**SISTEM MANAJEMEN LAYANAN WEB BERBASIS PLATFORM AS A SERVICE (PAAS) DENGAN API OPENSTACK**

**SKRIPSI**



**IDA BAGUS RATHU EKA SURYA WIBAWA**

**NIM. 1308605045**

**JURUSAN ILMU KOMPUTER**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS UDAYANA**

**BUKIT JIMBARAN**

**2018**

# LEMBAR PENGESAHAN TUGAS AKHIR

|  |  |  |
| --- | --- | --- |
| Judul | : | Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack |
| Nama | : | Ida Bagus Rathu Eka Surya Wibawa |
| NIM | : | 1308605045 |
| Tanggal Seminar | : | 6 Oktober 2018 |

Disetujui oleh:

|  |  |
| --- | --- |
| Pembimbing I | Penguji I |
|  |  |
| I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs.  NIP. 198012062006041003 | I Gede Santi Astawa, S.T., M.Cs.  NIP. 198012062006041003 |
| Pembimbing II | Penguji II |
|  |  |
| I Komang Ari Mogi, S.Kom., M.Kom.  NIP. 198409242008011007 | Gst. Ayu Vida Mastrika Giri, S.Kom., M.Cs.  NIP. 1990060620160322001 |
|  | Penguji III |
|  |  |
|  | Made Agung Raharja,S.Si.,M.Cs  NIK. 1985091920130122003 |
| Mengetahui,  Ketua Jurusan Ilmu Komputer  FMIPA Universitas Udayana  Ketua,  Dr. I Ketut Gede Suhartana, S.Kom., M.Kom  NIP. 197201102008121001 | |

# SURAT PERNYATAAN KEASLIAN KARYA ILMIAH

Yang bertanda tangan di bawah ini menyatakan bahwa naskah Skripsi dengan judul:

**SISTEM MANAJEMEN LAYANAN WEB BERBASIS PLATFORM AS A SERVICE (PAAS) DENGAN API OPENSTACK**

Nama : Ida Bagus Rathu Eka Surya Wibawa

NIM : 1308605045

Program Studi : Teknik Informatika

E-mail : eka.suryawibawa@mhs.cs.unud.ac.id

Nomor telp/HP : 083119171966

Alamat : Jalan Pratu Made Rambug Banjar Sasih No. 49 Sukawati, Gianyar, Bali

Belum pernah dipublikasikan dalam dokumen skripsi, jurnal nasional maupun internasional atau dalam prosiding manapun, dan tidak sedang atau akan diajukan untuk publikasi di jurnal atau prosiding manapun. Apabila di kemudian hari terbukti terdapat pelanggaran kaidah – kaidah akademik pada karya ilmiah saya, maka saya bersedia menanggung sanksi-sanksi yang dijatuhkan karena kesalahan tersebut, sebagaimana diatur oleh Peraturan Menteri Pendidikan Nasional Nomor 17 Tahun 2010 tentang Pencegahan dan Penanggulangan Plagiat di Perguruan Tinggi.

Demikian Surat Pernyataan ini saya buat dengan sesungguhnya untuk dapat dipergunakan bilamana diperlukan.

Denpasar, 12 Juni 2018

Yang membuat pernyataan,

(Ida Bagus Rathu Eka Surya Wibawa)

NIM. 1308605045

|  |  |  |
| --- | --- | --- |
| Judul | : | Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack |
| Nama | : | Ida Bagus Rathu Eka Surya Wibawa |
| NIM | : | 1308605045 |
| Pembimbing I | : | I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs. |
| Pembimbing II | : | I Komang Ari Mogi, S.Kom., M.Kom. |

# ABSTRAK

**Kata kunci:**

|  |  |  |
| --- | --- | --- |
| Judul | : | Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack |
| Nama | : | Ida Bagus Rathu Eka Surya Wibawa |
| NIM | : | 1308605045 |
| Pembimbing I | : | I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs. |
| Pembimbing II | : | I Komang Ari Mogi, S.Kom., M.Kom. |

# ABSTRACT

**Kata kunci:**

# KATA PENGANTAR

Puji syukur peneliti panjatkan kehadapan Tuhan Yang Maha Esa, karena berkat rahmat dan karunia-Nya, Proposal Tugas Akhir dengan judul “Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack” dapat diselesaikan dengan baik dan tepat waktu. Proposal ini diharapkan dapat menjadi pedoman dan arahan dalam melaksanakan penelitian. Peneliti mengucapkan terima kasih kepada berbagai pihak yang telah membantu dalam pembuatan proposal ini, antara lain:

1. Bapak I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs. sebagai pembimbing 1 yang telah mengkritisi, membimbing, dan menyempurnakan proposal tugas akhir ini.
2. Bapak I Komang Ari Mogi, S.Kom., M.Kom. sebagai pembimbing 2 yang telah membimbing, dan menyempurnakan proposa tugas akhir ini.
3. Bapak Agus Muliantara, S.Kom., M.Kom. selaku Ketua Jurusan Ilmu Komputer yang telah banyak memberikan masukan dalam proses pembuatan proposal tugas akhir ini.
4. Bapak/Ibu dosen di Jurusan Ilmu Komputer, yang telah meluangkan waktu turut memberikan saran dan masukan dalam penyempurnaan proposal tugas akhir ini.
5. Semua pihak yang telah memberi dukungan sehingga proposal ini dapat diselesaikan sesuai dengan waktu yang ditentukan.

Disadari pula bahwa sudah tentu laporan ini masih mengandung kelemahan dan kekurangan. Maka dari pada itu masukan dan saran penyempurnaan sangat diharapkan.

Bukit Jimbaran, Oktober 2017

Penyusun

(Ida Bagus Rathu Eka Surya Wibawa)

# DAFTAR ISI

[PROPOSAL TUGAS AKHIR i](#_Toc496729558)

[LEMBAR PENGESAHAN TUGAS AKHIR ii](#_Toc496729559)

[KATA PENGANTAR iii](#_Toc496729560)

[DAFTAR ISI iv](#_Toc496729561)

[DAFTAR GAMBAR vi](#_Toc496729562)

[DAFTAR TABEL vii](#_Toc496729563)

[1. Latar Belakang 1](#_Toc496729564)

[2. Rumusan Masalah 2](#_Toc496729565)

[3. Tujuan Penelitian 2](#_Toc496729566)

[4. Batasan Masalah 2](#_Toc496729567)

[5. Manfaat Penelitian 3](#_Toc496729568)

[6. Tinjauan Pustaka 3](#_Toc496729569)

[6.1. Tinjauan Studi 3](#_Toc496729570)

[6.2. Cloud Computing 6](#_Toc496729571)

[6.2.1. Jenis Cloud Computing 6](#_Toc496729572)

[6.3. Openstack 8](#_Toc496729573)

[6.4. Sistem Administrator 8](#_Toc496729574)

[6.5. Python 9](#_Toc496729575)

[6.6. RESTful API 9](#_Toc496729576)

[6.7. Backup 10](#_Toc496729577)

[6.8. Ansible 11](#_Toc496729578)

[6.9. Algoritma Threshold Adaptif 11](#_Toc496729579)

[6.10. Teknik Pengujian Perangkat Lunak 13](#_Toc496729580)

[6.10.1. BlackBox Testing 13](#_Toc496729581)

[6.10.2. Performance Testing 14](#_Toc496729582)

[7. Metode Penelitian 15](#_Toc496729583)

[7.1. Analisis Kebutuhan 15](#_Toc496729584)

[7.2. Kerangka Kerja Penelitian 15](#_Toc496729585)

[7.3. Perancangan Sistem 17](#_Toc496729586)

[7.3.1. Desain Arsitektur Kerja Sistem 17](#_Toc496729587)

[7.4. Flowchart Sistem 19](#_Toc496729588)

[7.4.1. Flowchart Konfigurasi Otomatis 20](#_Toc496729589)

[7.4.2. Flowchart Front-end Web 21](#_Toc496729590)

[7.5. Rancangan ERD 22](#_Toc496729591)

[7.6. Evaluasi Perancangan Sistem 23](#_Toc496729592)

[8. Jadwal Pelaksanaan Penelitian 23](#_Toc496729593)

[DAFTAR PUSTAKA 24](#_Toc496729594)

# DAFTAR TABEL

[Tabel 6.1 Tinjauan Studi 3](#_Toc496729600)

[Tabel 6.2 Tabel Pengujian Black Box 14](#_Toc496729601)

[Tabel 8.1 Rancangan Jadwal Pelaksanaan Penelitian 23](#_Toc496729602)

# DAFTAR GAMBAR

[Gambar 6.10.2.1 Kerangka Kerja Sistem 16](#_Toc496729595)

[Gambar 7.3.1.1 Desain Kerja Sistem 18](#_Toc496729596)

[Gambar 7.4.1.1 Flowchart Konfigurasi Otomatis 20](#_Toc496729597)

[Gambar 7.4.2.1 Flowchart Front-end Web 21](#_Toc496729598)

[Gambar 7.4.2.1 Entity Relationship Diagram Sistem 22](#_Toc496729599)

# DAFTAR LAMPIRAN

# BAB I

**PENDAHULUAN**

# Latar Belakang

Cloud computing merupakan teknologi yang saat ini mulai berkembang dalam banyak aktivitas teknologi informasi. Cloud computing merupakan model komputasi yang semua sumber daya yang ada dalam layanan cloud dijalankan dengan media jaringan internet. Dengan adanya cloud computing memudahkan para pengguna dalam melakukan komputasi tanpa harus melakukan instalasi aplikasi pada komputer, pengguna hanya perlu mengaksesnya melalui internet. Cloud computing memiliki beberapa fasilitas yang dapat dipilih oleh pengguna sesuai kebutuhan pengguna seperti Infrastructure as a Service(IaaS) sebagai penyedia infrastruktur pada pengguna, Platform as a Service yang dapat digunakan oleh pengembang aplikasi untuk mengembangkan aplikasi yang akan dibuat tanpa perlu menyediakan infrastruktur, database, framework aplikasi dan lain sebagainya. Serta Software as a Service(PaaS) yang memberikan software yang siap digunakan oleh pengguna.

Dalam layanan cloud dikelola oleh seorang sistem administrator atau sysadmin yang bertugas menginstalasi dan menkonfigurasi *server*, *menginstall* dan mengkonfigurasi *software* aplikasi, membuat dan mengelola *user*, *backup* dan *restore* file, konfigurasi keamanan server, serta memonitor keamanan jaringan agar layanan yang dibutuhkan dapat berjalan dengan baik. Permasalahan yang muncul adalah untuk melakukan hal tersebut *sysadmin* harus mengkonfirmasi kebutuhan pengguna, serta untuk melakukan instalasi dan konfigurasi pada virtual server membutuhkan waktu yang cukup lama. Melakukan backup pada banyak pengguna *server* virtual dapat menyita banyak waktu dari seorang sistem administrator karena harus mengelola banyak server virtual pengguna. Performa kinerja *server* utama cloud menjadi sebuah hal yang penting pula, bila *server* utama penyimpanan data pengguna mengalami masalah maka virtual mesin yang ada pada server utama mengalami masalah pula. Sehinga sangat penting menjaga performa dan kondisi server cloud.

