

Docker to Singularity Conversion

Peter Z. Vaillancourt
Computational Scientist,
Center for Advanced Computing (CAC)
Cornell University
XCRI Engineer, XSEDE





Docker to Singularity Conversion

Different Philosophies

Docker

- Focus on flexibility and cloud usability
- Daemon runs as root
- Isolated from host filesystem
- Not originally designed for interoperability with Singularity or HPC systems

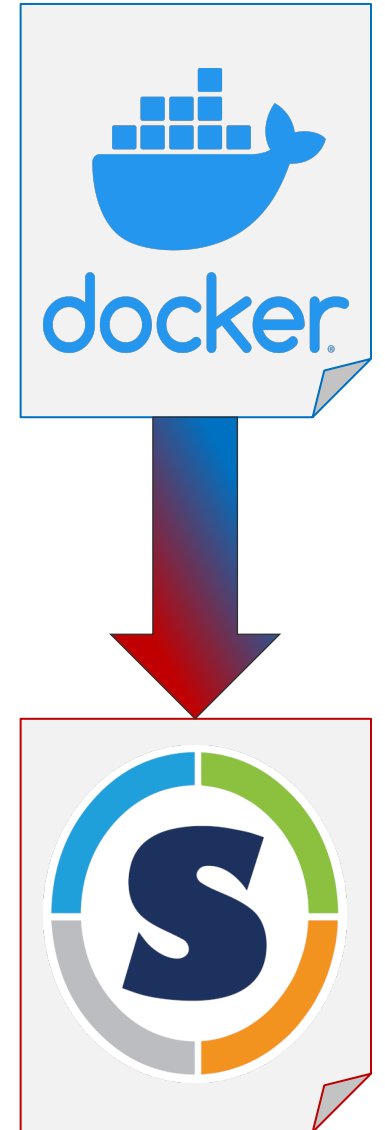
Singularity

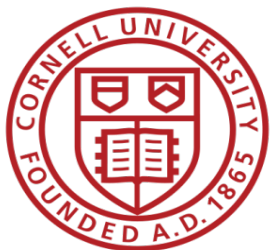
- Focus on security and HPC usability
- Runs in userspace
- Direct Filesystem access
- Designed for interoperability with Docker

Docker to Singularity Conversion

Overview of the Process


1. Create & Build the **Dockerfile**
2. Test the **Docker** container
3. Tag the **Docker** image and push to a **DTR**
4. Create the **Singularity** Definition file
5. Build & Test the **Singularity** container
6. (Optional) Push **Singularity** image to a registry





Docker to Singularity Conversion

Where to Start?

The Docker logo, a blue whale, is positioned behind the code. The code is displayed in a light blue box with a folded corner effect.

```
FROM python:3.6-buster  
SHELL ["/bin/bash", "-c"]  
USER root  
RUN apt-get update -y && \  
    apt-get install -y \  
        cmake \  
        liblapack-dev \  
        libblas-dev \  
        . . .
```

- Focus here is on the composition of the Dockerfile
 - For public images, find the Dockerfile where possible
 - Often available through Docker Hub links to GitHub repositories
 - Without the Dockerfile, you're taking a risk (security and conversion)
 - Keep these ideas in mind when building a Dockerfile
- Refer to Singularity Best Practices for Docker Images docs when attempting a conversion: https://sylabs.io/guides/3.5/user-guide/singularity_and_docker.html#best-practices

Docker to Singularity Conversion

Best Practices

1. Account for differences in the trust model of Docker vs. Singularity

- Do not create a user
- Do not use the USER command unless it's to specify "USER root"

```
FROM python:3.6-buster

SHELL ["/bin/bash", "-c"]

USER root

RUN apt-get update -y && \
    apt-get install -y \
    cmake \
    liblapack-dev \
    libblas-dev \
    . . .
```

Docker to Singularity Conversion

Best Practices

1. Account for differences in the trust model of Docker vs. Singularity

- Do not create a user
- Do not use the USER command unless it's to specify "USER root"

