UAD Universitas Ahmad Dahlan

TELKOMNIKA
(Telecommunication Computing Electronics and Control)

http://journal.uad.ac.id/index.php/TELKOMNIKA
ISSN 1693-6930, e-ISSN 2302-9293

SJR SCImago Journal & Country Rank
Q3 Electrical and Electronic Engineering SJR 2020 0.258

Vol 17, No 1

February 2019

DOI: http://dx.doi.org/10.12928/telkomnika.v17i1

Table of Contents

TELKOMNIKA Telecommunication, Computing, Electronics and Control
ISSN: 1693-6930, e-ISSN: 2302-9293
Universitas Ahmad Dahlan, 4th Campus
Jl. Ringroad Selatan, Kragilan, Tamanan, Banguntapan, Bantul, Yogyakarta, Indonesia 55191
Phone: +62 (274) 563515, 511830, 379418, 371120
Fax: +62 274 564604

UAD Universitas Ahmad Dahlan

TELKOMNIKA

(Telecommunication Computing Electronics and Control)

http://journal.uad.ac.id/index.php/TELKOMNIKA
ISSN 1693-6930, e-ISSN 2302-9293

SJR SCImago Journal & Country Rank

Q3 Electrical and Electronic Engineering SJR 2020 0.258

Editorial Team

TELKOMNIKA Telecommunication, Computing, Electronics and Control
ISSN: 1693-6930, e-ISSN: 2302-9293
Universitas Ahmad Dahlan, 4th Campus
Jl. Ringroad Selatan, Kragilan, Tamanan, Banguntapan, Bantul, Yogyakarta, Indonesia 55191

Phone: +62 (274) 563515, 511830, 379418, 371120
Fax: +62 274 564604

Source details

Telkomnika (Telecommunication Computing Electronics and Control)

Scopus coverage years: from 2011 to Present

Publisher: Universitas Ahmad Dahlan

ISSN: 1693-6930 E-ISSN: 2087-278X

Subject area: Engineering: Electrical and Electronic Engineering

Source type: Journal

CiteScore 2020
2.2

SJR 2020
0.258

SNIP 2020
0.746

View all documents > | Set document alert | Save to source list | Source Homepage

CiteScore | CiteScore rank & trend | Scopus content coverage

CiteScore 2020

$$2.2 = \frac{2{,}987 \text{ Citations } 2017 - 2020}{1{,}360 \text{ Documents } 2017 - 2020}$$

Calculated on 05 May, 2021

CiteScoreTracker 2021

$$2.6 = \frac{3{,}434 \text{ Citations to date}}{1{,}323 \text{ Documents to date}}$$

Last updated on 04 November, 2021 • Updated monthly

CiteScore rank 2020

Category	Rank	Percentile
Engineering └ Electrical and Electronic Engineering	#373/693	46th

View CiteScore methodology > | CiteScore FAQ > | Add CiteScore to your site

TELKOMNIKA, Vol.17, No.1, February 2019, pp.218~225
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v17i1.11599

Private cloud storage implementation using OpenStack Swift

Agustinus Noertjahyana*, Juan Reno, Henry Novianus Palit, Justinus Andjarwirawan
Informatics Engineering, Faculty of Industrial Technology, Petra Christian University
Siwalankerto St., Jawa Timur, Indonesia, 121-131, 031-2983456
*Corresponding author, e-mail: agust@petra.ac.id

Abstract

The use of distributed and parallel computer systems is growing rapidly, requiring an appropriate system to support its work processes. One technology that supports distributed computer systems is cloud computing. This system can generate the need to maximize the use of existing computing resources, one of which is in the form of cloud-based storage. The computer laboratory of Informatics Department of Petra Christian University has very large resources, but they have not been optimized in the utilization of existing storage devices. This condition gives the idea to utilize computers in the laboratory with cloud, so the storage can be used well. This implementation used the OpenStack cloud framework, which could provide IaaS service. From some existing OpenStack services, storage management used OpenStack Swift on its processing. OpenStack Swift is a cloud-based storage service that leverages various computing resources. After the implementation process, testing was done by way of data management, so storage could store, retrieve, and delete data. In addition, testing was also done by turning off some physical machines to ensure cloud services could remain well accessible, and measure the speed of data transfer in cloud storage. The resulting data was used to evaluate the cloud storage systems that had been created.

