# SSH

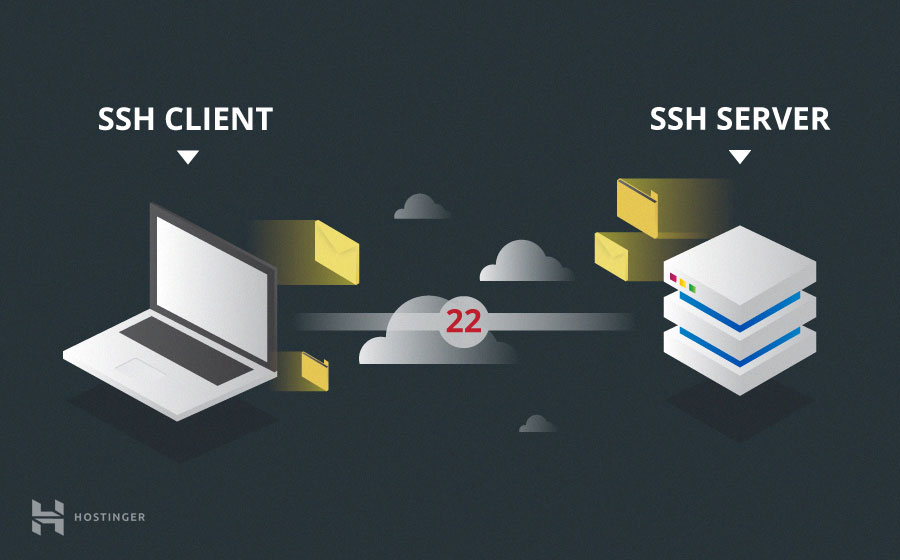
## What Is SSH?

SSH (Secure Shell) is a network communication protocol that enables two or more computers to communicate and share data with each other over the Internet.

## How Does SSH Work?

SSH works by making use of a **client-server model** to allow for **authentication** of two remote systems and **encryption** of the data that passes between them.

By default, SSH operates on port 22. The host (server) listens on port 22 (or any other) for incoming connections. It organizes the secure connection by authenticating the client and opening the correct shell environment if the verification is successful.

**[](https://www.hostinger.com/tutorials/wp-content/uploads/sites/2/2017/07/ssh-client-and-server.jpg)**

The client must start the SSH connection by initiating the **TCP handshake** with the server, ensuring a secured symmetric connection, verifying whether the identity displayed by the server match previous records (typically recorded in an RSA key store file), and presenting the required user credentials to authenticate the connection.

There are two stages to establishing a connection:

* First, both the systems must agree upon encryption standards to protect future communications.
* Second, the user must authenticate themselves. If the credentials match, then the user is granted access.

## SSH Installation

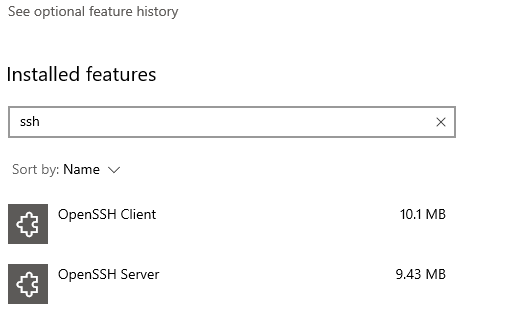
### On Debian / Ubuntu

1. On the server, run: $ sudo apt install openssh-server
2. On the client, run: $ sudo apt install openssh-client

### On Windows

From Settings > Apps > Optional Features, install:

1. On the server, OpenSSH Server.
2. On the client, OpenSSH Client.



## SSH Commands

The SSH command consists of 4 distinct parts:

$ ssh [options] <user-name>@<host-name-or-host-IP>

Where:

* user-name: The account you want to access. For example, you may want to access the root user, which is basically synonymous for system administrator with complete rights to modify anything on the system.
* host-name-or-host-IP: The computer you want to access. This can be an IP Address (e.g., 244.235.23.19) or a domain name (e.g., [www.xyzdomain.com](http://www.xyzdomain.com)).
* options: Additional settings you add to the command. For example:
  + -p <port-id>: Choose a TCP port
  + vvv: Choose level of verbose logs. The more "v" the more details.

