Lesson 2
The Apache Web Server

Contents

Note:
By now, your server should be using static IP settings.
You should be able to select the appropriate user login to work on and learn from all the remaining LAS practical exercises.

You should use sudo to execute commands that require administrative privileges.
You should be able to carry out simple firewall configuration via the GUI interface.
You should have the SELinux enforce mode set to Enforcing.
You should know the systemctl command to enable, start, stop and/or restart a system service.

1. Setting up a basic Apache2 Web Server

On server:

1. Install the Apache Web Server package and start it up.

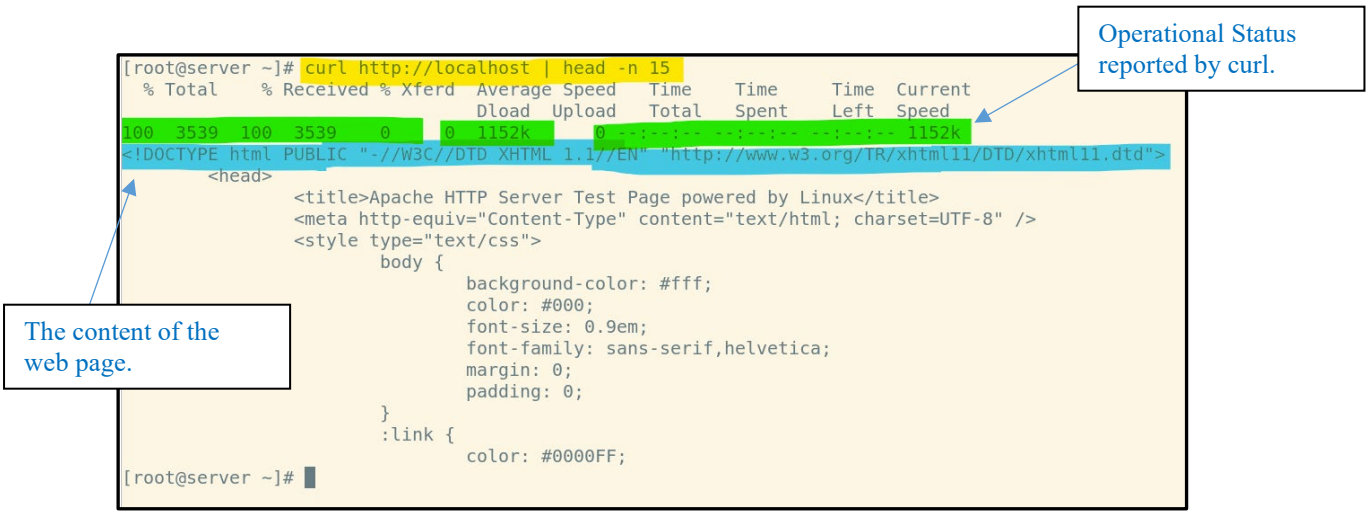
```
dnf -y install httpd
systemctl enable httpd
systemctl start httpd
```

Do a quick check with curl to test if the web server is running.

Type

```
curl http://localhost | head -n 15
```

You will see the first 15 lines of text content of the default web page retrieved from the http://localhost server. By that, you can tell if your server is running or not.

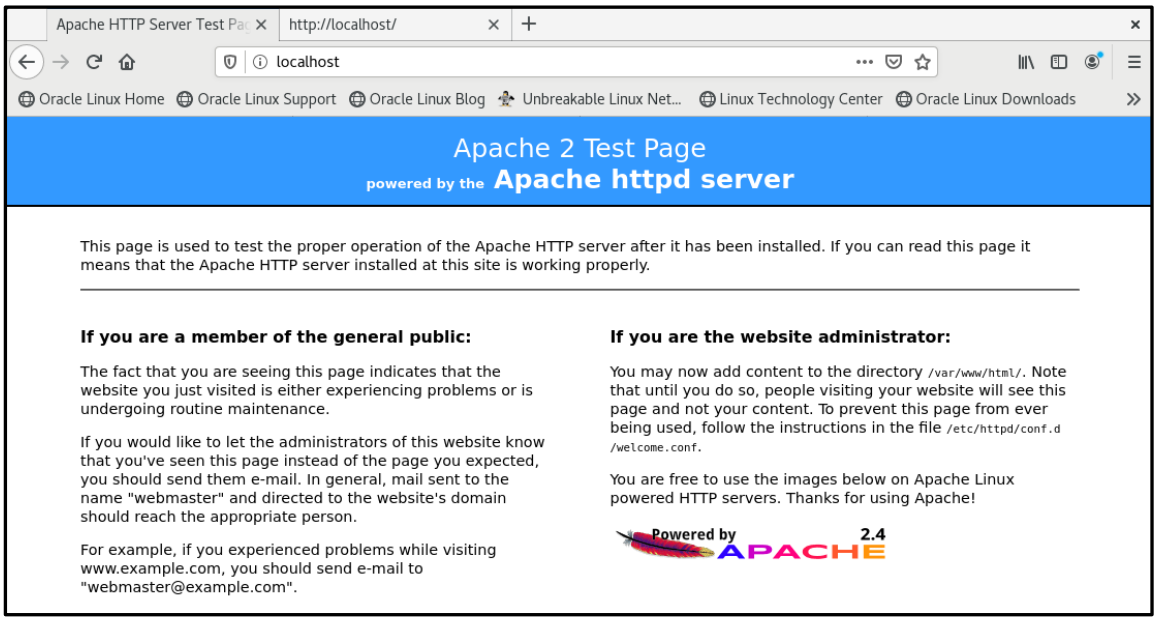


curl - is a simple command line tool for url content retrieval (You may see more of what curl can do from the recommended reference).

2. Start a web browser (Firefox) and browse http://localhost

You may compare the page content with the output from the curl command issued previously.

(Hint: you may right click on the web page and choose View Page Content to see the html tags.)



3. Create your own default webpage /var/www/html/index.html.

You may type

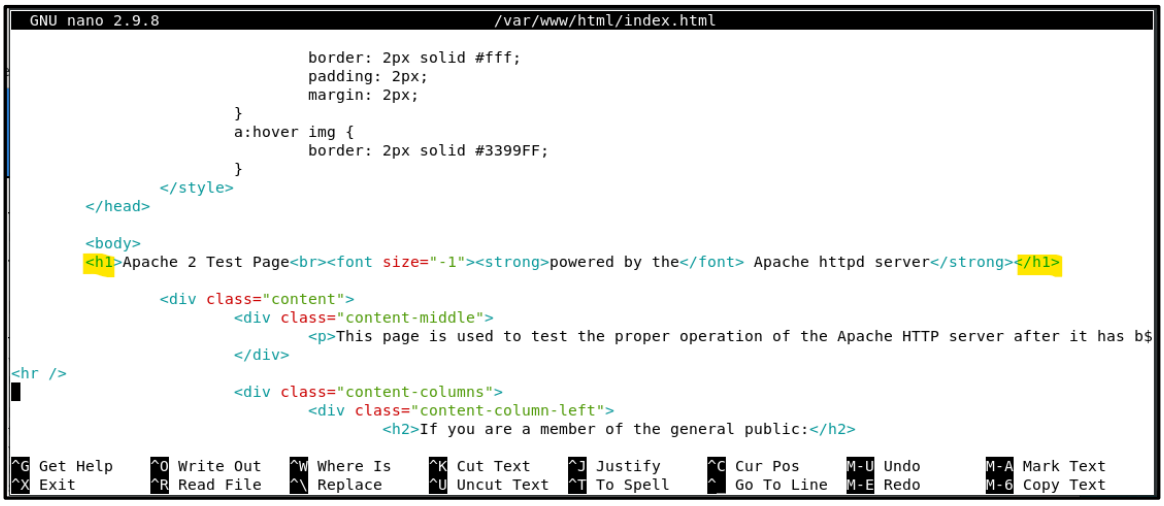
```
curl http://localhost  > /tmp/index.html
cp /tmp/index.html  /var/www/html/index.html
```

to clone the default page into the index.html file in the target folder.

You may modify it's original Header title with

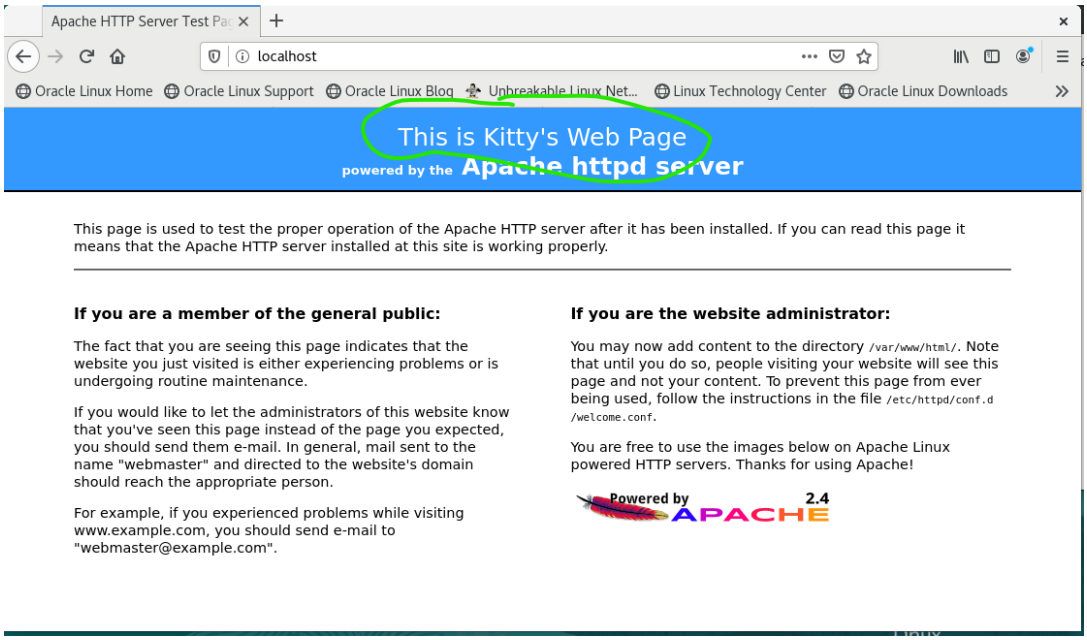
This is <your name>’s web site

(Note: Look for the <h1> tag to locate the target position)



Change the 'Apache 2 Test Page' with 'This is <Your name's> Web Page'

4. With the web browser, browse to http://localhost to view your web page again.



The system default index file of the apache web server is at /var/www/html/index.html

Previously, there is no index.html stored in the /var/www/html folder.

When such file is not found, the server then use one of the index files in from the /usr/share/httpd/noindex folder.

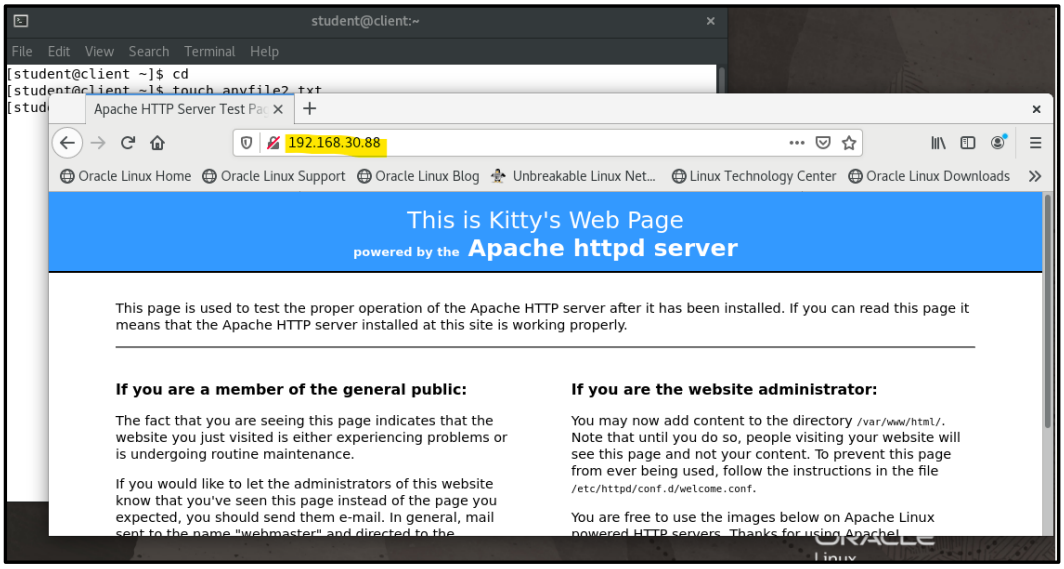
```
[root@server ~]# ls /usr/share/httpd/noindex
index.html
[root@server ~]#
```

The index.html file shown is the initial default page for a newly installed Apache 2 server.

5. Adjust the Firewall to allow remote clients to connect to your Apache Web Server.
   (Consult your tutor if you do not know how to do it.)
   (Hints: Firewall)

On client:

6. Start a web browser and browse to http://<server ip> and check if you can view the server's web page.



2. Basic Apache2 Configuration and Monitoring

<u>On server:</u>

7. View the apache main config file /etc/httpd/conf/httpd.conf. Look at the current settings for :

   `DocumentRoot`
   `DirectoryIndex`

   (Please read the associated comments carefully)

```
#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/var/www/html"

#
```

: : : : : : : : : :

```
#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>
```

Now you should understand why visiting the http://localhost will return the content of /var/www/html/index.html.

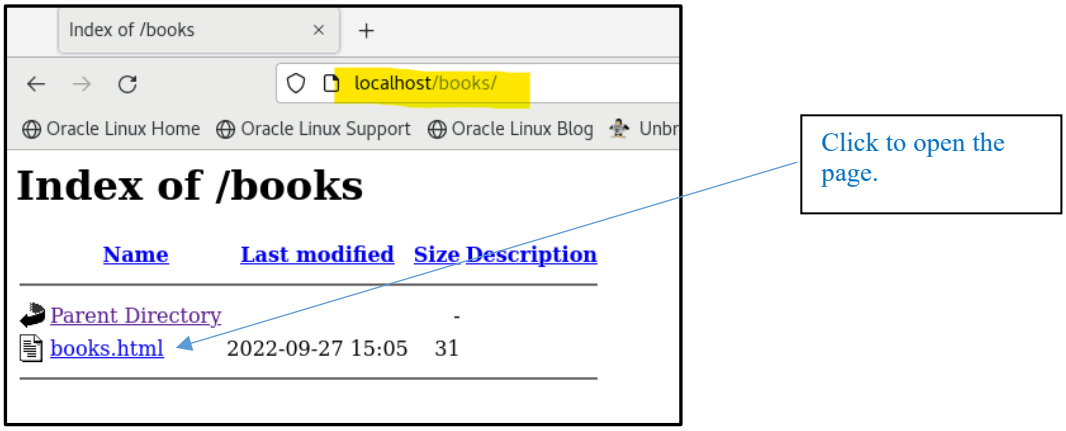
8. Create a new directory in the html directory.

   `mkdir -p /var/www/html/books`

9. Create a new webpage /var/www/html/books/books.html and enter the following data in it.

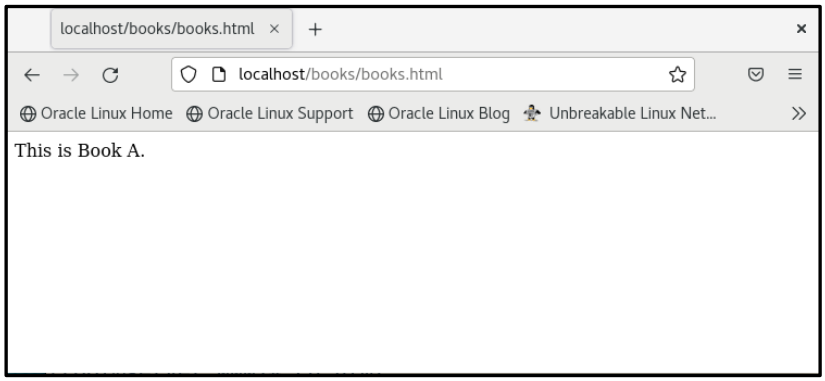
```
<html>
This is Book A.
</html>
```

10. With the web browser, browse to http://localhost/books (or http://<serverIP>/books). Because the books directory does not contain any of the files specified in DirectoryIndex, a listing of the files in the directory is displayed instead.



This directory listing display is not considered as a good practice as it allows users to get a list of the directory content easily. We will show you how to restrict this directory listing in the later part of this exercise.

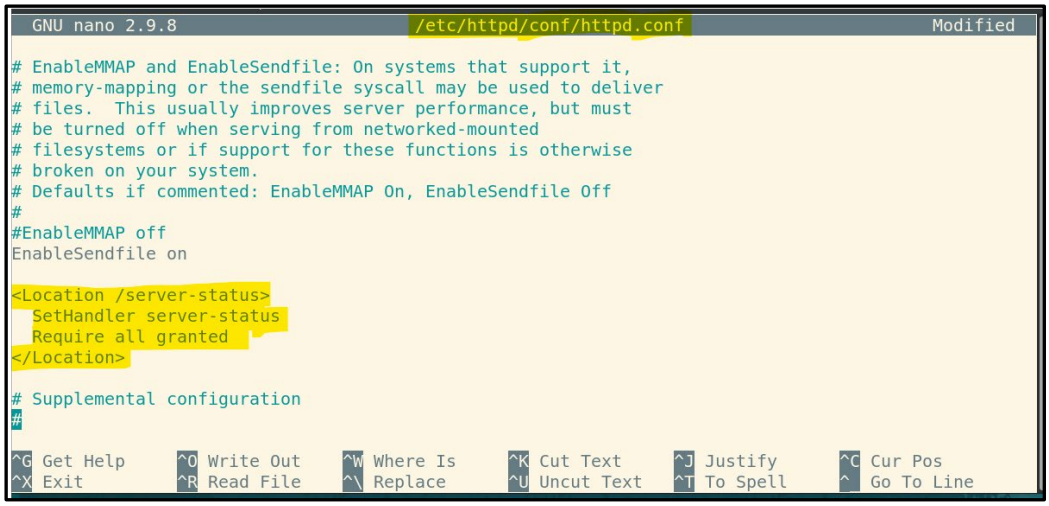
11. To view the books.html, click on the link shown in the directory listing:



12. Monitoring current server status.

Apache2 comes with an optional server-stauts module which can provide a monitor view of the server status. To enable this optional feature, insert the following near to the end of your apache server main config file (/etc/httpd/conf/httpd.conf), right before the 'Supplemental Configuration' section.

```
<Location /server-status>
  SetHandler server-status
  Require all granted
</Location>
```



The above enables the virtual url of http://localhost/server-status and allows everyone from anywhere to access and view the current server status.
You need to restart the server using

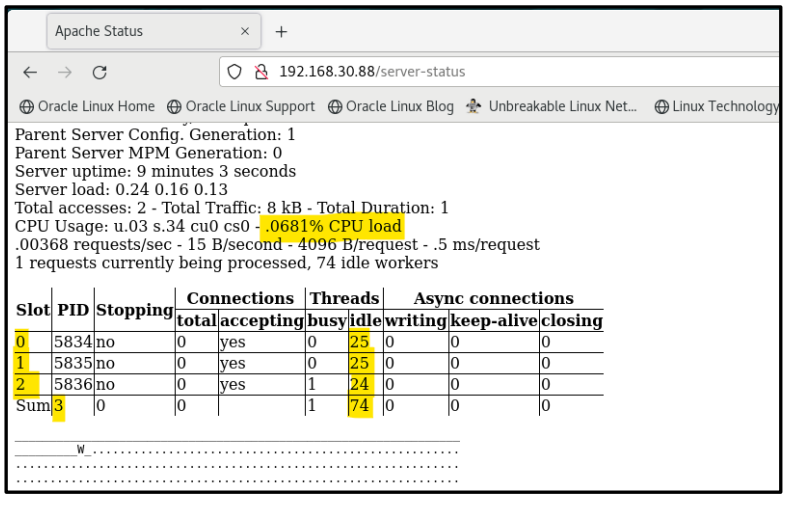
```
systemctl restart httpd
```

to activate the updated configuration.

On Client:

13. Try to access to the server status page via http://<server ip>/server-status.

First access to server-status page and scroll down a bit to view the number of started processes , and the current workers (pre-loaded running threads)



Pre-loaded worker running threads are determining by number of server process X threads per server.

In the above, there are 3 server processes, and each has 25 workers running. Total of 75 workers, and only one of them is currently 'busy'. The 'busy' one should be the one that responsible for delivering the page you are viewing.

