Nginx

High Performance Load Balancer, Web Server & Reverse Proxy

Techognite
Complete coverage on Tech

Agenda

- Introduction
- Architecture
- Configurations
- Server, Listener, Location
- Mime type
- gzip
- Reverse Proxy
- Load Balance
- LB Algorithms

Introduction

- Nginx is an Open-source web server and uses a non-threaded, event-driven architecture.
- NGINX, pronounced like “engine-ex”
- Load Balancer
- Reverse Proxy Server
- Content Cache

Basic Commands

- apt-get install nginx #Install Nginx on Ubuntu
- service nginx start #Start Nginx server
- service nginx stop #Stop the server
- service nginx restart #Stop and start the server
- service nginx status #Return the current status
- nginx –t #Verify the configurations

Architecture

- Nginx architecture consists of a master and workers process.
- Master process performs the privileged operations such as reading configuration and binding to ports.
- One master process can control multiple worker processes.
- The worker processes handle network connections, read and write content to disk, and communicate with other servers.
- Each worker process is a single-threaded.
- https://cdn.nginx.com/wp-content/uploads/2015/06/Screen-Shot-2015-06-08-at-12.36.30-PM.png

Architecture

- The worker process waits for events on the listen and connection sockets
- Any Event on listen socket creates a new connection.
- **worker_processes** directive Define the number of worker process should run.
- **worker_connections** maximum number of simultaneous connections for each worker process
- NGINX is single threaded, so best practice is total number of worker_processes should equivalent to number of core in the system.

Understand Nginx Configurations

1. /etc/nginx/nginx.conf #Location of Nginx configuration file.
2. worker_processes auto # No of worker process
3. pid /run/nginx.pid; #Location of Process Id
4. worker_connections 768; #No of connections per Worker Process
5. access_log /var/log/nginx/access.log; # Access Log
6. error_log /var/log/nginx/error.log; # Error Log
7. gzip on; # compression
8. include /etc/nginx/conf.d/*.conf; # To include any file

Basic Nginx Configurations

```
server {
        listen          80;
        server_name     _;
        location / {
                root    /var/www/html/;
                index   index.html;
        }
}
```

Server Context Example

```
server {
        listen                  80;
        server_name             tech.com;
        return                  200 "HELLO FROM TECH WORLD";
}
server {
        listen                  80 default_server;
        server_name             hello.com;
        return                  200 "HELLO WORLD!";
}
```

Server Context Configurations

- listen 80 default_server; #Server Port
- server_name _; #Server Name like tech.com
- root /var/www/html; #Root location of content
- index index.html; #Index file name in root location
- location / { } #specify action on specific location
- return 200 "HELLO WORLD!"; # Return response

Server Context listen Directive

- Listen 192.168.125.21:80; #from192.168.125.21 on 80 Port
- listen 127.0.0.1; # by default port :80 is used
- listen *:81; # All IP on 80 Port
- listen 81; # by default all ips are used
- listen localhost:80; # from Localhost
  But hostname may not be resolved upon nginx's launch
- Listen [::]:80; # IPv6 addresses
- listen [::1]; # IPv6 addresses

Multi Server Context

In case Multi server configuration:

- Server listing on IP:port, with a matching server_name directive;
- Server listing on IP:port, with the default_server flag;
- Server listing on IP:port, first one defined;
- If there are no matches, refuse the connection

Multi Server Context: wildcard

When there is ambiguity, nginx uses the following order:

```
server_name     abc.co, www.abc.co;          # exact match
server_name     *.abc.co;                    # wildcard matching
server_name     abc.*;                       # wildcard matching
server_name     ~^[0-9]*\.abc\.co$;          # regexp matching
```

- Exact name;
- Longest wildcard name starting with an asterisk.
- Longest wildcard name ending with an asterisk.
- First matching regular expression (in the order of appearance in the configuration file).

Multi Server Context: wildcard

Nginx will store 3 hash tables:

exact names, wildcards starting with an asterisk, and wildcards ending with an asterisk. If the result is not in any of the tables, the regular expressions will be tested sequentially.

So if define server name using wildcard, it will be a bit slower than the exact match.

Location Context Example

```
location /tech {
        root    /var/www/html/;
        index   index.html;
}

location /hello{
        root    /var/www/website/;
        index   index.html;
}
```

Location Context Configurations

- root /var/www/html; #Root location of content
- index index.html; #Index file name in root location
- return 200 "HELLO WORLD!"; # Return response
- autoindex on; #Index the content
- default_type text/html; # define mime types
- try_files $uri /index.html =404; # Try multiple paths,
- proxy_pass http://_server; # used in Reverse Proxy, LB

Modifier Location Context Configurations

• =	- Exact match	location = /match { }
• ^~	- Preferential match	location ^~ /match0 { }
• ~ && ~*	- Regex match	location ~* /match[0-9] { }
• no modifier	- Prefix match	location /match { }

- Nginx will first check for any exact matches.
- If it doesn't find any, it'll look for preferential ones.
- If this match also fails, regex matches will be tested
- If everything else fails, the last prefix match will be used.

