pawsey

Supercomputing User Training

Module 6: Using Software Containers

Pawsey Training Series

Supercomputing User Training

1. Supercomputing Introduction
2. Logging In
3. Filesystems Overview
4. Moving Data In and Out
5. Using Software Modules
6. Using Software Containers
7. Accounting Model Overview
8. Job Scheduling Overview
9. Running Jobs
10. Testing Job Runs
11. Managing Project Data

Outcomes for this Module

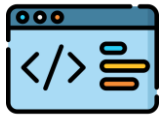
- Describe what are containers and why they can be useful in supercomputing
- Download and run software containers
- Describe what container modules are and how they simplify usage of containers
- Install and use container modules

✓ Prerequisite knowledge:

- ✓ **Bash shell basics**
- ✓ **User Training 02: Logging In**
- ✓ **User Training 05: Using Software Modules**

Watch for These Signs!

Definition of new concepts



Hands-on coding (demo)

Best practices

Exercises and solutions

Warnings (bad practices)

Links to user documentation

Icons made by Freepik from www.flaticon.com

Pawsey

Australian Government

NCRIS National Research Infrastructure for Australia

CSIRO

Curtin University

Murdoch University

Government of Western Australia

ECU Edith Cowan University

The University of Western Australia

Software Containers

What are Software Containers and Why Are They Useful?

Container
A unit of software (sandbox) that isolates an application and all its dependencies.
To be executed, it relies on a "container runtime" and on the underlying operating system.

Container Runtime
A utility that is required in order to manage and execute software containers.

Advantages

- Improved reproducibility and portability
  - Across infrastructures and environments
- Productivity
  - Build container once, download and run it anywhere (simpler software "installation")
- Scalability of software is kept or improved
  - Can exploit MPI parallelisation
  - Single container file can improve I/O performance of some workflows on parallel filesystems

Possible drawbacks

- Possible performance hit to achieve cross-architecture portability
- Installation of container runtime requires system administrators
- MPI support requires care by infrastructure

Good Use Cases for Containers

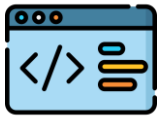
- Large volume of software to be installed (e.g. bioinformatics)
- Complex builds with lots of dependencies (e.g., deep learning frameworks)
- Dependency trees with high chances of package/version conflicts (e.g., Python and R workflows)
- Dependency trees with tens/hundreds of small packages (e.g. Python)
  - Single container file can improve performance on parallel filesystems
- Intensive I/O patterns in software that write a large number of small files (e.g. OpenFoam)
  - Single container file can improve performance on parallel filesystems

Containers at Pawsey

- The Singularity container runtime is available on all Pawsey supercomputers
  - On Setonix: `module load singularity/3.8.6`
- Configurations from Pawsey staff to reduce learning curve
  - Relevant filesystems available in containers by default
  - MPI performance tuned via module settings
- Option to install as Container Modules to further reduce learning curve (more on this later)
  - Several recipes for Container Modules contributed by Pawsey staff
- Collection of Pawsey-curated container images: MPI base images, Python for HPC, OpenFoam

More details @
- Containers
- Pawsey Webinar Series on Containers (Youtube)

Download and Run Containers with Singularity



DEMO: registry lookup, `singularity pull` and `singularity exec`

Container Tag
The version of a container. Usually contains info on the software version, and optionally additional strings.

Prefer versioned tags when using containers.
A container tag with a version improves reproducibility, compared to generic tags such as “latest”.

- Popular online registries to lookup containers
  - Red Hat Quay.io
  - Docker Hub
- Container examples
  - Ubuntu base image: `ubuntu:22.04`
  - Blast: `quay.io/biocontainers/blast:2.12.0--pl5262h3289130_0`

Download and Run Containers with Singularity

- Key commands
  - Download: `singularity pull docker://[registry/][repository/]name:tag`
  - Execute commands: `singularity exec container_file.sif command options`

- How to run GPU-enabled containers
  - Additional Singularity option
    - Nvidia GPUs: `singularity exec --nv container_file.sif command options`
    - AMD GPUs: `singularity exec --rocm container_file.sif command options`
  - Relevant Pawsey supercomputers
    - Nvidia: Topaz, Garrawarla
    - AMD: Setonix Phase 2 [future]

