SSH TUNNELLING FOR FUN AND PROFIT: AUTOSSH

By cytopia January 20, 2016 67 comments Administration, Command Line Fu autossh, ssh, ssh tunnelling

Now that you are able to create various forward or reverse SSH tunnels with lots of options and even simplify your live with `~/.ssh/config` you probably also want to know how make a tunnel persistent. By persistent I mean, that it is made sure the tunnel will always run. For example, once your ssh connection times out (By server-side timeout), your tunnel should be re-established automatically.

I know there are plenty of scripts out there which try to do that somehow. Some scripts use a while loop, others encourage you to run a remote command (such as tail) to make sure you don't run into timeout and various others. But actually, you don't want to re-invent the wheel and stick to bullet-proof already existing solutions. So the game-changer here is AutoSSH.

Article series

SSH tunnelling for fun and profit

1. Local vs Remote
2. Tunnel options
3. AutoSSH
4. SSH Config

TL;DR

```
autossh -M 0 -o "ServerAliveInterval 30" -o "ServerAliveCountMax 3" -L 5000:localhost:3306 cytopia@everythingcli.org
```

or fully configured (via `~/.ssh/config`) for background usage

```
autossh -M 0 -f -T -N cli-mysql-tunnel
```

What is AutoSSH

http://www.harding.motd.ca/autossh/README

> *Autossh is a program to start a copy of ssh and monitor it, restarting it as necessary should it die or stop passing traffic.*

Install AutoSSH

How to install AutoSSH on various systems via their package manager.

OS	Install method
Debian / Ubuntu	`$ sudo apt-get install autossh`
CentOS / Fedora / RHEL	`$ sudo yum install autossh`

OS	Install method
ArchLinux	`$ sudo pacman -S autossh`
FreeBSD	`# pkg install autossh` or `# cd /usr/ports/security/autossh/ && make install clean`
OSX	`$ brew install autossh`

Alternatively you can also compile and install AutoSSH from source:

```
wget http://www.harding.motd.ca/autossh/autossh-1.4e.tgz
gunzip -c autossh-1.4e.tgz | tar xvf -
cd autossh-1.4e
./configure
make
sudo make install
```

Note: Make sure to grab the latest version which can be found here: http://www.harding.motd.ca/autossh/.

Basic usage

```
usage: autossh [-V] [-M monitor_port[:echo_port]] [-f] [SSH_OPTIONS]
```

Ignore `-M` for now. `-V` simply displays the version and exits. The important part to remember is that `-f` (run in background) is not passed to the `ssh` command, but handled by `autossh` itself. Apart from that you can then use it just like you would use ssh to create any forward or reverse tunnels.

Let's take the basic example from part one of this article series (forwarding a remote MySQL port to my local machine on port 5000):

```
ssh -L 5000:localhost:3306 cytopia@everythingcli.org
```

This can simply be turned into an autossh command:

```
autossh -L 5000:localhost:3306 cytopia@everythingcli.org
```

This is basically it. Not much magic here.

Note 1: Before you use `autossh`, make sure the connection works as expected by trying it with `ssh` first.

Note 2: Make sure you use public/private key authentification instead of password-based authentification when you use `-f`. This is required for `ssh` as well as for `autossh`, simply because in a background run a passphrase cannot be entered interactively.

AutoSSH and `-M` (monitoring port)

With `-M` AutoSSH will continuously send data back and forth through the pair of monitoring ports in order to keep track of an established connection. If no data is going through anymore, it will restart the connection. The specified monitoring and the port directly above (+1) must be free. The first one is used to send data and the one above to receive data on.

Unfortunately, this is not too handy, as it must be made sure both ports (the specified one and the one directly above) a free (not used). So in order to overcome this problem, there is a better solution:

`ServerAliveInterval` and `ServerAliveCountMax` – they cause the SSH client to send traffic through the encrypted link to the server. This will keep the connection alive when there is no other activity and also when it does not receive any alive data, it will tell AutoSSH that the connection is broken and AutoSSH will then restart the connection.

