What place for the containers in the HPC world ?

07/05/17

Rémy Dernat – CNRS - ISE-M (CNRS/UM/IRD) / MBB platform LabEx CeMEB

JDEV2017

Containers

- Use some “isolate” Linux capabilities:
  - chroot,
  - **cgroups** *(kernel > 2.6.24, 2008)*,
  - **Namespaces** (see bellow),
  - overlayFS *(kernel 3.18, Dec. 2014)*.

- Possible **Namespaces**:
  - mnt (mount points, filesystems) *(kernel >= 2.4.19, 2002)*
  - pid (processes) *(Kernel >= 2.6.24, Jan. 2008)*
  - net (network stack) *(Kernel >= 2.6.24, Jan. 2008)*
  - ipc (InterProcess Communication) *(Kernel >= 2.6.19, Oct. 2006)*
  - uts (hostname / domainname) *(Kernel >= 2.6.19, Oct. 2006)*
  - user (UIDs/GIDs) *(complete in kernel >= 3.8, Feb. 2013)*
  - Cgroup Namespace (*Kernel >= 4.6 May 2016*)

Docker vs the rest of the world

- Some containers techs:
  - BSD Jails
  - Solaris Zones
  - OpenVZ
  - LXC/LXD
  - Docker
  - rkt
  - systemd-nspawn
- In the HPC world, there are some initiatives but there are still rare.

Containers in production environments

- 1. Mainly used in **cloud environments** (Amazon, Google cloud...)
- 2. Moderate/weakly used in **standards datacenters**:
  - Proxmox and OpenVZ / LXC containers
  - Continuous development/integration/delivery/testing/deployment
  - Except for some modern datacenters, but generally, these people are also using the cloud (point 1).
- 3. Poor usage in the **HPC** world (until… And except for... - *will see it later*...)

Why using containers in the HPC world ?

- **Mobility**: bring your working environment to the unfriendly HPC environment to benefit its hardware performances.
- Develop locally, build container, deploy at scale.

  Continuous Integration/Delivery/Deployment.

  DevOps aim: conciliates Ops and Devs.
- Less headaches for the HPC sysadmin (It is easy !), except for some containers techs/users that need additional security controls.

  → transfer partially the software stack responsibility to the user.

  It leverages the user independence and its linux skills. To achieve this, the administrator should only give some good practices and recipes to help the user.

Why using containers in the HPC world ?

- **Reproducible Research ?**

  Recipes + Versioning + packaging + publishing/sharing

> ***Reproducibility*** *is the ability of an entire analysis of an experiment or study to be* ***duplicated****, either by the same researcher or by someone else working independently, whereas* ***reproducing*** *an experiment is called* ***replicating*** *it. Reproducibility and replicability together are among the main principles of the scientific method. (source: Wikipedia)*

Docker and its environment vs HPC environment

- 2 possible approaches :
  - Containers **orchestration with a job scheduler** inside each container (runner or master).
  - Classical **Job scheduler which distribute containers**.
- Docker orchestrators: Swarm, YARN, Mesos DC/OS, Kubernetes, etc...

  vs HPC Job Scheduler (SGE, Torque, SLURM, etc…)

  Oriented Docker *o*rchestrators hardly integrate within existing HPC environments with another job scheduler.

Docker and its environment vs HPC environment

- Not HPC oriented. Created for the clouds:
  - **Micro-services, not “jobs”**...
  - The orchestrators distribute widely the containers without a true inquiry about the resources or scheduling and fair sharing.
  - Docker has **no HPC specific features**. It does not benefit the best of hardware/system performances (filesystem, network, specific accelerators (GPU, Xeon Phi...)).

Docker and its environment vs HPC environment - Security

- Not intended for multi-user use on the host(s)
  - Obvious security issues; the **docker daemon is running under *root***.

