IMPLEMENTASI DAN ANALISA PERFORMANSI PLATFORM KEYSTONE DALAM SERVICE IDENTITY BERBASIS OPENSTACK

IMPLEMENTATION AND PERFORMANCE ANALYSIS PLATFORM KEYSTONE IN IDENTTITY SERVICE BASED ON OPENSTACK

Sandi Purnama[1], Tody Ariefianto Wibowo, ST.,MT.[2], Joko Purnomo, ST.[3]

[1,2]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
[3]Senior Engineering, Ericsson Indonesia
[1]sandipurnamma@gmail.com, [2]tody.wibowo@gmail.com, [3]joko.purnomo@ericsson.com

Abstrak

OpenStack merupakan suatu sistem operasi SDN (Software Defined Network) yang mengontrol lingkungan komputasi besar, storage, dan sumber daya jaringan di seluruh datacenter. Semua dikelola melalui dashboard yang memberikan kontrol kepada administrator. Sistem ini merupakan perangkat lunak OpenSource yang dirilis di bawah ketentuan Lisensi Apache. OpenStack memiliki arsitektur modular yang terdiri dari beberapa bagian blok yaitu Application, Dashboard, Compute, Storage, dan Network. Pada tugas akhir kali ini hanya akan dibuat service identity pada OpenStack yang berfungsi sebagai penyedia keamanan dalam terselenggaranya pertukaran informasi antar *service* yang ada. Identity menggunakan platform keystone.
Dari hasil perancangan service storage pada tugas akhir ini dapat diambil kesimpulan bahwa kinerja platform dapat dikatakan berjalan dengan baik. Hal ini dikarenakan setiap permintaan user dapat terlayani dengan baik dari mulai pembuatan user baru, pembuatan instance, dan letak endpoint yang terlihat di sisi client. Dengan adanya sistem OpenStack ini maka sistem cloud computing akan semakin menjadi efektif dan efisien

Kata kunci : *OpenStack*, SDN, *Icehouse*, *Identity*, *Keystone*

Abstract

OpenStack is an operating system SDN (Software Defined Network) that control large computing environments, storage, and network resources across the datacenter. All managed through a dashboard that gives control to administrators. This system is an OpenSource software that is released under the terms of the Apache License. OpenStack has a modular architecture that consists of several parts of blocks that Application, Dashboard, Compute, Storage, and Network. At the end of this time the task will only be made on OpenStack Identity service that serves as a provider of security. Identity using platforms, namely Keystone. From the results of the design service identity in this thesis can be concluded that the performance of platform can be quite stable. This is because each user request can be served well from start of creating new user, creating new instance, and position of endpoint that client can see. With this OpenStack systems, cloud computing systems will be more effective and efficient.

Keywords: OpenStack, SDN, *Icehouse*, *Identity*, *Keystone*

1. Pendahuluan

Pada era teknologi seperti sekarang ini, perkembangan penggunaan jaringan komputer berbasis IP (Internet Protokol) semakin meningkat. Dalam perancangan jaringan dengan teknologi lama, terdapat beberapa perangkat jaringan seperti router dan switch. Jika kuantitas perangkat tersebut sangat banyak maka akan dirasa belum efektif. Karena saat ini perusahaan IT berkembang dengan pesat sehingga kebutuhan perangkat jaringan semakin besar pula. Dengan begitu perusahaan membutuhkan kapasitas resource yang besar dan kemampuan untuk mengatur banyak perangkat jaringan secara bersamaan. Sudah dapat dipastikan akan dibutuhkan waktu yang lama untuk konfigurasi perangkat-perangkat tersebut. Dari hal tersebut maka diperlukan suatu sistem untuk membuat suatu metode pendekatan pembangunan jaringan efisien yang dikenal dengan OpenStack yang merupakan sistem Software Defined Network (SDN) yang menggunakan protokol OpenFlow. OpenStack yang merupakan sebuah environment yang dapat menyatukan beberapa service yang seperti virtualization, compute, storage, identify, network dalam satu controller atau pengendali. Sistem ini menggunakan Linux sebagai sistem operasi. OpenStack dibangun dari beberapa paltform seperti Nova, Neutron, Swift, Cinder, Glance, Keystone, dan Telemetry.

2. Dasar Teori

2.1 Cloud Computing

Secara umum, definisi *cloud computing* (komputasi awan) merupakan gabungan pemanfaatan teknologi komputer (komputasi) dalam suatu jaringan dengan pengembangan berbasis internet (awan) yang mempunyai fungsi untuk menjalankan program atau aplikasi melalui komputer - komputer yang terkoneksi pada waktu yang sama, tetapi tak semua yang terkoneksi melalui internet menggunakan *cloud computing*.

