All you need to know about SSH for Ubuntu

A practical guide to setting up, securing, and getting the most out of SSH on Ubuntu.

Setup

Install OpenSSH and open the firewall:

```
sudo apt update && sudo apt install openssh-server -y
sudo systemctl enable --now ssh
sudo ufw allow ssh
```

Verify it's running with `sudo systemctl status ssh`, then connect from your client:

```
ssh user@server-ip
```

Key-Based Authentication

Always use keys over passwords. Generate one on your client:

```
ssh-keygen -t ed25519 -C "my-device"
```

Copy it to the server:

```
ssh-copy-id user@server-ip
```

If you set a passphrase (you should), use the SSH agent so you don't have to re-enter it every time:

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
```

```
# macOS: persist in Keychain
ssh-add --apple-use-keychain ~/.ssh/id_ed25519
```

Hardening the Server

Edit `/etc/ssh/sshd_config` — here are the settings that matter most:

```
PermitRootLogin no
PasswordAuthentication no
MaxAuthTries 3
LoginGraceTime 30
AllowUsers alice bob
```

Always test before restarting:

```
sudo sshd -t && sudo systemctl restart ssh
```

> Keep an existing session open while testing changes. Getting locked out of your own server is not fun.

Tips & Tricks

This is a non-exhaustive collection of things that make working with SSH much nicer. Most of these go into `~/.ssh/config`.

Host Aliases

Instead of typing `ssh -p 2222 deployer@prod.example.com -i ~/.ssh/id_ed25519_prod` every time, define it once:

```
Host prod
    HostName prod.example.com
    User deployer
    Port 2222
    IdentityFile ~/.ssh/id_ed25519_prod
```

Now just run `ssh prod`. This also works with `scp`, `rsync`, and `git`.

You can define as many as you want:

```
Host staging
    HostName staging.example.com
    User deployer

Host pi
    HostName 192.168.1.50
    User pi
```

Defaults and Wildcards

Set sane defaults for all connections:

```
Host *
    ServerAliveInterval 60
    ServerAliveCountMax 3
    AddKeysToAgent yes
    IdentitiesOnly yes
```

Or scope settings to a domain:

```
Host *.corp.example.com
    User admin
    IdentityFile ~/.ssh/id_ed25519_work
```

Port Forwarding (SSH Tunnels)

Local forward — access a remote service through a local port. Great for databases that only listen on localhost:

```
ssh -L 5432:localhost:5432 admin@db-server
# now connect to localhost:5432 as if you were on the server
```

In config (runs automatically when you connect):

```
Host db-tunnel
    HostName db-server.example.com
```

```
    User admin
    LocalForward 5432 localhost:5432
    LocalForward 8080 internal-app:8080
```

Remote forward — expose a local service through the remote server:

```
ssh -R 8080:localhost:3000 user@vps
# vps:8080 now points to your local port 3000
```

SOCKS proxy — route all traffic through the server:

```
ssh -D 1080 user@ssh-server
# configure your browser to use SOCKS5 proxy on localhost:1080
```

Background tunnels — run a tunnel without an interactive shell:

```
ssh -f -N -L 5432:localhost:5432 admin@db-server
```

For tunnels that should survive disconnects, use `autossh`:

```
sudo apt install autossh -y
autossh -M 0 -f -N -L 5432:localhost:5432 admin@db-server
```

Jump Hosts / Bastion

Reach a server that's only accessible through an intermediate host:

```
Host bastion
    HostName bastion.example.com
    User jump-user

Host internal
    HostName 10.0.0.5
    User admin
    ProxyJump bastion
```

`ssh internal` now transparently hops through the bastion. Chain multiple hops with `ProxyJump bastion,internal`.

Connection Multiplexing

Reuse an existing connection for instant subsequent logins:

```
Host *
    ControlMaster auto
    ControlPath ~/.ssh/sockets/%r@%h-%p
    ControlPersist 600
```

```
mkdir -p ~/.ssh/sockets
```

The first connection creates a socket. Every connection after that to the same host is near-instant.

Multiple Git Identities

Use different keys for different GitHub/GitLab accounts:

```
Host github-personal
    HostName github.com
    User git
    IdentityFile ~/.ssh/id_ed25519_personal

Host github-work
    HostName github.com
    User git
    IdentityFile ~/.ssh/id_ed25519_work
```

Then clone with `git clone git@github-work:company/repo.git`.

Escape Sequences

These work inside any SSH session (press Enter first):

Keys	What it does
`~.`	Force-disconnect a frozen session
`~C`	Open command line to add tunnels on the fly
`~#`	List active forwarded ports
`~?`	Show all escape sequences

Adding a tunnel mid-session via `~C`:

```
ssh> -L 3306:localhost:3306
Forwarding port.
```

Run Commands Remotely

```
ssh prod "df -h && free -m"
ssh prod 'bash -s' < local-script.sh
```

Mount Remote Filesystems (SSHFS)

```
sudo apt install sshfs -y
sshfs prod:/var/www ~/mounts/prod

# unmount
fusermount -u ~/mounts/prod    # Linux
umount ~/mounts/prod           # macOS
```

Skip Host Key Checking (Internal Networks Only)

```
Host 192.168.1.*
    StrictHostKeyChecking no
    UserKnownHostsFile /dev/null
```

> Only use this in trusted networks. It disables MITM protection.

Debugging

When something doesn't work, add `-v` flags:

```
ssh -vvv prod
```

To see exactly which config options apply to a host:

```
ssh -G prod
```

Quick Reference

Action	Command
Install SSH server	`sudo apt install openssh-server`
Generate key	`ssh-keygen -t ed25519`
Copy key to server	`ssh-copy-id user@host`
Local port forward	`ssh -L 8080:localhost:80 user@host`
Remote port forward	`ssh -R 9090:localhost:3000 user@host`
SOCKS proxy	`ssh -D 1080 user@host`
Background tunnel	`ssh -f -N -L 8080:localhost:80 user@host`
Jump host	`ssh -J bastion user@target`
Debug	`ssh -vvv user@host`

Revision #3
Created 2026-04-15 11:12:47 UTC by Luca Noah Caprez
Updated 2026-04-15 12:30:01 UTC by Luca Noah Caprez