

Containerized Linux Print Server using Octoprint - the long way



The long way is for if you care about the technical steps and want to reproduce it yourself. See the [short way to just get the LXC Octoprint image](#) and get up and running as easy as possible.

The primary purpose of the article is to bookmark and reference my efforts to get Octoprint as opposed to Octopi, up and running on a repurposed redundant HP Laptop.

It might not be obvious why we want to do all of this, so let me start by explaining why all this effort was undertaken. So typically we would run Octopi instead, so Octoprint on a raspberry Pi. The issue with this is that due to chip shortages Pi's are now scarce and very expensive and quite frankly, mine has become slow and frustrating, so running this on a Linux PC is just orders of magnitude faster and more reliable, the issue is that Octoprint only supports one printer at a time, so if we can containerize them, we can keep on expanding until we run out of USB ports, or hardware resources, RAM, CPU and disk space. One instance of Octoprint on my 8 year old HP-Pavilion executes gcode processing about 100 times faster than my Pi 3B and the webcams just never glitch or hang, not to mention print fails because a Pi overheated or stop communicating.

This should be straightforward, how hard can it be, right. I thought so too, but days later I realized that this has to be revisited and documented as I probably will not remember most of what was done by the next time I have to repeat this process.

The basic steps to complete the process were as follow:

- Setup The Base Linux OS on the Laptop
- Install and configure LXD/LXC(Linux Containers)
- Pass Through and set up the printer connection
- Pass through and set up the webcam connection
- Start up and set up Octoprint in the container

Setup The Base Linux OS on the Laptop

- Download Ubuntu 22.04 LTS ¹⁾
- Create a bootable ISO Using Rufus ^{2) 3)}
- Boot from the ISO and install Ubuntu
- Create a Bridge for LXC(Linux Containers)
- Create a partition on Linux that LXD can use as a storage pool

Create a Bridge for LXC(Linux Containers)

The default config typically looks like this:

```
root@hp-linux:~# nano /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  ethernets:
    eno1:
      dhcp4: true
  version: 2
```

Amend as follow to add a bridge called br0 to the config.

```
root@hp-linux:~# cat /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  ethernets:
    eno1: {}
  bridges:
    br0:
      dhcp4: true
      interfaces:
        - eno1
      #gateway4: 192.168.0.1
  version: 2
```

Adding a bridge allows lxc to assign network accessible DHC assigned IP addresses to the containers, i.e your container will look like regular machines on the network with similar ip addresses to other systems on the router. That will allow you to connect to octoprint using the ip or host name.



We will need br0 in the following steps setting up LXD

Create a partition on Linux that LXD can use as a storage pool

This step basically aims to utilize the remaining disk space that was not allocated by the ubuntu install. The HP system happens to have more that 1TiB of storage and the Linux base OS install only allocated 98GiB by default.

Lots of googling and many articles later ⁴⁾ ⁵⁾ I started with a few commands to asses what is available and how to partition that for use by LXD.

```
root@hp-linux:~# fdisk -l /dev/sda
Disk /dev/sda: 1.82 TiB, 2000398934016 bytes, 3907029168 sectors
Disk model: ST2000LM003 HN-M
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disklabel type: gpt
Disk identifier: 1D2EE2A6-79C5-4DBD-B799-42B48F45F62E
```

Device	Start	End	Sectors	Size	Type
/dev/sda1	2048	2203647	2201600	1G	EFI System
/dev/sda2	2203648	6397951	4194304	2G	Linux filesystem
/dev/sda3	6397952	3907026943	3900628992	1.8T	Linux filesystem

Command	Description
pvs	Display information about physical volumes
lvs	Display information about logical volumes
vgs	Display information about volume groups
pvdisplay	Display various attributes of physical volume(s)
lvdisplay	Display information about a logical volume
vgdisplay	Display volume group information
lvmdiskscan	List devices that may be used as physical volumes

