Paper No: SE-15 **Systems Engineering**

Docker incorporation is different from other computer system infrastructures: A review

W. M. C. J. T. Kithulwatta*
Faculty of Graduate Studies,
Sabaragamuwa University of Sri Lanka, Sri Lanka
chiranthajtk@gmail.com

B. T. G. S. Kumara
Dept. of Computing & Information Systems, Fac. of Applied Sciences, Sabaragamuwa University of Sri Lanka, Sri Lanka
kumara@appsc.sab.ac.lk

K. P. N. Jayasena
Dept. of Computing & Information Systems, Fac. of Applied Sciences, Sabaragamuwa University of Sri Lanka, Sri Lanka
pubudu@appsc.sab.ac.lk

R. M. K. T. Rathnayaka
Department of Physical Sciences & Technology, Fac. of Applied Sciences, Sabaragamuwa University of Sri Lanka, Sri Lanka
kapilar@appsc.sab.ac.lk

***Abstract* - Currently the computing world is getting complex, innovating and maturing with modern technologies. Virtualization is one of the old concepts and currently containerization has arrived as an alternative and innovative technology. Docker is the most famous and trending container management technology. Different other container management technologies and virtualization technologies are respective other corresponding technologies and mechanisms for Docker containerization. This research study aims to identify how Docker incorporation is different from other computer system infrastructure technologies in the perspective of architecture, features and qualities. By considering forty-five existing literatures, this research study was conducted. To deliver a structured review process, a thorough review protocol was conducted. By considering four main research questions, the research study was lined up. Ultimately, Docker architecture and Docker components, Docker features, Docker integration with other computing domains and Docker & other computing infrastructures were studied. After synthesizing all the selected research studies, the cream was obtained with plenty of knowledge contribution to the field of computer application deployment and infrastructure.**

Keywords* - *computer infrastructure, containers, docker, virtualization, virtual machines

I. INTRODUCTION

Computer virtualization has existed for a long time. As well, virtualization is an old conceptualization within the computing domain. Traditionally, most information technology (IT) services are bound with hardware components and virtualization enables those services in a virtual manner [1]. A software component called hypervisor, creates separate physical resources in the virtual environment. The hypervisor keeps on top of an operating system and ultimately, the virtual machine makes the interaction between end-users and the computing system. Figure 1 presents the virtualization stack architecture.

On top of any hardware platform, an operating system was launched and on top of that operating system, the hypervisor was launched. On the hypervisor, each virtual machine carries the full functional operating system. Each virtual machine provides a separated environment for the software applications and services.

Within virtualization, each virtual machine has a heavy weight since a virtual machine has a full set of functional operating systems. Therefore, an alternative and novel concept was arrived called containerization. Within the containerization, containers play a major role.

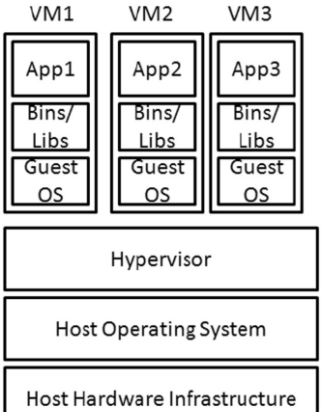

Fig. 1. Virtualization architecture [2]

Figure 2 presents the container architecture as a pictorial way [2]. According to the container architecture, it consists of a container engine instead of the hypervisor. On the container engine, each container keeps a packaged environment by including all fundamental dependencies to run the software applications. Each container provides an isolated environment for the software applications from the host computer infrastructure and other containers.

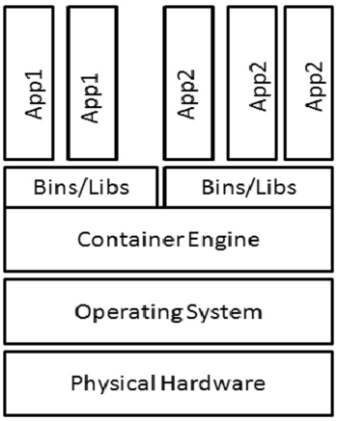

Fig. 2: Container architecture [2]

II. MOTIVATION

Within the practitioner of the containers, Docker is one of the available container management technologies. Other than Docker: Rkt and Linux containers are available as container technologies.

