Singularity Containers

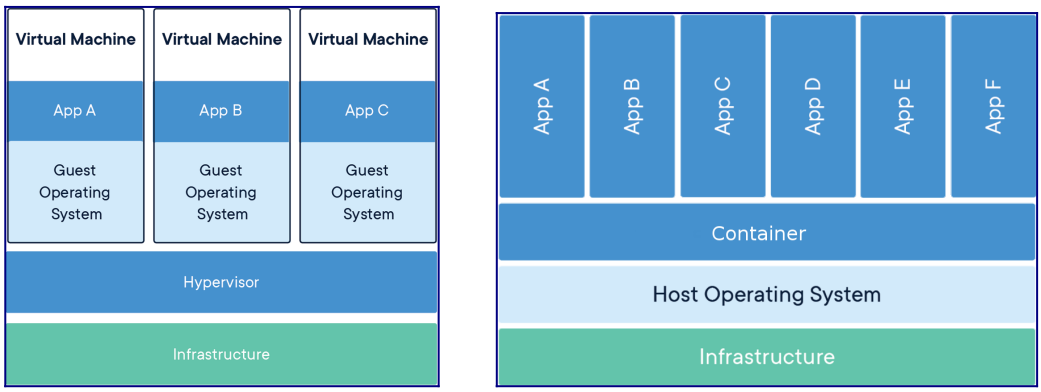
Material based on:

- https://singularity-docs.readthedocs.io
- https://singularity.lbl.gov/

Table of Contents

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 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.

What are containers



Containers: How are they useful

- Reproducibility
- Portability
- Depending on application and use-case, simple extreme scalability
- Next logical progression from virtual machines

Why do we want containers in HPC?

- Escape "dependency hell"
- Load fewer modules
- Local and remote code works identically every time
- One file contains everything and can be moved anywhere

Docker, the most popular container

Besides singularity images, singularity can also execute docker images.

- Docker – the most know and utilized container software
- Facilities workflow for creating, maintaining and distributing software
- Easy to install, well documented, standardized
- Used by many scientist

Docker on HPC: The problem

- Incompatibilities with scheduling managers (SLURM...)
- No support for MPI

- No native GPU support
- Docker users can escalate to root access on the cluster
- (Not allowed on HPC clusters)

Singularity: Containers for the HPC environment

- Package software and dependencies in one file
- Use same container in different SNIC clusters
- Limits user's privileges, better security
- Same user inside container as on host
- No need for most modules
- **Negligible performance decrease**

But I want to keep using docker

- Works great for local and private resources.
- No HPC center will install docker for you
- **Singularity can import Docker images**

Singularity workflow

Local computer (ROOT)
Create container (singularity build)
Install software
Install libraries

HPC cluster (USER)
(singularity shell)
(singularity exec)
(singularity help)
(singularity run)

Install singularity in Linux

- Install dependencies

```
build-essential libssl-dev uuid-dev libgpgme11-dev
squashfs-tools libseccomp-dev wget pkg-config git
```

- Install go (version 1.12.6, 2019-06-19)
- Install singularity (version 3.2.1-1, 2019-06-18)
- Follow instructions at https://www.sylabs.io/guides/3.2/user-guide/installation.html
- (Singularity cannot be installed on Mac or Windows, but can be used via VMs)

Launching a container

- Singularity sets up the container environment and creates the necessary namespaces.
- Directories, files and other resources are shared from the host into the container.
- All expected I/O is passed through the container: pipes, program arguments, std, X11
- When the application(s) finish their foreground execution process, the container and namespaces collapse and vanish cleanly

Using the container

Download and test an image

Download and test the latest UBUNTU image from docker hub

```
$ sudo singularity build my_image.sif docker://ubuntu:latest
INFO:    Starting build...
Getting image source signatures
...
Writing manifest to image destination
Storing signatures
INFO:    Creating SIF file...
INFO:    Build complete: my_image.sif
$ singularity shell my_image.sif
Singularity my_image.sif:~> cat /etc/*-release
Singularity my_image.sif:~> exit
```

Singularity hub

https://singularity-hub.org/

Example:

Go to singularity hub and find the hello-world container (https://singularity-hub.org/collections)

- build the container using singularity (shub://[full name of container])
- Use the container shell and get acquainted with it

```
$ singularity build my_image.sif shub://vsoch/hello-world
$ singularity shell my_image.sif
$ singularity run my_image.sif
```

How to build containers

Why must I be root?

