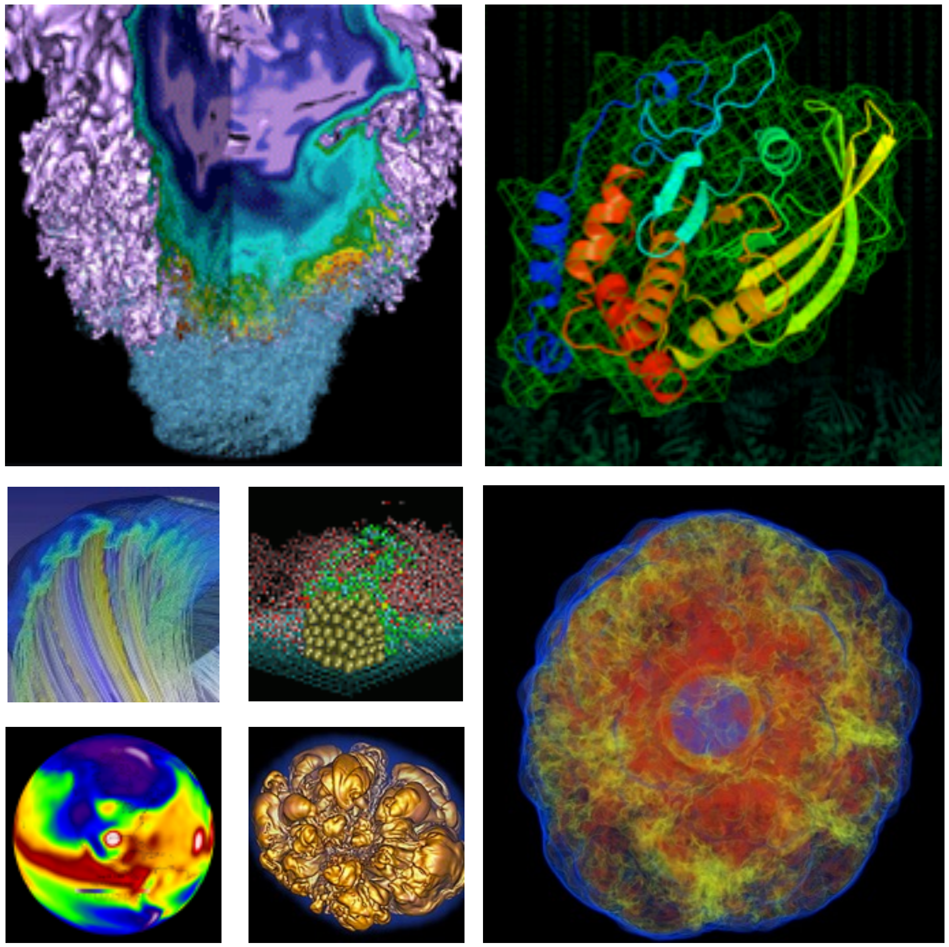
Shifter –Advanced Usage



Shane Canon

NERSC Data and Analytics Services

March 3, 2017

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.
- Basic syntax is:

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

- Shifter places some constraints on what paths can be mapped and where they can be mapped for added security.

Using Volume Mounts

```
canon@nid00275:~> ls $SCRATCH
MyQuota  spark  spark_conf_files

canon@nid00275:~> shifter --image ubuntu --volume $SCRATCH:/data bash
canon@nid00275:~$ ls /data
MyQuota  spark  spark_conf_files
canon@nid00275:~$ exit
```

PerNode Write Cache

- 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 Volume Mounts

```
canon@nid00172:~> shifter --image=ubuntu \
         --volume=$SCRATCH:/scratch:perNodeCache=size=100G bash
canon@nid00172:~$ df -h /scratch
Filesystem      Size  Used Avail Use% Mounted on
/dev/loop1      100G   33M  100G   1% /scratch
canon@nid00172:~$ dd if=/dev/zero bs=1k count=10M of=/scratch/output
10485760+0 records in
10485760+0 records out
10737418240 bytes (11 GB, 10 GiB) copied, 14.1891 s, 757 MB/s
canon@nid00172:~$ ls -lh /scratch/output
-rw-r--r-- 1 canon canon 10G Mar  3 04:01 /scratch/output
canon@nid00172:~$ exit
Exit
canon@nid00172:~> shifter --image=ubuntu \
         --volume=$SCRATCH:/scratch:perNodeCache=size=100G bash
canon@nid00172:~$ ls -l /scratch
total 0
```

