Journal of Theoretical and Applied Information Technology
30th September 2022. Vol.100. No 18
© 2022 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

UTILIZATION DOCKER SWARMS IN A CONTAINER TECHNOLOGY SYSTEM TO PROBLEM SOLVING LOAD BALANCE

[1]NURUL KHAIRINA, [1]SUSILAWATI, [2]RAHMAD SYAH*

[1]Faculty of Engineering, Universitas Medan Area. Medan. Indonesia
[2]Excellent Centre of Innovation and New Science, Universitas Medan Area, Jl. Kolam No. 1/Gedung PBSI, Medan, Sumatera Utara, Medan. Indonesia
e-mail: rahmadsyah@uma.ac.id*

ABSTRACT

Container technology is a type of virtualization technology that is used at the operating system level. This is more in line with the need for long-term data storage for day-to-day operations. Data mining is becoming increasingly important for everyone in order to gather relevant information. Large-scale data processing slows down several operations, including data collection, processing, organization, modeling, analysis, and storage. The most significant difference for most users is the use of a high-quality data-collection system and its distribution, as well as the use of a complex data-analysis system. Docker Swarm is used for the development of various types of distributed systems, which can aid in the resolution of all issues as well as the optimization of results. Machine Learning was used to perform horizontal.

Keywords: *Optimasi*, *Docker Swarm, Machine Learning, Raspberry pi, Support Vector Regression.*

1. INTRODUCTION

The server is required to process a service when the quantity of services requested by a client exceeds the server's maximum load capacity. This might enhance server performance and decrease network latency. If the system receives more requests than it can handle every hour, day, week, and month, the server's performance declines, and issues develop[1][2].

The cluster computer system was created using the boot method at the time of implementation[3][4]. Balancing is a technique for balancing traffic loads across multiple server clusters so that traffic can run optimally and at maximum speed[5]. To increase distributed system scalability, load balancing must be implemented on the web server cluster as a means of managing a high volume of requests or the server load brought on by numerous requests. The goal of optimization is to produce the best results within predetermined circumstances [6][7][8]. All of these actions ultimately aim to increase intended benefit while minimizing effort. Optimization can be defined as the process of figuring out the circumstances that give the function's minimal or greatest value since the amount of work necessary or the intended benefit can be stated as a function of the choice variable[9].

When utilizing a machine learning strategy for regression, Support Vector Regression (SVR) is an SVM technique. [10][11][12]. SVR is a regression function that satisfies all data inputs with an error and uses the function f(x) as a dividing line(). This will be carried out in an effort to optimize. The counterbalance is anticipated to accept exceptionally large or congested demand[13][14].

The amount of data generated when providing services to applications increases along with the scale of IoT, and there are strong expectations for risk resistance and delay sensitivity, which causes a significant performance bottleneck in the technological cloud. The main purpose of edge computing is to stream data and sink services to the edge. This approach would not only ease load on cloud servers but also lessen delay brought on by remote traffic and congestion, which was previously borne by the edge side of the cloud. It has the ability to implement the service locally, enhancing risk resistance, particularly in cases of unstable network conditions.

Container-based machine learning systems can support container lifecycle management by foreseeing real-time anomalies and failures through monitoring management tools and visualizations [15].

2. RELATED WORK

2.1 Docker Container

Docker is the most widely used container platform. Because they virtualize hardware, Docker containers are significantly lighter and quicker. Docker is often 26 times quicker than virtual machines and can run on anything from small devices to huge servers (VMs) [16][17][18].

The most widely used open-source, lightweight virtual machine-like solution for application-oriented container virtualization is called docker. Each independently running container is isolated by docker. Less overhead is produced when a straightforward docker design is used. [19]. The hosts that make up Docker are numerous. Docker running in cluster swarm mode with a manager (to manage membership) a worker, who manages the swarm service, and one more[20]. The ability to change the configuration of the service, including the network and volumes connected to it, without having to manually restart it is one of the key advantages of the swarm service. [21].

