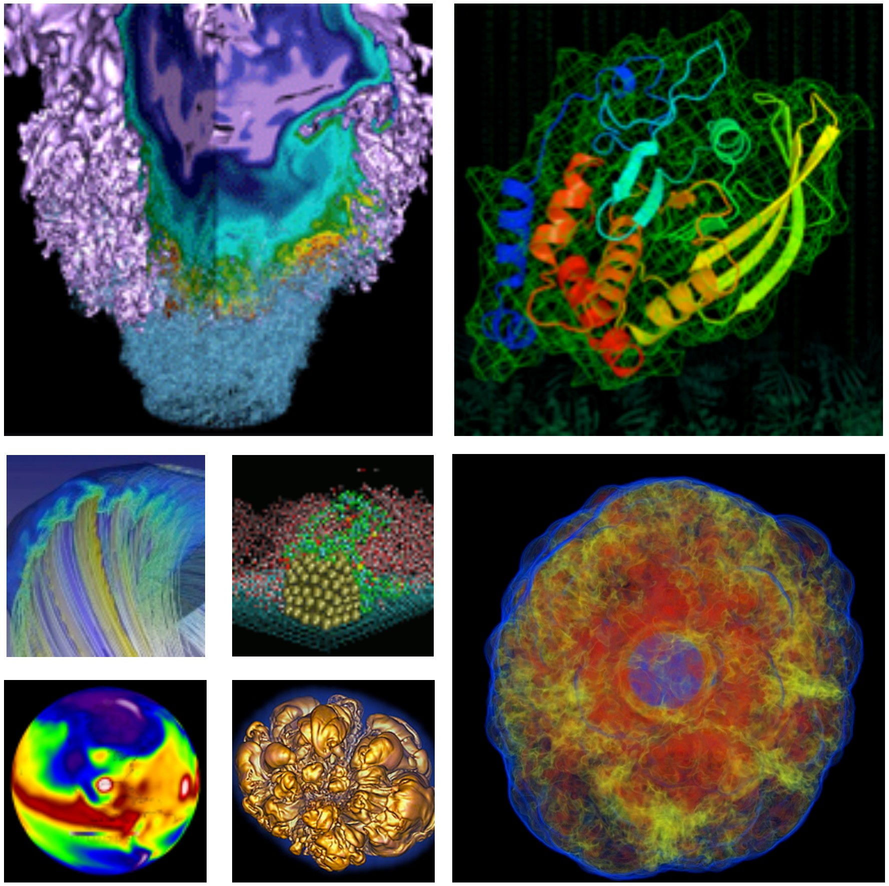
Using Docker and Shifter at NERSC



Shane Canon

NERSC Data and Analytics Services

February 24, 2017

Goals

- **Quick Intro to Docker and Shifter**
- **Examples of how Shifter is already being used**
- **Quick walk through of:**
  - Creating a Docker image
  - Running an container locally
  - Pushing an image to a registry
  - Pulling an image to NERSC
  - Running an image with Shifter

Shifter: Bringing Containers to HPC

- **Docker: open source, automated container deployment service**
  - Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run (code, runtime, system tools and libraries)
  - Guaranteed to operate the same, regardless of the environment in which it is running
- **NERSC has partnered with Cray to deliver Docker-like container technology through a new software package known as Shifter**

Shifter at NERSC

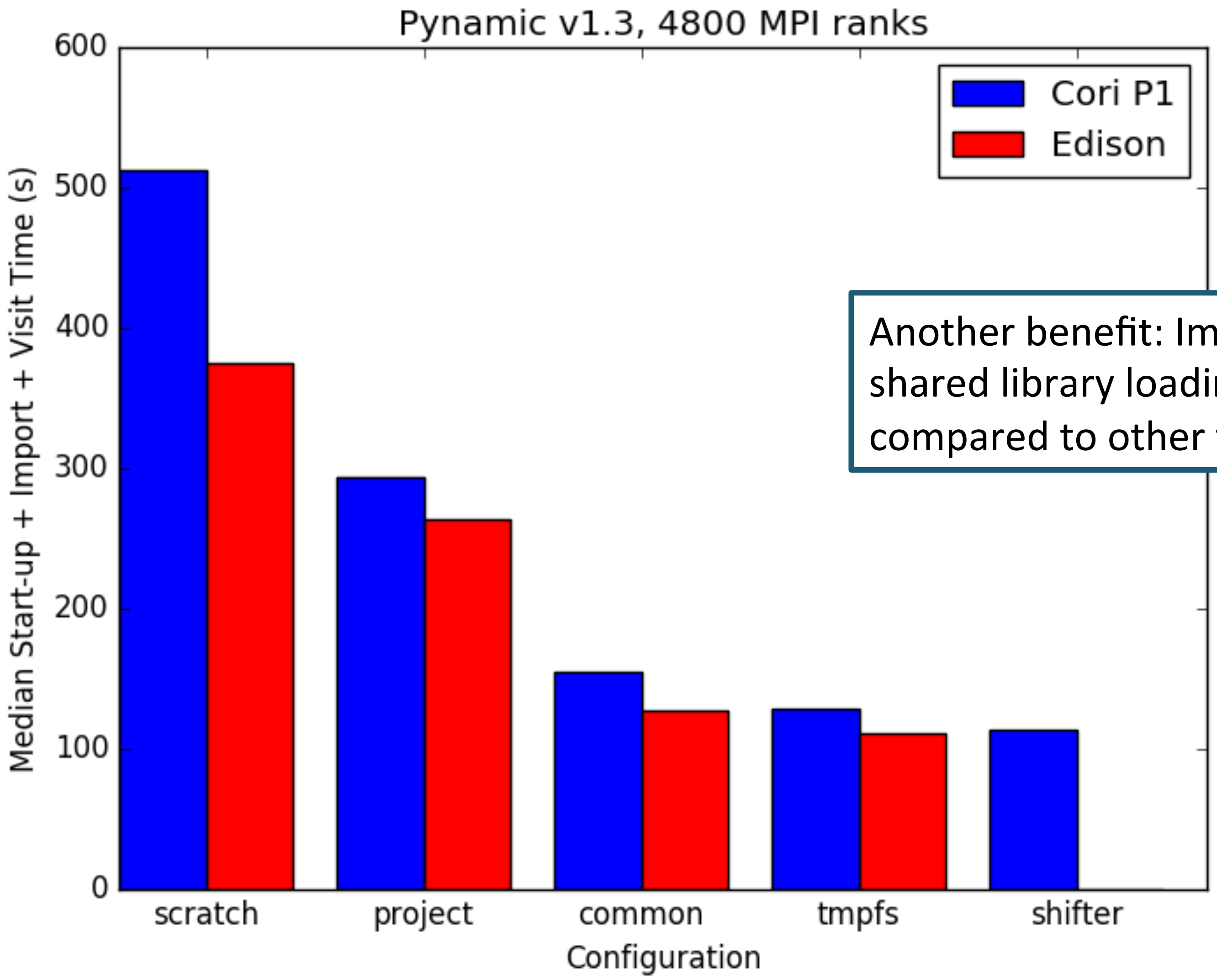
- **Secure and scalable way to deliver containers to HPC**
- **Implemented on Cori and Edison**
- **Supports Docker images and other image formats (ext4, squashfs)**
- **Basic Idea**
  - Users create custom images in desired OS
  - Upload image to docker hub and pull down on HPC system
  - Hooked into the batch system

http://www.nersc.gov/users/software/using-shifter-and-docker/

Why Use Shifter?

