Apache Tutorials for Beginners

This is Complete Apache tutorials for beginners.

In this tutorial, you will learn-

Install and Download Apache

- What is Apache?
- How to install Apache
- Install Apache on Linux Platform
- Install Apache from Source

Learn about Apache Virtual Host in 10 minutes

- What is Virtual Host?
- Types of Apache Virtualhost
- Name-based Virtual Host
- IP-based Virtual host

How to Run PHP/Ruby with Apache?

- What Apache needs to Run Php File?
- Php handlers in Apache
- How to run Ruby with Apache

How to Secure Apache Web Server?

- Hiding Apache version and OS information
- Disable Directory Listing
- Disabling unnecessary modules
- Restricting Access to files outside the web root directory
- Using mod_evasive to rebutting the DoS attack
- Using mod_security to enhance apache security
- Limiting request size

Apache Log Format

- Available Apache Directives
- Types of Apache Log Format
- Common Log Format
- Combined Log Format

Configure your very first Production Web Sever

Install and Download Apache

What is Apache?

Apache is a remarkable piece of application software. It is the most widely used Web Server application in the world with more than 50% share in the commercial web server market. Apache is the most widely used Web Server application in Unix-like operating systems but can be used on almost all platforms such as Windows, OS X, OS/2, etc. The word, Apache, has been taken from the name of the Native American tribe '*Apache'*, famous for its skills in warfare and strategy making.

It is a modular, process-based web server application that creates a new thread with each simultaneous connection. It supports a number of features; many of them are compiled as separate modules and extend its core functionality, and can provide everything from server side programming language support to authentication mechanism. Virtual hosting is one such feature that allows a single Apache Web Server to serve a number of different websites.

How to install Apache

There are numerous ways of installing the package or application. There are enlisted below -

1. One of the features of this open **source web application is that anyone can make installer as per their own environment.** This has allowed various vendors like Debian, Red Hat, FreeBSD, Suse etc. to customize the file location and configuration of apache taking into account other installed applications and base OS.

2. Apart from installing it from a vendor based installer, there is always the option of building and installing it from the source code. Installing Apache from source file is a platform independent & works for all OS.

The apache web server is a modular application where the administrator can choose the required functionality and install different modules as per his/her requirement.

All modules can be compiled as a **Dynamic Shared Objects** (DSO is an object file that could be shared by multiple apps while they are executing) that exists separately from the main apache file. The DSO approach is highly recommended, it makes the task of adding/removing/updating modules from the servers configuration very simple.

Install Apache:Linux Platform

On Red Hat or rpm based systems

If you are using an rpm (RedHat Package Manager is a utility for installing application on Linux systems) based Linux distribution i.e. Red Hat, Fedora, CentOs, Suse, you can install this application by either vendor specific Package Manager or directly building the rpm file from the available source tarball.

You can install Apache via the default Package Manager available on all Red Hat based distributions like CentOs, Red Hat and Fedora.

```
[root@amsterdam ~]# yum install httpd
```

The apache source tarball could be converted into an rpm file using the following command.

```
[root@amsterdam ~]# rpmbuild -tb httpd-2.4.x.tar.bz2
```

It is mandatory to have -devel package installed on your server for creating .rpm file from source.

Once you convert the source file into an rpm installer, you could use the following command to install Apache.

```
[root@amsterdam ~]# rpm –ivh httpd-2.4.4-3.1.x86_64.rpm
```

After the installation the server does not start automatically, in order to start the service, you have to use any of the following command on Fedora, CentOs or Red Hat.

```
[root@amsterdam ~]# /usr/sbin/apachectl start

[root@amsterdam ~]# service httpd start

[root@amsterdam ~]# /etc/init.d/httpd start
```

Install Apache from Source

Installing apache from the source require the –devel package to be installed on your server. .You can find the latest available version of Apache, you can download ithere . Once you download the source file move it to the /usr/local/src folder.

```
[root@amserversterdam ~] cd /usr/local/src

[root@amserversterdam ~] gzip -d httpd-2.2.26.tar.gz

[root@amserversterdam ~] tar xvf httpd-2.2.26.tar

[root@amserversterdam ~] httpd-2.2.26
```

In order to see all configuration option available for Apache, you can use ./configure –help option. The most common configuration option is –prefix={install directory name}.

```
[root@amserversterdam ~]./configure --help

[root@amserversterdam ~]./configure –prefix=/usr/local/apache –enable-so

[root@amserversterdam ~] make

[root@amserversterdam ~] make install
```

The above example shows the compilation of Apache within the /usr/local/apache directory with the DSO capability. The –enable-so option, can load required modules to apache at run time via the DSO mechanism rather than requiring a recompilation.

