VirtualHost Examples

This document attempts to answer the commonly-asked questions about setting up virtual hosts. These scenarios are those involving multiple web sites running on a single server, via name-based [1] or IP-based [2] virtual hosts. A document should be coming soon about running sites on several servers behind a single proxy server.

Topics

Running several name-based web sites on a single IP address. 1
Name-based hosts on more than one IP address. 2
Serving the same content on different IP addresses (such as an internal and external address). 3
Running different sites on different ports. 3
IP-based virtual hosting 4
Mixed port-based and ip-based virtual hosts 4
Mixed name-based and IP-based vhosts. 5
Using `_default_` vhosts 5
Migrating a name-based vhost to an IP-based vhost 6
Using the `ServerPath` directive 7
URI References 8

Running several name-based web sites on a single IP address.

Your server has a single IP address, and multiple aliases (CNAMES) point to this machine in DNS. You want to run a web server for `www.example1.com` and `www.example2.org` on this machine.

Note

Creating virtual host configurations on your Apache server does not magically cause DNS entries to be created for those host names. You *must* have the names in DNS, resolving to your IP address, or nobody else will be able to see your web site. You can put entries in your `hosts` file for local testing, but that will work only from the machine with those hosts entries.

Server configuration

```
# Ensure that Apache listens on port 80
Listen 80

# Listen for virtual host requests on all IP addresses
NameVirtualHost *

<VirtualHost *>
  DocumentRoot /www/example1
  ServerName www.example1.com

  # Other directives here

</VirtualHost>

<VirtualHost *>
  DocumentRoot /www/example2
  ServerName www.example2.org

  # Other directives here

</VirtualHost>
```

The asterisks match all addresses, so the main server serves no requests. Due to the fact that `www.example1.com` is first in the configuration file, it has the highest priority and can be seen as the *default* or *primary* server. That means that if a request is received that does not match one of the specified `ServerName` directives, it will be served by this first `VirtualHost`.

Note

You can, if you wish, replace `*` with the actual IP address of the system. In that case, the argument to `VirtualHost` *must* match the argument to `NameVirtualHost`:

```
NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>
# etc ...
```

However, it is additionally useful to use `*` on systems where the IP address is not predictable - for example if you have a dynamic IP address with your ISP, and you are using some variety of dynamic DNS solution. Since `*` matches any IP address, this configuration would work without changes whenever your IP address changes.

The above configuration is what you will want to use in almost all name-based virtual hosting situations. The only think that this configuration will not work for, in fact, is when you are serving different content based on differing IP addresses or ports.

Name-based hosts on more than one IP address.

Note

Any of the techniques discussed here can be extended to any number of IP addresses.

The server has two IP addresses. On one (`172.20.30.40`), we will serve the "main" server, `server.domain.com` and on the other (`172.20.30.50`), we will serve two or more virtual hosts.

Server configuration

```
Listen 80

# This is the "main" server running on 172.20.30.40
ServerName server.domain.com
DocumentRoot /www/mainserver

# This is the other address
NameVirtualHost 172.20.30.50

<VirtualHost 172.20.30.50>
  DocumentRoot /www/example1
  ServerName www.example1.com

  # Other directives here ...

</VirtualHost>

<VirtualHost 172.20.30.50>
  DocumentRoot /www/example2
  ServerName www.example2.org

  # Other directives here ...

</VirtualHost>
```

Any request to an address other than `172.20.30.50` will be served from the main server. A request to `172.20.30.50` with an unknown hostname, or no `Host:` header, will be served from `www.example1.com`.

Serving the same content on different IP addresses (such as an internal and external address).

The server machine has two IP addresses (`192.168.1.1` and `172.20.30.40`). The machine is sitting between an internal (intranet) network and an external (internet) network. Outside of the network, the name `server.example.com` resolves to the external address (`172.20.30.40`), but inside the network, that same name resolves to the internal address (`192.168.1.1`).

The server can be made to respond to internal and external requests with the same content, with just one `VirtualHost` section.