Same permissions in the container as outside...

To be root in the singularity image you must be root on the computer

Build a writable image

Since there are memory limitation on writing directly to image file, it is better to create a sandbox

```
$ sudo singularity build --sandbox my_sandbox my_image.sif
INFO:    Starting build...
INFO:    Creating sandbox directory...
INFO:    Build complete: my_sandbox
$ sudo singularity shell -w my_sandbox
Singularity: Invoking an interactive shell within container...
Singularity my_sandbox:~>
```

How do I execute commands in singularity

Commands in the container can be given as normal.

```
singularity exec my_image.sif ls
```

```
$ singularity shell my_image.sif
Singularity: Invoking an interactive shell within container...
Singularity my_image.sif:~> ls
```

Transfer files into container

Read mode: You can read/write to file system outside container and read inside container.

write mode: You can read/write inside container.

(**Remember:** In write mode you are user ROOT, home folder: /root)

How to transfer files into the container

my_folder is a local folder on your computer

```
$ sudo singularity exec -w my_sandbox mkdir singularity_folder
$ sudo singularity shell -B my_folder:/root/singularity_folder -w my_sandbox
Singularity my_sandbox:~> cp singularity_folder/file1 .
```

How to execute scripts outside image

- All the dependencies in image
- Script outside image

```
$ singularity exec -B my_folder:/root/singularity_folder my_sandbox
singularity_folder/script
```

Create your own container

- Go to docker hub and find the official latest ubuntu
- build the container using singularity
- Build a writeable sandbox
- Install necessary tools into the container (Compiler etc...)
    - apt-get update
    - apt-get install build-essential

```
$ sudo singularity build my_image.sif docker://ubuntu:latest
$ sudo singularity build --sandbox my_sandbox my_image.sif
$ sudo singularity shell -w my_sandbox
Singularity my_sandbox:~> apt-get update
Singularity my_sandbox:~> apt-get install build-essential
```

singularity.d folder

Startup scripts etc... for your singularity image

```
$ singularity exec my_image.sif ls -l /.singularity.d
total 7
-rw-r--r-- 1 root root   39 Jun 18 11:26 Singularity
drwxr-xr-x 2 root root 4096 Jun 18 11:22 actions
drwxr-xr-x 2 root root  173 Jun 18 11:26 env
```

```
-rw-r--r-- 1 root root  309 Jun 18 11:26 labels.json
drwxr-xr-x 2 root root    3 Jun 18 11:26 libs
-rwxr-xr-x 1 root root 1084 Jun 18 11:26 runscript
-rwxr-xr-x 1 root root   19 Jun 18 11:26 runscript.help
-rwxr-xr-x 1 root root   10 Jun 18 11:26 startscript
```

Important: Scripts must be executable and owned by root

Creating a script

runscript

```
#!/bin/sh

ls -l
```

command

```
$ singularity run my_image.sif
total 1
drwxr-xr-x 2 root root  76 Sep 11 17:05 file1
drwxr-xr-x 2 root root 139 Sep 11 17:23 file2
```

What is a help file and how is it used

runscript.help

```
This is a text file
```

command

```
$ singularity run-help my_image.sif
This is a text file
```

Build a new container from a sandbox

```
$ sudo singularity build my_new_image.sif my_sandbox
INFO:    Starting build...
INFO:    Creating SIF file...
INFO:    Build complete: my_new_image.sif
```

