Configure *monerod* as an I2P Hidden Service

24 June 2019

author: prefers to remain anonymous

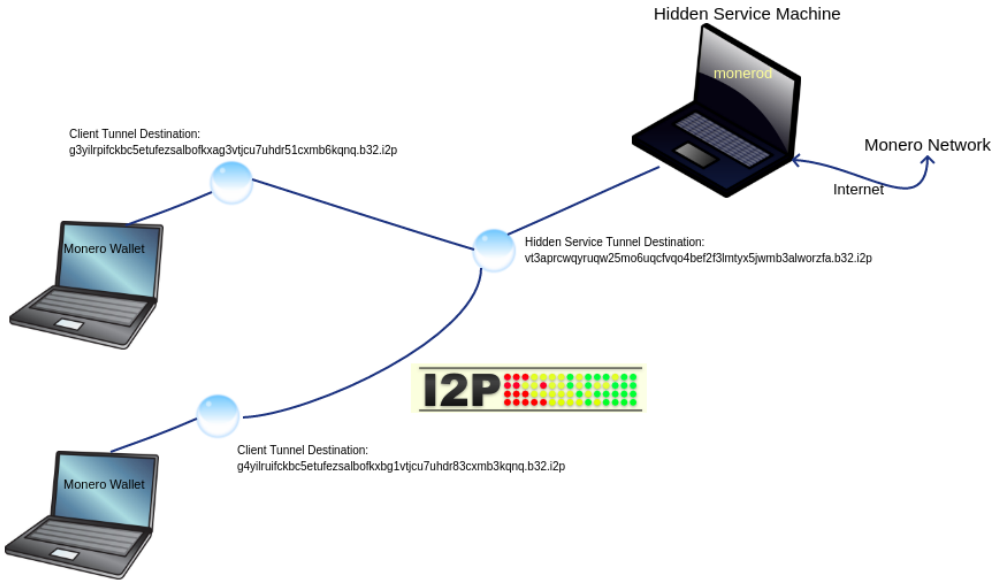
Introduction

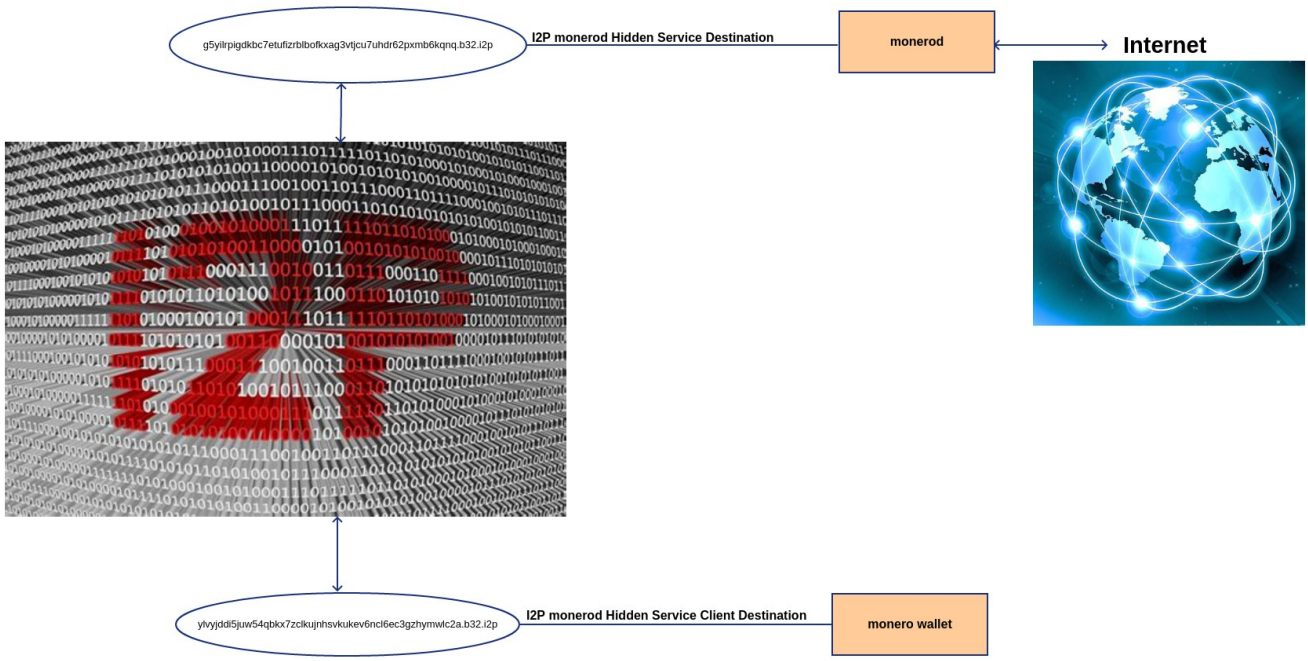
This paper will describe how to configure the Monero daemon (monerod) as an I2P hidden service. The value-added here is to provide an access path from Monero wallets to a remote Monero full node while hiding the Monero wallet machine's IP address. Hiding the Monero wallet client's IP address helps to maintain anonymity of Monero transactions at the network layer.

