Indonesian Journal of Electrical Engineering and Computer Science
Vol. 4, No. 1, October 2016, pp. 155 ~ 164
DOI: 10.11591/ijeecs.v4.i1.pp155-164

Private Cloud Storage Using OpenStack with Simple Network Architecture

Albert Sagala, Ruth Marlina Hutabarat
Cyber Security Research Centre (CSRC), Faculty of Electrical and Informatics Engineering,
Del Institute of Technology, North Sumatra, Indonesia
*Corresponding author, e-mail: albert@del.ac.id;ruth.hutabarat@del.ac.id

Abstract

The development of cloud computing was highly developed and increasingly in demand by many parties. Cloud computing was very helpful to user to perform data storage without preparing own physical device. Cloud storage is a part of cloud computing where it can facilitate user to upload and download files, view file status. The main barrier to implement the private cloud was the difficulty in configuration of many components involved. These conditions minimize the implementation of the private cloud in many parties. In this research, we demonstrate cloud storage architecture simply by using only three nodes with OpenStack. The first node is the controller, the second node is object storage and the third is object storage. Elaborating the component that should be activated to implement the private cloud. The result of this research will help parties who want to build private cloud storage with simple network architecture using OpenStack Object Storage.

Keywords*: cloud computing, openstack object storage, private cloud storage, openstack, virtualization*

1. Introduction

Cloud Computing is a combination of computer technology (computing) which is developed based on Internet (cloud) that can be accessible from anywhere. It has a model to collect the computing resources such as networks, servers, storage, applications and services [1]. It can also be called as Cloud Storage. It is a digital data storage technology that utilizes the virtual server as the storage media and have basic concept of Cloud Computing. Cloud computing is still an evolving paradigm. Its definitions, use cases, underlying technologies, issues, risks, and benefits will be refined in a spirited debate by the public and private sectors [2].

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. OpenStack project is an open source cloud computing platform that supports all types of cloud environments. This project need to be simple implementing, massive scalability, and having rich set of features [3]. This paper proposes the system architecture for building a private cloud which is capable of providing IaaS and PaaS as a service over the local area network. The proposed system architecture has been testing on a private cloud storage using OpenStack Object Storage with simple network architecture.

The rest of paper are organized as follow: Section 2 presents some of the most relevant related work project about Cloud and OpenStack object storage (swift); Section 3 describe the OpenStack services and architecture environment; Section 4 describe networks structure, OpenStack object storage (swift) architecture and components; Section 5 is implementation and testing of private OpenStack object storage (swift); and section 6 is main conclusions.

D Lakshmi Kurup [4] explains about *Comparative Study of Eucalyptus, OpenStak and Nimbus*. In paper discusses to deploy public and private cloud there are many open source software platform available such as Eucalyptus, OpenStak and Nimbus. Also in paper comparison presented benefit enterprises as well as research institutes in selecting best open source platforms to meet their technology demands. Sefraoui Omar [5] in paper *OpenStak: Toward an Open Source Solution for Cloud Computing* has a purpose to provide the computer industry with the opportunity to build a hosting architecture, massively scalable which is

Received May 10, 2016; Revised August 28, 2016; Accepted September 1, 2016

completely open source, while overcoming the constraints and the use of proprietary technologies. Sachdev Sidharth [6] explain in paper *How to Deployment of Private Cloud using OpenStak: An Open Source Cloud Computing Solution for Small and Mid-sized Organizations*. The purpose in paper describes the openness, adaptability and scalability which provide the flexibility to control the whole cloud infrastructure by using a dashboard service at administrator and client level and describes the establishment and deployment of cloud infrastructure (IaaS) for private network where users provide the tools for creating and managing virtual machines over the existing resources.

And then, research Peter Sempolinski [7] discusses *A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus*. This topic explains some of the common challenges that emerge in setting up any of these frameworks and suggest avenues of further research and development. These include the problem of fair scheduling in absence of money, eviction or preemption, the difficulties of network configuration, and the frequent lack of clean abstractions. Next research, Atre Aniruddha [8] in paper *SDK to Ease Access of Openstack Storage Swift Cloud Services* explain a library providing CRUD functionalities for CDMI based object storage and secondly an appropriate Graphical User Interface to provide simplicity. Motivation of this project is to fill the gap for use of CDMI implementation over OpenStack Swift by any novice user by developing a .NET SDK for windows audience.

