DigitalOcean   ☰

⌄   ⊞ Subscribe   ⬆ Share   ☰ Contents ⌄

How To Set Up Apache Virtual Hosts on Ubuntu 14.04 LTS

424

Posted April 22, 2014   ◎ 2.4m   APACHE   UBUNTU

By: Justin Ellingwood

Introduction

The Apache web server is the most popular way of serving web content on the internet. It accounts for more than half of all active websites on the internet and is extremely powerful and flexible.

Apache breaks its functionality and components into individual units that can be customized and configured independently. The basic unit that describes an individual site or domain is SCROLL TO TOP
`host`.

These designations allow the administrator to use one server to host multiple domains or sites off of a single interface or IP by using a matching mechanism. This is relevant to anyone looking to host more than one site off of a single VPS.

Each domain that is configured will direct the visitor to a specific directory holding that site's information, never indicating that the same server is also responsible for other sites. This scheme is expandable without any software limit as long as your server can handle the load.

In this guide, we will walk you through how to set up Apache virtual hosts on an Ubuntu 14.04 VPS. During this process, you'll learn how to serve different content to different visitors depending on which domains they are requesting.

Prerequisites

Before you begin this tutorial, you should create a non-root user as described in steps 1-4 here.

You will also need to have Apache installed in order to work through these steps. If you haven't already done so, you can get Apache installed on your server through `apt-get`:

```
sudo apt-get update
sudo apt-get install apache2
```

After these steps are complete, we can get started.

For the purposes of this guide, my configuration will make a virtual host for `example.com` and another for `test.com`. These will be referenced throughout the guide, but you should substitute your own domains or values while following along.

To learn how to set up your domain names with DigitalOcean, follow this link. If you do *not* have domains available to play with, you can use dummy values.

We will show how to edit your local hosts file later on 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.

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.

SCROLL TO TOP

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 goin`   SCROLL TO TOP
`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.
```

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

SCROLL TO TOP

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/
```

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 s **SCROLL TO TOP**
`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/
```

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 don   **SCROLL TO TOP**

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 qualified domain
```

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

Step Six — 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 instead, you can at least test the functionality of this process   SCROLL TO TOP
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 though, and is simply useful for testing purposes.

Make sure you are operating on your local computer for these steps 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 that I used in this guide, assuming that my VPS IP address is `111.111.111.111`, I could add the following lines to the bottom of my hosts file:

```
127.0.0.1    localhost
127.0.1.1    guest-desktop
111.111.111.111 example.com
111.111.111.111 test.com
```

This will direct any requests for `example.com` and `test.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 in order to test our virtual hosts.
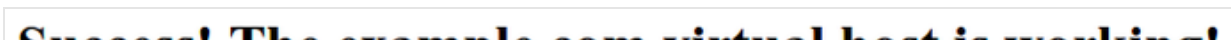
Save and close the file.

Step Seven — Test your Results

Now that you have your virtual hosts configured, you can test your setup easily by going to the domains that you configured in your web browser:

```
http://example.com
```

You should see a page that looks like this:

SCROLL TO TOP

Success! The example.com virtual host is working!

Success! The example.com virtual host is working!

Likewise, if you can visit your second page:

```
http://test.com
```

You will see the file you created for your second site:

Success! The test.com virtual host is working!

If both of these sites work well, you've 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.

If you need to access this 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.

By Justin Ellingwood

By: Justin Ellingwood   Upvote (424)   ⊡ Subscribe   ⬆ Share

SCROLL TO TOP

Introducing Projects on DigitalOcean

Organize your resources according to how you work.

READ MORE

Related Tutorials

How To Migrate your Apache Configuration from 2.2 to 2.4 Syntax.

