SSH Command

SSH is short for "*Secure Shell*". SSH can be defined as a protocol that is used for securely connecting to a remote system or server. It is secure because it can transfer the information in encrypted format among the client and the host.

It also transfers inputs through the client to a host. This command executes at the **22th port** of the *TCP/IP*.

Syntax of the SSH command:

1. ssh user_name@hist(IP/Domain_name)

Prerequisites of the SSH Command

Before we can create a secure Remote Desktop protocol using a remote machine, here are some command requirements to match:

- ❍The remote computer should be turned on at each time and has a network connection.
- ❍The server and client applications have to be enabled and installed.
- ❍We require the name or the IP address of the remote machine we wish to connect to.
- ❍We require to have mandatory permissions for accessing the remote system.
- ❍Various firewall settings are required to permit the remote connection.

Parts of the SSH Command

SSH command in Linux includes 3 distinct parts which are listed and discussed below:

- ❍*ssh command* guides the system for establishing the encrypted secure connection to the host machine.
- ❍*user_name* illustrates the account which is being authenticated over the host.
- ❍The *host* can be defined as a machine which could be a router or a computer that is being authenticated. It could be a domain name *(e.g. www.domainname.com)* or an IP address *(e.g. 192.168.1.124)*.

Note: The commands will implement as when they were specified to the host terminal directly after logging into a host computer. Using an SSH key pair or a public-private key pair for logging into a remote host is more protected than using the passwords.

To generate the public-private key, we can use the following command:

1. ssh-keygen

A private key should remain hidden at the time the public key should be copied to a remote host.

The connection would be created not with the password but with the help of an SSH key after copying a public key to a remote host.

Options in SSH Command

There are various options in the SSH command of Linux. These options are explained as follows:

1. **-1:** It is used for forcing the ssh command to apply only the SSH-1 protocol.
2. **-2:** It is used for forcing the ssh command to apply only the SSH-2 protocol.
3. **-4:** It is used for allowing only IPv4 addresses.
4. **-6:** It is used for allowing only IPv6 addresses.
5. **-a:** It is used for enabling the authentication agent connection forwarding procedure.
6. **-A:** It is used for disabling the authentication agent connection forwarding procedure.
7. **-C:** It is used for compressing every data (including stderr, stdout, stdin, and data for TCP and X11 forwarded connections) for a faster data transfer.
8. **-f:** It is used to request the ssh command for going to the background only before the execution of the command.
9. **-g:** It is used for allowing various remote hosts to link to the local forwarded ports.
10. **-p port_number:** It is a port for connecting to over the remote host.
11. **-n:** It is used for preventing the reading from the stdin.

12.**-q:** It is used for suppressing every warning and error.

13.**-V:** It is used for displaying the number of the version.

14.**-v:** It stands for Verbose mode and it is used for echoing everything it is operating at the time of creating any connection. This option is very helpful in connection failures debugging.

15.**-X:** It is used for enabling the X11 forwarding or GUI Forwarding.

16.**-c cipher_spec:** It is used for selecting the cipher specification to encrypt the session. A particular cipher algorithm would be selected only when both the server and the client support it.

Significantly, **ssh** is more secure as compared to other various protocols like telnet due to the data encryption.

There are 3 widely used techniques of encryption which are used by the SSH protocol are as follows:

1. Symmetrical encryption: Symmetrical encryption operates over the principles of the production of a single key to encrypt and data decryption as well. The secret key produced is sharing between the hosts and the clients for a more secure connection. This encryption technique is the most common and works best if data is decrypted and encrypted over an individual machine.

2. Asymmetrical encryption: Asymmetrical encryption technique is more secure due to it produces two distinct keys which are listed below:

❍Public key

❍Private key
A public key is shared to distinct types of host machines during the private key are over the client machine securely. Any secure connection is created with the help of the public-private pair of the key.

3. Hashing: In this method, one-way hashing can be defined as an authentication method that makes sure that the data is unchanged and comes through the genuine sender.

The hash function can be used for generating the hash code through the data. It's possible to re-produce the data through a hash value.

This value is measured at the side of the sender and the end of the receiver as well. The data is accurate if the hash values are the same.

