How to use Linux Containers (LXC)

Author

Asad Javed
Aalto University, Department of Computer Science
Finland

Table of Contents

1. Introduction

In this document, we will discuss Linux Containers (LXC) and how to use them. LXC is a fast and lightweight virtualization technique which creates multiple Linux systems on a single host. It provides virtualization at operating system level and shares the same kernel, thus reducing the overhead of using hypervisor such as Virtual Box, KVM, and VMware. The containers also share the resources such as CPU, Hard disk, RAM, network, etc. of the host OS. In order to completely understand the theory behind LXC and kernel features, check my second paper [1] which is provided along with this document.

In this tutorial, we will learn how to install LXC on Linux and then create and use virtual machines using LXC. Ubuntu 14.04 is used as a host OS.

2. Installing LXC on Ubuntu

There is a package named lxc which is already available in Ubuntu repositories. You can install it by typing the following command on a terminal.

```
$ sudo apt-get install lxc
```

After executing above command, the lxc will be installed. Now type the following command in order to install its directives.

```
$ sudo apt-get install lxctl lxc-templates
```

Now, type following command in order to check the installation. The output is shown in black.

```
$ sudo lxc-checkconfig
```

```
Kernel configuration not found at /proc/config.gz; searching...
Kernel configuration found at /boot/config-3.19.0-25-generic
--- Namespaces ---
Namespaces: enabled
Utsname namespace: enabled
Ipc namespace: enabled
Pid namespace: enabled
User namespace: enabled
Network namespace: enabled
Multiple /dev/pts instances: enabled

--- Control groups ---
Cgroup: enabled
Cgroup clone_children flag: enabled
Cgroup device: enabled
Cgroup sched: enabled
Cgroup cpu account: enabled
Cgroup memory controller: enabled
Cgroup cpuset: enabled

--- Misc ---
Veth pair device: enabled
Macvlan: enabled
Vlan: enabled
File capabilities: enabled

Note : Before booting a new kernel, you can check its configuration
usage : CONFIG=/path/to/config /usr/bin/lxc-checkconfig
```

3. Creating Linux containers

LXC is very easy to use as there are already many templates available by default. You can check all these templates by typing the following command.

```
$ sudo ls /usr/share/lxc/templates/
```

Output:

```
lxc-alpine    lxc-archlinux  lxc-centos  lxc-debian
lxc-altlinux  lxc-busybox    lxc-cirros  lxc-download
lxc-fedora  lxc-openmandriva  lxc-oracle  lxc-sshd    lxc-ubuntu-cloud
lxc-gentoo  lxc-opensuse      lxc-plamo   lxc-ubuntu
```

In the output above, you can see many different templates available with lxc package. These templates are used to create containers.

In order to create a container, the following command shows the syntax with one example.

Syntax: sudo lxc-create -n <any user-defined container name> -t <template>

```
$ sudo lxc-create -n ubuntu1 -t ubuntu
```

The above command creates a Ubuntu container with a name ubuntu1. The (-n) option is for giving the name to the container, and (-t) option is to assign template.

4. Starting containers

The following command is used to start containers. Because of this '-d' option, the container will start in the background, detached from the console.

