1. What is the advantage of using a “reverse proxy server”?

It is used in Content Delivery Networks. One could have multiple nodes with frontend servers acting as a reverse proxy to a single backend while the frontend could cache especially static content. And of course it is also possible to have multiple backends and to distribute the load among them using a single frontend, this would be a way of load balancing.

2. Why and where Nginx is a better choice than apache.

A. **Nginx is lightweight**

Nginx is one of the lightweight web servers out there. It has small footprints on a system compared to Apache which implements a vast scope of functionality necessary to run an application.

Because Nginx puts together a handful of core features, it relies on dedicated third-party upstream web servers such as an Apache backend, FastCGI, Memcached, SCGI, and uWSGI servers or application server, i.e language specific servers such as Node.js, Tomcat, etc.

Therefore its memory usage is far better suited for limited resource deployments, than Apache.

B. Nginx is can be used as a Load balancer

To realize high performance and uptime for modern web applications may call for running multiple application instances on a single or distributed HTTP servers. This may in turn necessitate for setting up load balancing to distribute load between your HTTP servers.

Today, load balancing has become a widely used approach for optimizing operating system resource utilization, maximizing flexibility, cutting down latency, increasing throughput, achieving redundancy, and establishing fault-tolerant configurations – across multiple application instances.

Nginx uses the following load balancing methods:

- round-robin (default method) – requests to the upstream servers are distributed in a round-robin fashion (in order of the list of servers in the upstream pool).
- least-connected – here the next request is proxied to the server with the least number of active connections.
- ip-hash – here a hash-function is used to determine what server should be selected for the next request (based on the client's IP address).
- Generic hash – under this method, the system administrator specifies a hash (or key) with the given text, variables of the request or runtime, or their combination. For example, the key may be a source IP and port, or URI. Nginx then distributes the load amongst the upstream servers by generating a hash for the current request and placing it against the upstream servers.
- Least time (Nginx Plus) – assigns the next request to the upstream server with the least number of current connections but favors the servers with the lowest average response times.

C. Nginx is an Excellent Frontend Proxy

One of the common uses of Nginx is setting it up as a proxy server, in this case it receives HTTP requests from clients and passes them to proxied or upstream servers that were mentioned above, over different protocols. You can also modify client request headers that are sent to the proxied server, and configure buffering of responses coming from the proxied servers.

Then it receives responses from the proxied servers and passes them to clients. It is much easier to configure as a proxy server compared to Apache since the required modules are in most cases enabled by default.

Reference: https://www.tecmint.com/why-nginx-better-than-apache/

3. What are worker nodes and worker connections? How to calculate the max server capacity using the above two?

`worker_processes` – The number of NGINX worker processes (the default is 1). In most cases, running one worker process per CPU core works well, and we recommend setting this directive to `auto` to achieve that. There are times when you may want to increase this number, such as when the worker processes have to do a lot of disk I/O.

`worker_connections` – The maximum number of connections that each worker process can handle simultaneously. The default is 512, but most systems have enough resources to support a larger number. The appropriate setting depends on the size of the server and the nature of the traffic, and can be discovered through testing.

The max capacity calculation would be as follows.

The worker_connections and worker_proceses from the main section allows you to calculate maxclients value:

max_clients = worker_processes * worker_connections

In a reverse proxy situation, max_clients becomes

max_clients = worker_processes * worker_connections/4

4. From what directory will NGINX automatically load server (virtual host) configurations when using the default /etc/nginx/nginx.conf configuration?