Working of SSH

We need two important components for establishing the SSH connection. These components are as follows:

❍SSH Client Component

❍SSH Server-side Component

1. SSH Client Component: The SSH client can be defined as an application we install on our computer which we will use for connecting to a server or another computer.

The client will use the given remote host data for initiating the connection. It also establishes an encrypted connection when the credentials are authenticated.

2. SSH Server-side Component: On the side of a server, there is an element known as SSH daemon. Constantly, it is listening to a particular TCP/IP port for feasible requests of client connection.

The SSH daemon responds with the protocol versions and software and these two will swap their identification information.

The SSH establishes a newer session for an appropriate environment if the given credentials are accurate.

The default version of SSH protocol for SSH client and SSH server communication is the 2 version.

How to implement an SSH Connection

Since establishing the SSH connection needs both a server and a client component, we need to ensure that they have been respectively installed over the remote and local machine.

The OpenSSH tool is widely used for various Linux distributions. It is an open-source tool of SSH and easy to install relatively. It needs access to our terminal over the computer and server that we use for the connection.

Note: By default, Ubuntu doesn't have any SSH server which is installed.

OpenSSH Client Installation

Before we proceed to install the SSH client, ensure that it's not installed already. Several distributions of Linux already contain an SSH client. We can also install PuTTY or other clients of our choice for gaining access to any server for Windows machines.

For checking if a client has existed on our Linux based systems, we are required to:

1.Load the SSH terminal. We can either click ***CTRL+ALT+T*** or find for "***terminal***" on our keyboard.

2.Types inside ssh and click ***Enter*** within the terminal.

3.We will get a response when the client has existed.

It means that we are ready for connecting to any virtual or physical machine remotely. Otherwise, we will need to install an OpenSSH client.

1.Execute the below command for installing the OpenSSH client on our computer.
sudo apt-get install openssh-client

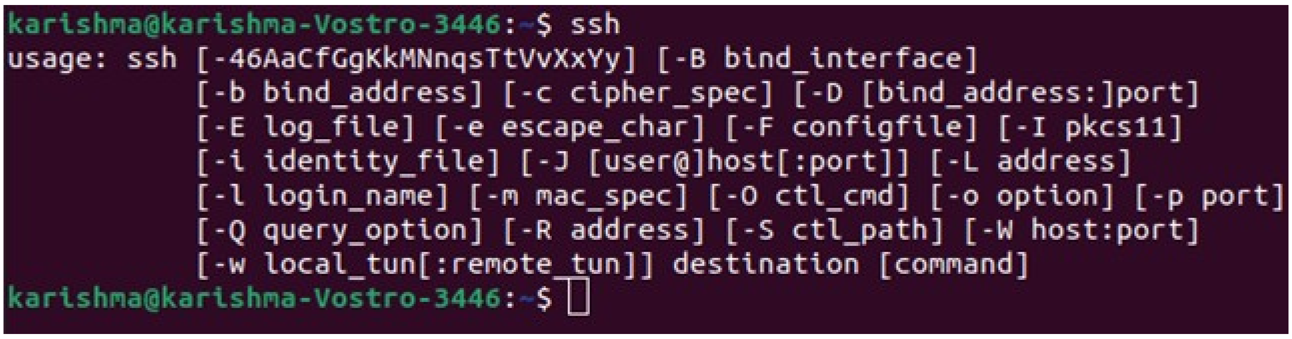
2.Input inside our superuser password if asked.

3.Press the Enter button to finish the installation process.

Now, we can access SSH on any machine along with the server-side application over it, given that we require the important privileges for **gaining access, IP address,** and **hostname** as well.

We can also check that the ssh is installed on our system or not by entering the below command:

1. ssh



```
karishma@karishma-Vostro-3446:~$ 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]
karishma@karishma-Vostro-3446:~$ 
```

OpenSSH Server Installation

A machine requires to include the server-side segment of an SSH software toolkit for accepting SSH connections.

