Singularity in batch jobs

All the CNAF computing servers support containerization through Singularity [24] . Singularity is a containerization tool for running software in a reproducible way on various different platforms. With the advent of new operating systems, programming languages, and libraries, containers offer a sustainable solution for running old software. The software is shipped in so-called *image*, i.e. a file or folder containing a minimal operating system, the application to run and all its dependencies.
This section of the Tier-1 User Guide is intended to drive the user towards a transition from a native application workflow to one based on containers.
Singularity supports several image formats:

- Singularity *.img* files
- Singularity images in registry via the *shub://* protocol
- Docker images in registry via the *docker://* protocol
- a tar archive, eventually bzipped or gzipped
- a folder

Obtain images

Official images for official software may have already been prepared by the software group within your experiment and be available through a shared filesystem (such as CVMFS), SingularityHub or other supported repository. Please check with your software manager about the support to Singularity.

Create a new image using a recipe (expert users)

A singularity recipe is a text file that contains all the instructions and configurations needed to build an image. It is organized in a header and a number of optional sections. The header is composed by a list of configuration settings in the form of "*keyword*: *value*" describing the system to use as base for the new image. Sections are identified by the *%* sign followed by a keyword. For a detailed list of all possible sections refer to [25].
An example of recipe file follows:

```
## Assume the file is called Recipe.txt and the final image is named name.img

# Headers
BootStrap: docker
From: cern/slc6-base

## Help section
%help
This text will be available by running

singularity help name.img

%labels
## Metadata to store (available through "singularity inspect name.img")
AUTHOR Carmelo Pellegrino
VERSION 1.0

## Environment variables setup
%environment
export LD_LIBRARY_PATH=/usr/local/lib/:$LD_LIBRARY_PATH

## Files to be copied in image from host system
%files
# source (in host system) [destination (in container, default "/")]
./code/ /

## Commands to execute in container during image building, after bootstrap
%post
yum -y update && yum -y upgrade && \
yum -y install --setopt=tsflags=nodocs \
devtoolset-6 compat-gcc-34-g77 \
cernlib-g77-devel cernlib-g77-utils

## Commands to be executed in default container instantiation
%runscript
exec /bin/bash
```

Build it with:

```
sudo singularity build name.img Recipe.txt
```

NOTE: since building a new image requires root access on the host system, it will not be possible on CNAF computing resources.

Run software

Singularity allows to run image-shipped software by simply prepending the singularity exec image.img text on the command line. The current working directory (CWD) is maintained and non-system files are available, so that, for example, the output of the ls command run in a container is not very different from the same command run natively.

Suppose the g77 compiler version 3.4 is contained in the name.img image. Example commands and their output follow:

```
$ cat hw.f
      program hello_world
      implicit none
      write ( *, * ) 'Hello, world!'
      stop
      end
 $ singularity exec name.img g77 --version # i.e.: "g77 --version" runs in a container
GNU Fortran (GCC) 3.4.6 20060404 (Red Hat 3.4.6-19.el6)
Copyright (C) 2006 Free Software Foundation, Inc.
GNU Fortran comes with NO WARRANTY, to the extent permitted by law.
You may redistribute copies of GNU Fortran
under the terms of the GNU General Public License.
For more information about these matters, see the file named COPYING
or type the command `info -f g77 Copying'.
 $ singularity exec name.img g77 hw.f -o hw
 $ singularity exec name.img ./hw
  Hello, world!
```

If a particular external folder is not available in container, for example *exp_software* or *cvmfs*, it can be forcedly bound by means of the `-B orig:dest` command line option. Many folders can also be bound at the same time:

```
$ singularity exec -B /cvmfs/:/cvmfs/ -B /opt/exp_software/:/opt/exp_software/ name.img ./hw
```

How to run *Geant4* using Singularity

Geant4 needs old verion of “`liblzma`” and “`libpcre`” libraries related to Scien-tific Linux 6, for this reason an SL6 environment (using Singularity container) is therefore required to compile/execute the software from there.

Recompile Geant4 inside a container instantiated from a Singularity image with SL6 (usually provided by CNAF User Support), so that the libraries used are actually those that require *Geant4*. To do this, just start a Singularity container with the following command:

```
$ singularity exec /opt/exp_software/cupid/UI_SL6.img /bin/bash
Singularity>
```

then in the folder where the *Geant* sources are present, they can be recompiled.

If it is necessary to be able to access a host operating system path, simply use the ``-B`" (binding path) option specifying the host path separated from the mountpoint in the container using the colons. For example, to access `/opt/exp_software/cupid`, change the previous command to:

```
$ singularity exec -B /opt/exp_software:/opt/exp_software /opt/exp_software/cupid/UI_SL6.img /bin/bash
Singularity>
```

Once *Geant* has been recompiled, to submit job, the executable must be written in order to launch the commands related *Geant* within the Singularity container.

To perform, for example, the “`cuspide`” application, a line like the following must be inserted in the script:

```
singularity exec --no-home -B /opt/exp_software:/opt/exp_software /opt/exp_software/cupid/UI_SL6.img /opt
/exp_software/cupid/geant4/bin/10.04.p02/cuspide
```

where the ``--no-home`" option causes Singularity to avoid mounting the user's home, since the job, not having the permissions to do so, would fail if it did.