2. Account for potential changes in the underlying Docker image

- Use a Singularity definition file to pull and convert
- Version pinning of Docker image can mitigate this, but not alleviate it entirely
- Do a "diff" before "pull"
- Also see: <https://singularityhub.github.io/container-diff/>



```
FROM python:3.6-buster

SHELL ["/bin/bash", "-c"]

USER root

RUN apt-get update -y && \
    apt-get install -y \
    cmake \
    liblapack-dev \
    libblas-dev \
    . . .
```

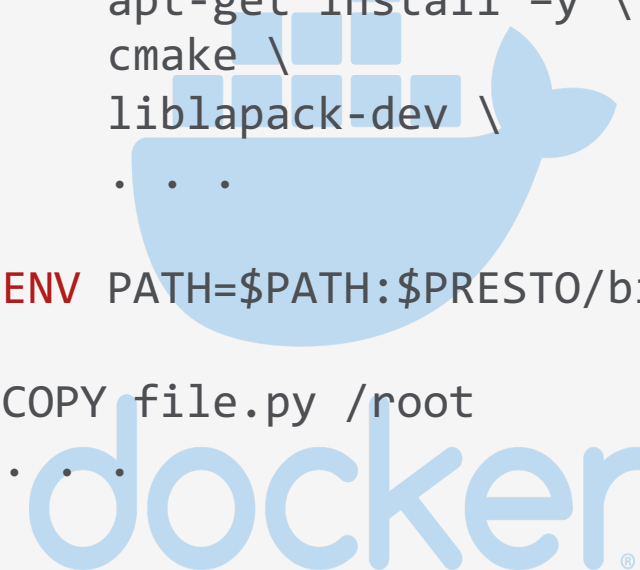
Docker to Singularity Conversion

Best Practices

3. Declare environment Variables in the Dockerfile

- Do not declare them in other files (i.e. .bashrc or .profile)
- Understand Singularity definition files:
https://sylabs.io/guides/3.5/user-guide/definition_files.html#definitionfiles

```
• • •  
RUN apt-get update -y && \  
    apt-get install -y \  
    cmake \  
    liblapack-dev \  
    • • •  
ENV PATH=$PATH:$PRESTO/bin  
  
COPY file.py /root  
• • •
```



Docker to Singularity Conversion

Best Practices

3. Declare environment Variables in the Dockerfile

- Do not declare them in other files (i.e. .bashrc or .profile)
- Understand Singularity definition files:
https://sylabs.io/guides/3.5/user-guide/definition_files.html#definitionfiles

4. Avoid installing to “/root”

- Not a blanket ban, but can sometimes cause issues
- User access remains the same as on host
- Cannot make changes to the read-only filesystem

```
...  
RUN apt-get update -y && \  
    apt-get install -y \  
    cmake \  
    liblapack-dev \  
...  
ENV PATH=$PATH:$PRESTO/bin  
COPY file.py /root  
...  
docker
```


Docker to Singularity Conversion

Best Practices

5. Prepare for “/” to be read-only

- Overlay FS can allow changes, but not allowed on some HPC systems: https://sylabs.io/guides/3.5/user-guide/persistent_overlays.html
- The default install locations of most trusted/maintained software will just work
- A good place to install may be a subdirectory of /opt or /usr/local

```
...  
COPY file.py /root  
  
ENV PATH=$PATH:$PRESTO/bin  
  
COPY file.py $HOME  
COPY file2.py $TMP  
  
...  
  
RUN ldconfig
```


