TUGAS AKHIR - KI091391

RANCANG BANGUN INFRASTRUKTUR *MULTI-TENANT* PADA KOMPUTASI AWAN MENGGUNAKAN KERANGKA KERJA CLOUDSTACK DENGAN STUDI KASUS APLIKASI MANAJEMEN RESTORAN

Dimas Yoga Pratama
NRP 5110100039

Dosen Pembimbing
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014

FINAL PROJECT - KI091391

Design and Implementation of Multi-Tenant Infrastructure on Cloud Computing using CloudStack Framework with Restaurant Management Application as a Case Study

Dimas Yoga Pratama
NRP 5110100039

Advisor
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2014

Rancang Bangun Infrastruktur *Multi-Tenant* pada Komputasi Awan menggunakan Kerangka Kerja CloudStack dengan Studi Kasus Aplikasi Manajemen Restoran

Nama Mahasiswa : Dimas Yoga Pratama
NRP : 5110100039
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing 1 : Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.
Dosen Pembimbing 2 : Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

ABSTRAK

Komputasi awan merupakan model lingkungan berbagi yang dibangun di atas infrastruktur IT yang tervirtualisasi dengan sangat efisien dan serba otomatis di mana sumber daya IT dapat disediakan sewaktu-waktu sesuai dengan kebutuhan dari mana saja melalui jaringan yang luas. Dengan menggunakan komputasi awan, pengguna tidak perlu menyediakan infrastruktur, konfigurasi, dan pemeliharaan perangkat lunak. Sistem *Multi-Tenant* merupakan konsep utama dalam komputasi awan. *Multi-Tenant* merupakan konsep di mana satu kumpulan sumber daya dapat dibagi penggunaanya oleh beberapa pengguna. CloudStack merupakan salah satu platform awan yang menawarkan infrastruktur komputasi awan sebagai layanan (IaaS). CloudStack mendukung pengembang dalam mengkustomisasi sebuah komputasi awan dengan dukungan API yang luas.

Dengan menjamurnya layanan yang dibangun dengan infrastruktur komputasi awan serta usangnya teknologi lama, dibuatlah sebuah sistem yang mengadopsi *multitenancy* pada infrastruktur komputasi awan dengan menggunakan CloudStack. Teknologi virtualisasi digunakan sebagai dasar *multitenancy*

sistem, sehingga data dan aplikasi setiap pengguna layanan tidak akan bercampur dengan data pengguna lain. Sistem ini juga menerapkan mekanisme skalabilitas di mana sistem secara otomatis akan menambah sumber daya perangkat ketika kapasitas sistem yang sudah ada tidak memungkinkan untuk melayani pengguna.

Sistem yang telah dibuat kemudian diuji fungsionalitasnya. Pengujian dilakukan melalui beberapa skenario yang telah ditentukan. Hasil pengujian menunjukkan sistem dapat berjalan pada infrastruktur komputasi awan, sistem mampu menerapkan *multitenancy* berbasis virtualisasi dalam menyediakan layanan aplikasi pada pengguna. Sistem juga menyediakan sebuah mekanisme *monitoring* kepada penyedia layanan.

***Kata kunci**:* **CloudStack, Komputasi Awan, *Multi-Tenant,* Sistem *Monitoring*.**

Design and Implementation of Multi-Tenant Infrastructure on Cloud Computing using CloudStack Framework with Restaurant Management Application as a Case Study

Student's Name	**: Dimas Yoga Pratama**
Student's ID	**: 5110100039**
Department	**: Teknik Informatika FTIf-ITS**
First Advisor	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.**
Second Advisor	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**

ABSTRACT

Cloud is a shared environment built on a highly efficient, automated, and preferably virtualized IT infrastructure where IT resources can be provisioned on demand from anywhere over a broad network. By using the cloud computing, user does not need to provide the infrastructure, configuration, and software maintenance. Multitenancy is the main concept in cloud computing, the idea of multitenancy is a set of resources that can be utilized by multiple users. CloudStack is a cloud platform that offers cloud computing infrastructure as a service (IaaS). CloudStack supports developers in customizing a cloud computing system with an extensive API support.

With the growth of services that are built based on cloud computing technology as well as outdated of the old technology, therefore created a system that adopts multitenancy in the cloud computing infrastructure using CloudStack Framework. Virtualization technology is used as a basis for multitenancy in this system, so that users data and application is not mixed up with other. This system also implements scalability mechanisms in which the system automatically adds the backup computing

resources when the capacity of the existing system is not enable to serve the users.

Functionality and performance of the system is tested and evaluated in various scenarios. The results of the experiment show that the proposed system is capable of implementing multitenancy in providing services to the users. The system also provides monitoring mechanisms for the corresponding service provider.

Keywords: **CloudStack, Cloud Computing, Multi-tenant, Monitoring System.**

LEMBAR PENGESAHAN

Rancang Bangun Infrastruktur *Multi-Tenant* pada Komputasi Awan menggunakan Kerangka Kerja CloudStack dengan Studi Kasus Aplikasi Manajemen Restoran

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
DIMAS YOGA PRATAMA
NRP : 5110100039

Disetujui oleh Dosen Pembimbing Tugas Akhir :

ROYYANA MUSLIM IJTIHADIE, S.Kom., M.Kom., Ph.D.
NIP: 197708242006041001 ..... (pembimbing 1)

WAKITHO WIBISONO, S.Kom., M.Eng., Ph.D.
NIP: 197410222000031001 ..... (pembimbing 2)

SURABAYA
JULI 2014

KATA PENGANTAR

Alhamdulillahirabil'alamin, segala puji bagi Allah Subhanahu Wata'alla, yang telah melimpahkan segala karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul : "***Rancang Bangun Infrastruktur Multi-Tenant pada Komputasi Awan menggunakan Kerangka Kerja CloudStack dengan Studi Kasus Aplikasi Manajemen Restoran"***

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT atas limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Mama, Papa, Om Tri, Mami, Dek Ditya, Dek Nonik, Dek Sulthan dan keluarga besar yang selalu memberikan dukungan penuh untuk menyelesaikan Tugas Akhir ini.
3. Bapak Royyana dan Bapak Waskitho selaku dosen pembimbing pertama dan kedua yang telah bersedia meluangkan waktu serta memberikan kepercayaan, dukungan, bimbingan kepada penulis.
4. Bapak, Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
5. Seluruh staf dan karyawan FTIf ITS yang banyak memberikan kelancaran administrasi akademik kepada penulis.
6. Ayu Dewanti Putri yang telah meluangkan waktunya untuk mendengarkan keluh kesah penulis selama mengerjakan Tugas Akhir ini.
7. Griya Asri Family, Dicky, Aji, Wido, Caca, Haryo, Kessya, Lala, Wildhan, dan Fitri yang selalu memberikan semangat dan masukan kepada penulis.

8. Seluruh teman Teknik Informatika 2010 yang telah bersama selama empat tahun atas ilmu, saran, dan dukungan terhadap pengerjaan Tugas Akhir ini.
9. Serta pihak-pihak lain yang tidak dapat disebutkan disini yang telah banyak membantu penulis dalam penyusunan Tugas Akhir ini.

Penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan dalam penulisan Tugas Akhir ini, maka penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, Juli 2014

Dimas Yoga Pratama

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Bab ini berisi penjelasan mengenai garis besar penulisan Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Beberapa tahun belakangan ini teknologi komputasi awan (*cloud computing*) menjadi perbincangan hangat di kalangan pemerhati teknologi. Meningkatnya kebutuhan baik organisasi maupun perorangan dalam kebutuhan komputasi serta penyimpanan data yang besar memaksa kita untuk beralih ke teknologi komputasi awan. Hal ini disambut dengan banyaknya penyedia layanan komputasi awan yang mulai bermunculan, seperti Rackspace[1], Amazon EC2[2], dan Bluelock[3] yang menyediakan IaaS (*Infrastructure as a Service*) hingga perusahaan seperti Salesforce[4], Dropbox[5] dan Office365[6] yang menyediakan SaaS (*Software as a Service*) [1]. Salah satu kelebihan teknologi awan dibandingkan dengan teknologi lama adalah pada teknologi awan, pihak penyedia layanan dapat dengan mudah melakukan *scaling* dalam hal RAM (*Random Access Memory*), CPU (*Central Processing Unit*), penyimpanan, dan lain sebagainya tanpa harus menunggu teknisi untuk mematikan *server*, memasang RAM baru, dan menyalakannya kembali [2]. Teknologi awan juga mendukung adanya virtualisasi perangkat keras sehingga memudahkan dalam

[1] Rackspace. www.rackspace.com
[2] Amazon EC2. www.aws.amazon.com/ec2/
[3] Bluelock. www.bluelock.com
[4] Salesforce. www.salesforce.com
[5] Dropbox. www.dropbox.com
[6] Office365. www.365*login*.com

menciptakan sebuah lingkungan *multi-tenant* yang menghemat sumber daya komputasi.

CloudStack merupakan platform perangkat lunak *open-source* yang dapat menggabungkan sumber daya komputasi untuk membangun jaringan infrastruktur komputasi awan [3]. CloudStack mempunyai fitur yang dapat menggabungkan berbagai macam perangkat keras yang berbeda menjadi satu portal tunggal. CloudStack mendukung penggunaan Hypervisor yang lebih luas daripada platform komputasi awan lainnya seperti KVM[7], Xen*Server*[8], dan VMware[9]. Cloudstack memudahkan pengembang dalam menciptakan layanan *multi-tenant*.

Pada Tugas Akhir ini akan dibangun sebuah sistem pada infrastruktur komputasi awan yang menyediakan layanan aplikasi manajemen restoran siap pakai kepada banyak pemilik restoran (*multi-tenant*) serta sistem *monitoring* yang digunakan untuk kebutuhan admin. Aplikasi yang diberikan pada *tenant* mempunyai struktur dan *database* yang sama namun aplikasi antar *tenant* tidak akan berhubungan. Sistem yang dibangun pada sebuah lingkungan komputasi awan mempunyai skalabilitas yang lebih tinggi daripada sistem yang dibangun pada teknologi lama, serta mendukung adanya virtualisasi yang dapat digunakan sebagai implementasi *multitenancy*, hal ini mampu menghemat *resources* dari suatu sumber daya perangkat komputasi.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut.

1. Bagaimana membangun infrastruktur komputasi awan menggunakan sumber daya mandiri dengan kerangka kerja CloudStack.

---

[7] KVM. www.linux-kvm.org
[8] Xen*Server*. www.xen*server*.org
[9] VMware. www.vmware.com

2. Bagaimana membangun sebuah sistem *multi-tenant* yang menyediakan layanan aplikasi siap pakai dengan infrastruktur komputasi awan.
3. Bagaimana membangun sebuah sistem *monitoring* terpadu yang dapat memantau dan mengukur data setiap aplikasi manajemen restoran milik pengguna serta memantau kapasitas dari infrastruktur komputasi awan.

1.3. Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, di antaranya sebagai berikut.

1. Sistem ini diaplikasikan pada platform CloudStack.
2. Sistem ini menyediakan satu jenis *image* yaitu Ubuntu 12.04 dengan kapasitas komputasi dan penyimpanan yang sama.
3. Sistem menyediakan satu jenis aplikasi manajemen restoran - *open-source* untuk setiap *tenant*.
4. Sistem memberikan satu buah VM (*Virtual Machine*) untuk setiap *tenant*.
5. Sistem ini menggunakan beberapa perangkat keras, yaitu:
    a. empat buah komputer.
    b. satu buah *switch*.

1.4. Tujuan dan Manfaat

Tujuan dari pengerjaan Tugas Akhir ini adalah sebagai berikut.

1. Sistem dapat menyediakan platform aplikasi manajemen restoran siap pakai secara *multi-tenant*.
2. Sistem dapat melakukan *monitoring* penggunaan aplikasi manajemen restoran pada setiap VM milik *tenant*.
3. Sistem dapat melakukan *monitoring* kapasitas infrastruktur komputasi awan.

Manfaat dari pengerjaan Tugas Akhir ini adalah sebagai berikut.

1. Secara otomatis memberikan layanan pada *tenant* sebuah VM yang terpasang aplikasi manajemen restoran siap pakai.
2. Memberikan kemudahan pada penyedia layanan dalam hal *monitoring* aplikasi manajemen restoran pada setiap VM dan *monitoring* kapasitas infrastruktur awan.

1.5. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu.

1. Penyusunan Proposal Tugas Akhir

   Pada tahap awal dalam memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Dalam proposal tersebut menjelaskan bagaimana membangun sebuah infrastruktur komputasi awan serta memasang sebuah aplikasi manajemen restoran pada setiap VM yang diciptakan dan secara otomatis memberikan alamat aplikasi VM tersebut pada *tenant* yang meminta.

2. Studi Literatur

   Tahap ini merupakan tahap pengumpulan informasi dan pembelajaran materi yang akan digunakan dalam pengerjaan Tugas Akhir. Studi literatur meliputi diskusi dan pemahaman mengenai topik yang berhubungan dengan Tugas Akhir, diantaranya mengenai:
   1. Implementasi CloudStack pada CentOS dengan KVM sebagai Hypervisor.
   2. Mekanisme pembuatan portal sistem *multi-tenant* dengan memanfaatkan CloudStack API (*Application Programming Interface*).
   3. Mekanisme *monitoring* VM serta pengiriman data

monitoring dari setiap VM ke portal.

3. Perancangan Sistem

   Tahap ini merupakan perancangan sistem dengan menggunakan studi literatur dan mempelajari konsep aplikasi yang akan dibuat. Dengan berbekal teori, metode, dan informasi yang sudah terkumpul pada tahap sebelumnya diharapkan dapat membantu dalam proses perancangan sistem.

4. Implementasi perangkat lunak

   Tahap ini merupakan tahap penerapan bagaimana sistem secara otomatis dapat menanggapi permintaan *tenant* melalui portal yang telah dibuat, memberikan VM yang sesuai dengan *template* yang telah disediakan sebelumnya. VM yang telah dibuat mempunyai alamat IP unik yang nantinya diberikan kepada *tenant*.

5. Pengujian dan Evaluasi

   Dalam tahap ini pengujian akan dilakukan pada empat computer. satu komputer berperan ganda sebagai MS (*Management Server*) dan *Host* VM dan dua komputer berperan sebagai *Host* VM serta satu komputer lain sebagai portal layanan. Pengujian akan dilakukan saat *tenant* melakukan *request* sebuah aplikasi manajemen restoran pada sistem, maka sistem akan menanggapi permintaan *tenant* dengan menciptakan sebuah VM yang terdapat aplikasi manajemen restoran dengan sebuah alamat IP unik yang nantinya alamat tersebut akan dikembalikan kepada *tenant* melalui kustom portal. Data VM dan aplikasi milik *tenant* yang telah tercipta akan dikumpulkan dan dikirimkan kembali menuju *server* pusat untuk keperluan *monitoring*. Tahap evaluasi akan dilakukan berdasarkan hasil dari tahap pengujian.

6. Penyusunan Buku Tugas Akhir

   Pada tahap ini disusun laporan Tugas Akhir sebagai dokumentasi pelaksanaan Tugas Akhir, yang mencakup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.6. Sistematika Penulisan

Secara garis besar, buku Tugas Akhir ini terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir. Selain itu juga berisi permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan buku Tugas Akhir.

Bab II Dasar Teori

Bab ini berisi penjelasan secara umum mengenai beberapa teori penunjang yang mendukung pengembangan Tugas Akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini berisi penjelasan mengenai diagram aplikasi, arsitektur perangkat lunak, diagram alir perangkat lunak, dan desain antarmuka perangkat lunak yang dibuat.

Bab IV Implementasi Perangkat Lunak

Bab ini berisi implementasi dari perancangan perangkat lunak yang telah dibuat pada bab sebelumnya. Implementasi berupa *pseudocode* dari fungsi utama dan *screenshot* perangkat lunak.

Bab V Pengujian dan Evaluasi

Bab ini berisi penjelasan kemampuan perangkat lunak dengan melakukan beberapa pengujian yaitu pengujian fungsionalitas dan pengujian performa dalam beberapa skenario.

Bab VI Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

BAB II
DASAR TEORI

Bab ini berisi penjelasan mengenai teori-teori yang menjadi dasar pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberi gambaran secara umum mengenai perangkat lunak yang dibuat. Adapun teori-teori tersebut meliputi Komputasi awan, Apache CloudStack, JSON, Hypervisor, KVM, CentOS, Libvirt, XML, PHP, dan Sistem *Monitoring*.

2.1. Komputasi Awan

Komputasi awan [4] adalah sebuah model lingkungan *multi-tenant* yang dibangun di atas infrastruktur teknologi informasi yang sangat efisien, otomatis, dan mendukung virtualisasi di mana sumber daya komputasi dapat disediakan sewaktu-waktu sesuai dengan kebutuhan. Komputasi awan dapat dapat disebarkan dengan berbagai model, seperti berikut.

1. *Private cloud*: Pada model ini, infrastruktur awan dioperasikan hanya untuk sebuah organisasi. *Private cloud* dapat dikelola oleh organisasi itu sendiri ataupun dengan menggunakan jasa penyedia infrastruktur awan.
2. *Public cloud*: Pada model ini, layanan komputasi awan disediakan untuk masyarakat umum atau kelompok industri berskala besar yang dimiliki dan dikelola oleh organisasi yang menyediakan layanan komputasi awan.
3. *Community cloud*: Pada model ini, infrastruktur komputasi awan dibagi oleh beberapa organisasi dan didukung oleh komunitas tertentu dengan tujuan yang sama. Infrastruktur ini dapat dikelola oleh organisasi atau penyedia pihak ketiga.

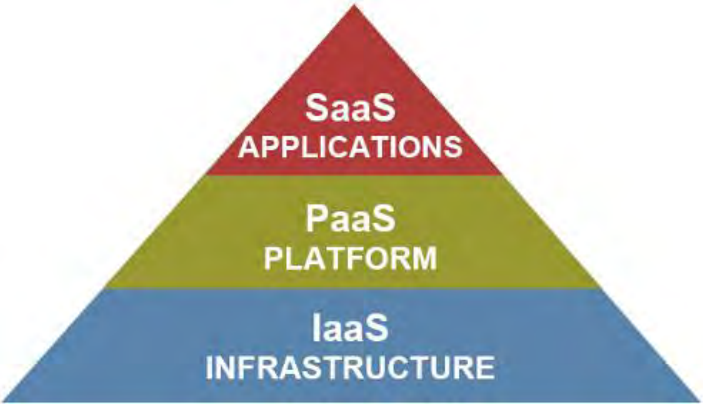
Struktur komputasi awan pada Gambar 2.1 mempunyai tiga model layanan yang dapat disediakan, seperti berikut.

1. *Software as a Service* (SaaS)

SaaS adalah layanan dari komputasi awan dimana pengguna dapat menggunakan perangkat lunak yang telah disediakan oleh penyedia layanan awan. Pengguna cukup tahu bahwa perangkat lunak bisa berjalan dan bisa digunakan dengan baik.

2. *Platform as a Service* (PaaS)
   PaaS adalah layanan komputasi awan yang menyediakan modul siap pakai. PaaS digunakan untuk mengembangkan aplikasi yang berjalan di atas platform tersebut. Keuntungan layanan PaaS bagi pengembang adalah mereka dapat fokus pada aplikasi yang mereka buat tanpa memikirkan pemeliharaan dari lingkungan komputasi.
3. *Infrastructure as a Service* (IaaS)
   IaaS adalah layanan yang menyewakan sumber daya teknologi komputasi meliputi media penyimpanan, *processing power*, *memory*, sistem operasi dan kapasitas jaringan yang digunakan untuk menjalankan aplikasi.

Pada Tugas Akhir ini akan dibangun sebuah komputasi awan dengan sebuah platform jadi yang diberikan sebagai layanan kepada *tenant*. Layanan tersebut berupa aplikasi manajemen restoran siap pakai.



Gambar 2.1. Struktur Komputasi Awan Berdasarkan Model Layanan [20]

2.2. Apache CloudStack

CloudStack [5] merupakan platform perangkat lunak *open-source* yang menawarkan infrastruktur komputasi awan sebagai layanan. CloudStack awalnya diciptakan oleh cloud.com[10]. Pada Mei 2010, sebagian besar dari perangkat lunak dibuat dibawah lisensi GPL v3, hampir 5% dari sumber kode tidak disebarkan untuk umum. Pada Juli 2011, Citrix[11] membeli cloud.com dan pada April 2012, Citrix mendonasikan CloudStack kepada Apache Software Foundation dan mengganti lisensi menjadi Apache 2.0.

Apache CloudStack menyediakan fitur canggih untuk menyediakan layanan *multi-tenant* dalam lingkungan komputasi awan yang aman. Hanya dengan satu klik, *server* virtual dapat digunakan dari *template* yang telah ditentukan. Instansi virtual dapat dimatikan, diberhentikan, dan dinyalakan kembali melalui antarmuka web, baris perintah, atau dengan memanggil CloudStack API (*Application Programming Interface*).

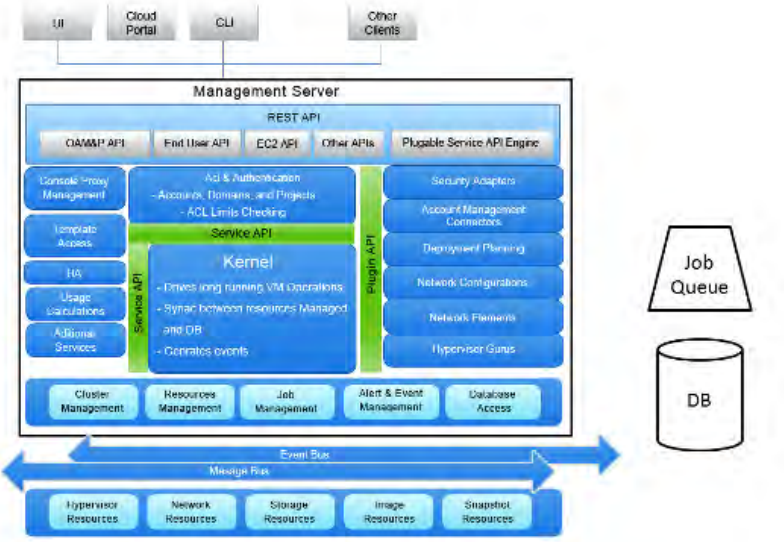
CloudStack mempunyai struktur yang bertingkat dapat dilihat pada Gambar 2.2, seperti manajemen dari banyak *host* fisik dari satu halaman muka. Struktur yang ada di CloudStack diantaranya sebagai berikut.

1. *Availability Zones*. merupakan deskripsi dari lokasi yang digunakan untuk alokasi VM (virtual machine) di penyimpanan data. Sebuah *Availability Zones* terdiri dari *Pod* dan Secondary Storage.
2. *Pod*s. Pengaturan perangkat keras dari *Clusters*. Sebuah *Pod* dapat terdiri dari satu *Cluster* atau lebih, dan terdapat sebuah arsitektur *switch* Layer–2.
3. *Clusters*. merupakan kumpulan dari *host* fisik yang identik yang berjalan di Hypervisor yang sama. Sebuah Cluster mempunyai tempat penyimpanan khusus yang bertugas untuk menyimpan VM.

---

[10] cloud.com. www.cloud.com
[11] Citrix. www.citrix.com

4. *Primary Storage*. merupakan salah satu bagian dari Cluster yang digunakan untuk penyimpanan VM.
5. *Secondary Storage*. digunakan untuk menyimpan image VM, *template*, dan *snapshot*.