Dari masalah tersebut penelitian ini diharapkan dapat mempermudah tugas – tugas dari sistem administrator dengan sistem manajemen berbasis cloud yang dibangun menggunakan API dari openstack dan flask python, sistem administrator dapat mendukung kinerja seorang sistem administrator mengelola setiap virtual server yang dimiliki. Serta melakukan instalasi dan konfigurasi secara otomatis bila ada pengguna baru yang membutuhkan sebuah server virtual untuk website, melakukan backup sistem pengguna secara berkala untuk menjaga data para user bila terjadi sesuatu yang tidak diinginkan serta sistem menerapkan metode threshold adaptif untuk mengoptimalkan kinerja resource sehingga sistem mengetahui tingkat kinerja yang optimal dalam mengelola virtual mesin yang ada.

# Rumusan Masalah

Berdasarkan latar belakang, dapat dirumuskan beberapa permasalahan dalam penelitian ini yaitu:

1. Bagaimana sistem yang dibangun dapat mendukung pekerjaan sistem administrator dalam mengelola server cloud.
2. Bagaimana membangun sistem yang memfasilitasi developer dalam mengembangkan sistem yang dimili berbasis cloud.
3. Bagaimana server cloud dapat mengoptimasi performa sumber dayanya dengan metode threshold adaptif.

# Tujuan Penelitian

Penelitian ini memiliki beberapa tujuan yang ingin dicapai, berdasarkan rumusan masalah yang ditemukan, yaitu:

1. Membuat developer mengurangi beban resource dapa komputer dalam mengembangkan aplikasi yang dibuat.
2. Mendukung sistem administrator dalam mengelola server agar kinerja sistem administrator menjadi optimal.
3. Sistem administrator dapat memantau performa sumber daya yang dimiliki pada server cloud.

# Batasan Masalah

Penelitian ini memiliki beberapa batasan masalah, guna fokus untuk mencapai tujuan yang sudah dijabarkan sebelumnya, yaitu:

1. Layanan cloud menggunakan openstack sebagai penyedia layanan cloud computing.
2. Sistem melayani virtual server untuk membuat layanan aplikasi web.
3. Sistem dibangun menggunakan python dengan framework flask.
4. Sistem tidak menyediakan domain untuk layanan pengguna.

# Manfaat Penelitian

Sistem yang buat dapat mendukung pekerjaan seorang administrator untuk mengelola server utama, serta memberikan wadah bagi para developer untuk bisa mengembangkan sistemnya secara online tanpa perlu menyiapkan server. Sehingga developer hanya fokus pada pengembangan aplikasinya.

# Metode Penelitian

Bagian ini akan menjelaskan mengenai langkah – langkah yang akan dilakukan dalam merancang sistem manajemen layanan web berbasis Platform as a Service (PaaS) dengan API openstack.

## Analisis Kebutuhan

Pada tahap ini dilakukan analisis kebutuhan sistem, analisis kebutuhan sistem meliputi data yang digunakan, pembelajaran dari referensi yang sudah ada dan perangkat yang digunakan baik perangkat lunak maupun perangkat keras:

1. Tahapan pertama yang dilakukan dalam penelitian ini adalah mengidentifikasi permasalahan. Tahap ini merupakan tahap yang paling penting dalam penelitian karena jalannya penelitian didasarkan atas permasalahan yang terjadi. Setelah menentukan masalah yang terjadi, tahapan yang diperlukan selanjutnya adalah menentukan rumusan masalah dan tujuan yang ingin dicapai dalam penenlitian. Pada penelitian ini identifikasi permasalahan dilakukan dengan menggunakan teknik observasi, dari teknik ini maka akan dapat diketahui mengenai keluhan – keluhan yang ada di lapangan.
2. Tahap kedua yang dilakukan dalam metodelogi penelitian ini adalah studi literatur. Studi literatur dilakukan dengan mengambil literatur – literatur pendukung dari jurnal – jurnal ilmiah, baik jurnal dalam negeri ataupun jurnal luar negeri dan dari beberapa buku. Dalam studi literatur ini, penulis mencari sumber terkait permasalahan – permasalahan yang perlu menjadi perbaikan dalam penelitian selanjutnya.

## Kerangka Kerja Penelitian

Bagian ini menjelaskan tentang bagaimana penelitian ini dilakukan. Berikut adalah beberapa proses penting yang dilakukan :

**Gambar 1. 1 Kerangka Kerja Sistem**

* + - 1. Instalasi kebutuhan konfigurasi VPS(Virtual Private Server)

Tahap pertama dalam penelitian ini dengan mempersiapkan sistem dan software yang dibutuhkan dalam mengkonfigurasi kebutuhan web server dalam menginstalasi modul-modul kebutuhan sistem seperti sistem operasi, database, framework, web server .

* + - 1. Menentukan proses konfigurasi kebutuhan aplikasi

Pada tahap ini mempersiapkan kebutuhan konfigurasi untuk membangun sebuah web server, dari web service sampai database yang diperlukan untuk menampung setiap user.

* + - 1. Melakukaan proses otomatisasi konfigurasi modul aplikasi

Pada tahap ini melakukan otomatisasi dalam mengkonfigurasi sebuah web server. Dimana proses otomatisasi menggunakan API dari Ansible.

* + - 1. Merancang web front-end untuk user

Pada tahap ini bila semua konfigurasi sudah berjalan dengan baik, dibuat web yang dapat digunakan oleh pengguna mengatur kebutuhan web yang dibuat serta dapat melihat informasi dari setiap VPS yang dimiliki.

* + - 1. Melakukan test pada hasil konfigurasi

Pada tahap ini menguji hasil konfigurasi dari web interface. Untuk menguji konfigurasi berjalan dengan baik atau tidak

1. Melakukan uji beban server

beban ini dilakukan untuk menguji tingkat kemampuan server dalam memberikan pelayanan pada pengemban aplikasi. Bila server tidak sanggup untuk melayani pengembang baru akan menolak menyediakan layanan.

1. Melakukan Backup Server

Ketika server user sudah berjalan, sistem akan melakukan backup data pengguna untuk menjaga ketersediaan data pengguna bila terjadi sesuatu yang dapat merugikan pengguna, seperti file data hilang atau rusak,dapat dikembalikan dengan restore pada sistem dengan menggunakan waktu check poin.

## Perancangan Sistem

Pada tahap Perancangan sistem, dilakukan berdasarkan hasil analisa kebutuhan sistem yang sudah dilakukan sebelumnya. Hal ini dilakukan agar, perancangan tidak keluar dari tujuan sistem yang dikembangkan. Berikut merupakan desain arsitektur kerja sistem:



**Gambar 1. 2 Desain Kerja Sistem**

Pada gambar 1.2 merupakan hubungan antara setiap perangkat lunak yang ada pada server utama. Berikut ini merupakan penjelasannya :

Web dan database

Database digunakan untuk menyimpan informasi pengguna serta informasi pada VPS yang dimiliki oleh setiap pengguna.

Web dan framework flask REST API

Flask REST API digunakan untuk menghubungkan web front-end dengan back-end. Dimana back-end dibuat dengan menggunakan framework flask dari python, selanjutnya dibuat API tersendiri agar web front-end dapat mengirim dan menerima informasi serta konfigurasi yang dilakukan yang nantinya akan diproses oleh back-end server.

Database dan ansible API

Hubungan database dengan ansible API akan mengambil informasi pengguna berupa username, email dan password yang nantinya akan dimasukan pada konfigurasi dalam VPS yang selanjutnya sebagai super amin pada VPS yang dibuat.

framework flask REST API dan asible API

Hubungan framework flask REST API dengan asible API. Ketika user melakukan konfigurasi pada web utama akan dikirim melalui REST API flask selanjutnya konfigurasi tersebut akan digunakan oleh API ansible untuk mengkonfigurasi VPS yang dibuat.

framework flask dan openstack API

Hubungan framework flask dan openstack API adalah pada flask dikonfigurasi untuk dapat terhubung dengan server devstack, dimana server openstack yang mengelola dalam membuat sebuah VPS.

openstack API dan server

API openstack digunakan untuk terhubung dengan server back-end yang dibuat dengan framework flask. Untuk dapat mengembangkan aplikasi openstack dari bahasa pemrograman yang berbeda.

ansible API dan Virtual Private Server

hubungan ansible API dengan VPS adalah ansible akan mengirimkan konfigurasi pada VPS melalui SSH yang yang dibuat. Selanjutnya pada vps akan melakukan konfigurasi yang diperlukan untuk membangun sebuah web aplikasi secara otomatis. Pengguna hanya perlu memilih keperluan yang ada menu web front-end.

server dan Virtual Private Server

hubungan antara server dan VPS(Virtual Private Server) adalah server mengelola dan memberikan resource pada VPS serta mengatur konektivitas setiap VPS yang dimiliki.

## Flowchart Sistem

Pada bagian ini akan menjelaskan proses yang dilakukan oleh perangkat lunak, bagaimana proses tersebut berjalan yang akan dijelaskan pada setiap flowchart berikut ini:

### Flowchart Konfigurasi Otomatis



**Gambar 1. 3 Flowchart Konfigurasi Otomatis**

Pada desain flowchart diatas merupakan garis besar gambaran system yang akan dikerjakan, bagaimana alur platform yang dibuat dapat mengotomatitasi dalam instalasi serta konfigurasi sebuah layanan pada virtual machine. Mulai dari memasukkan aplikasi pendukung yang ingin digunakan seperti database, web service,usernme dan e-mail administrator layanan aplikasi. Kemudian dari hasil inputan tersebut diolah oleh flask API yang dibuat untuk dimasukkan ke dalam konfigurasi yang terdapat ada Ansible API, selanjutnya dari ansible API dengan menggunakan SSH akan melakukan instalasi konfigurasi pada virtual server yang didapat setiap user. Bila konfigurasi telah selesai user akan menerima sebuah ip public untuk dapat mengkakses layanan aplikasi.

