RPi / Tor setup manual

RPi OS image

Download Raspbian Buster Lite [https://www.raspberrypi.org/downloads/raspbian/] on your machine

---

PRi SD card setup [on Linux]

Hash Key
Verify if the the hash key of the zip file matches the one on the **downloads** page [Buster Lite: SHA-256: 12ae6e17bf95b6ba83beca61e7394e7411b45eba7e6a520f434b0748ea7370e8]:

>> **sha256sum** <path to an image zip file>

Unzip
Unzip the zip file

>> **unzip** 2020-02-13-raspbian-buster-lite.zip

Mounted Devices
Check mounted devices

>> **df -h**

Your SD card [partition(s)] will show up on the list: dev/mmcblk0<#>

Unmount

>> **umount** /dev/mmcblk0p1

Image-to-SD
To write the image to the SD card, run the following command, but **make sure *of*= argument output is a correct device name, meaning the whole SD card and not one of its partitons!**

>> **sudo dd bs=4M status=progress if=**<path to .img file> **of=**/dev/mmcblk0
>> **sudo sync**

ssh File
Create a file named ssh and save it in a boot partition

>> **cd** <path to a boot partition>
>> **touch ssh**

SD card is good to go.
Insert the card before powering on the Raspberry Pi, and shutdown the Raspberry Pi before unplugging the card.

Configuring RPi

Default user: **pi**
Default pwd: **raspberry**
Default hostname: **raspberrypi**

SSH into RPi
To ssh into RPi, first, try

>> **ssh pi@raspberrypi**

If it doesn't work and

a. You have a screen and keyboard:
Insert SD and power the RPi. The **IP address** will be displayed in the console at the end of the boot process. Login with the default credentials and enable sshd

>> **sudo raspi-config**

Go to **Interfacing Options** and enable SSH.
If you did not take note of the IP yet, you can always do

>> **ifconfig eth0 | grep inet**

ssh in RPi

>> **ssh pi**@192.168.1.XXX

b. You don't have a screen:
Connect your machine to a router with an ethernet cable and run

>> **nmap -sn 192.168.1.1-255**

Adjust network mask, the RPi's default name is **raspberrypi**

Or if **raspberrypi** doesn't show up on the list:

>> **hostname -I**
>> **nmap -sn** <ip address of your machine>**/22**

Now connect the rpi to the router and map the network again

>> **nmap -sn** <ip address of your machine>**/22**

The additional ip address that showed up is of RPi

>> **ssh pi@**<RPi IP address>

Change a hostname

>> **sudo raspi-config**

Go to **Network Options**, select **Hostname** and rename it

If, for instance, Hostname is set to **kadut**, you may now ssh into the RPi this way

>> **ssh pi@**kadut

Add a User

Switch to root

>> **sudo -i**

Create a user

>> **adduser** xpub

To ssh to rpi using that username

>> **ssh** xpub@kadut

Add a user to a sudo group

>> **adduser** xpub **sudo**

To check if a user is sudo

>> **id** xpub

If **xpub** is a sudo user, the command should output **27(sudo)** at the end of the line, after **uid**, **gid** and**groups**

To switch to a different user

>> **sudo su -** <username>

Remove default pi user

>> **sudo -i**
>> **deluser pi**

List of users

>> **cut -d: -f1 /etc/passwd**

Delete user

>> **userdel** <username>

Setting locale

>> **sudo -i**
>> **echo "LC_ALL=en_US.UTF-8" >> /etc/environment**
>> **echo "en_US.UTF-8 UTF-8" >> /etc/locale.gen**
>> **echo "LANG=en_US.UTF-8" > /etc/locale.conf**
>> **locale-gen en_US.UTF-8**

SSH
Generate public/private keys on your machine:

>> **ssh-keygen -t ed25519 -b 320**

Your public [**id_ed25519.pub**] and private [**id_ed25519**] keys are stored in **.ssh** directory in the **Home** folder of your machine.

