DZone Refcardz

CONTENTS INCLUDE:

- Configuration Syntax
- .htaccess files
- Logging
- Authentication
- Access Control
- Redirecting and Rewriting...and more!

Essential Apache HTTP Server

By Rich Bowen

ABOUT APACHE HTTPD

Apache httpd is the world's most popular web server, running more than half of all the websites on the planet. It runs on every modern operating system, and has numerous modules – both included with the base distribution, and third-party modules – which provide a wide range of optional functionality, for every type of website.

Apache httpd (http://httpd.apache.org/) is a project of the Apache Software Foundation (http://apache.org/)

It is assumed in this this Refcard that you are running either httpd 2.2 or httpd 2.4. 2.4 is the recommended version of the server.

CONFIGURATION SYNTAX

Apache httpd is configured via one or more text configuration files with a simple directive syntax, which can be edited with any text editor of your choice. There are three types of lines that may appear in a configuration file:

Configuration file syntax

Type	Example	Purpose
Comment	# This is a comment	Explanatory notes
Container or Section	<Directory /var/www> ... </Directory>	Indicate the scope of influence of a particular block of directives
Directive	FallbackResource /index.php	A keyword with one or more arguments, to set a particular configuration option.

A comment must begin a line. That is, a comment may not start midway through a line.

The default configuration file is typically called httpd.conf, but may be called anything you choose.

Other configuration files may be included in the main configuration using the Include directive:

```
Include conf.d/vhosts/my_website.conf
```

You may include more than one configuration file in a directory using file

```
Include conf.d/vhosts/*.conf
```

File paths in httpd configuration files are assumed to be relative to the ServerRoot if they do not start with a leading slash. The ServerRoot is configurable and must be an absolute path.

```
ServerRoot "/usr/local/apache2"
```

File paths which include spaces must be enclosed in quotes. You may enclose any directive argument in quotes if you choose.

MINIMAL CONFIGURATION

A minimal configuration needs to tell httpd to listen to a network port and to tell it where to find content it is to serve. Other directives may be optional, depending on your particular needs.

Two examples are provided due to a slight syntax change between the 2.2 and 2.4 versions of the server.

Example: httpd 2.2

```
Listen *:80
ServerName www.example.com
ServerRoot /var/www
DocumentRoot /var/www/htdocs

<Directory /var/www/htdocs>
  Order allow,deny
  Allow from all
</Directory>
```

Example: httpd 2.4

```
Listen *:80
ServerName www.example.com
ServerRoot /var/www
DocumentRoot /var/www/htdocs

<Directory /var/www/htdocs>
  Require All granted
</Directory>
```

In the examples shown, site content will be placed in the /var/www/htdocs directory, which will then be available at the '/' URL – that is, http://www.example.com/ will serve content from that directory.

The hostname 'www.example.com' must resolve to the IP address of your server in order for requests to reach you. That is outside of the scope of the Apache httpd server itself.

.HTACCESS FILES

While most of your configuration will go in the main server configuration file(s), there may be some situations in which someone who doesn't have access to that file will need to modify the configuration. Examples include content providers who only have access to their own content, and are not server administrators.

.htaccess files exist for these situations. .htaccess files permit configuration changes on a per-directory basis. That is, a .htaccess file in a given directory may override the configuration for that directory, and subdirectories thereof.

Because .htaccess files override the main server configuration, you need to restrict what can be put in them for security reasons. The AllowOverride directive specifies what directives will be honored in .htaccess files.

Specifying an argument of None indicates that .htaccess files shouldn't be considered at all.

```
AllowOverride None
```

The AllowOverride directive may take one or more arguments, specifying categories of functionality that you wish to permit to be overridden:

```
AllowOverride Options AuthConfig
```

In 2.4, you may also list specific configuration directives that you wish to allow:

```
AllowOverrideList Redirect RedirectMatch
```

Directives specified by AllowOverrideList are added to the directives permitted by a preceding AllowOverride directive.