### Flowchart Front-end Web



**Gambar 1. 4 Flowchart Front-end Web**

Pada desain flowchart diatas merupakan alur kerja web dari sisi front-side atau pada sisi user. Setelah user melakukan registrasi pada website, selanjutnya user akan diarahkan ke menu instalasi dan konfigurasi dalam membangun layanan VPS. Menu akan dibuat secara user friendly sehingga user hanya perlu memilih menu konfigurasi system. Ketika user sudah selesai memilih menu konfigurasi, system akan memvalidasi informasi yang dimasukkan oleh user, bila sudah benar system akan mengirimkan hasil instalasi dan konfigurasi dalam sebuah data berrntuk JSON ke dalam system yang nantinya akan dikirim ke virtual sever untuk melakukan konfigurasi dan instalasi modul kebutuhan aplikasi selanjutnya sistem akan melakukan backup berkala untuk menjaga keamanan data pengguna bila tejadi satu kesalahan dapat dilakukan restore data.

## Rancangan ERD



**Gambar 1. 5 Entity Relationship Diagram Sistem**

Pada desain ERD diatas menjelaskan hubungan user dengan VPS yang dimiliki user memiliki sebuah vps dengan satu ip public untuk dapat mengkases server e-learning. Serta pada vps akan menyimpan log keadaan vps mulai dari kinerja dan resource yang dimiliki.

## Evaluasi Perancangan Sistem

Pada tahap ini dilakukan evaluasi terhadap desain perancangan sistem. Bila sistem telah sesuai dengan kebutuhan awal yang didefinisikan akan dilanjutkan ke tahap implementasi. Namun apabila desain sistem belum memenuhi kebutuhan awal yang didefinisikan, maka akan dilakukan perancangan ulang desain sistem.

# BAB II

**TINJAUAN PUSTAKA**

## Tinjauan Studi

Pada penelitian ini, peneliti menggunakan beberapa penelitian yang pernah dilakukan mengenai sistem layanan PaaS dan metode adaptive threshold. Beberapa penelitian tersebut yaitu:

**Tabel 2. 1 Tinjauan Studi**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Judul | Tahun | Penulis |
| 1. | A Dynamic Self-adaptive Resource-Load Evaluation Method in Cloud Computing | 2015 | Liyun Zuo, Lei Shu, Shoubin Dong, Zhangbing Zhou, Lei Wang |
| 2. | Adaptive Resource Management for Service Workflows in Cloud Environments | 2013 | Yi Wei dan M. Brian Blake dan Iman Saleh |
| 3. | A Heuristic Adaptive Threshold Algorithm on IaaS Clouds | 2015 | Qingxin Xia dan Yuqing Lan, Limin Xiao |
| 4. | The Design and Implementation of Resource Monitoring for Cloud Computing Service Platform | 2013 | Lei Xiaojiang, Shang Yanlei |
| 5. | An Adaptive Threshold Method to Address Routing Issues in Delay-Tolerant Networks | 2011 | Nicole Ng, Hwa Chang, Zhongjian Zou, Sai Tang |

Dalam jurnal A Dynamic Self-adaptive Resource-Load Evaluation Method in Cloud Computing memaparkan Sumber daya cloud membuat indikator evaluasi statis tradisional tidak dapat menggabarkan keadaan sumber daya secara akurat, sehingga pada penelitian ini menggukanan tiga indikator dinamis yaitu jumlah permintaan sumber daya, kapasitas komputasi sumber daya dan kekuatan beban sumber daya untuk mengevaluasi keadaan beban sumber daya. Algoritma self-adaptive digunakan untuk mengevaluasi beban sumber daya. Dengan algortima tersebut dapat membagi status load resource menjadi overload, normal dam idle. Pada sumber daya yng mengalami overload akan dimigrasi pada load balance, dan idle resource akan dilepas untuk menghemat energi bila waktu idle melebihi batas threshold.sehingga pada penelitian ini mengusulkan dua threshold adaptif yang dipengaruhi oleh indikator evaluasi secara dinamis. Pada percobaan menampilkan SDWM dengan keuntungan yang sangat baik dari waktu respond dan pemanfaatan sumber daya ketika sumber daya secara otomatis masuk dan keluar (Liyun Zuo, 2015).

Dalam jurnal Adaptive Resource Management for Service Workflows in Cloud Environments menjelaskan Memanajemen sumber daya merupakan bagian yang sangat penting pada alur kerja manajemen aktivitas di dalam cloud. Dengan algoritma adaptif Resources Management untuk layanan alokasi dan dealokasi secara dinamis pada alur kerja sumberdaya sebelum mereka mengeksekusi pada sistem cloud. Tujuan dari jurnal ini, algoritma dapat menjaga jumlah total sumber daya yang dimiliki oleh masing-masing layanan pada tingkat yang diinginkan sehingga sumber daya tersebut tidak terlalu banyak melakukan komunikasi dan tidak memanfaatkan banyak sumberdaya. Hasil prediksi adaptif digunakan untuk memandu algoritma dalam memilih kandidat sumber daya yang akan di lepas atau meminta sumber daya baru untuk dialokasikan. Dengan menjalankan simulasi pada data beban kerja secara sintetis. Algoritma tersebut melakukannya dengan baik daripada reaktif sumber daya Management yang ada dan dapat mencapai fluktuasi beban layanan yang lebih rendah dan kehilangan sumber daya (Saleh, 2013).

Dalam jurnal A Heuristic Adaptive Threshold Algorithm on IaaS Clouds memaparkan teknologi untuk menghemat energi IaaS telah banyak menarik perhatian, namun bagi penyedia layanan cloud IaaS menjamin penghematan energi dan kinerja di bawah kondisi Service Level Agreement (SLA). Pada jurnal ini menggunakan metode berbasis adaptif threshold untuk mengurangi tradeoff antara penghematan energi dan SLA. Dar dapat menemukan ambang batas secara optimal, penulis mengusulkan sebuah framework yang basisnya pada Ben kerja, yang mengintegrasikan secara mulus metode prediksi berbasis pilihan dan model untuk mengukur hubungan antara biaya migrasi mesin virtual (VM) dan listrik saat mesin fisik mati. Algoritma threshold dirancang untuk menangkap ambang batas secara efektif. Dengan metode ini dapat memperbaiki konsumsi energi sebesar 10-20 persen. Algoritma threshold adaptif lebih efisien dan mampu beradaptasi dengan baik terhadap beban kerja dengan variabel tinggi ( Xia, Lan, & Xiao, 2015).

Dalam jurnal “The Design and Implementation of Resource Monitoring for Cloud Computing Service Platform”. Membahas dengan cloud computing pengguna dapat menemukan solusi baru yang efektif untuk membangun sistem dan aplikasi berdasarkan kebutuhan mereka. karena cloud computing menghasilkan ketersediaan dan skalabilitas yang tinggi. pada bagian background tim tersebut mengembangkan subsistem yang digunakan untuk memonitor sumber daya, dengan bantuan dari platform yang dibuat, pada layer atas berjalan dengan lancar (Lei Xiaojiang, 2013).

Pada jurnal An Adaptive Threshold Method to Address Routing Issues in Delay-Tolerant Networks memaparkan algoritma routing berbasis epidermi memberikan solusi dimana menggunakan beberapa salinan paket untuk meningkatkan probailtas pengiriman paket sukes, namun banyak salinan paket tidak hanya meningkatkan beban jaringan tetapi juga menurunkan troughput, dari hal tersebut pada jural ini menggunakan metode threshold, metode threshold menggunakan pendekatan nilai statis. pada jurnal metode threshold adaptif mempengaruhi nilai threshold secara terus menerus sehingga dapat menyesuaikan dengan topologi secara dinamis. pada hasil simulasi tersebut memverifikasi ahwa nilai threshold adaptif melebihi metode threshold dengan mengurangi tingkat pengiriman paket. Pada penelitian ini mengji kinerja metode threshold adaptif melalui berbagai topologi sederhana, dengan metode threshold secara konsisten menunjukkan throughput yang tinggi dengan mengurangi tingkat pengiriman sekitar 3,9 detik. oleh karena itu metode adaptif threshold merupakan pendekatan secara praktis untuk menangani masalah routing pada Delay-Tolerant Networks(DTN) (Nicole Ng, 2011).

## Cloud Computing

Cloud computing mengacu pada aplikasi dan service yang berjalan dalam jaringan data terdistribusi dengan menggunakan sumberdaya virtual dan internet akses protokol pada umumnya. Hal ini dibedakan pada gagasan sumberr daya virtual dan detail dari mesin fisik sistem dalam software yang berjalan secara abtraksi dari user. (Sosinsky, 2011). Cloud Computing secara sederhana adalah “layanan teknologi informasi yang bisa dimanfaatkan atau diakses oleh pelanggannya melalui jaringan internet”. Komputasi awan adalah suatu konsep umum yang mencakup SaaS, Web 2.0, dan tren teknologi terbaru lain yang dikenal luas, dengan tema umum berupa ketergantungan terhadap Internet untuk memberikan kebutuhan komputasi pengguna. Sebagai contoh, Google Apps menyediakan aplikasi bisnis umum secara sharing yang diakses melalui suatu penjelajah web dengan perangkat lunak dan data yang tersimpan di server.

## Jenis Cloud Computing

Jenis-jenis dari Cloud Computing dapat dijabarkan sebagai berikut:

### IaaS (Infrastructure as a Service)

Terletak satu level lebih rendah dibanding PaaS. Ini adalah sebuah layanan yang “menyewakan” sumberdaya teknologi informasi dasar, yang meliputi media penyimpanan, processing power, memory, sistem operasi, kapasitas jaringan dan lainlain, yang dapat digunakan oleh penyewa untuk menjalankan aplikasi yang dimilikinya. Model bisnisnya mirip dengan penyedia data center yang menyewakan ruangan untuk co-location, tapi ini lebih ke level mikronya. Penyewa tidak perlu tahu, dengan mesin apa dan bagaimana caranya penyedia layanan menyediakan layanan IaaS. Yang penting, permintaan mereka atas sumberdaya dasar teknologi informasi itu dapat dipenuhi.

