# Singularity v3.5

Paul Preney, OCT, M.Sc., B.Ed., B.Sc.
preney@sharcnet.ca

School of Computer Science
University of Windsor
Windsor, Ontario, Canada



Jan. 29, 2020

# Table of Contents

# What is Singularity?

Singularity:

- is container software originally created by Berkeley Lab, now developed by **Sylabs.io**,
- uses a **secure, single-file** based container format,
  - NOTE: Container files can also be encrypted.
- is a **secure way** to use Linux **containers** on Linux **multi-user clusters**,
- as a way to **enable** users to have **full control of their environment**, and,
- as a way to **package scientific software** and **deploy that package** to **different clusters** having the same architecture.

**URL:** `https://sylabs.io`

# What is Singularity?

Singularity provides **operating-system-level virtualization** called **containers**.

A **container** is different from a **virtual machine**:

- containers have **less overhead**, and,
- can only run the **same operating system** inside the container.

# What is Singularity?

A container uses Linux **control groups** (cgroups), kernel **namespaces**, and an **overlay filesystem**:

- cgroups **limit, control, and isolate** resource usage (e.g., RAM, disk I/O, CPU),
- kernel namespaces **virtualize and isolate** operating system resources of a **group of processes** (e.g., process and user IDs, filesystems, network access), and,
- overlay filesystems enable the **appearance of writing** to an **underlying read-only** filesystem.

# What is Singularity?

Singularity was designed to enable containers to be used **securely without requiring any special permissions**.

- Singularity is the only container technology currently supported on Compute Canada systems.

**Docker**, another container technology, to run programs can require or provide potential access to **elevated permissions** leading to security concerns.

- If you want to use a Docker images on Compute Canada systems, you must do so via Singularity.

# Table of Contents

# Installing Singularity

Singularity is **already** installed on Compute Canada systems.

You can **install** Singularity on your own computer provided it is running **Linux** (natively or in a virtual machine).

- Installation instructions are available in the Singularity **User Guide's Quick Start** and **Administration Guide's Installation** documents.
  - e.g., `https://sylabs.io/guides/3.5/user-guide/quick_start.html`
  - e.g., `https://sylabs.io/guides/3.5/admin-guide/installation.html`

# Table of Contents

# Loading Singularity

To use Singularity on Compute Canada systems:

- If your container programs use OpenMPI, then load a version 3 or higher module:
  - e.g., `module load openmpi/3.1.2`
  - e.g., `module load openmpi/4.0.1`
- Then load the Singularity module:
  - e.g., `module load singularity/3.5`

# Loading Singularity (con't)

If using OpenMPI:

- In the Slurm job script use `srun` —not `mpirun` or `mpiexec`— to run your command, e.g.,
  - `srun singularity exec image.sif /path/to/your-program`
- Ensure no `module load` commands are in your Slurm script.
- Ensure **all executables** being **run inside** the container are properly built **inside** the singularity container.

# Table of Contents

# Images

**Before** using Singularity, you need to **create an image**.

A Singularity **image** is either a file or a directory **containing an installation of Linux**.

# Images (con't)

Singularity allows one to **create images** by:

- downloading a container hosted on **Docker Hub**, **Singularity Hub**, or a **Library**
  - Docker Hub: `https://hub.docker.com/`
  - Singularity Hub: `https://singularity-hub.org/`
  - Library defaults to `https://cloud.sylabs.io/library`
- using a container image file you already have,
  - `*.sif` Singularity Image Format (SIF); (Version v3.0+)
  - `*.sqsh` SquashFS format. (Version 2.4+)
  - `*.img` ext3 format. (Oldest format)
- using a "sandbox" **directory**, or,
- using a **Singularity recipe file**.

# Images (con't)

For example, to build the latest base 64-bit Debian image from Docker Hub:

- **SIF File:** `singularity build debian-from-dhub.sif docker://debian`
- **Sandbox Directory:** `singularity build -s debian-from-dhub docker://debian`

**Problem:** The result has all permissions "flattened" to that of your account.

- Commands requiring privileged access such as `apt`, `apt-get`, and `dpkg` will **not** work.

# Images (con't)

**Solution 1:** Build the desired image on a Linux machine where you have superuser (root) access via `sudo`.

- **SIF File:** `sudo singularity build debian-from-dhub.sif docker://debian`
- **Sandbox Directory:** `sudo singularity build -s debian-from-dhub docker://debian`

**Solution 2:** Submit a ticket requesting help from a Compute Canada person to build the image.

**Solution 3:** There are some Linux distributions, e.g., Gentoo, that can tolerate not having privileged access to install/remove software in a "permission flattened" container.

# Table of Contents

# Using Singularity: Overview

There are a number of ways to use Singularity:

1. Run a **single command** which executes and then stops running.
2. Run **many commands** in an interactive session.
3. Run a container instance to run **daemons** and have **backgrounded processes**.
    - No hung processes: everything is killed when the Singularity instance is stopped/killed!

# Running Commands

Given a container `image.sif` with `gcc` installed in it, one can check the version of `gcc` used with the **exec** command:

```
singularity exec image.sif gcc -v
```

# Running Commands (con't)

One can interactively use the container with the **shell** command:

```
singularity shell image.sif
```

To exit the container type `exit`.

# Running Commands (con't)

If one needs to run backgrounded and daemon processes, use the **instance start** and **instance stop** commands.

By running **instance start** and with a name, e.g., `myjob`, Singularity will start a new container instance:

```
singularity instance start image.sif myjob
```

# Running Commands (con't)

Container instances can be queried using **instance list**, e.g.,

```
singularity instance list
```

which will list the **daemon name**, its **PID**, and the **container image path**.

# Running Commands (con't)

Programs can be run using **exec** or **shell** as before, except the name of the instance prefixed with **instance://** and must also be specified, e.g.,

```
singularity instance start image.sif one
singularity exec image.sif instance://one ps -eaf
singularity shell image.sif instance://one
  nohup find / -type d >dump.txt
  exit
singularity exec image.sif instance://one ps -eaf
```

# Running Commands (con't)

An instance is **shut down** stopping all daemons, background processes, etc. by running the **instance stop** command, e.g.,

```
singularity instance stop image.sif myjob
singularity instance stop image.sif one
```

# Bind Mounts

Programs running inside a Singularity container **will not have access to files and directories** outside of your home and the current directory (when `singularity` was run).

Typically one would like to be able to access files and directories under:

- `/home`
- `/project`
- `/scratch`
- `/localscratch`

which can be done using the `-B`, **bind mount**, option when running `singularity`:

```
singularity shell -B /home -B /project -B /scratch -B /localscratch image.sif
singularity exec -B /home -B /project -B /scratch -B /localscratch \
  image.sif gcc /project/$USER/p.c
```

# Bind Mounts (con't)

The actual locations of all mounts can be made to **appear at a different location** *inside* the container, e.g.,

```
mkdir /localscratch/tmp/$USER
singularity shell -B /localscratch/tmp/$USER:/tmp image.sif
```

i.e., inside the container `/tmp` is actually the directory:

```
/localscratch/tmp/$USER
```

outside the container.

# Table of Contents

# Demonstration and Questions

Demonstration and questions.