LXD VM Containers

Dr. **Jose L. Muñoz Tapia**
Information Security Group (ISG)
Universitat Politècnica de Catalunya (UPC)

LXD

Outline

LXD

Concept

Storage

Basics

Network

Mounts

Images

Remotes

Migrations

Profiles

Command Summary

Concept

- LXD is a next generation system container manager.
- It offers a user experience similar to virtual machines but using Linux containers instead.
- Can take advantage of copy on write filesystems.
- LXD uses ZFS by default (but there are other storage options).
- In particular, LXD uses ZFS filesystems for:
  - Images.
  - Snapshots.
  - Containers.

Outline

LXD

- Concept
- Storage
- Basics
- Network
- Mounts
- Images
- Remotes
- Migrations
- Profiles
- Command Summary

LXD and ZFS

- By default, if installed with snap, LXD creates a pool in a file (loop backed ZFS) in:

  `/var/snap/lxd/common/lxd/disks/default.img`

- The default zpool defaults to:
  - 20% of the size of the partition behind /var/lib/lxd.
  - With a minimum size of 15GB.
  - And maximum size of 100GB.
- The value that's selected for your system is pretty clearly shown during "lxd init".

Production Storage

- ZFS on file-backed storage:
  - Has a rather poor performance.
  - In addition, when using a file-backed storage, if we fill the available space ZFS may crash and cause data loss.
- For production environments:
  - You should really be using a dedicated partition or disk for ZFS.
  - When initializing you can also configure your custom ZFS pool.
  - To create a ZFS pool:

```
# zpool create lxdcontainers /dev/sda1 /dev/sda2
# zpool status -v
# zpool list
```

  - Later you can add more partitions to the pool too:

```
# zpool add lxdcontainers /dev/sdb1
```

LXD with ZFS (*)

- Create a ZFS dataset:

```
# zfs create mypool/mydataset
# zfs list
```

- To create a new pool called ”pool1” on /dev/sdX (the ZFS Zpool will also be called ”pool1”):

```
$ lxc storage create pool1 zfs source=/dev/sdX
```

- To create a new pool on /dev/sdX with the ZFS Zpool called ”my-tank”:

```
$ lxc storage create pool1 zfs source=/dev/sdX zfs.pool_name=my-tank
```

- To use the existing ZFS Zpool ”my-tank”:

```
$ lxc storage create pool1 zfs source=my-tank
```

- To use the existing ZFS dataset ”my-tank/slice”:

```
$ lxc storage create pool1 zfs source=my-tank/slice
```

Outline

LXD

- Concept
- Storage
- Basics
- Network
- Mounts
- Images
- Remotes
- Migrations
- Profiles
- Command Summary

Install

- We can install lxd with snap:

```
$ sudo snap install lxd
$ sudo snap list
$ sudo snap restart lxd
```

- Access control for LXD is based on group membership:
  - The root user as well as members of the ”lxd” group can interact with the local daemon.
  - If the ”lxd” group is missing on your system, create it, then restart the LXD daemon.
  - You can then add trusted users to it.
  - Anyone added to this group will have full control over LXD.
  - Add your non-root Unix user to the lxd group:

```
$ sudo usermod -a -G lxd username
```

Init

Before you can create containers, you need to tell LXD a little bit about your storage and network needs:

```
$ lxd init

Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (btrfs, ceph, dir, lvm, zfs) [default=zfs]:
Create a new ZFS pool? (yes/no) [default=yes]:
Would you like to use an existing block device? (yes/no) [default=no]:
Size in GB of the new loop device (1GB minimum) [default=15GB]:
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, “”auto or “”none) [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, “”auto or “”none) [default=auto]:
Would you like LXD to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
```

Configuration

Configuration files are in `/var/lib/lxd` and `/var/snap/lxd/common/lxd`.

