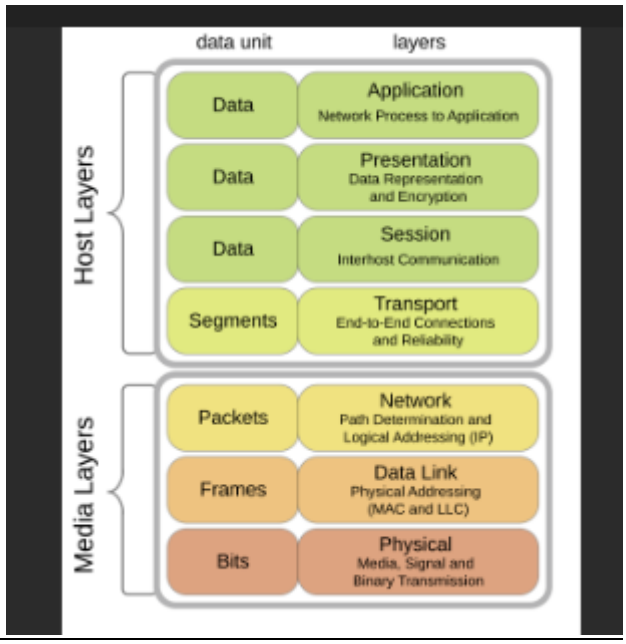
DEVOPS

Date: 8th Jan

FUNDAMENTALS OF NETWORKING

1. **Client-Server Architecture:**
   Client-server architecture is a network architecture in which two types of computers, or "nodes," communicate with each other over a network.
   Client talks to server. Client initiates the request.
   Client: Request resources
   Server: Provides Services
   Eg: Google, Instagram, fb

2. **OSI MODEL**



1. **Physical Layer(Bits):**

Function: This layer is responsible for the actual transmission of raw data (bits) over a physical medium (e.g., cables, wireless signals). It defines the hardware elements involved in data transfer.

Examples: Cables (Ethernet, fiber optic), network interface cards (NIC), switches, hubs, and wireless radio frequencies.

Protocols: Ethernet, Wi-Fi (IEEE 802.11), DSL, etc.

2. **Data link layer(Frames):**
   Function: The data link layer ensures reliable data transfer between two directly connected nodes.
   - Ensures error free
   - Manages node to node data transfer

- Error detection

3. **Network Layer: (Routing,IP addressing)-Packets**

This layer handles routing of data across different networks. It is responsible for logical addressing (IP addresses) and determining the best path for data transmission across multiple devices and networks.

- **IP address:** Unique routable address in a network which follows IP protocol.
- When you try to deliver from 1 node to another this layer helps u to determine best route for delivery.
- Sent packets and wait for confirmation and sents packets double(1,2,4,8,….)
- **Windowing**: Windowing controls how much data is sent at once, ensuring the receiver can handle it, and everything goes smoothly without errors or delays.
- **Windowing=** file download…initially showing 12 hrs..then suddenly decreasing hrs…then increases hrs…

4. **Transport Layer(Segments):**

The transport layer ensures reliable data transfer between two end systems, performing error correction and data flow control. It breaks large data into smaller segments for transmission and ensures that all pieces are received and properly ordered.

- Responsible for end to end delivery of msg
- Provide service to other layers
- Ensure data is sent in crct order/not.
- Data delivery, Error handling

5. **Session Layer**:

Dedicated Sessions happens during communication until response come
Manages session and dialogues

- Manages Connection
- Establishing conversation/Interaction
- **Connection oriented protocols(TCP,HTTP):** identifies lost packets
  Eg: Sending mails
- **Connectionless Protocol(UDP):**Don't follow….

UDP simply sends packets (datagrams) to the recipient. If packets are lost or corrupted, UDP doesn't attempt to retransmit them.

6. **Presentation Layer/Translation Layer:**
   The presentation layer is responsible for data translation, encryption, compression, and formatting.
   - Formatting data and its encryption(key)
7. **Application Layer:**
   This is the topmost layer, where user applications and network services directly interact. It provides network services to end users, such as email, file transfer, and web browsing.

3. **DATA FLOW IN OSI MODEL**

- **Application layer** creates data….This is where the application (e.g., web browser, email client) generates data to be sent to another device.
- **Presentation layer**: Formatting data and its encryption
- **Session Layer**: Connection Establishment
- **Transport Layer:** Data is broken into segments for reliable delivery.

HOW DATA FLOWS

Starts with application layer
Encoding: Always need an algorithm
Encode using an algo can be decoded as long as known the algo
Encryption:using keys

- **Application Layer** (Layer 7):
  - This is where the data is created. For example, when you send an email, this layer generates the email data.
- **Presentation Layer** (Layer 6):
  - The data is then formatted or converted (like encryption or compressing) so that it can be understood by the receiver.
- **Session Layer** (Layer 5):
  - The session is managed here. It keeps track of the conversation between the sender and receiver.
- **Transport Layer** (Layer 4):
  - The data is broken into smaller pieces (called packets or segments). This layer also makes sure the data gets delivered correctly (using protocols like TCP).
- **Network Layer** (Layer 3):
  - The data gets an **IP address** so it knows where to go. The data is sent over the best route to the destination.
- **Data Link Layer** (Layer 2):
  - The data gets a **MAC address** for local delivery. It's packaged into frames for easy transmission on the network.
- **Physical Layer** (Layer 1):
  - Finally, the data is converted into electrical signals, light pulses, or radio waves to be sent over the physical medium (like cables or Wi-Fi).

4. WHY DOES OSI MODEL MATTER?

- **Simplifies understanding**: It breaks down networking into 7 layers, making it easier to understand how data travels through a network.

- **Ensures compatibility**: It helps devices from different companies communicate smoothly by using standard rules.

- **Helps troubleshoot**: If there's a problem, you can figure out where it happened (e.g., cables, software, etc.).

5. DIFF B/W OSI MODEL AND TCP/IP MODEL?

- **OSI Model** has 7 layers, is theoretical, and is used to understand the general concept of networking.

- **TCP/IP Model** has 4 layers, is practical, and is the model used for actual communication on the Internet and networks.

6. HOST TO HOST COMMUNICATION

- Involves ……..
- packet creation
- Addressing
- Routing through network devices

RFC XMPP

Extensible Messaging and Presence Protocol (XMPP), which is a communication protocol for messaging

7. KEY COMPONENTS OF TCP/IP

- Packets not to be lost
- Divides large sets into smaller packets, and number them.

8. IP

When you send or receive data over the internet (like browsing a website or sending an email), IP is involved in determining how the data travels from one device to another.

- Restricted ones
- Allowing ones
- Addressing and routing packets across network.
- Ensures reaching correct destination
- Chances of repeated IP address=IP conflict
- PUBLIC IP
- PRIVATE IP

- ELASTIC IP

9. ICMP(Internet control msg Protocol)

- ICMP is used to report errors in communication
- Diagnosing network issues (e.g., using **ping** to check if a device is reachable).

It's a protocol used in networking to send error messages and operational information about network conditions.

10. ARP(Address Resolution Protocol)

ARP helps a device find out the **physical address** (MAC address) of another device when it knows its **logical address** (IP address).

- Converting Ip address to Mac Address(WHY?)
- When a device wants to send data to another device on the same network, it needs to know the **MAC address** (the unique physical address) of that device. But usually, the device only knows the **IP address**.
- when sending data on a local network, they use **MAC addresses**.
- Ensure the destination device is crct .

- The device sends a broadcast message asking, "Who has this IP address?" (e.g., "Who has 192.168.1.2?")
- **ARP Reply**: The device with that IP address replies with its **MAC address**.

- Now, the first device knows the MAC address and can send the data directly to the right device.

11. IGMP

MultiCast Communication.

- **IGMP (Internet Group Management Protocol)** is a protocol that helps devices on a network join or leave **multicast groups**. A multicast group is a group of devices that want to receive the same data (like video streams).

- **Router's Role**: The router uses IGMP to keep track of which devices want to receive which multicast data, so it knows where to send the information.

12. PROTOCOLS

- HTTP
  - Used for web based communication-rest using http
  - http-port number 80
  - https-port number 443
  - if u type google.com default connect on 80 on destination
  - in my machine….randomly pick up port number beyond the range of 1024

13. NETWORK INTERFACE CARD(NIC)

- inside laptop/desktop we can see
- hardware component
- also called as ethernet card.
- 2 types :wireless,wired
- In motherboard-Pci (Peripheral Component Interconnect)

PCI Slots: The motherboard has **PCI slots** (or connectors) where you can plug in expansion cards (like a **NIC**, graphics card, etc.) to add functionality to your computer.

