# Initial Server Setup with Ubuntu 16.04

Posted April 21, 2016 1.9m GETTING STARTED INITIAL SERVER SETUP SECURITY UBUNTU UBUNTU 16.04

710

By: Mitchell Anicas

Not using **Ubuntu 16.04**? Choose a different version:

## Introduction

When you first create a new Ubuntu 16.04 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.

## Step One — Root Login

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 the first tutorial in this series, 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 word with your server's public IP address):

```
$ ssh root@your_server_ip
```

Complete the login process by accepting the warning about host authenticity, if it appears, then providing your root authentication (password or private key). If it is your first time logging into the server with a password, you will 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 actually *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 Two — Create 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 Three — Root 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 16.04, 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 your user can run commands with superuser privileges! For more information about how this works, check out this sudoers tutorial.

If you want to increase the security of your server, follow the rest of the steps in this tutorial.

## Step Four — Add Public Key Authentication (Recommended)

The next step in securing your server is to set up public key authentication for your new user. Setting this up will increase the security of your server by requiring a private SSH key to log in.

### Generate a Key Pair

If you do not already have an SSH key pair, which consists of a public and private key, you need to generate one. If you already have a key that you want to use, skip to the *Copy the Public Key* step.

To generate a new key pair, enter the following command at the terminal of your **local machine** (ie. your computer):

```
$ ssh-keygen
```

Assuming your local user is called "localuser", you will see output that looks like the following:

```
ssh-keygen output
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/localuser/.ssh/id_rsa):
```

Hit return to accept this file name and path (or enter a new name).

Next, you will be prompted for a passphrase to secure the key with. You may either enter a passphrase or leave the passphrase blank.

**Note:** If you leave the passphrase blank, you will be able to use the private key for authentication without entering a passphrase. If you enter a passphrase, you will need both the private key *and* the passphrase to log in. Securing your keys with passphrases is more secure, but both methods have their uses and are more secure than basic password authentication.

This generates a private key, `id_rsa`, and a public key, `id_rsa.pub`, in the `.ssh` directory of the *localuser*'s home directory. Remember that the private key should not be shared with anyone who should not have access to your servers!

## Copy the Public Key

After generating an SSH key pair, you will want to copy your public key to your new server. We will cover two easy ways to do this.

**Note**: The `ssh-copy-id` method will not work on DigitalOcean if an SSH key was selected during Droplet creation. This is because DigitalOcean disables password authentication if an SSH key is present, and the `ssh-copy-id` relies on password authentication to copy the key.

If you are using DigitalOcean and selected an SSH key during Droplet creation, use option 2 instead.

### Option 1: Use ssh-copy-id

If your local machine has the `ssh-copy-id` script installed, you can use it to install your public key to any user that you have login credentials for.

Run the `ssh-copy-id` script by specifying the user and IP address of the server that you want to install the key on, like this:

```
$ ssh-copy-id sammy@your_server_ip
```

After providing your password at the prompt, your public key will be added to the remote user's `.ssh/authorized_keys` file. The corresponding private key can now be used to log into the server.

### Option 2: Manually Install the Key

Assuming you generated an SSH key pair using the previous step, use the following command at the terminal of your **local machine** to print your public key (`id_rsa.pub`):

```
$ cat ~/.ssh/id_rsa.pub
```

This should print your public SSH key, which should look something like the following:

```
id_rsa.pub contents
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDBGTO0tsVejssuaYR5R3Y/i73SppJAhme1dH7W2c47d4gOqB4izP0+fRLfvbz/t
```

Select the public key, and copy it to your clipboard.

To enable the use of SSH key to authenticate as the new remote user, you must add the public key to a special file in the user's home directory.

**On the server**, as the **root** user, enter the following command to temporarily switch to the new user (substitute your own user name):

```
# su - sammy
```

Now you will be in your new user's home directory.

Create a new directory called `.ssh` and restrict its permissions with the following commands:

```
$ mkdir ~/.ssh
$ chmod 700 ~/.ssh
```

Now open a file in `.ssh` called `authorized_keys` with a text editor. We will use `nano` to edit the file:

```
$ nano ~/.ssh/authorized_keys
```

Now insert your public key (which should be in your clipboard) by pasting it into the editor.

