A Comparative Analysis of Container Orchestration Tools in Cloud Computing

Anshita Malviya
Deptt. of Information Technology and Computer Application
Madan Mohan Malaviya University of Technology
Gorakhpur, India
anshitamalviya2010@gmail.com

Rajendra Kumar Dwivedi
Deptt. of Information Technology and Computer Application
Madan Mohan Malaviya University of Technology
Gorakhpur, India
rajendra.gkp@gmail.com

***Abstract*— Cloud Computing is an emerging technology that is used not only by developers but also by end-users. It has vital importance in the Information Technology (IT) industries as its future would create a great transition from conventional IT services. These days, containerization in cloud computing has become an important research area. The selection of container orchestration tools is one of the difficult tasks for the organizations involved in the management of the vast number of containers. These tools have their strengths, weaknesses, and functionalities which need to be considered. This paper presents a comparative analysis of the container orchestration tools. This analysis would help the professionals to decide whether they need an orchestrator bound to a single technology or an orchestrator which provides the independent solution. In this paper, four popular orchestration tools viz., Kubernetes, Docker Swarm, Mesos, and Redhat OpenShift are analyzed on various parameters viz., security, deployment, stability, scalability, cluster installation, and learning curve. We observed that Kubernetes has the best scheduling features whereas Docker Swarm is easy to use. We also found that Mesos has good scalability whereas OpenShift is a highly secure orchestration tool.**

Keywords— Cloud Computing, Container, Container Orchestration, Kubernetes, Docker Swarm, Apache Mesos, OpenShift.

I. Introduction

The application software and the resources that run on the Internet in spite of running locally on our computers constitute the cloud. The evolution of cloud computing took place from bundled software to resources and from consistent uses to dynamic clients [1]. Cloud computing facilitates the handling, arranging, and retrieving the hardware and software resources over the Internet. Creating, arranging, and personalizing the applications online are also allowed by it [1]. There is no need to install the software locally on our computer which means platform independence is also offered by cloud computing.

Deployment and service models make cloud computing viable and approachable. The type of access to the cloud is defined by the deployment models which are public, private, hybrid, and community. There are different service models on which cloud computing is based which are infrastructure-as-a-service (IaaS), Platform-as-a-service (PaaS), Software-as-a-Service (SaaS), Container-as-a-Service (CaaS) [1].

Online construction and deployment tools are offered by cloud computing. It is cost-effective and operates at high reliability, efficiency, and flexibility. If we want to use some external services, cloud computing acts as a trading subscription that is based on a pay-per-use model [2][3]. These days, we have noticed that there is enormous use and popularity of cloud computing which is supported by container technology [4]. In cloud computing, containerization helps in the management of applications by virtualizing them in a lightweight manner. In this paper, four popular orchestration tools viz., Kubernetes, Docker Swarm, Mesos, and Redhat OpenShift are analyzed on various parameters like security, deployment, stability, scalability etc.

The rest of the paper is organized as follows: The background of cloud computing is explained in Section II; Section III describes container orchestration and its platform; Section IV provides the analysis of orchestration tools; and Section V concludes the paper.

II. Cloud Computing

This section describes the background and importance of cloud computing and its different service models. At present time, cloud computing facilitates data accessing and storing in fast and efficient manner. Now, we can access our data online in comparison to accessing the data through local drives. Images, audios, documents, files, videos, etc constitute the term data. When people were unaware of the cloud computing technology, then the client-server architecture was used. At that time, if a user wanted to access their data, firstly that user had to connect to the server, and then only they could get their data. Then, the concept of distributed computing came which solved the above issue. This computing involves networking of all the computers and shared access of resources to the users when needed.

But, distributed computing also involves some limitations. Then, cloud computing came into existence which solved all the problems and limitations making data access and backup easier. It helps us to quickly and easily store information at any time and any place. Data accessing via mobile also became possible because of it. Cloud computing helps in the reduction of software and hardware price.

