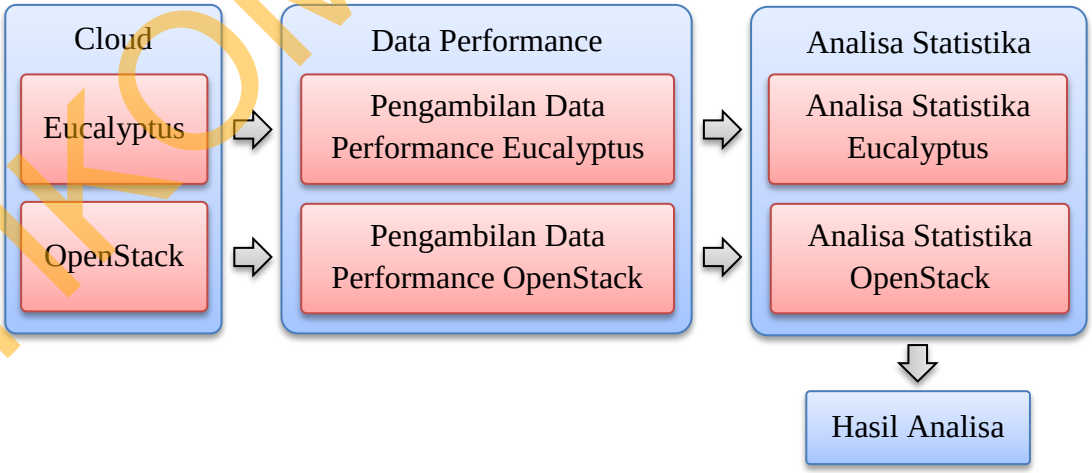
BAB III

METODOLOGI PENELITIAN

3.1 Model Penelitian

Metode penelitian yang digunakan dalam pengerjaan tugas akhir ini adalah studi kepustakaan dan melakukan analisis *performance* dari kedua *private cloud*. Dengan ini penulis berusaha untuk mengumpulkan data dan informasi-informasi, serta materi-materi dasar yang bersifat teoritis yang sesuai dengan permasalahan. Hal tersebut diperoleh dari buku-buku, materi perkualiahan, serta literatur dari internet.

Pada sub bab ini akan dibahas tentang perancangan sistem secara keseluruhan dari penelitian ini yaitu tentang analisa *performance cloud Eucalyptus* dan *OpenStack*. Adapun keseluruhan sistem pada penelitian tugas akhir analisis *performance cloud computing* berbasis *Platform as a Service* (PaaS) pada *Ubuntu server* ini sesuai dengan blok diagram pada Gambar 3.1.



Gambar 3.1 Blok Diagram Secara Umum

Pada Gambar 3.1 dapat dilihat bahwa secara umum penelitian ini dibagi dalam empat tahap. Pertama penulis akan melakukan instalasi serta konfigurasi

terhadap *Eucalyptus* dan *OpenStack* samapai dapat menjalankan sebuah *client* / *instance*, kemudian penulis akan melakukan pengambilan data dari *instance* yang berjalan tesebut.

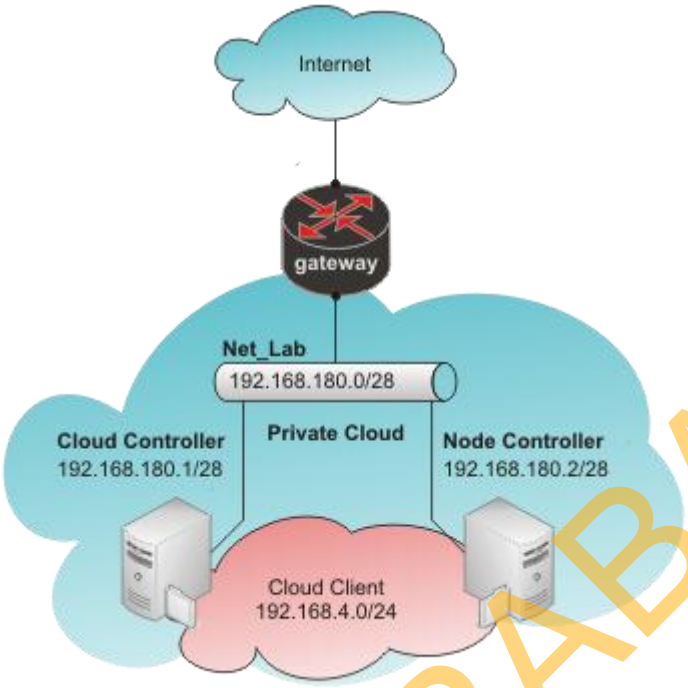
Kemudian pada tahap kedua penulis akan melakukan pengambilan data *performance* dengan cara melakukan *benchmarking* terhadap *Eucalyptus* dan *OpenStack* dengan menggunakan *benchmarking tools* berupa *phoronix test suite*. *Phoronix test suite* merupakan *benchmarking tools* yang banyak digunakan oleh para praktisi IT.

Setelah mendapatkan data maka langkah selanjutnya adalah melakukan pengujian terhadap data tersebut dan melakukan perhitungan dengan statistika menggunakan metode hipotesis untuk mengetahui hasil *performance* dari data yang telah didapat yang selanjutnya dijadikan acuan untuk menarik kesimpulan.

3.2 Perancangan *Cloud System*

Pada penelitian kali ini penulis menggunakan dua buah server. *Server* pertama dijadikan penulis sebagai *cloud controller* yang bertugas untuk mengatur jaringan yang diimplementasikan pada *cloud*, mengaplikasikan *rule-rule* yang ditetapkan oleh administrator serta dijadikan sebagai *web server* yang dapat digunakan untuk mengkonfigurasi *cloud server* serta memonitoring hasil kerja dari *cloud* tersebut.

Sedangkan untuk *server* yang kedua dijadikan sebagi *node controller* dimana kerja dari *node controller* adalah sebagai *server* yang nantinya akan menjalankan sebuah *instance* dimana *node controller* bertanggung jawab atas memory serta hardisk yang nantinya akan digunakan oleh *instance* yang berjalan.



Gambar 3.2 Topologi Jaringan *Private Cloud*

Pada Gambar 3.2 dapat diketahui bahwa *private cloud* yang dibangun dengan menggunakan jaringan lokal dengan menggunakan dua *server*, dimana *server* pertama menggunakan alamat *Internet Protocol* (IP) 192.168.180.1/28 dengan menggunakan nama *Cloud Controller* (CC). Sedangkan *server* kedua menggunakan alamat (IP) 192.168.180.2/28 dengan menggunakan nama *Node Controller* (NC) dan 192.168.180.14/28 sebagai *default gateway* dari kedua *server* tersebut.

Jika kedua server tersebut dikonfigurasi dengan benar maka akan terbentuk sebuah jaringan *cloud client*. C*loud client* ini dapat berisi bermacam-macam sistem operasi dengan *rule-rule* yang berbeda tergantung kebijakan *administrator* yang mengelola *private cloud* tersebut.

Untuk menghubungkan antara *cloud server* dengan *cloud client administrator* menggunakan *interface* br100 sebagai *bridge* antara jaringan *server* dengan jaringan *cloud client*. Dengan *bridge* ini maka *cloud client* dapat

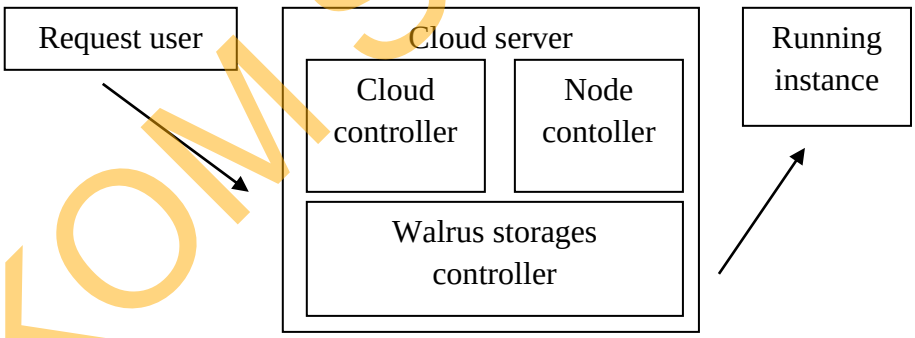
berhubungan dengan jaringan luar termasuk internet melalui jaringan komputer *server*.

Cloud client yang terbentuk dirancang dengan menggunakan alokasi IP 192.168.4.0/24. Alokasi IP tersebut mampu menyediakan 254 *user* / *cloud client* yang dapat dikonfigurasi serta diberikan *rule-rule* yang telah disepakati dengan *user* yang menggunakan *cloud client* tersebut.

Pada penelitian ini penulis menggunakan *Eucalyptus* dan *OpenStack* sebagai *cloud application service*. Penulis melakukan instalasi serta konfigurasi terhadap *Eucalyptus* dan *OpenStack*.

3.2.1 Perancangan *Eucalyptus Cloud*

Berikut ini adalah diagram blok yang dapat menggambarkan bagaimana sistem kerja *eucalyptus* secara umum.



Gambar 3.3 Diagram Blok *Eucalyptus* Secara Umum

Dari Gambar 3.3 dapat diketahui sistem kerja secara umum dari *eucalyptus* dimana awalnya adalah permintaan dari user yang meliputi *hardisk*, RAM, serta sistem operasi yang diinginkan. Kemudian permintaan tersebut akan diproses oleh server yang mempunyai 3 elemen penting yaitu *cloud controller* (CC), *node controller* (NC), dan *walrus storage controller* (WS3) yang kemudian akan menghasilkan *cloud computing*.

A. *Server* 1 (*Cloud Controller*)

Hal yang pertama dilakukan adalah melakukan penginstalan *server* seperti dijelaskan pada lampiran 1. Kemudian melakukan konfigurasi jaringan agar dapat terhubung ke internet dan dapat melakukan penginstalan paket yang dibutuhkan.

Konfigurasi jaringan seharusnya sudah dilakukan pada saat menginstall *Ubuntu server*, tetapi jika terlewatkan untuk mengkonfigurasi jaringan dapat dikonfigurasi manual pada */etc/network/interfaces* dengan menggunakan *editor* seperti *nano*, *vi*, *vim*, dsb. Berikut skrip untuk meng-konfigurasi jaringan pada *server* 1:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
      address           192.168.180.1
      netmask           255.255.255.240
      broadcast         192.168.180.15
      gateway           192.168.180.14
      dns-nameservers   222.124.29.226
```

Langkah selanjutnya adalah menginstal paket dari *cloud controller*, *cluster controller*, *walrus* dan *storage controller* yang tersedia di dalam *repository*. Aplikasi *eucalyptus-cloud*, *eucalyptus-cc*, *eucalyptus-walrus*, *eucalyptus-sc* merupakan bagian dari *cloud controller* yang nantinya akan bertanggung jawab untuk meng-*handle* beberapa *instance* yang berjalan. Sedangkan *qemu-kvm* dan *kvm-pxe* adalah aplikasi yang digunakan untuk menjalankan sebuah sistem operasi pada *local server*. Aplikasi ini yang nantinya akan digunakan untuk mengkonfigurasi sistem operasi yang akan di-*upload* ke dalam *cloud server*.

Utuk menginstalnya dapat dilakukan dengan perintah berikut ini:

```
$ sudo apt-get install eucalyptus-cloud eucalyptus-cc eucalyptus-
walrus eucalyptus-sc qemu-kvm kvm-pxe
```

Selanjutnya hal yang perlu diperhatikan adalah memberi identitas dan pengalamatan pada sebuah *cluster*. Pada penelitian ini penulis menggunakan cluster1 sebagai nama *cluster*-nya. *Cluster* ini sebenarnya adalah segmentasi dari *private cloud*, namun dalam penelitian ini hanya terdiri dari sebuah *cluster*. Sehingga *cluster* tersebut memiliki alamat IP sesuai perancangan yaitu 192.168.180.3 sampai 192.168.180.13.

Selanjutnya diperlukan komponen *euca2ools* yang berfungsi sebagai *Command Line Interface* (CLI) yang nantinya dapat digunakan sebagai jembatan untuk mengkonfigurasi *Cloud Controller*.

```
$ sudo apt-get install -y euca2ools
```

Langkah berikutnya adalah bagaimana mensinkronkan waktu antar *kedua cloud controller*. Apabila waktu antar *cloud* tidak sama, maka *cloud* tidak dapat berjalan dengan sempurna. Untuk melakukan hal ini diperlukan *Network Time Protocol* (NTP). Berikut ini adalah step-step untuk mengkonfigurasi NTP *server*. Pertama install NTP *server terlebih dahulu*.

```
$ sudo apt-get install -y ntp
```

Kemudian meng-*edit* */etc/ntp.conf* dan tambahkan beberapa konfigurasi dibawah ini.

```
server ntp.ubuntu.com
server 127.127.1.0
fudge 127.127.1.0 stratum 10
```

B. *Server* 2 (*Node Controller*)

Hal yang pertama dilakukan adalah melakukan penginstalan *server* seperti dijelaskan pada lampiran 1. Kemudian melakukan konfigurasi jaringan agar dapat terhubung ke internet dan dapat melakukan penginstalan paket yang dibutuhkan.

Konfigurasi jaringan seharusnya sudah dilakukan pada saat menginstall *Ubuntu server*, tetapi jika terlewatkan untuk mengkonfigurasi jaringan dapat dikonfigurasi manual pada */etc/network/interfaces* dengan menggunakan *editor* seperti *nano*, *vi*, *vim*, dsb. Berikut skrip untuk meng-konfigurasi jaringan pada *server* 2:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
      address           192.168.180.2
      netmask           255.255.255.240
      broadcast         192.168.180.15
      gateway           192.168.180.14
      dns-nameservers   222.124.29.226
```

Setelah mengkonfigurasi IP address, hal selanjutnya adalah menginstal paket dari *node controller*. Untuk menginstalnya dapat dilakukan dengan perintah berikut ini:

```
$ sudo apt-get install eucalyptus-nc
```

Kemudian hal dilakukan selanjutnya adalah mengkonfiguasi *bridge* yang nantinya akan digunakan sebagai *mode bridge* pada jaringan yang digunakan oleh *eucalyptus*. Untuk mengkonfigurasikannya dapat dilakukan dengan meng-*edit* */etc/eucalyptus/eucalyptus.conf* dan menambahkan beberapa konfigurasi dibawah ini:

```
VNET_PUBINTERFACE="eth0"
VNET_PRIVINTERFACE="br100"
VNET_BRIDGE="br100"
VNET_DHCPDAEMON="/usr/sbin/dhcpd3"
VNET_DHCPUSER="dhcpd"
VNET_MODE="MANAGED-NOVLAN"
```

Keterangan dari masing-masing skrip diatas adalah sebagai berikut :

1. `VNET_PUBINTERFACE="eth0"`, digunakan untuk memberitahukan terhadap aplikasi *eucalyptus* bahwa *public network* berada pada *interface eth0*.

2. `VNET_PRIVINTERFACE="br100"`, digunakan untuk memberitahukan terhadap aplikasi *eucalyptus* bahwa *private network* berada pada *interface br100*.
3. `VNET_BRIDGE="br100"`, digunakan untuk memberitahukan terhadap aplikasi *eucalyptus* bahwa jaringan *public* akan dijembatani oleh *interface br100* untuk bisa berhubungan dengan jaringan lokal.
4. `VNET_DHCPDAEMON="/usr/sbin/dhcpd3"`, digunakan untuk memberitahukan terhadap aplikasi *eucalyptus* bahwa *dhcp daemon* berada pada *direcrory* `/usr/sbin/dhcpd3`.
5. `VNET_DHCPUSER="dhcpd"`, digunakan untuk memberitahukan terhadap aplikasi eucalyptus bahwa jaringan *virtual* pada penelitian kali ini menggunakan DHCP.
6. `VNET_MODE="MANAGED-NOVLAN"`, digunakan untuk memberitahukan terhadap aplikasi *eucalyptus* bahwa jaringan *virtualisasi* yang dibangun tidak menggunakan VLAN.

