# IMPLEMENTATION AND PERFORMANCE ANALYSIS OF PRIVATE CLOUD USING OPENSTACK SWIFT DAN RCLONE

**Candra Aditama[1], Adri Priadana[2]**
Program Studi Teknik Informatika, Fakultas Teknik dan Teknologi Informasi
Universitas Jenderal Achmad Yani Yogyakarta
Jl. Siliwangi, Ringroad Barat, Banyuraden, Gamping, Sleman
Daerah Istimewa Yogyakarta
Email : [1]can.aditama@gmail.com, [2]adripriadana3202@gmail.com

**Abstract**

*Many companies need a storage system that can be accessed in real time by all parts of the company. Most digital data storage methods today still use conventional methods where data is stored on an external hard disk or public cloud. Storage with external hard disk media makes accessing data difficult and has the risk of data loss when storage media is damaged. On the other hand, the storage method using public cloud requires an internet connection with large bandwidth requirements and the company still has to spend a budget on renting it. This study aims to design digital data storage methods using a private cloud that can be accessed in real time by all parts of the company without having to spend a budget on hiring a storage media and renting an internet connection with large bandwidth requirements. A private cloud was built using OpenStack Swift as an object storage service provider and Rclone as a cross-platform computer data management application. The results of this study are the creation of a private cloud that runs object storage services using Swift storage objects with relatively light and high scalability to meet the needs of storing data effectively and efficiently. A private cloud-based data storage media with relatively light and high scalability to meet data storage needs. Data storage media that can be accessed easily without having to use an internet connection with large bandwidth requirements.*

***Keywords:*** *cloud computing, private cloud, Openstack, object storage, Swift, Rclone*

## 1. Latar Belakang

Kemudahaan dalam mengakses data saat ini menjadi salah satu hal yang sangat penting. Banyak instansi perusahaan yang membutuhkan sistem penyimpanan yang dapat diakses secara *real time* oleh seluruh bagian pada instansi perusahaan tersebut. Metode peyimpanan data digital saat ini kebanyakan masih menggunakan cara konvensional dimana data disimpan pada *external hard disk* atau *public cloud*. Penyimpanan menggunakan media *external hard disk* memiliki resiko data hilang ketika media penyimpanan rusak. Selain itu, penyimpanan menggunakan media *external hard disk* juga mengakibatkan sulitnya pengaksesan data. Di sisi lain, metode penyimpanan dengan menggunakan *public cloud* untuk menyimpan data digital dalam jumlah besar mebutuhkan koneksi internet dengan kebutuhan *bandwidth* yang besar pula. Hal ini mengakibatkan instansi perusahaan harus mengeluarkan anggaran yang tidak sedikit untuk berlangganan koneksi internet dengan kebutuhan *bandwidth* yang besar. Selain itu, instansi perusahaan masih harus mengeluarkan anggaran untuk menyewa media penyimanan menggunakan *public cloud* yang memiliki kapasitas penyimanan yang besar. Oleh karena itu,

dibutuhkan metode penyimpanan data digital yang dapat diakses secara *real time* oleh seluruh bagian pada instansi perusahaan tanpa harus mengeluarkan anggaran untuk menyewa media penyimanan dan menyewa koneksi internet dengan kebutuhan *bandwidth* yang besar.

Untuk mengatasi permasalahan tersebut, dirancang *private cloud* menggunakan OpenStack dengan memanfaatkan layanan *object storage* Swift sebagai *backend* dan Aplikasi Rclone sebagai *frontend*. OpenStack telah banyak dimanfaatkan untuk membuat *private cloud* [1]–[5]. Layanan *object storage* memungkinkan data yang disimpan pada *folder* komputer pengguna otomatis akan disinkronisasi dan di-*backup* ke file *server*. *Server* terdiri dari beberapa *node* komputer atau media penyimpanan, yang dapat bekerja bersamaan untuk melakukan replikasi data dan saling menggantikan jika terjadi permasalahan di salah satu node. *Private cloud* dipilih karena berjalan dibawah *firewall* sehingga keamanan lebih terjamin dan tidak membutuhkan *bandwidth* internet, cukup menggunakan *bandwidth* jaringan lokal. Dibandingkan dengan penyimpanan manual menggunakan *external hard disk*, data yang disimpan pada *private cloud* dapat dengan mudah diakses secara *online* saat dibutuhkan.

## 2. Metodologi Penelitian

Penelitian ini dilaksanakan dalam tiga tahapan. Tahapan pertama dalam dimulai dengan mempelajari layanan yang disediakan OpenStack dan arsitektur OpenStack. Tahapan kedua membangun infrasruktur jaringan komputer, arsitektur Swift, arsitektur Ring, serta arsitektur komponen pendukung OpenStack. Tahapan terakhir melakukan analisa performa terhadap hasil implementasi Swift. Adapun metode penelitian yang digunakan pada penelitian ini adalah metode "*The PPDIOO Network Lifecycle*" yang dikembangkan oleh Cisco dengan pendekatan siklus hidup jaringan yang mampu menjaga desain proses tetap terorganisir.

