# Implement SSH key-based authentication on Linux

SSH (Secured Shell) is a protocol which creates a cryptographically secured connection between the SSH client and remote servers.

Using SSH you can manage and administer remote servers securely. This can be helpful in many ways.

* Multi-server deployment
* Stop/start services remotely
* Automation

and all your creativity (hopefully)…

As a sysadmin, this is kind of basic stuff to know.

Let’s learn how…

# How To

I will generate a private key and a public key.

* The private key should be stored on your ssh client machine and must be kept secured.
* The public key must be copied to the remote server to login to that server from the SSH client machine with no password required.

You’ll learn the following.

* Install SSH application on server and local (not required if already installed)
* Generate SSH Keys
* Copy SSH Key to a remote server
* Log in to the remote server using SSH

For demonstration purpose, I have 2 servers (hosts) with below IP addresses, one system is a client and the other one is a server on which I will log in through ssh from a client machine.

* Client (user -> geekflare) : 192.168.56.102
* Remote (user -> ubuntu) : 192.168.56.101

**Installing SSH**

Before you follow the steps mentioned in this article, make sure openssh-server is installed on the servers. If it is not installed, run the commands below to install.

sudo apt-get install openssh-server

sudo systemctl enable ssh

sudo systemctl start ssh

Copy

You can run ssh command to check whether it is installed on the system or not.

geekflare@geekflare:~$ ssh

usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind\_interface]

[-b bind\_address] [-c cipher\_spec] [-D [bind\_address:]port]

[-E log\_file] [-e escape\_char] [-F configfile] [-I pkcs11]

[-i identity\_file] [-J [user@]host[:port]] [-L address]

[-l login\_name] [-m mac\_spec] [-O ctl\_cmd] [-o option] [-p port]

[-Q query\_option] [-R address] [-S ctl\_path] [-W host:port]

[-w local\_tun[:remote\_tun]] destination [command]

Copy

**Generate SSH Keys**

This needs to be done on a client server.

Run the ssh-keygen command to generate a SSH key. Just press enter when it asks for the file, passphrase, same passphrase. It generates a pair of keys in ~/.ssh directory by default. Id\_rsa is the private key and id\_rsa.pub is the associate public key.

geekflare@geekflare:~$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/home/geekflare/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /home/geekflare/.ssh/id\_rsa.

Your public key has been saved in /home/geekflare/.ssh/id\_rsa.pub.

The key fingerprint is:

SHA256:3XDm62tzJegGm8oAmFYCyeFZovJOuU42zNgyn9GzH30 geekflare@geekflare

The key's randomart image is:

+---[RSA 2048]----+

|o+.. |

|+o+ |

|oo. . . o |

|.. \* . \* |

| B . S . o. |

| O o . . . ... .|

|+ @ o o . E=. o |

| B + o + .o.= . |

| + ... o. oo+ |

+----[SHA256]-----+

Copy

It will generate two new files in ~/.ssh directory.

**Copy SSH Key to Remote Server**

The private key should be copied ~/.ssh folder on a remote server. Most of the servers should already have this folder if not, you need to create a folder.

And, to do so:

* Log in to the remote server with the user you would like to get connected. In my case, its ubuntu
* Ensure the present working directory is the user’s home directory and then create a .ssh folder. You can also use the following single command to create one

mkdir -p ~/.ssh

Copy

If you already have .ssh folder then take a backup of it.

Next, let’s push the public key from a client server.

On the client machine (192.168.56.102), run the command below to copy the public key on the remote server inside an authorized\_keys file in .ssh directory.

geekflare@geekflare:~$ cat .ssh/id\_rsa.pub | ssh ubuntu@192.168.56.101 'cat >> .ssh/authorized\_keys'

ubuntu@192.168.56.101's password:

Copy

Run the command below to set permissions on the authorized\_keys file on remote server.

geekflare@geekflare:~$ ssh ubuntu@192.168.56.101 "chmod 700 .ssh; chmod 640 .ssh/authorized\_keys"

Copy

Great, this concludes key is exchanged and permission is all set.

**Login to Remote Server using SSH**

Let’s test to see if it works!

Let’s login to the remote server (192.168.56.101) from a client machine (192.168.56.102) as geekflare user.

Run the command below to test, it won’t ask password this time.

geekflare@geekflare:~$ ssh ubuntu@192.168.56.101

Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-151-generic i686)

\* Documentation: https://help.ubuntu.com

\* Management: https://landscape.canonical.com

\* Support: https://ubuntu.com/advantage

346 packages can be updated.

11 updates are security updates.

Last login: Mon Jun 17 00:10:32 2019 from 192.168.56.101

Copy

Here you go…

I have logged in to remote server successfully. Running below command will give the IP details of the remote machine.

ubuntu@ubuntu:~$ ifconfig

enp0s3 Link encap:Ethernet HWaddr 08:00:27:9b:47:86

inet addr:10.0.2.15 Bcast:10.0.2.255 Mask:255.255.255.0

inet6 addr: fe80::5c62:3267:b752:fe5d/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:20239 errors:0 dropped:0 overruns:0 frame:0

TX packets:5406 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:22678039 (22.6 MB) TX bytes:701710 (701.7 KB)

enp0s8 Link encap:Ethernet HWaddr 08:00:27:a9:4a:6b

inet addr:192.168.56.101 Bcast:192.168.56.255 Mask:255.255.255.0

inet6 addr: fe80::54a9:761c:9034:21a2/64 Scope:Link

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:330 errors:0 dropped:0 overruns:0 frame:0

TX packets:197 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1000

RX bytes:42847 (42.8 KB) TX bytes:32774 (32.7 KB)

lo Link encap:Local Loopback

inet addr:127.0.0.1 Mask:255.0.0.0

inet6 addr: ::1/128 Scope:Host

UP LOOPBACK RUNNING MTU:65536 Metric:1

RX packets:997 errors:0 dropped:0 overruns:0 frame:0

TX packets:997 errors:0 dropped:0 overruns:0 carrier:0

collisions:0 txqueuelen:1

RX bytes:79654 (79.6 KB) TX bytes:79654 (79.6 KB)

Copy

**Conclusion**

Setting up SSH key exchange is very straightforward as you can see. I hope this helps you and interested in learning Linux administration and troubleshooting then check out this [Udemy course](https://www.udemy.com/red-hat-linux-administration-advance-level-troubleshooting/).