Once the installation completes, you can browse the web servers default page with your favorite browser. If firewall is enabled on your server, you must have to make exception for port 80 on your OS firewall. You can use the following command to open port 80.

```
iptables -I INPUT -p tcp --dport 80 -j ACCEPT
```

service iptables save

You can see the default **Apache2 Welcome screen** by browsing your server IP address.

Learn about Apache Virtual Host in 10 minutes

What is Virtual Host?

An Apache web server can host multiple websites on the **SAME** server. You do not need separate server machine and apache software for each website. This can achieved using the concept of **Virtual Host** or**VHost.**

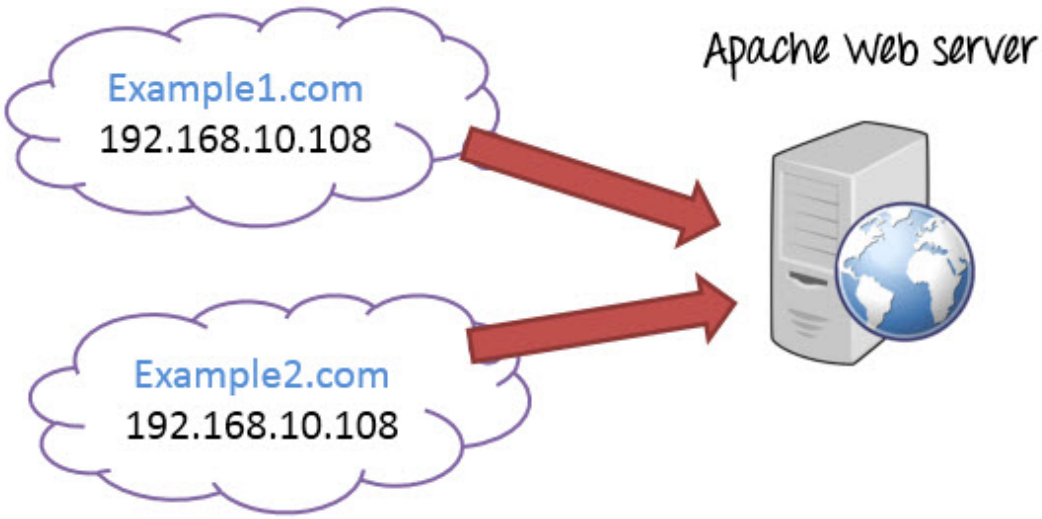
Any domain that you want to host on your web server will have a separate entry in apache configuration file.

Types of Apache Virtualhost

1. Name-based Virtual host
2. Address-based or IP based virtual host and.

Name-based Virtual Host

Name based virtual hosting is used to host multiple virtual sites on a single IP address.



In order to configure name based virtual hosting, you have to set the IP address on which you are going to receive the Apache requests for all the desired websites. You can do this by NameVirutalHost directive within the apache configuration i.e.**httpd.conf/apache2.conf file.**

Apache virtual host Example:

```
NameVirtualHost *:80

<VirtualHost 192.168.0.108:80>

ServerAdmin webmaster@example1.com

DocumentRoot /var/www/html/example1.com

ServerName www.example1.com

</VirtualHost>

<VirtualHost 192.168.0.108:80>

ServerAdmin admin@example2.com

DocumentRoot /var/www/html/example2.com

ServerName www.example2.com

</VirtualHost>
```

You can add as many virtual hosts, as per your requirement. You can check your web configuration files with:

```
[root@amsterdam ~]#httpd –t
Syntax OK
```

