

**TO
THE
NEW™**



Assessment -5

WEB SERVER

Trainee Name : Arun Parmar

Mentor Name : Ravi Kumar

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

A **reverse proxy server** is a type of **proxy server** that typically sits behind the firewall in a private network and directs client requests to the appropriate backend **server** 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?

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?

/etc/nginx/conf.d directory

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

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

```
log_format locate
```

```
'$remote_addr - $remote_user [$time_local]
```

```
$$$request" $status $body_bytes_sent '
```

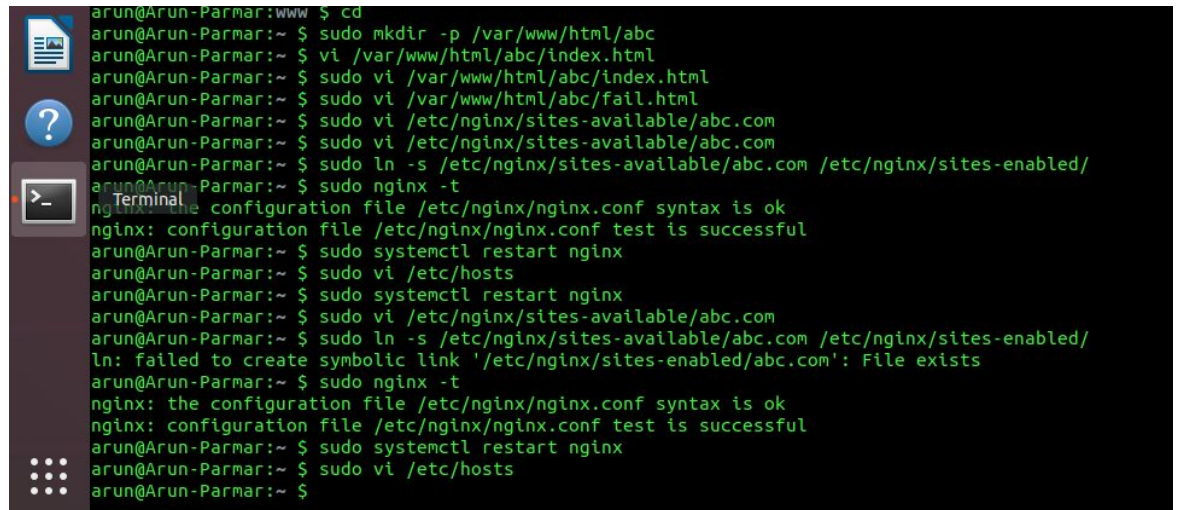
```
$$$http_referer" "$http_user_agent";
```

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.

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.

