Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 4, October 2024
DOI : https://doi.org/10.33395/sinkron.v8i4.14052
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Implementation Docker and Kubernetes Scaling Using Horizontal Scaler Method for Wordpress Services

Suryayusra[1)], Nova Destarina[2)*], Edi Surya Negara[3)], Edi Supratman[4)], Maria Ulfa[5)]
[1,2,3,4,5)] informatics engineering, Universitas Bina Darma Palembang, Indonesia
[1)]suryayusra@binadarma.ac.id , [2)]novadestarina092@gmail.com , [3)]e.s.negara@binadarma.ac.id ,
[4)]edi_supratman@binadarma.ac.id , [5)]mu@binadarma.ac.id

Submitted :Sept 5, 2024 | **Accepted** : Sept 13, 2024 | **Published** : Oct 3, 2024

Abstract: Container is a technology that has recently been widely used because of the additional features that are very easy and convenient to use, especially for web hosting service developers, with Container making it easier for system admins to manage applications including building, processing and running applications on Container. With Container the process of creating and using the system will be easier but along with too many user requests so that the service does not run optimally. Therefore, the Container must have good scalability and performance. Scalability is needed for systems that can adjust to the needs of user demand and performance is needed to maintain the quality of services provided. This research aims to implement scaling using Docker and Kubernetes in terms of scalability and performance. The parameters of comparison between Docker and Kubernetes are for scalability, scaling up and scaling down time and for performance. The method in this research uses the Action Research methodology, which is a research model that is simultaneously practiced and theorized. With the initial steps of problem identification, action planning, action implementation, observation and evaluation. Based on the results that have been obtained, Docker consumes more CPU & Memory Usage Resources, namely at 500 Users Kubernetes consumes Resources with an average of 94.47%-4.70% while in Kubernetes 89.11%-4.50 because in Kubernetes itself has a complex system, especially special component components such as APIs, Metrics Server, Kubernetes manager to run the Container. While in Docker only has Docker Manager and Docker Compose components.

Keywords: Container, Docker, Kubernetes, Load Testing, Scaling.

INTRODUCTION

Docker and Kubernetes have emerged as solutions to address infrastructure scalability. Docker enables the packaging of applications into containers. Docker emerged as a response to problems arising in the software development and deployment process. While Docker addressed the problem of packaging and running applications, Kubernetes was born to address the challenges of Container management at scale. By using Containers, the need arises to efficiently manage 2 concurrently running Containers (Putri et al., 2021). Containers are isolated environments that run on top of the same operating system but do not share address space with each other (Silva et al., 2019).

Kubernetes provides a solution to this problem by providing an open-source platform for Container application scalability. With Kubernetes, users can manage their applications in a secure, scalable way. Kubernetes provides a robust platform for scaling Containers (Rina & Ridha, 2021). One of the most widely used Container-based virtualization today is Docker. Docker is one of the useful tools for running containers. However, managing containers to create many services and handle many users is a big task for Docker (Sugiyatno & M, 2023). In Kubernetes, a Container is a Pod the smallest deployment unit that can contain one or more Containers. The Pod provides an execution environment that includes a shared network namespace, File system, and storage volumes (Zulfikar, 2022).

The purpose of this research is to compare the performance of the scaling process in Docker and Kubernetes. In Kubernetes using Horizontal Scaler method while in Docker using the same Horizontal Scaler method. The comparison parameters between Docker and Kubernetes are Load Testing for scalability, Scaling up and Scaling down time for performance.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 4, October 2024
DOI : https://doi.org/10.33395/sinkron.v8i4.14052
e-ISSN : 2541-2019
p-ISSN : 2541-044X

LITERATURE REVIEW

Scaling is the main problem in this case. Inadequate resources that are unable to cope with strong changes in workload over time, in which case the application experiences low performance or too much user demand, in which case the utilization of resources allocated to the Container is low. Therefore, scaling over time is required (Subhi et al., 2021). Load Testing is a performance testing technique where system response is measured under various conditions and loads. This test helps determine how software behaves when multiple users access the software simultaneously. Scalability is the ability of a Container to handle an increasing number of users. Scalability can be achieved through existing configurations or by adding Containers, thus allowing new users access to the application without significant delays (Yedutun et al., 2019). Performance is the result of work that can be achieved. In the context of information technology, it refers to how well a system or application operates in terms of responsiveness, speed, efficient use of resources, and ability to handle a given workload (Aruan & Rahayu, 2023).

