

# SSH: THE SECURE SHELL

What is ssh:

SSH is a cryptographic network protocol for secure network services

uses:

- It is used for the remote login
- Secure File Transfer (SFTP/SCP)
- Port Forwarding
- SOCKS protocols for web browsing through encrypted proxy
- Secure remote file mounting via SSHFS

login with ssh using password:

---

requirements:

- we have two server
  - 1) server1, ip:192.168.0.10/24
  - 2) server2, ip :192.168.0.11/24

1) step1

we need to install the openssh-server in server2[in centos server its actually pre-installed]

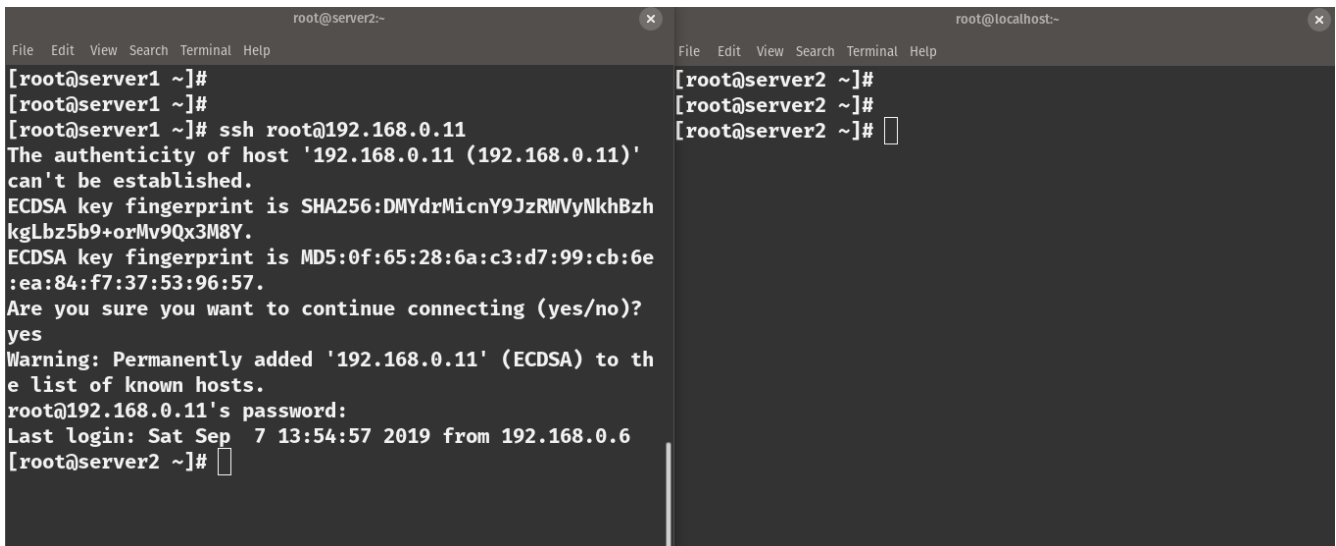
=>yum update -y

=>yum install sshd -y

2) from server1 use the command and give the password

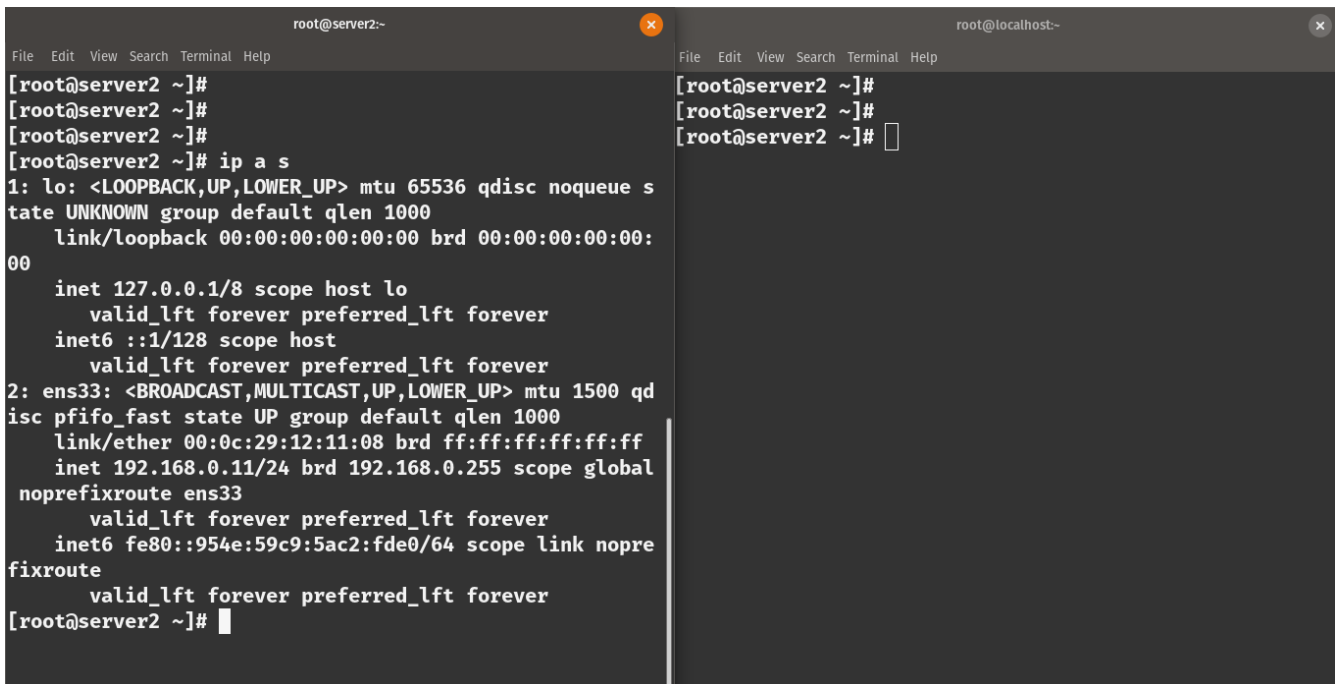
=>ssh [root@192.168.0.11](ssh://root@192.168.0.11)

password: <server2 password>

