

Host a Tor Relay

If you want to support the Tor project and you have some bandwidth to share (at least 10 Mbps in both directions, i.e. download and upload) you might consider hosting a Tor (Non-Exit) Relay. There are no legal implications as only fully encrypted Tor traffic is coming in outgoing on your internet connection. It can be done quite easily with a Raspberry Pi (needs at least 2 GB of RAM) or any similar hardware as described below.

Our assumption is that we connect the Raspberry Pi with an ethernet cable to our ISP router, it is running on Manjaro (Minimal version, also called ,headless' i.e. no graphical user interface).

Install Tor

This is straightforward, just install it with:

```
sudo pacman -Syu tor
```

then we need to provide the configuration for it (keeping the default file as reference):

```
sudo mv /etc/tor/torrc /etc/tor/torrc.default  
sudo nano /etc/tor/torrc
```

Our configuration for a Tor Relay will be something similar to:

```
NickName MyNewTorRelay  
ContactInfo myemail@example.org
```

```
User tor  
AvoidDiskWrites 1  
DataDirectory /var/lib/tor  
Log notice file /var/log/tor/tor.log
```

```
ORPort 4020 IPv4Only
```

```
ExitRelay      0
SocksPort      0
```

```
RelayBandwidthRate 42 Mbit
RelayBandwidthBurst 48 Mbit
```

The contact information is optional but might be quite handy for others if there should be something strange with the relay. It doesn't have to be an email address but could be any kind of text.

In case you are running more than just one Tor Relay you have to also include a „MyFamily“ option in the config above and list all your key-fingerprints of your Tor Relays in each of the torrc config files. You get the fingerprint with

```
sudo -u tor tor -list-fingerprint
```

and remember that there must be a \$ (Dollar-sign) at the beginning of each fingerprint.

Crucial: One might need to create the directories in the config file and make them owned by tor – they should look like:

```
drwx----- 5 tor  tor 4096 Dec 31 09:47 tor
```

You should check that the syntax of the config file is correct with:

```
sudo -u tor tor --verify-config
```

Port Forwarding on ISP Router

In our example the Raspberry Pi (our Tor Relay) sits behind a router which is the gateway into the internet (often provided by the ISP). With the tor configuration above we need to establish port forwarding on this internet router, so TCP traffic coming from the internet (on port 4020) is forwarded to the Tor Relay (on the same ports).

If you would like to use other ports to the outside world (internet) than on the Tor Relay server itself, the Tor config file (torrc) needs to have something like:

```
ORPort 80 NoListen  
ORPort 4020 NoAdvertise
```

The port forwarding on the ISP router then obviously has to forward port 80 to port 4020 on the Tor Relay.

The ports chosen are kind of arbitrary and we are free to take whatever we like. One advantage of advertising (i.e. using) ports 80 towards the internet is that they are very unlikely to be blocked as they are usually taken for http and https traffic. The drawback is that you can't use these ports for something else (like a web-presence). Also some routers seem to have issues with port-forwarding these ports (e.g. lost after a router-reboot).

The details on how port forwarding is configured on the internet router depends heavily on that device but usually each of these kind of routers offers this feature somehow (just search the internet in case this is not obvious).

Start and Test it!

First let's start Tor (so it picks up the latest configuration):

```
sudo systemctl start tor
```

Check the logs for what Tor does and if it complains about anything – the following commands might be useful to check for any errors:

```
sudo systemctl status tor.service  
sudo cat /var/log/tor/tor.log  
journalctl | grep Tor
```

You are perfectly fine if you see something like „*Self-testing*

indicates your ORPort is reachable from the outside„. If there are no issues your new Tor Relay will also become visible on the `torproject` `metrics-webpage` at `metrics.torproject.org/rs.html` (this might take a few hours though, so be patient).

One could also increase the level of logging information written by tor. Just change the option in the `/etc/tor/torrc` configuration file – after the „log“ statement one could place either `debug`, `info`, `notice`, `warn`, or `err`. Additionally, one could (temporarily, for debugging) turn off the scrubbing of sensitive information in the log-files as well. So for debugging include something like the following in the `torrc`

```
SafeLogging 0
Log info file /var/log/tor/tor.log
```

Once running fine one should keep the logging at the 'notice' level though.

To permanently enable Tor running it needs to be enabled (so it will be started automatically after a reboot):

```
sudo systemctl enable tor
```

Also note that it takes up to 2 months until a new Tor Relay gets fully used – and since there is not always traffic available it will mostly never really run at the full possible bandwidth. See this article for some background on it: blog.torproject.org/lifecycle-new-relay.

Backup of Tor's keys

If you want to be able to continue with the same Relay identity on another server (e.g. when moving servers or the server dies) one needs two key-files:

```
/var/lib/tor/keys/ed25519_master_id_secret_key
/var/lib/tor/keys/secret_id_key
```

If you ever set up a new Tor relay just overwrite the automatically generated key with these old ones and your new relay has the same identity as before.

NextCloud over Tor (onion service)

Using a cheap Raspberry Pi it's quite easy for someone with some general linux knowledge to set up an own nextcloud server which is available on the Tor network.