- Other end of ethernet connected to router,modem,switch….
- **How nic works?**

- **<u>Phy card installed inside laptop(part of motherboard)=NIC</u>**
- Connect computers to network often using unique mac address.=**48bit(data link layer)**
- Sends data by breaking into packets and adds header with destination,receiving nic checks the mac address and sents data to **os.**
- follows tcp/ip protocol
- From I NIC have multiple ports also then connect to multiple networks.
- So 2 diff ip address for same machine.(2 ethernet ports)

- The NIC provides the **connection** to the network, and its **CPU processes data** that is being transmitted or received. It makes sure that the data is correctly **prepared, formatted,** and **processed** before sending it out on the network, or it does the reverse when data is coming in. However, it is not the NIC's CPU itself that physically "connects" the device to the network — it's the hardware and connectors (like Ethernet ports or Wi-Fi antennas) that do that.

Key Functions of a NIC:

1. **Network Access**

2. **Data Framing**
3. **Error Detection:ensures data integrity**
4. **Packet Forwarding**

14. SUBMARINE CABLES MAP(optical fibre cables)

- Submarine cables are fibre optic cables under oceans that enable global internet connectivity by carrying data at high speed.

15. NETWORKING TERMS AND DEVICES

- **Nodes:** any connected devices
- **Host:** computers that store and process data
- **Client/server:**
- **Protocols:** for communication

16. LAN,WAN,MAN

- **LAN**: Local area (home/office), fast speed.
- **MAN**: City or large campus area, moderate speed.

- **WAN**: Wide area (country/world), slower speed.

- **Clusters:**Group of computers or servers
- **Y should we deploy to cluster?**
- Faster access, secured connection, scalability,elasticity.
- **Scalability:**
  - **Vertical/Scale up:**
    - Adding more capacity
    - Also scale down can happen to reduce the capacity.
    - No design considerations required.
  - **Horizontal Scaling/Scale out:**
    - Add more machines as completely new
    - Add load balancers to ur machine
    - Load balanacer is used because work is loaded to only 1 device otherwise…
    - Externalize the data,the application should be ***stateless***.
    - The application stored in database to ensure the application is stateless.

17. MODEM

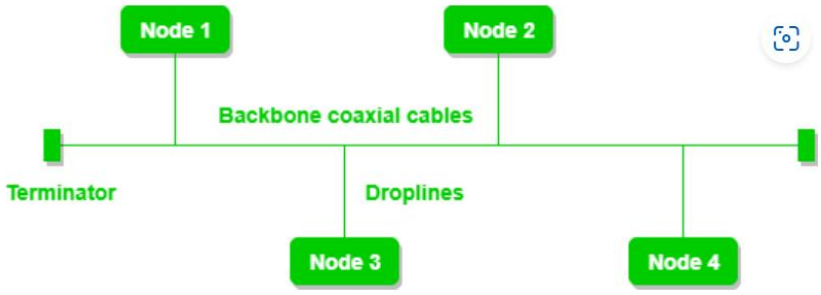
Convert signal for internet access.

18. ROUTER

Directs data b/w Networks, Managing devices on local network.

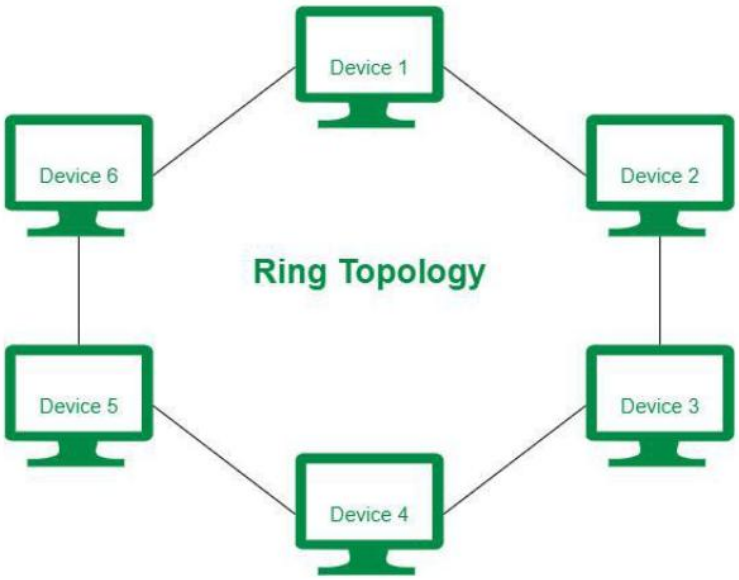
19. NETWORK TOPOLOGIES

- **Bus:**
  - All System connected through a single line



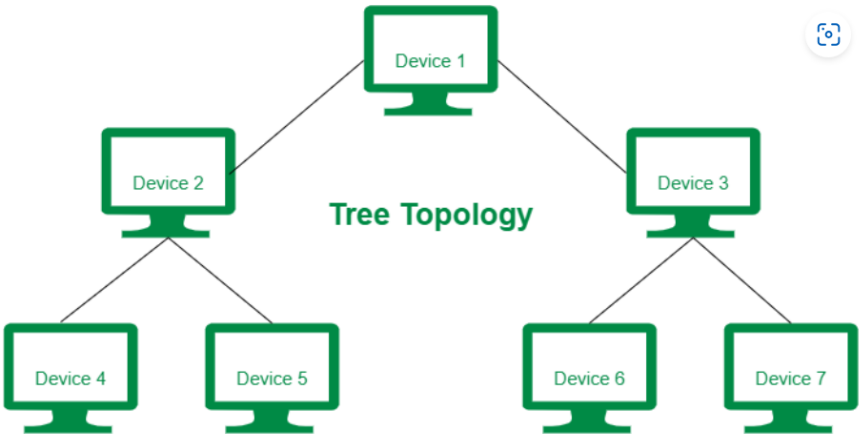
Bus Topology

- **Ring:**
  - Communication done through other devices in an ring structure.



Ring Topology

- **Tree:**
  - Balanced Binary tree: algorithimic complexity reduces to logn.
  - This topology has a hierarchical flow of data.

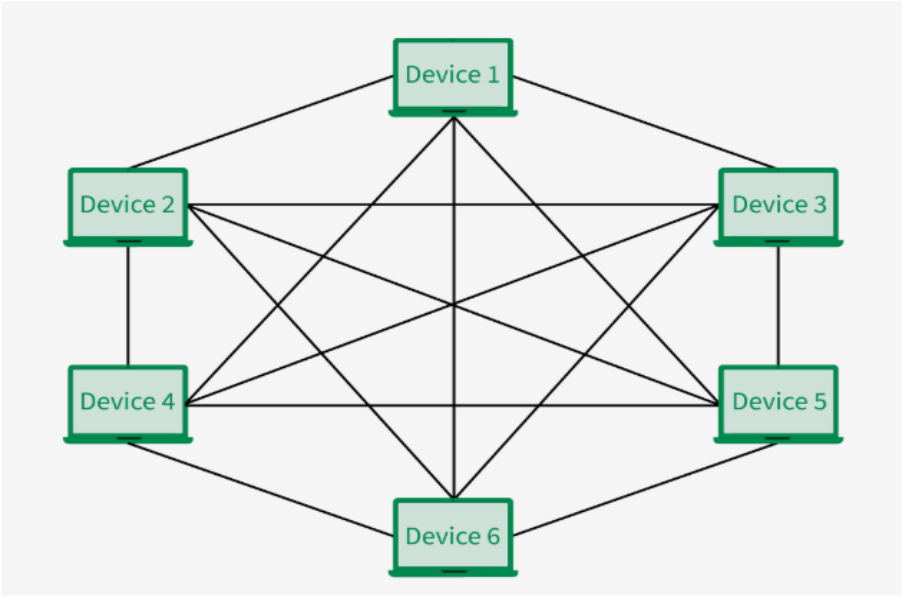


Tree Topology

❖ **Mesh:**

Adv: communication directly

Disadv: security,complicated structure



Mesh Topology

20. PEER TO PEER ARCHITECTURE

❖ Devices communicate directly without central server.
❖ Used in application like file sharing

21. Socket & Port

Socket: IP address +port
Port: Identifies specifies process or services

22. HTTP

- **Methods:**
  - **GET:**Retrieve Data
  - **POST:**Sends Data
  - **PUT:**Update data
  - **DELETE:** Removes Data

Date:9th jan

1. HTTP error/Status Code

HTTP status codes are issued by a web server in response to a client's request made to the server.

They represent the outcome of the request and help diagnose issues when interacting with websites or web services.

200: Series of respond starts from

400: Series of errors starts from

500: Internet Server error

2. Cookies

Small text file of data stored on the user's computer or device by a web browser.

Activity History.

HTTP has no memory. It treats each request as a new one, independent of the previous one.