Langkah berikutnya adalah bagaimana mensinkronkan waktu antar *kedua cloud controller*. Apabila waktu antar *cloud* tidak sama, maka *cloud* tidak dapat berjalan dengan sempurna. Untuk melakukan hal ini diperlukan *Network Time Protocol* (NTP). Berikut ini adalah step-step untuk mengkonfigurasi NTP *server*. Pertama install NTP *server terlebih dahulu*.

```
$ sudo apt-get install -y ntp
```

Kemudian meng-*edit* */etc/ntp.conf* dan tambahkan beberapa konfigurasi dibawah ini.

```
server 192.168.180.1
```

Langkah Terakhir adalah bagaimana mensinkronkan *eucalyptus user* antar *kedua cloud controller*. Apabila waktu antar *cloud* tidak sama, maka *cloud* tidak dapat berjalan dengan sempurna. Untuk melakukan hal ini diperlukan SSH *public key*. SSH *public key* adalah sebuah *key* yang berfungsi agar *eucalyptus user* di *cloud controller* dan *node controller* bisa sinkron. Caranya dengan meng-*install* SSH *public key*-nya *eucalyptus user* di *cloud controller* ke *eucalyptus user* di *node controller*.

Hal pertama yang harus dilakukan adalah men-*set password eucalyptus user* yang ada di *node controller*. Berikut perintahnya:

```
sudo passwd eucalyptus
```

Selanjutnya installasi S*SH public key*-nya *eucalyptus user* di *cloud controller* ke *eucalyptus user* di *node controller*. Berikut perintahnya:

```
$ sudo -u eucalyptus ssh-copy-id -i ~eucalyptus/.ssh/id_rsa.pub
eucalyptus@192.168.180.2
```

Terakhir hapus password *eucalyptus user* yang ada di *node controller*. Ini hanya optional, boleh dihapus boleh juga tidak. Berikut perintahnya:

```
$ sudo passwd -d eucalyptus
```

C. *Credential*

Credential adalah sebuah *identity service* yang digunakan oleh *Eucalyptus*. *Credential* sendiri sudah tersedia di *cloud controller* (*server 1*). Cara untuk mengambil / men-*download credential* sendiri ada 2, yaitu:

1. *Web browser*

Masuk ke *web browser*, lalu ketik *https://192.168.180.1:8443/*. Dimana 192.168.180.1 adalah IP dari *cloud controller* (*server 1*). Nantinya akan ada

tampilan *login page*, *username* dan *password* yang *default* adalah *admin*. Kemudian ada petunjuk pada layar untuk memperbarui *password admin* dan alamat *email*. Setelah selesai, klik *tab credential* yang terletak di bagian kiri atas layar. Klik tombol *credential Download* untuk mendapatkan *credential*-nya. Setelah *credential*-nya didapat, *copy credential*-nya ke *cloud controller* (*server 1*). Terakhir *extract credential* tersebut. Berikut perintah untuk meng-*extract*:

```
$ unzip -d ~/.euca mycreds.zip
```

2. *Command Line*

Pada *cloud controller* (*server 1*) dapat mengetik perintah sebagai berikut untuk mendapatkan *cerdential*-nya.

```
$ mkdir -p ~/.euca
$ chmod 700 ~/.euca
$ cd ~/.euca
$ sudo euca_conf --get-credentials mycreds.zip
```

Terakhir *extract credential* tersebut. Berikut perintah untuk meng-*extract*:

```
$ unzip mycreds.zip
$ ln -s ~/.euca/eucarc ~/.eucarc
$ cd -
```

Setelah *credential*-nya ter-*extract*, setalah itu lakukan validasi bahwa semuanya bekerja dengan benar. Berikut perintahnya:

```
$ ~/.euca/eucarc
$ euca-describe-availability-zones verbose
AVAILABILITYZONE     myowncloud      192.168.1.1
AVAILABILITYZONE     |- vm types     free / max   cpu   ram  disk
AVAILABILITYZONE     |- m1.small     0004 / 0004   1    192     2
AVAILABILITYZONE     |- c1.medium    0004 / 0004   1    256     5
AVAILABILITYZONE     |- m1.large     0002 / 0002   2    512    10
AVAILABILITYZONE     |- m1.xlarge    0002 / 0002   2   1024    20
AVAILABILITYZONE     |- c1.xlarge    0001 / 0001   4   2048    20
```

D. Pengaturan *Image* Pada *Eucalyptus*

Pada pengaturan *image* akan dijelaskan bagaimana sistem operasi *Ubuntu* dapat berjalan pada *eucalyptus system*, tidak hanya *Ubuntu* yang bisa berjalan pada *eucalyptus cloud*. Konfigurasian dapat dilakukan terhadap semua sistem operasi yang ada, karena *cloud* dirancang dengan sangat *flexible*. Hal ini memungkinkan para *developer cloud* untuk mengkonfigurasi *private cloud* mereka sesuai dengan *rule-rule* yang diterapkan pada *private cloud* mereka masing-masing.

1. Pembuatan Linux *Image*

Pada penelitian ini penulis membuat sebuah *disk* (*harddisk)* yang nantinya akan digunakan untuk penempatan sistem operasi yang akan di-*install* dengan cara sebagai berikut :

```
$ kvm-img create -f raw server.img 10G
```

Dimana *raw* adalah jenis *filesystem*, sedangkan *server.img* adalah nama *image* dan 10G adalah besar kapasitas *disk* yang disediakan sebesar 10 *GigaByte*.

2. Instalasi Sistem Operasi

Cara instalasi sistem operasi tidak jauh berbeda dengan bagaimana meng-*instal* sistem operasi pada PC (*Personal Computer*). Cuma perbedaannya adalah anda cukup meng-*install* satu kali saja kemudian selanjutnya dapat digunakan secara berulang-ulang.

Untuk sistem operasi *Ubuntu* anda dapat men-*download image* yang berada pada *http://release.ubuntu.com* dengan menggunakan *wget* atau *downloader* lainnya atau anda dapat membuat *file iso* dari CD atau DVD *drive*. Setelah *file iso*

tersedia maka anda cukup menjalankan perintah berikut dengan asumsi *file iso*-nya bernama Ubuntu-10.04-desktop.iso :

```
$ sudo kvm -m 256 -cdrom ubuntu-10.04-desktop-i386.iso -drive
file=server.img,if=scsi,index=0 -boot d -net nic -net user -
nographic -vnc :0
```

Kemudian kembali pada PC *desktop* yang terhubung satu jaringan dengan *server* dan jalankan *vncviewer* kemudian masukan IP dari *server* dimana perintah diatas dijalankan. Pada penelitian ini IP dari server tersebut adalah 192.168.180.1 sehinga pada *vncviewer* IP yang dimasukan adalah:

```
192.168.180.1:0
```

Atau dapat juga dijalankan dari *terminal* Ubuntu dengan perintah sebagai berikut:

```
vncviewer 192.168.180.1:0
```

Dimana :0 adalah *index* dari *vncvewer* yang memungkinkan anda menjalankan *multiple remoter desktop*.

Setelah proses instalasi sistem operasi selesai, maka selanjutnya men-*shutdown client* tersebut dengan perintah berikut :

```
sudo shutdown -P now
```

Kemudian menjalankan lagi *imge* dengan perintah sebagai berikut:

```
$ sudo kvm-m 256 -drive file=image.img,if=scsi,index=0,boot=on -
boot c -net nic -net user -nographic -vnc :0
```

Kemudian kembali lagi pada PC *desktop* anda yang terhubung satu jaringan dengan *server* dan jalankan kembali *vncviewer* kemudian masukan IP dari *server* dimana anda menjalankan perintah diatas, pada penelitian kali ini IP dari server tersebut adalah 192.168.180.1 sehinga pada *vncviewer* IP yang dimasukan adalah:

```
192.168.180.1:0
```

Atau dapat juga dijalankan dari *terminal* Ubuntu dengan perintah sebagai berikut:

```
$ vncviewer 192.168.180.1:0
```

Setelah muncul *desktop* dari *client Ubuntu* tadi, maka hal selanjutnya yang harus dilakukan adalah melakukan *update system* dan meng*install ssh* pada *client* agar nantinya *client* yang anda buat dapat di-*remote* dari jarak jauh menggunakan protokol *ssh* (*secure shell*).

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get install openssh-server curl
```

Lalu menambahkan beberapa skrip diatas "*exit 0*" pada */etc/rc.local*.

```
depmod -a
modprobe acpiphp
# simple attempt to get the user ssh key using the meta-data
service
# assuming ?user? is the username of an account that has been
created
mkdir -p /home/user/.ssh
echo >> /home/user/.ssh/authorized_keys
curl -m 10 -s
http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key
j
grep 'ssh-rsa' >> n
/home/user/.ssh/authorized_keys
echo "AUTHORIZED_KEYS:"
echo "************************"
cat /home/user/.ssh/authorized_keys
echo "************************"
```

Kemudian hapus *file /etc/udev/rules.d/70-persistent-net.rules* agar nantinya ketika sistem operasi dijalankan tidak terjadi masalah saat *cloud server* menambahkan IP secara automatis terhadap client yang dijalankan.

```
$ sudo rm –vrf /etc/udev/rules.d/70-persistent-net.rules
```

Setelah selesai dilakukan maka *copy*-kan *kernel* dan *initrd image client* ke *cloud controller* agar bisa di-*upload*. Pada penelitian kali ini *kernel client* adalah *vmlinuz-2.6.28-11-server* dan *initrd image client* adalah *initrd.img-2.6.28-11-server*. Berikut perintahnya:

```
$ scp /boot/initrd.img-2.6.28-11-server server@192.168.180.1:
/home/server/
$ scp /boot/vmlinuz-2.6.28-11-server server@192.168.180.1:
/home/server/
```

Selanjutnya men-*shutdown client* tersebut dengan perintah berikut :

```
$ sudo shutdown -P now
```

3. *Registering Image*

Tahap terakhir adalah meng-*upload file img* yang sudah dipasangi sistem operasi ke *eucalyptus*. Proses ini membutuhkan *kernel* dan *initrd image* yang telah di-*copy*-kan tadi. Sebelum proses meng-*upload*, pastikan *client* dalam posisi mati agar proses lancar.

Proses *registering* sendiri terdiri dari tiga tahap, yaitu:

a. *Registering Kernel Image*, membutuhkan *kernel client* yang sudah di*copy*kan tadi. Pada penelitian kali ini *kernel client* adalah *vmlinuz-2.6.28-11-server*.

```
$ euca-bundle-image -i vmlinuz-2.6.28-11-server --kernel true
$ euca-upload-bundle -b mybucket -m /tmp/vmlinuz-2.6.28-11-
server.manifest.xml
$ euca-register mybucket/vmlinuz-2.6.28-11-
server.manifest.xml
```

Kemudian simpan hasil *output* dari proses ini (eki-XXXXXXXX) karena akan dibutuhkan saat *registering disk image.*

b. *Registering Ramdisk Image*, membutuhkan *initrd image client* yang sudah di*copy*kan tadi. Pada penelitian kali ini *initrd image client* adalah *initrd.img-2.6.28-11-server*.

```
$ euca-bundle-image -i initrd.img-2.6.28-11-server --ramdisk
-true
$ euca-upload-bundle -b mybucket -m /tmp/initrd.img-2.6.28-
11-server.manifest.xml
$ euca-register mybucket/initrd.img-2.6.28-11-
server.manifest.xml
```

Simpan hasil *output* dari proses ini (eri-XXXXXXXX) karena akan dibutuhkan saat *registering disk image*.

c. *Registering disk image*, membutuhkan *key* hasil ouput dari proses *registering kernel image* (eki-XXXXXXXX) dan *registering disk image* (eri-XXXXXXXX). Pada penelitian ini *key*-nya adalah eki-XXXXXXXX dan eri-XXXXXXXX.

```
$ euca-bundle-image -i image.img --kernel eki-XXXXXXXX --
ramdisk eri-XXXXXXXX
$ euca-upload-bundle -b mybucket –m
/tmp/image.img.manifest.xml
$ euca-register mybucket/image.img.manifest.xml
```

Simpan hasil *output* dari proses ini (emi-XXXXXXXX) karena akan dibutuhkan saat menjalankan *instance*.

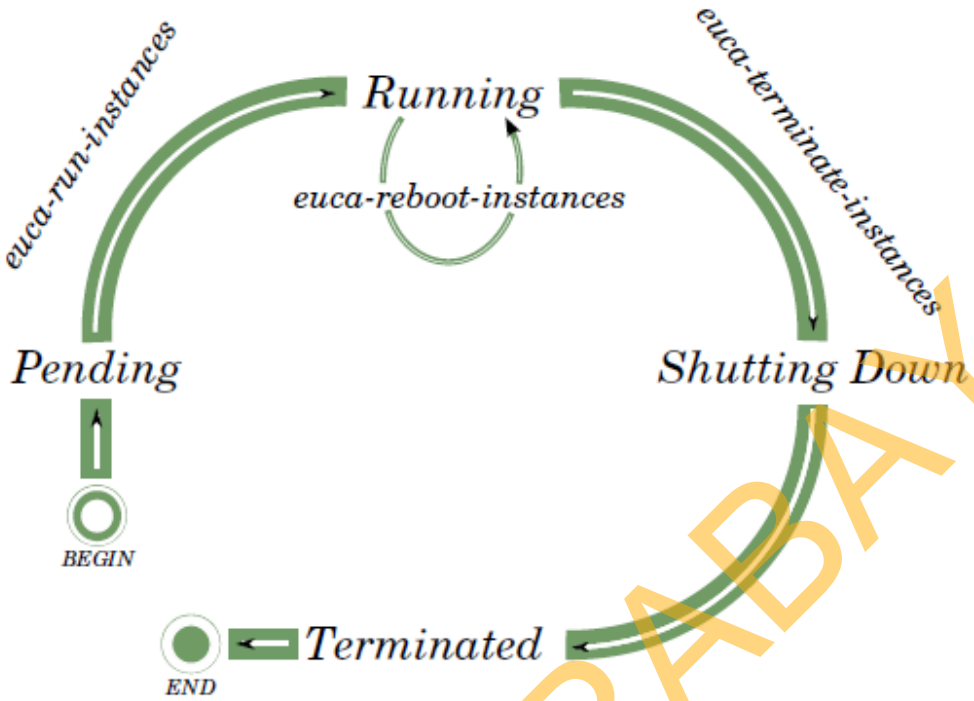
4. *Image Listing*

Setelah proses *registering image* selesai, dapat dilihat hasil *image* yang sudah di-*upload* dengan perintah seperti berikut:

```
$ euca-describe-images
IMAGE emi-70B70EC0 mybucket/image.img.manifest.xml admin available
public x86_64 machine
IMAGE eri-A2BE13EC mybucket/initrd.img-2.6.28-11-
server.manifest.xml admin available public x86_64 ramdisk
IMAGE eki-685F1306 mybucket/vmlinuz-2.6.28-11-server.manifest.xml
admin available public x86_64 kernel
```

E. *Instance*

Instance adalah nama lain dari *client* yang sedang *berjalan*, *instance* mempunyai *siklus* yang dapat digambarkan sebagai berikut:



Gambar 3.4 Siklus *Instance Eucalyptus* (Johnson, *Eucalyptus Beginner's Guide*)

Pada Gambar 3.4 dapat dilihat bahwa awal mula *instance* dijalankan, status *instance* masih *pending*. Pada tahap ini *instance* harus di-*run* supaya *instance* tersebut diberikan IP *address* agar *instance* siap untuk dijalankan. Setelah itu tahap selanjutnya adalah *shutting down* yaitu tahap dimana *instance* sudah selesai digunakan dan dalam proses dimatikan, dan tahap terakhir adalah *terminate*, dimana *instance* sudah benar-benar dalam keadaan sudah mati dan tidak dapat digunakan lagi.

Hal pertama yang harus dilakukan adalah membuat sebuah *key* yang nantinya akan digunakan oleh *ssh* untuk mengenali komputer yang digunakan, berikut adalah cara untuk membuat *key*:

```
$ cd ~/.euca
/.euca$ source eucarc
$ euca-add-keypair mykey > mykey.priv
$ chmod 600 mykey.priv
```

Disini harus dibuat *key* dengan nama *mykey* atau yang lainnya, untuk melihat keynya dapat menggunakan perintah berikut:

```
$ euca-describe-keypairs
KEYPAIR mykey
f7:ac:8e:f5:05:19:2b:31:28:8c:9b:d7:b8:07:0c:3c:b6:34:8f:79
KEYPAIR helloworld
12:96:b3:21:34:8d:6a:3f:92:2e:2b:70:23:ff:7f:51:b5:b7:ad:37
KEYPAIR ubuntu
f6:af:9a:59:65:35:32:c4:3a:c4:62:0e:e1:44:0f:71:29:03:2d:91
KEYPAIR lucid
74:04:70:33:ed:57:7a:30:36:1f:ca:c6:ec:d5:4f:10:34:1a:52:51
KEYPAIR karmic
01:f9:aa:5f:4d:20:e2:53:d1:29:d0:0f:e2:f3:8c:21:91:70:7e:c8
```

Jika ingin menghapus *key* yang sudah ada, dapat menggunakan cara berikut ini:

```
$ euca-delete-keypair helloworld
```

Setelah membuat *key*, maka langkah selanjutnya adalah menjalankan *instance*. Berikut adalah cara untuk menjalankan *instance*:

```
$ euca-run-instances emi-721D0EBA -k mykey -t c1.medium
RESERVATION r-55560977 admin admin-default INSTANCE i-50630A2A
emi-721D0EBA 0.0.0.0 0.0.0.0 pending mykey 2010-05-07T07:17:48.23Z
eki-675412F5 eri-A1E113E0
```

Dimana *emi*-721D0EBA adalah *output* dari *registering disk image*. *c1.medium* adalah *resource* yang disedakan.

Untuk melihat *resource* yang disediakan dapat menggunakan perintah berikut:

```
$ euca-describe-availability-zones verbose
```

Untuk dapat memastikan apakah *instance* sudah tercipta atau belum dan melihat status dari *instance* tersebut, dapat menggunakan perintah berikut:

```
$ euca-describe-instances
```

Untuk melakukan *reboot* terhadap *instance* dapat menggunakan perintah berikut:

```
$ euca-reboot-instances <id_dari_instance>
```

Untuk menghapus *instance* dapat menggunakan perintah berikut:

```
$ euca-terminate-instances <id_dari_instance>
```

Untuk melihat daftar *console* dapat menggunakan perintah berikut:

```
$ euca-get-console-output <id_dari_instance>
```

Setelah instance dalam keadaan *running* / *active,* dapat diakses melalui SSH dengan cara sebagai berikut:

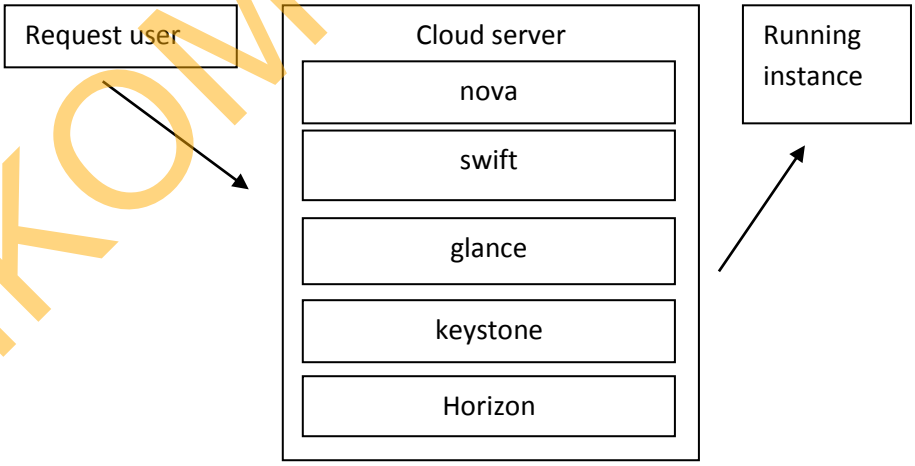
```
ssh -i <private_key> username@<ip_address>
```

misal: ada sebuah *instance* yang berjalan dengan *user client* dan IP 192.168.4.2 dengan *ssh key* adalah *mykey* maka cara untuk mengakses *instance* tersebut adalah:

```
ssh –i mykey client@192.168.4.2
```

3.2.2 Perancangan *OpenStack Cloud*

Berikut ini adalah diagram blok yang dapat menggambarkan bagaimana sistem kerja *openstack* secara umum.



Gambar 3.5 Diagram Blok *OpenStack* Secara Umum

Dari Gambar 3.5 dapat diketahui sistem kerja secara umum dari *openstack*, dimana awalnya adalah permintaan dari *user* yang meliputi *disk*, *memory*, serta sistem operasi yang diinginkan. Kemudian permintaan tersebut akan diproses oleh

server yang mempunyai 5 elemen penting yang kemudian akan menghasilkan *cloud computing*, yaitu:

1. *Nova* (*Compute Service*)
2. *Swift* (*Storage Service*)
3. *Glance* (*Imaging Service*)
4. *Keystone* (*Identity Service*)
5. *Horizon* (UI *Service* / *Web Base*)

A. *Server* 1 (*Cloud Controller*)

Hal yang pertama dilakukan adalah melakukan penginstalan *server* seperti dijelaskan pada lampiran 1. Kemudian melakukan konfigurasi jaringan agar dapat terhubung ke internet dan dapat melakukan penginstalan paket yang dibutuhkan.

Konfigurasi jaringan seharusnya sudah dilakukan pada saat menginstall *Ubuntu server*, tetapi jika terlewatkan untuk mengkonfigurasi jaringan dapat dikonfigurasi manual pada */etc/network/interfaces* dengan menggunakan *editor* seperti *nano, vi, vim,* dsb. Berikut skrip untuk meng-konfigurasi jaringan pada *server* 1:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
      address           192.168.180.1
      netmask           255.255.255.240
      broadcast         192.168.180.15
      gateway           192.168.180.14
      dns-nameservers   222.124.29.226
```

Setelah mengkonfigurasi IP *address*, hal selanjutnya adalah meng-*install bridge-utils* yang akan digunakan sebagai *mode bridge* pada jaringan yang

openstack. Untuk meng-*install bridge-utils* dapat dilakukan dengan perintah berikut ini :

```
sudo apt-get install -y bridge-utils
```

Langkah berikutnya adalah bagaimana mensinkronkan waktu antar *kedua cloud controller*. Apabila waktu antar *cloud* tidak sama, maka *cloud* tidak dapat berjalan dengan sempurna. Untuk melakukan hal ini diperlukan *Network Time Protocol* (NTP). Berikut ini adalah step-step untuk mengkonfigurasi NTP *server*. Pertama install NTP *server terlebih dahulu.*

```
$ sudo apt-get install -y ntp
```

Kemudian meng-*edit /etc/ntp.conf* dan tambahkan beberapa konfigurasi dibawah ini.

```
server ntp.ubuntu.com
server 127.127.1.0
fudge 127.127.1.0 stratum 10
```

1. Database

OpenStack membutuhkan *database* sebagai tempat penyimpanan data serta semua informasi tentang *client* yang berjalan. *OpenStack* mendukung berbagai macam *database* aplikasi seperti MySQL, *postgresql*, *sqlite*, dll.

a. MySQL

Untuk membuat MySQL berkerja harus dilakukan beberapa konfigurasi sebagai berikut. Instalasi MySQL dengan perintah berikut:

```
sudo apt-get install -y mysql-server
```

Pada saat instalasi berjalan, MySQL *server* akan meminta *password* yang digunakan sebagai *password* utama (*root password*). Pada penelitian ini *password* yang digunakan adalah "*myygreatsecret*". Setelah instalasi berhasil selanjutnya

mengganti konfigurasi pada */etc/mysql.my.cnf* dengan merubah pada baris *bind-address = 127.0.0.1* menjadi *bind-address = 0.0.0.0* untuk menjadikan MySQL *server* dapat menerima koneksi dari manapun.

b. *Postgresql*

Untuk *postgresql* database berjalan, dibutuhkan beberapa *driver* agar *postgresql* dapat dikonfigurasi dengan *openstack*. Cara instalasi *postgresql* adalah sebagai berikut :

```
sudo apt-get install -y postgresql python-psycopg2
```

Selanjutnya merubah konfigurasi "*listen_addresses = ''* menjadi "*listen_addresses = *''* pada */etc/postgresql/9.1/main/postgresql.conf* agar dapat menerima koneksi dari semua *host*.

c. SQLITE

Untuk *sqlite* tidak perlu melakukan konfigurasi. Konfigurasinya hanya akan dilakukan pada *nova*. Cukup dengan menginstall *sqlite* dengan perintah berikut:

```
sudo apt-get install -y sqlite
```

2. Keystone

Keystone adalah sebuah *identity service* yang digunakan oleh *OpenStack*. *Service* yang berfungsi sebagai sebuah *key* yang digunakan untuk mengidentifikasi apakah *user* tersebut berhak untuk menjalankan *service* atau tidak.

Berikut ini adalah langkah-langkah untuk mengkonfigurasi *keystone*, Hal pertama yang harus dilakukan adalah melakukan instalasi paket dari *keystone* yang terdapat pada *repository*.

```
sudo apt-get install keystone python-keystone python-
keystoneclient
```

Selanjutnya meng-*edit* skrip pada */etc/keystone/keystone.conf* agar dapat berfungsi untuk memberitahukan kepada *keystone*, bahwa *database* yang digunakan adalah *mysql* berserta *user* dan *password* serta memberitahukan kemana database harus disimpan. Berikut skrip yang diubah:

```
admin_token = ADMIN
```

menjadi

```
admin_token = admin
```

dan

```
connection = sqlite:////var/lib/keystone/keystone.db
```

menjadi

```
connection =
mysql://keystonedbadmin:keystonesecret@192.168.180.1/keystone
```

Keystone juga dapat dikonfigurasi dengan *database*. Berikut cara untuk mengkonfigurasi *keystone* dengan database :

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE DATABASE keystone;'
```

Untuk membuat *user* yang berkuasa atas *database keystone* dapat dijalankan perintah berikut ini :

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE USER
keystonedbadmin;'
```

Cara untuk mengkonfigurasi *password* pada *keystonedbadmin* sebagai berikut:

```
sudo mysql -uroot -pmygreatsecret -e "SET PASSWORD FOR
'keystonedbadmin'@'%' = PASSWORD('keystonesecret');"
```

Kemudian administrator harus memperbolehkan *user* "*keystonedbamin*" mengakses seluruh elemen database "*keystone*". Untuk memperbolehkannya dengan perintah berikut:

```
sudo mysql -uroot -pmygreatsecret -e "GRANT ALL PRIVILEGES ON
keystone.* TO 'keystonedbadmin'@ '%' identified by
'keystonesecret';"
```

Untuk mengaktifkan konfigurasi *user* pada *mysql database* dengan perintah sebagai berikut:

```
flush privileges;
```

Selanjutnya melakukan sinkronasi *keystone* dengan *database*. Maka menggunakan perintah berikut:

```
sudo keystone-manage db_sync
```

Kemudian menambahkan beberapa skrip agar *keystone* dapat berjalan. Skrip berikut dapat juga ditambahkan pada *~/.bashrc* agar setiap *login*, skrip diatas dapat di eksekusi secara langsung, sehingga tidak perlu lagi menjalankan perintah tersebut setiap kali *keystone* dijalankan.

```
export SERVICE_ENDPOINT="http://localhost:35357/v2.0"
export SERVICE_TOKEN=admin
```

a. Membuat *Tenant*

Dalam penelitian ini penulis membuat dua buah *tenant* yaitu *admin* dan *service*. *Tenant* dapat diciptakan dengan perintah berikut:

```
keystone tenant-create --name admin
keystone tenant-create --name service
```

b. Membuat *Users*

Pembuatan *user* untuk masing-masing *service* bertujuan agar *service* tersebut dapat berjalan dengan baik. Berikut adalah cara untuk menciptakan *user* dengan *keystone*:

```
keystone user-create --name admin --pass admin --email
admin@foobar.com
keystone user-create --name nova --pass nova --email
nova@foobar.com
keystone user-create --name glance --pass glance --email
glance@foobar.com
keystone user-create --name swift --pass swift --email
swift@foobar.com
```

c. Membuat *Roles*

Untuk menjalankan *OpenStack* perlu dibuat *role* untuk menentukan *user* mana yang dapat mengakses seluruh *service*. Pada penelitian ini dibuat dua buah *rule* yaitu *admin* dan *service*. Berikut adalah cara untuk membuat *role* dengan *keystone*:

```
keystone role-create --name admin
keystone role-create --name Member
```

d. Penambahan *roles* untuk *users* pada *tenant*

Secara garis besar *syntax* untuk menambahkan *role* adalah sebagai berikut:

```
keystone user-role-add --user $USER_ID --role $ROLE_ID --tenant_id
$TENANT_ID
```

Selanjutnya menambahkan *role admin* kepada *user admin* dengan *tenant admin* dengan perintah berikut:

```
keystone user-role-add --user 4ceff7f497604884922b99999249a5d3 –
role ab02dd90cae140f8a2670e6f43630f79 --tenant_id
1fc660d9bf8f41b18ac498cf4a18a261
```

Selanjutnya menambahkan *role admin* kepada masing-masing *service* yaitu *nova*, *glance*, *swift* dengan *tenant service* dengan perintah berikut:

```
keystone user-role-add --user ba3c7255b7df41339383aef042cde2fa --
role ab02dd90cae140f8a2670e6f43630f79 --tenant_id
1e4214cb7f9847408da5547e3c65e150
keystone user-role-add --user cad8c6084ecd40118ece0c2d94df9cf5 --
role ab02dd90cae140f8a2670e6f43630f79 --tenant_id
1e4214cb7f9847408da5547e3c65e150
keystone user-role-add --user ce97035b64ea4ed9b335991c699814c5 --
role ab02dd90cae140f8a2670e6f43630f79 --tenant_id
1e4214cb7f9847408da5547e3c65e150
```

Kemudian menambahkan Member *role* karena akan digunakan oleh *horizon* dan *swift*. Berikut perintahnya.

```
keystone user-role-add --user 4ceff7f497604884922b99999249a5d3 -
role 5009d6d48bc14d3c8786049d31da4fc1 --tenant_id
1fc660d9bf8f41b18ac498cf4a18a261
```

e. Membuat *service*

Ada beberapa *service* yang akan digunakan oleh *user*. Beberapa *service* tersebut adalah *nova-compute*, *nova-volume*, *glance*, *swift*, *keystone* dan *ec2*. Berikut *syntax* untuk membuat beberapa *service* diatas:

```
keystone service-create --name service_name --type service_type --
description 'Description of the service'
keystone service-create --name nova --type compute --description
'OpenStack Compute Service'
keystone service-create --name volume --type volume --description
'OpenStack Volume Service'
keystone service-create --name glance --type image --description
'OpenStack Image Service'
keystone service-create --name swift --type object-store --
description 'OpenStack Storage Service'
keystone service-create --name keystone --type identity --
description 'OpenStack Identity Service'
keystone service-create --name ec2 --type ec2 --description 'EC2
Service'
```

