CANONICAL

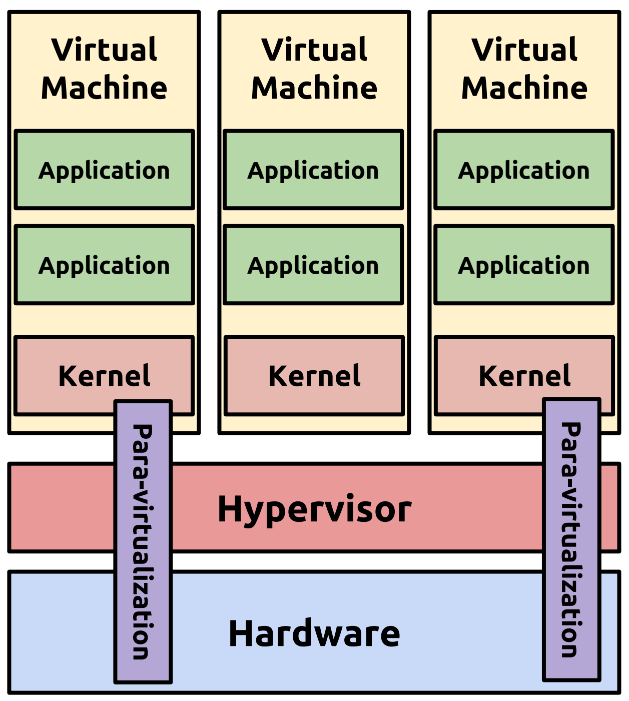
LXD
- next generation system container manager.

Paul Sim
Senior Technical Account Manager
paul.sim@canonical.com

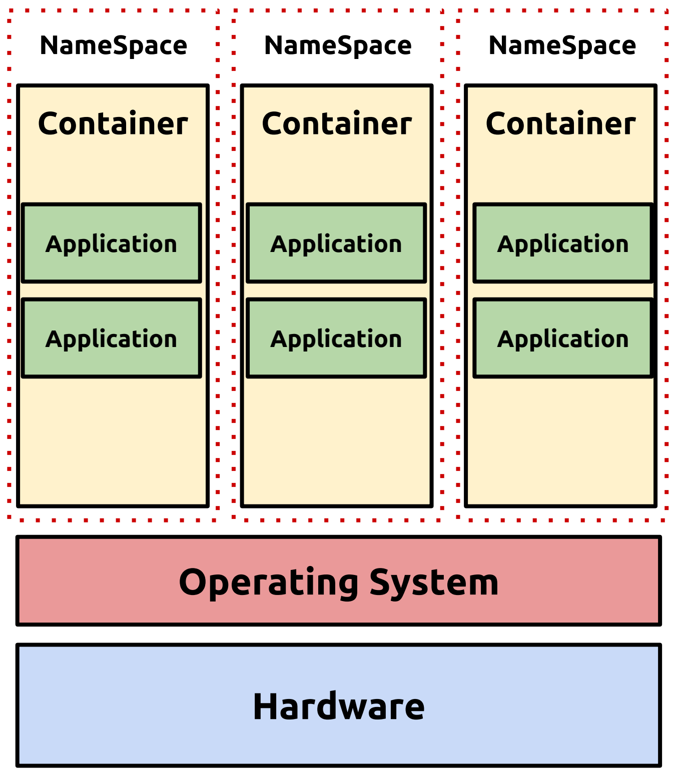
Agenda

- Virtualization
- LXD & LXC
- LXD vs Docker
- LXD basic
- LXD with MAAS/Juju

Virtualization



Type 1, Type 2



Lightweight virtualization

Namespace

Currently, Linux implements six different types of namespaces. The purpose of each namespace is to wrap a particular global system resource in an abstraction that makes it appear to the processes within the namespace that they have their own isolated instance of the global resource. **One of the overall goals of namespaces is to support the implementation of containers, a tool for lightweight virtualization (as well as other purposes) that provides a group of processes with the illusion that they are the only processes on the system.**

1. Mount namespaces (CLONE_NEWNS, Linux 2.4.19) isolate the set of filesystem mount points seen by a group of processes. Thus, processes in different mount namespaces can have different views of the filesystem hierarchy.
One use of mount namespaces is to create environments that are similar to chroot jails. However, by contrast with the use of the chroot() system call, mount namespaces are a more secure and flexible tool for this task. Other more sophisticated uses of mount namespaces are also possible. For example, separate mount namespaces can be set up in a master-slave relationship, so that the mount events are automatically propagated from one namespace to another; this allows, for example, an optical disk device that is mounted in one namespace to automatically appear in other namespaces.