Since HTTP doesn't remember anything, cookies and **JWT tokens help to maintain "memory" of who you are between requests.**

The browser saves cookies and sends them back to the server with every request.

For example, if you're logged in to a website, the server might send a cookie with your session ID. This session ID is like a "memory" for the server. Every time you make a request, the server checks this cookie to remember that you're logged in.

How do cookies work in HTTP?

- You visit a website for the first time, and the server sends a cookie.
- Your browser stores that cookie. The next time you visit the website, your browser sends the cookie back to the server.
- The server can use the cookie (like a session ID) to recognize that you're the same person who visited earlier.

- JWT is a way for the server to tell the client (your browser) who you are.
- When you log in, the server creates a JWT, which contains information about you (like your username, roles, etc.), and sends it to the client.

Without JWT or Cookies:

- You open a website and make a request. The server doesn't know who you are because HTTP is stateless.
- Every new request looks like it's coming from a completely new user.

JWTs allow users to send a token to the server with each request, so the server can verify who you are without needing to re-log in.

3. VPN

A VPN **(Virtual Private Network)** is a service that helps you create a secure connection between your device (like a phone, laptop, or computer) and the internet.

- private and secured communication.

Types of VPN:

1.Remote Access VPN

What it is: A remote access VPN allows users to connect to a private network (like a company’s internal network) from any location over the internet.

How it works:

The user connects to the VPN server using a secure connection.

Once connected, the user can access resources (files, apps) on the private network, just as if they were physically there.

Use cases: Employees working from home or traveling can securely access their company's network remotely.

2. Site-to-Site VPN (Router-to-Router VPN)

What it is: A site-to-site VPN connects two or more separate networks (such as offices) securely over the internet.

How it works:

Two or more networks (usually in different locations) are connected via a VPN server at each site.

The connection is secure and allows data to flow between them as if they were part of the same local network.

Use cases: Companies with multiple office locations can connect their networks securely, enabling seamless sharing of resources.

3.Cloud VPN:

What It Is: A VPN service that connects your network or device securely to cloud resources (e.g., AWS, Google Cloud).

How It Works: Your device connects to the cloud using an encrypted tunnel to access services hosted in the cloud.

4. Mobile VPN:

What It Is: A VPN designed for mobile devices (smartphones, tablets) that need to maintain a secure connection while switching networks (Wi-Fi to mobile data).

How It Works: The VPN keeps your device connected to a secure network even if it switches from Wi-Fi to cellular data.

5.SSL VPN:

What It Is: A VPN that uses SSL/TLS encryption (the same encryption as HTTPS) for secure communication between a web browser and the server.

How It Works: Users access the VPN through a web browser without needing special VPN software, securing specific services or applications.

6. Double VPN:

What It Is: A VPN configuration that routes traffic through two VPN servers for double encryption and increased anonymity(undefined).

How It Works: Your data is first encrypted by one VPN server, then re-encrypted and passed through a second VPN server.

4. CHECKSUM

error checking

sender 400g receiver 400g not altered

looks no of packets sent and received ...

A checksum is a small value calculated from a larger set of data, used to verify that the data hasn't been corrupted or altered during transmission or storage.

5. IP

IP Building Blocks:

- What it is: Elements used to address and route data across networks.
  - IP Address: A unique identifier for devices (IPv4 or IPv6).
  - Subnet Mask: Divides an IP address into network and host parts.
  - Gateway: Routes data between networks.
  - Router: Directs data packets to their destination.

IP Packet Structure:

- What it is: The format of data sent over an IP network.
  - Header: Contains source/destination IP, protocol, TTL, and checksum.
  - Data (Payload): The actual content being transmitted.

ICMP:

- What it is: A protocol for error reporting and diagnostics (e.g., unreachable destinations).
  - Uses: Includes Ping for checking device connectivity.

Ping:

- What it is: A tool that uses **ICMP** to test if a device is reachable and measures response time.
  - How it works: Sends **ICMP Echo Requests** and waits for a response.

Traceroute:

- What it is: A tool that tracks the **path** of packets to a destination by measuring each hop.
  - How it works: Increases TTL values to map the route and measure latency.

Summary:

- IP: Used for addressing and routing.
- ICMP: Handles error reporting and diagnostics.
- Ping: Tests connectivity.

- **Traceroute**: Traces the route and measures latency.

6.ARP in Detail

What is ARP (Address Resolution Protocol)?

ARP maps an IP address (used in the Network Layer) to a MAC address (used in the Data Link Layer) within a local network. It helps devices find each other and communicate.

How ARP Works:

ARP Request: A device sends a broadcast asking, "Who has this IP address?"

ARP Reply: The device with the matching IP responds with its MAC address.

Communication: The sender uses the MAC address to send data directly.

Why ARP is Necessary:

Local Network Communication: ARP allows devices to translate IP addresses to MAC addresses, enabling communication within the same network.

Bridges IP and MAC: ARP works between the Network Layer (IP) and Data Link Layer (MAC).

ARP Table:

Devices store IP-MAC mappings in an ARP table for quicker future communication.

7.TCP Layers

Application Layer (Layer 7):

What it is: The programs you use (like web browsers, email).

TCP's Role: TCP supports these programs to send data over the network (e.g., browsing websites using HTTP).

Transport Layer (Layer 4) - TCP:

What it is: Ensures the data gets to the right place without errors.

TCP's Role:

Breaks data into smaller parts (segments).

Checks for errors in the data.

Makes sure the data reaches the destination correctly and in the right order.

Network Layer (Layer 3):

What it is: Routes data between different networks (using IP addresses).

TCP's Role: Uses IP to make sure data reaches the correct network.

Data Link Layer (Layer 2):

What it is: Handles data transfer between devices on the same local network.

TCP's Role: Makes sure data can move from one device to another on the same network.

8. IP Addressing and Subnetting

cidr block= 192.168.1.0/24

24 is subnet

192.168.1.0-192.168.1.255=256

NETWORK ID= first IP

BROADCAST ID = last IP

Subnetting:

Subnetting is the process of dividing a larger network into smaller, more manageable sub-networks,

known as subnets. This is done by borrowing bits from the host portion of an IP address to create

additional network bits. Subnetting helps improve network performance, security, and address

management.

9. NAT (Network Address Translation)

Network Address Translation (NAT) is a technique used in networking to modify the source or destination IP addresses in the header of IP packets as they pass through a router or firewall. NAT enables multiple devices on a local network to share a single public IP address for accessing external networks like the internet.

10.SSL/TSL/HTTPS

SSL: SSL is a cryptographic protocol designed to provide secure communication over a computer network.

SSL encrypts the data transmitted between the client and server.

TSL: TLS is the successor to SSL, designed to improve upon SSL's weaknesses and provide stronger security for internet communications.

HTTPS: HTTPS is a secure version of the HTTP protocol, which is used for transferring data over the web.

11. Symmetric and Asymmetric Encryption

Symmetric: In symmetric encryption, the same key is used for both encryption and decryption.

Asymmetric: In asymmetric encryption, two different keys are used: a public key and a private key.

The public key is used to encrypt data, while the private key is used to decrypt it.

12. Monolithic Architecture

Monolithic architecture refers to a traditional model of software design where an entire application is built as a single, unified unit.

13. Microservices Architecture

The key idea behind microservices is to break down a monolithic application into smaller, modular pieces, each responsible for a specific aspect of the business logic or functionality.

//Authentication is the process of verifying the identity of a user, system, or entity attempting to access a resource. It answers the question: "Who are you?"

// Authorization is the process of granting or denying access to a specific resource or action based on the identity that was authenticated. It answers the question: "What can you do?

Date:10th Jan

1. Firewall

A **firewall** is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. It acts like a barrier between your network and the outside world, allowing safe traffic and blocking potentially harmful traffic.

- **Ingress**: Incoming traffic. Firewalls inspect data coming into your network from external sources.(e.g., someone trying to connect to your server)

- **Egress**: Outgoing traffic. Firewalls also monitor and control the data leaving your network. (e.g., you browsing a website).

  Example:

- Your home Wi-Fi router has a built-in firewall that prevents external devices (hackers) from directly accessing your devices (like your computer or smartphone). Only devices with allowed IP addresses can communicate with your network.

- **Stateful Firewall**: A stateful firewall maintains a table of the state of each active connection, so it can track and validate whether a response is from an established connection. This helps ensure that only legitimate return traffic is allowed.

Example of stateful:

- You are browsing a website. Your computer sends a request to the website server (e.g., `http://example.com`). The stateful firewall records this request (the connection state) and allows the response from the server back to your computer.
- Now, if a packet comes from the server **without a prior request** from your computer, the stateful firewall would block it because it doesn't match the state of an existing, valid connection.

- **Stateless Firewall**: A stateless firewall does not keep track of connection states. It filters packets based solely on pre-configured rules (such as source/destination IP address or port).

