

Linux Containers

Objectives:

After completing this lab you should be able to:

1. Explain LXC and differentiate between LXC containers and Docker Containers.
2. install and configure LXD.
3. Create and run containers.

Required resources:

- A PC with virtualBox and putty installed.
- Ubuntu server VM with a bridged adapter , 2 CPUs, and 4 GB of memory.
- Internet connection.

Setting up the Work Environment

- On Vbox create a VM for Ubuntu 64 bit.
 - HDD: 40 GB.
 - RAM: 4 GB.
 - CPU cores 2.
 - Bridged network adapter.
- During ubuntu server installation make sure to select the option to install **openSSH server**. If you missed that do it after starting the machine.
- Document the user name and password in the description field of the VM settings.
- After installation is complete record the machine ip address and do a repository update and dist-upgrade and shutdown you machine.

```
$ ip addr
$ sudo apt update && sudo apt -y dist-upgrade
$ shutdown -h 0
```
- Now lets make the VM start in headless mode: in Vbox right click the machine and select headless start from the start sub-menu.
- Open putty and start an ssh connection to the VM.

Install LXD

- Use the following command to install LXD on your VM:

```
sudo apt install lxd
```

Check the version of lxd installed using the following command:

```
lxd version
```

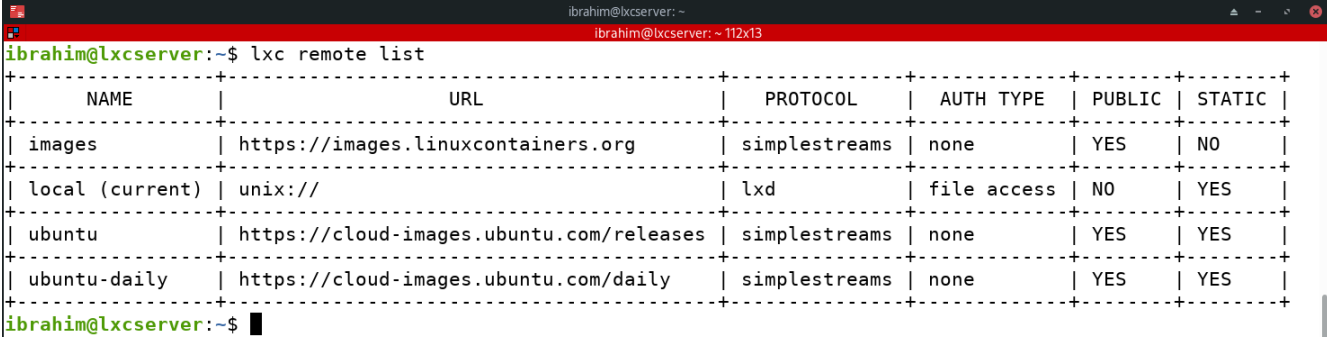
- To initialize LXD run the following command:

```
sudo lxd init
```

accept all default options except storage, set storage to use a local directory using the dir option.
To continue using lxd we will be using the client application called lxc.

LXC images

To check the image sources (servers) available use the following command:

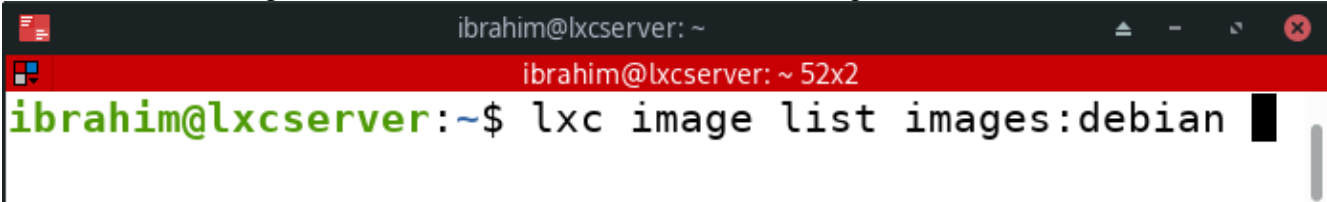


```
ibrahim@lxcserver:~$ lxc remote list
```

NAME	URL	PROTOCOL	AUTH TYPE	PUBLIC	STATIC
images	https://images.linuxcontainers.org	simplestreams	none	YES	NO
local (current)	unix://	lxd	file access	NO	YES
ubuntu	https://cloud-images.ubuntu.com/releases	simplestreams	none	YES	YES
ubuntu-daily	https://cloud-images.ubuntu.com/daily	simplestreams	none	YES	YES

```
ibrahim@lxcserver:~$
```

To search for an image on a certain remote server use the following command:



```
ibrahim@lxcserver:~$ lxc image list images:debian
```

Here we are searching for available Debian images on the images server. If you don't specify Debian as a search string it will list all available images on that server. If you don't specify a server it will list available local images that we have used previously.