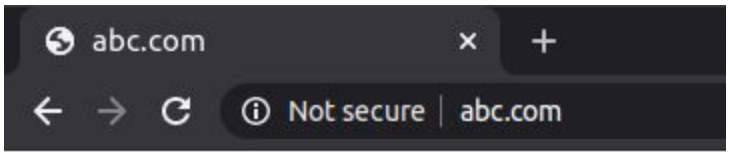
   /etc/nginx/sites-enabled

5. How to configure different log_format for different “location” block/directive?

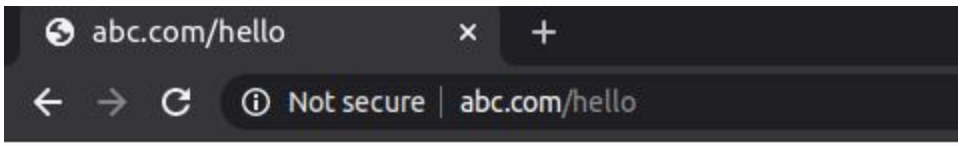
Reference:
https://www.tecmint.com/configure-custom-access-and-error-log-formats-in-nginx/

6. Host a site ABC.COM
    1. Create an index page and a fail-safe page. If a page for URI is not available, the fail-safe page is served.

```
error_page 404 /404.html;
location = /404.html {
        root /var/www/abc.com/html/;
        internal;
}
```



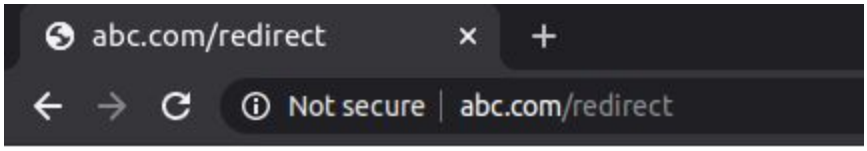
this is the pge of ABC.COM



error page

2. proxy pass to a website xyz.com on a particular URI.
3. redirect to above URI on /redirect/

```
        location /redirect {
                proxy_pass http://127.0.0.1:81/;
        }
```



this is the page of XYZ.COM

4. perform an HTTP to HTTPS redirection including non-www to www redirection.

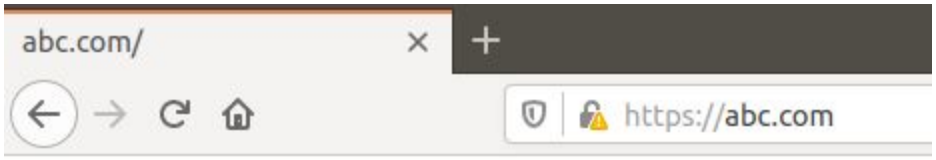
```
server {
    server_name  abc.com;
    return       301 http://www.abc.com$request_uri;
}
server {
        listen 80 default_server;
        listen [::]:80 default_server;
```

```
ayush@ayush:/etc$ curl -vv abc.com
* Rebuilt URL to: abc.com/
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to abc.com (127.0.0.1) port 80 (#0)
> GET / HTTP/1.1
> Host: abc.com
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Server: nginx/1.14.0 (Ubuntu)
< Date: Sun, 16 Feb 2020 10:40:48 GMT
< Content-Type: text/html
< Content-Length: 194
< Connection: keep-alive
< Location: http://www.abc.com/
<
<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.14.0 (Ubuntu)</center>
</body>
</html>
* Connection #0 to host abc.com left intact
```

```
ayush@ayush:/var/log/nginx$ curl -vv abc.com
* Rebuilt URL to: abc.com/
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to abc.com (127.0.0.1) port 80 (#0)
> GET / HTTP/1.1
> Host: abc.com
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
< Server: nginx/1.14.0 (Ubuntu)
< Date: Mon, 17 Feb 2020 04:24:14 GMT
< Content-Type: text/html
< Content-Length: 194
< Connection: keep-alive
< Location: https://www.abc.com/
<
<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.14.0 (Ubuntu)</center>
</body>
</html>
* Connection #0 to host abc.com left intact
```