Namespace

2. UTS namespaces (CLONE_NEWUTS, Linux 2.6.19) isolate two system identifiers—nodename and domainname—returned by the uname() system call; the names are set using the sethostname() and setdomainname() system calls. In the context of containers, the UTS namespaces feature allows each container to have its own hostname and NIS domain name.

3. IPC namespaces (CLONE_NEWIPC, Linux 2.6.19) isolate certain interprocess communication (IPC) resources, namely, System V IPC objects and (since Linux 2.6.30) POSIX message queues. The common characteristic of these IPC mechanisms is that IPC objects are identified by mechanisms other than filesystem pathnames. Each IPC namespace has its own set of System V IPC identifiers and its own POSIX message queue filesystem.

4. PID namespaces (CLONE_NEWPID, Linux 2.6.24) isolate the process ID number space. In other words, processes in different PID namespaces can have the same PID.

5. Network namespaces (CLONE_NEWNET, started in Linux 2.4.19 2.6.24 and largely completed by about Linux 2.6.29) provide isolation of the system resources associated with networking. Thus, each network namespace has its own network devices, IP addresses, IP routing tables, /proc/net directory, port numbers, and so on.

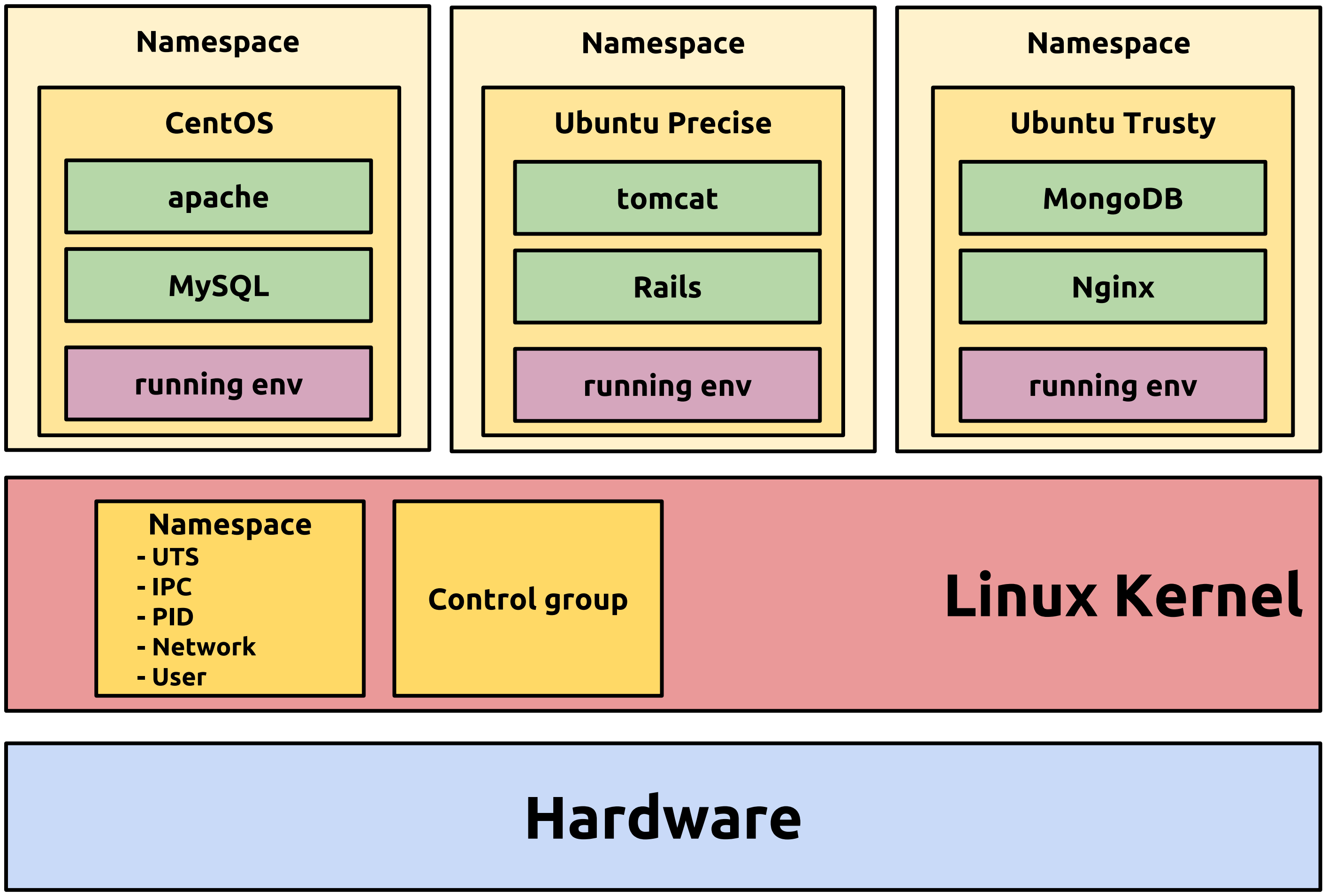
6. User namespaces (CLONE_NEWUSER, started in Linux 2.6.23 and completed in Linux 3.8) isolate the user and group ID number spaces. In other words, a process's user and group IDs can be different inside and outside a user namespace.

