TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

Containers in HPC

Petascale Computing Institute 2019

PRESENTED BY:

Joe Allen
Biomedical Informatics Research Associate
Texas Advanced Computing Center
The University of Texas at Austin
wallen@tacc.utexas.edu

Objectives for this session

- What is a container?
- Why are containers useful?

- How to find and use existing containers
- How to develop and use your own containers

- How to use containers on HPC [Stampede2, Blue Waters, Cori]

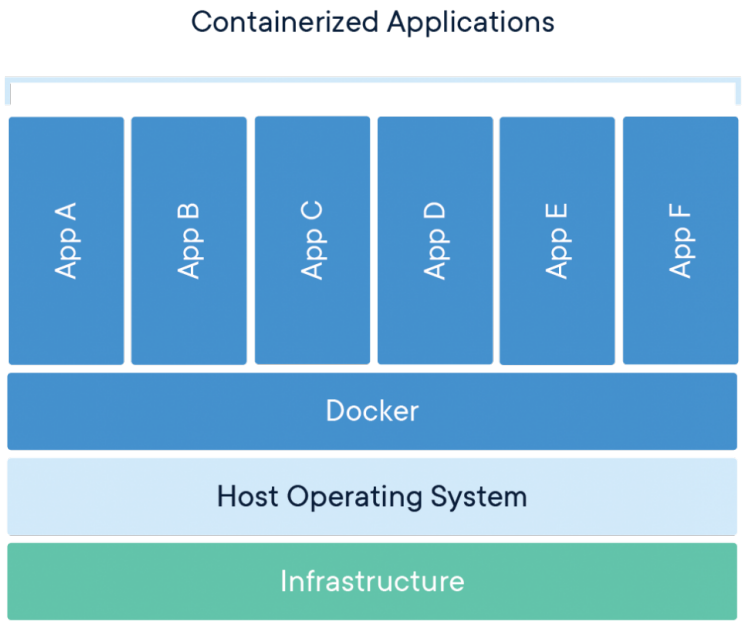
Follow along: https://wjallen.github.io/petascale/

What is a container?

- A standard unit of software that packages up **code** and all its **dependencies** so the application runs quickly and reliably from one computing environment to another

- Isolate application from environment to ensure **reproducibility** and **portability**

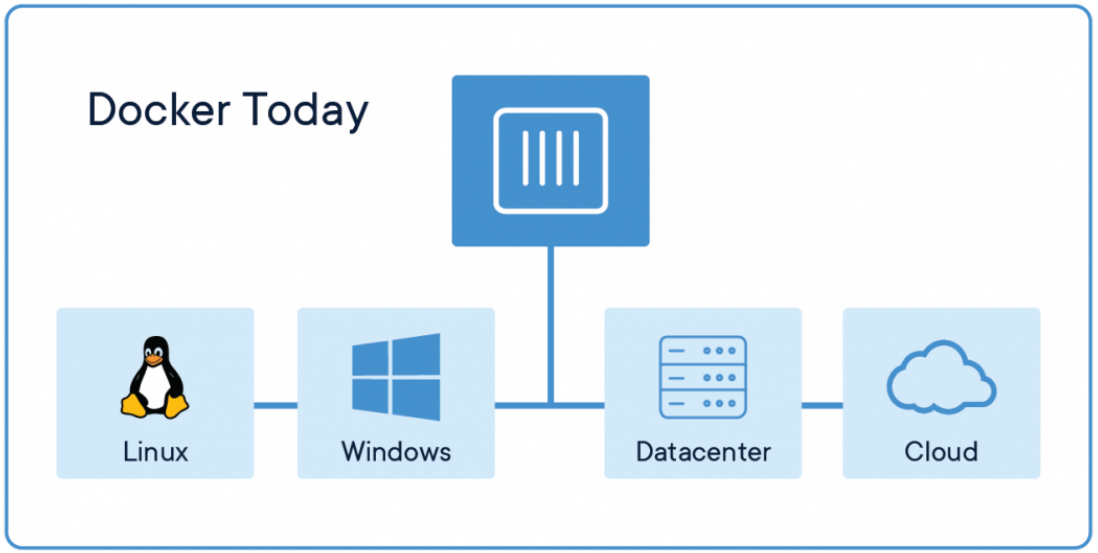
- Share the host OS system kernel, so relatively **lightweight** and **low overhead**



https://www.docker.com/

Container technologies

- **Docker (2013)** is the gold standard
- Docker containers can be deployed anywhere
- BUT, Docker grants superuser privileges and some containers may allow users root access to host files ☹
- Docker-compatible technologies **Singularity** (Stampede2) and **Shifter** (Blue Waters, Cori) were designed for HPC environments ☺



Why are containers useful?