$sudo openssl req -newkey rsa:2048 -nodes -keyout private.key -x509 -days 365 -out public.pem

```
ayush@ayush:/etc/nginx/sites-available$ cat abc.com
server{
        listen 80;
        server_name abc.com;
        return 302 https://www.abc.com;
        #location = /admin.html {
        #auth_basic "login required";
        #auth_basic_user_file /etc/nginx/.htpasswd;
        #}
        #location / {
          #      try_files $uri $uri/ =404;
        #}

}

server{
        listen 443 ssl;
        server_name www.abc.com;
        root /var/www/abc.com/html;
        index index.html;
        ssl_certificate /etc/nginx/ssl/public.pem;
        ssl_certificate_key /etc/nginx/ssl/private.key;
}
ayush@ayush:/etc/nginx/sites-available$
```

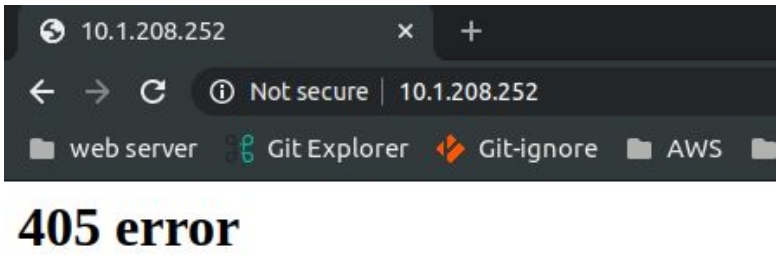


this is the pge of ABC.COM

5. Allow access to a set of particular IPs on a location block and return 405 to other IPs no matter if the page in that location exists.

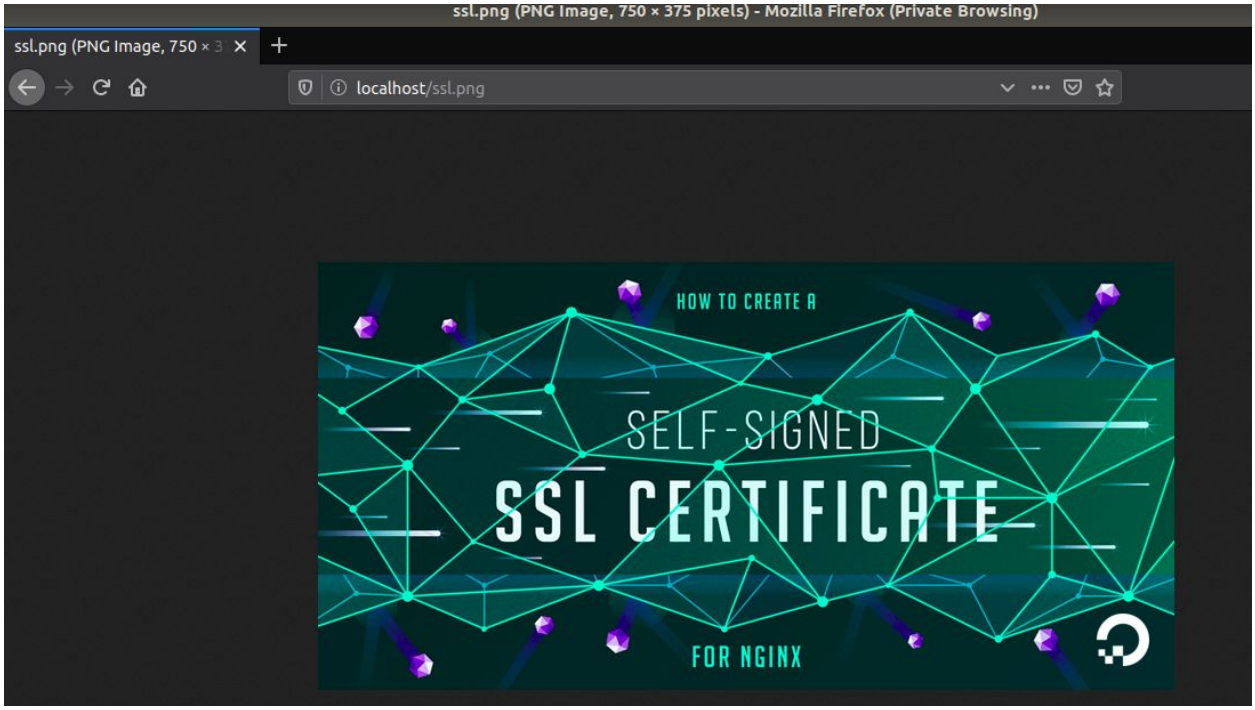
```
server{
        listen 80;
        server_name www.abc.com;
        index index.html;
       location /{
#               proxy_pass http://127.0.0.1:8080/sample/;
                allow 10.1.210.94;
                deny 10.1.208.252;
         }

        error_page 404 403 /404.html;
        location /404.html {
        return 404 "405 error";
        }
#       return 302 https://www.abc.com;
        }
```

