

docker containers

-

first steps

Sébastien Binet

LAL/IN2P3

2014-10-14



???



Docker: what is it ?



Docker: what is it ?

- <http://www.docker.io/>
- an open source project to pack, ship and run any application as a lightweight container

High level description

- kind of like a **lightweight** VM
- runs in its own process space
- has its own network interface
- can run stuff as `root`

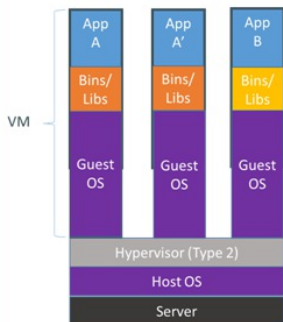
Low level description

- `chroot` on steroids
- container = isolated process(es)
- share kernel with host
- no device emulation

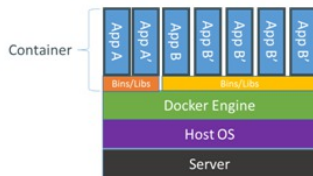
Docker: why ?

- same use cases than for VMs
- **speed**: boots in (milli)seconds
- **footprint**: 100-1000 containers on a single machine/laptop. small disk requirements

Containers vs. VMs



Containers are isolated, but share OS and, where appropriate, bins/libraries



Efficiency: *almost* no overhead

- processes are isolated but run straight on the host
- CPU performance = **native** performance
- memory performance = a few % shaved off for (optional) accounting
- network performance = small overhead

Efficiency: storage friendly

- unioning filesystems
- snapshotting filesystems
- copy-on-write

- provisioning takes a few milliseconds
- ... and a few kilobytes
- creating a new container/base-image takes a few seconds

Docker: how ?

- Linux Containers (LXC) (or via `libcontainer`)
- Control Groups and Namespaces
- AUFS (overlying filesystem), or BTRFS, or DeviceMapper
- Client-Server with an `http/REST` API

LXC

- Let's you run a Linux system within another Linux system
- A container is a group of processes on a Linux box, put together is an isolated environment
- From the inside, it looks like a VM. From the outside, it looks like normal processes
- *"chroot on steroids"*

Control Group & Namespaces

Linux kernel feature to limit, account and isolate resource usage:

- CPU
- Memory
- disk I/O

Docker: how ? (AUFS example)

AUFS

- File system that implements union mount

```
$ mount -t aufs -o br=/files1:/files2 none /files
```

- Supports copy-on-write (COW)

```
$ mount -t aufs -o br=/tmp=rw:/bin=ro none /files
```

Go

Docker can be seen as a Go binary scripting LXC and AUFS



docker



August-2014: release **1.2.0**

Requirements

- Linux Kernel 3.8 or above
- AUFS || BTRFS || DeviceMapper (to get CoW)
- 64-bit

More on: <http://docs.docker.io/en/latest/installation>

Hello World

- get a base container (ubuntu, centos, ...)

```
$ docker pull ubuntu
```

- list images already pulled in:

```
$ docker images
```

ubuntu	12.04	8dbd9e392a96	5 months ago	131.5 MB (virtual 131.5 MB)
ubuntu	latest	8dbd9e392a96	5 months ago	131.5 MB (virtual 131.5 MB)
ubuntu	precise	8dbd9e392a96	5 months ago	131.5 MB (virtual 131.5 MB)
ubuntu	12.10	b750fe79269d	6 months ago	24.65 kB (virtual 180.1 MB)
ubuntu	quantal	b750fe79269d	6 months ago	24.65 kB (virtual 180.1 MB)

- run an executable inside a container

```
$ docker run ubuntu:12.10 echo ``hello world``
```

Detached mode

- run a container in detached mode:

```
$ docker run -d ubuntu sh -c \  
  while true; do echo "hello"; sleep 1; done;
```

- get the container id:

```
$ docker ps
```

ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
78c88e279f26	ubuntu:12.04	/bin/sh -c while tru	14 seconds ago	Up 11 seconds	

- attach to the container

```
$ docker attach 78c88e279f26
```

- start/stop/restart a container

```
$ docker stop 78c88e279f26
```

Public registry

- pull an apache container from the registry:

```
$ docker search apache  
$ docker pull creack/apache2
```