2.2 SDN (Software Defined Network)

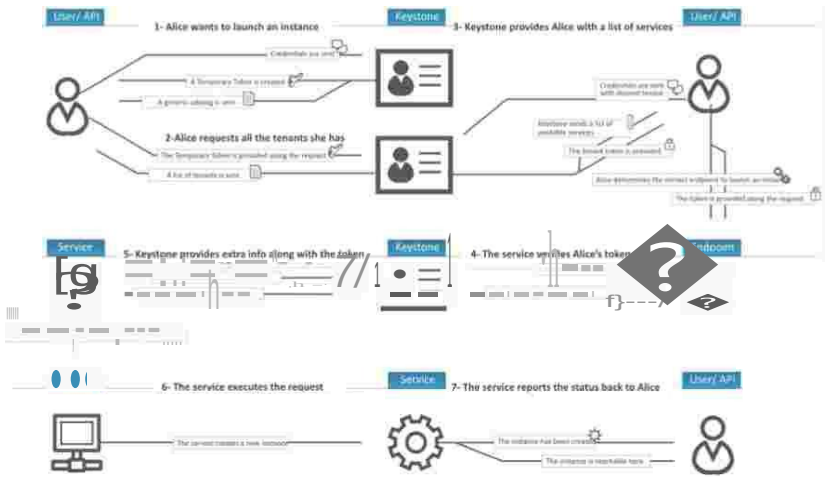
Software Defined Network merupakan pendekatan baru untuk merancang, membangun dan mengelola jaringan. SDN secara harfiah dapat dikatakan sebagai suatu sistem jaringan yang kinerjanya diatur oleh software tertentu. Ide mengenai SDN terus berkembang hingga tahun 2011 mucul organisasi *OpenFlow Network Foundation* yang dirintis oleh berbagai perusahaan di dunia. Beberapa perusahaan yang terkait antara lain *Google, Yahoo, dan NTT.*

2.3 OpenStack

OpenStack adalah sebuah platform awan yang terdiri dari software-software bebas dan *open source* untuk menyediakan layanan cloud IaaS (Infrastructure as a Service), baik pribadi maupun skala besar yaitu berupa sumber daya untuk komputasi dan penyimpanan data dalam bentuk mesin virtual. Pegembangan platform awan ini dikelola oleh *OpenStack* Foundation yang beranggotakan perusahan-perusahan TI terkemuka seperti AMD, AT&T, Canonical, Cisco, Dell, HP, Intel, Red Hat, Suse, Deutsche Telekom, VMware, dan tentunya juga termasuk IBM. *OpenStack* merupakan kerangka kerja *open source* penyedia infrastuktur komputasi awan.

2.4 OpenStack Identity Service (keystone)

OpenStack Identity Service merupakan salah satu *service* dari *OpenStack* yang berfungsi untuk menyediakan otentikasi dan otorisasi dalam penyelenggaraan pertukaran data antar *service* yang ada di dalam ruang lingkup *OpenStack*. *OpenStack Identity Service* juga dapat dikatakan layanan yang memberikan fungsi keamanan selama proses pertukaran informasi antar *service* yang terjadi pada *OpenStack*. Otorisasi melakukan verifikasi apakah pengguna memiliki akses ke layanan yang diminta.



Gambar 2.1 *Keystone Identity Service Process*

Dalam penerapannya, *OpenStack Identity service* menggunakan *platform* yang disebut *keystone. Keystone* menyediakan dua cara otentikasi yaitu dengan *username/password* dan dengan menggunakan *token*. *Identity Service* memiliki dua fungsi utama yaitu :

1. *User Management* : memantau histori *user* dan hak akses mereka
2. *Service Catalog* : menyediakan pilihan mengenai *service* yang tersedia dan dimana API endpoints mereka berada.

