SSH Shortcuts: Stop Typing the Whole Command

One config file that turns every long SSH command into a two-word alias

SSH series, part 3 of 3. Assumes you can connect (part 1) and have a key set up (part 2).

By now you connect with something like `ssh -i ~/.ssh/id_ed25519 -p 22 you@198.51.100.23`. Fat-finger one flag and it fails. Do that ten times a day across a few servers and you are memorizing IP addresses like phone numbers from 1985.

There is a file that fixes this for good. A few minutes of setup, and that whole command becomes `ssh main`.

What "~/.ssh/config" is

The file is `~/.ssh/config`, and it lives on your computer, not on the server. You give each machine a short name once and record its address, login, key, and port next to it. Every tool that speaks SSH reads this file, so the shortcut works not just for `ssh` but for `scp`, `rsync`, and `git` too.

Breaking the path apart: `~` is your home folder, `.ssh` is the hidden folder SSH already keeps your keys in (you made a key there in part 2), and `config` is a plain text file inside it that does not exist until you create it.

Check whether the folder is there, and create it if not:

```
ls -la ~/.ssh           # Mac / Linux   (on Windows: ls ~\.ssh)
mkdir -p ~/.ssh         # only if the folder does not exist
```

Open the file and write in it

Open the config file in a text editor; it gets created if it is not there yet.

```
nano ~/.ssh/config              # Mac / Linux
```

```
notepad ~\.ssh\config           # Windows
```

Paste the block below, then save and close. In nano, that is `Ctrl + O` then Enter to save and `Ctrl + X` to exit; in Notepad, `Ctrl + S` and close the window.

```
# ~/.ssh/config

Host main
    HostName 198.51.100.23
    User you
    Port 22
    IdentityFile ~/.ssh/id_ed25519

Host vps
    HostName 203.0.113.10
    User root
    IdentityFile ~/.ssh/id_ed25519

# defaults applied to every host above
Host *
    ServerAliveInterval 60
    AddKeysToAgent yes
```

Save it, and `ssh main` now expands to the full command with the right user, port, and key.

What each line does

- **Host** is the short name you type. Call it whatever you want (`main`, `vps`, `office`).
- **HostName** is the real address of the server, an IP or a domain name.
- **User** is the login name, so you drop the `you@` prefix.
- **Port** only matters if the server is not on the default, 22.
- **IdentityFile** points at the private key you made in part 2, so you drop the `-i` flag. If you have not set up a key yet, delete this line and SSH will ask for a password instead.
- **Host *** is a defaults block applied to every host above it. `ServerAliveInterval 60` keeps an idle session from dropping while you read; `AddKeysToAgent yes` means you enter your key passphrase once per day instead of once per connection.

Finding a server's address

For `HostName` you need the server's real address. It is on your provider's dashboard if the server is a VPS. If you are already logged into the server, `hostname -I` shows its local address and `curl ifconfig.me` shows the public one. If the server has a domain name pointed at it, you can use that instead of the number.

It works for more than ssh

The same aliases work anywhere SSH runs under the hood:

```
scp report.pdf main:/home/you/       # copy a file up to the server
rsync -a ./site/ vps:/var/www/        # sync a whole folder
git clone main:/srv/repo.git          # git over SSH, same short name
```

Two tricks worth knowing

Wildcards. One block can match a whole group of servers at once:

```
Host *.sample.url
    User you
```

Jump hosts. If a private server is only reachable by first connecting through a gateway, let SSH make the hop for you:

```
Host private
    HostName 10.0.0.5
    User you
    ProxyJump vps
```

Now `ssh private` connects through `vps` automatically. No manual two-step.

Lock the file down

On Mac and Linux, SSH ignores the config if its permissions are too loose. Set them once:

```
chmod 600 ~/.ssh/config
```

On Windows this step is not needed; the default permissions on your user folder are already correct.

> **Using AI to build yours:** You do not have to write the block by hand. Tell Claude or Grok what you have, in plain English: "Write me an SSH config entry. The server is at 198.51.100.23, I log in as `you`, my key is `id_ed25519`, and I want to call it `main`." It hands back the exact block to paste. When it does, paste any line you do not recognize back to it and ask what it does before you save. That is how you learn the tool instead of just copying it.

The part the tutorials skip: this file is a map of your whole practice

Your `~/.ssh/config` is a labeled inventory of every server that touches client data: its address, the account you log in as, and the path to the key that opens it. Paired with your `known_hosts` file, it is a roadmap to your entire setup, sitting in plaintext on your laptop.

- **Encrypt the laptop and `chmod 600` the file**, so a lost or stolen machine is an inconvenience, not a handed-over map of where everything lives.

- **Keep the private key passphrase-protected** (part 2). The config only points at the key; if the key itself is naked, the alias is a one-click door for whoever holds the laptop.
- **Be careful with `ForwardAgent yes`.** It is convenient for hopping between servers, but on a machine someone else controls it can let them borrow your keys behind your back. Prefer `ProxyJump`, which gets you through a gateway without exposing your agent.

Where to go next

That is the series: you understand SSH, you connect with a key, and you reach any server with a two-word alias. From here, "How to Copy Files Between Computers with SCP" uses these same shortcuts to move documents around.

The playbook

```
nano ~/.ssh/config          # or on Windows: notepad ~\.ssh\config
# add a Host block per server, save, and close
chmod 600 ~/.ssh/config     # Mac / Linux
ssh main                    # done
```

Set it once and you stop typing IP addresses for good.