Since monerod needs to connect to the global Monero network to maintain an up-to-date blockchain along with the ability to process transactions, there are risks of exposing the IP address of the machine running the monerod service to prying eyes. There are a multitude of approaches to mitigating this risk which will be discussed in subsequent papers which will include the use of VPNs, Tor, and eventually the Kovri project which will integrate Monero with the I2P network.

Even though the machine running the I2P monerod hidden service incurs the risk of IP address exposure, this risk does not fall on the clients (wallets) that use the hidden service. Providing access to the I2P monerod hidden service to multiple clients provides plausible deniability to users of the hidden service.

The diagrams below illustrates the domain space for this paper.





The I2P monerod hidden service communicates with client wallets through I2P tunnels. The tunnel endpoints are cryptographic constructs called *tunnel destinations (not IP address:port number). All data transported via I2P tunnels are encrypted end-to-end. For additional information regarding the I2P protocol see:* https://geti2p.net/en/docs/how/intro

The remainder of this paper will describe:

- Installing and running monerod on the “hidden service” computer
- Installing and running the I2P router on the “hidden service” computer
- Configuring the I2P monerod hidden service
- Configuring I2P client tunnels to use the I2P monerod hidden service
- Connecting wallet(s) to the I2P monerod hidden service

All of the computers used to provide the I2P monerod hidden service described in this paper were running a 64-bit Linux operating system.

Installing and Running *monerod* on Hidden Service Computer

First you need to download the latest Monero software package. Goto https://getmonero.org/downloads/

and download the proper package for your computer. Since we are using a 64-bit Linux computer for our hidden service machine, we are going to download the **Linux 64-bit Command-Line Tools Only** software package as shown in the image below.

Extract the downloaded file, then go to the directory in a terminal and type the following command:

./monerod

If everything was done successfully, the Monero daemon will be running and displaying output on your terminal screen. Please consult https://getmonero.org/ for specific information regarding the proper installation of the Monero software.

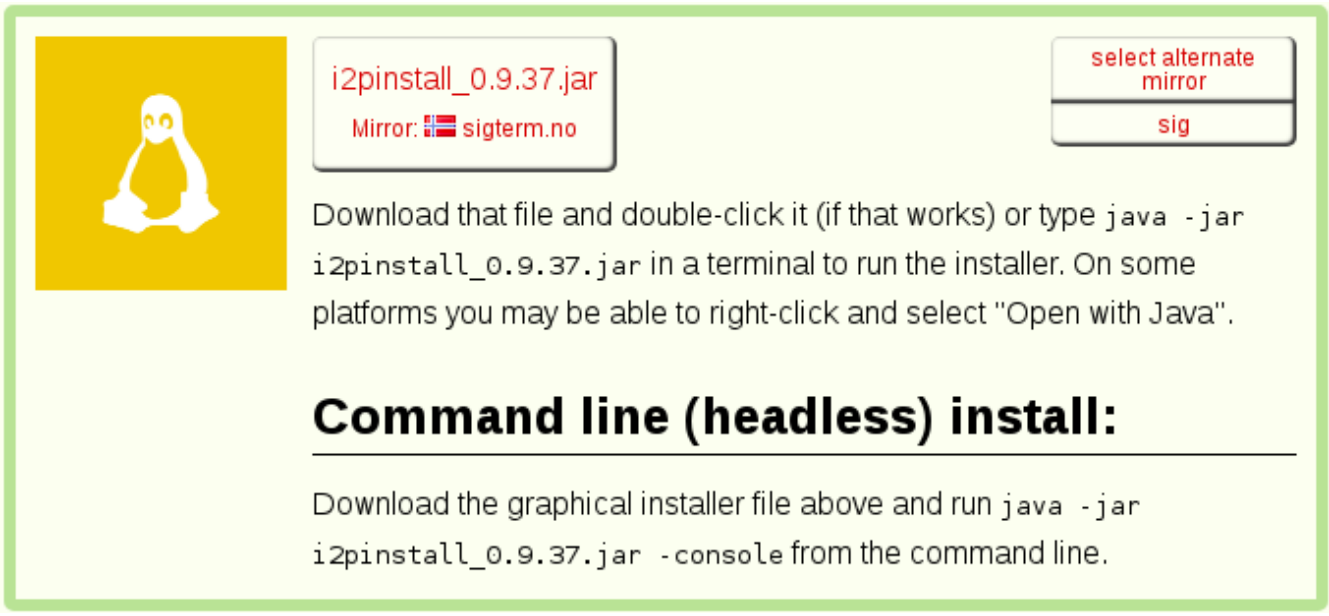
Installing and Running I2P Router on Hidden Service Computer

You must have a Java runtime environment on your computer to run the I2P router. I used the **openjdk-8-jre-headless** package. To install on Linux (Debian/Ubuntu) I ran the following commands:

sudo apt-get update
sudo apt-get install openjdk-8-jre-headless

When the java runtime environment is installed, download and install the I2P router from: https://geti2p.net/en/download

Follow the installation instructions on the I2P download page.



