

Octoprint on Linux

Change everything in **RED** to your username. Everything in **BOLD** copy and run in the terminal, everything in *italics*, copy and paste into config files.

Find the IP address of the server:

ifconfig -a

Update your server software to the newest version and reboot:

sudo apt update

sudo apt upgrade

sudo reboot now

cd ~

sudo apt install python-pip python-dev python-setuptools python-virtualenv git

libyaml-dev build-essential

mkdir OctoPrint && cd OctoPrint

virtualenv venv

source venv/bin/activate

pip install pip --upgrade

pip install <https://get.octoprint.org/latest>

sudo usermod -a -G tty **chris**

sudo usermod -a -G dialout **chris**

~/OctoPrint/venv/bin/octoprint serve

Now test to make sure you can get to octoprint on the server, use your IP in a browser:

<http://192.168.1.5:5000>

Now we need to make it so Octoprint starts up after reboot:

wget https://github.com/foosel/OctoPrint/raw/master/scripts/octoprint.init && sudo mv octoprint.init /etc/init.d/octoprint

wget https://github.com/foosel/OctoPrint/raw/master/scripts/octoprint.default && sudo mv octoprint.default /etc/default/octoprint

sudo chmod +x /etc/init.d/octoprint

Edit the user and remove comments in the default file:

sudo nano /etc/default/octoprint

Configuration for /etc/init.d/octoprint

The init.d script will only run if this variable non-empty.

OCTOPRINT_USER=chris

base directory to use

BASEDIR=/home/chris/.octoprint

configuration file to use

CONFIGFILE=/home/chris/.octoprint/config.yaml

On what port to run daemon, default is 5000

PORT=5000

Path to the OctoPrint executable, you need to set this to match your installation!

DAEMON=/home/chris/OctoPrint/venv/bin/octoprint

What arguments to pass to octoprint, usually no need to touch this

DAEMON_ARGS="--port=\$PORT"

Umask of files octoprint generates, Change this to 000 if running octoprint as its own, separate user

UMASK=022

Process priority, 0 here will result in a priority 20 process.

-2 ensures Octoprint has a slight priority over user processes.

NICELEVEL=-2

Should we run at startup?

START=yes

Add default to autostart:

sudo update-rc.d octoprint defaults

Check octoprint service status with:

sudo service octoprint status

Add your users to sudoers file so it can run shutdown commands:

sudo nano /etc/sudoers.d/octoprint-shutdown

chris ALL=NOPASSWD: /sbin/shutdown

Update octoprint service as well:

sudo nano /etc/sudoers.d/octoprint-service

chris ALL=NOPASSWD: /usr/sbin/service

Install haproxy:

sudo apt install haproxy

Make a copy of the haproxy config file and remove it:

sudo cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg_old

sudo rm /etc/haproxy/haproxy.cfg

Paste this config into the haproxy file:

sudo nano /etc/haproxy/haproxy.cfg

global

maxconn 4096

user haproxy

group haproxy

daemon

log 127.0.0.1 local0 debug

defaults

log global

mode http

option httplog

option dontlognull

retries 3

option redispatch

option http-server-close

option forwardfor

maxconn 2000

timeout connect 5s

timeout client 15min

timeout server 15min

frontend public

bind :::80 v4v6

use_backend webcam if { path_beg /webcam/ }

default_backend octoprint

backend octoprint

reqrep ^([^\:])\/(.*) \1\^2*

option forwardfor

server octoprint1 127.0.0.1:5000

backend webcam

```
reqrep ^([^\:]*)\ /webcam/(.*) \1\ /2  
server webcam1 127.0.0.1:8080
```

Enable haproxy:

sudo nano /etc/default/haproxy

Defaults file for HAProxy

#

*# This is sourced by both, the initscript and the systemd unit file, so do not
treat it as a shell script fragment.*

Change the config file location if needed

#CONFIG="/etc/haproxy/haproxy.cfg"

Add extra flags here, see haproxy(1) for a few options

#EXTRAOPTS="-de -m 16"

ENABLE=1

Check haproxy:

sudo service haproxy status

Restart haproxy just in case:

sudo service haproxy restart

Now we install webcam support:

cd ~

sudo apt install subversion libjpeg8-dev imagemagick ffmpeg libv4l-dev cmake

git clone https://github.com/jacksonliam/mjpg-streamer.git

cd mjpg-streamer/mjpg-streamer-experimental

export LD_LIBRARY_PATH=.