2.2 Method

The Support Vector Regression (SVR) algorithm, one of the machine learning algorithms, can resolve the overfitting issue and generate accurate predictions [22][23]. Overfitting is the tendency of data to exhibit near-perfect prediction accuracy during training or training phase. The optimal dividing line or hyperplane is what the SVR algorithm seeks to identify. by measuring the magnin in the hyperplane and identifying the ideal hyperplane. The margin is the separation between the nearest data point and the hyperplane. The data closest to the edge is the support vector [24].

3 RESEARCH AND METHODOLOGY

Machine learning model training tasks are carried out on the cloud because cloud servers provide an abundance of GPU/CPU, RAM, and other resources. The model is subsequently sent across the Internet to the edge. Models are run as services on edge computing platforms after being containerized by edge. Users can communicate with the computing platform and make service requests [25][26].

The core unit that delivers a service is a pod, which can be made up of a group of tightly coupled containers. Through a network of bridges, these internal containers effectively communicate with one another and share storage volumes to house resources. Pod provides machine learning prediction services[27][28].

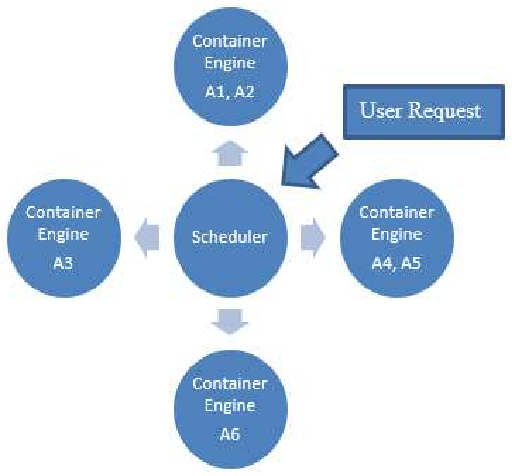


Figure 2 User Request

The edge computing platform builds a client container when it gets an image from a client. Containers exchange messages with each other. As shown in Figure 1, the expected outcome is then communicated to the user, and the client container is then destroyed.

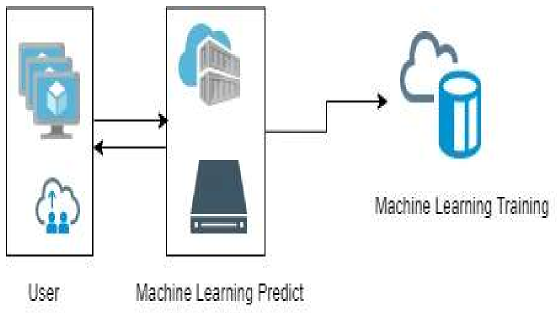


Figure 1 Container Machine Learning

3.1. Machine Learning

A container is made up of several image layers stacked on top of one another to form a single container. Heavy image layers are shared by numerous containers and are read-only. dependable system. Only the exclusive container layer is removed when a container is deleted; the shared image layer is left intact.

to obtain the highest edge-side machine learning deployment performance and application coexistence. We create the edge computing platform using lightweight Docker

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

container technology and the Kubernetes container choreography system, and we properly account for heterogeneity and resource constraints during the deployment process of the machine learning model. Using Docker technology, we package the machine learning model and related resources into containers. On the other hand, Kubernetes offers flexible resource scheduling for computation, storage, and other resources in addition to real-time monitoring, operation, and maintenance of containers [29][30]. As a result of the experiment, we learned how to install machine learning services on the edge side. The machine learning container workflow is shown in Figure 2.

3.2. Kubernetes

An open-source container cluster management system called Kubernetes can automate the deployment, growth, and upkeep of container clusters [31].

A master component and component nodes make up Kubernetes. Both host-installed master components and host-installed node components are referred to as Master and Node.js, respectively. work managing the platform's functionality Api-server offers a user interface for managing activities across many resources, whereas Kubectl is a tool for managing clusters. The cluster's default storage system is Kubernetes. While Controller Manager is in charge of running numerous controllers and assuring real-time service association and pod information, Scheduler is responsible for managing scheduled resources [32][33].

