= Session-(4.3) =

Create LXC Container Inside ZFS Storage

As we are going to work with **LXC**, so we have to remove **LXD** if it is already installed.

Find the LXD and related packages & remove them;

```
dpkg -l |grep lxd
apt remove -y --purge lxd lxd-client
```

Now install LXC

```
apt install -y lxc lxc-templates
```

We have already installed zfs and created our zfs pool in some previous sessions, so now we will use that zfs pool (named **vol1)** as lxc back-end storage.

```
zpool list
root@group1-node3:~# zpool list
NAME    SIZE  ALLOC   FREE  EXPANDSZ   FRAG    CAP  DEDUP  HEALTH  ALTROOT
vol1   19.9G   952K  19.9G         -     0%     0%  1.00x  ONLINE  -
```

Create your first container

```
lxc-create -n group1_n3_ct1 -t download -- --dist ubuntu --release bionic --arch amd64
```

Here, you just created an Ubuntu container (release=bionic/18.04, arch=amd64, variant=default)

Note: If it takes long-time (it may take long-time if Internet speed is bellow 2mbps/User**), then use our offline version of LXC ubuntu-18.04-image. Follow the instructions in the next page...**

```
cd /opt/

wget -c http://192.168.108.8/iso/groupX_nY_ctZ.tar.gz
tar zxvf /opt/groupX_nY_ctZ.tar.gz -C /var/lib/lxc
mv /var/lib/lxc/groupX_nY_ctZ /var/lib/lxc/group1_n3_ct1
sed -i 's/groupX_nY_ctZ/group1_n3_ct1/g' /var/lib/lxc/group1_n3_ct1/config
```

If you want to create a CentOS Container **(Optional)**

```
apt install librpm3 librpmbuild3 librpmio3 libsqlite0 python-rpm python-sqlite \
python-sqlitecachec python-urlgrabber rpm rpm-common rpm2cpio yum debootstrap

lxc-create -n centos_n1_ct1 -t centos -- -R 7 -a x86_64
```

To see the list of created containers

```
lxc-ls --fancy
```

Start and login into the created container

```
lxc-start  -n group1_n3_ct1 -d
lxc-attach -n group1_n3_ct1
```

For security reason, container images ship without user accounts and without a root password.

Inside Container, Install **openssh-server**, remove default user & activate root password and set permit root login to yes.

```
apt update
```

Also set local apt-cache-mirror,

```
sudo sed -i 's/archive.ubuntu.com/mirror.amberit.com.bd/g' /etc/apt/sources.list
sudo echo 'Acquire::http { Proxy "http://192.168.108.8:4444"; };' > /etc/apt/apt.conf.d/50apt-cacher

sudo apt update
sudo apt remove netplan.io
sudo apt install openssh-server ifupdown vim resolvconf net-tools

sudo vim /etc/ssh/sshd_config          ; configure ssh as you did it before
sudo /etc/init.d/ssh restart           ; restart ssh service
sudo passwd root                       ; set root password
sudo userdel -r ubuntu                 ; remove default user

sudo sed -i 's/groupX_nY_ctZ/group1_n3_ct1/g' /etc/hostname     ; change hostname as your own
sudo reboot
```

Now we will configure the **lxc-container** with zfs storage backend.

Option # 01 Hardway

```
lxc-ls --fancy
lxc-stop -n group1_n3_ct1
lxc-copy -n group1_n3_ct1 -N group1_n3_ct1_zfs

zfs create vol1/lxc/group1_n3_ct1_zfs
rsync -av /var/lib/lxc/group1_n3_ct1_zfs/ /vol1/lxc/group1_n3_ct1_zfs

rm -fr /var/lib/lxc/group1_n3_ct1_zfs/
ln -s /vol1/lxc/group1_n3_ct1_zfs /var/lib/lxc/
echo 'group1_n3_ct1_zfs' > /var/lib/lxc/group1_n3_ct1_zfs/rootfs/etc/hostname
```

Option # 02 Easy Way

Let’s create a shell script to complete the manual job in a easy way.…

```
vim /usr/bin/lxc-zfs-clone
```

```
#!/bin/bash
echo -e "\nSOURCE CONTAINER NAME : ";
read nameO;
echo -e "\nNEW CONTAINER NAME : ";
read nameN;
echo -e "\nStarting the cloning process... time depends on container size... ";
lxc-copy -n $nameO -N $nameN;

sleep 3
zfs create vol1/lxc/$nameN;

echo -e "\nConverting the container to zfs container... ";
rsync -av /var/lib/lxc/$nameN/ /vol1/lxc/$nameN ;

sleep 2
rm -fr /var/lib/lxc/$nameN;

sleep 2
ln -s /vol1/lxc/$nameN /var/lib/lxc/
echo "$nameN" > /var/lib/lxc/$nameN/rootfs/etc/hostname
lxc-ls --fancy

echo -e "\nALL DONE... ";
```

Save+Exit

```
chmod +x /usr/bin/lxc-zfs-clone
```

Now run the script and follow the onsreen instruction…

```
sudo lxc-zfs-clone
```

Prepare the networking; we will use previous bridge interface **bridge0**, which was created with **OVS** in some previous session.

```
vim /etc/network/bridge0.up
```

```
#!/bin/bash
BRIDGE="bridge0"
ovs-vsctl --may-exist add-br $BRIDGE
ovs-vsctl --if-exists del-port $BRIDGE $5
ovs-vsctl --may-exist add-port $BRIDGE $5
```

```
vim /etc/network/bridge0.dn
#!/bin/bash
ovsBr=bridge0
ovs-vsctl --if-exists del-port ${obsBr} $5
```

```
chmod +x /etc/network/bridge0.*
```

Now, add the bridge interface inside the container configuration file

```
vim /var/lib/lxc/group1_n3_ct1_zfs/config

#lxc.net.0.link = lxcbr0; comment-out/disable this line
lxc.net.0.script.up = /etc/network/bridge0.up; add this line
lxc.net.0.script.down = /etc/network/bridge0.dn; add this line
```

Start the container, login and put IP, DNS (as given by instructor) and try to get Internet

```
lxc-start -n group1_n3_ct1_zfs -d
lxc-ls –fancy

lxc-attach -n group1_n3_ct1_zfs

vim /etc/network/interfaces                    ; add the following lines
```

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
     address 192.168.108.xxx
     netmask 255.255.255.0
     gateway 192.168.108.1
```

Save+Exit

Now restart the network service….

```
/etc/init.d/networking restart
```

```
vim /etc/resolvconf/resolv.conf.d/head          ; put your nameserver IP
nameserver 192.168.108.1
```

```
service resolvconf restart
reboot
```

After reboot, check from the container that you are getting Internet.

```
ping google.com
```