5.2 LXD

Introduction

> This article is useable, but not complete. What's missing,
>
> - Translation table of commands from LXC to LXD
> - Some more key basics of using LXD covered in LXC
> - Network discussion differences
> - Advanced Networking as a second article

In this version, you **must** finish the LXC and LXC with Advanced Networking before starting this article.

Intro to go here explaining LXD versus LXC and why you want to use it. For now, key bullets,

- LXD sits on top of LXC and uses a higher level set of commands
- REST API to orchestrates containers locally and remotely
- Allows moving and copying between hosts
- Takes advantage of advanced file systems (in particular ZFS)
- Improves on security specifically running as unprivileged hence root inside container does not get root on host

...

Setup

On your host system, either install LXC **or** install LXD. Though LXD setups an LXC subsystem, If found that the packages are different.

A key feature of virtualization technology is taking snapshots and cloning. With traditional file systems, this is expensive in terms of storage and speed. The next generation volume manager and file system ZFS solves many of theses problems and it is recommended by Ubuntu to install and use with LXD,

```
sudo apt-get install zfsutils-linux # must be using Ubuntu 16.04 or higher.
```

The other key feature to use with LXD is network bridging. By it's nature, the containers created by LXD exist in their own network. The network bridge utils allow you to expose your container to the rest of the network,

```
sudo apt-get install bridge-utils
```

Finally install LXD,

```
sudo apt-get install lxd
```

Initial Configuration

Before using LXD you need to to do an initial configuration.

The configuration is evolving and you will notice differences between versions of Ubuntu. Because the material is rather tough, we'll archive the older Ubuntu lxd init and keep the most current release here,

LXD Init for LTS

```
# show version of Ubuntu,
lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.2 LTS
Release:    16.04
Codename:   xenial

# initialize LXD
sudo lxd init
```

For the most part you will be able to hit Enter and accept defaults. Some options you may go back and change but others will require lots of work so take your time.

I'll go over some of the more confusing options,

```
Name of the storage backend to use (dir or zfs) [default=zfs]:
Create a new ZFS pool (yes/no) [default=yes]?
Name of the new ZFS pool [default=lxd]:
Would you like to use an existing block device (yes/no) [default=no]?
Size in GB of the new loop device (1GB minimum) [default=18]:
```

LXD allows management over the network. This is useful in an environment with multiple host LXDs and you want the ability to centrally manage. In this article, I am choosing yes because we'll later use Virtual Machine Manager to show GUI management,

```
Would you like LXD to be available over the network (yes/no) [default=no]?
yes
Address to bind LXD to (not including port) [default=all]:
Port to bind LXD to [default=8443]:
Trust password for new clients:
Again:
```

LXD init will configure a bridge. In order to say yes to this you must have the bridge utils installed. In this article we covered that so say yes

```
Do you want to configure the LXD bridge (yes/no) [default=yes]?
```

You will be prompted with a "text ui" interface. Use your arrow keys to pick the buttons.

Introduction,

> Conifiguring lxd
>
> Containers need a bridge to connect them together and to the host for outside network connectivity. Choosing this option will let you configure the default LXD bridge to your liking. If you would rather not have LXD do this for you, then you will be asked whether you want to use an existing bridge or just do everything manually. Would you like to setup a network bridge for LXD containers now?
>
> **Yes**

The default bridge name is lxdbr0. It's just a name but suggest you leave as default. To get to the button, use the down arrow key,

> Conifiguring lxd
>
> A valid network interface name (e.g. lxdbr0).
>
> Bridge interface name: lxdbr0
>
> **Ok**

Go with IPv4 unless you have some special need to use IPv6,

> This is needed to provide IPv4 connectivity for your containers. Do you want to setup an IPv4 subnet?
>
> **Yes**

You will be assigned a random subnet. For the purposes of this article, think of a subnet as a way of naming your postal code. It's the difference between Canada and US postal code. When given time I'll provide more description and write or link to a good article on subnets,

> A random subnet was selected for you. This subnet was selected for your convenience and the next questions have been pre-answered accordingly. Please make sure this subnet isn't already in use somewhere on your network, if it is, change it to one which isn't. If you later notice network connectivity issues, re-configure lxd and pick a different subnet.
>
> **Yes**

IPv4 address. This prompt is not very clear. Going to go back and check shortly. It is either reserving this IP address or asking you for a sample to determine the subnet.