Load Testing is a critical approach in measuring and evaluating the performance of an application. In the context of Load Testing performance analysis. especially when faced with a high workload (Andrianto & Suyatno, 2024). The Central Processing Unit (CPU), often referred to as the processor or the brain of the computer, has the main function of processing and managing all the calculations and commands that allow the computer to operate. Since the CPU generates heat during its work process, it is usually equipped with fans and heat sinks to keep the temperature down (Rivki et al., 2019). Web hosting is a service provided by a web hosting provider that allows users or others to store and publish websites or web applications on the internet (Yosli, 2021).

The previous research related to comparing the performance of the Scaling process on Kubernetes and Docker Swarm. In Kubernetes using Horizontal Pod auto scaler method while in Docker Swarm using Scaling method. The comparison parameters between Kubernetes and Docker Swarm are Load Testing for scalability, Scaling up and Scaling down time for performance (Firdaus et al., 2020). Further previous research on Comparative Analysis of Docker Swarm Web Server Performance with Kubernetes Cluster, Before testing was carried out on two Cluster servers with different orchestration tools. Both Cluster servers are built on Google Cloud Platform with the same server specifications. Then docker swarm manager is connected to two swarm nodes, Kubernetes Cluster control plane is connected to two nodes. After that, install Nginx as a web server on the swarm manager and control plane, and replicated to other nodes. Both Cluster servers are then load tested using the Apache Jmeter application on a laptop that has been prepared. The measurement data results are organized based on the parameters used in this study (Prasetyo & Salimin, 2021). The next previous research with the title Implementation of Optimistic Concurrency Control on E-Commerce Application Systems Based on MicroServices Architecture Using Kubernetes, Testing is implemented with the Load Testing method. In Load Testing, several parameters need to be considered. The results obtained are in the form of a comparison of the amount of data. Experiments were carried out with several trials with different parameters (Ammar Dwi Anwari et al., 2021).

METHOD

The method in this study uses the Action Research methodology, which is a research model that is at the same time practicing and theorizing, or combining theory and implementing it in practice (Hasan, 2019).

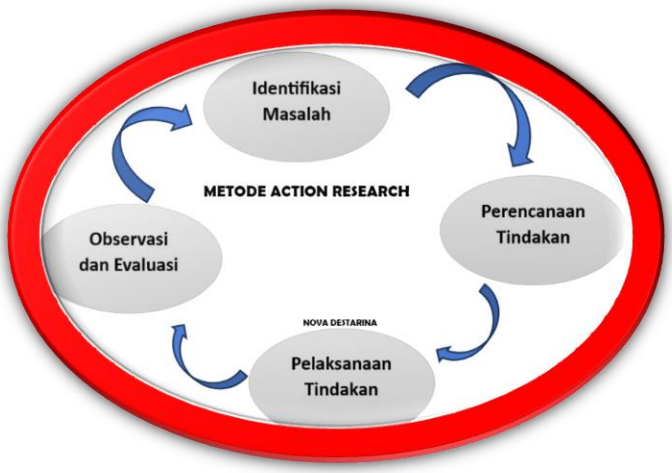

Figure 1. Action Research Methodology

In summary, the stages in Action Research consist of a cycle of problem identification is to identify problems or challenges related to the scalability of wordpres web hosting services using Docker and Kubernete, action

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 4, October 2024
DOI : https://doi.org/10.33395/sinkron.v8i4.14052
e-ISSN : 2541-2019
p-ISSN : 2541-044X

planning is to plan steps to implement Scaling on Docker and Kubernetes, action implementation is The planned steps are then implemented in a production environment using Docker and Kubernetes, and observation and evaluation of actions Observing system performance after the implementation of Scaling steps. collecting data related to performance and Scaling on the system for evaluation. This entire cycle is conducted based on the context and objectives of the action research being conducted.

