Al-Najah National University
Computer Network and Information Security
Network Administration Lab

LXC

Instructor:Dr. Ahmed Awad

Dima Bshara
Mohammed Adnan

1 Abstract

This experiment explains LXC and differentiates between LXC containers and Docker Containers, then how to install and configure LXD. Also, to create and run containers and make some management of the containers.

2 Introduction

LXC (a Linux container) is a virtualization solution on the operating system level that enables the creation and operation of many isolated Linux virtual environments (VE). These can be spun up on a single centralized host. Containers, which are isolation levels, can isolate certain apps or simulate a fully different host. LXC mitigates VM disadvantages. Linux containers enable the host CPU to effectively divide memory allocations into confinement levels known as namespaces. In comparison, a VM contains the whole operating system and machine setup/emulators, such as the hard disk, virtual CPUs, and network interfaces. The entire (enormous) virtualized environment typically takes some time to boot and consumes a large amount of RAM and CPU. LXC virtual environment has no hardware preload emulation. Each virtual environment (an OS or an application) is loaded in a container and executes without any additional overhead and no hardware emulation. This means no penalty from software with limited memory. In the end, LXC will improve the performance of the bare metal as it only bundles the OS/application that is required. Docker is a containerized virtual environment that makes it easier to develop, maintain, and deploy applications and services. Docker containers are incredibly light, and you don't have to configure or set up virtual machines and environments, meaning you have hypervisors in between. Docker containers work off directly from the host OS. Essentially it doesn't have a separate kernel to run its containers. It utilizes the same resources as the host OS. Docker exploits namespaces and control groups to let you use host OS resources more efficiently. Let's interpret this with a diagram.

The following figure shows the composition of each of them:

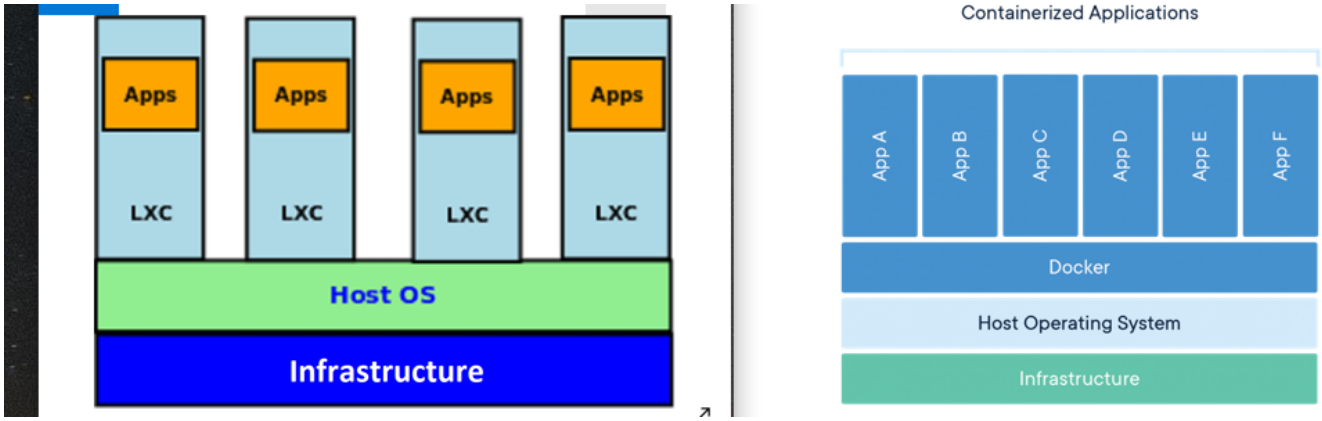

Figure 1: LXC and Docker.

the major difference between Linux and Docker containers is that the LXC focuses on OS containerization, while Docker thrives on application containerization. Docker is single-purpose application virtualization, and LXC is multi-purpose operating system virtualization.

In this case, LXC specializes in deploying Linux Virtual machines. A container is like a VM with a fully functional OS environment. However, the OS has to support and handle the features and capabilities of a Linux environment. You can SSH into an LXC container, operate it as an operating system, and install any application or services, and everything will work as expected.

This is not the case with Docker. Docker specializes in deploying applications. Docker containers aren't lightweight virtual machines. Thus they can't be considered as such.

Docker containers are limited to a single application due to their architecture. Although Docker runs natively in a Linux environment, it is not entirely dependent on Linux, and it supports other operating systems such as Windows and MacOS.

3 Procedure

3.1 Install LXD

1. We use the following command to install LXD on VM as figure 2 show.

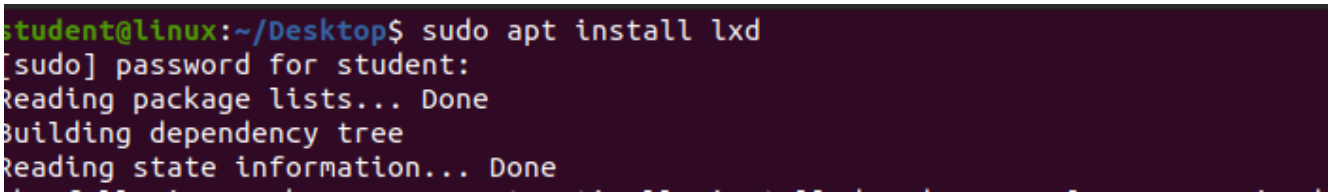

Figure 2: Install LXD.

