Creating Virtual Host In Apache2 Under Linux

The **VirtualHost** directive in the **httpd.conf** configuration file is used to set the values of **ServerName**, **DocumentRoot**, **ErrorLog** and **TransferLog** or **CustomLog** configuration directives to different values for each virtual host.

Multiple websites can be served from one computer, even if they have different hostnames. Each host name that is served from the single computer that hosts them all is referred to as a **virtual host**. There are **two** ways provided by Apache for setting up virtual hosts on a single computer, **IP based** and **Name based**.

IP based virtual hosts use the IP address of the connection to determine the correct virtual host. Hence, a **unique** IP address is required for each host.

With **name based** virtual hosting, Apache Web server relies on the client to deliver the hostname as part of the HTTP leaders sent to Apache. Using this technique, many different virtual hosts can share the **same IP address**.

Warning

Older browsers do not support delivering a hostname with their HTTP headers. This is not part of their **HTTP 1.1** header encoding. Hence these browsers will only work with IP based virtual hosts.

Apache can be configured to support multiple virtual hosts either by running a **separate httpd** daemon for each hostname, or by running a **single httpd** daemon, which supports all the virtual hosts.

If separate httpd daemons must be run for each host, separate installations of Apache for each virtual host have to be created. For each installation, use the **Listen directive** in the **httpd.conf** configuration file to select which IP address **or** virtual host that daemon services

For example, Listen 192.168.0.1:80.

A single http daemon can also be used to service to the main server and **all** its virtual hosts.

IP Based Virtual Hosts

If a moderately large Intranet environment is required, different websites for various departments or projects may have to be setup. Each of these websites lie on a **single Linux server**, configured to use multiple interfaces with independent host names and IP addresses (i.e. Virtual Domains). The first step is to set up multiple interfaces on the Ethernet card that connects the Linux Server to the network.

The existing interface will be **eth0**. Use the **ifconfig** command to add a new interface to it:

```
/sbin/ifconfig eth0:1 192.168.0.200
```

Now add a route to the new interface:

```
/sbin/route add -host 192.168.0.200 dev eth0:1
```

Make sure the 192.168.0.200 (or whatever is chosen) is a reserved IP address and cannot by dynamically assigned by any DHCP server that may also be servicing the network. Ping the new IP address as any other address on the network.

Next **edit** the **nameserver files** to give this interface a suitable name. If the Linux box is also running a DNS service configure it from there.

Change to the directory **/var/named**. Using any ASCII editor, edit the file named.xxx.yyy.forward (where xxx.yyy is your domain name). Scroll down the list until you find the entry **ws200** corresponding to the IP 192.168.0.200. Replace **ws200** with the name of the interface, (in this case reviews). **Restart** named using the command:

```
<System Prompt> /etc/rc.d/init.d/named restart
```

Alternately, create an entry in the**/etc/hosts** file for the virtual interface. Now any computer on the network should be able to ping the interface by name.

CREATING VIRTUAL HOST IN APACHE2 UNDER LINUX

The final step is to create a file to add the interface and route automatically at boot time. Create a file (For example: rc.virtual) in the **/etc/rc.d** directory and add the two lines to it. The first line of the script should be **#!/bin/sh**. Mark it executable (**chmod +x rc.virtual**), and edit the file **rc.local** in the same directory. At the last line of that file, add the line **/etc/rc.d/rc.virtual**. That will ensure that rc.virtual is run automatically at bootup. Similarly add as many virtual domains as required.

Warning

If a firewall is in use, remember to add appropriate lines to secure the interface. In this case, restricted access to a particular port may be required.

Add the correct domain name to the virtual host tag and specify a document root path, which defines where the HTML files are stored for that host. The correct permissions on this directory are important. Make sure that the root directory is universally readable. For example:

```
<VirtualHost 192.168.0.200>
    ServerAdmin webmaster@sct.com
    DocumentRoot /var/www/sct
</VirtualHost>
```

Name Based Virtual Hosts

Using the new name based virtual hosts is quite easy, and superficially looks like the old method. The notable difference between IP-based and Name-based virtual host configuration is the NameVirtualHost directive, which specifies a single IP address that should be used as a target for name-based virtual hosts.

For example, suppose that both www.sharanam.com and www.ivan.com point to the IP address 192.168.0.3. Then simply add the following to the Apache's httpd.conf:

```
ServerName 192.168.0.3
NameVirtualHost 192.168.0.3
<VirtualHost 192.168.0.3>
    DocumentRoot /var/www/sharanam
    ServerName www.sharanam.com
</VirtualHost>
```

```
<VirtualHost 192.168.0.3>
    DocumentRoot /var/www/ivan
    ServerName www.ivan.com
</VirtualHost>
```

The Following entries should be appended to the **hosts** file available under **/etc** directory**:**

```
192.168.0.3        www.sharanam.com
192.168.0.3        www.ivan.com
```

After making any changes to the **httpd.conf** ensure to restart the apache server to apply the new changes. This can be done by**:**

- Stop the Apache2 HTTPD service**:**

```
<System Prompt> /usr/local/apache2/bin/apachectl stop
```

- Start the Apache2 HTTPD service**:**

```
<System Prompt> /usr/local/apache2/bin/apachectl start
```