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

   Ans) A reverse proxy server is a server in which server he is connecting to. It gives the following benefits.

   - **Load Balancing:** A reverse proxy can provide a load balancing solution which will distribute the incoming traffic evenly among the different servers to prevent any single server from becoming overloaded. In the event that a server fails completely, other servers can step up to handle the traffic.

   - **Caching:** A reverse proxy can also cache content, resulting in faster performance.

   - **SSL Encryption:** Encrypting and decrypting SSL (or TLS) communications for each client can be computationally expensive for an origin server. A reverse proxy can be configured to decrypt all incoming requests and encrypt all outgoing responses, freeing up valuable resources on the origin server.

   - **Logging:** Health of all the servers can be checked with the reverse proxy server.

   - **Canary Deployment:** If we want to display different outputs for different requests without enabling the client to know which server is serving the request.

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

   1) Fast Static Content Processing: Nginx can perform a much better job at handling the static files from a specific directory.Also, the upstream server processes don’t get blocked because of the heavy, multiple static content requests as Nginx can process them concurrently. This significantly improves the overall performance of backend servers.

   2) Great for High Traffic Websites: If we talk about the speed and how many clients can be served on a high load, Nginx will always shine as a winner over Apache. This makes Nginx significantly lightweight and great for server resources. This is why most of the web developers prefer Nginx over Apache.

   3) Backend: If your website is PHP dependent, Nginx is the far best way to host application.

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

   Ans) 1. **Worker Node/Server Node**: A server node is a virtual node which is created by nginx to serve user requests. The details of each worker node/server node is mentioned in the server context of the nginx.conf file.

2. Worker Connections : NGINX can run multiple worker processes, each capable of processing a large number of simultaneous 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.

Max capacity (no of clients) = Product of total number of worker processes and number of worker connections in each process.

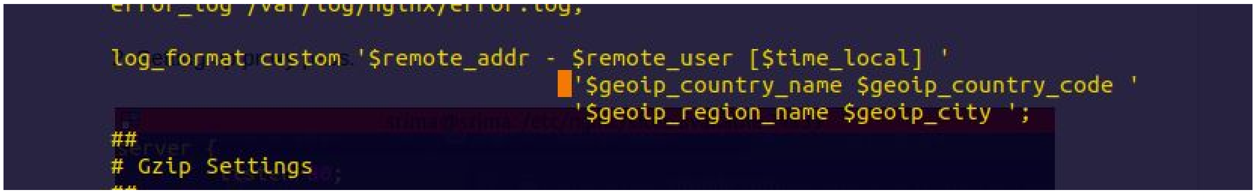
4.From what directory will NGINX automatically load server (virtual host) configurations when using the default /etc/nginx/nginx.conf configuration?

Ans) /etc/nginx/conf.d directory

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

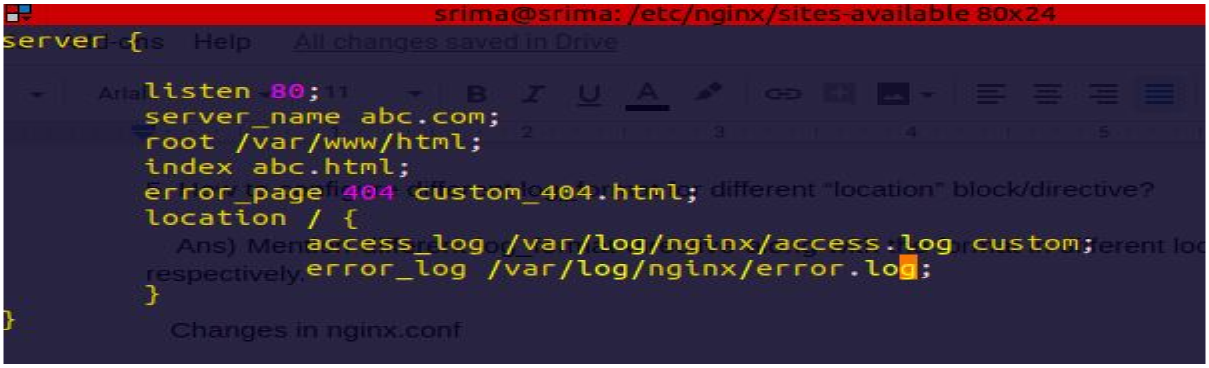
Ans) Mention different log_format directive along with the format in different location contexts respectively.

Changes in nginx.conf



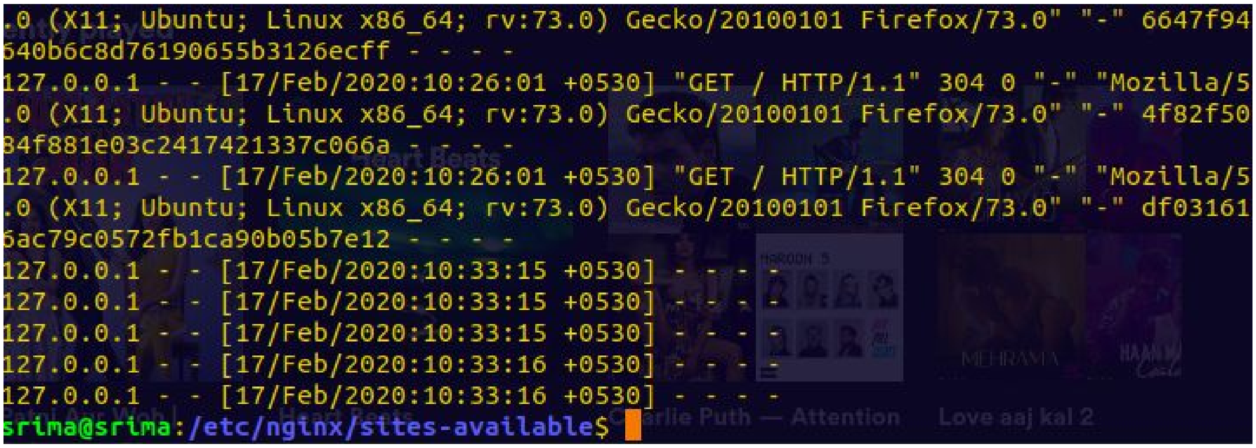
```
        error_log /var/log/nginx/error.log;

        log_format custom '$remote_addr - $remote_user [$time_local] '
                          '$geoip_country_name $geoip_country_code '
                          '$geoip_region_name $geoip_city ';
        ##
        # Gzip Settings
        ##
```

Changes in abc.com



```
server {
        listen 80;
        server_name abc.com;
        root /var/www/html;
        index abc.html;
        error_page 404 custom_404.html;
        location / {
            access_log /var/log/nginx/access.log custom;
            error_log /var/log/nginx/error.log;
        }
}
```

OUTPUT



```
.0 (X11; Ubuntu; Linux x86_64; rv:73.0) Gecko/20100101 Firefox/73.0" "-" 6647f94
640b6c8d76190655b3126ecff - - - -
127.0.0.1 - - [17/Feb/2020:10:26:01 +0530] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5
.0 (X11; Ubuntu; Linux x86_64; rv:73.0) Gecko/20100101 Firefox/73.0" "-" 4f82f50
84f881e03c2417421337c066a - - - -
127.0.0.1 - - [17/Feb/2020:10:26:01 +0530] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5
.0 (X11; Ubuntu; Linux x86_64; rv:73.0) Gecko/20100101 Firefox/73.0" "-" df03161
6ac79c0572fb1ca90b05b7e12 - - - -
127.0.0.1 - - [17/Feb/2020:10:33:15 +0530] - - - -
127.0.0.1 - - [17/Feb/2020:10:33:15 +0530] - - - -
127.0.0.1 - - [17/Feb/2020:10:33:15 +0530] - - - -
127.0.0.1 - - [17/Feb/2020:10:33:16 +0530] - - - -
127.0.0.1 - - [17/Feb/2020:10:33:16 +0530] - - - -
srima@srima:/etc/nginx/sites-available$
```

Here we can see that the log format has changed .