Using the commands above to discover what space is available to assign to the lxc storage pool.

```

root@hp-linux:~# pvs
  PV          VG          Fmt Attr PSize  PFree
  /dev/sda3  ubuntu-vg  lvm2 a--  <1.82t  735.96g
root@hp-linux:~# vgs
  VG          #PV #LV #SN Attr   VSize  VFree
  ubuntu-vg    1  2   0 wz--n-  <1.82t  735.96g
root@hp-linux:~# lvs
  LV          VG          Attr      LSize   Pool Origin Data%  Meta%  Move Log
Cpy%Sync Convert
  lv-lxc      ubuntu-vg  -wi-ao---- 1.00t
  ubuntu-lv   ubuntu-vg  -wi-ao---- 100.00g
root@hp-linux:~# pvdisplay
--- Physical volume ---
PV Name           /dev/sda3
VG Name           ubuntu-vg
PV Size           <1.82 TiB / not usable 4.00 MiB
Allocatable       yes
PE Size           4.00 MiB
Total PE          476150
Free PE           188406
Allocated PE      287744
PV UUID           rvz2ag-hDP0-Yb9l-eCr5-P0lK-vslK-Le7W4u

root@hp-linux:~# vgdisplay
--- Volume group ---
VG Name           ubuntu-vg
System ID
Format            lvm2
Metadata Areas    1
Metadata Sequence No 3
VG Access         read/write
VG Status         resizable
MAX LV            0
Cur LV           2

```

```
Open LV          2
Max PV           0
Cur PV          1
Act PV           1
VG Size          <1.82 TiB
PE Size          4.00 MiB
Total PE         476150
Alloc PE / Size  287744 / <1.10 TiB
Free PE / Size   188406 / 735.96 GiB
VG UUID          Dfph05-yeK8-Pkrb-JZh3-TMEU-jLF4-4pn1WY
```

```
root@hp-linux:~# lvsdisplay
```

```
--- Logical volume ---
```

```
LV Path          /dev/ubuntu-vg/ubuntu-lv
LV Name          ubuntu-lv
VG Name          ubuntu-vg
LV UUID          lupg0H-eV6M-l5YN-UjNV-j75K-P55e-a7hkeg
LV Write Access   read/write
LV Creation host, time ubuntu-server, 2023-05-12 10:52:34 +0000
LV Status         available
# open           1
LV Size          100.00 GiB
Current LE        25600
Segments          1
Allocation         inherit
Read ahead sectors auto
- currently set to 256
Block device      253:0
```

```
--- Logical volume ---
```

```
LV Path          /dev/ubuntu-vg/lv-lxc
LV Name          lv-lxc
VG Name          ubuntu-vg
LV UUID          b9oWe9-UBQg-yQ3R-Io3F-XcoY-lUm3-cy8GtK
LV Write Access   read/write
LV Creation host, time hp-linux, 2023-05-12 12:24:45 +0000
LV Status         available
# open           1
LV Size          1.00 TiB
Current LE        262144
Segments          1
Allocation         inherit
Read ahead sectors auto
- currently set to 256
Block device      253:1
```



The two commands used to set up the spare disk space was:

```
root@hp-linux:~# lvcreate -L 1T -n lv-lxc ubuntu-vg
root@hp-linux:~# mkfs.ext4 /dev/ubuntu-vg/lv-lxc
```



The first command creates a local volume of 1TiB names lv-lxc in the ubuntu-vg.

The second command creates an ext4 filesystem on the volume.

Using df and lsblk to initially inspect what we can see:

```
root@hp-linux:~# df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
tmpfs	779M	1.7M	778M	1%	/run
/dev/mapper/ubuntu--vg-ubuntu--lv	98G	12G	82G	13%	/
tmpfs	3.9G	0	3.9G	0%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
/dev/sda2	2.0G	272M	1.6G	15%	/boot
/dev/sda1	1.1G	6.1M	1.1G	1%	/boot/efi
tmpfs	1.0M	0	1.0M	0%	
/var/snap/lxd/common/ns					
tmpfs	779M	4.0K	779M	1%	/run/user/0

No new mounted filesystem, but in the block devices we now have ubuntu-vg-lv-lxc with 1TiB of space assigned.

