

# How To Configure SSH Key-Based Authentication on a Linux Server

Posted October 20, 2014

 1.4m

GETTING STARTED

LINUX BASICS

SECURITY

SYSTEM TOOLS

## Introduction

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

While there are a few different ways of logging into an SSH server, in this guide, we'll focus on setting up SSH keys. SSH keys provide an easy, yet extremely secure way of logging into your server. For this reason, this is the method we recommend for all users.

## How Do SSH Keys Work?

An SSH server can authenticate clients using a variety of different methods. The most basic of these is password authentication, which is easy to use, but not the most secure.

Although passwords are sent to the server in a secure manner, they are generally not complex or long enough to be resistant to repeated, persistent attackers. Modern processing power combined with automated scripts make brute forcing a password-protected account very possible. Although there are other methods of adding additional security ( `fail2ban` , etc.), SSH keys prove to be a reliable and secure alternative.

SSH key pairs are two cryptographically secure keys that can be used to authenticate a client to an SSH server. Each key pair consists of a public key and a private key.

The private key is retained by the client and should be kept absolutely secret. Any compromise of the private key will allow the attacker to log into servers that are configured with the associated public key without additional authentication. As an additional precaution, the key can be encrypted on disk with a passphrase.

The associated public key can be shared freely without any negative consequences. The public key can be used to encrypt messages that **only** the private key can decrypt. This property is employed as a way of authenticating using the key pair.

The public key is uploaded to a remote server that you want to be able to log into with SSH. The key is added to a special file within the user account you will be logging into called `~/.ssh/authorized_keys`.

When a client attempts to authenticate using SSH keys, the server can test the client on whether they are in possession of the private key. If the client can prove that it owns the private key, a shell session is spawned or the requested command is executed.

## How To Create SSH Keys

The first step to configure SSH key authentication to your server is to generate an SSH key pair on your local computer.

To do this, we can use a special utility called `ssh-keygen`, which is included with the standard OpenSSH suite of tools. By default, this will create a 2048 bit RSA key pair, which is fine for most uses.

On your local computer, generate a SSH key pair by typing:

```
ssh-keygen
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (/home/username/.ssh/id_rsa):
```

The utility will prompt you to select a location for the keys that will be generated. By default, the keys will be stored in the `~/.ssh` directory within your user's home directory. The private key will be called `id_rsa` and the associated public key will be called `id_rsa.pub`.

Usually, it is best to stick with the default location at this stage. Doing so will allow your SSH client to automatically find your SSH keys when attempting to authenticate. If you would like to choose a non-standard path, type that in now, otherwise, press ENTER to accept the default.

If you had previously generated an SSH key pair, you may see a prompt that looks like this:

```
/home/username/.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.

```
Created directory '/home/username/.ssh'.
```

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

```
Enter same passphrase again:
```

Next, you will be prompted to enter a passphrase for the key. This is an optional passphrase that can be used to encrypt the private key file on disk.

You may be wondering what advantages an SSH key provides if you still need to enter a passphrase. Some of the advantages are:

- The private SSH key (the part that can be passphrase protected), is never exposed on the network. The passphrase is only used to decrypt the key on the local machine. This means that network-based brute forcing will not be possible against the passphrase.
- The private key is kept within a restricted directory. The SSH client will not recognize private keys that are not kept in restricted directories. The key itself must also have restricted permissions (read and write only available for the owner). This means that other users on the system cannot snoop.
- Any attacker hoping to crack the private SSH key passphrase must already have access to the system. This means that they will already have access to your user account or the root account. If you are in this position, the passphrase can prevent the attacker from immediately logging into your other servers. This will hopefully give you time to create and implement a new SSH key pair and remove access from the compromised key.

Since the private key is never exposed to the network and is protected through file permissions, this file should never be accessible to anyone other than you (and the root user). The passphrase serves as an additional layer of protection in case these conditions are compromised.

A passphrase is an optional addition. If you enter one, you will have to provide it every time you use this key (unless you are running SSH agent software that stores the decrypted key). We recommend using a passphrase, but if you do not want to set a passphrase, you can simply press ENTER to bypass this prompt.

Your identification has been saved in `/home/username/.ssh/id_rsa`.

Your public key has been saved in `/home/username/.ssh/id_rsa.pub`.

The key fingerprint is:

`a9:49:2e:2a:5e:33:3e:a9:de:4e:77:11:58:b6:90:26 username@remote_host`

The key's randomart image is:

```

+--[ RSA 2048 ]-----+
|      ..o      |
|    E o= .     |
|    o. o       |
|      ..       |
|    ..S        |
|    o o.       |
|   =o.+        |
|. =++..        |
|o=++..        |
+-----+

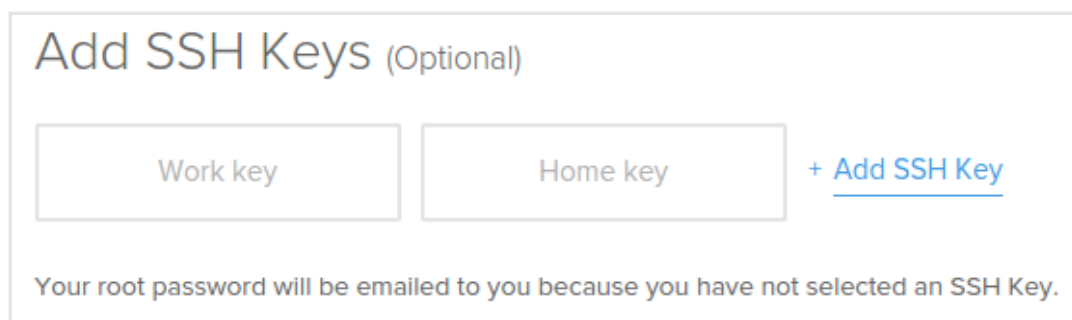
```

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 authentication to log in.

## How to Embed your Public Key when Creating your Server

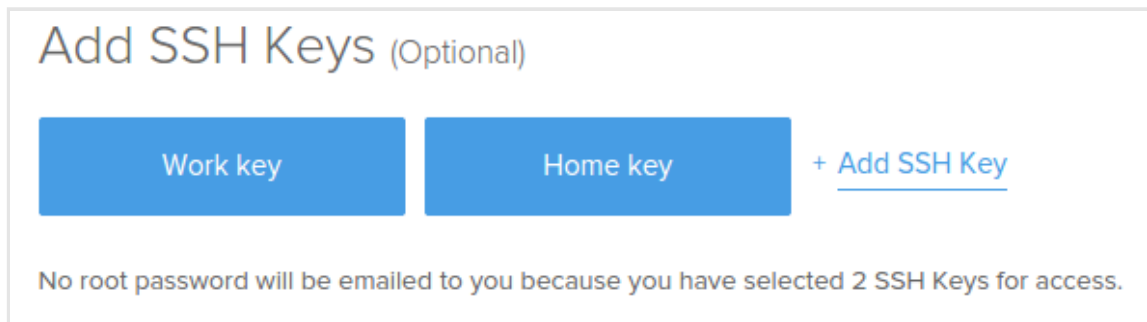
If you are starting up a new DigitalOcean server, you can automatically embed your SSH public key in your new server's root account.

Towards the bottom of the Droplet creation page, there is an option to add SSH keys to your server:



If you have already added a public key file to your DigitalOcean account, you will see it here as a selectable option (there are two existing keys in the example above: "Work key" and "Home key"). To

embed an existing key, simply click on it and it will highlight. You can embed multiple keys on a single server:

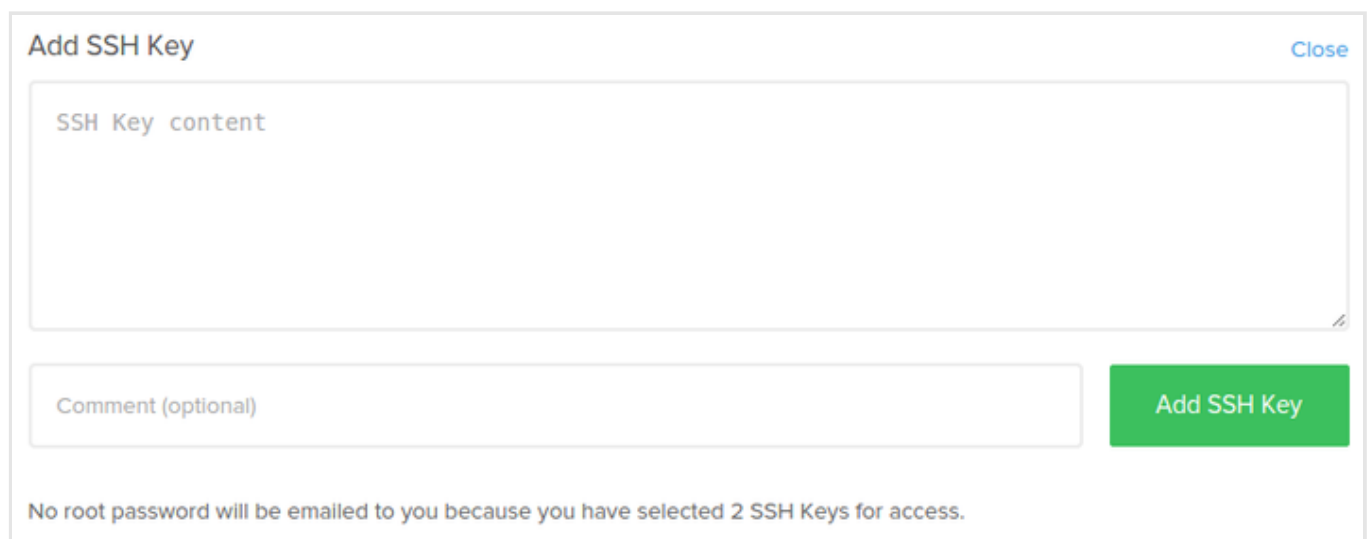


Add SSH Keys (Optional)

Work key Home key + [Add SSH Key](#)

No root password will be emailed to you because you have selected 2 SSH Keys for access.

If you do not already have a public SSH key uploaded to your account, or if you would like to add a new key to your account, click on the "+ Add SSH Key" button. This will expand to a prompt:



Add SSH Key Close

SSH Key content

Comment (optional)

Add SSH Key

No root password will be emailed to you because you have selected 2 SSH Keys for access.

In the "SSH Key content" box, paste the content of your SSH public key. Assuming you generated your keys using the method above, you can obtain your public key contents on your local computer by typing:

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

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQBNqqi1mHLnryb1FdbePrSZQdmXRZxGZbo0gTfg1ysq6KMNUNY2VhzmYN!
```