2. Then we check the version of lxd installed using the following command: **lxd version** as figure 3 show that version is **4.19**.

Figure 3: Version LXD.

3. After that, we initialize LXD using the following command **sudo lxd init** with the default option except storage, we set storage to use a local directory, as figure 4 show .

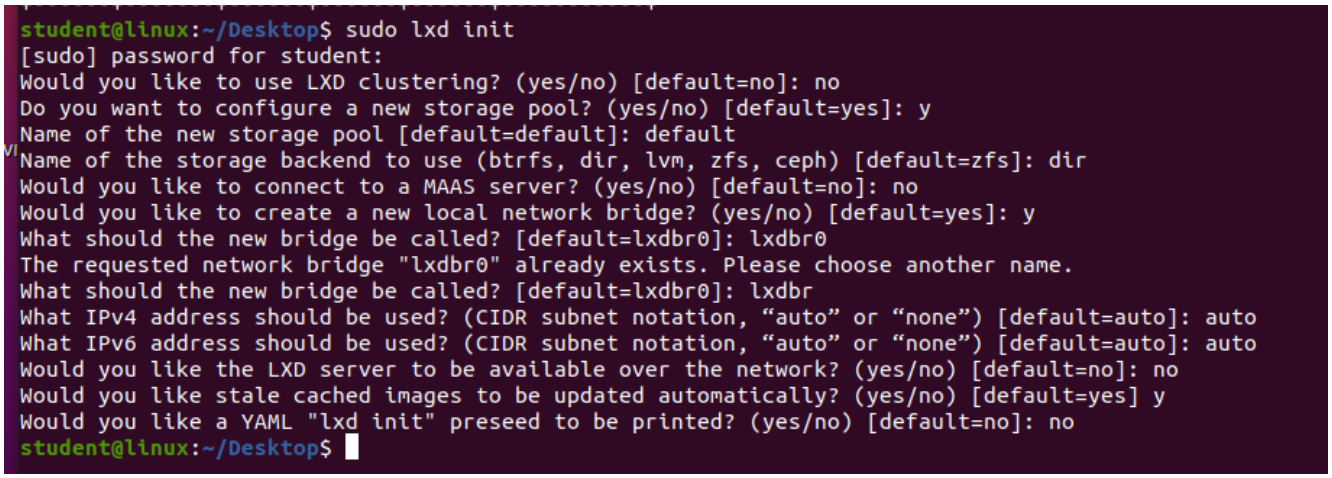

Figure 4: Initialize LXD.

3.2 LXC images

1. To check the image sources available we use the following command:**lxc remot list** as figure 5 show ther is default images that get by initialize LXD.

```
student@linux:~/Desktop$ lxc remote list
To start your first instance, try: lxc launch ubuntu:20.04

+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
|      NAME       |                   URL                   |   PROTOCOL    |  AUTH TYPE  | PUBLIC | STATIC | GLOBAL |
+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
| images          | https://images.linuxcontainers.org      | simplestreams | none        | YES    | NO     | NO     |
+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
| local (current) | unix://                                 | lxd           | file access | NO     | YES    | NO     |
+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
| ubuntu          | https://cloud-images.ubuntu.com/releases | simplestreams | none        | YES    | YES    | NO     |
+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
| ubuntu-daily    | https://cloud-images.ubuntu.com/daily   | simplestreams | none        | YES    | YES    | NO     |
+-----------------+-----------------------------------------+---------------+-------------+--------+--------+--------+
student@linux:~/Desktop$
```

Figure 5: Check images sources.

2. We search for available debian images on the images server, as figure 6 show .

```
student@linux:~/Desktop$ lxc image list images:debian
+------------------------------+--------------+--------+-----------------------------------------+--------------+-----------------+----------+------------------------
----------+
|            ALIAS             | FINGERPRINT  | PUBLIC |              DESCRIPTION                | ARCHITECTURE |      TYPE       |   SIZE   |         UPLOAD DATE
        |
+------------------------------+--------------+--------+-----------------------------------------+--------------+-----------------+----------+------------------------
----------+
| debian/9 (7 more)            | 8f1de607377b | yes    | Debian stretch amd64 (20211017_05:24)   | x86_64       | CONTAINER       | 67.07MB  | Oct 17, 2021 at 12:0
0am (UTC) |
+------------------------------+--------------+--------+-----------------------------------------+--------------+-----------------+----------+------------------------
----------+
| debian/9 (7 more)            | 85855badeda1 | yes    | Debian stretch amd64 (20211017_05:24)   | x86_64       | VIRTUAL-MACHINE | 221.00MB | Oct 17, 2021 at 12:0
0am (UTC) |
+------------------------------+--------------+--------+-----------------------------------------+--------------+-----------------+----------+------------------------
----------+
| debian/9/arm64 (3 more)      | 21ffdbc1bc58 | yes    | Debian stretch arm64 (20211017_05:48)   | aarch64      | VIRTUAL-MACHINE | 206.81MB | Oct 17, 2021 at 12:0
0am (UTC) |
+------------------------------+--------------+--------+-----------------------------------------+--------------+-----------------+----------+------------------------
----------+
| debian/9/arm64 (3 more)      | e99d430ed6ec | yes    | Debian stretch arm64 (20211017_05:48)   | aarch64      | CONTAINER       | 63.15MB  | Oct 17, 2021 at 12:0
0am (UTC) |
+------------------------------+--------------+--------+-----------------------------------------+--------------+-----------------+----------+------------------------
```

Figure 6: Check Debian images.

