Chapter 11. Web Servers

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. Web servers are also known as HTTP servers, as they use the *hypertext transport protocol* (HTTP).

11.1. The Apache HTTP Server

The web server available in Red Hat Enterprise Linux 7 is version 2.4 of the **Apache HTTP Server**, **httpd**, an open source web server developed by the Apache Software Foundation.

If you are upgrading from a previous release of Red Hat Enterprise Linux, you will need to update the **httpd** service configuration accordingly. This section reviews some of the newly added features, outlines important changes between Apache HTTP Server 2.4 and version 2.2, and guides you through the update of older configuration files.

11.1.1. Notable Changes

The Apache HTTP Server in Red Hat Enterprise Linux 7 has the following changes compared to Red Hat Enterprise Linux 6:

httpd Service Control

With the migration away from SysV init scripts, server administrators should switch to using the **apachectl** and **systemctl** commands to control the service, in place of the **service** command. The following examples are specific to the **httpd** service.

The command:

```
service httpd graceful
```

is replaced by

```
apachectl graceful
```

The **systemd** unit file for **httpd** has different behavior from the init script as follows:

- A graceful restart is used by default when the service is reloaded.
- A graceful stop is used by default when the service is stopped.

The command:

```
service httpd configtest
```

is replaced by

```
apachectl configtest
```

Private /tmp

To enhance system security, the **systemd** unit file runs the **httpd** daemon using a private **/tmp** directory, separate to the system **/tmp** directory.

Configuration Layout

Configuration files which load modules are now placed in the **/etc/httpd/conf.modules.d/** directory. Packages that provide additional loadable modules for **httpd**, such as *php*, will place a file in this directory. An **Include** directive before the main section of the **/etc/httpd/conf/httpd.conf** file is used to include files within the **/etc/httpd/conf.modules.d/** directory. This means any configuration files within **conf.modules.d/** are processed before the main body of **httpd.conf**. An **IncludeOptional** directive for files within the **/etc/httpd/conf.d/** directory is placed at the end of the **httpd.conf** file. This means the files within **/etc/httpd/conf.d/** are now processed after the main body of **httpd.conf**.

Some additional configuration files are provided by the *httpd* package itself:

- **/etc/httpd/conf.d/autoindex.conf** — This configures mod_autoindex directory indexing.
- **/etc/httpd/conf.d/userdir.conf** — This configures access to user directories, for example, **http://example.com/~username/**; such access is disabled by default for security reasons.
- **/etc/httpd/conf.d/welcome.conf** — As in previous releases, this configures the welcome page displayed for **http://localhost/** when no content is present.

Default Configuration

A minimal **httpd.conf** file is now provided by default. Many common configuration settings, such as **Timeout** or **KeepAlive** are no longer explicitly configured in the default configuration; hard-coded settings will be used instead, by default. The hard-coded default settings for all configuration directives are specified in the manual. See Section 11.1.13, “Installable Documentation” for more information.

Incompatible Syntax Changes

If migrating an existing configuration from **httpd 2.2** to **httpd 2.4**, a number of backwards-incompatible changes to the **httpd** configuration syntax were made which will require changes. See the following Apache document for more information on upgrading http://httpd.apache.org/docs/2.4/upgrading.html

Processing Model

In previous releases of Red Hat Enterprise Linux, different *multi-processing models* (MPM) were made available as different **httpd** binaries: the forked model, “prefork”, as **/usr/sbin/httpd**, and the thread-based model “worker” as **/usr/sbin/httpd.worker**.

In Red Hat Enterprise Linux 7, only a single **httpd** binary is used, and three MPMs are available as loadable modules: worker, prefork (default), and event. Edit the configuration file **/etc/httpd/conf.modules.d/00-mpm.conf** as required, by adding and removing the comment character # so that only one of the three MPM modules is loaded.

Packaging Changes

The LDAP authentication and authorization modules are now provided in a separate sub-package, *mod_ldap*. The new module **mod_session** and associated helper modules are provided in a new sub-package, *mod_session*. The new modules **mod_proxy_html** and **mod_xml2enc** are provided in a new sub-package, *mod_proxy_html*. These packages are all in the Optional channel.