OpenStack Object Storage (Swift) can integrate with another components like Drupal [9]. These projects using OpenStack Object Storage (Swift) integrated Drupal by module. This module gives scalability for Drupal to handle growing amounts of data and concurrency without any impact on performance and can file management integrated the OpenStack Object Storage (swift) as a stream wrapper for saving Drupal's files on the object store. OpenStack Object Storage (Swift) on File [10]. This project is to enables users to access the same data, both as an object and as a file. Swift-on-File can be especially useful in cases where access over multiple protocols is desired. For example, imagine deployment where video files are uploaded as objects over Swift's REST interface and legacy video transcoding software access those videos as files.

2. Research Method

First method in our research is study in OpenStack service and OpenStack architecture environment. The second method is design network structure, Swift architecture and Ring Architecture, and components. The last method in our research implementation OpenStack Object Storage.

2.1. OpenStack Services

In this case will discusses about OpenStack services will be shown on Figure 1.

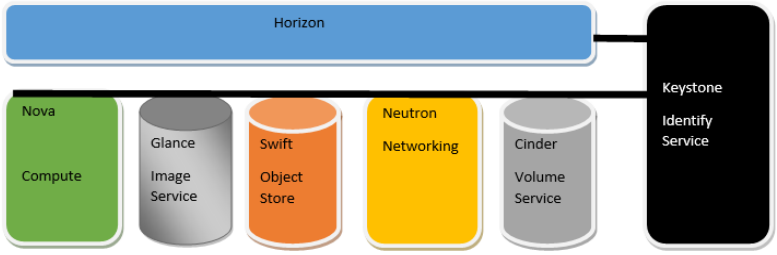

Figure 1. OpenStack Services

1. **Keystone (Identify Service)**

Keystone is a service used by OpenStack identity for authentication and high-level authorization. Authentication currently supports token-based and user authorization services.

2. **Nova (Compute)**

Nova is a service to manage cloud computing system. This component is also used to view information about the services signifies that OpenStack is running properly. If the service were raised state of error its means there are problems in the system OpenStack.

3. Neutron (Networking)

Networking service allows users to build multiple networks for each instance. This service also provides plugins for wide range of products that support virtual networking.

4. Glance (Image Service)

Glance is a service to manage images that will be used in manufacture OpenStack instance. This component will provide a catalog of services for storing and retrieving virtual disk image. This component is designed as standalone component for other components used in order to organize the collection of a large virtual disk image.

5. Cinder (Volume Service)

Cinder is a service that provides storage blocks are separate from the storage server. Storage block is called volume. Volumes have been made integrated with the compute nodes and the process of making the appropriate volume of the user desires.

6. Swift (Object Storage)

Provide data storage services and retrieve unstructured data objects via REST full, HTTP based API. This service can also store large amounts of data and can copies with redundancy.

7. Horizon (Dashboard)

Dashboard is a service that provides a graphical interface that can be accessed by the administrator. With a graphical interface that is provided by OpenStack can facilitate administrator for management of cloud computing such as setting the RAM, hard drive settings, memory settings, the manufacture of the instance.

2.2. OpenStack Environment Architecture

OpenStack services will be interconnected. Keystone will provide authentication and authorization service for other OpenStack services, also provides a catalog of endpoints for all OpenStack services. Architecture OpenStack Environment will be presented on Figure 2.

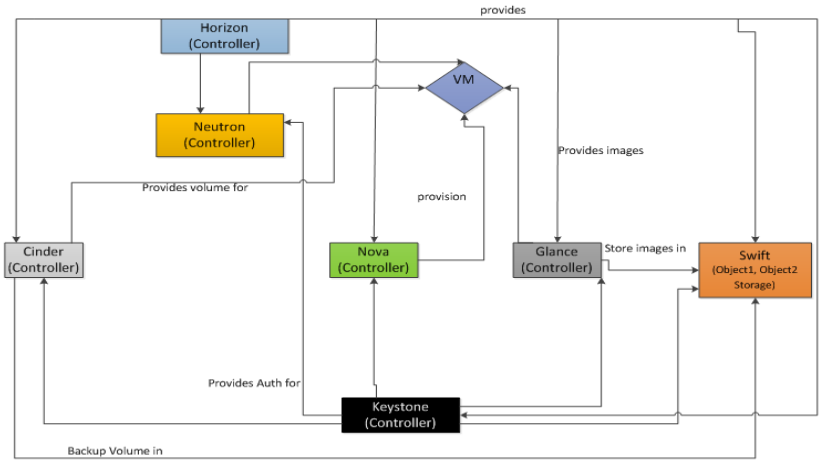

Figure 2. OpenStack Environment Architecture [11]

Figure 2 show the architecture of OpenStack where all nodes should be interconnected to perform its functions in accordance with configuration services performed on nodes. Nodes will respond if it is successfully configured as needed.

