The University of Queensland

Research Computing Centre

Containers User Guide

Document Status

Last updated on December 21 2020 by David.Green @ uq.edu.au

This guide may contain outdated information or may not conform to our current approach.
Please apply these instructions with some care and intelligence.

Background

The Problem

We often get requests to install software on the HPC. (that is not the problem!!)
Sometimes it just is not technically possible to install the software requested.

Our "vintage" operating system on the HPC provides a stable manageable environment but it does cause problems sometimes.
More information is provided in the Further Reading Section below.

Enter software containers ...

Software containers provide a flexible mechanism to deploy software within the seemingly rigid HPC environment.

Container based applications are intended to be portable, so that it becomes independent of the host system libraries.
Computational results should then be repeatable despite differences in the underlying OS and OS library versions.

Our choice is a container framework called Singularity

Why not use Docker ?

Docker containers run as root and require extensive root privileges.

Latest versions of Docker offer a separate user account, but since the container itself is running as root we still have security issues.

Features of Singularity and Singularity Containers

- Singularity is a user space container framework.
- Containers are read-only images that may be shared by multiple users and tasks.
- Containers run as a user process with broad access to the user's files.

- Containers can be used to run their pre-configured command sequence. The container itself can be run as a kind of binary.
- Containers can be used to execute an arbitrary command within the container.
- Containers can be used to start an interactive shell within the container.

- Creation and modification of containers on the HPC requires root privileges.
- Users may create their own containers on any linux machine (or VM) where they have root access and bring it to the UQ HPC
- Containers can be created from a definition file (a programmatic recipe card!).
- Definition files can specify one of a number of build mechanisms.

- Singularity is a project from the Lawrence Berkeley Laboratories in the USA.
- Singularity has a web site
- Singularity has a peer reviewed publication with DOI (full bibliographic details appear below)
- Singularity is used on UCSD's Comet HPC system.

Using Singularity on UQ HPC

We are in the process of re-organising the access to software that is provided using containers.

The software that has been made available using Software Containers is listed in a special section of the output of the command **`module avail`**.

```
--------------------------- /sw/Modules/ContainedApps ---------------------------
   R/3.6.0+tidyverse+rstan+rtdists         glibc/2.23               (D)
   R/3.6.0-container                       graftm/0.9.4
   R/3.6.1+CrocSim                         grass/7.0.3
   R/3.6.1+exSTRa                          neurodebian/0.37.2
   R/3.6.1+Monocle3+Giotto                 neurodebian/0.37.6
   R/3.6.1+Monocle3                        neurodebian/0.39.0+eddy
   R/3.6.1+Seurat                          neurodebian/0.39.0       (D)
   R/3.6.1-container                       openfoam/20161223
   R/3.6.3+Shawan                          openfoam/20200122        (D)
   R/4.0.0+Spatial                         paraview/5.4.1
   R/4.0.0                                 pepper+deepvariant/0.1.0
   R/4.0.2+Spatial                         picard/2.18.2
   R/4.0.2+tradeSeq+Seurat                 prokka/1.12
   bioconductor/3.6-rc2                    qgis/2.8.6
   bioconductor/3.8-rc2             (D)    quast/5.0.2
   biocontainers/maxquant/1.6.10.43        r+gcc/3.4.4
   biocontainers/seqkit/0.13.2             r+inla/3.5.0
   biolinux/8                              r+inla/20180712          (D)
   busco/3.0.2                      (D)    rjags/4.5.1
   centos/6.9-tinaroo2018           (D)    rjags/4.6.2--R-3.4.4     (D)
   centos6/9                               rstudio/1.0.143
   checkm/1.0.11                           rstudio/1.1.456-3.4.4
   cp2k/6.0                                rstudio/1.1.456-3.5.1    (D)
   crystalexplorer/17.5                    seewave/2.1.0
   deepvariant/1.0.0                       sevenz/20191203
   dssp/2.2.1                              vlab/4.4.1-xvfb
   glibc/2.17                              vlab/4.4.1               (D)
   glibc/2.19                              yacas/1.6.1
```

That a particular software package is provided using container, should not make it special or weird or any more difficult to use!

