CONTENTS

// Tutorial //

Initial Server Setup with Ubuntu 20.04

Published on April 23, 2020 · Updated on August 9, 2021

Ubuntu Security Ubuntu 20.04 Getting Started Initial Server Setup DigitalOcean Droplets

By Brian Boucheron

English

Introduction

When you first create a new Ubuntu 20.04 server, you should perform some important configuration steps as part of the initial setup. These steps will increase the security and usability of your server, and will give you a solid foundation for subsequent actions.

When you create a DigitalOcean Droplet, you can choose an Ubuntu version that will be added to your new Droplet automatically. Simplify your setup with our out-of-the-box solutions.

Step 1 – Logging in as root

To log into your server, you will need to know your **server's public IP address**. You will also need the password or — if you installed an SSH key for authentication — the private key for the **root** user's account. If you have not already logged into your server, you may want to follow our guide on how to Connect to Droplets with SSH, which covers this process in detail.

If you are not already connected to your server, log in now as the **root** user using the following command (substitute the highlighted portion of the command with your server's public IP address):

```
$ ssh root@your_server_ip
```

Copy

Accept the warning about host authenticity if it appears. If you are using password authentication, provide your **root** password to log in. If you are using an SSH key that is passphrase protected, you may be prompted to enter the passphrase the first time you use the key each session. If this is your first time logging into the server with a password, you may also be prompted to change the **root** password.

About root

The **root** user is the administrative user in a Linux environment that has very broad privileges. Because of the heightened privileges of the **root** account, you are *discouraged* from using it on a regular basis. This is because the **root** account is able to make very destructive changes, even by accident.

The next step is setting up a new user account with reduced privileges for day-to-day use. Later, we'll show you how to temporarily gain increased privileges for the times when you need them.

Step 2 – Creating a New User

Once you are logged in as **root**, you'll be able to add the new user account. In the future, we'll log in with this new account instead of **root**.

This example creates a new user called **sammy**, but you should replace that with a username that you like:

```
# adduser sammy
```
Copy

You will be asked a few questions, starting with the account password.

Enter a strong password and, optionally, fill in any of the additional information if you would like. This is not required and you can just hit `ENTER` in any field you wish to skip.

Step 3 – Granting Administrative Privileges

Now we have a new user account with regular account privileges. However, we may sometimes need to do administrative tasks.

To avoid having to log out of our normal user and log back in as the **root** account, we can set up what is known as *superuser* or **root** privileges for our normal account. This will allow our normal user to run commands with administrative privileges by putting the word `sudo` before the command.

To add these privileges to our new user, we need to add the user to the **sudo** group. By default, on Ubuntu 20.04, users who are members of the **sudo** group are allowed to use the `sudo` command.

As **root**, run this command to add your new user to the **sudo** group (substitute the highlighted username with your new user):

```
# usermod -aG sudo sammy
```
Copy

Now, when logged in as your regular user, you can type `sudo` before commands to run them with superuser privileges.

Step 4 – Setting Up a Basic Firewall

Ubuntu 20.04 servers can use the UFW firewall to make sure only connections to certain services are allowed. We can set up a basic firewall using this application.

Note: If your servers are running on DigitalOcean, you can optionally use DigitalOcean Cloud Firewalls instead of the UFW firewall. We recommend using only one firewall at a time to avoid conflicting rules that may be difficult to debug.

Applications can register their profiles with UFW upon installation. These profiles allow UFW to manage these applications by name. OpenSSH, the service allowing us to connect to our server now, has a profile registered with UFW.

You can see this by typing:

```
# ufw app list
```
Copy

```
Output
Available applications:
  OpenSSH
```

We need to make sure that the firewall allows SSH connections so that we can log back in next time. We can allow these connections by typing:

```
# ufw allow OpenSSH
```
Copy

Afterwards, we can enable the firewall by typing:

```
# ufw enable
```
Copy

Type `y` and press `ENTER` to proceed. You can see that SSH connections are still allowed by typing:

```
# ufw status
```
Copy

```
Output
Status: active
```

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

As **the firewall is currently blocking all connections except for SSH**, if you install and configure additional services, you will need to adjust the firewall settings to allow traffic in. You can learn some common UFW operations in our *UFW Essentials* guide.

Step 5 – Enabling External Access for Your Regular User

Now that we have a regular user for daily use, we need to make sure we can SSH into the account directly.