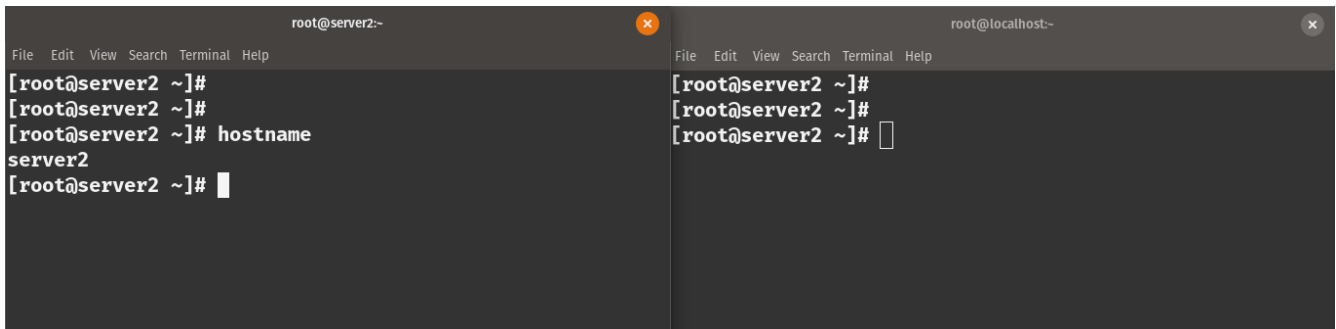


```
root@server2:~  
File Edit View Search Terminal Help  
[root@server1 ~]#  
[root@server1 ~]#  
[root@server1 ~]# ssh root@192.168.0.11  
The authenticity of host '192.168.0.11 (192.168.0.11)'  
can't be established.  
ECDSA key fingerprint is SHA256:DMYdrMicnY9JzRWVYNkhBzh  
kgLbz5b9+orMv9Qx3M8Y.  
ECDSA key fingerprint is MD5:0f:65:28:6a:c3:d7:99:cb:6e  
:ea:84:f7:37:53:96:57.  
Are you sure you want to continue connecting (yes/no)?  
yes  
Warning: Permanently added '192.168.0.11' (ECDSA) to th  
e list of known hosts.  
root@192.168.0.11's password:  
Last login: Sat Sep  7 13:54:57 2019 from 192.168.0.6  
[root@server2 ~]#  
  
root@localhost:~  
File Edit View Search Terminal Help  
[root@server2 ~]#  
[root@server2 ~]#  
[root@server2 ~]#
```

