2 Simple Steps to Set up Passwordless SSH Login on Ubuntu

Last Updated: June 24th, 2022 Xiao Guoan (Admin) 13 Comments Linux Server

This tutorial explains how to set up **passwordless SSH login** on an Ubuntu desktop. There're basically two ways of authenticating user login with OpenSSH server: **password authentication** and **public key authentication**. The latter is also known as **passwordless SSH login** because you don't need to enter your password.

2 Simple Steps to Set Up Passwordless SSH Login

Step 1: Generate a Public/Private Keypair on Your Ubuntu Desktop

On your Ubuntu desktop (not your server), enter the following command in a terminal window.

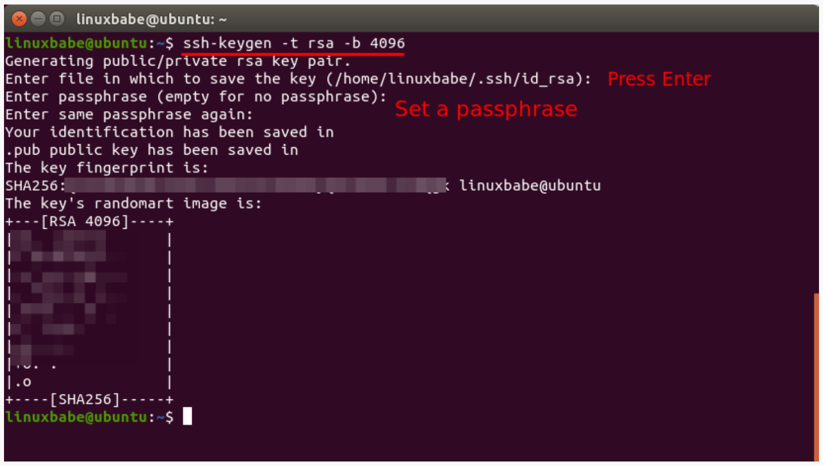
```
ssh-keygen -t rsa -b 4096
```

Where:

- `-t` stands for `type`. The above command generates an RSA type keypair. RSA is the default type.
- `-b` stands for `bits`. By default the key is 3072 bits long. We use a 4096 bits key for stronger security.

When asked which file to save the key, you can simply press `Enter` to use the default file. Next, enter a good passphrase at least 20 characters long. The passphrase is used to encrypt the private key.

- The private key (your identification) will be save in the **.ssh/id_rsa** file under your home directory.
- The public key will be save in the **.ssh/id_rsa.pub** file.



```
linuxbabe@ubuntu:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/linuxbabe/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
.pub public key has been saved in
The key fingerprint is:
SHA256:[redacted] linuxbabe@ubuntu
The key's randomart image is:
+---[RSA 4096]----+
[redacted]
|.o. .            |
|.o               |
+----[SHA256]-----+
linuxbabe@ubuntu:~$
```

From the randomart image we can see the length of the key (`RSA 4096`). Now run the following command.

```
file ~/.ssh/id_rsa
```

You should see the following output:

```
/home/username/.ssh/id_rsa: OpenSSH private key
```

If you see the “No such file or directory” error, that means the SSH keypair isn’t created. Run the `ssh-keygen -t rsa -b 4096` command to create it again.

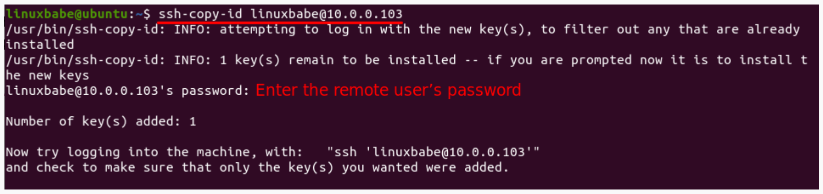
Step 2: Upload Your Public Key to Remote Linux Server

Hint: The remote server can run any Linux distro: Debian, Ubuntu, RHEL, CentOS, whatever, as long as it runs OpenSSH server, you can use the following method.

This can be easily done with `ssh-copy-id` command, which is shipped with the `openssh-client` package.

```
ssh-copy-id remote-user@server-ip
```

Enter the remote user’s password.