Get the public key onto RPi
To copy the public key from your machine into **authorized_keys** file on RPi

>> **cat ~/.ssh/id_ed25519.pub | ssh** xpub@kadut **"mkdir -p ~/.ssh && chmod 700 ~/.ssh && cat >> ~/.ssh/ authorized_keys"**

Or display the contents of **id_ed25519.pub** file

>> **cat** <path to id_ed25519.pub>

Copy the key and go to **.ssh** folder on RPi

>> **cd** <path to .ssh directory on the RPi>

and paste it into **authorized_keys** file

>> **sudo nano authorized_keys**
>> **Ctrl + x**
>> **y**

Disable pwd and root login

>> **sudo nano /etc/ssh/sshd_config**

Uncomment **PasswordAuthentication** and set it to **no**
Uncomment **PermitRootLogin prohibit-password**

Save and exit

>> **Ctrl + x**
>> **y**

Reload SSH

>> **sudo /etc/init.d/ssh restart**

Reboot RPi

>> **sudo reboot**

Login with a Host name
In order to ssh into RPi using only a **Host** name instead of **xpub@kadut**, modify a **config** file on your machine in **.ssh** folder:

Host watermelon
User xpub
Hostname kadut
Port 22
Identityfile <path to **id_ed25519** file on your machine>
Serveraliveinterval 30

Host can be different from **Hostname**: **watermelon**
Hostname can be either set to RPi IP address or a name you've set via sudo raspi-config: **kadut**

And ssh

>> **ssh** watermelon

Static Website as Tor Hidden Service
Run a static website and serve it as an onion site

Local HTTP server
As root:

Install nginx on the RPi

>> **apt install nginx**

In the browser from another computer on the network, check that you the default HTML page is properly served at:
_____http://192.168.1.XXX [you should see a small "Welcome to nginx!" text].

Create a non-default mini static website

>> **mkdir /var/www/**partyvan
>> **echo** "OHAI" **> /var/www/**partyvan**/index.html**

Disable nginx default site

>> **rm /etc/nginx/sites-enabled/default**

Create new nginx site

>> **nano /etc/nginx/sites-available/**partyvan

```
server {
   listen 80;

   root /var/www/partyvan;
   index index.html;

   server_name partyvan;   # Replace with onion address once you have one
}
```

Enable site

>> **ln -s /etc/nginx/sites-available/**partyvan **/etc/nginx/sites-enabled/**
>> **service nginx reload**

In the browser from another computer on the network, check that you the default HTML page is properly served:
http://192.168.1.XXX [you should see a small "OHAI" text].

Tor setup
Note: This is only valid for RPi2 and later.

>> **sudo nano /etc/apt/sources.list**

Add the Tor deb repos to /etc/apt/sources.list. At time of writing, stable Raspbian is based on Buster:

deb https://deb.torproject.org/torproject.org buster main
deb-src https://deb.torproject.org/torproject.org buster main

Add the GPG keys used to sign the packages from the Tor repos:

>> **curl https://deb.torproject.org/torproject.org/A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89.asc | gpg -- import**
>> **gpg --export A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89 | apt-key add -**

Install Tor

>> **apt update**
>> **apt install tor deb.torproject.org-keyring**

Edit **/etc/tor/torrc** and in the section about hidden services, add:

 # Partyvan site
HiddenServiceDir /var/lib/tor/partyvan/
HiddenServicePort 80 127.0.0.1:80

Restart Tor, this will generate the keys for the partyvan hidden service

>> **service tor restart**

If everything went well, there should be a **/var/lib/tor/partyvan/** folder with notably both public and private keys for the service (backup!) and the hostname information to reach the hidden service from onionland. To know the onion address of partyvan, simply do:

>> **cat /var/lib/tor/**partyvan**/hostname**

You will get something like c7phl5mrjy34...onion, if you paste this address in your Tor browser, torified browser or whatever you use, you should see the partyvan site!

Further tweakingCertificates
Certs are not needed for a hidden service like this one. You already get encrypted traffic via Tor itself. With that said, certs could be used as a means to authenticate the ownership over the hidden service, to prevent phishing. Legit certs who can be used in this context are very $$$ and avail from DigiCert.

Disable NGINX version signature
Don't let NGINX emit its version on error pages and in the “Server” response header field, uncomment the following in **/etc/nginx/nginx.conf**

server_tokens off;

Disable directory listing
Don't trust defaults, add this to your **/etc/nginx/sites-available/partyvan** in the **server** block:

```
location / {
    autoindex off;
}
```

Onion only serving
Don't serve HTTP on the clearnet, force NGINX to serve only on localhost. In **/etc/nginx/sites-available/**partyvan, replace **listen 80;** with listen **127.0.0.1:80;**.

Onionscan
There's a tool [untested at time of writing] that tests an onion address against known hidden service gotchas [https://github.com/gugronnier/onionscan/blob/master/doc/what-is-scanned-for.md]. It does not seem to be actively maintained, but it's possible to find more active forks like this one [https://github.com/gugronnier/onionscan].