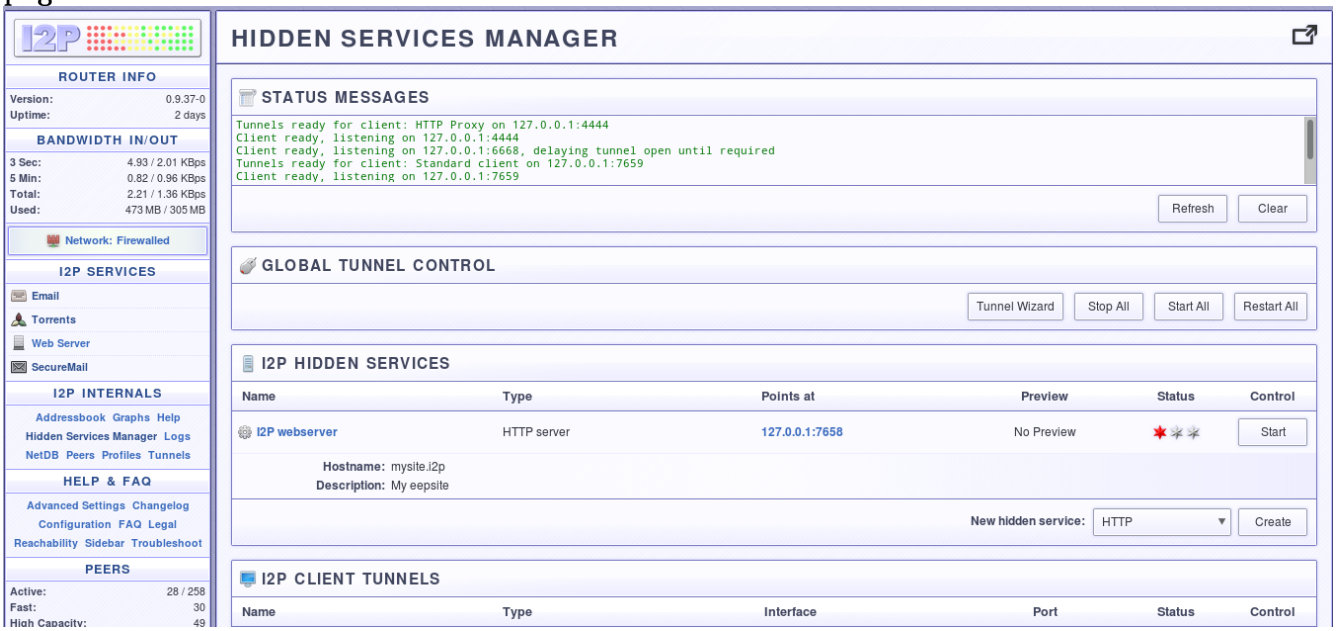
After installation, start the I2P router by going to the directory where I2P is installed and enter:

./i2prouter start

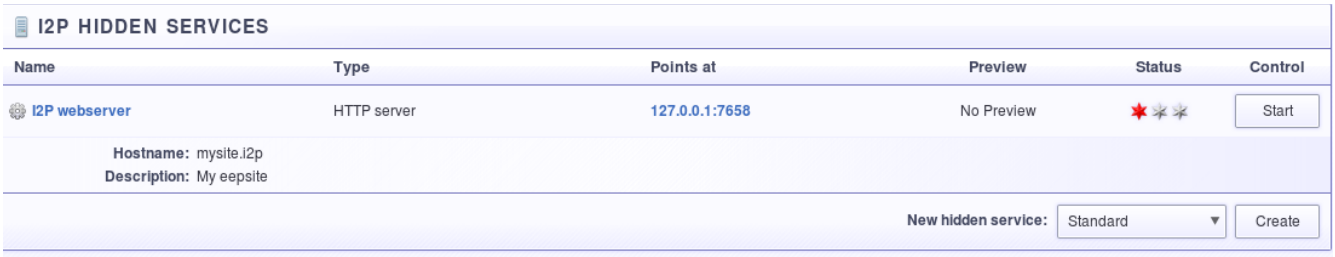
Configuring the I2P monerod Hidden Service

Go to the I2P Hidden Services Manager (on your I2P router machine) at:
http://127.0.0.1:7657/i2ptunnelmgr