The AutoSSH man page also recommends the second solution:

> *-M [:echo_port],*
> *…*
> *In many ways this [ServerAliveInterval and ServerAliveCountMax options] may be a better solution than the monitoring port.*

You can disable the built-in AutoSSH monitoring port by giving it a value of 0:

```
autossh -M 0
```

Additionally you will also have to specify values for `ServerAliveInterval` and `ServerAliveCountMax`

```
autossh -M 0 -o "ServerAliveInterval 30" -o "ServerAliveCountMax 3"
```

So now the complete tunnel command will look like this:

```
autossh -M 0 -o "ServerAliveInterval 30" -o "ServerAliveCountMax 3" -L 5000:localhost:3306 cytopia@everythingcli.or
```

Option	Description
ServerAliveInterval	ServerAliveInterval: number of seconds that the client will wait before sending a null packet to the server (to keep the connection alive). Default: 30
ServerAliveCountMax	Sets the number of server alive messages which may be sent without ssh receiving any messages back from the server. If this threshold is reached while server alive messages are being sent, ssh will disconnect from the server, terminating the session. Default: 3

AutoSSH and `~/.ssh/config`

In the previous article we were able to simplify the tunnel command via `~/.ssh/config`. Luckily autossh is also aware of this file, so we can still keep our configuration there.

This was our very customized configuration for ssh tunnels which had custom ports and custom rsa keys:

```
$ vim ~/.ssh/config
 Host cli-mysql-tunnel
    HostName      everythingcli.org
    User          cytopia
    Port          1022
    IdentityFile  ~/.ssh/id_rsa-cytopia@everythingcli
    LocalForward  5000 localhost:3306
```

We can also add the `ServerAliveInterval` and `ServerAliveCountMax` options to that file in order to make things even easier.

```
$ vim ~/.ssh/config
 Host cli-mysql-tunnel
    HostName      everythingcli.org
    User          cytopia
    Port          1022
    IdentityFile  ~/.ssh/id_rsa-cytopia@everythingcli
    LocalForward  5000 localhost:3306
    ServerAliveInterval 30
    ServerAliveCountMax 3
```

If you recall all the ssh options we had used already, we can now simply start the autossh tunnel like so:

```
autossh -M 0 -f -T -N cli-mysql-tunnel
```

AutoSSH environment variables

AutoSSH can also be controlled via a couple of environmental variables. Those are useful if you want to run AutoSSH unattended via `cron`, using shell scripts or during boot time with the help of `systemd` services. The most used variable is probably `AUTOSSH_GATETIME`:

> ***AUTOSSH_GATETIME***
> *How long ssh must be up before we consider it a successful connection. Default is 30 seconds. If set to 0, then this behaviour is disabled, and as well, autossh will retry even on failure of first attempt to run ssh.*

Setting `AUTOSSH_GATETIME` to 0 is most useful when running AutoSSH at boot time.

All other environmental variables including the once responsible for logging options can be found in the AutoSSH Readme.

AutoSSH during boot with systemd

If you want a permanent SSH tunnel already created during boot time, you will (nowadays) have to create a systemd service and enable it. There is however an important thing to note about systemd and AutoSSH: `-f` (background usage) already implies `AUTOSSH_GATETIME=0`, however `-f` is not supported by systemd.