- **Shifter allows you to fully customize your operating environment**
  - Want SL 6.X with 32 bit libraries? Use Shifter
  - Have a very complicated software stack with lots of dependencies? Use Shifter
- **Portability**
  - Can volume mount directories into shifter images
    - Have an /input and /output that are linked to directories in your scratch directory
  - Images are NERSC-independent, can be run anywhere

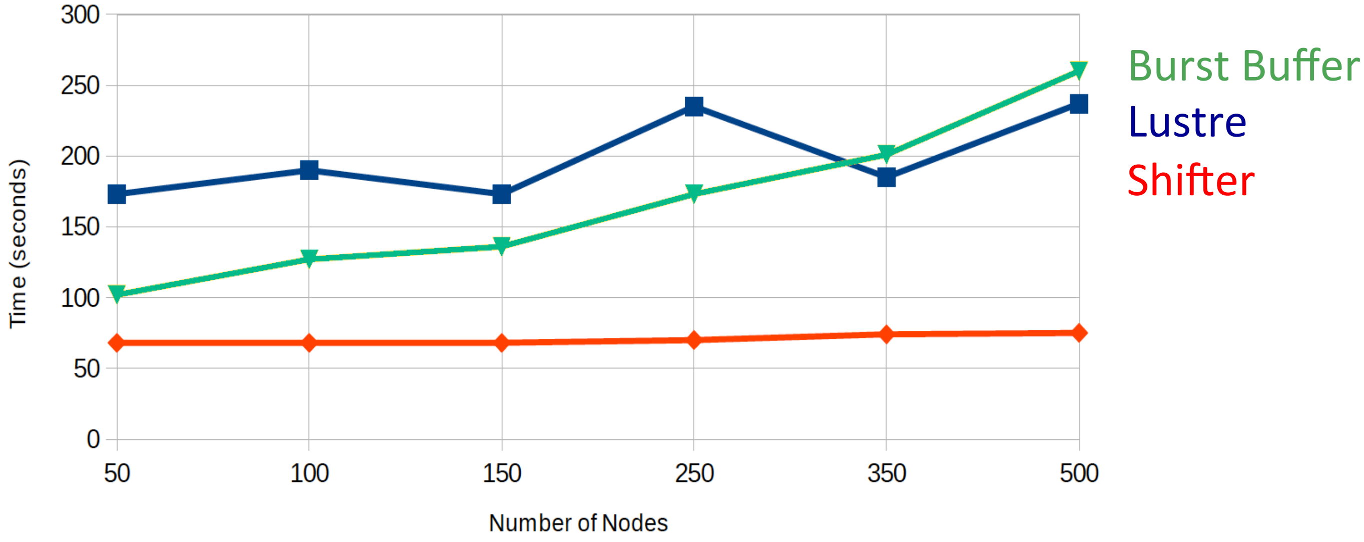
Shifter is Fast



Another benefit: Improved shared library loading times compared to other file system

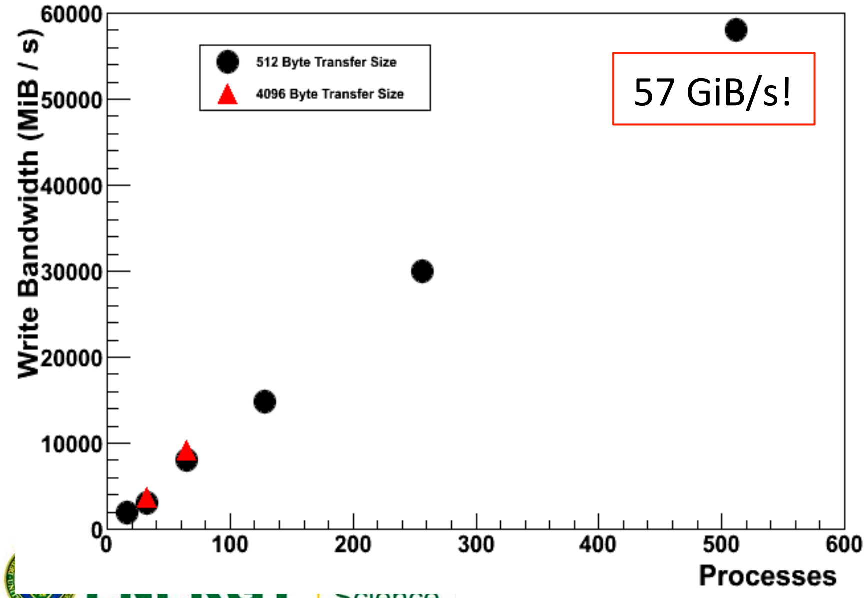
Even Big Images Load Quickly

- **As proof of concept created “Mega” CVMFS shifter image**
  - Full CVMFS stack pulled down and deduped with uncvmfs software stack. 1 – 3 TB ext4 file uncompressed, 300 GB compressed w/ squashfs
- **Use Shifter to load job**
  - Add a single flag to batch script “--image=<image name>”
  - ATLAS cvmfs repository is found at /cvmfs/atlas.cern.ch like normal