You can get information about storage with:

```
$ lxc storage info default
```

Create a Container

- Creating your first container is as simple as[1]:

```
$ lxc launch ubuntu:18.04 first
```

- Check that the container is created with:

```
$ lxc list
```

- Your container here is called ”first” but you also could let LXD give it a random name (by just executing without a name):

```
$ lxc launch ubuntu:18.04
```

[1] You can also try LXD containers online in `https://linuxcontainers.org/lxd/try-it`

Basic Usage i

- Shell and commands:
  - We can get an interactive shell in a running container with:

```
$ lxc exec first bash
```

  - Or with the equivalent command line:

```
$ lxc exec first -- /bin/bash
```

  - The previous can be also used to execute any command:

```
$ lxc exec first -- apt-get update
```

  - To exec a command with another user:

```
$ lxc exec webserver -- sudo --login --user myuser ls -la
```

Basic Usage ii

- Pull/push files:
  - To pull a file from the container:

```
$ lxc file pull first/etc/hosts .
```

  - To push a file to the container:

```
$ lxc file push hosts first/tmp/
```

- Stop/remove containers:
  - To stop the container:

```
$ lxc stop first
$ lxc info mycontainer --show-log
```

  - And to remove it entirely:

```
$ lxc delete first
```

Outline

LXD

- Concept
- Storage
- Basics
- Network
- Mounts
- Images
- Remotes
- Migrations
- Profiles
- Command Summary

Container Names from Host

- We can edit the network configuration as follows:

```
# lxc network edit lxdbr0
```

- We can get the IP address of the LXD gateway:

```
# lxc network get lxdbr0 ipv4.address
```

- To resolve names xxxx.lxd in the system we create a service in /etc/systemd/network/lxd.network

```
[Match]
Name=lxdbr0

[Network]
DNS=10.168.133.1
Domains=~lxd

[Address]
Address=10.168.133.1/24
Gateway=10.168.133.1
```

- Then, we enable and activate the service:

```
# systemctl enable systemd-networkd
# systemctl restart systemd-networkd
```

Port Forwarding with Proxy Devices

- In LXD, we can create a proxy device to do port forwarding:
  - LXD spawns a process (`forkproxy`) that does the proxying.
  - This proxy is not any special of LXD (works like any network proxy).
- You can run the following command on the host to create proxies:

```
$ lxc config device add c1 myport80 proxy listen=tcp:0.0.0.0:80 connect=tcp:10.168.133.7:80
```

  - The name of the container is `c1`.
  - The name for the proxy device (`myport80`) is arbitrary.
  - In the previous commands we set it up to listen on all (0.0.0.0) network interfaces on the host.
- We can list devices with:

```
$ lxc config device show c1
```

- To remove a proxy device:

```
$ lxc config device remove mycontainer myport80
```

Networks for LXDs i

- We can create LXD networks.
- An example is provided in the following profile:

```
$ lxc profile show privatepublicnetwork
config:
  user.network-config: |
    version: 1
    config:
      - type: physical
        name: eth0
        subnets:
          - type: dhcp
            ipv4: true
      - type: physical
        name: eth1
        subnets:
          - type: dhcp
            ipv4: true
description: LXD profile with private and public networks
devices:
  ...
```

Networks for LXDs ii

```
devices:
  eth0:
    name: eth0
    nictype: bridged
    parent: lxdbr0
    type: nic
  eth1:
    name: eth1
    nictype: macvlan
    parent: enp16s0
    type: nic
  root:
    path: /
    pool: default
    type: disk
name: privatepublicnetwork
used_by:
```

- The previous configuration uses ”cloud-init” which is a method for cloud instance initialization.
- The `user.network-config` under the `config:` section are cloud-init instructions that get passed verbatim to the container.

