SC 19 Tutorial: Best Practices

Shane Canon\textsuperscript{1}, Sameer Shende\textsuperscript{2}, Carlos Eduardo Arango\textsuperscript{3}, Andrew J. Younge\textsuperscript{4}

\textsuperscript{1}Lawrence Berkeley National Lab
scanon@lbl.gov

\textsuperscript{2}University of Oregon
sameer@cs.uoregon.edu

\textsuperscript{3}Sylabs Inc
eduardo@sylabs.io

\textsuperscript{4}Sandia National Labs
ajyoung@sandia.gov
Outline

• 13:30 – 13:45 Introduction to Containers in HPC (Younge)
• 13:45 – 14:15 How to build your first Docker container (Canon)
• 14:15 – 14:45 How to deploy a container on a supercomputer (Canon)
• 14:45 – 15:00 Best Practices (Canon)
• 15:00 – 15:30 -- Break --
• 15:30 – 16:00 Running an HPC app on the E4S container (Shende)
• 16:00 - 16:30 How to build a Singularity container image (Arango)
• 16:30 - 16:50 Running Singularity on a supercomputer & adv features (Arango)
• 16:50 - 17:00 Success Stories & Summary (Canon)

Link: https://tinyurl.com/yxbhpo35
General HPC Container Gotchas

- Containers run as the user, not root
- Images are mounted read-only
  - But home, scratch, lustre, … directories are probably available
- Some volume mount locations are disallowed
- Volumes currently can’t be mounted over each other
Best Practice - Build with a script, not manually

FROM ubuntu:14.04
LABEL maintainer="patsmith patsmith@patsmith.org"
ADD ./app /bin/app
RUN mv /bin/app /bin/hello && chmod a+rx /bin/hello
Best Practice – Use Trusted images

FROM foobar/python:3.7 # do you know foobar?

• Solution:

FROM python:3.7 # official image from Python Foundation
FROM library/python:3.7 # equivalently; "library/" is implied
FROM supercontainers/optimized-base/cts-bdw:2019-11-11 # trust us :)
Best Practice – Use versioned dependencies

Solution: (if you have a tagged release)

```bash
RUN git clone --branch v1.0.3 --depth 1 https://github.com/foo/bar.git
RUN cd bar && make install
```

Solution: (if you have a commit hash)

```bash
RUN git clone https://github.com/foo/bar.git
RUN cd bar && git checkout 4e3c9cc && make install
```
Best Practice – Combine RUN commands

RUN wget http://hostname.com/mycode.tgz
RUN tar xzf mycode.tgz
RUN cd mycode; make; make install
RUN rm -rf mycode.tgz mycode

• Solution:

RUN wget http://hostname.com/mycode.tgz && \
   tar xzf mycode.tgz && \
   cd mycode && make && make install && \
   rm -rf mycode.tgz mycode
Best Practice – Avoid Semicolons; Use Ampersands &&

RUN wget http://hostname.com/mycode.tgz ; \
  tar xzf mycode.tgz ; \
  cd mycode ; make ; make install ; \
  rm -rf mycode.tgz mycode

• Solution:

RUN wget http://hostname.com/mycode.tgz && \
  tar xzf mycode.tgz && \
  cd mycode && make && make install && \
  rm -rf mycode.tgz mycode
Best Practice – Order matters, use the build cache

ADD . /src
RUN apt-get update -y && apt-get install gcc
RUN cd /src && make && make install

• Solution:

RUN apt-get update -y && apt-get install gcc
ADD . /src
RUN cd /src && make && make install
Multi-stage Builds

• Added in Docker 17.05
• Allows a build to progress through stages
• Files can be copied from a stage to later stages
• Useful for splitting images between build and run time to keep image sizes small
• Can be used to make public images that make use of commercial compilers
Best Practice – Multi-stage Builds

FROM centos:7 as build
RUN yum -y install gcc make
ADD code.c /src/code.c
RUN gcc -o /src/mycode /src/code.c

FROM centos:7 as run
COPY --from=build /src/mycode /usr/bin/mycode
Other considerations

• Avoid very large images (>~5GB)
• Keep application data in Home, Scratch, Lustre, … and volume mount into the container if data is large
• Use volume mounts for rapid prototyping and testing, then add that into the image after code stabilizes
Time for a Break!
Questions?

Try it on our own and experiment with deploying your own HPC application in containers!