6. Host a site ABC.COM

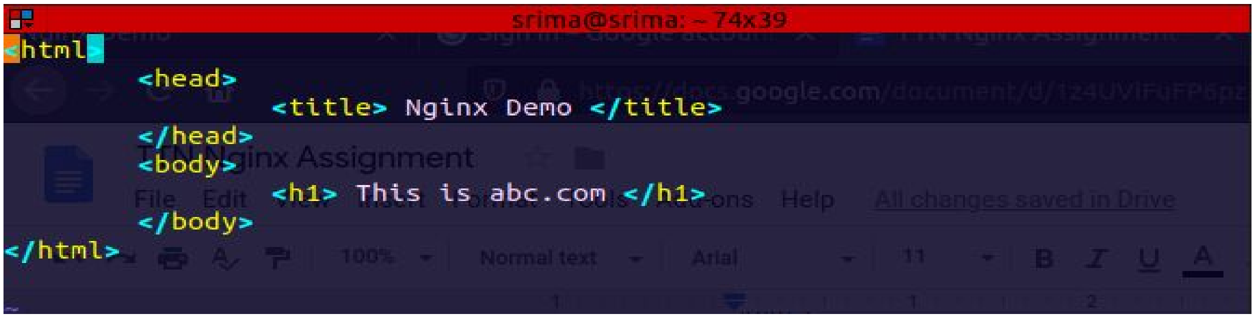
1. **Create an index page and a fail-safe page. If a page for URI is not available, the fail-safe page is served.**
2. **proxy pass to a website xyz.com on a particular URI.**
3. **redirect to above URI on /redirect/**
4. **perform an HTTP to HTTPS redirection including non-www to www redirection.**
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.**
6. **Place your images at /var/www/html/images. Only accept jpg/png/jpeg. Discard rest.**

Ans 6) 1.1 Make abc.html, xyz.html, custom_error.html in /etc/nginx/sites-available folder .

1.2 make a symlink in sites-enabled folder, everytime you make a change in any nginx configuration

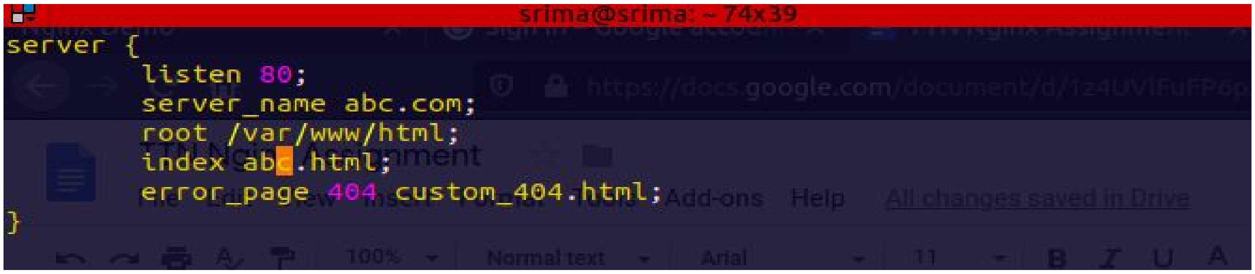
1.3 check configuration syntax using sudo nginx -t command and then reload nginx using sudo systemctl reload nginx.

abc.html in /var/www/html directory



```
<html>
        <head>
                <title> Nginx Demo </title>
        </head>
        <body>
                <h1> This is abc.com </h1>
        </body>
</html>
```

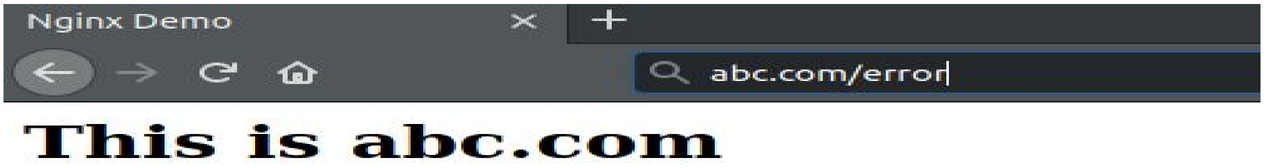
/etc/nginx/sites-available



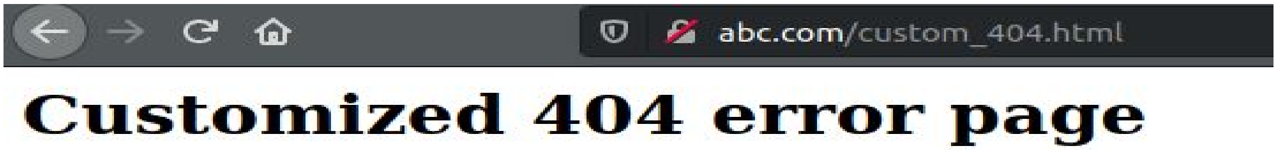
```
server {
        listen 80;
        server_name abc.com;
        root /var/www/html;
        index abc.html;
        error_page 404 custom_404.html;
}
```

OUTPUT

Hit on http://abc.com/error

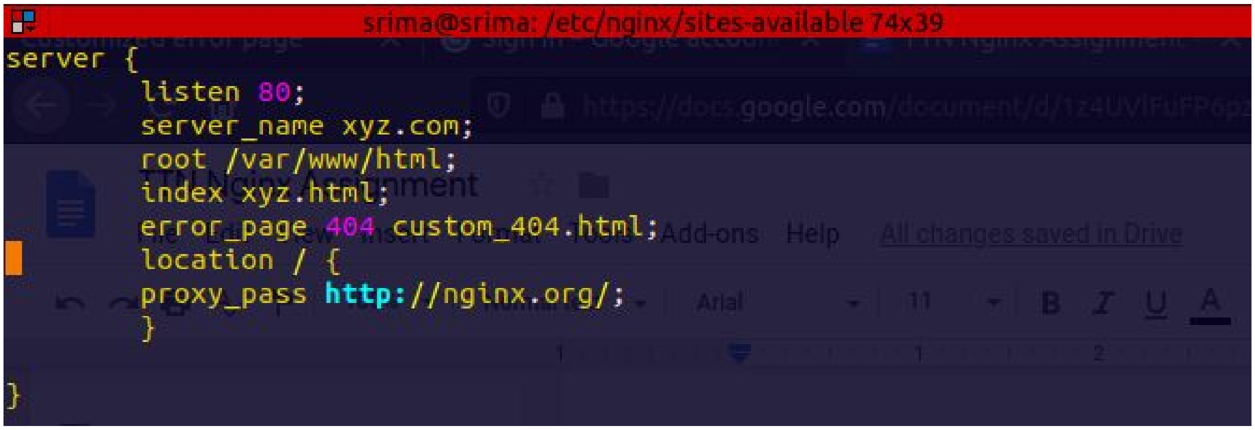


This is abc.com



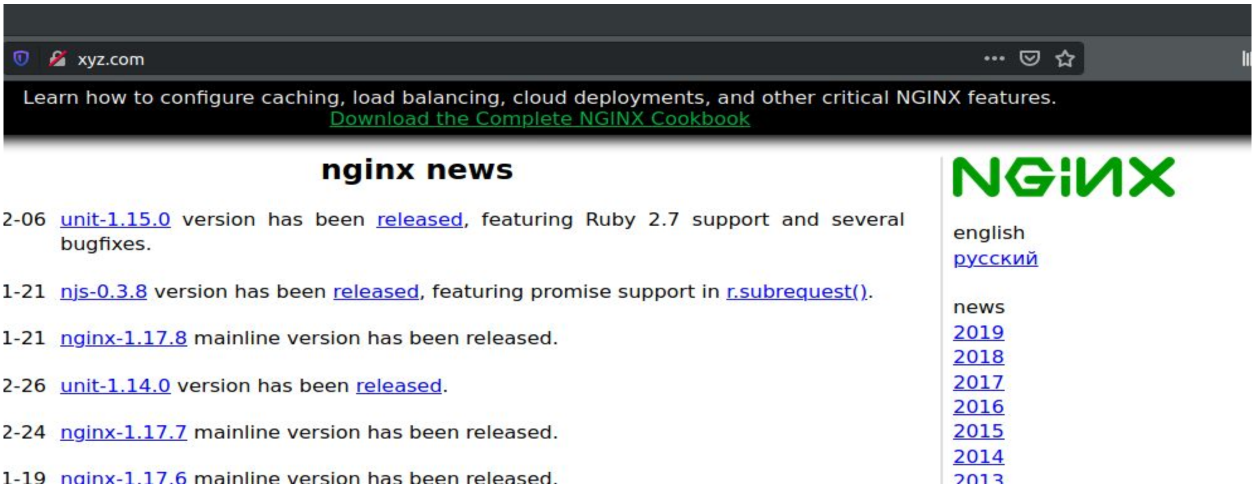
Customized 404 error page

6.2. Setting up proxy pass.



```
server {
        listen 80;
        server_name xyz.com;
        root /var/www/html;
        index xyz.html;
        error_page 404 custom_404.html;
        location / {
        proxy_pass http://nginx.org/;
        }

}
```

OUTPUT



Learn how to configure caching, load balancing, cloud deployments, and other critical NGINX features.
Download the Complete NGINX Cookbook

nginx news

2-06 unit-1.15.0 version has been released, featuring Ruby 2.7 support and several bugfixes.

