EPJ Web of Conferences **214**, 07004 (2019) https://doi.org/10.1051/epjconf/201921407004
CHEP 2018

ComputeOps: containers for High Performance Computing

Cécile Cavet[1,*], *Aurélien* Bailly-Reyre[5,8], *David* Chamont[6], *Olivier* Dadoun[8], *Alexandre* Dehne Garcia[3], *Pierre-Emmanuel* Guérin[2], *Pascale* Hennion[7], *Oleg* Lodygensky[6], *Gérard* Marchal-Duval[6], *Emmanuel* Medernach[4], *Victor* Mendoza[8], *Jérôme* Pansanel[4], *Richard* Randriatoamanana[2], *Andrea* Sartirana[7], *Martin* Souchal[1], and *Julien* Tugler[7]

[1]APC/FACe, Université de Paris, CNRS/IN2P3, CEA/Irfu, Obs. de Paris, 10 rue A. Domon et L. Duquet, 75013 Paris, France
[2]ICI-CNSC, 1 Rue de la Noe, 44321 Nantes, France
[3]CBGP, INRA, CIRAD, IRD, Montpellier SupAgro, University Montpellier, Montpellier, France
[4]IPHC, 23, rue du Loess - BP28, 67037 Strasbourg Cedex 2, France
[5]ISCD, BP380, 4, place Jussieu, 75252 Paris Cedex 5, France
[6]LAL, Centre Scientifique d'Orsay, Bat. 200 - BP34, 91898 Orsay Cedex, France
[7]LLR, Ecole Polytechnique, Rue de Fresnel, 91128 Palaiseau, France
[8]LPNHE, Campus de Jussieu, 4 place Jussieu, 75252 Paris Cedex 5, France

Abstract. The High Performance Computing (HPC) domain aims to optimize code to use the latest 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 users to encapsulate applications within its environment in processes. Containers have been recently highlighted because they provide multi-infrastructure environnement for both developers and system administrators. Furthermore, they offer reproducibility due to image building files. Two container solutions are emerging: **Docker** for microservices and **Singularity** for computing applications. We present here the **ComputeOps** project which investigates the container benefits for HPC applications.

1 Introduction

The **ComputeOps**[1] project, started in 2018, targets container technologies for the High Performance Computing (HPC) domain. The project will study and provide:

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

*e-mail: ccavet@apc.in2p3.fr

[1]ComputeOps Wiki page on the CNRS/IN2P3 GitLab:
https://gitlab.in2p3.fr/CodeursIntensifs/DecaLog/wikis/ComputeOps

The first year of the project has been dedicated to selection of container technology allowing to run container applications on HPC infrastructures. The project members have also selected several use cases as pilot applications. These pilot applications will demonstrate the interest of encapsulating environment in containers in the HPC domain. Furthermore, the project has started to provide tools for the research community: a private Singularity Hub has been settled and connected to the GitLab platform.

1.1 Partners

Several institutes or research laboratory groups are partners of the **ComputeOps** project. These partners are providing skills and/or R&D infrastructures as described in this section.

Aristote Virtual

Aristote[2] is a french learned society which includes both academic and industrial players in digital technologies. The Aristote Virtual working group was created in January 2018. The group deals with usability and performance of containers on HPC machines, and aims to embark new research communities on HPC. The four French national data centers and several regional ones are active members of the working group.

Ecole Centrale de Nantes, Institut de Calcul Intensif

The *Institut de Calcul Intensif*[3] (ICI) is a HPC research institute which was established at *Centrale Nantes* in January 2015, following its successful application in the *Pays de la Loire Connect Talent* call for projects. The institute associates a HPC research laboratory and supercomputing facilities. The project objectives are related to the democratization of numerical tools for massively parallel computing for a wide range of applications and challenges.
Since 2016, the *Centrale Nantes SuperComputing Centre* (CNSC), fully operated by ICI, hosts LIGER[4] supercomputer. This regional machine based on x86 superscalar instruction currently is ranked in top 5 in French Tier2-class. It is devoted to provide computational resources and expertise from the basic scientific research to unprecedented levels of precision, releasing new potential for innovation. LIGER is a BULL/Atos DLC720 x86 supercomputer with the following characteristic features:

- 281 TFlop/s Rpeak;
- 252 compute nodes and 14 GPU Nvidia K80 nodes;
- 6384 cores Intel Xeon E5v3;
- FDR Infiniband interconnect;
- 900 TB GPFS IO Fast Storage.

