

Set Up SSH Tunneling on a Linux / Unix / BSD Server To Bypass NAT

Article Number: 673 | Rating: Unrated | Last Updated: Fri, May 15, 2020 11:42 AM

I'm a new Linux / Unix system user. How can I set encrypted tunnel between my desktop/laptop computer and server in a remote data center to bypass the limits in a network? How do I create a reverse SSH tunnel on Unix-like systems?

SSH tunnelling can be thought as a poor-man's-VPN. It is handy in situations where you would like to hide your traffic from any body who might be listening on the wire or eavsdropping.

You can use such tunnel between your computer and your Unix/BSD/Linux server to bypass limits placed by a network or to bypass NAT, and more.

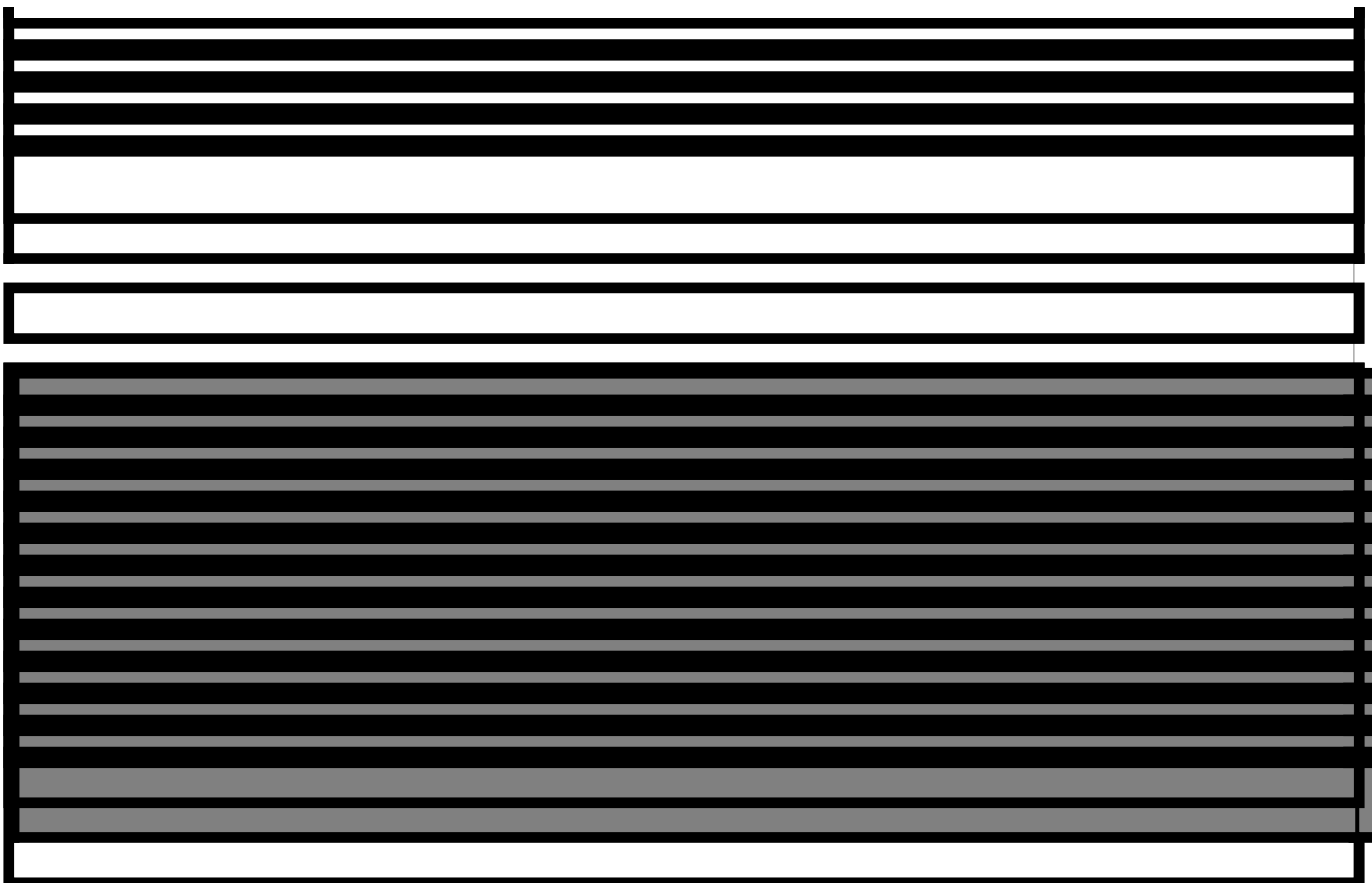
More about the Internet protocol, ports, tcp and udp