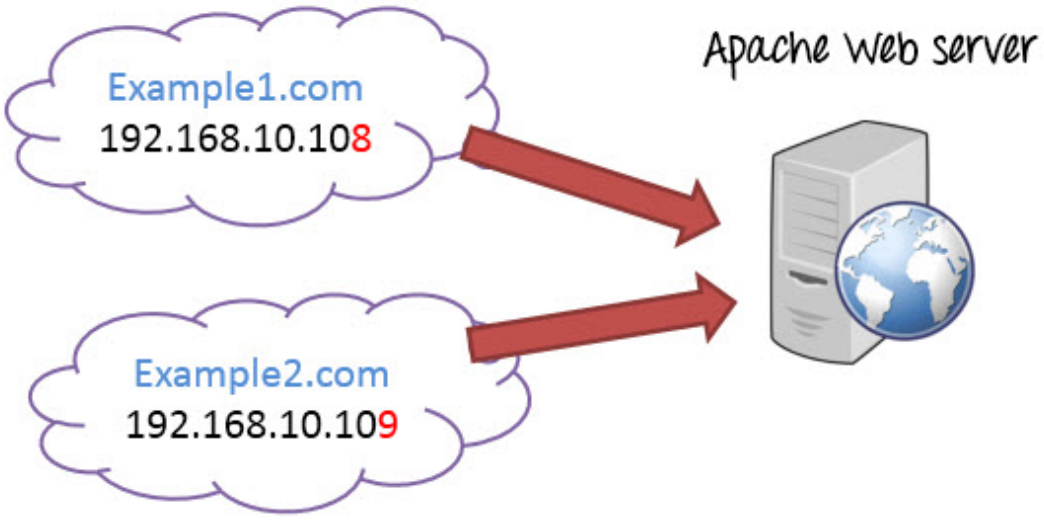
If the configuration file has some wrong syntax, it will throw an error

```
[root@115 conf.d]# httpd -t

Syntax error on line 978 of /etc/httpd/conf/httpd.conf:

Invalid command '*', perhaps misspelled or defined by a module not included in the server configura
tion
```

IP-based Virtual host

In order to setup IP based virtual hosting, you need more than one IP address configured on your server. So, the number of vhost apache will depend onnumber of IP address configured on your server. If your

server has 10 IP addresses, you can create 10 IP based virtual hosts.



In the above diagram two websites example1.com and example2.com were assigned different IPs and are using IP-based virtual hosting.

```
Listen 192.168.0.100:80

<VirtualHost 192.168.10.108:80>

ServerAdmin webmaster@example1.com

DocumentRoot /var/www/html/example1.com

ServerName www.example1.com

</VirtualHost>

<VirtualHost 192.168.10.109:80>

ServerAdmin admin@example2.com

DocumentRoot /var/www/html/example2.com

ServerName www.example2.com

</VirtualHost>
```

How to Run PHP/Ruby with Apache?