Prefix Match Location Context

When no modifier is specified, the path is treated as prefix

```
location /tech {
        # ...
}
```

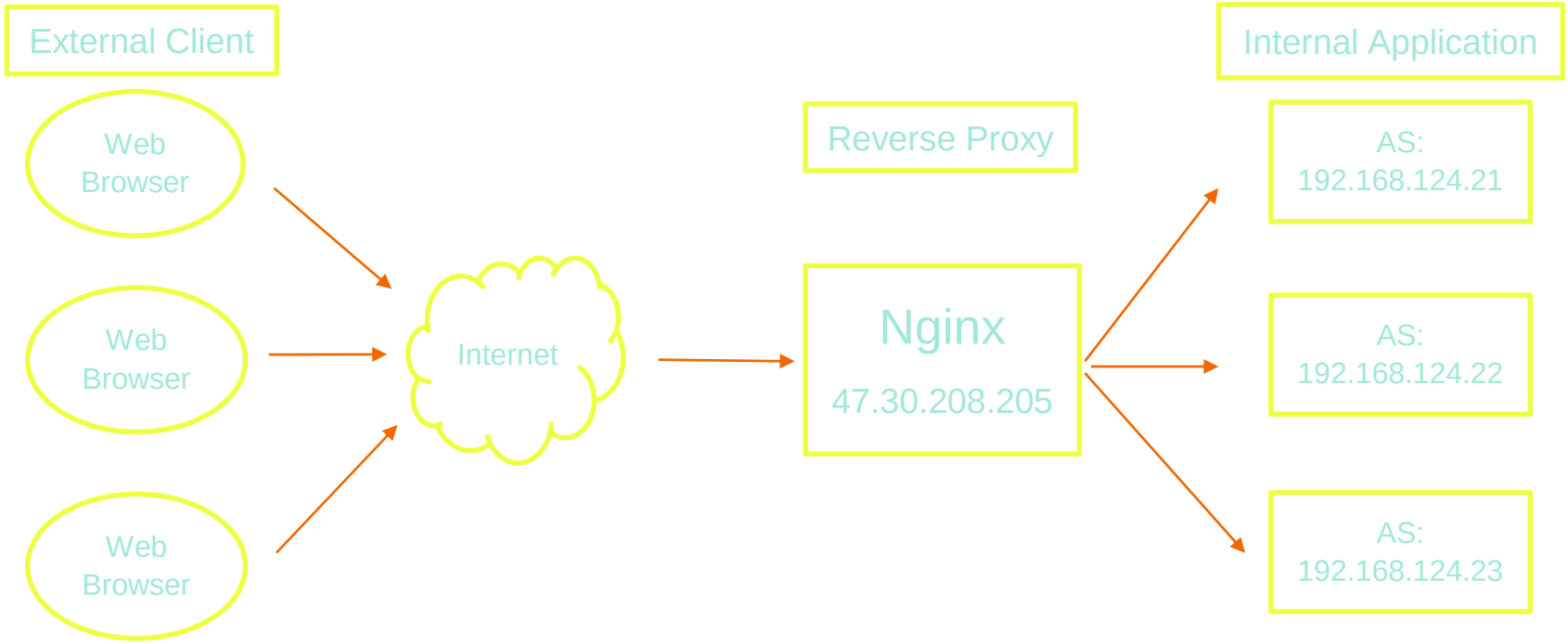
Valid match:

- /tech
- /tech/java/index.html
- /tech123
- /techworld

Nginx Reverse proxy

- Sits between internal applications and external clients, forwarding client requests to the appropriate server.
- Retrieves resources on behalf of a client from one or more servers.
- Does the request Routing
- Can prevent sever from DOS, Web based attacks.

Nginx Reverse proxy



Reverse proxy Example

```
location /admin{
        proxy_pass      http://192.168.124.22:8180/admin;
}

location /user{
        proxy_pass      http://192.168.124.21:8280/user;
}
```

http://47.30.208.205/user -> http://192.168.124.21:8280/user

http://47.30.208.205/admin -> http://192.168.124.22:8180/admin

Reverse proxy Configuration

- proxy_pass http://18.191.4.181:8180/admin;
  #Pass the request to specified Server
- proxy_set_header X-Real-IP $remote_addr;
  #Set the client IP in request header with name X-Real-IP

Client IP
220.220.220.1

Nginx IP
220.220.220.100

Application Server
Receive: 220.220.220.100

Reverse proxy Configuration

- proxy_set_header host $host;
  #Set the host Name in request header with name host

Server Name (Domain Name)	Nginx IP	Application Server
www.admin.com	220.220.220.100	220.220.220.200
www.user.com	220.220.220.100	220.220.220.200

Reverse proxy Configuration

```
location /admin{
        proxy_pass              http://192.168.124.22:8180/admin;
        proxy_set_header        X-Real-IP       $remote_addr;
        proxy_set_header        host            $host;
}
```

http://47.30.208.205/admin -> http://192.168.124.22:8180/admin
with client Real IP and Host Name

Request Redirection Configurtation

```
server {
        listen 80;
        server_name domain_name;
        return 301 https://$server_name$request_uri;
}
```

Every request coming to the this server on port 80 with **http**, will be redirected to **https** with same server name

http://192.168.125.32 will redirected to https://192.168.125.32

MIME type Configurations

- MIME full form **Multipurpose Internet Mail Extensions**
- **content-type** response header parameter is used to store this information.
- Example: Content-type= text/html
- Browser use the content type value to open the file with the proper extension/plugin
- include /etc/nginx/mime.types; #Include mime type files
- default_type application/octet-stream; # define default mime type

