**What Is CAP Theorem?**

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:

Consistency (all nodes see the same data even at the same time with concurrent updates )

Availability (a guarantee that every request receives a response about whether it was successful or failed)

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.

**What Is ACID Property Of A System?**

ACID is a acronym which is commonly used to define the properties of a relational database system, it stand for following terms

Atomicity - This property guarantees that if one part of the transaction fails, the entire transaction will fail, and the database state will be left unchanged.

Consistency - This property ensures that any transaction will bring the database from one valid state to another.

Isolation - This property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially.

Durable - means that once a transaction has been committed, it will remain so, even in the event of power loss.

**What Is Load Balancing Fail Over?**

Fail over means switching to another machine when one of the machine fails. Fail over is a important technique in achieving high availability. Typically a load balancer is configured to fail over to another machine when the main machine fails.

To achieve least down time, most load balancer support a feature of heart beat check. This ensures that target machine is responding. As soon as a hear beat signal fails, load balancer stops sending request to that machine and redirects to other machines or cluster.

**What are the differences between continuous integration, continuous delivery, and continuous deployment?**

Developers practicing continuous integration merge their changes back to the main branch as often as possible. By doing so, you avoid the integration hell that usually happens when people wait for release day to merge their changes into the release branch.

Continuous delivery is an extension of continuous integration to make sure that you can release new changes to your customers quickly in a sustainable way. This means that on top of having automated your testing, you also have automated your release process and you can deploy your application at any point of time by clicking on a button.

Continuous deployment goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production.

**What does SOLID stand for? What are its principles?**

S.O.L.I.D is an acronym for the first five object-oriented design (OOD) principles by Robert C. Martin.

S - Single-responsiblity principle. A class should have one and only one reason to change, meaning that a class should have only one job.

O - Open-closed principle. Objects or entities should be open for extension, but closed for modification.

L - Liskov substitution principle. Let q(x) be a property provable about objects of x of type T. Then q(y) should be provable for objects y of type S where S is a subtype of T.

I - Interface segregation principle. A client should never be forced to implement an interface that it doesn't use or clients shouldn't be forced to depend on methods they do not use.

D - Dependency Inversion Principle. Entities must depend on abstractions not on concretions. It states that the high level module must not depend on the low level module, but they should depend on abstractions.

**What Is BASE Property Of A System?**

BASE properties are the common properties of recently evolved NoSQL databases. According to CAP theorem, a BASE system does not guarantee consistency. This is a contrived acronym that is mapped to following property of a system in terms of the CAP theorem:

Basically available indicates that the system is guaranteed to be available

Soft stateindicates that the state of the system may change over time, even without input. This is mainly due to the eventually consistent model.

Eventual consistency indicates that the system will become consistent over time, given that the system doesn't receive input during that time.

**What's the difference between principles YAGNI and KISS?**

YAGNI (You aint gona need it) refers to over analyzing and implementing things that may or may not be needed. Sure algorithmic elegance is nice and all but most situation you dont need it. In general engineering terms you should be carefull not to include your own requirements so that you dont taint your customer needs with your own ideas that end up costing the project with little impact for the client.

KISS (Keep it simple stupid) refers to the fact that easy systems are easier to manage. Keeping things simple is not nesseserily less work (like YAGNI is) since it requires more knowlege to implement. They are sometimes similar but grow from different needs.

YAGNI grows from a too much future anticipation, overzealous workers if you may. KISS is a strategy that tries to counteract human tendency for design creep.