Some of the research topics run on LIGER are innovative and will explore massively parallel numerical techniques for multiphase computational fluid dynamics: anisotropic mesh adaptation and its coupling with immersed volume methods, imaging with automatic reconstruction of numerical models from 3D images, offline and online computations.

[2]Aristote: http://www.association-aristote.fr/doku.php/public:aristote:contact
[3]ICI: https://ici.ec-nantes.fr/
[4]LIGER supercomputer: https://ici.ec-nantes.fr/centrale-nantes-supercomputing/

P2IO ACP

Accelerated Computing for Physics (ACP), funded by the French P2IO consortium[5], is an R&D projet about AI computing on accelerated hardware (GPUs, manycore, FPGA). It focuses in particular on the adaptability of applications to different hardware via code generation and the definition of customizable environments via containers. It also provides **ComputeOps** a platform for the R&D activity, currently consisting of one machine with the following hardware:

- 2 Nvidia Tesla V100 GPU cards, 16 GB memory.
- 2 Intel(R) Xeon(R) Gold 6138 CPU @ 2.00 GHz, 20 physical cores each.
- 196 GB RAM.

A second machine equipped with FPGA and AMD processors is in the process of purchasing. Standard GPU and machine learning development tools and framework (CUDA libraries, PGI with OpenACC libraries, TensorFlow framework, etc.) are available on the platform in different OS environments (Scientific Linux 6, CentOS 7, Debian 9, etc.) via Singularity containers.

ACP also contributes **ComputeOps** some pilot applications (SMILEI, MEM, HAhRD, Supernovae electron cross-section application) detailed in the next Section 1.2.

CBGP

The Biology Center for Population Management[6] (CBGP) carries out research in the fields of systematics, genetics and ecology relevant to the management of populations and communities of organisms for the purposes of agriculture, public health and biodiversity. CBGP has been hosting its own HPC facilities since 2000. All of its IT services such as HPC computing, databases or storage have been containerized since 2009.

1.2 Pilot applications

The pilot applications have been chosen to tackle research in particle physics and gravitation. They have been selected in order to use the specific characteristics of the underlying hardware:

- **SMILEI**[7]: the code is an open-source, particle-in-cell code. As a multi-purpose code, it is designed for and applied to a wide range of plasma physics-related studies: from relativistic laser-plasma interaction to astrophysical plasmas. It is co-developed by both physicists and HPC experts, supported by GENCI[8] and Intel.
- **CMS MEM**[9]: this code uses the Matrix Element Method (MEM) for the analysis of the Higgs boson production with two top quarks (ttH channel) in the CMS experiment. The code, based on OpenCL and CUDA, is adapted for MPI as well as multi-GPU (>20 GPUs) execution.
- **HAhRD**[10]: HPC Algorithms for high Resolution Detectors (HAhRD) is a code developed in the context of 2018 Google Summer of Code (GSOC'18). It is a machine-learning application based on TensorFlow libraries for image reconstruction in the CMS High Granularity Calorimeter (HGCAL) optimized for running on extensible processor and GPU platforms.

[5] P2IO: http://www.labex-p2io.fr/
[6] CBGP: https://www6.montpellier.inra.fr/cbgp/
[7] SMILEI: http://www.maisondelasimulation.fr/smilei/
[8] GENCI: www.genci.fr/
[9] CMS MEM: https://indico.cern.ch/event/587955/contributions/2937584/
[10] HAhRD: https://github.com/grasseau/HAhRD/wiki

- **electron_capture**: this is a code computing electron capture rates, based on finite temperature Hartree-Fock and finite temperature random-phase approximation (RPA) using Skyrme interactions, used for astrophysics computation [2]. The application is coded in Fortran90 with OpenACC directives for running on GPU.
- **TensorFlow**: machine learning applications using the TensorFlow library.
- **LDC**: simulation pipeline for scientific challenges of the LISA gravitational wave detector[11]. The application is parallelized with MPI and used the PyMultiNest library for Bayesian analysis.
- **Geant4**: simulation for direct Dark-Matter detection experiments (Xenon & DarkSide).