Keywords*: cloud, distributed system, infrastructure as a service, OpenStack Swift, OpenStack, paralel computing, private cloud storage*

1. Introduction

The use of distributed and parallel computer systems is growing rapidly, requiring an appropriate system to support its working processes. Along with the evolving needs that exist, a computer system is also required to work quickly and have a low fault tolerant. One technology that supports distributed computer systems like this is cloud computing.

Cloud Computing is a combination of the use of computer technology and Internet-based development that is an abstraction of the hidden infrastructure [1-3]. In general, cloud computing utilizes more than one computer that has been connected to each other through a network. This distributed system can generate the need to maximize the use of existing computing resources, one of which is in the form of storage in the form of cloud storage. Computer systems like this can be pretty much found around us, some of them are in the computer laboratory of Informatics Department of Petra Christian University.

The computer lab of Petra Christian University's Informatics department is a considerable investment. However, in reality the use of the computers in the laboratory is not optimal in terms of lecturing activities, and asa storage device. Each computer in the laboratory has an average of 500 Gigabytes of storage. But the use of the storage is often uneven because one computer's storage may be used up, while other computers have plenty of storage space left. Such conditions, provide ideas to utilize computers in the laboratory with the cloud method for more efficient use of storage. With the specifications and existing computer facilities, they can be utilized to become a private cloud computing system.

2. Literature Review

Cloud computing proves to be so disruptive to provide anyone with on demand remote access to a large pool of third-party computing resources and services [4-8]. Private cloud is a

Received June 7, 2018; Revised October 3, 2018; Accepted November 5, 2018

Cloud Computing service, provided to meet the internal needs of an organization/company. In a company, usually the IT Department is responsible as the provider of cloud services, and other divisions within the company as its users [9]. As a Service Provider, of course, IT Department must be responsible for the service to run well in accordance with service quality standards that have been determined by the company, either infrastructure, platform or existing applications. There are several advantages in using private cloud, i.e.

- Data security is guaranteed because the internal organization or company manages its own system security.
- The internet bandwidth is saved when the service is accessed only from the organization's internal network.
- Business process does not depend on internet connection, but it still depends on local internet connection (intranet).

On the other hand, there are also some disadvantages that can arise with the use of private cloud, i.e. it can be a large investment because the internal company or organization itself must prepare its infrastructure, It takes manpower to care for and ensure the service goes well and smoothly. By using less skilled personnel, the system security is less secure because of poor settings.

Cloud Computing is a combination of the use of computer technology in a single computing and development with an internet base. According to NIST [10], there are five characteristics of a system called cloud computing, among others, as follows:

- Resource Pooling, which is a physical or virtual computing resource collected by service providers to meet the needs of many customers with multi-tenant models. These computing resources can be used dynamically by customers to meet their needs.
- Broad Network Access, which is a cloud service provider capability through a network that can be accessed using multiple end devices.
- Measured Service, which is a service to optimize and monitor services related to computing resources such as bandwidth, storage, processing, and so on.
- Rapid Elasticity, which is a service from cloud providers can be used by cloud consumer dynamically to raise or lower the service capacity. The service capacity provided is usually unlimited, and the consumer service can freely and easily select the desired capacity at any time.
- Self Service, which is a configuration service for Cloud Consumer independently services that want to be used through a system, without the need of human interaction with the cloud provider.

Beyond the existing characteristics, cloud computing has three types of services offered to customers or users concerned [11]. The services are described as follows: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS).

