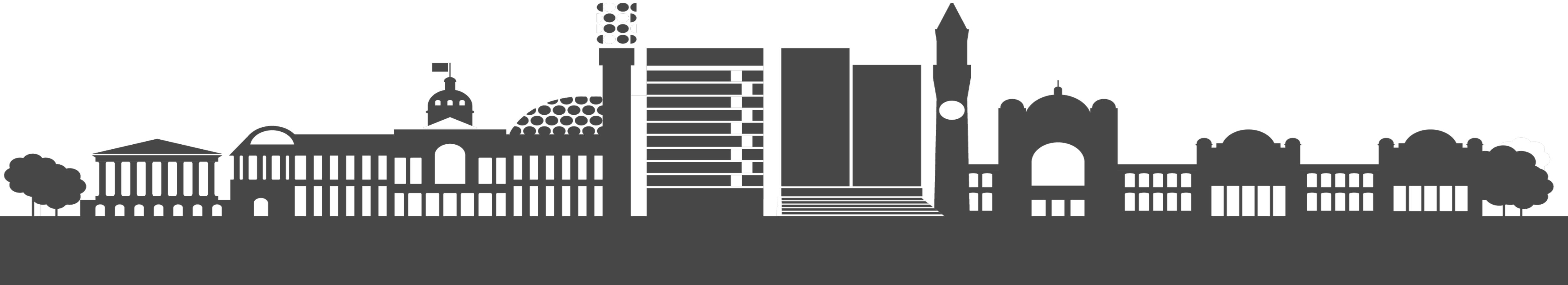
Baskerville

Self-Install for Python Apptainer Mon 27-11-2023

Baskerville RSE SimonH



Important Sites

- https://docs.baskerville.ac.uk
- https://apps.baskerville.ac.uk
- https://portal.baskerville.ac.uk
- https://docs.baskerville.ac.uk/self-install

System Python and which Python

When you first log in to Baskerville, the system Python is available, but this is almost always not what you want. To learn about the system Python, run these commands:

System Python and which Python

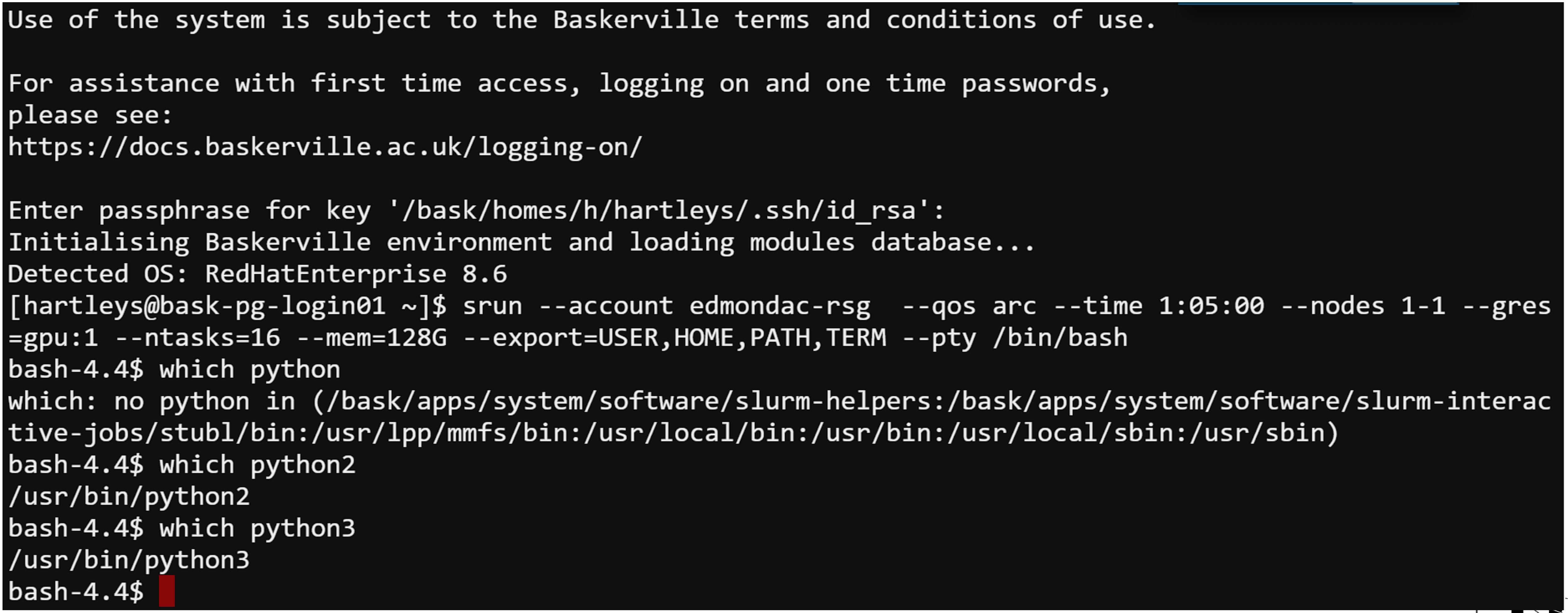
```
$python2 --version
Python 2.7.18

$which python2
/usr/bin/python2

$python3 --version
Python 3.6.8

$which python3
/usr/bin/python3
```