Networks for LXDs iii

- We specify `version: 1` so it works even on Ubuntu 16.04.
- Another example:

```
config:
  user.network-config: |
    version: 1
    config:
      - type: physical
        name: eth1
        subnets:
          - type: static
            ipv4: true
            address: 10.10.101.20
            netmask: 255.255.255.0
            gateway: 10.10.101.1
            control: auto
      - type: nameserver
        address: 10.10.10.254
```

- You can set the configuration from a file:

```
$ lxc config set c1 user.network-config - < myconfig.yml
```

Outline

LXD

- Concept
- Storage
- Basics
- Network
- Mounts
- Images
- Remotes
- Migrations
- Profiles
- Command Summary

Shared Host Directory (RO)

- To mount read-only (RO) the host's `/wwwdata/` directory onto `/var/www/html/` in the LXD container named `c1`, run:

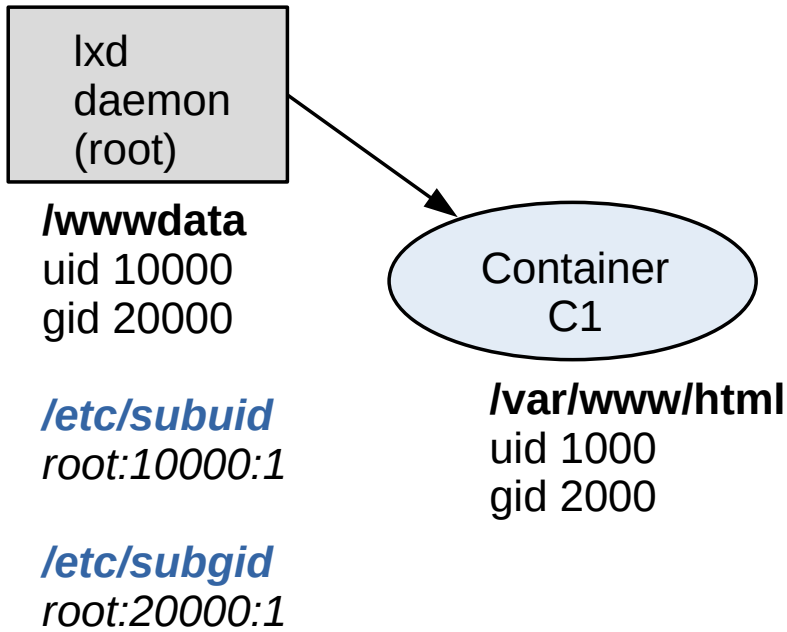
```
$ lxc config device add c1 sharedwww disk source=/wwwdata/ path=/var/www/html/
$ lxc config show c1
```

- If the directory in the container does not exist, it will be created automatically.

UID/GID Remaps

- To mount a directory read-write we have to take into account that LXD uses linux namespaces to isolate processes.
- But, by default, **even root is not allowed to remap UIDs/GIDs from the host inside containers**.

- To enable shared read-write access to folders we have to allow the LXD daemon to remap UIDs from the host to the container.
- The LXD daemon runs as user root.
- The permission to remap a UID in the host (e.g. 10000) is granted in the file /etc/subuid.



root: user allowed to remap
10000: Numerical subordinate ID
1: range or UID that can be remapped

Using the Remaps with LXD

- We have to explicitly allow the LXD daemon (which runs as root) to remap the desired *UID* and *GID*.
- Example for $UID = 10000$ and $GID = 20000$:

```
$ id
uid=10000(myuser) gid=20000(myuser)
groups=20000(myuser),4(adm),24(cdrom),27(sudo)...
```

- The permission is granted as follows:

```
# echo "root:10000:1" >> /etc/subuid
# echo "root:20000:1" >> /etc/subgid
```

  The previous needs to be done only once.

- The directory that you want to grant access **must belong to the previous *UID*/*GID* (10000/20000) in the hypervisor**.

