Para crear la imagen con Raspbian ejecutaremos:

```
sudo dd bs=4M if=2020-08-20-raspios-buster-armhf-lite.img of=/dev/sdh conv=fsync status=progress
sudo sync
# montar en /run/media/procamora/boot/
touch /run/media/procamora/boot/ssh
sudo sync
```

Instalación y configuración inicial

Lo primero que vamos a realizar es comprobar si tenemos bien configurados las configuraciones de idiomas, y así evitar posibles errores a la hora de actualizar e instalar nuevos programas. Para ello vamos a ejecutar el comando *locale* y ver si están configuradas todas las variables de entorno, en caso de que haya alguna vacia vamos a ejecutar:

```
export LANGUAGE=en_US.UTF-8
export LANG=en_US.UTF-8
export LC_ALL=en_US.UTF-8
sudo dpkg-reconfigure locales
sudo locale-gen en_US.UTF-8
```

Normalmente las que suelen estar vacías son: LANGUAGE y LC_ALL.

Una vez que ya tenemos configurados correctamente los locales podemos actualizar el repositorio, los paquetes y el firmware con los comandos:

```
sudo apt update && sudo apt dist-upgrade -y
sudo reboot

# Reinicio recomendable

#sudo rpi-update # no recomendable
sudo reboot
```

export LC_ALL=C.UTF-8 ## IMPORTANTE PARA QUE FUNCIONE descomprime_rar.py

Configuración inicial

Una vez que esto todo el sistema actualizado procedemos a ejecutar el comando y a ir nivel a nivel realizando todas las configuraciones de la Rasperry:

```
sudo raspi-config
```

1. Change User Password (Change password for the 'pi' user)
2. Network Options (Configure network settings) N1 Hostname N2 Wi-Fi N3 Network interfaces names (disable predicrable network interfaces)
3. Boot Options (Configure options for start-up)
4. Localisation Options (Set up language and regional settings to match your location) I1 Change Locale en_US.UTF-8 UTF-8 Default en_US.UTF-8 I2 Change Timezone Europe > Madrid I3 Change Keyborad Layaout I4 Change Wi-fi Country
5. Interfacing Options Deshabilitar todo menos SSH P2 SSH habilitar
6. Advanced Options (Configure advanced settings) A1 Expand Filesystem (Ensures that all of the SD card storage is available to the OS)

Instalación de paquetes

Los paquetes que vamos a necesitar siempre son los siguientes:

```
sudo apt install -y vim encfs rsync zsh git python3 python3-pip python3-dev trash-cli screen gcc make \
 cmake sshfs expect smbclient tar tcpdump htop build-essential wget curl tmux raspberrypi-kernel \
 raspberrypi-kernel-headers atop
```

En caso de instalar todos los servicios, necesitaremos los paquetes:

```
sudo apt install -y samba transmission-daemon transmission-cli transmission-common wakeonlan vsftpd apache2 a
```

Configuración zsh

Para la configuración del entorno con *zsh* vamos a utilizar el script que he desarrollado que automatiza todo el proceso:

```
git clone https://github.com/procamora/custom_workspace
cd custom_workspace
./custom_workspace.sh zsh
```

Fichero hosts

```
git clone https://github.com/StevenBlack/hosts
pip3 install --user -r hosts/requirements.txt

echo "# Custom host records are listed here.
192.168.1.1     router
192.168.1.71    rp4 backuppc.procamora.com owncloud.procamora.com procamora.myvnc.com
192.168.1.72    zero
192.168.1.73    rp3
192.168.1.74    win10
192.168.1.75    xiaomi
192.168.1.76    huawei
192.168.1.77    mia1
192.168.1.78    4770k
# End of custom host records.
" > hosts/myhosts

python3 hosts/testUpdateHostsFile.py
sudo python3 hosts/updateHostsFile.py --auto --replace > /dev/null # Tarda unos 15 segundos
```

