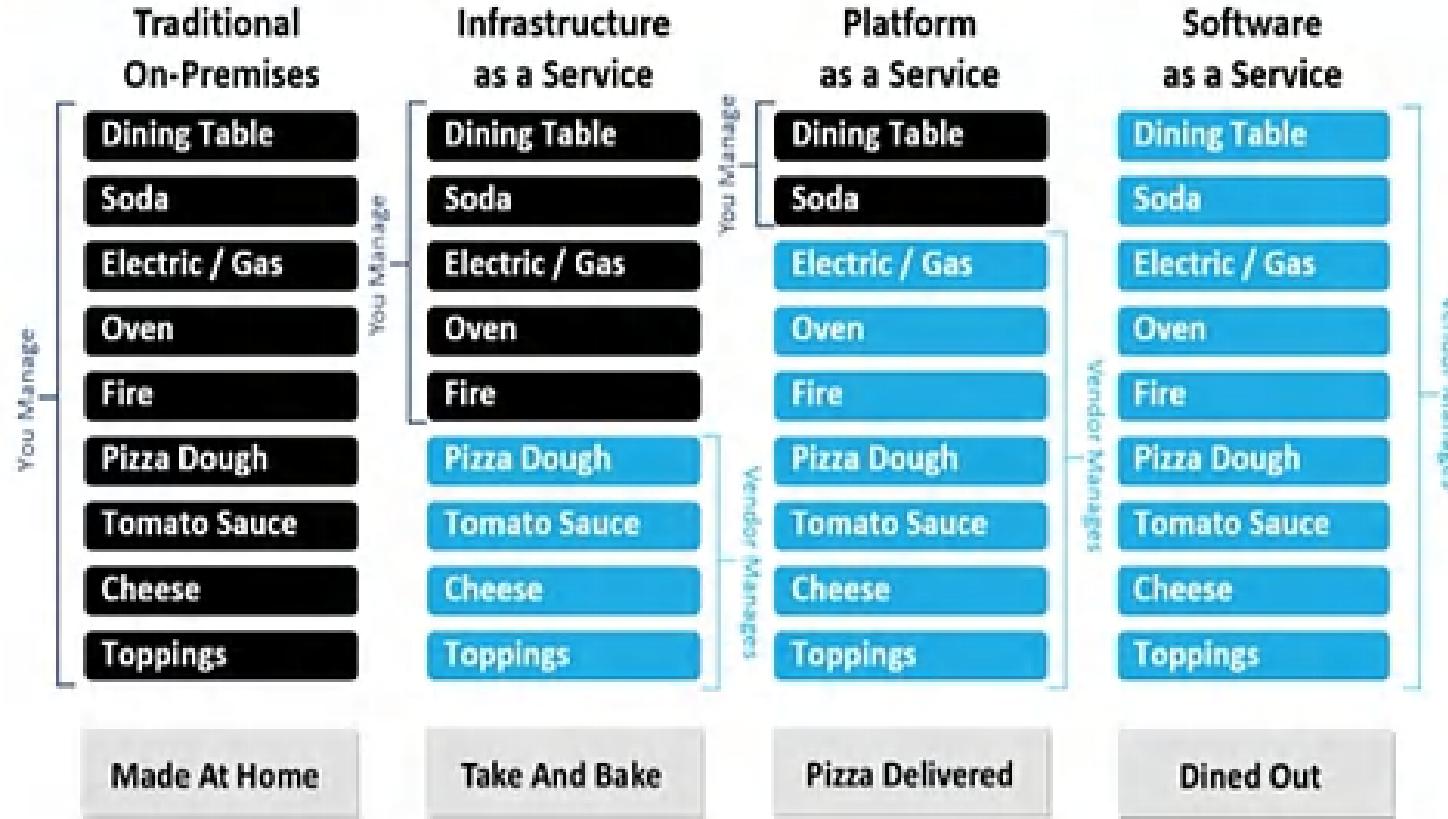
IaaS = take and bake (pick up the pizza, you cook it at home)

