Containers

Shipping containers have frequently been used as an analogy for computing containers because the container is standard, it does not care what is put inside, and it will be carried on any ship, or in the case of computing containers can run on many systems different from the one it was created on. Hence the logo for Docker, the most common container platform.

Docker is widely used by researchers for reasons we won’t get into here. See their documentation

Docker images cannot be run in a HPC environment. This has to do with the privileges required to run a Docker container. This issue is addressed with Singularity. It is a container technology that completely contains the authority so that when the image is run all privileges stay inside the container. This makes it ideal for the shared environment of a supercomputer. And even better is that a Docker image can be encapsulated inside a Singularity image. So the documentation here instructs how to take either a Docker image and run it from Singularity or create an image using Singularity only.

Singularity Overview

Singularity enables users to have full control of their environment. Singularity containers can be used to package entire scientific workflows, software and libraries, and even data. This means that you don’t have to ask your cluster admin to install anything for you - you can put it in a Singularity container and run. Did you already invest in Docker? The Singularity software can import your Docker images without having Docker installed or being a superuser. Need to share your code? Put it in a Singularity container and your collaborator won’t have to go through the pain of installing missing dependencies. Do you need to run a different operating system entirely? You can “swap out” the

- Singularity Overview
- Singularity Commands
- Singularity Changes in Version 3.0 - Not Implemented Yet
- Singularity Hub
- Singularity Tutorials
- Singularity, Nvidia and GPU’s
- Singularity, Python and Machine Learning
operating system on your host for a different one within a Singularity container. As the user, you are in control of the extent to which your container interacts with its host. There can be seamless integration, or little to no communication at all.

Here are some of the use cases we support using Singularity:

- You already use Docker and want to run your jobs on HPC
- You want to preserve your environment so that a system change will not affect your work
- You need newer or different libraries than are offered on HPC systems
- Someone else developed the workflow using a different version of Linux
- You prefer to use something other than Red Hat / CentOS, like Ubuntu

**Singularity Commands**

```
$ singularity --help
```

Linux container platform optimized for High Performance Computing (HPC)

**Usage:**

```
singularity [global options...]
```

**Description:**

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

**Options:**

- `-d, --debug` print debugging information (highest verbosity)
- `-h, --help` help for singularity
- `-q, --quiet` suppress normal output
- `-s, --silent` only print errors
- `-t, --tokenfile string` path to the file holding your sylabs authentication token (default "/home/u13/chrisreidy/.singularity/sylabs-token")
- `-v, --verbose` print additional information
- `--version` version for singularity

**Available Commands:**

- `build` Build a new Singularity container
- `capability` Manage Linux capabilities on containers
- `exec` Execute a command within container
- `help` Help about any command
- `inspect` Display metadata for container if available
- `instance` Manage containers running in the background
- `keys` Manage OpenPGP key stores
- `pull` Pull a container from a URI
- `push` Push a container to a Library URI
- `run` Launch a runscript within container
run-help  Display help for container if available
search  Search the library
shell  Run a Bourne shell within container
sign  Attach cryptographic signatures to container
test  Run defined tests for this particular container
verify  Verify cryptographic signatures on container
version  Show application version

Examples:

$ singularity help <command>

Additional help for any Singularity subcommand can be seen by appending
the subcommand name to the above command.

Singularity Changes in Version 3.0 - Not Implemented Yet

Overview of the new version of Singularity

For existing users, two of the biggest changes are; that the file type of images now is ".sif" for Singularity Image Format for cryptographically signed and verifiable container images; and that your container may not run outside your home directory unless you include binding to other directories like /extra

Singularity Hub

Singularity Hub lets you build and keep containers at their Hub. You maintain your recipes there and each time you need to pull one, it gets built there and then you retrieve the container.

This is very convenient for the scenario where you do not have access to root authority to build the container. The build takes place through the Hub.

This also lets you share containers  

Singularity Tutorials

There are tutorials located here

Singularity, Nvidia and GPU's

One of the most significant use cases for Singularity is to support machine learning workflows. The details are in the GPU section.

Singularity, Python and Machine Learning

More tutorials