Fichero crontab

```
# Example of job definition:
# .---------------- minute (0 - 59)
# | .------------- hour (0 - 23)
# | |       .---------- day of month (1 - 31)
# | |       | .------- month (1 - 12) OR jan,feb,mar,apr ...
# | |       | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | |       | | |
# * *       * * *   user    command to be executed

#GESTOR SERIES
*/20 *      * * *   pi      cd /home/pi/series_manager/ && python3 /home/pi/series_manager/app/utils/descarga
0 *         * * *   pi      cd /home/pi/series_manager/ && python3 /home/pi/series_manager/app/utils/descompr

# Mantenimiento
*/15 *      * * *   pi      find /home/pi/scripts/ -type f ! -executable -regex '.*\.\(sh\|py\|exp\)' -exec c
30 22       * * *   pi      cd /home/pi/decrypt_gdrive/ && bash rsync_rp3.sh > /tmp/rsync.log 2>&1
#0 *        * * *   pi      cd /home/pi/scripts && python3 /home/pi/scripts/insync.py  >> /tmp/insync.log 2>&
#*/15 *      * * *   pi      sudo find /media/HDD/* -type d -prune -print | grep -v backup | sed -r 's/\s+/\\
#*/15 *      * * *   pi      sudo find /media/HDD/* -type d -prune -print | grep -v backup | sed -r 's/\s+/\\

#DOMOTICA
#20 8       * * *   pi      /home/pi/tg/bin/telegram-cli -e 'msg domotica_pablo "/modo_automatico on 23"' >/t
#30 8       * * *   pi      /home/pi/tg/bin/telegram-cli -e 'msg domotica_pablo "/set_rele off"' >/tmp/tg_off

##WIKI PERSONAL
```

```
*/15 *      * * *   pi      cd /home/pi/wiki/ && bash pushgit.sh 1 >> /tmp/wiki_cont 2>&1

# Bots
#* *         * * *   root    sudo systemctl -q is-active mio_bot_recordatorios.service && echo YES   || sudo
* *         * * *   root    sudo systemctl -q is-active mio_bot_common_ports.service && echo YES    || sudo s
* *         * * *   root    sudo systemctl -q is-active mio_bot_dictionary.service && echo YES      || sudo s
* *         * * *   root    sudo systemctl -q is-active mio_bot_minitrue.service && echo YES        || sudo s
* *         * * *   root    sudo systemctl -q is-active mio_bot_series_manager.service && echo YES  || sudo s
```

Fichero fstab

```
echo "UUID=1eb2f592-bee9-462d-9ac5-263870d2c04a       /media/HDD      btrfs                   defaults

sudo mkdir -p /media/HDD
sudo mount -a
```

Gestión de claves privadas

Para dejar el directorio SSH con los permisos adecuados ejecutamos:

```
eval "$(ssh-agent -s)"

mkdir -p ~/.ssh
chmod 700 ~/.ssh

touch ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys  # le quitamos los permisos necesarios
```

En caso de querer generar un certificado SSH para servicios y sin contraseña podemos ejecutar los siguientes comandos, pero es mejor usar el que ya tengo generado y que esta añadido en el resto de hosts.

```
ssh-keygen -t rsa -b 1024 -f ~/.ssh/services -N "" -q -C "key used for automation service connections"
ssh-add ~/.ssh/services    # ponemos la contraseña y ya tenemos cargada la clave
ssh-add -l


ssh-copy-id -i ~/.ssh/services root@zero
ssh-copy-id -i ~/.ssh/services pi@zero

ssh-copy-id -i ~/.ssh/services root@4770k
ssh-copy-id -i ~/.ssh/services procamora@4770k
```

Swap

Raspberry OS suele venir configurado con 100Mb de swap, algo que en ciertas ocasiones en insuficiente, podemos aumentarlo a 2Gb con los siguientes comandos:

```
cat /proc/meminfo | grep Swap
sudo sed -i "s/CONF_SWAPSIZE=[[:digit:]]\+/CONF_SWAPSIZE=2048/" /etc/dphys-swapfile
sudo dphys-swapfile setup
sudo dphys-swapfile swapon
cat /proc/meminfo | grep Swap
```

Transmission

La instalación de Transmission requiere de algo mas de configuración a nivel de permisos, ya que si se dejan los permisos por defecto, a la hora de acceder por sshfs con el usuario pi nos encontraremos que no tenemos que no tenemos permisos para borrar las descargas.