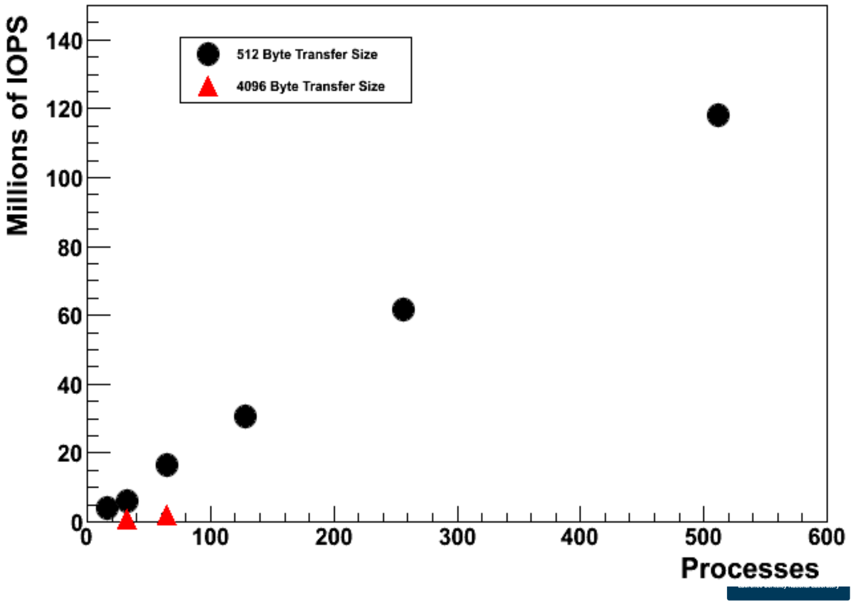


Loop Mounted FS for Super Fast I/O

- **Shifter can mount an xfs file system on each node**
  - Created when job starts and destroyed when job ends
  - Compute node “local disk”
  - Excellent I/O rates:
    - Small databases
    - Also good for “bad IO”



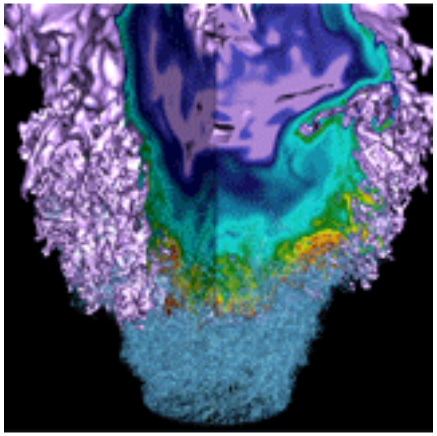
57 GiB/s!

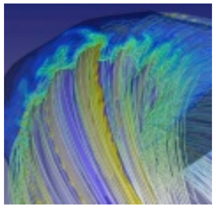


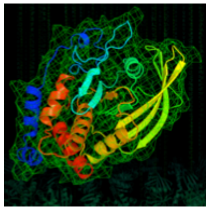
Shifter and MPI

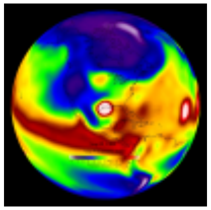
- MPI communication over Aries network is available by default for Shifter on Cori
- In the image, build and link against standard MPICH libraries
- Cray libraries swapped in at run time by front loading LD_LIBRARY_PATH

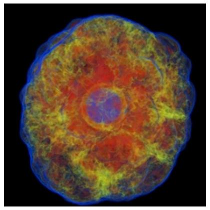
Walk Through

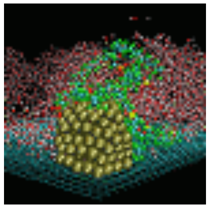












Basic Workflow

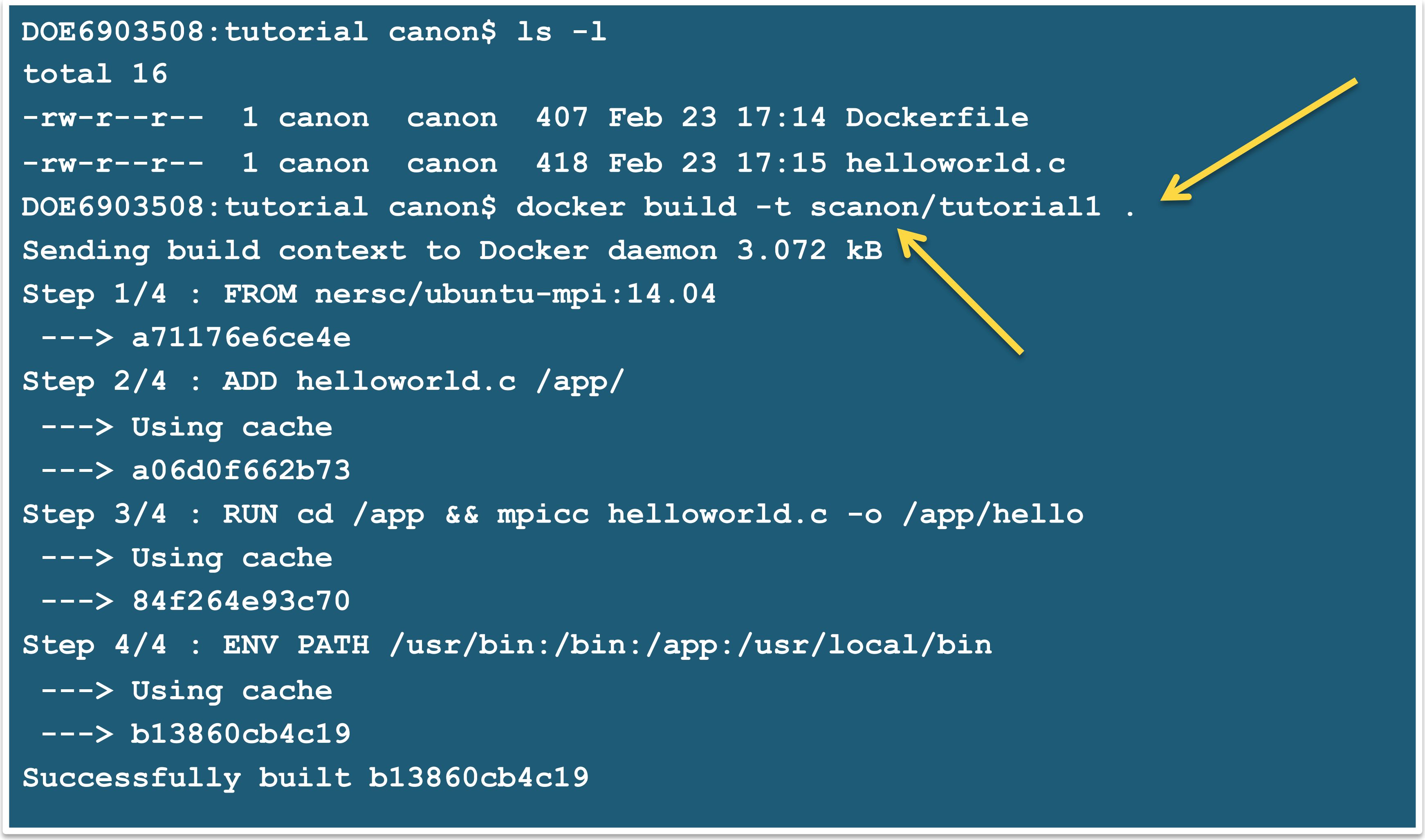
- Install Docker on your laptop or workstation
- Create a Dockerfile that installs and builds your application and any dependencies
- Test this Docker Image locally
- Push the image to a registry (DockerHub)
- Pull the image to NERSC
- Run the Image

