Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules

Singularity overview

Vanessa HAMAR

Disclaimer

- The information in this presentation was compiled from different Web sites, mainly Docker, Singularity, OSG and WLCG’s sites.

Overview

- Why containers?
- Containers comparison
- Singularity
  - Definition
  - Roles
- Versions
- Use cases
- Image distribution
- Security
- Singularity at CC
- Conclusions
- Some links

Why containers?

- “Containers are a solution to the problem of how to get software to run reliably when moved from one computing environment to another. This could be from a developer's laptop to a test environment, from a staging environment into production and perhaps from a physical machine in a data center to a virtual machine in a private or public cloud.”*

*cio.com

Containers – Brief comparison

- Docker:
  - DevOps, microservices.
  - Enterprise applications.
  - Developers/DevOps

- Shifter:
  - Use the large number of docker apps.
  - Provides a way to run them in HPC after a conversion process.
  - It also strip out all the requirements of root so that they are runnable as user process.
  - Scientific Application Users

- Singularity:
  - Application portability (single image file, contain all dependencies)
  - Reproducibility, run cross platform, provide support for legacy OS and apps.
  - Scientific Application Users

http://geekyap.blogspot.fr/2016/11/docker-vs-singularity-vs-shifter-in-hpc.html

Singularity

Singularity is a container solution created by necessity for scientific and application driven workloads.

Containers for Science

Singularity

- Singularity enables users:
  - To have full control of their environment
  - To package:
    - entire scientific workflows
    - software and libraries
    - and even data.
  - To use the resources of the host were they are running the container:
    - HPC interconnects
    - Resource managers
    - File systems
    - GPUs and/or accelerators, etc.

This means that:

Users do not have to ask to cluster administrators to install anything for them – they can put it in a Singularity container and run.

No scheduler changes necessary

Singularity

- Singularity does this by enabling:
  - Encapsulation of the environment
  - Containers are image based
  - No user contextual changes or root escalation allowed
  - No root owned daemon processes

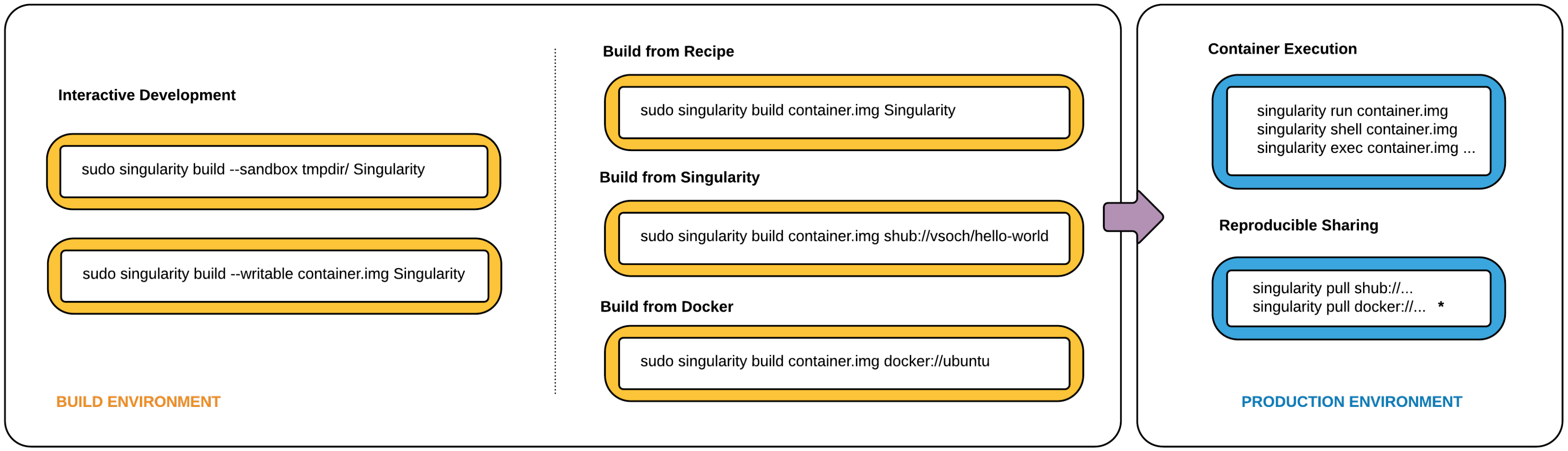
A user inside a Singularity container is the same user as outside the container

Singularity

- Network is transparent to the process.
- cgroups are not touched by Singularity, all is managed by the batch system.
- Support for MPI (OpenMPI, MPICH, IntelMPI)

- Singularity has two primary roles:

Container Image Generator **Container Runtime**



```
sudo singularity build --sandbox tmpdir/ Singularity
sudo singularity build --writable container.img Singularity
sudo singularity build container.img Singularity
sudo singularity build container.img shub://vsoch/hello-world
sudo singularity build container.img docker://ubuntu
```

```
singularity run container.img
singularity shell container.img
singularity exec container.img ...
```

```
singularity pull shub://...
singularity pull docker://... *
```

* Docker construction from layers not guaranteed to replicate between pulls

As root

As user

Users can make and customize containers locally, and then run them on shared resource

Singularity

- Image formats:
  - **squashfs**: is a compressed read-only file system that is widely used for things like live CDs/USBs and cell phone OS's
  - **ext3**: (also called writable) a writable image file containing an ext3 file system
  - **directory**: (also called sandbox) standard Unix directory containing a root container image
  - **tar.gz**: zlib compressed tar archive
  - **tar.bz2**: bzip2 compressed tar archive
  - **tar**: uncompressed tar archive

Default container format:
Squashfs >= version 2.4
Ext3 < 2.4

Singularity – Bind directories

- Singularity allows you to map directories on your host system to directories inside your container using bind mounts.