The functions nodes as follows [11]:

1. Controller Node

Controller node as the central node that will run the identify service, image service, compute and networking management. In controller node can support services such as SQL databases, message brokers, Network Time Protocol (NTP) and running most of the service block storage, object storage, telemetry and orchestration.

2. Object1 Storage Node

An object1 storage node also contains disk that is used by the object storage service for storing accounts, and as container object.

3. Object2 Storage Node

This node also contains disk that is used by the object storage service for storing accounts, and as container object.

2.3. Network Structure

The structure to build cloud storage with OpenStack Object Storage (Swift) as shown on Figure 3.

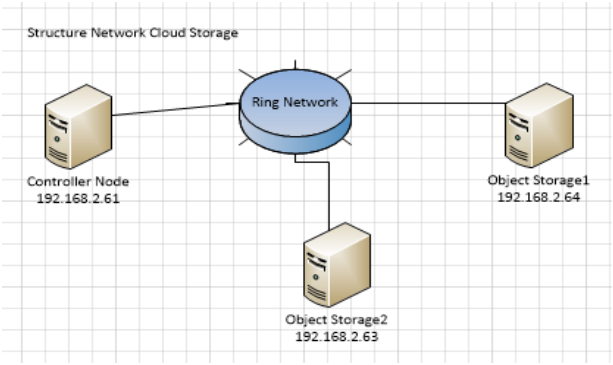

Figure 3. Network Structure

In Figure 3 there are 3 nodes that consists of one controller node and 2 object storage nodes. All nodes interconnected via a ring. Controller node identifies services that running in object storage nodes. Object Storage nodes have accounts, containers, and objects. A ring network serves as unifier of each object storage node connected to controller. The ring builder creates configuration files that each node uses to determine and deploying in storage architecture. In object storages node have account ring, container ring, and object ring. Account ring used to maintain lists of containers, container ring used to maintain lists of objects and object ring used to maintain lists of object locations on local devices. Table 1 shows the function of each node implemented.

Table 1. Specification Nodes

Nodes	IP Address	Functions
Controller	192.168.2.61	To manage services, networking plugin, running message queue, running service SQL database, NTP.
Object Storage 1	192.168.2.64	To run service object storage (Swift) and saving data such as account, container and objects.

2.4. Swift Architecture and Ring Architecture

In Figure 4, it can be seen there are servers, processes and rings. The server provides unified interface for OpenStack Object Storage architecture. It received a request to make the container, uploading files or to modify metadata, and can provide a list of container or the stored files. Object server is a server that can upload, modify, and retrieve objects stored on device successfully. Container server is basically directory object. And then account server can manage the accounts using object storage service established to provide listings.

Processes able managing process storing data, such as replication services, auditors, and updaters. While ring is mapping to physical location then user can read, write storage entity.

In Figure 5 there are 3 rings such as account ring, container ring and object ring to communicate with another nodes. The function of account ring used to maintain list of the containers. And then, container ring serves used to maintain list of objects. Nevertheless, the container does not track the location of the object. And then object ring used to maintain list of the location of objects on the local device. At the node controller configuration is also carried

accounts, containers, and objects to build a ring with another node. On architecture that the account, containers and objects are connected.

