

SSH Config Cheatsheet

By Dejan Panovski • Updated on Oct 6, 2026

SSH client config directives at a glance: Host and Match patterns, keys, jump hosts, keepalives, multiplexing, port forwarding, host key checking, and tokens for ~/.ssh/config.

The SSH client config file stores per-host settings so that `ssh myserver` replaces a long command with flags for the user, port, key, and jump host. This cheatsheet lists the `~/.ssh/config` directives you use most often, how OpenSSH matches and applies them, and the commands that show the settings ssh actually uses.

File Locations

ssh reads the command line first, then your config, then the system config.

<code>~/.ssh/config</code>	Per-user client config
<code>/etc/ssh/ssh_config</code>	System-wide client config for every user
<code>/etc/ssh/ssh_config.d/*.conf</code>	System drop-ins, loaded by <code>Include</code> on Ubuntu and Debian
<code>chmod 600 ~/.ssh/config</code>	Recommended permissions; ssh rejects a file others can write
<code>ssh -F ~/alt-config host</code>	Use a different config file and skip the defaults
<code>ssh -F none host</code>	Ignore every config file

Inspect the Effective Config

Check what ssh will use before you connect.

<code>ssh -G host</code>	Print every option that applies to host, then exit
<code>ssh -G host grep -i proxyjump</code>	Check one option
<code>ssh -v host</code>	Show which config lines and keys are applied while connecting
<code>ssh -o User=admin host</code>	Override a config value for one connection
<code>ssh -T git@github.com</code>	Test a host without opening a shell

Host Patterns

A `Host` line starts a block that runs until the next `Host` or `Match` line.

<code>Host web</code>	The alias <code>web</code> only
<code>Host web1 web2 db</code>	Any of the listed names
<code>Host *.example.com</code>	Every host under <code>example.com</code>
<code>Host 192.168.1.?</code>	Single-character wildcard
<code>Host *.example.com !legacy.example.com</code>	All but the negated host
<code>Host *</code>	Every host; put it at the end for defaults

The first value wins for most directives; repeated `IdentityFile` entries accumulate. Put specific blocks above general ones and `Host *` last.

Match Blocks

`Match` applies a block based on conditions, not just the name.

<code>Match host db*</code>	Hostname after <code>HostName</code> is applied
<code>Match originalhost db*</code>	Name as typed on the command line
<code>Match user root</code>	Remote user
<code>Match localuser alice</code>	Local account running <code>ssh</code>
<code>Match host *.corp exec "nc -z vpn.corp 22"</code>	Apply only when a command exits 0
<code>Match all</code>	Close a conditional block and match everything again

Connection Basics

The directives most host blocks need.

<code>HostName 203.0.113.10</code>	Real hostname or IP behind the alias
<code>User admin</code>	Remote login name
<code>Port 2222</code>	Remote port; default is 22
<code>AddressFamily inet</code>	Force IPv4 (<code>inet6</code> for IPv6)
<code>ConnectTimeout 10</code>	Give up after 10 seconds
<code>ConnectionAttempts 3</code>	Retry the connection before failing
<code>Compression yes</code>	Compress traffic on slow links

Keys and Authentication

Pick the key per host and stop ssh from trying every key it has.

<code>IdentityFile ~/.ssh/id_ed25519_work</code>	Private key for this host
<code>IdentitiesOnly yes</code>	Offer only the configured keys, not every agent key
<code>AddKeysToAgent yes</code>	Add the key to the running agent after first use
<code>CertificateFile ~/.ssh/id_ed25519-cert.pub</code>	SSH certificate to present
<code>PreferredAuthentications publickey</code>	Try public-key authentication only
<code>PasswordAuthentication no</code>	Disable password authentication; keyboard-interactive prompts can still occur
<code>KbdInteractiveAuthentication no</code>	Disable keyboard-interactive authentication, including its password prompts
<code>IgnoreUnknown UseKeychain</code>	Skip options that this ssh build does not know
<code>UseKeychain yes</code>	macOS only: store key passphrases in the keychain

Jump Hosts

Reach private hosts through a bastion.

<code>ProxyJump bastion</code>	Connect through the <code>bastion</code> alias
<code>ProxyJump admin@bastion.example.com:2222</code>	Jump host with user and port
<code>ProxyJump bastion1,bastion2</code>	Chain several jump hosts in order
<code>ProxyJump none</code>	Disable jumping; place this before a matching broader block
<code>ProxyCommand ssh -W %h:%p bastion</code>	Older equivalent of <code>ProxyJump</code>
<code>ssh -J bastion internal</code>	Same jump from the command line

Keepalives

Stop idle sessions from being dropped by NAT or firewalls.

