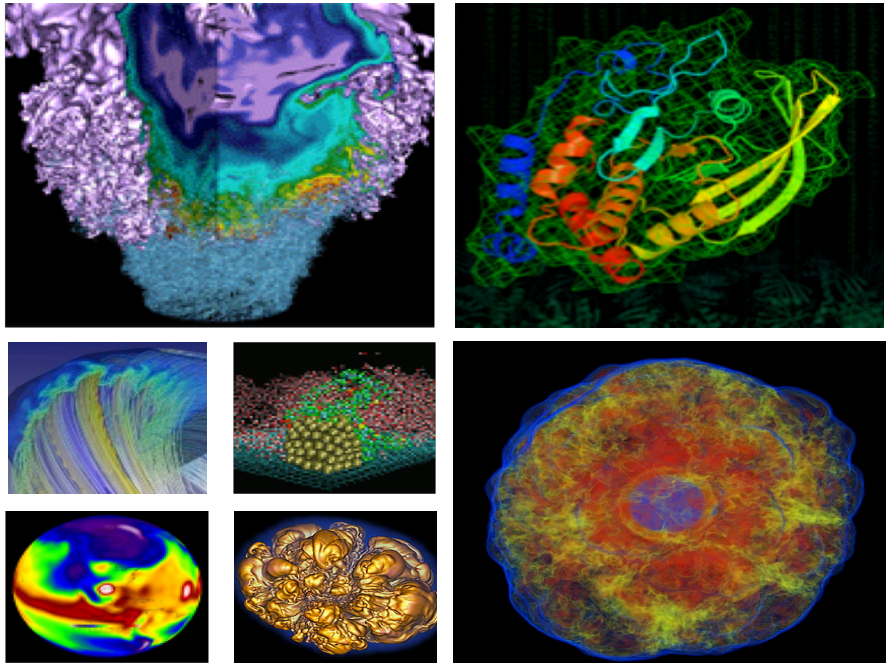
Experience supporting Containers with Shifter at NERSC



Shane Canon

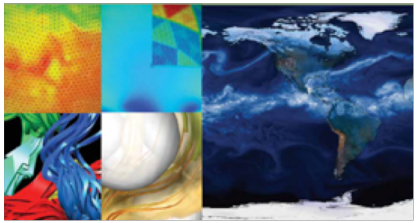
Data & Analytics Group, NERSC

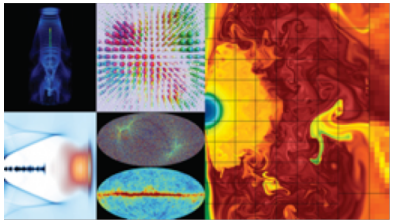
Agenda

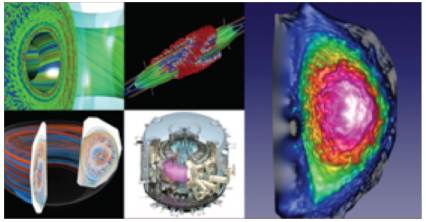
- Motivations for containers at NERSC
- Shifter: Goals and Design
- Experience supporting containers at NERSC

Why Containers at NERSC

- NERSC deploys advanced HPC and data systems for the broad Office of Science community
- Approximately 6000 users and 750 projects
- Growing number of users around Analyzing Experimental and Observational Data, ”Big Data” Analytics, and Machine Learning
- Shift towards converged systems that support traditional modeling and simulation workloads plus new models







The Struggles

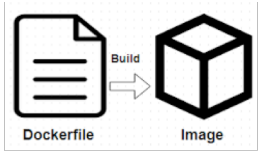
- My software doesn't build on this system…
- I'm missing dependencies…
- I need version 1.3.2 but this system has version 1.0.2..
- I need to re-run the exact same thing 12 months from now…
- I want to run this exact same thing somewhere else…
- I want my collaborators to have the same exact software as me…
- I've heard about these Containers, can I just run that?
- Can I run docker on this HPC system?

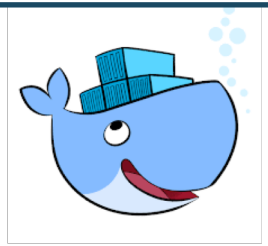
Docker Basics

- Build images that captures applications requirements.
- Manually commit or use a recipe file.

- Push an image to DockerCloud, a hosted registry, or a private Docker Registry.
- Share Images

- Use Docker Engine to pull images down and execute a container from the image.



