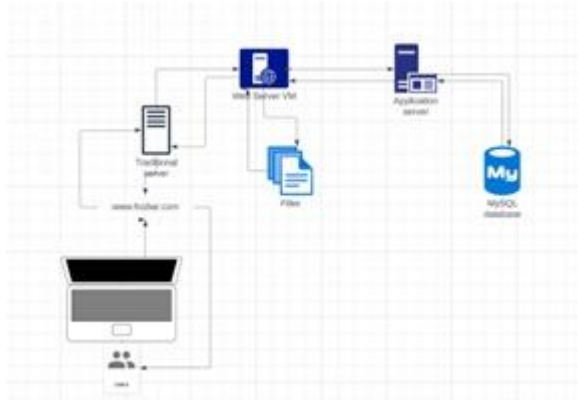


# SIMPLE WEB STACK

A lot of websites are powered by simple web infrastructure, a lot of time it is composed of a single server with a **LAMP stack**.

Below is the diagram illustrating this:



Here are the breakdown of some elements of web infrastructure:

## 1. **SERVER:**

In computing, a server refers to a computer or system that is dedicated to managing and providing resources, services, or data to other computers, known as clients, within a network. Servers play a central role in facilitating communication and resource-sharing among multiple devices. Here are key points to understand about servers:

- ❖ Resource Provider: Servers are designed to offer specific resources, services, or functionalities. Examples include file servers (providing file storage), web servers (hosting websites), database servers (managing databases), and email servers (handling email communication).
- ❖ Responds to Requests: Clients, which can be individual computers, devices, or other servers, send requests to the server to access its resources or services. The server processes these requests and provides the requested information or performs the required tasks.
- ❖ Always On: Servers are typically configured to operate continuously, 24/7, to ensure constant availability. This contrasts with client devices like personal computers or smartphones, which users may turn on and off as needed.
- ❖ Network Connection: Servers are connected to a network, enabling them to communicate with other devices. This network can be a local network within an organisation or the broader internet.
- ❖ Specialized Hardware/Software: Servers often have specialised hardware and software configurations to handle specific tasks efficiently. For example, a web server might use software like Apache or Nginx to serve web pages.

- ❖ Security Considerations: Servers may implement security measures to protect the data and services they provide. This can include firewalls, encryption, and access controls.

## 2. ROLE OF DOMAIN NAME:

A web domain is essential for making websites easy to find. It serves as a friendly address, helping people access websites in a way that's easy to remember. Here are its main roles:

- ❖ User-Friendly ID: A domain makes finding a website straightforward. It's a memorable way for people to navigate the internet instead of dealing with confusing numbers.
- ❖ Web Address: A domain is part of a website's URL (web address), indicating where a specific page or resource is located. In "https://www.example.com/page," "[www.example.com](https://www.example.com)" is the domain.
- ❖ Brand Representation: Domains often reflect a brand or identity. They help establish a unique online presence, making it easier for users to recognize and remember a website.
- ❖ Easy Access: Users access websites by typing the domain into a web browser. The domain is crucial for translating user-friendly names into numeric IP addresses through the Domain Name System (DNS).
- ❖ Email Identification: Domains are commonly used in email addresses. In "info@example.com," "example.com" is the domain, directing emails to the correct mail server.
- ❖ Business and Promotion: A well-chosen domain is valuable for businesses, aiding in brand recognition, marketing efforts, and online visibility. It's a key part of an organisation's online branding strategy.

## 3. The DNS record “www” in [www.foobar.com](https://www.foobar.com):

The DNS record associated with the "www" subdomain in "[www.foobar.com](https://www.foobar.com)" is typically a type of DNS record known as a "CNAME" record, which stands for Canonical Name. A CNAME record is used to alias one domain to another. In the context of the "www" subdomain, the CNAME record points to the main domain. This means that the "www" subdomain is an alias for (or points to) the main domain "foobar.com." When users enter "[www.foobar.com](https://www.foobar.com)" in a web

browser, the CNAME record directs the request to the same IP address or server as "foobar.com."