### 2.1. OpenStack

OpenStack merupakan perangkat lunak *open source* yang digunakan untuk membangun platform komputasi awan, baik publik maupun privat. Terdiri dari gabungan komponen - komponen yang saling berinteraksi untuk mengontrol sumber daya komputasi, penyimpanan data, dan jaringan. OpenStack biasanya diimplementasikan untuk menyediakan *platform infrastructure-as-a-service* (IaaS) [6]. OpenStack merupakan penyedia layanan object storage dapat diimplementasikan tanpa kerumitan, biaya, dan kebutuhan perangkat keras yang berlebihan [7]. Detail infrastruktur layanan OpenStack dapat dilihat pada Gambar 1.

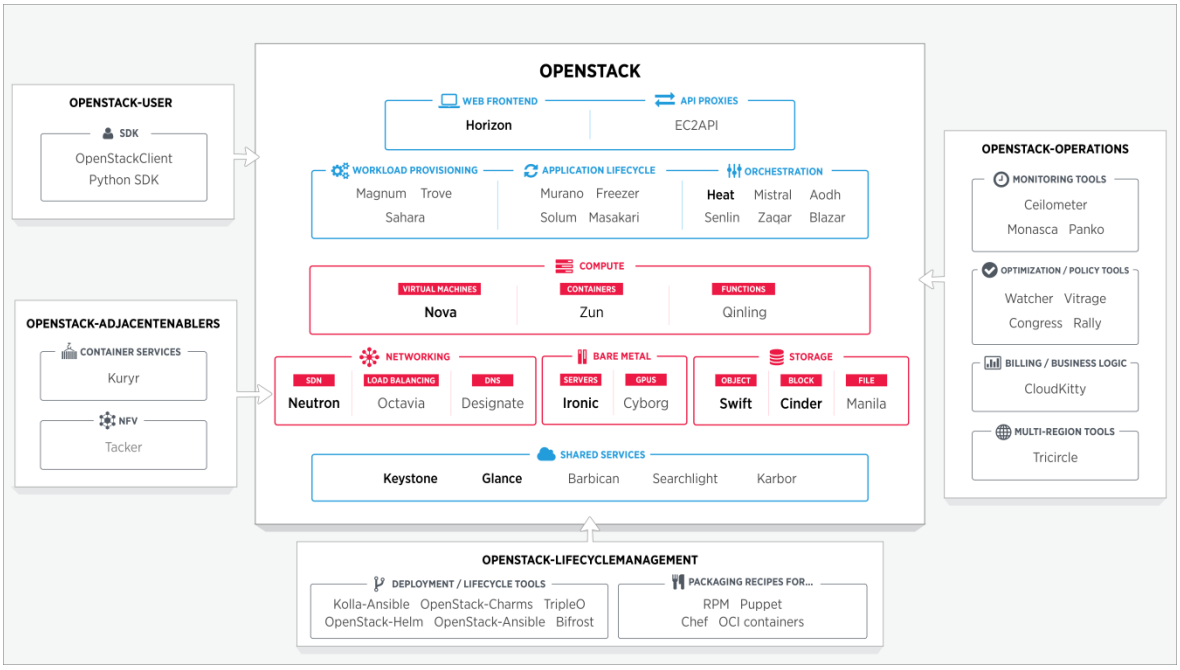
### 2.2. Object Storage Swift

*Object storage* Swift merupakan salah satu komponen dari OpenStack yang menyediakan layanan penyimpanan data via RESTful, API berbasis HTTP. Didesain untuk menyimpan data tidak terstruktur dalam skala yang besar. Digunakan untuk membangun layanan penyimpanan data dalam jumlah besar dengan kemampuan redudansi data, *high availability*, dan *scalable*. Swift berfokus pada penyimpanan objek, yang terintegrasi dengan baik dengan OpenStack dan divalidasi oleh praktik produksi massal sebagai salah satu komponen OpenStack [8]. Arsitektur Swift terlihat pada Gambar 2.

