Raspberry-Pi as a home server

Romain Pellerin
http://romainpellerin.eu

May 9, 2016

Contents

1 Setup

The Raspberry-Pi used in this tutorial is the B model, purchasable on the official website. Make sure you own all the required equipment to setup a Raspberry-Pi before starting.

All the commands that require root permissions will begin with `sudo`. All the other commands can be run as a regular user.

1.1 Minimal installation

1. Download a distribution: `http://www.raspberrypi.org/downloads/`. Raspbian is the best. I will assume you've downloaded Raspbian in this tutorial. Once the download is complete, it may be a good idea the verify the hash key (checksum).

2. Extract the zip downloaded and then install your distribution on your SD card. Unplug your SD card from your computer if already in. On your computer, in a terminal:

   Shell

   ```
   dh -h
   ```

   Plug in the SD card now. Type the command again and try to identify your SD card. The name of the SD card can be either */dev/mmcblk0p**X*** or */dev/sd**YX*** where **X** can be *0*, *1*, etc. and **Y** can be *a*, *b*, etc. The last part of the name which can be *pX* or *X* is the partition id.

   Shell

   ```
   umount /dev/mmcblk0p1 # Unmount all the partitions on the SD (if many)
   sudo dd bs=4M if=2014-09-09-wheezy-raspbian.img of=/dev/mmcblk0
   sync
   ```

   - `2014-09-09-wheezy-raspbian.img` is the image extracted from the downloaded zip
   - `/dev/mmcblk0` is the SD card (without the partition id at the end of the name)
   - You can use `bs=1M` in case of problem

   Wait until the `dd` returns (there's no feedback during the whole procress, it can takes up to 10 minutes).

   More information: `http://www.raspberrypi.org/documentation/installation/installing-images/linux.md`

3. Plug an Ethernet wire to your Pi, and then plug the Pi to a TV. Insert the SD card in your Pi, power it on, and wait a few seconds. When the 'Raspberry Pi Software Configuration tool' (raspi-config) appears on screen, simply exit and reboot. Now you can go headless thanks to SSH.

   Shell

   ```
   sudo reboot
   ```

1.2 Update your Raspberry-Pi

From a remote computer, connect to your Raspberry-Pi via SSH. Once connected, update it:

Shell

```
sudo aptitude update ; sudo aptitude upgrade # Upgrade packets
sudo aptitude install rpi-update # If this packet is missing
sudo reboot
sudo rpi-update ; sudo reboot # Upgrade the firmware and reboot
sudo aptitude update ; sudo aptitude upgrade # Upgrade packets
```

Gain some free space of storage.

1.3 Settings

After updating the Pi, you should set it up.

Shell

```
sudo raspi-config
```

1. Expand Filesystem: you should do this only if you don't plan to install Raspbian on a hard drive disk (see section 1.5)
2. Change User Password (recommanded!)
3. Enable Boot to Desktop/Scratch: Desktop Log in as user 'pi' at the graphical desktop if needed.
4. Internationalisation Options: you really should change the locale to avoid issues in the future (any language you want as default and the current language you daily use on your main computer, the one you'll use for SSH). Use UTF-8. You can also change the timezone and the keyboard layout to match your language (can only be done when a keyboard is connected)
   For the keyboard layout: Other > French > French > Default > The default for the keyboard layout > No compose key > Yes
5. Advanced Options > Overscan: configure this only if black bars are present on display. In that case, disable Overscan.
6. Advanced Options > Hostname: whatever you want
7. Advanced Options > Memory Split: increase allocated memory for GPU in order to play HD videos. (128MB should be sufficient for 1080p videos).

Then reboot with

Shell

```
sudo reboot
```

1.4 Advanced settings

1.4.1 Keyboard

As previously explained, you can change your keyboard layout with the graphical tool `rasp-config`. You can also do this by remplacing 'en' with 'fr' in /etc/default/keyboard. Then reboot.

1.4.2 Screen blank time

To avoid screen blanking after a while in the console (tty), change `BLANK_TIME` to 0 in /etc/kbd/config. To avoid screen blanking after a while in X, in /etc/lightdm/lightdm.conf, below [SeatDefault], add

File: /etc/lightdm/lightdm.conf

```
xserver-command=X -s 0 dpms
```

1.4.3 Auto login

As previously seen, the tool `rasp-config` allows you to configure auto login. But you can also do this with the command line. In /etc/inittab, replace

File: /etc/inittab

```
1:2345:respawn:/sbin/getty 115200 tty1
```

with

File: /etc/inittab

```
1:2345:respawn:/bin/login -f pi tty1 </dev/tty1 >/dev/tty1 2>&1
```

1.4.4 Auto start X server

As previously seen, the tool `rasp-config` allows you to configure auto start X. But you can also do this with the command line. At the end of /etc/rc.local, add

File: /etc/rc.local

```
su -l pi -c startx
```

1.4.5 Stopping Raspberry-Pi from auto changing source on TV

Shell

```
sudo echo "hdmi_ignore_cec_init=1" >> /boot/config.txt
```

1.4.6 Auto-mount an external hard disk drive

Shell

```
mkdir /home/pi/my_wonderful_hdd # Where the HDD will be mounted
sudo fdisk -l # Note the location of the HDD (something like /dev/sda1)
sudo mount -t auto /dev/sda1 /home/pi/my_wonderful_hdd # Mount it now
sudo echo "/dev/sda1 /home/pi/my_wonderful_hdd auto noatime 0 0" >> /etc/fstab # Auto mount at boot
```

1.4.7 Change hostname

Edit both /etc/hostname and /etc/hosts. That's all!

1.4.8 Reconfigure locales

Can also be achieved in the tool 'raspi-config', under 'Internationalisation Options' section.

Shell

```
sudo dpkg-reconfigure locales
```

1.4.9 Disable SWAP permanently

Swapping is bad for your SD card lifespan. You should disable it permanently.

Shell

```
sudo swapoff --all # Temporary
sudo update-rc.d dphys-swapfile remove
sudo apt-get remove dphys-swapfile # Permanently
```

1.5 Running from an external hard drive

> *The SD card of the Raspberry Pi is required for booting. While you may not be able to boot from an external hard drive, moving the / partition there will significantly speed up the Raspberry Pi. There are many uses where this configuration particularly useful. File servers and media servers are good examples.* – Source

I recommand to use a hard drive with its own power wire. **Note that you'll lose any data that may exists on your HDD**. Plug in the USB wire in the Raspberry-Pi, then:

Shell

```
sudo fdisk -l # Identify the hdd, let's say /dev/sda
sudo fdisk /dev/sda
d # Delete all the existing partitions if many
n # Create a new one
p # Primary
1
<Enter> # Hit enter key
+100G # 100GB, or something else
n # Create a second partition, swap
p
2
<Enter>
+2G
t # Change the type...
2 # Of the second partion
82 # Make it a swap partition
n # Create the data partition
p
3
<Enter>
<Enter>
p # Make sure everything is OK

w # Write the new partition table

sudo mkfs.ext4 /dev/sda3 # Format the data partition
sudo mkswap /dev/sda2 # Initialize the swap partition

sudo dd if=/dev/mmcblk0p2 of=/dev/sda1 bs=32M conv=noerror,sync # Make sure /dev/mmcblk0p2 is correct for you, see the output of fdisk earlier
# Using bs=32M is quite dangerous, bs=4096 is better but much slower

sudo e2fsck -f /dev/sda1 # Check the file system for errors. Press 'y' if any errors are found
sudo resize2fs /dev/sda1 # Resize the file system to fill the partition (because the
    original partition on the SD card smaller than the one on your HDD)

sudo cp /boot/cmdline.txt /boot/cmdline.orig
sudo nano /boot/cmdline.txt # Change /dev/mmcblk0p2 to be /dev/sda1, and add "
    bootdelay rootdelay" at the end of the line
sudo nano /boot/config.txt # Add "boot_delay=1"

sudo mount /dev/sda1 /mnt # Mount your HDD
sudo mkdir /mnt/data # Where sda3 will be mounted
sudo nano /mnt/etc/fstab
```

File: /mnt/etc/fstab

```
proc           /proc          proc    defaults         0       0
/dev/mmcblk0p1 /boot          vfat    defaults         0       2
/dev/sda1      /              ext4    defaults,noatime 0       1
/dev/sda2      none           swap    sw               0       0
/dev/sda3      /mnt/data      ext4    defaults         0       0
```

Save and exit. Finally, reboot and stop the system from using the swap file that it normally uses:

Shell

```
sudo sync
sudo reboot
sudo apt-get remove dphys-swapfile
```

1.6 Regular backups of your server

It's **highly recommended** to create regular backup of your whole server, or at least of your important files and directories.

