CONTENTS

// TUTORIAL //

How to Set Up SSH Keys on Ubuntu 22.04

Published on April 26, 2022

Getting Started System Tools Linux Basics Security Ubuntu Ubuntu 22.04

By Alex Garnett
Senior DevOps Technical Writer

Not using Ubuntu 22.04?
Choose a different version or distribution.

Ubuntu 22.04

Introduction

SSH, or secure shell, is an encrypted protocol used to administer and communicate with servers. When working with an Ubuntu server, chances are you will spend most of your time in a terminal session connected to your server through SSH.

In this guide, we'll focus on setting up SSH keys for an Ubuntu 22.04 installation. SSH keys provide a secure way of logging into your server and are recommended for all users.

Step 1 – Creating the Key Pair

The first step is to create a key pair on the client machine (usually your computer):

```
$ ssh-keygen
```

Copy

By default recent versions of `ssh-keygen` will create a 3072-bit RSA key pair, which is secure enough for most use cases (you may optionally pass in the `-b 4096` flag to create a larg[illegible]96-bit key).

After entering the command, you should see the following output:

```
Output
Generating public/private rsa key pair.
Enter file in which to save the key (/ your_home /.ssh/id_rsa):
```

Press enter to save the key pair into the `.ssh/` subdirectory in your home directory, or specify an alternate path.

If you had previously generated an SSH key pair, you may see the following prompt:

```
Output
/home/ your_home /.ssh/id_rsa already exists.
Overwrite (y/n)?
```

If you choose to overwrite the key on disk, you will **not** be able to authenticate using the previous key anymore. Be very careful when selecting yes, as this is a destructive process that cannot be reversed.

You should then see the following prompt:

```
Output
Enter passphrase (empty for no passphrase):
```

Here you optionally may enter a secure passphrase, which is highly recommended. A passphrase adds an additional layer of security to prevent unauthorized users from logging in. To learn more about security, consult our tutorial on How To Configure SSH Key-Based Authentication on a Linux Server.

You should then see the output similar to the following:

```
Output
Your identification has been saved in / your_home /.ssh/id_rsa
Your public key has been saved in / your_home /.ssh/id_rsa.pub
The key fingerprint is:
SHA256:/hk7MJ5n5aiqdfTVUZr+2Qt+qCiS7BIm5Iv0dxrc3ks user@host
The key's randomart image is:
+---[RSA 3072]----+
|                .|
|               + |
|              +  |
| .           o . |
|o       S   . o  |
| + o. .oo. ..  .o|
|o = oooooEo+ ...o|
|.. o *o+=.*+o....|
|    =+=ooB=o.... |
+----[SHA256]-----+
```

You now have a public and private key that you can use to authenticate. The next step is to place the public key on your server so that you can use SSH-key-based authentication to log in.

Step 2 – Copying the Public Key to Your Ubuntu Server

The quickest way to copy your public key to the Ubuntu host is to use a utility called `ssh-copy-id`. Due to its simplicity, this method is highly recommended if available. If you do not have `ssh-copy-id` available to you on your client machine, you may use one of the two alternate methods provided in this section (copying via password-based SSH, or manually copying the key).

user account that you have password-based SSH access to. This is the account to which your public SSH key will be copied.

The syntax is:

```
$ ssh-copy-id username @ remote_host
```

Copy

You may see the following message:

```
Output
The authenticity of host ' 203.0.113.1  ( 203.0.113.1 )' can't be established.
E[illegible]A key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.
Are [illegible]sure you want to continue connecting (yes/no)?  yes
```

This means that your local computer does not recognize the remote host. This will happen the first time you connect to a new host. Type “yes” and press `ENTER` to continue.

Next, the utility will scan your local account for the `id_rsa.pub` key that we created earlier. When it finds the key, it will prompt you for the password of the remote user’s account:

```
Output
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted n
 username @ 203.0.113.1 's password:
```

Type in the password (your typing will not be displayed, for security purposes) and press `ENTER`. The utility will connect to the account on the remote host using the password you provided. It will then copy the contents of your `~/.ssh/id_rsa.pub` key into a file in the remote account’s home `~/.ssh` directory called `authorized_keys`.

You should see the following output:

```
Output
Number of key(s) added: 1

Now try logging into the machine, with:   "ssh ' username @ 203.0.113.1 '"
and check to make sure that only the key(s) you wanted were added.
```

At this point, your `id_rsa.pub` key has been uploaded to the remote account. You can continue on to Step 3.

Copying the Public Key Using SSH

If you do not have `ssh-copy-id` available, but you have password-based SSH access to an account on your server, you can upload your keys using a conventional SSH method.

We can do this by using the `cat` command to read the contents of the public SSH key on our local computer and piping that through an SSH connection to the remote server.

On the other side, we can make sure that the `~/.ssh` directory exists and has the correct permissions under the account we’re using.

We can then output the content we piped over into a file called `authorized_keys` within this directory. We’ll use the `>>` redirect symbol to append the content instead of overwriting it. This will let us add keys without destroying previously added keys.