Gambar 2.2. Arsitektur CloudStack [19]

Dalam Tugas Akhir ini akan digunakan CloudStack dalam pondasi utama infrastruktur awan karena CloudStack mendukung pengembang dalam mengkustomisasi portal dengan fitur API yang luas dan dapat disesuaikan dengan layanan yang akan dibangun. Tugas Akhir ini juga menggunakan beberapa komponen tambahan agar infrastruktur komputasi awan dapat berjalan dengan maksimal, diantaranya.

1. CentOS: merupakan sistem operasi Linux dari Red Hat Entreprise Linux (RHEL).

2. KVM: merupakan Hypervisor pada sistem operasi Linux yang mengatur konfigurasi VM dalam satu perangkat.

2.2.1. Perbandingan Apache CloudStack dengan Kerangka Kerja *Open-source* Lain

Terdapat beberapa kerangka kerja *open-source* komputasi awan selain CloudStack yang mendukung pembuatan infrastruktur komputasi awan, diantaranya adalah OpenStack dan OpenNebula. Berikut penjelasan dari masing-masing kerangka kerja.

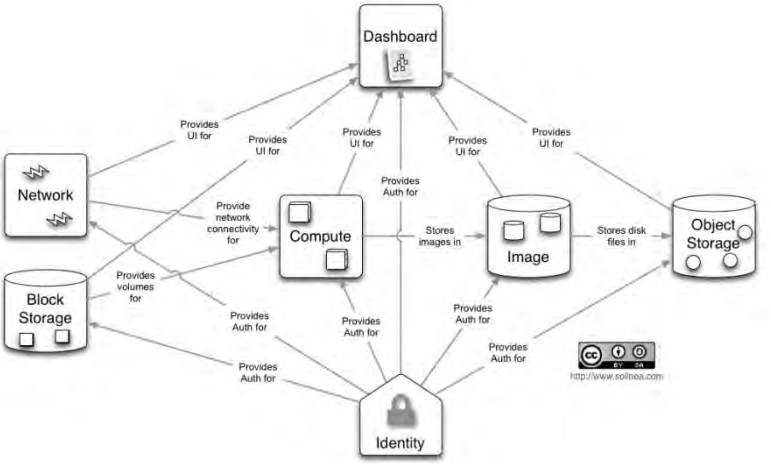
2.2.1.1. OpenStack

OpenStack merupakan perangkat lunak *open-source* yang menyediakan infrastruktur sebagai layanan (IaaS) [6] . OpenStack merupakan perangkat lunak *open-source* yang dirilis di bawah lisensi Apache. Proyek ini dikelola oleh OpenStack Foundation.

OpenStack dibagi menjadi beberapa komponen yang ditunjukkan pada Gambar 2.3. Penjelasan dari komponen – komponen tersebut adalah sebagai berikut.

1. *OpenStack Nova* (*Compute*) : *OpenStack Nova* merupakan mesin komputasi utama dalam kerangka kerja OpenStack. Nova digunakan sebagai penyebaran dan pengaturan VM dan *instance* lainnya.
2. *OpenStack Swift* (*Object Storage*) : *OpenStack Swift* merupakan sistem penyimpanan objek dan berkas pada VM.
3. *OpenStack Network* : *OpenStack Network* merupakan komponen yang bertanggung jawab dalam manajemen alamat IP pada VM dan *instance* lainnya.
4. *OpenStack Keystone* (*Identity*) : merupakan sebuah pusat manajemen layanan untuk memudahkan pengaturan pengguna.

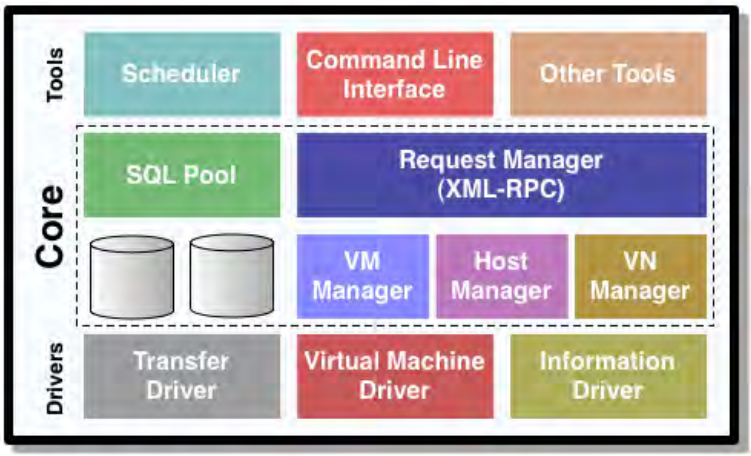
5. *OpenStack Horizon* (*Dashboard*) : merupakan layanan *dashboard* dari OpenStack untuk memudahkan pengaturan VM dan konfigurasi lainnya bagi pengguna.
6. *OpenStack Glance* (*Image*): merupakan manajemen *image* Dari VM OpenStack. OpenStack Glance mendukung berbagai macam image seperti. AMI, VHD, VDI, QCOW2, VMDK, dan OVF.



Gambar 2.3. Arsitektur OpenStack [21]

2.2.1.2. OpenNebula

OpenNebula [7] merupakan kerangka kerja komputasi awan *open-source* yang merupakan bagian dari sebuah proyek penelitian di tahun 2005. OpenNebula didirikan oleh Ignacio M. Llovrente dan Ruben S. Montero.



Gambar 2.4. Arsitektur OpenNebula [18]

Arsitektur OpenNebula dapat dibagi menjadi tiga layer. Hal ini dapat dilihat pada Gambar 2.4.

1. Tools, lapisan ini mengandung *tools* yang didistribusikan dengan OpenNebula, seperti CLI, penjadwalan, implementasi dari Libvirt API atau antarmuka *cloud* dengan metode *RESTful* dan juga aplikasi pihak ketiga yang dapat dengan mudah diciptakan menggunakan antarmuka XML-RPC atau API dari OpenNebula. Manajamen *tools* dikembangkan dengan menggunakan antarmuka yang disediakan oleh inti dari OpenNebula.
2. Core, inti dari OpenNebula terdiri dari sebuah set komponen untuk mengontrol dan melakukan *monitoring* VM, jaringan virtual, penyimpanan, dan *host*. Inti OpenNebula melakukan aksi seperti memantau *host*, atau membatalkan VM dengan memanggil *driver* yang sesuai.
3. Drivers, OpenNebula memiliki satu set modul yang dapat dicolokkan untuk berinteraksi dengan *middleware* tertentu, misalnya virtualisasi Hypervisor, layanan *cloud*, mekanisme

transfer sebuah berkas, atau layanan informasi. Adapter ini disebut sebagai *Drivers*.

2.3. JSON

JSON (*JavaScript Object Notation*) [8] merupakan sebuah format data tukar-menukar ringan. JSON memudahkan manusia untuk membaca dan menulis serta memudahkan komputer dalam mengurai dan menghasilkan data. JSON merupakan sebuah format teks yang independen namun memiliki pustaka yang dapat dikenali oleh berbagai macam bahasa pemrograman seperti C, C++, Java, JavaScript, Perl, Python dan lain-lain. Sifat inilah yang membuat JSON bahasa yang ideal untuk tukar-menukar data. JSON dibangun di atas dua struktur.

1. Sebuah koleksi dari nama/nilai yang berpasangan. Dalam berbagai bahasa pemrograman, koleksi ini dikenal sebagai objek, *record*, *struct,* kamus, *hash table*, *keyed list,* atau sebuah array asosiatif.
2. Sebuah daftar nilai yang urut, dalam sebagaian besar bahasa pemrograman dikenal sebagai array, vektor, *list*, *sequence*.

Pada JSON, sebuah objek merupakan himpunan tidak terstruktur dari pasangan atribut dan nilai. Deklarasi sebuah objek dimulai dengan '{' (kurung kurawal buka) dan diakhiri dengan '}' (kurung kurawal tutup). Setiap atribut diikuti oleh ':' (tanda titik dua) dan setiap pasangan atribut dan nilai dipisahkan oleh ',' (tanda koma). Sedangkan sebuah array merupakan kumpulan nilai yang terstruktur. Deklarasi array dimulai dengan '[' (kurung siku buka) dan diakhiri dengan ']' (kurung siku tutup). Setiap nilai dipisahkan oleh ',' (tanda koma). Sebuah nilai dapat berupa string yang diapit oleh tanda petik ganda, angka, true atau false, nilai null, objek, atau array. Gambar 2.5 merupakan contoh JSON.

Pada Tugas Akhir ini JSON digunakan dalam representasi hasil dari penggunaan API CloudStack, yang nantinya diolah lebih lanjut dalam sistem.

Gambar 2.5. Contoh JSON

2.4. CentOS

CentOS [9] adalah distribusi Linux yang didukung penuh oleh komunitas yang diberikan secara cuma-cuma kepada publik dari Red Hat untuk RHEL (Red Hat Entreprise Linux). CentOS pertama kali diliris pada bulan Mei 2004. CentOS Linux bertujuan untuk menggabungkan fungsional dengan RHEL. CentOS dibangun untuk menyediakan platform komputasi tanpa bayar berskala entreprise dan berusaha untuk mempertahankan kompatibilitas penuh dengan sumber utamanya, yaitu Red Hat. Setiap versi CentOS dipertahankan dengan durasi sepuluh tahun (dengan pembaruan keamanan dengan durasi dukungan dari Red Hat yang bevariasi tergantung sumber yang dirilis). Versi CentOS dirilis dua tahun setiap versi CentOS secara berkala diperbarui sekitar 6 bulan untuk mendukung perangkat keras baru. CentOS

memiliki keamanan yang terjamin, pemeliharaan yang mudah, dapat diprediksi dan diproduksi oleh Linux.

Pada Tugas Akhir ini menggunakan CentOS sebagai dasar dari sistem operasi infrastruktur komputasi awan karena CentOS merupakan salah satu sistem operasi yang kompatibel dengan kerangka kerja CloudStack.

2.5. NFS

NFS [10] atau Network File System adalah teknologi untuk pembagian sumber daya antara jaringan lokal. NFS memungkinkan data untuk disimpan pada *server* pusat dan dapat dengan mudah diakses dari perangkat klien dalam konfigurasi klien/*server*. NFS berasal dari mekanisme sistem berkas terdistribusi. Hal ini umumnya diimplementasikan dalam lingkungan komputasi di mana manajemen terpusat dari data dan sumber daya merupakan hal yang penting. NFS bekerja pada semua jaringan berbasis IP. NFS menggunakan TCP dan UDP untuk akses dan pengiriman data.

NFS diimplementasikan dalam model komputasi klien/*server*, dimana *server* NFS mengelola otentikasi, otorisasi dan manajemen dari klien serta data yang terbagi bersama.

Pada Tugas Akhir ini teknologi NFS digunakan dalam hal penyimpanan data VM dan *template*.

2.6. Hypervisor

Hypervisor [11] merupakan perangkat lunak yang bekerja di bawah platform komputasi awan. Hypervisor merupakan program yang dapat menjadikan sebuah perangkat keras mampu menjalankan VM yang berbeda. Hypervisor bekerja dengan membagi sumber daya yang dimiliki perangkat keras kepada VM yang tercipta di dalamnya, sehingga VM tersebut mempunyai spesifikasi prosesor dan RAM yang sama. Sebuah VM dapat

melakukan permintaan kepada Hypervisor dengan banyak cara, salah satunya dengan menggunakan API.

Terdapat dua tipe Hypervisor, *Bare metal* dan *Embedded Hypervisor*. *Bare metal* atau disebut juga dengan Hypervisor murni merupakan Hypervisor yang berjalan secara langsung pada perangkat keras. Sedangkan *Embedded* Hypervisor berjalan pada sistem operasi konvensional. VM pada Hypervisor ini berjalan pada lapisan ketiga pada perangkat keras.

Pada Tugas Akhir ini menggunakan KVM Hypervisor sebagai alat untuk melakukan virtualisasi perangkat keras agar dapat menjalankan VM *tenant*.

2.6.1. KVM

KVM [12] (*Kernel-based Virtual Machine*) adalah salah satu Hypervisor berbasis *open-source* dari Linux yang mendukung prosesor berekstensi virtualisasi Intel VT dan AMD-V. KVM diciptakan oleh organisasi yang sama dengan sistem operasi CentOS yaitu RHEL (RedHat Entreprise Linux). KVM terdiri dari modul kernel yang dapat dimuat, kvm.ko dan modul prosesor yang spesifik yaitu kvm-intel.ko atau kvm-amd.ko

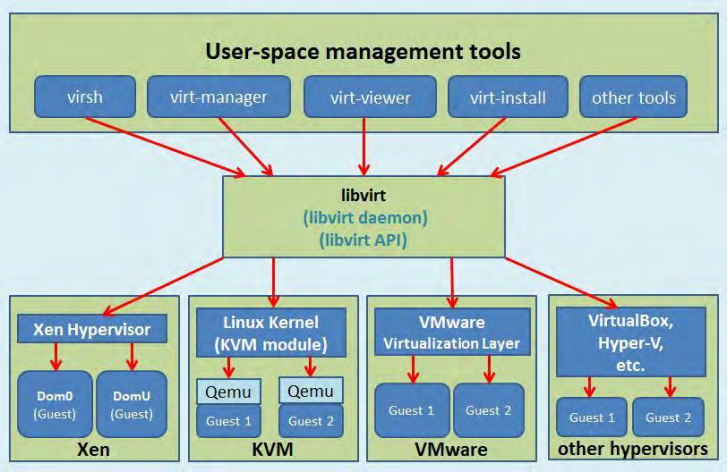
Dengan menggunakan KVM, satu buah perangkat keras dapat menjalankan beberapa VM berjenis Linux dan Windows yang tidak dimodifikasi. Setiap VM yang berjalan pada lingkungan KVM mempunyai *network card*, *storage*, *memory* dan *graphics adapter* yang dibagikan secara otomatis sesuai dengan konfigurasi dari KVM.

Tugas Akhir ini menggunakan Hypervisor KVM sebagai alat virtualisasi di tiap perangkat keras karena merupakan Hypervisor yang kompatibel dengan CentOS dan paling mudah diaplikasikan pada infrastruktur komptuasi awan, sehingga mengurangi resiko *crash* dalam infrastruktur komputasi awan.

2.7. Libvirt

Libvirt [13] adalah sebuah toolkit berbasis *open-source* dari RHEL yang mempunyai kapabilitas untuk berinteraksi dengan lingkungan virtualisasi. Libvirt mempunyai API, daemon, dan alat yang digunakan untuk memanajemen platform virtualisasi. API dari Libvirt digunakan sebagai orkestrasi dari layer Hypervisor dalam pengembangan sebuah lingkungan komputasi awan. Fungsi utama dari Libvirt adalah sebagai penghubung antar perangkat keras dengan lingkungan virtualisasi di dalamnya. Pada Gambar 2.6 merupakan cakupan Hypervisor dan API yang didukung Libvirt. Hypervisor yang didukung diantaranya adalah KVM, LXC, OpenVZ, UML, Xen, ESX, dan lain-lain serta API yang didukung diantaranya adalah virsh, virt-manager, virt-factory dan oVirt.

Pada Tugas Akhir ini Libvirt digunakan untuk mengelola virtualisasi yang ada pada perangkat komputasi.



Gambar 2.6. Libvirt API dan Hypervisor [22]

2.8. XML

XML (*Extensible Markup Language*) [14] merupakan bahasa marka untuk sebuah dokumen yang mengandung informasi terstruktur. Informasi terstruktur berisi kata, gambar dan lainnya dan beberapa indikasi dari peran apa yang konten mainkan, sebagai contoh konten pada bagian judul berbeda dari konten yang berada pada catatan kaki. Hampir setiap dokumen mempunyai struktur. Bahasa marka merupakan salah satu mekanisme untuk mengidentifikasi struktur dalam sebuah dokumen.

XML diciptakan agar dokumen yang mengandung struktur dapat digunakan melalui jaringan web, karena alternatif lain seperti HTML dan SGML tidak praktis untuk tujuan semacam ini. XML merupakan format yang sederhana dan fleksibel yang diturunkan dari SGML.

Dalam Tugas Akhir ini XML digunakan untuk media pengiriman data yang berasal dari *database* dari setiap VM menuju pusat manajemen *server*.

2.9. PHP

PHP (*Personal Home Page*) [15] merupakan sebuah bahasa pemrograman yang berjalan pada sebuah *webserver* dan digunakan untuk pengembangan situs web. Selain itu, PHP dapat digunakan dengan tujuan umum seperti membuat sebuah berkas XML, dan melakukan pengolahan *database* seperti menyeleksi, menambah, menghapus, atau memperbaharui data yang ada di dalam *database*. PHP dapat dikombinasikan dengan kode HTML (*Hypertext Markup Language*) untuk menghasilkan konten situs web yang dinamis.

PHP merupakan sebuah bahasa *server-side scripting*, cara kerja PHP adalah ketika terdapat permintaan (*request*) oleh klien dengan mengakses alamat URL pada *web browser*, *webserver* akan mengidentifikasi terlebih dahulu apakah laman yang diminta klien mengandung kode PHP. Jika terdapat kode PHP dalam laman

tersebut maka *webserver akan* mengolah kode PHP tersebut dan melakukan proses logika di dalamnya, kode PHP tersebut tidak diberikan kepada klien yang meminta melainkan tetap disimpan dalam sisi *webserver*, sehingga klien hanya menerima kode HTML biasa.

Pada Tugas Akhir ini, bahasa pemrograman PHP digunakan sebagai pembuatan portal manajemen restoran yang memanfaatkan API dari CloudStack untuk melayani *tenant*.

2.10. Webservice

Webservice [16] merupakan sistem perangkat lunak yang didesain untuk mendukung interoperabilitas antar mesin dalam sebuah jaringan. *Webservice* memiliki antarmuka yang diuraikan dalam format yang dapat diproses oleh mesin (khususnya WSDL). Sistem lain dapat berinteraksi dengan *webservice* dengan cara yang ditentukan dengan pesan SOAP, biasanya disampaikan menggunakan protokol HTTP dengan serialisasi XML dalam kaitannya dengan standar web lainnya.

2.11. Sistem *Monitoring*

Sistem *monitoring* [17] merupakan sebuah proses dalam sistem terdistribusi untuk mengumpulkan dan menampilkan data performansi secara *real-time* untuk komputer lokal ataupun komputer pada jaringan tergantung kriteria yang ditentukan. Sistem monitor dapat mengawasi aktivitas *server* dan meringkas performansinya.

Pada Tugas Akhir ini, sistem monitor diaplikasikan untuk mengawasi aplikasi pada VM setiap *tenant* dan mengawasi kinerja dan kapasitas dari infrastruktur komputasi awan.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi penjelasan mengenai analisis dan perancangan perangkat lunak yang dikembangkan. Perancangan merupakan bagian penting dari pengembangan perangkat lunak karena merupakan perencanaan perangkat lunak secara teknis. Adapun hal-hal yang dibahas dalam bab ini adalah deskripsi umum perangkat lunak, arsitektur perangkat lunak, diagram kasus penggunaan, perancangan *database*, diagram alir, dan desain antarmuka perangkat lunak.

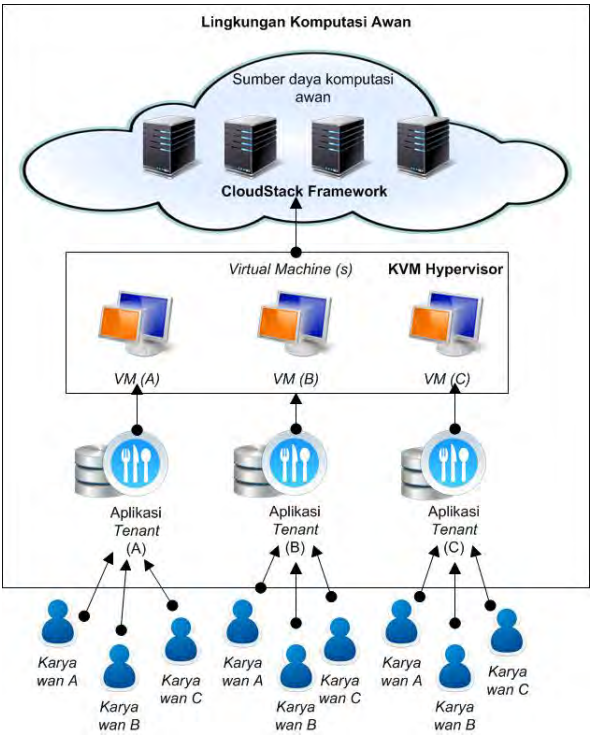
3.1. Deskripsi Umum

Pada Tugas Akhir ini dibangun sistem penyedia layanan aplikasi secara *multi-tenant* di atas infrastruktur komputasi awan. Sistem ini mengadopsi tingkatan *level multi-tenancy* pada infrastruktur dan platform, di mana *tenant* berbagi sumber daya komputasi dan platform aplikasi manajemen restoran dengan menggunakan teknologi virtualisasi.

Sistem ini menggunakan aplikasi manajemen restoran sebagai contoh aplikasi. Dalam kasus ini, *tenant* pada sistem ini merupakan pemilik restoran, setiap pemilik restoran mempunyai karyawan yang dapat mengakses aplikasi tersebut. Setiap aplikasi milik *tenant* ditempatkan pada sebuah VM yang berbeda antar *tenant*. VM tersebut dikumpulkan dalam satu layanan awan yang terdiri dari banyak sumber daya komputasi. Gambar 3.1 merupakan *multitenancy* dalam sistem.

Keseluruhan sistem ini terbagi menjadi dua bagian, pertama yaitu sebuah *webserver* portal layanan aplikasi dan kedua yaitu infrastruktur komputasi awan. Portal layanan aplikasi berfungsi sebagai media interaksi *tenant*. *Tenant* dapat melakukan permintaan sebuah aplikasi dengan mengakses portal ini, yang selanjutnya portal akan mengirimkan data tersebut menuju komputasi awan. Sedangkan infrastruktur komputasi awan

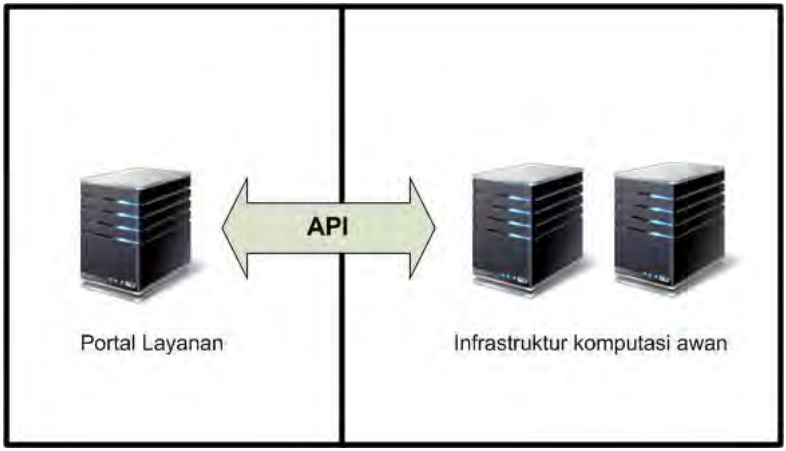
berfungsi sebagai penyedia lingkungan virtualisasi aplikasi milik *tenant*.