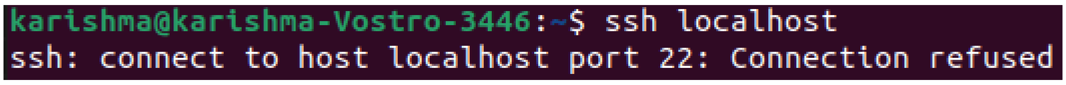
If we first wish to check that the OpenSSH server has existed on Ubuntu of a remote computer that requires to accept the SSH connections, we can try connecting to localhost:

1.On the server machine, open our terminal. We can either find "***terminal***" or click ***CTRL+ALT+T*** on our keyboard.

2.Type inside the ssh localhost and press Enter.

3.The response would seem like below for those systems without an SSH server installed:

```
1. username@host:~$ ssh localhost
2.
3. ssh: connect to host localhost port22: Connection refused
4.
5. username@host:~$
```



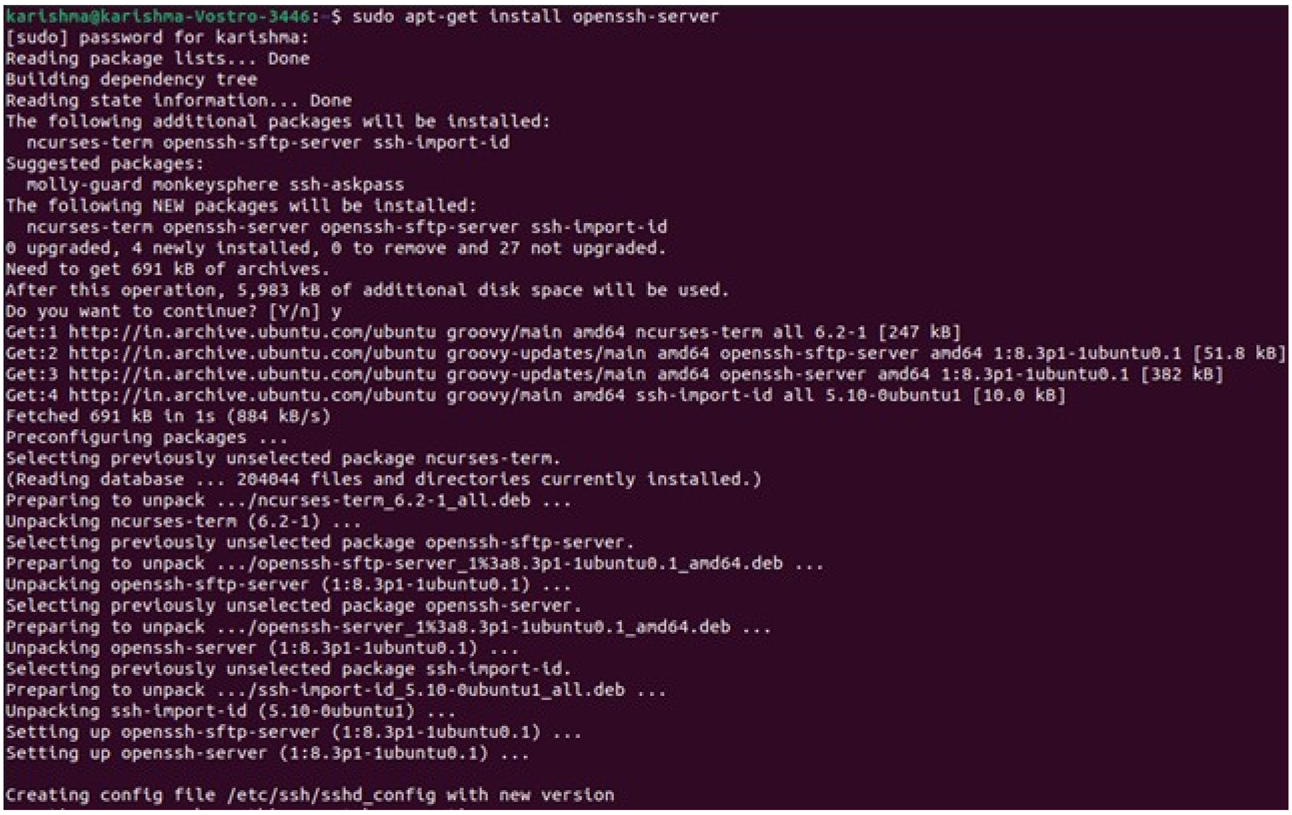
```
karishma@karishma-Vostro-3446:~$ ssh localhost
ssh: connect to host localhost port 22: Connection refused
```

We will have to install an OpenSSH server when the above is the situation. Open the terminal and:

1.Execute the below command for install an SSH server: sudo apt-get install openssh-server

2.Input inside our superuser password if asked.