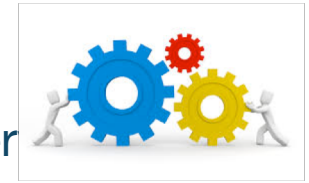


Why not just run Docker

- Security: Docker currently uses an all or nothing security model. Users would effectively have system privileges

```
> docker run -it -v /:/mnt --rm busybox
```

- System Architecture: Docker assumes local disk
- Integration: Docker doesn't play nice with batch systems.
- System Requirements: Docker typically requires a ver modern kernel
- Complexity: Running real Docker would add new layers of complexity



Long History with Custom Environments

- ~2003: ChrootOS(CHOS) – System to enable projects to have customized environments on a shared cluster (PDSF). Integrated with login and batch and used a custom kernel module to provide a seamless experience.
- ~2014: MyDock – Thin wrapper around Docker to allow users to securely run Docker on a shared cluster (Jesup/Carver).
- ~2015: Shifter: “User Defined Environments”

Shifter (HPC) versus Spin (Services)

Shifter

Runs processes as the user

Runs on the HPC systems

Best for:

Simulation or analysis runs

Need to run at scale

Need to read/write a lot of data

Spin

Runs with stock Docker and Rancher

Runs on dedicated hardware

Best for:

Running services or processes that need to run “indefinitely”

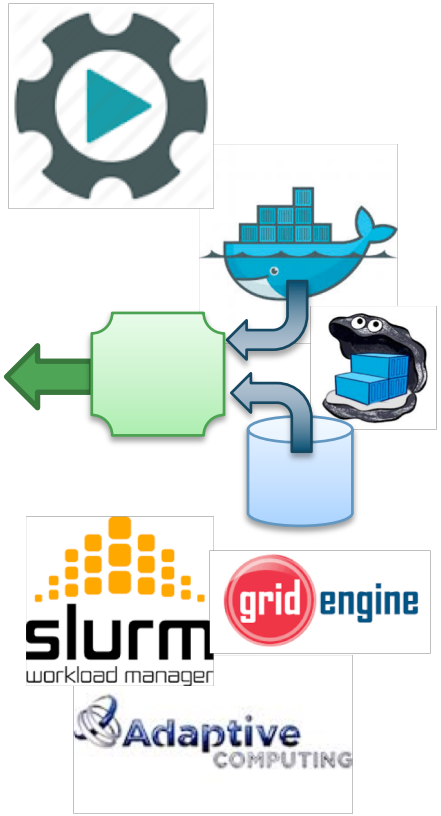
Services that need to be externally accessible

Solution: Shifter

- Design Goals:
  - User independence: Require no administrator assistance to launch an application inside an image
  - Shared resource availability (e.g., file systems and network interfaces)
  - Leverages or integrates with public image repos (i.e. DockerHub)
  - Seamless user experience
  - Robust and secure implementation
- Hosted at GitHub:
  - https://github.com/nersc/shifter

Shifter Components

- Shifter Image Gateway
  - Imports and converts images from DockerHub and Private Registries
- Shifter Runtime
  - Instantiates images securely on compute resources
- Work Load Manager Integration
  - Integrates Shifter with WLM
  - Implementations include Slurm, UGE, and Torque/Moab



Usage

- Use shifterimg pull to pull images from a registry
  - Only do this once or after an update

```
> shifterimg pull ubuntu:14.04
```

