For any web application, the API services work as the **building block**. Therefore it becomes very important for you, as a tester, to ensure that the **web services are bug-free** and facilitate the **exchange of information securely**.

**What is Client Server Architecture?**

Client server architecture in a simple sense can be stated as a **consumer-producer model where the client acts as the consumer i.e. the service requestor and the server is the producer, *i.e.* the service provider**. Let us see how they are described in the computing sense.

***What is a Client?***

A client in computing is a system or a program that connects with a remote system or software to fetch information. **The client makes a request** to the server and is responded with information. There may be three types of clients - thick, thin or hybrid client.

* ***A thick or fat client performs the operation itself without the need for a server****.* ***A personal computer*** *is a fine example of a thick client as the majority of its operations are independent of an external server.*
* ***A thin client uses the host computer's resources, where the thin client presents the data processed by an application server. A web browser is an example of a thin client.***
* ***A hybrid client is a combination of a thick and a thin client****. Just like the thick client it processes data internally but relies on the application server for persistent data.* ***Any online gaming run*** *on a system can be an example of a hybrid client.*

***What is a Server?***

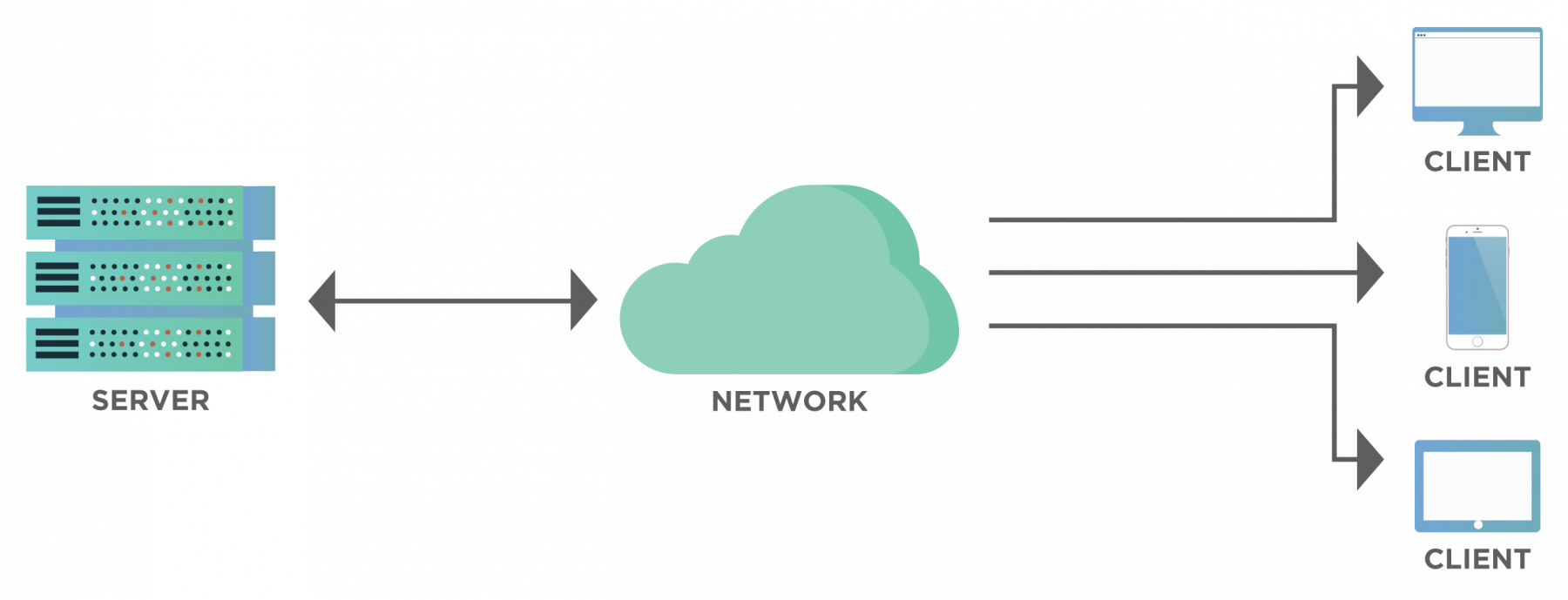
A server is a system or a computer program **that acts as the data provider**. It may provide data through LAN (*Local Area Network*) or WAN (*Wide Area Network*) using the internet. **The functionalities provided by the server are called services**. These services are provided as a response to the request made by the clients. Some of the common servers are-

* ***Database Server****- used to maintain and share the database over a network.*
* ***Application Server****- used to host applications inside a web browser allowing to use them without installation locally.*
* ***Mail Server****- used for email communication.*
* ***Web Server****- used to host web pages because of which worldwide web is possible.*
* ***Gaming Server****- used for playing multiplayer games.*
* ***File Server****- used for sharing files and folders over a network.*

These clients and servers do not necessarily be at the same location. They could either be located in different locations or may reside as different processes on the same computer. They are connected via the Web and interact via the ***HTTP protocol*** which will be discussed later in this article. There may be multiple clients requesting a server and alternatively, a client can request from multiple servers.