How To Get Started With mod_pagespeed with Apache on a CentOS and Fedora Cloud Server

How To Use the .htaccess File

How To Set Up Mod_Rewrite (page 2)

How To Create a Custom 404 Page in Apache

---

265 Comments

Leave a comment…

Log In to Comment   **SCROLL TO TOP**

michaelleblanc1 *April 24, 2014*

₀ Got stuck at the "Create the First Virtual Host File" step: my droplet has a file called "default" and "default-ssl" in sites-available, not "default.conf" as stated. And is the subsequent statement: "The default Ubuntu configuration requires that each virtual host file end in .conf." still correct?
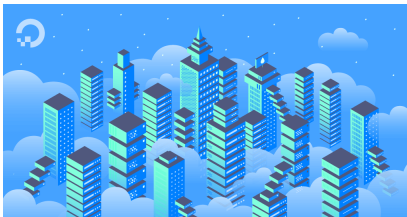
> **joshlsullivan** *February 28, 2015*
>
> ₀ Look for 000-default.conf

asb **MOD** *April 24, 2014*

₂ @michaelleblanc1: It sounds like you are on a different Ubuntu release, perhaps 12.04. This tutorial is specifically talking about 14.04 where the file has been renamed from simply "default" to "000-default.conf" Assuming that you are on 12.04, you should be fine, but you might also want to take a look at this article:

https://www.digitalocean.com/community/articles/how-to-set-up-apache-virtual-hosts-on-ubuntu-12-04-lts

> **How To Set Up Apache Virtual Hosts on Ubuntu 12.04 LTS**
> by Etel Sverdlov
>
> This tutorial shows you how to set up a new directory, how to grant the correct permissions, how to add content to a new page, how to turn on virtual hosting, and how to, if needed, set up local

kerem *April 28, 2014*

₁ Thanks for this great tutorial. My primary virtual host (ie example.com) is alive and well.

My droplet's ip address still points to /var/www/html though and provides the "Apache2 Ubuntu Default Page". I would like my ip to point to the new virtual host I created and hence to /var/www/test.com /public_html. How can I achieve this result?

Thank you.

> **joshlsullivan** *February 28, 2015*
>
> SCROLL TO TOP
>
> ₀ You have to "sudo cp /etc/apache2/sites-available/000-default.conf /etc/apache2/si_

/example.com.conf". The conf file tells the server where to direct traffic. Reread "Step Four — Create New Virtual Host Files". Let me know if I can help further.

---

ftthv *August 24, 2015*

9 you have disable the 000-default.conf

```
sudo a2dissite 000-default.conf
```

---

saifurrahman *December 14, 2016*

0 Thanks,its works!

---

unitoolkit *March 9, 2017*

0 *A million like for you *

---

nispcorp *June 15, 2017*

0 Thank you, you save my day.

---

stefkes *December 6, 2017*

0 thanks!

---

asb **MOD** *April 28, 2014*

0 @Kerem: Do you still have the default configuration enabled? If so, edit the file "/etc/apache2/sites-enabled/000-default.conf" and change "DocumentRoot /var/www/html" to "DocumentRoot /var/www/test.com/public_html"

MorganJohnstone *December 17, 2014*

0 Hi, I've set up two virtual hosts like the article details and I'm having a similar issue to @kerem I want to be able to host two websites on the server, but in the /sites-enabled/000-default.conf there is only space for one (1) document root. How do I add in /var/www/doc/public*html and /var/www/morgan/public*html? I've done (and triple checked) the settings and files in this article. and I'm not sure what

SCROLL TO TOP

jellingwood **MOD** *December 17, 2014*

1 @MorganJohnstone: Hello!

To serve two websites with Apache, you'll have to create a virtual host for each. The main sections that deals with this is Steps 4 and 5.

You'll want a virtual host file (a file that contains the configuration for a site between `<VirtualHost *:80>` and `</VirtualHost>` tags) for each of your sites. In one of them, you can have the document root pointed at `/var/www/doc/public_html`, and in the other, you can have the document root pointed at `/var/www/morgan/public_html`.

The `ServerName` and `ServerAlias` directives in each of these virtual hosts will determine which content will be served by matching the values against the client's request.

When you've finished creating a file for each of your sites, you need to enable them with the `a2ensite` command (this is covered in step 5). Afterwards, be sure to restart Apache.

If you're still having issues after following these steps, post back with exactly what is happening. Hope this helps!

---

kerem *April 29, 2014*

0 @Andrew SB, Thank you very much. This worked perfectly.

---

almeralmazan *May 1, 2014*

0 Thank you very much.. Your guide help me a lot. =)

1000km.s *May 3, 2014*

0 Hi i had troubles pointing the domain to i'ts directory witch can be fixed by editing
example.com.conf
before:
after:
Now is working but i'm not sure that this is right solution.