The basic types of service models of cloud computing are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [5]. Managing infrastructure over the internet, reducing cost, and avoiding the complexity in buying physical services are provided by the IaaS service model. PaaS provides computing platforms to the developers so that they could develop, run, test, and manage their software. Applications are provided to the users on their demand by cloud service providers with the help of the SaaS model. Containers, Serverless, Microservices, Internet of Things, and Artificial Intelligence are the emerging technologies that can be associated with cloud computing. Transforming industries with these latest technologies is possible because of the revolution brought by cloud computing. The world of computing is changing with the help of these technologies.

III. Container Orchestration Tools in Cloud Computing

In cloud computing, virtualization of the operating system is done by an approach known as Container. The deployment and management of software can be achieved with the help of containers. It runs the software applications in a remote environment by packing all the application's dependencies. The applications which are containerized can easily be drifted towards the cloud. The APIs given by the orchestration can be used to deploy, modify or clone the container easily and efficiently. Microservices, modernization of applications, DevOps, hybrid, and multi-cloud are some of the use cases favorable for accessing containers in the cloud [6], [7]. In this section, container orchestration and some of its tools are discussed on basis of their features, architecture, advantages, and disadvantages.

Nowadays, several industries have adopted the concept of containers rather than virtual machines because of their lightweight, portability, and agility to handle virtualization [8], [9], [10]. Containers can be divided into four categories, that is, System container, Application container, Container manager, and Orchestrator [11]. Kubernetes, Docker Swarm, Marathon, and Cloudify are some of the orchestrators. LXC, OpenVZ, WHC are examples of System containers, and Docker, LXC, WSC are examples of the application container. Container manager is further divided into On-premise and managed. Docker, LXD, OpenVZ, rkt are applications of On-premise and ECS, GCE, ACS come under Managed container manager.

Container orchestration refers to the accounting, deployment, and supervision of software applications that are containerized [12]. It is also used for the administration of the lifecycle of containers shown in Figure 1.

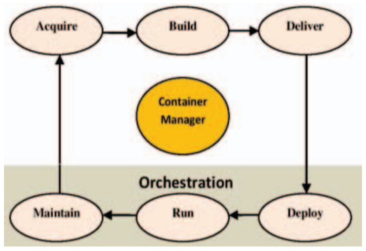

Fig. 1. Lifecycle of Container

Acquire, Build, Deliver, Run, and Maintain are the elements that constitute a container's lifecycle. Kubernetes, Docker Swarm, Apache Mesos, RedHat, and OpenShift are some of the orchestration tools. Among these the most widely used orchestration platform in the IT industry is Kubernetes.

Container orchestration is used by IT industries for container scaling, moving containers from one host to other in case of the dearth of resources in a particular host, allocating resources among containers, load balancing of services among containers, and many more. Deploying and managing hundreds of containers is a very tedious job, that is why container orchestration came into the picture. It reduces the complexity of the development of software applications. Simplified operations, resilience, and security are some of the major benefits of container orchestration.