```
root@hp-linux:~# lsblk
```

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINTS
loop0	7:0	0	63.3M	1	loop	/snap/core20/1879
loop1	7:1	0	73M	1	loop	/snap/core22/617
loop2	7:2	0	111.9M	1	loop	/snap/lxd/24322
loop3	7:3	0	53.2M	1	loop	/snap/snapd/19122
loop4	7:4	0	73.1M	1	loop	/snap/core22/634
loop5	7:5	0	55.6M	1	loop	/snap/core18/2745
loop6	7:6	0	108.5M	1	loop	/snap/lxdmosaic/247
loop7	7:7	0	63.5M	1	loop	/snap/core20/1891
sda	8:0	0	1.8T	0	disk	
└sda1	8:1	0	1G	0	part	/boot/efi
└sda2	8:2	0	2G	0	part	/boot
└sda3	8:3	0	1.8T	0	part	
└└ubuntu--vg-ubuntu--lv	253:0	0	100G	0	lvm	/
└└└ubuntu--vg-lv--lxc	253:1	0	1T	0	lvm	
sr0	11:0	1	1024M	0	rom	

```
root@hp-linux:~# lvs
```

LV	VG	Attr	LSize	Pool	Origin	Data%	Meta%	Move	Log
Cpy%Sync	Convert								
lv-lxc	ubuntu-vg	-wi-ao----	1.00t						

```
ubuntu-lv ubuntu-vg -wi-ao---- 100.00g
```

We can also look at the mountpoints for the block devices like this:

```
root@hp-linux:~# lsblk -o NAME,FSTYPE,LABEL,SIZE,MOUNTPOINT
NAME                                FSTYPE      LABEL                                SIZE
MOUNTPOINT
loop0                               squashfs     63.3M
/snap/core20/1879
loop1                               squashfs     73M
/snap/core22/617
loop2                               squashfs     111.9M
/snap/lxd/24322
loop3                               squashfs     53.2M
/snap/snapd/19122
loop4                               squashfs     73.1M
/snap/core22/634
loop5                               squashfs     55.6M
/snap/core18/2745
loop6                               squashfs     108.5M
/snap/lxdmosaic/247
loop7                               squashfs     63.5M
/snap/core20/1891
sda                                 1.8T
├sda1                              vfat         1G
/boot/efi
├sda2                              ext4         2G /boot
└sda3                              LVM2_member  1.8T
  └ubuntu--vg-ubuntu--lv          ext4         100G /
    └ubuntu--vg-lv--lxc           zfs_member   lxd_storage_pool_default 1T
sr0                                 1024M
```



lvdisplay gives us the path to the storage pool required for the next steps in LV Path that point to /dev/ubuntu-vg/lv-lxc

Install and configure LXD/LXC(Linux Containers)

- Install the latest stable version of LXD
- Initialise LXD
 - Specify the Bridge for Use that was set up in the OS setup section
 - Point the storage pool to the partition created previously
- Create and launch a container for octoprint

Install the latest stable version of LXD

At the time of writing this, see file date below right, LXD 5.0 was the stable release candidate, we want to make sure to run that. My distribution uses snap, so using snap to check versions installed you can run `snap list`.