To do so, you can set up a cron job daily with this script: https://github.com/rpellerin/dotfiles/blob/master/scripts/backupRaspberryPi.sh. The cron job might look like something:

crontab -e

```
0 0 * * * /home/pi/git/dotfiles/scripts/backupRaspberryPi.sh /home/pi/backup_list.txt
```

And the list might look like:

File: /home/pi/backup_list.txt

```
/etc/apache2
/var/www

/etc/transmission-daemon/settings.json

/etc/squid/

/etc/openvpn/server.conf
```

2 Security

2.1 Firewall (iptables)

Shell

```
wget --no-check-certificate https://raw.githubusercontent.com/rpellerin/dotfiles/master/scripts/firewall.sh
chmod 700 firewall.sh
sudo chown root:root firewall.sh
sudo mv firewall.sh /etc/init.d
sudo update-rc.d firewall.sh defaults
```

An alternative to make the configuration of iptables persistent can be found here: http://www.microhowto.info/howto/make_the_configuration_of_iptables_persistent_on_debian.html.

2.2 SSH

Shell

```
sudo aptitude install ssh # Both server and client, if not installed
```

You should allow login with public key. From a remote computer do this:

Shell

```
ssh-keygen -t rsa -o -a 50 # Accept the default directory, passphrase is useless if on a personal computer
ssh-copy-id -i ~/.ssh/id_rsa.pub pi@raspberrypi # raspberrypi is the IP of the Raspberry-Pi
```

Then you should strongly disable ssh password authentification (for security reasons) and some other feature. Here is a strong configuration file (/etc/ssh/sshd_config):

File: /etc/ssh/sshd_config

```
#HostKey /etc/ssh/ssh_host_dsa_key
UsePrivilegeSeparation yes
StrictModes yes
IgnoreRhosts yes
PermitEmptyPasswords no
PasswordAuthentication no
PubkeyAuthentication yes
PermitRootLogin no
ChallengeResponseAuthentication no
Banner /etc/issue.net # Edit it:) It will be displayed at login
UsePAM no
AllowUsers pi

# http://kacper.blog.redpill-linpro.com/archives/702
Ciphers aes256-ctr,aes192-ctr,aes128-ctr
MACs hmac-sha2-512,hmac-sha2-256,hmac-ripemd160
KexAlgorithms diffie-hellman-group-exchange-sha256
```

You should read this as well (it'll take 1 minute!). And this one if you have time.
It's highly recommended to change the default port!. If so, edit /etc/init.d/firewall.sh.
Finally:

Shell

```
sudo service ssh reload
```

IMPORTANT: in case you want to prevent SSH bruteforce attacks, read this `http://mysecureshell.sourceforge.net/fr/securessh.html#question3`. But the part 2.3 should be sufficient.

Tip: if you want to permanently disable SSH, in /etc/init/ssh.conf comment the line `#start on runlevel [2345]`.

2.3 fail2ban

Shell

```
sudo aptitude install fail2ban
sudo fail2ban-client -x start
```

The configuration file is located at: /etc/fail2ban/jail.conf. Make a copy under /etc/fail2ban/jail.local and edit it. This is an example of its content:

File: /etc/fail2ban/jail.local

```
[DEFAULT]
ignoreip = 127.0.0.1
findtime = 3600
bantime = 86400
maxretry = 3
action = %(action_mwl)s

[ssh] # You should also enable ssh-ddos
enabled = true
# Add any custom port on the next line
port    = ssh,sftp
filter  = sshd
logpath = /var/log/auth.log
maxretry = 3

[ssh-ddos]
enabled = true
port     = ssh,sftp
filter   = sshd-ddos
logpath = /var/log/auth.log
maxretry = 3

# Enable all Apache jails
# Check this out: https://github.com/fail2ban/fail2ban/issues/286

# Protect against DOS attack
# 360 requests in 2 min > Ban for 10 minutes
[http-dos]
enabled = true
port = http,https
```

```
filter = http-dos
logpath = /var/log/apache*/*access.log
maxretry = 360
findtime = 120
bantime = 600

[squid]
enabled = true
port    = all
filter = squid
logpath = /var/log/squid/access.log
maxretry = 3
```

File: /etc/fail2ban/filter.d/http-dos.conf

```
[Definition]

# Option: failregex
# Note: This regex will match any GET or POST entry in your logs, so basically
# all valid and not valid entries are a match.
# You should set up in the jail.conf file, the maxretry and findtime carefully
# in order to avoid false positives.

failregex = ^<HOST> -.*\"(GET|POST).*

# Option: ignoreregex
# Notes.: regex to ignore. If this regex matches, the line is ignored.
# Values: TEXT
#
ignoreregex =
```

You can now test that the http-dos jail is working with this tool from the package *apache2-utils*:

Shell

```
ab -n 500 -c 10 http://your-web-site-dot-com/
```

Optionally, you can protect your Apache server against w00tw00t attacks: http://doc.ubuntu-fr.org/fail2ban#exemple_avec_la_regle_anti-w00tw00t.

File: /etc/fail2ban/filter.d/squid.conf

```
# Fail2Ban filter for Squid attempted proxy bypasses
#
#

[Definition]

failregex = ^\s+\d\s<HOST>\s+[A-Z_]+_DENIED/403 .*$
           ^\s+\d\s<HOST>\s+NONE/405 .*$
ignoreregex =
```

Then restart the service:

Shell

```
sudo service fail2ban restart
```

More information: https://help.ubuntu.com/community/Fail2ban.

2.4 Sudoers

You should probably remove the ability to run "root" commands without typing pi's password.

Shell

```
sudo visudo # it will safely edit /etc/sudoers
```

In /etc/sudoers, : remove the "NOPASSWD: ".

2.5 Notifications via emails

2.5.1 Logs

Visit `http://doc.ubuntu-fr.org/logwatch`, or more simply:

Shell

```
sudo apt-get install logwatch
crontab -e

/usr/sbin/logwatch --output mail --mailto me@domain.com --detail high # To be written
    in cron
```

2.5.2 Available updates

Shell

```
sudo apt-get install cron-apt
```

Then add your address in /etc/cron-apt/config.

3 DynHOST

A DynHOST is useful to associate a domain name with a dynamic IP. For example, if you need to access your Raspberry-Pi outside you LAN and if your IP often change, you'll need a DynHOST. This tutorial has been tested with OVH. It comes from `http://dev.kprod.net/?q=dns-dynamique-avec-dynhost-ovh`.

