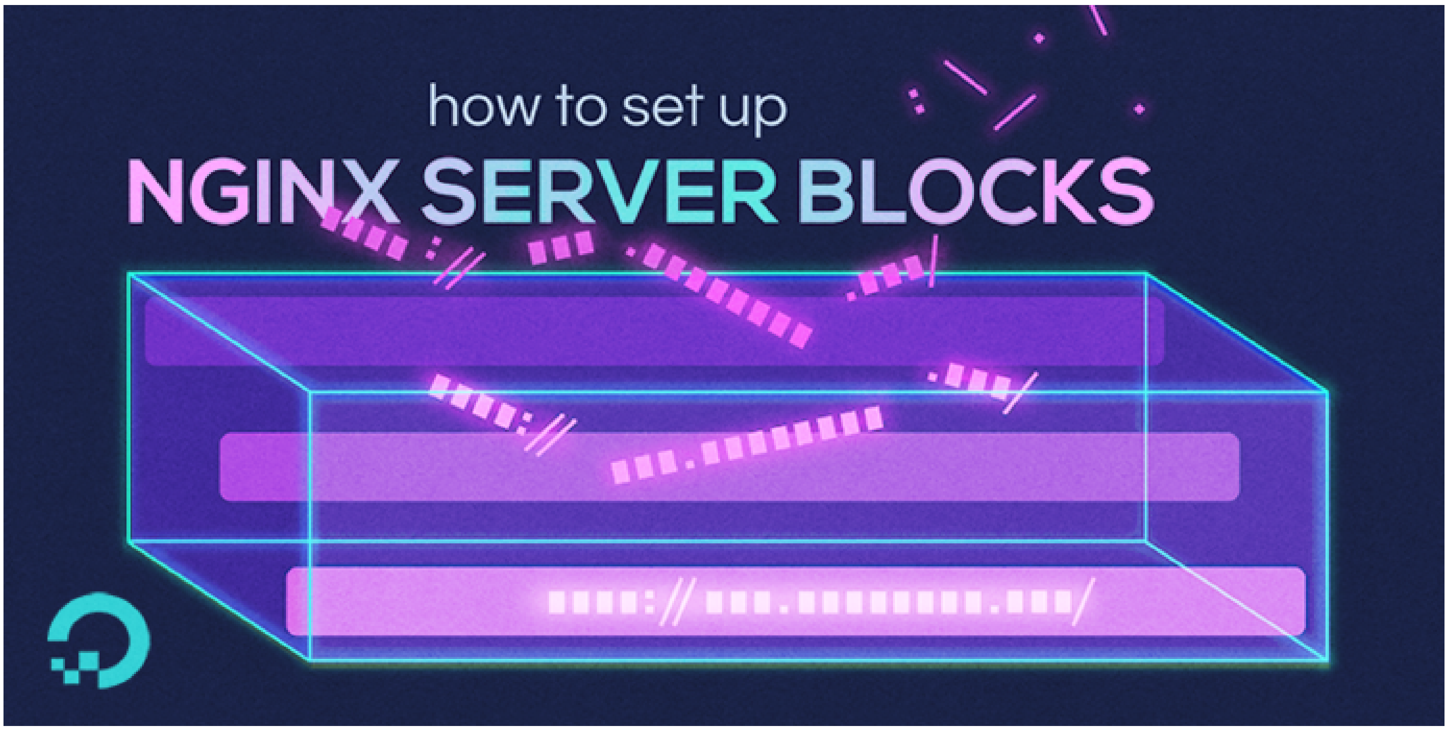
DigitalOcean

Subscribe



How To Set Up Nginx Server Blocks (Virtual Hosts) on Ubuntu 16.04

164

Posted May 19, 2016 734.9k GETTING STARTED NGINX UBUNTU UBUNTU 16.04

By: Justin Ellingwood

Introduction

When using the Nginx web server, **server blocks** (similar to the virtual hosts in Apache) can be used to encapsulate configuration details and host more than one domain off of a single server.

In this guide, we'll discuss how to configure server blocks in Nginx on an Ubuntu 16.04 server.

