



**FREE Live  
Classes**



Theory  
Coding



# Core Java

by Karthik Ponnusamy

**Course Contents:-**

- Core Java Basic Concepts
- OOP Concepts
- Fundamentals of Java Programming
- Java Control Flow Statements
- Deep dive into Collection Framework
- Deep dive into Multithreading
- Java 1.8 Features

**Batch Starts from Jan - 18 - 2022**  
**Mon to Thurs - 10 PM to 11 PM CST /**  
**9:30 AM to 10:30 AM IST**



# Day 01: Agenda

## Client Server Architecture

What is client server architecture?

Client server architecture example

Components of client server architecture

How does client server architecture works internally?

Types of client server architecture - 1-tier, 2-tier, 3-tier, N-tier?

Advantages and disadvantages of client-server architecture

# What is client server architecture?



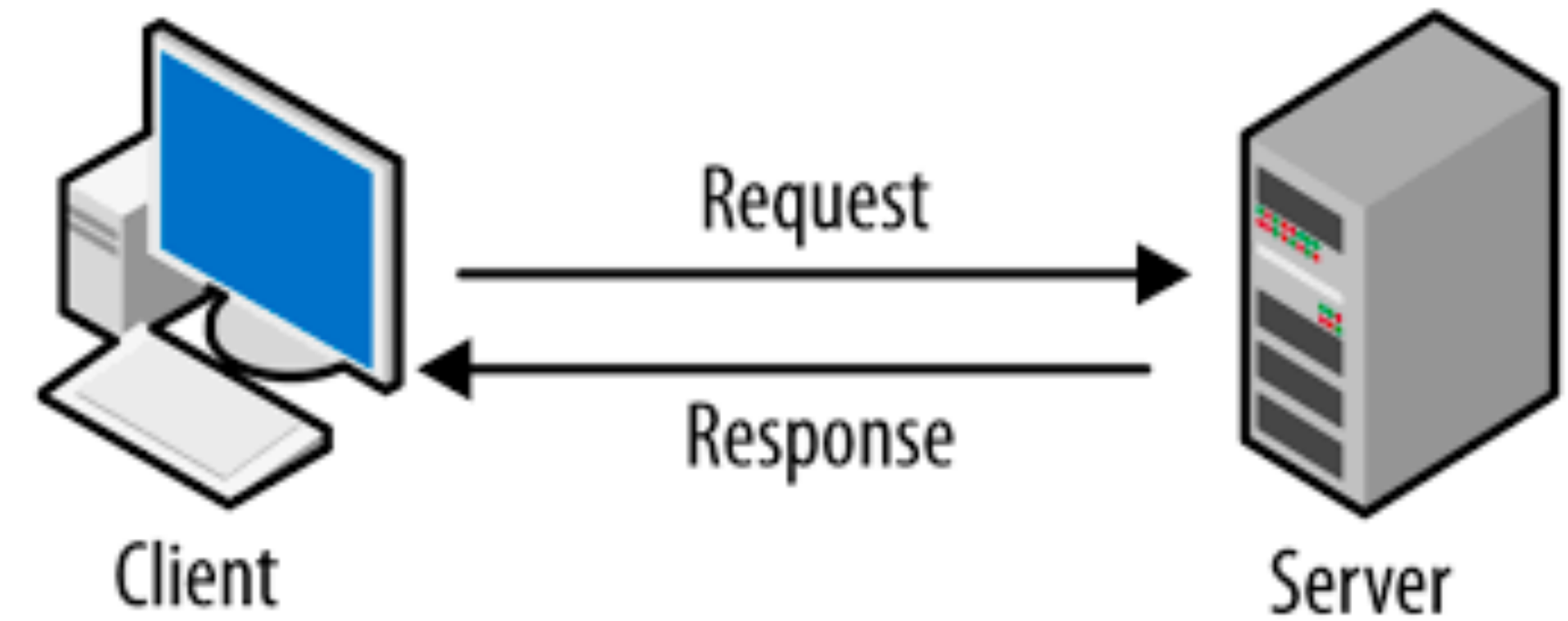
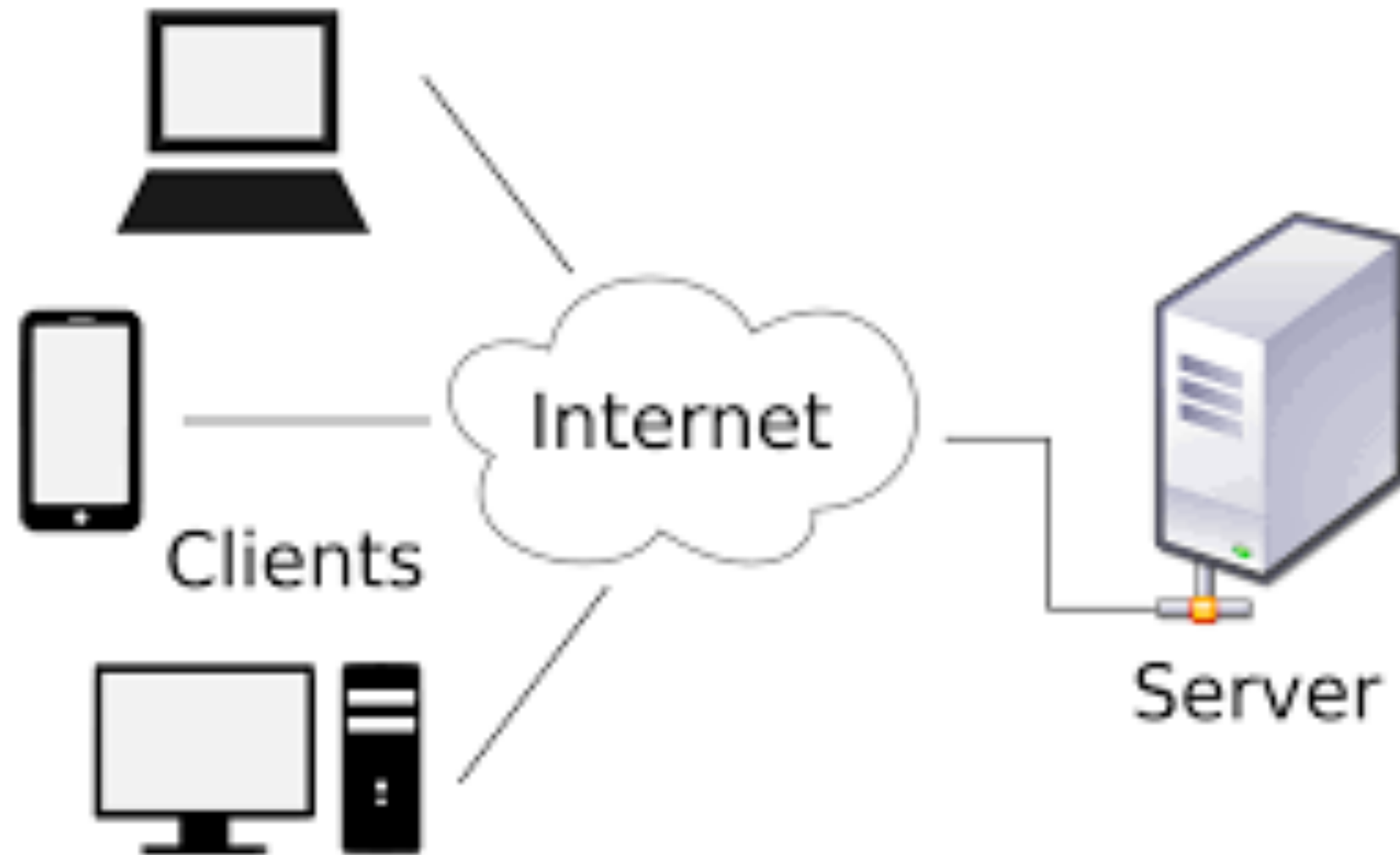
Client-server architecture is a computing model that **divides the work of a system** into two separate categories: clients and servers. In this architecture, **clients request services** or resources from servers, which then **provide the services** or resources requested by the clients.