> A valid IPv4 address. (e.g. 10.0.8.1): 10.139.135.1
>
> **Ok**

A valid CIDR mask. (e.g. 24),

> IPv4 CIDR mask:: 24
>
> **Ok**

Initial IP Address Range does not matter unless you want to use a lot of static IP addresses. I need to think about this a bit more... the idealism behind cloud is not require static, but reality (at least for now), I still find static useful and in some application designs (stateful apps, older PKI's... ect...) absolutely necessary.

> The first address to be handed out over DHCP (e.g. 10.0.8.2)
>
> First DHCP address: 10.139.135.2
>
> **Ok**

Pick default for the last address,

> The last address to be handed out over DHCP (e.g. 10.0.8.254)
>
> Last DHCP address: 10.139.135.254
>
> **Ok**

The DHCP leas number should be default unless you went and changed the range. If so you should adjust to match. Yeah I know the numbers do not add up (254 - 2 = 252) but I believe there is some fancy math for that... look up and link to my LXC articles.

The maximum number of DHCP leases that can be obtained. (e.g. 250)

Max number of DHCP clients: 250

Ok

For almost all networks this will be Yes. Tin google or to talk to Dickson or Andrew though at some point to put explanations here.

This is needed unless you are using a routed IPv4 subnet.

Do you want to NAT the IPv4 traffic?

Yes

Unless you really need IPv6 say no,

This is needed to provide IPv6 connectivity for your containers.

Do you want to setup an IPv6 subnet?

No

After this you will be taken out of the Text UI. The warning message is due to network bridge setup,

```
Warning: Stopping lxd.service, but it can still be activated by:
  lxd.socket
LXD has been successfully configured.
```

I hate doing this in Linux, but let's reboot to make sure the lxd service is ok,

```
sudo reboot
```

Add Users to the lxd Group

By default, users cannot yet use LXD until you add them to the reserved group. In this example, I'm adding my account,

```
sudo usermod -a -G lxd tin.pham
```

Reminder that if you use your own account, you need to log out then log back in.

Verify LXD is Working

Basic test to verify lxd is working and it will also generate your random client certificate (used by LXD to secure calls) on your account,

```
lxc image list
+-------+-------------+--------+-------------+------+------+-------------+
| ALIAS | FINGERPRINT | PUBLIC | DESCRIPTION | ARCH | SIZE | UPLOAD DATE |
+-------+-------------+--------+-------------+------+------+-------------+
```

At this point we have no LXD images.

Basic LXD Commands

To view your bridge information.

```
lxc profile edit default
```

Notice the following,

```
config: {}
description: Default LXD profile
devices:
  eth0:
    nictype: bridged
    parent: lxdbr0
    type: nic
name: default
```

What we see here is that the containers will be handed IPs from the LXD bridge and exist within the LXD network.

> Tin is refining the 2nd pass of this article ... I am currently at this point.

Images

...

Cached Images in Image Store

List images currently cached in the image store. There should be none to start,

```
lxc image list
```

Image Servers

LXD has 3 image server lists by default,

Image Server LIst	Purpose	Comment
ubuntu:	Ubuntu stable images.	We'll be working with this one.
ubuntu-daily:		
images:	All Linux distriibutions.	For example you can load a RedHat Linux alternative distrobution.