Gambar 3.1. *Multitenancy* dalam Sistem

Sistem ini menggunakan empat buah perangkat komputer, satu buah perangkat komputer digunakan sebagai *webserver* portal layanan, dan tiga buah perangkat komputer lain digunakan sebagai infrastruktur komputasi awan. Infrastruktur komputasi awan menggunakan kerangka kerja CloudStack dengan Hypervisor KVM yang bertanggung jawab dalam virtualisasi VM. Perangkat komputasi awan terdiri dari dua buah perangkat komputer, satu buah perangkat komputer berfungsi ganda sebagai MS

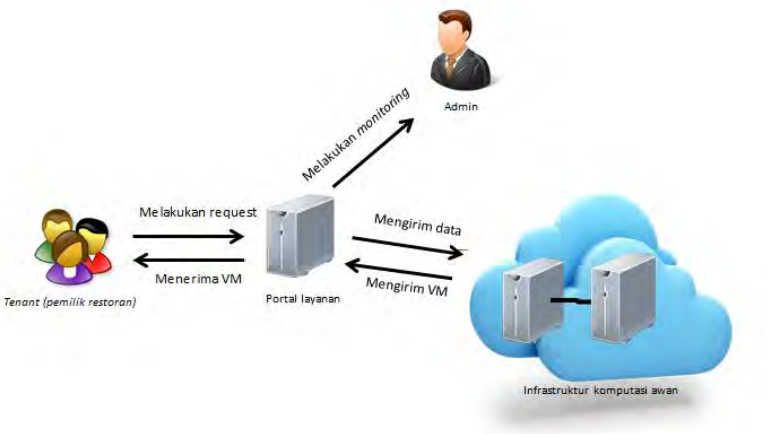
(*Management Server*), *Host* VM, dan penyimpanan data VM, dua perangkat lain dikhususkan untuk *Host* VM. Gambar 3.2 merupakan cara sistem dapat saling berkomunikasi.



Gambar 3.2. Komunikasi Sistem

Cara kerja sistem ini yaitu, sistem mengambil data *tenant* yang melakukan *request* layanan aplikasi manajemen restoran melalui portal yang telah disediakan. Kemudian, data tersebut dikirim melalui portal menuju perangkat komputasi awan dan diolah oleh sistem pada perangkat manajemen *server*. Selanjutnya, perangkat komputasi awan memproses data *tenant* tersebut dan membuat akun serta menciptakan VM berdasarkan *template* aplikasi manajemen restoran yang telah ditentukan sebelumnya. VM yang telah diciptakan dan dikonfigurasikan akan dikembalikan kepada *tenant* melalui kustom portal sehingga *tenant* dapat menggunakan aplikasi manajemen restoran tersebut. Setiap aplikasi manajemen restoran yang diberikan kepada *tenant* melalui portal memiliki alamat unik serta berada pada sebuah VM yang berbeda sehingga tidak ada *tenant* yang mendapatkan aplikasi manajemen restoran yang sama dengan data yang sama pula, *tenant* hanya mendapatkan struktur aplikasi dan *database* yang sama.

Di sisi lain, sistem dapat melakukan kontrol dan *monitoring* aplikasi manajemen restoran setiap *tenant* melalui portal admin, sistem ini bekerja dengan cara mengambil dan mengumpulkan data dari *database* setiap aplikasi manajemen restoran agar dapat dipantau secara *real-time*. Data setiap aplikasi manajemen restoran dikumpulkan dengan metode *pull* pada portal admin. Sistem juga dapat melakukan *monitoring* kapasitas dan kinerja infrastruktur awan. Gambar 3.3 merupakan gambaran cara kerja sistem secara keseluruhan.



Gambar 3.3. Diagram Cara Kerja Sistem

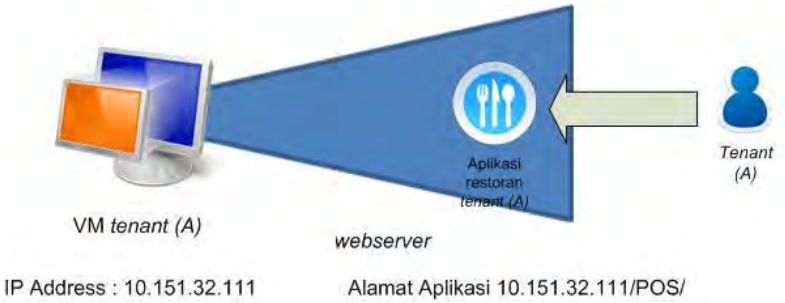
3.1.1. Aplikasi Manajemen Restoran pada *Virtual Machine*

Sistem ini menggunakan aplikasi manajemen restoran[12] *open-source* sebagai contoh layanan yang diberikan. Aplikasi manajemen restoran yang akan digunakan mencakup fungsi dasar

[12] PHPPOS. www.phppointofsale.com

seperti pembayaran, pengaturan karyawan, pengaturan stok, menampilkan laporan penjualan, dan lain sebagainya.

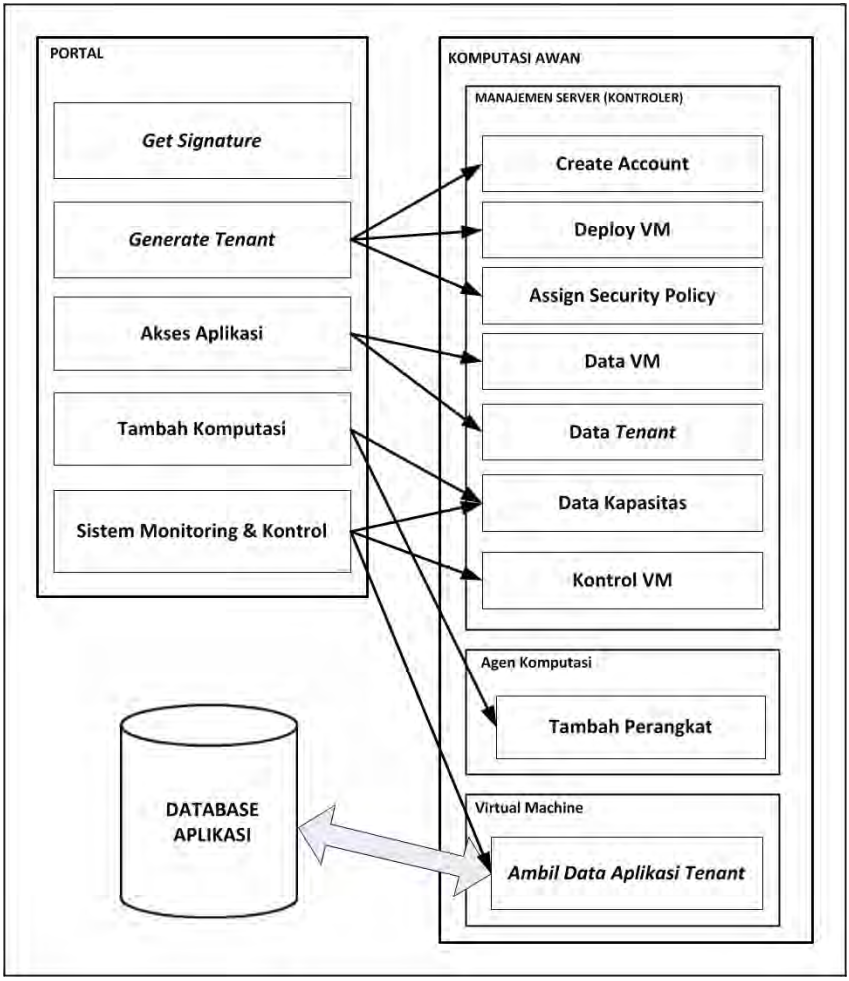
Aplikasi manajemen restoran ini akan ditempatkan pada *webserver* setiap VM milik *tenant*. *Webserver* ini secara otomatis dijalankan ketika sebuah VM diciptakan, kemudian sistem mengembalikan tautan berupa alamat menuju aplikasi yang sudah dibenamkan pada *webserver*, sehingga setiap *tenant* akan mengakses aplikasi sesuai alamat IP yang diberikan oleh sistem. Gambar 3.4. merupakan aplikasi manajemen restoran yang ditempatkan pada *webserver* sebuah VM.



Gambar 3.4. Aplikasi Manajemen Restoran pada VM

3.2. Arsitektur Umum Sistem

Arsitektur umum perangkat lunak dalam Tugas Akhir ini dapat dilihat pada Gambar 3.5. Arsitektur perangkat lunak terdiri dari portal dan infrastruktur komputasi awan. Portal merupakan sistem yang menyediakan layanan aplikasi restoran kepada *tenant* dan digunakan sebagai wadah untuk menampung semua fungsi dan logika dari perangkat lunak.



Gambar 3.5. Arsitektur Sistem

Arsitektur pada bagian infrastruktur komputasi awan berperan sebagai penyedia proses dan lingkungan virtualisasi wadah bagi VM yang digunakan untuk melayani permintaan

tenant. Pada bagian infrastruktur awan terdapat VM yang merupakan sub-bagian dari infrastruktur komputasi awan. Adapun penjelasan untuk modul-modul pada portal dan VM adalah sebagai berikut.

1. *Get Signature*
   Fungsi *Get Signature* digunakan untuk melakukan *hashing* pada API yang akan dipanggil pada portal. *Get signature* berfungsi untuk memberi kepercayaan pada portal penyedia layanan aplikasi manajemen restoran agar dapat menggunakan fungsi yang ditawarkan oleh infrastruktur komputasi awan.
2. *Generate Tenant*
   Fungsi *Generate Tenant* digunakan *tenant* untuk melakukan permintaan aplikasi manajemen restoran. Fungsi ini akan diproses pada manajemen *server*. Terdapat tiga bagian proses pada fungsi ini, yaitu *Create Account*, digunakan untuk menciptakan sebuah akun milik *Tenant*, *Deploy* VM, digunakan sebagai proses penciptaan sebuah VM berdasarkan data akun *tenant*, dan *Assign Security Policy*, digunakan sebagai konfigurasi agar VM *tenant* yang telah tercipta dapat diakses oleh jaringan luar.
3. Akses Aplikasi
   Fungsi Akses Aplikasi dipanggil ketika *tenant* mengakses aplikasi manajemen restoran yang tercipta berdasarkan ID *tenant* yang telah terdaftar. Terdapat dua proses pada fungsi ini, yaitu *Data Tenant*, digunakan sebagai pengecekan data *tenant* yang ingin mengakses aplikasi, dan *Data* VM digunakan sebagai pencocokan antara data *tenant* dengan VM milik *tenant* yang telah dibuat.
4. Sistem *Monitoring* & Kontrol
   Fungsi ini digunakan untuk memberikan informasi terkini dari kapasitas dan kinerja komputasi awan dan memberikan informasi dari aplikasi manajemen restoran yang digunakan tiap *tenant*. Fungsi ini juga dapat mengontrol aplikasi setiap *tenant*. Terdapat tiga proses pada fungsi ini, dua berasal dari manajemen *server*, yaitu Data kapasitas dan Kontrol VM, dan

satu berasal dari setiap VM milik *tenant* yaitu proses ambil data komputasi.

5. Tambah Komputasi
   Fungsi Tambah Komputasi terdiri dari beberapa proses, yaitu data kapasitas yang berasal dari manajemen *server*, serta proses tambah perangkat pada perangkat komputasi yang ingin ditambah.

3.3. Perancangan Diagram Kasus Penggunaan

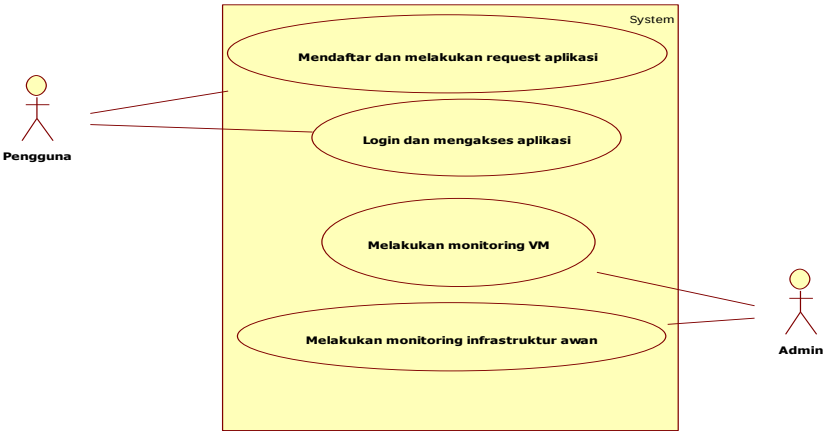
Diagram kasus penggunaan menggambarkan peran aktor yang terlibat dalam fungsionalitas perangkat lunak yang dibangun. Adapun perancangan diagram kasus penggunaan perangkat lunak dalam Tugas Akhir ini dapat dilihat pada Gambar 3.6.

Pada Tugas Akhir ini, terdapat dua aktor yang berperan dalam sistem ini, *Tenant* dan Admin. Aktor *Tenant* dapat menjalankan fitur yang ada pada perangkat lunak ini, yaitu sebagai berikut.

1. Mendaftar dan melakukan *request* aplikasi
   Tenant dapat melakukan pendaftaran sekaligus melakukan permintaan sebuah VM aplikasi manajemen restoran.
2. *Login* dan mengakses aplikasi
   Tenant dapat melakukan *login* dan mengakses ke halaman milik *tenant* yang berisi link menuju aplikasi yang telah tercipta.

Terdapat aktor lain dalam sistem ini yaitu Admin. Hal yang dapat dilakukan oleh Admin adalah sebagai berikut.

1. Melakukan *monitoring* dan kontrol aplikasi
   Admin dapat melakukan *monitoring* setiap aplikasi manajemen restoran yang digunakan oleh *tenant* pada laman admin untuk melakukan pengawasan dan pengontrolan pada aplikasi yang berjalan
2. Melakukan *monitoring* infrastruktur awan
   Admin dapat melakukan *monitoring* kapasitas dari infrastruktur awan.



Gambar 3.6. Diagram Kasus Penggunaan

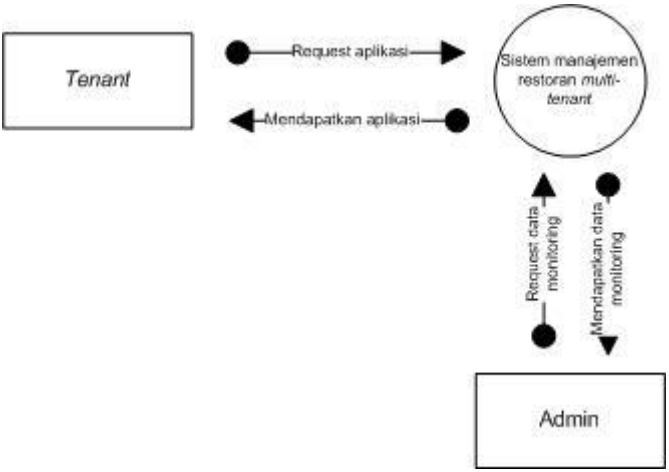
3.4. Perancangan Komputasi Awan

Komputasi awan dengan kerangka kerja CloudStack yang dibangun menggunakan tiga buah komputer, satu komputer berperan ganda sebagai manajemen *server* dan agen komputasi dan dua komputer lainnya hanya berperan sebagai agen komputasi. Manajemen *server* berfungsi sebagai pusat kontrol komputasi awan dan menyediakan fitur API, agen komputasi berfungsi sebagai tempat VM berada, prosesor agen komputasi harus mendukung *Hypervisor* KVM untuk proses virtualisasi. Jaringan infrastruktur komputasi awan digambarkan pada Gambar 3.7.

Gambar 3.7. Gambaran Arsitektur Komputasi Awan

3.5. Perancangan Diagram Alir Data *Level* 0

Diagram alir data *level* 0 adalah gambaran fungsionalitas perangkat lunak dan faktor eksternal yang terlibat. Diagram alir data *level* 0 dapat dilihat pada Gambar 3.8. Berdasarkan gambar tersebut, alir kerja sistem ini diawali dengan *tenant* melakukan *request* sebuah aplikasi manajemen restoran pada sistem, sistem mengembalikan sebuah aplikasi manajemen restoran yang siap pakai pada *tenant*, di sisi lain admin dapat melakukan permintaan *monitoring* pada sistem untuk mengawasi aplikasi setiap *tenant* dan infrastruktur komputasi awan. Admin juga mempunyai hak kontrol dalam mengatur aplikasi milik *tenant*.



Gambar 3.8. Diagram Alir Data *Level* 0

3.6. Perancangan Diagram Alir Data

Pada Tugas Akhir ini, alir setiap proses di dalam perangkat lunak digambarkan dengan diagram alir data untuk memudahkan pemahaman secara garis besar proses yang ada pada perangkat lunak ini. Diagram alir data yang terdapat pada perangkat lunak ini terdiri dari diagram alir data *generate tenant*, akses aplikasi, ambil data *tenant*, *monitoring* aplikasi, *monitoring* komputasi, dan tambah kapasitas.

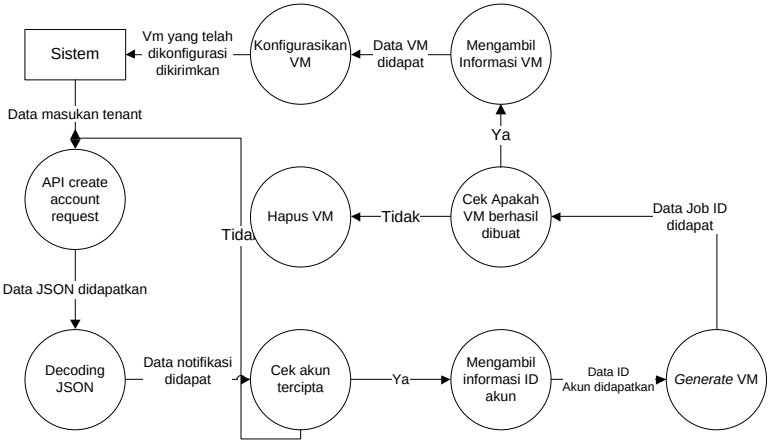
3.6.1. Diagram Alir Data Melakukan *Generate Tenant*

Proses *Generate tenant* berfungsi untuk menciptakan sebuah VM yang diminta setiap *tenant*. Proses ini menciptakan VM berdasarkan masukan berupa informasi ID *tenant*. Setelah VM tercipta akan dilakukan perijinan sekuritas dengan merequest permintaan kepada infrastruktur awan, perijinan sekuritas dilakukan agar sebuah VM *tenant* dapat diakses oleh jaringan

luar. VM yang tercipta menggunakan *template* aplikasi manajemen restoran yang sudah ditentukan sebelumnya.

Langkah pertama dalam proses ini berupa sistem meminta data masukan informasi *tenant*, kemudian sistem mengolah data tersebut dan memproses pembuatan akun sesuai dengan informasi *tenant*. Apabila akun berhasil dibuat. Data informasi berupa ID *tenant* yang dihasilkan dari proses pembuatan akun tadi digunakan sebagai salah satu masukan untuk proses menciptakan sebuah VM. Data ID *tenant* dibutuhkan untuk memberi tanda bahwa VM yang dibuat merupakan milik *tenant* tertentu.

Proses menciptakan sebuah VM membutuhkan data masukan berupa ID *tenant* yang melakukan *request*, ID *template* aplikasi manajemen restoran, dan jenis komputasi VM. Pada sistem ini, proses menciptakan sebuah VM secara otomatis dijalankan ketika sistem berhasil menjalankan proses pembuatan akun.



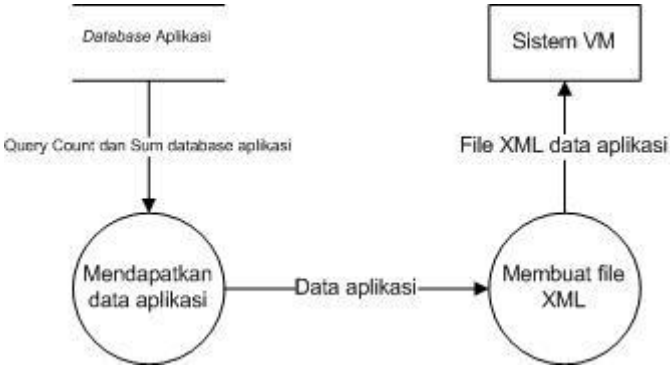
Gambar 3.9. Diagram Alir Data Melakukan *Generate Tenant*

Setelah sebuah VM milik *tenant* tercipta, VM tersebut akan dikonfigurasikan terlebih dahulu agar dapat diakses oleh

jaringan luar, konfigurasi tersebut membutuhkan sebuah API dari infrastruktur komputasi awan. Setelah konfigurasi VM selesai dilakukan, sistem mengambil informasi ID VM yang digunakan untuk proses selanjutnya. Gambar diagram alir data *Generate tenant* dapat dilihat pada Gambar 3.9.

3.6.2. Diagram Alir Data Mengambil Data Aplikasi *Tenant*

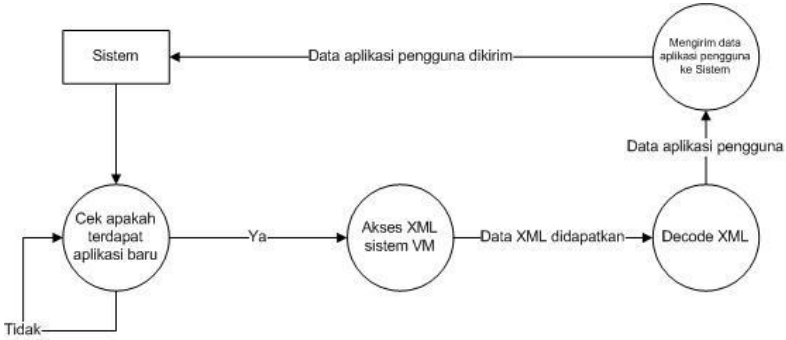
Tahap ini merupakan proses pengambilan data aplikasi *tenant* dari *database* aplikasi manajemen restoran menuju sistem VM. Langkah pertama, *database* yang menyimpan informasi aplikasi *tenant* akan dilakukan *query count* dan *sum* serta informasi *bandwidth* didapatkan dengan mengambil data dari *command line* Linux. Selanjutnya, data aplikasi *tenant* yang telah dilakukan *query* ke dalam berkas berekstensi XML. Berkas tersebut diletakkan ke dalam sistem VM pada *web service* untuk dapat dikonsumsi oleh sistem *monitoring* pada portal. Gambar diagram alir data ambil data *tenant* ditunjukkan oleh Gambar 3.10.



Gambar 3.10. Diagram Alir Data Mengambil Data Aplikasi *Tenant*

3.6.3. Diagram Alir Data Melakukan *Monitoring* Aplikasi *Tenant*

Monitoring aplikasi milik *tenant* berfungsi untuk mengumpulkan data setiap aplikasi *tenant* pada masing-masing VM dan menampilkan pada halaman sistem *monitoring*. Proses ini bekerja dengan mengonsumsi berkas XML pada setiap *tenant*. Gambar 3.11 merupakan diagram alir data untuk mengambil dan mengumpulkan data aplikasi pada setiap VM *tenant*. Pertama sistem melakukan pengecekan secara *real-time*, jika ada aplikasi yang terdapat pada lingkungan infrastruktur awan maka sistem akan mengakses berkas XML aplikasi tersebut. XML tersebut kemudian di *decode* untuk diambil data aplikasi. Kemudian data yang telah didapat digabungkan dengan data milik aplikasi lain dan ditampilkan oleh sistem pada halaman sistem *monitoring*.

