Containers in HPC: Singularity/ Apptainer

Grigory Shamov

May 21, 2026

University of Manitoba

What are containers and why they are popular

Containers are ways to encapsulate Software.

- Supposed to make software dependencies management easier.

Containers are Linux-specific tool of software isolation

- “Chroot” + “Linux namespaces” + runtime to run things + tools to manage things
- **Shares kernel** with the “host” Linux system: very little overhead
- Shares kernel with host, unlike Virtual Machines : bad security

The earliest and most popular container environment for long time was “**Docker**”.

Q: *Can I use Docker in your HPC environment?*

Another popular environment developed for HPC environments is/was Singularity.

The project since forked :

SingularityCE by Sylabs and **Apptainer** by the Linux Foundation

University of Manitoba

Software layers *(slide by Dr. Ali Kerrache)*

User layer: Python packages, Perl and R modules, home made codes, …

Software stacks: modules for Intel, PGI, OpenMPI, CUDA, MKL, high-level applications. Multiple architectures (avx2, avx512)

gentoo: GNU libc, autotools, make, bash, cat, ls, awk, grep, etc.

Gray area: Slurm, Lustre client libraries, IB/OmniPath/InfiniPath client libraries (all dependencies of OpenMPI) in Nix {or gentoo} layer, but can be overridden using PATH & LD_LIBRARY_PATH. Linux OS daemons

OS: Linux kernel, drivers.

User

Analysts

Sys. Admin

Glossary of Containers

- **Containerfile** - Recipe for building an image, including OS and software within the image. Usually a text file: Singularity *recipes*, *Dockerfiles* etc.
- **Image**- The result of building the recipe described in a Containerfile. Usually a form of archive (or number of archives i.e. “layers”) of a filesystem tree, or just the filesystem tree.
- **Container**- The running instance of an image. Can be a computing process, or a service daemon.
- **Container runtime**: A set of tools used for building and running containers, such as Docker, Podman, Singularity, Apptainer and many others
- **OCI (Open Container Initiative)**- common standard for container runtimes and container image formats. Podman and Docker are OCI-compliant, meaning their syntax is generally interchangeable.
- **Registry**- An online storage area for images. Typical examples are DockerHub or Quay.io.

Motivation for Singularity (Apptainer)

- Root access on shared HPC systems not possible for users
  - => no Docker, no **apt-get install** or **rpm install** or **dnf install**

- “Mobility of compute” : contain all the software dependencies in the container image

- Singularity was developed to run as a user, and as a regular process.
  - Mostly geared towards batch computing (a job starts and ends)
  - Can be used on shared filesystems like /project or /home
  - Can create container images from Docker images!
    - However, not every Docker image will work

- These days, rootless Podman and Pyxis provide alternatives for HPC, and Kubernetes provides rich orchestration capabilities Singularity lacks

Popular or current use cases

Singularity was invented for HPC (2015) when other engines required root

- Since then, rootless Podman and other container engines developed
- Singularity is focused on compute jobs
  - rather than multiple concurrent services where Docker/Podman/K8s shine