Example:

- Suppose you're sending a packet to a server over port `80` (HTTP). The stateless firewall simply looks at the packet's source and destination IP and port numbers and decides whether to allow it based on the rules.
- If a packet arrives on port `80` from a source IP that is not allowed by the firewall, it will be dropped, even if the packet is a response to a valid request (since it doesn't track the connection).

---

2. API Gateway-Middleman

An **API Gateway** is a server that acts as an entry point for all requests from clients to access your microservices or backend systems. It provides several essential functions, such as routing, authentication, security, rate limiting, and response formatting.

- **Gate for Security**: It ensures that only authorized users or systems can access the backend services by handling things like authentication and authorization.
- **Microservices Not Directly Contacted**: Microservices are isolated, and instead of talking directly to each other, they communicate through the API Gateway.
- **Restricted Access**: The gateway controls access, similar to how you might need permission from a teacher (the gatekeeper) to talk to another student in class.

Example:

- Imagine a company has three different services: a user authentication service, a payment service, and a product catalog service. The API Gateway handles incoming requests and directs them to the appropriate service without exposing them directly.

3.NAT (Network Address Translation)

- **NAT** translates private IP addresses to public IP addresses when devices on a private network need to communicate with the internet.
- **Port Numbers in Table**: NAT also keeps track of port numbers so that incoming responses can be directed back to the correct private device.

Example:

- You have multiple devices at home (e.g., laptop, smartphone), each with a private IP (e.g., `192.168.1.2`, `192.168.1.3`). When these devices send data to the internet, the router uses **NAT** to convert the private IP to a single public IP address (`203.0.113.10`).

1. Static NAT (Static Network Address Translation)

What it is: Static NAT is when a private IP address (inside your network) is permanently mapped to a specific public IP address (that the outside world can access). This means, every time a specific device inside your network sends or receives data, it always uses the same public IP.

- **Static IP Address**: This is an IP address that does not change. It is "static" because it remains constant and is typically assigned to a server or device that needs a constant identifier (like a website server).

Example: Imagine you have a computer at home with a private IP address `192.168.1.2`. You also have a router with a public IP address `203.0.113.5`.

- With **Static NAT**, every time the computer (`192.168.1.2`) wants to communicate with the internet, the router will always replace its private IP (`192.168.1.2`) with the public IP (`203.0.113.5`).
- Similarly, when someone on the internet wants to communicate with `203.0.113.5`, the router will forward the request to `192.168.1.2` (the computer in your house).

Key Point: The private IP `192.168.1.2` always maps to the same public IP `203.0.113.5`.

---

2. Dynamic NAT (Dynamic Network Address Translation)

What it is: Dynamic NAT is when a private IP address is mapped to a **temporary** public IP address from a pool of available public IP addresses. This mapping is dynamic and can change over time. It's not fixed like Static NAT.

Example: Let's say your home network has a private IP range `192.168.1.0/24`, meaning your devices are assigned IPs like `192.168.1.2`, `192.168.1.3`, etc.

- You have **three** devices inside your network (e.g., computer, smartphone, and laptop).
- Your router has a pool of **two** public IPs (`203.0.113.5` and `203.0.113.6`).

Now, when these devices access the internet:

- **Device 1 (192.168.1.2)** might be assigned the public IP `203.0.113.5`.
- **Device 2 (192.168.1.3)** might be assigned the public IP `203.0.113.6`.

The important thing here is:

- **The public IP address is assigned dynamically** and might change over time.
- **When one of the devices** no longer needs the public IP (e.g., finishes browsing), the router can **assign the public IP to another device** in the future.

Key Point: The mapping is temporary and changes. The router assigns a public IP from a pool of available public IPs.

-

---

4. PAT (Personal Access Token)

It acts as a substitute for your regular username and password but is more secure because it is limited in scope (what it can do) and time (how long it is valid).

PAT is a type of authentication token used to securely access APIs or systems without using a password. It's commonly used in cloud services and development environments to ensure that the request is coming from an authorized user or system.

- **API Communication**: When a user or system interacts with an API, they can generate a PAT, which is sent along with requests to confirm identity and authority to access specific resources.

How Does a PAT Work?

- Instead of using your main password to log in or access a service, you generate a **token** (a long string of characters) from the service (e.g., GitHub).
- This token is sent with your requests to prove that it's really you (authentication).
- The token can give access to certain parts of the service, and it can be limited to specific actions (like only accessing repositories, but not changing account settings).

Simple Example:

Let's use GitHub as an example to explain PAT:

1. **You want to access a private GitHub repository** (which is only available to certain users and not the public). However, GitHub doesn't want you to just use your regular password directly for this kind of request (because it's not secure enough).
2. **Generating a PAT**:
    - You log in to GitHub and create a **Personal Access Token (PAT)**.
    - This PAT is like a special password that GitHub gives you, which you can use instead of your regular password for this specific task (accessing a private repo).
3. **Using the PAT**:
    - Now, you want to access the private GitHub repository through an **API**. APIs allow applications to communicate with each other.
    - When you make a request to the GitHub API (for example, to fetch some data from your private repository), you include the **PAT** in the request headers. This is like telling GitHub, "Hey, this request is coming from me, and I have permission to do this."
4. **Why Use a PAT**:
    - **Security**: If someone gets hold of your PAT, they can only access the parts of GitHub that the PAT allows. It's more secure than giving out your password.
    - **Limited Scope**: The PAT can be set up to allow access to only certain parts of GitHub (like repositories), and can be limited in time (expire after a few hours, for example).

5. Microservices

In a **microservices architecture**, the application is broken into smaller, independently deployable services that communicate via APIs.

Example:

- An online store could have different microservices for user accounts, inventory, orders, and payments. Each of these services is independent, so they can be scaled and maintained separately.

- **Monolithic vs Microservices**: A **monolithic** application would have everything built into one big application, which becomes harder to scale as it grows. In **microservices**, each part of the app (like user authentication or payment) can scale independently, such as adding more instances of the payment service during a sale

6. CIDR and Subnetting

CIDR (Classless Inter-Domain Routing) is a flexible way of allocating and managing IP addresses. **Subnetting** divides an IP network into smaller sub-networks.

Example:

- Suppose you have the IP address `10.0.0.0/24`, meaning the first 24 bits represent the network, and the remaining 8 bits represent the hosts. This gives you 256 addresses for hosts (e.g., `10.0.0.1`, `10.0.0.2`, etc.).

---

7. Symmetric and Asymmetric Encryption

- **Symmetric Encryption** uses the same key for encryption and decryption. **Asymmetric Encryption** uses a pair of keys: a public key to encrypt and a private key to decrypt.

Example:

- **Symmetric Encryption**: You and your friend both have the same key (like a password) to encrypt and decrypt messages.
- **Asymmetric Encryption**: You use a **public key** to encrypt a message, and only the **private key** (kept secret) can decrypt it.

---

8. VPC (Virtual Private Cloud)

A **VPC** is a private, isolated network in the cloud where you can launch resources like virtual machines, storage, and databases, separated from others.

Example:

- You create a VPC with the IP range `10.0.0.0/16`. Inside the VPC, you can create subnets for different departments: `10.0.1.0/24` for HR, `10.0.2.0/24` for Finance, and so on. These subnets are isolated from each other but can communicate with each other via the VPC.

---

9. Asynchronous Queues and Subscribers

In an **asynchronous system**, tasks are added to a queue and processed later, allowing the main program to continue running without waiting.

Example:

- A user uploads a large file to a web service. Instead of waiting for the file to be processed immediately, the file is placed in a queue for later processing. Meanwhile, the user can continue using the service.
- **Subscriber**: A separate service (the subscriber) listens to the queue and processes the file when it's ready.

---

10. Server Farm

A **server farm** is a large collection of servers in one location, designed to process a large amount of data.

Example:

- Amazon Web Services (AWS) has server farms, or data centers, around the world to support their cloud services. Each data center contains thousands of servers that handle tasks like running websites, storing data, and processing requests.

---

12. IPSec

IPSec is a protocol used to secure network communications by encrypting and authenticating IP packets.

Example:

- When two companies want to establish a secure connection between their offices, they can use IPSec to encrypt all the data sent between them. This ensures that even if someone intercepts the data, they can't read it.

---

13. Threat, Vulnerability, Risk

- **Threat**: An event or action that could potentially cause harm (e.g., a hacker trying to break into a system).
- **Vulnerability**: A weakness in the system that could be exploited (e.g., outdated software that lacks security patches).
- **Risk**: The chance that a threat will exploit a vulnerability and cause damage (e.g., the likelihood of a hacker exploiting the vulnerability to steal sensitive data).