Once you run the command, you will be prompted to enter the password for the requested account. If your password is correct, you will be greeted with a remote terminal window.

## Tips

### Open GUI app with SSH

**Warning:**

This method only works with **Linux or Mac GUI app**. This means it cannot help you open a Windows GUI app (e.g., Notepad) via SSH-ing to a Windows guest.

Reason [here](https://superuser.com/a/1297214) and [here](https://superuser.com/a/1168000).

SSH-ing with GUI sounds hard to achieve, but it's actually easy and there are many ways to do. The simplest could be:

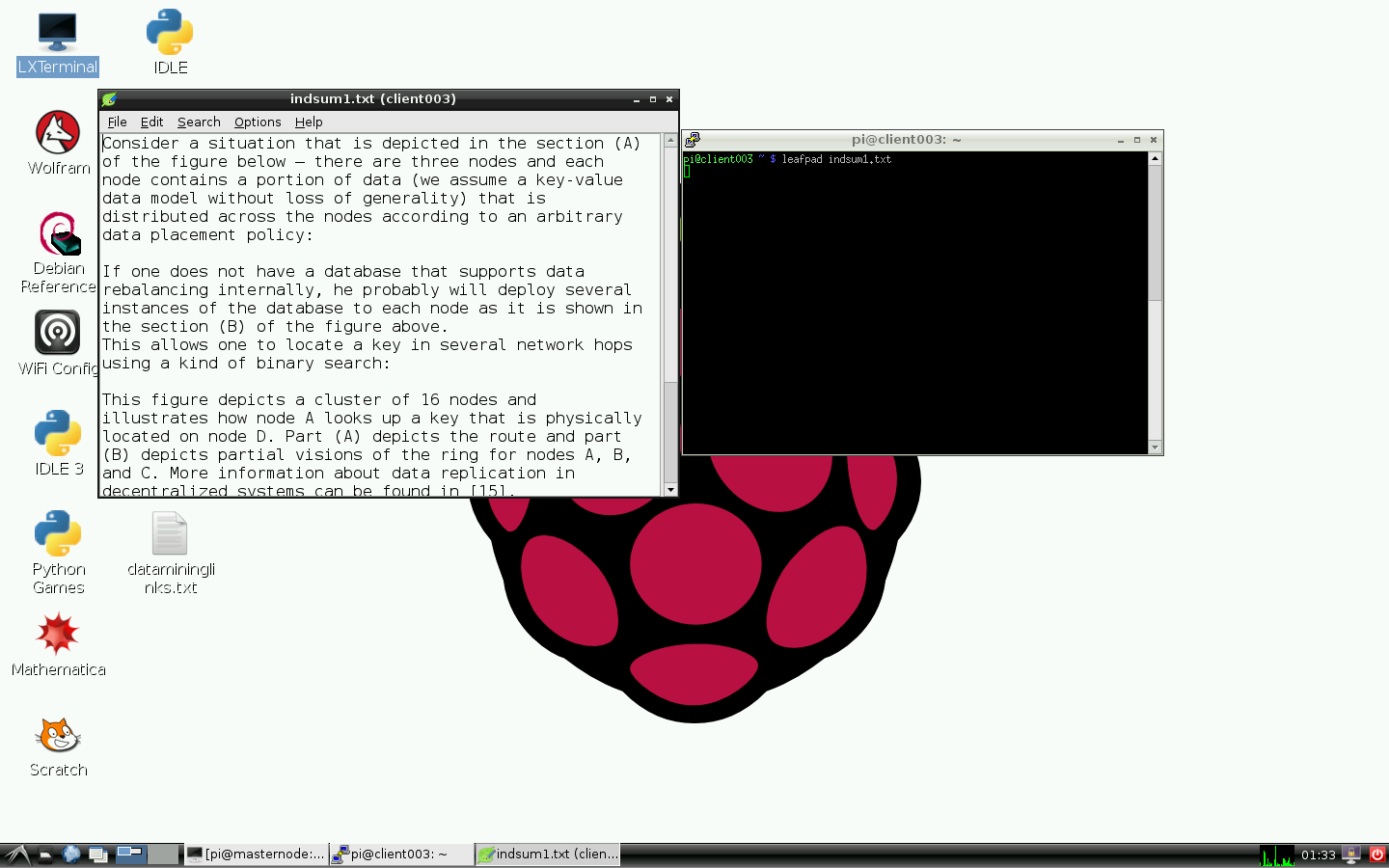
Step 1: Install a X server. For example, [Xming](https://sourceforge.net/projects/xming/) on Windows.

