Managing Research Workflows with Singularity Containers: Part 2

Andrew Monaghan
andrew.monaghan@Colorado.edu
https://www.rc.colorado.edu

Slides available for download at
https://github.com/rctraining/Singularity_Short_Course_Spring_2018

Adapted from a presentation by Martin Cuma (U. Utah; 1) and materials from https://singularity.lbl.gov/ and https://www.singularity-hub.org

Outline

- Review of fundamentals
- Moving containers to Summit
- Hands-on Examples
    - Loading Singularity and related modules
    - Running containers from Singularity & Docker Hubs
    - Binding directories
    - Running a multi-core job with a container
- Troubleshooting
- Summary

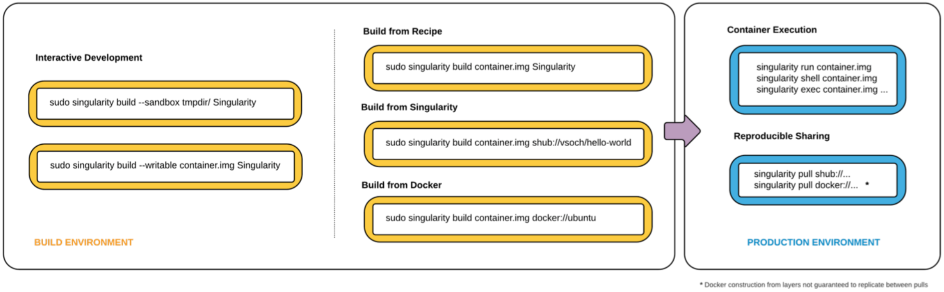
Questions from last week

1. Are there ever host/container incompatibilities?
    - Yes, there are. The known incompatibilities are covered at the end of this presentation.

2. Can I used licensed software in a singularity container?
    - There is little on this topic. However, it seems possible:
        - See Matlab examples on Singularity Hub
        - Also, some groups have noted Singularity as a solution for running proprietary software that only runs on specific Linux distributions.
    - Process: I **think** you would install the software in the container (doesn't usually require a license), and then establish a means to access an external license file or server so that the software can be used.
    - Likely not universally portable due to license requirement.

Review of Containers

The Singularity workflow



Interactive Development

```
sudo singularity build --sandbox tmpdir/ Singularity
sudo singularity build --writable container.img Singularity
```

Build from Recipe

```
sudo singularity build container.img Singularity
```

Build from Singularity

```
sudo singularity build container.img shub://vsoch/hello-world
```

Build from Docker

```
sudo singularity build container.img docker://ubuntu
```

BUILD ENVIRONMENT

Container Execution

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

Reproducible Sharing

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

PRODUCTION ENVIRONMENT

* Docker construction from layers not guaranteed to replicate between pulls

https://singularity.lbl.gov

Running containers

Now, on ***any system*** with Singularity, even without administrative privilege, you can retrieve and use containers:

- Download a container from Singularity Hub or Docker Hub
  - `singularity pull shub://some_image`
  - `singularity pull docker://some_image`

- Run a container
  - `singularity run mycont.img`

- Execute a specific program within a container
  - `singularity exec mycont.img python myscript.py`

- “Shell” into a container to use or look around
  - `singularity shell mycont.img`

- Inspect an image
  - `singularity inspect --runscript mycont.img`

Moving Containers

Moving containers

You've built your first container on your laptop. It is 3 Gigabytes. Now you want to move it to Summit to take advantage of the HPC resources. What's the best way?

Remember, containers are files, so you can transfer them to Summit just as you would a file:

- Command line utilities (scp, sftp)
- Globus (example follows)
- For more on data transfers to/from Summit: https://github.com/ResearchComputing/Research-Computing-User-Tutorials/wiki/Data-Transfers

Hands-on Examples

Logging in

If you are using a temporary account: In a terminal or Git Bash window, type the following:

```
ssh -X userNNNN@tutorial-login.rc.colorado.edu

Password:
```

***Note that `userNNNN` is a temporary account that I have assigned to you. If you don't have one, let me know.*

If you are using your regular RC account (e.g., 'monaghaa' for me), log in as you normally would.

