

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

**SKRIPSI**



**IDA BAGUS RATHU EKA SURYA WIBAWA  
1308605045**

**PROGRAM STUDI INFORMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS UDAYANA**

**JIMBARAN  
2021**

## **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 : guzekawibawa@gmail.com  
Nomor telp/HP : 081337482800  
Alamat : Jalan Pratu Made Rambug, Br. Sasih No. 49 Batubulan

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.

Badung, 29 Januari 2021

Yang membuat pernyataan,

Ida Bagus Rathu Eka Surya Wibawa

NIM. 1308605045

## 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 : 29 Januari 2021

Disetujui Oleh :

Pembimbing I

Penguji I

I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs.  
NIP. 19890127201212 1 001

I Gede Santi Astawa, S.T., M.Cs.  
NIP. 19801206200604 1 003

Pembimbing II

Penguji II

I Komang Ari Mogi, S.Kom., M.Kom.  
NIP. 19840924200801 1 007

Gst. Ayu Vida Mastrika Giri, S.Kom., M.Cs.  
NIP. 199006062018112 3 001

Penguji III

Made Agung Raharja, S.Si., M.Cs.  
NIP. 198509192013012 2 003

Mengetahui,  
Ketua Jurusan Ilmu Komputer  
FMIPA Universitas Udayana  
Ketua,

Dr. Ir. I Ketut Gede Suhartana, S.Kom., M.Kom  
NIP. 19720110200812 1 001

Judul : Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack  
Nama : Ida Bagus Rathu Eka Surya Wibawa (NIM : 1308605045)  
Pembimbing : 1. I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs.  
2. I Komang Ari Mogi, S.Kom., M.Kom.

## ABSTRAK

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 developer dalam melakukan komputasi tanpa harus melakukan instalasi aplikasi pada komputer, developer hanya perlu mengaksesnya melalui internet. Cloud computing memiliki beberapa fasilitas yang dapat dipilih oleh developer sesuai kebutuhan developer seperti Infrastructure as a Service(IaaS), Platform as a Service(PaaS). Serta Software as a Service(SaaS).

Pengembangan Platform as a Service sebagai salah satu teknologi cloud computing yang dapat digunakan oleh pengembang aplikasi untuk mengembangkan aplikasi yang akan dibuat tanpa perlu menyediakan infrastruktur, database, framework aplikasi dan lain sebagainya serta bersifat dinamis. Dalam pengembangan layanan cloud yang dikelola oleh seorang sistem administrator atau developer, tugas menginstalasi dan menkonfigurasi sistem pada server maupun software aplikasi dilakukan dengan otomatis dengan menggunakan platform otomatisasi sistem linux untuk efisiensi waktu dan manajemen penggunaanya dengan lebih mudah.

**Kata kunci:** *Cloud Computing, Platform as a Service, automation, ansible, API, Openstack*

Judul : Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack  
Nama : Ida Bagus Rathu Eka Surya Wibawa (NIM : 1308605045)  
Pembimbing : 1. I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs.  
2. I Komang Ari Mogi, S.Kom., M.Kom.

### **ABSTRACT**

*Cloud computing is a technology that is currently starting to develop in many information technology activities. Cloud computing is a computing model in which all resources in cloud services run on the internet network media. With cloud computing, it makes it easier for users to perform computations without having to install applications on computers, users only need to access them via the internet. Cloud computing has several facilities that can be selected by users according to user needs, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS). As well as Software as a Service (SaaS).*

*Development of Platform as a Service as a cloud computing technology that can be used by application developers to develop applications that will be created without the need to provide infrastructure, databases, application frameworks and so on and are dynamic. In developing cloud services that are managed by a system administrator or sysadmin, the task of installing and configuring systems on servers and application software is carried out automatically using the Linux system automation platform for time efficiency and easier management of users.*

**Key Words : Cloud Computing, Platform as a Service, automation, ansible, API, Openstack.**

## **KATA PENGANTAR**

Penelitian dengan judul " Sistem Manajemen Layanan Web Berbasis Platform as a Service (PaaS) dengan API Openstack " ini disusun dalam rangkaian kegiatan pelaksanaan Tugas Akhir di Jurusan Ilmu Komputer Universitas Udayana.

Selama menyelesaikan penelitian tugas akhir ini, penulis telah banyak menerima bimbingan, pengarahan, petunjuk dan saran. Untuk itu penulis mengucapkan terima kasih yang sebesar-besarnya kepada yang terhormat:

1. Bapak I Dewa Made Bayu Atmaja Darmawan, S.Kom., M.Cs. selaku Pembimbing I dan Bapak I Komang Ari Mogi, S.Kom., M.Kom. selaku Pembimbing II yang telah banyak membimbing dan membantu menyempurnakan penelitian ini.
2. Bapak Dr. I Ketut Gede Suhartana, S.Kom., M.Kom. selaku Ketua Jurusan Ilmu Komputer Universitas Udayana, Bapak I Gusti Ngurah Anom Cahyadi Putra, S.T., M.Cs. selaku Ketua Komisi Tugas Akhir Jurusan Ilmu Komputer Universitas Udayana yang telah banyak memberikan masukan dan motivasi sehingga memperlancar dalam proses pelaksanaan penelitian ini.
3. Bapak dan ibu dosen di Jurusan Ilmu Komputer Universitas Udayana yang bersedia meluangkan waktunya untuk memberikan masukan dalam penyempurnaan penelitian ini.
4. Rekan – rekan mahasiswa Jurusan Ilmu Komputer khususnya angkatan 2013 yang telah memberi dukungan, motivasi, semangat dan kerja sama dalam penelitian ini.

Penulis menyadari tugas akhir ini masih jauh dari kesempurnaan. Untuk itu dengan segala kerendahan hati, penulis mengharapkan kritik dan saran yang bermanfaat dari pembaca demi kesempurnaan tugas akhir ini.

Jimbaran, Januari 2021

Penulis

## DAFTAR ISI

SISTEM MANAJEMEN LAYANAN WEB BERBASIS <i>PLATFORM AS A SERVICE (PAAS)</i> DENGAN <i>API OPENSTACK</i> .....	i
SURAT PERNYATAAN KEASLIAN KARYA ILMIAH .....	i
LEMBAR PENGESAHAN TUGAS AKHIR .....	ii
ABSTRAK .....	iii
KATA PENGANTAR .....	v
DAFTAR ISI .....	vi
DAFTAR TABEL .....	ix
DAFTAR GAMBAR .....	x
DAFTAR LAMPIRAN .....	xii
BAB I PENDAHULUAN .....	2
1.1. Latar Belakang .....	2
1.2. Rumusan Masalah .....	3
1.3. Batasan Masalah .....	3
1.4. Tujuan Penelitian .....	4
1.5. Manfaat Penelitian .....	4
1.6. Metodologi Penelitian .....	4
1.6.1. Metode Pengumpulan Data .....	4
1.6.2. Kerangka Kerja Penelitian .....	5
1.6.3. Perancangan Sistem .....	6
1.6.4. Metode Pengembangan Perangkat Lunak .....	8
1.6.5. Rancangan <i>ERD (Entity Relationship Diagram)</i> .....	11
1.6.6. Evaluasi dan Validasi Hasil .....	11
BAB II TINJAUAN PUSTAKA .....	12
2.1. Tinjauan Empiris .....	12

2.2.	<i>Cloud Computing</i> .....	13
1.	<i>IaaS (Infrastructure as a Service)</i> .....	13
2.	<i>Platform as a Service (PaaS)</i> .....	14
3.	<i>Software as a Service (SaaS)</i> .....	14
2.3.	<i>Openstack</i> .....	15
2.4.	<i>REST API</i> .....	15
2.5.	<i>ANSIBLE</i> .....	16
BAB III ANALISIS DAN PERANCANGAN SISTEM .....		18
3.1.	Analisis Kebutuhan .....	18
3.1.1.	Kebutuhan Fungsional Sistem .....	18
3.1.2.	Kebutuhan Non Fungsional Sistem.....	19
3.2.	Penentuan Proses – proses dalam Operasi Sistem.....	19
3.3.	Perancangan Antarmuka Sistem.....	19
3.4.	Skenario Pengujian Sistem .....	22
3.4.1.	BlackBox Testing.....	22
3.4.2.	Performance Testing .....	23
BAB IV HASIL DAN PEMBAHASAN .....		24
4.1.	Gambaran Umum Sistem .....	24
4.2.	Lingkungan Perancangan dan Implementasi Sistem.....	24
4.3.	Implementasi Basis Data .....	25
4.4.	Implementasi Antarmuka .....	26
4.5.	Pengujian Sistem .....	38
4.5.1.	<i>Black Box Testing</i> .....	38
4.5.2.	<i>Performance Testing</i> .....	39
4.6.	Analisa.....	43



BABI V SIMPULAN DAN SARAN.....	45
5.1.    Simpulan.....	45
5.2.    Saran .....	45
DAFTAR PUSTAKA .....	46
LAMPIRAN.....	47

## DAFTAR TABEL

Tabel 3.1 Tabel Kebutuhan Fungsional .....	18
Tabel 3.2 Tabel Kebutuhan Non Fungsional .....	19
Tabel 3.3. Tabel Pengujian Black Box.....	22
Tabel 4.1. Hasil Pengujian ke-1 .....	38
Tabel 4.5. Hasil Pengujian ke-1 .....	41
Tabel 4.6. Hasil Pengujian ke-2 .....	42
Tabel 4.7. Hasil Pengujian ke-3 .....	43

## DAFTAR GAMBAR

Gambar 1.1. Kerangka kerja penelitian.....	5
Gambar 1.2. Desain Kerja Sistem .....	7
Gambar 1.3. <i>Flowchart</i> Konfigurasi Otomatis .....	9
Gambar 1.4. <i>Flowchart Front-end</i> Sistem .....	10
Gambar 1.5. Rancangan <i>Entity Relationship Diagram</i> .....	11
Gambar 3.1. Rancangan <i>Landing Page</i> .....	20
Gambar 3.2. Rancangan <i>form</i> pendaftaran.....	20
Gambar 3.3. Rancangan <i>form login</i> .....	21
Gambar 3.4. Rancangan membuat <i>instance server</i> baru.....	21
Gambar 4.1. Tabel basis data <i>user</i> .....	25
Gambar 4.2. Tabel basis data <i>project</i> .....	25
Gambar 4.3. Tampilan Menu Awal Openstack.....	26
Gambar 4.4. Tampilan Dashboard Openstack .....	27
Gambar 4.5. Tampilan Menu Detail .....	27
Gambar 4.6. Tampilan Menu Deteksi .....	28
Gambar 4.7. Tampilan Pilih File.....	28
Gambar 4.8. Tampilan Server .....	29
Gambar 4.9. <i>Login Form</i> pada <i>website</i> .....	30
Gambar 4.10. Buat Project Baru .....	30
Gambar 4.111. Proses Pemilihan Sistem Operasi.....	31
Gambar 4.122. Proses Pemilihan Flavor Server .....	32
Gambar 4.133. Tampilan pada Server Openstack.....	32
Gambar 4.144. Tampilan Proses Ansible.....	33
Gambar 4.155. Tampilan Virtual Server.....	33
Gambar 4.166. Tampilan Melakukan Instalasi XAMPP .....	34
Gambar 4.177. List Virtual Server pada Openstack .....	34
Gambar 4.188. Tampilan file YAML .....	35
Gambar 4.199. Set Host Virtual Server .....	35
Gambar 4.200. Instalasi XAMPP Berhasil .....	36

Gambar 4.211. Pengujian web Server.....	37
Gambar 4.222. Pengujian Database .....	37
Gambar 4.233. Persiapan proses simulasi.....	40
Gambar 4.244. Grafik Hasil Pengujian ke-1 .....	40
Gambar 4.255. Grafik Hasil Pengujian ke-2 .....	41
Gambar 4.266. Grafik Hasil Pengujian ke-3 .....	42

## DAFTAR LAMPIRAN

<i>Source Code Aplikasi</i> .....	47
-----------------------------------	----

## BAB I PENDAHULUAN

### 1.1. Latar Belakang

Pada era teknologi seperti sekarang ini, perkembangan teknologi sangat meningkat pesat, dimana para pengembang aplikasi saat ini dengan mudah membangun sebuah *web server* yang yang dapat dikerjakan secara bersama-sama dengan menggunakan komputasi awan ( *Cloud Computing*). Sebelum teknologi *cloud computing* ditemukan pengembang aplikasi berbasis *web* lebih sering mengembangkan aplikasi *web* secara lokal di perangkat komputer. Membuat *client* dari pengembang aplikasi *web* tidak dapat *memonitoring* atau menilai sejauh mana aplikasi web yang di buat oleh pengembang aplikasi *web* telah selesai, menyebabkan kurangnya dalam manajemen aplikasi *web*. Serta pada proses perilisan aplikasi web pun membutuhkan waktu yang tidak sebentar, banyak hal yang butuh diperhitungkan seperti penyimpanan data yang besar, ketersediaan *domain*, *data processing* dan masih banyak lainnya.