Step 2: Run the X server. Usually, it will run on background.

Step 3: SSH to the Linux/Mac guest machine with the X forwarding option:

* If using command line – run: ssh -X <guest-username>:<guest-hostname-or-ip>
* If using Putty: From the main windows, go to Connection > SSH > X11, and tick on Enable X11 forwarding. Then make a SSH connection to the guest machine.

Step 4: Once you accessed the guest, try opening an GUI app (e.g., gedit on Linux). You'll see the app opened and displayed with a full GUI window.



### Send command via SSH

From the host, you can control the guest by sending command to it. The guest will then execute the command.

If sending one command:

$ ssh <guest-username>:<guest-hostname-or-ip> "mkdir abc"

If sending multiple commands:

$ ssh <guest-username>:<guest-hostname-or-ip> "mkdir abc; cd abc; touch xyz.txt"

## Common Issues:

### Network error: Connection refused

There are many root causes for this issue.

1. Your SSH service is down

Check if SSH service is not started or down:

$ sudo service ssh status

If the output looks like "*Unit ssh.service could not be found.*", then SSH is down.

If so, simply start it:

$ sudo service ssh start

But if the output looks like "*Failed to start ssh.service: Unit ssh.service not found.*", then SSH Server or Client is not installed. You need to install it:

$ sudo apt install openssh-server

$ sudo apt install openssh-client

2. Check [here](https://kinsta.com/knowledgebase/ssh-connection-refused/).

### Warning: Remote host identification has changed!

<https://stackabuse.com/how-to-fix-warning-remote-host-identification-has-changed-on-mac-and-linux/>

### Permission denied (publickey)

Maybe missing username. Try change "$ ssh <host-name-or-host-IP" to "$ ssh <user-name>@<host-name-or-host-IP".

# IPs (Internet Protocols)

## TCP (Transmission Control Protocol)

### What Is TCP?

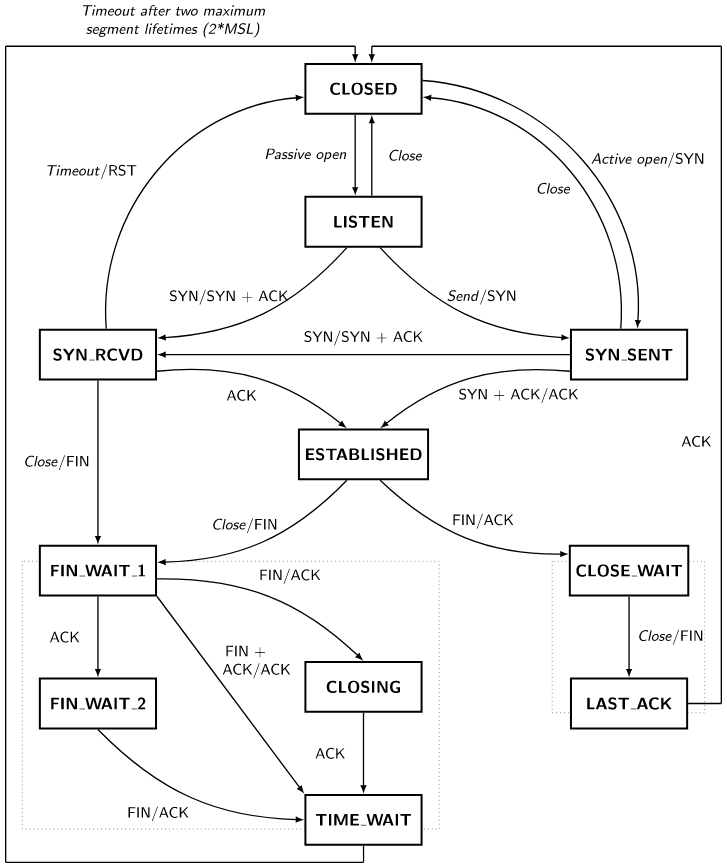
TCP is the **most commonly used protocol** for streaming over the Internet.