1-21 njs-0.3.8 version has been released, featuring promise support in r.subrequest().

1-21 nginx-1.17.8 mainline version has been released.

2-26 unit-1.14.0 version has been released.

2-24 nginx-1.17.7 mainline version has been released.

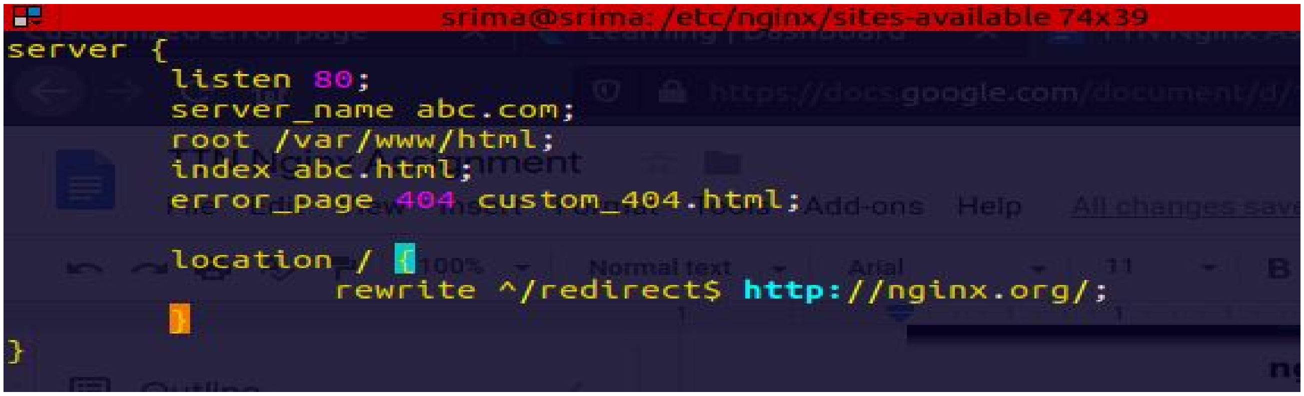
1-19 nginx-1.17.6 mainline version has been released.

NGINX

english
русский

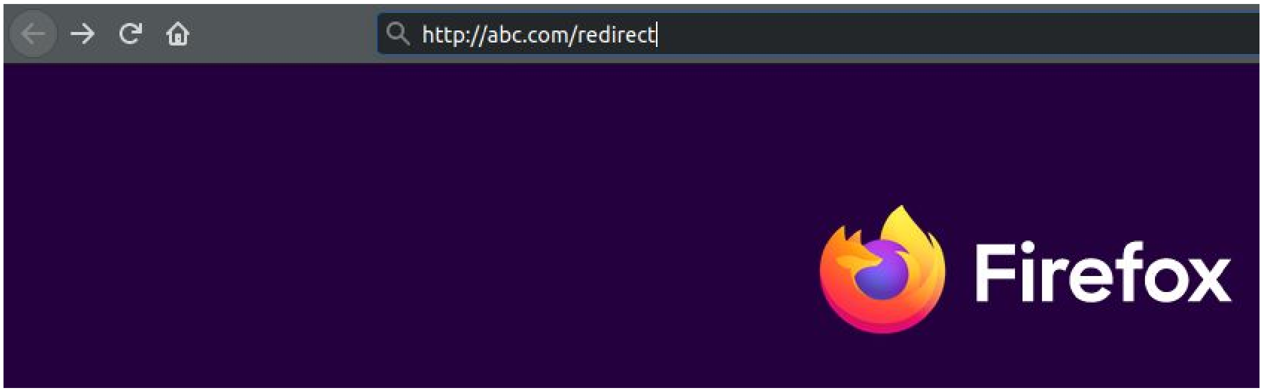
news
2019
2018
2017
2016
2015
2014
2013

6.3. redirect to above URI on /redirect/



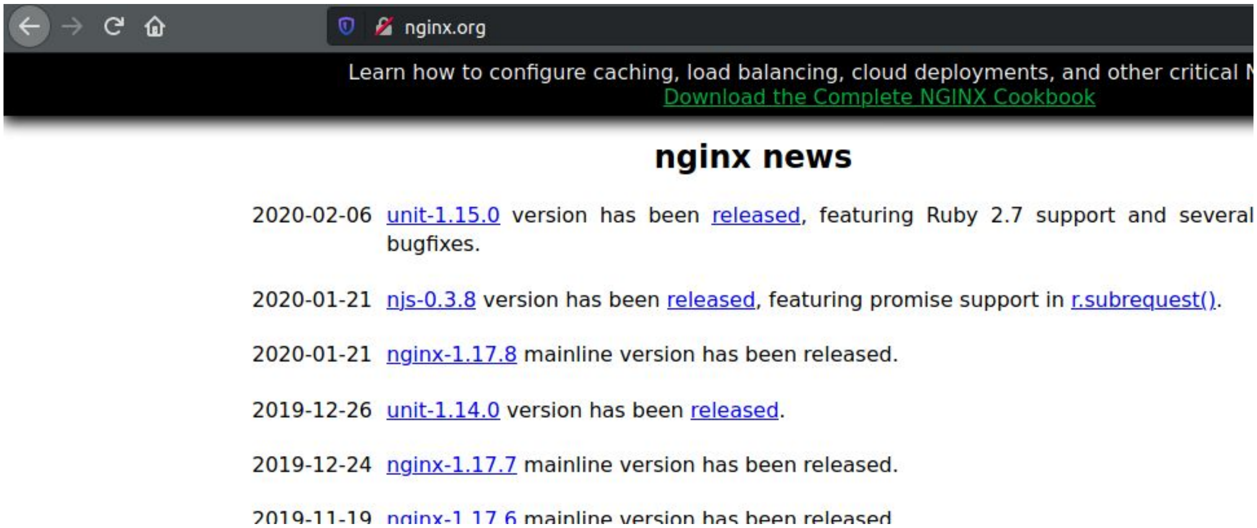
```
server {
        listen 80;
        server_name abc.com;
        root /var/www/html;
        index abc.html;
        error_page 404 custom_404.html;

        location / {
                rewrite ^/redirect$ http://nginx.org/;
        }
}
```

OUTPUT



http://abc.com/redirect

On pressing enter



nginx.org

Learn how to configure caching, load balancing, cloud deployments, and other critical N
Download the Complete NGINX Cookbook

nginx news

2020-02-06 unit-1.15.0 version has been released, featuring Ruby 2.7 support and several bugfixes.

2020-01-21 njs-0.3.8 version has been released, featuring promise support in r.subrequest().

2020-01-21 nginx-1.17.8 mainline version has been released.

2019-12-26 unit-1.14.0 version has been released.