Menurut Ricky W. Griffin menyatakan bahwa manajemen adalah sebuah proses pengorganisasian, pengkoordinasian, perencanaan, dan pengontrolan sumber daya agar dapat mencapai sasaran (*goals*) secara efisien dan efektif. Efektif sendiri sebuah tujuan mampu dicapai sesuai dengan apa yang telah direncanakan. Perlunya manajemen aplikasi *web* yang dibuat untuk memudahkan pengembang aplikasi beserta tim pengembang lainnya dapat saling meninjau kebutuhan aplikasi *web* yang di buat secara bersama – sama walau berada di lokasi yang berbeda.

Konsep komputasi awan ini sudah banyak menarik minat industri digital maupun pendidikan. Solusi berbasis cloud sepertinya menjadi kunci bagi organisasi teknologi informasi yang memiliki permasalahan dengan keterbatasan anggaran (Teng & Magoules,2010). Komputasi awan merupakan paradigma yang baru dalam komputasi terdistribusi, menyajikan banyak konsep teknologi, ide, serta arsitektur yang disajikan secara *service-oriented*. Pada pemanfaatanya komputasi awan mengubah cara bagaimana layanan informasi disediakan maupun di sebar. Dengan memanfaatkan teknologi komputasi awan dalam manajemen aplikasi

web, *developer* / pengembang aplikasi web bekerja secara efektif dan efisien dari segi waktu maupun perangkat. Saat ini sudah banyak penyedia layanan *cloud computing* untuk para pengembang aplikasi seperti *Google Cloud Platform*, *Amazon Web Services*, *Alibaba Cloud Server*, bahkan untuk di Indonesia ada *Biznet GIO cloud*.

Pada Penelitian sebelumnya, Nishant Kumar Singh dkk, menjelaskan pada perkembangan *cloud computing* pada sisi infrastruktur menuntut penyebaran layanan yang cepat, pada waktu yang bersamaan kebutuhan penyediaan secara otomatis ikut berperan dalam pengembangan aplikasi melalui komputasi awan. Maka dari itu, dalam Tugas Akhir ini penelitian dilakukan dengan membangun sebuah Platform website berbasis *cloud* untuk pengembang aplikasi web dalam membangun sebuah website secara otomatisasi untuk meningkatkan efisiensi kinerja tim secara bersamaan mampu manajemen kebutuhan pengembangan aplikasi yang dibuat.

## **1.2. Rumusan Masalah**

Berdasarkan latar belakang masalah dapat dirumuskan rumusan masalah sebagai berikut :

- 2.1. Bagaimana otomatisasi dengan *Ansible* berkerja pada ruang lingkup komputasi awan.
- 2.2. Bagaimana membangun sistem yang memfasilitasi developer dalam mengembangkan *website* ke dalam komputasi awan.

## **1.3. Batasan Masalah**

Agar penelitian lebih fokus, permasalahan yang dicakup tidak akan terlalu luas dan akan sesuai dengan maksud dan tujuan yang ingin dicapai. Ruang lingkup yang dibahas dalam penelitian ini akan dibatasi hanya pada :

1. Sistem dibuat dalam bentuk purwa rupa.
2. Layanan komputasi awan menggunakan *Openstack*.
3. Sistem meyalani *virtual server* untuk membangun layanan aplikasi *web*.
4. Sistem dibangun menggunakan Bahasa permrograman *python*.

5. Sistem tidak menyediakan *domain* untuk layanan yang diterima *developer website*

#### **1.4. Tujuan Penelitian**

Berdasarkan rumusan masalah, tujuan dari penelitian ini adalah sebagai berikut :

1. Membantu *developer website* mengurangi beban resource komputer dalam mengembangkan aplikasi yang dibuat.
2. Membantu *developer website* membuat sebuah *server website* dan melakukan instalasi *web server* secara otomatis.
3. Membantu efektivitas *developer website* dengan membangun *virtual server* secara *online*, serta fleksibilitas dalam pengembangan website yang dibuat.

#### **1.5. Manfaat Penelitian**

Dengan dilakukannya penelitian ini, diharapkan mampu mengurangi beban kerja komputer *developer website* dengan komputasi awan yang menyediakan *virtual server web* dan sebuah *platform digital* dalam mengembangkan *website*.

#### **1.6. Metodologi 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*.

##### **1.6.1. Metode Pengumpulan Data**

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:

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

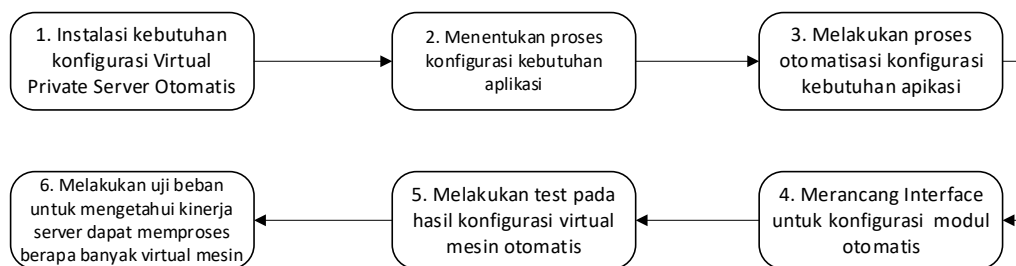


selanjutnya adalah menentukan rumusan masalah dan tujuan yang ingin dicapai dalam penelitian. Pada penelitian ini identifikasi permasalahan dilakukan dengan menggunakan teknik observasi, dari teknik ini maka akan dapat diketahui mengenai keluhan – keluhan yang ada di lapangan.

Tahap kedua yang dilakukan dalam metodologi 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..

### 1.6.2. Kerangka Kerja Penelitian

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



**Gambar 1.1. Kerangka kerja penelitian**

#### 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* .

#### 2. 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 *developer*.

3. Melakukan proses otomatisasi konfigurasi modul aplikasi

Pada tahap ini melakukan otomatisasi dalam mengkonfigurasi sebuah *web server*. Proses otomatisasi menggunakan Ansible.

4. Merancang *web front-end* untuk *developer website*.

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

5. Melakukan test pada hasil konfigurasi

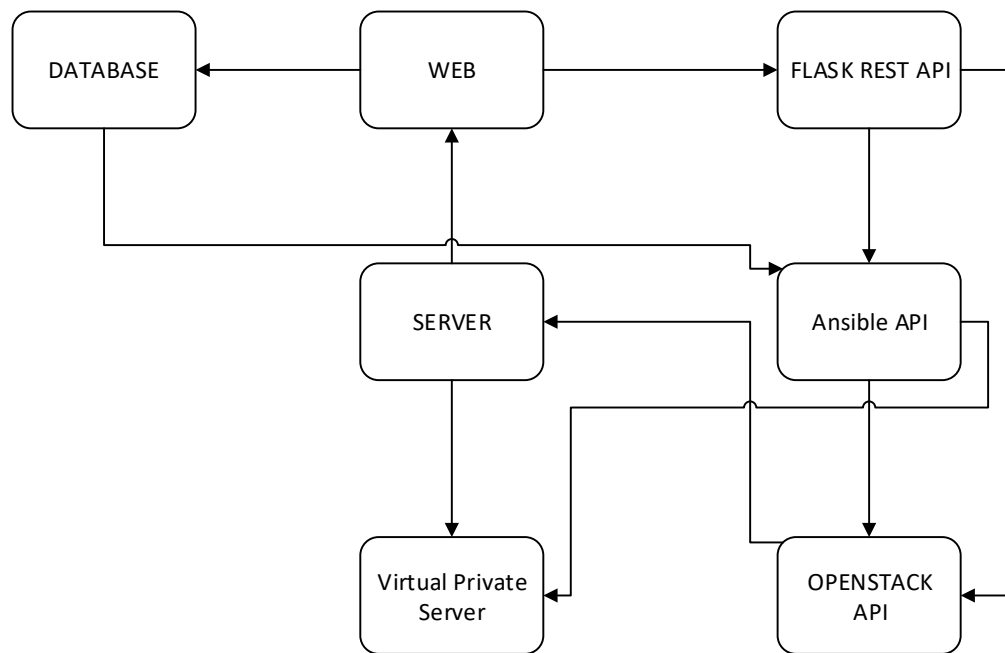
Pada tahap ini menguji hasil konfigurasi dari *web interface*. Untuk menguji konfigurasi berjalan dengan baik.

6. Melakukan uji beban *server*

beban ini dilakukan untuk menguji tingkat kemampuan *server* dalam memberikan pelayanan pada *developer* aplikasi.

### 1.6.3. 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 :

1. *Web dan database*

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

2. *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*.

3. *Database dan ansible API*

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

4. *framework flask REST API dan asible API*

Hubungan *framework flask REST API* dengan *ansible 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.

5. *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*.

6. *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.

7. *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. *Developer* hanya perlu memilih keperluan yang ada menu *web front-end*.

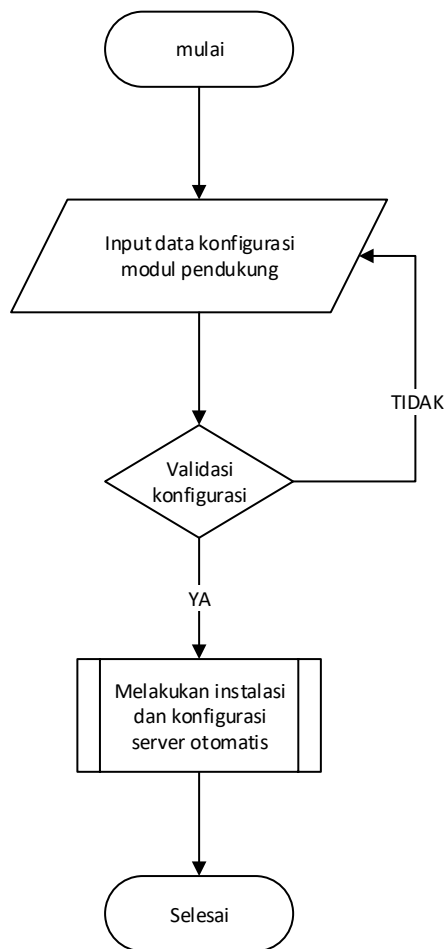
8. *server* dan *Virtual Private Server*

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

#### **1.6.4. Metode Pengembangan Perangkat Lunak**

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

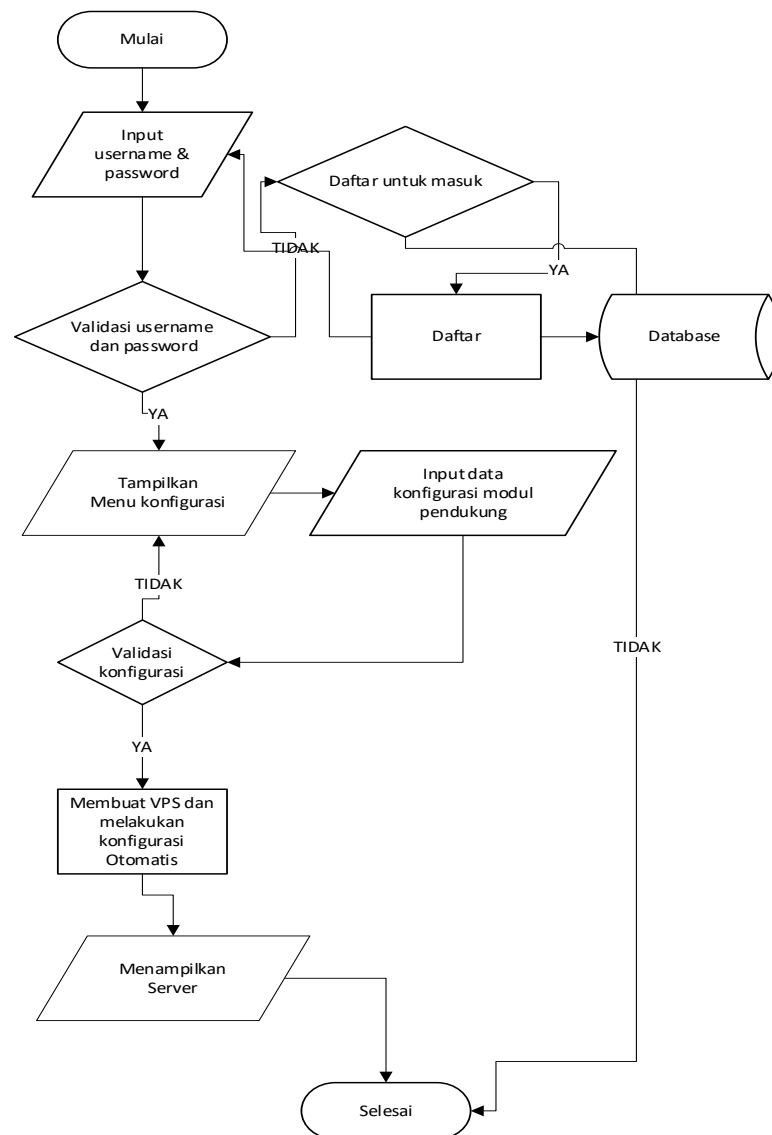
1. *Flowchart* Konfigurasi Otomatis



**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*, *username* 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 mengakses layanan aplikasi.