.htaccess files override other .htaccess files found at higher directories. For example, the file /var/www/htdocs/one/.htaccess will override configurations found in /var/www/htdocs/.htaccess As a side effect of this, enabling .htaccess files will cause a reduction in performance as httpd will need to check every directory for .htaccess files. Also, since the contents of .htaccess files are not cached, this happens on every request for resources in these directories. You are therefore recommended to enable .htaccess files only in directories where they are strictly necessary.

Never enable .htaccess files in the <Directory /> configuration block, as this enables .htaccess files for the entire filesystem.

MODULES

When you installed Apache httpd, you will have installed various additional modules. Although it is possible to build modules statically – ie, as part of the main httpd binary – modules are usually installed as dso's – Dynamic Shared Objects – which can be loaded or unloaded as desired using the LoadModule directive. Loadmodule indicates the name of the module, and the location of the file containing that module.

LoadModule dumpio_module modules/mod_dumpio.so

To unload a module, turn the LoadModule line into a comment by starting it with a '#' character.

VIRTUAL HOSTS

More than one website may be run on the same Apache httpd server. This functionality is called 'virtual hosting', or 'vhosts'. To configure a vhost, you must indicate what network interface (ie, IP address and port number) a vhost should listen on, and then provide basic configuration directives to define that vhost.

```
NameVirtualHost *:80
# The NameVirtualHost line is optional on 2.4

<Virtualhost *:80>
  ServerName example.com
  ServerAlias www.example.com

  DocumentRoot /var/www/vhosts/example

  ErrorLog /var/log/apache2/example_error.log
  CustomLog /var/log/apache2/example_access.log
</VirtualHost>

<Virtualhost *:80>
  ServerName example.com
  ServerAlias www.example.com

  DocumentRoot /var/www/vhosts/example

  ErrorLog /var/log/apache2/example_error.log
  CustomLog /var/log/apache2/example_access.log
</VirtualHost>
```

Multiple SSL virtual hosts may be hosted on the same IP address using SNI – Server Name Indication.

PERFORMANCE

Because every server has different needs, there are a variety of configuration options available to tune the performance of an httpd server. When using a threaded MPM, you can tune the number of available threads using the MaxSpareThreads and MinSpareThreads directives to ensure that there will always be an adequate number of idle threads waiting to handle requests as they come in. For the Prefork MPM, the MinSpareServers and MaxSpareServers serve the same purpose.

```
StartServers 10
MinSpareServers 5
MaxSpareServers 10
```

Additional discussion of performance tuning may be found at http://httpd.apache.org/docs/current/misc/perf-tuning.html

LOGGING

Apache httpd provides various logging mechanisms. The most common are the access log and the error log. Logging configuration is slightly different between 2.2 and 2.4.

The error log is configured with the ErrorLog and LogLevel directives. ErrorLog may be defined globally, or per virtualhost.

The ErrorLog directive specifies a location where the log entries will be written. This can be either a log file location or a program to which the log entries will be piped.

To specify a log file location:

```
ErrorLog "/var/log/httpd/error_log"
```

To specify a pipe that will process the log entries:

```
ErrorLog "|/usr/local/bin/httpd_errors"
```

The LogLevel directive sets the verbosity of the logging.

LogLevel may be set globally or per virtualhost in 2.2, and in 2.4 it can additionally be set per directory, or per module, as shown in the examples below.

In 2.2, LogLevel may be set to one of the following values.

Level	Description	Example
emerg	Emergencies - system is unusable.	.Child cannot open lock file. Exiting…
alert	Action must be taken immediately.	"getpwuid: couldn't determine user name from uid"
crit	Critical Conditions.	"socket: Failed to get a socket, exiting child"
error	Error conditions.	"Premature end of script headers"
warn	Warning conditions.	"child process 1234 did not exit, sending another SIGHUP"
notice	Normal but significant condition.	"httpd: caught SIGBUS, attempting to dump core in ..."
info	Informational.	"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."
debug	Debug-level messages	"Opening config file ..."