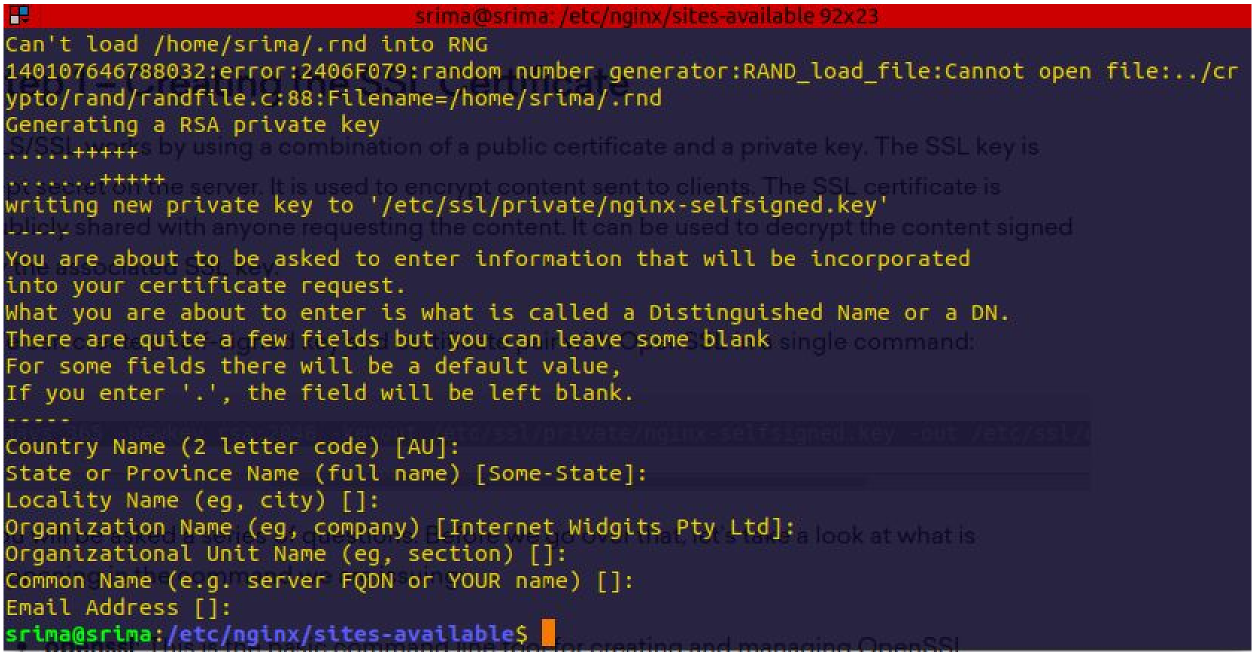
2019-12-24 nginx-1.17.7 mainline version has been released.

2019-11-19 nginx-1.17.6 mainline version has been released

6.4. Perform an HTTP to HTTPS redirection including non-www to www redirection.

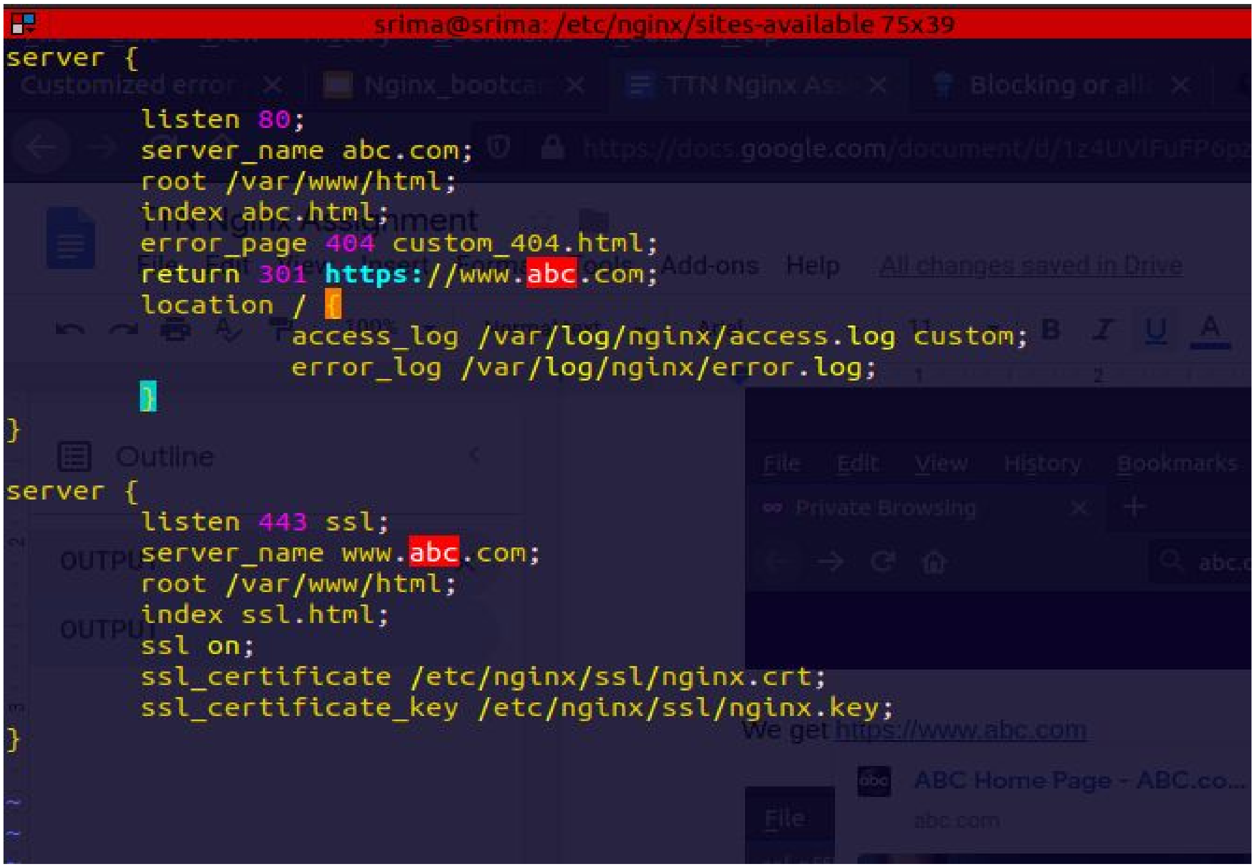
Generating Openssl certificate

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
```



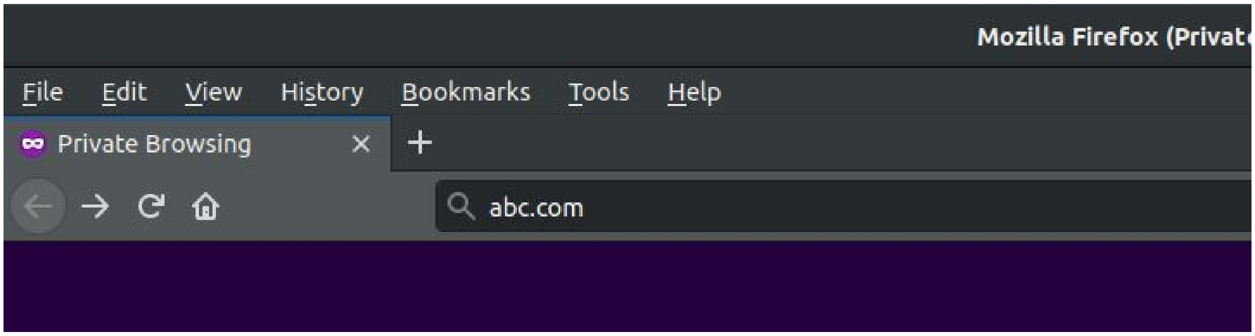
```
Can't load /home/srima/.rnd into RNG
140107646788032:error:2406F079:random number generator:RAND_load_file:Cannot open file:../cr
ypto/rand/randfile.c:88:Filename=/home/srima/.rnd
Generating a RSA private key
.....+++++
........+++++
writing new private key to '/etc/ssl/private/nginx-selfsigned.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
srima@srima:/etc/nginx/sites-available$ 
```

Configure abc.com server block , provide ssl certificate and key.



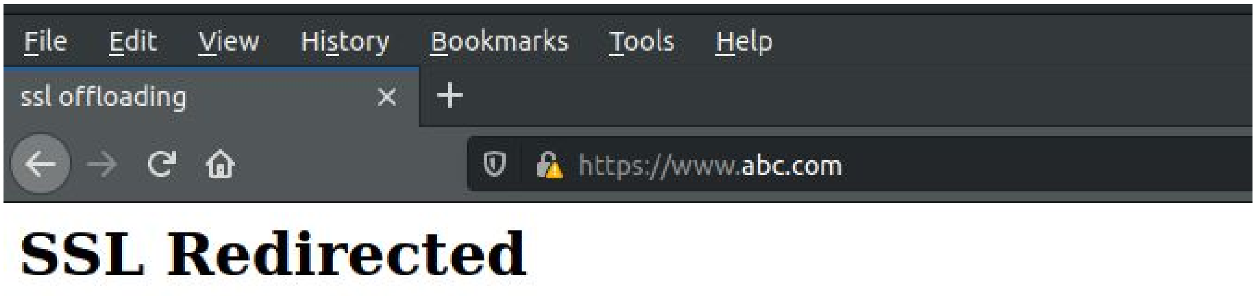
```
server {