2.1. OpenStack

OpenStack is a cloud platform that consists of several free and open source softwares to provide Cloud IaaS service both in personal and in large scale [12]. It can be interpreted that OpenStack is a service that acts as a middleware to unify the diversity of layers such as network, storage, hardware, operating system, and so forth. OpenStack consists of many parts that have different functions. Quoted from the OpenStack document [13], there are several components that are parts of OpenStack. These components include:

- Nova, whichis the main computing engine in OpenStack to deploy and manage large numbers of virtual machines and instances in handling computational tasks.
- Swift, which is a storage system for objects and files.
- Cinder, which is a block storage component, which is more analogous to the idea of a traditional computer that can access a specific location on a disk drive.
- Neutron, which provides networking capabilities for OpenStack. This helps to ensure that each component of the OpenStack deployment can communicate with each other quickly and efficiently.
- Horizon, which is the OpenStack dashboard of graphical interface. In this dashboard provides system administrators to see what is happening in the cloud and manage it as needed.
- Keystone, which is the service identity that is central to all usage in OpenStack cloud. All services provided by the cloud must have permission to use the service.

- Glance, which is an image service for OpenStack that refers to an image (or virtual copy) of the hard disk.
- Ceilometer, which is a telemetry service within the cloud to provide billing services to individual users.
- Heat, which is an orchestration component of OpenStack, to store the needs of cloud applications in a file that defines what resources are required for the application. This is necessary in managing the infrastructure to run cloud services.

2.2. OpenStack Swift

OpenStack Swift is popular open source software used to build very large-scale storage systems [14]. OpenStack Object Swift is a scalable multi-tenant object storage system, and it can manage unstructured data [15]. In this case, OpenStack Swift has several components to support existing object storage services. Existing components include the following:

- Proxy server, which plays a role in uploading files, modifying metadata, and creating containers. It can use cache to improve its performance.
- Account servers, which manage accounts related to Swift service.
- Container servers, which manage container mappings or folders contained in OpenStack Swift.
- Object servers, which manage actual objects on storage nodes like files and so on.
- Periodic process, which serves as a replication service in ensuring consistency or availability within the cluster.
- WSGI middleware, which authenticate OpenStack Identity.
- Swift Client, which serves as a user facility in sending user permissions commands via the command line.
- Swift-init, which creates a script that initializes in the ring file.
- Swift-recon, which retrieves information about clusters that have been collected by the swift-recon middleware.

3. System Planning

3.1. Working Scheme

OpenStack Swift provided the IaaS service with the system work scheme used as Figure 1.

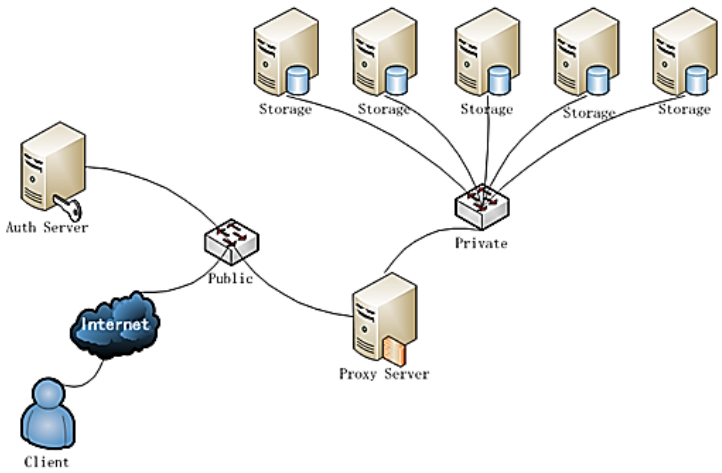

Figure 1. OpenStack Swift working scheme

As in Figure 1, the explanation of the Openstack Swift work scheme is as follows:

1. Auth server, in this case, is also known as node controller role in client authentication process that accessed the system. The node controller installed the keystone service which

was the means of authentication in OpenStack. The client provided information in the form of user and password to be verified by keystone service.

2. If the client authentication was successful on the controller node, then the process would proceed to the proxy server node. In the proxy server node, files sent by the client were processed for storage in the available storage nodes. In general, proxy server nodes played a role in every data management activity that involved storage nodes, i.e. storage, retrieval, and data deletion. Proxy server nodes also managed sync, balancing, and data replication processes.
3. Data that were received by proxy server node were then processed to the storage node. The storage process was divided into two, namely the container and the object. Object was stored in each container so that one container could have many objects, while one object was only contained in one container. The proxy server node passed the data and was received by the storage node in the hashing form. The successful data storage process returned the output to the client in the form of a successfully saved file name along with its hashing code.
4. During the process of retrieving and deleting data from the storage node, the proxy server sent a hashing code to recognize the file to be retrieved or deleted. This process did not return any output to the client, but the client could check directly the changes to the directory or object list in the system.