docker®

Docker to Singularity Conversion

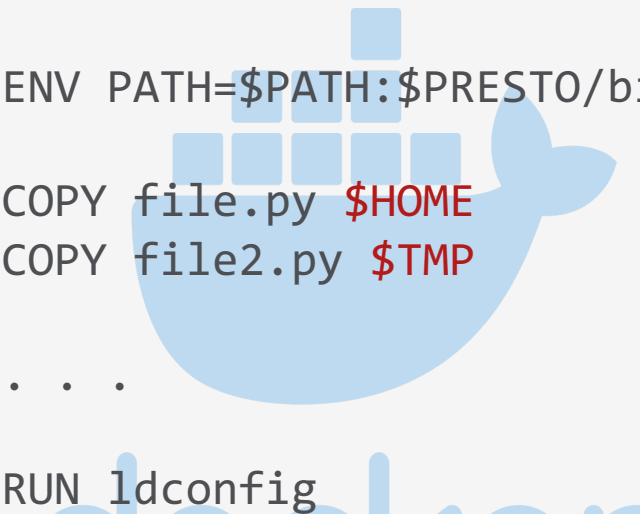
Best Practices

5. Prepare for “/” to be read-only

- Overlay FS can allow changes, but not allowed on some HPC systems: https://sylabs.io/guides/3.5/user-guide/persistent_overlays.html
- The default install locations of most trusted/maintained software will just work
- A good place to install may be a subdirectory of /opt or /usr/local

6. Avoid placing files in “\$HOME” or “\$TMP”

```
...  
COPY file.py /root  
  
ENV PATH=$PATH:$PRESTO/bin  
  
COPY file.py $HOME  
COPY file2.py $TMP  
  
...  
  
RUN ldconfig
```



Docker to Singularity Conversion

Best Practices

5. Prepare for “/” to be read-only

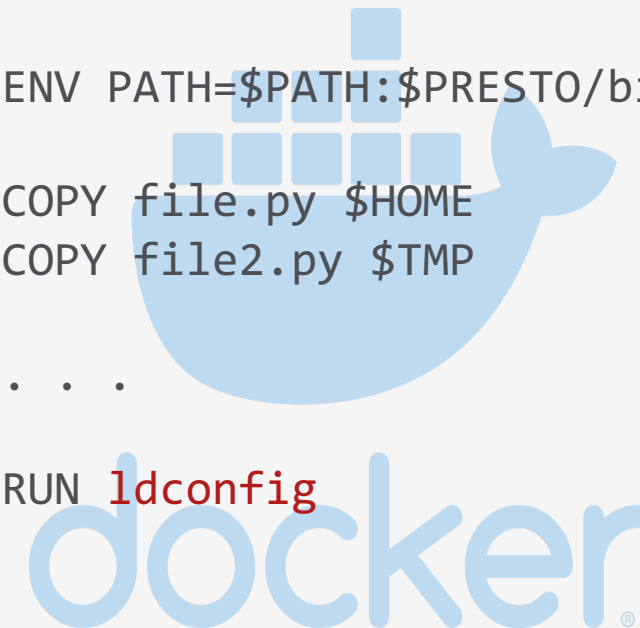
- Overlay FS can allow changes, but not allowed on some HPC systems: https://sylabs.io/guides/3.5/user-guide/persistent_overlays.html
- The default install locations of most trusted/maintained software will just work
- A good place to install may be a subdirectory of /opt or /usr/local

6. Avoid placing files in “\$HOME” or “\$TMP”

7. Ensure symbolically linked libraries are cached

- Can run “ldconfig” at or near the end of the Dockerfile

```
...  
COPY file.py /root  
  
ENV PATH=$PATH:$PRESTO/bin  
  
COPY file.py $HOME  
COPY file2.py $TMP  
  
...  
  
RUN ldconfig
```

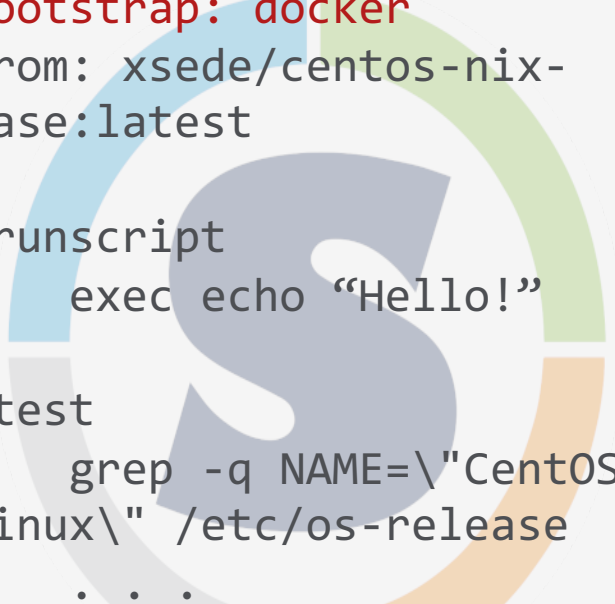


Docker to Singularity Conversion

Best Practices

8. Do not use plain text passwords

- Can use the “--docker-login” option for Singularity “pull” and “build” commands



```
Bootstrap: docker
From: xsede/centos-nix-
base:latest

%runscript
    exec echo "Hello!"

%test
    grep -q NAME=\"CentOS\
Linux\" /etc/os-release
    . . .
```