PaaS = pizza delivered

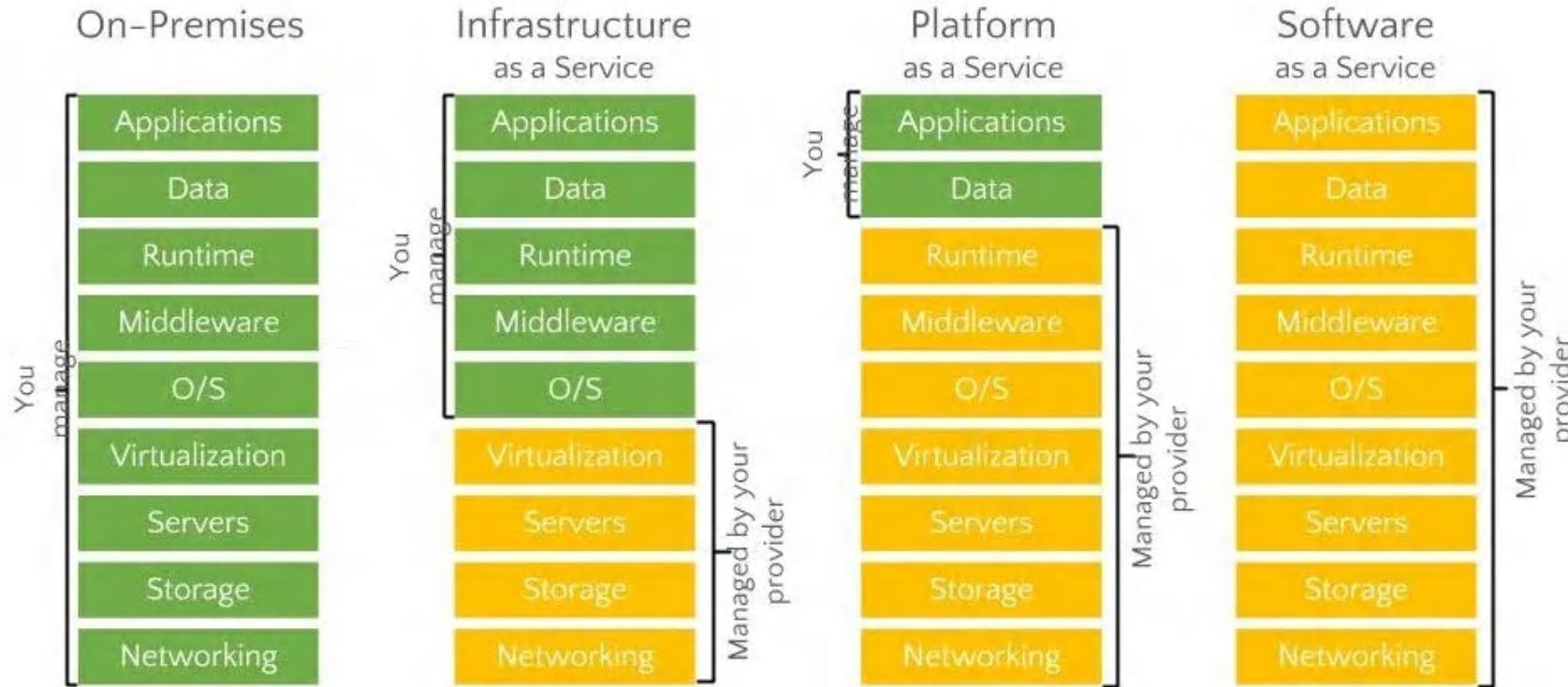
SaaS = dining in the restaurant



The Pizza & The Cloud – How Many Ways?



