

# Assessment 3– Web Server

**Trainee Name : Gargi Sharma**

**Mentor Name : Mr. Ravi Kumar**

**College Name : UPES**

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

Ans) A reverse proxy server is a server in which the server he is connecting to, 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.

## **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.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.

3. **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

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?**

**In nginx.conf :**

log\_format combined

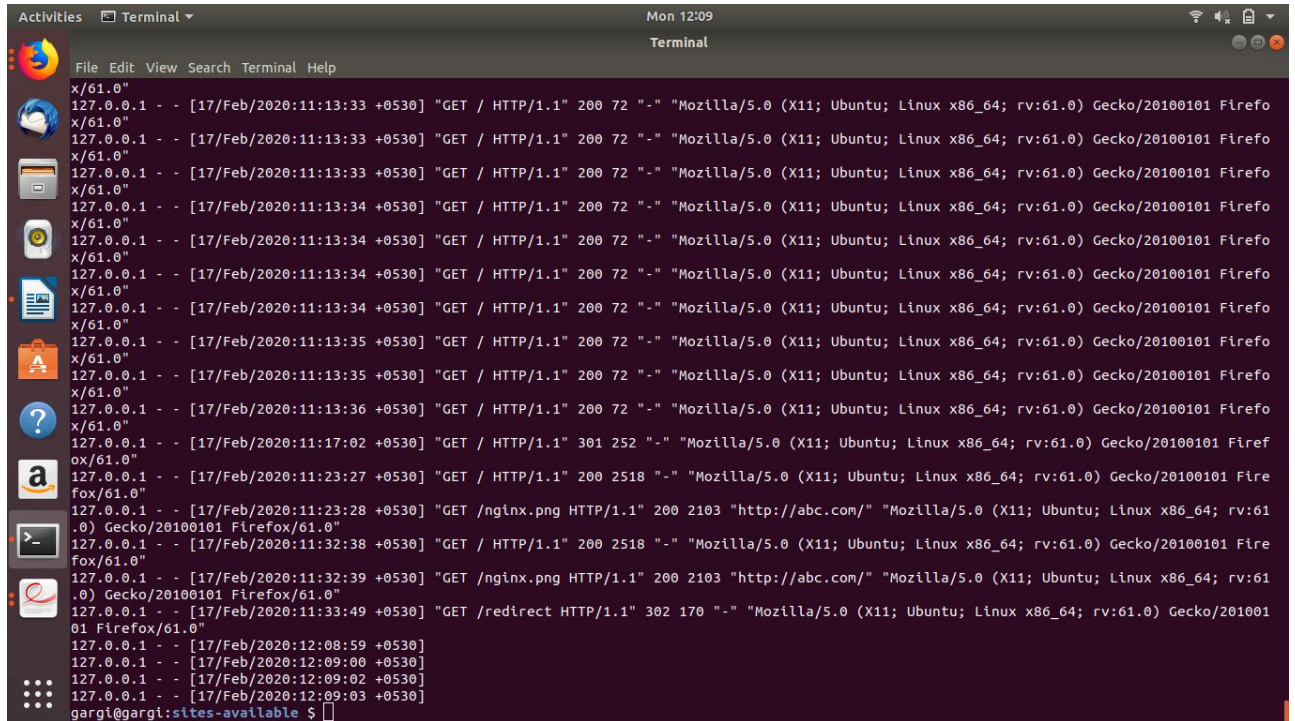
'\$remote\_addr - \$remote\_user [\$time\_local]

"\$request" \$status \$body\_bytes\_sent '

"\$http\_referer" "\$http\_user\_agent";

**In abc.com :**

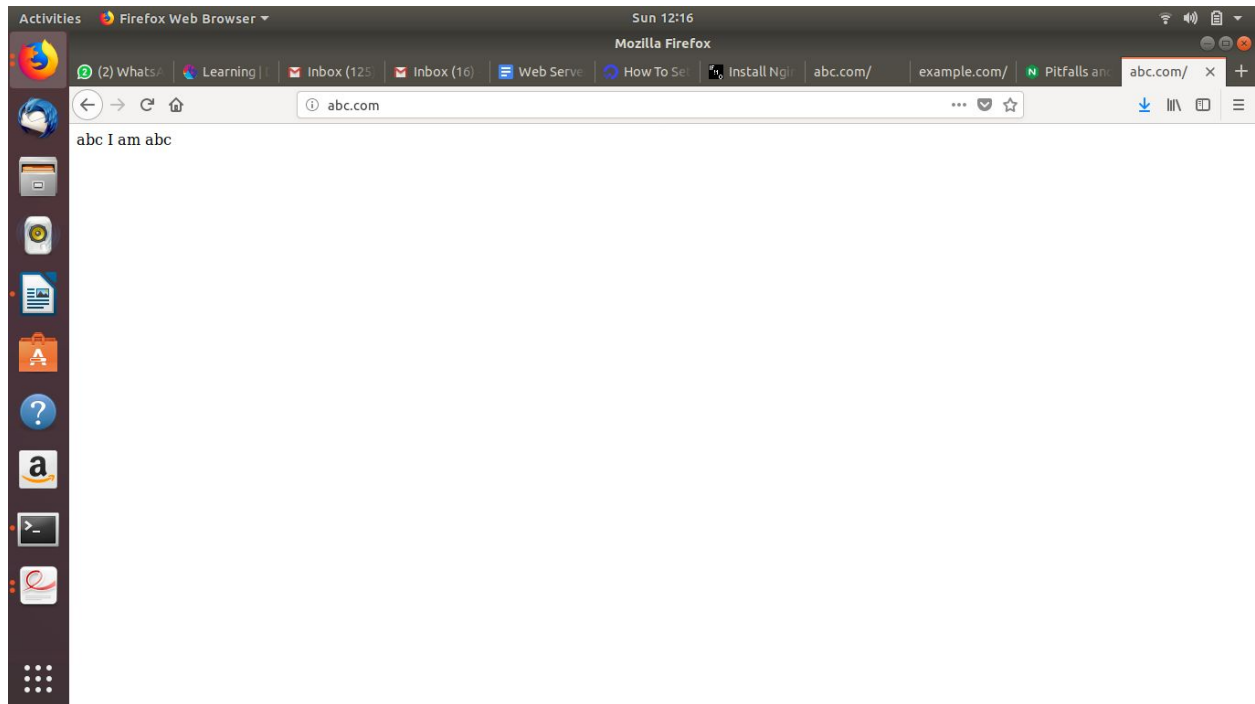
Access\_log /var/log/nginx/access.log combined;