> *http://www.freedesktop.org/software/systemd/man/systemd.service.html*
> *[…] running programs in the background using "&", and other elements of shell syntax are not supported.*

So in the case of `systemd` we need to make use of `AUTOSSH_GATETIME`. Let's look at a very basic service:

```
$ vim /etc/systemd/system/autossh-mysql-tunnel.service
[Unit]
Description=AutoSSH tunnel service everythingcli MySQL on local port 5000
After=network.target

[Service]
Environment="AUTOSSH_GATETIME=0"
ExecStart=/usr/bin/autossh -M 0 -o "ServerAliveInterval 30" -o "ServerAliveCountMax 3" -NL 5000:localhost:3306 cytop

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

Tell systemd that we have added some stuff:

```
systemctl daemon-reload
```

Start the service

```
systemctl start autossh-mysql-tunnel.service
```

Enable during boot time

```
systemctl enable autossh-mysql-tunnel.service
```

–

This is basically all I found useful about AutoSSH. If you thing I have missed some important parts or you know any other cool stuff, let me know and I will update this post.

Discussions

- SSH tunnelling for fun and profit: AutoSSH
  Hackernews
- r/commandline

–

‹ Convert PDF to MP4 SSH tunnelling for fun and profit: SSH Config ›

67 COMMENTS ON "SSH TUNNELLING FOR FUN AND PROFIT: AUTOSSH"

Pingback: SSH tunnelling for fun and profit: Tunnel options

Pingback: SSH tunnelling for fun and profit: local vs remote

Pingback: 1 – SSH tunnelling for fun and profit: AutoSSH

Pingback: === popurls.com === popular today

Steve Smith January 20, 2016 Reply

Depending on you usage you may also want to look at Mosh. It a UDP SSH alternative (but uses SSH for initial connection and authentication). The use of UDP enables true persistent roaming connections. It can also improve latency on lossy connections. Setup is mostly just a case of installing it on client and server.

https://mosh.mit.edu/

Tom Li January 20, 2016 Reply

No, I don't think Mosh is suitable for this use case, which is focused on using SSH as a SOCKS proxy to transport data. Mosh is a great tool for remote shell and system administration, but it doesn't have any ability to work as a proxy…

Peter Tripp January 20, 2016 Reply

When doing Remote Port forwards you'll also want to use: `-o ExitOnForwardFailure=yes`

Without this, if autossh detects the dropped connection before the remote end notices (likely) the subsequent reconnect will be unable to bind to your specified remote port because it's still bound to the stale SSH session. So the session will be up, but the reason for the session (the port forward) will be unavailable. With this extra option SSH immediately quits if that bind fails and autossh will just retry until it succeeds (after the remote end detects the stale session).

cytopia January 20, 2016 Reply

Thanks for the info. I wasn't aware of this flag.

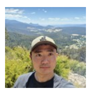
nicelydone May 23, 2017 Reply

@cytopia If you can its worth updating this post to include ExitOnForwardFailure. Such a crucial flag, which I originally missed

Pingback: Bookmarks for January 20th | Chris's Digital Detritus

Pingback: Weekendowa Lektura 2016-01-23 – bierzcie i czytajcie | Zaufana Trzecia Strona

Pingback: issue #12: Zabbix, GitLab, Tcpdive, Pact, Grafana, XKCD and many more - Cron Weekly: a weekly newsletter for Linux and Open Source enthusiasts


Francis Kim January 26, 2016 Reply

Cool trick! I'll be using this at work :3

Pingback: Links 26/1/2016: MPlayer 1.2.1, Parsix GNU/Linux 8.5 | Techrights

Pingback: AutoSSH intro | 0ddn1x: tricks with *nix

Pingback: NF.sec – Linux Security Blog - Raspberry PI – dostęp do skrzynki za mechanizmem NAT lub firewall

Sam Smith June 15, 2016 Reply

Ahh very helpful post. I'll be back for more tips in the future. Thanks for the useful share~

—

Sam Smith

Technology Evangelist and Aspiring Chef.

Large file transfers made easy.

http://www.innorix.com/en/DS

Pingback: SSH Basics and the SSH Config File – Richard Skumat's Website

Pingback: 内网服务器端口被外网访问的几种方法 – fingerection的技术博客

Pingback: Sensor network: postgresql on docker – OpenCoder

Pingback: 1月21日-每日安全知识热点 - 莹莹之色

Nuno Justo December 30, 2016 Reply

Great site... and well explained, but i've found a bug.

In the systemd service if you are doing a reverse ssh tunnel, flag for autossh should not be -NL but -NR

The rest is fine and well done.

Zack August 10, 2017 Reply

cant thank you enough been few days and couldn't get it to work with sysD

Stéphane April 17, 2019 Reply

also on the remote server (with the option -NR), you need to have GatewayPorts yes in the sshd_config, or it will bind tcp4 on 127.0.0.1 and not the external ip (ex: -NR *:remoteport:localhost:localport)

vlk January 14, 2017 Reply

very nice example especially with systemd service, but on my configuration (debian jessie) it was not working, the autossh was immediately killed by systemd.

there are my changes:

to prevent killing autossh after start is necessary to specify the service type and in case of autossh it is forking (Type=forking)
Environment="AUTOSSH_GATETIME=0″ is not necessary if you use parameter -f for autossh
I running autossh as normal user, so in section [service] add User=…

its look like this:

```
[Service]
User=me
Type=forking
ExecStart=/usr/bin/autossh -M 20100 -f mytunel -N
```

Ron October 26, 2020 Reply

if you read the article you will see that in the systemd service the -f flag is not used

Pingback: Reverse SSH tunnel | Webové stránky Jana Faixe

Pingback: Ssh into NATted VM via AutoSSH | Knowledge Base

Pingback: Túnel SSH | Monolito Nimbus

Vesel September 28, 2017 Reply

Good post.

Now I know a little bit more about Autossh

Thank you!

p.s.

Unfortunately the service script is missing something, because it won't boot ssh tunnel on my Ubuntu 16.04. at startup.

Saahib November 8, 2017 Reply

Awesome man.. really nice, precise and simple.

Saved my day :)

Pingback: Reverse tunnel | blackLabelWorkspace

Cesar April 17, 2018 Reply

What if the private key that I need to use isn't the default?… I can't get this to work when running systemctl start… I always get Host key verification failed, ssh exited with error status 255; restarting ssh.

Any help :c?

Andrejs May 25, 2018 Reply

then you can do something like this

'''

ExecStart=/usr/bin/autossh -M 0 -o "ExitOnForwardFailure=yes" -o "ServerAliveInterval 30" -o "ServerAliveCountMax 3" -NR 10022:127.0.0.1:22 user@jump.you.io -i /home/root/.ssh/id_rsa <– place where ssh key is

[Install]

WantedBy=multi-user.target

'''

Frieder August 23, 2018 Reply

Another option is to specify the user in the systemd file, so that the autossh command will run in the correct context:

```
[Service]
User=username
...
```

Pingback: Подключаемся к серверу за NAT при помощи туннеля SSH. Простая и понятная инструкция. | Многобукфф

FyLy August 23, 2018 Reply

autossh cannot handle ssh -E logfile option. For some debugging purpose, ssh -vvv can be very helpful, but this cannot be captured by autossh? (event increate LOGLEVEL?)

Sinhue Cuevas October 25, 2018 Reply

this is the most complete guide in SSH tunneling that I've seen over internet, congratulations and thank you

Pingback: Connecting to MySQL Remotely Using AutoSSH and SSH Tunneling - Amplitude Design, Inc.

Pingback: 内网穿透神器-Serveo – 前端开发，JQUERY特效，全栈开发，vue开发

Keavon Chambers February 13, 2019 Reply

If you are expecting to use your regular user's ~/.ssh config and keys, in your `.service` file under `[Service]` you need to include a `User=YOUR_USERNAME` line or else systemd will not use your configs and keys.

If things still aren't working, you may also need to ensure that you are including the `-N` flag along with your `-L` (local) or `-R` (remote) forwarding, or combined as `-NL` or `-NR`. The manpages for ssh specify that `-N` (the capitalized letter, not the lower-case which is a different flag) means:

> *Do not execute a remote command. This is useful for just forwarding ports (protocol version 2 only).*

One more method of troubleshooting is checking that `sudo systemctl status` shows your service, and whether its child processes include only a line for autossh or lines for both autossh and ssh. Only having autossh means that the ssh process failed to start, or was terminated after an unsuccessful connection (for example, having the wrong user's keys as described above).

Pingback: 試しにngrokからserveoに乗り換えてみた | Life Retrospective

Javier April 17, 2019 Reply

Great article! you really are a SSH ninja, thank you!

hexzample April 19, 2019 Reply

systemctl enable –now will enable and start the service in one command

Pingback: Serveo: Expose Local Servers to the Internet – TechBits

Pingback: Serveo: Expose Local Servers to the Internet - Elect Area

Pingback: Serveo: Expose Local Servers to the Internet - RAM NETWORK

Pingback: Serveo: Expose Local Servers to the Internet

Vikash Upadhyay July 2, 2019 Reply

Hi Team,

Its very good suggestion.

Jarno July 12, 2019 Reply

Absolutely superb article. Maybe the best-written HOWTO article ever.

ismail July 25, 2019 Reply

thanks verry much…

Kevin August 11, 2019 Reply

I had to add

