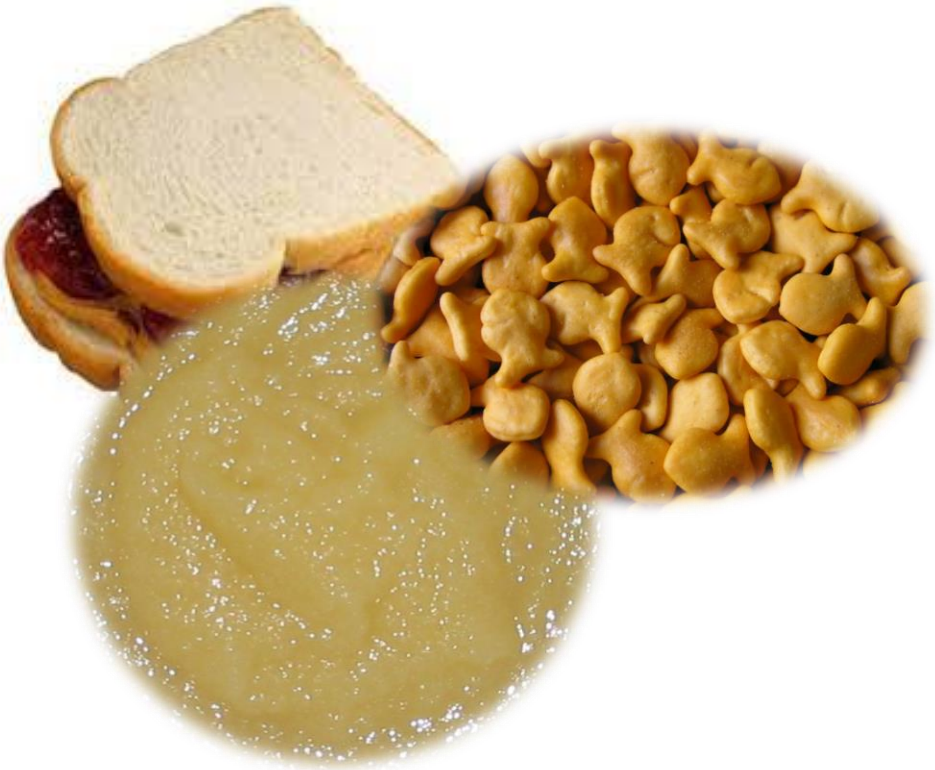
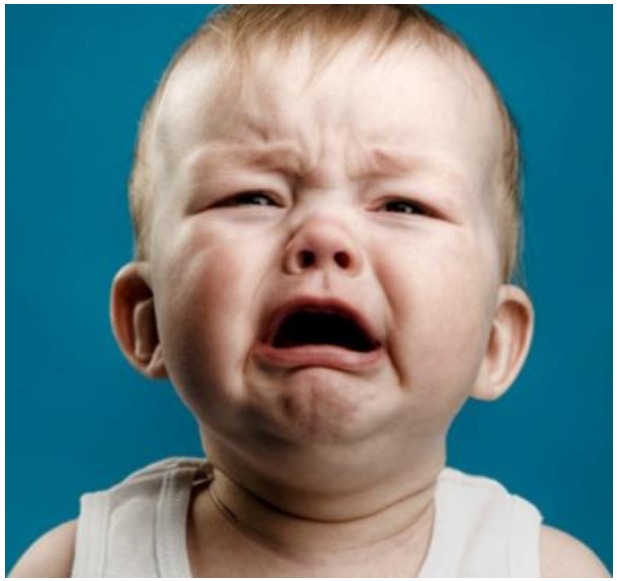
This Morning….