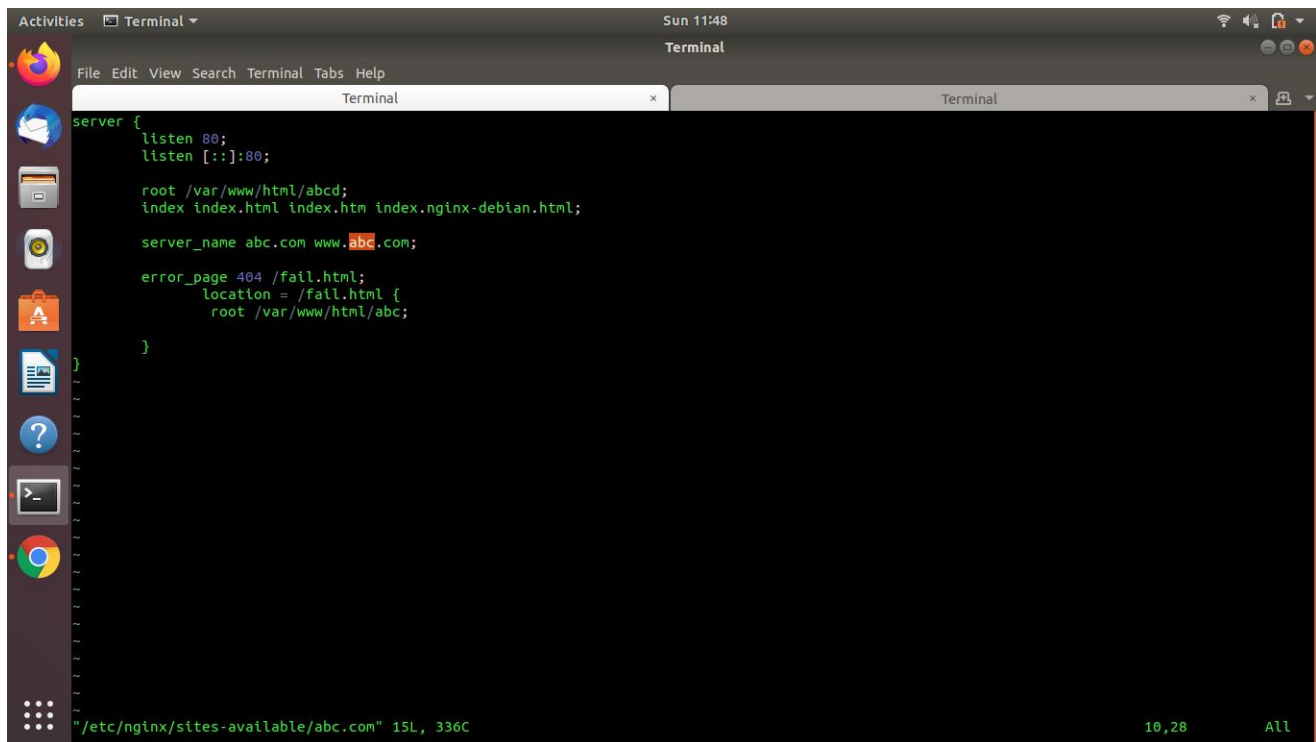
```
Activities  Terminal  Mon 12:09
Terminal
File Edit View Search Terminal Help
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:13:33 +0530] "GET / HTTP/1.1" 200 72 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:13:33 +0530] "GET / HTTP/1.1" 200 72 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:13:33 +0530] "GET / HTTP/1.1" 200 72 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:13:34 +0530] "GET / HTTP/1.1" 200 72 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:13:34 +0530] "GET / HTTP/1.1" 200 72 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:13:34 +0530] "GET / HTTP/1.1" 200 72 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:13:34 +0530] "GET / HTTP/1.1" 200 72 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:13:35 +0530] "GET / HTTP/1.1" 200 72 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:13:35 +0530] "GET / HTTP/1.1" 200 72 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:13:36 +0530] "GET / HTTP/1.1" 200 72 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:17:02 +0530] "GET / HTTP/1.1" 301 252 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:23:27 +0530] "GET / HTTP/1.1" 200 2518 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:23:28 +0530] "GET /nginx.png HTTP/1.1" 200 2103 "http://abc.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:32:38 +0530] "GET / HTTP/1.1" 200 2518 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:32:39 +0530] "GET /nginx.png HTTP/1.1" 200 2103 "http://abc.com/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:11:33:49 +0530] "GET /redirect HTTP/1.1" 302 170 "-" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:61.0) Gecko/20100101 Firefo
x/61.0"
127.0.0.1 - - [17/Feb/2020:12:08:59 +0530]
127.0.0.1 - - [17/Feb/2020:12:09:00 +0530]
127.0.0.1 - - [17/Feb/2020:12:09:02 +0530]
127.0.0.1 - - [17/Feb/2020:12:09:03 +0530]
gargi@gargi:sites-available $
```

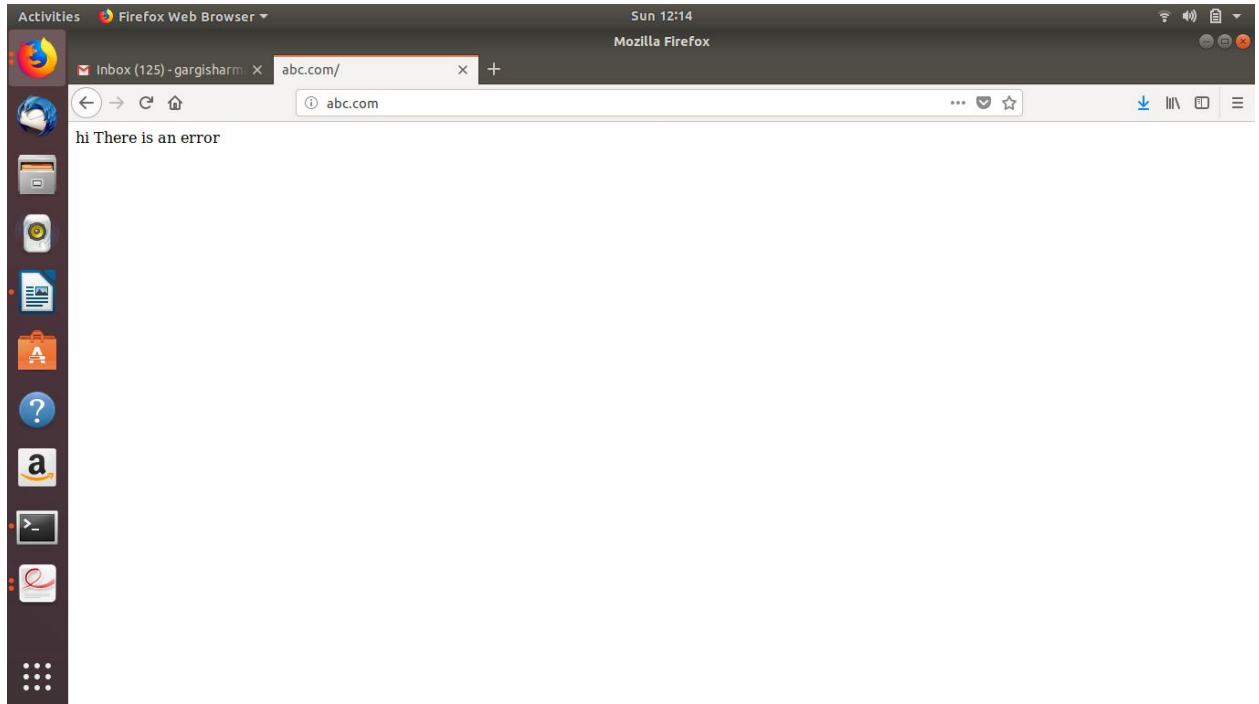
## 6. Host a site [ABC.COM](http://ABC.COM)

We will create a folder in `var/www/html` with name `abc` to keep all relevant files, afterwards we will create index page and fail safe page inside the same. Afterwards we will create a server block inside `sites-available`. Afterwards we will enable them by creating link in `sites-enabled`. Check for syntax error. Restart the server, edit the host file and your site is good to go.