On Server:

14. Scaling up/down the server performance by changing the number of pre-loaded worker instances.

    Edit the main config file /etc/httpd/conf/httpd.conf. After the DocumentRoot line, add a few lines to specify the number for StartServers , ThreadsPerChild , ServerLimit and MaxRequestWorkers.

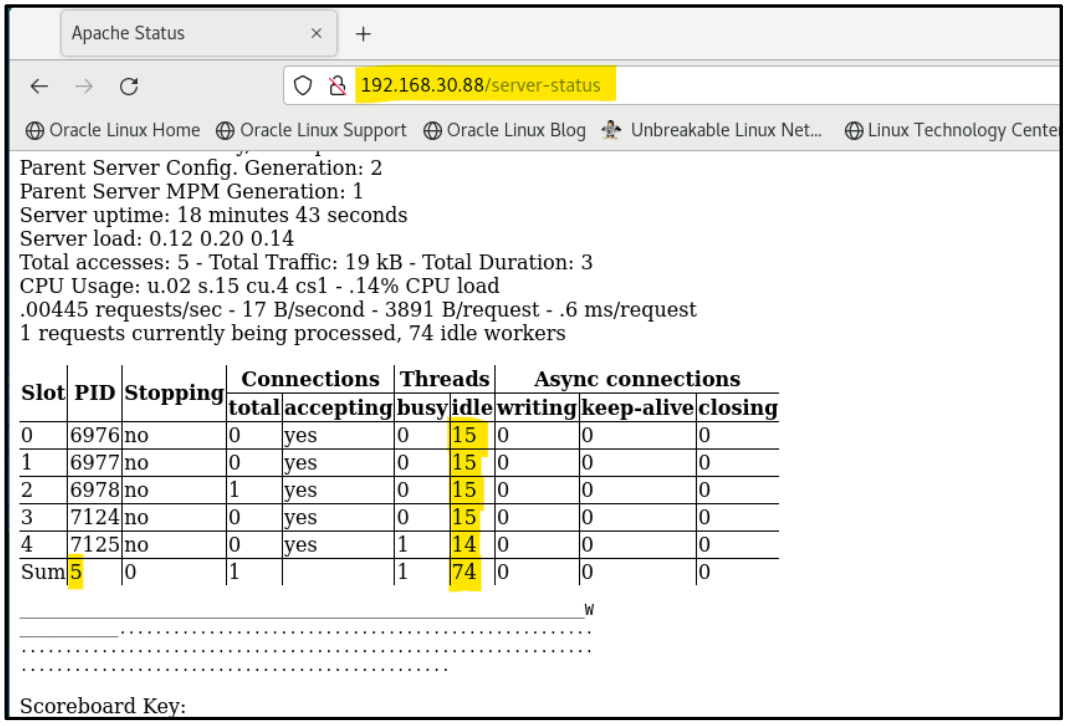
```
DocumentRoot "/var/www/html"
StartServers          5
ThreadsPerChild      15
ServerLimit          10
MaxRequestWorkers   150
```

Reload the httpd service. (You can also restart the service).

```
systemctl  reload  httpd
```

On Client:

Access to server-status page and view the number of current workers again. Do you see the difference?



Verify the number of processes and Threads reported are matching with your configuration.

On Server:

15. Try an alternate way to check for the number of the preloaded worker threads. Use the ps command (with the aux or -ef option) to check how many httpd servers owned by user apache have been started up.

```
[root@server ~]# ps -ef | grep httpd
root       13601       1  0 19:19 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     14156   13601  0 19:36 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     14157   13601  0 19:36 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     14158   13601  0 19:36 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     14159   13601  0 19:36 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     14302   13601  0 19:36 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
apache     14303   13601  0 19:36 ?        00:00:00 /usr/sbin/httpd -DFOREGROUND
root       14450    4409  0 19:39 pts/0    00:00:00 grep --color=auto httpd
[root@server ~]#
```

Take note that the first two listed httpd processes are the parent processes. They are not the part of the worker processes. Process 14157 to 14303 are the worker processes. There are total 5 of them. However, this list is not able to tell the actual number of workers running threads.

Optional Challenges:

Do the necessary configuration to control the access to the server-status page: Only the browser runs in the server can access to the page. The browser runs in the client will not be allowed to access to the page. Cannot use firewall configuration as the solution. (Hints: <Location /server-status> container.)

16. More on server monitoring methods.

    View the Apache log files. These files provide the primary sources for intrusion detections.

```
tail /var/log/httpd/access_log
tail /var/log/httpd/error_log
```

Optional Challenges:

Based on the content of /var/log/httpd/error_log, you may see two* annoying messages:

- [lbmethod_heartbeat:notice] …… AH02282: No slotmem from mod_heartmonitor.
- [mpm_event:warn] ……. AH00514: MaxRequestWorkers of 400 is not an integer ……

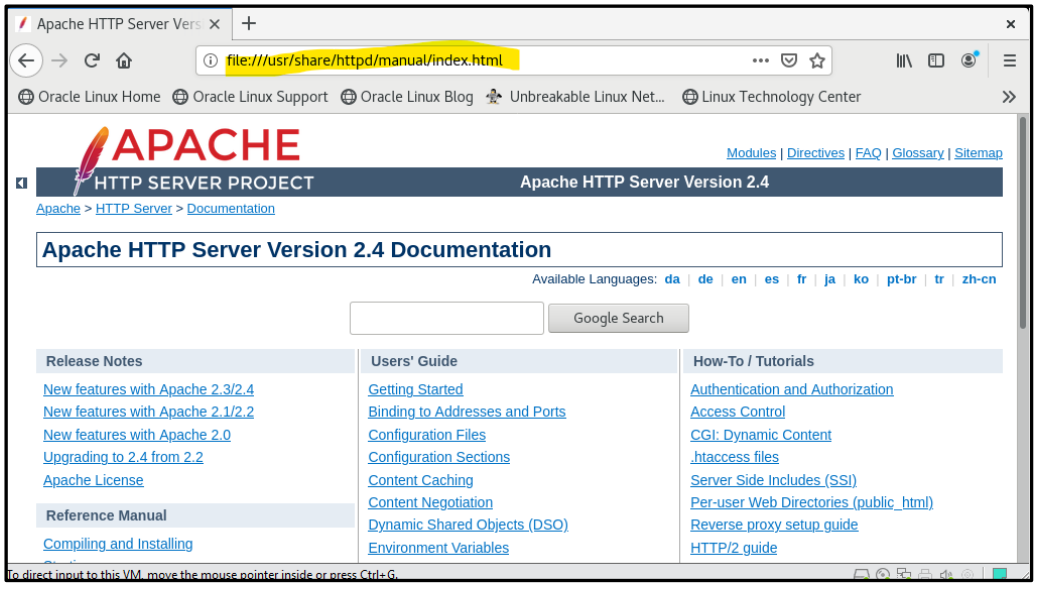
* The first one is a notice message, the second one is a warning message.

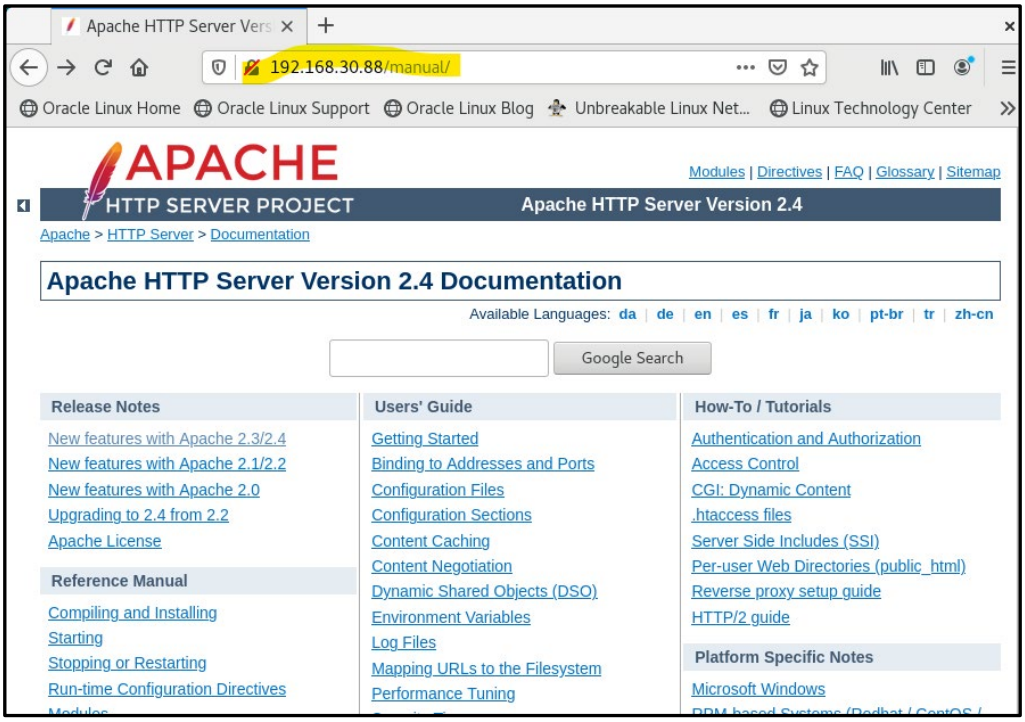
Try to resolve / suppress these two annoying messages (please spend no more than 15 minutes on this).

17. Install the httpd-manual package

```
dnf -y install httpd-manual
```

18. Using a web browser, **browse to** file:///usr/share/httpd/manual/index.html [at server]or **browse to** http://<Server IP>/manual [at client]. (Remember to run "systemctl reload httpd" or "systemctl restart httpd").





3. Containers: <directory> statement, Indexes

On server:

Currently, when you browse to http://localhost/books, a directory listing is displayed. It is not recommended to display such directory listings to web users. You will now configure your web server to disable directory listings by turning off the Indexes option.

1. Edit /etc/httpd/conf/httpd.conf. Append the following container to the end of the file.

```
<Directory /var/www/html/books>
    Options -Indexes
</Directory>
```

Or

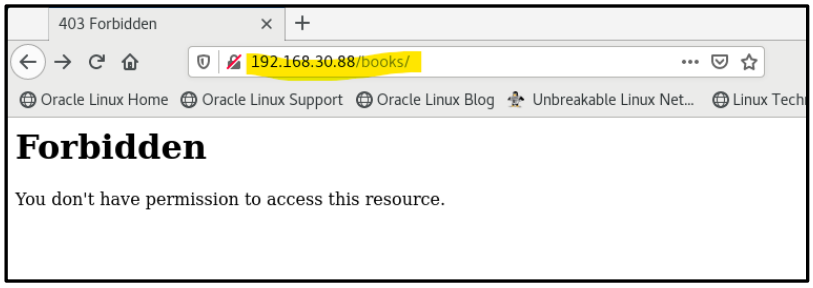
Create a new file , /etc/httpd/conf.d/books.conf. Use the following lines into this new conf file.

```
<Directory /var/www/html/books>
    Options -Indexes
</Directory>
```

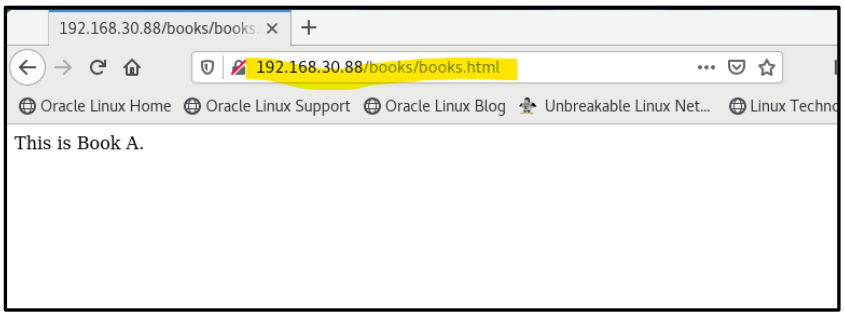
> Note:
> Using a separate configuration file to manage individual directory configuration is recommended as this approach is easier for future updates. All the *.conf files defined in /etc/httpd/conf.d folder will be process at the server's starting time.

2. Reload the httpd service.

3. Browse to http://localhost/books (on server) or http://<server ip>/books (on client) again. You will see a "Forbidden" message.



4. Verify you can still access to the books.html page by using the full url : http://localhost/books/books.html



4. Basic Access Control

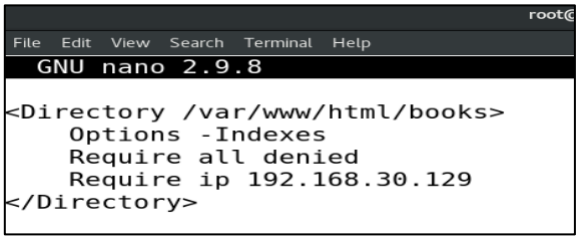
The above -Indexes configuration only prevents the directory listing operation, but it does not define any access control settings. Here we will try to define some access control measures at the directory level.

<u>On server:</u>

1. Edit /etc/httpd/conf/httpd.conf (or your /etc/httpd/conf.d/books.conf). Add the following lines in bold to the <Directory /var/www/html/books> section that you have defined earlier.

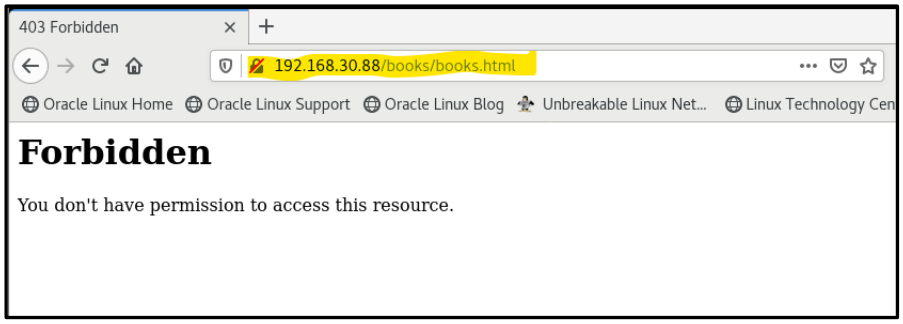
```
<Directory /var/www/html/books>
    Options -Indexes
    Require all denied
    Require ip your_client_ip
</Directory>
```

for example, if your client IP is 192.168.30.129:



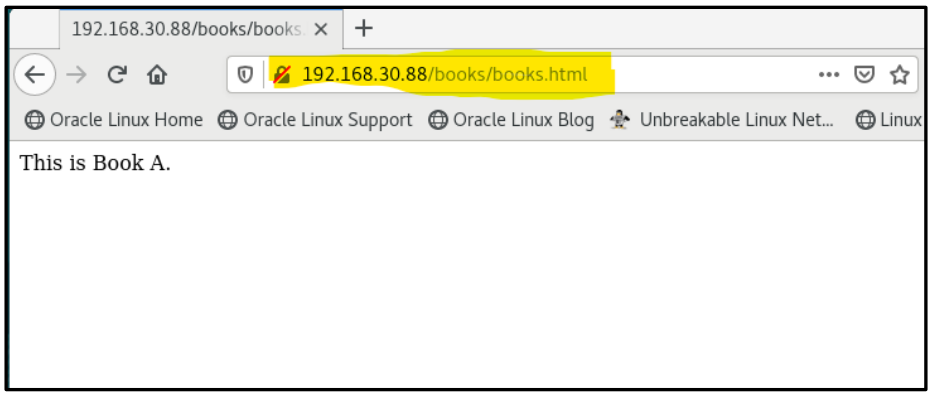
2. Reload the httpd service.
3. Browse to http://localhost/books/books.html. You will see a Forbidden error message.
4. Browse to http://<server ip>/books/books.html. You will again see a Forbidden error

message.



On client:

5. Browse to http://<*server_ip*>/books/books.html. You should be successful.



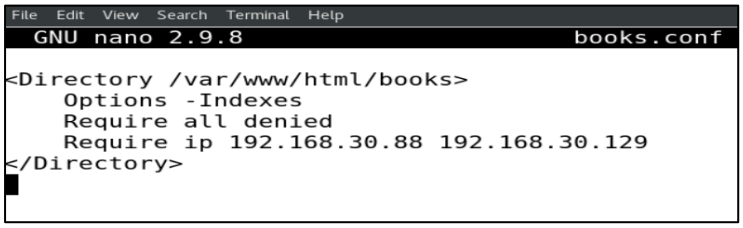
(Note: ensure the current ip address of your client matches the Require ip configuration.)

On server:

6. Edit /etc/httpd/conf/httpd.conf. Update the Require ip line to the <Directory> section that you added earlier.

```
<Directory /var/www/html/books>
    Options -Indexes
    Require all denied
    Require ip your_server_ip your_client_ip
</Directory>
```

For example:



7. Reload the httpd service.

Browse to

http://<your_server_ip>/books/books.html

And

http://localhost/books/books.html

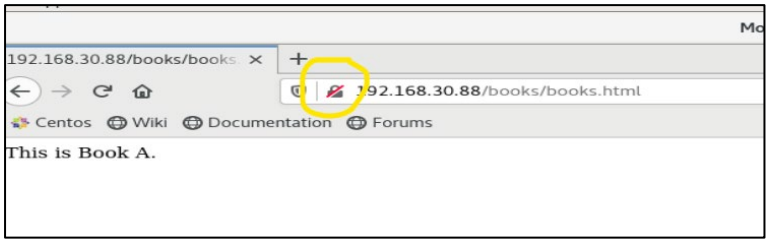
Do you find something very strange?

> Note: In order to allow the server to access to its own web content with http://localhost, you may add a 'Require local' configuration line. For example:
>
> ```
> Require ip      192.168.30.88  192.168.30.129
> Require local
> ```

Recommend Ref: https://httpd.apache.org/docs/current/howto/access.html
To learn how to define ranges of ip address in the Require option.

5. Securing Apache using SSL

In our previous exercises, we will see the 'connection is not secured' warning icon at the browser:

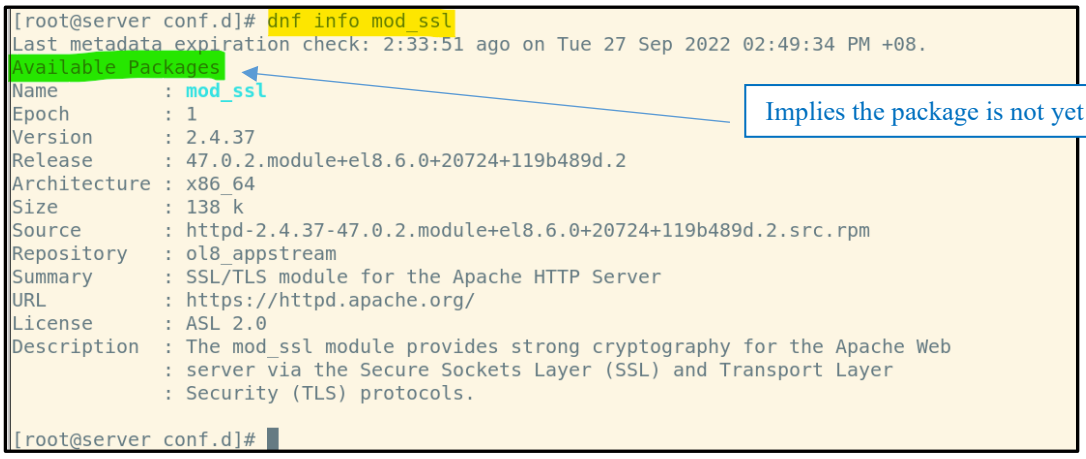


It is because we were using http protocol which is not secured.
Let's tighten the security by offering https protocol from our web server.

On server:

1. Verify if we have already installed the require package, mod_ssl. Type

```
dnf info mod_ssl
```



2. Install the SSL module for Apache (if needed.)

```
dnf install mod_ssl -y
```

3. The SSL module for Apache will generate a default ssl.conf file in the /etc/httpd/conf.d folder. In this ssl.conf configuration file, it will refer to the following two folders for the required Private key and Certificate for the https deployment.

   For Private key: /etc/pki/tls/private/
   For Certificate: /etc/pki/tls/certs/

   However, there is no usable private key nor certificate created at this initial state.