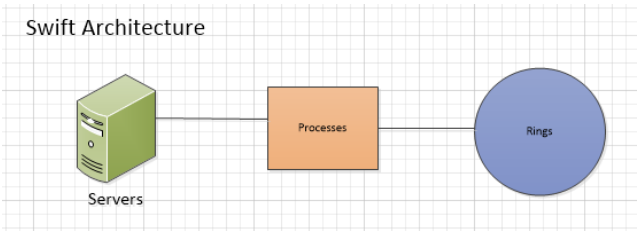

Figure 4. Swift Architecture

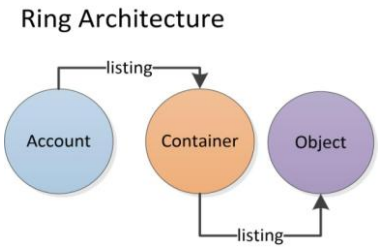

Figure 5. Ring Architecture

2.5. Components

Next, will explain components on OpenStack Object Storage (Swift) are as follows:

Table 2. Components

No	Names of Components	Function of Components
1	Proxy Servers (Swift-Proxy-Server)	The component serves to accept OpenStack Object Storage API and raw HTTP requests to upload files, modify metadata, and create containers.
2	Account Servers (Swift-Account-Server)	Swift-Account-Server use to manage account defined with Object Storage.
3	Container Servers (Swift-Container-Server)	Swift-Container-Server use to manage the mapping of containers or folders within Object Storage.
4	Object Servers (Swift-Object-Server)	The functions of Object Server are manages actual objects, such as files on the storage nodes.
5	WSGI Middleware	The function of WSGI is handles authentication and is usually OpenStack Identify.

The above components are configured on each object storage node and will be connected to controller node through the hoop. Ring is its medium of communication between object storage nodes that have been configured on controller node.

2.6. Implementation OpenStack Object Storage (Swift)

This section will discuss how to build cloud storage using OpenStack Object Storage (Swift) with simple network architecture. Appropriate network structure which has been designed that there are 3 nodes, consist of a controller node and 2 object storage nodes. Before configuration each node, there will discuss about the hardware specifications to support development of this case. The hardware specification present in the Table 3.

In Table 3 illustrates 4 main resources to build cloud storage with simple network architecture. There are different storage capacities on the controller with object storage. For object1, object2 have capacities 2 parts, namely dev/sda capacity of 70 GB and dev/sdb 100 of which 100 GB is used for the storage of objects / files.

Table 3. Hardware Specification

Resources	Controller Node	Object1 Storage Node	Object2 Storage Node
RAM	4 GB	4 GB	4 GB
Storage (HD)	100 GB	170 GB	170 GB
CPU	2	2	2
NIC	1	1	1

Step one to configure basic environment such as Network Time Protocol (NTP) to synchronize times each node. After the completion of the NTP configuration, then configure database (MariaDB) on the controller and configuring messaging server (RabbitMQ) used to coordinate operation status information between services that are running.

After successfully basic environment, next step installation and configuring in controller node such as keystone, object storage service (Swift), install and setup compute nova as a network, install and subsequent configuration in a glance and cinder as auxiliary storage media object in the object storage. For more information command installation in 3 nodes can be seen Table 4.

Table 4. Command Installation

Service Installations	Command Install in terminal	Description
NTP	yum install ntp	Install NTP on all nodes
Databases	yum install mariadb mariadb-server MySQL-python	Install on controller node
Messaging Server	yum install rabbitmq-server	Install on controller node
Keystone	yum install OpenStack-keystone python-keystoneclient	Install on controller node
Nova	yum install OpenStack-nova-api OpenStack-nova-cert OpenStack-novaconductor OpenStack-nova-console OpenStack-nova-novncproxy OpenStack-novascheduler	Install on controller node
Cinder	yum install OpenStack-cinder python-cinderclient python-oslo-db	Install on controller node
Glance	yum install OpenStack-glance python-glanceclient	Install on controller node
Swift in Controller Node	yum install OpenStack-swift-proxy python-swiftclient python-keystone-auth-token python-keystonemiddleware memcached	Install on controller node
Swift in Object1, Object2 Node	yum install OpenStack-swift-account OpenStack-swift-container OpenStack-swift-object	Install on all object node

After finished configuration, next will explain configuration step on each service that has been installed.

1. Controller Node

The first step will be configuration keystone, nova, cinder and configuration glance.

a. Makes keystone database, and then enter a random value used for administrative tokens during the configuration stage.
b. Creating tenants, users and user roles while running service.
c. Create and configure entity and API endpoints in the keystone.
d. And then, can verify operation.
e. Create databases of keystone, nova, cinder and glance. This databases will be store the user name, user role to run services has been configured.
f. Create service credentials for user, roles to running service nova.
g. The last step to verify operation after configuration.