#### 4. THE ROLE OF THE WEB SERVER:

The web server has a lot of roles in a web infrastructure. Here is a breakdown:

- ❖ Content Delivery: The web server's main job is to deliver web content to users when they ask for it. This content includes things like web pages, images, stylesheets, scripts, and more, basically everything that makes up a website.
- ❖ Request Handling: When a user's web browser requests a specific webpage or resource, the web server takes care of that request. It finds the requested files and sends them back to the user's browser so they can see the webpage.
- ❖ Protocol Support: Web servers support communication using the Hypertext Transfer Protocol (HTTP) or its secure version, HTTPS. They listen for incoming requests on a specific port, usually port 80 for HTTP and 443 for HTTPS.
- ❖ Processing Dynamic Content: In addition to serving static files, web servers often team up with application servers to handle dynamic content. This means they can generate content on the spot based on what the user is asking for.
- ❖ Security: Web servers help with security by doing things like using SSL/TLS encryption, especially when it comes to HTTPS. This makes sure that the information travelling between the server and the user's browser stays secure.
- ❖ Logging and Monitoring: Web servers keep track of information about incoming requests, errors, and what the server is up to. This kind of logging is essential for server administrators to keep an eye on how well the server is doing, find problems, and see what users are up to.
- ❖ Load Balancing: In more complex situations, web servers might work with load balancers. These balancers help spread out the incoming traffic across multiple servers. This is handy because it makes sure the servers don't get overwhelmed, especially during busy times.

#### 5. ROLE OF APPLICATION SERVER:

The role of an application server is to provide a runtime environment for the execution of software applications. It acts as an intermediary between the

client (typically a web browser or another application) and the backend database or other resources. Here's a breakdown of its key roles:

- ❖ Execution Environment: An application server provides a runtime environment where software applications can run. It handles the execution of application code, managing resources, and ensuring proper communication with other components.
- ❖ Middleware Integration: Application servers often act as middleware, facilitating communication and integration between different software components. They can handle tasks such as data access, security, and transaction management
- ❖ Concurrency and Scalability: Application servers manage the concurrent execution of multiple requests or sessions, ensuring that applications can handle multiple users simultaneously. They also support scalability by allowing the addition of more server instances to distribute the load.
- ❖ Connection Handling: Application servers manage and handle incoming connections from clients. They provide services such as connection pooling to optimise the use of resources and improve performance.
- ❖ Security and Authentication: Application servers often include features for securing applications. This includes handling user authentication, authorization, and protecting sensitive data.
- ❖ Transaction Management: For applications that involve database transactions, application servers provide transaction management services. They ensure that transactions are executed reliably and consistently, even in the event of failures.
- ❖ Load Balancing: In a distributed environment, multiple instances of an application server can be deployed. Load balancing is a common feature where incoming requests are distributed across these instances to optimise resource usage and improve overall system performance.
- ❖ Resource Management: Application servers manage resources such as database connections, threads, and memory. They help ensure efficient utilization of resources and prevent issues like memory leaks.

Essentially, an application server plays a crucial role in providing a runtime environment, facilitating communication between different software components, managing resources, ensuring security, and handling various aspects of concurrent and scalable application execution.

## 6. THE ROLE OF THE DATABASE:

Here are the breakdown of some roles the database plays in a web infrastructure:

- ❖ Data Storage: The database stores and organizes data for applications, ensuring efficient retrieval and updating.
- ❖ Data Retrieval: It provides a structured way to retrieve specific pieces of information, enabling quick and precise data access.
- ❖ Data Integrity: The database enforces rules to maintain the accuracy and consistency of stored data, preventing errors and ensuring reliability.
- ❖ Data Relationships: It allows the establishment of relationships between different sets of data, enabling complex queries and comprehensive data analysis.
- ❖ Data Security: The database implements access controls and authentication mechanisms to safeguard sensitive information from unauthorized access.
- ❖ Concurrency Control: In multi-user environments, the database manages concurrent access to ensure that multiple users can interact with the data without conflicts.
- ❖ Transaction Management: It supports transactions, ensuring that a series of operations either succeed entirely or fail entirely, maintaining data consistency.
- ❖ Scalability: Databases can scale vertically (adding more resources to a single server) or horizontally (distributing data across multiple servers) to handle growing amounts of data and increased demand.