Keystone juga terdiri dari beberapa grup layanan internal pada *OpenStack* yang terhubung pada satu atau banyak endpoint. Layanan tersebut diantaranya :

1. Token Service : membawa informasi mengenai otorisasi pengguna yang sudah terotentikasi
2. Catalog Service : berisi daftar *service* yang tersedia pada *user*
3. Policy Service : berfungsi untuk mengelola akses ke *service* tertentu oleh *user* atau kelompok tertentu

Komponen utama dari identity service adalah sebagai berikut :

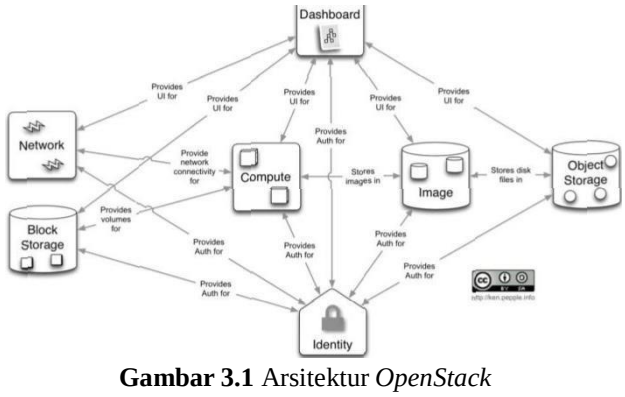
1. Endpoint : Setiap layanan *openstack* (Nova, Swift, Glance) yang berjalan pada port khusus dan pada URL khusus (host).
2. Regions : sebuah tempat yang mendefinisikan lokasi fisik khusus di dalam pusat data.
3. User : direpresentasikan sebagai seseorang, sistem, atau *service* yang menggunakan layanan *OpenStack service*
4. Services : Setiap komponen yang sedang terhubung ke atau yang diberikan melalui keystone.
5. Role : suatu ketetapan dalam melakukan serangkaian operasi atau *service* tertentu yang ada dalam layanan *OpenStack*
6. Tenant : sebuah proyek dengan semua layanan endpoint dan role yang berhubungan dengan pengguna yang merupakan anggota tertentu.

3. Perancangan Sistem

3.1 Blok Diagram Sistem

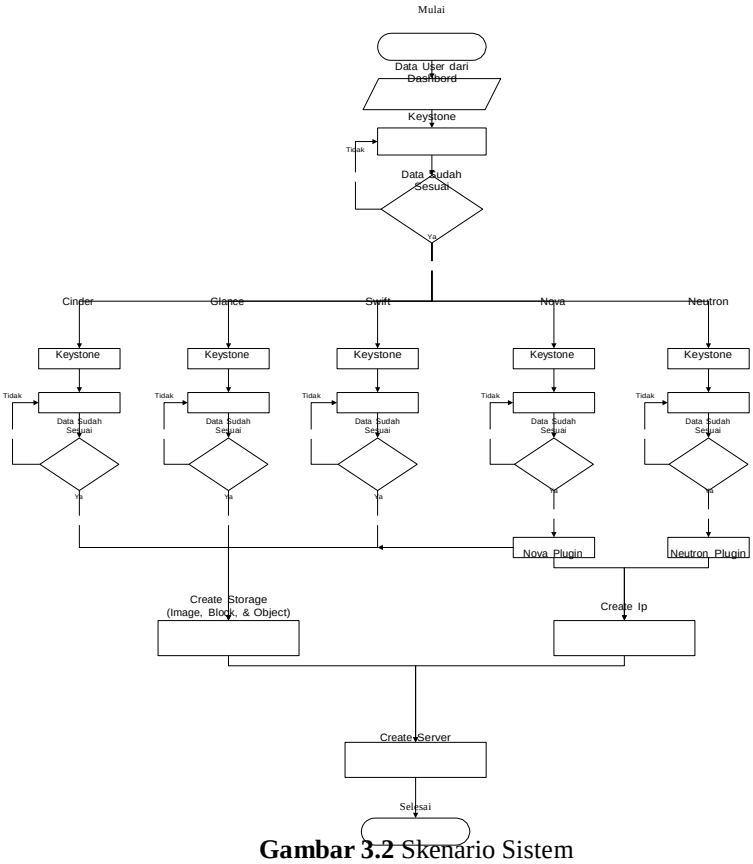
Dibawah ini merupakan gambar arsitektur secara keseluruhan dari *OpenStack* yang terdiri dari beberapa blok layanan dengan fungsi kerja masing-masing, yaitu :

- *Dashboard* : Sebagai user interface.
- *Network* : Jaringan yang berfugsi sebagai jembatan semua layanan pada *OpenStack.*
- *Compute* : Sistem yang mengatur dan mengontrol layanan-layanan pada *OpenStack.*
- *Storage* : Menyediakan media penyimpanan.
- *Identify* : Untuk keamanan data bagi *user*.