System Python and which Python



```
Use of the system is subject to the Baskerville terms and conditions of use.

For assistance with first time access, logging on and one time passwords,
please see:
https://docs.baskerville.ac.uk/logging-on/

Enter passphrase for key '/bask/homes/h/hartleys/.ssh/id_rsa':
Initialising Baskerville environment and loading modules database...
Detected OS: RedHatEnterprise 8.6
[hartleys@bask-pg-login01 ~]$ srun --account edmondac-rsg  --qos arc --time 1:05:00 --nodes 1-1 --gres
=gpu:1 --ntasks=16 --mem=128G --export=USER,HOME,PATH,TERM --pty /bin/bash
bash-4.4$ which python
which: no python in (/bask/apps/system/software/slurm-helpers:/bask/apps/system/software/slurm-interac
tive-jobs/stubl/bin:/usr/lpp/mmfs/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin)
bash-4.4$ which python2
/usr/bin/python2
bash-4.4$ which python3
/usr/bin/python3
bash-4.4$
```

PIP and Python

PIP is a recursive acronym for “Preferred Installer Program” or PIP Installs Packages. It is a command-line utility that installs, reinstalls, or uninstalls PyPI packages with one simple command: pip.

Common pip commands

View the help menu:

```
$pip -h
```

The help menu for the install command:

```
$pip install --help
```

Search the Python Package Index PyPI for a given package (e.g., jax):

```
$pip search jax
```

List all installed packages:

```
$pip list
```

Install pairtools and pyblast for version 3.5 of Python

```
$pip install python==3.5 pairtools pyblast
```

Common pip commands

Install a set of packages listed in a text file

```
$pip install -r requirements.txt
```

To see detailed information about an installed package such as sphinx:

```
$pip show sphinx
```

Upgrade the sphinx package:

```
$pip install --upgrade sphinx
```

Uninstall the pairtools package:

```
$pip uninstall pairtools
```

System Python and which Python

We see that both **python2** and **python3** are installed in a system directory.

Using which python can tell you useful information for virtual env’s

Check Quota

Python packages can require many gigabytes of storage. By default they are installed in your /home directory. Be sure to run the `my_quota` command before installing to make sure that you have space.

