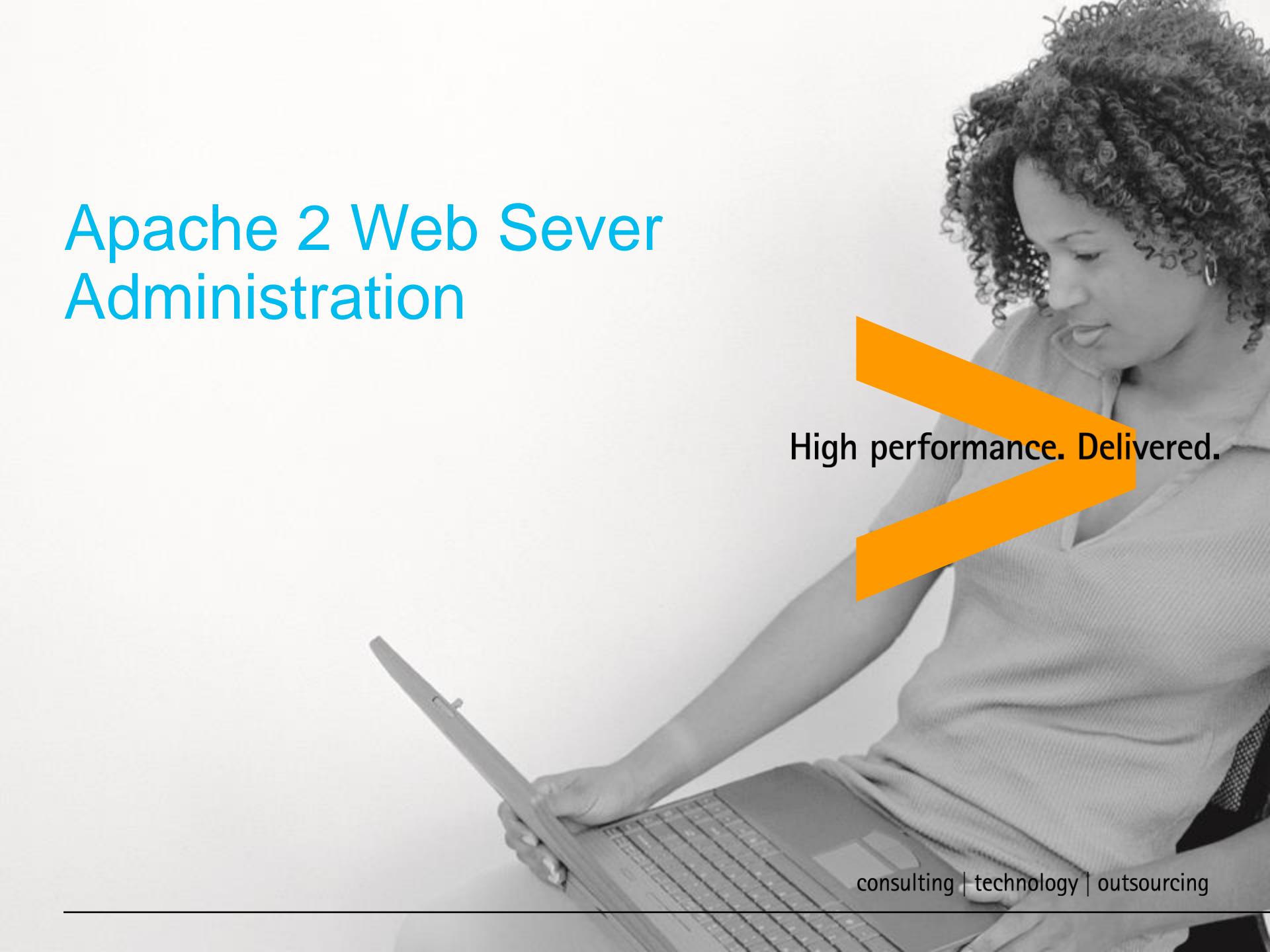


Apache 2 Web Server Administration



High performance. Delivered.

consulting | technology | outsourcing

Apache 2 Administration

Goals

- Introduction to Apache
- Installing Apache
- Managing Apache
- Modules & directives
- Configuring Apache
- Security
- Secure sockets layer (ssl) security

Apache 2 Administration

Goals

- Logs
- Dynamic web content
- Virtual hosts, redirection and indexing
- Redirection methods
- Enhance or limit functionality using modules
- Directory indexing
- Proxy servers and firewalls

Apache 2 Administration

Goals

- Monitoring apache web server
- Content Caching
- URL Mapping
- Apache web server status & server information
- Performance Tuning
- Improving performance

Apache 2 Administration

Goals

- Apache virtual host workflow and configuration in detail
- Apache MPM configurations and usages
- Server capacity (like RAM, ROM) needed for n number of users (say n=3000+).
- Apache tuning when there is high server load
- Other Apache mechanism for handling uncertain situations

APACHE

HTTP SERVER



The name Apache Server has been taken from Native American tribe '*Apache*', famous for its skills in warfare and strategy making





Apache Web Server (HTTP)

Apache is the most widely used web server software. Developed and maintained by Apache Software Foundation

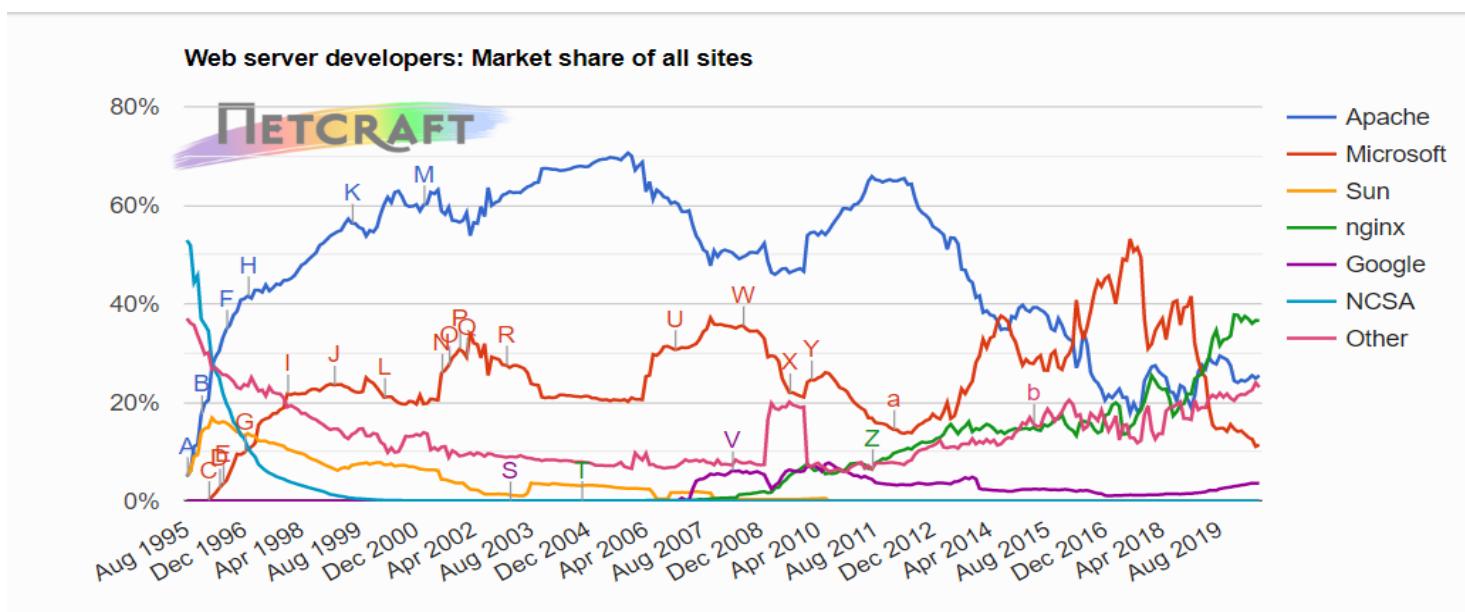


Introduction To Apache 2

- In the July 2020 survey Netcraft survey received responses from 1,234,228,567 sites across 260,658,118 unique domains and 10,221,919 web-facing computers.
- This represents a gain of 9.47 million sites and 180,000 computers, but a loss of 1.75 million domains.
- July 2020 most of the major server vendors saw gains in total sites this month: Apache gained 9.8 million sites.
- While Microsoft and nginx gained 5.4 million and 2.5 million sites respectively.
- LiteSpeed continued to see strong growth, gaining 1.95 million new sites this month.

Introduction To Apache 2

Developer	June 2020	Percent	July 2020	Percent	Change
nginx	448,673,487	36.63%	451,156,878	36.55%	-0.08
Apache	304,288,405	24.84%	314,054,523	25.45%	0.60
Microsoft	134,874,928	11.01%	140,264,332	11.36%	0.35
Google	43,449,240	3.55%	44,290,430	3.59%	0.04



Who is using Apache Web Server?

- Popular Companies Using Apache HTTP Server:
 - IBM
 - eBay
 - Adobe
 - PayPal
 - Linkedin
 - GTMetrix
 - Facebook
 - **Aspire**

Alternative to Apache HTTP Server

- Nginx
- Apache Tomcat
- Node.js
- Lighttpd
- Cherokee
- Microsoft IIS
- Appweb
- Hiawatha

Apache Rocks On

- Apache is a highly configurable Web Server with a modular design.
 - It is very easy to extend the capabilities of Apache Web server. Anyone with decent C or Perl programming expertise can write a module to perform a special function. This means that there are tons of Apache modules available for use.
- Apache is a free, open source technology.
- Apache works great with Perl, PHP, and other scripting languages.

Apache Rocks On

- Apache runs on Linux and other Unix systems.
- Apache also runs on Windows.

Apache: The Beginning

- In the early days of the Web, the National Center for Super Computing Applications (NCSA) created a Web server that became the number one Web server in early 1995.
- However, the primary developer of the NCSA Web server left NCSA about the same time, and the server project began to stall.
- In the meantime, people who were using the NCSA Web server began to exchange their own patches for the server and soon realized that a forum to manage the patches was necessary.
- **The Apache Group was born.** The group used the NCSA Web server code and gave birth to a new Web server called Apache.

Apache: The Beginning

- Originally derived from the core code of the NCSA Web server and a bunch of patches, the Apache server is now the talk of the Web server community.
- In three short years, it acquired the lead server role in the market.

Apache: The Beginning

- **The very first version (0.6.2)** of publicly distributed Apache was released in April 1995.
- The 1.0 version was released on December 1, 1995.
- The Apache Group has expanded and incorporated as a nonprofit group.
- The group operates entirely via the Internet.
- However, the development of the Apache server is not limited in any way by the group.
- Anyone who has the know-how to participate in the development of the server or its component modules is welcome to do so, although the group is the final authority on what gets included in the standard distribution of what is known as the Apache Web server.

Apache: The Beginning

- This allows literally thousands of developers around the world to come up with new features, bug fixes, ports to new platforms, and more.
- When new code is submitted to the Apache Group, the group members investigate the details, perform tests, and do quality control checks. If they are satisfied, **the code is integrated into the main Apache distribution.**

What Is Apache?

Supports virtual hosting

Allows you to implement highly dynamic coding

A popular choice among web developers

Compatible with Windows, macOS, Unix



A process-based, modular, open-source web server application

Apache vs IIS

Feature	IIS	Apache
Supported OS	Windows	Linux, Unix, Windows, Mac OS
User support & fixes	Corporate support	Community Support
Cost	Free, but bundled with Windows	Completely free
Development	Closed, proprietary	Open source
Security	Excellent	Good
Performance	Good	Good
Market Share	32%	42%

Current Apache Versions

Versions of Apache HTTP Server

Version	Initial release	Latest release
1.3	1998-06-06 ^[54]	2010-02-03 (1.3.42) ^[55]
2.0	2002-04-06 ^[56]	2013-07-10 (2.0.65) ^[57]
2.2	2005-12-01 ^[58]	2017-07-11 (2.2.34) ^[59]
2.4	2012-02-21 ^[60]	2020-04-01 (2.4.43) ^[61]

 Old version  Latest version

The Apache Feature List

- One of the greatest features that Apache offers is that it runs on virtually all widely used computer platforms.
- At the beginning, Apache used to be primarily a Unix based Web server, but that is no longer true.
- Apache not only runs on most flavors of Unix, but it also runs on Windows 2000/NT/9x and many other desktop and server-class operating systems such as Amiga OS 3.x and OS/2.
- **Apache HTTP Server versions later than 2.2 will not run on any operating system earlier than Windows 2000.**

The Apache Feature List

- Apache offers many other features including
 - fancy directory indexing
 - directory aliasing
 - content negotiations
 - configurable HTTP error reporting
 - SetUID execution of CGI Programs
 - resource management for child processes
 - server-side image maps
 - URL rewriting
 - URL spell checking
 - online manuals.

The Apache Feature List

- Apache offers many other features including
 - Simple, yet powerful file-based configuration
 - Support for CGI (Common Gateway Interface)(mod_cgi)
 - Support for FastCGI(mod_fcg)
 - Support for virtual hosts
 - Support for HTTP authentication
 - Integrated Perl(mod_perl)
 - Support for PHP scripting(mod_php)
 - Java Servlet support(jsp)

The Apache Feature List

- Apache offers many other features including
 - Handling of static files
 - Loadable dynamic modules
 - Auto-indexing
 - .htaccess
 - Compatible with IPv6
 - Supports HTTP/2
 - FTP connections
 - Gzip compression and decompression
 - Bandwidth throttling
 - Perl, PHP, Lua scripts
 - Load balancing
 - Session tracking
 - Geolocation based on IP address

The Apache Feature List

- Apache offers many other features including
 - Integrated Proxy server
 - Server status and customizable logs
 - Support for Server-Side Includes (SSI)
 - Support for Secured Socket Layer (SSL)(mod_ssl)

Fancy Indexing

- Fancy indexing shows the files plus additional information about them like the date the file is last modified, their size and description.
- To use that type of indexing, add the following two lines in a file named .htaccess in the destination folder:
 - Options +Indexes
 - IndexOptions +FancyIndexing

Directory Alias

- Directory alias are used to map between URLs entered in the browser to filesystem paths.
- Creating an alias on the installation server enables the URL: `http://servername/oss114` to map to the directory located at `/install/11-4` and solving my problem.

Content Negotiation

- Content negotiation is the ability of a web server to deliver the document that best matches the browser's preferences/capabilities.
- For example, if a resource exists in multiple languages, the web server can choose which variant it serves based on the Accept-Language header delivered by the browser.

SetUID execution of CGI Programs;

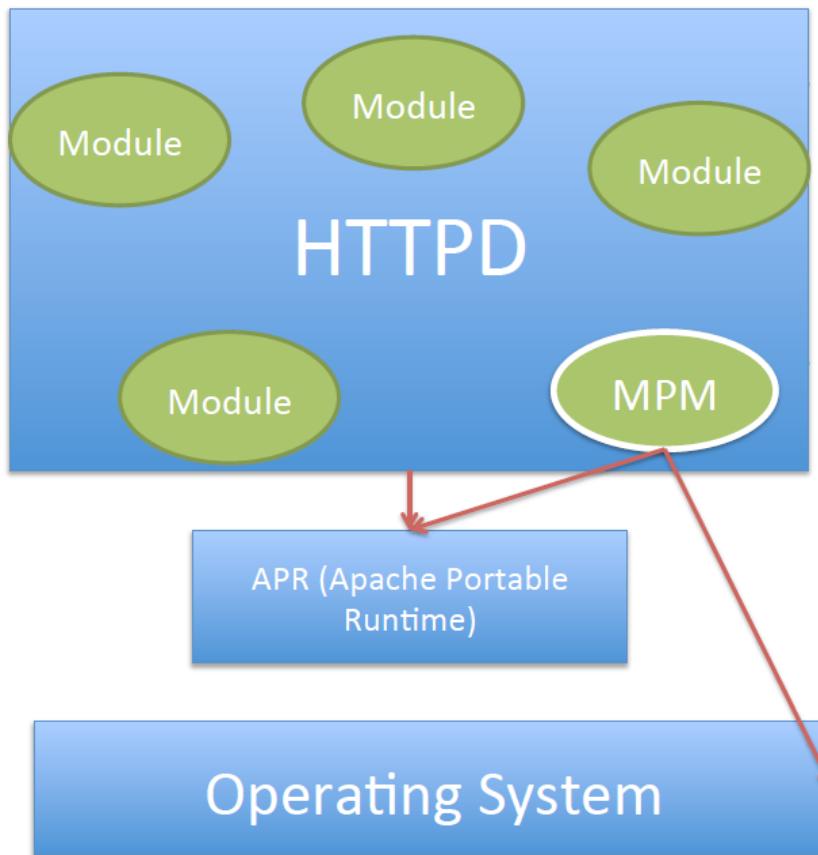
- CGI root access should have SetUID

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 be 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.

Apache 2 Architecture

Apache Architecture



small core
several modules

- compiled statically or loaded dynamically

Cross platform
utilities (APR)

Apache 2.4 Architecture

Multiprocessing modules

- The first major change in Apache 2.0 is the introduction of multiprocessing modules (MPMs).
- To understand why MPMs are created, you need to understand how Apache worked before.
- **Apache Version 1.3 or earlier used a preforking architecture.**
- In this architecture, an Apache parent process forked a set of child processes, which serviced the actual requests.
- The parent process simply monitored the children and spawned or killed child processes based on the amount of requests received.
- Unfortunately, this model didn't work well under platforms that are not process-centric such as Windows.
- So, the Apache Group came up with the MPMbased solution.

Apache 2.4 Architecture

- Each MPM is responsible for starting the server processes and for servicing requests via child processes or threads depending on the MPM implementation.
- Several MPMs are available.

How Apache Works: MPM

- MultiProcessing Modules (**MPMs**) since Apache2:
 - In apache 1.3 uses a preforking architecture
 - the parent creates/destroys children if required
 - does not work well on some platform (such Windows)
 - MPM offers several alternatives (implemented in MPM modules) :
 - prefork MPM (like Apache 1.3)
 - worker MPM (multiple child, each one with several **threads**)
 - winnt MPM: single process, multithread (specific for windows)

BeOS	beos
Netware	mpm_netware
OS/2	mpm_os2
Unix	prefork
Windows	mpm_winnt

```
<IfModule mpm_prefork_module>
  StartServers           5
  MinSpareServers       5
  MaxSpareServers       10
  MaxClients            150
  MaxRequestsPerChild   0
</IfModule>
```



We can tune parameters in
`/etc/apache2/apache2.conf`

```
<IfModule mpm_worker_module>
  StartServers           2
  MaxClients             150
  MinSpareThreads        25
  MaxSpareThreads         75
  ThreadsPerChild        25
  MaxRequestsPerChild    0
</IfModule>
```

Apache 2.4 Architecture

- **The prefork MPM**
- The prefork MPM mimics the Apache 1.3 or earlier architecture, creating a pool of child processes to service requests.
- Each child process has a single thread.
- For example, if Apache starts 30 child processes, it can service 30 requests simultaneously.
- If something goes wrong and the child process dies, only a single request is lost.

Apache 2.4 Architecture

- The number of child processes is controlled using a minimum and maximum setting.
- When the number of requests increases, new child processes are added until the maximum is reached.
- Similarly, when the requests fall, any extra child processes are killed.

Apache 2.4 Architecture

The threaded MPM

- This MPM enables thread support in Apache 2.0.
- This is like the prefork MPM, but instead of each child process having a single thread, each child process is allowed to have a specified number of threads.
- Each thread within a child process can service a different request.
- If Apache starts 30 child processes where each child is allowed to have at maximum 10 threads, then Apache can service $30 \times 10 = 300$ requests simultaneously.

Apache 2.4 Architecture

The threaded MPM

- If something goes wrong with a thread, for example, an experimental module causes the thread to die, then the entire process dies.
- This means that all the requests being serviced by the threads within the child process will be lost.
- However, because requests are distributed among threads on separate child processes, it is likely that a child's death will take down at maximum of $1/n$ of all the total connection, where n presents the number of all simultaneous connections.

Apache 2.4 Architecture

The threaded MPM

- A process is added or removed by monitoring its spare-thread count.
- For example, if a process has less than the minimum number of spare threads, a new process is added.
- Similarly, when a process has a maximum number of idle threads, it killed.
- All processes run under the same user and group ID assigned to Apache server.
- Because threads are more resource efficient than processes, this MPM is very scalable.

Apache 2.4 Architecture

The **perchild** MPM

- This is also new in Apache 2.0.
- In this MPM model a set number of child processes are started with a specified number of threads.
- As request load increases the processes add new threads as needed.
- When request count reduces, processes shrink their thread counts using a minimum and maximum thread count setting.
- The key difference between this module and the threaded MPM is that the process count is static and also each process can run using a different user and group ID.

Apache 2.4 Architecture

The **perchild** MPM

- This is also new in Apache 2.0.
- In this MPM model a set number of child processes are started with a specified number of threads.
- As request load increases the processes add new threads as needed.
- When request count reduces, processes shrink their thread counts using a minimum and maximum thread count setting.
- The key difference between this module and the threaded MPM is that the process count is static and also each process can run using a different user and group ID.
- This makes it easy to run different virtual Web sites under different user and group IDs.

Apache 2.4 Architecture

The winnt MPM

- This is the MPM for the Windows platform, including Windows 2000, Windows NT, and Window 9x.
- It is a multithreaded module.
- Using this module Apache will create a parent process and a child process.
- The child process creates all the threads that services the request.
- Also, this module now takes advantage of some Windows-only native function calls, which allows it to perform better than the earlier versions of Apache server on Windows platform.

Apache 2.4 Architecture

Filtering I/O

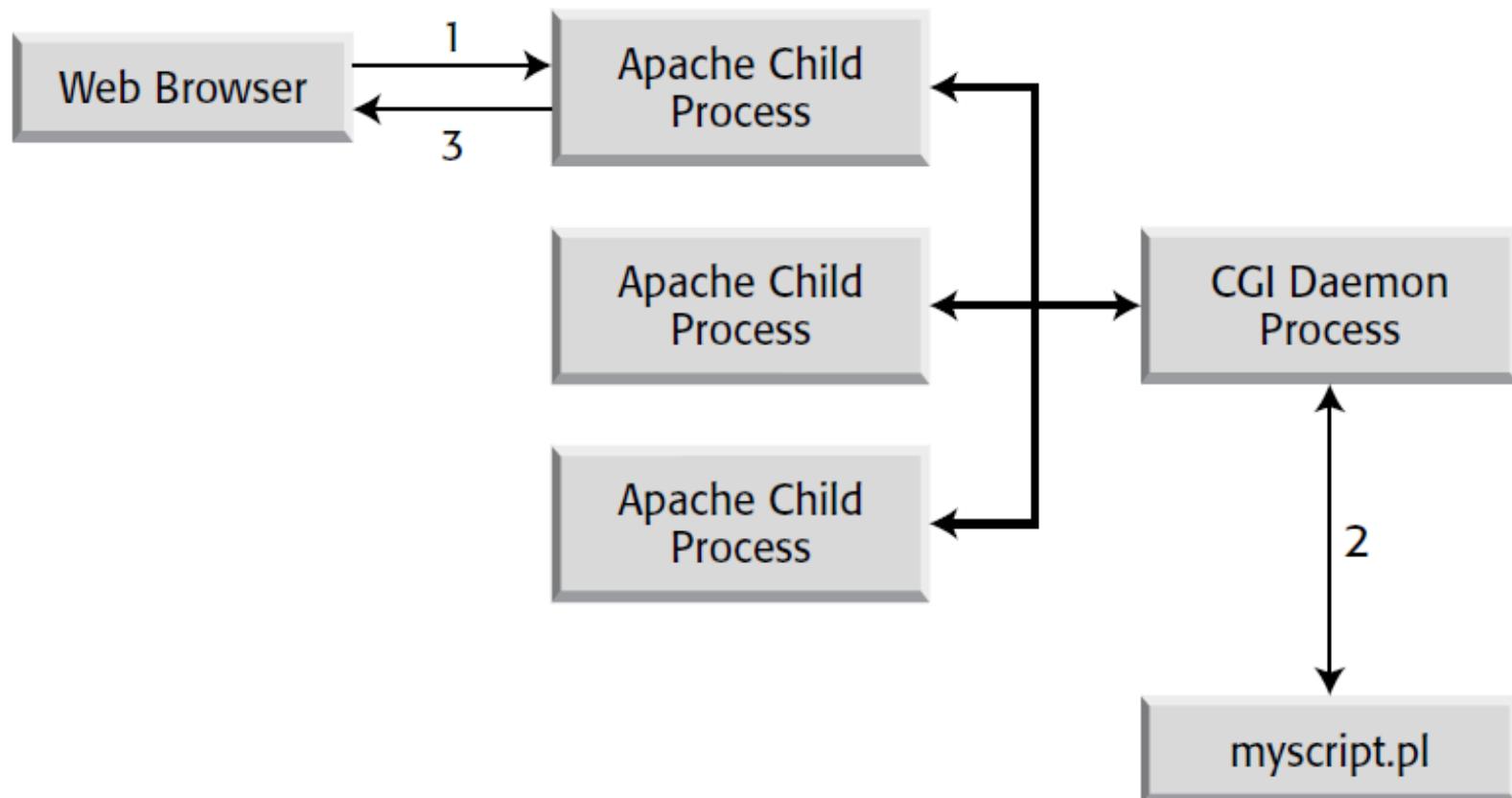
- Apache 2.0 now provides architecture for layered I/O.
- This means that one module's output can become another module's input.
- This filtering effect is very interesting.
- For example, the output produced by CGI scripts, which is processed by the mod_cgi module, can now be handed to the mod_include module responsible for SSIs.
- SSI (Server Side Includes) are directives that are placed in HTML pages, and evaluated on the server while the pages are being served.
- In other words, CGI scripts can produce output as SSI tags, which can be processed before the final output is sent to the Web browser.

Apache 2.4 Architecture

New CGI daemon

- Because many of the MPM modules use threads, executing CGI scripts become cumbersome when a thread gets such a request.
- The mod_cgi module still works, but not optimally for threaded MPMs, so mod_cgid was added.
- The mod_cgid module creates a daemon process, which spawns CGI processes and interacts with threads more efficiently.
- Figure shows how a CGI request for a script called myscript.pl is serviced.

Apache 2.4 Architecture



Apache 2.4 Architecture

- Here is how the CGI scripts are executed:
- 1. When the CGI request comes to a thread within a child process, it passes the request to the CGI daemon.
- 2. The CGI daemon spawns the CGI script and outputs the CGI script-generated data to the thread in the child process.
- 3. The thread returns the data to the Web browser.

Apache 2.4 Architecture

- When the main Apache server starts, it also starts the CGI daemon and establishes a socket connection.
- So, when a new child process is created, it inherits the socket connection and therefore does not have any need to create a connection to the CGI daemon for each request.
- The entire process improves CGI execution in the threaded environment.

Apache 2.4 Architecture

Apache Portable Run-Time

- In furtherance of the Apache Group's vision to create the most popular Web server in the world, it became clear that Apache's portability needed to be addressed in Apache 2.0.
- Prior to the current release, Apache had been dealing with portability internally, which made the code base less manageable.
- So, Apache Group introduced the Apache Portable Run-Time (APR). APR's purpose is to provide a single C interface to platform-specific functions so that code can be written once.
- This enables Apache to run more natively on platforms such as Windows, BeOS, Amiga, and OS/2.
- Because of APR, many programs, such as ApacheBench, can run on these platforms.

Understanding the Apache License

- Free software such as Apache, Perl (Practical Extraction and Reporting Language), and Linux (an x86-based Unix clone operating system) are getting a great deal of press because of Netscape's decision to make Netscape Communicator, one of the most popular Web browsers, available for free with its Mozilla project.
- Unfortunately, free software such as Apache, Perl, and Linux do not share the same licensing agreements, and so the media has created some confusion by associating these packages in the same licensing category.

