Uni.lu HPC School 2019
PS08: 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/`
PS08 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.

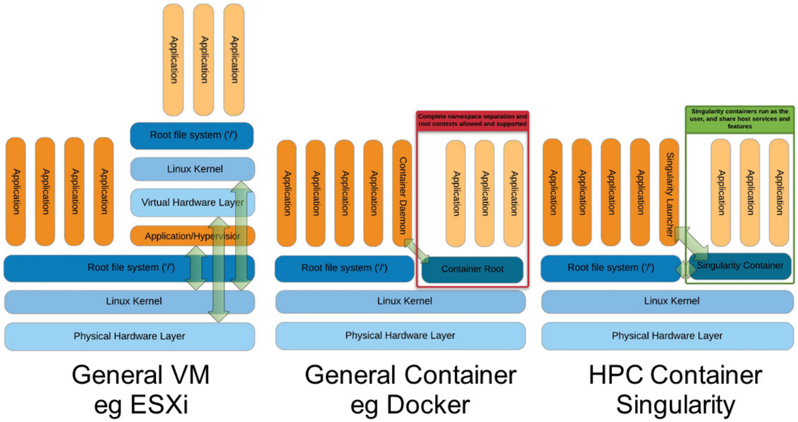
Common container systems

- **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

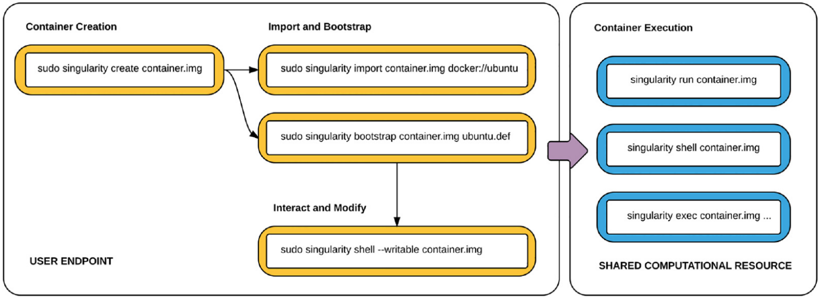
- **Docker** https://www.docker.com
  - ↪ A new (2013-) take on containers (OpenVZ and LXC came before)
  - ↪ High uptake in Enterprise (microservices) & science (reproducibility)
  - ↪ In use everywhere (esp. DevOps), available on most Cloud infra.
- **Shifter** https://github.com/NERSC/shifter
  -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


Container Creation
sudo singularity create container.img
Import and Bootstrap
sudo singularity import container.img docker://ubuntu
sudo singularity bootstrap container.img ubuntu.def
Interact and Modify
sudo singularity shell --writable container.img
USER ENDPOINT
Container Execution
singularity run container.img
singularity shell container.img
singularity exec container.img ...
SHARED COMPUTATIONAL RESOURCE

Many changes in newest v3 Singularity but workflow still similar.

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

- Debian & Ubuntu as of June 2019:
  - ↪ newer (not newest) Singularity 3.1.1 in Debian Sid
  - ↪ older Singularity 3.0.3 in Debian Buster (next stable in July 2019)
  - ↪ very old Singularity 2.6.1 in Ubuntu 19.04 - Disco

```
sudo apt-get update
sudo apt-get install singularity-container
```

- CentOS & RHEL: installation from EPEL
  - ↪ Singularity 3.2.1 as of June 2019 for CentOS/RHEL 7

```
sudo yum update -y
sudo yum install -y epel-release
sudo yum update -y
sudo yum install -y singularity-runtime singularity
```

See also: `https://sylabs.io/guides/3.0/user-guide/installation.html#install-on-linux`

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
export VM=sylabs/singularity-3.2-ubuntu-bionic64
vagrant init $VM
vagrant up
vagrant ssh
EOF
```

See also: https://sylabs.io/guides/3.0/user-guide/installation.html#install-on-windows-or-mac

Use on the UL HPC clusters

```
$> module load swenv/default-env/devel
```

only needed during HPC School, part of 2019 software env. soon

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

Now that Singularity is there. . .