- https://docs.baskerville.ac.uk/cheatsheet

Self-installing Python software

We provide some Python software. To install your own Python software on top of these we recommend using a virtual environment. To create this:

1. Load the required modules
2. In a suitable directory, create a virtual environment:

```
 python -m venv --system-site-packages venvname
```

3. Activate the virtual environment.

```
source venvname/bin/activate
```

4. Install your Python software, for example:

```
pip install packagename
```

Using Self-installed Python software

To use the Python software installed in the virtual environment:

1. Load the same modules as used when creating the virtual environment
2. Activate the virtual environment

```
source venvname/bin/activate
```

3.Use your Python software

Installing Python Packages from Source

In some cases you will be provided with the source code for your package. To install from source do:

```
$python setup.py install --prefix=</path/to/install/local
```

For help menu use **python setup.py --help-commands**. Be sure to update the appropriate environment variables in your **~/.bashrc** file:

```
$python setup.py install --prefix=</path/to/install/local
```

ModuleNotFoundError

Python users can encounter the situation where a module is not found. For example:

```
$python
>>> import bigCell
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named ‘bigCell'
```

ModuleNotFoundError

To see where Python is looking for modules, run the following commands:

```
$ python
>>> import sys
>>> print(sys.path)
```

ModuleNotFoundError

The search paths are stored in sys.path. If your module is not found in those paths then you will encounter a **ModuleNotFoundError**. The first entry in the list above (i.e., ' ') corresponds to the current directory. The most relevant path usually ends with "site-packages".

Note that sys.path is a Python list that can be modified to include an additional path:

```
>>> sys.path.append("/home/aturing/mymodules")
```

ModuleNotFoundError

One can also remove paths from sys.path if conflicting modules are being found in different directories.
The PYTHONPATH environment variable can also be used to modify the Python search paths.

```
>>> sys.path.remove("/home/aturing/mymodulesbad")
```

Common FAQ

- The installation directions that I am following say to use pip3. Is this okay?
  - Do not use pip3 for installing Python packages. pip3 is a component of the system Python and it will not work properly with Anaconda. Always do module load miniconda/4.10.3 and then use pip for installing packages.

Common FAQ

- I tried to install some packages but now none of my Python tools are working. Is it possible to delete all my Python packages and start over?

  - Yes. Packages installed by pip are in ~/.local/lib while conda packages and environments are in ~/.conda. If you made any environments with virtualenv you should remove those as well. Removing these directories will give you a clean start. Be sure to examine the contents first. It may be wise to selectively remove sub-directories instead. You may also need remove the ~/.cache directory and you may need to make modifications to your .bashrc file if you added or changed environment variables.

Common FAQ

- How are my pip packages built? Which optimization flags are used?
  - After loading the miniconda/conda module, run this command:

```
$module load bask-apps/live
$module load Miniconda3/4.10.3
$which python3
$python3 --version
$python3.9-config --cflags
```

Common FAQ

- To force a package to be built from source with certain optimization flags do, for example:
  - CFLAGS="-O1" pip install numpy -vvv --no-binary=numpy

Common FAQ

- Is it okay if I combine virtualenv and conda?
  - This is highly discouraged. While in principle it can work, most users find it just causes problems. Try to stay within one environment manager.
  - Note that if you create a conda environment you can use pip to install packages.
- Can I combine conda and pip?
  - Yes, and this tends to work well. A typical session may look like this:

```
$module load bask-apps/live
$module load Miniconda3/4.10.3
$conda create --name myenv python=3.9.5
$conda activate myenv
$pip install scitools
```

Test install pip chill

Lets try to create an environment with pip chill
Note that like pip, **virtualenv** is an executable, not a library. To create an isolated environment do:

```
$mkdir myenv
$virtualenv myenv
$source myenv/bin/activate
```

Why Container Environments

Deploying Applications:
Building software is often a complicated business, particularly on a shared and multi-tenant systems:
- HPC clusters have typically very specialized software stacks which might not adapt well to general purpose applications.
- OS installations are streamlined.

Some applications might need dependencies that are not readily available and complex to build from source.
- End users use Ubuntu, cluster typically use RHEL, or other specialized OS.

    sudo apt-get install will not work !!!!!

