ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

THE STATE OF CONTAINERIZATION IN CERN ACCELERATOR CONTROLS

R. Voirin*, T. Oulevey, M. Vanden Eynden, CERN, Geneva, Switzerland

Abstract

In industry, containers have dramatically changed the way system administrators deploy and manage applications. Developers are gradually switching from delivering monolithic applications to microservices. Using containerization solutions provides many advantages, such as: applications running in an isolated manner, decoupled from the operating system and its libraries; run-time dependencies, including access to persistent storage, are clearly declared. However, introducing these new techniques requires significant modifications of existing computing infrastructure as well as a cultural change. This contribution will explore practical use cases for containers and container orchestration within the CERN Accelerator Controls domain. We will explore challenges that have been arising in this field for the past two years and technical choices that we have made to tackle them. We will also outline the foreseen future developments.

CONTAINERS IN CONTROLS SYSTEMS

Containers in a Nutshell

Namespaces started being implemented in the Linux kernel in the 2000s. They provide isolation features on multiple levels: while the mount namespace prevents the process from accessing the rest of the Linux filesystem, the Process ID (PID) namespace creates an independent PID number space where the isolated process is given PID 1. There are eight namespaces in total: mount, PID, network, interprocess communication, time, time-sharing, user and cgroup.

Containers can be seen as industrialization of these isolation mechanisms, where the use of namespaces is concentrated in a single "containerization layer". A containerized application runs in an isolated manner, and requires its dependencies (libraries) to be embedded (Fig. 1).

The Open Container Initiative (OCI) provides three specifications defining how containerized applications are stored (the Image Format Specification), run (the Runtime Specification) and distributed (the Distribution Specification) [1].

CERN Use Cases

Having been used in industry for some time already, it was clear that the benefits brought by containers could translate to CERN's Accelerator Control system. In April 2020, a project was launched to introduce containers to the Accelerator Controls landscape, in order to bring added value in a variety of areas.

Firstly, is the ability to decouple from the underlying host operating system and the flexibility this brings. At CERN,

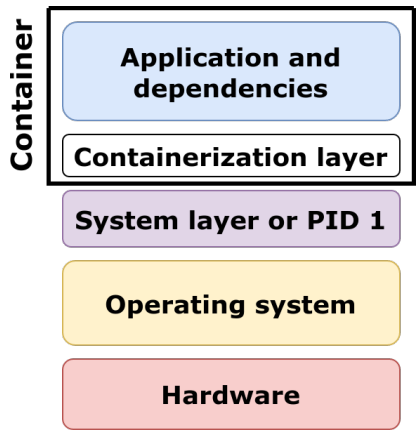

Figure 1: Overview of a containerized software stack.

WinCC OA is used to manage many industrial SCADA systems [2]. Version 3.16 officially runs on CentOS 7 and Red Hat Enterprise Linux (RHEL) 7, while 3.18 will be made to run on Red Hat Enterprise Linux 8. Due to the massive number of WinCC OA applications, many of them for critical systems, migration from version 3.16 to 3.18 must be applied progressively. In contrast, the operating system upgrade from CentOS 7 to the next platform, will concern all hosts at once. For this case, deploying WinCC OA in containers will make it possible to run version 3.16 in a CentOS 7-based container, while the underlying host is already upgraded to RHEL 8 or a derivative.

Containerization is also becoming a *de facto* standard for companies to deliver software to their clients. For example, SourceGraph [3] is used in the CERN Controls software community to index code, quickly search through it, and create statistics. All deployment options for self-hosted instances of SourceGraph are container-based.

Other software are delivered in container images for simpler deployment and upgrades. This is the case for the Nexus Repository Manager, which is used at CERN to manage Python libraries [4].

Another advantage of container-based software delivery is the idempotent behaviour of a product between development and operational environments. This can translate into two ways: streamlining the creation of development and operational releases in a similar way, and easily providing a containerized test setup. For the LHC Orbit Feedback (OFB), the latter has proven cost and time effective. Instead of running many test instances on a dedicated server and configuring them via a database, it is possible to run a local containerized copy and feed it directly with the desired parameters.