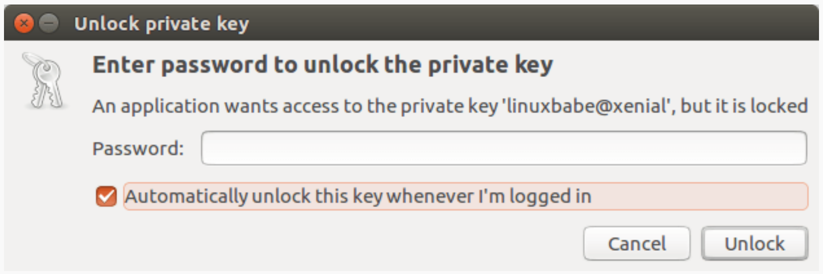


```
linuxbabe@ubuntu:~$ ssh-copy-id linuxbabe@10.0.0.103
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already
installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install t
he new keys
linuxbabe@10.0.0.103's password: Enter the remote user's password

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'linuxbabe@10.0.0.103'"
and check to make sure that only the key(s) you wanted were added.
```

The public key will be stored in the **.ssh/authorized_keys** file under the remote user’s home directory. Now SSH into the remote server.

```
ssh remote-user@server-ip
```

This time you need to enter your RSA **key passphrase** to unlock the private key. You can select automatic unlocking the key when logging in so you don’t have to enter the passphrase in the future.



Unlock private key

Enter password to unlock the private key

An application wants access to the private key 'linuxbabe@xenial', but it is locked

Password:

[✓] Automatically unlock this key whenever I'm logged in

Once you entered the correct key passphrase, you are logged into the remote Linux server. Now exit from the remote server.

```
exit
```

And SSH into the remote server again:

```
ssh remote-user@server-ip
```

This time you are automatically logged into the remote server, although you didn't type password or key passphrase. Also, you don't have to type password or key passphrase when using the `scp` command to transfer files. The `scp` command is also shipped by the `openssh-client` package, which is installed by default on Ubuntu desktop.

If your Ubuntu desktop doesn't open up a dialog to remember your key passphrase, then you can manually add the key passphrase with the following command.

```
secret-tool store --label="Unlock pass
word for: id_rsa" unique "ssh-store:/h
ome/username/.ssh/id_rsa"
```

Disabling Password Authentication

Although SSH key is now used by default to log into your server, you can still use normal password to log into the server on another computer. You don't want hackers to launch brute force attack to hack into your server, so it's a good practice to disable password authentication in OpenSSH server.

To disable password authentication, edit `/etc/ssh/sshd_config` file on the remote server.

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

Find this line:

```
#PasswordAuthentication yes
```

Change it to:

```
PasswordAuthentication no
```

Then find the `ChallengeResponseAuthentication` line. Make sure it’s value is set to `no` like below. If it’s set to `yes`, you can still use password to login.

```
ChallengeResponseAuthentication no
```

Save the file and restart SSH service.

Debian/Ubuntu

```
sudo systemctl restart ssh
```

RHEL/CentOS

```
sudo systemctl restart sshd
```

Now if you don’t have the corresponding private key in `~/.ssh` directory, you will see the following error when you try to SSH into your remote server.

```
Permission denied (publickey).
```

or

```
Read: Connection reset by peer
```

That means the remote server only allow SSH login using ssh keys and do not allow password authentication. **Note** that if you set `PasswordAuthentication` to `no` and `ChallengeResponseAuthentication` to `yes`, then you can still login using password. To disable password login, both of them must be set to `no`.

Backing up Your Public/Private Keypair

Once you disable SSH password authentication, it is very important to back up your SSH keys. If you lose the keys you will be locked out of your server. Back up your public/private keypair to a safe location such as your USB drive.

```
cp ~/.ssh/id_rsa* /path/to/safe/location/
```

You can also store your key pair in a folder, then compress the folder with encryption and send it to cloud storage like NextCloud.

You can also copy the key pair (both the private key and public key) to a new Linux computer and SSH into your server using SSH keys. Once you copied the key pair to a new computer, move them to the `.ssh/` directory of the new user.

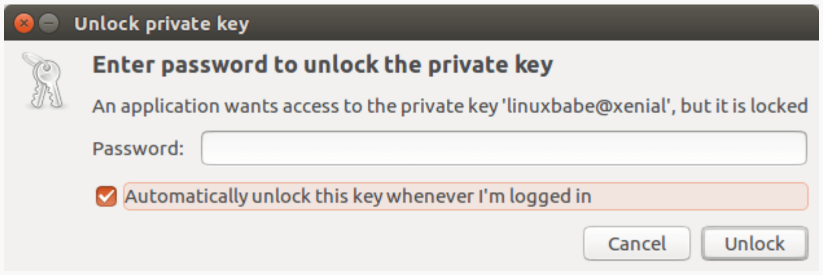
```
mv id_rsa* ~/.ssh/
```

You need to change the owner of the key pair to the user on the new computer.

```
sudo chown new-user:new-user ~/.ssh/id_rsa*
```

Now you can use SSH keys to log into remote server on the new computer.