2. Object Storage Nodes

The second step will discusses to configuration swift.

a. Create swift user and then add the admin role for swift user.
b. The next step create swift service entity and create the object storage service API endpoints.

c. If controller node used as proxy server then there will be additional configuration as follows:
   1) Install the package proxy server for swift.
   2) The next getting the proxy service configuration file from the source repository Object Storage.
   3) And the last proxy server configuration as needed.
d. The last step to verify operation after configuration.
e. Before performing verify operations on object storage node configuration, it will be carried out stage to configure the account server, container server and the object server.
f. After all steps finished, then create initial rings. Create an account for initial rings, container, and object rings. For create initial ring by ring builder files that each nodes to deploy the storage architecture.

For detail configuration can be seen from the flow chart in Figure 6.

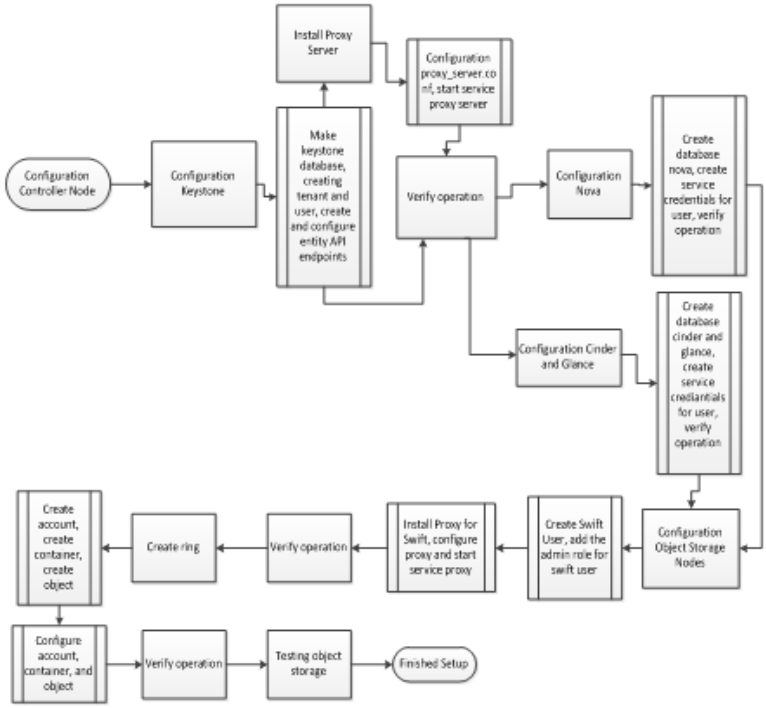

Figure 6. Flow Chart Configuration

Next, will explain to configure Object Storage Nodes. Explanation to perform installation on Object Storage Nodes can be viewed through the following steps:

1. Firstly, for each object storage nodes need to format each drive is provided for the storage media.
2. Furthermore, create the mount point directory structure. And next step insert the drive into the directory /etc/fstab to be recognized by the system.
3. Next step is necessary to mount the device (example: mount /srv/node/sdb1) according to the layout of the configuration directory.
4. Verify operation.

Result create of ring can be seen in the following:

a. Result of create account ring (configure in controller node).

```
[root@controller ~]# swift-ring-builder account.builder
account.builder, build version 2
1024 partitions, 3.000000 replicas, 1 regions, 1 zones, 2 devices, 0.00 balance
The minimum number of hours before a partition can be reassigned is 1
Devices:    id  region  zone      ip address  port  replication ip  replication port      name weight partitions balance meta
             0       1     1    192.168.2.64  6002    192.168.2.64              6002      sdb1 100.00       1536    0.00
             1       1     1    192.168.2.63  6002    192.168.2.63              6002      sdb2 100.00       1536    0.00
```

Figure 7. Result of Create Account Ring

In Figure 7 shown two accounts ring has been create and successfully. These accounts will be used in cloud storage. From the implementation finished, next step to generate awakened a cloud storage that can store data.

3. Results and Analysis

Results and Analysis has been finished, now display form testing in private cloud storage. For testing the cloud storage can use the following command.