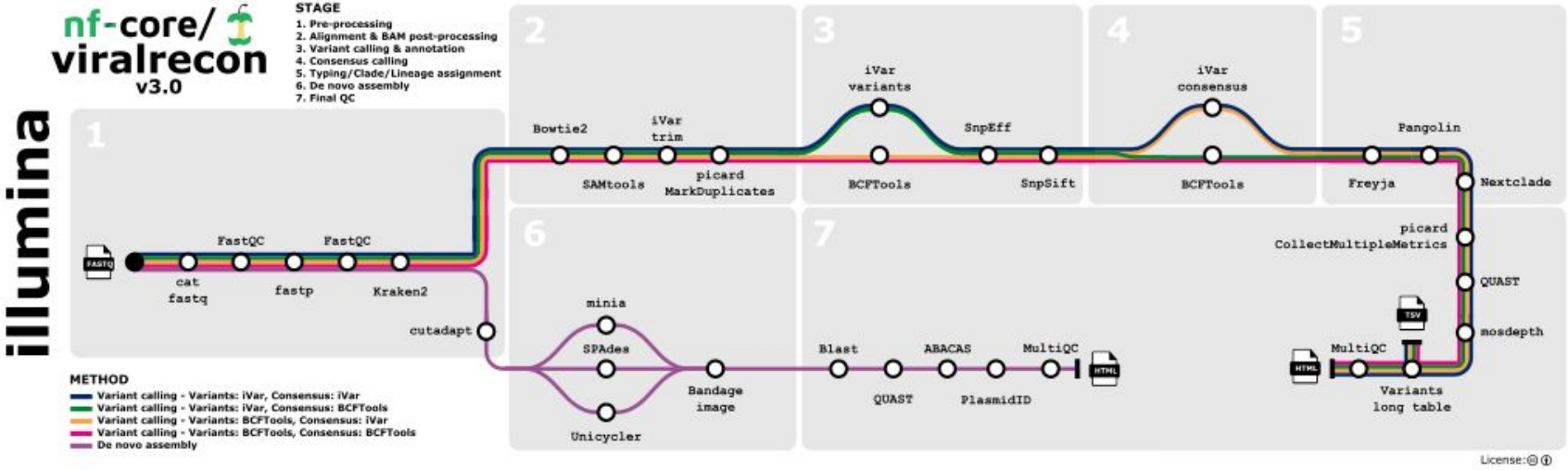
- The use case is largely the "*mobility of compute*" now!

  - Global computing "grids" like OpenScienceGrid (OSG)
  - Bioinformatics:
    - Nextflow, Galaxy.
  - A way of packaging a lot of software: Neurodesktop
  - Just running software your HPC admins may not like to maintain centrally
    - Browsers, Libreoffice, etc.

Popular areas / use cases

- Global computing “grids” like OpenScienceGrid (OSG):
  - *running a software on many HPC and HTC systems*, each of them different.
  - Distributed via CVMFS : /*cvmfs/singularity.opensciencegrid.org/ …*
  - “Sandbox containers”, unpacked images.
- Bioinformatics: *running many different software codes in a pipeline.*
  - Each software may have its dependencies (Python, Perl vers) conflicting with other items in the pipeline.
  - Often, can be installed with Conda, but HPC sysadmins dislike Conda.
  - Biocontainers, Galaxy (either CVMFS or downloadable https://depot.galaxyproject.org/singularity/)
  - StaPH containers repository
  - NextFlow is a popular tool to run pipelines, https://docs.alliancecan.ca/wiki/Nextflow .
- Similar distributions: Neurodesk project (medical imaging) https://neurodesk.github.io/

Popular areas / use cases



A NextFlow pipeline, picture from https://nf-co.re/viralrecon/3.0.0/

Running software in a Container

- Need the software, the container instance, and the container runtime
- Usually software comes with options (input and output files, flags etc.)

```
[~]$ Software_executable software_options
```

Then, to run it in the container a prefix is added:

```
[~]$ {Container runtime command and options} {Container Image or URI} \
       [Software_executable] software options
```

The container Image can be a local object (first “pulled” or “built”) or a URI in some of the repositories. An example:

```
[~]$  docker run docker://staphb/trimmomatic sh -c "echo Hello from inside the trimmomatic container
```

Example from NGC cloud: Sing. vs Docker

- NVidia provides a Container library.
  https://catalog.ngc.nvidia.com/orgs/hpc/collections/nvidia_hpc
- NAMD, a molecular dynamics package
  https://catalog.ngc.nvidia.com/orgs/hpc/containers/namd

Running with nvidia-docker

```
export NAMD_TAG={TAG} ; export NAMD_EXE=namd3 # TAG is the NAMD version number

docker run -it --rm --gpus all --ipc=host -v $PWD:/host_pwd -w /host_pwd \
nvcr.io/hpc/namd:$NAMD_TAG  ${NAMD_EXE} +p1 +devices 0 +setcpuaffinity {input_file}
```

Running with Singularity

```
export NAMD_TAG={TAG} ; export NAMD_EXE=namd3 # TAG is the NAMD version number

singularity run --nv -B $PWD:/host_pwd --pwd /host_pwd nvcr.io/hpc/namd:$NAMD_TAG  \
${NAMD_EXE} +p1 +devices 0 +setcpuaffinity {input_file}
```

