# **Apache HTTP Server Version 2.4**

# **VirtualHost Examples**

This document attempts to answer the commonly-asked questions about setting up virtual hosts (<code>/ index.html</code>). These scenarios are those involving multiple web sites running on a single server, via name-based (<code>/ name-based.html</code>) or IP-based (<code>/ ip-based.html</code>) virtual hosts.

- Running several name-based web sites on a single IP address.
- Name-based hosts on more than one IP address.
- Serving the same content on different IP addresses (such as an internal and external address).
- Running different sites on different ports.
- IP-based virtual hosting
- Mixed port-based and ip-based virtual hosts
- Mixed name-based and IP-based vhosts
- Using Virtual host and mod proxy together
- Using default vhosts
- Migrating a name-based vhost to an IP-based vhost
- Using the ServerPath directive

#### See also

Comments

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

Your server has multiple hostnames that resolve to a single address, and you want to respond differently for www.example.com and www.example.org.

#### 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.

The asterisks match all addresses, so the main server serves no requests. Due to the fact that the virtual host with ServerName www.example.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>.

The above configuration is what you will want to use in almost all name-based virtual hosting situations. The only thing

that this configuration will not work for, in fact, is when you are serving different content based on differing IP addresses or ports.

#### Note

You may replace \* with a specific IP address on the system. Such virtual hosts will only be used for HTTP requests received on connection to the specified IP address.

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.

## 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.example.com and on the other (172.20.30.50), we will serve two or more virtual hosts.

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.example.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.

```
<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. The example below illustrates that the name-matching takes place after the best matching IP address and port combination is determined.

```
Listen 80
Listen 8080
<VirtualHost 172.20.30.40:80>
    ServerName www.example.com
    DocumentRoot "/www/domain-80"
</VirtualHost>
<VirtualHost 172.20.30.40:8080>
    ServerName www.example.com
    DocumentRoot "/www/domain-8080"
</VirtualHost>
<VirtualHost 172.20.30.40:80>
    ServerName www.example.org
    DocumentRoot "/www/otherdomain-80"
</VirtualHost>
<VirtualHost 172.20.30.40:8080>
    ServerName www.example.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.example.com and www.example.org respectively.

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.example.com and www.example.org respectively. In each case, we want to run hosts on ports 80 and 8080.

```
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.example.com
</VirtualHost>
<VirtualHost 172.20.30.40:8080>
    DocumentRoot "/www/example1-8080"
    ServerName www.example.com
</VirtualHost>
<VirtualHost 172.20.30.50:80>
    DocumentRoot "/www/example2-80"
    ServerName www.example.org
</VirtualHost>
<VirtualHost 172.20.30.50:8080>
    DocumentRoot "/www/example2-8080"
    ServerName www.example.org
</VirtualHost>
```

## Mixed name-based and IP-based vhosts

Any address mentioned in the argument to a virtualhost that never appears in another virtual host is a strictly IP-based virtual host.

```
Listen 80
<VirtualHost 172.20.30.40>
    DocumentRoot "/www/example1"
    ServerName www.example.com
</VirtualHost>
<VirtualHost 172.20.30.40>
    DocumentRoot "/www/example2"
    ServerName www.example.org
</VirtualHost>
<VirtualHost 172.20.30.40>
    DocumentRoot "/www/example3"
    ServerName www.example.net
</VirtualHost>
# IP-based
<VirtualHost 172.20.30.50>
    DocumentRoot "/www/example4"
    ServerName www.example.edu
</VirtualHost>
<VirtualHost 172.20.30.60>
    DocumentRoot "/www/example5"
    ServerName www.example.gov
```

## Using Virtual host and mod\_proxy together

The following example allows a front-end machine to proxy a virtual host through to a server running on another machine. In the example, a virtual host of the same name is configured on a machine at 192.168.111.2. The ProxyPreserveHost On directive is used so that the desired hostname is passed through, in case we are proxying multiple hostnames to a single machine.

## 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.

```
<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.

```
<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.

```
<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.

Any use of \* in a virtual host declaration will have higher precedence than \_default\_.

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

The name-based vhost with the hostname www.example.org (from our name-based (/ #name) 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.

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.

```
<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.subl.domain.tld/subl/ is *always* served from the subl-vhost.

A request to the URL http://www.subl.domain.tld/ is only served from the subl-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.

