How To Set Up Nginx Server Blocks on CentOS 7

Posted November 5, 2014 184.1k NGINX CENTOS

By Josh Barnett
Become an author

Introduction

Nginx is one of the most popular web servers in the world, and is responsible for hosting some of the largest and highest-traffic sites on the Internet. In most cases, Nginx is lighter and more scalable than Apache, and can be used as a web server or as a reverse proxy.

Nginx uses **server blocks** to manage configurations for an individual site or domain. Server blocks allow one server to host multiple domains or interfaces by using a matching system. 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, without ever 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 traffic that all of the sites attract.

In this guide, we will walk through how to set up Nginx server blocks on a CentOS 7 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 with this guide, there are a few steps that need to be completed first.

ivileges. If
setup

You will also need to have Nginx installed in order to configure server blocks for it. If you want an entire LEMP (Linux, Nginx, MySQL, and PHP) stack on your server, you can follow our guide on setting up a LEMP stack in CentOS 7. If you only need Nginx, you can install it through Nginx's `yum` repository:

First, add the Nginx repository to your server's list of software sources.

```
sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch
```

Now you can use `yum` to download and install Nginx.

```
sudo yum install nginx
```

After these steps are complete, log in as your non-root user account through SSH and continue with the tutorial.

Note: The example configuration in this guide will make one server block for `example.com` and another for `example2.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 any real domains to play with, we will show you how to test your server block configuration with dummy values near the end of the tutorial.

Step One — Create the Directory Structure

First, we need to make a directory structure that will hold the site data to serve to visitors.

Our **document root** (the top-level directory that Nginx looks at to find content to serve) will be set to individual directories in the `/var/www` directory. We will create a directory here for each of the server blocks that we plan on making.

Within each of these directories, we will create an `html` directory that will hold our actual files. This gives us some flexibility in our hosting.

llows us to

```
sudo mkdir -p /var/www/example.com/html
sudo mkdir -p /var/www/example2.com/html
```

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

Grant Permissions

We now 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 with `chown`:

```
sudo chown -R $USER:$USER /var/www/example.com/html
sudo chown -R $USER:$USER /var/www/example2.com/html
```

The `$USER` variable will take the value of the user you are currently logged in as when you submit the command. 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 inside, 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 appropriate folders.

Step Two — Create Demo Pages for Each Site

Now that we have our directory structure in place, let's create some content to serve.

Because this is just for demonstration and testing, our pages will be very simple. We are just going to make an `index.html` page for each site that identifies that specific domain.

typing:

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

In this file, create a simple HTML document that indicates the site that the page is connected to. For this guide, the file for our first domain will look like this:

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

Save and close the file when you are finished.

We can copy this file to use as the template for our second site's `index.html` by typing:

```
cp /var/www/example.com/html/index.html /var/www/example2.com/html/index.html
```

Now let's open that file and modify the relevant pieces of information:

```
nano /var/www/example2.com/html/index.html
```

```
<html>
  <head>
    <title>Welcome to Example2.com!</title>
  </head>
  <body>
    <h1>Success! The example2.com server block is working!</h1>
  </body>
</html>
```

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

Server block files are what specify the configuration of our separate sites and dictate how the Nginx web server will respond to various domain requests.

To begin, we will need to set up the directory that our server blocks will be stored in, as well as the directory that tells Nginx that a server block is ready to serve to visitors. The `sites-available` directory will keep all of our server block files, while the `sites-enabled` directory will hold symbolic links to server blocks that we want to publish. We can make both directories by typing:

```
sudo mkdir /etc/nginx/sites-available
sudo mkdir /etc/nginx/sites-enabled
```

Note: This directory layout was introduced by Debian contributors, but we are including it here for added flexibility with managing our server blocks (as it's easier to temporarily enable and disable server blocks this way).

Next, we should tell Nginx to look for server blocks in the `sites-enabled` directory. To accomplish this, we will edit Nginx's main configuration file and add a line declaring an optional directory for additional configuration files:

```
sudo nano /etc/nginx/nginx.conf
```

Add these lines to the end of the `http {}` block:

```
include /etc/nginx/sites-enabled/*.conf;
server_names_hash_bucket_size 64;
```

The first line instructs Nginx to look for server blocks in the `sites-enabled` directory, while the second line increases the amount of memory that is allocated to parsing domain names (since we are now using multiple domains).

When you are finished making these changes, you can save and close the file. We are now ready to create our first server block file.

Create the First Server Block File

se as a
g file by

```
sudo cp /etc/nginx/conf.d/default.conf /etc/nginx/sites-available/example.com.conf
```

Now, open the new file in your text editor with root privileges:

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

Note: Due to the configurations that we have outlined, all server block files *must* end in `.conf`.

