Ubuntu 14.04.5 LTS

How To Set Up Apache Virtual Hosts

Step One — Create the Directory Structure

The first step that we are going to take is to make a 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 of the 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/example.com/public_html
sudo mkdir -p /var/www/test.com/public_html
```

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

Step Two — Grant Permissions

Now we have the directory structure for our files, but they are owned by our root user. If we want our regular user to be able to modify files in our web directories, we can change the ownership by doing this:

```
sudo chown -R $USER:$USER /var/www/example.com/public_html
sudo chown -R $USER:$USER /var/www/test.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 of 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.

Step Three — Create Demo Pages for Each Virtual Host

We have our directory structure in place. Let's create some content to serve.

We're just going for a demonstration, so our pages will be very simple. We're just going to make an index.html page for each site.

Let's start with example.com. We can open up an index.html file in our editor by typing:

```
nano /var/www/example.com/public_html/index.html
```

In this file, create a simple HTML document that indicates the site it is connected to. My file looks like this:

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

Save and close the file when you are finished.

We can copy this file to use as the basis for our second site by typing:

```
cp /var/www/example.com
/public_html/index.html /var/www/test.com/public_html/index.html
```

We can then open the file and modify the relevant pieces of information:

```
nano /var/www/test.com/public_html/index.html
```

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

Save and close this file as well. You now have the pages necessary to test the virtual host configuration.

Step Four — Create New Virtual Host Files

Virtual host files are the files that 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 can use as a jumping off point. We are going to copy it over to create a virtual host file for each of our domains.

We will start with one domain, configure it, copy it for our second domain, and then make the few further adjustments needed. The default Ubuntu configuration requires that each virtual host file end in .conf.

Create the First Virtual Host File

Start by copying the file for the first domain:

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

Open the new file in your editor with root privileges:

```
sudo nano /etc/apache2/sites-available/example.com.conf
```

The file will look something like this (I've removed the comments here to make the file more approachable):

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

As you can see, there's not much here. 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.

First, we need to change the ServerAdmin directive to an email that the site administrator can receive emails through.

```
ServerAdmin admin@example.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 example.com
ServerAlias www.example.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/example.com/public_html
```

In total, our virtualhost file should look like this:

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

Save and close the file.

Copy First Virtual Host and Customize for Second Domain

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.

Start by copying it:

```
sudo cp /etc/apache2/sites-
available/example.com.conf /etc/apache2/sites-available/test.com.conf
```

Open the new file with root privileges in your editor:

```
sudo nano /etc/apache2/sites-available/test.com.conf
```

You now need to modify all of the pieces of information to reference your second domain. When you are finished, it may look something like this:

```
<VirtualHost *:80>
    ServerAdmin admin@test.com
    ServerName test.com
    ServerAlias www.test.com
    DocumentRoot /var/www/test.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.

Step Five — Enable the New Virtual Host Files

Now that we have created our virtual host files, we must enable them. Apache includes some tools that allow us to do this.

We can use the a2ensite tool to enable each of our sites like this:

```
sudo a2ensite example.com.conf
sudo a2ensite test.com.conf
```

When you are finished, you need to restart Apache to make these changes take effect:

```
sudo service apache2 restart
```

You will most likely receive a message saying something similar to:

```
 * Restarting web server apache2
 AH00558: apache2: Could not reliably determine the server's fully qua
lified domain name, using 127.0.0.1. Set the 'ServerName' directive gl
obally to suppress this message
```

This is a harmless message that does not affect our site.

Unique solution ID: #1006
Author: Ed Casillas
Last update: 2018-07-09 03:08