It **requires an acknowledgement from the receiver before sending the next packet**. If the delivery fails, it will continue retrying until it successfully delivers the message.

However, TCP requires a handshake and active connection between the sender and the receiver, which uses additional network resources.

### States

TCP has 11 states. The following UML state diagram describes all of these states:



Explanation: <https://www.sobyte.net/post/2021-12/whys-the-design-tcp-time-wait/>

Amongst, the LISTEN, ESTABLISHED, TIME\_WAIT and CLOSED are probably the most common states we often see when inspecting TCP connections.

Example:

1. After TCP server created socket and before TCP client connects to the server:

$ netstat -a

Active Internet connections (w/o servers)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 localhost:60020 0.0.0.0:\* LISTEN

2. After TCP client connects to the server:

$ netstat -t

Active Internet connections (w/o servers)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 localhost:60020 localhost:37434 ESTABLISHED

3. After TCP client is disconnected with the server:

$ netstat -a

Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 localhost:47018 localhost:60020 TIME\_WAIT

4. After TCP client re-connects to the server:

$ netstat -t

Active Internet connections (w/o servers)

tcp 0 0 localhost:60020 localhost:47018 ESTABLISHED

**Notes**:

* The Foreign Address's port at step 2 and 4 are diferent. That’s because **each time a TCP client connects to the server, it chooses a random port**.
* The TIME\_WAIT state will last for seconds or minutes before it's changed to CLOSED.
* When using command netstat, be careful that:

$ netstat -t // Only show active TCP connections (TIME\_WAIT, ESTABLISHED, etc.).

// It doesn't show LISTEN state.

$ netstat -listen // Only show connections in LISTEN state

$ netstat -a | grep "tcp" // Show all active and inactive TCP connections. ALL!

*Details:*

<https://www.w3.org/People/Frystyk/thesis/TcpIp.html#TCP>

<https://hub.packtpub.com/understanding-network-port-numbers-tcp-udp-and-icmp-on-an-operating-system/>

<https://www.sobyte.net/post/2021-06/tcp-network-programming-best-practices/>

## UDP (User Datagram Protocol)

UDP **sends packages without guaranteeing** delivery or an acknowledgement of receipt. It makes a single attempt to send a packet, and if the delivery fails, it does not try again.

It’s much faster and uses fewer resources than other protocols, but should only be used on reliable networks such as localhost. UDP also doesn’t support encrypting logs.

*Details:*

<https://www.w3.org/People/Frystyk/thesis/TcpIp.html#UDP>

<https://hub.packtpub.com/understanding-network-port-numbers-tcp-udp-and-icmp-on-an-operating-system/>

*The table below is a comparison between TCP and UDP:*

|  |  |  |
| --- | --- | --- |
| **Basis** | **TCP** | **UDP** |
| Type of service | Connection-oriented protocol  (The communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data) | Datagram-oriented protocol  (There is no overhead for opening a connection, maintaining a connection, and close a connection) |
| Reliability | Reliable  (The delivery of data is guaranteed to the destination) | Not reliable  (Data is continuously sent to the receivers, whether or not they receive it) |
| Error checking mechanism | Provides extensive error-checking mechanisms  (flow control and acknowledgment of data) | Has only the basic error checking mechanism using checksums  (If the order is required, it has to be managed by the application layer) |
| Sequence | Packets arrive in order at the receiver. | There is no sequencing of data  (If the order is required, it has to be managed by the application layer) |
| Speed | Comparatively slower  (because of acknowledgement and bigger header length) | Faster, simpler  (because of no acknowledgement and smaller header length) |
| Retransmission | Retransmission of lost packets is possible | No retransmission of lost packets |
| Header length | 20-60 bytes | 8 bytes fixed-length |
| Weight | Heavy-weight | Lightweight |
| Handshaking techniques | SYN, ACK, SYN-ACK | No handshake |
| Use cases | Transferring data like images, files, web pages, etc. | Broadcasting, video streaming, multitask network transmission |
| Protocols | Used by HTTP, HTTPs, FTP, SMTP and Telnet. | Used by DNS, NTP, DHCP, TFTP, SNMP, RIP, VoIP |
| Stream type | Byte stream | Message stream |
| Overhead | Low but higher than UDP | Very low |

## ICMP (Internet Control Message Protocol)

ICMP has an entirely different function than TCP and UDP, as it's not a data packet transport protocol. It's a special type of packet used for **inter-device communication**, carrying everything from redirect instructions to timestamps for synchronization between devices.