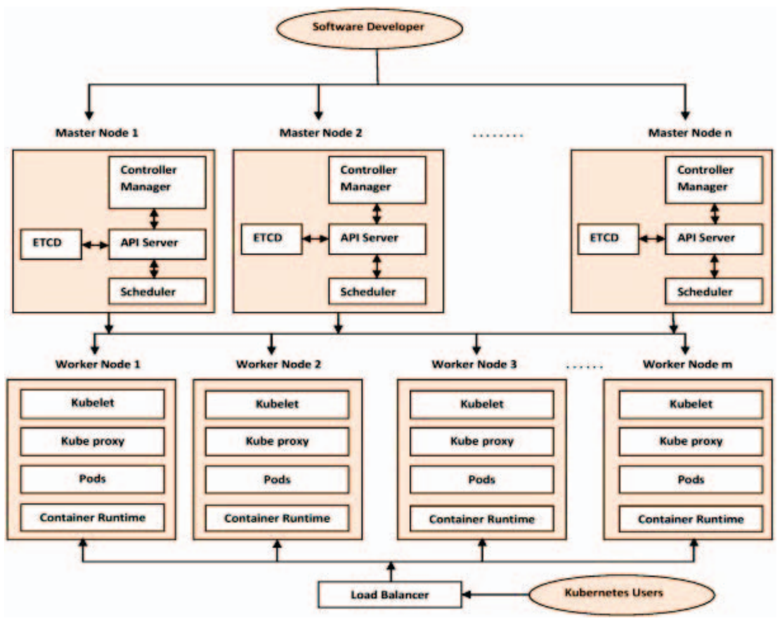

Fig. 2. Architecture of Kubernetes

A. Kubernetes

It is an open-source orchestration tool of containers developed by Google in 2008. In 2015, this tool was given by Google to Cloud Native Computing Foundation. Declarative automation and configuration are supported by Kubernetes. At present, it is the most widely used orchestration tool [13], [14]. The containers are orchestrated using YAML and JSON files in Kubernetes. The architecture of Kubernetes is depicted in Figure 2. Pods, nodes, and clusters are instigated in this tool. Containers are muffled into an edifice called pods where they can share their resources as well as local networks maintaining segregation among them.

In this orchestration platform, pods are also known as replication units which scale up and down as a unit. Pod instances run on nodes that act as a single machine. Cluster refers to the master machine where various nodes captivate resources together. Amazon elastic container service, azure Kubernetes services, rancher, istio, cloudify, knative, VMware tanzu, Redhat Openshift container platform, and Google Kubernetes engines are Kubernetes-as-a-service providers constructed on top of the platform of Kubernetes [15]. Storage is a challenging issue for Kubernetes because of its extensive nature.

Building of application services that reach several containers, scheduling, scaling, and managing health over time of those containers are all allowed by Kubernetes orchestration. This tool abolishes the manual deploying and scaling of containerized applications [16]. The main components of Kubernetes are cluster, control plane, pod, nodes, replication controllers, and labels. This platform offers extensive tools which give flexibility, ease of use, maximum productivity. It has the capability of managing microservices.

There are more than 109 tools that have container managing capabilities but among them, 89% use different configurations of Kubernetes according to a survey report of CNCF's Cloud Native Landscape. There are several features of Kubernetes such as load balancing, the discovery of service, orchestration storage, scaling, automatic rollouts and rollbacks, self-healing, management of secret and configuration, batch execution, IPv4/IPv6 dual-stack, and automated bin packing.

B. Docker Swarm

The building of containers is done by a specific platform known as Docker which includes Docker engine container runtime. As we know automation of a container's life cycle refers to container orchestration. In the Docker platform, the container orchestration tool which starts Docker containers automatically is the Docker swarm. Swarm refers to a group of physical and virtual machines that work together to run different Docker applications [17]. It is a fully assimilated and open-source orchestration tool.

Docker Swarm's architecture is shown in Figure 3. Docker swarm is used to package and run the applications as containers, deploy the containers and detect their images from other hosts. This tool came in 2003, years before Kubernetes. This leads to the popularization of the concept of container orchestration for organizations that wanted to use containers in place of virtual machines.

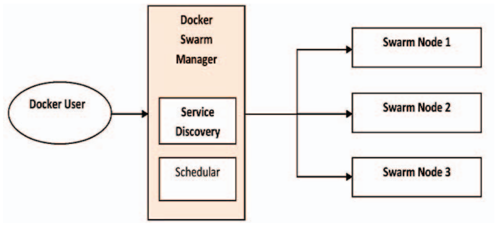

Fig. 3. Architecture of Docker Swarm

Docker swarm is suitable for organizations that give preference to easy orchestration for smaller applications. Both the Kubernetes and Docker swarm could be integrated by the organization to use the best of both tools and this has been done by Docker Enterprise Edition. Because of that now Docker provides flexibility in the choice of orchestration engine. Dockers do not run on Linux platforms and they have an issue while linking containers to storage, highly portable and aerobatic applications are allowed by Docker Swarm.