## 2. Flowchart Front-end Web



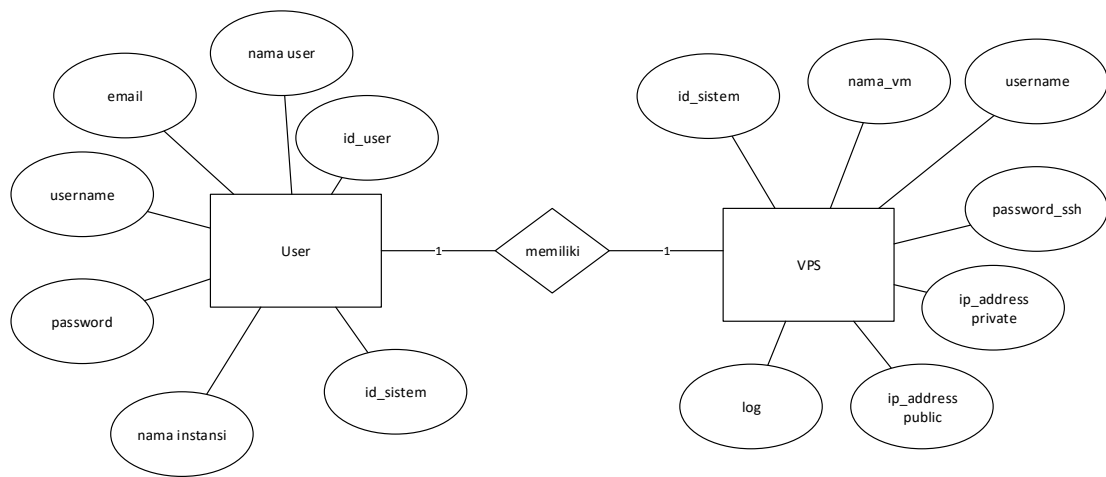
**Gambar 1.4. Flowchart Front-end Sistem**

Pada desain *flowchart Front-end Web* 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 berntuk *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 developer bila terjadi satu kesalahan dapat dilakukan *restore data*.

#### 1.6.5. Rancangan ERD (Entity Relationship Diagram)

Pada desain *Entity Relationship Diagram (ERD)* diatas menjelaskan hubungan user dengan VPS yang dimiliki user memiliki sebuah vps dengan satu ip public. Serta pada vps akan menyimpan log mulai dari kinerja dan resource yang dimiliki.



**Gambar 1.5. Rancangan Entity Relationship Diagram**

#### 1.6.6. Evaluasi dan Validasi Hasil

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

#### 2.1. Tinjauan Empiris

**a. *Unleashing Full Potential of Ansible Framework: University Labs Administration* (Pavel Masek, 2018)**

Pada penelitian ini, penulis menjelaskan desain infrastruktur merupakan fase siklus hidup perangkat lunak yang mendefinisikan dan mengkonfigurasi kebutuhan infrastruktur perangkat lunak tersebut serta jumlah dan jenis host fisik maupun mesin virtual. Mengikuti tren saat ini yang dikenal sebagai *Infrastructure-as-a-code* penulis membuat layer baru untuk kerangka kerja ansible sebagai kerangka kerja manajemen orkestrasi dan konfigurasi. *Ansible* menawarkan pengelolaan laboratoruim Universitas Brno Republik Ceko dari jaringan lokal maupun *publik*.

**b. *Implementation Of Configuration Management Virtual Private Server Using Ansible* (I Putu Hariyadi & Khairan Marzuki, 2020)**

Pada penelitian ini, penulis menjelaskan teknologi virtualisasi telah diterapkan oleh perguruan tinggi dengan program studi komputer untuk mendukung pembelajaran secara praktis dengan memberikan setiap pengguna *virtual private server (VPS)*, dengan teknologi container untuk mendukung kegiatan praktikum *network management (NM)*. Selanjutnya peneliti melakukan uji coba menggunakan ansible untuk melakukan otomatisasi manajemen jaringan diperoleh hasil *Modul cron Ansible* dapat digunakan untuk mengotomatisasi penjadwalan terkait start dan stop container secara periodik berdasarkan jadwal praktikum per kelompok, rata-rata waktu yang dibutuhkan untuk mengotomatisasi pembuatan objek *VPS* untuk 93 siswa adalah 40,7 menit berdasarkan 5 (lima) kali percobaan, berdasarkan waktu rata-rata tersebut dapat dihitung waktu untuk membuat satu *VPS* siswa benda adalah  $40,7 \text{ menit} = 2442 \text{ detik} / 93 = 26,25 \text{ detik}$ .



**c. *Implementation of Network Automation Using Ansible to Congure Routing Protocol in Cisco and Mikrotik Router with Raspberry PI ( Muhammad Fauzi Islami, Purnawarman Musa & Missa Lamsani, 2020 )***

Pada penelitian ini, penulis menggunakan *Ansible* sebagai otomasi jaringan yang merupakan terobosan baru dalam bidang teknologi khususnya dalam bidang rekayasa jaringan yang mulai memasuki dunia baru rekayasa jaringan itu sendiri, yaitu programabilitas dan otomasi jaringan. Tentunya hal ini sangat membantu para *network engineer* untuk mengurangi perjuangan mereka dalam kompleksitas infrastruktur jaringan yang ada. Terobosan semacam ini memastikan semua pengkila teknologi bahwa teknologi akan selalu berkembang dari waktu ke waktu. Kita harus bisa menyamakannya jika ingin tetap di IT.

## **2.2. 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 sumber daya virtual dan detail dari mesin fisik sistem dalam *software* yang berjalan secara abstraksi 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-jenis dari *Cloud Computing* dapat dijabarkan sebagai berikut:

### **1. *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.

## 2. *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 menanggung untung besar dari layanan *PaaS*

## 3. *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 Service Provider)*. 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 fitur – fitur aplikasi yang telah disediakan oleh penyedia saja yang dapat disewa oleh pelanggan

### **2.3. Openstack**

*OpenStack* adalah sistem aplikasi *cloud* yang mengelola sumberdaya seperti komputasi, penyimpanan dan jaringan, yg tersedia pada infrastruktur fisik seperti dalam sebuah fasilitas pusat-data (*data center*). *Admin* atau *sysadmin* 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 sysadmin 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 sysadmin kemampuan untuk sumber penyediaan melalui antarmuka web.

### **2.4. REST 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 menggunakan *Application Programming Interface (API)* untuk berkomunikasi dengan layanan *web*. Secara umum, API mengekspos seperangkat data dan fungsi untuk memfasilitasi interaksi antara program komputer dan memungkinkan mereka 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

## 2.5. ANSIBLE

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

Dengan *Ansible* *developer* 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 otomatisasi. Pada *Ansible* memerlukan *inventory* atau data server tujuan untuk dapat dilakukan otomatisasi. Pada penerapannya, *Ansible* menggunakan *playbook* dan *roles*, 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* yang 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 SISTEM

#### 3.1. 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 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 metodologi 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.

##### 3.1.1. Kebutuhan Fungsional Sistem

Kebutuhan fungsional sistem mencakup fungsi – fungsi yang mampu dilakukan oleh sistem. Kebutuhan fungsional sistem pada penelitian ini ditunjukkan pada tabel 3.1.

**Tabel 3.1 Tabel Kebutuhan Fungsional**

No.	Kebutuhan Fungsional	User
1	Membuat <i>virtual server</i> dengan <i>cloud</i>	√

2	Melakukan instalasi kebutuhan <i>package</i> dalam membangun <i>web server</i>	√
---	--	---

### 3.1.2. Kebutuhan Non Fungsional Sistem

Adapun kebutuhan non fungsional yang harus dipenuhi oleh sistem untuk melengkapi sistem secara keseluruhan, diantaranya sebagai berikut :

**Tabel 3.2 Tabel Kebutuhan Non Fungsional**

No.	Kebutuhan Non Fungsional	Keterangan
1	<i>Efficient</i>	Sistem mampu membantu kebutuhan pengembang aplikasi <i>web</i> secara tepat guna,
2	<i>User Friendly</i>	Sistem yang didukung dengan tampilan yang menarik dan mudah digunakan

### 3.2. Penentuan Proses – proses dalam Operasi Sistem

Setelah melakukan Teknik kajian pustaka pada tahap sebelumnya, secara garis besar proses-proses yang ada pada Sistem Manajemen Layanan *Web Berbasis Platform as a Service (PaaS)* dengan *API Openstack* adalah :

- Proses pembuatan *virtual server* secara otomatis dan sesuai dengan kebutuhan pengembang aplikasi *web*
- Proses *install package* secara otomatis dengan menggunakan *ansible*
- Proses memberikan *IP* secara otomatis dan kunci *private* untuk melakukan remote pada *virtual server cloud*.
- Proses pengembangan secara mandiri oleh pengembang aplikasi *web*, jika paket yang di butuhkan tidak ada dalam menu sistem.

### 3.3. Perancangan Antarmuka Sistem

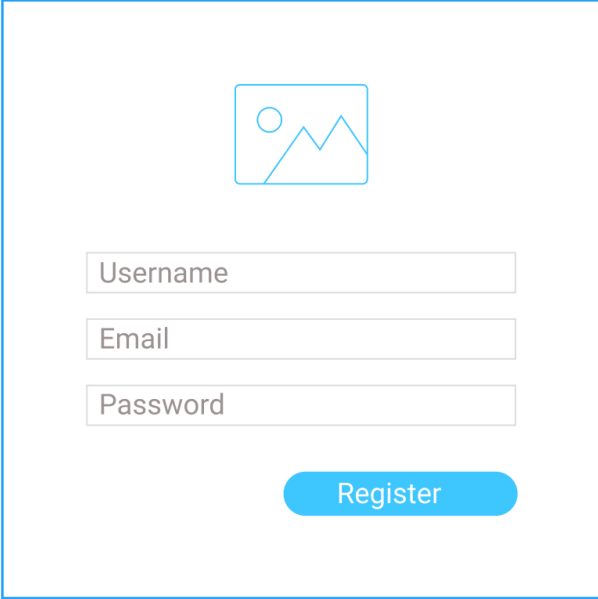
Pada tahap ini dijelaskan perancangan antarmuka sistem dalam memanajemen layanan web berbasis *Platform as a Service (PaaS)*, pada sistem

perancangan ini Sistem *administrator* dapat membuat dan mengembangkan websitenya pada sistem *cloud computing*.



**Gambar 3.1. Rancangan *Landing Page***

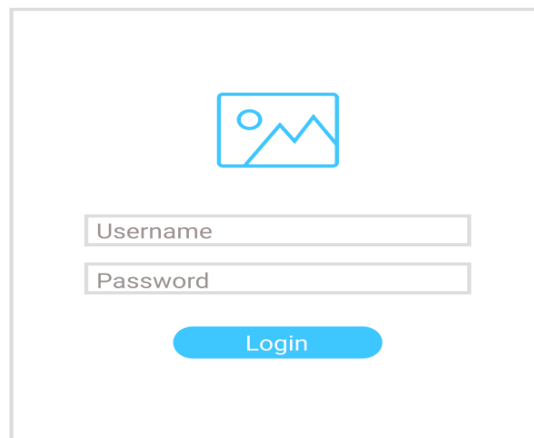
Gambar 3.1 merupakan rancangan landing page yang berisi informasi dari sistem cloud yang di kembangkan, serta berisi informasi layanan dan fitur yang diberikan kepada pengembang aplikasi yang akan menggunakan sistem *cloud*.