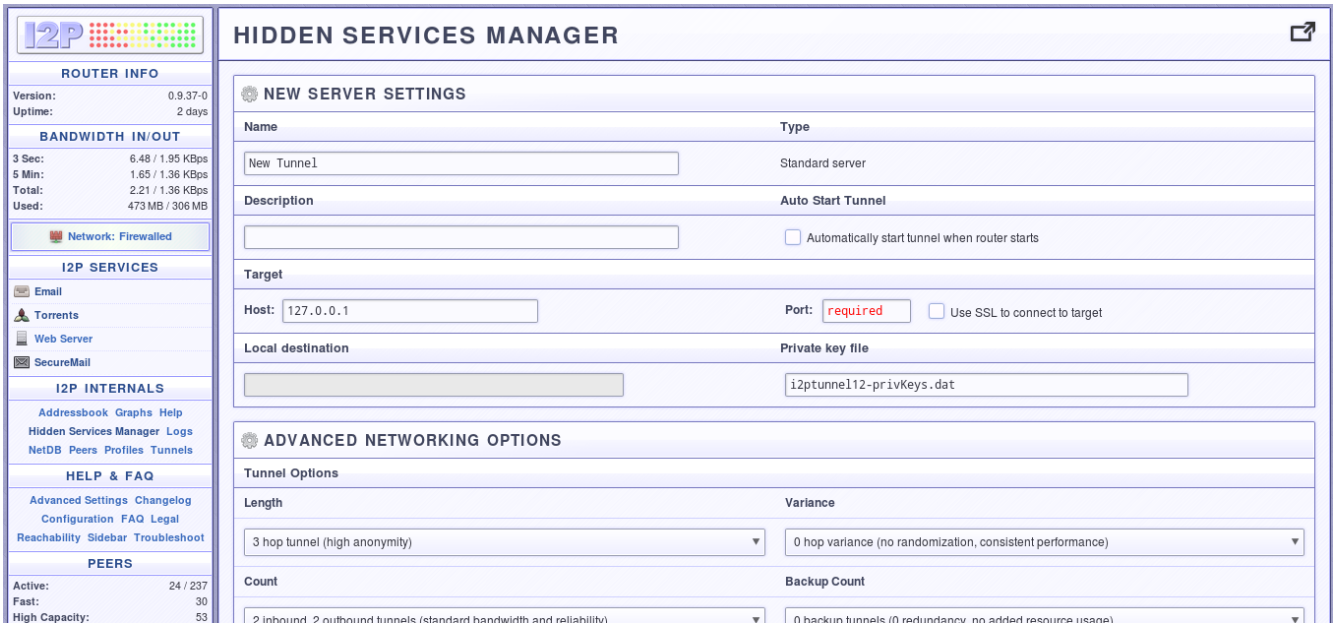
(The web page is served up by your local machine, not an external web site). You should see a page similar to the one shown below:



Create a new hidden service of type “Standard” in the list-box then click the *Create* button.



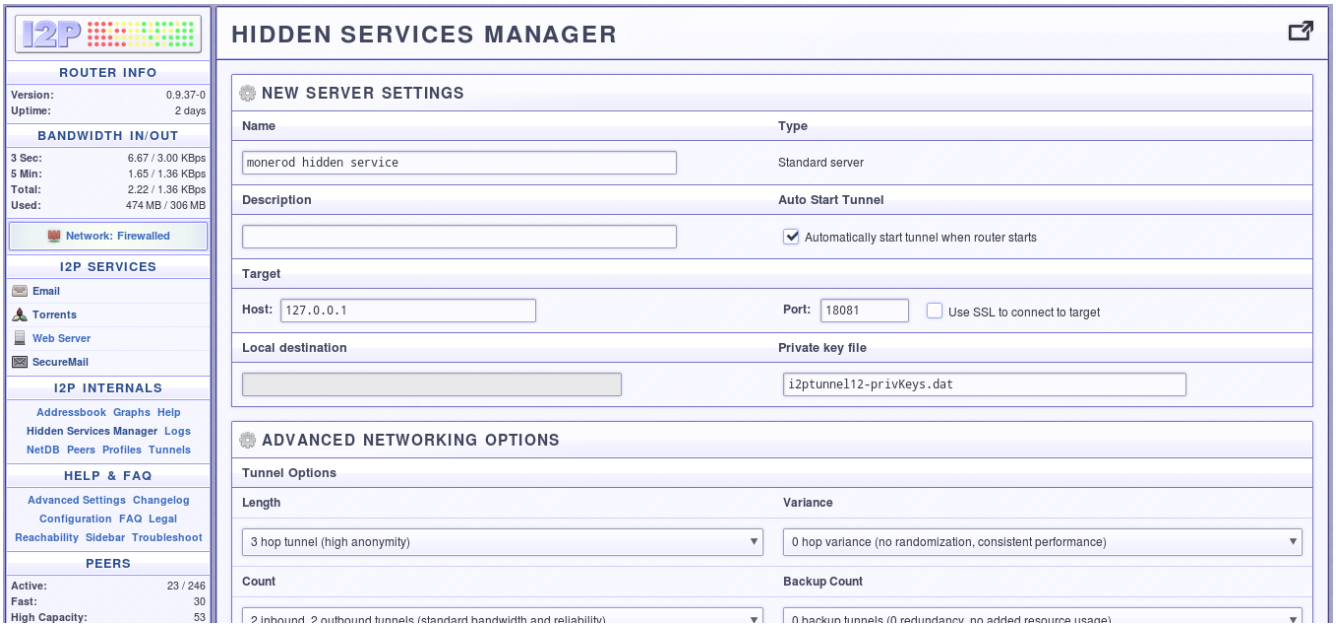
You will then be presented with a screen similar to the one shown below:



Enter in the form:

Name = monerod hidden service
Check the *Automatically start tunnel when router starts* checkbox
Port = 18081

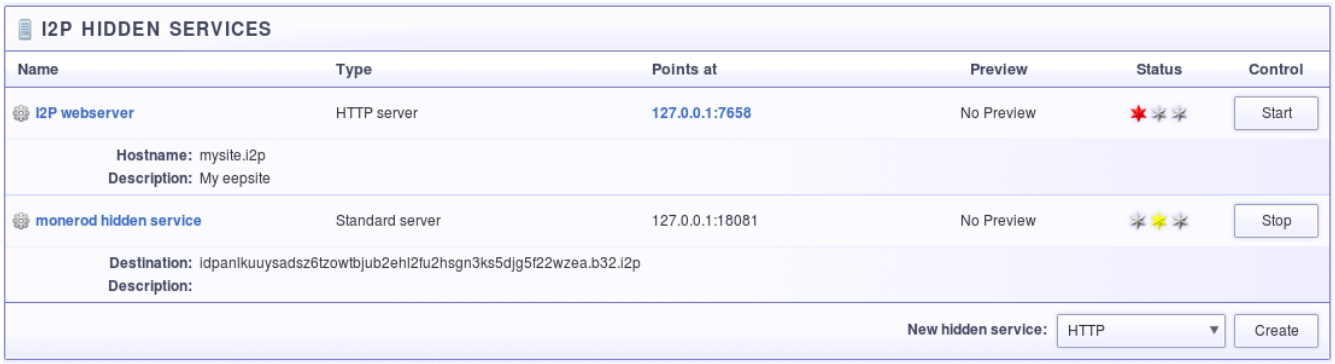
The your screen should look like the one shown below:



Optional:
If you want to require users to possess an encryption key to use the I2P monerod hidden service, you can encrypt the *leaseset* and generate the key. This will only allow clients with the encryption key to connect to the I2P monerod hidden service. To encrypt the leaseset and generate the encryption key, scroll down the page to the section labeled **Encrypt Leaseset**. Check the *Only allow clients with the encryption key to connect to this server* checkbox and then click the *Generate* button. The screen should resemble the one presented below.

Users of the I2P monerod hidden service will then need to add the service *destination* and the *encryption key* in their I2P Keyring on their client tunnel machine (http://127.0.0.1:7657/configkeyring).

When finished, click the *Save* button at the bottom of the form. You should now see a new *monerod hidden service* similar to the one shown below.



Your tunnel *destination* will be different and you will need to reference this tunnel destination later when you setup your client tunnels. When the light in the hidden service *Status* field is green, your monerod hidden service will be operational and route requests to the I2P hidden service to the Monero daemon running on the machine.

