2020 International Conference on Information Technology Systems and Innovation (ICITSI)
Bandung - Padang, October 19 - 23, 2020
ISBN: 978-1-7281-8196-7

High Availability Storage Server with Kubernetes

Ali Akbar Khatami
School of Electrical Engineering
Telkom University
Bandung, Indonesia
alivancisero@student.telkomuniversity.ac.id

Yudha Purwanto
School of Electrical Engineering
Telkom University
Bandung, Indonesia
omyudha@telkomuniversity.ac.id

Muhammad Faris Ruriawan
School of Electrical Engineering
Telkom University
Bandung, Indonesia
muhammadfaris@telkomuniversity.ac.id

***Abstract*— High availability server (HAS) is a concept in which the server resources continuously available through any interference of the virtual and physical server. HAS is widely used for various purposes such as online trading services, business and Big Data needs. To optimize storage, it needs a special way to minimize costs incurred and be able to optimize the existing server. This research has implemented a system of high availability server using Kubernetes and distributed storage. The system was capable of meeting the resource requests even though one instance was interrupted. And if an instance was down, the services stored in the cluster could still be accessed through other instances. Based on the reliability and availability testing, the system was capable of meeting the high availability criteria, by reaching an uptime rate of 100%.**

Keywords—High availability server, Kubernetes, big data, container, availability

I. INTRODUCTION

The problem of server availability on service is very important for users of a service. Server availability is very closely related to the quality offered by service, because when a server cannot serve a service, the service can be said to be less good (less quality). Because, when the server goes down, the service will automatically stop and cannot do anything.

A server that is always available when needed (High Availability Server) can help to improve the quality of service. The availability metric is very important in terms of quality of service and security [1]. For example, when we access a website with a server that is always available, the developer does not need to think about the availability of their website and can focus on developing their website. One of the platforms that can be used to support high availability is Kubernetes.

Docker is a container engine application that is open source, in its application, the docker is very easy and practical to develop an application using OS-level virtualization (Operating System) which is incorporated into a package called a container [2]. By using the docker, the developer does not need to think about the problem of the burden borne by the server because of the many virtualizations. Containers in the docker have a smaller size than the ordinary, because containers do not need to prepare the full operating system.

Kubernetes is a container management system that can deploy applications on multiple host servers. It based on docker technology which can schedule or scale containers on a cluster. Kubernetes is able to help in avoiding problems such as server downtime, physical damage, operating system failures, etc. to ensure the reliability and continuity of service.

There are 2 ways to form or create a cluster. A cluster can be created by using a physical or virtual server. A virtual server provides storage, network, and software access using remote access services. So there is no need for physical server maintenance such as cable maintaining or hardware maintenance [3].

This paper proposed a high availability storage server using a docker and Kubernetes. A Kubernetes cluster consists of several servers that are mutually sustainable in terms of resource sharing. Three virtual machines were used as Master Node, which runs three main Kubernetes processes. The processes consist of kube-apiserver, kube-controller-manager and kube-scheduler. By the use of cluster servers, the server can meet the high availability requirements. Clusters that have been created are tested using white box testing and scenarios testing.

II. RELATED WORKS

The first link with research or work is a study of the development of a High Availability and Integrity Layer (HAIL) in which allows the server to prove that the data on the server is intact and can be developed. HAIL cryptographically verifies and reallocates files. HAIL has advantages such as a guarantee of file integrity, low overhead, direct communication between client and server, and static file protection [4]. The HAIL's research aims is to test the resistance to disruptions that cause cloud services to problems, provide cloud services with hardware components reliable, providing evidence of efficient file availability by interacting directly with the cloud provider

From early research in [5], the evaluation of the requirement of better data storage management in SAN system was done by develop iSCSI SAN optimization using Internet Protocol (IP) Multipath. The I/O throughput and the ability of iSCSI SAN to handle fail over was tested. The result shows IP multipath enhances the performance and reliability of iSCSI SAN system.