3.3 Container management

- **Create a container**

1. Firstly we check for existing containers using command **lxc list** as figure 7 show the list is now empty since we haven't created any containers yet.

```
student@linux:~$ lxc list
+------+-------+------+------+------+-----------+----------+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS | LOCATION |
+------+-------+------+------+------+-----------+----------+
```

Figure 7: List container.

2. Then we create an ubuntu 16.04 container as figure 8 show start a new container from the ubuntu image server based on ubuntu 16.04 and name the new container ubuntu01 .

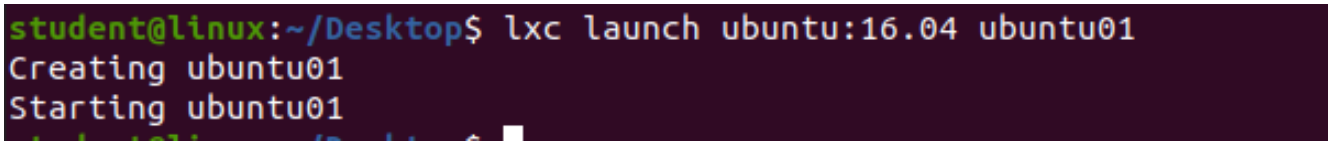

Figure 8: Create container.

3. We check existing containers again as figure 9 shows the container now is on the list.

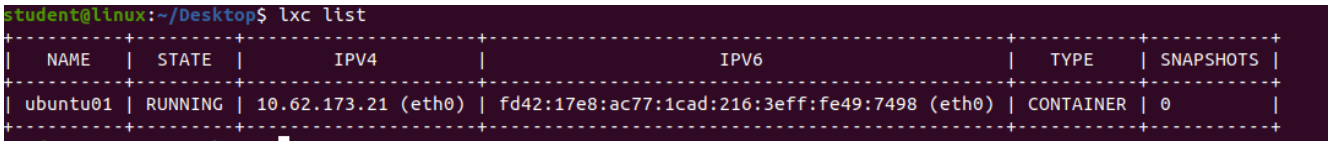

Figure 9: List container.

4. Then we restart, stop and start a container as figure 10 shows when stopping the container the statutes changed to stopped.

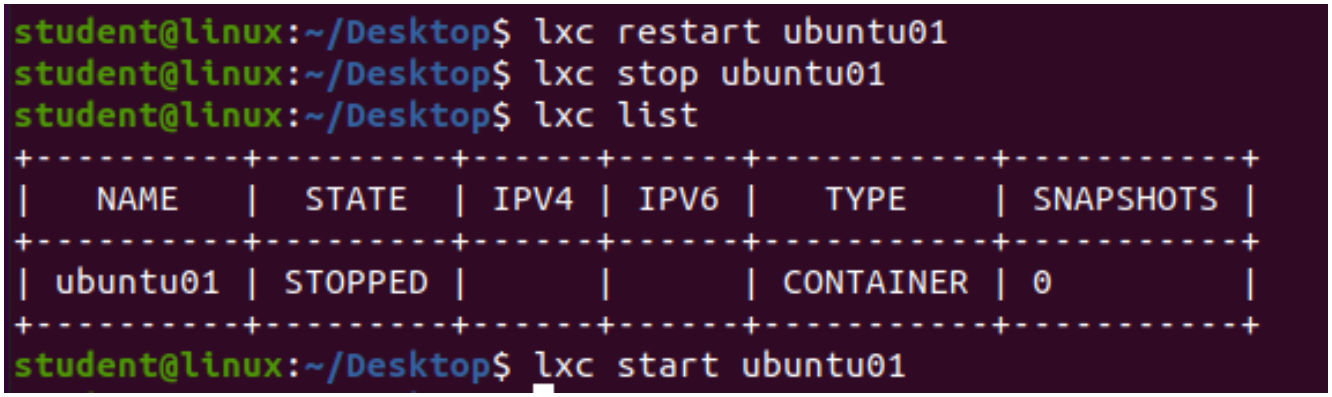

Figure 10: Restart,Stop,Start a container.