El primero paso sera instalar el servicio y habilitar para que se inicie con el sistema, una vez hecho esto lo paramos y lo configuramos. Es muy importante pararlo, ya que si se modifica estando encendido el fichero modificado se borrara y se mantendrá el fichero original.

```
sudo apt-get install transmission-daemon transmission-cli
sudo systemctl enable transmission-daemon.service
sudo systemctl stop transmission-daemon.service
```

Una vez parado el servicio, ya podemos modificar el fichero */etc/transmission-daemon/settings.json*. A continuación pongo mi fichero de configuración con las partes mas relevantes:

- *"rpc-username": "pi"*: El usuario para acceder por la interfaz web.
- *"rpc-password": "raspberry"*: La contraseña en texto plano para acceder, una vez que se inicie se pondrá su SHA1.
- *"watch-dir-enabled": true*: Habilitar que se añaden automáticamente todos los torrent que se en encuentren en un direcotrio
- *"watch-dir": "/home/pi/Downloads"*: Directorio desde donde añaden automáticamente todos los torrents.
- *"download-dir": "/media/HDD"*: Directorio donde se guardan las descargas una vez completadas.
- *"incomplete-dir": "/media/HDD/tmp"*: Directorio donde se guarnan las descargas durante el proceso de descarga.
- *"umask": 2*: Para que los permisos en las descargas sean 775.

```
echo '{
    "alt-speed-down": 20000,
    "alt-speed-enabled": false,
    "alt-speed-time-begin": 540,
    "alt-speed-time-day": 127,
    "alt-speed-time-enabled": false,
    "alt-speed-time-end": 1020,
    "alt-speed-up": 500,
    "bind-address-ipv4": "0.0.0.0",
    "bind-address-ipv6": "::",
    "blocklist-enabled": false,
    "blocklist-url": "https://github.com/sahsu/transmission-blocklist/releases/download/1.0.0/blocklist.gz",
    "cache-size-mb": 4,
    "dht-enabled": true,
    "download-dir": "/media/HDD",
    "download-limit": 100,
    "download-limit-enabled": 0,
    "download-queue-enabled": true,
    "download-queue-size": 6,
    "encryption": 1,
    "idle-seeding-limit": 30,
    "idle-seeding-limit-enabled": true,
    "incomplete-dir": "/media/HDD/tmp",
    "incomplete-dir-enabled": true,
    "lpd-enabled": false,
    "max-peers-global": 200,
    "message-level": 1,
    "open-file-limit": 32,
    "peer-congestion-algorithm": "",
    "peer-id-ttl-hours": 6,
    "peer-limit-global": 200,
    "peer-limit-per-torrent": 50,
    "peer-port": 51413,
    "peer-port-random-high": 65535,
    "peer-port-random-low": 49152,
    "peer-port-random-on-start": false,
    "peer-socket-tos": "default",
    "pex-enabled": true,
    "port-forwarding-enabled": false,
    "preallocation": 1,
    "prefetch-enabled": true,
    "queue-stalled-enabled": true,
    "queue-stalled-minutes": 30,
    "ratio-limit": 1,
```

```
    "ratio-limit-enabled": true,
    "rename-partial-files": true,
    "rpc-authentication-required": true,
    "rpc-bind-address": "0.0.0.0",
    "rpc-enabled": true,
    "rpc-host-whitelist": "",
    "rpc-host-whitelist-enabled": false,
    "rpc-password": "raspberry",
    "rpc-port": 9091,
    "rpc-url": "/transmission/",
    "rpc-username": "pi",
    "rpc-whitelist": "0.0.0.0",
    "rpc-whitelist-enabled": false,
    "scrape-paused-torrents-enabled": true,
    "script-torrent-done-enabled": false,
    "script-torrent-done-filename": "",
    "seed-queue-enabled": false,
    "seed-queue-size": 10,
    "speed-limit-down": 600,
    "speed-limit-down-enabled": false,
    "speed-limit-up": 750,
    "speed-limit-up-enabled": false,
    "start-added-torrents": true,
    "trash-original-torrent-files": true,
    "umask": 2,
    "upload-limit": 100,
    "upload-limit-enabled": false,
    "upload-slots-per-torrent": 14,
    "utp-enabled": true,
    "watch-dir": "/home/pi/Downloads",
    "watch-dir-enabled": true
}' | sudo tee /etc/transmission-daemon/settings.json
```

El siguiente paso es establecer los permisos necesarios para que todo funcione correctamente, teniendo en cuenta que al disco se va a acceder por varios usuarios, y se tiene que poder escribir y borrar datos.