Note

Before subscribing to the Optional and Supplementary channels see the Scope of Coverage Details. If you decide to install packages from these channels, follow the steps documented in the article called How to access Optional and Supplementary channels, and -devel packages using Red Hat Subscription Manager (RHSM)? on the Red Hat Customer Portal.

Packaging Filesystem Layout

The **/var/cache/mod_proxy/** directory is no longer provided; instead, the **/var/cache/httpd/** directory is packaged with a **proxy** and **ssl** subdirectory.

Packaged content provided with **httpd** has been moved from **/var/www/** to **/usr/share/httpd/**:

- **/usr/share/httpd/icons/** — The directory containing a set of icons used with directory indices, previously contained in **/var/www/icons/**, has moved to **/usr/share/httpd/icons/**. Available at **http://localhost/icons/** in the default configuration; the location and the availability of the icons is configurable in the **/etc/httpd/conf.d/autoindex.conf** file.
- **/usr/share/httpd/manual/** — The **/var/www/manual/** has moved to **/usr/share/httpd/manual/**. This directory, contained in the *httpd-manual* package, contains the HTML version of the manual for **httpd**. Available at **http://localhost/manual/** if the package is installed, the location and the availability of the manual is configurable in the **/etc/httpd/conf.d/manual.conf** file.
- **/usr/share/httpd/error/** — The **/var/www/error/** has moved to **/usr/share/httpd/error/**. Custom multi-language HTTP error pages. Not configured by default, the example configuration file is provided at **/usr/share/doc/httpd-*VERSION*/httpd-multilang-errordoc.conf**.

Authentication, Authorization and Access Control

The configuration directives used to control authentication, authorization and access control have changed significantly. Existing configuration files using the **Order**, **Deny** and **Allow** directives should be adapted to use the new **Require** syntax. See the following Apache document for more information http://httpd.apache.org/docs/2.4/howto/auth.html

suexec

To improve system security, the **suexec** binary is no longer installed as if by the **root** user; instead, it has file system capability bits set which allow a more restrictive set of permissions. In conjunction with this change, the **suexec** binary no longer uses the **/var/log/httpd/suexec.log** logfile. Instead, log messages are sent to **syslog**; by default these will appear in the **/var/log/secure** log file.

Module Interface

Third-party binary modules built against **httpd 2.2** are not compatible with **httpd 2.4** due to changes to the **httpd** module interface. Such modules will need to be adjusted as necessary for the **httpd 2.4** module interface, and then rebuilt. A detailed list of the API changes in version **2.4** is available here: http://httpd.apache.org/docs/2.4/developer/new_api_2_4.html.

The **apxs** binary used to build modules from source has moved from **/usr/sbin/apxs** to **/usr/bin/apxs**.

Removed modules

List of **httpd** modules removed in Red Hat Enterprise Linux 7:

mod_auth_mysql, mod_auth_pgsql

httpd 2.4 provides SQL database authentication support internally in the **mod_authn_dbd** module.

mod_perl

mod_perl is not officially supported with **httpd 2.4** by upstream.

mod_authz_ldap

httpd 2.4 provides LDAP support in sub-package *mod_ldap* using **mod_authnz_ldap**.

11.1.2. Updating the Configuration

To update the configuration files from the Apache HTTP Server version 2.2, take the following steps:

1. Make sure all module names are correct, since they may have changed. Adjust the **LoadModule** directive for each module that has been renamed.
2. Recompile all third party modules before attempting to load them. This typically means authentication and authorization modules.
3. If you use the **mod_userdir** module, make sure the **UserDir** directive indicating a directory name (typically **public_html**) is provided.
4. If you use the Apache HTTP Secure Server, see Section 11.1.8, “Enabling the mod_ssl Module” for important information on enabling the Secure Sockets Layer (SSL) protocol.

Note that you can check the configuration for possible errors by using the following command:

```
~]# apachectl configtest
Syntax OK
```

For more information on upgrading the Apache HTTP Server configuration from version 2.2 to 2.4, see http://httpd.apache.org/docs/2.4/upgrading.html.