The full command looks like this:

```
$ cat ~/.ssh/id_rsa.pub | ssh  username @ remote_host  "mkdir -p ~/.ssh && ~/.
```

You may see the following message:

```
Output
The authenticity of host ' 203.0.113.1  ( 203.0.113.1 )' can't be established.
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.
Are you sure you want to continue connecting (yes/no)?  yes
```

This means that your local computer does not recognize the remote host. This will happen the first time you connect to a new host. Type `yes` and press `ENTER` to continue.

Afterwards, you should be prompted to enter the remote user account password:

```
Output
 username @ 203.0.113.1 's password:
```

After entering your password, the content of your `id_rsa.pub` key will be copied to the end of the `authorized_keys` file of the remote user's account. Continue on to Step 3 if this was successful.

Copying the Public Key Manually

If you do not have password-based SSH access to your server available, you will have to complete the above process manually.

We will manually append the content of your `id_rsa.pub` file to the `~/.ssh/authorized_keys` file on your remote machine.

To display the content of your `id_rsa.pub` key, type this into your local computer:

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

You will see the key's content, which should look something like this:

```
Output
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCqql6MzstZYh1TmWWv11q5O3pISj2ZFl9HgH1JLknLLx44
```

Access your remote host using whichever method you have available.

Once you have access to your account on the remote server, you should make sure the `~/.ssh` directory exists. This command will create the directory if necessary, or do nothing if it already exists:

```
$ mkdir -p ~/.ssh
```

Copy

Now, you can create or modify the `authorized_keys` file within this directory. You can add the contents of your `id_rsa.pub` file to the end of the `authorized_keys` file, creating it if necessary, using this command:

```
$ echo public_key_string >> ~/.ssh/authorized_keys
```

Copy

In the above command, substitute the `public_key_string` with the output from the `cat ~/.ssh/id_rsa.pub` command that you executed on your local system. It should start with `ssh-rsa AAAA...`.

Finally, we'll ensure that the `~/.ssh` directory and `authorized_keys` file have the appropriate permissions set:

```
$ chmod -R go= ~/.ssh
```

Copy

This recursively removes all "group" and "other" permissions for the `~/.ssh/` directory.

If you're using the **root** account to set up keys for a user account, it's also important that the `~/.ssh` directory belongs to the user and not to **root**:

```
$ chown -R sammy:sammy ~/.ssh
```

Copy

In th[illegible]torial our user is named `sammy` but you should substitute the appropriate userna[illegible]nto the above command.

We can now attempt passwordless authentication with our Ubuntu server.

Step 3 – Authenticating to Your Ubuntu Server Using SSH Keys

If you have successfully completed one of the procedures above, you should be able to log into the remote host *without* providing the remote account's password.

The basic process is the same:

```
$ ssh  username @ remote_host
```

Copy

If this is your first time connecting to this host (if you used the last method above), you may see something like this:

```
Output
The authenticity of host ' 203.0.113.1  ( 203.0.113.1 )' can't be established.
ECDSA key fingerprint is fd:fd:d4:f9:77:fe:73:84:e1:55:00:ad:d6:6d:22:fe.
Are you sure you want to continue connecting (yes/no)?  yes
```

This means that your local computer does not recognize the remote host. Type "yes" and then press `ENTER` to continue.

If you did not supply a passphrase for your private key, you will be logged in immediately. If you supplied a passphrase for the private key when you created the key, you will be prompted to enter it now (note that your keystrokes will not display in the terminal session for security). After authenticating, a new shell session should open for you with the configured account on the Ubuntu server.

If key-based authentication was successful, continue on to learn how to further secure your system by disabling password authentication.

Step 4 – Disabling Password Authentication on Your Server

If you were able to log into your account using SSH without a password, you have successfully configured SSH-key-based authentication to your account. However, your password-based authentication mechanism is still active, meaning that your server is still exposed to brute-force attacks.

Befor[illegible]pleting the steps in this section, make sure that you either have SSH-key-based authentication configured for the **root** account on this server, or preferably, that you have SSH-key-based authentication configured for a non-root account on this server with `sudo` privileges. This step will lock down password-based logins, so ensuring that you will still be able to get administrative access is crucial.

> **Note:** If you provided an SSH key when creating your DigitalOcean droplet, password authentication may have been automatically disabled. You can still verify this by reading on.

Once you've confirmed that your remote account has administrative privileges, log into your remote server with SSH keys, either as **root** or with an account with `sudo` privileges. Then, open up the SSH daemon's configuration file:

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

Inside the file, search for a directive called `PasswordAuthentication`. This line may be commented out with a `#` at the beginning of the line. Uncomment the line by removing the `#`, and set the value to `no`. This will disable your ability to log in via SSH using account passwords:

/etc/ssh/sshd_config

```
. . .
PasswordAuthentication no
. . .
```

Save and close the file when you are finished by pressing `CTRL+X`, then `Y` to confirm saving the file, and finally `ENTER` to exit nano. To actually activate these changes, we need to restart the `sshd` service:

```
$ sudo systemctl restart ssh
```
Copy

As a precaution, open up a new terminal window and test that the SSH service is functioning correctly before closing your current session:

```
$ ssh  username @ remote_host
```
Copy

Once you have verified your SSH service is functioning properly, you can safely close all current server sessions.