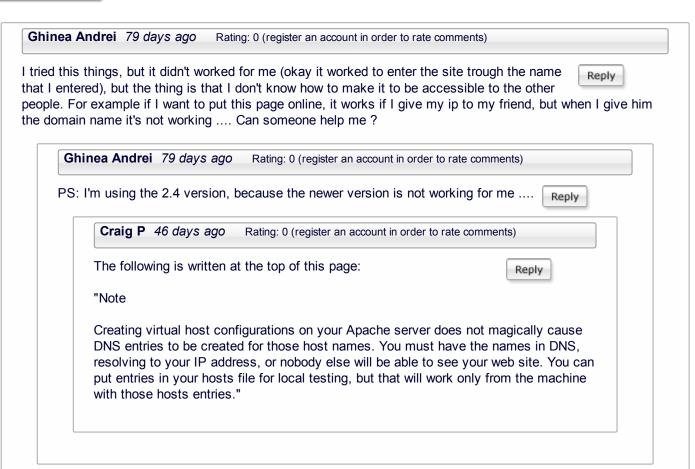
### **Comments**

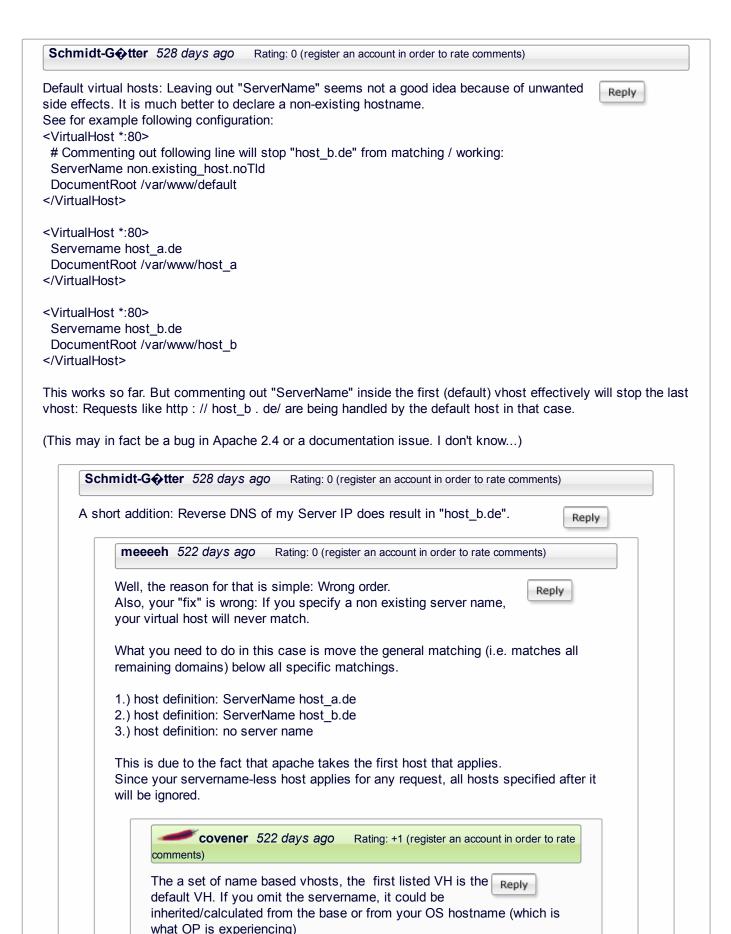
#### **Notice:**

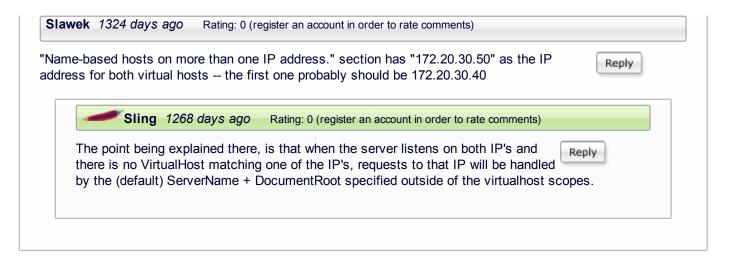
This is not a Q&A section. Comments placed here should be pointed towards suggestions on improving the documentation or server, and may be removed again by our moderators if they are either implemented or considered invalid/off-topic. Questions on how to manage the Apache HTTP Server should be directed at either our IRC channel, #httpd, on Freenode, or sent to our mailing lists.











Copyright 2016 The Apache Software Foundation. Licensed under the Apache License, Version 2.0.