Understanding the Apache License

- All free software is intended to be, well, free for all.
- However, there are some legal restrictions that the individual software licenses enforce.
- For example, Linux, which is made free by GNU Public License (GPL), requires that any changes to Linux be made public.
- Apache, on the other hand, does not require that changes made to Apache be made public by anyone.

Understanding the Apache License

- In short, think of Apache as free, copyrighted software published by the Apache Group.
- It is neither in the public domain nor is it shareware.
- Also note that Apache is not covered by GPL.

How to install & configure Apache on a Windows server

- Step 1 - Download Apache for Windows
- Step 2 - Unzip
- Step 3 - Configure Apache
- Step 4 - Start Apache
- Step 5 - Check Apache
- Step 6 - Install Apache as a Windows service
- Step 7 - Monitor Apache (optional)

How to install & configure Apache on a Windows server

Step 1 - Download



Apache Lounge

Webmasters

[Home](#)

[VC15](#)

[VC14](#)

[Additional](#)

NEW

6 May 2019
mod_fcgid 2.3.10

31 March 2019
httpd 2.4.39

25 March 2019
Snapshot httpd 2.4.39

28 February 2019
httpd 2.4.38 Updated

New C++ Redistributable
14.16.27027.1

21 January 2019
httpd 2.4.38

Apache 2.4 VC15 Windows Binaries and Modules

Apache Lounge has provided up-to-date Windows binaries and popular third-party modules for more than 15 years. We have hundreds of thousands of satisfied users: small and big companies as well as home users. Always build with up to date dependencies and latest compilers, and tested thorough. The binaries are referenced by the ASF, Microsoft, PHP etc. and more and more software is packaged with our binaries and modules.

The binaries, are build with the sources from ASF at httpd.apache.org, contains the latest patches and latest dependencies like zlib, openssl etc. which makes the downloads here mostly more actual then downloads from other places. The binaries **do not run** on XP and 2003. Runs on: 7 SP1, Vista SP2, 8 / 8.1, 10, Server 2008 SP2 / R2 SP1, Server 2012 / R2, Server 2016.

Build with the latest Windows® Visual Studio C++ 2017 aka VC15. VC15 has improvements, fixes and optimizations over VC14 in areas like Performance, MemoryManagement, New standard conformance features, Code generation and Stability. For example code quality tuning and improvements done across different code generation areas for "speed". And makes more use of modern processors and supported Windows editions (win 7 and up) internal features.

Be sure !! that you have installed the latest (14.16.27027.1) C++ Redistributable Visual Studio 2017 : [vc_redist_x64](#) or [vc_redist_x86](#).

Note: VC15 is backward compatible to VC14. That means, a VC14 module can be used inside a VC15 binary (for example PHP VC14 as module). Because this compatibility the version number of the Redistributable is 14.1x.xx. And after install is the Redistributable VS2015 updated from 14.0x.xx to VS2017 14.1x.xx (you can still use VC14).

Apache 2.4 binaries VC15

Info & Changelog

Apache 2.4.39 Win64

• [httpd-2.4.39-win64-VC15.zip](#) 31 Mar '19 17.264k

PGP Signature (Public PGP key), SHA1-SHA512 Checksums

How to install & configure Apache on a Windows server

- Step 2 - Unzip
 - Once the download has completed, open the downloaded "httpd-2.4.39-win64-VC15.zip" file, and extract its contents to a suitable location on your server i.e. C:\Apache24 or D:\Apache, etc

How to install & configure Apache on a Windows server

- **Step 3 - Configure Apache**

- Once you've extracted Apache, configure it.
- Start by locating the file "httpd.conf" (which will be in the "conf" sub directory), and open this in a standard text editor.
- By default, this configuration file assumes that you've extracted Apache to C:\Apache24.
- If however you've extracted Apache to a different location (i.e. D:\Apache), you'll need to update the \${SRVROOT} variable within in the httpd.conf file to point to the new location accordingly, i.e:
- Define SRVROOT "C:/Apache24" → Define SRVROOT "D:/Apache"

How to install & configure Apache on a Windows server

• Step 3 - Configure Apache

- If there is no SRVROOT variable present in your httpd.conf file, then instead we'll need to manually update all instances of "C:/Apache24", i.e.:
- ServerRoot "C:/Apache24" → ServerRoot "D:/Apache"
- DocumentRoot "C:/Apache24/htdocs" → DocumentRoot "D:/Apache/htdocs"
- <Directory "C:/Apache24/htdocs"> → <Directory "D:/Apache/htdocs">
- ScriptAlias /cgi-bin/ "C:/Apache24/cgi-bin/" → ScriptAlias /cgi-bin/ "D:/Apache/cgi-bin/"

How to install & configure Apache on a Windows server

- **Step 3 - Configure Apache**

- Regardless of where you extracted Apache to, you'll also need to make the following modifications to the http.conf file:
- A) Add "ExecCGI" to "Options" directive:
- Locate the following line:
- Options Indexes FollowSymLinks
- ...and append "ExecCGI":
- Options Indexes FollowSymLinks ExecCGI
- ...this tells Apache that CGI/Perl scripts are allowed outside of the cgi-bin directory

How to install & configure Apache on a Windows server

- **Step 3 - Configure Apache**

- B) Locate and uncomment the following line: (by removing the # symbol from the start of the line)
- AddHandler cgi-script .cgi
- ...and also add the following line:
- AddHandler cgi-script .pl
- ...These two lines tell Apache how to handle .cgi/.pl files (i.e. execute them rather than present them to a web browser)

How to install & configure Apache on a Windows server

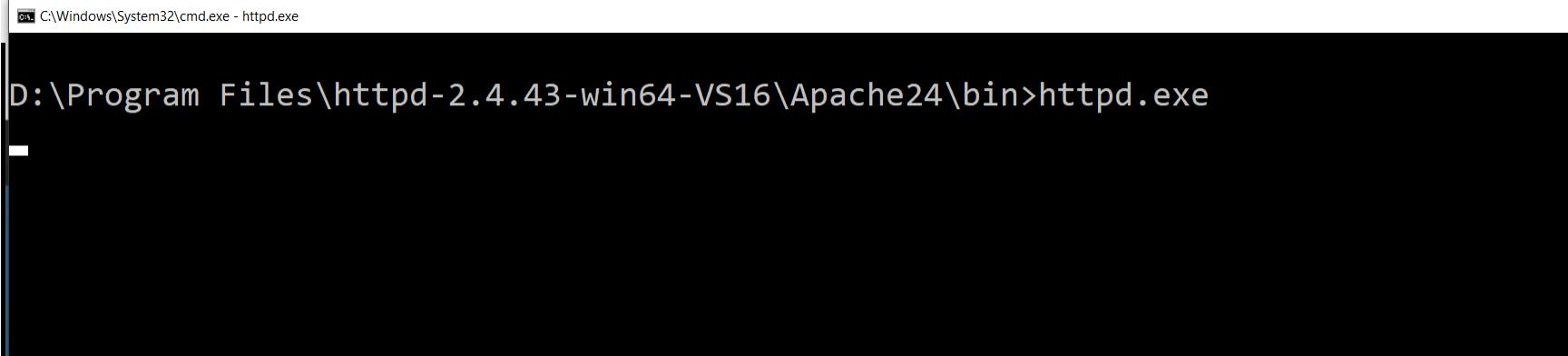
- **Step 3 - Configure Apache**

- C) Add the following line to the end of the httpd.conf file:
- ScriptInterpreterSource Registry
- ...this allows Apache to ignore the very first line of .cgi/.pl files which direct Apache to the install location of Perl, and instead determine the location of Perl from the Windows Registry

How to install & configure Apache on a Windows server

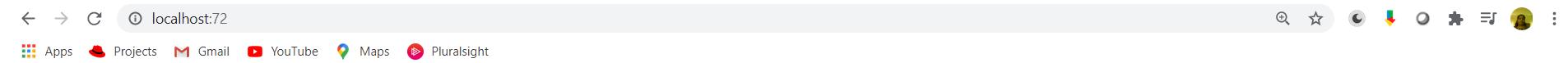
- **Step 4 - Start Apache**

- Open an command/PowerShell prompt in the "bin" folder at the location where you extracted Apache (Hold "Shift" whilst right-clicking and select "Open command window here" or "Open PowerShell window here"):



```
C:\> C:\Windows\System32\cmd.exe - httpd.exe
D:\Program Files\httpd-2.4.43-win64-VS16\Apache24\bin>httpd.exe
```

How to install & configure Apache on a Windows server



It works!

How to install & configure Apache on a Windows server

Step 6: Start as Window Service

```
D:\Program Files\httpd-2.4.43-win64-VS16\Apache24\bin>httpd.exe -k install
[Fri Jul 31 23:31:19.335427 2020] [mpm_winnt:error] [pid 21096:tid 460] AH00433: Apache2.4: Service is already installed
.

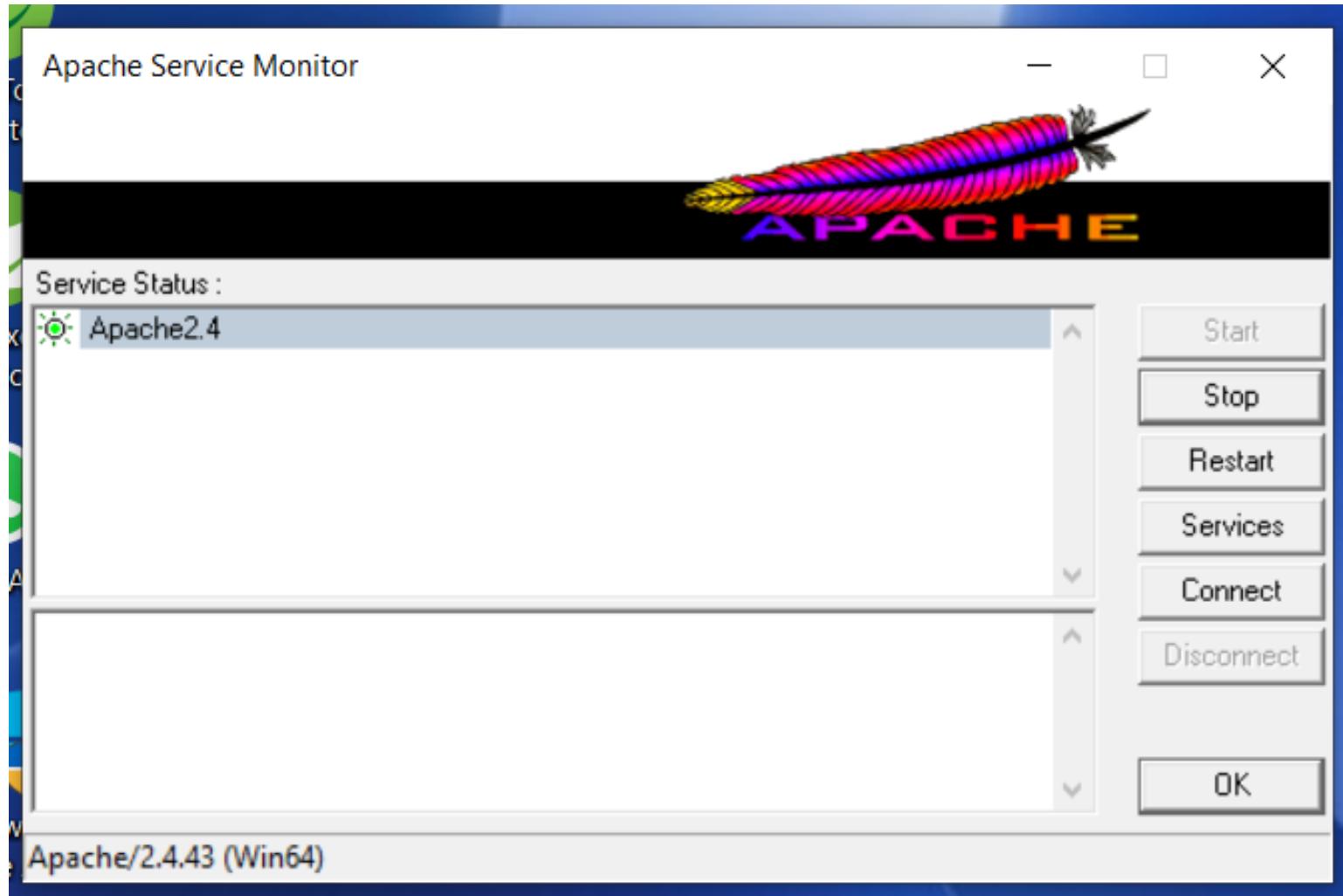
D:\Program Files\httpd-2.4.43-win64-VS16\Apache24\bin>net start apache2.4
The Apache2.4 service is starting.
The Apache2.4 service was started successfully.

D:\Program Files\httpd-2.4.43-win64-VS16\Apache24\bin>
```

How to install & configure Apache on a Windows server

- Step 7 - Monitor Apache (optional)
- To allow you to monitor the current state of your Apache server, as well as allow you to quickly start/stop/restart the server, Apache comes with a small utility called "Apache Monitor".
- Double click ApacheMonitor.exe from the bin folder to run the utility, or place a shortcut to it in your Startup folder so that it automatically runs whenever Windows starts.

How to install & configure Apache on a Windows server



Apache Directory structure

Name	Date modified	Type	Size
bin	21/04/2020 20:06	File folder	
cgi-bin	21/04/2020 20:06	File folder	
conf	21/04/2020 20:06	File folder	
error	21/04/2020 20:06	File folder	
htdocs	21/04/2020 20:06	File folder	
icons	21/04/2020 20:06	File folder	
include	21/04/2020 20:06	File folder	
lib	21/04/2020 20:04	File folder	
logs	31/07/2020 23:31	File folder	
manual	21/04/2020 20:06	File folder	
modules	21/04/2020 20:04	File folder	
ABOUT_APACHE.txt	21/02/2020 06:03	Text Document	14 KB
CHANGES.txt	21/04/2020 19:36	Text Document	1 KB
INSTALL.txt	17/05/2016 23:29	Text Document	4 KB
LICENSE.txt	21/04/2020 19:37	Text Document	44 KB
NOTICE.txt	21/04/2020 19:36	Text Document	3 KB
OPENSSL-NEWS.txt	21/04/2020 20:04	Text Document	43 KB
OPENSSL-README.txt	21/04/2020 20:04	Text Document	5 KB
README.txt	23/01/2014 22:03	Text Document	5 KB

Apache Directory structure

- include—Contains all the C header (include) files that are only needed if you develop Web applications that integrate with Apache or want to use use third-party software with Apache, which might require the header files.
- On a production server you can remove this directory.

Apache Directory structure

- lib — Houses the Apache Portable Run-Time (APR) library files, the files that are required for running Apache, and other support utilities such as ab.
- bin—Contains the programs shown in Table .
- conf—Houses the configuration files for Apache. It contains the files listed in Table

Apache Directory structure

- **htdocs**—This is the default document root directory for the main Apache server. The `httpd.conf` file sets the `DocumentRoot` directive to this directory.
- By default the `htdocs` directory also has the entire Apache manual installed in a subdirectory.
- **icons**—Used to store various Apache icons that are needed for displaying dynamically the build directory listing.

Apache Directory structure

- logs—Used to store the Apache server logs, the CGI daemon socket (cgisock), and the PID file (httpd.pid).
- cgi-bin—The default CGI script directory, which is set by using the ScriptAlias directive in httpd.conf. By default, Apache comes with two simple CGI scripts—printenv and test-cgi. Each of these scripts prints out CGI environment variables when requested via `http://server_name/cgi-bin/script_name` URL. These scripts are good for testing whether CGI configuration is working for you.

Apache Directory structure

- logs—Used to store the Apache server logs, the CGI daemon socket (cgisock), and the PID file (httpd.pid).
- cgi-bin—The default CGI script directory, which is set by using the ScriptAlias directive in httpd.conf. By default, Apache comes with two simple CGI scripts—printenv and test-cgi. Each of these scripts prints out CGI environment variables when requested via `http://server_name/cgi-bin/script_name` URL. These scripts are good for testing whether CGI configuration is working for you.

Apache Directory structure

- It is highly recommended that you remove the printenv and test-cgi scripts after you have your CGI configuration working.
- It is not a good idea to have a script that displays your system environment information to anyone in the world.
- The less you tell the world about how your system works or about what is available on your system, the more secure your system remains.

Apache Directory structure

Apache Programs in the bin Directory

<i>Apache Programs</i>	<i>Purpose</i>
ab	This is the ApacheBench program. It enables you to benchmark Apache server.
apachectl	This is a handy script that enables you to start, restart, and stop Apache server.
apxs	This is a tool for building and installing extension modules for Apache. It allows you to build DSO modules that can be used in Apache by using the mod_so module. For more information on this program, see http://your_server_name/manual/programs/apxs.htm .
htdigest	This program creates and updates user authentication information when message digest (MD5) authentication is being used.

Apache Directory structure

<i>Apache Programs</i>	<i>Purpose</i>
htpasswd	This program is used to create and update user authentication information used in basic HTTP authentication.
httpd	This is the Apache Web server program.
logresolve	This program converts (resolves) IP addresses from a log file to host names.
rotatelogs	This program rotates Apache logs files when they reach a certain size.

Apache Directory structure

Apache config Directory Contents

Configuration File	Purpose
httpd.conf	This is the Apache configuration file.
httpd-std.conf	This is the sample copy of the httpd.conf file, which is not required by Apache. For new Apache users, this file can act as a means for recovering the default httpd.conf.
highperformance.conf	This is a sample configuration file that shows some pointers for configuring Apache for high performance.
highperformance-std.conf	This is a sample copy of the highperformance.conf file, which is not required by Apache.
magic	This file stores the magic data for mod_mime_magic Apache module.
mime.types	<p>This file is used to decide which MIME-type headers are sent to the Web client for a given file.</p> <p>For more information about MIME types, please read RFC 2045, 2046, 2047, 2048, and 2077. The Internet media-type registry is at ftp://ftp.iana.org/in-notes/iana/assignments/media-types.</p>

Apache Directory structure

```
Administrator: Command Prompt
21/04/2020 07:54 PM          18,432 wintty.exe
05/04/2019 04:00 PM          86,016 zlib1.dll
 30 File(s)    9,340,904 bytes
   3 Dir(s)  163,210,084,352 bytes free

D:\Program Files\httpd-2.4.43-win64-VS16\Apache24\bin>htpasswd.exe -c -b passwd.txt eswaribala vigneshbala
Adding password for user eswaribala

D:\Program Files\httpd-2.4.43-win64-VS16\Apache24\bin>
```

Configuring Apache

- By default, Apache reads a single configuration file called `httpd.conf`.
- Every Apache source distribution comes with a set of sample configuration files.
- In the standard Apache source distribution, you will find a directory called `conf`, which contains sample configuration files with the `-dist` extension.
- The very first step you need to take before you modify this file is to create a backup copy of the original.

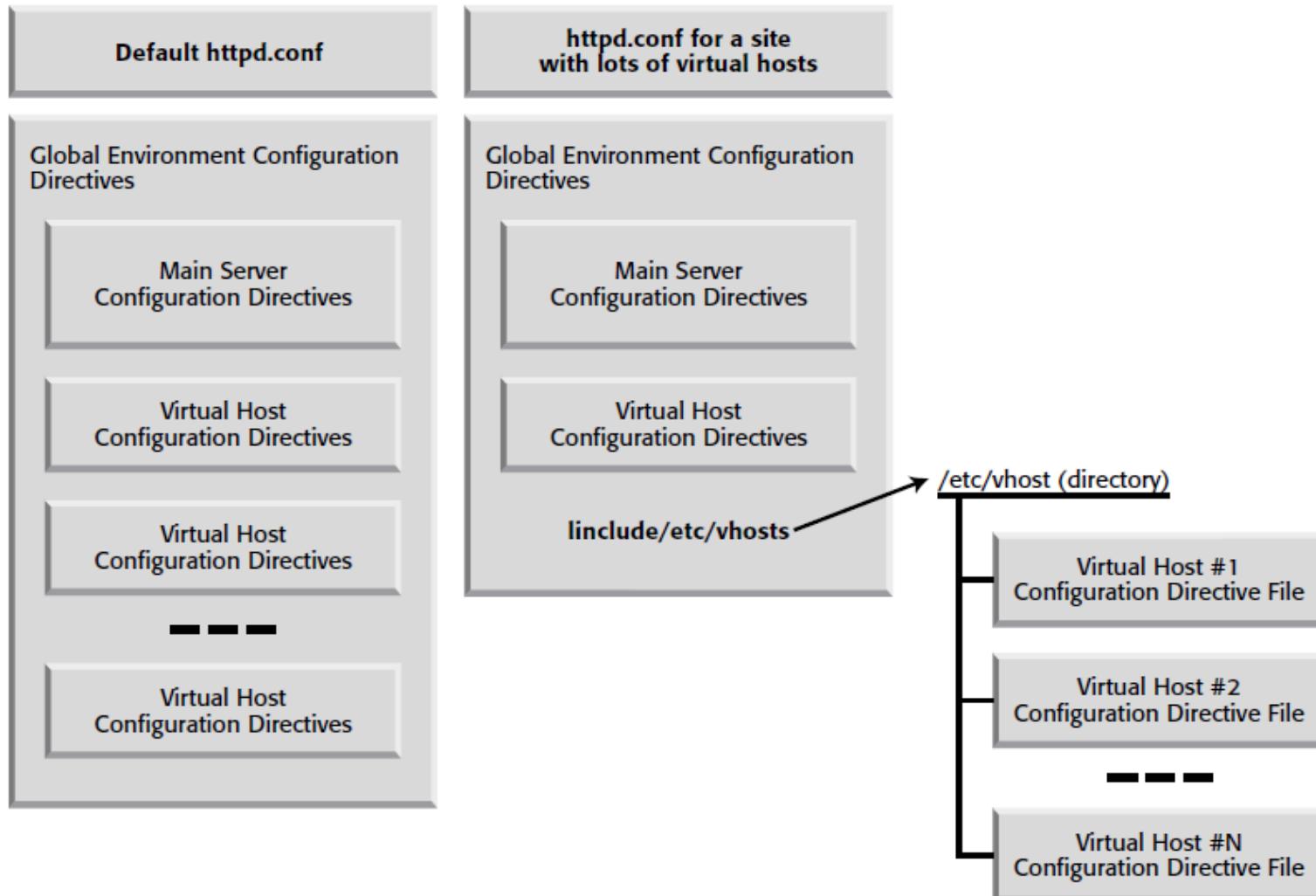
Configuring Apache

- The httpd.conf file has two types of information: comments and server directives.
- Lines starting with a leading # character are treated as a comment line; these comments have no purpose for the server software, but they serve as a form of documentation for the server administrator.
- You can add as many comments as you want; the server simply ignores all comments when it parses the file.

Configuring Apache

- Except for the comments and blank lines, the server treats all other lines as either complete or partial directives.
- A directive is like a command for the server. It tells the server to do a certain task in a particular fashion.
- While editing the httpd.conf file, you need to make certain decisions regarding how you want the server to behave.

Httpd.conf Segments



Httpd.conf Segments

- The left side of the figure shows how the default httpd.conf can be visualized in your mind.
- There are configuration directives that create the global server environment that applies to everything; there are configuration options that apply to the main (default) Web site Apache servers, and there are configuration directives that only apply to optional virtual hosts.
- Because Apache uses a single configuration file, a site with lots of virtual hosts will have a very large file and management of the configuration becomes very cumbersome.

Configuring the global environment for Apache

- The very first directive is ServerRoot, which appears as follows:
- ServerRoot “/usr/local/apache”
- This directive specifies the top-level directory of the Web server.
- The specified directory is not where you keep your Web contents.
- It is really a directory, which normally has these subdirectories:

Configuring the global environment for Apache

```
{ServerRoot Directory}
    ---bin
    ---conf
    ---htdocs
    ---htdocs/
        |
        +---manual
            |---developer
            |---howto
            |---images
            |---misc
            |---mod
            |---platform
            |---programs
            |---search
            +---vhosts

    |---icons

        |
        +---small

    ---logs
    ---cgi-bin
    +---include
```

Configuring the global environment for Apache

- /usr/local/apache is the parent directory for all server-related files.
- The default value for ServerRoot is set to whatever you choose for --prefix option during source configuration using the configure script.
- By default, the make install command executed during server installation copies all the server binaries in %ServerRoot%/bin, server configuration files in %ServerRoot%/conf, and so on

Configuring the global environment for Apache

- You should only change the value of this directive if you have manually moved the entire directory from the place on installation to another location.
- For example, if you simply run `cp -r /usr/local/apache /home/apache` and want to configure the Apache server to work from the new location, you will change this directive to `ServerRoot /home/apache`.
- Note that in such a case you will also have to change other direct references to `/usr/local/apache` to `/home/apache`.

Configuring the global environment for Apache

- Also note that whenever you see a relative directory name in the configuration file, Apache will prefix %ServerRoot% to the path to construct the actual path.

PidFile

- The PidFile directive sets the PID (process ID) file path.
- By default, it is set to logs/httpd.pid, which translates to %ServerRoot%/logs/httpd.pid (that is, /usr/local/apache/logs/httpd.pid).
- Whenever you want to find the PID of the main Apache process that runs as root and spawns child processes, you can run the cat %ServerRoot/logs/httpd.pid command.
- Don't forget to replace %ServerRoot% with an appropriate value.



```
eswaribala@DESKTOP-55AGI0I:/ eswaribala@DESKTOP-55AGI0I: cd eenvvars  
-bash: cd: eenvvars: No such file or directory  
eswaribala@DESKTOP-55AGI0I:/ cd envvars  
-bash: cd: envvars: Not a directory  
eswaribala@DESKTOP-55AGI0I:/ vi envvars  
eswaribala@DESKTOP-55AGI0I:/ vi apache2.conf  
eswaribala@DESKTOP-55AGI0I:/ cat %ServerRoot/logs/apache2.pid  
cat: %ServerRoot/logs/apache2.pid: No such file or directory  
eswaribala@DESKTOP-55AGI0I:/ ls  
apache2.conf  conf-available  conf-enabled  envvars  magic  mods-available  mods-enabled  ports.conf  sites-available  sites-enabled  
eswaribala@DESKTOP-55AGI0I:/ vi apache2.conf  
eswaribala@DESKTOP-55AGI0I:/ sudo nano envvars  
[sudo] password for eswaribala:  
eswaribala@DESKTOP-55AGI0I:/ cd ..  
eswaribala@DESKTOP-55AGI0I:/etc$ cd ..  
eswaribala@DESKTOP-55AGI0I:$ ls var  
backups  cache  crash  lib  local  lock  log  mail  opt  run  snap  spool  tmp  www  
eswaribala@DESKTOP-55AGI0I:$ ls var/run/apache2  
apache2.pid  
eswaribala@DESKTOP-55AGI0I:$ vi var/run/apache2/apache2.pid  
eswaribala@DESKTOP-55AGI0I:$ ps  
 PID TTY      TIME CMD  
 10 tty1      00:00:00 bash  
 675 tty1      00:00:00 ps  
eswaribala@DESKTOP-55AGI0I:$ vi var/run/apache2/apache2.pid  
eswaribala@DESKTOP-55AGI0I:$ ps  
 PID TTY      TIME CMD  
 10 tty1      00:00:00 bash  
 677 tty1      00:00:00 ps  
eswaribala@DESKTOP-55AGI0I:$
```



PidFile

Administrator: Command Prompt

```
D:\Program Files\httpd-2.4.43-win64-VS16\Apache24\bin>httpd -v
Server version: Apache/2.4.43 (Win64)
Apache Lounge VS16 Server built:   Apr 21 2020 16:23:13

D:\Program Files\httpd-2.4.43-win64-VS16\Apache24\bin>
```

Pid file

```
eswaribala@DESKTOP-55AGI0I: /etc/apache2
cron.monthly          hostname      magic        pm           shells       zsh_command_not_found
cron.weekly           hosts        magic.mime   polkit-1    skel
crontab               hosts.allow  mailcap     pollinate   sos.conf
cryptsetup-initramfs hosts.deny    mailcap.order popularity-contest.conf ssh
crypttab              init.d      manpath.config profile     ssl
eswaribala@DESKTOP-55AGI0I:/etc$ cd apache2
eswaribala@DESKTOP-55AGI0I:/etc/apache2$ ls
apache2.conf  conf-available  conf-enabled  envvars  magic  mods-available  mods-enabled  ports.conf  sites-available  sites-enabled
eswaribala@DESKTOP-55AGI0I:/etc/apache2$ -
```

D:\Program Files\httpd-2.4.43-win64-VS16\Apache24\logs

ScoreBoardFile

ScoreBoardFile is encapsulated within an if condition by using the

<IfModule . . .> container as shown below:

```
<IfModule !perchild.c>
```

```
ScoreBoardFile logs/apache_runtime_status
```

```
</IfModule>
```

- This tells Apache to set the ScoreBoardFile to %ServerRoot%/logs/ apache_runtime_status file only if you have chosen a multiprocessing module (MPM) other than perchild.