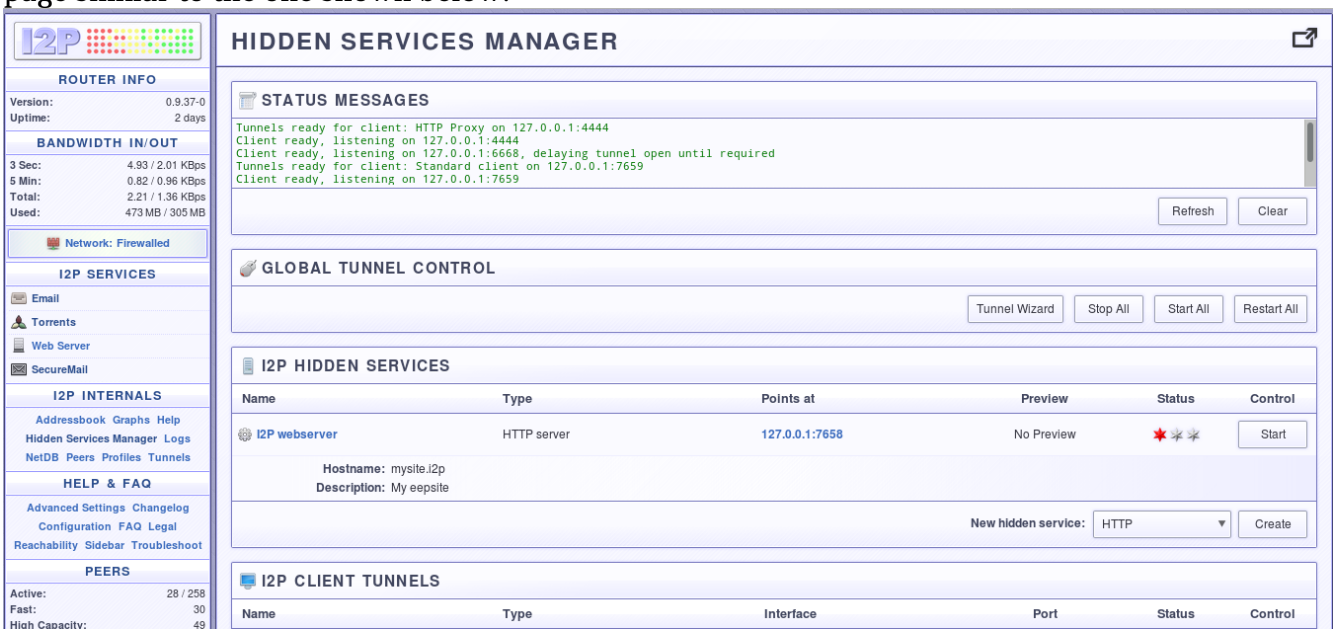
Configuring I2P Client Tunnels

On a different (client) machine, repeat the following steps presented above:

Install the Monero software package i.e., wallet (if not already installed)
Install java runtime environment (if not already installed)
Install I2P router (if not already installed)

On the client machine, go to the I2P Hidden Services Manager at:
http://127.0.0.1:7657/i2ptunnelmgr

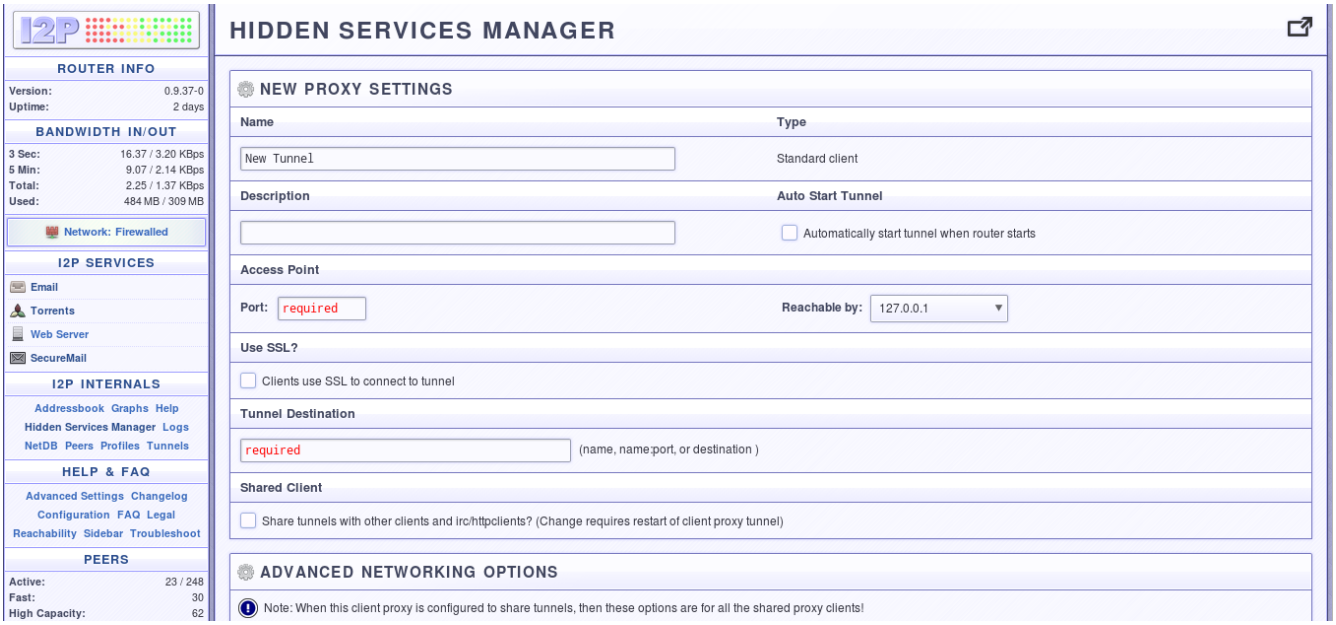
(The web page is served up by your local machine, not an external web site). You should see a page similar to the one shown below:



Scroll down on this page until you see the area for creating a new *I2P Client tunnel*.

Select a *Standard* tunnel type, then click the *Create* button.

You will then see a form similar to the one shown below:



Give the tunnel a name.

Enter **18081** for the *Port* number (enter over the red “required” text).

Enter the *hidden service tunnel destination* (from the creation of the hidden service above) in the *Tunnel Destination* field (that has the red “required” text in it).

