FREE eBook

LEARNING apache

Free unaffiliated eBook created from
Stack Overflow contributors.

#apache

Table of Contents

About 1

Chapter 1: Getting started with apache 2

Remarks 2

Versions 2

Various Apache httpd releases 2

Examples 2

Installation or Setup 2

Ubuntu Installation 2

Windows Installation 2

CentOS Installation 2

macOS Installation 3

[Ubuntu] Simple Hello World Example 3

Installing Requirements 3

Setting up the HTML 3

Visiting Your Webpage 4

To ensure the server is up. 4

Chapter 2: .htaccess files in Apache 5

Examples 5

Rewrite Engine 5

Force HTTPS 5

Enable CORS 6

Prerequisites 7

301 Redirection by Htaccess 7

Chapter 3: Apache Flume 8

Introduction 8

Examples 8

Streaming / Log Data 8

Chapter 4: How to create virtual host in Apache 9

Remarks 9

Examples 9

Name-based virtual host configuration 9
PHP Development Virtual Host 10
Virtual Host In WAMP 11
1) IP based vhosts 2) Multiple vhosts with the same Port 3) Defining vhosts using Macro (A 12
Force HTTPS using virtual host 13
Credits **14**

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: apache

It is an unofficial and free apache ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official apache.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with apache

Remarks

This section provides an overview of what apache is, and why a developer might want to use it.

It should also mention any large subjects within apache, and link out to the related topics. Since the Documentation for apache is new, you may need to create initial versions of those related topics.

Versions

Various Apache httpd releases

Version	Current version	Release
1.3	1.3.42	1998-06-06
2.0	2.0.65	2002-04-06
2.2	2.2.32	2005-12-01
2.4	2.4.25	2012-02-21

Examples

Installation or Setup

Detailed instructions on getting apache set up or installed.

Ubuntu Installation

```
sudo apt-get install apache2
```

Windows Installation

Check out the WAMP stack. WAMP stands for Windows, Apache, MySQL, PhpMyAdmin.

CentOS Installation

Apache 2.2 comes with CentOS6, whereas 2.4 comes with CentOS7, to install on either OS, run

```
yum -y install httpd
```

macOS Installation

macOS comes with Apache pre-installed,however,can install Apache via Homebrew

If you already have the built-in Apache running, it will need to be shutdown first, and any auto-loading scripts removed.

```
$ sudo apachectl stop
$ sudo launchctl unload -w /System/Library/LaunchDaemons/org.apache.httpd.plist 2>/dev/null
$ brew install httpd24 --with-privileged-ports --with-http2
```

[Ubuntu] Simple Hello World Example

This example will guide you through setting up a back end serving an a Hello World HTML page.

Installing Requirements

Order matters for this step!

- `sudo apt-get install apache2`

Setting up the HTML

Apache files live in `/var/www/html/`. Lets quickly get there. Make sure you're in your root directory first, `cd`, then `cd /var/www/html/`.

This `html` directory is where all your website files will live. Lets quickly make a simple Hello World file.

Using your favorite text editor, type the following in

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello World!</title>
</head>
<body>
    <h1>Hello World!</h1>
</body>
</html>
```

Save this file as `index.html` in the current directory and you're set to go!

Visiting Your Webpage

To visit the page you just created, in your browser of choice, go to `localhost`. If that doesn't work, try `127.0.0.1`. You should see "Hello World!" as a `h1`. You're done!

To ensure the server is up.

If you get a message that the browser can't connect to the server, first check to ensure the server is up.

```
$ ps -aef | grep httpd
```

You should see a few httpd processes if Apache is up and running.

Read Getting started with apache online: https://riptutorial.com/apache/topic/964/getting-started-with-apache

Chapter 2: .htaccess files in Apache

Examples

Rewrite Engine

The RewriteEngine module within Apache is used to dynamically rewrite URLs and paths depending on various expressions provided:

```
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [END]
</IfModule>
```

The above rules will rewrite PHP files to no longer show their extension, and so that index.php will just show as a naked domain (similar to the behavior normally seen in index.html). The above rule ships with WordPress.

Note that in Apache httpd 2.2.16 and later, this entire block can be replaced with a single line using the FallbackResource directive:

```
FallbackResource /index.php
```

Force HTTPS

.htaccess can be used to force your HTTP site to redirect to HTTPS.

Here's a quick way that doesn't require editing the code for your domain:

```
RewriteEngine On
RewriteCond %{HTTPS} =off
RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

> **Warning:** The code above assumes that you can trust `%{HTTP_HOST}` to point to your domain.
>
> If you need to be sure that the redirect location is your domain, replace `%{HTTP_HOST}` with your domain.