The pilot applications have been containerized in the Singularity container solution (see Section. 2.1.1). We will explain this choice in the next Section. The containerized applications are ready to be executed on R&D and production infrastructures.

2 Containers for computing

2.1 Solution description

Since 2005[12], several container solutions are emerging: Docker, Rkt, Singularity, etc. Among these solutions, Docker[13] has driven the community by providing a complete ecosystem: Docker daemon, client, Hub/private Registry, Compose, Swarm, Machine, etc. The Docker solution is evolving rapidly and it offers a user friendly environment for deploying micro-services. The micro-service concept has been quickly adopted by the industry and the web community due to the simple description of the system architecture. Furthermore, Docker containers can be run on multi-infrastructures such as bare metal system, local and cloud computing virtual machines (VMs) and on container clusters managed by an orchestrator. Processes in container can be accessed as root.

In the context of the **ComputeOps** project, we have studied the Docker solution for research in the HPC domain (see also [1]). The Docker's paradigm is based on the representation of a container as a lightweight VM that should host a single service. But this idea of containers as miniature VMs is a wrong approach for the HPC community. Indeed, micro-services can not be used on computing clusters due to security constraints or working methods different from the typical Docker use case. For example, on computing clusters and particularly on supercomputers, the container permission has to be set in the unprivileged mode similar to the user's home on the computing infrastructure. Moreover, software solutions should be ready to deploy on existing infrastructure or should be deployed easily without interfering with existing systems. The software solutions must take advantage of the hardware characteristics. Finally, the Docker container format is not easily transportable and shareable: scientists are looking for simple solutions to share, publish and ensure the reproducibility of codes and calculations. For all these reasons, Docker is not the optimal solution for executing scientific applications in an HPC environment. Fortunately, there are other solutions at this time that take advantage of the benefits of containerization, while adapting to the scientific environment. The **ComputeOps** team has chosen to focus on four container solutions that are restricting user permissions, fully open source and free. We are presenting them in the next Sections.

[11] LISA: https://www.elisascience.org/
[12] The Docker company has started to provide an innovative container solution in 2005.
[13] Docker: https://www.docker.com/

2.1.1 Singularity

Singularity[14] is an emerging container solution with an active community. The advantages of this solution are:

- **Compatibility with Docker**: a Docker image can be used to create a Singularity image or can be directly used from the Docker Hub to be run by Singularity;
- **Security**: unprivileged mode;
- **I/O**: transparency allowing a compatibility with MPI processes and X11 graphical export;
- **Scheduler**: native integration with schedulers. Singularity images can be submitted and executed as a job on a computing cluster;
- **GPU**: easy GPU integration in containers;
- **Mobility of computing**: ability to define, create and maintain a workflow and be confident that the workflow can be executed on different hosts, operating systems (as long as it is Linux) and service providers;
- **Single filesystem**: Singularity containers use a single file which is the complete representation of all the files within the container. This feature which facilitates mobility also facilitates the reproducibility of computing;
- **User freedom**: Singularity gives to users the freedom they need to install the applications, versions, and dependencies for their specific workflows without impacting the system.

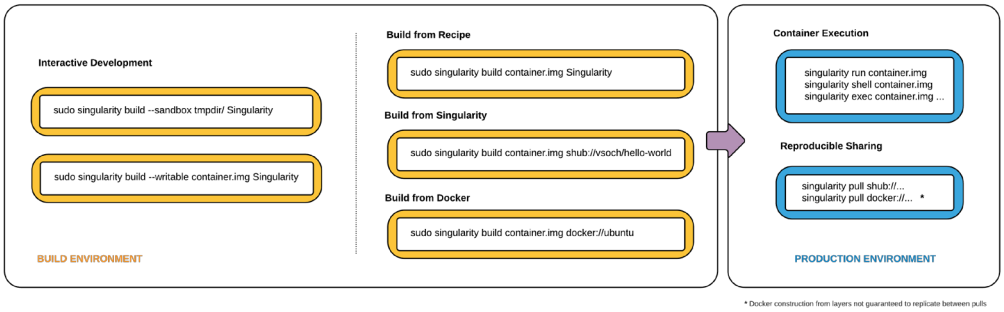

Figure 1. Singularity workflow: build step in privileged mode and execution step in unprivileged mode ©Sylabs.io.