https://github.com/NERSC/2016-11-14-sc16-Container-Tutorial

Example Dockerfile

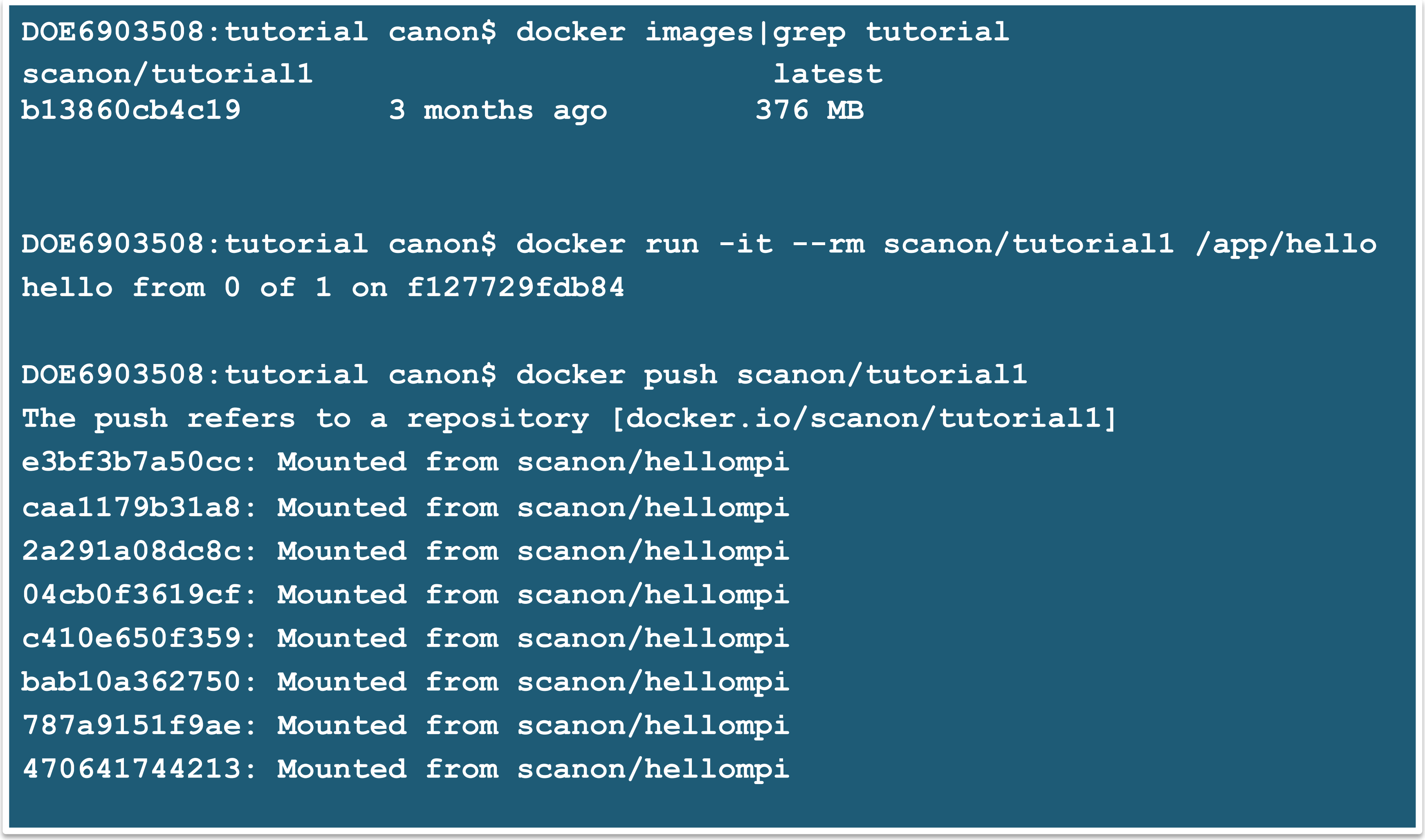
```
# This example makes use of an Ubuntu-based NERSC base image
# that already has MPI built and installed.
#
# This means the you just need to add your app code in and compile it.
#
# To build this example do:
# docker build -t <dockerhubid>/hellompi:latest .
#
FROM nersc/ubuntu-mpi:14.04

ADD helloworld.c /app/

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

ENV PATH=/usr/bin:/bin:/app:/usr/local/bin
```