Namespace

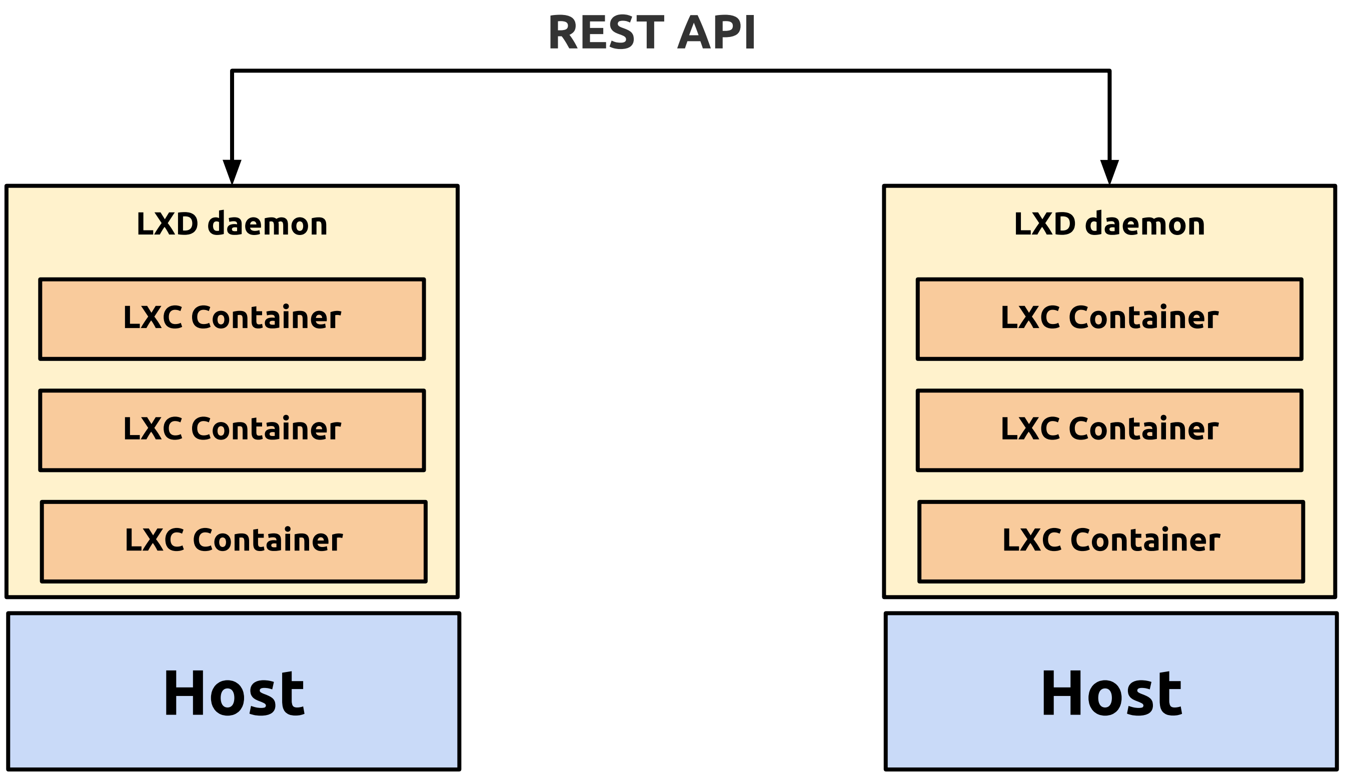
```
ubuntu@ubuntu:~# ps faux
...
root     16801  0.0  0.0  99536  5536 ?     Ss   07:48   0:00 [lxc monitor] /var/lib/lxd/containers c1
100000   16822  0.0  0.0  33524  4144 ?     Ss   07:48   0:00     \_ /sbin/init
100000   18358  0.0  0.0  19488   168 ?     S    07:48   0:00          \_ upstart-udev-bridge --daemon
100000   18364  0.0  0.0  49280  3184 ?     Ss   07:48   0:00          \_ /lib/systemd/systemd-udevd --daemon
...
ubuntu@ubuntu:~# sudo ls -l /proc/16801/ns
lrwxrwxrwx 1 root root 0 May 24 07:58 cgroup -> cgroup:[4026531835]
lrwxrwxrwx 1 root root 0 May 24 07:58 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 root root 0 May 24 07:58 mnt -> mnt:[4026531840]
lrwxrwxrwx 1 root root 0 May 24 07:58 net -> net:[4026531969]
lrwxrwxrwx 1 root root 0 May 24 07:58 pid -> pid:[4026531836]
lrwxrwxrwx 1 root root 0 May 24 07:58 user -> user:[4026531837]
lrwxrwxrwx 1 root root 0 May 24 07:58 uts -> uts:[4026531838]
ubuntu@ubuntu:~# sudo ls -l /proc/16822/ns
