# Singularity

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

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

Copyright © 2018 Paul Preney. All Rights Reserved.

Feb. 14, 2018

# Table of Contents

# What is Singularity?

Singularity is **open source** software created by **Berkeley Lab**:

- as 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:** `http://singularity.lbl.gov`

# 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?

For emphasis…

Singularity was **specifically designed** to enable containers to be used **securely without requiring any special permissions** especially on multi-user **compute clusters**.

Other container technologies, e.g., **Docker**, at times require various **elevated permissions** each with associated **security risks**.

For more information see:
`http://singularity.lbl.gov/docs-security`.

# Table of Contents

# Installing Singularity

Singularity is **already** installed on:

- `graham.computecanada.ca`,
- `cedar.computecanada.ca`, as well as,
- some legacy clusters such as `orca.sharcnet.ca`.

# Installing Singularity (con't)

You can install Singularity on your **own computer** as long as it is running **Linux** with a reasonably recent kernel version, e.g.,

- CentOS 6: 2.6.32 kernel (very old). Works with limitations.
- CentOS 7: 3.10.0 kernel. Works.

Instructions can be found on the Singularity web site: `http://singularity.lbl.gov`

# Table of Contents

# Loading Singularity

To use Singularity on `graham.computecanada.ca` or `cedar.computecanada.ca` use one of these module commands:

- `module load singularity/2.4`
- `module load singularity/2.3`

**NOTE:** This presentation only discusses Singularity v2.4.

# Loading Singularity (con't)

To use Singularity on `orca.sharcnet.ca` use one of these commands:

- `export PATH=/opt/sharcnet/singularity/2.4.0/bin:$PATH`
- `module load singularity/2.3.1`

**NOTE:** This presentation only discusses Singularity v2.4.

# Table of Contents

# Creating Images: Overview

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

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

# Different Ways To Create A Singularity Image

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

- downloading a container from **Singularity Hub**,
- downloading a container from **Docker Hub**,
- from a container you already have,
- from a **tarball** or a **directory**, or,
- from a **Singularity recipe file**.

# Different Ways To Create A Singularity Image (con't)

Suppose the Singularity Hub URL for a container you want is:

```
shub://singularityhub/ubuntu
```

then you would download that container by running:

```
singularity pull shub://singularityhub/ubuntu
```

**Singularity Hub URL:** `https://singularity-hub.org/`
Only use images from Singularity Hub if you trust those images!

# Different Ways To Create A Singularity Image (con't)

Suppose the Docker Hub URL for a container you want is:

```
docker://ubuntu
```

then you would download that container by running:

```
singularity pull docker://ubuntu
```

**Docker Hub URL:** `https://hub.docker.com/`
Only use images from Docker Hub if you trust those images!

# Manually Creating an Image Based On Your System

If you already have a **configured Intel 64-bit version of Linux** with **all needed software installed**, then start by **making a tarball** of your system, e.g.,

```
sudo tar -cvpf -C / mysystem.tar \
  --exclude=/dev --exclude=/proc --exclude=/sys
```

# Manually Creating an Image Based On Your System (con't)

**Notice**

All of the remaining steps all **must be done** on a system with **Singularity installed**.

If you **need to preserve** file and directory **permissions**, you **must** use a system, such as your own computer, where you have **root** access!

- e.g., your Linux distribution may require you to preserve permissions if you want to upgrade or install new software into the image later

If you **don't need to preserve permissions**, you can run the commands on the following slides on `graham`, `cedar`, etc. —simply omit **sudo** when typing in the commands.

# Manually Creating an Image Based On Your System (con't)

**Optional:** Untar the tarball in an empty directory:

```
sudo mkdir workdir
cd workdir
sudo tar -xvf mysystem.tar
cd ..
```

and use the interactive **shell** command to examine, delete, rename, files, directories, etc. as is appropriate for this image:

```
sudo singularity shell -w workdir
bash
```

When done, run the command `exit` twice to leave the interactive shell.

# Manually Creating an Image Based On Your System (con't)

Finally, create the image using the **build** command from your `workdir`:

```
sudo singularity build myimage.simg workdir
```

or from the original tarball:

```
sudo singularity build myimage.simg mysystem.tar
```

and transfer the read-only Singularity image, `myimage.simg`, to a cluster with Singularity installed. :-)

# Creating an Image Using A Recipe

The following Singularity recipe file:

copy-drive-into-container-recipe

```
1 Bootstrap: self
2 Exclude: /boot /dev /home /lost+found /media /mnt /opt /proc /run /sys
```

can be used to convert one's system directly into a container with:

```
sudo singularity build self.simg copy-drive-into-container-recipe
```

# Creating an Image Using A Recipe (con't)

The following Singularity recipe file:

update-existing-container-recipe

```
1  Bootstrap: localimage
2  From: ubuntu-16.04-x86_64.simg
3
4  %help
5  This is a modified Ubuntu 16.06 x86_64 Singularity container image.
6
7  %post
8    sudo apt-get -y update
9    sudo apt-get -y upgrade
10   sudo apt-get -y install build-essential git
11   sudo apt-get -y install python-dev python-pip python-virtualenv python-numpy python-matplotlib
12   sudo apt-get -y install vim
13   sudo apt-get clean
```

can be used to update a pre-existing Ubuntu Singularity image with this:

```
sudo singularity build new-ubuntu-image.simg update-existing-container-recipe
```

# 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.simg` with `gcc` installed in it, one can check the version of `gcc` used with the **exec** command:

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

# Running Commands (con't)

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

```
singularity shell image.simg
```

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., `session5`, Singularity will start a new container instance:

```
singularity instance.start image.simg session5
```

# 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.simg one
singularity exec image.simg instance://one ps -eaf
singularity shell image.simg instance://one
  nohup find / -type d >dump.txt
  exit
singularity exec image.simg 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.simg session5
singularity instance.stop image.simg one
```

# Bind Mounts

Per Compute Canada cluster configuration, Singularity has been set to automatically mount your `/home` directory.

Other mounts containing your files must be **manually specified** using one or more **`-B`** options to **exec** and **shell** commands.

# Bind Mounts (con't)

On Compute Canada configured systems, one would normally want to mount:

- /project
- /scratch
- /localscratch

e.g.,

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

# Bind Mounts (con't)

On SHARCNET legacy systems, one would normally want to mount:

- `/work`
- `/scratch`

e.g.,

```
singularity shell -B /work -B /scratch image.simg
singularity exec -B /work -B /scratch image.simg gcc /work/$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.simg
```

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

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

outside the container.

# Table of Contents

# HPC Issues With Singularity

If you are running software that needs access to the NVIDIA GPUs on the node, pass the `--nv` to the Singularity **exec** option, e.g.,

```
singularity exec --nv image.simg python3 ./tensorflow/label_image.py
```

# HPC Issues With Singularity (con't)

Message Passing Interface (MPI):

- Work has been done to integrate Singularity with OpenMPI v2.1.x.
- Running an MPI installation **across nodes** requires either:
  - installing and configuring OpenMPI v2.1.x with **all required networking fabrics inside the container**, or,
  - **bind-mounting** the system's OpenMPI and network fabric libraries **into the container**.
    - **NOTE:** There may be various incompatibility issues with library and glibc versions being used with this method.

Additional information: `http://singularity.lbl.gov/docs-hpc`.

# Table of Contents

# Thank You and Questions

Thank you and questions. :-)