SSH TUNNELLING FOR FUN AND PROFIT: TUNNEL OPTIONS

By cytopia January 13, 2016 10 comments Administration ssh, ssh tunnelling

If you have read the previous article of this series, you should be able to create forward and reverse tunnels with ease. In addition to the previously shown examples I will address some more advanced options for SSH tunnels in general.

Article series

SSH tunnelling for fun and profit

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

SSH Login shell

Remember the following example:

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

Once you have executed the above command, a tunnel is established. However, you will also be logged in into the remote server with a SSH session. If you simply want to do some port forwarding you will not need or might not even want a remote login session. You can disable it via `-N`, which is a very common option for SSH tunnels:

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

The `-N` option is also very useful when you want to create SSH tunnels via cron

Argument	Explanation
`-N`	After you connect just hang there (you won't get a shell prompt) **SSH man**: Do not execute a remote command. **Note**: Only works with SSHv2

So if you are not going to execute remote commands and will not need a login shell, you also do not need to request a pseudo terminal in the first place.

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

Argument	Explanation
`-T`	Disable pseudo-terminal allocation. This makes it also safe for binary file transfer which might contain escape characters such as `~C`.

SSH tunnel via cron

Imagine you want to have a SSH tunnel be established (or checked and if it doesn't run re-opened) via cron every hour. For that to work, SSH must go into background. For that we use `-f`.

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

Argument	Explanation
`-f`	Requests ssh to go to background just before command execution.

But hey, if SSH is in the background anyway, we do not need a login shell (`-N`) and therefore also do not need a tty (`-T`). So the full command ready for cron would be:

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

Note: Be aware that this example requires private/public key authentication as cron will not be able to enter passwords.

SSH tunnel on a non-standard port

What if the SSH server is listening on a non-standard port (not tcp22). You can always add a port option. Let's imagine SSH itself is listening on port 1022:

```
ssh -T -N -L 5000:localhost:3306 cytopia@everythingcli.org -p 1022
```

Argument	Explanation
`-p`	Port to connect to on the remote host.

SSH tunnel with a non standard private key

Let's assume you have many different private keys for different servers. If not explicitly specified, SSH will look for a file called `~/.ssh/id_rsa`. In this case however, your file is called `~/.ssh/id_rsa-cytopia@everythingcli`. So you will also pass this information to the tunnel command.

```
ssh -T -N -L 5000:localhost:3306 cytopia@everythingcli.org -i ~/.ssh/id_rsa-cytopia@everythingcli
```

SSH tunnel via SSH config

The most complex example from this tutorial is:

```
ssh -f -T -N -L 5000:localhost:3306 cytopia@everythingcli.org -p 1022 -i ~/.ssh/id_rsa-cytopia@everythingcli
```

We all are lazy-ass and don't want to type the whole thing every time we need a quick tunnel. This is where `~/.ssh/config` comes into play.

Adding user and host

```
$ vim ~/.ssh/config
 Host cli
    HostName      everythingcli.org
    User          cytopia
```

With this, we have created an alias `cli` for host `everythingcli.org` with user `cytopia`. Now our command can be written like this:

```
ssh -f -T -N -L 5000:localhost:3306 cli -p 1022 -i ~/.ssh/id_rsa-cytopia@everythingcli
```

Adding port and identity file

```
$ vim ~/.ssh/config
 Host cli
    HostName      everythingcli.org
    User          cytopia
    Port          1022
    IdentityFile  ~/.ssh/id_rsa-cytopia@everythingcli
```

Now the ssh command looks like this:

```
ssh -f -T -N -L 5000:localhost:3306 cli
```

Adding tunnel config

In the above example we have a generic configuration for the host `everthingcli.org` which will work for normal ssh connection as well as for establishing a tunnel. Let's copy all of the above block under a new alias `cli-mysql-tunnel` and add the tunnel specific configuration:

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

Now we can create the tunnel in a much shorter way:

```
ssh -f -T -N cli-mysql-tunnel
```

SSH tunnelling for fun and profit: local vs remote

Convert PDF to MP4

10 COMMENTS ON "SSH TUNNELLING FOR FUN AND PROFIT: TUNNEL OPTIONS"

Pingback: 2 – SSH tunnelling for fun and profit: Tunnel options

enrico villa January 14, 2016 Reply

About cronnig: I successfully use autossh since years, a totally reliable solution to setup a persistent tunnel

cytopia January 17, 2016 Reply

This will be covered in the next section.

Pingback: SSH tunnel options | 0ddn1x: tricks with *nix

Pingback: Links 17/1/2016: 4MLinux 16.0 Beta, Black Lab Linux 8 Alpha | Techrights

Pingback: SSH tunnelling for fun and profit: Autossh

Pingback: SSH tunnelling for fun and profit: SSH Config

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

Marcel November 30, 2016 Reply

Great article! Keep up the good =)

Greetings Marcel

Janae February 23, 2017 Reply

What a plsuraee to find someone who thinks through the issues

LEAVE A REPLY

Your email address will not be published.

Comment

Name Email Website

You can use GitHub Flavored Markdown to format your comment.

Pasting code:

```
```
code goes here
```
```

Post Comment