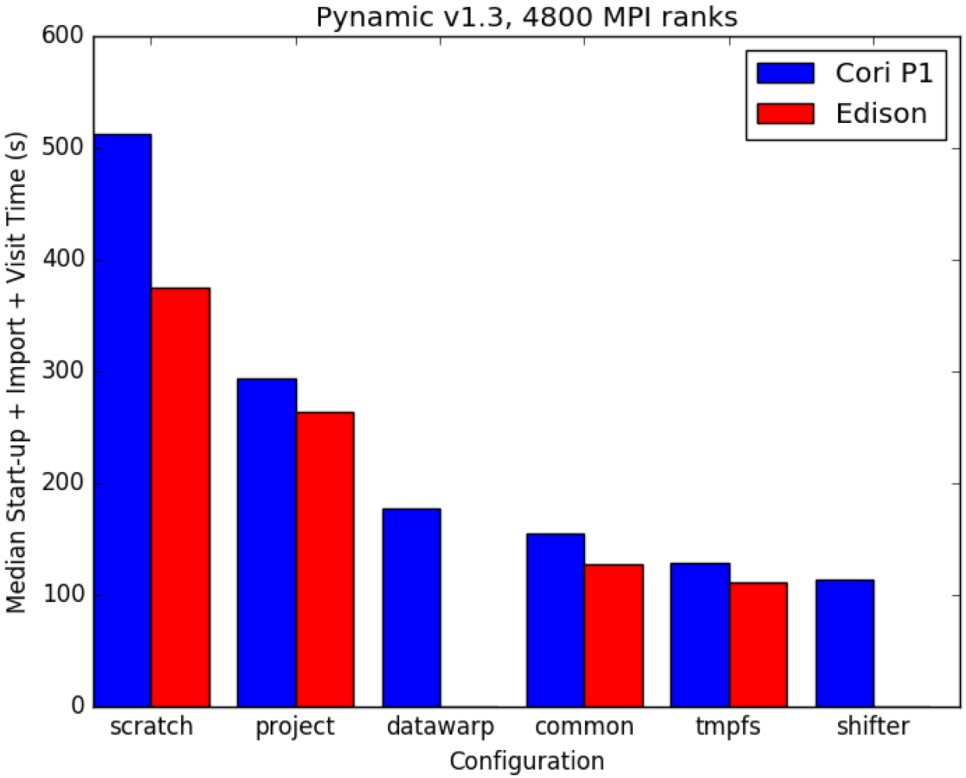
- Use shifter command to run a container with an image

```
> shifter --image=ubuntu:14.04 bash
$ lsb_release -a
No LSB modules are available.
Distributor ID:	Ubuntu
Description:	Ubuntu 14.04.5 LTS
Release:	14.04
Codename:	trusty
```

Containers and Science

- Productivity
  - Pick the OS that works best for your app and use the system package manager to install dependencies.
- Reusability and Collaboration
  - Share images across a project to avoid rebuilds and avoid mistakes
- Reproducibility
  - Everything you need to redo a scientific analysis can be in the image (apps, libraries, environment setup, scripts)
- Portability
  - Can easily run on different resources (of the same architecture)

Shifter accelerates many Apps



Shifter and MPI

- In Image
  - Add required libraries directly into image.
  - Users would have to maintain libraries and rebuild images after an upgrade.
- Managed Base Image (Golden Images)
  - User builds off of a managed image that has required libraries.
  - Images are built or provided as part of a system upgrade.
  - Constrained OS choices and a rebuild is still required.
- Volume Mounting
  - Applications built using ABI compatibility.
  - Appropriate libraries are volume mounted at run time.
  - No rebuild required, but may not work for all cases.

Running an MPI Job – Building Image

Dockerfile

```
FROM nersc/mpi-ubuntu:14.04

ADD . /app
RUN cd /app && \
    mpicc -o hello helloworld.c
```

```
> docker build –t scanon/hello .
> docker push scanon/hello
```

Running an MPI Job – Submit and run

submit.sl

```
#!/bin/sh
#SBATCH --image= scanon/hello
srun -np 10 shifter /app/hello
```

```
> sbatch submit.sl
```

How does Shifter differ from Docker?

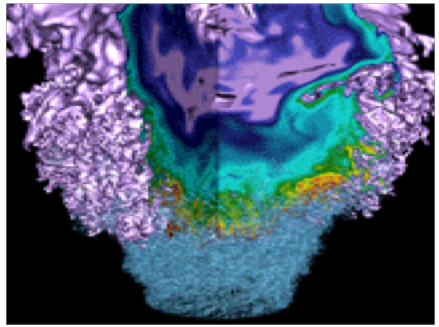
Most Noticeable

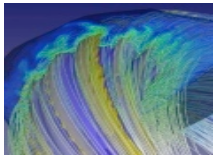
- Image read-only on the Computational Platform
- User runs as the user in the container – not root
- Image modified at container construction time (e.g. additional mounts)

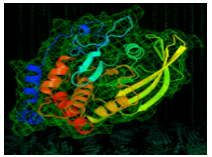
Less Noticeable:

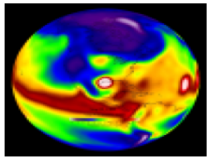
- Shifter only uses mount namespaces, not network or process namespaces
- Shifter does not use cgroups directly (integrated with the Workload Manager)
- Shifter uses individual compressed filesystem files to store images, not the Docker graph (slows down iterative updates)
- Shifter starts some additional services (e.g. sshd in container space)

