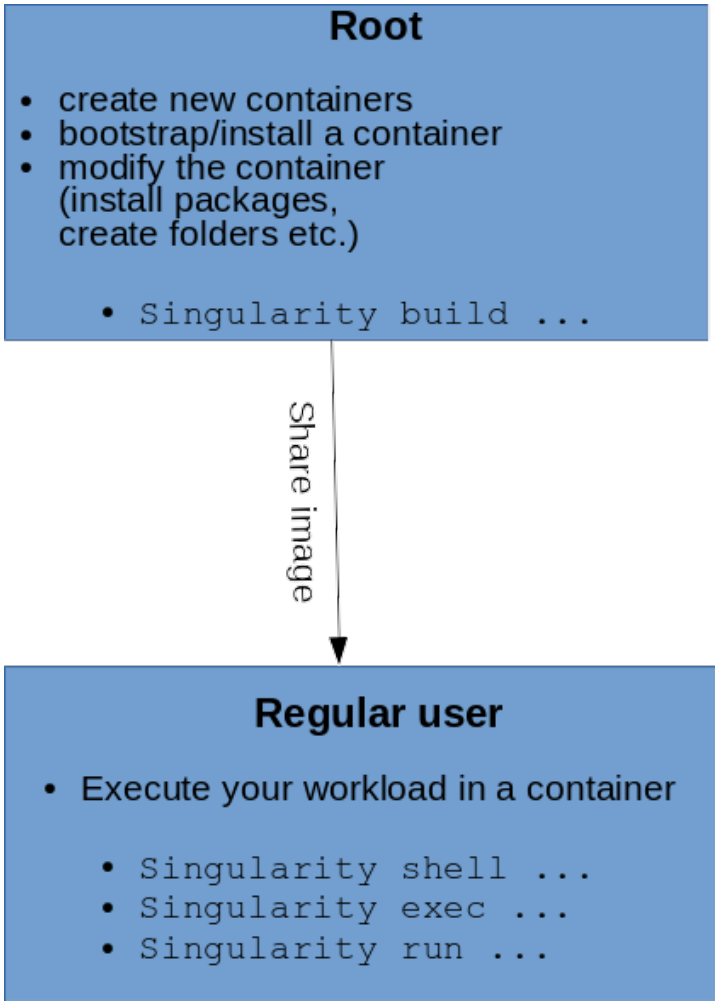
Singularity

- First public release in April 2016
- Current version 2.4 released in October – major changes in 2.3
- Support for native GPUs, Infiniband, MPI
- Network namespace isolation
- In SquashFS format (immutable, compressed and read-only Linux file system)
- To create a writable image, root privileges are required.

Singularity in HPC

- **Permissions:** user inside == user outside, if you want to have root permissions, you need to be root on the host system
- No root container daemon
- **Portability**: create an image on your desktop and use it on the cluster - customization of the runtime environment without administrator privileges on the cluster
- **IO** is passed directly through the container – **bound directories** from the host
- no impact on performance
- **limitations**: devices, drivers, stability of overlayfs, to bind directories from host, they need to exist in the container

Privileges


Root
create new containers
bootstrap/install a container
modify the container
(install packages,
create folders etc.)
Singularity build ...
Share image
Regular user
Execute your workload in a container
Singularity shell ...
Singularity exec ...
Singularity run ...

Build an image

- From Docker hub
- From Singularity hub
- From a definition file

Build image from Docker hub

- Docker hub repositories: https://hub.docker.com/explore/

singularity build ubuntu.img docker://ubuntu:latest

```
Docker image path: index.docker.io/library/ubuntu:latest
Cache folder set to /home/barbarak/.singularity/docker
[5/5] |==================================| 100.0%
Importing: base Singularity environment
Importing:
/home/barbarak/.singularity/docker/sha256:660c48dd555dcbfdfe19c80a30f557ac57a15f595250e67bfad1e5663
c1725bb.tar.gz
..
WARNING: Building container as an unprivileged user. If you run this container as root
WARNING: it may be missing some functionality.
Building Singularity image...
Singularity container built: ubuntu.img
Cleaning up...
```

