CONTENTS

RELATED

How To Install nginx on CentOS 6 with yum
View

Initial Server Setup with Ubuntu 12.04
View

// Tutorial //

How To Install Nginx on Ubuntu 20.04

Published on April 25, 2020 · Updated on January 5, 2022

Ubuntu Nginx Ubuntu 20.04

By Erin Glass
Senior Manager, DevEd

English

Not using Ubuntu 20.04?
Choose a different version or distribution.

Ubuntu 20.04

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. It is a lightweight choice that can be used as either a web server or reverse proxy.

In this guide, we'll discuss how to install Nginx on your Ubuntu 20.04 server, adjust the firewall, manage the Nginx process, and set up server blocks for hosting more than one domain from a single server.

Prerequisites

Before you begin this guide, you should have a regular, non-root user with sudo privileges configured on your server. You can learn how to configure a regular user account by following our Initial server setup guide for Ubuntu 20.04.

You will also optionally want to have registered a domain name before completing the last steps of this tutorial. To learn more about setting up a domain name with DigitalOcean, please refer to our Introduction to DigitalOcean DNS.

When you have an account available, log in as your non-root user to begin.

Step 1 – Installing Nginx

Because Nginx is available in Ubuntu's default repositories, it is possible to install it from these repositories using the `apt` packaging system.

Since this is our first interaction with the `apt` packaging system in this session, we will update our local package index so that we have access to the most recent package listings. Afterwards, we can install `nginx`:

```
$ sudo apt update
$ sudo apt install nginx
```

Copy

After accepting the procedure, `apt` will install Nginx and any required dependencies to your server.

Step 2 – Adjusting the Firewall

Popular Topics

Ubuntu
Linux Basics
JavaScript
React
Python
Security
MySQL
Docker
Kubernetes
Browse all topic tags

Free Managed Hosting →

All tutorials →

Questions

Q&A Forum
Ask a question
DigitalOcean Support

Before testing Nginx, the firewall software needs to be adjusted to allow access to the service. Nginx registers itself as a service with `ufw` upon installation, making it straightforward to allow Nginx access.

List the application configurations that `ufw` knows how to work with by typing:

```
$ sudo ufw app list
```
Copy

You should get a listing of the application profiles:

```
Output
Available applications:
  Nginx Full
  Nginx HTTP
  Nginx HTTPS
  OpenSSH
```

As demonstrated by the output, there are three profiles available for Nginx:

- **Nginx Full**: This profile opens both port 80 (normal, unencrypted web traffic) and port 443 (TLS/SSL encrypted traffic)
- **Nginx HTTP**: This profile opens only port 80 (normal, unencrypted web traffic)
- **Nginx HTTPS**: This profile opens only port 443 (TLS/SSL encrypted traffic)

It is recommended that you enable the most restrictive profile that will still allow the traffic you've configured. Right now, we will only need to allow traffic on port 80.

You can enable this by typing:

```
$ sudo ufw allow 'Nginx HTTP'
```
Copy

You can verify the change by typing:

```
$ sudo ufw status
```
Copy

The output will indicated which HTTP traffic is allowed:

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Nginx HTTP                 ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Nginx HTTP (v6)            ALLOW       Anywhere (v6)
```

Step 3 - Checking your Web Server

At the end of the installation process, Ubuntu 20.04 starts Nginx. The web server should already be up and running.

We can check with the `systemd` init system to make sure the service is running by typing:

```
$ systemctl status nginx
```
Copy

```
Output
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2020-04-20 16:08:19 UTC; 3 days ago
     Docs: man:nginx(8)
 Main PID: 2369 (nginx)
    Tasks: 2 (limit: 1153)
   Memory: 3.5M
   CGroup: /system.slice/nginx.service
           ├─2369 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           └─2380 nginx: worker process
```

As confirmed by this out, the service has started successfully. However, the best way to test this is to actually request a page from Nginx.

You can access the default Nginx landing page to confirm that the software is running properly by navigating to your server's IP address. If you do not know your server's IP address, you can find it by using the icanhazip.com tool, which will give you your public IP address as received from another location on the internet:

```
$ curl -4 icanhazip.com
```
Copy

When you have your server's IP address, enter it into your browser's address bar:

```
http://your_server_ip
```

You should receive the default Nginx landing page:

If you are on this page, your server is running correctly and is ready to be managed.

Step 4 – Managing the Nginx Process

Now that you have your web server up and running, let's review some basic management commands.

To stop your web server, type:

```
$ sudo systemctl stop nginx
```
Copy

To start the web server when it is stopped, type:

```
$ sudo systemctl start nginx
```
Copy

To stop and then start the service again, type:

```
$ sudo systemctl restart nginx
```
Copy

If you are only making configuration changes, Nginx can often reload without dropping connections. To do this, type:

```
$ sudo systemctl reload nginx
```
Copy

By default, Nginx is configured to start automatically when the server boots. If this is not what you want, you can disable this behavior by typing:

```
$ sudo systemctl disable nginx
```
Copy

To re-enable the service to start up at boot, you can type:

```
$ sudo systemctl enable nginx
```
Copy

You have now learned basic management commands and should be ready to configure the site to host more than one domain.

Step 5 – Setting Up Server Blocks (Recommended)

When using the Nginx web server, *server blocks* (similar to virtual hosts in Apache) can be used to encapsulate configuration details and host more than one domain from a single server. We will set up a domain called **your_domain**, but you should **replace this with your own domain name**.

Nginx on Ubuntu 20.04 has one server block enabled by default that is configured to serve documents out of a directory at `/var/www/html`. While this works well for a single site, it can become unwieldy if you are hosting multiple sites. Instead of modifying `/var/www/html`, let's create a directory structure within `/var/www` for our **your_domain** site, leaving `/var/www/html` in place as the default directory to be served if a client request doesn't match any other sites.

Create the directory for **your_domain** as follows, using the `-p` flag to create any necessary parent directories:

```
$ sudo mkdir -p /var/www/your_domain/html
```
Copy

Next, assign ownership of the directory with the `$USER` environment variable:

```
$ sudo chown -R $USER:$USER /var/www/your_domain/html
```
Copy

The permissions of your web roots should be correct if you haven't modified your `umask` value, which sets default file permissions. To ensure that your permissions are correct and allow the owner to read, write, and execute the files while granting only read and execute permissions to groups and others, you can input the following command:

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

Next, create a sample `index.html` page using `nano` or your favorite editor:

```
$ sudo nano /var/www/your_domain/html/index.html
```
Copy

Inside, add the following sample HTML:

/var/www/your_domain/html/index.html

```
<html>
    <head>
```
Copy

```
        <title>Welcome to your_domain!</title>
    </head>
    <body>
        <h1>Success!  The your_domain server block is working!</h1>
    </body>
</html>
```

Save and close the file by pressing `Ctrl+X` to exit, then when prompted to save, `Y` and then `Enter`.

In order for Nginx to serve this content, it's necessary to create a server block with the correct directives. Instead of modifying the default configuration file directly, let's make a new one at `/etc/nginx/sites-available/your_domain`:

```
$ sudo nano /etc/nginx/sites-available/your_domain
```
Copy

Paste in the following configuration block, which is similar to the default, but updated for our new directory and domain name:

/etc/nginx/sites-available/your_domain

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

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

        server_name your_domain www.your_domain;

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

Notice that we've updated the `root` configuration to our new directory, and the `server_name` to our domain name.

Next, let's enable the file by creating a link from it to the `sites-enabled` directory, which Nginx reads from during startup:

```
$ sudo ln -s /etc/nginx/sites-available/your_domain /etc/nginx/sites-enabled/
```
Copy

Note: Nginx uses a common practice called symbolic links, or symlinks, to track which of your server blocks are enabled. Creating a symlink is like creating a shortcut on disk, so that you could later delete the shortcut from the `sites-enabled` directory while keeping the server block in `sites-available` if you wanted to enable it.

Two server blocks are now enabled and configured to respond to requests based on their `listen` and `server_name` directives (you can read more about how Nginx processes these directives here):

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

To avoid a possible hash bucket memory problem that can arise from adding additional server names, it is necessary to adjust a single value in the `/etc/nginx/nginx.conf` file. Open the file:

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

Find the `server_names_hash_bucket_size` directive and remove the `#` symbol to uncomment the line. If you are using nano, you can quickly search for words in the file by pressing `CTRL` and `w`.

Note: Commenting out lines of code – usually by putting `#` at the start of a line – is another way of disabling them without needing to actually delete them. Many configuration files ship with multiple options commented out so that they can be enabled or disabled, by toggling them between active code and documentation.

/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
```
Copy