The Singularity workflow, shown in figure 1, is divided in two steps. First, the user builds an image in a *privileged mode* (on a local system or a VM) with a specific bootstrap method (a Singularity file, a Singularity Hub or a Docker Hub). Second, the Singularity container is executed in an *unprivileged mode* on a computing cluster within a batch scheduler. This workflow is valid for the other container solutions that we are presenting in this Section.

2.1.2 Shifter

Shifter[15] is a prototype implementation that the NERSC is developing and experimenting as a scalable way of deploying containers in an HPC environment. The solution works by

[14] Singularity: http://singularity.lbl.gov/
[15] Shifter: https://docs.nersc.gov/development/shifter/how-to-use/

converting user or staff generated images in Docker, VMs, or CHOS (another method for delivering flexible environments) to a common format. This common format then provides a tunable point to allow images to be scalably distributed on Cray supercomputers at the NERSC. The Shifter user interface enables a user to select an image from their Docker Hub account and then submit jobs which run entirely within the container.
Shifter is different from Singularity because it tries to keep Docker's paradigm: containers are represented as small VMs. In reality, Shifter is complicated to install and to setup in an existing environment, and the documentation is not yet complete. At this time, the Shifter solution is installed mainly on Cray supercomputers at the NERSC.

2.1.3 uDocker

uDocker[16] is a basic user tool to execute simple Docker containers in user space without requiring root privileges. The solution enables to pull and run Docker containers by non-privileged users in Linux systems where Docker is not available. It can be used in Linux batch systems and interactive clusters that are managed by other entities such as grid infrastructures or externally managed batch or interactive systems. uDocker "executes" the containers by simply providing a chroot like environment over the extracted container. The current implementation supports different methods to mimic chroot enabling execution of containers without requiring privileges under a chroot like environment.
uDocker is a simple tool written in Python. The solution has a minimal set of dependencies so it can be executed in a wide range of Linux systems. It has less features than Singularity.

2.1.4 CharlieCloud

One of the advantages of CharlieCloud[17] is that it allows users to create user space containers from Docker images but also from any tool that can generate a standard filesystem. It is an interesting solution for the **ComputeOps** applications but it is a slightly more complicated than Singularity to install and use for containerization. Moreover, it is not as flexible in the management of GPU libraries. The documentation is currently quite brief. This solution is not largely used in computer research laboratories.

3 Achievements

The **ComputeOps** project has started to deliver tools for the research community. The team has provided a Singularity Hub in order to host the images of the pilot applications deployed by the GitLab continuous integration/continuous deployment (CI/CD) plateform. Furthermore, tutorials have been realized since the beginning of the project in order to push the container adoption in the the research environment.

3.1 Singularity Hub

The Singularity private Hub for research[18] allows to manage Singularity images as shown in figure 2. This marketplace for Singularity images is working with (see also Sect. 2.1.1):

- an authentification mechanism based on a GitLab/GitHub account;
- collections of images for specific projects;

[16] uDocker: https://www.indigo-datacloud.eu/userspace-container-support
[17] CharlieCloud: https://hpc.github.io/charliecloud/
[18] ComputeOps Singularity Hub: https://sregistry.in2p3.fr

- manageable workflows;
- the ability for users to rate available containers and view recipe sources.

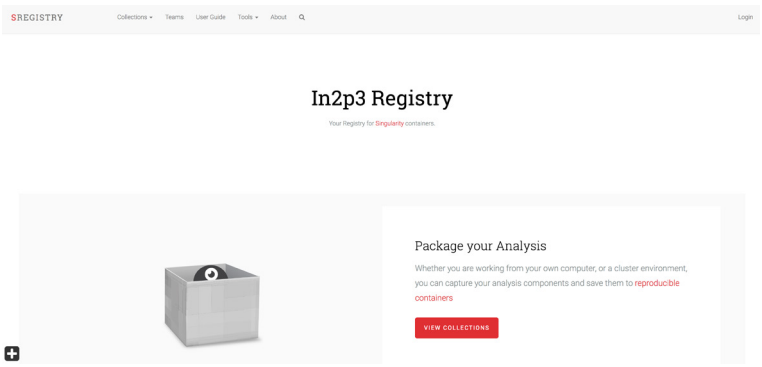

Figure 2. A Singularity private Hub for Research

3.1.1 The ComputeOps image collection

The **ComputeOps** image collection has several goals:

- share images of pilot applications;
- reviewer team experimentation for image validation;
- label processing and container maintenance.

