Singularity & Containers

Evan F. Bollig

11/14/2019
Outline

- Intro to Containers
- Docker vs Singularity
- Getting started with Singularity
  - Interactive
  - Batch
- Creating and Accessing Images
- Mapping Volumes
- Advanced Examples
Intro to Containers
Containers

Containers are an encapsulation of system environments, and a means to use it.

Source: Greg Kurtzer keynote at HPC Advisory Council 2017 @ Stanford
Containers

Containers are an encapsulation of system environments, and a means to use it.

Source: [Greg Kurtzer keynote at HPC Advisory Council 2017 @ Stanford](#)
Containers in Practice

Designed for Enterprise to encapsulate and virtualize micro-services
Containers in Practice

Designed for **Enterprise** to encapsulate and virtualize **micro-services**

Source: Greg Kurtzer keynote at HPC Advisory Council 2017 @ Stanford
Our Use Case

Our need and use-case is opposite of Enterprise
Our Use Case

Our need and use-case is opposite of Enterprise.
Our Use Cases

Our needs and use-cases are opposite of Enterprise

- Scientists need jobs to run.
  - ASAP! Right Now! Last week!
- Scientists need compute resources
  - Whatever is available!
- Scientists need applications to work
  - My grad student got this working 10 years ago! These authors already did it, so I’m sure it is easy to reproduce!

*Containers provide: application and environment compatibility*
Docker vs Singularity
So what about Docker!?

- Docker is the most well known and utilized container platform
- Designed primarily for network micro-service virtualization
- Facilitates creating, maintaining and distributing container images
- Containers are kinda reproducible
- Easy to install, well documented, standardized

For these reasons, it appears to be the answer to life, the universe and everything!

And many scientists have jumped on the bandwagon to made use of it
So why not just keep using Docker?

The good news:

You can! It works great for local and private resources. You can use it to develop and share your work with others using Docker-hub.

The bad news:

If you ever need to scale beyond your local resources, it maybe a dead end path! Docker, and other enterprise focused containers, are not designed for, efficient or even compatible with traditional HPC.

No HPC centers allow it!
What is Singularity?

A *containerization platform* developed at the Lawrence Berkeley National Laboratory.

While Docker assumes (and essentially requires) its users have elevated privileges on its host systems, Singularity was *designed for use in high-performance computing environments* in which the protection of shared resources is paramount.

Compared to Docker:

- **Your container runs as you.**
  - No need for "sudo" to run singularity
  - The user executing commands *inside* your container is you as well.

- **No user context switching inside containers:** "sudo" doesn't work in Singularity.
How does the Architecture compare?

Getting Started with Singularity
Singularity Workflow

Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific Containers for Mobility of Compute (under review)
Interactive Use

**singularity** works exactly like any other command

1. Get an Interactive Session:
   
   ```
   qsub -I -l walltime=30:00,nodes=1:ppn=2,mem=4gb -q small
   ```

2. Load the Module
   
   ```
   module load singularity
   ```

3. Use it!
   
   ```
   singularity run library://sylabsed/examples/lolcow
   ```

This requires CentOS 7. It will **NOT** work on Itasca.
Use Mesabi and the Lab/Interactive Cluster.
Follow MSI's standard format for PBS jobs:

```bash
#PBS -l walltime=0:30:00,mem=4GB,nodes=1:ppn=2
#PBS -q small
#PBS -m abe
#PBS -M boll0107@umn.edu
#PBS -W umask=0007
#PBS -A bollige

module load matlab
module load singularity

singularity run library://sylabsed/examples/lolcow
```

Running MPI in containers is possible, but may require special built containers to match the HPC environment (e.g., libverbs for MPI over Infiniband). See [https://singularity.lbl.gov/docs-hpc](https://singularity.lbl.gov/docs-hpc) for details.
Batch

If MPI is built into your container:

```
#PBS -l walltime=0:30:00,mem=4GB,nodes=1:ppn=12
#PBS -q small
#PBS -m abe
#PBS -M boll0107@umn.edu
#PBS -W umask=0007
#PBS -A bollige

module load singularity

singularity exec library://dev/image mpirun -np $PBS_NP -hostfile $PBS_NODEFILE command
```