> **Note:** Until verifying that you can log in and use `sudo` with your new user, we recommend staying logged in as **root**. This way, if you have problems, you can troubleshoot and make any necessary changes as **root**. If you are using a DigitalOcean Droplet and experience problems with your **root** SSH connection, you can regain access to Droplets using the Recovery Console.

The process for configuring SSH access for your new user depends on whether your server's **root** account uses a password or SSH keys for authentication.

If the root Account Uses Password Authentication

If you logged in to your **root** account *using a password*, then password authentication is *enabled* for SSH. You can SSH to your new user account by opening up a new terminal session and using SSH with your new username:

```
$ ssh sammy@your_server_ip
```
Copy

After entering your regular user's password, you will be logged in. Remember, if you need to run a command with administrative privileges, type `sudo` before it like this:

```
$ sudo command_to_run
```
Copy

You will be prompted for your regular user password when using `sudo` for the first time each session (and periodically afterwards).

To enhance your server's security, **we strongly recommend setting up SSH keys instead of using password authentication**. Follow our guide on setting up SSH keys on Ubuntu 20.04 to learn how to configure key-based authentication.

If the root Account Uses SSH Key Authentication

If you logged in to your **root** account *using SSH keys*, then password authentication is *disabled* for SSH. You will need to add a copy of your local public key to the new user's `~/.ssh/authorized_keys` file to log in successfully.

Since your public key is already in the **root** account's `~/.ssh/authorized_keys` file on the server, we can copy that file and directory structure to our new user account in our existing session.

The simplest way to copy the files with the correct ownership and permissions is with the `rsync` command. This will copy the **root** user's `.ssh` directory, preserve the permissions, and modify the file owners, all in a single command. Make sure to change the highlighted portions of the command below to match your regular user's name:

> **Note:** The `rsync` command treats sources and destinations that end with a trailing slash differently than those without a trailing slash. When using `rsync` below, be sure that the source directory (`~/.ssh`) **does not** include a trailing slash (check to make sure you are not using `~/.ssh/`).
>
> If you accidentally add a trailing slash to the command, `rsync` will copy the *contents* of the **root** account's `~/.ssh` directory to the `sudo` user's home directory instead of copying the entire `~/.ssh` directory structure. The files will be in the wrong location and SSH will not be able to find and use them.

```
# rsync --archive --chown=sammy:sammy ~/.ssh /home/sammy
```
Copy

Now, open up a new terminal session on your local machine, and use SSH with your new username:

```
$ ssh sammy@your_server_ip
```

Copy

You should be logged in to the new user account without using a password. Remember, if you need to run a command with administrative privileges, type `sudo` before it like this:

```
$ sudo command_to_run
```

Copy

You will be prompted for your regular user password when using `sudo` for the first time each session (and periodically afterwards).

Where To Go From Here?

At this point, you have a solid foundation for your server. You can install any of the software you need on your server now.

About the authors

Brian Boucheron Author

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

jasonheecs • April 24, 2020

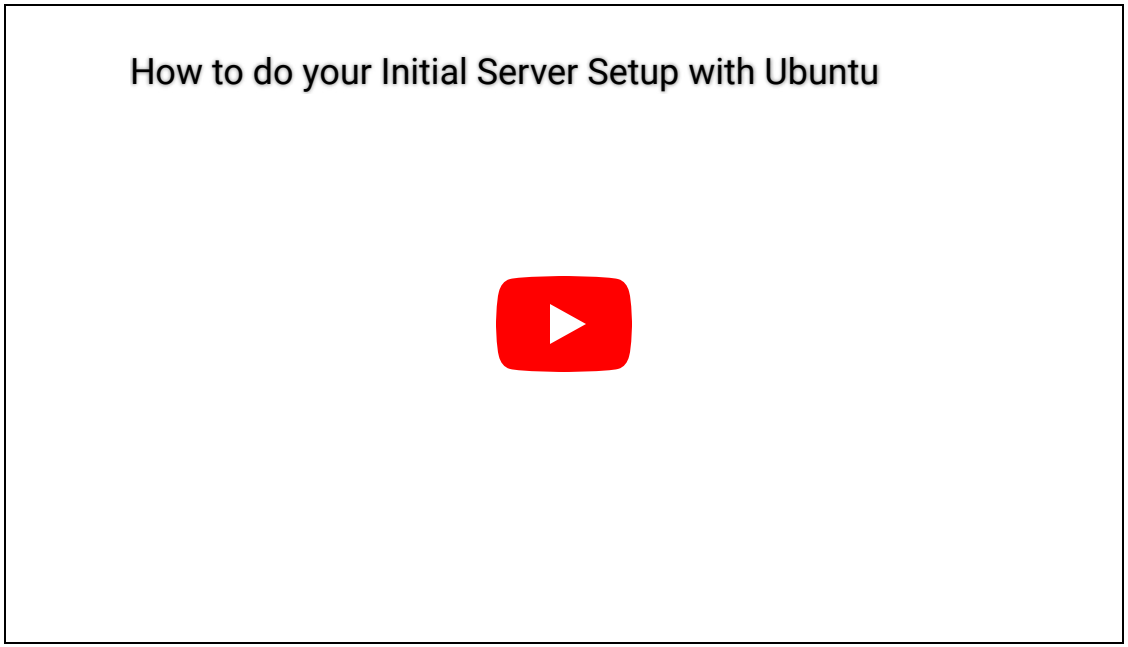
I have made a bash script to automate the setup process, hopefully this will be useful to someone else.