Containers are also a way to distribute software that can easily run regardless of the Linux distribution, or even the

* remi.voirin@cern.ch

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

operating system, as the Docker company created a desktop-oriented product for MacOS and Windows [5]. This has been successfully used for multiple PyQt training courses at CERN. It allows participants to run a particular Integrated Development Environment (IDE), faster, and with a lighter memory footprint than with a virtual machine.

Finally, there are many use cases where a clean and predictable software environment needs to be quickly generated, for instance in the context of Continuous Integration (CI) and Continuous Deployment (CD). A containerized environment is the most effective way to get this done.

Issues and Limitations

The differences between a virtual machine and an OCI container can be summarized in the most simple terms by the fact that running a virtual machine implies executing a guest operating system (kernel). Nevertheless, even in the absence of a kernel, container images need to provide shared libraries and basic building blocks like libc in order for an application to run properly. These components are provided by a Linux distribution, which means that they need to be managed and kept up to date.

While most containers will run the CERN supported Linux distribution, external containerized applications are typically made with others like Alpine and Debian. Importing such containers is similar to adding computers running these alternative distributions in the Controls environment. It therefore implies a need to track and address security issues for a broader set of software.

Another issue is the growing complexity of the overall software stack. In case of unexplained latency, lost packets, or unexpected behavior, debugging involves more components to analyze. For instance, using containers implies having virtual interfaces to forward packets to them. Capping CPU and memory usage involves the use of cgroups, which need to be understood and mastered in the system administration team. Using additional debugging techniques may be required, like eBPF [6].

Containerization is efficient for scaling and reproducing software instances. However, when using proprietary software, careful attention needs to be given to ensure licensing terms are not violated.

Required Components

In order to provide functional containerization in a Controls system, three components are required:

- a set of base images which will be used by developers as a foundation to package and deploy containerized software;
- a container registry to store images;
- a container engine, which is the piece of software made to run containers on hosts.

BASE IMAGES

Base images are made to replicate a Controls host (technical console or server) running CERN CentOS 7 in a lighter fashion. The "acc_cc7" base image is made from scratch by setting up a few packages (e.g. the YUM or DNF package manager, a text editor) in a directory, then loading the directory into the OCI image format.

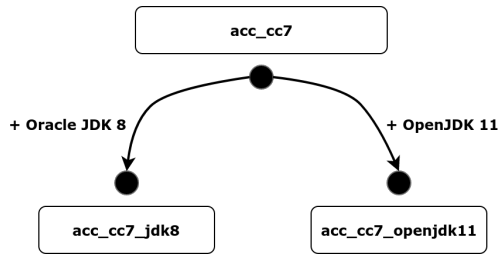

Figure 2: CERN CentOS 7 base images.

CERN Accelerator Controls images are configured to satisfy specific requirements, such as all Controls hosts running software from a set point in time called the "snapshot date" to give a homogeneous experience. This concept is applied within the container images thanks to a custom YUM repository configuration.

Images containing Java Development Kits (JDKs) are built on top of the acc_cc7 base image in order to run Java-based software, either with the Oracle JDK 8 or with OpenJDK 11 (Fig. 2). This is done by running an Ansible playbook which sets up the relevant Java packages and their configuration files within a temporary container, then extracting the resulting directory structure and placing it in a clean container image.

The overall process is launched every day in a CI/CD pipeline. It is light enough such that when a new major version of RHEL/CentOS is available, minimum changes are required to update the image generation process.

CONTAINER REGISTRY

Though the container registry can be described as simply as a repository of images, there are specific requirements that need to be satisfied. It must be able to store internal images (i.e. base images and containerized applications built on top), as well as images which come from public registries like the Docker Hub. Such images can not be pulled directly from Controls hosts, as they have no access to the Internet, hence the registry acts as curated proxy in this sense.

For security reasons, pushing images to the registry should be restricted to a specific process, while pulling should be done anonymously, allowing easy integration in automated software deployment processes. Following a market survey and discussing with other CERN teams working in this domain, it was decided to use the Harbor registry [7] (Fig. 3).

