Containers & Singularity

Introduction

> *HPC group will give guidance to create and use singularity container but will not manage user containers. If you decide to use singularity it is your responsibility to build and manage your containers. That includes managing your linux environment for development.*

Software has grown in complexity over the years making it difficult at times to simply run the software. Containers address this problem by bundling an application together with its software dependencies, scripts, documentation, license and a minimal operating system within a self-sustainable image so that when it comes to running the software everything "just works". By containerizing the application platform, it makes the software both **sharable** and **portable** while the output becomes **reproducible**.

Singularity: A Secure Alternative to Docker

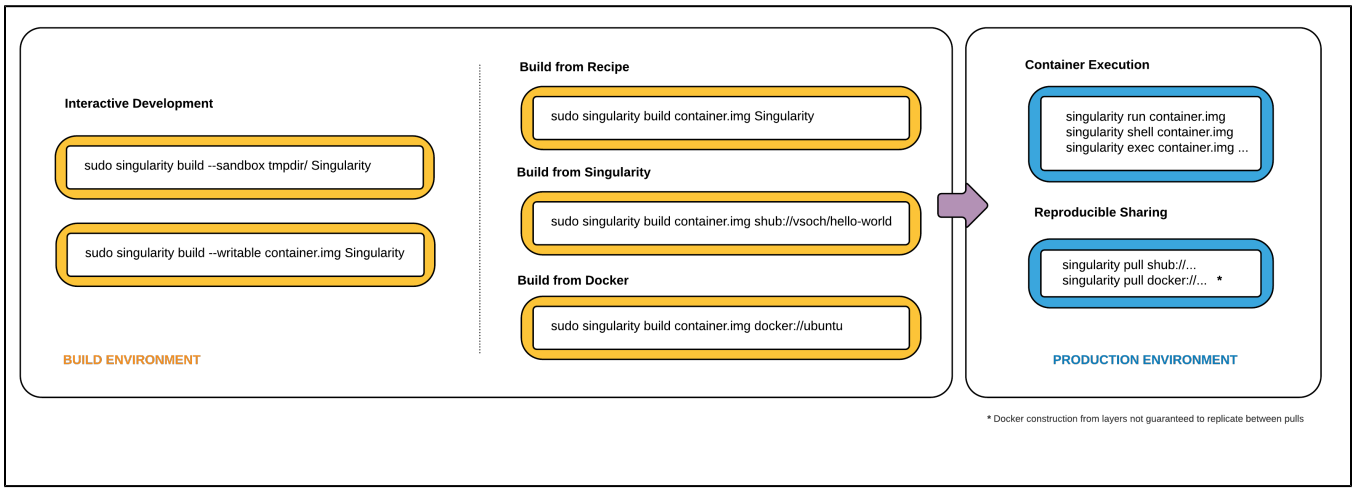
> *Singularity allows running Docker containers natively, and is a great replacement for Docker on HPC systems. This means that existing Docker container can be directly imported and natively run with Singularity.*

Singularity is a tool that we offer for running containers on our clusters, similar to Docker. Docker images are not secure because they provide means to gain root access to the system they are running on. Singularity is a secure HPC alternative container framework that uses a completely different implementation that doesn't require any elevated privileges to run containers, while also allowing direct interaction with existing Docker containers. Learn about the differences between virtual machines, Docker, and Singularity.

The main motivation to use Singularity over Docker is the fact that it's been developed with HPC systems in mind that solves these problems:

- Security: A user in the container is the same user as the one running the container, no privilege escalation
- Ease of deployment: No daemon running as root on each node, a container is simply executable
- Ability to run massively-parallel (MPI) applications by leveraging fast InfiniBand interconnects and GPUs with minimal performance loss

Singularity Workflow



More Documentation

The following documentation is intended for using Singularity on the MSK HPC clusters. For more complete documentation about building and running containers with Singularity, please see the Singularity documentation.

Singularity Usage

Singularity is a container platform specifically for high-performance computing.

Obtaining the Image: Using the *pull* Command

Some software is provided as a Singularity image with the .sif or .simg file extension. A Docker image could also be provided which must be converted to a Singularity image. For example, if the installation directions say:

```
$ docker pull godlovedc/lolcow:latest
```

Then download and convert the Docker image to a Singularity image with:

```
$ singularity pull docker://godlovedc/lolcow:latest
```

This will produce the file lolcow_latest.sif in the current working directory, where “latest” is a specific version of the software or a "tag". Other cases include images on the Singularity Cloud Library:

```
$ singularity pull library://sylabsed/examples/lolcow:1.0
```

In some cases the build command can be used to create the image, examples below:

```
$ singularity build lolcow.simg docker://godlovedc/lolcow
$ singularity build hello-world.simg shub://vsoch/hello-world
```

Unlike pull, build will convert the image to the latest Singularity image format after downloading it.