Gambar 3.1 Arsitektur *OpenStack*

3.2 Skenario Sistem



Gambar 3.2 Skenario Sistem

Keterangan :

- *User* melakukan login melalui *dashboard* yang berfungsi sebagai penghubung antara sistem *openstack* dengan *user*.
- Data *user* saat login di verifikasi oleh *keystone* apakah sudah sesuai atau belum, jika data sudah sesuai, *user* dapat masuk ke menu dashboard untuk memilih *feature* yang telah disediakan.
- Setelah *user* memilih *feature* dari server yang diinginkan, *Nova* pada *controller* mengidentifikasi akan ada penginstallan. Sebelum *Nova* melakukan penginstallan, *keystone* memberikan *Identity* API kepada *Nova* yang berfungsi untuk memberikan otorisasi dan fungsi keamanan pada saat penginstallan berlangsung.
- Setelah mendapatkan *identity* API, *Nova* siap melakukan penginstallan dengan menyediakan *Nova plugin*.
- *Storage service* yang terdiri dari (*Glance, Cinder,* dan *Swift*) akan diberikan *identity* API saat akan berkomunikasi dengan *Nova*. Jika *Identity* API sudah sesuai, *keystone* akan memberikan otorisasi untuk melanjutkan komunikasi.
- *Nova* yang sudah menyediakan *Nova plugin* mendapatkan tempat penginstallan maka *Storage Service* (*Glance, Cinder* dan *Swift*) akan meminta *Source* dari penginstallan server baik dalam format *image* ISO, *Virtual Disk*, atau yang lain.
- Setelah mendapatkan *Source* maka *Storage Service* akan segera melakukan penginstalan dan *preparing installation*.
- *Neutron* berfungsi memberikan IP kepada Nova untuk melakukan penginstallan melalui jaringan, namun sebelum *neutron* menyediakan IP, *Neutron* diberikan *identity* API untuk otorisasi melakukan penginstallan melalui jaringan. Selanjutnya *Neutron* siap menjembatani penginstallan melalui jaringan dengan menyediakan *Neutron plugin.*
- Setelah semua proses dilakukan, langkah selanjutnya adalah *create server.*

4. Pengujian Dan Analisis

4.1 Pendahuluan

Pengujian dan analisis pada *identity service* dilakukan dengan tujuan untuk mengetahui performansi dari *service* tersebut melalui *platform keystone*. Dengan dilakukan pengujian pada *platform* tersebut dapat diketahui apakah *platform* dapat bekerja dengan baik atau tidak.

4.2 Pengujian Internal Sistem OpenStack

Pengujian *internal* dari sistem *OpenStack* ini terdiri dari beberapa macam. Diantaranya pengujian dari pembuatan file *OpenStack shell (.sh),* kapasitas *user* dalam satu *tenant,* apakah satu *user* dapat memiliki banyak *tenant,* letak *api endpoint* dari masing-masing layanan yang terhubung dengan layanan *identity service,*dan pembuatan banyak *user* secara bersamaan.

4.2.1 Pengujian File OpenStack Shell (.sh)

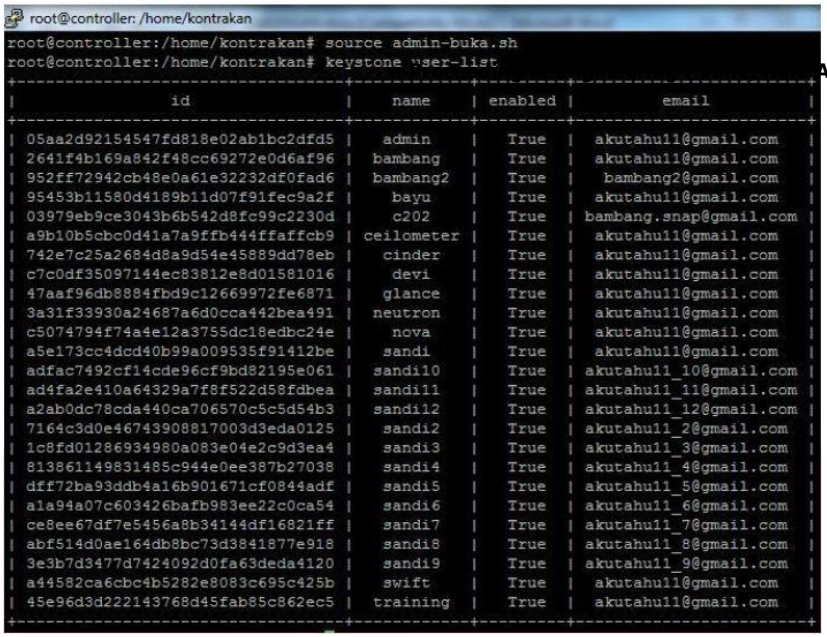
Pada sistem *OpenStack* ini, ada beberapa *file OpenStack shell* yang sudah dibuat, diantaranya admin-buka.sh, buka.sh, bambang.sh, devi.sh, dan sandi.sh. Berikut ini adalah isi dari masing-masing *file shell* tersebut.