Docker to Singularity Conversion

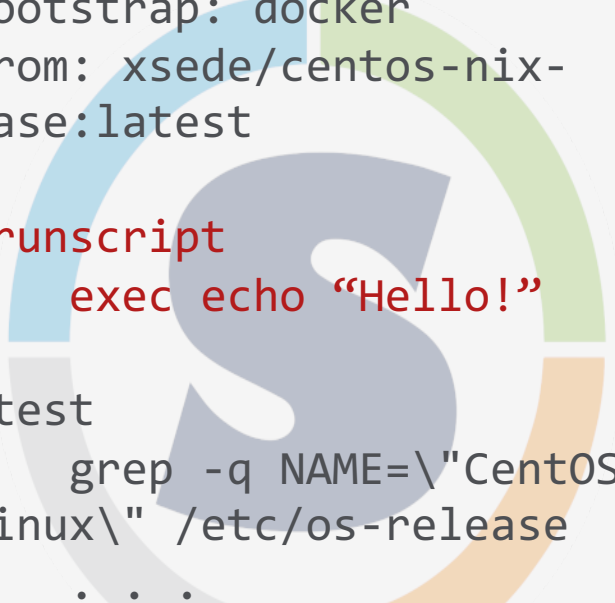
Best Practices

8. Do not use plain text passwords

- Can use the “--docker-login” option for Singularity “pull” and “build” commands

9. Use the “%runscript” environment to execute commands in the container

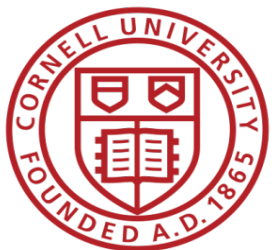
- Removes ambiguity



```
Bootstrap: docker
From: xsede/centos-nix-
base:latest

%runscript
exec echo "Hello!"

%test
grep -q NAME=\"CentOS\
Linux\" /etc/os-release
. . .
```



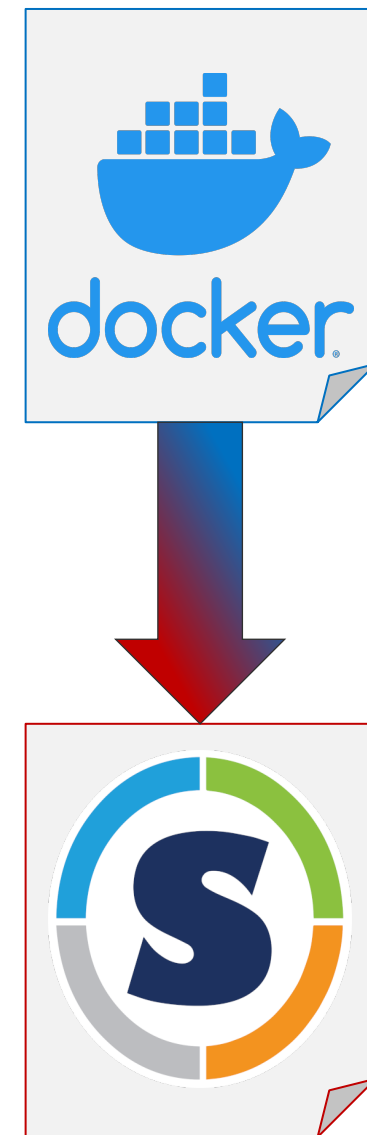
Docker to Singularity Conversion Summary

