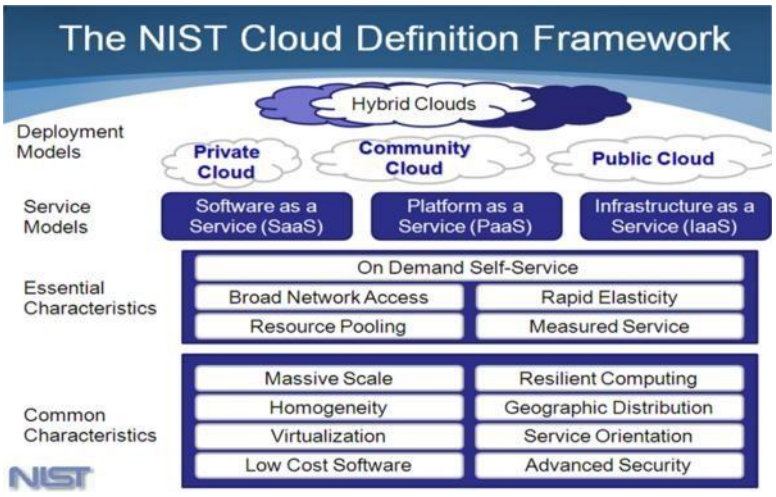
Materi Lanjut *Cloud computing*

2.1. Cloud Computing

Cloud computing adalah suatu model komputasi, dimana sumber daya seperti *processor, storage, network,* dan *software* menjadi abstrak dan diberikan sebagai layanan di jaringan/internet dengan menggunakan pola akses *remote*. Ketersediaan *on-demand* sesuai kebutuhan, mudah untuk dikontrol, dinamik dan skalabilitas yang hampir tanpa batas adalah beberapa atribut penting dari *cloud computing*. (Johnson dkk, 2010)

2.1.1. Model Layanan *Cloud Computing*

Menurut *National Institut of Science and Technology* (NIST) sebagai bagian dari Departemen Perdagangan Amerika, aspek - aspek standar *cloud*, yaitu :



Gambar 2.1. Aspek - aspek Standar *Cloud*

2.1.1.1. Model Pembayaran

Ada dua model pembayaran *Cloud Computing*, yaitu :

1. Berlangganan

   Model pembayaran ini biasanya pengguna akan membayar secara periodik bisa bulanan maupun tahunan.

2. *Pay per-use*

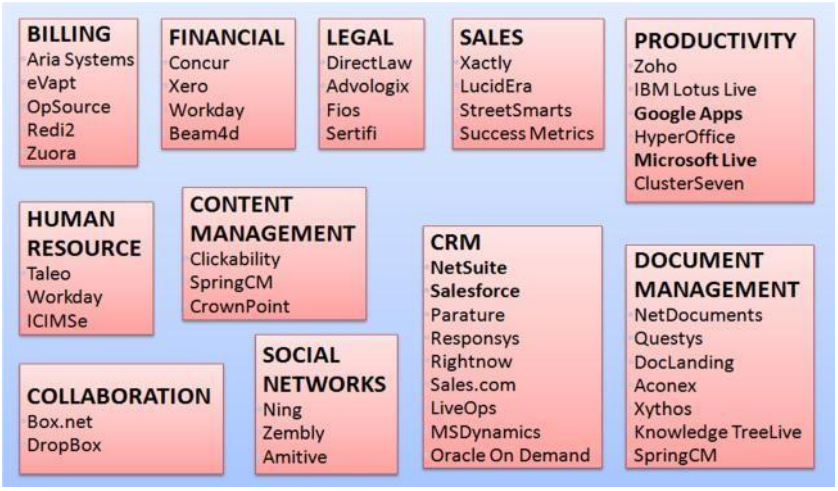
   Model pembayaran ini menguntungkan pengguna karena pengguna hanya membayar sesuai dengan layanan yang didapat (sesuai kebutuhan). (Sihotang, 2011)

2.1.1.2. Model Layanan

Ada tiga model layanan, yaitu:

1. *Software as a Services* (SaaS)

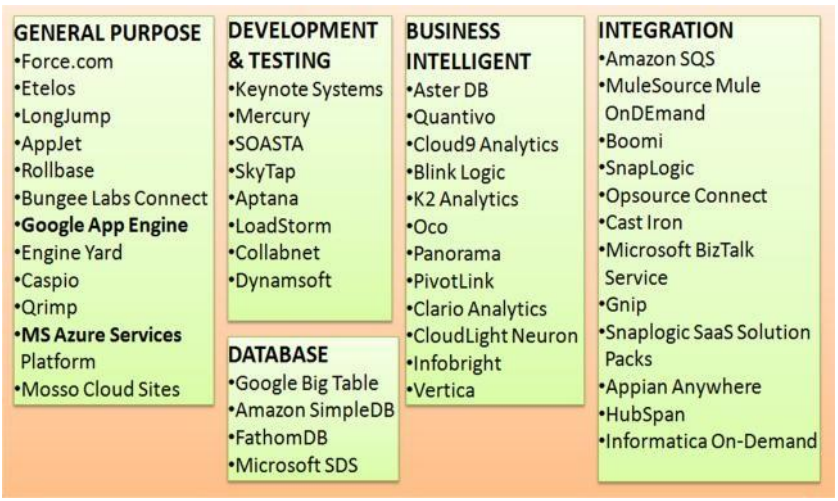
   SaaS merupakan perkembangan lanjutan dari konsep *Application Service Provider* (ASP). SaaS memiliki keterbatasan dalam pemanfaatan fitur aplikasi, karena *multi-tenant*, maka fitur-fitur biasanya bersifat umum. SaaS biasanya menyediakan fungsi untuk penyelesaian masalah, misalnya untuk *business intelegence*, *web conference*, *e-mail*, dan lain-lain. Pengguna dapat langsung menggunakan berbagai fitur yang disediakan oleh *provider*, misalnya *Google* dengan *Google Apps*-nya. Konsep SaaS ini masih terkendala pada pelanggan yang belum mendapat kendali penuh atas aplikasi yang disewa pelanggan. Selain itu, hanya fitur-fitur aplikasi yang telah disediakan oleh penyedia saja yang dapat disewa oleh pelanggan.



Gambar 2.2. Penyedia Jasa Layanan SaaS

2. *Platform as a Services* (PaaS)

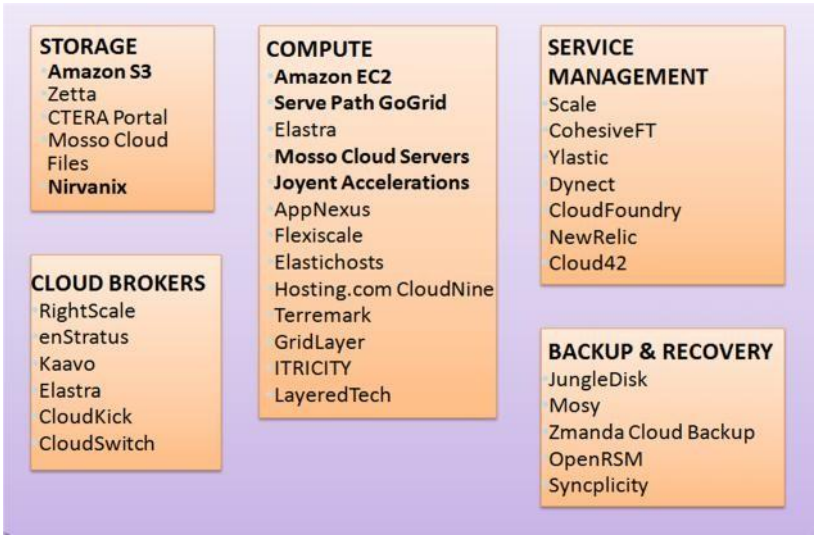
   PaaS merupakan layanan *cloud* yang menyediakan modul-modul siap pakai. Modul-modul ini dapat digunakan untuk menjalankan dan mengembangkan aplikasi yang hanya dapat berjalan di atas *platform* tersebut. Layanan ini melayani pelanggan untuk mengembangkan aplikasi baru menggunakan pengembangan API dan pengaturan konfigurasi. Pada konsep PaaS, pengguna juga tidak memiliki kendali terhadap sumber daya komputasi dasar, seperti : *memory*, *storage*, dan lain-lain, semuanya sudah diatur oleh *provider*.



Gambar 2.3. Penyedia Jasa Layanan PaaS

3. *Infrastructure as a Services (IaaS)*

   IaaS merupakan layanan yang memberikan fasilitas sewa bagi penggunanya untuk menjalankan aplikasi yang diinginkan. Pada konsep IaaS, pengguna dapat menyewa sumber daya komputasi dasar, seperti : *processing power*, *memory*, OS, dan lain-lain. IaaS memungkinkan pengguna melakukan penambahan/pengurangan kapasitas dengan mudah. IaaS menyediakan *virtual machines* dan *hardware* lainnya, serta sistem operasi yang dapat dikontrol melalui pelayanan API. (Sihotang, 2011)



Gambar 2.4. Penyedia Jasa Layanan IaaS

2.1.1.3. Jenis *Cloud*

Ada empat jenis *cloud*, yaitu:

1. *Private Cloud*

   Layanan ini memiliki infrastruktur layanan *cloud* yang dioperasikan hanya untuk sebuah perusahaan tertentu yang membutuhkan privatisasi data. Pengguna pada layanan ini biasanya berskala besar. Infrastruktur dapat dikelola sendiri atau campur tangan pihak lain (*stake holder*).

