Top 85 Software Architecture Interview Questions

*Q1*:

## What Is CAP Theorem?

Answer

**The CAP Theorem for distributed computing** was published by Eric Brewer. This states that it is not possible for a distributed computer system to simultaneously provide all three of the following guarantees:

1. Consistency (all nodes see the same data even at the same time with concurrent updates )
2. Availability (a guarantee that every request receives a response about whether it was successful or failed)
3. Partition tolerance (the system continues to operate despite arbitrary message loss or failure of part of the system)

The CAP acronym corresponds to these three guarantees. This theorem has created the base for modern distributed computing approaches. Worlds most high volume traffic companies (e.g. Amazon, Google, Facebook) use this as basis for deciding their application architecture. It's important to understand that only two of these three conditions can be guaranteed to be met by a system.

*Q2*:

## What Is Load Balancing?

Answer

**Load balancing** is simple technique for distributing workloads across multiple machines or clusters. The most common and simple load balancing algorithm is Round Robin. In this type of load balancing the request is divided in circular order ensuring all machines get equal number of requests and no single machine is overloaded or underloaded.

**The Purpose of load balancing is to**

* Optimize resource usage (avoid overload and under-load of any machines)
* Achieve Maximum Throughput
* Minimize response time

**Most common load balancing techniques in web based applications are**

1. Round robin
2. Session affinity or sticky session
3. IP Address affinity

*Q3*:

## Define Microservice Architecture

Answer

**Microservices**, aka **Microservice Architecture**, is an architectural style that structures an application as a collection of small autonomous services, modeled around a **business domain.**

*Q4*:

## What do you mean by lower latency interaction?

  Related To: [WebSockets](https://www.fullstack.cafe/interview-questions/websockets)

**Add to PDF** **Junior**

Answer

**Low latency** means that there is very little delay between the time you request something and the time you get a response. As it applies to webSockets, it just means that data can be sent quicker (particularly over slow links) because the connection has already been established so no extra packet roundtrips are required to establish the TCP connection.

*Q5*:

## Why use WebSocket over HTTP?

  Related To: [WebSockets](https://www.fullstack.cafe/interview-questions/websockets)

**Add to PDF** **Junior**

Answer

A **WebSocket** is a continuous connection between client and server. That continuous connection allows the following:

1. Data can be sent from server to client at any time, without the client even requesting it. This is often called server-push and is very valuable for applications where the client needs to know fairly quickly when something happens on the server (like a new chat messages has been received or a new price has been udpated). A client cannot be pushed data over http. The client would have to regularly poll by making an http request every few seconds in order to get timely new data. Client polling is not efficient.
2. Data can be sent either way very efficiently. Because the connection is already established and a webSocket data frame is very efficiently organized, one can send data a lot more efficiently that via an HTTP request that necessarily contains headers, cookies, etc...

*Q6*:

## What Is Scalability?

  Related To: [Scalability](https://www.fullstack.cafe/interview-questions/scalability)

**Add to PDF** **Junior**

Answer

Scalability is the ability of a system, network, or process to handle a growing amount of load by adding more resources. The adding of resource can be done in two ways

* **Scaling Up**  
  This involves adding more resources to the existing nodes. For example, adding more RAM, Storage or processing power.
* **Scaling Out**  
  This involves adding more nodes to support more users.

Any of the approaches can be used for scaling up/out a application, however the cost of adding resources (per user) may change as the volume increases. If we add resources to the system It should increase the ability of application to take more load in a proportional manner of added resources.

An ideal application should be able to serve high level of load in less resources. However, in practical, linearly scalable system may be the best option achievable. Poorly designed applications may have really high cost on scaling up/out since it will require more resources/user as the load increases.

*Q7*:

## What Is A Cluster?

**Add to PDF** **Junior**

Answer

A cluster is group of computer machines that can individually run a software. Clusters are typically utilized to achieve high availability for a server software. Clustering is used in many types of servers for high availability.

* **App Server Cluster**  
  An app server cluster is group of machines that can run a application server that can be reliably utilized with a minimum of down-time.
* **Database Server Cluster**  
  An database server cluster is group of machines that can run a database server that can be reliably utilized with a minimum of down-time.

*Q8*:

## Why Do You Need Clustering?

**Add to PDF** **Junior**

Answer

Clustering is needed for achieving high availability for a server software. The main purpose of clustering is to achieve 100% availability or a zero down time in service. A typical server software can be running on one computer machine and it can serve as long as there is no hardware failure or some other failure. By creating a cluster of more than one machine, we can reduce the chances of our service going un-available in case one of the machine fails.

Doing clustering does not always guarantee that service will be 100% available since there can still be a chance that all the machine in a cluster fail at the same time. However it in not very likely in case you have many machines and they are located at different location or supported by their own resources.

*Q9*:

## What defines a software architect?

**Add to PDF** **Junior**

Answer

An **architect** is the captain of the ship, making the decisions that cross multiple areas of concern (navigation, engineering, and so on), taking final responsibility for the overall health of the ship and its crew (project and its members), able to step into any station to perform those duties as the need arises (write code for any part of the project should they lose a member). He has to be familiar with the problem domain, the technology involved, and keep an eye out on new technologies that might make the project easier or answer new customers' feature requests.

*Q10*:

## Why is it a good idea for “lower” application layers not to be aware of “higher” ones?

  Related To: [Layering & Middleware](https://www.fullstack.cafe/interview-questions/layering-and-middleware)

**Add to PDF** **Junior**

Answer

The fundamental motivation is this:

You want to be able to rip an entire layer out and substitute a completely different (rewritten) one, and NOBODY SHOULD (BE ABLE TO) NOTICE THE DIFFERENCE.

The most obvious example is ripping the bottom layer out and substituting a different one. This is what you do when you develop the upper layer(s) against a simulation of the hardware, and then substitute in the real hardware.

Also layers, modules, indeed architecture itself, are means of making computer programs easier to understand by humans.

*Q11*:

## What is meant by the KISS principle?

**Add to PDF** **Junior**

Answer

**KISS**, a backronym for "keep it simple, stupid", is a design principle noted by the U.S. Navy in 1960. The KISS principle states that most systems work best if they are kept simple rather than made complicated; therefore simplicity should be a key goal in design, and that unnecessary complexity should be avoided.

*Q12*:

## What is Domain Driven Design?

  Related To: [DDD](https://www.fullstack.cafe/interview-questions/ddd)

**Add to PDF** **Junior**

Answer

**Domain Driven Design** is a methodology and process prescription for the development of complex systems whose focus is mapping activities, tasks, events, and data within a problem domain into the technology artifacts of a solution domain.

It is all about trying to make your software a model of a real-world system or process.

*Q13*:

## What is Test Driven Development?

  Related To: [Agile & Scrum](https://www.fullstack.cafe/interview-questions/agile-and-scrum)

**Add to PDF** **Junior**

Answer

**Test Driven Development (TDD)** is also known as test-driven design. In this method, developer:

1. first writes an **automated test** case which describes new function or improvement and
2. then creates small codes to **pass that test**, and
3. later **re-factors** the new code to meet the **acceptable standards**.

*Q14*:

## What does the expression “Fail Early” mean, and when would you want to do so?

Answer

Essentially, **fail fast** (a.k.a. **fail early**) is to code your software such that, **when there is a problem, the software fails as soon as and as visibly as possible**, rather than trying to proceed in a possibly unstable state.

**Fail Fast** approach won’t reduce the overall number of bugs, at least not at first, but it’ll make most defects much easier to find.

https://www.fullstack.cafe/interview-questions/software-architecture