Example:

- **Threat**: A hacker trying to steal data.
- **Vulnerability**: Your website has an outdated plugin with security flaws.
- **Risk**: High risk that the hacker will exploit the vulnerability to steal customer data.

---

14. Reverse Proxy

A **reverse proxy** forwards client requests to appropriate backend servers. It acts as an intermediary between the client and servers.

Example:

- A company has multiple backend services (e.g., one for customer service, one for billing). A reverse proxy sits in front of these services, so clients don't directly interact with them. Instead, the reverse proxy forwards requests to the right service.
- **SSL Termination**: A reverse proxy can handle the encryption/decryption (SSL/TLS) for web traffic, freeing up backend servers from this work.

---

15. IPv4 vs IPv6

- **IPv4**: Uses 32 bits for addresses (e.g., `192.168.1.1`), and has a limited number of addresses.
- **IPv6**: Uses 128 bits for addresses (e.g., `2001:0db8:85a3:0000:0000:8a2e:0370:7334`), offering a much larger address space.

Example:

- With **IPv4**, we have about 4 billion unique IP addresses. But with **IPv6**, the number of addresses is so large that it can support billions of devices worldwide, such as the growing number of IoT devices.

---

16. 3-Way Handshake

The **3-way handshake** is how a TCP connection is established between a client and a server.

Example:

1. **SYN**: The client sends a request to the server to start a connection.
2. **SYN-ACK**: The server responds with an acknowledgment, saying it's ready to accept data.
3. **ACK**: The client confirms the server's response, and the connection is established.

---

17. IEEE 802.11

This is the standard for **Wi-Fi** (wireless networking).

Example:

- If you have a wireless router at home, it uses **IEEE 802.11** to allow devices like laptops and phones to connect wirelessly to the internet.

---

18. How Wi-Fi Works

Wi-Fi allows devices to connect to the internet wirelessly using radio waves.

Example:

- Your smartphone connects to a **wireless router** at home. The router is connected to the internet through a **modem**. The router uses Wi-Fi (radio waves) to transmit data to your phone.

---

19. Richardson Maturity Model

The **Richardson Maturity Model** defines the stages of RESTful API design, from simple CRUD (Create, Read, Update, Delete) operations to more complex, hypermedia-driven interactions.

Example:

- **Level 1**: Your API exposes basic endpoints like `POST /user`, `GET /user/{id}`.
- **Level 2**: Your API adds more RESTful principles, like using HTTP methods properly (e.g., `GET`, `POST`, `PUT`, `DELETE`).
- **Level 3**: Your API implements **HATEOAS** (Hypermedia As The Engine of Application State), where the response includes links to other related resources.

---

20. 12-Factor App Principles

The **12-Factor App** principles are a set of guidelines for building scalable, maintainable applications, particularly for the cloud.

Example:

- **Codebase**: Store code in a version control system like Git.
- **Dependencies**: Manage external dependencies (like libraries) using package managers (e.g., npm for JavaScript).

- **Port Binding**: The app should expose its services on a specific port, not rely on the environment.

---

21. How IPSec Works

IPSec provides encryption and authentication for IP packets, ensuring secure communication over an IP network.

Example:

- Two companies want to securely exchange data over the internet. They use **IPSec** to encrypt the data and ensure only the intended recipient can decrypt it using a shared key.

---

22. Vulnerability, Threat, Risk

As explained in **Point 13**.

---

23. Layered Security

Layered security involves using multiple security measures at different levels to protect a system.

Example:

- **Network Level**: Use firewalls to block unauthorized traffic.
- **Host Level**: Use antivirus software to detect malware.
- **Application Level**: Use encryption to protect data in transit.

---

24. TCP vs UDP

- **TCP** is reliable, connection-oriented (ensures data arrives in order).
- **UDP** is faster but unreliable (data may be lost or out of order).

Example:

- **TCP**: When you browse a website, TCP ensures the webpage is fully loaded and in order.
- **UDP**: When streaming a video, UDP is used because speed is more important than ensuring every packet arrives.

---

25. Routers vs Switches

- **Router**: Connects different networks (e.g., your home network to the internet).
- **Switch**: Connects devices within the same network (e.g., connecting your computer to a printer).

---

26. DNS (Domain Name System)

DNS translates domain names (e.g., `www.example.com`) into IP addresses.

Example:

- You type `www.google.com` in your browser, and **DNS** translates it to the IP address of Google's server (`172.217.16.196`).

Date:11 th jan

1. Data centre:

A **Data Center** is a physical facility that houses a large number of **computers, servers, storage devices, and networking equipment**.
In simpler terms, it's like a "big building" where a company's digital "brain" (computers, databases, etc.) is located, allowing users or businesses to access their data, applications, or websites.

➢ Creating Virtual Machines (VMs) from One Box:

- **Virtualization** allows you to run multiple **virtual machines (VMs)** on a single physical machine or server. Each VM behaves like a separate computer with its own OS and resources (CPU, RAM, etc.), even though they're all running on the same physical hardware.
- This is achieved through **hypervisors**, such as **VMware**, **Hyper-V**, or **KVM**, which allocate resources from the physical machine to each virtual machine.
- **Different Linux Distributions:**
- **Linux distributions** (distros) are various versions of the Linux operating system tailored for different use cases or preferences. Some common families of Linux distros include:
  - **Ubuntu family**: Based on **Debian**, Ubuntu is user-friendly and often used in desktops and servers.
  - **Red Hat family**: Includes distributions like **Red Hat Enterprise Linux (RHEL)**, **CentOS**, and **Fedora**, which are used for enterprise servers and systems.
  - **Debian family**: Includes **Debian**, **Ubuntu**, and others.
  - **Oracle Linux**: A distribution of Linux developed by Oracle, often used in enterprise environments, especially with Oracle software.
- **APT and APT-get in Ubuntu:**

- **APT (Advanced Package Tool)** and **APT-get** are tools used in **Ubuntu** (and other Debian-based distributions) to manage software packages. APT helps you install, upgrade, and remove software from your system.
  - For example, to install a package, you would run:
    - bash
    - Copy code
    - sudo apt-get install <package-name>
  - APT automatically handles dependencies, ensuring that the required libraries or packages are also installed.
- ## Snapshots of VMs:
- **VM Snapshots** are used to capture the **state** of a virtual machine at a specific point in time. This allows you to revert the VM back to this exact state if needed.
  - **Why?**: Snapshots are used to preserve the VM's state so that you can:
    - **Restore** the machine to a previous configuration if something goes wrong.
    - **Test** changes without affecting the running system.
  - **Persistence**: Data stored in the **hard disk** is persistent, meaning it remains intact even after a reboot. However, data stored in **RAM** (memory) is lost after a reboot, so snapshots help capture and persist the VM's state.
- ## Program Data Goes to RAM:
- **RAM (Random Access Memory)** is temporary storage that holds the data needed by running programs. Once the program stops running, the data in RAM is lost.
- **Stack**: Used for storing local variables and function calls. The stack has a predefined size and is used for primitive data types (integers, chars, etc.).
- **Heap**: Used for dynamic memory allocation. Objects or data structures are stored here, and the memory is managed via pointers.
- ## String Pool in Java:
- In **Java**, a **String Pool** is a special storage area in memory where **strings** are stored. When you create a string, Java checks if the string already exists in the pool. If it does, Java reuses that string; if not, it adds the string to the pool.
- This helps reduce memory usage since multiple identical string objects are not created.
- ## Serialization:
- **Serialization** is the process of converting an object's **state** into a format (like a byte stream) that can be stored or transmitted. This is useful for saving an object's state (e.g., in a file or database) and later restoring it.
- **Example**: Saving an object’s state to a file, and then reading it back from the file later.
- ## VM Instance State Stored as Snapshot:
- When you **take a snapshot** of a VM, you capture its **state**: the running processes, memory, and configuration.
- This allows you to **restore** the VM to that exact state later, making it useful for backups, testing, or recovery scenarios.
- ## Persistence in VM:
- After you **install** software (like **Java**) on a VM, the changes are saved to the VM’s **hard disk**. So, even after a **reboot**, Java will still be installed because its data is persistent in the hard disk.
- ## Encrypted VM State:
- **Encryption** is often used to protect sensitive data. If a VM's state (or snapshot) is encrypted, it ensures that unauthorized users cannot view or modify the VM's configuration or data.
- ## Scaling VMs:

- **Scale factors** refer to the ability to **resize** a VM:
  - **Increase Size**: Allocate more resources (e.g., CPU, RAM, storage) to the VM.
  - **Decrease Size**: Reduce resources allocated to the VM to save costs or optimize performance.
  - This is useful when your application needs more resources (scale up) or when resources are not fully utilized (scale down).
- **DHCP Server:**
- **DHCP (Dynamic Host Configuration Protocol)** is used to **automatically assign IP addresses** to devices on a network.
- **How it works**: A **DHCP server** manages a **pool** of IP addresses and allocates them to devices (like VMs, computers, or smartphones) as they join the network.
- **Example**:
  - When a VM starts, it sends a **DHCP request** for an IP address.
  - The **DHCP server** responds with an **available IP address** from its pool, and the VM can use this IP to communicate on the network.
- ---
- **Summary:**
- **Virtual Machines (VMs)** allow you to run multiple OS instances on one physical machine, with each VM behaving like a separate computer.
- **Linux distributions** come in different families, like **Ubuntu** (Debian-based) and **Red Hat** (RHEL-based), with different tools (e.g., **APT**) for managing software.
- **VM Snapshots** save the entire state of a VM, allowing you to restore it to a previous configuration.
- **RAM** holds temporary data, and **hard disk** provides permanent storage for program data.
- **DHCP servers** automatically assign IP addresses to devices, making network management easier.