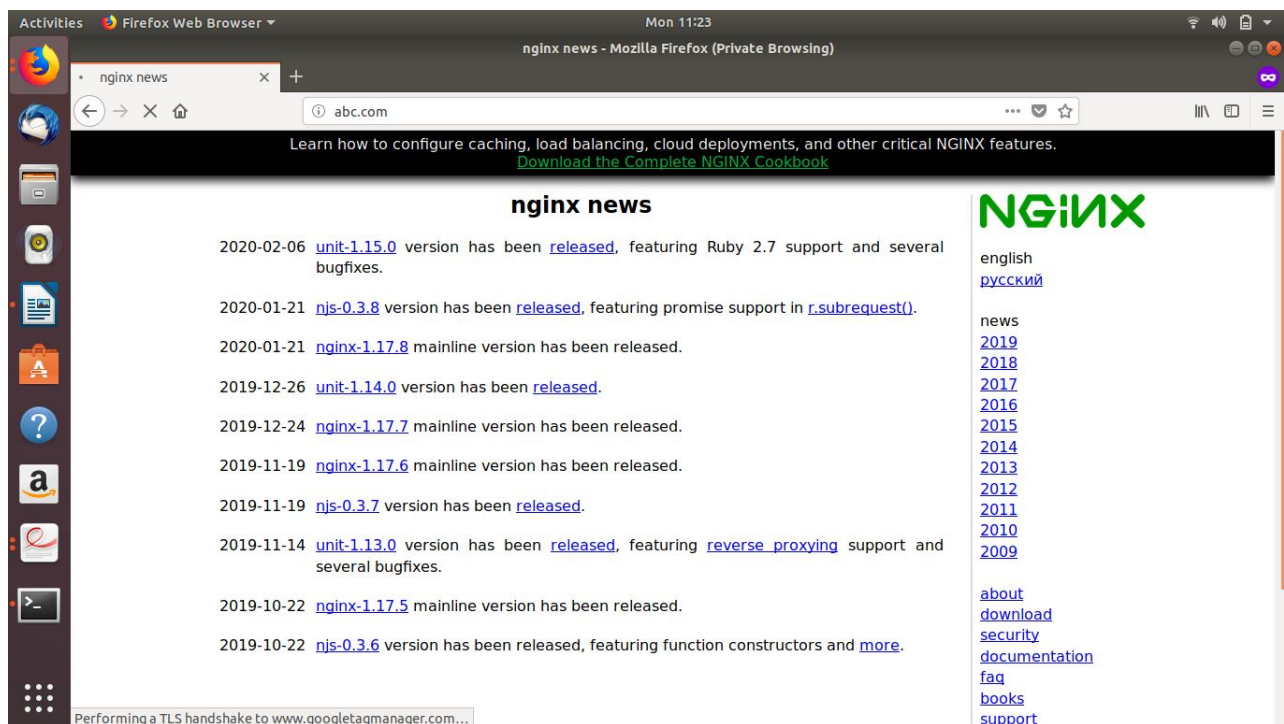


1. Create an index page and a fail-safe page. If a page for URI is not available, the fail-safe page is served.



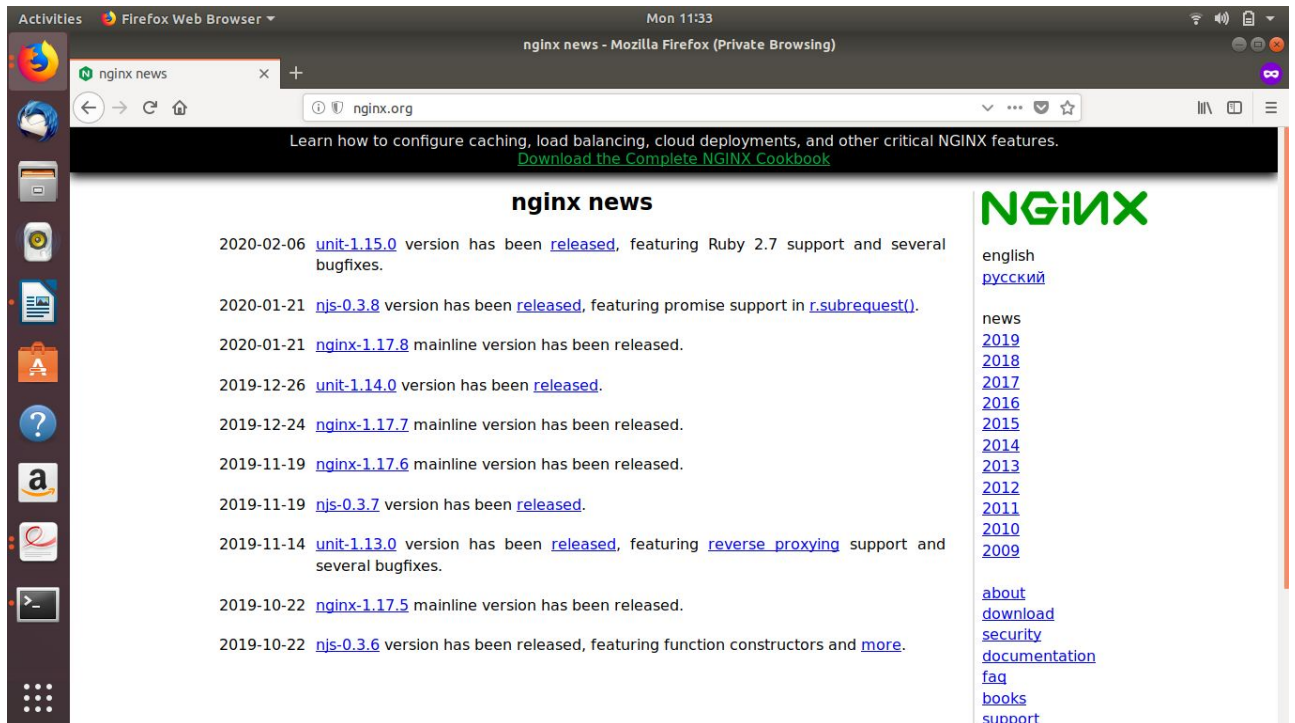


## 2. Proxy pass to a website [xyz.com](http://xyz.com) on a particular URI.



## 3. Redirect to above URI on `/redirect/location/`

```
Rewrite ^/redirect$ http://abc.com;  
}
```



#### 4. Perform an HTTP to HTTPS redirection including non-www to www redirection.

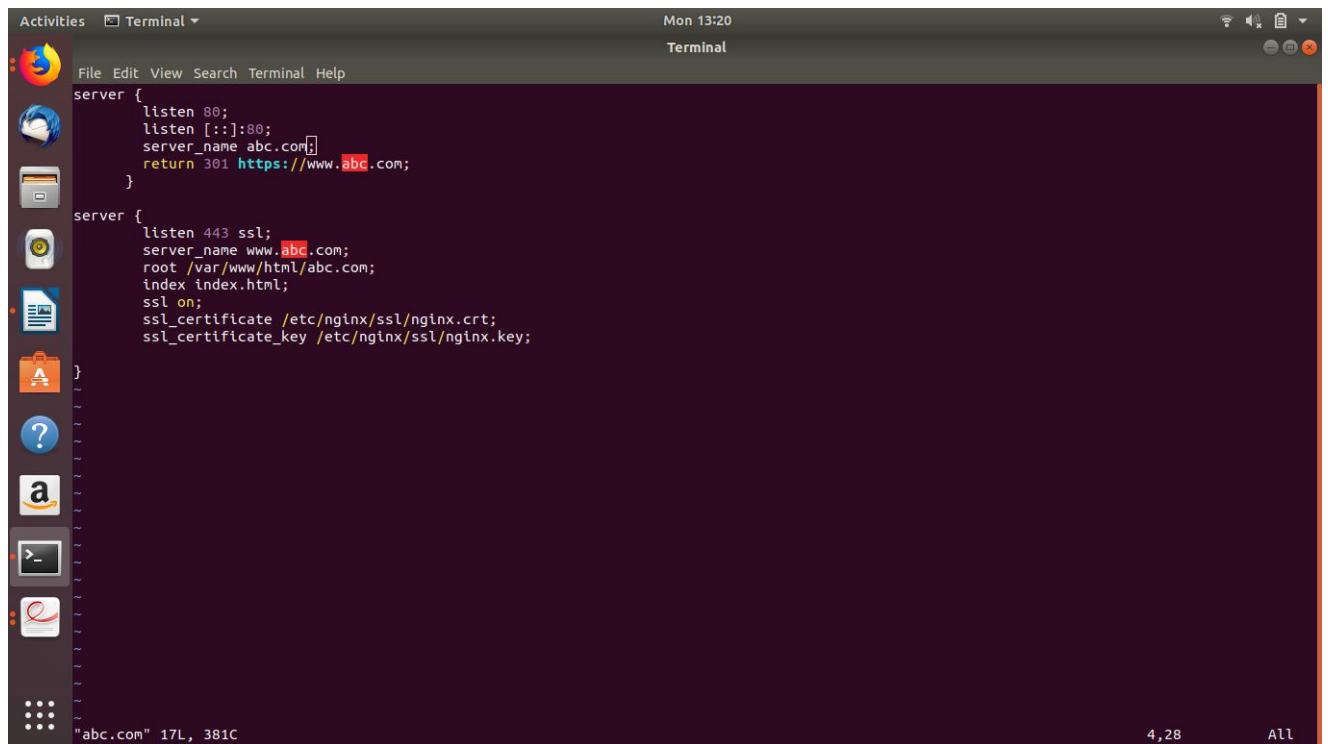
->Create directory /etc/nginx/ssl

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

-> ls

-> Edit /etc/nginx/abc.com

-> Edit /etc/hosts

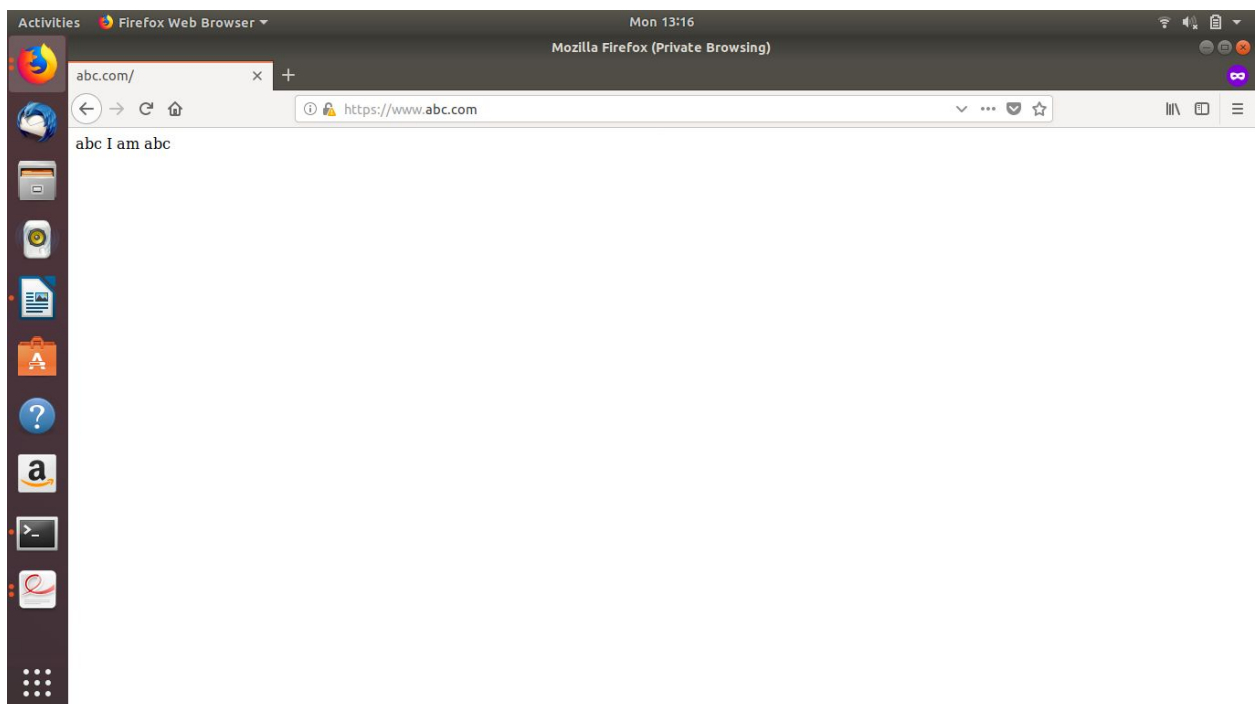


The image shows a Linux terminal window with a dark purple background. The title bar indicates 'Terminal' and the time 'Mon 13:20'. The terminal displays the configuration for two Nginx server blocks. The first block listens on port 80 and returns a 301 redirect to https://www.abc.com. The second block listens on port 443 for SSL, with the root directory set to /var/www/html/abc.com and the index file set to index.html. The SSL certificate and key are located at /etc/nginx/ssl/nginx.crt and /etc/nginx/ssl/nginx.key respectively. The terminal status bar at the bottom shows 'abc.com' 17L, 381C, a cursor position of 4,28, and a total of 111 lines.

```
server {  
    listen 80;  
    listen [::]:80;  
    server_name abc.com;  
    return 301 https://www.abc.com;  
}  
  
server {  
    listen 443 ssl;  
    server_name www.abc.com;  
    root /var/www/html/abc.com;  
    index index.html;  
    ssl on;  
    ssl_certificate /etc/nginx/ssl/nginx.crt;  
    ssl_certificate_key /etc/nginx/ssl/nginx.key;  
}
```