Shifter Gotchas

- Containers run as the user, not root
- Images are mounted read-only
- Some volume mount locations are disallowed
- Volumes currently can’t be mounted over each other

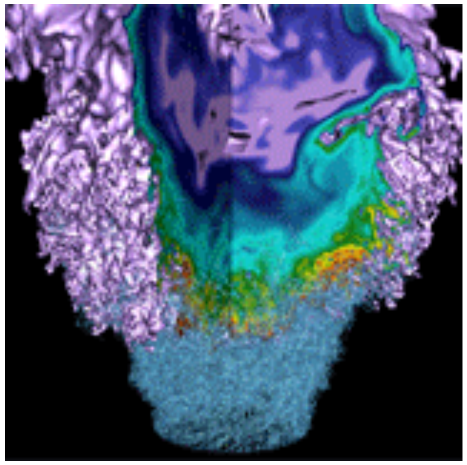
Shifter Gotchas Examples

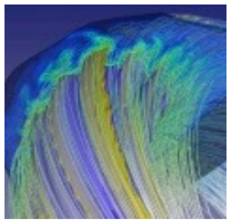
```
canon@nid00173:~> shifter --image=ubuntu  bash
canon@nid00173:~$ ls -ld /var/tmp/
drwxrwxrwt 2 root 0 3 Feb 14 23:29 /var/tmp/
canon@nid00173:~$ touch /var/tmp/blah
touch: cannot touch '/var/tmp/blah': Read-only file system

canon@nid00173:~> shifter --image=ubuntu --volume=$SCRATCH:/opt bash|head -
2
Invalid Volume Map: /scratch1/scratchdirs/canon:/opt, aborting! 1
Failed to parse volume map options

canon@nid00173:~> shifter --image=ubuntu --volume=$SCRATCH:/data --
volume=$SCRATCH/spark:/data/spark2 bash
Mount request path /var/udiMount/data/spark2 not on an approved device for
volume mounts.
FAILED to setup user-requested mounts.
FAILED to setup image.
```

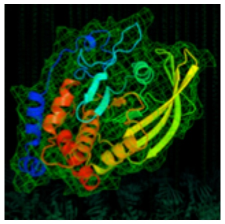
Work Arounds

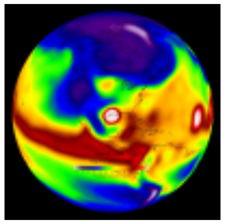
- Make sure critical paths and files are world-readable
- Rsync image contents to a volume mount, then volume mount the copy over the original to work around read-only limitation

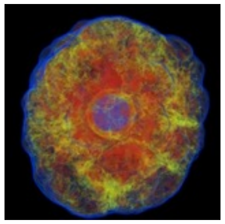
Optimizations

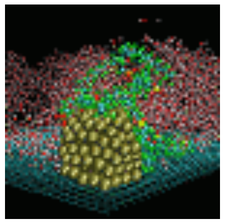












Dockerfile Best Practices

Bad:

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

Good:

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

Dockerfile Best Practices

Bad:

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

Good:

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

Dockerfile Best Practices

Bad:

```
ADD . /src

RUN apt-get update -y && atp-get install gcc

RUN cd /src && make && make install
```

Good:

```
RUN apt-get update -y && apt-get install gcc

ADD . /src

RUN cd /src && make && make install
```

Multi-Stage Builds

- New 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
```

Other Considerations

- Avoid very large images (> ~5 GB)
- Keep data in $SCRATCH 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 stabalizes

National Energy Research Scientific Computing Center