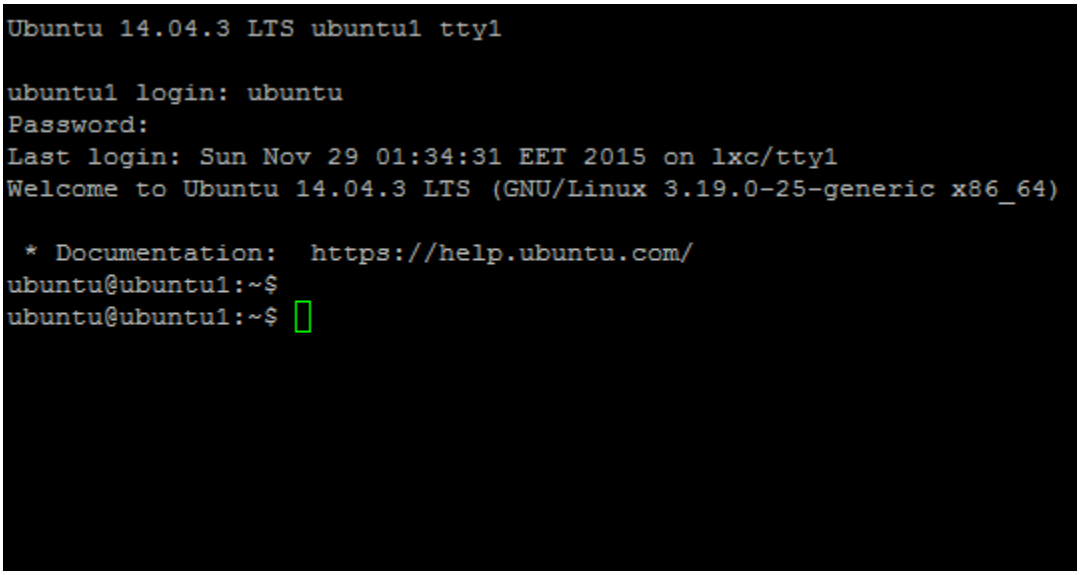
```
$ sudo lxc-start -n ubuntu1 -d
```

The above command launches your container in the background. In order to attach it to the console, use the below lxc-console command.

```
$ sudo lxc-console –n ubuntu1
```

The output below shows the console of the container which you created. Now you can us it a simple OS and install different packages and run processes.

Output:



This is the console of your created container, you can come back to original host terminal by pressing "Ctrl-a q" combination (first press Ctrl-a then release it and then press q). You can also close it using:

```
$ sudo poweroff
```

Now in order to check the list of created containers, type the following command:

```
$ sudo lxc-ls
```

The above command shows that there is only one container named 'ubuntu1'. You can also view the complete details of the container using:

```
$ sudo lxc-info -n ubuntu1
```

Output:

```
Name:           ubuntu1
State:          RUNNING
PID:            7806
IP:             10.0.3.85
CPU use:        1.27 seconds
BlkIO use:      22.03 MiB
Memory use:     28.72 MiB
KMem use:       0 bytes
Link:           vethGYDDKJ
 TX bytes:      1.59 KiB
 RX bytes:      1.82 KiB
 Total bytes:   3.41 KiB
```

The above output shows the complete information regarding the container ‘ubuntu1’, which you created. You can also check the running state of the container using:

```
$ sudo lxc-ls --fancy ubuntu1
```

Output:

```
NAME     STATE    IPV4       IPV6  AUTOSTART
----------------------------------------------
ubuntu1  RUNNING  10.0.3.85  -     NO
```

5. Cloning containers

It is also possible to clone an existing container, means make a similar copy. It can be done using the following command:

```
$ sudo lxc-clone ubuntu1 ubuntu2
```

The above command makes a copy of ‘ubuntu1’ and creates ‘ubuntu2’. The new container has the same structure and resources of the original one.

6. Stopping containers

In order to stop the running container, type the following ‘lxc-stop’ command:

```
$ sudo lxc-stop -n ubuntu1
```

To check the status, use the following command with option ‘fancy’.

```
$ sudo lxc-ls --fancy ubuntu1
```

Output:

```
NAME     STATE    IPV4  IPV6  AUTOSTART
---------------------------------------
ubuntu1  STOPPED  -     -     NO
```

You can also pause you container by typing

```
$ sudo lxc-freeze -n ubuntu1
```

The status can be check using:

```
$ sudo lxc-ls --fancy ubuntu1
```

Output:

```
NAME     STATE   IPV4       IPV6  AUTOSTART
----------------------------------------------
ubuntu1  FROZEN  10.0.3.85  -     NO
```

So, the container will get freeze as seen from the above output. You can unfreeze it by using 'lxc-unfreeze' command. The syntax is the same as 'lxc-freeze' command.

7. Deleting containers

In order to permanently delete the containers, type the following command:

```
$ sudo lxc-destroy -n ubuntu1
```

8. Controlling Cgroups

Control groups (cgroups) are one of the main kernel features which are used to isolate resource usage within a containers such as CPU time, system memory, disk bandwidth, network bandwidth, and monitoring. LXC uses cgroups in its implementation and we can use 'lxc-cgroup' command to control the access to system resources.