11.1.3. Running the httpd Service

This section describes how to start, stop, restart, and check the current status of the Apache HTTP Server. 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
```

For more information on the concept of targets and how to manage system services in Red Hat Enterprise Linux in general, see Chapter 8, *Managing Services with systemd*.

11.1.3.1. Starting the Service

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

```
~]# systemctl start httpd.service
```

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

```
~]# systemctl enable httpd.service
ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-
user.target.wants/httpd.service'
```

Note

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.

11.1.3.2. Stopping the Service

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

```
~]# systemctl stop httpd.service
```

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

```
~]# systemctl disable httpd.service
rm '/etc/systemd/system/multi-user.target.wants/httpd.service'
```

11.1.3.3. Restarting the Service

There are three different ways to restart a running **httpd** service:

1. To restart the service completely, enter the following command as **root**:

   ```
   ~]# systemctl restart httpd.service
   ```

   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.

2. To only reload the configuration, as **root**, type:

   ```
   ~]# systemctl reload httpd.service
   ```

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

3. To reload the configuration without affecting active requests, enter the following command as **root**:

   ```
   ~]# apachectl graceful
   ```

This causes the running **httpd** service to reload its configuration file. Any requests currently being processed will continue to use the old configuration.

For more information on how to manage system services in Red Hat Enterprise Linux 7, see Chapter 8, *Managing Services with systemd*.

11.1.3.4. Verifying the Service Status

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

```
~]# systemctl is-active httpd.service
active
```

11.1.4. Editing the Configuration Files

When the **httpd** service is started, by default, it reads the configuration from locations that are listed in Table 11.1, "The httpd service configuration files".

Table 11.1. The httpd service configuration files

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.

Although the default configuration should be suitable for most situations, it is a good idea to become at least familiar with some of the more important configuration options. Note that for any changes to take effect, the web server has to be restarted first. See Section 11.1.3.3, "Restarting the Service" for more information on how to restart the **httpd** service.

To check the configuration for possible errors, type the following at a shell prompt:

```
~]# apachectl configtest
Syntax OK
```

To make the recovery from mistakes easier, it is recommended that you make a copy of the original file before editing it.

11.1.5. Working with Modules

Being a modular application, the **httpd** service is distributed along with a number of *Dynamic Shared Objects* (DSOs), which can be dynamically loaded or unloaded at runtime as necessary. On Red Hat Enterprise Linux 7, these modules are located in **/usr/lib64/httpd/modules/**.

11.1.5.1. Loading a Module

To load a particular DSO module, use the **LoadModule** directive. Note that modules provided by a separate package often have their own configuration file in the **/etc/httpd/conf.d/** directory.

Example 11.1. Loading the mod_ssl DSO

```
LoadModule ssl_module modules/mod_ssl.so
```

Once you are finished, restart the web server to reload the configuration. See Section 11.1.3.3, "Restarting the Service" for more information on how to restart the **httpd** service.

11.1.5.2. Writing a Module

If you intend to create a new DSO module, make sure you have the *httpd-devel* package installed. To do so, enter the following command as **root**:

```
~]# yum install httpd-devel
```

This package contains the include files, the header files, and the **APache eXtenSion** (**apxs**) utility required to compile a module.

Once written, you can build the module with the following command:

```
~]# apxs -i -a -c module_name.c
```

If the build was successful, you should be able to load the module the same way as any other module that is distributed with the Apache HTTP Server.

11.1.6. Setting Up Virtual Hosts

The Apache HTTP Server's built in virtual hosting allows the server to provide different information based on which IP address, host name, or port is being requested.

To create a name-based virtual host, copy the example configuration file **/usr/share/doc/httpd-*VERSION*/httpd-vhosts.conf** into the **/etc/httpd/conf.d/** directory, and replace the **@@Port@@** and **@@ServerRoot@@** placeholder values. Customize the options according to your requirements as shown in Example 11.2, "Example virtual host configuration".

Example 11.2. Example virtual host configuration