## 7. SERVER COMMUNICATION PROCESS:

When a user requests a website, the communication between the server and the user's computer primarily happens through the Hypertext Transfer Protocol (HTTP) or its secure variant, HTTPS (HTTP Secure). This communication is facilitated by the use of the Internet Protocol (IP) and Transmission Control Protocol (TCP) at lower levels of the networking stack.

Here's a simplified overview of the process:

- ❖ HTTP/HTTPS Communication: When a user enters a website URL (e.g., www.foobar.com) in their web browser and hits Enter, the browser initiates an HTTP or HTTPS request. For HTTPS, the communication is encrypted using Transport Layer Security (TLS) to ensure secure data transmission.
- ❖ DNS Resolution: The user's computer performs a Domain Name System (DNS) lookup to translate the human-readable domain (e.g., www.foobar.com) into an IP address.
- ❖ Establishing a TCP Connection: The user's computer opens a TCP connection to the server's IP address. This connection is established on a specific port, commonly port 80 for HTTP or port 443 for HTTPS.
- ❖ Request and Response: The user's computer sends an HTTP or HTTPS request to the server, specifying the desired resource (e.g., the homepage). The server processes the request and sends back an HTTP or HTTPS response containing the requested content.
- ❖ Rendering the Web Page: The user's browser receives the response and renders the web page, displaying text, images, and other resources as instructed by the received HTML, CSS, and JavaScript.
- ❖ Closing the Connection: Once the communication is complete, the TCP connection is closed. This communication flow allows users to access and interact with web content hosted on a server. It relies on standard protocols (HTTP/HTTPS), networking protocols (IP, TCP), and encryption (TLS for HTTPS) to ensure secure and reliable data transfer between the user's computer and the server.

As perfect as a web infrastructure may seem, there are some issues that may arise with it. Here is a breakdown of these issues and best practices to avoid it:

1. SPOF (Single Point of Failure):
  - Issue: The presence of a Single Point of Failure means that if a critical component fails, the entire system may go down.
  - Impact: This poses a significant risk to the availability and reliability of the system. If the single point of failure fails, the entire application or service becomes inaccessible.
2. Downtime During Maintenance:
  - Issue: Requiring web servers to be restarted during maintenance or when deploying new code introduces downtime.
  - Impact: Users may experience interruptions in service, leading to a negative user experience. Downtime during maintenance can be disruptive, especially if the application needs to be frequently updated.

### 3. Limited Scalability:

- Issue: The inability to scale efficiently in response to high incoming traffic indicates a scalability bottleneck.
- Impact: During periods of increased user activity, the system may become slow or unresponsive. This limitation can hinder the ability to handle growth and may result in a poor user experience during peak times.

To address these issues, best practices might include:

1. Redundancy and High Availability: Introduce redundancy for critical components to eliminate single points of failure. This could involve using load balancing and having multiple instances of key services.
2. Rolling Deployments: Implement deployment strategies that minimize or eliminate downtime. Techniques such as canary releases or blue-green deployments can help roll out updates without affecting the entire system at once.
3. Scalability Solutions: Use scalable infrastructure, such as auto-scaling and load balancing, to handle increased traffic dynamically. This allows the system to adapt to varying workloads without compromising performance.

Addressing these issues contributes to a more robust and resilient infrastructure, ensuring better availability, reliability, and scalability for the application or service.