2. *Community Cloud*

   Layanan ini merupakan pengembangan dari layanan *private cloud*. Pengelolaannya juga dapat ditangani sendiri maupun pihak lain (*stake holder)*. Namun perbedaannya adalah layanan ini lebih menyajikan infrastruktur *cloud* untuk beberapa organisasi yang memiliki kepentingan yang sama, misalnya dari segi keamanan maupun visi misi yang akan dicapai.

3. *Public Cloud*

   Layanan ini lebih bersifat luas atau publik umum, dan dapat juga digunakan oleh grup perusahaan/industri besar. Layanan ini disediakan oleh *provider*.

4. *Hybrid Cloud*

   Layanan ini merupakan gabungan dari dua atau lebih infrastruktur *cloud* (*Private, Public* dan *Community*). Meskipun secara entitas tetap berdiri masing - masing, namun ternyata terdapat penghubung, yaitu dihubungkan oleh suatu mekanisme yang memungkinkan portabilitas data dan aplikasi antar *cloud*. (Sihotang, 2011)

2.1.1.4. Karakteristik *Cloud*

Ada lima karakteristik *cloud*, yaitu:

1. *On Demand Self Service*

   Pengguna dapat memesan dan mengelola layanan tanpa interaksi manusia dengan penyedia layanan. Interaksi dapat dilakukan dengan menggunakan *portal web*. Pengadaan dan perlengkapan layanan serta sumber daya yang terkait terjadi secara otomatis pada *provider cloud*.

2. *Broad Network Access*

   Layanan yang tersedia terhubung melalui jaringan *broadband*. Hal ini berguna untuk dapat mengakses *cloud* dengan baik melalui internet.

3. *Resource Pooling*

   Provider cloud memberikan layanan melalui sumber daya yang dikelompokkan pada satu atau lebih lokasi yang terdiri dari sejumlah *server multi-tenant*, yaitu mekanisme yang memungkinkan sejumlah sumber daya komputasi digunakan secara bersama-sama oleh banyak pengguna. Kebutuhan pengguna dapat secara dinamis terpenuhi oleh *provider*.

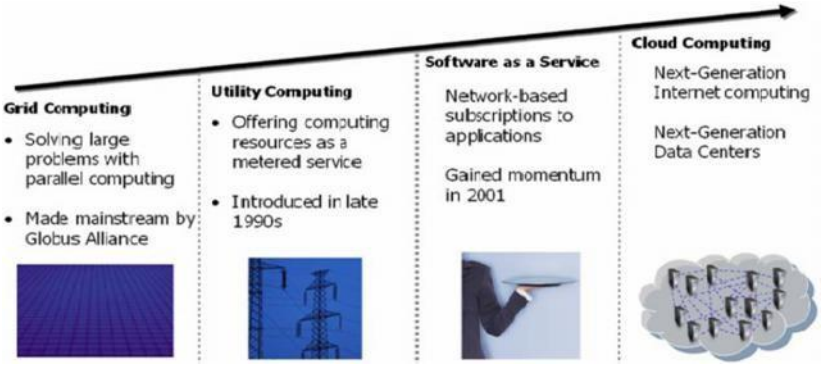
4. *Rapid Elasticity*

   Pengguna dapat membeli fasilitas *cloud* dengan jumlah tak terbatas dan dapat dibeli kapan pun diinginkan. Kapasitas komputasi yang disediakan pun dapat dengan mudah dan cepat disediakan, baik untuk penambahan maupun pengurangan kapasitas.

5. *Measured Service*

   Pengguna dapat mengetahui besar sumber daya komputasi apa saja yang telah digunakan secara transparan dengan suatu ukuran/*parameter cloud*, misalnya *storage, memory, processor, bandwidth*, aktivitas pengguna, dan sebagainya. Sehingga, pengguna tidak akan dirugikan oleh penyedia. (Sihotang, 2011)

2.1.2. Evolusi Model *Computing*



Gambar 2.5. Evolusi Model *Computing*

2.1.3. Komponen *Cloud Computing*

Komponen dasar *cloud computing* ada tiga, yaitu:

1. *Clients*

   Client dapat berupa Laptop , PC, *Mobile phone*, PDA, dan sebagainya.

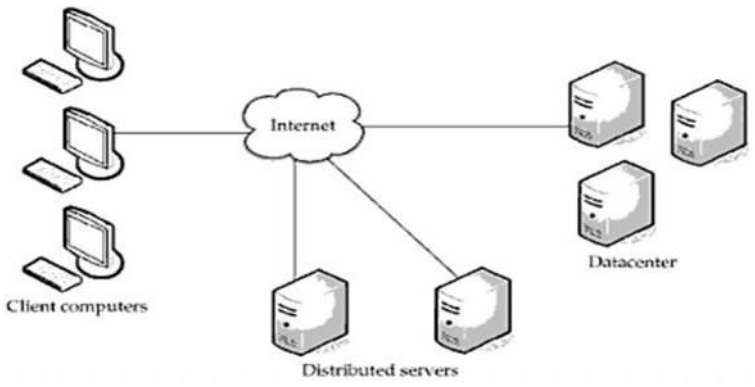
2. Data *Center*

   Data *Center* dapat berupa *hardware* yang merupakan kumpulan *server* di sebuah gedung dan juga *software* yang merupakan *virtualizing server*.

3. *Distributed Server*

   Distributed Server merupakan *server-server* yg tersebar di beberapa lokasi.

   (Velte dkk, 2010)



Gambar 2.6. Komponen *Cloud Computing*

2.1.4. Arsitektur *Cloud Computing*

Arsitektur *cloud computing* dapat dibagi ke dalam empat *layer*, yaitu:

1. *Layer Physical Hardware*

   Layer ini merupakan perangkat fisik *data center* yang di-virtualisasi untuk memberikan *platform* fleksibel untuk meningkatkan *utilisasi resources*.

2. *Layer Virtualization*

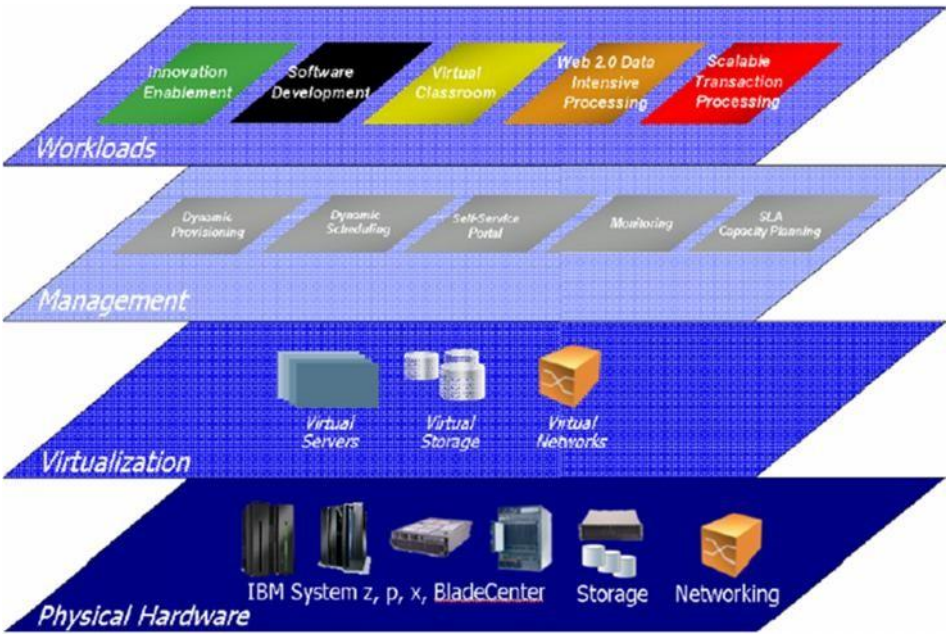
   Layer ini terdiri dari *Virtual Server, Virtual Storage* dan *Virtual Network*.

3. *Layer Management*

   Layer ini terdiri dari *Dynamic Provisioning, Dynamic Scheduling, Sel-Service Portal, Monitoring* dan *SLA Capacity Planning*.

4. *Layer Workloads*

   Layer ini terdiri dari *Innovation Enablement, Software Development, Virtual Classroom, Web 2.0 Data Intensive Processing* dan *Scalable Transaction Processing.* (Sihotang, 2011)



Gambar 2.7. Arsitektur *Cloud Computing*

2.1.5. Kelebihan *Cloud Computing*

Adapun kelebihan dari *cloud computing*, yaitu:

1. Hemat biaya.

   Penghematan biaya akan sangat terasa terutama bagi perusahaan, karena tidak perlu membayar *update* dan *upgrade hardware*, *software* maupun *maintenance*. *Cloud computing* tidak membutuhkan spesifikasi *hardware* dan *software* yang tinggi. *Software* untuk perkantoran juga biasanya akan lebih murah bahkan gratis dan jika berbayar pun maka akan tetap diuntungkan, karena hanya cukup menginstal satu saja pada *cloud*.

2. Keamanan data.

   Data yang tersimpan akan bertahan di *cloud*, sehingga tidak ada resiko kehilangan data karena kerusakan *hard disk* maupun *crash computer*. *Cloud* secara otomatis akan menduplikasi data, sehingga data dapat kembali di akses melalui komputer lain. *Cloud* juga menggunakan *Secure Sockets Layer* (SSL) sebagai sistem keamanan *cloud*.

