Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

RANCANG BANGUN MANAJEMEN ALOKASI VIRTUAL MACHINE DALAM LINGKUNGAN HYPERVISOR YANG HETEROGEN

FATHONI ADI KURNIAWAN
NRP 5114 100 020

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Dosen Pembimbing II
Bagus Jati Santoso, S.Kom., Ph.D

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR - KI141502

RANCANG BANGUN MANAJEMEN ALOKASI VIRTUAL MACHINE DALAM LINGKUNGAN HYPERVISOR YANG HETEROGEN

FATHONI ADI KURNIAWAN
NRP 5114 100 020

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Dosen Pembimbing II
Bagus Jati Santoso, S.Kom., Ph.D

DEPARTEMENT INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS - KI141502

DESIGN OF VIRTUAL MACHINE ALLOCATION MANAGEMENT IN HETEROGENEOUS HYPERVISOR ENVIRONMENT

FATHONI ADI KURNIAWAN
NRP 5114 100 020

Supervisor I
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

Supervisor II
Bagus Jati Santoso, S.Kom., Ph.D

DEPARTEMENT OF INFORMATICS
Faculty of Information Technology and Communication
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN MANAJEMEN ALOKASI VIRTUAL MACHINE DALAM LINGKUNGAN HYPERVISOR YANG HETEROGEN

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur Jaringan dan Komputer
Program Studi S1 Departemen Informatika Fakultas Teknologi Informasi dan Komunikasi Institut Teknologi Sepuluh Nopember

Oleh :

FATHONI ADI KURNIAWAN
NRP: 5114 100 020

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D. .....
NIP: 197708242006041001 (Pembimbing 1)

Bagus Jati Santoso, S.Kom., Ph.D. .....
NIP: 198611252018031001 (Pembimbing 2)

SURABAYA
Juni 2018

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN MANAJEMEN ALOKASI VIRTUAL MACHINE DALAM LINGKUNGAN HYPERVISOR YANG HETEROGEN

Nama	:	**FATHONI ADI KURNIAWAN**
NRP	:	**5114 100 020**
Departemen	:	**Informatika FTIK**
Pembimbing I	:	**Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D**
Pembimbing II	:	**Bagus Jati Santoso, S.Kom., Ph.D**

Abstrak

Virtual Machine merupakan teknik virtualisasi yang menyajikan perangkat keras yang dapat menjalankan perangkat lunak seperti perangkat keras fisik. Penyedia layanan Virtual Machine biasa disebut dengan Hypervisor. Hypervisor menangani manajemen Virtual Machine pada sebuah host.

Saat ini, penggunaan Virtual Machine dalam dunia teknologi sangat banyak dilakukan. Karena sangat banyaknya penggunaan Virtual Machine, maka dari itu banyak Hypervisor yang disediakan oleh pengembang, contohnya Vmware, Proxmox, Xen, Qemu dan lain-lain. Keragaman Hypervisor menyebabkan perbedaan cara pengoperasian. Hal ini menyebabkan sulitnya alokasi Virtual Machine. Masalah tersebut juga dialami oleh DPTSI ITS. Ketika pengguna membutuhkan Virtual Machine untuk kerperluan pengembangan aplikasi maupun server database, pengguna akan kebingungan untuk melakukan alokasi Virtual Machine. Pada akhirnya pengguna yang membutuhkan Virtual Machine akan menghubungi System Administrator yang mengerti tentang pengorperasian Hypervisor tertentu.

Dalam tugas akhir ini akan dibuat sebuah rancangan sistem

yang memungkinkan menjembatani cara penggunaan Hypervisor yang berbeda. Sistem akan diakses melalui interface yang disediakan untuk pengguna untuk manjemen alokasi Virtual Machine. Dari hasil uji coba, sistem dapat menangani permintaan alokasi virtual machine pada hypervisor yang berbeda berdasarkan pemilihan server terbaik. Selain dapat dapat menangani permintaan alokasi virtual machine pada hypervisor yang berbeda, sistem mampu mengalokasikan virtual machine, lima belas request dalam waktu hampir bersamaan.