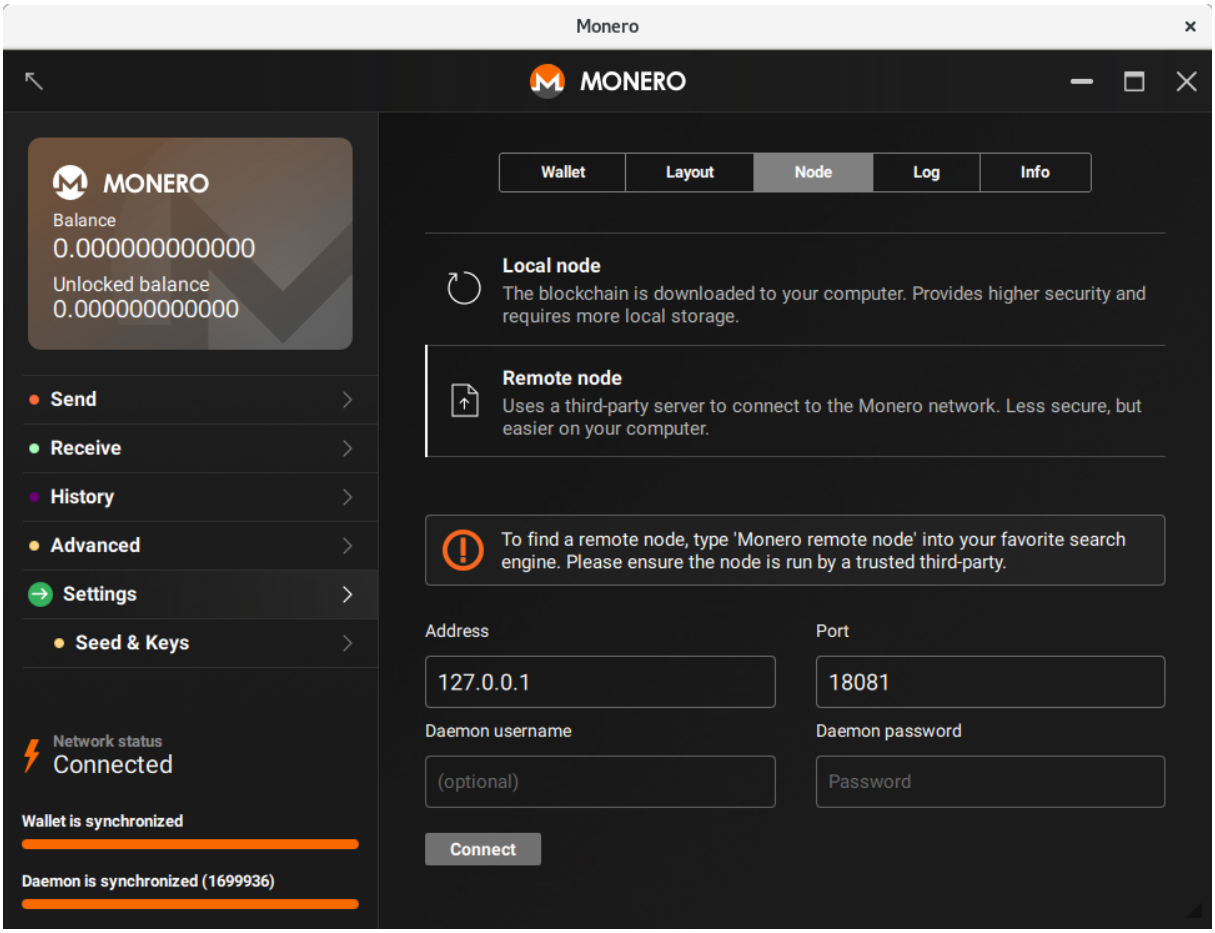
Then scroll to the bottom of the page and click the *Save* button.

You will need to start this tunnel prior to attempting to connect to the I2P monerod hidden service. When your done using the hidden service, you can stop the client tunnel.

Connecting Wallets to the I2P monerod Hidden Service

Now with the hidden service running on the hidden service machine and the client tunnel running on the client machine, you can connect your Monero wallet to the *I2P monerod hidden service.*

You can run the monero-wallet-cli on the client machine and it will access the Monero blockchain via the I2P monerod hidden service. You could also install the Monero GUI wallet client and it will access the I2P monerod hidden service as shown below.



Cycle times will be slower than running everything over the regular Internet because the I2P network is end-to-end encrypted and goes through multiple hops to reach destinations. However, remember that as a result, your wallet machine's IP address is no longer exposed to prying eyes and your network access to the Monero blockchain is anonymous.

Automating the I2P monerod hidden Service

To effectively operate an I2P monerod hidden service it is necessary to provide a level of service which attempts to keep both the monerod and i2prouter on the hidden service, machine up and running 24x7.

To do this on a Linux based operating system, *systemd* is utilized to automatically start monerd and the i2prouter when the computer boots up, and to restart monerod and i2prouter if either one terminates for any reason during operation.