        listen 80;
        server_name abc.com;
        root /var/www/html;
        index abc.html;
        error_page 404 custom_404.html;
        return 301 https://www.abc.com;
        location / {
                 access_log /var/log/nginx/access.log custom;
                error_log /var/log/nginx/error.log;

        }
}

server {
        listen 443 ssl;
        server_name www.abc.com;
        root /var/www/html;
        index ssl.html;
        ssl on;
        ssl_certificate /etc/nginx/ssl/nginx.crt;
        ssl_certificate_key /etc/nginx/ssl/nginx.key;
}

~
~
```

OUTPUT
On typing abc.com



abc.com

We get https://www.abc.com



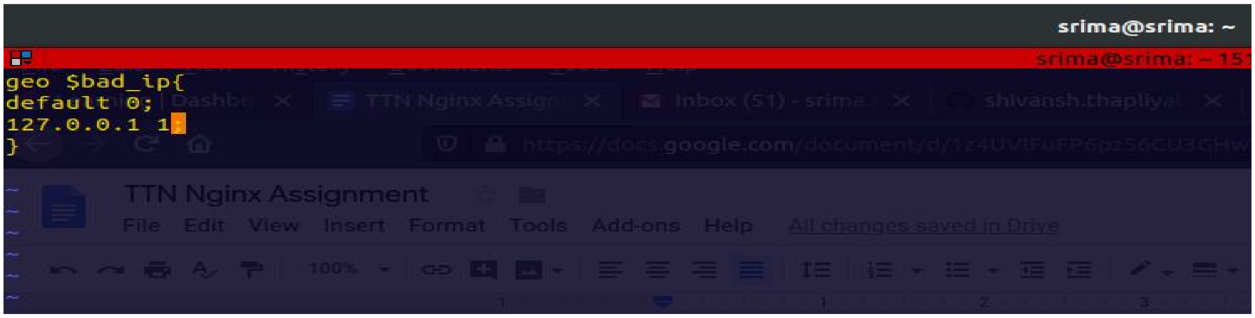
https://www.abc.com

SSL Redirected

6.5 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.

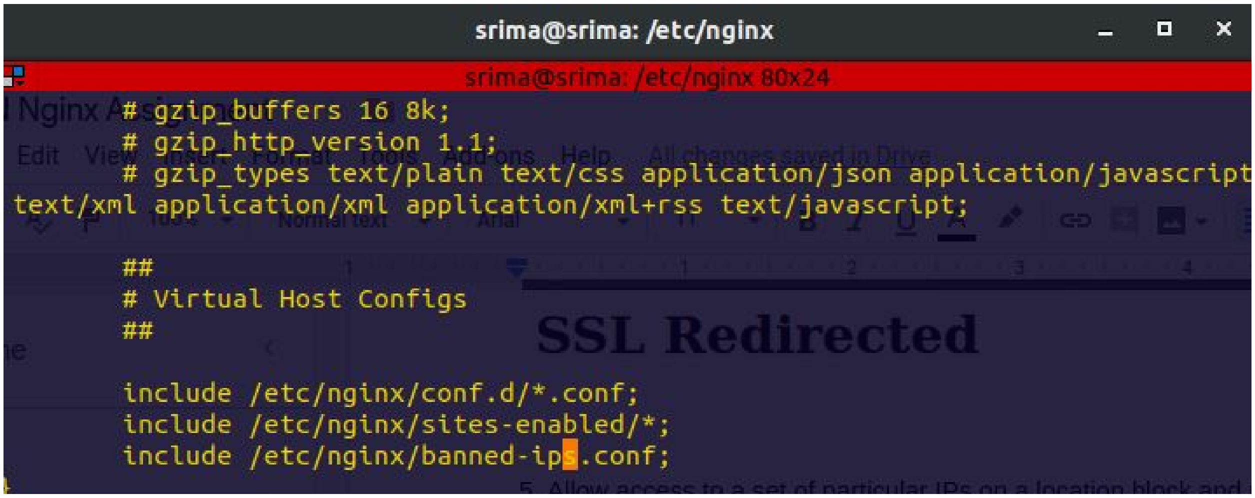
Sudo vim abc.com

banned-ips.conf



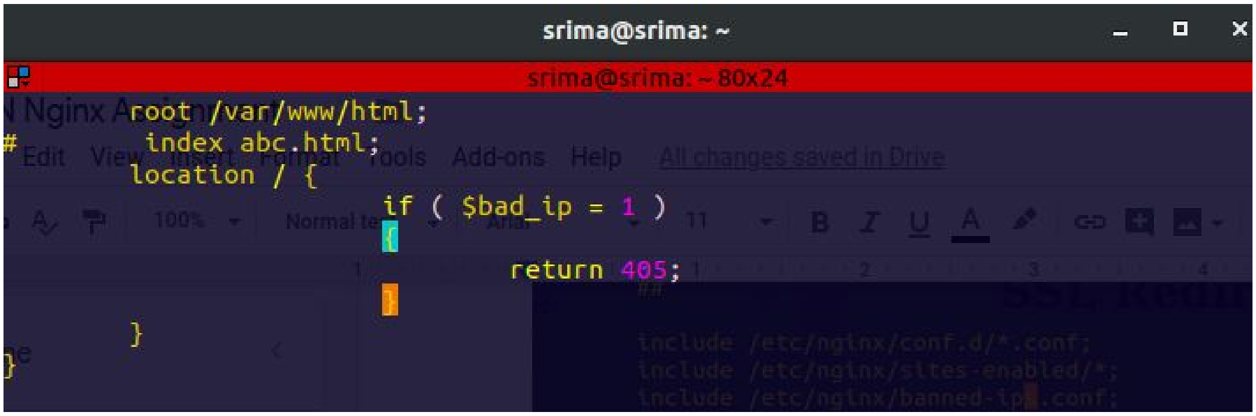
```
geo $bad_ip{
default 0;
127.0.0.1 1;
}
```

/etc/nginx.conf



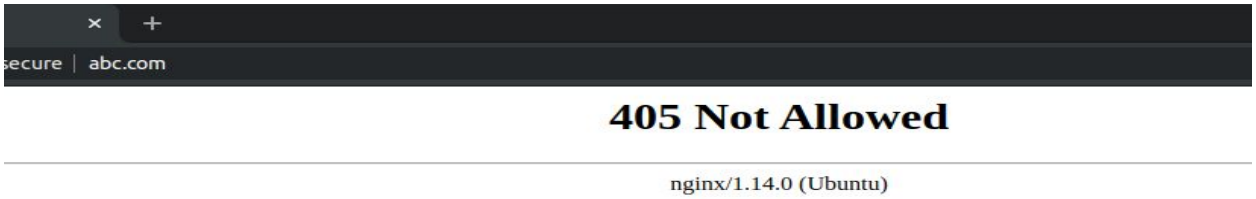
```
        # gzip_buffers 16 8k;
        # gzip_http_version 1.1;
        # gzip_types text/plain text/css application/json application/javascript
 text/xml application/xml application/xml+rss text/javascript;

        ##
        # Virtual Host Configs
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
        include /etc/nginx/banned-ips.conf;
```

abc.com