    Example :

```
$ docker run -ti -v /:/tmp/_root_host test_remy bash
# rm -rf /tmp/_root_host/
```

Docker and its environment vs HPC environment - Security

- Almost no certified images and secure images.
  - *“**30%** of the images on the Docker Registry contain vulnerabilities”(1)*

1.BanyanBlog – 2015
https://banyanops.com/blog/analyzing-docker-hub/

Securing containers

- *How to **secure** it ?*
  - Upgrade !
  - *Limit resources: cgroups + limits.conf/sysctl.conf on the host.*
  - *Docker:* capabilities, seccomp, user namespace, Docker Bench*(1)*, Docker-ee*(2)*, udocker*(3)*, Moby project*(4)* + LinuxKit*(5)*,...
  - *Image Analysis*: Clair*(6)*, quay.io security analyses, Docker Security Scanning*(7)*, ... ?
  - ***HPC focus***: Shifter*(8)*, Singularity*(9)*, CharlieCloud*(10)*…

1.https://dockerbench.com , 2.https://www.docker.com/enterprise-edition ,
3.https://github.com/indigo-dc/udocker , 4.https://mobyproject.org/ ,
5.https://github.com/linuxkit/linuxkit , 6.https://coreos.com/clair ,
7.https://docs.docker.com/docker-cloud/builds/image-scan/
8.https://www.nersc.gov/research-and-development/user-defined-images/
9.http://singularity.lbl.gov/ , 10.https://github.com/hpc/charliecloud

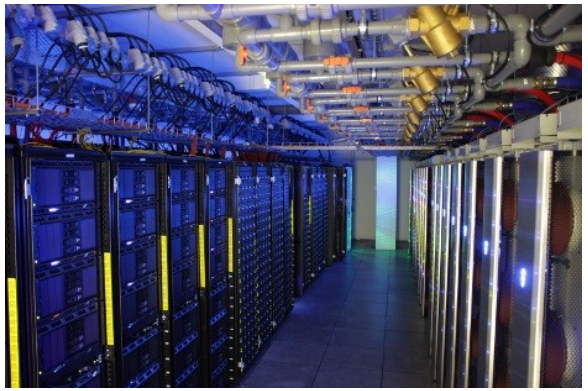
Comparison features table of Docker-like secure techs

Features	Docker-ee ($)	udocker	Shifter	Singularity	Charliecloud
Need a daemon	Yes	No	No	No	No
Permissions management	Yes	Not needed	Not needed	Not needed	Not needed
cgroups	Yes	No	Yes	No	No
Analyze images content	Yes (advanced edition ($$))	No	Yes / Partial	No	No
Access to the host devices	Yes (*--device* option)	No	No	Yes	Yes
True mapping of UIDs	No	Not for root	Not for root	Yes	No
All can be done from user	No (The admin needs to set permissions)	Yes	Yes (but it needs a gateway)	Yes (except bootstrap which requires root rights (*))	No
HPC ready	No	Yes (with some limitations)	Yes (and only for it (**))	Yes	Yes

(*) Can be done on a local machine and then transferred to the executing machine.
(**) it needs the corresponding infrastructure.

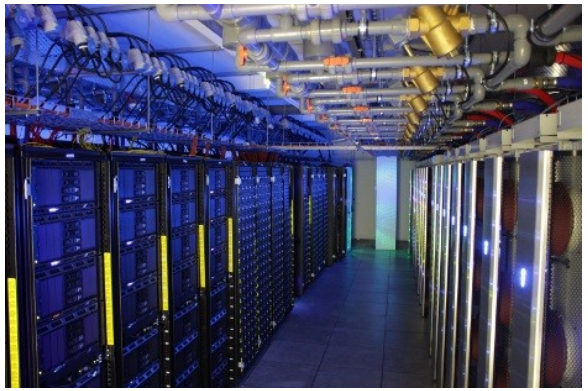
The containers in the HPC landscape

- Custom scripts
- Some initiatives :
  - Tight Integration with docker in HTCondor (*Docker Universe Applications*), Slurm cgroups, UGE container ed.,
  - The Dockstore,
  - cHPC,
  - Shifter,
  - Charliecloud,
  - Singularity