- Develop and deploy future-proof applications by creating packages that are self-contained
- Distribute production ready code that can run anywhere without installation, configuration, worrying about dependencies, etc.
- Mitigate portability issues related to applications
- **Enables reproducible science**

SWEEPING DECLARATION

Everyone who is ...

analyzing data with scientific software …

or performing some sort of numerical simulation …

on a local resource, HPC cluster, or cloud …

should learn to develop and/or use containers

Objectives for this session

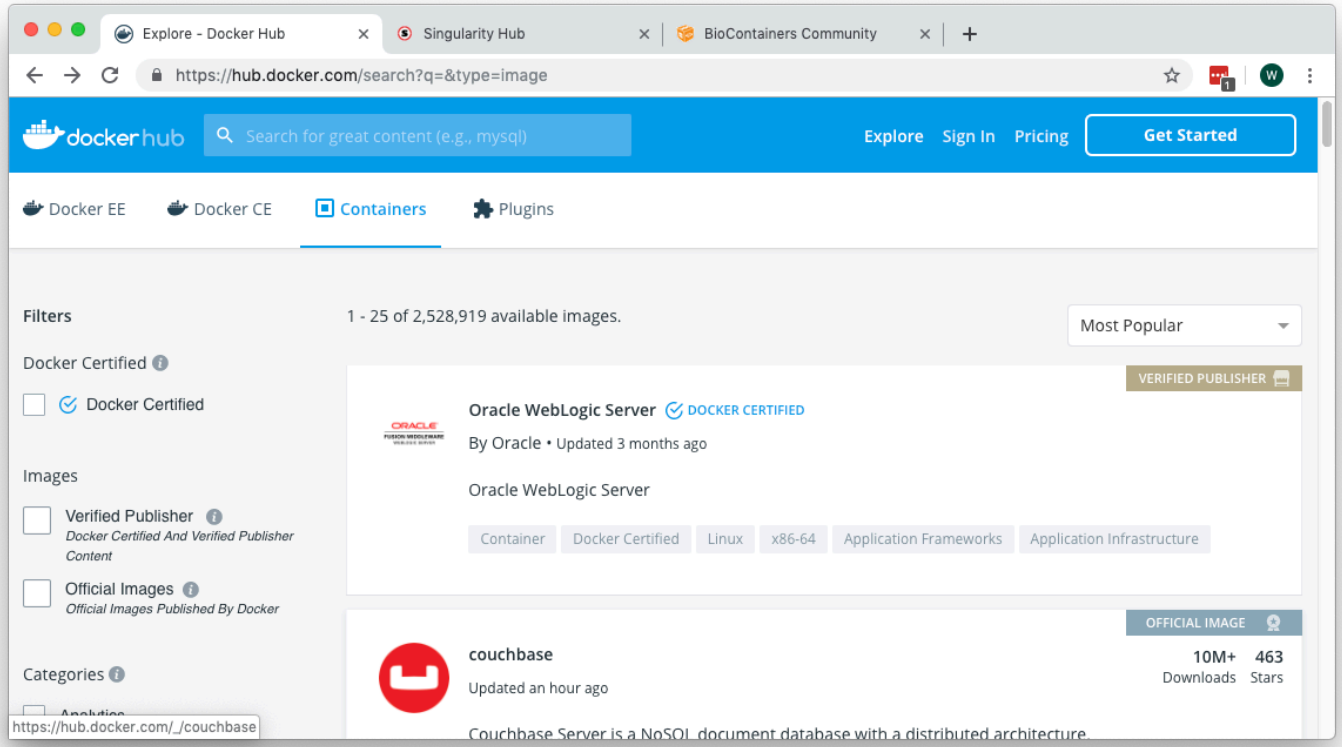
- What is a container?
- Why are containers useful?