Hit `CTRL-x` to exit the file, then `y` to save the changes that you made, then `ENTER` to confirm the file name.

Now restrict the permissions of the *authorized_keys* file with this command:

```
$ chmod 600 ~/.ssh/authorized_keys
```

Type this command **once** to return to the `root` user:

```
$ exit
```

Now your public key is installed, and you can use SSH keys to log in as your user.

To read more about how key authentication works, read this tutorial: How To Configure SSH Key-Based Authentication on a Linux Server.

Next, we'll show you how to increase your server's security by disabling password authentication.

## Step Five — Disable Password Authentication (Recommended)

Now that your new user can use SSH keys to log in, you can increase your server's security by disabling password-only authentication. Doing so will restrict SSH access to your server to public key authentication only. That is, the only way to log in to your server (aside from the console) is to possess the private key that pairs with the public key that was installed.

> **Note:** Only disable password authentication if you installed a public key to your user as recommended in the previous section, step four. Otherwise, you will lock yourself out of your server!

To disable password authentication on your server, follow these steps.

As **root** or **your new sudo user**, open the SSH daemon configuration:

```
$ sudo nano /etc/ssh/sshd_config
```

Find the line that specifies `PasswordAuthentication`, uncomment it by deleting the preceding `#`, then change its value to "no". It should look like this after you have made the change:

sshd_config — Disable password authentication

```
PasswordAuthentication no
```

Here are two other settings that are important for key-only authentication and are set by default. If you haven't modified this file before, you *do not* need to change these settings:

sshd_config — Important defaults

```
PubkeyAuthentication yes
ChallengeResponseAuthentication no
```

When you are finished making your changes, save and close the file using the method we went over earlier (`CTRL-X`, then `Y`, then `ENTER`).

Type this to reload the SSH daemon:

```
$ sudo systemctl reload sshd
```

Password authentication is now disabled. Your server is now only accessible with SSH key authentication.

## Step Six — Test Log In

Now, before you log out of the server, you should test your new configuration. Do not disconnect until you confirm that you can successfully log in via SSH.

In a new terminal on your **local machine**, log in to your server using the new account that we created. To do so, use this command (substitute your username and server IP address):

```
$ ssh sammy@your_server_ip
```

If you added public key authentication to your user, as described in steps four and five, your private key will be used as authentication. Otherwise, you will be prompted for your user's password.

> **Note about key authentication:** If you created your key pair with a passphrase, you will be prompted to enter the passphrase for your key. Otherwise, if your key pair is passphrase-less, you should be logged in to your server without a password.

Once authentication is provided to the server, you will be logged in as your new user.

Remember, if you need to run a command with root privileges, type "sudo" before it like this:

```
$ sudo command_to_run
```

## Step Seven — Set Up a Basic Firewall

Ubuntu 16.04 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.

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:

```
$ sudo 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:

```
$ sudo ufw allow OpenSSH
```

Afterwards, we can enable the firewall by typing:

```
$ sudo ufw enable
```

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

```
$ sudo ufw status
```

```
Output
Status: active

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

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.

## 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.

By: Mitchell Anicas

### Related Tutorials

How to Get Started with FreeBSD

Initial Server Setup with Debian 9

How to Set Up SSH Keys on Debian 9

How to Set Up SSH Keys on Ubuntu 18.04

Automating Initial Server Setup with Ubuntu 18.04

## 109 Comments

Leave a comment...