Some tools and hardware that will be used in the research are the specifications of the hardware used, among others :

Table 1. Hardware Specifications

Komponen	Spesifikasi
Nama Perangkat	Laptop Acer Aspire Lite 14
Processor	12th Gen Intel(R) Core(TM) i3-1215U 1.20 GHz
RAM	8 GB memory
HDD	512 GB HDD

The software to be used includes :

1. Docker Desktop as Software.
2. Docker as Container controller.
3. Kubernetes as Container controller.
4. Metric Server as monitoring in kubernetes.
5. Apache Jmeter as stressing tool for Wordpress web hosting.

RESULT

This section displays the results of the analysis. As explained earlier, the parameters chosen to be measured are Load Testing for scalability, Scaling up and Scaling down time for performance. The purpose of Load Testing is to see the scalability of each Container. By looking at the load on CPU & Memory usage with the specified User. CPU & Memory usage shows what percentage is needed in the process carried out in the experiment.

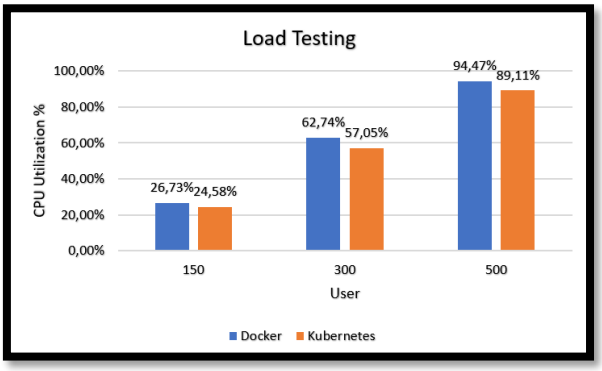

Figure 2. CPU Usage Load Testing Diagram

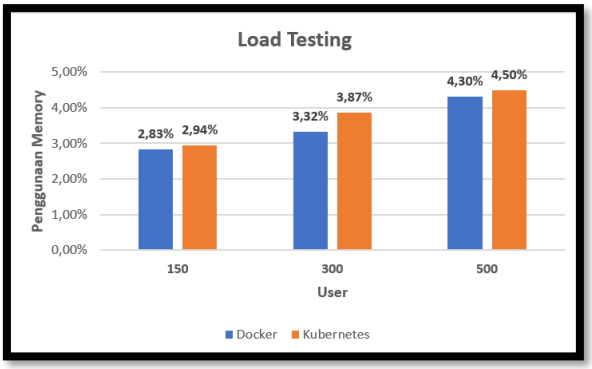

Figure 3. Memory Usage Load Testing Diagram

Seen from the picture the average results of Load Testing when generating 150 users consumes CPU & Memory usage with an average of 26.73%-2.83% while on Kubernetes 24.58%-2.94%. Furthermore, when generating 300 Users consumes CPU usage with an average of 62.74%-3.32% while in Kubernetes 57.05%-3.87%.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 8, Number 4, October 2024
DOI : https://doi.org/10.33395/sinkron.v8i4.14052
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Furthermore, when 500 users take up CPU usage with an average of 94.47%-4.30% while in Kubernetes 89.11%-4.50. Because in Kubernetes itself has a complex system, especially special components such as Metric Server, Kubernetes manager, and kubectl to run the Container. While in Docker only has Docker Manager and Docker Compose components.

The purpose of scaling up is to see the performance of each Container. By looking at the Container Creating time and the Container Load Testing time. The purpose of Scaling down is to see the performance of each Container. By measuring CPU usage when the service is not in use.

