

# APACHE INSTALLATION & BASIC CONFIGURATION



# OVERVIEW

- The Apache Webserver ( commonly just called Apache ) is an extremely popular open source web server. Most commonly run on Unix platforms, but also available for Windows systems.
- First released way back in 1995, it has since grown to an exceptional level of popularity, due to it's stability, flexibility and raw power. Apache currently runs on over 59% of all websites. Next is Microsoft IIS, at 21%.
- Through the application of hundreds of modules, Apache can be extended to perform virtually any task desired on the world wide web today.



# VERSIONS

- Apache keeps three separate versions:
  - **1.3** - Original major release. Extremely stable, highly deployed. Just EOL-ed February 2010.
  - **2.0** - First major update to Apache in years, including support for threading, IPv6 support, and more. Still maintained.
  - **2.2** - Current version of Apache, incorporating enhanced auth features, enhanced caching and more.
  - **2.4** - Latest and greatest current version of Apache.



# GETTING APACHE

- Apache is generally obtained via source code directly from the Apache Foundation:
  - `http://httpd.apache.org`
- Or in a pre-compiled form from a distributor.



# COMPILING

- Compiling from source is an attractive option because:
  - Users can tweak advanced settings and features
  - Users can wring extra performance from the server
- Compiling from source cons:
  - Compiling. :)
  - Understanding all of the options and implications to achieve an effective web server for the user's application.



# PRE-COMPILED

- Pre-compiled software is an attractive option because:
  - Generally available in package form, therefore bringing all of the benefits of packaged software ( dependencies, tracking, upgrades, management )
  - Do not need up front knowledge on Apache compile time options and settings
- The only real drawbacks to pre-compiled software are:
  - Often not compiled to target user's specific hardware
  - No control over compile time configuration options and features



# DOCUMENTATION

- The Apache documentation is *fantastic*. There is comprehensive information available on almost every aspect of the Apache webserver.
- **Knowing the location of and being able to read the Apache documentation is absolutely key to becoming an Apache Webserver Ninja.**
- Many of the labs will require you to dig through the documentation to find the appropriate modules, configuration directives and guides, so make sure to invest time in getting to know the format and layout.



# COMPILING

- Compiling is a multi-step, often complicated process.
- Fortunately, Apache is a very well maintained project, and the process of building the webserver has been finely tuned.
- The short form:
  - `./configure; make; make install`
- Through lab and demonstration, we will explore all of these steps...



# LAB

1. Download the latest 2.2.x version of Apache and build it. Enable the server information module ( `mod_info` ). Do not install this server - you can complete steps 2 and 3 by running the `httpd` command in your build folder as “`./httpd`”
2. Verify the built-in module by checking for it with the:  
  
`./httpd -l`  
  
command.
3. Explore some of the other information options to the `httpd` executable: `-v`, `-V`, `-l`, `-L`



# PACKAGES

- Installing a package is very simple on most distributions of linux:
  - `yum install httpd`
  - `apt-get install httpd`
- The lab machines already have an Apache package installed. For information on this package, run:
  - `rpm -qi httpd`



# CONTROLLING APACHE

- Controlling Apache is handled differently based on personal choice and operating system distribution:
  - **apachectl**: included with Apache and often used on non-linux unix systems.
  - **init script**: Generally the preferred way of controlling the server, as it will often tie in with any special system control tools and features. This is what will be used in the lab environment.



# LAB

1. Use the `service` command to start your Apache server:

```
service httpd start
```

2. Using Firefox, browse over to your webserver by typing “localhost” into the address bar. Can you see the welcome page?
3. Try restarting the server



# APACHE CONFIGURATION

- Apache configuration is accomplished via one or more configuration files. The primary configuration file is traditionally named `httpd.conf`.
- In the lab, this file is located under
  - `/etc/httpd/conf/httpd.conf`
- Note that on a default install from source, this file would be:
  - `/usr/local/apache2/conf/httpd.conf`



# APACHE CONFIGURATION

- Many of the packaged Apache installations make use of Apache's ability to **include** configuration files from the main configuration file.
- For example, on the lab machines, any file ending in a “.conf” in the following folder will be included as part of Apache's configuration:
  - `/etc/httpd/conf.d`



# LAB

1. Explore the various Apache configuration files on your machine. Can you make some educated guesses as to what some of the configuration directives control?
2. Try to find a configuration directive which appears to set the host name that the Apache server will use to identify itself in messages.
3. Locate a couple of directives which seem interesting or confusing and look up their meaning in the online Apache documentation.
4. Locate the Include directive and make a guess as to its operation. :)



# APACHE CONFIGURATION

- There are several very important basics to understand up front before trying to tackle Apache configuration:


- Context

- Override

- Module

- and of course Syntax

- Feel free to open a documentation window to reference as we cover each topic.