If there aren't any problems, restart Nginx to enable your changes:

```
$ sudo systemctl restart nginx
```
Copy

Nginx should now be serving your domain name. You can test this by navigating to

Nginx should now be serving your domain name. You can test this by navigating to `http://your_domain`, where you should see something like this:

Step 6 – Getting Familiar with Important Nginx Files and Directories

Now that you know how to manage the Nginx service itself, you should take a few minutes to familiarize yourself with a few important directories and files.

Content

- `/var/www/html`: The actual web content, which by default only consists of the default Nginx page you saw earlier, is served out of the `/var/www/html` directory. This can be changed by altering Nginx configuration files.

Server Configuration

- `/etc/nginx`: The Nginx configuration directory. All of the Nginx configuration files reside here.
- `/etc/nginx/nginx.conf`: The main Nginx configuration file. This can be modified to make changes to the Nginx global configuration.
- `/etc/nginx/sites-available/`: The directory where per-site server blocks can be stored. Nginx will not use the configuration files found in this directory unless they are linked to the `sites-enabled` directory. Typically, all server block configuration is done in this directory, and then enabled by linking to the other directory.
- `/etc/nginx/sites-enabled/`: The directory where enabled per-site server blocks are stored. Typically, these are created by linking to configuration files found in the `sites-available` directory.
- `/etc/nginx/snippets`: This directory contains configuration fragments that can be included elsewhere in the Nginx configuration. Potentially repeatable configuration segments are good candidates for refactoring into snippets.

Server Logs

- `/var/log/nginx/access.log`: Every request to your web server is recorded in this log file unless Nginx is configured to do otherwise.
- `/var/log/nginx/error.log`: Any Nginx errors will be recorded in this log.

Conclusion

Now that you have your web server installed, you have many options for the type of content to serve and the technologies you want to use to create a richer experience.

If you'd like to build out a more complete application stack, check out the article How To Install Linux, Nginx, MySQL, PHP (LEMP stack) on Ubuntu 20.04.

In order to set up HTTPS for your domain name with a free SSL certificate using *Let's Encrypt*, you should move on to How To Secure Nginx with Let's Encrypt on Ubuntu 20.04.

About the authors

Erin Glass Author

Senior Manager, DevEd

Open source advocate and lover of education, culture, and community.

Still looking for an answer? Ask a question Search for more help

Was this helpful? Yes No

Comments

10 Comments

Leave a comment...

This textbox defaults to using **Markdown** to format your answer.

You can type **!ref** in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

Sign In or Sign Up to Comment

kxvx • September 20, 2020

I just wanted to mention to all those who may be having problems - Be sure you allow SSL/port 443 connections in your firewall!

`sudo ufw allow https` -or- `sudo ufw allow 443`

Hours spent trying all kinds of different configurations, turns out the solution was as simple as adding this firewall rule!

Show replies Reply

louisd • October 27, 2021

Regarding the people who have had trouble:

Where the guide says “your_domain”, you really need to enter the FQDN to your server, not just your domain name.

For example, my server FQDN is “log.contoso.local” and that is what I can successfully ping, so that is what I must enter for all occurrences of “your_domain”.

Reply

hak33m16 • August 5, 2021

Doesn’t seem to work as of August 2021. Following the steps exactly (other than having to manually enable ufw with `sudo ufw enable`) and trying to visit `http://your_domain` (with `.com` obviously) or https results in an error in my browser reading:

```
Unable to connect

Firefox can’t establish a connection to the server at YOUR_DOMAIN
```

However, directly visiting the address of my server does work, so it must be an issue with the configuration of the server blocks section

I know my DNS records are configured correctly because I’m migrating from a functioning server on Ubuntu 16 to one using Ubuntu 20/this guide

Any ideas anyone? Is this assuming that I’m using a specific version of nginx?