A wireframe of a registration form. At the top center is a square icon containing a stylized landscape with a sun and mountains. Below the icon are three text input fields labeled 'Username', 'Email', and 'Password'. At the bottom right is a blue rounded rectangle button labeled 'Register'.

**Gambar 3.2. Rancangan *form* pendaftaran**

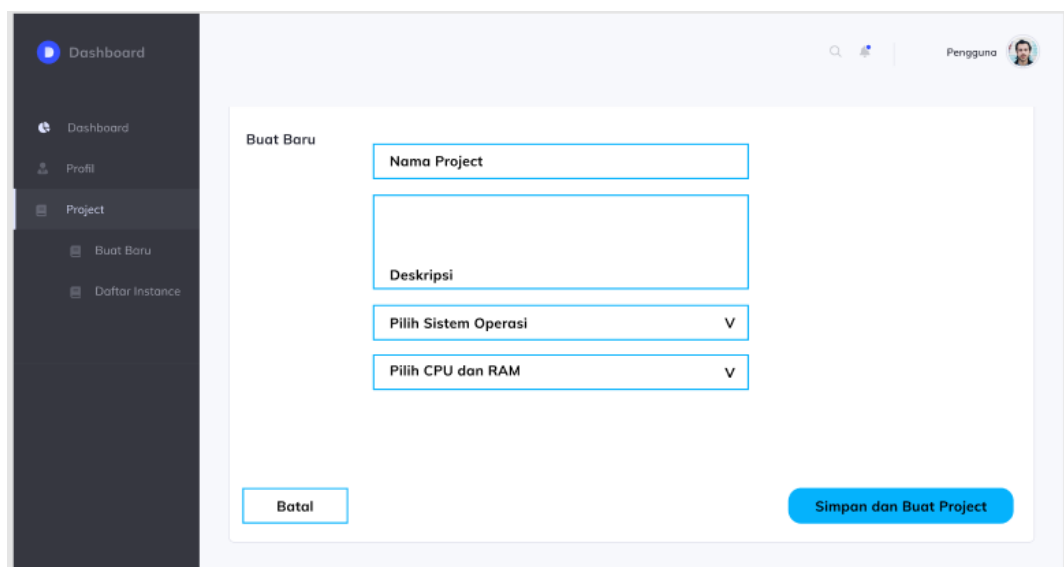


Pada gambar 3.2 merupakan rancangan sistem untuk dapat masuk ke dalam sistem cloud, pengembang aplikasi *web* harus melakukan pendaftaran, ini dimaksudkan untuk mempermudah pengelolaan sistem *virtual server cloud*.


 A login form design within a light gray rectangular frame. At the top center is a blue icon of a picture with a mountain and a sun. Below the icon are two white input fields with gray borders. The first field is labeled 'Username' and the second is labeled 'Password'. Below these fields is a blue rounded rectangular button with the word 'Login' in white text.

**Gambar 3.3. Rancangan form login**

Pada gambar 3.3 merupakan rancangan untuk login form untuk dapat masuk dalam sistem yang ingin dibuat.


 A screenshot of a web application interface for creating a new server instance. On the left is a dark sidebar with a menu containing 'Dashboard', 'Dashboard', 'Profil', 'Project', 'Buat Baru', and 'Daftar Instance'. The main content area has a light gray header with a search icon, a notification bell, and a user profile labeled 'Pengguna'. Below the header is a white box titled 'Buat Baru' containing four form fields: 'Nama Project' (text input), 'Deskripsi' (text area), 'Pilih Sistem Operasi' (dropdown menu with a downward arrow), and 'Pilih CPU dan RAM' (dropdown menu with a downward arrow). At the bottom of the form are two buttons: a white 'Batal' button and a blue 'Simpan dan Buat Project' button.

**Gambar 3.4. Rancangan membuat instance server baru**

Pada gambar 3.4 merupakan rancangan tampilan dalam membangun sebuah *virtual server cloud*, pengembang aplikasi *web* hanya perlu memilih kebutuhan

*server* yang diinginkan sesuai kebutuhan aplikasi yang dibuat seperti sistem operasi, jumlah unit processing, dan kapasitas penyimpanan.

### 3.4. 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*.

#### 3.4.1. 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 3.3. Tabel Pengujian Black Box**

<i>Identifikasi</i>	
<i>Nama Kasus Uji</i>	
<i>Deskripsi</i>	
<i>Kondisi Awal</i>	
<i>Tanggal Pengujian</i>	
<i>Penguji</i>	
<i>Skenario</i>	
1.	
2.	
(Dst...)	

<i>Hasil Yang Diharapkan</i>	<i>Hasil Yang Didapatkan</i>	<i>Kesimpulan</i>

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.

#### **3.4.2.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 mendefinisikan kombinasi waktu respons dan beban kerja.

## **BAB IV**

### **HASIL DAN PEMBAHASAN**

#### **4.1. Gambaran Umum Sistem**

Sistem yang dibuat bertujuan membantu para pengembang aplikasi berbasis *web*, dengan meneriakkan *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 penelitian 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*.

#### **4.2. Lingkungan Perancangan dan Implementasi Sistem**

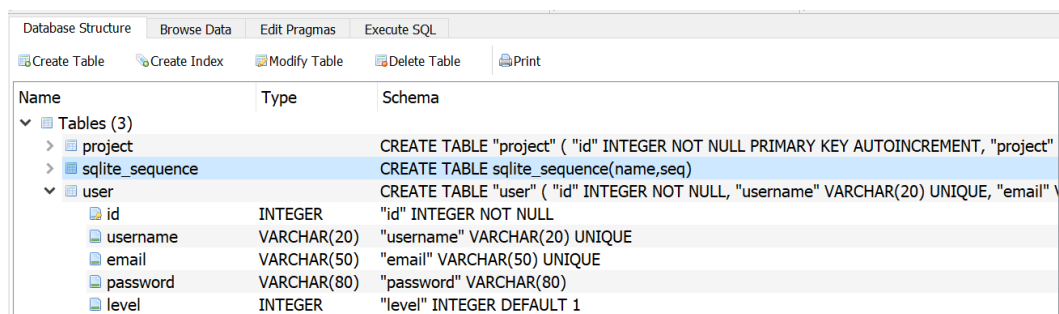
Sistem dirancang dan diimplementasikan pada sistem operasi *Ubuntu* dalam virtualisasi menggunakan *Vmware*, server dan sistem berjalan pada Laptop dengan spesifikasi i7, RAM 16 GB, dengan sistem operasi windows 10 64bit dan dengan support virtualisasi.

Sistem diimplementasikan menggunakan bahasa pemrograman *python* dengan *frameworks Flask*. Dalam perancangan dan implementasi sistem digunakan beberapa perangkat lunak untuk memenuhi semua kebutuhan sistem. Berikut adalah perangkat lunak yang digunakan :

- a. *VMware*
- b. *Visual Studio Codes*
- c. *Putty*
- d. *Jmeters*

### 4.3. Implementasi Basis Data

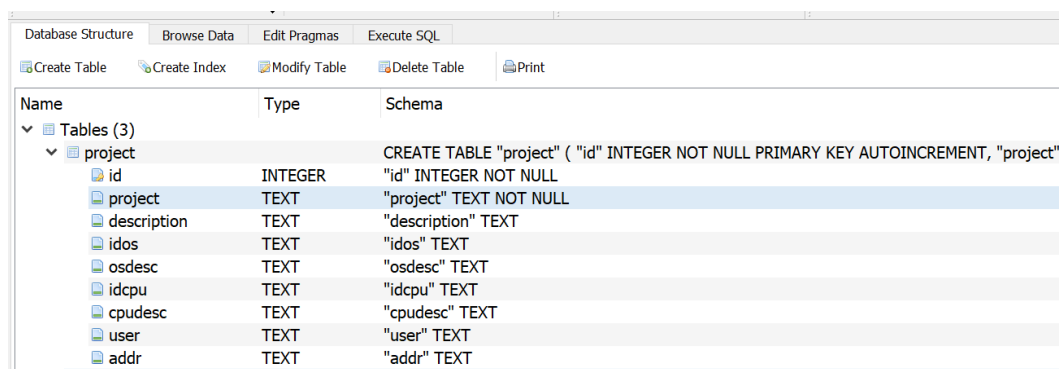
Pada sub bab ini akan dibahas mengenai tahap implementasi basis data pada sistem menginisiasi basis data dibutuhkan dua table, yaitu user atau developer aplikasi dan project.



Name	Type	Schema
Tables (3)		
project		CREATE TABLE "project" ( "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, "project"
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
user		CREATE TABLE "user" ( "id" INTEGER NOT NULL, "username" VARCHAR(20) UNIQUE, "email"
id	INTEGER	"id" INTEGER NOT NULL
username	VARCHAR(20)	"username" VARCHAR(20) UNIQUE
email	VARCHAR(50)	"email" VARCHAR(50) UNIQUE
password	VARCHAR(80)	"password" VARCHAR(80)
level	INTEGER	"level" INTEGER DEFAULT 1

**Gambar 4.1. Tabel basis data *user***

Pada gambar 4.1 tabel user berisi informasi id yang digenerate secara otomatis, username, email, password dalam bentuk hash dan level pengguna.



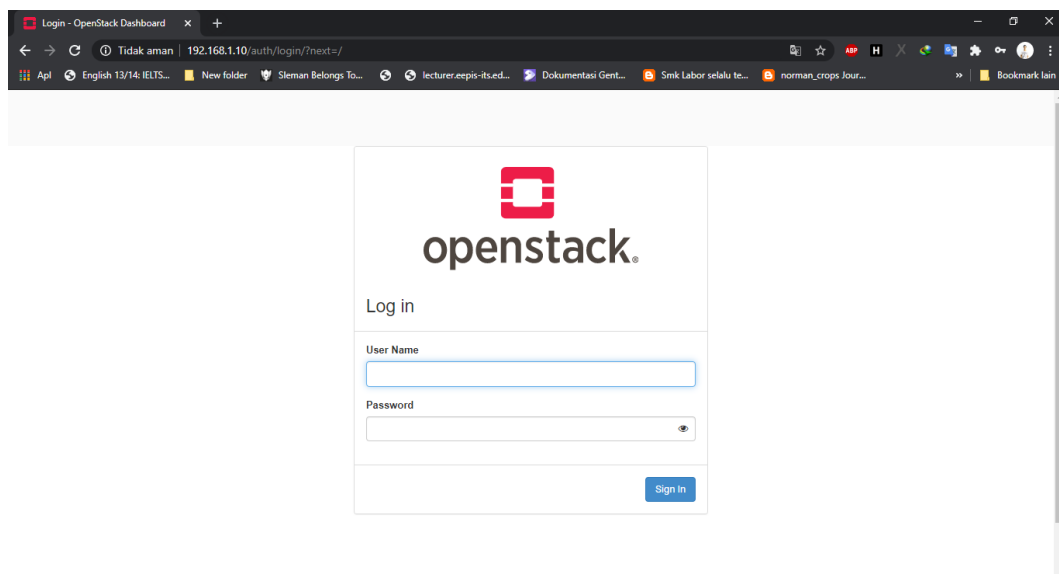
Name	Type	Schema
Tables (3)		
project		CREATE TABLE "project" ( "id" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, "project"
id	INTEGER	"id" INTEGER NOT NULL
project	TEXT	"project" TEXT NOT NULL
description	TEXT	"description" TEXT
idos	TEXT	"idos" TEXT
osdesc	TEXT	"osdesc" TEXT
idcpu	TEXT	"idcpu" TEXT
cpudesc	TEXT	"cpudesc" TEXT
user	TEXT	"user" TEXT
addr	TEXT	"addr" TEXT

**Gambar 4.2. Tabel basis data *project***

Selanjutnya pada gambar 4.2 tabel project berisi ID *project*, nama *project*, ID sistem operasi ID *CPU*, ID penyimpanan, User, dan IP Address server.