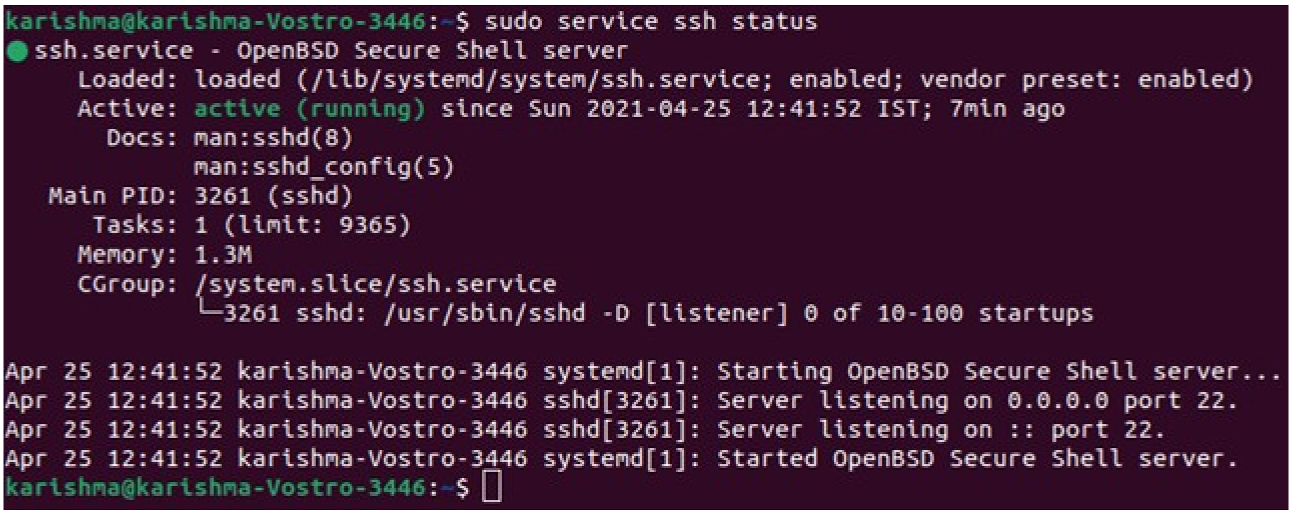
3.Press ***Enter*** and ***Y*** for allowing the installation process to proceed later with the disk space prompt.



```
karishma@karishma-Vostro-3446:~$ sudo apt-get install openssh-server
[sudo] password for karishma:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-sftp-server ssh-import-id
Suggested packages:
  molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 4 newly installed, 0 to remove and 27 not upgraded.
Need to get 691 kB of archives.
After this operation, 5,983 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu groovy/main amd64 ncurses-term all 6.2-1 [247 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu groovy-updates/main amd64 openssh-sftp-server amd64 1:8.3p1-1ubuntu0.1 [51.8 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu groovy-updates/main amd64 openssh-server amd64 1:8.3p1-1ubuntu0.1 [382 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu groovy/main amd64 ssh-import-id all 5.10-0ubuntu1 [10.0 kB]
Fetched 691 kB in 1s (884 kB/s)
Preconfiguring packages ...
Selecting previously unselected package ncurses-term.
(Reading database ... 204044 files and directories currently installed.)
Preparing to unpack .../ncurses-term_6.2-1_all.deb ...
Unpacking ncurses-term (6.2-1) ...
Selecting previously unselected package openssh-sftp-server.
Preparing to unpack .../openssh-sftp-server_1%3a8.3p1-1ubuntu0.1_amd64.deb ...
Unpacking openssh-sftp-server (1:8.3p1-1ubuntu0.1) ...
Selecting previously unselected package openssh-server.
Preparing to unpack .../openssh-server_1%3a8.3p1-1ubuntu0.1_amd64.deb ...
Unpacking openssh-server (1:8.3p1-1ubuntu0.1) ...
Selecting previously unselected package ssh-import-id.
Preparing to unpack .../ssh-import-id_5.10-0ubuntu1_all.deb ...
Unpacking ssh-import-id (5.10-0ubuntu1) ...
Setting up openssh-sftp-server (1:8.3p1-1ubuntu0.1) ...
Setting up openssh-server (1:8.3p1-1ubuntu0.1) ...

Creating config file /etc/ssh/sshd_config with new version
```