1. First, download this tarball.

   Shell

   ```
   wget "http://www.bozorokus.net/DynHost.tgz"
   ```

2. Extract its content into your home folder (3 files).

   Shell

   ```
   tar xzf DynHost.tgz
   ```

3. Edit dynhost like this:

   File: dynhost.sh

   ```
   #! /bin/sh

   # dynhost
   # OVH - DynHost
   # Original file: http://guides.ovh.com/DynDns
   # Modified file: http://dev.kprod.net/?q=dns-dynamique-avec-dynhost-ovh

   #sous domaine et domaine choisi pour le DynHost
   HOST=subdomain.domain.com
   #login (domaine + suffixe) defini dans les identifiants DynHost
   LOGIN=loginDynHost
   #password (defini dans les identifiant DynHost egalement)
   PASSWORD=passwdDynHost

   #chemin complet vers le repertoire contenant le script
   PATH_APP=/home/pi/DynHost
   OPTIONS=""

   getip() {
       IP=‘lynx -dump http://monip.org | grep ’IP’ | sed ’s/.*: //;’‘
       OLDIP=‘cat $PATH_APP/old.ip‘
   }

   echo ‘date‘ >> $PATH_APP/dynhost.log
   getip

   if [ "$IP" ]; then
       if [ "$OLDIP" != "$IP" ]; then
           echo -n "old IP: " >> $PATH_APP/dynhost.log
           echo $OLDIP >> $PATH_APP/dynhost.log
   ```

```
        echo -n "new IP: " >> $PATH_APP/dynhost.log
        echo $IP >> $PATH_APP/dynhost.log
        echo "update !" >> $PATH_APP/dynhost.log
        if [ "$OPTIONS" = "" ]; then
            OPTIONS="-a $IP"
        fi

        python $PATH_APP/ipcheck.py $OPTIONS $LOGIN $PASSWORD $HOST >> $PATH_APP/dynhost.log
        echo -n "$IP" > $PATH_APP/old.ip
     else echo "no update" >> $PATH_APP/dynhost.log
     fi
else
    echo "error getting ip" >> $PATH_APP/dynhost.log
fi
```

and make it runnable

Shell

```
chmod +x dynhost.sh
```

4. Install lynx

Shell

```
sudo aptitude install lynx
```

5. Eventually, create a cron job (every minute):

Shell

```
  $ crontab -e
  */1 * * * * /home/pi/DynHost/dynhost
```

4 Apache

4.1 Basic installation

Shell

```
sudo apt-get update ; sudo apt-get install apache2 php5 php5-mysql php5-curl curl php5-gd php5-json
# Here, using apt-get instead of aptitude because of dependencies conflicts
# php5-gd enables images manipulation
# php5-json provides a module for JSON functions in PHP scripts.
# php5-intl provides a module to ease internationalisation of PHP scripts.
```

The default user and group that launch Apache2 are www-data. If you wanna change it, edit /etc/apache2/envvars.

4.2 HTTPS

4.2.1 Prerequisite

Shell

```
sudo a2enmod ssl # Enable module SSL
sudo service apache2 restart # Restart Apache
```

Then, you'll need to create your self-signed certificate. If you want to use a third-party certificate, go check these website: `http://www.startssl.com/` and `https://www.cacert.org/`. **Here is a very good tutorial:** `https://www.openssl.org/docs/HOWTO/certificates.txt`.

Shell

```
sudo aptitude install ssl-cert openssl
cd /etc/ssl
```

4.2.2 Generating the certificate

For the *Common Name*, use your domain or a wildcard for several subdomains: for example yourdomain.com or *.yourdomain.com !
Let's get started:

Shell

```
sudo openssl req -new -x509 -nodes -days 1000 -newkey rsa:2048 -sha256 -out server.crt -keyout server.key
sudo chmod o-rw server.key # Security
```

4.2.3 Setup Apache

Enable the ssl website if in a different file than the non-ssl one.

Shell

```
sudo a2ensite default-ssl
service apache2 restart
```

Starting with Apache 2.4, *NameVirtualHost* no longer has any effect, so you can remove it from /etc/apache2/ports.conf and from /etc/apache2/sites-*/*.
Make sure /etc/apache2/ports.conf contains:

File: /etc/apache2/ports.conf

```
NameVirtualHost *:80
Listen 80
<IfModule mod_ssl.c>
    NameVirtualHost *:443
    Listen 443
</IfModule>
```

Don't forget to add these three lines in your *VirtualHost*:

File: /etc/apache2/sites-enabled/whatever

```
SSLEngine on
SSLCertificateFile  /etc/ssl/server.crt
SSLCertificateKeyFile /etc/ssl/server.key
```

Edit /etc/apache2/mods-available/ssl.conf:

File: /etc/apache2/mods-available/ssl.conf

```
# Took from https://www.jeveuxhttps.fr/Apache2
SSLProtocol all -SSLv2 -SSLv3
SSLHonorCipherOrder on
SSLCipherSuite ALL:!aNULL:!eNULL:!LOW:!EXP:!RC4:!3DES:+HIGH:+MEDIUM
SSLCompression off
```

One last step:

Shell

```
sudo service apache2 restart # Restart Apache
```

Your secured website is now fully working!

4.3 Build a lightweight media player

If you want to be able to play local files on the Raspberry-Pi from a webpage accessible via your LAN, check this wonderful Git repository out: `https://github.com/rpellerin/raspberry-pi-media-center`.
Otherwise, if you simply want to overwrite your whole current website and replace it by the lightweight media player, do this:

Shell

```
wget --no-check-certificate https://raw.githubusercontent.com/rpellerin/raspberry-pi-
    media-center/master/setup.sh -O /tmp/airpi.sh && chmod +x /tmp/airpi.sh && /tmp/
    airpi.sh
```

Nice, isn't it? :)

As `youtube-dl` is not installed via the tool `aptitude`, you may want to set up a cron job at every 1:00am to update it automatically. Use this script: `https://github.com/rpellerin/dotfiles/blob/master/scripts/updateYoutubeDl.sh`. Then, in cron:

crontab -e

```
1 0 * * * /home/pi/git/dotfiles/scripts/updateYoutubeDl.sh /home/pi/git/youtube-dl
```

4.4 Add several websites

Fortunately, Apache lets you easily manage several websites (multiple domains or subdomains for example). You simply have to create *VirtualHosts*.

4.4.1 VirtualHosts

Here is an example of a file located at /etc/apache2/sites-enabled/whatever (it's a symlink to /etc/apache2/sites-available/whatever):

File: /etc/apache2/sites-enabled/whatever

```
LogLevel warn
ErrorDocument 500 "Something"
ErrorDocument 404 "Something"
ErrorDocument 503 "Something"

# Main server
ServerName yourdomain.com
DocumentRoot /var/www/www

<Directory />
  Options FollowSymLinks
  AllowOverride None
###### APACHE 2.4 ########
# Require all denied
###### APACHE 2.2 #######
  Order deny,allow
  Deny from all

</Directory>

<Directory /var/www/>
  Options -Indexes +FollowSymLinks +Multiviews
  AllowOverride All
  Order allow,deny
  Allow from all
</Directory>

<Directory "/var/www_alternative">
    Options +Indexes -FollowSymLinks -Multiviews
    AllowOverride None
    Order allow,deny
    allow from all
    php_flag engine off
      # Disable PHP #1
    RemoveHandler .php
      # Disable PHP #2
    DirectoryIndex disabled
      # index.html index.php disabled

    AuthName "Forbidden"
    AuthType Basic
    AuthUserFile "/var/www_alternative/.htpasswd"
    Require valid-user
```

```
</Directory>

<Files ~ "^\.git">
    Order deny,allow
    Deny from all
</Files>

<Directorymatch "^/.*/\.git/">
    Order deny,allow
    Deny from all
</Directorymatch>

################################## www ##################################

<VirtualHost *:80>
  ServerAdmin contact@yourdomain.com

  ServerName yourdomain.com
  ServerAlias www.yourdomain.com
  ServerAlias www.*

  DocumentRoot /var/www/www

  ErrorLog ${APACHE_LOG_DIR}/yourdomain.com.error.log
  # Possible values include: debug, info, notice, warn, error, crit,
  # alert, emerg.
  # LogLevel warn
  CustomLog ${APACHE_LOG_DIR}/yourdomain.com.access.log combined
</VirtualHost>

<VirtualHost *:443>
  # Same content as above

  SSLEngine on
  SSLCertificateFile  /etc/ssl/server.crt
  SSLCertificateKeyFile /etc/ssl/server.key
</VirtualHost>

# Add any extra VirtualHost you want
```