Obtaining the Image: Working from a Dockerfile

Some software is provided as a Dockerfile instead of an actual container. In this case, if you have Docker installed on your local machine (e.g., laptop) then you can create the Docker image yourself and then transfer it to one of the HPC clusters where the Singularity image can be built.

On your local machine, after making the Docker image, get the image id by running this command:

```
$ docker images
```

Next, save that image as a tar file. In this example, the image ID is *c230486ba945*.

```
$ docker save c230486ba945 -o myimage.tar
```

Copy myimage.tar to one of the HPC clusters using scp and then create the Singularity image. The commands will look as follows:

```
$ scp myimage.tar <user-id>@lilac.mskcc.org:/path/to/destination
$ ssh <user-id>@lilac.mskcc.org
$ cd /path/to/destination
$ module load singularity/3.7.1
$ singularity build [IMAGE NAME].sif docker-archive://myimage.tar
```

Building Singularity Images

Singularity images can be built from scratch using a definition file which is a text file that specifies the base image, the software to be installed and other information. However, you need root access to the build Singularity containers and therefore you won’t be able to do so on the cluster. Possible options are:

- Building on a Linux system to which you have root (admin) access
- Building on a virtual machine
- Or consider creating Docker images since it has a larger community with more support and then converting it into a Singularity image

Detailed documentation about building Singularity container images is available here.

Running Containers

To run the **default command** within the Singularity image:

```
$ singularity run ./[IMAGE NAME].sif <arg-1> <arg-2> ... <arg-N>
```

> *Note: Some containers may not have a default command. Refer to the complete documentation for the singularity run command.*

To run a **specific command** that is defined within the container, use singularity exec:

```
$ singularity exec ./[IMAGE NAME].sif python3 myscript.py
```

Use the **shell command** to run a shell within the container. This command is useful for searching for certain files within the container.

```
$ singularity shell ./[IMAGE NAME].sif
Singularity> cat /myscript.py
Singularity> cd /
Singularity> ls -l
Singularity> exit
```

Available Files and Storage Space

A running container automatically bind mounts these paths:

- /home/<user>
- the directory from which the container was ran

This makes it easy for software within the container to read or write files on the cluster filesystems. For instance, if your image is looking for an argument that specifies the path to your data then one can simply supply the path:

```
$ singularity run [IMAGE NAME].sif -d /lila/data/user/mydata
```

Binding Directories

Additionally, there are two options to create your own custom bind mounts within your containers.

- The --bind option to bind directories:

```
$ singularity exec --bind /data:/mnt [IMAGE NAME].sif
```

To bind multiple directories in a single command:

```
$ singularity shell --bind /opt,/data:/mnt [IMAGE NAME].sif
```

- Using the environment variable $SINGULARITY_BIND instead of the command line argument:

```
$ export SINGULARITY_BIND="/opt,/data:/mnt"
$ singularity shell [IMAGE NAME].sif
```

For more information see bind mounting on the Singularity website.

Environment Variables

Singularity by default exposes all environment variables from the host inside the container. Use the --cleanenv argument to prevent this:

```
$ singularity run --cleanenv [IMAGE NAME].sif <arg-1> <arg-2> ... <arg-N>
```

One can then define an environment variable within the container:

```
$ export SINGULARITYENV_MYVAR="Overridden"
```

With the above definition, MYVAR will have the value "Overridden". You can also modify the PATH environment variable within the container using definitions as follows:

```
$ export SINGULARITYENV_PREPEND_PATH=/opt/important/bin
$ export SINGULARITYENV_APPEND_PATH=/opt/fallback/bin
$ export SINGULARITYENV_PATH=/only/bin
```

More information can be found on the Singularity website on Environment and Metadata.

Inspecting the Definition File

As mentioned earlier, a Singularity definition file is the recipe by which the image was made. One can inspect the definition file to learn more about a particular image such as the following:

```
$ singularity inspect --deffile lolcow_1.0.sif
BootStrap: library
From: ubuntu:16.04

%post
    apt-get -y update
    apt-get -y install fortune cowsay lolcat

%environment
    export LC_ALL=C
    export PATH=/usr/games:$PATH

%runscript
    fortune | cowsay | lolcat
```

However, if the image was taken from Docker Hub then a definition file will not be available.