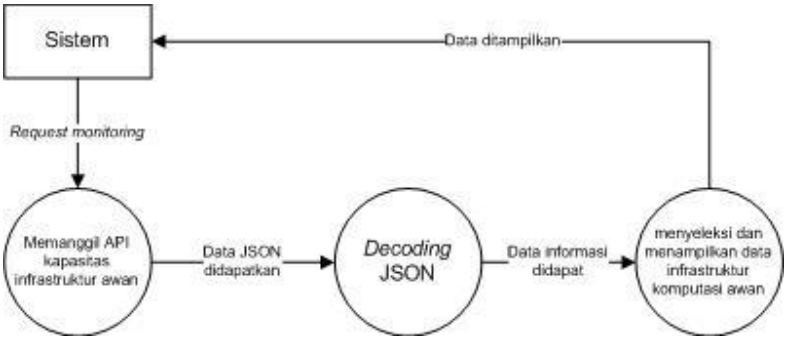


Gambar 3.11. Diagram Alir Data Melakukan *Monitoring* Aplikasi

3.6.4. Diagram Alir Data Melakukan *Monitoring* Infrastruktur Komputasi Awan

Gambar 3.12 merupakan diagram alir data dalam melakukan monitoring infrastruktur awan. Tahap ini merupakan proses pengambilan data kapasitas dan *tenant* pada infrastruktur komputasi awan. Pertama-tama, sistem akan memanggil API yang

digunakan untuk mengecek kapasitas dan *tenant* infrastruktur awan. API dikirimkan dengan HTTP *request* dari portal menuju infastruktur komputasi awan. Hasil dari request tersebut berupa data berformat JSON. JSON yang diterima oleh sistem dilakukan *decoding* data dan hasilnya berupa informasi infrastruktur komputasi awan. Data tersebut kemudian ditampilkan pada halaman sistem *monitoring*.



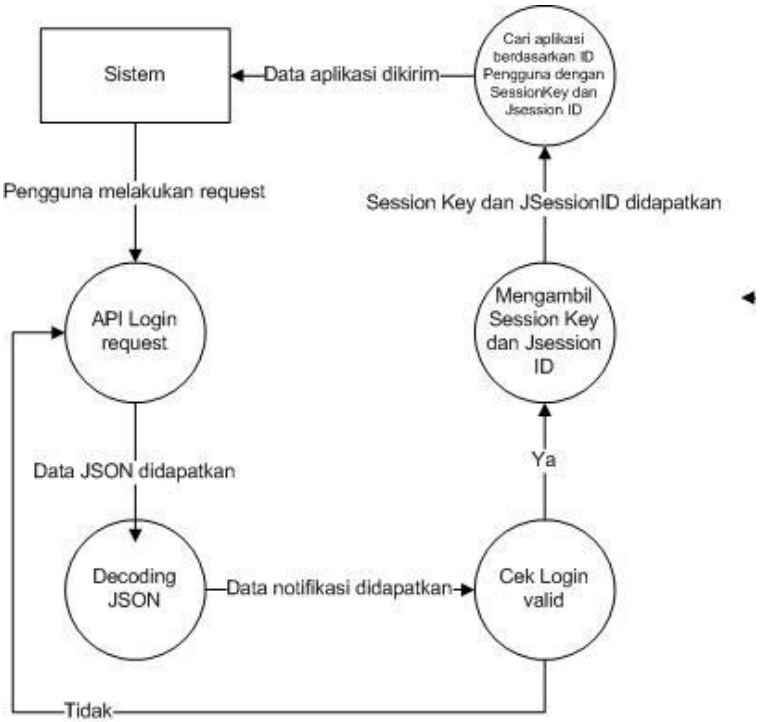
Gambar 3.12. Diagram Alir Data Melakukan *Monitoring* Infrastruktur Komputasi Awan

3.6.5. Diagram Alir Data Mengakses Aplikasi

Akses aplikasi digunakan *tenant* untuk mengakses aplikasi manajemen restoran yang sudah diciptakan melalui proses *generate* VM. Fungsi ini membutuhkan masukan berupa *username* dan *password* milik *tenant*.

Gambar 3.13 merupakan diagram alir data akses aplikasi. Saat portal menerima *request* berupa akses aplikasi dari *tenant*, maka sistem akan memproses dengan API *Login* dari CloudStack dengan parameter yang *tenant* masukkan, API dikirimkan dengan HTTP *request* dari portal menuju infrastruktur komputasi awan. Infrastruktur awan segera merespon dengan mengembalikan data berupa JSON. JSON yang diterima oleh portal akan dilakukan

decoding data yang nantinya hasil *decode* tersebut berisi sebuah objek yang menentukan apakah proses *login* berhasil atau tidak. Sistem akan mengecek data notifikasi pada JSON tersebut. Jika pada objek terdapat notifikasi yang menyatakan *login* berhasil, maka sistem akan menginformasikan kepada *tenant* bahwa *login* berhasil.



Gambar 3.13. Diagram Alir Data Mengakses Aplikasi

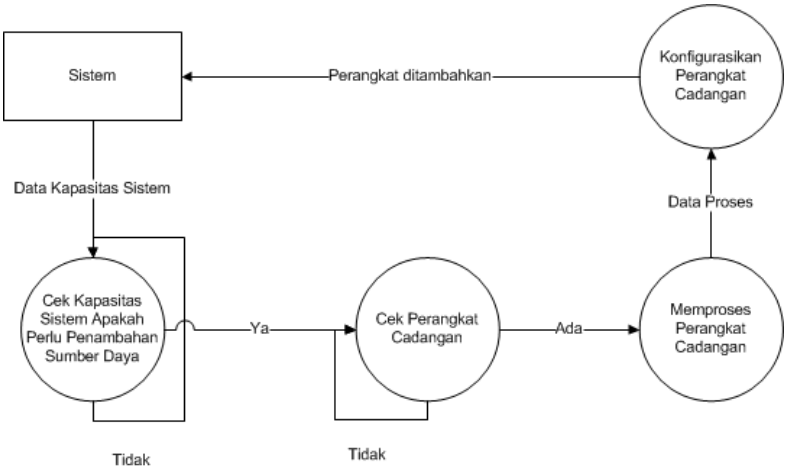
Selanjutnya sistem akan melakukan *redirect* menuju halaman *dashboard*. Pada halaman *dashboard* sistem akan melakukan pencarian aplikasi yang sesuai dengan ID akun *tenant*, setelah pencarian berhasil sistem menampilkan akses alamat URL

pada portal layanan sesuai dengan ID akun dari *tenant* yang melakukan *login.*

3.6.6. Diagram Alir Data Menambah Komputasi

Tambah komputasi berfungsi sebagai penyedia sumber daya cadangan pada infrastruktur komputasi awan ketika hasil monitoring infrastruktur awan mencapai batas maksimal yang telah ditentukan. Fungsi ini membutuhkan masukan berupa nilai CPU dan RAM yang digunakan. Apabila melebih persentase yang telah ditentukan, maka sistem secara otomatis akan melakukan penambahan perangkat *server* dengan perangkat yang telah ditentukan dan dikonfigurasikan sebelumnya.

Gambar 3.14 merupakan diagram alir data tambah komputasi. Sistem pada awalnya melakukan pengecekan beban kerja komputasi, apabila beban kerja melebihi batas yang telah ditentukan maka sistem akan mengecek apakah ada perangkat cadangan yang tersedia, apabila terdapat perangkat, sistem segera melakukan proses penambahan perangkat dan melakukan konfigurasi pada perangkat tersebut.



Gambar 3.14. Diagram Alir Menambah Komputasi

3.7. Rancangan Antarmuka

Pada Tugas Akhir ini sistem menggunakan situs web untuk menyediakan layanan bagi *tenant*. Dalam situs web ini disediakan form daftar untuk memfasilitasi *tenant* dalam mendaftar akun dan menciptakan VM. form *login* untuk menyediakan akses untuk *tenant* yang telah mendaftar, halaman *dashboard tenant* yang digunakan untuk menyediakan alamat aplikasi untuk *tenant* dan halaman admin yang digunakan admin untuk mengawasi aplikasi *tenant* dan infrastruktur awan.

3.7.1. Rancangan Antarmuka Halaman Utama

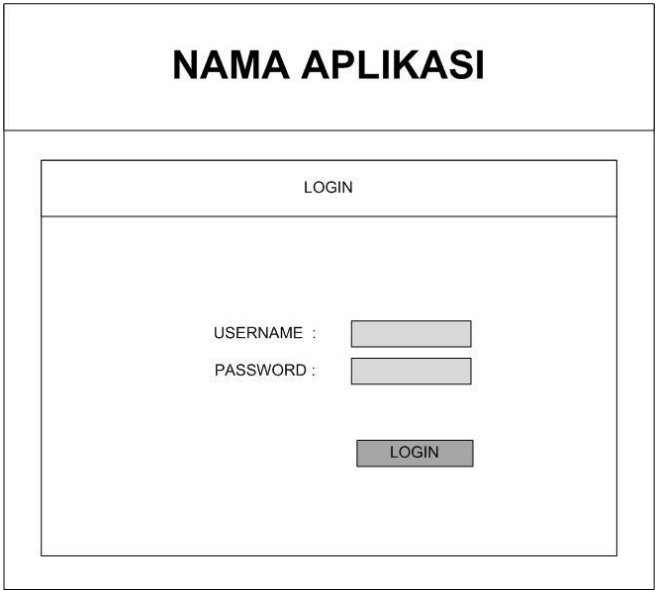
Rancangan antarmuka halaman utama dapat dilihat pada Gambar 3.15. Ketika *tenant* mengakses situs web tersebut akan terdapat sebuah form untuk mendaftar dan membuat VM. Terdapat lima *item textbox* dan satu tombol pada halaman ini, *item textbox* yaitu email, akun, nama, *username*, dan *password* yang harus diisi

Gambar 3.15. Rancangan Antarmuka Halaman Utama

agar *tenant* dapat mulai menggunakan layanan dan mengakses aplikasi manajemen restoran.

3.7.2. Rancangan Antarmuka *Login*

Gambar 3.16 merupakan rancangan antarmuka halaman *login* dari situs web. Setelah pengguna mengisi data *tenant* pada halaman buat akun, *tenant* akan diarahkan menuju halaman *login*, terdapat dua *item textbox* yaitu *username* dan *password* serta terdapat satu tombol *login*. *tenant* dapat melakukan *login* dengan *username* dan *password* yang telah *tenant* daftarkan pada halaman registrasi.

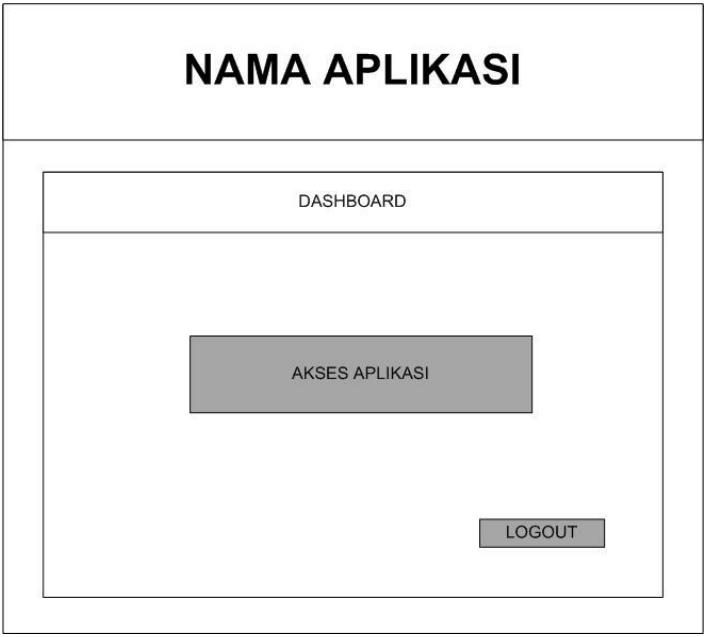


Gambar 3.16. Rancangan Antarmuka Halaman *Login*

3.7.3. Rancangan Antarmuka *Dashboard*

Gambar 3.17. merupakan rancangan antarmuka *dashboard*, *tenant* yang telah melakukan *login* akan diarahkan ke

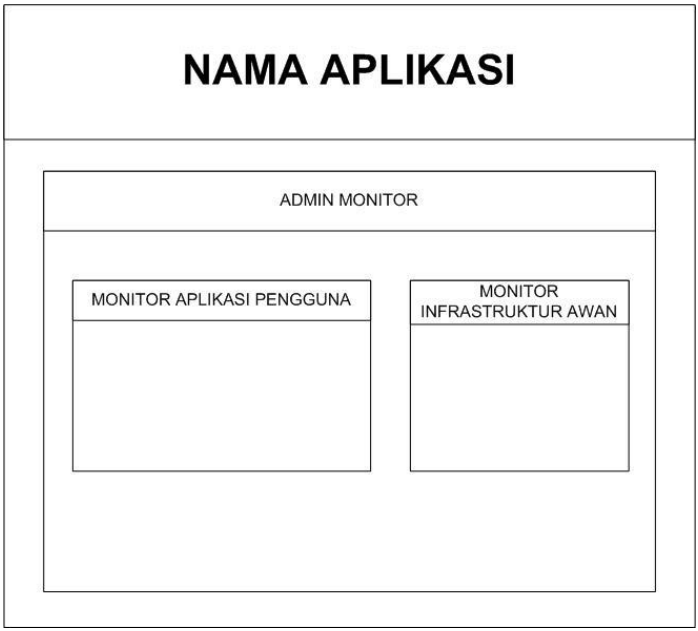
halaman *dashboard tenant*. Terdapat tombol akses aplikasi dan logout, tombol akses aplikasi digunakan untuk mengakses aplikasi manajemen restoran *tenant*. Tombol logout berfungsi untuk keluar dari halaman *dashboard*.



Gambar 3.17. Rancangan Antarmuka Halaman *Dashboard*

3.7.4. Rancangan Antarmuka Admin

Gambar 3.18. merupakan rancangan antarmuka admin, Pada halaman ini, semua informasi sistem ditampilkan, terdapat tabel monitor aplikasi *tenant* yang digunakan untuk mengawasi pergerakan aplikasi setiap *tenant* dan tabel monitor infrastruktur awan yang digunakan untuk melakukan pengawasan dalam kapasitas infrastruktur awan.



Gambar 3.18. Rancangan Antarmuka Halaman Admin

BAB IV
IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi sistem yang dikembangkan. Implementasi sistem merupakan bentuk realisasi dari perancangan sistem yang telah dijelaskan pada bab sebelumnya. Adapun hal-hal yang dibahas dalam bab ini adalah lingkungan implementasi sistem, implementasi sistem disertai dengan *pseudocode*, dan *screenshot* hasil implementasi sistem.

4.1. Lingkungan Implementasi

Pada Tugas Akhir ini, lingkungan untuk melakukan implementasi sistem penyedia layanan aplikasi secara *multi-tenant* terdiri dari perangkat keras dan perangkat lunak. Adapun penjelasan mengenai lingkungan implementasi adalah sebagai berikut.

4.1.1. Lingkungan Implementasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk pembangunan sistem Tugas Akhir ini menggunakan empat buah komputer dan satu buah *switch*. Spesifikasi dari perangkat yang digunakan adalah sebagai berikut.

- Dua komputer merek Acer
  - Intel (R) Core (TM) i3-2120 CPU @ 3.30GHz.
  - 4 GB DDR3.
  - 500GB WDC WD5000AAKX-2, dan
  - Realtek RTL8111/8168B PCI Express Gigabit Ethernet.
- Satu komputer merek HP
  - Intel (R) Core (TM) i3-3220 CPU @ 3.30Ghz.
  - 2 GB DDR3.
  - 500GB WDC WD5000AAKX-6, dan
  - Realtek RTL8111/8168B PCI Express Gigabit Ethernet
- Satu komputer merek Lenovo
  - Windows 8 Pro 64-Bit.

  - Prosesor Intel® Core™ i3-3240 CPU @ 3.40GHz.
  - 4 GB DDR 3.
- Satu *switch* merek D-Link.

4.1.2. Lingkungan Implementasi Perangkat Lunak

Lingkungan implementasi sistem yang digunakan untuk menunjang pengembangan sistem ini terdiri atas sebuah sistem operasi, beberapa kakas, dan beberapa pustaka. Adapun lingkungan implementasi sistem tersebut adalah sebagai berikut sebagai berikut.

- Microsoft Windows 8 Ultimate 64-Bit sebagai sistem operasi portal layanan.
- CentOS minimal versi 6.5 64-Bit sebagai sistem operasi perangkat infrastruktur awan.
- Ubuntu 12.04 versi 32-Bit digunakan sebagai sistem operasi VM.
- KVM sebagai Hypervisor yang berjalan pada setiap perangkat keras infrastruktur komputasi awan.
- Libvirt sebagai pengelola VM pada setiap Hypervisor dalam infrastruktur komputasi awan.
- MySQL sebagai RDBMS (*Relational Database Management System*) untuk menyimpan data informasi aplikasi *tenant*.
- Apache xampp sebagai platform *web server* untuk menjalankan portal sistem.
- Apache CloudStack 4.3 sebagai kerangka kerja infrastruktur komputasi awan.

Tugas Akhir ini menggunakan empat buah komputer. Dua buah komputer merek Acer dan HP bertugas sebagai perangkat yang digunakan sebagai komputasi awan (*backend* sistem), satu buah komputer merek Lenovo bertugas sebagai *webserver* sistem penyedia layanan aplikasi secara *multi-tenant*.

Pada infrastruktur komputasi awan, satu buah komputer Acer berperan ganda sebagai pusat manajemen komputasi awan

(*Management Server*) dan sebagai agen komputasi, dan satu buah komputer Acer dan HP lainnya berperan hanya sebagai agen komputasi. Perangkat sistem berjalan pada jaringan yang terintegrasi dengan laboratorium NCC. Arsitektur perangkat dan jaringan digambarkan pada Gambar 4.1.

Gambar 4.1. Arsitektur Jaringan Sistem

Tabel 4.1. Alokasi IP dan Fungsi dari Perangkat Sistem

Kode	Jenis Komputer	Alamat IP	Fungsi Perangkat	Nama Host
KA 1	Acer	10.151.32.51	Manajemen *Server* dan Agen Komputasi	Srvr1.cloud.priv
KA2	Acer	10.151.32.52	Agen Komputasi	Host1.cloud.priv
KA3	HP	10.151.32.53	Agen Komputasi	Host2.cloud.priv
Portal	Lenovo	10.151.32.54	Portal layanan (*frontend*)	Portal

Pada Tabel 4.1 merupakan tabel alokasi IP dan fungsi dari perangkat yang terdapat pada kesuluruhan sistem.

4.2. Implementasi Perangkat Lunak

Implementasi pada perangkat lunak terbagi menjadi implementasi *server* sebagai wadah untuk menampung fungsi dan logika dari perangkat lunak dan implementasi situs web. Setiap bagian implementasi akan dijelaskan pada subbab berikut ini.

4.2.1. Implementasi *Generate Tenant*

Implementasi *Generate Tenant* merupakan bentuk implementasi dari diagram alir *Generate Tenant*. *Input* yang dibutuhkan dalam implementasi ini adalah data *tenant* berupa ID akun. *Output* yang dikeluarkan dalam implementasi ini adalah informasi VM yang telah diciptakan.

Adapun untuk *pseudocode* dari implementasi ini dapat dilihat pada Gambar 4.2.

Prosedur 1. *Generate Tenant*

Input: Data *tenant*

```
1  akun ⇦ account
2  email ⇦ email
3
4  firstname ⇦ firstname
5  lastname ⇦ lastname
6  username ⇦ username
7  password ⇦ password
8  Inisialisasi API create account dengan paramater akun,
9  email, firstname, lastname, username dan password.
10 HTTP request API create account
11 response ⇦ Decode JSON dari HTTP request API create
12 account
13      IF response.getnotif() == valid
14
15          ID akun ⇦ response.getID()
16          ID akun
17
```

Prosedur 1. *Generate Tenant*

Input: Data *tenant*

```
          Inisialisasi API DeployVirtualMachine dengan parameter Idakun
          HTTP request API deploy virtual machine
          Response ⇦ Decoding JSON dari HTTP request API deploy virtual machine

        IF response.getjobid() == berhasil
             infovm ⇦ response.getinfovm()
             konfigurasi VM berdasarkan data infoVM agar dapat diakses oleh jaringan luar
             tampilkan VM yang telah dikonfigurasi.
        ELSE
             Gagal
             infovm ⇦ response.getinfovm()
             HTTP request API destroyvm dengan paramater infovm
Tampilkan informasi datavm
     ELSE
        Akun gagal tercipta
```

Output: Informasi VM yang telah diciptakan

Gambar 4.2. Implementasi Generate *Tenant*

Penjelasan dari implementasi *Generate Tenant* adalah sebagai berikut.

1. Inisialisasi API *create account* request dengan parameter *tenant*.
2. HTTP request API *create account* pada infrastruktur komputasi awan.
3. *Decode* JSON yang diterima dari infrastruktur komputasi awan.
4. Cek notifikasi dari JSON, apabila berhasil, maka ambil ID *tenant* dari JSON.

5. Inisialisasi API *deploy virtual machine* dengan parameter ID akun *tenant*.
6. HTTP request *API deploy virtual machine* pada infrastruktur komputasi awan.
7. Data JSON didapat.
8. *Decode* JSON.
9. Cek informasi dari JSON, apabila berhasil, maka ambil informasi *virtual machine* yang telah tercipta, *redirect* menuju halaman *login*. Jika gagal, jalankan API *delete virtual machine* dengan parameter ID *virtual machine*.

4.2.2. Implementasi Akses Aplikasi

Implementasi *pseudocode* dari akses aplikasi dapat dilihat pada Gambar 4.3. *Input* yang diperlukan adalah *username* dan *password tenant*. Output yang keluar merupakan data akun *tenant* beserta akses aplikasi manajemen restoran milik *tenant*.

Prosedur 2. Akses Aplikasi

Input: username dan password

```
username ⇦ username
password ⇦ password
inisialisasi API login dengan parameter username dan
password
HTTP request API login
Response ⇦ Decoding JSON dari hasil request API
    IF response.notif() == valid
            sessionkey ⇦ response.getsession()
            jsessionid ⇦ getheader()
            listvirtualmachine berdasarkan ID pengguna
dengan session key dan jsessionid.
            redirect "dashboard"
            Tampilkan aplikasi.
       ELSE
              Login error
              Redirect login
```

Prosedur 2. Akses Aplikasi
Input: username dan password
Output: Data informasi pengguna dan data *virtual machine*

Gambar 4.3. Implementasi Akses Aplikasi

Penjelasan dari implementasi akses aplikasi adalah sebagai berikut.

1. Inisialisasi *login* API dengan parameter data *tenant*.
2. HTTP request *login* pada infrastruktur komputasi awan.
3. *Decode* JSON yang didapat.
4. Cek informasi *login* dari JSON, apabila data valid maka ambil session dan jsessionid yang digunakan untuk memanggil daftar aplikasi yang dimiliki *tenant*, *redirect* menuju halaman *dashboard*. Akses aplikasi ditampilkan.
5. Jika gagal. *Redirect* menuju halaman *login*.

4.2.3. Implementasi Ambil Data *Tenant*

Gambar 4.4 merupakan *pseudocode* bentuk implementasi dari diagram alir ambil data *tenant*. Adapun *input* untuk implementasi ini adalah *database* yang berisi informasi aplikasi pengguna dan informasi bandwitdh dari terminal Linux.

Penjelasan dari *pseudocode* implementasi alir ambil data pengguna adalah sebagai berikut.

1. Membuka koneksi ke *database*.
2. Mendapatkan *database* aplikasi milik *tenant*.
3. Mendapatkan *Bandwidth* dari terminal Linux.
4. Mendapatkan *IP Address* dari terminal Linux.
5. Melakukan deklarasi *query* ‘select SUM’ untuk menjumlah nilai baris dalam *database*.
6. Melakukan deklarasi *query* ‘count ROW’ untuk menjumlah baris dalam *database*.
7. Buat berkas XML yang berisi variabel *Bandwitdh*, *IP Address*, query ’select’ dan query ‘count’.