- Researcher’s code often tends to comes from some old repos.

What Containers are trying to Solve

Portability and Reproducibility:

- Running applications on multiple systems typically needs replicating the installations multiple times making it hard to keep consistency.

- It would be useful to publish the exact application used to run a calculation for reproducibility or documentation purpose.

- As a user can I minimize the part of the software stack I have no control on, to maximize reproducibility without sacrificing performance too much?

Singularity/Apptainer Glossary

Singularity/Apptainer – the software
– As in “Singularity 3.8” or “Apptainer 1.0”

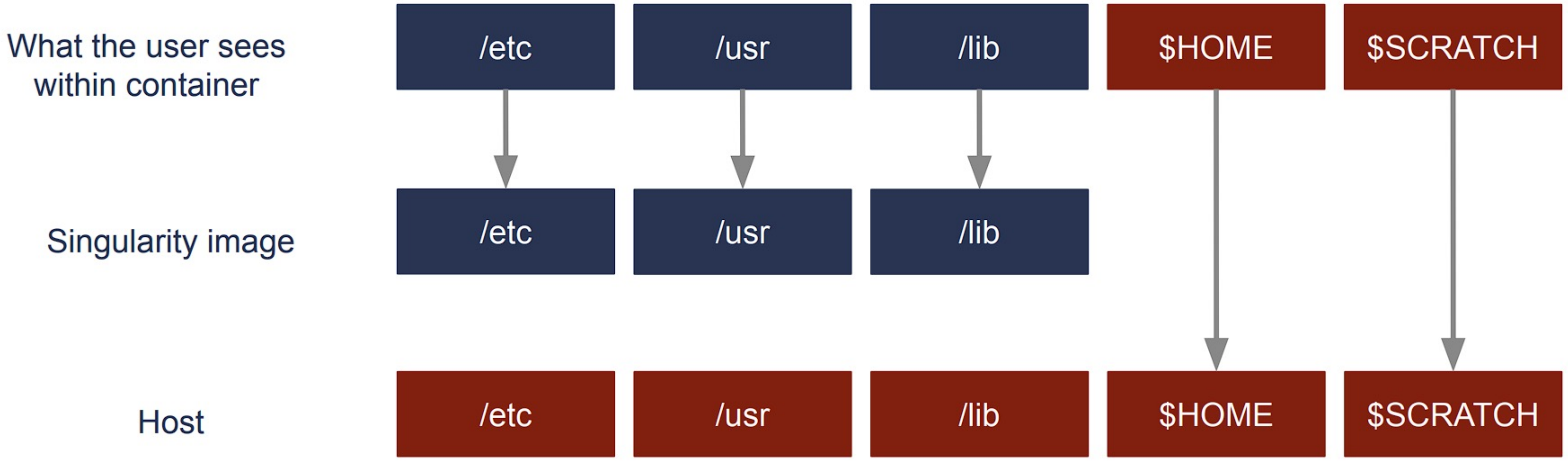
- **Image** – a compressed, usually read-only file
  - – Example: “Build a Matlab 2021a image”
  - – Writable image: use --sandbox option

- **Container**
  - – The technology: “containers vs. virtual machines”
  - – An instance of an image
    - Example:

      “process my data in a Singularity container of Matlab”

- **Host** – computer/supercomputer where the image is run

Singularity/Apptainer



Singularity/Apptainer

Singularity/Apptainer provides a container runtime and an ecosystem for managing images that is suitable for multi-tenant systems and HPC.