ScoreBoardFile

Because the default MP including Linux, is threaded instead of perchild, the if condition will be true and Apache will set the ScoreBoardFile directive.

This directive is used to point to a file, which is used to exchange run-time status information between Apache processes for most operating systems,

Timeout, KeepAlive, MaxKeepAliveRequests, and KeepAliveTimeout



- Timeout sets the server timeout in seconds. The default should be left alone.
- The next three directives KeepAlive, MaxKeepAliveRequests, and KeepAliveTimeout are used to control the keep-alive behavior of the server.
- You do not need to change them.

Timeout, KeepAlive, MaxKeepAliveRequests, and KeepAliveTimeout



- IfModule containers Apache will use one of the next three <IfModule . . .> containers based on which MPM you chose.
- For example, if you configured Apache using the default MPM mode (threaded) on a Linux system, then the following <IfModule . . .> container will be used:
 - <IfModule threaded.c>
 - StartServers 3
 - MaxClients 8
 - MinSpareThreads 5
 - MaxSpareThreads 10
 - ThreadsPerChild 25
 - MaxRequestsPerChild 0
 - </IfModule>

Timeout, KeepAlive, MaxKeepAliveRequests, and KeepAliveTimeout



- On the other hand, if you chose --with-mpm=prefork during source configuration by using the configure script, then the following <IfModule . . .> container will be used:
 - <IfModule prefork.c>
 - StartServers 5
 - MinSpareServers 5
 - MaxSpareServers 10
 - MaxClients 20
 - MaxRequestsPerChild 0
 - </IfModule>
- Similarly, the --with-mpm=perchild option forces Apache to use the last <IfModule . . .> container

Directives for threaded (default) MPM behavior

- If you did not change the default MPM behavior during source compilation and used the threaded behavior, so the directives that you need to consider are
- StartServers
- MaxClients
- MinSpareThreads
- MaxSpareThreads
- ThreadsPerChild
- MaxRequestPerChild

StartServers

- StartServers tells Apache to start three child servers as it starts.
- You can start more servers if you want, but Apache is pretty good at increasing number of child processes as needed based on load.
- So, changing this directive is not required.

MaxClients

- In the default threaded MPM mode, the total number of simultaneous requests that Apache can process is $\%MaxClients\% \times \%ThreadsPerChild\%$.
- So, because the default for MaxClients is 8 and the default for ThreadsPerChild is 25, the default maximum for simultaneous requests is 200 (that is, 8 times 5).
- If you use the preforking MPM mode, the maximum requests is limited to $\%MaxClients\%$.
- The default maximum of 200 simultaneous requests should work well for most sites, so leave the defaults.

MinSpareThreads



-
- The MinSpareThreads directive specifies the minimum number of idle threads.
 - These spare threads are used to service requests and new spare threads are created to maintain the minimum spare thread pool size.
 - You can leave the default settings alone.

MaxSpareThreads



-
- The MaxSpareThreads directive specifies the maximum number of idle threads; leave the default as is.
 - In the default threaded mode, Apache kills child processes to control minimum and maximum thread count.

ThreadsPerChild



- This directive defines how many threads are created per child process.
- If you are running Apache on a Windows system, set ThreadsPerChild to the maximum number of simultaneous requests that you want to handle, because on this platform there is only one child process, and it owns all the threads.

MaxRequestPerChild



- The final directive for the global environment is MaxRequestPerChild, which sets the number of requests a child process can serve before getting killed.
- The default value of zero makes the child process serve requests forever.
- I do not like to the default value because it enables Apache processes to slowly consume large amounts of memory when a faulty mod_perl script, or even a faulty third-party Apache module, leaks memory.

MaxRequestPerChild



-
- If you do not plan on running any third-party Apache modules or mod_perl scripts, you can keep the defaults or else set it to a reasonable number.
 - A setting of 30 ensures that the child process is killed after processing 30 requests.

Configuring the main server

- The main server configuration applies to the default Web site Apache serves.
- This is the site that will come up when you run Apache and use the server's IP address or host name on a Web browser.

Port

- The very first directive in this section is the Port directive, which sets the TCP port that Apache listens to for connections.
- The default value of 80 is the standard HTTP port.
- If you change this to another number, such as 8080, you can only access the server using a URL such as <http://hostname:8080/>.
- You must specify the port number in the URL if the server runs on a nonstandard port.

Port

- There are many reasons for running Apache on nonstandard ports, but the only good one I can think of is that you do not have permission to run Apache on the standard HTTP.

User and Group directives

- The User and Group directives tell Apache the user (UID) and group (GID) names to use.
- These two directives are very important for security reasons.
- When the primary Web server process launches a child server process to fulfill a request, it changes the child's UID and GID according to the values set for these directives.
- Refer to Figure to see how the primary Web server process that listens for the connection runs as a root user process, and how the child processes run as different user/group processes.
- If the child processes are run as root user processes, a potential security hole will be opened for attack by hackers.

User and Group directives

- Enabling the capability to interact with a root user process maximizes a potential breach of security in the system; hence, this is not recommended.
- Rather, I highly recommend that you choose to run the child server processes as a very low-privileged user belonging to a very low-privileged group.
- In most Unix systems, the user named nobody (usually UID = -1) and the group named nogroup (usually GID = -1) are low-privileged.
- You should consult your /etc/group and /etc/passwd files to determine these settings.

User and Group directives

- If you plan to run the primary Web server as a nonroot (regular) user, it will not be able to change the UID and GID of child processes, because only root user processes can change the UID or GID of other processes.
- Therefore, if you run your primary server as the user named ironsheik, then all child processes will have the same privileges as ironsheik.
- Similarly, whatever group ID you have will also be the group ID for the child processes.

ServerAdmin

- ServerAdmin defines the e-mail address that is shown when the server generates an error page.
- Set this to your e-mail address.
- Now you need to set the host name for the Server using the ServerName directive.
- This directive is commented out by default because Apache install cannot guess what host name to use for your system.
- So if the host name is called www.domain.com, set ServerName directive accordingly.

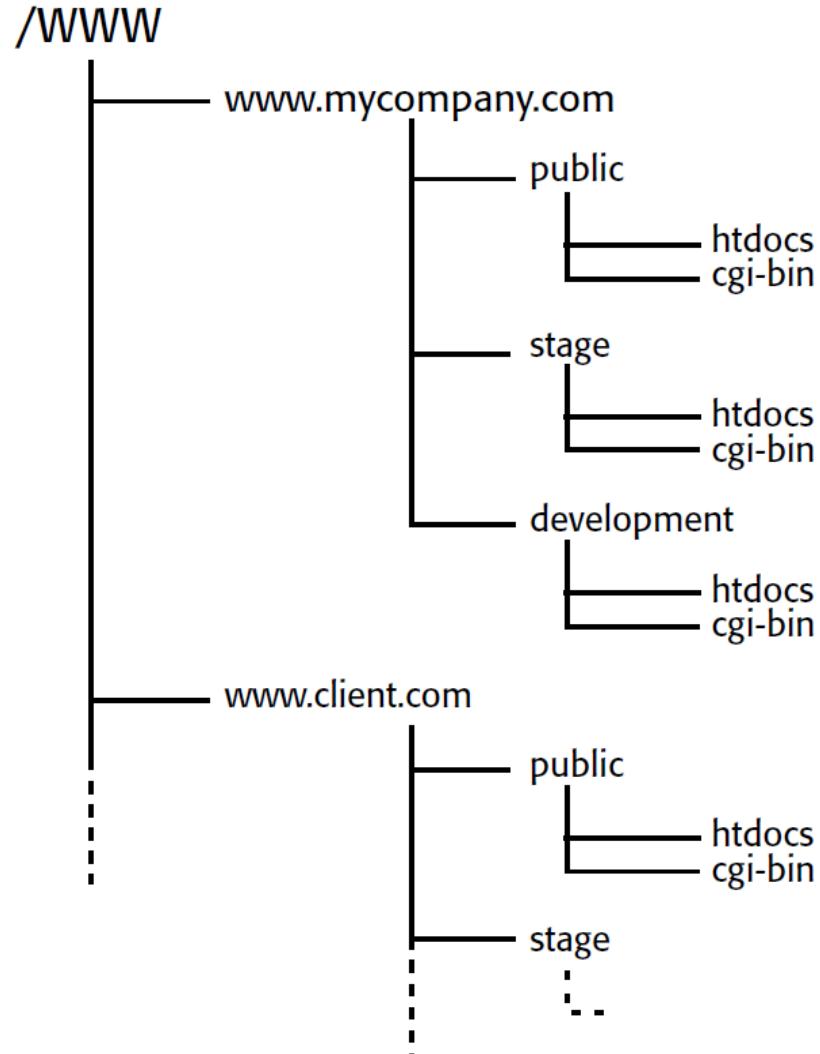
DocumentRoot

- Like all other Web servers, Apache needs to know the path of the top-level directory where Web pages will be kept.
- This directory is typically called the document root directory. Apache provides a directive called DocumentRoot, which can be used to specify the path of the top-level Web directory.
- This directive instructs the server to treat the supplied directory as the root directory for all documents. This is a very important decision for you to make.
- For example, if the directive is set as:
- DocumentRoot / then every file on the system becomes accessible by the Web server

DocumentRoot

- You can protect files by providing proper file permission settings, but setting the document root to the physical root directory of your system is definitely a major security risk.
- Instead, you should point the document root to a specific subdirectory of your file system.
- If you have used the --prefix=/usr/local/apache option in configuring the Apache source, this directive will be set as:
- DocumentRoot “/usr/local/apache/htdocs”

Preferred Web Directory Structure



Preferred Web Directory Structure

- I chose to create a partition called /www, and under it there are subdirectories for each Web site hosted by my system.
- /www/www.mycompany.com/ has three subdirectories: public, stage, and development.
- Each of these subdirectories has two subdirectories: htdocs and cgi-bin.
- The htdocs subdirectory is the document root directory, and the cgi-bin subdirectory is used for CGI scripts.
- So, the DocumentRoot setting for the www.mycompany.com Web site is:
- DocumentRoot
“/www/www.mycompany.com/public/htdocs”

Preferred Web Directory Structure

- The advantage of this directory structure is that it keeps all Web documents and applications under one partition (/www).
- This enables easy backups, and the partition can be mounted on different systems via the Network File System (NFS) in case another machine in the network is given the task to provide Web services.

Directory container directives

- The next set of directives are enclosed in a <Directory . . .> container as
- shown here:
- <Directory />
- Options FollowSymLinks
- AllowOverride None
- </Directory>
- The scope of the enclosed directives is limited to the named directory (with any subdirectories); however, you may only use directives that are allowed in a directory context.

Directory container directives

- Here the Options and the AllowOverride directives apply to %DocumentRoot% that is root (/) or the top-level directory of the main Web site.
- Because directives enclosed within a directory container apply to all the subdirectories below the named directory, the directives apply to all directories within %DocumentRoot%.

Directory container directives

- The Options directive is set to FollowSymLinks, which tells Apache to allow itself to traverse any symbolic within %DocumentRoot%.
- Because the Options directive is only set to follow symbolic links, no other options are available to any of the directories within %DocumentRoot%.
- Effectively, the Options directive is:
- Options FollowSymLinks -ExecCGI -Includes -Indexes -MultiViews

Apache Rocks On

- Apache runs on Linux and other Unix systems.
- Apache also runs on Windows.
- Installing Apache
- To install Apache, install the latest meta-package apache2 by running:
 - sudo apt update
 - sudo apt install apache2
 - Adjusting Firewall
 - sudo ufw app list**
 - sudo ufw allow 'apache'**

Apache Rocks On

- Apache runs on Linux and other Unix systems.
- Apache also runs on Windows.
- Installing Apache
- To install Apache, install the latest meta-package apache2 by running:
 - sudo apt update
 - sudo apt install apache2
 - Adjusting Firewall
 - sudo ufw app list**
 - sudo ufw allow 'apache'**

Apache Rocks On

- sudo ufw status
- Sudo whereis apache2
- sudo service apache2 start
- sudo service --status-all
- sudo apache2ctl start
- sudo vi /etc/apache2/ports.conf //to change port number
- #to suppress warning when we start server
- #add this in ports.conf
- **AcceptFilter https none**
- **AcceptFilter http none**

Apache Rocks On

```
eswaribala@DESKTOP-55AGI0I: /  
-bash: cd: httpd: No such file or directory  
eswaribala@DESKTOP-55AGI0I:/usr/sbin$ cd ..  
eswaribala@DESKTOP-55AGI0I:/usr$ cd ..  
eswaribala@DESKTOP-55AGI0I:$ /etc/httpd/conf/httpd.conf  
-bash: /etc/httpd/conf/httpd.conf: No such file or directory  
eswaribala@DESKTOP-55AGI0I:$ cd /etc/httpd/conf/httpd.conf  
-bash: cd: /etc/httpd/conf/httpd.conf: No such file or directory  
eswaribala@DESKTOP-55AGI0I:$ vi /etc/hosts  
eswaribala@DESKTOP-55AGI0I:$ whereis apache2  
apache2: /usr/sbin/apache2 /usr/lib/apache2 /etc/apache2 /usr/share/apache2 /usr/share/man/man8/apache2.8.gz  
eswaribala@DESKTOP-55AGI0I:$
```

Apache Rocks On

- sudo service apache2 stop
- sudo service apache2 restart
- sudo service apache2 reload (after config changes)
- sudo apachectl -t (test config changes)
- sudo apachectl configtest
- sudo apache2 –v
- Sudo apache2ctl –S (to find errors in conf)

localhost:76

Apps Projects Gmail YouTube Maps Pluralsight

Apache2 Ubuntu Default Page



ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   '-- ports.conf
|-- mods-enabled
|   '-- *.Load
|   '-- *.conf
|-- conf-enabled
|   '-- *.conf
|-- sites-enabled
|   '-- *.conf
```

List Important configuration files

- `$ ls -lh /usr/lib/apache2/modules/` [To List Available Modules]
- `$ ls -lh /etc/apache2/mods-available/` [To List Available Modules]
- `$ ls -lh /etc/apache2/mods-enabled/` [To List Enabled Modules]
- `$ ls -lh /var/www/html` [Apache2 Default Web root]
- `$ ls -lh /var/www/` [Other Website Web root]
- `$ less /etc/apache2/apache2.conf` [Apache2 Main Configuration File]
- `$ less /etc/apache2/ports.conf` [Apache2 Ports Configuration File]
- `$ less /var/log/apache2/error.log` [Apache2 Error Log File]
- `$ less /var/log/apache2/access.log` [Apache2 Access Log File]

Modify Default Sites

- \$ sudo mkdir -p /var/www/test.com
- \$ cd /var/www/test.com
- \$ sudo nano index.html
- <head>
- <title>Test.com index page</title>
- </head>
- <h1>Hello, welcome to test.com! It works!</h1>
- <h2>That is all I have to say. If you don't see this then it doesn't work.</h2>
- </body>
- </html>

Modify Default Sites

- \$ cd /etc/apache2/sites-available/
- \$ sudo nano test.com.conf
- <VirtualHost *:80>
 - ServerAdmin carla@localhost
 - DocumentRoot /var/www/test.com
 - ServerName test.com
 - ServerAlias www.test.com
 - ErrorLog \${APACHE_LOG_DIR}/error.log
 - CustomLog \${APACHE_LOG_DIR}/access.log combined
- </VirtualHost>

Modify Default Sites

- \$ sudo a2ensite test.com.conf
- Enabling site test.com.
- To activate the new configuration, you need to run:
 - service apache2 reload

Configuration

- Apache2 is configured by placing directives in plain text configuration files.
- These directives are separated between the following files and directories:
 - **/etc/apache2**: The Apache configuration directory.
 - All of the Apache configuration files reside here.
 - **/etc/apache2/apache2.conf**: The main Apache configuration file.
 - This can be modified to make changes to the Apache global configuration.
 - This file is responsible for loading many of the other files in the configuration directory.

Configuration

- **httpd.conf**: historically the main Apache2 configuration file, named after the httpd daemon. **In other distributions (or older versions of Ubuntu), the file might be present.** In Ubuntu, **all configuration options have been moved to apache2.conf** and the below referenced directories, and this file no longer exists.

Configuration

- **/etc/apache2/ports.conf:** This file specifies the ports that Apache will listen on.
 - By default, Apache listens on port 80 and additionally listens on port 443 when a module providing SSL capabilities is enabled.
- **/etc/apache2/sites-available/:** The directory where per-site virtual hosts can be stored.
 - Apache will not use the configuration files found in this directory unless they are linked to the sites-enabled directory.
 - Typically, all server block configuration is done in this directory, and then enabled by linking to the other directory with the a2ensite command.

Configuration

- **/etc/apache2/sites-enabled/**: The directory where enabled per-site virtual hosts are stored.
 - Typically, these are created by linking to configuration files found in the sites-available directory with the a2ensite.
 - Apache reads the configuration files and links found in this directory when it starts or reloads to compile a complete configuration.
- **/etc/apache2/conf-available/, /etc/apache2/conf-enabled/**: These directories have the same relationship as the sites-available and sites-enabled directories, but are used to store configuration fragments that do not belong in a virtual host.
 - Files in the conf-available directory can be enabled with the a2enconf command and disabled with the a2disconf command.

Configuration

- **conf-available**: this directory contains available configuration files. All files that were previously in `/etc/apache2/conf.d` should be moved to `/etc/apache2/conf-available`.
- **conf-enabled**: holds symlinks to the files in `/etc/apache2/conf-available`. When a configuration file is symlinked, it will be enabled the next time apache2 is restarted.
- **envvars**: file where Apache2 environment variables are set.

Configuration

- **/etc/apache2/mods-available/, /etc/apache2/mods-enabled/:** These directories contain the available and enabled modules, respectively.
 - Files in ending in .load contain fragments to load specific modules, while files ending in .conf contain the configuration for those modules.
 - Modules can be enabled and disabled using the a2enmod and a2dismod command.

Configuration

- **mods-enabled**: holds symlinks to the files in /etc/apache2/mods-available. When a module configuration file is symlinked it will be enabled the next time apache2 is restarted.
- **ports.conf**: houses the directives that determine which TCP ports Apache2 is listening on.
- **magic**: instructions for determining MIME type based on the first few bytes of a file.

Server Logs

- **/var/log/apache2/access.log**: By default, every request to your web server is recorded in this log file unless Apache is configured to do otherwise.
- **/var/log/apache2/error.log**: By default, all errors are recorded in this file. The LogLevel directive in the Apache configuration specifies how much detail the error logs will contain.

Apache2 Options

```
eswaribala@DESKTOP-55AGI0I:/$ whereis apache2
apache2: /usr/sbin/apache2 /usr/lib/apache2 /etc/apache2 /usr/share/apache2 /usr/share/man/man8/apache2.8.gz
eswaribala@DESKTOP-55AGI0I:/$ apache2 -h
Usage: apache2 [-D name] [-d directory] [-f file]
           [-C "directive"] [-c "directive"]
           [-k start|restart|graceful|graceful-stop|stop]
           [-v] [-V] [-h] [-l] [-L] [-t] [-T] [-S] [-X]
Options:
-D name          : define a name for use in <IfDefine name> directives
-d directory     : specify an alternate initial ServerRoot
-f file          : specify an alternate ServerConfigFile
-C "directive"   : process directive before reading config files
-c "directive"   : process directive after reading config files
-e level         : show startup errors of level (see LogLevel)
-E file          : log startup errors to file
-v               : show version number
-V               : show compile settings
-h               : list available command line options (this page)
-l               : list compiled in modules
-L               : list available configuration directives
-t -D DUMP_VHOSTS : show parsed vhost settings
-t -D DUMP_RUN_CFG : show parsed run settings
-S               : a synonym for -t -D DUMP_VHOSTS -D DUMP_RUN_CFG
-t -D DUMP_MODULES : show all loaded modules
-M               : a synonym for -t -D DUMP_MODULES
-t -D DUMP_INCLUDES: show all included configuration files
-t               : run syntax check for config files
-T               : start without DocumentRoot(s) check
-X               : debug mode (only one worker, do not detach)
eswaribala@DESKTOP-55AGI0I:/$
```

Show All Loaded Modules in Apache 2

Apache2ctl -t -D DUMP_MODULES

```
eswaribala@DESKTOP-55AGI0I:~$ apache2ctl -t -D DUMP_MODULES
Loaded Modules:
core_module (static)
so_module (static)
watchdog_module (static)
http_module (static)
log_config_module (static)
logio_module (static)
version_module (static)
unixd_module (static)
access_compat_module (shared)
alias_module (shared)
auth_basic_module (shared)
authn_core_module (shared)
authn_file_module (shared)
authz_core_module (shared)
authz_host_module (shared)
authz_user_module (shared)
autoindex_module (shared)
deflate_module (shared)
dir_module (shared)
env_module (shared)
filter_module (shared)
mime_module (shared)
mpm_event_module (shared)
negotiation_module (shared)
reqtimeout_module (shared)
setenvif_module (shared)
status_module (shared)
eswaribala@DESKTOP-55AGI0I:~$
```



Show All Loaded Modules in Apache 2

Apache2ctl -M

```
eswaribala@DESKTOP-55AGI0I: ~
setenvif_module (shared)
status_module (shared)
eswaribala@DESKTOP-55AGI0I:~$ apachectl -M
Loaded Modules:
core_module (static)
so_module (static)
watchdog_module (static)
http_module (static)
log_config_module (static)
logio_module (static)
version_module (static)
unixd_module (static)
access_compat_module (shared)
alias_module (shared)
auth_basic_module (shared)
authn_core_module (shared)
authn_file_module (shared)
authz_core_module (shared)
authz_host_module (shared)
authz_user_module (shared)
autoindex_module (shared)
deflate_module (shared)
dir_module (shared)
env_module (shared)
filter_module (shared)
mime_module (shared)
mpm_event_module (shared)
negotiation_module (shared)
reqtimeout_module (shared)
setenvif_module (shared)
status_module (shared)
eswaribala@DESKTOP-55AGI0I:~$
```

Show list of compiled modules

```
eswaribala@DESKTOP-55AGI0I:~$ apachectl -l
Compiled in modules:
core.c
mod_so.c
mod_watchdog.c
http_core.c
mod_log_config.c
mod_logio.c
mod_version.c
mod_unixd.c
eswaribala@DESKTOP-55AGI0I:~$
```

Go to Modules Folder

```
eswaribala@DESKTOP-55AGI0I:/usr/lib/apache2/modules$ cd /usr/lib/apache2/modules
eswaribala@DESKTOP-55AGI0I:/usr/lib/apache2/modules$ ls
httpd.exp
mod_access_compat.so
mod_actions.so
mod_alias.so
mod_allowmethods.so
mod_asis.so
mod_auth_basic.so
mod_auth_digest.so
mod_auth_form.so
mod_authn_anon.so
mod_authn_core.so
mod_authn_dbd.so
mod_authn_dbm.so
mod_authn_file.so
mod_authn_socache.so
mod_authnz_fcgi.so
mod_authnz_ldap.so
mod_authz_core.so
mod_authz_dbd.so
mod_authz_dbm.so
mod_authz_groupfile.so
mod_authz_host.so
mod_authz_owner.so
mod_authz_user.so
mod_autoindex.so
mod_brotli.so
mod_bucketeer.so
mod_buffer.so
mod_cache.so
mod_cache_disk.so
mod_cache_socache.so
mod_case_filter.so
mod_case_filter_in.so
mod_cern_meta.so
mod_cgi.so
mod_cgid.so
mod_charset_lite.so
mod_data.so
mod_dav.so
mod_dav_fs.so
mod_dav_lock.so
mod_dbd.so
mod_deflate.so
mod_dialup.so
mod_dir.so
mod_dumpio.so
mod_echo.so
mod_env.so
mod_expires.so
mod_ext_filter.so
mod_file_cache.so
mod_filter.so
mod_headers.so
mod_heartbeat.so
mod_heartmonitor.so
mod_ident.so
mod_imagemap.so
mod_include.so
mod_info.so
mod_lbmethod_bybusyness.so
mod_lbmethod_byrequests.so
mod_lbmethod_bytraffic.so
mod_lbmethod_heartbeat.so
mod_ldap.so
mod_log_debug.so
mod_log_forensic.so
mod_lua.so
mod_macro.so
mod_md.so
mod_mime.so
mod_mime_magic.so
mod_mpm_event.so
mod_mpm_prefork.so
mod_mpm_worker.so
mod_negotiation.so
mod_proxy.so
mod_proxy_ajp.so
mod_proxy_balancer.so
mod_proxy_connect.so
mod_proxy_express.so
mod_proxy_fcgi.so
mod_proxy_fdpass.so
mod_proxy_ftp.so
mod_proxy_hcheck.so
mod_proxy_html.so
mod_proxy_http.so
mod_proxy_http2.so
mod_proxy_scgi.so
mod_proxy_uwsgi.so
mod_proxy_wstunnel.so
mod_ratelimit.so
mod_reflector.so
mod_remoteip.so
mod_reqtimeout.so
mod_request.so
mod_rewrite.so
mod_sed.so
mod_session.so
mod_session_cookie.so
mod_session_crypto.so
mod_session_dbd.so
mod_setenvif.so
mod_slotmem_plain.so
mod_slotmem_shm.so
mod_socache_dbm.so
mod_socache_memcache.so
mod_socache_redis.so
mod_socache_shmcbs.so
mod_speling.so
mod_ssl.so
mod_status.so
mod_substitute.so
mod_suexec.so
mod_unique_id.so
mod_userdir.so
mod_usertrack.so
mod_vhost_alias.so
mod_xml2enc.so
eswaribala@DESKTOP-55AGI0I:/usr/lib/apache2/modules$ ■
```

Basic Settings

- Apache2 ships with a virtual-host-friendly default configuration.
- That is, it is configured with a single default virtual host (using the Virtual Host directive) which can be modified or used as-is if you have a single site, or used as a template for additional virtual hosts if you have multiple sites.
- If left alone, the default virtual host will serve as your default site, or the site users will see if the URL they enter does not match the ServerName directive of any of your custom sites.
- To modify the default virtual host, edit the file /etc/apache2/sites-available/000-default.conf.

Method-1: How to find the Process ID (PID) of a program running on Linux using the pidof Command



- The pidof command is used to find the process ID of the running program.
- It prints those IDs into the standard output.
- To demonstrate this, we are going to find the Apache2 process id from Ubuntu 20 system.