8. Menutup koneksi *database*.

Prosedur 3. Ambil Data Tenant	
Input: *database* aplikasi pengguna serta informasi IP dan Bandwidth dari terminal linux	
1 2 3 4 5 6 7 8	Membuka koneksi ke *database* Load *database* aplikasi pengguna Bandwidth ⇦ Bandwitdh IP Address ⇦ IP Address Query1 ⇦ select SUM (*) from table Query2 ⇦ count rows from table XML(Bandwidth,IPAddress,Query1,Query2) Menutup koneksi *database*
Output: Berkas XML berisi *database* informasi aplikasi pengguna	

Gambar 4.4. Implementasi Ambil Data Aplikasi *Tenant*

4.2.4. Implementasi *Monitoring* Infrastruktur Komputasi Awan

Gambar 4.5 merupakan *pseudocode* bentuk implementasi dari diagram alir *monitoring* infrastruktur komputasi awan. Proses ini menghasilkan keluaran berupa Informasi *Bandwidth* dan data kondisi infrastruktur komputasi awan. Penjelasan dari implementasi *monitoring* infrastruktur awan adalah sebagai berikut.

1. Inisialisasi API cek kapasitas dan *user* dari infrastruktur komputasi awan.
2. Inisialisasi API kontrol VM apabila dibutuhkan sebuah aksi untuk mengontrol VM milik *tenant*.
3. Decode JSON yang diterima setelah pemanggilan API.
4. Ambil dan seleksi data sesuai dengan yang ingin ditampilkan pada sistem.
5. Cek kondisi VM apakah aktif atau tidak.

6. Apabila VM aktif tampilkan tombol ‘stop’, jika VM mati tampilkan tombol ‘start’.

Prosedur 4. *Monitoring* Infrastruktur Komputasi Awan

Input: -

```
1  Inisialisasi API listCapacity
2  Inisialisasi API listUser
3  Inisialisasi API start dan stop VM
4  response ⇦ decode JSON dari API ListCapacity
5  response2 ⇦ decode JSON dari API ListUser
6  cek notifikasi dari response dan response2
7  seleksi data response dan response2 yang akan
8        ditampilkan.
9  Cek kondisi VM
10   If kondisi VM == stop
11        Enable tombol start
12   ELSE kondisi VM == start
13        Enable tombol stop
14
15
```

Output: Informasi Bandwidth dan data kondisi infrastruktur komputasi awan

Gambar 4.5. Implementasi *Monitoring* Infrastruktur Komputasi Awan

4.2.5. Implementasi *Monitoring* Aplikasi

Gambar 4.6 merupakan *pseudocode* bentuk implementasi dari diagram alir *Monitoring* aplikasi. Proses ini membutuhkan berkas XML dari aplikasi milik *tenant*.

Penjelasan dari implementasi *monitoring* aplikasi adalah sebagai berikut.

1. Inisialisasi API *virtual machine* dari infrastruktur komputasi awan.
2. Decode JSON yang diterima.
3. Ambil data jumlah *virtual machine* dari JSON.

4. Untuk setiap jumlah *virtual machine* dalam JSON, ambil informasi IP setiap *virtual machine* dari JSON.Kemudian Ambil XML dari setiap IP *Virtual Machine*, jika IP valid, maka masukkan dan tampilkan data bandwidth dan transaksi dari setiap *virtual machine*. Jika IP tidak valid, tampilkan pesan 'error'.

Prosedur 5. *Monitoring* Aplikasi

Input: Berkas XML

```
1  Inisialisasi API listVirtualMachine
2  response ⇦ decode JSON dari API
3  ListVirtualMachine
4  s ⇦ response.countVM()
5  FOR (i=0, i<s, i++)
6     IP ⇦ response.IP(i)
7     IF(XML(IP))
8         t ⇦ XML->transaksi
9         u ⇦ XML->bandwidth
10    ELSE
11        VIRTUAL MACHINE ERROR
12 Tampilkan t, u
```

Output: Informasi Bandwidth dan data aplikasi

Gambar 4.6. Implementasi *Monitoring* Aplikasi

4.2.6. Implementasi Tambah Komputasi

Pada Gambar 4.7 merupakan *pseudocode* bentuk implementasi dari diagram alir Tambah Komputasi. Proses ini membutuhkan data hasil monitoring kapasitas infrastruktur komputasi awan. Penjelasan dari implementasi Tambah Komputasi adalah sebagai berikut.

1. Set R dengan nilai persentase RAM yang digunakan.
2. Set C dengan nilai persentase CPU yang digunakan.
3. Cek kapasitas nilai tersebut, apabila R lebih dari 70% atau C lebih dari 50% maka, cek apakah perangkat cadangan ada,

4. Apabila ada, lakukan inisialisasi proses penambahan perangkat.
5. Decode JSON hasil dari respon penambahan perangkat.
6. Apabila berhasil konfigurasi perangkat.
7. Tampilkan data perangkat baru.

Prosedur 6. Tambah Komputasi

Input: Data Kapasitas

```
1  R ⇦ kapasitasRAM
2  C ⇦ kapasitasCPU
3
4  Cek if R >= 70% atau C >= 50%
5      if perangkat? == ada
6           Response ⇦ Inisialisasi API proses
7  tambah host.
8           Response ⇦ decode JSON
9           If Response == berhasil
10             Konfigurasikan perangkat
11 Tampilkan data perangkat ke sistem.
```

Output: Data perangkat baru

Gambar 4.7. Implementasi Tambah Komputasi

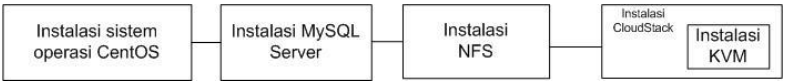
4.3. Implementasi Infrastruktur Komputasi Awan

Pada subbab ini akan dibahas mengenai implementasi infrastruktur komputasi awan menggunakan CloudStack. Implementasi dilakukan pada dua buah perangkat komputer ACER. Satu komputer digunakan sebagai manajemen *server* dan satu komputer digunakan sebagai agen komputasi.

4.3.1. Implementasi Perangkat Manajemen *Server*

Implementasi perangkat manajemen *server* dilakukan pada satu buah perangkat komputer dengan pemasangan beberapa modul untuk mendukung kinerja manajemen *server* pada komputasi awan. Pada Gambar 4.8 merupakan tahapan

implementasi manajemen *server* pada satu buah perangkat komputer. Pada tiap tahap tersebut diperlukan konfigurasi agar manajemen *server* dapat berjalan dengan baik. Konfigurasi tiap tahap dijelaskan pada Lampiran Implementasi infrastruktur komputasi awan.



Gambar 4.8. Tahap Implementasi Perangkat Manajemen *Server*

4.3.2. Implementasi Perangkat Agen Komputasi

Implementasi perangkat agen komputasi dilakukan pada satu buah perangkat komputer ACER dan HP dengan pemasangan beberapa modul untuk mendukung kinerja dari agen komputasi. Gambar 4.9. merupakan tahapan implementasi perangkat agen komputasi. Diperlukan konfigurasi agar perangkat agen komputasi dapat berjalan dengan baik pada infrastruktur awan. Konfigurasi tiap tahap dijelaskan pada Lampiran Implementasi infrastruktur komputasi awan.

Gambar 4.9. Tahap Implementasi Perangkat Agen Komputasi

4.4. Implementasi Antarmuka Sistem

Implementasi antarmuka sistem ini merupakan sebuah situs web yang terdiri dari halaman daftar, *login*, *dashboard*, dan

halaman admin. Tiap bagian antarmuka sistem akan dijelaskan pada masing-masing subbab.

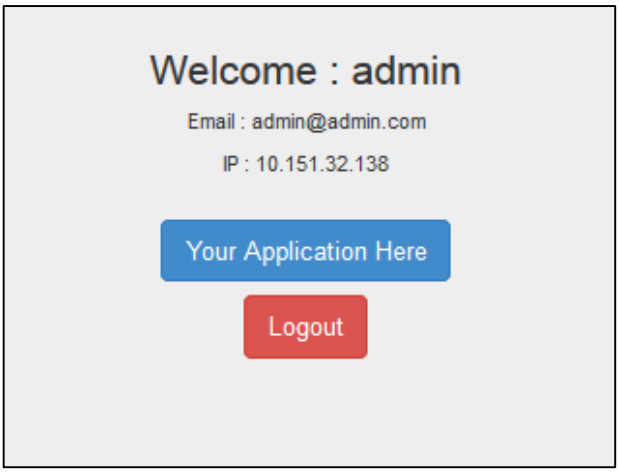
4.4.1. Implementasi Halaman Antarmuka Registrasi

Gambar 4.10 merupakan *screenshot* hasil implementasi antarmuka halaman registrasi. Gambar 4.10 merupakan halaman antarmuka ketika *tenant* ingin menggunakan layanan aplikasi manajemen restoran, *tenant* diharuskan mendaftar terlebih dahulu dengan mengisi *item* yang ada.

Gambar 4.10. Antarmuka Halaman Registrasi

4.4.2. Implementasi Antarmuka *Dashboard*

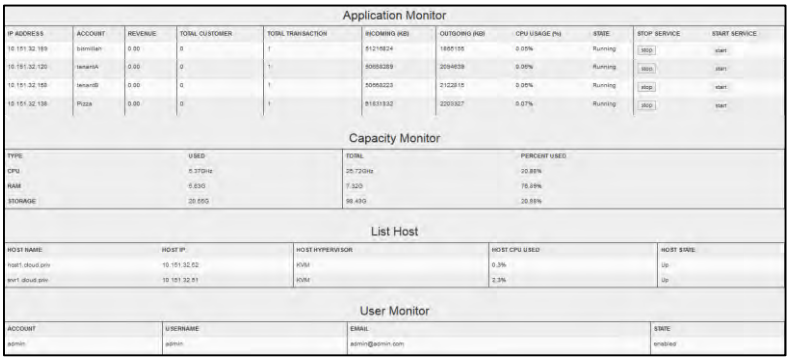
Gambar 4.11 merupakan *screenshot* hasil implementasi antarmuka halaman *dashboard*. Gambar 4.11 merupakan halaman *dashboard* milik *tenant* yang berisi informasi dari aplikasi manajemen restoran miliknya. *Tenant* dapat mengakses aplikasi manajemen restoran melalui tombol yang disediakan pada halaman ini.



Gambar 4.11. Antarmuka Halaman *Dashboard*

4.4.3. Implementasi Antarmuka Sistem *Monitoring*

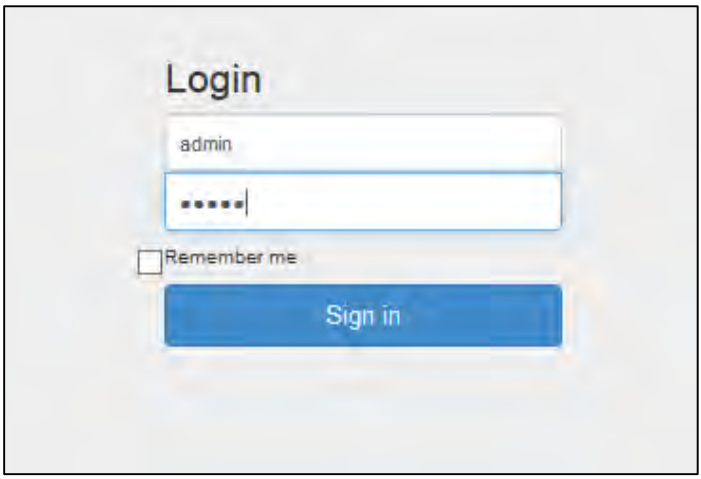
Gambar 4.12 merupakan *screenshot* hasil implementasi antarmuka halaman sistem *monitoring*. Gambar 4.12 merupakan informasi dari aplikasi milik *tenant*, dalam tabel ini, admin dapat mengontrol setiap aplikasi milik *tenant*.



Gambar 4.12. Antarmuka Halaman Monitoring

4.4.4. Implementasi Antarmuka *Login*

Gambar 4.13 merupakan *screenshot* hasil implementasi antarmuka halaman *login*. Gambar 4.13 merupakan halaman antarmuka ketika *tenant* ingin melakukan *login* serta mengakses aplikasi manajemen restoran sesuai dengan akun miliknya.



Gambar 4.13. Antarmuka Halaman *Login*

BAB V
PENGUJIAN DAN EVALUASI

Bab ini berisi penjelasan mengenai hasil pengujian dan evaluasi dari implementasi perangkat lunak yang dikembangkan. Adapun hal-hal yang dibahas dalam bab ini adalah lingkungan uji coba perangkat lunak, uji coba fungsionalitas perangkat lunak. Uji coba ini dilakukan dengan beberapa skenario ujicoba.

5.1. Lingkungan Uji Coba

Lingkungan uji coba menjelaskan mengenai lingkungan dimana perangkat lunak akan diuji. Lingkungan ujicoba ini terdiri dari lingkungan ujicoba perangkat keras dan alat kakas. Lingkungan uji coba perangkat keras terdiri dari sebuah komputer yang berfungsi sebagai *webserver* sistem penyedia layanan aplikasi manajemen restoran, dan dua buah komputer komputasi awan yang digunakan sebagai lingkungan virtualisasi. Spesifikasi lingkungan uji coba perangkat keras adalah sebagai berikut.

- Komputer merek Lenovo
  Spesifikasi perangkat keras
  - Prosesor Intel® Core™ i3-3240 CPU @ 3.40GHz
  - Memori 4GB DDR3

  Spesifikasi perangkat lunak
  - Windows 8 Pro 64-Bit
  - Apache xampp sebagai platform *web server* untuk menjalankan situs web.
- Dua buah komputer merek ACER sebagai komputasi awan.
  Spesifikasi perangkat keras
  - Prosesor Intel (R) Core (TM) i3-2120 CPU @ 3.30GHz.
  - Memori 4GB DDR3.
  - 500GB WDC WD5000AAKX-2, dan
  - Realtek RTL8111/8168B PCI Express Gigabit Ethernet.

  Spesifikasi perangkat lunak
  - CentOS 6.5 Minimal.

- Satu buah komputer merek HP sebagai komputasi awan.
  Spesifikasi perangkat keras
  - Prosesor Intel (R) Core (TM) i3-2120 CPU @ 3.30GHz.
  - Memori 4GB DDR3.
  - 500GB WDC WD5000AAKX-2, dan
  - Realtek RTL8111/8168B PCI Express Gigabit Ethernet.

  Spesifikasi perangkat lunak
  - CentOS 6.5 Minimal.

5.2. Skenario Pengujian

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Pengujian yang dilakukan adalah pengujian kebutuhan fungsionalitas dan pengujian performa. Pengujian fungsionalitas menggunakan metode kotak hitam (*blackbox*). Metode ini menekankan pada kesesuaian hasil keluaran sistem.

5.2.1. Pengujian Fungsionalitas

Pengujian fungsionalitas sistem dilakukan untuk menguji apakah fungsionalitas sistem berjalan sesuai dengan yang diharapkan. Pengujian dilakukan pada keseluruhan fungsi baik itu pada sisi sistem pusat dan sistem VM. Uji coba fungsionalitas meliputi semua alir program yang sudah dijelaskan pada Bab III yakni.

a. *Generate Tenant* (UC-01)
   Generate Tenant merupakan fungsi yang digunakan sistem untuk menciptakan sebuah VM sesuai dengan permintaan *tenant*. Fungsi ini membutuhkan parameter berupa data *tenant* dan keluaran berupa alokasi VM dengan alamat IP unik.
b. Akses Aplikasi (UC-02)
   Setelah fungsi *Generate Tenant* dijalankan, *tenant* dapat mengakses sebuah aplikasi dengan menggunakan data *tenant*. Fungsi ini yang bertugas dalam pengaksesan aplikasi setiap *tenant*. Pengujian fitur *multi-tenancy* akan dijabarkan pada uji coba fungsionalitas ini.

c. Ambil Data Aplikasi *Tenant* (UC-03)
   Setelah fungsi *Generate Tenant* dijalankan, setiap VM yang digunakan sudah terpasang sebuah *script* yang digunakan untuk memantau data aplikasi setiap *tenant*. Fungsi ini bertugas sebagai pengambilan data aplikasi setiap *tenant* yang nantinya dirubah sebagai berkas XML.
d. *Monitoring* Aplikasi *Tenant* (UC-04)
   Setelah fungsi Ambil Data Aplikasi *Tenant* berhasil dijalankan pada setiap VM *tenant* yang tercipta. Fungsi ini membutuhkan masukan berupa berkas XML dari setiap VM milik *tenant* (UC-03). Berkas XML yang terkumpul akan diambil datanya dan ditampilkan pada halaman sistem *monitoring*.
e. *Monitoring* Infrastruktur Komputasi Awan (UC-05)
   Fungsi *Monitoring* Infrastruktur Komputasi Awan digunakan sebagai pemantauan kondisi infrastruktur komputasi awan. Data yang dapat ditampilkan seperti kapasitas CPU, kapasitas penyimpanan, dan kapasitas RAM.
f. Tambah Komputasi (UC-06)
   Fungsi Tambah Komputasi digunakan sebagai penambahan sumber daya komputasi pada infrastruktur komputasi awan dengan mengambil perangkat cadangan yang telah dipersiapkan sebelumnya.

5.2.1.1. Pengujian Fungsionalitas *Generate Tenant*

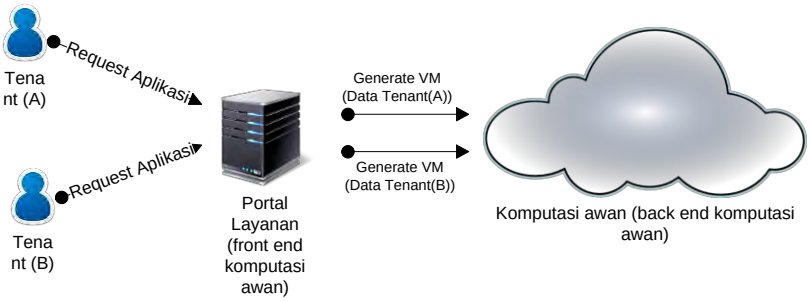
Pengujian melakukan *Generate Tenant* merupakan pengujian untuk menguji fungsionalitas sistem dalam menyediakan fasilitas pembuatan VM berdasarkan informasi ID *tenant*. Tujuan pengujian ini adalah untuk memastikan fungsionalitas sistem tersebut dapat berjalan dengan baik. Rincian prosedur pengujian melakukan *Generate Tenant* dapat dilihat pada Tabel 5.1.

Berdasarkan skenario pada Tabel 5.1, sistem akan menciptakan VM secara otomatis berdasarkan ID *tenant* yang melakukan *request* aplikasi. Hasil keluaran uji coba fungsionalitas

generate VM merupakan data dengan format JSON yang berisi informasi VM yang telah diciptakan.

Tabel 5.1. Prosedur Uji Coba *Generate Tenant*

ID	UJ-01
Nama	**Uji Coba Melakukan *Generate Tenant***
Tujuan Pengujian	Menguji fungsionalitas untuk melakukan pembuatan VM berdasarkan informasi *tenant*
Skenario	***Tenant* melakukan request aplikasi.**
Kondisi Awal	Sistem menampilkan sebuah form *register* dan *build* VM
Masukan	**Informasi *Tenant***
Hasil Yang Didapat	*Tenant* terdaftar dan VM diciptakan sesuai dengan ID milik *tenant*.
Hasil Pengujian	**Berhasil**



Gambar 5.1. Ilustrasi Uji Coba *Generate Tenant*

Uji Coba dilakukan sesuai dengan ilustrasi pada Gambar 5.1, pada kasus ini, terdapat dua *tenant* yang melakukan *request aplikasi* manajemen restoran pada portal layanan. *Tenant* A dan B diharuskan mengisi sebuah *form* yang telah disediakan pada portal layanan. Gambar 5.2 merupakan contoh *tenant* A dan B yang melakukan pengisian *form* permintaan aplikasi.

Gambar 5.2. Tenant Mengisi Permintaan Aplikasi. (a) Tenant A (b) Tenant B

Tabel 5.2 merupakan rincian lengkap data *tenant* A dan *tenant* B yang diisi pada *form* untuk proses request aplikasi pada portal.

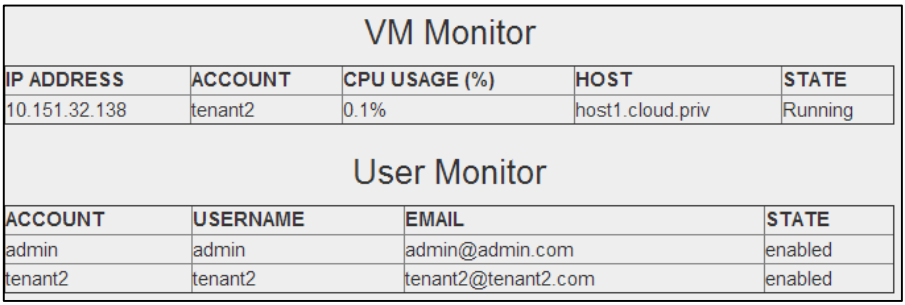
Tabel 5.2 Rincian Data *Tenant* A dan *Tenant* B

Atribut	Tenant A	Tenant B	Catatan
First Name	Tenant	Tenant	-
Last Name	A	B	-
Account	TenantA	TenantB	Data Account digunakan sebagai acuan pembuatan VM
Email Address	TenantA@TenantA.com	TenantB@TenantB.com	-

Atribut	Tenant A	Tenant B	Catatan
Username	TenantA	TenantB	-
Password	TenantA	TenantB	-

Setelah *tenant* mengisi form yang telah disediakan, sistem segera memproses masing-masing data tersebut dengan menjalankan fungsi *generate tenant* dengan parameter data pada Tabel 5.2.

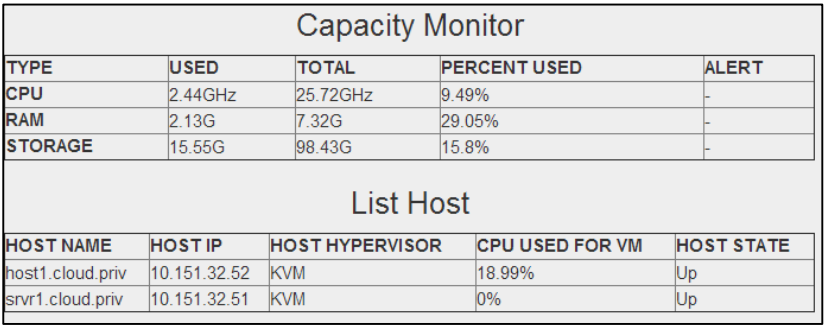
Gambar 5.3. merupakan tampilan sistem sebelum *Tenant* A dan B melakukan *request* sebuah aplikasi. Tampak hanya satu *tenant* yang terdapat pada sistem yaitu "tenant2" dengan alamat IP 10.151.32.138.



Gambar 5.3. Tampilan Sistem *Monitoring* Aplikasi Sebelum *Tenant* A dan B Melakukan *Request* Aplikasi

Gambar 5.4 merupakan kondisi sistem ketika hanya terdapat satu *tenant* pada komputasi awan. Gambar 5.5. merupakan tampilan sistem sesudah *tenant* A dan *tenant* B melakukan *request* sebuah aplikasi pada sistem, terlihat bahwa aplikasi *tenant* A dan *tenant* B telah tercipta dengan alamat IP 10.151.32.125 untuk *tenant* A dan 10.151.32.119 untuk *tenant* B. Gambar 5.6 merupakan kondisi sistem komputasi awan ketika proses *request tenant A* dan B berhasil, tampak

terdapat peningkatan dalam penggunaan CPU,RAM dan STORAGE pada komputasi awan.



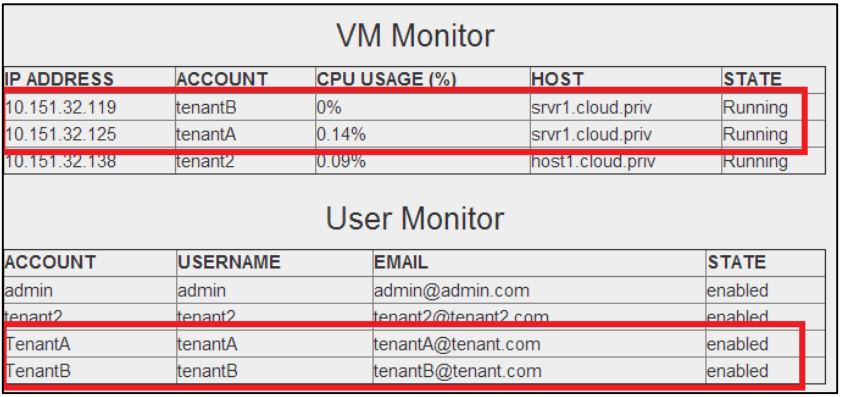
Capacity Monitor

TYPE	USED	TOTAL	PERCENT USED	ALERT
CPU	2.44GHz	25.72GHz	9.49%	-
RAM	2.13G	7.32G	29.05%	-
STORAGE	15.55G	98.43G	15.8%	-

List Host

HOST NAME	HOST IP	HOST HYPERVISOR	CPU USED FOR VM	HOST STATE
host1.cloud.priv	10.151.32.52	KVM	18.99%	Up
srvr1.cloud.priv	10.151.32.51	KVM	0%	Up

Gambar 5.5. Kondisi Sistem Komputasi Awan dengan Satu *Tenant*



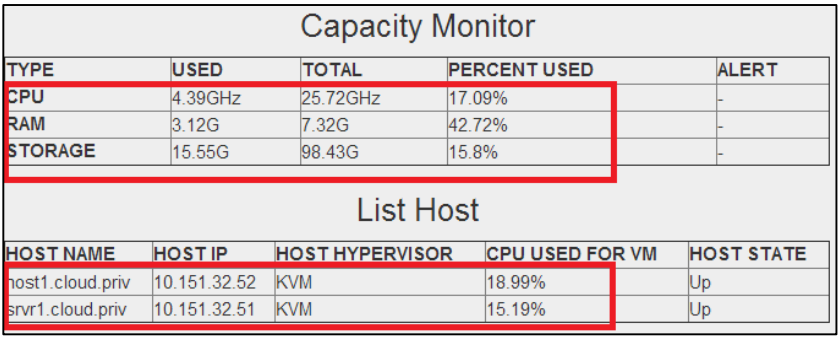
VM Monitor

IP ADDRESS	ACCOUNT	CPU USAGE (%)	HOST	STATE
10.151.32.119	tenantB	0%	srvr1.cloud.priv	Running
10.151.32.125	tenantA	0.14%	srvr1.cloud.priv	Running
10.151.32.138	tenant2	0.09%	host1.cloud.priv	Running

User Monitor

ACCOUNT	USERNAME	EMAIL	STATE
admin	admin	admin@admin.com	enabled
tenant2	tenant2	tenant2@tenant2.com	enabled
TenantA	tenantA	tenantA@tenant.com	enabled
TenantB	tenantB	tenantB@tenant.com	enabled

Gambar 5.4. Perubahan Data Sistem *Monitoring* Aplikasi Setelah *Tenant* A dan B Melakukan *Request* Aplikasi pada Portal Layanan



Capacity Monitor

TYPE	USED	TOTAL	PERCENT USED	ALERT
CPU	4.39GHz	25.72GHz	17.09%	-
RAM	3.12G	7.32G	42.72%	-
STORAGE	15.55G	98.43G	15.8%	-

List Host

HOST NAME	HOST IP	HOST HYPERVISOR	CPU USED FOR VM	HOST STATE
host1.cloud.priv	10.151.32.52	KVM	18.99%	Up
srvr1.cloud.priv	10.151.32.51	KVM	15.19%	Up

Gambar 5.6. Perubahan Kondisi Sistem Komputasi Awan ketika Dua *Tenant* (A dan B) Tercipta

Perubahan kondisi sistem komputasi awan dapat dilihat pada Tabel 5.3. Sebelum *tenant* A dan B tercipta, *host* srvr1.cloud.priv tidak terdapat *tenant* sehingga beban CPU pada saat ini sebesar 0%, kondisi sistem awan secara kesuluruhan adalah sebesar 9,49% pada CPU, RAM sebesar 29.05% dan STORAGE sebesar 15,8%. Pada saat *tenant* A dan B tercipta, kedua *tenant* ini ditempatkan secara otomatis pada *host* srvr1.cloud.priv, sehingga beban CPU pada *host* ini mengalami peningkatan sebesar 15,19%, dengan peningkatan sistem secara kesuluruhan sebesar 17,09% pada CPU, 42,72% pada RAM dan 15,8% pada STORAGE.

Tabel 5.3. Perubahan Kondisi Sistem Setelah *Tenant* A dan B Tercipta

Kondisi	Host	Nama *Tenant*	CPU *HOST* (%)	Kondisi Sistem Kesuluruhan		
				CPU (%)	RAM (%)	STORAGE (%)
Sebelum *tenant* A dan B tercipta	Host1.cloud.priv	Tenant2	18.99	9.49	29.05	15.8
	Srvr1.cloud.priv	Tidak Ada *Tenant*	0			
Sesudah tenant A	Host1.cloud.priv	Tenant2	18.99	17.09	42.72	15.8
	Srvr1.cloud.priv	TenantA TenantB	15.19			

Kondisi	Host	Nama *Tenant*	CPU *HOST* (%)	Kondisi Sistem Kesuluruhan		
				CPU (%)	RAM (%)	STORAGE (%)
dan B tercipta						

5.2.1.2. Pengujian Melakukan Akses Aplikasi

Pengujian melakukan akses aplikasi merupakan pengujian terhadap kemampuan sistem dalam menyediakan hak akses aplikasi pada setiap *tenant*. Tujuan uji coba melakukan akses aplikasi adalah untuk menguji apakah fungsionaliltas fungsi ini berjalan sesuai dengan rancangan. Rincian prosedur uji coba melakukan akses aplikasi dapat dilihat pada Tabel 5.4.

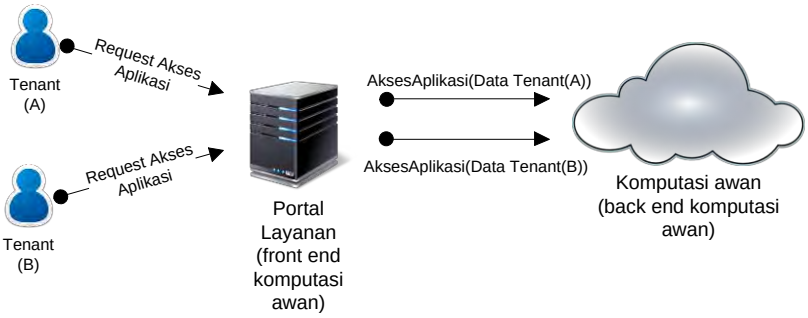
Tabel 5.4. Prosedur Uji Coba Melakukan Akses Aplikasi

ID	UJ-02
Nama	**Uji Coba Melakukan Akses Aplikasi**
Tujuan Pengujian	Menguji fungsionalitas untuk mengakses aplikasi berdasarkan ID *tenant*
Skenario	***Tenant* mengakses aplikasi manajemen restoran**
Kondisi Awal	Sistem menampilkan halaman *login*
Masukan	***Username* dan *password***
Hasil Yang Didapat	Aplikasi manajemen restoran sesuai dengan milik *Tenant*
Hasil Pengujian	**Berhasil**

Berdasarkan Tabel 5.4, kondisi awal uji coba akses aplikasi adalah hasil keluaran dari uji coba UJ-01. Skenario dari pengujian ini adalah *tenant* melakukan *login* pada sistem portal layanan dan sistem mengarahkan *tenant* menuju halaman *dashboard* milik *tenant*. Pada halaman *dashboard* terdapat akses menuju aplikasi manajemen restoran milik *tenant*.

Uji coba dilakukan sesuai dengan ilustrasi pada Gambar 5.7. Dalam kasus ini, *Tenant* A dan B mencoba untuk mengakses aplikasi mereka. Sistem menyediakan halaman *login* sebagai sarana *tenant* untuk mengakses aplikasi mereka. Gambar 5.8 merupakan tenant B mencoba mengakses aplikasi melalui portal.

Setelah *tenant* B mengisi data yang tertera pada Gambar 5.8, sistem segera melakukan proses akses aplikasi berdasarkan ID *tenant* tersebut. Apabila ID *tenant* valid, sistem segera mengarahkan *tenant* pada halaman *dashboard*, dalam proses ini sistem menampilkan alamat aplikasi manajemen restoran sesuai dengan ID *tenant.*

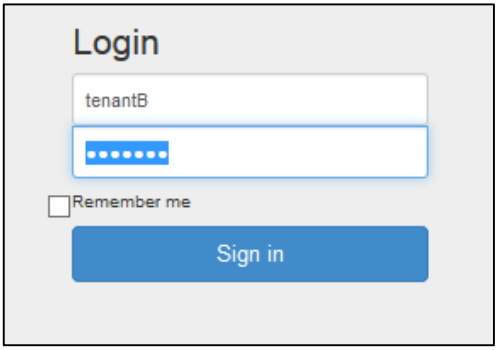


Gambar 5.7. Ilustrasi Uji Coba Akses Aplikasi

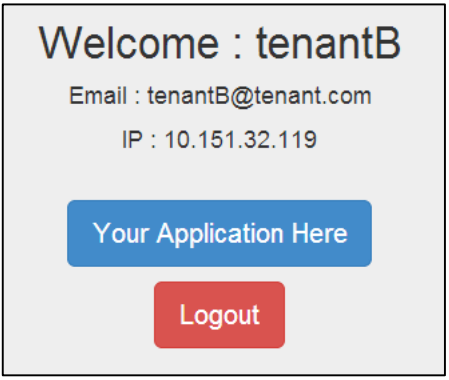
Gambar 5.9 merupakan halaman *dashboard* TenantB yang berhasil melakukan *login*, pada halaman *dashboard* tersebut tersedia informasi alamat IP unik aplikasi manajemen restoran milik TenantB serta tombol akses menuju aplikasi.

Gambar 5.10 merupakan aplikasi manajemen restoran milik *tenant* B yang telah tercipta melalui proses uji coba UJ-01. Terlihat

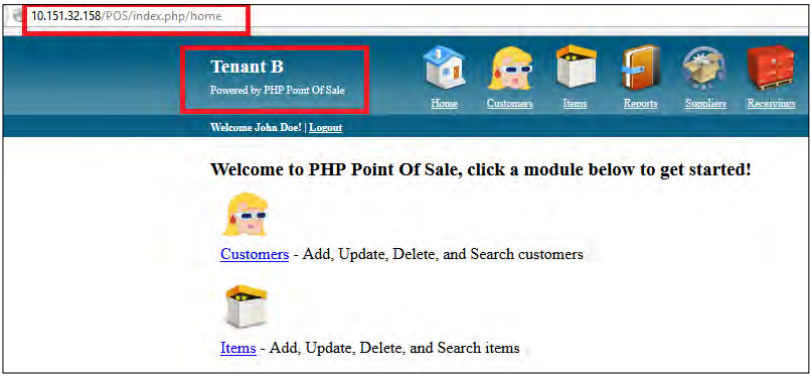
bahwa IP 10.151.32.158 merupakan alamat aplikasi manajemen restoran milik *tenant* B. Sebagai perbandingan, Gambar 5.11 merupakan aplikasi manajemen restoran milik *tenant* A dengan alamat IP aplikasi manajemen restoran 10.151.32.120. Dapat disimpulkan dari hasil uji coba ini sistem dapat melakukan *multitenancy* dengan baik.



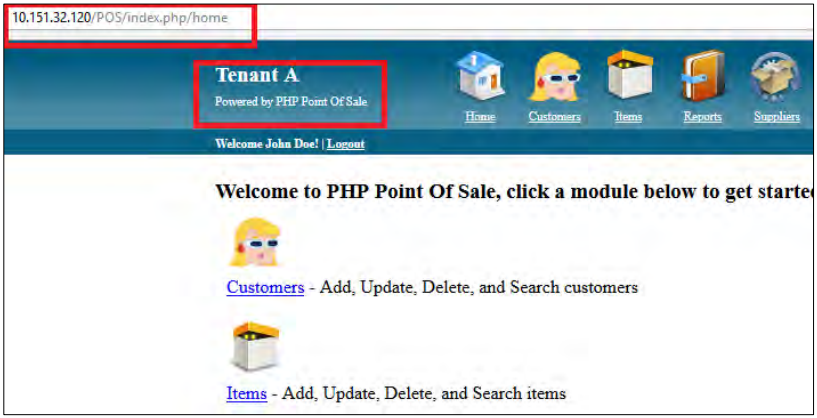
Gambar 5.8. *Tenant* B Melakukan Login pada Sistem



Gambar 5.9. Halaman *Dashboard Tenant* B



Gambar 5.11. Aplikasi *Tenant* B



Gambar 5.10. Aplikasi *Tenant* A

5.2.1.3. Pengujian Ambil Data Aplikasi *Tenant*

Pengujian melakukan ambil data *tenant* merupakan pengujian terhadap kemampuan sistem dalam mengambil data pada *database* aplikasi milik setiap *tenant*. Tujuan uji coba ini adalah untuk menguji apakah fungsionalitas sistem VM tersebut bekerja dengan baik.

Adapun rincian prosedur uji coba melakukan ambil data *tenant* dapat dilihat pada Tabel 5.5.

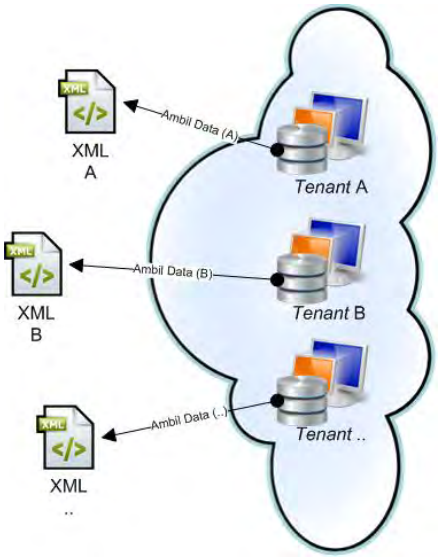
Tabel 5.5. Prosedur Uji Coba Ambil Data Aplikasi Tenant

ID	UJ-03
Nama	**Uji Coba Melakukan Ambil Data Aplikasi *Tenant***
Tujuan Pengujian	Menguji fungsionalitas untuk mengambil data aplikasi *tenant*
Skenario	**VM aplikasi manajemen restoran milik *tenant* telah aktif**
Kondisi Awal	Sistem VM mempunyai informasi *database tenant*
Masukan	***Database* informasi aplikasi *tenant***
Hasil Yang Didapat	Berkas XML berisi data aplikasi *tenant*
Hasil Pengujian	**Berhasil**

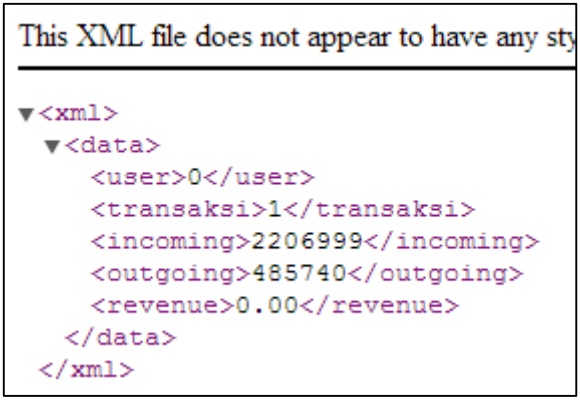
Berdasarkan Tabel 5.5, kondisi awal dari uji coba ini adalah sistem VM mengakses *database* berisi informasi aplikasi milik *tenant*. Adapun skenario dari pengujian ini adalah sistem VM mengambil informasi aplikasi milik *tenant* dari *database* setiap VM, dari informasi tersebut dibuat sebuah berkas XML. Gambar 5.12 merupakan ilustrasi pengambilan data aplikasi *tenant.*

Hasil keluaran uji coba ambil data aplikasi *tenant* dapat dilihat pada Gambar 5.13. Pada Gambar 5.13, berkas XML berisi informasi data aplikasi *tenant* dan *bandwidth* yang digunakan tiap *tenant*.

Informasi yang terdapat pada berkas ini adalah total user, total transaksi, *bandwitdh*, dan *revenue*.



Gambar 5.12. Ilustrasi Pengambilan Data pada Setiap VM *Tenant*



Gambar 5.13. Hasil Uji Coba Ambil Data *Tenant*

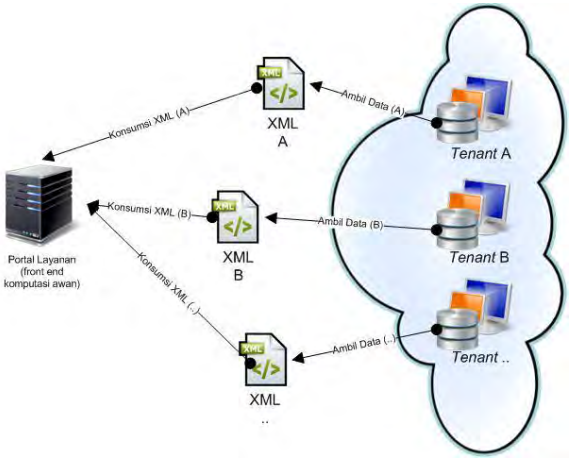
5.2.1.4. Pengujian Sistem Monitoring Aplikasi

Pengujian melakukan sistem *monitoring* merupakan pengujian terhadap kemampuan sistem dalam melakukan *monitoring* aplikasi milik *tenant*. Tujuan uji coba *monitoring* ini adalah untuk menguji apakah fungsionalitas sistem tersebut berjalan dengan baik. Adapun rincian prosedur uji coba *monitoring* dapat dilihat pada Tabel 5.6.

Tabel 5.6. Prosedur Uji Coba Monitoring Aplikasi

ID	UJ-04
Nama	**Uji Coba Melakukan *Monitoring* Aplikasi**
Tujuan Pengujian	Menguji fungsionalitas untuk melakukan *monitoring* aplikasi setiap *tenant*
Skenario	- **Sistem mengambil informasi data aplikasi *tenant* dari berkas XML pada sistem VM setiap *tenant* dan menampilkan data aplikasi pada halaman *monitoring*.** - ***Tenant* B menggunakan aplikasi miliknya** - **Webserver *Tenant* B dilakukan stress *test* menggunakan Apache Benchmark.**
Kondisi Awal	Sistem mempunyai alamat IP setiap sistem VM yang akan dimonitoring
Masukan	**Berkas *XML* berisi informasi aplikasi *tenant***
Hasil Yang Didapat	Data semua aplikasi *tenant.*
Hasil Pengujian	**Berhasil**

Berdasarkan Tabel 5.6, kondisi awal uji coba *monitoring* aplikasi milik *tenant* adalah sistem memiliki berkas XML setiap VM milik *tenant* hasil keluaran uji coba UJ-03. Cara kerja uji coba ini adalah sistem mengambil dan menampilkan informasi dari berkas XML yang berisi data aplikasi setiap *tenant*. Gambar 5.14 merupakan ilustrasi pengambilan data XML pada setiap aplikasi *tenant*. Gambar 5.15 merupakan kondisi awal sistem monitoring. Terdapat tiga *tenant* pada kondisi awal, yaitu TenantA, TenantB dan Tenant2 dengan data *customer*, *account*, *revenue*, *transaction*, *bandwidth* dan CPU *usage* yang tertera pada Gambar 5.15.



Gambar 5.15. Ilustrasi Sistem Monitoring Aplikasi

VM Monitor

IP ADDRESS	ACCOUNT	REVENUE	TOTAL CUSTOMER	TOTAL TRANSACTION	INCOMING (KB)	OUTGOING (KB)	CPU USAGE (%)	HOST	STATE
10.151.32.115	tenantB	0.00	0	0	34970	28524	0.86%	host1.cloud.priv	Running
10.151.32.205	tenantA	0.00	0	0	44517	34748	0.13%	srvr1.cloud.priv	Running
10.151.32.138	tenant2	0.00	0	0	4686874	1837721	0.09%	host1.cloud.priv	Running

Gambar 5.14. Sistem Monitoring Aplikasi

Pengujian sistem monitoring aplikasi dilakukan dengan menggunakan dua skenario sebagai berikut.

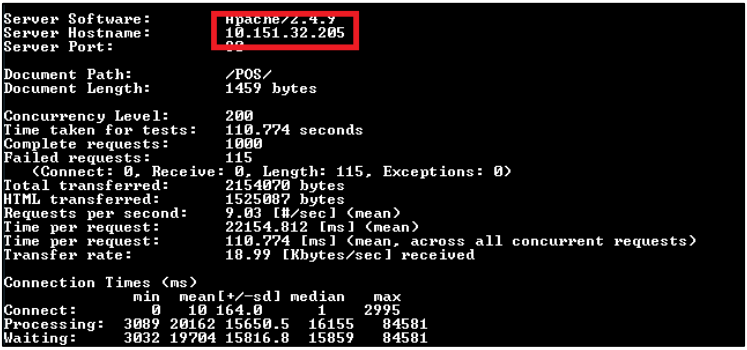
1. Skenario pertama, *Tenant* B dengan alamat aplikasi 10.151.32.115 menggunakan aplikasi manajemen restoran miliknya, dengan menjalankan fungsionalitas aplikasi manajemen restoran seperti pencatatan data pelanggan dan pembayaran.
2. Skenario kedua, dengan menggunakan kakas Apache Benchmark yang berfungsi sebagai uji stres pada webserver VM *tenant* untuk memantau penggunaan CPU VM *tenant*. Pada skenario ini, VM *tenant* yang akan diuji adalah VM milik *tenant* A.

Tabel 5.7 merupakan hasil dari menjalankan skenario 1 dan 2. Skenario 0 merupakan kondisi awal sistem monitoring ketika tidak terdapat pergerakan pada VM yang dipantau. Ketika skenario 1 dijalankan, terlihat pada Tabel 5.7 bahwa *Revenue*, *Total Customer*, dan *Total Transaction* dari *Tenant* B mengalami perubahan nilai. Ketika skenario 2 dijalankan, terlihat pada Tabel 5.7 bahwa terjadi peningkatan CPU VM milik *tenant* A sebesar 10,44%.