3. Kompatibilitas.

   Dengan *cloud computing*, sistem operasi apapun dapat saling bertukar data. Kompatibilitas sistem tidak berpengaruh di *cloud computing*. Pengguna juga tidak perlu khawatir akan kompatibilitas file dokumen, misalnya antara *word* 2003 dengan *word* 2007 ataupun *open office*. Pengguna dapat berbagi data dalam bentuk file *flash* ataupun dokumen terbuka yang dapat diakses melalui *browser* apapun.

4. Kinerja lebih baik.

   Kinerja komputer akan terasa lebih cepat, karena tidak seperti PC *desktop* yang menjalankan banyak *software*, sehingga sumber daya yang digunakan tidak membebani komputer.

5. Kolaborasi lebih mudah.

   Pengguna akan dimudahkan dalam bertukar data antar komputer satu dengan yang lain, jarak tidak lagi menjadi masalah, misalnya bertukar data dengan relasi di luar kota bahkan di luar negeri.

6. Kemudahan mengakses dokumen pribadi.

   Pengguna dapat mengakses dokumen pribadinya dari komputer manapun selama terhubung ke internet dan dokumen tersimpan di *cloud*. Pengguna cukup *login* menggunakan *username* dan *password*.

7. Kemudahan *update* dan *upgrade software*.

   Pengguna tidak perlu disibukkan lagi dalam *update* dan *upgrade software*, karena *provider cloud* biasanya sudah memberikan layanan *update* dan *upgrade software* secara otomatis. Ketika pengguna mengakses *login*, maka pengguna langsung mendapat versi terbaru tanpa harus membayar biaya *upgrade* tersebut.

8. *Customization*.

   Sebelum *cloud* muncul, pengguna mengalami kesulitan dalam menyeragamkan kinerja antar komputer dan juga membutuhkan konfigurasi yang sulit. (Sihotang, 2011)

2.1.6. Kendala *Cloud Computing*

Adapun kendala *cloud computing*, yaitu:

1. *Service level*

   Terdapat kemungkinan *provider cloud* tidak akan konsisten dengan kinerja aplikasi yang ditawarkan. Pengguna pun harus memahami *service level* mengenai *transaction response time*, *data protection* dan kecepatan data *recovery*. Hal ini tentunya bagi pengguna menjadi kurang efektif dan efisien.

2. *Privacy*

   Kerahasiaan data kemungkinan akan jatuh ke pihak lain, karena perusahaan lain juga melakukan *hosting*. Hal ini dapat terjadi tanpa sepengetahuan pengguna. Seorang *hacker* dapat memanfaatkan kelemahan melalui perusahaan lain yang memiliki *hosting* sama dengan perusahaan pengguna. Kerahasiaan data perusahaan juga berpeluang untuk dilihat oleh penyedia layanan ISP. Hal ini dikarenakan *provider* memiliki akses yang sangat tinggi terhadap jaringan yang disediakan.

3. *Data ownership*

   Terdapat resiko kehilangan kepemilikan data karena kemungkinan data akan hilang apabila terjadi pemutusan pelayanan. Selain itu, apabila *server* utama rusak dan data tidak di-*backup*, maka pengguna akan kehilangan semua datanya. Pengguna juga dikhawatirkan dengan hilangnya *privacy*, karena *provider* juga berhak atas kepemilikan data pengguna. Hal ini tentunya sangat merugikan pengguna yang ingin mengakhiri kontrak dengan *provider*.

4. *Data mobility*

   Terdapat kemungkinan berbagi data antar *cloud service*. Pengguna juga dibingungkan dengan bagaimana cara mendapatkan datanya kembali jika suatu saat melakukan pemutusan layanan *cloud.* Apabila data tidak dikembalikan oleh *provider,* maka data akan jatuh ke pihak lain.

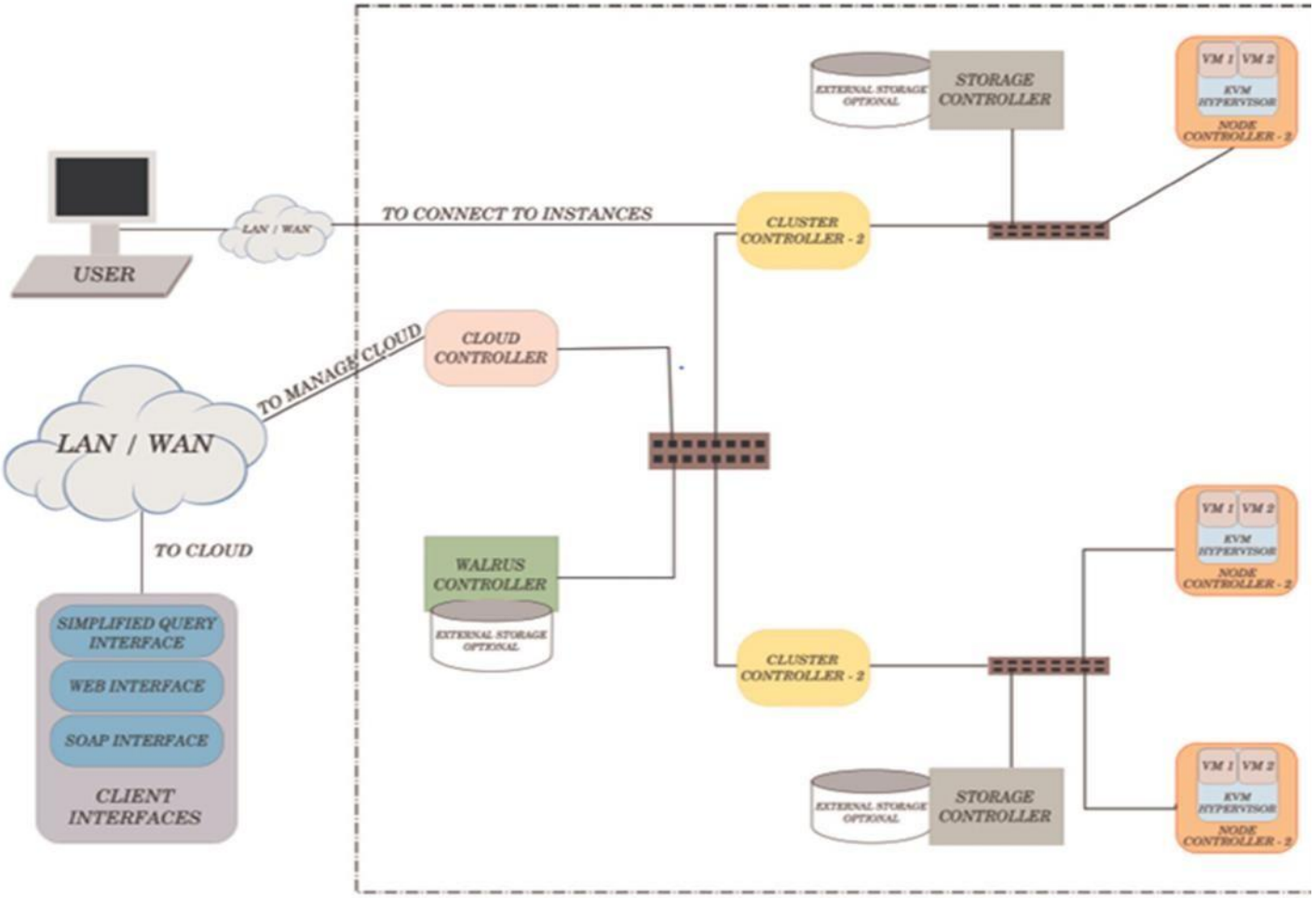
5. *Internet connection*

   Dalam mengakses *cloud* diperlukan koneksi internet, tanpa internet maka pengguna tidak bisa berkutik. Koneksi internet yang dibutuhkan pun haruslah yang stabil dan tinggi. Jika koneksinya rendah maka aplikasi dan dokumen yang berukuran besar tidak dapat diolah, dikarenakan *web app* membutuhkan banyak *bandwidth* untuk *download*. Selain itu, layanan internet khususnya di Indonesia belum optimal dalam penyediaan *bandwidth*, penyediaan layanan internet ke daerah-daerah, dan juga harga layanan yang masih mahal. (Sihotang, 2011)

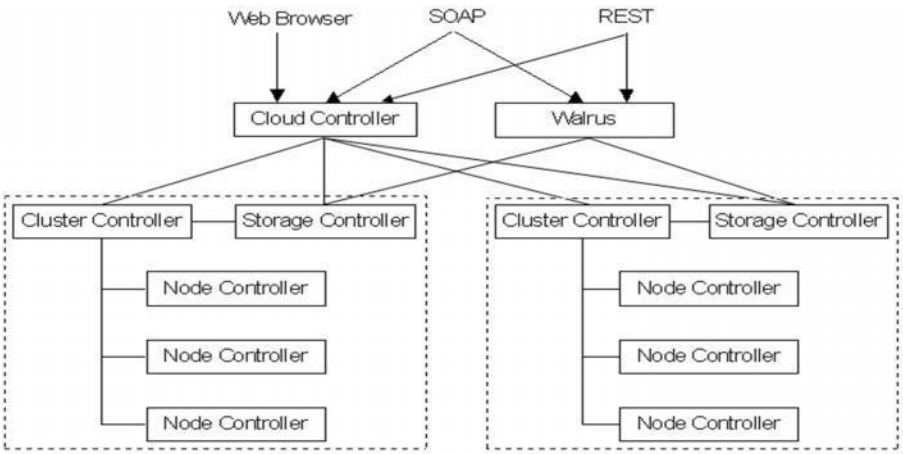
2.2. *Ubuntu Enterprise Cloud* (UEC)

2.2.1. Pengertian UEC

UEC merupakan tipe dari *cloud computing* yang termasuk ke dalam jenis *private cloud*. UEC adalah suatu aplikasi *stack* dari *Canonical* yang termasuk ke dalam edisi *Ubuntu Server*. *Canonical* adalah salah satu perusahaan *software* yang menyediakan aplikasi yang mendukung untuk membuat *cloud*. UEC menggunakan *Eucalyptus* bersama sejumlah *software open source* lainnya. (Anonim c, 2011)



Gambar 2.8. Arsitektur UEC 1



Gambar 2.9. Arsitektur UEC 2

2.2.2. Komponen UEC

2.2.2.1. *Node Controller* (NC)

NC adalah suatu *server* dengan prosesor yang memiliki kemampuan *Virtualization Technology* (VT), yang mampu menjalankan KVM (*Kernel based Virtual Machine*) sebagai *hypervisor* UEC. VM yang dijalankan pada *hypervisor* dan dikontrol oleh UEC biasanya disebut sebagai *"Instance"*.

