# Singularity containers for HPC

## Westgrid Webinar, Feb. 5, 2020

Dr. Grigory Shamov, HPC Specialist, Site Lead
UManitoba/WG

University of Manitoba

faculty of SCIENCE
discover the unknown + invent the future

compute canada | calcul canada

## How to deal with software complexity?

- Controlling software dependencies
- Configuring software packages
- Distributing software (in a portable way).
- High Performance Computing:
  - Reproducibility, Mobility of computing?
- One of the approaches is to isolate/containerize software/apps together with their dependencies

## Outline

- What are containers (In Linux)
- Containers in Enterprise and High-Performance Computing
- Singularity basics, new capabilities in Singularity 3.x
  - Building containers from recipes
  - Pulling from DockerHub
  - Singularity libraries and builders (SHub, Sylabs)
  - NVidia GPUs and Singularity
  - MPI parallel codes with Singularity
- Singularity use cases, best practices

## Glossary

Operating system - All the software that let you interact with a computer,  run applications, UI, etc. ; consists of the “kernel” and “userland” parts.

Kernel - Central piece of software that manages hardware and provides resources (CPU, IO, memory, filesystems ) to the processes it is running.

Users - Linux separates access for end users, system service accounts and root. Fine grained control (sudo, capabilities).

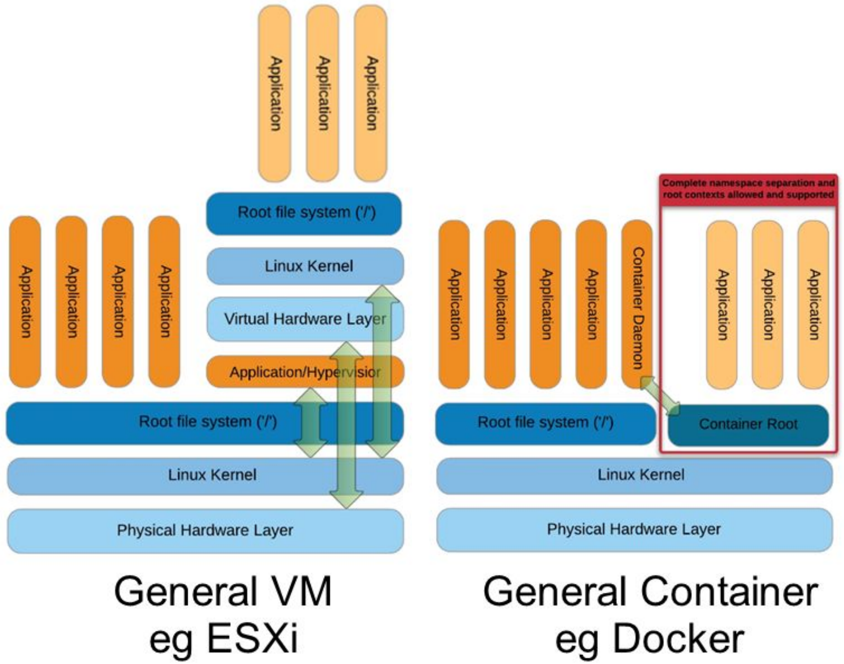
Program and Process - process is an instance of a running program with resources allocated by kernel . Processes associated with users.

Daemon process - a process that runs long time, “in the background”

# Virtual Machines or Containers?

What is the difference, and which of the m to use  when?

1. VMs offer true isolation , maximal flexibility at performance and space cost (improving a lot recently)
    a. Can run any combination of host and guest OS
    b. You can improve performance at cost of losing flexibiity and isolation
2. Containers are an OS-level mechanism of isolating userland parts of OS together with a given application
    a. Chroot on steroids; namespaces for processes
    b. Security issues of sharing the same kernel, privilege escalation
    c. No performance overhead



source: Greg Kurtzer keynote at HPC Advisory Council 2017 @ Stanford

## Containers: Docker use cases

Developed as Containers platform for services/daemons.

- Enterprise computing, Microservices approach
  - Load balancers, oversubscription
  - Orchestration tools (Google Kubernetes)
  - Many containers per node, oversubscription
- Runs  as root or a service user; cgroups for RM
- Uses commodity, Internet network stack extensively

Very popular with software developers; thanks to Recipes and the huge registry at DockerHub (https://hub.docker.com) . Made its way to Research Computing software dev.!