The necessary support files would be installed, then we can inspect if an SSH server is executing over the machine by using the following command:

```
1. sudo service ssh status
```

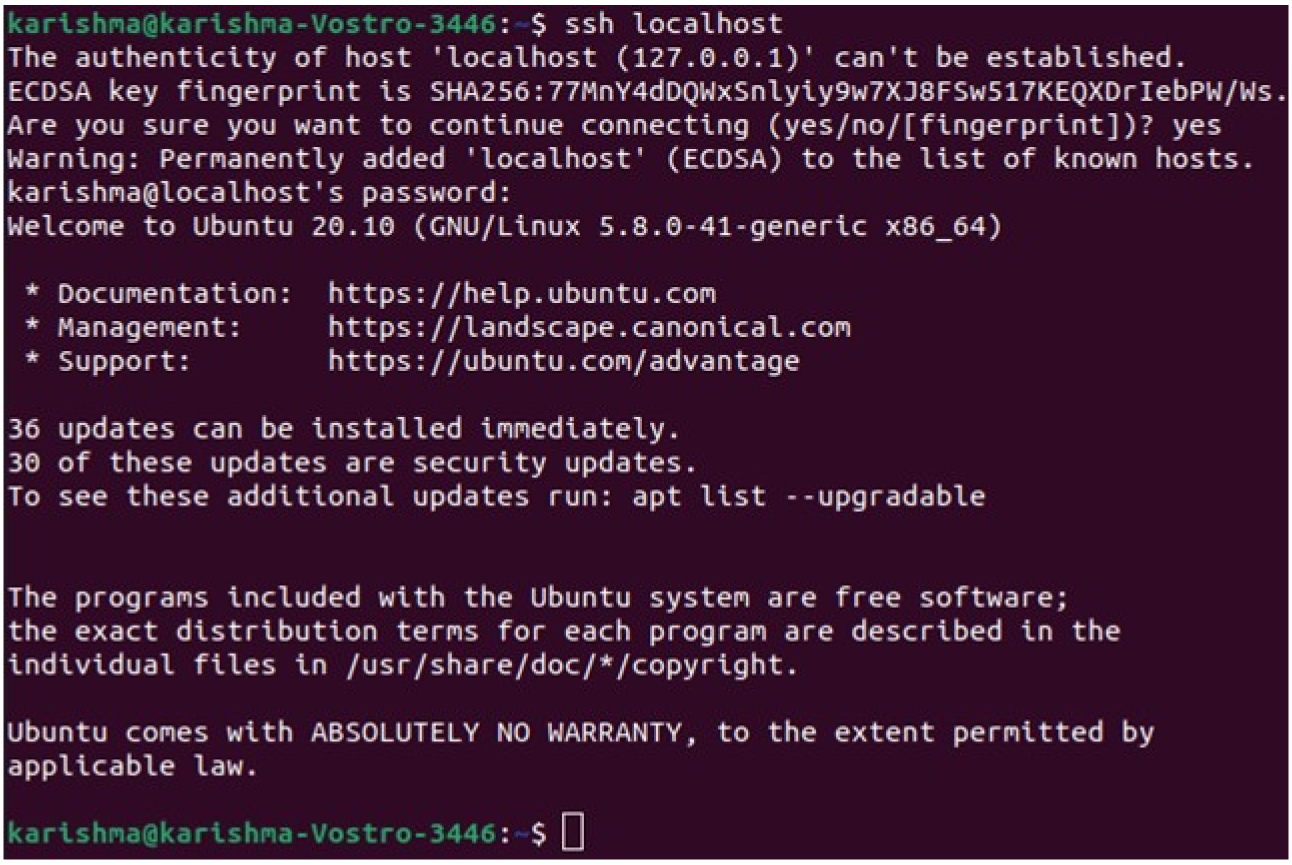


```
karishma@karishma-Vostro-3446:~$ sudo service ssh status
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2021-04-25 12:41:52 IST; 7min ago
       Docs: man:sshd(8)
             man:sshd_config(5)
   Main PID: 3261 (sshd)
      Tasks: 1 (limit: 9365)
     Memory: 1.3M
     CGroup: /system.slice/ssh.service
             └─3261 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Apr 25 12:41:52 karishma-Vostro-3446 systemd[1]: Starting OpenBSD Secure Shell server...
Apr 25 12:41:52 karishma-Vostro-3446 sshd[3261]: Server listening on 0.0.0.0 port 22.
Apr 25 12:41:52 karishma-Vostro-3446 sshd[3261]: Server listening on :: port 22.
Apr 25 12:41:52 karishma-Vostro-3446 systemd[1]: Started OpenBSD Secure Shell server.
karishma@karishma-Vostro-3446:~$ 
```