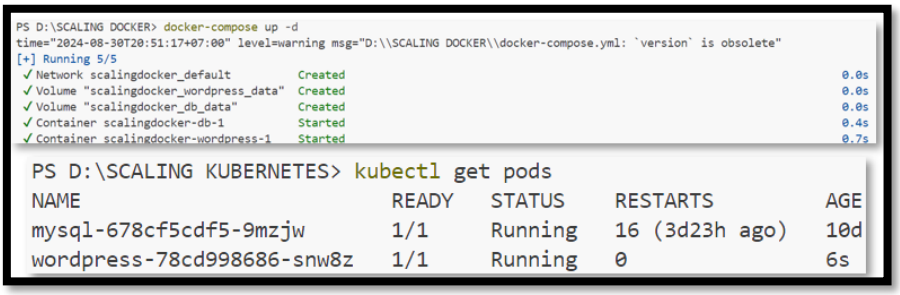

Figure 4. Docker built in 0.7s & Kubernetes built in 6s

Creating a Docker Container takes 0.7s (0.7 seconds) while creating a new Kubernetes Pod (Container) on a node takes 6s (6 seconds).

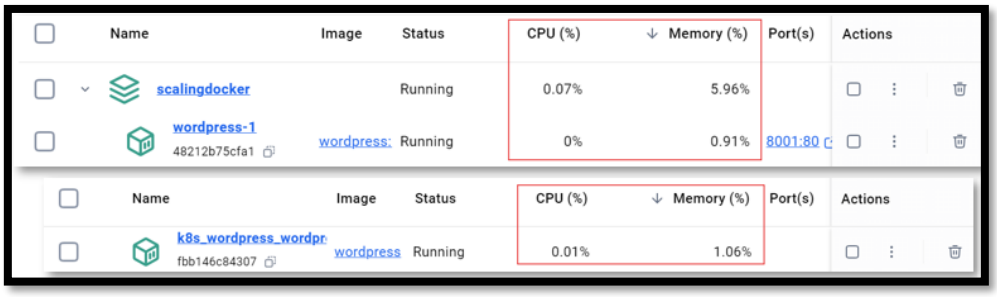

Figure 5. CPU and Memory utilization in Docker and Kubernetes when services are no longer in use

When the Request Generator finishes generating the User. Pod (Container) or service will return to 0%. and CPU & Memory Usage of each node will return to normal.

DISCUSSIONS

The results of the analysis of all tests carried out in this final project research research can draw the following conclusions :

1. Based on the results that have been obtained, Docker consumes more CPU & Memory Usage Resources, namely at 500 Users Kubernetes consumes Resources with an average of 94.47%-4.70% while in Kubernetes 89.11%-4.50 because in Kubernetes itself has a complex system, especially special component components such as API, Metrics Server, Kubernetes manager to run the Container. While in Docker only has Docker Manager and Docker Compose components.
2. Based on the results that have been obtained Scaling up in Kubernetes is superior because of scaling but in terms of Restart Docker is faster, namely with an average time of 2.1 seconds while Kubernetes 27 seconds. For Scaling down Docker is favored in terms of removing the Container. Because the removal is done with an average time of 1.4 seconds. Although Kubernetes looks longer in deleting but in Kubernetes there is a Container deletion with an average time of 11 seconds. Based on the data above, Kubernetes is more suitable for companies that deal with unpredictable users such as e-commerce, namely Lazada, Shopee, Tokopedia, Olx, Amazon. While Docker is more suitable for companies that deal with predictable users such as internal companies with a certain number of employees.

CONCLUSION

This research certainly still has shortcomings and can be explored further. Therefore, the author provides several suggestions, namely :

1. The scaling process can be improved by using other methods to utilize CPU and memory usage metrics, so that the scaling process can be more effective.

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

2. Comparing Containers by combining the Scaling process with other processes so as to get which Container is the best

REFERENCES

Ammar Dwi Anwari, Rizky Januar Akbar, & Royyana Muslim Ijtihadie. (2021). Implementasi Optimistic Concurrency Control Pada Sistem Aplikasi E-Commerce Berdasarkan Arsitektur Microservices Menggunakan Kubernetes. *Jurnal Teknik Its* , *10*(2).