3.2. Network Design

From the network design shown in Figure 2 for the use of OpenStack Swift, the network used was 192.168.11.0 with subnet mask/24, ie 255.255.255.0. The Default Gateway that was used in accordance with the IP Address was owned by the router, i.e. 192.168.11.1. While the DNS server used was 203.189.120.4 and 203.189.120.7. This network used several computers in the this research, which were divided into one controller node, one proxy server node, three storage nodes, and one client.

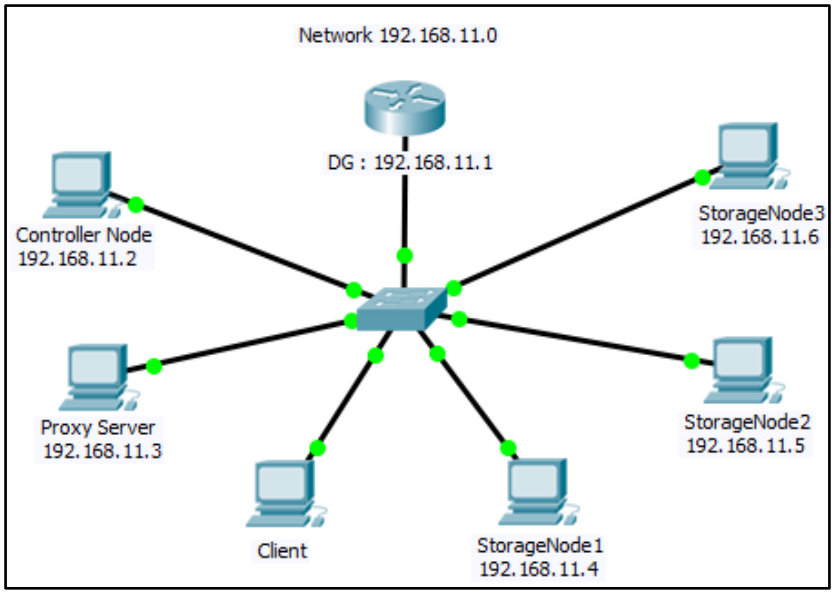

Figure 2. Network diagram configuration

3.3. Storage Node Disk Partition

The disk partition was performed only on the Storage Node, which provided the capacity for the Operating System as well as OpenStack Swift itself. This partition was created because Swift could not be performed on a disk used by other system. In this case, the disk partition used for Swift was 100 Gigabyte, while Ubuntu used 250 Gigabyte. The disk partition was done as in Figure 3.

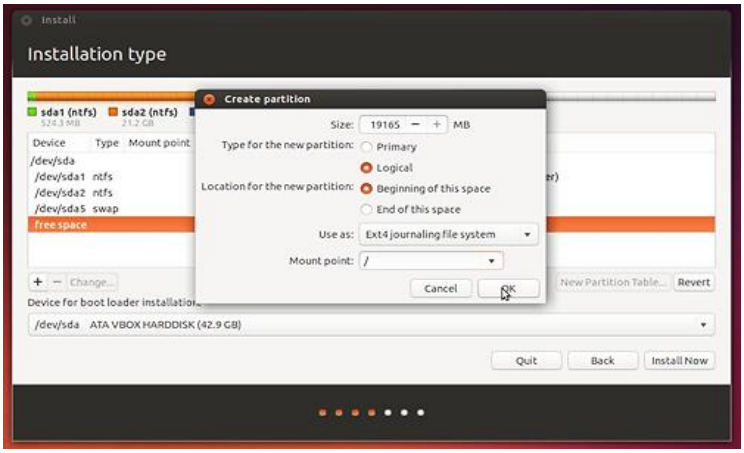

Figure 3. Disk partition in storage node

3.4. Network Configuration

Routers played a role in forming a new network, which were used in the Openstack system. It was intended that the ongoing process did not disrupt the network outside the system. The router used was the Buffalo AirStation Router, with the following steps:

1. The admin performed a Router reset, then accessed to 192.168.11.1 address with username using root without password. This address displayed the configuration page of the existing router.
2. The admin configured in the Wireless Connection menu section to manage the existing network with existing DHCP IP Pool. In this case, the admin created a new network on 192.168.11.0 with Pool of 64 as shown in Figure 4.