According to the official Docker documentation website, more than eleven million developers are engaged for Docker developments. As well, more than seven million Docker based software applications are made. More than thirteen billion Docker images are downloaded for the Docker based practitioner usages [3].

As mentioned in the official Docker documentation, most of the widely used computing tools are engaged with Docker containerization. Few of them are Bitbucket, GitLab, GitHub, NGINX, Redis, Jenkins, JFrog, MongoDB, Visual Studio Code, etc. [3].

Currently most industries and clients are using Docker oriented software applications and few of those clients are Paypal, Adobe, Netflix, University of Calgary, PathFactory, etc. [3]. Furthermore, Docker trusted contents are offered by Docker verified publishers as reliable Docker packaged blocks. Some of those publishers are Amazon Web Services (AWS), RedHat, Datadog, etc. [3].

Therefore, it depicts that Docker has higher practitioner engagement. Hence Docker is having a higher trend within the practical approach. Currently there is a higher competition for Docker among other container technologies and other infrastructure approaches like virtual instances. This research study was designed to identify the differences for Docker with other computer system infrastructure approaches.

The overall research study brings answers for the below research questions (RQs).

RQ1: *What kinds of components are embedded in the Docker architecture?*

RQ2: *What kind of benefits are available for the Docker based container approaches?*

RQ3: *What kind of computing areas/domains are integrated with Docker?*

RQ4: *How do Docker and other infrastructure approaches differ?*

III. Research Methodology

To obtain a thorough review analysis, the research study followed a highly structured review protocol. The ultimate review protocol is with eight steps. Table I presents the applied protocol as steps. The table I is with three columns. The first column presents the review protocol step number, second column presents the respective step name and third column presents the step in more descriptively.

TABLE I. REVIEW PROTOCOL IN STEPS

Step Number	Step Name	Step in Detail
Step 1	Need for the review	Identify the need for the review and the need was identified at section II.
Step 2	Research Questions	Declare the research questions and research questions were identified at section II.
Step 3	Identify the search strings	The search string was declared to select primary literatures. The identified search string was declared below (1).
Step 4	Primary literature selection	By using the identified search string, primary literatures were selected. The search string was browsed in the Google Scholar. Then primary studies were selected from the scientific databases including IEEE-Xplore, ACM Digital Library, Springer and Science Direct.
Step 5	Inclusion/ Exclusion	To filter the papers from the domain, the paper inclusion and exclusion criteria was applied.
Step 6	Quality Assessment	To filter the inapplicable literatures from the primary literature bulk, paper quality assessment was executed. After the process, 45 papers were finalized.
Step 7	Synthesizing	On top of the selected papers, the synthesizing was applied.
Step 8	Final reporting	By including the final observations, investigations and results of the research study, a final research report was made.

The search string:

$$(\text{Docker}) \wedge [(\text{infrastructure}) \vee (\text{cloud}) \vee (\text{containers})] \quad (1)$$

Other than the scientific databases, the official Docker documentation website was used as primary literature to identify the latest updates on Docker container technology.

For the review study, the research papers were selected by applying the search string. Docker container technology was introduced in 2013. Hence the selected literatures were published from 2014 to 2020. The table II presents the all referred literatures. The table II is with two columns: first column is for the study topic and second column is for the citation number.