The code above does this:

1. Enable RewriteEngine.
2. Continue if the current request is not using HTTPS.
3. Do a HTTP 301 redirect to `https://%{HTTP_HOST}%{REQUEST_URI}`, where

- `%{HTTP_HOST}` is the host requested by the browser and
- `%{REQUEST_URI}` is the URI requested by the browser (everything after the domain).

Warning: Your web application must be able to handle HTTPS requests, and Apache for your host should be configured with a valid site certificate.

Note that it is significantly more efficient to simply do a `Redirect` in the http vhost than to do these multiple per-request comparisons in a .htaccess file. See http://wiki.apache.org/httpd/RedirectSSL for further discussion of this technique.

Enable CORS

To enable ***Cross-Origin Resource Sharing*** (**CORS**) in Apache you'll need to set at least one HTTP header which changes it (the default behaviour is to block CORS). In the following example, we're going to be setting this HTTP header inside `.htaccess`, but it can also be set in your site `your-site.conf` file or the Apache config file. Regardless of how your configuration looks like, you can set the relevant HTTP headers in any Apache config block, i.e. `<VirtualHost>`, `<Directory>`, `<Location>`, and `<Files>`.

There are a few CORS related HTTP headers which you can return in the response:

```
Access-Control-Allow-Origin

Access-Control-Allow-Credentials
Access-Control-Allow-Methods
Access-Control-Max-Age
Access-Control-Allow-Headers
Access-Control-Expose-Headers
```

Some of the above are required for "preflight" requests. Some HTTP clients (namely, modern browsers) perform a request **before** your desired request just to see if they have authorisation to make the actual request on the server. See https://en.wikipedia.org/wiki/Cross-origin_resource_sharing for more on the preflight request.

The main HTTP header we need is `Access-Control-Allow-Origin` and that's we're going to set. However, the same principle applies pretty much to all of them (you just need to know what to return).

The following example sets the required HTTP header within a `<Directory>` config block to enable an SSL protected client Full Qualified Domain Name (FQDN):

```
<Directory /path/to/your/site/>
        Header set Access-Control-Allow-Origin "https://my.CLIENT.domain"
</Directory>
```

After we've set this on the **server**, we can now perform a request from https://my.client.domain to our server and it should respond.

Note: A lot of people use `Access-Control-Allow-Origin: "*"` which is a wildcard, to mean requests from **ALL** domains should be accepted. This is usually ill-advised unless you're running some sort

of a public API or repository of files. Also, please note the context of you HTTP header setting. You might want to allow HTTP requests for an API, but not for "hotlinking" images etc. You can set this header anywhere you want within your Apache config flow to **only** set it in specific situations. For example, the following would **only** set the CORS HTTP header when the requested path is **not** a file or directory (suits a public API which disallows image hotlinking):

```
<Directory /path/to/your/site/>
    Options +FollowSymlinks
    Options +Indexes
    RewriteEngine On

    #Make sure it's not a specific file or directory that they're trying to reach
    RewriteCond %{SCRIPT_FILENAME} !-f
    RewriteCond %{SCRIPT_FILENAME} !-d
    Header set Access-Control-Allow-Origin "*"
    RewriteRule ^(.*)$ index.php/$1 [L]
</Directory>
```

Prerequisites

You've got to have mod_headers installed and enabled: `a2enmod headers`

301 Redirection by Htaccess

The HTTP response status code 301 Moved Permanently is used for permanent URL redirection, meaning current links or records using the URL that the response is received for should be updated. The new URL should be provided in the Location field included with the response. The 301 redirect is considered a best practice for upgrading users from HTTP to HTTPS. write this code in htaccess file for PHP-APACHE

```
Redirect 301 /oldpage/ /newpage/
```

Here is an example using an htaccess file to redirect to a non www with an SSL attached to the domain.

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteCond %{HTTP_HOST} ^www\.(.*)$ [NC]
RewriteRule ^(.*)$ http://%1/$1 [R=301,L]

RewriteCond %{HTTPS} on
RewriteCond %{HTTP_HOST} ^www\.(.*)$ [NC]
RewriteRule ^(.*)$ https://%1/$1 [R=301,L]