NC berinteraksi dengan sistem operasi dan *hypervisor* yang berjalan di *node*. NC juga berinteraksi dengan CC. NC akan menanyakan sistem operasi yang berjalan di *node* untuk mengetahui sumber daya fisik yang digunakan *node*, seperti jumlah *core*, besar memori, ketersediaan *disk space* dan juga mengecek status dari VM *instance* yang berjalan di *node* dan memberikan informasi tersebut ke CC.

Fungsi:

1. Mengumpulkan data yang terkait dengan ketersediaan dan penggunaan sumber daya di *node* dan melaporkannya ke CC.
2. Manajemen siklus kehidupan dari *instance*.

2.2.2.2. *Cluster Controller* (CC)

CC mengatur satu atau lebih NC dan menjalankan/mengatur *instance* pada NC. CC mengatur jaringan *instance* yang berjalan di *node* sesuai dengan permintaan mode jaringan dari *Eucalyptus*. CC berkomunikasi dengan CLC dan banyak NC.

Fungsi:

1. Menerima permintaan dari CLC untuk menjalankan *instance*.
2. Memutuskan NC mana yang akan digunakan untuk menjalankan *instance*.
3. Mengatur ketersediaan *virtual network* untuk *instance*.
4. Mengumpulkan informasi tentang NC yang terdaftar pada CC dan melaporkannya ke CLC.

2.2.2.3. *Walrus Storage Controller* (WS3)

WS3 memberikan layanan penyimpanan yang sederhana menggunakan API REST dan SOAP yang kompatibel dengan API S3.

Fungsi:

1. Menyimpan *machine image* dan *snapshot*.
2. Menyimpan dan memberikan layanan *file* menggunakan API S3.

2.2.2.4. *Storage Controller* (SC)

SC menyediakan *block storage* yang digunakan oleh *instance*. Layanan ini mirip dengan layanan *Elastic Block Storage* (EBS) dari *Amazon Web Service* (AWS).

Fungsi:

1. Membuat *device* EBS dan *snapshot* untuk *volume*.
2. Memberikan layanan *block storage* melalui protokol AoE / iSCSI ke *instance*.

2.2.2.5. *Cloud Controller* (CLC)

CLC adalah *front end* dari seluruh infrastruktur *cloud*. CLC memberikan antar muka layanan *web* yang kompatibel dengan EC2/S3 ke *client.* CLC berinteraksi dengan seluruh komponen infrastruktur *Eucalyptus*. CLC mengetahui ketersediaan dan penggunaan sumber daya di *cloud* maupun status *cloud*.

Fungsi:

1. Mengawasi ketersediaan sumber daya berbagai komponen infrastruktur *cloud*, termasuk *hypervisor* pada NC yang digunakan untuk manajemen *instance* dan CC untuk mengatur *node hypervisor*.
2. Arbitrasi sumber daya untuk menentukan *cluster* yang digunakan *instance*.
3. Mengawasi *instance* yang sedang berjalan. (Anonim c, 2011)

2.2.3. Topologi Mesin Fisik UEC

1. Satu Mesin Fisik

   Mesin A: CLC/*Walrus*/CC/SC/NC

2. Minimal Dua Mesin Fisik
   a. Mesin A: CLC/*Walrus*/CC/SC (*Interface User*)
   b. Mesin B: NC (*VM Hosting*)
   c. (lainnya: NC, NC, ...NC) (*VM Hosting*)
3. Minimal Tiga Mesin Fisik
   a. Mesin A: CLC/*Walrus* (*Interface User*)
   b. Mesin B: CC/SC (*Back End*)
   c. Mesin C: NC (*VM Hosting*)
   d. (lainnya: NC, NC, ...NC) (*VM Hosting*)
4. Minimal Empat Mesin Fisik

a. Mesin A: CLC (*Interface User*)
b. Mesin B: *Walrus* (*Interface User*)
c. Mesin C: CC/SC (*Back End*)
d. Mesin D: NC (*VM Hosting*)
e. (lainnya NC, NC, ...NC) (*VM Hosting*)

5. Minimal Lima Mesin Fisik
   a. Mesin A: CLC/*Walrus* (*Interface User*)
   b. Mesin B: CC1/SC1 (*Back End*)
   c. Mesin C: NC1 (*VM hosting*)
   d. Mesin D: CC2/SC2 (*Back End*)
   e. Mesin E: NC2 (*VM hosting*)
   f. (lainnya CC/SC/NC, CC/SC/NC, ...CC/SC/NC) (Ideaman007, 2010)

2.2.4. *Setup* UEC

Setup yang digunakan adalah contoh *setup* minimal di *Eucalyptus*. CC dan NC dapat diinstal dalam satu mesin. Akan tetapi, *setup* ini tidak mendukung jaringan *mode Managed* dan *Managed-NOVLAN*.

Tabel 2.1. Spesifikasi *Hardware Server*

Hardware	***Server* 1**		***Server* 2**		***Server* 3**	
	Minimum	**Disarankan**	**Minimum**	**Disarankan**	**Minimum**	**Disarankan**
CPU	1GHz	2 x 2GHz	VT *extensions*	VT (64Bit) *Multicore*	VT *extensions*	VT (64Bit) *Multicore*
Memory	1 GB	2 GB	1 GB	4 GB	1 GB	2 GB
Disk	5400rpm IDE	7200rpm SATA	5400rpm IDE	7200rpm SATA or SCSI	5400rpm IDE	7200rpm SATA or SCSI
Disk Space	40GB	200GB	40GB	100GB	40GB	100GB
Networking	100Mbps	1000Mbps	100Mbps	1000Mbps	100Mbps	1000Mbps

Tabel 2.2. Spesifikasi Instalasi UEC

	***Server* 1**	***Server* 2**	***Client***
Fungsi	CLC, CC, SC dan Walrus	NC	*Bundling* dan *Interface Web Client*
No. NIC	eth0- (*Enterprise* N/W)	eth0- (*Eucalyptus* N/W)	eth0- (*Enterprise* N/W)
	eth1-(*Eucalyptus* N/W)		eth1- (*Eucalyptus* N/W)
IP Address	eth0 – 192.168.10.121 (set saat instalasi)	eth0 – 192.168.20.2 (set saat instalasi)	eth0 – ???
	eth1 – 192.168.20.1 (set setelah instalasi)		eth1 – ???
Hostname	server1.example.com	server2.example.com	client1.example.com
Nama Server Enterprise	192.168.10.2	192.168.10.2	192.168.10.2
	192.168.10.3	192.168.10.3	192.168.10.3
IP Gateway	192.168.10.100	192.168.20.1	192.168.10.2
			192.168.10.3

Penjelasan :

1. *Gateway* pada *Server* 2 di *set* ke IP CC, yaitu 192.168.20.1 . Hal ini memungkinkan *Server* 2 terhubung ke jaringan *enterprise* melalui *Server* 1.
2. *Server* 1 adalah *64-bit server* dan *Server* 2 adalah *64-bit server* dengan *VT-enabled*.
3. Jaringan *enterprise* menggunakan kelas C *private network* 192.168.10.0/ 255.255.255.0
4. *IP address* yang dialokasikan untuk *instance cloud* adalah *IP address* publik : 192.168.10.200 - 192.168.10.220 (*Range Enterprise*).
5. *Default name* c*luster* "cluster1".
6. Komponen UEC akan menggunakan kelas C *private network* untuk interkoneksi : 192.168.20.0/255.255.255.0. (Anonim c, 2011)

2.3. Eucalyptus

2.3.1. Pengertian *Eucalyptus*

Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems (EUCALYPTUS) adalah *platform software* untuk implementasi *cloud computing private* di *cluster* komputer. Ada dua edisi *Eucalyptus*, yaitu *enterprise* dan *open source*. Saat ini, *Eucalyptus* hanya mengekspor antarmuka yang kompatibel dengan layanan *Amazon* EC2 dan S3. Pengembangan *Eucalyptus* disponsori oleh *Eucalyptus Systems*, sebuah perusahaan *startup* yang mengerjakan *software* yang kompatibel dengan banyak distribusi *Linux*, seperti : *Ubuntu, Red Hat Enterprise Linux* (RHEL), *CentOS*, *SUSE Linux Enterprise Server* (SLES), *OpenSUSE*, *Debian*, dan *Fedora*.

Eucalyptus juga bisa menggunakan berbagai teknologi virtualisasi, seperti: *VMWare*, *Xen*, dan KVM *hypervisors* untuk mengimplementasikan abstraksi *cloud* yang ada. *Eucalyptus* menerapkan metode IaaS untuk *cloud private* dan *cloud hybrid*.