4. Checking what is inside these two folders:

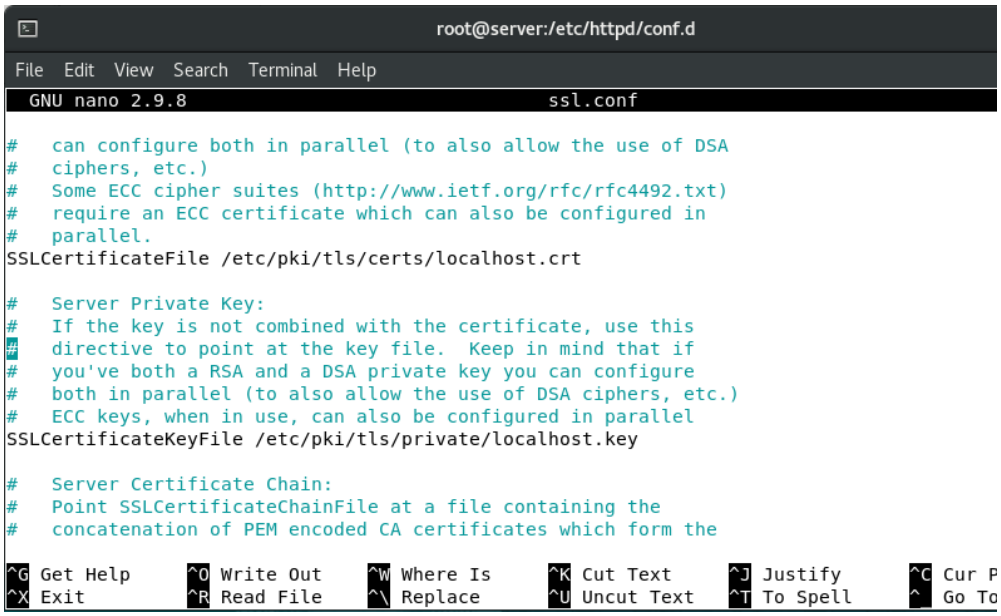
```
[root@server conf.d]# ls /etc/pki/tls/private
[root@server conf.d]#
[root@server conf.d]#
[root@server conf.d]# ls /etc/pki/tls/certs
ca-bundle.crt  ca-bundle.trust.crt
[root@server conf.d]#
[root@server conf.d]#
[root@server conf.d]#
```

   The private directory is empty.

   The certs directory stores two standard crt files. These files contain the default Trusted Root Certificate Authorities Certificates. We cannot use any of them to host our own https web site.

5. Generation of your own key pair (consists of a private key and a public key) and place the public key in a certificate to meet the https hosting requirement.

   First examine the content of the /etc/httpd/conf.d/ssl.conf file.



   You need to generate the required localhost.key (The Private key) and localhost.crt (Self-signed ssl certificate which contains the public key) files using the openssl utility:

   As root, type in the following in one command line:

```
openssl req -newkey rsa:2048 -nodes -keyout \
```

```
/etc/pki/tls/private/localhost.key -x509 -days 365 -out \
/etc/pki/tls/certs/localhost.crt
```

(Note: The above should be a single command line, we can use the back slash escape character '\' to enter the long single line command in multiple lines.)

You will also be prompted to provide some required information to define the SSL certificate. Below is screen shot running the openssl command. You can refer to the highlight input to reply to the prompted questions.

```
[root@server conf.d]# openssl req -newkey rsa:2048 -nodes -keyout \
> /etc/pki/tls/private/localhost.key -x509 -days 365 -out \
> /etc/pki/tls/certs/localhost.crt
Generating a RSA private key
..............................................+++++
.....................+++++
writing new private key to '/etc/pki/tls/private/localhost.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) [XX]:SG
State or Province Name (full name) []:Singapore
Locality Name (eg, city) [Default City]:Dover
Organization Name (eg, company) [Default Company Ltd]:DISM
Organizational Unit Name (eg, section) []:LAS
Common Name (eg, your name or your server's hostname) []:server.example.com
Email Address []:las@example.com
[root@server conf.d]#
```

Just ensure that you have entered "server.example.com" when it is prompted for the Common Name. (The second last one.)

6. You may now examine the content of the two files generated by the above command.

```
cat /etc/pki/tls/certs/localhost.crt
```

```
[root@server conf.d]# cat /etc/pki/tls/certs/localhost.crt
-----BEGIN CERTIFICATE-----
MIID+TCCAuGgAwIBAgIUL7nJQPs2fyk/r7HSI0kejE6IO/UwDQYJKoZIhvcNAQEL
BQAwgYsxCzAJBgNVBAYTAlNHMRIwEAYDVQQIDAlTaW5nYXBvcmUxDjAMBgNVBAcM
BURvdmVyMQ0wCwYDVQQKDARESVNNMQwwCgYDVQQLDANMQVMxGzAZBgNVBAMMEnNl
cnZlci5leGFtcGxlLmNvbTEeMBwGCSqGSIb3DQEJARYPbGFzQGV4YW1wbGUuY29t
MB4XDTIyMDkyNzA5MzMzN1oXDTIzMDkyNzA5MzMzN1owgYsxCzAJBgNVBAYTAlNH
MRIwEAYDVQQIDAlTaW5nYXBvcmUxDjAMBgNVBAcMBURvdmVyMQ0wCwYDVQQKDARE
SVNNMQwwCgYDVQQLDANMQVMxGzAZBgNVBAMMEnNlcnZlci5leGFtcGxlLmNvbTEe
MBwGCSqGSIb3DQEJARYPbGFzQGV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEAv8PBWqZCagGTGhz/dZfZeyc8ME/Hlxi1A7f7adetFoDc
IWh43BxGefg7I0ZIf+Kx0PLaN8PM2r5oc1lFlCSv6cl1j0SbMO22+ZNX3eopxa+e
Gb+aOTPc2PMhyztilGmSfIGlmnLm0cVCZQSYhkNiShD0pJs7Lij56W1iSQNTEHAx
DtpNYBIHWZIp4KYUKipqfcYCjpfNI+Nzen54xc4iYmloTmNawFbw6kVWGmfQg6cQ
kF2Yl805aT89n3YBp3ke4H1pIY3r9LDAB3U5gcg3rxUcHesPAnJYYayGConoRF4n
s1++y9jac2LtMRjO7vsQrhSV4h8rIgl+Vh96FsDu+wIDAQABo1MwUTAdBgNVHQ4E
FgQUbDIbb3R00Jj2i71+YrBz/n1O+7YwHwYDVR0jBBgwFoAUbDIbb3R00Jj2i71+
YrBz/n1O+7YwDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAQEAFi6W
URpZfz4ksneb9JaFUNsVL1ZykZdgjYFCwjaED/+XJEnLbCWtyQk4OdrxkCj9Sf30
NCg9DII18cVHnMPbSrS+n5Fd6IGsXEiop/poirbs8uHX6n1d+jS7DvN/cY+tOUGr
jVuXWRRqljr1NZ3XYEqQDCJeqDpd+keQRJL+bS6opXvXAiYMwMoxOJOLR3clmUa7
P/SCrC1EGHl2VIRdrUR+lcQktonflBhWOL5bakcTV7a+Eze3HsPMSeDyXGsBVacM
AskB/A4yvE7aULX/odd/8TjFK8mf9gGhy1AqSEr98uuOl+3UQ9ZVoj3ow+42bI7A
3VCkCMHEIUxoVlXAkg==
-----END CERTIFICATE-----
[root@server conf.d]#
```

(The above is the certificate which contains the public key (in PEM format), no need to keep secret.)

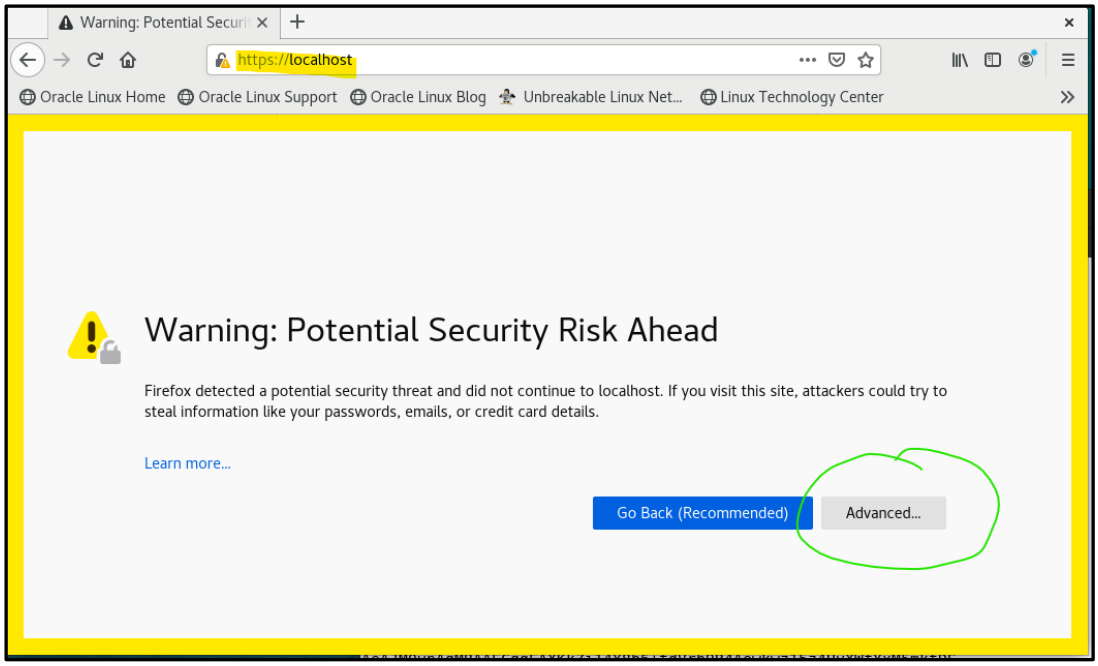
```
cat /etc/pki/tls/private/localhost.key
```

(This file contains the private key (in PEM format) which is part of the key pair.)

For security reasons, you should never show your private key to the others.

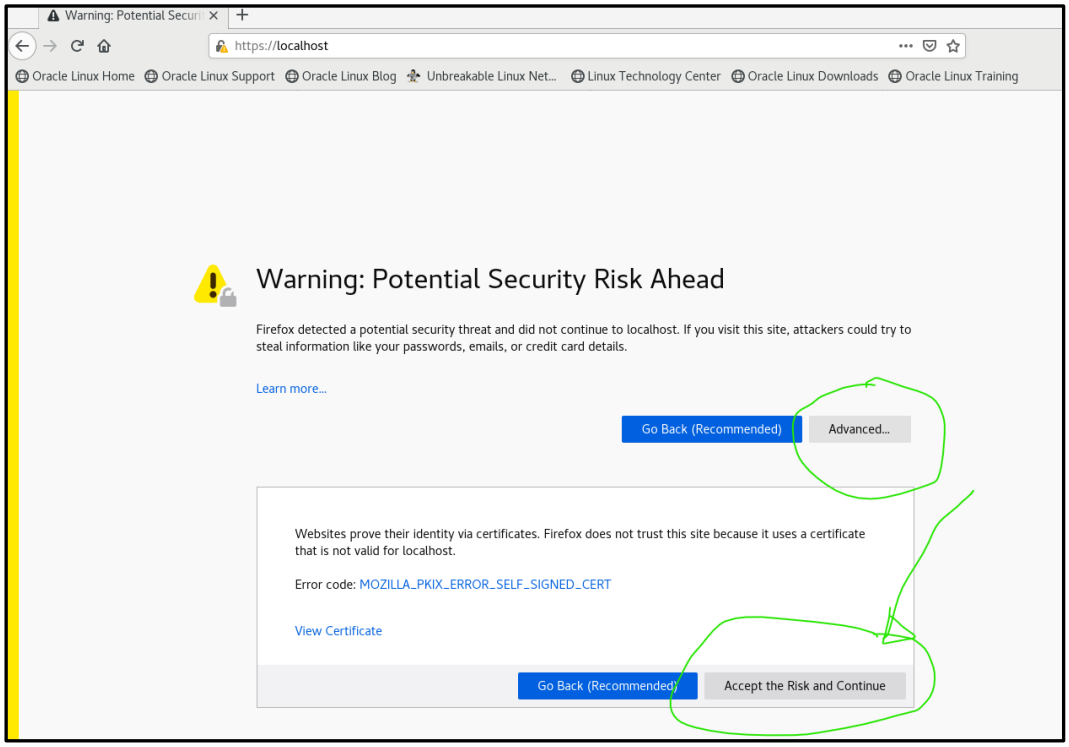
Reference - [What is PEM format ?]

7. Now we are all set. Reload/Restart your httpd. Browse to 'https://localhost'. You will see a warning as the certificate is only a test certificate and is not signed by an Authorised Certificate Authority.

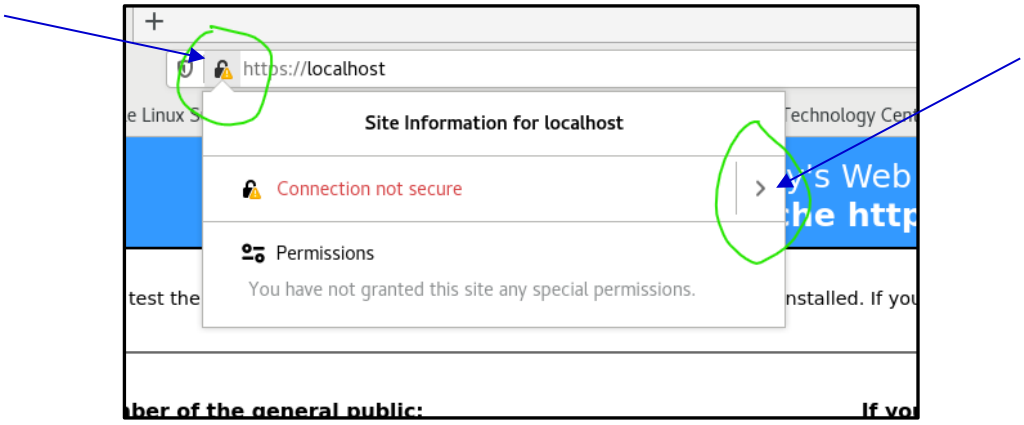


8. The certificate we are using is self-signed, and not signed by an Authorised Certificate Authority, thus, the browser issues the warning.

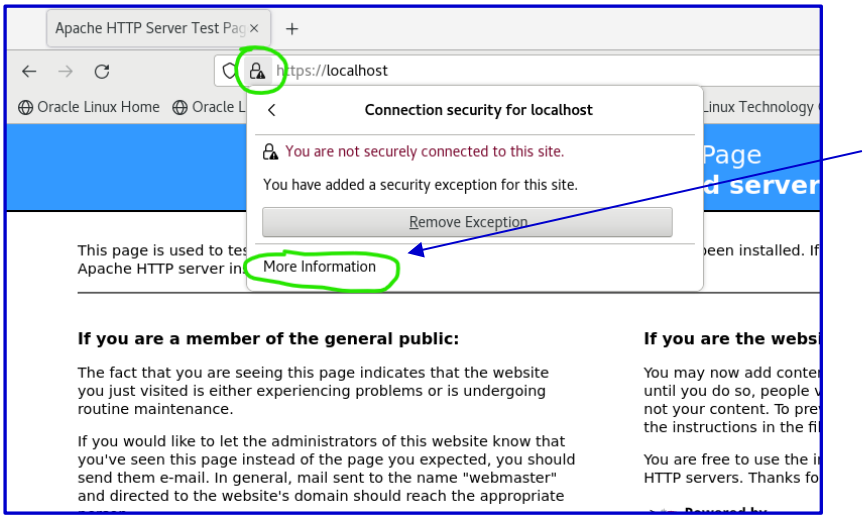
   To accept the risk (if you are sure) and proceed, you may click on the advanced button to reveal more option. Click on the 'Accept the risk and continue' button, then you can load in the default localhost index page.

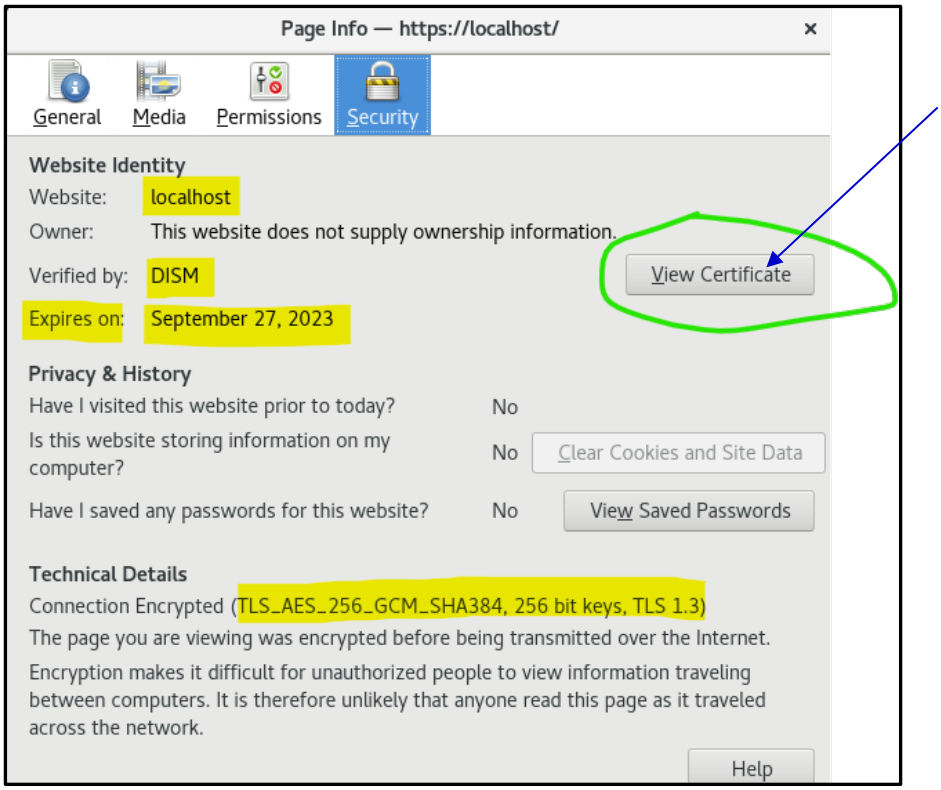


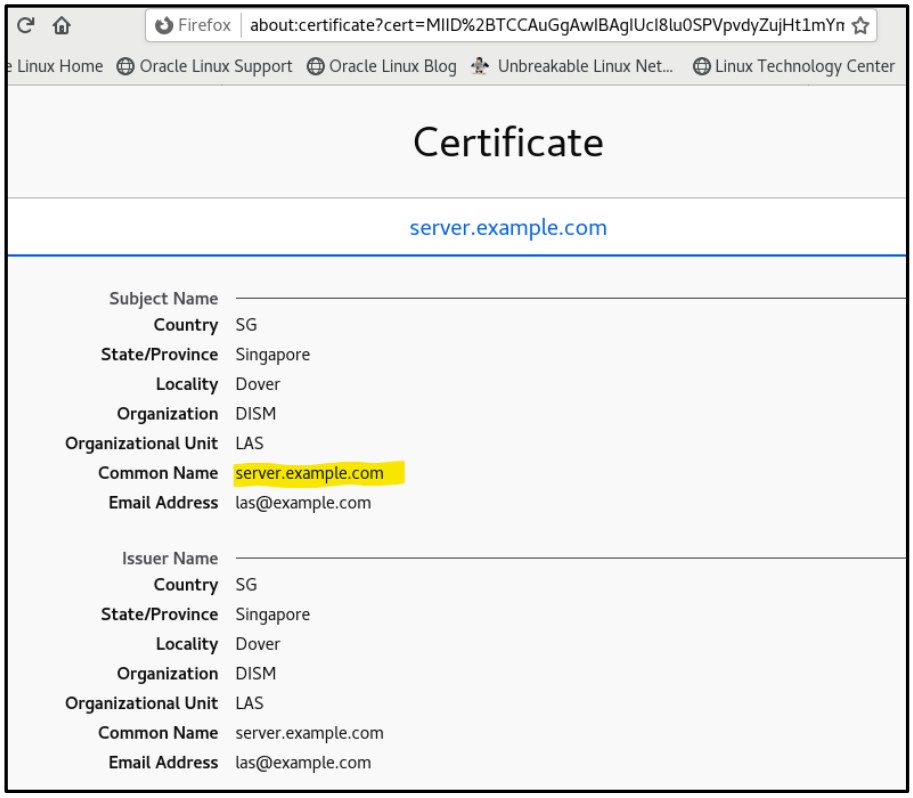
9. While in the Firefox web browser, you can view the details of the encryption being used by the web server for this https connection. Click on the padlock in the address bar and select More Information (see following diagram)



10. Go for view more information and select Security Tab and click on the View Certificate button to get some more details of this certificate.







