W23 Advisory #2

Installing and Using LXD on CentOS 7.5 1804 (core), with ZFS

Following are command line instructions (with commentary provided in red text) to install and initialize LXD on a CentOS 7.5 1804 (core) host, with ZFS as a backing store. Additionally, there are instructions to create an Ubuntu 18.04 container with LXD, and similar to our Debian-family examples in Chapter W23, give that container an automatically-assigned IP address via macvlan, on a LAN that the host is connected to.

The instructions are broken into 5 parts:

0. Preliminaries for CentOS 7.5.

I. Installing LXD with snap.

II. Creating a zpool fas an LXD backing store, and initializing LXD to use that zpool.

III. Configuring and launching an Ubuntu 18.04 container with ZFS backing store that gets its IP address from a DHCP server on the LAN the host is connected to.

IV. Further exercises that allow you to install nginx on the container, and ssh from and to it from another computer on your LAN.

*Note: Be aware that the following instructions were all executed on a fresh install of CentOS 7.5 1804 (core). This is similar to Stephane Graber’s installation instructions at the provided web link in Advisory 1.*

**0. Preliminaries for CentOS 7.5**

As done for CentOS in Appendix A of the printed book, the following commands ensure that you can use yum for the installations on the host that follow, and that you can use the epel-release repo where the packages are found-

[bob@localhost]$ **su**  
Password: **zzz**

[root@localhost]# **systemctl stop packagekit**

[root@localhost]# **systemctl disable packagekit**

[root@localhost]# **yum install epel-release**

Output truncated…

**I. Installing LXD with snap.**

The following allow you to install snap-

[root@localhost]# **yum install yum-plugin-copr epel-release**

[root@localhost]# **yum copr enable ngompa/snapcore-el7**

[root@localhost]# **yum install snapd**

[root@localhost]# **systemctl enable --now snapd.socket**

The following allow you to install ZFS-

[root@localhost]# **yum install http://download.zfsonlinux. --disable zfs**

[root@localhost]# **yum-config-manager --enable zfs-kmod**

[root@localhost]# **yum install zfs**

Some kernel configuration modifications are necessary here-

[root@localhost]# **grubby --args="user\_namespace.enable= --update-kernel="$(grubby --default-kernel)"**

[root@localhost]# **grubby --args="namespace.unpriv\_ --update-kernel="$(grubby --default-kernel)"**

[root@localhost]# **echo "user.max\_user\_namespaces= > /etc/sysctl.d/99-userns.conf**

[root@localhost]# **reboot**

Now you’re ready to actually install LXD with snap-

[root@localhost]# **snap install lxd**

**II. Creating a zpool as an LXD backing store, and initializing LXD to use that zpool.**

You need to use a whole disk for the ZFS backing store that is NOT the system disk! In our case we used a USB thumb drive, which is designated in /dev as /dev/sdb.

[root@localhost]# **sudo zpool create -f lxd /dev/sdb**

Next you need to initialize LXD as follows. This will give your subsequently-created containers a ZFS file system on the zpool previously created -

[root@localhost]# **lxd init**  
Would you like to use LXD clustering? (yes/no) [default=no]: **<Enter>**  
Do you want to configure a new storage pool? (yes/no) [default=yes]: **<Enter>**  
Name of the new storage pool [default=default]: **lxd**  
Name of the storage backend to use (btrfs, ceph, dir, lvm, zfs) [default=zfs]: **<Enter>**  
Create a new ZFS pool? (yes/no) [default=yes]: **no**  
Name of the existing ZFS pool or dataset: **lxd**  
Would you like to connect to a MAAS server? (yes/no) [default=no]: **<Enter>**  
Would you like to create a new local network bridge? (yes/no) [default=yes]: **<Enter>**  
What should the new bridge be called? [default=lxdbr0]: **<Enter>**  
What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]: **<Enter>**  
What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]: **<Enter>**  
Would you like LXD to be available over the network? (yes/no) [default=no]: **<Enter>**  
Would you like stale cached images to be updated automatically? (yes/no) [default=yes] **<Enter>**  
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]: **<Enter>**

Finally, make sure you are in the lxd group by using the following command, where you substitute your username for bob-

[root@localhost]# **usermod -a -G lxd bob**

[root@localhost]# **exit**

[bob@localhost]$

**III. Configuring and launching an Ubuntu 18.04 container on the ZFS backing store, and that gets its IP address from a DHCP server on the LAN the host is connected to.**

In this phase of the instructions, we first modify the default lxc profile to include a macvlan virtual network interface, instead of using the traditional bridged network interface approach.

