

Mass Virtual Hosting using Apache

The article was obtained at the following URL: <http://www.kfwebs.net/articles/article/34>
The article might be distributed further as long as it is provided as it is, with the credits stated.
The Article was written and first published by KF Webs, at <http://www.kfwebs.net>
#####

Here is an article about configuring a Mass Virtual Host scheme using mod_rewrite as its base. This is a very scalable system which offer a grand flexibility as for its implementation.
Added: 2006-03-08 13:46:57 - Modified: 2006-03-10 17:27:35 - Level: Advanced

Table of Contents (TOC)

1. [What is Mass Virtual Hosting](#)
2. [Configuring Apache](#)
3. [Adding a new domain](#)
 - ◆ [Default content](#)
4. [Testing the configuration](#)
5. [Logfiles](#)
6. [Common Gateway Interface \(CGI\)](#)
7. [Scalability](#)
8. [Rounding off](#)

What is Mass Virtual Hosting

Virtual Hosting is when you host multiple websites on the same server. If you host only two or three sites it is simple to add multiple [VirtualHost directives](#), but if you host more than 10-30 websites it get cumbersome. That is when you switch to a [Mass Virtual Hosting](#) approach. That is, one VirtualHost directive responsible for a number of different hosts. This is the approach your hosting company use when you order Virtual Shared Hosting, and it is the approach you want to use if you have a dedicated server that is supposed to run multiple websites.

Mass Virtual Hosting simplifies administration as well as the memory requirements as well as the time it take to reload or restart Apache.

Configuring Apache

One approach to mass virtual hosting is using [mod_rewrite](#). The basis for the scheme in this article is as follows:

```
<VirtualHost *:80>  
ServerAdmin hostmaster1@kfwebs.com
```

```

ServerName host1.kfwebs.net
Include /etc/apache2/vhost_mass.conf
LogFormat "%{Host}i %h %l %u %t \"%r\" %s %b" vcommon
LogFormat "%{Host}i %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"" vcombined
CustomLog /webs/mass/access_log vcombined
RewriteEngine On
RewriteMap lowercase int:tolower
RewriteCond %{REQUEST_URI} !^/icons/
RewriteCond ${lowercase:%{SERVER_NAME}} ^(?:www\.)?(.\+)$
RewriteRule ^/(.*)$ /webs/mass/%1/$1
<Directory "/">
AllowOverride All
</Directory>
</VirtualHost>

```

Text version

In addition to the actual mod_rewrite it is important to notice the inclusion of vhost_mass.conf. As many have another virtual host block as their default virtual host (the first virtual block that is defined is the default one, which is used when no other server name or server alias is matched) it is necessary have server aliases for all the domains that are to be handled by this block. By including a separate file it is possible to alter it using the scripts that are described below.

Adding a new domain

This script make it very easy to add more hosts to the server. By default it assign ownership to the user kristianf, as that is the username responsible for most domains on the server this article is based upon. If another user is to have access to it, one would issue an chown -R username, and create a symbolic link to the user's home directory.

If you mainly add domains for different users, it is recommended to automate this task as well, by adding an additional argument to the script that you use for the chown and to symlink to the appropriate user's home directory.

```

#!/bin/bash
HOST=$1
AARG=$2
if [ "${HOST}x" == "x" ]; then
echo "usage: addvhost <domainname>";
else
mkdir /webs/mass/$HOST
echo "This is a virtual host of KF Webs" >> /webs/mass/$HOST/index.php
chown -R kristianf:apache /webs/mass/$HOST
chmod -R 770 /webs/mass/$HOST
echo "ServerAlias www.$HOST $HOST" >> /etc/apache2/vhost_mass.conf

```

```
if [ "${AARG}" != "nograce" ]; then
/usr/sbin/apache2ctl graceful
fi
fi
```

[Text version](#)

Default content

To have a default holding page for domains that have yet to be altered by the client the script above require a little modification. To accomplish this one would utilize a skeleton directory that get linked to every new virtual host that get created. This is easily accomplished by altering the addvhost file. Instead of using echo "This is a virtual host of KF Webs" >> /webs/mass/\$HOST/index.php one would utilize **cp -Ral /path/to/skeleton/* /webs/mass/\$HOST/**

Removing a domain

```
#!/bin/bash
export HOST=$1
export AARG=$2
if [ "${HOST}x" == "x" ]; then
echo "usage: delvhost <domainname>";
else
rm -rf /webs/mass/$HOST
perl -lne 'use Env; print $_ unless $_ eq "ServerAlias www.$ENV{HOST} $ENV{HOST}"'
/etc/apache2/vhost_mass.conf > /etc/apache2/vhost_mass.conf.tmp && mv /etc/apache2/vhost_mass.conf.tmp
/etc/apache2/vhost_mass.conf
if [ "${AARG}" != "nograce" ]; then
/usr/sbin/apache2ctl graceful
fi
fi
```

[Text version](#)

Testing the configuration

The domain name passive12.net has a wildcard domain in its zone. This result in that every subdomain get pointed to the server the system is implemented on. Using this to test the implementation using the following command we get:

```
for i in `seq 1 20000`; do addvhost test$i.passive12.net nograce; done; apache2ctl graceful
```

And with that the server suddenly has *20000 more hosts*. The nograce argument to the addvhost command is important in this context, as it tell the script not to reload Apache every time it add a domain. It is more

efficient to rather let it add all the hosts before doing an `apache2ctl graceful` manually for the changes to take effect.

As the default content method mentioned above is used, a test afterwards shows that there are 20,000 hard links to the inode that the default file `index.php` is stored in.

```
alpha test15001.passive12.net # stat index.php | grep Links
Device: 803h/2051d Inode: 2151491 Links: 20000
```

Logfiles

As of now the access log is a combined log format that is logged to a central file in order to avoid an excessive number of open file pointers. In order to get a log file per virtual host you would utilize a script called `split-logfile2` ([which follows Apache](#)). Using this such as `split-logfile2 < /webs/mass/access_log` will give you `domainname.log` for each domain and place it in the current directory. You can implement this in your log-rotate script to be done once a day (or as often that make sense for your configuration)

The mentioned `VirtualHost` block give you access to two different log formats, both splittable by `split-logfile2`. This is the Common Log Format (CLF) and the NCSA extended/combined log format. You can of course alter it to any other log-format you want it to, just remember to keep the host as the first column, so that `split-logfile2` can operate on the file.

Common Gateway Interface (CGI)

Although not recommend, in some situations it is wanted that the customers have access to CGI. This can be achieved by adding the following rewrite rule:

```
RewriteCond ${lowercase:%{SERVER_NAME}} ^(?:www\.)?(.+)$
RewriteRule ^/cgi-bin/(.*)$ /webs/mass/%1/cgi-bin/$1 [T=application/x-httpd-cgi]
```

and adding `RewriteCond %{REQUEST_URI} !^/cgi-bin/` to the existing rewrite rule

Scalability

As you get an even higher number amount of sites you want to modify the provided rewrite-rule a little. Most should however have an adequate setup without following these steps.

Step one would be to store it as `/webs/mass/e/example.com`, hence separating on the first character. As the number grow even more (or is expected to reach a high number) you want it to look like `/webs/mass/e/ex/example.com` and then `/webs/mass/e/ex/exa/example.com`. This is to avoid limitations in the file system and speed up looking for the directories.

Using step 3 as an example, here is what the rewrite rule would look like