- **Admin-buka.sh**

  File admin-buka.sh diatas berfungsi sebagai proses otentikasi dan otorisasi apabila ingin menjalankan operasi pada *platform keystone.* Berikut ini *capture* apabila tidak melakukan *source file admin-buka.sh* diatas.

Gambar 4.1 *Unsource File Admin-buka.sh*

Pada gambar diatas akan dilakukan proses menampilkan daftar *user* yang sudah dibuat, namun operasi tersebut tidak dapat dijalankan karena administrator sistem *OpenStack* belum melakukan *source* pada *file admin-buka.sh*. Berikut ini *capture* apabila sudah melakukan *source file admin-buka.sh*

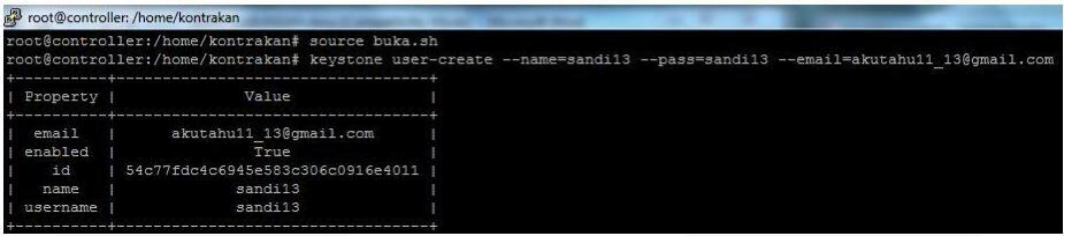


Gambar 4.2 *Source File Admin-buka.sh*

Pada gambar diatas penggunaan *command keystone user-list* berhasil dijalankan karena administrator sistem *OpenStack* sudah melakukan *source file admin-buka.sh.*

- **Buka.sh**

  File buka.*sh* diatas berfungsi sebagai proses otentikasi dan otorisasi apabila ingin membuat *user* atau *tenant* atau *role* pada *platform keystone.* Berikut ini *capture* apabila ingin membuat *user* baru.



Gambar 4.3 Pembuatan *User* Baru

Pada gambar diatas penggunaan *command keystone use-create* berhasil dijalankan karena administrator sistem *OpenStack* sudah melakukan *source file* buka.sh.

- **Bambang, Devi, Sandi.sh**

  Pada *file* bambang, devi, dan sandi.sh, perbedaan hanya terjadi pada *username, password*, dan *tenant.* Perbedaan ketiga hal tersebut dimaksudkan pada saat melakukan operasi *source* pada masing-masing file tersebut, operasi setiap *platform* pada *OpenStack* (co : nova, cinder, neutron) hanya akan menampilkan data berdasarkan *tenant* yang sudah dibuat sebelumnya.

4.2.2 Kapasitas User dalam satu Tenant

Kapasitas satu *user* dalam satu buah *tenant* dalam sistem *OpenStack* yang dibuat ini dapat terdiri dari banyak *user* namun pada pembuatan *instance* dalam satu *tenant*, dibatasi maksimal hanya sepuluh *instance.*

4.2.3 Letak API Endpoint Service

Melalui gambar 4.22, letak *api endpoint* dari masing-masing *service* dapat diketahui. Berikut ini letak masing-masing *api endpoint service.*

1. *Identity service* :
   - *-admin url* : http://controller:35357/v2.0
   - *-public url* : http://controller:5000/v2.0
2. *Metering* :
   - *-admin url* : http://controller:8777
   - *-public url* : http://controller:8777
3. *Object-store* :
   - *-admin url* : http://192.168.100.40:8080/v1/AUTH_%(tenant_id)s
   - *-public url* : http://192.168.100.40:8080/v1/AUTH_%(tenant_id)s
4. *Image* :
   - *-admin url* : http://controller:9292
   - *-public url* : http://controller:9292