lrwxrwxrwx 1 100000 100000 0 May 24 07:51 cgroup -> cgroup:[4026532327]
lrwxrwxrwx 1 100000 100000 0 May 24 07:48 ipc -> ipc:[4026532269]
lrwxrwxrwx 1 100000 100000 0 May 24 07:48 mnt -> mnt:[4026532267]
lrwxrwxrwx 1 100000 100000 0 May 24 07:48 net -> net:[4026532271]
lrwxrwxrwx 1 100000 100000 0 May 24 07:48 pid -> pid:[4026532270]
lrwxrwxrwx 1 100000 100000 0 May 24 07:48 user -> user:[4026532266]
lrwxrwxrwx 1 100000 100000 0 May 24 07:48 uts -> uts:[4026532268]
ubuntu@ubuntu:~# sudo ls -l /proc/18364/ns
lrwxrwxrwx 1 100000 100000 0 May 24 07:58 cgroup -> cgroup:[4026532327]
lrwxrwxrwx 1 100000 100000 0 May 24 07:58 ipc -> ipc:[4026532269]
lrwxrwxrwx 1 100000 100000 0 May 24 07:58 mnt -> mnt:[4026532267]
lrwxrwxrwx 1 100000 100000 0 May 24 07:58 net -> net:[4026532271]
lrwxrwxrwx 1 100000 100000 0 May 24 07:58 pid -> pid:[4026532270]
lrwxrwxrwx 1 100000 100000 0 May 24 07:58 user -> user:[4026532266]
lrwxrwxrwx 1 100000 100000 0 May 24 07:58 uts -> uts:[4026532268]
```