```
root@hp-linux:~# snap list
Name      Version      Rev    Tracking      Publisher    Notes
core18    20230426     2745   latest/stable canonical✓    base
core20    20230503     1891   latest/stable canonical✓    base
core22    20230503     634    latest/stable canonical✓    base
lxd       5.0.2-838e1b2 24322  5.0/stable    canonical✓    -
lxdmosaic 0+git.c6f53f3f 247    latest/stable turtle0x1     -
snapd     2.59.2       19122  latest/stable canonical✓    snapd
```

If you don't have a stable candidate, it can be removed and reinstalled as follows:

```
root@hp-linux:~# snap remove lxd
root@hp-linux:~# snap install lxd --channel=5.0/stable
```

Initialise LXD

The next step is to edit configurations and view logs, you need to get this set up.

SSH comes bundled with Linux and is already set up and installed, we just need some minor edits to allow users to log in using password authentication. The Linux preferred way is to create RSA tokens and it's a great idea, just beyond the scope of this document. ⁶⁾

We are just going to allow a password authentication for now:

We need to:

- edit the ssh daemon configuration first
- restart the service

We can do it from our host Linux server OS directly on the container like this:

```
dev@hp-linux:~$ lxc exec octo -- bash -c "nano /etc/ssh/sshd_config"
```

That opens the `sshd_config` file on the container and allows us to edit it locally.

- find and uncomment/edit `PasswordAuthentication no` to `yes`

```
# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication no
#PermitEmptyPasswords no
```

```
# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication no
PasswordAuthentication yes
```

```
#PermitEmptyPasswords no
```



Then remember to restart sshd

```
dev@hp-linux:~$ lxc exec octo -- bash -c "systemctl restart sshd"
```

Now test ssh connectivity

```
dev@hp-linux:~$ ssh octo@octo
octo@cr6's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-71-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Thu May 18 09:37:40 UTC 2023

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how
MicroK8s
  just raised the bar for easy, resilient and secure K8s cluster
deployment.

  https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

13 updates can be applied immediately.
5 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

9 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Thu May 18 09:34:52 2023 from 192.168.0.32
octo@octo:~$
```

We are now connected as the octo user on the oct instance of lxc. I use PuTTY⁷⁾ to connect from my Windows PC into a Linux shell session to work on the backends of the Linux headless systems.

Start up and set up Octprint in the container

The last section follows the instruction from the Octoprint community forum for [published here](#)⁸⁾

There is a current issue with some of the key plugins I need to work, i.e. Filament manager and Spool manager that causes failures running under Python 3.10 on the Linux setup. Forcing the install to Python 3.9 in the virtual environment creation, resolves the issues.

Set up a Python 3.9 virtual environment (venv)

- add repositories to get access to 3.9
- install 3.9

```
octo@octo:~# sudo add-apt-repository universe           #add universe as a
repo option
octo@octo:~# sudo apt update                           #and update the
repo source list on the instance
octo@octo:~# sudo apt install python3.9                #try to install
3.9, stop here on success
octo@octo:~# sudo add-apt-repository ppa:deadsnakes/ppa #on fail add
deadsnakes
octo@octo:~# sudo apt install python3.9                #now 3.9 should
install
```

- Create and change into the OctoPrint folder
- Create a 3.9 venv(Virtual Environment)
- Activate the new environment for further steps to follow

```
octo@octo:~$ mkdir OctoPrint && cd OctoPrint
octo@octo:~/OctoPrint$ python3.9 -m venv venv
octo@octo:~/OctoPrint$ source venv/bin/activate
(venv) octo@octo:~/OctoPrint$
```

Note the last line showing our 3.9 environment to be activated.

Add pip and wheel updates

```
(venv) octo@octo:~/OctoPrint$ pip install --upgrade pip wheel
Requirement already satisfied: pip in ./venv/lib/python3.9/site-packages
(22.0.4)
Collecting pip
  Using cached pip-23.1.2-py3-none-any.whl (2.1 MB)
Collecting wheel
  Using cached wheel-0.40.0-py3-none-any.whl (64 kB)
Installing collected packages: wheel, pip
  Attempting uninstall: pip
    Found existing installation: pip 22.0.4
    Uninstalling pip-22.0.4:
      Successfully uninstalled pip-22.0.4
Successfully installed pip-23.1.2 wheel-0.40.0
```

Install Octoprint

On successful completion, proceed to install octoprint