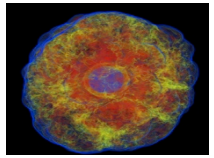
Other Tips and Tricks

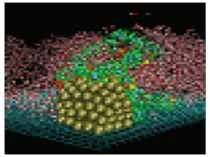












Volume Mounts

- Volume Mounts provide a way to map external paths into container paths.
- This allows paths in the container to be abstracted so it can be portable across different systems.
- All runtimes support volume mounts but the syntax may vary.
- Basic syntax is:

```
-volume <external path>:<container path>
```

Using Volume Mounts

```
canon@cori06:~> ls $SCRATCH/myjob
config  data.in

canon@cori06:~> shifter --image=ubuntu \
        --volume=$SCRATCH/myjob:/data bash
~$ ls /data/
config  data.in
```

PerNode Write Cache (Shifter)

- PerNodeWrite extends the volume concept to create temporary writeable space that aren't shared across nodes.
- These spaces are ephemeral (removed on exit)
- These are node local and the size can be adjusted
- Performs like a local disk but is more flexible
- Basic syntax is

```
--volume <external path>:<container path>:perNodeCache=size=XXG
```

Using PerNode Mounts

```
canon@cori06:~> shifter --image=ubuntu \
      --volume=$SCRATCH:/scratch:perNodeCache=size=100G /bin/bash
~$ df -h /scratch/
Filesystem      Size  Used Avail Use% Mounted on
/dev/loop4      100G   33M  100G   1% /scratch
~$ dd if=/dev/zero bs=1k count=10M of=/scratch/output
10737418240 bytes (11 GB, 10 GiB) copied, 22.2795 s, 482 MB/s
~$ ls -lh /scratch/output
-rw-r--r-- 1 canon canon 10G Nov  9 23:38 /scratch/output
~$ exit
canon@cori06:~> shifter --image=ubuntu \
     --volume=$SCRATCH:/scratch:perNodeCache=size=100G /bin/bash
~$ ls -l /scratch
total 0
```

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

Dockerfile – Multistage build

```
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
COPY --from=build /src/mycode /usr/bin/mycode
```

Multistage Builds with Intel Compilers

- We have a model for compiling applications using Intel Compilers
- Requires tunneling a connection to NERSC's license server
- Uses multistage build so resulting runtime image doesn't contain any licensed software
- Images are hosted in NERSC's registry which requires a NERSC login
- Documented on NERSC's web site
- NERSC Postdoc Jonathan Madsen figured this all out

Jonathan Madsen

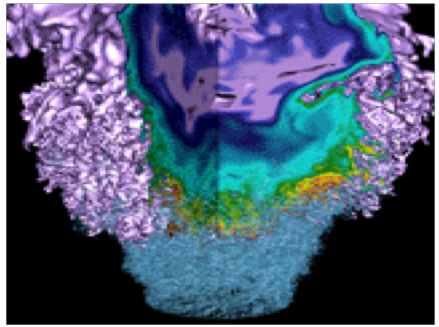
Example Dockerfile

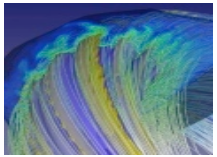
```
######## Stage 1 ########
FROM registry.services.nersc.gov/nersc/intel_cxx_mpi_devel as builder
# ... build your code with "devel" image variant ...
# ... recommended to use a common install prefix, such as "/opt/local", e.g.
#

ENV CC /opt/intel/bin/icc
ENV CXX /opt/intel/bin/icpc

RUN cd ${HOME}/package_a && \
    ./configure --prefix=/opt/local && \
    make install -j8 && \
    cd ${HOME}/package_b && \
    mkdir build_package_b && \
    cd build_package_b && \
    cmake -DCMAKE_INSTALL_PREFIX=/opt/local .. && \
    make install -j8

######## Stage 2 ########
# ... don't have to clean above, just copy over installation
FROM registry.services.nersc.gov/nersc/intel_cxx_mpi_runtime
COPY --from=builder /opt/local/ /opt/local
RUN echo '/opt/local/lib' > /etc/ld.so.conf.d/package-libs.conf && \
    ldconfig
```