Building and running image



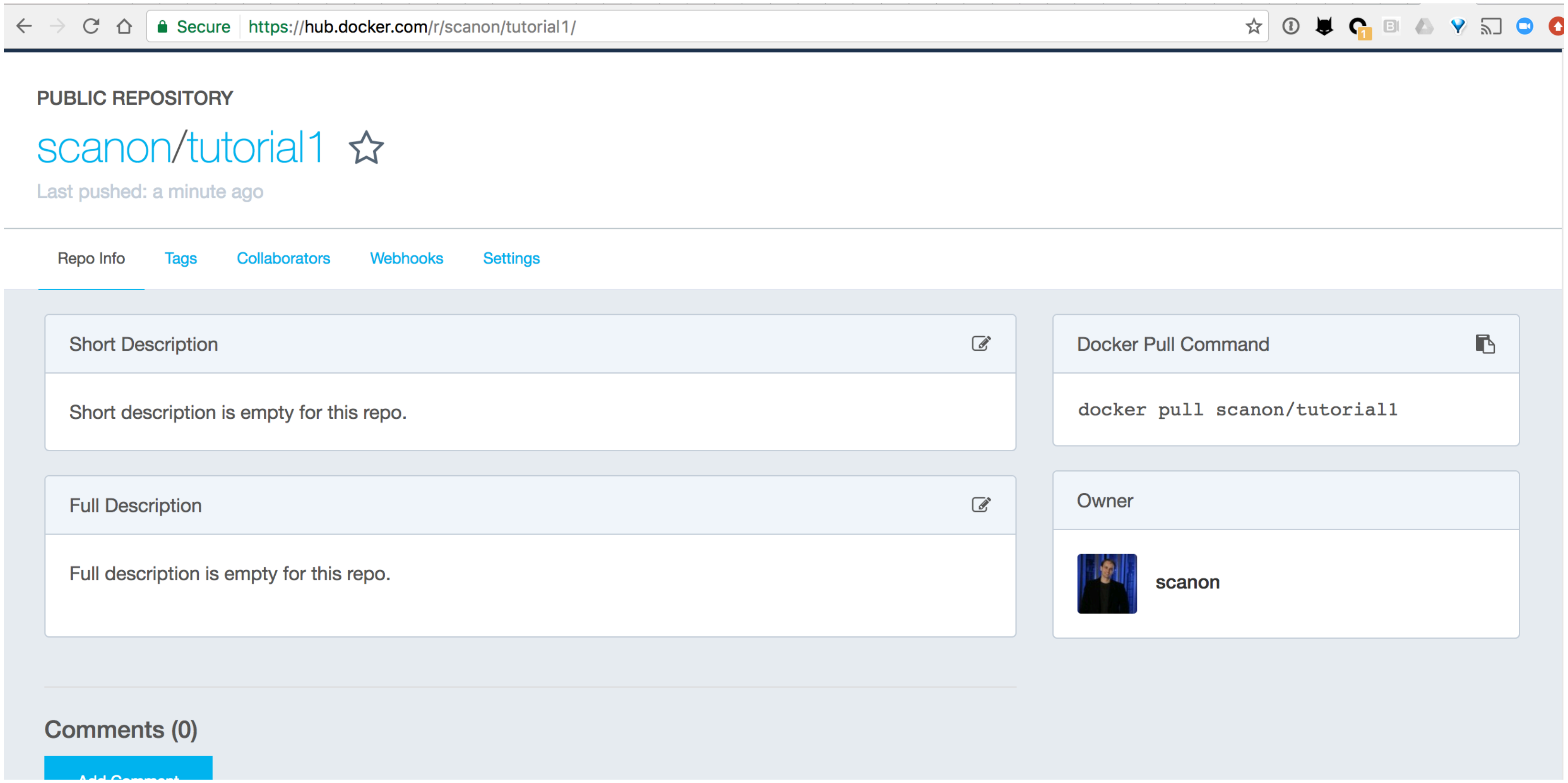
```
DOE6903508:tutorial canon$ ls -l
total 16
-rw-r--r--  1 canon  canon  407 Feb 23 17:14 Dockerfile
-rw-r--r--  1 canon  canon  418 Feb 23 17:15 helloworld.c
DOE6903508:tutorial canon$ docker build -t scanon/tutorial1 .
Sending build context to Docker daemon 3.072 kB
Step 1/4 : FROM nersc/ubuntu-mpi:14.04
 ---> a71176e6ce4e
Step 2/4 : ADD helloworld.c /app/
 ---> Using cache
 ---> a06d0f662b73
Step 3/4 : RUN cd /app && mpicc helloworld.c -o /app/hello
 ---> Using cache
 ---> 84f264e93c70
Step 4/4 : ENV PATH /usr/bin:/bin:/app:/usr/local/bin
 ---> Using cache
 ---> b13860cb4c19
Successfully built b13860cb4c19
```

Running and pushing the image



```
DOE6903508:tutorial canon$ docker images|grep tutorial
scanon/tutorial1                         latest
b13860cb4c19        3 months ago        376 MB



DOE6903508:tutorial canon$ docker run -it --rm scanon/tutorial1 /app/hello
hello from 0 of 1 on f127729fdb84

DOE6903508:tutorial canon$ docker push scanon/tutorial1
The push refers to a repository [docker.io/scanon/tutorial1]
e3bf3b7a50cc: Mounted from scanon/hellompi
caa1179b31a8: Mounted from scanon/hellompi
2a291a08dc8c: Mounted from scanon/hellompi
04cb0f3619cf: Mounted from scanon/hellompi
c410e650f359: Mounted from scanon/hellompi
bab10a362750: Mounted from scanon/hellompi
787a9151f9ae: Mounted from scanon/hellompi
470641744213: Mounted from scanon/hellompi
```

Image on Dockerhub



PUBLIC REPOSITORY

scanon/tutorial1

Last pushed: a minute ago

Short Description

Short description is empty for this repo.

Full Description

Full description is empty for this repo.