- What is a container & why should I care?
- What is Singularity? Is it like Docker?
- Fetching & running containers on BioHPC
- Building containers for BioHPC

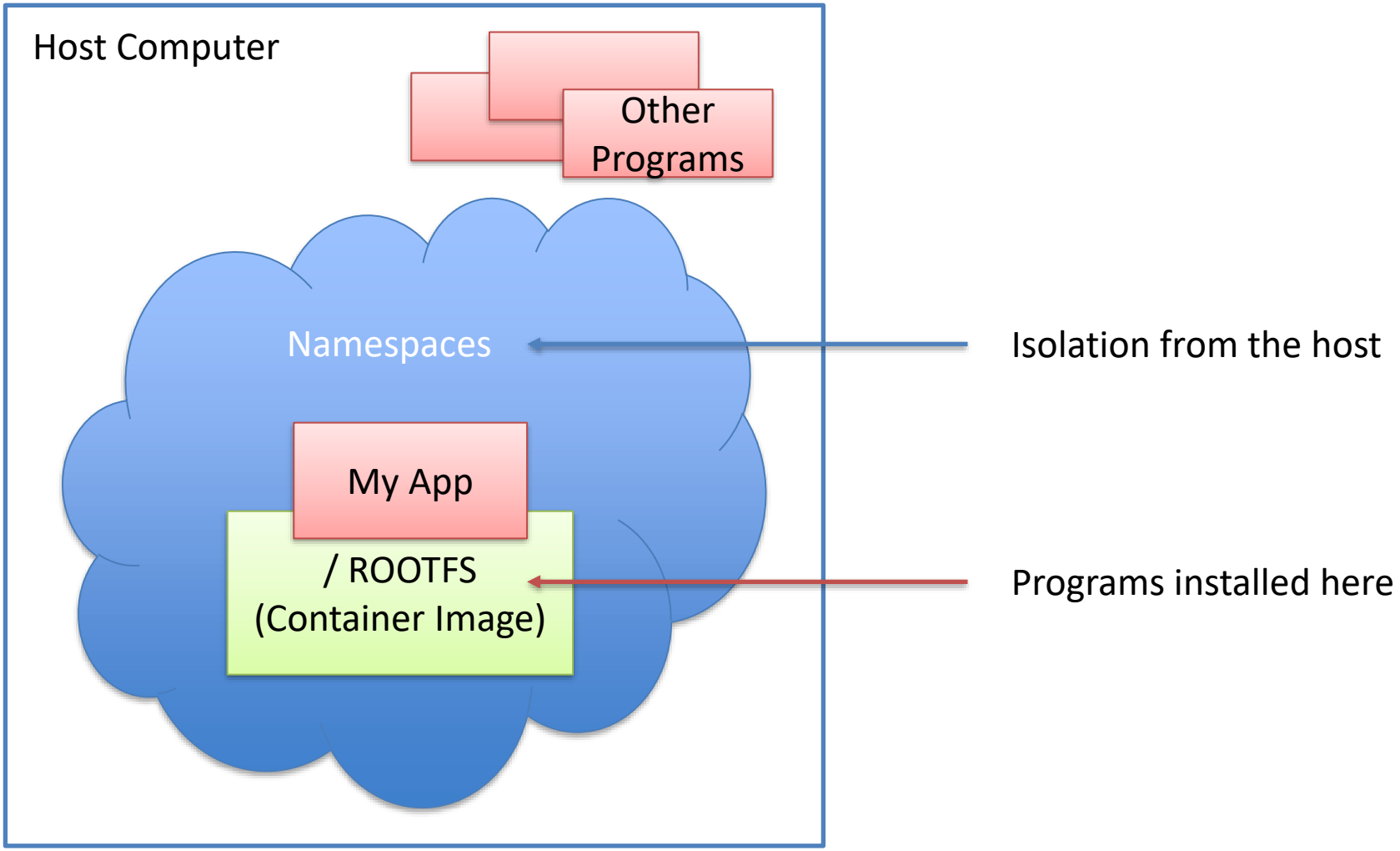
What is a Container?





What is a Container?

What is a Container?