Note: This self-signed cert is issued to 'server.example.com', thus, your firefox will give you warning when you are accessing the site via the host name, 'localhost'. Do you recall you have set your server to have the name 'server.example.com' in the previous step? you may try to access https://server.example.com, it will still have the warning, as the certificate is issued by server.example.com (and your browser is not configured to trust certificates issued by it. Adding exception only allow you to browse the page but the warning remains.

6. Set up Name Based Virtual Host

<u>On server:</u>

1. To get multiple hostnames to resolve to your IP, add the following 3 lines to /etc/hosts

   your_server_ip server.example.com
   your_server_ip www.flowers.com flowers
   your_server_ip www.fruits.com fruits

   For example:

```
 GNU nano 2.9.8                    /etc/hosts                         Modified

127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.30.88 server.example.com
192.168.30.88 www.flowers.com   flowers
192.168.30.88 www.fruits.com    fruits
```

   (Note: The /etc/hosts file allows the system to keep static Fully Qualified Domain Name (FQDN) and host Names. It provides a simple way to simulate of hosting a domain in a localhost

environment.) `

2. Test your settings by pinging the 3 FQDNs:

```
[root@server ~]# ping -c 1 -n server.example.com
PING server.example.com (192.168.30.88) 56(84) bytes of data.
64 bytes from 192.168.30.88: icmp_seq=1 ttl=64 time=0.057 ms

--- server.example.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.057/0.057/0.057/0.000 ms
[root@server ~]# ping -c 1 -n www.flowers.com
PING www.flowers.com (192.168.30.88) 56(84) bytes of data.
64 bytes from 192.168.30.88: icmp_seq=1 ttl=64 time=0.048 ms

--- www.flowers.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.048/0.048/0.048/0.000 ms
[root@server ~]# ping -c 1 -n www.fruits.com
PING www.fruits.com (192.168.30.88) 56(84) bytes of data.
64 bytes from 192.168.30.88: icmp_seq=1 ttl=64 time=0.060 ms

--- www.fruits.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.060/0.060/0.060/0.000 ms
[root@server ~]#
```

(As shown at the above, all the 3 different FQDNs are all resolved with the same IP address. You may try to ping the 2 host names too: flowers and fruits.)

3. As root, create a directory /var/www/flowers.

4. Create a file “index.html” in the flowers directory with the following minimum content:

```
<html>
<H1>Flowers</H1>
<a href='http://www.fruits.com'>Go to Fruits</a>
</html>
```

5. Create a directory /var/www/fruits.

6. Create a file “index.html” in the fruits directory with the following minimum content :

```
<html>
<H1>Fruits</H1>
<a href='http://www.flowers.com'>Go to Flowers</a>
</html>
```

7. Go to the directory /etc/httpd/conf.d.
8. Create a new file flowers.conf and enter the following (or you can also append these lines to the end of /etc/httpd/conf/httpd.conf) :

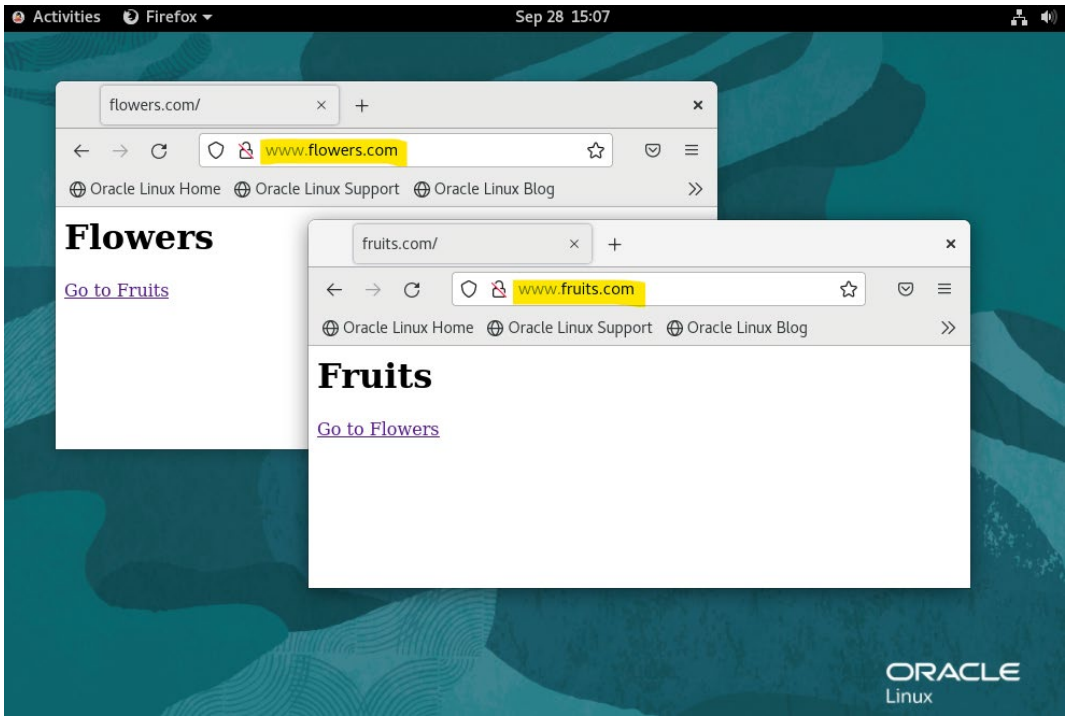
```
<VirtualHost your_server_ip:80>
```

```
    ServerName www.flowers.com
    DocumentRoot /var/www/flowers
    ErrorLog /var/log/httpd/flowers-error_log
    CustomLog /var/log/httpd/flowers-access_log
combined
</VirtualHost>
```

9. Create a new file fruits.conf and enter the following (or you can also append these lines to the end of /etc/httpd/conf/httpd.conf) :

```
<VirtualHost your_server_ip:80>
    ServerName www.fruits.com
    DocumentRoot /var/www/fruits
    ErrorLog /var/log/httpd/fruits-error_log
    CustomLog /var/log/httpd/fruits-access_log combined
</VirtualHost>
```

10. Restart or reload the Apache server.
11. Browse to http://www.flowers.com and http://www.fruits.com.
    You may need to clear the cache from your fire-fox browser if you only see your original default page.



12. View the relevant log files to verify the logging operations is working fine*.

```
tail /var/log/httpd/flowers-access_log
tail /var/log/httpd/fruits-access_log
tail /var/log/httpd/flowers-error_log
tail /var/log/httpd/fruits-error_log
```

*Simple Trick to validate the error_log configuration:-
Login as root, create two new text files : /var/www/flowers/forAdminOnly.html and /var/www/fruits/forAdminOnly.html with the following content:

```
<html>
<H1>Protected page</H1>
This page can only be viewed by the root.
</html>
```

Ensure these two files can only be viewed by the root user:

```
#chmod 0600 /var/www/flowers/forAdminOnly.html
#chmod 0600 /var/www/fruits/forAdminOnly.html
```

Use Firefox to browse to http://www.flowers.com/forAdminOnly.html will trigger the error log. Same effect applies to http://www.fruits.com/forAdminOnly.html.

7. Configure the access control to enable co-authoring of web content

It is a common practice that a team of web designers will take care the web content of a web site. We will explore one of the possible group-based file accessing scheme to support such common operations.

1. Work on the follow tasks to configure the assess right of the web content of the sites.

Task :

Allow the members of the ppm group (peter, paul and mary) to be able to create, read, modify and delete the web pages belonging to www.fruits.com. Any ppm member can modify pages that were created by another ppm member.

Hint :

a. Change the group owner of /var/www/fruits directory (**and its contents**) to ppm.
(you can user chgrp command with the -R (recursive) option)

b. Change the file permissions of /var/www/fruits directory (**and its contents**) so that the group ppm can read and write.
(you can user chmod command with the -R (recursive) option)

c. Set the SetGID bit on /var/www/fruits directory. (you can use "chmod g+s /var/www/fruits") - no need to use -R option this time.

To verify you have completed all of the above, you may type :

```
ls -l /var/www
```

```
[root@server ~]# chgrp -R ppm /var/www/fruits
[root@server ~]# chmod -R g=rwx /var/www/fruits
[root@server ~]# chmod g+s /var/www/fruits/
[root@server ~]# ls -l /var/www
total 0
drwxr-xr-x. 2 root root  6 Aug  5 20:00 cgi-bin
drwxr-xr-x. 2 root root 49 Sep 28 16:42 flowers
drwxrwsr-x. 2 root ppm  24 Sep 28 16:58 fruits
drwxr-xr-x. 3 root root 37 Sep 27 15:04 html
[root@server ~]#
```

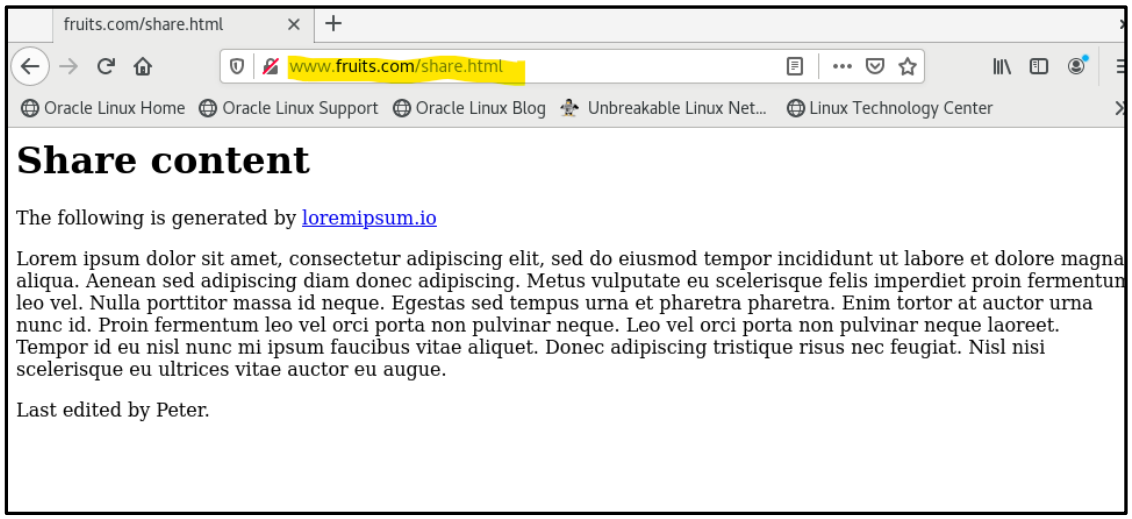
to check.

(Take note that, you may use groups command to check whether peter and mary are in the ppm groups.

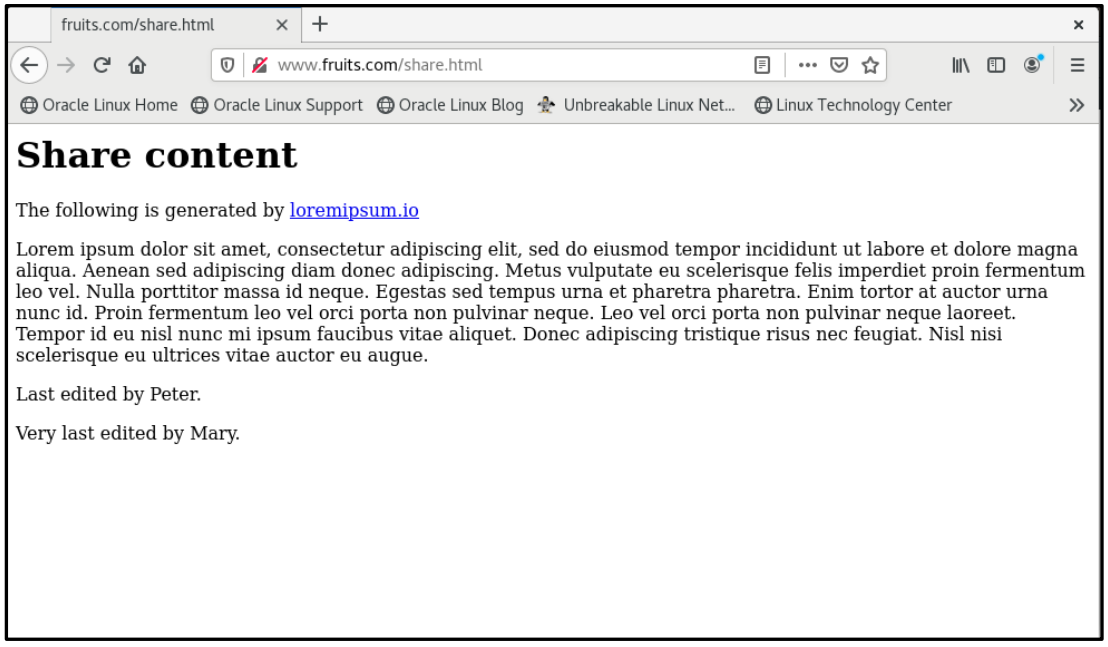
To test, login as peter (or use 'su - peter') and update a web page in /var/www/fruits directory. Login as another ppm member and test that this user can modify the same web page. Demo to your tutor that you have accomplished this task.

For example:

First Peter create a new file, share.html, and put in some content. For example:



Later, login as mary (or su - mary) to edit the same file and appended her name:



Have you figured out why the above can work ?

If not sure. Try this:

Clear the SetGID bit on /var/www/fruits directory. (you can use “chmod g-s /var/www/fruits”)

Then let mary 'create' a new file, mary.html , in the /var/www/fruits directory. Verify that peter cannot update the mary.html.

You may consult your tutor for the explanations.

8. User Authentication (web site access control for browsing)

Apache provide its own user management system which is independent from the Linux user management system. The apache user accounts can be used for apache basic authentication. We can use htpasswd command to create and maintain apache user accounts. The account information is encrypted (or based on hashes) and stored at a specific file assigned by the apache administrator. There are different encryption methods available:
bcrypt - considered to be the most secure.
md5 - the default mode. (since ver 2.2.18)
crypt - was the default before ver 2.2.18 . It is considered insecure nowadays.
sha - even older than crypt. Insecure by today's standard.

(ref: man htpasswd(1))

<u>On server:</u>

1. Use the htpasswd command to create 2 Apache users and store their info in /etc/httpd/conf/flowers-users. Set their passwords to be identical to their user id. (make it easier to remember.)

```
htpasswd -c    -B  /etc/httpd/conf/flowers-users bob
```

The –c option is used to create the file.
The -B option is used to select bcrypt hashing.

```
htpasswd   -m /etc/httpd/conf/flowers-users alice
```

The -m is to select MD5 hashing. Omit the -m will have the same effect. The -c option is not used because you are now adding the second user to the flowers-users file.

You may examine the content of the flowers-users file.

```
[root@server ~]# htpasswd -c    -B  /etc/httpd/conf/flowers-users bob
New password:
Re-type new password:
Adding password for user bob
[root@server ~]# htpasswd -m  /etc/httpd/conf/flowers-users alice
New password:
Re-type new password:
Adding password for user alice
[root@server ~]# cat /etc/httpd/conf/flowers-users
bob:$2y$05$9ErVAgIhjQ7B/cEAJI7Lj.jpg3LYS.R9F1ZGnd82nds.IZ49j2rqK
alice:$apr1$MYiorPr6$.QS6E9J/u8XbTtoY8dcFj/
[root@server ~]#
```

Note: bob and alice are not related to any Linux user accounts in the server.

The password for bob is longer than the one for alice, as their hashing schemes are different.

2. Create the file /var/www/flowers/.htaccess and add the following lines

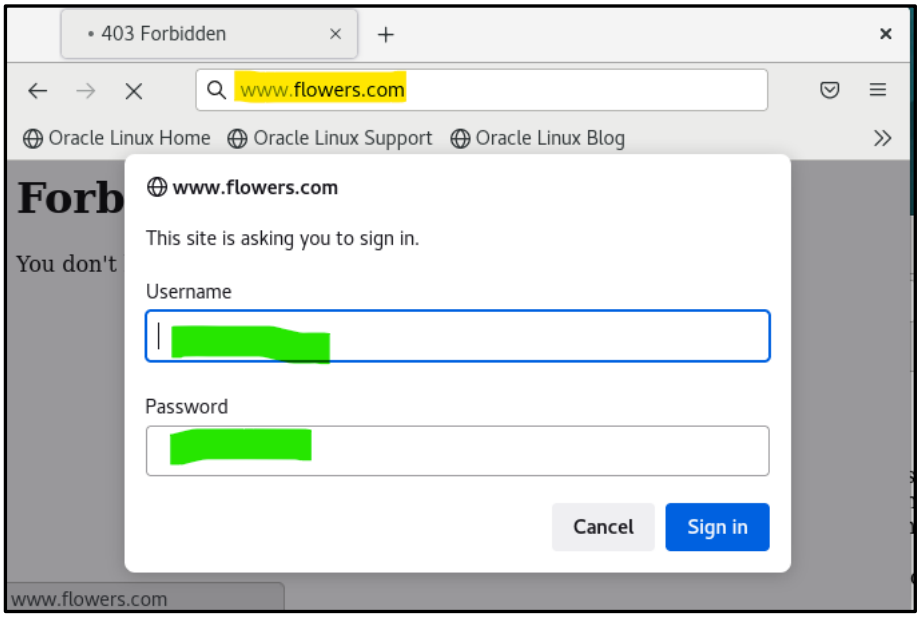
```
AuthType basic
AuthName "Flowers Website"
AuthUserFile /etc/httpd/conf/flowers-users
Require user bob
```

3. Edit the file /etc/httpd/conf.d/flowers.conf and add the following container to the end of the file:

```
<Directory /var/www/flowers>
    AllowOverride AuthConfig
</Directory>
```

```
  GNU nano 2.9.8                    flowers.conf

<VirtualHost 192.168.30.88:80>
        ServerName www.flowers.com
        DocumentRoot /var/www/flowers
        ErrorLog /var/log/httpd/flowers-error_log
        CustomLog /var/log/httpd/flowers-access_log combined
</VirtualHost>
<Directory /var/www/flowers>
    AllowOverride AuthConfig
</Directory>
```

4. Restart or reload Apache server and browse www.flowers.com. Try logging in as both Apache users alice and bob. Alice should not be able to access www.flowers.com, but bob is able to access http://www.flowers.com. (you may use new private window in firefox to switch between the web users)



5. Edit the file /var/www/flowers/.htaccess and add the user "alice" to the Require user list.

```
AuthType basic
AuthName "Flowers Website"
AuthUserFile /etc/httpd/conf/flowers-users
Require user bob alice
```

6. You do not need to restart the Apache Server for the changes to take effect because you made the change in the .htaccess file. Close the Web Browser or clear the browser cache and visit www.flowers.com again. You can login as user alice now.

7. View the log files for www.flowers.com to see what sort of information is captured (/var/log/httpd/flowers-access_log and /var/log/httpd/flowers-error_log)

```
[root@server conf.d]# tail /var/log/httpd/flowers-error_log
[Thu Sep 23 17:27:23.850307 2021] [auth_basic:error] [pid 3278:tid 140072474691328] [client
 192.168.30.88:40116] AH01617: user bob: authentication failure for "/": Password Mismatch
[Thu Sep 23 17:28:50.416362 2021] [authz_core:error] [pid 3280:tid 140072372401920] [client
 192.168.30.88:40126] AH01631: user alice: authorization failure for "/":
[root@server conf.d]#
```

The above are some entries found in the flowers-error.log.

One is the auth_basic:error caused by a password Mismatch error. The other is simply authz-core:error.

```
[root@server conf.d]# tail /var/log/httpd/flowers-access_log
192.168.30.88 - - [23/Sep/2021:17:25:14 +0800] "GET / HTTP/1.1" 401 381 "-" "Mozilla/5.0 (X
11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
192.168.30.88 - bob [23/Sep/2021:17:27:23 +0800] "GET / HTTP/1.1" 401 381 "-" "Mozilla/5.0
(X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
192.168.30.88 - bob [23/Sep/2021:17:27:27 +0800] "GET / HTTP/1.1" 304 - "-" "Mozilla/5.0 (X
11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
192.168.30.88 - bob [23/Sep/2021:17:27:27 +0800] "GET /favicon.ico HTTP/1.1" 404 196 "-" "M
ozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
192.168.30.88 - - [23/Sep/2021:17:28:44 +0800] "GET / HTTP/1.1" 401 381 "-" "Mozilla/5.0 (X
11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
192.168.30.88 - alice [23/Sep/2021:17:28:50 +0800] "GET / HTTP/1.1" 401 381 "-" "Mozilla/5.
0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
192.168.30.88 - - [23/Sep/2021:17:28:53 +0800] "GET /favicon.ico HTTP/1.1" 401 381 "-" "Moz
illa/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
192.168.30.88 - - [23/Sep/2021:17:30:00 +0800] "GET / HTTP/1.1" 401 381 "-" "Mozilla/5.0 (X
11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
192.168.30.88 - alice [23/Sep/2021:17:30:05 +0800] "GET / HTTP/1.1" 200 81 "-" "Mozilla/5.0
 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
192.168.30.88 - alice [23/Sep/2021:17:30:05 +0800] "GET /favicon.ico HTTP/1.1" 404 196 "-"
"Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0"
[root@server conf.d]#
```

The above are some entries found in the flowers-access_log for cross reference check. We see bob can access to the www.flowers.com/index.html first then alice can do so at the later time.