make

Test mjpg streamer:

sudo ./mjpg_streamer -i "./input_uvc.so" -o "./output_http.so"

Create webcam startup scripts don't use sudo:

cd ~

mkdir scripts

nano /home/chris**/scripts/webcam**

```
#!/bin/bash
# Start / stop streamer daemon

case "$1" in
    start)
        /home/chris/scripts/webcamDaemon >/dev/null 2>&1 &
        echo "$0: started"
        ;;
    stop)
        pkill -x webcamDaemon
        pkill -x mjpg_streamer
        echo "$0: stopped"
        ;;
    *)
        echo "Usage: $0 {start|stop}" >&2
        ;;
esac
```

Create webcam Daemon script don't use sudo:
nano /home/chris**/scripts/webcamDaemon**

```
#!/bin/bash

MJPEGSTREAMER_HOME=/home/chris/mjpg-streamer/mjpg-streamer-experimental
MJPEGSTREAMER_INPUT_USB="input_uvc.so"
MJPEGSTREAMER_INPUT_RASPICAM="input_raspicam.so"

# init configuration
camera="auto"
camera_usb_options="-r 640x480 -f 10"
camera_raspi_options="-fps 10"

if [ -e "/boot/octopi.txt" ]; then
    source "/boot/octopi.txt"
fi

# runs MJPG Streamer, using the provided input plugin + configuration
function runMjpgStreamer {
```

```

    input=$1
    pushd $MJPEGSTREAMER_HOME
    echo Running ./mjpg_streamer -o "output_http.so -w ./www" -i "$input"
    LD_LIBRARY_PATH=. ./mjpg_streamer -o "output_http.so -w ./www" -i "$input"
    popd
}

# starts up the RasPiCam
function startRaspi {
    logger "Starting Raspberry Pi camera"
    runMjpgStreamer "$MJPEGSTREAMER_INPUT_RASPICAM $camera_raspi_options"
}

# starts up the USB webcam
function startUsb {
    logger "Starting USB webcam"
    runMjpgStreamer "$MJPEGSTREAMER_INPUT_USB $camera_usb_options"
}

# we need this to prevent the later calls to v4l2ctl from blocking
# I have no idea why, but that's how it is...
v4l2ctl version

# echo configuration
echo camera: $camera
echo usb options: $camera_usb_options
echo raspi options: $camera_raspi_options

# keep mjpg streamer running if some camera is attached
while true; do
    if [ -e "/dev/video0" ] && { [ "$camera" = "auto" ] || [ "$camera" = "usb" ] ; }; then
        startUsb
    elif [ "`v4l2ctl get_camera`" = "supported=1 detected=1" ] && { [ "$camera" = "auto" ] || [
"$camera" = "raspi" ] ; }; then
        startRaspi
    fi

    sleep 120
done

```

Edit webcam file startup permissions:

```

sudo chmod +x /home/chris/scripts/webcam
sudo chmod +x /home/chris/scripts/webcamDaemon

```

Add webcams to startup:

sudo nano /etc/rc.local

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
```

```
/home/chris/scripts/webcam start
```

```
exit 0
```

Start webcam:

sudo /home/chris/scripts/webcam start

Change rc.local permissions:

sudo chmod +x /etc/rc.local

Install avahi:

sudo apt install avahi-daemon

Edit hostname file with server name:

sudo nano /etc/hostname

Edit hosts:

sudo nano /etc/hosts

```
127.0.0.1    localhost.localdomain localhost
::1         localhost6.localdomain6 localhost6
```

```
# The following lines are desirable for IPv6 capable hosts
```

```
::1         localhost ip6-localhost ip6-loopback
```

fe00::0 ip6-localnet
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

127.0.1.1 octolinux

Now reboot:

sudo reboot now

Shutdown commands:

Restart OctoPrint: sudo service octoprint restart

Restart system: sudo shutdown -r now

Shutdown system: sudo shutdown -h now

Webcam links:

Stream URL: /webcam/?action=stream

Snapshot URL: <http://127.0.0.1:8080/?action=snapshot>

Path to FFMPEG: /usr/bin/ffmpeg

Install linux desktop if you would like one:

sudo apt-get install ubuntu-desktop