### Platform as a Service (PaaS)

Konsepnya hampir serupa dengan IaaS. Namun Platform disini adalah penggunaan operating system dan infrastruktur pendukungnya. Yang cukup terkenal adalah layanan dari situs Force.Com serta layanan dari para vendor server.Seperti namanya, PaaS adalah layanan yang menyediakan modul-modul siap pakai yang dapat digunakan untuk mengembangkan sebuah aplikasi, yang tentu saja hanya bisa berjalan diatas platform tersebut. Seperti juga layanan SaaS, pengguna PaaS tidak memiliki kendali terhadap sumber daya komputasi dasar seperti memory, media penyimpanan, processing power dan lain-lain, yang semuanya diatur oleh provider layanan ini. Pionir di area ini adalah Google AppEngine, yang menyediakan berbagai tools untuk mengembangkan aplikasi di atas platform Google, dengan menggunakan bahasa pemrograman Phyton dan Django. Kemudian Salesforce juga menyediakan layanan PaaS melalui Force.com, menyediakan modul-modul untuk mengembangkan aplikasi diatas platform Salesforce yang menggunakan bahasa Apex. Dan mungkin yang jarang sekali kita ketahui, bahwa Facebook juga bisa dianggap menyediakan layanan PaaS, yang memungkinkan kita untuk membuat aplikasi diatasnya.Salah satu yang berhasil menangguk untung besar dari layanan PaaS

### Software as a Service (SaaS)

Berada satu tingkat diatas PaaS dan IaaS, dimana disini yang ditawarkan adalah software atau suatu aplikasi bisnis tertentu. Contoh yang paling mutakhir adalah SalesForce.Com, Service-Now.Com, Google Apps, dsb. SaaS ini merupakan layanan Cloud Computing yang paling dahulu populer.Software as a Service ini merupakan evolusi lebih lanjut dari konsep ASP (Application ServiceProvider). Sesuai namanya, SaaS memberikan kemudahan bagi pengguna untuk bisa memanfaatkan sumberdaya perangkat lunak dengan cara berlangganan. Sehingga tidak perlu mengeluarkan investasi baik untuk in house development ataupun pembelian lisensi. Dengan cara berlangganan via web, pengguna dapat langsung menggunakan berbagai fitur yang disediakan oleh penyedia layanan. Hanya saja dengan konsep SaaS ini, pelanggan tidak memiliki kendali penuh atas aplikasi yang mereka sewa.Hanya fiturfitur aplikasi yang telah disediakan oleh penyedia saja yang dapat disewa oleh pelanggan.

## Openstack

OpenStack adalah sistem aplikasi cloud yang mengelola sumberdaya seperti komputasi, penyimpan dan jaringan, yg tersedia pada infrastruktur fisik seperti dalam sebuah fasilitas pusat-data (data center). Admin atau pengguna dapat mengendalikan dan melakukan provisioning atas sumber-daya ini melalui dashboard / antar-muka web. Developer dapat mengakses sumber-daya tersebut melalui sejumlah API standar (Mulyana, 2017).

Openstack merupakan platform cloud-computing open source yang memungkinkan pengguna untuk membangun sebuah "IAAS" Infrastruktur sebagai service cloud yang bergerak secara massal pada komoditas hardware dan skala. Openstack mengontrol kolam besar komponen komputasi awan di seluruh datacenter, semua dikelola melalui dashboard yang menyediakan administrator kontrol penuh sambil memberikan pengguna kemampuan untuk sumber penyediaan melalui antarmuka web.

## Sistem Administrator

System administrator atau sysadmin merupakan sebuah pekerjaan yang bertugas mengatur dan memelihara dan mengoperasikan sistem komputer dan jaringan komputer. Dalam perusahaan maupun sebuah organisasi sysadmin sangat dibutuhkan dalam mengelola dan mengamankan data pada server yang dimiliki perusahaan (Collings & Kurt , 2015). Adapun tugas sysadmin sebagai berikut:

1. Menginstalasi dan menkonfigurasi server.
2. Menginstall dan mengkonfigurasi software aplikasi.
3. Membuat dan mengelola user.
4. Backup dan restore file.
5. Konfigurasi keamanan server.
6. Memonitor keamanan jaringan

## Python

Python merupakan bahasa pemrograman yang sangat powerful yang memiliki beberapa kesamaan dengan Fortran, salah satu bahasa pemrograman paling awal, namun lebih baik dari fortran. Python memungkinkan anda untuk menggunakan variabel tanpa perlu mendeklarasikannya. Sehingga kita tidak dipaksa untuk mendefinisikan class dengan python.

Python dikembangkan oleh Guido van Rossum pada tahun 1990 di CWI, Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. Versi terakhir yang dikeluarkan CWI adalah 1.2. (Doty, 2008).

Tahun 1995, Guido pindah ke CNRI sambil terus melanjutkan pengembangan Python. Versi terakhir yang dikeluarkan adalah 1.6. Tahun 2000, Guido dan para pengembang inti Python pindah ke BeOpen.com yang merupakan sebuah perusahaan komersial dan membentuk BeOpen PythonLabs. Python 2.0 dikeluarkan oleh BeOpen. Setelah mengeluarkan Python 2.0, Guido dan beberapa anggota tim PythonLabs pindah ke DigitalCreations.

## RESTful API

REST (REpresentational State Transfer) merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Umumnya menggunakan HTTP (Hypertext Transfer Protocol) sebagai protocol untuk komunikasi data. REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000.

REST API merupakan web service yang bertujuan untuk mendukung kebutuhan server web pada suatu kebutuhan situs atau aplikasi lainnya. program client mengunakan Application Programming Interface (API) untuk berkomunikasi dengan layanan web. Secara umum, API mengekspos seperangkat data dan fungsi untuk memfasilitasi interaksi antara proram komputer dan memungkinkan merekas saling bertukar informasi (Masse, 2012).

Dalam pengaplikasiannya, REST lebih banyak digunakan untuk web service yang berorientasi pada resource. Maksud orientasi pada resource adalah orientasi yang menyediakan resource-resource sebagai layanannya dan bukan kumpulan-kumpulan dari aktifitas yang mengolah resource tersebut.

## Ansible

Ansible merupakan sebuah softaware yang bisa membantu seorang sistem administrator untuk melakukan otomasi pada server. ansible merupakan teknologi yang digunakan untuk melakukan otomasi, memudahkan dalam melakukan konfigurasi server, tujuan dibuat ansible membuat hal tersebut menjadi sederhana dan mudah.namun tetap fokus pada keamana dan keandalan dalam melakukan otomasi. ansible menggunakan OpenSSH untuk transportasi ( dengan mode socket yang cepat).

Dengan ansible sysadmin dapat melakukan instalasi, deployment hingga melakukan update server. Sistem kerja yang dimiliki oleh ansible membutuhkan koneksi khusus berupa SSH. Ansible bekerja di koneksi SSH remote client yang ingin di deploy atau dilakukan otomasi. Pada ansible memerlukan inventory atau data server tujuan untuk dapat dilakukan otomasi. Pada penerapannya, ansible menggunakan playbook dan oles, dimana konfigurasi tersebut dalam format markup YAML dan environment variabel dapat ditulis dalam bentuk JSON. .

ansible dirancang untuk memudahkan para sysadmin dan para pakar IT mengelola lingkungan server dengan mudah. ansible mengelola mesin dengan cara yang tidak biasa, tidak pernah bertanya cara melakukan upgrade daemon jarak jauh atau masalah karena tidak dapat mengelola sistem karena daemon sistem terhapus.

Ansible merupakan salah satu jenis Configuration Management Tools ayan dapat digunakan merubah proses infrastruktur manajemen dari program manual menjadi otomatis. Dalam zaman cloud kehadiran ansible membantu para sysadmin atau para devops dalam instalasi dan konfigurasi server dengan otomatis, oleh karena itu ansible menjadi satu platform yang digunakan untuk mengelola server – server.

# BAB III

**ANALISIS DAN PERANCANGAN**

## Analisis Kebutuhan Sistem

Pada tahap ini dilakukan analisis kebutuhan sistem, analisis kebutuhan sistem meliputi data yang digunakan, pembelajaran dari referensi yang sudah ada dan perangkat yang digunakan baik perangkat lunak maupun perangkat keras:

1. Tahapan pertama yang dilakukan dalam penelitian ini adalah mengidentifikasi permasalahan. Tahap ini merupakan tahap yang paling penting dalam penelitian karena jalannya penelitian didasarkan atas permasalahan yang terjadi. Setelah menentukan masalah yang terjadi, tahapan yang diperlukan selanjutnya adalah menentukan rumusan masalah dan tujuan yang ingin dicapai dalam penelitian. Pada penelitian ini identifikasi permasalahan dilakukan dengan menggunakan teknik observasi, dari teknik ini maka akan dapat diketahui mengenai keluhan – keluhan yang ada di lapangan.
2. Tahap kedua yang dilakukan dalam metodelogi penelitian ini adalah studi literatur. Studi literatur dilakukan dengan mengambil literatur – literatur pendukung dari jurnal – jurnal ilmiah, baik jurnal dalam negeri ataupun jurnal luar negeri dan dari beberapa buku. Dalam studi literatur ini, penulis mencari sumber terkait permasalahan – permasalahan yang perlu menjadi perbaikan dalam penelitian selanjutnya.

### Kebutuhan Fungsional

### Kebutuhan Non Fungsional

## Perancangan Sistem

Pada tahap Perancangan sistem, dilakukan berdasarkan hasil analisa kebutuhan sistem yang sudah dilakukan sebelumnya. Hal ini dilakukan agar, perancangan tidak keluar dari tujuan sistem yang dikembangkan

### *Desain Arsitektur Kerja Sistem*



Pada gambar 7.3.1.1 merupakan hubungan antara setiap perangkat lunak yang ada pada server utama. Berikut ini merupakan penjelasannya :

*Web* dan *database*

*Database* digunakan untuk menyimpan informasi pengguna serta informasi pada *VPS* yang dimiliki oleh setiap pengguna.

*Web* dan *framework flask* *REST API*

Flask REST API digunakan untuk menghubungkan web *front-end* dengan *back-end*. Dimana *back*-*end* dibuat dengan menggunakan *framework flask* dari *python*, selanjutnya dibuat *API* tersendiri agar *web front-end* dapat mengirim dan menerima informasi serta konfigurasi yang dilakukan yang nantinya akan diproses oleh *back-end server*.

*Database* dan *ansible* *API*

Hubungan *database* dengan *ansible API* akan mengambil informasi pengguna berupa *username*, *email* dan *password* yang nantinya akan dimasukkan pada konfigurasi dalam v*irtual machine* yang selanjutnya sebagai *super admin* pada *VPS* yang dibuat.

*Framework flask REST API* dan *asible API*

Hubungan *framework flask REST API* dengan *asible API*. Ketika *user* melakukan konfigurasi pada *web* utama akan dikirim melalui *REST API flask* selanjutnya konfigurasi tersebut akan digunakan oleh *API* *ansible* untuk mengkonfigurasi *virtual machine* yang dibuat.

*Framework flask* dan *openstack API*

Hubungan *framework flask* dan *openstack API* adalah pada *flask* dikonfigurasi untuk dapat terhubung dengan *server* *devstack*, dimana *server* *openstack* yang mengelola dalam membuat sebuah *virtual machine*.

*Openstack* *API* dan *server*

*API openstack* digunakan untuk terhubung dengan server *back-end* yang dibuat dengan *framework flask*. Untuk dapat mengembangkan aplikasi *openstack* dari bahasa pemrograman yang berbeda.