Para hacer esto se va a crear el grupo *hdd* al que vamos a meter los usuarios que necesitan tener acceso al disco, que serán *pi*, *debian-transmission* y *backuppc* (en caso de usarse). Al usuario pi que se la a añadir el grupo como secundario, pero al usuario *debian-transmission* se le va a poner este grupo como principal, para que al descargar los torrent le ponga permisos del grupo *hdd* y así el resto de usuarios tengan permisos sobre estos.

Finalmente vamos a establecer los permisos 775 en el disco para indicar el los usuarios del grupo *hdd* tienen control total:

```
sudo addgroup hdd
sudo usermod -a -G hdd pi
sudo usermod -g hdd debian-transmission # el grupo principal de debian-transmission tiene que ser HDD para qu
sudo usermod -a -G hdd www-data
sudo usermod -a -G hdd backuppc # cuidado que puede que aun no exista
sudo usermod -a -G www-data backuppc



sudo chown pi:hdd /media/HDD/
sudo chmod 755 /media/
sudo chmod 775 /media/HDD/ -R

mkdir -p /home/pi/Downloads
sudo chown pi:hdd /home/pi/Downloads -R          # debian-transmission permiso leer torrent
sudo chmod 775 /home/pi/Downloads
```

Una vez configurado el servicio, ya podemos iniciarlo, es importante hacer un *start* y no un *restart*, ya que de este modo la contraseña que se encuentra en texto plano sera sustituida por su hash.

```
sudo systemctl start transmission-daemon.service
```

owncloud

```
echo "ServerName 127.0.0.1" | sudo tee -a /etc/apache2/apache2.conf

my_lang=$(cat /etc/default/locale | grep LANG | awk -F = '{print $2}')
sudo sed -E -i.back "s/LANG\=(.*)$/LANG=$my_lang/g" /etc/apache2/envvars



# Create a Virtual Host Configuration
FILE="/etc/apache2/sites-available/owncloud.conf"
echo 'Alias /owncloud "/var/www/owncloud/"

<Directory /var/www/owncloud/>
  Options +FollowSymlinks
  AllowOverride All

 <IfModule mod_dav.c>
  Dav off
 </IfModule>

 SetEnv HOME /var/www/owncloud
 SetEnv HTTP_HOME /var/www/owncloud
</Directory>' | sudo tee $FILE


sudo a2ensite owncloud.conf
sudo a2enmod dir env headers mime rewrite setenvif
sudo systemctl restart apache2


#Configure the Database
sudo mysql -u root -e "CREATE DATABASE IF NOT EXISTS owncloud; \
GRANT ALL PRIVILEGES ON owncloud.* \
  TO owncloud@localhost \
  IDENTIFIED BY 'ppu908#m1'";


# Download owncloud
cd /var/www
sudo wget https://download.owncloud.org/community/owncloud-complete-20200731.tar.bz2
sudo tar -xjf owncloud-complete-20200731.tar.bz2

sudo chown www-data:www-data /var/www/owncloud/ -R
cd /var/www/owncloud/


sudo mkdir -p /media/HDD/owncloudData/
sudo chown www-data:hdd /media/HDD/owncloudData/
sudo chmod 700 /media/HDD/owncloudData/


firefox http://192.168.1.71/owncloud/

#CONFIG OWNCLOUD
#owncloud
```

```
#admin/ppu908#m2

#/media/HDD/owncloudData/

#owncloud
#ppu908#m1
#owncloud


sudo systemctl enable apache2 mariadb
```

VPN

```
curl -L https://install.pivpn.io | bash
```

sudo su -c "echo 1 > /proc/sys/net/ipv4/ip_forward" sudo sysctl -w net.ipv4.ip_forward=1 sudo iptables -t nat -A POSTROUTING -s 10.6.0.0/24 -o eth0 -j MASQUERADE