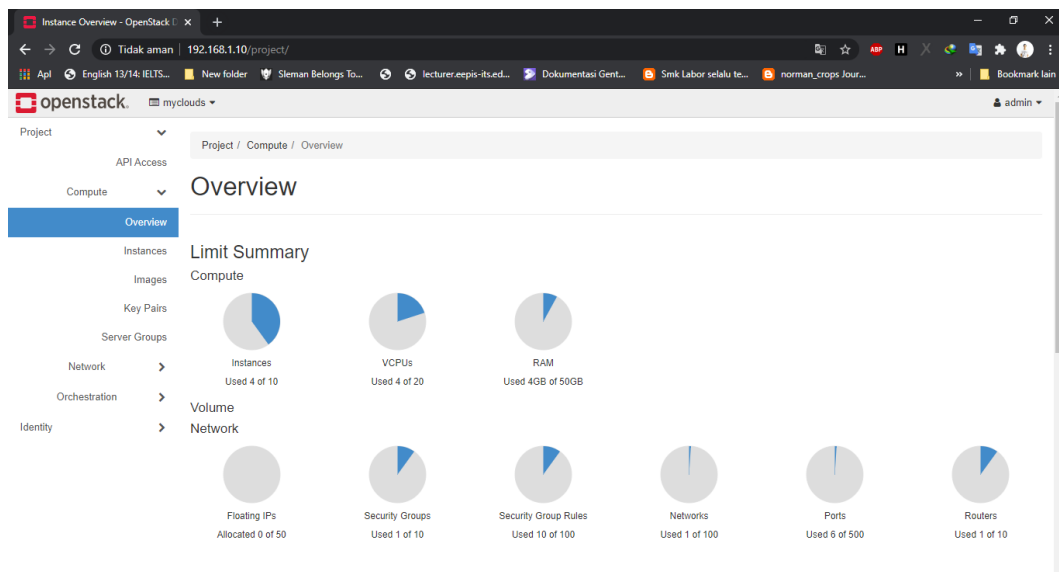
#### 4.4. Implementasi Antarmuka

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 :



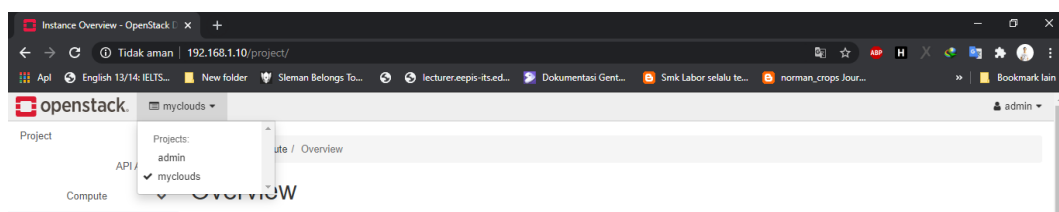
**Gambar 4.3. Tampilan Menu Awal Openstack**

Gambar 4.3 Merupakan 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 mengetahui kata sandi dari *openstack* dapat diakses melalui *command-line*



**Gambar 4.4. Tampilan Dashboard Openstack**

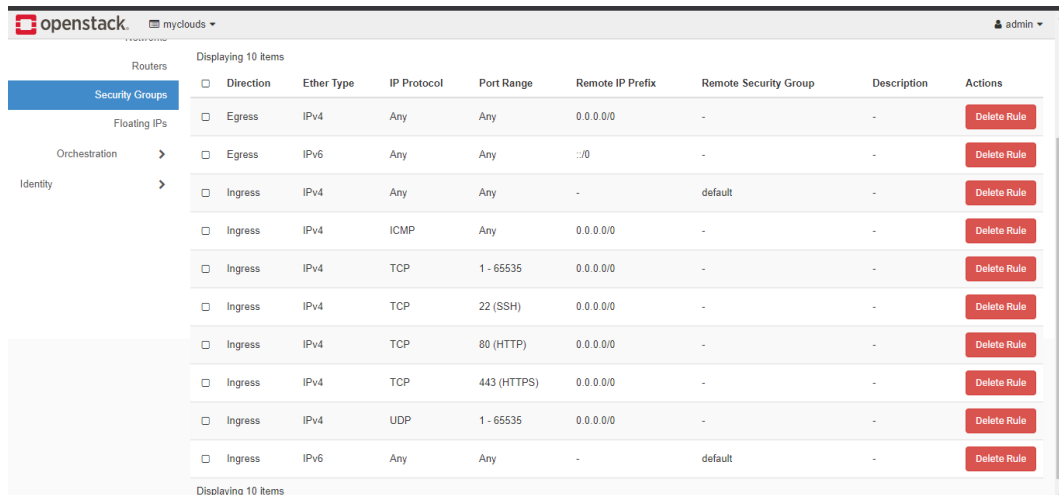
Gambar 4.4. Bila username dan kata sandi benar, maka akan diarahkan ke menu dashboard. Pada menu *dashboard* kita dapat mengetahui berapa maksimum *user* yang dapat dibuat oleh *server openstack*, berapa *core CPU* yang masih tersedia dan dapat digunakan dalam pembuatan instance / virtual machine.



**Gambar 4.5. Tampilan Menu Detail**

Gambar 4.5. 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

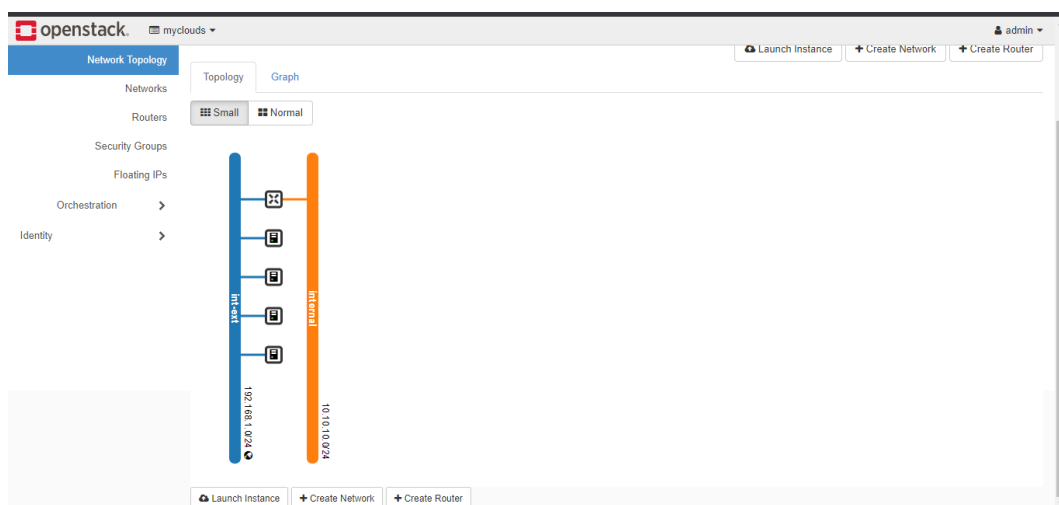
menghapus projectnya tanpa perlu menginstall ulang sistem ataupun *server openstack*.



Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group	Description	Actions
Egress	IPv4	Any	Any	0.0.0.0/0	-	-	Delete Rule
Egress	IPv6	Any	Any	:::0	-	-	Delete Rule
Ingress	IPv4	Any	Any	-	default	-	Delete Rule
Ingress	IPv4	ICMP	Any	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv4	TCP	1 - 65535	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv4	TCP	22 (SSH)	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv4	TCP	80 (HTTP)	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv4	TCP	443 (HTTPS)	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv4	UDP	1 - 65535	0.0.0.0/0	-	-	Delete Rule
Ingress	IPv6	Any	Any	-	default	-	Delete Rule

**Gambar 4.6. Tampilan Menu Deteksi**

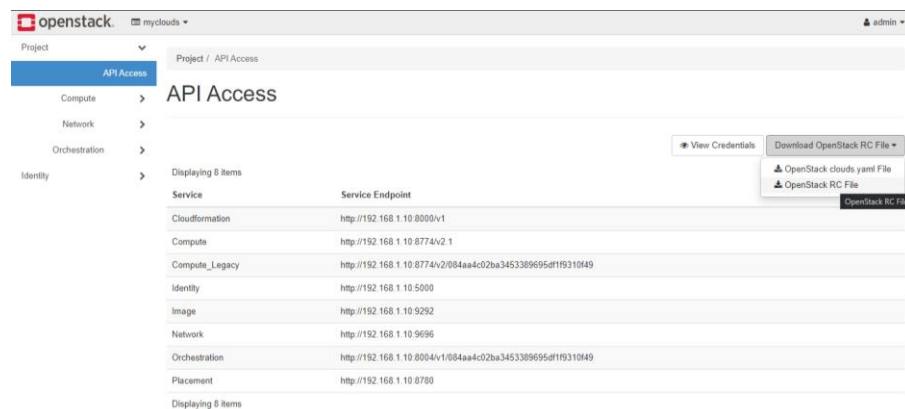
Untuk menu *Security Group*, berguna untuk memberi akses port yang bisa digunakan oleh instance. *Port* yang digunakan oleh instance tergantung kebutuhan yang perlukan, karena setiap *instance* memungkinkan menggunakan *port* yang berbeda – beda. Mungkin saja ada instance yang dibuat untuk penyimpanan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



**Gambar 4.7. Tampilan Pilih File**



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*.



Service	Service Endpoint
Cloudformation	http://192.168.1.10:8000/v1
Compute	http://192.168.1.10:8774/v2.1
Compute_Legacy	http://192.168.1.10:8774/v2/084aa4c02ba3453389695d1f9310449
Identity	http://192.168.1.10:5000
Image	http://192.168.1.10:9292
Network	http://192.168.1.10:9696
Orchestration	http://192.168.1.10:8004/v1/084aa4c02ba3453389695d1f9310449
Placement	http://192.168.1.10:8780

**Gambar 4.8. Tampilan Server**

Gambar 4.8. merupakan tampilan dari *server* untuk menerima *file APK* yang dikirim oleh *client*. Setelah *file* diterima, *server* akan menjalankan proses ekstraksi fitur seperti yang ditunjukkan pada gambar 4.9. *Server* dijalankan pada *ip 0.0.0.0* dengan *port 5000*. Pada gambar 4.8. tampak *server* sedang *standby listening client*. Ketika *server* menerima *APK* dari *client*, maka akan terlihat tampilan seperti pada Gambar 4.19.

**Gambar 4.9. Login Form pada website**

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 memasukkan username dan *password* yang sudah di buat pada form registrasi.

**Gambar 4.10. Buat Project Baru**

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 memasukkan nama *project*. Nama *project* nantinya akan menjadi nama instance baru sebagai *hostname*, untuk deskripsi

bersifat opsional, dapat ditambahkan keterangan ataupun tidak. Lalu pilih sistem operasi yang akan digunakan sebagai server pada lingkungan aplikasi yang akan dibuat. Lalu pilih *CPU*, *CPU* disini berisi berapa core yang dibutuhkan 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 4.11.

**Gambar 4.111. Proses Pemilihan Sistem Operasi**

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

**BUAT PROJECT BARU**

Nama Project:

Deskripsi:

Sistem Operasi:

CPU:

Copyright © 2016 Company. All rights reserved. Anything you want

**Gambar 4.122. Proses Pemilihan Flavor Server**

Pada segi antarmuka terlihat pada gambar 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

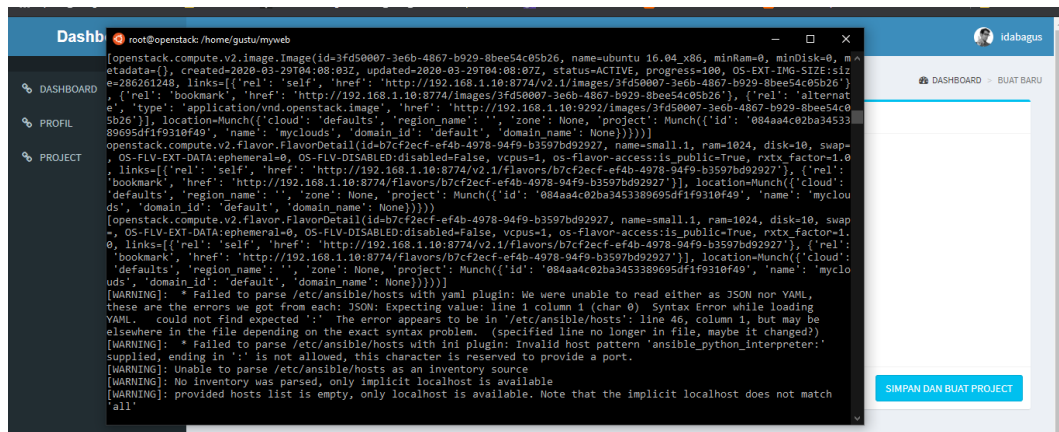
**Instances**

Displaying 2 Items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
cobaajasatu	ubuntu 16.04_x86	192.168.1.225	small.1	keyspair	Active	nova	None	Running	18 minutes	Create Snapshot
noqn	ubuntu 16.04_x86	192.168.1.201	small.1	keyspair	Active	nova	None	Running	2 days, 15 hours	Create Snapshot