Using Singularity or Apptainer

- You will need the (a?) Singularity engine installed.
  - https://github.com/sylabs/singularity (sources, RPMS)
  - https://github.com/apptainer/apptainer ; also in EPEL
  - Needs root privileges to install

- On the Alliance systems (and MagicCastes), Apptainer is installed as a module
  $> ***module load apptainer***
- On Grex, Singularity-CE is installed as a module
  $> ***module load singularity***

- Then, “apptainer” or “singularity” will be in the PATH Lets run a first container?
  $> ***singularity help*** (or ***apptainer help***)
  $> ***singularity exec library://lolcow cowsay "Mooo"***
  $> ***singularity run docker://godlovedc/lolcow*** (this will work with ***apptainer***)

Binding directories into the container

- Singularity containers are immutable ; how do we let them access our data?
  - (mostly, *–writable-tmpfs* and *–overlay* features may work)
  - Docker used to have “volume” containers for data

- Singularity containers are safe to use on HPC’s cluster file systems, like /home/ or /project or local scratch $TMPDIR or $SLURM_TMPDIR
  - *- - bind* or *-B* options to bind host directory into container
    - *- - bind /scratch:/workdir* binds /workdir in the container to */scratch*
    - *- - bind /opt* binds /opt on host to /opt in the container
  - /home/$USER , /tmp , /proc, /sys, /dev mounts by default
  - GPU drivers mounts by default with *–nv* or *–roce*
  - *- - containall* prevents default mounts if needed

- Let’s try to bind and contain directories using a image..

Demo 1: entrypoints , inside and outside

- Pull the lolcow image, from Docker or Library, on CCEnv software stack

- Try and find and understand difference between
  - apptainer run
  - apptainer exec
  - apptainer shell

- Using lolcow image, try to access files on directories other than home
- Using lolcow image, try *–containall* option.

Demo 2: manual bioinformatics

In this exercise, we will get multiple software items to run Google's DeepVariant.

- Get a ch20 genome with Wget
- Find a **BWA-mem2** container in a public repo (dockerHub, quay.io?)
- Index the genome with **bwa-mem2**

Using Singularity/Apptainer as part of larger workflow systems like Nextflow

- Manual runs like we just did are great for debugging?

Demo 3: Singularity on CVMFS

In this exercise, we will get a genomics software as a container via CVMFS

- Get a ch20 genome with Wget
- Find a **BWA-mem2** container image on one of CVMFs-s
- Index the genome with ***bwa-mem2***

Using Singularity/Apptainer as part of larger workflow systems like Nextflow

- Manual runs like we just did are great for debugging?

Demo 3: manual bioinformatics

In this exercise, we will get to run Google's DeepVariant demo on a GPU (or CPU)

- https://github.com/google/deepvariant/blob/r1.6/docs/deepvariant-quick-start.md
- Run DeepVariant as a GPU job, obtain the "Variants"
- Walkthrough here: TBD

Using Singularity/Apptainer as part of larger workflow systems like Nextflow

  - Manual runs like we just did are great for debugging?

Examples of use cases

- Using Singularity to encapsulate Conda (reduces number of files)
  - Conda is a chrooted environment that manages Python libraries
  - Also includes all the binary/OS dependencies, large number of small files

```
Bootstrap: docker
From: continuumio/miniconda:latest
%files
    # the file below must be present along the Singularity.def recipe
    environment.yml
%post
    ENV_NAME=mytest
    echo ". /opt/conda/etc/profile.d/conda.sh" >> $SINGULARITY_ENVIRONMENT
    echo "conda activate $ENV_NAME" >> $SINGULARITY_ENVIRONMENT
    . /opt/conda/etc/profile.d/conda.sh
    conda env create -f environment.yml -p /opt/conda/envs/$ENV_NAME
    conda clean --all
%runscript
    exec "$@"
```

environment.yml :

```
name: my_env
channels:
  - defaults
dependencies:
  - numpy=1.18.1
  - pandas=1.0.1
  - scikit-learn=0.22.1
```


University
of Manitoba