***Kata-Kunci**: middleware, hypervisor, vmware, proxmox, ahp, virtual machine*

DESIGN OF VIRTUAL MACHINE ALLOCATION MANAGEMENT IN HETEROGENEOUS HYPERVISOR ENVIRONMENT

Name	:	**FATHONI ADI KURNIAWAN**
NRP	:	**5114 100 020**
Department	:	**Informatics FTIK**
Supervisor I	:	**Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D**
Supervisor II	:	**Bagus Jati Santoso, S.Kom., Ph.D**

Abstract

Virtual Machine is a virtualization technique that presents hardware that can run like physical hardware. Virtual Machine Service Provider is usually called Hypervisor. Hypervisor handles Virtual Machine management on a host. Currently, in technology, Virtual Machine uses in the world. Due to the huge number of Virtual Machine's use, therefore many Hypervisors are provided by developers, for example Vmware, Proxmox, Xen, Qemu and others. Hypervisor diversity causes different ways to operate. This causes the difficulty of allocating Virtual Machine. The problem is also experienced in DPTSI ITS. When users need Virtual Machine for application development and database servers, users will be confused to do Virtual Machine allocations. In the end, user who needs a Virtual Machine will contact the System Administrator who understands certain Hypervisor interpretations. In this final project will be created a system design that allows to bridge manajement task at the different Hypervisor. The system will be accessed through the interface provided for the user to allocate Virtual Machine. From the test results, the system can handle virtual machine allocation requests on different hypervisors based on the best server

selection. In addition to being able to handle virtual machine allocation requests on different hypervisors, the system is able to allocate virtual machines, fifteen requests in the same time.

***Kata-Kunci**: middleware, hypervisor, vmware, proxmox, ahp, virtual machine*

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **Rancang Bangun Manajemen Alokasi *Virtual Machine* dalam Lingkungan *Hypervisor* yang Heterogen**. Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Departemen Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari. Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT atas anugerahnya yang tidak terkira kepada penulis dan Nabi Muhammad SAW.
2. Keluarga penulis yang selalu menyemangati.
3. Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D selaku pembimbing I yang telah membantu, membimbing dan memotivasi penulis mulai dari pengerjaan proposal hingga terselesaikannya Tugas Akhir ini.
4. Bapak Bagus Jati Santoso, S.Kom., Ph.D selaku pembimbing II yang juga telah membantu, membimbing dan memotivasi penulis mulai dari pengerjaan proposal hingga terselesaikannya Tugas Akhir ini.
5. Teman-teman *Administrator* laboratorium AJK.
6. Darlis Herumurti, S.Kom., M.Kom., selaku Kepala Departemen Informatika ITS pada masa pengerjaan Tugas Akhir, Bapak Radityo Anggoro, S.Kom., M.Sc., selaku

koordinator TA dan segenap dosen Departemen Informatika yang telah memberikan ilmu dan pengalamannya.

7. Serta semua pihak yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2018

Fathoni Adi K

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

BAB I

PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir dan sistematika penulisan.

1.1 Latar Belakang

Virtual Machine merupakan teknik virtualisasi yang menyajikan perangkat keras yang dapat menjalankan perangkat lunak seperti perangkat keras fisik. Penyedia layanan *Virtual Machine* biasa disebut dengan *Hypervisor*. *Hypervisor* menangani manajemen *Virtual Machine* pada sebuah *host*.

Saat ini, penggunaan *Virtual Machine* dalam dunia teknologi sangat banyak dilakukan. Karena sangat banyaknya penggunaan *Virtual Machine*, maka dari itu banyak *Hypervisor* yang disediakan oleh pengembang, contohnya Vmware, Proxmox, Xen, Qemu dan lain-lain. Keragaman *Hypervisor* menyebabkan perbedaan cara pengoperasian. Hal ini menyebabkan sulitnya alokasi *Virtual Machine*. Masalah tersebut juga dialami oleh DPTSI ITS. Ketika pengguna membutuhkan *Virtual Machine* untuk kerperluan pengembangan aplikasi maupun server database, pengguna akan kebingungan untuk melakukan alokasi *Virtual Machine*. Pada akhirnya pengguna yang membutuhkan *Virtual Machine* akan menghubungi *System Administrator* yang mengerti tentang pengorperasian *Hypervisor* tertentu.

Oleh karena itu dibutuhkan cara untuk mengelola alokasi *Virtual Machine* pada *Hypervisor* yang berbeda. Salah satunya dengan membuat sebuah *Middleware* yang bertugas untuk menjembatani cara penggunaan *Hypervisor* yang berbeda. *Middleware* akan diakses melalui *interface* yang disediakan untuk pengguna untuk manjemen alokasi *Virtual Machine*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut :

1. Bagaimana cara membuat *Middleware* berbasis *REST API* yang digunakan untuk menjembatani penggunaan *Hypervisor* yang berbeda?
2. Bagaimana cara mengimplementasikan *Middleware* berbasis *REST API* melalui *interface* web dan *command line*?
3. Bagaimana cara menentukan *Host* yang tersedia untuk alokasi *Virtual Machine* menggunakan algoritma *Decision Making (Analytical Hierarchy Process)*?
4. Bagaimana cara melakukan *provisioning* pada saat alokasi *virtual machine* baru?