5. *Network* :
   -admin url : http://controller:9696
   -public url : http://controller:9696
6. *Volume* :
   -admin url : http://controller:8776/v1/%(tenant_id)s
   -public url : http://controller:8776/v1/%(tenant_id)s
7. *Volumev2* :
   -admin url : http://controller:8776/v2/%(tenant_id)s
   -public url : http://controller:8776/v2/%(tenant_id)s
8. *Compute* :
   -admin url : http://controller:8774/v2/%(tenant_id)s
   -public url : http://controller:8774/v2/%(tenant_id)s
9. *Compute2* :
   -admin url : http://controller:8774/v4/%(tenant_id)s
   -public url : http://controller:8774/v4/%(tenant_id)s
10. *Compute3* :
    -admin url : http://controller:8774/v6/%(tenant_id)s
    -public url : http://controller:8774/v6/%(tenant_id)s

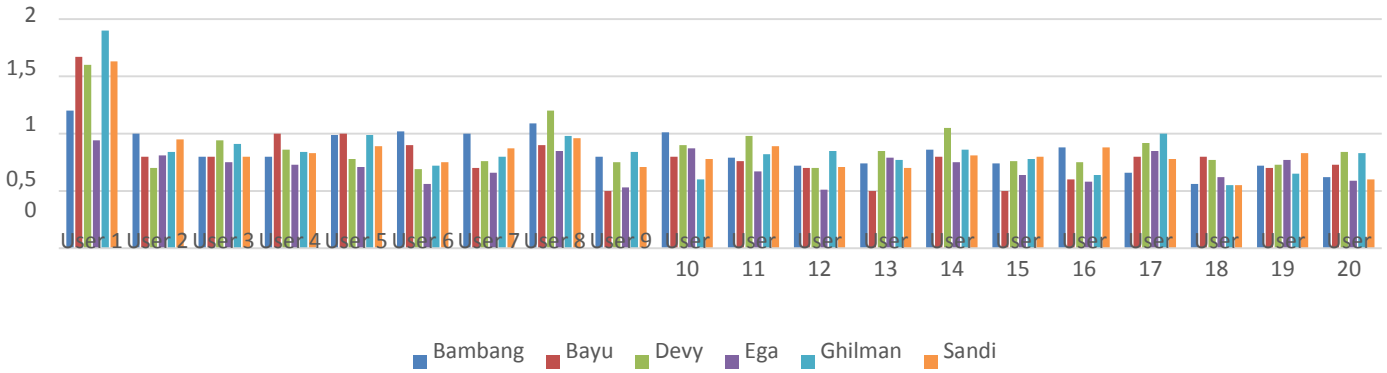
4.2.4 Pengukuran User Create

Pengukuran *user create* yang dilakukan pada sistem *OpenStack* yang dibuat ini dilakukan untuk mengetahui berapa lama waktu yang dibutuhkan oleh *platform keystone* dalam melakukan pembuatan *user* baru secara bersamaan melalui beberapa komputer. Pada pengukuran ini dikakukan tiga skenario dalam percobaannya dimana skenario pertama adalah percobaan pembuatan tujuh *user* secara bersamaan, skenario kedua adalah percobaan pembuatan empat belas *user* secara bersamaan, dan skenario ketiga adalah percobaan pembuatan dua puluh *user* secara bersamaan. Berikut ini tabel dari percobaan pembuatan *user* secara bersamaan tersebut.

Tabel 4.1 Percobaan *Create* Dua Puluh *User* Secara Bersamaan

Create User (detik)	Bambang	Bayu	Devy	Ega	Ghilman	Sandi
User 1	1,2	1,67	1,6	0,94	1,9	1,63
User 2	1	0,8	0,7	0,81	0,84	0,95
User 3	0,8	0,8	0,94	0,75	0,91	0,8
User 4	0,8	1	0,86	0,73	0,84	0,83
User 5	0,99	1	0,78	0,71	0,99	0,89
User 6	1,02	0,9	0,69	0,56	0,72	0,75
User 7	1	0,7	0,76	0,66	0,8	0,87
User 8	1,09	0,9	1,2	0,85	0,98	0,96
User 9	0,8	0,5	0,75	0,53	0,84	0,71
User 10	1,01	0,8	0,9	0,87	0,6	0,78
User 11	0,79	0,76	0,98	0,67	0,82	0,89
User 12	0,72	0,7	0,7	0,51	0,85	0,71
User 13	0,74	0,5	0,85	0,79	0,77	0,7
User 14	0,86	0,8	1,05	0,75	0,86	0,81
User 15	0,74	0,5	0,76	0,64	0,78	0,8
User 16	0,88	0,6	0,75	0,58	0,64	0,88
User 17	0,66	0,8	0,92	0,85	1	0,78
User 18	0,56	0,8	0,77	0,62	0,55	0,55
User 19	0,72	0,7	0,73	0,77	0,65	0,83
User 20	0,62	0,73	0,84	0,59	0,83	0,6