```
$ singularity
Usage: singularity [global options...] <command>
Available Commands:
  apps        List available apps within a container
  build       Build a Singularity image
  cache       Manage the local cache
  exec        Run a command within a container
  inspect     Show metadata for an image
  instance    Manage containers running as services
  pull        Pull an image from a URI
  push        Upload image to the provided library (def:"cloud.sylabs.io")
  remote      Manage singularity remote endpoints
  run         Run the user-defined default command within a container
  run-help    Show the user-defined help for an image
  search      Search a Library for images
  shell       Run a shell within a container
  sign        Attach a cryptographic signature to an image
  test        Run the user-defined tests within a container
  verify      Verify cryptographic signatures attached to an image
[...]
```

Quick start with Singularity (I)

```
$> singularity pull docker://python:3.8.0b1-alpine3.9
```

```
$> singularity exec python_3.8.0b1-alpine3.9.sif python3
```

```
$> singularity shell python_3.8.0b1-alpine3.9.sif
```

```
./python_3.8.0b1-alpine3.9.sif
Python 3.8.0b1 (default, Jun  5 2019, 23:34:27)
[GCC 8.3.0] 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 (tiny) Alpine Linux & Python 3.8.0b1 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: container development

```
sudo singularity build --sandbox \
     python_3.7.3-stretch docker://python:3.7.3-stretch
sudo singularity exec --writable \
     python_3.7.3-stretch/ pip3 install numpy nose test
singularity exec python_3.7.3-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: container development

```
sudo singularity build --sandbox \
     python_3.7.3-stretch docker://python:3.7.3-stretch
sudo singularity exec --writable \
     python_3.7.3-stretch/ pip3 install numpy nose test
singularity exec python_3.7.3-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.

- Production image (default build mode)
  - ↪ done when we know steps to install software & customize image
  - ↪ customization commands in a **definition** file (more details later)

```
sudo singularity build image.sif recipe.def
```

Now image can be transferred, e.g. to the Iris cluster and used normally.

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.sif python_3.7.3-stretch/
singularity exec --bind /work/projects/myprj/:/mnt \
            custom.sif python3 /mnt/my_nice_code.py
singularity exec --bind /work/projects/myprj:/work/projects/myprj \
            --bind /scratch/users/$USER:/scratch/users/$USER \
            custom.sif python3 /work/projects/myprj/nice_code.py -o \
            /scratch/users/$USER/output_dir/
```

With the first command we create a compressed, **SIF - Singularity Image File** from the sandbox folder.
Then, we run the python3 interpreter from this image on code and data existing outside the container.
More details on SIF: https://archive.sylabs.io/2018/03/sif-containing-your-containers/

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 3.6 + Jupyter in the container."
    yum -y install epel-release
    yum -y install python36 python36-pip
    pip3 install jupyter
```

Building containers from scratch (II)

A minimal container definition file

ubuntu-custom.def

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

%runscript
    exec "python3" "$@"

%post
    echo "==== Installing Python 3.7 + 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.sif sandbox_dir
```

```
$> singularity exec production_image.sif 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"
    yum install rdma-core-devel
    wget https://www.open-mpi.org/software/ompi/v3.1/\
                downloads/openmpi-3.1.3.tar.bz2
    tar xf openmpi-3.1.3.tar.bz2 && cd openmpi-3.1.3
    ./configure --prefix=/usr/local --enable-shared
                --enable-mpirun-prefix-by-default\
                --with-verbs --with-pmix:w
    make && make install
    mpicc examples/ring_c.c -o /usr/local/bin/mpi_ring
```

Containers with MPI support (II)

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

```
$> module load swenv/default-env/devel
```

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

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

Recall: build on your workstation, run on the Iris cluster

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!

Thank you for your attention...

Questions?

http://hpc.uni.lu

High Performance Computing @ uni.lu

Prof. Pascal Bouvry
Dr. Sebastien Varrette
Valentin Plugaru
Sarah Peter
Hyacinthe Cartiaux
Clement Parisot
Dr. Fréderic Pinel
Dr. Emmanuel Kieffer

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