En caso de fallo por actualización del Kernel

```
sudo echo "1" > /boot/.firmware_revision # cambiar hash firmware para forzar descargarlo de nuevo
sudo rpi-update
sudo reboot
sudo apt-get install --reinstall wireguard-dkms
```

configurar firewall con input drop src 192.168.1.1

tmux

insync

Usar el articulo: add-count-insync-portable-cli

```
wget https://procamora.github.io/downloads/insync/insync-armhf_1.3.17.36167_i386.tar.bz2 -O $HOME/Downloads/i
tar -xjf $HOME/Downloads/insync-armhf_i386.tar.bz2 -C $HOME
cd $HOME/insync-portable
firefox https://insynchq.com/auth
./insync-portable add_account --auth-code 2/5ACzcYmorAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
./insync-portable start
./insync-portable manage_selective_sync pablojoserocamora@gmail.com
./insync-portable get_sync_progress
cd -
ln -s ~/pablojoserocamora@gmail.com/scripts/ ~
mkdir ~/decrypt_gdrive/
```

Si manege_selective da un fallo de *curses.error: setupterm: could not find terminal*. Ejecutar el comando:

```
```bash
export TERM=xterm
export TERMINFO=/etc/terminfo
```
```

ftp

https://www.solvetic.com/tutoriales/article/3212-como-instalar-servidor-ftp-linux-centos7/

unrar

Para podemos descomprimir ficheros rar protegidos con contraseña vamos a descargarnos el paquete unrar non-free para arm:

```
wget https://procamora.github.io/downloads/unrar_5.2.7-0.1_armhf.deb
sudo dpkg -i unrar_5.2.7-0.1_armhf.deb
```

Importante: No instalar el paquete disponible en los repositorios (unrar-free), ya no funciona con rar con contraseña.

motd

```
sudo rm /etc/motd
sudo rm /etc/update-motd.d/10-uname
#sudo sed -i -E 's/^#?PrintLastLog (yes|no)$/PrintLastLog no/' /etc/ssh/sshd_config
sudo wget https://raw.githubusercontent.com/gagle/raspberrypi-motd/master/motd.sh -O /etc/update-motd.d/10-st
sudo chown root:root /etc/update-motd.d/10-stats
sudo chmod +x /etc/update-motd.d/10-stats
#sudo systemctl restart sshd
```

Programas propios

```
git config --global user.email "pablojoserocamora@gmail.com"
git config --global user.name "procamora"

# Minitrue
git clone git@github.com:procamora/minitrue $HOME/minitrue
sudo pip3 install -r $HOME/minitrue/requirements.txt
sudo apt install -y nmap texlive-latex-recommended texlive-latex-extra texlive-lang-english
chmod u+x $HOME/minitrue/bot_minitrue.py
chmod 644 $HOME/minitrue/mio_bot_minitrue.service
sudo cp $HOME/minitrue/mio_bot_minitrue.service /lib/systemd/system/



# Series Manager
git clone git@github.com:procamora/Gestor-Series $HOME/series_manager
pip3 install --user -r $HOME/series_manager/requirements_bot.txt
chmod u+x $HOME/series_manager/app/utils/bot_series.py
chmod 644 $HOME/series_manager/mio_bot_series_manager.service
sudo cp $HOME/series_manager/mio_bot_series_manager.service /lib/systemd/system/



# Bot Dicctionary
git clone git@github.com:procamora/bot_dictionary $HOME/bot_dictionary
pip3 install --user -r $HOME/bot_dictionary/requirements.txt
chmod u+x $HOME/bot_dictionary/bot_dictionary.py
chmod 644 $HOME/bot_dictionary/mio_bot_dictionary.service
sudo cp $HOME/bot_dictionary/mio_bot_dictionary.service /lib/systemd/system/



# Bot Common Ports
git clone git@github.com:procamora/bot_common_ports $HOME/bot_common_ports
pip3 install --user -r $HOME/bot_common_ports/requirements.txt
chmod u+x $HOME/bot_common_ports/bot_common_ports.py
chmod 644 $HOME/bot_common_ports/mio_bot_common_ports.service
sudo cp $HOME/bot_common_ports/mio_bot_common_ports.service /lib/systemd/system/



# Bot Promox
git clone git@github.com:procamora/bot_proxmox $HOME/bot_proxmox
pip3 install --user -r $HOME/bot_proxmox/requirements.txt
sudo apt install -y wakeonlan
```

```
chmod u+x $HOME/bot_proxmox/bot_proxmox.py
chmod 644 $HOME/bot_proxmox/mio_bot_proxmox.service
sudo cp $HOME/bot_proxmox/mio_bot_proxmox.service /lib/systemd/system/


# Wiki personal

pip3 install --user -r $HOME/wiki-personal/requirements.txt

# Manage systemd
sudo systemctl daemon-reload
sudo systemctl enable mio_bot_minitrue.service && sudo systemctl start mio_bot_minitrue.service
sudo systemctl enable mio_bot_series_manager.service && sudo systemctl start mio_bot_series_manager.service
sudo systemctl enable mio_bot_dictionary.service && sudo systemctl start mio_bot_dictionary.service
sudo systemctl enable mio_bot_common_ports.service && sudo systemctl start mio_bot_common_ports.service
sudo systemctl enable mio_bot_proxmox.service && sudo systemctl start mio_bot_proxmox.service

# pull all repos
DIRS=( minitrue series_manager bot_dictionary bot_common_ports bot_proxmox )
for dirs in "${DIRS[@]}"; do
    echo $HOME/$dirs
    git -C $HOME/$dirs pull && git -C $HOME/$dirs status
done
```

