RPI3 download box

Image download

https://www.raspberrypi.com/documentation/computers/getting-started.html#installing-the-operating-system

Direct image of RaspiOS Lite
https://downloads.raspberrypi.org/raspios_lite_armhf/images/raspios_lite_armhf-2021-11-08/2021-10-30-raspios-bullseye-armhf-lite.zip

RPI Imager on a debian based system:

```
sudo apt-get install qml-module-qtquick2 qml-module-qtquick-controls2 qml-module-qtquick-layouts qml-module-qtquick-templates2 qml-module-qtquick-window2 qml-module-qtgraphicaleffects libqt5quickcontrols2-5 libqt5quicktemplates2-5
wget https://downloads.raspberrypi.org/imager/imager_latest_amd64.deb
sudo dpkg -i imager_*_amd64.deb
```

Flash Micro-SD card using Imager selecting ROS lite (without desktop and apps) OS, press Ctrl-Shift-X for advanced menu and configure ssh, passwords, hostname.

set static IP

```
sudo vi /etc/dhcpcd.conf
```

```
interface eth0
static ip_address=192.168.0.100/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1 8.8.8.8

interface wlan0
static ip_address=192.168.0.100/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1 8.8.8.8
```

if/when adjusting hostname in /etc/hostname, follow up and change /etc/hosts with the new hostname as well:

```
127.0.0.1 newhostname
```

Transmission

```
sudo apt install transmission-daemon
```

then https://pimylifeup.com/raspberry-pi-transmission/

Prowlarr install:

https://wiki.servarr.com/prowlarr/installation

```
cat >> ProwlarrInstall.sh << "_EOF_"
#!/bin/bash
### Description: Prowlarr Debian install
### Originally from the Radarr Community

set -euo pipefail

# Am I root?, need root!
if [ "$EUID" -ne 0 ]; then
    echo "Please run as root."
    exit
fi
# Const
### Update these variables as required for your specific instance
app="prowlarr"               # App Name
app_uid="prowlarr"           # {Update me if needed} User App will run as
and the owner of it's binaries
app_guid="prowlarr"          # {Update me if needed} Group App will run as.
app_port="9696"              # Default App Port; Modify config.xml after
install if needed
app_prereq="curl sqlite3"    # Required packages
app_umask="0002"             # UMask the Service will run as
app_bin=${app^}              # Binary Name of the app
installdir="/opt"            # {Update me if needed} Install Location
bindir="${installdir}/${app^}" # Full Path to Install Location
branch="develop"             # {Update me if needed} branch to install
datadir="/var/lib/prowlarr/" # {Update me if needed} AppData directory to
use

# Create User / Group as needed
if ! getent group "$app_guid" >/dev/null; then
    groupadd "$app_guid"
    echo "Group [$app_guid] created"
fi
if ! getent passwd "$app_uid" >/dev/null; then
    useradd --system -g "$app_guid" "$app_uid"
    echo "User [$app_uid] created and added to Group [$app_guid]"
else
    echo "User [$app_uid] already exists"
fi

if ! getent group "$app_guid" | grep -qw "${app_uid}"; then
    echo "User [$app_uid] did not exist in Group [$app_guid]"
```

```
    usermod -a -G "$app_guid" "$app_uid"
    echo "Added User [$app_uid] to Group [$app_guid]"
else
    echo "User [$app_uid] already exists in Group [$app_guid]"
fi

# Stop the App if running
if service --status-all | grep -Fq "$app"; then
    systemctl stop $app
    systemctl disable $app.service
fi
# Create Appdata Directory
# AppData
mkdir -p $datadir
chown -R $app_uid:$app_uid $datadir
chmod 775 $datadir
# Download and install the App
# prerequisite packages
apt install $app_prereq
ARCH=$(dpkg --print-architecture)
# get arch
dlbase="https://$app.servarr.com/v1/update/$branch/updatefile?os=linux&runti
me=netcore"
case "$ARCH" in
"amd64") DLURL="${dlbase}&arch=x64" ;;
"armhf") DLURL="${dlbase}&arch=arm" ;;
"arm64") DLURL="${dlbase}&arch=arm64" ;;
*)
    echo_error "Arch not supported"
    exit 1
    ;;
esac
echo "Downloading..."
wget --content-disposition "$DLURL"
tar -xvzf ${app^}.*.tar.gz
echo "Installation files downloaded and extracted"
# remove existing installs
echo "Removing existing installation"
rm -rf $bindir
echo "Installing..."
mv "${app^}" $installdir
chown $app_uid:$app_uid -R $bindir
rm -rf "${app^}.*.tar.gz"
# Ensure we check for an update in case user installs older version or
different branch
touch $datadir/update_required
chown $app_uid:$app_guid $datadir/update_required
echo "App Installed"
# Configure Autostart
# Remove any previous app .service
echo "Removing old service file"
```

```
rm -rf /etc/systemd/system/$app.service
# Create app .service with correct user startup
echo "Creating service file"
cat <<EOF | tee /etc/systemd/system/$app.service >/dev/null
[Unit]
Description=${app^} Daemon
After=syslog.target network.target
[Service]
User=$app_uid
Group=$app_guid
UMask=$app_umask
Type=simple
ExecStart=$bindir/$app_bin -nobrowser -data=$datadir
TimeoutStopSec=20
KillMode=process
Restart=on-failure
[Install]
WantedBy=multi-user.target
EOF
# Start the App
echo "Service file created. Attempting to start the app"
systemctl -q daemon-reload
systemctl enable --now -q "$app"
# Finish Update/Installation
host=$(hostname -I)
ip_local=$(grep -oP '^\S*' <<<"$host")
echo ""
echo "Install complete"
echo "Browse to http://$ip_local:$app_port for the ${app^} GUI"
# Exit
exit 0
_EOF_

chmod 755 ProwlarrInstall.sh
sudo ./ProwlarrInstall.sh
```