The node component collaborates with Master to handle Node.js while running on a managed host. Node components include Kubelet and Kube proxy. Running and maintaining the pods on the Node is the responsibility of the Kubelet. The fundamental component of a service is a pod. The Kub-proxy is in charge of facilitating communication within the cluster of pods. Figure 3 illustrates a cluster with every pod connected to the same Master.

4 EXPERIMENT

4.1 RaspberryPi

One laptop serves as the user, one laptop serves as the host node, one laptop serves as the master host, and one mobile wifi device serves as a communication relay.

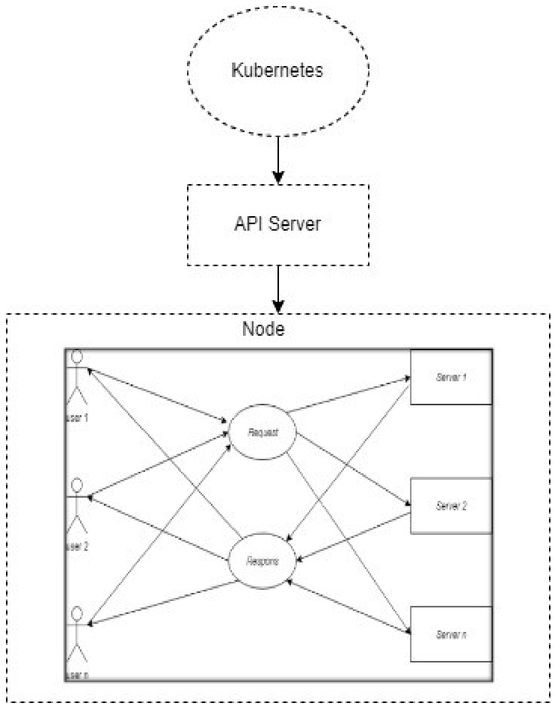


Figure 3 Kubernetes

The following describes the current system's operating principle [34][35]:

1. A request is made by the user.
2. The user agrees to the request and navigates to the RaspberryPi Server section.
3. The Raspberry Pi server continues to accept user requests and uses Haproxy as a load balancer for this probe.
4. The Raspberry Pi cluster node is added to the Haproxy load balancer.
5. RaspberryPi Node Cluster indicates whether or not the Raspberry Pi Node Cluster still has service availability or can meet user demands.
6. If node 1 is ready, returns the result; otherwise, notifies haproxy as the load balancer.
7. Response Outcomes This section differentiates between successful and error response results and returns them to the user.
8. Requests per second is a scalability metric for the system's throughput.
9. Requests per second tests are carried out by running a series of services per second from the client.
10. This service is sent to the web server to use the load balancer to determine the web server's performance. This is intended to

test how many requests a web server running on a Docker Swarm container cluster can handle.

11. $Requests\ per\ second = hits\ per\ second\ (hits/s)$
12. Loss Requests is a request sent from a client to an overloaded web server because the web server cannot handle exceeding the web server's maximum capacity and the web server cannot accept the upgrade.
    $\boldsymbol{RL} = \boldsymbol{u} * \mathrm{e}$

4.2 How to Apply Support Vector Regression (SVR)[36][37]

1. Data sharing for training and testing.
2. Determine the kernel function and loss function that will be used for forecasting.
3. Determine the C and kernel parameters' values.
4. Determine the beta and bias values.
5. Predicting the results of the tests.
6. Using the best kernel functions, loss functions, and parameters to forecast the future

5 RESULT AND DISCUSSION

Following the modification of the formulation to make the Support Vector Regression non-linear, the following equation displays the best approximation surface: [38][39][40]

$$F(x) = \sum_{i=1}^{1}(a_{j-}^{*} a_j)\,(K(x_i, x) + \lambda^2) \quad (2)$$

This method is helpful for finding the best dividing line for the Support Vector Regression function in each step of the calculation [41]. The sequential learning process includes the following steps:

Initialization $\alpha_i = 0, \alpha_i^* = 0,$ then calculate the matrix R_{ij}

$$R_{ij} = (K(x_{i,}x) + \lambda^2 \text{ for i, j=1, ... n} \quad (3)$$

Information:

Rij	= matriks hessian
K	= kernel function
$x_{i,}$	= data ke – i
$x_{j,}$	= data ke – j
λ	= Scalar Variable

For each training data i – 1 until n calculated:

a. $E_i = y_i - \sum_{j=1}^{1}(a_j^* - a_j)\, R_{ij}$
b. $\delta\alpha_i^* = min\{max[y(E_i - \varepsilon), -\alpha_i], C - \propto_{i\}}$
c. $\alpha_i^* = \delta\alpha_i^*$

In the regression there are residuals, for example residuals (r) defined by
subtracting the scalar y output from the estimate f(x) by r = y – f(x) :

$$E(r) = \begin{cases} 0 & \text{for } |r| \le \varepsilon \\ |r| - \varepsilon & \text{For others} \end{cases} \quad (5)$$

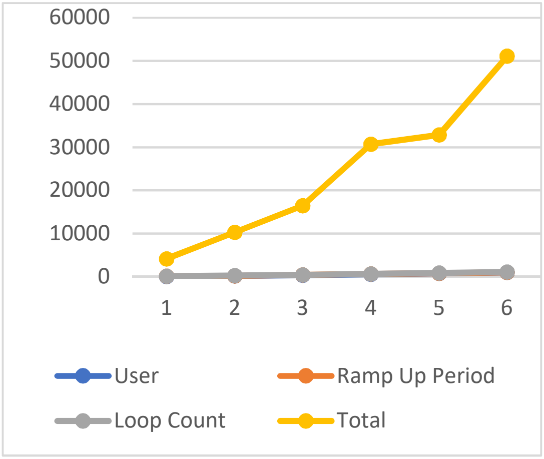
Use kernel functions while using Support Vector Regression techniques to solve non-linear issues. Replace the inner product (X_i and X_j) with kernel functions for solving a linear problem in a high-dimensional space. With this kernel function, higher-dimensional feature spaces can be related to without the need for explicit mapping calculations [42][43]. The Support Vector Algorithm Work regression is determined by the type of kernel function to be employed and the kernel parameter parameters. (1)

The effectiveness of the system is assessed under varied loads. By increasing the number of containers from one to four, machine learning applications and containers are measured. These measurements for the scenarios in Table 1 comprise the following: memory usage for the application container, application execution time, memory usage during execution, and network I/O blocks [44][45][46][47].

Table 1. User Data

User	*Ramp Up Period*	*Loop Count*	*Total*
500	30	40	20000
500	40	50	25000
1000	30	40	40000
1000	40	50	50000
1500	30	40	60000
1500	40	50	75000
2000	30	40	80000
2000	40	50	100000
2500	30	40	100000
2500	40	50	125000
3000	30	40	120000
3000	40	50	150000
5000	30	50	250000
8000	30	50	400000
10000	30	50	500000
20000	40	50	1000000

When Docker Containers are employed, ideal performance increases from a minimum of 5 percent to a maximum of 9 percent, while CPU overhead climbs from a minimum of 130 percent to a high of 185 percent when they are not. Figure 4 illustrates how the Docker Container can manage the load while lowering CPU use, enabling the device to function optimally without suffering any harm.



When Docker Container is used, the execution

Figure 5 User Time Periode

time for a single container is 40 seconds, which is not significantly different from when Docker Container is not used. When Docker Container is not used, the execution time for a single container increase from 55 seconds to 135 seconds. The exceptional outcomes are measured by a considerable difference from when Docker

Containers are not employed when the number of containers is increased to four. It has been shown that executing programs in a lightweight virtualization environment based on Docker Containers saves time and money. Table 2 and Figure 5 provide examples of this.