A recent study in dynamic scaling has proposed a method by which resource managers compare the number of VMs (Virtual Machines) on Virtual Machines [6]. Each VM operates and selects a server with more capacity. If the resource manager verifies that the server does not meet the requirements requested, then the resource manager will move the request to another server. But if no server meets the requirements, then the resource manager will create a new VM to fulfill the request. In the second method, the resource manager has to collect the service information available on each VM and choose the server with the fewest services. If the server does not meet the requirements, the request will be moved to another VM. But if there is no VM that meets the requirements, then the request will not be moved anywhere.

Most research uses only single storage media such a hard disk or other devices used on computers. Of course, it is very difficult to synchronize a lot of storage to make work easier [7].

However, this research uses an external storage media ETCD cluster, where each node shares storage. If one node has a problem, then the other node is able to backup storage.

III. System Design

A. System Overview

This system is implemented for making a system with high availability where when the server experiences interference both from outside and inside the server environment. This system is intended to be used by a data analyst to assist data processing operational needs related to the availability of data storage systems.

B. System Component

1) High Availability Server

With the increase in traffic on the server, high server availability is certainly very necessary. One of the advantages of high availability servers is the availability of servers during heavy traffic that allows interference with the server system so that even though there is interference with the server, services can still be run. In addition, the high availability of the server also guarantees that the data available on the server have high availability as well so that the data can be accessed at any time.

One method that can be applied to high availability servers is to use a cluster system on the server. The advantage of a cluster system is that it allows the server to provide services with high availability. When one node has a problem, the other node will detect the failure and reconfigure the system so that the services on the problematic node will be taken over by the other node [8].

The average availability of Kubernetes cluster is defined as follows:

$$Availability = \frac{MTBF}{(MTBF+MTTR)} x100 \quad (1)$$

Where MTBF (Mean Time Between Faults) is the average server uptime, while MTTR (Mean Time to Repair) is the average time needed to restore service after one of the servers has been down.

2) Server Cluster

Clusters are defined as the number of independent servers working on a system. One computer is connected to network through a high-speed communication system where a set of computers can be arranged in parallel or distributed [9]. A cluster uses all the resources that are owned by each computer as if all the resources are in one single computer that allows to anticipate if there is a failure on one node.

With the use of clusters, servers that have limited resources can be helped by other computers. The more servers in a cluster, the more optimal the resulting performance because a cluster will take all available resources into one server with large resources and can also increase the availability of services that are running. One example of using a cluster server is I tested the cluster.

3) Kubernetes

One cluster system that can be used is Kubernetes. It is an arrangement of the docker. It can also be written as K8s that in Greek means pilot. Therefore, Kubernetes can be interpreted as a captain in a ship with a container load [7].

Kubernetes must deploy containers, load balancing containers, and scaling up & descaling down containers. A Kubernetes cluster consists of several nodes (servers) that share roles either as a master or as a worker. The master on Kubernetes functions as a regulator of each pod, detects if there is a problem with the pod and manages the cluster system. While the worker functions to run services from an existing pod. Pods in Kubernetes are services that contain images of containers that can be managed by the Kubernetes system. A pod can be duplicated according to the workers available on the Kubernetes cluster, so that if there is a failure or disruption in a pod, then another pod will take over the task of the pod that is experiencing failure or disruption. Figure 1 is an example of the Kubernetes cluster. Kubernetes has several inseparable components like ETCD, API Server, Kube scheduler and Kube controller manager [10].