5. To delete a container we use delete command but first we must stop the container as figure 11 show .

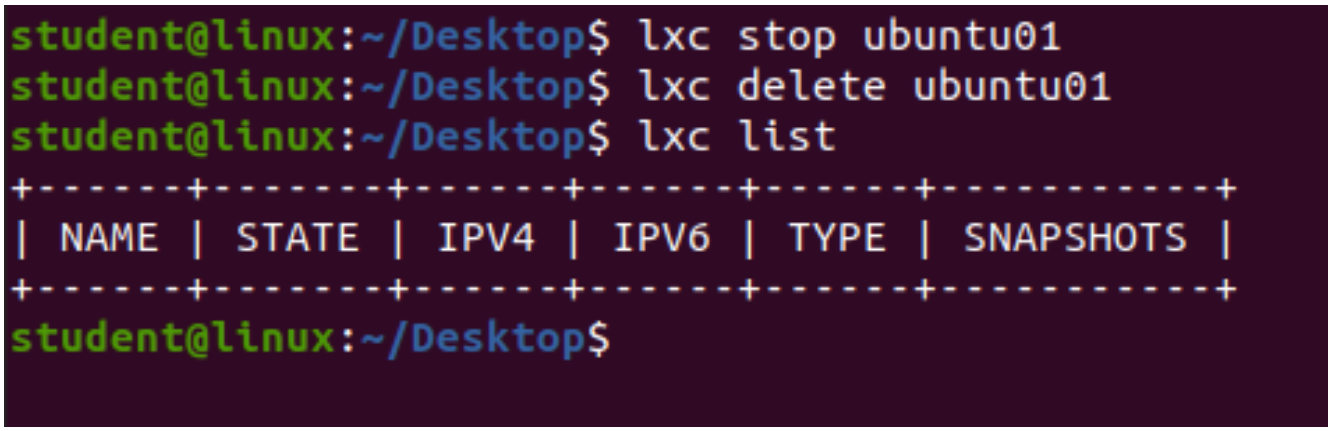

Figure 11: Deleting a container.

6. We recreate a new container then we execute the **nproc** command inside our container that lists the number of CPUs that the container sees as figure 12 shows there are 2 CPUs it depends on the number of CPUs has on our system because containers can see all system resources.

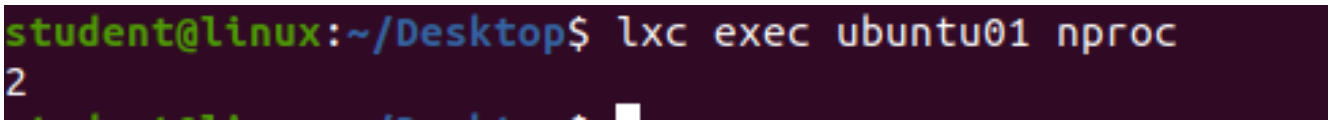

Figure 12: Number of CPU.

7. We use another way to execute command inside the container by open the bash as figure 13 show.

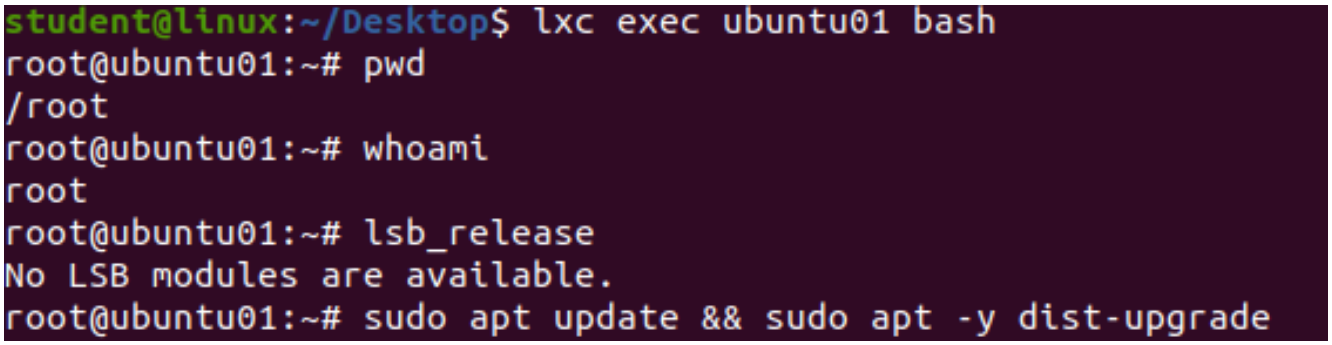

Figure 13: Number of CPU.

8. Then, we exit by pressing ctrl+d or typing exit, and we list the current containers to see his state (ubuntu01), as shown in figure 14. **Question:** Is it running or stopped? How would you compare docker containers to LXC containers now?

   The status for the LXC container will be running. If we press ctrl+d or type exit in docker, his state will be stoped, unlike the LXC.

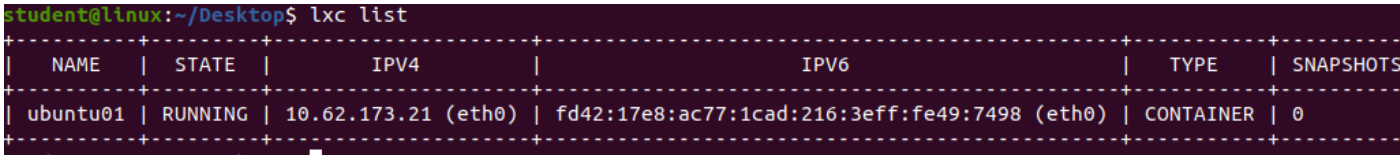

Figure 14: Container status.