Paste this value, in its entirety, into the larger box. In the "Comment (optional)" box, you can choose a label for the key. This will be displayed as the key name in the DigitalOcean interface:

Add SSH Key

Close

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQBNqqi1mHLnryb1FdbePrSZQdmXRZxGZbo0gTfglysq6KMNUNY2VhzmYN9JYW
39yNtjhVxqfW6ewc+eHiL+IRRM1P5ecDAaL3V0ou6ecSurU+t9DR4114mzNJ5SqNxMgiJzbXdhR+j55Gjfxdk0Fyzx
M3a5qpVcGZEXiAzGzhHytUV51+YGnuLgaZ37nebh3UlyC+KJev4MYIVww0tWmY+9GniRSQlgLLUQZ+FcBUjaqhwqVq
sHe4F/woW1IHe7mf63GxyBavVc+llrEzRbM0111MogZUcOWDI9w7UIm8Z0TnhJsk7jhJzG2GpSXZHmly/a/buFaaF
nmfZ4MPkgJD username@example.com
```

New key

Add SSH Key

When you create your Droplet, the public SSH keys that you selected will be placed in the `~/.ssh/authorized_keys` file of the root user's account. This will allow you to log into the server from the computer with your private key.

## How To Copy a Public Key to your Server

If you already have a server available and did not embed keys upon creation, you can still upload your public key and use it to authenticate to your server.

The method you use depends largely on the tools you have available and the details of your current configuration. The following methods all yield the same end result. The easiest, most automated method is first and the ones that follow each require additional manual steps if you are unable to use the preceding methods.

### Copying your Public Key Using SSH-Copy-ID

The easiest way to copy your public key to an existing server is to use a utility called `ssh-copy-id`. Because of its simplicity, this method is recommended if available.

The `ssh-copy-id` tool is included in the OpenSSH packages in many distributions, so you may have it available on your local system. For this method to work, you must already have password-based SSH access to your server.

To use the utility, you simply need to specify the remote host that you would like to connect to and the user account that you have password SSH access to. This is the account where your public SSH key will be copied.

The syntax is:

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

You may see a message like this:

```
The authenticity of host '111.111.11.111 (111.111.11.111)' 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 just 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:

```
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that ai  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to  
username@111.111.11.111'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 will see output that looks like this:

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

Now try logging into the machine, with: `"ssh 'username@111.111.11.111'"`  
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 onto the next section.

## Copying your 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 outputting the content of our public SSH key on our local computer and piping it through an SSH connection to the remote server. On the other side, we can make sure that the `~/.ssh` directory exists under the account we are using and then output the content we piped over into a file called `authorized_keys` within this directory.

We will 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 will look like this:

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

You may see a message like this:

```
The authenticity of host '111.111.11.111 (111.111.11.111)' 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 just 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 will be prompted with the password of the account you are attempting to connect to:

```
username@111.111.11.111'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 to the next section if this was successful.

## Copying your Public Key Manually

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

The content of your `id_rsa.pub` file will have to be added to a file at `~/.ssh/authorized_keys` on your remote machine somehow.



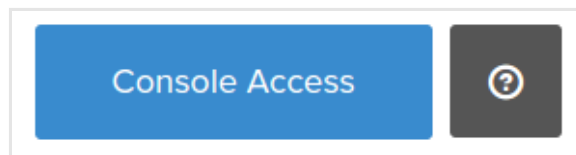
To display the content of your `id_rsa.pub` key, type this into your local computer:

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

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

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQCqql6MzstZYh1TmWwv11q503pISj2ZF19HgH1JLknLLx44+tXfJ7mIrKI
```

Access your remote host using whatever method you have available. For instance, if your server is a DigitalOcean Droplet, you can log in using the web console in the control panel:



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

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

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:

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

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...`

If this works, you can move on to try to authenticate without a password.

## Authenticate to your 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* the remote account's password.

The basic process is the same:

```
ssh username@remote_host
```

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

```
The authenticity of host '111.111.11.111 (111.111.11.111)' 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 just 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 required to enter it now. Afterwards, a new shell session should be spawned for you with the account on the remote system.

If successful, continue on to find out how to lock down the server.

## Disabling Password Authentication on your Server

If you were able to login to 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.

Before completing 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 an account on this server with `sudo` access. This step will lock down password-based logins, so ensuring that you have will still be able to get administrative access is essential.

Once the above conditions are true, log into your remote server with SSH keys, either as root or with an account with `sudo` privileges. Open the SSH daemon's configuration file:

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

Inside the file, search for a directive called `PasswordAuthentication`. This may be commented out. Uncomment the line and set the value to "no". This will disable your ability to log in through SSH using account passwords:

```
PasswordAuthentication no
```

Save and close the file when you are finished. To actually implement the changes we just made, you must restart the service.

On Ubuntu or Debian machines, you can issue this command:

```
sudo service ssh restart
```

On CentOS/Fedora machines, the daemon is called `sshd`:

```
sudo service sshd restart
```

After completing this step, you've successfully transitioned your SSH daemon to only respond to SSH keys.

## Conclusion

You should now have SSH key-based authentication configured and running on your server, allowing you to sign in without providing an account password. From here, there are many directions you can head. If you'd like to learn more about working with SSH, take a look at our [SSH essentials guide](#).