What ICMP is probably best known for, however, is **echo requests**. This is pretty much what it sounds like. One device sends out an ICMP packet to another, telling the receiver to send a reply confirming it received the request. The receiver then responds with a new ICMP packet, the echo reply, confirming the request.

*Details:*

<https://hub.packtpub.com/understanding-network-port-numbers-tcp-udp-and-icmp-on-an-operating-system/>

# Telnet

## What Is Telnet?

Telnet is a terminal emulation program for TCP/IP networks that allows you to access another computer on the Internet or local area network by logging in to the remote system.

**Warning**

Telnet protocol is **unencrypted** and therefore insecure. It's NOT recommended to use Telnet on servers today. To manage server over the network, use SSH instead. Using a Telnet client is something different; it's actually useful to, e.g., test mail or web server.

## How Does Telnet Work?

By default, Telnet operates on port 23.

…

## Telnet Installation

### On Debian / Ubuntu

<https://www.howtoforge.com/how-to-install-and-use-telnet-on-ubuntu/>

### On Windows

## Telnet Commands

The Telnet command consists of 4 distinct parts:

$ telnet [options] <host-name-or-host-IP> <port>

Where:

* host-name-or-host-IP: The computer you want to access. This can be an IP Address (e.g., 244.235.23.19) or a domain name (e.g., [www.xyzdomain.com](http://www.xyzdomain.com)).
* options: Additional settings you add to the command. For example, …
* port: Communication port.

Once you run the command, you will be prompted to enter the password for the requested account. If your password is correct, you will be greeted with a remote terminal window.

# FTP / sFTP

[How To Set Up vsftpd for a User's Directory on Ubuntu 18.04 | DigitalOcean](https://www.digitalocean.com/community/tutorials/how-to-set-up-vsftpd-for-a-user-s-directory-on-ubuntu-18-04)

[Install and configure vsftpd in Ubuntu (test-usertechnologies.com)](https://www.emiprotechnologies.com/technical_notes/odoo-technical-notes-59/post/install-and-configure-vsftpd-475)

## What Are They?

FTP (File Transfer Protocol) is a network protocol which was widely used for **moving files between a client and server**. It has since been replaced by faster, more secure, and more convenient ways of delivering files.

FTP default port is 21.

Here is the list of some well-known FTP servers:

* **FTPD**
* **VSFTPD**: Stands for Very Secure File Transfer Protocol Daemon. It is GPL-licensed FTP server for UNIX systems, including Linux. It is secure, stable and extremely fast.
* **PROFTPD**
* **PUREFTPD**

Similarly, sFTP (Secure File Transfer Protocol) is also FTP but it's much more secured. SFTP default port is 22

Here is the list of some well-known sFTP servers:

* **sftp**: Typically part of SSH server, such as OpenSSH, which is a widely used in Linux distributions.

## How Do They Work?

## Installtion

### vsftpd

#### Installing vsftpd

$ sudo apt update

$ sudo apt install vsftpd

Note: After installing, save the original configuration as a backup:

$ sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.orig

#### Opening the Firewall

First, chek the firewall status to see if it’s enabled:

$ sudo ufw status

1. If firewall is off, we can skip this step:

Status: inactive

2. If firewall is on, and for example – when only SSH is allowed through, we'll need to add rules for FTP traffic:

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

Let’s open ports 20 and 21 for FTP, port 990 for when we enable TLS, and ports 40000-50000 for the range of passive ports we plan to set in the configuration file:

$ sudo ufw allow 20/tcp

$ sudo ufw allow 21/tcp

$ sudo ufw allow 990/tcp

$ sudo ufw allow 40000:50000/tcp

$ sudo ufw status

Our firewall rules should now look like this:

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

990/tcp ALLOW Anywhere

20/tcp ALLOW Anywhere

21/tcp ALLOW Anywhere

40000:50000/tcp ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

20/tcp (v6) ALLOW Anywhere (v6)

21/tcp (v6) ALLOW Anywhere (v6)

990/tcp (v6) ALLOW Anywhere (v6)