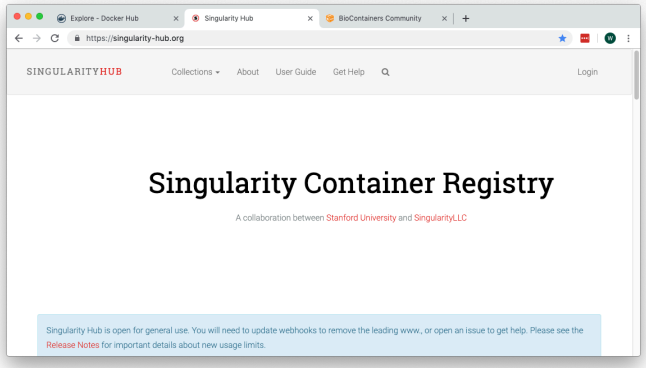
- How to find and use existing containers
- How to develop and use your own containers

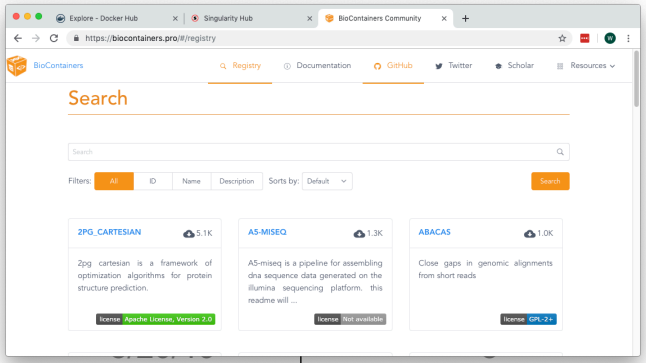
- How to use containers on HPC [Stampede2, Blue Waters, Cori]

How to find existing containers

- Docker Hub
  - https://hub.docker.com/
  - 2.5M+ container images
  - **You should make an account!**
- Singularity Hub
  - https://singularity-hub.org/
  - Still maturing
  - Requires GitHub account link
- BioContainers
  - https://biocontainers.pro/
  - Domain focused
  - ~8K unique containers
  - (Many more if you include tags)