Shared Host Directory (RW)

- Once the root user is allowed to remap a certain *UID* and *GID*, we need to tell the LXD daemon what to remap inside each container.
- For example:

```
# echo -en "uid 10000 1000\ngid 20000 2000" | lxc config set first raw.idmap -
```

- With the previous commmand we are specifying:

  $UID_{host} = 10000 \rightarrow UID_{container} = 1000$

  $GID_{host} = 20000 \rightarrow GID_{container} = 2000$
- You can do multiple mappings of several *UID*s and *GID*s separating them with "\n".
- We need to restart to make the mappings effective:

```
$ lxc restart c1
```

- And finally create the corresponding disk device:

```
$ lxc config device add c1 myhomedir disk source=/wwwdata path=/var/www/html
```

Shared Host Directory RW with Privileged Container (*)

Another option (instead of using uid/gid mappings) is to use a privileged container:

```
$ lxc launch ubuntu privilegedContainerName -c security.privileged=true
$ lxc config device add privilegedContainerName shareName disk source=/wwwdata path=/var/www/html
```

If your container already exists:

```
$ lxc config set privilegedContainerName security.privileged true
```

Outline

LXD

- Concept
- Storage
- Basics
- Network
- Mounts
- Images
- Remotes
- Migrations
- Profiles
- Command Summary

Images i

- In LXD, containers are created from images.
- Images are either:
  1. Generated from an existing container.
  2. Downloaded from an image server.
- The image has a **unique identifier which is the hash** (SHA256) of its representation (as a compressed tarball or for split images, the concatenation of the metadata and rootfs tarballs).
- Images can be referenced by:
  1. Their full hash.
  2. A shortest unique partial hash.
  3. An alias name (if one is set).
- For the LXD snap, images are expected to be in: `/var/snap/lxd/common/lxd/images`

Images ii

- Images are accessible from that path at any time (it's not temporary storage).
- You can check your locally available images for containers:

```
$ lxc image list
```

- You can list available container images from a remote repository (in the example is the ubuntu repository):

```
$ lxc image list ubuntu:

+-------------+--------------+--------+-----------------+--------+------+------------------+
| ALIAS       | FINGERPRINT  | PUBLIC |   DESCRIPTION   |  ARCH  |   SIZE   |  UPLOAD DATE |
+-------------+--------------+--------+-----------------+--------+----------+--------------+
| b (11 more) | 5b72cf46f628 | yes    | ubuntu 18.04... | x86_64 | 180.37MB | Apr 24, 2019 |
+-------------+--------------+--------+-----------------+--------+----------+--------------+
| c (5 more)  | 4716703f04fc | yes    | ubuntu 18.10... | x86_64 | 313.29MB | Apr 2, 2019  |
+-------------+--------------+--------+-----------------+--------+----------+--------------+
....
```

Images iii

- Note:
  - The first two columns for the alias and fingerprint provide an identifier that can be used to specify the container image when launching it.
  - The output snippet shows the container images Ubuntu versions 18.04 LTS, 18.10, etc.
  - When creating a container we can just specify the short alias.
  - For example, ubuntu:b means that the repository is ubuntu and the container image has the short alias b (for bionic, the codename of Ubuntu 18.04 LTS).

Images iv

- To get more information about a particular image:

```
$ lxc image info ubuntu:b
Fingerprint: 5b72cf46f628b3d60f5d99af48633539b2916993c80fc5a2323d7d841f66afbe
Size: 180.37MB
Architecture: x86_64
Public: yes
Timestamps:
    Created: 2019/04/24 00:00 UTC
    Uploaded: 2019/04/24 00:00 UTC
    Expires: 2023/04/26 00:00 UTC
    Last used: never
Properties:
    release: bionic
    version: 18.04
    architecture: amd64
    label: release
    serial: 20190424
    description: ubuntu 18.04 LTS amd64 (release) (20190424)
    os: ubuntu
Aliases:
    - 18.04
    - 18.04/amd64
    - b
    - ...
Cached: no
Auto update: disabled
```

Image Auto Update i

- When using remote images, LXD will:
  1. Automatically cache images for you.
  2. Remove them upon expiration.
- LXD can keep images up to date:
  - By default, any image which comes from a remote server and was requested through an alias will be automatically updated by LXD.
  - This can be achieved with `images.auto_update_cached`.
  - Every 6 hours (unless `images.auto_update_interval` is set), the LXD daemon will go look for more recent version of all the images in the store which are marked as auto-update and have a recorded source server.
- When a new image is found:
  - It is downloaded into the image store.
  - The aliases pointing to the old image are moved to the new one.
  - The old image is removed from the store.