This runs one container on one node with MPI inside the container.
Batch

If MPI is built into your container:

```bash
#PBS -l walltime=0:30:00,mem=4GB,nodes=12:ppn=1
#PBS -q small
#PBS -m abe
#PBS -M boll0107@umn.edu
#PBS -W umask=0007
#PBS -A bollige

module load singularity

mpirun -np $PBS_NP -hostfile $PBS_NODEFILE singularity exec library://dev/image command
```

This runs multiple containers (one on each node) with one process inside each container.
Run a container based on the provided image:

```
 singularity run shub://GodloveD/lolcow
 singularity run docker://library/python
 singularity run ./my-local-image.simg
```

shub:// = Singularity Hub
docker:// = Docker Hub

More on these later...
PULL

Cache the image on your local filesystem to improve startup rates.

```bash
singularity run library://sylabsed/examples/lolcow
```

Versus:

```bash
singularity pull library://sylabsed/examples/lolcow:latest

singularity run ./lolcow_latest.sif
```

NOTE: Images count against your Quota!
EXEC

Bypass the default use script and execute a command inside the container.

```
singularity exec path_to_container command_goes_here so_do_parameters
```

```
singularity exec ./lolcow_latest.sif date
```
SHELL

Launch a BASH shell in the container for interactive use

```
singularity shell ./lolcow_latest.sif
Singularity lolcow_latest.sif:~> date
Singularity lolcow_latest.sif:~> df
```
Mapping Volumes
Volume?

A **Volume** is a storage. This name comes from the Enterprise use-case.

Volumes = Directories.

Binding/Mapping?

You can bind/map directories from the **Host** machine into a **Guest** container

```
singularity run -B /host/path:/guest/path my_images/coolimagel.simg
```
Map Example: Accessing Software Modules

Use the module command on the host to get tips on what environment variables need to be set inside the container

```
module show <MODULE>
```

Then map the module path:

```
singularity exec -B /path/to/module:/opt/module docker://ubuntu:latest
```
Creating and Accessing Images
**DISABLED** root == **MISSING** Singularity Commands

<table>
<thead>
<tr>
<th>CONTAINER MANAGEMENT COMMANDS (requires root):</th>
</tr>
</thead>
<tbody>
<tr>
<td>bootstrap</td>
</tr>
<tr>
<td>copy</td>
</tr>
<tr>
<td>create</td>
</tr>
<tr>
<td>expand</td>
</tr>
<tr>
<td>export</td>
</tr>
<tr>
<td>import</td>
</tr>
<tr>
<td>mount</td>
</tr>
</tbody>
</table>

MSI does not provide root/sudo for any user. **These are purposely disabled.**

If you need these commands, install and run Singularity on your own machine or a virtual machine in VirtualBox, OpenStack, AWS, etc. See, e.g. [https://msi.umn.edu/stratus](https://msi.umn.edu/stratus)
Docker Hub

https://hub.docker.com/

Not all images will work with singularity.

- Entrypoint scripts are usually broken.
- Expectations of root cannot be met.
- Prefix: `docker://`
Singularity Hub

https://singularity-hub.org/

MSI is running Singularity version 3.4.

- **Con:**
  - Requires tight Github coupling
  - No longer recommended
- **Pro:** Images should all work!
- **Prefix:** `shub://`
Sylabs Cloud

https://singularity-hub.org/

The Singularity Community fully backs this project

- Images should all work!
- Enables remote building!
- Enables image signing

- Prefix: `library://`
Other Image Sources

Store and share your images to avoid redundant storage consumption:

/home/$GROUP/shared/  -- Share with your Group

/home/$GROUP/public/   -- Share with all of MSI

Artifactories also work (e.g., artifactory.umn.edu), but authentication might require you to export the **SINGULARITY_DOCKER_PASSWORD** environment variable or CLI argument.

Remember to `chmod g+rX` or `chmod go+rX` your images!
Singularity can now build without root!