In the terminal, the response should seem the same as this when the SSH service is executing.

Other ways for testing the OpenSSH server is properly installed or not is to accept the connections for trying executing the command, i.e., ***ssh localhost*** again in our terminal prompt. The outcome will appear as:

Enter ***y*** or ***yes*** to continue.



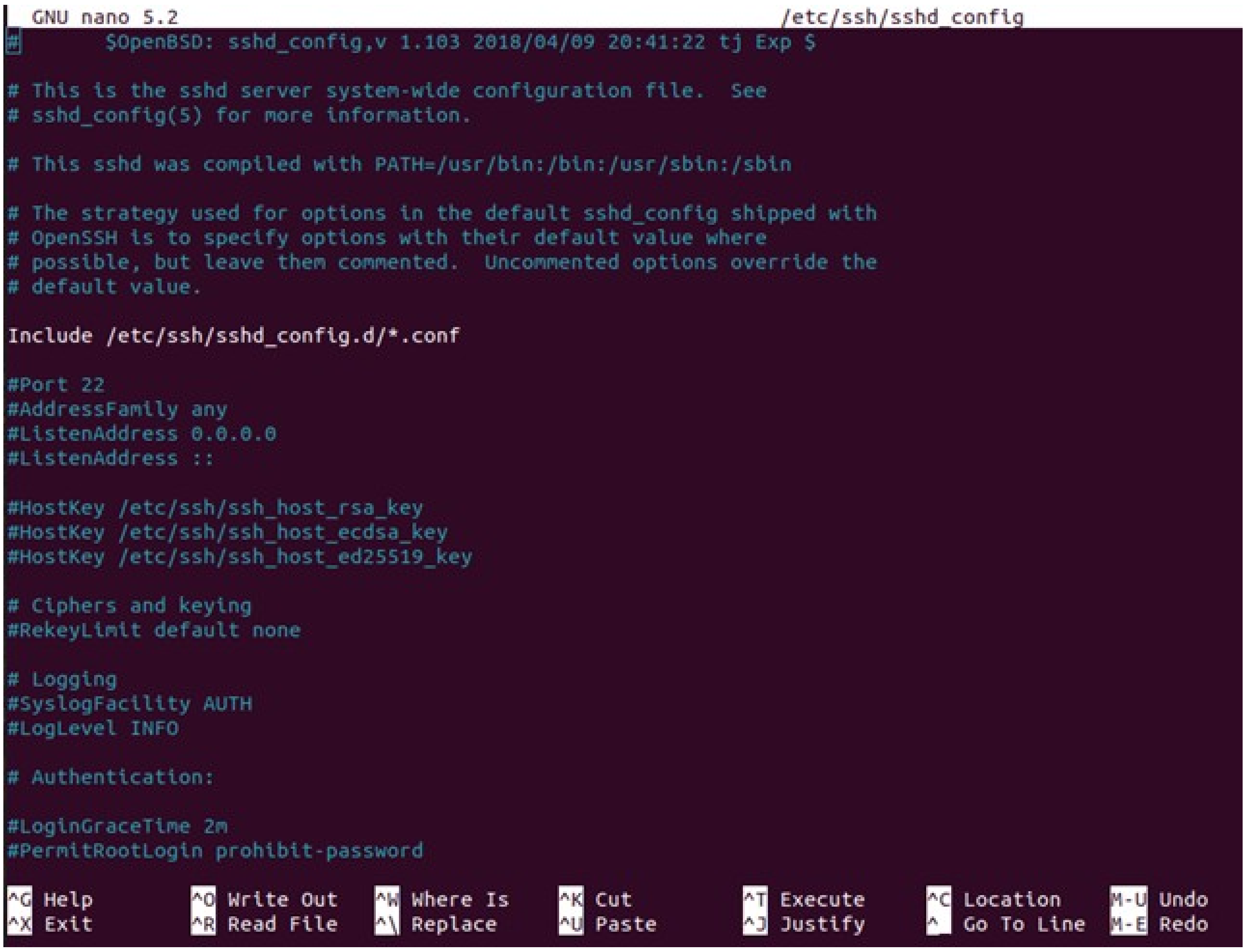
```
karishma@karishma-Vostro-3446:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:77MnY4dDQWxSnlyiy9w7XJ8FSw517KEQXDrIebPW/Ws.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
karishma@localhost's password:
Welcome to Ubuntu 20.10 (GNU/Linux 5.8.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

36 updates can be installed immediately.
30 of these updates are security updates.
To see these additional updates run: apt list --upgradable


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

karishma@karishma-Vostro-3446:~$ 
```