Image Auto Update ii

- You can enable auto-update on a per-image basis by editing the config:

```
$ lxc image edit <img>
```

- Set `auto_update: true` as a top-level item in the config.
- The user can also request a particular image be kept up to date when manually copying an image from a remote server.
- Note about creation of new containers and image updates:
  - Consider that a new upstream image update is published.
  - The local LXD has the previous image in its cache.
  - Next, the user requests a new container to be created from this image.
  - Then, LXD will use the previous version of the image rather than delay the container creation.
  - This behavior only happens if the current image is scheduled to be auto-updated and can be disabled by setting `images.auto_update_interval` to 0.

Containers of Old Images

Note about images used in previous containers:

- Due to the way copy-on-write works in ZFS, parent filesystems can't be removed until all children are gone.
- As a result, LXD will automatically rename any removed but still referenced object to a random deleted/ path and keep it until such time the references are gone and it can safely be removed.
- Having auto update means that new containers will depend on the updated image.
- But old ones still point to the old image.

Creating an Image i

- To create our image, first we create a snapshot from a container:

```
$ lxc snapshot mycont mysnapshot
$ lxc list mycont
+--------+---------+ // ... +-----------+
| NAME   | STATE   | // ... | SNAPSHOTS |
+--------+---------+ // ... +-----------+
| mycont | RUNNING | // ... | 1         |
+--------+---------+ // ... +-----------+
```

- Next, we create an image from ”mysnapshot” (publish)

```
$ lxc publish mycont/myshapshot --alias myimage

container published with fingerprint:
a1fa8864fe0aa0c3cf395fd647d55b5b5292cfce932073782976389a6fee9612
```

- We can export an image to a tarball:

```
$ lxc image export myimage
Image exported successfully!
```

Creating an Image ii

- We can see the exported image:

```
$ ls -alht a1fa8864fe0aa0c3cf395fd647d55b5b5292cf*
```

- You can import the image tarball:

```
$ lxc image import a1fa8864fe0aa0c3cf395fd647d5...612.tar.gz --alias myimage
Image imported fingerprint: a1fa8864fe0aa0c3cf395fd647d55b5b5292cfce932073782976389a6fee9612

$ lxc image list
```

- The data and the metadata are kept when you restore the image.
- Finally we can start a container from the image:

```
$ lxc launch myimage myvm
```

SNAP Applications

- To install snaps inside the container we first update and upgrade the system:

```
container$ sudo apt install update && apt upgrade -y
```

- Then, install the packages squashfuse and fuse:

```
container$ sudo apt install squashfuse fuse
```

- Finally, install **snapd**:

```
container$ sudo apt install snapd
```

- Now you can install **snap** packages as always:

```
container$ sudo snap install YOUR_PACKAGE
```

Outline

LXD

- Concept
- Storage
- Basics
- Network
- Mounts
- Images
- Remotes
- Migrations
- Profiles
- Command Summary

Remotes

- We can see the current daemon configuration with:

```
$ lxc config show
```

- To activate remote access:

```
lxc config set core.https_address "[::]:8443"
```

- To disable remote access delete the previous line, which makes it listen again only to localhost.
- The remote command:

```
lxc remote [command]

Available Commands:
  add         Add new remote servers
  get-default Show the default remote
  list        List the available remotes
  remove      Remove remotes
  rename      Rename remotes
  set-url     Set the URL for the remote
  switch      Switch the default remote
```

Remotes & HTTPS i

- By default, LXD has no password for security reasons.
- To set a password for HTTPS on the host where LXD is running:

```
localclient$ lxc config set core.trust_password "mysupersecret"
```

- This will set the remote password that you can then use to do lxc remote add:

```
localclient$ lxc remote add remotename @IP --password=mysupersecret
```

- You can also access the server without setting a password by copying the client certificate from `snap/lxd/current/.config/lxc/client.crt`.
- To create this certificate, try to add a repository in a client and the command will autogenerate the certificate.