CANONICAL

Linux Container

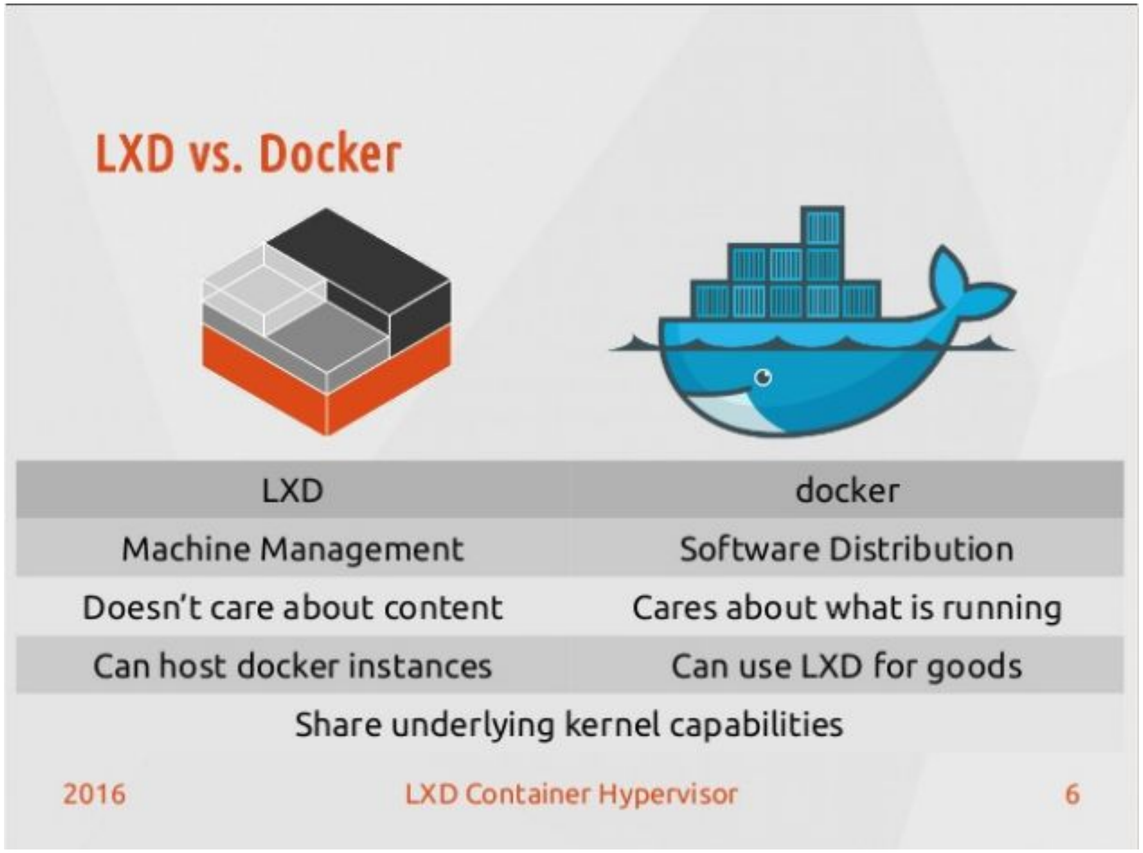


LXD & LXC



LXD isn't a rewrite of LXC, in fact it's building on top of LXC to provide a new, better user experience. Under the hood, LXD uses LXC through liblxc and its Go binding to create and manage the containers.
It's basically an alternative to LXC's tools and distribution template system with the added features that come from being controllable over the network.

LXD vs Docker



LXD vs. Docker

LXD	docker
Machine Management	Software Distribution
Doesn't care about content	Cares about what is running
Can host docker instances	Can use LXD for goods
Share underlying kernel capabilities	

2016 LXD Container Hypervisor 6

LXD : a machine container
Docker : an application container

https://www.slideshare.net/danialbehzadi/lxd-container-hypervisor-66045556

LXD Basic

- **Installation**
  - $sudo apt-get install lxd lxd-client
  - $sudo apt install zfsutils-linux   #optional
- **Initialization**
  - $sudo lxd init
- **Launch a container**
  - $sudo lxc launch ubuntu:14.04 c1
- **Manage a container**
  - $sudo lxc stop|start|restart|pause|delete <c_name> [--force|--stateful]

LXD Basic

- **Configurations**
  - $sudo lxc config edit|show <c_name> [--expanded]
  - Live config change
    - $sudo lxc config set|unset <c_name> <key> <value>
    - e.g)$sudo lxc config set c1 limits.cpu.allowance 10%
  - For profile
    - $sudo lxc profile edit|show <profile_name>
  - In default, lxc config doesn't show configurations inherited by profiles. Use --expanded option to see it.

    https://github.com/lxc/lxd/blob/master/doc/configuration.md
- **Image**
  - $sudo lxc image list
  - To see repositories
    - $sudo lxc remote list
  - To import/download an image into local
    - $sudo lxc image copy ubuntu:16.04 local:
    - $sudo lxc image copy images:debian/8/amd64 local:

LXD Basic

- **Creating a custom Image from a container**
  1) $sudo lxc stop c1
  2) $sudo lxc publish c1 --alias my-image
  3) $sudo lxc image list

- **Remote LXD**
  1) Allow remote connection
     - $sudo lxc config set core.https_address [::]:8443