To configure this level of service, two service files were written:

/etc/systemd/system/monerod.service:

```
[Unit]
Description=monerod.service
After network.target=monerod.service

[Service]
Type=forking
ExecStart=/home/user/monero-v0.13.0.4/monerod –detach –pidfile /tmp/monerod.pid
ExecStop=/home/user/monero-v0.13.0.4/monerod exit
PIDFile=/tmp/monerod.pid
User=user
Group=user
Restart=always
RestartSec=90

[Install]
WantedBy=multi-user.target
```

/etc/systemd/system/i2p.service:

```
[Unit]
Description=i2p.service
After network.target=i2p.service

[Service]
Type=forking
ExecStart=/home/user/i2p/i2prouter start
ExecStop=/home/user/i2p/i2prouter stop
ExecReload=/home/user/i2p/i2prouter restart
User=user
Group=user
Reload=always
Restart=always
RestartSec=90

[Install]
WantedBy=multi-user.target
```

*You can tweak the parameters in the two service files shown above as you see fit. Obviously the **User** and **Group** parameters will need to be modified for your machine. The path specifiers to **monerod** and the **i2prouter** will need to reflect the locations on your computer.*

Once the service files are placed into the */etc/systemd/system* directory, you will need to issue the following comands:

```
sudo systemctl enable monerod.service     # enables the monerod services in systemd
sudo systemctl enable i2p.service         # enables the i2p.service in systemd
sudo systemctl daemon-reload              # reloads the systemd daemon
```

When the computer reboots, the monerod and i2p services should be automatically started.

You can issue the following commands to check the status of the services:

```
sudo systemctl status monerod.service
sudo systemctl status i2p.service
```

Read up on systemd to increase your knowledge.

When the services are up and running, you can open a terminal window and issue a *monerod exit* command to stop monerod. Then wait for systemd to restart it. Verify that the service is restarted and you should be good.

Similarly you can issue an *i2prouter stop* command. Then verify that the i2prouter has terminated. Then wait to verify that *systemd* has restarted the i2prouter. **As long as you configured the I2P monerod hidden service to - *Automatically start tunnel when router starts*** – your I2P monerod hidden service should be well positioned to attempt a 7x24 level of service.

VPN Option

I have installed an openvpn server on a VPS machine running Ubuntu 16.04.2 LTS. The version of openvpn installed is:

OpenVPN 2.3.10 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL] [PKCS11] [MH] [IPv6] built on
Jun 22 2017

An .ovpn file generated from the openvpn server mentioned above was placed on my I2P monerod hidden service machine along with the openvpn client software to provide an operational VPN connection for the Monero daemon enabling the I2P monerod hidden service.

The hidden service machine was configured to prevent DNS leaks (checked with: https://www.dnsleaktest.com/).

The Monero daemon on the hidden service machine is started as:

```
cd monero-v0.13.0.4
./monerod –p2p-bind-ip 10.8.0.3
```

In my present setup, the Monero daemon doesn't support incoming connections on port 18080. I'm in the process of determining how to correct that.
This lack of incoming connections still allows for the use of Monero, but it doesn't allow others to connect to the node to help synchronize their nodes.

Placing the I2P monerod hidden service machine behind a VPN provides an additional layer of plausible deniability in that the Monero daemon's access to the global Monero network now exposes the IP address of the VPN server rather than the IP address of the hidden services machine.

The risk exposure involves running a Monero full node that allows external client connectivity. All external client wallets that interact with the I2P monerod hidden service still do so anonymously through the I2P network.

If you wish to take this option, you will need to gain access to a VPN service that you can connect your hidden service machine to as a VPN client.

Conclusion

If you have any questions or would like assistance to get an I2P monerod hidden service established you can feel free to contact me via I2P-Bote.

Use I2P-Bote email destination:

iVCMAA149vlVByA0DfoTnboySJvfQl3d0nv~fmhkPH~qmlOl6VrNqMKYd3gpw4kLMDhWmc~aTs07z9UlSd1fkO

Appendix

Monero and Anonymous Network Transport Options

Reflecting back on the **Kovri** project, I believe that explicitly integrating I2P connectivity into the Monero protocol would have been suboptimal. I think a better strategic approach would be to provide multiple options for network transports within the Monero protocol.

If Monero would select any one specific anonymous network to provide network anonymity and that anonymous network were to become obsolete, Monero would have to make significant changes to maintain competitive advantage. However, if Monero were designed to be flexible enough to accept one or more network transports the protocol would be more agile. Taking a *Plug-and-Play* approach, and allowing for a multitude of network transport adapters, to attain anonymity at the network layer, would make the Monero protocol anti-fragile.

Accomodating multiple options for network transport would add to Monero's decentralization. If Monero provided for network transport over the Internet, I2P, and Tor, with transactions sent over those different network transports, Monero's decentralization would be increased significantly.

So picking a single anonymous network to add anonymity at the network layer, in my opinion, would be a mistake. Allowing the user to be in control, to select from many network transport options, would be a smarter approach.