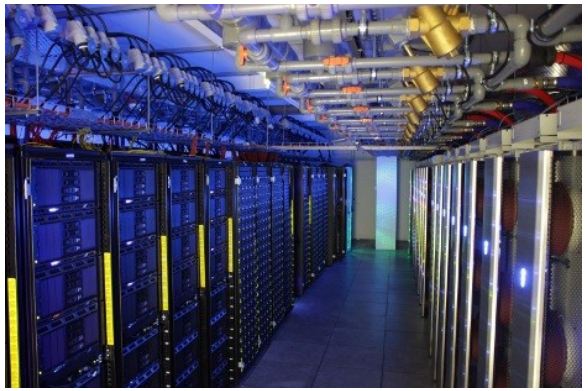
The containers in the HPC landscape

- Custom scripts
- Some initiatives :
  - Tight Integration with docker in HTCondor (*Docker Universe Applications*), Slurm cgroups, UGE container ed.,
  - The Dockstore,
  - cHPC,
  - Shifter,
  - Charliecloud,
  - Singularity



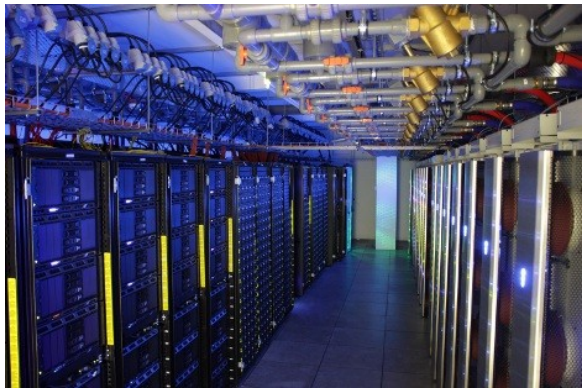
The containers in the HPC landscape

- Custom scripts
- Some initiatives :
  - Tight Integration with docker in HTCondor (*Docker Universe Applications*), Slurm cgroups, UGE container ed.,
  - The Dockstore,
  - cHPC,
  - Shifter,
  - Charliecloud,
  - Singularity



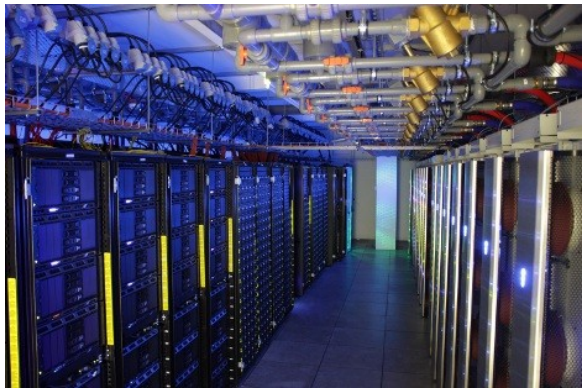
The containers in the HPC landscape

- Custom scripts
- Some initiatives :
  - Tight Integration with docker in HTCondor (*Docker Universe Applications*), Slurm cgroups, UGE container ed.,
  - The Dockstore,
  - cHPC,
  - Shifter,
  - Charliecloud,
  - Singularity



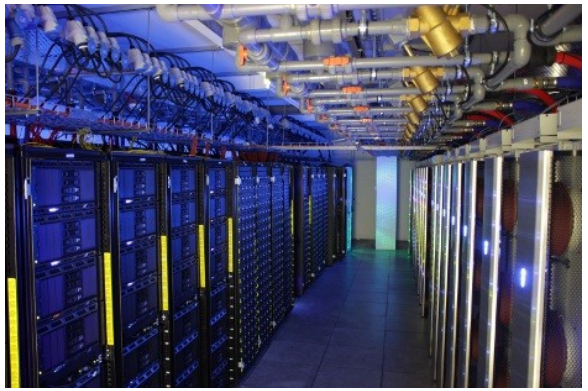
The containers in the HPC landscape

- Custom scripts
- Some initiatives :
  - Tight Integration with docker in HTCondor (*Docker Universe Applications*), Slurm cgroups, UGE container ed.,
  - The Dockstore,
  - cHPC,
  - Shifter,
  - Charliecloud,
  - Singularity



The containers in the HPC landscape

- Custom scripts
- Some initiatives :
  - Tight Integration with docker in HTCondor (*Docker Universe Applications*), Slurm cgroups, UGE container ed.,
  - The Dockstore,
  - cHPC,
  - Shifter,
  - Charliecloud,
  - Singularity



Singularity

- http://singularity.lbl.gov/
- Mostly developed par Gregory M. Kurtzer (CentOS, Warewulf, …) at Lawrence Berkeley National Laboratory,
- Motto BYOE (Bring You Own Environment),
- The container is **only one file**,
- Developed with HPC in mind: MPI, CUDA (…) !
- Pulling from docker to build a singularity container is possible ! As well as Dockerfile conversion to a Singularity Spec file.
- Possibility to push the image in a singularity hub or even in a docker registry (docker hub, local…).
- Integrates smoothly with any Job Scheduler, like any application.

Singularity

- True mapping of UID and GID inside/outside the container,
- User escalation is not possible,
- Bind writable directories /tmp, /var/tmp and $HOME to the container (by default).
- No (root) daemon,
- Possibility to include unit test.
- Singularity workflow:
  - *Dockerfile* → *the user*,
  - Creating/Building the image → root or user *(*)*,
  - Execution → the user,

  (*) Bootstrapping is available for root only, but you can build it on your local machine and then transfer it to a cluster as it is only one file !
- Possibility to use Pipe:

```
cat /path/to/python/script.py | singularity exec \
/tmp/Demo.img python
```

Singularity Spec File (.def)

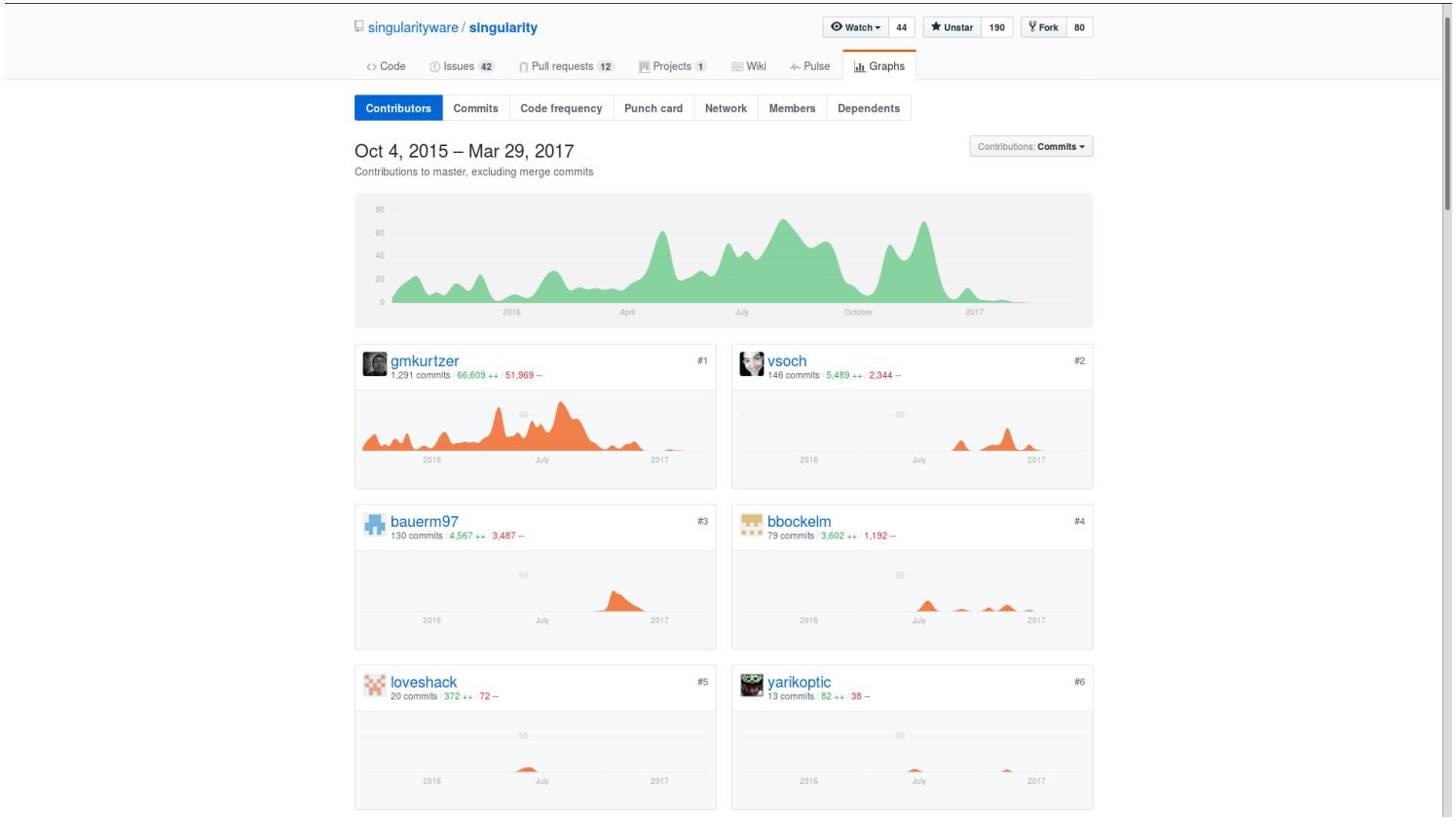
```
sudo singularity create –size 2048 file.img
sudo singularity bootstrap file.img file.def
```