4.4.2 Bonus: DNS zone

If you own a domain name and want to manage several subdomains with *DynHOST*, here is a sample of a working DNS zone:

Sample of a DNS zone hosted by OVH

```
$TTL 86400
@ IN SOA dns20.ovh.net. tech.ovh.net. (2014082000 86400 3600 3600000 86400)
                        IN NS     ns20.ovh.net.
                        IN NS     dns20.ovh.net.
                        IN MX 1   mx0.ovh.net.
                        IN MX 100 mxb.ovh.net.
                     60 IN DYNHOST 1.2.3.4
```

```
                    600 IN TXT   "v=spf1 include:mx.ovh.com ~all"
                        IN TXT   "1|yourdomain.com"
_autodiscover._tcp      IN SRV   0 0 443 mailconfig.ovh.net.
_imaps._tcp             IN SRV   0 0 993 ssl0.ovh.net.
_submission._tcp        IN SRV   0 0 465 ssl0.ovh.net.
autoconfig              IN CNAME mailconfig.ovh.net.
imap                    IN CNAME ssl0.ovh.net.
mail                    IN CNAME ssl0.ovh.net.
pop3                    IN CNAME ssl0.ovh.net.
smtp                    IN CNAME ssl0.ovh.net.
www                     IN CNAME yourdomain.com.
blog                    IN CNAME yourdomain.com.
```

4.5 Monitor your websites

Go have a look at this amazing project or, more simply:

Shell

```
sudo aptitude install goaccess
```

Then, all you have to write is one of the following lines:

Shell

```
goaccess -f /var/log/apache2/access.log -a
# OR
zcat -f /var/log/apache2/*.log* | goaccess
```

4.6 Advanced configuration

First, improve security. In /etc/apache2/conf.d/security, uncomment the following lines:

File: /etc/apache2/conf.d/security

```
ServerSignature Off
Header set X-Frame-Options: "sameorigin" # Require mod_headers
ServerTokens Prod
```

Then, make the *ServerName* you set compliant with your hostname:

File: /etc/hostname

```
yourdomain.com
```

File: /etc/hosts

```
127.0.0.1  yourdomain.com localhost
```

Finally, enable some useful mods and restart Apache:

Shell

```
sudo a2enmod headers
sudo a2enmod rewrite
sudo a2enmod expires
sudo a2enmod negotiation
sudo service apache2 restart
```

4.7 Additional reading

- Difference between _*default*_ and *
- Generate *VirtualHosts* automatically
- Great tutorial
- How to secure Apache with SSL #1
- How to secure Apache with SSL #2
- Full documentation about SSL
- How to manage several domains/subdomains
- Use Apache as a reverse proxy
- Compression #1
- Compression #2

5 Subversion

Apache Subversion (also known as Subversion, or SVN) is a software versioning and revision control system. It is really handy when several programmers are involved in the same project. SVN can be accessed in different ways:

- Locally (file://directory/project)
- Remotely (svn://project/project1). SVN users need to be created
- Via Apache (http://project/project1). The Apache module **dav_svn** is required and dav_svn users need to be created.

We will first setup SVN without Apache. Follow the next steps to install a SVN repository on your Raspberry-Pi.

More details are available here: `http://doc.ubuntu-fr.org/subversion` (french) and `http://www.bortzmeyer.org/access-subversion-apache.html` (french).

5.1 Basic installation

Shell

```
sudo aptitude update
sudo aptitude install subversion
sudo mkdir /var/svn # Repositories will be installed there
sudo svnadmin create /var/svn/project1
```

Then, edit /var/svn/project1/conf/svnserve.conf:

File: /var/svn/project1/conf/svnserve.conf

```
[general]
# Non-auth users: none/read/write
anon-access = none
# Auth users: none/read/write
auth-access = write
# password file, located in the same directory
password-db = passwd
# This option specifies the authentication realm of the repository.
# If two repositories have the same authentication realm, they should
# have the same password database, and vice versa. The default realm
# is repository's uuid.
realm = project1
```

Save and exit. Then, edit /var/svn/project1/conf/passwd:

File: /var/svn/project1/conf/passwd

```
[users]
# name = password
romain = mypassword
```

Save and exit. As these passwords are not encrypted, you should forbid anyone from accessing this file.

Shell

```
sudo chmod o-r /var/svn/project1/conf/authz
sudo chmod o-r /var/svn/project1/conf/passwd
```

Then, run the SVN daemon with

Shell

```
sudo svnserve -d -r /var/svn
```

Subversion will listen on port 3690. You can access it at **svn://raspberry-pi-IP/project1**.

5.2 SVN with Apache

At the moment, SVN has been manually launched with the current user (and its rights) and is only accessible via the svn+ssh protocol. It might be a good idea to allow users to access the repository from a web browser. Here is how to do it.

Shell

```
sudo aptitude install apache2 libapache2-svn
sudo a2enmod dav_svn # In order to enable the dav_svn module
```

Then, we have two possibilities:

1. Setup the repositories one by one. This way you can easily setup a repository in a different way than the other ones, but you'll have to restart Apache each time.
2. Give a global setup to a parent repository (that will contains all the repositories)

First, we'll look at the first possibility.

5.2.1 Setup repositories one by one

Here, we're gonna create a repository called **project1** reachable via `http://raspberry-pi-IP/project1_url`. Open /etc/apache2/mods-enabled/dav_svn.conf with root rights and do the following:

File: /etc/apache2/mods-enabled/dav_svn.conf

```
<Location /project1_url> # Uncomment and change URL
DAV svn # Uncomment
SVNPath /var/svn/project1 # Uncomment and change the path
</Location> # Uncomment
```

Save and exit. Restart Apache:

Shell

```
sudo service apache2 restart
```

Then, you'll have to give all the rights to the Apache user.

Shell

```
rm -rf /var/svn/project1 # To make sure any former project exists
sudo svnadmin create /var/svn/project1
find /var/svn/project1/ -type f -exec chmod 640 {} \;
find /var/svn/project1/ -type d -exec chmod 770 {} \;
chown -R www-data:www-data /var/svn/project1
```

Then, make sure that `http://raspberry-pi-IP/project1_url` is responding. **DO NOT LAUNCH THE SVN DAEMON ANYMORE**. Apache handles it.

To add another repositoy, copy the content between `<Location>` and `</Location>` into a file. Don't forget the change the URL.

5.2.2 Global setup

Here, we're gonna create a repository called **project1** reachable via `http://raspberry-pi-IP/project1_url`. Open /etc/apache2/mods-enabled/dav_svn.conf with root rights and do the following:

File: /etc/apache2/mods-enabled/dav_svn.conf

```
<Location /svn> # Uncomment and change URL if needed
DAV svn # Uncomment
SVNParentPath /var/svn # Uncomment and change the path
SVNListParentPath On # Uncomment ONLY if you want Apache to list all the available repositories at /svn
</Location> # Uncomment
```

Save and exit. Restart Apache:

Shell

```
sudo service apache2 restart
```

Then make sure that `http://raspberry-pi-IP/svn` is responding (lists of all repositories or eror 403 you haven't enable that option). **DO NOT LAUNCH THE SVN DAEMON ANYMORE**. Apache handles it.

