ComputeOps: container for High Performance Computing

C. Cavet, S. Gadrat and the ComputeOps project members

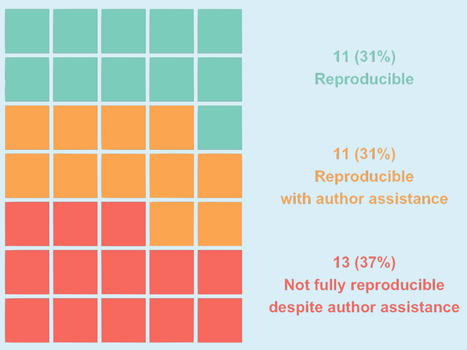
ccavet@apc.in2p3.fr & sebastien.gadrat@cc.in2p3.fr

Abstract

The High Performance Computing (HPC) domain aims to optimize code in order to use the last multicore and parallel technologies including specific processor instructions. In this computing framework, portability and reproducibility are key concepts. A way to handle these requirements is to use Linux containers. These "light virtual machines" allow to encapsulate applications within its environment in Linux processes. Containers has been recently rediscovered due to their abilities to provide both multi-infrastructure environnement for developers and system administrators and reproducibility due to image building file. Two container solutions are emerging: **Docker** for micro-services and **Singularity** for computing applications. We present here the **ComputeOps** project which has the goal to study the benefit of containers for HPC applications.

HPC

▶ **Portability and reproducibility/repeatability in research.**



Reproducibility of scientific results from articles @T. Hardwicke and al.

▶ **With container solution**:

- portability: on multi-environment;
- repeatability: encapsulation of a specific environment in a shareable container image.

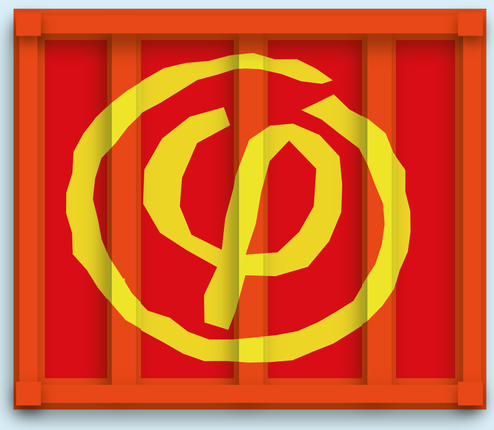
French partners

▶ **ComputeOps partners**:

- **Aristote**: GENCI, IDRIS (french supercomputers).
- **Institutes and labs**.
- **P2IO ACP**:
  - GPGPU: two cards of 1GB;
  - CUDA 9.2 and 9.0, last Nvidia driver;
  - TensorFlow library;
  - **New: FPGA card** (Alveo U280 Xilinx Data Center Accelerator FPGA card.).

Physique des 2 Infinis et des Origines

ComputeOps project



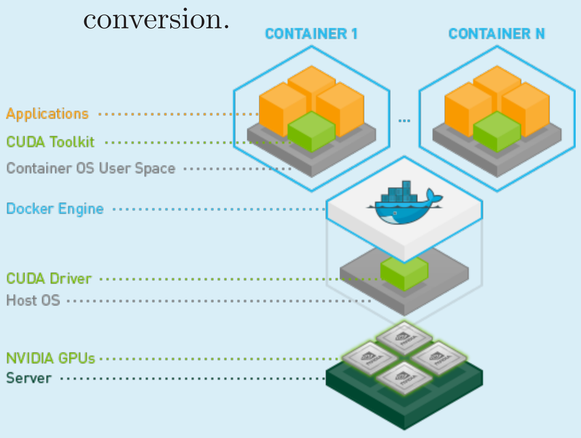
▶ **Goal:** the project [1] studies and provides tools for the community.

- container technology interoperability.
- portability on various architectures and infrastructures.
- repeatability of computing.
- good practices for writing recipes.

Containers on GPU

▶ **Goal:** a solution to access accelerated hardware from containers.

- NVidia GPUs: containers mount the suitable runtime environment from the host;
- NVidia-docker [6]: a convenient stack including a flag at runtime bind-mounting librairies and devices, and specific developments;
- Limitations: Docker to Singularity conversion.



