Basic Apache Web Services

Getting Apache

Before you can do anything else, you need to install Apache. You can download the rpms from some of your regular sites, or from http://download.fedora.redhat.com/pub/

Fedora Downloads You can also download the Apache source from: **Apache Downloads**

Download the archive from a mirror and expand it using the *tar* command with the *gzip*

option. This will create a directory named **http-...**. cd to that directory and perform the Linux-install-3-step.

1. ./configure
2. make
3. make install

The problem with this method is that things can go in locations that don't match with the locations assumed by other Fedora packages. This isn't likely to be a big problem, and you can always install other packages from source, so it is your choice.

Running Apache

Apache services are provided by a daemon. This would normally be accomplished via a startup file in /etc/rc.d.init.d and the appropriate links in the rc.x directories. This is usually done automatically by the install process or the rpm's, so you don't have to do anything other than use **chkconfig** to specify the desired runlevels and/or **service** to start the server.

Apache comes with another executable named **apachectl** which can be used to manage the daemon. The syntax is *apachectl command* where *command* can be the following:

start	Start the Apache daemon.
stop	Stops the Apache daemon.
restart	Restarts the Apache daemon by sending it a SIGHUP.
fullstatus	Displays a full status report.
status	Displays a brief status report.
graceful	Gracefully restarts the Apache daemon by sending it a SIGUSR1. If the daemon is not ru
configtest	Run a configuration file syntax test.

This is considered to be a better solution than the services command because this script is Apache-aware and handles errors and signals in a more appropriate manner.

Configuring Apache

The normal Red Hat/Fedora location of the configuration files for Apache is /etc/httpd. This directory contains a subdirectory named **conf** that has the specific configuration files, a directory of configuration files for extensions (**conf.d**) and there are also links to other directories that contains logs, the run information and modules. For a simple system, the default links are probably acceptable and you are interested only in the files in the **conf** directory.

The **conf** directory contains several files, but the only one of immediate consequence is **httpd.conf**. This file contains one directive per line, unless a backslash ( ) is the last character on the line to specify a continuation.

Directive syntax is: *Directive* lt *value* gt , where value can can be a string, a number, a URL, a pathname or just about anything else. Each directive has its own set of possible values. As you can see, the list of **Directives** is substantial and we will consider only a small portion. The default file is acceptable for most situations and the changes needed require a limited set of directives.

The directives can be divided into the following categories:

Directive Class	Usage
General Configuration	A hodge-podge of things that don't fit anywhere else and provide features that set the overall behaviour of the server.
Block	Limit the application of the enclosed directives to a specific set of entities.
Logging	Control the way in which the server logs.
Performance Tuning	Determine the parameters the affect the perceived performance of the server.
File Typing	Set the ways in which the server handle file content.
Mapping	Set the procedures to be used in mapping addresses and applications.
Scripting	Set parameters that control the access to CGI scripts and their handling, particularly for debugging.
Directory Indexing	Determine the handling of the display of directory contents.
Access Control	Control the access that the browser has to local resources.

An Example httpd.conf File

Discussing the directives in categorical fashion is less than informative until you know something about their use. First, look at an **Example httpd.conf**.

As indicated in the header comments in this file, it is divided into three parts: Global environment for the server, definition of the default server (non-virtual hosts), and definition for virtual hosts.

Global Environment

Taking the directives from the example file in order, we see the following settings:

The first few directives set some basic functionality for the server:

ServerTokens OS
: Sets the type of information that the server will return if asked by a client. The choices are *min* for only a name and version number, *OS* to include the operating system name and *full* to include information about modules.

ServerRoot ”/etc/httpd”
: The location of the server root directory where the configuration, error and log files are kept.

PidFile run/httpd.pid
: The file where the server pid is stored when it starts.

The next few control the handling of individual connections to the server. This includes the length of time the connection is allowed to be idle and the resource balance between opening new connections and keeping connections open.

Timeout 300
: The timeout period when receives and sends timeout. For example, once a transfer had started, the maximum time in seconds that can expire for the transfer to complete.

KeepAlive Off
: Whether persistent connections are allowed. This can be used to improve performance in responding to repeated requests, but at the expense of higher overhead. The possible values are *Off* and *On*.

MaxKeepAliveRequests 100
: The maximum number of requests to allow on a connection before closing it. Larger numbers improve performance on a connection, but at the expense of higher memory needs.

KeepAliveTimeout 15
: The time a connection can be idle before it is closed.

