# SSH

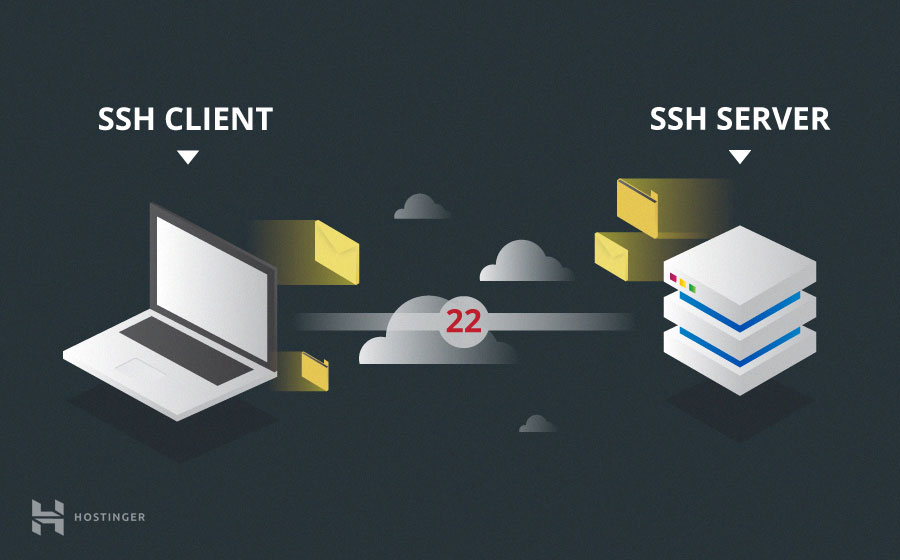
## What Is SSH?

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

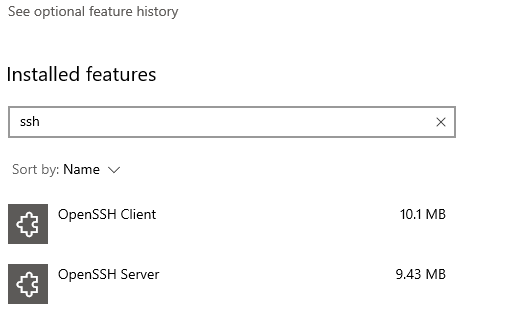
### On Debian / Ubuntu

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 Command

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, option for port (-p <port-id>) is used to select the TCP 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.

## Tips

### SSH between Windows Host and Linux Guest which run on VMWare Player / Virtual Box on another computer

It's very easy to creating a SSH connection **between two computers**, or **between a host and a virtual machine running on this host**. Just execute the SSH command and done!

However, thing becomes **complicated when SSH-ing between a host and virtual machine guest which run on another computer (we can call it a 'remote VM guest')**.

In this example, we'll connect from a Windows host to a Linux guest which run on VMWare Player / Virtual Box on another computer via SSH.

Note: We'll use NAT network instead of Bridge.

**I - VMWare Player / VMWare Workstation**

**1. Modify NAT config**

Open and edit C:\ProgramData\VMware\vmnetnat.conf with Administrator permission as followings:

In [incomingtcp] section, add a line:

2244 = <vm-guest-linux-ip>:22

Note: Besides port 2244, you can choose any port you want.

After saving the file, run Windows command to restart VMWare NAT service (so the config can be reloaded) by running following commands:

$ net stop "VMWare NAT Service"

$ net start "VMWare NAT Service"

Note: If these commands won't work, you can restart the service with *Services* app on Windows.

To make sure things work, try SSH into your Linux guest from your Windows host (though it's not our purpose) with the command:

$ ssh -p 2244 <linux-username>@localhost

**2. Modify Windows Firewall config**

Trying to SSH to the guest from another host won't be allowed because of the Windows Firewall rules.

To make it possible, you have two ways:

1. Disable Windows Firewall
2. Add new rule by:

Open Control Panel > Windows Defender Firewall > Advanced settings. New an 'Inbound Rules'.

Select 'Port' as rule type.

Select TCP and set the port '2244'

Save

**3. SSH to your Linux VM guest from remote Windows host**

Simply run this command:

$ ssh -p 2244 <linux-username>@<windows-hostname-or-ip>

Note: windows-hostname-or-ip is of the local Windows host which the virtual machine run on, NOT of the remote host we're typing on.

**II - Virtual Box**

**1. Modify NAT config**

Settings > Network > Adapter[n] , click 'Port Forwarding'. Configure a NAT rule, for instance:

* Name: External SSH incoming
* protocol: TCP
* Host IP: Your Windows host public IP
* Host port: The port the remote SSH connects to (e.g., port 2244)
* Guest IP: Your virtual machine Linux guest IP
* Guest port: it’s TCP 22 for SSH

**2. SSH to your Linux VM guest from another Windows host**

Simply run this command:

$ ssh -p 2244 <linux-username>@<windows-hostname-or-ip>

Note: windows-hostname-or-ip is of the local Windows host which the virtual machine run on, NOT of the remote host we're typing on.