Example of Docker to Singularity Conversion

Here is an example of converting directions for Docker to Singularity:

DOCKER	SINGULARITY
`$ docker run -v /host/output:/data -v /host /release89:/refdata ecogenomic/gtdbtk --help`	`$ singularity run -B /host/output:/data -B /host/release89: /refdata </path/to>/gtdbtk_1.1.1.sif --help`

The Singularity image in the above can be obtained with:

```
$ singularity pull docker://ecogenomic/gtdbtk:1.1.1
```

To learn about binding a directory within the container to a directory on the host, please refer to the -B flag from this command: $ singularity help run

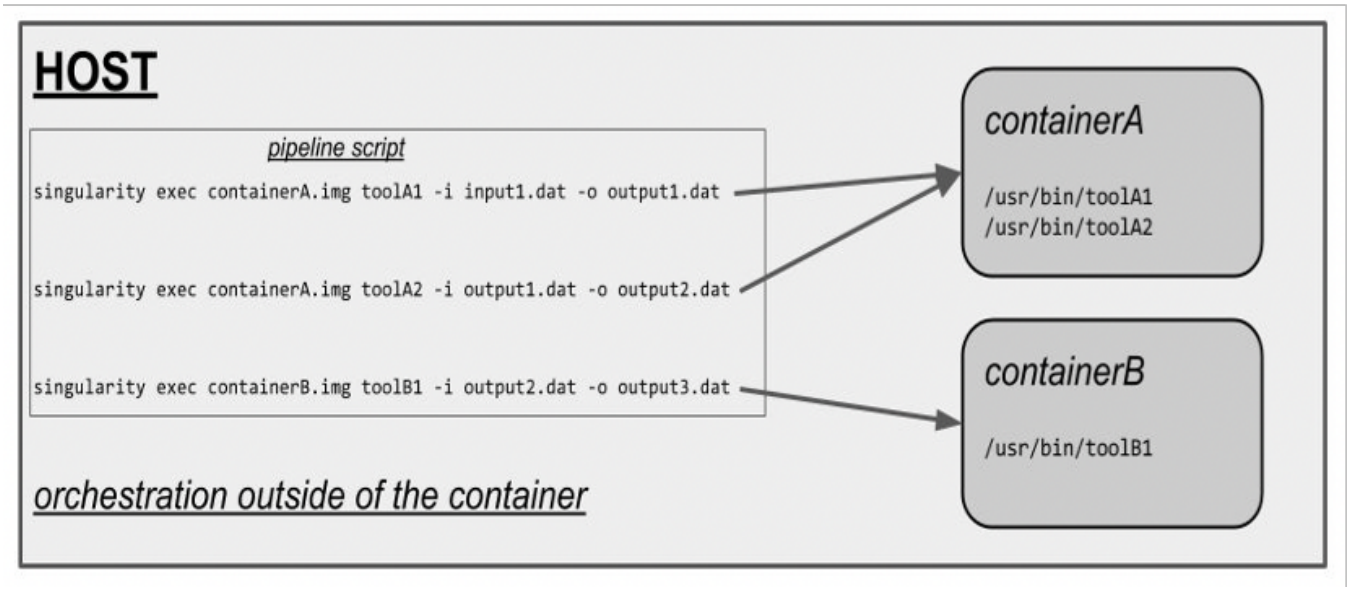
$ singularity help run

```
...
-B, --bind strings          a user-bind path specification.  spec has
                              the format src[:dest[:opts]], where src and
                              dest are outside and inside paths.  If dest
                              is not given, it is set equal to src.
                              Mount options ('opts') may be specified as
                              'ro' (read-only) or 'rw' (read/write, which
                              is the default). Multiple bind paths can be
                              given by a comma separated list.
```

Container Design Strategies

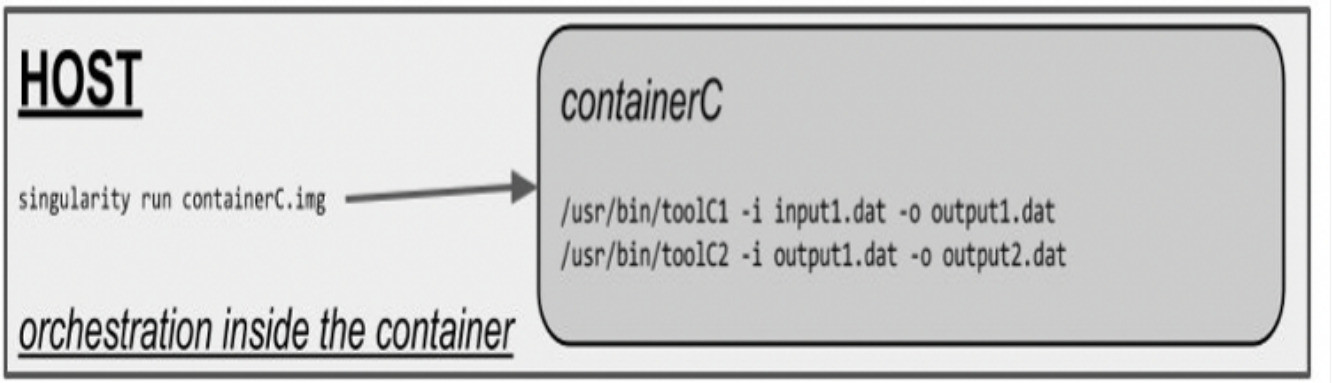
There are different ways to design containers when their purpose is to encapsulate pipelines: make the orchestration **either inside the container**, or to make it from **outside the container** and simply make the calls to software located inside the container.