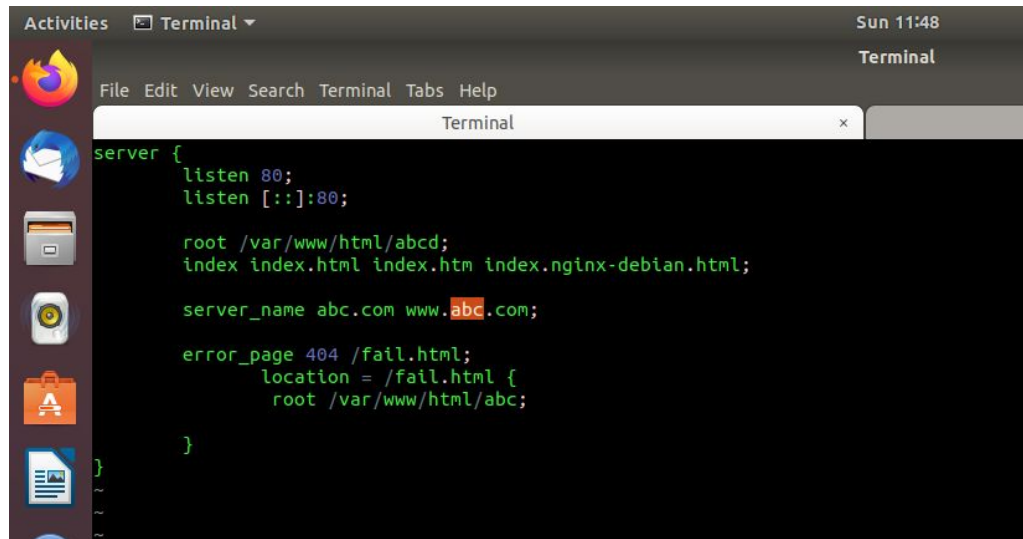
Afterwards we will create a server block inside `sites-available`. Afterwards we will enable them by creating link in `sites-enabled`. Check for syntax error. Restart the server, edit the host file and your site is good to go.



```
arun@Arun-Parmar:~$ cd /var/www/html
arun@Arun-Parmar:~$ sudo mkdir -p /var/www/html/abc
arun@Arun-Parmar:~$ vi /var/www/html/abc/index.html
arun@Arun-Parmar:~$ sudo vi /var/www/html/abc/index.html
arun@Arun-Parmar:~$ sudo vi /var/www/html/abc/fail.html
arun@Arun-Parmar:~$ sudo vi /etc/nginx/sites-available/abc.com
arun@Arun-Parmar:~$ sudo vi /etc/nginx/sites-available/abc.com
arun@Arun-Parmar:~$ sudo ln -s /etc/nginx/sites-available/abc.com /etc/nginx/sites-enabled/
arun@Arun-Parmar:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
arun@Arun-Parmar:~$ sudo systemctl restart nginx
arun@Arun-Parmar:~$ sudo vi /etc/hosts
arun@Arun-Parmar:~$ sudo systemctl restart nginx
arun@Arun-Parmar:~$ sudo vi /etc/nginx/sites-available/abc.com
arun@Arun-Parmar:~$ sudo ln -s /etc/nginx/sites-available/abc.com /etc/nginx/sites-enabled/
ln: failed to create symbolic link '/etc/nginx/sites-enabled/abc.com': File exists
arun@Arun-Parmar:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
arun@Arun-Parmar:~$ sudo systemctl restart nginx
arun@Arun-Parmar:~$ sudo vi /etc/hosts
arun@Arun-Parmar:~$
```

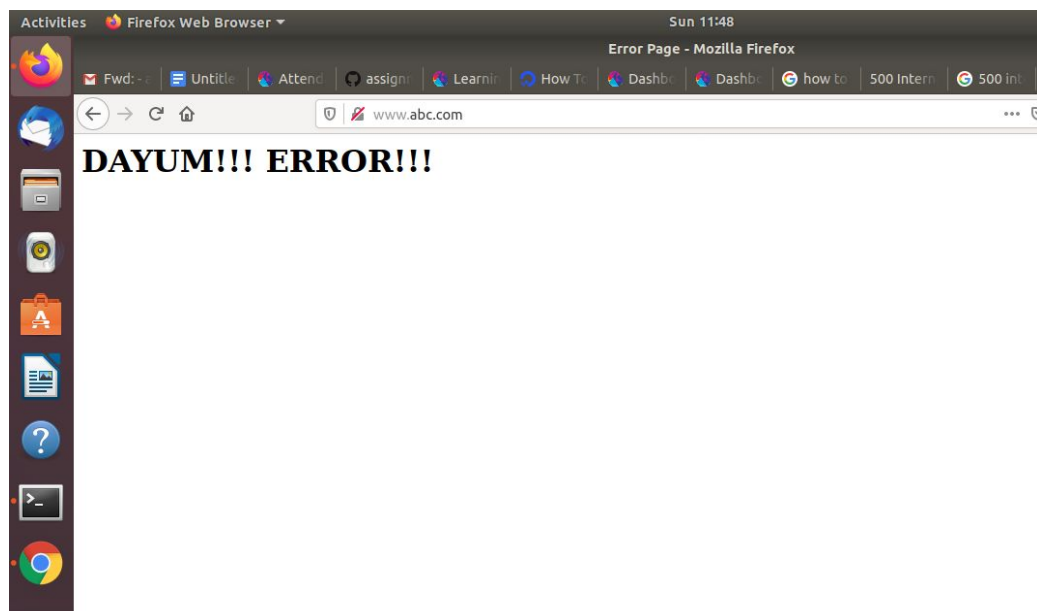


Now we will change the name of folder we are using so that it produces a 404 error and will edit the error page.

