Linux Containers (LXC)

Qingwei Lan (404458762)
University of California, Los Angeles

May 9, 2016

1 Introduction to Linux Containers

Linux Containers is an implementation for the "containers" concept. Traditionally virtualization starts with the physical hardware at the bottom, then comes the kernel and operating system, and above that is the hypervisor. Atop the hypervisor are the virtual machine images that allow the user to run many different systems. A container is similar to the traditional virtualization method except that it doesn't have the hypervisor layer.

A major advantage of containers over traditional virtual machines is speed since containers do not run an individual operating system but shares the same host operating system. This allows it to make efficient use of system calls and hardware resources without going through the hypervisor layer.

Linux Containers provides a virtual envoronment complete with its own CPU, memory, and I/O, which is made possible by cgroups and namespaces in the Linux kernel.

2 Commonly Used LXC Commands

In this section I will introduce some of the most commonly used commands in the current stable 2.0 version of LXC. These command binaries can be found in the `src/lxc` directory after compilation through the command "`./autogen.sh && ./configure && make`".

`lxc-create, lxc-destroy`

These two commands are used to create and destroy containers, taking the container name as the parameter. For example to create and destroy a container named `p1`, we execute the following commands

```
$ lxc-create --name p1
$ lxc-destroy --name p1
```

`lxc-start, lxc-stop`

These two commands are used to start a container and stop a container, taking the container name as the parameter. We could also start a container in the background by setting an optional parameter `--daemon`. Stopping a container kills all processes within the container. For example to start and stop a container named `p1`, we execute the following commands

```
$ # runs in background if --daemon is set
$ lxc-start --name p1 [--daemon]
$ lxc-stop --name p1
```

`lxc-freeze, lxc-unfreeze`

These two commands are used to temporarily stop all processes in containers or resume the stopped processes. The two commands take the container name as the parameter. For example to freeze and unfreeze a container named `p1`, we execute the following commands

```
$ lxc-freeze --name p1
$ lxc-unfreeze --name p1
```

`lxc-ls`

This command is used to list all containers in the system. Because it is built on top of the GNU `ls` program, it can be run with the options of the `ls` program. Following are example commands

```
$ lxc-ls # list all containers
$ lxc-ls -Cl # list all containers in one column
$ lxc-ls -l # list all containers and permissions
```

`lxc-info`

This command is used for displaying the information of a specific container, taking the name of the container as the parameter. For example to show the information of a container named `p1`, we execute the following command

```
$ lxc-info --name p1
```

`lxc-monitor`

This command is used for monitoring the states of one or more containers. It takes the container names (in regular expression format) as the parameter. The command will

output state changes for the specified containers. Following are example commands to monitor containers named `a` and `b` or all containers in the system

```
$ lxc-monitor --name a # monitor container a
$ lxc-monitor --name 'a|b' # monitor containers a and b
$ lxc-monitor --name '.*' # monitor all containers
```

`lxc-wait`

This command waits for a container to reach one or more specific states and exits. It takes the container name and intended states as parameters. Following are examples to wait on a container called `p1`

```
$ # exits when RUNNING is reached
$ lxc-wait --name p1 --states RUNNING

$ # exits when RUNNING or STOPPED is reached
$ lxc-wait --name p1 --states RUNNING|STOPPED
```

`lxc-attach`

This command starts a process inside a running container, taking the name of container and the command to run as parameters. The container has to be running in order for this command to work. To spawn a new shell inside a container named `p1`, execute the following command

```
$ lxc-attach --name p1
```

`lxc-copy`

This command creates a copy of existing containers and optionally starts the copy. It takes the to-be-copied container name as a parameter. This command is intended to replace the deprecated commands `lxc-clone` and `lxc-start-ephemeral`.

3 New Directory Methods

For the first part of my project I will be adding some direcory methods to the outer layer of the container. The methods will be `lxc-pd` (print directories in a container) and `lxc-search` (search for a file in container)

3.1 `lxc-pd` (print-directory)

This command will be used to print directories in a container specified by the `--name` or `-n` option. It will print recursively for `N` levels (default is 1) starting from the root directory. The number of levels can be specified with the option `--level` or `-l` followed by the number of levels. An example usage will be

```
$ lxc-pd --name p1 --level 5
```

This will print directories in container `p1` for up to 5 levels of recursion.

3.2 `lxc-search` (search)

This command will search for a file in a container specified by the specified by the `--name` or `-n` option. The file is specified by the option `--file` or `-f` followed by the file name. An example usage will be

```
$ lxc-search --name p1 --file file-name
```

This will search for a file named `file-name` inside the container named `p1`