Untuk hasil *service* yang telah dengan melakukan perintah berikut:

```
$ keystone service-list
```

f. Membuat *endpoint*

Secara garis besar *syntax* untuk membuat *endpoint* adalah sebagai berikut:

```
keystone endpoint-create --region region_name --service_id
service_id –publicurl public_url --adminurl admin_url --
internalurl internal_url
```

Untuk membuat *endpoint* dari *nova-compute* dapat dilakukan dengan perintah berikut:

```
keystone endpoint-create --region myregion --service_id
1b66f8d977464dd0b944bf056b6c00f0 --publicurl
'http://192.168.180.1:8774/v2/$(tenant_id)s' --adminurl
'http://192.168.180.1:8774/v2/$(tenant_id)s' --internalurl
'http://192.168.180.1:8774/v2/$(tenant_id)s'
```

Kemudian membuat *endpoint* dari *nova-volume* dengan perintah sebagai berikut:

```
keystone endpoint-create --region myregion --service_id
9dd00bd106404c01b53d638416d8f99a --publicurl
'http://192.168.180.1:8776/v1/$(tenant_id)s' --adminurl
'http://192.168.180.1:8776/v1/$(tenant_id)s' --internalurl
'http://192.168.180.1:8776/v1/$(tenant_id)s'
```

Kemudian membuat *endpoint* dari *glance* dengan perintah sebagai berikut:

```
keystone endpoint-create --region myregion --service_id
7c657b8ca37148bca6387af7d22eef34 --publicurl
'http://192.168.180.1:9292/v1' --adminurl
'http://192.168.180.1:9292/v1' --internalurl
'http://192.168.180.1:9292/v1'
```

Kemudian membuat *endpoint* dari *swift* dengan perintah sebagai berikut:

```
keystone endpoint-create --region myregion --service_id
5c7b7ca2c3944fde8315b6622996c83b --publicurl
'http://192.168.180.1:8080/v1/AUTH_$(tenant_id)s' --adminurl
'http://192.168.180.1:8080/v1' --internalurl
'http://192.168.180.1:8080/v1/AUTH_$(tenant_id)s'
```

Kemudian membuat *endpoint* dari *keystone* dengan perintah sebagai berikut:

```
keystone endpoint-create --region myregion --service_id
e8eeed522a9345d3ad3aa9c2682981e9 --publicurl
http://192.168.180.1:5000/v2.0 --adminurl
http://192.168.180.1:35357/v2.0 --internalurl
http://192.168.180.1:5000/v2.0
```

Kemudian membuat *endpoint* dari *ec2* dengan perintah sebagai berikut:

```
keystone endpoint-create --region myregion --service_id
0e57a0a39bc74bcf903b2f4521b28da5 --publicurl
http://192.168.180.1:8773/services/Cloud --adminurl
http://192.168.180.1:8773/services/Admin --internalurl
http://192.168.180.1:8773/services/Cloud
```

3. Glance

Glance berfungsi untuk mengatur sistem operasi yang akan dijalankan. Biasanya *glance* akan digunakan untuk mengatur *file storage* yang digunakan oleh sistem operasi yang berjalan. Berikut ini adalah cara bagaimana mengkonfigurasi *glance* beserta cara sinkronisasi dengan *database* yang digunakan. Untuk meng-*install glance* cukup menjalankan perintah berikut ini:

```
sudo apt-get install glance glance-api glance-client glance-common
glance-registry python-glance
```

a. Mengkonfigurasi *Database* Dengan *Glance*

Secara *default glance* menggunakan *sqlite* sebagai databasenya. Namun dapat juga mengkonfigurasi *glance* dengan database lainnya yang dikehendaki. Pada penelitian ini nama database yang akan digunakan adalah "*glance*" dengan *user* "*glancedbadmin*" dan *password* "*glancesecret*".

b. Konfigurasi *MySQL* Dengan *Glance*

Pertama yang harus dilakukan untuk mengintegrasikan *glance* dengan *mysql* adalah dengan membuat terlebih dahulu nama *database* yang akan digunakan dengan perintah berikut ini:

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE DATABASE glance;'
```

Selanjutnya membuat user yang berkuasa atas database *glance* dengan perintah berikut ini:

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE USER glancedbadmin;'
```

Kemudian mengkonfigurasi *password* pada *glancedbadmin* dengan perintah berikut ini:

```
sudo mysql -uroot -pmygreatsecret -e "SET PASSWORD FOR
'glancedbadmin'@'%' = PASSWORD('glancesecret');"
```

Selanjutnya administrator harus memperbolehkan *user* "*glancedbamin*" mengakses seluruh *elemen* database "*glance*". Berikut perintahnya:

```
sudo mysql -uroot -pmygreatsecret -e "GRANT ALL PRIVILEGES ON
glance.* TO 'glancedbadmin'@ '%' identified by 'glancesecret';"
```

Kemudian mengaktifkan konfigurasi *user* pada *mysql database* dengan perintah berikut ini:

```
flush privileges;
```

Kemudian melakukan perubahan skrip pada */etc/glance/glance-registry.conf*.

```
sql_connection =
```

Menjadi

```
sql_connection =
mysql://glancedbadmin:glancesecret@192.168.180.1/glance
```

c. Konfigurasi *Postgresql* Dengan *Glance*

Untuk mengkonfigurasi *postgresql* dapat dilakukan dengan perintah berikut:

```
sudo su - postgres
psql
CREATE user glancedbadmin;
ALTER user glancedbadmin with password 'glancesecret';
CREATE DATABASE glance;
GRANT ALL PRIVILEGES ON database glance TO glancedbadmin
identified by glancesecret;
\q
Exit
```

Kemudian melakukan perubahan skrip pada */etc/glance/glance-registry.conf*.

```
sql_connection =
```

menjadi

```
sql_connection =
postgresql://glancedbadmin:glancesecret@192.168.180.1/glance
```

Kemudian *restart* pada *glance* dengan perintah berikut:

```
sudo /etc/init.d/ glance-registry restart
```

d. Konfigurasi Pada *Glance*

Untuk mengkonfigurasi *glance* agar dapat berjalan sesuai dengan penelitian ini, maka perlu mengubah beberapa parameter. Pertama melakukan perubahan skrip pada */etc/glance/glance-api-paste.ini*.

```
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%
```

menjadi

```
admin_tenant_name = service
admin_user = glance
admin_password = glance
```

Kemudian melakukan perubahan skrip pada */etc/glance/glance-registry-paste.ini* seperti pada */etc/glance/glance-api-paste.ini*.

Kemudian melakukan perubahan skrip pada */etc/glance/glance-registry.conf*.

```
sql_connection =
```

menjadi

```
sql_connection =
mysql://glancedbadmin:glancesecret@10.10.10.2/glance
```

Karena pada penelitian ini penulis menggunakan keystone, maka harus menambahkan skrip seperti berikut pada */etc/glance/glance-registry.conf*

```
[paste_deploy]
flavor = keystone
```

Kemudian menambahkan skrip pada /etc/glance/glance-api.conf seperti berikut:

```
[paste_deploy]
flavor = keystone
```

Selanjutnya membuat skema pada *mysql* untuk *database glance* dengan perintah berikut:

```
sudo glance-manage version_control 0
sudo glance-manage db_sync
```

Kemudian *restart glance service* dengan perintah berikut:

```
sudo restart glance-api
sudo restart glance-registry
```

Kemudian untuk dapat melihat hasilnya, maka harus menjalankan perintah seperti berikut:

```
export SERVICE_TOKEN=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin
export OS_AUTH_URL="http://localhost:5000/v2.0/"
export SERVICE_ENDPOINT=http://localhost:35357/v2.0
```

Selanjutnya dapat menambahkanya ke *~/.bashrc* untuk menjadikannya otomatis saat login. Untuk melihat apakah konfigurasinya sudah benar, dapat menjalankan perintah:

```
glance index
```

Jika konfigurasinya benar, maka tidak akan mendapatkan *output* apapun karena belum mempunyai *image* saat pertama kali. *Glance index* akan menampilkan *output* jika telah meng*upload image* kepada *glance database*.

4. *Nova*

Langkah pertama yang harus dilakuan adalah melakukan penginstalan paket nova yang sudah tersedia di repository. Berikut adalah cara menginstall paket dari *nova*.

```
sudo apt-get install nova-api nova-cert nova-compute nova-compute-
kvm nova-doc nova-network nova-objectstore nova-scheduler nova-
```

```
volume rabbitmq-server novnc nova-consoleauth python-pip qemu-kvm
kvm-pxe
```

Selanjutnya melakukan installasi *euca2ools* sebagai CLI (*Command Line Interface*) yang dapat digunakan sebagai jembatan untuk mengkonfigurasi *nova*.

```
sudo apt-get install -y euca2ools
```

Kemudian melakukan installasi *unzip* yang dapat digunakan untuk mengekstrak file yang berekstensi *zip*.

```
Sudo apt-get install -y unzip
```

a. Konfigurasi *Database* Dengan *Nova*

Nova memerlukan *database* untuk menyimpan setiap data serta perubahan yang terjadi padanya. Pada penelitian ini nama *database* yang akan digunakan adalah "*nova*" dengan *user* "*novadbadmin*" dan *password* "*novasecret*".

b. MySQL

Hal pertama yang harus dilakukan untuk mengintegrasikan *nova* dengan *mysql* adalah dengan membuat terlebih dahulu nama *database* yang akan digunakan dengan perintah berikut ini:

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE DATABASE nova;'
```

Selanjutnya membuat *user* dengan nama "novadbadmin" dengan perintah sebagai berikut:

```
sudo mysql -uroot -pmygreatsecret -e 'CREATE USER novadbadmin;'
```

Kemudian membuat *password* "*novasecret*" untuk "*novadbadmin*" dengan perintah sebagai berikut:

```
sudo mysql -uroot -pmygreatsecret -e "SET PASSWORD FOR
'novadbadmin'@'%' = PASSWORD('novasecret');"
```

Kemudian memberikan akses penuh "*novadbadmin*" kepada *database* "*nova*" dengan perintah sebagai berikut:

```
sudo mysql -uroot -pmygreatsecret -e "GRANT ALL PRIVILEGES ON
nova.* TO 'novadbadmin'@'%' identified by 'novasecret' ;"
```

Setelah itu menganktifkan konfigurasi *user* pada *mysql database* dengan perintah sebagai berikut:

```
flush privileges;
```

c. *Postgresql*

Hal pertama yang harus dilakukan untuk mengintegrasikan *nova* dengan *postgresql* adalah dengan melakukan konfigurasi seperti berikut:

```
sudo su - postgres
psql
CREATE user novadbadmin;
ALTER user novadbadmin with password 'novasecret';
CREATE DATABASE nova;
GRANT ALL PRIVILEGES ON database nova TO novadbadmin;
\q
Exit
```

Selanjutnya menambahkan beberapa baris skrip berikut ini pada */etc/postgresql/9.1/main/pg_hba.conf*

```
host all all 192.168.180.0/24 md5
host all all 192.168.4.0/24 md5
```

Hal tersebut harus dilakukan agar *postgresql* dapat menerima koneksi dari *network* 192.168.180.0 dan 192.168.4.0. Kemudian melakukan *restart* pada *postgresql* dengan perintah:

```
sudo /etc/init.d/postgresql restart
```

d. File Konfigurasi *Nova*

Sebelum melakukan perubahan pada skrip konfigurasi *nova*, alangkah baiknya jika dilakukan *backup* terlebih dahulu terhadap file konfigurasi tersebut dengan cara sebagai berikut:

```
sudo cp /etc/nova/nova.conf /etc/nova/nova.conf.backup
```

Kemudian melakukan perubahan skrip pada */etc/nova/nova.conf* menjadi sebagai berikut:

```
--dhcpbridge_flagfile=/etc/nova/nova.conf
--dhcpbridge=/usr/bin/nova-dhcpbridge
--logdir=/var/log/nova
--state_path=/var/lib/nova
--lock_path=/run/lock/nova
--allow_admin_api=true
--use_deprecated_auth=false
--auth_strategy=keystone
--scheduler_driver=nova.scheduler.simple.SimpleScheduler
--s3_host=192.168.180.1
--ec2_host=192.168.180.1
--rabbit_host=192.168.180.1
--cc_host=192.168.180.1
--nova_url=http://192.168.180.1:8774/v1.1/
--routing_source_ip=192.168.180.1
--glance_api_servers=192.168.180.1:9292
--image_service=nova.image.glance.GlanceImageService
--iscsi_ip_prefix=192.168.4
--sql_connection =
mysql://novadbadmin:novasecret@192.168.180.1/nova
--ec2_url=http://192.168.180.1:8773/services/Cloud
--keystone_ec2_url=http://192.168.180.1:5000/v2.0/ec2tokens
--api_paste_config=/etc/nova/api-paste.ini
--libvirt_type=kvm
--libvirt_use_virtio_for_bridges=true
--start_guests_on_host_boot=true
--resume_guests_state_on_host_boot=true
# vnc specific configuration
--novnc_enabled=true
--novncproxy_base_url=http://192.168.180.1:6080/vnc_auto.html
--vncserver_proxyclient_address=192.168.180.1
--vncserver_listen=192.168.180.1
# network specific settings
--network_manager=nova.network.manager.FlatDHCPManager
--public_interface=eth0
--flat_interface=eth0
--flat_network_bridge=br100
--fixed_range=192.168.4.2/28
--floating_range=192.168.180.0/28
--network_size=32
--flat_network_dhcp_start=192.168.4.1
--flat_injected=False
```

```
--force_dhcp_release=True
--iscsi_helper=tgtadm
--connection_type=libvirt
--root_helper=sudo nova-rootwrap
--verbose=True

--
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
--my_ip=192.168.180.1
```

Jika menggunakan *database* berupa *postgresql*, maka harus melakukan perubahan baris skrip berikut:

```
--sql_connection=mysql://novadbadmin:novasecret@192.168.180.1/nova
```

menjadi

```
--
sql_connection=postgresql://novadbadmin:novasecret@192.168.180.1/n
ova
```

Dan jika menggunakan *database sqlite*, maka harus melakukan perubahan baris skrip berikut:

```
--sql_connection=mysql://novadbadmin:novasecret@192.168.180.1/nova
```

menjadi

```
--
sql_connection=sqlite://novadbadmin:novasecret@192.168.180.1/nova
```

Selanjutnya melakukan penginstalan *iscsitarget* dan *iscsitarget-dkms* dengan perintah berikut ini:

```
sudo apt-get -y install iscsitarget iscsitarget-dkms
```

Untuk mengaktifkan *iscsitarget* sebagai *daemon*, maka harus melakukan perubahan parameter *false* menjadi *true* pada */etc/default/iscsitarget*. Kemudian *restart* dengan perintah berikut:

```
sudo /etc/init.d/iscsitarget restart
```

e. Konfigurasi *Physical Volume*

Physical volume adalah sebuah *hard drive virtual* yang disiapkan untuk menampung semua data yang berasal dari *openstack*. Drive ini juga akan digunakan oleh *virtual operating system* yang berjalan sebagai *virtual drive user* masing-masing. Berikut *cara* untuk menyiapkan *virtual drive*.

Hal pertama yang harus dilakukan adalah *user* harus dalam posisi sebagai *root*. Untuk menandai bahwa user *root* atau bukan bisa dilihat di akhir *console* / *terminal* dengan tanda "#" apabila user bertindak sebagai *user* biasa maka akhir *console* / *terminal* akan berupa tanda "$". Jika user berada pada level biasa maka harus menjalankan perintah *sudo -i*, *sudo su*, atau *sudo bash*. Kemudian melakukan beberapa langkah berikut:

```
#echo 1 > /sys/class/scsi_device/0\:0\:0\:0/device/rescan
#fdisk /dev/sda
```

Maka hasil dari fdisk adalah sebagai berikut:

```
The number of cylinders for this disk is set to 9137.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)
Command (m for help):
```

Kemudian mengetikan huruf "p" untuk melihat daftar partisi *drive* yang ada dan "n" untuk membuat partisi baru.