Now let's create our repository.

Shell

```
sudo svnadmin create /var/svn/project1
sudo chown -R www-data:www-data /var/svn/project1
```

5.2.3 Authentication

Edit /etc/apache2/mods-enabled/dav_svn.conf.

File: /etc/apache2/mods-enabled/dav_svn.conf

```
AuthType Basic
AuthName "SVN Repository"
AuthUserFile /etc/apache2/dav_svn.passwd
AuthzSVNAccessFile /etc/apache2/access # Optional
Require valid-user
```

If you have added the line `AuthzSVNAccessFile /etc/apache2/access`, this is an example of the content of this file:

File: /etc/apache2/access

```
[groups]
company = john, mickael

[project1:/]
# Anyone can read
* = r
```

```
@company = rw

[project2:/]
* =
mickael = rw

# Specific rights for a specific folder in project1. author (another user not in
    company) can read and write
[project1:/documentation/user]
author = rw
```

Then, creates an allowed user, change the owner of the file and restart Apache:

Shell

```
sudo htpasswd -cs /etc/apache2/dav_svn.passwd john # Only use -c for the first user,
    if you want to add others, don't use the -c option, only -s
sudo chown www-data:www-data /etc/apache2/dav_svn.passwd
sudo service apache2 restart
```

That's all!

5.3 Pre-commit and post-commit hooks

This part comes from http://www.thedumbterminal.co.uk/?action=showArticle&articleId=90 and http://technique.arscenic.org/services-web/subversion-et-trac/article/utiliser-les-pre-post-commit-hooks

Subversion allows you to run a script before a commit is made (pre-commit). If this script exits with a non-zero exit status then the commit is not allowed. You can as well execute post-commit hooks such as sending an email for instance.

You may need to execute the following command in order to make it work (not tested):

Shell

```
sudo aptitude install subversion-tools
```

If our repository was located here /var/lib/svn/repos, then we need to create a pre-commit or post-commit hook file here: /var/lib/svn/repos/hooks/pre-commit. For example:

Shell

```
cd /var/lib/svn/repos/hooks/
sudo cp pre-commit.tmpl pre-commit
sudo cp post-commit.tmpl post-commit
sudo chmod +x pre-commit
sudo chmod +x post-commit
```

This file needs to be executable and readable by the subversion user (or apache user if using webDAV). Any message that the script prints to STDERR will sent to the client committing.

The first argument passed to the script is the location of the repository.
The second is the revision number.

5.3.1 Pre-commit hook

Here is a pre-commit script that will limit a repository size to 100 Megs:

File: /var/lib/svn/repos/hooks/pre-commit

```
#!/bin/bash
# Pre-commit hook to limit the size of a repository
REPOS=\"$1\"
TXN=\"$2\"

#max size for repository (MB)
QUOTA=\"100\"

MEGS=‘du -sm $REPOS | sed -r \"s/^([0-9\\.]+).+/\\1/\"‘

if [ \"$MEGS\" -gt \"$QUOTA\" ]; then
       #send error message to stderr
       echo \"Repository is over ${QUOTA}M in size\" 1>&2
       exit 1;
fi
exit 0;
```

5.3.2 Post-commit hook

And here is a post-commit hook that send emails:

File: /var/lib/svn/repos/hooks/post-commit

```
#!/bin/sh
export LANG="en_US.UTF-8"
REPOS="$1"
REV="$2"
AUTHOR=$(svnlook author --revision $REV $REPOS) # Retrieve the author of the commit
/usr/share/subversion/hook-scripts/commit-email.pl "$REPOS" "$REV" --from "
   user@domain.tld" -s "$AUTHOR |" mailing-list@domain.tld # Send the email
```

5.3.3 Testing

To test that the scripts are working:

Shell

```
cd /var/svn/mon_depot/hooks/
./post-commit /var/lib/svn/repos/ xx # Where xx is a revision number
```

6 Git

http://linuxfr.org/users/ife/journaux/pourquoi-github-saimal-quelques-alternatives#toc_6
http://vietlux.blogspot.fr/2012/07/how-to-set-up-your-own-private-git.html
http://blog.adminrezo.fr/2014/04/auto-hebergement-un-serveur-git-personnalise-sous-debian-ubuntu/
http://git-scm.com/book/en/v2/Git-on-the-Server-Setting-Up-the-Server
https://www.clever-cloud.com/blog/engineering/2015/06/09/git-server-30k-improvement/

7 Create your own seedbox

Creating a seedbox is a good thing for sharing torrents.

7.1 Basic installation

First install it. Then choose a directory for downloaded files, let’s say /transmission-downloads.

Shell

```
sudo aptitude update ; sudo aptitude install transmission-daemon
sudo mkdir /transmission-downloads
sudo chmod 777 /transmission-downloads
sudo service transmission-daemon stop
sudo nano /var/lib/transmission-daemon/info/settings.json # Configuration file, you have to change ’download-dir’ and ’incomplete-dir’ to /transmission-downloads. You should change ’rpc-username’ and ’rpc-password’ as well (in plain text). You should also disable both ’rpc-whitelist-enabled’ and ’rename-partial-files’
sudo service transmission-daemon start # Start the daemon
```

That’s all!

7.2 Change default user that own the downloaded files

If you want to execute the daemon with another user (pi for example), do the following:

1. Stop the daemon and then change the user in /etc/init.d/transmission-daemon to ’pi’
2. Then

   Shell

   ```
   sudo chown pi:pi /etc/transmission-daemon -R
   sudo chown pi:pi /var/lib/transmission-daemon -R
   sudo chown pi:pi /etc/defaults/transmission-daemon -R
   ```

3. Start the daemon.

7.3 Ports and firewall

Add these rules to your firewall:

Shell

```
# Don’t break established connections
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# Torrents: 51413 is the peer port
iptables -A INPUT -p tcp --dport 51413 -j ACCEPT
iptables -A INPUT -p udp --dport 51413 -j ACCEPT
iptables -A OUTPUT -m owner --gid-owner debian-transmission -j ACCEPT
# Web page: 9091 is the webpage port
iptables -A INPUT -p tcp --dport 9091 -j ACCEPT
```

8 Samba

Samba allows you to share public/protected folders over your LAN, with other computers.
Let's say you wanna share:

- /home/pi/Public: Anyone would be able to read and write
- /home/pi/Music: Anyone would ONLY be able to read
- /home/pi/Private: This folder wouldn't be visible over the LAN, and would require credentials to access

Here is how to do it:

Shell

```
sudo aptitude-get install samba samba-common-bin # samba-common-bin has the smbpasswd utility
```

Now, edit /etc/samba/smb.conf like this (for example):

File: /etc/samba/smb.conf

```
[global]
  workgroup = WORKGROUP
  server string = %h server
  netbios name = raspberrypi
  dns proxy = no
  log file = /var/log/samba/log.%m
  max log size = 1000
  syslog = 0
  panic action = /usr/share/samba/panic-action %d
  security = user
  encrypt passwords = true
  passdb backend = tdbsam
  obey pam restrictions = yes
  unix password sync = yes
  passwd program = /usr/bin/passwd %u
  passwd chat = *Enter\snew\s*\spassword:* %n\n *Retype\snew\s*\spassword:* %n\n *password\supdated\ssuccessfully* .
  pam password change = yes
  map to guest = bad user
  usershare allow guests = yes

[Public]
  path =/home/pi/Public
  read only = no
  locking = no
  guest ok = yes
  force user = pi

[Music]
  path =/home/pi/Music
  read only = yes
```

```
   locking = no
   guest ok = yes
   force user = pi

[Private]
   browseable = no
   path = /home/pi/Private
   writable = yes
   username = John
   only user = yes
   create mode = 0600
   directory mask = 0700
```

Then save it and exit. More information about this file can be found at https://www.samba.org/samba/docs/using_samba/ch06.html.
Then, add the user ’John’ to Raspbian and SAMBA:

Shell

```
sudo useradd John -m -G users # -m create the home directory, -G add the user to the group ’users’
sudo passwd John # Set the password for John
sudo smbpasswd -a JeanLouis # Add John to samba
```

Then, give John the rights for the directory /home/pi/Private and restart SAMBA:

Shell

```
sudo chown John:John /home/pi/Private/
sudo service samba restart
```

That’s all!