1.3 Batasan Masalah

Dari permasalahan yang telah diuraikan di atas, terdapat beberapa batasan masalah pada tugas akhir ini, yaitu:

1. *Hypervisor* yang didukung adalah Proxmox dan Vmware Vsphere.
2. OS yang disediakan untuk *Virtual Machine* adalah *template* sistem operasi Ubuntu dan Debian.
3. Uji coba aplikasi akan menggunakan *REST API*.

1.4 Tujuan

Tujuan pembuatan tugas akhir ini antara lain:

1. Membuat sebuah *Middleware* yang digunakan untuk menjembatani penggunaan *Hypervisor* yang berbeda.
2. Mengimplementasikan metode pengambilan keputusan untuk pendistribusian *virtual machine* yang efisien

berdasarkan penggunaan *Memory*, *CPU* dan *Storage* pada server

1.5 Manfaat

Manfaat dari pembuatan tugas akhir ini adalah mempermudah pengelolaan *Virtual Machine* dalam lingkungan *Hypervisor* yang beragam (heterogen) sehingga pengguna tidak perlu tahu bagaimana cara menggunakan masing-masing *Hypervisor*.

(Halaman ini sengaja dikosongkan)

BAB II

TINJAUAN PUSTAKA

2.1 Virtualisasi

Virtualisasi merupakan komponen terpenting dalam komputasi awan dengan memisahkan perangkat keras dengan sistem operasi yang berjalan. Virtualisasi memiliki kemampuan untuk membagi sumber daya fisik yang ada untuk dijadikan sumber daya *virtual* dan dapat menjadikan berbagai sumber daya fisik yang ada menjadi satu sumber daya *virtual*. Virtualisasi membawa banyak perubahan pada perusahaan IT saat ini.

Virtualisasi dan *multitasking* pada sistem operasi memiliki kemampuan untuk mengizinkan pemusatan berbagai server *virtual* pada sebuah komputer fisik. Ketika sebuah kelompok ingin mengerjakan sebuah pekerjaan tertentu pada dua atau lebih server dan salah satu server gagal karena sumber daya habis terpakai, virtualisasi dapat memindahkan pekerjaan dan server tersebut pada komputer fisik yang lain. Hal tersebut merupakan salah satu keuntungan menggunakan virtualisasi. Selain itu virtualisasi dapat menghemat energi yang dipakai dan biaya pembelian komputer fisik. Sehingga virtualisasi dapat meningkatkan keuntungan perusahaan[1].

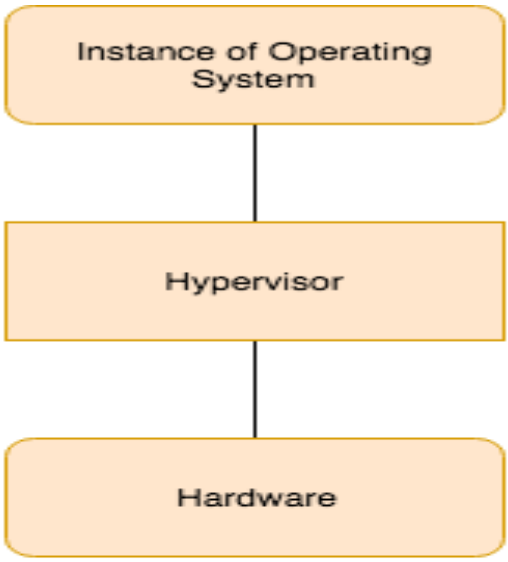
2.2 *Hypervisor*

Pada virtualisasi, terdapat sebuah *layer* yang berada diantara perangkat keras dan *virtual machine*. *Layer* tersebut disebut *Hypervisor*. *Hypervisor* menyediakan standarisasi *CPU*, *memory* dan *storage* untuk *virtual machine*. *Hypervisor* memiliki dua jenis yaitu, *Bare-metal Hypervisor* dan *Hosted Hypervisor*[1].

2.2.1 *Bare-metal Hypervisor*

Jenis *hypervisor* ini dilakukan instalasi pada perangkat keras x86 secara langsung. Setelah melakukan instalasi *hypervisor*,