now you are logged in in server 2



```
root@server2:~  
File Edit View Search Terminal Help  
[root@server2 ~]#  
[root@server2 ~]#  
[root@server2 ~]#  
[root@server2 ~]# ip a s  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue s  
tate UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:  
00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qd  
isc pfifo_fast state UP group default qlen 1000  
    link/ether 00:0c:29:12:11:08 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.0.11/24 brd 192.168.0.255 scope global  
noprofixroute ens33  
        valid_lft forever preferred_lft forever  
    inet6 fe80::954e:59c9:5ac2:fde0/64 scope link nopre  
fixroute  
        valid_lft forever preferred_lft forever  
[root@server2 ~]#  
  
root@localhost:~  
File Edit View Search Terminal Help  
[root@server2 ~]#  
[root@server2 ~]#  
[root@server2 ~]#
```



```
root@server2~  
File Edit View Search Terminal Help  
[root@server2 ~]#  
[root@server2 ~]#  
[root@server2 ~]# hostname  
server2  
[root@server2 ~]#  
  
root@localhost~  
File Edit View Search Terminal Help  
[root@server2 ~]#  
[root@server2 ~]#  
[root@server2 ~]#
```

## login with ssh without password(more secure way):

using password to login with ssh is one way but it is not very secure the other way is to use a private and public key pair. we use a public private key pair for login rather than a password.

Step 1:

see if there is an existing key  
=> **ls -l ~/.ssh**

Step 2:

Create the key pair from server1

[syntax:ssh-keygen -t <algorithm> -b <size>]

=>**ssh-keygen -t rsa -b 4096**

```

[root@server1 ~]#
[root@server1 ~]#
[root@server1 ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:RQGTsHDt7SkKJFZX9ZbQ0lrjEkE3qi+IfpZwvErq2ng root@server1
The key's randomart image is:
+---[RSA 4096]-----+
  . o++=B=o          |
  .o..ooo==o         |
  . ... .o*+.        |
  o .   .o+..        |
  . o .   S. o        |
    o.o...o          |
    o+.+....         |
  oE+  *   .         |
o++ o+              |
+---[SHA256]-----+
[root@server1 ~]# 

```

[it will ask you for a passphrase for now we skip it we will discuss it later]

Step 3:

we need to send the public key to ther server2.we can do it manually or we can do it using this command

=>ssh-copy-id server2@192.168.0.11

```
[root@server1 ~]#
[root@server1 ~]#
[root@server1 ~]# ssh-copy-id root@192.168.0.11
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/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 the new keys
root@192.168.0.11's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'root@192.168.0.11'"
and check to make sure that only the key(s) you wanted were added.

[root@server1 ~]# ssh root@192.168.0.11
Last login: Sat Sep  7 15:29:15 2019 from 192.168.0.10
[root@server2 ~]#
```

Step 3:

login with

=>ssh root@192.168.0.11

and this time no password will be asked.

[

### **what is passphrase:**

sometime the ssh connectivity is used by you sometimes not. for example you can make a cron job to connect automatically to a server for data backup. when you are going to use the ssh only its a good idea to use a passphrase .but for automation you should not use it cause there will be no one to type the passphrase .when you use a script to automatically connect to a server don't use any passphrase.

]

### **copy file with SCP(Secure copy and paste):**

syntax:

**scp <local\_file> <destination>**

we are going to send a file name '**test.txt**' from server1 to server2

**=>scp test.txt 192.168.0.11/test.txt**

```
root@server2:~
File Edit View Search Terminal Help
[root@server2 ~]#
[root@server2 ~]#
[root@server2 ~]#
[root@server2 ~]# touch test.txt
[root@server2 ~]# echo "hello" > test.txt
[root@server2 ~]# scp test.txt 192.168.0.11
[root@server2 ~]#

root@localhost:~
File Edit View Search Terminal Help
[root@server2 ~]#
[root@server2 ~]#
[root@server2 ~]#
[root@server2 ~]# ls
192.168.0.11 anaconda-ks.cfg test.txt
[root@server2 ~]# cat test.txt
hello
[root@server2 ~]#
```