BackupPC

```
wget -q -O - https://gist.githubusercontent.com/procamora/c32acb2f3ca4fb49c66e879644b11dc2/raw/76eb16ddaee211
sudo chmod 755 /etc/BackupPC
sudo htpasswd  /etc/BackupPC/BackupPC.users backuppc



sudo usermod -a -G hdd backuppc # cuidado que puede que aun no exista
sudo usermod -a -G www-data backuppc

sudo chown backuppc:www-data /etc/BackupPC/
sudo chown backuppc:www-data /etc/BackupPC/BackupPC.users



firefox http://localhost/BackupPC_Admin
```

Vamos a instalar el servicio BackupPC para para realizar copias de seguridad de los distintos hosts.

```
sudo apt install backuppc rsync

sudo htpasswd /etc/backuppc/htpasswd backuppc          # Establecer nuestra propia password


sudo systemctl enable backuppc
sudo systemctl stop backuppc
```

Como estamos en una Raspberry Pi y no queremos hacer los backup de los equipos en una Micro SD de 32Gb, vamos a cambiar el directorio donde se guardan los backup, para ello lo primero sera editar el fichero de configuración y poner el nuevo directorio:

```
sudo vim /etc/backuppc/config.pl

# Añadir

$Conf{TopDir} = '/media/HDD/backuppc';
$Conf{LogDir} = '/media/HDD/backuppc/log';
```

Una vez que tenemos indicado el nuevo directorio copiamos todos los directorios y ficheros que habían en el directorio por defecto al nuevo:

```
sudo cp -r /var/lib/backuppc/* /media/HDD/backuppc/
```

Para evitar poner las IPs vamos a indicar el nombre asociada a cada IP.

```
echo "# personal hosts
192.168.1.71        rp3
192.168.1.72        zero
192.168.1.141       xiaomi
192.168.1.144       4770k
" >> /etc/hosts
```

c

Establecemos los permisos necesarios para el directorio. Este paso esta enlazado con el de Transmission, ya que ahí es donde se crea el grupo *hdd*.

```
sudo usermod -a -G hdd backuppc
sudo chown backuppc:backuppc /media/HDD/backuppc/ -R
```

Para poder usar rsync en Linux sin necesidad de poner contraseñas es necesario generar un certificado SSH sin contraseña (o usar uno ya disponible como el que tengo para servicios). Este certificado tiene que estar en el directorio de backuppc, por lo que es necesario logearnos con este para generar/usar el certificado.

Puede fallar al copiar los certificados en caso de que el fichero de SSH no tenga puesto *permitRootLogin yes*.

```
sudo su - backuppc          # login user backuppc
ssh-copy-id root@4770k      # envia el certificado id_rsa.pub, sino se llama asi renombrarlo
ssh-copy-id root@xiaomi
ssh-copy-id root@zero

# login a servers para anadir los fingerprint
/usr/bin/ssh -q -x -l root rp4
/usr/bin/ssh -q -x -l root xiaomi
/usr/bin/ssh -q -x -l root zero
```

Después seria necesario loguearse a cada maquina para que se quede registrado el fingerprinter de cada una y así no lo vuelva a pedir.