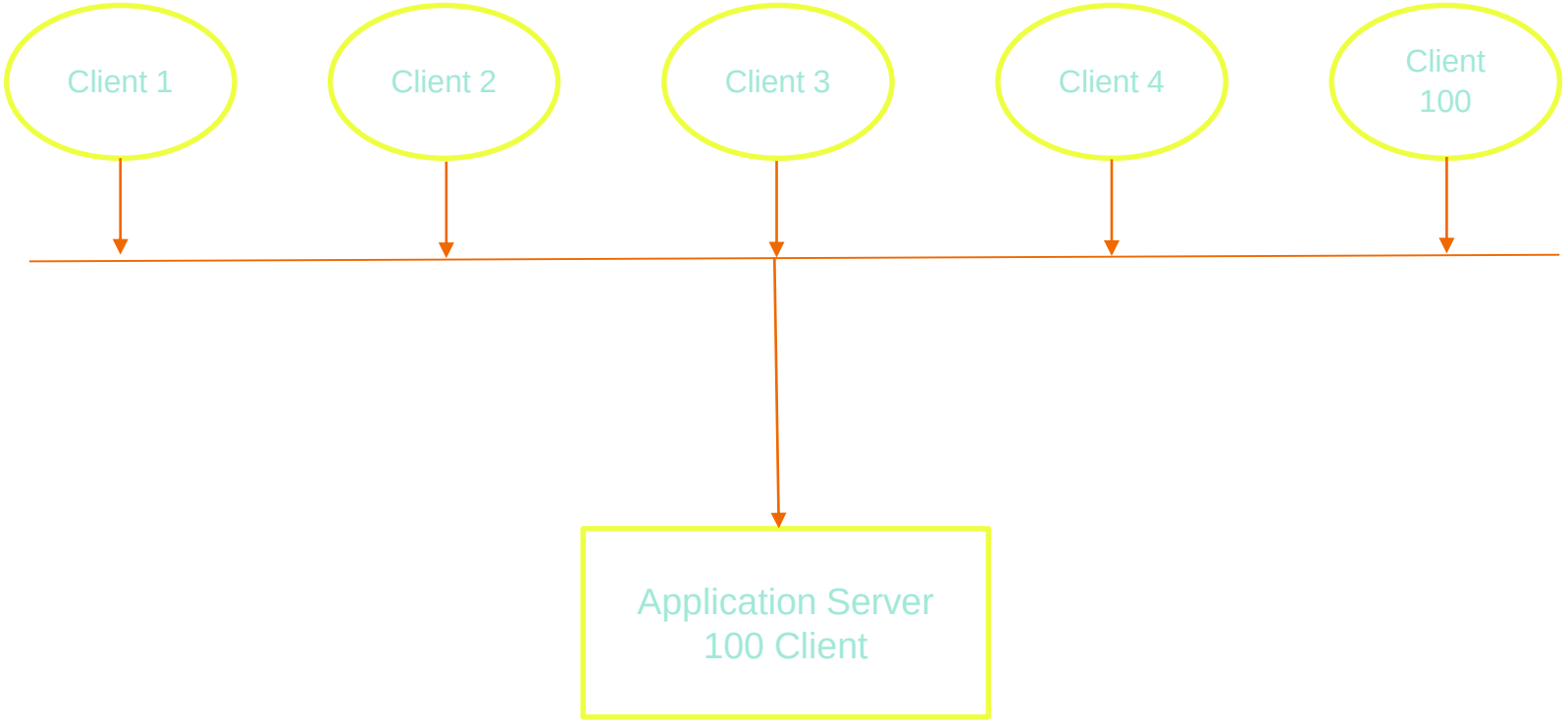
GZIP

- GZIP is used for response compression.
- Used for performance upgrade.
- gzip_comp_level, which is the level of compression and ranges from 1 to 10. Generally,
- Value of gzip_comp_level should not be above 6—the gain in terms of size reduction is insignificant, as it needs a lot more CPU usage. gzip_types is a list of response types that compression will be applied on.

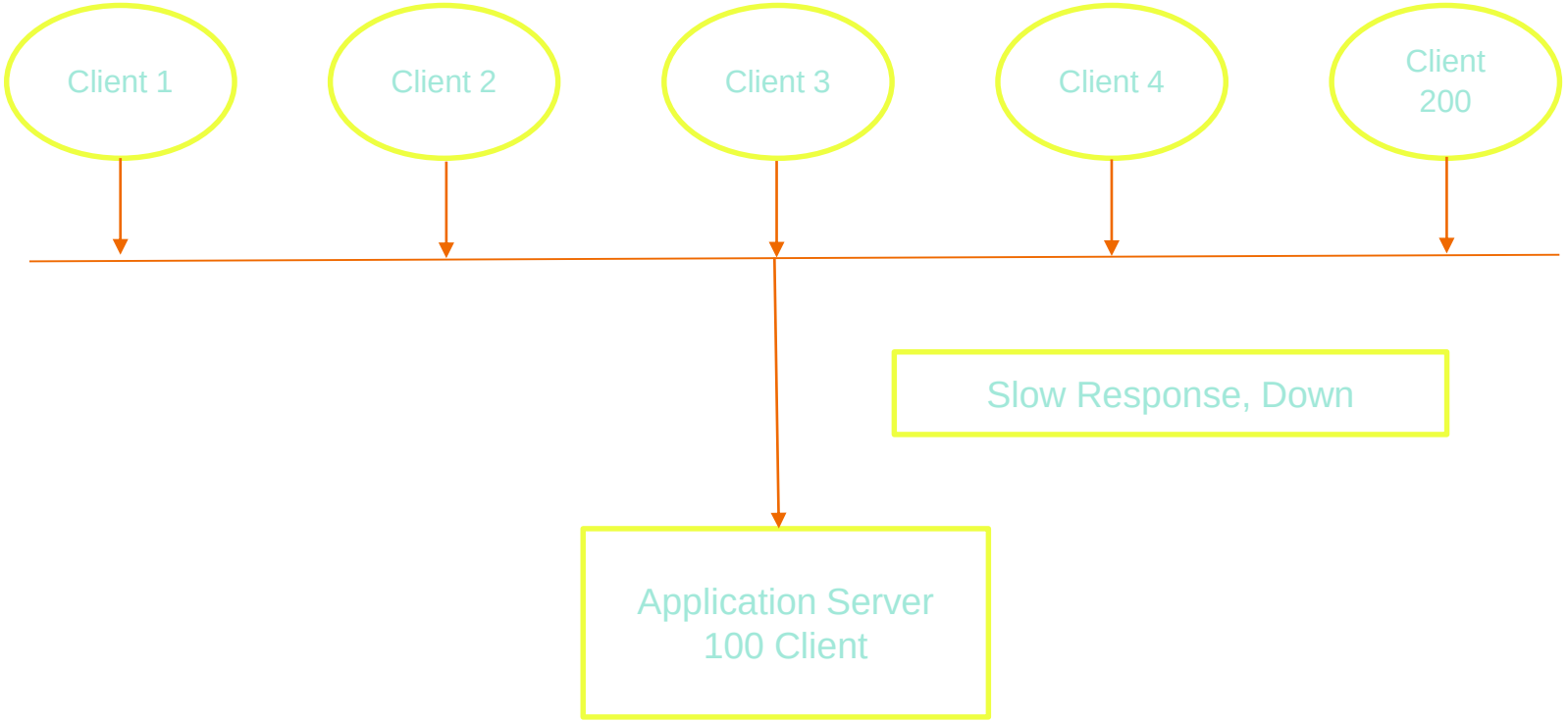
Nginx Load Balancer

A load balancer is a server that distributes network or application traffic across a cluster of servers. Load balancing improves responsiveness and increases availability of applications.