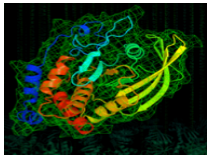
Shane's Advice

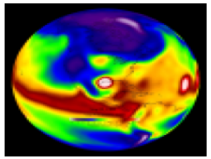
- Use a Dockerfile - Leverages the build caching and is the most portable
- Use a multi-stage build – Helps with size and licensing issues
- Create base images – Base images that are common across a suite of applications
- Version images – Helps with reproducibility
- Leverage ABI – Best supported in MPICH but hopefully this improves
- Use Volume Mounts – Separate data from the application. Use volume mounts for data
- Run as non-root – Design your images to run as non-root (most portable and safe)

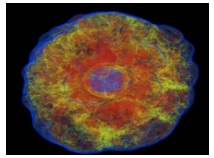
Use Case Example and Summary

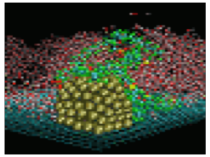






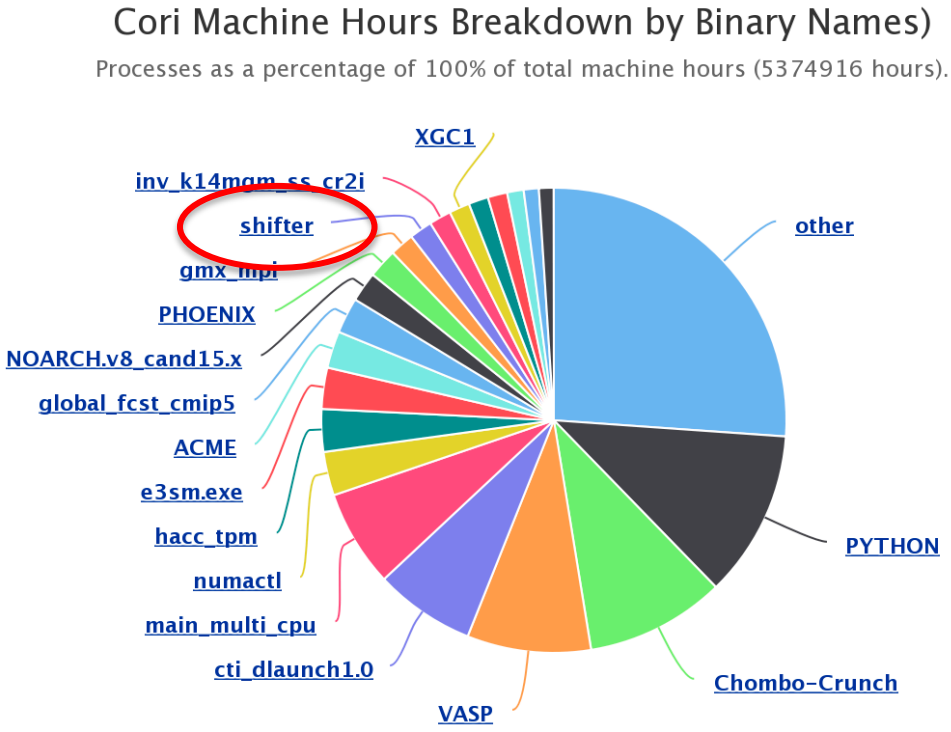






Stats on Shifter Use at NERSC

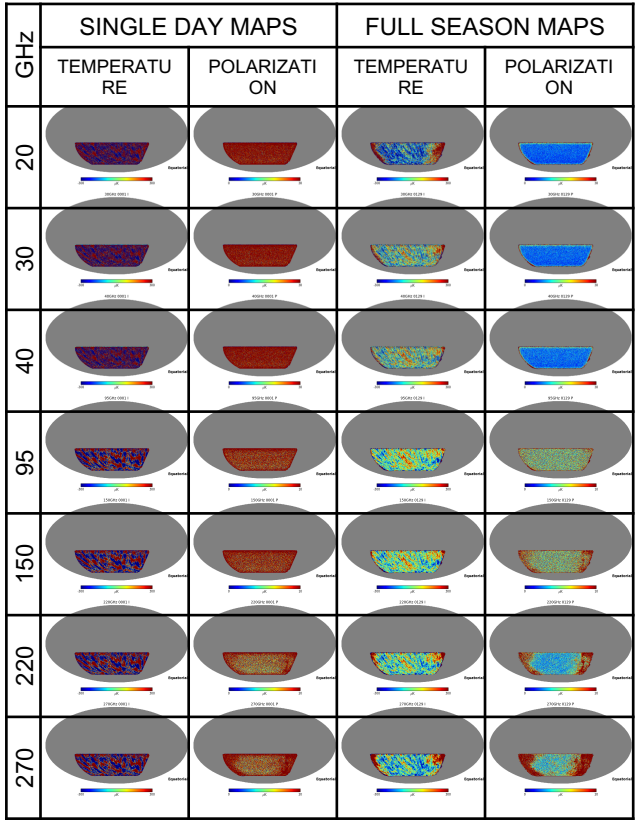
- 10M+ Image Lookups
- 1900+ Unique Image tags
- 700+ Unique Users
- Still a small fraction of NERSC overall use (~3%)