Docker Pull Command

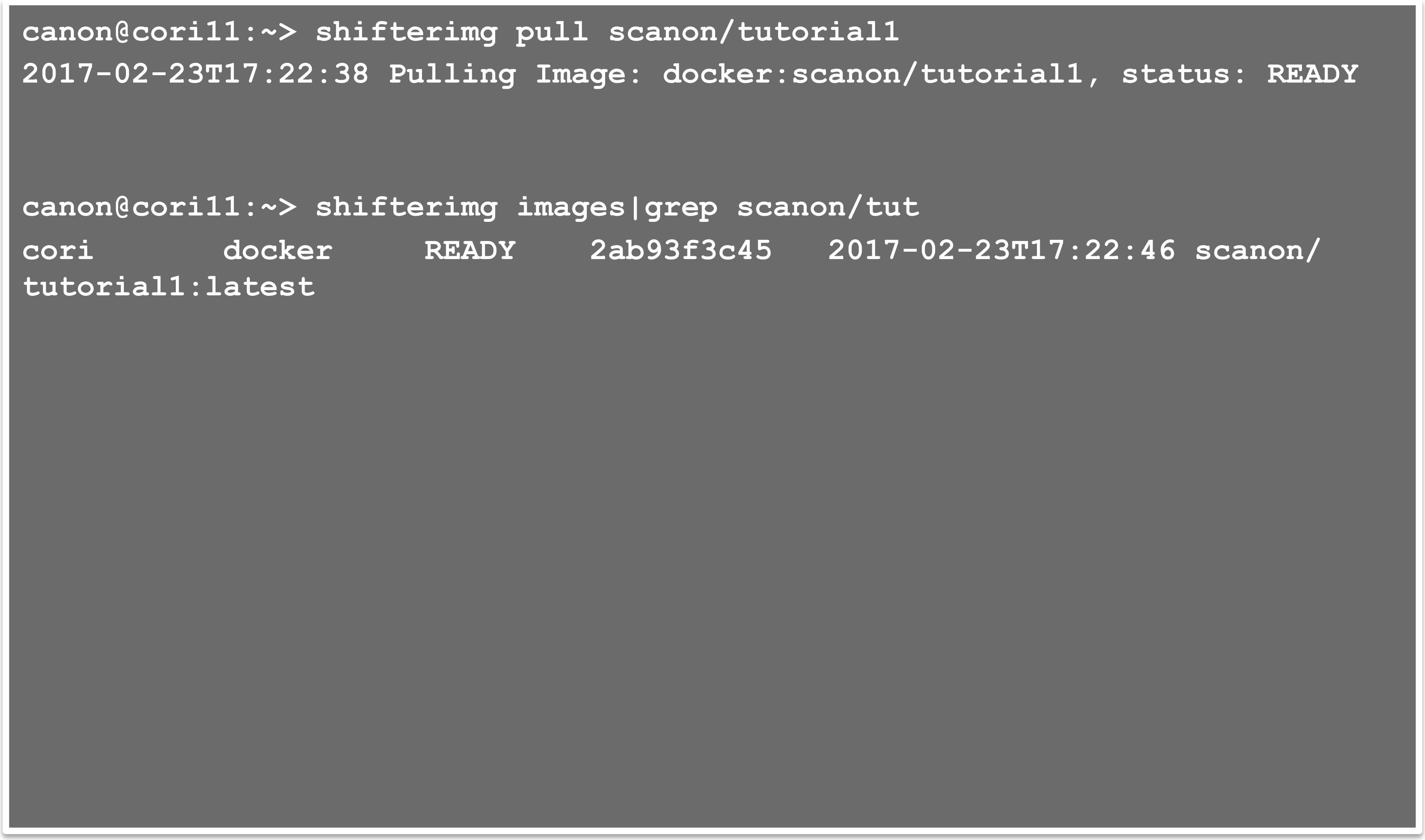
```
docker pull scanon/tutorial1
```

Owner

scanon

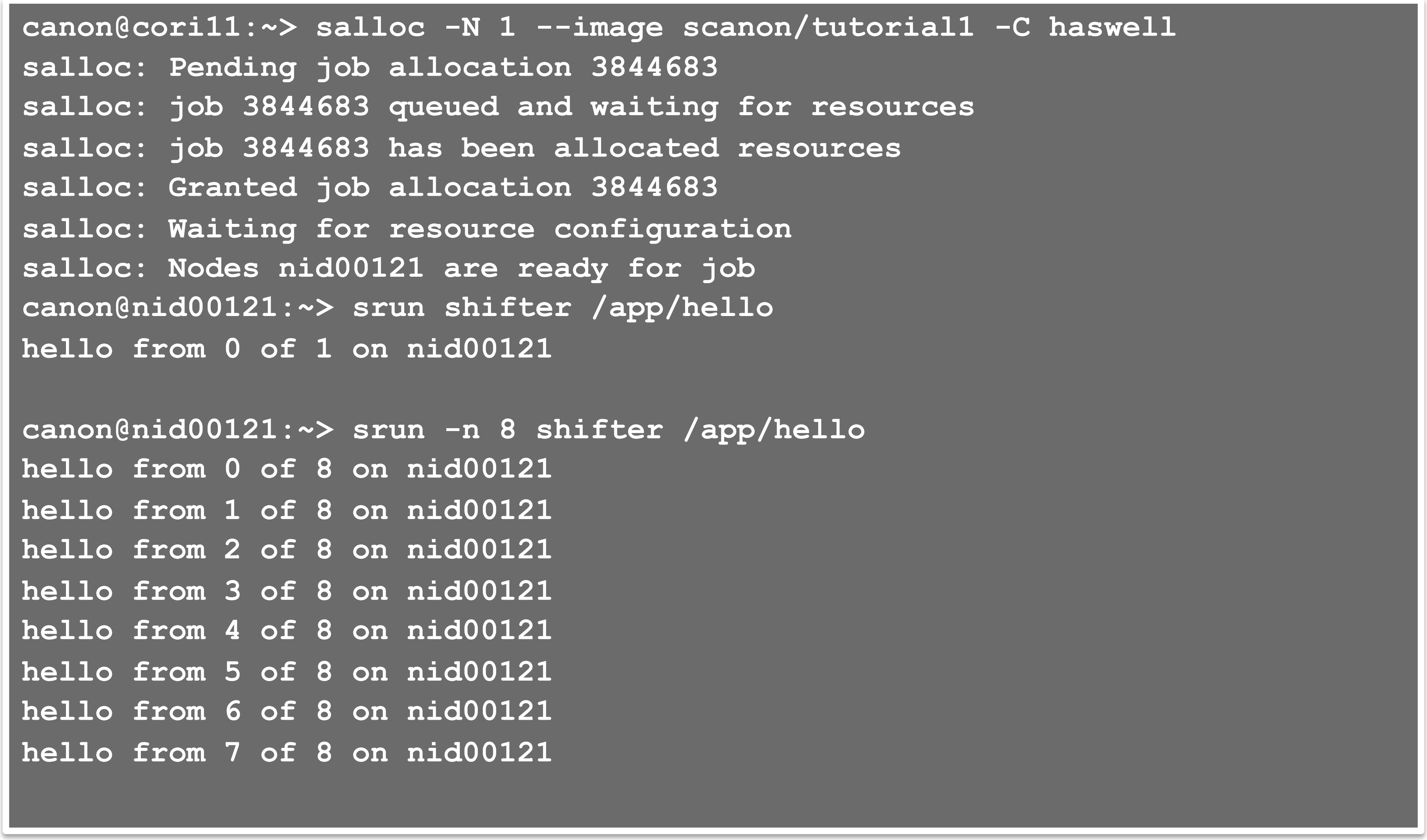
Comments (0)

Pull the image to NERSC and run



```
canon@cori11:~> shifterimg pull scanon/tutorial1
2017-02-23T17:22:38 Pulling Image: docker:scanon/tutorial1, status: READY


canon@cori11:~> shifterimg images|grep scanon/tut
cori        docker     READY    2ab93f3c45   2017-02-23T17:22:46 scanon/
tutorial1:latest
```

Pull the image to NERSC and run



```
canon@cori11:~> salloc -N 1 --image scanon/tutorial1 -C haswell
salloc: Pending job allocation 3844683
salloc: job 3844683 queued and waiting for resources
salloc: job 3844683 has been allocated resources
salloc: Granted job allocation 3844683
salloc: Waiting for resource configuration
salloc: Nodes nid00121 are ready for job
canon@nid00121:~> srun shifter /app/hello
hello from 0 of 1 on nid00121

canon@nid00121:~> srun -n 8 shifter /app/hello
hello from 0 of 8 on nid00121
hello from 1 of 8 on nid00121
hello from 2 of 8 on nid00121
hello from 3 of 8 on nid00121
hello from 4 of 8 on nid00121
hello from 5 of 8 on nid00121
hello from 6 of 8 on nid00121
hello from 7 of 8 on nid00121
```

Outlook

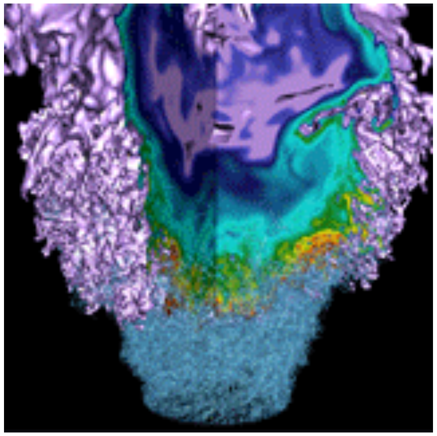
- **Shifter is being successfully used by many users including users from HEP, NP, and BES**
- **Future Shifter plans**
  - Ability to overlay multiple shifter images
  - Private shifter images for groups with access limitations
- **Shifter is an easy way to improve performance and get portability for your science environment**