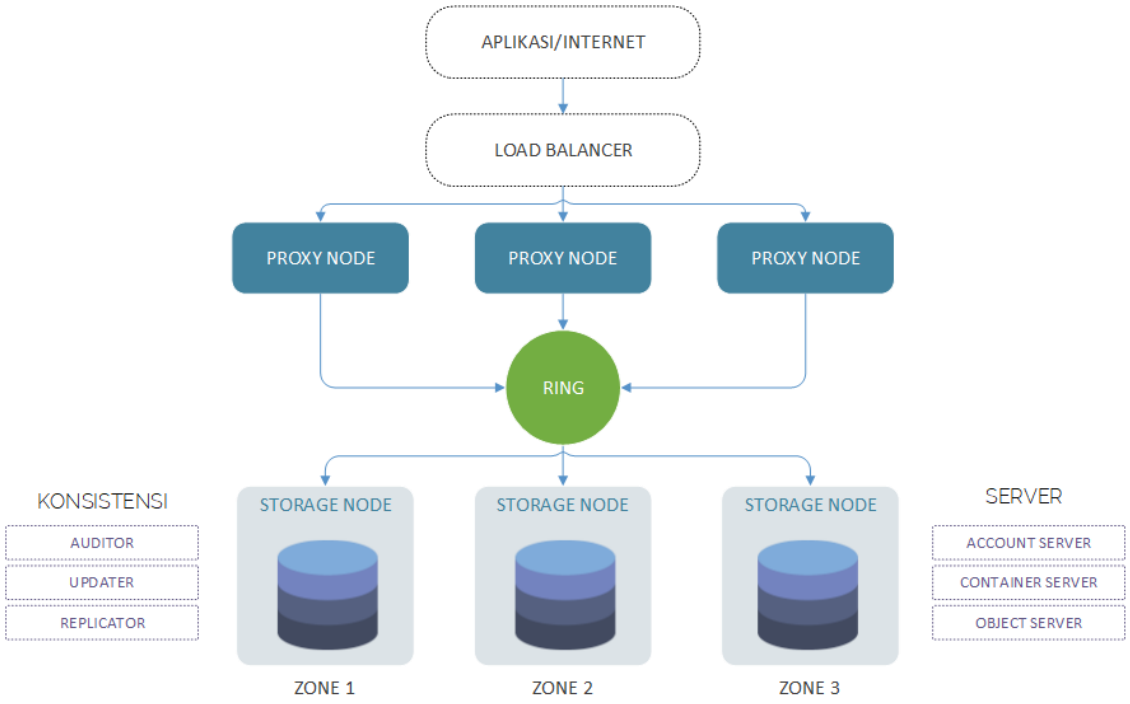
### 2.3. Rclone

Rclone merupakan salah satu perangkat lunak gratis dan *open source* yang digunakan untuk memindah, menyalin, dan sinkronisasi data lintas *platform* komputer. Rclone memiliki dukungan

yang tinggi terhadap banyak layanan *cloud storage*, salah satunya OpenStack Swift. Rclone dibuat dengan menggunakan bahasa Python dan tersedia di hampir semua *platform* sistem operasi. Secara bawaan Rclone tidak memiliki antar muka berbasis grafis, namun karena dukungan *open source* ada beberapa versi pengembangan yang memiliki antar muka untuk memudahkan pengguna dalam memanajemen berkas pada komputer untuk di integrasikan dengan komputasi awan.



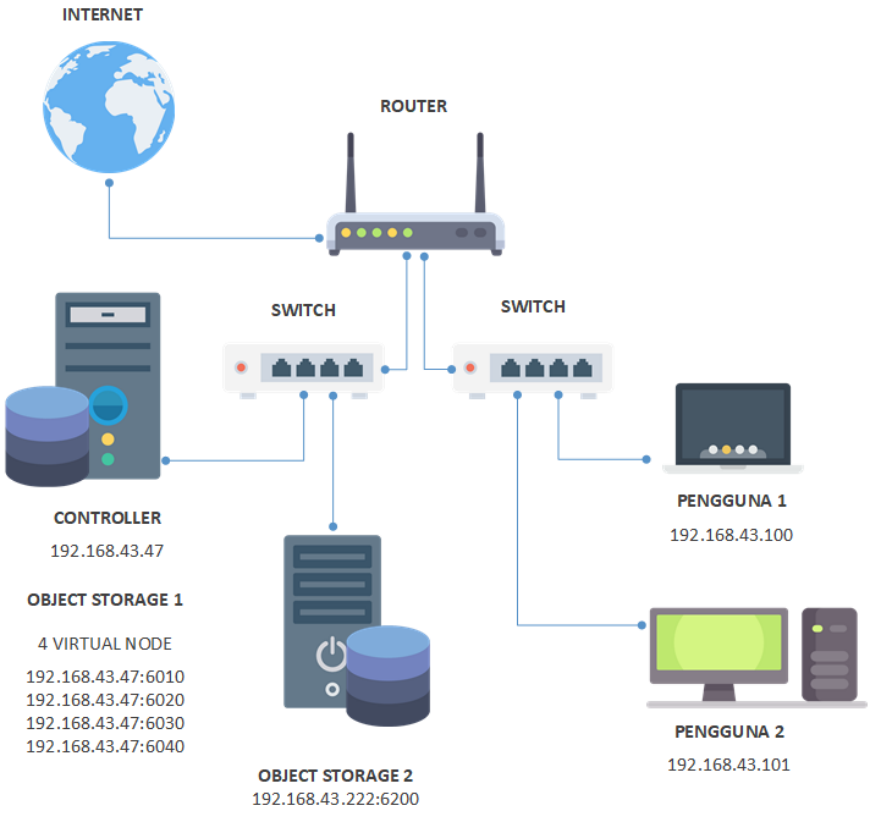

Gambar 1 Arsitektur OpenStack




Gambar 2 Arsitektur OpenStack Swift

## 2.4. Topologi Jaringan

*Private cloud* dibangun dengan jaringan sederhana menggunakan topologi jaringan star dan dalam satu segmen IP Address. Jaringan *private cloud* berada dibawah *router* dan *firewall* terdiri dari satu perangkat *controller*, 4 *node virtual*, dan satu *node* fisik berupa laptop. Empat node virtual merupakan partisi hardisk pada komputer controller yang dijadikan *loopback device*, dimana partisi atau *folder* direpresentasikan seolah – olah merupakan media penyimpanan dari komputer lain. Topologi dapat dilihat pada Gambar 3.




Gambar 3 Topologi jaringan OpenStack Swift

## 3. Hasil dan Pembahasan

Dalam penelitian ini berhasil diimplementasikan layanan *object storage* menggunakan OpenStack Swift dengan memanfaatkan satu perangkat komputer sebagai *controller*, empat *virtual node*, dan satu komputer sebagai *node* fisik. OpenStack Swift berhasil menyediakan layanan penyimpanan berkas yang efektif dan efisien meskipun hanya menggunakan sumber daya komputer dan infrastruktur jaringan yang cukup sederhana.