3.4 Cloning a container

1. To replicate an existing container we use the following command as figure 15 show .

```
student@linux:~/Desktop$ lxc copy ubuntu01 ubuntu02
lxc list
student@linux:~/Desktop$ lxc list
+----------+---------+--------------------+----------------------------------------------+-----------+-----------+
|   NAME   |  STATE  |        IPV4        |                    IPV6                      |   TYPE    | SNAPSHOTS |
+----------+---------+--------------------+----------------------------------------------+-----------+-----------+
| ubuntu01 | RUNNING | 10.216.8.101 (eth0) | fd42:9158:f0be:7ccc:216:3eff:fe89:f0be (eth0) | CONTAINER | 0         |
+----------+---------+--------------------+----------------------------------------------+-----------+-----------+
| ubuntu02 | STOPPED |                    |                                              | CONTAINER | 0         |
+----------+---------+--------------------+----------------------------------------------+-----------+-----------+
```

Figure 15: Cloning a container.

2. Then we start the Ubuntu and check it statutes as figure 16 show, after that start a bash session and ping ubuntu01 as figure 17 show .

```
student@linux:~/Desktop$ lxc start ubuntu02
student@linux:~/Desktop$ lxc list
+----------+---------+--------------------+----------------------------------------------+-----------+-----------+
|   NAME   |  STATE  |        IPV4        |                    IPV6                      |   TYPE    | SNAPSHOTS |
+----------+---------+--------------------+----------------------------------------------+-----------+-----------+
| ubuntu01 | RUNNING | 10.216.8.101 (eth0) | fd42:9158:f0be:7ccc:216:3eff:fe89:f0be (eth0) | CONTAINER | 0         |
+----------+---------+--------------------+----------------------------------------------+-----------+-----------+
| ubuntu02 | RUNNING | 10.216.8.141 (eth0) | fd42:9158:f0be:7ccc:216:3eff:fe8b:a80f (eth0) | CONTAINER | 0         |
+----------+---------+--------------------+----------------------------------------------+-----------+-----------+
```

Figure 16: Start container.

```
student@linux:~/Desktop$ lxc exec ubuntu02 bash
root@ubuntu02:~# ping 10.216.8.101
PING 10.216.8.101 (10.216.8.101) 56(84) bytes of data.
64 bytes from 10.216.8.101: icmp_seq=1 ttl=64 time=0.092 ms
64 bytes from 10.216.8.101: icmp_seq=2 ttl=64 time=0.044 ms
^Z
[1]+  Stopped                 ping 10.216.8.101
root@ubuntu02:~#
```

Figure 17: Start a bash session and ping .

3. We login to the container using the ubuntu username as figure 18 show .

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

ubuntu@ubuntu01:~$
```

Figure 18: Switch container.

4. After, that we set the current user password to be (ubuntupass) as shown in 19. Then, we install OpenSSH-server as figure 20 shows. In the end, make an ssh connection from our host machine to the ubuntu container as shown in figure 21. **Note** as figure 21 show that the server sees the connection, but, we don't have the public key to make a connection (establish), for that we need to have the public key on both sides, we can send our key by using secure copy or by using a text file with USP.

```
ubuntu@ubuntu01:~$ sudo passwd
sudo: unable to resolve host ubuntu01: Connection timed out
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
ubuntu@ubuntu01:~$
```

Figure 19: Set password for container.

```
ubuntu@ubuntu01:~$ sudo apt install openssh-server
sudo: unable to resolve host ubuntu01: Connection timed out
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-server is already the newest version (1:7.2p2-4ubuntu2.10).
The following package was automatically installed and is no longer required:
  libfreetype6
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ubuntu@ubuntu01:~$
```

Figure 20: install Openssh server.

```
root@ubuntu02:~# ssh ubuntu01@10.72.68.184
The authenticity of host '10.72.68.184 (10.72.68.184)' can't be established.
ECDSA key fingerprint is SHA256:/UzQD8yQBnQgMtIwyFHknoSxTcLEiJYhwbByWlhM0OA.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.72.68.184' (ECDSA) to the list of known hosts.
Permission denied (publickey).
root@ubuntu02:~# ssh ubuntu01@10.72.68.184
Permission denied (publickey).
root@ubuntu02:~# ssh ubuntu01@10.72.68.184
Permission denied (publickey).
root@ubuntu02:~# exit
exit
```

Figure 21: make ssh connection from our host to container.

3.5 Rename a Container

1. To rename a container we used the move command as shown in figure 22. **Note** we cannot rename a running container so we stop it first.

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

Figure 22: rename container

3.6 File sharing

1. To share files with containers, we can use the LXC file command shown in figure 23. The **push** command, used for sharing a file with the container, then after removing the shared file, using the **pull** command to restore the removable file with a new name (Restoredfile). first.