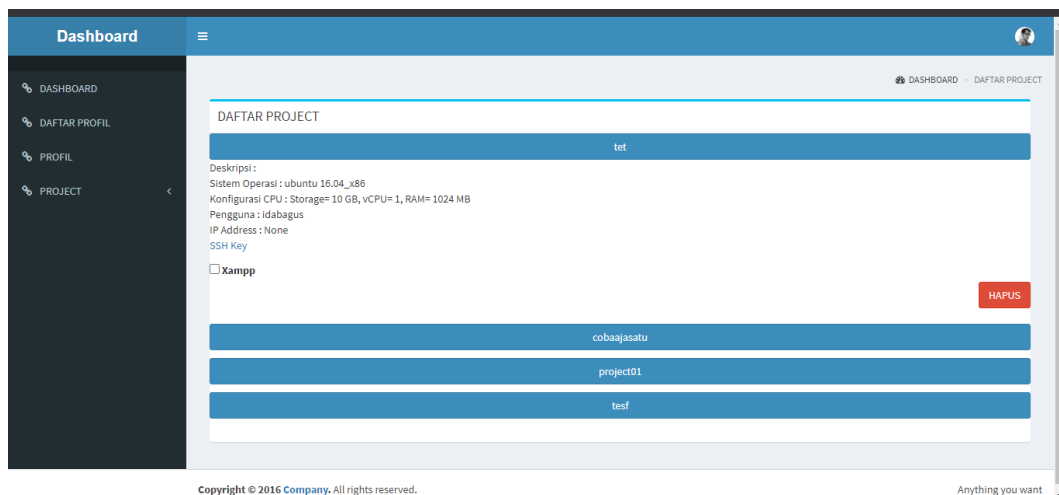
**Gambar 4.133. Tampilan pada Server Openstack**

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.



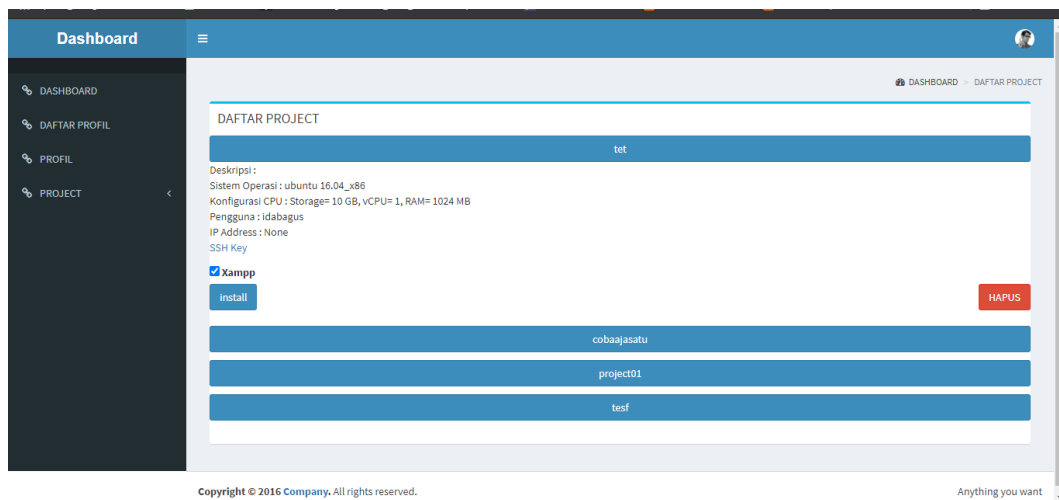
**Gambar 4.144. Tampilan Proses Ansible**

Setelah proses ansible dijalankan, proses debug akan muncul seperti gambar 4.14 keluaran data yang ditampilkan oleh hasil debug pada terminal berupa *json*.



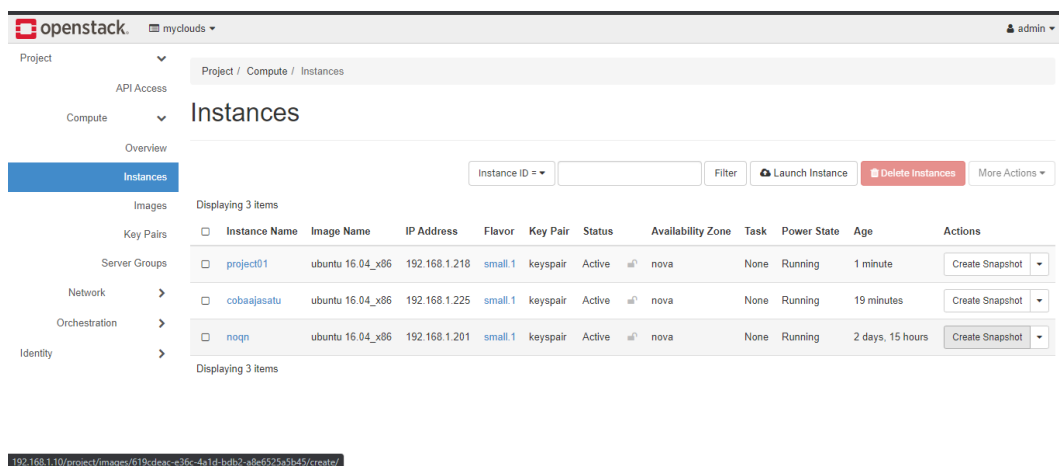
**Gambar 4.155. Tampilan Virtual Server**

Pada gambar 4.15, Ketika proses pembuatan *instance* selesai, sistem akan otomatis mengarahkan ke list instance yang sudah kita buat.



**Gambar 4.166. Tampilan Melakukan Instalasi XAMPP**

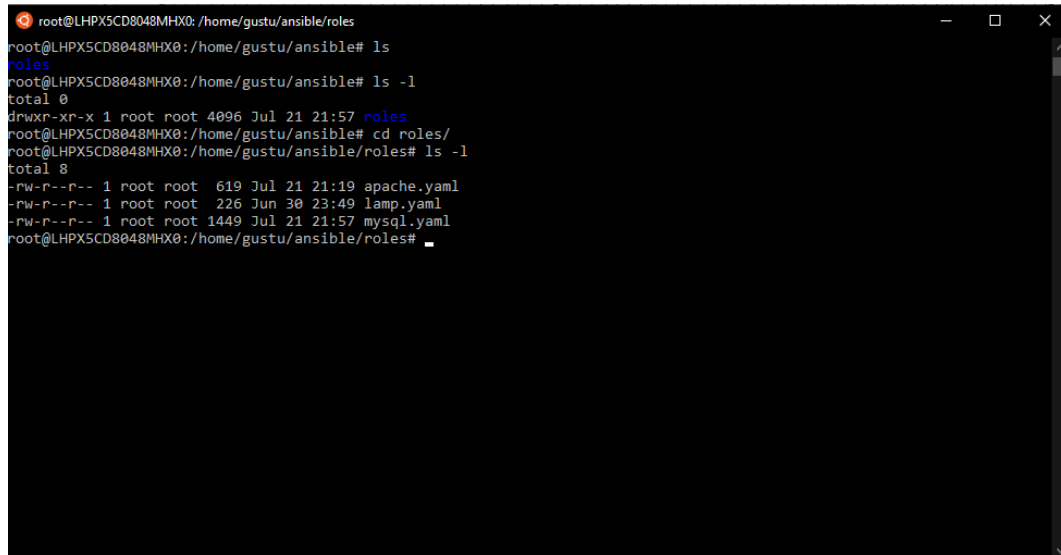
Pada proses ini sistem hanya akan membuat instance baru dan menginstall sistem operasi saja, bila pengguna ingin menginstall paket *XAMPP* / *LAMP*, pengguna hanya perlu memberi centang pada menu xampp dan selanjutnya pilih menu install. Maka sistem akan melakukan instalasi *XAMPP* pada *instance server*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



**Gambar 4.177. List Virtual Server pada Openstack**

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



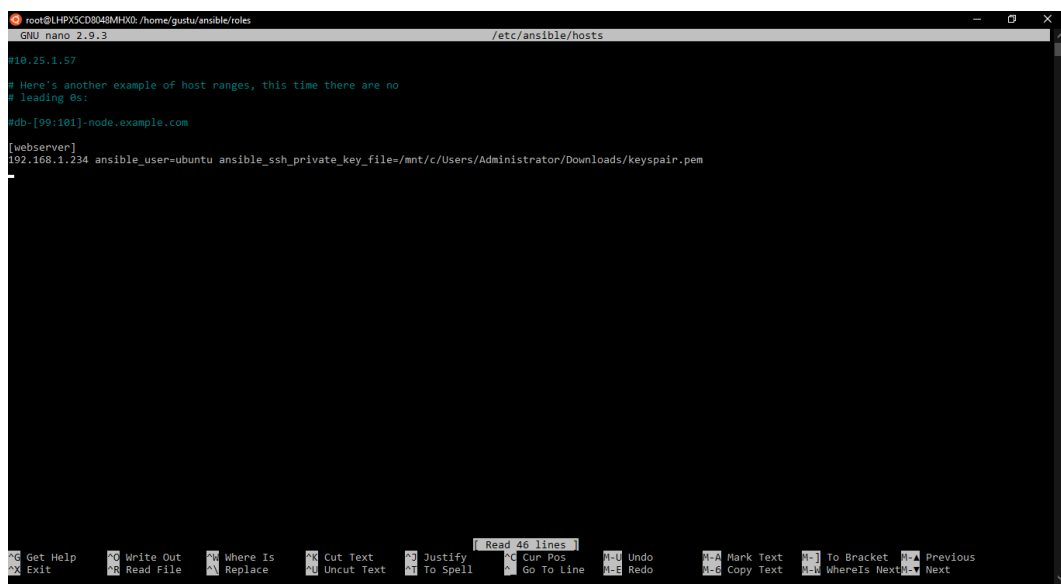
```

root@LHPX5CD8048MHX0: /home/gustu/ansible/roles
root@LHPX5CD8048MHX0: /home/gustu/ansible# ls
roles
root@LHPX5CD8048MHX0: /home/gustu/ansible# ls -l
total 0
drwxr-xr-x 1 root root 4096 Jul 21 21:57 roles
root@LHPX5CD8048MHX0: /home/gustu/ansible# cd roles/
root@LHPX5CD8048MHX0: /home/gustu/ansible/roles# ls -l
total 8
-rw-r--r-- 1 root root 619 Jul 21 21:19 apache.yaml
-rw-r--r-- 1 root root 226 Jun 30 23:49 lamp.yaml
-rw-r--r-- 1 root root 1449 Jul 21 21:57 mysql.yaml
root@LHPX5CD8048MHX0: /home/gustu/ansible/roles#

```

**Gambar 4.188. Tampilan file YAML**

Pada gambar 4.18 merupakan tampilan files yaml, pada direktori *ansible* ditambahkan sebuah folder bernama *roles*, jadi folder *roles* ini nantinya akan di eksekusi oleh *lamp.yaml* yang sudah berisikan perintah tugas yang akan di install yaitu *Apache2*, *PHP* dan *MySQL*.



