Server farm,Firewall,Proxy server,DHCP,Richardson maturity model,12 factor app,port mapping,

**1. Server Farm**

* **Definition**: A server farm is a collection of servers housed together in one location, used for processing, storing, or managing large-scale applications or services.
* **Key Characteristics**:
  + Typically used to ensure high availability, load balancing, and fault tolerance.
  + Servers may be grouped based on similar tasks, such as web servers, database servers, or application servers.
  + Managed through automated tools for scaling and monitoring.

**2. Firewall**

* **Definition**: A firewall is a network security device or software that monitors and controls incoming and outgoing network traffic based on predefined security rules.
* **Types**:
  + **Packet Filtering Firewall**: Examines packets and allows or blocks them based on IP addresses, ports, and protocols.
  + **Stateful Inspection Firewall**: Tracks the state of active connections and uses that information to determine which network packets to allow.
  + **Proxy Firewall**: Acts as an intermediary between clients and servers, hiding the client’s actual IP address.
  + **Next-Generation Firewall (NGFW)**: Includes advanced features like deep packet inspection, intrusion prevention systems, and application-level filtering.
* **Common Uses**:
  + Protects internal networks from external threats.
  + Used in corporate networks to ensure security and enforce security policies.

**3. Proxy Server**

* **Definition**: A proxy server is an intermediary server that sits between a client and a destination server, typically used to filter requests, improve performance, or hide client information.
* **Types**:
  + **Forward Proxy**: Relays requests from internal users to external servers.
  + **Reverse Proxy**: Relays requests from external users to internal servers (often used for load balancing).
  + **Transparent Proxy**: Does not modify requests or responses and is usually used for caching and monitoring.
  + **Caching Proxy**: Stores copies of frequently requested content to improve speed and reduce load on the origin server.
* **Common Uses**:
  + Content filtering and monitoring.
  + Network traffic management.
  + Load balancing and high availability.

**4. DHCP (Dynamic Host Configuration Protocol)**

* **Definition**: DHCP is a network management protocol that automatically assigns IP addresses to devices on a network, along with other relevant network configuration details.
* **How it works**:
  + **DHCP Discover**: A client broadcasts a request for an IP address.
  + **DHCP Offer**: A DHCP server responds with an offer, including an IP address and lease time.
  + **DHCP Request**: The client accepts the offered IP address.
  + **DHCP Acknowledgement**: The DHCP server confirms the lease and assigns the IP address.
* **Benefits**:
  + Simplifies IP address management in large networks.
  + Reduces IP address conflicts.
  + Automatically configures network settings such as DNS and gateway addresses.

**5. Richardson Maturity Model (RMM)**

* **Definition**: A model used to assess the maturity of web APIs based on how well they follow REST (Representational State Transfer) principles.
* **Levels**:
  + **Level 0**: The API is just an RPC (Remote Procedure Call) over HTTP, where each URL corresponds to an action (not RESTful).
  + **Level 1**: Introduces resources, where URLs are mapped to entities and HTTP methods (GET, POST, etc.) are used.
  + **Level 2**: Uses HTTP methods properly (GET, PUT, POST, DELETE) and supports status codes and standard response formats.
  + **Level 3**: Hypermedia-driven APIs (HATEOAS), where clients can navigate the API based on responses (i.e., clients don't need to know URL structure in advance).
* **Purpose**: Helps developers understand how RESTful their API is and how to improve its design.

**6. 12-Factor App**

* **Definition**: A methodology for building scalable, maintainable web applications, especially in a cloud-native environment, focusing on best practices and principles.
* **12 Factors**:
  1. **Codebase**: One codebase tracked in version control.
  2. **Dependencies**: Explicitly declare and isolate dependencies.
  3. **Config**: Store config in environment variables.
  4. **Backing Services**: Treat services like databases as attached resources.
  5. **Build, Release, Run**: Separate build, release, and run stages.
  6. **Processes**: Execute the app as one or more stateless processes.
  7. **Port Binding**: Export services via port binding.
  8. **Concurrency**: Scale out via process model (horizontal scaling).
  9. **Disposability**: Processes should be disposable (easy to start/stop).
  10. **Dev/Prod Parity**: Keep development, staging, and production as similar as possible.
  11. **Logs**: Treat logs as event streams.
  12. **Admin Processes**: Run administrative tasks as one-off processes.
* **Goal**: Enables cloud applications to be easy to scale, maintain, and deploy.

**7. Port Mapping**

* **Definition**: Port mapping refers to the process of associating a specific port on a network device (such as a router or firewall) with a service or application running on an internal network.
* **Common Techniques**:
  + **Port Forwarding**: Maps an external port to an internal port for external access to a specific device or service.
  + **NAT (Network Address Translation)**: Involves port mapping to enable multiple devices on a local network to share a single public IP address.
* **Common Uses**:
  + Allowing remote access to a server behind a router/firewall (e.g., accessing a web server on port 80).
  + Managing network traffic for various services (e.g., gaming, web hosting, FTP).