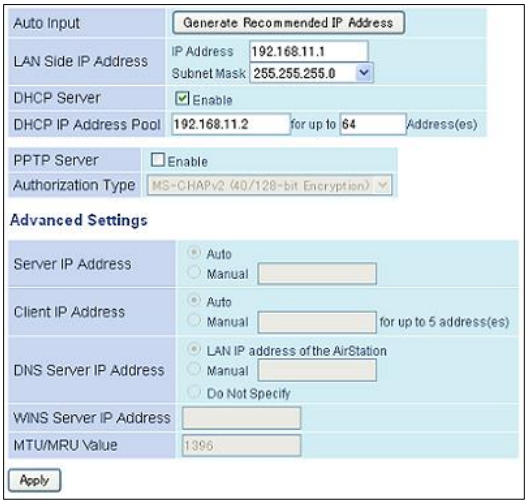

Figure 4. Configuration on router

3.5. IP Address Configuration

For each node, it needed to be assigned a static IP Address, so that the available nodes could be identified with each other. The subnet mask used was 255.255.255.0 or/24. The default gateway used was 192.168.11.1 in accordance with the existing IP Address on the Router. The DNS server used was 203,189,120.4 based on the existing network. These three configurations were applied to all nodes used. On the other hand, the node controller had IP Address 192.168.11.2, the proxy server node had IP Address 192.168.11.3, and the storage node had a range of IP addresses starting from 192.168.11.4 to a number of existing storage nodes.

3.6. Basic Environment

One of the settings that needed to be set up to install on OpenStack was the Basic Environment, or basic environment. This environment became the physical basis in shaping the system. Some of the required environments were storage nodes and node controllers with the Ubuntu Server 14.04 LTS operating system. For each node used, the hardware specifications used were as follows: Processor: Intel Core i5-3340@3.1 GHz (4 cores/4 threads), RAM: 16 GB, Disk: 250 GB and Connection: 1 interface 100Mbps Ethernet.

In addition, the installation of the basic components was required by OpenStack in the form of OpenStack packages. In installing OpenStack packages, the Juno cloud repository needed to be added to the source-list of the Advanced Package Tool (APT). After adding the cloud repository, the existing APT needed to apt-get update and apt-get dist-upgrade.

3.7. Framework and Application

The framework used was OpenStack Pike version, so the software was a number of components that operated on OpenStack based cloud system. These components were separated into several nodes: a node controller, a proxy server node, as well as multiple storage nodes. In the design of private clouds in the laboratory, the components used were shown in Figure 5.

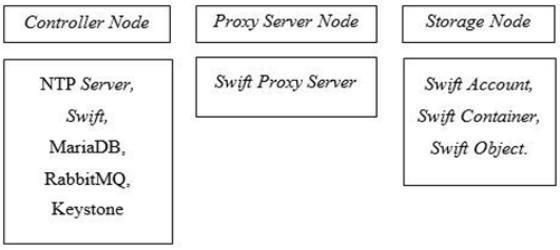

Figure 5. Nodes schema

4. Testing and System Evaluation

4.1. Data Management

In the storage, data management process became the main priority. The basic form of data management was when the system could run its role to store, retrieve, and delete the files it contains. This was intended for the storage to be run and used well by the user. There were three kinds of data management, namely storage, retrieval, and data deletion.

4.2. Storage Node Deactivation

The second test was related to distributed systems, which had more than one storage node to form a cloud storage. Testing was done by turning off one or more storage nodes and evaluating system performance. The evaluation was related to the process of data management in the storage that had been made, and indicated whether or not there was disruption in the process of storing, retrieving, and deleting data. The storage node disabling conditions were illustrated in Figure 6 and 7.