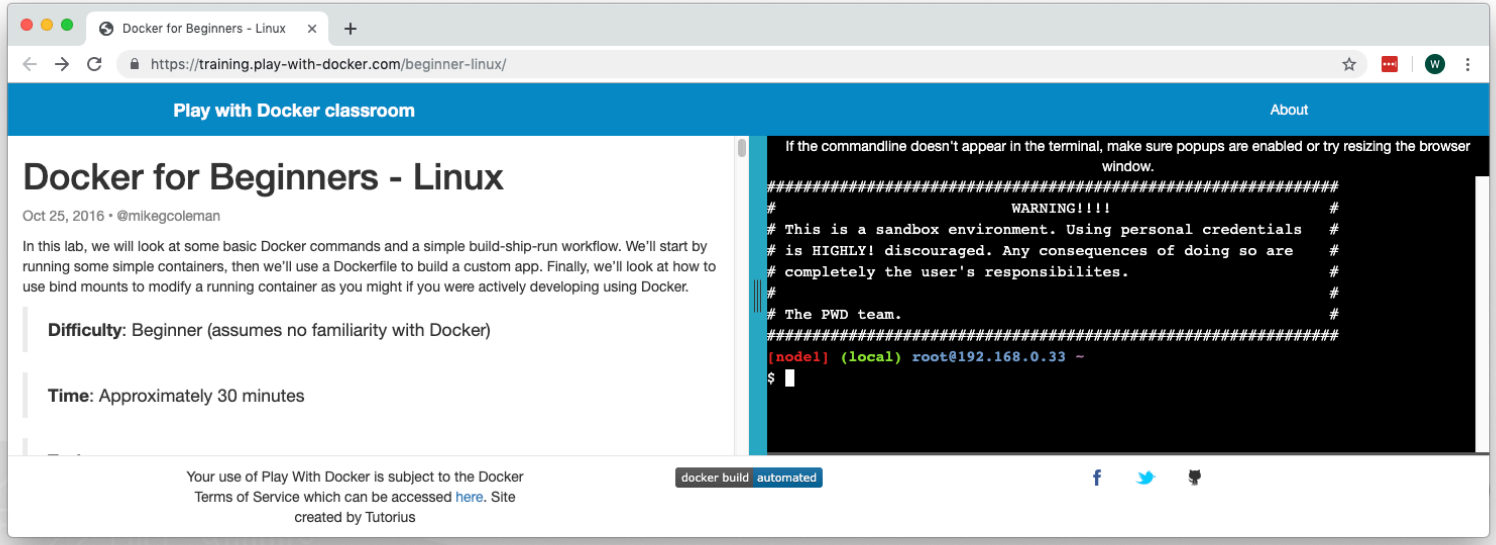






Getting started with Docker

- Install local client
  - https://docs.docker.com/docker-for-windows/install/
  - https://docs.docker.com/docker-for-mac/install/
  - Linux users can **apt-get** or **yum install** as appropriate (Google is your friend)
- Don't want to download?
  - https://training.play-with-docker.com/beginner-linux/
  - (Docker Hub login required)



Getting started with Docker

- Open up your favorite Terminal (Mac, Linux), or the Docker Terminal (Windows) which comes with the distribution
- Try out some basic commands:

```
$ docker version      # show version information

$ docker images       # show images you have pulled

$ docker ps           # show running containers

$ docker run hello-world

      # ...actually don't do this
```

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:6540fc08ee6e6b7b63468dc3317e3303aae178cb8a45ed3123180328bcc1d20f
Status: Downloaded newer image for hello-world:latest


Hello from Docker!
This message shows that your installation appears to be working correctly.


To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.


To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash


Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/


For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

```
# this is a little better...

$ docker pull hello-world:latest
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest:
sha256:6540fc08ee6e6b7b63468dc3317e3303aae178cb8a45ed3123180328bcc1d20f
Status: Downloaded newer image for hello-world:latest

$ docker images
REPOSITORY          TAG          IMAGE ID         CREATED          SIZE
hello-world         latest       fce289e99eb9     7 months ago     1.84kB

$ docker run hello-world:latest
Hello from Docker!
This message shows that your installation appears to be working correctly.
...

$ docker inspect hello-world        # more information about container image
```

```
# real world example

$ docker pull biocontainers/fastqc:v0.11.5_cv4
v0.11.5_cv4: Pulling from biocontainers/fastqc
34667c7e4631: Pull complete
...
c3b3dcd1b3a5: Pull complete
Digest:
sha256:387748462c7fc280b7959ceda0f6251190d2e4b9ebc0585d24e7bcb58bdcf2bf
Status: Downloaded newer image for biocontainers/fastqc:v0.11.5_cv4

$ docker run --rm biocontainers/fastqc:v0.11.5_cv4 fastqc --help

            FastQC - A high throughput sequence QC analysis tool

SYNOPSIS

        fastqc seqfile1 seqfile2 .. seqfileN
 fastqc [-o output dir] [--(no)extract] [-f fastq|bam|sam]
           [-c contaminant file] seqfile1 .. seqfileN
...
```

Unpacking the ‘`docker run`’ command

```
docker run --rm biocontainers/fastqc:v0.11.5_cv4 fastqc --help
```

Run something

Remove the container when the process completes

The name of the container and version tag

The command to run

```
# interactive example

$ docker run --rm -it biocontainers/fastqc:v0.11.5_cv4 /bin/bash

biodocker@f195d8ee9d32:/data$ pwd
/data

biodocker@f195d8ee9d32:/data$ whoami
biodocker

biodocker@f195d8ee9d32:/data$ which fastqc
/usr/local/bin/fastqc

biodocker@f195d8ee9d32:/data$ fastqc --help

            FastQC - A high throughput sequence QC analysis tool

SYNOPSIS

      fastqc seqfile1 seqfile2 .. seqfileN

    fastqc [-o output dir] [--(no)extract] [-f fastq|bam|sam]
```

Unpacking the interactive '`docker run`' command

```
docker run --rm -it biocontainers/fastqc:v0.11.5_cv4 /bin/bash
```

Run something

Remove the container when the process completes, and connect your terminal to the container runtime

The name of the container and version tag

The type of shell to start

Quick recap

- Find a container on Docker Hub and pull it to your local environment

```
docker pull <container:tag>
```