-> sudo service nginx reload

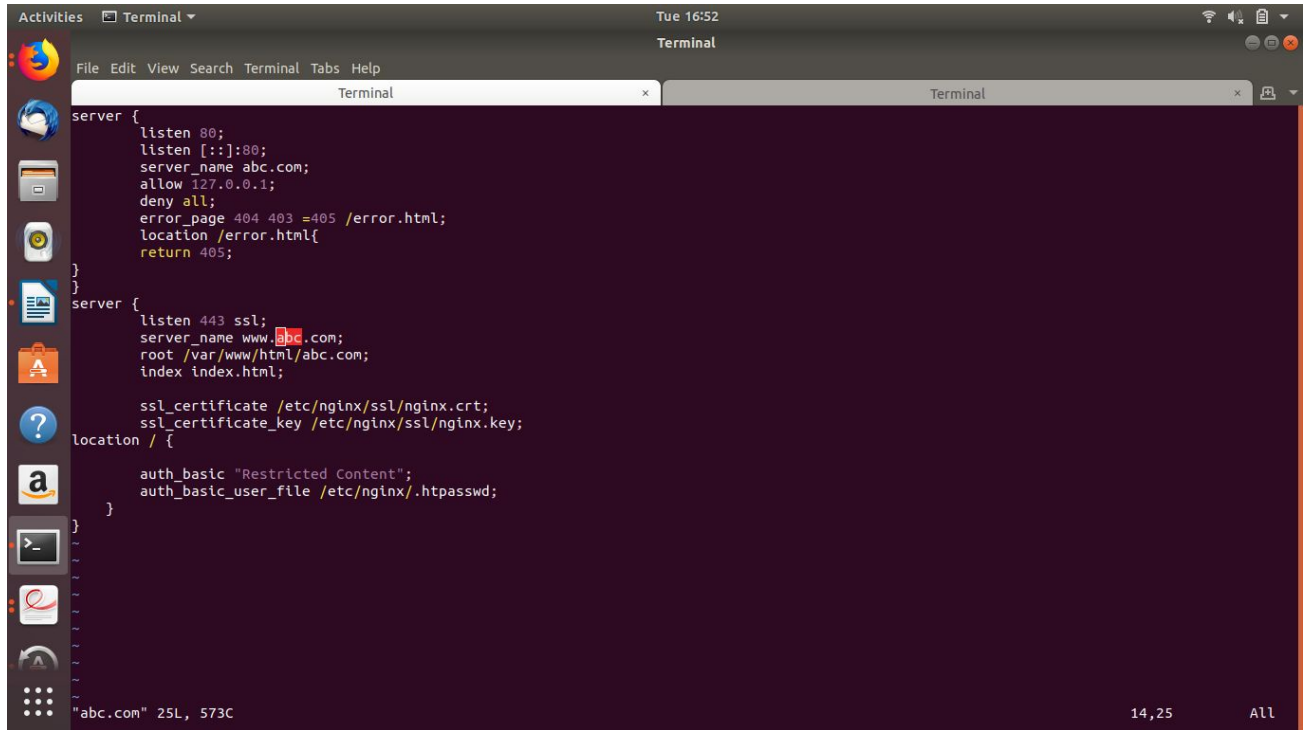
->open http://abc.com





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.

abc.com :



```
server {
    listen 80;
    listen [::]:80;
    server_name abc.com;
    allow 127.0.0.1;
    deny all;
    error_page 404 403 =405 /error.html;
    location /error.html{
        return 405;
    }
}

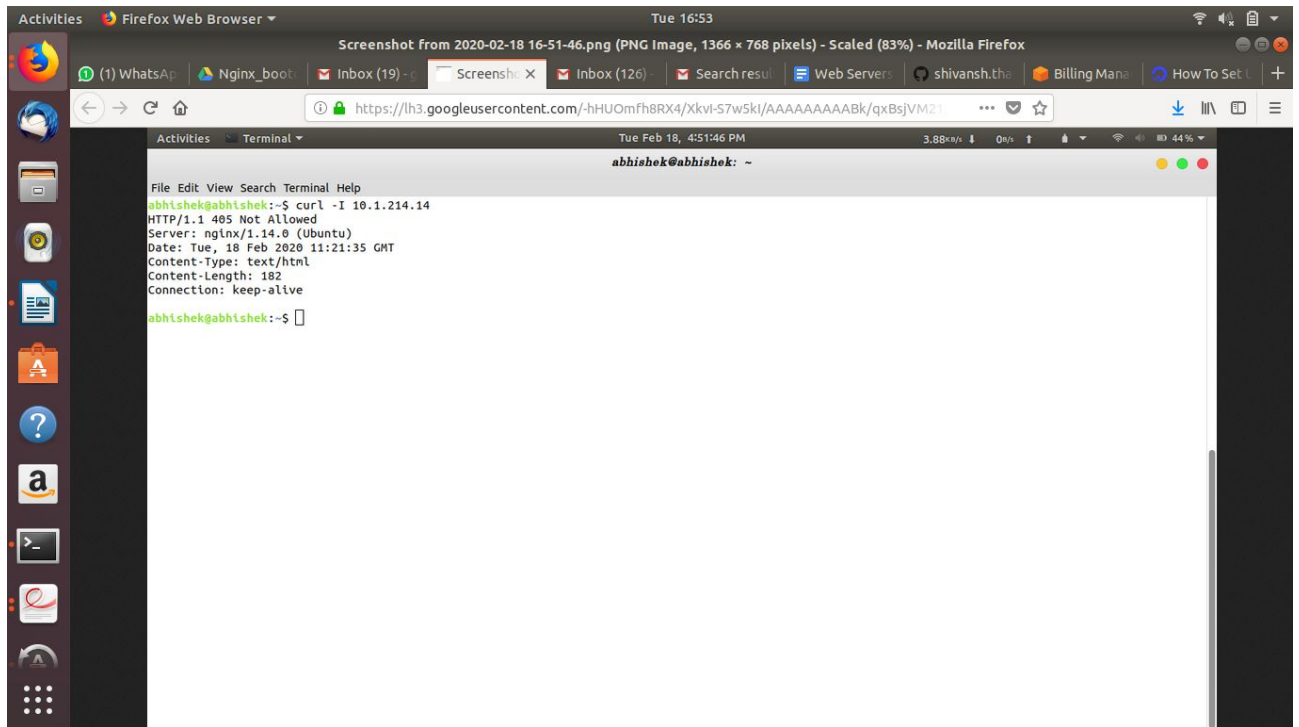
server {
    listen 443 ssl;
    server_name www.abc.com;
    root /var/www/html/abc.com;
    index index.html;

    ssl_certificate /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key /etc/nginx/ssl/nginx.key;
    location / {
        auth_basic "Restricted Content";
        auth_basic_user_file /etc/nginx/.htpasswd;
    }
}
```