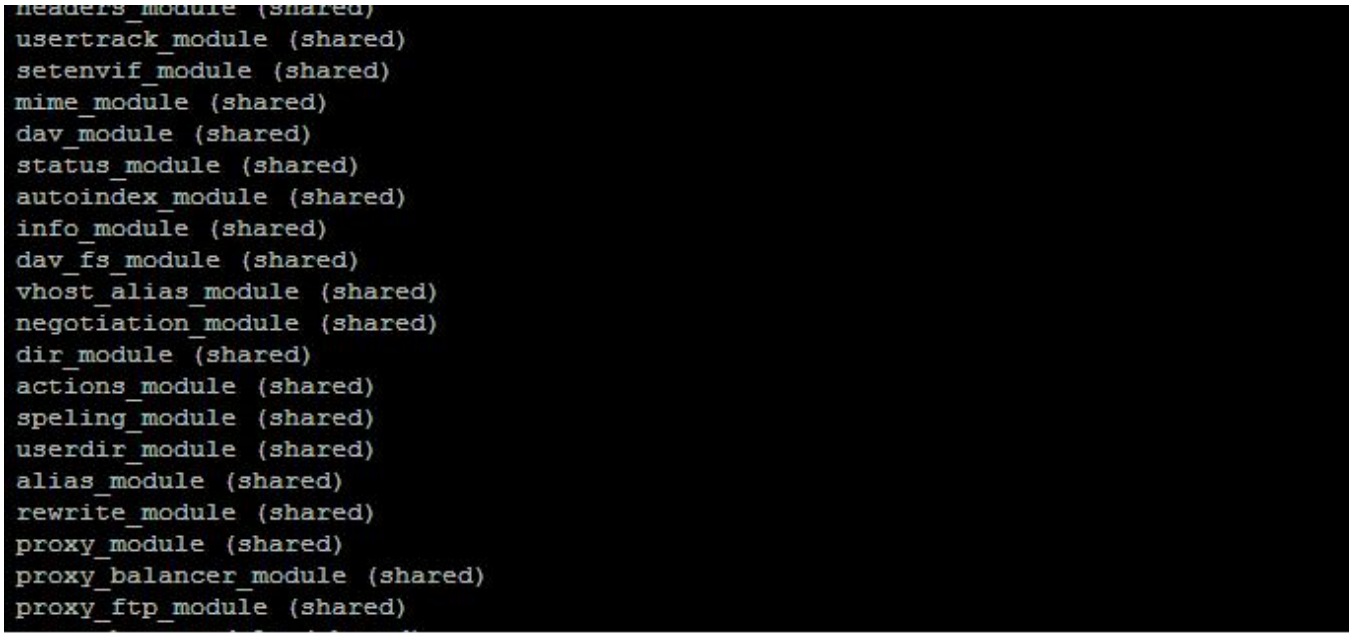
What Apache needs to Run Php File?

Running Php files on Apache needs **mod_php** enabled on your server. It allows Apache to interpret .Php files. It has Php handlers that interpret the Php code in apache and send HTML to your web server.

If mod_php is enabled on your server, you will have a file named php.conf in /etc/httpd/conf.d/ directory. You can also check it with:

```
httpd -M | grep "php5_module"
```

The output will be similar to:



Php handlers in Apache

- mod_php
- CGI
- FastCGI
- suPHP

mod_phpis the oldest PHP handler, it makes PHP part of apache and does not call any external PHP process. This module is installed by default in every Linux distribution repository, so enabling/disabling this module is very easy.

If you are using **FastCGI** as your PHP handler, you can set multiple versions of PHP to be used by different accounts on your server.

FastCGI i.e. **mod_fastcgi** is an extension of **mod_fcgid**, where as **mod_fcgid** is a high performance alternative of CGI i.e. mod_cgi . It starts sufficient number of instances of CGI to handle concurrent web requests. It also uses suexec to support different users with their own instances of PHP and improves web security.

Running ruby files on Apache needs mod_ruby to be enabled. Apache can also handle ruby files through FastCGI. It is possible to use multiple version of ruby with the help of mod_fcgid i.e. FastCGI.

You can also install apache passenger and configure Apache to use it for serving ruby pages.

(Phusion Passenger also known as "**passenger**" is a free web server module that is designed to integrate with Apache and Nginx)

Steps to install mod_ruby on your server -

```
cd /tmp

wget http://www.modruby.net/archive/mod_ruby-1.2.6.tar.gz

tar zxvf mod_ruby-1.2.6.tar.gz

cd mod_ruby-1.2.6/

./configure.rb --with-apr-includes=/usr/include/apr-1

make

make install
```

How to run Ruby with Apache

We have to add the mod_ruby module to the Apache configuration i.e. /etc/httpd/conf.d/ruby.conf and add the following line.

LoadModule ruby_module modules/mod_ruby.so

If you like to enable or disable these modules, you have to edit the apache configuration file and comment or uncomment these modules, if the web server is already compiled with these modules.

How to Secure Apache Web Server

Securing your web server is very important, it means allowing others to see only the intended information & protecting your data and restricting access.

These are common things that enhance your Apache web servers' security.

1) Hiding Apache version and OS information:

Apache displays its version and the name of the operating system in errors as shown in below screenshot.

A hacker can use this information to launch an attack using the publicly available vulnerabilities in the particular version of the server or OS.