Browse to http://IP:9696 for the Prowlarr GUI

Prowlarr fetches updated index definitions automatically. However, custom definitions can be added. Create Custom directory in either ~/.config/Prowlarr/Definitions/Custom or /var/lib/prowlarr/Definitions/Custom Any custom yml files go in this directly and need to have appropriate permissions and ownership for the prowlarr user to read them. Prowlarr caches the definitions, so while working with them, either wait for the cache to time out or restart Prowlarr after each change. https://wiki.servarr.com/prowlarr/indexers

Sonarr install:

https://sonarr.tv/#downloads-v3-linux-debian

On RPI3, disable h265.x265,HEVC as RPI does not have hardware decoding for these: settings → profiles → under release profiles click + , give it a name (no HEVC/265) and add under “must not contain” x265 h265 hevc separated by spaces and click save.

To exclude 4k/raw content, go to settings → profiles and edit the Any profile, remove all -2160p, Remux, Raw-HD and unknown entries.

Radarr install:

https://wiki.servarr.com/radarr/installation

On RPI3, disable h265.x265,HEVC as RPI does not have hardware decoding for these: settings → indexers → under restrictions profiles click + , add under “must not contain” x265 h265 hevc separated by spaces and click save.

To exclude 4k/raw and low quality content, go to settings → profiles and edit the Any profile, remove all -2160p, Remux, Raw-HD and BR-Disk entries as well as Telecine, Telesync, Cam, Workprint and unknown.

Kodi on RPI:

https://forums.raspberrypi.com/viewtopic.php?t=251645

Kodi on the Raspberry Pi 0/1/2/3 will only function if you are using the Broadcom drivers! This is the “Original non-GL desktop driver” on raspi-config, which is currently the default on the Raspbian images for the RPi 0/1/2/3. If you select the open-source OpenGL driver, it won't work!

Kodi on Raspbian requires a minimum of 160 MB of RAM dedicated to the GPU to function properly! This can be done by running “raspi-config” → “Advanced Options” → “Memory Split” → 160. If you have a RPi 2/3, the recommended is 256 MB of RAM for the GPU.

-Kodi 18 on the Raspberry Pi 2/3 supports 10bit video files (at least h264 and h265/HEVC) but they are software decoded. The Pi 3B / 3B+ may do 720p 10bit and 1080p 10bit low bitrate only at max! For that you need at least 300MB of RAM for the GPU.

Raspbian by default, doesn't play some video codecs like VP6, VP8, MJPEG, Theora, etc, so to be able to play this codecs, you need to go to “raspi-config” → “Interfacing Options” → “Camera” → Enable, or just add a new line in /boot/config.txt with:

```
start_x=1
```

on RPI3 add to/enable in /boot/config.txt for DRM to work:

```
[all]
dtoverlay=vc4-kms-v3d
```

```
sudo tee -a /lib/systemd/system/kodi.service <<_EOF_
[Unit]
```

```
Description = Kodi Media Center
After = remote-fs.target network-online.target
Wants = network-online.target

[Service]
User = pi
Group = pi
Type = simple
ExecStart = /usr/bin/kodi-standalone
Restart = on-abort
RestartSec = 5

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

In Kodi enable remote control with user/password kodi Transmission default login user/password transmission

URLs/Ports:

http://IP:7878 radarr http://IP:8989 sonarr http://IP:9091 transmission (username/password: transmission) http://IP:9696 prowlarr http://IP:8080 kodi (username/password: kodi)