9. Reset Virtual Hosts settings

Disable the Virtual Hosts in case they may interfere with other future practical exercises.

On server:

1. Rename the VirtualHost conf file for www.fruits.com from fruits.conf to fruits.conf.disabled

   Or

   Edit the fruits.conf file, to insert a '#' in every line.

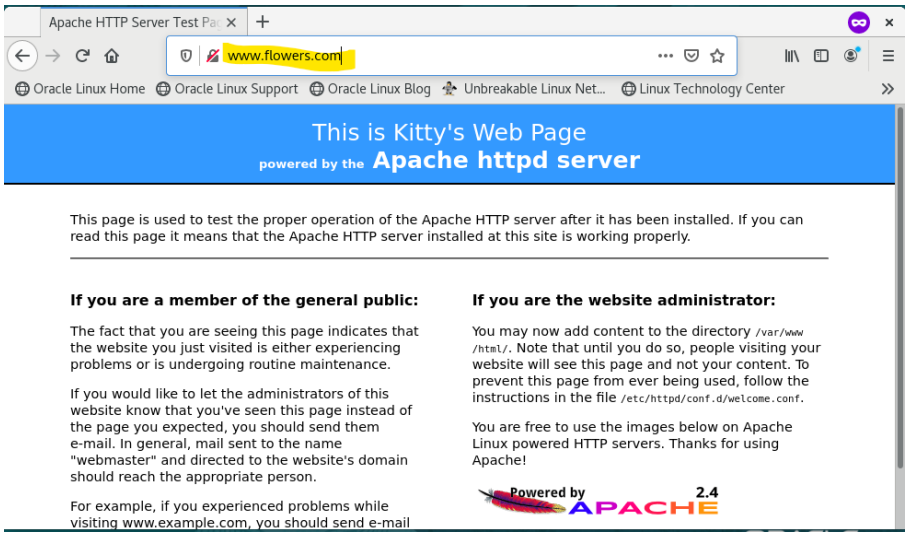
2. Rename the VirtualHost conf file for www.flowers.com from flowers.conf to flowers.conf.disabled

   Or

   Edit the flowers.conf file, to insert a '#' in every line.

```
[root@server ~]# cd /etc/httpd
[root@server httpd]# cd conf.d
[root@server conf.d]# ls
autoindex.conf  flowers.conf  manual.conf  ssl.conf      welcome.conf
books.conf      fruits.conf   README       userdir.conf
[root@server conf.d]# mv flowers.conf flowers.conf.disabled
[root@server conf.d]# mv fruits.conf fruits.conf.disabled
[root@server conf.d]# ls
autoindex.conf  flowers.conf.disabled  manual.conf  ssl.conf      welcome.conf
books.conf      fruits.conf.disabled   README       userdir.conf
[root@server conf.d]#
```

3. Restart or reload the httpd service. Verify your www.fruits.com and www.flowers.com are no longer showing you the corresponding pages. (Note: Need to reload / refresh the page to verify.)

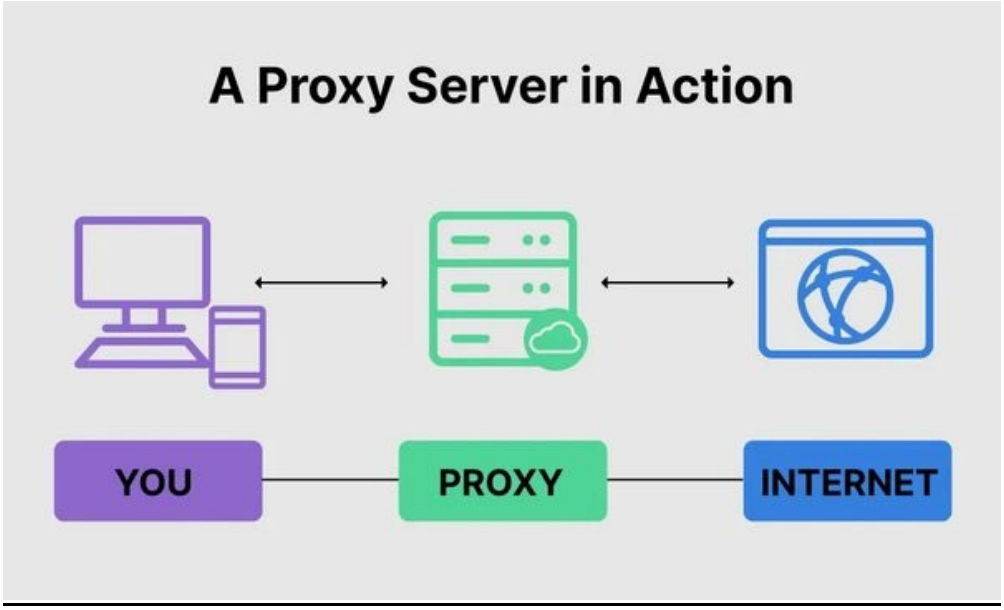


Instead, it will show you the default page of the web server. But why ? Can you give an explanation to your tutor ?

10. Squid Proxy Server

Proxy servers can be used to **filter** and **log** web surfing traffic for security and/or access control purposes.
Squid is a proxy cache server software available on the Linux platform.
A proxy server can be setup in an office network and let all other client machines in the same network to use it for web surfing connections. In this setup, all the incoming and outgoing web traffic can be filtered and/or logged at one single point.
In this exercise, we will setup our server VM to be a proxy server, and the client VM can use it to surf the web.

(recommended ref: http://www.squid-cache.org/)



(~source: https://qph.cf2.quoracdn.net/main-qimg-6efca5a4fb96d835b041adf868345e68-pjlq)

On server:

1. Test that you can browse the Internet (eg google.com or yahoo.com)

   If the server cannot access to the Internet, it will not be able to act as proxy server for other clients !

2. Install the squid package.

   ```
   dnf -y install squid
   ```

3. Edit /etc/squid/squid.conf.

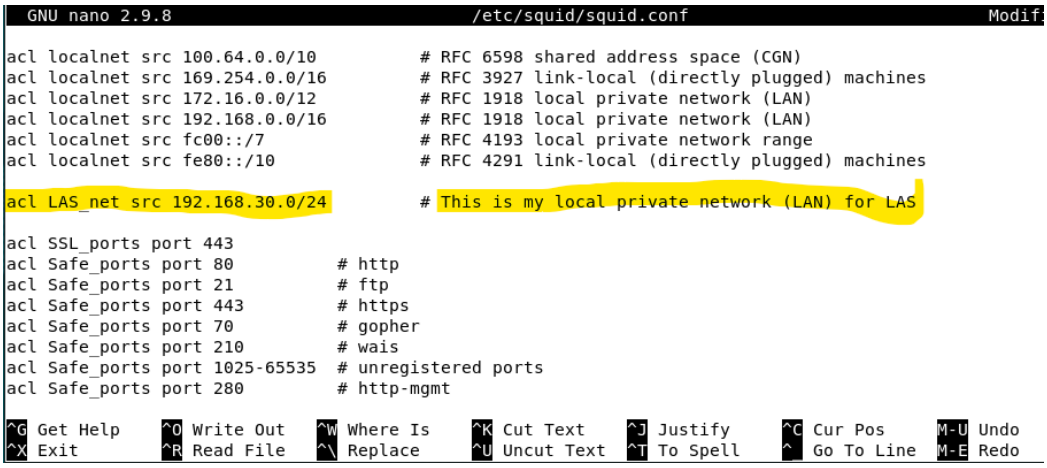
4. Create an Access Control List (acl) for your own subnet. Look for the lines starting with "acl" and add the line in bold.

   acl LAS_net src 192.168.30.0/24

   Replace this with your own subnet. Eg. If your IP is 172.16.10.88 and netmask is 255.255.0.0, then your subnet is 172.16.0.0/16

   You can give any name to your acl.

   For your reference:



5. Create the http_access for LAS_net. Look for the following line and add the line in bold.

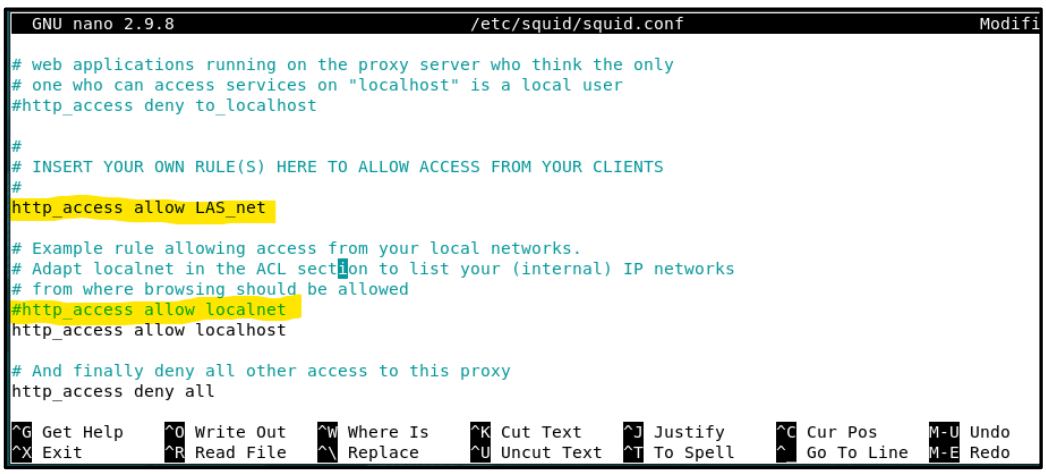
   # INSERT YOUR OWN RULE(S) ...

   http_access allow LAS_net

   This means Squid will allow all http access from the source 192.168.30.0/24

6. Look for the following line and insert an '#' to the line to disallow the default pre-defined 'local_net' to access to the proxy.

   For your reference:



Why do we want to disable the local_net ?

The default settings of local_net covers many private LAN ip addresses ranges.

Ie.

```
acl localnet src 0.0.0.1-0.255.255.255  # RFC 1122 "this" network (LAN)
acl localnet src 10.0.0.0/8             # RFC 1918 local private network (LAN)
acl localnet src 100.64.0.0/10          # RFC 6598 shared address space (CGN)
acl localnet src 169.254.0.0/16         # RFC 3927 link-local (directly plugged) machines
acl localnet src 172.16.0.0/12          # RFC 1918 local private network (LAN)
acl localnet src 192.168.0.0/16         # RFC 1918 local private network (LAN)
acl localnet src fc00::/7               # RFC 4193 local private network range
acl localnet src fe80::/10              # RFC 4291 link-local (directly plugged) machines
```

Disable the local_net is to ensure a tighten control of the access.

7. In the config file, set the parameter visible_hostname to define your hostname. (Add a new line at the end of the file if the parameter visible_hostname is not defined yet)

```
visible_hostname  server.example.com
```

Set this value to your host name.

The value you configured here will be displayed in the error message.

By default, **visible_hostname** will take the auto-detected hostname value. (Sometimes, the service provider may not want to reveal its real hostname.)

Note: you can check for your server's hostname by the

hostnamectl

or

hostname

or

uname -n

commands

8. Save the squid config file and **start the squid service.** (You should know how to enable and start a service by now, right?)
9. By default, Squid runs on tcp Port 3128. Adjust the Firewall on the server to allow remote clients to access your Squid service.

   You may use the following command to open the port 3128 (runtime only).

```
firewall-cmd --add-port=3128/tcp
```

10. Check /var/log/messages if there are any info/error messages related to the squid service.

```
tail /var/log/messages
```

```
[root@server ~]# tail /var/log/messages
Sep 28 17:50:15 server systemd[1]: Started Hostname Service.
Sep 28 17:50:20 server systemd[1]: fprintd.service: Succeeded.
Sep 28 17:50:45 server systemd[1]: systemd-hostnamed.service: Succeeded.
Sep 28 17:53:23 server org.gnome.Shell.desktop[2034]: Window manager warning: last_user_time (114
11316) is greater than comparison timestamp (11411260).  This most likely represents a buggy clie
nt sending inaccurate timestamps in messages such as _NET_ACTIVE_WINDOW.  Trying to work around..
.
Sep 28 17:53:23 server org.gnome.Shell.desktop[2034]: Window manager warning: W3 appears to be on
e of the offending windows with a timestamp of 11411316.  Working around...
Sep 28 17:53:23 server kernel: perf: interrupt took too long (3485 > 3243), lowering kernel.perf_
event_max_sample_rate to 57000
Sep 28 17:55:29 server systemd[1]: Starting Squid caching proxy...
Sep 28 17:55:29 server squid[8696]: Squid Parent: will start 1 kids
Sep 28 17:55:29 server squid[8696]: Squid Parent: (squid-1) process 8698 started
Sep 28 17:55:29 server systemd[1]: Started Squid caching proxy.
[root@server ~]#
```

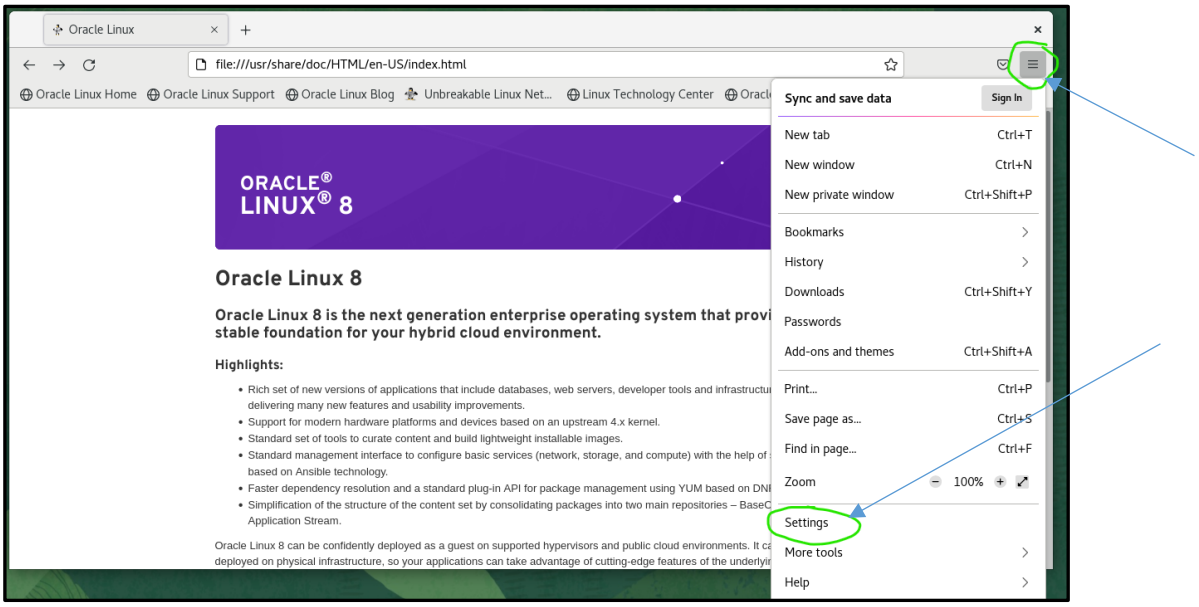
11. You may use the netstat command to check if squid is listening on tcp port 3128:

```
netstat -tunap | grep 3128
```

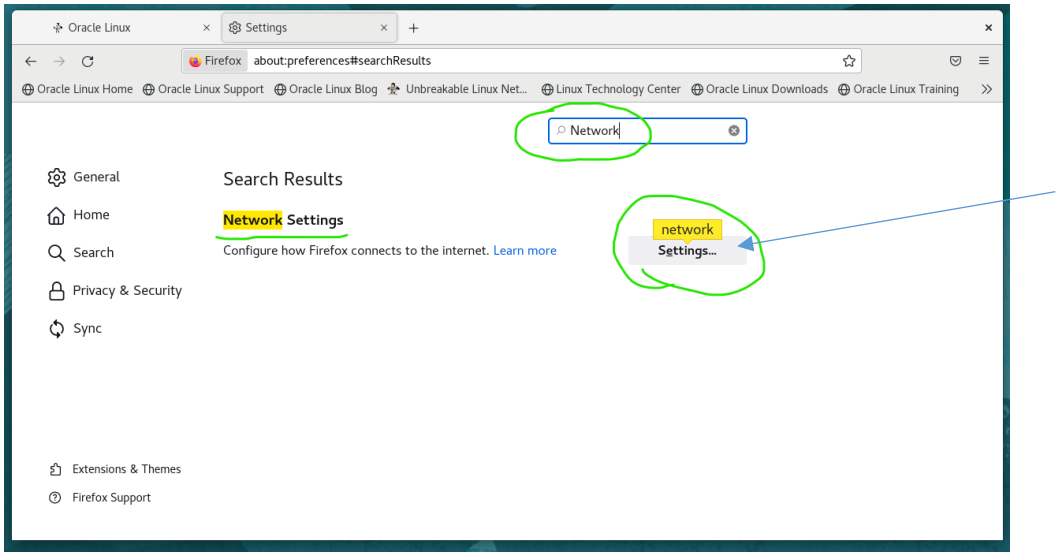
```
[root@server ~]# netstat -tunap | grep 3128
tcp6       0      0 :::3128                 :::*                    LISTEN      8698/(squid-1)

[root@server ~]#
```

<u>On client:</u>

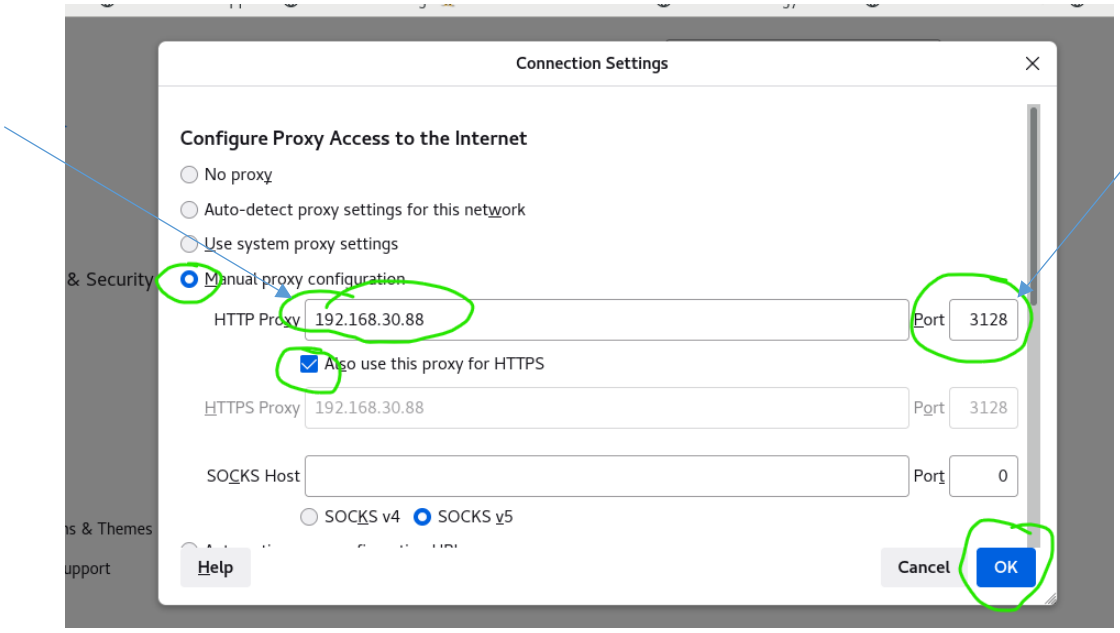
12. Login as student. Configure the Firefox browser to use your proxy that runs on the server. Click on the browser menu icon. Scroll down and click on the Settings menu to bring up the preferences tab.