The collection is still in the building process and has just been started its adoption by the research community.

3.1.2 A Comprehensive Software Archive Network

In the continuity of the **ComputeOps** collection, another initiative has been started. Based on the model of the Comprehensive Archive Network like CRAN for R, CPAN for Perl or CTAN for Tex, a Comprehensive Software Archive Network (CSAN) based on container solutions has been initiated. This network of experts will provide pre-packaged versions of scientific tools that are simple to install and simple to use validated by a peer-review process. The pre-packaged tools will be compliant with best practices in terms of security and HPC compliancy (CPU, RAM, and network optimized). Therefore, CSAN should be considered as a trusted third party which allows scientists to focus on their science and not on the tricky software installation process on their laptop or an exascale HPC cluster.

3.2 CI with Singularity

In order to deliver continuously Singularity images on distributed infrastructures, the team has provided a CI/CD workflow based on GitLab-CI and Singularity components. The workflow, schematized in Figure 3, has three stages:

- *Commit*: the code hosted on the GitLab-CI starts a pipeline on commit. The pipeline running in containers can build a Docker image (Docker-in-Docker) and pushes it on the GitLab registry (Docker private Registry);

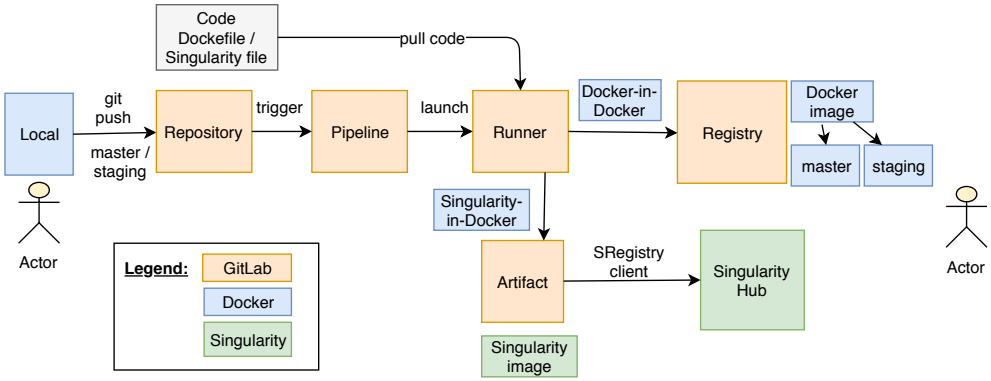

Figure 3. CI/CD with Singularity: a new commit triggers the build of a Singularity image that will be pushed on a Singularity Hub

- *Singularity-in-Docker*: the next pipeline step automatically builds Singularity images and publishes/updates images on hubs. By using a pre-packaged Docker image with Singularity and SRegistry client, the pipeline builds an image and stores it in GitLab as an artifact. In order to authenticate on the Singularity Hub, the user token is stored in the environment variable settings;
- *Deploy in production*: on computing clusters, the Singularity solution is often available. Automatically build images can be pulled from the Singularity Hub and executed within the scheduler.

3.3 Tutorials

We have realized several tutorials and trainings for beginners and advanced users:

- Deep dive in container technologies: *MaitresNageurs/EnBarque*[19];
- Containers in production: *IN2P3 IT school 2018* [20];
- How to create an OpenMPI container: *SBAC-PAD Symposium* [21].

The tutorials are based on Docker and Singularity solutions in order to cover both micro-services and scientific applications.

4 Acknowledgements

The project is financed by the french CNRS/IN2P3 institute as the DecaLog Master Project for transversal Research & Development on Computing and Data.

References

[1] C. Arango, R. Dernat, J. Sanabria, arXiv:1709.10140 (2017)
[2] A. F. Fantina, E. Khan, G. Colò, N. Paar, D. Vretenar, Phys. Rev. C **86**, 3 035805 (2012)

[19] Piscine: https://gitlab.in2p3.fr/MaitresNageurs/EnBarque
[20] IN2P3 IT school: https://indico.in2p3.fr/event/17124/timetable/#20180607
[21] SBAC PAD tutorial: https://gitlab.in2p3.fr/alexandre.dehne-garcia/TP_singularity_EcoleConteneursProd