NVidia-docker stack @NVidia.

Singularity v3

▶ **New features:**

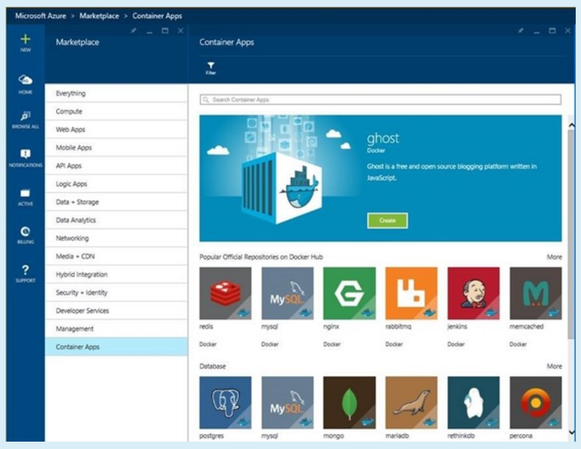
- source language: Go;
- SIF image format: signature, tracability and encryption;
- Docker registry: ORAS compatibility for storing SIF images;
- Sylabs cloud: building and storing images.

CSAN

▶ **Goal:** a solution to reuse and share the same applications in a specific field of research.

▶ **A Comprehensive Software Network Archive (CSAN):**

- /!\ work in progress;
- a software solution based on Singularity [7] and Guix [4] package manager allowing a full reproducibility;
- a web portal for sharing images;
- an expert panel for image validation.



Microsoft Azure container Marketplace @Microsoft.

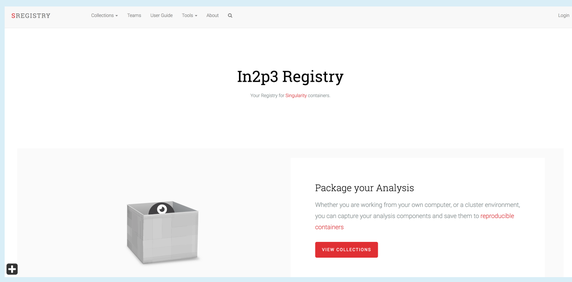
Sharing container image

▶ **Hubs, registries and Marketplaces:** provide catalogues of images.

- sharing images with tagged versions;
- giving metadata information: owner, recipe, code repository, etc.

▶ **Private Hubs:**

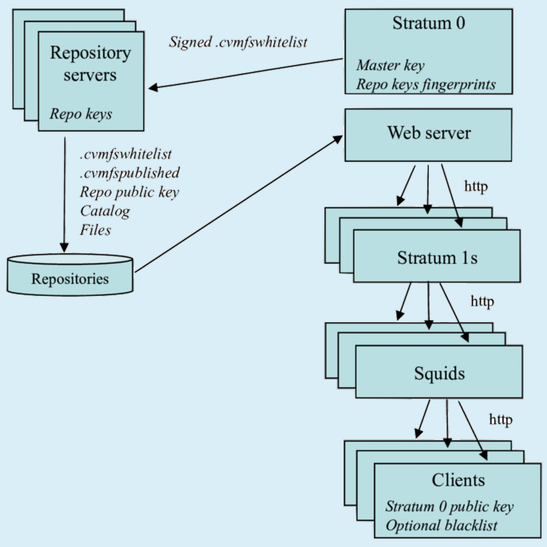
- an authentification mechanism relying on a GitLab/GitHub account;
- image collections for specific projects;
- manageable workflows;
- the ability for users to rate available containers and view recipe sources.



ComputeOps SHub [2].

▶ **File system:**

- CernVM-FS [3]: a software distribution system allowing to deploy software on distributed infrastructures.
- Software deployment: one dedicated server (stratum-0) which will mirror and cache the software on all the registered servers.



CernVM-FS workflow @CERN.