Show replies Reply

Bobby Iliev • October 19, 2020

Hello,

For anyone interested, I just created a similar video demo on how to do the initial server setup as described in this tutorial:



Hope that this helps!

Regards, Bobby

Show replies Reply

07de5ac34b8348feb5f5996072b1f7 • September 9, 2021

Much of the firewall was already preconfigured after deploying a Wordpress image from the DO marketplace, but the rest was golden. Thanks a lot!

Reply

Hector Lopez Monroy • April 30, 2020

The last command as root should be:

```
$ rsync --archive --chown=sammy:sammy /root/.ssh/ /home/sammy/.ssh/
```

Otherwise the `authorized_keys` file is created in the user's home folder.

Show replies Reply

saisonxiang • August 20, 2023

This website taught me how to enable my MacOS to SSH tunnel into my Linux terminal prompt without installing non-native apps or extensions. The instructions were clear and up to date. It took a long time to figure out.

Reply

spicymoodles • March 8, 2023

The end note in Step #3 is misleading: “Now, when logged in as your regular user, you can type `sudo` before commands to run them with superuser privileges.”

For a first timer creating SSH keys this threw me off. I tried logging in as USER but obviously it didn’t work. It’s only later at step #5 where it’s explained how to copy the public key from ROOT to USER too.

Should change “NOW…” to LATER :)

Otherwise everything else worked fine thank you

Reply

Alexandr Dubinin • March 28, 2022

If you faced with `Permission denied (publickey)`, there few tips that help you in troubleshooting:

You may point directly to ssh key that you want use by `-i` parameter

```
ssh user_user@user_server_ip -i ~/.ssh/key_name.pub
```

For logs on client side use `-v` flag

```
ssh user_user@user_server_ip -v
```

Logs on server side

```
cat /var/log/auth.log
```

Reply

LargeNavyCrab • March 14, 2022

Possible troubleshooting for people having “Permission denied(publickey)” issues. Try this:

Check if you have .ssh folder in your root and new user folder:

- a. enter “cd” (← return to root folder)
- b. enter “cd .ssh”

If not exist, then you’ve made some mistake adding publickey in your droplet. Try again re-creating your droplet.

- c. enter “cd” (← return to root folder)
- d. enter “cd home/sammy” (← use your user name here)
- e. enter “cd .ssh”

If not exist, probably rsync command did not end well. Instead of:

“rsync --archive --chown=sammy:sammy ~/.ssh /home/sammy”

try substitute “–archive” with “-a”.

Then, repeat steps from “.c” to “.e” and check again if .ssh folder has been created.

If folder still not exist, try having a lecture here, it may help.

Resolve this first, then check again logging in with your new user.

If you’re still having issues:

- a. go here

- b. repeat the process following OpenSSH or PutTy steps but, instead of create another “root” user (people using PutTy: field “Auto-login username”), use the user name you create before (eg: “sammy”).

Save your new session and try open it.

Hope it helps.

Reply

Jimmy Olano • January 8, 2022

THX Mr. Boucheron!