```
Command (m for help): p

Disk /dev/sda: 75.1 GB, 75161927680 bytes
255 heads, 63 sectors/track, 9137 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Device Boot Start End Blocks Id System
/dev/sda1 * 1 14 104391 83 Linux
Partition 1 does not end on cylinder boundary.
/dev/sda2 14 4178 33450007+ 8e Linux LVM
/dev/sda3 4178 8354 33549073+ 8e Linux LVM
Create your new LVM partition in the free space:

Command (m for help): n
```

```
Command action
e extended
p primary partition (1-4)
```

Kemudian memilih huruf “p” dan cukup tekan *enter* untuk *default* instalasi.

```
Selected partition 4
First cylinder (8355-9137, default 8355):
Using default value 8355
Last cylinder or +size or +sizeM or +sizeK (8355-9137, default
9137):
Using default value 9137
```

Selanjutnya menekan huruf “t” untuk menentukan tipe *hard drive* dan mengisikan huruf “8e” untuk menentukan *hard drive* tersebut sebagai *linux* LVM.

```
Command (m for help): t
Partition number (1-4): 4
Hex code (type L to list codes): 8e
```

Kemudian mengetik huruf “w” agar *drive* yang telah dibuat bisa digunakan. Setelah hal tersebut dilakukan, maka dapat dibuat sebuah *virtual drive* dengan perintah seperti berikut:

```
# pvcreate /dev/sda4
```

Jika mengalami error sebagai berikut:

```
Device /dev/sda4 not found (or ignored by filtering)
```

Maka solusinya adalah sebagai berikut berikut:

```
# partprobe -s
```

Maka akan menghasilkan output sebagai berikut:

```
/dev/sda: msdos partitions 1 2 3 4
```

Kemudian melakukan *pvcreate* sekali lagi, jika masih *error* yang sama maka buat harus melakukan *resize* partisi dari awal atau menggunakan *external hard drive*. Jika dengan *partprobe* berhasil, maka akan menghasilkan *output* sebagai berikut :

```
# pvcreate /dev/sda4
Physical volume "/dev/sda4" successfully created
```

Setelah itu membuat *volume group* dengan nama *nova-volumes* dengan cara berikut:

```
sudo vgcreate nova-volumes /dev/sda4
```

Setelah berhasil, kemudian melakuan perubahah kepemilikan serta hak akses terhadap */etc/nova* dengan perintah berikut:

```
sudo chown -R root:nova /etc/nova
sudo chmod 644 /etc/nova/nova.conf
sudo chmod +x /etc/nova/ -R
```

Kemudian melakukan pergantian beberapa baris skrip pada */etc/nova/api-paste.ini*

```
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%
```

menjadi

```
admin_tenant_name = service
admin_user = nova
admin_password = nova
```

Kemudian melakukan perintah berikut ini untuk membuat skema *nova* pada *database* yang sudah dibuat.

```
sudo nova-manage db sync
```

Berikut cara untuk menyediakan *range* dari IP yang akan digunakan sebagai IP *private* dari *virtual* sistem operasi yang akan berjalan nantinya.

```
nova-manage network create private --
fixed_range_v4=192.168.4.32/27 --num_networks=1 -bridge=br100 --
bridge_interface=eth1 --network_size=32
```

Setelah itu melakukan beberapa perintah berikut:

```
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin
export OS_AUTH_URL="http://localhost:5000/v2.0/"
```

Kemudian siapkan *range* dari IP *public* yang akan digunakan sebagai IP kedua agar *virtual* sistem operasi yang berjalan dapat diakses oleh *public*. Cara menyediakan IP *public* sebagai berikut:

```
sudo nova-manage floating create --ip_range=192.168.180.0/28
```

Hal ini dapat diartikan akan disediakannya IP *public* sebanyak 16 IP yang dimulai dari 192.168.180.1 – 192.168.180.13. Hal ini dapat dilihat dari CIDR yang ada yaitu /28.

5. *OpenStack Dashboard*

Openstack juga menyediakan *web* untuk melihat semua informasi tentang *service*-nya. Untuk dapat menjalankan *web* tersebut, maka harus melakukan beberapa instalasi paket *openstack dashboard* yang sudah tersediah di *repository* berikut:

```
sudo apt-get install openstack-dashboard && sudo
/etc/init.d/apache2 restart
```

Jika berjalan dengan sempurna maka *web* dari openstack dapat diakses hanya dengan mengetikan IP dari *server* yang telah dipasangi *openstack dashboard* pada *browser*.

6. SWIFT

Langkah pertama yang harus dilakuan adalah melakukan penginstalan paket *swift* yang sudah tersedia di *repository*. Berikut adalah cara menginstall paket dari *swift*.

```
sudo apt-get install swift swift-proxy memcached swift-account
swift-container swift-object xfsprogs curl python-pastedeploy
```

a. Konfigurasi *Physical Device* Pada *Swift*

Ada dua cara yang dapat dilakukan untuk mengkonfigurasi swift. Dengan menggunakan *Physical Device* (*Partition*) *as a storage* dan *Loopback Device* (*File*) *as storage*.

1. Physical Device (Partition) as a storage

   Hal pertama yang harus dilakukan adalah menyiapkan sebuah *hard drive* yang nantinya akan digunakan sebagai *storage* oleh *swift*, pada peneletian kali ini *hard drive* yang digunakan adalah */dev/sdb*, sehingga cara pengkonfigurasiannya adalah sebagai berikut :

   ```
   sudo fdisk /dev/sdb
   ```

   Kemudian membuat partisi baru dengan menggunakan *fdisk*, sehingga hasil yang didapat adalah terciptanya partisi baru dengan nama */dev/sdb1*.

2. Loopback Device (File) as storage

   Dibandingkan dengan cara pertama, cara kedua ini lebih mudah dan tidak membutuhkan *hard drive*. Disini hanya dibutuhkan sebuah *file* yang akan dijadikan *loopback* oleh *swift* sebagai *hard drivenya*. Pada penelitian ini akan dibuat sebuah *file* dengan alokasi *size* sebesar satu juta KiB (976.56 MiB), sehingga tercipta file *loopback* kurang lebih sebesar 1GiB. Ukuran file dapat disesuaikan dengan merubah nilai dari *seek*.

   ```
   sudo dd if=/dev/zero of=/srv/swift-disk bs=1024 count=0 seek=1000000
   ```

   Setelah itu langkah berikutnya membuat *xfs system*. Jika menggunakan *physical partition*, maka konfigurasinya sebagai berikut:

   ```
   sudo mkfs.xfs -i size=1024 /dev/sdb1
   sudo tune2fs -l /dev/sdb1 |grep -i inode
   ```

   Jika anda menggunakan loopback file, maka konfigurasinya sebagai berikut:

```
sudo mkfs.xfs -i size=1024 /srv/swift-disk
file /srv/swift-disk
```

Dengan hasil output:

```
swift-disk1: SGI XFS filesystem data (blksz 4096, inosz 1024,
v2 dirs)
```

Storage device harus ter*mount* secara *otomatis*, untuk itu harus dibuat *mount point*-nya dengan asumsi *mount point* adalah */mnt/sdb1*.

```
sudo mkdir /mnt/sdb1
```

Setelah itu agar ter-*mount* secara otomatis, maka harus ditambahkan beberapa baris skrip berikut pada baris terakhir dari */etc/fstab*.

Untuk physical partition

```
/dev/sdb1 /mnt/sdb1 xfs

noatime,nodiratime,nobarrier,logbufs=8 0 0
```

Untuk loopback file

```
/srv/swift-disk /mnt/sdb1 xfs
loop,noatime,nodiratime,nobarrier,logbufs=8 0 0
```

Selanjutnya melakukan *mounting /mnt/sdb1* serta membuat beberapa *mount point* dan memberi akses terhadap *user* "*swift*" dengan *grup* "*swift*".

```
sudo mount /mnt/sdb1
sudo mkdir /mnt/sdb1/1 /mnt/sdb1/2 /mnt/sdb1/3 /mnt/sdb1/4
sudo chown swift:swift /mnt/sdb1/*
sudo ln -s /mnt/sdb1/1 /srv/1
sudo ln -s /mnt/sdb1/2 /srv/2
sudo ln -s /mnt/sdb1/3 /srv/3
sudo ln -s /mnt/sdb1/4 /srv/4
sudo mkdir -p /etc/swift/object-server /etc/swift/container-
server /etc/swift/account-server /srv/1/node/sdb1
/srv/2/node/sdb2 /srv/3/node/sdb3 /srv/4/node/sdb4
/var/run/swift
sudo chown -R swift:swift /etc/swift /srv/[1-4]/
```

Setelah itu memasukan beberapa baris skrip berikut pada */etc/rc.local* sebelum baris *"exit 0"*

```
mkdir /var/run/swift
chown swift:swift /var/run/swift
```

b. Konfigurasi *Rsync*

Rsync digunakan oleh *swift* untuk menjaga konsistensi dari objek dan melakukan *update* apabila terjadi perubahan pada objek tersebut. *Rsync* juga terkonfigurasi dengan semua *storage* yang ada. Untuk mengkonfigurasi *rsync* cukup menambahkan skrip berikut pada */etc/rsyncd.conf*

```
# General stuff
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /run/rsyncd.pid
address = 127.0.0.1
# Account Server replication settings
[account6012]
max connections = 25
path = /srv/node1/
read only = false
lock file = /run/lock/account6012.lock
[account6022]
max connections = 25
path = /srv/node2/
read only = false
lock file = /run/lock/account6022.lock
[account6032]
max connections = 25
path = /srv/node3/
read only = false
lock file = /run/lock/account6032.lock
# General stuff
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /run/rsyncd.pid
address = 127.0.0.1
# Account Server replication settings
[account6012]
max connections = 25
path = /srv/node1/
read only = false
lock file = /run/lock/account6012.lock
[account6022]
max connections = 25
path = /srv/node2/
read only = false
lock file = /run/lock/account6022.lock
[account6032]
max connections = 25
path = /srv/node3/
read only = false
lock file = /run/lock/account6032.lock
```

Selanjutnya melakukan perubahan parameter *false* dengan *true pada RSYNC_ENABLE=false* di */etc/default/rsync*.

c. Konfigurasi *Swift*

Berikut adalah file konfigurasi *swift* pada */etc/swift/swift.conf* yang berisikan skrip berikut ini :

```
[swift-hash]
# random unique (preferably alphanumeric) string that can never
change (DO NOT LOSE)
swift_hash_path_suffix = xxxx
```

Dimana xxxx didapatkan dari nilai random. Nilai random bisa didapatkan dengan menjalankan perintah berikut ini :

```
od -t x8 -N 8 -A n < /dev/random
```

d. Konfigurasi *Proxy Server*

Untuk mengkonfigurasi *proxy server* pada *swift* harus dibuat *file* konfigurasi pada */etc/swift/proxy-server.conf* yang berisikan skrip berikut ini:

```
[DEFAULT]
bind_port = 8080
user = swift
swift_dir = /etc/swift
[pipeline:main]
# Order of execution of modules defined below
pipeline = catch_errors healthcheck cache authtoken keystone
proxy-server
[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true
set log_name = swift-proxy
set log_facility = LOG_LOCAL0
set log_level = INFO
set access_log_name = swift-proxy
set access_log_facility = SYSLOG
set access_log_level = INFO
set log_headers = True
account_autocreate = True
[filter:healthcheck]
```

```
use = egg:swift#healthcheck
[filter:catch_errors]
use = egg:swift#catch_errors
[filter:cache]
use = egg:swift#memcache
set log_name = cache
[filter:authtoken]
paste.filter_factory =
keystone.middleware.auth_token:filter_factory
auth_protocol = http
auth_host = 127.0.0.1
auth_port = 35357
auth_token = admin
service_protocol = http
service_host = 127.0.0.1
service_port = 5000
admin_token = admin
admin_tenant_name = service
admin_user = swift
admin_password = swift
delay_auth_decision = 0
[filter:keystone]
paste.filter_factory =
keystone.middleware.swift_auth:filter_factory
operator_roles = admin, swiftoperator
is_admin = true
```

e. Konfigurasi *Account Server*

Untuk membuat *account server* harus dibuat file konfigurasi pada */etc/swift/account-server.conf* dengan berisikan baris skrip berikut:

```
[DEFAULT]
bind_ip = 0.0.0.0
workers = 2
[pipeline:main]
pipeline = account-server
[app:account-server]
use = egg:swift#account
[account-replicator]
[account-auditor]
[account-reaper]
```

Selanjutnya menambahkan beberapa balis skrip berikut pada /etc/swift/account-server/1.conf.

```
[DEFAULT]
devices = /srv/node1
mount_check = false
bind_port = 6012
user = swift
log_facility = LOG_LOCAL2
```

```
[pipeline:main]
pipeline = account-server
[app:account-server]
use = egg:swift#account
[account-replicator]
vm_test_mode = no
[account-auditor]
[account-reaper]
```

Karena pada penelitian ini menggunakan beberapa *mount point*, maka harus dibuat beberapa *copy file* tadi menjadi beberapa bagian.

```
sudo cp /etc/swift/account-server/1.conf /etc/swift/account-
server/2.conf
sudo cp /etc/swift/account-server/1.conf /etc/swift/account-
server/3.conf
sudo cp /etc/swift/account-server/1.conf /etc/swift/account-
server/4.conf
```

Setelah itu melakukan pengeditan pada masing-masing *bind_port* menjadi 60x2 dimana x adalah jumlah dari *account server* yang diciptakan. Misal pada penelitian ini penulis mempunyai 4 acount *server* maka pada *account server* pertama *bind_port* berisikan 6012 sedangkan pada *account server* kedua berisikan 6022 dan seterusnya. Kemudian melakukan perubahan juga pada bagian *log_facility = LOG_LOCALx* dimana x adalah x+1 dari *account server* yang penulis ciptakan, misal pada *account server* pertama LOG_LOCAL adalah 2 sedangkan pada *account server* kedua LOG_LOCAL adalah 3 dan begitu seterusnya.

f. Konfigurasi *Container Server*

Konfigurasi *container server* dapat dilakukan dengan membuat file */etc/swift/container-server.conf* dengan berisikan skrip berikut ini:

```
[DEFAULT]
bind_ip = 0.0.0.0
workers = 2
[pipeline:main]
pipeline = container-server
[app:container-server]
```

```
use = egg:swift#container
[container-replicator]
[container-updater]
[container-auditor]
[container-sync]
```

Selanjutnya menambahkan beberapa balis skrip berikut pada */etc/swift/container-server/1.conf*.

```
[DEFAULT]
devices = /srv/node1
mount_check = false
bind_port = 6011
user = swift
log_facility = LOG_LOCAL2
[pipeline:main]
pipeline = container-server
[app:container-server]
use = egg:swift#container
[container-replicator]
vm_test_mode = no
[container-updater]
[container-auditor]
[container-sync]
```

Kemudian melakukan penggandaan file tersebut sama seperti pada *account server*.