Build image from Singularity hub

- Singularity hub: https://singularity-hub.org/collections
- For instance if I need Gromacs:

```
$ singularity build gromacs.img shub://michael-tn/gromacs:cent7
Cache folder set to /home/barbarak/.singularity/shub
Progress |==================================| 100.0%
Building from local image: /home/barbarak/.singularity/shub/michael-tn-gromacs-master-cent7.simg
WARNING: Building container as an unprivileged user. If you run this container as root
WARNING: it may be missing some functionality.
Building Singularity image...
Singularity container built: gromacs.img
Cleaning up...
```

Build image from definition file

singularity build centos6-minimal.img centos6.def

```
BootStrap:yum
OSVersion: 6.9
MirrorURL: http://ftp.arnes.si/mirrors/centos.org/6.9/os/x86_64/
UpdateURL: http://ftp.arnes.si/mirrors/centos.org/6.9/os/x86_64/
Include: yum git

%setup

%runscript
    echo "Running the container..."

%post
    echo "Installing the packages inside the container"
     rpm -vv --rebuilddb
    echo "Installing Development tools"
     yum -y groupinstall "Development Tools"

    echo "Installing basic packages"
    yum -y install vim-enhanced man-db wget ntp gfal2-all xrootd-client autofs nfs-utils git perl perl-Data-
Dumper automake autoconf libtool gcc gcc-c++ glibc flex make autofs
```

Writable images

- You can build a writable image, but root privileges are required

```
sudo singularity build --writable centos.img docker://centos:latest
```

- Existing images can also be modified, by running

```
sudo singularity shell --writable centos.img
```

- You can also pull an image from a public hub an modify it (like in the previous step)

```
sudo singularity pull docker://centos:latest
```

```
sudo singularity shell centos-7.4.img
```

Sandbox images

- Also “sandbox images” are an option
- If you want to create a container inside a writable directory

```
sudo singularity build --sandbox folder/ docker://ubuntu:latest
```

- The specified directory operates like a container in an image file
- You can make changes in a container, which will persist until the container is run

How we use it in grid?

- As a replacement for RTE-s
- User can run it himself:
  - using a publicly available image (e.g. Docker hub)
  - using an image on the shared storage (e.g. NFS shares, CVMFS)
  - using his own image, placed on dCache or some other external location

RTEs

- we modify the submit-SLURM-job, so that singularity script is executed
- in singularity script we specify the conditions:

```
if RTE XY is chosen

then execute singularity container xy, binding
directories A, B, C and D
```

Grid users

- Run singularity in their shell script, using a publicly available image or an image on NFS

```
$ singularity exec docker://python:latest /usr/local/bin/python hello.py

library/python:latest

Downloadinglayer: sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4

Downloadinglayer: sha256:e41da2f0bac3da1769ecdac8b0f5df53c1db38603e39b9e261cafd10caf904de

…

…

Hello World: The Python version is 3.6.0
```