```
<VirtualHost *:80>
    ServerAdmin webmaster@penguin.example.com
    DocumentRoot "/www/docs/penguin.example.com"
    ServerName penguin.example.com
    ServerAlias www.penguin.example.com
    ErrorLog "/var/log/httpd/dummy-host.example.com-error_log"
    CustomLog "/var/log/httpd/dummy-host.example.com-access_log" common
</VirtualHost>
```

Note that **ServerName** must be a valid DNS name assigned to the machine. The **<VirtualHost>** container is highly customizable, and accepts most of the directives available within the main server configuration. Directives that are *not* supported within this container include **User** and **Group**, which were replaced by **SuexecUserGroup**.

Note

If you configure a virtual host to listen on a non-default port, make sure you update the **Listen** directive in the global settings section of the **/etc/httpd/conf/httpd.conf** file accordingly.

To activate a newly created virtual host, the web server has to be restarted first. See Section 11.1.3.3, "Restarting the Service" for more information on how to restart the **httpd** service.

11.1.7. 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*. Red Hat Enterprise Linux also supports the use of Mozilla NSS as the TLS implementation. Support for Mozilla NSS is provided by the **mod_nss** module.

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. This section 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.

11.1.7.1. An Overview of Certificates and Security

Secure communication is based on the use of keys. In conventional or *symmetric cryptography*, both ends of the transaction have the same key they can use to decode each other's transmissions. On the other hand, in public or *asymmetric cryptography*, two keys co-exist: 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. For more information on this, see Table 11.2, "Information about CA lists used by common web browsers".

Table 11.2. Information about CA lists used by common web browsers

Once generated, add the key and certificate locations to the **/etc/httpd/conf.d/ssl.conf** configuration file:

```
SSLCertificateFile /etc/pki/tls/certs/hostname.crt
SSLCertificateKeyFile /etc/pki/tls/private/hostname.key
```

Finally, restart the **httpd** service as described in Section 11.1.3.3, "Restarting the Service", so that the updated configuration is loaded.

11.1.12. Configure the Firewall for HTTP and HTTPS Using the Command Line

Red Hat Enterprise Linux does not allow **HTTP** and **HTTPS** traffic by default. To enable the system to act as a web server, make use of **firewalld**'s supported services to enable **HTTP** and **HTTPS** traffic to pass through the firewall as required.

To enable **HTTP** using the command line, issue the following command as **root**:

```
~]# firewall-cmd --add-service http
 success
```

To enable **HTTPS** using the command line, issue the following command as **root**:

```
~]# firewall-cmd --add-service https
 success
```

Note that these changes will not persist after the next system start. To make permanent changes to the firewall, repeat the commands adding the **--permanent** option.

11.1.12.1. Checking Network Access for Incoming HTTPS and HTTPS Using the Command Line

To check what services the firewall is configured to allow, using the command line, issue the following command as **root**:

```
~]# firewall-cmd --list-all
public (default, active)
  interfaces: em1
  sources:
  services: dhcpv6-client ssh
output truncated
```

In this example taken from a default installation, the firewall is enabled but **HTTP** and **HTTPS** have not been allowed to pass through.

Once the **HTTP** and **HTTP** firewall services are enabled, the **services** line will appear similar to the following:

```
services: dhcpv6-client http https ssh
```

For more information on enabling firewall services, or opening and closing ports with **firewalld**, see the Red Hat Enterprise Linux 7 Security Guide.

11.1.13. Additional Resources

To learn more about the Apache HTTP Server, see the following resources.

Installed Documentation

- **httpd(8)** — The manual page for the **httpd** service containing the complete list of its command-line options.
- **genkey(1)** — The manual page for **genkey** utility, provided by the *crypto-utils* package.
- **apachectl(8)** — The manual page for the Apache HTTP Server Control Interface.

Installable Documentation

- http://localhost/manual/ — The official documentation for the Apache HTTP Server with the full description of its directives and available modules. Note that in order to access this documentation, you must have the *httpd-manual* package installed, and the web server must be running.

  Before accessing the documentation, issue the following commands as **root**:

```
~]# yum install httpd-manual
~]# apachectl graceful
```

Online Documentation

- http://httpd.apache.org/ — The official website for the Apache HTTP Server with documentation on all the directives and default modules.
- http://www.openssl.org/ — The OpenSSL home page containing further documentation, frequently asked questions, links to the mailing lists, and other useful resources.