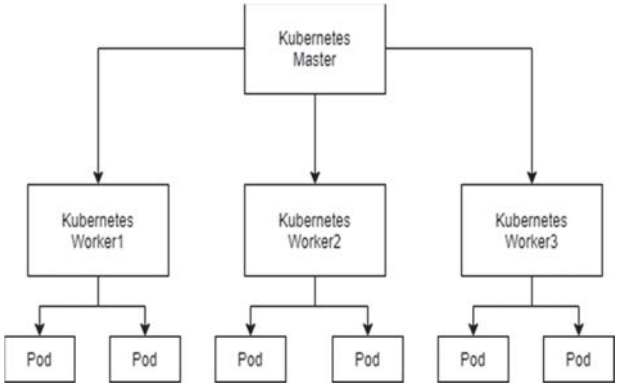

Figure 1. Example of a Kubernetes cluster diagram

4) Docker

Containerization is a way to run applications on the same server or a server cluster. Each application is run isolated in a container called a container. Multiple containers can be run on the same server or cluster server by sharing resources on that server [11].

Docker is a platform created based on containerization technology. Docker can be used as a virtual server that can run several containers at the same time. This can make it easier for developers to develop several applications at once in one server. The container can run services from the docker image where the docker image contains the application you want to run. Docker is also used for running Kubernetes-based cloud services such as EKS, AKS, and GKE [12].

5) Load Balancer

Load Balancing is a method for distributing traffic loads on two or more connection lines equally which such depicted in Figure 2. Load Balancing is used to scale conventionally the traffic load to each of the existing servers to keep the data stored even if the resource is problematic [13]. By using load balancing between servers, the server will not be overloaded by traffic, because the load balancer will distribute overloaded server traffic to servers that have less traffic.

6) MTTF (Mean Time To Failure)

In the field of industry, MTTF (Mean Time to Failure) is a parameter to measure the reliability of a system. MTTF is used to calculate the availability of a valid system when the failure rate is constant or only focus on good conditions (Steady State).

However, MTTF calculations do not apply if the failure rate is not constant [14].

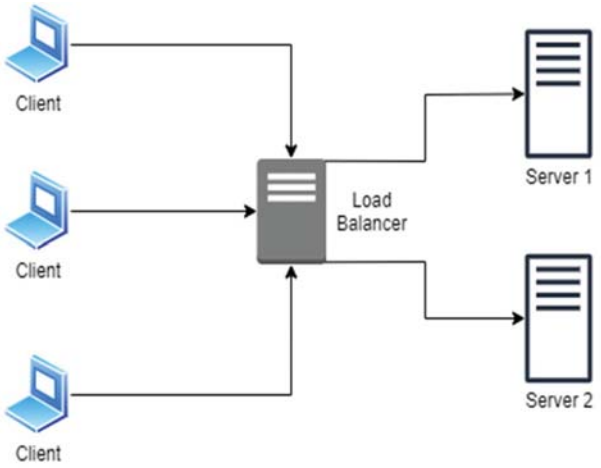

Figure 2. Example of Load Balancing diagram.

$$MTTF = \frac{Uptime}{NumberofFailure} \quad (2)$$

C. Kubernetes Component

1) Kube APIServer

Kube-APIserver is a component in the master node that exposes the Kubernetes API and is a frontend of the master node. One of the uses of the Kube-APIserver is to scale up horizontally the resources needed by the Kubernetes service. In addition, the API Kubernetes also functions to see a list of services that are supported by the Kubernetes cluster and manage the service.

2) ETCD

Etcd is a key-value storage medium that is used to store data from existing pods in the Kubernetes cluster. In its implementation, etcd can be divided into two, namely internal etcd and external etcd. Implementing external etcd requires creating a etcd cluster that supports the creation of high availability Kubernetes clusters. One of the advantages of implementing external etcd is that even though the Kubernetes cluster has been reset, data from the previous Kubernetes cluster can still be reused.

3) Kube Scheduler

Kube scheduler is a component on the master node that functions to observe pods created in the Kubernetes cluster and has not been assigned to the worker node and then selects a worker node to run pod services. Kube scheduler pays attention to several factors in placing pods to worker nodes such as resource requirements, affinity specifications, data localization, and workload from worker nodes.