As can be seen from above commands that we have created one container named 'ubuntu1'. In order to check the number of CPU cores on which this container can run on, type the following command:

```
$ sudo lxc-cgroup -n ubuntu1 cpuset.cpus
```

This command prints '0' as output. It means that this container is running on CPU core 0. If you want to restrict a container to cores 0 and 1, type the following command:

```
$ sudo lxc-cgroup -n ubuntu1 cpuset.cpus 0,1
```

The above command will only work if you have multiple cores in your system. Otherwise it would give an error.

Similarly, to see and change the container share of CPU time, you can use the following commands.

```
$ sudo lxc-cgroup -n ubuntu1 cpu.shares
```

```
$ sudo lxc-cgroup -n ubuntu1 cpu.shares 512
```

The first command display the default share value of CPU time. The second command change that value to 512. In my case the default value was 1024.

Furthermore, you can also limit memory of the container. There are two limits: one is soft and one is hard. If the system detects memory contention or low memory than it automatically limit the container memory to the value set by soft limit, otherwise the value of memory is the hard limit. It can be set using the following commands.

```
$ sudo lxc-cgroup -n ubuntu1 memory.soft_limit_in_bytes 53687091
```

```
$ sudo lxc-cgroup -n ubuntu1 memory.limit_in_bytes 1073741824
```

The first command sets the soft limit to 512 MB and the second command sets hard limit to 1024 MB.

9. LXC Networking

Every container has an IP address in order to be available on network. In Linux ‘bridges’ are used to connect VMs to a network. Bridges are a basic functionality of the Linux kernel and are usually created using the ‘bridge-utils’ package. You can configure two main types of bridges:

Host Bridge: In this bridge, the containers are directly connected to the host network and appear and behave as other physical machines on your network.

NAT Bridge: This is a private network within your host machine and containers are connected to this network. This is a standalone bridge and all networking happens through host IP.

In order to create a direct host bridge in Ubuntu, edit the file ‘/etc/network/interfaces’ to create a bridge named ‘br0’. The file would look like the below output.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet manual

auto br0
iface br0 inet dhcp
bridge_ports eth0
bridge_stp off
bridge_fd 0
bridge_maxwait 0
```

The above output shows that there is a bridged network ‘br0’. Now you can configure containers to connect to this bridge and they will get their IPs from the router your host is connected to and be on the same network.

Containers are by default stored in ‘/var/lib/lxc’ directory. Every container has a separate config file which is also created when you create a container. For example, you have already created a container named ‘ubuntu1’, so its config file can be seen by typing the following command.

```
$ cat /var/lib/lxc/ubuntu1/config
```

Output:

```
# Template used to create this container: /usr/share/lxc/templates/lxc-ubuntu
# Parameters passed to the template:
# For additional config options, please look at lxc.container.conf(5)

# Common configuration
lxc.include = /usr/share/lxc/config/ubuntu.common.conf

# Container specific configuration
lxc.rootfs = /var/lib/lxc/ubuntu1/rootfs
lxc.mount = /var/lib/lxc/ubuntu1/fstab
lxc.utsname = ubuntu1
lxc.arch = amd64

# Network configuration
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = lxcbr0
lxc.network.hwaddr = 00:16:3e:65:8a:e2
```

In the above output, the parameter ‘lxc.network.link’ is used to change the network bridge. The value lxcbr0 shows that this container is by default connected to lxcbr0 bridge. In order to change it, replace this value with br0, which you created above in ‘/etc/network/interfaces’ file. Moreover, instead of editing separate config file for each container, you can also set a system wide setting by editing a file ‘/etc/lxc/default.conf’ so that containers created have a default network settings.

The contents of '/etc/lxc/default.conf' file are shown below.

```
lxc.network.type = veth
lxc.network.flags = up
lxc.network.link = lxcbr0
lxc.network.name = eth0
lxc.network.hwaddr = 00:16:3e:xx:xx:xx
lxc.network.mtu = 1500
```

This is a common system file and all the containers are connected to the default lxcbr0 bridge. If you change your bridge setting here than it would adopt by all containers.

References

[1] Asad Javed, "Linux Containers: An Emerging Cloud Technology"
[2] Oracle Linux, 2015 [Online]. URL: https://docs.oracle.com/cd/E37670_01/E37355/html/