9 VPN

9.1 The server part

Shell

```
sudo aptitude update ; sudo aptitude install openvpn
sudo -s
cp -r /usr/share/doc/openvpn/examples/easy-rsa/2.0 /etc/openvpn/easy-rsa
cd /etc/openvpn/easy-rsa
nano vars
```

Edit the variable **EASY_RSA** and **KEY_SIZE**:

File: /etc/openvpn/easy-rsa/vars

```
export EASY_RSA="/etc/openvpn/easy-rsa"
export KEY_SIZE=2048
export KEY_COUNTRY="FR"
export KEY_PROVINCE="Department"
export KEY_CITY="City"
export KEY_ORG="domain.com"
export KEY_EMAIL="contact@domain.com"
export KEY_NAME="MyName"
export KEY_OU=" "
export KEY_CN=changeme
```

Save and exit.

Shell

```
source ./vars
./clean-all
./build-ca
# Everything should be prefilled. You can leave the 'Organizational Unit Name' blank
    (hit space key if not already blank). Pay attention to the 'Common Name', it must
     be explicit (e.g. 'FnameLname')
./build-key-server server # You can replace server with any name you wish
# Answer whatever you want to the questions, except for Common Name (which MUST be
    the same name you specified, here it's 'server'), Challenge Password (must be
    left blank) and Sign the certificate? (answer 'y'). Type 'y' to 1 out of 1
    certificate requests certified, commit?
```

Perform the next step for every client you want to set up (a client = a connection through the VPN).

Shell

```
./build-key-pass client1 # Or ./build-key client1 if you don't need PEM pass phrase (
    not recommended)
# Write a PEM pass phrase but the challenge password must be left blank. Don't use
    the same 'Common Name' as at the ./build-ca step. Here the 'Common Name' must be
    'client1'.

# The next command is optional but it seems it's better to do this
```

```
openssl rsa -in keys/client1.key -des3 -out keys/client1.3des.key # You can use the
    same pass phrase and PEM pass phrase as before
```

Then:

Shell

```
./build-dh
openvpn --genkey --secret keys/ta.key
cp /usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz /etc/openvpn/
cd /etc/openvpn
gunzip server.conf.gz
nano /etc/openvpn/server.conf
```

Here are all the important and required lines

File: /etc/openvpn/server.conf

```
port 1194
proto udp #Some people prefer to use tcp. Don't change it if you don't know.
dev tun
ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/server.crt # SWAP WITH YOUR CRT NAME
key /etc/openvpn/easy-rsa/keys/server.key # SWAP WITH YOUR KEY NAME
dh /etc/openvpn/easy-rsa/keys/dh1024.pem # If you changed to 2048, change that here!
server 10.8.0.0 255.255.255.0
# server and remote endpoints
#ifconfig 10.8.0.1 10.8.0.2 # I don't know the goal of this
push "route 192.168.1.0 255.255.255.0" # SWAP THE IP NUMBER WITH YOUR HOME LAN IP
# Set primary domain name server address to the SOHO Router
# If your router does not do DNS, you can use Google DNS 8.8.8.8
push "dhcp-option DNS 192.168.2.1" # This should already match your router address
    and not need to be changed. If it doesn't work, use 8.8.8.8 (tested, this works!)
# Override the Client default gateway by using 0.0.0.0/1 and
# 128.0.0.0/1 rather than 0.0.0.0/0. This has the benefit of
# overriding but not wiping out the original default gateway.
push "redirect-gateway def1 bypass-dhcp"
client-to-client
duplicate-cn
keepalive 10 120
tls-auth /etc/openvpn/easy-rsa/keys/ta.key 0
cipher AES-128-CBC
comp-lzo
user nobody
group nogroup
persist-key
persist-tun
status /var/log/openvpn-status.log
log-append /var/log/openvpn.log
verb 6
mute 20
```

Taken from this file.
Then edit /etc/sysctl.conf:

File: /etc/sysctl.conf

```
net.ipv4.ip_forward=1 # Uncomment this line
```

Then apply the changes:

Shell

```
sysctl -p
```

If you followed the section 2.1, add the following to /etc/init.d/firewall.sh, and comment:

File: /etc/init.d/firewall.sh

```
UDP_SERVICES="1194" # openvpn
# Choose one of the two next lines
iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j SNAT --to-source 192.168.XX.X
# OR, if the IP address on eth0 is not static and reliable
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

# Choose one of the next two set of lines
iptables -A FORWARD -s 10.8.0.0/24 -j ACCEPT
iptables -A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
# OR
iptables -A FORWARD -i tun0 -o eth0 -j ACCEPT
iptables -A FORWARD -i eth0 -o tun0 -j ACCEPT

#echo 0 > /proc/sys/net/ipv4/ip_forward # Comment
```

Otherwise, add the following to /etc/firewall-openvpn-rules.sh

File: /etc/firewall-openvpn-rules.sh

```
#!/bin/sh
iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j SNAT --to-source 192.168.XX.X
# Replace 192.168.XX.X with your Raspberry-Pi IP
# 10.8.0.0 is the default address for Raspberry Pi for clients that are connected to the VPN
```

Then do:

Shell

```
chmod 700 /etc/firewall-Openvpn-rules.sh
chown root /etc/firewall-Openvpn-rules.sh
nano /etc/network/interfaces
```

Below 'iface eth0 inet dhcp', add the following:

File: /etc/network/interfaces

```
     pre-up /etc/firewall-openvpn-rules.sh
```

You can check out /usr/share/doc/openvpn/examples/sample-config-files/firewall.sh, it's pretty useful! Finally, reboot your Raspberry-Pi:

Shell

```
reboot
```

9.2 The client part

Shell

```
sudo -s
cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf /etc/openvpn/easy-rsa/keys/default_client.conf
cd /etc/openvpn/easy-rsa/keys
nano default_client.conf
```

File: /etc/openvpn/easy-rsa/keys/default_client.conf

```
client
dev tun
proto udp
remote <YOUR PUBLIC IP ADDRESS HERE OR DOMAIN NAME> <PORT>
resolv-retry infinite
nobind
user nobody
group nobody
persist-key
persist-tun
mute-replay-warnings
ns-cert-type server
key-direction 1
cipher AES-128-CBC
comp-lzo
verb 1
mute 20
# tls-auth, ca, cert, key are useless, the script below takes care of it
```

Save and exit. Then:

Shell

```
nano /etc/openvpn/easy-rsa/keys/MakeOPVN.sh
```

File: /etc/openvpn/easy-rsa/keys/MakeOPVN.sh

```
#!/bin/bash

# Default Variable Declarations
DEFAULT="default_client.conf"
FILEEXT=".ovpn"
CRT=".crt"
KEY=".3des.key"
CA="ca.crt"
TA="ta.key"

#Ask for a Client name
echo "Please enter an existing Client Name:"
read NAME


#1st Verify that client's Public Key Exists
if [ ! -f $NAME$CRT ]; then
 echo "[ERROR]: Client Public Key Certificate not found: $NAME$CRT"
 exit
fi
```

```
echo "Client’s cert found: $NAME$CR"


#Then, verify that there is a private key for that client
if [ ! -f $NAME$KEY ]; then
 echo "[ERROR]: Client 3des Private Key not found: $NAME$KEY"
 exit
fi
echo "Client’s Private Key found: $NAME$KEY"

#Confirm the CA public key exists
if [ ! -f $CA ]; then
 echo "[ERROR]: CA Public Key not found: $CA"
 exit
fi
echo "CA public Key found: $CA"

#Confirm the tls-auth ta key file exists
if [ ! -f $TA ]; then
 echo "[ERROR]: tls-auth Key not found: $TA"
 exit
fi
echo "tls-auth Private Key found: $TA"

#Ready to make a new .opvn file - Start by populating with the default file
cat $DEFAULT > $NAME$FILEEXT

#Now, append the CA Public Cert
echo "<ca>" >> $NAME$FILEEXT
cat $CA >> $NAME$FILEEXT
echo "</ca>" >> $NAME$FILEEXT

#Next append the client Public Cert
echo "<cert>" >> $NAME$FILEEXT
cat $NAME$CRT | sed -ne ’/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p’ >> $NAME$FILEEXT
echo "</cert>" >> $NAME$FILEEXT

#Then, append the client Private Key
echo "<key>" >> $NAME$FILEEXT
cat $NAME$KEY >> $NAME$FILEEXT
echo "</key>" >> $NAME$FILEEXT

#Finally, append the TA Private Key
echo "<tls-auth>" >> $NAME$FILEEXT
cat $TA >> $NAME$FILEEXT
echo "</tls-auth>" >> $NAME$FILEEXT

echo "Done! $NAME$FILEEXT Successfully Created."

#Script written by Eric Jodoin
```

Taken from *https://gist.github.com/laurenorsini/10013430*.

Then:

Shell

```
cd /etc/openvpn/easy-rsa/keys/
chmod 700 MakeOPVN.sh
./MakeOPVN.sh # Repeat for each client your previously created
cd ..
chmod a+rx /etc/openvpn/easy-rsa/keys/
```

Finally, from a remote computer, get the file (.opvn) to set up the client software:

Shell

```
scp -r pi@serveur:/etc/openvpn/easy-rsa/keys .
```

9.3 Troubleshooting

On Linux, you may encounter issues with DNS names resolution. Here is what to do on your client:

Shell

```
sudo aptitude install resolvconf
cd /etc
# Backup your original resolv.conf just in case
cp resolv.conf resolv.conf.orig
```

Then, at the end of your .opvn file, add theses lines before the certificates:

File: whatever.opvn

```
script-security 2
up /etc/openvpn/update-resolv-conf
down /etc/openvpn/update-resolv-conf
```

9.4 Additional readings

- `http://readwrite.com/2014/04/10/raspberry-pi-vpn-tutorial-server-secure-web-browsing`
- `http://wiki.korben.info/creation_dun_serveur_openvpn`
- **Best one:** `https://openvpn.net/index.php/open-source/documentation/howto.html`

10 Proxy

Basically, a proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers. It can be seen as an intermediate between you and a remote server (a web server for example). It allows you to filter the traffic (block some websites, redirect, reverse proxy, etc).

On Linux, you have two possibilities. One very light-weight, **tinyproxy** with no extra features. An other one has many interesting features (authentification for example) and is called **Squid**. I recommend you to choose **tinyproxy** only for local use or temporarely.

10.1 tinyproxy

Shell

```
sudo aptitude install tinyproxy
sudo nano /etc/tinyproxy.conf
```

In the conf file, you can allow any IP address by commenting all the lines `Allow`. You can also block some websites, or only accept some of them (see `Filter` in the file). Example of this file:

File: /etc/tinyproxy.conf

```
# Change the default port
Port 3129

# Only allow the websites in the filter instead of allowing everything excepted those ones
FilterDefaultDeny Yes

ConnectPort 443
#ConnectPort 563
```

Once edition done, do:

Shell

```
sudo service tinyproxy restart
```

More information: http://doc.ubuntu-fr.org/tinyproxy

10.2 Squid

10.2.1 Installation and settings

We'll install Squid and set up a basic authentification, plus some others interesting settings.

Shell

```
sudo aptitude install squid apache2-utils # apache2-utils required for htpasswd tool
sudo touch /etc/squid/users
sudo chown root:proxy /etc/squid/users
sudo chmod 0640 /etc/squid/users
sudo htpasswd -m /etc/squid/users <login> # Replace <login> with the name you want
# Repeat the last command as many times as required
```

Everything should be OK with users. You can check by doing this:

Shell

```
sudo /usr/lib/squid/ncsa_auth /etc/squid/users
<login> <password>
```

Now, the best part! Edit the conf file /etc/squid/squid.conf (you should make a backup first):

File: /etc/squid/squid.conf

```
############################### SECURITY ###############################
httpd_suppress_version_string on


################################# LOGS #################################
# Date format for logs
#logformat squid %ts.%03tu %6tr %>a %Ss/%03Hs %<st %rm %ru %un %Sh/%<A %mt
logformat squid %tl %un %6tr %>a %Ss/%03Hs %<st %rm %ru %Sh/%<A %mt

access_log /var/log/squid/access.log squid
useragent_log         /var/log/squid/useragent.log
referer_log           /var/log/squid/referer.log


########################### AUTHENTICATION ###########################
auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/users
auth_param basic children 5 # Number of simultaneous clients
auth_param basic realm Please authenticate
auth_param basic credentialsttl 2 hours


################################ ACL ################################
acl all src all          # ACL to allow any network
acl lan src 192.168.1.0/24 # ACL to allow LAN

acl purge method PURGE
acl CONNECT method CONNECT

acl SSL_ports port 443 # HTTPS
acl Safe_ports port 80 # Port HTTP
acl Safe_ports port 443 # Port HTTPS

acl deny_domain url_regex -i "/etc/squid/filter" # File with blocked websites

acl Users proxy_auth REQUIRED # ACL to allo authentificated users


http_port 3128
icp_port              0
snmp_port             0

visible_hostname domain.tld
cache_mgr contact@domain.tld

# Disable cache
# Read this http://wiki.squid-cache.org/SquidFaq/ConfiguringSquid#
    Can_I_make_Squid_proxy_only.2C_without_caching_anything.3F
cache deny all

# Depending on your needs...
#forwarded_for delete
via off

acl apache rep_header Server ^Apache
broken_vary_encoding allow apache

http_access deny purge
http_access deny CONNECT !SSL_ports
http_access deny !Safe_ports
http_access deny deny_domain
http_access allow Users
http_access deny all
```

Everything should be OK now. Simply restart **Squid**:

Shell

```
sudo service squid restart
```

10.2.2 Additional readings

More information:

- `http://doc.ubuntu-fr.org/squid`
- `http://www.it-connect.fr/tutoriels/unix-linux/squid/`
- `http://wiki.squid-cache.org/SquidFaq/ConfiguringSquid`
- `http://wiki.squid-cache.org/Features`

Authentification: `http://wiki.squid-cache.org/Features/Authentication`
If you want to set up a proxy between your LAN and the Internet read this: `http://wiki.squid-cache.org/ConfigExamples/Intercept/LinuxRedirect`

11 Mail server

Here is how to build your own mail server.
http://www.isalo.org/wiki.debian-fr/index.php/Configuration_dun_serveur_mail_avec_Postfix
http://nicodewaele.free.fr/Site/Stockage/Gnu-Linux/serveur-mail-postfix-courier-imap-ubuntu.pdf

Here is how to simply send emails.
http://doc.ubuntu-fr.org/ssmtp

12 ownCloud

ownCloud is a software system for what is commonly termed "file hosting". As such, ownCloud is very similar to the widely used Dropbox, with the primary difference being that ownCloud is free and open-source, and thereby allowing anyone to install and operate it without charge on a private server, with no limits on storage space (except for hard disk quota or capacity) or the number of connected clients. — Wikipédia

12.1 Basic installation

12.1.1 Requirements

First, download the lastest version available on the official website and extract it.