[bob@localhost ~]$ **lxc profile list**

+---------+---------+

| NAME | USED BY |

+---------+---------+

| default | 1 |

This step makes a duplicate copy of the lxc profile.

[bob@localhost ~]$ **lxc profile copy default promac**

[bob@localhost ~]$ **lxc profile list**

+------------+---------+

| NAME | USED BY |

+------------+---------+

| default | 1 |

+------------+---------+

| promac | 0 |

+------------+---------+

[bob@localhost ~]$ **lxc profile show promac**

config: {}

description: Default LXD profile

devices:

eth0:

name: eth0

nictype: bridged

parent: lxdbr0

type: nic

root:

path: /

pool: lxd

type: disk

name: promac

used\_by: []

This step shows us the nic type and parent, or host, interface name.

[bob@localhost ~]$ **ip route show default 0.0.0.0/0**

default via 192.168.0.1 dev em1 proto dhcp metric 100

This step allows us to change the nictype and parent fields of the copied profile.

[bob@localhost ~]$ **lxc profile device set promac eth0 nictype macvlan**

[bob@localhost ~]$ **lxc profile device set promac eth0 parent em1**

[bob@localhost ~]$ **lxc profile show promac**

config: {}

description: Default LXD profile

devices:

eth0:

name: eth0

nictype: macvlan

parent: em1

type: nic

root:

path: /

pool: lxd

type: disk

name: promac

used\_by: []

Now the critical step, launch an Ubuntu 18.04 container with a ZFS backing store (since that was how we initialized lxd), and the macvlan interface named promac. We named the container containera.

[bob@localhost ~]$ **lxc launch -p promac ubuntu:18.04 containera**

Output truncated...

Creating containera

Starting containera

[bob@localhost ~]$ **lxc exec containera ip route**

default via 192.168.0.1 dev eth0 proto dhcp src 192.168.0.33 metric 100

192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.33

192.168.0.1 dev eth0 proto dhcp scope link src 192.168.0.33 metric 100

Now check the network connection from the container to another host on the network. Give an lxc exec command on the host that executes the ping command on the container containera, and that pings another computer on the network with an IP address of 192.168.0.6.

[bob@localhost ~]$ **lxc exec containera -- ping -c 3 192.168.0.6**

PING 192.168.0.6 (192.168.0.6) 56(84) bytes of data.

64 bytes from 192.168.0.6: icmp\_seq=1 ttl=64 time=0.550 ms

64 bytes from 192.168.0.6: icmp\_seq=2 ttl=64 time=0.305 ms

64 bytes from 192.168.0.6: icmp\_seq=3 ttl=64 time=0.373 ms

--- 192.168.0.6 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 2000ms

rtt min/avg/max/mdev = 0.305/0.409/0.550/0.104 ms

From the host, open a Bash shell on the container containera, and execute the ip addr show command to find out its IP address.

[bob@localhost ~]$ **lxc exec containera bash**

root@containera:~# **ip addr show**

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

    inet 127.0.0.1/8 scope host lo

       valid\_lft forever preferred\_lft forever

    inet6 ::1/128 scope host

       valid\_lft forever preferred\_lft forever

8: eth0@if2: <BROADCAST,MULTICAST,UP,LOWER\_

    link/ether 00:16:3e:98:e6:59 brd ff:ff:ff:ff:ff:ff link-netnsid 0

    inet 192.168.0.33/24 brd 192.168.0.255 scope global dynamic eth0

       valid\_lft 67171sec preferred\_lft 67171sec

    inet6 fe80::216:3eff:fe98:e659/64 scope link

       valid\_lft forever preferred\_lft forever

root@containera:~# **exit**

[bob@localhost ~]$

**IV. Further exercises that ask you to install nginx on the container, and ssh from and to it from another computer on your LAN.**

To follow these instructions up, use the appropriate sections of Chapter W23, that deal with Debian-family systems, as reference to allow you to install nginx on the container we named containera, and open a web page on it with your systems’ web browser. On our Debian-family Ubuntu 18.04 system installed on containera, and its nginx installation, the default index.html page was found at /var/www/html, and was named index.nginx-debian.html. Also enable the use of ssh to connect to the container from another computer system on the network (NOT from the host that the container is running on). Then use ssh to communicate between the container and another computer, and vice versa .