TABLE II. THE LIST OF SELECTED LITERATURES

Topic of the Literature	Citation Number
What is Virtualization?	[1]
Exploring the support for high performance applications in the container runtime environment	[2]
Empowering App Development for Developers | Docker	[3]
Docker overview	[4]
Containers & Docker: Emerging roles & future of Cloud technology	[5]
Advantages of Docker	[6]
Performance comparison between Linux containers and virtual machines	[7]
An Introduction to Docker and Analysis of its Performance	[8]
Performance Comparison Analysis of Linux Container and Virtual Machine for Building Cloud	[9]
An updated performance comparison of virtual machines and Linux containers	[10]
Virtualization and containerization of application infrastructure: A comparison	[11]
A Comparative Study of Containers and Virtual Machines in Big Data Environment	[12]
Evaluation of Docker as Edge Computing Platform	[13]
Using Docker in High Performance Computing Applications	[14]
The research and implementation of cloud computing platform based on Docker	[15]
A Study of Security Vulnerabilities on Docker Hub	[16]
Evaluation of Docker Containers Based on Hardware Utilization	[17]
Docker Cluster Management for the Cloud - Survey Results and Own Solution	[18]

Topic of the Literature	Citation Number
Leveraging microservices architecture by using Docker technology	[19]
To Docker or Not to Docker: A Security Perspective	[20]
Measuring Docker Performance: What a mess!!!*	[21]
Containers and Cloud: From LXC to Docker to Kubernetes	[22]
Docker ecosystem – Vulnerability Analysis	[23]
Distributed Systems of Microservices Using Docker and Serfnode	[24]
Model-Driven Management of Docker Containers	[25]
Feasibility of Fog Computing Deployment based on Docker Containerization over RaspberryPi	[26]
Improvement of Container Scheduling for Docker using Ant Colony Optimization	[27]
Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research	[28]
Autonomic Vertical Elasticity of Docker Containers with ELASTICDOCKER	[29]
Integrating Containers into Workflows: A Case Study Using Makeflow, Work Queue, and Docker	[30]
DIVDS: Docker Image Vulnerability Diagnostic System	[31]
Orchestrating Docker Containers in the HPC Environment	[32]
A Docker Container Anomaly Monitoring System Based on Optimized Isolation Forest	[33]
An Empirical Analysis of the Docker Container Ecosystem on GitHub	[34]
Containers & Docker: Emerging Roles & Future of Cloud Technology	[35]
In Search of the Ideal Storage Configuration for Docker Containers	[36]
Measurement and Evaluation for Docker Container Networking	[37]
Building A Virtual System of Systems Using Docker Swarm in Multiple Clouds	[38]
A Defense Method against Docker Escape Attack	[39]
DoCloud: An elastic cloud platform for Web applications based on Docker	[40]
CoMICon: A Co-operative Management System for Docker Container Images	[41]
FID: A Faster Image Distribution System for Docker Platform	[42]
Orchestration of Containerized Microservices for IIoT using Docker	[43]
A Holistic Evaluation of Docker Containers for Interfering Microservices	[44]
Application deployment using Microservice and Docker containers:Framework and optimization	[45]

IV. Docker Architecture

The Fig. 3 presents the Docker architecture in a pictorial view. To design the fundamental Docker architecture, client and server architecture has been used. Docker daemon, Docker client and Docker registries are the main components for the Docker architecture [4

The Docker daemon has a main responsibility to manage Docker objects. Main Docker objects are containers, images, volumes and network. One Docker daemon can communicate with other Docker daemons. To make the interaction with Docker, Docker client was used as the fundamental way. While using the Docker commands on the Docker client, it sends those commands to Docker daemon. One Docker client can communicate with more than one Docker daemons [4].

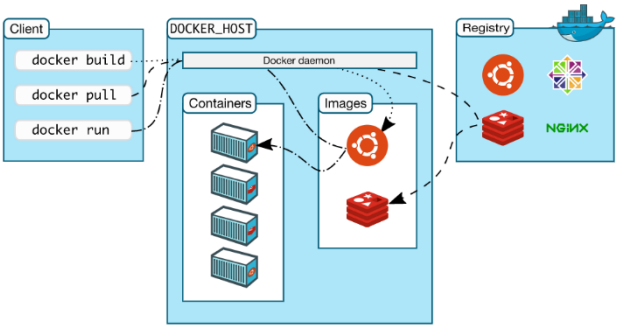

Fig. 3: Docker architecture [4]

The Docker daemon has a main responsibility to manage Docker objects. Main Docker objects are containers, images, volumes and network. One Docker daemon can communicate with other Docker daemons. To make the interaction with Docker, Docker client was used as the fundamental way. While using the Docker commands on the Docker client, it sends those commands to Docker daemon. One Docker client can communicate with more than one Docker daemons [4].