```
student@linux:~$ touch shardefile
student@linux:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  shardefile  snap  Templates  Videos
student@linux:~$ lxc file push shardefile newubuntu/root/
student@linux:~$ lxc exec newubuntu ls
shardefile
student@linux:~$ rm shardefile
student@linux:~$ lxc file pull newubuntu/root/sharedfile restorfile
Error: Not Found
student@linux:~$ lxc file pull  newubuntu/root/shardefile restoredfile
student@linux:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  restoredfile  snap  Templates  Videos
student@linux:~$ rm shardefile
```

Figure 23: File sharing & restoring

3.7 Snapshots

1. Snapshot means that if we need to save the state of our existing app running on LXD. We will use it as shown in figure 24, so now if we want to remove those files that are we create or something happened for this container as shown in figure 25, we can restore it by using the snapshot container.

```
student@linux:~$ lxc exec newubuntu bash
root@newubuntu:~# ls
shardefile
root@newubuntu:~# mkdir test
root@newubuntu:~# mkdir test
mkdir: cannot create directory ‘test’: File exists
root@newubuntu:~# mkdir test2
root@newubuntu:~# mkdir test3
root@newubuntu:~# mkdir test4
root@newubuntu:~# exit
exit
student@linux:~$ lxc snapshot newubuntu snap01
student@linux:~$ lxc list
+-----------+---------+---------------------+----------------------------------------------+-----------+-----------+
|   NAME    |  STATE  |        IPV4         |                     IPV6                     |   TYPE    | SNAPSHOTS |
+-----------+---------+---------------------+----------------------------------------------+-----------+-----------+
| newubuntu | RUNNING | 10.72.68.149 (eth0) | fd42:7ec2:ea59:b7a4:216:3eff:fe53:41e9 (eth0) | CONTAINER | 1         |
+-----------+---------+---------------------+----------------------------------------------+-----------+-----------+
| ubuntu02  | RUNNING | 10.72.68.184 (eth0) | fd42:7ec2:ea59:b7a4:216:3eff:feb8:f017 (eth0) | CONTAINER | 0         |
+-----------+---------+---------------------+----------------------------------------------+-----------+-----------+
student@linux:~$
```

Figure 24: create snapshot

```
student@linux:~$ lxc exec newubuntu bash
root@newubuntu:~# ls
shardefile  test  test2  test3  test4
root@newubuntu:~# rm test test2 test3 test4
rm: cannot remove 'test': Is a directory
rm: cannot remove 'test2': Is a directory
rm: cannot remove 'test3': Is a directory
rm: cannot remove 'test4': Is a directory
root@newubuntu:~# rmdir test test2 test3 test4
root@newubuntu:~# ls
shardefile
root@newubuntu:~# exit
exit
student@linux:~$ lxc restore newubuntu snap01
student@linux:~$ lxc exec newubuntu bash
root@ubuntu01:~# ls
shardefile  test  test2  test3  test4
root@ubuntu01:~#
```

Figure 25: Restore snapshot

3.8 Setting limits

1. All host hardware is exposed to the container, which means every container sees that all resources are available for him to consume as it needs. That's is not good, for that we are setting some limitations. In first we login to newubuntu and checked the number of CPUs and memory size available to the container as shown in figure 26

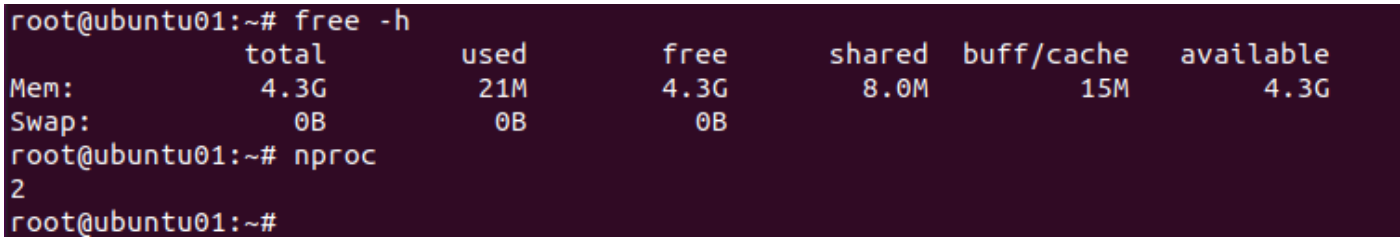

Figure 26: # of Cpu & memory size

2. After that, we will limit those two things as shown in figure 27. Then we will check again to ensure and verify of limitation, so as figure 28 shows that the CPU now is 1, and the memory size is 488Mb.

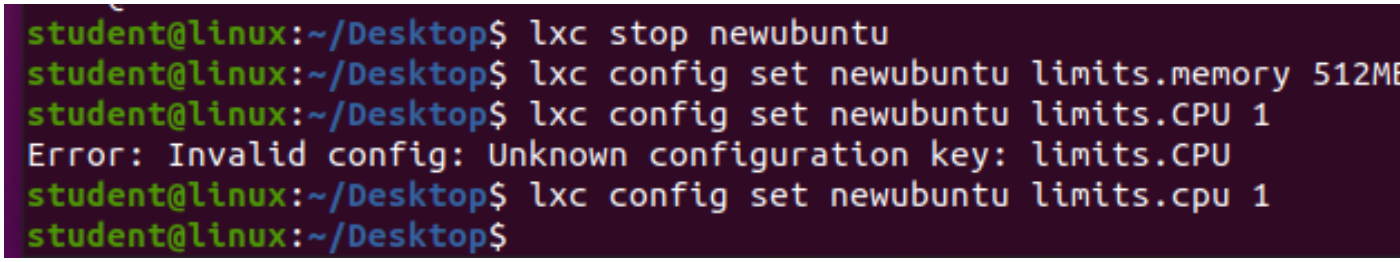

Figure 27: Limitation # of Cpu & memory size

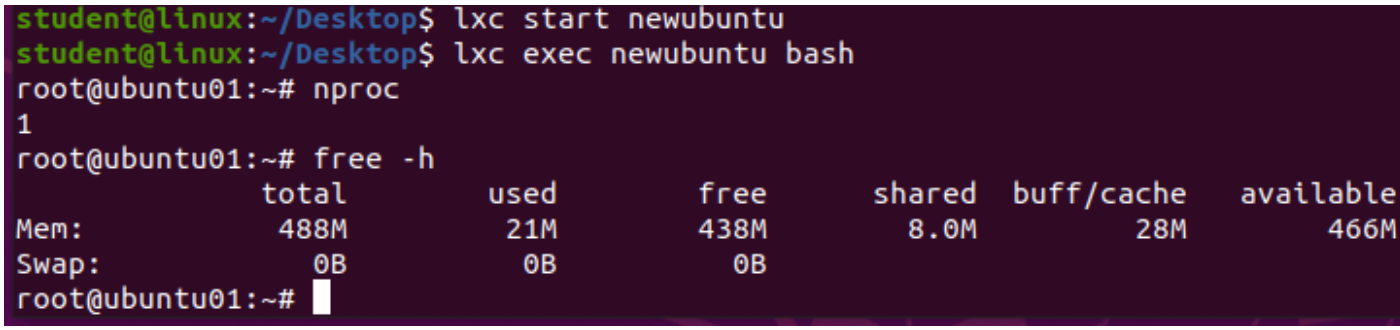

Figure 28: verify # of Cpu & memory size

3.9 Networking

1. At first we install apache2 on ubuntu02 container as figure 29 show.

