APACHE [HTTP]

* It is most popular, powerful and secure web server in the world.
* Port = **80/http** , **443/https**.
* Daemon = **httpd**.
* Package = **httpd**.
* Document root = **/var/www/html**.
* Config file = /**etc/httpd/conf/httpd.conf**.
* **httpd -t** = syntax check.
* If http repository is not responding while installing = **yum clean all**.
* Install apache = **yum install httpd**
* Start the service = **service httpd start**
* Open your browser, type your ip, if it shows default httpd page means that you are installed apache successfully.
* Go to **/var/www/html**, write some content in it, and again search your ip, it will shows you the content you wrote in html dir.
* To see logs along with **date** and **time** format add **“ %D”** in **log format combined** linein httpd.conf file.
* To secure apache with **https**, you need to install ssl in your system.
* **yum install mod\_ssl openssl** = to install ssl.
* After installing ssl. You have to generate private key, csr (certificate signing request) and self signed certificate.
* **openssl genrsa –out red.key 2048** = To create private key.
* **openssl req –new –key name.key –out name.csr** = To create certificate signing request.
* It will ask you details about country, state, company etc. fill it up all the details.
* **openssl –x509 –req –days 365 –in name.csr –signkey name.key –out name.crt** = To create self signed certificate.
* After creating keys, move them to their respective directories.

**Cp name.crt /etc/pki/tls/certs/**

**Cp name.key /etc/pki/tls/private/**

**Cp name.csr /etc/pki/tls/private/**

* Go to **/etc/httpd/conf.d/ssl.conf** file, under virtual host line, uncomment the **document root** and change **servername** to your requirement.
* Find **sslcertificatefile** and **sslcertificatekeyfile** and replace the keys with your keys & restart.
* Go to browser, search your ip with https url, you will see security error with https extension.
* Ignore (or) add an exception to that error, then you will see your web page along with self signed ssl certificate extension (https).

**VIRTUAL HOST**

* It is a method of hosting multiple websites in a single server.
* There are types of virtual hosting in apache.

**Name based hosting.**

**Ip based hosting**

**Port based hosting**.

**NAME BASED HOSTING**

* It is used for hosting multiple websites with a single ip in a single server.
* We have to create two virtual hosts for two domains and map those domains with ip in **/etc/hosts** file.
* Go to **httpd.conf**, copy the last 7 lines of virtual host and edit the lines according to your requirement.
* In Centos 7, Virtual host lines are not available by default. You have to copy them from outside, create a new conf file in conf.d dir, paste and edit the lines as per your requirement. mention your newly created virtual hosts file in httpd.conf file.
* **IncludeOptional conf.d/\*.conf**…..add this line in the last of httpd.conf in centos 7. To read conf files which are in conf.d directory.

**<VirtualHost \*:80>**

**ServerName www.test101.com**

**ServerAlias test101.com**

**DocumentRoot /var/www/html/test101/**

**Errorlog "/var/www/html/test101/error.log"**

**Customlog "/var/www/html/test101/access.log" common**

**</VirtualHost>**

* Copy the above lines in a file and change them according to your requirement for 2nd domain (server, document root, errorlog file, accesslog file).
* Check the syntax with **httpd –t** command.
* If you face any error like this **“[warn] \_default\_ VirtualHost overlap on port 80, the first has precedence “** .just add your virtual host in httpd.conf.

**NameVirtualHost \*:80**

* Add an entry in **/etc/hosts** file with your server ip and server name.

**Serverip 1st domain name**

**Serverip 2nd domain name**

* Restart and check your server with domain name, it will show the content.
* The errors logs and access logs are stored in a file specified by you in virtual host file.
* When a user made request, apache open a connection and user can make one request from that connection. After that request is completed, connection will be closed by apache. This would slow the performance of server. For this, we use **KeepAlive**.
* Keep Alive opens the connection for specific no of requests as we specified. After that it will close the connection. This would speed up the performance of server.
* Go to httpd.conf, add (or) edit

**Keep Alive on**

**MaxKeepAliveRequests 100 =** Max requests per connection.

**KeepAliveTimeout 15 =** Timeout b/w requests, if connection is idle for this specified time, connection will be closed.

**PORT BASED HOSTING**

* It is used to host multiple websites with different ports in a single server.
* For this type of virtual hosting, you need two different ports.
* First make sure that the ports you choose should be open and you have to make an entry in httpd.conf to allow it.
* Go to httpd.conf, add the port which you to allow for http.

**Listen 9000**

* Create a virtualhost for the port based with diffenent data.

**<VirtualHost \*:9000>**

**ServerName www.test101.com**

**ServerAlias test101.com**

**DocumentRoot /var/www/html/test101/**

**Errorlog "/var/www/html/test101/error.log"**

**Customlog "/var/www/html/test101/access.log" common**

**</VirtualHost>**

* Copy site data to html dir as mentioned in configuration file.
* Restart the server and check in browser with your ip. You will see your 1st site.
* Now check your apache server with your ip and port, you will see your 2nd site, which you have configured for the port based hosting.
* By default, you can’t access virtual hosts from remote machine, you have to add **dns entries** in **/etc/hosts** file in **remote machine**.
* After configuring apache, to use your local domains for a long term and to give access of your apache server to all end users over the internet, you need to buy a domain.

**APACHE MPM [Multi processing Modules]**

* Apache uses MPM for handling incoming requests and managing process.
* Apache provides three MPM…
* **Prefork MPM** = Prefork launches multiple child process. Each child process handles one connection at a time. It uses high memory of all other MPM.
* **Worker MPM** = worker launches multiple child process at a time. Each child processes creates many threads and each thread handles one connection at a time. It uses low memory compared to prefork.
* **Event MPM** = It works similar to worker. But, it allows requests to be served simultaneously by passing the processing requests to support threads, allowing main threads to take up new requests.
* By default, MPM will be installed while installing apache. You just need to start the server and check which MPM is running.
* Go to **/etc/httpd/conf.modules.d/00-mpm.conf** file, check for these lines.

**#LoadModule mpm\_prefork\_module modules/mod\_mpm\_prefork.so**

**#LoadModule mpm\_worker\_module modules/mod\_mpm\_worker.so**

**#LoadModule mpm\_event\_module modules/mod\_mpm\_event.so**

* Uncomment the line for which module you want to use. Save and restart the server.

**Httpd - V| grep MPM**...To find which module is running.

* **<IfModule worker.c>**

**ServerLimit 40**

**StartServers 5**

**MinSparethreads 5**

**MaxSpareThreads 10**

**MaxClients 150**

**Threadsperchild 25**

**MaxRequestsPerChild 3000**

**</IfModule>**

* **Server limit** = Maximum no of child process to run in total.
* **Start servers** = no of child process should start at apache server startup time.
* **Minimum spare threads** = min no of threads which are kept spare(idle).
* **Maximum spare threads** = max no of threads which are kept spare(idle).
* **Max clients** = max no of client connections.
* **Threads per child =** Max no of threads a child process can start (or) run.
* **Max req per child** = Max no of threads should be handled by a child process. Once the limit reached, the child process will die.
* Copy all these lines to **httpd.conf** file and restart the server.
* **Netstat –nt | grep :80** = To see no of concurrent connections.
* **Netstat –nt | grep :80 | wc –l** = To see how many concurrent connections (only number).
* **Netstat –an | grep :80** = To see active internet connections to apache server.
* **netstat -ntu | awk '{print $5}' | cut -d: -f1 | sort | uniq -c | sort –n** = To calculate, count no of connections currently established from each ip address to server.