RewriteEngine On
RewriteCond %{SERVER_PORT} 80
RewriteRule ^(.*)$ https://example.com/$1 [R,L]
```

Read .htaccess files in Apache online: https://riptutorial.com/apache/topic/2089/-htaccess-files-in-apache

Chapter 3: Apache Flume

Introduction

`Apache Flume` is a tool/service/data ingestion mechanism for collecting aggregating and transporting large amounts of streaming data such as log files, events (etc...) from various sources to a **centralized data store**.

Flume is a highly reliable, distributed, and configurable tool. It is principally designed to copy streaming data (log data) from various web servers to `HDFS`.

Examples

Streaming / Log Data

Generally, most of the data that is to be analyzed will be produced by various data sources like applications servers, social networking sites, cloud servers, and enterprise servers. This data will be in the form of log files and events.

Log file – In general, a log file is a file that lists events/actions that occur in an operating system. For example, web servers list every request made to the server in the log files.

On harvesting such log data, we can get information about –

the application performance and locate various software and hardware failures. the user behavior and derive better business insights. The traditional method of transferring data into the HDFS system is to use the put command. Let us see how to use the put command.

Read Apache Flume online: https://riptutorial.com/apache/topic/9630/apache-flume

Chapter 4: How to create virtual host in Apache

Remarks

The main entry point for Apache's `VirtualHost` is at Apache Virtual Host documentation. From there, you have general documentation about virtual host configuration, and reference documentation about `VirtualHost` and related directives as well.

Examples

Name-based virtual host configuration

Name-based virtual hosting on Apache is described on the Apache website as such:

> With name-based virtual hosting, the server relies on the client to report the hostname as part of the HTTP headers. Using this technique, many different hosts can share the same IP address.

Therefore, more than one website can be hosted on one server through this method. On ubuntu, the configuration files are in /etc/apache2/sites-available. In that directory, you will find 000-default.conf. That is the default configuration, all requests will be sent to this configuration file until others have been set up.

To set up a virtual host, here **example.com** will be used, but you should replace it with your **domain.com**. Copy the default file:

```
cp 000-default.conf example.com.conf
```

The configuration file can have the following directives:

```
<VirtualHost *:80>
        ServerAdmin admin@example.com
        ServerName example.com
        ServerAlias www.example.com

        DocumentRoot /var/www/example.com/html

        ErrorLog /var/log/apache/logs/error.log
        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        CustomLog /var/log/apache/logs/access.log combined
</VirtualHost>
```

- The first line, indicates that all requests on port 80 (default http port) should be matched. You

can also have a IP address instead of * which is the IP of the server.
- `ServerAdmin` is the contact details of website admin used for displaying with http error messages.
- `ServerName` is the domain name of website.
- `ServerAlias` is a secondary name of the website, usually will be www.domain.com
- `DocumentRoot` is the root folder loaded when we browse a website.
- `ErrorLog` is the file in where errors are directed
- `LogLevel`. is the level of errors to be sent to the log
- `CustomLog` is the file where access information is directed

Edit the file replacing example.com with your website domain name and appropriate directory for the website files.

Save the file and enable the site with the following Apache command:

```
sudo a2ensite example.com.conf
```

Reload apache

```
sudo service apache2 reload
```

A few more things that must be checked:

- Ensure your DNS for your domain is set up for the correct IP (this may take time to propogate)
- Ensure your port 80 is open on the firewall
- Ensure your file permissions are setup correctly on the server files - ownership should be www-data:www-data and directory permissions should be 750 and file permissions should be 640.

Your virtual host should be up and running! You can repeat this for other websites on the same server, with a different configuration file (using the same naming convention) and different directories under /var/www.

PHP Development Virtual Host

This is an example on how to control PHP error logging in a virtual host site for development and debugging. Assumptions

- The PHP module has been installed.
- Development environment is not for production.

```
<VirtualHost *:80>
    ServerName example.com
    DocumentRoot /var/www/domains/example.com/html
    ErrorLog /var/www/domains/example.com/apache.error.log
    CustomLog /var/www/domains/example.com/apache.access.log common
    php_flag log_errors on
    php_flag display_errors on
```

```
    php_value error_reporting 2147483647
    php_value error_log /var/www/domains/example.com/php.error.log
</VirtualHost>
```

Note: The Virtual Host configuration is for development only because the display_errors is enabled and you do not want that in production.

Virtual Host In WAMP

Assuming that you are working with Windows 7 PC

Step 1: GOTO -> C:\Windows\System32\drivers\etc Where you will find a file named “hosts”, kindly copy it and paste it at the same location. A copy file of hosts will be created there.

Now we need to make some modifications in this file but if you try to edit it with any editor like notepad or notepad++, it will not allow you to save the file.

Now again copy the same file and paste it on your desktop, now you can edit this file easily.

You will find one or many entries like: 127.0.0.1 localhost In that file. Now add another line below that line, for example: 127.0.0.1 myproject1.local By this way you have defined a new sub-domain “myproject1.local” which can work in place of “localhost/myproject1”.

Step 2: Okay, now it’s time to define the root path to access this newly created domain right? GOTO : C:\wamp\bin\apache\Your-Apache-Version\conf\extra Here you will find a file named “httpd-vhosts”. Open it in editor and paste the below lines in it.