- run the image and check the ports

```
$ docker run -d creack/apache2  
$ docker ps
```

ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
369602483ae9	creack/apache2:latest	/usr/sbin/apache2ctl	4 seconds ago	Up 1 seconds	49153->80, 49154->443

Also available from the browser:

<https://registry.hub.docker.com/>

- **run docker interactively:**

```
$ docker run -i -t ubuntu bash
root@bf72b1a06e6c:/# apt-get update
Reading package lists... Done
```

```
root@bf72b1a06e6c:/# apt-get install memcached
[...]
root@bf72b1a06e6c:/# exit
```

- **commit the resulting container**

```
$ docker commit `docker ps -q -l` binet/memcached
ab59e4b14266
```

- **run the image**

```
$ docker run -d -p 11211 -u daemon binet/memcached memcached
ab59e4b14266
```

```
## build gaudi
FROM hepsw/slc-base
MAINTAINER binet@cern.ch

ENV MYSITEROOT /opt/lhcb-sw
ENV CMTCONFIG x86_64-slc6-gcc48-opt

RUN mkdir -p $MYSITEROOT

## install some system dependencies
RUN yum install -y bzip2 freetype tar which

## retrieve install
RUN curl -O -L http://cern.ch/binet/sw/dist/lbpkr && \
    chmod +x ./lbpkr

## install (source+binaries)
RUN ./lbpkr install GAUDI_v25r1_x86_64_slc6_gcc48_opt
```

- build the container

```
$ docker build --tag=hepsw/gaudi:v25r1 .  
$ docker tag hepsw/gaudi:v25r1 hepsw/gaudi:latest
```

- run the container (and test the build)

```
$ docker run -i -t hepsw/gaudi /bin/bash  
[hepsw/gaudi] $ cd /scratch  
[hepsw/gaudi] $ gaudirun.py \  
    $GAUDIEXAMPLESROOT/options/TupleEx.py
```

- bind mounts

```
$ docker run -i -t hepsw/gaudi \  
    -v /host/build/results:/scratch  
    /bin/bash
```

- copy files from container to host

```
$ docker cp hepsw/gaudi:/scratch /host/build/results
```


Docker: when ? where ?

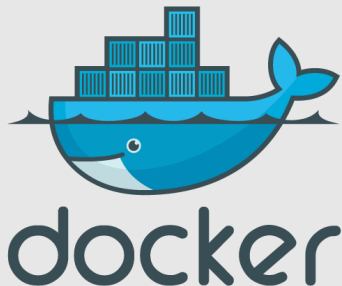
- OpenStack-Havana has support for running `docker` containers
 - so since beginning 2014: technically possible to run those on `openstack.cern.ch` (some configuration required though [[RQF0335022](#)])
 - support for containers on Google Compute Engine
-
- since version `docker-0.7` no AUFS hard-requirement. multiple backends:
 - ▶ BTRFS
 - ▶ DeviceMapper
 - packaged for Fedora-20
 - available on RHEL/SLC-6.5 (`apt-get install docker-io`)



- [kvm-and-docker-lxc-benchmarking](#)
 - ▶ **executive summary:** `docker` delivers very close to the bare-metal performances (consistently better than `KVM` save for some `mysql` tests)
- **tests** `docker` containers creation, guest CPU/Mem/IO performances, ...

- no container backend for MacOSX (yet?)
 - it is foreseen that at some point a `jail`-based backend will appear
 - ▶ support for other BSDs too
 - in the meantime: `boot2docker`
 - ▶ launches a very thin Linux-VM where the `docker` daemon is installed
 - ▶ installs the `docker` client on the host
 - ▶ talks via HTTP/REST to the daemon
-
- `boot2docker` works also for Windows (TM)

- easily distribute dev-environments
- easily provision build and dev-environments
- easily relocate binaries (remember: `chroot` on steroids!)
- provision efficient performance-wise environments (production)
- easily build and deploy appliances (multi-containers micro-services apps)
- easily manage clusters of containers (mesos/kubernetes+docker)



```
hepsw/docks (github)  
hepsw/containers (docker public registry)
```