13. Look for the Network Settings section (or you can use the search feature). Enter to the Settings panel.



At the popup Connection Settings panel, select Manual proxy configuration. Enter your server IP and Port 3128 (Squid's default port) values. Set 'Also use this proxy for HTTPS protocols. Click OK. Click Close.



(The above is assuming the ip address of the server is 192.168.30.88)

14. First visit a few 'unique' http web pages* (at your client VM) then view the Squid access log (at your server VM) to see details of the web page you have accessed.

```
tail /var/log/squid/access.log
```

```
[root@server ~]# tail /var/log/squid/access.log
1664422331.405    806 192.168.30.130 TCP_MISS/200 29057 GET http://www.testingmcafeesites.com/ - HIER_DIRECT/10
0.21.215.181 text/html
1664422331.615    197 192.168.30.130 TCP_MISS/404 1580 GET http://www.testingmcafeesites.com/favicon.ico - HIER
_DIRECT/100.21.215.181 text/html
1664422331.682    417 192.168.30.130 TCP_TUNNEL/200 39 CONNECT pagead2.googlesyndication.com:443 - HIER_DIRECT/
142.250.80.98 -
1664422336.543    199 192.168.30.130 TCP_MISS/200 971 GET http://www.testingmcafeesites.com/testcat_au.html - H
IER_DIRECT/100.21.215.181 text/html
1664422346.439    197 192.168.30.130 TCP_MISS/200 944 GET http://www.testingmcafeesites.com/testcat_bu.html - H
IER_DIRECT/100.21.215.181 text/html
1664422434.726    562 192.168.30.130 TCP_REFRESH_ABORTED/000 0 GET http://www.testingmcafeesites.com/testcat_bu
.html - HIER_NONE/- -
1664422440.755    980 192.168.30.130 TCP_MISS/200 940 GET http://www.testingmcafeesites.com/testcat_hl.html - H
IER_DIRECT/18.236.36.28 text/html
1664422445.062    220 192.168.30.130 TCP_MISS/200 966 GET http://www.testingmcafeesites.com/testcat_gv.html - H
IER_DIRECT/18.236.36.28 text/html
1664422447.503    216 192.168.30.130 TCP_REFRESH_UNMODIFIED/304 345 GET http://www.testingmcafeesites.com/testc
at_gv.html - HIER_DIRECT/18.236.36.28 -
1664422449.348    216 192.168.30.130 TCP_REFRESH_UNMODIFIED/304 345 GET http://www.testingmcafeesites.com/testc
at_gv.html - HIER_DIRECT/18.236.36.28 -
[root@server ~]#
```

This is to prove that your squid proxy is really working.

The log info tells you the ip address of the client , and you will see a few TCP_MISS entries, as these entries have not cached yet.

*Note:

You may refer to the following web page to use the http links listed there.

Url for http testing: http://www.testingmcafeesites.com/

15. At the client, **open new tab** at the firefox browser, and revisit some of the web pages.

Check on the squid access log again (at the server), you may find some log entries show the TCP_MEM_HIT FLAG. These flags imply the squid server is successful provide the web pages content from the cache.

```
[root@server conf.d]# tail /var/log/squid/access.log
1664423211.351      4 192.168.30.130 TCP_MISS/403 534 GET http://192.168.30.88/books/book.html - HIER_DIRECT/19
2.168.30.88 text/html
1664423226.802     13 192.168.30.130 TCP_MISS/200 3964 GET http://192.168.30.88/ - HIER_DIRECT/192.168.30.88 te
xt/html
1664423226.862      4 192.168.30.130 TCP_MISS/200 4639 GET http://192.168.30.88/icons/apache_pb2.gif - HIER_DIR
ECT/192.168.30.88 image/gif
1664423237.048      2 192.168.30.130 TCP_MISS/403 534 GET http://192.168.30.88/books/books.html - HIER_DIRECT/1
92.168.30.88 text/html
1664423360.009 172542 192.168.30.130 TCP_TUNNEL/200 1821 CONNECT www.google.com:443 - HIER_DIRECT/142.251.40.19
6 -
1664423440.136     12 192.168.30.130 TCP_MISS/200 447 GET http://192.168.30.88/books/books.html - HIER_DIRECT/1
92.168.30.88 text/html
1664423444.324      1 192.168.30.130 TCP_REFRESH_UNMODIFIED/304 304 GET http://192.168.30.88/books/books.html -
 HIER_DIRECT/192.168.30.88 -
1664423667.117    628 192.168.30.130 TCP_MISS/200 892 POST http://ocsp.digicert.com/ - HIER_DIRECT/93.184.220.2
9 application/ocsp-response
1664423669.972      0 192.168.30.130 TCP_MEM_HIT/200 980 GET http://www.testingmcafeesites.com/testcat_au.html
- HIER_NONE/- text/html
1664423674.785      0 192.168.30.130 TCP_MEM_HIT/200 29065 GET http://www.testingmcafeesites.com/index.html - H
IER_NONE/- text/html
[root@server conf.d]#
```

16. Now you may try to visit some https web pages (e.g. https://www.google.com). When you check on the squid access log, you can tell the squid proxy is working but there is no more TCP_MISS nor TCP_MEM_HIT cases for **https connections**. They are all reported as TCP_TUNNEL connections. All https traffic is encrypted, and they pass thru the squid proxy, thus, there is no MISS nor HIT cases.

```
[root@server ~]# tail /var/log/squid/access.log
1664424902.943   1389 192.168.30.130 TCP_TUNNEL/200 7079 CONNECT ap.lijit.com:443 - HIER_DIRECT/216.52.2.39 -
1664424903.100    456 192.168.30.130 TCP_MISS/200 892 POST http://ocsp.digicert.com/ - HIER_DIRECT/93.184.220.2
9 application/ocsp-response
1664424903.170   1617 192.168.30.130 TCP_TUNNEL/200 6372 CONNECT cm.g.doubleclick.net:443 - HIER_DIRECT/142.250
.64.98 -
1664424903.454   1062 192.168.30.130 TCP_TUNNEL/200 10181 CONNECT s0.2mdn.net:443 - HIER_DIRECT/142.250.65.198
-
1664424903.539   1046 192.168.30.130 TCP_TUNNEL/200 6050 CONNECT googleads4.g.doubleclick.net:443 - HIER_DIRECT
/142.250.65.226 -
1664424903.603    652 192.168.30.130 TCP_TUNNEL/200 979 CONNECT ap.lijit.com:443 - HIER_DIRECT/216.52.2.39 -
1664424905.761    761 192.168.30.130 TCP_MISS/200 854 POST http://ocsp.pki.goog/gts1c3 - HIER_DIRECT/142.250.65
.163 application/ocsp-response
1664424905.763    767 192.168.30.130 TCP_MISS/200 854 POST http://ocsp.pki.goog/gts1c3 - HIER_DIRECT/142.250.65
.163 application/ocsp-response
1664424906.286   1971 192.168.30.130 TCP_TUNNEL/200 6725 CONNECT fonts.googleapis.com:443 - HIER_DIRECT/142.250
.80.10 -
1664424907.579   1257 192.168.30.130 TCP_TUNNEL/200 22363 CONNECT fonts.gstatic.com:443 - HIER_DIRECT/142.250.6
4.99 -
[root@server ~]#
```

<u>On server:</u>

17. Edit /etc/squid/squid.conf:

Create a couple of acl for websites/domain you are going to block.

You may add these additional acls right after your LAS_net acl.
Look for the following line and add the lines in bold.

acl LAS_net src 192.168.30.0/24

acl bad_sites dstdomain .yahoo.com

acl bad_sites dstdomain .org

For your reference:

```
 GNU nano 2.9.8                          /etc/squid/squid.conf                          Modified

acl localnet src 169.254.0.0/16         # RFC 3927 link-local (directly plugged) machines
acl localnet src 172.16.0.0/12          # RFC 1918 local private network (LAN)
acl localnet src 192.168.0.0/16         # RFC 1918 local private network (LAN)
acl localnet src fc00::/7               # RFC 4193 local private network range
acl localnet src fe80::/10              # RFC 4291 link-local (directly plugged) machines

acl LAS_net src 192.168.30.0/24
acl bad_sites   dstdomain       .yahoo.com
acl bad_sites   dstdomain       .org

acl SSL_ports port 443
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443         # https
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535  # unregistered ports
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line M-E Redo
```

Note: we can add more domains to associated with the 'bad_sites' acl name if needed.

18. Create the http_access rule for bad_sites. Look for the following line and add the line in bold.

    http_access allow LAS_net

    http_access deny bad_sites

    For your reference:

```
 GNU nano 2.9.8                          /etc/squid/squid.conf                          Modified


#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#

http_access allow LAS_net
http_access deny bad_sites


# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
# http_access allow localnet
http_access allow localhost

# And finally deny all other access to this proxy
http_access deny all

# Squid normally listens to port 3128

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  M-U Undo
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line M-E Redo
```

19. **Restart** the squid service.
20. Check /var/log/messages if there are any error messages.

    For your reference:

```
[root@server ~]# systemctl restart squid
[root@server ~]# tail /var/log/messages
Sep 29 14:20:52 server org.gnome.Shell.desktop[2269]: libinput error: event1  - AT Translated Set 2 keyboard: c
lient bug: event processing lagging behind by 15ms, your system is too slow
Sep 29 14:21:00 server systemd[1]: fprintd.service: Succeeded.
Sep 29 14:28:12 server systemd[1]: Stopping Squid caching proxy...
Sep 29 14:28:43 server squid[4209]: Squid Parent: squid-1 process 4212 exited with status 0
Sep 29 14:28:43 server systemd[1]: squid.service: Succeeded.
Sep 29 14:28:43 server systemd[1]: Stopped Squid caching proxy.
Sep 29 14:28:43 server systemd[1]: Starting Squid caching proxy...
Sep 29 14:28:43 server squid[5743]: Squid Parent: will start 1 kids
Sep 29 14:28:43 server squid[5743]: Squid Parent: (squid-1) process 5746 started
Sep 29 14:28:43 server systemd[1]: Started Squid caching proxy.
[root@server ~]#
```

On client:

21. Test if you can still browse https://www.yahoo.com or http://www.acm.org. You are still able to access these sites. This is because the allow rule for LAS_net will allow all http access from your subnet, even those going to destination domain that is marked in the bad_sites.

22. Edit /etc/squid/squid.conf. (at the server)

23. Change the order of your two http_access rules so that the deny rule comes before the access rule.

    http_access deny bad_sites

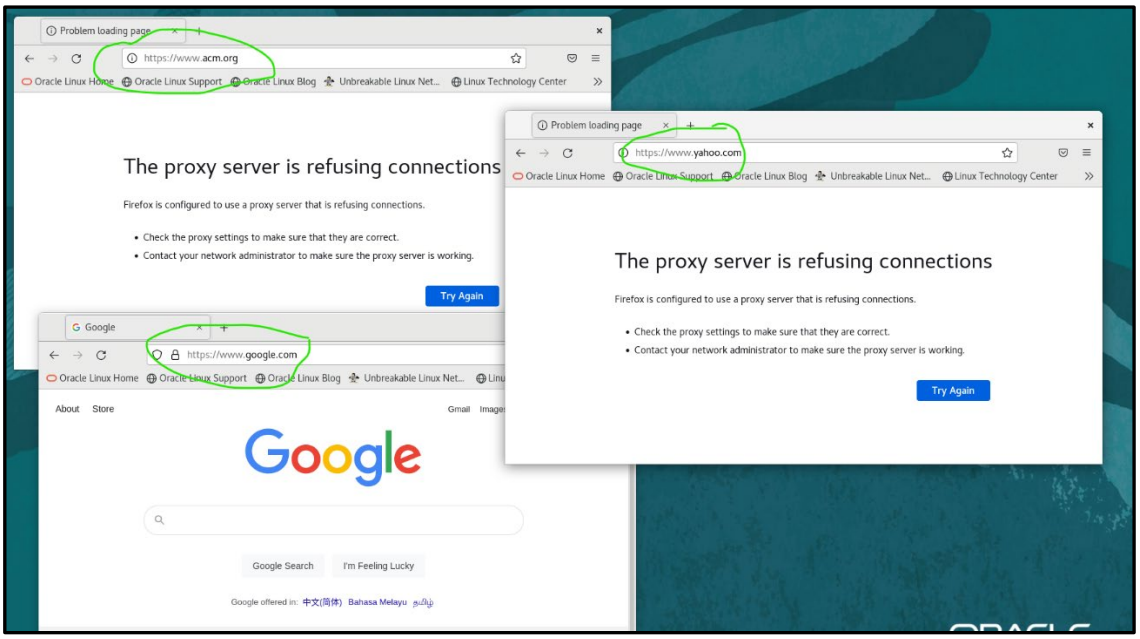
    http_access allow LAS_net

24. **Restart** the squid service.

25. Check /var/log/messages if there are any error messages.

26. Test if you can still browse the two blacklisted domains. You should not be able to access to .yahoo.com nor .org domains.

    For your reference:



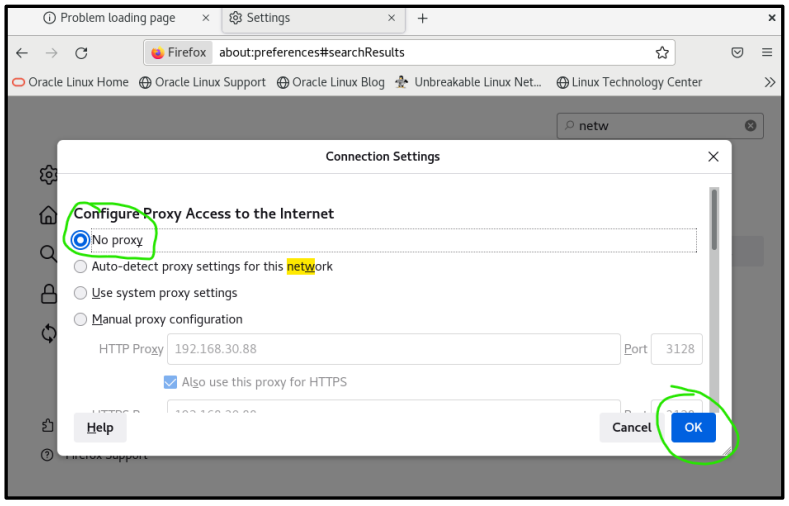
27. View the Squid access log to see details of the web request that have been blocked. (Hint: Look for the key word TCP_DENIED)

```
[root@server squid]# cat access.log | grep DENIED
1664433299.532      0 192.168.30.130 TCP_DENIED/403 4007 CONNECT www.acm.org:443 - HIER_NONE/- text/html
1664433349.303      0 192.168.30.130 TCP_DENIED/403 4025 CONNECT sg.news.yahoo.com:443 - HIER_NONE/- text/html
1664433360.418      0 192.168.30.130 TCP_DENIED/403 4013 CONNECT www.yahoo.com:443 - HIER_NONE/- text/html
1664433377.668      0 192.168.30.130 TCP_DENIED/403 4007 CONNECT www.acm.org:443 - HIER_NONE/- text/html
[root@server squid]#
```

(The above is for your reference.)

On Client:

28. Reset your Firefox web browser to use the original proxy settings (Set back to “use system proxy settings” or “no proxy”) and press the OK button.



29. Test if you can now browse the .yahoo.com and .org domains. (You may need to restart Firefox.)

On Server:

30. You may disable your squid service now. Type:

```
systemctl disable squid
systemctl stop squid
```

The following topics are optional - They will not be tested in the ST2412 MST / EST.

These optional topics may be considered useful to those are interested in web development.

It will also be related to your final assignment, therefore, you are highly recommended to try on it during the mid semester vacation break.

11. Web Server Gateway Interface (WSGI)

"WSGI is a standard described on PEP* 3333 and basically, provides a standard interface between web applications written in Python and Webservers. That means, WSGI gives portability to your Python Web Application across many different Web Servers, without any additional configurations on your NGINX, Apache, etc ..."
(Ref: https://stackoverflow.com/questions/1813394/why-should-i-use-wsgi)
** Python Enhancement Proposal*

Native web servers such as Apache are designed to handle many requests concurrently.

Python based web applications/frameworks are not made to process thousands of requests and determine how to best route them from the server. With WSGI to link up the web server and the Python applications, they can handle thousands of requests concurrently and route them from the web server through the best possible means.

You will install wsgi module on your apache server and a testing python based dynamic web page.

On server:

1. Install the python3 and the essential utilities.

```
dnf install python3 python3-pip -y
```

2. Check which version of python you are using.

```
python3 --version
```

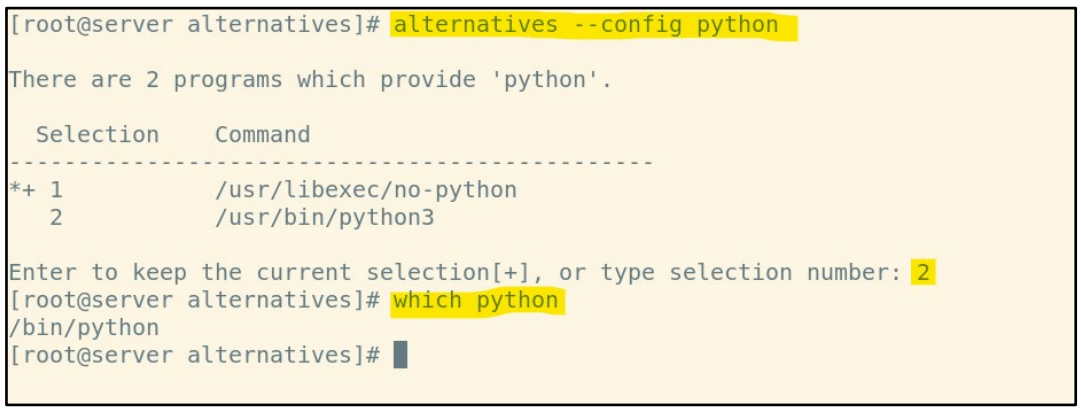
3. Confirm the location of Python3 and set it as the default python for the system.

```
which python3
which python
```

As shown in the above, there is no python being installed in the system.

We can use the interactive alternatives command to associate 'python' with the installed python3 via a symbolic linkage.

```
alternatives --config python
```



As shown, the alternatives package provides 2 options to associate with 'python'. We can choose the 2nd option.

You can verify your settings with

```
python --version
```

```
python3 --version
```

```
[root@server alternatives]# python3 --version
Python 3.6.8
[root@server alternatives]# python --version
Python 3.6.8
[root@server alternatives]#
```

4. Install the Apache WSGI module:

```
dnf -y install python3-mod_wsgi
```

5. Finally, restart your Apache service and check if wsgi_module is load.

   Use the following to list out all the loaded Apache modules

```
httpd -M | grep ^wsgi_module
```

```
[root@server squid]# systemctl restart httpd
[root@server squid]# httpd -M | grep wsgi_module
 proxy_uwsgi_module (shared)
 wsgi_module (shared)
[root@server squid]#
```

12. Deploying a python-based dynamic web page to the Apache2 server

In real applications, Python developers may use powerful frameworks (DJango, Flask, …)to implement the application servers. In this exercise, we use a simple python script to demonstrate the working concept of WSGI.