Prerequisites

We're going to be using a non-root user with `sudo` privileges throughout this tutorial. If you do

initial

You will also need to have Nginx installed on your server. The following guides cover this procedure:

- How To Install Nginx on Ubuntu 16.04: Use this guide to set up Nginx on its own.
- How To Install Linux, Nginx, MySQL, PHP (LEMP stack) in Ubuntu 16.04: Use this guide if you will be using Nginx in conjunction with MySQL and PHP.

When you have fulfilled these requirements, you can continue on with this guide.

Example Configuration

For demonstration purposes, we're going to set up two domains with our Nginx server. The domain names we'll use in this guide are **example.com** and **test.com**.

You can find a guide on how to set up domain names with DigitalOcean here. If you do not have two spare domain names to play with, use dummy names for now and we'll show you later how to configure your local computer to test your configuration.

Step One: Set Up New Document Root Directories

By default, Nginx on Ubuntu 16.04 has one server block enabled by default. It is configured to serve documents out of a directory at `/var/www/html`.

While this works well for a single site, we need additional directories if we're going to serve multiple sites. We can consider the `/var/www/html` directory the default directory that will be served if the client request doesn't match any of our other sites.

We will create a directory structure within `/var/www` for each of our sites. The actual web content will be placed in an `html` directory within these site-specific directories. This gives us some additional flexibility to create other directories associated with our sites as siblings to the `html` directory if necessary.

We need to create these directories for each of our sites. The `-p` flag tells `mkdir` to create any necessary parent directories along the way:

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

Now that we have our directories, we will reassign ownership of the web directories to our normal user account. This will let us write to them without `sudo`.

Note

Depending on your needs, you might need to adjust the permissions or ownership of the folders again to allow certain access to the `www-data` user. For instance, dynamic sites will often need this. The specific permissions and ownership requirements entirely depend on what your configuration. Follow the recommendations for the specific technology you're using.

We can use the `$USER` environmental variable to assign ownership to the account that we are currently signed in on (make sure you're not logged in as `root`). This will allow us to easily create or edit the content in this directory:

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

The permissions of our web roots should be correct already if you have not modified your `umask` value, but we can make sure by typing:

```
$ sudo chmod -R 755 /var/www
```

Our directory structure is now configured and we can move on.

Step Two: Create Sample Pages for Each Site

Now that we have our directory structure set up, let's create a default page for each of our sites so that we will have something to display.

Create an `index.html` file in your first domain:

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

Inside the file, we'll create a really basic file that indicates what site we are currently accessing. It will look like this:

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

```
<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.

Since the file for our second site is basically going to be the same, we can copy it over to our second document root like this:

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

Now, we can open the new file in our editor:

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

Modify it so that it refers to our second domain:

/var/www/test.com/html/index.html

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

Save and close this file when you are finished. We now have some pages to display to visitors of our two domains.

Step Three: Create Server Block Files for Each Domain

Now that we have the content we wish to serve, we need to actually create the server blocks that will tell Nginx how to do this.

By default, Nginx contains one server block called `default` which we can use as a template for our own configurations. We will begin by designing our first domain's server block, which we will then copy over for our second domain and make the necessary modifications.

Create the First Server Block File

As mentioned above, we will create our first server block config file by copying over the default file:

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

Now, open the new file you created in your text editor with `sudo` privileges:

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

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

/etc/nginx/sites-available/example.com

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

First, we need to look at the listen directives. **Only one of our server blocks on the server can have the `default_server` option enabled.** This specifies which block should serve a request if the `server_name` requested does not match any of the available server blocks. This shouldn't happen very frequently in real world scenarios since visitors will be accessing your site through your domain name.

You can choose to designate one of your sites as the "default" by including the `default_server` option in the `listen` directive, or you can leave the default server block enabled, which will serve the content of the `/var/www/html` directory if the requested host cannot be found.

In this guide, we'll leave the default server block in place to serve non-matching requests, so we'll remove the `default_server` from this and the next server block. You can choose to add the option to whichever of your server blocks makes sense to you.

/etc/nginx/sites-available/example.com

```
server {
        listen 80;
        listen [::]:80;
        . . .
}
```

Note

You can check that the `default_server` option is only enabled in a single active file by typing:

```
$ grep -R default_server /etc/nginx/sites-enabled/
```

If matches are found uncommented in more than on file (shown in the leftmost column), Nginx will complain about an invalid configuration.

The next thing we're going to have to adjust is the document root, specified by the `root` directive. Point it to the site's document root that you created:

/etc/nginx/sites-available/example.com