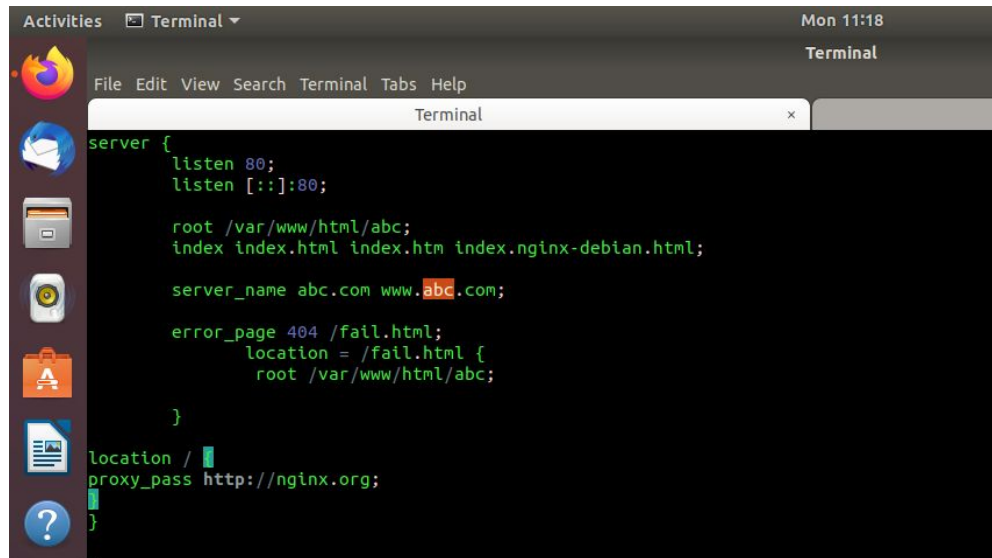


A terminal window titled 'Terminal' showing an Nginx configuration file. The configuration includes a 'server' block with 'listen 80;' and 'listen [::]:80;', a 'root' directive pointing to '/var/www/html/abcd', and an 'error_page 404 /fail.html;' directive with a corresponding location block. The server name is set to 'abc.com' and 'www.abc.com'. The terminal is part of a desktop environment with a sidebar containing various application icons.

```
server {  
    listen 80;  
    listen [::]:80;  
  
    root /var/www/html/abcd;  
    index index.html index.htm index.nginx-debian.html;  
  
    server_name abc.com www.abc.com;  
  
    error_page 404 /fail.html;  
    location = /fail.html {  
        root /var/www/html/abc;  
    }  
}
```



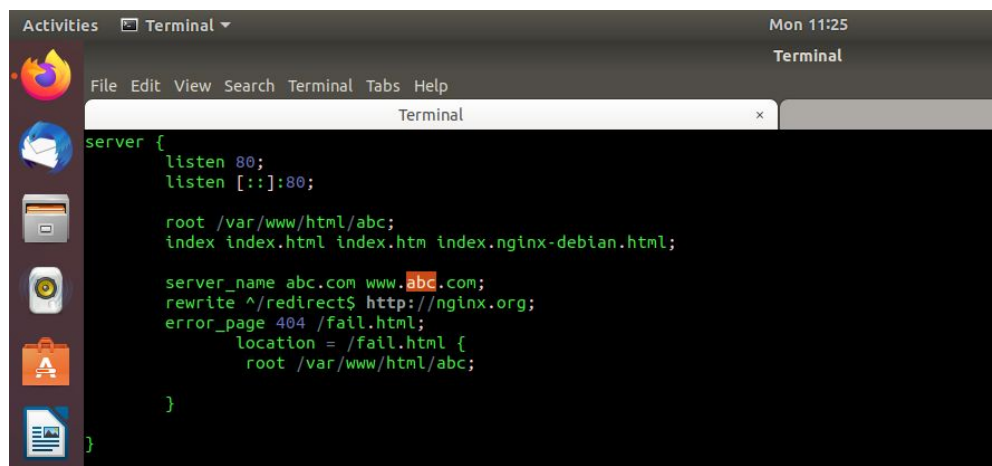
2. proxy pass to a website xyz.com on a particular URI.