Ignoring the commented lines, the file will look similar to this:

```
server {
    listen  80;
    server_name localhost;

    location / {
        root  /usr/share/nginx/html;
        index  index.html index.htm;
    }

    error_page  500 502 503 504  /50x.html;
    location = /50x.html {
        root  /usr/share/nginx/html;
    }
}
```

The first thing that we're going to have to adjust is the `server_name`, which tells Nginx which requests to point to this server block. We'll declare the main server name, `example.com`, as well as an additional alias to `www.example.com`, so that both `www.` and non-`www.` requests are served the same content:

```
server_name example.com www.example.com;
```

Note: Each Nginx statement must end with a semi-colon (;), so check each of your statement lines if you are running into problems later on.

int it to

```
root /var/www/example.com/html;
```

We'll also want to add a `try_files` command that ends with a 404 error if the desired filename or directory is not found:

```
try_files $uri $uri/ =404;
```

When you are finished, your file will look something like this:

```
server {
    listen  80;

    server_name example.com www.example.com;

    location / {
        root  /var/www/example.com/html;
        index  index.html index.htm;
        try_files $uri $uri/ =404;
    }

    error_page  500 502 503 504  /50x.html;
    location = /50x.html {
        root  /usr/share/nginx/html;
    }
}
```

That is all we need for a basic configuration, so save and close the file to exit.

Create the Second Server Block File

Now that we have our first server block file established, we can create our second one by copying that file and adjusting it as needed.

Start by copying it with `cp`:

```
sudo cp /etc/nginx/sites-available/example.com.conf /etc/nginx/sites-available/example2.com.conf
```

```
sudo nano /etc/nginx/sites-available/example2.com.conf
```

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

```
server {
    listen  80;

    server_name example2.com www.example2.com;

    location / {
        root  /var/www/example2.com/html;
        index  index.html index.htm;
        try_files $uri $uri/ =404;
    }

    error_page  500 502 503 504  /50x.html;
    location = /50x.html {
        root  /usr/share/nginx/html;
    }
}
```

When you are finished making these changes, you can save and close the file.

Step Four — Enable the New Server Block Files

Now that we have created our server block files, we need to enable them so that Nginx knows to serve them to visitors. To do this, we can create a symbolic link for each server block in the `sites-enabled` directory:

```
sudo ln -s /etc/nginx/sites-available/example.com.conf /etc/nginx/sites-enabled/example.com.conf
sudo ln -s /etc/nginx/sites-available/example2.com.conf /etc/nginx/sites-enabled/example2.com.conf
```

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

Step Five — Set Up Local Hosts File (Optional)

If you have been using example domains instead of actual domains to test this procedure, you can still test the functionality of your server blocks 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. However, this will only work from your local computer, and is simply useful for testing purposes.

Note: Make sure that you are operating on your local computer for these steps and not your VPS server. You will need access to the administrative credentials for that computer.

If you are on a Mac or Linux computer, edit your local `hosts` 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 followed by the domain that you want to use to reach that VPS:

```
127.0.0.1   localhost
127.0.1.1   guest-desktop
server_ip_address example.com
server_ip_address example2.com
```

This will direct any requests for `example.com` and `example2.com` on our local computer and send them to our server at `server_ip_address`.

Step Six — Test Your Results

Now that you have your server blocks configured, you can test your setup easily by going

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

You should see a page that looks like this:

Likewise, if you visit your other domains, you will see the files that you created for them.

If all of the sites that you configured work well, then you have successfully configured your new Nginx server blocks on the same CentOS server.

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

Conclusion

At this point, you should now have a single CentOS 7 server handling multiple sites with separate domains. You can expand this process by following the steps we outlined above to make additional server blocks later. There is no software limit on the number of domain names Nginx can handle, so feel free to make as many as your server is capable of handling.

By Josh Barnett

Related

TUTORIAL

How To Configure Nginx to Use Custom Error Pages on CentOS 7

Nginx is a high performance web server capable of serving content with flexibility and power. ...

TUTORIAL

How To Redirect www to Non-www with Nginx on CentOS 7

This tutorial will show you how to redirect a www URL to non-www, e.g. `www.example.com` to ...

TUTORIAL

How To Install Nginx on CentOS 7

This articles covers how to install Nginx, a web server, on a CentOS 7 virtual server. This can be done with the CentOS ...

TUTORIAL

How To Install Node.js on CentOS 8

In this guide, we will show you three different ways of getting Node.js installed on a CentOS 8 server: using `dnf` to install the ...