Shifter in Action

Measuring the Composition of the Universe

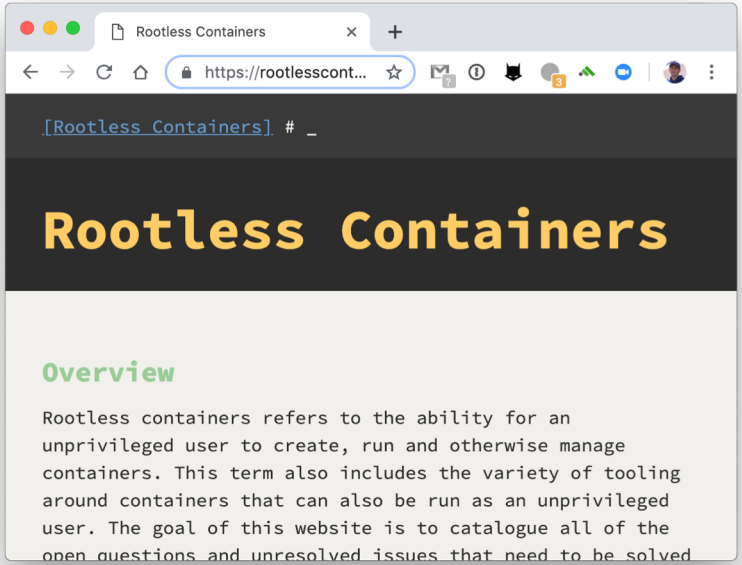
- **CMB – S4** - Ambitious collection of telescopes to measure the remnants of the Big Bang with unprecedented precision
- Simulated 50,000 instances of telescope using **600,000 cores** on Cori KNL nodes.
- Why Shifter?
  - Python wrapped code needs to start at scale
  - Without Shifter application timed out during startup
- Uses Intel Compilers



Courtesy of Ted Kisner

What's next?

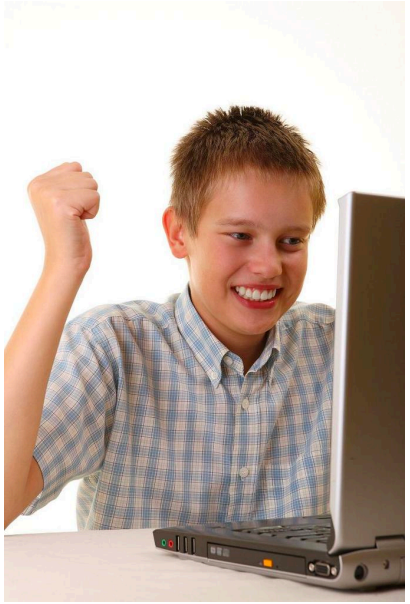
- Continued training and development best practices (in partnership with ECP projects)
- Preparing for Perlmutter (GPU support) – Leverage work done by CSCS
- Exploring rootless/nosuid for Shifter
- Exploring emerging rootless support in the general docker/container sphere (Docker, podman)



Summary

Containers are great

- ✓ Productivity – Get exactly what you need for your application
- ✓ Portable – Run the same software on different resources (assuming architectural compatibility)
- ✓ Sharable – Collaborators can run the same code as you with less chance of problems
- ✓ Reproducible – Run the same image later
- ✓ Performant – Can actually speed up applications in some cases



Questions…

Shane Canon: scanon@lbl.gov

Upcoming Events:

- June 11-13 - Jupyter Facilities Workshop
- June 20 – HPCW at ISC-HP
- Nov - SC19: Stay tuned

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research of the U.S. Department of Energy under Contract No. **DE-AC02-05CH11231**.