In order to prevent Apache webserverfromdisplaying this information, we have to modify

"server signature" option available in the apache configuration file. By default, it is "on", we need to set it "off".

```
vim /etc/httpd/conf/httpd.conf
```

```
ServerSignature Off

ServerTokens Prod
```

We have also set "ServerTokens Prod" that tells the web server to return only apache and suppress the OS major and minor version

After modifying the configuration file, you have to restart/reload your apache web server to make it effective.

```
service httpd restart
```

2) Disable Directory Listing

If your document root directory does not have an index file, by default your apache web server will show all the content of the document root directory.

This feature could be turn off for a specific directory through “options directive” available in the Apache configuration file.

```
<Directory /var/www/html>

    Options -Indexes

</Directory>
```

3) Disabling unnecessary modules

It is good practice to disable all unnecessary modules that are not in use. You can see list of enabled module available in your apache configuration file -

```
[root@amsterdam ~]#httpd –M

perl_module (shared)

php5_module (shared)

proxy_ajp_module (shared)

python_module (shared)

ssl_module (shared)
```

Many of the listed modules can be disabled likemod_imap, mod_include, mod_info, mod_userdir, mod_autoindex, as they are hardly used by any production web servers.

```
vi /etc/httpd/conf/httpd.conf

#LoadModule auth_digest_module modules/mod_auth_digest.so
```

Once you commented the module, save the file.

Restart apache services with following command.

```
/etc/init.d/httpd restart
```

4) Restricting Access to files outside the web root directory

If you like to make sure that files that is outside the web root directory are not accessible, you have to make sure that the directory is restricted with “Allow” and “Deny option” in your web server configuration file.

```
<Directory/>

Options None

AllowOverride None

Order deny,allow

Deny from all

</Directory>
```

Once you restrict acess outside the web root directoy, you will not be able to access any file located on any other folder on your web server, you will get 404 return code.

5) Using mod_evasive to rebutting the DoS attack

If you like to protect your web server from Dos (i.e. Denial of Service) you must enable the module mod_evasive. It is a third party module that detects Dos attack and prevents the attack from doing as much damage as it would do if left to run its course. It could be downloaded here.

Download the above File

6) Using mod_security to enhance apache security

This module works as a firewall for Apache and allows you to monitor traffic in real time. It also prevents the web server from brute force attacks. The mod_security module could be installed with the default package manager of your distribution.

7) Limiting request size

Apache does not have any restriction on the total size of the http request that could lead to a DoS attack. You can limit the request size of an Apache directive “LimitRequestBody” with the directory tag. The value could be set anything from 0 to 2 GB (i.e. 2147483647 bytes) as per your requirement.

```
<Directory "/var/www/html/uploads">

   LimitRequestBody 512000
```

```
</Directory>
```

Apache Log Format

Apache logs provide detailed information that help to detect common issues with server.

In order create access logs, mod_log_configmodule must be enabled.

Three directives available in apache config file i.e.

- TransferLog: Creating a log file.
- LogFormat : Specifying a custom format.
- CustomLog : Creating and formatting a log file.

TransferLog directive is available in the apache configuration file and it rotates virtual host log files as per set parameters.

```
<VirtualHost www.example.com>

  ServerAdmin webmaster@example.com

  DocumentRoot /usr/www/example/httpd/htdocs/

  ServerName www.example.com

  ServerAlias example.com www.example

  ErrorLog /usr/www/example/httpd/logs/error_log

  TransferLog/usr/www/example/httpd/logs/accesslog

  CustomLog /usr/www/example/httpd/logs/accesslog combined

</VirtualHost>
```

Two types of Apache Log Format

- Common Log Format
- Combined Log Format.

You can enable them by editing the apache configuration file i.e. apache2.conf (Debian/ubuntu) or httpd.conf (rpm based systems) file

Common Log Format

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

CustomLog logs/access_log common
```

Common Log generated by Apache

```
[Wed Oct 11 14:32:52 2000] [error] [client 127.0.0.1] client denied by server configuration: /expor
t/home/live/ap/htdocs/test
```

Combined Log Format

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

CustomLog log/access_log combined
```

Here,

- %h is the remote host
- %l is the identity of the user determined by identd
- %u is the user name determined by HTTP authentication
- %t is the time the server finished processing the request.
- %r is the request line from the client. ("GET / HTTP/1.0")
- %>s is the status code sent from the server to the client (500, 404 etc.)
- %b is the size of the response to the client (in bytes)
- Referer is the page that linked to this URL.
- User-agent is the browser identification string.