4) Kube Controller Manager

Kube controller manager is a component located in the master node that functions to run the controller such as replication controller, endpoint controller, node controller, service account and token controller. The functions of these nodes are:

- node controller: Function to provide respond if the number and readiness of the node are reduced.
- Replication controller: Serves to maintain the number of pods to match the number of needs of each replication controller object that is on the system.
- Service account and token controller: Serves to load accounts and API token access on each namespace that are created.

D. Kubernetes Node Component

There are several dependencies needed to run the Kubernetes instances. Dependencies are packages that were needed by an application to run perfectly. In the implementation, there are requirements of several dependencies to run the Kubernetes cluster. The dependencies were Kubelet, Kube proxy, Container runtime and Kubectl. All these components serve to maintain the existing pods in the Kubernetes cluster. The functions of each node component were as follows:

- Kubelet: Serves to ensure that the container is run inside the pod
- Kube proxy: Serves to maintain network rules and to continue the connection shown to a host.
- Container runtime: A software that functions to run containers.
- Kubectl: Is a basic command to get information or change configuration in Kubernetes.

E. Kubernetes Specification

A Kubernetes cluster consists of several servers that are mutually sustainable in terms of resource sharing. The number of servers used is 7 servers with 3 master nodes, 3 worker nodes and 1 load balancer used as endpoints. Each server has limited resources because it is made on a computer with specifications such as Intel Core i7, 13GB DDR3 RAM and 1TB HDD. The list of available servers in the Kubernetes cluster can be seen in Figure 3.

```
NAME          STATUS   ROLES    AGE   VERSION
master-node   Ready    master   2d9h  v1.16.1
master-node2  Ready    master   2d9h  v1.16.1
master-node3  Ready    master   2d9h  v1.16.1
worker01      Ready    <none>   2d9h  v1.16.1
worker02      Ready    <none>   2d9h  v1.16.2
worker03      Ready    <none>   2d9h  v1.16.3
```

Figure 3. List of Nodes in Kubernetes *Cluster.*

F. IP Kubernetes Cluster Configuration

In an out implementation, we used several virtual machines. Three virtual machines were used as Master Node, which runs three main Kubernetes processes. The processes consist of kube-apiserver, kube-controller-manager and kube-scheduler.

Another three virtual machines were used as Worker nodes, which runs the storage system. And one virtual machine runs HAProxy. The HAProxy virtual machine acts as a load balancer by rerouting traffic from the inactive worker node to active worker nodes. Table I shows lists of Kubernetes cluster nodes.

TABLE I. LIST OF CLUSTER IN VIRTUAL MACHINES

No.	Virtual Machine
1.	Master-*Node*
2.	Master-*Node*2
3.	Master-*Node*3
4.	Worker01
5.	Worker02
6.	Worker03
7.	HAProxy

IV. RESULT AND ANALYSIS

A. Whitebox Testing

In this paper, the testing scenarios conducted were the readiness of the system when launched, data access to the system, data transfer availability, and stress testing. We also tested several failure scenarios and how the system responds. In Table II, the Kubernetes cluster was launched. The availability of the cluster and the nodes and pods status were logged. The results were all the clusters, nodes and pods were capable of starting up normally.

TABLE II. KUBERNETES CLUSTER LAUNCH

Cases and Test Results				
No.	**Scenario**	**Expected Output**	**Actual Output**	**Conclusion**
1.	Cluster turned on	Cluster Up	Cluster Up	Valid
2.	Checking nodes status	Status Node Ready	Status nodes Ready	Valid
3.	Checking pods status	Status Pod Running	Status pods Running	Valid

In Table III, the data access testing reported, which has tested whether the pods could be accessed from outside the server. In this test, this research accesses the pods remotely and then execute the pods. The results were all of the pods could be accessed, and all of the pods could be executed.