Each setting includes log entries from higher severities. It is recommended that you set LogLevel to at least crit or error.

```
ErrorLog error
```

In 2.4, LogLevel may be set to any of those values, or to one of the following additional values:

Level	Description	Example
trace1	Trace messages	"proxy: FTP: control connection complete"
trace2	Trace messages	"proxy: CONNECT: sending the CONNECT request to the remote proxy"
trace3	Trace messages	"openssl: Handshake: start"
trace4	Trace messages	"read from buffered SSL brigade, mode 0, 17 bytes"
trace5	Trace messages	"map lookup FAILED: map=rewritemap key=keyname"
trace6	Trace messages	"cache lookup FAILED, forcing new map lookup"
trace7	Trace messages, dumping large amounts of data	"| 0000: 02 23 44 30 13 40 ac 34 df 3d bf 9a 19 49 39 15 |"
trace8	Trace messages, dumping large amounts of data	"| 0000: 02 23 44 30 13 40 ac 34 df 3d bf 9a 19 49 39 15 |"

In 2.4, ErrorLog may be configured at different levels per module, in case you want more information from a particular module without increasing it on any others:

```
ErrorLog crit ssl:warn
```

You can also set ErrorLog in a <Directory> block, which is not possible in 2.2.

The access log, which logs request information, is configured using the LogFormat and CustomLog directives.

LogFormat configures the information that will be put in a log file, and assigns an alias to a particular format. The following line defines the common log file format.

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

In the common format, the fields have the following meaning:

Format String	Meaning
%h	Address of client.
%l	Remote logname from identd, if supplied. .-. otherwise.
%u	Remote username, if the request was authenticated. .-. otherwise.
%t	The date and time that the request was received.
%r	The first line of the request.
%s	The HTTP status code of the response.
%b	The total number of bytes returned to the client, not including headers.

Numerous other format strings are available, and are documented at http://httpd.apache.org/docs/current/mod/mod_log_config.html#formats

Once a LogFormat is defined, it can then be used in a CustomLog directive, which may be set on a per-virtualhost basis:

```
CustomLog "/var/log/httpd/access_log" common
```

As with ErrorLog, CustomLog may point to the location of a log file, or may specify a program to which the messages will be piped.

Using the 'env=' syntax, CustomLog may specify a conditional environment variable to determine whether the log entry will be made:

```
SetEnvIf Request_URI \.gif$ gif-image
CustomLog gif-requests.log common env=gif-image
```

In the example shown here, the specified log file will be written to only if the requested URL has a .gif file extension.

AUTHENTICATION

httpd provides mechanisms for requiring password authentication for access to resources, supporting both the Basic and Digest HTTP authentication protocols. While Digest is slightly more difficult to configure, it is more secure, since Basic authentication passes your credentials (username and password) in plaintext across the connection, which may be intercepted if the connection is not secured with SSL.

User and group information may be stored in a variety of different places, (the authentication "provider") including flat files, dbm files, sql databases, or LDAP.

An authentication configuration block must specify an AuthType (Basic or Digest) and an AuthProvider (file, dbm, etc).

```
AuthType Basic
AuthName "Restricted Files"
AuthBasicProvider file
AuthUserFile /usr/local/apache/passwd/passwords
Require user rbowen
```

To restrict access to a group of users:

```
AuthType Basic
AuthName "By Invitation Only
AuthBasicProvider file
AuthUserFile /usr/local/apache/passwd/passwords
AuthGroupFile /usr/local/apache/passwd/groups
Require group GroupName
```

User files can be created using the htpasswd utility, or the htdigest utility for Digest authentication.

ACCESS CONTROL

You can control access to your server based on a number of different criteria.

In httpd 2.2 and earlier, use the 'Allow from' and 'Deny from' keywords, and the order in which they are applied is controlled with the Order directive. In 2.4 and later, much more specific requirement combinations may be used.