- DO run commands entirely as the root user in your Dockerfile
- DO a diff on the base image before building a new image of your container
- DO use the ENV directive for environment variables
- DO install to a subdirectory of /opt or /usr/local (recommended)
- DO run “ldconfig” near the end of your Dockerfile
- DO protect secure information

Docker to Singularity Conversion

Example Completed Conversion

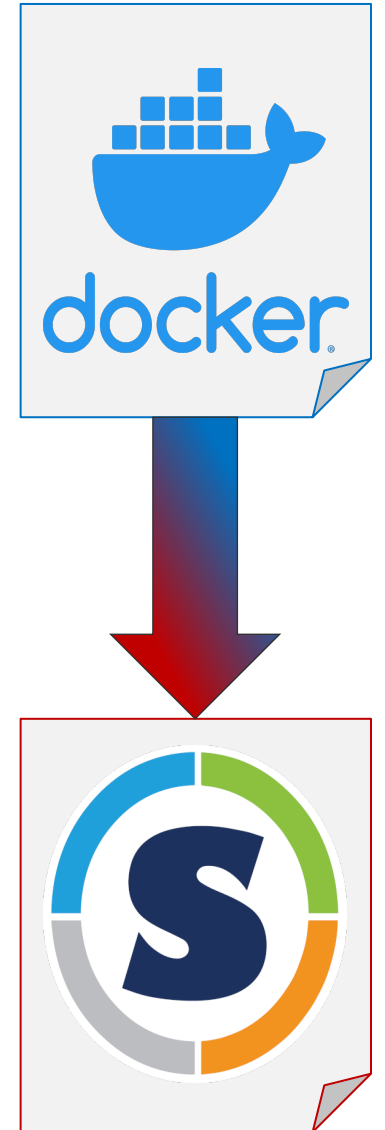
1. **Dockerfile:** <https://github.com/federatedcloud/docker-PRESTO/blob/master/Dockerfile>
 - `docker build ...`



Docker to Singularity Conversion

Example Completed Conversion

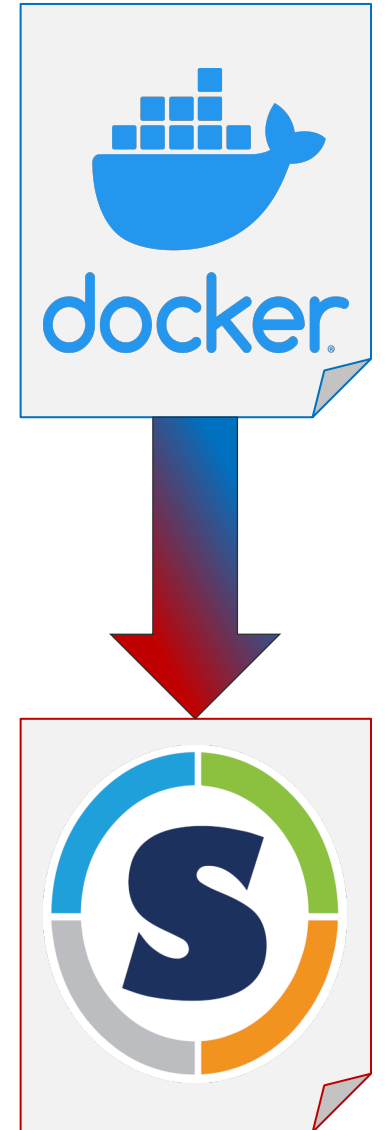
1. **Dockerfile:** <https://github.com/federatedcloud/docker-PRESTO/blob/master/Dockerfile>
 - `docker build ...`
2. Test the **Docker** container
 - `docker image`
 - `docker run <image> <command>`
 - Or `docker run --name=<name> <image> sleep 1000000 &`
`docker exec --it <name> /bin/bash`