Show replies Reply

cadecadeskywalker • May 4, 2021

A quick comment for those who may be frustrated like I was-- when it says to update your server_name value to 'your domain,' that value needs to include the closer. IE 'example.com' or 'example.net' or whatever. Makes a big difference!

Reply

brendang • December 1, 2020

when i try to run `sudo systemctl restart nginx` i recieve the following error:

```
Job for nginx.service failed because the control process exited with error code.
See "systemctl status nginx.service" and "journalctl -xe" for details.
```

running `journalctl -xe` it says:

```
pam_unix(sudo:auth): Couldn't open /etc/securetty: No such file or directory
```

I'm not sure what this means or what step i could have messed up in order to recieve this error message. Any help would be appreciated!

Show replies Reply

randyjohnson • May 18, 2020

Just an FYI, For me the UFW was not active. I had to run `sudo ufw enable` on Step 2.

Show replies Reply

Gobind Rana • December 8, 2022

I am using this multi time ...

Reply

Gobind Rana • December 8, 2022

Best Step Ever

Reply

ALI NEJADBAHRAM • December 6, 2022

Easy Peasy

peace of Cake

Thank you

I enjoyed

Reply

ktra99 • November 19, 2022

Article is not accurate I'm afraid. If you are new to digital ocean and are looking to work with nginx among other things. Take a turn and look at other alternatives. It will save you countless hours

Show replies Reply

Load More Comments

Congratulations on unlocking the whale ambience easter egg! Click the whale button in the bottom left of your screen to toggle some ambient whale noises while you read.

Thank you to the Glacier Bay National Park & Preserve and Merrick079 for the sounds behind this easter egg.

Interested in whales, protecting them, and their connection to helping prevent climate change? We recommend checking out the Whale and Dolphin Conservation.

Reset easter egg to be discovered again / Permanently dismiss and hide easter egg

GET OUR BIWEEKLY NEWSLETTER

Sign up for Infrastructure as a Newsletter.

HOLLIE'S HUB FOR GOOD

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

BECOME A CONTRIBUTOR

You get paid; we donate to tech nonprofits.

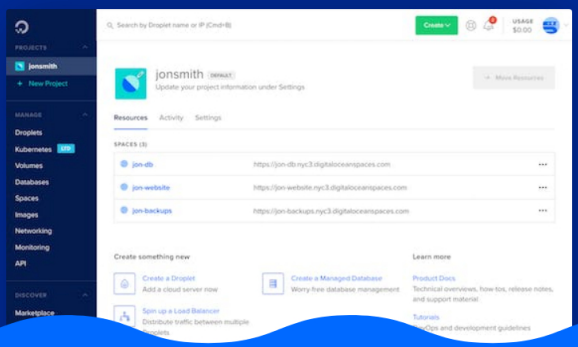
Featured on Community Kubernetes Course Learn Python 3 Machine Learning in Python Getting started with Go Intro to Kubernetes

DigitalOcean Products Virtual Machines Managed Databases Managed Kubernetes Block Storage Object Storage Marketplace VPC Load Balancers

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you're running one virtual machine or ten thousand.

Learn More



Company

About
Leadership
Blog
Careers
Customers
Partners
Channel Partners
Referral Program
Affiliate Program
Press
Legal
Security
Investor Relations
DO Impact

Products

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

Community

Tutorials
Q&A
CSS-Tricks
Write for DOnations
Currents Research
Hatch Startup Program
deploy by DigitalOcean
Shop Swag
Research Program
Open Source
Code of Conduct
Newsletter Signup
Meetups

Solutions

Website Hosting
VPS Hosting
Web & Mobile Apps
Game Development
Streaming
VPN
SaaS Platforms
Cloud Hosting for Blockchain
Startup Resources

Contact

Support
Sales
Report Abuse
System Status
Share your ideas