<code>ServerAliveInterval 60</code>	Send a keepalive after 60 seconds without server data
<code>ServerAliveCountMax 3</code>	Disconnect after 3 unanswered keepalives
<code>TCPKeepAlive yes</code>	TCP-level keepalives; on by default

Multiplexing

Reuse one TCP connection for later sessions to the same host.

<code>ControlMaster auto</code>	Open a master connection, or reuse an existing one
<code>ControlPath ~/.ssh/cm-%C</code>	Socket path; %C keeps it short and unique
<code>ControlPersist 10m</code>	Keep the master open 10 minutes after the last session
<code>ssh -O check host</code>	Check whether a master connection is running
<code>ssh -O exit host</code>	Close the master connection

Port Forwarding

Set up tunnels every time you connect to a host.

<code>LocalForward 8080 localhost:80</code>	Local port 8080 to port 80 on the remote side
<code>LocalForward 3307 db.internal:3306</code>	Reach a host that only the server can see
<code>RemoteForward 9000 localhost:3000</code>	Remote port 9000 to local port 3000
<code>DynamicForward 1080</code>	SOCKS proxy on local port 1080
<code>ExitOnForwardFailure yes</code>	Abort if a forward cannot be set up
<code>ForwardAgent yes</code>	Forward your agent; set it only for trusted hosts
<code>ForwardX11 yes</code>	Forward X11 for GUI applications

Host Key Checking

Control how ssh treats new and changed server keys.

<code>StrictHostKeyChecking ask</code>	Prompt for new hosts; the default
<code>StrictHostKeyChecking accept-new</code>	Trust new hosts, still refuse changed keys
<code>StrictHostKeyChecking yes</code>	Refuse any host not in <code>known_hosts</code>
<code>UserKnownHostsFile ~/.ssh/known_hosts_lab</code>	Separate <code>known_hosts</code> file for a group of hosts
<code>UserKnownHostsFile /dev/null</code>	Do not record keys; for throwaway VMs only
<code>HashKnownHosts yes</code>	Hash hostnames written to <code>known_hosts</code>
<code>UpdateHostKeys yes</code>	Accept additional keys the server announces

Session and Environment

Change what happens after you log in.

<code>RemoteCommand</code> <code>tmux new -A -s main</code>	Run a command instead of a login shell
<code>RequestTTY</code> <code>yes</code>	Allocate a terminal; needed with <code>RemoteCommand</code> for interactive tools
<code>SetEnv</code> <code>APP_ENV=prod</code>	Send a variable; the server must allow it with <code>AcceptEnv</code>
<code>SendEnv</code> <code>LANG LC_*</code>	Pass local variables by name
<code>LogLevel</code> <code>ERROR</code>	Hide warnings; <code>DEBUG</code> for troubleshooting
<code>Include</code> <code>~/.ssh/config.d/*</code>	Load more files; put it at the top of the file

Tokens

Expanded in `ControlPath`, `IdentityFile`, `RemoteCommand`, and similar directives. Supported tokens vary by directive.

<code>%h</code>	Remote hostname after <code>HostName</code>
<code>%n</code>	Hostname as typed on the command line
<code>%p</code>	Remote port
<code>%r</code>	Remote user
<code>%u</code>	Local user
<code>%d</code>	Local home directory
<code>%C</code>	Hash of local host, remote host, port, remote user, and jump-host setting
<code>%%</code>	A literal <code>%</code>

`ProxyCommand` accepts only `%%`, `%h`, `%n`, `%p`, and `%r`.

Troubleshooting

Bad owner or permissions on <code>~/.ssh/config</code>	Check ownership with <code>ls -l ~/.ssh/config</code> ; the owner must be you or root. Fix incorrect ownership, then run <code>chmod 600 ~/.ssh/config</code>
Bad configuration option	Typo, or an option this ssh version lacks; see <code>man ssh_config</code>
Option is ignored	An earlier block set it first; check <code>ssh -G host</code>
Too many authentication failures	Add <code>IdentitiesOnly yes</code> and an <code>IdentityFile</code>
Permission denied (publickey)	<code>ssh -v host</code> and look for <code>Offering public key</code>
too long for Unix domain socket	Shorten <code>ControlPath</code> ; use <code>%C</code> instead of <code>%r@%h:%p</code>

Related Guides

Use these articles for full SSH client workflows.

<u>Using the SSH Config File</u>	Host blocks, patterns, precedence, and examples
<u>SSH Command in Linux</u>	ssh options and connection examples
<u>How to Set Up SSH Tunneling</u>	Local, remote, and dynamic port forwarding
<u>SSH Folder Files and Permissions</u>	What lives in <code>~/.ssh</code> and the correct modes
<u>SSH Cheatsheet</u>	ssh, keys, agent, scp, and sftp commands