Tabel 5.7. Hasil Menjalankan Skenario 1 dan 2

SKENARIO	HOST	NAMA TENANT	CPU VM TENANT (%)	REVENUE ($)	TOTAL CUSTOMER	TOTAL TRANSACTION
0	Host1.cloud.priv	Tenant 2	0.09	0.00	0	0
	Srvr1.cloud.priv	Tenant A	0.13	0.00	0	0
		Tenant B	0.86	0.00	0	0
1	Host1.cloud.priv	Tenant 2	0.10	0.00	0	0
	Srvr1.cloud.priv	Tenant A	0.10	0.00	0	0
		Tenant B	0.27	160.00	3	2

SKENARIO	HOST	NAMA TENANT	CPU VM TENANT (%)	REVENUE ($)	TOTAL CUSTOMER	TOTAL TRANSACTION
2	Host1.cloud.priv	Tenant 2	0.10	0.00	0	0
	Srvr1.cloud.priv	Tenant A	10.44	0.00	0	0
		Tenant B	0.09	160.00	3	2



Gambar 5.16. Pengujian Penggunaan CPU pada VM milik *tenant* A

Gambar 5.16 merupakan *stress test* pada VM milik *tenant* A dengan alamat IP 10.151.32.205 menggunakan kakas Apache Benchmark untuk meningkatkan penggunaan dari CPU. Pada Gambar 5.17 merupakan tampilan sistem *monitoring* setelah dilakukan

VM Monitor

IP ADDRESS	ACCOUNT	REVENUE	TOTAL CUSTOMER	TOTAL TRANSACTION	INCOMING (KB)	OUTGOING (KB)	CPU USAGE (%)	HOST	STATE
10.151.32.115	tenantB	160.00	3	2	565254	932803	0.07%	host1.cloud.priv	Running
10.151.32.205	tenantA	0.00	0	1	996461	3928091	10.44%	srvr1.cloud.priv	Running
10.151.32.138	tenant2	0.00	0	1	417405	219292	0.1%	srvr1.cloud.priv	Running

Gambar 5.17. Perubahan Data Sistem *Monitoring*

beberapa skenario, terlihat perubahan data dibandingkan dengan kondisi awal sistem monitoring pada Gambar 5.14.

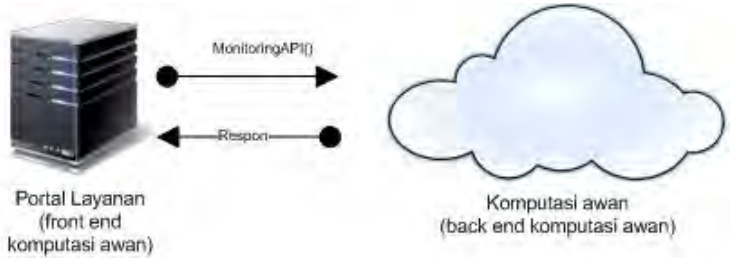
5.2.1.5. Pengujian Sistem Monitoring Infrastruktur Awan

Pengujian melakukan sistem *monitoring* infrastruktur awan merupakan pengujian terhadap kemampuan sistem dalam melakukan *monitoring* data pada infrastruktur komputasi awan. Tujuan uji coba *monitoring* ini adalah untuk menguji apakah fungsionalitas sistem tersebut berjalan dengan baik. Adapun rincian prosedur uji coba *monitoring* dapat dilihat pada Tabel 5.8.

Tabel 5.8. Prosedur Uji Coba Monitoring Infrastruktur Awan

ID	UJ-05
Nama	**Uji Coba Melakukan *Monitoring* Infrastruktur Komputasi Awan**
Tujuan Pengujian	Menguji fungsionalitas untuk melakukan *monitoring* infrastruktur komputasi awan.
Skenario	**Sistem mengambil informasi infrastruktur komputasi awan.**
Kondisi Awal	Sistem diijinkan untuk mengakses API infrastruktur komputasi awan.
Masukan	-
Hasil Yang Didapat	Data *monitoring*.
Hasil Pengujian	**Berhasil**

Uji coba dilakukan pada sistem portal dengan mengakses API dari infrastruktur komputasi awan. Gambar 5.18 merupakan ilustrasi pemanggilan API dari portal layanan.



Gambar 5.18. Ilustrasi Sistem *Monitoring* Infrastruktur Awan

Gambar 5.19 merupakan hasil uji coba monitoring infrastruktur komputasi awan, terdapat informasi berupa kapasitas RAM, CPU, dan STORAGE pada tabel capacity monitor serta terdapat informasi berupa nama perangkat komputasi, alamat IP, penggunaan CPU, dan kondisi dari perangkat komputasi pada tabel list *host*.

Capacity Monitor

TYPE	USED	TOTAL	PERCENT USED	ALERT
CPU	4.39GHz	25.72GHz	17.09%	-
RAM	3.12G	7.32G	42.72%	-
STORAGE	15.55G	98.43G	15.8%	-

List Host

HOST NAME	HOST IP	HOST HYPERVISOR	CPU USED FOR VM	HOST STATE
host1.cloud.priv	10.151.32.52	KVM	18.99%	Up
srvr1.cloud.priv	10.151.32.51	KVM	15.19%	Up

Gambar 5.19. Sistem *Monitoring* Infrastruktur Awan

5.2.1.6. Pengujian Tambah Komputasi

Pengujian fungsionalitas tambah komputasi merupakan pengujian terhadap kemampuan sistem dalam menambah sumber daya perangkat komputasi secara otomatis ketika penggunaan sistem melebihi batasan yang sudah ditetapkan. Tujuan uji coba tambah komputasi ini adalah untuk menguji apakah fungsionalitas sistem tersebut berjalan dengan baik. Adapun rincian prosedur uji coba tambah komputasi dapat dilihat pada Tabel 5.9.

Tabel 5.9. Prosedur Uji Coba Menambah Komputasi

ID	UJ-06
Nama	**Uji Coba Menambah Komputasi**
Tujuan Pengujian	Menguji fungsionalitas untuk melakukan penambahan perangkat komputasi pada infrastruktur komputasi awan
Skenario	**Ketika beban penggunaan melebihi batas yang telah ditentukan, maka fungsi tambah komputasi berjalan**
Kondisi Awal	Sistem mempunyai data perangkat yang ingin ditambahkan
Masukan	**Data kapasitas berupa penggunaan RAM**
Hasil Yang Didapat	Perangkat komputasi baru ditambahkan.
Hasil Pengujian	**Berhasil**

Output dari fungsionalitas tambah komputasi adalah sebuah perangkat komputasi baru yang tergabung pada infrastruktur

komputasi awan secara otomatis. Berdasarkan skenario pada Tabel 5.9, fungsi ini berjalan ketika data kapasitas komputasi melebihi batas yang telah ditentukan, untuk menentukan batasan kapasitas dari perangkat yang sudah ada, dilakukan uji coba terlebih dahulu pada perangkat komputasi yang sudah ada dengan menambah *tenant* pada perangkat hingga perangkat mengalami kelebihan kapasitas. Acuan yang digunakan dalam fungsi ini adalah penggunaan RAM pada perangkat komputasi. Tabel 5.10 merupakan uji coba untuk melihat seberapa besar batas kemampuan kapasitas sistem dengan dua perangkat komputasi dalam menampung VM milik *tenant*.

Tabel 5.10. Uji Coba Kapasitas Sistem dengan Dua Perangkat Komputasi

JUMLAH TENANT	PENGGUNAAN RAM TOTAL		PENGGUNAAN RAM PADA HOST (GB)		STATUS TENANT
	(GB)	(%)	SRVR1.CLOUD	HOST1.CLOUD	
0	1,13 / 7,32	15	1,10 / 3,66	1,13 / 3,66	RUNNING
1	2,13 / 7,32	29	1,10 / 3,66	2,13 / 3,66	RUNNING
2	2,63 / 7,32	35	1,10 / 3,66	2,63 / 3,66	RUNNING
3	3,13 / 7,32	42	1,10 / 3,66	3,13 / 3,66	RUNNING
4	3,63 / 7,32	49	1,60 / 3,66	3,13 / 3,66	RUNNING
5	4,13 / 7,32	56	1,60 / 3,66	3,63 / 3,66	RUNNING
6	4,63 / 7,32	63	2,10 / 3,66	3,63 / 3,66	RUNNING
7	5,13 / 7,32	70	2,60 / 3,66	3,63 / 3,66	RUNNING
8	5,63 / 7,32	76	3,10 / 3,66	3,63 / 3,66	RUNNING
9	6,13 / 7,32	83	2,50 / 3,66	3,63 / 3,66	RUNNING

JUMLAH TENANT	PENGGUNAAN RAM TOTAL		PENGGUNAAN RAM PADA HOST (GB)		STATUS TENANT
	(GB)	(%)	SRVR1.CLOUD	HOST1.CLOUD	
10	6,23 / 7,32	85	2,60 / 3,66	3,63 / 3,66	ERROR

KETERANGAN :

: **Batas kapasitas sistem dengan dua perangkat komputasi mampu melayani *tenant***

: ***Tenant* terakhir yang mampu diproses oleh sistem**

Dari hasil percobaan pada Tabel 5.10 dapat disimpulkan bahwa sistem dengan konfigurasi seperti yang dijelaskan pada Halaman 61 mampu menanggapi permintaan sebanyak 9 *tenant*, selebihnya sistem akan mengembalikan pesan error yang menunjukkan bahwa kapasitas sistem telah melampui batas. Dari percobaan di atas didapatkan nilai selisih RAM sebesar 1,20 GB sebagai batas kemampuan sistem komputasi awan dalam menciptakan VM milik *tenant*. Nilai ini menjadi acuan dalam menentukan kapan sistem harus menjalankan fungsi tambah komputasi agar sistem selalu mempunyai kondisi prima dalam melayani kebutuhan *tenant*. Pada Tabel 5.11 merupakan hasil pengujian ketika fungsi tambah komputasi dijalankan. Dari hasil percobaan pada Tabel 5.11, sistem mampu menambah perangkat komputasi ketika kapasitas sistem mencapai batas yang sudah ditentukan sebelumnya hal ini ditunjukkan dengan *Host2.cloud.priv* yang telah aktif. Sehingga RAM total menunjukkan peningkatan sebesar 1,66 GB menjadi 8,96 GB sesuai dengan perangkat komputasi yang baru ditambahkan. Dengan kapasitas sistem yang bertambah, sistem secara total mampu menampung hingga 12 *tenant*.

Tabel 5.11. Pengujian Sistem dengan Menerapkan Fungsi Tambah Komputasi

JUMLAH TENANT	PENGGUNAAN RAM TOTAL		PENGGUNAAN RAM PADA SETIAP HOST (GB)			STATUS TENANT
	(GB)	(%)	SRVR1	HOST1	HOST2	
0	1,13 / 7,32	15	1,10 / 3,66	1,13 / 3,66	DOWN	RUNNING
1	2,13 / 7,32	29	1,10 / 3,66	2,13 / 3,66	DOWN	RUNNING
2	2,63 / 7,32	35	1,10 / 3,66	2,63 / 3,66	DOWN	RUNNING
3	3,13 / 7,32	42	1,10 / 3,66	3,13 / 3,66	DOWN	RUNNING
4	3,63 / 7,32	49	1,60 / 3,66	3,13 / 3,66	DOWN	RUNNING
5	4,13 / 7,32	56	1,60 / 3,66	3,63 / 3,66	DOWN	RUNNING
6	4,63 / 7,32	63	2,10 / 3,66	3,63 / 3,66	DOWN	RUNNING
7	5,13 / 7,32	70	2,60 / 3,66	3,63 / 3,66	DOWN	RUNNING
8	5,63 / 7,32	76	3,10 / 3,66	3,63 / 3,66	DOWN	RUNNING
9	6,13 / 7,32	83	2,50 / 3,66	3,63 / 3,66	DOWN	RUNNING
10	6,63 / 8,96	73	3,00 / 3,66	3,63 / 3,66	0,1 / 1,64	RUNNING
11	7,13 / 8,96	79	3,50 / 3,66	3,63 / 3,66	0,1 / 1,64	RUNNING
12	7,63 / 8,96	85	3,50 / 3,66	3,63 / 3,66	0,6 / 1,64	RUNNING

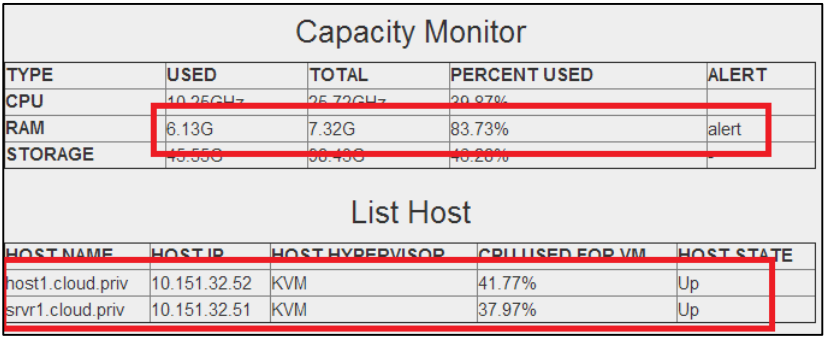
JUMLAH TENANT	PENGGUNAAN RAM TOTAL		PENGGUNAAN RAM PADA SETIAP HOST (GB)			STATUS TENANT
	(GB)	(%)	SRVR1	HOST1	HOST2	

KETERANGAN :

: **Fungsi tambah komputasi dijalankan untuk menambah perangkat komputasi**

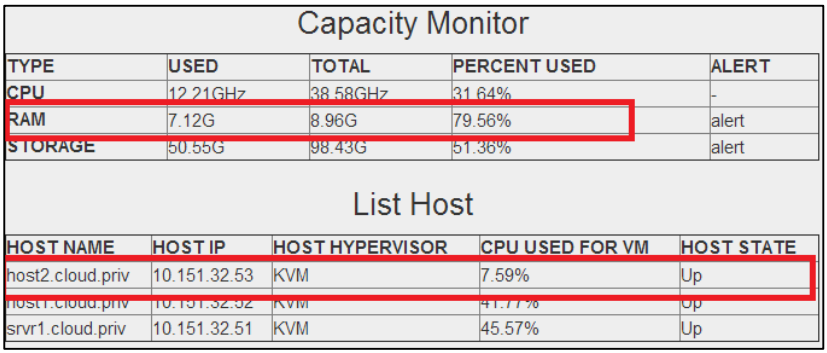
: **Perangkat komputasi telah ditambahkan, terdapat *host* baru dengan nama *host* Host2.cloud.priv**

Gambar 5.20 merupakan kondisi sistem pada saat komputasi awan masih menggunakan dua perangkat, hal ini menunjukkan bahwa kapasitas komputasi berada pada kondisi normal. Ketika kapasitas komputasi melebihi batas, maka sistem secara otomatis melakukan proses tambah komputasi untuk menambah perangkat.



Gambar 5.20. Kondisi Awal Sistem Komputasi Awan

Pada Gambar 5.21 menunjukkan bahwa sistem menggunakan tiga perangkat komputasi. Tampak perangkat baru dengan nama host2.cloud.priv tergabung dengan infrastruktur komputasi awan.



Capacity Monitor

TYPE	USED	TOTAL	PERCENT USED	ALERT
CPU	12.21GHz	38.58GHz	31.64%	-
RAM	7.12G	8.96G	79.56%	alert
STORAGE	50.55G	98.43G	51.36%	alert

List Host

HOST NAME	HOST IP	HOST HYPERVISOR	CPU USED FOR VM	HOST STATE
host2.cloud.priv	10.151.32.53	KVM	7.59%	Up
host1.cloud.priv	10.151.32.52	KVM	41.77%	Up
srvr1.cloud.priv	10.151.32.51	KVM	45.57%	Up

Gambar 5.21. Hasil Menjalankan Fungsi Tambah Komputasi

5.2.2. Pengujian Performa

Pengujian performa sistem merupakan pengujian untuk menguji apakah performa sistem dapat diandalkan. Uji coba ini dilakukan untuk mengetahui berapa waktu respon sistem ketika banyak *tenant* melakukan *request* sebuah aplikasi pada sistem secara bersamaan.

5.2.2.1. Pengujian Performa Waktu Respon pada Sistem

Pengujian performa waktu respon pada sistem dilakukan untuk mengukur kemampuan sistem dalam menanggapi *request* dari *tenant* ketika *tenant* melakukan *request* aplikasi pada portal layanan secara bersamaan. Untuk menghitung kecepatan respon sistem dalam menanggapi *request*, diperlukan jumlah *tenant* yang melakukan

request, waktu mulai *request*, dan waktu sistem selesai mengolah *request*.

Tabel 5.12 merupakan sebuah contoh uji coba menghitung rata-rata waktu yang dibutuhkan untuk mengolah permintaan setiap *tenant*. Adapun jumlah pengujian dilakukan sebanyak sepuluh kali dengan penambahan satu *tenant* setiap pengujian yang dilakukan.

Tabel 5.12. Menghitung Selisih Waktu Pengolahan *Request Tenant*

Banyak *Tenant*	ID *Tenant*	Waktu Mulai *Request* (h:m:s)	Waktu Sistem Selesai Mengolah (h:m:s)	Selisih Waktu (s)
1	A	16:50:51:8356	16:50:56:8630	5
Rata-Rata Waktu Pengolahan *Request* 1 *Tenant*				**5**
2	A	16:52:10:1257	16:52:21:3837	11
	B	16:52:10:1257	16:52:22:2472	12
Rata-Rata Waktu Pengolahan *Request* 2 *Tenant*				**11.5**
3	A	16:56:47:5094	16:57:02:6025	15
	B	16:56:47:5094	16:57:00:8296	13
	C	16:56:47:5094	16:57:02:5583	15
Rata-Rata Waktu Pengolahan *Request* 3 *Tenant*				**14.3**
4	A	17:23:21:6722	17:23:44:5014	23
	B	17:23:21:6722	17:23:44:5546	23
	C	17:23:21:6722	17:23:44:7233	22
	D	17:23:21:6722	17:23:44:7799	22
Rata-Rata Waktu Pengolahan *Request* 4 *Tenant*				**22.5**
5	A	17:32:48:8066	17:33:14:8453	26
	B	17:32:48:8066	17:33:17:8205	25
	C	17:32:48:8066	17:33:19:7275	31
	D	17:32:48:8066	17:33:14:3848	26
	E	17:32:48:8066	17:33:17:9291	29
Rata-Rata Waktu Pengolahan *Request* 5 *Tenant*				**27.4**
6	A	17:39:13:8293	17:39:47:8102	34
	B	17:39:13:8293	17:39:48:2931	35
	C	17:39:13:8293	17:39:49:8834	36
	D	17:39:13:8293	17:39:45:6826	32

	E	17:39:13:8293	17:39:48:2360	35
	F	17:39:13:8293	17:39:47:8107	34
Rata-Rata Waktu Pengolahan *Request* 6 *Tenant*				**34.3**
7	A	17:59:23:5421	18:00:01:1434	38
	B	17:59:23:5421	18:00:00:3567	37
	C	17:59:23:5421	18:00:04:3489	41
	D	17:59:23:5421	18:00:02:7348	39
	E	17:59:23:5421	17:59:59:9847	36
	F	17:59:23:5421	18:00:01:5557	38
	G	17:59:23:5421	18:00:00:6574	37
Rata-Rata Waktu Pengolahan *Request* 7 *Tenant*				**38**
8	A	18:30:12:6301	18:30:57:4563	45
	B	18:30:12:6301	18:30:55:7631	43
	C	18:30:12:6301	18:30:58:2631	46
	D	18:30:12:6301	18:31:00:1867	48
	E	18:30:12:6301	18:30:56:2731	44
	F	18:30:12:6301	18:30:55:9631	43
	G	18:30:12:6301	18:30:57:8731	45
	H	18:30:12:6301	18:30:58:7637	46
Rata-Rata Waktu Pengolahan *Request* 8 *Tenant*				**45**
9	A	18:40:01:4535	18:40:56:8270	55
	B	18:40:01:4535	18:40:58:3214	57
	C	18:40:01:4535	18:41:00:2134	59
	D	18:40:01:4535	18:40:55:4636	54
	E	18:40:01:4535	18:40:58:2370	57
	F	18:40:01:4535	18:40:59:7220	58
	G	18:40:01:4535	18:40:55:6453	54
	H	18:40:01:4535	18:40:52:1082	51
	I	18:40:01:4535	18:41:01:2304	60
Rata-Rata Waktu Pengolahan *Request* 9 *Tenant*				**56**
10	A	19:10:10:3432	19:11:13:8927	63
	B	19:10:10:3432	19:11:15:2152	65
	C	19:10:10:3432	19:11:16:2351	66
	D	19:10:10:3432	19:11:12:3512	62
	E	19:10:10:3432	19:11:17:6457	67
	F	19:10:10:3432	19:11:15:7061	65
	G	19:10:10:3432	19:11:13:1432	63

	H	19:10:10:3432	19:11:17:6571	67
	I	19:10:10:3432	19:11:19:7563	69
	J	19:10:10:3432	19:11:17:5491	67
Rata-Rata Waktu Pengolahan *Request* 10 *Tenant*				**65.4**

5.3. Evaluasi Hasil Uji Coba

Pada Evaluasi hasil uji coba, dilakukan pemaparan hasil uji coba yang telah dilakukan sebelumnya, yaitu uji coba fungsionalitas dan uji coba performa. Evaluasi ini akan dijelaskan pada subbab berikut.

5.3.1. Evaluasi Hasil Uji Coba Fungsionalitas

Sesuai dengan uji coba fungsionalitas yang telah dilakukan sebelumnya terhadap kinerja sistem, maka dapat dibuat kesimpulan keberhasilan uji coba sebagaimana ditunjukkan pada Tabel 5.13.

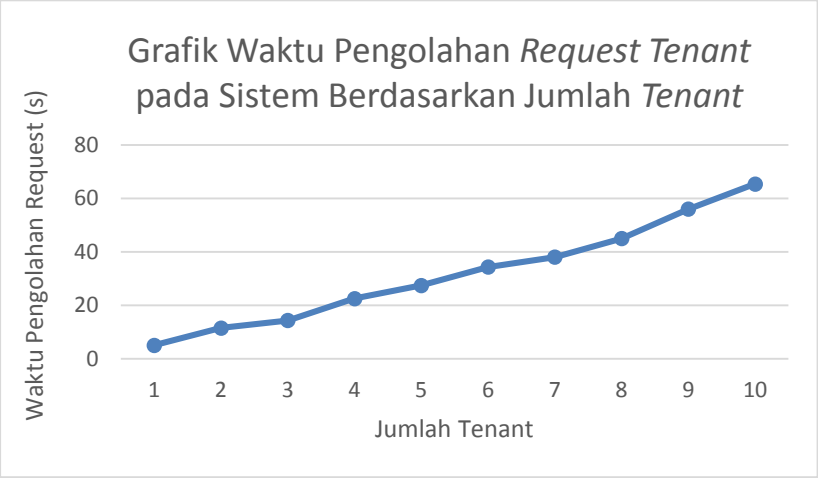
Tabel 5.13. Hasil Uji Coba Fungsionalitas

No	ID Uji Coba	Nama Uji Coba	Hasil Uji Coba
1.	UJ-01	*Generate* VM	Berhasil
2.	UJ-02	Akses Aplikasi	Berhasil
3.	UJ-03	Ambil Data Aplikasi *Tenant*	Berhasil
4.	UJ-04	Sistem *Monitoring* Aplikasi	Berhasil
5.	UJ-05	Sistem *Monitoring* Infrastruktur Komputasi Awan	Berhasil
6.	UJ-06	Tambah Komputasi	Berhasil

Pada Tabel 5.11 menyimpulkan bahwa semua fungsionalitas yang diusulkan pada sistem ini berhasil dijalankan.

5.3.2. Evaluasi Hasil Uji Coba Performa

Sesuai hasil uji coba pengujian performa sistem yang terdapat pada Tabel 5.12, maka didapat data waktu pengolahan *request tenant* yang merupakan selisih waktu saat *tenant* melakukan *request* hingga *request* tersebut selesai diolah oleh sistem, sehingga dari setiap data waktu tersebut dapat dirata-rata berdasarkan jumlah *tenant* yang melakukan *request*. Gambar 5.22. merupakan grafik waktu pengolahan *request tenant* pada sistem berdasarkan jumlah *tenant* yang melakukan *request*. Dapat disimpulkan bahwa seiring meningkatnya jumlah *tenant* yang melakukan *request* aplikasi pada portal layanan secara bersamaan, menyebabkan waktu pengolahan *request* aplikasi menjadi semakin lama dikarenakan beban kerja yang semakin berat.



Gambar 5.22. Grafik Waktu Pengolahan *Request Tenant* pada Sistem Berdasarkan Jumlah *Tenant*

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini dibahas mengenai kesimpulan yang dapat diambil dari perancangan sistem, implementasi, hingga dengan hasil pengujian selama pengerjaan Tugas Akhir. Pada bab ini juga dapat menjawab pertanyaan yang dijabarkan pada Bab 1.

Pembuatan Tugas Akhir ini pasti memiliki beberapa kelebihan dan kekurangan dari hasil yang telah dicapai dari pembuatan sistem. Semua kelebihan dan kekurangan Tugas Akhir ini juga akan dijabarkan pada bab ini. Untuk memperbaiki semua kelebihan dan kekurangan dari sistem, akan dijelaskan pada subbab saran.

6.1. Kesimpulan

Dari hasil selama proses perancangan, implementasi, serta pengujian yang dilakukan selama pengerjaan Tugas Akhir ini, dapat diperoleh kesimpulan sebagai berikut.

1. Sistem dapat berjalan di atas infrastruktur komputasi awan berbasis CloudStack.
2. Sistem dapat menyediakan aplikasi manajemen restoran secara *multi-tenant* pada infrastruktur komputasi awan dengan struktur aplikasi dan *database* yang sama pada semua *tenant*.
3. Sistem dapat melakukan *monitoring* dan kontrol komputasi awan baik secara infrastruktur dan aplikasi setiap *tenant*.
4. Sistem mampu menambahkan perangkat komputasi secara otomatis ketika penggunaan sistem melebihi batas pemakaian.

6.2. Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Diantaranya adalah sebagai berikut:

1. *Multitenancy* pada sistem ini berbasis virtualisasi, di mana *tenant* menggunakan VM yang berbeda dengan *tenant* lainnya. Untuk pengembangan selanjutnya sistem diharapkan dapat menerapkan *multitenancy* pada tingkat aplikasi, sehingga lebih meningkatkan efektifitas dan efisiensi kinerja sumber daya perangkat komputasi.
2. Sistem dapat memberikan sebuah VM dengan kapasitas dan komputasi dinamis sesuai dengan penggunaan setiap *tenant*.
3. Memberikan variasi aplikasi sesuai dengan kebutuhan *tenant* yang meminta.

DAFTAR PUSTAKA

[1] A. Budiyanto, "Pengantar Cloud Computing," Mei 2012. [Online]. Available: http://www.cloudindonesia.or.id/wp-content/uploads/2012/05/E-Book-Pengantar-Cloud-Computing-R1.pdf. [Diakses 10 Maret 2014].

[2] "A Virtual Dedicated *Server* Hosting Alternative: Cloud Computing," 2014. [Online]. Available: https://www.profitbricks.com/dedicated-*server*-vs-cloud. [Diakses 10 Februari 2014].

[3] 2013. [Online]. Available: http://cloudstack.apache.org/about.html. [Diakses 15 Maret 2014].

[4] N. Sabharwal dan R. Shankar, Apache CloudStack Cloud Computing, 1st penyunt., S. Balan dan L. Chanana, Penyunt., Birmingham: Packt Publishing Ltd, 2013, pp. 7-10.

[5] N. Sabharwal dan R. Shankar, Apache CloudStack Cloud Computing, 1st penyunt., S. Balan dan L. Chanana, Penyunt., Birmingham: Packt Publishing, 2013, p. 17.

[6] OpenStack, "Chapter 3. OpenStack Projects, History, and Releases Overview," OpenStack, July 2011. [Online]. Available: http://docs.openstack.org/training-guides/content/module001-ch003-core-projects.html. [Diakses 27 Juni 2014].

[7] OpenNebula, "An Overview of OpenNebula 4.4," 2002. [Online]. Available: http://archives.opennebula.org/documentation:rel4.4:intro. [Diakses 27 Juni 2014].

[8] 2010. [Online]. Available: http://json.org/json-id.html. [Diakses 27 Juni 2014].

[9] 2014. [Online]. Available: http://www.centosblog.com/what-is-centos/.

[10] L. Kun dan A. Williams, “What is NFS?,” April 2005. [Online]. Available: http://searchenterprisedesktop.techtarget.com/definition/Network-File-System. [Diakses 10 Juni 2014].

[11] C. Jansen, 2014. [Online]. Available: http://www.techopedia.com/definition/4790/Hypervisor. [Diakses 15 Maret 2014].

[12] KVM, “Main Page - KVM,” 2010. [Online]. Available: http://www.linux-kvm.org/page/Main_Page. [Diakses 22 Mei 2014].

[13] Libvirt, “What is libvirt?,” 2010. [Online]. Available: http://wiki.libvirt.org/page/FAQ. [Diakses 28 Mei 2014].

[14] W3C, “Extensible Markup Language,” 29 Oktober 2013. [Online]. Available: http://www.w3.org/XML/. [Diakses 1 Juli 2014].

[15] PHP, “PHP: General Information - Manual,” 2014. [Online]. Available: http://id1.php.net/manual/en/faq.general.php. [Diakses 22 Mei 2014].

[16] W3C, “Webservice Glossary,” 11 February 2004. [Online]. Available: http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice. [Diakses 22 Mei 2014].

[17] Microsoft, “Monitor System,” 22 Agustus 2005. [Online]. Available: http://technet.microsoft.com/en-us/library/cc787699(v=ws.10).aspx. [Diakses 22 Mei 2014].

[18] 2014. [Online]. Available: http://archives.opennebula.org/documentation:archives:rel2.0:architecture.

[19] N. Sabharwal dan R. Shankar, Apache CloudStack Cloud Computing, 1st penyunt., S. Balan dan L. Chanana, Penyunt., Birmingham: Packt Publishing, 2013, p. 25.

[20] N. Sabharwal dan R. Shankar, Apache CloudStack Cloud Computing, 1st penyunt., S. Balan dan L. Chanana, Penyunt., Birmingham: Packt Publishing, 2013, p. 9.

[21] OpenStack, “Introduction to OpenStack,” 2013. [Online]. Available: http://docs.openstack.org/training-guides/content/module001-ch004-openstack-architecture.html. [Diakses 10 Februari 2014].

[22] Juni 2013. [Online]. Available: http://smilejay.com/2013/03/libvirt-introduction/. [Diakses 25 Mei 2014].

BIODATA PENULIS

Penulis, Dimas Yoga Pratama, lahir di kota Malang pada tanggal 11 April 1992. Penulis adalah anak pertama dari empat bersaudara dan dibesarkan di kota Malang, Jawa Timur.

Penulis menempuh pendidikan formal di SDN Lowokwaru VIII Malang (1998-2004), SMPN 3 Malang (2004-2007), SMAN I Malang (2007-2010). Pada tahun 2010, pennulis memulai pendidikan S1 jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur.

Di jurusan Teknik Informatika, penulis mengambil bidang minat *Net Centric Computing* dan memiliki ketertarikan di bidang Sistem Terdistribusi, Komputasi Grid, dan Teknologi Antar Jaringan. Penulis juga aktif dalam organisasi kemahasiswaan seperti Himpunan Mahasiswa Teknik Computer (HMTC). Penulis dapat dihubungi melalui alamat email smidh.d@gmail.com

LAMPIRAN IMPLEMENTASI INFRASTRUKTUR KOMPUTASI AWAN

Implementasi Perangkat Manajemen *Server*

Implementasi manajemen *server* dilakukan pada satu buah perangkat komputer. Manajemen *server* berfungsi sebagai pusat kontrol dari infrastruktur komputasi awan dalam satu *Cluster*. Diperlukan beberapa perangkat lunak pendukung dan konfigurasi agar manajemen *server* dapat berjalan dengan baik. Pemasangan manajemen *server* diharapkan pada perangkat komputer berspesifikasi tinggi. Tahap pemasangan manajemen *server* dijelaskan sebagai berikut.

1. Instalasi Sistem Operasi

Pada tahap ini instalasi sistem operasi dilakukan pada perangkat komputer sebagai dasar dari infrastruktur komputasi awan. Sistem operasi CentOS merupakan anjuran dalam pondasi infrastruktur komputasi awan untuk meminimalisir *crash* dalam lingkungan komputasi awan.

2. Konfigurasi Cloudstack *Repository*

Mengkonfigurasikan sistem operasi agar dapat mengenali repository dari CloudStack dengan *command Yum*.

a. Mengaktifkan repository *CloudStack*

```
cd /etc/yum.repos.d
vi /etc/yum.repos.d/cloudstack.repo
```

b. Tambahkan CloudStack repository pada /etc/yum.repos.d/cloudstack.repo

```
[cloudstack]
```

```
name=cloudstack
baseurl=http://cloudstack.apt-get.eu/rhel/4.3/
enabled=2
gpgcheck=0
```

3. Menyiapkan sistem operasi agar mendukung kerangka kerja CloudStack

Sebelum dilakukan instalasi Manajemen *Server* pada perangkat komputer, terlebih dahulu sistem operasi pada perangkat komputer harus dikonfigurasi sedemikian rupa sehingga dapat mendukung kerangka kerja CloudStack.

a. Cek *fully qualified hostname*

```
Hostname –fqdn
```

b. Apabila command cek hostname tidak menunjukkan *fully qualified hostname*, maka konfigurasikan agar sistem operasi menampilkan fqdn dengan menambahkan baris terakhir /etc/hosts menjadi berikut ini.

```
10.151.32.51 srvr1.cloud.priv
```

c. Install NTP diperlukan untuk mensikronisasikan clock pada *server* dalam infrastruktur komputasi awan.

```
yum install ntp
chkconfig ntpd on
service ntpd start
```

d. Set SELinux menjadi *permissive*

```
#setenforce 0
```

Untuk memastikan SELinux menjadi permissive selama sistem berjalan, diperlukan konfigurasi pada berkas /etc/selinux/config dan mengubah kondisi menjadi *permissive*.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
# targeted - Targeted processes are protected,
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

4. Instalasi NFS

Pada konfigurasi infrastruktur awan ini menggunakan teknologi NFS sebagai primary dan secondary storage. Dua NFS berbagi diperlukan agar *primary* dan *secondary storage* pada infrastruktur komputasi awan berjalan dengan baik.

```
yum install nfs-utils
```

Konfigurasi NFS diperlukan untuk menyediakan *storage*. Konfigurasi NFS dengan menambahkan baris dibawah ini pada berkas /etc/exports.

```
/secondary *(rw,async,no_root_squash)
/primary *(rw,async,no_root_squash)
Save file
mkdir /primary
mkdir /secondary
```

Merubah konfigurasi domain pada berkas /etc/idmapd.conf.

```
Domain=cloud.priv
```

menambahkan baris di bawah ini pada berkas /etc/sysconfig/nfs

```
LOCKD_TCPPORT=32803
LOCKD_UDPPORT=32769
MOUNTD_PORT=892
RQUOTAD_PORT=875
STATD_PORT=662
STATD_OUTGOING_PORT=2020
```

Melakukan konfigurasi pada *firewall* agar menyetujui koneksi NFS pada infrastruktur komputasi awan pada berkas /etc/sysconfig/iptables

```
-A INPUT -s 10.151.32.50/24 -m state --state NEW -p udp --dport 111 -j ACCEPT
-A INPUT -s 10.151.32.50/24 -m state --state NEW -p tcp --dport 111 -j ACCEPT
-A INPUT -s 10.151.32.50/24 -m state --state NEW -p tcp --dport 2049 -j ACCEPT
-A INPUT -s 10.151.32.50/24 -m state --state NEW -p tcp --dport 32803 -j ACCEPT
-A INPUT -s 10.151.32.50/24 -m state --state NEW -p udp --dport 32769 -j ACCEPT
-A INPUT -s 10.151.32.50/24 -m state --state NEW -p tcp --dport 892 -j ACCEPT
-A INPUT -s 10.151.32.50/24 -m state --state NEW -p udp --dport 892 -j ACCEPT
-A INPUT -s 10.151.32.50/24 -m state --state NEW -p tcp --dport 875 -j ACCEPT
-A INPUT -s 10.151.32.50/24 -m state --state NEW -p udp --dport 875 -j ACCEPT
-A INPUT -s 10.151.32.50/24 -m state --state NEW -p tcp --dport 662 -j ACCEPT
-A INPUT -s 10.151.32.50/24 -m state --state NEW -p udp --dport 662 -j ACCEPT
```

Menkonfigurasikan NFS untuk berjalan secara otomatis

```
service rpcbind start
service nfs start
chkconfig rpcbind on
chkconfig nfs on
```

5. Instalasi Database *Server*

CloudStack Manajemen *server* menggunakan MySQL *server* untuk menaruh semua data infrastruktur komputasi awan.

Pada konfigurasi ini, MySQL *server* dipasang secara *single node* bersamaan dengan manajemen *server*.

a. Install MySQL *server*.

```
yum –y install mysql-server
```

b. Buka file konfigurasi /etc/my.cnf.
c. Masukkan baris di bawah ini agar manajemen *server* dapat berkomunikasi dengan MySQL. Konfigurasi ini mendukung *single node* manajemen *server*.

```
innodb_rollback_on_timeout=1
innodb_lock_wait_timeout=600
max_connections=350
log-bin=mysql-bin
binlog-format = 'ROW'
exit file
service mysqld start
chkconfig mysqld on
```

6. Instalasi Manajemen *server*

Setelah semua langkah selesai dilakukan, maka instalasi manajemen *server* dapat dilakukan

```
yum -y install cloudstack-management
```

instalasi *database*

```
cloudstack-setup-databases cloud:password@localhost --deploy-as=root
```

7. Instalasi Sistem VM pada manajemen *server*

CloudStack menggunakan sistem *virtual machine* untuk menyediakan fungsionalitas pada akses *virtual machine* milik *tenant*, menyediakan layanan jaringan, dan mengatur berbagai aspek

dalam penyimpanan. Dalam infrastruktur awan ini, menggunakan Hypervisor KVM dalam pengaturan VM.

```
/usr/share/cloudstack-common/scripts/storage/secondary/cloud-install-sys-tmplt -m /secondary -u http://download.cloud.com/templates/4.3/systemvm64template-2014-01-14-master-kvm.qcow2.bz2 -h kvm -F
```

8. Mengatur zona pada infrastruktur CloudStack

Zona merupakan entitas terbesar pada CloudStack, diperlukan konfigurasi zona pada setiap infrastruktur awan yang dibangun

```
Name = ZoneNCC
Public DNS 1 = 202.46.129.2 // DNS ITS
Public DNS 2 = 10.151.32.6 // DNS NCC
Internal DNS1 = 202.46.129.2 // DNS ITS
Internal DNS2 = 10.151.32.6 // DNS NCC
```

9. Mengatur POD pada infrastruktur CloudStack

```
Name = PodNCC
Gateway = 10.151.32.1 // Gateway NCC
Netmask = 255.255.255.0
Reserved IP = 10.151.32.60-10.151.32.80 // Pengalokasian IP pada sistem VM
Guest gateway = 10.151.32.1 // Gateway NCC
Guest netmask = 255.255.255.0
Guest reserved IP  = 10.151.32.90-10.151.32.200
```

10. Mengatur konfigurasi Cluster dan penyimpanan NFS pada infrastruktur CloudStack

```
Name = ClusterNCC
Hypervisor = KVM //Hypervisor yang digunakan dalam infrastruktur komputasi awan
PrimaryStorage = PrimaryNCC
Server = 10.151.32.51
NFSPath = 10.151.32.51/primary/
SecondaryStorage = SecondaryNCC
Server = 10.151.32.51
```

```
NFSPath = 10.151.32.51/Secondary/
```

Implementasi Perangkat Agen Komputasi

Implementasi agen komputasi dilakukan pada satu buah perangkat komputer. Agen komputasi berperan sebagai wadah virtualisasi VM dalam satu *Cluster*. Diperlukan beberapa perangkat lunak pendukung dan konfigurasi agar agen komputasi dapat berjalan dengan baik. Pemasangan agen komputasi diharuskan pada perangkat komputer yang mendukung virtualisasi. Instalasi infrastruktur awan ini menggunakan Hypervisor KVM.Tahap pemasangan agen komputasi dijelaskan sebagai berikut.

1. Instalasi Sistem Operasi

Pada tahap ini instalasi sistem operasi dilakukan pada perangkat komputer sebagai dasar dari infrastruktur komputasi awan. Sistem operasi CentOS merupakan anjuran dalam pondasi infrastruktur komputasi awan untuk meminimalisir *crash* dalam lingkungan komputasi awan.

2. Konfigurasi Cloudstack *Repository*

Mengkonfigurasikan sistem operasi agar dapat mengenali repository dari CloudStack dengan *command Yum*.

a. Mengaktifkan repository *CloudStack*

```
cd /etc/yum.repos.d
vi /etc/yum.repos.d/cloudstack.repo
```

b. Tambahkan CloudStack repository pada /etc/yum.repos.d/cloudstack.repo\

```
[cloudstack]
```

```
name=cloudstack
baseurl=http://cloudstack.apt-get.eu/rhel/4.3/
enabled=2
gpgcheck=0
```

3. Menyiapkan sistem operasi agar mendukung kerangka kerja CloudStack

Sebelum dilakukan instalasi Manajemen *Server* pada perangkat komputer, terlebih dahulu sistem operasi pada perangkat komputer harus dikonfigurasi sedemikian rupa sehingga dapat mendukung kerangka kerja CloudStack.

a. Cek *fully qualified hostname*

```
Hostname –fqdn
```

b. Apabila command cek hostname tidak menunjukkan *fully qualified hostname*, maka konfigurasikan agar sistem operasi menampilkan fqdn dengan menambah baris terakhir /etc/hosts menjadi berikut ini.

```
10.151.32.51 srvr1.cloud.priv
```

c. Install NTP diperlukan untuk mensikronisasikan clock pada *server* dalam infrastruktur komputasi awan.

```
yum install ntp
chkconfig ntpd on
service ntpd start
```

d. Set SELinux menjadi *permissive*

```
#setenforce 0
```

Untuk memastikan SELinux menjadi permissive selama sistem berjalan, diperlukan konfigurasi pada berkas /etc/selinux/config dan mengubah kondisi menjadi *permissive*.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
# targeted - Targeted processes are protected,
# mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

4. Konfigurasi Agen Komputasi

Untuk mengatur *instance* dalam sebuah *host* CloudStack menggunakan agen. Agen ini berkomunikasi dengan manajemen *server* dan mengontrol semua *instance* pada *host*

```
yum install cloudstack-agent
```

5. Instalasi Libvirt

CloudStack menggunakan Libvirt sebagai pengatur VM, konfigurasi diperlukan agar Libvirt dapat berjalan dengan baik pada perangkat agen komputasi.

a. Untuk mengaktifkan *live migration* pada infrastruktur komputasi awan, libvirt harus dikonfigurasikan untuk menerima koneksi TCP. Konfigurasi ini terdapat pada berkas /etc/libvirt/libvirtd.conf

```
listen_tls = 0
listen_tcp = 1
tcp_port = "16509"
auth_tcp = "none"
```

```
mdns_adv = 0
```

b. Menambahkan sintaks berikut ini pada /etc/sysconfig/libvirtd

```
LIBVIRTD_ARGS="--listen"
```

c. Restart Libvirt

```
Service libvirtd restart
Chkconfig libvirtd on
```

6. Konfigurasi QEMU

Tambahkan baris pada berkas /etc/libvirt/qemu .conf

```
vnc_listen=0.0.0.0
```

7. Konfigurasi *Network Bridges*

Untuk meneruskan lalu-lintas data pada *instance* sedikitnya membutuhkan dua *bridge*: *public* dan *private.* Konfigurasi *bridge* ini diharuskan ada pada setiap Hypervisor.

Pada instalasi agen komputasi awan, perangkat komputer menggunakan satu NIC (eth0) dengan tiga VLAN dengan keterangan sebagai berikut.

a. VLAN 100 untuk manajemen dari Hypervisor.
b. VLAN 200 untuk jaringan publik dari *instance* (cloudbr0).
c. VLAN 300 untuk jaringan *private* dari *instance* (cloudbr1)

VLAN 100 diberikan IP 10.151.32.52/24 (*host IP*) dengan gateway 10.151.32.1. Berikut langkah konfigurasi.

a. Konfigurasi eth0 pada perangkat agen komputasi.

```
vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
HWADDR=00:04:xx:xx:xx:xx
```

```
ONBOOT=yes
HOTPLUG=no
BOOTPROTO=none
TYPE=Ethernet
```

b. Konfigurasi interface pada VLAN100 (manajemen Hypervisor)

```
vi /etc/sysconfig/network-scripts/ifcfg-eth0.100
DEVICE=eth0.100
HWADDR=00:04:xx:xx:xx:xx
ONBOOT=yes
HOTPLUG=no
BOOTPROTO=none
TYPE=Ethernet
VLAN=yes
IPADDR=10.151.32.52
GATEWAY=10.151.32.52
NETMASK=255.255.255.0
```

c. Konfigurasi interface pada VLAN200 (jaringan publik)

```
vi /etc/sysconfig/network-scripts/ifcfg-eth0.200
DEVICE=eth0.200
HWADDR=00:04:xx:xx:xx:xx
ONBOOT=yes
HOTPLUG=no
BOOTPROTO=none
TYPE=Ethernet
VLAN=yes
BRIDGE=cloudbr0
```

d. Konfigurasi interface pada VLAN300 (jaringan *private*)

```
vi /etc/sysconfig/network-scripts/ifcfg-eth0.300
DEVICE=eth0.300
HWADDR=00:04:xx:xx:xx:xx
ONBOOT=yes
HOTPLUG=no
BOOTPROTO=none
TYPE=Ethernet
```

```
VLAN=yes
BRIDGE=cloudbr1
```

e. Menambahkan *Bridge* di atas VLAN interface (cloudbr0)

```
vi /etc/sysconfig/network-scripts/ifcfg-cloudbr0
DEVICE=cloudbr0
TYPE=Bridge
ONBOOT=yes
BOOTPROTO=none
IPV6INIT=no
IPV6_AUTOCONF=no
DELAY=5
STP=yes
```

f. Menambahkan Bridge cloudbr1 yang digunakan sebagai jaringan *private*

```
vi /etc/sysconfig/network-scripts/ifcfg-cloudbr1
DEVICE=cloudbr1
TYPE=Bridge
ONBOOT=yes
BOOTPROTO=none
IPV6INIT=no
IPV6_AUTOCONF=no
DELAY=5
STP=yes
```

g. Restart network yang telah dikonfigurasikan

```
Service network restart
```

8. Konfigurasi *Firewall*

Membuka ports yang dibutuhkan manajemen *server* untuk berkomunikasi dengan agen komputasi.

```
iptables -I INPUT -p tcp -m tcp --dport 22 -j ACCEPT
iptables -I INPUT -p tcp -m tcp --dport 1798 -j ACCEPT
iptables -I INPUT -p tcp -m tcp --dport 16509 -j ACCEPT
iptables -I INPUT -p tcp -m tcp --dport 5900:6100 -j ACCEPT
```

```
iptables -I INPUT -p tcp -m tcp --dport 49152:49216 -j ACCEPT
iptables-save > /etc/sysconfig/iptables
```

9. Tambah Agen komputasi yang telah dikonfigurasi.

```
Hostname = 10.151.32.52
Username = root
Password = dimas92
```