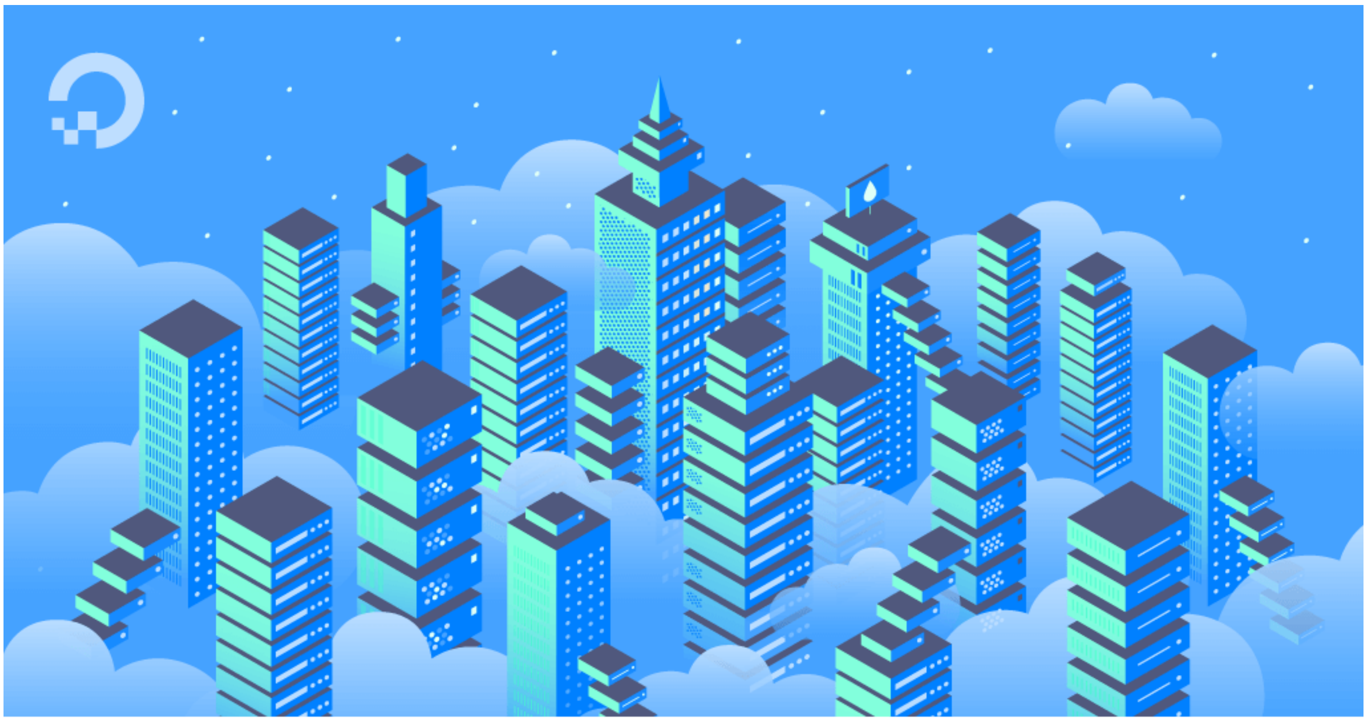
Log In to Comment

5 hederaleaf *April 22, 2016*

Hello!

Is there any particular reason that this guide doesn't recommend the installation of `ntp` and the configuration of swap space the way that the Initial Server Setup with Ubuntu 14.04 guide did?

Thanks!



**Initial Server Setup with Ubuntu 14.04**
by Justin Ellingwood

When you start a new server, there are a few steps that you should take every time to add some basic security and set a solid foundation. In this guide, we'll walk you through the basic steps necessary to hit the ground running with Ubuntu 14.04.

---

AnandKumar *April 25, 2016*

0 I don't find any reason not to use `ntp` or `swap`. Either the writer has forgotten OR (most likely) there will be another article for those things.

---

iamkingsleyf *April 25, 2016*

2 DO no longer recommend creating SWAP, that it wears the machine

---

zeokat *April 17, 2017*

0 Why DO not recommend use SWAP? If you have a Droplet with 512MB of RAM it can be usefull or am i wrong?

---

fernandopimenta *April 18, 2017*

1 Hi @zeokat,

SSDs are 20 to 25x slower than RAM.

When the system needs to use swap intensively, it will work slower.

Best,

Fernando Pimenta

---

sandwich *September 9, 2016*

0 I was wondering the same thing.

---

anaggh *May 3, 2016*

3 Do not change the default SSH port in /etc/ssh/sshd_config like it was mentioned in the previous Ubuntu tutorials. If you do that and enable ufw app OpenSSH, you will lock yourself out of the VPS.

---

mateomarconi *May 11, 2016*

2 Or allow the port that you put in the /etc/ssh/sshd_config for more security

d1str0 *September 12, 2016*

5 Not security, obscurity.

AbleMac *May 3, 2018*

0 FYI: All the characters of your password are on my keyboard. IMHO, password security is only obscurity.

Muhenur10000 *May 11, 2016*

0 What is your vision?

jftuga *May 18, 2016*

5 http://manpages.ubuntu.com/manpages/xenial/man8/ufw.8.html

ufw supports connection rate limiting, which is useful for protecting against brute-force login attacks. When a limit rule is used, ufw will normally allow the connection but will deny connections if an IP address attempts to initiate 6 or more connections within 30 seconds. See http://www.debian-administration.org/articles/187 for details.

