HTTP

HTTP (Hypertext Transfer Protocol) server, or a *web server*, is a network service that serves content to a client over the web. This typically means web pages, but any other documents can be served as well. The web servers available in Red Hat Enterprise Linux are:

- Apache HTTP Server
- nginx

The Apache HTTP Server

Apache HTTP Server 2.2, a robust, full-featured open source web server developed by the Apache Software Foundation, that is included in Red Hat Enterprise Linux. It describes the basic configuration of the httpd service, and covers advanced topics such as adding server modules, setting up virtual hosts, or configuring the secure HTTP server.

Running the httpd Service

To be able to use the httpd service, make sure you have the httpd installed. You can do so by using the following command:

yum install httpd

Starting the Service

To run the httpd service, type the following at a shell prompt as root:

#systemctl start httpd

If you want the service to start automatically at the boot time, use the following command:

 # systemctl enable httpd

This will enable the service for runlevel 2, 3, 4, and 5.
If running the Apache HTTP Server as a secure server, a password may be required after the machine boots if using an encrypted private SSL key.

Stopping the Service

To stop the running httpd service, type the following at a shell prompt as root:

 # Systemctl stop httpd

To prevent the service from starting automatically at the boot time, type:

 # systemctl disable httpd

This will disable the service for all targets.

Restarting the Service

There are three different ways to restart a running httpd service:
To restart the service completely, enter the following command as root:

 # systemctl restart httpd

This stops the running httpd service and immediately starts it again. Use this command after installing or removing a dynamically loaded module such as PHP.
To only reload the configuration, as root, type:

 # systemctl reload httpd

This causes the running httpd service to reload its configuration file. Any requests being currently processed will be interrupted, which may cause a client browser to display an error message or render a partial page.
This causes the running httpd service to reload its configuration file. Any requests being currently processed will use the old configuration.

Verifying the Service Status

To verify that the httpd service is running, type the following at a shell prompt:

~]# systemctl status httpd

Editing the Configuration Files

When the httpd service is started, by default, it reads the configuration from locations that are listed below:

Path	Description
/etc/httpd/conf/httpd.conf	The main configuration file.
/etc/httpd/conf.d/	An auxiliary directory for configuration files that are included in the main configuration file.

Common httpd.conf Directives

The following directives are commonly used in the /etc/httpd/conf/httpd.conf configuration file:

<Directory>

The <Directory> directive allows you to apply certain directives to a particular directory only. It takes the following form:

```
<Directory directory>
directive
  …
</Directory>
```

The directory can be either a full path to an existing directory in the local file system, or a wildcard expression.

This directive can be used to configure additional cgi-bin directories for server-side scripts located outside the directory that is specified by ScriptAlias. In this case, the ExecCGI and AddHandler directives must be supplied, and the permissions on the target directory must be set correctly (that is, 0755).

Example

Using the <Directory> directive

```
<Directory /var/www/html>
  Options Indexes FollowSymLinks
  AllowOverride None
  Order allow,deny
  Allow from all
</Directory>
```

<VirtualHost>

The <VirtualHost> directive allows you apply certain directives to particular virtual hosts only. It takes the following form:

```
<VirtualHost address[:port]…>
directive
  …
</VirtualHost>
```

The address can be an IP address, a fully qualified domain name, or a special form as described in Table 18.2, “Available <VirtualHost> options”.

Available <VirtualHost> options

Option	Description
*	Represents all IP addresses.
default	Represents unmatched IP addresses.

Example Using the <VirtualHost> directive

```
<VirtualHost *:80>
  ServerAdmin webmaster@penguin.example.com
  DocumentRoot /www/docs/penguin.example.com
  ServerName penguin.example.com
  ErrorLog logs/penguin.example.com-error_log
  CustomLog logs/penguin.example.com-access_log common
</VirtualHost>
```

CustomLog

The CustomLog directive allows you to specify the log file name and the log file format. It takes the following form:

CustomLog path format

The path refers to a log file, and must be relative to the directory that is specified by the ServerRoot directive (that is, /etc/httpd/ by default). The format has to be either an explicit format string, or a format name that was previously defined using the LogFormat directive.

Example: Using the CustomLog directive

CustomLog logs/access_log combined

Deny

The Deny directive allows you to specify which clients are denied access to a given directory. It takes the following form:

Deny from client…

The client can be a domain name, an IP address (both full and partial), a network/netmask pair, or all for all clients.