```
        root /var/www/html;
#        index abc.html;
        location / {
                        if ( $bad_ip = 1 )
                        {
                                return 405;
                        }

        }
}
```

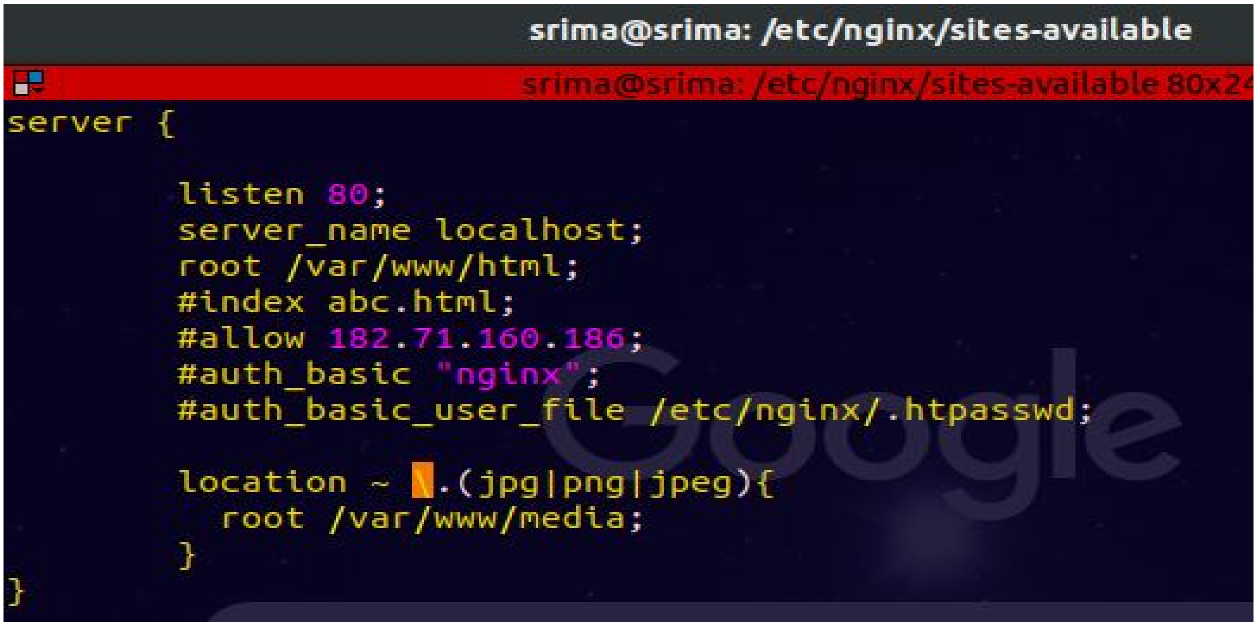
OUTPUT



405 Not Allowed

nginx/1.14.0 (Ubuntu)

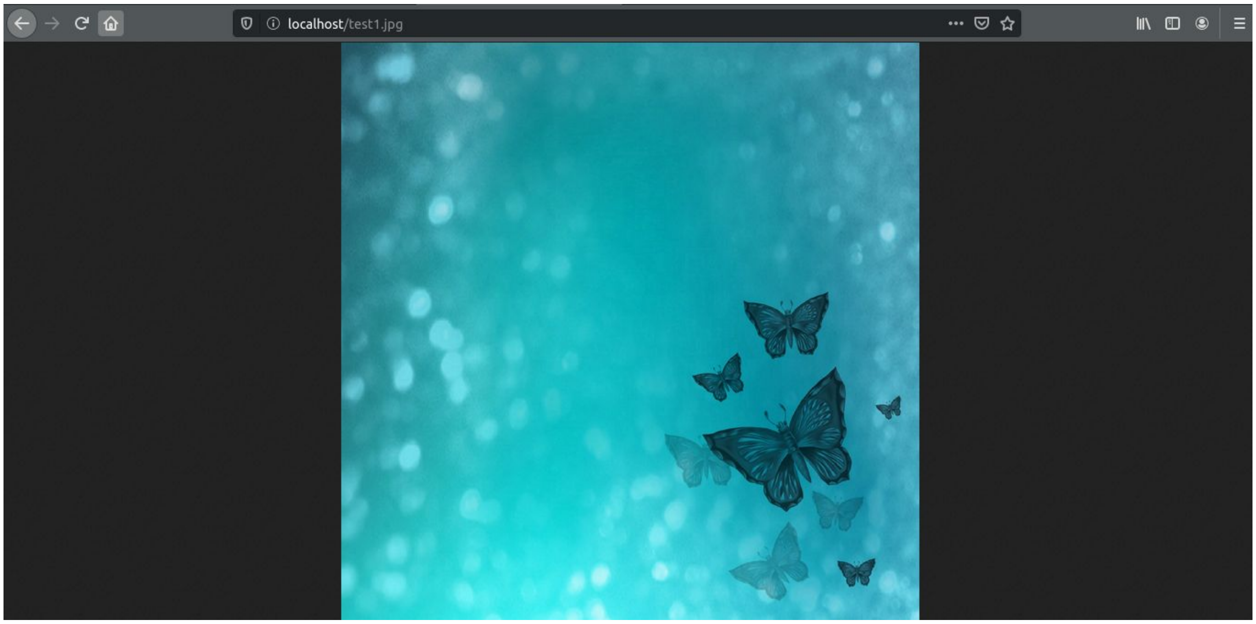
6.6 Place your images at /var/www/html/images. Only accept jpg/png/jpeg. Discard rest.



srima@srima: /etc/nginx/sites-available

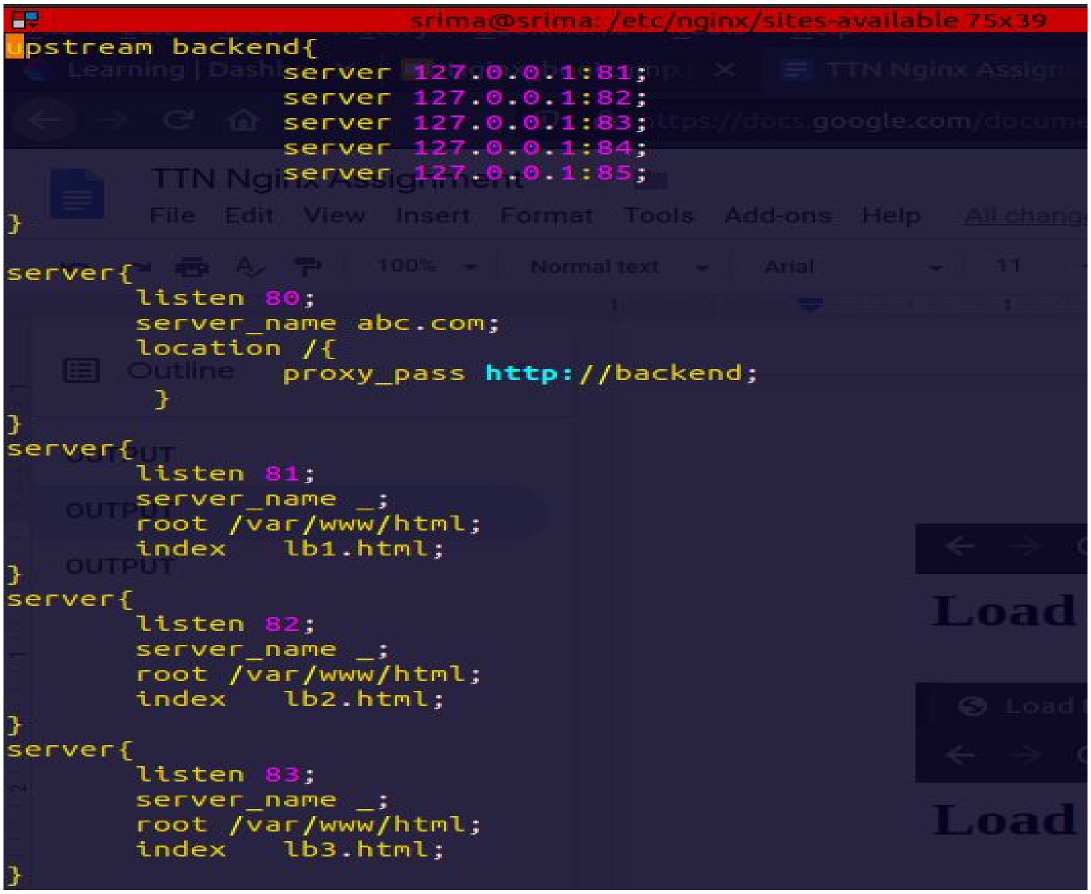
srima@srima: /etc/nginx/sites-available 80x2