What is a Container? – RootFS / Container Image

Container Image / RootFS looks like a Linux distribution

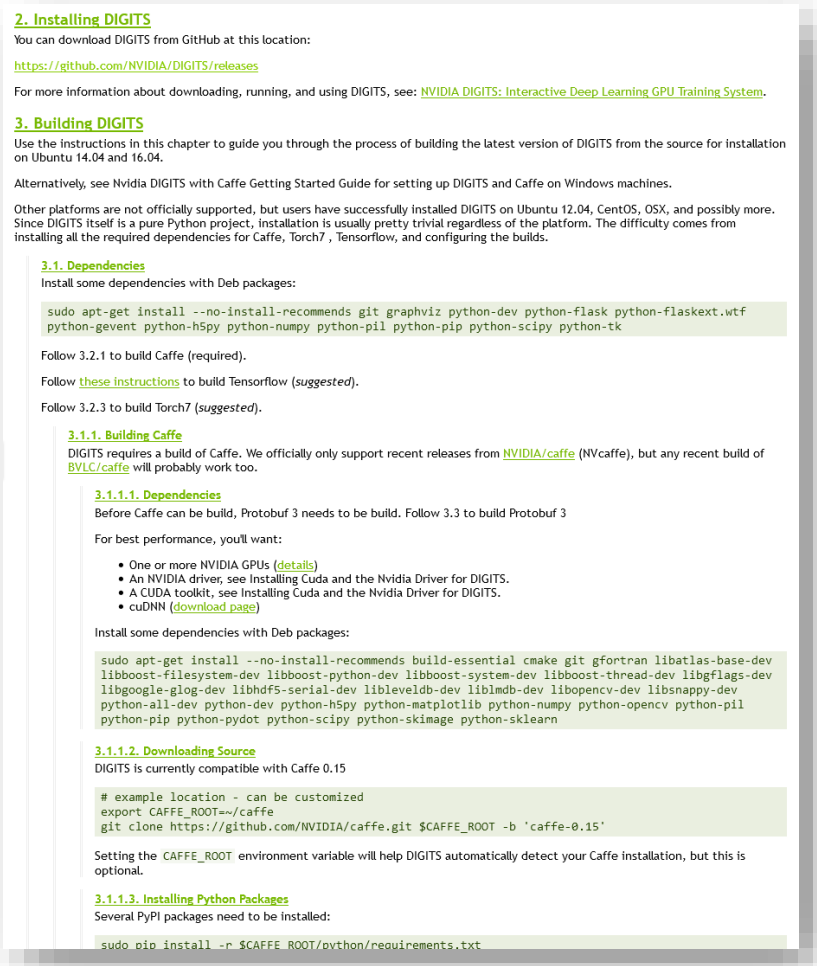
```
02:34 PM $ singularity exec ubuntu.simg ls /
archive  dev          home   lib64  opt      root  singularity  tmp  work
bin      environment  home2  media  proc     run   srv          usr
boot     etc          lib    mnt    project  sbin  sys          var
```

It's Linux, but with only the programs needed installed

Has no kernel….

Will use drivers and devices (GPU, network etc….) from the host
Can (optionally) see files from the host

Why Use a Container? – Try things out with Ease!



2. Installing DIGITS

You can download DIGITS from GitHub at this location:

https://github.com/NVIDIA/DIGITS/releases

For more information about downloading, running, and using DIGITS, see: NVIDIA DIGITS: Interactive Deep Learning GPU Training System.

3. Building DIGITS

Use the instructions in this chapter to guide you through the process of building the latest version of DIGITS from the source for installation on Ubuntu 14.04 and 16.04.

Alternatively, see Nvidia DIGITS with Caffe Getting Started Guide for setting up DIGITS and Caffe on Windows machines.

Other platforms are not officially supported, but users have successfully installed DIGITS on Ubuntu 12.04, CentOS, OSX, and possibly more. Since DIGITS itself is a pure Python project, installation is usually pretty trivial regardless of the platform. The difficulty comes from installing all the required dependencies for Caffe, Torch7 , Tensorflow, and configuring the builds.