Harbor deployment is managed by the cloud infrastructure team within the IT Department. It fulfills the main functional requirements and it integrates well with additional security features such as an image validation policy and vulnerability scanning.

Validation Policy and Vulnerability Scanning

The Controls computing infrastructure team has to find a balance between delivering a smooth experience to develop-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

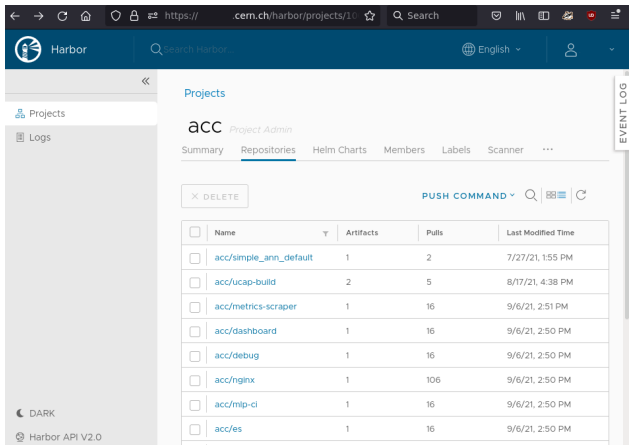

Figure 3: The Harbor registry.

ers, and at the same time, enforcing a security policy where only a certain set of images are allowed.

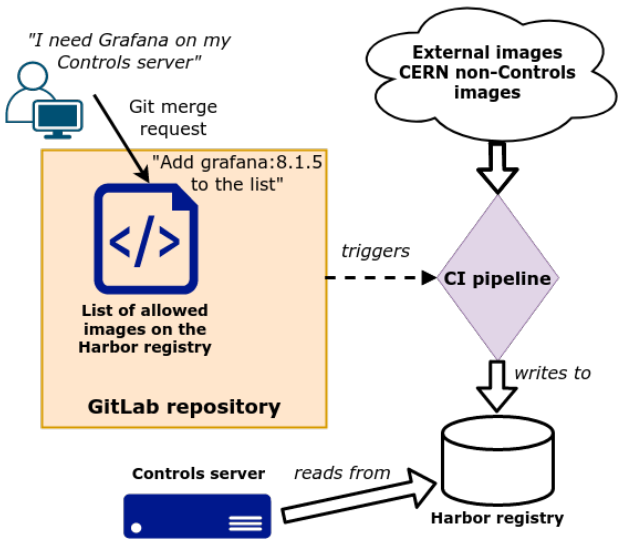

Figure 4: Image request and validation mechanism.

The Controls computing infrastructure team has designed a mechanism where developers can call a script which, after filling appropriate parameters, will create a merge request on a GitLab repository to add a new image to the Harbor-based Controls container registry (Fig. 4).

Only images that are relevant for operation are accepted. Later in the chain, vulnerability scanning is invoked from the Harbor registry, giving insight on the security status of the image. In 2021, Trivy [8] is the tool used to perform these scans (Fig. 5).

CONTAINER ENGINE

When the use of containerization technologies exploded in the 2010s, Docker was the default choice for a container engine. Since then, the company behind Docker has adapted its business model, whereby the use of some products need to be paid for [9]. This change in policy, combined with the

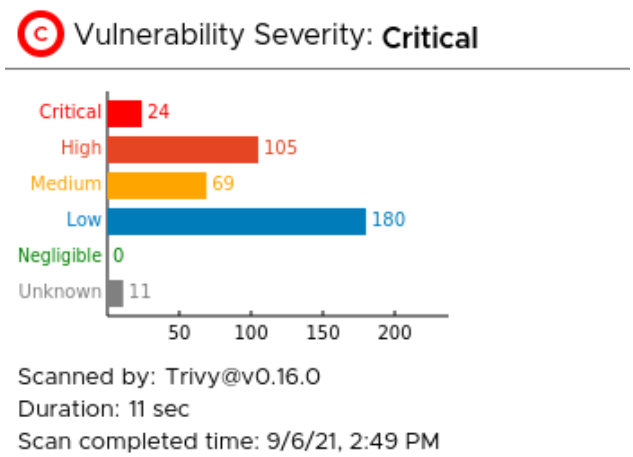

Figure 5: Scan result of a critically vulnerable image.

potential for a sudden price increase, raises concerns when it comes to choosing a container engine for Controls, that must remain stable over the course of a full LHC Run (e.g. 5 years).

