

# SSH TUNNELLING FOR FUN AND PROFIT: LOCAL VS REMOTE

 By cytopia  January 9, 2016  15 comments  Administration, Command Line Fu  ssh, ssh tunnelling

## Local vs Remote SSH port forwarding

When it comes to the art of SSH tunnelling, there are basically two options where to relay a port to.

You can relay a port from a remote server to your local machine with `ssh -L`, hence called **local port forwarding**. A very basic use-case is if your remote server has a MySQL database daemon listening on port `3306` and you want to access this daemon from your local computer.

The second option is to make your local port available on a remote server (`ssh -R`). **Remote port forwarding** might come in handy if you for example want to make your local web-server available on a port of a public server, so that someone can quickly check what your local web-server provides without having to deploy it somewhere publicly.

It should now be pretty easy to remember: **Local** and **remote** port forwarding always refers to **where to relay the port to**. The SSH command syntax uses the same easy to remember abbreviations: `-L` (forward to my local machine) and `-R` (forward to my remote machine).

Article series
<b>SSH tunnelling for fun and profit</b> <ul style="list-style-type: none"><li>1. Local vs Remote</li><li>2. <a href="#">Tunnel options</a></li><li>3. <a href="#">AutoSSH</a></li><li>4. <a href="#">SSH Config</a></li></ul>

## TL;DR

Remote MySQL server (remote port 3306) to local machine on local port 5000:

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

Local web-server (local port 80) to remote server on remote port 5000:

```
ssh -R 5000:localhost:80 cytopia@everythingcli.org
```

## Local port forwarding

(Make a remote port available locally).

In this example we are going to make a remote MySQL Server (Port 3306) available on our local computer on port 5000.

Let’s start with the general syntax of local port forwarding:

```
ssh -L <LocalPort>:<RemoteHost>:<RemotePort> sshUser@remoteServer
```

Argument	Explanation
LocalPort	The port on your local machine where the whole thing should be reachable.
RemoteHost	This specifies on which interface inside the remote server ( remoteServer ) the daemon is listening on. This can be either 127.0.0.1 , localhost , a specific IP address or even 0.0.0.0 which refers to all interfaces. If you are unsure, simply ssh into the remote machine and check all interfaces for port 3306 by issuing: netstat -an   grep 3306   grep LISTEN .
RemotePort	This is the actual port on the remote machine ( remoteServer ) you want to relay to your local machine. In our case (MySQL listens on 3306 by default) it is simply 3306
sshUser	This is the SSH username you have on the remote server
remoteServer	The address (IP or hostname) by which your remote server is reachable via ssh

Now let’s simply forward our remote MySQL server to our local machine on port 5000 .

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

That’s all the magic! You can now simply reach the remote database from your local machine with mysql --host=127.0.0.1 --port=5000 or any other client.

But wait... which local address does it listen on?

Yes, you are right! The complete syntax is:

```
ssh -L [<LocalAddress>]:<LocalPort>:<RemoteHost>:<RemotePort> sshUser@remoteServer
```

Argument	Explanation
LocalAddress	The local address is an optional parameter. If you do not specify it, the remote port will be bound locally to all interfaces ( 0.0.0.0 ). So you can also only bind it locally to your 127.0.0.1 (on your local machine).

This is the full example:

```
ssh -L 127.0.0.1:5000:localhost:3306 cytopia@everythingcli.org
```

## Remote port forwarding

(Make a local port available remotely).

In this example we are going to make our local web-server (Port 80) available on a remote server on Port 5000.

Let’s start with the general syntax of remote port forwarding:

```
ssh -R <RemotePort>:<LocalHost>:<LocalPort> sshUser@remoteServer
```

Argument	Explanation
RemotePort	The port on your remote server ( remoteServer ) where the whole thing should be reachable.

Argument	Explanation
LocalHost	This specifies on which interface inside your local computer the daemon is listening on. This can be either <code>127.0.0.1</code> , <code>localhost</code> , a specific <code>IP address</code> or even <code>0.0.0.0</code> which refers to all interfaces. If you are unsure, simply check all interfaces (on your local machine) for port <code>80</code> by issuing: <code>netstat -an   grep 80   grep LISTEN</code> .
LocalPort	This is the actual port on your local machine you want to relay to the remote server ( <code>remoteServer</code> ). In our case (The web-server listens on 80 by default) it is simply <code>80</code>
sshUser	This is the SSH username you have on the remote server
remoteServer	The address (IP or hostname) by which your remote server is reachable via ssh

Now let’s simply forward our local web-server to our remote machine on port `5000` .

```
ssh -R 5000:localhost:80 cytopia@everythingcli.org
```

That’s all the magic! You can now simply reach your local webserver via <http://everythingcli.org:5000>.