OUTPUTS: Download and Run Containers with Singularity

```
$ cat /etc/os-release
NAME="SLES"
VERSION="15-SP2"
VERSION_ID="15.2"
PRETTY_NAME="SUSE Linux Enterprise Server 15 SP2"
ID="sles"
ID_LIKE="suse"
ANSI_COLOR="0;32"
CPE_NAME="cpe:/o:suse:sles:15:sp2”
```

```
$ cd $MYSCRATCH
$ module load singularity/3.8.6
$ singularity pull docker://ubuntu:22.04
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
..
INFO:    Creating SIF file...

$ singularity exec ubuntu_22.04.sif cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.1 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.1 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms..
UBUNTU_CODENAME=jammy
```

EXERCISE: Pull and Execute Containers

1. Which version of Python 3 is available at login on Setonix?
   - tip: use `python3 --version`
2. Go to your `$MYSCRATCH` directory
3. Load the Singularity module
4. Download the container `python:3.9.13-slim`
   - tip: remember the `docker://` prefix!
5. Check the Python 3 version within this container

OUTPUTS: Pull and Execute Containers

```
$ python3 --version
Python 3.6.13

$ cd $MYSCRATCH
$ module load singularity/3.8.6
$ singularity pull docker://python:3.9.13-slim
INFO:    Converting OCI blobs to SIF format
INFO:    Starting build...
..
INFO:    Creating SIF file...

$ singularity exec python_3.9.13-slim.sif python3 --version
Python 3.9.13
```

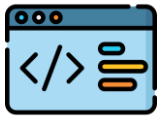
Container Modules: the What and the Why

Container Module
An interface to a container (a way to provide it), so that the included software can be run using the same commands as for a traditional installation.

- Container modules make using software containers more accessible, cutting their learning curve
- Singularity Registry HPC (SHPC) is a tool to automate the installation of containers as container modules
  - SHPC uses shell scripts (on Setonix) or shell functions to wrap and hide the Singularity execution syntax
  - SHPC has a database of lightweight recipes for each container, to know about available version tags and software commands (to write appropriate wrappers)
- A number of applications on Setonix are installed as container modules
  - Most bioinformatics packages
  - OpenFoam
  - Python for HPC containers

More details @
- SHPC (Singularity Registry HPC)

Lookup, Install and Use Container Modules



DEMO: `shpc show`, `shpc install`, then standard use of software

```
$ module load shpc/0.0.53
$ shpc show -f python --v
..
python:3.9.2-slim
..
$ shpc install python:3.9.2-slim
singularity pull --name /software/projects/..
..
Module python:3.9.2-slim was created.
$ module unload shpc/0.0.53

$ module avail python
..
python/3.9.2-slim/module
..
$ module load python/3.9.2-slim/module
$ python --version
Python 3.9.2
```

Lookup and installation phase (once-off for a given software)

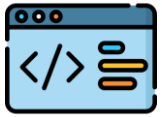
- Load the SHPC module (only for lookup and installation)
- Look for a specific package
- Install package with version tag (as found in output above)
- Unload SHPC module (not needed to use the software)

Software use phase: as any traditional software module

- NOTE: the extra `/module` suffix will go away in future versions of SHPC

Summary

- Terms we learnt
  - Container
  - Container Runtime
  - Container Tag
  - Container Module



- Tasks we learnt
  - Download and run containers: `singularity pull` and `singularity exec`
  - Search and install container modules: `shpc show` and `shpc install`

- Always prefer versioned tags when using containers

Pawsey

Australian Government

NCRIS National Research Infrastructure for Australia

CSIRO

Curtin University

Murdoch University

Government of Western Australia

ECU Edith Cowan University

The University of Western Australia

Getting Help

Getting Help

https://support.pawsey.org.au

Pawsey has extensive User Support Documentation.

Areas covered include:

- System user guides
- Knowledge Base
- Pawsey-supported software list
- Maintenance logs
- Policies and terms of use

For further assistance, contact the help desk, via User Support Portal.

Help us to help you by providing details, such as:

- Which resource
- Error messages
- Location of files
- SLURM job id
- Your username if having login issues
- Never tell us (or anyone) your password!

Become a Pawsey Friend and receive our Newsletter:
https://pawsey.org.au/pawsey-friends/

Q & A Session