40000:50000/tcp (v6) ALLOW Anywhere (v6)

#### Create a test user and a user directory

Add a test user:

$ sudo adduser test-user

Create the 'ftp' folder, set its ownership and remove its write permissions:

$ sudo mkdir /home/test-user/ftp

$ sudo chown nobody:nogroup /home/test-user/ftp

$ sudo chmod a-w /home/test-user/ftp

Create the directory for uploading files, assign ownership to the user and set full permission:

$ sudo mkdir /home/test-user/ftp/files

$ sudo chmod 777 /home/test-user/ftp/files

$ sudo chown test-user:test-user /home/test-user /ftp/files

A permissions check on the 'ftp' directory should return the following:

$ sudo ls -la /home/test-user/ftp

Output

total 12

dr-xr-xr-x 3 nobody nogroup 4096 Aug 26 14:01 .

drwxr-xr-x 3 test-user test-user 4096 Aug 26 13:59 ..

drwxr-xr-x 2 test-user test-user 4096 Aug 26 14:01 files

Finally, let’s add a 'test.txt' file to use when we test:

$ echo "vsftpd test file" | sudo tee /home/test-user/ftp/files/test.txt

#### Modify configuration

Open configuration file:

$ sudo nano /etc/vsftpd.conf

To allow user to upload the files, uncomment the 'write\_enable'setting.

write\_enable=YES

To allow FTP user to access the assigned directory to it, uncomment 'chroot\_local\_user':

chroot\_local\_user=YES

Add a 'user\_sub\_token' to insert the username in our 'local\_root' directory path so that our configuration will work for this user and any additional future users. Add these settings anywhere in the file:

user\_sub\_token=$USER

local\_root=/home/$USER/ftp

Also limit the range of ports that can be used for passive FTP to make sure enough connections are available:

pasv\_min\_port=40000

pasv\_max\_port=50000

[OPTIONAL]

To allow FTP access on a case-by-case basis, let’s set the configuration so that users have access only when they are explicitly added to a list, rather than by default:

userlist\_enable=YES

userlist\_file=/etc/vsftpd.userlist

userlist\_deny=NO

Finally, let’s add our user to /etc/vsftpd.userlist. Use the -a flag to append to the file:

echo "test-user" | sudo tee -a /etc/vsftpd.userlist

When you’re done making the changes, save the file and exit the editor.

Restart the daemon to load the configuration changes:

$ sudo systemctl restart vsftpd

#### Testing FTP Access

We’ve configured the server to allow only the user 'test-user' to connect via FTP. Let's make sure that this works as expected.

**Users other than 'test-user' should FAIL to connect**

Let’s try connecting as 'sudo\_user'. It should also be denied access:

$ ftp -p <your-server-ip>

Output

Connected to <your-server-ip>.

220 (vsFTPd 3.0.3)

Name (your-server-ip>:default): sudo\_user

530 Permission denied.

ftp: Login failed.

Close the connection:

ftp> bye

**The user 'test-user' should be able to connect, read, and write files**

Let's try connecting as 'test-user. It should also be accepted access:

$ ftp -p <your-server-ip>

Output

Connected to <your-server-ip>.

220 (vsFTPd 3.0.3)

Name (your-server-ip>:default): test-user

331 Please specify the password.

Password: <your-user-pass>

230 Login successful.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp>

Now change into the 'files' directory and use the 'get' command to transfer the test file we created earlier to our local machine:

cd files

get test.txt

Output

227 Entering Passive Mode (...).

150 Opening BINARY mode data connection for test.txt (16 bytes).

226 Transfer complete.

16 bytes received in 0.0101 seconds (1588 bytes/s)

ftp>

Next, upload the file with a new name to test write permissions:

put test.txt upload.txt

Output

227 Entering Passive Mode (...).

150 Ok to send data.

226 Transfer complete.

16 bytes sent in 0.000894 seconds (17897 bytes/s)

Close the connection:

bye

#### Securing Transactions (optional)

<https://www.digitalocean.com/community/tutorials/how-to-set-up-vsftpd-for-a-user-s-directory-on-ubuntu-18-04> (from Step 6)

#### Common errors

**Error 1**: 550 Create directory operation failed.

Reason: This error occurs when subdirectory in FTP root directory has not got full access permission, i.e., 777.