The next few control the number of child servers that are allowed to run and a limit on the amount of traffic they can handle. One consideration here is that some pinhead could decide to open a connection and then ask for pages as fast as possible to disrupt service. As always, the balance is between response time to the individual user and the overall response time to all users.

The directives are enclosed in a block directive

< IfModule prefork.c >

.

.
< /IfModule >
This causes the included directives to be loaded only if the named module has been included by compiling it in or by dynamic loading. prefork.c may not be the server MPM core module you are using but for a default installation, it is likely. You can find out by executing:

apachectl -l To get a list of installed modules. See the next section for more on this.

StartServers 8
: The number of servers to start when the daemon begins.

MinSpareServers 5
: The minimum number of spare servers. If the number of idle servers falls below this number, the server creates additional servers.

MaxSpareServers 20
: The maximum number of spare servers. If the number of idle servers is above this number, the server destroys some.

MaxClients 150
: The maximum number of simultaneous clients allowed. The limit is 256.

MaxRequestsPerChild 1000
: The maximum number of requests allowed on a child process before it is closed.

The next set of directives are the same, but they are inside of a different Interface Module block: < IfModule worker.c >

The *worker* module implements a multi-process, multi-threaded server which is more efficient than the prefork server and would be used if your system has the resources for it.

The next set of directives control a third multiprocessing module called the *perchild* module. This is a non-functional module at this point and can be ignored.

The next two directives are fundamental server properties:

Listen 80
: The port to listen on. This is normally 80 but could be changed for a variety of reasons. You can also specify an IP address if you have a multi-homed host.

Include conf.d/*.conf
: The configuration files to be loaded. This is relative to the Document Root defined earlier.

The next section loads the desired modules into the server. There is a lengthy list, where each has the syntax: LoadModule pathname_of_module The pathname is relative to the

Document Root and they are always shared library files (ending in the *.so* suffix). These modules are loaded into the server and constitute a major part of the capabilities.

This section is followed by a couple of module loads that are based on the type of multi-processing being used. In other words, certain modules are loaded depending on other modules that have been loaded.

The next directive sets the behavior of the server when a full status request is made by apachectl. The default is to leave this off, but if you are troubleshooting, you might want to turn it on.

ExtendedStatus On
: Report all information for a full status report.

Main Server Configuration

These directives configure the main server which handles all requests not handled by a virtual server.

The first set of directives define the running parameters and identification for the server.

User apache
: The server runtime user.

Group apache
: The server runtime group.

ServerAdmin root@localhost
: Email address for reporting server problems.

ServerName localhost
: The DNS name of the host the server is running on. The default is localhost with the 127.0.0.1 address if this is left undefined.

UseCanonicalName Off
: Determines how the server creates self-referencing URLs and variables. If Off, it uses the hostname and port supplied, otherwise it uses the ServerName directive.

DocumentRoot ”/var/www/html”
: The directory which is the root for the served documents tree.

Apache allows for directory-by-directory access control and this requires substantial configuration. This is where you are most likely to have problems in a basic configuration.

The first set of directives are inside of a *Directory Block*.

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
```

The Directory Block has a name which specifies the directory name relative to the Document Root. The directives inside apply to the given directory and all directories below it in the tree. In this case, the name is the root name, so the directives inside apply to the root directory and all directories below it. Both of these directives have complex capabilities that will be discussed later.

Options FollowSymLinks
: *Options* indicates that what follows is an option statement, and *FollowSymLinks* specifies that the server is to follow symbolic links if they exist.

AllowOverride None
: AllowOverride specifies how *.htaccess* files; *None* means that the .htaccess files are ignored. *All* means that all default directives can be overridden in a .htaccess file.

The next set of directives defines the access properties of the document root.

```
<Directory /var/www/docroot>
    .
    .
</Directory>
```

Where */var/www/docroot* would be your document root.

Options Indexes FollowSymLinks
: Indexes allows the server to display a list of files if there is no index.html file (or equivalent) in the directory. FollowSymLinks it the same as it was before.

Allow Override None
: Do not allow .htaccess files to override default access control.

Order deny,allow
: When evaluating access control, first apply the deny properties and then the allow properties.

Allow From All
: Allow access from any host.

The next set of directives creates a special arrangement for the root directory. It disables autoindex (because it would allow outsiders to collect valuable information about your Document Root directory) and to provide a default error page if there is no index.html page.