The SSH daemon on your Ubuntu server now only responds to SSH-key-based authentication. Password-based logins have been disabled.

Conclusion

You should now have SSH-key-based authentication configured on your server, allowing you to sign in without providing an account password.

If you'd like to learn more about working with SSH, take a look at our SSH Essentials Guide.

About the authors

Alex Garnett Author
Senior DevOps Technical Writer

Still looking for an answer? Ask a question

Search for more help

Was this helpful? Yes No

Comments

8 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

mauriciops • August 4, 2023

On Ubuntu Server 22.04 edit the file `/etc/ssh/sshd_config.d/50-cloud-init.conf` setting `PasswordAuthentication` to `no`. Then restart the ssh service.

Reply

jldo • June 23, 2023

editing the sshd_config file to set “PasswordAuthentication no” does not work. I’ve tried it on 3 different servers.

even changing the port from 22 to anything does not take effect even after rebooting.

u please advice.

Show replies Reply

shahremun • March 23, 2023

after enabling PasswordAuthentication no still possible to login with password i tried both RSA and ECDSA type and still no luck

Show replies Reply

MassiveAzureScubaDiver • March 8, 2023

how i can manage commands and change name of keys?

Show replies Reply

Reek • February 17, 2023

Issue: Step 4 — Disabling Password Authentication on Your Server don't work

Prerequisites:

1. Create new Droplet - Ubuntu 22.04
2. `sudo apt-get update`
3. `sudo apt-get dist-upgrade`

In my `/etc/ssh/sshd_config` the row `PasswordAuthentication no` already exist.

Additionally, tried to set `UsePAM no` - but it's still possible to auth with password.

Show replies Reply

Simon • October 23, 2022

mment has been deleted

nekoizmase • May 6, 2022

You need to generate key using ECDSA or ED25519 algorithm instead of RSA

Show replies Reply

nekoizmase • May 6, 2022

It isn’t working for me. After copying key, password is still required.

Show replies Reply

Try DigitalOcean for free

Click below to sign up and get **$200 of credit** to try our products over 60 days!

Sign up

Popular Topics

Ubuntu

Linux Basics

JavaScript

Pyth

MySQL

Docker

Kubernetes

All tutorials →

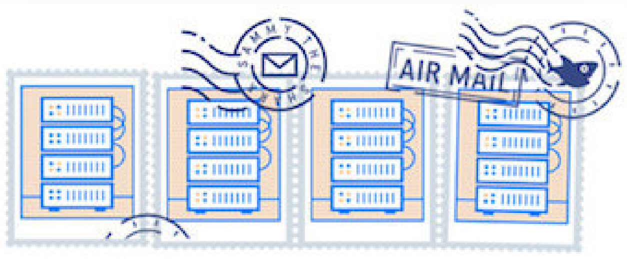
Talk to an expert →

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

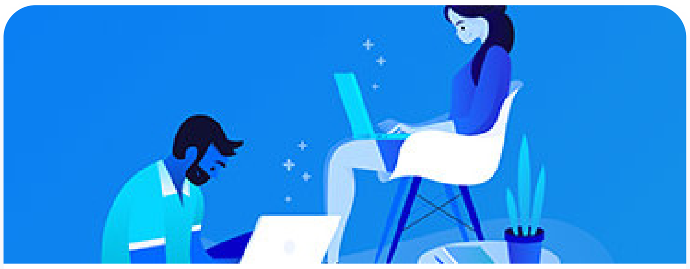
Sign up for Infrastructure as a Newsletter.

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

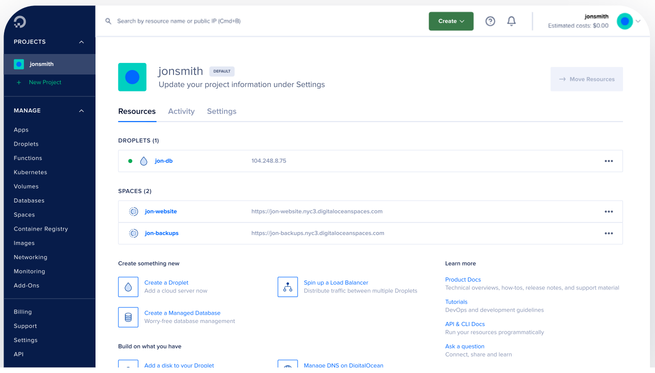
Cloudways Virtual Machines Managed Databases Managed Kubernetes

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

Sign up and get $200 in credit for your first 60 days with DigitalOcean.

Get started

This promotional offer applies to new accounts only.

Company

Products

Community

Solutions

Contact