```
(venv) octo@octo:~/OctoPrint$ pip install octoprint
Collecting octoprint
  Using cached OctoPrint-1.8.7-py2.py3-none-any.whl (3.9 MB)
Collecting OctoPrint-FileCheck>=2021.2.23 (from octoprint)
  Using cached OctoPrint_FileCheck-2021.2.23-py2.py3-none-any.whl (19 kB)
```

Wait for it to run to completion and address any errors that might show up. Hopefully no issues will arise, mine ran through without failures here.

Assign octo user run permissions

Now let us assign two more permissions to our octo user.

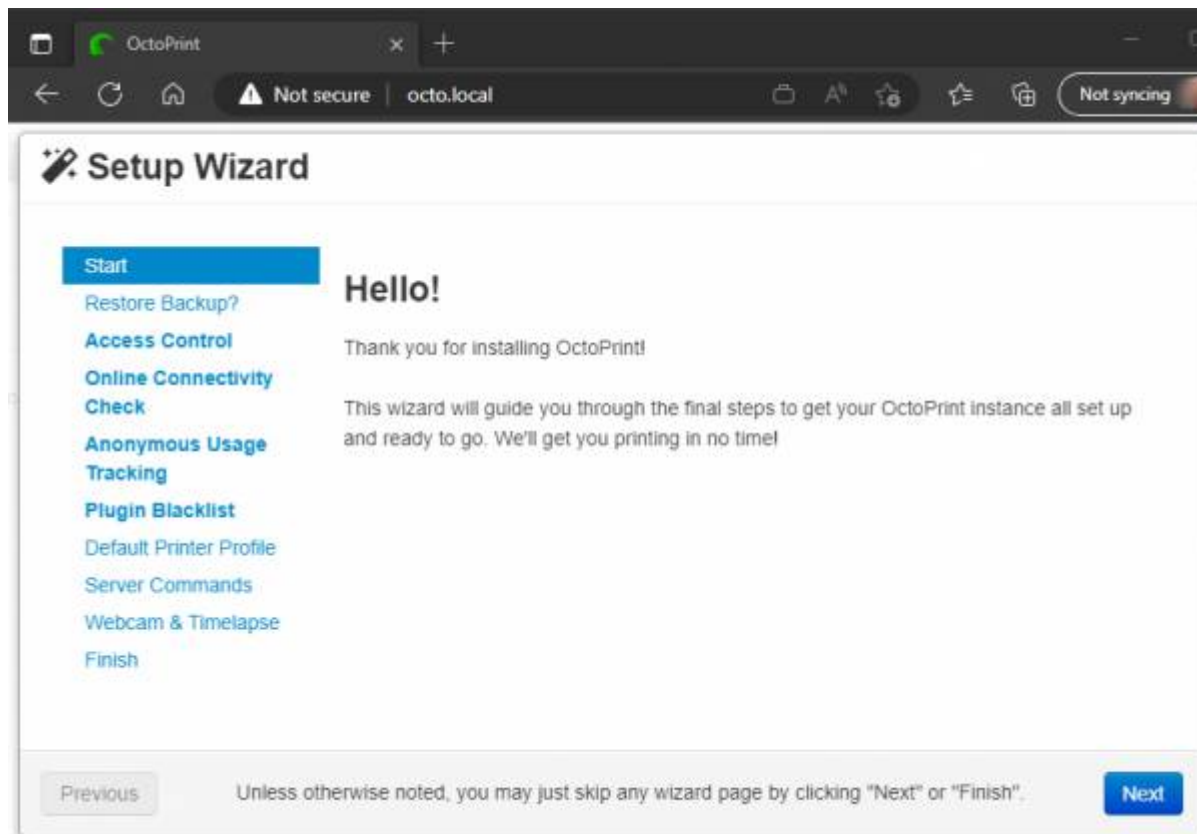
```
(venv) octo@octo:~/OctoPrint$ sudo usermod -aG tty octo
(venv) octo@octo:~/OctoPrint$ sudo usermod -aG dialout octo
```

The commands above add octo to the tty and dialout groups, so octo can run terminal session and connect to our devices like the prinyter port and camera ports.

Do a test run