2.2 and earlier, recipes

```
# Allow all access
<Direcctory /var/www/htdocs>
  Order allow,deny
  Allow from all
</Directory>

# Deny all access:
<Directory /var/www/htdocs/secret
  Order deny,allow
  Deny from all
</Directory>

# Allow from certain addresses
<Directory /var/www/htdocs/intranet
  Order deny,allow
  Deny from all
  Allow from 192.68
</Directory>

# Deny from certain addresses
<Directory /var/www/htdocs/blog
  Order allow,deny
  Allow from all
  Deny from 223.10.0
</Directory>

# Require a particular env var
<Directory /var/www/htdocs/env>
  Order deny,allow
  Deny from all
  Allow from env=ALLOWED
</Directory>
```

2.2 and earlier also provide the Satisfy keyword, so that you can indicate that either all requirements listed must be honored (using the all keyword), or that any one of them is sufficient (using the any keyword).

```
# Require someone to be in the group “marketing”
# and also on the local network
<Directory /var/www/htdocs/marketing>
  Order deny,allow
  Deny from all


  Require group marketing
  Allow From 192.168
  Satisfy all
</Directory>
```

2.4 and later, recipes

```
# Allow all access
<Directory /var/www/htdocs>
  Reqire all granted
</Directory>

# Deny all access
<Directory /var/www/htdocs/secret>
  Require all denied
</Directory>

# Allow from certain addresses
<Directory /var/www/htdocs/intranet
  Require ip 192.168
</Directory>

# Deny from certin addresses
<Directory /var/www/htdocs/blog>
  Require not ip 223.10.0
</Directory>

# Require a particular env var
<Directory /var/www/htdocs/env>
  Require env ALLOWED
</Directory>

# Require an expression (new in 2.4)
<Directory /var/www/htdocs/other>
  Require expr %{TIME_HOUR} -ge 9 && %{TIME_HOUR} -le 17
</Directory>
```

2.4 also provides the <RequireAll>, <RequireAny> and <RequireNone> containers to allow you to combine several requirements in a variety of different ways.

```
# Require both authentication and an IP address
<Directory /var/www/htdocs/marketing>
  <RequireAll>
    Require group marketing
    Require ip 192.168
  </RequireAll>
</Directory>
```

You can nest multiple blocks to compose complicated authorization requirements.

Finally, note that while the 2.2 syntax is deprecated in 2.4, you can continue to use it by loading mod_access_compat, in order to ease the transition to the new syntax.

SSL

SSL – Secure Socket Layer – is a cryptographic protocol that enables secure communication to your HTTP server. Apache httpd's mod_ssl is a standard module that provides https (secure HTTP) for your web server.

To enable mod_ssl, you'll need a valid SSL certificate. To configure your server, you'll need the following:

```
LoadModule ssl_module modules/mod_ssl.so
Listen 443
<VirtualHost *:443>
  ServerName www.example.com
  SSLEngine on
  SSLCertificateFile /path/to/www.example.com.cert
  SSLCertificateKeyFile /path/to/www.example.com.key
</VirtualHost>
```

In the example, www.examle.com.cert is your SSL certificate, and www.example.com, key is your SSL key.

MOD_INFO

mod_info provides configuration information about your running Apache httpd server. To enable it:

```
LoadModule info_module modules/mod_info.so
<Location /server-info>
  SetHandler server-info

  # 2.2
  # Order deny,allow
  # deny from all
  # Allow from 127.0.0

  # 2.4
  # Require ip 127.0.0

</Location>
```

This enables the server-info handler for the URL /server-info, so loading http://localhost/server-info will display configuration information about your server.

It is recommended that you control access to this information, as it will be very useful to someone attempting to compromise your server. The commented-out sections of the example configuration show how you might do this for 2.2 and 2.4.

MOD_STATUS

mod_status provides real-time status about your running web server, including current request information and statistical load information. To enable it:

```
LoadModule status_module modules/mod_status.so
<Location /server-status>
    SetHandler server-status
    # Require host .example.com
    # Require ip 127
</Location>
```

Requesting the URL http://localhost/server-status will load the server-status page. Append the ?refresh=N URL argument to reload the page ever N seconds. Add the ?auto URL argument to return the page in a format that is more readily machine-parseable, if you want to automatically process the output with a script.