  - **System defined bind points:** The system administrator has the ability to define what bind points

  - **User defined bind points:** If the system administrator has enabled user control of binds, users will be able to request their own bind points within your container.

- Option enable overlay = yes directories can be created inside the container to be bind.

Singularity – Package version

Repository	Version
Epel	2.2.1
WLCG	2.4.2
OSG	2.4.2
Singularity	2.4.2

2.4.2 Standard Container Integration Format

Use cases

LSST - Deep Learning Challenges at CC

Context

LSST (Large Survey Synoptic Telescope) project members will have to deal with tons of data and want to explore deep learning solutions to automatically process images for what they call deblending (i.e separate astronomical objects from the sky background).

Deep learning challenges such as Kaggle (www.kaggle.com) are now popular on the internet.

A Challenge is made of a given problem to resolve, for example, « I want to know if this is a cat or a dog on this image » and a dataset to help the machine to learn. Challengers have to analyze the problem and propose a model to train.

A CNRS team developed RAMP, a deep learning challenge platform. Challengers submit their codes that are processed on RAMP backend. There are two phases of submission, and between them, each team can view the other teams code and adapt is own code with others ideas. This synergy gives better results. At the end of the process the score is computed. The team with the highest score wins.

Project from CC side (cf. workflow schema)

We have to connect to the RAMP platform and grab new submissions, then send submission code to the gpu batch farm. After batch is complete, results must be parsed and inserted to the RAMP platform database.

Why Singularity here ?

Gpus are Nvidia Tesla K80.

Nvidia provide updates for GPU libraries (Cuda, CuDNN)

Machine learning frameworks versions need to match gpu libraries versions

Example : Tensorflow 1.5, a famous Deep Learning framework requires CUDA Toolkit 8.0 and cuDNN v6.0 for gpu support
Moreover, each RAMP challenge requires a particular python development environment (different modules or versions of modules)

The advantage of using Singularity is obvious !

Use cases

- A local user needs to execute OpenMP code in Debian and our WNs are SL6 or CentOS7.

- ATLAS and CMS are also testing singularity in order to run SL6 containers in CentOS7.

Singularity – Image distribution

- From Brian Bockelman, pre-GDB, July 2017 presentation:
  - Given our heavy investment in CVMFS, it seems very natural to leverage it for image distribution.
    - Given CVMFS implementation details, images should be distributed as flat directories - Cache will work at the individual file level.
    - Flat directories work with non-setuid mode; single-image formats don’t.

- The OSG has now also installed unprivileged singularity 2.3.1 in cvmfs:

/cvmfs/oasis.opensciencegrid.org/mis/singularity/el[67]-x86_64/bin/singularity

Singularity - Security

- **User Namespace:** Singularity supports the user namespace natively and can run completely unprivileged (“rootless”) since version 2.2 (October 2016) but features are severely limited.

- **SetUID:** This is the default usage model for Singularity because it gives the most flexibility in terms of supported features and legacy compliance. It is also the most risky from a security perspective.

    - For that reason, Singularity has been developed with transparency in mind.

    - OSG-SEC-2017-10-27 Linux kernel vulnerability affecting Singularity. This vulnerability, can be exploited to cause a denial-of-service (DoS) condition or to execute arbitrary code

Commands

- Invoking a container

Executing a command:

```
singularity exec /cvmfs/singularity.in2p3.fr/images/cc/official/centos/x86_64/7/7.4/ ls
```

Interactive container:

```
singularity shell /cvmfs/singularity.in2p3.fr/images/cc/official/centos/x86_64/7/7.4/

singularity shell --bind /sps/hep/phenix:/srv /cvmfs/singularity.in2p3.fr/images/cc/official/sl/x86_64/6/6.9/
```

- Submitting a job in the computing cluster

A simple script:

mon_job_singularity.sh

```
#!/bin/bash
singularity exec /cvmfs/singularity.in2p3.fr/images/cc/official/sl/x86_64/6/6.9/ $HOME/my_script.sh
```

Normal submission:

```
qsub -q long -l os=cl7 mon_job_singularity.sh
```

Singularity at CC

- Version installed in user interfaces and worker nodes:
  - 2.4.2 from WLCG repository

- Image distribution
  - /cvmfs/singularity.in2p3.fr
    - Images as directories and bootstrap files for
      - Debian
      - SL6
      - CentOS7
      - Ubuntu

- Configuration options:
  - SetUID
  - Overlay

Conclusions

- A very nice idea ☺
- A new project - evolving fast
  - Some backward compatibility problems
  - Some bugs can be found
- Waiting information about version and configuration parameters from experiments.
- We need to do more tests.

Thanks to:

- Benoit Delaunay
- Bertrand Rigaud
- Emmanouil Vamvakopoulos
- Pascal Calvat
- Rachid Lemrani
- Sebastien Gadrat
- Yvan Calas

Questions

Where to find information

- Links:
  - Singularity - User guide
  - HPC Containers singularity
  - Docker vs Singularity vs Shifter

- Mailing lists:
  - “wlcg-containers (WLCG container working group)” wlcg-containers@cern.ch
  - “Singularity” singularity@lbl.gov

- Groups
  - Conteneurs IN2P3