Pulling Image Server Lists

Let's look at the ubuntu: list,

```
lxc image list ubuntu: | less
+--------------------+-------------+--------+---------------------------
---------------------+---------+----------+-------------------------------+
|       ALIAS        | FINGERPRINT  | PUBLIC |
DESCRIPTION                    |  ARCH   |   SIZE   |          UPLOAD DATE
|
+--------------------+-------------+--------+---------------------------
---------------------+---------+----------+-------------------------------+
| p (5 more)         | 6041c5e200b6 | yes    | ubuntu 12.04 LTS amd64
(release) (20161205)     | x86_64  | 156.47MB | Dec 5, 2016 at 12:00am
(UTC)  |
+--------------------+-------------+--------+---------------------------
---------------------+---------+----------+-------------------------------+
| p/armhf (2 more)   | c19b1fff3336 | yes    | ubuntu 12.04 LTS armhf
(release) (20161205)     | armv7l  | 135.19MB | Dec 5, 2016 at 12:00am
(UTC)  |
+--------------------+-------------+--------+---------------------------
---------------------+---------+----------+-------------------------------+
| p/i386 (2 more)    | ce5c6821eebb | yes    | ubuntu 12.04 LTS i386
(release) (20161205)      | i686    | 139.28MB | Dec 5, 2016 at 12:00am
(UTC)  |
+--------------------+-------------+--------+---------------------------
---------------------+---------+----------+-------------------------------+
...
# Reduce the list to your machine architecture, in my case and most it will
be amd64
lxc image list ubuntu: amd64
```

The output table from list is not very clear. What you reference when using the "launch" command is the main part of the ALIAS name ignoring anything in brackets. For example to install and launch other distrobusions based off of the images: list which has all Linux distributions,

Example "launch" Command Reference	Column	Note
ubuntu:16.04		This will download from the Ubuntu image server grabbing the latest 16.04 matching your machines architecture, in my case amd64.
ubuntu:6041c5e200b6	FINGERPRINT	Will specifically reference the specific image in the list.
images:centos/7		Go against the "images" image server, download the latest cantos version 7 matching your machines architecture.
images:centos/6/amd64		Specifies the latest version 6 and the architecture to use.

At this point LXD is all setup and ready to use.

Spinning Up an LXD Container

Creating and starting a container can be done with one command,

```
lxc launch ubuntu:16.04 container01
```

This command does the following simultaneously,

1. References the **ubuntu:** Image Server List
    a. Looks for 16.01 images
    b. Matches the current architecture of the machine you are on (in my example
2. Checks the cache **Image Store** for the required image
3. Download (if not in Image Store) the target image
4. Setup Container called container01 with default settings
5. Install the target image into container01
6. Start container01

Replace **launch** with **init** if you would like the container to not start by itself.

We can see the downloaded target image,

```
lxc image list
+-------+--------------+--------+----------------------------------------
----+--------+----------+------------------------------+
| ALIAS | FINGERPRINT  | PUBLIC |                 DESCRIPTION
|  ARCH  |   SIZE   |         UPLOAD DATE          |
+-------+--------------+--------+----------------------------------------
----+--------+----------+------------------------------+
|       | f4c9feb3e401 | no     | ubuntu 16.04 LTS amd64 (release)
(20161205) | x86_64 | 143.53MB | Dec 22, 2016 at 3:37am (UTC) |
+-------+--------------+--------+----------------------------------------
----+--------+----------+------------------------------+
```

Now instead of empty, we see the downloaded image which matches our architecture.

Also, let's look at the running container. The commands are only slightly different than using straight LXC,

```
lxc list
+-------------+---------+----------------------+------+------------+------
-----+
|    NAME     |  STATE  |         IPV4         | IPV6 |    TYPE    |
SNAPSHOTS |
+-------------+---------+----------------------+------+------------+------
-----+
| container01 | RUNNING | 10.94.217.171 (eth0) |      | PERSISTENT | 0
|
+-------------+---------+----------------------+------+------------+------
-----+

# Inspect the container,
lxc info container01

Name: container01
Remote: unix:/var/lib/lxd/unix.socket
Architecture: x86_64
Created: 2016/12/22 03:37 UTC
Status: Running
Type: persistent
Profiles: default
Pid: 19681
Ips:
  eth0: inet 10.94.217.171 vethDXUYOE
  eth0: inet6 fe80::216:3eff:fe07:69ea vethDXUYOE
  lo: inet 127.0.0.1
  lo: inet6 ::1
Resources:
  Processes: 26
  Disk usage:
    root: 144.32MB
  Memory usage:
    Memory (current): 23.24MB
    Memory (peak): 45.28MB
# ...
```

For illustration of flexiblity and preparing for the next section, we will update and setup Apache inside of **container01**.