TABLE III. DATA ACCESS

Cases and Test Results				
No.	**Scenario**	**Expected Output**	**Actual Output**	**Conclusion**
1.	Accessing Pod	Pod can be accessed	Pod can be accessed	Valid
2.	Executing Pod	Pod ca be executed	Pod ca be executed	Valid

The data transfer availability testing was reported in Table IV. This research tests whether the pods could accept any incoming data and could serve any outgoing data requests. The accepted data testing scenario was carried out by uploading 10 GB of data from a remote server to the pod. The data serving scenario was carried out by populating the pod with 10 GB of data and requesting the data from a remote server. The results show that the system could accept 10 GB of incoming data and could server 10 GB of outgoing data.

TABLE IV. DATA TRANSFER AVAILABILITY

Cases and Test Results				
No.	**Scenario**	**Expected Output**	**Actual Output**	**Conclusion**
1.	Transfer data from server to pod using scp	Data can be uploaded into the pod	Data can be uploaded into the pod	Valid
2.	Download data from the pod	Data can be downloaded from the pod	Data can be downloaded from the pod	Valid

Table V is the report of stress testing. The stress testing was carried out to test the maximum number of stressors that could be handled by the system. Stressor is a software that could stress test a server system, by exercising floating-point, integer, bit manipulation and control flow load to the server. The tool used was stress-ng. The results show that the system was capable of handling up to two stressors with multiple stress testing scenarios. However, when the number of stressors increased to either four stressors or four virtual stressors, one of the cluster instances went down. Therefore, the maximum load that could be handled without any instances down was two, and with more load, this research needed a load balancer to increase availability.

TABLE V. STRESS TEST

Cases and Test Results				
No.	**Scenario**	**Expected Output**	**Actual Output**	**Conclusion**
1.	Stress testing uses a stress-ng application using a 2 vm stressor.	All Instances running properly	All Instances running properly	Valid
2.	Stress testing uses a stress-ng application using all 4 stressors.	All Instances running properly	1 instance down	Not Valid
3.	Stress testing uses a stress-ng application using a 4 vm stressor.	All Instances running properly	1 instance down	Not Valid

In case of failure, this research tested how the system responds when one of the virtual machines were turned off. Based on the research testing, the system responds quite well, with no downtime. This test can be seen in Table VI.

TABLE VI. FAILURE SCENARIO TESTING

Cases and Test Results				
No.	**Scenario**	**Expected Output**	**Actual Output**	**Conclusion**
1.	Turn off 1 server	No downtime	Zero downtime Kubernetes cluster	Valid

B. Reliability Testing

Reliability testing is used to calculate the resilience of a server against interference that might occur with a predetermined formula. The reliability tests were carried out by stress-tested the system for 36 hours straight, or 2160 minutes. On average, the number of failures in the testing was two (2). Therefore, we could calculate the mean time to failure (MTTF),

by using the formula (2). The result shows that the average MTTF value is 1080 minutes.

C. Availability Testing

Availability testing was carried out to test the availability of the system in case of node down or failure. The tests were carried out by letting the system run normally for 24 hours and logging the failures which cause a node to down. The MTBF (Mean Time Between Faults) and MTTR (Mean Time to Repair) were logged in minutes. Then we calculate the availability rate with the formula (1).

Samples of the results from the availability testing were logged in Table VII. From the results, we got the availability rate of 100%. It also shows that the server is always available even if one of the nodes is down.

TABLE VII. AVAILABILITY TESTING

	Date of Node *Down*	**Caused Node *Down***	**MTBF (Minutes)**	**MTTR (Minutes)**	***Availability (%)***
1.	30 January 2020	Turned off Server master-node2	960	0	100
2.	5 February 2020	Stress testing uses a stress-ng application using all 4 stressors.	1430.3	0	100
3.	6 February 2020	Stress testing uses a stress-ng application using a 4 vm stressor.	1425.67	0	100

V. CONCLUSION

This research has implemented a system of high availability server using Kubernetes cluster. Based on the white box testing, the system implemented could run normally. And based on the reliability and availability testing, the system was capable of meeting the high availability criteria, by reaching an uptime rate of 100%. The average mean time to failure was 1080 minutes. However, combined with the cluster there was no downtime. The system was capable of meeting the resources request even though one instance was interrupted. And if an instance was down, the services stored in the cluster could still be accessed through other instances.