Sometimes, I would like to create two user accounts on my local Ubuntu computer to do different tasks. However, I still want to use the SSH key when I log into a different user account. Simply copy the SSH key pair (both the private key and public key) to the new users' `~/.ssh/` folder. You need to enter your RSA **key passphrase** to unlock the private key. You can select automatic unlocking the key when logging in so you don't have to enter the passphrase in the future.


Unlock private key

Enter password to unlock the private key

An application wants access to the private key 'linuxbabe@xenial', but it is locked

Password:

[✓] Automatically unlock this key whenever I'm logged in

Storing Key Passphrase in SSH Agent

Hint: Please don't follow this section if your local computer has a graphical user interface.

If you are using a command line only Linux box, you may find that you need to enter the key passphrase every time you SSH into other Linux servers. That's because your key passphrase is not stored by SSH agent.

Install and configure **keychain** on the SSH client box.

```
sudo apt install keychain
```

Then edit **.bash_profile** or **.profile** file. Append the following text into it so these two commands will be executed every time the user login.

```
/usr/bin/keychain $HOME/.ssh/id_rsa
source $HOME/.keychain/$HOSTNAME-sh
```

Now logout and log back in. You will see something like:

```
Last login: Thu Dec 17 20:38:39 2015 from 74.125.128.103

* keychain 2.7.1 ~ http://www.funtoo.org
* Found existing ssh-agent: 17651
* Adding 1 ssh key(s): /home/<username>/.ssh/id_rsa
Enter passphrase for /home/<username>/.ssh/id_rsa:
* ssh-add: Identities added: /home/<username>/.ssh/id_rsa
```

When key chain starts, it checks for a running ssh-agent, otherwise it starts one. You need to enter the key passphrase this time. The key passphrase will be remembered across user logins, but when the system reboots, you have to enter it again.

Now as long as the ssh server has the public key and the ssh client you are working on right now has private/public keypair and keychain successfully configured, you can ssh into the ssh server without typing key passphrase.

Changing Private Key Passphrase

If you ever need to change your private key passphrase, you can do so with this command:

```
ssh-keygen -f ~/.ssh/id_rsa -p
```

Enter your old passphrase and then enter a new passphrase.

Pro Tip: Use Screen to Keep Your Session Alive

Have you been doing work on the remote server and suddenly your computer is disconnected from Internet and you can no longer continue the running job on the server? You can use the wonderful `screen` utility to keep your session alive. Install `screen` on the Ubuntu server:

```
sudo apt install screen
```

Then start screen:

```
screen
```

Upon first launch, you will see the startup message. Simply press `Enter` to end. Then you will be able to run commands as usual.

If you have a long-running job on the server and you don't need to do other things on the server now, you can press **Ctrl+A**, release those keys, and then press **D** key to detach from the current Screen session. You will see a message like below.

```
[detached from 32113.pts-1.focal]
```

This tells you that the previous Screen session ID is 32113. You can log out from the SSH session and even shut down your local computer. Don't worry, the job on the server is still running. When you need to come back and check the progress, SSH into your server and run the following command to get the previous Screen Session ID.

```
screen -ls
```

Sample output:

```
There is a screen on:
        32113.pts-1.focal       (05/1
9/2020 03:45:29 PM)     (Detached)
1 Socket in /run/screen/S-linuxbabe.
```

Then you can re-attach to the previous Screen session.

```
screen -r 32113
```

If you are in a Screen session and suddenly your Internet connection drops, then you can run the following command on the server when you have Internet connection again.

```
screen -d -r 32113
```

This time we need the `-d` option because the previous Screen session wasn't detached. We need to detach it first (`-d`) , then reattach to it (`-r`).

If you don't like to see the Screen startup message every time, edit the Screen config file.

```
sudo nano /etc/screenrc
```

Find the following line. Remove the `#` character to uncomment it.

```
#startup_message off
```

Save the file.

Pro Tip #2: Automatically Start Screen Session

Edit the `.bashrc` file.

```
nano ~/.bashrc
```

Add the following line at the bottom of this file.

```
if [ -z "$STY" ]; then screen -RR; fi
```

Save and close the file. Next time when you SSH into the remote server, a Screen session will be automatically started.

Next Step

I hope this tutorial helped you to set up passwordless ssh login on Ubuntu. You may also want to set up automatic security update.

- Set Up Automatic Security Update (Unattended Upgrades) on Ubuntu

If you want FTP access to the Ubuntu server, you can set up pure-FTPd server.

- How to Set Up a Secure FTP Server with Pure-FTPd on Ubuntu