Uni.lu HPC School 2018

PS14: HPC Containers: Singularity

Uni.lu High Performance Computing (HPC) Team

V. Plugaru

University of Luxembourg (UL), Luxembourg

http://hpc.uni.lu

Latest versions available on Github:

UL HPC tutorials: https://github.com/ULHPC/tutorials

UL HPC School: http://hpc.uni.lu/hpc-school/

PS14 tutorial sources: ulhpc-tutorials.rtfd.io/en/latest/virtualization/singularity

Summary

Main Objectives of this Session

- **Discussion on container systems**
  - ↪ what they are and where they help
  - ↪ common container systems
  - ↪ will focus on **Singularity** container system

The tutorial will show you...

- how to use **Singularity** containers on the UL HPC platform
  - ↪ how to build containers from a definition file
  - ↪ how to import pre-existing containers
  - ↪ how to use applications embedded in containers
- containerized parallel applications execution

Summary

A brief intro. to containers

Purpose of containers?

- **Application portability**
  - ↪ containers bundle together an entire runtime env. (OS to apps.)
  - ↪ easy replication of environments
- Services isolation
  - ↪ separate microservices in different containers
- Do more with less
  - ↪ fast instantiation and tear-down
  - ↪ little memory/CPU overhead

A brief intro. to containers

Purpose of containers?

- **Application portability**
  - ↪ containers bundle together an entire runtime env. (OS to apps.)
  - ↪ easy replication of environments
- Services isolation
  - ↪ separate microservices in different containers
- Do more with less
  - ↪ fast instantiation and tear-down
  - ↪ little memory/CPU overhead

Technology main points

- OS-level virtualization - **light virtualization**
  - ↪ don't spin up a full virtual machine
- Close to native **bare metal** speed
  - ↪ user software and libraries run on host kernel

Common container systems

- **Docker** https://www.docker.com
  - ↪ A new (2013-) take on containers (OpenVZ and LXC came before)
  - ↪ High uptake in Enterprise (microservices) & science (reproducibility)
  - ↪ In use everywhere (esp. DevOps), available on most Cloud infra.

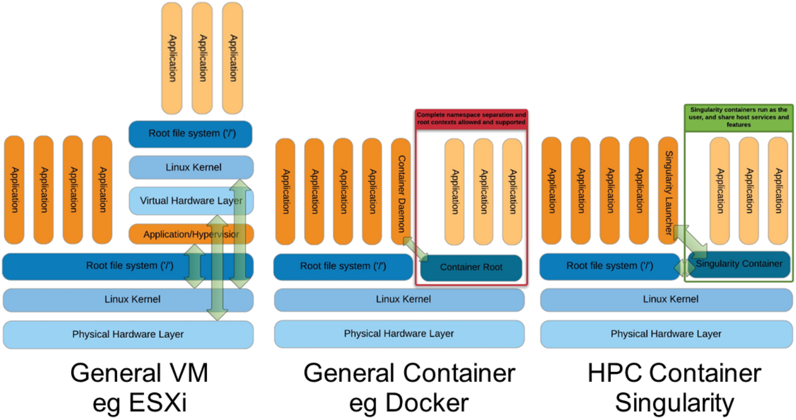
Common container systems

- **Docker** https://www.docker.com
  - ↪ A new (2013-) take on containers (OpenVZ and LXC came before)
  - ↪ High uptake in Enterprise (microservices) & science (reproducibility)
  - ↪ In use everywhere (esp. DevOps), available on most Cloud infra.
- **Shifter** https://github.com/NERSC/shifter
  - ↪ *Linux containers for HPC*, developed at NERSC
  - ↪ Uses Docker functionality but makes it safe in shared HPC systems
  - ↪ Image gateway used to convert Docker images before use

Common container systems

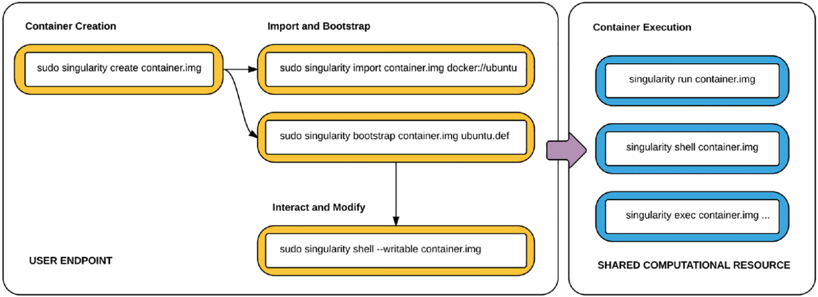
- **Docker** https://www.docker.com
  - ↪ A new (2013-) take on containers (OpenVZ and LXC came before)
  - ↪ High uptake in Enterprise (microservices) & science (reproducibility)
  - ↪ In use everywhere (esp. DevOps), available on most Cloud infra.
- **Shifter** https://github.com/NERSC/shifter
  - ↪ *Linux containers for HPC*, developed at NERSC
  - ↪ Uses Docker functionality but makes it safe in shared HPC systems
  - ↪ Image gateway used to convert Docker images before use
- **Singularity** https://github.com/singularityware/singularity
  - ↪ *Containers for science*, developed at LBNL
  - ↪ Not based on Docker, but can directly import/run Docker images
  - ↪ Also HPC oriented, diff. take to running MPI software than Shifter
  - ↪ Provides an Image Registry https://github.com/singularityhub/sregistry

High level view of containers vs full virt.