```
server {
        listen 80;
        listen [::]:80;

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

}
```

Next, we need to modify the `server_name` to match requests for our first domain. We can additionally add any aliases that we want to match. We will add a `www.example.com` alias to demonstrate.

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

/etc/nginx/sites-available/example.com

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/example.com/html;
        index index.html index.htm index.nginx-debian.html;

        server_name example.com www.example.com;
        location / {
                try_files $uri $uri/ =404;
        }
}
```

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

Create the Second Server Block File

Now that we have our initial server block configuration, we can use that as a basis for our second file. Copy it over to create a new file:

```
$ sudo cp /etc/nginx/sites-available/example.com /etc/nginx/sites-available/test.co
```

Open the new file with `sudo` privileges in your editor:

```
$ sudo nano /etc/nginx/sites-available/test.com
```

Again, make sure that you do not use the `default_server` option for the `listen` directive in this file if you've already used it elsewhere. Adjust the `root` directive to point to your second domain's document root and adjust the `server_name` to match your second site's domain name (make sure to include any aliases).

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

/etc/nginx/sites-available/test.com

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/test.com/html;
        index index.html index.htm index.nginx-debian.html;

        server_name test.com www.test.com;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

When you are finished, save and close the file.

Step Four: Enable your Server Blocks and Restart Nginx

Now that we have our server block files, we need to enable them. We can do this by creating symbolic links from these files to the `sites-enabled` directory, which Nginx reads from during startup.

We can create these links by typing:

```
$ sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/
$ sudo ln -s /etc/nginx/sites-available/test.com /etc/nginx/sites-enabled/
```

These files are now in the enabled directory. We now have three server blocks enabled, which are configured to respond based on their `listen` directive and the `server_name` (you can read more about how Nginx processes these directives here):

- `example.com`: Will respond to requests for `example.com` and `www.example.com`
- `test.com`: Will respond to requests for `test.com` and `www.test.com`
- `default`: Will respond to any requests on port 80 that do not match the other two blocks.

In order to avoid a possible hash bucket memory problem that can arise from adding additional server names, we will go ahead and adjust a single value within our `/etc/nginx/nginx.conf` file. Open the file now:

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

Within the file, find the `server_names_hash_bucket_size` directive. Remove the `#` symbol to uncomment the line:

/etc/nginx/nginx.conf

```
http {
    . . .

    server_names_hash_bucket_size 64;

    . . .
}
```

Save and close the file when you are finished.

Next, test to make sure that there are no syntax errors in any of your Nginx files:

```
$ sudo nginx -t
```

If no problems were found, restart Nginx to enable your changes:

```
$ sudo systemctl restart nginx
```

Nginx should now be serving both of your domain names.

Step Five: Modify Your Local Hosts File for Testing(Optional)

If you have not been using domain names that you own and instead have been using dummy values, you can modify your local computer's configuration to let you to temporarily test your Nginx server block configuration.

This will not allow other visitors to view your site correctly, but it will give you the ability to reach each site independently and test your configuration. This basically works by intercepting requests that would usually go to DNS to resolve domain names. Instead, we can set the IP addresses we want our local computer to go to when we request the domain names.

Note

Make sure you are operating on your local computer during these steps and not your VPS server. You will need to have root access, be a member of the administrative group, or otherwise be able to edit system files to do this.

If you are on a Mac or Linux computer at home, you can edit the file needed by typing:

```
local $ sudo nano /etc/hosts
```

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

You need to know your server's public IP address and the domains you want to route to the server. Assuming that my server's public IP address is `203.0.113.5`, the lines I would add to my file would look something like this:

/etc/hosts

```
127.0.0.1   localhost
. . .
```

```
203.0.113.5 example.com www.example.com
203.0.113.5 test.com www.test.com
```

This will intercept any requests for `example.com` and `test.com` and send them to your server, which is what we want if we don't actually own the domains that we are using.

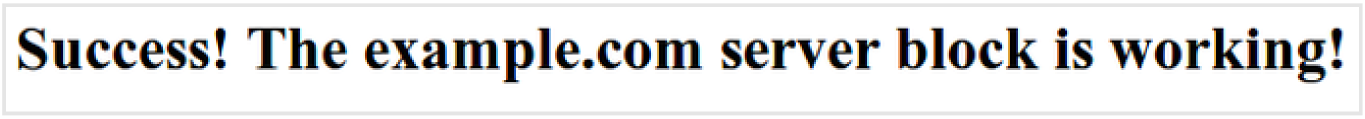
Save and close the file when you are finished.