Docker swarm is also redundant to offer high availability to the applications. Proper load balancing of Docker applications is taken care of by Swarm managers. When the workload increases the swarm managers ensure high scalability by bringing up worker nodes. Decentralization of access and collaboration are allowed by Docker swarm's distributed environment [18].

The interactions with the clusters are done by Docker commands. The joining of clusters is done by a machine called nodes and the activities of clusters are handled by the swarm manager. The two main components of the Docker swarm are the manager and worker node. Management of cluster integrated with Docker engine decentralized design scaling, declarative service model, multi-host networking, load balancing, state reconciliation are some of the main features of Docker swarm.

C. Apache Mesos

Mesos was instigated by some Ph.D. scholars at the University of California, Berkeley. Its original name was Nexus but was renamed later to Mesos. Apache software foundation declared its first version on 27 July 2016. Apache Mesos is an open-source cluster management tool that performs container orchestration efficiently [19]. The components of Mesos are presented in Figure 4. It is a lightweight cross-platform with high availability and, is easy to use. Linux, Windows, and Mac operating systems are suitable for Mesos and C++, Python, Java are the languages supported by its APIs. This platform provides allocation and sharing of resources in distributed frameworks.

Crone's schedulers are used by Mesos for starting and stopping services. For scaling services and balancing loads, this tool uses Marathon API. Several software projects are built on Mesos by Apache such as Avrora, Batch scheduling, solutions of data storage and big data processing, marathon, singularity, and much more [20], [21].

Linear scalability, GUI, various APIs for constructing applications, LXC segregation allying tasks, controlling consolidation for fault-tolerant replication of master are some of the main features of Apache Mesos. For running workloads such as Big data, etc clustering of virtual machines or physical machines can also be done by Mesos as it is not only a tool for containers.

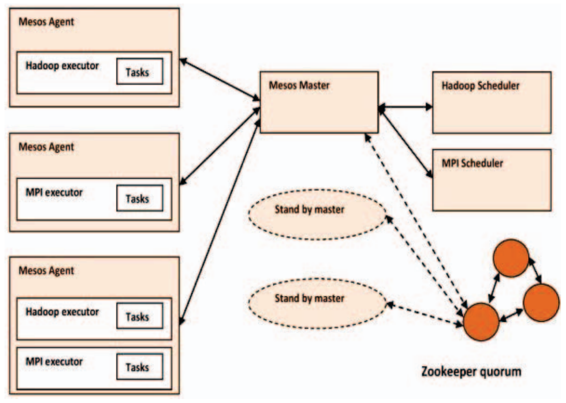

Fig. 4. Architecture of Apache Mesos

Marathon is a framework in this tool that is used for managing and deploying containers on a cluster efficiently. Kubernetes cluster can also run on the Mesos cluster. The main components in Mesos architecture are Master Mesos, Master Agents, Marathon API, and scheduler. Apache Mesos is used by various organizations like Apple, Netflix, eBay, and many more.

D. OpenShift

Red Hat has developed many containerized software products and OpenShift is among them. Kubernetes services are extended by OpenShift which is a hybrid and business-class platform. OpenShift framework is constructed on business class Linux Operating Systems. The lifecycle of the containerized program is automatized by it [22]. Databases can be easily be created by OpenShift.

The marketplace of Red Hat helps to purchase many guaranteed applications which in turn helps in billing, visibility, administration, and responsive support [23]. Platform-as-a-Service (PaaS) and Container-as-a-Service (Caas) both are offered by OpenShift. Faster production, Integrated Container embedded operator Hub, open-source and vendor-skeptic platform are some of the salient features of the Red Hat OpenShift platform. Figure 5 and 6 show the layered and component architectures of Redhat OpenShift, respectively.

