**Part 7- Practical 5**

**Aim:** Install, Configure, And Troubleshoot Linux Web Server (Apache).

**How Web server Works**

First, your browser sends a request to the server.

The server takes the requested file or page from you and maps it to the corresponding file from the server. The server sends the file back to the browser with some information such as its MIME type, the length of the content and some other useful information.

Sometimes the requested file is a static page like HTML pages or dynamic pages like PHP, Java, Perl or any other server-side language.

For example, when you type www.yourDomain.com, the browser queries the [**DNS server**](https://likegeeks.com/linux-dns-server/) about the IP address of the computer: www.yourDomain.com. Once the browser gets the response from the DNS, it starts a TCP connection on port 80 and asks for the default web page, then this page is sent to you and that’s all.

**Install Apache Web server**

You can install Apache server on Red Hat based distros using the following command:

$ dnf -y httpd

Or if you are using a Debian-based distro, you can install it like this:

$ apt-get -y install apache2

The Apache web server service is called **httpd** on Red Hat based distros like CentOS, while it is called **apache2** in Debian based distros.

If you are using a firewall like iptables, you should [**add a rule**](https://likegeeks.com/linux-iptables-firewall-examples/#Adding-iptables-Rules) for port 80.

$ iptables -I INPUT 1 -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT

Or if you are [**using firewalld**](https://likegeeks.com/linux-security-tricks/#Using-Firewalld), you can use the following command:

$ firewall-cmd --add-port=80/tcp

To start your service and enable it on boot:

$ systemctl start httpd

$ systemctl enable httpd

You can check if your service is running or not, using the following command:

$ systemctl status httpd

Now open your browser and visit **http://localhost** or**http://[::1]/** if you are using IP v6 and if your installation goes well, you should see your HTML homepage.

**Configuring Apache Web server**

You can add files to Apache in the

/var/www/html

 directory for top-level pages.

Just remember to make sure that any files or directories placed in that directory are world-readable.

The default index page is index.html.

The Apache configuration files are in

/etc/httpd/conf/

 directory.

On Debian based systems like Ubuntu, you may find it at

/etc/apache2/apache2.conf

 file.

We can’t discuss every option for Apache on a single post, but we will discuss the most important options.

You call them options or directives.

**Virtual Host Types**

There are two types of virtual hosts that you can define in Apache web server:

* Name-based virtual hosts
* IP-based virtual hosts

The NameVirtualHost directive defines which addresses can be virtual hosts; the asterisk (\*) means any name or address on this server. You can write them like this:

NameVirtualHost \*

<VirtualHost \*>

ServerName www.example.com

DocumentRoot "/home/user1/public\_html/"

</VirtualHost>

<VirtualHost \*>

ServerName www.example2.com

DocumentRoot "/ home/user2/public\_html/"

</VirtualHost>

If you have more than one IP address and you want to use SSL certificate, the website must be on a dedicated IP address. You can write IP-based virtual hosts like this:

<VirtualHost 192.168.1.2>

ServerName www.example.com

DocumentRoot "/home/user1/public\_html/"

</VirtualHost>

<VirtualHost 192.168.1.3>

ServerName www.example2.com

DocumentRoot "/ home/user2/public\_html/"

</VirtualHost>

**Apache Process Ownership**

We know from the [**Linux process management**](https://likegeeks.com/linux-process-management/) that each process inherits its permissions from its parent process.

This fact is true for all processes except for applications with the SETUID bit set, they inherit permissions from the owner, not the parent process. A good example is the /bin/su.

If a normal user runs /bin/su program, it does not inherit the permission from adam, but it acts as a root user running it.

Since Apache web server needs to bind port 80, and this needs root privileges.

After binding to port 80, Apache can run as a normal user and only read files that have permissions to read them.

Based on the [**Linux distro you use**](https://likegeeks.com/best-linux-distro-2017/), the user could be one of the following:

nobody, www, apache, www-data, or daemon.

I delayed introducing two more options for apache till reaching that point.

**User Option**

This specifies the user ID which the web server will use to answer requests.

User www-data

**Group Option**

This specifies the group that Apache web server will use to read files.

Group www-data

Security is very important for sites that use executable scripts such as CGI or PHP scripts.

The user you will use will have the [**permission to read and write**](https://likegeeks.com/main-linux-commands-easy-guide/#chmod-Command) the content of all sites on the server. But we want to ensure that only the members of a particular site can read their own site only.

This is very important because if a site got compromised, the attacker will be able to read all files since the apache user has permission to do that.

So how to solve this problem?

**suEXEC Support**

A popular method is to use suEXEC. suEXEC is a program that runs with root permissions and makes CGI programs run as the user and group IDs of a specific user, not the Apache server user.

You can specify the user on each virtual host like this:

<VirtualHost www.example.com>

SuExecUserGroup adam adamGroup

</VirtualHost>

Just that simple.

**Apache Authentication**

You may want to restrict some parts to specific visitors. It’s like a password protected directory.

In Apache, you can store authentication information file called *.*htpasswd file.

You can use the htpasswd command to do that.

First, create the .htpasswd file using the htpasswd command:

$ htpasswd -c /home/adam/.htpassswd myuser

The -c option is needed the first time you run htpasswd, but when you need to add more users you shouldn’t use -c because it will overwrite the file.

Then create a .htaccess file in the public\_html folder and write the following:

<Location /vip>

AuthName "test"

AuthType Basic

AuthUserFile /home/adam/.htpasswd

Order deny,allow

require valid-user

</Location>

AuthName is required, you can use any string you want.

AuthType Basic says that you’re using htpasswd style user file.

AuthUserFile points to the file that contains the generated password from htpasswd command.

The Order line indicates that Apache must deny access by default, and only allow access for users specified in the htpasswd file.

The require directive means any user in the .htpasswd file is allowed.

**Troubleshooting Apache Web server**

If you modify the httpd.conf file and restart or reload Apache web server and it did not work, then you have typed a wrong configuration, however, this is not the only case that you need to troubleshoot Apache, you may look at the apache logs to see how the service works so you can diagnose the problem and solve it.

The two main log files for apache are**error\_log** and **access\_log** files.

You can find these files in

/var/log/apache2/

  directory if you are using Debian based distros.

The access\_log file contains every request to Apache web server with the details about client requested that resource.

The error\_log file contains errors of Apache web server.

You can use [**tail command**](https://likegeeks.com/basic-linux-commands-part2/#Viewing-End-of-File) to watch the log file:

$ tail -f /var/log/httpd/error\_log