If your software is in the list above, then just load the module using a command like `module load biolinux/8`
Find out what shortcuts the module provides for you by using a command like `module display biolinux/8` command.
There will be an alias that will allow you to launch the container, and perhaps another alias to allow you to feed it commands to process.

If your favorite (containerized) software is not listed above then read on ...

The following sections on "Singularity is (just) Software" and "Our Singularity Container Catalog" will eventually be unnecessary.

Singularity is (just) Software

If you **`module load singularity`** you get access to the singularity software.

Loading the module also gets you some useful environment variables:

- SC the location of singularity container images
- SD the location of singularity container definition files

Brief help on the use of the singularity command is available using `singularity -h`
More detailed information is available at the Singularity website

You will not be able to use any "container management commands" like bootstrap, create, import etc.

Our Singularity Container Catalog

A brief description of each image is provided using the `module help singularity` command.

The containers are given names that hopefully indicate their content and function.
Most container names include version information.
Most containers will have a short cut name that will automatically run the container's runscript if invoked (see "run" Usage Mode below).

The images are contained in files with the .sapp extension and are located in the $SC directory.

Singularity Usage Options

exec

The exec command will execute an arbitrary command within the container.

```
module purge
module load singularity/2.5.2

uqdgree5@tinaroo1:~> singularity exec $SC/centos7.sapp /bin/cat /etc/centos-release
CentOS Linux release 7.3.1611 (Core)
```

You can feed more complex multi-line commands to your container using an executable script file :

```
singularity exec ./tensorflow.sapp  /home/uqdgree5/file_full_of_commands_to_run_in_container
```

run

The run command will launch the pre-defined run-script within the container.

The run script is declared in the definition file.
For containers supporting software with a single GUI, the GUI will launch automatically.
For command line containers, the container will launch a bash shell.

Because the image file is "executable", and is in your PATH after you load the singularity module, and most containers have a shortcut name, you can run the container in a variety of ways.

```
uqdgree5@tinaroo1:~> module load singularity

uqdgree5@tinaroo1:~> singularity run $SC/demo.sapp
This is what happens when you run the DeMo container...

uqdgree5@tinaroo1:~> $SC/demo.sapp
This is what happens when you run the DeMo container...

uqdgree5@tinaroo1:~> demo.sapp
This is what happens when you run the DeMo container...

uqdgree5@tinaroo1:~> DeMo
This is what happens when you run the DeMo container...
```

shell

The shell command will run a Bourne (/bin/sh) shell within the container.
Most of us find the `sh` shell a bit difficult to use and probably prefer `bash`.
Starting bash without .profile and .bashrc loading will avoid problems if your profile conflicts with the container environment.

Once your shell session starts in the container you can start bash over the top.
(just remember to exit out)

```
uqdgree5@tinaroo1:~> singularity shell $SC/cdo.sapp
Singularity: Invoking an interactive shell within container...
Singularity.cdo.sapp> /bin/bash  --norc --noprofile
Singularity.cdo.sapp> history
    1  history
```

```
Singularity.cdo.sapp> exit
exit
Singularity.cdo.sapp> exit
uqdgree5@tinaroo1:~>
```

You can also feed a complex set of commands to the container shell using a pipe and a here document
This is the best way to utilise a software container within a non-interactive batch job.

```
#Don't forget to preface this with any pre-container set up (eg. #PBS directives)

/bin/cat << TO_HERE | singularity shell $SC/tensorflow.sapp
cd mythesis
pwd

OUT_FILE="output.txt"

MY_HOME_DIR="/home/uqdgree5"

./my_simulation > $OUT_FILE

#the following will exit the singularity container
exit
TO_HERE

#Include any post-container commands (eg. remaining PBS job code)
```

Invoking shells within shells can get confusing!