Now deep learning frameworks are no longer sysadmin's pain

```
$ singularity exec --nv docker://tensorflow/tensorflow:latest-gpu python
/tmp/tensorflow-mnist.py
Docker image path: index.docker.io/tensorflow/tensorflow:latest-gpu
Cache folder set to /home/nsc0001/.singularity/docker
Creating container runtime...
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting MNIST_data/train-images-idx3-ubyte.gz
2017-11-29 08:17:07.316974: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Found device 0 with properties:
name: Tesla K40m major: 3 minor: 5 memoryClockRate(GHz): 0.745
pciBusID: 0000:02:00.0
totalMemory: 11.17GiB freeMemory: 11.09GiB
2017-11-29 08:17:07.603517: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Found device 1 with properties:
name: Tesla K40m major: 3 minor: 5 memoryClockRate(GHz): 0.745
pciBusID: 0000:82:00.0
totalMemory: 11.17GiB freeMemory: 11.09GiB
2017-11-29 08:17:07.604677: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1045] Device peer to peer matrix
2017-11-29 08:17:07.604766: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1051] DMA: 0 1
2017-11-29 08:17:07.604790: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1061] 0:   Y N
2017-11-29 08:17:07.604806: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1061] 1:   N Y
2017-11-29 08:17:07.604829: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Creating TensorFlow device (/device:GPU:0) -> (device:
0, name: Tesla K40m, pci bus id: 0000:02:00.0, compute capability: 3.5)
2017-11-29 08:17:07.604869: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Creating TensorFlow device (/device:GPU:1) -> (device:
1, name: Tesla K40m, pci bus id: 0000:82:00.0, compute capability: 3.5)
Device mapping:
/job:localhost/replica:0/task:0/device:GPU:0 -> device: 0, name: Tesla K40m, pci bus id: 0000:02:00.0, compute capability: 3.5
/job:localhost/replica:0/task:0/device:GPU:1 -> device: 1, name: Tesla K40m, pci bus id: 0000:82:00.0, compute capability: 3.5
2017-11-29 08:17:07.994373: I tensorflow/core/common_runtime/direct_session.cc:299] Device mapping:
/job:localhost/replica:0/task:0/device:GPU:0 -> device: 0, name: Tesla K40m, pci bus id: 0000:02:00.0, compute capability: 3.5
/job:localhost/replica:0/task:0/device:GPU:1 -> device: 1, name: Tesla K40m, pci bus id: 0000:82:00.0, compute capability: 3.5
step 0, training accuracy 0
step 100, training accuracy 0.86
step 200, training accuracy 0.9
step 300, training accuracy 0.88
step 400, training accuracy 0.86
step 500, training accuracy 0.92
step 600, training accuracy 0.96
step 700, training accuracy 0.96
step 800, training accuracy 0.98
step 900, training accuracy 0.98
step 1000, training accuracy 0.94
step 1100, training accuracy 0.98
step 1200, training accuracy 0.96
step 1300, training accuracy 0.94
step 1400, training accuracy 1
```

GPU and Singularity

- Before version 2.3, GPUs could be used only if the CUDA toolkit and NVIDIA drivers were installed in the container (problem: different versions of OS-es, compilers etc.)
- Now native support for GPUs, using --nv option, only CUDA has to be installed in the container
- This allows portability of the same container (we tested the same one on 2 different grid clusters and in Azure) and it worked

Infiniband and Singularity

- IB libraries not seen in the container by default
- Some modifications required on the host:
  add to /etc/singularity/init

```
for i in `ldconfig -p | grep -E "/libib|/libgpfs|/libnuma|/libmlx|/libnl"`; do
if [ -f "$i" ]; then
message 2 "Found a library: $i\n"
if [ -z "${SINGULARITY_CONTAINLIBS:-}" ]; then
SINGULARITY_CONTAINLIBS="$i"
else
SINGULARITY_CONTAINLIBS="$SINGULARITY_CONTAINLIBS,$i"
fi
fi
done
if [ -z "${SINGULARITY_CONTAINLIBS:-}" ]; then
message WARN  "Could not find any IB-related libraries on this host!\n";
else
export SINGULARITY_CONTAINLIBS
fi
```

- Then install OpenMPI and IB-related packages in the container
- Also OpenMPI from the host machine can be used

OpenMPI and Singularity

- No bandwidth and latency differences when using OpenMPI from host or container – but be careful to link MPI in container to IB libraries (otherwise only 60% of total bandwidth seen)
- Example:

```
 # uses MPI on the host machine
mpirun singularity exec -B /home/user image.img hellompi


# uses MPI in the container
singularity exec -B /net/jost/home test.img mpirun hellompi
```

Singularity plugin for SLURM

- https://github.com/singularityware/singularity/tree/master/src/slurm
- Added to /etc/slurm/plugstack.conf:
  ```
  required /path/to/singularity.so
  ```
- eg. for CMS:
  ```
  #SBATCH --singularity-image=/cvmfs/cms.cern.ch/rootfs/x86_64/centos7/latest
  ```

IPv6-only worker nodes and Singularity

- We only have IPv6 worker nodes
- To access the sites via IPv4, we use HTTP proxies
- Singularity by default doesn't support them
- In order to use the proxy, modify /usr/libexec/singularity/python/base.py and add:

```
os.environ['http_proxy']=”http://host:port”
os.environ['https_proxy']=”http://host:port”
```