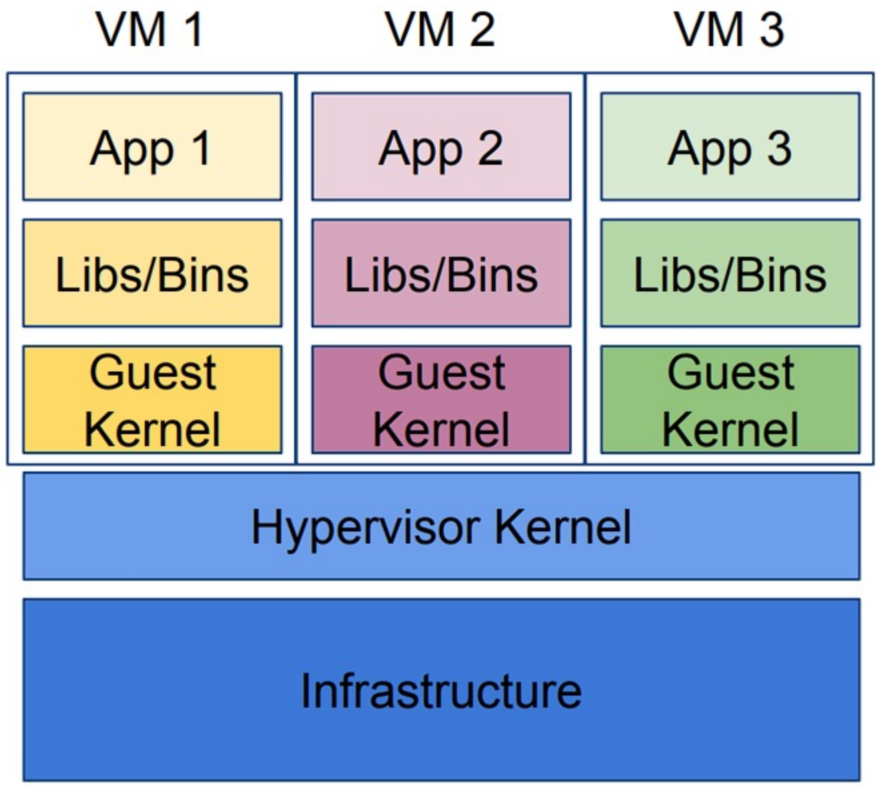
Important aspects :

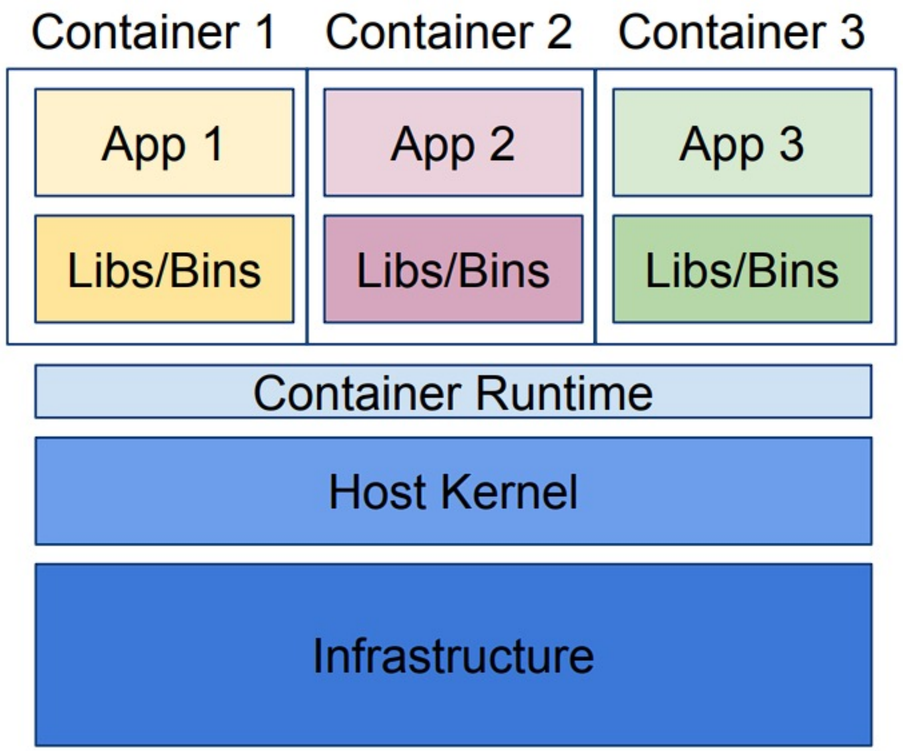
- no need to have elevated privileges at runtime, although root privileges are needed to build the images.
- each applications will have its own container
- containers are not fully isolated (e.g. host network is available)
- users have the same uid and gid when running an application
- containers can be executed from local image files, or pulling images from a docker registry