Server configuration

```
NameVirtualHost 192.168.1.1
NameVirtualHost 172.20.30.40

<VirtualHost 192.168.1.1 172.20.30.40>
  DocumentRoot /www/server1
  ServerName server.example.com
  ServerAlias server
</VirtualHost>
```

Now requests from both networks will be served from the same `VirtualHost`.

Note:

On the internal network, one can just use the name `server` rather than the fully qualified host name `server.example.com`.

Note also that, in the above example, you can replace the list of IP addresses with `*`, which will cause the server to respond the same on all addresses.

Running different sites on different ports.

You have multiple domains going to the same IP and also want to serve multiple ports. By defining the ports in the "NameVirtualHost" tag, you can allow this to work. If you try using <VirtualHost name:port> without the NameVirtualHost name:port or you try to use the Listen directive, your configuration will not work.

Server configuration

```
Listen 80
Listen 8080

NameVirtualHost 172.20.30.40:80
NameVirtualHost 172.20.30.40:8080

<VirtualHost 172.20.30.40:80>
  ServerName www.example1.com
  DocumentRoot /www/domain-80
</VirtualHost>
```

```
<VirtualHost 172.20.30.40:8080>
  ServerName www.example1.com
  DocumentRoot /www/domain-8080
</VirtualHost>

<VirtualHost 172.20.30.40:80>
  ServerName www.example2.org
  DocumentRoot /www/otherdomain-80
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
  ServerName www.example2.org
  DocumentRoot /www/otherdomain-8080
</VirtualHost>
```

IP-based virtual hosting

The server has two IP addresses (`172.20.30.40` and `172.20.30.50`) which resolve to the names `www.example1.com` and `www.example2.org` respectively.

Server configuration

```
Listen 80

<VirtualHost 172.20.30.40>
  DocumentRoot /www/example1
  ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.50>
  DocumentRoot /www/example2
  ServerName www.example2.org
</VirtualHost>
```

Requests for any address not specified in one of the `<VirtualHost>` directives (such as `localhost`, for example) will go to the main server, if there is one.

Mixed port-based and ip-based virtual hosts

The server machine has two IP addresses (`172.20.30.40` and `172.20.30.50`) which resolve to the names `www.example1.com` and `www.example2.org` respectively. In each case, we want to run hosts on ports 80 and 8080.

Server configuration

```
Listen 172.20.30.40:80
Listen 172.20.30.40:8080
Listen 172.20.30.50:80
Listen 172.20.30.50:8080

<VirtualHost 172.20.30.40:80>
  DocumentRoot /www/example1-80
  ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.40:8080>
  DocumentRoot /www/example1-8080
  ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.50:80>
  DocumentRoot /www/example2-80
  ServerName www.example1.org
</VirtualHost>

<VirtualHost 172.20.30.50:8080>
  DocumentRoot /www/example2-8080
  ServerName www.example2.org
</VirtualHost>
```

Mixed name-based and IP-based vhosts

On some of my addresses, I want to do name-based virtual hosts, and on others, IP-based hosts.

Server configuration

```
Listen 80

NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>
  DocumentRoot /www/example1
  ServerName www.example1.com
</VirtualHost>

<VirtualHost 172.20.30.40>
  DocumentRoot /www/example2
  ServerName www.example2.org
</VirtualHost>

<VirtualHost 172.20.30.40>
  DocumentRoot /www/example3
  ServerName www.example3.net
</VirtualHost>

# IP-based
<VirtualHost 172.20.30.50>
  DocumentRoot /www/example4
  ServerName www.example4.edu
</VirtualHost>

<VirtualHost 172.20.30.60>
  DocumentRoot /www/example5
  ServerName www.example5.gov
</VirtualHost>
```

Using `_default_` vhosts

`_default_` vhosts for all ports

Catching *every* request to any unspecified IP address and port, *i.e.*, an address/port combination that is not used for any other virtual host.

Server configuration

```
<VirtualHost _default_:*>
```

```
  DocumentRoot /www/default
</VirtualHost>
```

Using such a default vhost with a wildcard port effectively prevents any request going to the main server.