Eucalyptus mengimplementasi *web service Amazon Web Service* (AWS) API yang memungkinkan *Eucalyptus* memiliki interopabilitas dengan berbagai layanan AWS dan *tools Eucalyptus* punya banyak *command* yang disebut *Euca2ools*. *Tools* tersebut dapat dipakai secara internal untuk berinteraksi dengan instalasi *private cloud* dari *Eucalyptus* atau secara eksternal untuk berinteraksi dengan layanan *public cloud*, termasuk *Amazon* EC2.

Eucalyptus awalnya hanya merupakan program riset di bidang *High Performance Computing* (HPC), yaitu sebuah tesis yang diawasi oleh Profesor Rich Wolski di Departemen Ilmu Komputer, University of California, Santa Barbara. (Anonim a, 2011)

2.3.2. Fitur *Eucalyptus*

Beberapa fitur dari *Eucalyptus*, yaitu:

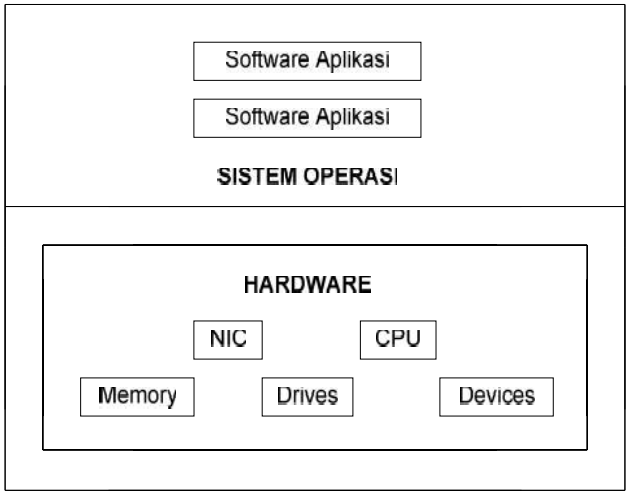
1. Kompatibel dengan API *web service* dari *Amazon.*
2. Instalasi dan *deployment* dari *source* berupa paket *Debian* (DEB) dan *Red Hat Package Manager* (RPM).
3. Jaminan keamanan komunikasi antara proses internal menggunakan SOAP dan *Web Services Security* (*WS-Security).*
4. Mendukung mesin *virtual* untuk *Linux* dan *Windows*.
5. Mendukung banyak *cluster* sebagai *cloud* tunggal.
6. IP *elastic* dan *security group*.
7. Manajemen pengguna dan *group*.
8. Laporan akuntansi.
9. Kebijakan penjadwalan dan SLA (*Service Level Agreement*) yang bisa diatur secara fleksibel. (Anonim a, 2011)

2.4. Virtual Box

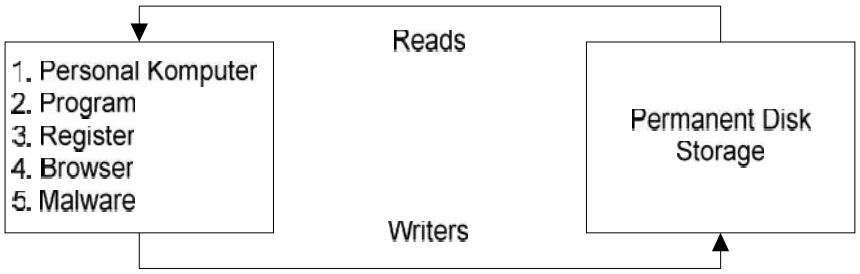
2.4.1. Pengertian *Virtual Box*

Virtual Box merupakan suatu *platform* aplikasi yang mendukung virtualisasi. Teknologi *Virtual Box* ini tidak sama dengan komputer dengan sistem operasi *dual boot*, dimana suatu komputer dapat dipilih untuk *boot* dengan sistem operasi yang berbeda. Pada sistem operasi *dual boot*, hanya satu sistem operasi yang berfungsi, yaitu sistem operasi yang dipilih pada saat *boot*. Sedangkan dengan *Virtual Box* beberapa sistem operasi dapat dijalankan/berfungsi secara bersamaan pada *Virtual Machine* (VM)-nya masing-masing. Pengguna dapat menginstal dan menjalankan VM sebanyak yang diinginkan. *Software* ini sangat cocok untuk uji coba di bidang jaringan. Setiap VM yang dibuat oleh *Virtual Box* tersebut bekerja seolah-olah merupakan komputer biasa yang berdiri sendiri.

Virtual Box dikembangkan oleh perusahaan *software* Jerman yang bernama *Innotek* pada tahun 2007, kemudian dimiliki oleh *Sun Microsystems* pada bulan Februari 2008. *Software* ini didistribusikan dibawah lisensi *Personal Use and Evaluation License* (PUEL) sebagai *software Open Source Edition* (OSE) yang dirilis gratis dibawah lisensi GNU *General Public License*. (Vojtesek dkk, 2009)



Gambar. 2.10. *Physical Computer*

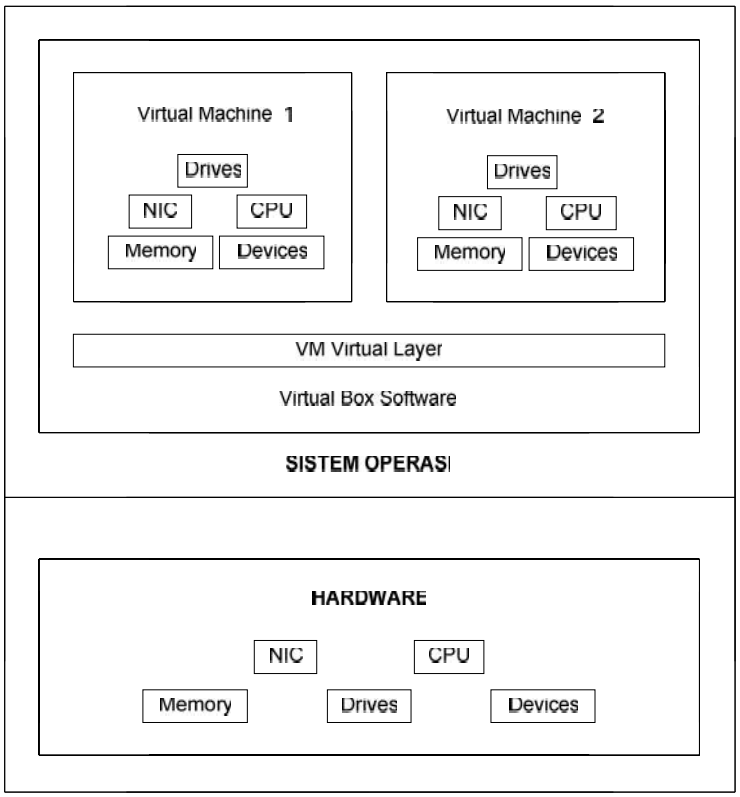


Gambar 2.11. Alur Komputer Standar

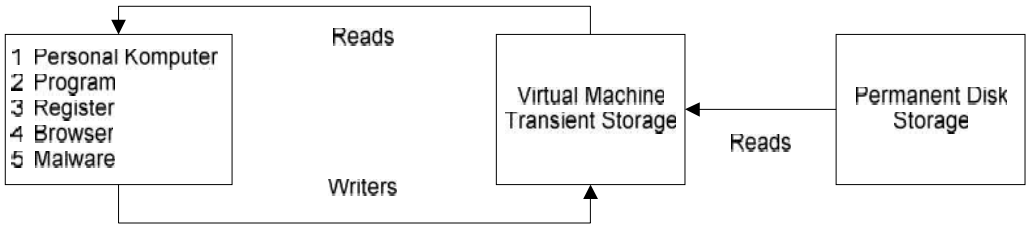
Virtual Machine (VM) adalah komputer semu yang dibuat dengan *software Virtual Box*. Komputer-komputer semu ini hanya hanya dapat dijalankan pada komputer dimana *software Virtual Box* telah terinstalasi.

Struktur VM tak berbeda dengan struktur *physical computer*, hanya komponen-komponen yang dimilikinya semua dalam bentuk semu, seperti *virtual* CPU, *virtual memory, virtual drive,* dan *virtual network*.

Setiap VM yang dibuat disimpan sebagai file-file biasa pada komputer *host* sehingga VM tersebut dengan mudah dapat dipindahkan ke komputer lain. VM duplikat (*clone*) dapat dibuat hanya dengan menyalin (*copy*) file-file VM tersebut.



Gambar. 2.12. *Virtual Computer*



Gambar 2.13. Alur Komputer Virtualisasi

2.4.2. Karakteristik *Virtual Box*

Karakteristik *Virtual Box* ada sembilan, yaitu:

1. Dapat menjalankan OS secara bersamaan.

   Virtual Box dapat menjalankan lebih dari satu OS sekaligus secara bersamaan. OS VM pada *Virtual Box* memiliki spesifikasi sistem yang sama dengan OS pada PC yang sebenarnya.

2. Mudah dalam instalasi *software*.

Instalasi OS VM pada *Virtual Box* tidak menyebabkan kerusakan pada *host computer*, karena dapat dengan mudah dihapus tanpa menyebabkan *crash*.

3. Penggabungan infrastuktur.

   Virtualisasi secara signifikan dapat mengurangi biaya *hardware* dan listrik.

4. Kompatibel.

   Virtual Box dapat dijalankan pada *host operating* 64-bit maupun 32-bit.

5. Tidak membutuhkan *hardware* virtualisasi.

   Virtual Box tidak membutuhkan spesifikasi prosesor yang mendukung *Virtualization Technology* (VT). *Virtual Box* mendukung semua prosesor.