Still looking for an answer?

Ask a question

Search for more help

Leave a comment...

Sign In to Comment

klaudiom November 9, 2014

1

Hi,

You have a typo. Missing semicolon on listen 80.

I have followed these instructions but it seems like it's not pointing in the directory I've created.

Are you sure these configs shouldn't be in `/etc/nginx/conf.d/` ?

Reply Report

> **jbarnett** November 10, 2014
>
> 0
>
> Thank you for pointing out that typo. It has been corrected.
>
> In step three, we covered how to make sure that Nginx reads the server block directories that we created. You need to add `include /etc/nginx/sites-enabled/*.conf;` to the `http{}` block in `/etc/nginx/nginx.conf`. Once that's in place, be sure to restart Nginx to apply the changes: `sudo systemctl restart nginx`
>
> Reply Report

mediagenic January 15, 2015

0

Hey Josh,

Great article!

Just wondering, would there be anyway to setup a automatic server block creation once a new site has been added to the DNS? I have noticed there is a wildcard option stated here;

Also because I am currently not using a domain for root, I had to setup server block for the IP, although I'm sure this should of been the root var/www/html/ folder anyway? for some reason it wasn't picking this up and I had to create a folder var/www/IPADDRESS/html/

Any ideas?

Reply Report

rrrmanish March 8, 2015

1

under grant permissions chown code ,it should be "public_html" not "html" in the end

Reply Report

> admin9347e85d2ac3470a68143 March 15, 2017
>
> 0
>
> Wish they'd edit that.
>
> Reply Report

jacobwmead April 29, 2015

0

I have set up a couple of these server blocks, and every time it is set up the domains are getting files only from /usr/share/nginx/html

I have even tried creating ones with the exact code above and using http://example.com as my hostmod

Reply Report

> jacobwmead April 29, 2015
>
> 0
>
> I figured out its the symlink from sites-available to sites-enabled. I set it up to go to a regular file in sites-enabled for now until i figure out why my symlinks won't work
>
> Reply Report

alanacweaver February 7, 2016

0

If I already have server configured and PrestaShop installed on one domain, how do I create the server blocks for my other domains?

Reply Report

rroopstr May 10, 2016

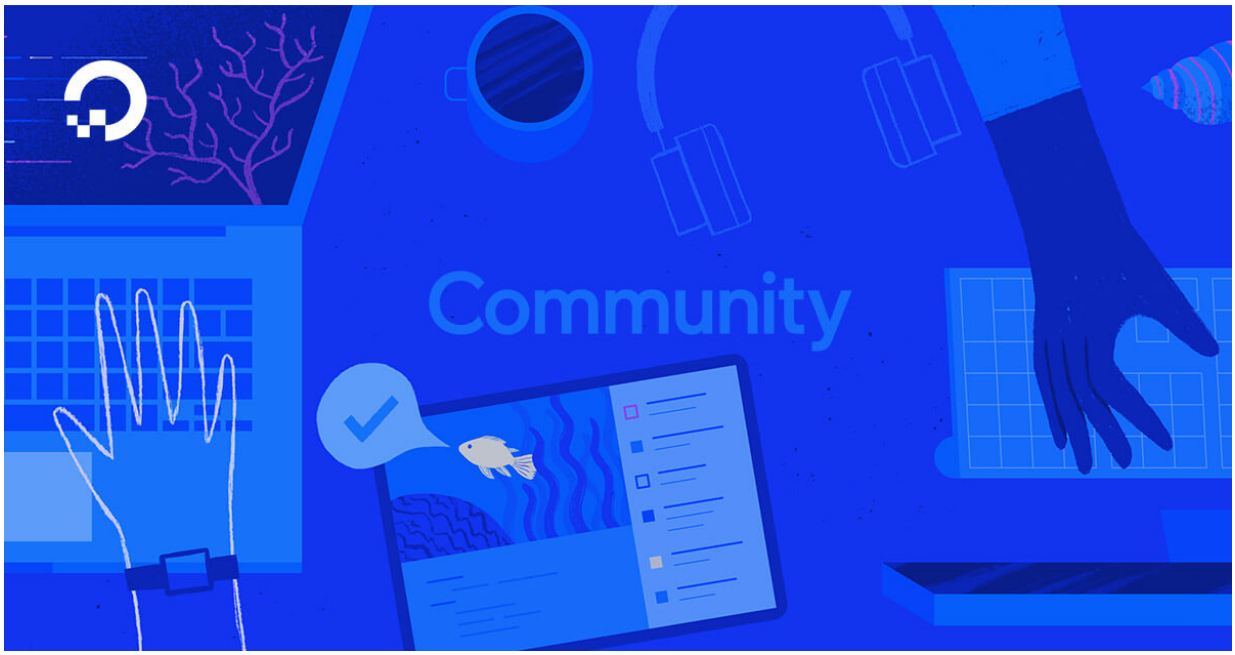
0

Hi I configured nginx and when I write my ip address on the browser i get
"Welcome to nginx! If you see this page, the nginx web server is successfully installed and working. Further configuration is required. For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com. Thank you for using nginx."

