DifferentUserIDsUsingReverseProxy

One frequently requested feature is to run different virtual hosts under different userids. Unfortunately, due to the basic nature of unix permission handling, this is impossible. (Although it is possible to run CGI scripts under different userids using suexec or cgiwrap.) You can, however, get the same effect by running multiple instances of Apache httpd and using a reverse proxy to bring them all into the same name space.

This same technique can also be useful if you want certain virtual hosts to be run under a non-threaded server (prefork) while others run with threading (worker or event).

Main host

This instance does not actually serve requests, but rather proxies them to other servers running on other ports.

```
Listen 80
NameVirtualHost *:80
User httpd
Group httpd

ProxyRequests Off

<VirtualHost *:80>
  ServerName host1.example.com
  ProxyPass / http://localhost:81/
  ProxyPassReverse / http://localhost:81/
</VirtualHost>

<VirtualHost *:80>
  ServerName host2.example.com
  ProxyPass / http://localhost:82/
  ProxyPassReverse / http://localhost:82/
</VirtualHost>
```

Back-end hosts

These hosts do the real work. They must each be started independently. If they share the same httpd binary (and hence the same mpm), you can start them as follows:

```
apachectl -f /usr/local/apache2/conf/host1.conf -k start
apachectl -f /usr/local/apache2/conf/host2.conf -k start
```

Host 1

```
Listen 81
ServerName localhost:81
User host1user
Group host1group
DocumentRoot /usr/local/apache2/htdocs1
# etc...
```

Host 2

```
Listen 82
ServerName localhost:82
User host2user
Group host2group
DocumentRoot /usr/local/apache2/htdocs2
# etc...
```

Combining into one config file

If the virtual (back-end) hosts share many common config elements, it may be easier if all the hosts share a config file. This can be accomplished by wrapping the parts above in <IfDefine hostx> sections, and then using

```
apachectl -Dhostx -k start
```

to start each host.

Running unprivileged back-end hosts

If you use unprivileged ports for the back-end hosts (for example, replacing 81 and 82 above with 8001 and 8002) then you may choose to start these hosts directly under the less-privileged userids (host1user and host2user) in place of root. This will allow you to give complete control of these back-end servers to host1user and host2user. These users could then edit httpd.conf and manage log files and server restarts without needing root privileges. You may also choose to run the back-end hosts in a chroot environment, in a FreeBSD jail, or under other restricted permissions (using SELinux, for example).

When starting httpd under a less-privileged userid, you'll need to adjust certain directives such as `PIDFile` and `CustomLog` to point to locations writable by the less-privileged user.

Alternative Proxy Configuration

If back-end hosts might be added and removed frequently, it could be helpful to be able to do this without modifying the configuration of the front-end server. Here is an example config that allows back-end hosts to be added and deleted by editing a simple text database:

```
Listen 80
ProxyRequests Off
ProxyPreserveHost On
RewriteEngine On

# A dbm map might be better if you have more than a few dozen hosts
RewriteMap host txt:/usr/local/etc/apache22/host_to_ip
RewriteMap tolower int:tolower

# Don't put the space between the }} }. I had to put it in
# because of Wiki formatting rules.
RewriteCond ${host:${tolower:%{HTTP_HOST}} } (.+)
RewriteRule (.*) http://%1$1 [P]

<Directory />
  Deny from all
</Directory>
```

To add or delete a user server, simply edit the host_to_ip file, which might look like this:

```
foo.com 127.0.0.1:8000
www.foo.com 127.0.0.1:8000
bar.com 127.0.0.1:8001
www.bar.com 127.0.0.1:8001
```

Because the `ProxyPreserveHost` directive is used in place of `ProxyPassReverse` in this configuration, you need to ensure that the back-end hosts use the following config in order to get the correct hostname and port on server-generated redirects:

```
UseCanonicalName Off
```