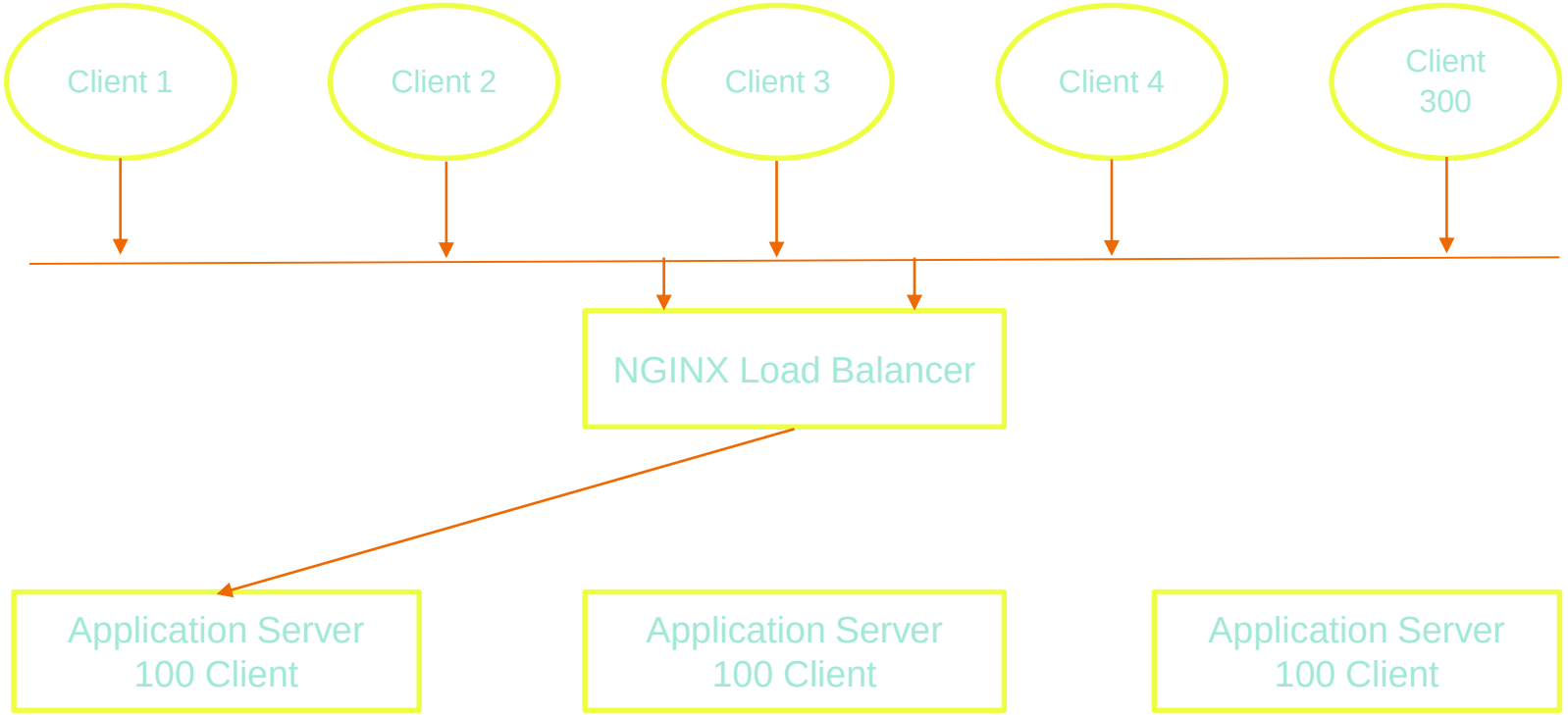
Load Balancer



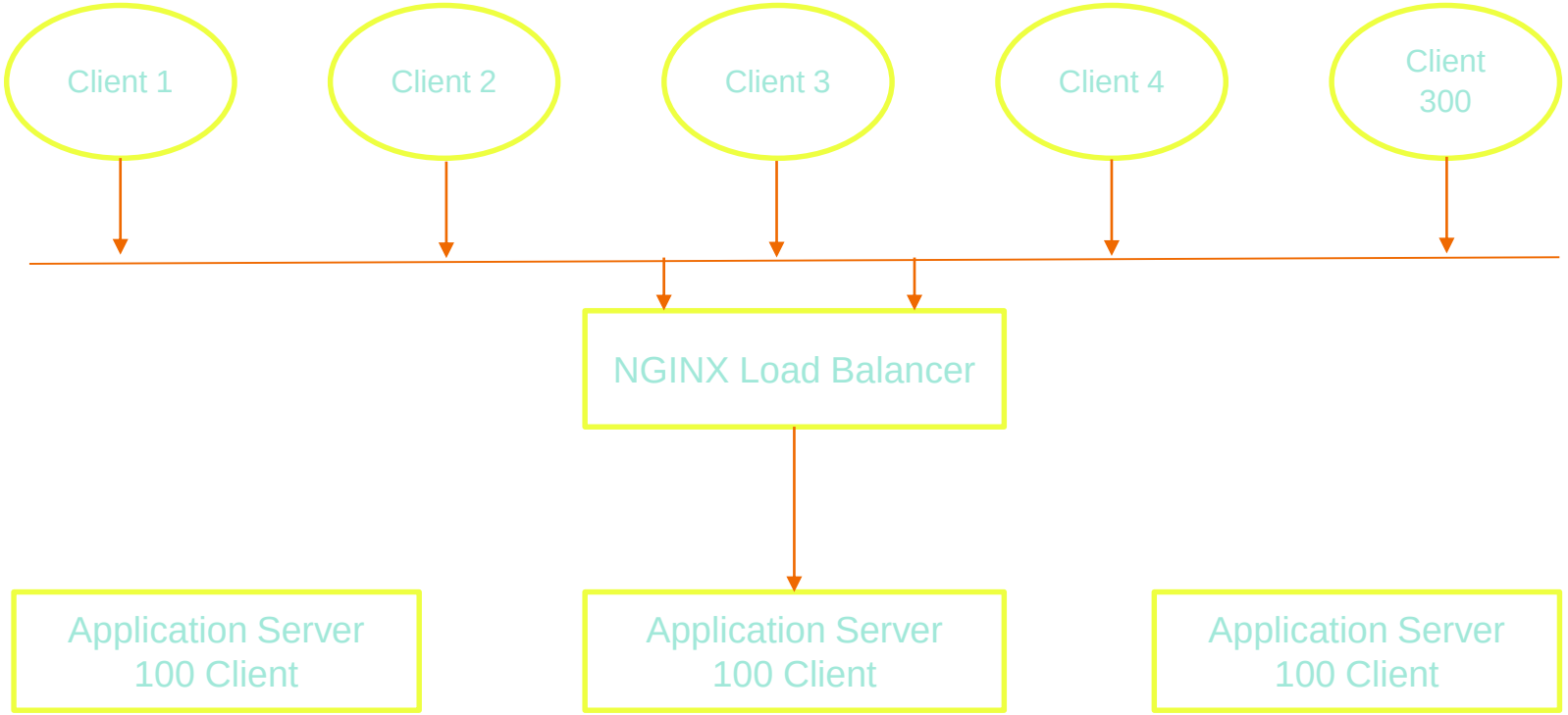
Load Balancer



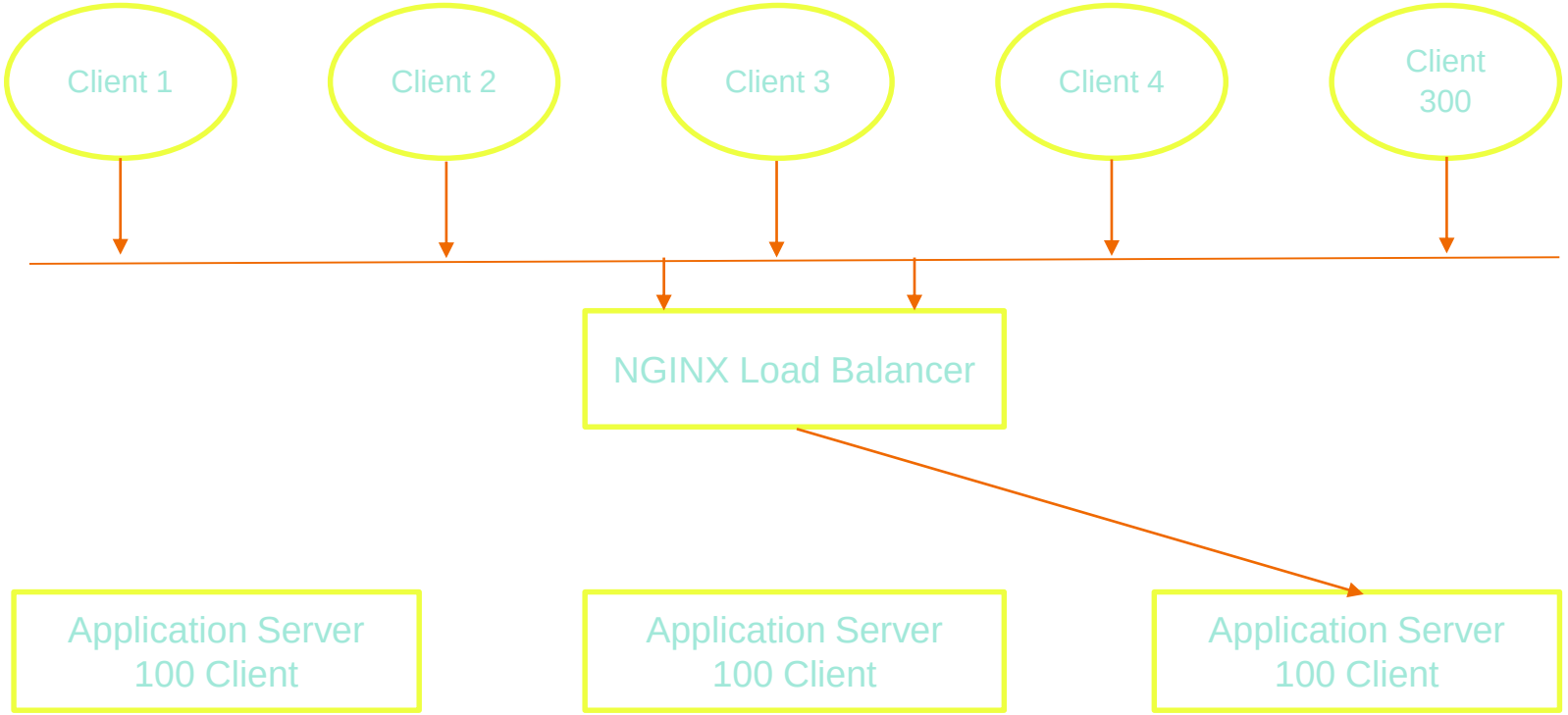
Load Balancer



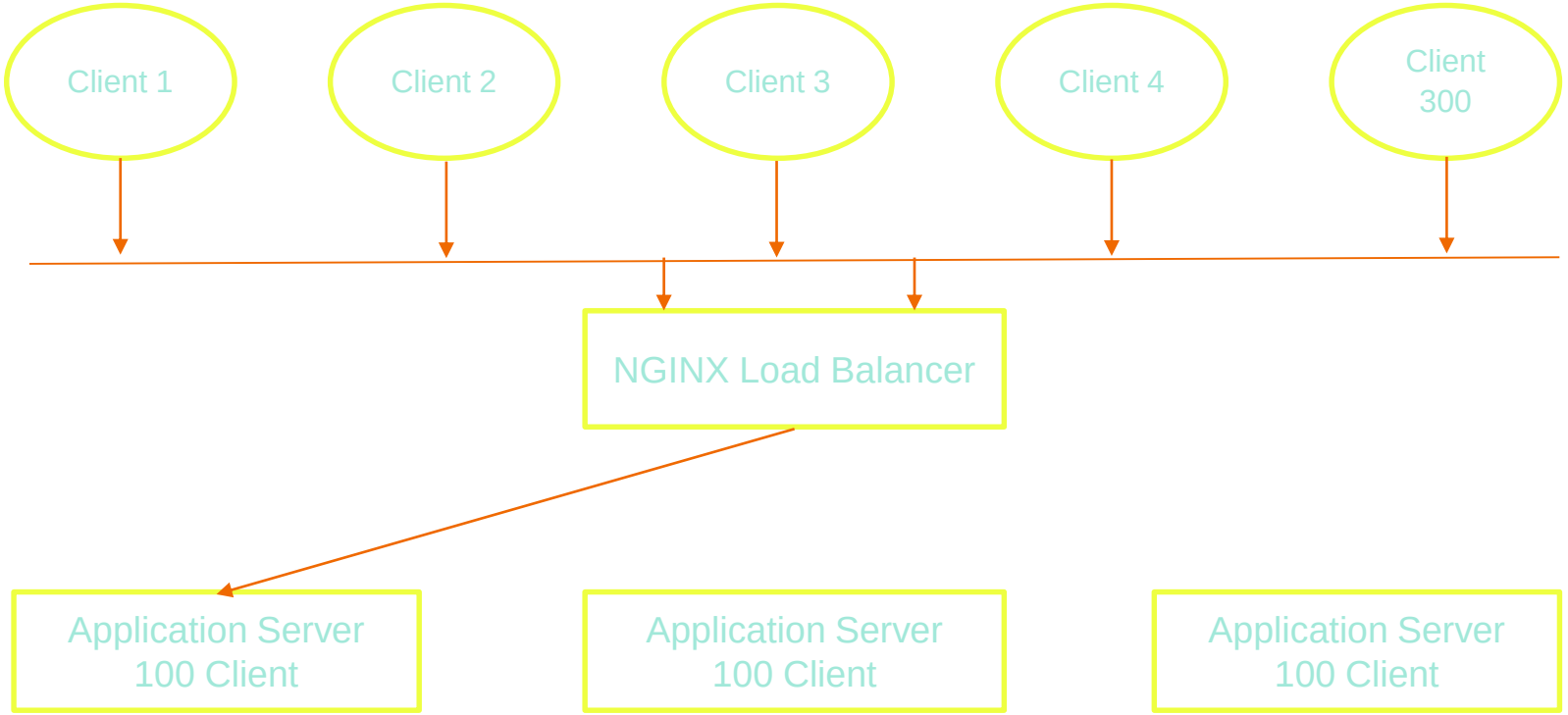
Load Balancer



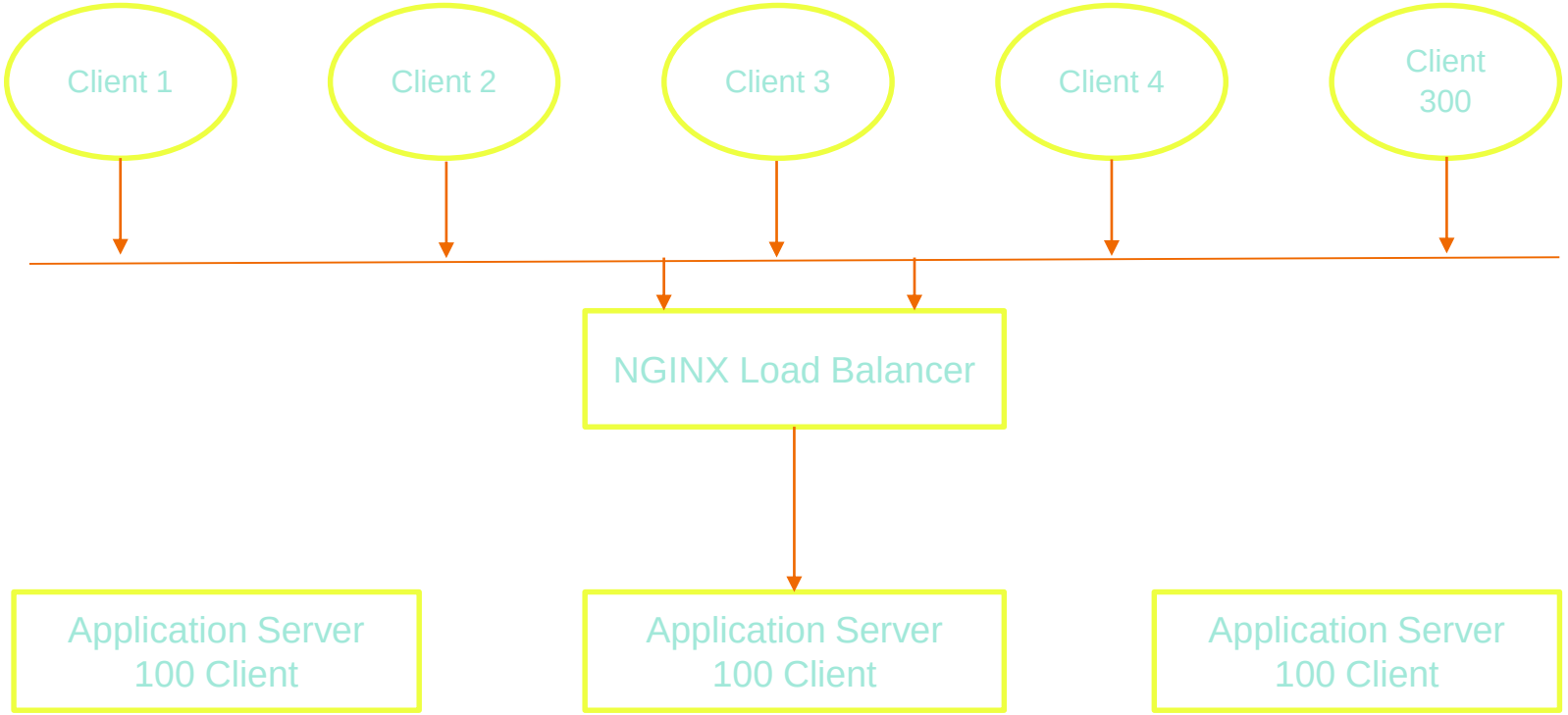
Load Balancer



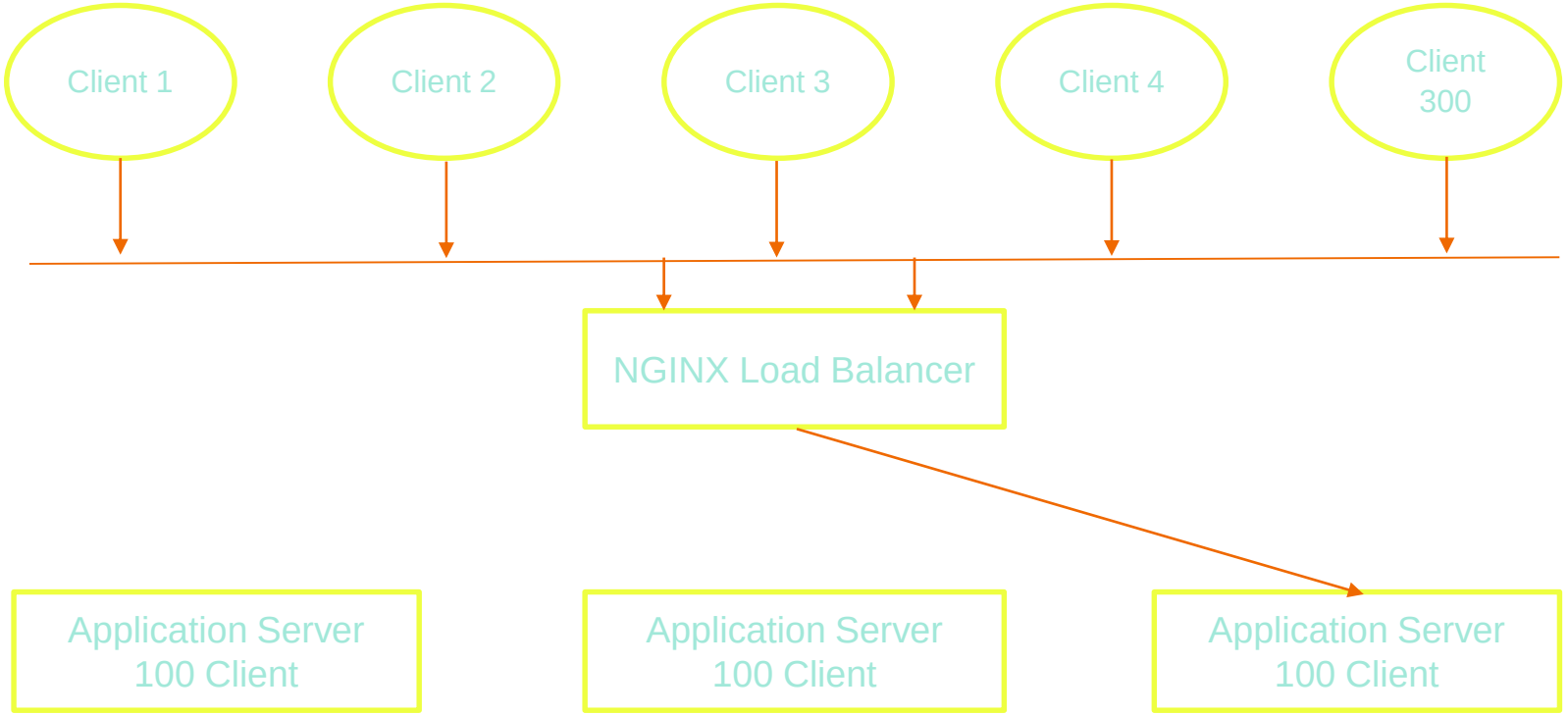
Load Balancer



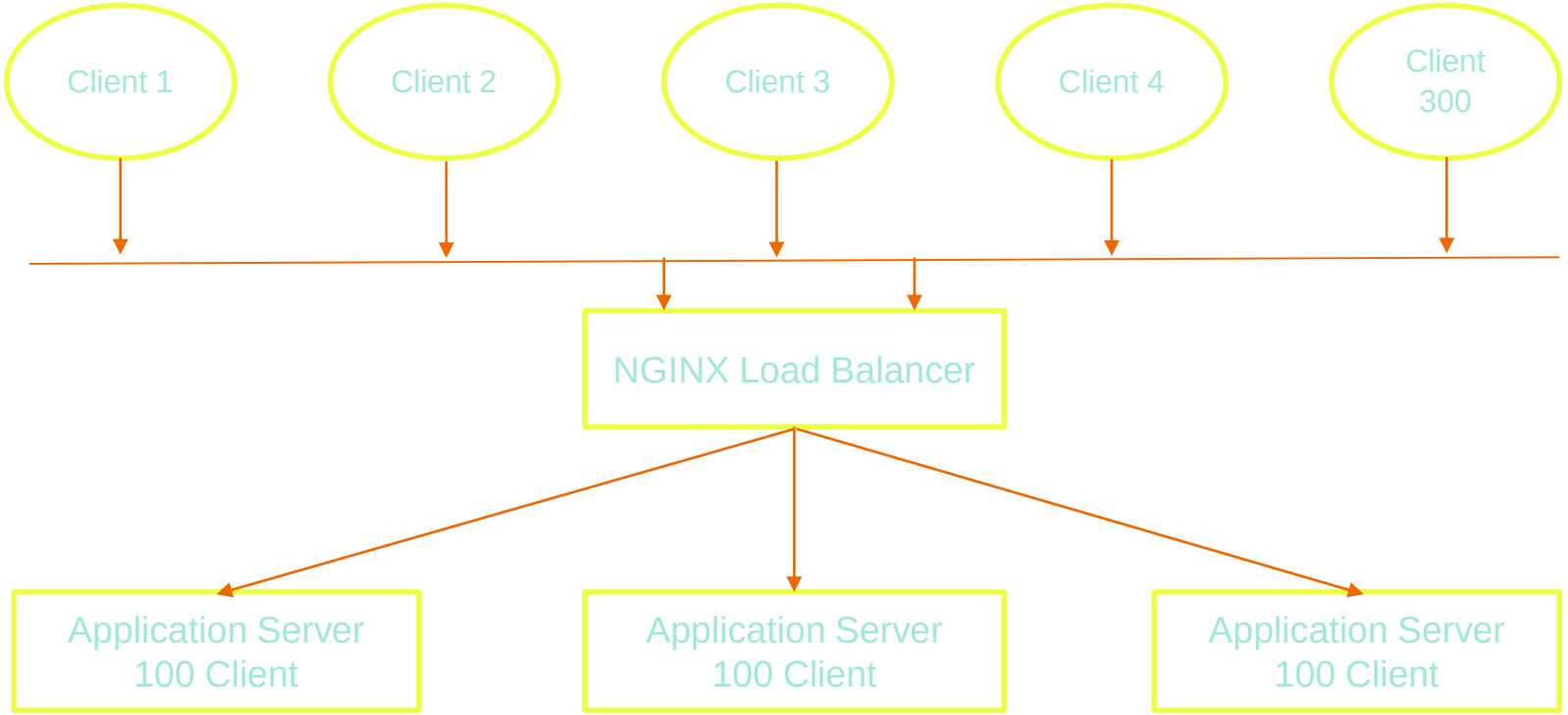
Load Balancer



Load Balancer



Load Balancer



Nginx Load Balancer Configuration

```
server {
        listen          80;
        server_name     _;
        location /admin {
                proxy_pass      http://remote_application_server;
        }
}
upstream remote_application_server {
        server 18.191.4.181:8180;
        server 18.191.4.182:8180;
