ComputeOps: Container for High Performance Computing

Cécile Cavet[1,*], *Martin* Souchal[1,**], *Sébastien* Gadrat[2,***], *Gilles* Grasseau[3], *Andrea* Satirana[3], *Aurélien* Bailly-Reyre[4,5], *Olivier* Dadoun[5], *Victor* Mendoza[5], *David* Chamont[6], *Gérard* Marchal-Duval[6], *Emmanuel* Medernach[7] and *Jérôme* Pansanel[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]Centre de Calcul de l'IN2P3, 21 Avenue Pierre de Coubertin, 69100 Villeurbanne, France
[3]LLR, Ecole Polytechnique, Rue de Fresnel, 91128 Palaiseau, France
[4]ISCD, BP380, 4, place Jussieu, 75252 Paris Cedex 5, France
[5]LPNHE, Campus de Jussieu, 4 place Jussieu, 75252 Paris Cedex 5, France
[6]IJCLab, CNRS Université Paris-Saclay Bât. 100, Faculté des sciences, F-91405 Orsay Cedex
[7]IPHC, 23, rue du Loess - BP28, 67037 Strasbourg Cedex 2, France

Abstract. The High Performance Computing (HPC) domain aims to optimize code in order 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 to encapsulate applications within its environment in Linux processes. Containers have 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 status of the **ComputeOps** project which has the goal to study the benefit of containers for HPC applications.

1 Introduction

Containers provide flexible strategies for packaging, deploying and running isolated application processes within multi-user systems, thus enabling scientific reproducibility. The recent re-discovery of Linux containers has given rise to an innovative way to encapsulate and share codes and services.

We will present here a status report of the ComputeOps project, highlighting on the progress made since the last proceeding published for CHEP 2018 [1].

2 Container solution evaluation

Several groups and research groups are partners of the ComputeOps project. Each partner provides skills and/or R&D infrastructures, as well as pilot applications, to tackle the main

*e-mail: ccavet@apc.in2p3.fr
**e-mail: souchal@apc.in2p3.fr
***e-mail: sebastien.gadrat@cc.in2p3.fr

research topics in particle physics and astronomy. A complete description of the partners and the pilot applications can be found in the CHEP 2018 proceeding [1].

A first step of the project was to evaluate the main container solutions available: Docker, Singularity, Rkt, uDocker, Shifter, CharlieCloud and Kata containers [1]. The conclusion of this evaluation has shown that the Singularity [2] solution is the best compromise for the time being. The solution can easily be installed in local clusters and in computing centers and can be used without privileged mode. Singularity is the most widely used container solution in HPC centers and has a large users community. It is also fully compatible with Continuous Integration (CI) which is an important feature for a large community of researchers.

It is however important to note that all these containers solutions are evolving quickly, and some other solutions seem quite promising.

3 Sharing container images

In order to host the container images, catalogues of images such as hubs, registries and marketplaces are traditionally provided in the ecosystem built around a container technology. Catalogues of images are usually linked with source code management tools, hosting the recipe used to build the image in a fully transparent approach.

We will see in this section that other tools may be used for sharing container images between users. An other important aspect will also be approached with the automatic construction of the container image. Indeed, the GitLab [4] source code management tool also provides a Continuous Integration and Continuous Deployment (CI/CD) plateform which allows to automatically build images of applications and deploy them in catalogues of images or direcly on infrastructures.

3.1 Hubs, registries and marketplaces

Public hubs such as Docker Hub [5] and Singularity Hub [6] are based on open source solutions and provide a registry service and a web portal. Both hubs offer to users the possibility to use public images and to manage their own container images.

For a specific user community such as the IN2P3 research community [7], private hubs have also started to flourish. They are based on:

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

In order to manage quality and security issues in container images, image scanning can be used in a CI process to check whether the image is compliant with the lastest security releases. Several tools such as Clair [8] and Trivy [9] have been developed during the last years in order to provide a way to search for Common Vulnerabilities Exposures (CVEs) inside container images.