*Ansible API* dan *virtual server*

Hubungan *ansible API* dengan *virtual* mesin adalah *ansible* akan mengirimkan konfigurasi pada *VPS* melalui *SSH* yang yang dibuat. Selanjutnya pada virtual mesin akan melakukan konfigurasi yang diperlukan untuk membangun sebuah *web* aplikasi secara otomatis. Pengguna hanya perlu memilih keperluan yang ada *menu* *web front-end.*

*Server* dan *Virtual* Mesin

Hubungan antara *server* dan *virtual* mesin adalah *server* mengelola dan memberikan *resource* pada *virtual* mesin serta mengatur konektivitas setiap *virtual* mesin yang dimiliki.

### *Flowchart* *System*

Pada bagian ini akan menjelaskan proses yang dilakukan oleh perangkat lunak, bagaimana proses tersebut berjalan yang akan dijelaskan pada setiap *flowchart* berikut ini:

### *Flowchart* Konfigurasi Otomatis



Gambar 3.2.2.1 Flowchart Konfigurasi Otomatis

Pada desain *flowchart* 3.2.2.1merupakan garis besar gambaran *system* yang akan dikerjakan, bagaimana alur *platform* yang dibuat dapat mengotomatitasi dalam instalasi serta konfigurasi sebuah layanan pada *virtual machine*. Mulai dari memasukkan aplikasi pendukung yang ingin digunakan seperti *database, web service, username* dan *e-mail administrator* layanan aplikasi. Kemudian dari hasil *input* tersebut diolah oleh *flask API* yang dibuat untuk dimasukkan ke dalam konfigurasi yang terdapat ada *Ansible API*, selanjutnya dari ansible *API* dengan menggunakan *SSH* akan melakukan instalasi konfigurasi pada *virtual server* yang didapat setiap *user*. Bila konfigurasi telah selesai *user* akan menerima sebuah *ip public* untuk dapat mengakses layanan aplikasi.

### *Flowchart* Konfigurasi Otomatis



Gambar 7.4.3.1 *Flowchart Back-End*

Pada gambar flowchart 7.4..3.1 menjelaskan alur kerja sistem *back-end* sistem. Tahap pertama yang harus dilakukan adalah melakukan komunikasi dengan *openstack* agar dapat menggunakan *API* dari *openstack*. Melakukan komunikasi dengan melakukan *request* *token* *authentikasi*, hal ini dimaksudkan agar setiap komunikasi yang dibuat dapat melakukan *update* *data* pada *API* dan yang memiliki *token* dari *autentikasi* saja yang dapat berkomunikasi. Selanjutnya setelah melakukan *request*, sistem tetap akan melakukan validasi *token* *id* dari *openstack*, dikarena setiap melakukan *request* *token* memiliki batas waktu yang ditentukan. Setelah bisa melakukan komunikasi dengan *openstack* selanjutnya menerima masukan konfigurasi dari *front*-end dan sudah divalidasi oleh administrator, selanjutnya merubah setiap masukan konfigurasi dalam ke *YAML*, *YAML* merupakan format untuk memberi perintah pada *ansible* agar dapat melakukan instalasi maupun konfigurasi secara otomatis. Setelah *file* siap, sistem akan membuat *virtual* mesin dan mengirimkan konfigurasi tersebut ke *virtual* mesin dan langsung melakukan konfigurasi secara otomatis. Selanjutnya sistem akan kembali melakukan mengecekkan apakah sudah semua *terinstall* dan *terkonfigurasi* dengan baik, sehingga bila ada gagal dalam dilakukan *instalasi* kembali. Setelah semua dicek dan sukses maka selanjutnya sistem akan menampilkan *virtual* mesin siap dan sebuah *ip public* untuk diakses oleh *developer* dan *ssh* *key* bila *developer* memiliki kebutuhan aplikasi khusus yang tidak dimiliki oleh sistem.

### *Entity Relational Diagram*



Gambar 7.4.3.1 *Entity Relationship Diagram* Sistem

Pada desain *ERD* diatas menjelaskan hubungan user dengan *VPS* yang dimiliki user memiliki sebuah vps dengan satu *ip public* untuk dapat mengkases *server*. Serta pada *vps* akan menyimpan *log* keadaan *vps* mulai dari kinerja dan *resource* yang dimiliki.

### Perancangan Antar Muka

### Skenario Pengujian Sistem

Bagian ini menjelaskan mengenai skenario pengujian sistem yang akan dilakukan pada Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack. Pengujian sistem ini dilakukan dengan menguji black box dan *Performance Testing.*

### BlackBox Testing

Black Box Testing atau dikenal sebagai “Behaviour Testing” merupakan suatu metode pengujian yang digunakan untuk menguji executable code dari suatu perangkat lunak terhadap perilakunya. Pendekatan Black Box Testing dapat dilakukan jika kita sudah memiliki executable code. Orang-orang yang terlibat dalam Black Box Testing adalah tester, end-user, dan developer.

Fokus dari pengujian ini ialah pada kebutuhan fungsional perangkat lunak, sehingga memungkinkan tester mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu untuk program. Kesalahan yang ditemukan dalam pengujian, nantinya dapat disimpulkan apakah kesalahan tersebut murni dikarenakan kesalahan dari aplikasi atau kesalahan implementasi dari tester.

**Tabel 3. 1 Tabel Pengujian Black Box**

|  |  |  |
| --- | --- | --- |
| *Identifikasi* |  | |
| *Nama Kasus Uji* |  | |
| *Deskripsi* |  | |
| *Kondisi Awal* |  | |
| *Tanggal Pengujian* |  | |
| *Penguji* |  | |
| *Skenario* | | |
| *1.* |  | |
| *2.* |  | |
| *(Dst...)* |  | |
| ***Hasil Yang Diharapkan*** | ***Hasil Yang Didapatkan*** | ***Kesimpulan*** |
|  |  |  |

Untuk pengujian antarmuka pengguna atau rancangan skenario pengujian balck box dari sistem ini, dilakukan dua jenis pengujian yaitu pengujian secara happy path yaitu pengujian yang dilakukan dengan cara yang benar, serta pengujian secara alternative path yaitu mencoba segala kemungkinan yang mungkin terjadi pada sistem.

### Performance Testing

Teknik pengujian memvalidasi perilaku perangkat lunak terhadap teknik pengujian software dari sisi kecepatan. Kecepatan ini dalam konteks pengujian dalam mengukur waktu respon perangkat lunak ketika berada jumlah kerja yang berlebih yang biasa dikenal dengan beban kerja. Untuk memperlihatkan kecepatan sebenarnya sebuah perangkat lunak harus dilakukan pengujian performance testing.

Tujuan dari performance testing untuk memvalidasi kecepatan sebuah perangkat lunak terhadap kebutuhan sistem yang cepat. Secara umum harus mendefisinikan kombinasi waktu respons dan beban kerja

# BAB IV

**HASIL DAN PEMBAHASAN**

## Gambaran Umum Sistem

Sistem yang dibuat bertujuan membantu para pengembang aplikasi berbasis web, dengan meneriakan platform web server berbasis cloud computing, dengan menyediakan virtual machine berbasis cloud/ instance. Pengembang web dapat dengan mudah menjalankan sebuah web yang dibuat dengan mengakses alamat IP yang di berikan. Karena berbasis cloud, pengembang web dapat mengakses virtual machine yang di buat dimana saja secara online. Namun pada penetilian ini pengembang hanya dapat menggunakan jaringan Local untuk mengkases server cloud / private cloud.

Dalam implementasinya, server virtual yang diberikan pada pengembang web, secara otomatis sudah terinstall LAMP package. Namun, bila pengembang web ingin menggunakan package web server yang berbeda dapat melakukan instalasi manual dengan mengakses virtual machine menggunakan SSH.

## Lingkungan Implementasi

Sistem dirancang dan diimplementasikan pada sistem operasi Ubuntu dalam virtualisasi menggunakan Vmware, server dan sistem berjalan pada Laptop dengan spesifikasi i7, RAM 16 GB, dengan sistem operasi windows 10 64bit dan dengan support virtualisasi.

## Implementasi Basis Data

|  |
| --- |
| app.config['SECRET\_KEY'] = 'Idabagusrathuekasuryawibawa!'  app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = True  app.config['SQLALCHEMY\_DATABASE\_URI']= 'sqlite:////home/gustu/myweb/database/database.db'  Bootstrap(app)  db = SQLAlchemy(app)  login\_manager = LoginManager()  login\_manager.init\_app(app)  login\_manager.login\_view = 'login'  level = 0  class User (UserMixin, db.Model):  id = db.Column(db.Integer,primary\_key=True)  username = db.Column(db.String(20), unique=True)  email = db.Column(db.String(50), unique=True)  password = db.Column(db.String(80))  level = db.Column(db.Integer)  class Project(UserMixin, db.Model):  id = db.Column(db.Integer,primary\_key=True)  project = db.Column(db.String(20), unique=True)  description = db.Column(db.String(255))  idos = db.Column(db.String(128))  osdesc = db.Column(db.String(50))  idcpu = db.Column(db.String(128))  cpudesc = db.Column(db.String(50))  user = db.Column(db.String(20))  addr = db.Column(db.String(255))  @login\_manager.user\_loader  def load\_user(user\_id):  return User.query.get(int(user\_id))  class LoginForm(FlaskForm):  username = StringField('username', validators=[InputRequired(), Length(min=4, max=20)])  password = PasswordField('password', validators=[InputRequired(), Length(min=8, max=80)])  remember = BooleanField('remember me')  class RegisterForm(FlaskForm):  email = StringField('email', validators=[InputRequired(), Email(message='email salah'), Length(max=50)])  username = StringField('username', validators=[InputRequired(), Length(min=4, max=20)])  password = PasswordField('password', validators=[InputRequired(), Length(min=8, max=80)])  class ProjectForm(FlaskForm):  namaproject = StringField('namaproject', validators=[InputRequired(), Length(max=20)])  deskripsi = StringField('deskripsi')  sistem\_operasi = StringField('sistem\_operasi')  cpu = StringField('cpu') |

|  |
| --- |
| def signup():  form=RegisterForm()  if form.validate\_on\_submit():  hashed\_password = generate\_password\_hash(form.password.data, method='sha256')  new\_user = User(username=form.username.data, email=form.email.data, password=hashed\_password, level=1)  db.session.add(new\_user)  db.session.commit()  return render\_template('login\_new.html', form=LoginForm())  return render\_template('signup.html', form=form) |