Docker client and Docker daemon can be executed on the same infrastructure. According to the designed way, Docker client can be connected to a remote Docker daemon [4].

To store and archive the Docker images, a dedicated location was allocated in the Docker architecture called Docker registry. According to the use-cases, publicly available Docker registry or private Docker registries can be used. Users can pull Docker images from the Docker registry or push the Docker images to Docker registry [4].

Docker image is one of the most important parts of the Docker architecture and it consists of a read-only template with a set of instructions to create a Docker container. By using a Dockerfile, specific Docker images can be created. As well, Docker container is the executable instance of a Docker image. By using Docker application programming interface or command line interface, Docker containers can be created, stopped, started, moved or deleted [4].

V. Docker Features

This section emphasizes the Docker related advantages, incorporations and compatibility of the Docker with other computing technologies.

A. Docker Advantages

Table III presents the Docker advantages in a more advanced way. The first column denotes the Docker advantages and the second column denotes the advantages more descriptively.

B. Docker Integrated Areas/Domains/Communities

The Docker containerized technology is not only dedicated as the software application deploying environment. According to the referred literature studies, Docker container technology was integrated with different computing domains and areas. As a summarized list, the below list presents those Docker engaged computing areas [1] - [45].

- Edge Computing
- Computer Networking

- Cloud Computing
- Computer Security
- Grid Computing
- Distributed Computing
- Operating Systems
- Web Engineering
- High Performance Computing
- System Engineering
- Internet of Things
- Autonomous Computing
- Parallel Computing
- Microservices

Hence, the above list depicts that Docker was spread in a variety of computing domains. Within the above domains, Docker was used as a runtime environment, virtualization and an operating system. Mainly, Docker was used for development, testing, deploying and experimenting purposes.

TABLE III. DOCKER ADVANTAGES

Docker Advantage	Advantage in more Descriptively
Lightweight	Docker is with lightweight containers and images than traditional virtual machines. Since traditional virtual machines carry a full set of operating systems, a virtual machine is heavy weight. Furthermore, one virtual machine consumes heavy resources from the host computer infrastructure to execute a full set of operating systems [5].
Portable	Docker containers and images can be moved as one module within any computer system infrastructure easily: therefore, Docker is portable. Due to the portability, Docker images can be shared with different hosts easily. However, traditional virtual machines can be moved within different hosts but it is more heavy and has to follow more steps [5] [6].
Scalability	Docker is providing a facility to scale the Docker containers and services by up and/or down the number of replicas. Docker takes the responsibility to upgrade or downgrade the number of replicas very smoothly, without making any effect on the software service. Therefore, Docker can be provisioned more easily than the virtual machines [7].
Best fit for microservices	According to the microservices architecture, software applications need a separated and isolated environment. Therefore, Docker makes an isolated environment within the Docker containers and it helps to give the best software environment for the microservices softwares. Without making any conflicts with other modules or components, Docker provides the best fit for microservices [7].
Optimal resource utilization	Docker container structure shares the host computer resources among the Docker objects. Docker has the facility to allocate limitations for each Docker object to utilize the host memory, CPU, disk space and network. Due to those limitations and constraints, Docker has optimal resource utilization [5].

As well, Docker container technology has presented an excellent research path in computing. Research scholars have presented that Docker brings a strong research direction.

For the government of Docker or other container farms, container orchestration solutions are needed. Therefore, Kubernetes has been identified as the best fit for Docker container orchestration with amazing and fantastic features & functions. Mainly identified Kubernetes features for Docker are automatic rollouts, automated roll backs, storage orchestration, load balancing, service discovery, configuration management, batch execution, horizontal scaling, self-healing, automated bin packing, etc.

C. Docker and Other Corresponding Approaches

Docker has been identified as the best computer infrastructure for the software application deployments. Other than Docker, there are different kinds of container management technologies and virtual environments. Most scholars have made different comparisons among Docker and other corresponding approaches.

