



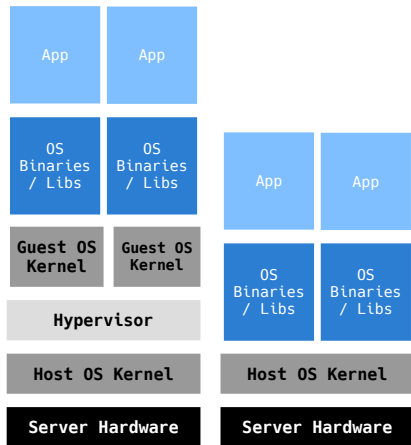
Containers: Introduction

Jakob Blomer

GDB
April 12th, 2017

Container pros and cons:

- 😊 Smaller virtualization overhead for system calls, I/O, memory translation
- 😊 Better at overcommitting with idle services
- 😊 Boots faster (with caveats)
- 😊 Orchestration tools available
- 😞 Weaker isolation
- 😞 No “privileged operations”, e.g. mount
- 😞 Linux only
- 😞 More moving parts



Linux containers are not a first class citizen

Virtualization of individual kernel resources

Useful utilities: `unshare`, `nsenter`, `/proc/PID/ns`, `/proc/PID/mountinfo`

pid : Virtual process identifiers:

```
sudo unshare -fork -pid /bin/bash
echo $$ → 1
```

user : Virtual uid/gid mappings. Enables fake root:

```
unshare -U -r /bin/bash
```

net : detach network adapters

mount : detach directory tree from parent process

mount points can be

private : complete isolation between process groups

shared : mounts are propagated upwards and downwards

slave : mounts are only propagated downwards

more : inter-process communication, host name, ...

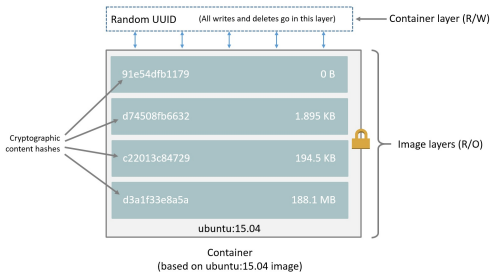
Powerful, but: complex to handle manually, hard to diagnose!

Hierarchical resource containers, confines applications

- Steered through the cgroups file system:

```
$ mkdir /sys/fs/cgroup/memory/small
$ echo $((1024*1024)) >
/sys/fs/cgroup/memory/small/memory.limit_in_bytes
$ echo $$ > /sys/fs/cgroup/memory/small/tasks
$ cat /sys/fs/cgroup/memory/small/tasks
13600
13658
$ firefox
Killed
```
- Higher level interfaces: cgconfig, cgcreate, cgdelete, ...
- Controllers for memory, cpu pinning, device access, freezing, dots

Useful in its own right, e.g. HTCondor, benchmarks

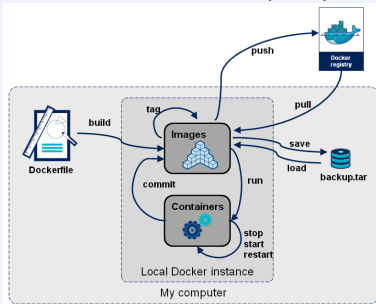


Source: Docker

- The “image” is usually a tarball with the root file system of the container
- Docker can assemble images from multiple tarballs in “layers”
- The layered approach requires a union file system to create a single root mount point
- Another option: bind mount of writable parts into a read-only root file system (/var, /tmp, /home, ...)