The client is the user interface or application that runs on a user's device and sends requests to the server. The client can be a web browser, a mobile app, or a desktop application, among others. Clients are responsible for **initiating requests to servers** and displaying the results to users.

The **server is the computer that provides services to clients**. Servers can be physical or virtual machines that run software to provide services to clients. Servers receive requests from clients, process them, and return the results. Servers can provide various services such as web hosting, file storage, email, database access, and many others.

The client-server architecture is widely used in distributed computing environments, where multiple devices or systems work together to perform complex tasks. It provides a scalable and flexible model for building complex systems that can be accessed from anywhere in the world, as long as there is an internet connection.

# What is client server architecture?





# Client server architecture example



An example of a client-server architecture is the **World Wide Web**. The web browser on a client device, such as a laptop or smartphone, requests web pages from web servers hosted on remote machines. The web server receives the request and sends back the requested web page to the client's web browser.

Here is a breakdown of the client-server architecture in the context of the World Wide Web:

**The client:** The client is the web browser that runs on a user's device, such as a laptop or smartphone. The web browser sends requests to the web server for web pages.

**The server:** The server is the computer that hosts the web server software and the web pages that are requested by clients. The web server software receives requests from clients and sends back the requested web pages.

**HTTP protocol:** The communication between the client and the server is done through the HTTP (Hypertext Transfer Protocol) protocol. HTTP is a client-server protocol, where the client sends requests to the server, and the server sends back responses.

**URL:** The client sends a URL (Uniform Resource Locator) to the server to request a specific web page. The URL contains the domain name of the server and the path to the requested web page.

**HTML:** The web pages sent back by the server are in HTML (Hypertext Markup Language) format. The client's web browser interprets the HTML code and displays the web page to the user.

This is just one example of the client-server architecture, which is used in many other distributed computing environments, such as email, file transfer, and database access.

# Components of client server architecture



## Client:

The client component is responsible for **initiating requests** to the server and receiving responses. Clients can be any device with the ability to send requests, such as a web browser, mobile app, or desktop application.

## Server:

The server component is responsible for **receiving requests from clients, processing the requests, and sending back responses**.

Servers can be physical or virtual machines that run server software, such as web servers, database servers, or email servers.

Other important components of the client-server architecture include:

## Network:

The network is the **communication medium** that connects the client and the server. The network can be a local area network (LAN), a wide area network (WAN), or the internet.

## Protocol:

Protocols are the **rules and formats that govern the communication** between clients and servers. Examples of protocols used in client-server architecture include HTTP, FTP, SMTP, and TCP/IP.

## Middleware:

Middleware is the software that **connects clients and servers**, provides **translation services** between different protocols, and manages communication between them. Examples of middleware include web application servers, message brokers, and database middleware.



# How does client server architecture works internally?

The client-server architecture works internally through a series of steps that involve communication between the client and server components. Here is a basic overview of how the client-server architecture works internally:

## **The client sends a request to the server:**

The client initiates communication by sending a request to the server. The request can be a simple message, a file transfer request, or a request for a web page. The request includes the necessary information, such as the data to be transferred and the type of operation to be performed.

## **The server receives the request:**

The server component receives the request from the client. It then processes the request and performs the necessary operation based on the request type.

## **The server sends a response to the client:**

Once the server has completed the requested operation, it sends a response back to the client. The response includes the requested data or an acknowledgment that the operation has been completed.

## **The client receives the response:**

The client receives the response from the server and processes it based on the type of operation performed. For example, if the client requested a file transfer, it will save the transferred file to its local storage.

## **Communication is terminated:**

Once the operation is completed and the response has been received, the communication between the client and server is terminated. The client is then free to initiate a new request if needed.

This is a basic overview of how the client-server architecture works internally. However, the actual implementation can be much more complex, involving multiple layers of protocols, middleware, and security mechanisms to ensure reliable and secure communication between the client and server components.



# Types of client server architecture



**1-tier architecture**, also known as single-tier architecture, is a type of software architecture in which **all the components of an application run on a single machine**. In other words, the **user interface, application logic, and database** are all combined into a single program, and there is no separation between the client and server components. In this architecture, the user interacts directly with the application running on the same machine. There is no need for network communication or remote server processing, which means that this architecture is relatively simple and easy to develop.

**Examples** of applications that use 1-tier architecture include **standalone desktop applications and simple websites with a few static pages**.

However, this architecture has several limitations, such as lack of scalability, poor fault tolerance, and security vulnerabilities, which makes it unsuitable for complex and large-scale applications. As a result, most modern applications use multi-tier client-server architectures such as 2-tier, 3-tier, or N-tier architectures.

## 2-Tier Architecture:

Also known as client-server architecture, this model consists of two layers: a **client and a server**. The client sends requests to the server, which processes the request and sends back a response. This architecture is used in **applications that require a high level of interactivity and responsiveness**, such as online gaming and stock trading applications.