Header, %setup, %post, %test, %files, and %runscript sections

```
Bootstrap: docker
From: ubuntu:latest
IncludeCmd: yes

%runscript
    echo "I can put here whatever I want to happen when the user runs my container!"
    exec echo "Hello " "$@"
%post
   echo "Here we are installing software and other dependencies for the container!"
   apt-get update
   apt-get install -y git vim
```

Singularity - activity



Singularity : numbers & use cases

- Singularity big users:
  - TACC 400k cores
  - Oak Ridge Titan 300k cores
  - GreenCube GSI Helmotz Center Heavy Ion 300k cores
  - Open Science Grid + Cern =~ 500k singularity instances/day
- Use cases:
  - Nextflow
    - Center for Genomic Regulation (CRG), Pasteur Institute, Sanger Institute (UK), SciLifeLab (Sweden)
  - Dolmades (wine apps.)
  - UT Southwestern BioHPC (biocontainers)
    - Blender + GLIBC2.19

Reproducibility

- **Reproducibility**
  - What we want in Science (reproducible research),
  - Undeniable advantages for the containers,
  - But… *Will see it later*
- **Versioning** and **sharing** simple files to be able to rebuild the same image:
  - *Dockerfile* for Docker
  - *Def/Specification file* for Singularity
  - Registries to push and share images.

Reproducibility

- What could **change** results between the host and the container (on the same host) ?
  - **Libraries** (used, versions…),
  - Compilation if needed (gcc ?, options, versions…),
  - **Environment variables**,
  - **External resources**:
    - Programs called (present or not, version...),
    - Downloads (the image, codes and packages…),
    - (non) Mounted filesystems (local bind, nfs…),
    - Random numbers,
    - Accesses and permissions with other programs and on resources :
      - Competitions, security (apparmor/selinux/seccomp profiles), …
- How to **solve** these problems (except for random numbers) ?
  - Using a .Spec file/Dockerfile, **running tests** (checksums, ldd (libraries), test section, scripts), **local resources**, and removing/controlling restrictions/checking accesses….

Reproducibility

- What could **change** results between the host and the container (on a different host) ?
  - *Each point from previous slide* +,
  - The **kernel version** and **loaded drivers** (presents ?, versions...),
  - The **processor** (brand (Intel/AMD), type/architecture (amd64 ? arm ? floating point precision...), available/activated features (AVX, SSE…)...), endianness ?,
  - Specific **local configurations**:
    - Local time, locales, encoding,
    - Network configuration (network stack, DNS…),
    - Sourced files
    - …
- How to **solve** all these problems ?
  - See previous slide → **tests**...