Now, we have set up our server for accepting the requests of the SSH connection from a distinct computer with the help of the SSH client.

Important Tip

Now, we can edit the configuration file of the SSH daemon. For example, we can modify the port (default) for various SSH connections. We can execute the following command inside the terminal prompt:

1. sudo nano /etc/ssh/sshd_config



```
  GNU nano 5.2                                        /etc/ssh/sshd_config
#       $OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^  Go To Line  M-E Redo
```

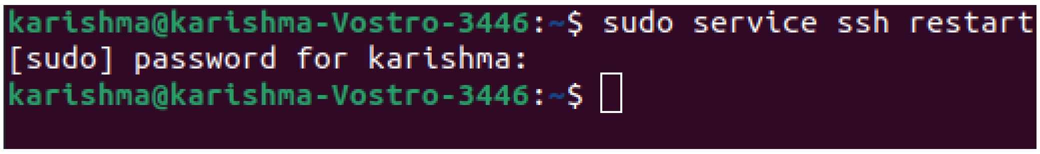
The configuration file would open inside the editor of our choice. We can use nano in this type of case.

We can run the following command if we have to install Nano:

1. sudo apt-get install nano

> Note: We have to reboot the SSH service all the time we create any modifications to the file, i.e., sshd_config by executing the following command:

1. sudo service ssh restart



```
karishma@karishma-Vostro-3446:~$ sudo service ssh restart
[sudo] password for karishma:
karishma@karishma-Vostro-3446:~$ 
```

Connection via SSH

Now, we have the OpenSSH server and client installed on all the machines we need. We can create a remote connection securely with our servers. For doing so:

1. Start the SSH terminal prompt on our machine and execute the below command:

1. ssh your_username@host_ip_address
   When the username on our local machine is the same as the one over the server we are trying to link to, we can only type:

1. ssh host_ip_address
   After that press the **'Enter'** button.