3.1. Dependencies

Install some dependencies with Deb packages:

```
sudo apt-get install --no-install-recommends git graphviz python-dev python-flask python-flaskext.wtf
python-gevent python-h5py python-numpy python-pil python-pip python-scipy python-tk
```

Follow 3.2.1 to build Caffe (required).

Follow these instructions to build Tensorflow (*suggested*).

Follow 3.2.3 to build Torch7 (*suggested*).

3.1.1. Building Caffe

DIGITS requires a build of Caffe. We officially only support recent releases from NVIDIA/caffe (NVcaffe), but any recent build of BVLC/caffe will probably work too.

3.1.1.1. Dependencies

Before Caffe can be build, Protobuf 3 needs to be build. Follow 3.3 to build Protobuf 3

For best performance, you'll want:

- One or more NVIDIA GPUs (details)
- An NVIDIA driver, see Installing Cuda and the Nvidia Driver for DIGITS.
- A CUDA toolkit, see Installing Cuda and the Nvidia Driver for DIGITS.
- cuDNN (download page)

Install some dependencies with Deb packages:

```
sudo apt-get install --no-install-recommends build-essential cmake git gfortran libatlas-base-dev
libboost-filesystem-dev libboost-python-dev libboost-system-dev libboost-thread-dev libgflags-dev
libgoogle-glog-dev libhdf5-serial-dev libleveldb-dev liblmdb-dev libopencv-dev libsnappy-dev
python-all-dev python-dev python-h5py python-matplotlib python-numpy python-opencv python-pil
python-pip python-pydot python-scipy python-skimage python-sklearn
```

3.1.1.2. Downloading Source

DIGITS is currently compatible with Caffe 0.15

```
# example location - can be customized
export CAFFE_ROOT=~/caffe
git clone https://github.com/NVIDIA/caffe.git $CAFFE_ROOT -b 'caffe-0.15'
```

Setting the `CAFFE_ROOT` environment variable will help DIGITS automatically detect your Caffe installation, but this is optional.