```
eswaribala@DESKTOP-55AGI0I: ~
eswaribala@DESKTOP-55AGI0I:~$ pidof apache2
108 107 105
eswaribala@DESKTOP-55AGI0I:~$ -
```

Method-1: How to find the Process ID (PID) of a program running on Linux using the pidof Command



- In the above output you may have difficulties identifying the process ID because it displays all PIDs (including parent and child) against the process name.
- So we need to find the Parent Process PID (PPID), which is what we are looking for.
- This may be the first number. In my case it is 108 and it is sorted in descending order.

Method-2: How to find the Process ID (PID) of a program running on Linux using the pgrep Command



- The pgrep command looks at the processes currently running and lists the process IDs that match the selection criteria.

```
eswaribala@DESKTOP-55AGI0I: ~
eswaribala@DESKTOP-55AGI0I:~$ pgrep apache2
105
107
108
eswaribala@DESKTOP-55AGI0I:~$
```

Method-3: How to find the Process ID (PID) of a program running on Linux using the pstree Command



- The pstree command shows running processes as a tree.
- The tree is rooted at either pid or init if pid is omitted.
- If a user name is specified in the pstree command then it's shows all the process owned by the corresponding user.
- pstree visually merges identical branches by putting them in square brackets and prefixing them with the repetition count.

Method-3: How to find the Process ID (PID) of a program running on Linux using the pstree Command



- The pstree command shows running processes as a tree.
- The tree is rooted at either pid or init if pid is omitted.
- If a user name is specified in the pstree command then it's shows all the process owned by the corresponding user.
- pstree visually merges identical branches by putting them in square brackets and prefixing them with the repetition count.

To get all the process
pstree -p | grep "apache2"



To get only the parent process

```
pstree -p | grep "apache2" | head -1
```

```
eswaribala@DESKTOP-55AGI0I:~$  
eswaribala@DESKTOP-55AGI0I:~$ pstree -p | grep "apache2" | head -1  
init(1)-+-apache2(105)-+-apache2(107)-+-{apache2}(110)  
eswaribala@DESKTOP-55AGI0I:~$
```

Method-4: How to find the Process ID (PID) of a program running on Linux using the ps Command



- The ps command displays information about a selection of the active processes.
- It displays the process ID (pid=PID), the terminal associated with the process (tname=TTY), the cumulated CPU time in [DD-]hh:mm:ss format (time=TIME), and the executable name (ucmd=CMD). Output is unsorted by default.

ps aux | grep "apache2"

```
eswaribala@DESKTOP-55AGI0I:~$ ps aux | grep "apache2"
root      105  0.0  0.0  14592  3508 ?          Ss   11:07  0:00 /usr/sbin/apache2 -k start
www-data   107  0.0  0.0  2006008 4184 ?          Sl   11:07  0:00 /usr/sbin/apache2 -k start
www-data   108  0.0  0.0  2005984 4224 ?          Sl   11:07  0:00 /usr/sbin/apache2 -k start
eswarib+  1009  0.0  0.0  16208  1292 tty1        S    14:51  0:00 grep --color=auto apache2
eswaribala@DESKTOP-55AGI0I:~$
```

Nano Useful Keys

- move forward one character: Ctrl+F (^F)
- move back one character: Ctrl+B (^B)
- move forward one word: Ctrl+Space (^Space)
- move back one word: Alt+Space (M-Space)
- move to the previous line: Ctrl+P (^P)
- move to the next line: Ctrl+N (^N)
- move to the next page: Ctrl+V (^V)
- move to the previous page: Ctrl+Y (^Y)
- move to the beginning of the line: Ctrl+A (^A)
- move to the end of the line: Ctrl+E (^E)

Nano Useful Keys

- **Search a Text File**
- To search for a particular word or part of a text inside the editor, use the “where is” option with the Ctrl+W shortcut (^W). This will open a search prompt where you can type in the text you want to find. To continue to the next result, use Alt+W (M-W).
- The search bar can also find specific line numbers. Press Ctrl+T (^T) while in it and the line number you want to find.

Nano Useful Keys

- Replace Text
- To replace text in the file, first open the search bar with **Ctrl+W (^W)** and then press **Ctrl+R (^R)**. It will open a search bar to type in what you want to replace, as seen in the image below.

Select, Copy, Cut and Paste Text

- To select part of a file, navigate to the beginning of the text, press the Alt+A shortcut (M-A) and use the arrow keys to move over the text you wish to select.
- Next, you can copy the selected text with the Alt+6 combination (M-6) or cut with Ctrl+K (^K). If you use these shortcuts without selecting any text prior, it will copy or cut the entire line of text.
- To paste text, use Ctrl+U (displayed as ^U).

Save a File

- To save a file, use the Ctrl+O (^O) keyboard combination. It will ask you to enter a file name or confirm the name of an existing file.

Apache Performance Tuning: Swap Memory

- Before we get into the nitty-gritty of Apache tuning, we need to understand what happens when a VPS server or Dedicated server goes unresponsive due to a poorly optimized configuration.
- An over-tuned server is one that is configured to allow more simultaneous requests (`ServerLimit`) than the server's hardware can manage.
- Servers set in this manner have a tipping point, once reached, the server will become stuck in a perpetual swapping scenario.

Apache Performance Tuning: Swap Memory

- Meaning the Kernel is stuck rapidly reading and writing data to and from the system swap file.
- Swap files have read/write access speeds vastly slower than standard memory space.
- The swap files' latency causes a bottleneck on the server as the Kernel attempts to read and write data faster than is physically possible or more commonly known as thrashing.

Apache Performance Tuning: Swap Memory

- If not caught immediately, thrashing spirals the system out of control leading to a system crash.
- If thrashing is left running for too long, it has the potential of physically harming the hard drive itself by simulating decades of read/write activity over a short period.
- When optimizing Apache, we must be cautious not to create a thrashing scenario.
- We can accomplish this by calculating the thrashing point of the server based on several factors.

Estimate the Thrashing Point

- Calculating the estimated thrashing point or ServerLimit of a server uses a simple equation:

$$\text{(buff/cache} - \text{Reserved})/\text{Avg.Apache}$$

- buff/cache: The total memory used by the Kernel for buffers and cache.
- Reserved: The amount of memory reserved for non-Apache processes.
- Available: The difference between buff/cache and Reserved memory.
- Avg.Apache: The average of all running Apache children during peak operational hours.

Estimate the Thrashing Point

- The thrashing point value is equal to the number Apache children the server can run; this applies to either threaded or non-thread children.
- When the number of children running in memory meets the calculated thrashing point, the server will begin to topple.

Buffer/Cache Memory

- On modern Linux systems, the buffer/cache can be derived using the /proc/meminfo file by adding the Buffers, Cached and Slab statistics.
- Using the free command, we can quickly grab this information, as in the example below:

```
168 167 165
eswaribala@DESKTOP-55AGI0I:~$ free
              total        used        free      shared  buff/cache   available
Mem:    16607084     8063152     8314580          17720      229352     8410200
Swap: 29255396     176732    29078664
eswaribala@DESKTOP-55AGI0I:~$ ■
```

Reserved Memory

- Reserved memory is a portion of memory held for other services aside from Apache.
- Some of the biggest contenders for additional memory outside of Apache are MySQL, Tomcat, Memcache, Varnish, and Nginx.
- It is necessary to examine these services configs to determine a valid reserved memory.
-  Rule-of-Thumb:
 - Save 25% of the total buff/cache memory for any extra services ran on the server.

Average Apache Memory

- Finding the average size of Apache processes is relatively simple using the ps command to list the RSS (Resident Set Size) of all running httpd processes.
- This example uses a short awk script to print out the average instead of listing the sizes.
- ```
ps -yIC apache2 | awk '{x += $8;y += 1} END {print "Process Memory Usage (MB): "x/1024; print "Average Process Size (MB): "x/((y-1)*1024); print "Total Number of Processes: "(y-1)}'
```

# Average Apache Memory

```
eswaribala@DESKTOP-55AGI0I:~$ ps -ylC apache2 | awk '{x += $8;y += 1} END {print "Process Memory Usage (MB): "x/1024; print "Average Pr
occess Size (MB): "x/((y-1)*1024); print "Total Number of Processes: "(y-1)}'
Process Memory Usage (MB): 7.43359
Average Procces Size (MB): 2.47786
Total Number of Processes: 3
eswaribala@DESKTOP-55AGI0I:~$
```

**Thrashing point =  $6132308/536865=11.4$**

# Average Apache Memory

- ps o rss -C apache2

⌚ eswaribala@DESKTOP-55AGI0I:/etc/apache2/sites-available

```
Total Number of Processes: 8
eswaribala@DESKTOP-55AGI0I:/etc/apache2/sites-available$ ps o rss -C apache2
 RSS
17140
4420
3644
4576
4120
3360
3360
3360
eswaribala@DESKTOP-55AGI0I:/etc/apache2/sites-available$ █
```

# Calculate the Thrashing Point

- Once collected divided the Available memory by Avg. Apache, rounding down to the nearest whole number. Available memory is the buff/cache memory minus the Reserved memory.

| Variable        | KB     | Notes                                           |
|-----------------|--------|-------------------------------------------------|
| buffer/cache    | 708436 | Gathered from the <b>free</b> command           |
| Reserved        | 177109 | <b>Rule-of-thumb:</b> up to 25% of buffer/cache |
| Available       | 531327 | <b>buffer/cache</b> minus <b>Reserved</b>       |
| Avg Apache      | 22200  | Obtained using the <b>ps</b> command            |
| Thrashing Point | 23     | Available divided by Avg Apache (round down)    |

The table summarizes how to derived each variable to calculate an estimated thrashing point.

# Calculate the Thrashing Point

| General Thrashing Point Estimates |          |        |          |
|-----------------------------------|----------|--------|----------|
| Memory                            | Requests | Memory | Requests |
| 2 GB                              | 23       | 12GB   | 140      |
| 4 GB                              | 46       | 16 GB  | 187      |
| 6 GB                              | 70       | 24 GB  | 281      |
| 8 GB                              | 93       | 32 GB  | 374      |

Conservative estimates based on server memory, assuming ~80% of memory available and a base size for Apache children instance of 64k.

# Apache Performance Tuning: MPM Modules

- The keystone for understanding Apache server performance is by far the Multiprocessing Modules (MPMs).
- These modules determine the basis for how Apache addresses multiprocessing.
- Multiprocessing means running multiple operations simultaneously in a system with multiple central processing units (CPU Cores).

# Apache Performance Tuning: MPM Modules

---

- These modules are:
  - MPM Prefork
  - MPM Worker
  - MPM Event
  - Other MPMS

# Apache Performance Tuning: MPM Modules

-  MPM Prefork

Avoid using MPM Prefork whenever possible. Its inability to scale well with increased traffic will quickly outpace the available hardware on most system configurations.

# Apache Performance Tuning: MPM Modules

---

-  MPM Worker

- A hybrid pre-forking, multi threaded, multiprocessing web server.
- In the same fashion as MPM Prefork, MPM Worker uses the same approach with a single master parent process governing all children within its server pool.
- MPM Worker has set the foundation for multi threaded multiprocessing in Apache servers which became stable in Apache 2.2.

# Apache Performance Tuning: MPM Modules

-  MPM Worker

- The threaded configuration allows Apache to service hundreds of requests with ease while retaining only a dozen or so child processes in memory.
- The MPM Worker make for both a high capacity and low resource solution for web service.
- The KeepAliveTimeOut directive currently defines the amount of time Apache will wait for requests.
- When utilizing KeepAlive with MPM Worker use the smallest KeepAliveTimeout as possible (1 second preferably).

# Apache Performance Tuning: MPM Modules

-  MPM Event

- Based off the MPM Worker source code, MPM Event shares configuration directives with MPM Worker.
- It works nearly identical to MPM Worker except when it comes to handling KeepAlive requests.
- MPM Event uses a dedicated Listener thread in each child process.
- This Listening thread is responsible for directing incoming requests to an available worker thread.
- The Listening thread solves the issue encountered by MPM Worker which locks entire threads into waiting for the KeepAliveTimeout.

# Apache Performance Tuning: MPM Modules

-  MPM Event

- The Listener approach of MPM Event ensures worker threads are not “stuck” waiting for KeepAliveTimeout to expire.
- This method keeps the maximum amount of worker threads handling as many requests as possible.
- MPM Event is stable in Apache 2.4, older versions can use MPM Worker as an alternative

# Apache Performance Tuning: MPM Modules

- Which MPM is the best?
- When considering optimization, it is essential to understand there is no such thing as a one-size-fits-all Apache configuration.
- Correctly choosing an MPM requires analysis of many moving variables like traffic, site code, server type, PHP Handler and available hardware.
- Every server is unique making the best MPM an entirely subjective choice.

# Apache Performance Tuning: MPM Modules

---

- If your application code does not support multi-threading, then your choice will inevitably be MPM Prefork purely on a compatibility basis.
- MPM Prefork includes software modules like mod\_php (DSO).
- MPM Worker without KeepAlive performs very well if your application is a high-performance load balanced API system.
- The scalability and flexibility of MPM Event is a solid choice for hosting multiple small to medium sites in a shared hosting configuration.

# Apache Performance Tuning: MPM Modules

---

- Most simple servers setups operate well under the self-governing default configuration of MPM Event, making it an ideal starting point for optimization tuning.
- Once chosen, an MPM can then move onto Configuration Directives to review which settings pertain to server performance and optimization.

## View installed modules

- You can determine which MPM Apache2 is currently built with by executing apache2 -l

```
eswaribala@DESKTOP-55AGI0I: ~
eswaribala@DESKTOP-55AGI0I:~$ sudo apache2 -l
Compiled in modules:
 core.c
 mod_so.c
 mod_watchdog.c
 http_core.c
 mod_log_config.c
 mod_logio.c
 mod_version.c
 mod_unixd.c
eswaribala@DESKTOP-55AGI0I:~$
```

## View installed modules

---

- This command will list all modules, both static and shared:
- MPMs can be built as static modules on all platforms.
- `sudo apachectl -M`

# View installed modules

```
eswaribala@DESKTOP-55AGI0I:/$ sudo apache2ctl -M
[sudo] password for eswaribala:
Loaded Modules:
core_module (static)
so_module (static)
watchdog_module (static)
http_module (static)
log_config_module (static)
logio_module (static)
version_module (static)
unixd_module (static)
access_compat_module (shared)
alias_module (shared)
auth_basic_module (shared)
authn_core_module (shared)
authn_file_module (shared)
authz_core_module (shared)
authz_host_module (shared)
authz_user_module (shared)
autoindex_module (shared)
deflate_module (shared)
dir_module (shared)
env_module (shared)
filter_module (shared)
mime_module (shared)
mpm_event_module (shared)
negotiation_module (shared)
reqtimeout_module (shared)
setenvif_module (shared)
status_module (shared)
eswaribala@DESKTOP-55AGI0I:/$
```

# View installed modules

eswaribala@DESKTOP-55AGI0I: /usr/lib/apache2/modules

```
core_module
eswaribala@DESKTOP-55AGI0I:/usr/lib/apache2/modules$ ls
httpd.exp
mod_access_compat.so
mod_actions.so
mod_alias.so
mod_allowmethods.so
mod_asis.so
mod_auth_basic.so
mod_auth_digest.so
mod_auth_form.so
mod_authn_anon.so
mod_authn_core.so
mod_authn_dbd.so
mod_authn_dbm.so
mod_authn_file.so
mod_authn_socache.so
mod_authnz_fcgi.so
mod_authnz_ldap.so
mod_authz_core.so
mod_authz_dbd.so
mod_authz_dbm.so
mod_authz_groupfile.so
mod_authz_host.so
mod_authz_owner.so
mod_authz_user.so
mod_autoindex.so
mod_brotli.so
mod_bucketeer.so
mod_buffer.so
mod_cache.so
mod_cache_disk.so
mod_cache_socache.so
mod_case_filter.so
mod_case_filter_in.so
mod_cern_meta.so
mod_cgi.so
mod_cgid.so
mod_charset_lite.so
mod_data.so
mod_dav.so
mod_dav_fs.so
mod_dav_lock.so
mod_dbd.so
mod_deflate.so
mod_dialup.so
mod_dir.so
mod_dumpio.so
mod_echo.so
mod_env.so
mod_expires.so
mod_ext_filter.so
mod_file_cache.so
mod_filter.so
mod_headers.so
mod_heartbeat.so
mod_heartmonitor.so
mod_http2.so
mod_ident.so
mod_imagemap.so
mod_include.so
mod_info.so
mod_lbmethod_bybusyness.so
mod_lbmethod_byrequests.so
mod_lbmethod_bytraffic.so
mod_lbmethod_heartbeat.so
mod_ldap.so
mod_log_debug.so
mod_log_forensic.so
mod_lua.so
mod_macro.so
mod_md.so
mod_mime.so
mod_mime_magic.so
mod_mpms_event.so
mod_mpms_prefork.so
mod_mpms_worker.so
mod_negotiation.so
mod_proxy.so
mod_proxy_ajp.so
mod_proxy_balancer.so
mod_proxy_connect.so
mod_proxy_express.so
mod_proxy_fcg.i.so
mod_proxy_fdpass.so
mod_proxy_ftp.so
mod_proxy_hcheck.so
mod_proxy_html.so
mod_proxy_http.so
mod_proxy_http2.so
mod_proxy_scgi.so
mod_proxy_uwsgi.so
mod_proxy_wstunnel.so
mod_ratelimit.so
mod_reflector.so
mod_remoteip.so
mod_reqtimeout.so
mod_request.so
mod_rewrite.so
mod_sed.so
mod_session.so
mod_session_cookie.so
mod_session_crypto.so
mod_session_dbd.so
mod_setenvif.so
mod_slotmem_plain.so
mod_slotmem_shm.so
mod_socache_dbm.so
mod_socache_memcache.so
mod_socache_redis.so
mod_socache_shmcb.so
mod_speling.so
mod_ssl.so
mod_status.so
mod_substitute.so
mod_suexec.so
mod_unique_id.so
mod_userdir.so
mod_usertrack.so
mod_vhost_alias.so
mod_xml2enc.so
eswaribala@DESKTOP-55AGI0I:/usr/lib/apache2/modules$
```

# Apache Performance Tuning: MPM Modules

- **Enable/Disable Modules**

```
eswaribala@DESKTOP-55AGI0I: /$ eswaribala@DESKTOP-55AGI0I: /$ sudo a2enmod rewrite
Module rewrite already enabled
eswaribala@DESKTOP-55AGI0I: /$ sudo service apache2 restart
 * Restarting Apache httpd web server apache2
sleep: cannot read realtime clock: Invalid argument

eswaribala@DESKTOP-55AGI0I: /$ sudo a2dismod rewrite
Module rewrite disabled.
To activate the new configuration, you need to run:
 service apache2 restart
eswaribala@DESKTOP-55AGI0I: /$ sudo service apache2 reload
 * Reloading Apache httpd web server apache2
 *
eswaribala@DESKTOP-55AGI0I: /$
```

# Apache Performance Tuning: MPM Modules

- **Check Modules Status**

```
⌚ eswaribala@DESKTOP-55AGI0I: /
*
eswaribala@DESKTOP-55AGI0I:/$ sudo a2query -m rewrite
No module matches rewrite (disabled by site administrator)
eswaribala@DESKTOP-55AGI0I:/$ ■
```

# Apache Performance Tuning: MPM Modules

- **List Apache2 Modules**