Container management

Create a container

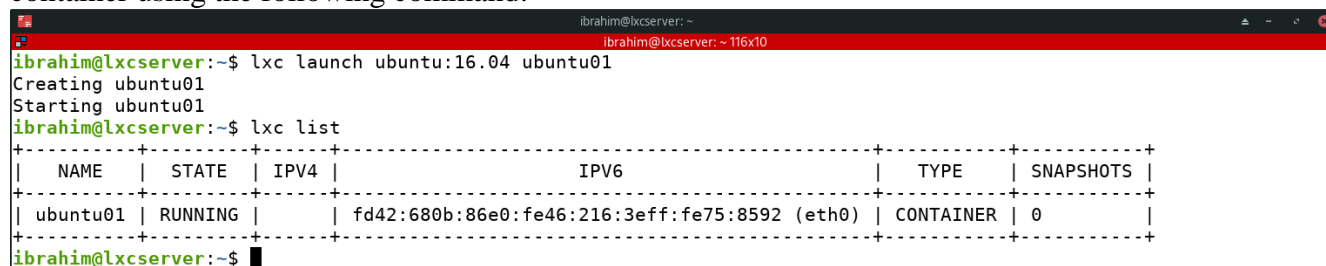
before starting your first container use the **lxc list** command to check for existing containers.



```
ibrahim@lxcserver: ~$ lxc list
```

NAME	STATE	IPV4	IPV6	TYPE	SNAPSHOTS
------	-------	------	------	------	-----------

The list is now empty since we haven't created any containers yet. Let's create an ubuntu 16.04 container using the following command:



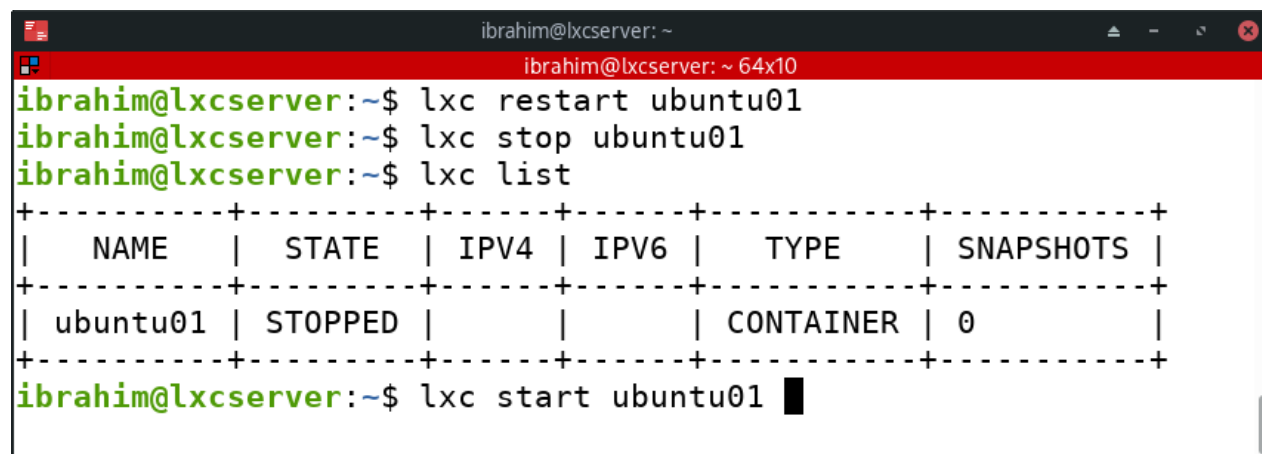
```
ibrahim@lxcserver:~$ lxc launch ubuntu:16.04 ubuntu01
Creating ubuntu01
Starting ubuntu01
ibrahim@lxcserver:~$ lxc list
```

NAME	STATE	IPV4	IPV6	TYPE	SNAPSHOTS
ubuntu01	RUNNING		fd42:680b:86e0:fe46:216:3eff:fe75:8592 (eth0)	CONTAINER	0

this command translates as follows: start a new container from the ubuntu image server based on ubuntu 16.04 and name the new container ubuntu01.