<u>On server:</u>

1. Create a new folder to store the WSGI web content

```
mkdir -p /var/www/las_wsgi
```

2. Create the file /var/www/las_wsgi/test_wsgi.py with the following content:

```
from datetime import datetime

def application(environ, start_response):
    status = '200 OK'
    part1 = '''
    <html>
    <body>
        <div style="width: 100%; font-size: 40px;
        font-weight: bold; text-align: center;">
            LAS WSGI Test Page
```

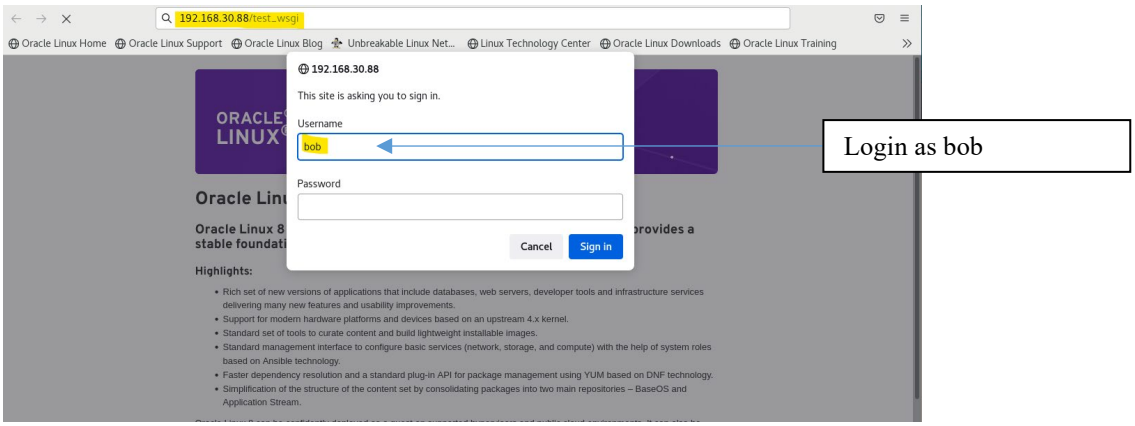
```
        </div>
        <div style="font-size:24px; text-align:center;">
    '''
    part2= '''
        </div>
    </body>
    </html>
    '''
    now = datetime.now()
    dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
    html_text = f"{part1}{dt_string}{part2}"

    html=html_text.encode("utf-8")
    response_header = [('Content-type','text/html')]
    start_response(status,response_header)
    return [html]
```

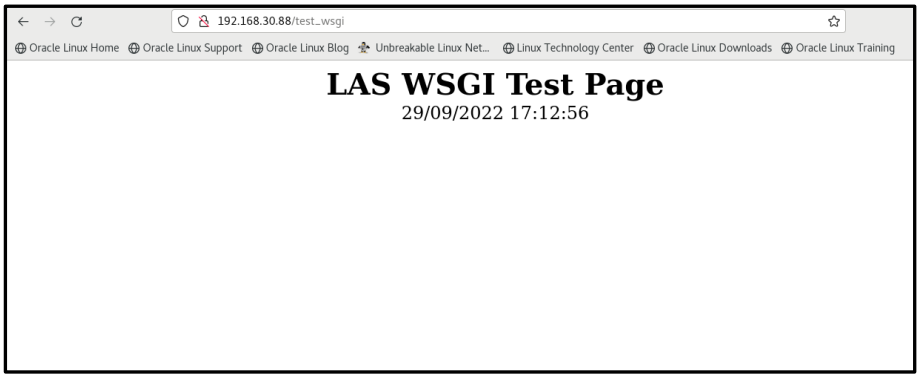
(Note: The above is a python3 source code, so you need to ensure the indentations are all correctly aligned.)

3. Create the file /etc/httpd/conf.d/las_wsgi.conf with the following content:

```
WSGIScriptAlias  /test_wsgi  /var/www/las_wsgi/test_wsgi.py
<Directory /var/www/las_wsgi >
  AuthType basic
  AuthName "WSGI DEMO"
  AuthUserFile /etc/httpd/conf/flowers-users
  Require all denied
  Require user bob alice
</Directory>
```

4. Restart your Apache server to let the new configuration take effect.
   Browse to:
   http://<Your ServerIP>/test_wsgi





This web page contains dynamic content of the current date time. Try to reload the page and you will see the effect.

(Note: wsgi also works with https, you can try it.)

13. Load Tests for a web site/web page

Load test for a web site is a must before deploying the web site to the production system. From the result of the load tests, you may discover hidden bugs or performance issues in various parts of the system.
There are many tools available to perform load testing on web servers. We will be using Apache JMeter. Apache JMeter requires the Java Run-time Environment (JRE) which may not be installed to our system yet.

You will normally run load tests to verify the performance and reliability of your Web Server under simulated loading. As the Apache HTTPD web server is running on your client, we will install and run Apache JMeter to load test on your server

On client:

1. Install java openjdk to your system:

```
dnf install java-17-openjdk-devel -y
```

To verify the installation:

```
java --version
```

```
[root@client ~]# java --version
openjdk 17.0.2 2022-01-18 LTS
OpenJDK Runtime Environment 21.9 (build 17.0.2+8-LTS)
OpenJDK 64-Bit Server VM 21.9 (build 17.0.2+8-LTS, mixed mode, sharing)
[root@client ~]#
```

2. Download the Apache JMeter binaries from **jmeter.apache.org.**

You may download and save the tgz (tar gzip) file and the corresponding sha512 hash checksum file.

3. Verify the tgz file against the sha512 checksum:

```
[student@client ~]$ cd ~/Downloads/
[student@client Downloads]$ ls
apache-jmeter-5.5.tgz  apache-jmeter-5.5.tgz.sha512
[student@client Downloads]$
```

Change directory to the folder that contains the tgz and sha512 files. (Should be the Download folder.)

To compute the sha512 checksum value of the downloaded tgz file, type:

sha512sum <Target file path/name>

```
[student@client Downloads]$ ls
apache-jmeter-5.5.tgz  apache-jmeter-5.5.tgz.sha512
[student@client Downloads]$ sha512sum apache-jmeter-5.5.tgz
d5d1ce795e9baf18efd3a13ecda150b4da80c3173a2c7ef0da2a5546ac6862b1edd2a2f4e52d971c
7da05d879362c28dca6bf218c5f7570b5cc98f7ba73c92af  apache-jmeter-5.5.tgz
[student@client Downloads]$ cat apache-jmeter-5.5.tgz.sha512
d5d1ce795e9baf18efd3a13ecda150b4da80c3173a2c7ef0da2a5546ac6862b1edd2a2f4e52d971c
7da05d879362c28dca6bf218c5f7570b5cc98f7ba73c92af *apache-jmeter-5.5.tgz
[student@client Downloads]$
```

Verify the output of the sha512sum matches the content of the sha512 file.
In case the values do not match. You need to re-download the tgz file.

Note: This checksum verification is to protect you from downloading a corrupted file.

4. Extract the files from the JMeter tarball from the tgz file.

```
tar -xvzf <The tgz file>
```

(Note. You may download a newer version than 5.5)

```
[student@client Downloads]$ ls
apache-jmeter-5.5.tgz  apache-jmeter-5.5.tgz.sha512
[student@client Downloads]$ tar -xvzf apache-jmeter-5.5.tgz
```

The tar command shall extract the binary content of the Jmeter to a new subfolder:

```
apache-jmeter-5.5/docs/api/org/apache/log/package-tree.html
[student@client Downloads]$ ls
apache-jmeter-5.5  apache-jmeter-5.5.tgz  apache-jmeter-5.5.tgz.sha512
[student@client Downloads]$
```

In this case, the new folder is apache-jmeter-5.5

5. Change directory to the JMeter **bin** directory.

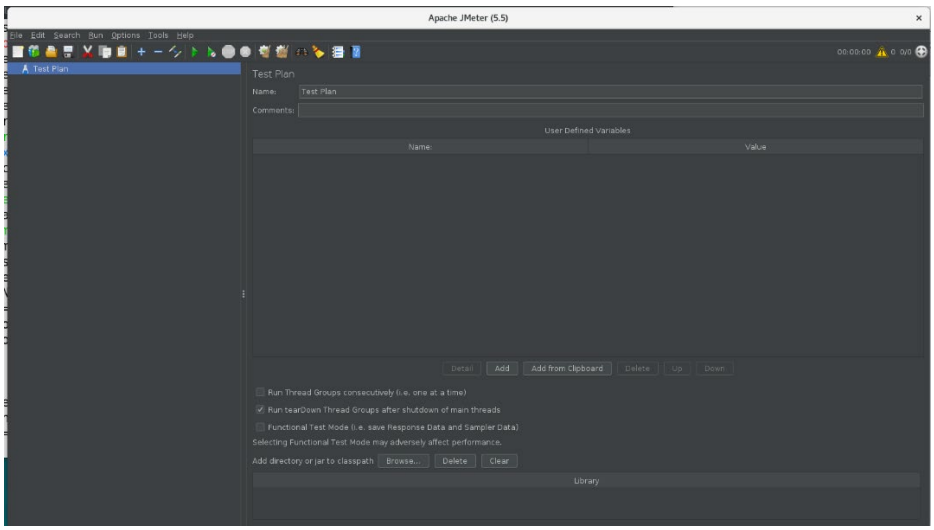
```
cd apache-jmeter-5.5/bin
```

```
[student@client Downloads]$ ls
apache-jmeter-5.5  apache-jmeter-5.5.tgz  apache-jmeter-5.5.tgz.sha512
[student@client Downloads]$ cd apache-jmeter-5.5/bin
[student@client bin]$ ls
ApacheJMeter.jar          jmeter-n.cmd              report-template
BeanShellAssertion.bshrc  jmeter-n-r.cmd            saveservice.properties
BeanShellFunction.bshrc   jmeter.properties         shutdown.cmd
BeanShellListeners.bshrc  jmeter-server             shutdown.sh
BeanShellSampler.bshrc    jmeter-server.bat         stoptest.cmd
create-rmi-keystore.bat   jmeter.sh                 stoptest.sh
create-rmi-keystore.sh    jmeter-t.cmd              system.properties
examples                  jmeterw.cmd               templates
hc.parameters             krb5.conf                 threaddump.cmd
heapdump.cmd              log4j2.xml                threaddump.sh
heapdump.sh               mirror-server             upgrade.properties
jaas.conf                 mirror-server.cmd         user.properties
jmeter                    mirror-server.sh          utility.groovy
jmeter.bat                reportgenerator.properties
[student@client bin]$
```

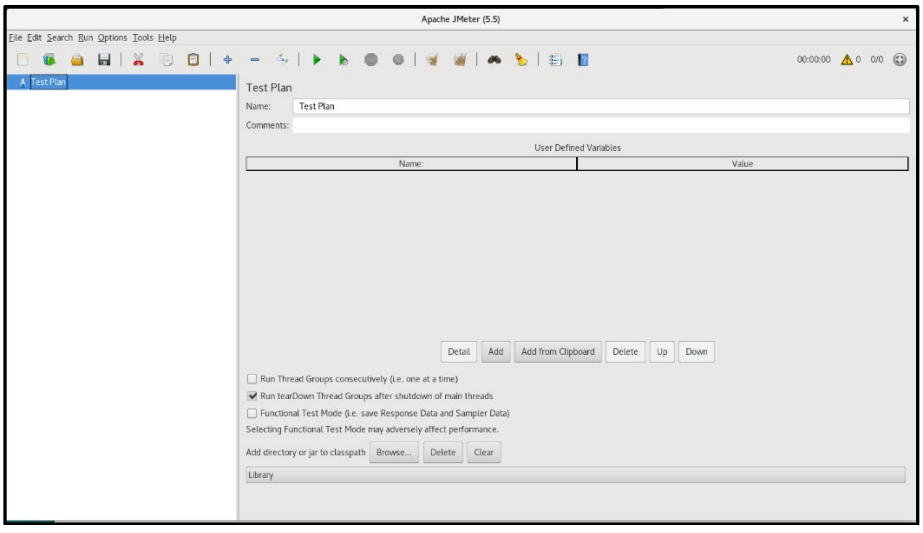
The executable binary file to start Jmeter is the <JMeterFolder>/bin/jmeter.

6. Run JMeter.

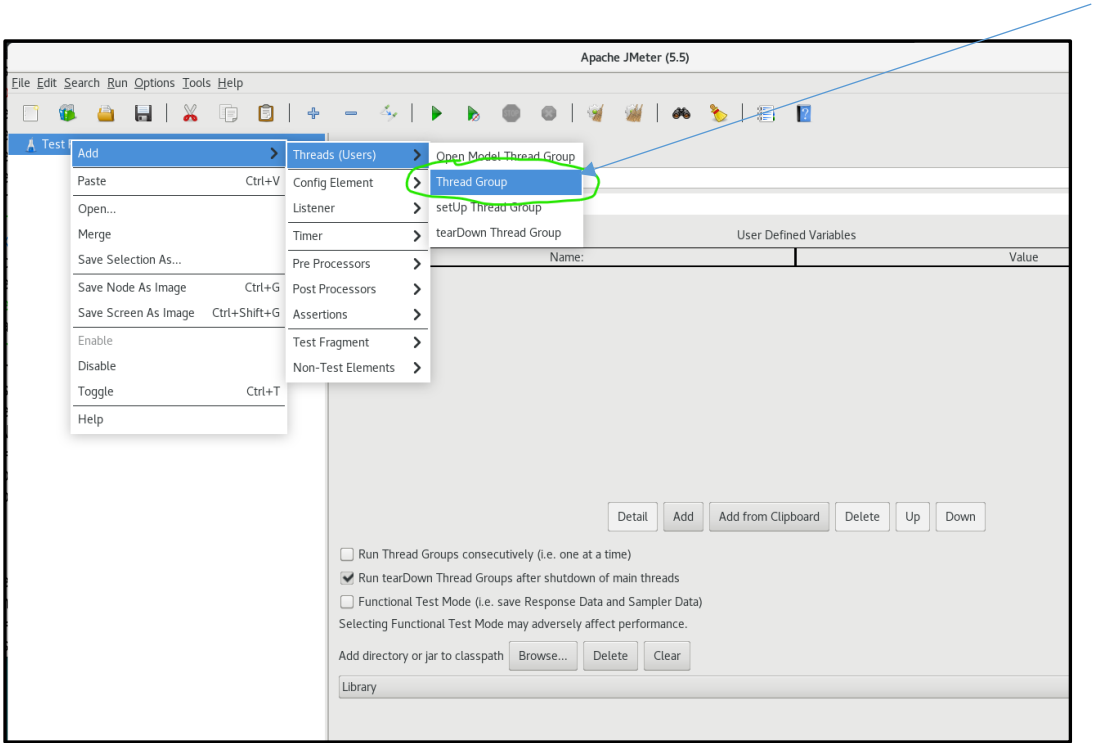
```
./jmeter
```



You may switch to the other color scheme similar to the following by select the 'system' scheme from the Option -> Look and feel menu.



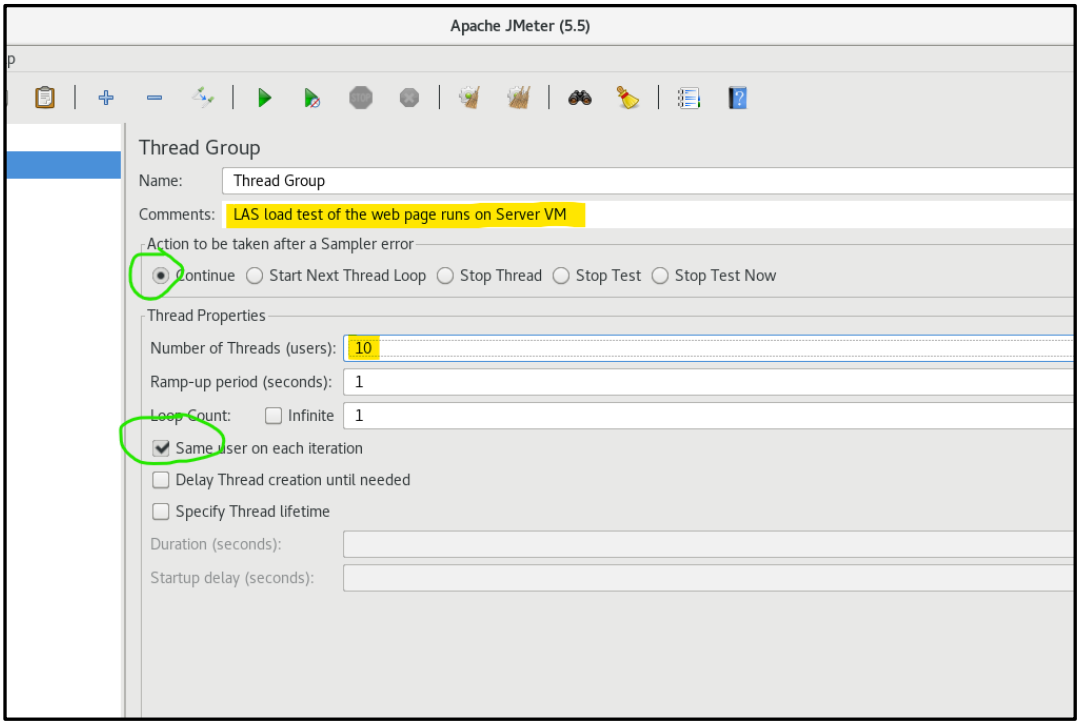
7. Define a new test plan.



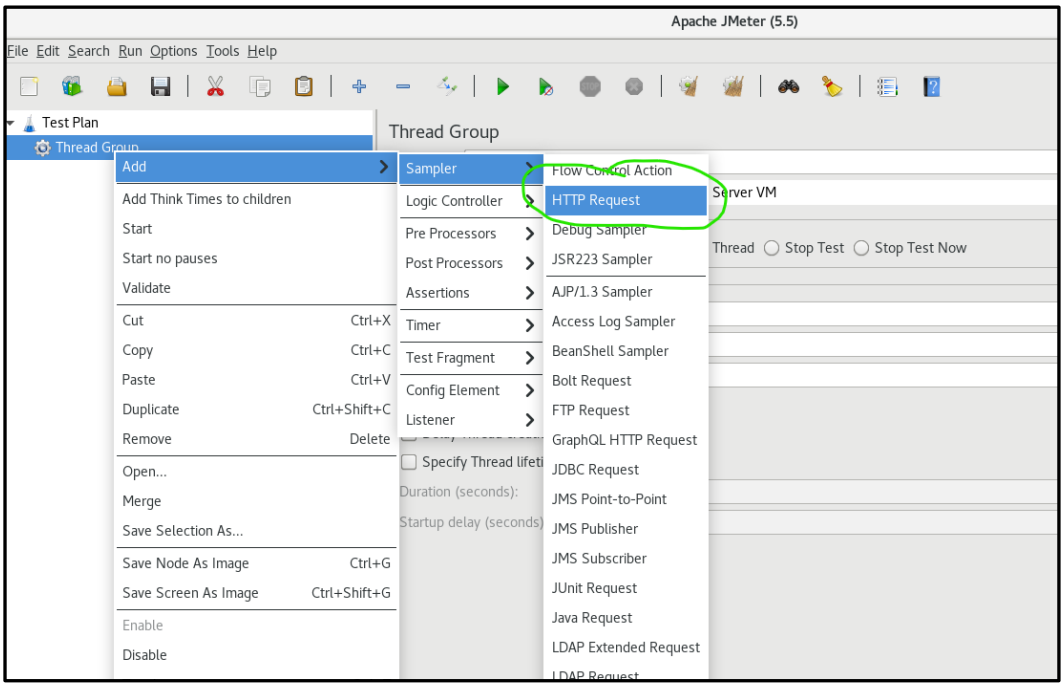
Right click on Test Plan and select Add, Threads(Users) then click at Thread Group.

8. Change Number of Threads (users) to “10” to mimic 10 users.

   You may type in some meaningful comment to describe the purpose of the test.



9. In the left-hand side column, right-click on Thread Group and select Add, Sampler, HTTP Request.



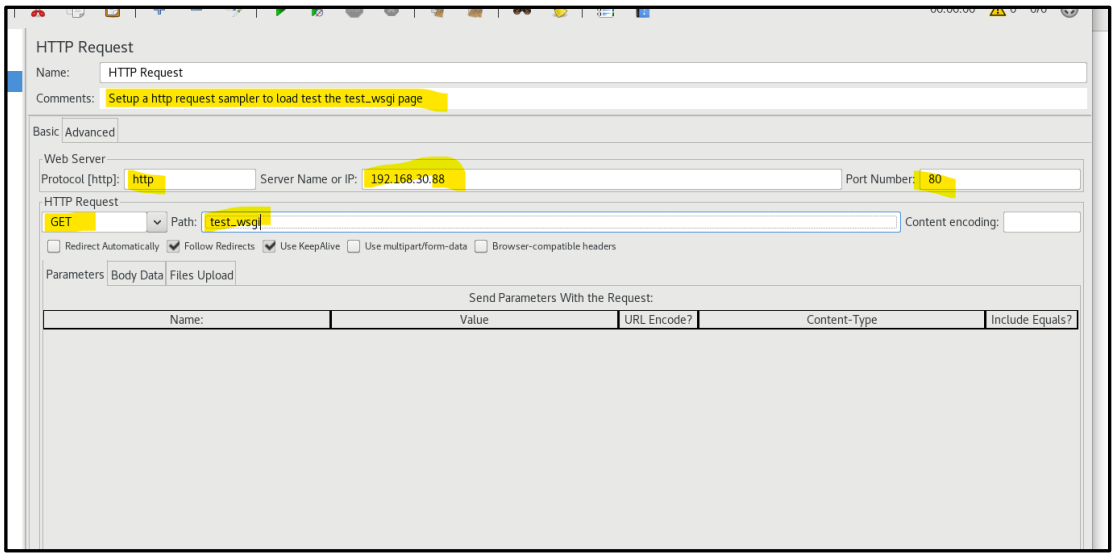
10. Fill out the parameters for the HTTP Request:

    Comments: Setup a http request sampler to load test the test_wsgi page.