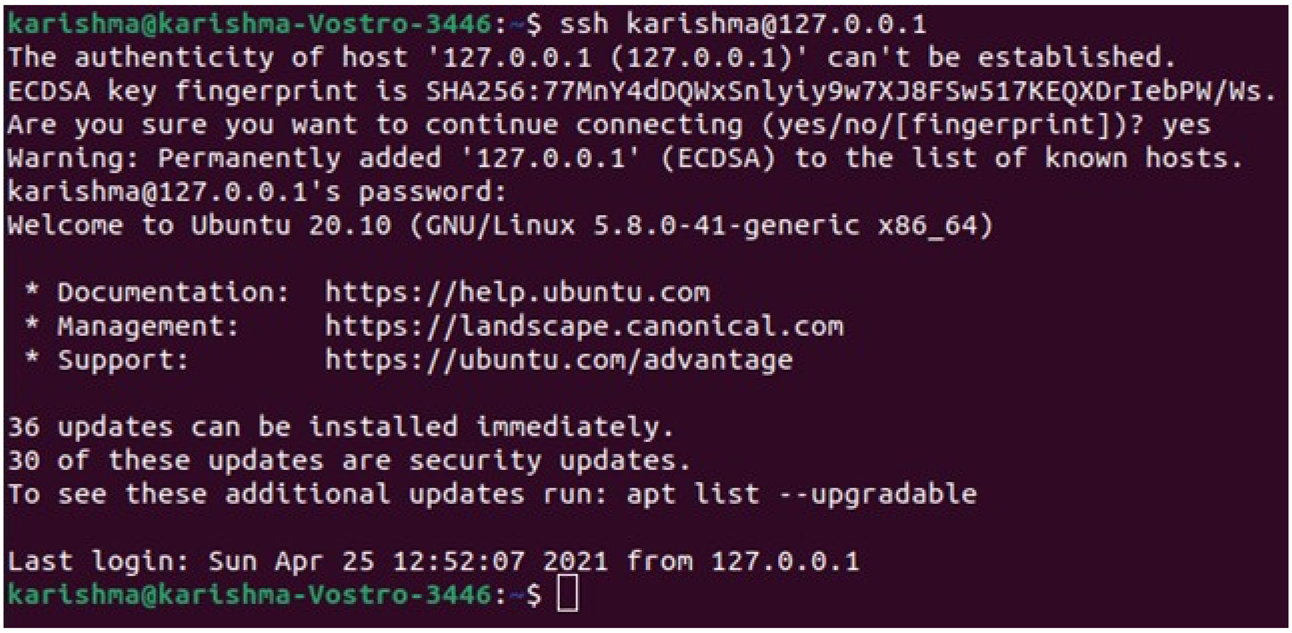
2. Type inside our password and press the **Enter** button.

> Note: We will not receive any feedback over the screen during typing. If we are pasting our password, ensure that it is safely stored and not in any text file.

3. If we are connecting to any server for the first time, it will confirm us if we wish to continue connecting. We have to only type yes and press the **Enter** button.

This information occurs this time only since a remote server has not been recognized on our local machine.

4. Now, an ECDSA key fingerprint is included and we are connected to a remote server.



```
karishma@karishma-Vostro-3446:~$ ssh karishma@127.0.0.1
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:77MnY4dDQWxSnlyiy9w7XJ8FSw517KEQXDrIebPW/Ws.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '127.0.0.1' (ECDSA) to the list of known hosts.
karishma@127.0.0.1's password:
Welcome to Ubuntu 20.10 (GNU/Linux 5.8.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

36 updates can be installed immediately.
30 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Last login: Sun Apr 25 12:52:07 2021 from 127.0.0.1
karishma@karishma-Vostro-3446:~$ 
```

When the computer we are trying to connect to remotely is on a similar network, it's best to use a private IP address rather than a public IP address. We will otherwise need to use only a public IP address.

Additionally, ensure that we know the accurate TCP port OpenSSH understands the connection claims and the various settings of port forwarding are accurate.

Port 22 is the default when nobody modified the configuration inside the file, i.e., ***sshd_config***. Also, we may only append the port number later to the host IP address.

Now, we can control and manage the remote machine with the help of our terminal. If we have any trouble connecting to the remote server, we need to ensure that:

❍The remote machine IP address is accurate.

❍The port SSH daemon is receiving is not blocked through a firewall or incorrectly forwarded.

❍Our password and user name are accurate.

❍The SSH software is properly installed.