6. Terdapat aplikasi *guest additions*.

   Guest additions adalah paket *software* yang dapat diinstal untuk mendukung VM mengembangkan kemampuan dan juga menyediakan tambahan integrasi dan komunikasi dengan sistem *host*. *Guest additions* mendukung VM untuk menyesuaikan resolusi video, akselerasi grafik 3D, dan sebagainya.

7. Mendukung USB *device*.

   Virtual Box mengimplementasikan *virtual USB controller* yang mendukung terhubungnya USB *device* ke VM, tanpa harus menginstal *driver* tertentu pada *host*. Dukungan USB tidak terbatas pada kategori *device* tertentu.

8. Resolusi *Multiscreen*.

   VM *Virtual Box* mendukung resolusi *screen* lebih banyak dari *physical screen.*

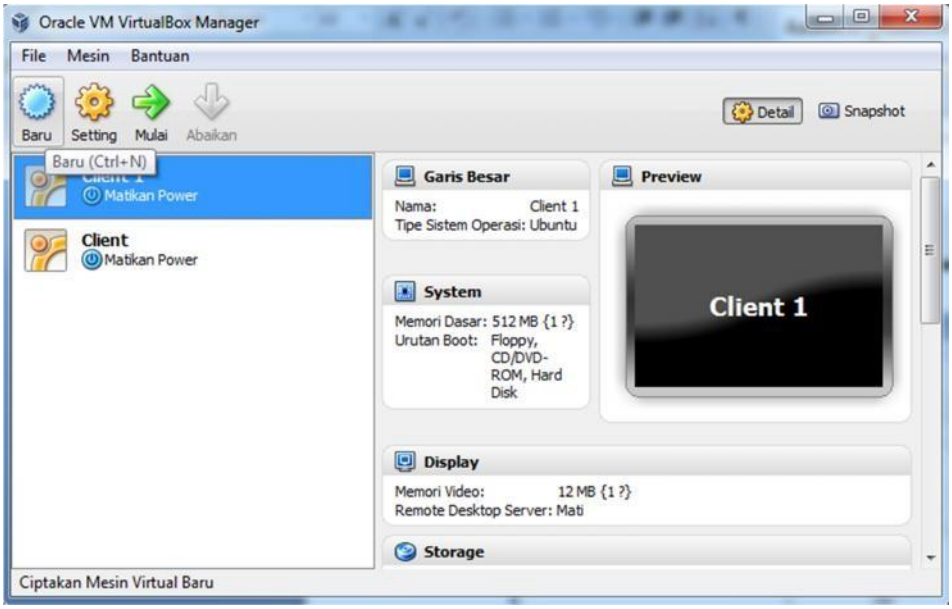
9. Dapat *boot* PXE (*Preboot Execution Environment*) *Network*.

   Virtual Box network cards mendukung penuh *remote booting* melalui PXE. (Singh dan Bindal, 2011)

2.4.3. Langkah – langkah Pembuatan VM

Langkah-langkah pembuatan VM, yaitu:

1. Klik tombol “Baru” untuk membuat VM baru.

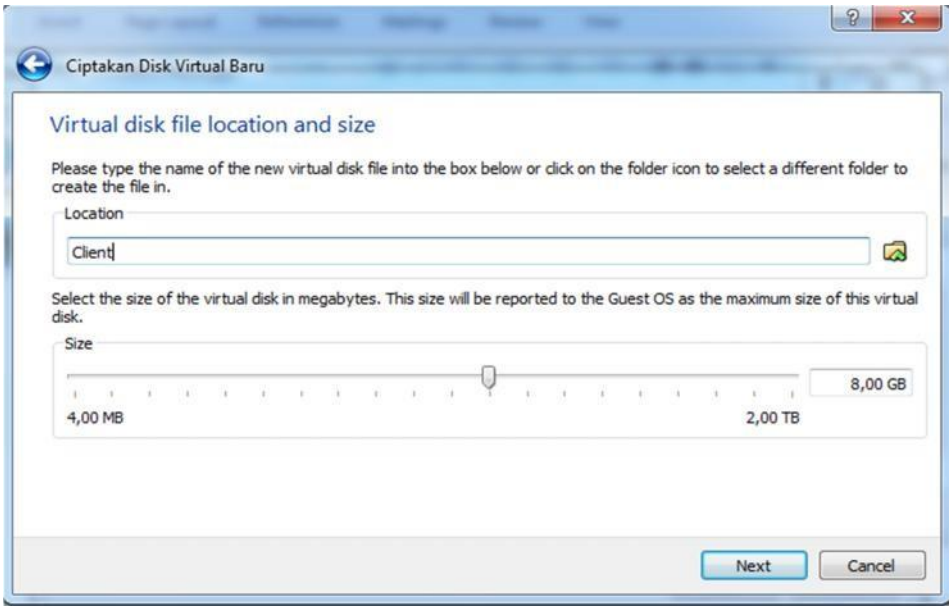


Gambar 2.14. *Create* VM

2. Konfigurasi nama mesin, OS dan versinya.

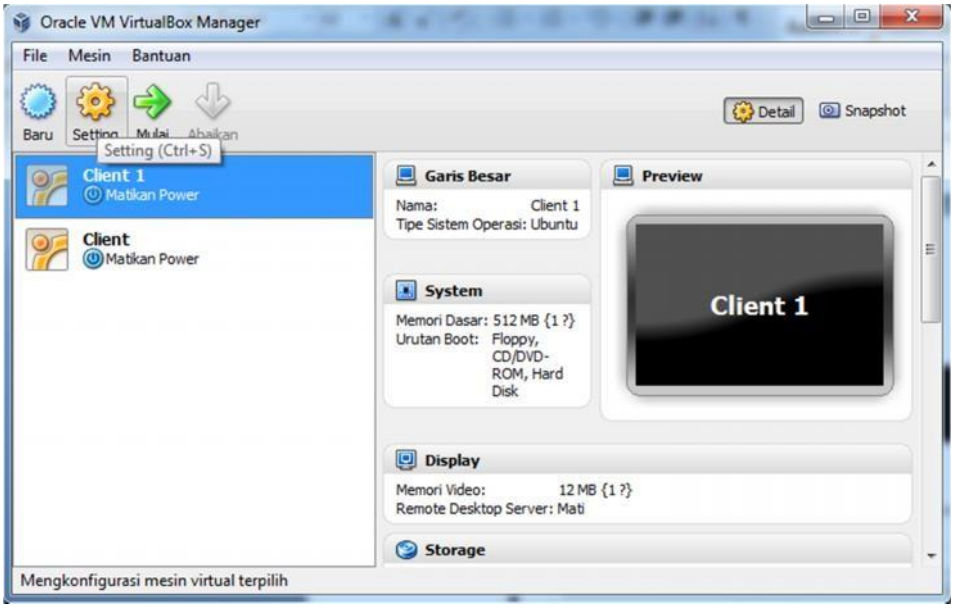
Gambar 2.15. Konfigurasi Nama dan OS VM

3. Konfigurasi lokasi *file* dan ukuran *file*.



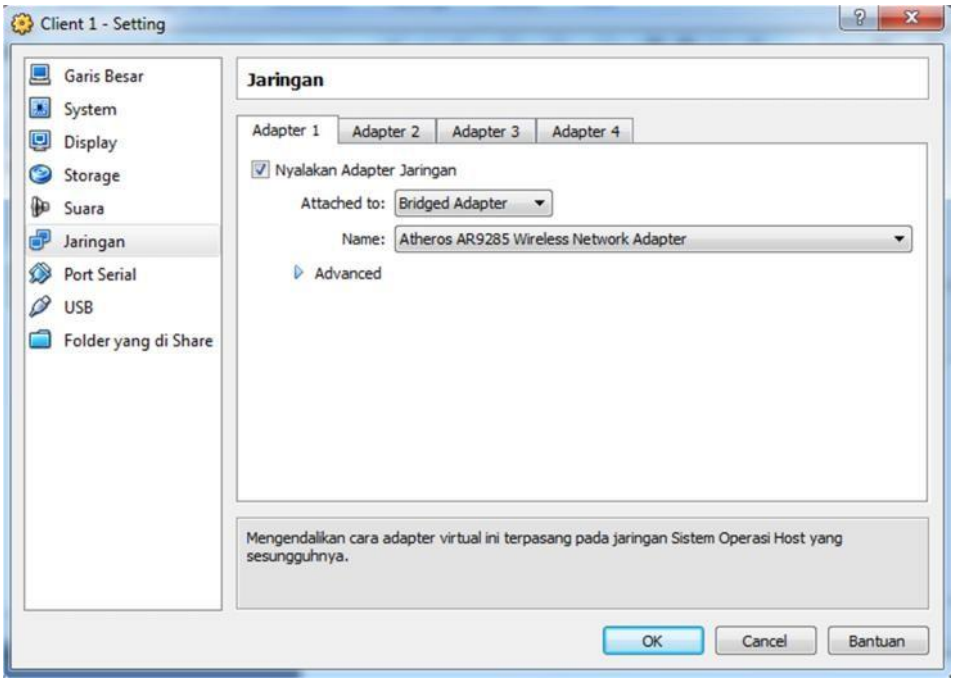
Gambar 2.16. Konfigurasi Lokasi dan Ukuran *File*