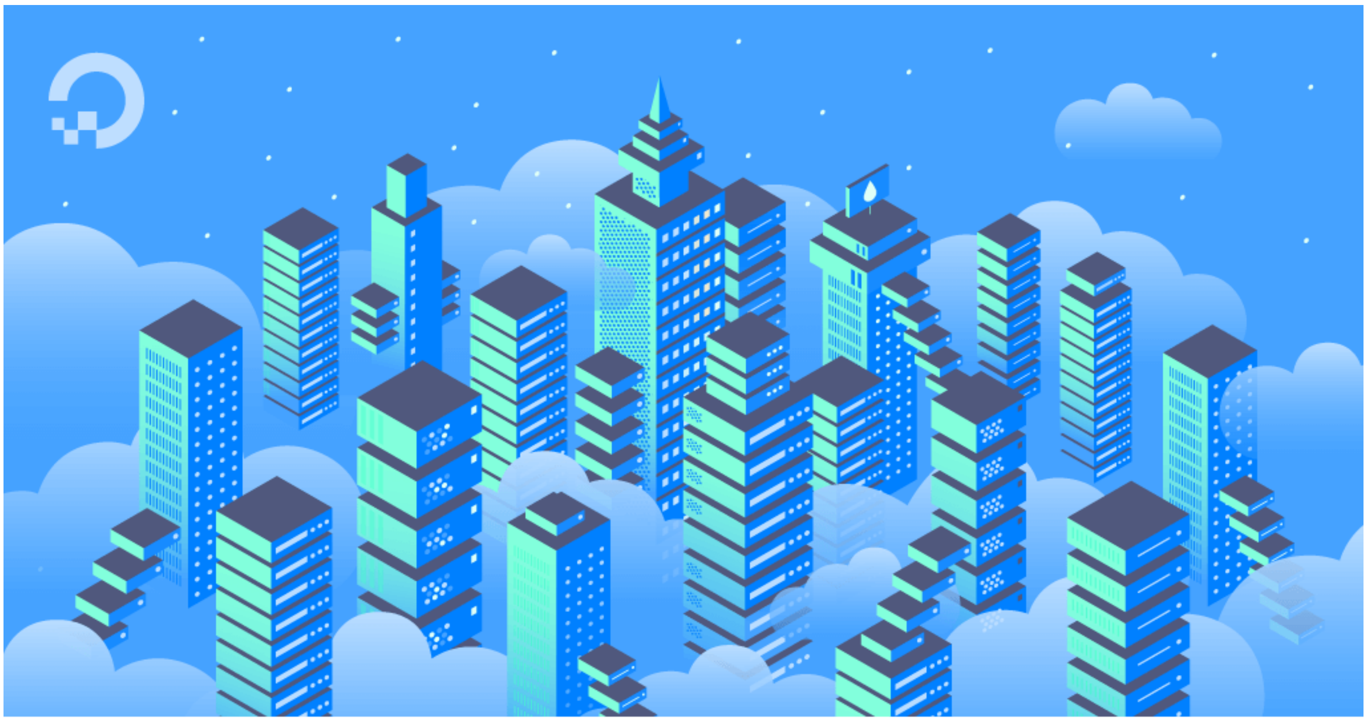
Typical usage is:

```
       ufw limit ssh/tcp
```

johnmeilleur *May 27, 2016*

10 Would it be good to suggest users disable root login via SSH as described in Step Five — Configure SSH Daemon?

https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-14-04



Initial Server Setup with Ubuntu 14.04

by Justin Ellingwood

When you start a new server, there are a few steps that you should take every time to add some basic security and set a solid foundation. In this guide, we'll walk you through the basic steps necessary to hit the ground running with Ubuntu 14.04.

sandwich *September 9, 2016*

0 Also wondering this @manicas

d1str0 *September 12, 2016*

0 Yes, it would be, but it's mostly up to you.

In reality, you should never have to log in as root. Ubuntu for desktop has root disabled/inaccessible for this reason.

johndcarmack *January 1, 2017*

0 I agree. This is the only quibble I have with the tutorial.

Alpers *May 27, 2016*

8 Thanks for the nice article. Just a quick note I thought could be helpful:

I would disable ***root login*** from *sshd_config* file. This would make tracking easier since everything would be recorded in the system log when *sudo* is used. If you needed to fall back to root, you still could do so using *sudo sulogin* command.