Untuk mempermudah analisa, berikut ini grafik dari ke tiga tabel diatas



Gambar 4.4 Grafik *Create* Dua Puluh *User* Secara Bersamaan

Dari ketiga grafik diatas, dapat dilihat bahwa pembuatan *user* yang dilakukan secara bersamaan melalui *user* bambang, bayu, devy, ega, ghilman, dan sandi hanya memerlukan waktu :

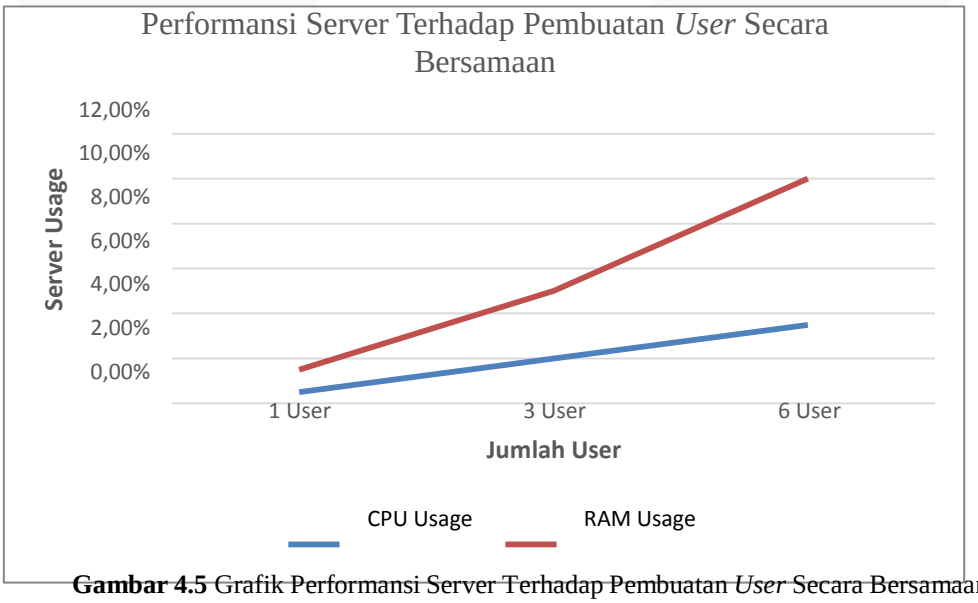
1. Kurang dari 0,26 detik pada percobaan pembuatan tujuh *user* secara bersamaan
2. Kurang dari 0,5 detik pada percobaan pembuatan empat belas *user* secara bersamaan.
3. Kurang dari 2 detik pada percobaan pembuatan dua puluh *user* secara bersamaan.

Hal ini menandakan performansi dari *platform keystone* sangat baik karena toleransi suatu sistem melakukan *create user* adalah 5 detik. Selain pengukuran pembuatan *user* secara bersamaan diatas, dilakukan juga pengukuran performansi server terhadap pembuatan *user* secara bersamaan diatas. Berikut ini tabel pengukuran performansi server terhadap pembuatan *user* yang dilakukan secara bersamaan.

Tabel 4.2 Performansi Server Terhadap Pembuatan *User* Secara bersamaan

Jumlah Pembuatan User Secara Bersamaan	Server Usage	
	CPU	RAM
1 User	0,5%	1,5%
3 User	2%	5%
6 User	3,5%	10%

Untuk mempermudah analisa, berikut ini grafik dari tabel diatas



Gambar 4.5 Grafik Performansi Server Terhadap Pembuatan *User* Secara Bersamaan

Dari grafik diatas, dapat dilihat bahwa server memiliki performansi yang baik karena penggunaan CPU dan RAM dibawah 11%. Selain itu berdasarkan grafik diatas dapat disimpulkan bahwa semakin banyak pembuatan *user* secara bersamaan, semakin besar juga penggunaan CPU dan RAM server.

5. Kesimpulan

1. Sistem *OpenStack* yang dibuat dapat berjalan dengan baik, hal ini ditandai dengan jalannya VNC pada *dashboard*
2. Pembuatan file *OpenStack shell* berfungsi sebagai otentikasi dan otorisasi terhadap serangkaian operasi yang akan dilakukan
3. Dalam sistem *OpenStack* yang dibuat, satu *tenant* tediri dari banyak *user* namun satu *tenant* hanya dapat membuat sebanyak sepuluh *instance*
4. Dalam sistem *OpenStack* yang dibuat, satu *user* dapat memiliki banyak *tenant.*
5. *Service id* berperan penting dalam hal penentuan *api endpoint.*
6. *Api endpoint service* tediri dari dua macam *url,* diantaranya *admin url* dan *public url*

Daftar Pustaka:

[1] Alex. "Apa Itu Cloud Computing?". 26 April 2012. http://www.cloudindonesia.or.id/apa-itu-cloud-computing.html diakses pada tanggal 4 Februari 2015.