3.2 Filesystem

CernVM-FS [11] (usually referred as CVMFS) is a software distribution system allowing to deploy software on world-wide distributed infrastructures. It has been developed to assist High Energy Physics (HEP) collaborations to deploy their software on all sites where they are running jobs. The software deployment is usually made on one dedicated server (so

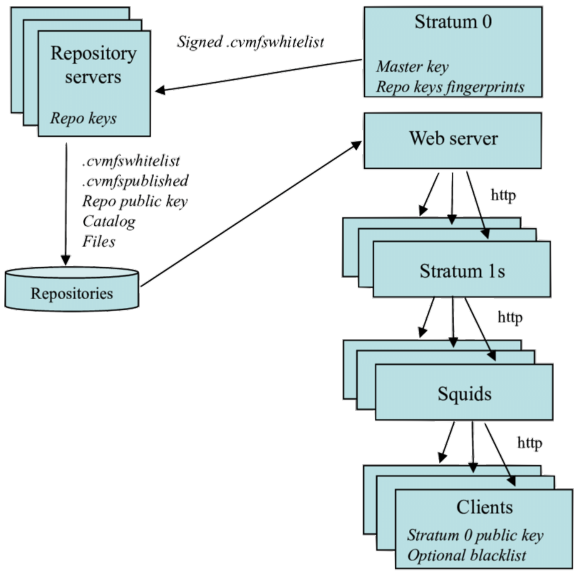

Figure 1. CernVM workflow to publish data in CernVM-FS [12].

called stratum-0) which will mirror the software on all the registered servers (see Fig.1 for the detailed workflow).

CVMFS appears quite convenient for sharing and publishing container images (definition files will however rather be stored in GitLab). The preferred image format will be the Singularity unpacked (directory tree) images as this format will allow I/O optimisation thanks to advanced CVMFS features. This format is breaking the container portability, though the publication workflow will ensure that the image is distributed to all clients.

3.3 A Comprehensive Software Archive Network (CSAN)

In the model of Comprehensive Archive Networks such as CRAN [13] for R, CPAN [14] for Perl or CTAN [15] for TeX, the ComputeOps group is currently working on building a Comprehensive Software Archive Network (CSAN). This CSAN will be based on Singularity [2] and the Guix [16] package manager which will allow a full reproducibility of the container including all the software installed inside it thanks to the Guix package manager. A user-friendly web portal, such as the Microsoft® Azure Container Marketplace (see Fig. 2), will be built to allow users to easily search for containers. Image validation (done by experts) will be required before uploading the images in the CSAN.

The CSAN portal will offer a wide variety of containers provided by the community, validated by experts, to the scientific community. It will allow a subtantial gain of time for most of the users, granting them access to containers in a friendly way.

3.4 Continous integration with containers

Continous integration (CI), such as the one provided by GitLab, allows to define workflows to automatically build and distribute container images.

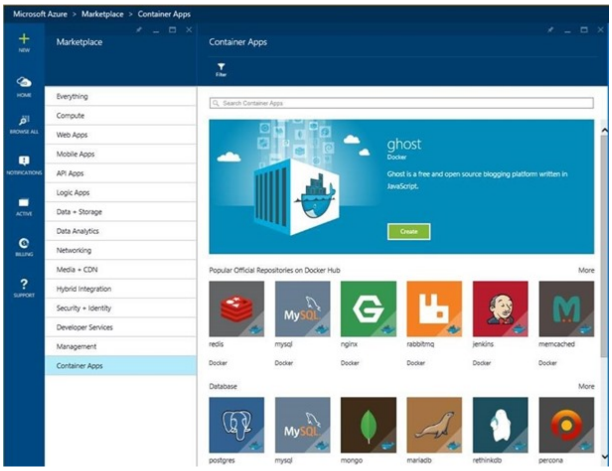

Figure 2. The Microsoft® Azure Containers Marketplace web portal.

The CI workflow involving containers is usually made of several steps:

1. code commit: the code hosted on the GitLab-CI starts a new pipeline on each commit or upon some specific conditions;
2. pipeline runner: GitLab-CI provides a container runner based on Docker, in which one can build a Docker or Singularity image following the recipe provided;
3. automatic deployment of built images and/or automatic pushing to a registry;
4. image scanning: check the CVEs (Common Vulnerabilities Exposure) in the final image (see section 3.1).

This workflow automation for building images is an important part in the portability and reproducibility requirements.

4 Orchestrator versus scheduler

4.1 Schedulers

Schedulers (or Workload Managers) are designed to handle jobs with finite time, especially when dealing with multiple users, providing job queueing mechanisms and complex fair resources sharing rules. Managing containerized applications with schedulers will be almost straight forward with Singularity as the Singularity runtime can be called with a simple commandline and does not require any dependancy, whereas Docker requires a daemon running on the host. In all cases, Singularity can be used through a simple script wrapper in case we need to set up some specific environment. This actually explains the rapid adoption of Singularity in all kind of data centers.