```
server {

        listen 80;
        server_name localhost;
        root /var/www/html;
        #index abc.html;
        #allow 182.71.160.186;
        #auth_basic "nginx";
        #auth_basic_user_file /etc/nginx/.htpasswd;

        location ~ \.(jpg|png|jpeg){
          root /var/www/media;
        }
}
```

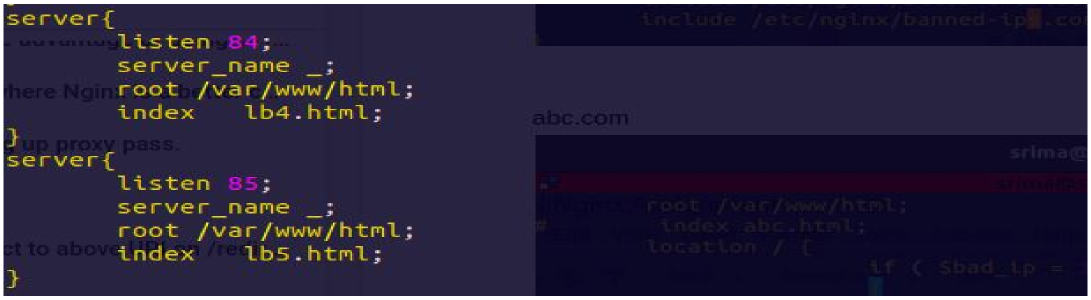


7.Create a load balancer with 5 backends. Explain different types of load balancing methods.

/etc/nginx.sites-available/lb.com

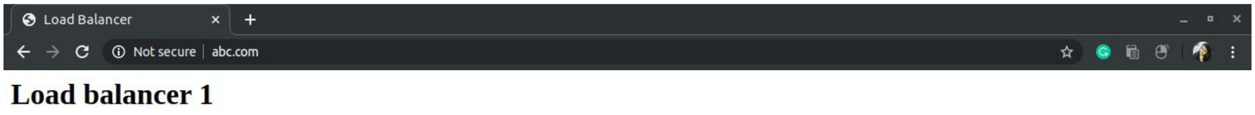


```
upstream backend{
                server 127.0.0.1:81;
                server 127.0.0.1:82;
                server 127.0.0.1:83;
                server 127.0.0.1:84;
                server 127.0.0.1:85;

}

server{
        listen 80;
        server_name abc.com;
        location /{
                proxy_pass http://backend;
          }
}
server{
        listen 81;
        server_name _;
        root /var/www/html;
        index   lb1.html;
}
server{
        listen 82;
        server_name _;
        root /var/www/html;
        index   lb2.html;
}
server{
        listen 83;
        server_name _;
        root /var/www/html;
        index   lb3.html;
}
```

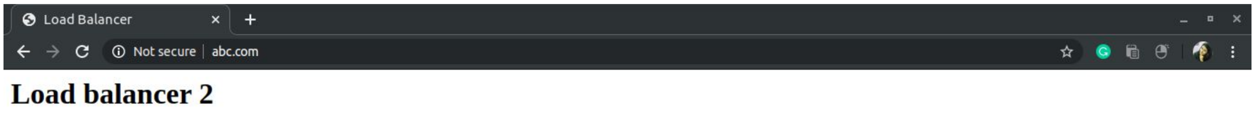


```
server{
        listen 84;
        server_name _;
        root /var/www/html;
        index   lb4.html;
}
server{
        listen 85;
        server_name _;
        root /var/www/html;
        index   lb5.html;
}
```

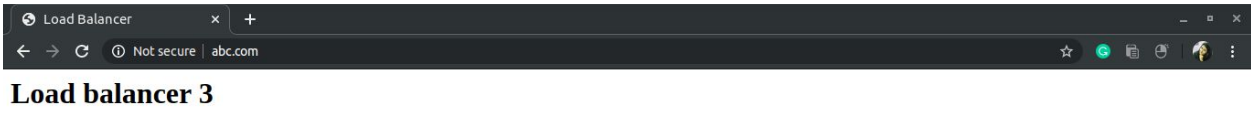
OUTPUT



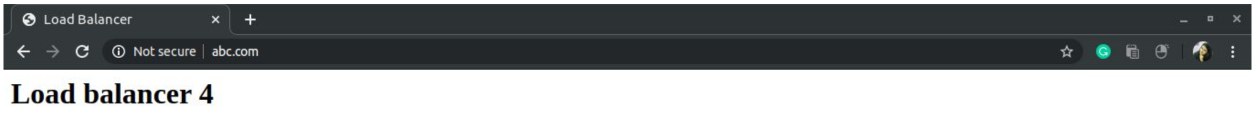
Load balancer 1



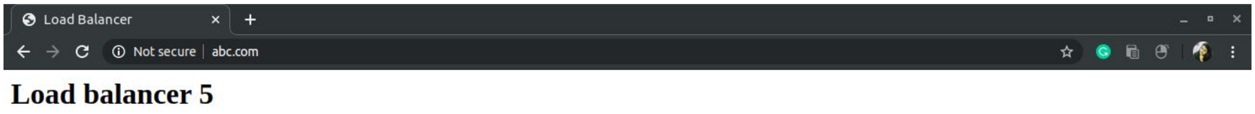
Load balancer 2



Load balancer 3



Load balancer 4

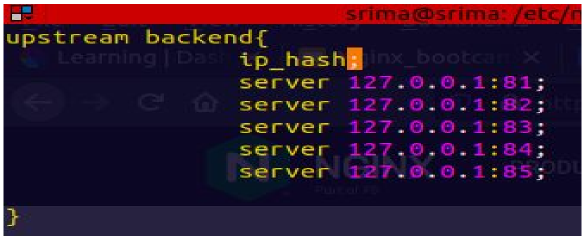


Load balancer 5

a. Round Robin (By default)

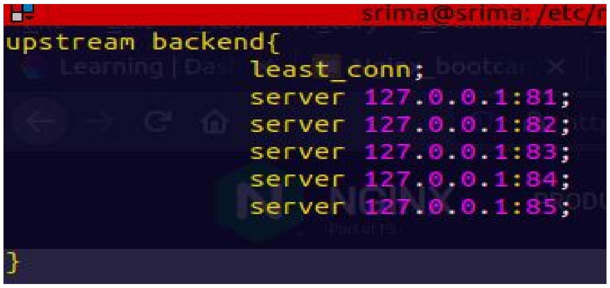
This load balancing algorithm follows normal round robin algorithm, redirects the requests one by one to all the servers. The above method is by default round robin method.

b. Ip Hash Method: The server to which a request is sent is determined from the client IP address. In this case, either the first three octets of the IPv4 address or the whole IPv6 address are used to calculate the hash value. The method guarantees that requests from the same address get to the same server unless it is not available.



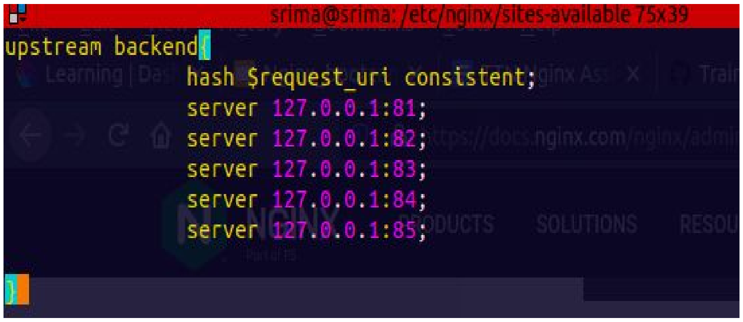
```
upstream backend{
               ip_hash;
               server 127.0.0.1:81;
               server 127.0.0.1:82;
               server 127.0.0.1:83;
               server 127.0.0.1:84;
               server 127.0.0.1:85;
}
```

c. Least Connection: A request is sent to the server with the least number of active connections, again with server weights taken into consideration:



```
upstream backend{
            least_conn;
            server 127.0.0.1:81;
            server 127.0.0.1:82;
            server 127.0.0.1:83;
            server 127.0.0.1:84;
            server 127.0.0.1:85;

}
```

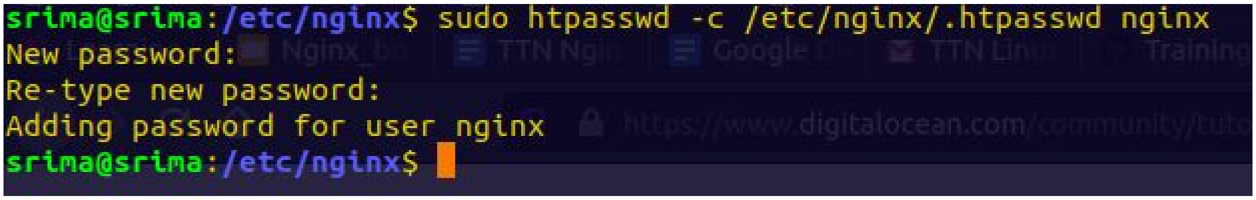
d. **Hash method:** The server to which a request is sent is determined from a user-defined key which can be a text string, variable, or a combination.



```
upstream backend{
            hash $request_uri consistent;
            server 127.0.0.1:81;
            server 127.0.0.1:82;
            server 127.0.0.1:83;
            server 127.0.0.1:84;
            server 127.0.0.1:85;

}
```

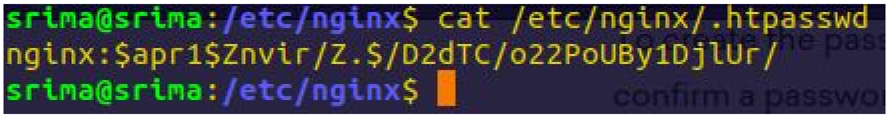
8. Setup Basic Auth (Popup asking for username and password) in a particular location block. (The Basic Auth should not be asked for TTN IP)

Set basic auth password using htpasswd.



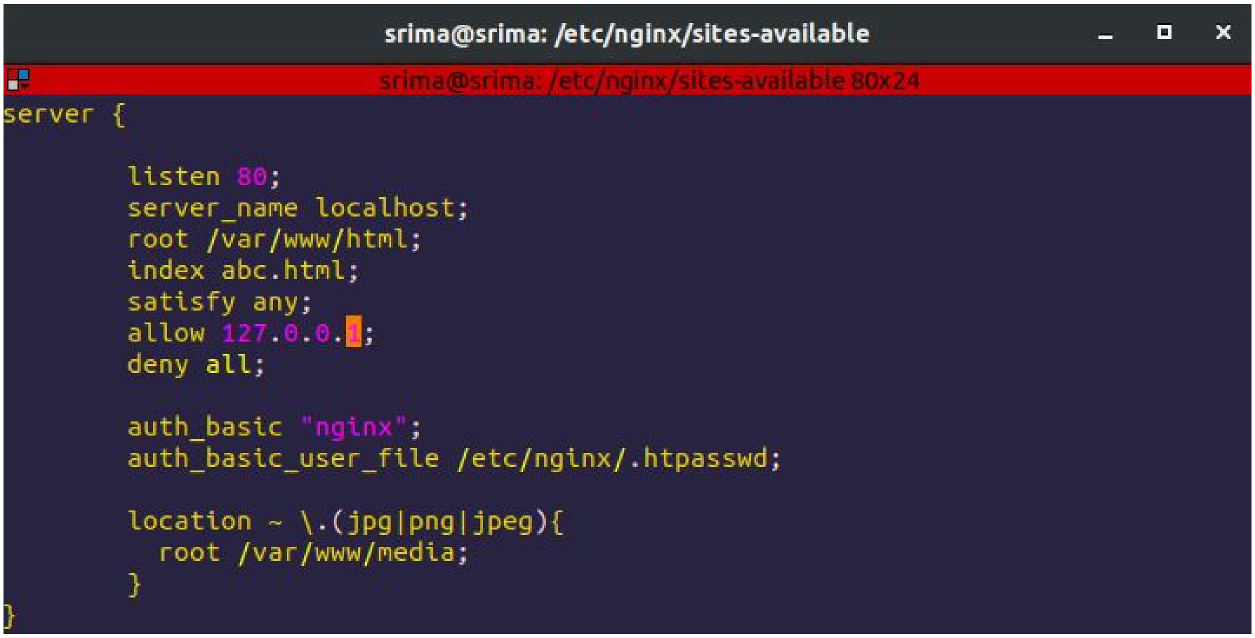
```
srima@srima:/etc/nginx$ sudo htpasswd -c /etc/nginx/.htpasswd nginx
New password:
Re-type new password:
Adding password for user nginx
srima@srima:/etc/nginx$
```

Username and hashed password



```
srima@srima:/etc/nginx$ cat /etc/nginx/.htpasswd
nginx:$apr1$Znvir/Z.$/D2dTC/o22PoUBy1DjlUr/
srima@srima:/etc/nginx$
```

abc.com

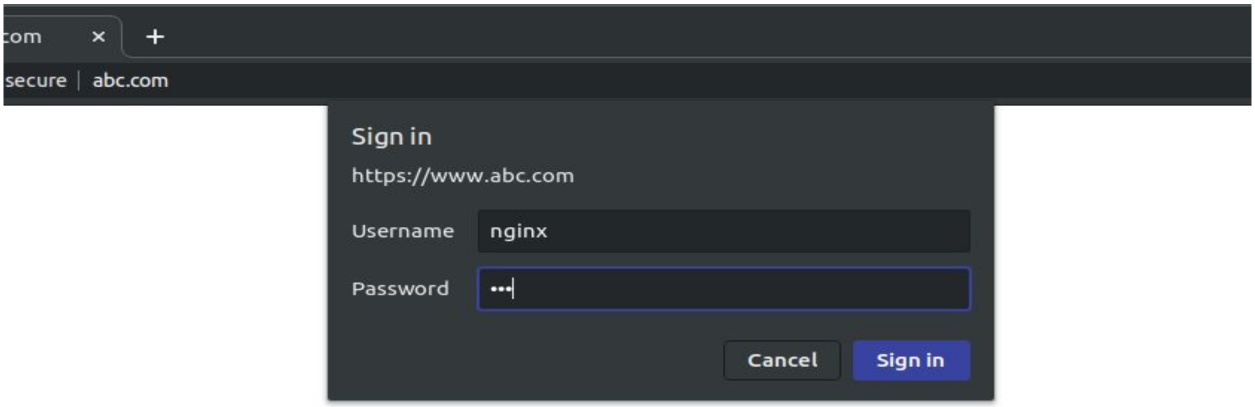

srima@srima: /etc/nginx/sites-available
srima@srima: /etc/nginx/sites-available 80x24

```
server {

        listen 80;
        server_name localhost;
        root /var/www/html;
        index abc.html;
        satisfy any;
        allow 127.0.0.1;
        deny all;

        auth_basic "nginx";
        auth_basic_user_file /etc/nginx/.htpasswd;

        location ~ \.(jpg|png|jpeg){
          root /var/www/media;
        }
}
```

srima@srima: /etc/nginx/sites-available
srima@srima: /etc/nginx/sites-available 80x24

```
server {

        listen 80;
        server_name localhost;
        root /var/www/html;
        index abc.html;
        satisfy any;
        allow 127.0.0.1;
        deny all;

        auth_basic "nginx";
        auth_basic_user_file /etc/nginx/.htpasswd;

        location ~ \.(jpg|png|jpeg){
          root /var/www/media;
        }
}
```

OUTPUT



Sign in

https://www.abc.com

Username nginx

Password •••