- **In Aws vm called as EC2 instance**
- **All services using vm**
- **Vpc doesn't belong to vm**
- **subnet**

Key Components of a Data Center:

1. **Servers**: These are the computers that process and store data.
2. **Storage Devices**: These devices hold the data, like hard drives or solid-state drives (SSDs).
3. **Networking Equipment**: Devices like routers, switches, and firewalls that manage data traffic between servers and external networks.
4. **Cooling Systems**: To keep all the equipment from overheating, data centers are equipped with powerful cooling systems.
5. **Power Supply**: Data centers have backup generators and uninterruptible power supplies (UPS) to ensure continuous power in case of outages.
6. **Security system**

TYPES OF DATA CENTERS

1. On-premise Data Centers:

- **What it is**: A data center **owned and managed by your company** on your premises (in your office or building).
- **Example**: A company has its own servers and equipment in a server room within its office.
- **Pros**: Full control over your infrastructure.
- **Cons**: High costs and maintenance.

2. Colocation Data Centers:

- **What it is**: You **rent space** in a data center and place your own equipment there. The provider takes care of power, cooling, and security.
- **Example**: A company rents space in a facility and uses its own servers and equipment.
- **Pros**: No need to build a data center yourself, but you still control the hardware.
- **Cons**: You still manage your own equipment.

3. Cloud Data Centers:

- **What it is**: These are **remote data centers managed by cloud providers** (like AWS, Microsoft Azure, Google Cloud). You rent resources like storage, servers, and apps over the internet.
- **Example**: Using cloud services like **Amazon Web Services (AWS)** or **Google Cloud** to store data or run applications without owning hardware.
- **Types**:
    - **IaaS**: Rent virtual machines and infrastructure.
    - **PaaS**: Rent a platform for building apps without managing servers.
    - **SaaS**: Use software like **Gmail** or **Dropbox** without managing anything.

Tomcat:

- **What it is**: A server used to run **Java-based web applications**.
- **Example**: If you're running a Java-based website, Tomcat is used to host and serve the site.

Design Considerations:

1. Reliability:

- **What it is**: Ensures the data center is always available and operates without interruptions.
- **Key Factors**:
    - **Fault tolerance**: The ability to keep working even if something fails (e.g., backup systems or redundant parts).
    - **Backup and Redundancy**: Having backup systems like power sources (generators, UPS) and additional hardware to prevent downtime.

2. Security:

- **What it is**: Protecting the data center from unauthorized access and cyber threats.
- **Key Factors**:
    - **Physical security**: Locks, cameras, and guards to protect the physical premises.
    - **Cybersecurity**: Firewalls, encryption, and intrusion detection to safeguard data and systems from hackers.

3. Scalability:

- **What it is**: The ability to grow and handle increased demand.
- **Key Factors**:
    - As your company or services grow, the data center should be able to expand (add more servers, storage, etc.) without issues.

4. Energy Efficiency:

- **What it is**: Reducing energy consumption and costs.
- **Key Factors**:
    - Using **energy-efficient hardware**, **cooling systems**, and **power-saving techniques** helps lower costs and reduce environmental impact.

5. Disaster Recovery:

- **What it is**: Plans and systems to recover quickly after a disaster (like a fire, flood, or power failure).
- **Key Factors**:
    - **Backups**: Keeping copies of important data off-site.
    - **Disaster recovery plans**: Having processes to restore operations and minimize downtime in case of an emergency.

Summary:

- **Reliability**: Ensures uptime with backups and redundancy.
- **Security**: Protects against physical and cyber threats.
- **Scalability**: Allows growth as your needs increase.
- **Energy Efficiency**: Reduces energy consumption and costs.
- **Disaster Recovery**: Prepares for emergencies to minimize downtime.

These factors are crucial for ensuring that the data center can operate smoothly, securely, and efficiently.

Types of Data Centers:

1. **On-premise Data Centers**:
    - **Owned and Operated**: The company owns and manages the data center.
    - **Complete Control**: The company has full control over the hardware, software, and security.

- **High Initial Investment**: Setting up an on-premise data center requires a significant upfront cost for equipment and infrastructure.
- **Example**: A large corporation builds its own data center to store its servers and sensitive data.

Why Load Balancing is Important?

- **Ensure Fault Tolerance**: Load balancing helps distribute traffic to multiple servers, ensuring that if one server fails, others can take over, ensuring uninterrupted service.
- **Optimize Resource Usage**: By balancing the load, it ensures no single server gets overwhelmed, which can improve performance and reduce downtime.

Common Load Balancing Techniques:

1. **Round Robin**: Distributes requests to servers one by one in a circular order. Each server gets a request in turn.
2. **Least Connections**: Routes traffic to the server with the least active connections, ensuring servers are not overloaded.
3. **Least Response Time**: Directs requests to the server with the fastest response time, improving performance.
4. **Source IP Hashing**: Uses the IP address of the client to determine which server will handle the request, ensuring that requests from the same client are consistently directed to the same server.
5. **Weighted Round Robin**: Distributes traffic based on the server's weight or capability. More powerful servers get more traffic.

Types of Load Balancers:

1. **Hardware Load Balancers**:
   - **High Upfront Cost**: Requires purchasing physical devices (hardware).
   - **Example**: A company buys a dedicated hardware appliance that handles all incoming network traffic and distributes it to the servers.
2. **Software Load Balancers**:
   - **Application Layer**: Operates at the application layer (Layer 7) and can handle specific traffic based on the type of request.
   - **Performance**: Generally not as reliable or fast as hardware-based load balancers.
   - **Example**: Software like NGINX or HAProxy used for load balancing requests to web servers.

Benefits of Load Balancers:

1. **High Availability (HA)**: Ensures that the application remains available even if one or more servers fail by automatically redirecting traffic to healthy servers.

Key Components of High Availability (HA):

- **Redundancy**: Extra servers, power supplies, and components to avoid single points of failure.
- **Failover**: Automatic switching to backup systems when the primary system fails.
- **Health Monitoring**: Constant monitoring of systems to detect failures early and reroute traffic before it causes a problem.
- **Load Balancing**: Distributing network traffic to multiple servers to avoid overloading any single server and ensure a smooth user experience.

Summary:

- **Load Balancing** is crucial for ensuring fault tolerance, improving performance, and managing traffic to multiple servers.
- **Types of Load Balancers**: Hardware (expensive) vs. Software (more flexible but less reliable).
- **Key Components of HA**: Redundancy, failover, health monitoring, and load balancing ensure high availability and minimal downtime.

Storage Types

• **Direct Attached Storage (DAS)**:

* DAS refers to storage devices that are directly attached to a single computer or server, without being connected to a network. It is typically used for personal or small-scale applications.

* How it works: DAS storage is directly connected via interfaces such as USB, SATA, SAS, or Thunderbolt.

* Local storage directly attached to servers.

*Lack of scalability

*Potential to lost data

*Device specific

* Examples: External hard drives, internal hard drives, SSDs (Solid-State Drives).

• **Network Attached Storage (NAS)**:

*NAS is a storage device connected to a network, allowing multiple users and devices to access data over the network. It's often used for centralized file storage and sharing.

*How it works: NAS devices typically use Ethernet or Wi-Fi to connect to a local area network (LAN), and they present storage over protocols like SMB/CIFS (Windows), NFS (Linux/Unix), or AFP (Apple).

*File-based storage accessible over a network.

*Moderate scalability and perfomance

*Examples: Synology NAS, QNAP NAS, WD My Cloud.

• **Storage Area Networks (SAN)**:

*SAN is a high-speed network that connects storage devices (such as disk arrays) with servers, enabling block-level data access. Unlike NAS, which provides file-level access, SAN provides block-level access to data, typically used for large-scale enterprise applications.