Step Six: Test your Results

Now that you are all set up, you should test that your server blocks are functioning correctly. You can do that by visiting the domains in your web browser:

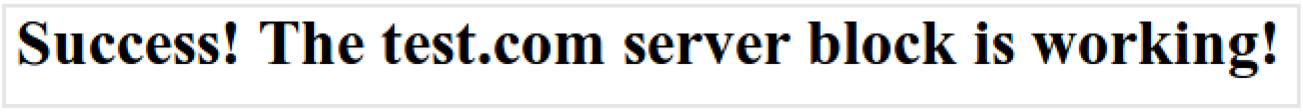
```
http://example.com
```

You should see a page that looks like this:


Success! The example.com server block is working!

If you visit your second domain name, you should see a slightly different site:

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


Success! The test.com server block is working!

If both of these sites work, you have successfully configured two independent server blocks with Nginx.

At this point, if you adjusted your `hosts` file on your local computer in order to test, you'll probably want to remove the lines you added.

If you need domain name access to your server for a public-facing site, you will probably want to purchase a domain name for each of your sites. You can learn how to set them up to point to your server here.

Conclusion

You should now have the ability to create server blocks for each domain you wish to host from the same server. There aren't any real limits on the number of server blocks you can create, so long as your hardware can handle the traffic.

By: Justin Ellingwood

Upvote (164) Subscribe

Related Tutorials

How to Deploy a Symfony 4 Application to Production with LEMP on Ubuntu 18.04

How To Install WordPress with LEMP on Debian 9

How To Install Linux, Nginx, MySQL, PHP (LEMP stack) on Debian 9

How To Create a Self-Signed SSL Certificate for Nginx on Debian 9

How To Set Up Django with Postgres, Nginx, and Gunicorn on Debian 9

53 Comments

Leave a comment...

Log In to Comment

iamkingsleyf *May 28, 2016*

1 Hello;

I have a user "**dev**" and the site is installed in dev directory. whenever i upload a .zip file like wp theme and extract it permissions check to "**root**" instead of "**dev**" who is added to the sudo group.

any idea how to do this? i don't want to be changing permissions whenever i upload themes via command like.

On serverpilot when if you upload the file with root, so long as you extract it with **serverpilot** the file permission goes to **serverpilot** user.

Thanks

squarology *June 16, 2016*

1 thank you so much!!

san22 *July 3, 2016*

1 Thanks .. very well written..
I noticed a typo..
to check that the default*server option is only enabled in a single active file by typing:*
*grep -R default*server /etc/nginx/sites-enabled/

should be
grep -R default_server /etc/nginx/sites-available/

tobyforever *November 4, 2016*

2 sites-enabled is symlink for enabled site configs so it is ok

georgetour *August 9, 2016*

1 This how guides should be made.Thank you.

luismuzquiz *September 13, 2016*

1 Hello. What´s the difference between

`listen 80;`
and `listen [::]:80;`

Thanks in advanced!

jellingwood MOD *September 13, 2016*

2 @luismuzquiz: Hello! The first line applies to connections made using IPv4, while the second line is used for IPv6 connections. Both are needed if you desire connectivity using both protocols. Hope that helps!

luismuzquiz *September 14, 2016*

1 thanks!

govindsaini1986 *September 21, 2016*

1 Hi
I have followed all the steps but site is not running. when I access domain yournexthosting.com in browser it download a file automatically.
when I access yournexthosting.com/index.html it downloads index.html page..
plz help.

adrianvalenz *September 26, 2016*

1 Yea I'm having a similar issue. I just get the Nginx welcome screen.

phlozzle *October 22, 2016*

1 What are some specifics? I might be able to help debug this.

phlozzle *October 22, 2016*

1 Thanks for this, it's super helpful. I wanted to host a Django server and node server on one Ubuntu instance, and this got me started in the right direction. I wrote a little bit about setting up multiple server blocks

brandonb3daf1f9 *October 25, 2016*

1 I am using LEMP on Ubutu 16.04 with a virtual host(server block). When I log into phpmyadmin, I get a 404 page. But when I go back to /phpmyadmin I am logged in. So it is logging me in, but redirecting to a 404. The url (after logging in) is index.php(Plus a ton of other characters). Any idea what is going on?

