Introduction to Singularity

Henric Zazzi

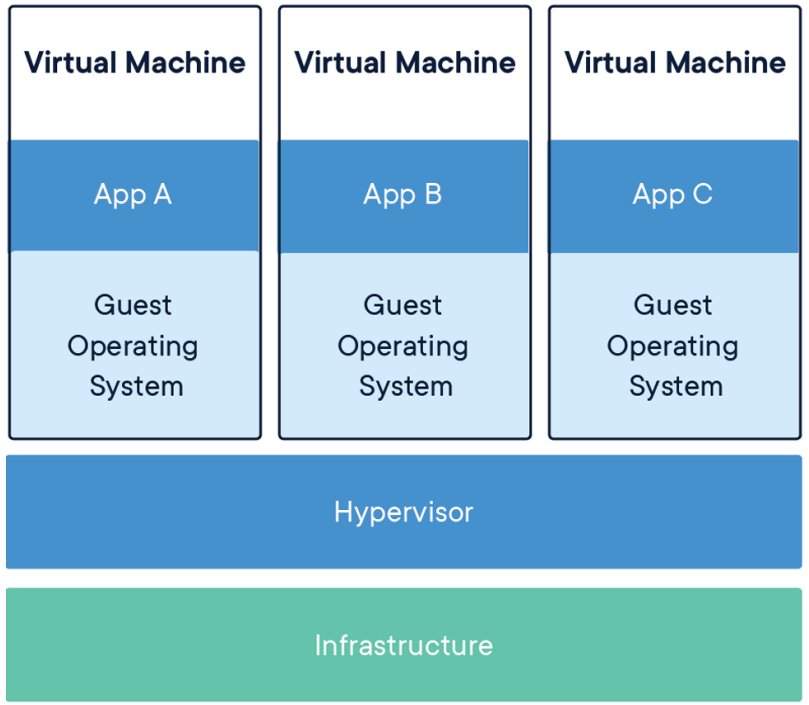
Overview

- What are containers
- Docker, the most popular container
- Singularity: Containers for the HPC environment
- installation of singularity
- How to build containers
- Installing software in container
- Running your container in an HPC environment
- Creating recipes for singularity

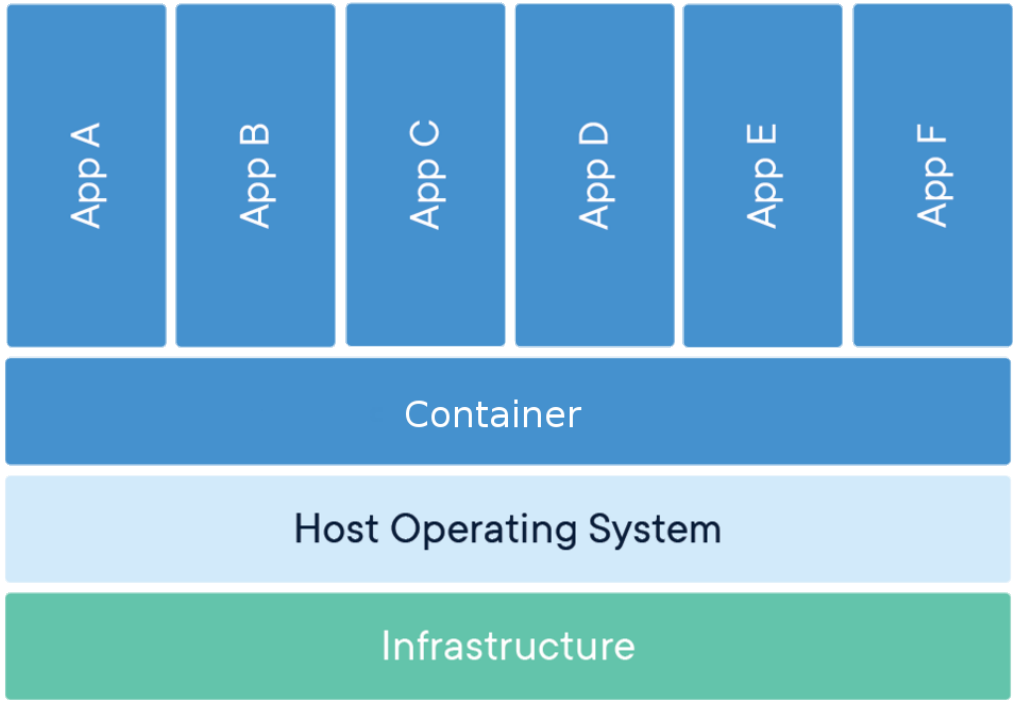
What are containers

A container image is a lightweight, standalone, executable package of software that includes everything needed to run an application.

