TroubleshootingVhosts

Some Virtual hosts troubleshooting information/tips based on the common questions about them in #httpd

TroubleShooting Virtual Hosts

The page will concentrate on Name Based virtual hosts mainly because it is what most people use and it is the one many people seem to have problems with. Version specifc information is marked as such.

The Basics

Virtual Hosts are much simpler than most people seem to think but in some cases are made more complicated by some unusual 'default' virtual host setups installed by some of the major distributions.

First things first, **read the documentation**!

- http://httpd.apache.org/docs/current/vhosts/
- http://httpd.apache.org/docs/current/vhosts/examples.html
- ExampleVhosts

Secondly, and this cannot be stressed enough, use `apachectl -S` or `httpd -S` whenever any changes are made to the configuration. It outputs various lines of information that are vital to the troubleshooting of virtual host configurations. See below for further information.

Thirdly, although there are a couple of exceptions, if an access log is defined then httpd will *always* log something to it when it serves a request and an entry will always be logged in the error log on a 4xx return code. An error log entry will also normally be written on a 5xx return code, however when using 3rd party modules, CGIs or launguages such as php it is possible for these to be hijacked and nothing will be in the log. If you cannot find the access/error log entry then you are either looking in the wrong log or the request didn't actually reach the web server.

Virtual Host Requirements

- (httpd prior to v2.3.11) You *must* have a NamedVirtualHost directive for each IP+port combination in use, it should be IP:port or *:port and should come before any of the actual virtual hosts.

```
NameVirtualhost *:80
```

- Each %3CVirtualHost%3E directive should have an IP:port inside it, in httpd prior to v2.3.11 this should match the NamedVirtualHost directive. Unless you know exactly what you are doing, do not mix IP:port and *:port as the results may be unpredictable.

```
<VirtualHost *:80>
```

- Each virtual host *must* have its own unique ServerName. It is this that must match the host component of the URL that the user types into their browser. If another virtual host is defined with the same server name (or server alias) then whichever virtual host comes first in the configuration will be the only one that works with that name.

```
ServerName foo.com
```

- Each virtual host should also have its own distinct DocumentRoot. Although it is possible, and in a few cases needed, to share the document root, if this is required it is normally better to use a single virtual host with server aliases.

```
DocumentRoot /var/www/foo
```

The First Virtual Host

With named base virtual hosts, the first one apache finds in the configuration files is special. This is the one that requests will be passed to if apache has no way to determine which specific virtual host to use. It is the *default* virtual host.

In general, if the user types `http://my.domain.com/my/url/path` into their browser, then it is the string `my.domain.com` that is matched against the host component of the ServerName, or any ServerAlias directive (Remember Server{{`Name can contain an optional schema and port). Anything that does not match a Server}}`Name or Server`Alias in the configuration will be served by the default virtual host.

Note: It is the contents of the `Host` header that is actually used by httpd here rather than the host component of the URL, though these would normally be the same.

Virtual Host Example

Here is a very simple two virtual host example that we will use to highlight the output of `apachectl -S`.

```
# NameVirtualhost is needed in httpd prior to v2.3.11
NameVirtualHost 192.168.0.1:80

# Any request to foo.com, or indeed anything other than bar.com that resolves
# to 192.168.0.1 will be served by this first virtual host
<VirtualHost 192.168.0.1:80>
  ServerName foo.com
  ServerAlias www.foo.com
  DocumentRoot /var/www/foo
</Virtualhost>

# Only URLs that start http:///bar.com/ or http://www.bar.com will be served
# by this virtual host
<VirtualHost 192.168.0.1:80>
  ServerName bar.com
  ServerAlias www.bar.com
  DocumentRoot /var/www/bar
</Virtualhost>
```

Problem Solving Tips

The ErrorLog is always the first place to look when any problems arise, but there are things that can be done to make troubleshooting virtual hosts easier.