---

meteoroxide *December 20, 2016*

1 Its totally shocking. In every article since 2012, you always skip the part where you are supposed to create the virtual host and instead copy the default.

---

block2trade *January 21, 2017*

1 can anyone help at the end I get an error 404 not found? it happens to any site even if I use http://ipadd/test.php same error

---

luis02lopez *January 28, 2017*

1 @jellingwood
Thanks in advance for the tutorial.
I'm experiment some problems, I follow all the steps and this is my server configuration:

```
http://pastebin.com/6E3mdjrR
```

Even though when I go to:

```
http://kinbu.localhost/
```

I got "HTTP ERROR 500" page It's not working.

Plase help. Thanks.

---

bambangyudhotomo *March 13, 2017*

1 this tutorial work great. But www.example.com not working. Only example.com work.
I already added CNAME for www
could you help me please?

---

jellingwood MOD *March 13, 2017*

0

@bambangyudhotomo It can take awhile for DNS to propogate (up to 48 hours in some cases) so if it was a recent change, I would wait a bit longer. The other component is to make sure that your Nginx server blocks are set up to respond to `www.example.com` requests as well.

In your Nginx server blocks, make sure that your `server_name` points to both the bare domain *as well as* the "www" domain. So in this case, make sure your server block is set up like this:

```
server {
    . . .
    server_name example.com www.example.com;
    . . .
}
```

If you forget to do this, Nginx won't know which block to use for `www.example.com` requests. Don't forget to restart Nginx to pick up your changes. Hope that helps.

---

bambangyudhotomo *March 14, 2017*

0 @jellingwood thank you so much. It worked! I forgot that it need up to 48 hours for DNS to propogate.

---

devlim *March 14, 2017*

0 i try to use domain *demo* www.demo.demo, it redirect me to router page. the 192.168.0.1 Page

am i using wrong server's public IP address? and i used `curl ifconfig.me` to get my server public IP address.

anyone encounter this issue?

---

bambangyudhotomo *April 4, 2017*

0 Can I place server block from /var/www/example.com/html to another directory like /home/someuser/example.com/html ?

---

chinmayrajyaguru *April 25, 2017*

0 Thank you for making a beautiful & clean tutorial. I would like to know, **How can I create subdomain?**
Again, thanks!

michiel0 *April 28, 2017*

0 Hello,

Unfortunately I am stuck at step three because I can't find the /etc/nginx/sites-available/ and site-enable directories / config files in my clean nginx install. The server is running and shows the 'welcome to nginx' message in my browser. That html file is located in /usr/share/nginx/html. But I cant figure out whats going wrong while missing the default server block config in /etc/nginx/... Many thanks in advance for your help!!

---

jellingwood MOD *April 28, 2017*

0 @michiel0 If you do not have an `/etc/nginx/sites-available` or `/etc/nginx/sites-enabled` directory, it's likely a sign that you have installed Nginx from a location other than Ubuntu's default repositories (like the Nginx project's repositories) or that you are trying to complete this guide on a different version of Ubuntu. This guide assumes that you are using the Nginx package available in Ubuntu 16.04's default repositories. If your setup looks different, it's very possible this won't work.

---

nikolakovacevicvz *May 21, 2017*

0 Hi!

I have followed this tutorial and I have setup nginx and added two blocks for my two domains and created symlinks, but only one domain is working. First domain throws Server not found error in Firefox, and second loads perfectly.

I tried removing symlink for second domain and what happend is that the second domain is now loading first domains page, and first domain is still not working.

First server block looks like this:

server {
listen 80;
listen [::]:80;

```
root /var/www/firstnamelastname/html;

index index.html index.php index.htm index.nginx-debian.html;

server_name firstnamelastname.from.hr www.firstnamelastname.from.hr;

location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    try_files $uri $uri/ =404;
}
```

}

and second server block looks like this:

server {
listen 80;
listen [::]:80;

```
  root /var/www/named.technology/html;

  index index.html index.php index.htm index.nginx-debian.html;

  server_name named.technology;

  location / {
      # First attempt to serve request as file, then
      # as directory, then fall back to displaying a 404.
      try_files $uri $uri/ =404;
  }
```

}

Nginx error log is empty, firewall is set to allow Nginx Full, both domains point to ns1.digitalocean.com - ns3.digitalocean.com

I can't seem to find a problem

Can anyone help me troubleshoot this?

