Contents

Categories

Virtualization

How-To Overview Introduction

Getting started with Apache HTTP Server

Jan Kuparinen – Version F34 onwards – Last review: 2020-11-15 | Needs Review!

Contents

Installing HTTPD
Securing Apache HTTPD
Using mod_ssl
Installing webapps
Configuring Apache HTTPD
Enabling access to web applications
Opening firewall ports
Disabling Test Page

> The Apache HTTP Server is one of the most commonly-used web servers. This section acts as a quick-start guide to deploying and configuring Apache on Fedora.

Installing HTTPD

This procedure describes the steps to install Apache **HTTPD** on Fedora.

1. Install **HTTPD** packages.

2. Start the **HTTPD** service.

```
sudo systemctl start httpd.service
```

> **Note**
>
> To enable auto start of **HTTPD** service at boot, execute the following command:
>
> ```
> sudo systemctl enable httpd.service
> ```

Navigate to http://localhost to access the Apache test page. You may not be able to access the server from any other host. To access the server from other hosts, see Opening firewall ports.

Securing Apache HTTPD

To enable TLS/SSL support, download and install one of the following packages:

- mod_ssl, based on OpenSSL
- mod_gnutls, based on GnuTLS
- mod_nss, based on NSS

Using mod_ssl

Installing mod_ssl

The mod_ssl package will be automatically enabled post installation. Install the mod_ssl package using the following command:

```
sudo dnf install mod_ssl -y
```

Generating a new certificate

To generate a new certificate, refer to Create a certificate using OpenSSL.

If you already have a certificate generated on another computer, do the following:

1. Move the certificate and the key file to the correct folder

```
sudo mv key_file.key /etc/pki/tls/private/myhost.com.key
sudo mv certificate.crt /etc/pki/tls/certs/myhost.com.crt
```

2. Ensure that the following parameters are correct:
   a. SELinux contexts

```
restorecon /etc/pki/tls/private/myhost.com.key
restorecon /etc/pki/tls/certs/myhost.com.crt
```

   b. Ownership

```
sudo chown root.root /etc/pki/tls/private/myhost.com.key
sudo chown root.root /etc/pki/tls/certs/myhost.com.crt
```

   c. Permissions

```
sudo chmod 0600 /etc/pki/tls/private/myhost.com.key
sudo chmod 0600 /etc/pki/tls/certs/myhost.com.crt
```

After installing the existing certificate, set up the certificate using mod_ssl configuration.

mod_ssl configuration

The default TLS/SSL configuration is contained in the file `/etc/httpd/conf.d/ssl.conf`. In the `ssl.conf` file, following are the directives that specify where the TLS/SSL certificate and key are located:

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

These directives are enclosed in a block defining a virtual host:

```
SSLCertificateFile /etc/pki/tls/certs/localhost.crt
...
SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
...
</VirtualHost>
```

To define a different location for these files, do the following:

1. Create a copy of the `/etc/httpd/conf.d/ssl.conf` file and renew the file to `z-ssl-local.conf` .
2. Edit the following lines in the `z-ssl-local.conf` file:

```
<VirtualHost _default_:443>
SSLCertificateFile /etc/pki/tls/certs/www.myhost.org.crt
SSLCertificateKeyFile /etc/pki/tls/private/www.myhost.org.key
</VirtualHost>
```

This file will override the two settings for the `_default_:443` virtual host; all other settings from `ssl.conf` will be retained.

Settings for individual virtual hosts

To use SSL/TLS for a specific virtual host with a different certificate as default, do the following:

1. Open that virtual host's configuration file `/etc/httpd/conf.d/hostname.conf` .
2. Insert these lines between `<VirtualHost hostname:port>` and `</VirtualHost>` :

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

Installing webapps

of web applications is recommended. These packages will be configured following the distribution's best practices which help to ensure the security of the installation.

For instance, by installing static files to locations the web server does not have the ability to write to, and doing access control with configuration files rather than `.htaccess` files, which are slightly more vulnerable to attack.

Packaged web applications will also be configured to work with SELinux, which provides significant security benefits.

You will also receive updates through the usual Fedora update process, making it easier to keep your installation up to date.

They will also often have the default configuration tweaked according to Fedora's conventions, meaning you have to do less work to get the application up and running.

Most web applications are simply packaged according to their name. For instance, you can install Wordpress by executing the following command:

```
sudo dnf install wordpress
```

Packaged web applications will usually provide Fedora-specific instructions in a

`/usr/share/doc/wordpress/README.fedora` and `/usr/share/doc/wordpress/README.fedora-multiuser`.

Packaged web applications usually restrict access by default so you can access them only from the server host itself, to ensure you can run all initial configuration safely and things like administration interfaces are not left accessible to the public. For information on how to broaden access, see Enabling access to web applications.

Web applications commonly require the use of a database server. This Quick Docs article provides information on installing and configuring PostgreSQL and this wiki page about MariaDB on Fedora.