The following is a "map" of where you go if you want to use

- bash shell,
- within a container shell,
- within an interactive job
- within a HPC login.

Don't forget to logout of it all when as you leave!

```
+ HPC login
+ + qsub -I -A ...
+ + + run singularity shell
+ + + + start bash shell
+ + + + + enter linux commands in bash running within the container
+ + + + exit the bash shell
+ + + exit the singularity shell
+ + exit the interactive job
+ HPC logout
```

test

The test command will execute any test code defined within the container.

Binding to Filesystems

All UQ RCC curated containers are deployed with the following "bind points" built into their filesystems:

- /nvme
- /30days
- /90days
- /QRISdata
- /TMPDIR
- /local

Singularity allows you to bind a path from the HPC filesystem into the container filesystem.
The syntax of the bind option is `-B external_location:internal_location`.

In an interactive batch job you might want to work on collection data as well as scratch and TMPDIR.

```
module purge
module load singularity/2.2.1
singularity shell -B /30days/$USER:/30days -B /90days/$USER:/90days -B /QRISdata/Q1234:/QRISdata -B $TMPDIR:/TMPDIR $SC/cdo
```

Using X11 Applications

If your container has software that generates X11 graphics, you can (in a suitable environment) display and interact with that

graphical user interface (GUI).

You could start that GUI via the container shell, or exec commands.
Most containers that RCC build have a handy alias for the main executable.

The GUI could also be the command that is launched automatically when you `run` the container.
In a suitable X11 enabled interactive batch job, you could do this

```
module purge

module load qgis/2.8.6

#To run qgis I can find the alias
uqdgree5@awoonga1:~/Tests> alias | grep singularity
alias qgis='/sw/Containers/singularity/bin/run_singularity run /sw/Containers/singularity/images/qgis-2.8.6'

#and run the alias
uqdgree5@awoonga1:~/Tests> qgis
```

If an alias for the main executable is not available, you can start a singularity shell and run your favourite executable that way.

For information about how to use X11 see the User Guides about Using PBSPro and Connecting to HPC.

Using InfiniBand and MPI

Currently, we are not installing IB drivers into the containers.
However that option is being explored.

Access to the storage over IB is mediated by the host.
You just need to bind to the storage location on the host.

Currently, we are not installing MPI packages into the containers.
However that option is being explored.

Being able to support multi-container message-passing jobs is something that is on our roadmap.

Using HPC Modules ... DON'T

Software containers are intended to be self contained.
All the software provided by the container should "just work".
This situation is similar to the BioLinux nodes on Euramoo that don't have modules because all bioinformatics tools are provided in standard path locations.

You would not need to load software modules from the HPC system into the container.
In fact, it would often be counterproductive to do so.

Some software is provided with a environment script that must be sourced.
The VLAB container is such an example.
You need to `source /opt/vlab/bin/sourceme.sh`

Using Software Containers in Batch Jobs

Your best option here is to create a text file with the commands that you would like the container to execute for you.

Then you invoke the container shell and feed the file of commands to it. (see the section on the `singularity shell` sub-command above.

Here is a prototype job script for a contained application that has aliases for `singularity run` and the `singularity shell`

```
#You will need your usual job specification lines, at the top
#PBS -A ...
#PBS -l ...

#We use an alias as a short cut. You need to allow this to work in non-interactive settings.
shopt -s expand_aliases

#Gotta load the contained app module
module load graftm/0.9.4

#Because "shell" is defined in the module you can use this shorthand once you have loaded the module.
#I have assumed that the file containing the commands to run lives in the $PBS_O_WORKDIR
#R Users may find it useful to @export LANG=C@ to stop warnings about a bunch of variables with names like LC_ ....
cat $PBS_O_WORKDIR/RunTheseCommandsInTheContainer.txt | shell
```

