Passwordless SSH - 4 Steps
Author - Nic Maurel

SSH stands for very secure shell. It's function is exactly it's name, can encrypt a remote connecion to another UNIX or LINUX pc's shell using port 22 as a default. But so what you say? Well what if you don't want to be around when 2 computers connect? This could be useful in many different ways and make your workload a LOT lighter. For instance :

- Setup a script to do remote commands to another pc
- Start or stop certain services, using a kill command
- Execute a list of commands sitting either on the local pc or the remote pc
- Get or put files securly using sftp or scp.

The above examples are just a few things I can think off of the top of my head, the possibilities are endless. First of all we have to know how the 2 pcs authenticate with each other. Usually if you are sitting at your local pc and you want to connect to a remote pc with ssh you need the following

- You need to know a username and password on the remote machine eg. root
- Ensure that sshd is running on the remote machine. Type this on the remote machine if you have local access to it (you would have to be root to run these commands)

```
# ps -ef | grep sshd
  root     3138     1  0 Mar20 ?        00:00:07 /usr/sbin/sshd
  root      727  3138  0 11:15 ?        00:00:00 /usr/sbin/sshd
  root      770   729  0 11:15 pts/0    00:00:00 grep sshd
```

- Lastly to connect to the remote machine you will need an ssh client. Type this on your local LINUX machine

```
# ssh -V
OpenSSH_3.9p1, SSH protocols 1.5/2.0, OpenSSL 0x0090701f
```

Please note: if the ssh command and grep for sshd did not work it is probably either not installed on your system or not running. The best thing to do would be to consult your documentation or download a binary or source from the openssh website. Okay now that we have established that sshd is running we should now be able to connect to a remote pc from the local one. For reference I will be refering to the local pc as *[Local]#* and the remote pc as *[remote]#*

Step number 1-Connecting to remote
First try connecting -

[root@local]# ssh root@remote

This will only happen the first time you connect -

The authenticity of host 'remote (xxx.xxx.xxx.xxx)' can't be established.
RSA key fingerprint is 9f:61:f3:c6:f5:f5:37:e4:fb:59:eb:ed:4d:a9:62:45.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'remote' (RSA) to the list of known hosts.
root@remote's password:

Type your password and if you have typed it correctly you should be prompted with a remote root shell something like this -

[root@remote]#

Well done you have successfully connected to the remote machine. Now for the fun part!

Step number 2-Creating the keys
Setting up Public key encryption is tricky but not that difficult. Here is what we are aiming for:

local has its Private Key	----->	remote has locals's Public key
id_rsa		id_rsa.pub

Time to create Local's keys. Running this command as root means that only root on local will be able to use these keys

```
    [root@local]# ssh-keygen -t rsa -P ""
```

I have chosen to create rsa keys as personal preference, if you prefer dsa then that is fine, they both have very strong encryption but the only marginal difference is that dsa is faster at generating keys and rsa is faster at authenticating, but this is a millisecond time difference. You will then be prompted to specify where to put the keys, press <*enter*> to accept the default *~/.ssh/id_rsa* otherwise you can specify elsewhere.

TIP: ~ stands for the user's home directory, so if you are root this would most likely be /root

From the *sshkeygen* command 2 keys are generated *id_rsa* and *id_rsa.pub*. PLEASE NEVER EXPOSE the *id_rsa*! This could lead to dire consequences.

Step number 3-Copying Public key

```
        [root@local]# cd ~/.ssh
```

There are a number of ways we can copy it accross as openssh also comes with sftp and scp. Before we do that one must understand the consequences of using passwordless ssh, if someone else has access to you local machine then they are going to have access to your remote host (if the person realises it). From a security perspective it is good practice to use a more restricted user than root, so at least there is some damage control. So lets create a user on remote (you can skip this step if you still want to use root but I don't recommend it)

```
    [root@remote]# useradd -d /home/bob -c "SSH USER" bob    <---- create user bob
    [root@remote]# passwd bob                                <---- create your password for bob
    [root@remote]# su - bob                                  <---- change to bob
    [bob@remote]$ mkdir .ssh                                 <---- create place for public key
```

This would basically create the place for the public key. Now that we have bob we need to copy the public key from local to remote into */home/bob/.ssh* . I will show you two ways to copy the key across - once again it is personal preference.

```
    [root@local]# scp ~/.ssh/id_rsa.pub bob@remote:/home/bob/.ssh/
    bob@remote's password:
    id_rsa.pub       100% |*****************************|  235      00:00
```

-- OR --

```
    [root@local]# sftp bob@remote
    bob@remote's password:
    sftp> lcd ~/.ssh
    sftp> cd /home/bob/.ssh
    sftp> put id_rsa.pub
    Uploading id_rsa.pub to /home/bob/.ssh/id_rsa.pub
```

Once the *id_rsa.pub* has copied to */home/bob/.ssh/id_rsa.pub*, you need to place the key into the *.ssh/ authorized_key2* file. You can do this from the *.ssh* directory with: *cat ./id_rsa.pub >> ./authorized_keys2*

Step number 4-Setup Permissions

This is probably the trickiest part of passwordless ssh and this is where most people give up. Don't be discouraged this is the final step and we are almost there.

```
[root@remote]# cd /home/bob
```

Because of the nature of ssh the .ssh directory

```
[root@remote]# chmod -R 700 ./.ssh
```

Bob must own the file and directory as well

```
[root@remote]# chown -R bob ./.ssh
```

Here is the tricky part where most people get caught out

```
[root@remote]# ls -al
    drwx------   5 bob    bob       4096 May 12 13:46 .
    drwxr-xr-x   5 root   root      4096 May 12 13:45 ..     ------------> Note the 755 permission
    -rwx------   1 bob    bob         22 May 12 13:46 .bash_history
    -rwx------   1 bob    bob         24 May 12 13:45 .bash_logout
    -rwx------   1 bob    bob        191 May 12 13:45 .bash_profile
    -rwx------   1 bob    bob        124 May 12 13:45 .bashrc
    -rwx------   1 bob    bob        120 May 12 13:45 .gtkrc
    drwx------   2 bob    bob       4096 May 12 13:53 .ssh
```

This 755 permission is a permission of the directory below *-/home*. The directory below must be set at 755 or better or passwordless ssh will not work. And that's it! You are now ready to ssh passwordless onto remote from local. Here is a quick test

```
[root@local]# ssh bob@remote ls -al
```

if the directory listing shows up without a password prompt then you have done it! Well done!

Here's a final tip to leave you with you troubleshoot with this command to see the steps that ssh takes

```
[root@local]# ssh -vv bob@remote        <----- the "-vv" shows verbose connection
```