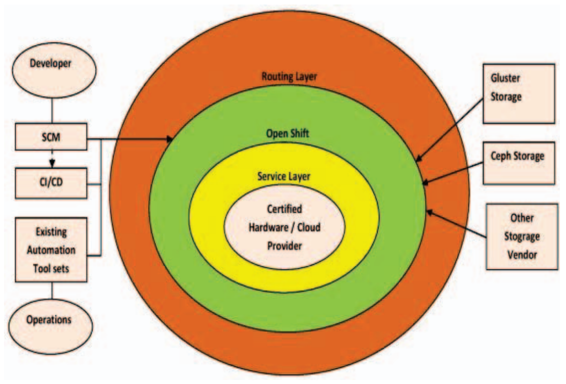

Fig. 5. Layered Architecture of OpenShift

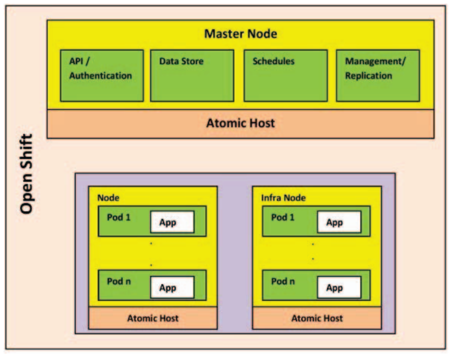

Fig. 6. Component Architecture of OpenShift

TABLE I. COMPARISON OF KUBERNETES, DOCKER SWARM, MESOS, AND OPENSHIFT

S. No.	Parameters	Kubernetes	Docker Swarm	Mesos	OpenShift
1.	Initial Release Date	July 2015, V1.16 in Sept. 2019	March 2013, Stable release July 2019	July 2016, Stable release August 2019	4th May 2011, Stable release Oct. 18, 2021
2.	Deployment	Almost any platform	Linux, Windows, and macOS	Ubuntu, Debian Jessie, CentOS	RHELAH, Fedora, or CentOS
3.	Security	Does not come with built-in authentication or authorization capabilities	Based on Network Level Security	Does not provide authentication by default	Strict Security policies
4.	Maturity/Stability	Very mature	Mature but still evolving	Very mature, especially for very clusters counting in the thousands of servers	Evolving
5.	Scalability	Medium to large clusters	Small to medium scale clusters	Large to Very large scale clusters	Small to medium scale clusters
6.	Cluster Installation	Slightly complex to set up.	Very easy to install and setup	Generally easy to install and set up but considerably more complex with large setups.	Easy to setup
7.	Best features	Best pods scheduling features	Easy to use and more native to docker	Scale in 1000s and Rack/Host-based constraints	Easy scaling and deployment
8.	Images Supported	Docker and rkt, restricted	Docker-image format	Mostly Docker-image	OCI, Docker-container image
9.	Learning Curve	Rapid	Easy	Rapid	Easy
10.	Support	Large active community	-	-	Small community
11.	Product vs. Project	Open-source project	Open Source Plateform	Open Source Framework	Open Source, Commercial Product

It is built on the top of Kubernetes. Both, community and business version of OpenShift is available. The projects of OpenShift are maintained by Red Hat. For container management and orchestration it provides. Kubernetes features as well as out-of-the-box components. On AWS cloud, Openshift is found as a managed service. For deployment of applications OpenShift online project of OpenShift is used. Openshift online is a PaaS service. The other project of OpenShift is OpenShift Dedicated which is provided as a managed service.

IV. Comparative Analysis of Orchestration Tools

In this section, we summarized the differences among four container orchestration tools using various parameters. Table I depicts a comparative analysis of Kubernetes, Docker Swarm, Apache Mesos, and OpenShift platforms. It is seen that OpenShift is the oldest platform among them released on 4th May 2011 but it is still evolving and its latest version came on 18th of October, 2021. Kubernetes platform can run in almost every operating system but others are restricted to some operating systems.