Una vez hecho esto, la configuración de SSL por parte de apache seria la siguiente:

```
sudo mkdir -p /etc/apache2/ssl/            # Creamos directorio donde guardar certificados

# Enviamos los certificados
scp procamora@4770k:/home/procamora/Documents/CA/server/backuppc.key /etc/apache2/ssl/
scp procamora@4770k:/home/procamora/Documents/CA/server/backuppc.cert /etc/apache2/ssl/

scp procamora@4770k:/home/procamora/Documents/CA/ca/cacert.cert /etc/apache2/ssl/

# Cogemos la plantilla de ssl y la modificamos
sudo cp /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-available/backuppc.conf
```

Editamos el fichero que acabamos de crear añadiéndole la ruta de los certificados, el DocumentRoot y ServerName.

```
sudo vim /etc/apache2/sites-available/backuppc.conf
```

```
<IfModule mod_ssl.c>
        <VirtualHost _default_:443>
            ServerAdmin webmaster@localhost
            ServerName backuppc.procamora.com
            ServerAlias backuppc.procamora.com

            DocumentRoot /usr/share/backuppc/cgi-bin/
```

```
                # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
                #LogLevel info ssl:warn

                ErrorLog ${APACHE_LOG_DIR}/error.log
                CustomLog ${APACHE_LOG_DIR}/access.log combined

                #   SSL Engine Switch:
                #   Enable/Disable SSL for this virtual host.
                SSLEngine on
                SSLCertificateFile      /etc/apache2/ssl/backuppc.cert
                SSLCertificateKeyFile   /etc/apache2/ssl/backuppc.key

                #SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt
                #SSLCACertificatePath /etc/ssl/certs/
                SSLCACertificateFile /etc/apache2/ssl/cacert.cert
                #SSLCARevocationPath /etc/apache2/ssl.crl/
                #SSLCARevocationFile /etc/apache2/ssl.crl/ca-bundle.crl
                #SSLVerifyClient require
                #SSLVerifyDepth  10
                #SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
                <FilesMatch "\.(cgi|shtml|phtml|php)$">
                                SSLOptions +StdEnvVars
                </FilesMatch>
                <Directory /usr/lib/cgi-bin>
                                SSLOptions +StdEnvVars
                </Directory>
        </VirtualHost>
</IfModule>
```

Una vez que están configurado el fichero, lo que faltara sería: habilitar el modulo de SSL, habilitar el VirtualHost que hemos creado y reiniciar la configuración de Apache:

```
sudo a2enmod ssl
sudo a2ensite backuppc.conf
sudo systemctl reload apache2
```

Por ultimo, para poder hacer backup locales es necesario que el usuario backuppc pueda ejecutar el comando tar como sudo son contraseña:

```
sudo echo "backuppc ALL=NOPASSWD: /bin/tar" > /etc/sudoers.d/backuppc
```

Una vez realizada toda esta configuración, ya solo quedaría conectarnos por el navegador a la url: http://192.168.1.71/backuppc/ o a https://backuppc.procamora.com.

En la sección host tendremos que añadir todos los host a los que queremos realizar copias de seguridad y en la sección xfer pondremos que el protocolo por defecto es rsync y los directorios a los que queremos realizar copias de seguridad. También podemos excluir ficheros como (*.iso, *.class, *.pyc, etc). Dentro de cada host podemos sobrescribir estas reglas poniendo algunas mas especificas.

```
http://backuppc.sourceforge.net/faq/localhost.html
http://backuppc.sourceforge.net/faq/ssh.html
https://backuppc.procamora.com/backuppc/index.cgi?action=view&type=docs#_conf_xferloglevel_SSH
```

> Recuperar fichero comprimido por backuppc sudo perl -MCompress::Zlib -e 'undef $/; print uncompress(<>)' < /media/HDD/backuppc/pc/rp3/119/f%2fetc%2f/fbackuppc/fapache.conf

d

sudo mkdir /lib/systemd/system/mio sudo touch /lib/systemd/system/mio/client_ssh.service sudo chmod 644 /lib/systemd/system/mio/client_ssh.service