For basic usage refer to:

https://docs.baskerville.ac.uk/containerisation/
https://www.sylabs.io/docs/
https://apptainer.org/

Singularity/Apptainer



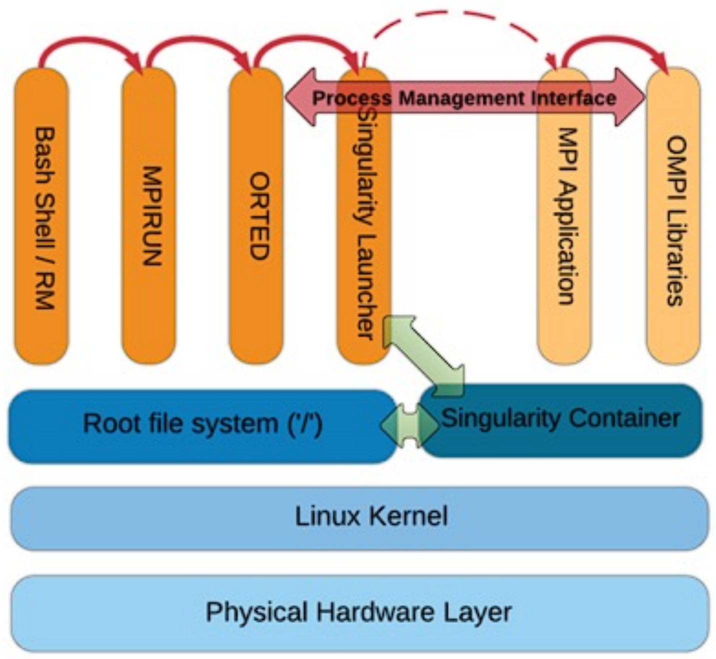


VMs:
hardware virtualization + OS

Containers:
User defined software stack

Singularity/Apptainer MPI

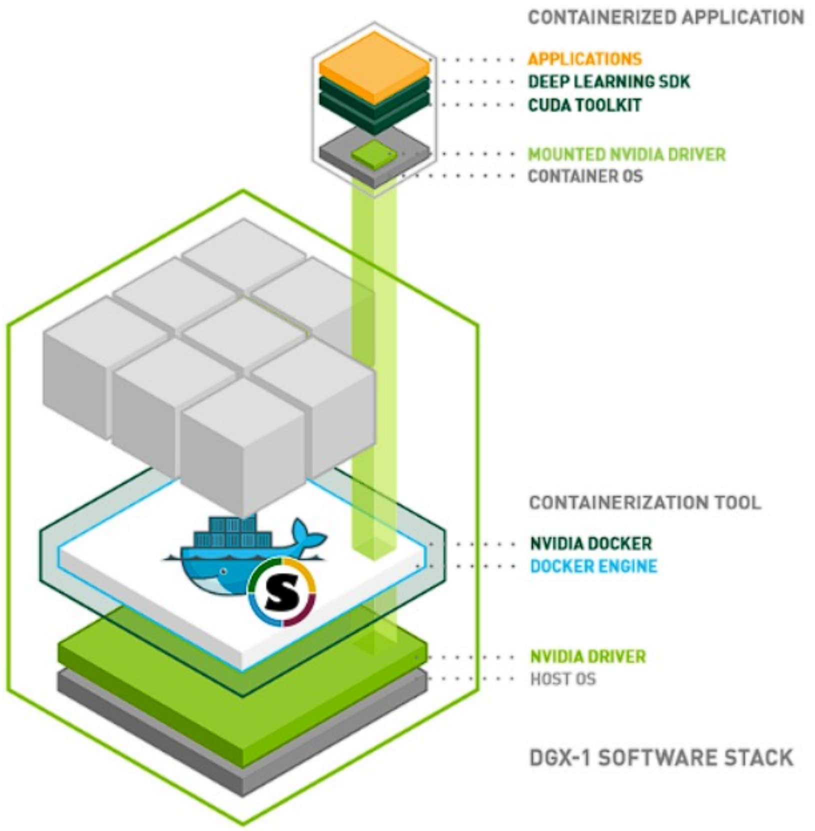
- Use same Message Passing Interface (MPI) distribution and version within container as would be used outside the container.
- If using Infiniband (IB), install same OFED drivers and libraries inside the container as used on underlying HPC hardware.