Navigating

Now type the following commands:

```
ssh -X scompile

cd /scratch/summit/<your_username>

cp -r /scratch/summit/monaghaa/singularity_sc .

cd singularity_sc

ls
```

Getting on a compute node

Now start an interactive job:

```
sinteractive –ntasks=2 --reservation=tutorial1
```

** also add `--account=ucb-general` if logged into your own account

…And load singularity

```
module load singularity
```

Building a container from Singularity Hub

Pull our example from last week:

```
singularity pull –name “wow.img”
shub://monaghaa/singularity_git_tutorial
```

…And run it

```
singularity run wow.img
```

…And look at the script inside

```
singularity exec wow.img cat /code/hello.sh
```

Building a container from Docker Hub

Let's grab the stock docker python container:

```
singularity pull –name "pythond.img" docker://python
```

…And run python

```
singularity exec pythond.img python
```

…And shell into the container and look around

```
singularity shell pythond.img
```

…try `"ls /"` What directories do you see?

Binding directories

On Summit, most host directories are "bound" (mounted) by default. But on other systems, or in some instances on Summit, you may want to access a directory that is not already mounted. Let's try it.

Note that the "/opt" directory in "pythond.img" is empty. But the Summit "/opt" directory is not. Let's bind it:

```
singularity shell –bind /opt:/opt pythond.img
```

…It isn't necessary to bind like-named directories like we did above. Try binding your /home/$USER directory to /opt.

***Note: If your host system does not allow binding, you will need to create the host directories you want mounted when you build the container (as root on, e.g., your laptop)

Running a multi-core job with a container

Syntax (after loading singularity and openmpi):

```
“mpirun –np X singularity exec container.img executable-in-container”
```

1. mpirun is called by the resource manager or the user directly from a shell
2. Open MPI then calls the process management daemon (ORTED)
3. The ORTED process launches the Singularity container requested by the mpirun command
4. Singularity builds the container and namespace environment
5. Singularity then launches the MPI application within the container
6. The MPI application launches and loads the Open MPI libraries
7. The Open MPI libraries connect back to the ORTED process via the Process Management Interface (PMI)
8. At this point the processes within the container run as they would normally directly on the host.

http://singularity.lbl.gov/docs-hpc

Running a multi-core job with a container

The container “mpitest.img” is in your working directory. Let’s run it:

```
module purge

module load singularity gcc openmpi

mpirun -np 4  -mca btl ^openib singularity exec
            mpitest.img /mpi-hello
```

…Now let’s see what version of openmpi is running in the container and on the host:

```
singularity exec mpitest.img mpirun --version

mpirun --version
```

Troubleshooting & Caveats

Troubleshooting (1)

Container/host environment conflicts

- Container problems are often linked with how the container “sees” the host system. Common issues:
    - The container doesn’t have a bind point to a directory you need to read from / write to
    - The container will “see” python libraries installed in your home directory (and perhaps the same is true for R and other packages. If this happens, set the PYTHONPATH environment variable in your job script so that it points to the container paths first.
        - `export PYTHONPATH=<path-to-container-libs>:$PYTHONPATH`

- To diagnose the issues noted above, as well as others, “shelling in” to the container is a great way to see what’s going on inside. Also, look in the singularity.conf file for system settings (can’t modify).

Troubleshooting (2)

Compatibility issues

- Open MPI version issues. General advice as of now is to use the same or a newer version of MPI in the container compared to host.
    - If you have an older version of Open MPI on the host system (e.g., 1.10) → then use the same version of Open MPI in your container.
    - Some nuances based on the communications fabric used on host. Summit uses Intel Omni-Path.
    - We will soon post some best practices for optimizing use of the Summit versions of MPI and the Omni-Path fabric

- GLIBC forward compatibility (e.g., running a Centos-7 container on a Centos-5 host may cause issues)

- For more troubleshooting, see: http://singularity.lbl.gov/faq#troubleshooting

Caveats

We didn’t cover overlays. These are additional images that are ”laid” on top of existing images, enabling the user to modify a container environment without modifying the actual container. Useful because:

1. Overlay images enable users to modify a container environment even if they don’t have root access (though changes disappear after session)
2. Root users can permanently modify overlay images without modifying the underlying image.
3. Overlays are a likely way to customize images for different HPC environments without changing the underlying images.
4. More on overlays: http://singularity.lbl.gov/docs-overlay

We didn’t cover GPU usage with containers. See:
http://singularity.lbl.gov/archive/docs/v2-3/tutorial-gpu-drivers-open-mpi-mtls

Thank you!

Please fill out the survey:
http://tinyurl.com/curc-survey16

My contact information:
Andrew.Monaghan@Colorado.edu

Additional learning resources:
Slides and Examples from this course:
https://github.com/rctraining/Singularity_Short_Course_Spring_2018

Web resources:
https://singularity.lbl.gov/user-guide *(user guide for Singularity)*
https://www.singularity-hub.org/ *(Singularity Hub)*