A terminal window titled 'Terminal' with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help) and a status bar (Mon 11:18). The terminal displays the following Nginx configuration:

```
server {  
    listen 80;  
    listen [::]:80;  
  
    root /var/www/html/abc;  
    index index.html index.htm index.nginx-debian.html;  
  
    server_name abc.com www.abc.com;  
  
    error_page 404 /fail.html;  
    location = /fail.html {  
        root /var/www/html/abc;  
    }  
  
    location / {  
        proxy_pass http://nginx.org;  
    }  
}
```

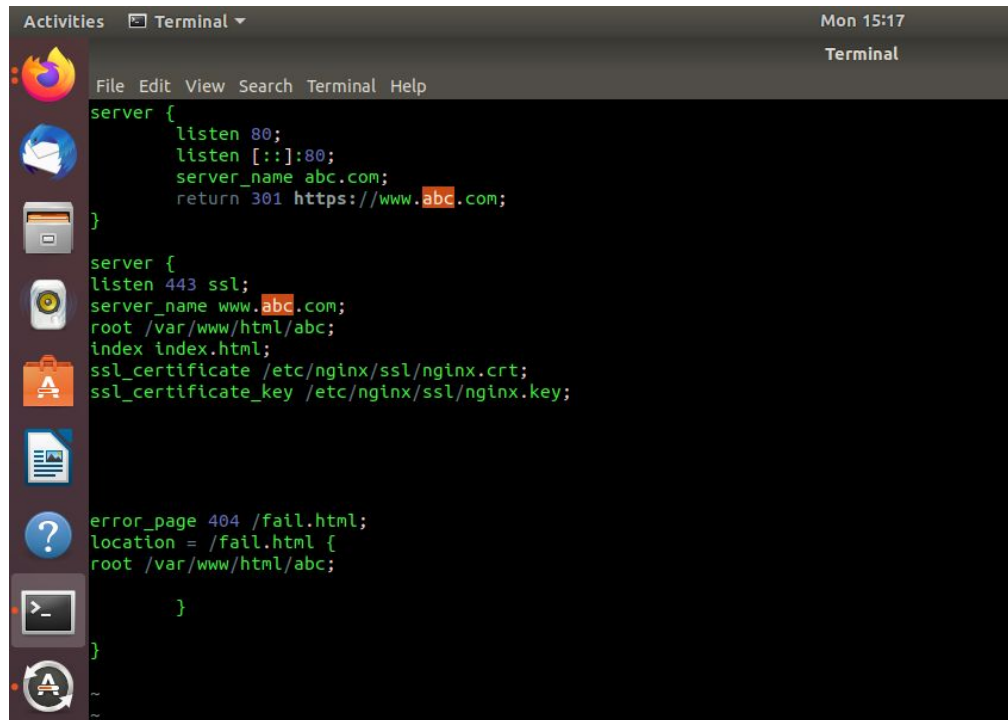
3. redirect to above URI on /redirect/



A terminal window titled 'Terminal' with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help) and a status bar (Mon 11:25). The terminal displays the following Nginx configuration:

```
server {  
    listen 80;  
    listen [::]:80;  
  
    root /var/www/html/abc;  
    index index.html index.htm index.nginx-debian.html;  
  
    server_name abc.com www.abc.com;  
    rewrite ^/redirect$ http://nginx.org;  
    error_page 404 /fail.html;  
    location = /fail.html {  
        root /var/www/html/abc;  
    }  
}
```