```
-o StrictHostKeyChecking=no
```

to my

```
ExecStart=/usr/bin/autossh -M 0 -o "StrictHostKeyChecking=no" -o "ExitOnForwardFailure=yes" -o
"ServerAliveInterval 30" -o "ServerAliveCountMax 3" -i /home/kevin/.ssh/mykey.pem -NR
21388:localhost:22 ubuntu@35.21.10.50 -p 22
```

the first time in order to get the .ssh/known_hosts file to update and avoid an error message of “can’t open /dev/tty: No such device or address” that was showing up in syslog.

Lucian May 19, 2020 Reply

Thanks, you saved my day

Pingback: Public localhost, tại sao không ? • Học Lỏm

LabanM October 8, 2019 Reply

using localhost for the port redirect makes the connection super slow for IPv4 connections since it seems to try to use IPv6 first. Using 127.0.0.1 makes it much faster to connect.

This was very slow for me

```
-NR 10022:localhost:22
```

This way was much faster

```
-NR 10022:127.0.0.1:22
```

Teymur October 11, 2019 Reply

You are a pro

hsafe December 31, 2019 Reply

I enjoyed every bit of your post …awesome and thanks man

tbr January 23, 2020 Reply

Awesome article! Thank you very much! Helped me a lot!

Pingback: SSH Tunnelling – autossh – Stuff I'm Up To

Pingback: Bypassing the NAT – Reverse SSH tunnel with port forwarding! – ideaman924's blog

Harry March 4, 2020 Reply

Excellent in-depth article. Thank you.

Starc December 1, 2020 Reply

After trying to get this to work for Ubuntu 18 I discovered:

1.When using systemd you must omit the -f from the ExecStart, otherwise when autossh drops into the background systemd thinks the process quit and it just keeps trying to call it again, preventing autossh from functioning.

2. After=network.target should be changed to After=network-online.target so that this only attempts to run if its connected to a network (important for wifi-connected devices).

Pingback: grafana, prometheus로 OpenWrt 라우터 모니터링 하기 - qquack.org

نقل اثاث في دبي July 5, 2021 Reply

After=network.target should be changed to After=network-online.target so that this only attempts to run if its connected to a network (important for wifi-connected devices).

Pingback: SSHトンネルでLAN内のAndroid TVに外部からアクセス | あくまで暫定措置としてのブログ

Pingback: 通过 SSH 反向代理访问内网服务，并增强连接可靠性 | 7f - 柒风博客

Ben in Seattle March 24, 2022 Reply

This page is still a handy resource after all these years. I want to note that the `-M 0` flag has not been necessary if you are using Debian GNU/Linux or its descendants (Ubuntu, Mint, etc). Since 2004, Debian has automatically selected two free ports for monitoring. Personally, I also add in ClientAliveCountMax and ServerAliveInterval, as this site documents.

See http://bugs.debian.org/238150#msg6