Andrianto, L. D., & Suyatno, D. F. (2024). Analisis Performa Load Testing Antara Mysql Dan Nosql Mongodb Pada RestAPI Nodejs Menggunakan Postman. *Journal of Emerging Information System and Business Intelligence (JEISBI)*, *5*(1), 18–26. https://ejournal.unesa.ac.id/index.php/JEISBI/article/view/58157%0Ahttps://ejournal.unesa.ac.id

Aruan, M. C., & Rahayu, W. (2023). Analisis Performa Algoritma Kompresi Data dalam Penyimpanan dan Transfer Data. *LANCAH: Jurnal Inovasi Dan Tren*, *1*(2), 228–232.

Firdaus, B. A., Suryani, V., & Karimah, S. A. (2020). Analisis Performansi Proses Scaling pada Kubernetes dan Docker Swarm Menggunakan Metode Horizontal Scaler. *E-Proceeding of Engineering*, *7*(2), 7793–7808.

Hasan. (2019). Action Research : Desain Penelitian Integratif untuk Mengatasi Permasalahan Masyarakat. *AKSES: Jurnal Ekonomi Dan Bisnis*, *4*(8), 12. https://publikasiilmiah.unwahas.ac.id/index.php/AKSES/article/view/523

Prasetyo, S. E., & Salimin, Y. (2021). Analisis Perbandingan Performa Web Server Docker Swarm dengan Kubernetes Cluster. *CoMBInES - Conference on Management, Business, Innovation, Education and Social Sciences*, *1*(1), 825–833. https://journal.uib.ac.id/index.php/combines/article/view/4512

Putri, S. N., Arif, M., & Ridha, F. (2021). *Implementasi Clustered Container Dengan Docker Sarm*. 201–208.

Rina, N. K. S., & Ridha, M. A. F. (2021). The Implementasi Kubernetes Cluster Menggunakan KVM. *ABEC Indonesia*, 209–217. https://abecindonesia.org/proceeding/index.php/abec/article/view/151%0Ahttps://abecindonesia.org/proceeding/index.php/abec/article/download/151/149

Rivki, M., Bachtiar, A. M., Informatika, T., Teknik, F., & Indonesia, U. K. (2019). *No 主観的健康感を中心とした在宅高齢者における 健康関連指標に関する共分散構造分析Title*. *112*.

Silva, V. G. da, Kirikova, M., & Alksnis, G. (2019). Containers for Virtualization: An Overview. *Applied Computer Systems*, *23*(1), 21–27. https://doi.org/10.2478/acss-2018-0003

Subhi, R., Ruslianto, I., Ristian, U., Rekayasa, J., Komputer, S., Mipa, F., Tanjungpura, U., Prof, J., Hadari, H., & Pontianak, N. (2021). Implementasi Teknik Scaling Pada Sistem Manajemen Balancing Server Berbasis Website. *Jurnal Komputer Dan Aplikasi*, *09*(02), 316–326.

Sugiyatno, & M, I. (2023). Analisis Perbandingan Performasi Respon Waktu Web Server dan Failover Antara Kubernetes Dan Docker Swarm pada Container Orchestration. *Jurnal Informatika Komputer, Bisnis Dan Manajemen*, *21*(3), 43–53. https://doi.org/10.61805/fahma.v21i3.9

Yedutun, K., Noertjahyana, A., & Novianus Palit, H. (2019). Implementasi Container Kubernetes untuk Mendukung Scalability. *Jurnal Infra*, *7*(2), 1–5.

Yosli, R. (2021). Meningkatkan Kapasitas Hosting, Mengelola Content Management System Untuk Kenyamanan Memakai Website Berbayar. *JAVIT : Jurnal Vokasi Informatika*, 31–37. https://doi.org/10.24036/javit.v1i2.6

Zulfikar, A. (2022). *Penggunaan Docker Dan Kubernetes Pipeline Dalam Pengembangan Aplikasi Prediksi Cacat Perangkat Lunak Melalui Pendekatan Mlops*.

*name of corresponding author