MoxieProxy *June 1, 2017*

0 Hi, are you still having this issue? You will find in Nginx's documentation in reference to server names that:

" They may be defined using exact names, wildcard names, or regular expressions:

server {
listen 80;
server_name example.org www.example.org;
...
} "

---

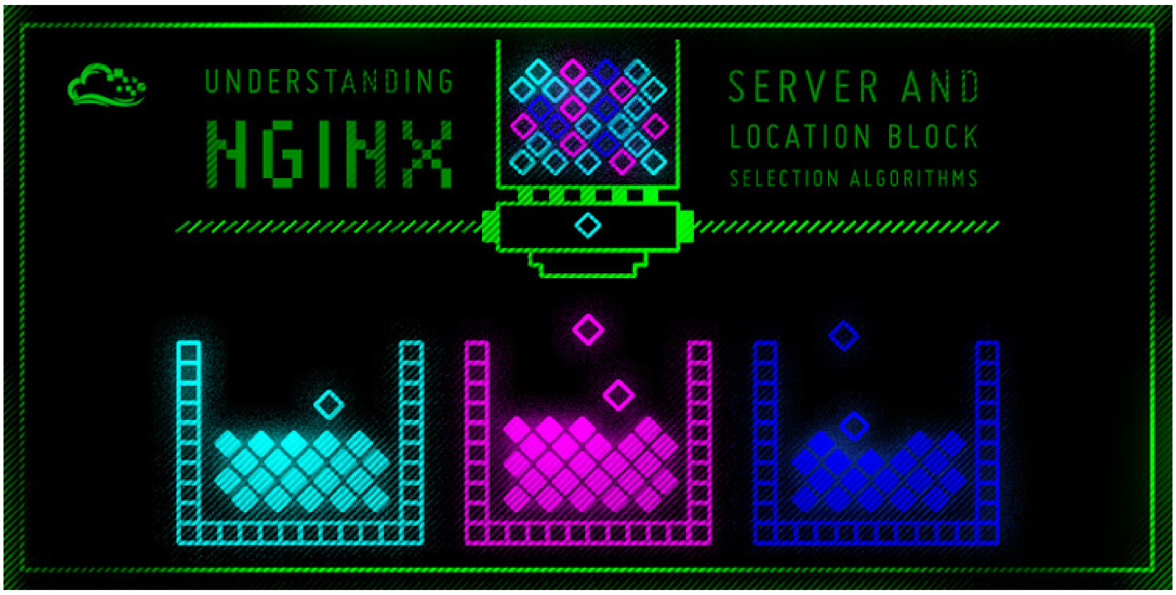
n8techy *June 17, 2017*

0 I am using ssl through LetsEncrypt on my website. I tried this tutorial to add another website, and when i go to the new domain it redirects me to my first domain that uses ssl. Any ideas?

jellingwood MOD *June 19, 2017*

0 @n8techy Hey there. Nginx uses a combination of the `listen` directive (which dictates which port the server block is configured to respond to), the `default_server` option for the `listen` directive (which defines one server block per port to use as a fallback), and the `server_name` directive (which designates the IP addresses or domain names that the server block is supposed to respond to). You can find a more detailed explanation here.

If you are having this issue, the first thing I'd do is make sure each of your server blocks uses the `server_name` directive to define exactly which domain it is supposed to serve requests for. While evaluating each of your server blocks, consider removing any `default_server` declarations from the `listen` directives you may find. If more than one server block uses the same `server_name` value and `listen` port, you will run into problems.



