Hosting multiple websites with Apache2

Posted by Steve on Thu 6 Jul 2006 at 22:03

One of the most common Apache2 questions I've seen on Debian mailing lists is from users who wonder how to host multiple websites with a single server. This is very straightforward, especially with the additional tools the Debian package provides.

We've previously discussed some of the tools which are included in the Apache2 package, but what we didn't do was show they're used from start to finish.

There are many different ways you can configure Apache to host multiple sites, ranging from the simple to the complex. Here we're only going to cover the basics with the use of the NameVirtualHost directive. The advantage of this approach is that you don't need to hard-wire any IP addresses, and it will just work^{tm}. The only thing you need is for your domain names to resolve to the IP address of your webserver.

For example if you have an Apache server running upon the IP address `192.168.1.1` and you wish to host the three sites `example.com`, `example.net`, and `example.org` you'll need to make sure that these names resolve to the IP address of your server.

(This might mean that you need `example.com` *and* `www.example.com` to resolve to the same address. However that is a choice you'll need to make for yourself).

Since we'll be hosting multiple websites on the same host it makes a lot of sense to be very clear on the location of each sites files upon the filesystem. The way I suggest you manage this is to create a completely seperate document root, cgi-bin directory, and logfile directory for each host. You can place these beneath the standard Debian prefix of `/var/www` or you may use a completely different root - I use `/home/www`.

If you've not already done create the directories to contain your content, etc, as follows:

```
root@irony:~# mkdir /home/www

root@irony:~# mkdir /home/www/www.example.com
root@irony:~# mkdir /home/www/www.example.com/htdocs
root@irony:~# mkdir /home/www/www.example.com/cgi-bin
root@irony:~# mkdir /home/www/www.example.com/logs

root@irony:~# mkdir /home/www/www.example.net
root@irony:~# mkdir /home/www/www.example.net/htdocs
root@irony:~# mkdir /home/www/www.example.net/logs
root@irony:~# mkdir /home/www/www.example.net/cgi-bin

root@irony:~# mkdir /home/www/www.example.org
root@irony:~# mkdir /home/www/www.example.org/htdocs
root@irony:~# mkdir /home/www/www.example.org/logs
root@irony:~# mkdir /home/www/www.example.org/cgi-bin
```

Here we've setup three different directory trees, one for each site. If you wanted to have identical content it might make sense to only create one, and then use symbolic links instead.

The next thing to do is to enable virtual hosts in your Apache configuration. The simplest way to do this is to create a file called `/etc/apache2/conf.d/virtual.conf` and include the following content in it:

```
#
#  We're running multiple virtual hosts.
#
NameVirtualHost *
```

(When Apache starts up it reads the contents of all files included in `/etc/apache2/conf.d`, and files you create here won't get trashed on package upgrades.)

Once we've done this we can create the individual host configuration files. The Apache2 setup you'll find on Debian GNU/Linux includes two directories for locating your site configuration files:

`/etc/apache2/sites-available`

> This contains configuration files for sites which are available but not necessarily enabled.

`/etc/apache2/sites-enabled`

> This directory contains site files which are enabled.

As with the `conf.d` directory each configuration file in the `sites-enabled` directory is loaded when the server starts - whilst the files in `sites-available` are completely ignored.

You are expected to create your host configuration files in `/etc/apache2/sites-available`, then create a symbolic link to those files in the `sites-enabled` directory - this will cause them to be actually loaded/read.

Rather than actually messing around with symbolic links the Debian package includes two utility commands `a2ensite` and `a2dissite` which will do the necessary work for you as we will demonstrate shortly.

Lets start with a real example. Create `/etc/apache2/sites-available/www.example.com` with the following contents:

```
#
#  Example.com (/etc/apache2/sites-available/www.example.com)
#
<VirtualHost *>
        ServerAdmin webmaster@example.com
        ServerName  www.example.com
        ServerAlias example.com

        # Indexes + Directory Root.
        DirectoryIndex index.html
        DocumentRoot /home/www/www.example.com/htdocs/

        # CGI Directory
        ScriptAlias /cgi-bin/ /home/www/www.example.com/cgi-bin/
        <Location /cgi-bin>
                Options +ExecCGI
        </Location>


        # Logfiles
        ErrorLog  /home/www/www.example.com/logs/error.log
        CustomLog /home/www/www.example.com/logs/access.log combined
</VirtualHost>
```

Next create the file `www.example.net`:

```
#
#  Example.net (/etc/apache2/sites-available/www.example.net)
#
<VirtualHost *>
        ServerAdmin webmaster@example.net
        ServerName  www.example.net
        ServerAlias example.net

        # Indexes + Directory Root.
        DirectoryIndex index.html
        DocumentRoot /home/www/www.example.net/htdocs/

        # CGI Directory
        ScriptAlias /cgi-bin/ /home/www/www.example.net/cgi-bin/
        <Location /cgi-bin>
                Options +ExecCGI
        </Location>


        # Logfiles
        ErrorLog  /home/www/www.example.net/logs/error.log
        CustomLog /home/www/www.example.net/logs/access.log combined
</VirtualHost>
```

Finally create the file `www.example.org`:

```
#
#  Example.org (/etc/apache2/sites-available/www.example.org)
#
<VirtualHost *>
        ServerAdmin webmaster@example.org
        ServerName  www.example.org
        ServerAlias example.org

        # Indexes + Directory Root.
        DirectoryIndex index.html
        DocumentRoot /home/www/www.example.org/htdocs/

        # CGI Directory
        ScriptAlias /cgi-bin/ /home/www/www.example.org/cgi-bin/
        <Location /cgi-bin>
                Options +ExecCGI
        </Location>


        # Logfiles
        ErrorLog  /home/www/www.example.org/logs/error.log
        CustomLog /home/www/www.example.org/logs/access.log combined
</VirtualHost>
```

Now we've got:

- Three directories which can be used to contain our content.
- Three directories which can be used to contain our logfiles.
- Three directories which can be used to contain our dynamic CGI scripts.
- Three configuration files which are being ignored by Apache.

To enable the sites simply run:

```
root@irony:~# a2ensite www.example.com
Site www.example.com installed; run /etc/init.d/apache2 reload to enable.

root@irony:~# a2ensite www.example.net
Site www.example.net installed; run /etc/init.d/apache2 reload to enable.

root@irony:~# a2ensite www.example.org
Site www.example.org installed; run /etc/init.d/apache2 reload to enable.
```

This will now create the symbolic links so that `/etc/apache2/sites-enabled/www.example.org`, etc, now exist and will be read.

Once we've finished our setup we can restart, or reload, the webserver as the output above instructed us to do with:

```
root@irony:~# /etc/init.d/apache2 reload
```

```
Reloading web server config...done.
root@irony:~#
```