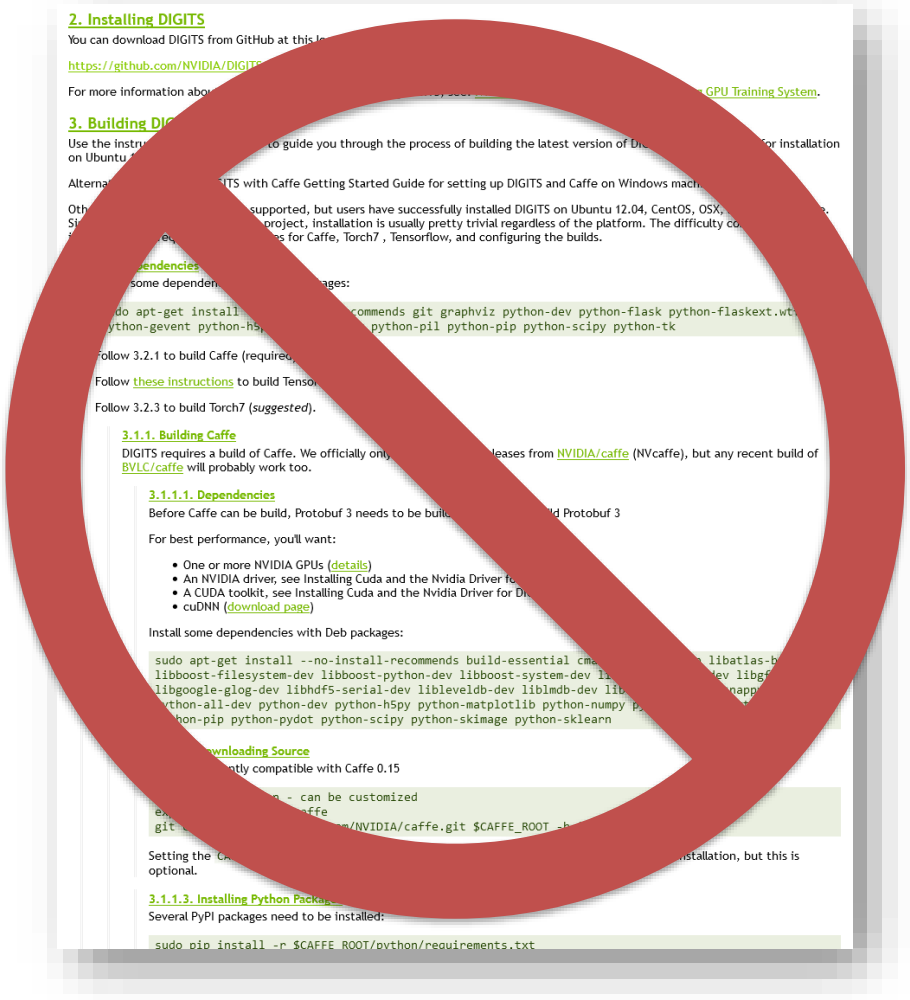
3.1.1.3. Installing Python Packages

Several PyPI packages need to be installed:

```
sudo pip install -r $CAFFE_ROOT/python/requirements.txt
```

Installation instructions for NVIDIA DIGITS

Why Use a Container? – Try things out with Ease!




```
singularity run --nv
    docker://nvidia/digits
```

Why Use a Container? - Messy Dependencies

SuperApp

I need…

Python 3.6

numpy <1.14

R 3.3

hdf5.8

I need…

Python 3.6

numpy >= 1.15

R 3.5

hdf5.10

Why Use a Container? - Long term reproducibility

Time marches on, and software moves quickly

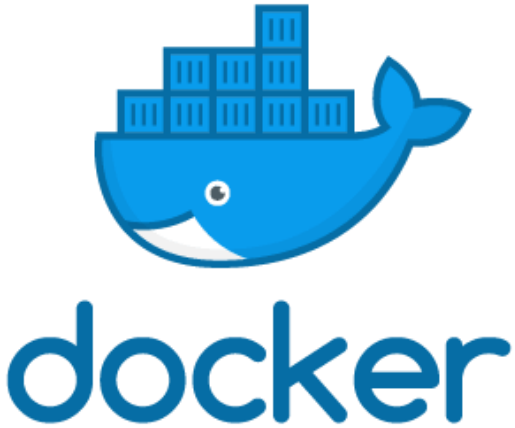
In 5 years maybe a study needs to be reproduced

Can the software stack be reconstructed on a current system?

Containers are a ‘snapshot’ that can capture the software stack used for a study, allowing it to be run in future

R 3.5.3 (March, 2019)
R 3.5.2 (December, 2018)
R 3.5.1 (July, 2018)
R 3.5.0 (April, 2018)
R 3.4.4 (March, 2018)
R 3.4.3 (November, 2017)
R 3.4.2 (September, 2017)
R 3.4.1 (June, 2017)
R 3.4.0 (April, 2017)
R 3.3.3 (March, 2017)
R 3.3.2 (October, 2016)
R 3.3.1 (June, 2016)
R 3.3.0 (April, 2016)
R 3.2.5 (April, 2016)
R 3.2.4 (March, 2016)
R 3.2.3 (December, 2015)
R 3.2.2 (August, 2015)
R 3.2.1 (June, 2015)
R 3.2.0 (April, 2015)
R 3.1.3 (March, 2015)
R 3.1.2 (October, 2014)
R 3.1.1 (July, 2014)
R 3.1.0 (April, 2014)
R 3.0.3 (March, 2014)
R 3.0.2 (September, 2013)
R 3.0.1 (May, 2013)
R 3.0.0 (April, 2013)
R 2.15.3 (March, 2013)
R 2.15.2 (October, 2012)
R 2.15.1 (June, 2012)
R 2.15.0 (March, 2012)
R 2.14.2 (February, 2012)
R 2.14.1 (December, 2011)
R 2.14.0 (November, 2011)
R 2.13.2 (September, 2011)
R 2.13.1 (July, 2011)
R 2.13.0 (April, 2011)
R 2.12.2 (February, 2011)
R 2.12.1 (December, 2010)
R 2.12.0 (October, 2010)

What's Docker?