**Disabling Root Login**

Type the following command to edit the *sshd_config* file.

```
sudo vim /etc/ssh/sshd_config
```

Find the following line

```
PermitRootLogin yes
```

Change it to *no*

```
PermitRootLogin no
```

Save and quit VIM.

Restart the SSH daemon:

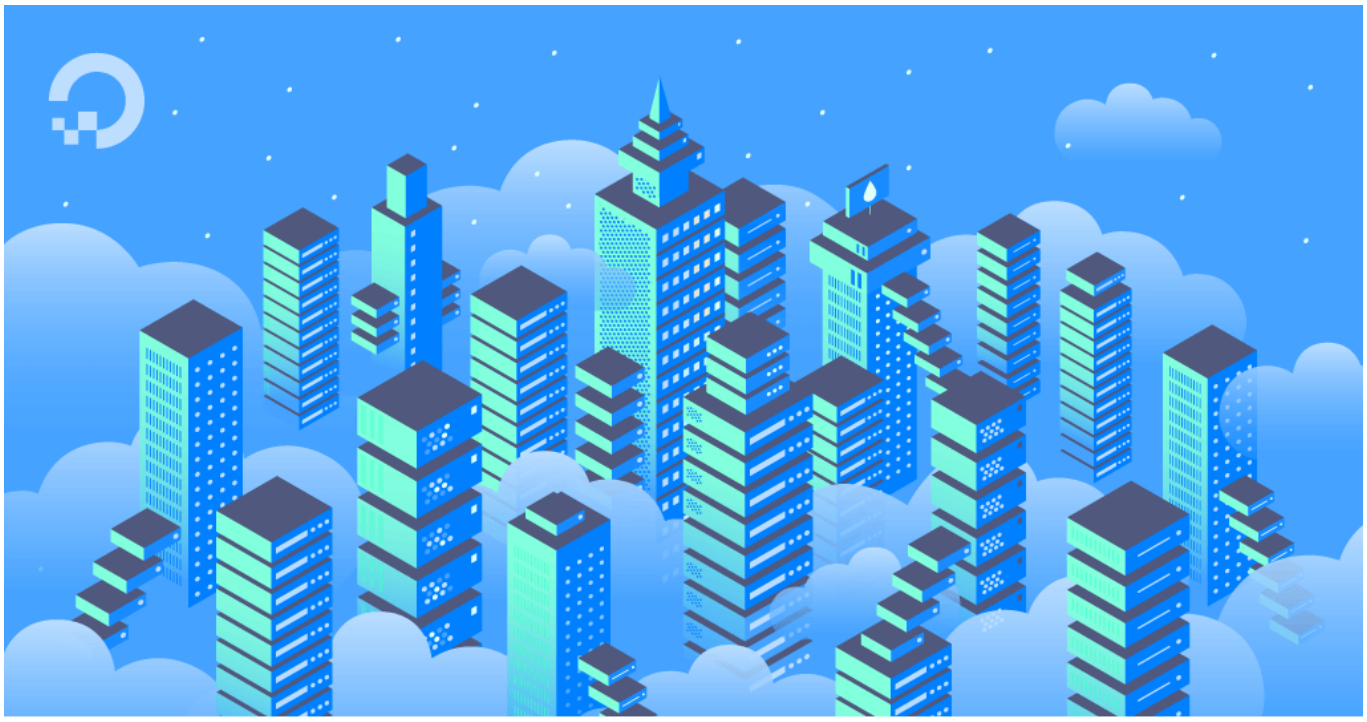
```
sudo service sshd restart
```

Done!