The Internet protocol is nothing but a set of rules for sending information between your desktop and the server on the Internet (or WAN or Lan). Each computer at least has one IP address. Each rule is like a language and in computer science you call it as a protocol. One can use protocols for communication over the Internet.

Common application protocol

For example, you can use HTTP (Hypertext Transfer Protocol) or HTTPS (Hypertext Transfer Protocol Secure) protocol to view images or download files from the Internet. You can use DNS (Domain Name System) protocol translates domain names such as `www.cyberciti.biz`, which can be easily memorized by humans, to the numerical IP addresses such as `75.126.153.206` and vice versa. You can use `ssh` (Secure Shell) for secure data communication, remote command-line login, remote command execution, and more.

Our sample setup



Where,

- randomhost.net – You have an accounts on this Linux/Unix based server.
- client1.cyberciti.biz – Your private desktop/laptop computer that you use to connect to server1.cyberciti.biz server. You need to use loopback interface with the IP address 127.0.0.1. Only apps installed on the desktop such as browser, irc client, email client and more have access to 127.0.0.1.

Example: SSH tunnel for an IRC client

A tunnel between local port 8888 on the local interface (IP 127.0.0.1) and the IRC server at irc.freenode.net, bound to a remote machine's port 6667. You are going connect to it using the loopback interface:

```
ssh -L 8888:127.0.0.1:8888 user@remote-machine
```

```
ssh -L 8888:127.0.0.1:8888 user@remote-machine
```

```
ssh -L 8888:127.0.0.1:8888 user@remote-machine
```

If you login to your shell account with: `ssh myuser@randomhost.net` for SSH tunnelling you have to add additional arguments. It goes like this:

```
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net
```

If the server/shell account you are using to tunnel through is listening on a different port, for example 2745, it would be written like this:

```
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net  
ssh -L 2745:irc.freenode.net:6667 myuser@randomhost.net
```

This will open a tunnel between your computer to `irc.freenode.net` through your

shell account on randomhost.net. All traffic will go through your shell account box as encrypted SSH traffic before reaching irc.freenode.net. Your computer now acts like a irc server listening to port 8888. Replace it with any ports you want above 1024 to avoid conflict.

To connect to your local port as if it's irc server. On irssi this would be:

```
server 127.0.0.1 8888
servername irc.freenode.net
nick nick
servername irc.freenode.net
server 127.0.0.1 8888
servername irc.freenode.net
nick nick
servername irc.freenode.net
server 127.0.0.1 8888
servername irc.freenode.net
nick nick
servername irc.freenode.net
```

This will also apply for any other irc clients such as X-Chat. Use 127.0.0.1/8888 for server name and you are good to go. Other fields remain the same.