Example . Using the Deny directive

Deny from 192.168.1.1

DirectoryIndex

The DirectoryIndex directive allows you to specify a document to be served to a client when a directory is requested (that is, when the URL ends with the / character). It takes the following form:

DirectoryIndex filename…

The filename is a name of the file to look for in the requested directory. By default, the server looks for index.html, and index.html.var.

Example Using the DirectoryIndex directive

DirectoryIndex index.html index.html.var

DocumentRoot

The DocumentRoot directive allows you to specify the main directory from which the content is served. It takes the following form:

DocumentRoot directory

The directory must be a full path to an existing directory in the local file system. The default option is /var/www/html/.

Example Using the DocumentRoot directive

DocumentRoot /var/www/html

ErrorDocument

The ErrorDocument directive allows you to specify a document or a message to be displayed as a response to a particular error. It takes the following form:
ErrorDocument error-code action
The error-code has to be a valid code such as 403 (Forbidden), 404 (Not Found), or 500 (Internal Server Error). The action can be either a URL (both local and external), or a message string enclosed in double quotes (that is, ").

Example Using the ErrorDocument directive
ErrorDocument 403 "Access Denied"
ErrorDocument 404 /404-not_found.html

ErrorLog

The ErrorLog directive allows you to specify a file to which the server errors are logged. It takes the following form:
ErrorLog path
The path refers to a log file, and can be either absolute, or relative to the directory that is specified by the ServerRoot directive (that is, /etc/httpd/ by default). The default option is logs/error_log

Example Using the ErrorLog directive
ErrorLog logs/error_log

Include

The Include directive allows you to include other configuration files. It takes the following form:
Include filename
The filename can be an absolute path, a path relative to the directory specified by the ServerRoot directive, or a wildcard expression. All configuration files from the /etc/httpd/conf.d/ directory are loaded by default.

ExampleUsing the Include directive
Include conf.d/*.conf

Listen

The Listen directive allows you to specify IP addresses or ports to listen to. It takes the following form:
Listen [ip-address:]port [protocol]
The ip-address is optional and unless supplied, the server will accept incoming requests on a given port from all IP addresses. Since the protocol is determined automatically from the port number, it can be usually omitted. The default option is to listen to port 80.
Note that if the server is configured to listen to a port under 1024, only superuser will be able to start the httpd service.

Example . Using the Listen directive
Listen 80

PidFile

The PidFile directive allows you to specify a file to which the process ID (PID) of the server is stored. It takes the following form:
PidFile path
The path refers to a pid file, and can be either absolute, or relative to the directory that is specified by the ServerRoot directive (that is, /etc/httpd/ by default). The default option is run/httpd.pid.

Example . Using the PidFile directive
PidFile run/httpd.pid

ServerAdmin

The ServerAdmin directive allows you to specify the email address of the server administrator to be displayed in server-generated web pages. It takes the following form:
ServerAdmin email
The default option is root@localhost.
This directive is commonly set to webmaster@hostname, where hostname is the address of the server. Once set, alias webmaster to the person responsible for the web server in /etc/aliases, and as superuser, run the newaliases command.

Example . Using the ServerAdmin directive
ServerAdmin webmaster@penguin.example.com

ServerName

The ServerName directive allows you to specify the host name and the port number of a web server. It takes the following form:
ServerName hostname[:port]
The hostname has to be a fully qualified domain name (FQDN) of the server. The port is optional, but when supplied, it has to match the number specified by the Listen directive.
When using this directive, make sure that the IP address and server name pair are included in the /etc/hosts file.

Example . Using the ServerName directive
ServerName penguin.example.com:80

ServerRoot

The ServerRoot directive allows you to specify the directory in which the server operates. It takes the following form:
ServerRoot directory
The directory must be a full path to an existing directory in the local file system. The default option is /etc/httpd/.

