

How To Install Nginx on Ubuntu 18.04 [Quickstart]

Posted July 23, 2018 33.1k

NGINX

QUICKSTART

UBUNTU

UBUNTU 18.04



By: Kathleen Juell

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 more resource-friendly than Apache in most cases and can be used as a web server or reverse proxy.

In this guide, we'll explain how to install Nginx on your Ubuntu 18.04 server. For a more detailed version of this tutorial, please refer to [How To Install Nginx on Ubuntu 18.04](#).

Prerequisites

Before you begin this guide, you should have the following:

- An Ubuntu 18.04 server and a regular, non-root user with sudo privileges. Additionally, you will need to enable a basic firewall to block non-essential ports. You can learn how to configure a regular user account and set up a firewall by following our [initial server setup guide for Ubuntu 18.04](#).

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, you can install it using the `apt` packaging system.

Update your local package index:

```
$ sudo apt update
```

Install Nginx:

```
$ sudo apt install nginx
```

Step 2 – Adjusting the Firewall

Check the available `ufw` application profiles:

```
$ sudo ufw app list
```

Output

Available applications:

```
  Nginx Full
  Nginx HTTP
  Nginx HTTPS
  OpenSSH
```

Let's enable the most restrictive profile that will still allow the traffic you've configured, permitting traffic on port `80`:

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

Verify the change:

```
$ sudo ufw status
```

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

Check with the `systemd` init system to make sure the service is running by typing:

```
$ systemctl status nginx
```

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 2018-04-20 16:08:19 UTC; 3 days ago

Docs: `man:nginx(8)`

Main PID: 2369 (`nginx`)

Tasks: 2 (limit: 1153)

CGroup: `/system.slice/nginx.service`

└─2369 `nginx`: master process `/usr/sbin/nginx -g daemon on; master_process on;`

└─2380 `nginx`: worker process

Access the default Nginx landing page to confirm that the software is running properly through your IP address:

`http://your_server_ip`

You should see the default Nginx landing page:

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.

Step 4 – Setting Up Server Blocks (Recommended)

When using the Nginx web server, you can use *server blocks* (similar to virtual hosts in Apache) to encapsulate configuration details and host more than one domain from a single server. We will set up a domain called **example.com**, but you should **replace this with your own domain name**. To learn more about setting up a domain name with DigitalOcean, see our [introduction to DigitalOcean DNS](#).

Create the directory for **example.com**, using the `-p` flag to create any necessary parent directories:

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

Assign ownership of the directory:

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

The permissions of your web roots should be correct if you haven't modified your `umask` value, but you can make sure by typing:

```
$ sudo chmod -R 755 /var/www/example.com
```

Create a sample `index.html` page using `nano` or your favorite editor:

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

Inside, add the following sample HTML:

```
                                /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.

Make a new server block at `/etc/nginx/sites-available/example.com`:

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

Paste in the following configuration block, updated for our new directory and domain name:

```
                                /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;
    }
}
```

Save and close the file when you are finished.

Enable the file by creating a link from it to the `sites-enabled` directory:

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

Two server blocks are now enabled and configured to respond to requests based on their `listen` and `server_name` directives:

- `example.com`: Will respond to requests for `example.com` and `www.example.com`.
- `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
```

Find the `server_names_hash_bucket_size` directive and remove the `#` symbol to uncomment the line:

```
                                /etc/nginx/nginx.conf

...
http {
    ...
    server_names_hash_bucket_size 64;
    ...
}
...
```

Test for syntax errors:

```
$ sudo nginx -t
```

Restart Nginx to enable your changes:

```
$ sudo systemctl restart nginx
```

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


Success! The example.com server block is working!

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 this article on [how to configure a LEMP stack on Ubuntu 18.04.](#)

By: Kathleen Juell

 Upvote (16)  Subscribe  Share



We just made it easier for you to deploy faster.

[TRY FREE](#)

Related Tutorials

[How To Deploy a PHP Application with Kubernetes on Ubuntu 16.04](#)

[How To Ensure Code Quality with SonarQube on Ubuntu 18.04](#)

[How To Set Up a Private Docker Registry on Ubuntu 18.04](#)

[How to Set Up an Nginx Ingress with Cert-Manager on DigitalOcean Kubernetes](#)

[How To Install Elasticsearch, Logstash, and Kibana \(Elastic Stack\) on CentOS 7](#)

0 Comments

Leave a comment...

[Log In to Comment](#)



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2019 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Write for DOnations](#) [Shop](#)