Virtual Machine



Container



Containers: How are they useful

- Reproducibility
- Portability
- Depending on application and use-case, simple extreme scalability
- Next logical progession 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

The Docker container software

- The most know and utilized container software
- Facilites workflow for creating, maintaining and distributing software
- Easy to install, well documented, standardized
- Used by many scientist

Docker on HPC: The problem

- Incompabilities 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
- Negligable performance decrease

But I want to keep using docker

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

Singularity hub

https://singularity-hub.org/

Container Collections

Want to build containers?

LOGIN

One collection is created for each connected Github repository. In that collection, several containers will automatically be built: one for each uniquely named recipe file found in the master branch of the Github repository.
Read more about recipe file naming or build options.

Enter Keywords Here

Name	Builds	Description	Stars	Downloads	Last Modified
researchapps/quantum_state_diffusion	109	Solving quantum state diffusion numerically.	1	22	2017-10-19
vsoch/singularity-hello-world	102		1	9098	2018-03-02
vsoch/pe-predictive	100	run pe-finder to predict pumonary embolism from radiology reports	0	15	2017-10-17
TomHarrop/singularity-containers	27		0	157	2018-09-10
dominik-handler/AP_singu	16		0	621	2018-09-06
QuentinLetourneur/Let-it-bin	15	Optimize workflow for binning metagenomic short reads from multiple samples	0	2	2018-09-10
marcc-hpc/tensorflow	13	Singularity file wrapping Dockerfile of tensorflow/tensorflow:latest-gpu	1	4417	2018-09-09

Singularity Versions

singularityCE (Community Edition)

- Latest version: 3.11.0 (2023-02-17)
- Installed on Dardel: 3.10.4

Singularity workflow

Local computer

Root access

1. Create container
2. *singularity build*
3. Install software
4. Install libraries

HPC Cluster

User access

1. *singularity shell*
2. *singularity exec*
3. *singularity help*
4. *singularity run*

Install singularity on your computer

You need a local installed copy of singularity to create your container

```
$ sudo apt-get update && sudo apt-get install -y \
    build-essential libssl-dev uuid-dev libgpgme11-dev \
    squashfs-tools libseccomp-dev pkg-config \
    libglib2.0-dev
# Google GO language is needed
$ sudo apt install golang
$ git clone --recurse-submodules https://github.com/sylabs/singularity/
$ cd singularity
$ ./mconfig && make -C ./builddir && sudo make -C ./builddir install
```

More information at ...

https://docs.sylabs.io/guides/3.0/user-guide/installation.html

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

How to build containers

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
...
INFO:    Creating SIF file...
INFO:    Build complete: my_image.sif
$ singularity shell my_image.sif
Singularity> cat /etc/*-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=22.04
DISTRIB_CODENAME=jammy
Singularity> exit
```

Paths for building containers

```
singularity build [image].sif [name]
```

From	Path
Singularity hub	shub://[name]
Docker hub	docker://[name]
Local	[name]
Sandbox	[Sandox folder name]
Recipe	[recipe name]

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> ls
```

Do it yourself

Exercise 1: Download a container

1. Go to singularity hub and find the hello-world container (https://singularityhub.github.io/singularityhub-archive/)
2. build the container using singularity
3. Use the container shell and get acquainted with it

Installing software in container

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 writeable 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:    Verifying bootstrap image my_image.sif
...
INFO:    Creating sandbox directory...
INFO:    Build complete: my_sandbox
$ singularity shell my_sandbox
Singularity>
```

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

Binding folders

```
... -B [local folder]:[singularity folder] ...
```

- Enables transferring of data to container
- Enables accessing external data from within the container
- Enables running external scripts from within the container
- For using PDC filesystem you must bind to *cfs/klemming*

How to use binding to run local scripts

1. Create local folder and internal singularity folder

```
$ mkdir my_folder
$ sudo singularity exec -w my_sandbox mkdir /usr/local/sing
```

2. Starting container and bind folders

   The file *myscript*, residing in my_folder, will be executed as within the container

```
$ singularity exec -B my_folder:/usr/local/sing -w my_sandbox /usr/local/sing/myscript
```

3. Obscuring container folder */opt*

```
$ singularity exec -B my_folder:/opt my_sandbox /opt/myscript
```

Example on how to transfer files into the container

1. Create local folder and internal singularity folder

```
$ mkdir my_folder
$ sudo singularity exec -w my_sandbox mkdir singularity_folder
```

2. Starting container as *root* and bind folders

```
$ sudo singularity shell -B my_folder:/root/singularity_folder -w my_sandbox
```

3. Copy *file1* to container folder

```
Singularity> cp singularity_folder/file1 .
```

Do it yourself:

Exercise 2: Create your own container

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

Initiating your container

singularity.d folder

Startup scripts etc... for your singularity image