```
sudo cp /etc/swift/container-server/1.conf /etc/swift/container-
server/2.conf
sudo cp /etc/swift/container-server/1.conf /etc/swift/container-
server/3.conf
sudo cp /etc/swift/container-server/1.conf /etc/swift/container-
server/4.conf
```

Setelah itu melakukan pengeditan masing-masing *bind_port* dan *log_facility* sama seperti pada *account server*.

g. Konfigurasi *Object Server*

Untuk mengkonfigurasi *object server* maka harus dibuat file */etc/swift/object-server.conf* dengan berisikan skrip berikut ini:

```
[DEFAULT]
bind_ip = 0.0.0.0
workers = 2
[pipeline:main]
pipeline = object-server
```

```
[app:object-server]
use = egg:swift#object
[object-replicator]
[object-updater]
[object-auditor]
```

Selanjutnya menambahkan beberapa balis skrip berikut pada */etc/swift/object-server/1.conf*

```
[DEFAULT]
devices = /srv/node1
mount_check = false
bind_port = 6010
user = swift
log_facility = LOG_LOCAL2
[pipeline:main]
pipeline = object-server
[app:object-server]
use = egg:swift#object
[object-replicator]
vm_test_mode = no
[object-updater]
[object-auditor]
```

Kemudian melakukan penggandaan file tersebut sama seperti pada *account server* diatas.

```
sudo cp /etc/swift/object-server/1.conf /etc/swift/object-
server/2.conf
sudo cp /etc/swift/object-server/1.conf /etc/swift/object-
server/3.conf
sudo cp /etc/swift/object-server/1.conf /etc/swift/object-
server/4.conf
```

Setelah itu melakukan pengeditan masing-masing *bind_port* dan *log_facility* sama seperti pada *account server*.

h. Membangun *Swift Rings*

Rings adalah *component* yang sangat penting pada *swift*. Untuk membangun *rings* harus menjalankan perintah berikut ini:

```
pushd /etc/swift
sudo swift-ring-builder object.builder create 18 3 1
sudo swift-ring-builder object.builder add z1-127.0.0.1:6010/sdb1
1
sudo swift-ring-builder object.builder add z2-127.0.0.1:6020/sdb2
1
```

```
sudo swift-ring-builder object.builder add z3-127.0.0.1:6030/sdb3
1
sudo swift-ring-builder object.builder add z4-127.0.0.1:6040/sdb4
1
sudo swift-ring-builder object.builder rebalance
sudo swift-ring-builder container.builder create 18 3 1
sudo swift-ring-builder container.builder add z1-
127.0.0.1:6011/sdb1 1
sudo swift-ring-builder container.builder add z2-
127.0.0.1:6021/sdb2 1
sudo swift-ring-builder container.builder add z3-
127.0.0.1:6031/sdb3 1
sudo swift-ring-builder container.builder add z4-
127.0.0.1:6041/sdb4 1
sudo swift-ring-builder container.builder rebalance
sudo swift-ring-builder account.builder create 18 3 1
sudo swift-ring-builder account.builder add z1-127.0.0.1:6012/sdb1
1
sudo swift-ring-builder account.builder add z2-127.0.0.1:6022/sdb2
1
sudo swift-ring-builder account.builder add z3-127.0.0.1:6032/sdb3
1
sudo swift-ring-builder account.builder add z4-127.0.0.1:6042/sdb4
1
sudo swift-ring-builder account.builder rebalance
```

Dengan catatan apabila mendapatkan *error "unable to find container-sync config section in /etc/swift/container-server.conf"* cukup menambahkan [*container-sync*] pada */etc/swift/container-server.conf* dan jika mendapatkan error *"unable locate config for object-expirer"* maka harus menambahkan skrip berikut ini pada file */etc/swift/object-expirer.conf*.

```
[DEFAULT]
# swift_dir = /etc/swift
# user = swift
# You can specify default log routing here if you want:
# log_name = swift
# log_facility = LOG_LOCAL0
# log_level = INFO
# log_address = /dev/log
# You can enable default statsD logging here if you want:
# log_statsd_host = localhost
# log_statsd_port = 8125
# log_statsd_default_sample_rate = 1
# log_statsd_metric_prefix =
[object-expirer]
# interval = 300
# auto_create_account_prefix = .
# report_interval = 300
[pipeline:main]
pipeline = catch_errors cache proxy-server
```

```
[app:proxy-server]
use = egg:swift#proxy
# See proxy-server.conf-sample for options
[filter:cache]
use = egg:swift#memcache
# See proxy-server.conf-sample for options
[filter:catch_errors]
use = egg:swift#catch_errors
# See proxy-server.conf-sample for options
```

i. Menjalankan *Servis Swift*

Untuk menjalankan *swift* cukup dengan perintah *sudo swift-init main start*. Kemudian jalankan *rest api* dengan perintah :

```
sudo swift-init main start
sudo swift-init rest start
```

B. *Server* 2 (*Node Controller*)

Hal yang pertama dilakukan adalah melakukan penginstalan *server* seperti dijelaskan pada lampiran 1. Kemudian melakukan konfigurasi jaringan agar dapat terhubung ke internet dan dapat melakukan penginstalan paket yang dibutuhkan.

Konfigurasi jaringan seharusnya sudah dilakukan pada saat menginstall *Ubuntu server*, tetapi jika terlewatkan untuk mengkonfigurasi jaringan dapat dikonfigurasi manual pada */etc/network/interfaces* dengan menggunakan *editor* seperti *nano, vi, vim*, dsb. Berikut skrip untuk meng-konfigurasi jaringan pada *server* 2:

```
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static
      address           192.168.180.2
      netmask           255.255.255.240
      broadcast         192.168.180.15
      gateway           192.168.180.14
      dns-nameservers   222.124.29.226
```

Kemudian melakukan *restart* untuk mengaktifkan konfigurasi yang sudah dibuat dengan perintah:

```
$ sudo /etc/init.d/networking restart
```

Langkah berikutnya adalah bagaimana mensinkronkan waktu antar *kedua cloud controller*. Apabila waktu antar *cloud* tidak sama, maka *cloud* tidak dapat berjalan dengan sempurna. Untuk melakukan hal ini diperlukan *Network Time Protocol* (NTP). Berikut adalah langkah untuk mengkonfigurasi NTP *server*. Pertama install NTP *server terlebih dahulu*.

```
$ sudo apt-get install –y ntp
```

Kemudian meng-*edit* */etc/ntp.conf* dan tambahkan beberapa konfigurasi dibawah ini.

```
server 192.168.180.1
```

Langkah selanjutnya adalah melakukan instalalasi paket nova-compute yang sudah ada didalam repository. Berikut cara melakukan instalasinya:

```
sudo apt-get install -y nova-compute
```

Selanjutnya melakukan penambahan baris skrip berikut pada */etc/nova/nova.conf*

```
--dhcpbridge_flagfile=/etc/nova/nova.conf
--dhcpbridge=/usr/bin/nova-dhcpbridge
--logdir=/var/log/nova
--state_path=/var/lib/nova
--lock_path=/run/lock/nova
--allow_admin_api=true
--use_deprecated_auth=false
--auth_strategy=keystone
--scheduler_driver=nova.scheduler.simple.SimpleScheduler
--s3_host=192.168.180.1
--ec2_host=192.168.180.1
--rabbit_host=192.168.180.1
--cc_host=192.168.180.1
--nova_url=http://192.168.180.1:8774/v1.1/
--routing_source_ip=192.168.180.1
--glance_api_servers=192.168.180.1:9292
--image_service=nova.image.glance.GlanceImageService
```

```
--iscsi_ip_prefix=192.168.4
--sql_connection =
mysql://novadbadmin:novasecret@192.168.180.1/nova
--ec2_url=http://192.168.180.1:8773/services/Cloud
--keystone_ec2_url=http://192.168.180.1:5000/v2.0/ec2tokens
--api_paste_config=/etc/nova/api-paste.ini
--libvirt_type=kvm
--libvirt_use_virtio_for_bridges=true
--start_guests_on_host_boot=true
--resume_guests_state_on_host_boot=true
# vnc specific configuration
--novnc_enabled=true
--novncproxy_base_url=http://192.168.180.1:6080/vnc_auto.html
--vncserver_proxyclient_address=192.168.180.1
--vncserver_listen=192.168.180.1
# network specific settings
--network_manager=nova.network.manager.FlatDHCPManager
--public_interface=eth0
--flat_interface=eth0
--flat_network_bridge=br100
--fixed_range=192.168.4.2/28
--floating_range=192.168.180.0/28
--network_size=32
--flat_network_dhcp_start=192.168.4.1
--flat_injected=False
--force_dhcp_release=True
--iscsi_helper=tgtadm
--connection_type=libvirt
--root_helper=sudo nova-rootwrap
--verbose=True

--
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
--my_ip=192.168.180.2
```

C. Pengaturan *Image* Pada *OpenStack*

Pada pengaturan *image* akan dijeaskan bagaimana sistem operasi *Ubuntu* dapat berjalan, tidak hanya *Ubuntu* yang bisa berjalan pada *openstack cloud*. Penulis bisa mengkonfigurasi semua jenis sistem operasi yang ada, karena *cloud* dirancang dengan sangat *flexible*. Hal ini memungkinkan para *developer cloud* untuk mengkonfigurasi *private cloud* mereka sesuai dengan *rule-rule* yang diterapkan pada perusahaan mereka masing-masing.

1. Pembuatan *Linux Image*

Langkah pertama yang harus dilakukan yaitu pembuatan sebuah *drive* / *hard disk* yang nantinya akan digunakan untuk penempatan sistem operasi yang akan di *install*. Cara membuat *image* sebagai berikut :

```
kvm-img create –f qcow2 server.img 10G
```

Dimana *qcow2* adalah jenis *filesystem*, sedangkan *server.img* adalah nama *image* dan 10G adalah besar kapasitas *hard disk* yang disediakan sebesar 10 *GigaByte*.

2. Instalasi Sistem Operasi

Cara melakukan instalasi sistem operasi tidak jauh berbeda dengan bagaimana menginstal sistem operasi yang biasa dilakukan pada PC (*Personal Computer*), Cuma perbedaannya adalah anda cukup meng-*install* satu kali saja kemudian selanjutnya dapat digunakan secara berulang-ulang.

Untuk sistem operasi *Ubuntu* anda dapat men*download image* yang berada pada http://release.ubuntu.com dengan menggunakan *wget* atau *downloader* lainnya atau anda dapat membuat *file iso* dari CD atau DVD *drive*. Setelah file iso tersedia maka anda cukup menjalankan perintah berikut dengan asumsi file isonya bernama Ubuntu-10.04-desktop.iso :

```
sudo kvm -m 256 -cdrom ubuntu-12.04-server-amd64.iso -drive
file=server.img,if=virtio,index=0 -boot d -net nic -net user -
nographic -vnc :0
```

Kemudian kembali pada PC *desktop* anda yang terhubung satu jaringan dengan *server* dan jalan kan *vncviewer* kemudian masukan IP dari *server* dimana anda menjalankan perintah diatas, pada penelitian kali ini IP dari server tersebut adalah 192.168.180.1 sehinga pada *vncviewer* IP yang dimasukan adalah:

```
192.168.180.1:0
```

Atau dapat menjalankan dari *terminal* Ubuntu dengan perintah sebagai berikut:

```
vncviewer 192.168.180.1:0
```

Dimana :0 adalah *index* dari *vncvewer* yang memungkinkan anda menjalankan *multiple remoter desktop*.

Setelah muncul *desktop* dari *file iso Ubuntu* tadi maka hal selanjutnya yang harus anda lakukan adalah melakukan *update system* dan meng*install ssh* pada *client* agar nantinya *client* yang anda buat dapat diremote dari jarak jauh menggunakan protokol *ssh* (*secure shell*).

```
sudo apt-get update && sudo apt-get dist-upgrade –y && sudo apt-
get install openssh-server –y
```

Selanjutnya menghapus *file /etc/udev/rules.d/70-persistent-net.rules* agar ketika sistem operasi dijalankan tidak terjadi masalah saat *cloud server* menambahkan IP secara automatis terhadap *client* yang dijalankan.

```
sudo rm –vrf /etc/udev/rules.d/70-persistent-net.rules
```

Setelah selesai dilakukan, maka selanjutnya mematikan *client* tersebut dengan perintah berikut:

```
sudo shutdown –P now
```

Tahap selanjutnya adalah meng-*upload file img* yang sudah dipasangi sistem operasi *Ubuntu* kepada aplikasi *glance* dengan perintah:

```
glance add name="<nama image>" is_public=true container_format=ovf
disk_format=qcow2 < <nama file>.img
```

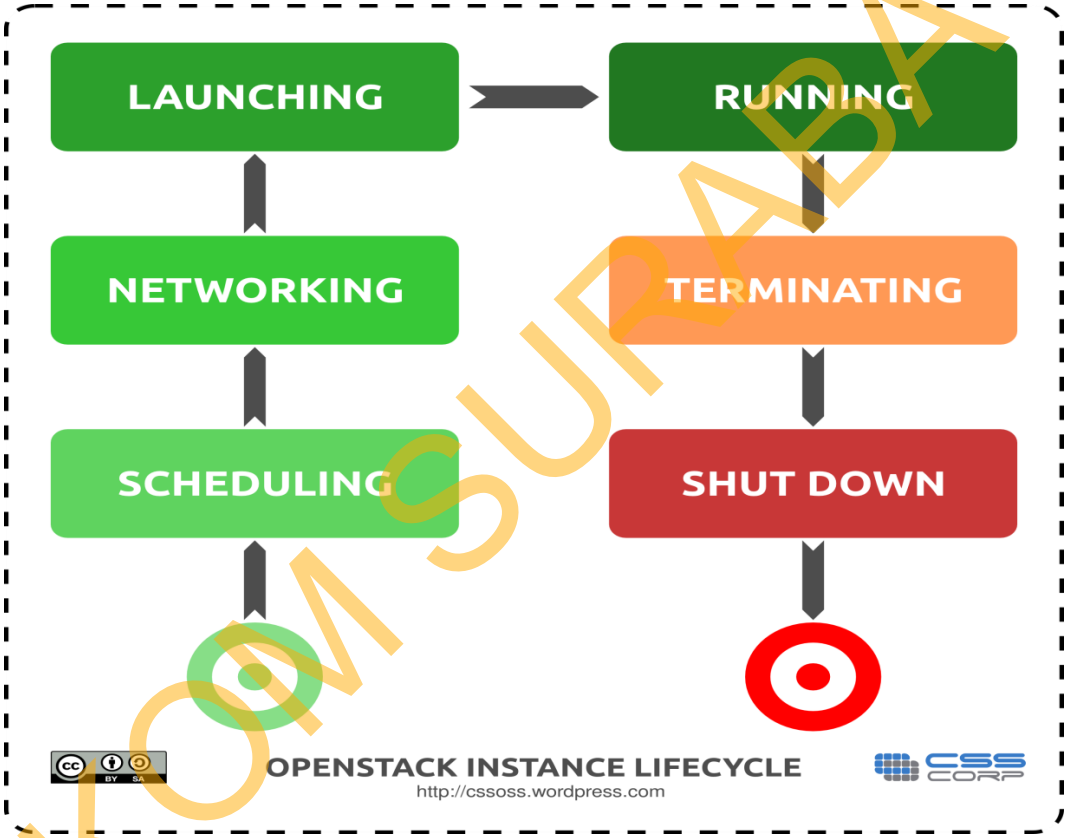
Pada penelitian ini digunakan ubuntu10.04-desktop sebagai nama client dan *server.img* sebagai nama *image*-nya maka perintah yang harus dijalankan adalah:

```
glance add name="Ubuntu-10.04-desktop" is_public=true
container_format=ovf disk_format=qcow2 < server.img
```

Kemudian tunggu proses sampai selesai, untuk melihat apakah *file image* sudah ter*upload* anda dapat mengguanakan skrip *glance index* untuk melihatnya.

D. *Instance*

Instance adalah nama lain dari *client* yang sedang *berjalan*, *instance* mempunyai *siklus* yang dapat digambarkan sebagai berikut:



Gambar 3.6 Siklus *Instance OpenStack* (Jha, OpenStack Beginner's Guide)

Pada Gambar 3.6 dapat dilihat bahwa awal mula *instance* dijalankan dia berada pada *fase scheduling*, pada tahap ini *instace* melakukan *booting operating system* yang sudah ditentukan pada awal tadi, kemudian tahap selanjutnya adalah *networking* pada tahap ini *instance* akan diberikan IP *address* berdasarkan konfigurasi pada aplikasi *nova*, setelah itu tahap berikutnya adalah *launching* pada tahap ini *instance* sudah siap untuk dijalankan selanjutnya untuk menuju pada *fase*

running instance harus di-*reboot* untuk menerapkan setting yang sudah dilakukan pada tahap-tahap sebelumnya, dan tahap selanjutnya adalah *running* dimana *instance* sudah berjalan dan siap untuk digunakan, setelah itu tahap selanjutnya adalah *terminating* yaitu tahap dimana *instance* sudah selesai digunakan dan dalam proses dimatikan, dan tahap terakhir adalah *shutdown*, dimana *instance* sudah benar-benar dalam keadaan sudah mati dan tidak dapat digunakan lagi.

Ada dua cara untuk menjalankan instance yaitu:

1. *Openstack Command Line Tools*

Hal pertama yang harus dilakukan adalah membuat sebuah *key* yang akan digunakan oleh *ssh* untuk mengenali komputer yang digunakan. Berikut cara untuk membuat *key*:

```
ssh-keygen
cd .ssh
nova keypair-add --pub_key id_rsa.pub mykey
```

Pada penelitian ini dibuat *key* dengan nama *mykey*. Untuk melihatnya dapat menggunakan perintah berikut:

```
server@controller:/var/www$ nova keypair-list
+-------+-------------------------------------------------+
|  Name |                   Fingerprint                   |
+-------+-------------------------------------------------+
| mykey | 6e:97:01:96:26:1a:f4:7d:14:70:f6:39:fc:e4:73:15 |
+-------+-------------------------------------------------+
```

Gambar 3.7 *Keypair List*

Jika ingin menghapus *key* yang sudah ada, dapat menggunakan cara seperti berikut ini:

```
nova keypair-delete mykey2
```

Setelah membuat *key*, maka langkah selanjutnya adalah menjalankan *instance*. Berikut adalah cara untuk menjalankan *instance*:

```
nova boot --flavor 1 --image 9bab7ce7-7523-4d37-831f-c18fbc5cb543
--key_name mykey myinstance
```

Untuk melihat *flavor* / *resource* yang disediakan dapat menggunakan perintah seperti berikut:

```
nova flavor-list
```

Kemudian dapat memastikan apakah *instance* sudah tercipta atau belum dengan perintah seperti berikut:

```
nova list
```

Untuk melakukan *reboot* terhadap *instance* dapat menggunakan perintah berikut:

```
nova reboot $id_dari_instance
```

Untuk menghapus *instance* dapat menggunakan perintah berikut:

```
nova delete  $id_dari_instance
```

Untuk melihat daftar *console* dapat menggunakan perintah berikut:

```
nova console-log $nama_dari_instance
```

Setelah *instance* dalam keadaan *running* / *active* dapat diakses melalui SSH dengan perintah sebagai berikut :

```
ssh -i <private_key> username@<ip_address>
```

misal: administrator mempunyai *instance* yang berjalan dengan *user client* dan IP 192.168.4.2 dengan *ssh key* adalah *mykey* maka cara untuk mengakses *instance* tersebut adalah :

```
ssh -i mykey client@192.168.4.2
```

2. *Openstack WEB Base*

Jika menjalankan *instance* melalui *web base*, maka berikut adalah langkah-langkahnya:

a. Login pada *web openstack*

b. Masuk pada *tab project*

c. Masuk pada pada *tab Access & Security*

d. Klik pada *tab create keypair*

e. Isikan *keypair name* dengan sembarang nama kemudian klik *create keypair*

f. Anda akan diarahkan untuk men*download* sebuah *file* yang berekstensi *.pem, download file* tersebut karena nantinya *file* tersebut akan anda gunakan untuk mengakses *intance*.

g. Setelah proses *download* selesai sekarang beralihlah ke *tab security groups*, klik *edit rules*

h. Pada *tab add rule* isikan dengan dua *rules* berikut:

   a. Pada IP *protokol* pilih TCP, *from port* 22, *to port* 22, *source group* CIDR, CIDR 0.0.0.0/0, *rules* ini berfungsi untuk mengizinkan akses *ssh* kepada *instance*. Kemudian klik *add rule*.

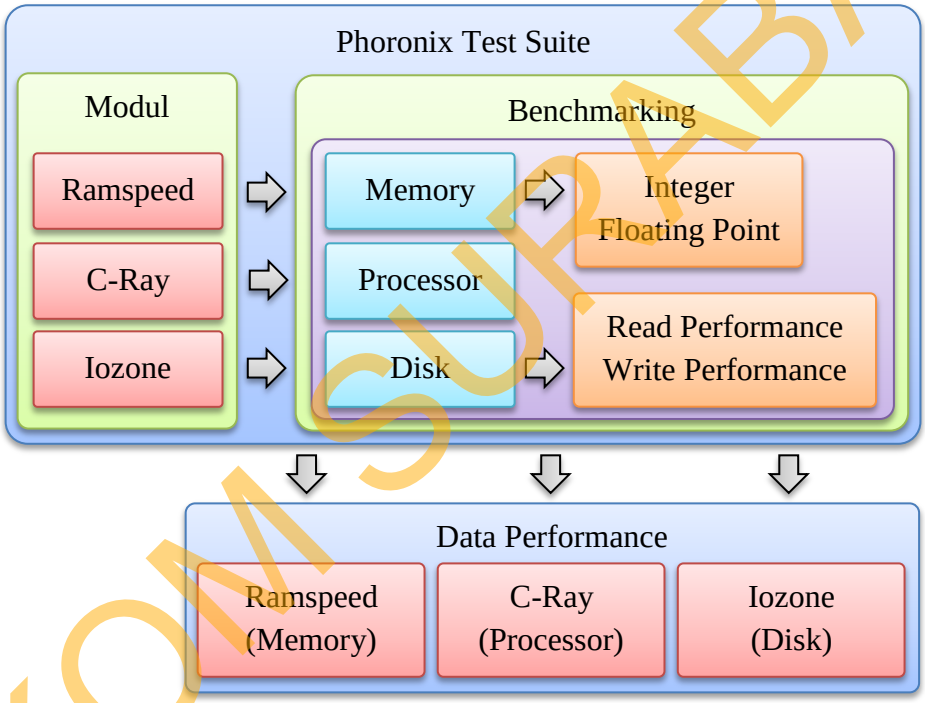
   b. Pada IP *protokol* pilih ICMP, *from port* -1, *to port* -1, *source group* CIDR, CIDR 0.0.0.0/0, rules ini berfungsi untuk mengizinkan *packet icmp / echo request* kepada *instance*. Kemudian klik *add rule*.

i. Pindah ke *tab images & Snapshoot*, kemudian cari *tab images*, klik *launch* pada *image* yang anda kehendaki, kemudian anda akan dihadapkan pada tabel *launch instance*, isikan *server name* dengan nama yang anda kehendaki. Pada *tab* flavor pilih dari daftar yang tersedia yang sesuai dengan *rule* / konfigurasi anda, pada penelitian kali ini digunakan *flavor m1.tiny*. Pada *tab keypair* pilih *keypair* yang sudah anda buat sebelumnya, kemudian klik *launch instance*, setelah itu tunggu sampai *instance* berada pada *state active*, mungkin untuk menuju *state active* akan sedikit lebih lama.

j. Untuk mengakses *intance* caranya sama dengan yang ada pada CLI tutorial.

3.3 Data Performance

Untuk mendapatkan data *performance* sistem dari *Eucalyptus* dan *OpenStack*, penulis perlu melakukan pengujian dengan melakukan *benchmarking* terhadap beberapa variabel dari *Eucalyptus* dan *OpenStack*. Untuk lebih jelasnya dapat dilihat pada Gambar 3.12.



Gambar 3.8 Diagram Blok Pengujian *Performance* Sistem

Dari Gambar 3.8 dapat dilihat bahwa untuk melakukan benchmarking *performance Eucalyptus* dan *OpenStack*, penulis menggunakan *phoronix test suite* sebagai *tools* untuk mendapatkan data *performance*.

Phoronix test suite merupakan sebuah aplikasi pemanggil modul yang digunakan untuk menjalan beberapa modul yang telah diintegrasikan dengan *phoronix test suite*. Alasan mengapa penulis menggunakan *phoronix test suite*, karena aplikasi ini mendukung banyak modul untuk melakukan *benchmarking*

terhadap sebuah sistem operasi. Selain itu aplikasi ini juga banyak direkomendasikan oleh para administrator dan juga para praktisi teknologi informasi.

Aplikasi ini mempunyai banyak modul yang bisa di gunakan untuk mengambil sampel data dari *cloud client* yang berjalan, namun pada penelitian ini penulis hanya menggunakan modul *ramspeed* untuk menguji *memory* (RAM), c-ray untuk menguji *processor* dan *iozone* untuk menguji *disk* (*harddisk*).

Data *performance* merupakan data hasil dari pengujian *benchmarking* yang dilakukan dengan menggunakan *benchmarking tools* yaitu *phoronix test suite*. Data ini juga nantinya akan dilakukan pengujian apakah data dari pengujian terhadap *Eucalyptus* dan *OpenStack* ini layak untuk dilakukan analisa statistika.

Pengujian untuk mendapatkan data *performance* ini dilakukan sebanyak 30 kali pada masing-masing *cloud computing*. Dengan data sebanyak 30 ini diharapkan dapat menjadi standar pengambilan data untuk melakukan perhitungan statistika.

Data *performance* yang diambil dari *Eucalyptus* dan *OpenStack cloud* ada tiga bagian. Yang pertama adalah data *performance* dari *memory* atau sering disebut dengan RAM. Untuk pengambilan data performance pada *memory* akan dilakukan oleh *phoronix test suite* dengan menggunakan modul *ramspeed*. Penulis melakukan pengujian dengan dua variabel yaitu dengan pengujian tipe data *floating point* dan tipe data *integer*. Dimana tipe data *integer* dianggap mewakili perhitungan yang dilakukan oleh sistem operasi sedangkan tipe data *floating point* dianggap mewakili data-data perkantoran.

Sedangkan bagian kedua adalah mengambil data *performance* dari *disk* pada instance yang berjalan. Pada penelitian kali ini penulis menggunakan modul *iozone* dari *phoronix test suite* untuk mendapatkan data *performance disk*. Pada bagian ini penulis melakukan pengujian dengan dua variabel yaitu *read performance* dan *write performance*. Dimana kedua variabel tersebut dianggap telah mewakili *performance* dari *hardisk*.

Kemudian bagian ketiga adalah pengambilan data *performance* dari *processor* akan dilakukan oleh *phoronix test suite* dengan menggunakan modul *c-ray*. Modul tersebut akan dijalankan pada sisi *client* / *instance* yang sedang berjalan.

3.3.1 Pengambilan Data *Performance*

Pengambilan data pada *Eucalyptus* dan OpenStack dilakukan pada *client* yang berjalan. Pada penelitian ini penulis mengunakan sistem operasi *Ubuntu* sebagai *instance* dengan menggunakan *memory* 512 *MB* dan *harddisk* 10 GB. Kemudian penulis akan melakukan pengambilan data terhadap *instance* tersebut dengan membagi menjadi tiga bagian yaitu pengujian terhadap *memory* (RAM), *disk* (*harddisk*) serta *processor*.

Berikut ini adalah cara pengambilan data *performance* dari *memory* (RAM) dengan menggunakan modul *ramspeed* dari *phronix test suite*. Hal pertama yang harus dilakukan penulis adalah menginstall modul *ramspeed* seperti Gambar 3.9.

```
ramspeed:
    1 File Needed / 0.07 MB
    Downloading: ramspeed-2.6.0.tar.gz                [0.07MB]
    Downloading ..............................................
    Installation Size: 0.72 MB
    Installing Test
```

Gambar 3.9 Penginstalan Modul *Ramspeed*

Kemudian menjalankan modul *ramspeed* pada *phoronix test suite* seperti Gambar 3.10.

```
eucalyptus@eucalyptus-cloud:~$ phoronix-test-suite run ramspeed
PHP Notice:  Use of undefined constant IS_NVIDIA_LINUX - assumed 'IS_NVIDIA_LINUX' in /usr/share/phoronix-test-suite/pts-core
/objects/phodevi/components/phodevi_system.php on line 1052

========================================
Test Configuration: RAMspeed
========================================


Type:

1: Copy
2: Scale
3: Add
4: Triad
5: Average
6: Test All Options

Enter Your Choice:
```

Gambar 3.10 Menjalankan Modul *Ramspeed*

Setelah itu penulis dapat memilih jenis *variabel benchmark* apa yang akan dijalankan. Pada Gambar 3.11 penulis memilih *test all options* sebagai *variabel benchmark*.

```
Benchmark:

1: Integer
2: Floating Point
3: Test All Options

Enter Your Choice: 3

Would you like to save these test results (Y/n)? Y
Enter a name to save these results: Y
Enter a unique name for this test run: ram01
```

Gambar 3.11 Pemilihan Variabel *Benchmark Memory*

Kemudian modul *ramspeed* akan berjalan dan menghasilkan hasil berupa variabel *integer* dan *floating point* seperti pada Gambar 3.12 dan 3.13.

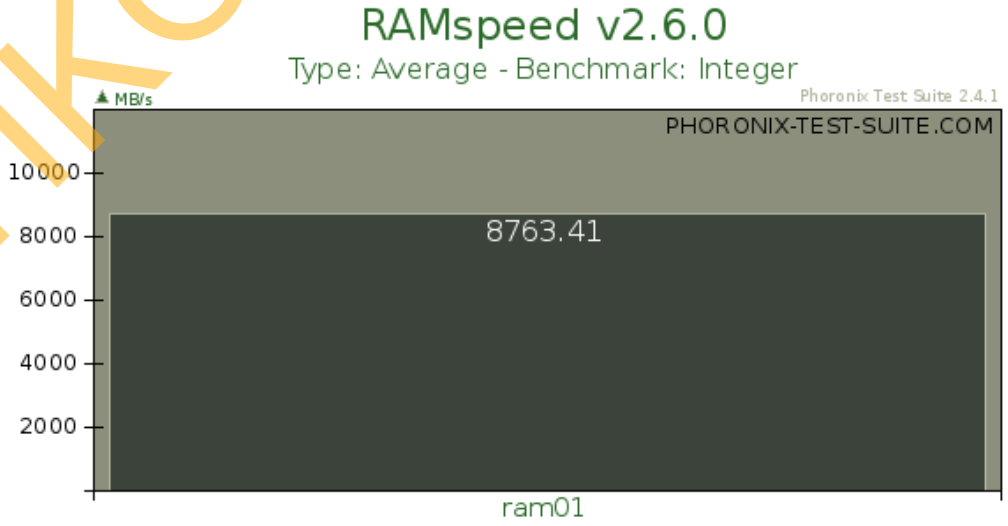
```
RAMspeed:
      ramspeed [Type: Average - Benchmark: Integer]
      Test Run 1 of 2
      Estimated Time Remaining: 4 Minutes
      Estimated Test Run-Time: 2 Minutes
      Expected Trial Run Count: 1
            Started Run 1 @ 10:54:53