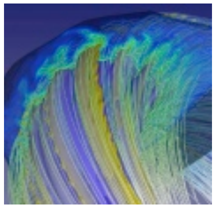
National Energy Research Scientific Computing Center

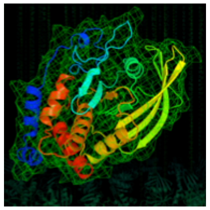
Hello World

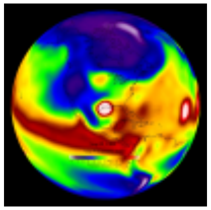
```
DOE6903508:shifter canon$ cat helloworld.c
// Hello World MPI app
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    int size, rank;
    char buffer[1024];

    MPI_Init(&argc, &argv);

    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    gethostname(buffer, 1024);

    printf("hello from %d of %d on %s\n", rank, size, buffer);

    MPI_Barrier(MPI_COMM_WORLD);

    MPI_Finalize();
    return 0;
}
```

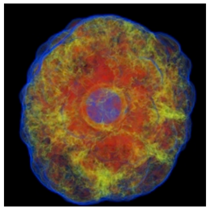
Screen Shots for Backup

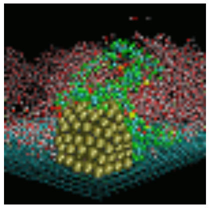












Dockerfile

```
DOE6903508:shifter canon$ cat Dockerfile
# This example makes use of an Ubuntu-based NERSC base image
# that already has MPI built and installed.
#
# This means the you just need to add your app code in and compile it.
#
# To build this example do:
# docker build -t <dockerhubid>/hellompi:latest .
#
# And to test:
# docker run -it --rm <dockerhubid>/hellompi:latest /app/hello

FROM nersc/ubuntu-mpi:14.04

ADD helloworld.c /app/

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

ENV PATH=/usr/bin:/bin:/app:/usr/local/bin
```

Building and Pushing

```
DOE6903508:shifter canon$ docker build -t scanon/hellompi .
Sending build context to Docker daemon 4.096 kB
Step 1 : FROM nersc/ubuntu-mpi:14.04
 ---> a71176e6ce4e
Step 2 : ADD helloworld.c /app/
 ---> 3ebe2000ff1b
Removing intermediate container c88799e39fa2
Step 3 : RUN cd /app && mpicc helloworld.c -o /app/hello
 ---> Running in f3ec8a1c55f3
 ---> 68491ca5b944
Removing intermediate container f3ec8a1c55f3
Step 4 : ENV PATH /usr/bin:/bin:/app:/usr/local/bin
 ---> Running in eef831fe8c61
 ---> 132ba04bff6d
Removing intermediate container eef831fe8c61
Successfully built 132ba04bff6d
DOE6903508:shifter canon$ docker push scanon/hellompi
The push refers to a repository [docker.io/scanon/hellompi]
2c96e3a94e72: Pushed
b0da9b74490f: Pushed
2a291a08dc8c: Mounted from scanon/hello
04cb0f3619cf: Mounted from scanon/hello
c410e650f359: Mounted from scanon/hello
bab10a362750: Mounted from scanon/hello
787a9151f9ae: Mounted from scanon/hello
470641744213: Mounted from scanon/hello
latest: digest: sha256:9225f1ce3fc3f6a644306899019df55e7632bc9a2e2f5a0a46e933b810a32805 size:
1990
```

Approach 1 – In the image

Dockerfile

```
FROM centos:6
## update packages and install dependencies
RUN yum upgrade -y && \
    yum -y install csh tar numpy scipy matplotlib gcc
WORKDIR /
## replace mpi4py with cray-tuned one
ADD optcray_cori.tar /
ADD mpi4py-1.3.1.tar.gz /usr/src
ADD mpi.cfg /usr/src/mpi4py-1.3.1/
RUN cd /usr/src/mpi4py-1.3.1 && \
    chmod -R a+rX /opt/cray && chown -R root:root /opt/cray && \
    python setup.py build && \
    export MPI4PY_LIB=$( rpm -ql $(rpm -qa | grep mpi4py | head -1) | egrep "lib$" ) && \
    export MPI4PY_DIR="${MPI4PY_LIB}/.." && \
    python setup.py install && \
    cd / && rm -rf /usr/src/mpi4py-1.3.1 && \
    echo "/opt/cray/wlm_detect/default/lib64/libwlm_detect.so.0" >>/etc/ld.so.preload && \
    (echo "/opt/cray/mpt/default/gni/mpich2-gnu/48/lib\n/opt/cray/pmi/default/lib64”;\
     echo “/opt/cray/ugni/default/lib64\n/opt/cray/udreg/default/lib64”;\
     echo “/opt/cray/xpmem/default/lib64\n/opt/cray/alps/default/lib64”) \
     >> /etc/ld.so.conf && \
    ldconfig
```

```
> docker build –t scanon/myapp:1.1 .
> docker push scanon/myapp:1.1
```

