autossh Self-healing ssh tunnels

START HERE

`autossh -M 0 <ssh args>`
Restart ssh, no monitor port

`-o ExitOnForwardFailure=yes`
Retry if a forward fails

`autossh -M 0 -N -L <lport>:<dst>:<dport> <host>`
Persistent local forward

`autossh -M <port> <host>`
Monitor via port and port+1

`autossh -M 0 -f -N -R <rp>:localhost:<p> <host>`
Background reverse tunnel

`AUTOSSH_GATETIME=0 autossh ...`
Keep retrying at boot

`-o ServerAliveInterval=30 -o ServerAliveCountMax=3`
Detect a dead link

`kill -USR1 <pid>`
Force a new ssh now

autossh supervisor → **spawns** → **ssh** your tunnel or session → **keepalive or test data** → **Remote host** sshd → **ssh exits or stalls** → **Restart** backs off up to the poll time

CONCEPTS

autossh runs ssh for you: every argument except its own flags goes to ssh unchanged. Get the plain ssh command working first, then put autossh in front.

Use key or agent login: autossh cannot answer a password prompt, and -f makes that impossible.

RUNNING AUTOSSH

`autossh -M 0 <host>`
Supervised login session
A normal exit (typing exit) ends autossh too; only failures restart ssh.

`autossh -M 0 -N -L <lport>:<dst>:<dport> <host>`
Keep a local forward alive

`autossh -M 0 -N -R <rp>:localhost:<p> <host>`
Keep a reverse tunnel alive

`autossh -M 0 -N -D <port> <host>`
Keep a SOCKS proxy alive

`autossh -M 0 -f -N <ssh args>`
Drop to the background
-f is autossh's own: ssh cannot then ask for passwords or passphrases.

`autossh -M 0 -N -J <jump> <host>`
Supervise a tunnel via a jump host

`autossh -M 0 -i <key> -p <port> <user>@<host>`
Choose key and port as for ssh

`autossh -M 0 -F <config> <host>`
Use another ssh config file

`autossh -V`
Show the autossh version

Prefer an entry in ~/.ssh/config for the forwards and keepalives, then run just autossh -M 0 -f -N <host>.

MONITORING

`-M 0`
Disable the monitor port, rely on ssh keepalives
Needs ServerAliveInterval/ServerAliveCountMax so ssh exits when the link dies.

MONITORING (cont.)

`-M <port>`
Monitor with a port pair
Uses <port> and <port>+1 locally and on the remote; both must be free.

`-M <port>:<echo_port>`
Monitor via a remote echo service
Port 7 is the inetd echo service, often disabled.

`-o ServerAliveInterval=30 -o ServerAliveCountMax=3`
Ssh quits after ~90 s of silence

`-o ExitOnForwardFailure=yes`
Exit ssh if a forward cannot bind

`-o ConnectTimeout=10`
Fail fast when the host is down

`-o BatchMode=yes`
Never prompt: fail instead

`-o StrictHostKeyChecking=accept-new`
Trust new hosts on first connect

On the server side ClientAliveInterval in sshd_config closes stale listening sockets when the client vanishes, so a restarted reverse tunnel can bind again.

ENVIRONMENT VARIABLES

`AUTOSSH_GATETIME=<s>`
Seconds ssh must survive to count as started (30)
0 disables the gate and retries even if the first ssh fails with status 1. Useful at boot.

`AUTOSSH_POLL=<s>`
Seconds between connection tests (600)

`AUTOSSH_FIRST_POLL=<s>`
Delay before the first test

`AUTOSSH_PORT=<port>`
Monitor port; overrides -M

`AUTOSSH_DEBUG=1`
Debug log, copied to stderr

`AUTOSSH_LOGLEVEL=<0-7>`
Syslog level, 7 chattiest

`AUTOSSH_LOGFILE=<file>`
Log to a file instead of syslog

`AUTOSSH_PIDFILE=<file>`
Write the autossh pid

`AUTOSSH_PATH=<path>`
Use this ssh executable

ENVIRONMENT VARIABLES (cont.)

Variables go in front of the command (AUTOSSH_GATETIME=0 autossh ...) or in a systemd Environment= line.

CONTROL AND TROUBLESHOOTING

`kill -USR1 <pid>`
Kill ssh and start a new one

`kill -HUP <pid>`
Prod autossh to retry now after back-off

`kill <pid>`
Stop autossh and its ssh child
SIGTERM, SIGINT and SIGKILL are treated as deliberate.

`pgrep -af autossh`
Find running autossh processes

`AUTOSSH_DEBUG=1 autossh -M 0 -v <ssh args>`
Debug autossh and ssh together

`ss -tlnp | grep <port>`
Check that the tunnel port is listening

If autossh exits at once, ssh failed in the first 30 s (gate time) or with status 1: run the ssh command alone to see the error.

Repeated quick failures make autossh back off, up to the poll time (10 minutes by default).

SERVICES

systemd unit

```
[Unit]
Description=Tunnel to <host>
After=network-online.target
Wants=network-online.target

[Service]
User=<user>
Environment=AUTOSSH_GATETIME=0
ExecStart=/usr/bin/autossh -M 0
-N \
  -o ServerAliveInterval=30 -o
ServerAliveCountMax=3 \
  -o ExitOnForwardFailure=yes \
  -L <lport>:localhost:<dport>
<host>
Restart=on-failure
RestartSec=10

[Install]
WantedBy=multi-user.target
```

SERVICES (cont.)

`systemctl enable --now autossh-<name>`
Start the unit now and at boot

`journalctl -u autossh-<name> -f`
Follow the unit's log

`@reboot AUTOSSH_GATETIME=0 autossh -M 0 -f -N <host>`
Crontab line to start at boot

Under systemd do not use -f: stay in the foreground so systemd can supervise the process.

Run the unit as a normal user with a passphrase-less key restricted on the server (command=, permitopen=).

RECIPES

Reach a remote database

```
autossh -M 0 -f -N \
  -o ServerAliveInterval=30 -o
ServerAliveCountMax=3 \
  -o ExitOnForwardFailure=yes \
  -L 5432:localhost:5432
<user>@<host>
psql -h localhost -p 5432
```

Reverse tunnel to a home machine

```
# on the home machine
autossh -M 0 -f -N \
  -o ServerAliveInterval=30 -o
ServerAliveCountMax=3 \
  -o ExitOnForwardFailure=yes \
  -R 2222:localhost:22
<user>@<vps>
# from the VPS
ssh -p 2222 <homeuser>@localhost
```

Persistent SOCKS proxy

```
autossh -M 0 -f -N -D 1080 \
  -o ServerAliveInterval=30 -o
ServerAliveCountMax=3 <host>
curl --socks5-hostname
localhost:1080 <url>
```

Debug a tunnel that will not stay up

```
AUTOSSH_DEBUG=1
AUTOSSH_LOGFILE=/tmp/autossh.log
\
  autossh -M 0 -N -v -L
8080:localhost:80 <host>
tail -f /tmp/autossh.log
```