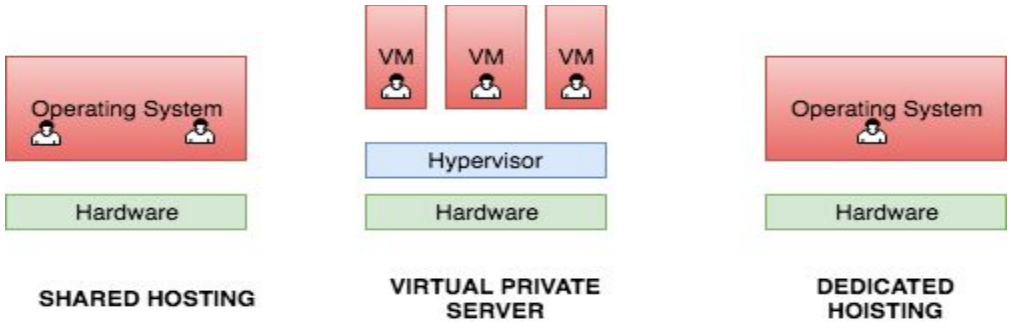
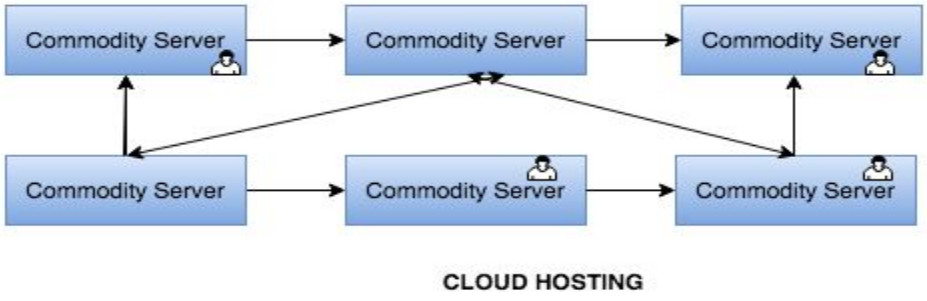
# SCALABILITY

- **WEB HOSTING**
  - ***Shared Hosting***
    → Multiple users are hosted on the same Operating System and hence share server resources. User does not have dedicated resources allocated to it.
  - ***VPS***(Virtual Private Server) or ***VDS***(Virtual Dedicated Server) or ***Private Cloud***.
    → Several virtual machines are hosted on the same physical server using a hypervisor and each user has a dedicated VM allocated to it. But users may share the physical space. Eg; Godaddy.
  - ***Dedicated Hosting***
    → Each user has his own dedicated Operating system hosted on a dedicated physical server.
  - ***Cloud Hosting or Public Cloud***
    → Each user has dedicated resources allocated it it on a network of commodity servers. Eg: Amazon EC2

    Cloud hosting has advantages over VPS
    1. *Availability* → When a hardware failure occurs then server is allocated on different physical machine.
    2. *Secure* → Each user has dedicate resources
    3. *Cost effective* → Pay as you use
    4. *Scalable* → As new resources can be provisioned from the network of resources when the existing system reaches its optimum capacity.




.

# SCALABILITY

**LINKS**

http://blog.webspecia.com/web-hosting/difference-between-cloud-vps
https://www.inmotionhosting.com/support/website/difference-between-shared-vps-dedicated-hosting

- **MULTI-TENANCY**

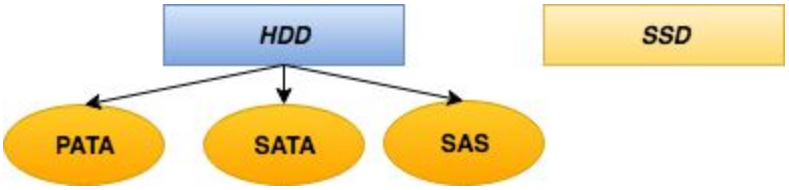
  → Is a software architecture in which a *single instance* of software which runs on a single server and *servers multiple tenants*. Eg Salesforce.com

  → But multi-tenant architecture has security compliance issues as resources and data are shared and hence we have the concept of ***multi-instance.*** Eg: AWS. Salesforce also acknowledges this and hence has introduced the concept of ***Superpods.***

# SCALABILITY

- **VERTICAL SCALING**
  → Adding more resources to the same physical machine.
  → But there is a ceiling because of physical limitations.