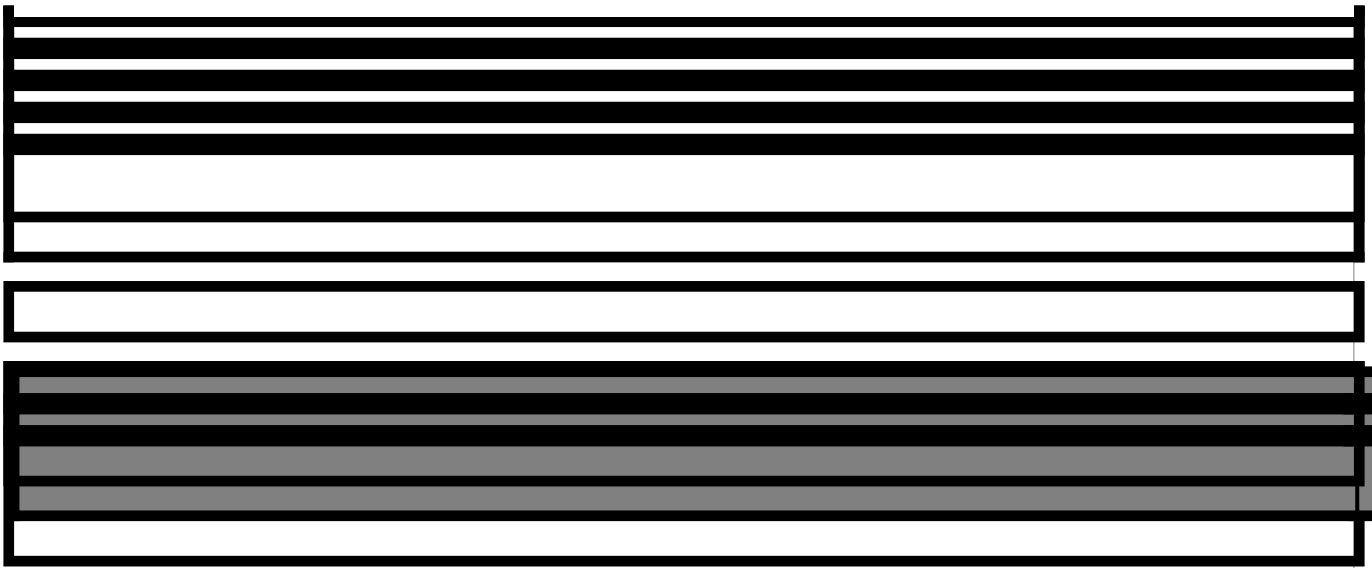
Howto setup a reverse SSH tunnel

In a scenario where a machine is behind NAT or company firewall a normal SSH tunnel won't work. To overcome this, we have to make use of reverse SSH tunnel. To achieve this, you need an internet reachable box along with the machine behind NAT/firewall. During this guide we will call the machine behind NAT/firewall a NATbox and internet reachable machine an OPENbox.

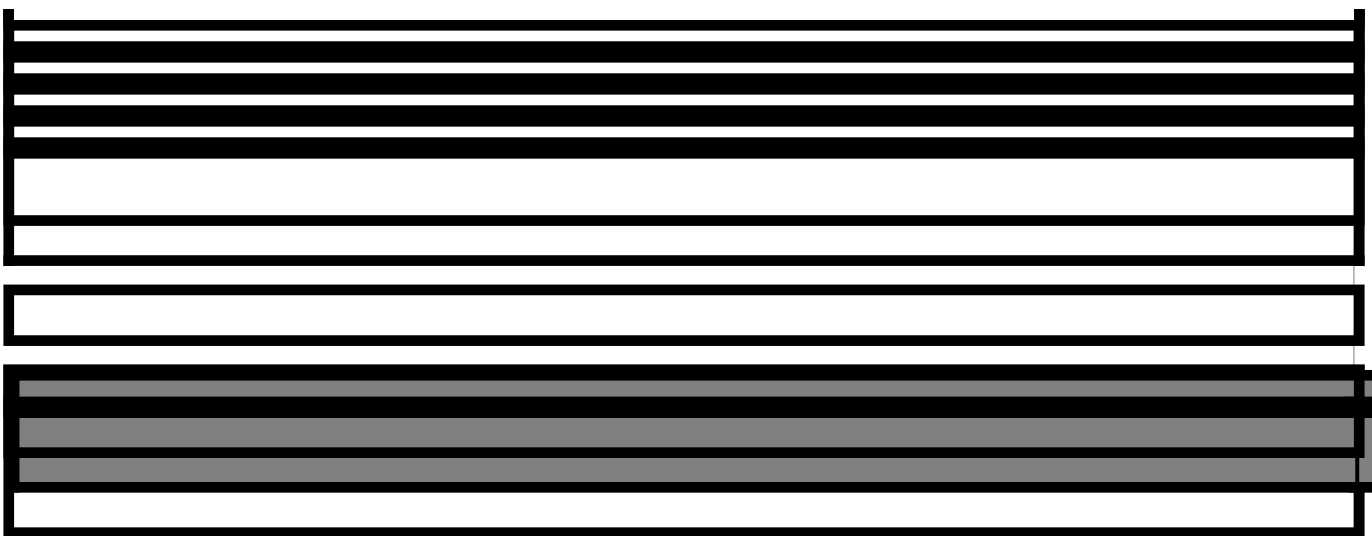
For reverse SSH Tunnel, there are basically three ports involved. One is the SSH