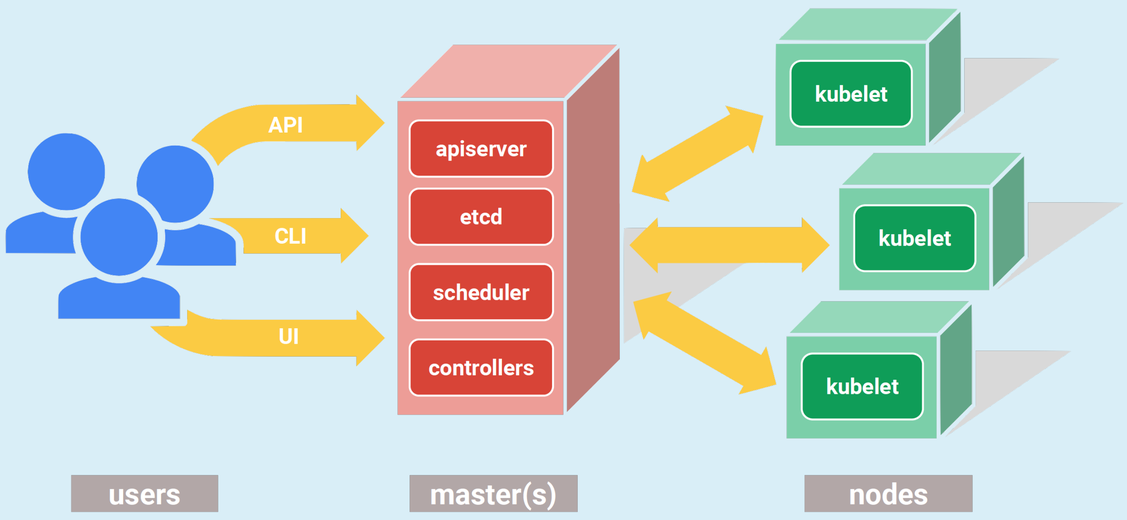
Orchestrator vs scheduler

▶ **Scheduler**:

- manage job distribution from the master node to the cluster of computing nodes;
- handle jobs with finite time: highly efficient when dealing with multiple users, providing job queueing mechanisms and complex fair resources sharing rules.

▶ **Orchestrator**:

- manage containerized workloads, providing primarily resource management;
- features: service reliability, isolation, scalability, etc.

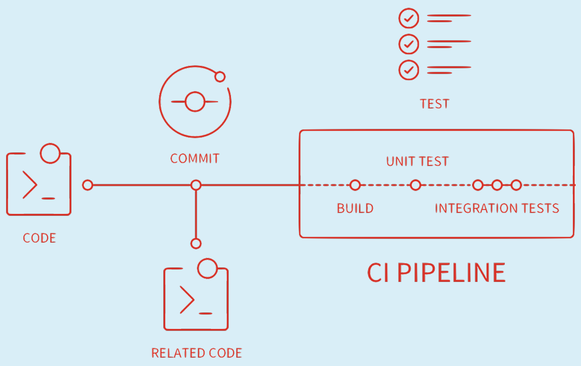


Architecture of Kubernetes [5] orchestrator @Google.

CI with containers

▶ **Goal:** a way to automatically build and share container images on registries.

- Commit on GitLab-CI: run the Docker image creation stage and push images on the private registry;
- Singularity in Docker (SinD): automatically build Singularity images with infrastructure specifications (bind volumes) and publish/update them on hubs;
- Singularity Hub (SHub): collections of automatically built images;
- Run in production: on a HPC cluster within the scheduler.



How to contribute?

▶ **Open to suggestions /contributions/collaborations:**

- More information on the wiki of the ComputeOps project [1].
- Email to `computeops-l@in2p3.fr`.

References

[1] ComputeOps wiki: `https://gitlab.in2p3.fr/CodeursIntensifs/DecaLog/wikis/ComputeOps`
[2] ComputeOps Shub: `https://sregistry.in2p3.fr`
[3] CVMFS: `http://guix.gnu.org/`
[4] Guix: `http://guix.gnu.org/`
[5] Kubernetes: `https://kubernetes.io/`
[6] NVidia-docker: `https://github.com/NVIDIA/nvidia-docker`
[7] Singularity: `https://sylabs.io/`

The project is founded by the CNRS IN2P3 (french Institute for Nuclear Physic and Particule Physic) and is part of the DecaLog Master Project.