But wait... which remote address does it listen on?

Yes, you are right! The complete syntax is:

```
ssh -R [<RemoteAddress>]:<RemotePort>:<LocalHost>:<LocalPort> sshUser@remoteServer
```

Argument	Explanation
RemoteAddress	The remote address is an optional parameter. If you do not specify it, the remote port will be bound remotely (on <code>remoteServer</code> ) to all interfaces ( <code>0.0.0.0</code> ). So you can also only bind it remotely to a specific interface.

This is the full example:

Assuming the IP address of everythingcli.org is 109.239.48.64 and you only want to bind it to this IP.

```
ssh -R 109.239.48.64:5000:localhost:80 cytopia@everythingcli.org
```

But wait... it doesn’t work

By default, the listening socket on the server will be bound to the loopback interface only. This may be overridden by specifying `RemoteAddress` . Specifying a `RemoteAddress` will only succeed if the server’s `GatewayPorts` option is enabled (on the remote server):

```
$ vim /etc/ssh/sshd_config
GatewayPorts yes
```

# Some more details

## Ports below 1024

Every system user can allocate ports above and including 1024 (high ports). Ports below that require root privileges. So If you want to relay any port to a port to for example 10, you must do that like so:

As you allocate a low port on your local machine, you must either do that as root (locally) or with sudo (locally):

```
sudo ssh -L 10:localhost:3306 cytopia@everythingcli.org
```

As you allocate a low port on the remote server, you will need to ssh into the machine as root:

```
ssh -R 10:localhost:80 root@everythingcli.org
```

## Summary

Article Name	SSH tunnelling for fun and profit: local vs remote port forwarding
Description	SSH Tunnel - Local and Remote Port Forwarding Explained.
Author	cytopia

[← check\\_drupal: Monitoring drupal with nagios](#)

[SSH tunnelling for fun and profit: Tunnel options >](#)

15 COMMENTS ON “SSH TUNNELLING FOR FUN AND PROFIT: LOCAL VS REMOTE”



PlatonIsGoodAtCoding January 9, 2016 [Reply](#)

Good article. How about adding a section talking about the -D option ?



cytopia January 9, 2016 [Reply](#)

Thanks for the proposal. This is indeed intended to be a very basic introduction of coming the series.

**Pingback:** [Links 10/1/2016: Linux Mint 17.3 “Rosa” Xfce and KDE Released | Techrights](#)

**Pingback:** [Local and remote \(SSH\) port forwarding | Oddn1x: tricks with \\*nix](#)

**Pingback:** [SSH tunnelling for fun and profit: Tunnel options](#)

**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 Blog](#)



Richard February 8, 2016 [Reply](#)

Excellent series. I've linked to you from my blog.

I've still got to read all of your posts on ssh tunneling, but it helped an awful lot a few weeks ago.

Thanks.




cytopia February 9, 2016 [Reply](#)

Hi Richard,

nice to hear that you liked it and thanks for the credit. I ping-backed your article links.

**Pingback:** Oku – Hesap, Kitap ve Hayat

**Pingback:** Creating a local socks proxy for Firefox – Richard Skumat's Website



Sam Smith   June 15, 2016   [Reply](#)

Excellent. Just what I was looking for. Appreciate the concise writing as well. Cheers~


—

Sam Smith

Technology Evangelist and Aspiring Chef.

Large file transfers made easy.

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



Gagan   March 28, 2021   [Reply](#)

I have been trying to understand the SSH concept but with almost no luck.

This is gold.

Thanks

**Pingback:** [pwncat - Netcat on Steroids with Firewall, IDS/IPS Evasion, Bind and Reverse Shell, Self-Injecting Shell and Port Forwarding Magic — SkyNet Tools](#)

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

Developed by Think Up Themes Ltd. Powered by Wordpress.