Shifter: Containers in HPC environments

HPC Advisory Council Switzerland
Miguel Gila, CSCS
March 21, 2016
Docker
What is Docker and how does it work?

- “Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.” [*]
What is Docker and how does it work?

- A Docker image is a file that contains a filesystem within it. It can be a full filesystem (i.e. CentOS) or partial (i.e. python3.4-slim). Images are stored either on a public registry (Docker hub) or private.
- Docker leverages the namespaces feature of the kernel to isolate processes.

On its simplest form, Docker basically:

1. Pulls an image to the local system
2. Creates a chrooted environment with the image (=container)
3. Runs our application in the container ('isolated' from the host thanks to kernel namespaces)

However, it can also do other things:
- Isolate network by creating NAT or bridge devices
- Can use a nice GUI
Docker in HPC environments

- Docker is a nice tool, but it’s not built for HPC environments, because:
  - Does not integrate well with workload managers
  - Does not isolate users on shared filesystems
  - Requires running a daemon on all nodes
  - Not designed to run on diskless clients
  - Network is, by default, ‘NATed’
  - Building Docker is done within a Docker container. It can be done outside, but is a complex task (Go language, seriously??)

- But after all, a sysadmin can make anything to work on a cluster, right?
  - We can create (and hopefully maintain) monstrous wrappers to run Docker containers...
What is Shifter and how does it work?

- Shifter is a container-based solution thought from the ground up for HPC environments
- It leverages the current Docker environment and can use Docker images to create containers

Shifter basically

1. Pulls an image to a shared location (/scratch)
2. Creates a loop device with the image (=container)
3. Creates a chrooted environment on the loop device
4. Runs our application in chrooted environment

- Designed to work on HPC clusters and, particularly, on Cray systems
- It is possible to choose which filesystems to expose to the containers
Architecture of Shifter

- Shifter consists on two parts
  - **udiRoot** is responsible for creating the loop devices, doing chroot and cleaning up after the binary execution is done. Workload manager plugins are available. Written in C
  - **imageGateway** is responsible for fetching a Docker image, converting it to a *squashfs* file and transfer it to a shared location on a filesystem. It also keeps track of the images and tells udiRoot which are available. Written in Python
Workflow

1. User creates an image on his/her computer and pushes it to Docker Hub

```
$ docker build .
Uploading context 10240 bytes
Step 1 : FROM busybox
Pulling repository busybox
--- e9aa60c60128MB/2.284 MB (100%) endpoint: https://cdn-registry-1.docker.io/v1/
Step 2 : RUN ls -lh /
--- Running in 9c9e81692ae9
total 24
drwxr-xr-x  2 root root  4.0K Mar 12  2013 bin
drwxr-xr-x  2 root root  4.0K Oct 19 00:19 dev
drwxr-xr-x  2 root root  4.0K Oct 19 00:19 etc
drwxr-xr-x  2 root root  4.0K Nov 15 23:34 lib
lrwxrwxrwx  1 root root 0 Mar 12 2013 lib64 -> lib
dr-xr-xr-x 116 root root 0 Nov 15 23:34 proc
lrwxrwxrwx  1 root root 3 Mar 12 2013 sbin -> bin
dr-xr-xr-x 13 root root 0 Nov 15 23:34 sys
drwxr-xr-x  2 root root 4.0K Mar 12 2013 tmp
drwxr-xr-x  2 root root 4.0K Nov 15 23:34 usr
--- Running in 02071fceb21b
Step 3 : CMD echo Hello world
--- Running in f52f38b7823e
Successfully built f52f38b7823e
Removing intermediate container 9c9e81692ae9
Removing intermediate container 02071fceb21b
```

```
$ docker push miguelgila/wlcg_wn:20161212
```

Or private registry
Workflow

1. User creates an image on his/her computer and pushes it to Docker Hub
2. User tells imageGateway to pull his image and make it available

```bash
$ ssh santis01
$ module load shifter
$ shifterimg pull docker:ubuntu:14.04
$ sleep 300 # (:-)
$ shifterimg images
```