  - **CPU**
    → Adding more cores which perform tasks in parallel
    → CPU performs ***scheduling*** of processes and ***time slicing***.
  - **DISK**
    - **HDD**
      - PATA (Parallel ATA) → Legacy
      - SATA → General purpose desktops and laptops. Spins at *7200 RPM*
      - SAS (Serial Attached SCSI) → Generally used for *databases* (to read and write quickly) and *high availability servers* as it is 2x time faster than SATA. Spins at *15000 RPM.*
    - **SSD** (Solid State Disk) → Highly expensive




  - **RAID**
    *Redundant Array of Inexpensive DIsks*. A ***storage virtualization*** technology that combines multiple data drive components into one or more logical units for the purpose of *data redundancy* and *performance improvement*.

- **HORIZONTAL SCALING**
  - Adding *more nodes* with less power and *spreading out the resources* using commodity hardware.
  - Horizontally scaled architecture is more *cost-effective* and more *resilient to failures.*
  - Horizontally scaled architecture required the use a of load balancer.

# SCALABILITY

- **LOAD BALANCING**
  - Balancing the load across several backend servers.
  - Client need to hit only *one IP Address* and the load balancer uses several strategies to balance the load across several servers.
  - The load balancer may also act as a reverse proxy.

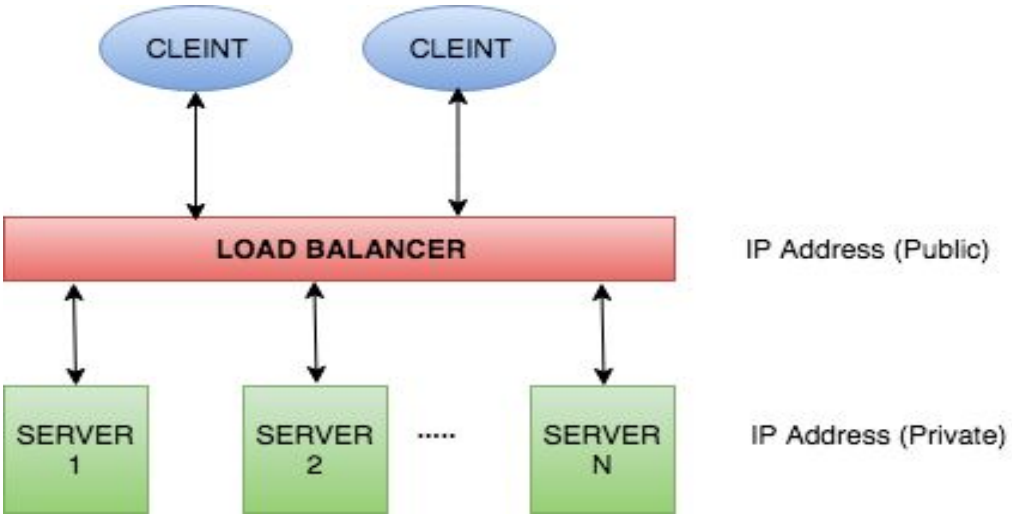
***Strategies for first request***

- Round Robin → (*Downside*: A user may be a power user and using more server resources.)
- Least loaded server
- The load balancer can be a *load balancing DNS server* like ***BIND*** which maps a web address to several IP addresses and serve the IP addresses in a *round robin* strategy or *geographical* strategy.