Object storage Swift dibangun dengan Controller memiliki empat *virtual node* berupa loopback device, dan satu *node* fisik berupa komputer Object Storage 2. *Virtual node* berupa *folder* dengan *filesystem Extens File System* (XFS) yang direpresentasikan seolah - olah merupakan perangkat terpisah. Sedangkan komputer Object Storage 2 juga menjalankan layanan OpenStack Swift dan menjadi *node* tambahan bagi Controller. Detail spesifikasi perangkat keras dapat dilihat pada Tabel 1

Tabel 1 Spesifikasi perangkat

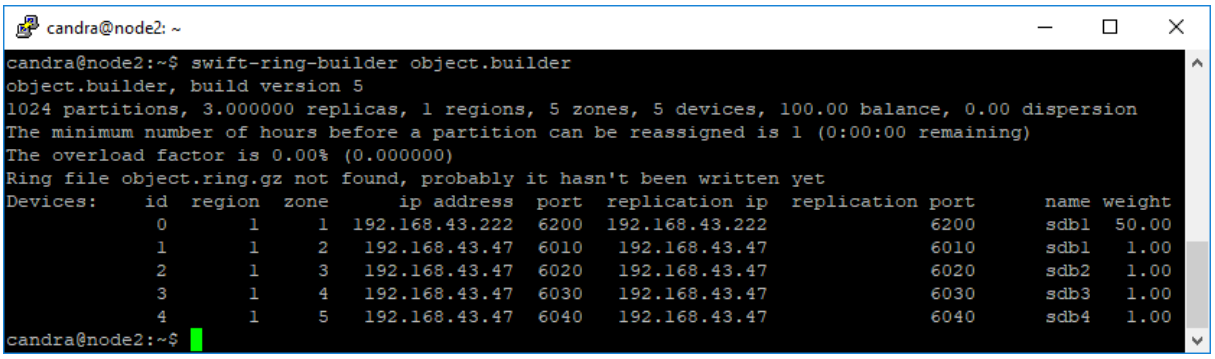
| **Sumber Daya** | **Controller** | **Virtual Node 1 - 4** | **Object Storage 2** |
|---|---|---|---|
| **Processor** | Intel i5 7th Generation | Intel i5 7th Generation | Intel i3 7th Generation |
| **RAM** | 4GB | 4GB | 2GB |
| **HDD** | 500 GB | 10 GB | 500 GB |
| **NIC** | 1 | 1 | 1 |

Tahapan instalasi *object storage* Swift diawali dengan membangun jaringan komputer. Komputer server menggunakan sistem operasi Ubuntu dan komputer pengguna menggunakan Windows 10. Setelah lingkungan dasar sistem *object storage* terbangun, selanjutnya melakukan instalasi dan konfigurasi Swift. Adapun detail penjabaran konfigurasi adalah sebagai berikut:

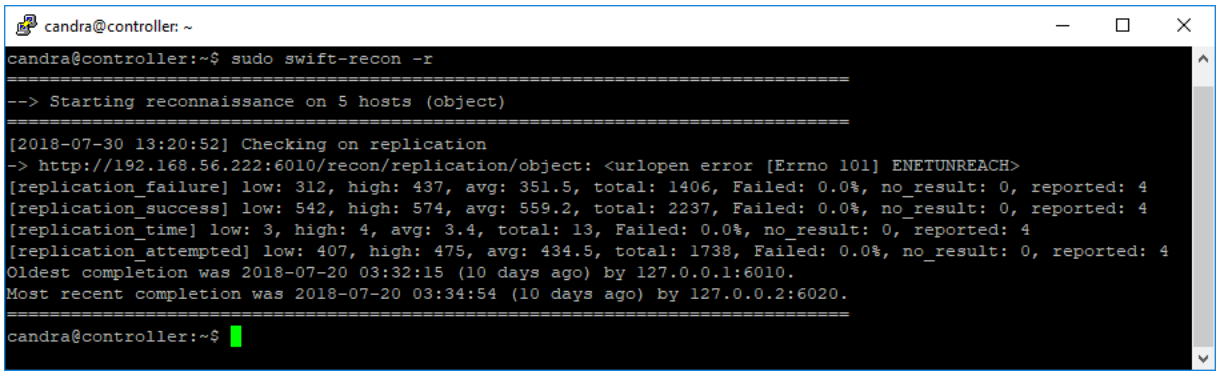
1. Controller
    a. Membuat akun pengguna dan peran pengguna tersebut.
    b. Konfigurasi dan inisiasi Ring.
    c. Menambah Ip Address masing – masing node ke dalam Ring.
    d. Menjalankan layanan Rsyncd dan memcached.
    e. Verifikasi konfigurasi.
2. Node
    a. Konfigurasi *bind-ip* dengan *ip address* Controller.
    b. Alokasi *hard disk* sebagai media penyimpanan.
    c. Menjalankan layanan Rsyncd dan memcached.
    d. Verifikasi konfigurasi.
3. Virtual Node
    a. Membuat folder dengan *filesystem* XFS.
    b. *Mounting* folder sebagai *virtual node*.
    c. Menjalankan layanan Rsyncd dan memcached.