Example Using the ServerRoot directive
ServerRoot /etc/httpd

Common ssl.conf Directives

The Secure Sockets Layer (SSL) directives allow you to customize the behavior of the Apache HTTP Secure Server, and in most cases, they are configured appropriately during the installation. Be careful when changing these settings, as incorrect configuration can lead to security vulnerabilities.
The following directive is commonly used in /etc/httpd/conf.d/ssl.conf:

Setting Up an SSL Server

Secure Sockets Layer (SSL) is a cryptographic protocol that allows a server and a client to communicate securely. Along with its extended and improved version called Transport Layer Security (TLS), it ensures both privacy and data integrity. The Apache HTTP Server in combination with mod_ssl, a module that uses the OpenSSL toolkit to provide the SSL/TLS support, is commonly referred to as the SSL server
Unlike an HTTP connection that can be read and possibly modified by anybody who is able to intercept it, the use of SSL/TLS over HTTP, referred to as HTTPS, prevents any inspection or modification of the transmitted content. The below provides basic information on how to enable this module in the Apache HTTP Server configuration, and guides you through the process of generating private keys and self-signed certificates.

An Overview of Certificates and Security

Secure communication is based on the use of keys: a private key that is kept a secret, and a public key that is usually shared with the public. While the data encoded with the public key can only be decoded with the private key, data encoded with the private key can in turn only be decoded with the public key.

To provide secure communications using SSL, an SSL server must use a digital certificate signed by a Certificate Authority (CA). The certificate lists various attributes of the server (that is, the server host name, the name of the company, its location, etc.), and the signature produced using the CA's private key. This signature ensures that a particular certificate authority has signed the certificate, and that the certificate has not been modified in any way.

When a web browser establishes a new SSL connection, it checks the certificate provided by the web server. If the certificate does not have a signature from a trusted CA, or if the host name listed in the certificate does not match the host name used to establish the connection, it refuses to communicate with the server and usually presents a user with an appropriate error message.

By default, most web browsers are configured to trust a set of widely used certificate authorities. Because of this, an appropriate CA should be chosen when setting up a secure server, so that target users can trust the connection, otherwise they will be presented with an error message, and will have to accept the certificate manually. Since encouraging users to override certificate errors can allow an attacker to intercept the connection, you should use a trusted CA whenever possible.

When setting up an SSL server, you need to generate a certificate request and a private key, and then send the certificate request, proof of the company's identity, and payment to a certificate authority. Once the CA verifies the certificate request and your identity, it will send you a signed certificate you can use with your server. Alternatively, you can create a self-signed certificate that does not contain a CA signature, and thus should be used for testing purposes only.

Enabling the mod_ssl Module

If you intend to set up an SSL or HTTPS server using mod_ssl, you cannot have another application or module, such as mod_nss configured to use the same port. Port 443 is the default port for HTTPS.

To set up an SSL server using the mod_ssl module and the OpenSSL toolkit, install the mod_ssl and openssl packages. Enter the following command as root:

 # yum install mod_ssl openssl

This will create the mod_ssl configuration file at */etc/httpd/conf.d/ssl.conf*, which is included in the main Apache HTTP Server configuration file by default. For the module to be loaded, restart the httpd service.

Class Notes:-

HTTP (Hyper Text Transfer Protocol)

Apache Web server

http→port no :- 80
https (secure)→port no :- 443
yum install httpd -y
conf file :- /etc/httpd/conf/httpd.conf
Include conffile path
User apache
Group apache
ServerAdmin email/root@localhost -- to send mails if any issues
ServerName websitename(www.alc.org)

DocumentRoot "pathto root dir" (default /var/www/html)
<Directory "pathto root dir"/>
Options FollowSymLinks -- if any linkfiles allow access to that files
AllowOverride none
Order allow,deny
Allow from all
</Directory>
DirectoryIndex filename ---------deafult page to load when no other files are mentioned
NOTE:- One Website ---------- one port
--> To configure with new port number add entry as Listen portno

Creating virtual website

cd /etc/httpd/conf.d
vi server10.conf
<virtual virtualhostname:portno><VirutalHost *:portno>
DocumentRoot path/to/dir
servername virtualhostname
DirectoryIndex filename.html
</VirtualHost>
To test the syntax of httpd conf
#httpd -t

HTTPS
Secure, port no :- 443
encryption and de-cryption method

Private key -- To De-crypt the data
Public key -- To encrypt the data
key -- csr
key -- csr -- crt
SSL

1.
yum install mod_ssl
openssl
2. Generate keys
#openssl genrsa -out filename.key bit(1024,2048)
gendsa

#openssl genrsa -out

CSR
#openssl req -new -key pathtokeyfile.key -out ashok.csr

#openssl req -new -key server2.ashok.net.key -out server2.ashok.net.csr

CRT (CERTIFICATE)

openssl x509 -req -days 365 -signkey server2.key -in server2.csr -out server2.crt