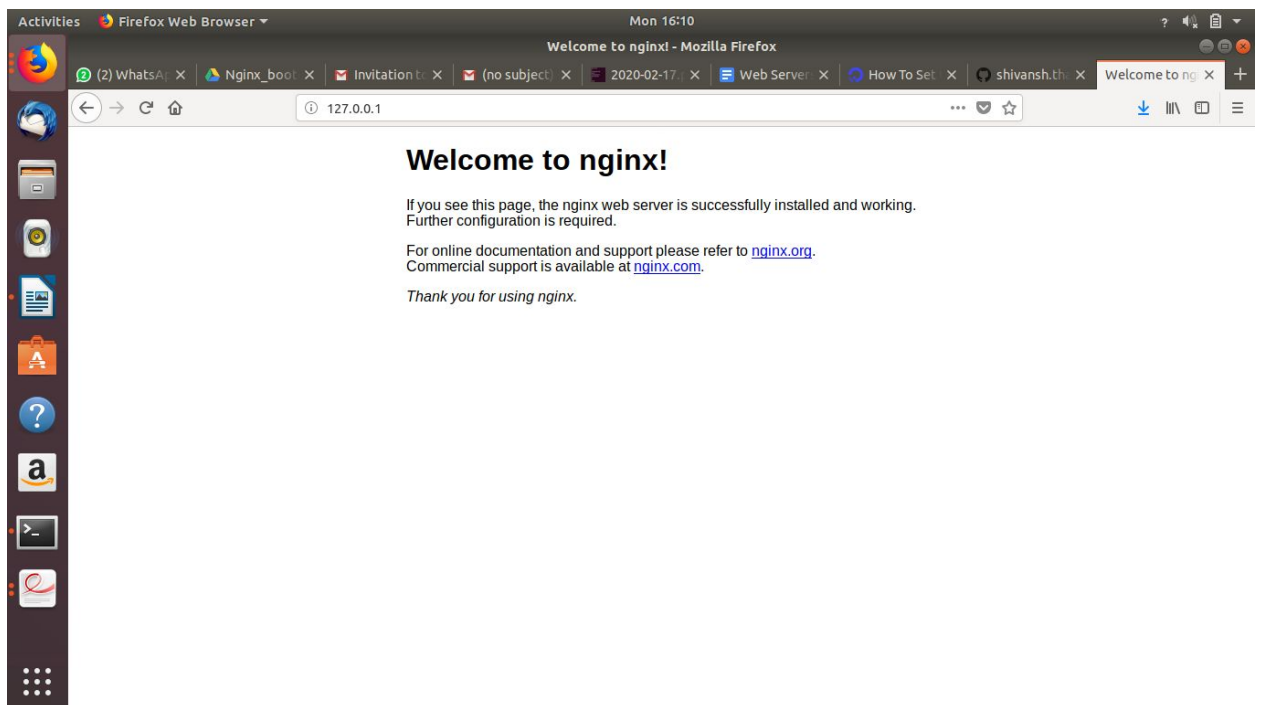
"abc.com" 25L, 573C 14,25 All

Remote Machine:





## My Local Machine:

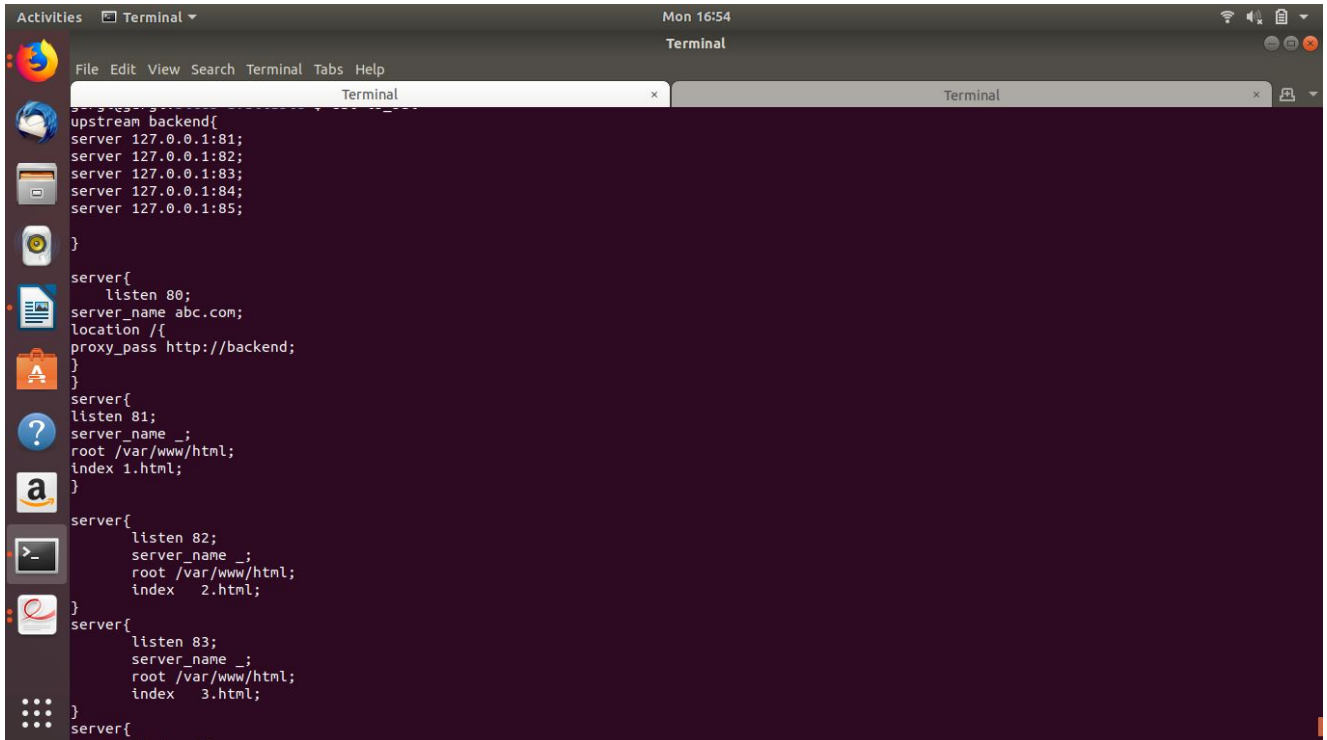


## 6. Create a load balancer with 5 backends. Explain different types of load balancing methods.

->Create 5 html files in /var/www/html:

1.html,2.html,3.html,4.html,5.html

->Go to sites available and create a load balancer file lb\_bal



```
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;
}

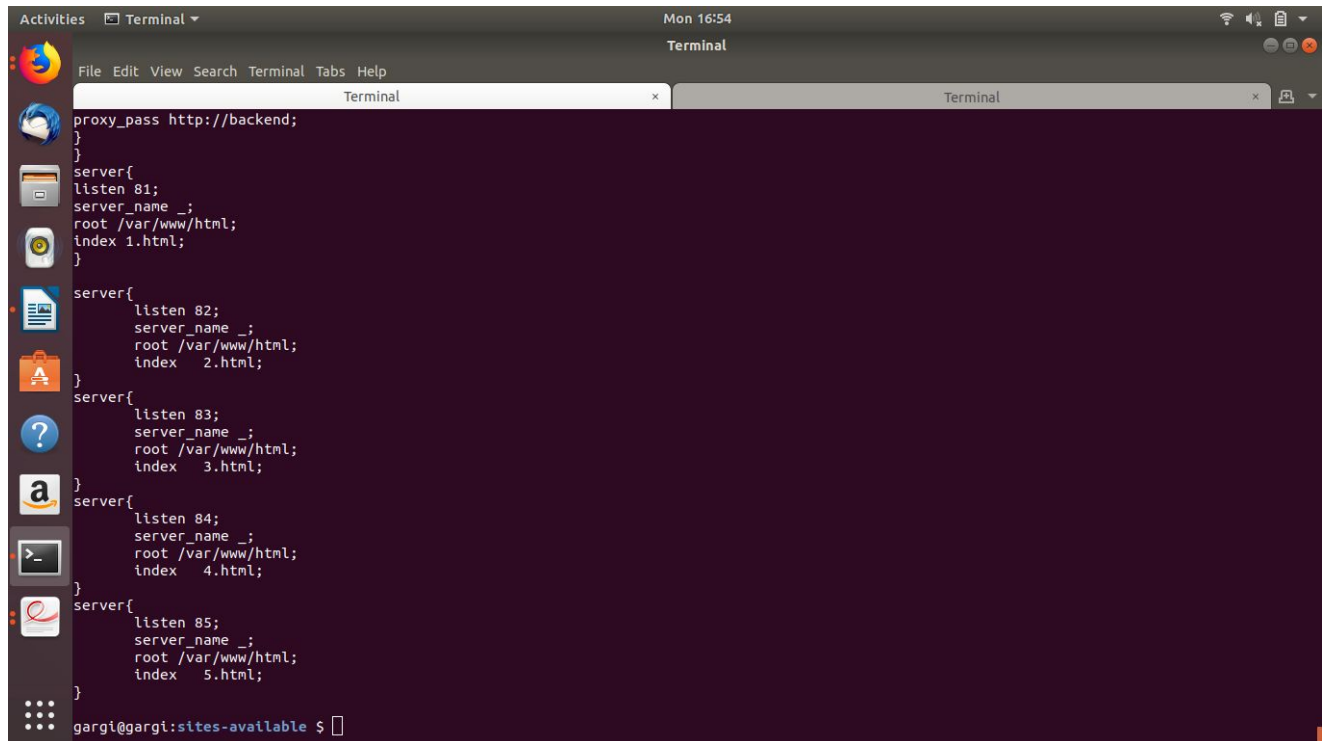
server{
    listen 80;
    server_name abc.com;
    location /{
        proxy_pass http://backend;
    }
}

server{
    listen 81;
    server_name _;
    root /var/www/html;
    index 1.html;
}

server{
    listen 82;
    server_name _;
    root /var/www/html;
    index 2.html;
}

server{
    listen 83;
    server_name _;
    root /var/www/html;
    index 3.html;
}

server{
```



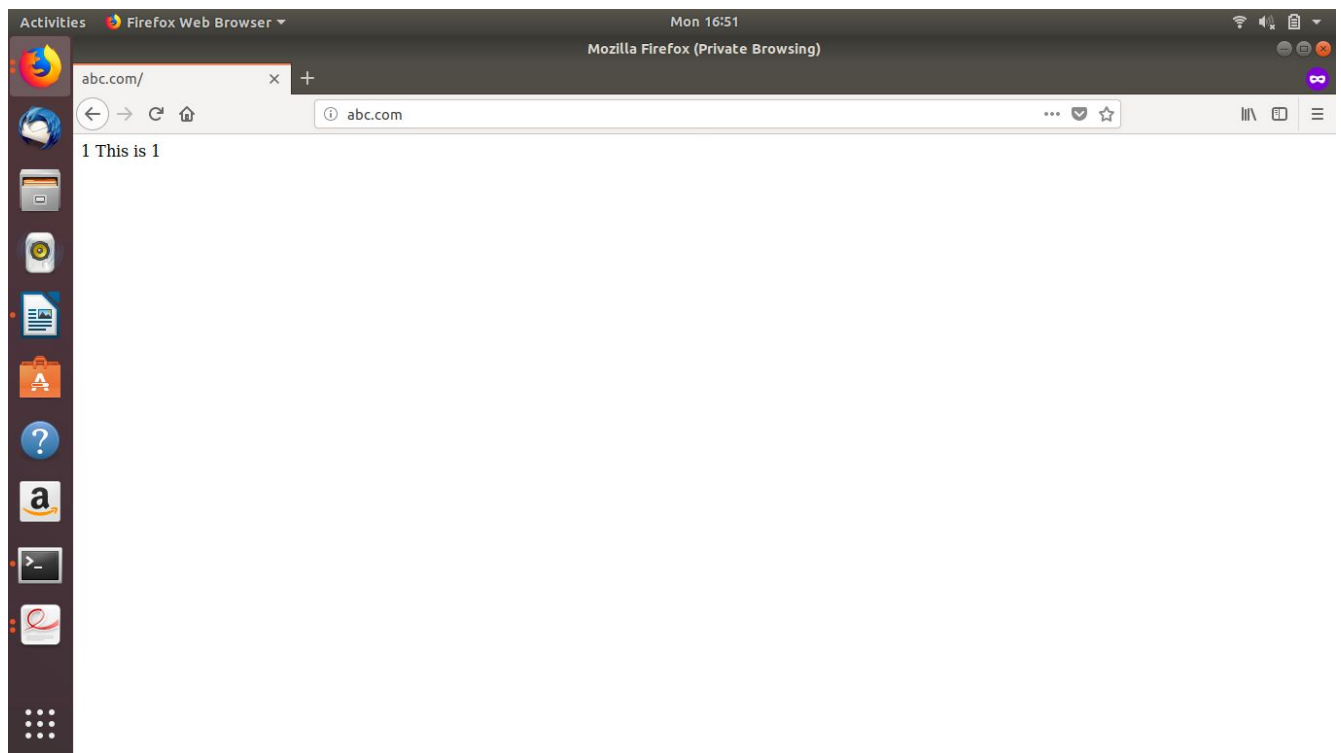
A terminal window titled 'Terminal' with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help). The window shows the configuration for the 'sites-enabled' directory in Nginx. The configuration includes a proxy\_pass directive and five server blocks listening on ports 81 through 85, each serving a different index file (1.html to 5.html) from the root directory /var/www/html. The prompt at the bottom is 'gargi@gargi:sites-available \$'.