## Containers: HPC use case

- In HPC world jobs are ran in the batch mode: queue, start, run, end cycle.
- Often long running compute, large state (memory, data on disk).
- Often a whole node or many nodes per job, statically allocated.
  - Worst case scenarios, no CPU oversubscription possible.
- Close access to hardware for speed; specialized interconnects, RDMA, direct GPU access, zero-copy IPC
- Shared systems, pretend to be a single large machine with RM Scheduler, shared network FS, users connecting directly on the machine.

## Can I run Docker on HPC?

Short answer is often no.

- Security model: no *root* or *sudo* is given on shared machines
- Cgroups resource management will conflict with HPCs RM

But users really want it? Containers for HPC:

**Shifter**; **Singularity**; **CharlieCloud** -- either based on Docker or inter-operable with it; running containers in user space, as a user.

- Zero performance overhead for either of them (an SC19 paper).

## Singularity

Created first at LBNL, now developed by a company (SyLabs)

https://sylabs.io/

Mobility of compute, reproducible research with Containers.

Developed for HPC use case: runs as a regular user, can access shared filesystems. Interoperable with Docker; can run services as well.

Supported on ComputeCanada's HPC machines.

Many updates with version 3. (switch to Golang, new container formats, new build services, overlays, etc. -- will talk today!)

# Singularity : getting started

If you have *a)* Singularity container image, and *b)* Singularity runtime installed, you can run your app in the container in thes.

**./my-app [options] my-input.dat**

**[singularity command] [singularity-options] ./container.sif [options] input.dat**

For example,

**singularity run ./lolcow.sif**

**singularity exec -B /scratch:/scratch ./R-INLA.sif R Benchmark.R**

# Singularity : getting the runtime

On ComputeCanada HPC machines: **module spider singularity; ml singularity/3.5**

With OS package manager: **apt-get install singularity-container** (old versions on Ubuntu 18.04+) or **yum install singularity** (EPEL, has a recent version).

Build from sources: see documentation at SyLabs. (https://sylabs.io/guides/3.5/admin-guide/installation.html)

For CentOS 7: install dependencies; sudo yum groupinstall -y 'Development Tools' && sudo yum install -y \

openssl-devel libuuid-devel libseccomp-devel wget squashfs-tools cryptsetup

Download the sources: *wget \ https://github.com/sylabs/singularity/releases/download/v3.5.2/singularity-3.5.2.tar.gz*

*rpmbuild -ta singularity-3.5.2.tar.gz*; find the RPMS and install: *sudo yum localinstall singularity-*.rpm*

# How to get Singularity container images?

## Build your own; pull others' images; build on top of othersimages

Lets start with pulling a small image. (Things to consider on CC systems: network and FS performance; memory and threads  to pack/unpack the container). **singularity pull** is the command. https://hub.docker.org is the Registry.

**singularity pull lolcow1.sif docker://godlovedc/lolcow**

Or running it right away: **singularity run docker://godlovec/lolcow:latest ;** or opening a **shell** in the container!

- Images are cached , under $HOME/singularity
- Docker layers are cached too. **singularity pull docker://jfloff/alpine-python:2.7**
- We can control cache location with SINGULARITY_CACHE environment (and SINGULARITY_TMP)
- **singulatity cache {list|clean}** commands to manage cache.
- **singularity inspect** shows the image's metadata ; **--runscript**, **--deffile** opttions for SIF images are useful.

## Where to get Singularity container images?

### Pulling containers

Public containers repositories from where to "*pull*" or *build* containers.

- DockerHub public registry. **docker://**
- SyLabs public library (new). **library://**
- SingularityHub. **shub://**

Private repositories

- Require authentication. **singularity pull --docker-login docker://your-private-repo/container:tag**

CVMFS distributions! They distribute container images in an unpacked directory format. CVMFS handling various optimizations and caching of this format.

- OpenScienceGrid: https://opensciencegrid.org/docs/worker-node/install-singularity/

# Building Singularity container images

## Building might require sudo, because some OS parts belong to root

Container formats in Singularity 3.x

- SIF: the default, read-only, compressed SquashFS image. Can be signed. Can be encrypted (new in 3.x)
- Sandbox: a directory tree. Writable. Use **--sandbox** option to commands that build containers.
- Overlay: read-only SIF formats can be enhanced by a writable ext3 layed (new , experimental, requires kernel 3.18+)

Building from recipes and/or other containers with the singularity build command

- **singularity build {target} {source}** .
- **singularity build alpine.sif alpine-dir/** ; **singularity build alpine.sif Singularity** ; **singularity build alpine.sif docker://jfloff/alpine-python:2.7 ,** etc.
- Interactive container development: the **singularity shell** command. But finalize with a recipe!