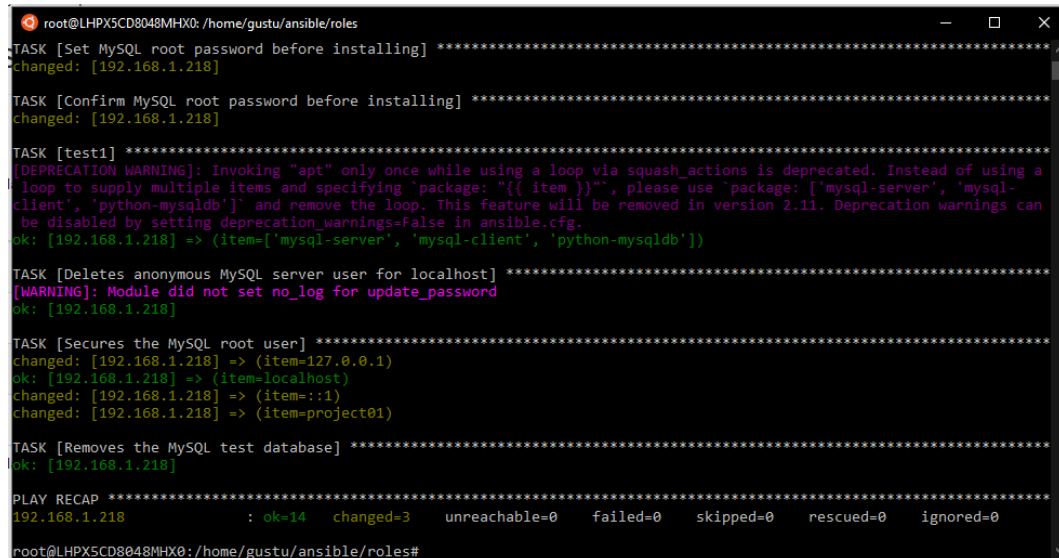
```

root@LHPX5CD8048MHX0: /home/gustu/ansible/roles
GNU nano 2.9.3 /etc/ansible/hosts
#10.25.1.57
# Here's another example of host ranges, this time there are no
# leading 0s:
#db-[99:101]-node.example.com
[webservers]
192.168.1.234 ansible_user=ubuntu ansible_ssh_private_key_file=/mnt/c/Users/Administrator/Downloads/keysair.pem

```

**Gambar 4.199. Set Host Virtual Server**

Pada gambar 4.19, sistem membuat alamat ip address dibuat pada file *hosts* digunakan untuk mengenali server mana yang akan dieksekusi ketika file *yml* dijalankan, serta menyertakan *username* dan *password* dari *instance* yang telah dibuat secara default melalui openstack.



```

root@LHPX5CD8048MHX0: /home/gustu/ansible/roles
TASK [Set MySQL root password before installing] *****
changed: [192.168.1.218]

TASK [Confirm MySQL root password before installing] *****
changed: [192.168.1.218]

TASK [test1] *****
[DEPRECATION WARNING]: Invoking "apt" only once while using a loop via squash_actions is deprecated. Instead of using a
loop to supply multiple items and specifying `package: "{{ item }}"`, please use `package: ['mysql-server', 'mysql-
client', 'python-mysqldb']` and remove the loop. This feature will be removed in version 2.11. Deprecation warnings can
be disabled by setting deprecation_warnings=False in ansible.cfg.
ok: [192.168.1.218] => (item=['mysql-server', 'mysql-client', 'python-mysqldb'])

TASK [Deletes anonymous MySQL server user for localhost] *****
[WARNING]: Module did not set no_log for update_password
ok: [192.168.1.218]

TASK [Secures the MySQL root user] *****
changed: [192.168.1.218] => (item=127.0.0.1)
ok: [192.168.1.218] => (item=localhost)
changed: [192.168.1.218] => (item=:1)
changed: [192.168.1.218] => (item=project01)

TASK [Removes the MySQL test database] *****
ok: [192.168.1.218]

PLAY RECAP *****
192.168.1.218      : ok=14  changed=3  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0

root@LHPX5CD8048MHX0: /home/gustu/ansible/roles#

```

**Gambar 4.200. Instalasi XAMPP Berhasil**

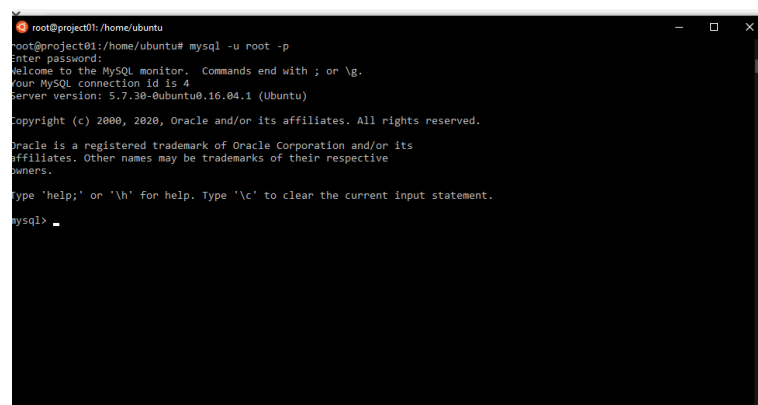
Pada gambar 4.20 merupakan hasil setelah menjalankan perintah untuk melakukan instalasi *lamp.yml*. Bila sukses melakukan instalasi pada *instance* akan muncul pada gambar di atas semua hasil debug tidak ada kegagalan ketika instalasi.





**Gambar 4.211. Pengujian web Server**

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. Selanjutnya bila pengguna ingin menambahkan *web* yang dibuatnya dalam kasus ini diambil menggunakan bahasa pemrograman *PHP*, pengguna hanya perlu menyalin *web* yang sudah dibuat ke */var/www/html*.



**Gambar 4.222. Pengujian Database**

Pada gambar 4.22 merupakan tampilan dari database yang sudah terinstall pada *instance*, pengguna dapat melakukan pengecekan kembali dengan *meremote instance server* yang sudah dibuat dengan menggunakan *SSH*. Serta melakukan

perintah *mysql -u root -p* untuk mengecek ataupun untuk membuat *project database* di dalam *instance server openstack*.

#### 4.5. Pengujian Sistem

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

##### 4.5.1. 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 sysadmin. Berikut merupakan tabel pengujian menggunakan black box :

**Tabel 4.1. Hasil Pengujian ke-1**

Kode Uji	Butir Uji	Teknik Pengujian	Hasil Pengujian
OS1	<i>Developer</i> baru melakukan pendaftaran untuk dapat masuk ke dalam sistem	<i>Black Box</i>	Diterima
OS2	<i>Developer</i> melakukan <i>login</i> ke dalam sistem	<i>Black Box</i>	Diterima
OS3	<i>Admin</i> dan <i>developer</i> memiliki antarmuka yang berbeda.	<i>Black Box</i>	Diterima
OS4	<i>Developer</i> dapat membuat instance (server virtual) baru untuk pertama kali,	<i>Black Box</i>	Diterima
OS5	<i>Developer</i> dapat membuat kembali instance (server virtual) baru.	<i>Black Box</i>	Diterima
OS6	<i>Developer</i> dapat memilih sendiri kebutuhan spesifikasi <i>prosessor</i> , <i>RAM</i> ,	<i>Black Box</i>	Diterima

	penyimpanan dan sistem Operasi yang akan di buat		
OS7	<i>Developer</i> dapat melihat list instance yang telah dibuat.	<i>Black Box</i>	Diterima
OS8	<i>Developer</i> dapat mendownload SSH Key, untuk mengakses instance melalui SSH.	<i>Black Box</i>	Diterima
OS9	<i>Developer</i> dapat mengakses instance melalui SSH dengan menggunakan SSHkey.	<i>Black Box</i>	Diterima
OS10	<i>Developer</i> dapat melihat IP publik dari instance melalui web	<i>Black Box</i>	Diterima
OS11	<i>Developer</i> dapat melakukan instalasi paket Apache, PHP dan MySQL (LAMPP) secara otomatis.	<i>Black Box</i>	Diterima
OS12	<i>Developer</i> dapat mengganti kata sandi untuk masuk ke dalam sistem	<i>Black Box</i>	Diterima
	Penggu tidak dapat masuk ke <i>dashboard</i> bila tidak melakukan <i>login</i> terlebih dahulu atau belum terdaftar dalam <i>web</i> admin	<i>Black Box</i>	Diterima
OS13	<i>Developer</i> dapat keluar atau <i>logout</i> dari sistem pada menu yang sudah disediakan.	<i>Black Box</i>	Diterima

#### 4.5.2. Performance Testing

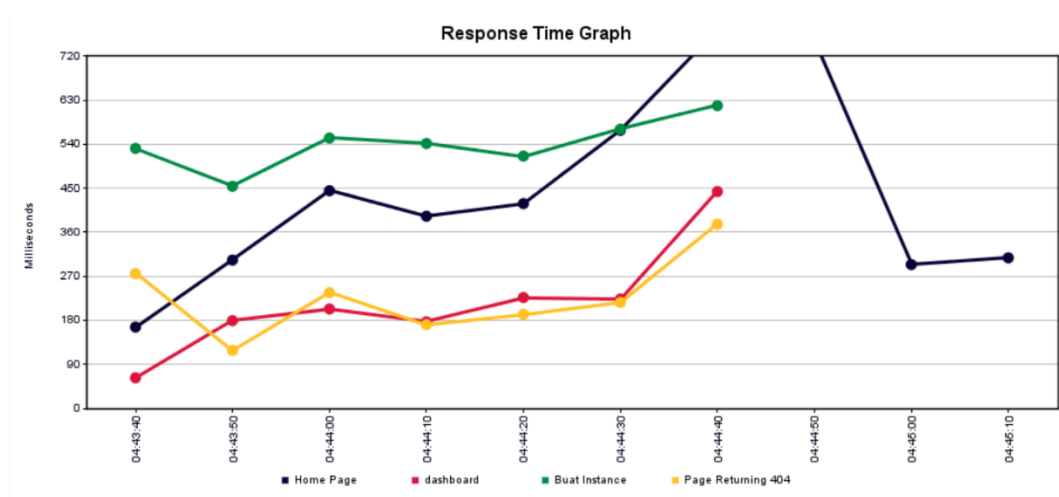
Pada pengujian *Performance Testing* merupakan lanjutan dari pengujian fungsionalitas sistem yang sudah dibuat. *Performance Testing* bertujuan memvalidasi “kecepatan” sistem yang dibuat. Kecepatan dalam hal ini peneliti melakukan pengujian dengan mengukur aspek waktu respon sistem. Peneliti melakukan pengujian kinerja dalam ruang lingkup yang mendekati ruang lingkup

sebenarnya. Pengujian kinerja sistem ini dicapai dengan melakukan simulasi menggunakan perangkat lunak aplikasi *Jmeter*.

Pengujian dilakukan sebanyak 3x dengan menggunakan simulasi dengan menggunakan *jmeter*, pengujian pertama dilakukan simulasi 100 pengembang aplikasi web ( *User of Threads* ) dalam 1 detik ( *Rump-up Period* ) dan di ulang selama 10 kali ( *Count* ) dengan mendapatkan data uji sebanyak 1000 total sampel selama 60 detik seperti pada gambar.

**Gambar 4.233. Persiapan proses simulasi**

Pada gambar merupakan tampilan aplikasi untuk melakukan pengujian performa pada *web server*, dengan skenario *web server* akan di akses oleh 100 pengguna per detiknya dengan total jumlah sample sebanyak 1000.



**Gambar 4.244. Grafik Hasil Pengujian ke-1**

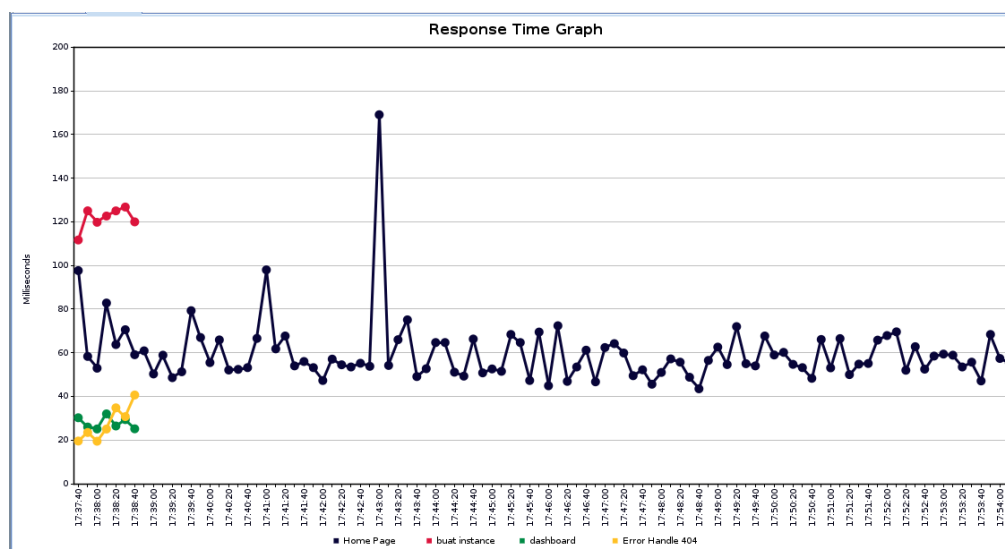
Pada gambar 4.24 merupakan hasil dari pengujian waktu respon web server, dapat dilihat pada gambar waktu yang dibutuhkan pada pembuatan instance cukup lebih lama dibandingkan dalam mengakses alamat *home page*, *dashboard* atau *page*

*error*, dikarenakan dalam alamat buat instance ini melakukan autentikasi pada server openstack. Sehingga membuat waktu respon yang dibutuhkan dalam menampilkan halaman membuat sebuah instance baru menjadi sedikit lebih lama melebihi 720 *milliseconds* dalam suatu waktu.

**Tabel 4.5. Hasil Pengujian ke-1**

Label	# Sample	Average	Min	Max	Std. Dev	Error %	Throughput	Received KB/sec	Avg. Bytes
Home Page	314	463	49	2174	292.54	0.00%	3.2/sec	159.19	51151.0
dashboard	272	226	27	2971	262.31	0.00%	4.5/sec	54.81	12329.0
Buat Instace	268	539	241	1511	171.88	0.00%	4.5/sec	68.40	15626.0
Page returning 404	261	208	20	2659	227.84	0.00%	4.5/sec	1.72	393.0
total	1115	264	20	2971	283.84	0.00%	11.3/sec	235.22	21284.7

Pada table 4.2 merupakan hasil pengujian performa web server yang jumlah akses pengguna sebanyak 100 pengguna per detiknya. Dari hasil tabel pengujian dengan 100 akses pengguna didapat hasil rata – rata 264, tidak ditemukan error pada setiap halaman yang dikunjungi dengan persentase *error* 0%, serta jumlah rata – rata penggunaan data dari total keseluruhan halaman *website* yaitu 21.284,7 Bytes.



**Gambar 4.255. Grafik Hasil Pengujian ke-2**

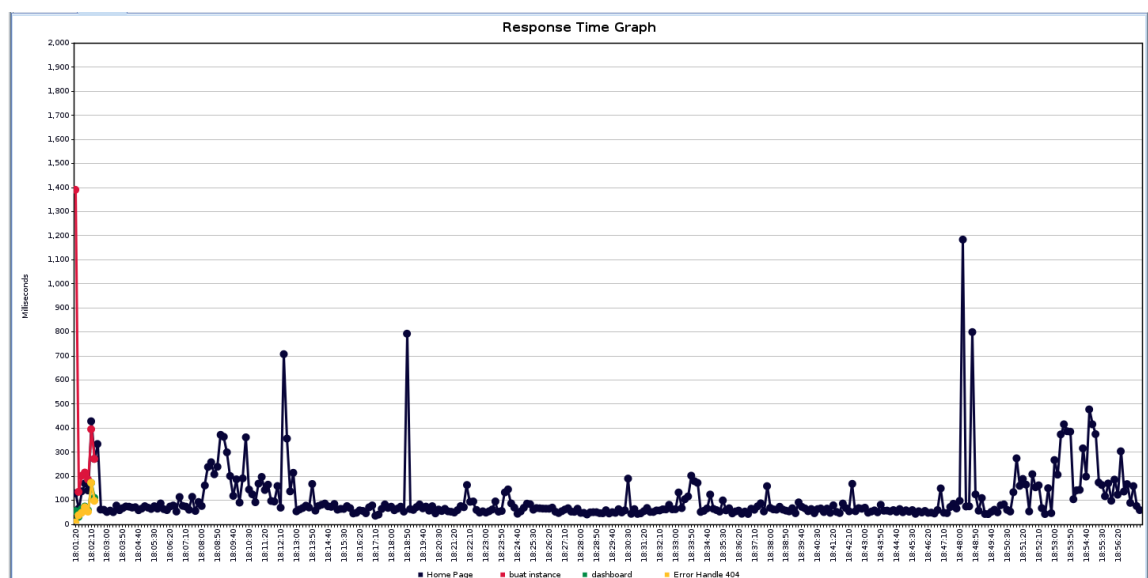
Pada gambar 4.25 merupakan hasil dari pengujian waktu respon web server dengan jumlah akses pengguna 1000 per detiknya dan diulang sebanyak 10 kali

akses, dapat dilihat dari grafik pengujian hasil performa dari halaman website diperoleh hasil kurang dari 100 milliseconds, hasil berbeda dari pengujian awal dengan akses pengguna sebanyak 1.000 yang mencapai 720 *milliseconds*.

**Tabel 4.6. Hasil Pengujian ke-2**

Label	# Sample	Average	Min	Max	Std. Dev	Error %	Throughput	Received KB/sec	Avg. Bytes
Home Page	1448	61	29	1314	52.23	0.00%	1.5/sec	74.20	51151.0
dashboard	544	123	84	244	26.60	0.00%	9.2/sec	139.66	15626.0
Buat Instace	544	27	9	220	21.59	0.00%	9.2/sec	111.01	12405.0
Page returning 404	534	28	6	1061	65.60	0.00%	9.2/sec	3.52	393.0
total	3106	60	6	1314	57.50	0.00%	3.1/sec	89.31	29416.2

Pada table 4.3 merupakan hasil pengujian performa web server yang jumlah akses pengguna sebanyak 1.000 pengguna per detik berulang sebanyak 10 kali. Dari hasil tabel pengujian dengan 1.000 akses pengguna didapat hasil rata – rata 60, tidak ditemukan error pada setiap halaman yang dikunjungi dengan persentase error 0%, serta jumlah rata – rata penggunaan data dari total keseluruhan halaman website yaitu 29.416,2 Bytes.



**Gambar 4.266. Grafik Hasil Pengujian ke-3**

Pada gambar 4.26 merupakan hasil dari pengujian waktu respon web server dengan jumlah akses pengguna 10.000 per detiknya dan diulang sebanyak 10 kali selama 60 detik akses, dari data grafik yang diperoleh jumlah lonjakan dalam mengakses halaman website 1.200 milliseconds cukup lama dalam membuka sebuah halaman website

**Tabel 4.7. Hasil Pengujian ke-3**

Label	# Sample	Average	Min	Max	Std. Dev	Error %	Throughput	Received KB/sec	Avg. Bytes
Home Page	10453	172	26	21740	624.40	0.00%	1.0/sec	52.22	51151.0
dashboard	511	241	87	3909	232.29	0.00%	8.6/sec	130.48	15626.0
Buat Instace	510	80	9	1422	85.88	0.00%	9.0/sec	108.77	12405.0
Page returning 404	497	76	6	1051	85.50	0.00%	8.9/sec	3.42	393.0
total	11971	167	6	21740	586.75	0.00%	1.2/sec	53.64	45876.5

Pada table 4.4 merupakan hasil pengujian performa web server yang jumlah akses pengguna sebanyak 10.000 pengguna per detik berulang sebanyak 10 kali. Dari hasil tabel pengujian dengan 10.000 akses pengguna didapat hasil rata – rata 167, tidak ditemukan error pada setiap halaman yang dikunjungi dengan persentase *error* 0%, serta jumlah rata – rata penggunaan data dari total keseluruhan halaman website yaitu 45.876,5 Bytes.

#### 4.6. Analisa

Dari hasil pengujian fungsional dengan Black – Box Testing untuk menguji validasi fungsionalitas dari sistem yang dibuat diperoleh hasil , sistem yang dibuat berjalan dengan baik dengan semua fungsi dan fitur dari sistem yang diberikan berjalan baik

Dari pengujian performa dengan menggunakan Jmeter, pengujian dilakukan sebanyak 3 kali dengan menguji beban kerja sistem yang di akses oleh banyak penggunanya, diperoleh hasil, dari tabel pengujian dengan 100 akses pengguna didapat hasil rata – rata 264, dengan jumlah rata – rata penggunaan data dari total keseluruhan halaman website yaitu 21.284,7 Bytes, pengujian dengan 1000 akses pengguna didapat hasil rata – rata 60, dengan jumlah rata – rata penggunaan data

dari total keseluruhan halaman website yaitu 29.416,2 Bytes. Pengujian dengan 10000 akses pengguna didapat hasil rata – rata 167, dengan jumlah rata – rata penggunaan data dari total keseluruhan halaman website yaitu 45.876,5 Bytes. Namun pada hasil throughput data pengujian pertama mendapatkan hasil 11,3 per detik, pengujian kedua memperoleh hasil 3.1 per detik sedangkan pengujian ketiga dengan jumlah 10.000 akses pengguna per detiknya memperoleh 1,2 per detik, lebih cepat dari dua pengujian sebelumnya. Akan tetapi dari hasil grafik pengujian ketiga lebih lama dalam membuka halaman website hingga mencapai 1.200 *milliseconds*.



## **BABI V**

### **SIMPULAN DAN SARAN**

#### **5.1. Simpulan**

Berdasarkan penelitian yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut :

1. Sistem yang dibangun dapat diimplementasikan dengan Sistem *Cloud computing* secara otomatisasi dengan menggunakan *ansible* dengan prinsip kerja sistem membuat *virtual server* secara otomatis dan melakukan instalasi paket *LAMP* secara otomatis.
2. Dari pengujian performa dengan menggunakan Jmeter yang dilakukan sebanyak 3 kali dengan menguji beban kinerja sistem yang di akses oleh banyak pengguna, diperoleh hasil, dari tabel pengujian dengan 100 akses pengguna didapat hasil rata – rata 264, dengan jumlah rata – rata penggunaan data dari total keseluruhan halaman website yaitu 21.284,7 Bytes, pengujian dengan 1.000 akses pengguna didapat hasil rata – rata 60, dengan jumlah rata – rata penggunaan data dari total keseluruhan halaman website yaitu 29.416,2 Bytes. Pengujian dengan 10.000 akses pengguna didapat hasil rata – rata 167, dengan jumlah rata – rata penggunaan data dari total keseluruhan halaman website yaitu 45.876,5 Bytes.

#### **5.2. Saran**

Untuk pengembangan sistem lebih lanjut, penulis ingin memberikan beberapa saran sebagai berikut :

1. Dapat dikembangkan lebih lanjut dapat menambahkan fitur keamanan data pengguna, serta dapat dikembangkan secara dinamis untuk kebutuhan penyimpanan, jaringan, maupun untuk komputasinya.
2. Dapat menambahkan monitoring instance untuk dapat melihat kesehatan setiap instance / virtual server untuk pengguna. Jadi pengguna pun dapat memantau keadaan virtual servernya sendiri. Hal tersebut dapat menjadi pertimbangan untuk penggunaanya.

## DAFTAR PUSTAKA

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

## LAMPIRAN

### 1. Source Code Aplikasi

#### a. 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')

```

### a. Fungsi Sign-Up

```
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)
```

### b. Fungsi Login

```
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)
```

### c. Fungsi enkripsi password basis data

```

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

```

#### **d. Fungsi menampilkan kebutuhan *storage* dan *memory* server**

```

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

```

#### **e. Fungsi untuk menampilkan *list virtual server***

```

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
```

#### **f. Fungsi membuat *project* baru**

```
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)
```

#### **g. Fungsi menyimpan *project* baru**

```
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)

```

### h. Fungsi melakukan eksekusi *file .YAML*

```
fin = open("yaml/deploy_"+current_user.username+".yaml", "wt")
    fin.write(data).
```

```
p = subprocess.Popen("ansible-playbook " +
"yaml/deploy_"+current_user.username+".yaml", shell=True,
stdout=subprocess.PIPE)
    stdout, stderr = p.communicate()
```

### i. Fungsi menghapus *virtual mesin 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

```

#### **j. *Authentikasi API Openstack***

```

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)

```

```

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

```

### **k. Baris kode proses membuat *virtual server* baru dengan *ansible***

```

- name: Deploy on OpenStack
  hosts: localhost
  gather_facts: false
  tasks:
    - name: Deploy an instance
      os_server:
        state: present
        name: <name>
        image: <image>
        key_name: keypair
        wait: yes

```

```

    flavor: <flav>
    auto_floating_ip: yes
    network: int-ext
    meta:
        hostname: webserver.localdomain

```

### l. Baris kode proses menghapus *virtual* mesin dengan *ansible*

```

- name: remove an instance
  hosts: localhost
  tasks:
    - name: remove an instance
      os_server:
        name: <name>
        state: absent

```

### m. Baris perintah menginstalasi *apache2* dengan *ansible*

```

- 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

```

### n. Baris perintah menginstalasi *MySQL* dengan *ansible*

```
- 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

```

#### **o. Baris perintah menginstalasi *LAMP* dengan *ansible***

```

- 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

```

#### **p. Baris perintah untuk mengakses *virtual* mesin melalui *server***

```

[webserver]
192.168.1.234                                ansible_user=ubuntu
ansible_ssh_private_key_file=/mnt/c/Users/Administrator/Downloads/
keyspair.pem

```