Understanding Nginx Server and Location Block Selection Algorithms

by Justin Ellingwood

Nginx is one of the most popular web servers in the world. In this guide, we will discuss how Nginx selects the server and location block that will handle a given client's request. We will go over the algorithm in place, as well as the directives and options you can use

n8techy *June 19, 2017*

0 I had my default server set in my first site. I have removed it and restarted nginx, but its still redirecting to the first domain. Here are my server blocks

```
#Primary Site

server {
        listen 80;
        listen [::]:80;
        server_name main.domain.net;
        return 301 https://$server_name$request_uri;
}

server {
        # SSL configuration
        #
        listen 4 default_server43 ssl http2;
        listen [::]:4 default_server43 ssl http2;
        include snippets/ssl-main.domain.net.conf;
        include snippets/ssl-params.conf;


## New Site

server {
        listen 80;
        listen [::]:80;
        server_name newsite.net;
        return 301 http://$server_name$request_uri;
#}

#server {
        # SSL configuration
        #
#       listen 443 ssl http2;
#       listen [::]:443 ssl http2;
#       include snippets/ssl-newsite.net.conf;
#       include snippets/ssl-params.conf;
```

---

jellingwood MOD *June 20, 2017*

0 @n8techy Hey there. There seem to be a few problems here, but I'm not sure if they're transcription errors. The second block (the one dealing with SSL for the first site) has a `default_server` in the middle of its port specification. You'll need to fix that to get this to run. If you're removing the `default_server` option, you should also make sure you set the `server_name` like you did with the regular port 80 block. It should look like this:

```
. . .
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name main.domain.net;
```

```
        include snippets/ssl-main.domain.net.conf;
        include snippets/ssl-params.conf;
    }
    . . .
```

Make sure you include the closing brace at the end there. It isn't present in the comment you posted.

Next, in your third server block (the port 80 block for the new site), you have a 301 redirect that's pointing back to itself. It's redirecting to `http://$server_name$request_uri`, which is the exact same block. I'm not sure if you set it up that way to help yourself debug, but that configuration will create a redirect loop (it'll try to look up the redirect, but it will always match the same block). You should either remove the `redirect` line, or uncomment the SSL block.

Either of those errors should have caused Nginx to refuse to restart. Are you restarting the the Nginx process in between tests?

First, check the syntax by typing:

```
$ sudo nginx -t
```

If that complains about anything, go back to your configuration file to try to find the problem. Once that test passes, you can restart Nginx by typing:

```
$ sudo systemctl restart nginx
```

Beyond that, it's a bit difficult to debug since it appears you have some syntax errors in the comment at least, if not on your server itself. Hopefully this gets you started in the right direction though.

---

0 n8techy *June 20, 2017*

I have been restarting and i can get this working on another vm only difference is i'm not using ssl. Also I have cleaned up the code, and restarted nginx with no issues, but it still redirects to my primary site. Let me know what you think.

```
##Primary site

server {
        listen 80;
        listen [::]:80;

        server_name primary-site.com;
}

server {
        # SSL configuration
        #
        listen 443 ssl http2;
        listen [::]:443 ssl http2;
        server_name primary-site.com;
        include snippets/ssl-primary-site.com.conf;
        include snippets/ssl-params.conf;

        root /var/www/html;

        # Add index.php to the list if you are using PHP
        index index.php index.html index.htm index.nginx-debian.htm

        location ~ /.well-known {
                allow all;
        }
...
}

## New Site
server {
        listen 80;
        listen [::]:80;

        server_name new-site.com;

        root /var/www/new-site.com;

        # Add index.php to the list if you are using PHP
        index index.html index.php index.htm index.nginx-debian.htm
...
}
```

jellingwood MOD *June 21, 2017*

0 @n8techy If you're still having problems, it may be that at some point during this process, you set up a 301 redirect from your new domain to your old domain. 301 redirects are cached for a very long time by most browsers because the specification says that they indicate a permanent move. Try accessing the server from a different browser or, better yet, a different computer to see if you are able to access your new domain. You will have to look up the instructions for how to clear

301 redirects in your original browser if you are able to access the new domain from a different client.

A 301 redirect usually causes browsers to not even make the original request the second time the redirect is processed. This means that if in the past the browser came across instructions to permanently redirect your new domain to your old domain, any requests for the new domain will be automatically translated to the old domain in the browser before reaching out to the server. This means that your log files might not indicate the redirect activity on subsequent requests. However, it's worth checking into anyways. You can find the Nginx logs at `/var/log/nginx/access.log` and `/var/log/nginx/error.log` to see if you can find any additional information to help you troubleshoot.

---

digitalsaurus *July 7, 2017*

0 Fantastic guide—thanks so much!!

Load More Comments