***Strategies for subsequent requests***

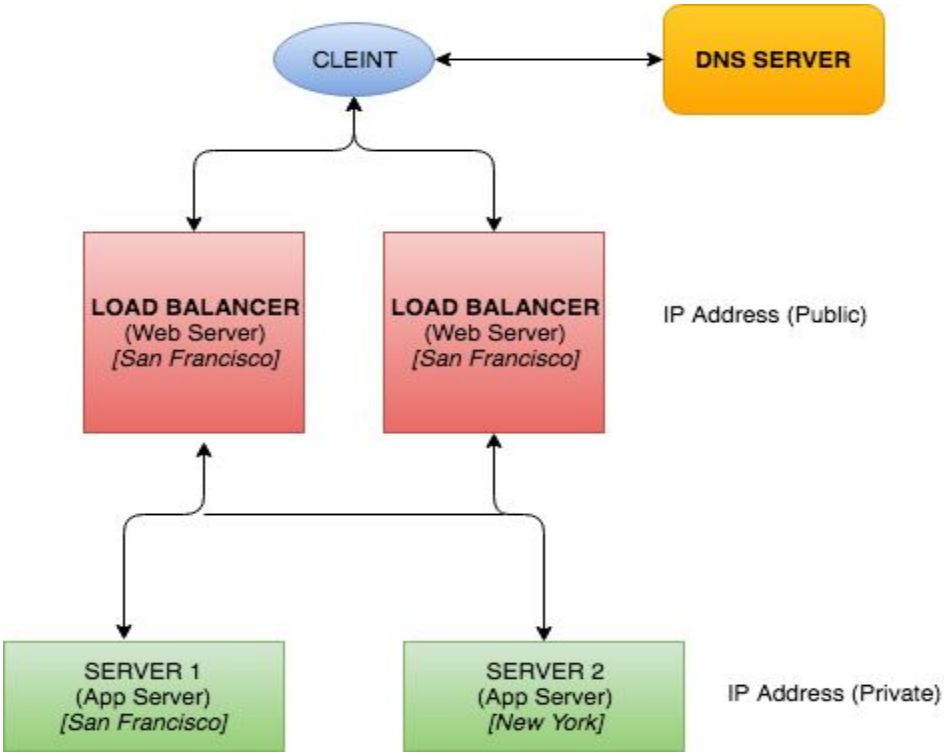
- Cookie-based routing with a specified ***TTL****(Time to Live).*

**`ARCHITECTURE 1**




**ARCHITECTURE 2**

# SCALABILITY




- The load balancer *does not* maintain any state and makes it easy for routing.
- Typically, OS and browser caches avoid subsequent DNS lookups but are constrained by ***TTL (Time to live).***
- The client-server communication is based on *cookie*.

**LINKS**

https://www.digitalocean.com/community/tutorials/how-to-configure-dns-round-robin-load-balancing-for-high-availability

# SCALABILITY

- **SESSION MANAGEMENT**

  A user session is managed by the server by:

  1. ***Cookie based***
     - ➢ Server sends a cookie (JSessionID) to the browser in response to a request.
     - ➢ The browser maintains it and sends the cookie in subsequent requests to the server
  2. ***URL rewriting.****(JavaEE server specific)*
     - ➢ If cookie based session management is not supported, the url rewriting is used.
     - ➢ The (JSessionID) is added as a url parameter as follows.

       *http://www.abc.com;jsessionid=123xyz*
  3. ***Session Storage*** *(Introduced in HTML5)*
     - ➢ Stores the user session data in the browser and gets cleared when the session is closed.
     - ➢ The session data is not sent back to the server.
     - ➢ Has a higher storage limit than cookie.
  4. ***Local Storage*** *(Introduced in HTML5)*
     - ➢ Stores the user data in the browser and can span across sessions.
     - ➢ Has a higher storage limit than session storage.


| | Cookies | Local Storage | Session Storage |
|---|---|---|---|
| **Capacity** | 4kb | 10mb | 5mb |
| **Browsers** | HTML4 / HTML 5 | HTML 5 | HTML 5 |
| **Accessible from** | Any window | Any window | Same tab |
| **Expires** | Manually set | Never | On tab close |
| **Storage Location** | Browser and server | Browser only | Browser only |
| **Sent with requests** | Yes | No | No |

**LINKS**

***http://conceptpill.com/how-session-management-works-in-tomcat-web-apps/***
***https://scotch.io/@PratyushB/local-storage-vs-session-storage-vs-cookie***
***https://www.youtube.com/watch?v=AwicscsvGLg***

# SCALABILITY

- **SESSION MANAGEMENT-(LOAD BALANCER)**

  A load balancer manages user sessions in the following manner:

  1. ***Sticky session****(Session affinity)*
     - ➢ The *load balancer* can use several strategies to identify to which server the subsequent user requests should be directed to.
       - IP based routing
       - (User Session → Server) mapping.
     - ➢ The user sessions are maintained on the dedicated server,
  2. ***Multicast***
     - ➢ *Each server* in the cluster *maintains a replica* of the user session.
     - ➢ Most in-efficient as it utilizes more server resources.
  3. ***Shared Persistence***
     - ➢ A *shared storage* like file server, database or memcached can is used for storing session data.
     - ➢ Additional backup can be provided to avoid a single point of failure.

  **LINKS**

  ***https://devcentral.f5.com/articles/sessions-sessions-everywhere***