Singularity Containers

Burak Himmetoglu
ETS & CSC
bhimmetoglu@ucsb.edu

10/26/2017

What is a Container?

A container is an encapsulation of system environments

From Greg Kutzer

You can add:

- Operating system
- Software packages
- Database server software
- …

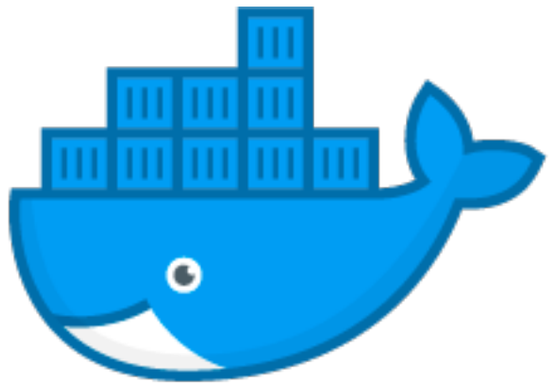
In the container and ship it!

Why containers are useful?

- Reproducibility
- Solving dependency problems
- Portability between systems
- One file contains everything to do your work
- You can bring your own container on a remote server
- No root privileges needed to run

Why Singularity?

vs.



docker

- Docker has problems on HPC systems..
- Singularity integrates well with resource managers, respects root privileges...

Setting Up

- A local Linux installation is needed to build containers
- If you are a Mac/PC users, you can easily setup a virtual machine (e.g. VirtualBox) running your favorite Linux distro

Singularity:	http://singularity.lbl.gov/
VirtualBox:	https://www.virtualbox.org/wiki/Downloads
Tutorial: Setting up Ubuntu on VirtualBox	https://linus.nci.nih.gov/bdge/installUbuntu.html

On Knot

Update your environment

```
# Singularity
export PATH=/sw/csc/singularity/bin/:$PATH
export LD_LIBRARY_PATH=/sw/csc/singularity/lib/singularity:$LD_LIBRARY_PATH
```

Building containers

Example:

You are using your local system, so you have root access:

```
$ sudo singularity create ubuntu.img
$ sudo singularity expand –size 8192 ubuntu.img
$ sudo singularity bootstrap ubuntu.img ubuntu.def
```

Create an image file

Expand its size to 8gb if needed

Bootstrap using options in file

Your container will run on a host system

The container can access your home directory on host

Bootstrap file

Anatomy of the bootstrap file:

```
% setup
  # Commands to be executed on host outside container during bootstrap

% post
  # Commands to be executed inside the container during bootstrap

% runscript
  # Commands to be executed when the container “runs”

%test
  # Commands to be executed within the container at close of bootstrap process
```

Using the image

```
$ singularity shell <path_to_img_file>
```

Run the container in shell mode (interactive)

```
$ singularity exec <path_to_img_file>  executable.x
```

Execute a program using the container.
You can place this in a job submission file to submit into the queue

```
$ singularity run <path_to_img_file>
```

Runs the commands in the %runscript

Example: Ubuntu with Python packages

The bootstrap file:

```
% post
   # Commands to be executed inside the container during bootstrap
   apt-get -y update
   apt-get -y install gfortran
   apt-get install -y —force-yes make
   apt-get install vim wget python python-pip git
   apt-get install -y --force-yes python-dev python-numpy python-matplotlib python-h5py
   apt-get install -y --force-yes python-setuptools
   pip install --upgrade pip
   pip install scipy
   pip install pandas
   pip install tables
   pip install pyparsing
   pip install scikit-learn
```

Build the container:

```
$ sudo rm -f ubuntu.img
$ sudo singularity create ubuntu.img
$ sudo singularity expand --size 8192 ubuntu.img
$ sudo singularity bootstrap ubuntu.img ubuntu.def
```

Example: Ubuntu with Python packages

- The container can access your home directory on host
- But not the ones lower in the directory tree
- What if you need to access to a shared lower level directory?

Bind that directory during or after setup:

```
$ sudo singularity exec -B `pwd`:/mnt -w ubuntu.img mkdir /local_scratch
```

- Can you make changes in the container?

Yes, but not recommended: Build it writable, changes will be saved

```
$ sudo singularity build —writable ubuntu.img
$ sudo singularity shell —writable ubuntu.img
```

Further Resources

Online Documentation:

http://singularity.lbl.gov/quickstart

Check the repo for examples:

https://github.com/bhimmetoglu/CSC-Computing-2017