been very careful with folders names and sudo vim commands, everything seems fine, however when typing my domains on a web browser i get DNS error:

ERR*NAME*NOT_RESOLVED

My old registrar already points to my new IP address provided by the new VPS provider. They provided me initially a VPS with CentOs 6.7 and Zpanel, where I succesfully uploaded all my websites. A week later I decided to remove ZPanel and install some new software, so I upgraded into Centos 7.2 and am in the process of configuring LEMP server.



How To Set Up Nginx Server Blocks on CentOS 7
by Josh Barnett

Nginx uses to manage configurations for an individual site or domain. Server blocks allow one server to host multiple domains or interfaces by using a matching system. This is relevant to anyone looking to host more than one site off of a single VPS. In

Reply Report

dhruba May 24, 2016

0

The digital ocean guides are a fantastic and expansive resource. I've applied so many of them now. This one was a gem and worked first time. Thank you!

Reply Report

Palkesh June 6, 2017

0

Hi Josh Barnett!

Nice tutorial but one problem.

sudo ln -s /etc/nginx/sites-available/example.com.conf /etc/nginx/sites-enabled/example.com.conf

It should be this

sudo ln -s /etc/nginx/sites-available/example.com.conf /etc/nginx/sites-enabled/

Reply Report

haxpor7322349dd August 13, 2017

0

Thank you so much! Very informative and works great!
Especially I like the set up in Debian to have sites-available, and sites-enabled.

Reply Report

kulinaleks88 September 28, 2017

0

A disgusting article.
I'm very sorry that I started configuring my server for it.
Everything works with .html files, but as soon as I deal with .php, I download the file. How to solve this problem?

Reply Report

ramijames November 9, 2017

0

You have to enable PHP. This is a separate process. Make sure that PHP is properly installed.

My conf looks like

```
server {
    listen  80;

    server_name site.co www.site.co;

    location / {
        root  /var/www/site.co;
        index index.php;
        try_files $uri $uri/ =404;
    }

    error_page  500 502 503 504  /50x.html;
    location = /50x.html {
        root  /var/www/site.co;
    }
```

```
            fastcgi_index index.php;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            include fastcgi_params;
        }

    }
```

Reply Report

yyhtech June 20, 2018

1

I already finished all steps. But Still error “403 Forbidden nginx/1.14.0” in two sites. How can I fix that?

Reply Report

rizal4 February 12, 2019

0

same with me.Anyone can help us

Reply Report

rizal4 February 13, 2019

0

I follow this article and successfull done it..

https://linuxize.com/post/how-to-set-up-nginx-server-blocks-on-centos-7/

Reply Report

showaltb August 1, 2019

1

If selinux is enforcing, you’ll need to change the context of the files. I resolved this with:

```
chcon -R -t httpd_sys_content_t /var/www
semanage fcontext -a -t httpd_sys_content_t /var/www
```

Reply Report

bishale26c September 11, 2019

0

Hi @showaltb - could you please elaborate on this?

Reply Report

dr4dr4 April 19, 2019

0

Just to let anybody know when copying the config files and editing the details.
The config file is not in said directory but one up from there.

ot be

I just removed everything other than the server block declarations and it's now starting as it should.

Apologies if this is obvious but it was not obvious to me straight from the instructions.

Reply Report

BECOME A CONTRIBUTOR

You get paid; we donate to tech nonprofits.

GET OUR BIWEEKLY NEWSLETTER

Company

About
Leadership
Blog
Careers
Partners
Referral Program
Press
Legal & Security

Products

Products Overview
Pricing
Droplets
Kubernetes
Managed Databases
Spaces
Marketplace
Load Balancers
Block Storage
Tools & Integrations
API
Documentation
Release Notes

Community

Tutorials
Q&A
Tools and Integrations
Tags
Product Ideas
Meetups
Write for DOnations
Droplets for Demos
Hatch Startup Program
Shop Swag
Research Program
Open Source
Code of Conduct

Contact

Get Support
Trouble Signing In?
Sales
Report Abuse
System Status