port of workstation, we use it forward the reverse tunnel port into it. The second, is the reverse tunnel port which gets forwarded to workstation's SSH port. The third, is the SSH port of the public box, we need that port to SSH into public box.

From outside if you use public box's SSH port, you log in to that box. If you use reverse tunnel port you get forwarded to workstation. You must be careful about usernames when doing this.



To establish reverse tunnel from workstation, you will use:



And setup the reverse tunnel on a different random port and forward it to local SSH port.

Example

So, let us consider, OPENbox is listening to SSH on port 1234. Type following on NATbox:

```
ssh -R 2222:192.168.1.1:22 192.168.1.2:1234
```

This will initiate a connection from behind the NATd/firewalled box to the publicly reachable box listening to SSH on port 1234. Once the connection is established it will create a reverse tunnel at port 22 for the remote party to connect and get in. Type the following command on OPENbox:

```
ssh -R 2222:192.168.1.1:22 192.168.1.2:1234
```


```
[REDACTED]
```

Since the NATd/firewalled box has an established connection to OPENbox, the tunnel will go through the same channel. In addition, type the following from anywhere else to access NATbox which will tunnel the traffic through OPENbox:

```
[REDACTED]
```

This requires an additional setup on the OpenSSH server, add the lines to/etc/ssh/sshd_config

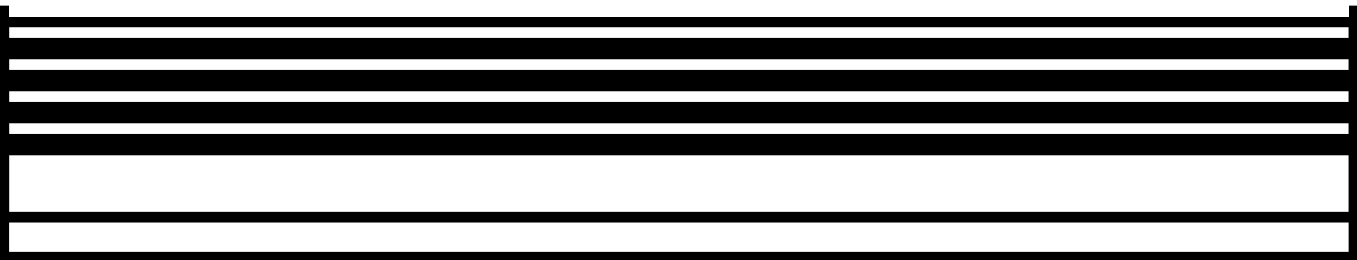
```
[REDACTED]
```



Save and close the file. Make sure you restart/reload the SSHD on the remote server.

Summary

The syntax is as follows to access remote server port without modifying firewall settings:



Where,

1. **-f** : Requests ssh to go to background just before command execution.
2. **-L port** : Specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side.
3. **-p port** : Port to connect to on the remote host.
4. **-R** : Specifies that the given port on the remote (server) host is to be forwarded to the given host and port on the local side.
5. **-D port**: Specifies a local “dynamic” application-level port forwarding.
6. **-C** : Requests compression of all data. This is useful for speeding up connection.
7. **-q** : Quiet mode. Causes most warning and diagnostic messages to be suppressed.
8. **-N**: Do not execute a remote command. This is useful for just for warding ports.

Posted - Fri, May 15, 2020 11:42 AM. This article has been viewed 12112 times.

Online URL: <http://kb.ictbanking.net/article.php?id=673>