Combined Log generated by Apache:

```
199.187.122.91 - - [06/Mar/2014:04:22:58 +0100] "GET /robots.txt HTTP/1.1" 404 1228 "-" "Mozilla/4.
0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)"
```

Custom Log creates separate log file for each Virtual Host on your server. It needs to be specified in the virtual host section of the config file.

You can see below mentioned virtual host configuration, generated log will be custom for that virtual host and the format will be combined.

Configure your very first Production Web Sever

1. In order to have a running production web server, you need a dedicated **node** (Physical/Virtual or cloud instance) running Linux/Unix, Windows, MacOS etc.

2. The Web Server must have a **direct network connection** and a **staticIP address** configured on it.

3. It needs to have all the **modules** required for running web pages. If a web server processes PHP pages, it needs to have PHP module enabled.

4. It also needs to have a good **Antivirus** application configured and running for securing the Web Server from Malware or Virus attacks. You also need mechanism to **update** the configured antivirus/anti malware application on regular basis without any manual intervention in order to get maximum benefit from them.
5. If you have hundreds of domains to be hosted on your web server, you must have to implement**limitations on file system quota for each domain, number of databases each domain can create, number of email accounts per domain etc.**
6. If your web server has been setup for **shared hosting services, users on your web server needs to be restricted**. A shared hosting user should have least user privilege so that he does not damage important files & break the entire server. Apache does not provide any such functionality and needs different third party applications, customization of OS to achieve this.
7. If you are adding a **new domain** on your web server, it needs editing hundreds of configuration file to enable all features for the added domain.
8. If one of the hosted domains requires **different PHP setting** than rest of the domains, implementing this in core Apache web server is very complex and needs customization of your web server in great extent.
9. A production web server needs a **firewall** to block unwanted traffic that could cause high load on your server. Implementing **IPTABLE** rules with command line is very complex. It needs expertise of core Linux/Unix environment to write effective firewall rules for blocking unwanted traffic. IPTABLE is based on netfilter module; it is an OS level firewall that allows an administrator to create rules for incoming/outgoing traffic on the server.
10. A production web server requires several different applications like **Email**, **FTP** for file upload, **Domain Name System** for parked domains. Managing all these applications on a core Linux/Unix system requires expertise on the respective technologies.

So, one can say that managing a web server for multiple domains is very complex task and requires editing hundreds of configuration file, customizing each application to fulfill the desired result. Troubleshooting any miss configuration will be very difficult for beginners.

The Solution using Cpanel or similar software

Cpanel provide a graphical way of managing your web server. It is meant to provide mass hosting services that is easy to use and configure. cPanel has reduced the technical barriers to entry into the hosting and web server management. It makes complex task easier, it provides many useful and easy to use web interfaces that perform common system administration tasks required to operate a web server.

cPanel compiles its own version of software.

If you have to recompile your web server i.e. apache on normal Linux platform, you have to manually select/search the module that is required. cPanel provides Easyapache functionality that is a script based web server compilation method.

It not only provides you web services but also Mail, DNS, FTP and many more services that is required for your web application.

A task that needs expertise on core Linux/Unix based hosting like installing SSLs, recompiling Apache with different PHP modules, updating Web Security, configuring effective IPTABLE rules, Adding ftp users, creating mail accounts for each domain, scanning your document root with antivirus and creating databases are easy to complete with cPanel.

It provides a lot of scripts that fixes, install and troubleshoot common administrative tasks.

It provides a backup and restore functionality eliminating the need to manually copy files to backup storage. If you are backing up your domain, cPanel will create a tar file that will contain document root folder, email accounts and mails, ftp accounts, databases, DNS records and other applications.

It also provides a robust documentation, andhas a very big community of users where you could discuss and get solution of your issues.

So, one can say that cPanel is a best application for managing your web server with required features. It provides you, easy to use interface for managing your domain and a mechanism to avoid complexity of managing core Web Server.

There are many competing products to cPanel like Plesk, ISPConfig, Ajenti, Kloxo, Open Panel, Zpanel etc.