`/etc/httpd/conf/httpd.conf` is the main Apache configuration file. Custom confirguration files are specified under `/etc/httpd/conf.d/*.conf`. If the same settings are specified in both `/etc/httpd/conf/httpd.conf` and a `.conf` file in `/etc/httpd/conf.d/`, the setting from the `/etc/httpd/conf.d/` file will be used.

Files in `/etc/httpd/conf.d/` are read in alphabetical order: a setting from `/etc/httpd/conf.d/z-foo.conf` will be used over a setting from `/etc/httpd/conf.d/foo.conf`. Similarly, a setting from `/etc/httpd/conf.d/99-foo.conf`, will be used over a setting from `/etc/httpd/conf.d/00-foo.conf`.

As a best practice, do not modify `/etc/httpd/conf/httpd.conf` or any of the `/etc/httpd/conf.d` files shipped by Fedora packages directly. If you make any local changes to these files, then any changes to them in newer package versions will not be directly applied. Instead, a `.rpmnew` file will be created, and you will have to merge the changes manually.

It is recommended to create a new file in `/etc/httpd/conf.d/` which will take precedence over the file you wish to modify, and edit the required settings. For instance, to change a setting specified in `/etc/httpd/conf.d/foo.conf` you could create the file `/etc/httpd/conf.d/z-foo-local.conf`, and place your setting in that file.

Note

After making any changes to your server configuration, execute the following command:

```
sudo systemctl reload httpd.service
```

Certain changes may require Apache to be fully restarted. To fully restart Apache, execute the following command:

```
sudo systemctl restart httpd.service
```

By default Fedora-packaged web applications are usually configured such that, access is allowed only from the localhost. This is defined by the file `/etc/httpd/conf.d/webapp.conf` which contains the following settings:

```
<Directory /usr/share/webapp>
    <IfModule mod_authz_core.c>
        # Apache 2.4
        Require local
    </IfModule>
    <IfModule !mod_authz_core.c>
        # Apache 2.2
        Order Deny,Allow
        Deny from all
        Allow from 127.0.0.1
        Allow from ::1
    </IfModule>
</Directory>
```

Before allowing general access to the webapp, ensure to do the following:

- ✓ Webapp has been configured correctly
- ✓ Administration interface and other sensitive areas are not accessible without appropriate authentication
- ✓ Database configuration is secure, if the application uses a database

To broaden access to the application, create a file `/etc/httpd/conf.d/z-we-bapp-allow.conf` . To allow access to all systems on a typical local network, add the following lines into the file:

```
<Directory /usr/share/webapp>
    <IfModule mod_authz_core.c>
        # Apache 2.4
        Require local
        Require ip 192.168.1
    </IfModule>
    <IfModule !mod_authz_core.c>
        # Apache 2.2
        Order Deny,Allow
        Deny from all
```

```
        Allow from 192.168.1.1
    </IfModule>
</Directory>
```

Once the application is correctly configured, add the following configuration to allow access from any host:

```
<Directory /usr/share/webapp>
    <IfModule mod_authz_core.c>
        # Apache 2.4
        Require all granted
    </IfModule>
    <IfModule !mod_authz_core.c>
        # Apache 2.2
        Order Deny,Allow
        Allow from all
    </IfModule>
</Directory>
```

Opening firewall ports

> **Important**
>
> This exposes your computer to the Internet and potential attackers. Secure your system and your Apache installation properly before exposing your server to the Internet.

Apache uses port 80 for plain http connections and port 443 for TLS/SSL connections by default. To make this service available from other computers or the Internet, allow Apache through the firewall using any one the following commands:

To allow Apache through the firewall at each boot:

- For plain HTTP connections:

```
sudo firewall-cmd --permanent --add-service=http
```

```
sudo firewall-cmd --permanent --add-service=https
```

To allow Apache through the firewall instantly:

- For plain HTTP connections:

```
sudo firewall-cmd --add-service=http
```

- For TLS/SSL connections:

```
sudo firewall-cmd --add-service=https
```

> **Note**
>
> If your server is running in a network with a NAT router, you will also need to configure your router to forward the HTTP and HTTPS ports to your server, if you wish to allow access from outside your local network.

Disabling Test Page

To disable the test page, comment out all the lines in the file `/etc/httpd/conf.d/welcome.conf` using `#` as follows:

```
# <LocationMatch "^/+$">
#    Options -Indexes
#    ErrorDocument 403 /.noindex.html
# </LocationMatch>

# <Directory /usr/share/httpd/noindex>
#    AllowOverride None
#    Require all granted
# </Directory>

# Alias /.noindex.html /usr/share/httpd/noindex/index.html
```

Additional resources

- Apache TLS/SSL documentation
- Apache security tips
- OwnCloud

Want to help? Learn how to contribute to Fedora Docs ›

Last build: 2023-09-12 14:50:21 UTC | Last content update: 2023-08-28