Sources:
Greg Kurtzer keynote slides at HPC Advisory Council 2017 @ Stanford (highly recommended read!)
http://geekyap.blogspot.com/2016/11/docker-vs-singularity-vs-shifter-in-hpc.html

Singularity in a nutshell



user endpoint: your workstation (admin. privileges required)
shared computational resource: UL HPC clusters

Source: Kurtzer GM, Sochat V, Bauer MW (2017) Singularity: Scientific containers for mobility of compute. PLoS ONE 12(5): e0177459

Install on your workstation - Linux (I)

- Prerequisite for full functionality

```
apt-get install squashfs-tools # Debian, Ubuntu, etc.
yum install squashfs-tools # RHEL, CentOS, etc.
```

- Installation from source

```
VERSION=2.5.1
wget https://github.com/singularityware/singularity/releases/\
download/$VERSION/singularity-$VERSION.tar.gz
tar xvf singularity-$VERSION.tar.gz
cd singularity-$VERSION
./configure --prefix=/usr/local
make
sudo make install
```

See also: http://singularity.lbl.gov/install-linux

Install on your workstation - Linux (II)

- Installation from distro. repository (Debian, Ubuntu)

```
apt-get install singularity-container
```

- Installation from self-built package (RHEL, CentOS)

```
yum install rpm-build libarchive-devel
VERSION=2.5.1
wget https://github.com/singularityware/singularity/releases/\
download/$VERSION/singularity-$VERSION.tar.gz
rpmbuild -ta singularity-$VERSION.tar.gz
sudo yum install ~/rpmbuild/RPMS/*/singularity-[0-9]*.rpm
```

Install on your workstation - macOS

- Prerequisites - install Brew, VirtualBox and Vagrant

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/\
Homebrew/install/master/install)"
brew cask install virtualbox
brew cask install vagrant
brew cask install vagrant-manager
```

- Initialize an Ubuntu VM and install Singularity inside

```
mkdir singularity-vm && cd singularity-vm
vagrant init bento/ubuntu-18.04
vagrant up --provider virtualbox
vagrant ssh -c /bin/sh <<EOF
    sudo apt-get update
    sudo apt-get -y install singularity-container
EOF
```

See also: http://singularity.lbl.gov/install-mac

Use on the UL HPC clusters

```
$> module use /opt/apps/resif/data/devel/default/modules/all
```

only needed during HPC School, in new & default software env. soon

```
$> module load tools/Singularity
```

Now that Singularity is there. . .

```
$ singularity
[...]
CONTAINER USAGE COMMANDS:
    exec        Execute a command within container
    run         Launch a runscript within container
    shell       Run a Bourne shell within container
    test        Launch a testscript within container

CONTAINER MANAGEMENT COMMANDS:
    apps        List available apps within a container
    bootstrap   *Deprecated* use build instead
    build       Build a new Singularity container
    check       Perform container lint checks
    inspect     Display container\'s metadata
    mount       Mount a Singularity container image
    pull        Pull a Singularity/Docker container to $PWD

COMMAND GROUPS:
    image       Container image command group
    instance    Persistent instance command group
```

Quick start with Singularity (I)

```
$> singularity pull docker://python:3.6.5-stretch
```

```
$> singularity exec python-3.6.5-stretch.simg python3
```

```
$> singularity shell python-3.6.5-stretch.simg
```

```
./python-3.6.5-stretch.simg
Python 3.6.5 (default, Jun  6 2018, 19:19:24)
[GCC 6.3.0 20170516] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Running python from within the container.")
```

This brought us an immutable image with Debian Stretch OS & Python 3.6.5 from the Docker Registry.
The image is not writeable, but has access to our home directory by default.

Quick start with Singularity (II)

- Sandbox mode

```
sudo singularity build --sandbox \
     python-3.6.5-stretch docker://python:3.6.5-stretch
sudo singularity exec --writable \
     python-3.6.5-stretch pip3 install numpy nose
singularity exec python-3.6.5-stretch \
     python3 -c "import numpy; numpy.test()"
```

This time the Docker Image was downloaded and unpacked to a directory (sandbox mode).
Changes within the directory can be made persistent with the *writable* flag.

Quick start with Singularity (II)

- Sandbox mode