  2) on working host

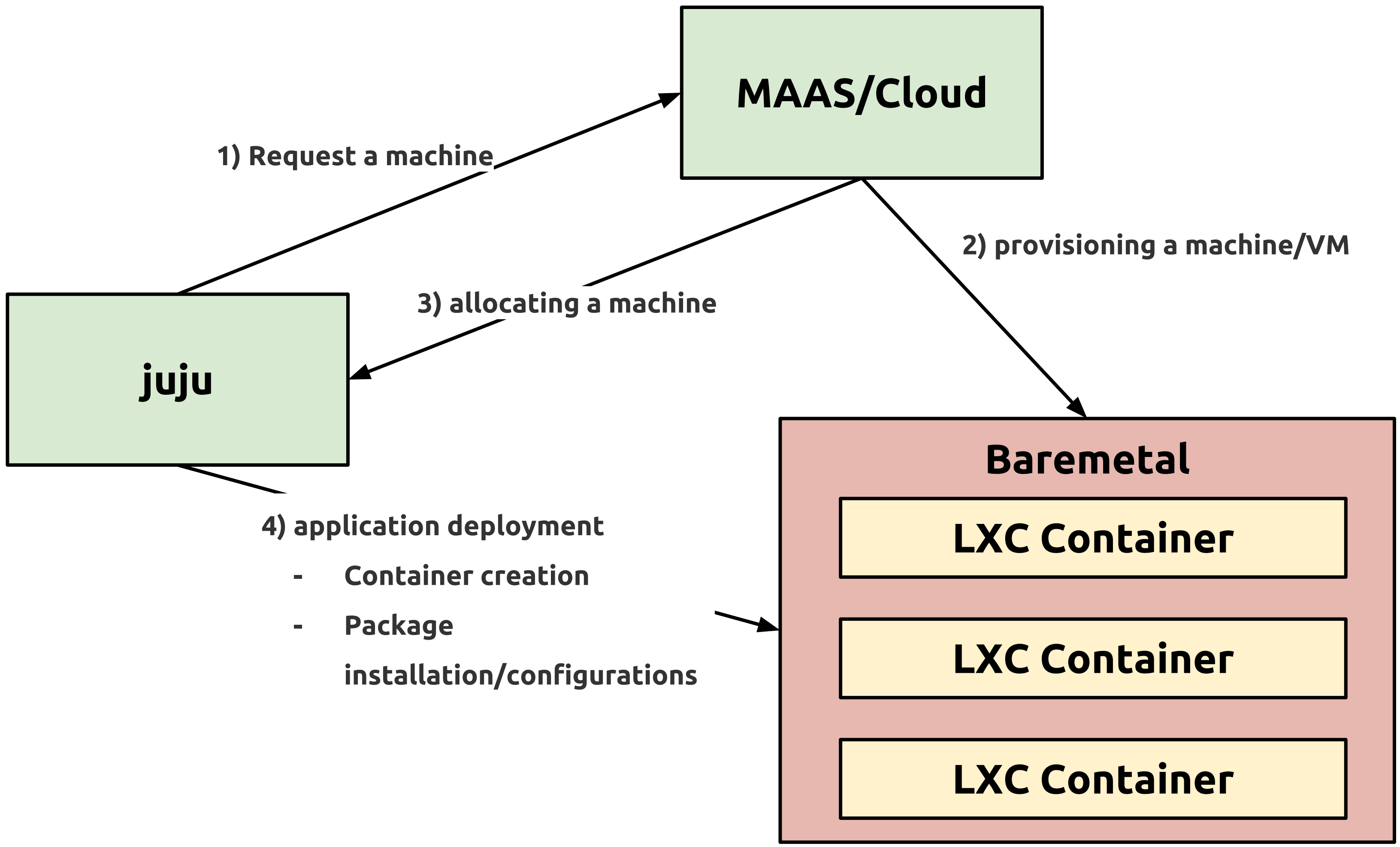
     - $sudo lxc remote add <remote_name> <IP>
     - e.g) $sudo lxc remote add ubuntu-6 172.30.1.119
  - Working with the remote LXD
     - $sudo lxc list ubuntu-6:
     - $sudo lxc start ubuntu-6:c1
  - Container migration
     - $sudo lxc stop ubuntu-6:c1
     - $sudo lxc move c1 ubuntu-6:c1

LXD with MAAS/juju

- **MAAS**
  - **M**etal **A**s **a** **S**ervice.
  - Baremetal provisioning tool
  - *https://maas.io/*

- **juju**
  - Application modelling tool.
  - Application deployment, configuration and scale
  - *https://jujucharms.com*

LXD with MAAS/juju


MAAS/Cloud
1) Request a machine
2) provisioning a machine/VM
3) allocating a machine
juju
4) application deployment
- Container creation
- Package installation/configurations
Baremetal
LXC Container
LXC Container
LXC Container