Restart/Stop/Start a container

use the following commands:



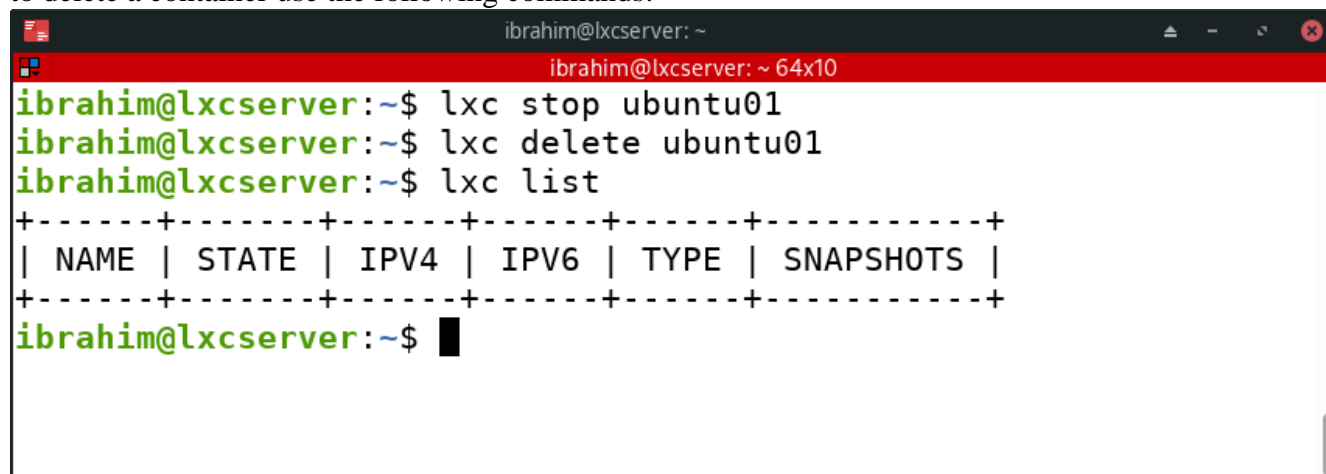
```
ibrahim@lxcserver:~$ lxc restart ubuntu01
ibrahim@lxcserver:~$ lxc stop ubuntu01
ibrahim@lxcserver:~$ lxc list
```

NAME	STATE	IPV4	IPV6	TYPE	SNAPSHOTS
ubuntu01	STOPPED			CONTAINER	0

```
ibrahim@lxcserver:~$ lxc start ubuntu01
```

Deleting a container

to delete a container use the following commands:



```
ibrahim@lxcserver: ~  
ibrahim@lxcserver: ~ 64x10  
ibrahim@lxcserver:~$ lxc stop ubuntu01  
ibrahim@lxcserver:~$ lxc delete ubuntu01  
ibrahim@lxcserver:~$ lxc list  
+-----+-----+-----+-----+-----+-----+  
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |  
+-----+-----+-----+-----+-----+-----+  
ibrahim@lxcserver:~$
```

Executing commands inside the container:

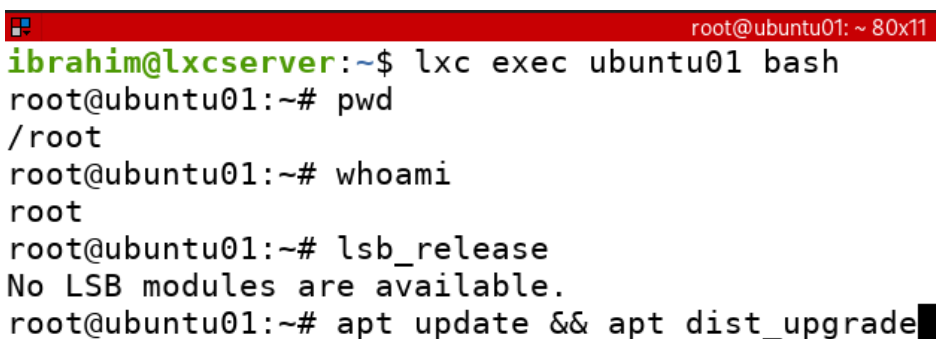
recreate ubuntu01 container. Use the following to execute the nproc command inside our container:



```
ibrahim@lxcserver: ~ 80x11  
ibrahim@lxcserver:~$ lxc exec ubuntu01 nproc  
1
```

here we executed the nproc command that lists the number of CPUs that the container sees. This will vary depending on the number of CPUs you have on your system. Containers can see all system resources and can take over them!!!. we will limit that for security reasons.

Another way is to open the bash of the container and that way we can execute as many commands as we want. Execute the commands below:



```
ibrahim@lxcserver:~$ lxc exec ubuntu01 bash  
root@ubuntu01:~# pwd  
/root  
root@ubuntu01:~# whoami  
root  
root@ubuntu01:~# lsb_release  
No LSB modules are available.  
root@ubuntu01:~# apt update && apt dist_upgrade
```

exit by pressing ctrl+d or typing exit. List the current containers and see the state of the ubuntu01 container. Is it running or stopped? How would you compare docker containers to lxc containers now?