4.2 Orchestrator: the Kubernetes era

Orchestrators have been designed to manage containerized workloads, providing primarily resource management, but also a lot of interesting features such like service reliability,

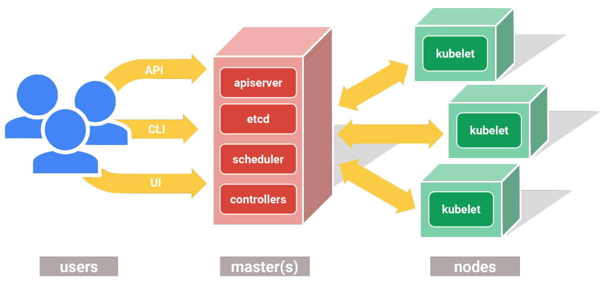

Figure 3. Schematic view of the Kubernetes architecture at Google [17].

isolation or scalability. However, they are more suited for hosting long running services, not computing jobs. Indeed they have been designed to manage micro-service containers such as web services in production. Consequently, they lack finite job scheduling mechanism (no management of potential constraints and links between jobs, input data and output data), and multi-users management.

Among the various orchestrators solution, Kubernetes [18] is the world's most popular production-grade container orchestration platform (see Fig. 3), and the most important project of the Cloud Native Computing Foundation (CNCF) [19], which provides the following advantages:

- velocity: evolving quickly, while staying available;
- scalability: services are replicated and they support auto-scaling using pre-defined configurations;
- portability: applications deployed using Kubernetes can be easily transferred between environments;
- efficiency: applications can be co-located on the same machine without impacting the application themselves due to containerization.

In order to propose a job scheduling mechanism, Kubernetes has a job processing feature allowing to run jobs. Since a couple of years, Kubernetes is indeed evolving in a way that containerized HPC workloads will soon be able to be run in a Kubernetes cluster [20]. Furthermore, Sylabs [2] is currently working on enabling Singularity containers orchestration with Kubernetes [21].

5 Conclusion

Containers have proven that they are able to provide a pragmatic and efficient solution to the problem of packaging complex software dependencies into self-contained, ready-to-run executable runtimes that can be easily deployed in a portable way. Singularity, in particular, provides an HPC-friendly container runtime that streamlines adoption in data centers.

The ComputeOps project goal is to explore the benefit of the usage of containers for the IN2P3 scientific communities, suggesting tools and recipes to ease the use of these technologies.

In this context, the group previously set up a private Singularity Hub [10], and is currently working on a Comprehensive Software Archive Network.

We also provide recipes and documentation to use the various tools required to build images, to deploy or store those images, to run containers while meeting scientific contraints. We also provide tutorials on these technologies.

Ultimately the project aim is to facilitate the use of such technologies when defining computing models for new astronomy and particle physics research collaborations and groups at the french CNRS/IN2P3 research institute.

6 Acknowledgements

The project is founded by the french CNRS/IN2P3 institute and is part of the DecaLog Master Project for transversal Research & Development in Computing and Data.

References

[1] C. Cavet, A. Bailly-Reyre, D. Chamont, O. Dadoun, A. Dehne Garcia, P.-E. Guérin, P. Hennion, O. Lodygensky, G. Marchal-Duval, E. Medernach, V. Mendoza, J. Pansanel, R. Randriatoamanana, A. Sartirana, M. Souchal and J. Tugler, EPJ Web of Conferences **214**, 07004 (2019) CHEP 2018.
[2] https://sylabs.io/
[3] https://www.docker.com/
[4] https://about.gitlab.com
[5] https://hub.docker.com/
[6] https://singularity-hub.org/
[7] https://in2p3.cnrs.fr/fr/institut-national-de-physique-nucleaire-et-de-physique-des-particules-0
[8] https://github.com/quay/clair
[9] https://github.com/aquasecurity/trivy
[10] https://sregistry.in2p3.fr
[11] https://cernvm.cern.ch/portal/filesystem
[12] D. Dykstra and J. Blomer, Journal of Physics: Conference Series **513**, 042015 (2014).
[13] https://cran.r-project.org/
[14] https://www.cpan.org/
[15] https://www.ctan.org/
[16] http://guix.gnu.org/
[17] Initially taken from the Kubernetes Project webpage https://kubernetes.io/
[18] https://kubernetes.io/
[19] https://www.cncf.io/
[20] https://kubernetes.io/blog/2017/08/kubernetes-meets-high-performance/
[21] https://sylabs.io/guides/cri/1.0/user-guide/k8s.html