4. Klik tombol “Setting” untuk konfigurasi sistem VM.



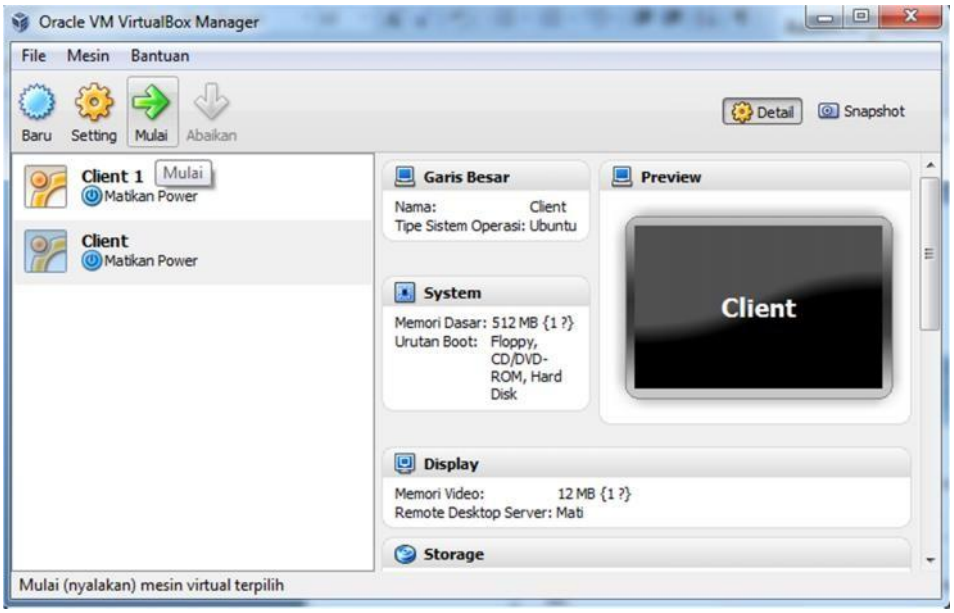
Gambar 2.17. *Setting* VM

5. Konfigurasi sistem VM. Setelah selesai klik OK.



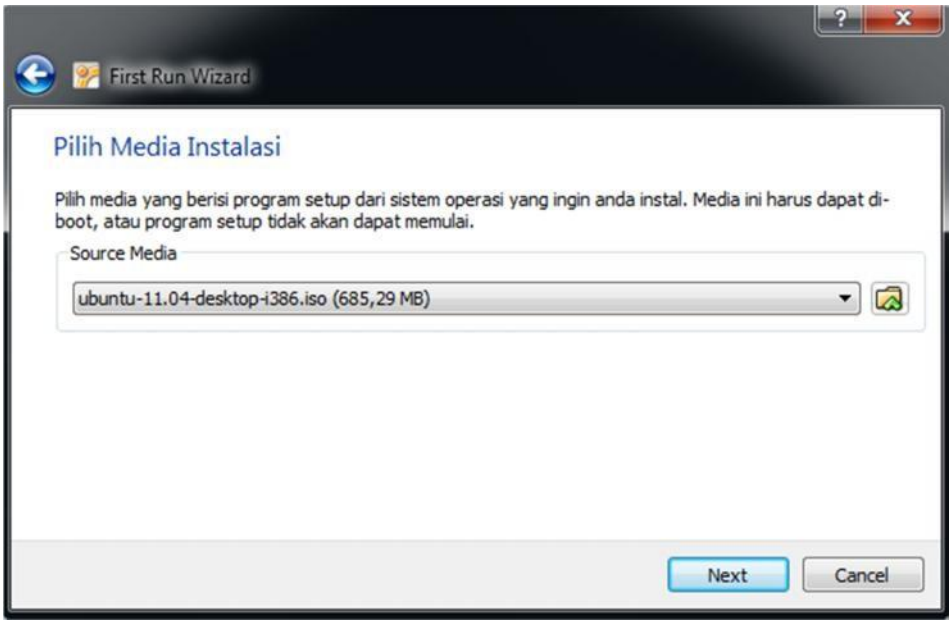
Gambar 2.18. Konfigurasi Sistem VM

6. Klik tombol “Mulai” untuk memulai proses instalasi VM.



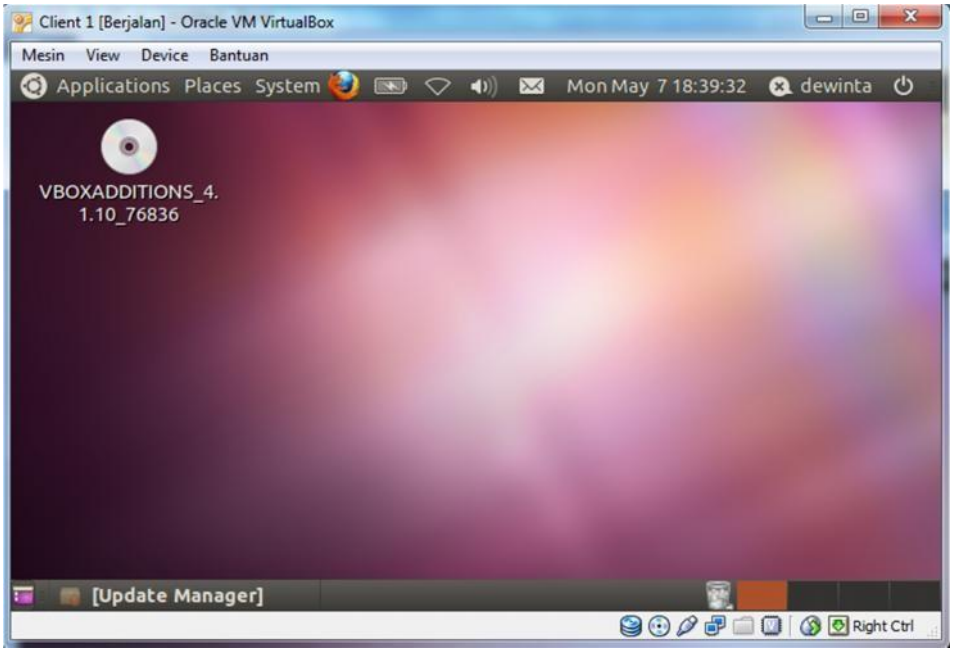
Gambar 2.19. *Start* VM

7. Pilih media instalasi pada tombol *browse*.



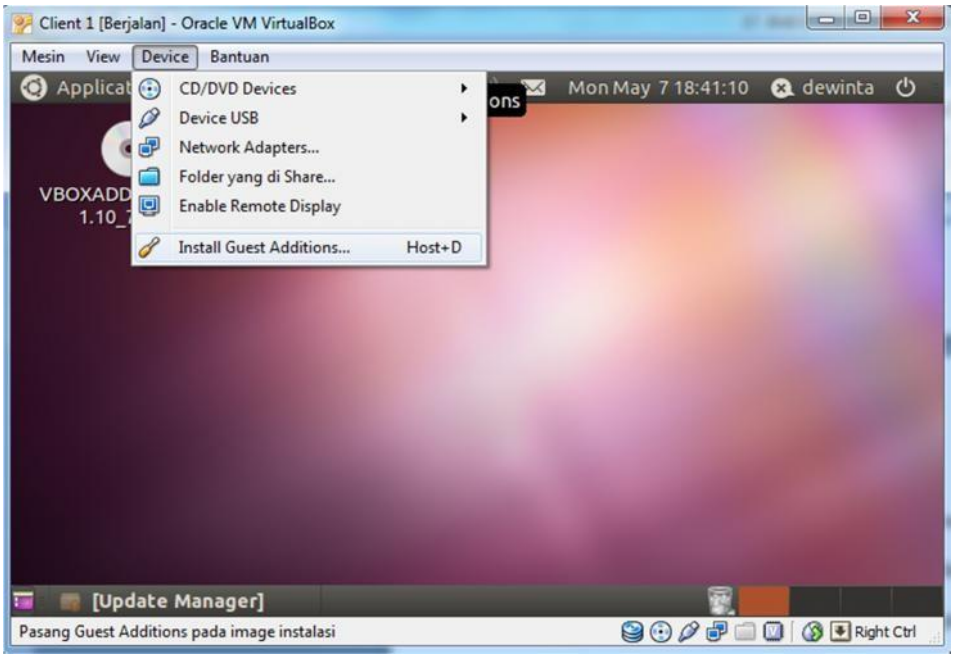
Gambar 2.20. Pemilihan Media Instalasi

8. Selesai proses instalasi selesai, maka akan muncul tampilan berikut :



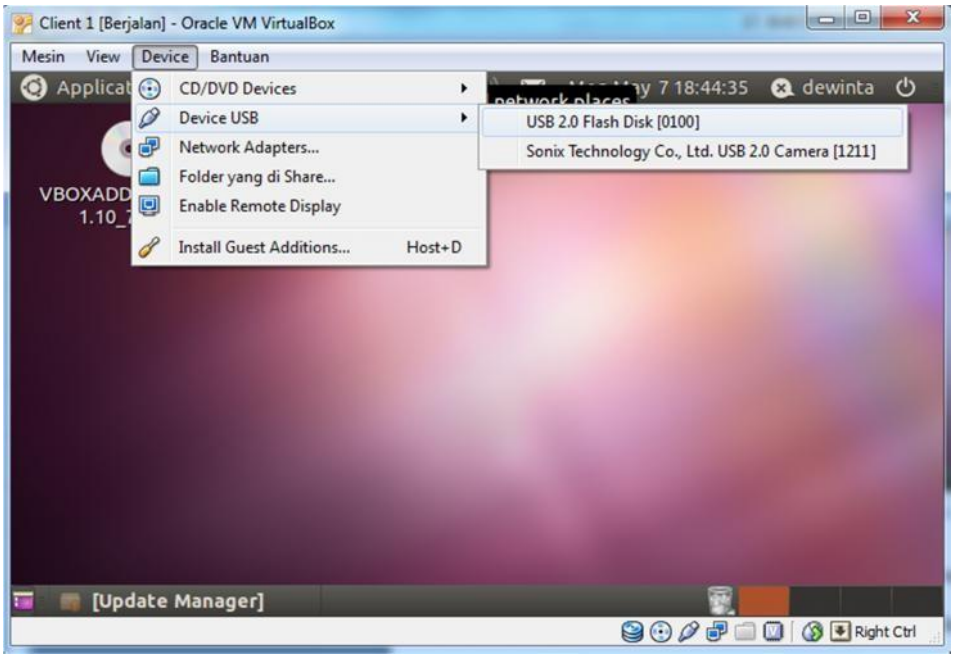
Gambar 2.21. Tampilan VM *Ubuntu Desktop*