```
eswaribala@DESKTOP-55AGI0I: /
```

```
eswaribala@DESKTOP-55AGI0I:/$ sudo a2query -m
access_compat (enabled by maintainer script)
alias (enabled by maintainer script)
auth_basic (enabled by maintainer script)
authn_core (enabled by maintainer script)
authn_file (enabled by maintainer script)
authz_core (enabled by maintainer script)
authz_host (enabled by maintainer script)
authz_user (enabled by maintainer script)
autoindex (enabled by maintainer script)
deflate (enabled by maintainer script)
dir (enabled by maintainer script)
env (enabled by maintainer script)
filter (enabled by maintainer script)
mime (enabled by maintainer script)
mpm_event (enabled by maintainer script)
negotiation (enabled by maintainer script)
reqtimeout (enabled by maintainer script)
setenvif (enabled by maintainer script)
status (enabled by maintainer script)
eswaribala@DESKTOP-55AGI0I:/$ ■
```

# How To Configure Apache HTTP with MPM Event and PHP-FPM



- **Step 1 — Changing the Multi-Processing Module**
  - sudo service apache2 stop
  - sudo a2dismod php7.2
  - sudo a2dismod mpm\_prefork
  - sudo a2enmod mpm\_event
- **Step 2 — Configuring Apache HTTP to Use the FastCGI Process Manager**
  - sudo apt install php-fpm
  - sudo apt install libapache2-mod-fcgid (httpd to php commn lib)

# How To Configure Apache HTTP with MPM Event and PHP-FPM



- **Step 3 — Configuring Apache HTTP to Use the FastCGI Process Manager**
  - sudo a2enconf php7.4-fpm
  - sudo a2enmod proxy
  - sudo a2enmod proxy\_fcgi
  - cat /etc/apache2/conf-enabled/php7.4-fpm.conf
  - sudo apachectl configtest
  - sudo service apache2 restart

# How To Configure Apache HTTP with MPM Event and PHP-FPM

---



- **Step 3 — Checking Your Configuration**

- sudo apachectl -M | grep 'mpm'
- sudo apachectl -M | grep 'proxy'
- sudo nano /var/www/phpsite.com/info.php
- sudo service php7.4-fpm start
- <http://localhost:84/info.php>

# Apache Performance Tuning: MPM Directives

- **General Optimization**
- IfModule
  - An important directive to learn when working with Apache servers is the IfModule conditional statement.
  - There are two parts to the IfModule statement.
  - A beginning, which also accepts a module name or module source file name, as well as a closing statement.
  - When the provided module is loaded into Apache, then all directives between the beginning IfModule statement and the closing IfModule statement are also read into the Apache running configuration.

# Apache Performance Tuning: MPM Directives

- <ifModule mpm\_prefork\_module>
- MaxSpareServers 16
- </ifModule>
- Timeout 60

# Apache Performance Tuning: MPM Directives

- IfModule statements are used to maintain compatibility within Apache configuration between module changes.
- Maintaining compatibility is done by grouping directives into IfModule statements, so they are only used when the required module is loaded.
- Ensuring a syntactically correct configuration file even when swapping modules.
- Appropriately wrapping everything in an IfModule statement is a best practice standard with Apache and should be adhered to for superior compatibility in config files.

# Apache Performance Tuning: MPM Directives

- Timeout
  - The numerical value of seconds Apache waits for all common I/O events.
  - Apache will abandon requests fail to complete before the provided Timeout value.
  - Determining the right Timeout depends on both traffic habits and hosted applications.
  - Ideally, Timeout should be as low as possible while still allowing the vast majority of regular traffic to operate without issue.
  - Large timeouts, those above 1 minute, open the server to SlowLoris style DOS attacks and foster a long wait in the browser when it encounters a problem.
  - Lower timeouts allow Apache to recover from errant stuck connections quickly.

# Apache Performance Tuning: MPM Directives

- Timeout
  - It becomes necessary to strike a balance between the two extremes.

**Tip:**

| Timeout |                         |
|---------|-------------------------|
| Syntax  | Timeout <i>number</i>   |
| Default | 60                      |
| Modules | core/none               |
| Doc     | <a href="#">Timeout</a> |

Avoid increasing the global *Timeout* when addressing issues with a single script, or user, that requires a long *Timeout*. Problems can usually be resolved by a .htaccess file or include file to increase the *Timeout* directive for that specific script.

# Apache Performance Tuning: MPM Directives

- KeepAlive
- A simple on|off toggle enables the KeepAlive protocols with supported browsers.
- The KeepAlive feature can provide as much as a 50% reductions in latency, significantly boosting the performance of Apache.
- KeepAlive accomplishes this by reusing the same initial connections a browser creates when connecting to Apache for all follow-up requests which occur within a short period.

# Apache Performance Tuning: MPM Directives

- KeepAlive is a powerful feature and in general, should be enabled in most situations.
- It works great for reducing some of the CPU and Network overhead with modern element heavy websites.
- For example, an easy way to visualize KeepAlive is with the “**hold the door**” phrase.
- Imagine a queue of people entering a building through a single doorway.
- Each person is required to open the door, walk through it, then close the door before the next person does the same process.
- Mostly, that’s how Apache works without KeepAlive.
- When enabled, the door stays open until all the people in line are through the door before it closes again.

# Apache Performance Tuning: MPM Directives

## KeepAlive

| KeepAlive      |                                  |
|----------------|----------------------------------|
| <b>Syntax</b>  | KeepAlive on off                 |
| <b>Default</b> | on                               |
| <b>Modules</b> | core/none                        |
| <b>Doc</b>     | <a href="#"><u>KeepAlive</u></a> |

# Apache Performance Tuning: MPM Directives

- MaxKeepAliveRequests
- Sets a limit on the number of requests an individual KeepAlive connection is permitted to handle.
- Once reached, Apache forces the connection to terminate, and creates a new one for any additional requests.
- Determining an ideal setting here is open to interpretation.
- Generally, you want this value to be at least as high as the largest count of elements (HTML, Text, CSS, Images, Etc..) served by the most heavily trafficked pages on the server.

# Apache Performance Tuning: MPM Directives

## Rule-of-Thumb:

| <b>MaxKeepAliveRequests</b> |                                      |
|-----------------------------|--------------------------------------|
| <b>Syntax</b>               | MaxKeepAliveRequests number          |
| <b>Default</b>              | 100                                  |
| <b>Modules</b>              | core/none                            |
| <b>Doc</b>                  | <a href="#">MaxKeepAliveRequests</a> |

Set *MaxKeepAliveRequests* to double that of the largest count of elements on common pages. (Services like [webpagetest.org](http://webpagetest.org) or [gtmetrix.com](http://gtmetrix.com) can count elements on a page).

# Apache Performance Tuning: MPM Directives

- KeepAliveTimeout
- This directive is measured in seconds and will remain idle waiting for additional requests from its initiator.
- Since these types of connections are only accessible to their initiator, we want to keep KeepAliveTimeout very low.
- A low value prevents too many KeepAlive connections from locking out new visitors due to connection priority.

# Apache Performance Tuning: MPM Directives

## Tip:

| KeepAliveTimeout |                                  |
|------------------|----------------------------------|
| Syntax           | KeepAliveTimeout number          |
| Default          | 5                                |
| Modules          | core/none                        |
| Doc              | <a href="#">KeepAliveTimeout</a> |

A large *MaxKeepAliveRequests* directive with a very low *KeepAliveTimeout* allows active visitors to reuse connections while also quickly recovering threads from idle visitors.

**Configuration:** Set

*MaxKeepAliveRequests* to 500+, Set *KeepAliveTimeout* to 2

**Requirements:** Works best on MPM

Event.

# MPM Event/Worker Optimization

- The two modules, MPM Event, and MPM Worker for most intents and purposes operate identically.
- The difference is apparent in the way each handles KeepAlive requests.
- The MPM Worker locks threads for the duration of the KeepAlive process and directly affects the number of available threads able to handle new requests.
- The MPM Event uses a Listener thread for each child.
- These Listener threads handle standard requests, and KeepAlive requests alike meaning thread locking will not reduce the capacity of the server.
- Without thread locking, MPM Event is the superior choice but only in Apache 2.4.
- Before Apache 2.4 the MPM Event was unstable and prone to problems.

# MPM Event/Worker Optimization

---

- **ServerLimit**
- ServerLimit represents the upper limit of children Apache is allowed.
- The practical usage for ServerLimit is creating a hard ceiling in Apache to protect against input errors with MaxRequestWorkers.
- The cap prevents spawning vastly more children than a system can handle, resulting in downtime, revenue loss, reputation loss or even data loss.

# MPM Event/Worker Optimization

- **ServerLimit**
- ServerLimit ties in directly with the thrashing point.
- The thrashing point is the maximum number of children Apache can run before memory usage tips the scale into perpetual swap.
- Match the ServerLimit to the calculated thrashing point to safeguard the server.

| ServerLimit    |                                    |
|----------------|------------------------------------|
| <b>Syntax</b>  | <code>ServerLimit number</code>    |
| <b>Default</b> | 16                                 |
| <b>Module</b>  | <i>mpm_event module</i>            |
| <b>s</b>       | <i>mpm_worker_module</i>           |
| <b>Doc</b>     | <a href="#"><u>ServerLimit</u></a> |

# MPM Event/Worker Optimization

- **ThreadsPerChild**
- Used to define the limit of threads that each Apache child can manage.
- Every thread running can handle a single request.
- The default of 25 works well for most cases and is a fair balance between children and threads.
- There is an upper limit on this directive as well, and the limit is controlled by the ThreadLimit directive, which defaults to 64 threads.
- The adjustments to increase ThreadsPerChild past 64 threads also need to be made to ThreadLimit.

# MPM Event/Worker Optimization

- **ThreadsPerChild**
- Increasing this value allows each child to handle more requests keeping memory consumption down while allowing a larger MaxRequestWorkers directive.
- A key benefit of running more threads within each child is shared memory cache access.
- Threads from one child cannot access caches from another child.
- Boosting the number of threads per child squeezes out more performance due to this sharing of cache data.
- The major downside for increased threads per child occurs during child recycling.
- The capacity of the server is diminished by the number of threads configured for each child when that child process is eventually recycled (graceful restart).

# MPM Event/Worker Optimization

- **ThreadsPerChild**

| ThreadsPerChild |                                     |
|-----------------|-------------------------------------|
| <b>Syntax</b>   | <code>ThreadsPerChild number</code> |
| <b>Default</b>  | 25                                  |
| <b>Module</b>   | <i>mpm_event module</i>             |
| <b>s</b>        | <i>mpm_worker_module</i>            |
| <b>Doc</b>      | <a href="#">ThreadsPerChild</a>     |

| CPU<br>Cores | MPM Event/Worker |   |                 |   |                   |
|--------------|------------------|---|-----------------|---|-------------------|
|              | ServerLimit      | x | ThreadsPerChild | = | MaxRequestWorkers |
| 2            | 2                | x | 64              | = | 128               |
| 4            | 4                | x | 50              | = | 200               |
| 8            | 8                | x | 25              | = | 200               |
| 16           | 16               | x | 25              | = | 400               |
| 32           | 32               | x | 16              | = | 512               |

Examples of various *ThreadsPerChild* configurations in Worker based MPMS.

# MPM Event/Worker Optimization

- **ThreadsPerChild**
- Inversely the opposite reaction is achieved by lowering ThreadsPerChild.
- Fewer threads per child require more children to run an equal amount of MaxRequestWorkers.
- Since children are full copies of Apache, this increases Apache's overall memory footprint but reduces the impact when recycling children.
- Fewer threads mean fewer potential "stuck" threads during the recycle procedure, keeping the higher capacity of requests available overall children.
- Having fewer threads per child provides increased shared memory isolation.
- For instance, dropping ThreadsPerChild to 1 gives the same request isolation of MPM Prefork but also inherits its massive performance tax as well, requiring one child per one request.

# MPM Event/Worker Optimization

## Tip:

When setting *ThreadsPerChild* always consider the server environment and hardware.

- A memory-heavy shared server hosting numerous independent accounts might opt for a lower *ThreadsPerChild*, reducing the potential impact of one user affecting another.
- A dedicated Apache server in a high capacity load balanced configuration can choose to increase *ThreadsPerChild* significantly for a better overall performance of each thread.

# MPM Event/Worker Optimization

---

- **ThreadLimit**
- Used to set the maximum value of ThreadsPerChild.
- This directive is a hard ceiling for ThreadsPerChild.
- It helps protect against typographical errors with the ThreadLimit.
- ThreadsPerChild directive which could quickly spin a server out of control if too many threads are allowed due to an input error.
- This setting need to be adjusted in some high-end servers when the system needs more than the default of 64 threads per child.

# MPM Event/Worker Optimization

- ThreadLimit

| ThreadLimit    |                             |
|----------------|-----------------------------|
| <b>Syntax</b>  | ThreadLimit number          |
| <b>Default</b> | 64                          |
| <b>Module</b>  | <i>mpm_event module</i>     |
| <b>s</b>       | <i>mpm_worker_module</i>    |
| <b>Doc</b>     | <a href="#">ThreadLimit</a> |

# MPM Event/Worker Optimization

- **MaxRequestWorkers / MaxClients**
- The directive sets the limit for active worker threads across all running children and acts as a soft ceiling with ServerLimit taking control as the hard limit.
- When the number of total running threads has reached or exceeded MaxRequestWorkers, Apache no longer spawns new children.MaxRequestWorkers/MaxClients.
- Determining the MaxRequestWorkers is a critical part of server optimization.
- An optimal setting is based on several changing variables.

# MPM Event/Worker Optimization

---

- **MaxRequestWorkers / MaxClients**
- This means its configuration needs to be reevaluated and tailored periodically over time, changed by watching traffic habits and system resource usage.
- The Apache status Scoreboard is an effective tool for analysis of Apache performance.
- It is typical of Worker based MPM systems to run an isolated third-party PHP handler like Mod\_fcgid, PHP-FPM, and mod\_lsapi.

# MPM Event/Worker Optimization

---

- **MaxRequestWorkers / MaxClients**
- These modules are responsible for processing PHP code outside of Apache and frees up Apache to handle all other non-PHP requests such as HTML, TEXT, CSS, Images, etc...
- These requests are far less taxing on server resources which allows Apache to handle larger volumes of requests, such as those beyond 400 MaxRequestWorkers.

# MPM Event/Worker Optimization

- **MaxRequestWorkers / MaxClients**

| CPU Cores | MPM Event/Worker |   |                 |   | MaxRequestWorkers |
|-----------|------------------|---|-----------------|---|-------------------|
|           | ServerLimit      | x | ThreadsPerChild | = |                   |
| 2         | 2                | x | 64              | = | 128               |
| 4         | 4                | x | 50              | = | 200               |
| 8         | 8                | x | 50              | = | 400               |
| 16        | 16               | x | 32              | = | 512               |
| 32        | 32               | x | 32              | = | 1024              |

Examples of various *MaxRequestWorkers* configurations using Worker based MPM

# MPM Event/Worker Optimization

- **MinSpareThreads**
- The least number of Threads that should remain open, waiting for new requests.
- MinSpareThreads is a multiple of ThreadsPerChild and cannot exceed MaxSpareThreads, though it can match it.

## Rule-of-Thumb:

Set *MinSpareThreads* to equal 50% of *MaxRequestWorkers*.

# MPM Event/Worker Optimization

---

- **MinSpareThreads**
- Spare threads are idle workers threads.
- These threads are merely waiting for new incoming requests and are governed by the Apache child process that spawned them.
- If there are less available threads than MinSpareThreads, the Apache parent will generate a new child with another ThreadsPerChild worth of threads.

# MPM Event/Worker Optimization

- **MinSpareThreads**

| CPU Cores | MPM Event/Worker |   |                 |   | MaxRequestWorkers |
|-----------|------------------|---|-----------------|---|-------------------|
|           | ServerLimit      | x | ThreadsPerChild | = |                   |
| 2         | 2                | x | 64              | = | 128               |
| 4         | 4                | x | 50              | = | 200               |
| 8         | 8                | x | 50              | = | 400               |
| 16        | 16               | x | 32              | = | 512               |
| 32        | 32               | x | 32              | = | 1024              |

Examples of various *MaxRequestWorkers* configurations using Worker based MPM

# MPM Event/Worker Optimization

- **MaxSpareThreads**
- This directive governs the total number of idle threads allowed on the server across all children.
- Any threads above this limit direct their parent to shut down to reduce memory consumption during off-peak hours.

| MaxSpareThreads |                                                     |
|-----------------|-----------------------------------------------------|
| <b>Syntax</b>   | <code>MinSpareThreads number**</code>               |
| <b>Default</b>  | 250                                                 |
| <b>Modules</b>  | <i>mpm_event_module</i><br><i>mpm_worker_module</i> |
| <b>Doc</b>      | <a href="#">MinSpareThreads</a>                     |

\*\* This value must be evenly divisible by **ThreadsPerChild**. If not, Apache will force it to the closest multiple, rounding down.

# MPM Event/Worker Optimization

- **MaxSpareThreads**
- This directive governs the total number of idle threads allowed on the server across all children.
- Any threads above this limit direct their parent to shut down to reduce memory consumption during off-peak hours.

| MaxSpareThreads |                                                     |
|-----------------|-----------------------------------------------------|
| <b>Syntax</b>   | <code>MinSpareThreads number**</code>               |
| <b>Default</b>  | 250                                                 |
| <b>Modules</b>  | <i>mpm_event_module</i><br><i>mpm_worker_module</i> |
| <b>Doc</b>      | <a href="#">MinSpareThreads</a>                     |

\*\* This value must be evenly divisible by **ThreadsPerChild**. If not, Apache will force it to the closest multiple, rounding down.

# MPM Event/Worker Optimization

---

- **MaxSpareThreads**
- Having a limit to the number of idle open threads is excellent for smaller servers with hardware constraints.
- However, it mostly unneeded on today's modernizing hardware.

# MPM Event/Worker Optimization

- **MaxSpareThreads**

**Tip:**

Configuring Apache as an open throttle is a high-performance configuration for servers with significant RAM and multiple CPU cores. When running the open throttle configuration, all available threads become available at all time. Apache's memory usage will stay near its peak at all times, a side effect due to running all the configured children into memory preemptively. This configuration will produce the best possible response times from Apache by maintaining persistent open connections ready to do work and removing the overhead of spawning new processes in response to traffic surges.

**Configuration:** Match both *MinSpareThreads* and *MaxSpareThreads* to *MaxRequestWorkers*.

**Requirements:** Make sure there is enough server RAM to run all *MaxRequestWorkers* at once.

# MPM Event/Worker Optimization

- **StartServers**
- This directive governs the initial amount of children the Apache Parent process spawns when the Apache service is started or restarted.
- This is commonly left unchanged since Apache continuously checks the current running children in conjunction with ThreadsPerChild and compare it to MinSpareThreads to determine if more children get forked.
- This process is repeated perpetually, with a doubling of new children on each iteration, until MinSpareThreads is satisfied.

# MPM Event/Worker Optimization

- **StartServers**

## Rule-of-Thumb:

Manually calculating StartServers is done by dividing *MaxRequestWorkers* by *ThreadsPerChild*, rounding down to the nearest whole number. This process forces all children to be created without delay at startup and begins handling requests immediately. This aspect is especially useful in modern Apache servers which require periodic restarts to load in directive changes.

| StartServers   |                                 |
|----------------|---------------------------------|
| <b>Syntax</b>  | <code>StartServer number</code> |
| <b>Default</b> | 3                               |
| <b>Module</b>  | <i>mpm_event module</i>         |
| <b>s</b>       | <i>mpm_worker_module</i>        |
| <b>Doc</b>     | <a href="#">StartServers</a>    |

# MPM Event/Worker Optimization

- **MaxConnectionsPerChild / MaxRequestsPerChild**
- The number of requests a single Apache child process can handle equals a cumulative total on the child server across all threads it controls.
- Each request handled by a thread counts toward this limit to its parent.
- Once the child server has reached its limit, the child is then recycled.
- This directive is a stop-gap for accidental memory leaks.
- Some code executed through Apache threads may contain memory leaks.
- Leaked memory are portions of memory that subprocess failed to release properly, so they are inaccessible to any outside processes.

# MPM Event/Worker Optimization

- **MaxConnectionsPerChild / MaxRequestsPerChild**
- The longer a leaking program is left running, the more memory it will leak.
- Setting a MaxConnectionsPerChild limit is a specific method for assuring Apache is periodically recycling programs to reduce the impact of leaked memory on the system.
- When using external code handlers like Mod\_fcgid, PHP-FPM or mod\_Isapi, it becomes necessary to set MaxConnectionsPerChild to 0 (unlimited), doing so prevents periodic error pages caused by Apache terminating threads prematurely.

# MPM Event/Worker Optimization

- **MaxConnectionsPerChild / MaxRequestsPerChild**

## Rule-of-Thumb:

### MaxConnectionsPerChild/MaxRequestsPerChild

|                                                                            |                                                                 |
|----------------------------------------------------------------------------|-----------------------------------------------------------------|
| <b>Syntax</b>                                                              | <code>MaxConnectionsPerChild number</code>                      |
| <b>Default</b>                                                             | <code>0 (unlimited)</code>                                      |
| <b>Modules</b>                                                             | <code>mpm_event_module</code><br><code>mpm_worker_module</code> |
| <b>Doc</b>                                                                 | <a href="#">MaxRequestWorkers</a>                               |
| <i>MaxConnectionsPerChild replaced MaxRequestsPerChild with Apache 2.4</i> |                                                                 |

If the server encounters a memory leak never set the *MaxConnectionsPerChild / MaxRequestsPerChild* too low, instead start with 10,000 and reduce it incrementally.

# MPM Prefork Optimization

- This MPM Prefork section details the use and performance considerations for various directives when running this module.
- This MPM is a non-threaded multi-processor designed for compatibility.
- It consists of a single Apache parent process, which is used to govern all new Apache processes also known as children.
- The following directives show how Apache is capable of performance tuning when using MPM Prefork.
- Unlike Worker based MPMs, optimizing MPM Prefork is generally simple and straightforward.

# MPM Prefork Optimization

---

- There is a 1:1 ratio of Apache processes to incoming requests.
- However, MPM Prefork does not scale well with hardware and the more traffic it encounters, the more hardware it will need to keep up with the pace.
- It should be noted that some directives behave differently based on which MPM is loaded.

# MPM Prefork Optimization

- **MaxRequestWorkers / MaxClients**
- Used to control the upper limit of children that the Apache parent server is allowed to have in memory at one time.
- These children (also called workers) handle requests on a 1:1 ratio.
- This translates into the maximum number of simultaneous requests the server can handle.
- **MaxRequestWorkers / MaxClients** If this directive is too low, Apache under-utilizes the available hardware which translates to wasted money and long delays in page load times during peak hours.

# MPM Prefork Optimization

- **MaxRequestWorkers / MaxClients**
- Alternatively, if this directive is too high, Apache outpaces the underlying hardware sending the system into thrashing (link to thrashing article) scenario which can lead to server crashes and potential data loss.

| MaxRequestWorkers/MaxClients |                                                     |
|------------------------------|-----------------------------------------------------|
| <b>Syntax</b>                | MaxRequestWorkers/MaxClients number                 |
| <b>Default</b>               | 256                                                 |
| <b>Modules</b>               | <i>mpm_event_module</i><br><i>mpm_worker_module</i> |
| <b>Doc</b>                   | <a href="#">MaxRequestWorkers</a>                   |

*MaxConnectionsPerChild replaced MaxRequestsPerChild with Apache 2.4*

# MPM Prefork Optimization

---

- **MinSpareServers**
- This directive defines a minimum number of spare children the Apache parent process can maintain in its memory.
- An additional server is a preforked idle Apache child that is ready to respond to a new incoming request.
- Having idle children waiting for new requests is essential for providing the fastest server response times.
- When the total idle children on the server drop below this value, a new child is preforked at the rate of one per second until this directive is satisfied.

# MPM Prefork Optimization

---

- **MinSpareServers**
- The “one per second” rule is in place to prevent surges of the creation process that overload the server, however, this failsafe comes at a cost.
- The one per second spawn rate is particularly slow when it comes to handling page requests.
- So it’s highly beneficial to make sure enough children are preforked and ready to handle incoming requests.
-

# MPM Prefork Optimization

- **MinSpareServers**

## Rule of Thumb:

Never set this to zero. Setting this to 25% of *MaxRequestWorkers* ensures plenty resources are ready and waiting for requests.

| MinSpareServers |                                            |
|-----------------|--------------------------------------------|
| <b>Syntax</b>   | <code>MaxConnectionsPerChild number</code> |
| <b>Default</b>  | <code>0 (unlimited)</code>                 |
| <b>Module</b>   | <code>mpm_event module</code>              |
| <b>s</b>        | <code>mpm_worker_module</code>             |
| <b>Doc</b>      | <a href="#">MaxRequestWorkers</a>          |

# MPM Prefork Optimization

---

- **MaxSpareServers**
- MaxSpareServers controls the maximum number of idle Apache child servers running at one time.
- An idle child is one which is not currently handling a request but waiting for a new request.
- When there are more than MaxSpareServers idle children, Apache kills off the excess.
- If the MaxSpareServers value is less than MinSpareServers, Apache will automatically adjust MaxSpareServers to equal MinSpareServers plus one.
- Like with MinSpareServers, this value should always be altered with available server resources in mind.

# MPM Prefork Optimization

- **MaxSpareServers**

## Rule of Thumb:

| MaxSpareServers |                                     |
|-----------------|-------------------------------------|
| <b>Syntax</b>   | <code>MaxSpareServers number</code> |
| <b>Default</b>  | 10                                  |
| <b>Module</b>   | <code>mpm_prefork_module</code>     |
| <b>Doc</b>      | <a href="#">MaxSpareServers</a>     |

Set this to double the value of *MinSpareServers*.

# MPM Prefork Optimization

---

- **MaxSpareServers**
- Configuring Apache as an open throttle is a high-performance configuration for servers with significant RAM and multiple CPU cores.
- When running the open throttle configuration, all available Apache children become available at all times.
- As a side effect of running open throttle, the Apache memory usage will stay near its peak at all times, due to running all the configured children into memory preemptively.
- This configuration will produce the best possible response times by maintaining persistent open connections.

# MPM Prefork Optimization

---

- **MaxSpareServers**
- Furthermore, in response to traffic surges, it removes the overhead that comes from spawning new processes.
- Configuration: Match both MinSpareServers and MaxSpareServers to MaxRequestWorkers.
- Requirements: Make sure there is enough server RAM to run all MaxRequestWorkers at once.

# MPM Prefork Optimization

---

- **StartServers**
- Created at startup, are the initial amount of Apache child servers.
- This seldom changed directive only impacts Apache startup and restart processes.
- Generally not altered because Apache uses internal logic to work out how many child servers should be running.
- Many modern servers periodically restart Apache to address configuration changes, rotate log files or other internal processes.
- When this occurs during a high load traffic surge, every bit of downtime matters.
- You can manually set the StartServers directive to mirror that of your MinSpareServers to shave off time from the Apache startup.

# MPM Prefork Optimization

- **StartServers**

## Rule of Thumb:

The *StartServers* directive should mirror that of *MinSpareServers*.

| StartServers   |                                                     |
|----------------|-----------------------------------------------------|
| <b>Syntax</b>  | MaxSpareServers number                              |
| <b>Default</b> | 5                                                   |
| <b>Modules</b> | <i>mpm_event_module</i><br><i>mpm_worker_module</i> |
| <b>Doc</b>     | <a href="#">StartServers</a>                        |

# MPM Prefork Optimization

- **ServerLimit**
- The ServerLimit directive represents the upper limit of MaxRequestWorkers.
- This setting is generally used as a safeguard or ceiling against input errors when modifying MaxRequestWorkers.
- ServerLimit Default Setting.
- It becomes necessary to adjusted ServerLimit when the server is expected to handle more than the default of 256 requests simultaneously.
- ServerLimit ties in directly with the thrashing point.
- The thrashing point is the maximum number of children Apache can run before memory usage tips the scale into perpetual swap.
- Match the ServerLimit to the calculated thrashing point to safeguard the server.

# MPM Prefork Optimization

- **ServerLimit**

| <b>ServerLimit</b> |                                    |
|--------------------|------------------------------------|
| <b>Syntax</b>      | <code>ServerLimit number</code>    |
| <b>Default</b>     | 256                                |
| <b>Modules</b>     | <code>mpm_prefork_module</code>    |
| <b>Doc</b>         | <a href="#"><u>ServerLimit</u></a> |

# MPM Prefork Optimization

- **ServerLimit**

**Note:**

Increasing *ServerLimit* is not recommended with MPM Prefork. Running more than 256 simultaneous requests is hardware intensive when using the MPM Prefork module.

# MPM Prefork Optimization

- **MaxConnectionsPerChild / MaxRequestsPerChild**
- This directive equals the number of requests a single Apache child server can handle.
- This directive is a stop-gap for accidental memory leaks.
- Code executed through Apache may contain faults which leak memory.
- These leaks add up over time making less and less of the shared memory pool of the child usable.
- The way to recover from leaked memory is to recycle the affected Apache child process.
- Setting a MaxConnectionsPerChild limit will protect from this type of memory leakage.

# MPM Prefork Optimization

- **MaxConnectionsPerChild / MaxRequestsPerChild**

## Note:

### MaxConnectionsPerChild/MaxRequestsPerChild

|                |                                                                 |
|----------------|-----------------------------------------------------------------|
| <b>Syntax</b>  | <code>MaxConnectionsPerChild number</code>                      |
| <b>Default</b> | <code>0 (unlimited)</code>                                      |
| <b>Modules</b> | <code>mpm_event_module</code><br><code>mpm_worker_module</code> |
| <b>Doc</b>     | <a href="#">MaxRequestWorkers</a>                               |

*MaxConnectionsPerChild replaced MaxRequestsPerChild with Apache 2.4*

Rule-of-Thumb: Never set this too low. If the server encounters memory leak issues start with 10,000 and reduce incrementally.

# Apache Performance Tuning: Configuring MPM Directives



- On Ubuntu servers, Apache configuration files are located in /etc/apache2/
- Backup existing apache2.conf file:
  - `cp -p /etc/apache2/apache2.conf{,.bak.$(date +\%F_\%H\%M\%S)}`
- `ls -lah /etc/apache2/apache2.conf*`
- Open file for editing with your favorite editor:
- `vim /etc/apache2/apache2.conf`

# Apache Performance Tuning: Configuring MPM Directives



- Append the necessary directive changes to the very bottom of the config file.

| MPM Prefork Example:                                                                                                                                                                                                                                           | MPM Event Example:                                                                                                                                                                                                                                                              | MPM Worker Example:                                                                                                                                                                                                                                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>Timeout 30 &lt;IfModule mpm_prefork_module&gt; KeepAlive On MaxKeepAliveRequests 500 KeepAliveTimeout 3  ServerLimit 23 StartServers 12  MinSpareServers 12 MaxSpareServers 23  MaxRequestWorkers 23 MaxConnectionsPerChild 10000 &lt;/IfModule&gt;</pre> | <pre>Timeout 30 &lt;IfModule mpm_event_module&gt; KeepAlive On KeepAliveTimeout 3 MaxKeepAliveRequests 500  ThreadsPerChild 25 ServerLimit 23 MaxRequestWorkers 400  StartServers 16 MinSpareThreads 200 MaxSpareThreads 400  MaxRequestsPerChild 10000 &lt;/IfModule&gt;</pre> | <pre>Timeout 30 &lt;IfModule mpm_worker_module&gt; KeepAlive On KeepAliveTimeout 1 MaxKeepAliveRequests 500  ThreadsPerChild 25 ServerLimit 23 MaxRequestWorkers 400  StartServers 16 MinSpareThreads 200 MaxSpareThreads 400  MaxRequestsPerChild 10000 &lt;/IfModule&gt;</pre> |

# Apache Performance Tuning: Configuring MPM Directives

---



- sudo apachectl -t
- Sudo service apache2 start
- Sudo apachectl start

# How To Configure Apache to Use Custom Error Pages on Ubuntu



- **Creating Your Custom Error Pages**
- Go to /var/www/html
- echo "<h1 style='color:red>Error 404: Not found :-(</h1>" | sudo tee /var/www/html/custom\_404.html
- echo "<p>I have no idea where that file is, sorry. Are you sure you typed in the correct URL?</p>" | sudo tee -a /var/www/html/custom\_404.html
- echo "<h1>Oops! Something went wrong...</h1>" | sudo tee /var/www/html/custom\_50x.html
- echo "<p>We seem to be having some technical difficulties. Hang tight.</p>" | sudo tee -a /var/www/html/custom\_50x.html

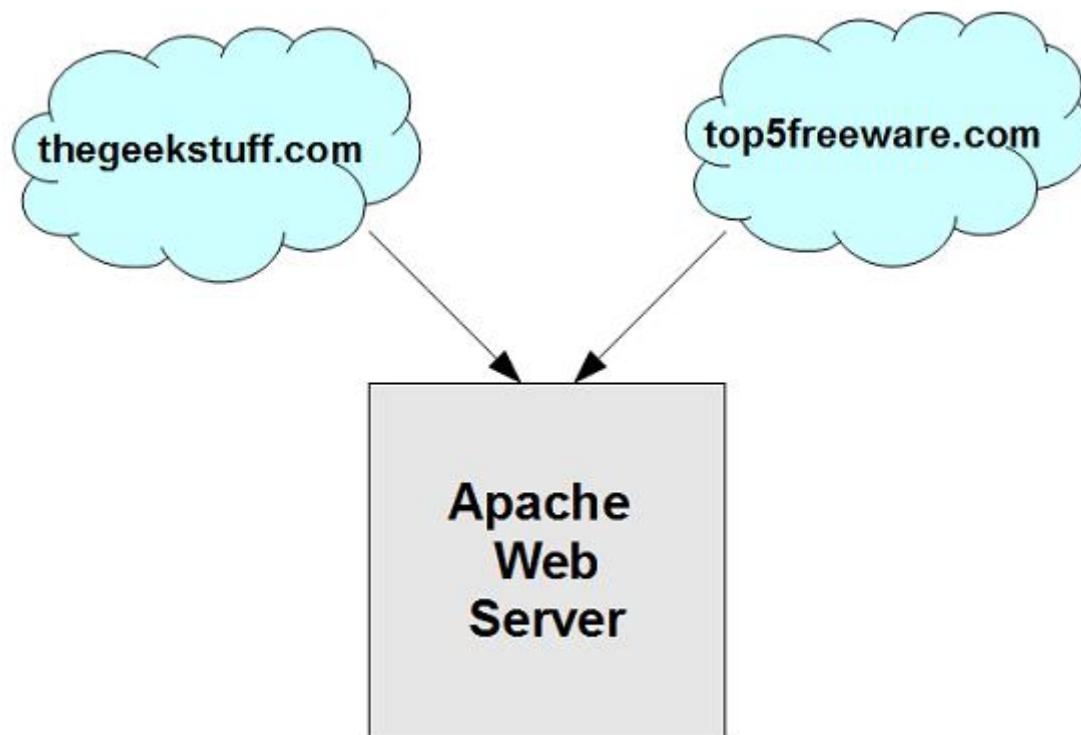
# How To Configure Apache to Use Custom Error Pages on Ubuntu



- **Configuring Apache to Use your Error Pages**
- sudo nano /etc/apache2/sites-enabled/test.com.conf
- Refer for the changes
- sudo a2ensite test.com
- Sudo service apache2 reload

# How To Setup Apache Virtual Host Configuration

- Using Apache Virtual Host, you can run several websites on the same server.
- For example, I can run both [thegeekstuff.com](http://thegeekstuff.com) and [top5freeware.com](http://top5freeware.com) on a single physical server that has one Apache webserver running on it.



# How To Setup Apache Virtual Host Configuration

- There are two types of Apache virtual host configurations:
- 1) IP-Based Virtual Host and 2) Name-based Virtual Host.
- Name-based virtual host is recommended for most scenarios.

# How To Setup Apache Virtual Host Configuration

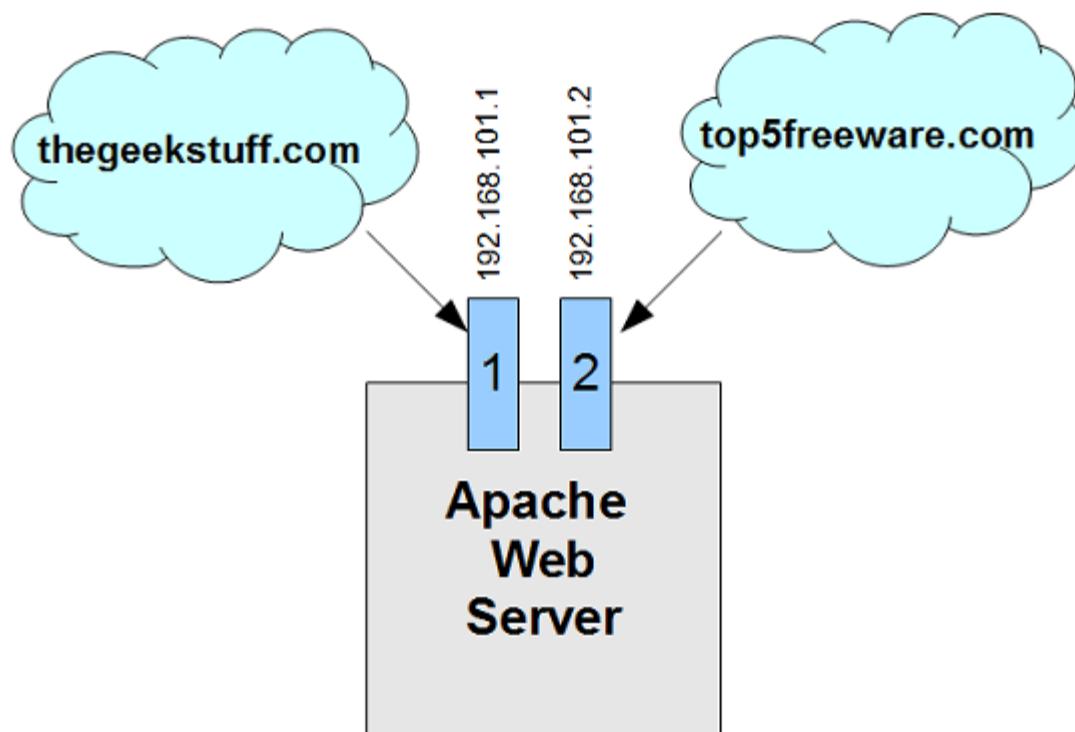
- **IP-Based Virtual Host**
- In this configuration, when you are pointing two websites (with different ip-address) to the server that runs Apache, that physical server should have two different ip-address configured.
- This means that the server should have two ethernet cards, each one of them configured to the ip-address of the corresponding website that Apache virtual host will be serving.
- So, this is not practical for most aspects, and you should not be using this.

# How To Setup Apache Virtual Host Configuration

- **IP-Based Virtual Host**
- In the following example, the server contains two NIC cards, one is configured with 192.168.101.1 ip-address for [thegeekstuff.com](http://thegeekstuff.com), another is configured with 192.168.102.1 for [top5freeware.com](http://top5freeware.com).
- Both these ip-address are served by a single Apache webserver running on that server using IP-Based virtual host.

# How To Setup Apache Virtual Host Configuration

- IP-Based Virtual Host



# How To Setup Apache Virtual Host Configuration

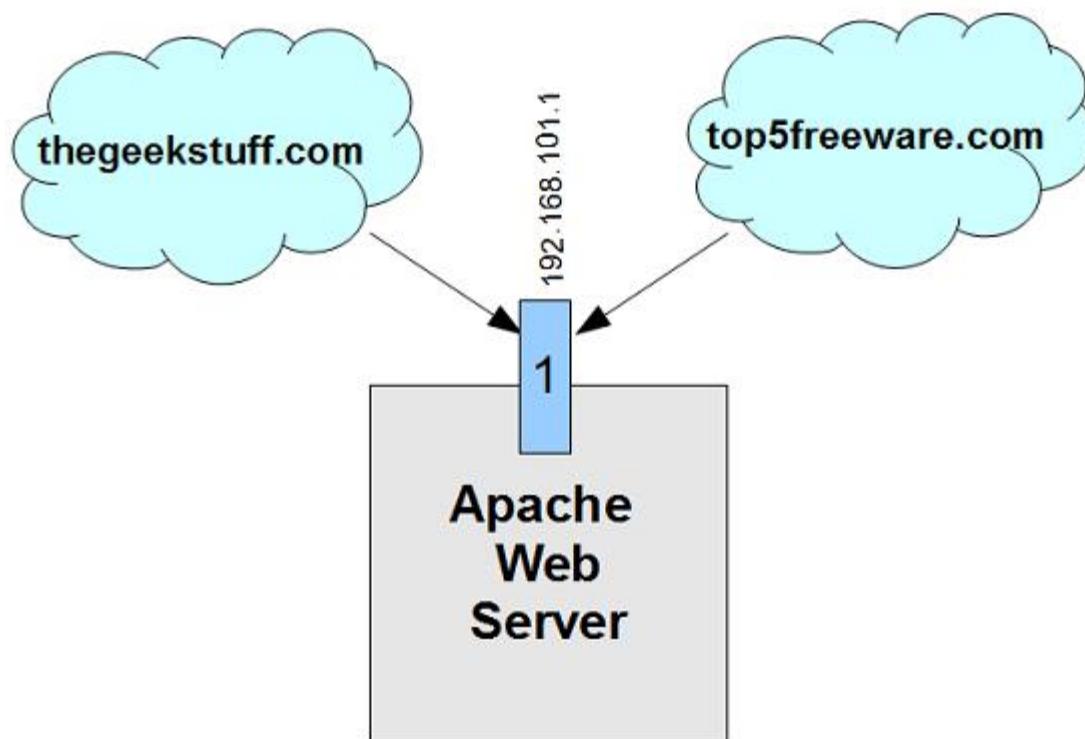
- **Name-Based Virtual Host**
- In this configuration, when Apache webserver receives a request, it looks for the hostname in the HTTP header, and depending on the hostname, it servers different websites.
- This is very easy, as you need only one ip-address on that physical server; but, you update the DNS with multiple website names pointing to the same ip-address.
- For all practical purpose, you'll be using only Name-based virtual host configuration.

# How To Setup Apache Virtual Host Configuration

- **Name-Based Virtual Host**
- In the following example, the server contains only one NIC card, which is configured with 192.168.101.1 ip-address.
- The DNS entry for both thegeekstuff.com and top5freeware.com website points to 192.168.101.1 ip-address.
- When Apache receives a request, it looks for the hostname entry in the HTTP header, and serves the corresponding website.

# How To Setup Apache Virtual Host Configuration

- **Name-Based Virtual Host**



# How To Setup Apache Virtual Host Configuration

---

- sudo mkdir -p /var/www/geekstuff.com/public\_html
- sudo mkdir -p /var/www/top5freeware.com/public\_html
- sudo chown -R \$USER:\$USER /var/www/  
geekstuff.com/public\_html
- sudo chown -R \$USER:\$USER  
/var/www/top5freeware.com/public\_html
- sudo chmod -R 755 /var/www

# How To Setup Apache Virtual Host Configuration

---

- sudo nano  
/var/www/geekstuff.com/public\_html/index.html
- sudo nano  
/var/www/top5freeware.com/public\_html/index.html
- sudo cp /etc/apache2/sites-available/000-default.conf  
/etc/apache2/sites-available/geekstuff.com.conf
- sudo nano /etc/apache2/sites-available/top5free.com.conf

# How To Setup Apache Virtual Host Configuration

---

- sudo a2ensite geekstuff.com
- sudo a2ensite top5free.com
- Sudo service apache2 reload
- sudo nano /etc/hosts

# How To Setup Apache Virtual Host Configuration