There were three kinds of testing related to storage node disabling, i.e. storage, retrieval, and data deletion. In testing for data retrieval, two cases were used. The first case was the retrieval when one or more storage nodes were turned off as in Figure 6, with storage of all storage nodes lit up. The second was to retrieve data from the system associated with testing data storage, where the condition of some storage nodes was turned off as shown in Figure 6. From the first case test, the system succeeded in providing the data to be stored by the client.

Testing in the storage of a file try.txt was done with two things, namely disabling only on storage node 3 as in Figure 6 as first case,and disabling the storage node 2 and storage node 3 as in Figure 6 as second case.

In the process of deleting data, there were two types of testing performed. The first case was when only one storage node was active, assuming the file storage in the storage node was evenly distributed as shown in Figure 6. The second case was when at least two active storage nodes with file storage in the storage node were evenly distributed as in Figure 7. From the first

case, the client did not succeed in deleting the test.txt file from the storage node swift. Then, in the second case, the same case as the first case occured where the client also could not delete the files contained in the storage node. The resulting output was the same, i.e. Service Unavailable (Error 503). Based on the error, it showed that file deletion had to involve all storage nodes in active condition. Test conditions were not only on the object, but also on the process of removal of containers in swift.

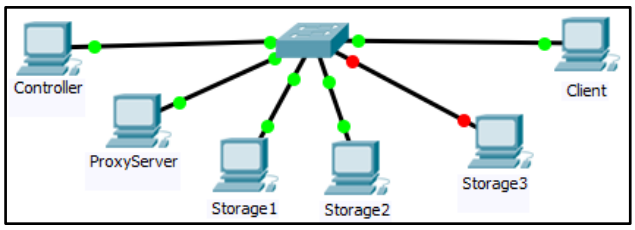

Figure 6. Storage node no.3 deactivation

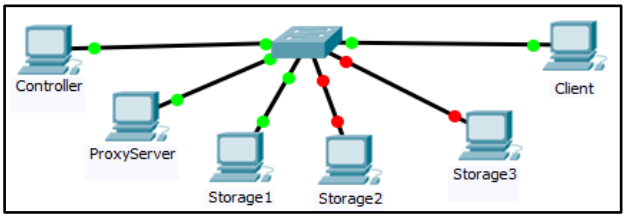

Figure 7. Storage node no.2 and no.3 deactivation

4.3. Data Transfer

The third test was related to the speed of data transfer from client to system in the form of storage node. This test was performed to compare the speed of data transfer in swift with disk to disk in general. In testing data transfer, three types of files with varying sizes were used, i.e.

1. Try1.zip for 10 kilobytes as the first case
2. Try 2.zip for 1 megabyte as the second case
3. Try3.zip for 100 megabytes as the third case

Recording was done in the process of storing, retrieving, and deleting data in the storage node, and using measurements in seconds. Data transfer testing measured three main things, namely storage, retrieval, and data deletion.

4.3.1. Store Data

For the first case, the client uploaded the try1.zip file to the swift system in the three active storage nodes. For the second case, the client uploaded the try2.zip file to the swift system in the three active storage nodes. For the third case, the client uploaded the try3.zip file to the swift system in the three active storage nodes. The results of the data storage test are shown in Table 1.

Table 1. Store Data Time Testing

No.	Condition	Case 1	Case 2	Case 3
1	All Storage Node : Active	28.17s	28.75s	41.04s
2	One of Storage Nodes Inactive	32.56s	35.18s	58.22s

4.3.2. Retrieve Data

For the first case, the client downloaded the file coba1.zip from the swift system in the three active storage nodes. For the second case, the client downloaded the try2.zip file from the swift system in the three active storage nodes. For the third case, the client downloaded the file coba3.zip from the swift system in the three active storage nodes shown in Table 2.

Table 2. Retrieve Data Time Testing

No.	Condition	Case 1	Case 2	Case 3
1	All Storage Node : Active	18.23s	47.08s	>15m
2	One of Storage Nodes Inactive	22.52s	1m12s	>20m

4.3.3. Delete Data

For the first case, the client deleted delete1.zip file in the swift system in the three active storage nodes. For the second case, the client deleted delete2 a try2.zip file in the swift system in the three active storage nodes. For the third case, the client deleted delete3 cobaz.zip file in the swift system in three active storage nodes. After the entire file deletion process succeeded, the execution time of file deletion processing is shown in the Table 3.