```
<VirtualHost *:80>
    ServerAdmin webmaster@dummy.example.com
    DocumentRoot "c:/wamp/www/myproject1/”
    ServerName myproject1.local
    ErrorLog "logs/myproject1.local-error.log"
    CustomLog "logs/myproject1.local.log" common
</VirtualHost>
```

Now you are almost there to access the project which resides at “c:/wamp/www/myproject1/”

Step3: GOTO : C:\wamp\bin\apache\your-Apache-Version\conf

Find a file named **“httpd.conf”**, copy it and paste it at the same location for safety. Open file in editor and find a word **“# Virtual hosts”**, below you will find a line **“Include conf/extra/httpd-vhosts.conf”** If it is commented then make it uncommented and restart your wamp-server’s services.

Go to your web-browser and write myproject1.local, you can see the project running now.

Now you might face a problem that your localhost will not work by using localhost as a URL. No Worries…paste this code in **“httpd-vhosts”** file.

```
<VirtualHost *:80>
    ServerAdmin webmaster@dummy.example.com
```

```
    DocumentRoot "c:/wamp/www"
    ServerName localhost
    ErrorLog "logs/localhost-error.log"
    CustomLog "logs/localhost.log" common
</VirtualHost>
```

Restart all the services of WAMP, the work is done.

Thanks & cheers **Chintan Gor**

1) IP based vhosts 2) Multiple vhosts with the same Port 3) Defining vhosts using Macro (Apache2.4)

1) IP based vhosts

```
 <VirtualHost 192.168.13.37>
     ServerName example.com
     DocumentRoot /var/www/domains/example.com/html
     ErrorLog /var/log/example.com/error.log
     CustomLog /var/log/example.com/access.log common
 </VirtualHost>

 <VirtualHost 192.168.47.11>
     ServerName otherurl.com
     DocumentRoot /srv/www/htdocs/otherurl.com/html
     ErrorLog /var/log/otherurl.com/error.log
     CustomLog /var/log/otherurl.com/access.log common
 </VirtualHost>
```

Just change the port to your given IP(s). The port is irrelevant for the decision which vhost is chosen.

2) Multiple vhosts with the same Port

Since NameVirtualHost isn't needed anymore you can just write multiple vhosts with the same port.

```
<VirtualHost *:80>
     DocumentRoot /srv/www/htdocs/otherurl.com/html
     ErrorLog /var/log/otherurl.com/error.log
     CustomLog /var/log/otherurl.com/access.log common
</VirtualHost>

<VirtualHost *:80>
     ServerName example.com
     ServerAlias ex1.com ex2.com
     DocumentRoot /var/www/domains/example.com/html
     ErrorLog /var/log/example.com/error.log
     CustomLog /var/log/example.com/access.log common
 </VirtualHost>
```

Here the opposite applies: the IP is irrelevant, but if the request is received on port 80 the name you entered is evaluated. Did you call ex1.com the 2nd vhost gets picked. And if you called any other url (like otherurl.com, but also example3.com) the first one will be picked. You can use this

vhost as a 'fallback' if you will.

3) Defining vhosts using Macro (Apache2.4)

```
<Macro VHost $port $host>
    <VirtualHost *:$port>
        Servername $host
        DocumentRoot /srv/www/htdocs/$host
        ErrorLog /var/log/$host/error.log
    </VirtualHost>
</Macro>

Use VHost 80 example.com
Use VHost 443 secure_example.com
```

Creates two vhosts, one for port 80, one for 443, and sets the used variables accordingly.

Force HTTPS using virtual host

Use **Redirect** to force users to connect to the secure URL.

```
<VirtualHost *:80>
    ServerName example.com
    SSLProxyEngine on
    Redirect permanent / https://secure_example.com/
</VirtualHost>
```

The rest of the configuration can be put in the ssl virtual host (port 443) since everything is redirected.

```
<VirtualHost _default_:443>
    ServerName secure_example.com
    ServerAdmin webmaster@example.com
    DocumentRoot /var/www/domains/secure_example.com/html
    ErrorLog /var/log/secure_example.com/error.log
    CustomLog /var/log/secure_example.com/access.log common
    SSLEngine On
    ...
</VirtualHost>
```

Read How to create virtual host in Apache online: https://riptutorial.com/apache/topic/4856/how-to-create-virtual-host-in-apache

Credits

S. No	Chapters	Contributors
1	Getting started with apache	Community, fab, Flamewires, hjpotter92, James, Katie, Kuhan, Nicholas Qiao, Rich Bowen
2	.htaccess files in Apache	Chintan Gor, Deltik, ezra-s, Luke Bearl, Rich Bowen, SimpleAnecdote
3	Apache Flume	Vinod Kumar
4	How to create virtual host in Apache	Chintan Gor, Clutch, fab, Harikrishnan, Hello Fishy, Katie, Olaf Dietsche