[2] Presekal, Alfan. "Software Defined Network (SDN) Sebuah Paradigma Dunia Jaringan". 21 November 2012. http://alfan.presekal.com/software-defined-network-sdn-paradigma-baru-dunia-jaringan/ diakses pada tanggal 4 februari 2015

[3] Messerschmitt, Zwei "Pengertian SaaS, PaaS dan paaS". 24 Februari 2012. https://zweimesserschmitt.wordpress.com/2012/02/24/pengertian-saas-paas-dan-iaas/ diakses pada tanggal 9 Februari 2015.

[4] Putra, Hasdi. "Definisi Openflow". 27 Februari 2013 http://paneliti.wordpress.com/2013/02/27/defenisi-openflow/ diakses pada 18 maret 2014

[5] Gudanglinux."IBM Tawarkan OpenStack Untuk Solusi di Awan". 5 Maret 2013 https://gudanglinux.wordpress.com/category/application-software/cloud/ diakses pada tanggal 18 maret 2014

[6] Kuntoaji."OpenStack, Perangkat Lunak Cloud Computing Open Source". 25 Juli 2010 http://kuntoaji.blogspot.com/2010/07/openstack-perangkat-lunak-cloud.html diakses pada tanggal 18 maret 2014

[7] OpenStack Foundation."OpenStack Storage". http://www.openstack.org/software/openstack-storage/ diakses pada tanggal 11 Februari 2015

[8] Mulyana Eueung. "Buku Komunitas SDN-RG". http://eueung.gitbooks.io/buku-komunitas-sdn-rg/content/pengantar_sdn/README.html diakses pada tanggal 3 April 2015

[9] Oasisbiru. "API (application programming interface)". http://oasisbiru.blogspot.com/2013/01/api-application-programming-interface.html diakses pada tanggal 17 Januari 2015

[10] Wikipedia. "Token Akses". http://id.wikipedia.org/wiki/Token_akses diakses pada tanggal 10 Desember 2014

[11] Septian, Ridwan Fajar, 2013. **Python "Belajar Pemrograman Python Dasar"**, Bandung : POSS-UPI.

[12] Imam Prasetyo."SSL (Secure Socket Layer)". 30 April 2013 http://ilmukomputer.org/2013/04/30/ssl-secure-socket-layer/ diakses pada tanggal 18 april 2014

[13] Rio Handicha."Pengertian dan Manfaat Memakai SSH". Juni 2014 http://riohandicha.blogspot.com/2014/06/pengertian-dan-manfaat-memakai-ssh.html diakses pada tanggal 19 april 2015

[14] Sidik Prasetiyo."Sekilas Tentang Public Key Infrastructure (PKI)". 12 Desember 2012 http://sidikprst.blogspot.com/2012/12/sekilas-tentang-public-key.html diakses pada tanggal 9 mei 2015

[15] Culesshabrur."REST API". 31 Januari 2013 http://s4nbao.blogspot.com/2013/01/rest-api.html diakses pada tanggal 9 april 2015

[16] Masim Vavai Sugianto." Limitasi VMWare vSphere/ESXi 5.x Free Edition dan Tips Memilih Versi Lisensi yang Paling Tepat". 6 Desember 2012 https://www.excellent.co.id/product-services/vmware/limitasi-vmware-vsphereesxi-5-x-free-edition-dan-tips-memilih-versi-lisensi-yang-paling-tepat/ diakses pada tanggal 3 april 2015

[17] hendsmountenerings."SQL Injection". https://hendsmountenerings.wordpress.com/sql-injection/ diakses pada tanggal 2 Februari 2015

[18] Ahmad Fathi Hadi."Mengenal lebih dalam XSS (Cross Site Scripting)". 26 Januari 2013 http://blog.fathihadi.net/xss-cross-site-scripting/ diakses pada tanggal 17 april 2015

[19] https://www.youtube.com/watch?v=OVLz6RgOjIY

[20] https://www.youtube.com/watch?v=v510asVxePc

[21] http://docs.openstack.org/icehouse/install-guide/install/apt/content/

[22] http://docs.openstack.org/