Remotes & HTTPS ii

- Then, copy the certificate to the server and add it with:

```
remoteserver$ lxc config trust add client.crt
```

- For a production setup:
  - It's recommended that `core.trust_password` is unset after all clients have been added.
  - This prevents brute-force attacks trying to guess the password.
  - Furthermore, `core.https_address` should be set to the single address where the LXD server should be available (rather than the any address 0.0.0.0).
  - Firewall rules should be set to only allow access to the LXD port from authorized hosts/subnets.

Using Our Remotes

- To list remotes:

```
localclient$ lxc remote list
```

- The `lxc` command line tool can talk to multiple LXD servers and defaults to talking to the local one.
- Use all the same command as above but prefixing the container and images name with the remote host like:

```
localclient$ lxc exec host-a:first -- apt-get update
```

Making Images Public

- The LXD daemon works as an image server:
  - To make our LXD image available for our server users, we have to modify the public parameter, it's false by default:

```
localclient$ lxc image edit myimage
```

  - It will open image properties in system default text editor.
  - Replace false in the last line with true and save the file.
- You have just shared your LXD image!
- Now, other users can add our server as a public image server.

Using Public Remotes for Images

- We can use public remotes:

```
localclient$ lxc remote add remotename IP --public
```

- They can use following command to create containers from our image:

```
localclient$ lxc launch remotename:myimage
```

Outline

LXD

- Concept
- Storage
- Basics
- Network
- Mounts
- Images
- Remotes
- Migrations
- Profiles
- Command Summary

Migrations i

- We can create a snapshot, tar it, and import it on the other server.
- We also have commands to directly copy the snapshot:

```
$ lxc snapshot www-vm
$ lxc info www-vm
$ lxc copy www-vm/snap0 server2:www-vm --verbose
```

- Then, you can start the container in server2.
- Finally, we can also try a live migration, which moves the container with its current state (processes, open files etc.)

Migrations ii

- To enable live migrations we need to install a tool on both hosts called CRIU, which is available in Ubuntu via:

```
$ sudo apt install criu
```

- Then, launch your container with the following,

```
$ lxc launch ubuntu mycontainer
$ sleep 5s # let the container get to an interesting state
$ lxc move host1:mycontainer host2:mycontainer
```

- And if everything goes right you'll have migrated the container.
- Note. Live migration is still in experimental stages and may not work for all workloads.

Outline

LXD

- Concept
- Storage
- Basics
- Network
- Mounts
- Images
- Remotes
- Migrations
- Profiles
- Command Summary

Profiles

- Limits for container VMs are implemented with profiles.
- We can list profiles with:

```
$ lxc profile list
```

- When we start LXD we will have only one default profile, used by 0 containers.
- We can see the profiles used by our container:

```
$ lxc info mycontainer | grep Profiles
Profiles:default
```

- We can delete and rename profiles wiht:

```
$ lxc profile show default
$ lxc profile delete default
$ lxc profile rename default something
```

- But **the default profile cannot be deleted or renamed**.

Using Profiles i

- We can get and set profile values:

```
$ lxc profile get default limits.cpu
```

- The previous is empty in the default profile.
- Let's create a new profile:

```
$ lxc profile create custom
$ lxc profile list
$ lxc profile delete custom
$ lxc profile copy default custom #another way of creating a profile
$ lxc exec mycontainer bash
mycontainer$ nproc ; free -h
```

- We will see that the container has the same amount of CPUs and memory as the host.