Informasi *node* yang terdaftar pada Ring dapat dilihat ada Gambar 4.

Node yang telah terdaftar dalam Ring otomatis akan melakukan sinkronisasi dan replikasi data kemudian saling menggantikan jika ada salah satu node yang mengalami permasalahan. Gambar 5 menunjukkan laporan replikasi data yang telah terjadi.

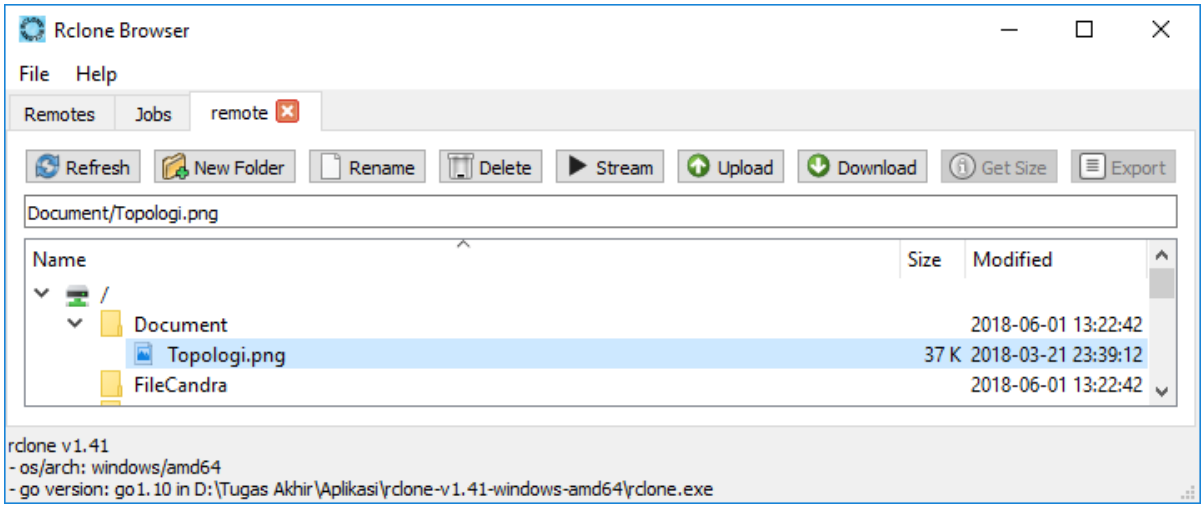



Gambar 4 Hasil konfigurasi Ring



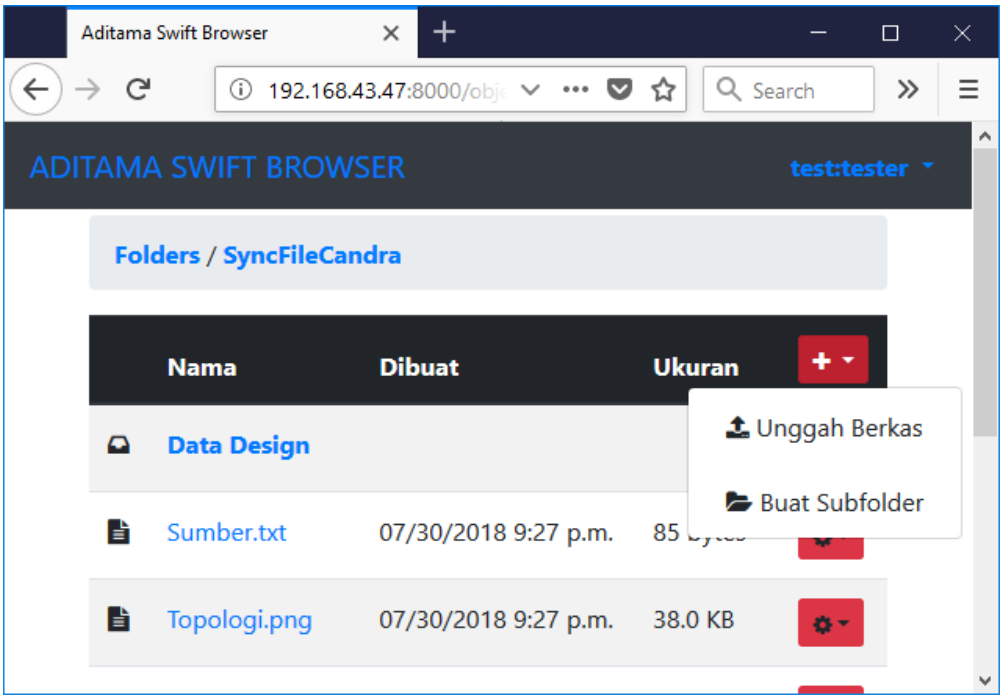

Gambar 5 Informasi replikasi yang telah berjalan

Pengguna dapat berinteraksi dengan object storage Swift dengan menggunakan aplikasi Rclone. Pengguna dapat dengan mudah mengunggah berkas ke *server* atau mengunduh dokumen secara grafis. Gambar 6 menampilkan antarmuka Rclone.



