CONFIGURING APACHE 2 ON DEBIAN, UBUNTU

The Debian distribution of Linux includes the Apache web server, both the venerable version 1 and the more modern version 2. The Debian maintainers have a peculiar way of arranging the configuration files for Apache 2.0 which is not documented in the standard Apache documentation. This introduction should help you get acclimated to the Debian way of configuring Apache 2.0.

Active Configuration Files

```
# /etc/apache2/apache2.conf - pulls in additional

# configurations in this order:

Include /etc/apache2/mods-enabled/*.load

Include /etc/apache2/mods-enabled/*.conf

Include /etc/apache2/httpd.conf

Include /etc/apache2/ports.conf

Include /etc/apache2/conf.d/[^.#]*

Include /etc/apache2/sites-enabled/[^.#]*
```

Debian stores its Apache 2.0 configuration files in the directory `/etc/apache2`. Normally the main Apache configuration file is called `httpd.conf`. Although that file exists on Debian, it is only there for compatibility with other software that expects it to exist. The real configuration starts with the file `apache2.conf`. You can still add configuration statements to `httpd.conf`, as `apache2.conf` includes it, but you would do well to ignore that fact. Your hand-edited changes should go elsewhere (see below).

Debian adds another configuration file, `ports.conf`, which contains the `Listen` directives telling the Apache server what IP address and port to listen to

(Apache 2 no longer uses the`Port` directive.) I'm not sure why the maintainers decided to break this out into a separate file.

The best place to put your own custom configurations is in the `conf.d` directory. Files in this directory are included as part of the "global" server configuration and will apply to all virtual hosts (see below). For example, if all the sites on your server use the **Yahoo User Interface libraries**, you might store one copy of the libraries in a central location that can be shared across all sites, and create a file `/etc/apache2/conf.d/yui.conf` with an `Alias`directive to map it to the same URL space for all sites.

Apache Modules

```
# Files related to Apache modules

/etc/apache2/mods-enabled/*.load

/etc/apache2/mods-enabled/*.conf

/etc/apache2/mods-available/*.load

/etc/apache2/mods-available/*.conf

/usr/sbin/a2enmod

/usr/sbin/a2dismod
```

One of the great advantages of the Apache web server is its modular architecture. You can add or remove functionality as dictated by your requirements. In the default Apache build, you would find a section near the top of your `httpd.conf` file with instructions to load each module. Later in the file, you would find configuration sections specific to each module, possibly wrapped in a `<IfModule>` directive. This arrangement can be tricky from the perspective of a system administrator who may need to install or uninstall various Apache modules. Identifying the configuration changes that are required by a module or that require a specific module can be difficult to do by hand and even harder to automate with a script.

To make things easier on the server administrator, Debian takes advantage of the fact that Apache configuration files may contain an `Include` directive which pulls in additional configuration files. Debian creates two non-standard directories: `/etc/apache2/mods-enabled`and `/etc/apache2/mods-available`. Whenever you install an Apache module from a Debian package, the module will drop one or two files into the `mods-available` directory. The mandatory `${module}.load` contains the Apache `Load` directive to load the module into your web server. The optional `${module}.conf` file contains additional configuration directives necessary for the operation of the module.

Installing a module from a Debian package makes it available to your server, but does not (necessarily) automatically activate the module in your server. To activate the module, use the `a2enmod` command:

```
a2enmod ${module}

/etc/init.d/apache2 force-reload
```

The `a2enmod` command will create symbolic links in the `mods-enabled` directory pointing to your`${module}.load` and, if it exists, `${module}.conf`. To force a running Apache to re-read its configuration files and thus load the module, you must then send it the `force-reload` signal.

Likewise, to disable a module:

```
a2dismod ${module}

/etc/init.d/apache2 force-reload
```

Issue either command without a module argument and it will print a list of appropriate module names.

You can, of course, manually manage the symbolic links in the `mods-enabled` directory, but it is safer and easier to use the provided scripts.

Virtual Hosts

```
# Files related to Apache virtual hosts

/etc/apache2/sites-enabled/[^.#]*

/etc/apache2/sites-available/*

/usr/sbin/a2ensite

/usr/sbin/a2dissite
```

A *Virtual Host* is just a web site served by your Apache server. Virtual hosts are managed just like modules. Each site gets its own configuration file that contains all the Apache directives that pertain only to that site. These files (or symbolic links to them) should be placed in the `sites-available` directory. There are no strict naming requirements for these files (files beginning with `.` or `#` will be ignored), but for convenience you should name each site configuration file to match the domain name it is serving. There is no need to add a “conf” extension. For example, the vhost file used for this web site is named `www.control-escape.com`.

To activate any of these sites, use the `a2ensite` command, which operates identically to the`a2enmod` command mentioned above. There is a respective `a2dissite` command for disabling a site.

Even if you only run one web site on your server, Apache is still configured to have one virtual host, the *Default Virtual Host*. The default virtual host is treated specially by the`a2ensite` script. If you look in your `sites-enabled` directory you will find that the link has been named `000-default`. The number is prepended to the name by the `a2ensite` script to ensure that the default virtual host is the first one included by Apache (which sorts the files alphabetically). If you want other sites to be loaded in a particular order other than alphabetical, you can rename the links here, but you should always ensure that the default virtual host is the first one loaded.

Source : http://www.control-escape.com/web/configuring-apache2-debian.html