Table 2 User Time Period

User	*Ramp Up Period*	*Loop Count*	*Total*
100	30	40	4000
200	40	50	10000
400	30	40	16000
600	40	50	30000
800	30	40	32000
1000	40	50	50000

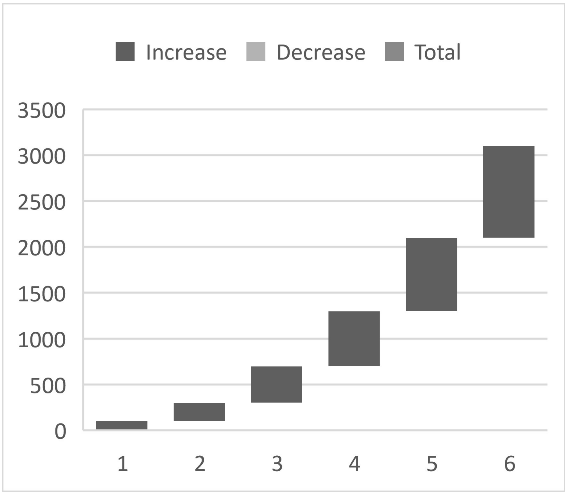


Figure 4 User Load

6 CONCLUSION

This paper optimizes the development and deployment of edge computing platforms by combining machine learning, Docker, and Kubernetes technologies. As a consequence of the experiment, we learned that the computing platform is capable of deploying machine learning services. Due to improved network conditions between the cloud and customers, edge computing platforms can offer users more reliable machine learning services. As a result, Docker containers can be made horizontally elastic in response to varying workloads for real-time applications. The findings demonstrate that machine learning may continue to perform well even when the workload varies.

ACKNOWLEDGMENT

This research was completed in the lab. Faculty of Engineering, University of Medan Area and thanks to LP2M Medan Area University for providing support and assistance.

REFERENCES

[1] K. Ye, Y. Kou, C. Lu, Y. Wang and C. -Z. Xu, "Modeling Application Performance in Docker Containers Using Machine Learning Techniques," 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), 2018, pp. 1-6, doi: 10.1109/PADSW.2018.8644581

[2] Buchs, D., Klikovits, S., Linard, A., Mencattini, R., & Racordon, D. (2018, June). A model checker collection for the model checking contest using docker and machine learning. In *International Conference on Applications and Theory of Petri Nets and Concurrency* (pp. 385-395). Springer, Cham.

[3] Yadav, M. P., & Yadav, D. K. (2021). Maintaining container sustainability through machine learning. *Cluster Computing*, *24*(4), 3725-3750.

[4] Abdullah, M., Iqbal, W., Bukhari, F., & Erradi, A. (2020). Diminishing returns and deep learning for adaptive CPU resource allocation of containers. *IEEE Transactions on Network and Service Management*, *17*(4), 2052-2063.

[5] Huang, Y., cai, K., Zong, R., & Mao, Y. (2019, March). Design and implementation of an edge computing platform architecture using docker and kubernetes for machine learning. In *Proceedings of the 3rd International Conference on High Performance Compilation, Computing and Communications* (pp. 29-32).

[6] Ramos, F., Viegas, E., Santin, A., Horchulhack, P., dos Santos, R. R., & Espindola, A. (2021, June). A machine learning model for detection of docker-based APP overbooking on kubernetes. In *ICC 2021-IEEE International Conference on Communications* (pp. 1-6). IEEE.

[7] Chiang, R. C. (2020). Contention-aware container placement strategy for docker swarm with machine learning based clustering algorithms. *Cluster Computing*, 1-11.

[8] Kim, B. S., Lee, S. H., Lee, Y. R., Park, Y. H., & Jeong, J. (2022). Design and Implementation of Cloud Docker Application Architecture Based on Machine Learning in Container Management for Smart Manufacturing. *Applied Sciences*, *12*(13), 6737.

[9] Xu, P., Shi, S., & Chu, X. (2017, August). Performance evaluation of deep learning tools in docker containers. In *2017 3rd International Conference on Big Data Computing and Communications (BIGCOM)* (pp. 395-403). IEEE.

[10] Awad, M., & Khanna, R. (2015). Support vector regression. In *Efficient learning machines* (pp. 67-80). Apress, Berkeley, CA.

[11] Ceperic, E., Ceperic, V., & Baric, A. (2013). A strategy for short-term load forecasting by support vector regression machines. *IEEE Transactions on Power Systems*, *28*(4), 4356-4364.