Ref: <https://slmeng.medium.com/how-to-ssh-into-vmware-player-virtual-box-guest-linux-os-remotely-host-os-is-window-10-40cb348c996f>

### 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:

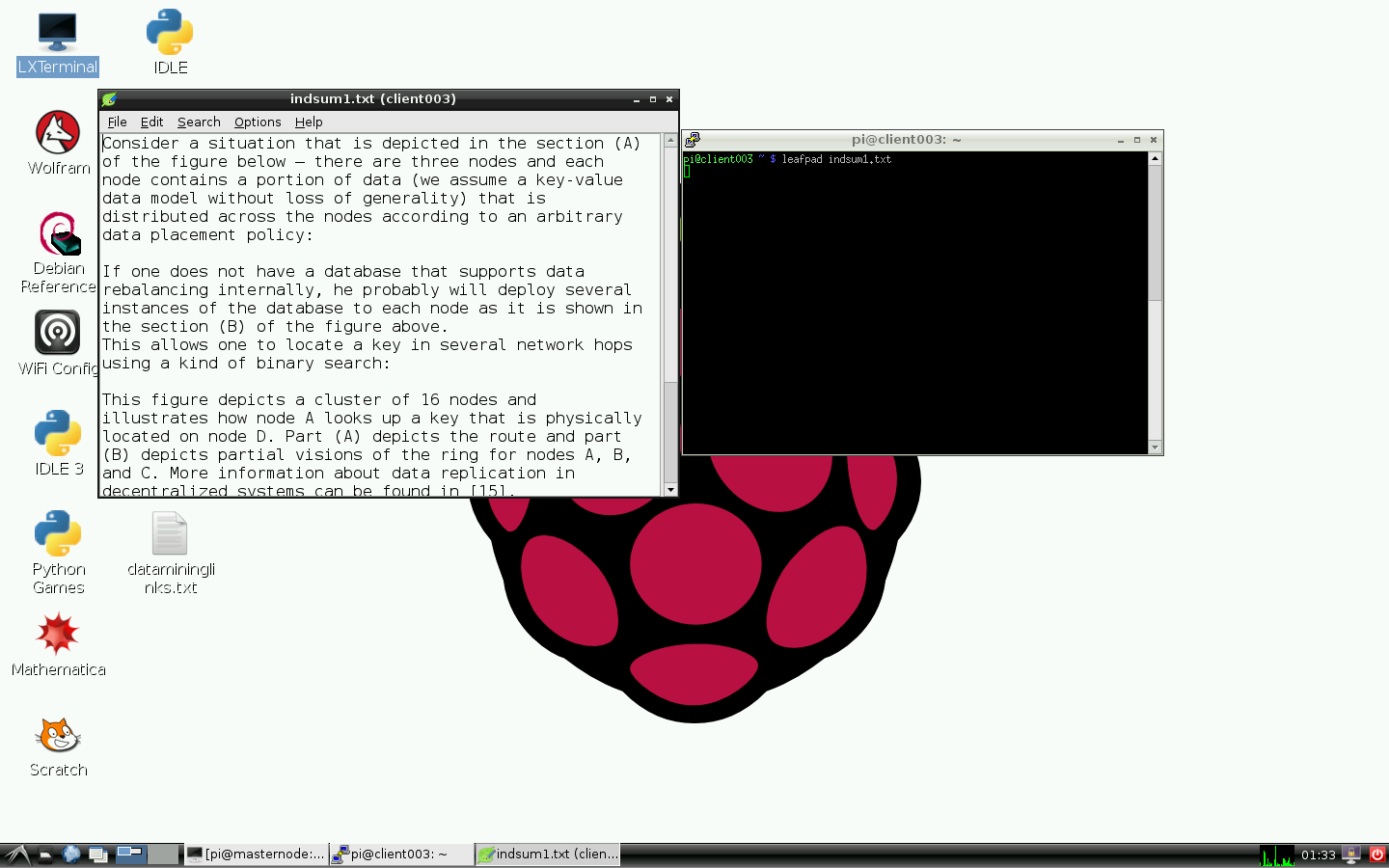
Step 1: Install a X server. For example, [Xming](https://sourceforge.net/projects/xming/) on Windows.

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.



## Common Issues:

### Warning: Remote Host Identification Has Changed!

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

# IPs (Internet Protocols)

## TCP (Transmission Control Protocol)

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

## UDP (User Datagram Protocol)

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

## ICMP (Internet Control Message Protocol)

<https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol>

**Comparison (TCP, UDP, and ICMP)**:

<https://www.pingplotter.com/wisdom/article/packet-type-differences#:~:text=The%20Internet%20Control%20Message%20Protocol,a%20traditional%20data%20packet%20protocol.&text=One%20device%20sends%20out%20an,confirming%20it%20received%20the%20request>.

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

<https://www.geeksforgeeks.org/differences-between-tcp-and-udp/>

# 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 Command

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

File Transfer Protocol

FTP default port: 21

SFTP default port: 22

[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.

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

## 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>