Table 3. Delete Data Time Testing

Condition	Case 1	Case 2	Case 3
All Storage Nodes Active	16.64s	19.82s	22.34s

5. Conclusion

From the design results of the private cloud storage system in the laboratory, it can be concluded that: The development of cloud-based private storage in OpenStack Swift can overcome the data loss that may occur due to the destruction of a physical machine in a computer lab. The stored data can still be accessed properly by using other computers in one network and the same system. With the existence of private storage system through swift, unused storage can be utilized in large amount in each physical machine. Each physical machine has a hard disk of one Terabyte that can be used in part for private storage. The use of this capacity in addition to overcome the data loss can also be used to store all types of files through the client connected in the system.

References

[1] M Jamil. Cloud Computing Teori dan Aplikasi. Yogyakarta: Penerbit Deepublish. 2016.
[2] Z Wang, H Chen, Y Fu, D Liu, Y Ban. Workload balancing and adaptive resource management for the swift storage system on cloud. *Future Generation Computer Systems*. 2015; 51: 120-131.
[3] L Sen, Chan FTS, Yang J, Niu B. Understanding the effect of cloud computing on organizational agility: An empirical examination. *International Journal of Information Management*. 2018; 43: 98-111.
[4] D Pietro, Riccardo, Giacobbe, Maurizio, Puliafito, Carlo, Scarpa, Marco. J2CBROKER as a Service: A Service Broker Simulation Tool Integrated in OpenStack Environment. 2018: 261-277.
[5] N Frederic, Y Yang. A Literature Survey on Resource Management Techniques, Issues and Challenges in Cloud Computing. *TELKOMNIKA Telecommunication Computing Electronics and Control.* 2017; 15(4): 1918-1928.
[6] J Jintao, Y Wensen, G Lei. Research on Batch Scheduling in Cloud Computing. *TELKOMNIKA Telecommunication Computing Electronics and Control.* 2016; 14(4): 1454-1461.
[7] I Pietri, R Sakellariou. Mapping virtual machines onto physical machines in cloud computing: A survey. *ACM Computing Surveys (CSUR).* 2016; 49(3): 49.
[8] PT Endo, *et al*. *Self-organizing strategies for resource management in Cloud Computing: State-ofthe-art and challenges*. Cloud Computing and Communications (LatinCloud), 2nd IEEE Latin American Conference on. 2013.
[9] PGSC Nugraha. *Implementasi Private Cloud Computing Sebagai Layanan Infrastructure as a Service (IAAS) Menggunakan OpenStack*. Bali: Jurnal Ilmiah Ilmu Komputer Universitas Udayana. 2015.
[10] NIST. SP 800-145. The NIST Definition of Cloud Computing. Computer Security Resource Center. 2011.
[11] S Martinelli, H Nash, B Topol. Identity, Authentication, and Access Management in OpenStack: Implementing and Deploying Keystone. California USA: IBM Incorporation. 2015.
[12] I Anwar. Cloud Matrix Book. Meruvian Cloud Team. 2011.
[13] T Rosano. An Overview of OpenStack Architecture. Portugal: Polytechnic Institute of Coimbra. 2014.
[14] Kota T, Masahiro S. Recent Activities Involving openstack swift. Regular Articles. *NTT Technical Review*. 2015; 13(12).
[15] OpenStack. *Object Storage*. URI=https://docs.openstack.org/security-guide/object-storage.html. 2017.

ICW 2018
TELKOMNIKA

2018 1st INTERNATIONAL CONFERENCE AND WORKSHOP ON
TELECOMMUNICATION, COMPUTING,
ELECTRICAL, ELECTRONICS AND CONTROL

September 18-21, 2018 | Royal Ambarrukmo Hotel, Yogyakarta, Indonesia

CERTIFICATE OF APPRECIATION

is awarded to

Henry Novianus Palit

In recognition and appreciation of your contribution as

Presenter

for paper entitled

Private Cloud Storage Implementation Using OpenStack Swift

Tole Sutikno, Ph.D.
Director, LPPI UAD

Anton Yudhana, Ph.D.
Conference Chair