9. Instal “Guest Additions” agar mendapatkan fasilitas lengkap VM.



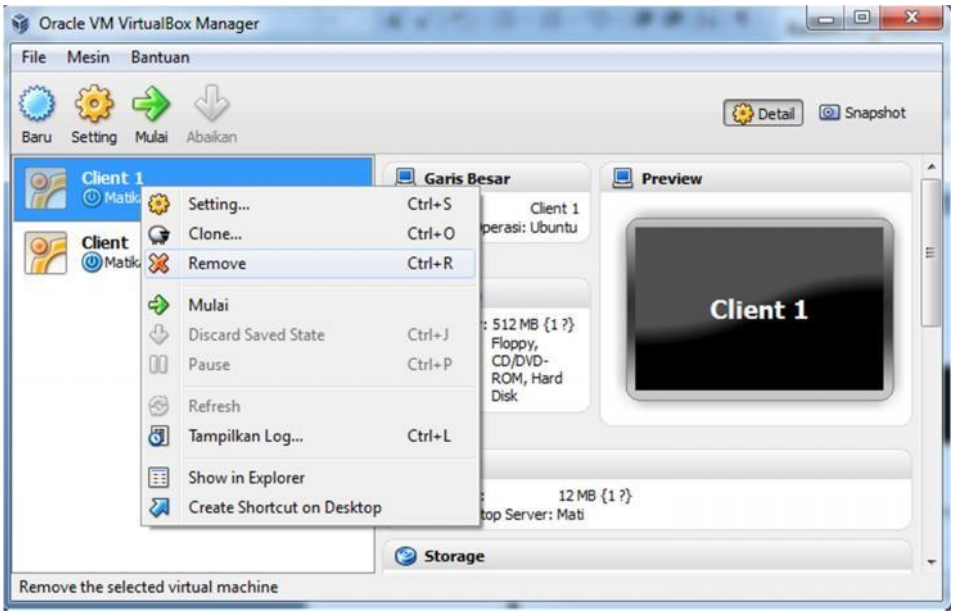
Gambar 2.22. *Install Guest Additions*

10. Klik “Device USB” dan pilih USB 2.0 *Flash Disk* untuk mendeteksi keberadaan *Flash Disk* pada VM.



Gambar 2.23. *Device USB Flash Disk*

11. Jika ingin menghapus VM yang sudah dibuat, maka klik kanan VM yang ingin dihapus. Kemudian pilih “Remove”.



Gambar 2.24. *Remove* VM

2.5. Perintah Dasar *Linux*

2.5.1. Acpi

Perintah untuk melihat estimasi baterai.

```
$ acpi
```

2.5.2. Apt-Get

Perintah untuk menginstal atau *uninstall* aplikasi di *Linux*.

Format *Install*: apt-get install <<nama_paket>>

```
$ apt-get install ntp
```

Format *Uninstall*: apt-get remove <<nama_paket>>

```
$ apt-get remove ntp
```

Selain itu, perintah ini dapat juga digunakan untuk mengecek pembaruan dari aplikasi yang terinstal di *Linux*, yaitu :

```
$ apt-get update
```

Sedangkan untuk memperbarui aplikasi *Linux*, digunakan setelah *update*, yaitu :

```
$ apt-get upgrade
```

2.5.3. Cat

Perintah untuk menampilkan isi dari sebuah *file* di layar.

Format: cat <<alamat_file>>

```
$ cat /var/log/eucalyptus/registration.log
```

2.5.4. Cd

Perintah untuk keluar ataupun masuk ke direktori suatu folder.

Format keluar: cd <<tanda titik dua kali>>

```
$ cd ..
```

Format masuk: cd <<folder>>

```
$ cd etc
```

2.5.5. Chmod

Perintah untuk menambah dan mengurangi izin pengguna untuk mengakses *file* atau direktori. Pengguna dapat menggunakan sistem *letter coding* atau sistem *numeric coding*. Ada tiga jenis *permission* yang dapat diubah, yaitu : r untuk *read*, w untuk *write*, dan x untuk *execute*.

2.5.5.1. Sistem *Letter Coding*

Permission dapat diubah untuk masing-masing u (*user*), g (*group*), o (*other*) dan a (*all*) hanya dengan memberi tanda *plus* (+) untuk menambah izin dan tanda *minus* (-) untuk mencabut izin.

Misalnya, untuk memberikan izin baca dan eksekusi *file* coba 1 kepada *owner* dan

group, perintahnya adalah :

```
$ chmod ug+rx coba1
```

Sedangkan untuk mencabut izin-izin tersebut menggunakan perintah :

```
$ chmod ug-rx coba1
```

2.5.5.2. Sistem *Numeric Coding*

P*ermission* untuk *user*, *group* dan *other* ditentukan dengan menggunakan kombinasi angka-angka, 4, 2 dan 1, dimana 4 (*read*), 2 (*write*) dan 1 (*execute*). Misalnya, untuk memberikan izin baca(4), tulis(2) dan eksekusi(1) *file* coba 2 kepada *owner*, perintahnya adalah:

```
$ chmod 700 coba2
```

2.5.6. Clear

Perintah untuk menghapus/membersihkan layar.

```
$ clear
```

2.5.7. Cp

Perintah untuk menyalin *file* atau *copy*, misalnya untuk menyalin *file* 1 menjadi *file* 2.

Format: cp <<file1>> <<file2>>

```
$ cp siswa siswi
```

2.5.8. *Ctrl* + C

Perintah untuk keluar dari suatu perintah, misalnya ingin menghentikan *ping*.

2.5.9. *Ctrl* + H

Perintah untuk memunculkan *file hidden*.

2.5.10. Date

Perintah untuk melihat tanggal dan waktu sekarang.

```
$ date
```

2.5.11. Df

Perintah untuk melihat *available use memory*.

```
$ df
```

2.5.12. Exit

Perintah untuk keluar dari *root*.

```
$ exit
```

2.5.13. Grep

Global Regular Expresion Parse (GREP) adalah perintah untuk mencari *file – file*

yang mengandung teks dengan kriteria yang diinginkan.

Format: grep <<teks>> <<file>>

Misalnya, akan mencari *file-file* yang mengandung teks marginal di direktori.

```
$ grep nc /etc/eucalyptus
```

2.5.14. Halt

Perintah ini hanya bisa dijalankan oleh *super user* atau *login* sebagai *root*. Perintah ini untuk

memberitahu *kernel* supaya mematikan sistem atau *shutdown*.

```
$ halt
```

2.5.15. Ifconfig

Perintah untuk melihat konfigurasi jaringan, seperti : *IP Address*, *Netmask*, dan sebagainya.

```
$ ifconfig
```

2.5.16. Ls

Perintah untuk menampilkan isi dari sebuah *direktori*. Bila dijalankan tanpa *option*, maka akan menampilkan seluruh *file nonhidden* secara *alphabet*.

```
$ ls
```

2.5.17. Mkdir

Perintah untuk membuat direktori baru.

Format: mkdir <<nama folder>>

```
$ mkdir cloud
```

2.5.18. Nano

Perintah untuk mengedit suatu *file*.

```
$ nano /etc/network/interfaces
```

Setelah selesai, simpan menggunakan ctrl+o dan keluar menggunakan ctrl+x.

2.5.19. Reboot

Perintah untuk me-*restart* komputer.

```
$ reboot
```

Selain itu, pengguna juga dapat menggunakan alt+ctrl+del untuk *restart* komputer.

2.5.20. Rm

Perintah untuk menghapus *file* dan secara *default rm* tidak menghapus direktori. Hati-hati menggunakan perintah ini terutama dengan *option* -r yang secara rekursif dapat menghapus seluruh *file*.

```
$ rm ~/.ssh/known_hosts
```

2.5.21. Sudo

Perintah untuk *root*.

```
$ sudo
```

2.5.22. Su

Perintah untuk *login* sementara sebagai *user* lain. Bila *user* ID tidak disertakan maka komputer menganggap *user* ingin *login* sementara sebagai *super user* atau *root*. Bila *user* bukan *root* dan *user* lain memiliki *password*, maka *user* harus memasukkan *password*. Namun bila *user* adalah *root*, maka *user* dapat *login* sebagai *user* lain tanpa perlu mengetahui *password user* tersebut.

```
$ sudo su
```

2.5.23. Tail

Perintah untuk menampilkan 10 baris terakhir dari suatu *file*. *Default* baris yang ditampilkan adalah 10, tetapi pengguna dapat menentukan sendiri berapa baris yang ingin ditampilkan.

Format: tail <<jumlah baris>> <<file>>

```
$ tail 15 /var/log/eucalyptus/axis2c.log
```

(Anonim b, 2011)