```
student@linux:~/Desktop$ lxc exec ubuntu02 bash
root@ubuntu02:~# apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Figure 29: Install apache2

2. Then from the lxcserver machine we execute curl http://[ubuntu02_IP]. Which will display the html code for the apache, as shown in figure 30.

```
root@ubuntu02:~# curl http://10.216.8.141

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2014-03-19
    See: https://launchpad.net/bugs/1288690
```

Figure 30: Curl command

3. Then, if we want to access the apache page from the same VM browse, it will show the apache page. But, if we access it from another machine in the same network, it will not open because we will need port forward as shown in figure 31, this command in the figures 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. Figure 32 also shows the apache page for the second VM after using port forward.

```
student@linux:~$ lxc config device add ubuntu02 frwrdport proxy listen=tcp:0.0.0.0:80 connect=tcp:127.0.0.1:80
Device frwrdport added to ubuntu02
```

Figure 31: Port-forward command

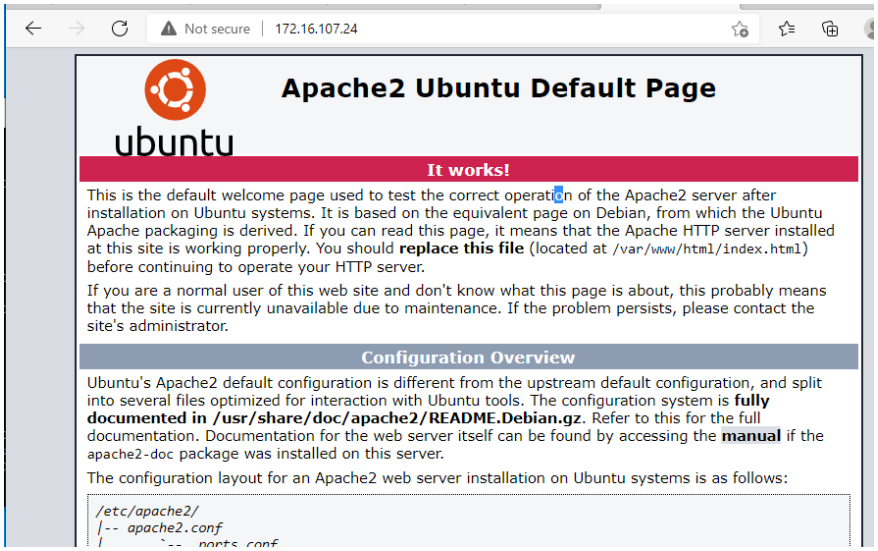

Figure 32: Apache website

3.10 Clean Up

Now, we stop and delete all containers as shown in figure 33.

```
student@linux:~$ lxc stop newubuntu
student@linux:~$ lxc rm newubuntu
student@linux:~$ lxc stop ubuntu02
student@linux:~$ lxc rm ubuntu02
student@linux:~$ lxc list
+------+-------+------+------+------+-----------+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+------+-------+------+------+------+-----------+
student@linux:~$
```

