**Cloud And Microservice Interview Question Answer**

# Q1. What is the Rest Principles?

# Ans: There are ****Six Principles****

1. **Client-Server**: Separation of concerns is the principle behind the client-server constraints. By separating the user interface concerns from the data storage concerns, we improve the portability of the user interface across multiple platforms and improve scalability by simplifying the server components.
2. **Stateless**: communication must be stateless, as in the client-stateless-server (CSS) style. Each request from the client to the server must contain all of the information necessary to understand the request. Session state is therefore kept entirely on the client.
3. **Cacheable**: To improve network efficiency, we add cache constraints to form the client-cache-stateless-server style. Cache constraints require that the data respond to a request with the implicit or explicit label as cacheable or non-cacheable. If a response is cacheable, then a client cache is given the right to reuse that response data for later, equivalent requests.
4. **Layered System**: A client cannot ordinarily tell whether it is connected directly to the end server or an intermediary along the way. Intermediary servers may improve system scalability by enabling load-balancing and by providing shared caches. Layers may also enforce security policies.
5. **Code-on-Demand**: REST allows client functionality to extend by downloading and executing code in the form of applets or scripts. Simplifies clients by reducing the number of features required to be pre-implemented. It allows features to download after deployment improves system extensibility.
6. **Uniform Interface**: By applying the software engineering principle of generality to the component interface, the overall system architecture becomes simplified, and the visibility of interactions is improved. Implementations decouple from the services they provide, which encourages independent evolvability. REST defines by four interface constraints: identification of resources, manipulation of resources through representations, self-descriptive messages, and Hypermedia as the engine of application state.  
   **Self-descriptive Messages**: Each message includes enough information to describe how to process the message.  
   **Resource-Based**: Individual resources are identified in requests using URIs as resource identifiers. The resources themselves are conceptually separate from the representations that return to the client.  
   **Manipulation of Resources Through Representations**: When a client represents a resource, including any metadata attached, it has enough information to modify or delete the resource on the server, provided it has permission to do so.  
   **Hypermedia as the Engine of Application State (HATEOAS)**: Clients deliver state via body contents, query-string parameters, request headers, and the requested URI (the resource name). Services provide the state to clients via body content, response codes, and response headers.

## **Q2. What is C-A-P?**

The definition of CAP theorem seems simple and quick. But wait!! What do you mean by Consistency, Availability and Partition tolerance?? Let’s define these terms in a distributed computing environment.

### 1.1. Consistency

A service that is consistent operates fully or not at all. Consistency is ‘C’ in ACID properties in non-distributed systems, as applied to the ideal properties of database transactions. It means that data will never be persisted (rolled back) that breaks certain pre-set constraints.

Consistency, in a distributed environment, means **all client programs who are reading the data from the cluster see the same data at any given point in time**. Two clients fetching data from two nodes should not see different data at all.

### 1.2. Availability

Availability means – **we should be able to retrieve the data that we have stored in the distributed system, no matter what happens inside the cluster**. If we make a request, then we must get a response from the system; even if a node (or many nodes) in the cluster goes down.

### 1.3. Partition Tolerance

Partition Tolerance means that the cluster (as a whole) continues to function even if there is a “partition” (communications break) between two nodes (both nodes are up, but can’t communicate).

No set of failures less than total network failure is allowed to cause the system to respond incorrectly.