***Visualising the client server architecture***

The below diagram visualizes a typical client-server model-



As shown in the diagram, **a single server may serve the requests of multiple clients.** Similarly, **a client may be requesting data from different servers**. For example, consider an example of *Google*. Here, Google acts as the server and the users sitting at different places act as the clients.

Another example that we come across in our everyday life is the Online Banking Portal in which the browser that we use to open a portal acts as the client while the database and the software for banking act as the server. This propagates resource sharing across multiple users and thereby results in cost-efficiency along with time saving.

**How does the Client Server Architecture work?**

The client server model acts like a consumer-server relationship. But how does it work? Let us see some of the flow of information in a client-server architecture through a series of steps as discussed below-

1. ***The HTTP communication protocol helps establish the connection between the client and the server.***
2. ***The client sends a request in the form of an XML or a JSON over the active connection. The client and server both understand the message format.***
3. ***Upon receiving the client request, the server searches the requested data and sends back the relevant details as a response in the same format in which the request was received.***



The above diagram shows the communication process between the client and the server. The client sends an ***HTTP request***, for which the server sends an ***HTTP response***.

***Visualising the flow of client server architecture***

Assume that you need to go shopping from your home to a shop across the road. You may travel on a bicycle and on reaching can order the salesman the goods you need. Now if the salesman finds the goods, he will bring them to you else he would let you know about their unavailability.

Now, you may link the above example with the navigation to a website, the street between your home and the shop is the internet connection. The mode of transport that you chose to travel is the [***TCP/IP***](http://www.tcpipguide.com/), defining the communication protocol for the data to travel through the internet. The address of the shop is the DNS (*Domain Name Server*) of the website. Your communication language with the salesman is the HTTP (*Hypertext Transfer Protocol*) which defines the language for interaction between the client and the server. Your order request is the [***HTTP request***](https://www.toolsqa.com/client-server/http-request/) and the update on item availability is the [***HTTP response***](https://www.toolsqa.com/client-server/http-response/).

When the web browser sends a request to the server with the DNS of the website, and the server approves the client request, it sends a 200 OK success message. This message means that the server has located the website and it sends back the website files in small chunks of data to the browser. The browser then collects and assembles these small chunks to form the complete website and displays it to us.

**What are the different types of Client-Server architectures?**

Client-server architecture has the following four types -

1. *1-Tier Architecture*
2. *2-Tier Architecture*
3. *3-Tier Architecture*
4. *N-Tier Architecture*

***1-Tier Architecture***

In 1-tier architecture, **the business logic, data logic, and the user interface all reside on the same machine**.

The environment is simple and cheap because **the client and the server lie on the same system**, but the data variance leads to the repetition of work. Such systems store data in a local file or a shared driver. Examples of 1-tier applications are the **MP3 player** or the MS Office files.

***2-Tier Architecture***

The 2-tier architecture provides the best environment in terms of performance due to the absence of any intervening server.

**The user interface resides on the client side while the database on the server side. The database and business logic can be stored either at the client or the server end, but they must remain unchanged.**

If both reside at the client end then the architecture is called ***fat client thin server architecture***. On the contrary, if both reside at the server end, the architecture is called ***thin client fat server architecture***. An online ticket reservation system generally uses a 2-tier architecture.

***3-Tier Architecture***

The 3-tier architecture involves a middleware used for interaction between the client and the server. Though it is expensive but is very easy to use. The middleware improves performance and flexibility. It stores the business and the data logic. The three layers in the 3-tier architecture are-

* *Presentation Layer(Client tier)*
* *Application Layer(Business tier)*
* *Database Layer(Data tier)*

**Almost all web applications are examples of 3-tier architecture**.

***N-Tier Architecture***

The n-tier architecture is the scaled form of 3-tier architecture. In such an environment, the processing, data management, and presentation function are isolated in different layers. The isolation makes the system easy to manage and maintain. This is also referred to as multi-tier architecture.

**What are the advantages of using Client-Server Architecture?**

Client-server architecture has the following advantages -

* *It maintains data at one central location.*
* *It provides backup and data recovery options.*
* *Accessing the data from a single server help in cost-efficiency with less maintenance.*
* *The capacity of the client and the server can be individually modified.*

**Issues of Client-Server Architecture?**

Client-server architecture has the following disadvantages -

* *A virus can easily attack a client present on the server.*
* *In case of network failure, the entire architecture of an application can suffer.*
* *Man-in-the-middle attack or data spoofing is possible during data transmission.*
* *Due to vulnerability to different kinds of attacks, it requires a special and secure network operating system.*
* *Possibility of loss of data packets either completely or modification because of some intrusion during the transmission.*

**Key Takeaways**

So, what are the takeaways from this article?

* *The client server model is similar to a salesman-customer relationship.*
* *The communication between the client and the server facilitates through the HTTP protocol.*
* *The client-server architecture is categorized into 1-tier, 2-tier, 3-tier and n-tier.*
* *We discussed the various advantages of the client-server model including its centralized framework.*
* *We then saw the issues with the architecture as to how the client-server model is vulnerable to different attacks.*