Remotely execute update of the container from the host which are run as root,

```
lxc exec container01 apt-get update
lxc exec container01 apt-get dist-upgrade
```

And now we'll actually create a bash to simulate going in as a console. Notice the prompt change indicating you are root inside of container01,

```
lxc exec container01 bash
root@container01:~#
```

Now we install Apache and then exit back to our host,

```
apt install apache2
exit # Takes you back to your host.
```

Managing Containers Between Hosts

The most compelling reason to use LXD is ability to transport between containers. Setup a second LXD host on the same network. In this example we end up with two hosts,

Host	Containers	Comment
myhost01	Where we setup container01 with Apache running inside.	This will be the host we can to copy the container from also called **remote host**.
myhost02	Just empty at the moment.	This will be the client also called the **local machine**.

Expose Remote Host

In order for the local machine to connect, the remote host needs to be setup to be exposed on the network with a password. Following the instructions here, that work has been done while initializing LXD.

Configure Local Machine

myhost02 (local machine) needs to be made aware of the remote host. I suppose there are multiple ways of doing this and interested in if there is automatic discovery. For now, I'm going to use direct IP address.

```
lxc remote add myhost01 192.168.0.109
Certificate fingerprint:
7344c171bdc30a20e215e536a7959353b55ab6243a019819fb62385fa02d26b2
ok (y/n)? y
Admin password for myhost01:
Client certificate stored at server:  myhost01
```

Now the list has been updated to have myhost02 as an entry,

```
lxc remote list
+-----------------+-----------------------------------------+------------
---+--------+--------+
|      NAME       |                  URL                    |   PROTOCOL
| PUBLIC | STATIC |
+-----------------+-----------------------------------------+------------
---+--------+--------+
| images          | https://images.linuxcontainers.org      |
simplestreams | YES    | NO     |
+-----------------+-----------------------------------------+------------
---+--------+--------+
| local (default) | unix://                                 | lxd
| NO     | YES    |
+-----------------+-----------------------------------------+------------
---+--------+--------+
| myhost01        | https://192.168.0.109:8443              | lxd
| NO     | NO     |
+-----------------+-----------------------------------------+------------
---+--------+--------+
| ubuntu          | https://cloud-images.ubuntu.com/releases |
simplestreams | YES    | YES    |
+-----------------+-----------------------------------------+------------
---+--------+--------+
| ubuntu-daily    | https://cloud-images.ubuntu.com/daily    |
simplestreams | YES    | YES    |
+-----------------+-----------------------------------------+------------
---+--------+--------+
```

Interacting with Remote Host

Interaction is exactly the same as a local container except you specify the registered lxd host name,

```
lxc list myhost01: # Notice myhost01 has container01 which we had setup.
+-------------+---------+----------------------+------+------------+------
-----+
|    NAME     |  STATE  |         IPV4         | IPV6 |    TYPE    |
SNAPSHOTS |
+-------------+---------+----------------------+------+------------+------
-----+
| container01 | RUNNING | 10.94.217.171 (eth0) |      | PERSISTENT | 0
|
+-------------+---------+----------------------+------+------------+------
-----+


lxc info myhost01:container01
lxc info myhost01:container01

Name: container01
Remote: https://192.168.0.109:8443
Architecture: x86_64
Created: 2016/12/22 03:37 UTC
Status: Running
Type: persistent
Profiles: default
Pid: 22659
Ips:
  eth0: inet 10.94.217.171 vethCIJK85
  eth0: inet6 fe80::216:3eff:fe07:69ea vethCIJK85
  lo: inet 127.0.0.1
  lo: inet6 ::1

Resources:
  Processes: 81
  Disk usage:
    root: 147.63MB

  Memory usage:
    Memory (current): 20.07MB
    Memory (peak): 42.18MB
# ....


# While local machine has no containers.
lxc list
+------+-------+------+------+------+-----------+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+------+-------+------+------+------+-----------+
```

Also keep this concept in mind even when referencing cached images.

Copying Containers Between Hosts

We will now copy myhost01, container01 (which is running Apache into) into myhost02. The copy is identical except no snapshots and volatile keys are regenerated. For example, the MAC address will be reset and a new host name embedded.

There are two key methods. Clone semi-live using snapshots and shutting down first.

> Investigate if cloning between different lxd versions is supported and to what extent.