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

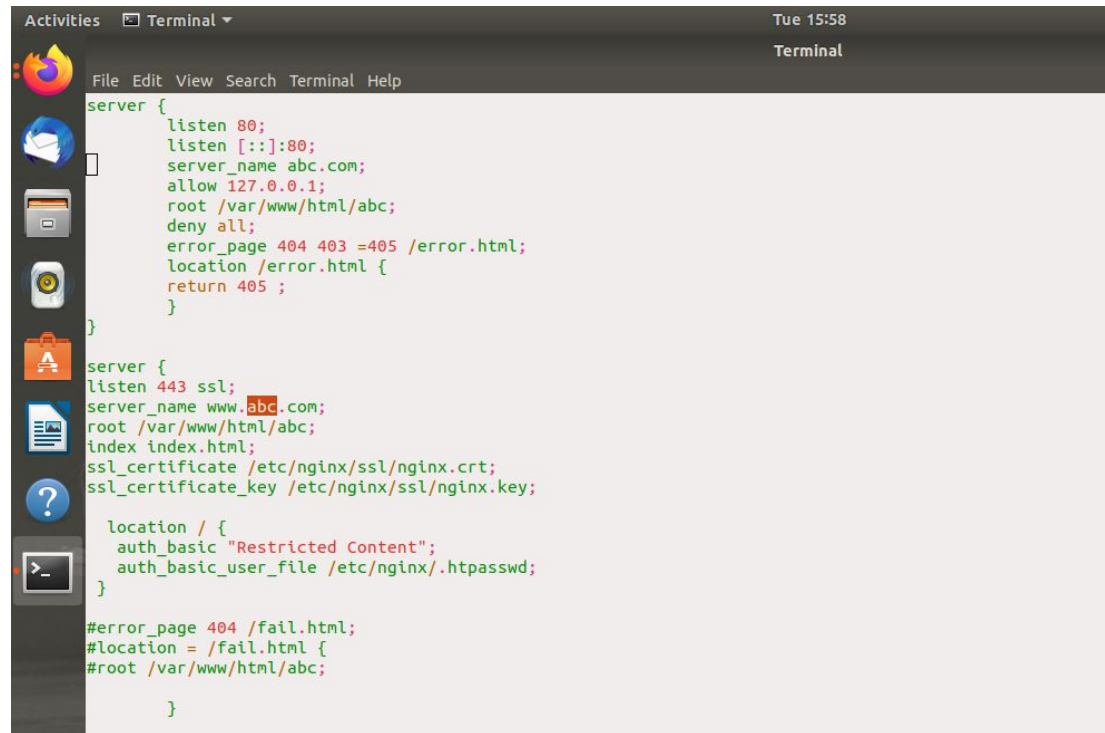


The image shows a Linux terminal window with the title bar 'Activities Terminal' and a timestamp 'Mon 15:17'. The terminal displays the configuration for an Nginx server. The first server block listens on port 80 and returns a 301 redirect to https://www.abc.com. The second server block listens on port 443 with SSL, serving index.html from /var/www/html/abc. An error_page 404 /fail.html is also configured.

```
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;  
    index index.html;  
    ssl_certificate /etc/nginx/ssl/nginx.crt;  
    ssl_certificate_key /etc/nginx/ssl/nginx.key;  
  
    error_page 404 /fail.html;  
    location = /fail.html {  
        root /var/www/html/abc;  
    }  
}
```



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.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal displays the configuration for two Nginx server blocks. The first block listens on port 80 and is configured to return a 405 status code for any request. The second block listens on port 443 (SSL) and is configured to return a 405 status code for any request. The configuration is as follows:

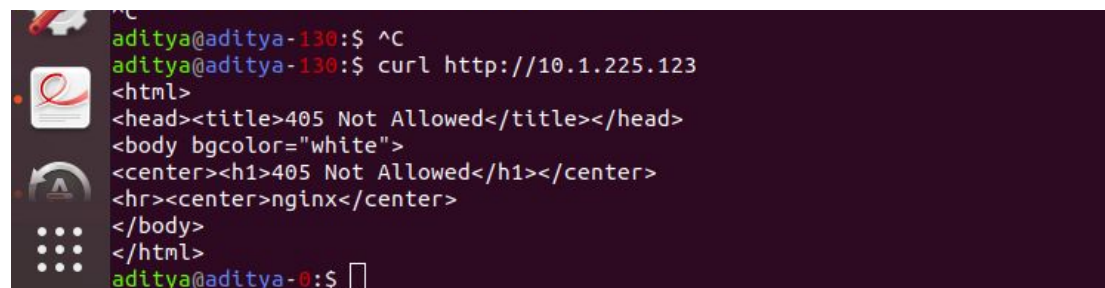
```
server {
    listen 80;
    listen [::]:80;
    server_name abc.com;
    allow 127.0.0.1;
    root /var/www/html/abc;
    deny all;
    error_page 404 403 =405 /error.html;
    location /error.html {
        return 405 ;
    }
}

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

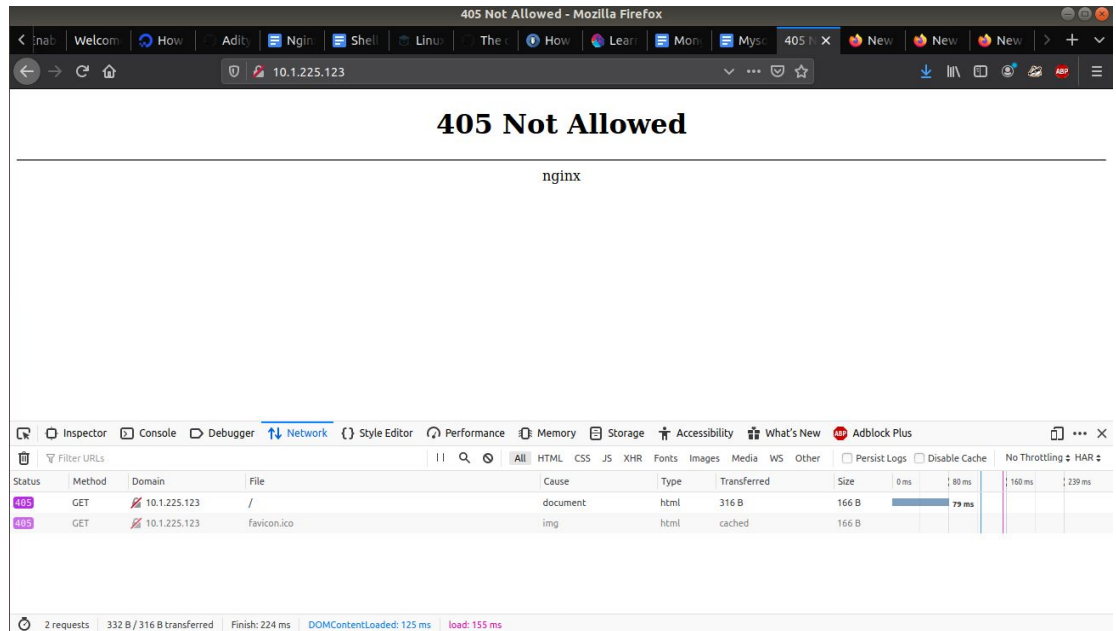
    #error_page 404 /fail.html;
    #location = /fail.html {
    #root /var/www/html/abc;
    }
}
```

Remote Machine:



The screenshot shows a remote terminal window with the following commands and output:

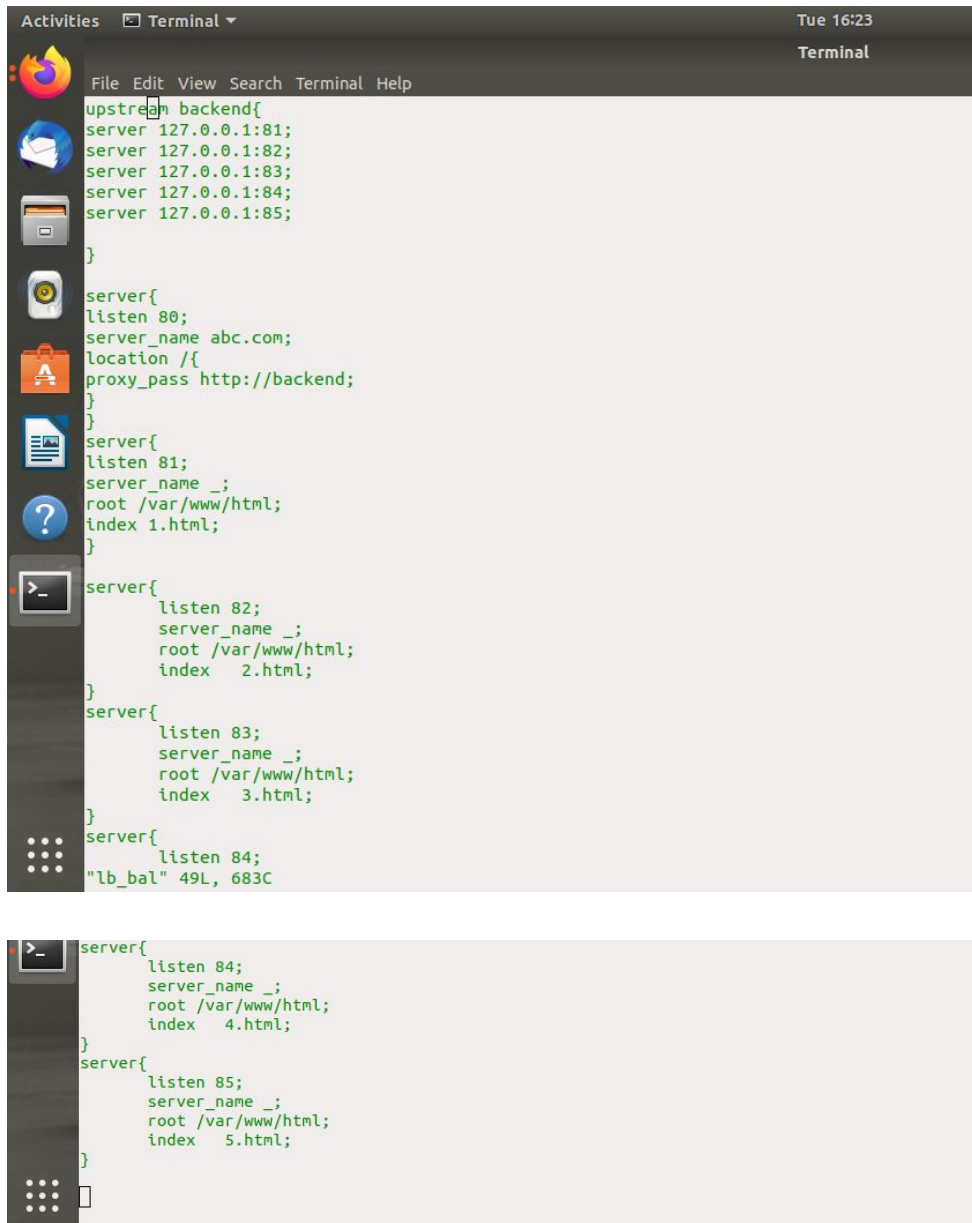
```
aditya@aditya-130:$ ^C
aditya@aditya-130:$ curl http://10.1.225.123
<html>
<head><title>405 Not Allowed</title></head>
<body bgcolor="white">
<center><h1>405 Not Allowed</h1></center>
<hr><center>nginx</center>
</body>
</html>
aditya@aditya-0:$
```