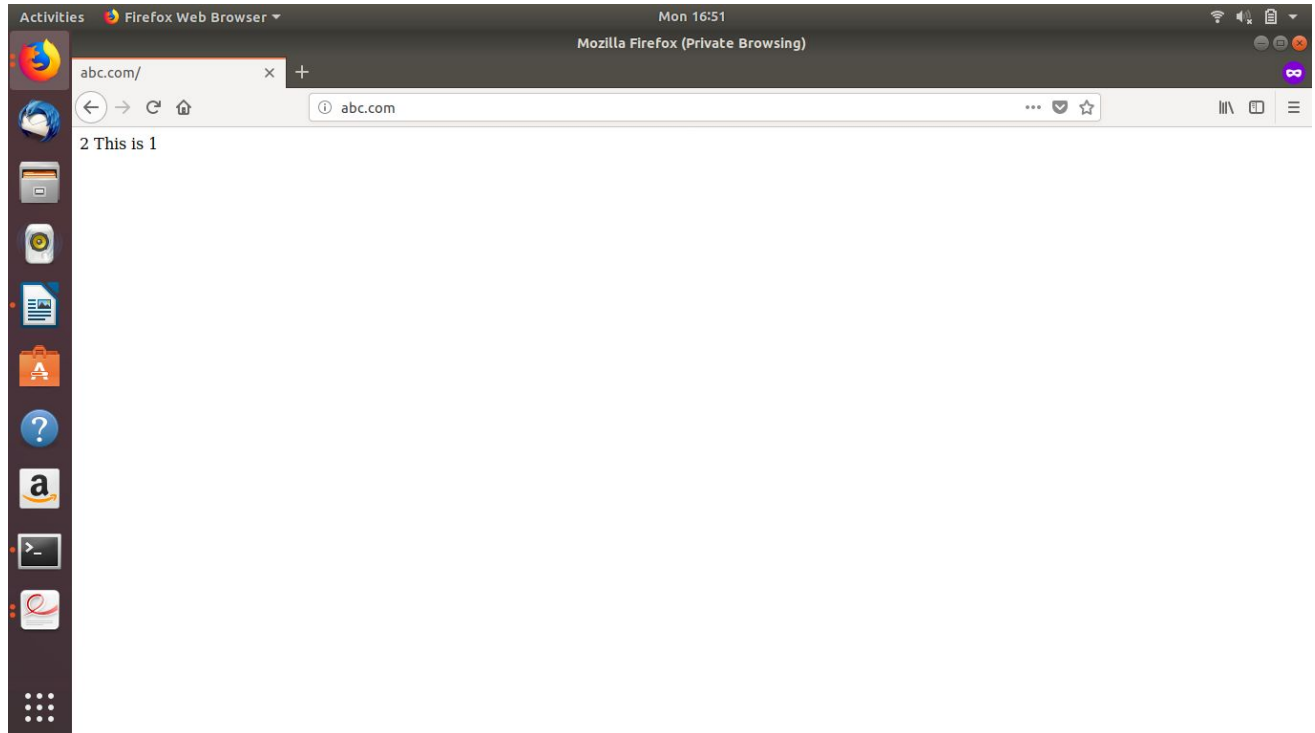
```
proxy_pass http://backend;
}
}
server{
listen 81;
server_name _;
root /var/www/html;
index 1.html;
}

server{
listen 82;
server_name _;
root /var/www/html;
index 2.html;
}
server{
listen 83;
server_name _;
root /var/www/html;
index 3.html;
}
server{
listen 84;
server_name _;
root /var/www/html;
index 4.html;
}
server{
listen 85;
server_name _;
root /var/www/html;
index 5.html;
}

gargi@gargi:sites-available $
```

- > Create a symlink for the same in sites-enabled
- > Delete the symlink for abc.com in sites-enabled
- >Reload the service





Different types of load balancing techniques :

Round Robin : This is the default technique. It runs through the list of upstream servers in sequence and assigns the request to them one by one in round robin fashion.

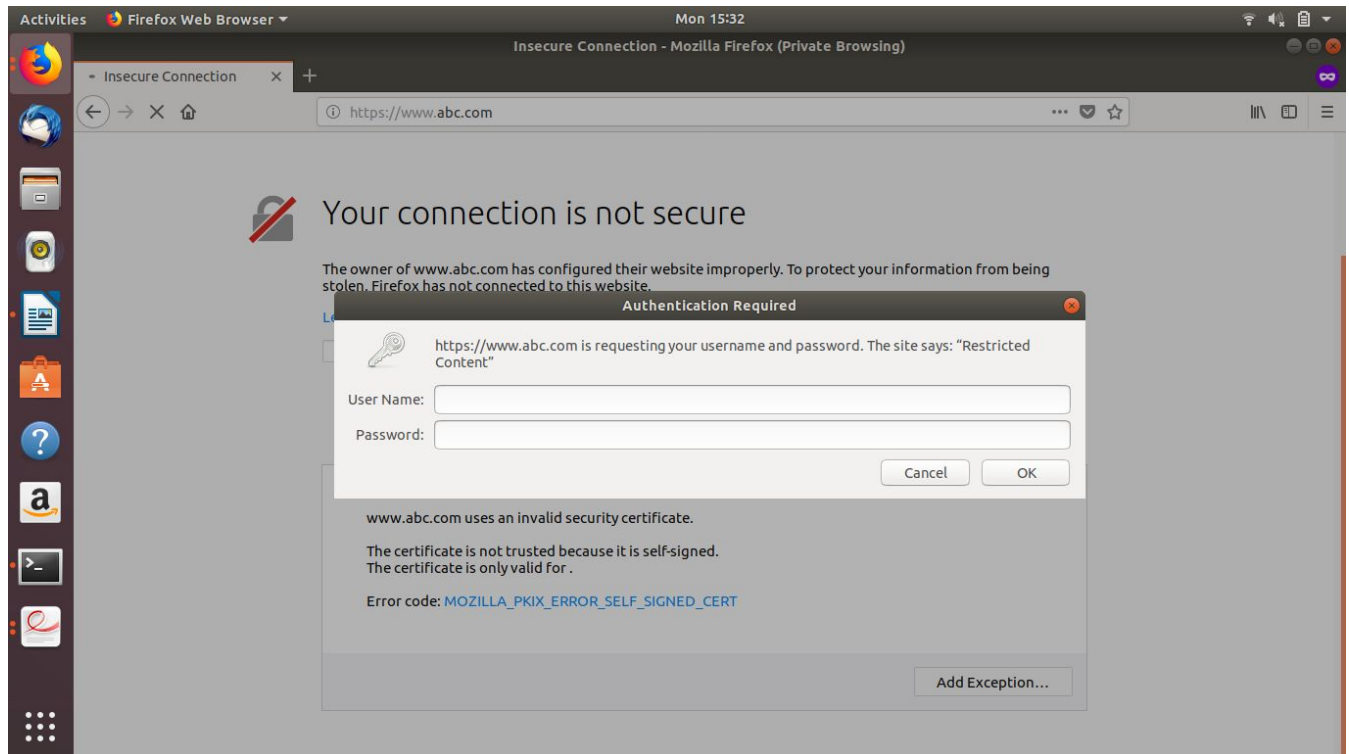
Hash: For each request the load balancer calculates a hash that is based on the combination of text and NGINX variables that we specify, and associates the hash with one of the servers. It sends all requests with that hash to that server, so this method establishes a kind of session persistence.

IP Hash: It is available only for HTTP. The hash is based on the client's ip address.

Least Connections: The load balancer compares the current number of active connections it has to each server, and sends the request to the server with the fewest connections.

Least Time: The load balancer mathematically combines two metrics for each server, the current number of active connections and a weighted average response time for past requests and sends the request to the server with the lowest value.

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



If we enter wrong credentials, the following page will be displayed:

