RANCANG BANGUN ARSITEKTUR LABORATORIUM VIRTUAL JARINGAN KOMPUTER DENGAN VIRTUALISASI BERBASIS *CONTAINER*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Dwiyanto Gunawan
NIM: 135150109111018

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

RANCANG BANGUN ARSITEKTUR LABORATORIUM VIRTUAL JARINGAN KOMPUTER DENGAN VIRTUALISASI BERBASIS *CONTAINER*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Dwiyanto Gunawan
NIM: 135150109111018

Skripsi ini telah diuji dan dinyatakan lulus pada
26 Agustus 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Achmad Basuki, S.T, M.MG, Ph.D	Adhitya Bhawiyuga, S.Kom, M.S
NIP: 19741118 200312 1 002	NIK: 201405 890720 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 30 Agustus 2016

Dwiyanto Gunawan

NIM: 135150109111018

KATA PENGANTAR

Puji dan syukur Penulis panjatkan kehadirat Allah SWT, karena hanya dengan rahmat dan bimbingannya Penulis dapat menyelesaikan tugas akhir dengan judul "Rancang Bangun Arsitektur Laboratorium Virtual Jaringan Komputer Dengan Virtualisasi Berbasis *Container*" dengan baik. Tanpa rahmat dan bimbingan dari Tuhan Yang Maha Esa, maka niscaya Penulis tidak akan dapat menyelesaikan tugas akhir ini dengan baik dan tepat waktu.

Penulisan dan penyusunan laporan skripsi ini dapat terlaksana dengan baik karena adanya bantuan secara langsung maupun tidak langsung dari pihak tertentu diantaranya:

1. Bapak Achmad Basuki, ST., MMG., Ph.D selaku dosen pembimbing I yang telah banyak memberikan ilmu dan saran untuk laporan skripsi ini.
2. Bapak Adhitya Bhawiyuga, S.Kom, M.S selaku dosen pembimbing II yang juga banyak memberikan ilmu dan saran untuk laporan skripsi ini.
3. Kedua orang tua penulis Sukarji dan Su'in yang telah memberi motivasi, kasih sayang serta dukungan moril dan materil.
4. Semua teman-teman FILKOM, khususnya SAP Informatika/Ilmu Komputer 2013 terima kasih atas segala bantuan dan dukungannya selama ini.
5. Segenap dosen dan karyawan FILKOM Universitas Brawijaya yang telah membantu pelaksanaan skripsi ini.
6. Teman-teman Nomaden Coffee, terima kasih atas dukungan semangat, do'a, dan pengertiannya.
7. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Semoga jasa dan amal baik mendapatkan balasan dari Allah SWT. Dengan segala kerendahan hati, penulis menyadari sepenuhnya bahwa skripsi ini masih jauh dari sempurna karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Akhirnya semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa Filkom Universitas Brawijaya.

Malang, 30 Agustus 2016

Dwiyanto Gunawan

dwiyantogunawan@gmail.com

ABSTRAK

Perkembangan Teknologi Informasi yang begitu pesat menuntut harus semakin ditingkatkannya kualitas sumber daya manusia. Filkom Universitas Brawijaya merupakan salah satu fakultas yang bergerak di bidang Teknologi Informasi. Penerimaan mahasiswa di Filkom yang setiap tahunnya semakin bertambah ini tidak didukung dengan sarana laboratorium yang mencukupi, terutama masalah jumlah fasilitas perangkat yang kurang membuat mahasiswa harus berbagi jadwal untuk menggunakan laboratorium. Container dengan virtualisasi tingkat sistem operasi memungkinkan menjalankan banyak komputer virtual pada komputer fisik tunggal. Komputer virtual yang berjalan saling berbagi sumber daya dari satu komputer fisik yang sama. Penelitian ini bertujuan untuk merancang dan membuat sistem *back end* laboratorium virtual jaringan komputer yang menawarkan solusi terbatasnya jumlah perangkat praktikum dan efektifitas waktu. Penelitian ini menggunakan teknologi *container* dari Docker Container sebagai penyedia virtualisai dan Openstack sebagai pengelola lingkungan virtualisasi. Berdasarkan hasil penelitian yang dilakukan, sistem yang dibangun mampu mengimplentasikan virtualisasi sebagai media praktikum jaringan komputer. Sistem ini mempunyai fitur membuat komputer virtual, pengaksesan komputer virtual, dan modul praktikum jaringan komputer. Sistem ini mampu membuat komputer virtual dalam waktu rata-rata kurang dari tiga detik dan jumlah komputer yang dipesan oleh aplikasi *front-end* tidak berpengaruh terhadap waktu pembuatan komputer virtual.

Kata kunci: Docker, Laboratorium virtual, *Back-end*, *Virtual machine*, Praktikum

ABSTRACT

The development of information technology should be followed by human resources quality improvement. Faculty of Computer Science "Universitas Brawijaya" is one of the educational institutions which concern with information technology. Hardware and software facilities in the laboratories is no longer sufficient due to the number of students has increased every year, It can cause students have to share a schedule for using the laboratory. Container with the virtualization operating system level able to run multiple virtual computers on a single physical computer. This research aims to design and make the back end system of virtual computer networking laboratory to overcome the number of practical tools limitation and time effectiveness enhancement. This research uses container technology from Docker as virtualization provider and Openstack as the manager of virtualization environment. Based on the results of research, the system is able to implement the virtualization as a practice tool for laboratory of computer networking. This system has a feature to make a virtual computer, access a virtual computer, and the module of computer networks practice. The system is able to create a virtual computer within an average time of less than three seconds and the number of computers ordered by the front-end application does not affect virtual computer creating time.

Keyword: Docker, Virtual Laboratory, Back end, Virtual machine, practical

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Perkembangan Teknologi Informasi yang begitu pesat menuntut harus semakin ditingkatkannya kualitas sumber daya manusia, agar sumber daya yang ada di Indonesia tidak tertinggal dengan negara-negara lain di dunia. Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya merupakan salah satu fakultas yang ada di Universitas Brawijaya. Fakultas ini merupakan fakultas yang bergerak di bidang Teknologi Informasi. Fakultas ini bertujuan untuk mencetak lulusan yang kompeten di bidang teknologi informasi.

Dalam proses perkuliahan di Fakultas Ilmu Komputer Universitas Brawijaya tidak sedikit mata kuliah yang membutuhkan laboratorium untuk melakukan praktikum. Hal ini bertujuan agar mahasiswa bisa mengimplementasikan secara nyata materi yang sebelumnya didapatkan pada saat perkuliahan di dalam kelas. Namun pada kenyataannya fasilitas perangkat yang terdapat di laboratorium tidak sebanding dengan banyaknya mahasiswa yang akan melakukan praktikum. Hal ini tentu saja tidak ideal untuk mahasiswa bisa melakukan praktikum dengan efektif, dikarenakan tidak semua mahasiswa mendapatkan perangkat untuk melakukan praktikum.

Salah satu solusi untuk mengatasi masalah tersebut adalah dengan membuat laboratorium virtual yang dapat memberikan perangkat praktikum berupa komputer virtual, sehingga pemanfaatan laboratorium sebagai media praktikum bisa lebih banyak mengakomodir mahasiswa yang ingin melakukan praktikum. Untuk mewujudkan hal tersebut maka diperlukan sistem laboratorium virtual baik *front end* maupun *back end* yang saling mendukung satu sama lainnya sehingga laboratorium virtual ini dapat mengatasi masalah-masalah yang telah disebutkan.

Adapun penelitian sebelumnya dilakukan oleh Dzulqarnain yang menggunakan virtualisasi untuk penyediaan *virtual machine*. Penelitian ini dilakukan dalam lingkungan pembelajaran laboratorium jaringan Fakultas Ilmu Komputer, Universitas Brawijaya. *Virtual machine* dalam penelitian ini disediakan bagi mahasiswa untuk dapat digunakan dalam proses praktikum. Teknologi virtualisasi yang digunakan dalam penelitian ini adalah KVM (*Kernel-based Virtual Machine*) yaitu salah satu teknologi virtualisasi penuh (*full virtualization*) yang dikembangkan oleh Linux (Dzulqarnain, 2015).

Selanjutnya penelitian yang dilakukan oleh Patcharee Basu dkk. (Basu, et al., 2013). Penelitian tersebut menggabungkan *remote computer laboratory* dalam lingkungan pembelajaran jarak jauh. Pada penelitan tersebut juga menggunakan KVM (*Kernel-Based Virtual Machine*) untuk menyediakan virtualisasi. KVM yang termasuk ke dalam *full virtualization* memiliki kelemahan *guest* tidak bisa mendapat kecepatan akses penuh saat mengakses *hardware* (misalnya menulis/membaca data dari dalam *harddisk* atau bertukar data lewat antarmuka jaringan). Lambatnya akses *input/output* ini karena, virtualisasi ini meng-emulasi semua perangkat keras dan menempatkan sumber daya ke dalam CPU dan harus

menjalankan sistem operasi secara penuh pada tiap *virtual machine*. Sehingga dibutuhkan kapasitas dan *memory* yang besar, karena *virtual machine* berbagi daya dengan komputer fisik (Tirtawidjaja, 2014).

Berbeda dengan *full virtualization* yang harus menjalankan sistem operasi secara penuh, container tidak menggunakan sistem operasi secara penuh, *container* lebih kecil dari *virtual machine* dan memungkinkan untuk *start up* yang lebih cepat dengan performa yang lebih baik, karena *container* berbagi *kernel* dengan sistem operasi *host*.

Penelitian ini terfokus pada rancangan sistem *back end* dari laboratorium *virtual* jaringan komputer. Rancangan sistem *back end* ini membahas beberapa hal seperti kebutuhan objek praktikum, yang meliputi banyaknya komputer virtual yang digunakan oleh praktikan dalam melaksanakan praktikum, mekanisme pengaturan jadwal untuk melaksanakan praktikum, dan pengaturan kondisi ketika *virtual machine* ketika sudah tidak digunakan oleh praktikan. Untuk mendukung rancangan tersebut maka digunakanlah teknik virtualisasi.

Berdasarkan penelitian yang dijelaskan sebelumnya, maka rancangan sistem *back end* yang dibuat menggunakan *container* sebagai komputer virtual. Sedangkan Docker *engine* digunakan sebagai perangkat lunak penyedia virtualisasi. Penelitian ini menggunakan *container* karena dapat memberikan penghematan konsumsi sumber daya tanpa *overhead* virtualisasi. Kemudian Openstack digunakan untuk manajemen lingkungan komputasi, *storage*, dan sumber daya jaringan dari sistem *back end* yang dibuat. Rancangan sistem *back end* tersebut dapat menyediakan lingkungan praktikum berupa laboratorium virtual jaringan komputer.

1.2 Rumusan Masalah

Berdasarkan permasalahan yang diangkat pada bagian latar belakang, maka rumusan masalah dapat disusun sebagai berikut:

1. Bagaimana menyediakan komputer virtual berbasis *container* sebagai perangkat praktikum mahasiswa?
2. Bagaimana mekanisme untuk mengakses komputer virtual?
3. Bagaimana melakukan sinkronisasi antara *back end* dan *front end* dari sistem laboratortium virtual?

1.3 Tujuan

Tujuan dari penelitian ini adalah mengatasi masalah keterbatasan waktu dan daya tampung laboratorium serta memberikan keefektifan kepada semua pihak baik mahasiswa, dosen, maupun asisten dalam melakukan praktikum.

1.4 Manfaat

Manfaat Perancangan Sistem Laboratorium ini adalah sebagai berikut:

1. Bagi penulis
   - Mengaplikasikan ilmu yang diperoleh selama mengikuti perkuliahan
   - Mendapat pengetahuan tentang perancangan sistem laboratorium virtual.
2. Bagi pengguna
   - Menjadi referensi penelitian terkait perancangan sistem laboratorium virtual.
   - Penelitian dapat menjadi referensi desain infrastruktur terkait sistem laboratorium virtual.

1.5 Batasan Masalah

Agar permasalahan yang dirumuskan lebih terfokus, maka penelitian ini dibatasi oleh hal-hal sebagai berikut :

1. Sistem ini hanya diterapkan di laboratorium Jaringan Komputer FILKOM.
2. Sistem yang diusulkan hanya menyediakan modul pengujian untuk praktikum premrograman *socket*.
3. *Back end* untuk sistem ini menggunakan Docker Engine dan Openstack.
4. *Front end* untuk sistem ini berbasis web yang di tulis menggunakan bahasa pemrograman PHP dan HTML.
5. Komputer virtual atau *instance* yang digunakan adalah *container*.
6. Hak akses untuk *instance* terbatas oleh hak akses yang diberikan Docker.
7. Penyimpanan data sistem ini Menggunakan *Database Management System* (DBMS) MySql.

1.6 Sistematika Pembahasan

Sistematika pembahasan dari penyusunan penelitian yang direncanakan adalah sebagai berikut:

BAB I PENDAHULUAN

Pendahuluan terdiri dari latar belakang, identifikasi dan pembatasan masalah, rumusan masalah, tujuan dan manfaat serta sistematika pembahasan dari proyek akhir ini. Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya setiap tahunnya menerima ratusan mahasiswa baru. Besarnya jumlah dan terus meningkatnya jumlah mahasiswa di FILKOM ini berimbas pada masalah kurangnya sarana infrastruktur baik kelas maupun laboratorium yang memadai. Pembuatan laboratorium virtual merupakan salah satu metode untuk mengatasi masalah tersebut. Laboratorium secara *online* yang dapat diakses dari mana saja dapat memberikan keluasaan praktikan dan meningkatkan efektivitas dosen/asisten dalam menyampaikan materinya. Untuk mewujudkan hal tersebut maka diperlukan sistem laboratorium virtual baik *front end* maupun *back end* yang saling mendukung satu sama lainnya sehingga laboratorium virtual ini dapat mengatasi masalah-masalah yang ada. Dalam penelitian ini sistem laboratorium virtual yang dibuat dikhususkan ke praktikum jaringan komputer. Selain itu sebuah

sistematika pembahasan digunakan sebagai acuan untuk melakukan pembahasan terhadap penelitian yang dilakukan secara sistematis.

BAB II LANDASAN KEPUSTAKAAN

Pada Bab II dilakukan pengkajian pustaka terhadap teori-teori yang berkaitan dan menunjang dalam penyelesaian penelitian ini. Perancangan dan Implementasi Arsitektur Laboratorium Virtual menjadi fokus pada penelitian ini. Virtualisasi digunakan sebagai metode untuk membuat komputer virtual, sedangkan Docker dan Openstack digunakan sebagai perangkat untuk membuat hal tersebut.

BAB III METODOLOGI

Pada Bab III akan dijelaskan mengenai langkah-langkah membangun arsitektur sistem. Penjelasan langkah-langkah perencanaan, implementasi, pengujian, serta analisis kebutuhan sistem dibahas secara umum.

BAB IV HASIL

Pada Bab IV dijelaskan mengenai penerapan dari arsitektur sistem secara terperinci dengan langkah-langkah pengerjaan serta menampilkan gambar-gambar dari implementasi yang dilakukan. Implementasi dilakukan berdasarkan sistematika yang dibuat pada bab sebelumnya. Berikutnya, pengujian terhadap sistem dilakukan menggunakan skenario yang telah dibuat pada bab sebelumnya.

BAB V PEMBAHASAN

Pada Bab V disajikan data hasil pengujian dan analisis terhadap implementasi arsitektur sistem laboratorium virtual yang telah dibangun. Hasil pengujian didapatkan dari pengujian yang telah dilakukan sesuai dengan skenario pengujian yang telah ditentukan. Dari hasil tersebut dapat dilakukan analisis terhadap kinerja sistem, serta mengetahui bagaimana sistem dapat mencapai tujuan

BAB VI PENUTUP

Kesimpulan dari pelaksanaan penelitian ini dibuat berdasarkan hasil pengujian dan analisis terhadap perancangan dan implementasi arsitektur sistem laboratorium virtual purwarupa praktikum jaringan komputer yang dibangun dan dirangkum pada bab penutup. Untuk pengembangan dari hasil dalam penelitian ini, maka diberikan saran-saran untuk perbaikan dan penyempurnaan dari penelitian ini.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Penelitian terkait

Penelitian-penelitian sebelumnya yang terkait dengan perancangan sistem laboratorium virtual dilakukan untuk mengatasi masalah keterbatasan sumber daya manusia, efektifitas biaya serta daya tampung yang tersedia. Penelitian ini dilakukan oleh Patcharee Basu dkk. (Basu, et al., 2013) yang menggabungkan *remote computer laboratory* dan model pembelajaran jarak jauh. Sistem yang dibangun bertujuan untuk memberikan solusi terhadap permasalahan yang dijelaskan pada BAB I.

Sistem ini menggunakan teknologi virtualisasi serta StarBED *computing testbed* (Basu, et al., 2013) yang dapat menghasilkan daya tampung laboratorium yang lebih besar dengan peralatan serta biaya yang lebih sedikit. Komponen utama dalam sistem ini adalah adanya suatu sistem yang secara otomatis dapat memberikan koreksi serta masukan terhadap pekerjaan seseorang, sehingga dapat memberikan efektifitas waktu dalam pembelajaran.

Untuk latihan dengan laboratorium secara remote, sistem ini menggunakan SSH untuk pengguna bisa terhubung dengan *virtualization server* dan menjalankan sebuah *script* untuk meng-akses *virtual machine*. Sedangkan StarBED digunakan untuk pelatihan kepada 42 pengguna dari sebuah *cluster* yang terdiri dari 67 StarBED *node* (Pentium 4 3,2 GHz, 8 GB RAM). StarBED menyediakan sebuah *testbed* nyata dengan pengaturan manajemen yang fleksibel (Basu, et al., 2013).

Kemudian penelitian yang terkait dengan laboratorium virtual berikutnya dilakukan oleh Le Xu dkk. yang berbasis awan(*cloud based)* yang disebut dengan V-Lab. Sistem ini menyediakan sebuah lingkungan eksperimen berupa perangkat untuk eksperimen materi keamanan jaringan yang dapat dikonfigurasi ulang memanfaatkan teknologi virtualisasi(seperti Xen atau KVM) dan *software defined networking solutions* (SDN) seperti *switch* OpenFlow. Siswa (pengguna) dapat secara aman meng-akses sistem melalui OpenVPN, dan siswa dapat meng-kontrol *virtual machine* secara *remote*, serta melakukan tugas eksperimen (Xu, et al., 2014).

Perangkat fisik dari sistem V-Lab ini terdiri dari sebuah *cluster* dari server awan (*cloud server*) dengan kemampuan performa yang tinggi dan mendukung virtualisasi, sebuah switch HP OpenFlow, sebuah server *Storage Area Network* (SAN) iSCSI yang menyediakan *storage* untuk *virtual machine* dan *backup* redundansi, serta *Uninterrupable power supply* (UPS). Sistem mampu menyediakan sampai dengan 1000 *virtual machine.* Komponen utama sistem *back-end* dari V-Lab ini adalah *Xen cloud platform* (XCP) dan OpenStack. Keduanya XCP dan OpenStack adalah platform virtual *open-source* (Xu, et al., 2014).

Sistem ini juga menyediakan rangkaian eksperimen, fase yang pertama eksperimen yang meliputi pengetahuan dasar jaringan, seperti penggunaan SSH dan VNC untuk mengakses *remote host*, praktikum dengan perintah seperti PING

dan Ipconfig. Sedangkan fase yang kedua, melanjutkan fase yang pertama namun dengan masalah keamanan jaringan yang lebih kompleks, seperti serangan SSL-strip-based MITM, pemfilteran paket berbasis Iptables, dan *Snort-based intrusion detection* (Xu, et al., 2014).

Kemudian penelitian yang dilakukan oleh Dzulqarnain (Dzulqarnain, 2015) dalam lingkungan pembelajaran laboratorium jaringan Fakultas Ilmu Komputer, Universitas Brawijaya. Sistem yang dibangun bertujuan untuk memberikan solusi terhadap keterbatasan kapasitas laboratorium.

Sistem ini menggunakan teknologi virtualisasi dan memberikan fitur untuk penyediaan *virtual machine* untuk mahasiswa, memberikan fitur penyimpanan VM (*save state*) sehingga mahasiswa dapat menyimpan perkejaannya dan sewaktu-waktu dapat melanjutkan kembali pekerjaan yang sebelumnya, serta memberikan kebutuhan praktikum jaringan komputer berupa materi praktikum dan VM yang dapat digunakan untuk proses praktikum (Dzulqarnain, 2015).

Teknologi virtualisasi yang digunakan dalam penelitian tersebut menggunakan KVM *(Kernel-based Virtual Machine),* yaitu salah satu teknologi virtualisasi penuh *(full virtualization)* yang dikembangkan oleh Linux (Dzulqarnain, 2015). Teknologi virtualisasi ini meng-emulasi semua perangkat keras dan menempatkan sumber daya ke dalam CPU. Sehingga dibutuhkan kapasitas dan *memory* yang besar, karena *virtual machine* berbagi daya dengan komputer fisik (Tirtawidjaja, 2014).

Penelitian ini menghasilkan sistem yang dapat membantu mengatasi masalah ketersediaan sumber daya, waktu, dan daya tampung laboratorium yang sebelumnya terbatas oleh biaya, waktu pelaksanaan serta tempat yang disediakan (Dzulqarnain, 2015).

2.2 Virtualisasi

Menurut Kamus Besar Bahasa Indonesia, virtual berarti secara nyata, sedangkan akhiran –isasi menyatakan makna melakukan, proses, usaha, atau kegiatan. Berarti virtualisasi adalah proses menyatakan atau membuat sesuatu menjadi nyata. Sedangkan dalam ilmu komputer, virtualisasi bisa diartikan sebagai pembuatan suatu bentuk simulasi dari sesuatu yang asalnya bersifat fisik, misalnya sistem operasi, perangkat penyimpanan data atau sumber daya jaringan.

Virtualisasi memungkinkan untuk menjalankan beberapa mesin virtual pada mesin fisik tunggal, dengan masing-masing mesin virtual berbagi sumber daya dari satu komputer fisik di beberapa lingkungan. Mesin virtual yang berbeda dapat menjalankan sistem operasi yang berbeda dan beberapa aplikasi pada komputer fisik yang sama (Menken & Blokdijk, 2010). Teknologi virtualisasi saat ini banyak digunakan untuk mendukung *cloud computing scalability*.

Sederhananya, virtualisasi adalah emulasi perangkat keras dalam sebuah platform perangkat lunak. Hal ini memungkinkan satu komputer untuk mengambil peran beberapa komputer (Furht & Escalante, 2010).

Virtualisasi perangkat keras mengacu pada upaya menciptakan mesin virtual (*Virtual Machine*) yang bekerja layaknya sebuah komputer lengkap dengan sistem operasi. Penggunaan teknologi mesin virtual dimanfaatkan untuk menyelesaikan masalah heterogenitas lingkungan (perbedaan versi *library* atau *tool* dari aplikasi). Kemudian saat ini muncul teknologi virtualisasi baru yaitu *container* yang bisa menjadi alternatif untuk menyelesaikan masalah heterogenitas lingkungan (Adiputra, 2015). Salah satu perbedaan dari kedua teknologi virualisasi tersebut adalah pada penggunaan sistem operasi. *Virtual machine* menggunakan sistem operasi secara penuh beserta *kernel*, sedangkan *container* hanya menyediakan *library* yang dibutuhkan untuk menjalankan aplikasi tertentu dan berbagi kernel dengan komputer *host*.

2.2.1 Virtual Machine

IBM mendefinisikan *virtual machine* sebagai sebuah salinan dari fisik *hardware* yang utama dari sebuah mesin yang terlindung dan terisolasi secara penuh (Semnanian, et al., 2011). Agar fisik perangkat keras yang utama bisa berkomunikasi dengan dengan *virtual machine*, sebuah *layer* dibutuhkan diantara mereka untuk mengakomodasi komunikasi.

Virtual machine monitor (VMM) atau *hypervisor* adalah *layer* perangkat lunak yang menyediakan virtualisasi. Dalam dunia virtualisasi, komputer fisik merujuk sebagai sebuah *host* dan *virtual machine* yang berada pada komputer tersebut disebut *guest* (Semnanian, et al., 2011). Sebuah VMM dapat berjalan pada perangkat keras (*native* VM) atau di atas sebuah sistem operasi (*hosted* VM).

Teknologi *virtual machine* memiliki beberapa keunggulan, antara lain:

- Dari segi keamanan VM memiliki perlindungan yang lengkap pada berbagai sistem sumber daya, yaitu dengan meniadakan pembagian sumber daya secara langsung, sehingga tidak ada masalah proteksi dalam VM.
- Memungkinkan untuk mendefinisikan suatu jaringan dari *virtual machine* (VM). Tiap-tiap bagian mengirim informasi melalui jaringan komunikasi virtual. Sekali lagi, jaringan dimodelkan setelah komunikasi fisik jaringan diimplementasikan pada perangkat lunak.

Namun konsep *virtual machine* juga memiliki kesulitan:

- Sistem penyimpanan, dalam pembuatan *virtual machine* harus disediakan disk virtual atau yang dikenal *minidisk*, dimana ukuran daya penyimpanan identik dengan ukuran sebenarnya. Dengan demikian, pendekatan VM juga menyediakan sebuah antarmuka yang identik dengan perangkat keras yang mendasari.

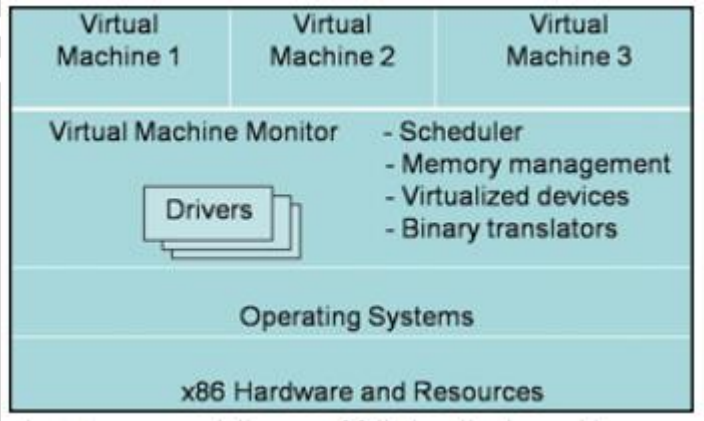
Menurut Suresh ide dibalik virtualisasi adalah untuk mem-virtualisasi sumber daya perangkat lunak atau perangkat keras menggunakan solusi dengan metode virtualisasi pada layer perangkat lunak atau metode virtualisasi pada tingkat perangkat keras (Suresh & Kannan, 2014).

2.2.1.1 Virtualisasi Perangkat Lunak

Solusi ini di-implementasikan oleh layer perangkat lunak dan tidak memerlukan perubahan dari arsitektur perangkat keras. Berdasarkan teknik yang digunakan untuk virtualisasi sumber daya CPU, kontrol layer virtualisasi, dan jumlah porsi eksekusi perangkat keras dan perangkat lunak, virtualisasi perangkat lunak dibagi menjadi virtualisasi penuh (*full virtualization*) dan virtualisasi paruh (*para-virtualization*) (Suresh & Kannan, 2014).

Virtualisasi Penuh (Full Virtualization)

Virtualisasi penuh adalah virtualisasi yang pada layer virtualisasi berjalan sebagai bagian dari sistem operasi, yang mensimulasikan seluruh perangkat keras ke dalam konstruksi perangkat lunak. Konstruksi ini berfungsi sebagai perangkat keras aslinya. Perangkat lunak yang didesain untuk perangkat keras tersebut akan berfungsi seperti komputer sebenarnya (Suresh & Kannan, 2014).

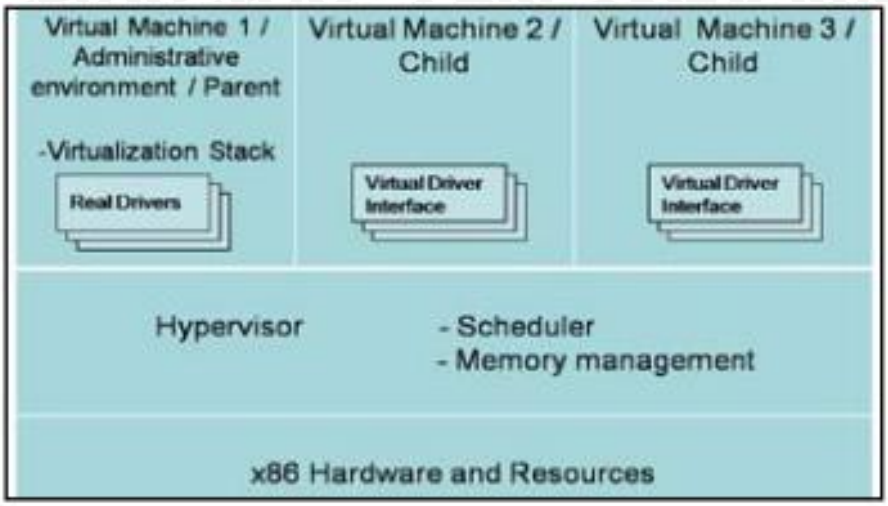


Gambar 2.1 Virtualisasi Penuh **(Suresh & Kannan, 2014)**

Solusi yang menerapkan virtualisasi penuh mencakup keluarga *hypervisor* VMWare, Virtualbox dari Oracle, QEMU dari Fabrice Bellard, dan KVM dari Red Hat (Jones, 2010).

Virtualisasi Paruh (Para-virtualization)

Para-virtualisasi adalah teknik virtualisasi yang dilakukan dengan memodifikasi atau meng-kompilasi *guest* OS dan menjalankannya sebagai proses biasa tanpa perlu akses perangkat lunak. Layer virtualisasi yang berjalan langsung pada perangkat keras bertanggung jawab untuk penjadwalan sumber daya dan *multiplexing* VM (Suresh & Kannan, 2014).



Gambar 2.2 Para-virtualisasi **(Suresh & Kannan, 2014)**

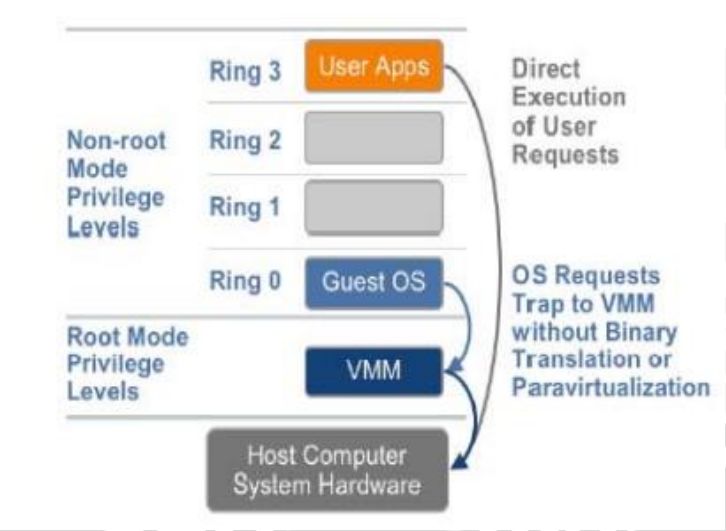
Solusi yang menerapkan para-virtualisasi mencakup *RedHat's Xen,* TRANGO, Wind River, dll (Jones, 2010).

2.2.1.2 Virtualisasi Perangkat Keras

Solusi ini memberikan peningkatan desain pada perangkat keras untuk membantu mengurangi kerumitan dan overhead dari virtualisasi perangkat lunak dalam rangka untuk memaksimalkan keuntungan dari virtualisasi (Suresh & Kannan, 2014).

Hardware Assisted Virtualization

Hardware Assisted Virtualization adalah jenis virtualisasi penuh di mana arsitektur *microprosesor* memiliki instruksi khusus untuk membantu virtualisasi perangkat keras. Instruksi ini memungkinkan konteks virtual untuk melakukan penyesuaian sehingga *guest* dapat menjalankan instruksi *privileged* langsung pada prosesor tanpa mempengaruhi *host*. Virtualisasi ini memanfaatkan kemampuan perangkat keras yang tersedia di dalam prosesor-prosesor termutakhir dari Intel (Intel VT) dan Advanced Micro Devices (AMD-V) untuk menyediakan performa mendekati sistem aslinya. Keduanya mem-fokuskan pada instruksi *privileged* yang memungkinkan mereka untuk berjalan dalam modus eksekusi prosesor model baru yang menjadikan VMM atau *hypervisor* berada di atas perangkat keras *host* dan di bawah *guest* OS (Semnanian, et al., 2011).



Gambar 2.3 Pendekatan Hardware Assisted **(Semnanian, et al., 2011)**

2.2.2 Container

Container berarti memberikan isolasi dan manajemen sumber daya dalam lingkungan Linux. Sebuah container dari sistem operasi memberikan sebuah cara untuk meng-isolasi sebuah proses dari keseluruhan sistem. Container menjalankan instruksi asli ke *core* CPU, menghilangkan kebutuhan untuk emulasi pada tingkat instruksi. Tidak ada ketergantungan pada emulasi *hardware* memberikan manfaat kinerja yang lebih baik jika dibanding dengan *full virtualization,* tetapi *container* terbatas pada sistem operasi yang dapat digunakan sebagai sistem operasi *guest* (Dua, et al., 2014).

Tabel 2.1 membandingkan *Virtual Machine* dan *Container* pada beberapa faktor seperti kinerja, keamanan isolasi, jaringan, penyimpanan (Dua, et al., 2014).

Tabel 2.1 Perbandingan *Virtual Machine* dan *Container* **(Dua, et al., 2014)**

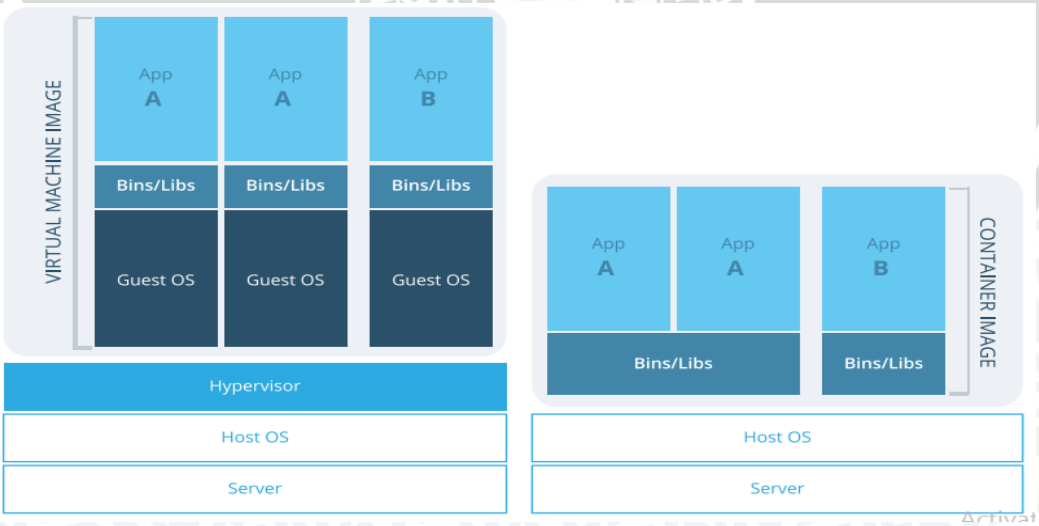
Parameter	Virtual Machine	Container
Guest OS	Setiap VM berjalan pada *hardware* virtual dan *Kernel* dimuat ke dalam *memory region* sendiri	Semua guest OS berbagi *Kernel* yang sama. Image kernel dimuat ke dalam *memory* fisik
Komunikasi	Melalui *Ethernet device*	Mekanisme IPC standar seperti *signal, pipe, socket*
Keamanan	Bergantung pada implementasi *hypervisor*	Kendali akses *mandatory* dapat dimanfaatkan
Performa	Mesin Virtual mendapat *overhead* kecil sebagai instruksi dari Machine dijabarkan dari *Guest* ke *Host OS*	Container memberikan kinerja yang mendekati aslinya dibandingkan dengan *underlying Host OS*

Isolasi	Berbagi *library*, file dan lain-lain antar tamu dan antara *guest* dengan *host* OS tidak memungkinkan	Subdirektori dapat dengan transparan dipasang dan bisa dibagi pakai
Startup Time	Membutuhkan beberapa menit untuk *boot-up*	Membutuhkan beberapa detik untuk *boot-up*
Storage	VM membutuhakn lebih banyak *storage* karena keseluruhan *kernel* dan program terkait harus ter-instal dan dijalankan	Container membutuhkan storage yang lebih kecil karena *base* OS adalah bagi pakai

Adapun beberapa kekurangan dari *container*, seperti keterbatasan pada dukungan sistem operasi, visibilitas, mitigasi risiko, administrasi, dan *orchestration*. Kemudian kemampuan *container* untuk penggunaan dengan skala lebih besar yang belum terbukti (Patrizio, 2015). Container juga memiliki kelemahan pada sisi isolasi, dikarenakan *container* berbagi sebuah kernel dan komponen dari sistem operasi. Sehingga kelemahan dan serangan memiliki potensi yang lebih besar untuk terjadi (Bigelow, 2015).

2.3 Docker

Docker adalah sebuah *platform* terbuka untuk pengembang, administarator sistem atau siapapun yang bertujuan menggunakan sebuah *platform* untuk membangun, mendistribusikan dan menjalankan aplikasi dengan mudah dan cepat dalam lingkungan sistem operasi yang terpisah dalam Docker (Seo, et al., 2014).



Gambar 2.4 Arsitektur Virtual Machine dan Docker Container **(Cacciatore, et al., 2015)**

Virtual machine menjalankan sistem operasi secara penuh dengan manajemen memori sendiri yang terpasang dengan *driver* perangkat virtual terkait. Dalam *virtual machine*, sumber daya ditiru untuk *guest* OS, dan *hypervisor*, yang memungkinkan untuk menjalankan banyak *instance* dari satu atau lebih sistem operasi secara paralel pada satu mesin (*host*). Setiap *guest* OS berjalan sebagai entitas sendiri dari sistem *host*.

Di sisi lain Docker *container* dijalankan dengan menggunakan Docker *engine* bukan *hypervisor*. Karena tidak menggunakan sistem operasi secara penuh, *container* lebih kecil dari *virtual machine* dan memungkinkan untuk *start up* yang lebih cepat dengan performa yang lebih baik, isolasi yang lebih rendah, dan kompatibilitas yang lebih besar, karena container berbagi *kernel* dengan *host*.

2.4 Laboratortium Virtual

Pada umumnya, sebuah laboratorium virtual adalah sebuah lingkungan yang terdistribusi yang menyediakan akses ke berbagai macam perangkat sains dan sumber daya untuk komputasi (Davoli, et al., 2010).

Adapun beberapa kategori dari laboratorium virtual yang telah ada, sebagai berikut:

a. *Virtual Application Laboratories*: laboratorium ini menggunakan virtualisasi *desktop*, dimana simulasi dan penyelesaian masalah terbatas oleh algoritma yang telah ditetapkan. Model laboratorium ini biasanya tidak mengizinkan siswa untuk menyimpan data atau kondisi dari *remote server,* oleh karena itu siswa harus menyelesaikan tugas dalam satu sesi (Xu, et al., 2014).

b. *Shared-Host Laboratories*: laboratorium ini dibangun dengan sebuah *pool* komputer yang tetap dengan akses secara *remote desktop*. Setiap komputer mendukung untuk beberapa siswa login secara bersamaan (Xu, et al., 2014).

c. *Single-VM laboratories*: laboratorium ini memberikan VM yang telah ditentukan (*template*) bagi siswa. Sebuah VM dapat diminta atau dibuang oleh siswa, atau siswa dapat membuat VM mereka sendiri. *Single-VM* biasanya tidak memiliki portal manajemen yang memberikan sumber daya yang dapat disesuaikan untuk tiap pengguna (siswa). Selain itu, VM biasanya berjalan pada komputer *desktop* atau laptop biasa, dan mereka tidak dapat mendukung untuk lingkungan jaringan multi VM yang lebih rumit (Xu, et al., 2014).

d. *Multi-VM laboratories:* laboratorium ini memberikan banyak VM yang dapat berjalan di sistem awan (*cloud*) atau di komputer siswa. Lingkungan *multi-VM* memungkinkan siswa untuk membangun konfigurasi untuk eksperimen yang lebih rumit. Namun, laboratorium ini tidak memberikan konfigurasi jaringan yang fleksibel, isolasi yang cukup, atau kemampuan untuk konfigurasi ulang (Xu, et al., 2014).

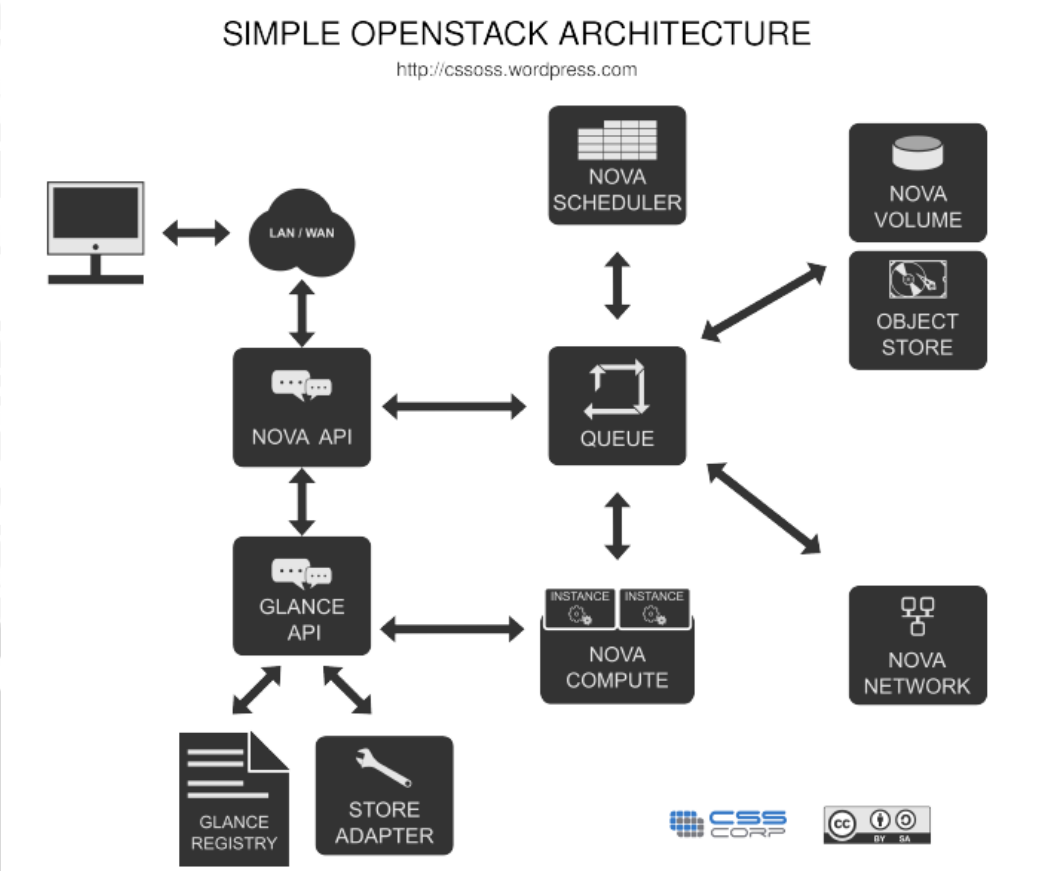
e. *Multi-VM and Multinetwork laboratories:* laboratorium ini memanfaatkan sepenuhnya kapasitas virtualisasi dari virtualisasi awan (*cloud*) untuk memberikan lingkungan eksperimen dengan banyak VM dan beberapa jaringan virtual yang ter-dedikasi. Sistem ini menawarkan portal manajemen berbasis web untuk instruktur dan siswa dapat mengelola dan membuat sumber daya virtual yang *user-friendly* (Xu, et al., 2014).

2.5 Openstack

Openstack adalah sebuah sistem operasi awan (*Cloud OS*) berskala besar yang merupakan kolaborasi dari para pengembang dan ahli dalam komputasi awan(*cloud computing*) yang meproduksi *platform cloud computing* terbuka untuk publik dan *private cloud* (Kumar, et al., 2014). OpenStack mengontrol lingkungan komputasi besar, *storage*, dan sumber daya jaringan di seluruh *data center*, semua dikelola melalui *dashboard* yang memberikan kontrol *administrator* sekaligus memberdayakan pengguna mereka ke sumber daya penyediaan melalui antarmuka web (Web GUI).

Openstack memungkinkan perusahaan atau siapa saja untuk membangun dan menawarkan layanan komputasi awan(*cloud computing*) dengan memanfaatkan perangkat lunak *open source* yang bebas. Pada awalnya proyek ini berfokus pada menawarkan infrastruktur sebagai layanan (IaaS), adapun komponen Openstack meliputi:

- Openstack Compute: perangkat lunak untuk mengatur , mengelola, dan membuat *virtual machine*. Perangkat lunak yang memberikan layanan ini disebut "Nova" (Radez, 2015).
- Openstack Object Store: perangkat lunak untuk *redundant storage* dari objek statis. Perangkat lunak yang memberikan layanan ini disebut “Swift” (Radez, 2015).
- Openstack Image Service: memberikan layanan query dan layanan penyimpanan untuk *virtual disk image*. Perangkat lunak yang memberikan layanan ini disebut “Glance” (Radez, 2015).
- Openstack Storage Service: memberikan layanan manajemen blok penyimpanan. Perangkat lunak yang menyediakan layanan ini disebut “Cinder” (Radez, 2015).
- Openstack UI Service: memberikan layanan antarmuka berbasis web. Perangkat lunak yang memberikan layanan ini disebut “Dashboard” (Radez, 2015).
- Openstack Network Service: memberikan layanan manajemen jaringan. Perangkat lunak yang memberikan layanan ini disebut “Neutron” (Radez, 2015).



Gambar 2.5 Arsitektur Openstack

2.6 Virtual Network Computing

VNC (*Virtual Network Computing*) adalah sebuah sistem *monitoring* untuk melihat secara jarak jauh dari layar atau peristiwa pada komputer melalui jaringan komputer terdistribusi seperti Internet. Dalam VNC, mesin server menyediakan tidak hanya aplikasi dan data tetapi juga lingkungan *desktop* secara keseluruhan yang dapat diakses dari mesin yang terhubung dengan internet dengan menggunakan *software NC* sederhana (Richardson, et al., 1998).

Tujuan dari membangun VNC ini sendiri adalah untuk mempermudah dalam pengawasan atau memonitoring kinerja dari suatu sistem. VNC ini sendiri menggunakan protocol yang sederhana berbasis RFB (*Remote Frame Buffer*) (Tomar & Shaney, 2013). VNC ini tersedia di sistem operasi Linux dan Microsoft Windows. VNC ini bersifat *cross-platform* dimana perangkat lunak ini dapat di gunakan untuk komputer dengan sistem operasi yang berbeda. Misalnya menggunakan VNC untuk melakukan koneksi komputer dengan OS Windows Vista melalui Ubuntu Linux. Keungulan VNC dibandingkan dengan perangkat lunak lainnya:

- *Multi platform*, server ini dapat di gunakan dengan baik di lingkungan Windows, Linux, Beos, Macintos, Unix dll. VNC *client* dan VNC *server* dapat

saling di akses misalnya dari sistem Windows ke sistem Linux maupun sebaliknya.

- *Client-server*. Terdiri dari apikasi *server* dan *client* dan harus di instal di kedua sisi.
- *HTTP support*. VNC dapat di akses menggunakan *default port* 5900 atau 5901, untuk TCP maupun *port* 5800 atau 5801 untuk HTTP. Jadi sebuah *VNC server* juga dapat diakses oleh VNC client menggunakan sebuah *browser* seperti Mozilla Firefox, Opera, dan Internet Explorer dengan menggunakan *java aplet*.
- Transparan. Apabila sebuah komputer Windows dipasang VNC server, akan muncul sebuah icon kecil logo VNC di sebelah kanan taskbar yang akan berubah warna apabila komputer tersebut sedang diakses. VNC juga mengharuskan kita memasang password untuk bisa diaktifkan. Sebelum password dipasang, ia tidak akan mau bekerja.

2.7 Munin

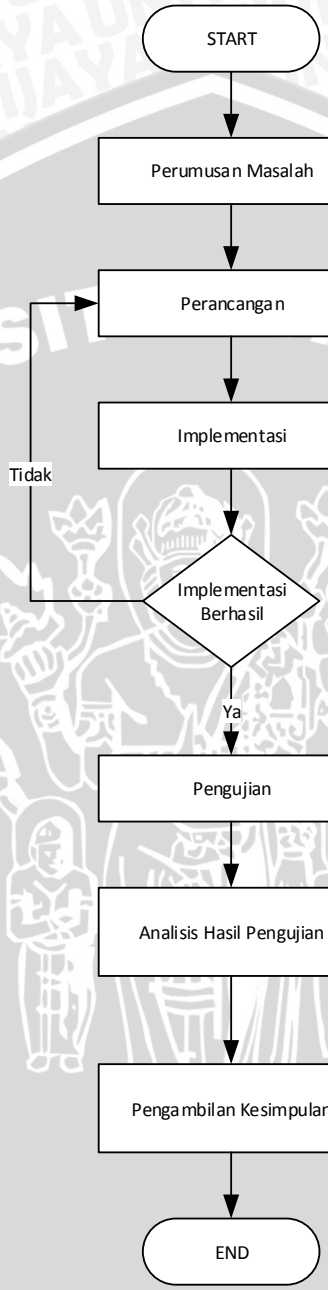
Munin adalah alat *monitoring* sumber daya jaringan yang dapat membantu menganalisis kecenderungan sumber daya. Hal ini dirancang untuk sangat *plug and play*. Sebuah instalasi standar yang menyediakan banyak grafik yang baik (Jawwad, 2014).

Munin ditulis dengan Perl dan menggunakan RRDtool untuk membuat grafik yang dapat diakses melalui antarmuka web. Sekitar 500 *plugin monitoring* tersedia dalam Munin, hal ini dimaksudkan untuk mempermudah menentukan “apa yang berbeda” ketika masalah kinerja terjadi dan untuk memberikan visibilitas kapasitas dan sumber daya (Jung, 2009).

Munin *monitoring* meninjau semua komputer dan mengingat apa yang dilihat. Hal ini menyajikan semua informasi dalam grafik melalui antarmuka web. Penekanannya adalah pada kemampuan *plug and play*. Menggunakan munin dapat dengan mudah memantau kinerja komputer server, jaringan, SAN, dan aplikasi (Jawwad, 2014).

BAB 3 METODOLOGI

Pada bab metodologi penelitian dijelaskan langkah-langkah yang akan dilakukan pada penelitian ini, yakni sebagai berikut:



Gambar 3.1 Metodologi Penelitian

3.1 Perumusan Masalah

Permasalahan yang dihadapi dalam penelitian ini adalah bagaimana menerapkan sistem laboratorium jaringan komputer virtual secara *online* yang dapat mengakomodir kebutuhan mahasiswa akan sarana dan prasarana layaknya seperti ketika melaksanakan praktikum di laboratorium. Seperti yang telah

dijelaskan sebelumnya praktikum yang dilaksanakan di laboratorium memiliki beberapa kekurangan seperti daya tampung kelas yang terbatas, sarana dan prasarana yang tidak mencukupi serta waktu penggunaan laboratorium yang terbatas. Salah satu cara yang dapat digunakan untuk mengatasi permasalahan ini adalah dengan membuat laboratorium virtual yang dapat mengakomodir kebutuhan para penggunanya. Dalam mengakomodir kebutuhan penggunanya, laboratorium virtual ini mengadopsi metode virtualisasi, dimana setiap pengguna dalam hal ini mahasiswa mempunyai komputer virtual tersendiri sesuai dengan kebutuhan praktikum sehingga setiap mahasiswa dapat melakukan praktikum secara efektif dan efisien.

3.2 Studi Literatur

Studi literature dilakukan untuk menambah referensi dan pengetahuan yang diperlukan dalam mengerjakan penelitian dan penulisan laporan skripsi. Adapun yang perlu menjadi bahan studi literatur adalah dasar-dasar teori untuk dapat mendesain dan menerapkan sistem laboratorium virtual dan cara pengujian sistem yang meliputi:

1. Virtualisasi
   - Docker Container
   - Novadocker
   - Openstack
2. VNC
3. *Performance Testing*

3.3 Perancangan

Proses perancangan pada penelitian ini dibagi menjadi dua bagian, yaitu pada tahap pertama adalah proses analisis kebutuhan, tahap kedua adalah perancangan *back end* sistem laboratorium jaringan komputer. Tahapan analisis kebutuhan terdiri dari analisis kebutuhan akan praktikum jaringan komputer, apa yang membuat praktikum tersebut dapat terlaksana, dan membuat daftar kebutuhan dari perangkat keras dan perangkat lunak yang digunakan untuk melakukan penelitian ini. Tahap perancangan sistem *back end* terdiri dari perancangan desain sistem laboratorium secara umum yang menjelaskan mengenai proses kerja dari sistem secara umum serta perancangan sistem praktikum jaringan komputer virtual yang menjelaskan mengenai alur kerja sistem serta bagaimana sistem menjalankan kegiatan praktikum.

3.3.1 Analisa Kebutuhan Sistem

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dalam pembuatan sistem laboratorium virtual jaringan komputer. Oleh karena itu diperlukan definisi dari proses praktikum itu sendiri serta apa yang

membuat suatu sesi pembelajaran dapat dikatakan sebagai suatu proses praktikum.

Menurut Kamus Besar Bahasa Indonesia praktikum adalah bagian dari pengajaran yang bertujuan agar siswa mendapat kesempatan untuk menguji dan melaksanakan dalam keadaan nyata apa yang diperoleh dalam teori; pelajaran praktik. Praktikum yang dimaksud dalam penelitian ini dikhususkan ke dalam praktikum mata kuliah jaringan komputer. Praktikum dapat dilakukan di ruang kelas maupun laboratorium. Kegiatan praktikum diikuti oleh mahasiswa yang terdaftar dalam mata kuliah yang diselenggarakan oleh jurusan atau program studi. Materi praktikum yang disampaikan kepada mahasiswa dirancang oleh dosen mata kuliah jaringan komputer. Dalam setiap pelaksanaan sesi praktikum mahasiswa mendapat perangkat praktikum berupa komputer. Waktu pelaksanaan sesi praktikum diatur sesuai jadwal mata kuliah.

Dari penjelasan tersebut, maka penulis melakukan analisis kebutuhan mengenai fungsional dari sistem tersebut.

Beberapa kebutuhan tersebut yaitu:

1. Sistem yang dibangun dapat menyediakan komputer virtual untuk sesi praktikum.
2. Mahasiswa dapat menggunakan komputer virtual (*container/instance*) sesuai dengan sesi praktikum yang dipilih.
3. Apabila sesi praktikum telah habis mahasiswa dapat menyimpan pekerjaan mahasiswa.
4. Pekerjaan mahasiswa dimuat/dijalankan kembali ketika sesi praktikum selanjutnya dimulai.
5. Komputer virtual (*container/instance*) yang digunakan harus dapat digunakan untuk melaksanakan praktikum jaringan komputer.
6. Sistem yang dibangun dapat digunakan untuk proses praktikum jaringan komputer.

Dari variable-variabel yang telah disebutkan pada kebutuhan fungsional sistem, maka dibutuhkan beberapa komponen berikut yang bersifat non-fungsional untuk menunjang penelitian ini, dimana komponen-komponen yang dibutuhkan terbagi dalam dua lingkungan, lingkungan perangkat keras dan lingkungan perangkat lunak. Berikut detail dari lingkungan yang dibutuhkan:

- ❖ Lingkungan perangkat keras (*Hardware*)
  - Satu unit PC yang digunakan sebagai server
    - Prosesor : Intel QuadCore
    - Harddisk : 50 GB
    - Memory : 4 GB
    - Sistem Operasi : Ubuntu Server 14.04.3
    - Interface : eth0
    - Alamat IP : 192.168.68.134 / 27
- ❖ Lingkungan perangkat lunak (*Software*)
  - Linux Ubuntu Server 14.04
  - Docker 1.9.1
  - Openstack Kilo

- NoVNC Library

❖ Kebutuhan fitur sistem

- Penyediaan Kelas

  Fitur ini berfungsi untuk dosen atau asisten membuat komputer virtual sesuai dengan jumlah mahasiswa atau praktikan.

- *Remote* Komputer Virtual

  Fitur ini menyediakan fungsi kepada mahasiswa untuk melakukan remote terhadap komputer virtual yang telah disediakan.

- *Start* Komputer Virtual

  Sistem yang dibangun mampu untuk menjalankan komputer virtual secara otomatis.

- *Stop* Komputer Virtual

  Sistem yang dibangun mampu untuk menghentikan atau mematikan komputer virtual

- Menyalakan Kembali Komputer Virtual

  Sistem yang dibangun mampu untuk melakukan proses menyalakan kembali komputer virtual yang sedang dalam keadaan mati.

- *Destroy* Komputer Virtual

  Sistem yang dibangun mampu untuk menghapus komputer virtual yang telah dibuat.

3.3.2 Perancangan *Back End* Sistem Laboratorium Virtual Jaringan Komputer

Perancangan sistem dilakukan agar penelitian dapat berjalan dengan baik dan sistematis. Perancangan pada penelitian ini meliputi beberapa tahapan, yaitu:

3.3.2.1 Identifikasi Aktor

Perancangan identifikasi aktor digunakan untuk mengidentifikasi aktor-aktor yang berperan dalam sistem. Rancangan identifikasi aktor sistem laboratorium virtual jaringan komputer dapat dilihat pada Tabel 3.1.

Tabel 3.1 Identifikasi Aktor

Aktor	Deskripsi
Dosen	Dosen dapat membuat *class* dan mempunyai akses penuh terhadap *class* tersebut.
Mahasiswa	Mahasiswa dapat masuk ke dalam *class* (*enrollment*) yang dibuat oleh dosen dan mengikuti praktikum.
Asisten	Asisten dapat masuk ke dalam kelas untuk me-*monitoring* pekerjaan mahasiswa.

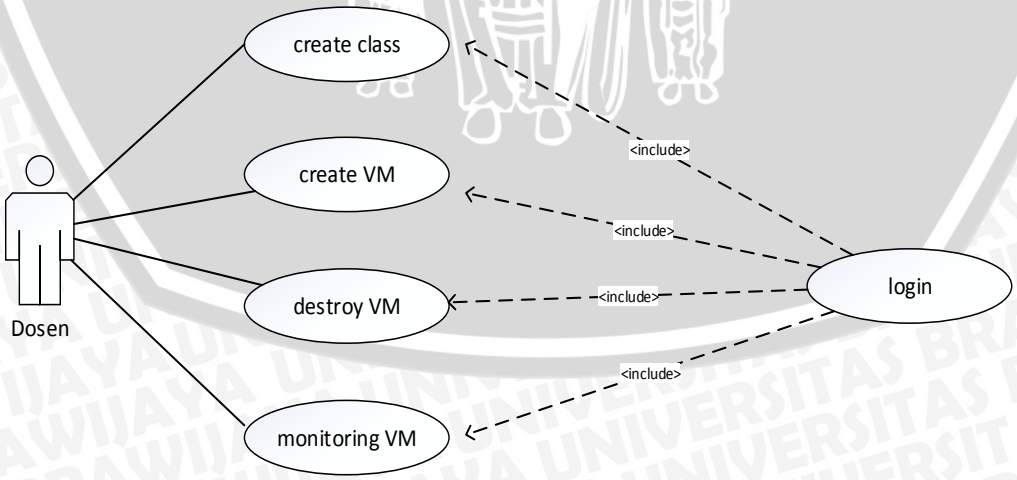
3.3.2.2 Kebutuhan Pengguna

Perancangan identifikasi kebutuhan pengguna untuk mengidentifikasi kebutuhan dari pengguna dalam sistem. Rancangan identifikasi kebutuhan pengguna sistem laboratorium virtual jaringan komputer dapat dilihat pada Tabel 3.2.

Tabel 3.2 Kebutuhan Pengguna

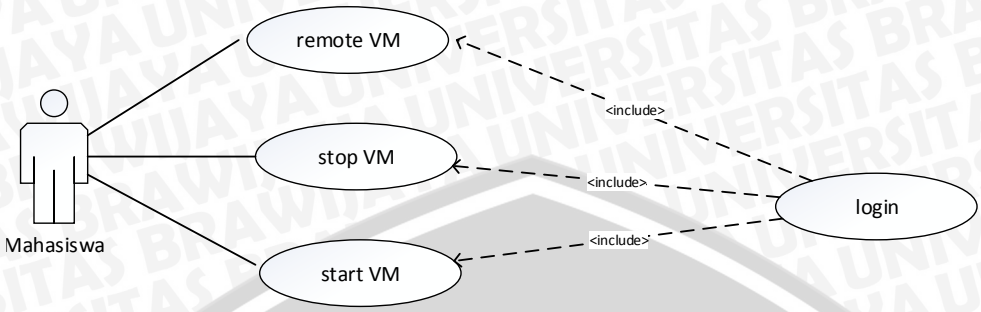
Identifikasi	Kebutuhan	*Use case*
LV_01	Aplikasi menampilkan *form* autentikasi untuk masuk ke dalam sistem	*Login*
LV_02	Aplikasi mampu membuat komputer virtual	*Create* VM
LV_03	Aplikasi mampu memberikan akses komputer virtual	*Remote* VM
LV_04	Aplikasi mampu untuk menjalankan komputer virtual dari kondisi *shutoff* menjadi kondisi *running*	*Start* VM
LV_05	Aplikasi mampu untuk menghentikan/*shutdown* komputer virtual	*Stop* VM
LV_06	Aplikasi mampu untuk menghapus komputer virtual	*Destroy* VM
LV_07	Aplikasi mampu digunakan untuk memonitor komputer virtual	*Monitoring VM*

Gambar 3.2 merupakan diagram *use case* dari aktor dosen yang mengacu dari daftar tabel kebutuhan pengguna.



Gambar 3.2 Use Case Diagram Dosen

Gambar 3.3 merupakan diagram *use case* dari aktor mahasiswa yang mengacu dari daftar tabel kebutuhan pengguna.



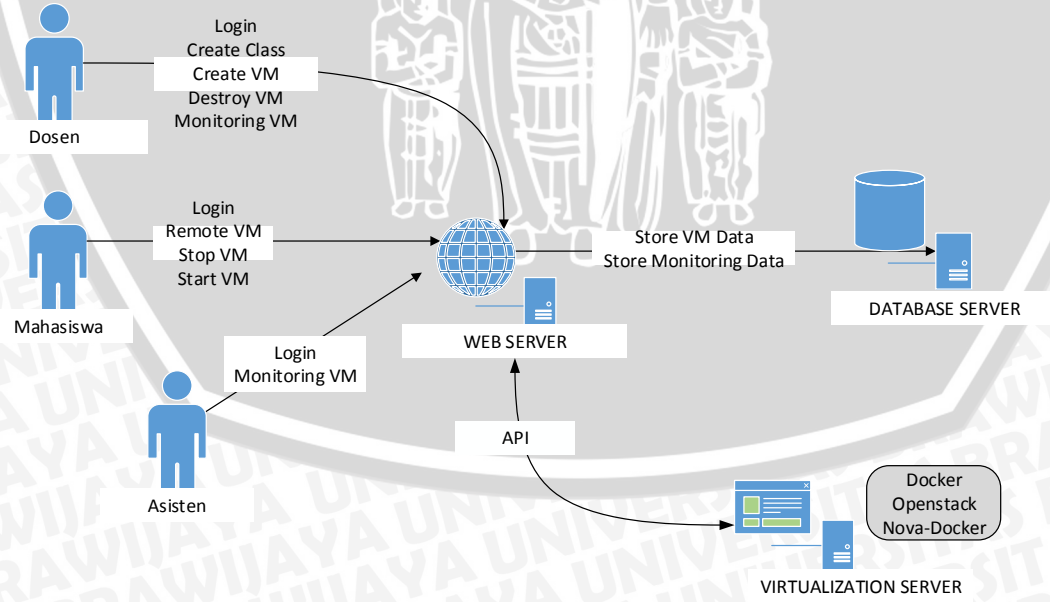
Gambar 3.3 Use Case Diagram Mahasiswa

Gambar 3.4 merupakan diagram *use case* dari aktor asisten yang mengacu dari daftar tabel kebutuhan pengguna.

Gambar 3.4 Use Case Diagram Asisten

3.3.2.3 Perancangan Arsitektur Sistem Laboratorium Virtual Jaringan Komputer

Perancangan ini meliputi bagaimana arsitektur tiap server yang digunakan. Hal ini bertujuan untuk memberikan gambaran mengenai implementasi perangkat keras dari penelitian yang akan dilakukan. Rancangan yang digunakan dalam penelitian ini adalah sebagai berikut:



Gambar 3.5 Topologi Rancangan Implementasi

Gambar 3.5 merupakan rancangan keseluruhan sistem *back-end* dan *front-end* dari sistem laboratorium virtual. Bagian *front-end* mengembangkan pada bagian *web server* yang akan menampung aplikasi berbasis web untuk digunakan sebagai antarmuka ke *user* sistem, dan mengembangkan bagian *database server* untuk memanajemen penyimpanan data dari sistem, sedangkan bagian *back-end* mengembangkan bagian *virtualization server* sebagai media penyimpanan komputer virtual(*container/instance*) dan menyediakan *Application Programming Interface* (API) berupa *output* sebagai hasil proses dari *input* yang diberikan oleh *web server* sebagai sinkronisasi *virtualization server* dan *database server.*

Web server akan menampung *front-end* dari sistem yang berupa aplikasi berbasis web dari sistem laboratorium virtual dimana aplikasi web ini akan menyediakan kebutuhan *user* untuk melaksanakan praktikum jaringan komputer.

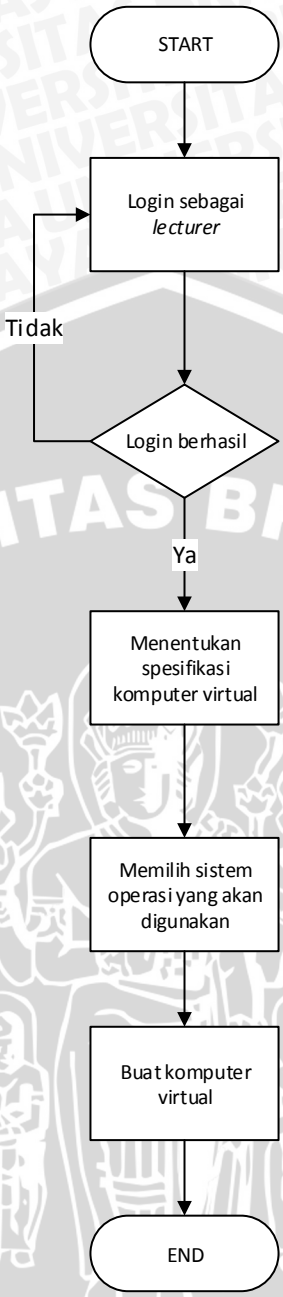
Database *server* akan bertindak sebagai tempat penyimpanan data dari sistem dan juga sebagai alat sinkronisasi dari *back-end* dan aplikasi *front-end*.

Virtualization server akan bertindak sebagai penyedia kebutuhan mahasiswa dalam hal ini komputer virtual atau sistem operasi sebagai media dalam melakukan proses praktikum jaringan komputer. Dalam proses pembuatan komputer virtual tersebut, *virtualization server* akan bekerja sama dengan *web* server dimana ketika pengguna melakukan pemesanan komputer virtual maka *web server* akan mengirimkan *input* data pemesanan ini kepada *virtualization server*. Berdasarkan data inilah *virtualization server* akan membuat komputer virtual. Setelah itu server akan memberikan *output* kepada *web server* yang berupa hasil dari proses *input* yang dimasukkan *web* server kepada *virtualization server*, yang dimana hasil tersebut akan disimpan didalam *database* server.

Pada sistem ini *user* akan terhubung dengan *web server* dimana aplikasi akan memproses perintah *user* sesuai dengan bagian yang diinginkan. *User* yang ingin mengakses komputer virtual yang berada pada *virtualization server*, maka *web server* akan menjalankan *proxy VNC* yang akan digunakan sebagai jembatan penghubung antara komputer virtual dengan *user*. Setelah *user* terhubung dengan komputer virtual, *user* bisa melakukan praktikum dengan mengkonfigurasi komputer virtual sesuai materi yang diberikan.

3.3.2.4 Perancangan Sistem Penyediaan Komputer Virtual

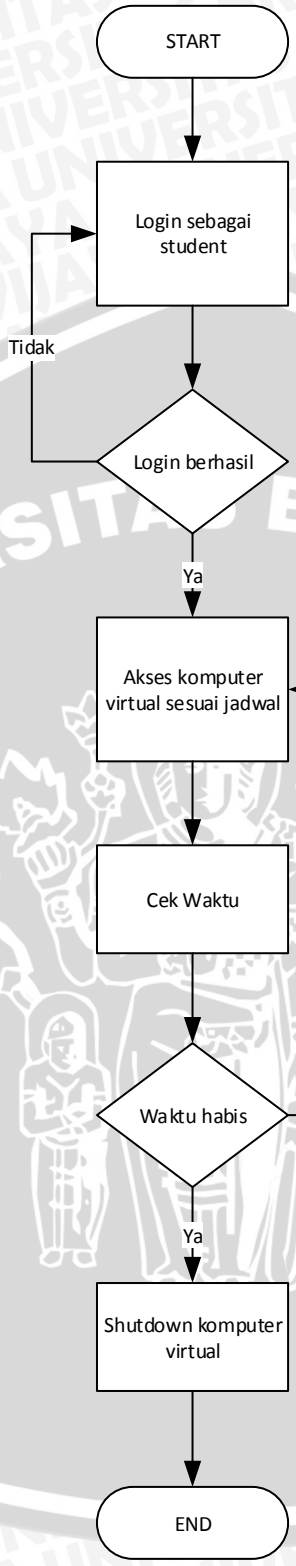
Sistem penyediaan komputer virtual ini berfungsi untuk memberikan komputer virtual kepada mahasiswa untuk digunakan dalam praktikum. Bagian perancangan sistem untuk dosen, dimana sistem ini akan menyediakan fitur penyediaan kelas yang akan menyediakan komputer virtual sesuai kebutuhan beserta kode untuk akses ke komputer virtual. Berikut merupakan *flowchart* cara kerja sistem penyediaan komputer virtual:



Gambar 3.6 Flowchart Penyediaan Komputer Virtual

3.3.2.5 Perancangan Sistem Penyimpanan Pekerjaan Mahasiswa

Setiap mahasiswa kemungkinan akan membutuhkan waktu yang berbeda-beda untuk menyelesaikan tugas praktikum yang diberikan. Oleh karena itu dibutuhkan mekanisme penyimpanan pekerjaan mahasiswa sehingga mahasiswa dapat melanjutkan pekerjaannya tanpa perlu memulai dari awal. Berikut merupakan *flowchart* dari cara kerja sistem penyimpanan pekerjaan mahasiswa:

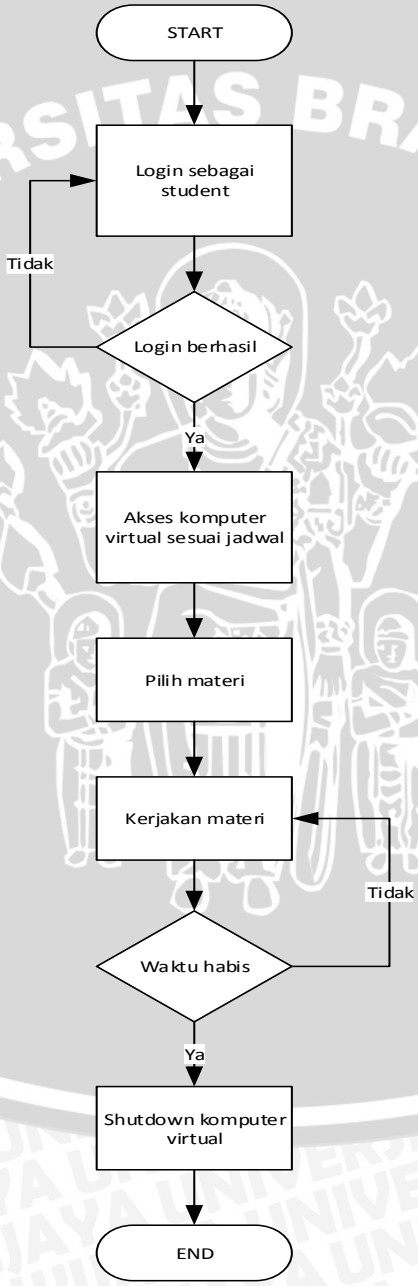


Gambar 3.7 Flowchart Sistem Penyimpanan

3.3.2.6 Perancangan Praktikum Jaringan Komputer

Perancangan materi praktikum digunakan untuk memenuhi kebutuhan mahasiswa dalam melakukan praktikum. Mahasiswa diberikan akses untuk menggunakan komputer virtual sebagai perangkat untuk melakukan praktikum sesuai dengan materi yang diberikan oleh dosen atau asisten.

Materi yang digunakan dalam penelitian ini masih dikhususkan pada materi praktikum jaringan komputer. *Requirement* komputer virtual yang digunakan dalam praktikum akan menyesuaikan dengan materi yang sedang diberikan. Gambar 3.8 merupakan *flowchart* dari cara kerja pengerjaan materi praktikum.



Gambar 3.8 *Flowchart* Praktikum

3.4 Implementasi

Implementasi sistem sesuai dengan perancangan sistem yang telah dibuat sebelumnya. Pada implementasi sistem ini memiliki beberapa tahapan, yaitu:

1. Implementasi perangkat keras dan konfigurasinya.

   Pada tahap ini akan diaplikasikan rancangan implementasi. Konfigurasi dilakukan untuk memastikan perangkat keras telah berjalan dengan baik.

2. Implementasi perangkat lunak pendukung

   Pada tahap ini diaplikasikan perangkat lunak untuk menunjang jalannya sistem. Perangkat lunak tersebut meliputi Ubuntu 14.04 sebagai sistem operasi dari *server*, *Library* NoVNC sebagai perangkat lunak penghubung antara komputer mahasiswa dengan komputer virtual yang dibuat *server*, Docker Engine digunakan untuk penyedia komputer virtual (*container*), dan Openstack mengontrol lingkungan virtualisasi.

3. Implementasi materi praktikum.

   Pada tahap ini diterapkan materi praktikum sesuai materi yang sedang diberikan dengan menggunakan Ubuntu *container* sebagai sistem operasi dari komputer virtual mahasiswa.

Dari beberapa tahapan implementasi di atas, diharapkan sistem dapat berjalan sesuai dengan yang diharapkan, yaitu:

a. Memberikan fitur penyediaan komputer virtual sehingga dapat menyediakan komputer virtual sesuai kebutuhan mahasiswa.

b. Memberikan mekanisme penyimpanan komputer virtual serta fitur pengaturan sumber daya sehingga mahasiswa dapat menyimpan pekerjaannya dan sewaktu-waktu dapat melanjutkan kembali serta dapat menjalankan praktikum dengan lancar.

c. Memberikan kebutuhan praktikum jaringan komputer berupa komputer virtual yang dapat digunakan untuk proses praktikum.

Lingkungan penerapan program disesuaikan dengan pengujian yang dilakukan. Pada penelitian ini pengujian dilakukan dengan menjalankan sistem secara keseluruhan untuk mengetahui apakah sistem mampu untuk mengaplikasikan laboratorium virtual yang dapat menyediakan sarana dan prasarana yang memadai kepada mahasiswa.

3.5 Pengujian

Pengujian pada penelitian ini dilakukan untuk menguji performa kapasitas dan availabilitas sistem. Performa yang diuji adalah berapa waktu yang dibutuhkan sistem untuk menyediakan komputer virtual, berapa lama waktu yang dibutuhkan sistem untuk menyalakan kembali komputer virtual, berapa lama waktu yang

dibutuhkan sistem untuk menyimpan pekerjaan mahasiswa, dan beban kerjanya, serta pengujian sistem untuk praktikum materi jaringan komputer.

3.5.1 Pengujian Penyediaan Komputer Virtual

Skenario pada pengujian penyediaan komputer virtual akan dilakukan dengan menghitung waktu pembuatan komputer virtual berdasarkan jumlah yang dibuat. Jumlah yang akan dibuat berbeda pada tiap skenario pengujiannya. Terdapat 4 skenario pengujian yang akan dilakukan, terlihat seperti pada tabel 3.1.

Tabel 3.1 Tabel Pengujian Waktu Penyedian Komputer Virtual

No	Jumlah Komputer Virtual yang Dibuat	Waktu yang Dibutuhkan Sistem Untuk Membuat
1	5	**Diketahui pada saat pengujian**
2	10	
3	15	
4	20	

Dalam pengujian penyediaan komputer virtual juga dilakukan pengukuran terhadap beban kerja CPU dan *memory* yang digunakan ketika proses penyediaan komputer virtual. Pengukuran waktu dari setiap skenario pengujian dilakukan menggunakan *stopwatch* dimulai dari eksekusi perintah sampai dengan diketahui semua komputer virtual yang dipesan telah berhasil dibuat dan dilakukan sebanyak 10 kali untuk tiap skenario, dari 10 data waktu yang telah didapatkan kemudian diambil rata-rata waktu dengan cara total waktu dari 10 kali pengujian per skenario dibagi 10. Waktu rata-rata tersebut yang disajikan pada tabel pengujian waktu penyediaan komputer virtual. Selanjutnya untuk mendapatkan data dari beban kerja CPU dan *memory* dengan mengambil data yang disajikan oleh perangkat lunak Munin.

3.5.2 Pengujian Waktu yang Dibutuhkan untuk Menyalakan Kembali Komputer Virtual

Pengujian ini bertujuan untuk menghitung lama waktu yang dibutuhkan sistem untuk menjalankan kembali komputer virtual yang berada dalam kondisi mati. Terdapat 4 skenario yang akan dilakukan untuk pengujian menyalakan kembali komputer virtual, detil pengujian dapat dilihat pada tabel 3.2.

Tabel 3.2 Tabel pengujian Menyalakan Kembali Komputer Virtual

No	Jumlah Komputer Virtual yang Dibuat	Waktu yang Dibutuhkan Sistem Untuk Menyalakan Kembali Komputer Virtual
1	5	**Diketahui pada saat pengujian**
2	10	
3	15	
4	20	

Dalam pengujian menyalakan komputer virtual waktu dari setiap skenario pengujian dilakukan menggunakan *stopwatch* dimulai dari eksekusi perintah sampai dengan diketahui semua komputer virtual yang dipesan telah berhasil dinyalakan kembali (dari kondisi *shutoff* menjadi *running)* dan dilakukan sebanyak 10 kali untuk tiap skenario, dari 10 data waktu yang telah didapatkan kemudian diambil rata-rata waktu dengan cara total waktu dari 10 kali pengujian tersebut dibagi 10. Waktu rata-rata yang didapatkan disajikan pada tabel pengujian waktu menyalakan komputer virtual.

3.5.3 Pengujian Waktu yang Dibutuhkan untuk Menyimpan Komputer Virtual

Skenario pada pengujian ini adalah dengan menghitung lama waktu yang dibutuhkan sistem untuk menjadikan komputer virtual dari kondisi *running* ke kondisi mati(*shutoff*). Terdapat 4 skenario yang akan dilakukan untuk pengujian menyimpan komputer virtual, detil pengujian dapat dilihat pada tabel 3.3.

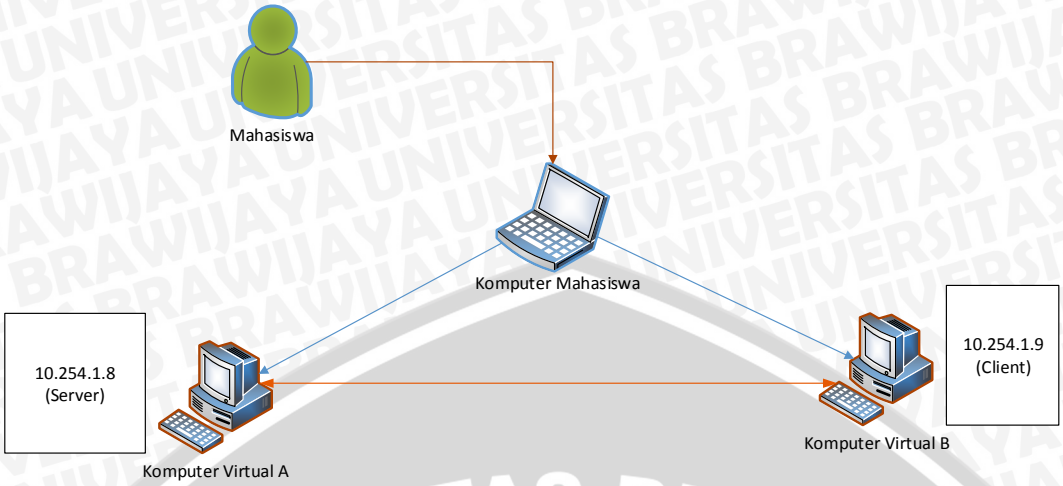
Table 3.3 Tabel pengujian Menjalankan Kembali Komputer Virtual

No	Jumlah Komputer Virtual yang Dibuat	Waktu yang Dibutuhkan Sistem Untuk Menyimpan Komputer Virtual
1	5	**Diketahui pada saat pengujian**
2	10	
3	15	
4	20	

Dalam pengujian menyimpan kondisi komputer virtual atau dalam penelitian ini menjadikan komputer virtual dari kondisi *running* menjadi *shutoff*, waktu dari setiap skenario pengujian dilakukan menggunakan *stopwatch* dimulai dari eksekusi perintah sampai dengan diketahui semua komputer virtual yang dipesan telah berhasil di-*shutdown* (dari kondisi *running* menjadi *shutoff)* dan dilakukan sebanyak 10 kali untuk tiap skenario, dari 10 data waktu yang telah didapatkan kemudian diambil rata-rata waktu dengan cara total waktu dari 10 kali pengujian tersebut dibagi 10. Waktu rata-rata yang didapatkan disajikan pada tabel pengujian waktu menyimpan kondisi komputer virtual.

3.5.4 Pengujian Sistem untuk Melakukan Praktikum Jaringan Komputer

Skenario pada pengujian ini adalah dengan melakukan praktikum jaringan komputer berupa materi pemrograman *socket*. Pengujian dilakukan dengan tujuan untuk mengetahui apakah fitur sudah berjalan dengan baik atau belum. Gambar 3.9 merupakan topologi skenario pengujian fitur sistem untuk melakukan praktikum pemrograman *socket*.



Gambar 3.9 Topologi Pengujian Materi Praktikum Premrograman *Socket*

3.6 Analisa Hasil

Dari data yang sudah didapat dari pengujian sistem, dilakukan analisa untuk mengetahui hasil yang akan digunakan untuk menarik kesimpulan dari penelitian yang dilakukan. Analisa hasil digunakan untuk mengetahui kelayakan dari sistem yang sudah dibuat.

3.7 Penarikan Kesimpulan

Tahapan ini merupakan tahapan terakhir dari penelitian dimana dari hasil analisa data yang sudah dilakukan ditarik kesimpulan tentang sistem yang sudah dirancang.

BAB 4 HASIL

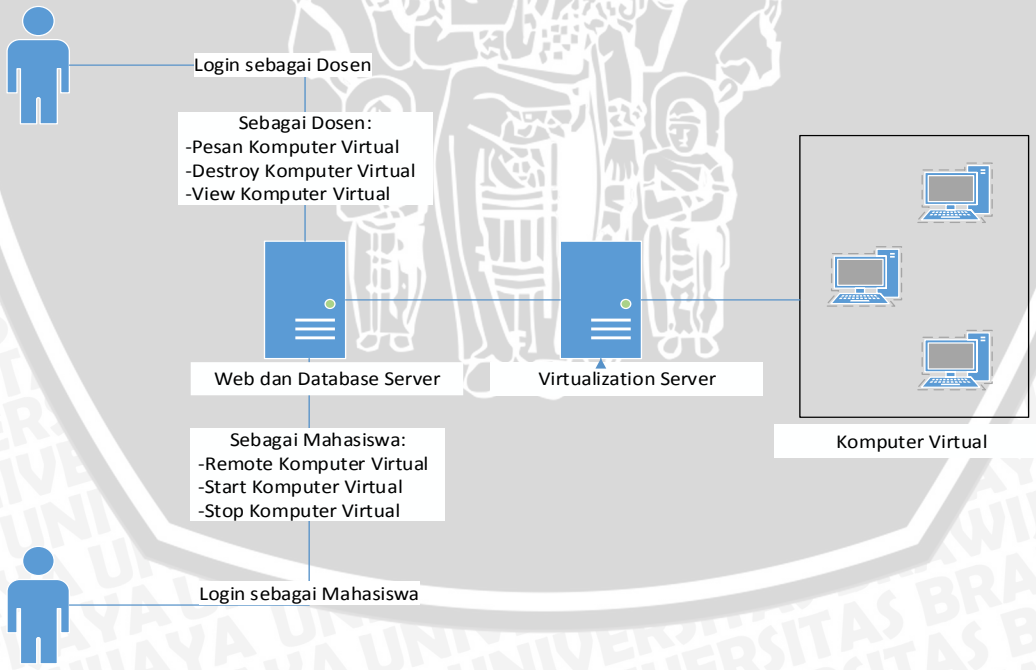
Pada bab ini akan dijelaskan langkah-langkah untuk menerapkan sistem laboratorium virtual jaringan komputer serta melakukan pengujian terhadap sistem tersebut. Pada tahap implementasi, langkah-langkah yang dilakukan adalah instalasi dan konfigurasi serta menerapkan sistem untuk mengatur sumber daya dari server. Dalam hal ini, langkah-langkah tersebut mengacu pada perangkat keras dan perangkat lunak yang digunakan.

4.1 Implementasi

Implementasi dilakukan sesuai dengan desain yang telah dibuat. Beberapa hal yang dilakukan untuk menerapkan sistem tersebut diantaranya adalah mempersiapkan perangkat keras dan instalasi perangkat lunak, konfigurasi antarmuka jaringan, dan menerapkan sistem pemesanan kelas, pengaturan jadwal, serta *remote* komputer virtual.

4.1.1 Rancangan Implementasi

Host yang digunakan dalam implementasi ini sebanyak 1 buah host yang bertindak sebagai *virtualization server*, web *server* dan *database server*. *Virtualization server* akan terhubung dengan komputer virtual yang berada didalamnnya. Rancangan dari implementasi sistem Laboratorium Virtual Jaringan Komputer pada penelitian ini tampak seperti pada Gambar 4.1.



Gambar 4.1 Rancangan Implementasi Sistem Laboratorim Virtual Jaringan Komputer

4.1.2 Spesifikasi Sistem

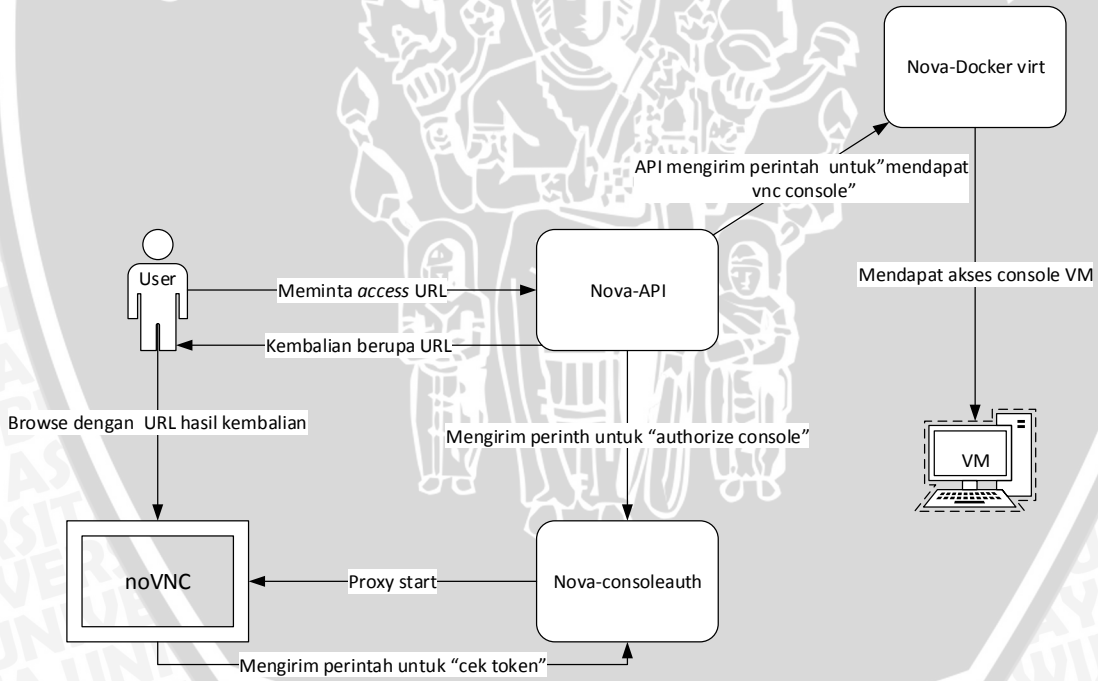
Untuk melaksanakan implementasi dari penelitian ini, dibutuhkan beberapa komponen yang berfungsi untuk menunjang keberhasilan implementasi. Beberapa komponen yang dibutuhkan adalah :

Server merupakan 1 unit PC yang didalamnya digunakan sebagai *virtualization server*, *web server* dan *database server*. Spesifikasinya adalah sebagai berikut

Prosesor	: Intel QuadCore
Harddisk	: 50 GB
Memory	: 4 GB
Sistem Operasi	: Ubuntu Server 14.04.3
Interface	: eth0
Alamat IP	: 192.168.68.134 / 27

4.1.3 Implementasi Sistem

Implementasi dari sistem *back end* Laboratorium virtual membutuhkan beberapa komponen yang digunakan untuk kepentingan-kepentingan fitur yang telah dijelaskan. Gambar 4.2 merupakan komponen yang dibutuhkan dalam implementasi sistem.



Gambar 4.2 Komponen Implementasi dari Sistem *Black End*

Komponen Komponen yang dibutuhkan diantaranya :

4.1.3.1 NoVNC

NoVNC adalah perangkat lunak berbasis *web* yang bertindak sebagai penghubung antara komputer pengguna dengan *virtual machine* pada *server*.

Instalasi VNC dilakukan pada server yang pertama *download* noVNC dengan perintah:

```
1  git clone git://github.com/kanaka/noVNC
```

Selanjutnya instal TightVNC dengan perintah:

```
1  sudo apt-get install tightvncserver
```

Tightvnc server di sini menyediakan *startup script* yang dapat digunakan untuk menjalankan *X desktop* yang disediakan oleh VNC secara terpisah. X *desktop* merupakan salinan tampilan *desktop* dari komputer yang sedang di-*remote*.

Langkah selanjutnya yaitu menjalankan VNC server. Untuk menjalankan VNC server digunakan perintah:

```
1  sudo vncserver :1
```

VNC server yang telah dijalankan akan berjalan di sisi *background.* Penggunaan angka "1" pada perintah di atas adalah untuk mengarahkan *port* yang digunakan *VNC server*, *port* dasar yang digunakan VNC server adalah "5900". Jadi pada perintah diatas akan menggunakan *port* "5901", atau jika ingin menggunakan *port* "5902" ganti angka "1" pada perintah di atas dengan angka "2", dan seterusnya.

Setelah VNC server berjalan install Websockify dengan perintah:

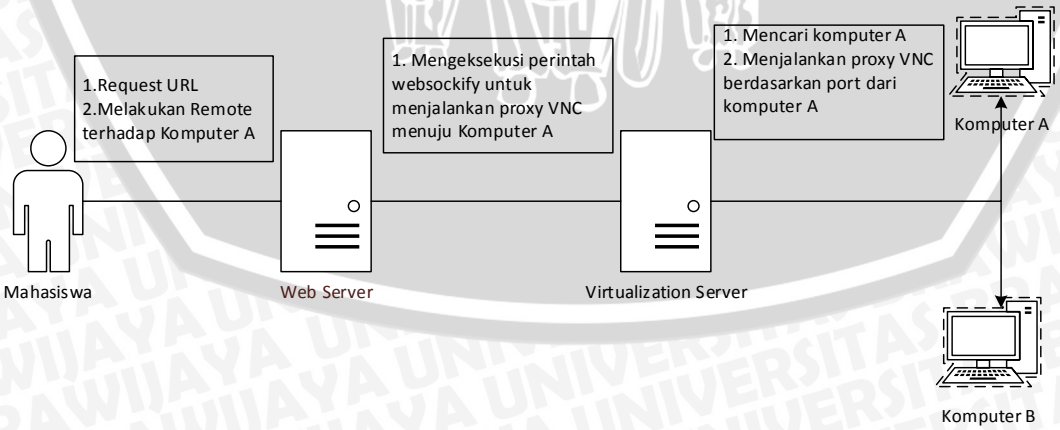
```
1  sudo apt-get install websockify
```

Websockify adalah sebuah web *socket* untuk TCP *proxy/bridge*. Dengan adanya *websockify*, memungkinkan untuk sebuah browser dapat terhubung dengan aplikasi, server, atau *service*.

Kemudian jalankan websockify dari dalam folder noVNC, dengan perintah:

```
1  /home/YOUR_USER_NAME/noVNC/utils/websockify --web ./ 8787
2  localhost:5901
```

NoVNC siap digunakan sebagai *proxy* ke *virtual machine* yang berada di *virtualization server*. Gambar 4.2 merupakan alur kerja dari NoVNC.



Gambar 4.3 Alur Kerja NoVNC

Dari gambar 4.3 dapat dilihat bahwa ketika mahasiswa ingin mengakses komputer virtual A yang berada v*irtualization server*, maka *web server* akan

menjalankan perintah untuk menjalankan *proxy* VNC yang berada di *web server*, *proxy* ini akan mengubungkan alamat ip *virtualization server* yang di dalamnya terdapat komputer virtual yaitu komputer A. *Virtualization server* akan menerima perintah yang dikirimkan oleh web *server* dan akan mengizinkan komputer komputer virtual A untuk dapat diakses melalui halaman web berdasarkan *port* yang terhubung dengan *proxy* VNC. Hasil dari proses tersebut mahasiswa akan mendapatkan *URL* untuk dapat mengakses tampilan *desktop* dari komputer virtual A.

4.1.3.2 Docker

Pada penelitian ini Docker diimplementasikan di dalam *virtualization server* yang berfungsi sebagai penyedia komputer virtual dengan jenis Docker *container*. Docker yang digunakan pada penelitan ini adalah Docker versi 1.9.1. Instalasi Docker dilakukan menggunakan perintah:

```
1  apt-get -y install lxc docker
```

Agar Docker bisa berjalan tanpa memerlukan mode *user root* server, maka perlu dilakukan pengubahan hak milik Docker yang awalnya hanya berada pada *user root* untuk ditambahkan untuk *user* biasa dengan cara melakukan konfigurasi pada file *"/etc/default/docker".* Perlu ditambahkan *user* yang akan digunakan pada bagian Docker *option*.

Tabel 4.1 File Konfigurasi *"/etc/default/docker"*

```
1  # Docker Upstart and SysVinit configuration file
2
3  # Customize location of Docker binary (especially for
4  development testing).
5  #DOCKER="/usr/local/bin/docker"
6
7  # Use DOCKER_OPTS to modify the daemon startup options.
8  #DOCKER_OPTS="--dns 8.8.8.8 --dns 8.8.4.4"
9  DOCKER_OPTS='-G dev'
10
11 # If you need Docker to use an HTTP proxy, it can also be
12 specified here.
13 #export http_proxy="http://127.0.0.1:3128/"
14
15 # This is also a handy place to tweak where Docker's
16 temporary files go.
17 #export TMPDIR="/mnt/bigdrive/docker-tmp"
```

Tambahkan *user* yang akan digunakan ke dalam Docker *option*, yaitu pada baris ke 9. Pada sistem ini menggunakan *user* "dev", penggunaan *user* dev pada Docker option dapat ditambahkan dengan skrip:

```
1  DOCKER_OPTS='-G dev'
```

Penggunaan Docker pada penelitian ini dihubungkan dengan Openstack sebagai perangkat lunak yang berfungsi untuk mengelola lingkungan virtualisasi. Dibutuhkan *plugin* Nova-docker agar Openstack bisa bekerja untuk mengelola

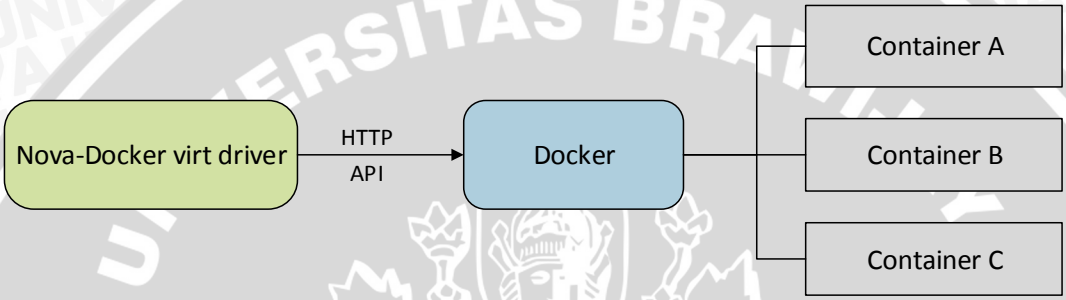
Docker. Instalasi Nova-docker dilakukan dengan terlebih dahulu men-*download* Nova-docker dengan menjalankan perintah:

```
1 git clone -b stable/kilo
2   http://github.com/stackforge/nova-docker.git
```

Pada perintah tersebut ditambahkan perintah *stable/kilo*, karena menyesuaikan dengan penggunaan versi dari Openstack yaitu Openstack Kilo. Untuk meng-*install* Nova-docker jalankan perintah "*pip install .*" dari dalam folder penyimpanan Nova-docker.

```
1 sudo pip install .
```

Gambar 4.4 merupakan alur kerja integrasi Docker dan Openstack dengan Nova-docker *driver* sebagai penghubung.



Gambar 4.4 Alur Kerja Docker dan Openstack

Dari gambar 4.3 dapat dilihat Nova-docker *driver* berkomunikasi dengan Docker menggunakan HTTP API. *Container-container* tersebut yang kemudian digunakan oleh sistem laboratorium virtual sebagai komputer virtual.

4.1.3.3 Openstack

Openstack digunakan sebagai perangkat lunak yang mengelola lingkungan virtualisasi, komputasi, *storage*, dan antarmuka jaringan. Pada penelitian ini menggunakan Openstack kilo. Proses instalasi Openstack pada penelitian ini menggunakan DevStack. DevStack merupakan serangkaian *script* digunakan untuk dengan cepat membangun lingkungan Openstack secara lengkap. Sebelum menjalankan DevStack untuk proses instalasi Openstack, perlu dilakukan konfigurasi untuk menyesuaikan dengan kebutuhan sistem. Konfigurasi tersebut disimpan di dalam file *local.conf.* Tabel 4.2 merupakan isi konfigurasi dari file *local.conf.*

Tabel 4.2 Konfigurasi DevStack

```
1 [[local|localrc]]
2 HOST_IP=192.168.68.134
3 ADMIN_PASSWORD=123456
4 MYSQL_PASSWORD=123456
5 RABBIT_PASSWORD=123456
6 SERVICE_PASSWORD=123456
7 FLOATING_RANGE=192.168.12.0/24
8 FLAT_INTERFACE=eth0
9 Q_FLOATING_ALLOCATION_POOL=start=192.168.12.150,end=192.168.12.254
```

```
10 PUBLIC_NETWORK_GATEWAY=192.168.12.15
11 SERVICE_TOKEN=super-secret-admin-token
12 VIRT_DRIVER=novadocker.virt.docker.DockerDriver
13 RECLONE=yes
14
16 DEST=/opt/stack
17 SERVICE_DIR=$DEST/status
18 DATA_DIR=$DEST/data
19 LOGFILE=$DEST/logs/stack.sh.log
20 LOGDIR=$DEST/logs
21
22 # The default fixed range (10.0.0.0/24) conflicted with an address
23 # range I was using locally.
24 FIXED_RANGE=10.254.1.0/24
25 NETWORK_GATEWAY=10.254.1.1
26
27 # Services
28 disable_service n-net
29 enable_service n-cauth
30 enable_service q-svc
31 enable_service q-agt
32 enable_service q-dhcp
33 enable_service q-l3
34 enable_service q-meta
35 disable_service horizon
36 disable_service tempest
37
38 # Introduce glance to docker images
39 [[post-config|$GLANCE_API_CONF]]
40 [DEFAULT]
41 container_formats=ami,ari,aki,bare,ovf,ova,docker
42
43 # Configure nova to use the nova-docker driver
44 [[post-config|$NOVA_CONF]]
45 [DEFAULT]
46 compute_driver=novadocker.virt.docker.DockerDriver
```

Penjelasan konfigurasi pada tabel 4.2 sebagai berikut:

1. Baris 2 merupakan alamat IP server.
2. Baris 3 merupakan password dari Admin.
3. Baris 4 merupakan password dari MySQL.
4. Baris 5 merupakan password dari Rabbit.
5. Baris 8 merupakan antarmuka jaringan yang digunakan di Openstack
6. Baris 9 merupakan *pool* dari *floating* IP yang akan digunakan komputer virtual yang bisa digunakan untuk akses ke pihak luar.
7. Baris 10 merupakan alamat gateway yang digunakan di Openstack untuk berkomunikasi dengan pihak luar.
8. Baris 12 merupakan driver virtualisasi yang digunakan Openstack untuk mengakses Docker *container.*
9. Baris 16-20 merupakan direktori penyimpanan *log* dari aktivitas Openstack.
10. Baris 24 merupakan alamat local yang akan digunakan oleh komputer virtual.

11. Baris 28-36 merupakan daftar *service* yang digunakan dan yang tidak digunakan dalam Openstack. *Enable* menunjukkan bahwa *service* tersebut akan dijalankan di Openstack, sedangkan *disable* menunjukkan bahwa *service* tersebut tidak dijalankan di Openstack.
12. Baris 39-41 merupakan konfigurasi untuk *service* Glance, agar Glance dapat membaca format *image* dari Docker.
13. Baris 41 merupakan konfigurasi driver untuk proses komputasi yang didalamnya proses komputasi akan ditangani oleh *driver* nova-docker.

Setelah konfigurasi selesai Openstack siap di-install dengan mengeksekusi file *"stack.sh",* "*stack.sh"* merupakan sebuah *script* yang telah disediakan oleh Devstack. Setelah proses instalasi selesai perlu untuk di-instal "rootwrap" yang merupakan file konfigurasi dari nova-docker.

Proses selanjutnya adalah mengunggah Docker *image* ke dalam *service* Glance, untuk bisa melakukan proses tersebut masuk ke dalam Openstack dengan menggunakan *user* "admin". Pada penelitian ini digunakan Docker *image* Ubuntu 14.04, untuk men-download *image* tersebut jalankan perintah:

```
1  docker pull rastasheep/ubuntu-sshd:14.04
```

Setelah image berhasil di-download jalankan perintah "*docker save rastasheep/ubuntu-sshd:14.04 | glance image-create --is-public=True --container-format=docker --disk-format=raw --name rastasheep/ubuntu-sshd:14.04*"

```
1  docker save rastasheep/ubuntu-sshd:14.04 | glance
2      image-create      --is-public=True      --container-
3      format=docker --disk-format=raw
4       --name rastasheep/ubuntu-sshd:14.04
```

Perintah ini digunakan untuk mengunggah Docker image ke dalam Glance. Dalam perintah tersebut digunakan format *container* yaitu "*docker*" dan format disk "*raw*".

Setelah proses mengunggah Docker *image* ke Glance berhasil, selanjutnya komputer virtual siap untuk dijalankan dengan menggunakan *image* yang telah tersimpan di dalam Glance.

4.1.4 Implementasi Sistem Penyediaan Komputer Virtual

Berdasarkan rancangan sistem yang telah dijelaskan sebelumnya adalah bagian sistem untuk dosen, dimana bagian ini berfungsi untuk membuat komputer virtual yang akan digunakan oleh mahasiswa. Dalam sistem *back end* dosen bisa membuat komputer virtual dengan menjalankan perintah:

```
1  nova boot –image "nama_image" –flavor "nama_flavor"
2     nama_komputer_virtual
```

Pembuatan komputer virtual ini *user* (dosen) dapat memilih spesifikasi dari komputer virtual yang akan dibuat dan image yang akan digunakan atau yang disebut dengan *flavor*. Untuk mendapatkan daftar dari spesifikasi yang disediakan oleh Openstack jalankan perintah:

```
1  nova flavor-list
```

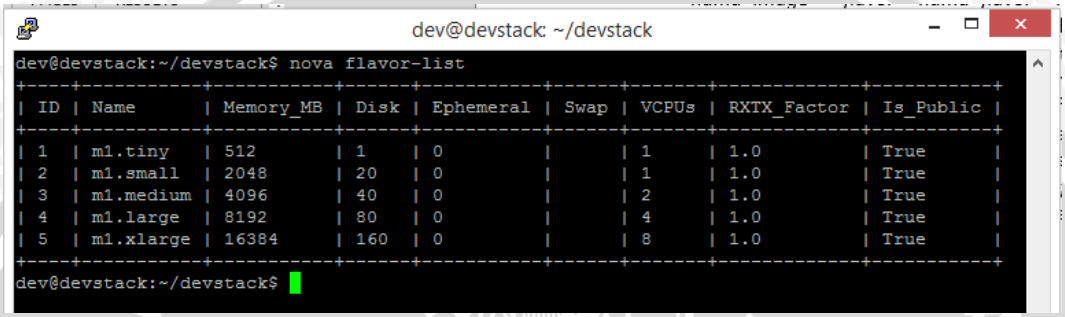
Sedangkan untuk mendapatkan daftar *image* yang dapat digunakan jalankan perintah:

```
1  glance image-list
```

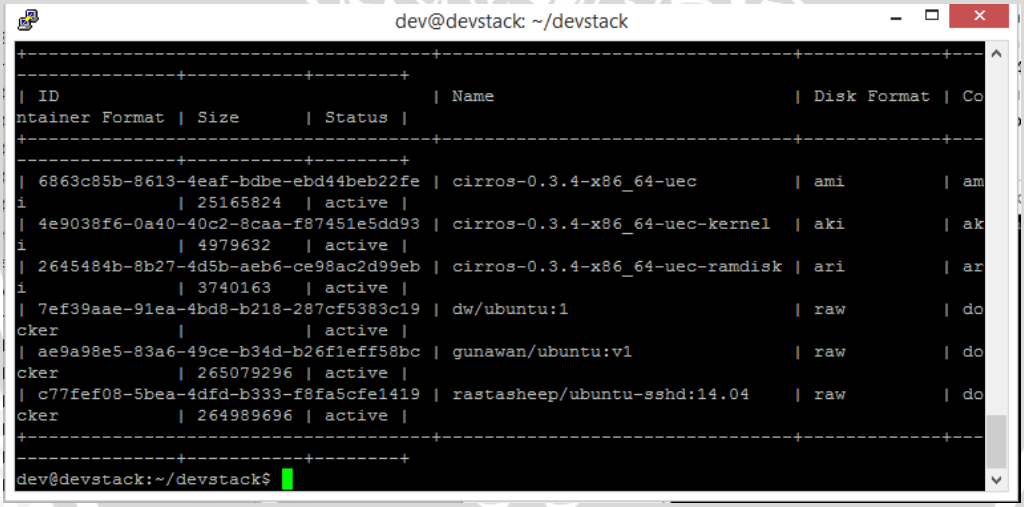
Untuk mendapatkan daftar dari komputer virtual yang telah dibuat jalankan perintah:

```
1  nova list
```

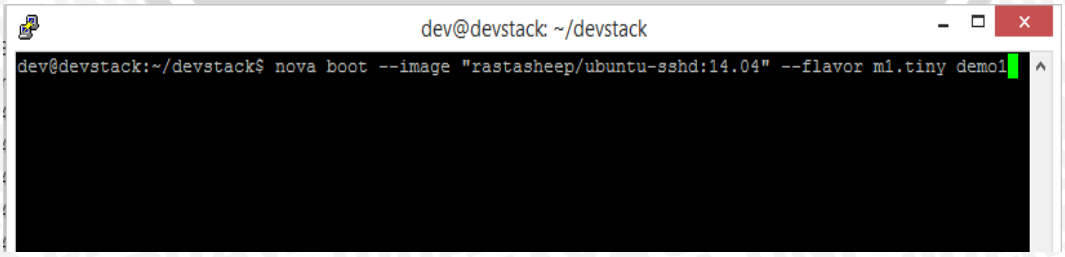
Gambar 4.5 merupakan daftar *flavor* yang bisa digunakan, gambar 4.6 merupakan daftar dari image yang tersedia, gambar 4.7 merupakan proses untuk membuat sebuah komputer virtual dan gambar 4.8 merupakan daftar dari komputer virtual yang telah dibuat.



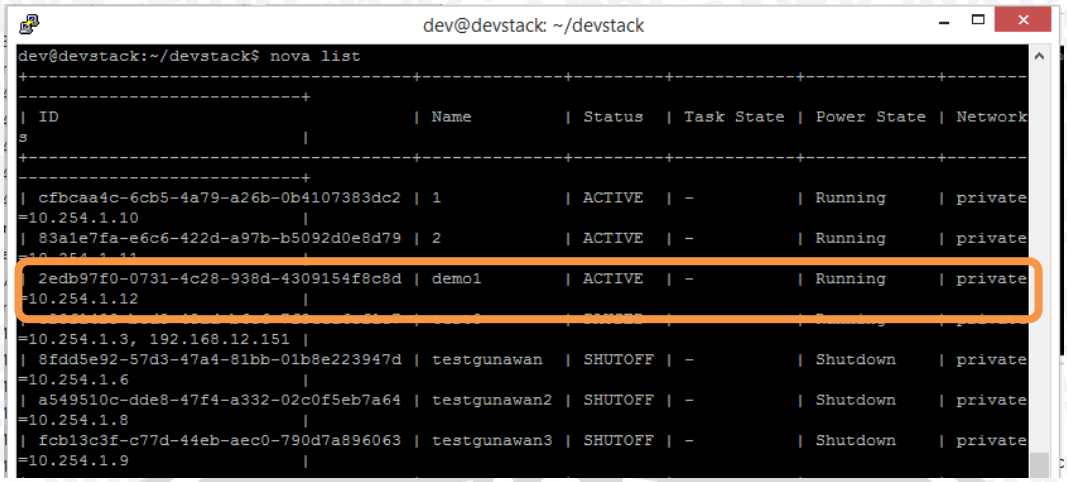
Gambar 4.5 Daftar *Flavor*



Gambar 4.6 Daftar *Image* yang Tersedia



Gambar 4.7 Eksekusi Perintah Membuat Komputer Virtual



Gambar 4.8 Daftar Komputer Virtual yang Telah Dibuat

Dari gambar 4.8 dapat dilihat perintah untuk membuat komputer virtual dengan nama "demo1" telah berhasil dijalankan. Komputer virtual demo1 dalam kondisi *running* dan mendapat IP 10.254.1.12.

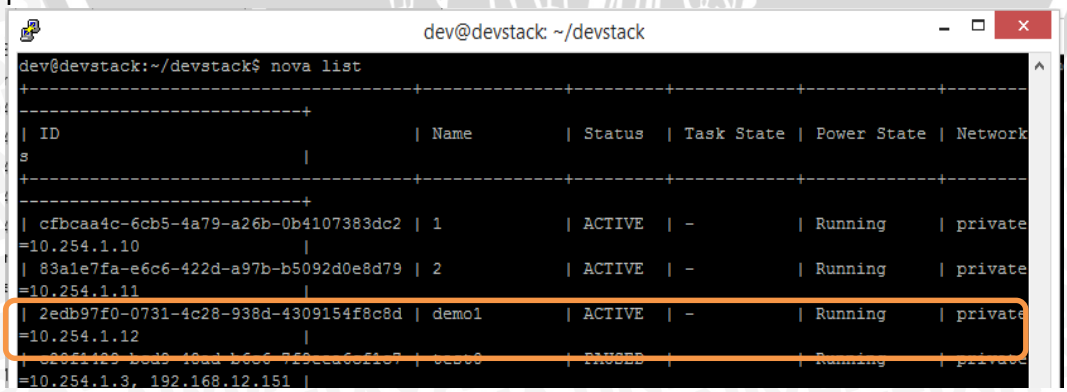
4.1.5 Implementasi Sistem Stop Komputer Virtual

Dalam satu sesi praktikum, mahasiswa diberikan waktu sesuai dengan lamanya waktu dalam jadwal untuk mengakses komputer virtual-nya. Waktu yang diberikan ini tentunya tidak akan cukup untuk melaksanakan seluruh proses kegiatan praktikum mulai dari pemahaman materi praktikum hingga pengerjaan tugas praktikum. Oleh karena itu pada penelitian ini mahasiswa dapat men-*shutdown* komputer virtual yang sedang digunakan.

Untuk men-shutdown komputer virtual yang sedang digunakan dapat menjalankan perintah:

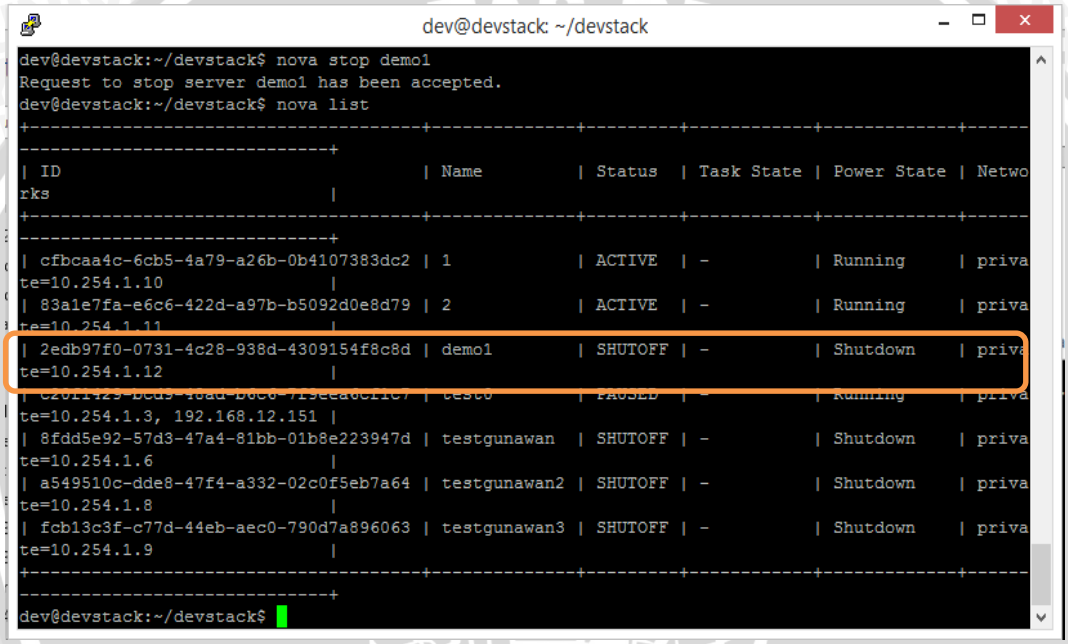
```
1  nova stop nama_komputer_virtual /ID_komputer_virtual
```

Gambar 4.9 merupakan kondisi awal dari komputer virtual demo1 yang sebelumnya telah dibuat, gambar 4.10 merupakan proses untuk mematikan komputer virtual dan gambar 4.11 merupakan kondisi setelah dilakukan eksekusi perintah.



Gambar 4.9 Kondisi Awal Komputer Virtual

Gambar 4.10 Menghentikan Komputer Virtual

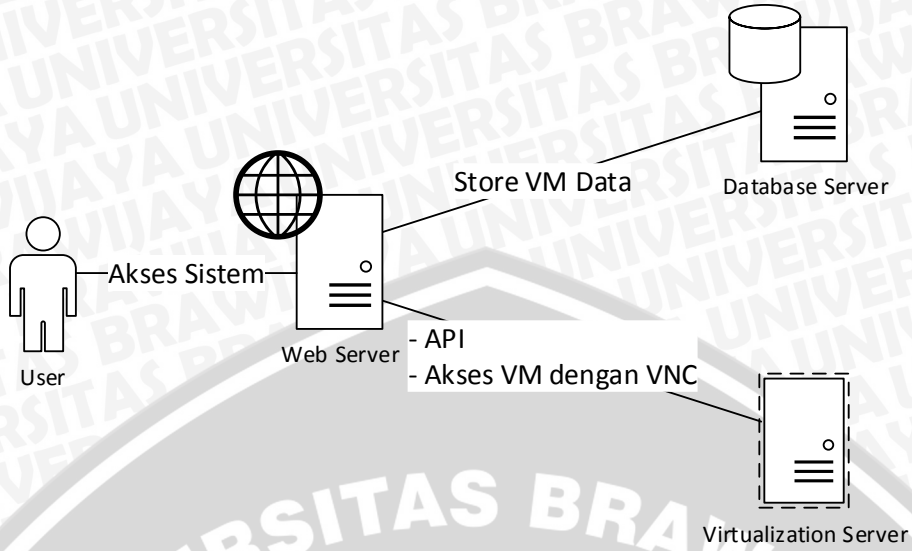


Gambar 4.11 Kondisi Akhir Komputer Virtual

Dari gambar 4.11 dapat dilihat proses untuk menghentikan komputer virtual telah berhasil. Dari yang awalnya komputer virtual berada pada kondisi *running* menjadi *shutdown*.

4.1.6 Hubungan Kerja Sama *Back End* dan *Front End*

Hubungan kerja sama *back end* dan *front end* sistem, bertujuan untuk menciptakan lingkungan laboratorium virtual jaringan komputer yang dapat mengakomodasi pengguna laboratorium dalam melaksanakan proses praktikum. Gambar 4.12 merupakan sinkronisasi dari *back end* dan *front end* sistem laboratorium virtual jaringan komputer.



Gambar 4.12 Sinkronisasi *Back End* dan *Front End* Sistem

Bagian *front end* dari sistem ini terdiri dari *web* server yang bertindak sebagai penyedia layanan antarmuka untuk pengguna berbasis web dan *database* server bertindak sebagai tempat penyimpanan kebutuhan sistem. Sedangkan pada bagian sistem *back end* terdapat *virtualization server* yang bertindak sebagai penyedia kebutuhan mahasiswa berupa komputer virtual.

Dalam proses pembuatan komputer virtual, *virtualization server* bekerja sama dengan *web server*, sebelum seorang pengguna bisa melakukan *request* untuk membuat komputer virtual melalui antarmuka, dilakukan terlebih dahulu cek autentikasi *tenant* yang dilakukan dengan menggunakan API untuk mendapat *token* dan *tenant.* Setelah berhasil pengguna bisa melakukan *request* melalui antarmuka sistem, yang kemudian data atau dalam hal ini *request* akan dikirim kepada *virtualization server* melalui API untuk membuat komputer virtual sesuai dengan *request* dari pengguna. Kemudian data hasil dari pembuatan komputer virtual tersebut, oleh *virtualization server* akan dikirim ke *database server* untuk disimpan. API lain yang digunakan dalam sistem ini yaitu, API untuk mengambil alamat IP dari komputer virtual, API untuk mendapatkan daftar *image*, API untuk mendapatkan daftar *flavor,* API untuk mendapatkan daftar dari komputer yang telah dibuat, dan API untuk memberikan *action* kepada komputer virtual seperti *os-stop* untuk mematikan komputer virtual dan *os-start* untuk menghidupkan komputer virtual.

Untuk dapat mengakses komputer virtual yang dibuat sebelumnya *virtualization server* menyediakan VNC. Ketika mahasiswa ingin mengakses komputer virtual tersebut, maka *web server* akan menjalankan perintah untuk menjalankan *proxy* VNC yang berada di *web server*. *Virtualization server* akan menerima perintah yang dikirimkan oleh web *server* dan akan mengizinkan komputer virtual untuk dapat diakses melalui halaman web berdasarkan *port* yang terhubung dengan *proxy* VNC.

4.2 Pengujian

Sebelum dilakukan pengujian terhadap performa kapasitas dan availibitas sistem, terlebih dahulu dilakukan pengujian fungsional sistem apakah sistem telah berjalan lancar apa belum. Pengujian yang akan dilakukan dalam penelitian ini adalah pengujian untuk menguji performa kapasitas dan availibilitas sistem. Performa yang diuji adalah seberapa lama waktu yang dibutuhkan sistem untuk menyediakan komputer virtual, berapa lama waktu yang dibutuhkan sistem untuk menyalakan kembali komputer virtual, serta berapa lama waktu yang dibutuhkan sistem untuk mematikan komputer virtual dan beban kerja dari pengujian penyediaan komputer virtual dan pengujian komputer virtual digunakan untuk mengerjakan materi praktikum.

Dalam penelitian ini, pengujian dibedakan menjadi 4 tahap. Tahap pertama adalah pengujian sistem penyediaan komputer virtual untuk mengetahui berapa lama waktu dan beban kerja yang dibutuhkan oleh sistem *back end* untuk membuat komputer virtual. Tahap kedua adalah pengujian untuk mengetahui berapa lama waktu yang dibutuhkan oleh sistem untuk menyalakan kembali komputer virtual yang berada pada kondisi *shutoff.* Tahap ketiga adalah pengujian untuk mengetahui berapa lama waktu yang dibutuhkan oleh sistem untuk membuat komputer virtual dari kondisi *running* ke kondisi mati(*shutoff).* Tahap keempat pengujian sistem untuk mengetahui hasil dari sistem digunakan untuk melakukan praktikum pemrograman *socket*.

4.2.1 Pengujian Sistem Penyediaan Komputer Virtual

Pada pengujian sistem penyediaan komputer virtual, pengujian dilakukan dengan menjalankan fungsi *registerServer* untuk membuat komputer virtual. Jumlah pembuatan komputer virtual disesuaikan dengan skenario pengujian.

Terdapat 4 skenario pengujian dalam tahap ini, dimana skenario pertama akan dibuat 5 komputer virtual, skenario kedua akan dibuat 10 komputer virtual, skenario ketiga akan dibuat komputer virtual, dan skenario keempat akan dibuat 20 komputer virtual. Adapun untuk mengetahui berapa lama waktu yang dibutuhkan untuk membuat komputer virtual berdasarkan skenario diatas digunakan cara dengan melihat berapa lama yang dibutuhkan sistem untuk membuat komputer virtual. Setiap skenario pengujian penyediaan komputer virtual dilakukan sebanyak 10 kali untuk kemudian diambil rata-rata waktu dari keempat skenario pengujian untuk mengetahui total waktu dari setiap skenario. Kemudian rata-rata waktu yang dibutuhkan untuk membuat setiap komputer virtual diambil dari hasil total waktu pembuatan dibagi dengan banyaknya komputer virtual. Sedangkan untuk mengetahui beban kerja dari sistem menggunakan perangkat lunak Munin.

4.2.2 Pengujian Sistem Menyalakan Kembali Komputer Virtual

Pada pengujian sistem proses menyalakan kembali komputer virtual, pengujian dilakukan dengan menjalankan perintah yang terdapat di dalam fungsi

runSSH untuk menjalankan perintah menyalakan kembali komputer virtual. Jumlah komputer virtual disesuaikan dengan skenario pengujian.

Terdapat 4 skenario pengujian dalam tahap ini, dimana skenario pertama proses akan dilakukan terhadap 5 komputer virtual, skenario kedua proses akan dilakukan terhadap 10 komputer virtual, skenario ketiga proses akan dilakukan terhadap 15 komputer virtual, dan skenario keempat akan dilakukan terhadap 20 komputer virtual. Adapun untuk mengetahui berapa lama waktu yang dibutuhkan untuk menyalakan kembali komputer virtual tersebut digunakan cara dengan menghitung berapa lama waktu yang dibutuhkan sistem untuk menjalankan kembali komputer virtual yang berada dalam kondisi mati(*shutoff*) menjadi *running*. Setiap skenario pengujian menyalakan kembali komputer virtual dilakukan sebanyak 10 kali untuk kemudian diambil rata-rata waktu dari keempat skenario pengujian untuk mengetahui total waktu dari setiap skenario. Kemudian rata-rata waktu yang dibutuhkan untuk menyalakan setiap komputer virtual diambil dari hasil total waktu dibagi dengan banyaknya komputer virtual.

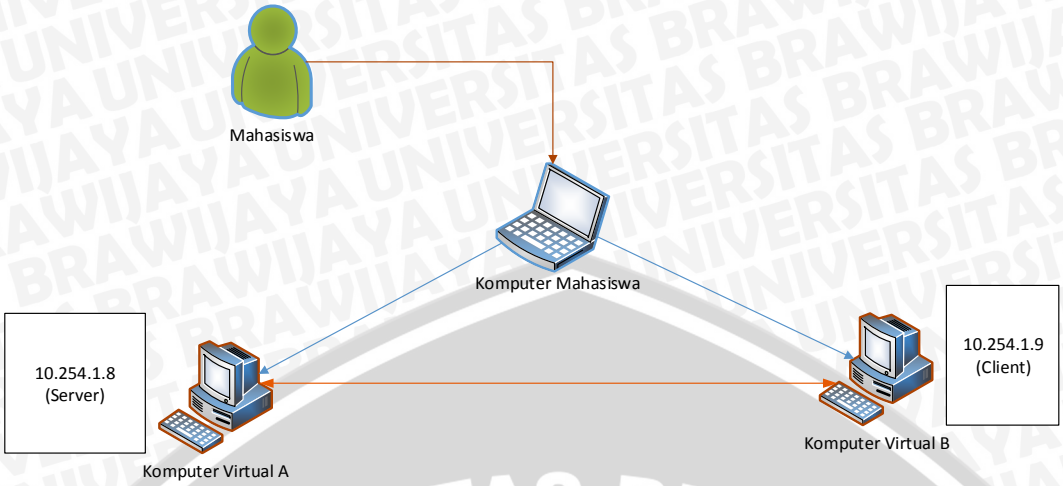
4.2.3 Pengujian Sistem Shutdown Komputer Virtual

Pada pengujian sistem shutdown komputer virtual, pengujian dilakukan dengan menjalankan fungsi *shutdown* komputer virtual untuk menjalankan perintah penyimpanan pada komputer virtual. Jumlah komputer virtual yang akan dimatikan disesuaikan dengan skenario pengujian.

Terdapat 4 skenario pengujian dalam tahap ini, dimana skenario pertama proses penyimpanan akan dilakukan terhadap 5 komputer virtual, skenario kedua akan dilakukan terhadap 10 komputer virtual, skenario ketiga proses akan dilakukan pada 15 komputer virtual, dan skenario keempat akan dilakukan terhadap 20 komputer virtual. Adapun untuk mengetahui berapa lama waktu yang dibutuhkan untuk proses *shutdown* komputer virtual tersebut digunakan cara dengan menghitung berapa lama waktu yang dibutuhkan sistem untuk menjadikan komputer virtual dari kondisi *running* ke kondisi mati(*shutoff*). Setiap skenario pengujian *shutdown* komputer virtual dilakukan sebanyak 10 kali untuk kemudian diambil rata-rata waktu dari keempat skenario pengujian. Kemudian rata-rata waktu yang dibutuhkan untuk melakukan proses *shutdwon* setiap komputer virtual diambil dari hasil total waktu dibagi dengan banyaknya komputer virtual.

4.2.4 Pengujian Sistem untuk Melakukan Praktikum Jaringan Komputer

Skenario pada pengujian ini adalah dengan melakukan praktikum jaringan komputer berupa materi pemrograman *socket*. Pengujian dilakukan dengan tujuan untuk mengetahui apakah fitur sudah berjalan dengan baik atau belum. Gambar 4.13 merupakan topologi skenario pengujian fitur sistem untuk melakukan praktikum pemrograman *socket*.



Gambar 4.13 Topologi Pengujian Materi Praktikum Premrograman *Socket*

Dari gambar di atas dapat dilihat mahasiswa akan melakukan praktikum pemrograman *socket* sesuai dengan topologi yang diberikan. Setiap mahasiswa akan diberi akses ke dua buah komputer virtual, pada komputer virtual A akan diimplementasikan program untuk membuat *socket server* dan menerima permintaan koneksi dari klien(komputer virtual B). Pengujian dilakukan untuk mengetahui apakah sistem berhasil digunakan untuk praktikum pemrograman *socket* atau tidak.

BAB 5 PENGUJIAN DAN ANALISIS

5.1 Hasil Pengujian

Pengujian yang dilakukan dalam penelitian ini meliputi beberapa macam pengujian, meliputi pengujian untuk mengukur waktu pembutan komputer virtual, kemudian pengujian program untuk mengukur waktu menyalakan kembali komputer virtual, dan pengujian untuk mengukur waktu untuk *shutdown* komputer virtual.

5.1.1 Hasil Pengujian Sistem Penyediaan Komputer Virtual

Pengujian dilakukan terhadap sistem *back end* penyedian komputer virtual dimana dilakukan pengujian sesuai skenario dengan hasil pengujian seperti yang disajikan pada Tabel 5.1.

Tabel 5.1 Tabel Hasil Pengujian Sistem Penyediaan Komputer Virtual

Skenario	Jumlah Komputer Virtual yang Dibuat	Total Waktu yang Dibutuhkan Sistem Untuk Membuat Komputer Virtual (dalam detik)	Rata-rata per komputer virtual (dalam detik)
1	5	27,971	5,594
2	10	39,053	3,905
3	15	60,210	4,014
4	20	68,128	3,406

Tabel 5.2 merupakan hasil beban kerja CPU dari pengujian sistem untuk membuat 5, 10, 15, dan 20 komputer virtual yang diambil dari hasil pengukuran dengan menggunakan perangkat lunak Munin.

Tabel 5.2 Beban Kerja CPU Pembuatan Komputer Virtual

Skenario	Jumlah Komputer Virtual Yang Dibuat	Beban kerja yang dibutuhkan Sistem untuk Membuat Komputer Virtual (%)
1	5	32
2	10	34
3	15	36
4	20	40

Tabel 5.4 merupakan hasil beban kerja *memory* dari pengujian sistem untuk membuat 5, 10, 15, dan 20 komputer virtual.

Tabel 5.3 Beban Kerja *memory* Pembuatan Komputer Virtual

Skenario	Jumlah Komputer Virtual Yang Dibuat	Beban kerja *Memory* yang dibutuhkan Sistem untuk Membuat Komputer Virtual(GB)
1	5	2,9
2	10	3
3	15	3,05
4	20	3,13

5.1.2 Hasil Pengujian Sistem Menyalakan Kembali Komputer Virtual

Pengujian dilakukan terhadap sistem *back end* menyalakan kembali komputer virtual dimana dilakukan pengujian sesuai skenario dengan hasil pengujian seperti yang disajikan pada Tabel 5.4.

Tabel 5.4 Tabel Hasil Pengujian Sistem Menyalakan Kembali Komputer Virtual

Skenario	Jumlah VM yang Dibuat	Total Waktu yang Dibutuhkan Sistem untuk Menyalakan Kembali VM (dalam detik)	Rata-rata per komputer virtual (dalam detik)
1	5	12,887	2,577
2	10	35,264	3,526
3	15	43,778	2,918
4	20	65,542	3,277

5.1.3 Hasil Pengujian Sistem *Shutdown* Komputer Virtual

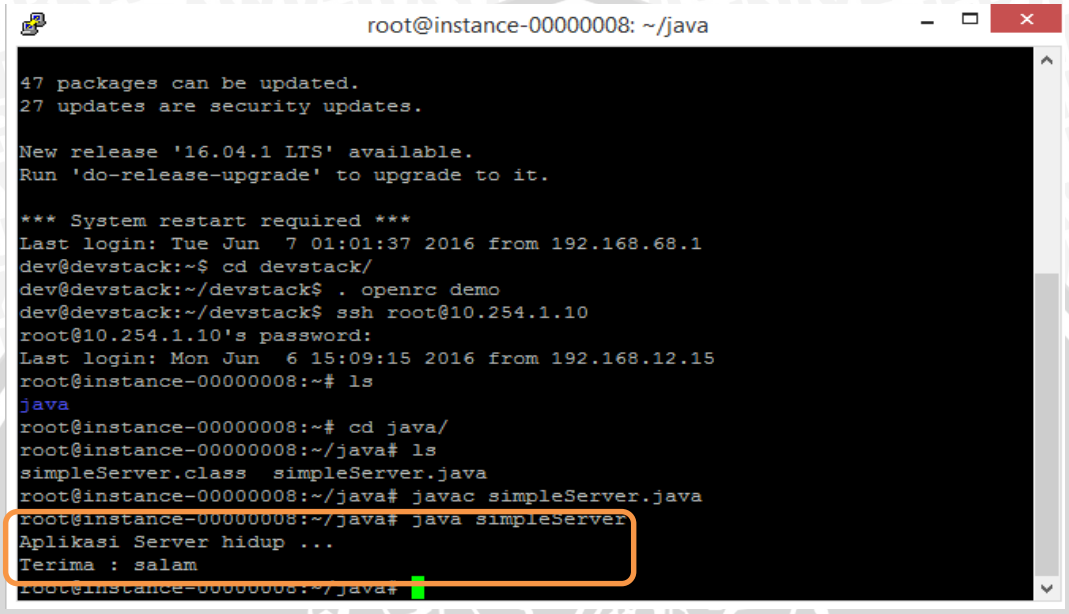
Pengujian dilakukan terhadap sistem *back end* melakukan proses *shutdown* komputer virtual dimana dilakukan pengujian sesuai skenario dengan hasil pengujian seperti yang disajikan pada Tabel 5.5.

Tabel 5.5 Tabel Hasil Pengujian Sistem *Shutdown* Komputer Virtual

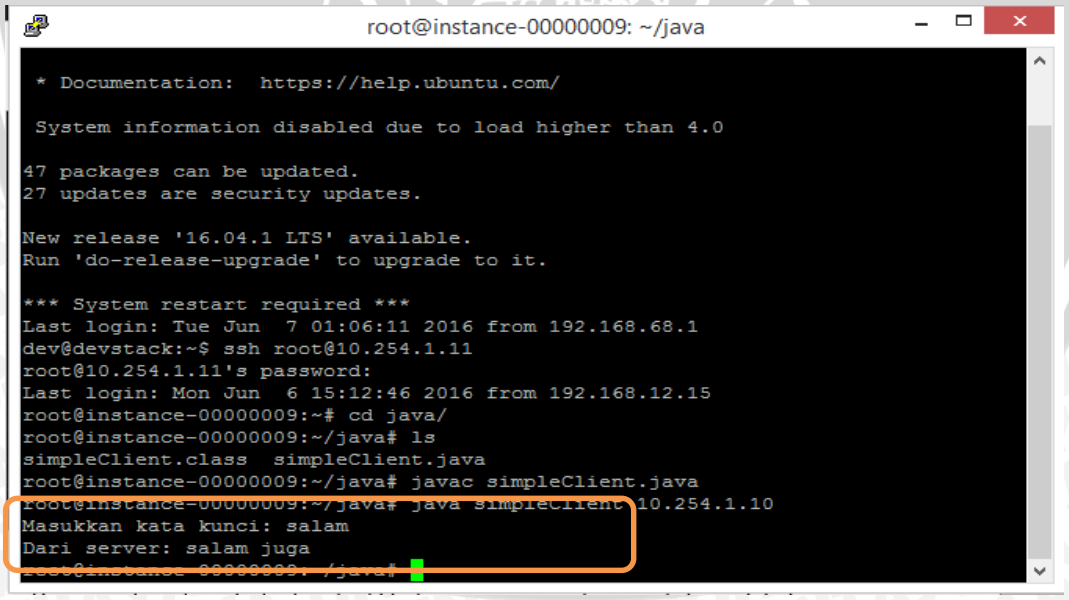
Skenario	Jumlah komputer virtual yang Dibuat	Total Waktu yang Dibutuhkan sistem untuk Shutdown Kondisi terakhir dari komputer virtual (dalam detik)	Rata-rata per komputer virtual (dalam detik))
1	5	7,722	1,544
2	10	14,323	1,432
3	15	19,552	1,303
4	20	21,078	1,053

5.1.4 Hasil Pengujian Sistem untuk Melakukan Praktikum Jaringan Komputer

Pengujian dilakukan dengan melakukan implementasi pemrograman socket sesuai dengan skenario yang telah dibuat sebelumnya. Topologi pengujian ini mengacu pada gambar 4.13. Dapat dilihat gambar 5.1 merupakan hasil *screenshot* dari *server socket* dan 5.2 merupakan *screenshot* dari *client socket.*



Gambar 5.1 Aplikasi Server



Gambar 5.2 Aplikasi Client

Dari gambar tersebut dapat dilihat pengujian sistem untuk melakukan praktikum pemrograman socket berhasil dilakukan. Gambar 5.2 menunjukkan *client* mengirimkan kata kunci, yaitu "salam" dan dari sisi server memberikan respon berupa kata "salam juga".

5.1.5 Hasil Pengujian Sistem untuk Komputer Virtual Melakukan Ping

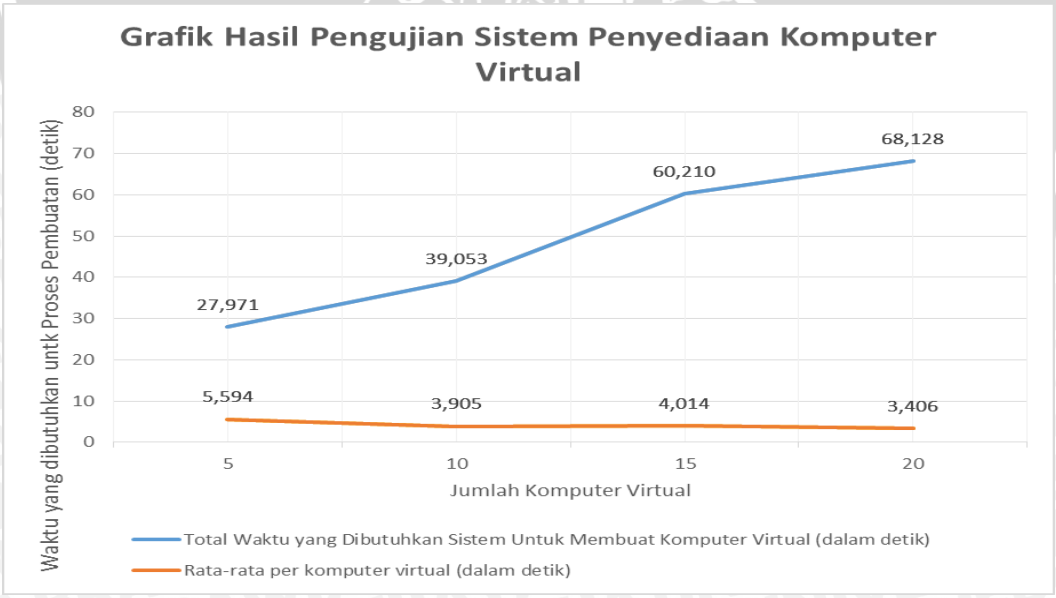
Pengujian dilakukan terhadap sistem *back end* untuk komputer virtual melakukan ping dilakukan pengujian sesuai skenario dengan hasil pengujian seperti yang disajikan pada tabel 5.6

Tabel 5.6 Hasil Pengujian Sistem untuk Komputer Virtual Melakukan Ping

Skenario	Jumlah Komputer Virtual Yang Dijalankan	Beban kerja yang dibutuhkan Sistem untuk Komputer Virtual Melakukan Ping (%)
1	5	15
2	10	19
3	15	23
4	20	24

5.2 Analisis

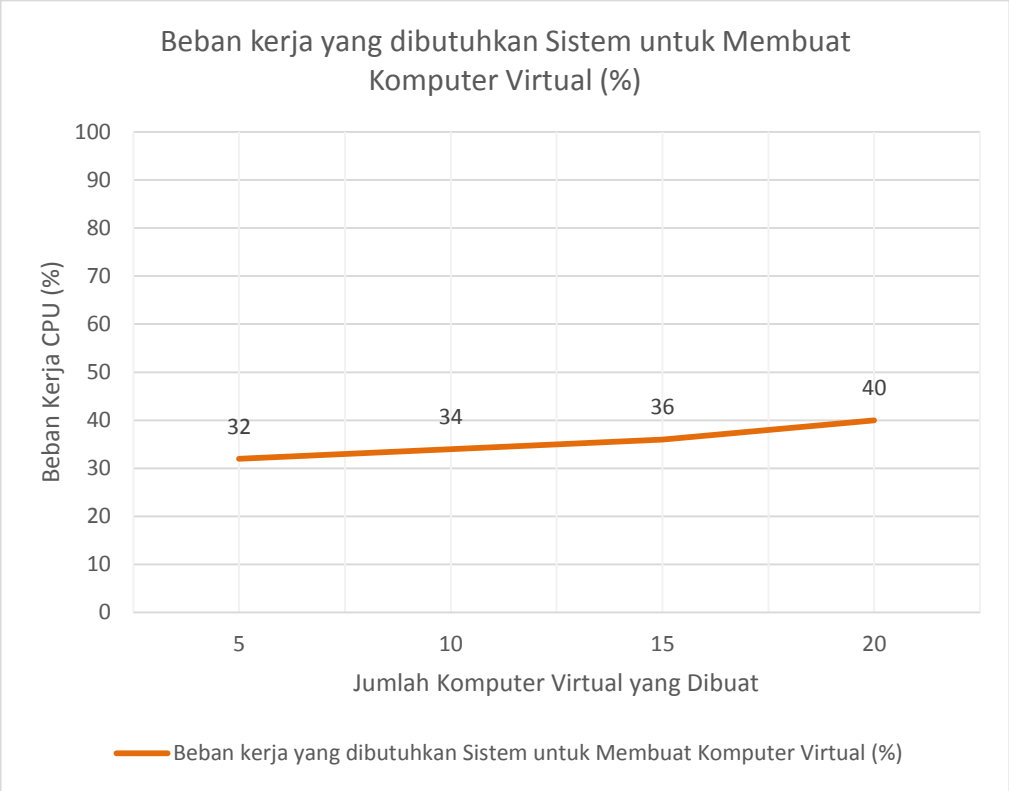
Analisis dilakukan terhadap hasil dari pengujian yang telah diterapkan. Sistem *back end* yang dibangun mampu menyediakan kebutuhan yang telah disebutkan sebelumnya. Pengujian dilakukan untuk mengetahui seberapa lama waktu yang dibutuhkan dari sistem dalam membuat, menyalakan kembali, dan menyimpan kondisi terakhir dari komputer virtual Pengujian sistem penyedian komputer virtual dilakukan dengan 4 skenario yang berbeda dimana tiap-tiap skenario terdapat perbadaan dalam jumlah komputer virtual yang dibuat. Hasil pengujian sistem penyediaan komputer virtual telah ditampilkan pada tabel 5.1. Grafik dari pengujian sistem penyediaan komputer virtual dapat dilihat pada gambar 5.3.



Gambar 5.3 Grafik Kinerja Sistem Penyediaan Komputer Virtual

Gambar 5.3 merupakan grafik waktu yang dibutuhkan untuk membuat komputer virtual untuk tiap skenario, grafik menunjukkan adanya hubungan dimana semakin banyak komputer virtual yang dipesan maka waktu yang dibutuhkan untuk membuat komputer virtual yang dipesan juga semakin lama.

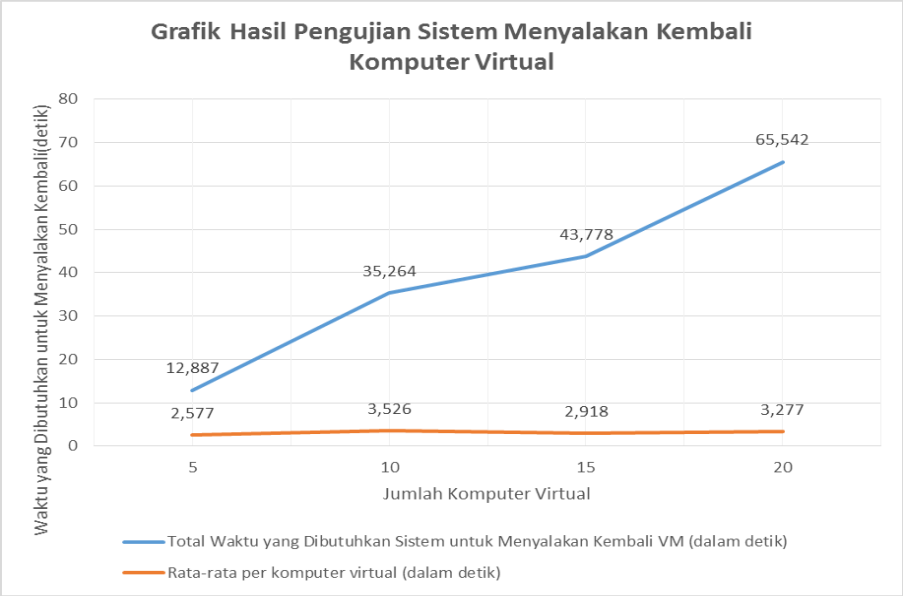
Pada grafik ini juga dapat ditarik kesimpulan bahwa untuk setiap komputer virtual yang dibuat dalam tiap skenario, sistem hanya membutuhkan waktu ± 4 detik. Kondisi ini masih dikatakan dapat ditolerir.



Gambar 5.4 Grafik Beban Kerja CPU untuk Membuat Komputer Virtual

Gambar 5.4 merupakan grafik beban kerja yang dibutuhkan untuk membuat komputer virtual untuk tiap skenario, grafik menunjukkan adanya hubungan dimana semakin banyak komputer virtual yang dipesan maka sumber daya CPU yang dibutuhkan untuk membuat komputer virtual yang dipesan juga semakin besar.

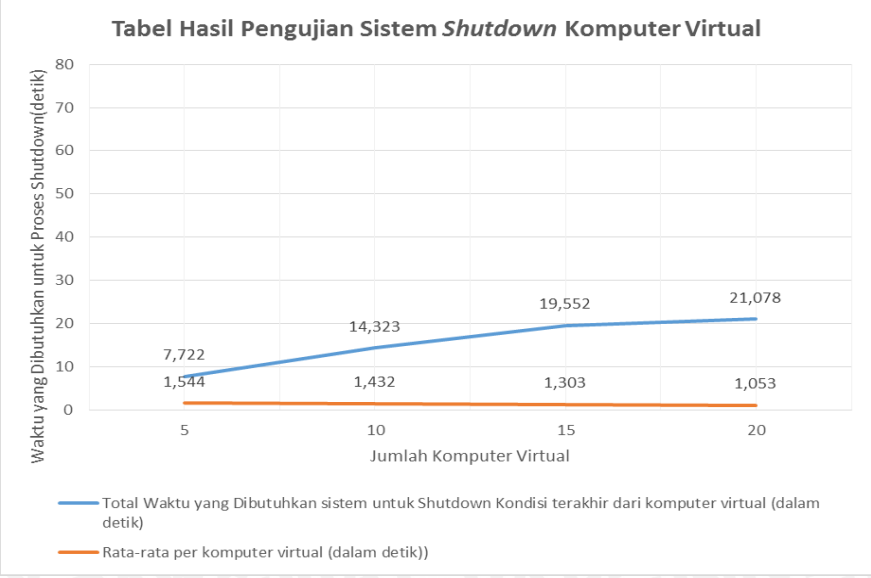
Pengujian sistem dalam menyalakan kembali komputer virtual dilakukan dengan 4 skenario yang berbeda dimana tiap-tiap skenario terdapat perbedaan dalam jumlah komputer virtual yang dinyalakan kembali. Hasil pengujian sistem dalam proses menyalakan kembali komputer virtual telah ditampilkan pada tabel 5.4. Grafik dari pengujian sistem dapat dilihat pada gambar 5.5.



Gambar 5.5 Grafik Kinerja Sistem Dalam Menyalakan Kembali Komputer Virtual

Pada grafik yang ditunjukkan pada Gambar 5.5 dapat ditarik kesimpulan bahwa dalam melakukan proses menyalakan kembali, semakin banyak komputer virtual yang dinyalakan kembali maka semakin lama juga waktu yang dibutuhkan. Dalam melakukan proses ini dalam tiap skenario, sistem hanya membutuhkan waktu ± 3 detik untuk tiap komputer virtual.

Pengujian sistem dalam *shutdown* komputer virtual dilakukan dengan 4 skenario yang berbeda dimana tiap-tiap skenario terdapat perbedaan dalam jumlah komputer virtual yang di-*shutdown*. Hasil pengujian sistem dalam melakukan *shutdown* komputer virtual telah ditampilkan pada tabel 5.5. Grafik dari pengujian sistem dapat dilihat pada gambar 5.6.



Gambar 5.6 Grafik Kinerja Sistem dalam Shutdown Komputer Virtual

Pada grafik yang ditunjukkan pada Gambar 5.6 dapat ditarik kesimpulan bahwa dalam melakukan proses shutdown komputer virtual, semakin banyak komputer virtual yang di-shutdown maka semakin lama juga waktu yang dibutuhkan. Dalam melakukan proses ini untuk dalam tiap skenario, sistem hanya mebutuhkan waktu rata ± 1,3 detik untuk tiap komputer virtual.

BAB 6 PENUTUP

Bab 6 berisi kesimpulan dari hasil implementasi dan analisis sistem serta saran untuk pengembangan sistem.

6.1 Kesimpulan

Dari hasil implementasi, pengujian, dan analisis dari penerapan aplikasi laboratorium virtual, dapat disimpulkan bahwa:

1. Laboratorium virtual jaringan komputer terdiri dari *front end* dan *back end*, di mana sistem yang dibangun menggunakan virtualisasi di lingkungan Linux ini mampu menyediakan komputer virtual dengan jenis *container* dan memberikan akses *remote* kepada mahasiswa menggunakan *virtual network computing*.
2. Dalam sistem ini laboratorium virtual yang dibangun, dalam setiap komputer virtual di-install paket J*ava Runtime environment* (JRE) dan *Java Development Kit* (JDK) untuk melakukan praktikum pemrograman *socket*.
3. Berdasarkan hasil kinerja sistem, didapatkan hasil bahwa semakin banyak komputer virtual yang dipesan maka semakin lama waktu yang dibutuhkan untuk membuatnya, serta sumber daya yang digunakan berbanding lurus dengan jumlah komputer virtual yang dibuat.
4. Berdasarkan hasil analisa pengujian sistem, didapatkan bahwa dibutuhkan waktu ± 4 detik untuk membuat sebuah komputer virtual, dan sistem membutuhkan waktu ± 3 detik untuk menyalakan sebuah komputer virtual dari kondisi *shutoff* menjadi *running*, serta sistem mebutuhkan waktu ± 1,3 detik untuk mematikan sebuah komputer virtual.

6.2 Saran

Saran yang dapat disampaikan penulis untuk pengembangan aplikasi laboratorium virtual adalah:

1. Perlu dilakukan penelitian lebih lanjut untuk menambah materi-materi praktikum agar bisa mengakomodasi lebih banyak materi praktikum yang lain.
2. Kedepannya sistem ini bisa diimplementasikan ke dalam laboratorium lainnya yang ada di FILKOM Universitas Brawijaya.

DAFTAR PUSTAKA

Adiputra, F., 2015. CONTAINER DAN DOCKER: TEKNIK VIRTUALISASI DALAM PENGELOLAAN BANYAK APLIKASI WEB. *Jurnal Simantec,* 4(June, 3 2015), pp. 167-176.

Basu, P. et al., 2013. Collaborating Remote Computer Laboratory and Distance Learning Approach for Hands-on IT Education. *Advanced Science and Technology Letters Advanced Science and Technology Letters,* Volume 8, pp. 222-229.

Bigelow, S., 2015. *Search server virtualization.* [Online] Available at: http://searchservervirtualization.techtarget.com/feature/Five-cons-of-container-technology
[Diakses 15 Agustus 2016].

Cacciatore, K. et al., 2015. Exploring Opportunities : Containers and OpenStack. p. 2.

Davoli, F., Meyer, N., Pugliese, R. & Zappatore, S., 2010. *Remote Instrumentation and Virtual Laboratories.* Poznan, Poland: Springer.

Dua, R., Raja, R. & Kakadia, D., 2014. Virtualization vs Containerization to support PaaS. *2014 IEEE International Conference on Cloud Engineering,* pp. 610-614.

Dzulqarnain, 2015. Perancangan dan Implementasi Arsitektur Laboratorium Virtual Purwarupa Praktikum Jaringan Komputer. *Computer Science Minor Thesis.*

Furht, B. & Escalante, A., 2010. *Handbook of Cloud Computing.* New York: Springer.

Jakma, P. & Lamparter, D., 2014. Introduction to the Quagga Routing Suite. *IEEE Network,* pp. 42-48.

Jawwad, A., 2014. Implementation Application Munin And Monit For Server Monitoring On Debian Squezee Server (Study Case: SMK Tunas Harapan Pati Departement of Animation).

Jones, T., 2010. *Datamation.* [Online] Available at: http://www.datamation.com/netsys/article.php/3884091/Virtualization.htm
[Diakses 15 Agustus 2016].

Jung, P., 2009. *Linux Journal.* [Online] Available at: http://www.linuxjournal.com/article/10248
[Diakses 15 Agustus 2016].

Kumar, R. et al., 2014. Open Source Solution for Cloud Computing Platform Using OpenStack. *International Journal of Computer Science and Mobile Computing,* Volume III, pp. 89-98.

Menken, I. & Blokdijk, G., 2010. *Cloud Computing Virtualization Specialist Complete Certification Kit: Virtualization.* 2nd penyunt. London: Emereo Pty Ltd.

Patrizio, A., 2015. *Datamation.* [Online] Available at: http://www.datamation.com/data-center/virtualization-vs.-containers-what-you-need-to-know.html [Diakses 15 Agustus 2016].

Radez, D., 2015. *OpenStack Essentials.* Birmingham: Packt Publishing.

Richardson, T., Stafford-fasser, Q., Wood, K. R. & Hopper, A., 1998. Virtual Network Computing. *IEEE Internet Computing,* pp. 34-38.

Sawant, A. & Meshran, B., 2013. Network programming in Java using Socket. *International Journal of Engineering Research and Applications (IJERA),* 3(1, January -February 2013), pp. 1299-1305.

Semnanian, A., Pham, J., Englert, B. & Wu, X., 2011. Virtualization Technology and its Impact on Computer Hardware Architecture. *Eighth International Conference on Information Technology: New Generations,* pp. 719-724.

Seo, K.-T.et al., 2014. Performance Comparison Analysis of Linux Container and Virtual Machine for Building Cloud. *Advanced Science and Technology Letters,* Volume 66, pp. 105-111.

Smith, J. & Nair, R., 2005. *Virtual Machines Versatile Platform For System and Process.* San Francisco: Elsevier.

Suresh, S. & Kannan, M., 2014. A Study on System Virtualization Techniques. *International Journal of Advanced Research in Computer Science & Technology,* 2(1 Jan-March 2014), pp. 134-139.

Tirtawidjaja, T., 2014. *tedytirta.* [Online] Available at: http://tedytirta.com/2014/05/06/virtualisasi-dengan-kvm-linux/ [Diakses 18 08 2016].

Tomar, H. & Shaney, G., 2013. Virtual Network Computing- A Prodigious Technology For. *Journal of Engineering Research and Applications,* 3(6, Nov-Dec 2013), pp. 59-63.

Xu, L., Huang, D. & Tsai, W.-T., 2014. Cloud-Based Virtual Laboratory for Network Security Education. *IEEE TRANSACTIONS ON EDUCATION,* Volume 57, pp. 145-150.