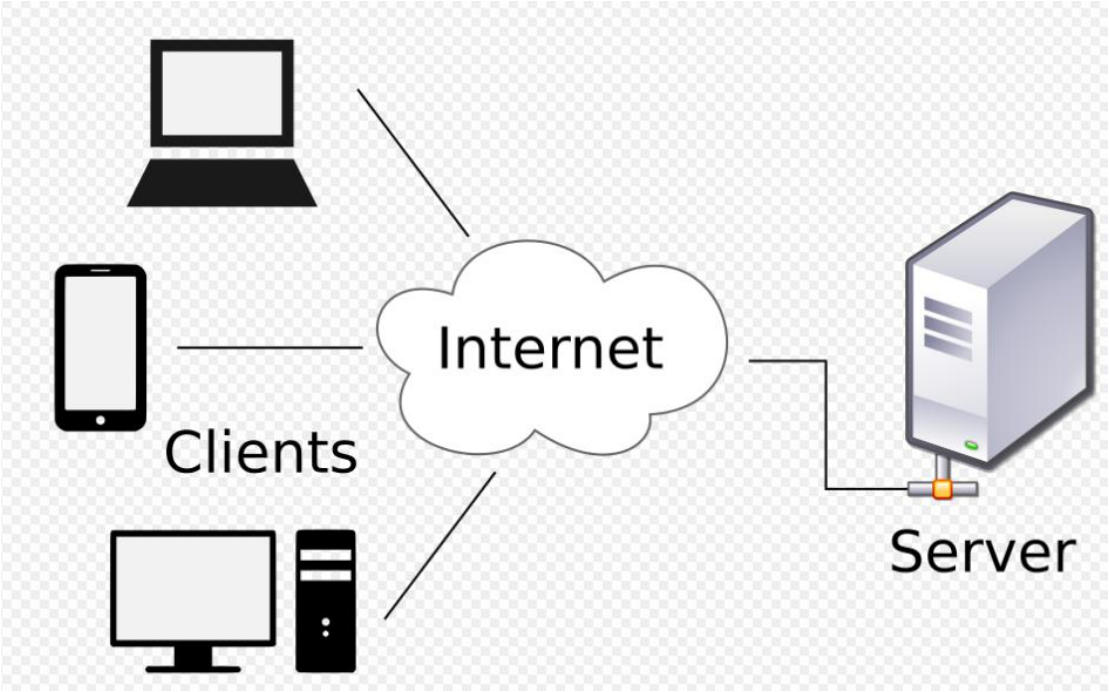
What is Client-Server Architecture?



To understand Client/Server architecture lets take an example of a Restaurant.
When a customer requests an order. Request goes to restaurant's kitchen where food prepared as per demand and served to the customer.

Similarly, Client-Server model works. Where server host runs one or more server programs which share their resources with clients.And client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.
Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

DevOps Tools which works on Client/Server Architecture?

Popular tools as Chef, Puppet and SaltStack works on this architecture model.
Lets dive little deeper into this. A web UI or CLI tool(Client) is used to issue commands(requests). These commands received by Server which is responsible for executing commands and storing the state of the requesting system.
Server communicates with agents running on servers to execute commands.
There are many drawbacks/limitations using tools based on client/server architecture model. Such as :-

1. Extra software need to install and run on every server which required to be configured
2. Extra server required for High availability
3. Extra maintenance required on this server to upgrade/backup etc..

Its very difficult to maintain these extra resources. Also its difficult to identify fail over cases, cause of failure.

There are Tools which works on Client Only architecture.
Ansible, CloudFormation and Terraform use a client-only architecture.
For CloudFormation AWS handles all the server details so that an end user only have to think about the client code.
The Ansible client works by connecting directly to servers over SSH.
Terraform uses cloud provider APIs to provision infrastructure