Build is a “swiss army knife” command:

```bash
singularity build lolcow.sif library://sylabsed/examples/lolcow
```

Singularity remote login

```bash
singularity build --remote file-out.sif \
   docker://ubuntu:18.04
```

NOTE: Images count against your Quota when downloaded!
Sandbox

Another use-case for build: local sandbox of image:

```bash
singularity build --sandbox ubuntu/ library://ubuntu

singularity build new-image.sif ./ubuntu
```
Build from Definition File

BootStrap: library
From: ubuntu:16.04

%post
  apt-get -y update
  apt-get -y install fortune cowsay lolcat

%environment
  export LC_ALL=C
  export PATH=/usr/games:$PATH

%runscript
  fortune | cowsay | lolcat

%labels
  Author GodloveD

Another use-case for build:
local sandbox of image:

```
singularity build \   
  --remote \ 
  mycow.sif \ 
  ./mycow.def
```
Signing Images (This is BETA for MSI:)

`singularity remote login`

`singularity build --remote mynewcow.sif ./mycow.def`

`singularity key newpair`

`singularity sign mynewcow.sif`

`singularity verify mynewcow.sif`
Verify Images (This is BETA for MSI)

```
singularity pull \library://centos
```

```
singularity verify \centos_latest.sif
```

![Terminal](image)
Advanced Examples
Quick Reference

- Run container from Docker Hub:
  ```bash
  singularity run docker://publishing_user/container_name
  ```

- Running container from Singularity Hub:
  ```bash
  singularity run shub://publishing_user/container_name
  ```

- Running a local container image:
  ```bash
  singularity run path_to_container
  ```

- Passing a command to a container:
  ```bash
  singularity exec path_to_container command_goes_here so_do_parameters
  ```

- Mounting a directory in a container:
  ```bash
  singularity run -B path_on_host:path_in_container
  shub://publishing_user/container_name
  ```
Hands-on Challenge

1. Pick your favorite software module
2. Search for a container image on DockerHub/SingularityHub/SylabsCloud
3. Pull the image into your group Shared directory
   a. Remember to chmod it for the group
4. Run the container
   a. Write a file to your home directory. Does it persist when you exit the container?
5. Shell into the container to see what the Guest environment looks like
6. Exec an alternate command
7. Map a directory to a new mount point.
   a. Write a file and see if it persists.
BioTools

Do any of these apply?

- Tools are difficult to install or have many dependencies
- Tools are updated frequently or locked versions are needed
- Tools are intended to run on Ubuntu but MSI only offers CentOS

https://hub.docker.com/u/biocontainers
http://singularity-hub.org/collections/630

- docker://biocontainers/canu
- docker://biocontainers/bwa:0.7.15
- shub://qbicsoftware/qbic-singularity-qiime:latest
MPI Jobs

To span multiple nodes:

- A container is limited to one node.
- MPIrun wraps singularity.
- Singularity image needs MPI baked into it.
- Leverage variables from your PBS script.

```
module load ompi singularity
singularity search openmpi
singularity exec library://paul/bil/openmpi $PWD/mpi_test.exe
mpirun -np $PBS_NP singularity exec library://paul/bil/openmpi \
    $PWD/mpi_test.exe
```
Keras and Tensorflow

Keras is a machine learning tool that can use multiple backends.

One of the backends is Tensorflow, which can also be used directly for lower-level machine learning tasks. Tensorflow can take advantage of GPU resources on the K40 nodes at MSI.
TensorFlow using Singularity

export sdir=/home/support/public/tutorials/singularity

cp $sdir/mnist.py .

singularity pull \
    shub://schanzel/singularity-tensorflow-keras-py3:latest

singularity exec \
    ./schanzel-singularity-tensorflow-keras-py3-master-latest.simg\n
    python mnist.py
TensorFlow using Singularity with GPUs

Within a Singularity container, we can get access to the GPU resources on the host as long as we use the “--nv” flag.

```bash
export simg=schanzel-singularity-tensorflow-keras-gpu-py3-master-latest.simg

singularity exec --nv $sdir/$simg python mnist.py
```

Note: this requires you to be on a GPU node. Request an interactive session with:

```bash
qsub -I -l nodes=1:ppn=24:gpus=2,walltime=20:00 -q k40
```
How do I get Help?

Email: help@msi.umn.edu

Call: 612 626-0802  or  Campus Phone: 6-0802

Hours: 9 am - 4pm, M-F