[12] Xu, S., An, X., Qiao, X., Zhu, L., & Li, L. (2013). Multi-output least-squares support vector regression machines. *Pattern Recognition Letters*, *34*(9), 1078-1084.

[13] Li, E., Yang, F., Ren, M., Zhang, X., Zhou, J., & Khandelwal, M. (2021). Prediction of blasting mean fragment size using support vector regression combined with five optimization algorithms. *Journal of Rock Mechanics and Geotechnical Engineering*, *13*(6), 1380-1397.

[14] Jiang, H., Zhang, Y., Muljadi, E., Zhang, J. J., & Gao, D. W. (2016). A short-term and high-resolution distribution system load forecasting approach using support vector regression with hybrid parameters optimization. *IEEE Transactions on Smart Grid*, *9*(4), 3341-3350.

[15] Kumar, K., & Kurhekar, M. (2017, December). Sentimentalizer: Docker container utility over Cloud. In *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)* (pp. 1-6). IEEE.

[16] Kaewkasi, C., & Chuenmuneewong, K. (2017, February). Improvement of container scheduling for docker using ant colony optimization. In *2017 9th international conference on knowledge and smart technology (KST)* (pp. 254-259). IEEE.

[17] Rad, B. B., Bhatti, H. J., & Ahmadi, M. (2017). An introduction to docker and analysis of its performance. *International Journal of Computer Science and Network Security (IJCSNS)*, *17*(3), 228.

[18] Naik, N. (2016, October). Building a virtual system of systems using docker swarm in multiple clouds. In *2016 IEEE International Symposium on Systems Engineering (ISSE)* (pp. 1-3). IEEE.

[19] Al-Rakhami, M., Alsahli, M., Hassan, M. M., Alamri, A., Guerrieri, A., & Fortino, G. (2018, August). Cost efficient edge intelligence framework using docker containers. In *2018 IEEE 16th Intl Conf on*

Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech) (pp. 800-807). IEEE.

[20] Brady, K., Moon, S., Nguyen, T., & Coffman, J. (2020, January). Docker container security in cloud computing. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 0975-0980). IEEE.

[21] Santos, E. A., McLean, C., Solinas, C., & Hindle, A. (2018). How does Docker affect energy consumption? Evaluating workloads in and out of Docker containers. *Journal of Systems and Software*, *146*, 14-25.

[22] Lee, Y., Oh, S., & Choi, D. H. (2008). Design optimization using support vector regression. *Journal of Mechanical Science and Technology*, *22*(2), 213-220.

[23] Li, E., Yang, F., Ren, M., Zhang, X., Zhou, J., & Khandelwal, M. (2021). Prediction of blasting mean fragment size using support vector regression combined with five optimization algorithms. *Journal of Rock Mechanics and Geotechnical Engineering*, *13*(6), 1380-1397.

[24] Zhang, F., & O'Donnell, L. J. (2020). Support vector regression. In *Machine Learning* (pp. 123-140). Academic Press.

[25] Pahl, C., & Lee, B. (2015, August). Containers and clusters for edge cloud architectures--a technology review. In *2015 3rd international conference on future internet of things and cloud* (pp. 379-386). IEEE.

[26] Park, M., Bhardwaj, K., & Gavrilovska, A. (2020). Toward lighter containers for the edge. In *3rd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 20)*.

[27] Ribeiro, M., Grolinger, K., & Capretz, M. A. (2015, December). Mlaas: Machine learning as a service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)* (pp. 896-902). IEEE.

[28] Doan, T. V., Nguyen, G. T., Salah, H., Pandi, S., Jarschel, M., Pries, R., & Fitzek, F. H. (2019, September). Containers vs virtual machines: Choosing the right virtualization technology for mobile edge cloud. In *2019 IEEE 2nd 5G World Forum (5GWF)* (pp. 46-52). IEEE.

[29] Bernstein, D. (2014). Containers and cloud: From lxc to docker to kubernetes. *IEEE cloud computing*, *1*(3), 81-84.