Docker image

Or private registry
Workflow

1. User creates an image on his/her computer and pushes it to Docker Hub
2. User tells imageGateway to pull his image and make it available
3. User runs SBATCH and prepends shifter to his/her executable

```
#!/bin/bash
#SBATCH --job-name="shifter_osversion"
#SBATCH --nodes=1
#SBATCH --time=00:30:00
#SBATCH --exclusive
#SBATCH --output=/users/miguelgi/jobs/out/shifter_hostname.stdout.log.%j
#SBATCH --error=/users/miguelgi/jobs/out/shifter_hostname.stdout.log.%j
#SBATCH --image=miguelgila/wlcg_wn:20151218
#SBATCH --imagevolume=/users:/users
#SBATCH --imagevolume=/scratch/santis:/scratch/santis
module load slurm
module load shifter/15.12.0
shifter --volume=/scratch/santis:/scratch/santis --volume=/users:/users
cat /etc/redhat-release
```
Use cases
### Full OS containers

- Cray compute nodes run CLE (a version of SLES 11)
- With Shifter it is possible to run applications built for specific OS:

<table>
<thead>
<tr>
<th>CentOS 6 – non-interactive session</th>
</tr>
</thead>
</table>
| [miguelgi@santis01]@[11:34:45]-[/~examples]:(~$ salloc -t 00:15:00 -N1 --image=docker:centos:6.7  
salloc: Granted job allocation 16313  
salloc: Waiting for resource configuration  
salloc: Nodes nid00012 are ready for job  
[miguelgi@santis01]@[11:34:50]-[/~examples]:(~$ srun shifter cat /etc/redhat-release  
CentOS release 6.7 (Final)  
[miguelgi@santis01]@[11:35:03]-[/~examples]:(~$ srun shifter yum --version | head -4  
3.2.29  
Installed: rpm-4.8.0-47.el6.x86_64 at 2015-08-19 18:25  
Committed: Lubos Kardos <lkardos@redhat.com> at 2015-06-15  
[miguelgi@santis01]@[10:52:40]-[/~examples]:(~$ salloc -t 00:15:00 -N1 --image=docker:debian:7.9  
salloc: Granted job allocation 16294  
salloc: Waiting for resource configuration  
salloc: Nodes nid00012 are ready for job  
[miguelgi@nid00012]@[10:58:29]-[/~examples]:(~$ srun --pty shifter /bin/bash  
[miguelgi@nid00012]@[10:58:31]-[/~examples]:(~$ cat /etc/debian_version  
7.9  
[miguelgi@nid00012]@[10:58:33]-[/~examples]:(~$ uname -a  
Linux nid00012 3.0.101-0.46.1.1_0.0502.8814-cray_ari_c_#1 SMP Tue Aug 25 21:41:26 UTC 2015 x86_64 GNU/Linux  
[miguelgi@nid00012]@[10:59:46]-[/~examples]:(~$ apt-get --version | head -n1  
apt 0.9.7.9 for amd64 compiled on Oct 17 2014 09:15:56 |}

<table>
<thead>
<tr>
<th>Debian 7.9 – interactive session</th>
</tr>
</thead>
</table>
| [miguelgi@nid00012]@[11:34:50]-[/~examples]:(~$ srun --pty shifter /bin/bash  
[miguelgi@nid00012]@[11:35:31]-[/~examples]:(~$ cat /etc/debian_version  
7.9  
[miguelgi@nid00012]@[10:58:33]-[/~examples]:(~$ uname -a  
Linux nid00012 3.0.101-0.46.1.1_0.0502.8814-cray_ari_c_#1 SMP Tue Aug 25 21:41:26 UTC 2015 x86_64 GNU/Linux  
[miguelgi@nid00012]@[10:59:46]-[/~examples]:(~$ apt-get --version | head -n1  
apt 0.9.7.9 for amd64 compiled on Oct 17 2014 09:15:56 |}
Application containers: Python/Ruby

- Can run application specific containers:

**Python 3.5.1 – interactive session**

[miguelgi@santis01]-[09:57:23]-[-]:$ salloc -t 01:00:00 -n1 --image=docker:python:3.5.1
salloc: Granted job allocation 15274
salloc: Waiting for resource configuration
salloc: Nodes nid00012 are ready for job

[miguelgi@santis01]-[10:01:04]-[-]:$ python -V
Python 3.5.1

[miguelgi@santis01]-[09:01:08][-]:$ which python
/usr/local/bin/python

**Ruby 2.1.8 – interactive session**

[miguelgi@santis01]-[09:57:23]-[-]:$ salloc -t 01:00:00 -n1 --image=docker:ruby:2.1.8
salloc: Granted job allocation 15280
salloc: Waiting for resource configuration
salloc: Nodes nid00012 are ready for job

[miguelgi@santis01]-[10:22:14][-]:$ ruby -v
ruby 2.1.8p440 (2015-12-16 revision 53160) [x86_64-linux]

[miguelgi@santis01]-[09:22:28][-]:$ ruby -v
ruby 2.1.8

[miguelgi@santis01]-[10:22:05][-]:$ srun shifter ./myscript.py
Ruby 2.1.8

[miguelgi@santis01]-[10:22:08][-]:$ cat myscript.py
#!/usr/bin/env python
import platform
print(platform.python_version())

**Python 3.2 – non-interactive session**

[miguelgi@santis01]-[10:16:35][-]:$ salloc -t 01:00:00 -n1 --image=docker:python:3.2-slim
salloc: Granted job allocation 15278
salloc: Waiting for resource configuration
salloc: Nodes nid00012 are ready for job

[miguelgi@santis01]-[10:19:34][-]:$ srun shifter ./myscript.py
3.2.6

**Ruby 2.1.8 – non-interactive session**

[miguelgi@santis01]-[10:22:05][-]:$ srun shifter ./myscript.rb
Ruby 2.1.8

[miguelgi@santis01]-[10:22:08][-]:$ cat myscript.rb
#!/usr/bin/env ruby
puts RUBY_VERSION

[myscript.py]
$ cat myscript.py
#!/usr/bin/env python
import platform
print(platform.python_version())

[myscript.rb]
$ cat myscript.rb
#!/usr/bin/env ruby
puts RUBY_VERSION
Multi-node containers

- It is possible to run the same container across multiple nodes:

```
[miguelgi@santis01]-[10:41:21]-[~/examples]$: srun shifter ./hostname.py
3.2.6
nid00013
3.2.6
nid00014
```

- Working on getting MPI across nodes to function
- Working on getting GPUs to be accessible to containers with good results so far (native performance!)
Practical use case: WLCG Swiss Tier-2

- CSCS operates the cluster Phoenix on behalf of CHIPP, the Swiss Institute of Particle Physics

- Phoenix runs Tier-2 jobs for ATLAS, CMS and LHCb, 3 experiments of the LHC at CERN and part of WLCG (Worldwide LHC Computing Grid)

- WLCG jobs need and expect RHEL-compatible OS. All software is precompiled and exposed in a cvmfs\[*\] filesystem

- But Cray XC compute nodes run CLE, a modified version of SLES 11 SP3

- So, how do we get these jobs to run on a Cray?

\[*\] https://cernvm.cern.ch/portal/filesystem
Practical use case: WLCG Swiss Tier-2

- Using Shifter, we are able to run unmodified ATLAS, CMS and LHCb production jobs on a Cray XC40 TDS
- Jobs see standard CentOS 6 containers
- Nodes are shared: multiple single-core and multi-core jobs, from different experiments, can run on the same compute node
- Job efficiency is comparable in both systems