AUTOINDEX

If you specify a DirectoryIndex directive, a request for a directory will return that file. Typically this is index.html, or index.php, or some other standard index file.

If this file doesn't exist in the directory, mod_autoindex will provide a directory listing instead. This functionality is enabled with the directive:

```
Options +Indexes
```

The directive may be tweaked by adding a header, footer, and style sheet, and with a variety of other options.

```
HeaderName /style/header.html
ReadmeName /style/footer.html
IndexStyleSheet /style/index.css
IndexOptions FancyIndexing HTMLTables
```

See the mod_autoindex documentation for numerous other options.

CGI

CGI – the Common Gateway Interface – is the oldest way to provide dynamic content for the web. It's a specification for how programs should produce output that can be passed on to web clients.

You may configure your Apache httpd server for CGI programs in one of two ways. Either specify a directory where CGI programs must reside:

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

With this configuration, any file placed in that directory will be treated as a CGI program.

Or specify file extensions that will always be treated as CGI:

```
<Directory /var/www/htdocs>
  Options +ExecCGI
  AddHandler cgi-script .cgi
</Directory>
```

With this configuration, any file with a .cgi extension will be treated as a CGI program.

A CGI program must emit a valid HTTP header, followed by output to be displayed in the browser:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "Hello World!";
```

Input to a CGI program arrives via environment variables and STDIN. CGI libraries exist for all modern programming languages, and you are encouraged to use one of them, rather than inventing your own input parsing algorithms.

REDIRECTING AND REWRITING

You will frequently need to redirect one request location to another, and there are a variety of ways to do this.

The Redirect directive does a simple redirection from one location to another.

```
Redirect /one.html /another.html
Redirect /something/here.html http://elsewhere.com/there.html
```

The Alias directive maps a URL path to a directory path:

```
Alias /images /var/www/images
```

RewriteRule, on the other hand, provides regular-expression based URL manipulation, as well as a variety of other request-time request modifications.

The syntax of the RewriteRule directive is as follows:

```
RewriteRule regular-expression target [FLAG]
```

The regular expression is matched against the requested URL, and, if it matches, the URL is rewritten to the target value, which may include backrefereces from the matched regular expression.

One or more flag may be added to the end of the directive to modify the manner in which the transformation is made. Flags include the following:

Flag	Meaning
CO	Set a cookie with the response
F	Return a 403 Forbidden response
H	Force the response to be handled with the specified Handler
L	If this rule matches, don.t.run any further rules
P	Proxy the resulting target, using mod_proxy
R	Return a HTTP Redirect response to the client

Full documentation, and numerous examples, are available at http://httpd.apache.org/docs/rewrite/

COMMAND LINE UTILITIES

Several command-line utilities come with the Apache http server.

httpd
This is the main server binary, and is used to start, stop, and restart your server. It's what you'll see in your server process lists, although in some distributions of the server it will be called apache or httpd2. Some of the available command-line options are shown below.

Option	Purpose
httpd .k.start	Starts the server
httpd .k.stop	Stops the server
httpd .k.restart	Reloads the configuration file and restarts child processes
httpd .V.	Show configuration settings
httpd .S.	Dump virtual host configuration
httpd .M	List loaded modules
httpd .h.	Show all options

apachectl
This is a shell script intended to be used as a SysV init-style script. It also serves as a pass-thru to httpd for other options.

Command	Purpose
apachectl configtest	Runs a syntax check on the configuration files.
apachectl start	Starts the httpd daemon
apachectl restart	Restarts the httpd daemon if it.s.running. Starts it otherwise
apachectl stop	Halts the httpd daemon
apachectl graceful	Restarts the daemon gracefully, waiting for existing connections to complete
apachectl graceful-stop	Halts the httpd daemon, waiting for existing connections to complete

ab
Performs basic benchmarking by making repeated requests to the same resource. For example:

```
ab –n 1000 –c 10 http://localhost/index.html
```

The above invocation will request the specified URL 1000 times, using 10 concurrent threads to do so.

apxs
Asssts in the building and installing of extension modules from source code. For example, to build and install a third-party module mod_security, you'd download the source file, and then run:

```
apxs –cia mod_security.c
```

This uses the following command-line switches:

Switch	Meaning
-c	Compile the source to a .so file
-i	Install the resulting .so file in the modules directory
-a	Append the necessary LoadModule directive to the configuration file

htpasswd
Create or edit password files for use with file-based basic authentication.

Create a password file

```
httpasswd –c passwordfile username
```

Add a user to an existing password file

```
htpasswd passwordfile username2
```

htdigest
Operates just like htpasswd, for digest authentication. An additional argument, the authentication realm, is incorporated into the resulting password hash.

rotatelogs
A piped log handler that automatically rotates the log files based on size or time. The name of the resulting file can be specified using date and time information.

Rotate the log when it reaches 500M:

```
CustomLog "|bin/rotatelogs /var/logs/logfile 500M" common
```

Rotate the log every 24 hours:

```
CustomLog "|bin/rotatelogs -t /var/logs/logfile 86400" common
```

The above log files are created with the default file name of /var/logs/logfile.nnnn where nnnn is the system time stamp at which the log file starts.

To specify the resulting file names in a friendlier format:

```
ErrorLog "|bin/rotatelogs /var/logs/errorlog.%Y-%m-%d-%H_%M_%S 5M"
```

This syntax uses the standard strftime formats. The above configuration will result in a log file named in the format errorlog.YYYY-mm-dd-HH_MM_SS.

MORE INFORMATION

More information about the Apache HTTP Server may be obtained from the httpd website at http://httpd.apache.org/

The complete documentation is available at http://httpd.apache.org/docs/current/ for the current version, and at http://httpd.apache.org/docs/2.2 for version 2.2. You may also substitute specific version numbers for earlier versions. (i.e., 2.0 or 1.3)

Information about participating in the development of the project is available at http://httpd.apache.org/dev/

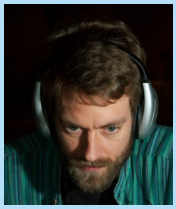
There are several mailing lists run by the Apache httpd project, where you can get help, or participate in discussion about the future of the project. To subscribe to one of these lists, send email to LISTNAME-subscribe@httpd.apache.org from the email address you wish to subscribe. To unsubscribe, send email to LISTNAME-unsubscribe@httpd.apache.org

List	Purpose
users@httpd.apache.org	User discussion and technical support
dev@httpd.apache.org	Discussion of the development and future direction of the httpd product
docs@httpd.apache.org	Improvement of the httpd documentation
cvs@httpd.apache.org	Commit messages for each change to the httpd code

An annual conference is held by the Apache Software Foundation, and it usually includes content about the httpd project. You can see upcoming events at http://apachecon.com/

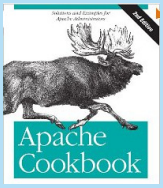
Live (or close to live) support is available on the channel #httpd on the freenode.net IRC network.

ABOUT THE AUTHOR



Rich Bowen is a world-renowned expert on Apache httpd. He has written several books on Apache server, including the Apache Cookbook and Apache Administrator's Handbook, and has contributed extensively to Apache httpd documentation since 2000. Rich regularly speaks at conferences, helped found the Habari blog software project, and currently serves as the Community Growth Hacker at SourceForge, where he gets to work with thousands of amazing Open Source projects.

RECOMMENDED BOOK



This book tackles everything from beginner problems to those faced by experienced users. For every problem addressed in the book, you will find a worked-out solution that includes short, focused pieces of code you can use immediately. You also get explanations of how and why the code works, so you can adapt the problem-solving techniques to similar situations.

DZone, Inc.
150 Preston Executive Dr.
Suite 201
Cary, NC 27513

888.678.0399
919.678.0300

Refcardz Feedback Welcome
refcardz@dzone.com

Sponsorship Opportunities
sales@dzone.com

ISBN-13: 978-1-936502-67-7
ISBN-10: 1-936502-67-4
50795
9 781936 502677
$7.95

Version 1.0