```
Command Prompt
curl: (6) Could not resolve host: www.geekstuff.com

C:\Users\Balasubramaniam>curl -I www.geekstuff.com
curl: (6) Could not resolve host: www.geekstuff.com

C:\Users\Balasubramaniam>curl -I http://localhost:76
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2020 21:18:55 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Fri, 07 Aug 2020 20:29:24 GMT
ETag: "a3-5ac4f75f3239c"
Accept-Ranges: bytes
Content-Length: 163
Vary: Accept-Encoding
Content-Type: text/html

C:\Users\Balasubramaniam>curl -I http://localhost:78
HTTP/1.1 200 OK
Date: Fri, 07 Aug 2020 21:18:57 GMT
Server: Apache/2.4.41 (Ubuntu)
Last-Modified: Fri, 07 Aug 2020 20:30:19 GMT
ETag: "a5-5ac4f793758e1"
Accept-Ranges: bytes
Content-Length: 165
Vary: Accept-Encoding
Content-Type: text/html

C:\Users\Balasubramaniam>
```

# Apache Security Principles

---

- **Security Definitions**
- Security can be defined in various ways.
- One school of thought defines it as reaching the three goals known as the CIA triad:
- Confidentiality
  - Information is not disclosed to unauthorized parties.
- Integrity
  - Information remains unchanged in transit or in storage until it is changed by an authorized party.
- Availability
  - Authorized parties are given timely and uninterrupted access to resources and information.

# Common Security Vocabulary

---

- Weakness
  - A less-than-ideal aspect of a system, which can be used by attackers in some way to bring them closer to achieving their goals.
  - A weakness may be used to gain more information or as a stepping-stone to other system parts.
- Vulnerability
  - Usually a programming error with security consequence

# Common Security Vocabulary

---

- Exploit
  - A method (but it can be a tool as well) of exploiting a vulnerability.
  - This can be used to break in or to increase user privileges (known as privilege elevation).
- Attack vector
  - An entry point an adversary could use to attempt to break in.
  - A popular technique for reducing risk is to close the entry point completely for the attacker.
  - Apache running on port 80 is one example of an entry point.

# Common Security Vocabulary

---

- Attack surface
  - The area within an entry point that can be used for an attack.
  - This term is usually used in discussions related to the reduction of attack surface. For example, moving an ecommerce administration area to another IP address where it cannot be accessed by the public reduces the part of the application accessible by the attacker and reduces the attack surface and the risk.

# Security Process Steps

---

- 1. Understand the environment and the security requirements of the project.
- 2. Establish a security policy and design the system.
- 3. Develop operational procedures.
- 4. Configure carefully.
- 5. Perform maintenance and patch regularly.
- 6. Monitor.
- 7. Handle attacks.

# Major reasons why attacks take place

---

| Reason             | Description                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| To grab an asset   | Attackers often want to acquire something valuable, such as a customer database with credit cards or some other confidential or private information.                                                                                                                                                                                                          |
| To steal a service | This is a special form of the previous category. The servers you have with their bandwidth, CPU, and hard disk space are assets. Some attackers will want to use them to send email, store pirated software, use them as proxies and starting points for attacks on other systems, or use them as zombies in automated distributed denial of service attacks. |
| Recognition        | Attacks, especially web site defacement attacks, are frequently performed to elevate one's status in the underground.                                                                                                                                                                                                                                         |
| Thrill             | Some people love the thrill of breaking in. For them, the more secure a system, the bigger the thrill and desire to break in.                                                                                                                                                                                                                                 |
| Mistake            | Well, this is not really a reason, but attacks happen by chance, too.                                                                                                                                                                                                                                                                                         |

# Typical attacks on web systems

| Attack type                                 | Description                                                                                                                                                                    |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Denial of service                           | Any of the network, web-server, or application-based attacks that result in denial of service, a condition in which a system is overloaded and can no longer respond normally. |
| Exploitation of configuration errors        | These errors are our own fault. Surprisingly, they happen more often than you might think.                                                                                     |
| Exploitation of Apache vulnerabilities      | Unpatched or unknown problems in the Apache web server.                                                                                                                        |
| Exploitation of application vulnerabilities | Unpatched or unknown problems in deployed web applications.                                                                                                                    |

# Typical attacks on web systems

| Attack type                                 | Description                                                                                                                                                                                                  |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Denial of service                           | Any of the network, web-server, or application-based attacks that result in denial of service, a condition in which a system is overloaded and can no longer respond normally.                               |
| Exploitation of configuration errors        | These errors are our own fault. Surprisingly, they happen more often than you might think.                                                                                                                   |
| Exploitation of Apache vulnerabilities      | Unpatched or unknown problems in the Apache web server.                                                                                                                                                      |
| Exploitation of application vulnerabilities | Unpatched or unknown problems in deployed web applications.                                                                                                                                                  |
| Attacks through other services              | This is a “catch-all” category for all other unmitigated problems on the same network as the web server. For example, a vulnerable MySQL database server running on the same machine and open to the public. |

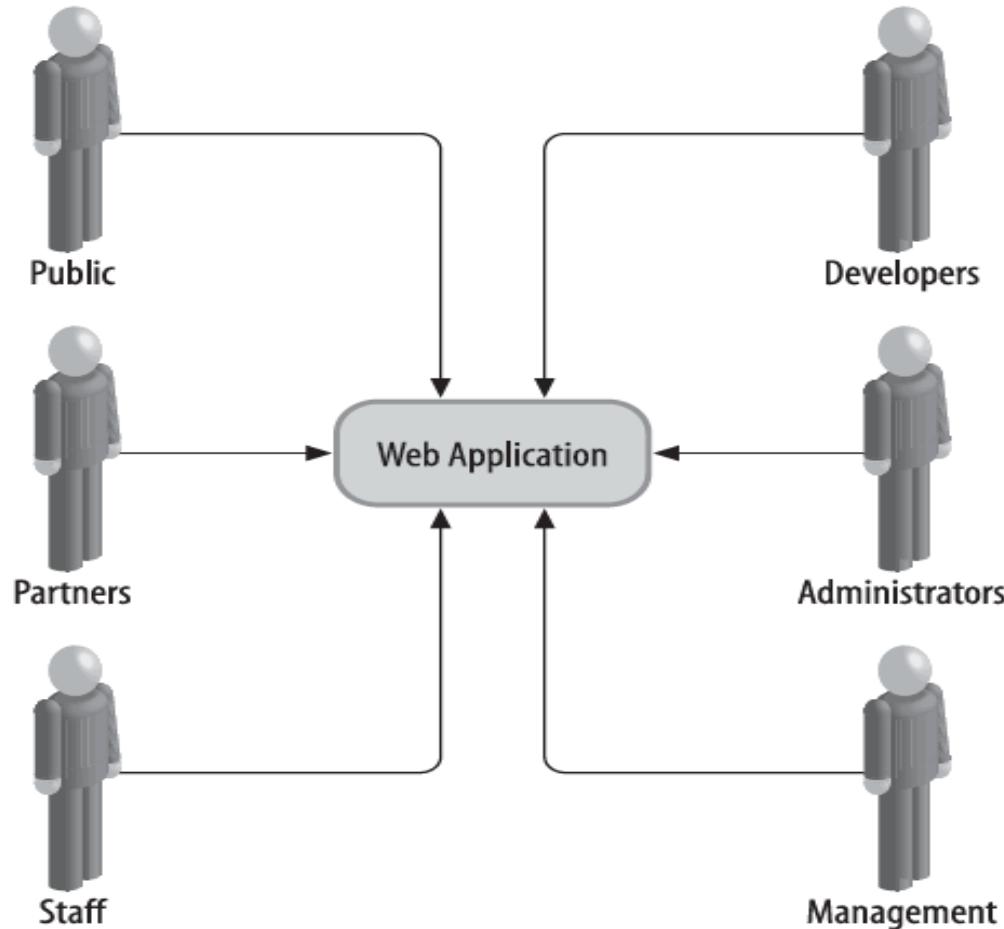
# Web Application Architecture Blueprints

---

- User view
- Network view
- Apache view

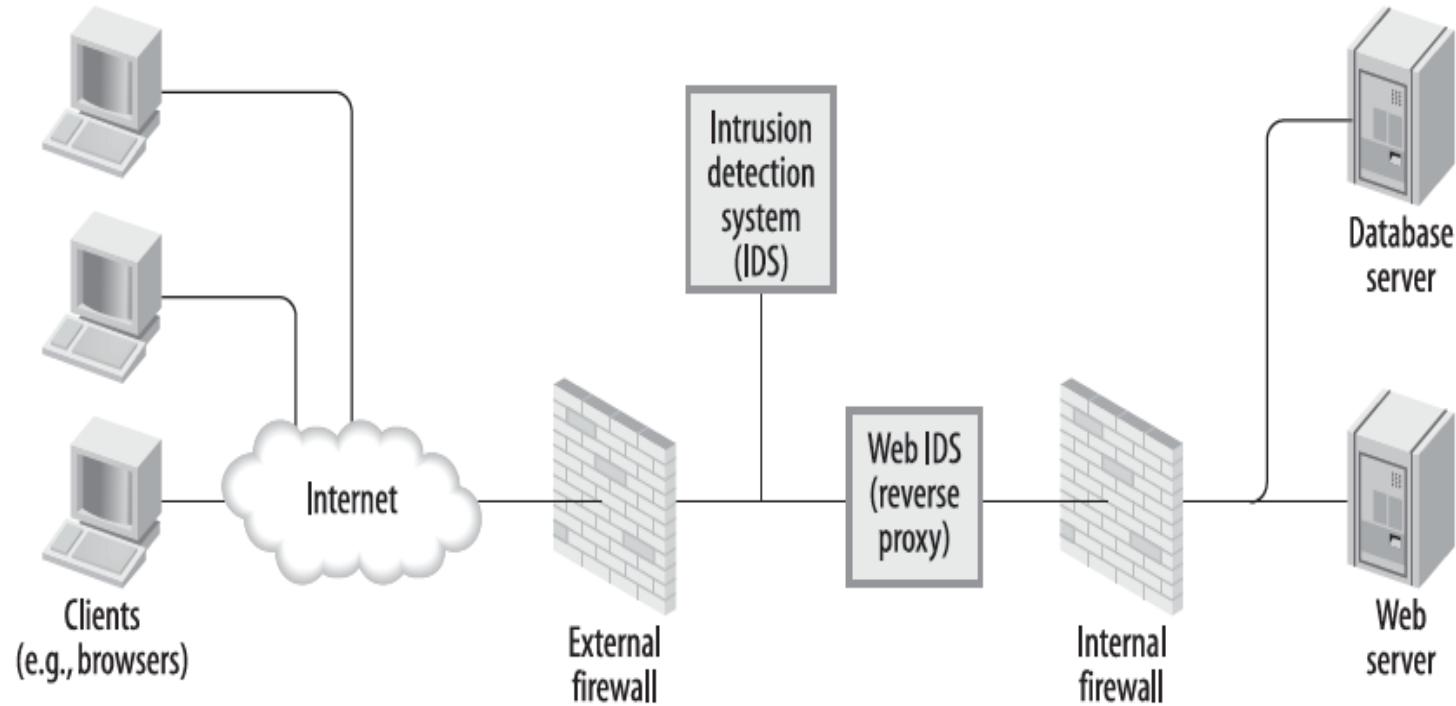
# Web Application Architecture Blueprints

## Web architecture: user view



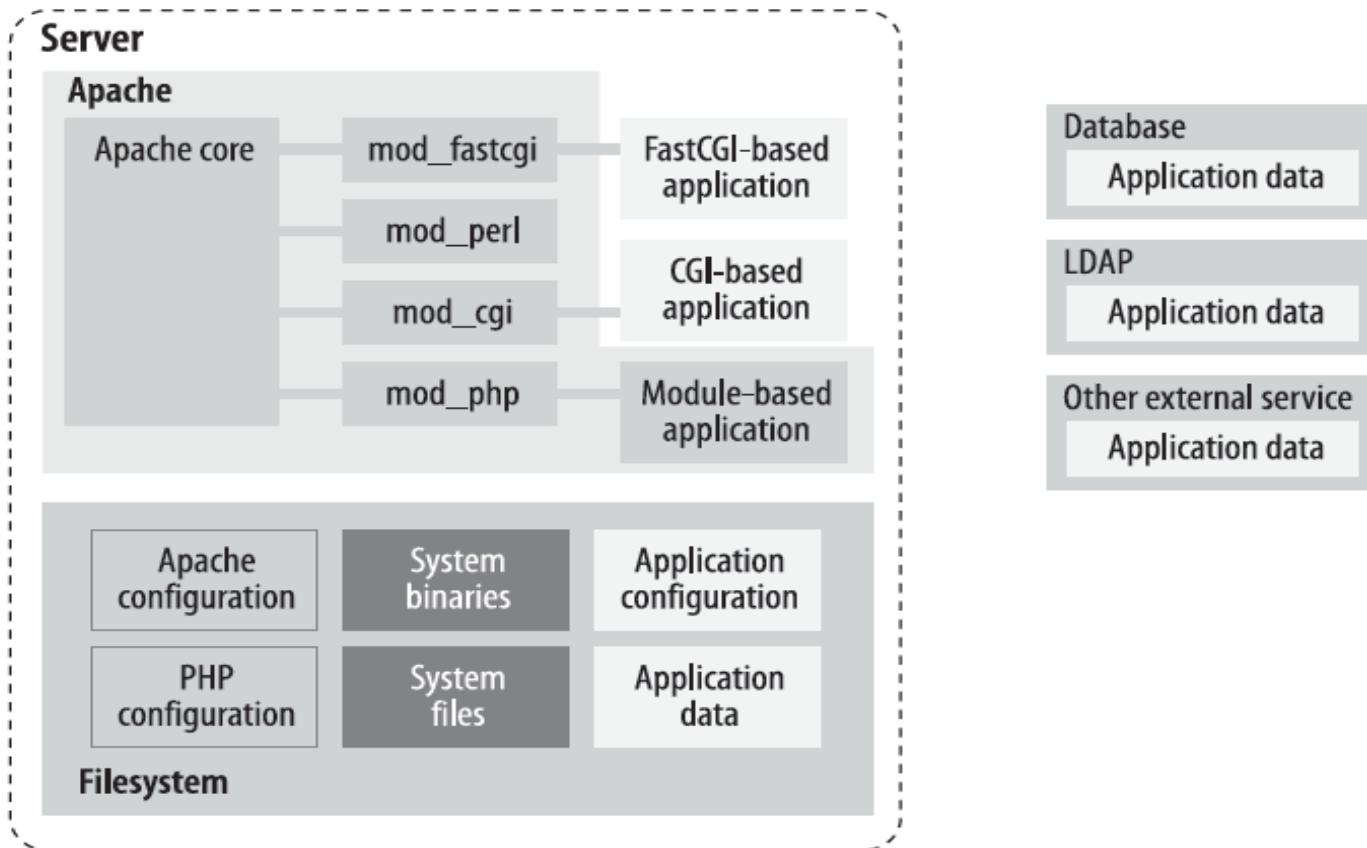
# Web Application Architecture Blueprints

## Network View



# Web Application Architecture Blueprints

## Web architecture: Apache view



# Setting Up the Server User Account

---

- Upon installation, Apache runs as a user nobody.
- While this is convenient (this account normally exists on all Unix operating systems), it is a good idea to create a separate account for each different task.
- The idea behind this is that if attackers break into the server through the web server, they will get the privileges of the web server.
- The intruders will have the same privileges as in the user account.
- By having a separate account for the web server, we ensure the attackers do not get anything else free.

# Setting Up the Server User Account

---

- To create a new account, execute the following two commands while running as root.
- `# sudo groupadd aspire`
- `# sudo useradd aspireuser -g aspire -d /dev/null -s /sbin/nologin`
- These commands create a group and a user account, assigning the account the home directory `/dev/null` and the shell `/sbin/nologin`
- Add the following two lines to the Apache configuration file `apache2.conf`:
- `User aspireuser`
- `Group aspire`

# Setting Apache Binary File Permissions

---

- After creating the new user account your first impulse might be to assign ownership over the Apache installation to it.
- For Apache to run on port 80, it must be started by the user root.
- Allowing any other account to have write access to the apache2.conf binary would give that account privileges to execute anything as root.

# Setting Apache Binary File Permissions

---

- This problem would occur, for example, if an attacker broke into the system.
- Working as the Apache user (aspireuser), he would be able to replace the apache2.conf binary with something else and shut the web server down.
- The administrator, thinking the web server had crashed, would log in and attempt to start it again and would have fallen into the trap of executing a Trojan program.

# Setting Apache Binary File Permissions

---

- That is why we make sure only root has write access:
- `# chown -R root:root /etc/apache2`
- `# chmod -R go-w /etc/apache2`
- No reason exists why anyone else other than the root user should be able to read the Apache configuration or the logs:
  - `# chmod -R go-r /etc/apache2/conf`
  - `# chmod -R go-r /var/log/apache2`

# Configuring Secure Defaults

---

- Apache will serve any file it can access.
- This is probably not what most people want; a configuration error could accidentally expose vital system files to anyone caring to look.
- To change this, we would deny access to the complete filesystem and then allow access to the document root only by placing the following directives in the apache2.conf configuration file

# Configuring Secure Defaults

---

- <Directory />
- Order Deny,Allow
- Deny from all
- </Directory>
- <Directory /var/www/htdocs>
- Order Allow,Deny
- Allow from all
- </Directory>

# Options directive

---

- All
  - All options listed below except MultiViews. This is the default setting.
- None
  - None of the options will be enabled.
- ExecCGI
  - Allows execution of CGI scripts.
- FollowSymLinks
  - Allows symbolic links to be followed.
- Includes
  - Allows server-side include

# Options directive

---

- IncludesNOEXEC
  - Allows SSIs but not the exec command, which is used to execute external scripts. (This setting does not affect CGI script execution.)
- Indexes
  - Allows the server to generate the list of files in a directory when a default index file is absent.
- MultiViews
  - Allows content negotiation.
- SymLinksIfOwnerMatch
  - Allows symbolic links to be followed if the owner of the link is the same as the owner of the file it points to.

## AllowOverride directive

---

- In addition to serving any file it can access by default, Apache also by default allows parts of configuration data to be placed under the web server tree, in files normally named .htaccess.
- Configuration information in such files can override the information in the apache2.conf configuration file.
- Though this can be useful, it slows down the server (because Apache is forced to check whether the file exists in any of the subfolders it serves) and allows anyone who controls the web server tree to have limited control of the web server.

## AllowOverride directive

---

- This feature is controlled with the AllowOverride directive, which, like Options, appears within the <Directory> directive specifying the directory to which the options apply.

# AllowOverride directive

---

- The AllowOverride directive supports the following options:
  - AuthConfig
    - Allows use (in .htaccess files) of the authorization directives
  - FileInfo
    - Allows use of the directives controlling document types
  - Indexes
    - Allows use of the directives controlling directory indexing
  - Limit
    - Allows use of the directives controlling host access

# AllowOverride directive

---

- Options
  - Allows use of the directives controlling specific directory functions (the Options and XbitHack directives)
- All
  - Allows all options listed
- None
  - Ignores .htaccess configuration files

## AllowOverride directive

- For our default configuration, we choose the None option. So, our `<Directory>` directives are now:

```
<Directory />
```

```
 Order Deny,Allow
```

```
 Deny from all
```

```
 Options None
```

```
 AllowOverride None
```

```
</Directory>
```

```
<Directory /var/www/htdocs>
```

```
 Order Allow,Deny
```

```
 - Allow from all
```

- ```
</Directory>
```

Enabling CGI Scripts

- Only enable CGI scripts when you need them. When you do, a good practice is to have all scripts grouped in a single folder (typically named cgi-bin).
- The alternative solution is to enable script execution across the web server tree, but then it is impossible to control script execution.

Enabling CGI Scripts

- To allow execution of scripts in the /var/www/cgi-bin directory,
- include the following <Directory> directive in the configuration file:
- <Directory /var/www/cgi-bin>
 - Options ExecCGI
 - SetHandler cgi-script
- </Directory>
- An alternative is to use the ScriptAlias directive, which has a similar effect:
- ScriptAlias /cgi-bin/ /var/www/cgi-bin/

Logging

- Having a record of web server activity is of utmost importance.
- Logs tell you which content is popular and whether your server is underutilized, overutilized, misconfigured, or misused.
- Two types of logs exist.
- The access log is a record of all requests sent to a particular web server or web site.

Logging

- To create an access log, you need two steps.
- First, use the LogFormat directive to define a logging format.
- Then, use the CustomLog directive to create an access log in that format:
- `LogFormat "%h %l %u %t \"%r\" %>s %b\n\"%{Referer}i\" \"%{User-Agent}i\""` combined
- `CustomLog /var/www/logs/access_log combine`

Logging

- The error log contains a record of all system events (such as web server startup and shutdown) and a record of errors that occurred during request processing.
- For example, a request for a resource that does not exist generates an HTTP 404 response for the client, one entry in the access log, and one entry in the error log.
- Two directives are required to set up the error log, just as for the access log.
- The following LogLevel directive increases the logging detail from a default value of notice to info.
- The ErrorLog directive creates the actual log file:
 - LogLevel info

SSL and TLS

- HTTP was designed under the assumption that data transmission would be secure.
- This is a perfectly valid assumption; it makes sense to put a separate communication layer in place to worry about issues such as confidentiality and data integrity.
- Unfortunately, a solution to secure data transmission was not offered at the same time as HTTP.

SSL and TLS

- By today's standards, the Internet was not a very secure place in the early days.
- It took us many years to put mechanisms in place for secure communication.
- Even today, millions of users are using insecure, plaintext communication protocols to transmit valuable, private, and confidential information

SSL and TLS

- Not taking steps to secure HTTP communication can lead to the following weaknesses:
 - Data transmission can be intercepted and recorded with relative ease.
 - For applications that require users to authenticate themselves, usernames and passwords are trivial to collect as they flow over the wire.
 - User sessions can be hijacked, and attackers can assume users' identities.

SSL and TLS

- Not taking steps to secure HTTP communication can lead to the following weaknesses:
 - Data transmission can be intercepted and recorded with relative ease.
 - For applications that require users to authenticate themselves, usernames and passwords are trivial to collect as they flow over the wire.
 - User sessions can be hijacked, and attackers can assume users' identities.

SSL and TLS

- Some cases require protection:
 - When a web site needs to collect sensitive information from its users (e.g., credit card numbers), it must ensure the communication cannot be intercepted and the information hijacked.
 - The communication between internal web applications and intranets is easy to intercept since many users share common network infrastructure (for example, the local area network).
 - Encryption is the only way to ensure confidentiality.
 - Mission-critical web applications require a maximum level of security, making encryption a mandatory requirement.

SSL and TLS

- To secure HTTP, the Secure Sockets Layer (SSL) protocol is used.
- The first commercial SSL implementation to be released was SSLv2, which appeared in 1994.
- Version 3 followed in 1995.
- Netscape also released the SSLv3 reference implementation and worked with the Internet Engineering Task Force (IETF) to turn SSL into a standard.
- The official name of the standard is Transport Layer Security (TLS), and it is defined in RFC 2246 (<http://www.ietf.org/rfc/rfc2246.txt>).
- TLS is currently at version 1.0, but that version is practically the same as SSLv3.1.

SSL and TLS

- SSL lives above TCP and below HTTP in the Open Systems Interconnection (OSI) mode.

SSL belongs to level 6 of the OSI model

	Layers	Protocols
Application	7 Application	HTTP
	6 Presentation	SSL/TLS
	5 Session	—
Transport	4 Transport	TCP
	3 Network	IP
Media	2 Data Link	Varies, e.g. PPP
	1 Physical	Varies, e.g. ADSL

SSL Communication Summary

- SSL is a hybrid protocol.
- It uses many of the cryptographic techniques described earlier to make communication secure.
- Every SSL connection consists of essentially two phases:
 - Handshake phase.
 - Data-exchange phase

Is SSL Secure?

- The answer is yes and no.
- From a technical point of view, transmission can be made secure provided proper encryption algorithms are used together with key lengths of sufficiently large sizes.
- For example, bulk encryption using the RC4 algorithm and a key length of 128 bits, with an initial handshake using 1024-bit RSA, is considered to be reasonably secure for the moment.
- But SSL can be a complex protocol to configure and use.
- Some level of knowledge is required to deploy a reasonably safe installation.

Man in the middle attacks

- Looking at the issue of SSL security from the point of view of a client who wishes to participate in an SSL session, there is a problem known as the man-in-the-middle (MITM) attack.
- MITM attacks refer to the situation where an attacker can intercept communication between two parties.
- Each party believes that it is talking to the other party but, in fact, everything goes through the attacker first.
- MITM attacks can be performed with little difficulty provided the attacker is on the same local network as the victim

Man in the middle attacks

- When a client application is preparing to establish communication with an SSL server it starts with a domain name and resolves it to the numerical IP address first.
- This is the weakest point of the process.
- Using dsniff, it is trivial to intercept domain name resolution requests and send a fake IP address (one the attacker controls) in response

Man in the middle attacks

- Believing the given IP address is correct, the client will send all traffic for that domain name to the attacker.
- The attacker will talk to the real server on the victim's behalf.
- This is all the work required to intercept nonencrypted protocols.
- But since the SSL protocol specifies server authentication in the handshake phase, the attacker needs to put in more effort when that protocol is used.

Man in the middle attacks

- The attacker cannot successfully pose as the target server since he is not in the possession of its private key.
- He can attempt to send some other certificate to the client, one for which he has the private key.

Man in the middle attacks

- There are four things the attacker can do:
 - **Use a self-signed certificate or a CA-signed certificate** that was made for some other web site.
 - This will result in a warning message being generated by the user's web browser, but the attacker may hope the user will click through it (and people do).

Man in the middle attacks

- There are four things the attacker can do:
 - **Somehow convince the user to accept his own root CA.**
 - A browser will automatically initiate the import procedure when a link to a root CA not known to the browser is encountered.
 - If the attacker is successful in having his root CA accepted, then he will be able to generate any number of certificates for any web site.

Man in the middle attacks

- There are four things the attacker can do:
 - **Take out a CA-signed certificate for the target web site by falsely posing as the target company's representative with the CA.**
 - This should be difficult since CAs are supposed to validate the identities of all who ask them to sign certificates

Man in the middle attacks

- There are four things the attacker can do:
 - **Use a root CA certificate to generate a perfectly valid certificate for the target web site if one of the root CA certificates that comes preconfigured with browsers is compromised somehow.**

Man in the middle attacks

- The only solution to MITM attacks is to enable both server and client authentication.
- In this case, the attacker will not be able to prove himself to the server as being the genuine client, and as a result the handshake phase of the session fails.

Nontechnical issues

- **It is not an end-to-end solution**
 - SSL creates a secure channel for transmission, but does not care what happens to data before it reaches the channel and after it is decrypted.
 - It secures transmission but does not secure storage.
 - Many people seem to forget this, or do not care.
 - I have seen many web sites that have SSL installed on the web server level, only to send credit card details to an email address using some form-to-email script.
 - Unattended software handling sensitive data must always use public-key cryptography to store data securely.

Nontechnical issues

- **Users lack understanding of browser warnings**
 - You will find that many end users do not care about security and do not understand the implications of their actions.
 - I have observed how people dismiss browser warnings that come up because certificates are self-signed, invalid, or expired.
 - This makes MITM attacks easy to execute.

Nontechnical issues

- **Users lack understanding of browser warnings**
 - If an attacker manages to redirect the user to his web site instead of the original, the user will blindly ignore the warning and enter the trap.
 - The solution to this is to change the way browsers behave, and make them refuse connections to sites with invalid certificates.

Nontechnical issues

- **User interfaces are inadequate**
 - Today's Internet browsers are educating users about SSL and security.
 - You typically get a small yellow icon in a corner somewhere when you connect to a secure web site.
 - That is not enough.
 - User interfaces should be changed to constantly remind the user the communication is secure, in an effort to raise awareness.
 - A good way to do this would be to have a bold red line surrounding the browser window

Nontechnical issues

- **Browsers have inadequate functionality**
 - In fact, browsers do not pay much attention to security at all.
 - Imagine an attacker who copies the design of a web site, purchases a valid certificate from a well-known CA in the name of the target web site (it has been done), and installs the web site at a server somewhere.
 - If he manages to intercept users' domain name resolution traffic (by breaking into an ISP's DNS server or by performing a MITM attack, for example), whenever someone requests the target web site he will send them to the phony version instead.

Nontechnical issues

- **Browsers have inadequate functionality**
 - Thinking she is at the correct site, the user will attempt to authenticate to the web site and thus disclose her username and password to the attacker.
 - The correct thing for a browser to do is to compare the copy of the certificate it stored upon first visit to the web site requested by the user with the copy offered to it now.
 - Any changes could result in immediate termination of the session

OpenSSL

- OpenSSL is the open source implementation (toolkit) of many cryptographic protocols.
- Almost all open source and many commercial packages rely on it for their cryptographic needs.
- OpenSSL is licensed under a BSD-like license, which allows commercial exploitation of the source code.
- You probably have OpenSSL installed on your computer if you are running a Unix system.
- If you are not running a Unix system or you are but you do not have OpenSSL installed, download the latest version from the web site (<http://www.openssl.org>).

OpenSSL

- The standard port for HTTP communication over SSL is port 443.
- To connect to a remote web server using SSL, type something like the following, where this example shows connecting to Thawte's web site:
- **\$ openssl s_client -host www.thawte.com -port 443**

OpenSSL

```

eswaribala@DESKTOP-55AGI0I:/etc/php/7.4/fpm/conf.d
eswaribala@DESKTOP-55AGI0I:/etc/php/7.4/fpm/conf.d$ openssl s_client -host www.thawte.com -port 443
CONNECTED(00000003)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert High Assurance EV Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 Extended Validation Server CA
verify return:1
depth=0 businessCategory = Private Organization, jurisdictionC = US, jurisdictionST = Utah, serialNumber = 5299537-0142, C = US, ST = Utah, L = Lehi, O = "DigiCert, Inc.", CN = digicert.com
verify return:1
---
Certificate chain
0 s:businessCategory = Private Organization, jurisdictionC = US, jurisdictionST = Utah, serialNumber = 5299537-0142, C = US, ST = Utah, L = Lehi, O = "DigiCert, Inc.", CN = digicert.com
    i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 Extended Validation Server CA
1 s:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 Extended Validation Server CA
    i:C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert High Assurance EV Root CA
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIEItjCCB56gAwIBAgIQCu5Ga1hR41i ahM0Sw heNjANBgkqhkiG9w0BAQsFADB1
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRG1naUN1cnQgSW5jMRkwFwYDVQQLExV3
d3cuZG1naWn1cnQuY29tMTQwMgYDVQQDEytEaWdpQ2VydCBTSEEyIEV4dGVuZGVk
IFZhbGlkYXRpb24gU2VdmVyIENBMB4XDTE5MTIwNDAwMDAwMFoXDTIxMTIwODEy
MDAwMFowgb0xHTAbBgNVBA8MFFByaXZhdGUgT3JnYW5pemF0aW9uMRMwEQYLKwYB
BAGCNzwCAQMTA1VTMRUwEwYLKwYBBAGCNzwCAQITBFV0YWgxFTATBgNVBAUTDDUy
OTk1MzctMDE0MjELMAkGA1UEBhMCVVwxDTALBgNVBAgTBGV0YWgxDTALBgNVBAcT
BExlaGkxFzAVBgnVBAoTDkRpZ21DZXJ0LCBjbmMuMRUwEwYDVQQDEwxkaWdpY2Vy
dC5jb20wggiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDAeRYb/RLb1jGZ
IB//DrEdyKYMqqajwBlrr3t2paAWNuDJizvVkJMIzdJesI1pA58Myenxp5Dp8GJ
u/VhBf//v/HAZHUE4xwu104Fg6A1BwUEKgVKERf+7kTt17Lf9fcMIjMyL+FeyPXb
DOFBH+ej/nYaneFLch2j2xwZg1+Thk0qB1GE8WNAK+fvbEuM0SOeH9RkYFCNGPRS
KsLn0GvaCnnD4LfNDyMqYop0IpaqXoREEnkRv1MVS0w+hBj497wnn0+/GZegfzwU
iS60h+Pj1DfmdCP18qoS7tRd0qnfU3N3s+PYEd3R63LMcIfbgXNEEWBNKpiH9+8f
-----END CERTIFICATE-----

```



Type here to search



OpenSSL

- You may be wondering what VeriSign is doing signing a Thawte certificate; Thawte is a CA.
- VeriSign recently bought Thawte; though they remain as two different business entities, they are sharing a common root certificate.

OpenSSL

- The details of the negotiated connection with the remote server are near the end of the output:
 - New, TLSv1/SSLv3, Cipher is EDH-RSA-DES-CBC3-SHA
 - Server public key is 1024 bit
 - SSL-Session:
 - Protocol : TLSv1
 - Cipher : EDH-RSA-DES-CBC3-SHA
 - Session-ID: 6E9DBBBA986C501A88F0B7ADAFEC6529291C739EB4CC2114EE62036D9B
 - F04C6E
 - Session-ID-ctx:
 - Master-Key: 0D90A33260738C7B8CBCC1F2A5DC3BE79D9D4E2FC7C649E5A541594F37
 - 61CE7046E7F5034933A6F09D7176E2B0E11605
 - Key-Ag : None
 - Krb5 Principal: None
 - Start Time: 1090586684
 - Timeout : 300 (sec)
 - Verify return code: 20 (unable to get local issuer certificate)

OpenSSL

- To find the location of the OpenSSL configuration and shared files, type:
- **\$ openssl ca**

⌚ Select eswaribala@DESKTOP-55AGI0I: /etc/php/7.4/fpm/conf.d

```
eswaribala@DESKTOP-55AGI0I:/etc/php/7.4/fpm/conf.d$ openssl ca
Using configuration from /usr/lib/ssl/openssl.cnf
```

OpenSSL

```

eswaribala@DESKTOP-55AGI0I:/usr/lib/ssl/certs
eswaribala@DESKTOP-55AGI0I:/etc/php/7.4/fpm/conf.d$ cd /usr/lib/ssl
eswaribala@DESKTOP-55AGI0I:/usr/lib/ssl$ ls
certs misc openssl.cnf private
eswaribala@DESKTOP-55AGI0I:/usr/lib/ssl$ cd certs
eswaribala@DESKTOP-55AGI0I:/usr/lib/ssl/certs$ ls
02265526.0
03179a64.0
043e3b3e
062cdee6.0
064e0aa9.0
06dc52d5.0
080911ac.0
09789157.0
0b1b94ef.0
0bf05006.0
0c4c9b6c.0
0f6fa695.0
106f3e4d.0
116bf586.0
128805a3.0
1636090b.0
18856ac4.0
1d3472b9.0
1e08bfd1.0
1e09d511.0
244b5494.0
2ae6433e.0
2b349938.0
2c543cd1.0
2e4eed3c.0
2e5ac55d.0
32888f65.0
349f2832.0
GeoTrust_Primary_Certification_Authority.pem
GeoTrust_Primary_Certification_Authority_-_G2.pem
GeoTrust_Primary_Certification_Authority_-_G3.pem
GeoTrust_Universal_CA.pem
GeoTrust_Universal_CA_2.pem
GlobalSign_ECC_Root_CA_-_R4.pem
GlobalSign_ECC_Root_CA_-_R5.pem
GlobalSign_Root_CA.pem
GlobalSign_Root_CA_-_R2.pem
GlobalSign_Root_CA_-_R3.pem
GlobalSign_Root_CA_-_R6.pem
Global_Chambersign_Root_-_2008.pem
Go_Daddy_Class_2_CA.pem
Go_Daddy_Root_Certificate_Authority_-_G2.pem
Hellenic_Academic_and_Research_Institutions_ECC_RootCA_2015.pem
Hellenic_Academic_and_Research_Institutions_RootCA_2011.pem
Hellenic_Academic_and_Research_Institutions_RootCA_2015.pem
Hongkong_Post_Root_CA_1.pem
ISRG_Root_X1.pem
IdenTrust_Commercial_Root_CA_1.pem
IdenTrust_Public_Sector_Root_CA_1.pem
Izenpe.com.pem
LuxTrust_Global_Root_2.pem
Microsec_e-Szigno_Root_CA_2009.pem
'NetLock_Arany_=Class_Gold_=Fótanúsítvány.pem'
Network_Solutions_Certificate_Authority.pem
OISTE_WISeKey_Global_Root_GA_CA.pem
OISTE_WISeKey_Global_Root_GB_CA.pem

```



Type here to search



21:45 14/08/2020 ENG 25

Apache and SSL

- If you are using Apache from the 2.x branch, the support for SSL is included with the distribution.
- You can use mod_ssl (<http://www.modssl.org>) or Apache-SSL (<http://www.apache-ssl.org>).
- The mod_ssl implementation made it into Apache 2 and is more widely used, so it makes sense to make it our choice here

Step 1 – Creating the SSL Certificate

- TLS/SSL works by using a combination of a public certificate and a private key.
- The SSL key is kept secret on the server.
- It is used to encrypt content sent to clients.
- The SSL certificate is publicly shared with anyone requesting the content.
- It can be used to decrypt the content signed by the associated SSL key.
- We can create a self-signed key and certificate pair with OpenSSL in a single command:

Step 1 – Creating the SSL Certificate

- `sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt`

Step 1 – Creating the SSL Certificate

```
eswaribala@DESKTOP-55AGI0I:/usr/lib/ssl/certs$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt
[sudo] password for eswaribala:
Generating a RSA private key
.....+++++
.....+++++
writing new private key to '/etc/ssl/private/apache-selfsigned.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:TN
Locality Name (eg, city) []:Chennai
Organization Name (eg, company) [Internet Widgits Pty Ltd]:VEBIT
Organizational Unit Name (eg, section) []:IT
Common Name (e.g. server FQDN or YOUR name) []:eswaribala
Email Address []:parameswaribala@gmail.com
eswaribala@DESKTOP-55AGI0I:/usr/lib/ssl/certs$
```

Step 1 – Creating the SSL Certificate

- **openssl**: This is the basic command line tool for creating and managing OpenSSL certificates, keys, and other files.
- **req**: This subcommand specifies that we want to use X.509 certificate signing request (CSR) management.
 - The “X.509” is a public key infrastructure standard that SSL and TLS adheres to for its key and certificate management.
 - We want to create a new X.509 cert, so we are using this subcommand.

Step 1 – Creating the SSL Certificate

- **-x509:** This further modifies the previous subcommand by telling the utility that we want to make a self-signed certificate instead of generating a certificate signing request, as would normally happen.
- **-nodes:** This tells OpenSSL to skip the option to secure our certificate with a passphrase.
 - We need Apache to be able to read the file, without user intervention, when the server starts up.
 - A passphrase would prevent this from happening because we would have to enter it after every restart.

Step 1 – Creating the SSL Certificate

- **-days 365:** This option sets the length of time that the certificate will be considered valid. We set it for one year here.
- **-newkey rsa:2048:** This specifies that we want to generate a new certificate and a new key at the same time.
 - We did not create the key that is required to sign the certificate in a previous step, so we need to create it along with the certificate.
 - The rsa:2048 portion tells it to make an RSA key that is 2048 bits long.
- **-keyout:** This line tells OpenSSL where to place the generated private key file that we are creating.
- **-out:** This tells OpenSSL where to place the certificate that we are creating.

Step 2 – Configuring Apache to Use SSL

- We have created our key and certificate files under the /etc/ssl directory.
- Now we just need to modify our Apache configuration to take advantage of these.
- We will make a few adjustments to our configuration:
 - We will create a configuration snippet to specify strong default SSL settings.
 - We will modify the included SSL Apache Virtual Host file to point to our generated SSL certificates.
 - (Recommended) We will modify the unencrypted Virtual Host file to automatically redirect requests to the encrypted Virtual Host.
- When we are finished, we should have a secure SSL configuration

Creating an Apache Configuration Snippet with Strong Encryption Settings



- First, we will create an Apache configuration snippet to define some SSL settings.
- This will set Apache up with a strong SSL cipher suite and enable some advanced features that will help keep our server secure.
- The parameters we will set can be used by any Virtual Hosts enabling SSL.
- Create a new snippet in the /etc/apache2/conf-available directory. We will name the file ssl-params.conf to make its purpose clear:
- `sudo nano /etc/apache2/conf-available/ssl-params.conf`

Creating an Apache Configuration Snippet with Strong Encryption Settings



- To set up Apache SSL securely, we will be using the recommendations by Remy van Elst on the Cipherli.st site.
- This site is designed to provide easy-to-consume encryption settings for popular software.
- Create /etc/apache2/conf-available/ssl-params.conf

Modifying the Default Apache SSL Virtual Host File

- Next, let's modify /etc/apache2/sites-available/default-ssl.conf, the default Apache SSL Virtual Host file.
- If you are using a different server block file, substitute its name in the commands below.
- Before we go any further, let's back up the original SSL Virtual Host file:
- `sudo cp /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-available/default-ssl.conf.bak`

Modifying the Default Apache SSL Virtual Host File

- sudo nano /etc/apache2/sites-available/default-ssl.conf
- We will set the normal things we'd want to adjust in a Virtual Host file (ServerAdmin email address, ServerName, etc., and adjust the SSL directives to point to our certificate and key files.

```
⌚ eswaribala@DESKTOP-55AGI0I: /etc/ssl/certs
eswaribala@DESKTOP-55AGI0I:/etc/ssl/certs$ ls *.crt
apache-selfsigned.crt  ca-certificates.crt
eswaribala@DESKTOP-55AGI0I:/etc/ssl/certs$
```

(Recommended) Modifying the HTTP Host File to Redirect to HTTPS



- As it stands now, the server will provide both unencrypted HTTP and encrypted HTTPS traffic.
- For better security, it is recommended in most cases to redirect HTTP to HTTPS automatically.
- If you do not want or need this functionality, you can safely skip this section.
- To adjust the unencrypted Virtual Host file to redirect all traffic to be SSL encrypted, we can open the /etc/apache2/sites-available/000-default.conf file:
- `sudo nano /etc/apache2/sites-available/000-default.conf`

(Recommended) Modifying the HTTP Host File to Redirect to HTTPS



- Inside, within the VirtualHost configuration blocks, we need to add a Redirect directive, pointing all traffic to the SSL version of the site:
- /etc/apache2/sites-available/000-default.conf
- <VirtualHost *:80>
- ...
- Redirect "/" "https://your_domain_or_IP/"
- ...
- </VirtualHost>

Step 3 – Adjusting the Firewall



- If you have the ufw firewall enabled, as recommended by the prerequisite guides, you might need to adjust the settings to allow for SSL traffic.
- **sudo ufw enable**
- **sudo ufw allow 'OpenSSH'**
- **Luckily, Apache registers a few profiles with ufw upon installation.**
- We can see the available profiles by typing:
- **sudo ufw app list**
- You can see the current setting by typing:
- **sudo ufw status**

Step 3 – Adjusting the Firewall



- To additionally let in HTTPS traffic, we can allow the “Apache Full” profile and then delete the redundant “Apache” profile allowance:
 - **sudo ufw allow 'Apache Full'**
 - **sudo ufw delete allow 'Apache'**
 - **sudo ufw status**

Step 4 – Enabling the Changes in Apache



- Now that we've made our changes and adjusted our firewall, we can enable the SSL and headers modules in Apache, enable our SSL-ready Virtual Host, and restart Apache.
- We can enable mod_ssl, the Apache SSL module, and mod_headers, needed by some of the settings in our SSL snippet, with the a2enmod command:
 - sudo a2enmod ssl
 - sudo a2enmod headers

Step 4 – Enabling the Changes in Apache



- Next, we can enable our SSL Virtual Host with the a2ensite command:
- sudo a2ensite default-ssl
- We will also need to enable our ssl-params.conf file, to read in the values we set:
- sudo a2enconf ssl-params
- At this point, our site and the necessary modules are enabled. We should check to make sure that there are no syntax errors in our files. We can do this by typing:
- sudo apache2ctl configtest

Step 4 – Enabling the Changes in Apache



-
- **Sudo service apache2 restart**

Step 5 – Testing Encryption

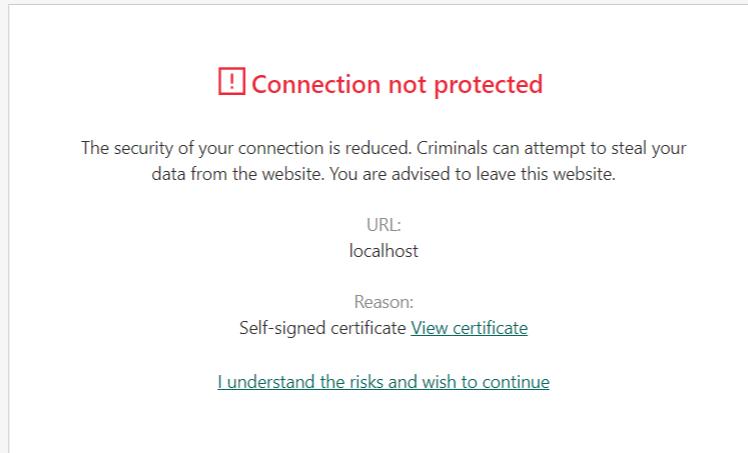


-
- Now, we're ready to test our SSL server.
 - Open your web browser and type https:// followed by your server's domain name or IP into the address bar:
 - **https://server_domain_or_IP**

https://localhost

https://localhost

Projects Gmail YouTube Maps Pluralsight



kaspersky



Aspire.com

https://localhost

Projects Gmail YouTube Maps Pluralsight

Aspire Rocks.....

That is all I have to say. If you don't see this then it doesn't work.



Managing users and authentication in Apache with PostgreSQL



- Commands to install required Apache modules bdb, authn_dbd, authn_socache and socache_shmcb:
- # Enable required modules
- sudo a2enmod dbd
- sudo a2enmod authn_dbd
- sudo a2enmod authn_socache
- sudo a2enmod socache_shmcb
- # Restart server
- sudo service apache2 restart
- # List enabled Apache modules
- apache2ctl -M

Managing users and authentication in Apache with PostgreSQL



- **Step 1 — Installing PostgreSQL**
 - sudo apt install postgresql
 - sudo service postgresql start
 - * Starting PostgreSQL 12 database server
 - sudo apt install postgresql-contrib
- **Step 2 — Using PostgreSQL Roles and Databases**
 - Switching Over to the postgres Account
 - Switch over to the postgres account on your server by typing:
 - sudo -i -u postgres
 - psql

Managing users and authentication in Apache with PostgreSQL



① Select eswaribala@DESKTOP-55AGI0I:/etc/apache2/sites-available

```
eswaribala@DESKTOP-55AGI0I:/etc/apache2/sites-available$  
eswaribala@DESKTOP-55AGI0I:/etc/apache2/sites-available$ sudo -i -u postgres  
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 4.4.0-19041-Microsoft x86_64)
```

```
* Documentation: https://help.ubuntu.com  
* Management: https://landscape.canonical.com  
* Support: https://ubuntu.com/advantage
```

System information as of Fri Aug 14 23:56:25 IST 2020

```
System load: 0.52      Users logged in: 0  
Usage of /home: unknown  IPv4 address for eth2: 172.19.96.1  
Memory usage: 58%      IPv4 address for eth4: 192.168.112.1  
Swap usage: 1%         IPv4 address for wifi1: 192.168.0.8  
Processes: 29
```

```
44 updates can be installed immediately.  
7 of these updates are security updates.  
To see these additional updates run: apt list --upgradable
```

This message is shown once once a day. To disable it please create the
/var/lib/postgresql/.hushlogin file.