Shell

```
wget "http://download.owncloud.org/community/owncloud-latest.tar.bz2"
tar -xjf owncloud-latest.tar.bz2
```

Make sure you already have Apache and PHP installed, along with SQLite.

Shell

```
sudo apt-get update
sudo apt-get install apache2 php5 php5-mysql mysql-server php5-json php-xml
    -parser php5-gd php5-imagick php5-intl php5-mcrypt libmcrypt4 curl php5-
    curl php5-cgi php5-cli php-pear php-apc # php-apc will significantly
    improve performance (might be replaced by php5-apcu)
# Check other requirements at https://doc.owncloud.org/server/8.0/
    admin_manual/installation/source_installation.html#prerequisites and at
    https://forum.owncloud.org/viewtopic.php?f=8&t=18742
```

If you wish to choose SQLite over MySQL (which is not recommended at all), do the following:

Shell

```
sudo apt-get install php5-sqlite
```

12.1.2 Create MySQL database and user for ownCloud

Shell

```
mysql -u root -p

CREATE USER 'ownclouduser'@'localhost' IDENTIFIED BY 'Password';
create database owncloud;
GRANT ALL ON owncloud.* TO 'ownclouduser'@'localhost';
flush privileges;
exit;

sudo mysql_secure_installation
```

Configure your ownCloud following this tutorial (it's often done automatically).

12.1.3 Installation

Then, copy ownCloud to your web server, in the desired directory.

Shell

```
cp -r owncloud /var/www # Will create the directory /var/www/owncloud/
rm -rf owncloud*
sudo chown -R www-data:www-data /var/www/owncloud
```

You may improve security following this tutorial, but **only after having visited your ownCloud webpage at least once**.

Configure your Apache virtual hosts, following section 4.4. Make sure you set `AllowOverride All` for /var/www/owncloud/. You also need to enable some mods:

Shell

```
sudo a2enmod rewrite
sudo a2enmod headers
```

Then, change some settings in /var/www/owncloud/.htaccess:

File: /var/www/owncloud/.htaccess

```
php_value upload_max_filesize 16G
php_value post_max_size 16G
php_value memory_limit 512M
php_value output_buffering 0
php_value default_charset "UTF-8"
```

You may apply the same changes in /etc/php5/apache2/php.ini. **It's optional and useless**. You can check that everything has been propagated by creating a file in your ownCloud directory (/var/www/owncloud/infos.php for example) with `<?php phpinfo();` inside.

File: /etc/php5/apache2/php.ini

```
upload_max_filesize = 16G
post_max_size = 16G
output_buffering = 0
```

Restart Apache2:

Shell

```
sudo service apache2 restart
```

You can now access your ownCloud webpage.

12.2 Security

To use ownCloud with HTTPS, please refer to section 4.2. Then, force HTTPS, either in the admin webpage or by editing /var/www/owncloud/config/config.php:

File: /var/www/owncloud/config/config.php

```
'forcessl' => true,
```

You should use encryption on the server-side as well. Simply enable the app called 'Encryption' in the web interface, while logged in as Admin. **However, please note that it often makes ownCloud unstable.**

12.3 Tutorials

For the full tutorial, please go to the official documentation.
To setup clients, read: `http://doc.ubuntu-fr.org/owncloud#utilisation_avec_nautilus`.
Official website: `http://owncloud.org/`

13 RSS aggregator

Here we're gonna use FreshRSS.

13.1 Basic setup

Shell

```
sudo apt-get update
sudo apt-get install apache2 php5 php5-mysql mysql-server php5-json php-xml-parser curl php5-curl # Check other requirements at https://github.com/FreshRSS/FreshRSS#requirements and at http://freshrss.org/#requirements
wget https://github.com/FreshRSS/FreshRSS/archive/master.zip
unzip master.zip
mv FreshRSS-master/ /var/www/freshrss
sudo su
chown -R www-data:www-data /var/www/freshrss/
find /var/www/freshrss/ -type f -print0 | xargs -0 chmod 0440
find /var/www/freshrss/ -type d -print0 | xargs -0 chmod 0550
chmod u+w /var/www/freshrss/data -R
su www-data
crontab -e
```

crontab -e

```
* * * * php /var/www/freshrss/app/actualize_script.php > /tmp/FreshRSS.log 2>&1
```

13.2 Configure Apache

Configure your Apache virtual hosts, following section 4.4 if needed. **You should set your `DocumentRoot` to /var/www/freshrss/p.**

/etc/apache2/sites-enabled/freshrss

```
<VirtualHost *:80>
    DocumentRoot /var/www/freshrss/p
    ServerName demo.freshrss.org

    <Directory /var/www/freshrss/p>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

We're done!

14 Go further

Here are some ideas for your Raspberry-Pi.

14.1 Make regular backups of your main computer

Check out some useful tools such as `rsync` and `cron`, or read these tutorials:

- `http://doc.ubuntu-fr.org/tutoriel/sauvegarder_home_avec_rsync`
- `http://godefroy.me/mes-solutions-de-backups-et-de-synchronisations-rsync-cron-a91945`

14.2 50 ideas for your Raspberry-Pi

- `http://korben.info/idees-raspberry-pi.html`

14.3 Set up your Raspberry-Pi as a wireless hotspot

- `http://elinux.org/RPI-Wireless-Hotspot`

14.4 Add DLNA support

- `http://www.linuxjournal.com/content/raspberry-pi-perfect-home-server?page=0,1`

14.5 Build a print server

- `http://www.linuxjournal.com/content/raspberry-pi-perfect-home-server?page=0,1`

14.6 Remotely manage your whole server

- `http://doc.ubuntu-fr.org/webmin`

14.7 Build a dedicated server

- `http://www.skyminds.net/serveur-dedie-installation-dapache-php-mysql-et-webmin/#crayon-53d5839562c34565908359`

14.8 Some other ultimate tutorials

- `http://www.instructables.com/id/Ultimate-Pi-Based-Home-Server/`
- `https://mattwilcox.net/archives/setting-up-a-secure-home-web-server-with-raspberry-pi/`
- `http://blog.idleman.fr/sommaire-raspberry-pi-arduino-domotique-robotique-et-pleins-de-trucs-en-ique/`

- http://www.alsacreations.com/tuto/lire/621-Configuration-d-un-serveur-dedie-de-A-a-Z.html

14.9 Run program at start-up

- http://www.stuffaboutcode.com/2012/06/raspberry-pi-run-program-at-start-up.html

14.10 Firewall and Rkhunter

- http://www.alsacreations.com/tuto/lire/622-Securite-firewall-iptables.html

14.11 Handle Livebox loopback issue

- http://www.sheldon.fr/2014/10/la-livebox-et-son-loopback-go-fuck/

14.12 Syncing

- http://www.darkcoding.net/software/sync-a-unix-way/

14.13 Many little tutorials

- http://wiki.queret.net/start

14.14 Wireless Raspberry Pi print server

- http://www.linuxuser.co.uk/tutorials/wireless-raspberry-pi-print-server

14.15 Test your server: security

- http://blog.adminrezo.fr/2012/03/tutoriel-analyse-de-failles-de-securite-avec-le-scanner-openvas/

14.16 Create your own PopCorn Time

- http://beunlike.com/faire-son-popcorn-time-en-10-min/