Most well known container 'stack'

Didn't invent the concept of a container

Did make containers easy to create, run, and distribute

Docker Hub

De Facto place to share containers

Thousands and thousands of apps ready to go
(of variable quality!)

Why Can't we Use Docker?

- Docker uses an always-on service to manage containers
  Doesn't fit with HPC batch job way of working

- Docker runs as root user by default
  Can't let people do this on a shared system
  Needs newer Linux features to lock this down more

- Not easy to use HPC resources (GPUs, Infiniband Network, MPI apps)

- Way containers are stored is slow on HPC parallel filesystems

What's Singularity?

- Container runtime developed for HPC

- Runs containers as a normal user, without a system service

- Open-Source (free as in $)
  Originated at Berkeley Lab
  Now led by the Sylabs Inc. startup

- Widely adopted by small and large HPC centers
  Good chance you can take containers elsewhere without issue

Singularity on BioHPC

Default version…

```
module add singularity/2.6.1
```

Will eventually disappear due to end of security patches
Most public Singularity images were built with 2.x

Newest version…

```
module add singularity/3.5.3
```

Use whenever possible
- *Can run images built with 2.6.1*
- *Builds images that cannot be run with 2.6.1*

Running Docker Containers with Singularity

Name of container on Docker Hub

```
$ singularity run docker://godlovedc/lolcow
 _____________________________
< Be different: conform. >
 -----------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

1. Fetches container layers from Docker Hub
2. Creates a (temporary) Singularity container image
3. Runs the default program in the container

Running Docker Containers with Singularity

Name of image file to create

```
# Download the container into a permanent image file
$ singularity pull my_blast.sif docker://biocontainers/blast:2.2.31
…

# Now run it from the image file
$ singularity exec my_blast.sif blastp -version
blastp: 2.2.31+
Package: blast 2.2.31, build Apr 23 2016 15:49:47
```

This is a blast container, version 2.2.31 from the biocontainers project

https://biocontainers.pro – has 7,400 packages available, ready to use

What's a Singularity .simg / .sif file?

A single, compressed file holding:

- The programs and libraries installed in the container
- Instructions of what to run, and how to run it when `singularity run` is used

A .simg/.img file comes from Singularity 2.x

Can run on 3.x

A .sif file comes from Singularity 3.x

Cannot be run on 2.x

Supports additional features, such as signing and verifying containers

```
$ singularity inspect my_blast.sif

{
     "org.label-schema.build-date": "Friday_15_March_2019_15:37:5_CDT",
     "org.label-schema.schema-version": "1.0",
     "org.label-schema.usage.singularity.deffile.bootstrap": "docker",
     "org.label-schema.usage.singularity.deffile.from": "biocontainers/blast:2.2.31",
     "org.label-schema.usage.singularity.version": "3.0.2-3.g7d4659e"
}
```

***inspect** gives some basic details*

Different Ways to Run a Container

```
# run - start the default script/program in the container
$ singularity run lolcow_latest.sif
 ________________________________________
/ You will always get the greatest       \
\ recognition for the job you least like. /
 ----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

```
# exec - start a specified program in the container
$ singularity exec lolcow_latest.sif fortune
You'll feel much better once you've given up hope.
```

```
# We can use the container like other programs… e.g pipe something into it
$ echo "Hello BioHPC Users" | singularity exec lolcow_latest.sif cowsay
 ____________________
< Hello BioHPC Users >
 --------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Different Ways to Run a Container - Shell

`singularity shell` will give you a shell inside the container, where you can work

Use `exit` or Ctrl-D to leave the container

```
$ singularity shell lolcow_latest.sif

Singularity lolcow_latest.sif:~> cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.3 LTS"

Singularity lolcow_latest.sif:~> fortune
You will inherit some money or a small piece of land.