Approach 2 - Golden Image

Dockerfile

```
FROM registry.services.nersc.gov/cori:latest

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

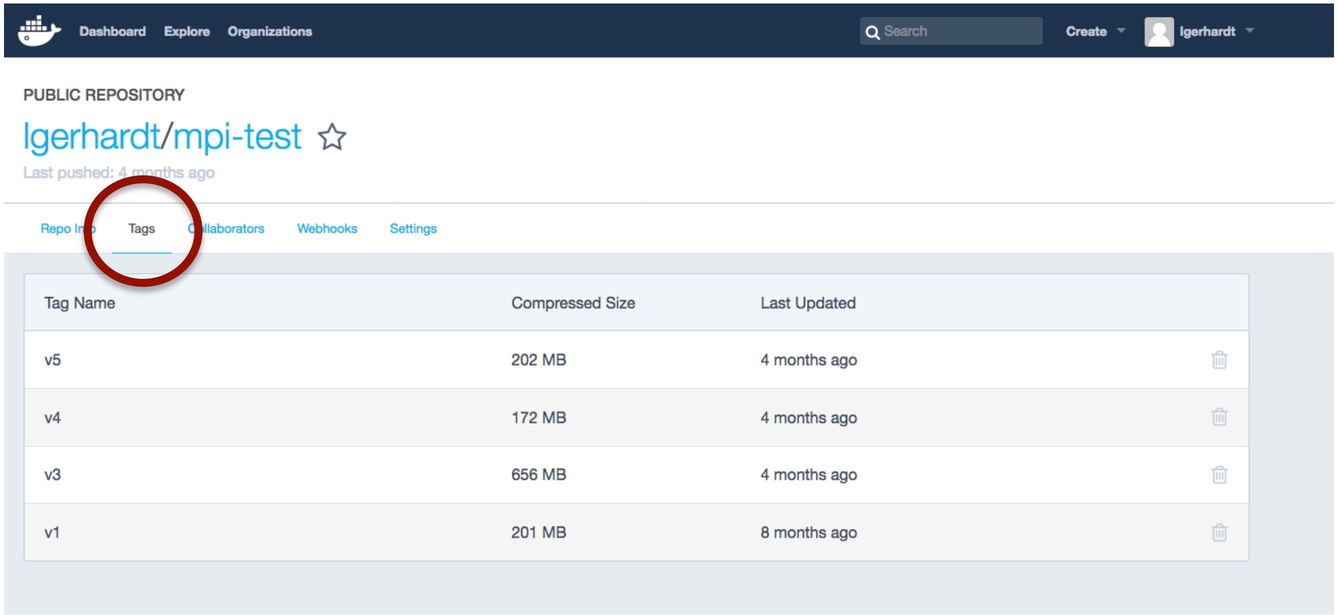
```
> docker build –t scanon/myapp:1.1 .
> docker push scanon/myapp:1.1
```

Pulling Down an Image from Dockerhub

On Cori or Edison

```
shifterimg pull docker:lgerhardt/mpi-test:v5
```

Format is source:image_name: tag



PUBLIC REPOSITORY

lgerhardt/mpi-test ☆

Last pushed: 4 months ago

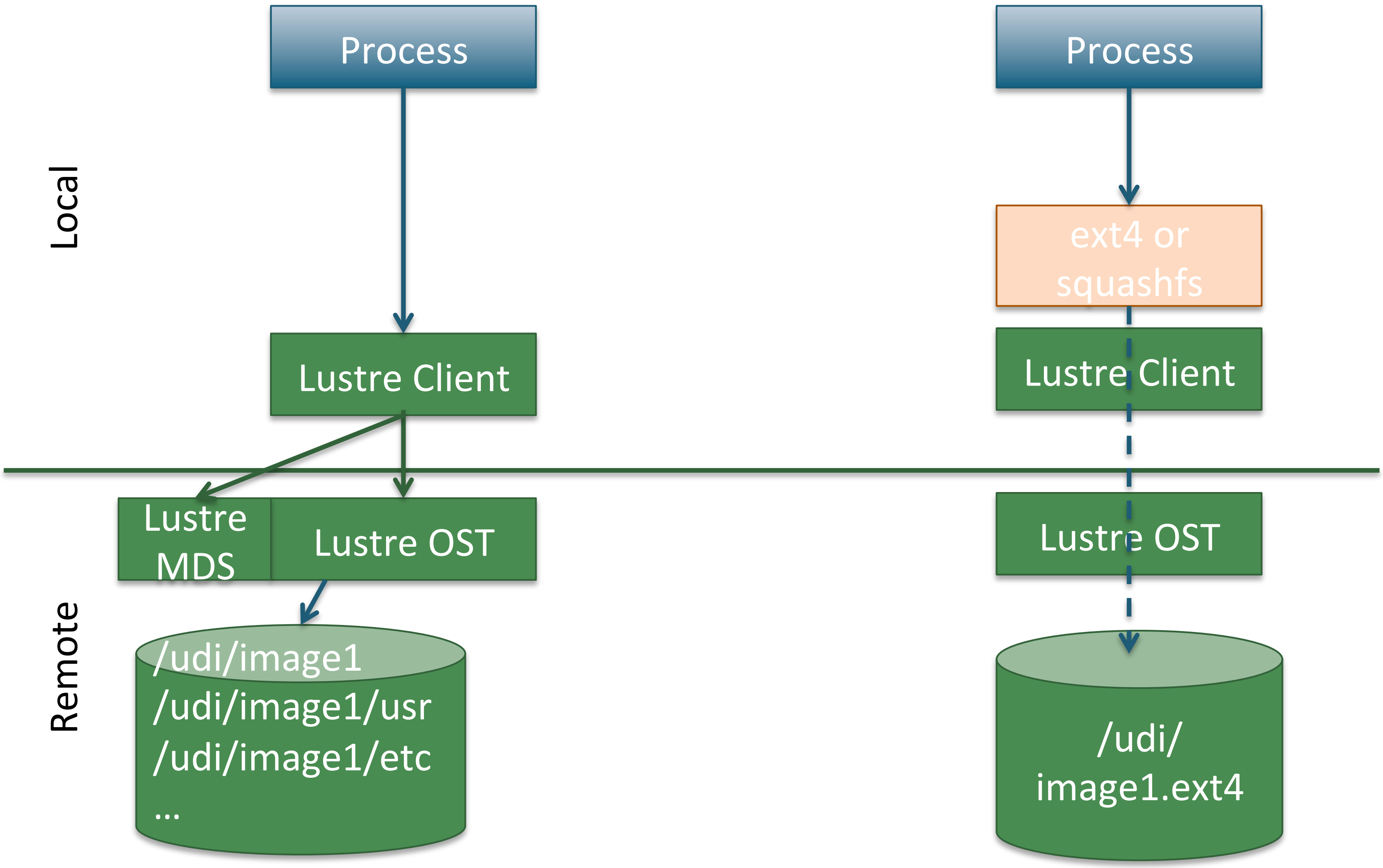
Tag Name	Compressed Size	Last Updated
v5	202 MB	4 months ago
v4	172 MB	4 months ago
v3	656 MB	4 months ago
v1	201 MB	8 months ago

Running A Job in A Shifter Image

```
#!/bin/bash
#SBATCH --image=docker:image_name:latest
#SBATCH --volume="/global/cscratch1/sd/lgerhard:/output”
#SBATCH --volume=“/global/cscratch1/sd/lgerhard/shifter_tmp:/tmp
:perNodeCache=size=200G”



#SBATCH --nodes=1
#SBATCH --partition=regular
#SBATCH -C haswell

srun -n 32 shifter python myPythonScript.py args
```

Many more commands at
http://www.nersc.gov/users/software/using-shifter-and-docker/using-shifter-at-nersc/

File System flow – Traditional vs Shifter


Local
Remote
Process
Lustre Client
Lustre MDS
Lustre OST
/udi/image1
/udi/image1/usr
/udi/image1/etc
…
Process
ext4 or squashfs
Lustre Client
Lustre OST
/udi/
image1.ext4