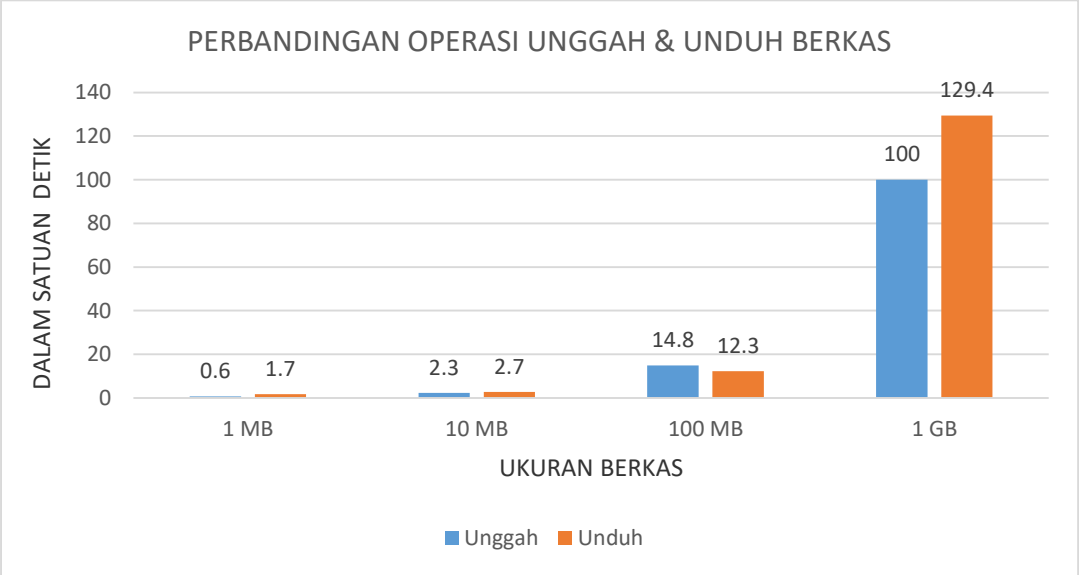

Gambar 6 Antarmuka Rclone Browser

Bagi pengguna yang akan mengakses layanan object storage via *web browser* atau *smartphone*, dibangun antar muka berbasis *web* dengan *framework* Django. Gambar 7 menampilkan antarmuka layanan Swift berbasis *web*.