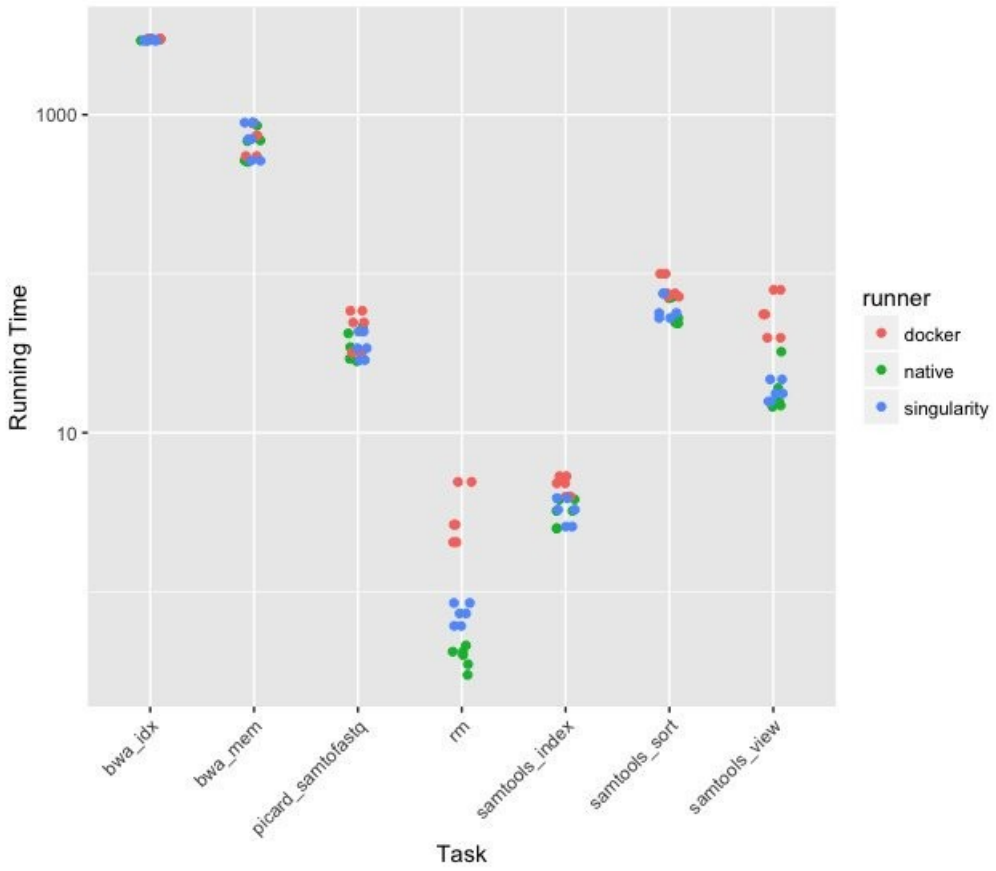
Performances and reproducibility

- Comparing reproducibility tests between **bare-metal** vs **singularity** vs **docker** vs **lxc**:
  - **Building a binary** and checking it with sha1sum and libraries with ldd:
    - Gives the **same results on a similar host** (sha1sum and libraries if it has been built with the same distro, the same version of gcc and in the same way).
    - A binary compiled on the host and then transferred on the container gives the same results (HPL linpack benchmarks) as if it was built on the container.

Performances and reproducibility

- Comparing some benchmarks between **bare-metal** vs **singularity** vs **docker** vs **lxc**:
  - Checking **startup** benchmarks,
  - Checking **network** benchmarks,
  - Checking **CPU** benchmarks,
  - Checking **HDD** IOs benchmarks,
  - Checking **Memory** benchmarks,
  - Checking **GPU** benchmarks,

Jonathan Dursi – bioinfo tools benchmarks Singularity vs Docker and Native



(c) Jonathan Dursi. Y Axis in seconds

Benchmarks ref.

Brendan Gregg:

http://brendangregg.com/blog/2017-05-15/container-performance-analysis-dockercon-2017.html

Paulo Di Tommaso:

https://www.nextflow.io/blog/2016/more-fun-containers-hpc.html

Jonathan Dursi:

https://github.com/CanDIG/images_bakeoff

Vanessa Soch:

https://github.com/vsoch/singularity-scientific-example/blob/master/scripts/10.generate_graphs.ipynb
http://www.vanessasaur.us/singularity-scientific-example/results/

Eduardo Arango / Rémy Dernat:

https://github.com/remyd1/containers-benchs/

Merci