Virtual machines use an extra layer called ***hypervisor*** and the hypervisor is between the host operating system and guest operating system (*Figure 1 presents the location of the hypervisor according to the virtualization architecture*). However, containers add up an additional layer between the host operating system and where the applications are virtualized and executed. It was known as a container engine. Since containers do not use any guest operating system, it makes a considerable performance difference between container technology and virtual machine technology [8].

Below tables IV, V and VI present the performances of different container vendors and virtual machines. According to the paper [9], Docker container performance is better than KVM (Kernel-based Virtual Machines) in terms of boot time and calculation speed [9]. But another research paper has proved that there is no difference of wastage of host resources between Docker and KVM but there is a noticeable difference in execution as KVM is faster than Docker containers [10]. The research paper [11] has presented that LXC (Linux Containers) takes a longer time to accomplish a defined task. But XenServer took less time than LXC. LXC is a better container in the sense of fewer wasted resources while Xen is better in the sense of equally distributing resources.

TABLE IV. DOCKER AND KVM

Reference: [9]	
Docker	**KVM**
Short boot time	Long boot time
Calculation speed is faster	Calculation speed is slower
No guest operating system	Works independently

According to the above summarized Table IV, KVM is working independently due to KVM having a hypervisor and Docker has no guest operating system. But Docker shares the host operating system resources.

As mentioned in the literature, LXC consumes less overhead on the parameter of host computer resources. Same as that, XenServer has consumed more overhead. The

author has identified that XenServer was better in the sense of distributing host computer resources equally. But LXC was not like that and LXC was better in the sense of executing fully isolated processors [11].

TABLE V. XEN-SERVER AND COREOS

Reference: [11]	
XenServer (Xen)	**CoreOS (LXC)**
More overhead (regarding wastage of resources)	Less overhead (regarding wastage of resources)
Less time to accomplish request	Longer time to accomplish request
Better in sense of equally distributing resources	Better in sense of executing isolated processes

According to the above summarized Table VI, Docker and KVM have presented a more mature innovation than native approach. As well, KVM has demonstrated very less host computer resources wastage than both native and Docker approaches.

TABLE VI. NATIVE, DOCKER AND KVM

Reference: [10]		
Native	**Docker**	**KVM**
Overhead (regarding wastage of resources)	Slightly less overhead than native	Slightly less overhead than native and Docker
Slow execution equal to Docker	Slow execution equal to native	Fast execution
-	Mature innovation	Mature innovation

Apart from the above comparisons, a recent research paper has presented differences between containers and virtual machines. A container consists of executable software application binaries and executable codes. All fundamentally necessary software dependencies need to run a container. Containers are using Linux kernel mechanisms to allocate resources. The authors have said that engineers can allow allocating resources for the containers like network configurations, CPU and memory at the time of container creation. The allocated resources may be adjusted dynamically but any container cannot use more resources than being specified [12].

The paper [12] has expressed that, the first difference between containers and virtual machines is: containers are more lightweight than virtual machines. The due reason is: containers include only executable applications and their dependencies. The containers which are on the same machine, share the host operating system resources among containers. Respective virtual machines do not share the host operating system resources. Virtual machines contain a full set of operating systems. Furthermore, the same paper [12] has presented that virtual machines can run as any operating system that is different from the host machine. But containers need to use the same operating system as the host machine.

The authors of the paper [12] have presented the second comparison on the hypervisor. For the virtual machine environment, the hypervisor is necessary to use such as VMware ESXi and KVM. It is not required for containers. Virtual machines are functioning as an independent machine by keeping all control of all resources under the virtual machines. Furthermore, virtual machines are running as non-privileged mode and containers are running on privileged mode. It depicts that virtual machines cannot execute many privileged instructions. As well as, for the execution of instructions, the virtual machine environment is needed to translate all virtual machine instructions to executable commands to which that needs to run on the host. However, containers make communication with the host directly by system calls and it does not require any intermediate mechanism to convert instructions [12].