        server 18.191.4.183:8180;
}
```

Weight: Nginx Load Balancer

When the available resources between pool of hosts are not equal, we define weight to favour some servers over others.

```
upstream remote_application_server {
        server 18.191.4.181:8180 weight=4;
        server 18.191.4.182:8180 weight=1;
}
```

Health Check: Nginx Load Balancer

Nginx implement passive server health checks to know which servers are available from pool of servers.

```
upstream remote_application_server {
        server 18.191.4.181:8180  max_fails=3 fail_timeout=30s;
        server 18.191.4.182:8180 max_fails=3 fail_timeout=30s;
}
```

Health Check: Nginx Load Balancer

- **max_fails**: Number of fail attempt after which server will assume down and load will not share to this server.
- **fail_timeout**, which also defines how long the server should be considered failed.
- If **max_fails** is set to a value greater than 1 then subsequent fails must happen within a specific time frame for the fails to count.
- By default the **fail_timeout** is set to 10 seconds.
- Once Server is marked as failed, next request will not be sent to this server
- If Next health check start returning successful, the server is again marked live and included in the load balancing as normal.

Round-robin Algorithm: Nginx Load Balancer

- Round-robin:
- Round-robin scheme each server is selected in turns according to the order.
- Request goes to each server one by one.
- Load balancing with nginx uses a round-robin algorithm by default

Least connections Algorithm: Nginx Load Balancer

Least connections:

- This method directs the requests to the server with the least active connections at that time
- Should used where requests takes longer to complete.

```
upstream remote_application_server {
        least_conn;
        server 18.191.4.181:8180;
        server 18.191.4.182:8180;
}
```

IP hashing Algorithm: Nginx Load Balancer

IP hashing:

- IP Hashing techniques first select the sever and send all the subsequent request to the same server coming from the same IP.
- Uses the visitors IP address as a key to determine which host should be selected to fulfill the request

Condition for IP hashing.

- Client IP should same for next request.
- Server should be available, if server is down same request can be sent any of the server mention in upstream pool.

IP hashing Algorithm: Nginx Load Balancer

Configuration:

```
upstream remote_application_server {
        ip_hash;
        server 18.191.4.181:8180;
        server 18.191.4.182:8180;
        server 18.191.4.183:8180;
}
```

Backup Server: Nginx Load Balancer

- We can mark server as down, while performing maintenance. This time request will not send to this server.
- Marking a server backup will be used, when all other servers are unavailable.

```
upstream remote_application_server {
        server 18.191.4.181:8180;
        server 18.191.4.182:8180 down;
        server 18.191.4.183:8180 backup;
}
```


Ask Your Query