A default vhost never serves a request that was sent to an address/port that is used for name-based vhosts. If the request contained an unknown or no `Host:` header it is always served from the primary name-based vhost (the vhost for that address/port appearing first in the configuration file).

You can use `AliasMatch` or `RewriteRule` to rewrite any request to a single information page (or script).

`_default_` vhosts for different ports

Same as setup 1, but the server listens on several ports and we want to use a second `_default_` vhost for port 80.

Server configuration

```
<VirtualHost _default_:80>
  DocumentRoot /www/default80
  # ...
</VirtualHost>

<VirtualHost _default_:*>
  DocumentRoot /www/default
  # ...
</VirtualHost>
```

The default vhost for port 80 (which *must* appear before any default vhost with a wildcard port) catches all requests that were sent to an unspecified IP address. The main server is never used to serve a request.

`_default_` vhosts for one port

We want to have a default vhost for port 80, but no other default vhosts.

Server configuration

```
<VirtualHost _default_:80>
DocumentRoot /www/default
...
</VirtualHost>
```

A request to an unspecified address on port 80 is served from the default vhost any other request to an unspecified address and port is served from the main server.

Migrating a name-based vhost to an IP-based vhost

The name-based vhost with the hostname `www.example2.org` (from our name-based example, setup 2) should get its own IP address. To avoid problems with name servers or proxies who cached the old IP address for the name-based vhost we want to provide both variants during a migration phase.
The solution is easy, because we can simply add the new IP address (`172.20.30.50`) to the `VirtualHost` directive.

Server configuration

```
Listen 80
ServerName www.example1.com
DocumentRoot /www/example1

NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40 172.20.30.50>
  DocumentRoot /www/example2
  ServerName www.example2.org
  # ...
</VirtualHost>

<VirtualHost 172.20.30.40>
  DocumentRoot /www/example3
  ServerName www.example3.net
  ServerAlias *.example3.net
  # ...
</VirtualHost>
```

The vhost can now be accessed through the new address (as an IP-based vhost) and through the old address (as a name-based vhost).

Using the `ServerPath` directive

We have a server with two name-based vhosts. In order to match the correct virtual host a client must send the correct `Host:` header. Old HTTP/1.0 clients do not send such a header and Apache has no clue what vhost the client tried to reach (and serves the request from the primary vhost). To provide as much backward compatibility as possible we create a primary vhost which returns a single page containing links with an URL prefix to the name-based virtual hosts.

Server configuration

```
NameVirtualHost 172.20.30.40

<VirtualHost 172.20.30.40>
  # primary vhost
  DocumentRoot /www/subdomain
  RewriteEngine On
  RewriteRule ^/.* /www/subdomain/index.html
  # ...
</VirtualHost>

<VirtualHost 172.20.30.40>
DocumentRoot /www/subdomain/sub1
  ServerName www.sub1.domain.tld
  ServerPath /sub1/
  RewriteEngine On
  RewriteRule ^(/sub1/.*) /www/subdomain$1
  # ...
</VirtualHost>

<VirtualHost 172.20.30.40>
  DocumentRoot /www/subdomain/sub2
  ServerName www.sub2.domain.tld
  ServerPath /sub2/
  RewriteEngine On
  RewriteRule ^(/sub2/.*) /www/subdomain$1
  # ...
</VirtualHost>
```

Due to the `ServerPath` directive a request to the URL `http://www.sub1.domain.tld/sub1/` is *always* served from the sub1-vhost.
A request to the URL `http://www.sub1.domain.tld/` is only served from the sub1-vhost if the client sent a correct `Host:` header. If no `Host:` header is sent the client gets the information page from the primary host.
Please note that there is one oddity: A request to `http://www.sub2.domain.tld/sub1/` is also served from the sub1-vhost if the client sent no `Host:` header.
The `RewriteRule` directives are used to make sure that a client which sent a correct `Host:` header can use both URL variants, *i.e.*, with or without URL prefix.

URI References

[1] http://httpd.apache.org/docs-2.1/vhosts/name-based.html
[2] http://httpd.apache.org/docs-2.1/vhosts/ip-based.html