Docker revolves around a UNIX daemon running as root. It creates an additional attack surface on every Controls host where it would be deployed, and also consumes some RAM even when no container is launched. It becomes a single point of failure where containers are children of this process, and in case of any issues, they become orphans (PID 1 becomes the parent).

In the meantime, OCI alternatives have emerged, like Podman [10]. This software is supported by Red Hat and is included in the RHEL distribution and its derivatives.

Podman does not require a daemon, which addresses the problems stated above. Being daemonless allows easy integration with LUMENS, the Controls tool to manage and run user processes. With only slight adjustments, it is possible to treat container startup similarly to any other systemd service.

Podman can be configured to run rootless containers, which means that they run completely within the scope of a UNIX account. This provides an increased level of security and practicality, as root may not be involved at all.

Rootless Containers

Even if all the required components are present (i.e. a recent enough Linux kernel supporting namespaces and the container engine to use them), some configuration is still required to enable rootless containers. First, user namespaces need be enabled. This is done thanks to the the max_user_namespaces kernel parameter, which is set to 0 by default. A high enough value needs to be set to accommodate as many containers as a Controls host can support. At CERN, this is set to 8192 giving room to run a few hundred containers.

During the daily Ansible run, which enforces the general configuration on Controls hosts, the list of users who can run rootless containers will be extracted from LDAP. This list is then written in the /etc/subuid and /etc/subgid

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

files in the proper format: <username>:<start of subuid range>:<number of subuids>. To avoid collisions, subuids and subgids are calculated by shifting the user ID 8 bits to the left, and leaving the last 11 bits for subuids and subgids (Table 1). Therefore, the number of subuids and subgids is constant for all users: $2^{11} - 1 = 2047$.

Table 1: Example of subuid range for a given UID

User ID	Beginning of range	End of range
1000	1000 << 11 = 2048000	2050047

Finally, user-writable directories in /opt are created to store images, containers and their metadata. This is done by again taking the list of relevant users in LDAP, then making sure these directories exist for each one of them with proper Ansible tasks.

CONTAINER ORCHESTRATION AND THE FUTURE OF THE ACCELERATOR DATA CENTER

Moving Away from Bare Metal

The operation of the CERN Accelerator complex relies on a dedicated data center that was put in operation for the LHC start-up in 2008. A total of about 400 high-availability servers are used for the operation of critical LHC infrastructure systems such as Cryogenics as well as for all systems required for beam operation.

Over the years, this critical infrastructure remained as a 100% bare metal facility. Consequently, each process required for the operation of the CERN Accelerator complex runs on a dedicated server with a fixed IP address. So far this infrastructure offered excellent availability figures, however, it is clear that it can be optimized to make better usage of the overall computing power and to be more agile in terms of hardware maintenance effort (the current annual system administration effort takes up to 4 days). This is where virtualization and container orchestration technologies enter the game.

Relevance of Container Orchestration in Controls

A simple way to summarize container orchestration is that instead of abstracting a single computer, an entire computing infrastructure is abstracted. An orchestrator takes care of the placement, scheduling, scaling, failover and health monitoring of containers. It also provides an API to interact with it, and accepts input with declarative configuration (generally in YAML format).

Stateless web services are one of the best use cases for orchestrated environments as they benefit from horizontal scalability, redundancy, and load balancing techniques without manual configuration of additional software like HAProxy and nginx.

Control systems are usually made of very diverse hardware and software with variety of monitoring and software deployment methods. Since it is not feasible for Controls infrastructure to be entirely deployed in orchestrated containers, it means that multiple deployment and mechanisms must be maintained, resulting in a growing stack and overall, a more diverse and complex infrastructure to manage.