<b>Description:</b>	Activates a CGI script for a particular handler or content-type
<b>Syntax:</b>	Action <i>action-type</i> <i>cgi-script</i> [virtual]
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_actions
<b>Compatibility:</b>	The <code>virtual</code> modifier and handler passing were introduced in Apache 2.1

← “Ripped” straight from Apache docs!



# CONTEXT

- Every configuration directive that can be used in an Apache config file encompasses something known as a context.
- The context defines exactly which areas of the configuration the directive can be used. Areas include:
  - server config
  - virtual host
  - .htaccess
  - ...

<u>Description:</u>	Activates a CGI script for a particular handler or content-type
<u>Syntax:</u>	Action <i>action-type</i> <i>cgi-script</i> [virtual]
<u>Context:</u>	server config, virtual host, directory, .htaccess
<u>Override:</u>	FileInfo
<u>Status:</u>	Base
<u>Module:</u>	mod_actions
<u>Compatibility:</u>	The virtual modifier and handler passing were introduced in Apache 2.1



# CONTEXT

- Understanding the context is important for several reasons.
- Directives are only allowed in their defined contexts. For example, the context for the Script directive is “server config, virtual host, directory”. This means, for example, that the Script directive can not be used in a .htaccess file.
- Additionally, context often defines the *scope* of the directive’s effect as well. When a directive can be used in a “virtual host” context, that implies that it’s effect will only be observed for the given vhost.



# OVERRIDE

- The override attribute defines the AllowOverride configuration option that must be in place to allow the directive to be used in a .htaccess file.
- Using a .htaccess file implies *overriding* certain server configuration directives from within the directory containing the .htaccess file.

For example, the Action directive can only be used in a .htaccess file if there is an AllowOverride option containing “FileInfo”

<b>Description:</b>	Activates a CGI script for a particular handler or content-type
<b>Syntax:</b>	Action action-type cgi-script [virtual]
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_actions
<b>Compatibility:</b>	The virtual modifier and handler passing were introduced in Apache 2.1



# MODULE

- The module attribute tells the reader which Apache module must be available in order to use the configuration directive.
- To use the Action directive, the mod\_actions module must be compiled in or loaded at run time.

<b>Description:</b>	Activates a CGI script for a particular handler or content-type
<b>Syntax:</b>	Action <i>action-type</i> <i>cgi-script</i> [virtual]
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_actions
<b>Compatibility:</b>	The virtual modifier and handler passing were introduced in Apache 2.1



# SYNTAX

- Finally, the syntax attribute provides a very brief synopsis of how the configuration directive is used.
- For additional details, consult the rest of the supplied documentation.

<u>Description:</u>	Activates a CGI script for a particular handler or content-type
<u>Syntax:</u>	Action <i>action-type</i> <i>cgi-script</i> [virtual]
<u>Context:</u>	server config, virtual host, directory, .htaccess
<u>Override:</u>	FileInfo
<u>Status:</u>	Base
<u>Module:</u>	mod_actions
<u>Compatibility:</u>	The virtual modifier and handler passing were introduced in Apache 2.1



# DEFAULT CONFIGURATION

- Using the basic knowledge from before, combined with the extensive documentation provided with Apache, let's walk through the default configuration file and try to make some sense of the directives...



# LAB

1. Create a new directory under `/var/www/html` called “`site1`”. Update your Apache configs to use this new folder for serving content. Place a test index file in there to verify your “website” is working correctly.
2. Read up on the `mod_info` module and try to update your Apache config to allow viewing of the server configuration information remotely, via the `mod_info` module.



```
slideshow.end();
```