## 3-Tier Architecture:

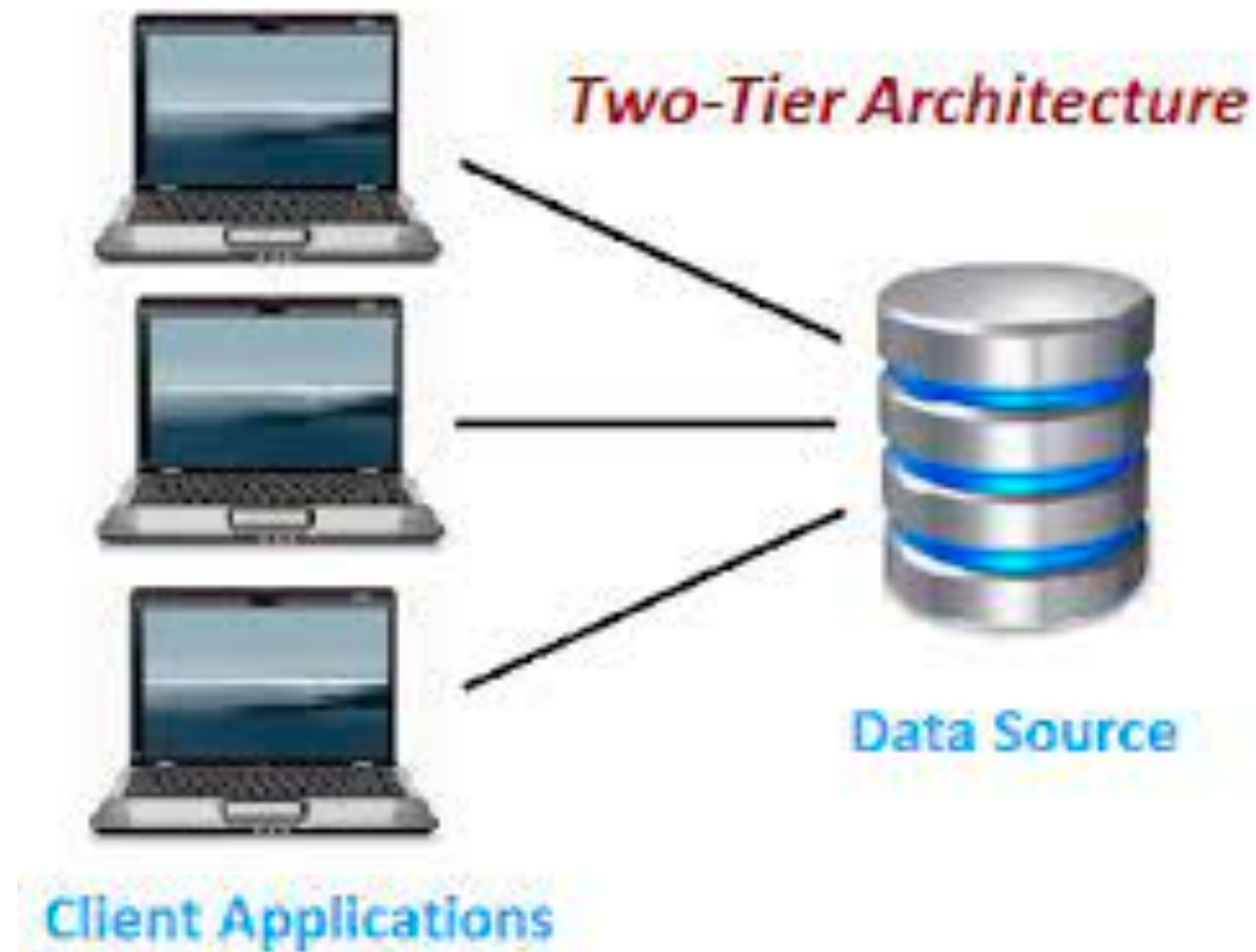
Also known as multi-tier architecture, this model consists of three layers: a **client, an application server, and a database server**. The client sends requests to the application server, which processes the request and communicates with the database server to retrieve or update data. This architecture is used in applications that **require a high level of scalability and security, such as e-commerce and banking applications**.

## N-Tier Architecture:

This model consists of multiple layers, with **each layer performing a specific set of functions**. In addition to the client, application server, and database server layers found in the 3-tier architecture, this model can include **additional layers for security, middleware, and other specialized functions**. This architecture is used in complex applications that **require a high level of flexibility and modularity, such as enterprise resource planning (ERP) systems and large-scale data processing applications**.