*How it works: SANs often use Fibre Channel, iSCSI, or FCoE (Fibre Channel over Ethernet) to connect storage devices and servers. Data is accessed as blocks rather than files.

*High-speed network of storage devices, often used for large-scale enterprise data storage.

*Examples: EMC VMAX, NetApp FAS, Dell PowerMax

TYPES OF STORAGES

PRIMARY STORAGE

Primary storage refers to the storage that is directly accessible by the CPU and is used to store data and instructions that are actively being processed.

1. **RAM (Random access memory)**

   Description: RAM is the most common type of primary storage and is used to store data and instructions that the CPU needs to access quickly while performing tasks.

   Characteristics:

- Fast read and write access.
- Volatile memory (data is lost when power is turned off).
- Temporarily stores data being processed by running applications.

1. **ROM(Read only memory)**

   Definition: ROM is a type of non-volatile memory used in computers and other electronic devices to store permanent data or instructions that are not meant to be altered or modified during normal operation.

   Characteristics:

- Non volatile
- Read only
- Slower access

SECONDARY STORAGE

1. **HDD (Hard Disk Drives)**
2. **SSD (Solid-State Drives)**

Description: HDDs are mechanical storage devices that use spinning disks to read/write data. They are widely used for long-term storage.

Characteristics:

1. Larger capacity compared to primary storage.
2. Slower read/write speeds due to mechanical components.
3. Non-volatile (data persists even without power).

Description: SSDs are storage devices that use flash memory to store data, providing faster read/write speeds compared to HDDs.

Characteristics:

1. Faster than HDDs but generally more expensive.
2. Larger capacity compared to primary storage.
3. Non-volatile, retains data without power.

OPTICAL DISC

- An **optical disc** is a storage medium that uses laser light to read and write data on a reflective surface.
- Optical discs are widely used for storing data such as software, music, videos, and backups.
- Example : CD, DVD, Floppy Disc

RAID LEVELS

RAID 0 (Striping)

- **Configuration**: Data is split into blocks and distributed across multiple disks (at least 2).
- **Redundancy**: No redundancy—if one drive fails, all data is lost.
- **Performance**: High performance, as data is read and written in parallel to multiple drives.
- **Capacity**: Total capacity is the sum of the capacities of all disks.
- **Use Case**: Suitable for applications requiring high performance and where data loss is not critical (e.g., temporary data, non-essential files).

RAID 1 (Mirroring)

- **Configuration**: Data is duplicated (mirrored) across two or more disks.
- **Redundancy**: High redundancy—if one drive fails, the data is still available from the other drive(s).
- **Performance**: Good read performance (because the system can read from multiple disks), but write performance is similar to a single disk.
- **Capacity**: Total capacity is the size of one drive (since data is duplicated).
- **Use Case**: Suitable for situations where data integrity is critical and write performance is not as important (e.g., personal computers, critical data storage).

RAID 5 (Striping with Parity)

- **Configuration**: Data is striped across multiple disks (at least 3), with parity information distributed across all disks.
- **Redundancy**: Moderate redundancy—if one disk fails, the data can be rebuilt using the parity data from the remaining disks.
- **Performance**: Good read performance, but write performance is slower compared to RAID 0 and RAID 1 due to the overhead of parity calculations.
- **Capacity**: Total capacity is the sum of all disks minus one (because one disk is used for parity).
- **Use Case**: Suitable for applications that require a balance of redundancy, performance, and storage capacity (e.g., file servers, databases).

RAID 6 (Striping with Double Parity)

- **Configuration**: Similar to RAID 5 but with **two sets of parity data**, which are stored across different disks (requires at least 4 disks).
- **Redundancy**: High redundancy—can tolerate the failure of **two disks** simultaneously without data loss.
- **Performance**: Read performance is good, but write performance is slower than RAID 5 because of double parity calculations.
- **Capacity**: Total capacity is the sum of all disks minus two (because two disks are used for parity).
- **Use Case**: Suitable for environments where data protection is more important than write performance (e.g., critical business data storage).

RAID 10 (RAID 1+0)

- **Configuration**: Combines the features of RAID 1 and RAID 0. Data is mirrored (RAID 1) and then striped (RAID 0).
- **Redundancy**: High redundancy—can tolerate the failure of one disk per mirrored pair.
- **Performance**: High performance for both read and write operations, as data is striped (RAID 0) and mirrored (RAID 1).
- **Capacity**: Total capacity is the sum of half of the disks (since data is mirrored).
- **Use Case**: Suitable for applications that require both high performance and redundancy (e.g., databases, high-performance servers).

BACKUP AND RECOVERY

A **backup** is the process of creating a duplicate copy of data that can be restored in case the original data is lost, corrupted, or inaccessible.

TYPES OF BACKUP

Full Backup

- **Definition**: A full backup is a complete copy of all selected data. It copies everything, including all files, folders, and system data (depending on the configuration).
- **How It Works**: Every time a full backup is performed, all data is backed up in its entirety, regardless of whether it has changed since the last backup.

Incremental Backup

- **Definition**: An incremental backup only backs up the data that has changed since the **last backup** (whether it was a full backup or the most recent incremental backup).
- **How It Works**: After an initial full backup, subsequent incremental backups only capture changes made to files since the last backup. This can be multiple times over a period.

Differential Backup

- **Definition**: A differential backup captures all the changes made since the last **full backup**. Unlike incremental backups, differential backups do not rely on previous differential backups, but only on the full backup.

Mirror Backup

- **Definition**: A mirror backup creates an exact copy (or "mirror") of the selected data. It is similar to a full backup but continuously synchronizes data between the source and the backup location.

3-2-1 BACKUP STRATEGY

The **3-2-1 backup strategy** is a widely recommended method for ensuring robust data protection and recovery. It helps mitigate the risks of data loss from various types of disasters (e.g., hardware failure, cyberattacks, accidental deletions). This strategy involves creating

multiple copies of data and storing them in different locations to increase redundancy and resilience.2 Copies stores in two different media types and one in offsite.

BASIC SERVER COMPONENTS

- **MOTHER BOARD:** A **motherboard** is the central printed circuit board (PCB) in a computer that connects and allows communication between various hardware components. It serves as the backbone of the computer, providing essential connections for components like the **CPU, RAM**
- **CPU**
- **RAM**
- **NIC**
- **STORAGE DRIVE**

LOAD BALANCING

- **ROUND ROBIN: Round Robin** is one of the simplest and most commonly used load balancing algorithms. It is a method used to distribute client requests (or traffic) across a group of servers or resources in a circular order.
- **LEAST CONNECTION: Least Connections** is a dynamic load balancing algorithm that directs incoming traffic to the server with the **fewest active connections** at the time of the request. This method is designed to distribute load based on the number of active connections each server is currently handling, aiming to prevent overloading any single server.
- **LEAST RESPONSE TIME: Least Response Time** is a dynamic load balancing algorithm that routes incoming client requests to the server with the **quickest response time** at the moment of the request. The goal of this algorithm is to optimize user experience by sending traffic to the server that is not only least loaded but also currently capable of processing requests the fastest.
- **SOURCE IP HASHING: Source IP Hashing** is a load balancing algorithm that uses the **client's IP address** to determine which server in the pool should handle a particular request. The key idea behind this approach is to ensure that requests from the same client IP address are always directed to the same backend server, creating session persistence (also called sticky sessions).
- **WEIGHTED ROUND ROBIN: Weighted Round Robin (WRR)** is an enhancement of the traditional **Round Robin** load balancing algorithm. In **Weighted Round Robin**, each server in the pool is assigned a **weight** that reflects its capacity or performance. The load balancer distributes incoming requests across the servers, but it gives more requests to servers with higher weights, effectively allowing more powerful servers to handle more traffic.

TYPES OF LOAD BALANCER

HARDWARE LOAD BALANCER

Hardware load balancer is a physical appliance designed specifically for load balancing tasks. It is a dedicated device with specialized hardware and software to handle traffic distribution efficiently.

SOFTWARE LOAD BALANCER

A software load balancer is a software application that runs on general-purpose hardware (such as a server or virtual machine) to perform load balancing tasks. It uses algorithms and protocols to distribute traffic among multiple servers.

NETWORK SECURITY KEY CONCEPTS

CONFIDENTIALITY

Confidentiality is one of the core principles of **information security**, ensuring that sensitive information is only accessible to those authorized to view it. It involves preventing unauthorized individuals, organizations, or systems from accessing data, ensuring that private and personal information is protected from exposure or misuse.

INTEGRITY

Integrity refers to the concept of maintaining and assuring the accuracy, consistency, and trustworthiness of data throughout its lifecycle. It ensures that data is not altered, tampered with, or corrupted—either accidentally or maliciously—while it is being stored, transmitted, or processed.