Singularity lolcow_latest.sif:~> exit
exit
```

Using GPU(s) with a Container

A docker or singularity container built to use an NVIDIA GPU should be run with the `--nv` option

```
$ singularity pull docker://tensorflow/tensorflow:latest-gpu-jupyter

$ singularity run --nv –B /run tensorflow_latest-gpu-jupyter.sif
[I 21:53:26.613 NotebookApp] Serving notebooks from local directory: /home2/dtrudgian
[I 21:53:26.613 NotebookApp] 0 active kernels
[I 21:53:26.613 NotebookApp] The Jupyter Notebook is running at:
[I 21:53:26.613 NotebookApp] http://localhost:8888/?token=0cf2e15efb5e388ef310f25eb6afaf7588ebf487f9092c05
[I 21:53:26.613 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip
confirmation).
[W 21:53:26.615 NotebookApp] No web browser found: could not locate runnable browser.
[C 21:53:26.615 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=0cf2e15efb5e388ef310f25eb6afaf7588ebf487f9092c05
```

-***B /run*** is needed for some images (using e.g. jupyter, that give an error otherwise)

Docker vs Singularity

Most docker containers run successfully under Singularity

Some need to be run with special options

Few *cannot be used*

Docker is optimized for running services in isolated containers

Think big web companies running at huge scale

Singularity is optimized for running applications in shared HPC environments

Default isolation of Singularity is 'leaky'

Let some things through, to make common HPC tasks easier

Docker vs Singularity 1 – Mounted filesystems

Docker containers usually mount nothing by default

Singularity on BioHPC mounts:

- /project
- /work
- /archive

Nice easy access to your data

- Home directory
  (/home2/s178722)

Can get complicated!

Docker vs Singularity 1 – Mounted filesystems

Things in your home directory can leak into the container… causing strange behaviour…

```
$ docker run python:2.7
$ pip list
...
pip (9.0.3)
setuptools (39.0.1)
virtualenv (15.2.0)
wheel (0.30.0)
```

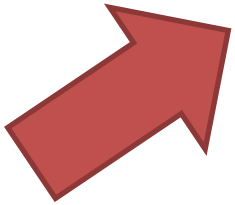
```
$ singularity exec docker://python:2.7
pip list
...
clair-singularity (0.1.0)
coverage (4.5.1)
funcsigs (1.0.2)
more-itertools (4.1.0)
pathspec (0.5.5) pip (9.0.3)
pluggy (0.6.0)
py (1.5.3)
pytest (3.5.0)
pytest-cov (2.5.1)
PyYAML (3.12)
requests-toolbelt (0.8.0)
setuptools (39.0.1)
virtualenv (15.2.0)
wheel (0.30.0)
yamllint (1.11.0)
```

Docker vs Singularity 1 – Mounted filesystems

Things in your home directory can leak into the container… causing strange behaviour…

```
$ docker run python:2.7
$ pip list
...
pip (9.0.3)
setuptools (39.0.1)
virtualenv (15.2.0)
wheel (0.30.0)
```

```
$ singularity exec docker://python:2.7
pip list
...
clair-singularity (0.1.0)
coverage (4.5.1)
funcsigs (1.0.2)
more-itertools (4.1.0)
pathspec (0.5.5) pip (9.0.3)
pluggy (0.6.0)
py (1.5.3)
pytest (3.5.0)
pytest-cov (2.5.1)
PyYAML (3.12)
requests-toolbelt (0.8.0)
setuptools (39.0.1)
virtualenv (15.2.0)
wheel (0.30.0)
yamllint (1.11.0)
```



$HOME/.local/lib/python2.7

Docker vs Singularity 1 – Mounted filesystems

We can increase the containment of Singularity with `--contain`

```
$ singularity exec --contain docker://python:2.7
pip list
...
pip (9.0.3)
setuptools (39.0.1)
virtualenv (15.2.0)
wheel (0.30.0)
```

Need to add back other cluster filesystems with -B option…