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

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

-Go to sites available and create a load balancer file lb_bal



The image shows a terminal window with a dark theme. The top bar indicates 'Activities', 'Terminal', and the time 'Tue 16:23'. The terminal displays Nginx configuration code. The first block defines an upstream named 'backend' with five servers on ports 127.0.0.1:81 through 85. The second block is a server configuration listening on port 80, acting as a proxy for the 'backend' upstream. The third block is a server configuration listening on port 81, serving '1.html' from '/var/www/html'. The fourth block is a server configuration listening on port 82, serving '2.html' from '/var/www/html'. The fifth block is a server configuration listening on port 83, serving '3.html' from '/var/www/html'. The sixth block is a server configuration listening on port 84, serving '4.html' from '/var/www/html'. The seventh block is a server configuration listening on port 85, serving '5.html' from '/var/www/html'. The terminal prompt is visible at the bottom left of the second screenshot.

```
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;
}

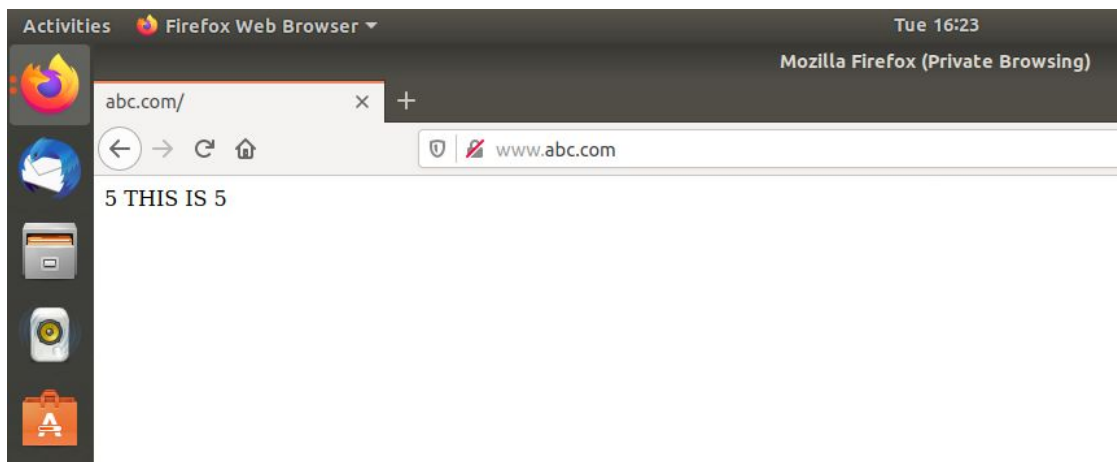
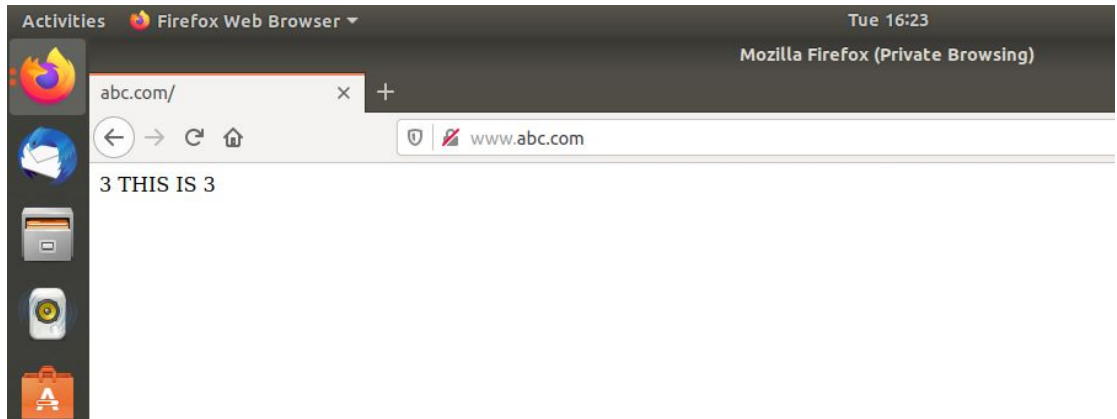
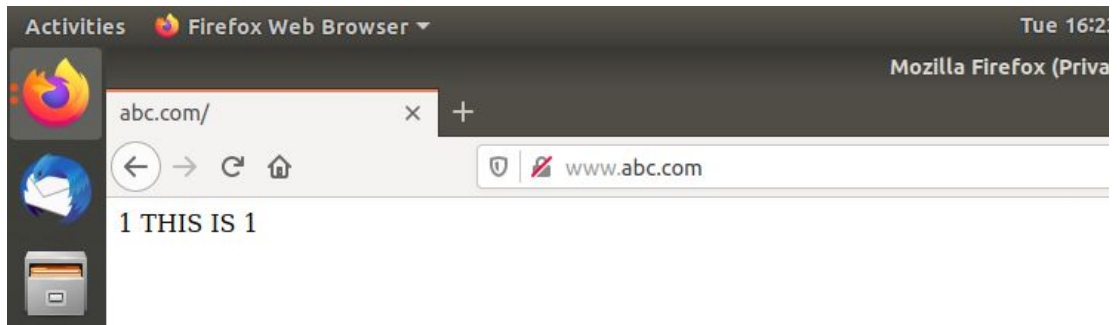
server{
listen 82;
server_name _;
root /var/www/html;
index 2.html;
}

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

server{
listen 84;
server_name _;
root /var/www/html;
index 4.html;
}

server{
listen 85;
server_name _;
root /var/www/html;
index 5.html;
}
```

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



8. Setup Basic Auth (Popup asking for username and password) in a particular location block.

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

    #error_page 404 /fail.html;
    #location = /fail.html {
    #root /var/www/html/abc;
    }
}
```

"abc.com" 34L, 639C