AVAILABILITY

Availability refers to the concept of ensuring that information, systems, and services are accessible and usable by authorized individuals when needed.

> **Network security strategy to** control access**,** identify threats**, and** ensure safe communication **within and outside a network.**

- **Port Filtering**: Allows or blocks traffic based on port numbers. Example: Allowing HTTP traffic through port 80 and blocking FTP traffic on port 21.
- **IP Address Filtering**: Allows or blocks traffic based on the source or destination IP address. Example: Allowing access only from **192.168.1.100**.
- **Application Filtering**: Filters traffic based on the type of application or protocol. Example: Blocking social media apps or torrents while allowing HTTP and HTTPS traffic.
- **Logging and Monitoring**: Logging stores the history of network events, while monitoring actively tracks network activity to alert for suspicious behavior. Example: Logging all traffic and monitoring for failed login attempts.

Network security is crucial for protecting systems and data from various threats. Here are some **common network security threats** and **measures** to counter them:

1. Malware

- **What it is**: Malware (short for malicious software) refers to software designed to harm or exploit a computer, network, or device. This includes viruses, worms, trojans, ransomware, and spyware.
- **How it works**: Malware often enters systems through email attachments, infected websites, or vulnerabilities in software. Once installed, it can damage files, steal data, or give hackers control of the system.
- **Example**: A virus might attach itself to a legitimate file and spread to other systems when that file is shared.
- **Prevention Measures**:
    - Use **antivirus software** to detect and remove malware.
    - **Regular software updates** to fix vulnerabilities.
    - **User education** to avoid opening suspicious emails or attachments.

---

2. Phishing Attacks

- **What it is**: Phishing is a social engineering attack where attackers trick individuals into providing sensitive information like usernames, passwords, or credit card details by pretending to be a trustworthy entity (such as a bank or service provider).
- **How it works**: Phishing emails or websites appear legitimate but are actually fake. They usually ask the victim to click on a link and input their private information, which is then stolen.
- **Example**: An email pretending to be from your bank asking you to "verify" your account details by clicking on a link.
- **Prevention Measures**:
    - **Educate users** to be cautious of unsolicited emails, especially those asking for personal info.
    - **Multi-factor authentication (MFA)** adds an extra layer of security.
    - **Spam filters** to block suspicious emails.

---

3. Denial of Service (DoS) Attack

- **What it is**: A DoS attack is an attempt to disrupt the normal functioning of a network, service, or server by overwhelming it with excessive traffic, rendering it unavailable to legitimate users.
- **How it works**: The attacker floods a network or website with so much traffic that the server cannot handle it, causing a service outage.
- **Example**: A website might go down because thousands of fake users visit it at once, overwhelming the server's resources.
- **Prevention Measures**:
    - Use **firewalls** and **intrusion detection systems (IDS)** to filter out malicious traffic.

    - Implement **rate-limiting** to control the amount of traffic allowed.
    - **Content delivery networks (CDNs)** can help distribute the load.
    - **DDoS protection** services (e.g., Cloudflare).

---

4. Man-in-the-Middle (MITM) Attack

- **What it is**: A MITM attack occurs when an attacker secretly intercepts and possibly alters the communication between two parties, without them knowing.
- **How it works**: The attacker positions themselves between the sender and the receiver and can eavesdrop, steal sensitive information, or even modify the communication.
- **Example**: An attacker intercepts an unsecured Wi-Fi connection to steal login credentials while you log into your bank account.
- **Prevention Measures**:
    - **Use encryption** (HTTPS, SSL/TLS) for secure communications.
    - Always connect to secure and trusted networks.
    - Use **VPNs** when connecting to public Wi-Fi.

---

5. SQL Injection

- **What it is**: SQL injection is a type of attack where malicious SQL code is inserted into a query, exploiting vulnerabilities in a website or application that interacts with a database.
- **How it works**: The attacker inputs malicious SQL code into input fields (like login forms or search bars). The website's database then executes this code, giving the attacker unauthorized access to sensitive data.
- **Example**: An attacker might enter a code like `DROP TABLE users;` in a login form to delete a database table.
- **Prevention Measures**:
    - Use **parameterized queries** to prevent user input from altering the SQL query.
    - Regularly update and **patch database management systems (DBMS)**.
    - Validate user input to ensure only allowed values are entered.

---

6. Network Security Measures

To protect against the threats mentioned above, organizations implement various network security measures:

- **Firewalls**: Act as a barrier between your network and the internet, controlling incoming and outgoing traffic based on rules.
- **Intrusion Detection Systems (IDS)**: Monitors network traffic for suspicious activity and potential threats.
- **Intrusion Prevention Systems (IPS)**: Detects and responds to potential threats by blocking malicious activities in real-time.

- **Encryption**: Encrypts data to prevent unauthorized access (e.g., using **HTTPS** for web traffic or **VPNs** for private network access).
- **Access Control**: Ensures only authorized users can access certain resources or information through methods like **role-based access control (RBAC)** or **multi-factor authentication (MFA)**.
- **Antivirus Software**: Scans for and removes malware from systems and networks.
- **Patch Management**: Regularly updates software and systems to fix vulnerabilities.
- **Backup and Recovery**: Ensures data is backed up and can be restored in case of a cyberattack or failure.

What is Virtualization?

Virtualization allows you to create multiple "virtual" machines (VMs) on a single physical machine. These VMs can run their own operating systems and applications independently of each other, as if they were separate physical machine

Types of Virtualization:

1. **Server Virtualization:**
    - **What it is**: Divides a physical server into multiple virtual servers.
    - **Example**: A single physical server can run 3 virtual machines, each running a different operating system (like Windows, Linux, etc.).
    - **Benefit**: Better resource utilization and cost savings.
2. **Storage Virtualization:**
    - **What it is**: Combines multiple physical storage devices into a single virtual storage pool.
    - **Example**: A storage system may combine hard drives from multiple servers into a single storage unit for easier management.
    - **Benefit**: Easier to manage and allocate storage.
3. **Network Virtualization:**
    - **What it is**: Creates a virtualized network infrastructure where network resources (like switches, routers) are abstracted and managed centrally.
    - **Example**: A single physical network device could support multiple virtual networks.
    - **Benefit**: Better flexibility and easier network management.
4. **Desktop Virtualization:**
    - **What it is**: Allows users to access virtual desktops from any device.
    - **Example**: Using a virtual desktop infrastructure (VDI), you can work on a "virtual" computer even though the hardware is located elsewhere.
    - **Benefit**: Centralized management and easier updates/upgrades.

How Virtualization Works:

1. **Hypervisor**:
    - The software layer that enables virtualization. It sits between the hardware and the virtual machines.
    - **Types of Hypervisors**:

        - **Type 1 (Bare-metal)**: Runs directly on hardware (e.g., VMware ESXi, Microsoft Hyper-V).
        - **Type 2 (Hosted)**: Runs on top of a host operating system (e.g., VirtualBox, VMware Workstation).
2. **Virtual Machine (VM)**:
    - A virtualized instance that behaves like a physical computer with its own OS and resources, but it runs on top of the hypervisor.
    - **Example**: You can create a VM to run a web server, another VM for a database, and a third one for an application.

Benefits of Virtualization:

1. **Cost Savings**: Reduces the need for physical hardware, lowering power, cooling, and maintenance costs.
2. **Improved Resource Utilization**: Run multiple VMs on a single physical machine to use hardware resources more efficiently.
3. **Flexibility and Scalability**: Easily add more VMs to scale out your infrastructure.
4. **Isolation**: Each VM is isolated, so if one crashes, it doesn't affect others.
5. **Disaster Recovery**: Easier to back up and restore entire VMs, improving disaster recovery.

PORT FORWARDING

Port Forwarding allows external devices (like someone on the internet) to access specific services (like a web server or game server) inside a private network, like your home or office.

Why Use Port Forwarding?

1. **Access Internal Services Externally**: If you want someone to visit your website or use a service inside your home network (e.g., a game server), port forwarding sends that request to the right internal device.
2. **Security and Control**: Only specific services (like a web server) are exposed to the outside world, while other services (like your internal file server) are hidden and secure.

How It Works:

- A request comes to your router, asking to use a specific service (for example, **port 80** for a website).
- The router forwards that request to the correct device on your internal network (e.g., a web server).
- The device responds, and the router sends the response back to the original requester.

Why Use It for Hiding Services?

Port forwarding helps **hide your internal network** by only exposing specific ports (like port 80 for web traffic) to the outside world, keeping other internal systems safe from unwanted access.

Example: If your web server is on IP 192.168.1.100 and your public IP is 203.0.113.1, you set port forwarding to allow access to the web server through **203.0.113.1:80**. External users can reach your server without knowing about other internal systems.

In short, **port forwarding** connects external requests to the right device inside your network and keeps other devices hidden.