Furthermore, the paper [12] has discussed image files of virtual machines and containers. Virtual machines have their own images and containers share some of their images. Container images are created as a layered architecture. To create an image on an existing image, the platform adds another layer on the original image. Image files of different virtual machines are isolated from each other [12].

The authors of [12] have presented their research findings as researchers and practitioners pay their attention to containers instead of virtual machines. Containers are more cost-effective. Furthermore, containers usually consist of tens of Megabytes (MB) while virtual machines can take about several Gigabytes (GB). To run an application, a container uses very fewer resources than virtual machines due to containers not needing to maintain an operating system. Containerized platforms do not contain any hypervisor and containers present more performance than virtual machines [12].

VI. CONCLUSIONS

Containerization was identified as an alternative for virtualization. Within the practitioner of the container management technologies, Docker keeps and plays a major role. Currently millions and billions of customer interactions are with Docker container management. Docker has client and server architecture. As well, Docker daemon, Docker client, Docker registry and Docker objects play main roles in the Docker platform. Those components and modules help to carry answers for the RQ1. Docker has many available features and benefits. Some of them are scalability, portability, lightweight, best fit for microservices and optimal resource utilization. Hence those features provide the answers to the RQ2.

Without limiting to software application launching on Docker, Docker containerization was engaged with many computing technologies. Few of them are fog computing, cloud computing, grid computing, Internet of Things, microservices, etc. Those are answered to the RQ3. As presented above in the *V.C* section, many of Docker and other infrastructure technologies were discussed. Hence those are answered to the RQ4.

The scholarly research articles present that Docker has a higher engagement with all kinds of computing technologies. Docker plays a major role in computer system administration engineering.

REFERENCES

[1] "What is Virtualization? ", 2021. [Online]. Available: https://www.redhat.com/en/topics/virtualization/what-is-virtualization [Accessed: 09- Jul- 2021].

[2] J. Martin, A. Kandasamy and K. Chandrasekaran, "Exploring the support for high performance applications in the container runtime environment", Human-centric Computing and

Information Sciences, vol. 8, no. 1, 2018. Available: 10.1186/s13673-017-0124-3

[3] "Empowering App Development for Developers | Docker", Docker, 2021. [Online]. Available: https://www.docker.com/. [Accessed: 09- Jul- 2021].

[4] "Docker overview", Docker Documentation, 2021. [Online]. Available: https://docs.docker.com/get-started/overview/. [Accessed: 09- Jul- 2021].

[5] S. Singh and N. Singh, "Containers & Docker: Emerging roles & future of Cloud technology," 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2016, pp. 804-807, doi: 10.1109/ICATCCT.2016.7912109.

[6] Vase, Tuomas. "Advantages of Docker." (2015).

[7] A. M. Joy, "Performance comparison between Linux containers and virtual machines," 2015 International Conference on Advances in Computer Engineering and Applications, 2015, pp. 342-346, doi: 10.1109/ICACEA.2015.7164727.

[8] B. B. Rad, H. J. Bhatti, M. Ahmadi, "An Introduction to Docker and Analysis of its Performance", IJCSNS International Journal of Computer Science and Network Security, vol. 17, no. 3, March 2017.

[9] K. Seo, H. Hwang, I. Moon, O. Kwon and B. Kim, "Performance Comparison Analysis of Linux Container and Virtual Machine for Building Cloud", 2014. Available: 10.14257/astl.2014.66.25.

[10] W. Felter, A. Ferreira, R. Rajamony and J. Rubio, "An updated performance comparison of virtual machines and Linux containers," 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2015, pp. 171-172, doi: 10.1109/ISPASS.2015.7095802.

[11] M. J. Scheepers, "Virtualization and containerization of application infrastructure: A comparison", 21st Twente Student Conference on IT, pp. 1-7, 2014

[12] Q. Zhang, L. Liu, C. Pu, Q. Dou, L. Wu and W. Zhou, "A Comparative Study of Containers and Virtual Machines in Big Data Environment," 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp. 178-185, doi: 10.1109/CLOUD.2018.00030.