Solution:

$ sudo chmod 777 <FTP-sub-dir-path>

**Error 2**: 500 OOPS: vsftpd: refusing to run with writable root inside chroot()

Reason: This error comes up when either FTP root directory has got full access permission or not the actual required permissions. Home directory must not be writable by the user.

Solution:

$ sudo chmod a-w <FTP-root-path>

### sftp

sftp -P 51983 worker@tri-ho-trihojlr.vbee.lge.com

> get /home/worker/vcm/nad/vcm-sa2150p\_231107\_2022.zip

# SCP

## What Is SCP?

Secure Copy Protocol (SCP) is a means of **securely transferring computer files between a local host and a remote host**, or between two remote hosts. It is based on the Secure Shell (SSH) protocol.

With SCP, you can copy a file or directory:

* From your local system to a remote system.
* From a remote system to your local system.
* Between two remote systems from your local system.

## How Does SCP Work?

## SCP Commands

Syntax:

$ scp [OPTION] [user@]SRC\_HOST:]file1 [user@]DEST\_HOST:]file2

**Where:**

* OPTION – Common [scp options](https://linux.die.net/man/1/scp) are:
* -r: Copy directories recursively
* -P: Specifies the remote host SSH port. Used when SSH port is not 22.
* -p: Preserves files modification and access times.
* -q: Suppresses the progress meter and non-error messages.
* -C: Compresses the data as it is sent to the destination machine.
* -c: Specifies cipher (encryption algorithms) to use
* [user@]SRC\_HOST:]file1 – Source file.
* [user@]DEST\_HOST:]file2 – Destination file

**Notes:**

* Local files should be specified using an absolute or relative path, while remote file names should include a user and host specification.
* SCP relies on SSH for data transfer, so it **requires an SSH key or password** to authenticate on the remote systems.
* To be able to copy files, you must have at least **read permissions on the source file** and **write permission on the target system**.
* Be careful when copying files that share the same name and location on both systems, SCP will **overwrite files without warning**.

**Examples:**

**1.** Copy file file.txt from a local to a remote system (IP: 10.10.0.2) in directory /remote/dir:

$ scp file.txt remote-username@10.10.0.2:/remote/dir

If you don’t specify a remote directory, the file will be copied to the remote user home directory.

You will be prompted to enter the user password, and the transfer process will start.

Remote-username@10.10.0.2's password:

file.txt 100% 0 0.0KB/s 00:00

**2.** Copy file file.txt from a local to a remote system (IP: 10.10.0.2) in directory /remote/dir. Then change the file name to newfile.txt:

$ scp file.txt remote-username@10.10.0.2:/remote/dir/newfile.txt

**3.** If SSH on the remote host is listening on a port other than the default 22, then specify the port:

$ scp -P 2322 file.txt remote-username@10.10.0.2:/remote/dir

**4.** Copy a directory from a local to remote system:

$ scp -r /local/dir remote-username@10.10.0.2:/remote/dir

**5.** Copy the file /files/file.txt from the remote host host1.com to the directory /files on the remote host host2.com:

$ scp user1@host1.com:/files/file.txt [user2@host2.com:/files](mailto:user2@host2.com:/files)

## Tips

### Using cipher to ensure data encryption during file transfer

If you don't specify any cipher, then the chosen one will be the highest in the client's order of preference that is also supported by the server. Allowed values are: 'aes128-ctr', 'aes128-cbc', 'aes192-ctr', 'aes192-cbc', 'aes256-ctr', 'aes256-cbc', 'blowfish-cbc', 'arcfour', 'arcfour128', 'arcfour256', 'cast128-cbc', '3des-cbc', etc.

To get a list of supported ciphers on your system, run:

$ ssh -Q cipher

You can also set this value to 'none', then data won't be encrypted.

Note that the choice of cipher can impact the performance and compatibility of the connection.

### Increasing file transfer speed

There are some ways to do that:

* + - 1. Separate the file into small parts. Then copy all of them in parallel.
      2. Choose a cipher with highest performance. Or disable using cipher (set 'none')

# Rsync

rsync -e 'ssh -p 51983' worker@tri-ho-trihojlr.vbee.lge.com:/home/worker/vcm/vcm-sa2150p\_231107\_1622.zip .