Another point of concern is related to security: lots of institutions prevent their Control systems from directly accessing the Internet. It means that the orchestrator has to be physically deployed on-site, and managed by a local team. While deploying software on orchestrators is convenient and eases developers' lives, the underlying infrastructure is quite complex. It implies dedicating additional time to maintain it, increasing total costs, unless orchestrated containers completely replace the legacy stack.

Finally, one of the best selling points for container orchestration in industry is complete scalability. When a company relies on microservices that need to be replicated to manage sudden load increases, and for which the computing power can be rented from one of the big cloud providers, it is easy to understand the interest in this technology. Benefits are less apparent when servers have to be physically provisioned on site (i.e. they have a fixed cost), and that computing resource don't need to be regularly redistributed to different applications over time.

Previous Studies

There were two previous attempts at evaluating container orchestration for CERN Accelerator Controls in the late 2010s.

In 2018, the LHC Operations software team attempted to set up Kubernetes clusters to measure the performance of the LHC Coupling Analysis Service in containers [11]. Despite the use of infrastructure as code with Ansible, the complexity of deploying Kubernetes clusters in an environment without Internet connectivity was clearly highlighted.

Between 2018 and 2019, a study took place in the Accelerator Controls Group to evaluate container orchestration in a specific context, including the use of Nexus as an image registry, and creation of a proof-of-concept of base image. The conclusion, was that in-house deployment of Kubernetes was dismissed, due to human resources considerations and incompatibility with the accelerator schedule. Major Kubernetes versions were released every three months, whereas some Control system upgrades cannot be scheduled to take place more often than once per year. Upgrading Kubernetes on an annual basis could be quite risky, representing giant leaps across multiple versions.

CERN IT Container Orchestration Infrastructure

Outside of Controls, OpenStack is used as the backbone of the CERN IT data center since 2012. The IT container deployment platform is based on OpenStack technology, together with an open-source component called Magnum [12]. Magnum is an Openstack API service developed by the OpenStack Container Team, with numerous patches from CERN contributors. It allows to run orchestrators such as Docker Swarm or Kubernetes. This section will give more

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

details on the Kubernetes use case, which is the main orchestrator in use. Nevertheless, it is important to note that Magnum templates can be added for different orchestrators, without changing the underlying architecture and hardware.

CERN OpenStack [13] offers a self-service solution to get an orchestrator running in the private cloud. CERN users can request a cluster with various storage, compute (CPU, GPU) or traffic routing options. On the storage front, CERN offers support for CephFS [14], CVMFS [15] and EOS [16], which are quite different from industry standards.

Magnum can spawn a cluster on virtual machines or the bare metal infrastructure thanks to the integration with other OpenStack APIs. Once created, CERN users have full control on their clusters and can use industry standard tools to access the Kubernetes APIs and deploy their applications. Around 650 clusters are running at CERN [17], using around 13,000 cores, 30 TB of RAM and more than 150 PB of local storage. Under the hood, the operating system is Fedora CoreOS, which is an automatically-updating, minimal operating system for running containerized workloads securely and at scale.

Helm [18] is a package manager for Kubernetes and gives users the ability to deploy existing sets of containers with almost no code or knowledge of the underlying infrastructure. Users are encouraged to store and deploy their applications using the CERN internal repository.

For monitoring, the IT Department largely relies on Prometheus [19] for gathering cluster metrics, which is provided by default. Users can also add their own custom metrics.

This paper wouldn't be complete without a mention of the HashiCorp product, Nomad [20]. Nomad is a modern, easy to install and lightweight workload scheduler. By creating a pool of compute, storage, and networking, Nomad can decide where it's most efficient to run tasks. The deployment consists of a single binary and the built-in task drivers plug-in allows to run all sorts of workloads to also bring orchestration benefits to existing services. A few teams in the IT Department have put in place self-managed solutions based on the open source version of Nomad.

As shown in this section, container orchestration is a complex subject and requires interactions with a lot of existing IT infrastructure components. The investment to have something working in an environment that spans servers, networks, storage and development processes is very challenging for small teams.