Alternatively, you can use the singularity exec command to execute your script file.
Imagine you have a commands file like this (named exec.sh) that you need to run in a BioLinux8 container.

```
#Simple demo of using the singularity exec command
echo
cat /etc/os-release
```

Then you could execute that script using

```
module load biolinux/8
singularity exec /sw/Containers/singularity/images/BioLinux8 exec.sh
```

or

if an appropriate alias has been created for your favorite container, it would be as simple as

```
module load biolinux/8
biolinux-exec exec.sh
```

Getting Help

Submit a support request to rcc-support@uq.edu.au after you have read all the relevant user guides and links provided herein.

If you have ideas for a container that would enhance your research productivity, you have options.

1. Use Singularity v2.2.1 on a machine you have root access on to build the image yourself.
   You will be able to run it on UQ HPC, from your own your copy or by adding the container to the catalog.
2. Create a definition file and submit it to RCC for addition to the container catalog.
   This could perhaps be approached as an overlay onto a pre-existing container or docker image.
3. Request that a definition file and image be created for you based on information the software tool's website.

The third option may take more time, compared to the other two mechanisms.

Things are simplest when a linux distribution provides your favourite software in a standard repository.
Even I can build those!! :-)

```

```

Further Reading

More about The Problem

A lot of the problem stems from the evolution of the versions of GLIBC that underpin every linux distribution:

Linux	**GLIBC**
CentOS 6	2.12
Ubuntu 12.04	2.15
CentOS 7	**2.17**
Ubuntu 14.04	2.19
Fedora 22	2.21
Ubuntu 16.04	2.23
Fedora 25	2.24
Ubuntu 18.04	2.27
Ubuntu 20.04	2.31
At 20200805	2.32

Our HPC systems are built using ROCKS and run CentOS 7 as their operating system which is built on GLIBC 2.17.
This is a strength because that operating system provides a stable manageable HPC environment.
This is also a weakness because the GLIBC version is a long way behind more contemporary operating systems.
You can check what GLIBC version is in vogue using the command **`ldd --version`**

Software containers gives us the ability to provide more up-to-date GLIBC environments without rebuilding the clusters.

A number of containers have been created based on different linux versions that are based on glibc. For details `module avail glibc`

```

```

Singularity Project Website

http://singularity.lbl.gov

The following diagram illustrates how containers are created and where and how they can be used.

The USER ENDPOINT must be a system administrator on the system creating the container and installing or modifying the content of the container.
The SHARED RESOURCE phase is how users on HPC utilise the container.

Singularity Publication

Kurtzer GM, Sochat V, Bauer MW (2017)
Singularity: Scientific containers for mobility of compute.
PLoS ONE 12(5): e0177459. https://doi.org/10.1371/journal.pone.0177459

Singularity Module Help Page

```
uqdgree5@tinaroo1:~> module help singularity

----------- Module Specific Help for 'singularity/2.2.1' ----------

    Sets environment for Singularity Container system.

    CONTAINERS SHOULD ONLY BE RUN FROM WITHIN INTERACTIVE OR REGULAR BATCH JOBS

    PLEASE DO NOT RUN CONTAINERS ON LOGIN OR REMOTE DESKTOP NODES

    For background and overview please refer to the Container User Guide which is
    linked from http://rcc.uq.edu.au/hpc

    The Container Catalogue
    Each entry includes brief description and suggested usage.

    cdo
        Climate Data Operators container based on Ubuntu:16.04 docker image.
        singularity shell -B /30days/$USER:/30days       \
                          -B /90days/$USER:/90days       \
                          -B /QRISdata/Q1234:/QRISdata   \
                          -B $TMPDIR:/TMPDIR             \
                          $SC/cdo.sapp

    centos7
        Base CentOS operating system
        singularity shell $SC/centos7.sapp

    centos7plus
        Base CentOS operating system plus additional packages like X11

    chewbacca7
        CentOS Linux release 7.3.1611 (Core) docker image with Chewbacca python enhancements.
        singularity shell $SC/chewbacca7.sapp

    rstudio-xenial
        Ubuntu 16.04LTS (Xenial) with RStudio 1.0.143
        rstudio-xenial.sapp
        You may need to set UTF-8 mode in RStudio global options for saving files. Needs X11.

    tensorflow
        TensorFlow CPU version based on tensorflow:tensorflow docker image.
        singularity shell $SC/tensorflow.sapp

    vlab
        Species simulation software based on Fedora release 22
        singularity shell $SC/vlab.sapp
        Requires X11 environment for full functionality. Also source /opt/vlab/bin/sourceme.sh
```