[30] Marathe, N., Gandhi, A., & Shah, J. M. (2019, April). Docker swarm and kubernetes in cloud computing environment. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)* (pp. 179-184). IEEE.

[31] Mao, Y., Fu, Y., Gu, S., Vhaduri, S., Cheng, L., & Liu, Q. (2020). Resource management schemes for cloud-native platforms with computing containers of docker and kubernetes. *arXiv preprint arXiv:2010.10350*.

[32] Oliveira, C., Lung, L. C., Netto, H., & Rech, L. (2016, September). Evaluating raft in docker on kubernetes. In *International Conference on Systems Science* (pp. 123-130). Springer, Cham.

[33] Luksa, M. (2017). *Kubernetes in action*. Simon and Schuster.

[34] Bellavista, P., & Zanni, A. (2017, January). Feasibility of fog computing deployment based on docker containerization over raspberrypi. In *Proceedings of the 18th international conference on distributed computing and networking* (pp. 1-10).

[35] Pahl, C., Helmer, S., Miori, L., Sanin, J., & Lee, B. (2016, August). A container-based edge cloud paas architecture based on raspberry pi clusters. In *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)* (pp. 117-124). IEEE.

[36] Lapins, M., Arvidsson, S., Lampa, S., Berg, A., Schaal, W., Alvarsson, J., & Spjuth, O. (2018). A confidence predictor for logD using conformal regression and a support-vector machine. *Journal of cheminformatics*, *10*(1), 1-10.

[37] Cavalcanti, M., Inacio, P., & Freire, M. (2021, August). Performance Evaluation of Container-Level Anomaly-Based Intrusion Detection Systems for Multi-Tenant Applications Using Machine Learning Algorithms. In *The 16th International Conference on Availability, Reliability and Security* (pp. 1-9).

[38] Verkerken, M., D'hooge, L., Wauters, T., Volckaert, B., & De Turck, F. (2020, October). Unsupervised machine learning techniques for network intrusion detection on modern data. In *2020 4th Cyber Security in Networking Conference (CSNet)* (pp. 1-8). IEEE.

[39] Gore, R., Banerjea, S., Tyagi, N., Saurav, S., Acharya, D., & Verma, V. (2020, December). An Efficient Edge Analytical Model on Docker Containers for Automated Monitoring of Public Restrooms in India. In *2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)* (pp. 1-6). IEEE.

[40] Han, Q., Yang, S., Ren, X., Zhao, C., Zhang, J., & Yang, X. (2020). OL4EL: Online Learning for Edge-Cloud Collaborative Learning on Heterogeneous Edges with Resource Constraints. *IEEE Communications Magazine*, *58*(5), 49-55.

[41] Anand, P., Rastogi, R., & Chandra, S. (2020). A class of new support vector regression models. *Applied Soft Computing*, *94*, 106446.

[42] Bhavsar, H., & Panchal, M. H. (2012). A review on support vector machine for data classification. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, *1*(10), 185-189.

[43] Kavitha, S., Varuna, S., & Ramya, R. (2016, November). A comparative analysis on linear regression and support vector regression. In *2016 online international conference on green engineering and technologies (IC-GET)* (pp. 1-5). IEEE.

[44] Tharwat, A. (2019). Parameter investigation of support vector machine classifier with kernel functions. *Knowledge and Information Systems*, *61*(3), 1269-1302.

[45] Toka, L., Dobreff, G., Fodor, B., & Sonkoly, B. (2021). Machine learning-based scaling management for kubernetes edge clusters. *IEEE Transactions on Network and Service Management*, *18*(1), 958-972.

[46] Dartois, J. E., Boukhobza, J., Knefati, A., & Barais, O. (2019). Investigating machine learning algorithms for modeling ssd i/o performance for container-based virtualization. *IEEE transactions on cloud computing*, *9*(3), 1103-1116.

[47] Marquez, J. D., & Castillo, M. (2021, February). Performance comparison: Virtual machines and containers running artificial intelligence applications. In *International Conference on Information Technology & Systems* (pp. 199-209). Springer, Cham.marques