```
sudo singularity build --sandbox \
     python-3.6.5-stretch docker://python:3.6.5-stretch
sudo singularity exec --writable \
     python-3.6.5-stretch pip3 install numpy nose
singularity exec python-3.6.5-stretch \
     python3 -c "import numpy; numpy.test()"
```

This time the Docker Image was downloaded and unpacked to a directory (sandbox mode).
Changes within the directory can be made persistent with the *writable* flag.

- Writable image (deprecated, may not work)

```
sudo singularity build --writable \
     python-3.6.5-stretch.img  docker://python:3.6.5-stretch
sudo singularity exec --writable \
     python-3.6.5-stretch.img pip3 install numpy nose
singularity exec python-3.6.5-stretch.img \
     python3 -c "import numpy; numpy.test()"
```

Quick start with Singularity (III)

Containers' access to the HPC filesystem(s)

- Home directories are bind mounted by default
- Your user(name) and group(s) are dynamically added
  - ↪ thus files created maintain normal permissions
- Other paths need to be explicitly set

```
sudo singularity build custom.simg python-3.6.5-stretch/
singularity exec --bind /work/projects/myprj/:/mnt \
            custom.simg python3 /mnt/my_nice_code.py
singularity exec --bind /work/projects/myprj:/work/projects/myprj \
            --bind /scratch/users/$USER:/scratch/users/$USER \
            custom.simg python3 /work/projects/myprj/nice_code.py -o \
            /scratch/users/$USER/output_dir/
```

With the first command we create a compressed, SquashFS immutable image from the sandbox folder. Then, we run the python3 interpreter from this image on code and data existing outside the container.

Building containers from scratch (I)

A minimal container definition file

centos7-custom.def

```
BootStrap: yum
OSVersion: 7
MirrorURL: http://mirror.centos.org/centos-%{OSVERSION}/
                         %{OSVERSION}/os/$basearch/
Include: yum

%runscript
    exec "python3" "$@"

%post
    echo "==== Installing Python 3.4 + Jupyter in the container."
    yum -y install epel-release
    yum -y install python34 python34-pip
    pip3 install jupyter
```

Building containers from scratch (II)

A minimal container definition file

ubuntu-custom.def

```
BootStrap: debootstrap
OSVersion: xenial
MirrorURL: http://eu.archive.ubuntu.com/ubuntu/
Include: software-properties-common

%runscript
    exec "python3" "$@"

%post
    echo "==== Installing Python 3.5 + Tensorflow."
    add-apt-repository universe
    apt-get update
    apt-get install -y python3 python3-pip
    export LC_ALL=C
    python3 -m pip install tensorflow
```

Building containers from scratch (III)

```
$> sudo singularity build -sandbox sandbox_dir template.def
```

```
$> sudo singularity shell -writable sandbox_dir
```

```
$> sudo singularity build production_image.simg sandbox_dir
```

```
$> singularity exec production_image.simg python3 nice_code.py
```

Containers with MPI support (I)

```
BootStrap: yum
OSVersion: 7
MirrorURL: http://mirror.centos.org/centos-%{OSVERSION}/
                 %{OSVERSION}/os/$basearch/
Include: yum wget

%post
    yum groupinstall -y "Development Tools" "Infiniband Support"
    wget https://www.open-mpi.org/software/ompi/v2.1/\
                downloads/openmpi-2.1.3.tar.bz2
    tar xf openmpi-2.1.3.tar.bz2 && cd openmpi-2.1.3
    ./configure --prefix=/usr/local --enable-shared
                --enable-mpi-thread-multiple\
                --enable-mpirun-prefix-by-default\
                --without-ucx --disable-pmix-dstore\
                --with-pmix=internal
    make && make install
    mpicc examples/ring_c.c -o /usr/local/bin/mpi_ring
```

Containers with MPI support (II)

```
$> sudo singularity build mpi-ex.simg mpi-ex.def
```

```
$> module use /opt/apps/resif/data/devel/default/modules/all
```

```
$> module load toolchain/foss tools/Singularity
```

```
$> mpirun singularity exec mpi-ex.simg /usr/local/bin/mpi_ring
```

Recall: the build happens on your workstation, execution on the HPC clusters

Conclusion and Practical Session start

We've discussed

1. setting up Singularity on your workstation
2. common Singularity commands
3. how to download existing Docker registry images
4. how to create and customize containers locally
5. how to run Singularity containers on the UL HPC platform

And now..

Short DEMO time!

Conclusion and Practical Session start

We've discussed

1. setting up Singularity on your workstation
2. common Singularity commands
3. how to download existing Docker registry images
4. how to create and customize containers locally
5. how to run Singularity containers on the UL HPC platform

And now..

Short DEMO time!

Your Turn!

Questions?

High Performance Computing @ uni.lu

Prof. Pascal Bouvry
Dr. Sebastien Varrette
Valentin Plugaru
Sarah Peter
Hyacinthe Cartiaux
Clement Parisot

University of Luxembourg, Belval Campus
Maison du Nombre, 4th floor
2, avenue de l'Université
L-4365 Esch-sur-Alzette
mail: hpc@uni.lu