Docker to Singularity Conversion

Example Completed Conversion

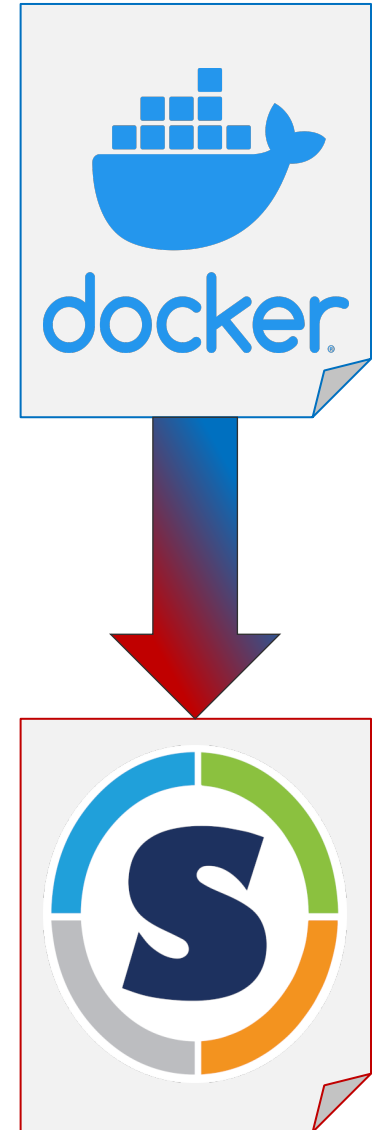
3. Tag the **Docker** image and push to a **DTR**
 - `docker tag <image> <org/repo:tag>`
 - `docker push <org/repo:tag>`



Docker to Singularity Conversion

Example Completed Conversion

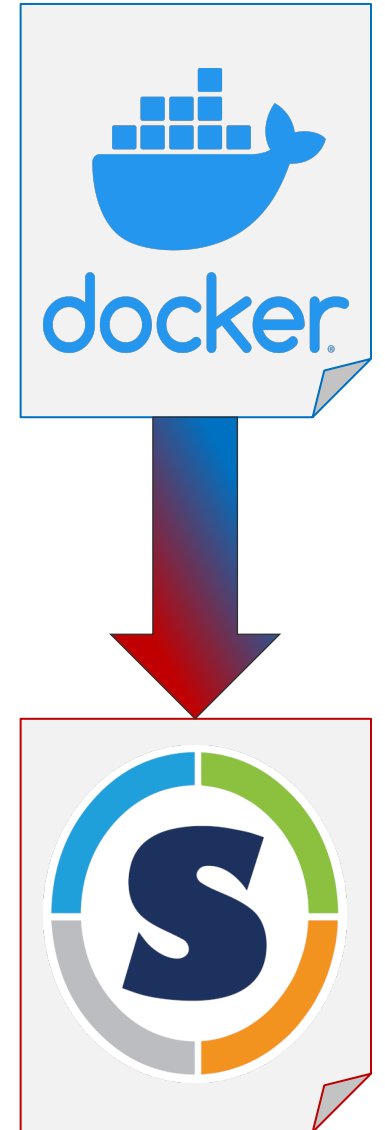
3. Tag the **Docker** image and push to a **DTR**
 - `docker tag <image> <org/repo:tag>`
 - `docker push <org/repo:tag>`
4. **Singularity** Definition file:
<https://github.com/federatedcloud/singularity-PRESTO/blob/master/Singularity>



Docker to Singularity Conversion

Example Completed Conversion

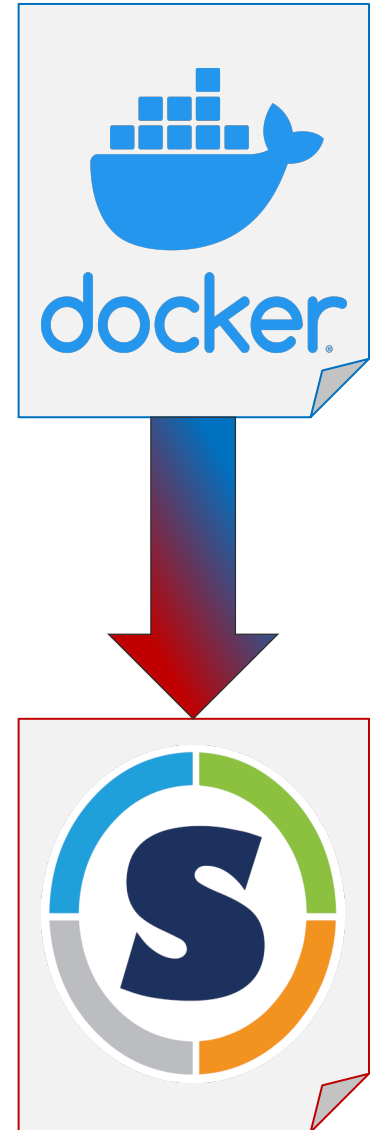
5. Build & Test the **Singularity** container
 - **singularity build ...**