```
postgres@DESKTOP-55AGI0I:~$ psql  
psql (12.2 (Ubuntu 12.2-4))  
Type "help" for help.
```

```
postgres=#
```

Managing users and authentication in Apache with PostgreSQL



- Accessing a Postgres Prompt Without Switching Accounts.
- You can also run the command you'd like with the postgres account directly with sudo.
- **sudo -u postgres psql**
- **Again, you can exit the interactive Postgres session by typing:**
- **\q**

Managing users and authentication in Apache with PostgreSQL



- **Step 3 — Creating a New Role**
- **`sudo -u postgres createuser –interactive`**
- You can get more control by passing some additional flags. Check out the options by looking at the man page:
- **`man createuser`**

Setting a password for the postgres user



- On Linux systems, there is no default password set.
- To set the default password:
 - Run the psql command from the postgres user account:
 - sudo -u postgres psql postgres
 - Set the password:
 - \password postgres
 - Enter a password.

Allowing local connections

- As a super user, open /etc/postgresql/12/main/pg_hba.conf (Ubuntu) or
- Scroll down to the line that describes local socket connections. It may look like this:
 - local all all peer
 - Change the peer method to md5.

Allowing local connections

```
eswaribala@DESKTOP-55AGI0I:/etc/postgresql/12/main$ ls
conf.d  environment pg_ctl.conf  pg_hba.conf  pg_ident.conf  postgresql.conf  start.conf
eswaribala@DESKTOP-55AGI0I:/etc/postgresql/12/main$ sudo nano pg_hba.conf
eswaribala@DESKTOP-55AGI0I:/etc/postgresql/12/main$ sudo service postgresql restart
* Restarting PostgreSQL 12 database server
[ OK ]
eswaribala@DESKTOP-55AGI0I:/etc/postgresql/12/main$
eswaribala@DESKTOP-55AGI0I:/etc/postgresql/12/main$ psql -U postgres -W
Password:
psql (12.2 (Ubuntu 12.2.4))
Type "help" for help.

postgres=# \dt
      List of relations
 Schema |     Name      | Type  | Owner
-----+-----+-----+-----+
 public | auth        | table | root
 public | playground  | table | root
(2 rows)

postgres=# ■
```

Managing users and authentication in Apache with PostgreSQL



- **Step 4 — Creating a New Database**
- `sudo -u postgres createdb testdb`
- **Step 5 — Opening a Postgres Prompt with the New Role**
- **`sudo adduser user1`**
- If you want your user to connect to a different database, you can do so by specifying the database like this:
- **`psql -d postgres`**

Managing users and authentication in Apache with PostgreSQL



- Once logged in, you can get check your current connection information by typing:
- \conninfo**
- Step 6 — Creating and Deleting Tables**
- CREATE TABLE playground (
 - equip_id serial PRIMARY KEY,
 - type varchar (50) NOT NULL,
 - color varchar (25) NOT NULL,
 - location varchar(25) check (location in ('north', 'south', 'west', 'east', 'northeast', 'southeast', 'southwest', 'northwest')),
 - install_date date
-);

Managing users and authentication in Apache with PostgreSQL



- To view table
- \d
- If you want to see just the table without the sequence, you can type:
- \dt
- Step 7 — Adding, Querying, and Deleting Data in a Table
- `INSERT INTO playground (type, color, location, install_date) VALUES ('slide', 'blue', 'south', '2017-04-28');`

Managing users and authentication in Apache with PostgreSQL



- `INSERT INTO playground (type, color, location, install_date) VALUES ('swing', 'yellow', 'northwest', '2018-08-16');`
- `SELECT * FROM playground;`

Managing users and authentication in Apache with PostgreSQL



- To view table
- \d
- If you want to see just the table without the sequence, you can type:
- \dt
- Step 7 — Adding, Querying, and Deleting Data in a Table

Postgres Db for Apache2 Auth

```
① eswaribala@DESKTOP-55AGI0l: /etc/apache2/sites-available
postgres=# create table auth (username varchar(255) primary key not null, passwd varchar(255), groups varchar(255))
postgres-# \dt
      List of relations
 Schema |     Name      | Type | Owner
-----+-----+-----+-----+
public | auth        | table | root
public | playground  | table | root
(2 rows)

postgres-# ■
```

Postgres Db for Apache2 Auth



eswaribala@DESKTOP-55AGI0I: /etc/apache2/sites-available

```
eswaribala@DESKTOP-55AGI0I:/etc/apache2/sites-available$ htpasswd -bnS test test
test:{SHA}qUqP5cyxm6YcTAhz05Hph5gvu9M=
```

```
eswaribala@DESKTOP-55AGI0I:/etc/apache2/sites-available$
```

Postgres Db for Apache2 Auth

```
eswaribala@DESKTOP-55AGI0I:/etc/apache2/sites-available
```

```
eswaribala@DESKTOP-55AGI0I:/etc/apache2/sites-available$ sudo psql -d postgres
psql (12.2 (Ubuntu 12.2-4))
Type "help" for help.

postgres=# INSERT INTO auth (username, passwd, groups) VALUES('test','{SHA}qUqP5cyxm6YcTAhz05Hph5gvu9M=','testgroup');
INSERT 0 1
postgres=# ■
```

Postgres Db for Apache2 Auth

```
⌚ eswaribala@DESKTOP-55AGI0I: /etc/apache2/sites-available
```

```
INSERT 0 1
postgres=# SELECT * FROM auth;
 username |          passwd          |    groups
-----+-----+-----+
 test   | {SHA}qUqP5cyxm6YcTAhz05Hph5gvu9M= | testgroup
(1 row)

postgres=#
```

Postgres Db for Apache2 Auth

- sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/000-default.conf.bkp
- Edit 000-default.conf file adding dbd params
- sudo apt-get install libaprutil1-dbd-pgsql

Advanced PHP Configuration

- Step 1: Determine the Location of your php.ini File
 - From time to time, you will need to manage the critical php.ini file.
 - The file location may vary a bit depending on the PHP version running on your server.
 - The configuration file is located on:
 - /etc/php/php_version/apache2/php.ini
- Step 2: Opening the PHP configuration file for editing
- Step 3: Making changes on the php.ini file

Advanced PHP Configuration

- The default PHP settings may work for the majority of websites or web applications.
- However, your environment may demand some values to be tweaked a bit to ensure that your website is running smoothly.
- In most cases, you will be editing the below PHP settings:

Advanced PHP Configuration

- PHP max_execution_time
 - This sets the maximum execution time in seconds that a PHP script is allowed to run before it is terminated.
 - Sometimes, you might have demanding scripts that should run for a few minutes and you need to change this value.
 - The default value is 30 seconds but you can set it to a larger value
 - Default value:
 - **max_execution_time =30**
 - Change to any value e.g. 1800
 - **max_execution_time =1800**

Advanced PHP Configuration

- PHP upload_max_filesize
 - The default value for this directive is 2M (two Megabytes).
 - This value controls the maximum size of files that you upload using PHP scripts.
 - Sometimes, it is necessary to change this value if you anticipate uploading big files.

Advanced PHP Configuration

- PHP upload_max_filesize
 - For instance, if you are uploading a large database via phpMyAdmin, you will need to change this value.
 - Default value:
 - **upload_max_filesize=2M**
 - Change to a large value e.g. 16M
 - `upload_max_filesize=16M`

Advanced PHP Configuration

- PHP post_max_size
- This value limits the amount of data allowed on post data.
- It usually affects PHP scripts that use a lot of web forms.
- The value also controls files uploaded via a PHP script, hence, it should always be larger than ‘upload_max_filesize’.
- The default value for ‘post_max_size’ is 8M.
- Default value:
 - **post_max_size =8M**
- Customize it depending on your needs e.g.
 - **post_max_size =32**

Advanced PHP Configuration

- PHP memory_limit
- The default value for PHP 7.2 ‘memory_limit’ is 128M.
- Sometimes, poorly written PHP scripts may consume a lot of server’s memory and affect other applications running on your VPS.
- To avoid this, PHP ‘memory_limit’ controls the amount of memory allocated to a script.
- Default value
 - **memory_limit = 128M**
- Custom value example
 - **memory_limit = 256M**

Advanced PHP Configuration

- PHP memory_limit
- You can also use -1 if you want to allocate an unlimited amount of memory to your PHP script depending on the available RAM on your VPS
- **memory_limit = -1**

Advanced PHP Configuration

- PHP Error Reporting Settings
- You can control the behavior of error reporting in PHP using the below directives:
- `display_errors`:> Set this value to ‘On’ or ‘Off’ depending on whether you want PHP to display errors when scripts are run. In PHP 7.2 the default value is ‘Off’
 - **display_errors = Off**
 - You can turn error reporting on by changing the value to ‘On’:
 - `display_errors = On`

Advanced PHP Configuration

- `log_errors`: This value tells whether errors from a script should be saved on the server's log file.
- Instead of displaying errors to regular users in a production environment, you should log them.
- The default value in PHP 7.4 is 'On'
- **`log_errors = On`**
- You can switch error logging off by changing the value to:
- `log_errors = Off`

Advanced PHP Configuration

- `error_reporting`: This directive dictates the error reporting level.
- For PHP versions greater than 5.3, the default value is '`E_ALL & ~E_DEPRECATED & ~E_STRICT`'
 - **`error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT`**
- You may change the value depending on the errors that you want to be reported.
- For instance, to include notices, use the value below
 - **`error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT & ~E_NOTICE`**

Advanced PHP Configuration

- PHP Date/Time settings
- You can also change the default timezone used by PHP scripts.
- Find the line:
- **\$; date.timezone=**
- Uncomment it by removing the semicolon and then enter your preferred time zone.
- You can check the list of support time zones on the official PHP website (<http://php.net/manual/en/timezones.php>)
- For instance, if you want to change the time zone to New York City, use the value below:
- **date.timezone= "America/New_York"**
- Once you finish editing the php.ini file, press CTRL + X, Y and hit Enter to save the changes.

Find Built-in PHP Modules

- To see the set of compiled-in PHP modules type the following command:
- # php -m
- PHP User and Group ID
- ps aux | grep php-cgi

Find Built-in PHP Modules

- **Limit PHP Access To File System**
- The open_basedir directive set the directories from which PHP is allowed to access files using functions like fopen(), and others.
- If a file is outside of the paths defined by open_basedir, PHP will refuse to open it.
- You cannot use a symbolic link as a workaround.
- For example only allow access to /var/www/html directory and not to /var/www, or /tmp or /etc directories:
- ; Limits the PHP process from accessing files outside
- ; of specifically designated directories such as /var/www/html/
- open_basedir="/var/www/html/"
- ; -----
- ; Multiple dirs example
- ;
open_basedir="/home/httpd/vhost/cyberciti.biz/html/:/home/httpd/vhost/nixcraft.com/html/:/home/httpd/vhost/theos.in/html/"
- ; -----

Apache 2 Security

- **Step 1 — Installing the Apache Utilities Package**
- sudo apt update
- sudo apt install apache2-utils
- **Step 2 — Creating the Password File**
- sudo htpasswd -c /etc/apache2/.htpasswd eswari
- sudo htpasswd /etc/apache2/.htpasswd another_user
- cat /etc/apache2/.htpasswd

Apache 2 Security

- **Step 3 — Configuring Apache Password Authentication**
- Option 1: Configuring Access Control within the Virtual Host Definition (Preferred)
- sudo nano /etc/apache2/sites-enabled/default-ssl.conf
- Refer defaultssl
- sudo apache2ctl configtest
- sudo systemctl restart apache2

Apache 2 Security

- **Step 3 — Configuring Apache Password Authentication**
- Option 2: Configuring Access Control with .htaccess Files
- `sudo nano /etc/apache2/apache2.conf`
- `<Directory /var/www/>`
- `Options Indexes FollowSymLinks`
- `AllowOverride All`
- `Require all granted`
- `</Directory>`

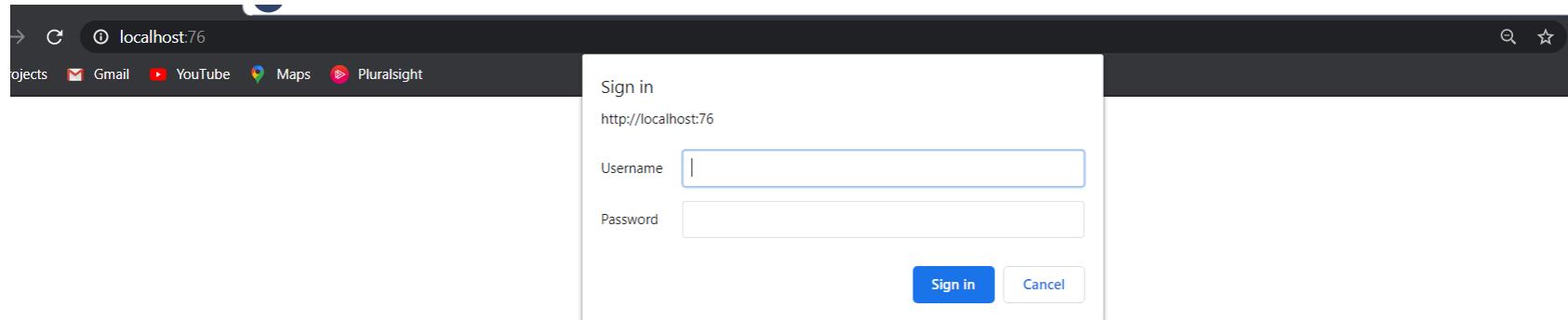
Apache 2 Security

- **Step 3 — Configuring Apache Password Authentication**
- Option 2: Configuring Access Control with .htaccess Files
- `sudo nano /var/www/html/.htaccess`
- `/var/www/html/.htaccess`
- `AuthType Basic`
- `AuthName "Restricted Content"`
- `AuthUserFile /etc/apache2/.htpasswd`
- `Require valid-user`

Apache 2 Security

- **Step 3 — Configuring Apache Password Authentication**
- **sudo systemctl restart apache2**
- **sudo systemctl status apache2**

Apache 2 Security



Ubuntu Linux Add a User To Group www-data (Apache Group)



- The procedure is as follows:
- Open the terminal application.
- Login to Ubuntu server using ssh.
- Add a new user named foo to www-data group by running useradd -g www-data foo command.
- Add an existing user bar to Apache's www-data group on Ubuntu using usermod -a -G www-data bar command.
- Verify new modification on Linux by using the id command.

Ubuntu Linux Add a User To Group www-data (Apache Group)



- How to see a list of www-data members
- grep ^www-data /etc/group
- How do I add an existing user named vivek to group Apache group www-data?
- sudo adduser {USER-NAME-HERE} {GROUP-NAME-HERE}
- sudo adduser vivek www-data
- id vivek

Ubuntu Linux Add a User To Group www-data (Apache Group)



- Add a new user to the www-data group
- sudo useradd -g www-data vivek
- ### set the password for vivek user ###
- sudo passwd vivek
- id vivek
- groups vivek

Ubuntu Linux Add a User To Group www-data (Apache Group)



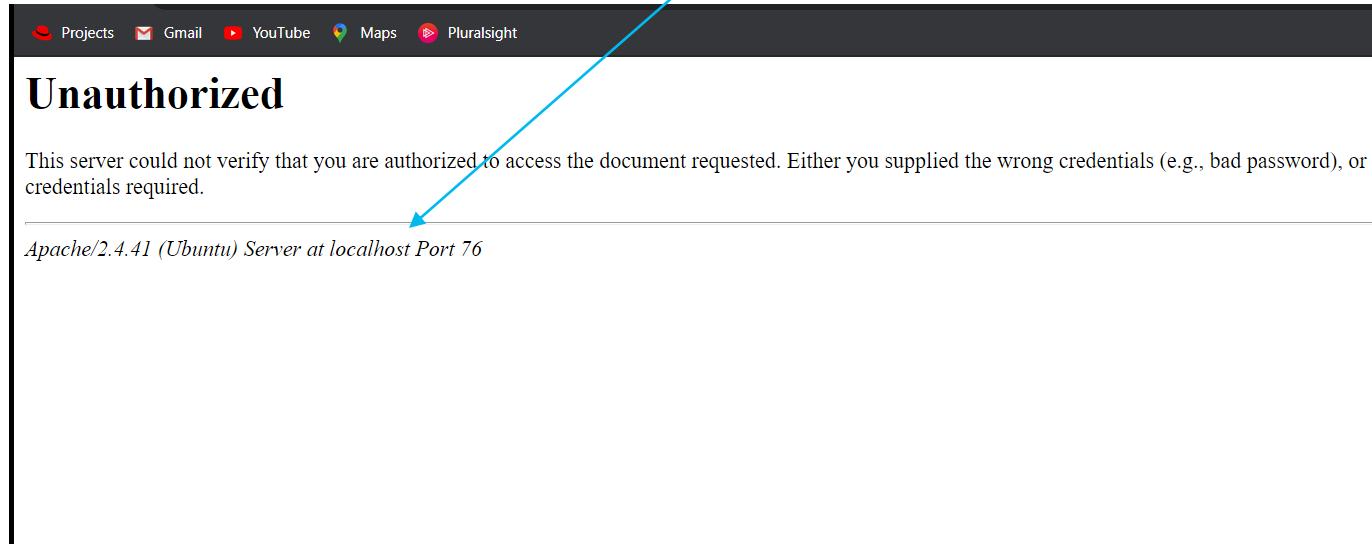
- www-data supplementary groups membership
- To add a new user called vivek to supplementary groups called www-data and ftp, enter:
- sudo groupadd vivek
- sudo useradd -g vivek -G www-data,ftp vivek
- sudo passwd vivek
- id vivek
- groups vivek

Ubuntu Linux Add a User To Group www-data (Apache Group)



- Add a existing user to www-data group
- Type the following command to add an existing user called vgite to the www-data group, enter:
- `sudo usermod -a -G www-data vgite`
- `id vgite`
- `groups vgite`

Apache Web Server Security and Hardening



Apache Web Server Security and Hardening

- In above picture, you can see that Apache is showing its version with the OS installed in your server.
- This can be a major security threat to your web server as well as your Linux box too.
- To prevent Apache to not to display these information to the world, we need to make some changes in Apache main configuration file.
- Open configuration file with nano editor and search for “ServerSignature”, its by default On.
- We need to Off these server signature and the second line “ServerTokens Prod” tells Apache to return only Apache as product in the server response header on the every page request, It suppress the OS, major and minor version info.

Apache Web Server Security and Hardening

- ServerSignature Off
- ServerTokens Prod

Apache Web Server Security and Hardening

- **Disable Directory Listing**
- By default Apache list all the content of Document root directory in the absence of index file. Please see the image below.
- We can turn off directory listing by using Options directive in configuration file for a specific directory.
- For that we need to make an entry in httpd.conf or apache2.conf file.
 - <Directory /var/www/html>
 - Options -Indexes
 - </Directory>

Apache Web Server Security and Hardening

- **Keep updating Apache Regularly**
- apt-get install apache2
- Disable Unnecessary Modules
- Run Apache as separate User and Group
- Create Apache User and Group
- # groupadd http-web
- # useradd -d /var/www/ -g http-web -s /bin/nologin
http-web

Apache Web Server Security and Hardening

- Now you need to tell Apache to run with this new user and to do so, we need to make an entry in /etc/httpd/conf/httpd.conf and restart the service.
- Open /etc/httpd/conf/httpd.conf with vim editor and search for keyword “User” and “Group” and there you will need to specify the username and groupname to use.
 - User http-web
 - Group http-web

Apache Web Server Security and Hardening

- **Use Allow and Deny to Restrict access to Directories**
- We can restrict access to directories with “Allow” and “Deny” options in httpd.conf file. Here in this example, we’ll be securing root directory, for that by setting the following in the httpd.conf file.
- <Directory />
- Options None
- Order deny,allow
- Deny from all
- </Directory>

Apache Web Server Security and Hardening

- Options “None” – This option will not allow users to enable any optional features.
- Order deny, allow – This is the order in which the “Deny” and “Allow” directives will be processed. Here it will “deny” first and “allow” next.
- Deny from all – This will deny request from everybody to the root directory, nobody will be able to access root directory.

Apache Web Server Security and Hardening

- Use mod_security and mod_evasive Modules to Secure Apache
- These two modules “mod_security” and “mod_evasive” are very popular modules of Apache in terms of security.
- Mod_security
- Where mod_security works as a firewall for our web applications and allows us to monitor traffic on a real time basis.
- It also helps us to protect our websites or web server from brute force attacks.
- You can simply install mod_security on your server with the help of your default package installers.

Apache Web Server Security and Hardening

- **Install mod_security on Ubuntu/Debian**
- \$ sudo apt-get install libapache2-modsecurity
- \$ sudo a2enmod mod-security
- \$ sudo /etc/init.d/apache2 force-reload

Apache Web Server Security and Hardening

- **Mod_evasive**
- mod_evasive works very efficiently, it takes one request to process and processes it very well.
- It prevents DDOS attacks from doing as much damage.
- This feature of mod_evasive enables it to handle the HTTP brute force and Dos or DDos attack.
- This module detects attacks with three methods.
- If so many requests come to a same page in a few times per second.
- If any child process trying to make more than 50 concurrent requests.

Apache Web Server Security and Hardening

- If any IP still trying to make new requests when its temporarily blacklisted.
- mod_evasive can be installed directly from the source.
- Here, we have an Installation and setup guide of these modules which will help you to set up these Apache modules in your Linux box.
- https://www.tecmint.com/protect-apache-using-mod_security-and-mod_evasive-on-rhel-centos-fedora/

Apache Web Server Security and Hardening

- **8. Disable Apache's following of Symbolic Links**
- By default Apache follows symlinks, we can turn off this feature with FollowSymLinks with Options directive.
- And to do so we need to make the following entry in main configuration file.
- Options -FollowSymLinks
- And, if any particular user or website need FollowSymLinks enable, we can simply write a rule in “.htaccess” file from that website.
- # Enable symbolic links
- Options +FollowSymLinks

Apache Web Server Security and Hardening

- **Turn off Server Side Includes and CGI Execution**
- We can turn off server side includes (mod_include) and CGI execution if not needed and to do so we need to modify main configuration file.
- Options -Includes
- Options -ExecCGI
- We can do this for a particular directory too with Directory tag. Here In this example, we are turning off Includes and Cgi file executions for “/var/www/html/web1” directory.
- <Directory "/var/www/html/web1">
- Options -Includes -ExecCGI
- </Directory>

Apache Web Server Security and Hardening

- **Turn off Server Side Includes and CGI Execution**
- We can turn off server side includes (mod_include) and CGI execution if not needed and to do so we need to modify main configuration file.
- Options -Includes
- Options -ExecCGI
- We can do this for a particular directory too with Directory tag. Here In this example, we are turning off Includes and Cgi file executions for “/var/www/html/web1” directory.
- <Directory "/var/www/html/web1">
- Options -Includes -ExecCGI
- </Directory>

Apache Web Server Security and Hardening

- Here are some other values with can be turned On or off with Options directive.
- Options All – To enable All options at once. This is the default value, If you don't want specify any values explicitly in Apache conf file or .htaccess.
- Options IncludesNOEXEC – This option allows server side includes without the execute permission to a command or cgi files.
- Options MultiViews – Allows content negotiated multiviews with mod_negotiation module.
- Options SymLinksIfOwnerMatch – It's similar to FollowSymLinks. But, this will follow only when the owner is the same between the link and the original directory to which it is linked.

Apache Web Server Security and Hardening

- **Limit Request Size**
- By default Apache has no limit on the total size of the HTTP request i.e. unlimited and when you allow large requests on a web server its possible that you could be a victim of Denial of service attacks.
- We can Limit the requests size of an Apache directive “LimitRequestBody” with the directory tag.
- You can set the value in bytes from 0 (unlimited) to 2147483647 (2GB) that are allowed in a request body.
- You can set this limit according to your site needs, Suppose you have a site where you allows uploads and you want to limit the upload size for a particular directory.
- Here in this example, user_uploads is a directory which contains files uploaded by users. We are putting a limit of 500K for this.
- `<Directory "/var/www/myweb1/user_uploads">`
- `LimitRequestBody 512000`
- `</Directory>`

Apache Web Server Security and Hardening

- **Protect DDOS attacks and Hardening**
- You cannot completely protect your web site from DDos attacks. Here are some directives which can help you to have a control on it.
- TimeOut : This directive allows you to set the amount of time the server will wait for certain events to complete before it fails.
- Its default value is 300 secs.
- It's good to keep this value low on those sites which are subject to DDOS attacks.
- This value totally depends on kind of request you are getting on your website. Note: It could pose problems with some CGI scripts.

Apache Web Server Security and Hardening

- MaxClients : This directive allows you to set the limit on connections that will be served simultaneously.
- Every new connection will be queued up after this limit.
- It is available with Prefork and Worker both MPM. The default value of it is 256.
- KeepAliveTimeout : Its the amount of time the server will wait for a subsequent request before closing the connection. Default value is 5 secs.
- LimitRequestFields : It helps us to set a limit on the number of HTTP request's header fields that will be accepted from the clients.
- Its default value is 100. It is recommended to lower this value if DDos attacks are occurring as a result of so many http request headers.
- LimitRequestFieldSize : It helps us to set a size limit on the HTTP Request header.

Apache Web Server Security and Hardening

- **Enable Apache Logging**
- Apache allows you to logging independently of your OS logging.
- It is wise to enable Apache logging, because it provides more information, such as the commands entered by users that have interacted with your Web server.
- To do so you need to include the mod_log_config module. There are three main logging-related directives available with Apache.
- TransferLog: Creating a log file.
- LogFormat : Specifying a custom format.
- CustomLog : Creating and formatting a log file.

Apache Web Server Security and Hardening

- You can also use them for a particular website if you are doing Virtual hosting and for that you need to specify it in the virtual host section.
- For example, here is my website virtual host configuration with logging enabled.
- <VirtualHost *:80>
- DocumentRoot /var/www/html/example.com/
- ServerName www.example.com
- DirectoryIndex index.htm index.html index.php
- ServerAlias example.com
- ErrorDocument 404 /story.php
- ErrorLog /var/log/httpd/example.com_error_log
- CustomLog /var/log/httpd/example.com_access_log combined
- </VirtualHost>

Apache Web Server Security and Hardening

- **Securing Apache with SSL Certificates**
- you can secure your all the communication in an encrypted manner over the Internet with SSL certificate.
- Suppose you have a website in which people login by proving their Login credentials or you have an E- Commerce website where people provides their bank details or Debit/Credit card details to purchase products, by default your web server send these details in plain – text format but when you use SSL certificates to your websites, Apache sends all this information in encrypted text.
- You can purchase SSI certificates from So many different SSL providers like namecheap.com.
- If you are running a very small web business and do not willing to purchase an SSL certificate you can still assign a Self signed certificate to your website.
- Apache uses the mod_ssl module to support SSL certificate.

Apache Web Server Security and Hardening

- # openssl genrsa -des3 -out example.com.key 1024
- # openssl req -new -key example.com.key -out exmaple.csr
- # openssl x509 -req -days 365 -in example.com.com.csr -signkey example.com.com.key -out example.com.com.crt

Apache Web Server Security and Hardening

- Once your certificate has been created and signed. Now you need to add this in Apache configuration. Open main configuration file with vim editor and add the following lines and restart the service.
- <VirtualHost 172.16.25.125:443>
- SSLEngine on
- SSLCertificateFile /etc/pki/tls/certs/example.com.crt
- SSLCertificateKeyFile /etc/pki/tls/certs/example.com.key
- SSLCertificateChainFile /etc/pki/tls/certs/sf_bundle.crt
- ServerAdmin ravi.saive@example.com
- ServerName example.com
- DocumentRoot /var/www/html/example/
- ErrorLog /var/log/httpd/example.com-error_log
- CustomLog /var/log/httpd/example.com-access_log common
- </VirtualHost>

Questions



Module Summary

- Web Server Administration
- Apache virtual host workflow and configuration in detail
- Apache MPM configurations and usages
- Server capacity (like RAM, ROM) needed for n number of users (say n=3000+).
- Apache tuning when there is high server load
- Other Apache mechanism for handling uncertain situations