Edit your own container

- Create a help file
- Create/Edit the runscript running hello world
- Create a new container from the sandbox

Tip: You can use the editor in your VM and then transfer the file

```
$ sudo singularity shell -w my_sandbox
Singularity my_sandbox:~> echo "This is my help file" >
/.singularity.d/runscript.help
Singularity my_sandbox:~> exit
$ sudo singularity build my_new_image.sif my_sandbox
$ singularity run-help my_new_image.sif
```

Running your container in an HPC environment

Requirements

- OpenMPI version must be the same in container and cluster
- Compiler and version must be the same in container and cluster
- You need to bind to the Lustre file system so it can be detected

What are the required tools

wget, build-essential, lzip, m4, libgfortran3, gmp, mpfr, mpc, zlib, gcc, openmpi, cmake, python, cuda

Send in a singularity batch job and execute

```
#!/bin/bash -l
#SBATCH -J myjob
#SBATCH -t 1:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH -o output_file.o
module add gcc/6.2.0 openmpi/3.0-gcc-6.2 singularity
mpirun -n 8 singularity exec -B /cfs/data hello_world.sif hello_world_mpi
```

Run an HPC container

- Login into sol-login.fysik.su.se
- send in a job for the hello-world image
- Use the hello_world image on Fyikum’s singularity repository

(**Tip:** With the singularity module use the **Path:** $FYSIKUM_SHUB)

```
#!/bin/bash -l
#SBATCH -J myjob
#SBATCH -A <Allocation ID>
#SBATCH --reservation=<Reservation ID>
#SBATCH -t 1:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
#SBATCH -o output_file.o
module add gcc/6.2.0 openmpi/3.0-gcc-6.2 singularity
mpirun -n 8 singularity exec -B /cfs/data $FYSIKUM_SHUB/hello_world.sif
hello_world_mpi
```

Singularity Recipes

A Singularity Recipe is the driver of a custom build, and the starting point for designing any custom container. It includes specifics about installation software, environment variables, files to add, and container metadata.

Singularity recipes is also a good practice for keeping your image up-to-date

How to build from a recipe

A recipe is a textfile explaining what should be put into the container

```
sudo singularity build my_image.sif my_recipe.def
```

Recipe format

```
# Header
Bootstrap: docker
From: ubuntu:latest
# Sections
%help
  Help me. I'm in the container.
%files
    mydata.txt /home
%post
    apt-get -y update
    apt-get install -y build-essential
%runscript
    echo "This is my runscript"
```

Header

What image should we start with?

- *Bootstrap:*
    - shub
    - docker
    - localimage
- *From:*
    - The name of the container

```
# Header
Bootstrap: docker
From: ubuntu:latest
```

Section: %help

Some information about your container. Valuable to put information about what software and versions are available in the container

```
%help
  This container is based on UBUNTU 16.04. GCC v6.2 installed
```

Section: %post

What softwares should be installed in my container.

```
%post
    apt-get -y update
    apt-get install -y build-essential
```

(No interaction in the scripts)
(We do not need sudo in the container)

Section: %files

What local files should be copied into my container

```
%files
    # <filename> <singularity path>
    myfile.txt /opt
```

Section: %runscript

What should be executed with the run command.

```
%runscript
    mysoftware -param1 -param2
```

Create a recipe

- Based on UBUNTU
- Install compilers
- Create a help text
- Create a runscript
- Run the recipe

(**Tip:** You can use the editor in your VM and then transfer the file)

my_recipy.rec

```
# Header
Bootstrap: docker
From: ubuntu:latest
# Sections
%help
  Help me. I'm in the container.
%runscript
  echo "This is my runscript"
%post
  apt-get -y update
  apt-get install -y build-essential
```

Building:

```
sudo singularity build my_new_image.sif my_recipy.rec
```