Container Orchestration and Accelerator Data Center Management

In 2021, service continuity and recovery is considered as a high-priority topic with the CERN accelerator domain. One of the means to achieve this from the computing infrastructure perspective, is to allow critical services to easily recover, by starting them from a different data center.

Accelerator Controls back-end services are provided by around 400 servers in a single location. IP networks are physically segmented in the data center, no anycast range is available, and enabling routing protocols up to the host isn't available. Storage is also provided by NFS servers that are single points of failure.

Prior to introducing technical solutions for computing service continuity and flexible data center management, it will be required to invest in an evolution of the network infrastructure, as well as deploying scalable storage.

The hardware team has identified: key servers to physically duplicate, required rack space, power consumption, and expected recovery times from daily backups. It was also crucial to focus on logical aspects and how services and data can be transferred, or kept in a working state, when moving from one data center to another.

Virtualization and container orchestration were analyzed (Fig. 6). To sum up, all these technologies solve the same problem which is, providing service continuity in the case of a catastrophic event in a data center. They also enable the hardware team to manage servers in a more generic way, by looking at a global set of systems instead of individual treatment for each of them. Thanks to virtualization and container orchestration, physical computing resources can also be shared, using a logical split for different use cases. Currently, a team requiring back-end resources needs to ask for a full server.

While orchestrators provide APIs to deploy applications and offer features like load-balancing and autoscaling, they can only be used to host cloud-ready applications. Currently, only a few use cases of specific Controls sub-systems are ready e.g. the Machine Learning Platform (MLP) [21] and the Unified Controls Acquisition and Processing (UCAP). Stateless web services could be easily transferred as well. However, many Controls applications would need significant architectural changes. Also, one third of Accelerator data center servers are dedicated to running WinCC OA backends to control SCADA systems, and this specific software won't be ready for orchestration for some years to come.

From the development and deployment perspective, CI/CD integration is better with orchestrators, but they imply rethinking process deployment and monitoring, currently done using LUMENS. CI/CD integration would be the same between current servers and potential virtual machines.

Kubernetes, as provided by the CERN IT Department, runs as a proof-of-concept. It would need further refinements in order to run mission critical applications, such as the ability to run multiple cluster masters to mitigate the impact of localized hardware failure. Currently, if one server fails, orchestrated applications still run but can't be easily stopped or restarted. Other orchestrators like OpenShift and Nomad can be deployed at an additional cost, as training / consultancy for the system administration team would be needed in order to deploy an optimal setup.

Container orchestration will not avoid the fate of non-cloud-ready applications in the case of a disaster in the CERN Accelerator data center. Given the large number of use cases that cannot be orchestrated in the foreseeable future, flexibility should come with the use of virtual ma-

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THBL03

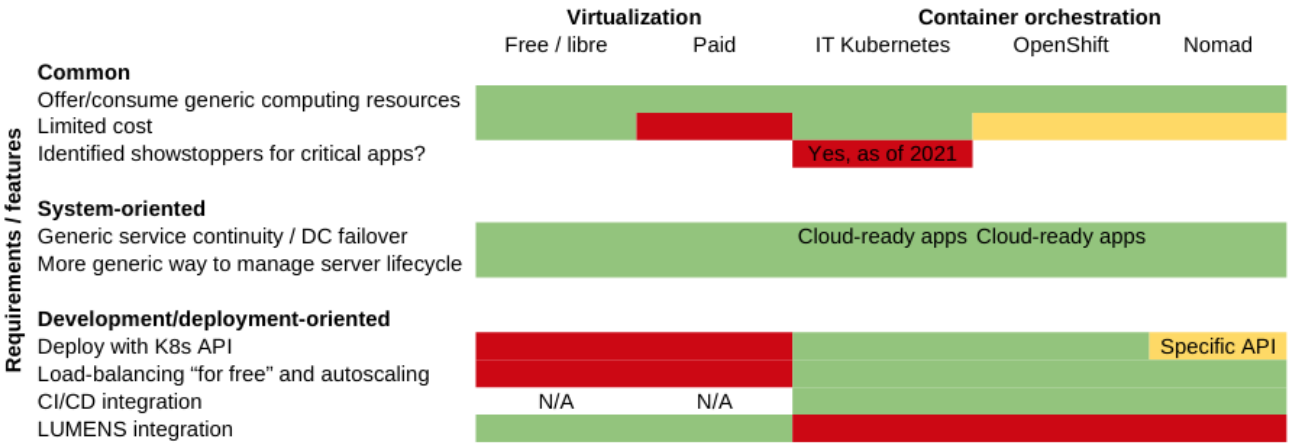

Figure 6: Summary of options for computing service continuity and flexible data center management.