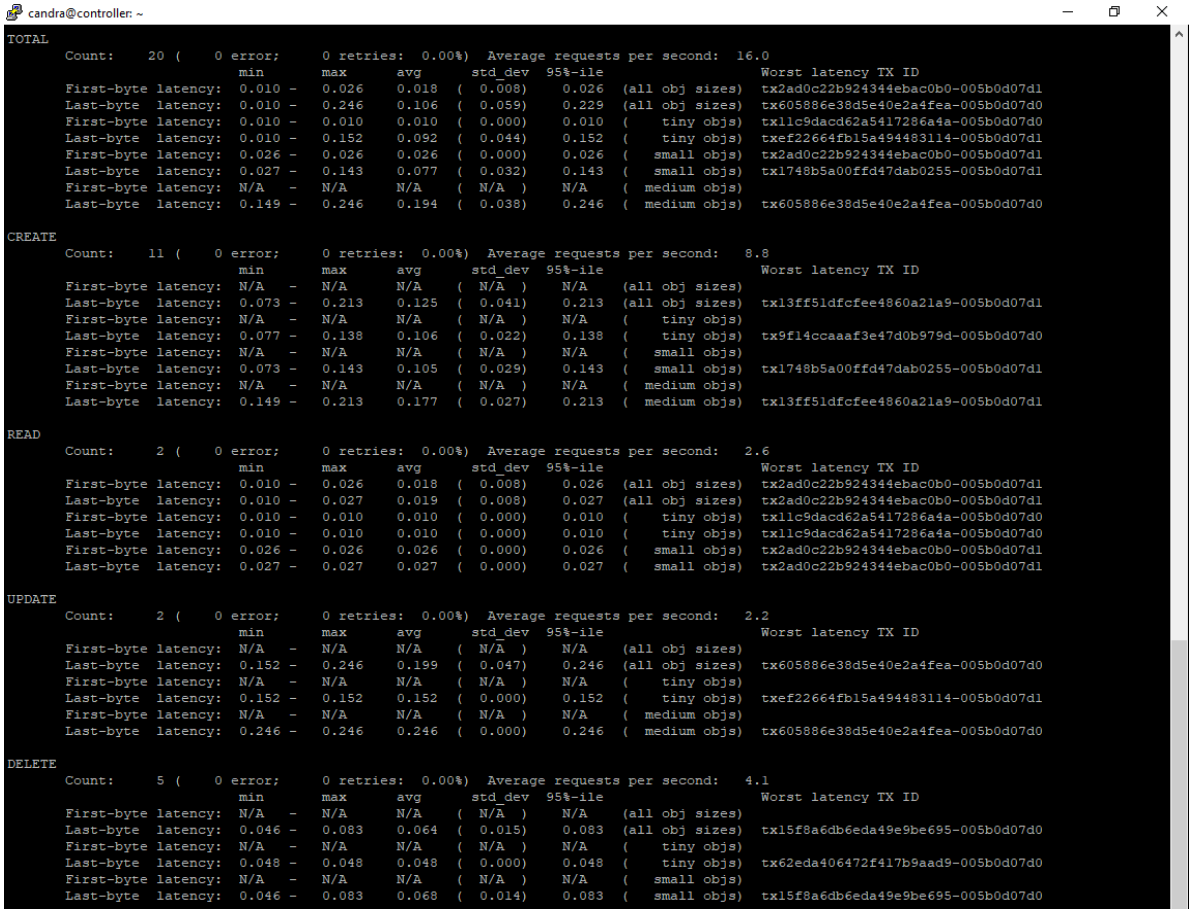

Gambar 7 Antarmuka Pengelolaan Berkas Berbasis Web

Tahap selanjutnya adalah melakukan pengujian untuk mengetahui kemampuan layanan *object storage* Swift dilakukan beberapa pengujian yang pertama perfoma unggah dan unduh berkas dalam ukuran 1 MB, 10 MB, 100 MB, dan 1 GB diunggah dari komputer pengguna ke Server OpenStack Swift. Adapun hasil pengujian dapat dilihat pada Gambar 8.



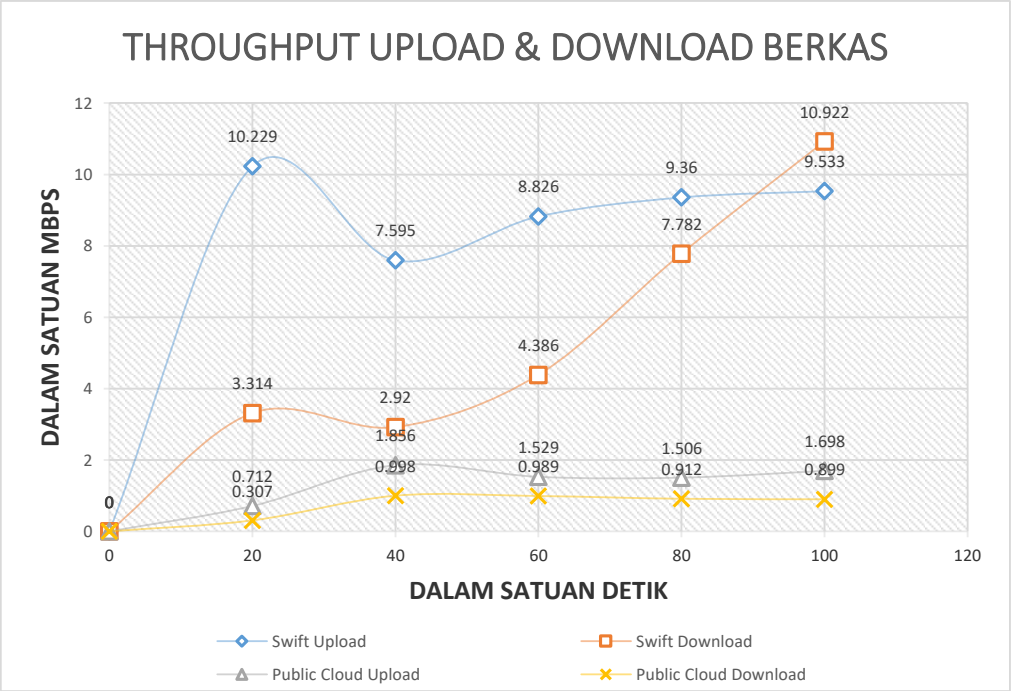

Gambar 8 Hasil Pengujian Unggah dan Unduh Berkas

Pengujian dapat dilakukan secara otomatis dengan menggukana Ssbench dengan mensimulasikan operasi CRUD dengan ukuran berkas berdasarkan skenario tertentu. Pada penelitian ini menggunakan skenario *small test* dengan jumlah operasi 100 kali. Gambar 9 menunjukkan hasil uji performa ssbench dengan ukuran skenario *small test*.




Gambar 9 Hasil Pengujian Ssbench

Dari pengujian diatas, tidak terdapat error operasi dan *latency byte* pertama masih dibawah 1 detik. Selanjutnya dilakukan pengujian perbandingan performa private cloud Swift dengan public cloud. Gambar 10 menunjukkan perbandingan hasil pengujian.