```
<LocationMatch "^/$">
    .
    .
</LocationMatch>
```

LocationMatch identies a block that is to be applied if a request for a page matches. In this instance, the match is to the regular expression

```
^/$
```

which means a name that contains only a beginning, the Document Root name and and end.

Options -Indexes
: Turn off the ability to see a list of files.

ErrorDocument 403 /error/noindex.html
: ErrorDocument indicates what should happen if there is an error. It is followed by the error code to return (403 means *not found*) and then a URL to display.

The next section defines the properties of the *User Directory*. The Apache server allows for a user directory type to be defined, which includes the name type, location and default access properties. By default, the UserDir capability is disabled to prevent outsiders from using it to identify the presence of usernames on the system, but most systems want it enabled. The userdir service is a module, so this set of directives is included in an IfModule block:

```
<IfModule mod_userdir.c>
    .
    .
</IfModule>
```

UserDir disable
: Present only to disable UserDir functionality.

UserDir public_html
: Enable the use of / username/ naming for user directories.

Access to the user directories is controlled by a block of the form:

```
<Directory /home/*/www>
   .
   .
</Directory>
```

This specifies that user directories are of the form */home/username/www*. The directives applied are:

AllowOverride FileInfo AuthConfig Limit
: This specifies that users can override the default access controls with .htaccess files for document types (FileInfo), authorization directives (AuthConfig) and host access (Limit).

Options Multiviews Indexes SymLinksIfOwnerMatch IncludesNoExec
: This specifies that multiple views of content are OK (MultiView), directory listings are allowed (Indexes), symbolic links can be followed only if the owner of the linked file is the same (SymLinksIfOwnerMatch) and the #exec commands are disabled but server-side includes are allowed (IncludesNoExec).

If access control is allowed, you need to describe the access control, which is accomplished with a *Limit* block and a *LimitExcept* block. The difference is that a Limit block limits access for specific HTTP operations, while a LimitExcept block allows access to only specific methods. Therefore, LimitExcept blocks are more restrictive and preferred over Limit blocks.

The following block limits the *GET*, *POST* and *OPTIONS* options by specifying that access is allowed only from a single domain. The problem with the Limit block is that it specifies the restrictions on certain operations, but things not mentioned are allowed.

```
<Limit GET POST OPTIONS>
   Order allow, deny
   Allow from all
   Deny from ! 153.90.192.0/21
</Directory>
```

The following provides the same protection, but it denies everything by default, and so is not susceptible to new, unrecognized or unstated methods. Typically, LimitExcept should be used unless there is some overriding reason.

```
<LimitExcept GET POST OPTIONS>
   Order deny, allow
   Deny from all
   Allow from 153.90.192.0/21
</Directory>
```

The remainder of the file has a number of simple directives:

DirectoryIndex index.html index.html.var homepage.html homepage.htm
: The default file to serve if a directory is requested.

AccessFileName .htaccess
: Default access control file name.

This file needs some control to insure that outsiders can't see the content:

```
<Files ~ "^\.ht">
   Order allow,deny
   Deny from all
</Files>
```

TypesConfig /etc/mime.types
: The location of the MIME types file.

DefaultType text/plain
: The default MIME type to use in serving pages.

MIME typing uses information from the files themselves to determine type. The MIME magic file location is given by:

```
<IfModule mod_mime_magic.c>
   MIMEMagicFile conf/magic
</IfModule>
```

HostnameLookups Off
: When logging server activity, turn of the hostname lookup so only IP addresses are reported.

Logging Error logging is controlled by two directives

ErrorLogs logs/error_log
: Error logging is done under the configuration directory in the logs directory in the file error_log.

logLevel warn
: All events of magnitude greater than or equal to this level are reported. The levels are identical to those of syslog.

Access logging is very handy to have and it is controlled by:

CustomLog logs/access_log combined
: Access logging is done under the configuration directory in the logs directory in the file access_log.

The *combined* name is defined in the file like this:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
```

You can look this up in the documentation, but it specifies the fields to be output for any entry in the log file.

ServerSignature On
: Cause error pages, ftp directory listings and other displays to have a server signature.

IndexingOptions controls the way indexes are displayed.

IndexOptions FancyIndexing FoldersFirst VersionSort NameWidth=*
: FancyIndexing gives a nicer display, FoldersFirst produces the folders before the files, VersionSort specifies how versions should be compared and NameWidth=* allows the filenames to be as long as necessary (an integer value can be used). and other displays to have a server signature.