<table>
<thead>
<tr>
<th>JOBID</th>
<th>USER</th>
<th>ACCOUNT</th>
<th>NAME</th>
<th>NODELIST</th>
<th>ST</th>
<th>REASON</th>
<th>START_TIME</th>
<th>END_TIME</th>
<th>TIME_LEFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>82471</td>
<td>atlasprd</td>
<td>atlas</td>
<td>a53eb5f8_34f0</td>
<td>nid00043</td>
<td>R</td>
<td>None</td>
<td>15:03:33</td>
<td>23:54:18</td>
<td>1 8</td>
</tr>
<tr>
<td>82476</td>
<td>cms04</td>
<td>cms</td>
<td>gridjob</td>
<td>nid00043</td>
<td>R</td>
<td>None</td>
<td>15:08:39</td>
<td>11:59:24</td>
<td>1 2</td>
</tr>
<tr>
<td>82451</td>
<td>lhcbpit</td>
<td>lhcb</td>
<td>gridjob</td>
<td>nid00043</td>
<td>R</td>
<td>None</td>
<td>15:00:10</td>
<td>11:50:55</td>
<td>1 2</td>
</tr>
<tr>
<td>82447</td>
<td>lhcbpit</td>
<td>lhcb</td>
<td>gridjob</td>
<td>nid00043</td>
<td>R</td>
<td>None</td>
<td>14:59:31</td>
<td>11:50:16</td>
<td>1 2</td>
</tr>
<tr>
<td>82448</td>
<td>lhcbpit</td>
<td>lhcb</td>
<td>gridjob</td>
<td>nid00043</td>
<td>R</td>
<td>None</td>
<td>14:59:31</td>
<td>11:50:16</td>
<td>1 2</td>
</tr>
<tr>
<td>82449</td>
<td>lhcbpit</td>
<td>lhcb</td>
<td>gridjob</td>
<td>nid00043</td>
<td>R</td>
<td>None</td>
<td>14:59:31</td>
<td>11:50:16</td>
<td>1 2</td>
</tr>
<tr>
<td>82450</td>
<td>lhcbpit</td>
<td>lhcb</td>
<td>gridjob</td>
<td>nid00043</td>
<td>R</td>
<td>None</td>
<td>14:59:31</td>
<td>11:50:16</td>
<td>1 2</td>
</tr>
<tr>
<td>82446</td>
<td>lhcbpit</td>
<td>lhcb</td>
<td>gridjob</td>
<td>nid00043</td>
<td>R</td>
<td>None</td>
<td>14:48:10</td>
<td>11:38:46</td>
<td>1 2</td>
</tr>
<tr>
<td>82444</td>
<td>lhcbpit</td>
<td>lhcb</td>
<td>gridjob</td>
<td>nid00043</td>
<td>R</td>
<td>None</td>
<td>14:48:10</td>
<td>11:38:46</td>
<td>1 2</td>
</tr>
</tbody>
</table>
Wrap-up
Docker vs. Shifter

- Can isolate network by creating NAT or bridge devices. What about IB?
- Users can write as root on exposed RW filesystems
- Needs a local daemon running
- Isn’t SLURM-friendly
- Can run on multiple nodes with own tool (Swarm)
- Can use GPUs?
- MPI?
- Can run images on private registry

Docker

- It shows all /dev, /sys and /proc to the container environment. Easy
- Users can write as their $USER on any exposed RW filesystem
- Does not need a local daemon on CN
- Is SLURM-friendly (SPANK plugin)
- Can run on multiple nodes with WLM integration
- Can use GPUs. Working on it!
- MPI on its way!
- Can run images on private registry
Conclusion

- Shifter works very well on our HPC environment
- It’s being constantly developed and new features are appearing on a weekly basis
- It’s open source and developed by the HPC community

- It needs some additional work to cover basic HPC use cases (MPI)
- Interacting with some parts of Shifter is not very user-friendly:
  - The process of pulling images is easy, but has no visual feedback
  - At times, error messages are difficult to understand
- No ACLs yet
Reference links

- NERSC info: [http://www.nersc.gov/research-and-development/user-defined-images/](http://www.nersc.gov/research-and-development/user-defined-images/)

- The code: [https://github.com/NERSC/shifter](https://github.com/NERSC/shifter)
Questions?
Thank you for your attention.