Singularity/Apptainer GPU

- GPU-accelerated containers also require an interface for accessing GPU drivers and libraries on the underlying host system.

- Apptainer allows you to bind mount the GPU driver and its supporting libraries at runtime with the --nv option.



Singularity/Apptainer

```
 $ apptainer help

Linux container platform optimized for High Performance Computing (HPC) and
Enterprise Performance Computing (EPC)

Usage:
  apptainer [global options...]

Description:
  Apptainer containers provide an application virtualization layer enabling
  mobility of compute via both application and environment portability. With
  Apptainer one is capable of building a root file system that runs on any
  other Linux system where Apptainer is installed.

Options:
  -d, --debug     print debugging information (highest verbosity)
  -h, --help      help for apptainer
      --nocolor   print without color output (default False)
  -q, --quiet     suppress normal output
  -s, --silent    only print errors
  -v, --verbose   print additional information

Available Commands:
 build       Build an Apptainer image
  cache       Manage the local cache
  capability  Manage Linux capabilities for users and groups
  exec        Run a command within a container
  help        Help about any command
  inspect     Show metadata for an image
  instance    Manage containers running as services
  key         Manage OpenPGP keys
  oci         Manage OCI containers
  plugin      Manage apptainer plugins
  pull        Pull an image from a URI
  push        Upload image to the provided URI
  remote      Manage apptainer remote endpoints
  run         Run the user-defined default command within a container
  run-help    Show the user-defined help for an image
  search      Search a Container Library for images
  shell       Run a shell within a container
  sif         siftool is a program for Singularity Image Format (SIF) file manipulation
  sign        Attach a cryptographic signature to an image
  test        Run the user-defined tests within a container
  verify      Verify cryptographic signatures attached to an image
  version     Show the version for Apptainer

Examples:
  $ apptainer help <command> [<subcommand>]
  $ apptainer help build
  $ apptainer help instance start


For additional help or support, please visit https://www.apptainer.org/docs/y!
```

Singularity/Apptainer

The main Apptainer command

```
apptainer [options] [subcommand options] ...
```

has three essential subcommands:

- **build**: Build your own container from scratch using a Singularity definition (or recipe) file; download and assemble any existing Singularity container; or convert your containers from one format to another (e.g., from Docker to Singularity)

- **shell**: Spawn an interactive shell session in your container.

- **exec**: Execute an arbitrary command within your container

Singularity/Apptainer

The main Apptainer tutorial:

COW say! Moo/ Hello World

```
apptainer pull docker://ghcr.io/apptainer/lolcow

apptainer shell lolcow_latest.sif
>Apptainer lolcow_latest.sif:~> whoami

apptainer exec lolcow_latest.sif cowsay moo
```

UNIVERSITY OF BIRMINGHAM | BEAR BIRMINGHAM ENVIRONMENT FOR ACADEMIC RESEARCH

Time for Q’s