Redirection One of the features of Apache that is indispensible is *redirection* or *aliasing*, which allows you to specify a mapping for a directory name. For example, suppose you have directory containing icons for use by people building web pages on your system at /usr/local/www/icons/. But this is somewhat inconvenient to use, and you might change it to another location. You could solve these problems with:

```
Alias /icons/ /usr/local/www/icons/

<Directory "/usr/local/www/icons">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
/Directory>
```

A reference to /icons/ is redirected to /usr/local/www/icons/, so access can be gained with a simple */icon/* pathname rather than the whole string, and the access controls specified. Also, the target directory can be outside of the directory tree.
Another choice is the *AliasMatch* directive

```
AliasMatch regular-expression pathname
```

which is equivalent to Alias, but makes use of standard regular expressions instead of simple prefix matching. The supplied regular expression is matched against the URL-path, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a filename. For example, to activate the /icons directory, one might use:

```
AliasMatch ^/icons(.*) /usr/local/apache/icons/$1
```

maps any reference with "/icons" at the beginning to the given directory with the parenthesized expression substituted for

`$1`

. For example, /icons/doodlebug.gif maps to /usr/local/apache/icons/doodlebug.gif.

The *ScriptAlias* directive gives a location for cgi-bin scripts. Having a directory that is given special permissions may be important to avoid security issues. ScriptAlias operation is identical to the Alias directive. For example:

```
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
```

The ScriptAliasMatch is identical to AliasMatch, except that it sets a mapping for CGI-bin scripts.

```
ScriptAliasMatch ^/cgi-bin(.*) /usr/local/apache/cgi-bin$1
```

To avoid having users running cgi-bin programs that are unprotected, you might do this:

```
ScriptAliasMatch /home/(.*)/cgi-bin(.*) /usr/local/apache/$1/cgi-bin/$2
```

Icons and Descriptions Apache uses icons to represent files and filename extensions and it allows descriptions to be placed in file listings. Those icons are managed by the following directives:

AddIconByEncoding (CMP, /icons/compressed.gif) x-compress x-gzip
: Specifies that x-compress and x-gzip encoded files should be represented by /icons/compressed.gif and CMP is the text representation if the icon is unavailable.

AddIconByType (TXT, /icons/text.gif)
: Specifies that files of type gif should be represented by /icons/text.gif and TXT is the text representation if the icon is unavailable.

DefaultIcon /icons/unknown.gif
: The default icon for files which do not have an icon set.

AddDescription ”GZIP compressed document” .gz
The given message is displayed for files ending with the ”.gz” suffix.
Blah Blah
/¿ The given message is displayed for files ending with the ”.gz” suffix.
and a whole bunch of other stuff.
Blah Blah
/¿ The given message is displayed for files ending with the ”.gz” suffix.
and a whole bunch of other stuff.

MIME Types

Error Responses You can configure the server error responses by creating your own versions of the error pages. The normal setup looks like this:

```
   Alias /error/ "/var/www/error/"

<IfModule mod_negotiation.c>
<IfModule mod_include.c>
    <Directory "/var/www/error">
        AllowOverride None
        Options IncludesNoExec
        AddOutputFilter Includes html
        AddHandler type-map var
        Order allow,deny
        Allow from all
        LanguagePriority en es de fr
        ForceLanguagePriority Prefer Fallback
    </Directory>
    ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
    ErrorDocument 401 /error/HTTP_UNAUTHORIZED.html.var
    ErrorDocument 403 /error/HTTP_FORBIDDEN.html.var
      .
   </IfModule>
   </IfModule>
```

It is a nice exercise to try to customize one or more error pages. In order to recognize the error message codes, look at the documentation for HTTP.

Languages and Character Sets You can specify a languages and character sets for Apache, but there probably isn't much need typically. The pertinent directives are:

AddLanguage da .dk
: Use the MIME language da if a file has an extension of .dk.

LanguagePriority en da nl et fr
: Establish the desired order of languages to use in the case of language negotiation.

AddDefaultCharset ISO-8859-1
: Set the default character set.

File Mapping There are a number of special file handling directives.

ReadmeName README.html
: Specifies of a special file to be appended to directory listings if it exists. In other words, it doesn't get sorted

HeaderName HEADER.html
: The name of a file to be prepended to directory listings if it exists. In other words, it doesn't

IndexIgnore .??* * *# HEADER* README* RCS CVS *,v *,t

A set of regular expressions describing files that are not to be displayed in listings. In this case, files beginning with ”.”, ending in ” ” or ”#”, the HEADER and README files, RCS and CVS directories and version files.

There are as you can imagine, hundreds of additional directives and combinations of directives, but many of these deal with advanced features. In the following, we will look at some specific applications of the directives, but you are encouraged to look at the manuals and the web for more advanced configurations.