Now you won't be able to login using *root*. If you need root privileges use the sudo command as described above in this article. However, if it *ever* becomes necessary to fall back to the root user, you can do so issuing the following command:

```
sudo sulogin
```

**Footnote:**
If you are not familiar with VIM, you can use nano or use the following article on VIM.
https://www.digitalocean.com/community/tutorials/installing-and-using-the-vim-text-editor-on-a-cloud-server#modal-editing



**Installing and Using the Vim Text Editor on a Cloud Server**

by Justin Ellingwood

In this article, we will cover how to install and utilize the vim text editor. The vim editor is an improvement on the standard "vi" editor standard on every POSIX-compliant system. We will go over the basics of modal editing and how chaining commands can help you accomplish editing tasks

---

mikerscope *June 12, 2016*

4 I would have included Fail2ban

sudo apt install fail2ban -y

your gonna get hammered by brute force attacks on your ssh port anyway why not make them waste time.

---

lagom *June 14, 2016*

0 Great tutorial, much appreciated.

---

devmtl123 *June 16, 2016*

0 I'm having an issue when I enable ufw. I'm locking myself down :-(

```
andy@SVR015:~$ sudo ufw app list
Available applications:
OpenSSH
mosh
andy@SVR015:~$ sudo ufw allow OpenSSH
Rules updated
Rules updated (v6)
andy@SVR015:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
```

Then I can not reconnect at all ...
I tried this 4 times within the last 90 minutes ... Never had an issue with Ubuntu 14.04

Any ideas ?

---

asimbatajoo *July 6, 2016*

0 In the ssh_config file i've set
PasswordAuthentication no
PermitRootLogin without-password

When I run the command
ssh root/user@SERVER*IP*ADDRESS
It asks for the password. I have entered my root password but it says permission denied. And after 3 attempts it says connection closed by host because of Authentication Failure

---

Vardaloupas *July 17, 2016*

0 I complete this guide and all is ok. The only problem I have is that the second user I create I add him ass su bat when I connect from putty and when I try to execute something with sudo the command line prompt me for a password. (The connection with putty is working correct without need a password)

---

idefix *October 24, 2016*

1 I have the same problem. I can login with my new sudo user with the SSH key but when I run the command "sudo nano /etc/ssh/sshd_config" it still asks for password. And my SSH key is password free.

---

tamouse *November 27, 2016*

1 `sudo` will always ask for your password (at least the first time you use it in a certain time frame); it's not about logging in at that point, it's about authorization to raise your user's privileges.

---

GodAtum *June 3, 2018*

0 I have the same issue. i do not have a password, so how can i make this work?

Got it, its the password i created in step 2!

---

dekkermichelle1 *July 19, 2016*

1 Hi Mitchell,

Love your tuts. I ran into trouble where I lock myself out of my droplet. After 2 days of struggling with ssh I finally figured out what was wrong. When you enable the fire wall, you must also add the port. Like so
Step 1
sudo ufw enable
Step 2
sudo ufw allow 22

SSH works out of port 22 by default. If you don't do this you will think you missed a step, delete you droplet and try again. Just to end up back where you started with no solution. If you can just add the second step this will save newbies like me tons of frustration.

I also include a link to https://help.ubuntu.com/12.04/serverguide/firewall.html. There is a ton of useful info.

sal626 *December 7, 2017*

0 No need to delete the droplet when you get yourself locked. Just go to your Droplets page and click on "More" link of the desired droplet and then click "Access console".

Or you can open the droplet by clicking its name from the Droplets listing page, then click "Access" link from the menu on the left and then click "Launch Console".

After the console is being launched, enter "root" as a login or the new user that you created earlier, then enter the password for that user.

You are now logged in to your droplet console.

amata *July 20, 2016*

3 I keep getting **Permission Denied (publickey).** after Step Five. Now can't login neither root or new user.

Zabba *July 21, 2016*

0 Same here! Luckily I kept my root login alive and rolled back the settings in sshd_config.
Any Ideas?

Zabba *July 22, 2016*

0 You need to set the ownership of the .ssh directory and the authorized_keys file to the newly created user, not root.

Log in as root:

```
chown yourusername -R ~/.ssh
```

tamouse *November 27, 2016*

0 If you entered `su - yourusername` prior to creating the `.ssh` directory, it will be under `yourusername`'s ownership already. Make sure to include the dash inbetween.

axelpale *October 10, 2016*

0

Do you use DSA keys? They are denied by default in Ubuntu 16.04 due to security issues. Switch to RSA. See https://blackbricksoftware.com/bit-on-bytes/139-dsa-keys-in-openssh-v7-ubuntu-16-04

icristiano *November 22, 2016*

0 Same here...

tenil *August 12, 2017*

2 I Figured it out. When we pass the ssh-rsa content, sometimes we skip one "s" in the beginning of the file. Check it.
Correct:
ssh-rsa ....
Incorect:
sh-rsa ...
Welcome.

dmz75 *July 28, 2016*

1 I'm a bit confused why disabling root login, which was in previous server setups and is a trivial change to add while disabling password logins, was not included.