Table 5. Command to testing Cloud Storage

No	Command	Description
1	swift stat	Display about information for the account, container, or object
2	swift list	Display list of the containers for the account or the objects for a container
3	swift upload	For upload files/data to the given container
4	swift download	To download data/files/objects from container

For testing command swift stat to view information like that:

```
[root@controller ~]# swift stat
                     Account: AUTH_8abfd0d439ef4f989a908ff3c041e07e
                  Containers: 6
                     Objects: 27
                       Bytes: 107450809
Containers in policy "policy-0": 6
   Objects in policy "policy-0": 27
     Bytes in policy "policy-0": 107450809
    X-Account-Project-Domain-Id: default
                  Connection: keep-alive
                X-Timestamp: 1444637315.38098
                  X-Trans-Id: tx4804d5775b044a52b0d44-005631f2d8
                Content-Type: text/plain; charset=utf-8
               Accept-Ranges: bytes
[root@controller ~]#
```

Figure 8. Swift Stat

```
root@controller:~
[root@controller ~]# swift upload Cyber /home/Test/Drawing1.vsd
/home/Test/Drawing1.vsd
[root@controller ~]# swift upload Cyber /home/Test/Internet\ Sehat.rar /home/Test/Lap-TA-14-TK04.doc
/home/Test/Internet Sehat.rar
/home/Test/Lap-TA-14-TK04.doc
[root@controller ~]#
```

Figure 9. Swift Upload

In Figure 8, seen that there are six container available and 27 objects is stored in containers. For the reliability of these systems will continue to provide services during controller node and the object node of life and do not crash. We can see in connection which explains the system will still keep-alive. To upload in cloud storage object can be seen in more detail through the Figure 9.

From the picture above it can be seen that there objects successfully uploaded into the cloud storage. Next scenario to testing cloud storage using command swift download, for detail can present from Figure 10.

```
[root@controller ~]# swift download Cyber home/Test/Internet\ Sehat.rar home/Test/RAB\ Proposal.pdf home/Test/Drawing
1.vsd home/Test/testing.jpg home/Test/Vazquez\ Sounds\ -\ Let\ It\ Be\ \(Cover\).mp4
home/Test/RAB Proposal.pdf [auth 0.270s, headers 0.430s, total 0.440s, 1.789 MB/s]
home/Test/Drawing1.vsd [auth 0.358s, headers 0.459s, total 0.460s, 0.877 MB/s]
home/Test/testing.jpg [auth 0.383s, headers 0.519s, total 0.519s, 0.598 MB/s]
home/Test/Internet Sehat.rar [auth 0.288s, headers 0.495s, total 0.544s, 6.900 MB/s]
home/Test/Vazquez Sounds - Let It Be (Cover).mp4 [auth 0.326s, headers 0.436s, total 1.270s, 92.172 MB/s]
[root@controller ~]#
```

Figure 10. Swift Download

Objects stored in the cloud storage successfully downloaded. At this stage the user can download multiple objects at the same time and only takes a very short time (i.e. the time required to download 92.172MB sized object is 1,270 s).

And then, the last scenario testing cloud storage using command swift list like that:

```
[root@controller ~]# swift list Cyber
home/Test/Drawing1.vsd
home/Test/Hearthphone.doc
home/Test/Internet Sehat.rar
home/Test/Lap-TA-14-TK04.doc
home/Test/PROGRAM KREATIVITAS MAHASISWA.docx
home/Test/RAB Proposal.pdf
home/Test/Smart Shoes.doc
home/Test/TI.pptx
home/Test/Tong sampah bijak.doc
home/Test/Vazquez Sounds - Let It Be (Cover).mp4
home/Test/testing.jpg
[root@controller ~]#
```

Figure 11. Swift List

From the result of Figure displays list object that has been uploaded to the cloud storage. These files are stored Cyber container which has a directory / home / Test /.

Based on testing performed by uploading, downloading and can object to list and views the status of the scenarios that have been done that have functioning private cloud storage. Users can see the same results through dashboard object storage below.