      Test Results:
            8763.41

      Average: 8763.41 MB/s
```

Gambar 3.12 Hasil *Benchmark Memory* Variabel *Integer*

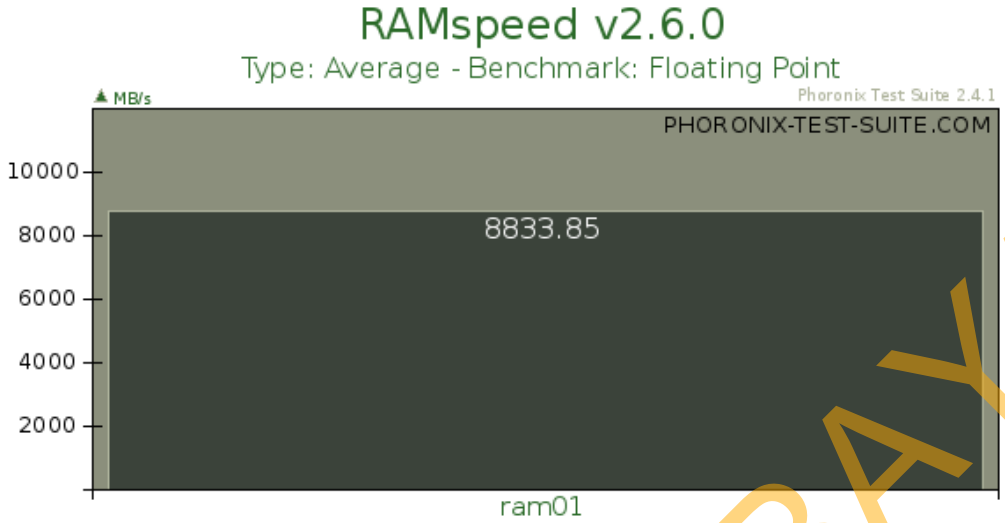
```
RAMspeed:
      ramspeed [Type: Average - Benchmark: Floating Point]
      Test Run 2 of 2
      Estimated Test Run-Time: 2 Minutes
      Expected Trial Run Count: 1
            Started Run 1 @ 10:56:48

      Test Results:
            8833.85

      Average: 8833.85 MB/s
```

Gambar 3.13 Hasil *Benchmark Memory* Variabel *Floating Point*

Hasil dari *benchmark memory* bisa dilihat melalui *web base* seperti Gambar 3.14 dan 3.15. Hasil *benchmark* ini juga dapat di-*upload* pada situs resmi *phoronix test suite*.



Gambar 3.14 Hasil *Web Benchmark Memory* Variabel *Integer*



Gambar 3.15 Hasil *Web Benchmark Memory* Variabel *Floating Point*

Setelah melakukan pengujian terhadap *memory*, maka langkah selanjutnya adalah melakukan pengujian terhadap *harddisk*. Untuk melakukan pengujian terhadap *harddisk* pada *instance* yang berjalan, penulis menggunakan *phoronix test suite* dengan modul *iozone*.

Berikut adalah cara untuk pengambilan data *performance disk* (*harddisk*) dengan menggunakan modul *iozone*. Hal pertama yang harus dilakukan penulis adalah menginstall modul *iozone* seperti Gambar 3.16.

```
eucalyptus@eucalyptus-cloud:~$ phoronix-test-suite install iozone
PHP Notice:  Use of undefined constant IS_NVIDIA_LINUX - assumed 'IS_NVIDIA_LINUX' in /usr/share/phoronix-test-suite/pts-core
/objects/phodevi/components/phodevi_system.php on line 1052

============================================================
WARNING: It appears that the Phoronix Test Suite is already running.
For proper results, only run one instance at a time.
============================================================


    iozone:
        1 File Needed / 1.50 MB
        Downloading: iozone3_323.tar                    [1.50MB]
        Estimated Download Time: 1m .........................
        Installation Size: 2.3 MB
        Installing Test
```

Gambar 3.16 Penginstalan Modul *Iozone*

Kemudian menjalankan modul *iozone* pada *phoronix test suite* seperti Gambar 3.17.

```
eucalyptus@eucalyptus-cloud:~$ phoronix-test-suite run iozone
PHP Notice:  Use of undefined constant IS_NVIDIA_LINUX - assumed 'IS_NVIDIA_LINUX' in /usr/share/phoronix-test-suite/pts-core
/objects/phodevi/components/phodevi_system.php on line 1052

============================================================
WARNING: It appears that the Phoronix Test Suite is already running.
For proper results, only run one instance at a time.
============================================================


========================================
Test Configuration: IOzone
========================================


Size:

1: 512MB
2: 2GB
3: 4GB
4: 8GB
5: Test All Options

Enter Your Choice:
```

Gambar 3.17 Menjalankan Modul *Iozone*

Setelah itu penulis dapat memilih jenis *variabel benchmark* apa yang akan dijalankan. Pada Gambar 3.18 penulis memilih *test all options* sebagai *variabel benchmark*.

```
Disk Test:

1: Write Performance
2: Read Performance
3: Test All Options

Enter Your Choice: 3

Would you like to save these test results (Y/n)? Y
Enter a name to save these results: io01
Enter a unique name for this test run: io01
```

Gambar 3.18 Pemilihan Variabel *Benchmark Disk*

Kemudian modul *iozone* akan berjalan dan menghasilkan hasil berupa variabel *write performance* dan *read performance* seperti pada Gambar 3.19 dan 3.20.

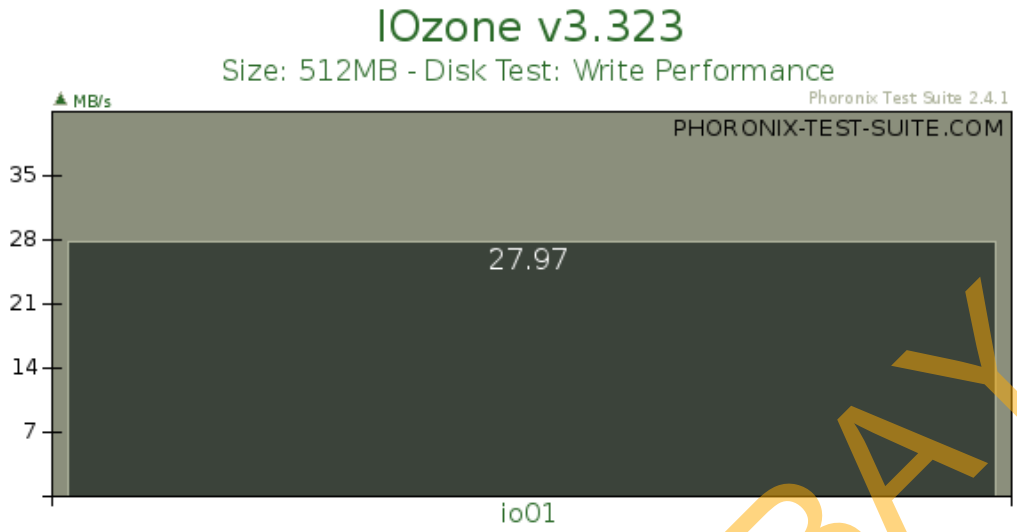
```
IOzone:
      iozone [Size: 512MB - Disk Test: Write Performance]
      Test Run 1 of 2
      Estimated Time Remaining: 7 Minutes
      Estimated Test Run-Time: 4 Minutes
      Expected Trial Run Count: 3
            Started Run 1 @ 10:58:00
            Started Run 2 @ 10:58:19
            Started Run 3 @ 10:58:38
            Started Run 4 @ 10:59:00
            Started Run 5 @ 10:59:19
            Started Run 6 @ 10:59:39

      Test Results:
            79.232421875
            77.859375
            70.04296875
            78.6025390625
            78.099609375
            84.3525390625

      Average: 78.03 MB/s
```

Gambar 3.19 Hasil *Benchmark Disk* Variabel *Write Performance*

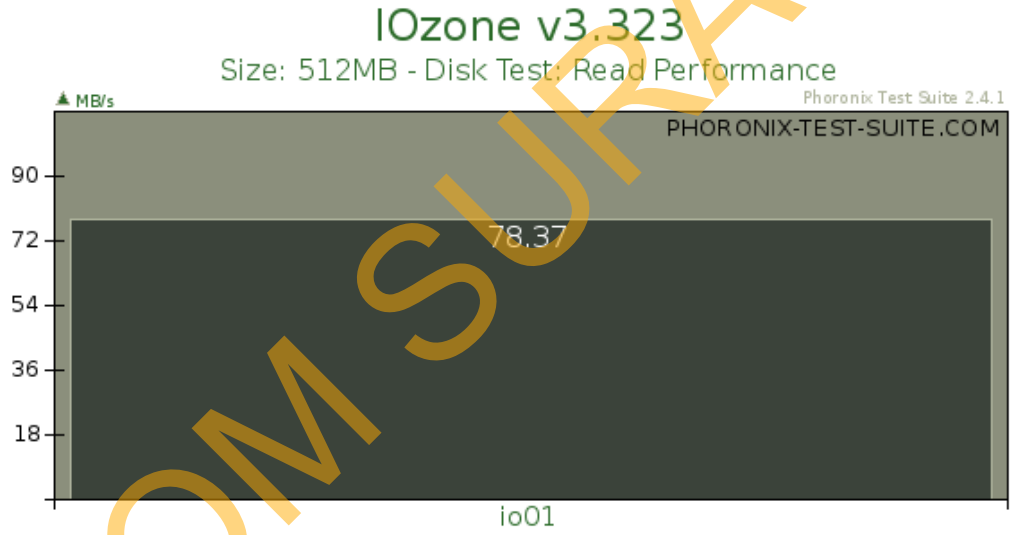
```
IOzone:
      iozone [Size: 512MB - Disk Test: Read Performance]
      Test Run 2 of 2
      Estimated Test Run-Time: 6 Minutes
      Expected Trial Run Count: 3
            Started Run 1 @ 10:41:19
            Started Run 2 @ 10:41:51
            Started Run 3 @ 10:42:23

      Test Results:
            76.396484375
            81.30078125
            77.4404296875

      Average: 78.37 MB/s
```

Gambar 3.20 Hasil *Benchmark Disk* Variabel *Read Performance*

Hasil dari *benchmark disk* bisa dilihat melalui *web base* seperti Gambar 3.21 dan 3.22. Hasil *benchmark* ini juga dapat di-*upload* pada situs resmi *phoronix test suite*.



Gambar 3.21 Hasil *Web Benchmark Disk* Variabel *Write Performance*



Gambar 3.22 Hasil *Web Benchmark Disk* Variabel *Read Performance*

Setelah step kedua selesai maka step terakhir dalam pengambilan data *performance Eucalyptus* adalah melakukan pengambilan data *performance* dari *processor*. Untuk mendapatkan data *performance* dari *processor* penulis menggunakan modul *c-ray* dari *phoronix test suite*.

Berikut ini adalah cara untuk mendapatkan data *performance processor Eucalyptus* dengan menggunakan modul *c-ray*. Hal pertama yang harus dilakukan penulis adalah menginstall modul *c-ray* seperti Gambar 3.23.

```
eucalyptus@eucalyptus-cloud:~$ phoronix-test-suite install c-ray
PHP Notice:  Use of undefined constant IS_NVIDIA_LINUX - assumed 'IS_NVIDIA_LINUX' in /usr/share/phoronix-test-suite/pts-core
/objects/phodevi/components/phodevi_system.php on line 1052

      c-ray:
            1 File Needed / 0.22 MB
            Downloading: c-ray-1.1.tar.gz                    [0.22MB]
            Estimated Download Time: 1m .............................
            Installation Size: 6 MB
            Installing Test
```

Gambar 3.23 Penginstalan Modul *C-Ray*

Kemudian menjalankan modul *c-ray* pada *phoronix test suite* seperti Gambar 3.24.

```
eucalyptus@eucalyptus-cloud:~$ phoronix-test-suite run c-ray
PHP Notice:  Use of undefined constant IS_NVIDIA_LINUX - assumed 'IS_NVIDIA_LINUX' in /usr/share/phoronix-test-suite/pts-core
/objects/phodevi/components/phodevi_system.php on line 1052

Would you like to save these test results (Y/n)? Y
Enter a name to save these results: cray01
Enter a unique name for this test run: cray01
```

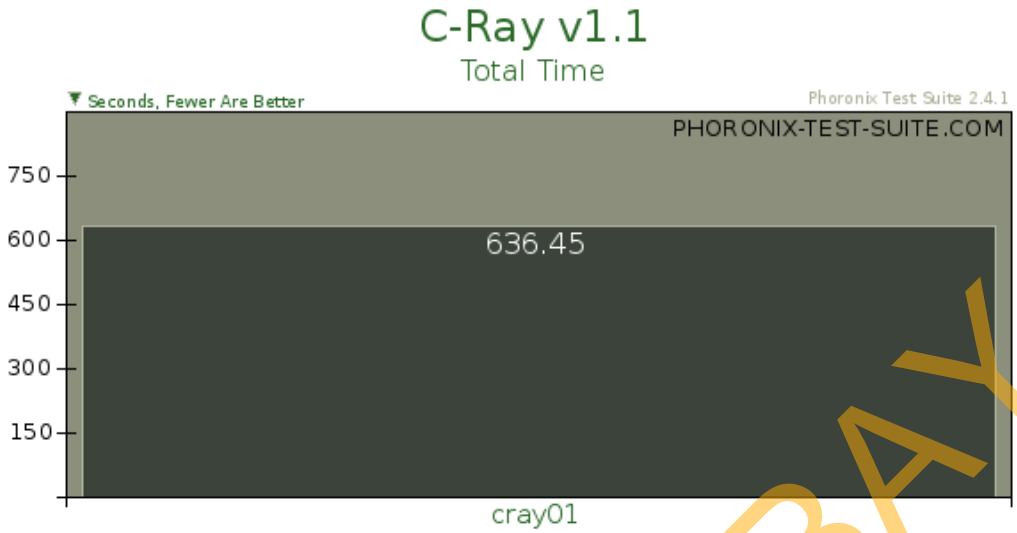
Gambar 3.24 Menjalankan Modul *C-Ray*

Kemudian modul *c-ray* akan berjalan dan menghasilkan hasil seperti pada Gambar 3.25.

```
C-Ray:
      c-ray
      Expected Trial Run Count: 3
            Started Run 1 @ 09:37:14
            Started Run 2 @ 09:47:13
            Started Run 3 @ 09:57:10
            Started Run 4 @ 10:08:03
            Started Run 5 @ 10:18:15
            Started Run 6 @ 10:29:02

      Test Results:
            596.805
            594.998
            650.197
            609.899
            645.635
            721.202

      Average: 636.45 Seconds
```

Gambar 3.25 Hasil *Benchmark Processor*

Hasil dari *benchmark processor* bisa dilihat melalui *web base* seperti Gambar 3.26. Hasil *benchmark* ini juga dapat di-*upload* pada situs resmi *phoronix test suite*.



Gambar 3.26 Hasil *Web Benchmark Processor*

Setelah melakukan tiga step diatas, maka pengambilan data *performance* telah selesai.

3.4 Analisa Statistika

Analisis statistika merupakan hasil akhir yang ingin dicapai dalam penelitian ini. Pada bagian ini yang dilakukan adalah melakukan analisis berdasarkan data *performance*. Data *performance* merupakan hasil dari *benchmarking* dari *phoronix test suite*.

Untuk menganalisa data *performance*, penulis menggunakan pengujian hipotesis karena pengujian ini didasarkan atas analisa data dan populasi data.

Pada pengujian hipotesis ini penulis menggunakan pengujian rata-rata antara data performance *Eucalyptus* dan *OpenStack*. Namun sebelum uji rata-rata dapat dilakukan, uji variasi harus dilakukan terlebih dahulu untuk mengetahui sifat dari data uji homogen atau heterogen.

Setelah uji variansi dilakukan, selanjutkan dilakukan uji hipotesis untuk mengetahui *rata-rata* dari data *performance* mana yang lebih tinggi. Pada

penelitian ini *performance* pada *cloud* dilihat berdasarkan pada nilai hasil uji hipotesis yang tinggi.