OpenShift offers more security than Kubernetes. Docker Swarm is the least popular orchestration tool among them. The Software development community as well as the practitioners and researchers use to decide their orchestration tools according to their needs and application complexity. The growth of orchestration platforms of containers will increase with the new technologies of cloud deployment.

V. Conclusion and Future Scope

We analysed four popular orchestration tools viz., Kubernetes, Docker Swarm, Mesos, and Redhat OpenShift on various parameters viz., security, deployment, stability, scalability, cluster installation, and learning curve. We observed that Kubernetes has the best scheduling features whereas Docker Swarm is easy to use. We also found that Mesos has good scalability whereas OpenShift is a highly secure orchestration tool. We also noticed that Kubernetes is the most popular and widely used platform among the tools discussed in this paper. We also found that the second most popular orchestration tool in the IT industries is OpenShift.

A good technical skill and knowledge of these platforms are required to use these tools. Therefore, we can do research on their architecture to make them developer and user friendly. We can also work on deployment, management, and development of containerized applications. Some more container orchestration tools can be analysed in terms of the quality of the deployed applications

References

[1] M. D. Neto, “A Brief History of Cloud Computing,” ibm.com. [Online]. Available: https://www.ibm.com/blogs/cloud-computing/2014/03/18/a-brief-hist ory-of-cloud-computing-3/. (Accessed Sep. 1, 2021).

[2] A. Modak, S. D. Chaudhary, P. S. Paygude and S. R. Idate, “Techniques to Secure Data on Cloud: Docker Swarm or Kubernetes,” in Proc. of the 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018), pp. 7-12, April 2018, doi:10.1109/icicct.2018.8473104.

[3] C. Cerin, T. Menouer, W. Saad and W. B. Abdallah, ”A New Docker Swarm Scheduling Strategy,” in Proc. of the 2017 IEEE 7th International Symposium on Cloud and Service Computing, pp. 112-117, Nov. 2017, doi: 10.1109/ SC2.2017.24.

[4] L. Qilong and Y. Fang, “Multi-Algorithm Collaboration Scheduling Strategy for Docker Container,” in Proc. of the International Conference on Computer Systems, Electronics and Control (ICCSEC), pp. 1367-1371, Dec. 2017, doi: 10.1109/ICCSEC.2017.8446688.

[5] R. K. Dwivedi, “From Grid Computing to Cloud Computing and Security Issues in Cloud Computing,” Technia - International Journal of Computing Science and Communication Technologies-IJCSCT, vol. 5, no.1, pp. 805-809, July 2012.

[6] S. Singh and N. Singh, “Containers and Docker: Emerging roles and Future of Cloud Technology,” in Proc. of the 2nd International Conference on Applied and Theoretical Computing and Communication Technology(iCATccT), pp. 804-807, Jan. 2016, doi: 10.1109/ICATCCT.2016.7912109.

[7] D. Bernstein, “Containers and Cloud: From LXC to Docker to Kubernetes,” IEEE Cloud Computing, vol. 1, no. 3, pp. 81-84, Sept. 2014, doi: 10.1109/MCC.2014 .51.

[8] L. Herrera-Izquierdo, and M. Grab, “A Performance Evaluation between Docker Container and Virtual Machines in Cloud Computing Architectures,” MASKANA, CEDIA, 2017, vol. 7, pp. 127-133, 2017.

[9] C. Ruiz, E. Jeanvoire and L. Nussbaum, “Performance Evaluation of Container for HPC,” in Parallel Processing Workshops, vol. 9523, pp. 813-824, 2015, doi: 10.1007/ 978-3-319-27308-2_65.

[10] A. M. Potdar, D. G. Narayan, S. Kengond, M. M. Mulla, “Performance Evaluation of Docker Container and Virtual Machine,” Procedia Computer Science, vol. 171, pp. 1419-1428, 2020, doi: 10.1016/j.procs.2020.04. 152.