```
singularity exec --contain -B /project -B /archive …
```

Docker vs Singularity 2 - USER

Some containers use a USER instruction in their Dockerfile

Install software as a specific USER, to be run as that USER.

With Singularity on BioHPC you are **always** your own username in the container, so you cannot access things in the container restricted to another USER

*No simple solution**

** Email biohpc-help@utsouthwestern.edu for assistance*

Building a Container 1 – Start Vagrant Machine

Vagrant is a tool to manage Virtual Machines for development
Singularity needs root to build images – can use root on BioHPC inside a VM
Available on workstations / thin-clients, or at www.vagrantup.com

```
$ mkdir singularity_build
$ cd singularity_build
$ export VAGRANT_HOME=/work/biohpcadmin/s178722/.vagrant (define your PATH against home space issue, create first)
$ vagrant init sylabs/singularity-3.5-ubuntu-bionic64   OR
$ vagrant init singularity-3.5-ubuntu-bionic64 https://vagrantcloud.com/sylabs/boxes/singularity-3.5-ubuntu-bionic64/versions/20191206.0.0/providers/virtualbox.box
…
$ vagrant up
…
$ vagrant ssh

# UTSW Proxies
$ sudo sh -c "echo 'export http_proxy=http://proxy.swmed.edu:3128' > /etc/profile.d/proxy.sh"
$ sudo sh -c "echo 'export https_proxy=http://proxy.swmed.edu:3128' >> /etc/profile.d/proxy.sh"

$ exit
```

Building a Container 2 – A Singularity Recipe File

```
BootStrap: docker
From: ubuntu:16.04
```

Bootstrap/From – what we start from

```
%post
apt-get -y update
# install some basic tools
apt-get -y install curl tar gzip

apt-get clean
apt-get autoremove

# now, download and install julia
curl -sSL "https://julialang-s3.julialang.org/bin/linux/x64/1.0/julia-1.0.1-linux-x86_64.tar.gz" > julia.tgz
tar -C / -zxf julia.tgz
rm -f julia.tgz
```

%post – run inside the container on build

```
%runscript
julia "$@"
```

%runscript – what `singularity run` does

%environment – set inside the container

```
%environment
export PATH=/julia-1.0.1/bin:$PATH
export LD_LIBRARY_PATH=/julia-1.0.1/lib:/julia-1.0.1/lib/julia:$LD_LIBRARY_PATH
export LC_ALL=C
```


Adapted from: https://github.com/sylabs/examples/tree/master/lang/julia

Building a Container 3 – Singularity Build

```
# Enter the VM if not there already
$ vagrant ssh

# Become root in the VM
$ sudo su -

# Build the container
$ cd /vagrant
$ singularity build julia.sif julia.def

# Leave the VM and stop it (exit twice)
$ exit
$ exit

$ vagrant halt
```

Building a Container 4 – Running on BioHPC

The container .sif file should be in our `singularity_build` dir on BioHPC

```
$ module add singularity/3.5.3

$ cd ~/singularity_build

$ singularity run julia.sif hello-world.jl
Hello world!
For full tutorial, visit: https://github.com/sylabs/examples/lang/julia
```

```
10:01 AM $ singularity run julia.sif
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.0.1 (2018-09-29)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia>
```

Sources of Singularity Containers

(Other than using docker:// containers)

Sylabs Library

https://cloud.sylabs.io

```
singularity pull library://…
```

Sylabs equivalent of Docker Hub
Singularity 3.x containers

Singularity Hub

https://singularity-hub.org

```
singularity pull shub://…
```

Stanford based project
Singularity 2.x containers

Galaxy Project Depot

https://depot.galaxyproject.org/singularity/

Further Reading

- BioHPC Guide (long-form version of this session)
  https://portal.biohpc.swmed.edu/content/guides/singularity-containers-biohpc/

- Singularity Documentation
  https://www.sylabs.io/guides/3.0/user-guide/

- Example Singularity container recipes
  https://github.com/sylabs/examples