- Run a command (e.g. a scientific application) inside a container
  - Useful for performing analysis or simulation

```
docker run --rm <container:tag> <command>
```

- Start up an interactive shell inside a container
  - Useful for debugging, testing executables in an existing container
  - Useful for developing a new container from scratch

```
docker run --rm -it <container:tag> <shell>
```

How to develop your own containers

- There are a couple ways to develop your own containers, but there is one way that is **reproducible** and **well documented** => the **`Dockerfile`**

- General steps might include:

1. Choose a base operating system
2. Install dependencies, other useful packages
3. Install scientific application
4. Set any environment variables that might be helpful

```
$ pwd
/Users/username/fastqc-dev-folder

$ ls
Dockerfile

$ cat Dockerfile
FROM ubuntu:16.04

RUN apt-get update && apt-get upgrade -y \
    && apt-get install -y default-jre perl wget zip

RUN wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.7.zip \
    && unzip fastqc_v0.11.7.zip \
    && rm fastqc_v0.11.7.zip \
    && chmod +x /FastQC/fastqc

ENV PATH "/FastQC:$PATH"
```

Choose a base operating system

Update and install necessary packages

Install the application

Use environment variable to add executable to PATH

```
$ docker build -t username/fastqc:0.11.7 ./
Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM ubuntu:16.04
 ---> 5e13f8dd4c1a
...
Successfully built 2005acfb2869
Successfully tagged username/fastqc:0.11.7

$ docker images
REPOSITORY          TAG         IMAGE ID          CREATED              SIZE
username/fastqc     0.11.7      2005acfb2869      16 minutes ago       460MB
hello-world         latest      fce289e99eb9      7 months ago         1.84kB

$ docker run --rm username/fastqc:0.11.7 which fastqc
/FastQC/fastqc

$ docker push username/fastqc:0.11.7
The push refers to repository [docker.io/username/fastqc]
e79142719515: Mounted from library/ubuntu
aeda103e78c9: Mounted from library/ubuntu
2558e637fbff: Mounted from library/ubuntu
f749b9b0fb21: Mounted from library/ubuntu
0.11.7: digest: sha256:9e42ab85eedec90228d7fa8ba94b4d6dfe33b2173584e88b190d size: 1575
```

Quick recap #2

- Build a container image from a `Dockerfile`

```
docker build –t <container:tag> ./
```

- Store your credentials for Docker Hub locally

```
docker login
```

- Push the container image to Docker Hub

```
docker push <container:tag>
```

Getting more help with Docker

- The command line tools are very well documented:

```
$ docker --help              # show all docker options and summaries

$ docker COMMAND --help      # show options and summaries for a particular
                             # command
```

- Find support online:
  - https://docs.docker.com/get-started/

Miscellaneous Docker tips

1. Save your `Dockerfiles` – GitHub is a good place for this
2. You probably don't need `ENTRYPOINT` or `CMD`
3. Usually better to use `COPY` instead of `ADD`
4. Order of operations in the `Dockerfile` is important; combine steps where possible
5. Avoid `latest` tag; use explicit tag callouts
6. The command `docker system prune` is your friend
7. Use `docker-compose` for multi-container pipelines and microservices
8. Considerations for one tool per container vs. multiple tools per container

Objectives for this session

- What is a container?
- Why are containers useful?

- How to find and use existing containers
- How to develop and use your own containers

- How to use containers on HPC [Stampede2, Blue Waters, Cori]