chines, for which floating IPs will allow services to quickly continue in a second data center.

CONCLUSION

Containerization applies to different concepts and can be used on two different scales: plain containerization (by deploying on the current hardware and system stack), and container orchestration (by generically dedicating computing power to it).

At CERN, plain containerization is in place with fully functional base images, a container registry to store images, as well as a container engine configured to run rootless containers. It is used for operation and is fully supported.

Orchestration is deployed as a proof-of-concept, with software infrastructure provided by the IT Department, but it is currently not ready to run operational services. Introducing container orchestration on a massive scale implies a full review of software development practices. It can be used as a way to manage a data center in a more flexible manner, provided that the necessary techniques reach a critical mass adoption.

ACKNOWLEDGEMENTS

The author expresses his sincere appreciation to Karim Ben Othman for helping set up containerization blocks and writing effective Python libraries.

The author thanks Frank Locci for his work on integrating Podman containers in LUMENS, as well as Steve Traylen for providing a solution for management of subuids and subgids.

REFERENCES

[1] Open Container Initiative GitHub repository, https://github.com/opencontainers/.

[2] P. Golonka, L. Davoine, M. Zimny, and L. Zwalinski, "Virtualisation and Software Appliances as Means for Deployment of SCADA in Isolated Systems", presented at the 18th Int. Conf. on Acc. and Large Exp. Physics Control Systems (ICALEPCS'21), Shanghai, China, 2021, paper THPV049, this conference.

[3] Install Sourcegraph, https://docs.sourcegraph.com/admin/install

[4] Nexus3 container image, https://hub.docker.com/r/sonatype/nexus3/.

[5] Docker Desktop, https://www.docker.com/products/docker-desktop

[6] eBPF, https://ebpf.io/.

[7] Harbor, https://goharbor.io/.

[8] Trivy GitHub repository, https://github.com/aquasecurity/trivy

[9] Docker Subscriptions, https://www.docker.com/blog/updating-product-subscriptions/.

[10] Podman, https://podman.io/

[11] Marc Benedi and Andrea Calia, "Construction of a Kubernetes cluster in the CERN Technical Network and automating its deployment with Ansible", 2018. https://cds.cern.ch/record/2636772

[12] CERN Openstack Magnum git repository, https://gitlab.cern.ch/cloud-infrastructure/openstack-magnum

[13] CERN Openstack users documentation https://clouddocs.web.cern.ch/containers/README.html

[14] Container Storage and CephFS: CSI CephFS, https://techblog.web.cern.ch/techblog/post/container-storage-cephfs-csi-part1/.

[15] CVMFS CSI driver, https://gitlab.cern.ch/cloud-infrastructure/cvmfs-csi

[16] EOS storage provisioner, https://clouddocs.web.cern.ch/containers/tutorials/eos.html

[17] Statistics on Kubernetes clusters running in the CERN infrastructure, https://monit-grafana-open.cern.ch/d/ti8EaxwZk/kubernetes-public?orgId=16

[18] The package manager for Kubernetes, https://helm.sh

[19] Prometheus is an open-source systems monitoring and alerting, https://prometheus.io/.

[20] Nomad by HashiCorp, https://www.nomadproject.io/.

[21] J.-B. de Martel, R. Gorbonosov, and Dr. N. Madysa, "Machine Learning Platform: Deploying and Managing Models in the CERN Control System", 18th Int. Conf. on Acc. and Large Exp. Physics Control Systems (ICALEPCS'21), Shanghai, China, 2021, paper MOBL03, this conference.