Using Containers for GPU Workloads

NVIDIA GPU Technology Conference
San José, California

Christian Brauner
Software Engineer, Canonical Ltd.
christian.brauner@ubuntu.com
@braun_eric
https://brauner.github.io

Serge Hallyn
Principal Engineer, Cisco
shallyn@cisco.com
@sehh
https://s3hh.wordpress.com
Who we are

LXC

- Venerable container manager
- Umbrella project
  - Make containers better
  - Contributions to
    - Kernel
    - Other projects (shadow, glibc, ...)
  - Create, foster new software projects
    - Lxcfs
    - Lxd
    - Pylxd
    - Cgmanager
    - Libresource
Containers: A userspace fiction

Early uses of ‘containers’ (before containers):

- Jails
- VPS
- Plan 9
- MLS (/tmp polyinstantiation)
- Checkpoint/restart
- Borg

Newer uses of containers features:

- NOVA network (openstack)
- Sandstorm (sandbox)
- Chrome (sandbox)
- FTP daemons
- Actual containers (lxc, lxd, docker, …)
Building blocks

- Namespaces
  - Mounts
  - PID
  - UTS
  - IPC
  - Cgroup
  - Network
  - User
  - (time, ima, LSM, ...)

- Capabilities bounding set
- LSM
- Cgroups
- Seccomp
- Devpts

Advantages:
- No emulated hardware
- No guest kernel
- Flexible sharing with host
- Easy introspection/debugging from host
LXC and LXD
LXC vs LXD

LXC

- No long-running daemon
- Completely unprivileged use
- Local use only
- (lxcpath, container)

LXD

- Privileged long-running daemon
- Image based
- Remote based
  - Macbook client:
    - `lxc remote add host2 host2.example.org`
    - `lxc launch ubuntu:xenial host2:i1`
    - `lxc exec host2:i1 touch /tag`
    - `lxc publish host2:i1 host3: --alias img2`
User namespace

Requirements:

● Uid separation (c1.1000 != c2.1000)
● Container root privileged over container
● Container root not privileged over host
● Able to nest

Details:

● Userids are *mapped*
● Capabilities targeted to user ns
● Namespaces, resources owned by a ns
● Hardware belongs to initial user ns
● Uid 1000 can always map uid 1000
● Root can *delegate* other uids to 1000
● (demonstrate)
Using Devices In Containers

- Very fast networking
  *Infiniband, SR-IOV*
- Interacting with devices
  *cell phones, scientific equipment*
- Dedicated block storage
  *physical disks or partitions*
- Computation
  *GPUs*
Using Devices In Containers

- Device access is handled by the host kernel
  The hardware doesn't need any special capabilities.
- Device nodes are identified and passed to the container
  The workload doesn't need to be container-aware.
- Devices can be shared very efficiently
  The same device can be passed to multiple containers, allowing for simultaneous access if the kernel driver supports this.
- Devices can be attached and detached on the fly
  They are just files or kernel constructs so can be moved around, added and removed as needed without requiring a reboot of the host or container.
Demo Time