I tested it on a brand new droplet with Ubuntu 21.10, DO Bangalore data center.

```
jimmy@ubuntu-s-1vcpu-1gb-blr1-01:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 21.10"
NAME="Ubuntu"
VERSION_ID="21.10"
VERSION="21.10 (Impish Indri)"
VERSION_CODENAME=impish
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=impish
jimmy@ubuntu-s-1vcpu-1gb-blr1-01:~$
```

All is perfect.

Reply

jetspace100 • December 26, 2021

I'm setting up a Ubuntu machine as per your tutorial

You are suggesting that the out of the box root account should be disabled and a new user with sudo access be created.

When installing applications (node, mongo, nginx etc) should these apps be installed using the new account ? And in which context should these apps run ?

Could you please explain a bit more on this.

Thanks

Reply

Load More Comments

Popular Topics

Ubuntu

Linux Basics

JavaScript

Python

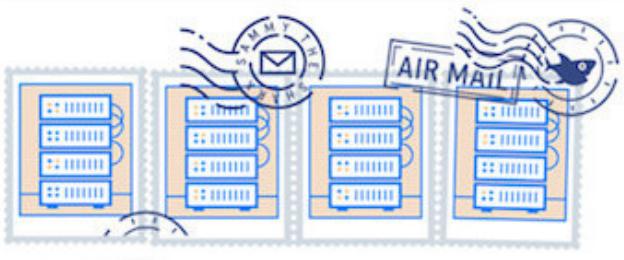
MySQL

Docker

Kubernetes

All tutorials →

Talk to an expert →



Get our biweekly newsletter

Sign up for Infrastructure as a Newsletter.

Sign up →

Hollie's Hub for Good

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

Learn more →

Become a contributor

You get paid; we donate to tech nonprofits.

Learn more →

Featured on Community

Kubernetes Course | Learn Python 3 | Machine Learning in Python | Getting started with Go | Intro to Kubernetes

DigitalOcean Products

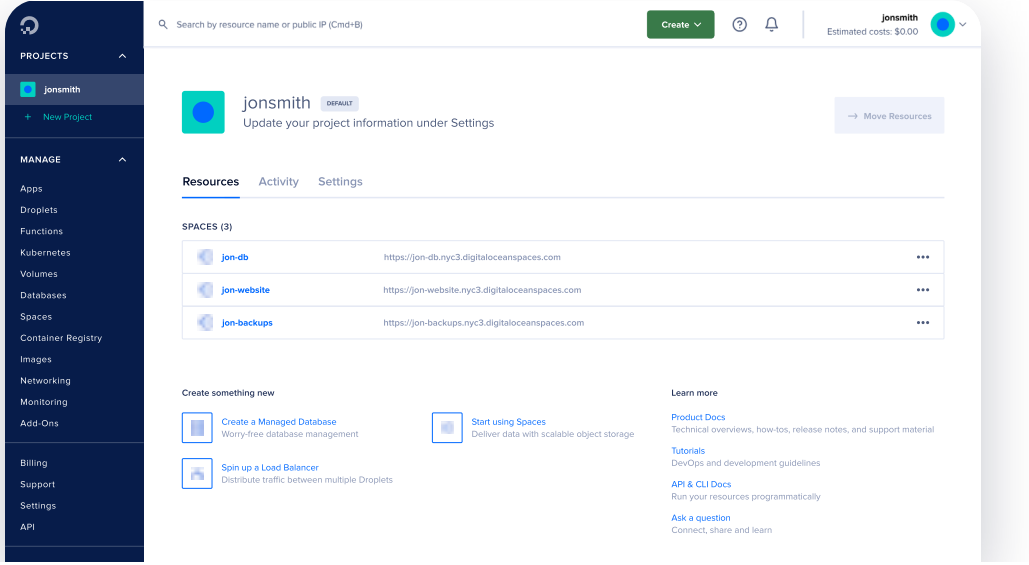
Cloudways | Virtual Machines | Managed Databases | Managed Kubernetes | Block Storage | Object Storage

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 →



Get started for free

Enter your email to get $200 in credit for your first 60 days with DigitalOcean.

Email address Send My Promo

New accounts only. By submitting your email you agree to our Privacy Policy.

Company

About

Leadership

Products

Products Overview

Droplets

Community

Tutorials

Q&A

Solutions

Website Hosting

VPS Hosting

Blog
Careers
Customers
Partners
Channel Partners
Referral Program
Affiliate Program
Press
Legal
Privacy Policy
Security
Investor Relations
DO Impact
Nonprofits

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

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

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