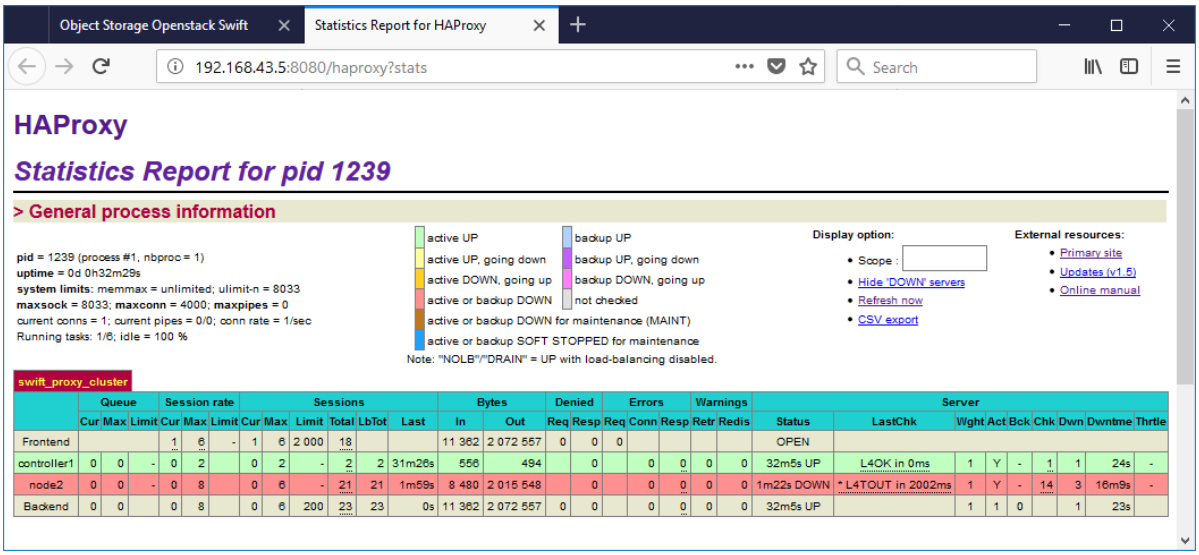



Gambar 10 Perbandingan *Thoughput Public Cloud* dan *Private Cloud*

Sedangkan untuk memantau performa Swift, dapat dengan menjalankan layanan StatsD sebagai penyedia *metric* dikombinasikan dengan Graphite sebagai *collector*. Gambar menunjukkan tampilan grafik performa Swift. Tampilan grafis Graphite dapat dilihat pada Gambar 11.


Gambar 11 *Monitoring* Swift Menggunakan Graphite

Untuk menjalankan layanan yang *high availability* pada OpenStack Swift, dapat dijalankan dengan menggunakan komponen HAProxy (*High Availability Proxy*). Cara kerja HAProxy adalah dengan menjalankan *floating IP* yang mewakili beberapa *IP Address* sekaligus. Pengguna hanya perlu mengakses layanan berdasarkan satu *floating IP*, namun dibelakang *floating IP* terdiri dari beberapa *server* yang akan melayani permintaan pengguna sesuai dengan *server* yang aktif, atau berdasarkan prioritas tertentu. Gambar 12 menunjukkan Object Sorage 2 sedang *down*, namun layanan *object storage* tetap berjalan.




Gambar 12 Object Storage 2 Terpantau sedang Mati/*Down*

## 4. Kesimpulan

Dari hasil perancangan, implementasi, pengujian dan analisa yang telah dilakukan terhadap object storage OpenStack Swift, dapat disimpulkan bahwa,

1. *Object storage* OpenStack Swift dapat diimplementasikan dengan baik menggunakan sumber daya yang tidak terlalu tinggi.
2. *Object storage* Swift dapat digunakan untuk menyediakan layanan penyimpanan data yang besar dan *scalable* yang ukuran media penyimpanannya dapat ditambah tanpa mengubah konfigurasi dan infrastruktur jaringan yang sudah ada.
3. Performa OpenStack Swift dapat diuji secara otomatis menggunakan Ssbench atau diuji dengan cara menghitung dan mencatat parameter yang muncul pada saat menjalankan fitur OpenStack Swift.
4. Untuk memudahkan pengguna dalam memanfaatkan layanan OpenStack Swift, pengguna dapat menggunakan antarmuka berbasis GUI diantaranya perangkat lunak Rclone untuk pengguna desktop, Syncany untuk sinkronisasi berkas pada komputer pengguna ke server, SME Swift Xplorer untuk pengguna berbasis mobile, dan Swift Browser untuk antarmuka berbasis Web.
5. Swift memiliki fitur replikasi sebagai mekanisme *disaster recovery* dan HAProxy sebagai penyedia layanan yang *high availability*.

**Daftar Pustaka**

[1] Chi, Y., Li, G., Chen, Y., & Fan, X. (2018, July). Design and Implementation of OpenStack Cloud Platform Identity Management Scheme. In 2018 International Conference on Computer, Information and Telecommunication Systems (CITS) (pp. 1-5). IEEE.

[2] Surateno, S., Jullev, E. S., & Pratama, D. Y. (2018). Implementation Private Cloud Computing Using Openstack With The Use Of KVM Hypervisor And Glusterfs. In Proceeding of the 1st International Conference on Food and Agriculture.

[3] Bhatia, G., Al Noutaki, I., Al Ruzeiqi, S., & Al Maskari, J. (2018, March). Design and implementation of private cloud for higher education using OpenStack. In 2018 Majan International Conference (MIC) (pp. 1-6). IEEE.

[4] Mangal, G., Kasliwal, P., Deshpande, U., Kurhekar, M., & Chafle, G. (2015, November). Flexible cloud computing by integrating public-private clouds using openstack. In 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM) (pp. 146-152). IEEE.

[5] Gaikwad, C., Churi, B., Patil, K., & Tatwadarshi, P. N. (2017, March). Providing storage as a service on cloud using OpenStack. In 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS) (pp. 1-4). IEEE.

[6] Sumunar, R. A. G. (2014). Analisis Fungsi Compute pada Private Cloud Computing dengan Model Layanan Infrastructure as a Service Menggunakan OpenStack (Doctoral dissertation, Program Studi Teknik Informatika FTI-UKSW).

[7] Sagala, A., & Hutabrat, R. (2016). Private Cloud Storage Using OpenStack with Simple Network Architecture. Indonesian Journal of Electrical Engineering and Computer Science, 4(1), 155-164.

[8] Kong, W., & Luo, Y. (2016, May). Multi-level image software assembly technology based on OpenStack and Ceph. In 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference (pp. 307-310). IEEE.