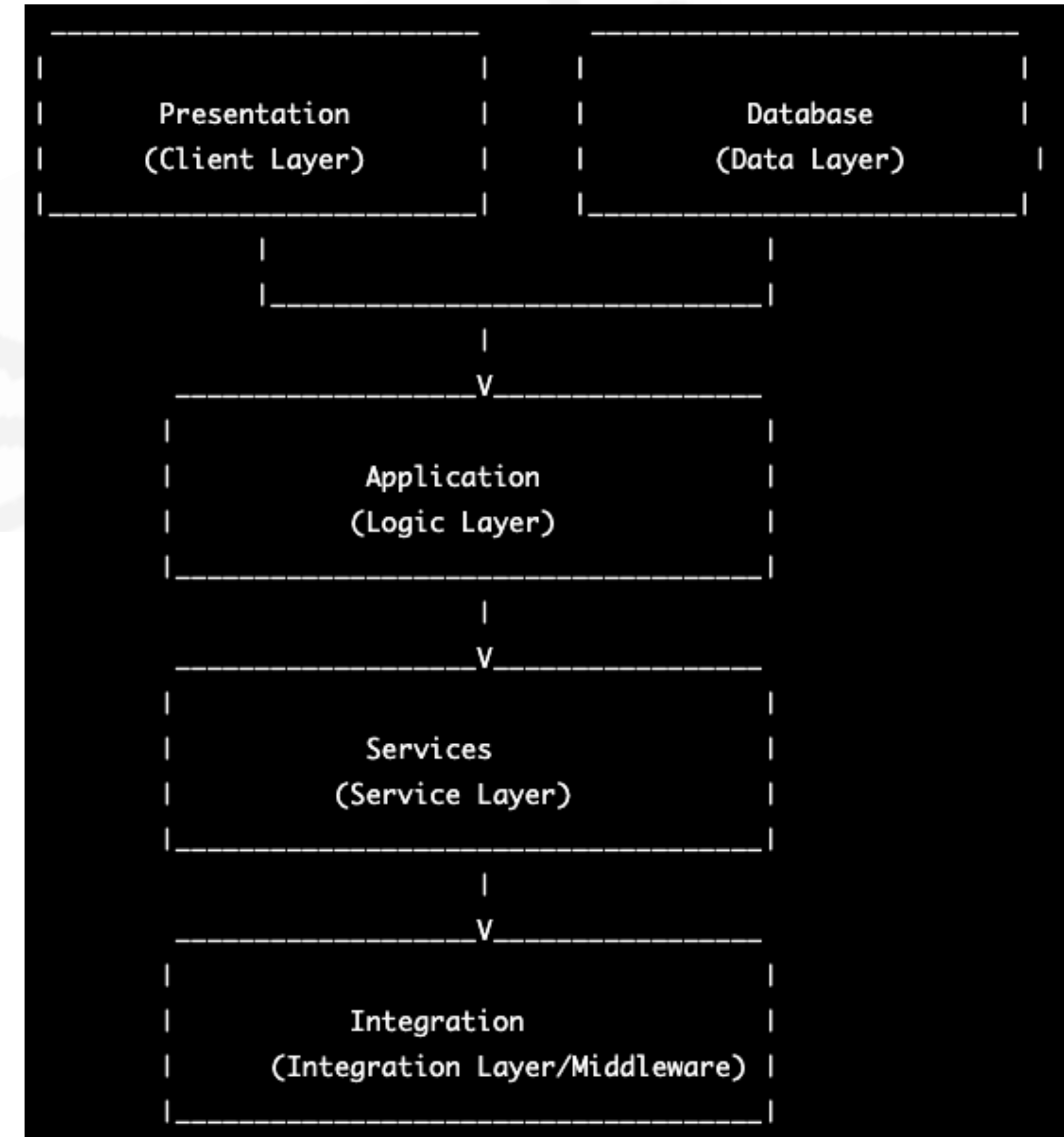
# Types of client server architecture



## 3 - Tier Architecture



## N - Tier Architecture



# Advantages and disadvantages of client-server architecture

## Advantages:

**Scalability:** The client-server architecture can be easily scaled by adding or removing servers or clients depending on the demand, making it suitable for large-scale applications.

**Centralized management:** The server component can be used to manage and control the application and data centrally, providing better security and easier maintenance.

**Resource sharing:** The server component can provide resources and services to multiple clients, reducing the duplication of effort and resources.

**Flexibility:** The client-server architecture enables the separation of concerns between the client and server components, allowing each component to be developed and deployed independently.

**Better performance:** The client-server architecture can reduce the workload of the client and improve performance by delegating some of the processing to the server component.

## Disadvantages:

**Complexity:** The client-server architecture can be complex to design, develop, and maintain, requiring specialized knowledge and expertise.

**Network dependence:** The client-server architecture depends on network communication, which can be slow, unreliable, or vulnerable to security threats.

**Single point of failure:** The server component can be a single point of failure, affecting the entire application if it fails or goes down.

**Cost:** The client-server architecture can be more expensive to implement and operate than other architectures due to the need for servers, network infrastructure, and specialized software.

**Security risks:** The client-server architecture can be vulnerable to security risks such as data breaches, denial-of-service attacks, and other network-based attacks.



