

# SSH Tunnel

## Forwarding

In this example, the remote port 3306 on the remote server will be forwarded to local port 3307 using `login@remoteserver` for ssh credentials. The connection is pushed in the background and the port will remain forwarded as long as this ssh connection remains active.

```
ssh login@remote-server -L 3307:127.0.0.1:3306 -N &
```

Multiple ports can be forwarded by simply repeating the `-L` command:

```
ssh remote-host -L 8822:REMOTE_IP_1:22 -L 9922:REMOTE_IP_2:22
```

## Full Tunnel

SSH has a `-w` option which will set up `tun` devices on either end and transport the traffic between them. If you say `ssh -w 0:0 hostname`, it will set up a `tun0` on both ends. Then you just need to `ifconfig` the two tunnel endpoints. You can use the `LocalCommand` setting to do one `ifconfig` and the remote command to do the other, for example:

```
ssh -o PermitLocalCommand=yes \  
-o LocalCommand="ifconfig tun0 192.168.0.1 netmask 255.255.255.252" \  
-w 0:0 $HOSTNAME \  
'ifconfig tun0 192.168.0.2 netmask 255.255.255.252; sleep 900000'
```

Note that on the server you will need to set `PermitTunnel yes` in `/etc/ssh/sshd_config` and restart SSH. This needs to run as root on both ends to build the tunnel interface.

From:  
<http://wuff.dyndns.org/> - **Wulf's Various Things**

Permanent link:  
<http://wuff.dyndns.org/doku.php?id=linux:ssh-tunnel&rev=1527325067>

Last update: **2023/05/29 11:53**