- **A 3-step pipeline: using two containers that have dependencies installed inside.**



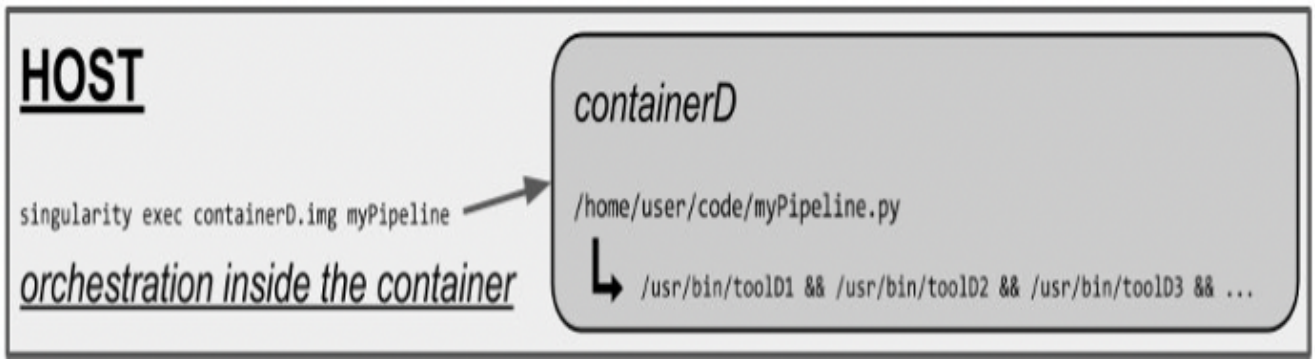
The pipeline is run on the host via a bash script. Each step is calling a tool located in one of the two containers using the **"singularity exec"** command.

- **A 2-step pipeline is encapsulated in a container.**



The **ENTRYPOINT** of the container defines the steps to be executed when the container is called using "**singularity run**".

- **A 3-step pipeline is encapsulated in a container.**



A script defining the execution steps (**python** / **bash** / **snakemake** / ...) is called from outside using the "**singularity exec**" command. If the script is mounted inside the container, it can be easily changed from outside without recreating the container.

Example Workflow

A guide on how to use a BWA singularity container:

```
$ singularity exec tools.sif bwa
Program: bwa (alignment via Burrows-Wheeler transformation)
Version: 0.7.17-r1188

$ singularity exec ./tools.sif bwa mem testbwa/ref.fa testbwa/*/*.fastq.gz > output/Mapped_reads.bwa.sam

# to submit to Juno cluster
$ bsub singularity exec ./tools.sif bwa mem testbwa/ref.fa testbwa/*/*.fastq.gz > output/Mapped_reads.bwa.sam
Or
bsub < bwabsub.sh
```

Demonstration (click to download and view):

Additional Help

For a comprehensive overview of Singularity, please refer to the Singularity user guide here. The help menu will provide the most up-to-date help as shown below:

Singularity Help

```
$ singularity --help

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

Usage:
  singularity [global options...]

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

Options:
  -c, --config string   specify a configuration file (for root or
                        unprivileged installation only) (default
                        "/opt/local/singularity/3.7.1/etc/singularity/singularity.conf")
  -d, --debug           print debugging information (highest verbosity)
  -h, --help            help for singularity
      --nocolor         print without color output (default False)
  -q, --quiet           suppress normal output
  -s, --silent          only print errors
  -v, --verbose         print additional information
      --version         version for singularity

Available Commands:
  build       Build a Singularity image
  cache       Manage the local cache
  capability  Manage Linux capabilities for users and groups
  config      Manage various singularity configuration (root user only)
  delete      Deletes requested image from the library
  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 Singularity plugins
  pull        Pull an image from a URI
  push        Upload image to the provided URI
  remote      Manage singularity remote endpoints, keyservers and OCI/Docker registry credentials
  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 digital signature(s) 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 Singularity

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


For additional help or support, please visit https://www.sylabs.io/docs/
```

User Group Sessions

Click here to view the Singularity presentation - Juno user group meeting on March 21, 2019

Singularity (Version 2.2) is installed as a RPM on the lilac cluster and does not need any additional modules to be loaded. Singularity containers are each stored in a single file, unlike Docker containers, for mobility and reproducibility.

Websites

http://singularity.lbl.gov/#home

https://github.com/singularityware/singularity