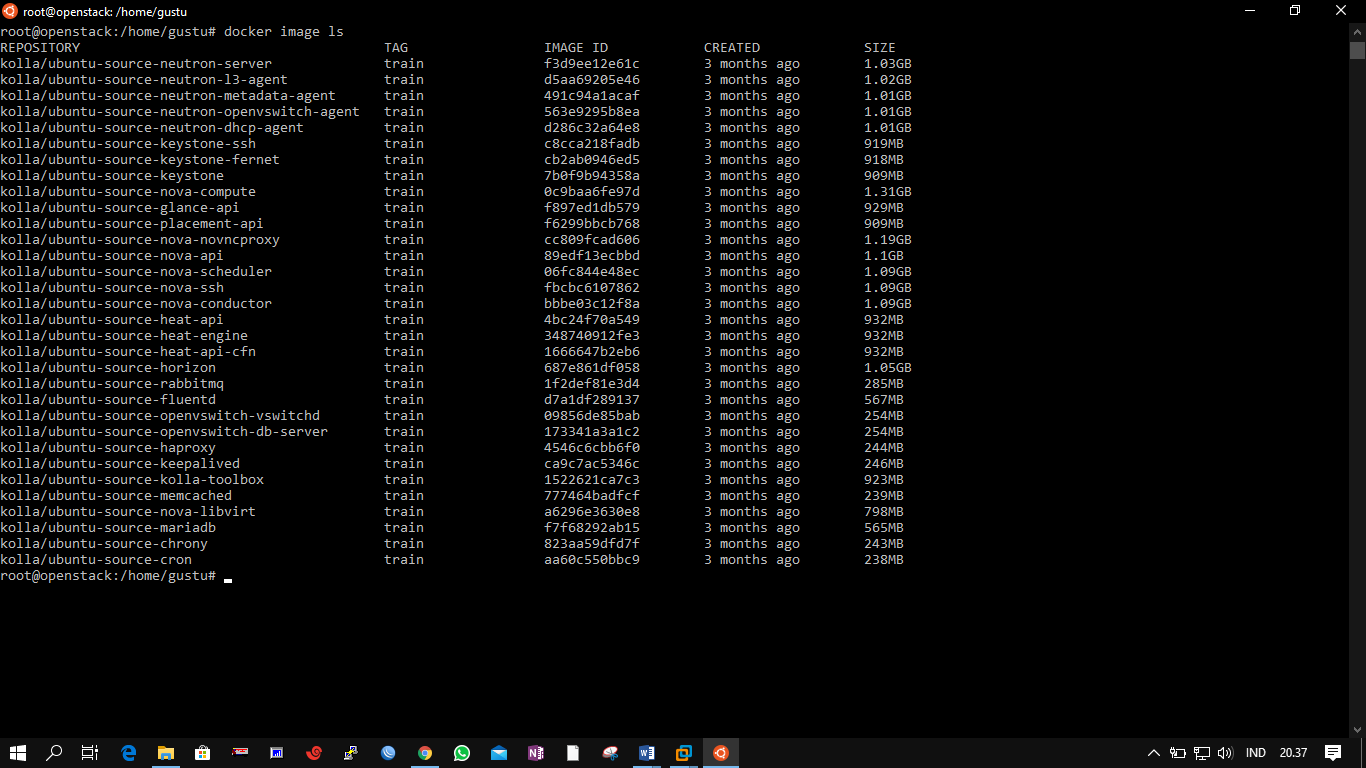
|  |
| --- |
| def login():  global level  form=LoginForm()  if form.validate\_on\_submit():  user = User.query.filter\_by(username=form.username.data).first()  if user:  if check\_password\_hash(user.password, form.password.data) and user.level >= 0:  login\_user(user, remember=form.remember.data)  level = user.level  return redirect(url\_for('dashboard'))  return redirect(url\_for('login'))  return render\_template('login.html', form=form) |

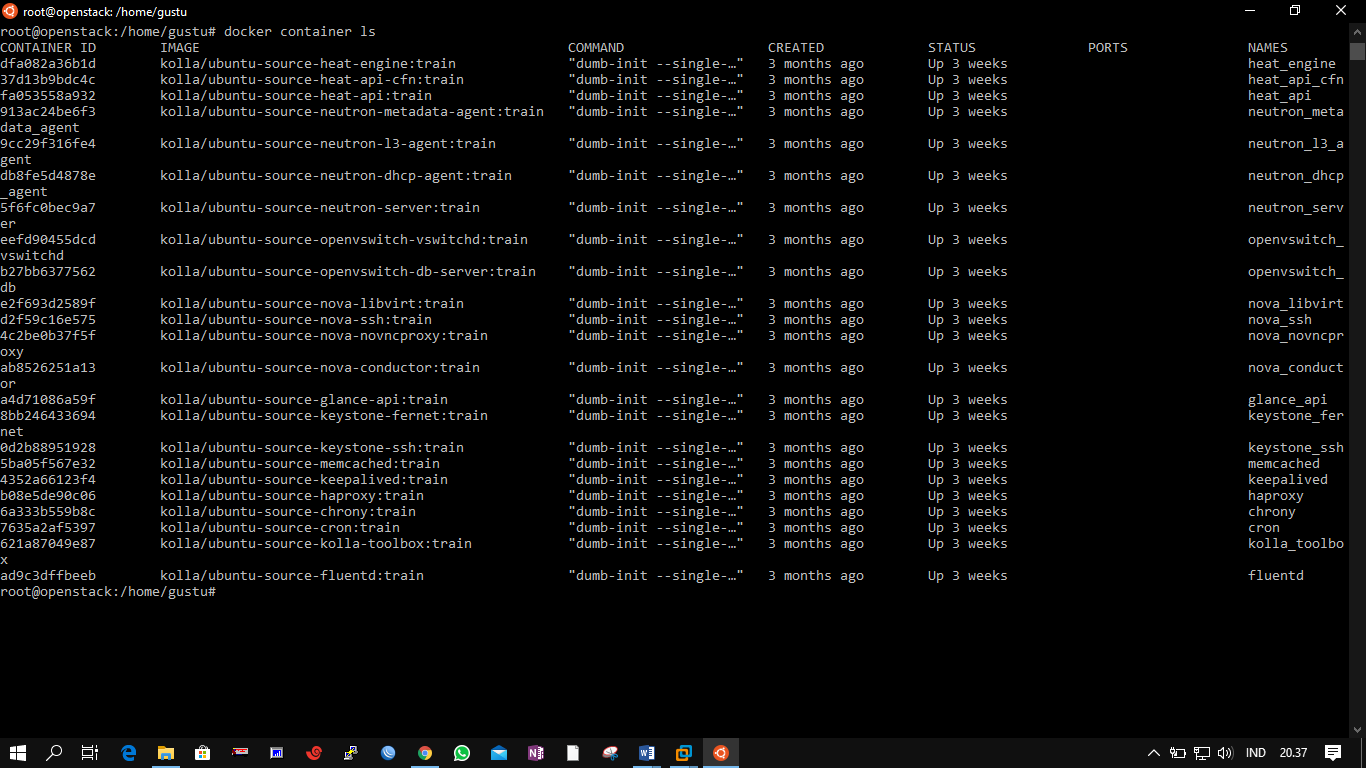
|  |
| --- |
| def profil\_save():  data = request.json['data']  hashed\_password = generate\_password\_hash(data["password"], method='sha256')  user = User.query.filter\_by(username=data["username"]).first()  user.email = data["email"]  user.password = hashed\_password  db.session.merge(user)  db.session.commit()  json\_data = {"status": True}  response = app.response\_class(  response=json.dumps(json\_data),  mimetype='application/json'  )  return response |

## Implementasi Sistem

Sistem membutuhkan server Openstack sebagai infrastruktur layanan cloud computing. Openstack merupakan dasar sistem yang digunakan untuk membangun sebuah sistem cloud. Openstack memberikan kemampuan untuk mengabstraksi sumber daya seperti komputasi ( compute ), penyimpanan ( storage ), dan sumber jaringan ( networking resource ), yang disatukan menjadi kumpulan sumber daya yang nantinya akan dibagi ke pengguna dengan aman. Openstack merupakan software server yang bersifat open source, dan pengembangan – pengembangan aplikasi dengan openstack sebagai inti sistem sangat terbuka lebar dengan disediakannya API untuk pengembangan sistem yang berjalan di atas sistem openstack.

Pada penelitian ini. Openstack yang dijalankan untuk mendukung kebutuhan pengembangan sistem menggunakan framework tools dari openstack yaitu Kolla-Ansible, framework ini memudahkan dalam pengembangan openstack yang dapat disesuaikan dengan layanan yang dibutuhkan dan komponen infrastruktur dalam docker containers.





Tampilan awal openstack

Dalam server yang digunakan, server openstack dapat diakses melalui web interface. Openstack menyediakan pilihan pengembangan sistemnya dapat melalui CLI ( Command-Line Interface) maupun melalui web interface. Namun untuk pengembangan yang lebih dalam disarankan menggunakan CLI, pada penelitian ini bisa dikembangankan dengan openstack horizon / openstack melalui web interfaces.



Menu awal untuk masuk ke dalam sistem openstack diharuskan login terlebih dahulu. Untuk dapat login openstack secara otomatis menggenerate kata sandi untuk kepentingan keamanan. Namun kita dapat mengubahnya bila dirasa sangat sulit diingat. Untuk dapat mengngetahui kata sandi dari openstack dapat diakses melalui command-line

Dashboard Openstack

Bila username dan kata sandi benar, maka akan diarahkan ke menu dashboard. Pada menu dashboard kita dapat mengetahui berapa maksimun user yang dapat dibuat oleh server openstack, berapa core CPU yang masih tersedia dan dapat digunakan dalam pembuatan instance / virtual machine.



Membuat Project untuk pengembangkan sistem

pada menu Identity, peneliti membuat project baru pada sistem yang berjalan di atas server openstack. Sehingga bila terjadi sesuatu pada sistem yang dibuat tidak mempengaruhi server openstack secara keseluruhan, dan dikemudian hari ada penelitian lebih lanjut dalam pengembangan cloud computing, bisa dilakukan pada nama project yang berbeda.



Jadi server openstack secara keseluruhan tidak terganggu atau mengalami masalah bila salah satu project mengalami masalah dalam pengembangannya. Administrator hanya perlu menghapus projectnya tanpa perlu menginstall ulang sistem ataupun server openstack.

Pengaturan security groups

Untuk menu Security Group, berguna untuk memberi akses port yang bisa digunakan oleh instance. Port yang digunakan oleh instance tergantung kebutuhan yang perlukan, karena setiap instamce memungkinkan menggunakan port yang berbeda – beda. Mungkin saja ada instance yang dibuat untuk penyimpanan aplikasi, ataupun mungkin digunakan untuk jadikan aplikasinya dapat diakses secara publik.



Topologi jaringan openstack dengan instance

Pada menu Network Topology, berisi informasi network IP yang yang instance peroleh, dan di jaringan mana instance terhubung. Pada server openstack secara otomatis setiap membuat project baru akan dialihkan ke jaringan private network, sehingga dalam mengembangkan sebuah project setiap user mendapat alokasi ip private masing – masing dan secara otomatis dilakukan NAT (Network Address Translation) oleh server openstack sehingga meski mendapatkan ip private dari server openstack dapat tetap mendapatkan akses ke jaringan internet, namun tidak dapat digunakan meremote instance melalui jaringan public.