How to use containers on HPC

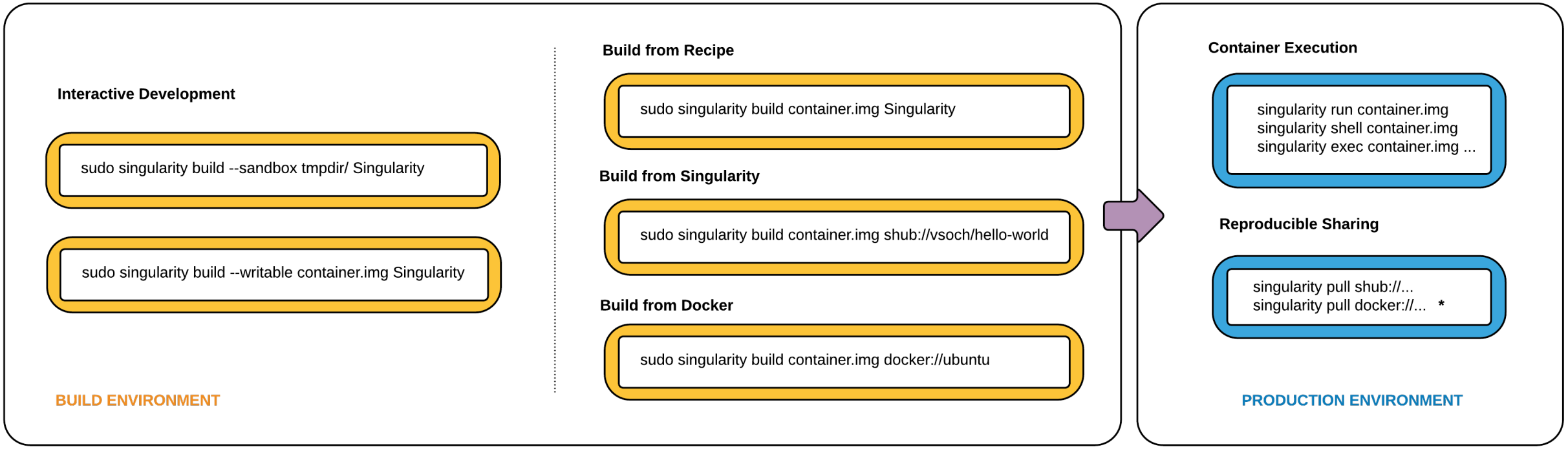
- As mentioned, security concerns preclude users from running Docker containers at most HPC centers
- At **TACC**, users can run **Singularity**
- At **NERSC** and **NCSA**, users can run **Shifter**

Container technologies: Singularity

- **Singularity (2016)**
- Developed at LBL
- Designed for HPC
- Can pull Docker containers
- Outside user = inside user
- Can auto-mount shared filesystems
- MPI aware => scalable
- GPU aware => CUDA runtimes must match

https://singularity.lbl.gov/ | https://sylabs.io/

Container technologies: Singularity



* Docker construction from layers not guaranteed to replicate between pulls

- Full disclosure: I don't prefer Singularity native build methods
- I prefer **docker build / push** followed by **singularity pull**

https://singularity.lbl.gov/ | https://sylabs.io/

Container technologies: Shifter

- **Shifter (2015)**
- Developed at NERSC
- Designed for HPC
- Can pull Docker containers (and other formats)
- Root (user) is squashed
- Can mount shared filesystems
- MPI aware => scalable
- Images usually need to be pre-cached on cluster

https://github.com/NERSC/shifter
https://docs.nersc.gov/programming/shifter/how-to-use/
https://bluewaters.ncsa.illinois.edu/shifter

Container technologies: Shifter

General Shifter Workflow

The general workflow for using Docker images with Shifter on Blue Waters, is as follows:

1.	Create	(build) a Docker image on your computer
2.	Upload	(push) the image to a registry (Docker Hub)
3.	Download	(pull) the image from the registry to an HPC system (Blue Waters)
4.	Launch	use the container in a job

- **Essentially identical to the recommended Singularity workflow**

https://github.com/NERSC/shifter
https://docs.nersc.gov/programming/shifter/how-to-use/
https://bluewaters.ncsa.illinois.edu/shifter

Hands on: Run an HPC job

Scenario: You are a researcher with raw data (`SP1.fq`) that you need to analyze. The computation is expensive and you don't want to tie up your local Linux workstation. However, the HPC cluster runs on CentOS and your application (`FastQC`) only runs on Ubuntu.