- Give each virtual host its own access and error log. This is a simple way to separate out the requests to each virtual host and in particular verify that a request is actually being served by the virtual host you think it should be. An alternative to this is to add `%v` to the relevant LogFormat.
- In conjunction with the access/error logs, make a request that is guaranteed to generate a 404 Not Found error. You can then search the access and error logs for the entries relating to this request.
- Use a command line tool to access a specific virtual host. This is particularly useful when there are redirects in place and you wish to make sure they are working correctly. Commonly used tools for the troubleshooting of a web server from the command line are curl and wget. If the perl LWP module is installed then lwp-request may also be used (often usable as `GET`). It is worth learning these tools and their options, but here are some examples by way of an introduction.

  ```
  # Get the headers from foo.com. Output in the file headers.txt
  curl -s -D headers.txt -o /dev/null http://foo.com/
  GET -uUsSed http://foo.com/ > headers.txt

  # Using just the host header instead of the fqdn. Output in the file headers.txt
  curl -H "Host: foo.com" -s -D headers.txt -o /dev/null http://192.168.0.1/
  wget --header="Host: foo.com" -nv --save-headers=on -O headers.txt http://192.168.0.1/
  GET -H "Host: foo.com" -uUsSed http://192.168.0.1/ > headers.txt

  # Get the headers and content of the page to the terminal
  curl -s -i http://foo.com/
  wget -nv --save-headers=on -O - http://foo.com/
  GET -uUsSe http://foo.com/
  ```

- *Every* time a modification is made to httpd's configuration you should be running `apachectl -S` to test it before the web server is restarted. Below are two examples of the output from this command for httpd v2.2.x and 2.4.x with line numbers added for clarity. A line by line explanation of each follows. The output was generated using a basic httpd install with the example virtual host configuration above.

```
# Example output from httpd 2.2
1: VirtualHost configuration:
2: 192.168.0.1:8080       is a NameVirtualHost
3:        default server foo.com (/etc/httpd/conf.d/vhosts.conf:11)
4:        port 80 namevhost foo.com (/etc/httpd/conf.d/vhosts.conf:11)
5:        port 80 namevhost bar.com (/etc/httpd/conf.d/vhosts.conf:18)
```

1. The first line of output, unless there are warnings/errors.
2. If there is no Name{{`Virtual}}`Host directive this line will not be present.
3. The 'default' virtual host is the one with server name 'foo.com'. It was defined in the file /etc/httpd/conf.d/vhosts.conf and started on line 11.
4. A second entry refering to the default virtual host. It runs on port 80 and it's server name is foo.com. The file and start line are repeated.
5. We have another name based virtual host running on port 80 with server name bar.com. It was defined in the file /etc/httpd/conf.d/vhosts.conf and started on line 18.

```
# Example output from httpd 2.4
1: VirtualHost configuration:
2: 192.168.0.1:80         is a NameVirtualHost
3:          default server foo.com (/etc/httpd/conf.d/vhosts.conf:2)
4:          port 80 namevhost foo.com (/etc/httpd/conf.d/vhosts.conf:2)
5:                  alias www.foo.com
6:          port 80 namevhost bar.com (/etc/httpd/conf.d/vhosts.conf:9)
7:                  alias www.bar.com
8: <After this are several lines of output relating to the global configuration>
```

6. The first line of output, unless there are warnings/errors.
7. httpd 2.4 automatically works out the need for a name based virtual host (a warning is output if the directive is present).
8. The 'default' virtual host is the one with server name 'foo.com'. It was defined in the file /etc/httpd/conf.d/vhosts.conf and started on line 2.
9. A second entry refering to the default virtual host. It runs on port 80 and it's server name is foo.com. The file and start line are repeated.
10. The name virtual host foo.com has an alias www.foo.com (this output is new in httpd 2.4).
11. We have another name based virtual host running on port 80 with server name bar.com. It was defined in the file /etc/httpd/conf.d/vhosts.conf and started on line 9.
12. The name virtual host bar.com has an alias www.bar.com.
13. After the virtual hosts information, the output contains lines useful for debugging the global configuration.

In the ouput, if there are any duplicate server names or aliases (other than the default one and the namevhost immediately after it) then there are virtual hosts with the same name (or Alias) and only the first one in the list will work.

With httpd prior to v2.3.11, if you do not have a Name{{`Virtual}}{{Host directive you will not see the line ending in }}is a NameVirtualHost` and depending on the exact configuration you may see something similar to one of the following warnings:

```
 # If the virtual hosts are defined with IP:port
[Fri Jan 11 22:27:30 2013] [warn] VirtualHost 192.168.0.1:80 overlaps with VirtualHost 192.168.0.1:80, the
first has precedence, perhaps you need a NameVirtualHost directive

 # If the virtual hosts are defined with *:port
[Fri Jan 11 22:28:04 2013] [warn] _default_ VirtualHost overlap on port 80, the first has precedence
```