```
$ singularity exec my_image.sif ls -l /.singularity.d
-rw-r--r-- 1 root root   39 Feb 17 09:27 Singularity
drwxr-xr-x 2 root root 4096 Feb 17 09:27 actions
drwxr-xr-x 2 root root 4096 Feb 17 09:27 env
-rw-r--r-- 1 root root  459 Feb 17 09:27 labels.json
drwxr-xr-x 2 root root 4096 Feb 17 09:27 libs
-rwxr-xr-x 1 root root 1933 Feb 17 09:27 runscript
-rwxr-xr-x 1 root root 10   Feb 17 09:27 runscript.help
-rwxr-xr-x 1 root root   10 Feb 17 09:27 startscript
```

Important: The files must be executable and owned by root

Creating a script

Example of runscript file

```
#!/bin/sh
echo "Hello world!"
```

Executing the runscript file

```
$ singularity run my_image.sif
Hello world!
```

What is a help file and how is it used

Example of runscript.help file

```
This is a text file
```

Print the information within

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

Do it yourself:

Exercise 3: Edit your own container

1. Create a help file
2. Create/Edit the runscript printing *Hello world!*

Tip: You can use an editor in your VM or create it and then transfer the file

Creating recipes for singularity

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

How to build from a recipe

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

```
sudo singularity build [container].sif [recipe].def
```

Recipes for images that can be used on PDC clusters can be found at https://github.com/PDC-support/PDC-SoftwareStack/tree/master/other/singularity

Printing how a container was built

```
singularity inspect --deffile [container]
```

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 22.04. GCC 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
    <local filename> <singularity path>
```

Section: %runscript

What should be executed with the run command.

```
%runscript
    <software executable> -<parameter1>
```

Do it yourself:

Exercise 4: Create a recipe

1. Based on UBUNTU
2. Install compilers
3. Create a help text
4. Create a runscript
5. Run the recipe

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

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 at PDC so it can be detected
- You can use *build* but only from other images and only sandboxes
- You can **ONLY** run *sandbox* and not *SIF* files
  - A singularity file is copied to all nodes whereas a *sandbox* folder structure is not

Transfer of sandbox file

```
# On local computer
sudo tar czf <sandbox name>.tar.gz <sandbox name>
scp <sandbox name>.tar.gz <username>@dardel.pdc.kth.se:/cfs/klemming/home/<u>/<username>
# On Dardel
tar xfp <sandbox name>.tar.gz
```

Transfer a SIF file

```
scp <SIF file> <username>@dardel.pdc.kth.se:/cfs/klemming/home/<u>/<username>
singularity build --sandbox <sandbox name> <SIF file>
```

What are the required tools

- **Packages:** wget git bash gcc gfortran g++ make
- **Source:** MPICH

```
ml PDC
ml singularity
```

- In folder *$PDC_SHUB* you can find already built images at PDC

Executes singularity on 2 nodes

```
#!/bin/bash -l
# The -l above is required to get the full environment with modules
# Set the allocation to be charged for this job
#SBATCH -A 202X-X-XX
# The name of the script is myjob
#SBATCH -J myjob
# Only 1 hour wall-clock time will be given to this job
#SBATCH -t 1:00:00
# Number of nodes
#SBATCH --nodes=2
# Using the shared partition as we are not using all cores
#SBATCH -p shared
# Number of MPI processes per node
#SBATCH --ntasks-per-node=12
# Run the executable named myexe
ml PDC singularity
srun -n 24 --mpi=pmi2 singularity exec <sandbox folder> <myexe>
```

Executes GPU enabled code with singularity

```
#!/bin/bash -l
# The -l above is required to get the full environment with modules
# Set the allocation to be charged for this job
# not required if you have set a default allocation
#SBATCH -A 201X-X-XX
# The name of the script is myjob
#SBATCH -J myjob
# Only 1 hour wall-clock time will be given to this job
#SBATCH -t 1:00:00
# Number of nodes
#SBATCH --nodes=1
# Using the GPU partition which is at the moment is under testing
#SBATCH -p gpu-tst
# Run the executable named myexe
ml PDC singularity
srun -n 1 singularity exec --rocm -B /cfs/klemming <sandbox folder> <myexe>
```

Exercise 5: Run a HPC container

1. Login into dardel.pdc.kth.se
2. send in a job for the hello-world sandbox
3. Use the hello_world in PDCs singularity repository

Tip: With the singularity module use the **Path:** $PDC_SHUB

Useful links

- https://www.pdc.kth.se/support/documents/software/singularity.html
- https://github.com/PDC-support/PDC-SoftwareStack/tree/master/other/singularity
- https://sylabs.io/docs/