Container Engines

- Docker: most influential one, introduced the push-pull model for containers



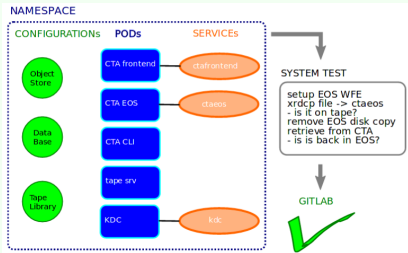
Source: <http://blog.octo.com/en/docker-registry-first-steps>

- Singularity: interesting new engine from the HPC world, very lightweight
- lxc, rkt, systemd-nspawn

Container Orchestration

- Mesos and DC/OS: two-level cluster scheduler, good for production services
- Kubernetes: container orchestration, good for running ensembles of containers
- Docker Swarm

Example: Test cluster with Kubernetes

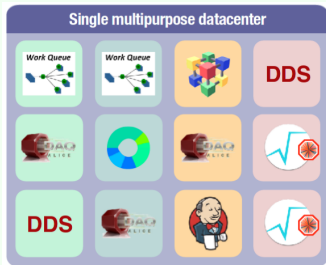


Source: Julien Leduc

Container Orchestration

- Mesos and DC/OS: two-level cluster scheduler, good for production services
- Kubernetes: container orchestration, good for running ensembles of containers
- Docker Swarm

Example: Production Services Cluster on Mesos



Source: Dario Berzano

Bind Mount

```
docker run -v /cvmfs:/cvmfs:shared ... or  
docker run -v /cvmfs/sft.cern.ch:/cvmfs/sft.cern.ch ...
```

- Cache shared by all containers on the same host

Docker Volume Driver

```
https://gitlab.cern.ch/cloud-infrastructure/docker-volume-cvmfs/  
docker run --volume-driver cvmfs -v  
cms.cern.ch:/cvmfs/cms.cern.ch ...
```

- Integrates with Kubernetes

From Inside Container

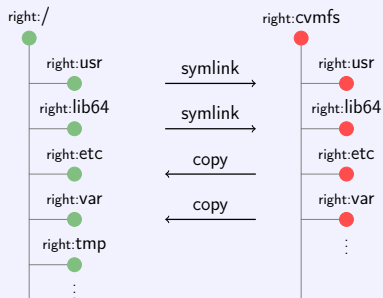
```
docker run --privileged ...
```

- Probably not very much used in practice

Options for CernVM-FS

- 1 Fuse, mapped from host
 - Shared cache
 - Requires privileges on the host
- 2 Using Parrot-Cvmfs
 - Pure user-space (ptrace)
 - Can impact performance and stability

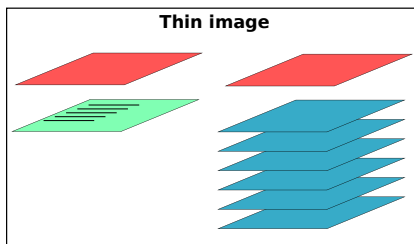
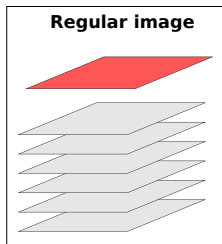
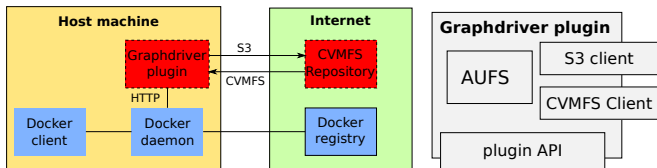
Root file system (/) layout



Limitations

Can be used to run tasks, does not allow derived containers

Work by N Hardi, expected H2/2017



■ read-write layer

■ thin image layer

■ local read-only layer

■ read-only layer on CVMFS

- Containers used for
 - Isolation: e.g. replacing glexec, resource containment
 - Virtual environments:
 - CentOS on Ubuntu, SL4 on CentOS7 (data preservation)
 - Unit of scheduling in distributed systems: Kubernetes, Mesos
- Docker/Singularity for isolation + CernVM-FS for image distribution:
 - Works out of the box with Singularity
 - Bind mounts and volume driver for experiment software in Docker
 - Full support for Docker's pull – commit – push lifecycle:
 - CernVM-FS graph driver (expected H2/17)
- There are certain dangers with containers
 - More moving parts (and moving targets) in your system
 - Containers foster an attitude of “capturing the mess”
 - Requires automation: containers need to be disposable items (e.g. no carriers for storage, databases)