





Introduction

When you first create a new Ubuntu 18.04 server, there are a few configuration steps that you should take early on as part of the basic setup. This will increase the security and usability of your server and will give you a solid foundation for subsequent actions.

Note: The guide below demonstrates how to manually complete the steps we recommend for new Ubuntu 18.04 servers. Following this procedure manually can be useful to learn some basic system administration skills and as an exercise to fully understand the actions being taken on your server. As an alternative, if you wish to get up and running more quickly, you can <u>run our initial</u> server setup script which automates these steps.

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 your Droplet with SSH, which covers this process in detail.

If you are not already connected to your server, go ahead and log in 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
```

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 part of the power inherent with the **root** account is the ability to make very destructive changes, even by accident.

The next step is to set up an alternative user account with a reduced scope of influence for day-to-day work. We'll teach you how to gain increased privileges during the times when you need them.

Step 2 — Creating a New User

Once you are logged in as **root**, we're prepared to add the new user account that we will use to log in from now on.

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

```
# adduser sammy
```

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 each command.

To add these privileges to our new user, we need to add the new user to the **sudo** group. By default, on Ubuntu 18.04, users who belong to 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 word with your new user):

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

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

Step 4 — Setting Up a Basic Firewall

Ubuntu 18.04 servers can use the UFW firewall to make sure only connections to certain services are allowed. We can set up a basic firewall very easily using this application.

Note: If your servers are running on DigitalOcean, you can optionally use <u>DigitalOcean Cloud</u>

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.

Different 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
```

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

Afterwards, we can enable the firewall by typing:

ufw enable

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

ufw status

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 acceptable traffic in. You can learn some common UFW operations in this 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 log into the Droplet using the DigitalOcean 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
```

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
```

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 18.04 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
```

Now, open up a new terminal session and using SSH with your new username:

```
$ ssh sammy@your_server_ip
```

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
```

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.

\bigcirc	Upvote (103)	
------------	--------------	--

^{-†} Subscribe

Introducing Projects on DigitalOcean

Organize your resources according to how you work.

READ MORE

How to Get Started with FreeBSD

How To Set Up an OpenVPN Server on Debian 9

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

How To Create a Self-Signed SSL Certificate for Apache in Debian 9

How To Set Up vsftpd for a User's Directory on Debian 9

10 Comment	S			
Leave a comment				

Log In to Comment

- ____jasonheecs May 3, 2018
- I have made a bash script to automate the setup process, hopefully this will be useful to someone else.
- henforshey May 11, 2018
- 5 Hey, Justin! Thanks for the great tutorial. In your last step, rsync --archive -chown=sammy:sammy ~/.ssh /home/sammy, it might be worth pointing out that there should be
 no trailing slash on ~/.ssh. I only mention that because when I was typing it in, I hit <tab> (to
 autocomplete, out of habit) and bash turned that into /root/ssh/.

I tried to log in with my new user, but because rsync doesn't copy the source directory itself when it is appended with a trailing slash, my **authorized_keys** file was just hanging out in my new user's home directory. A quick note might save someone else the trouble. Thanks again!

ô	<u>@benforshey</u> Yeah, that's a good call. Thar	ks for the	input! I've	added a n	ote to h	nelp ι	ısers
1	avoid that problem in the future.						

- ^ ajbozdar June 5, 2018
- o This saved my day. Thank you very much.
- odtrtest July 31, 2018
- ^o Hi..i followed this steps but I still cannot SSH login, i remove the trailing slash followed this rsync but still error..what am i doing wrong? please guide...thanks (rsync --archive --chown=sammy:sammy .ssh /home/sammy). with trailing slash and without trailing slash still cannot login with SSH.
 - ^ benforshey August 2, 2018
 - Hey, @odtrtest! Are you replacing sammy with your own username?



- ^ avanpoppelen June 3, 2018
- 1 It's also probably advisable to include a Step 6: Disable SSH root login.

This can be done by setting PermitRootLogin to no in /etc/ssh/sshd_config.



- minaz June 5, 2018
- ² I also prefer to install fail2ban to prevent anybody trying to hack into the system.

apt-get install fail2ban

- ^ infoac253e1b43f September 5, 2018
- The old 16-04 tutorial made a lot more sense with how to get the keys in the correct place for ssh on the new user.
- ^ storemalt September 17, 2018
- You might not be able to access using the new user you created, make sure you allow OpenSSH on uwf by (logged in as root or access it via the droplet console):

ufw allow OpenSSH

in my case i also allowed port 22

ufw allow 22



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



Copyright © 2018 DigitalOcean™ Inc.

Community Tutorials Questions Projects Tags Newsletter RSS

Distros & One-Click Apps Terms, Privacy, & Copyright Security Report a Bug Write for DOnations Shop