Untuk dapat menjadikan instance yang dibuat dapat diremote melalui jaringan public. Pada konfigurasi jaringan penelitian ini dipilih yang public atau dalam hal ini bernama int-ext.

API Openstack



## Antar Muka Sistem

Implementasi antarmuka sistem pada penelitian ini bangun dengan menggunakan bahasa pemrograman python dengan framework flask. Pada sub-bab ini akan menampilkan implementasi antarmuka website, diantaranya landing page, login, dashboard, menu membuat instance baru, dan daftar instance yang telah dibuat, menu profil. Berikut ini adalah pemaparannya :

Pada penelitian ini, pengguna baru harus melakukan registrasi untuk dapat masuk ke dalam sistem,

Setelah melakukan pendaftaran, pengguna akan diarahkan ke form login, selanjutnya pengguna dapat memasukan username dan password yang sudah di buat pada form registrasi



Setelah proses login, pengguna akan diarahkan langsung ke dashboard menu. Pada dashboard menu, selanjutnya membuat virtual mesin baru pada menu “project”, kemudian pilih “buat baru”



Setelah dipilih akan muncul menu seperti pada gambar , selanjutnya pengguna memasukan nama project. Nama project nantinya akan menjadi nama instance baru sebagai hostname, untuk deskripsi bersifat opsional, dapat ditambahkan keterangan ataupun tidak. Lalu pilih siste operasi yang akan digunakan sebagai server pada lingkungan aplikasi yang akan dibuat. Lalu pilih CPU, CPU disini berisi berapa core yang di butuhkan untuk lingkungan sistem yang akan dibangun oleh pengguna, seberapa besar penyimpanan dan memori RAM yang dibutuhkan untuk mengembangkan sistem yang akan dibuat. Lebih jelasnya dapat dilihat pada gambar berikut



Pada proses yang di tampilkan pada gambar , pada backend dibuat fungsi untuk mendapatkan list CPU,RAM dan Storage didefinisikan dengan flavor, dengan mengauthentikasikan terlebih dahulu dengan API openstack menggunakan osconn. Dilanjutkan dengan fungsi yang akan ditampilkan, compute.flavors() untuk menampilakn list cpu ram dan storage, compute.images() untuk menampilkan list sistem operasi yang disediakan.

|  |
| --- |
| def getflavor():  flavors = osconn.compute.flavors()  for flavor in osconn.compute.flavors():  print(flavor)  json\_data = [flavor]  itr = True  while itr:  try:  json\_data.append(flavors.next().to\_dict())  except:  itr = False  # print(json\_data)  return json\_data  # fungsi untuk melihat list sistem operasi yang tersedia  def getimages():  images = osconn.compute.images()  for image in osconn.compute.images():  print(image)  json\_data = [image]  itr = True  while itr:  try:  json\_data.append(images.next().to\_dict())  except:  itr = False  # print(json\_data)  return json\_data |

Pada fungsi getproject() digunakan untuk mencetak semua list instance / virtual mesin yang sudah dibuat. Keluaran dari fungsi berikut ini berupa json selanjutnya dapat di olah kembali untuk ditampilkan di front-end.

|  |
| --- |
| def getproject():  servers = osconn.compute.servers()  for server in osconn.compute.servers():  print(server)  json\_data = [server]  itr = True  while itr:  try:  json\_data.append(servers.next().to\_dict())  except:  itr = False  # print(json\_data)  return json\_data |

Pada fungsi di bawah ini merupakan fungsi untuk membuat instance baru dengan menampilkan kebutuhan flavor mesin dan sistem operasi yang akan di install pada instance lalu untuk menyimpan konfigurasi diberi nama sesuai user yang sedang login, dalam hal ini user dapat mengetahui berapa banyak instance yang sudah dibuatnya.

|  |
| --- |
| def buatbaru(): json\_data = {} json\_data.update({"flavors": getflavor()}) json\_data.update({"images": getimages()})  return render\_template('buatbaru.html',json\_data = json\_data, level=level, name= current\_user.username) |

Pada segi antarmuka terlihat pada gambar dibawah tampilan proses dalam membuat instance baru. Pada sistem operasi hanya menampilkan satu sistem operasi saja dikarenakan pada server openstack hanya ada 1 sistem operasi yang dibuat. Begitu pula dengan CPU, RAM dan penyimpanan yang tersedia.



Pada gambar dibawah ini merupakan server openstack sebelum proses pembuatan instance baru,



Selanjutnya ketika melakukan klik pada tombol buat dan simpan project, sistem aplikasi akan mengirimkan perintah untuk membuat sebuah instance baru pada server openstack dengan menggunakan API dan ansible playbook yang sudah dibuat pada tabel berikut ini.

|  |
| --- |
| def project\_save():  form=ProjectForm()  osdesc = ""  images = getimages()  for image in images:  if form.sistem\_operasi.data == image["id"]:  osdesc = image["name"]  break  cpudesc = ""  flavors = getflavor()  cduname = ""  for flavor in flavors:  if form.cpu.data == flavor["id"]:  cpudesc = "Storage= "+str(flavor["disk"])+" GB, vCPU= "+str(flavor["vcpus"])+", RAM= "+str(flavor["ram"])+" MB"  cpuname = str(flavor["name"])  break  new\_project = Project(  project=form.namaproject.data,  description=form.deskripsi.data,  idos=form.sistem\_operasi.data,  osdesc=osdesc,  idcpu=form.cpu.data,  cpudesc=cpudesc,  user=current\_user.username  )  db.session.add(new\_project)  db.session.commit()  fin = open("yaml/deploy.yaml", "rt")  data = fin.read()  data = data.replace('<name>', form.namaproject.data)  data = data.replace('<image>', osdesc)  data = data.replace('<flav>', cpuname)  fin.close()  fin = open("yaml/deploy\_"+current\_user.username+".yaml", "wt")  fin.write(data)  fin.close()  p = subprocess.Popen("ansible-playbook " + "yaml/deploy\_"+current\_user.username+".yaml", shell=True, stdout=subprocess.PIPE)  stdout, stderr = p.communicate()  project = None  if level == 0:  project = Project.query.all();  else:  project = Project.query.filter\_by(user=current\_user.username).all();  return render\_template('projectlist.html', project=project, level=level) |

API openstack digunakan untuk mengambil informasi dari nama sebuah project lalu selanjutnya dari nama tersebut selanjutnya akan dibuat file dengan ekstensi yaml (.yaml) dengan baris perintah

|  |
| --- |
| fin = open("yaml/deploy\_"+current\_user.username+".yaml", "wt")  fin.write(data). |

Setelah file yaml selesai dibuat, sistem akan langsung mengeksekusi file yaml menggunakan perintah ansible playbook, didefinsinikan dengan p untuk memanggil perintah eksternal dari python.

|  |
| --- |
| p = subprocess.Popen("ansible-playbook " + "yaml/deploy\_"+current\_user.username+".yaml", shell=True, stdout=subprocess.PIPE)  stdout, stderr = p.communicate() |

Setelah proses ansible dijalankan, proses debug akan muncul seperti gambar dibawah ini.keluaran data yang ditampilkan oleh hasil debug pada terminal berupa json.



Ketika proses pembuatan instance selesai, sistem akan otomatis mengarahkan ke list instance yang sudah kita buat, seperti pada gambar berikut ini



Pada proses ini sistem hanya akan membuat instance baru dan menginstall sistem operasi saja, bila pengguna ingin menginstall paket XAMPP / LAMP, pengguna hanya perlu memberi centang pada menu xampp dan selanjutnya pilih menu install. Maka sistem akan melakukan instalasi XAMPP pada instance server pengguna secara otomatis dengan menggunakan ansible.





Pada gambar di atas tampilan dari server openstack ketika berhasil membuat sebuah instance melalui website yang digunakan, dibutuhkan waktu beberapa menit untuk membuat instance sampai instance tersebut dapat siap digunakan oleh para pengguna untuk mengembangkan aplikasinya. Untuk dapat mengakses instance yang dibuatnya, pengguna dapat melakukannya melalui SSH ke ip address instance yang dibuat.

|  |
| --- |
| def project\_del():  data = request.json['data']  project = Project.query.filter\_by(id=data["id"]).first()  fin = open("yaml/delete.yaml", "rt")  data = fin.read()  data = data.replace('<name>', project.project)  fin.close()  fin = open("yaml/delete\_"+current\_user.username+".yaml", "wt")  fin.write(data)  fin.close()  p = subprocess.Popen("ansible-playbook " + "yaml/delete\_"+current\_user.username+".yaml", shell=True, stdout=subprocess.PIPE)  stdout, stderr = p.communicate()  db.session.delete(project)  db.session.commit()  json\_data = {"status": True}  response = app.response\_class(  response=json.dumps(json\_data),  mimetype='application/json'  )  return response |

Pada source code diatas digunakan untuk menghapus intance yang sudah di buat oleh pengguna, seluruh data yang ada pada instance akan di hapus secara menyeluruh. Penting untuk melakukan backup berkala dan melakukan pengecekan secara menyeluruh sebelum menghapus instance.

