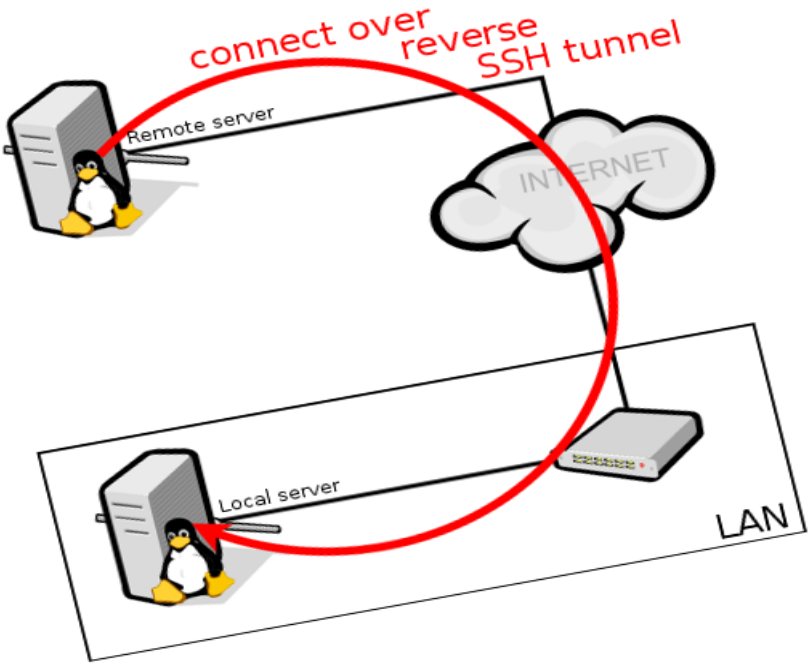
HOW TO CREATE PERSISTENT REVERSE SSH TUNNEL

Sometimes I want to access private server at home from different network while being on the go. The easiest way to do this is to use auto ssh utility to create secure and persistent reverse SSH tunnel to the publicly available personal server.



First step

Connect to the **remote server** and create `sshtunnel` user.

```
$ sudo useradd -s /usr/sbin/nologin -m sshtunnel
```

It is very important to disable shell access, but create home directory to store SSH configuration (*known hosts* and *authorized keys*).

Second step

Create `sshtunnel` user on the **local machine**.

```
$ sudo useradd -s /usr/sbin/nologin -m sshtunnel
```

Switch to the `sshtunnel` user and create SSH key pair.

```
$ sudo su sshtunnel -s /bin/bash

sshtunnel$ ssh-keygen -t rsa -b 2048 -q -N "" -f ~/.ssh/tunnel_key_a
```

Third step

Upload `~/.ssh/tunnel_key_a.pub` file to the **remote server** and perform the following operations while still being connected to it.

Create missing `.ssh` directory.

```
$ sudo su -s /bin/sh sshtunnel -c "mkdir ~/.ssh"
```

Add uploaded public key to the pool of keys authorized for authentication.

```
$ echo 'no-agent-forwarding,no-user-rc,no-X11-forwarding,no-pty' $(cat tunnel_key_a.pub) |

sudo su -s /bin/bash sshtunnel -c "tee >> ~/.ssh/authorized_keys"
```

Now you can remove uploaded public key.

Fourth step

Add **remote server** to the known servers pool on the **local machine**.

```
$ ssh-keyscan -H -t rsa remote-server | sudo su -s /bin/sh sshtunnel -c "tee >>
~/.ssh/known_hosts"
```

Fifth step

Use the following command on the **local machine** to test reverse SSH tunnel to **remote server**.

```
$ sudo su -s /bin/sh sshtunnel -c "autossh -v -i ~/.ssh/tunnel_a remote-server -N -R
9000:localhost:22"
OpenSSH_6.7p1 Debian-3, OpenSSL 1.0.1j 15 Oct 2014
debug1: Reading configuration data /etc/ssh/ssh_config
[..]
Authenticated to remote-server ([111.222.333.444]:22).
[..]
debug1: remote forward success for: listen 60654, connect 127.0.0.1:60655
debug1: remote forward success for: listen 9000, connect localhost:22
debug1: All remote forwarding requests processed
```

Sixth step

Add `autossh` command to the `/etc/rc.local` script to start ssh tunnel at boot.

```
$ sudo sed -i -e '$i # create sshtunnel\nsu -s /bin/sh sshtunnel -c "autossh -f -i

~/.ssh/tunnel_key_a remote-server -N -R 9000:localhost:22"\n' /etc/rc.local

$ cat /etc/rc.local

#!/bin/sh -e

#

# rc.local

#

# This script is executed at the end of each multiuser runlevel.

# Make sure that the script will "exit 0" on success or any other

# value on error.

#

# In order to enable or disable this script just change the execution

# bits.

#

# By default this script does nothing.


# create sshtunnel

su -s /bin/sh sshtunnel -c "autossh -f -i ~/.ssh/tunnel_key_a remote-server -N -R

9000:localhost:22"
```

```
exit 0
```

Now you should be able to connect to the SSH service on **local machine** from **remote server**using port `9000`, while `autossh` is managing SSH tunnel.

```
remote-server$ ssh -l milosz -p 9000 localhost
```

Source: https://blog.sleeplessbeastie.eu/2014/12/23/how-to-create-persistent-reverse-ssh-tunnel/