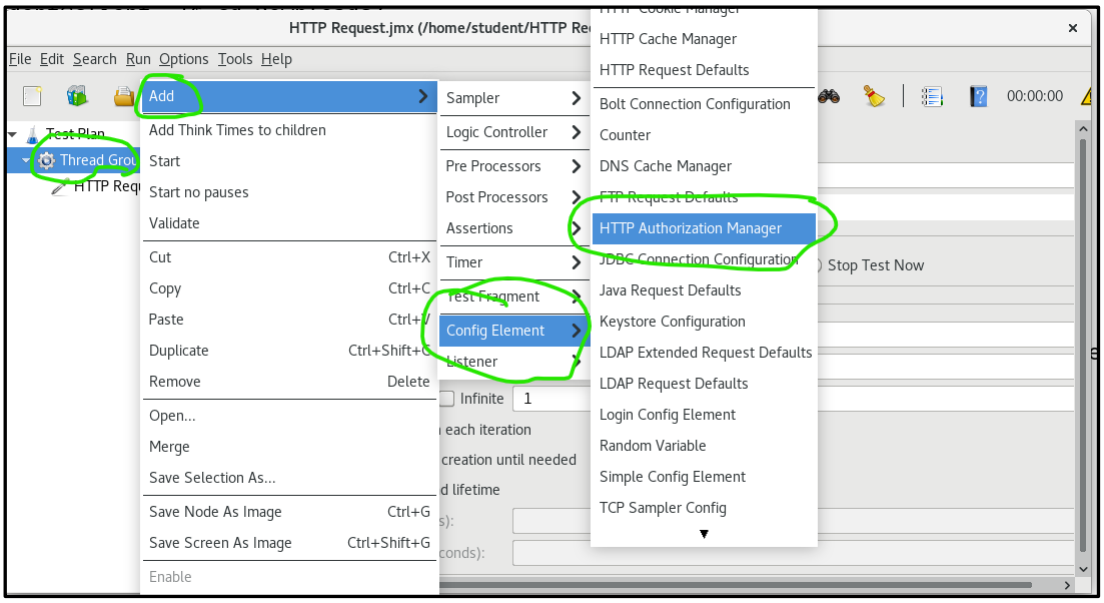
    Under Web Server, enter http for the protocol, your server IP address and the Apache web service port number (ie. 80).

    For Method, select GET.

    For Path, if you are testing the default home page of the Apache server, leave it blank. In our case, you set the Path to test_wsgi to let the JMeter to test run your test_wsgi page. (Note: Your test_wsgi requires basic authentication, therefore, you need to follow the next step to add in a HTTP Authorization manager in this thread group.).



11. In the left-hand side column, right-click Thread Group and select Add, Config Element, HTTP Authorization Manager.



12. Fill out the authentication credential to access to the target web page:

    You may enter some meaningful comment for the comments field.

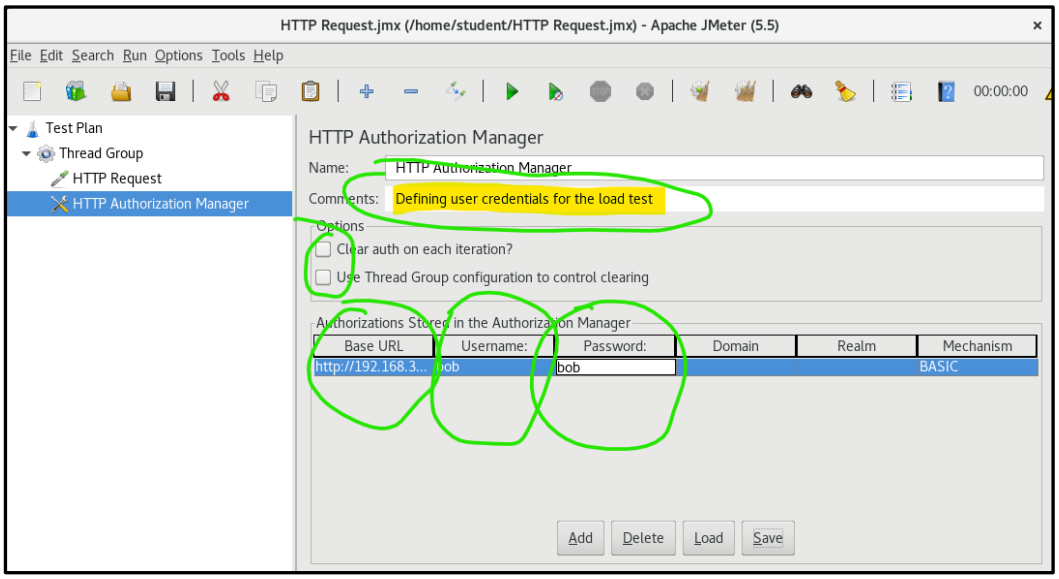
    Click at the Add button to define a new authentication entry.

    Set Base URL to http://192.168.30.88/test_wsgi. (Your URL may be different.)

    Set Username to 'bob'.

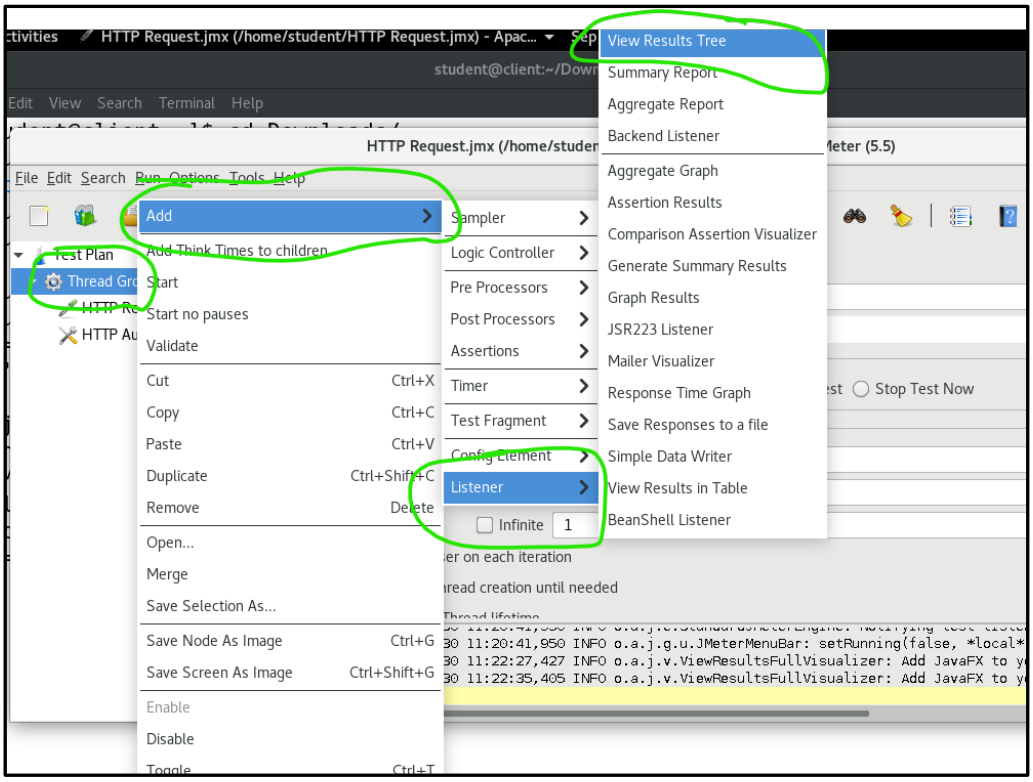
    Set Password to 'bob'.

    If you want to save the entry to a separate text file, you may click the save button.

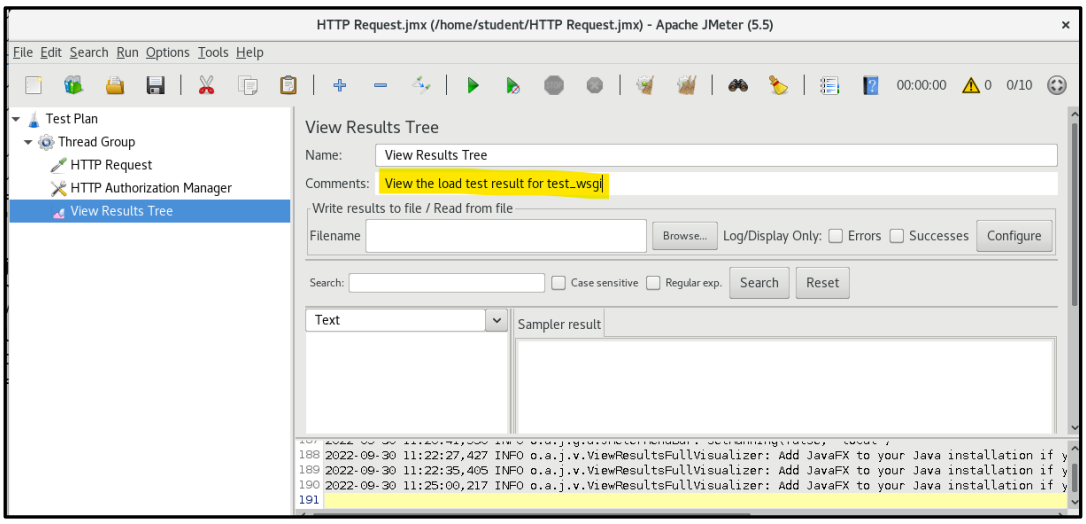


13. At this point, your load test is runnable. You may try to run it by clicking the run menu. However, there is no test result captured. To capture the test result, to need to add in a listener element.

    In the left-hand side column, right-click Thread Group and select Add, Listener, View Results Tree.



    To keep it simple, we do not fill out any specific configuration for this View Results Tree listener element.

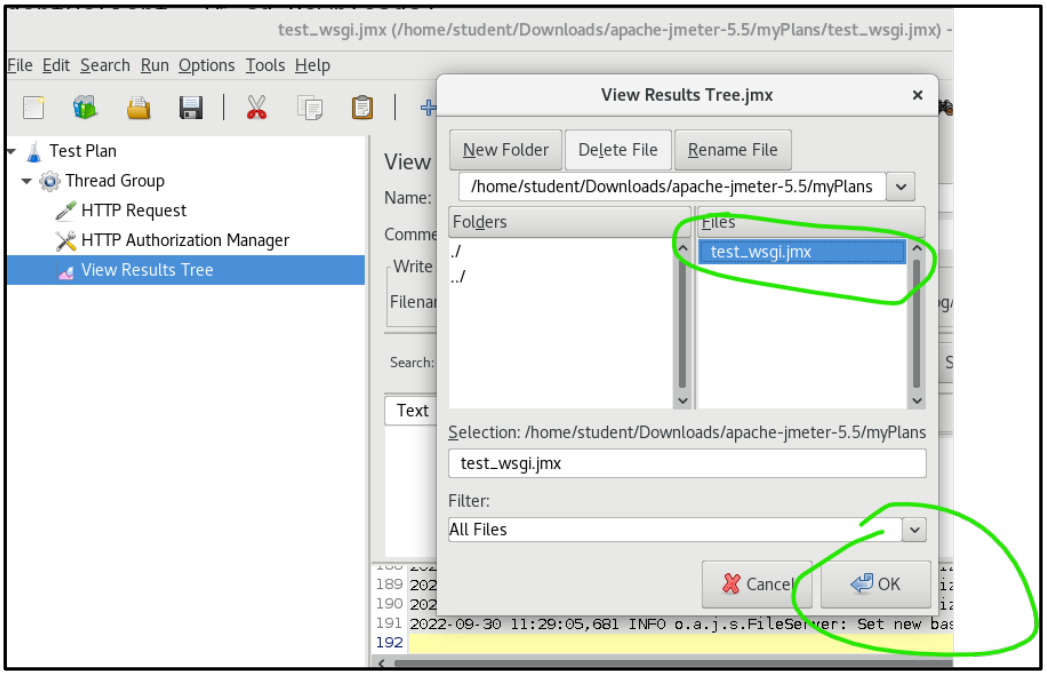


    As shown at the above, we only filled out the comment field.

14. All the basic definitions of the test plan has been completed. We can save the test plan to a file.

    At the left-hand column, select Test Plan. From the File menu, select "Save Test Plan as" and save your test plan as "test_wsgi.jmx" in your preferred directory. In the following

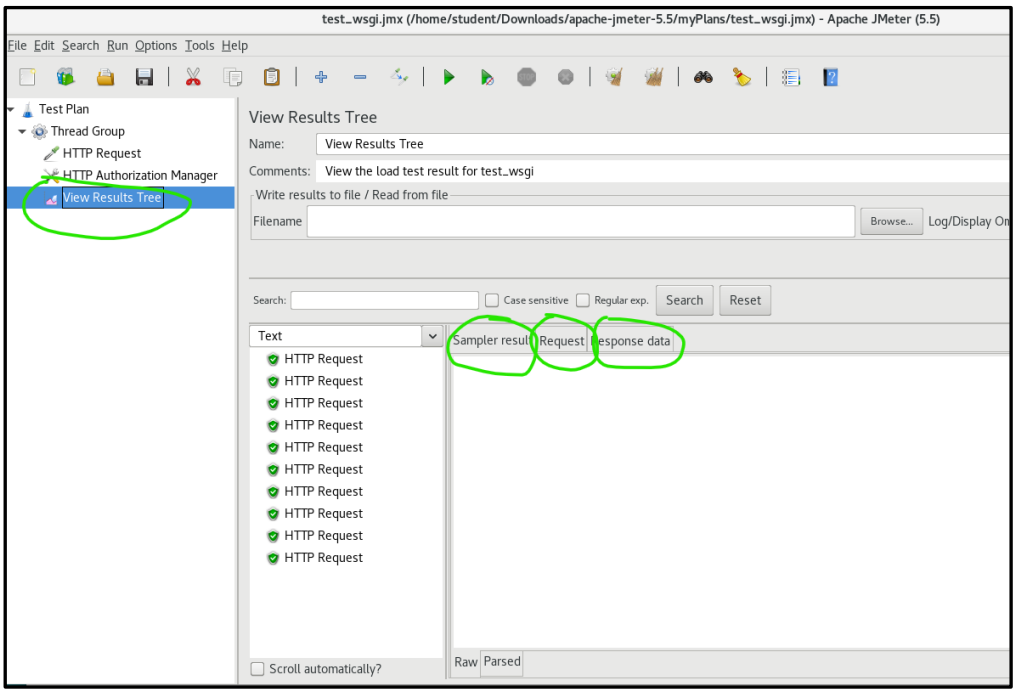
example, the target path is "/home/student/Downloads/apache-jmeter-5.5/myPlans".



15. From the Run menu, select Start to start the test (Or click at the start button).

    The test only takes a few seconds to complete since our test plan is a very simple one and it only iterates 10 times.

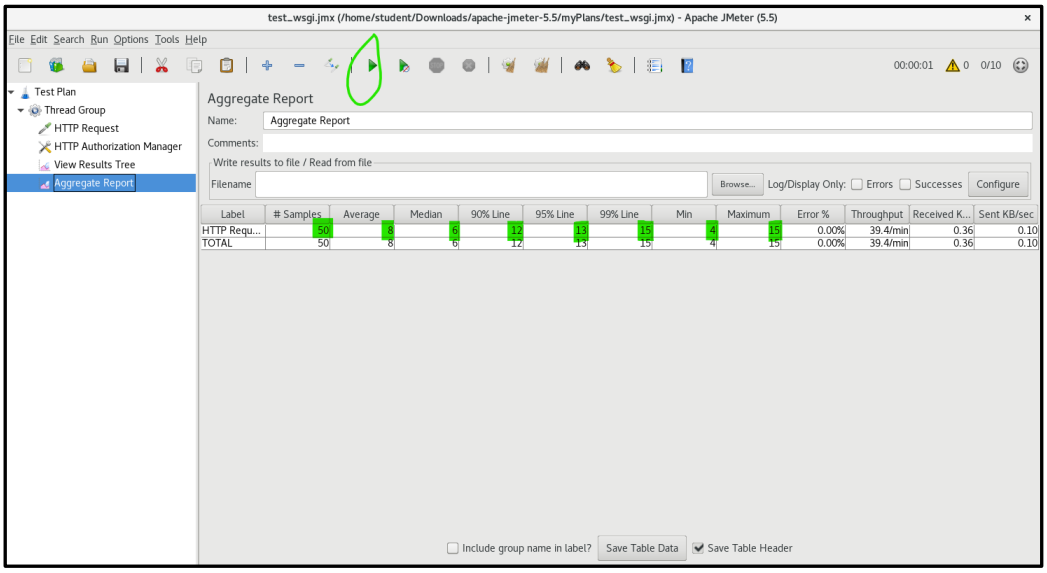
    To view the test result. Select the View Result Tree Node at the left-hand side column. You will see ten HTTP Requests will be listed. Select any one of them to see the results of that HTTP request. You can see the Load time and Latency (in milliseconds) of each HTTP request.



16. You can click on the Request and Response data tabs to see the data being transmitted.
17. To view better reports, right-click Thread Group and select Add, Listener, Aggregate

Report.

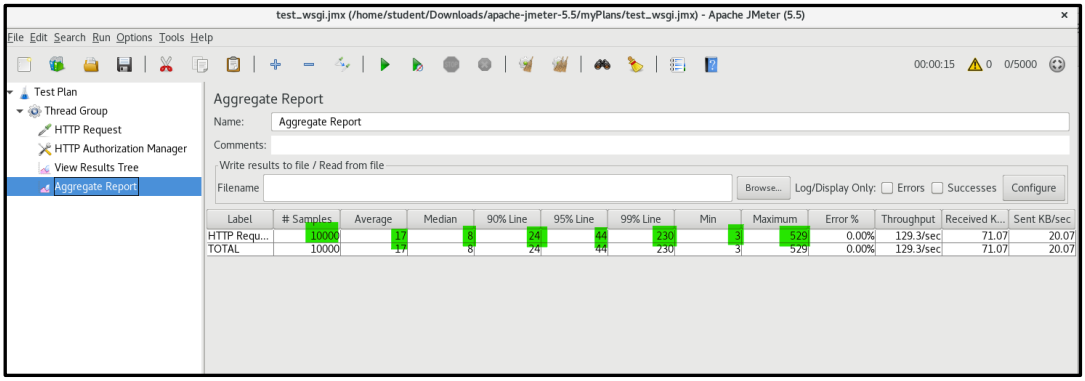
18. From the **Run menu**, select **Clear All** to clear the previous test results.
19. From the Run menu, select Start to run the test again.
20. Click on Aggregate Report to view the statistics of the HTTP requests. All times are shown in the unit of milliseconds.



The above sample is obtained after 5 runs of the test plan, thus, the results are based on the total of 50 sample entries. The above Report shows the Average completion time for each of the http requests is 8 milliseconds (ms). The minimum time is 4 ms, the maximum time is 15 ms. 90% of the requests take no more than 12 ms to complete while 99% of the requests take no more than 15 ms to complete.

21. To test with a heavier loading case. First clear all the existing result data. In the left hand column, click on Thread Group and change the Number of Threads to “5000” to mimic 5000 users accessing your apache Web Server.

    Clear all the previous results and repeat the Load Test and compare the results when 10 users were simulated.



Based on the above, 2 runs of the heavier loading test with 5000 simulated users each. The average time is gone up to 17 ms and the maximum time is gone up to 529 ms. (ie. 0.529 seconds)

You can try this Load Test against the other pages of your Httpd server. Adjust the number of server processes to view the different outcome.

Note : this is just a simple test in our lab environment, so do not form any conclusions on different web server performance just on this. However, load testing is an essential test you need to carryout to ensure the quality and reliability of a web server (and the associated application server). Sure it will help to troubleshoot a web application and impress your lecturer / future clients with a sound load testing report for your school assignments or real projects.

Additional Reference:

- 15 Tips On How to Use 'Curl' Command in Linux - https://www.tecmint.com/linux-curl-command-examples/
- How to install mod_ssl on RHEL 8 / CentOS 8 with httpd Apache webserver - https://linuxconfig.org/how-to-install-mod-ssl-on-redhat-8
- How to Install Apache mod_wsgi Module on CentOS 8 - https://tecadmin.net/install-apache-mod-wsgi-centos-8/
- Install Python On CentOS 8 - https://phoenixnap.com/kb/install-python-on-centos-8

End of Practical