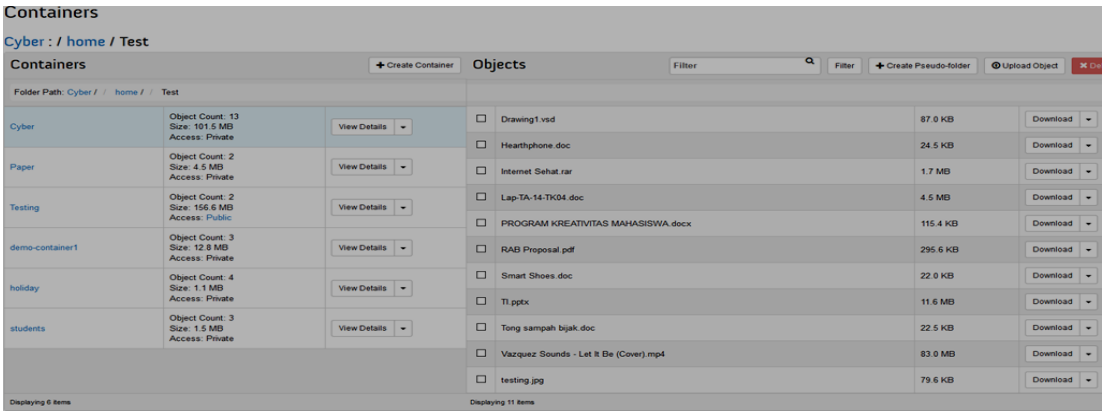

Figure 12. Dashboard Private Cloud Storage

4. Conclusion

Based on research has been done, it can be concluded that we may build a private cloud storage with OpenStack object storage using simple network architecture without thinking about the hassle, costs a lot and redundant hardware resources. This research can be a proof of concept on how to build private cloud storage for their own interests. In the testing step, we test the private cloud in do such function which always exist in the file sharing service, such that, upload and download large files in the form of .GIF .JPEG .doc mp4 file. We got the result from testing required to download 92.172MB sized object is 1,270 s for file mp4

References

[1] LS Girish, Dr Guruprasad HS. Building Private Cloud using OpenStack. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*. 2014.
[2] P Mell, T Grance. The NIST Definition of Cloud Computing. National Institute of Standards and Technology. USA. 2009
[3] Sonali Yadav. Comparative Study on Open Source Software for Cloud Computing Platform: Eucalyptus, OpenStack and Opennebula. *International Journal of Engineering and Science.* 2013; 3(10): 51-54.
[4] D Lakshmi Kurup, Chandawalla Chandni, Parekh Zalak, Sampat Kunjita. Comparative Study of Eucalyptus, OpenStack, and Nimbus. *International Jurnal of Soft Computing and Engineering (IJSCE)*. 2015; 4(6).
[5] Sefraoui Omar, Aissaoui Mohammed, Eleuldj Mohsine Eleuldj. OpenStack: Toward an Open-Source Solution for Cloud Computing. *International Journal of Computer Application*. 2012; 55(03).
[6] Sachdev Sidharth, Mahajan Amit. Deployment of Private Cloud using OpenStack: An Open Source Cloud Computing Solution for Small and Mid-sized Organizations. *International Journal of Advances Research in Computer Science and Software Engineering.* 2013; 3(10).
[7] Peter Sempolinski, Douglas Thain. *A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus.* 2010 IEEE Second International Conference on Cloud Computing Technology and Science. 2010: 417-426.
[8] Atre Aniruddha, Lalingkar Prasanna, Rao Amit, Shingre Puskaraj. SDK To Ease Access of Openstak Storage Swift Cloud Services. *International Journal of Scientific & Technology Research*. 2014; 3(9).
[9] Project Modul OpenStack Storage with Drupal. https://www.drupal.org/project/OpenStack_storage.
[10] Project Swift on File. https://github.com/stackforge/swiftonfile.
[11] OpenStack Foundation All Right. OpenStack Juno Installation Guide for Red Hat Enterprise Linux 7, Centos 7, and Fedora 20. 2015.
[12] Deploying-OpenStack-Object-Storage-Swift. http://www.slideshare.net/reidrac/deploying-OpenStack-object-storage-swift.
[13] Anggeriana Herwin. Pengembangan Elemen Cloud Computing dalam Sistem Teknologi Informasi. *Journal of Information System & Technology*. 2011.
[14] Samer Al-Kiswany, Dinesh Subhraveti, Prasenjit Sarkar, *Matei Ripeanu. VMFlock: Virtual Machine Co-Migration for the Cloud.* 20th International Symposium on high performance distributed computing. New York, USA. 2011: 159-170.
[15] Ken Pepple. Deploying OpenStack. First Edition. 2011.
[16] OpenStack Object Storage. http://www.OpenStack.org/software/OpenStack-Storage). 2015