Figure 33: Remove and delete

3.11 To Do

1. In first, we create a container based on ubuntu 14.04 release and install and configure ssh server, as shown in figure 34

```
student@linux:~$ lxc launch ubuntu:14.04 ubuntu01
Creating ubuntu01
Retrieving image: rootfs: 1% (1.08MB/s)
Retrieving image: Unpack: 100% (2.97GB/s)
Starting ubuntu01
student@linux:~$
student@linux:~$
student@linux:~$ lxc exec ubuntu01 bash
root@ubuntu01:~# apt install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-server is already the newest version.
The following packages were automatically installed and are no longer required:
  libfreetype6 os-prober
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@ubuntu01:~# exit
exit
```

Figure 34: create container install ssh

2. Then, we configure port forwarding so that we can ssh into the container from our windows machine using port 5000 as shown in figure 35. Figure 36 shows that we try to make an ssh connection to that port, as figure 13 show that the server sees the connection, however, we don't have the public key to make a connection (establish), for that we need to have the public key in both side, we can send our key by using secure copy or by using a text file with USP.

```
student@linux:~$ lxc config device add ubuntu01 frwrdport proxy listen=tcp:0.0.0.0:5000 connect=tcp:127.0.0.1:22
Device frwrdport added to ubuntu01
student@linux:~$ lxc list
+----------+---------+---------------------+----------------------------------------------+-----------+-----------+
|   NAME   |  STATE  |        IPV4         |                     IPV6                     |   TYPE    | SNAPSHOTS |
+----------+---------+---------------------+----------------------------------------------+-----------+-----------+
| ubuntu01 | RUNNING | 10.72.68.152 (eth0) | fd42:7ec2:ea59:b7a4:216:3eff:fe6c:ea95 (eth0) | CONTAINER | 0         |
+----------+---------+---------------------+----------------------------------------------+-----------+-----------+
student@linux:~$ lxc exec ubuntu01 bash
root@ubuntu01:~# sudo passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@ubuntu01:~# exit
exit
student@linux:~$
```

Figure 35: Port-forward for SSH

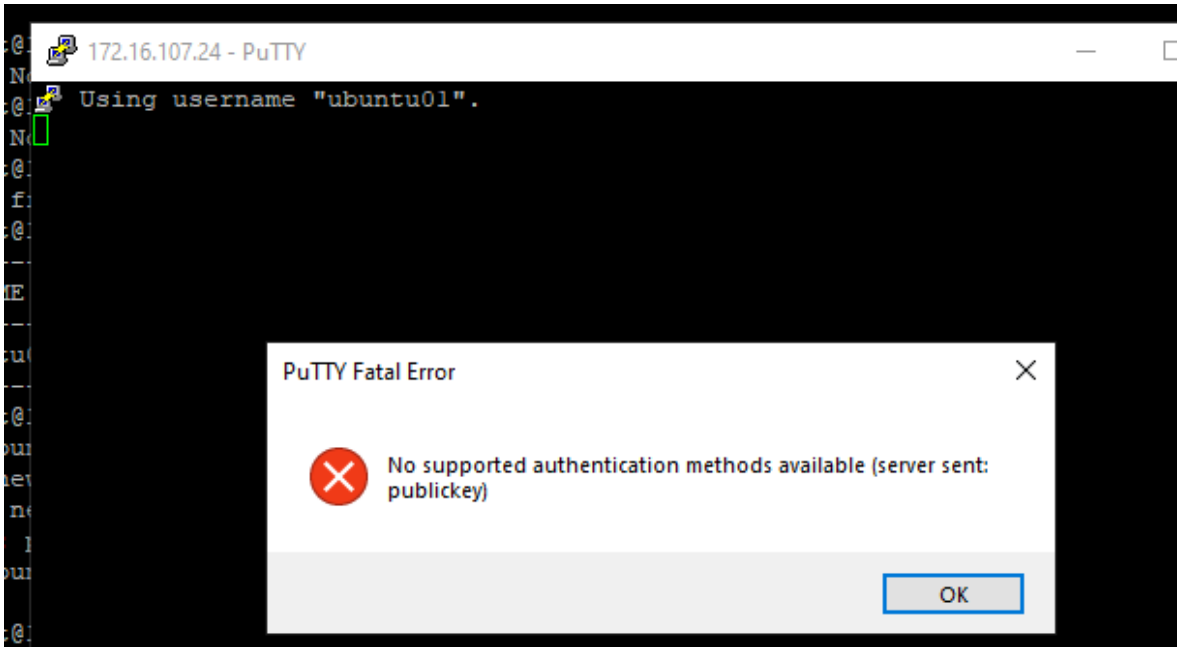

Figure 36: ssh connection in port 5000

4 Conclusion

LXC offers the advantages of a VE on Linux, most notably the ability to isolate your Linux OS from one another. It's a cheaper and faster way to build compared to a VM, but it demands additional Linux knowledge and experience. On the other hand, LXC's capabilities have been significantly enhanced by Docker. Docker is garnering a rising user base due to its evident benefits and capacity to share and duplicate any Docker-created packages. It is taking the lead above VMs and VEs.

5 Reference

https://www.section.io/engineering-education/lxc-vs-docker-what-is-the-difference-and-why-docker-is-better/
https://en.wikipedia.org/wiki/LXC
https://linuxcontainers.org/lxd/introduction/