1. Log in to your favorite cluster
2. Stage your input data
3. Pull the container
4. Prepare a job template
5. Submit the job
6. Check the results

Hands on: Run an HPC job

1. Log in to your favorite system

Stampede2

```
$ ssh USER@stampede2.tacc.utexas.edu
```

Blue Waters

```
$ ssh USER@bwbay.ncsa.illinois.edu
```

Cori

```
$ ssh USER@cori.nersc.gov
```

Hands on: Run an HPC job

2. Stage your input data

Stampede2 / Blue Waters / Cori

```
[login]$ wget https://wjallen.github.io/petascale/SP1.fq
[login]$ head SP1.fq
@@cluster_2:UMI_ATTCCG
TTTCCGGGGCACATAATCTTCAGCCGGGCGC
+
9C;=;=<9@4868>9:67AA<9>65<=>591
@cluster_8:UMI_CTTTGA
TATCCTTGCAATACTCTCCGAACGGGAGAGC
+
1/04.72,(003,-2-22+00-12./.-.4-
@cluster_12:UMI_GGTCAA
GCAGTTTAAGATCATTTTATTGAAGAGCAAG
```

Sample data: Jay Hesselberth, Genome Analysis Workshop, https://molb7621.github.io/workshop/index.html

3. Pull the container

Stampede2

```
[login]$ idev
...
[compute]$ module load tacc-singularity python3
[compute]$ singularity pull --name wallen-fastqc-0.11.7.simg docker://wallen/fastqc:0.11.7
Singularity container built: /work/03439/wallen/singularity_cache/wallen-fastqc-0.11.7.simg
[compute]$ ls $WORK/singularity_cache/
wallen-fastqc-0.11.7.simg*
```

Blue Waters

```
[login]$ qsub -I -l nodes=1:ppn=1 -l walltime=00:30:00
...
[compute]$ module load shifter
[compute]$ shifterimg pull docker:wallen/fastqc:0.11.7
2019-08-19T15:44:49 Pulling Image: docker:wallen/fastqc:0.11.7, status: READY
[compute]$ shifterimg images | grep fastqc
bluewaters docker     READY    6d2726df2e   2019-08-19T15:44:14 wallen/fastqc:0.11.7
```

Cori

```
[login]$ salloc -N 1 -C haswell -q interactive -t 00:30:00
...
[compute]$ shifterimg pull docker:wallen/fastqc:0.11.7
2019-08-19T14:02:58 Pulling Image: docker:wallen/fastqc:0.11.7, status: READY
[compute]$ shifterimg images | grep fastqc
cori       docker     READY    6d2726df2e   2019-08-19T14:02:57 wallen/fastqc:0.11.7
```

Hands on: Run an HPC job

4-5. Prepare a job template and submit the job

Stampede2

```
[login]$ cat singularity_job.slurm
#!/bin/bash
#SBATCH -J myjob
#SBATCH -o myjob.o%j
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -t 00:10:00
#SBATCH -p skx-dev
#SBATCH -A myalloc   # Allocation name

module load tacc-singularity

SIMG=$WORK/singularity_cache/wallen-fastqc-
0.11.7.simg

singularity exec $SIMG fastqc SP1.fq

[login]$ sbatch singularity_job.slurm
...
Submitted batch job 4197252
```

Blue Waters

```
[login]$ cat shifter_job.pbs
#!/bin/bash
#PBS -N testjob
#PBS -e $PBS_JOBID.err
#PBS -o $PBS_JOBID.out
#PBS -l nodes=1:ppn=1:xe
#PBS -l walltime=00:10:00
#PBS -A myalloc
#PBS -l gres=shifter16

module load shifter
IMG=docker:wallen/fastqc:0.11.7

aprun -b shifter --image=$IMG fastqc SP1.fq

[login]$ qsub shifter_job.pbs
INFO: Job submitted to account: myalloc
10240521.bw
```

Cori

```
[login]$ cat shifter_job.slurm
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=00:10:00
#SBATCH --qos=debug
#SBATCH --constraint=haswell
#SBATCH --image=docker:wallen/fastqc:0.11.7

srun -n 1 shifter fastqc SP1.fq

[login]$ sbatch shifter_job.slurm
Submitted batch job 24000106
```

Expected Output (all)

```
[login]$ ls
SP1.fq  SP1_fastqc.html  SP1_fastqc.zip
```

Closing thoughts

- Are reproducibility and provenance important in computational science?
  - (Trick question, of course they are)
- How do you achieve reproducibility and provenance in computational science?
- Integration between GitHub and Docker Hub
  - Pushing your code to GitHub automatically updates container images
  - Use tags and refer to tags in publications
- Containers as modules (Stampede2)
  - `module help biocontainers`
  - `module load biocontainers`

Questions?

Joe Allen
wallen@tacc.utexas.edu