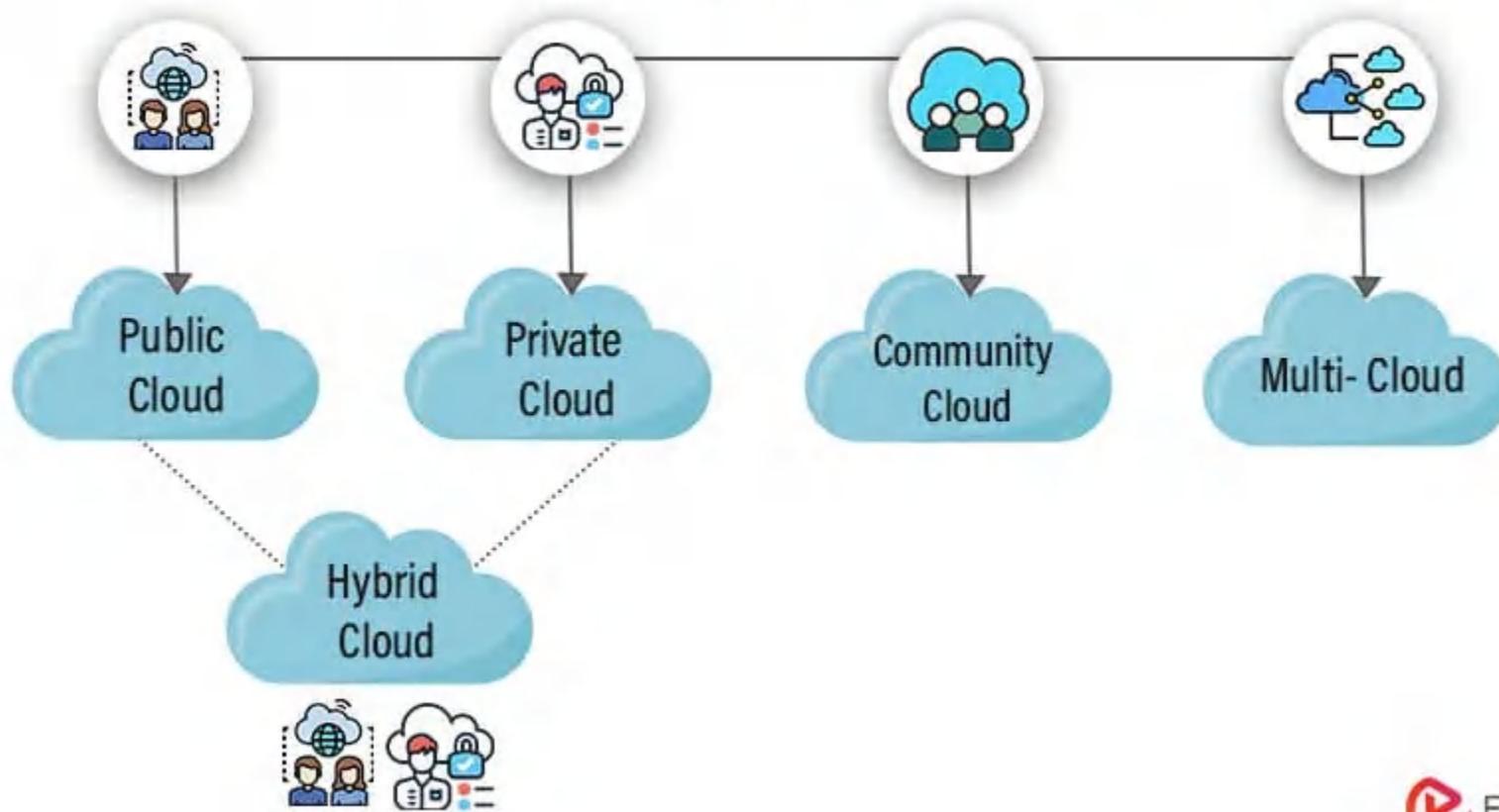
Cloud Providers Model





Which cloud model provides platforms for development?

Cloud Deployment Models



9 Reasons Why Cloud Computing is Important for Business Growth



Scalability



Innovation



Cost-Effective



Improved
DevOps



Enhanced
Compliance & Security



Flexibility



Lowered Complexity
of IT Infrastructure
Management



Multiple Service
Offerings



Continued Industry
Growth



How does cloud computing improve scalability?

Course Recap

1. Virtualization & Containerization (Docker)
2. Serverless Computing & API Gateways
3. Cloud Storage & Databases (Amazon S3, RDS)
4. Networking (VPC)
5. CDN (Amazon CloudFront)
6. Security & Compliance in the Cloud
7. Queue & Pub -Sub (Amazon SQS, SNS)

Virtualization & Containerization (Docker)

- **Virtualization:** Running multiple OS instances on a single physical machine (e.g., VMware, Hyper -V).
- **Containerization:** Lightweight, portable environments for applications (e.g., Docker, Kubernetes).
- **Use Cases:** Microservices, DevOps, efficient resource utilization.



Which of the following is a containerization tool?

Serverless Computing & API Gateways

- **Serverless Computing:** No need to manage servers (AWS Lambda, Azure Functions).
- **API Gateway:** Manages and secures API requests (AWS API Gateway).
- **Use Cases:** Scalable applications, event -driven architectures.



Which AWS service is used to manage API requests?

Cloud Storage & Databases (Amazon S3, RDS)

Amazon S3: Scalable object storage for files, backups, and static content.

Amazon RDS: Managed relational databases (MySQL, PostgreSQL, etc.).

Use Cases: Web applications, data analytics, disaster recovery.



Which AWS service is an object storage solution?

CDN (Amazon CloudFront)

Content Delivery Network (CDN) : Speeds up content delivery using global edge locations.

Amazon CloudFront : Integrates with S3, EC2, and other AWS services.

Use Cases: Website acceleration, video streaming, DDoS protection.



What is the main purpose of a CDN?



Presenting with animations, GIFs or speaker notes? Enable our [Chrome extension](#)

Security & Compliance in the Cloud

Identity & Access Management (IAM): Controls user permissions.

Encryption: Data encryption at rest and in transit (AWS KMS, SSL/TLS).

Compliance: GDPR, ISO, HIPAA, SOC for regulated industries.



Which AWS service is used for managing user access?

Queue & Pub-Sub (Amazon SQS, SNS)

Amazon SQS: Message queue for decoupling applications.

Amazon SNS: Pub-sub messaging for real -time notifications.

Use Cases: Event-driven architectures, microservices communication.



Which AWS service is used for a message queue?

Future Trends in Cloud Computing

1. AI & Machine Learning in the Cloud
2. Edge Computing & IoT
3. Multi -Cloud & Hybrid Cloud
4. Quantum Computing

Career Paths:

- Cloud Engineer
- DevOps Engineer
- Solutions Architect
- Security Specialist

Certifications

1. AWS Certified Cloud Practitioner
2. AWS Certified Solutions Architect
3. Microsoft Azure Administrator
4. Google Cloud Associate Engineer
5. Kubernetes & DevOps Certifications



If the election were held in 2025 in Bangladesh, which party would you vote for?