[13] B. I. Ismail et al., "Evaluation of Docker as Edge computing platform," 2015 IEEE Conference on Open Systems (ICOS), 2015, pp. 130-135, doi: 10.1109/ICOS.2015.7377291.

[14] M. T. Chung, N. Quang-Hung, M. Nguyen and N. Thoai, "Using Docker in high performance computing applications," 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE), 2016, pp. 52-57, doi: 10.1109/CCE.2016.7562612.

[15] D. Liu and L. Zhao, "The research and implementation of cloud computing platform based on docker," 2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing(ICCWAMTIP), 2014, pp. 475-478, doi: 10.1109/ICCWAMTIP.2014.7073453.

[16] R. Shu, X. Gu and W. Enck, "A Study of Security Vulnerabilities on Docker Hub", Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, 2017, pp.269-280, doi: 10.1145/3029806.3029832

[17] Preeth E N, F. J. P. Mulerickal, B. Paul and Y. Sastri, "Evaluation of Docker containers based on hardware utilization," 2015 International Conference on Control Communication & Computing India (ICCC), 2015, pp. 697-700, doi: 10.1109/ICCC.2015.7432984.

[18] R. Peinl, F. Holzschuher and F. Pfitzer, "Docker Cluster Management for the Cloud - Survey Results and Own Solution", Journal of Grid Computing, vol. 14, no. 2, pp. 265-282, 2016. doi: 10.1007/s10723-016-9366-y.

[19] D. Jaramillo, D. V. Nguyen and R. Smart, "Leveraging microservices architecture by using Docker technology," SoutheastCon 2016, 2016, pp. 1-5, doi: 10.1109/SECON.2016.7506647.

[20] T. Combe, A. Martin and R. Di Pietro, "To Docker or Not to Docker: A Security Perspective," in IEEE Cloud Computing, vol. 3, no. 5, pp. 54-62, Sept.-Oct. 2016, doi: 10.1109/MCC.2016.100.

[21] E. Casalicchio and V. Perciballi, "Measuring Docker Performance", Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion, 2017, pp. 11-16, doi: 10.1145/3053600.3053605.

[22] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," in IEEE Cloud Computing, vol. 1, no. 3, pp. 81-84, Sept. 2014, doi: 10.1109/MCC.2014.51.

[23] A. Martin, S. Raponi, T. Combe and R. Di Pietro, "Docker ecosystem – Vulnerability Analysis", Computer Communications, vol. 122, pp. 30-43, 2018. doi: 10.1016/j.comcom.2018.03.011.

[24] J. Stubbs, W. Moreira and R. Dooley, "Distributed Systems of Microservices Using Docker and Serfnode," 2015 7th International Workshop on Science Gateways, 2015, pp. 34-39, doi: 10.1109/IWSG.2015.16.

[25] F. Paraiso, S. Challita, Y. Al-Dhuraibi and P. Merle, "Model-Driven Management of Docker Containers," 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), 2016, pp. 718-725, doi: 10.1109/CLOUD.2016.0100.

[26] P. Bellavista and A. Zanni, "Feasibility of Fog Computing Deployment based on Docker Containerization over RaspberryPi", Proceedings of the 18th International Conference on Distributed Computing and Networking, 2017, pp. 1-10 doi: 10.1145/3007748.3007777.

[27] C. Kaewkasi and K. Chuenmuneewong, "Improvement of container scheduling for Docker using Ant Colony Optimization," 2017 9th International Conference on Knowledge and Smart Technology (KST), 2017, pp. 254-259, doi: 10.1109/KST.2017.7886112.

[28] J. Cito and H. C. Gall, "Using Docker Containers to Improve Reproducibility in Software Engineering Research," 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), 2016, pp. 906-907.

[29] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah and P. Merle, "Autonomic Vertical Elasticity of Docker Containers with ELASTICDOCKER," 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), 2017, pp. 472-479, doi: 10.1109/CLOUD.2017.67.