```
(venv) octo@octo:~/OctoPrint$ ~/OctoPrint/venv/bin/octoprint serve
2023-05-18 12:49:46,585 - octoprint.startup - INFO - 
*****
**
2023-05-18 12:49:46,586 - octoprint.startup - INFO - Starting OctoPrint
1.8.7
2023-05-18 12:49:46,586 - octoprint.startup - INFO - 
*****
**
```

So we have concluded the basics to get a base install for Octoprint working.



Autostart Octoprint as a Linux Service

```
wget
https://github.com/OctoPrint/OctoPrint/raw/master/scripts/octoprint.service
&& sudo mv octoprint.service /etc/systemd/system/octoprint.service
```

Setting up SSH access to your container

SSH access is not a requirement for Octoprint to work, however if you need to access the linux environment backend shell, and run useful linux commands to edit configurations and view logs, you need to get this set up.

SSH comes bundled with Linux and is already set up and installed, we just need some minor edits to allow users to log in using password authentication. The linux preferred way is to create RSA tokens and it's a great idea, just beyond the scope of this document. ⁹⁾

We are just going to allow a password authentication for now:

We need to:

- edit the ssh daemon configuration first
- restart the service

We can do it from our host Linux server OS directly on the container like this:

```
dev@hp-linux:~$ lxc exec octo -- bash -c "nano /etc/ssh/sshd_config"
```

That opens the sshd_config file on the container and allows us to edit it locally.

- find and uncomment/edit PasswordAuthentication no to yes

```
# To disable tunneled clear text passwords, change to no here!  
#PasswordAuthentication no  
#PermitEmptyPasswords no
```

```
# To disable tunneled clear text passwords, change to no here!  
#PasswordAuthentication no  
PasswordAuthentication yes  
#PermitEmptyPasswords no
```



Then remember to restart sshd

```
dev@hp-linux:~$ lxc exec octo -- bash -c "systemctl restart sshd"
```

Now test ssh connectivity

```
dev@hp-linux:~$ ssh octo@octo  
octo@cr6's password:  
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-71-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
System information as of Thu May 18 09:37:40 UTC 2023  
  
* Strictly confined Kubernetes makes edge and IoT secure. Learn how  
MicroK8s  
  just raised the bar for easy, resilient and secure K8s cluster  
deployment.  
  
https://ubuntu.com/engage/secure-kubernetes-at-the-edge  
  
Expanded Security Maintenance for Applications is not enabled.  
  
13 updates can be applied immediately.  
5 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
9 additional security updates can be applied with ESM Apps.  
Learn more about enabling ESM Apps service at https://ubuntu.com/esm
```

```
Last login: Thu May 18 09:34:52 2023 from 192.168.0.32
octo@octo:~$
```

We are now connected as the octo user on the oct instance of lxc. I use PuTTY¹⁰⁾ to connect from my Windows PC into a Linux shell session to work on the backends of the Linux headless systems.

References

1)

<https://ubuntu.com/download/server>

2)

<https://rufus.ie/en/>

3)

https://softwaresupply.net/kb/how-to-create-a-bootable-usb-stick/?gclid=EAlaIQobChMIh_qe98L8_gIVIGt9Ch0FcwL7EAAYASAAEgJUj_D_BwE

4)

<https://linuxopsys.com/topics/check-unallocated-space-linux#:~:text=Unallocated%20space%20means%20that%20the,a%20particular%20drive%20or%20partition.>

5)

<https://askubuntu.com/questions/1029040/how-to-manually-mount-a-partition>

6) , 9)

<https://www.ibm.com/docs/en/sia?topic=kbaula-enabling-rsa-key-based-authentication-unix-linux-operating-systems-3>

7) , 10)

<https://www.putty.org/>

8)

<https://community.octoprint.org/t/setting-up-octoprint-on-a-raspberry-pi-running-raspberry-pi-os-debian/2337>