Docker to Singularity Conversion

Example Completed Conversion

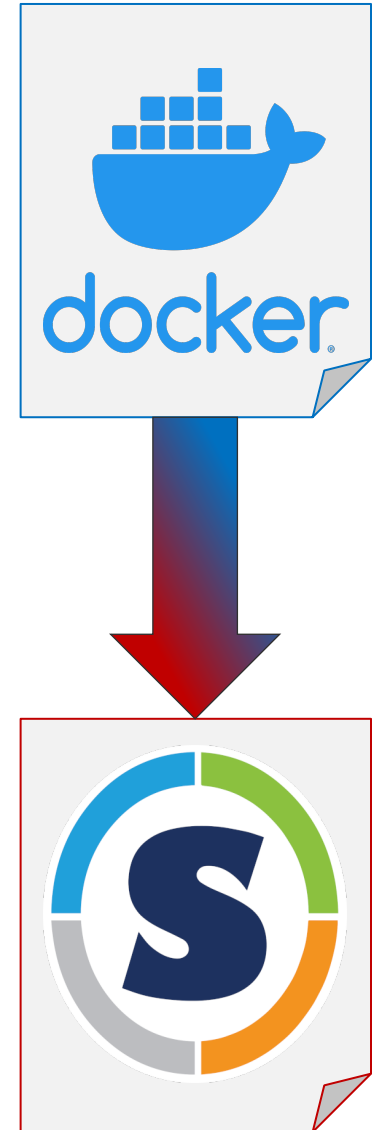
5. Build & Test the **Singularity** container
 - **singularity build ...**
6. (Optional) Push **Singularity** image to a registry

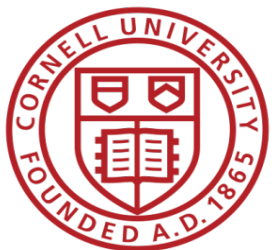


Docker to Singularity Conversion

Overview of the Process

1. Create & Build the **Dockerfile**
2. Test the **Docker** container
3. Tag the **Docker** image and push to a **DTR**
4. Create the **Singularity** Definition file
5. Build & Test the **Singularity** container
6. (Optional) Push **Singularity** image to a registry

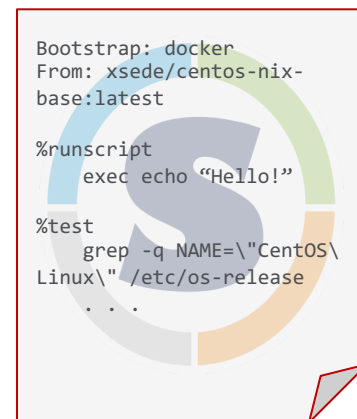


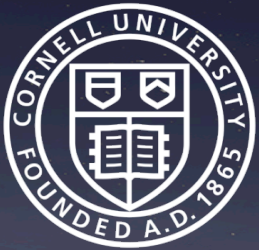


Docker to Singularity Conversion

Also See

- All Tier 1 XSEDE systems have Singularity, for versions see: <https://portal.xsede.org/software#/>
- If you need to troubleshoot the conversion, see: https://sylabs.io/guides/3.5/user-guide/singularity_and_docker.html#troubleshooting
- For other conversion tools, especially for non-Linux users, see: <https://github.com/singularityhub/docker2singularity>





XSEDE

Extreme Science and Engineering
Discovery Environment

Now try it out for yourself!

https://github.com/XSEDE/Container_Tutorial



Peter Vaillancourt
Computational Scientist
Center for Advanced Computing (CAC)
Cornell University