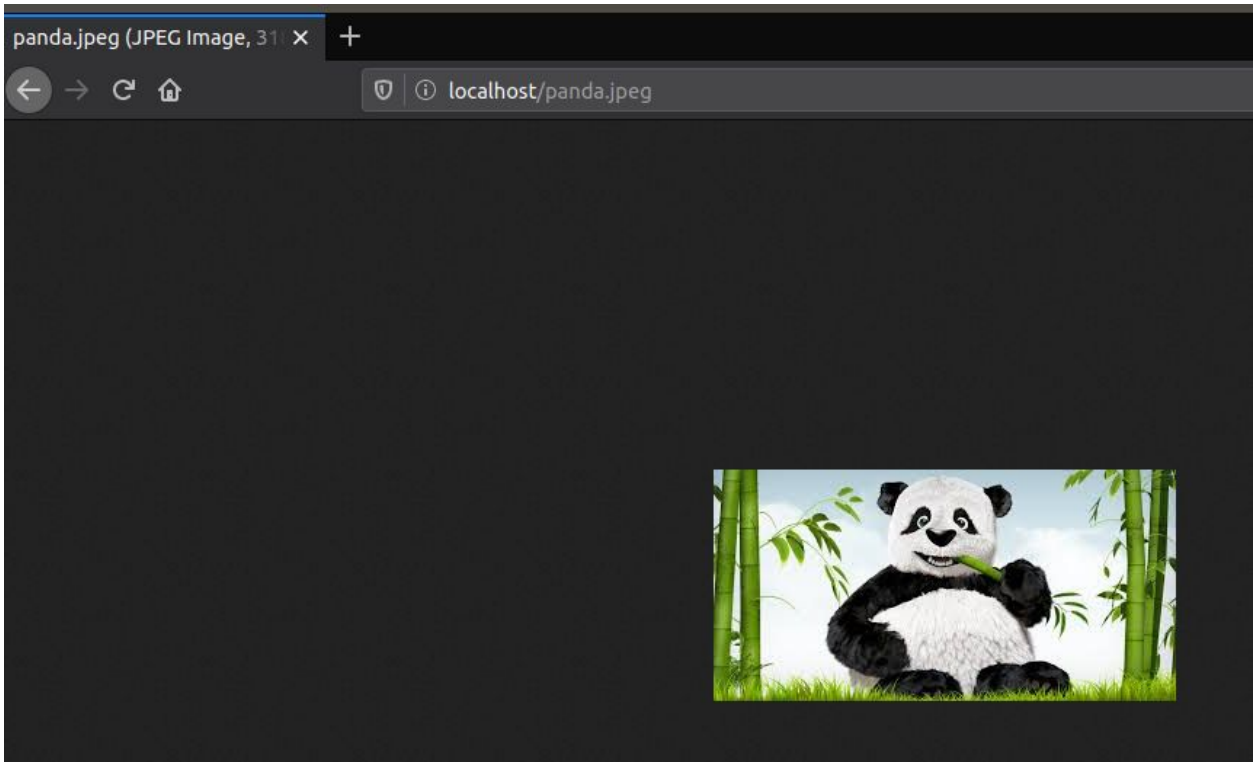
On 10.1.210.94

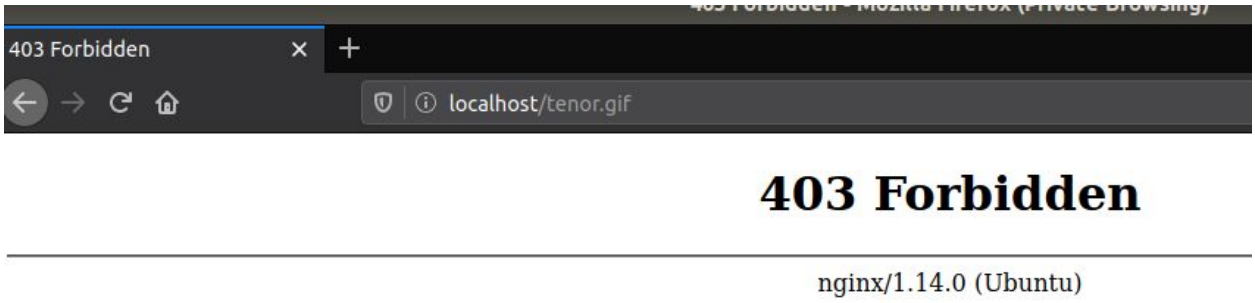


6. Place your images at /var/www/html/images. Only accept jpg/png/jpeg. Discard rest

```
location ~ \.(jpg|jpeg|png) {
 root /var/www/html/images;
  }
 location ~ .* {
 deny all;
 }
```

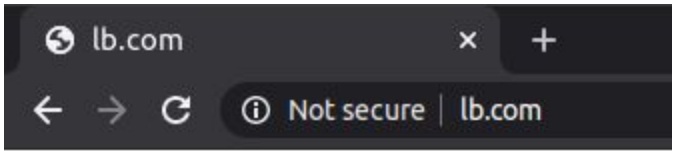




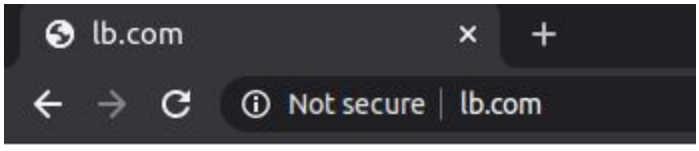



```
ayush@ayush:~$ cd /var/www/html/images/
ayush@ayush:/var/www/html/images$ ls
panda.jpeg  ssl.png  tenor.gif
ayush@ayush:/var/www/html/images$
```

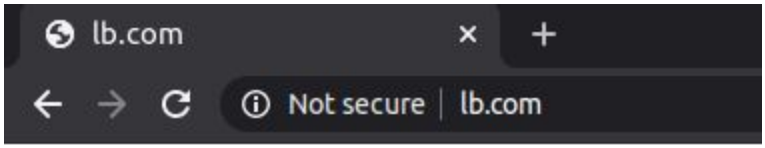
7. Create a load balancer with 5 backends. Explain different types of load balancing methods.



83



85



82

```
ayush@ayush:/etc/nginx/sites-enabled$ cat load
server {
        listen 82;
        server_name localhost;
        index index_82.html;
        root /var/www/html;
}
server {
        listen 83;
        server_name localhost;
        index index_83.html;
        root /var/www/html;
}
server {
        listen 84;
        server_name localhost;
        index index_84.html;
        root /var/www/html;
}
server {
        listen 85;
        server_name localhost;
        index index_85.html;
        root /var/www/html;
}
```

```
ayush@ayush:/etc/nginx/sites-enabled$ cat loadbalancer
upstream backend{
        server 127.0.0.1:82;
        server 127.0.0.1:83;
        server 127.0.0.1:84;
        server 127.0.0.1:85;
}

server {
        listen 80;
        server_name lb.com;
        location /{
                proxy_pass http://backend;
        }
}
ayush@ayush:/etc/nginx/sites-enabled$
```

8. Setup Basic Auth (Popup asking for username and password) in a particular location block. (The Basic Auth should not be asked for TTN IP)

```
$sudo htpasswd /etc/nginx/.htpasswd nginx
```