Using Profiles ii

- Now, let's edit the profile:

```
$ lxc profile edit custom
...
config:
    limits.cpu: 1
```

- Another way of editing is to use set:

```
$ lxc profile set custom limits.cpu 2
```

- We can apply a profile to running container and changes are immediately applied:

```
$ lxc profile add mycontainer custom
$ lxc profile set custom limits.memory "2GB"
$ lxc profile remove mycontainer custom
$ lxc delete mycontainer -f
```

Using Profiles iii

- Start a container with profile:

```
$ lxc launch ubuntu:18.04 mycontainer --profile custom
```

- The last profile applied takes precedence.
- To allow nesting (another LXD or Docker inside LXD)

```
$ lxc profile set custom security.nesting true
```

- We can also pipe a profile:

```
$ cat myprofile.txt | lxc profile edit myprofile
```

- You can see the configuration properties available in: `https://lxd.readthedocs.io/en/stable-3.0/containers`

Outline

LXD

- Concept
- Storage
- Basics
- Network
- Mounts
- Images
- Remotes
- Migrations
- Profiles
- Command Summary

Basic LXD Commands i

Basics:

```
$ sudo snap install lxd
$ sudo usermod -a -G lxd username
$ lxd init
$ lxc launch ubuntu:18.04 first
$ lxc list
$ lxc exec first bash
$ lxc exec first -- apt-get update
$ lxc exec webserver -- sudo --login --user myuser ls -la
$ lxc file pull first/etc/hosts .
$ lxc file push hosts first/tmp/
$ lxc stop first
$ lxc info mycontainer --show-log
$ lxc delete first
```

Storage:

```
# zpool create mypool /dev/sda1 /dev/sda2
# zpool add mypool /dev/sdb1
# zpool status -v
# zpool list
# zfs create mypool/mydataset
# zfs list
```

```
$ lxc storage info default
$ lxc storage create pool1 zfs source=/dev/sdX
$ lxc storage create pool1 zfs source=my-tank
$ lxc storage create pool1 zfs source=my-tank/slice
```

```
$ lxc storage create pool1 zfs source=/dev/sdX zfs.pool_name=my-tank
```

Basic LXD Commands ii

Limits and profiles:

```
$ lxc profile list
$ lxc info mycontainer | grep Profiles
$ lxc profile show default
$ lxc profile delete default
$ lxc profile rename default something
$ lxc profile get default limits.cpu
$ lxc profile create custom
$ lxc profile copy default custom
$ lxc profile edit custom
$ lxc profile set custom limits.cpu 2
$ lxc profile add mycontainer custom
$ lxc profile apply mycontainer default,microk8s
$ lxc profile set custom limits.memory "2GB"
$ lxc profile remove mycontainer custom
$ lxc launch ubuntu:18.04 mycontainer --profile custom
$ lxc profile set custom security.nesting true
$ cat myprofile.txt | lxc profile edit myprofile
```

Images:

```
$ lxc image list
$ lxc image list ubuntu:
$ lxc image info ubuntu:b
$ lxc image edit <img>
$ lxc snapshot mycont mysnapshot
$ lxc list mycont
$ lxc publish mycont/myshapshot --alias myimage
$ lxc image export myimage
$ lxc image import HASH.gz --alias myimage
```

Migrations:

```
$ lxc snapshot www-vm
$ lxc info www-vm
$ lxc copy www-vm/snap0 server2:www-vm --verbose
$ sudo apt install criu
$ lxc launch ubuntu mycontainer
$ lxc move host1:mycontainer host2:mycontainer
```

Basic LXD Commands iii

Remotes:

```
$ lxc config show
localclient$ lxc config set core.trust_password "mysupersecret"
localclient$ lxc remote add remotename @IP --password=mysupersecret
remoteserver$ lxc config trust add client.crt
localclient$ lxc remote list
localclient$ lxc exec host-a:first -- apt-get update
localclient$ lxc remote add remotename IP --public
localclient$ lxc launch remotename:myimage
```

Network:

```
$ lxc config device add c1 myport80 proxy listen=tcp:0.0.0.0:80 connect=tcp:127.0.0.1:80
$ lxc config device add c1 myport443 proxy listen=tcp:0.0.0.0:443 connect=tcp:127.0.0.1:443
$ lxc config device show c1
$ lxc config device remove mycontainer myport80
```

Mount:

```
$ lxc config device add c1 sharedwww disk source=/wwwdata/ path=/var/www/html/
```