Configure Apache to Host Multiple Virtual Servers

Saturday, February 24, 2018 2:41 PM

NOTE: Assumes you have already installed a LAMP Server (see **Installations**).

Apache separates functionality and components into individual units that can be customized and configured independently. The basic unit that describes an individual site or domain is called a virtual host.

These designations allow administrator to use one server to host multiple domains or sites from a single interface or IP by using a matching mechanism. This is relevant to anyone looking to host more than one site from a single VPS (Virtual Private Server).

Each domain configured directs the visitor to a specific directory holding that site's information, with no indication other sites exit. This scheme is expandable without any software limit if your server can handle the load.

The following examples will show how to set up Apache virtual hosts on Raspbian. This information will show how to serve different content to visitors based on the domains requested.

This example will make virtual hosts for site1.com and site2.com.

Later, you will see how to edit your local hosts file to test the configuration if you are using dummy values. This will allow you to test your configuration from your home computer, even though your content won't be available through the domain name to other visitors.

1. Create the Directory Structure - that will hold the site data that we will be serving to visitors.

   Our document root (the top-level directory that Apache looks at to find content to serve) will be set to individual directories under the /var/www directory. We will create a directory here for both virtual hosts we plan on making.

   Within each of *these* directories, we will create a public html folder that will hold our actual files. This gives us some flexibility in our hosting. For instance, for our sites, we're going to make our directories like this:

   sudo mkdir -p /var/www/site1.com/public_html
   sudo mkdir -p /var/www/site2.com/public_html

   The portions in red represent the domain names that we want to serve from our VPS.

2. Grant Permissions - The directories we created are owned by our root user. We will change ownership so regular users can modify files in our web directories:

   sudo chown -R $USER:$USER /var/www/site1.com/public_html
   sudo chown -R $USER:$USER /var/www/site2.com/public_html

   The $USER variable will take the value of the user you are currently logged in as when you press "ENTER". By doing this, our regular user now owns the public_html subdirectories where we will be storing our content.

   We should also modify our permissions a little bit to ensure that read access is permitted to the

general web directory and all the files and folders it contains so that pages can be served correctly:

sudo chmod -R 755 /var/www

Your web server should now have the permissions it needs to serve content, and your user should be able to create content within the necessary folders.

Create Simple Demo Pages for Each Virtual Host

nano /var/www/site1.com/public_html/index.html
Enter the following code then save and close the file:

```
<html>
  <head>
    <title>Welcome to Site1.com!</title>
  </head>
  <body>
    <h1>Success!  The site1.com virtual host is working!</h1>
  </body>
</html>
```

Copy this file to create the second site:

cp /var/www/site1.com/public_html/index.html /var/www/site2.com/public_html/index.html

Open the file and change site1 to site2 in both places then save and close the file:

nano /var/www/site2.com/public_html/index.html

```
<html>
  <head>
    <title>Welcome to Site2.com!</title>
  </head>
  <body>
    <h1>Success!  The site2.com virtual host is working!</h1>
  </body>
</html>
```

3. Create New Virtual Host Files - specify the actual configuration of our virtual hosts and dictate how the Apache web server will respond to various domain requests.

   Apache comes with a default virtual host file called 000-default.conf that we will copy to create a virtual host file for each of our domains. Start with one domain, configure it, copy it for our second domain, and then make the changes needed. The default configuration requires that each virtual host file end in .conf.

   Create the First Virtual Host File by copying the file for the first domain:

   sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/sites-available/site1.com.conf

   Open the new file in your editor with root privileges:

sudo nano /etc/apache2/sites-available/site1.com.conf

The file will contain information like:

```
<VirtualHost *:80>
   ServerAdmin webmaster@xxxxx.yyy
   DocumentRoot /var/www/html
   ErrorLog ${APACHE_LOG_DIR}/error.log
   CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

We will customize the items here for our first domain and add some additional directives. This virtual host section matches *any* requests that are made on port 80, the default HTTP port.

Change the ServerAdmin directive to an email where the administrator can receive emails:

ServerAdmin admin@site1.com

After this, we need to *add* two directives. The first, called ServerName, establishes the base domain that should match for this virtual host definition. This will most likely be your domain. The second, called ServerAlias, defines further names that should match as if they were the base name. This is useful for matching hosts you defined, like www:

```
ServerName site1.com
ServerAlias www.site1.com
```

The only other thing we need to change for a basic virtual host file is the location of the document root for this domain. We already created the directory we need, so we just need to alter the DocumentRoot directive to reflect the directory we created:

DocumentRoot /var/www/site1.com/public_html

The completed virtualhost file should look like this:

```
<VirtualHost *:80>
   ServerAdmin admin@site1.com
   ServerName site1.com
   ServerAlias www.site1.com
   DocumentRoot /var/www/site1.com/public_html
   ErrorLog ${APACHE_LOG_DIR}/error.log
   CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Save and close the file.

Now that we have our first virtual host file established, we can create our second one by copying that file and adjusting it as needed:

sudo cp /etc/apache2/sites-available/site1.com.conf /etc/apache2/sites-available/site2.com.conf

Open the new file with root privileges in your editor:

sudo nano /etc/apache2/sites-available/site2.com.conf

Modify all the of information necessary to reference your second domain:

```
<VirtualHost *:80>
   ServerAdmin admin@site2.com
   ServerName site1.com
   ServerAlias www.site2.com
   DocumentRoot /var/www/site1.com/public_html
   ErrorLog ${APACHE_LOG_DIR}/error.log
   CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Save and close the file when you are finished.
Enable the New Virtual Host Files using the Apache a2ensite tool:

sudo a2ensite site1.com.conf
sudo a2ensite site2.com.conf

Restart Apache to make these changes take effect:

sudo systemctl reload apache2
sudo service apache2 restart (XXX?)

If you receive a message saying something like: * Restarting web server apache2 AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1. Set the 'ServerName' directive globally to suppress this message, ignore it.

Set Up Local Hosts File (Optional) - If you haven't been using actual domain names that you own to test this procedure and have been using some example domains, you can test the functionality of this process by temporarily modifying the hosts file on your local computer.

This will intercept any requests for the domains that you configured and point them to your VPS server, just as the DNS system would do if you were using registered domains. This will *only work from your computer* and is simply useful for testing purposes.

Make sure you are operating on your local computer and not your VPS server. You will need to know the computer's administrative password or otherwise be a member of the administrative group.

If you are on a Mac or Linux computer, edit your local file with administrative privileges by typing:

sudo nano /etc/hosts

If you are on a Windows machine, you can find instructions on altering your hosts file here.

The details that you need to add are the public IP address of your VPS server followed by the domain you want to use to reach that VPS.

For the domains used in this guide, if the VPS IP address is 111.111.111.111, you could add the following lines to the bottom of the hosts file:

127.0.0.1 localhost
127.0.1.1 guest-desktop
111.111.111.111 site1.com
111.111.111.111 site2.com

This will direct any requests for site1.com and site2.com on our computer and send them to our server at 111.111.111.111. This is what we want if we are not actually the owners of these domains to test our virtual hosts.

Save and close the file.

4. Test your Results – open a browser and enter:

   http://site1.com

   You should see a page that looks like this:

   Success! The site1.com virtual host is working!

   Visit the second page:

   http://site2.com

   Success! The site2.com virtual host is working!

   If both sites work, you have successfully configured **two** virtual hosts on the same server.

   If you adjusted your home computer's hosts file, you may want to delete the lines you added now that you verified that your configuration works. This will prevent your hosts file from being filled with entries that are not actually necessary.

   To access these sites long term, consider purchasing a domain name for each site you need and setting it up to point to your VPS server.

   Conclusion: If you followed along, you should now have a single server handling two separate domain names. You can expand this process by following the steps we outlined above to make additional virtual hosts.

   There is no software limit on the number of domain names Apache can handle, so feel free to make as many as your server is capable of handling.

   NOTE: When you test your sites, If you see this, most likely the index.html file is missing, you are in the wrong directory, don't have permission(s), etc.

By Justin Ellingwood