1000km.s *May 3, 2014*

0 Sorry "which" not "witch ".

SCROLL TO TOP

aveeshkumar *May 7, 2014*

1 I also had to specify

Require all granted

in the conf file as per
http://httpd.apache.org/docs/2.4/upgrading.html

aveeshkumar *May 7, 2014*

0 Posting comment filtered out the <> tags for
lt tag directory /var/www/test.com/public_html/ gt tag
Require all granted
lt tag /directory gt tag

in the conf file as per
http://httpd.apache.org/docs/2.4/upgrading.html

roosterx813 *May 8, 2014*

0 It worked for me,by doing this tutorial i learned about config files, command line, and got a website!
Thank-you,
roosterscomputerrepair.with-linux.com

r2d2t2 *May 20, 2014*

0 This works for me for 12.04 just by creating 000-default.conf instead of default default.conf.
Now I am looking for how to incorporate secure (https) website as I have startssl.com certificate and key.
Thanks for WONDERFUL work D.O.

mulyana2205 *May 22, 2014*

0 i got stuck in sudo a2ensite example.com.conf
my terminal said "Site example.com does not exist"
i miss something, or this tutorial not working in ubuntu 14.10?

SCROLL TO TOP

joshlsullivan *February 28, 2015*

○ The conf must be missing from Apache.

netominas *September 11, 2015*

○ I'm with same errror, where this setting?

sanik *August 24, 2015*

○ happened to me too but then I skipped to edit my hosts file and everything works fine

netominas *September 11, 2015*

○ I'm with same errror, where this setting?

sanik *September 12, 2015*

○ I got an error when command this

a2ensite example.com.conf

instead I proceed to edit my host file

/etc/hosts

kamaln7 **MOD** *September 12, 2015*

○ Hi @netominas, does `/etc/apache2/sites-available/sitename.conf` exist?

sanik *September 13, 2015*

○ yes, you need to create that conf file for each of your domain, then add your domain to the hosts file

netominas *September 17, 2015*

○ ow worked, I was creating the file sitenam without de ".conf". Tks! :)

SCROLL TO TOP

farook *June 15, 2016*

o does /etc/apache2/ exist? i have to disable http to https redirect pls help

asb **MOD** *May 22, 2014*

o @mulyana2205: Does the file " `/etc/apache2/sites-available/example.com.conf` " actually exist? When you run " `sudo a2ensite` " make sure you use the name of the actual file you created.

kasnca *May 27, 2014*

o You don't have permission to access / on this server.

kamaln7 **MOD** *May 28, 2014*

o @kasnca: Do you have an index file in your documentroot?

fabian.socarras *June 7, 2014*

o @kasnca: I was getting the same. Try adding:

Require all granted

It worked for me.

isahappyperson *June 8, 2014*

o i copied the /etc/apache2/sites-available/000-default.conf
over to /etc/apache2/sites-available/ididntsee.com.conf and edit it from there but am still getting the default apache 2 page instead of the index.html page

I created two virtual host using two domain names I own and want to host on one droplet. The IP for one is still propagating but the first one is set and shows the default apache2 page.

SCROLL TO TOP

isahappyperson *June 8, 2014*

Scratch the first comment. I started over from scratch, new server with new ip - directed both domain names to new IP, followed all directions including new non-root user and now all I am getting is 404 not found error on one site and "could not connect to isahappyperson.com" error on the other.

I took each step slowly as to not make any mistakes so I have no clue what to do now. Does anyone know what is wrong?

droplet server is on Ubuntu 14.04 LTS

---

isahappyperson *June 8, 2014*

update:
I directed the domain names to the new IP in the DO dashboard but failed to change them where I registered the domain names.

So, now one site works fine - is showing the custom index.html
but the other site is showing the apache2 default index.html

Any thoughts?

---

cbunting99 *October 14, 2014*

http://www.unixmen.com/setup-virtual-hosts-apache-ubuntu-14-04-lts/

DO is nothing but copy and paste tutorials.. I don't think half of the writers ever try these themselves..

Load More Comments

SCROLL TO TOP

Community   Tutorials   Questions   Projects   Tags   Newsletter   RSS 🔊

Distros & One-Click Apps   Terms, Privacy, & Copyright   Security   Report a Bug   Write for DOnations   Shop

SCROLL TO TOP