[11] E. Casalicchio, “Container Orchestration: A Survey,” in Systems Modeling: Methodologies and Tools, EAI/Springer Innovations in Communication and Computing, pp. 221-235, 2018, doi: 10.1007/978-3-319-92378-9_14.

[12] S. Hoque, M. S. de Brito, T. Magedanz, A. Willner, O. Keil, “Towards Container Orchestration in Fog Infrastructures,” in Proc. of the 2017 IEEE 41st Annual Computer Software and Applications Conference, pp. 294-299, July 2017, doi: 10.1109/ COMPSAC. 2017.248.

[13] N. Marathe, A. Gandhi and J. M. Shah, “Docker Swarm and Kubernetes in Cloud Computing Environment,” in Proc. of the Third International Conference on Trends in Electronics and Informatics (ICOEI 2019), pp. 179-184, April 2019, doi: 10.1109/ICOEI. 2019.8862654.

[14] I. M. A. Jawarneh, P. Bellavista, F. Bosi, L. Foschini, G. Martuscelli, R. Montanari and A. Palopoli, “Container Orchestration Engines: A Thorough Functional and Performance Comparison,” in Proc. of the IEEE International Conference on Communications (ICC). May 2019, doi: 10.1109/ICC.2019.8762053.

[15] T. Wei, M. Malhotra, B. Gao, T. Bednar, D. Jacoby, and Y. Coady, “No such thing as a “free launch”? Systematic benchmarking of containers,” in Proc. of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp. 1-6, Aug. 2017, doi: 10.1109/PACRIM. 2017.8121922.

[16] C. Zhu, B. Han and Y. Zhao, “A Comparative Study of Spark on the Bare metal and Kubernetes,” in Proc. of the 2020 6th International Conference on Big Data and Information Analysis (BigDIA), pp. 117-124, Dec. 2020, doi: 10. 1109/BIGDIA 51454, 2020.00027.

[17] B. Kang, J. Jeong and H. Choo, “Docker Swarm and Kubernetes Containers for Smart Home Gateway,” IT Professional, vol. 23, no. 4, pp. 75-80, doi: 10.1109/mitp.2020.3034116.

[18] Y. Pan, I. Chen, F. Brasileiro, G. Jayaputera and R. O. Sinnott, "A Performance Comparison of Cloud-Based Container Orchestration Tools,” in Proc. of the 2019 IEEE International Conference on Big Knowledge (ICBK), pp. 191-198, Nov. 2019, doi: 10.1109/ICBK-2019.00033.

[19] M. Moravcik and M. Kontsek, “Overview of Docker Container Orchestration Tools,” in Proc. of the 18th International Conference on Emerging eLearning Technologies and Applications (ICETA), Nov. 2020, doi: 10.1109/ICETAS 1985.2020.9379236.

[20] N. Zhou, Y. Georgiou, L. Zheng, H. Zhou and M. Pospieszny, “Container Orchestration on HPC Systems,” in Proc. of the 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), pp. 34-36, Oct. 2020, doi: 10.1109/CLOUD49709.2020 .0017.

[21] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lenge, C. A. F. De Rose, "Performance Evaluation of Container-based

Virtualization for High-Performance Computing Environments," in Proc. of the 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 233-240, Feb. 2013, doi: 10.1109/PDP. 2013.41.

[22] A. R. Putri, R. Munadi and R. M. Negara, "Performance Analysis of Multi Services on Container Docker, LXC, and LXD," Bulletin of Electrical Engineering and Informatics, vol. 9, no. 5, pp. 2008-2011, Oct. 2020, doi: 10.11591/ee.v9i4. 1953.

[23] M. Aly, F. Khomh, S. Yacout, "Kubernetes or OpenShift? Which Technology Best Suits Eclipse HONO IT Deployments," in Proc. of the 2018 IEEE International Conference on Service-Oriented Computing and Applications (SOCA), pp. 113-120, Nov. 2018, doi: 10.1109/SOCA.2018.00024.