A Sample Session

Batch System Upgrade to PBSPro

In late 2017, all three clusters (Awoonga, FlashLite and Tinaroo) changed their batch system from Torque to PBSPro.

This was necessary to obtain better performance of the batch system under extreme loads and also to provide a number of diagnostic tools to help us, to help users.

A "wrapper script" has been deployed so that, in most cases, existing Torque job submissions will be understood and translated into PBSPro systax as the job is submitted.
You should work to migrate your job scripts to PBSPro syntax, and can use the `qstat -f JOB_NUMBER` command to learn about the PBSPro syntax corresponding to your submitted torque syntax.

A PDF copy of the PBSPro User Guide can be obtained from the Altair PBSWorks website for a copy of the user guide.
The user guides and documentation modules and other information about using the clusters currently contain examples of Torque syntax or output.

The following section or document may contain torque syntax or sample torque output.
It will be updated as soon as possible.
When a page or section has been updated, this notice will be removed from it.

The following is a log of a simple session.

```
   Job has been validated and ID will appear next if submission was within current queue resource limits.

   Interactive job should commence soon.

   qsub: waiting for job 82030 to start
qsub: job 82030 ready

########################## Execution Started ############################
JobId:82030
UserName:uqdgree5
GroupName:qris-uq
ExecutionHost:tn405d
################################################################################

uqdgree5@tn405d:~> echo $TMPDIR
/state/partition1/82030

uqdgree5@tn405d:~> module load singularity

uqdgree5@tn405d:~> singularity shell -B /30days/uqdgree5:/30days -B /90days/uqdgree5:/90days -B /QRISdata/Q0224:/QRISdata -
Singularity: Invoking an interactive shell within container...

Singularity.cdo.sapp> df -h
Filesystem                                          Size  Used Avail Use% Mounted on
/dev/loop0                                          1.3G  849M  320M  73% /
/dev/sda5                                           852G   51M  852G   1% /tmp
/dev/sda1                                            48G   31G   15G  68% /etc/hosts
devtmpfs                                             64G  204K   64G   1% /dev
tmpfs                                                64G     0   64G   0% /dev/shm
/dev/sda2                                            29G  1.1G   27G   4% /var/tmp
/dev/gpfs1                                           25T   12T   14T  47% /90days
10.255.122.70:/gpfs/general/collections/Q0224/Q0224  1.1P   97T  932T  10% /QRISdata/Q0224

Singularity.cdo.sapp> ls -sal /TMPDIR
total 4
0 drwxr-xr-x  2 uqdgree5 qris-uq    6 May 24 05:21 .
4 drwxr-xr-x 29 root     root    4096 May 24 03:24 ..

Singularity.cdo.sapp> bash
uqdgree5@tn405d:~$ which cdo
/usr/bin/cdo
uqdgree5@tn405d:~$ which ncdump
/usr/bin/ncdump

uqdgree5@tn405d:~$ exit
exit

Singularity.cdo.sapp> exit

uqdgree5@tn405d:~> exit
logout

qsub: job 82030 completed
uqdgree5@tinaroo1:~>
```