## Implementasi kebutuhan Ansible

|  |
| --- |
| osconn = connection.Connection(auth\_url=os.environ['OS\_AUTH\_URL'],  project\_name=os.environ['OS\_PROJECT\_NAME'],  username=os.environ['OS\_USERNAME'],  password=os.environ['OS\_PASSWORD'],  user\_domain\_id="default",  project\_id=os.environ['OS\_PROJECT\_ID'],  project\_domain\_id=os.environ['OS\_PROJECT\_DOMAIN\_ID'],  compute\_api\_version='2.1',  verify=False) |

Pada source code ini, merupakan fungsi untuk melalukan authentikasi ke openstack agar dapat menjalankan API yang disediakan pada server Openstack. Fungsi ini akan di eksekusi di awal sehingga sebelum menjalankan sistem yang dibuat, administartor server harus menjalankan file openrc.sh yang yang berisi data authentikasi yang dibutuhkan agar dapat berkomunikasi dengan server Openstack.

|  |
| --- |
| export OS\_AUTH\_URL=http://192.168.1.10:5000 # With the addition of Keystone we have standardized on the term \*\*project\*\*  # as the entity that owns the resources.  export OS\_PROJECT\_ID=084aa4c02ba3453389695df1f9310f49  export OS\_PROJECT\_NAME="myclouds"  export OS\_USER\_DOMAIN\_NAME="Default"  if [ -z "$OS\_USER\_DOMAIN\_NAME" ]; then unset OS\_USER\_DOMAIN\_NAME; fi  export OS\_PROJECT\_DOMAIN\_ID="default"  if [ -z "$OS\_PROJECT\_DOMAIN\_ID" ]; then unset OS\_PROJECT\_DOMAIN\_ID; fi  # unset v2.0 items in case set  unset OS\_TENANT\_ID  unset OS\_TENANT\_NAME  # In addition to the owning entity (tenant), OpenStack stores the entity  # performing the action as the \*\*user\*\*.  export OS\_USERNAME="admin"  # With Keystone you pass the keystone password.  echo "Please enter your OpenStack Password for project $OS\_PROJECT\_NAME as user $OS\_USERNAME: "  read -sr OS\_PASSWORD\_INPUT  export OS\_PASSWORD=$OS\_PASSWORD\_INPUT  # If your configuration has multiple regions, we set that information here.  # OS\_REGION\_NAME is optional and only valid in certain environments.  export OS\_REGION\_NAME="RegionOne"  # Don't leave a blank variable, unset it if it was empty  if [ -z "$OS\_REGION\_NAME" ]; then unset OS\_REGION\_NAME; fi  export OS\_INTERFACE=public  export OS\_IDENTITY\_API\_VERSION=3 |

Pada baris kode diatas merupakan isi dari dari file myclouds-openr.sh. File myclouds-openr.sh diperoleh dari server openstack ketika admin ingin mengembangkan sistem openstack berjalan pada sistem pihak ketiga dengan menggunakan API yang tersedia di server openstack. File ini dijalankan sebelum program siap dijalankan agar API dapat digunakan.

|  |
| --- |
| - name: Deploy on OpenStack  hosts: localhost  gather\_facts: false  tasks:  - name: Deploy an instance  os\_server:  state: present  name: <name>  image: <image>  key\_name: keyspair  wait: yes  flavor: <flav>  auto\_floating\_ip: yes  network: int-ext  meta:  hostname: webserver.localdomain |

Pada source code diatas digunakan untuk membuat sebuah instance. Dikarenakan server openstack dan web aplikasi berjalan pada sistem yang sama host yang dituju mengarah pada localhost. Localhost disini merupakan ip server openstack. Selanjutnya menambahkan perintah yang akan di eksekusi pada baris task, baris task berisi kebutuhan yang di perlukan dalam membuat instance baru dengan menambahkan baris API yang terhubung ke server openstack untuk mendeploy instance baru, perintah tersebut bernama os\_server.

|  |
| --- |
| - name: Deploy on OpenStack  hosts: localhost  gather\_facts: false  tasks:  - name: Deploy an instance  os\_server:  state: present  name: tesf  image: ubuntu 16.04\_x86  key\_name: keyspair  wait: yes  flavor: small.1  auto\_floating\_ip: yes  network: int-ext  meta:  hostname: webserver.localdomain |

Pada baris kode ini merupakan contoh perintah yang dibuat secara otomatis dan langsung di eksekusi oleh ansible dengan mengirimkan perintah tersebut ke server openstack. Selanjutnya openstack membuat instance baru dengan nama instance dan sepesifikasi server sesuai baris kode yang dikirimkan ansible ke API openstack.

|  |
| --- |
| - name: remove an instance  hosts: localhost  tasks:  - name: remove an instance  os\_server:  name: <name>  state: absent |

Pada baris kode diatas merupakan barisk kode yang digunakan untuk menghapus instance server yang dibuat oleh para pengguna bila sudah tidak dibutuhkan. Baris kode memanggil API os\_server dan mencari nama instance yang akan dihapus, selanjutnya akan akan kembali di eksekusi oleh baris kode ... dan bari kode db.session.delete(project) langsung menghapus isi dalam database sesuai nama project yang dibuat.

|  |
| --- |
| - name: install apache & php  remote\_user: ubuntu  hosts: all  become: true  become\_user: root  gather\_facts: true  tasks:  - name: Update apt-get repo and cache  apt: update\_cache=yes force\_apt\_get=yes cache\_valid\_time=3600  - name: "Install apache2"  package: name=apache2 state=present  - name: "Install apache2-php5"  package: name=libapache2-mod-php state=present  - name: "Install php-cli"  package: name=php-cli state=present  - name: "Install php-mcrypt"  package: name=php-mcrypt state=present  - name: "Install php-gd"  package: name=php-gd state=present |

|  |
| --- |
| - name: Install MySQL for production ready server  user: ubuntu  hosts: all  become: True  become\_user: root  vars:  MySQL\_root\_pass: root  tasks:  - name: Update apt-get repo and cache  apt: update\_cache=yes force\_apt\_get=yes cache\_valid\_time=3600  - name: Set MySQL root password before installing  debconf: name="mysql-server" question="mysql-server/root\_password" value="{{MySQL\_root\_pass | quote}}" vtype="password"  - name: Confirm MySQL root password before installing  debconf: name="mysql-server" question="mysql-server/root\_password\_again" value="{{MySQL\_root\_pass | quote}}" vtype="password"  - name: test1  apt: package={{ item }} state=present force=yes update\_cache=yes cache\_valid\_time=3600  when: ansible\_os\_family == "Debian"  with\_items:  - mysql-server  - mysql-client  - python-mysqldb  - name: Deletes anonymous MySQL server user for localhost  mysql\_user: user="" state="absent" login\_password="{{ MySQL\_root\_pass }}" login\_user=root  - name: Secures the MySQL root user  mysql\_user: user="root" password="{{ MySQL\_root\_pass }}" host="{{ item }}" login\_password="{{MySQL\_root\_pass}}" login\_user=root  with\_items: - 127.0.0.1 - localhost - ::1  - "{{ ansible\_fqdn }}"  - name: Removes the MySQL test database  mysql\_db: db=test state=absent login\_password="{{ MySQL\_root\_pass }}" login\_user=root |

|  |
| --- |
| - name: install LAMP Stack  hosts: all  remote\_user: ubuntu  become: true  become\_user: root  gather\_facts: true  - name: Include Apache  import\_playbook: apache.yaml  - name: Include MySQL  import\_playbook: mysql.yaml |





Pada gambar diatas, sistem membuat alamat ip address dibuat pada file hosts digunakan untuk mengenali server mana yang akan dieksekusi ketika file yaml dijalankan, serta menyertakan username dan password dari instance yang telah dibuat secara default melalui openstack.

|  |
| --- |
| [webserver] 192.168.1.234 ansible\_user=ubuntu ansible\_ssh\_private\_key\_file=/mnt/c/Users/Administrator/Downloads/keyspair.pem |



Bila sukses melakukan instalasi pada instance akan muncul pada gambar di atas semua hasil debug tidak ada kegagalan ketika instalasi.



Untuk menguji webserver yang telah kita buat dapat mengakses ip address yang di miliki instance melalui web browser, bila muncul seperti gambar di atas web server yang di buat dinyatakan berhasil melakukan instalasi.



## Pengujian Sistem

Pada penelitian ini pengujian sistem “Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) Dengan API Openstack”, pengujian akan di lakukan dengan metode black box testing dan performance testing. Pengujian dilakukan dengan tujuan aplikasi yang dibuat dapat berjalan sesuai kebutuhan pengguna.

### Black Box Testing

Black Box Testing merupakan pengujian fungsional dari sistem yang dibuat. Tujuan dari pengujian fungsional adalah untuk memvalidasi perilaku perangkat lunak yang dibuat terhadap fungsionalitas kebutuhan para pengguna.

|  |  |  |  |
| --- | --- | --- | --- |
| **Kode Uji** | **Butir Uji** | **Teknik Pengujian** | **Hasil Pengujian** |
| OS1 | Pengguna baru melakukan pendaftaran untuk dapat masuk ke dalam sistem | Black Box | Diterima |
| OS2 | Pengguna melakukan login ke dalam sistem | Black Box | Diterima |
| OS3 | Admin dan pengguna memiliki antarmuka yang berbeda. | Black Box | Diterima |
| OS4 | Pengguna dapat membuat instance (server virtual) baru untuk pertama kali, | Black Box | Diterima |
| OS5 | Pengguna dapat membuat kembali instance (server virtual) baru. | Black Box | Diterima |
| OS6 | Pengguna dapat melihat list instance yang telah dibuat. | Black Box | Diterima |
| OS7 | Pengguna dapat mengakses instance melalui SSH dengan menggunakan SSHkey. | Black Box | Diterima |

### Performance Testing

Pada pengujian *Performance Testing* merupakan lanjutan dari pengujian fungsionalitas sistem yang sudah dibuat. Performance Testing bertujuan memvalidasi “kecepatan” sistem yang dibuat. Kecepatan dalam hal ini peneliti melakukan pengujian dengan mengukur aspek waktu respon sistem. Peneliti melakukan pengujian kinerja dalam ruang lingkup yang mendekati ruang lingkup sebenarnya. Pengujian kinerja sistem ini dicapai dengan melakukan simulasi menggunakan perangkat lunak aplikasi *Jmeter*.

# BAB V

**KESIMPULAN DAN SARAN**

## Kesimpulan

## Saran

# DAFTAR PUSTAKA

Anggeriana, H. (2011). *Cloud Computing.*

Ansible. (t.thn.). *ansible.com*. Diambil kembali dari http://docs.ansible.com/ansible/latest/index.html: http://docs.ansible.com/ansible/latest/index.html

Collings, T., & Kurt , W. (2015). *Duties of the System Administrator. In Red Hat Linux Networking System Administrator (chap. 1).*

Corporation, Exabyte. (2004). *The Basic Backup Guide.* cororado.

Doty, S. (2008). *Python Basics.*

Gerald D. Everett, R. M. (2007). *Software Testing: Testing Across the Entire Software Development Life Cycle 1st Edition.* Canada: IEEE Press.

Lei Xiaojiang, S. Y. (2013). *The Design and Implementation of Resource Monitoring for Cloud Computing Service Platform*.

Liyun Zuo, L. S. (2015). *A Dynamic Self-adaptive Resource-Load Evaluation Method in Cloud Computing.*

Masse, M. (2012). REST API. Dalam *REST API Design Rulebook* (hal. 5). America: O’Reilly Media, Inc.

Mulyana, E. (2017, 10 05). *Pengantar Openstack*. Diambil kembali dari https://eueung.gitbooks.io: https://eueung.gitbooks.io/buku-komunitas-sdn-rg/content/index.html

Nicole Ng, H. C. (2011). *An Adaptive Threshold Method to Address Routing Issues in Delay-Tolerant Networks.*

Saleh, Y. W. (2013). *Adaptive Resource Management for Service Workflows in Cloud Environments.*

Sosinsky, B. (2011). *Cloud Computing Bible.* Canada: Wiley Publishing, Inc.

Xia, Q., Lan, Y., & Xiao, L. (2015). *A Heuristic Adaptive Threshold Algorithm on IaaS Clouds.*