Shutdown Approach

Log into myhost02

```
lxc stop myhost01:container01
# Verify container stopped
lxc list myhost01:
+-------------+---------+------+------+------------+-----------+
|    NAME     |  STATE  | IPV4 | IPV6 |    TYPE    | SNAPSHOTS |
+-------------+---------+------+------+------------+-----------+
| container01 | STOPPED |      |      | PERSISTENT | 0         |
+-------------+---------+------+------+------------+-----------+

lxc copy myhost01:container01 web01
```

A copy of container01 has been **copied** to myhost02 and given the container name **web01**. Everything is the same except for (...),

```
lxc list
+-------+---------+------+------+------------+-----------+
| NAME  |  STATE  | IPV4 | IPV6 |    TYPE    | SNAPSHOTS |
+-------+---------+------+------+------------+-----------+
| web01 | STOPPED |      |      | PERSISTENT | 0         |
+-------+---------+------+------+------------+-----------+
```

...

Semi-Live Approach

...

Taking Snapshots

Snapshots are a great way to backup a container before upgrading or changing any files that will break a container.

```
lxc snapshot container01 snapshotname01
```

Using lxc list will show that your container now has a snapshot.

```
lxc list
+-------------+---------+------+------+------------+-----------+
|    NAME     |  STATE  | IPV4 | IPV6 |    TYPE    | SNAPSHOTS |
+-------------+---------+------+------+------------+-----------+
| container01 | STOPPED |      |      | PERSISTENT | 1         |
+-------------+---------+------+------+------------+-----------+
```

The snapshot name can be viewed using the lxc info command

```
lxc info container01
Name: container01
Remote: https://192.168.0.109:8443
Architecture: x86_64
...
Snapshots:
  snapshotname01 (taken at 2017/02/02 15:33 UTC) (stateless)
```

To restore a snapshot

```
lxc restore container01 snapshotname01
```

Renaming a snapshot

```
lxc move container01/snapshotname01 container01/snaptestname01
lxc info container01
...
Snapshots:
  snaptestname01 (taken at 2017/02/02 15:33 UTC) (stateless)
```

Creating a container from a snapshot

```
lxc copy container01/snaptestname01 container02
```

Deleting a snapshot

```
lxc delete container01/snaptestname01
```

Moving Containers

With copy, the new container is identical in every way **except** no snapshots and volatile keys (ie MAC address and hostnames) will be regenerated.

Creating an **exact** duplicate or **moving** a container is a different concept but just as easy.

Current Limitations and Considerations

- Consistency of host IDs which impact containers
- Architecture
- Baselining host patches and OS layer
- No concept yet or documentation on Sparse Containers like Solaris...

Advanced Tuning of Containers

Depending on your needs there are advanced configurations options with LXD. Below are key considerations I think about,

1. Security
2. Advanced Networking
    a. Tunnelling ect... as covered in LXC with Advanced Networking
    b. Joining LXD host networks together
3. Capping Resource Utilization
4. Sparse Containers like Solaris...
5. Permissions Inside and Outside of Containers

File Transfer with Containers

Files can now be transferred from host to containers or container to container with the new lxc push and pull commands.

```
lxc file push sourcefile container/path/to/directory/
lxc file pull container/path/to/file directory
```

Reconfigure

If you need to reconfigure the bridge you can by using this command.

```
sudo dpkg-reconfigure -p medium lxd
```

Reference

Initializing LXD - https://insights.ubuntu.com/2016/03/16/lxd-2-0-installing-and-configuring-lxd-212/

Official Ubuntu Documentation - https://linuxcontainers.org/lxd/getting-started-cli/

Try Online Interactive Tutorial - https://linuxcontainers.org/lxd/try-it/

Remote Container Management - https://www.stgraber.org/2016/03/19/lxd-2-0-your-first-lxd-container-312/

Possible Sparse Containers Approach - https://www.hastexo.com/blogs/florian/2016/02/21/containers-just-because-everyone-else/

Live Container Migrations - https://bobcares.com/blog/lxc-live-migration-to-minimize-business-downtime/

Mounting a local folder - https://tribaal.io/nicer-mounting-home-in-lxd.html