Cloning a container

To replicate an existing container use the following command:

```
ibrahim@lxcserver:~$ lxc copy ubuntu01 ubuntu02
ibrahim@lxcserver:~$ lxc list
```

NAME	STATE	IPV4	IPV6	TYPE	SNAPSHOTS
ubuntu01	RUNNING	10.175.15.99 (eth0)	fd42:680b:86e0:fe46:216:3eff:fe71:2c98 (eth0)	CONTAINER	0
ubuntu02	STOPPED			CONTAINER	0

now lets do some tasks:

1. start ubuntu02.
2. start a bash session.
3. ping ubuntu01.

Was it successful or failed? _____

By default you will start bash as the root user. Every container image has a default username, in ubuntu container images the default username is ubuntu. Let's login to the container using the ubuntu username and do some tasks. Use the following command:

```
ibrahim@lxcserver:~$ lxc exec ubuntu01 su - ubuntu
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

```
ubuntu@ubuntu01:~$ █
```

now do the following:

1. set the current user password to ubuntupass.
2. install openssh server.
3. exit out of this container.
4. open a bash shell on ubuntu02.
5. ssh into ubuntu01.

Call your instructor to verify that you did this task before proceeding.

Rename a Container

To rename a container use the move command as follows:

```
ibrahim@lxcserver:~$ lxc move ubuntu01 newubuntu
Error: Renaming of running container not allowed
ibrahim@lxcserver:~$ lxc stop ubuntu01
ibrahim@lxcserver:~$ lxc move ubuntu01 newubuntu
ibrahim@lxcserver:~$ lxc start newubuntu
```

you cannot rename a running container.

File sharing

To share files with containers you can use the `lxc file` command as follows:

```
ibrahim@lxcserver:~$ touch sharedfile
ibrahim@lxcserver:~$ ls
sharedfile  snap
ibrahim@lxcserver:~$ lxc file push sharedfile newubuntu/root/
ibrahim@lxcserver:~$ lxc exec newubuntu ls
sharedfile
ibrahim@lxcserver:~$ rm sharedfile
ibrahim@lxcserver:~$ lxc file pull newubuntu/root/sharedfile restoredfile
ibrahim@lxcserver:~$ ls
restoredfile  snap
```

Snapshots

login to newubuntu and create some files and directories in the root directory. Then create a snapshot called `snap01`.

```
ibrahim@lxcserver:~$ lxc snapshot newubuntu snap01
ibrahim@lxcserver:~$ lxc list
```

NAME	STATE	IPV4	IPV6	TYPE	SNAPSHOTS
newubuntu	RUNNING	10.175.15.99 (eth0)	fd42:680b:86e0:fe46:216:3eff:fe71:2c98 (eth0)	CONTAINER	1
ubuntu02	STOPPED			CONTAINER	0

Login again delete the files you created then restore the container from the snapshot and check if your files are back.

```
ibrahim@lxcserver:~$ lxc restore newubuntu snap01
```

Setting limits

All host hardware is exposed to the container which means every container sees that all resources are available for it to consume as it needs. This is not a good thing that is why we are setting some limitations.

Login to newubuntu and check number of CPUs and memory size available to the container. List commands and output below:

now lets limit those two things:

```
ibrahim@lxcserver:~$ lxc stop newubuntu
ibrahim@lxcserver:~$ lxc config set newubuntu limits.memory 512MB
ibrahim@lxcserver:~$ lxc config set newubuntu limits.cpu 1
```

now start the container and check the memory and CPU that the container sees. There are other ways to set the limits using profile files but this will be left for you to explore.

One other thing to explore is nested containers :).

Networking

1. install apache2 on ubuntu02 container.
2. from the lxcserver machine execute **curl http://[ubuntu02_IP]**. Did it work?
3. from your host machine try to open the VM IP address inside a browser. Did it work? Now open the container IP address in the browser. Did it work?
4. to configure port forwarding execute the following command:

```
lxc config device add ubuntu02 frwrdport proxy listen=tcp:0.0.0.0:80
connect=tcp:127.0.0.1:80
```

this command will add a device called frwrdport to the container named ubuntu02 which will make our host VM listen on port 80 and forward any requests to the container.

Now from your host open the browser and type the VM IP address this should open the container default Apache website.

Other options include creating a bridged network for the container to connect directly to our LAN but we will leave that for you to explore and learn.

Clean up

Stop and delete all containers. List commands bellow:

To Do:

1. Create a container based on ubuntu 14.04 release.
2. install and configure ssh server.
3. configure port forwarding so that we can ssh into the container from our windows machine using port 5000.

End