Was it intentional? Is there something new in 16.04 that necessitates leaving it? Was it left out unintentionally? If so, with this procedure, which I would normally tell staff to use as is, going to be updated?

jordanbrauer *August 4, 2016*

0 I have the same question.

However, since it seems we will not be receiving an answer *any time soon*, I think it is safe to assume that it was simply forgotten about when writing the tutorial, but because disabling/abstracting the 'root' user access is common practice across all platforms and systems, we as readers can assume that it is still safe to do so.

I also have not read anything about 'root' user access in 16.04 being necessary. 16.04 is pretty close to 14.04 from what I have read and can understand... so I think disabling 'root' user access on your server is still the correct practice. Just make sure you have a new sudo user set up before doing so. :)

joeklinck *August 14, 2016*

1 When someone is trying to hack into your server they need 3 things: a users name, your IP and your password. If you have a root user that can still login in then you are giving them one of those 3 for free. If you have created a sudo user and blocked the root user from logging in then that is one more thing they have to guess. Because every time the try to log in as root it they will be denied.

Another reason is that as the root user you have access to everything and thus have the ability to screw up your server. Having a user with sudo privileges you must use the sudo command before altering any sensitive files.

linux64kb *March 20, 2018*

0 Ubuntu 14.04 and 16.04 does not really have a root password. It prompts you during installation for a custom user name. It will allow to use sudo and gain root access, but not directly. I find this a really clever and good solution. Although technically it's possible to set/enable root login it's not a good idea because of reasons explained by joeklinck.

jammastahk *August 16, 2016*

1 I seem to be having trouble with the public key installation. I am able to generate and copy the public key. However, when it comes to installing the key I am able to open the key (nano ~/.ssh/authorized_keys) but not able to save. I hit CTR-x to exit the file, but am not prompted to save the changes. I'm just brought back to the root user prompt. Any ideas what I may be doing wrong?

helgatheviking *September 7, 2016*

2 The ubuntu PPA package is out of date and I've been advised to use certbot which I've installed from github.

```
sudo apt-get purge --auto-remove letsencrypt

sudo apt update
sudo apt install git

sudo git clone https://github.com/certbot/certbot /opt/certbot
sudo ln -s /opt/certbot/certbot-auto /usr/bin/certbot

certbot certonly --webroot -w /var/www/example.com -d example.com -d www.example.com
```

**Using the webroot plugin means you don't have to stop apache in order to install or renew the cert.* * That was a problem I ran in to.

NB: my apache sites were already configured to look in /etc/letsencrypt/live/example/fullchain.pem

But I had to remove the line from my `/etc/apache2/sites-available/default-ssl.conf`

```
Include /etc/letsencrypt/options-ssl-apache.conf
```

since that not longer exists, and trying to include it prevents apache from starting.

FWIW, there is an unofficial PPA

luismuzquiz *September 11, 2016*

0 I think there is an error on this

I created my user then exited the droplet and login again, as my new user, using my password, created my .key folder and authorized_keys file, pasted my public key and then:

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

but whey i try to login again it kept asking for my password instead of using my keys. It did not worked until i did this:

```
sudo chown username -R ~/.ssh
```

tamouse *November 27, 2016*

0 This is pretty unusual. I'm wondering if you forgot the dash between the 'su' and 'username' in step 5?

luismuzquiz *December 11, 2016*

1 Yes i did. Whats the difference between su username and su - username ?

tamouse *January 2, 2017*

0 As I understand it, leaving off the '-' only gives you the permission level of the user, but putting it in essentially is like logging in (i.e. runs .bash_profile and puts you in their home dir).

In trying this out, though, I found it actually made no difference, so ignore that remark!

Load More Comments