[30] C. Zheng and D. Thain, "Integrating Containers into Workflows: A Case Study Using Makeflow, Work Queue, and Docker ", Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing, 2015, pp. 31-38 doi: 10.1145/2755979.2755984.

[31] S. Kwon and J. Lee, "DIVDS: Docker Image Vulnerability Diagnostic System," in IEEE Access, vol. 8, pp. 42666-42673, 2020, doi: 10.1109/ACCESS.2020.2976874.

[32] J. Higgins, V. Holmes and C. Venters, "Orchestrating Docker Containers in the HPC Environment", Lecture Notes in Computer Science, pp. 506-513, 2015. doi: 10.1007/978-3-319-20119-1_36.

[33] Z. Zou, Y. Xie, K. Huang, G. Xu, D. Feng and D. Long, "A Docker Container Anomaly Monitoring System Based on Optimized Isolation Forest," in IEEE Transactions on Cloud Computing, doi: 10.1109/TCC.2019.2935724.

[34] J. Cito, G. Schermann, J. E. Wittern, P. Leitner, S. Zumberi and H. C. Gall, "An Empirical Analysis of the Docker Container Ecosystem on GitHub," 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), 2017, pp. 323-333, doi: 10.1109/MSR.2017.67.

[35] S. Singh and N. Singh, "Containers & Docker: Emerging roles & future of Cloud technology," 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2016, pp. 804-807, doi: 10.1109/ICATCCT.2016.7912109.

[36] V. Tarasov et al., "In Search of the Ideal Storage Configuration for Docker Containers," 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W), 2017, pp. 199-206, doi: 10.1109/FAS-W.2017.148.

[37] H. Zeng, B. Wang, W. Deng and W. Zhang, "Measurement and Evaluation for Docker Container Networking," 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2017, pp. 105-108, doi: 10.1109/CyberC.2017.78.

[38] N. Naik, "Building a virtual system of systems using docker swarm in multiple clouds," 2016 IEEE International Symposium on Systems Engineering (ISSE), 2016, pp. 1-3, doi: 10.1109/SysEng.2016.7753148.

[39] Z. Jian and L. Chen, "A Defense Method against Docker Escape Attack", Proceedings of the 2017 International Conference on Cryptography, Security and Privacy - ICCSP '17, 2017, pp. 142-146, doi: 10.1145/3058060.3058085.

[40] C. Kan, "DoCloud: An elastic cloud platform for Web applications based on Docker," 2016 18th International Conference on Advanced Communication Technology (ICACT), 2016, pp. 1-1, doi: 10.1109/ICACT.2016.7423439.

[41] S. Nathan, R. Ghosh, T. Mukherjee and K. Narayanan, "CoMICon: A Co-Operative Management System for Docker Container Images," 2017 IEEE International Conference on

Cloud Engineering (IC2E), 2017, pp. 116-126, doi: 10.1109/IC2E.2017.24.

[42] W. Kangjin, Y. Yong, L. Ying, L. Hanmei and M. Lin, "FID: A Faster Image Distribution System for Docker Platform," 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS*W), 2017, pp. 191-198, doi: 10.1109/FAS-W.2017.147.

[43] J. Rufino, M. Alam, J. Ferreira, A. Rehman and K. F. Tsang, "Orchestration of containerized microservices for IIoT using Docker," 2017 IEEE International Conference on Industrial Technology (ICIT), 2017, pp. 1532-1536, doi: 10.1109/ICIT.2017.7915594.

[44] D. N. Jha, S. Garg, P. P. Jayaraman, R. Buyya, Z. Li and R. Ranjan, "A Holistic Evaluation of Docker Containers for Interfering Microservices," 2018 IEEE International Conference on Services Computing (SCC), 2018, pp. 33-40, doi: 10.1109/SCC.2018.00012.

[45] X. Wan, X. Guan, T. Wang, G. Bai and B. Choi, "Application deployment using Microservice and Docker containers: Framework and optimization", Journal of Network and Computer Applications, vol. 119, pp. 97-109, 2018. doi: 10.1016/j.jnca.2018.07.003.