## Copy file with SFTP(Secure File Transfer Protocol):

its a interactive process for sending file over SSH. its a sub system for ssh

=>sftp 192.168.0.10

sftp> cd /etc

[go to etc directory]

sftp> get redhat-release

[download the file]

```
[root@server2 ~]#
[root@server2 ~]#
[root@server2 ~]#
[root@server2 ~]# em -rf redhat-release
-bash: em: command not found
[root@server2 ~]# rm -rf redhat-release
[root@server2 ~]# ls
192.168.0.11 anaconda-ks.cfg test.txt
[root@server2 ~]# sftp 192.168.0.11
root@192.168.0.11's password:
Connected to 192.168.0.11.
sftp> ls
192.168.0.11 anaconda-ks.cfg test.txt
sftp> cd /etc
sftp> get redhat-release
Fetching /etc/redhat-release to redhat-release
/etc/redhat-release 100% 38 18.5KB/s 00:00
sftp> ^D
[root@server2 ~]#
[root@server2 ~]# cat redhat-release
CentOS Linux release 7.6.1810 (Core)
[root@server2 ~]#
```

## Port Forwarding:

Port forwarding allows us to access from one system to another system and use their network services .for exmple you are running a web server in the server2 in port 80.you can access it with a browser or see the html using this command in server2

=>**curl localhost**

```
[root@server2 ~]# curl localhost
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"><html><head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Apache HTTP Server Test Page powered by CentOS</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <!-- Bootstrap -->
    <link href="/noindex/css/bootstrap.min.css" rel="stylesheet">
    <link rel="stylesheet" href="/noindex/css/open-sans.css" type="text/css" />

<style type="text/css"><!--
body {
    font-family: "Open Sans", Helvetica, sans-serif;
    font-weight: 100;
    color: #ccc;
    background: rgba(10, 24, 55, 1);
    font-size: 16px;
}

h2, h3, h4 {
    font-weight: 200;
}
```

but you cant browse it with the server1using curl .you have to do port forwarding to established that connection.



```
[root@server1 ~]#  
[root@server1 ~]# curl 192.168.0.11  
curl: (7) Failed connect to 192.168.0.11:80; No route to host  
[root@server1 ~]# █
```

So if we forward the port 80 of the server2 to port 8000 in server1 we can access the content of the web server in server2 with server1 in port 8000

command from server1:

=>ssh -L 8000:localhost:80 root@192.168.0.11

```
[root@server2 ~]# ssh -L 8000:localhost:80 root@192.168.0.10  
The authenticity of host '192.168.0.10 (192.168.0.10)' can't be established.  
ECDSA key fingerprint is SHA256:Vb8jzXFWtxe/Z7yco6NR2IPPJ+1uotVhlseVEx+/e2o.  
ECDSA key fingerprint is MD5:bd:62:cb:ab:28:3b:ad:47:61:da:b5:8f:d8:b6:85:4c.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '192.168.0.10' (ECDSA) to the list of known hosts.  
root@192.168.0.10's password:  
Last login: Sat Sep  7 16:44:29 2019 from 192.168.0.6  
[root@server1 ~]# █
```

it will forward the port and we can access the resources from server1. it can be very useful for accessing a file that is behind a firewall.

```
[root@server1 ~]#  
[root@server1 ~]# curl localhost:8000  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"><html><head>  
<meta http-equiv="content-type" content="text/html; charset=UTF-8">  
    <title>Apache HTTP Server Test Page powered by CentOS</title>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
  
    <!-- Bootstrap -->  
    <link href="/noindex/css/bootstrap.min.css" rel="stylesheet">  
    <link rel="stylesheet" href="noindex/css/open-sans.css" type="text/css" />  
  
<style type="text/css"><!--
```

## Copy configuration :

ssh server and configuration file is in the ‘/etc/ssh/’ directory.

- 1) ‘**sshd\_config**’ is the ssh server configuration file
- 2) ‘**ssh\_config**’ is the ssh client configuration file

Lets see the server configuration file and important propertise

**vim /etc/ssh/sshd\_server**

---

PasswordAuthentication yes

Port 22

PubkeyAuthentication yes

X11Forwarding yes

# PermitRootLogin no

---

→ you can change the port from 22 to any port you want but default is 22

```
# If you want to change the port on a SELinux system, you have to tell
# SELinux about this change.
# semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
#
#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

→ password authentication is set to no for some cloud server  
Because the use public private key pair which is more secure

→ X11 forwarding is by default set to yes. if you want to  
work with a gui interface this will let you do this

```
#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PermitTTY yes
#PrintMotd yes
#PrintLastLog yes
```

→ Permit root login is set to no. It should be always set to no because root login can make major security risk

```
#LoginGraceTime 2m
#PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes
```