## Singularity recipes

1. Specifies from where to **Bootstrap** from something (OS repo, docker, etc.)
2. Modifies the container in **%post**, copies **%files**
3. Sets the **%environment**
4. Defines entry point in **%runscript**

More examples at Github (https://github.com/sylabs/singularity/blob/master/examples)

```
Bootstrap: docker
From:  rocker/r-ver:latest
%post
       apt-get update
       apt-get install -y libssl-dev libsasl2-dev jags
autoconf automake
       apt-get install -y curl wget libudunits2-dev bash
libicu-dev libeigen3-dev
       apt-get install -y gcc-multilib g++-multilib
       # generic R packages
       R -e "install.packages('ggplot2')"
       # skipped a few packsges #
       R -e "install.packages('R2jags')"
       #R2OpenBUGS
       wget
http://pj.freefaculty.org/Ubuntu/15.04/amd64/openbugs/openbugs
_3.2.3.orig.tar.gz
       tar xzf openbugs_3.2.3.orig.tar.gz
       cd openbugs-3.2.3
       ./configure
       make && make check && make install
       R -e "install.packages('R2OpenBUGS')"
```

# Remote building services / libraries

## What if you do not have sudo anywhere? How do you distribute your Sing. images?

The original build service: V. Sochat's **SingularityHUB**

- Link your Github repo with recipe to https://singularity-hub.org ; wait for it to build; setup auto rebuild hooks.
- Pull the container from anywhere like so (putting your URI of course): **singularity pull shub://vsoch/hello-world**
- Right now locked down due to an abuse by a malicious user; limits are set to the number of pulls.

The new **SyLabs Cloud** service.

- Register with an identity provider (Google, FB, MS, Github) at https://cloud.sylabs.io ; Get an access token.
- Either use **--remote** build option from a local Singularity installation or deposit a recipe
- Pull the container from anywhere with **singulatity pull library://your-name/aproject/your-image:tag**

## Running containers on HPC machines

Access to filesystems using --bind | -B options.

- Bind-mount is a Linux kernel mechanism. **-B outside:inside**
- Some paths are  mounted by default. /home , /tmp, …
- Sysadmin can configure more/less paths by default
- You can prevent mounting paths by --containall (other?) options

```
singularity exec -B /sbb/scratch:/global/scratch -B /scratch:/scr Tf.sif python mytf.py
```

## Running GPU software in containers

Singularity supports NVidia through bind-mounding the GPU drivers and base CUDA libraries. **--nv** flag foes it.

- **singularity exec --nv -B /sbb/scratch:/mnt tensorflow.sif python my-tf.py**
- NVidia provides readily made containers for a large number of HPC apps.
    - https://ngc.nvidia.com/catalog/containers
- Example: NAMD on GPU from NVIDIA

Singularity also supports AMD with **--roce** flag.

# Running MPI software in containers

## MPI libraries are using a high-performance interconnect, RDMA etc. so they are hard to containerize

MPI  is a standard; for message passing interface. MPI comes with several implementations (OpebnMPI, MPICH, IntelMPI, PlatformMPI, Cray MPI, .. ). Thus no generic --mpi flag is easy to implement for the containers. The following modes can be thought of::

1. Inside container (less interesting, won't work across the nodes); singularity exec my.simg mpiexec hello.mpi
2. Hybrid mode: same (or similar) MPI in container and outside. Mpiexec singularity run hello.mpi
3. Bind mode: host's MPI libraries and drivers are mounted into the container. Mpiexec singularity

Examples of the recipes at Sylabs documentation page (https://sylabs.io/guides/3.5/user-guide/mpi.html).

Less of a use case? Most MPI software in HPC world  comes as sources.

However, MPI+X model might be useful (try  GAMESS-US with MPI with GPU, or LAMMPS+GPU, etc.).

# Services with Singularity

A  new feature in v. 2.4, expanded in v.3.4+

If you really need to run a daemon / service (which is not usually an HPC workload)/

- The command is **singularity instance** (**singularity instance start http.sif my-web**)
- It can use its own Cgroups mechanism (--cgroups flags) to manage the resources.
- It can run in a privileged mode, as root, with fine-grained capabilities. (--addcap )
- Documented at SyLabs site: https://sylabs.io/guides/3.5/user-guide/running_services.html

## Conclusions

Containerization technology is mature and usable.

If applied wisely it can help with mobility, reproducibility of research.

The Singularity is one of the most popular, production ready containers for HPC.