What SSH Is and Why You'd Use It

The plain-English on-ramp to controlling a server from your keyboard

SSH series, part 1 of 3. Next: How and Why to Use an SSH Key.

You keep hearing it. "Just SSH in and check the logs." "SSH into the box and restart it." Someone set up your website, or your document server, or your backup machine, and reaching it means this three-letter command everyone acts like you were born knowing. Here is what it actually is, in plain English, before anyone asks you to install a single thing.

What SSH is

SSH stands for Secure Shell. It is a private, encrypted connection from your computer to another computer, so you can control that other computer by typing commands to it. Instead of clicking around a screen, you type an instruction and the far machine carries it out.

Two words, two promises. "Shell" means you are working at a command line, giving the machine typed instructions. "Secure" means everything traveling over that connection is encrypted, so anyone watching the network in between sees scrambled noise, not your commands, your passwords, or your files.

The one idea that makes it click: client and server

Almost all the confusion around SSH comes from missing this. There are two roles.

The **client** is the thing you type into. That is your laptop. On a Mac and on Linux the client is already installed; on Windows 10 and 11 it is already installed too. The client is all you need in order to connect out to another machine.

The **server** is the machine you are connecting to. That is your VPS, your office box, the server someone stood up for you. It runs a small piece of software that sits and listens for incoming connections. If you are connecting to a machine that someone already set up, its server side is already handled and you do not touch it.

The rule of thumb: you are almost always the client, and the server is the thing on the other end. When a guide talks about "installing the server," it means the machine you want to reach, not your laptop.

Check that you have the client

You almost certainly do. Open a terminal (on a Mac, the Terminal app; on Windows, PowerShell) and type:

```
ssh
```

If you get back a page of usage text, a list of options and flags, the client is installed and you are ready. If instead you get "command not found," see the setup guides linked at the end. Nearly everyone gets the usage text.

Your first connection

Every SSH connection has the same shape:

```
ssh username@address
```

`username` is your login name on the server, and `address` is the server's IP number or domain name. For example:

```
ssh you@198.51.100.23
```

The first time you connect to a given machine, it asks whether you are sure you want to continue and shows a long string of characters. Type `yes`. That string is the server's fingerprint, and SSH is memorizing it so it can warn you later if something ever tries to impersonate that machine.

Then it asks for your password. Type it and press Enter. You will not see dots or characters appear as you type; that is deliberate, not a frozen screen. Once you are in, the prompt changes to the server's, and from that moment every command you type runs on the server, not on your laptop. When you are done, type `exit` to come back to your own machine.

Why a lawyer would bother

Here is the honest answer: a command line is about power, access, and authority. A graphical dashboard is a layer someone else built and set between you and the machine, offering only the buttons they decided to give you. The terminal cuts straight through all that noise and does the thing you actually want, directly. It is not as pretty as a graphical interface; it is far more efficient, and it does not stop at the edge of a menu someone else designed.

That matters the moment you run anything yourself: a website, a document server, a backup box, a small tool that holds client files. SSH is the private door to those machines. The alternative is letting a vendor's web dashboard be the only way in, which means someone else holds the keys to the place your client data lives. SSH is how you hold those keys yourself. Every self-hosting move in these tips,

your own e-signing, your own servers, your own backups, assumes you can reach the machine, and SSH is how you reach it.

The part the tutorials skip: "in" means the live machine

When you SSH in, you are not looking at a copy or a preview. You are operating the real, running machine. A file you delete is really deleted; a service you stop is really stopped. That is the power, and it is the responsibility, in the same breath.

Two habits from day one. Connect only to machines you own or are clearly authorized to touch. And read a command before you run it, because a server has no recycle bin waiting to catch a mistake. Passwords over SSH work, and they are also the weakest link in this chain; the next tip in the series replaces them with something better.

Where to go next

- If the machine you want to reach is not yet set up to accept connections, because you are building the server itself, start with "How to Set Up SSH on Linux" or "How to Set Up SSH on Windows."
- Otherwise, the next tip in this series is "How and Why to Use an SSH Key," which stops the password prompts and makes the connection both easier and safer.

The playbook

```
# you (the client)  ->  the server
ssh you@server-address     # connect
# type "yes" the first time, then your password
exit                       # leave, back on your own machine
```

That is SSH: one command, one encrypted line, one machine controlling another.