VI. REFERENCES

[1] Purwanto, Y., Kuspriyanto, Hendrawan, Rahardjo, B. (2014). Minimal Triangle Area Mahalanobis Distance for Stream Homogeneous Group-based DDoS Classification, International Journal on Electrical Engineering & Informatics 10 (2), 2018, 369-383.

[2] Zhao, A., Huang, Q., Huang, Y., Zou, L., Chen, Z., & Song, J. (2019). Research on Resource Prediction Model Based on Kubernetes Container Auto-scaling Technology. IOP Conference Series: Materials Science and Engineering, 569, 052092.

[3] Santi, D., Rumani, R. M., & Purwanto, Y. (2013). Implementasi Dan Analisis Performansi Raid Pada Data Storage Infrastructure As a Service (Iaas) Cloud Computing. JSM STMIK Mikroskil, 14(2), 99–107.

[4] Bowers, K. D., Juels, A., & Oprea, A. (2009). HAIL: A high-availability and integrity layer for cloud storage. Proceedings of the ACM Conference on Computer and Communications Security, 187–198. https://doi.org/10.1145/1653662.1653686

[5] Ariefianto, T., Purwanto, Y., Wiratama, H. (2013), Storage Area Network based-on Internet Small Computer Standard Interface optimization using Internet Protocol Multipathing, International Conference of Information and Communication Technology (ICoICT), Bandung, 20-22 March 2013, https://doi.org/10.1109/ICoICT.2013.6574590

[6] Namikata, M., Sato, K., Iizuka, K., & Ueda, K. (2015). Methods of Dynamic Scaling with VM for High Availability Server Clusters. 0086(c), 115–117

[7] Ruriawan, M. F., Anggono, B., Siahaan, I. A., & Purwanto, Y. (2019). Development of digital evidence collector and file classification system with K-Means algorithm. Proceedings - 2019 IEEE Asia Pacific Conference on Wireless and Mobile, APWiMob 2019, 64–68. https://doi.org/10.1109/APWiMob48441.2019.8964232

[8] Dias, D. M., Kish, W., Mukherjee, R., & Tewari, R. (1996). Scalable and highly available web server. Digest of Papers - COMPCON - IEEE Computer Society International Conference, 85–92. https://doi.org/10.1109/cmpcon.1996.501753

[9] Ljubojević, M., Bajić, A., & Mijić, D. (2019). Implementation of High-Availability Server Cluster by Using Fencing Concept. 2019 18th International Symposium INFOTEH-JAHORINA, INFOTEH 2019 - Proceedings, (March), 20–22. https://doi.org/10.1109/INFOTEH.2019.8717752

[10] Marathe, N., Gandhi, A., & Shah, J. M. (2019). Docker swarm and kubernetes in cloud computing environment. Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2019, 2019-April(Icoei), 179–184. https://doi.org/10.1109/icoei.2019.8862654

[11] Shah, J., & Dubaria, D. (2019). Building modern clouds: Using docker, kubernetes google cloud platform. 2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC 2019, 184–189. https://doi.org/10.1109/CCWC.2019.8666479

[12] Pereira Ferreira, A., & Sinnott, R. (2019). A performance evaluation of containers running on managed kubernetes services. Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom, 2019-Decem, 199–208. https://doi.org/10.1109/CloudCom.2019.00038

[13] 'DevOps; Puppet, Docker and Kubernetes – Learning path' by Thomas Uphill, Arundel, Khare, Saito, Lee and Carol Hsu, Packt Publications, First Edition, 2017

[14] Ford, D., Labelle, F., Popovici, F. I., Stokely, M., Truong, V. A., Barroso, L., Quinlan, S. (2019). Availability in globally distributed storage systems. Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2010, 61–