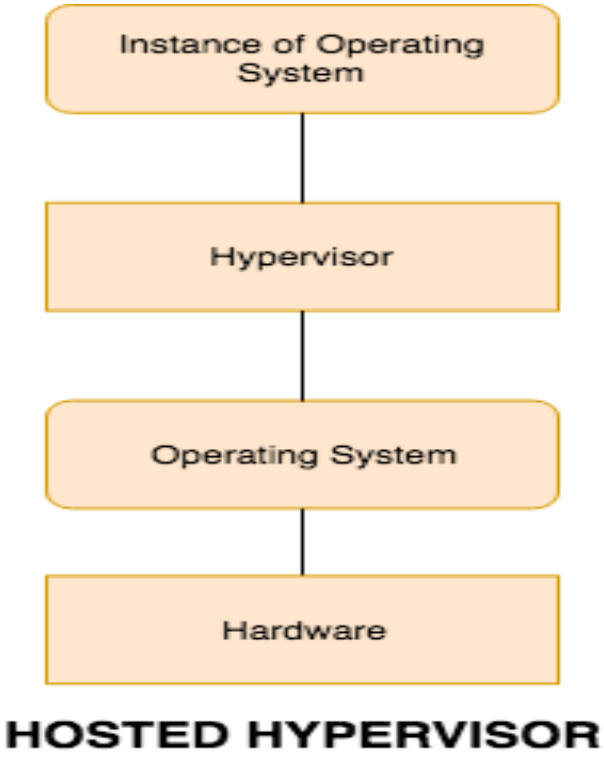
pengguna dapat melakukan instalasi sistem operasi pada (*virtual machine*) atau yang sering disebut *instance*. Jenis *hypervisor* ini lebih efisien dibanding *Hosted hypervisor*.



Gambar 2.1: Arsitektur *Bare-metal Hypervisor*

2.2.2 *Hosted Hypervisor*

Jenis *hypervisor* ini dilakukan instalasi pada sistem operasi yang sudah terinstalasi pada perangkat keras sebelumnya. *Instance* dibuat setelah melakukan instalasi *hypervisor*.



Gambar 2.2: Arsitektur *Hosted Hypervisor*

2.3 Python

Python adalah bahasa pemrograman interpretatif, interaktfi dan berorientasi objek. Python menggabungkan modul, pengecualian, penulisan secara dinamis, tipe data dinamis yang sangat tinggi dan kelas. Python memiliki antarmuka ke banyak *system call* dan pusataka diberbagai sistem dan dapat diperluas ke bahasa pemrograman C atau C++. Python dapat berjalan pada berbagai sistem operasi seperti Unix, Linux, Mac Os dan Windows.

Python adalah bahasa pemrograman tingkat tinggi yang dapat diterapkan pada berbagai masalah. Bahasa ini dilengkapi pustaka yang besar untuk melakukan pemrosesan *string*, protokol internet, rekayasa perangkat lunak dan antarmuka sistem operasi[2].

Dilihat dari kelebihan, bahasa pemrograman Python dapat

digunakan dalam pengembangan aplikasi yang kompleks. Pada tugas akhir ini, bahasa pemrograman Python digunakan untuk pembuatan *middleware*.

2.4 Flask

Flask adalah kerangka aplikasi web Python yang ringan. Flask dirancang untuk memulai membuat web dengan cepat dan mudah, dengan kemampuan untuk membuat aplikasi web sampai tingkat yang rumit. Flask dibuat dengan terintegrasi dengan modul Werkzeug dan Jinja. Flask termasuk salah satu kerangka aplikasi web Python yang populer.

Flask didesain tidak memiliki depedensi dan tata letak kerangka aplikasi, dengan demikian pengembang memiliki kebebasan untuk mengatur kerangka aplikasinya sendiri serta menambahkan modul yang diperlukan sesuai kebutuhan. Flask memiliki berbagai ekstensi yang dikembangkan oleh komunitas sehingga dapat menambahkan berbagai fungsi dengan mudah[3].

Flask memiliki kelebihan yaitu sangat ringan dan sangat sederhana dalam proses pengembangan. Sehingga Flask sangat cocok digunakan untuk pembuatan *HTTP Rest API*. Pada tugas akhir ini, Flask akan digunakan untuk pembuatan *HTTP Rest API*.

2.5 Python Celery

Celery adalah modul Python yang berguna untuk antrian pekerjaan bersifat asinkron yang berdasarkan pengiriman pesan secara distribusi. Celery berfokus pada operasi *real-time* namun juga mendukung penjadwalan. Unit yang dieksekusi disebut dengan *task* yang dijalankan bersamaan pada satu server atau lebih yang menggunakan mekanisme *multiprocessing*.

Celery dapat diintegrasikan pada kerangka kerja web dengan

mudah. Selain itu meskipun Celery ditulis dengan Python, tetapi protokolnya dapat diimplementasikan dalam bahasa apa pun. Ini juga dapat beroperasi dengan bahasa lain menggunakan *webhooks*[4].

Karena memiliki kemudahan dalam integrasi dengan kerangka kerja web, Python Celery digunakan sebagai manajemen *queue* pada tugas akhir ini.

2.6 Pyvmomi

Pyvmomi adalah Python SDK untuk Vmware Sphere yang digunakan untuk mengatur Vmware ESX, ESXI dan Vcenter. Pyvmomi dikembangkan langsung oleh Vmware untuk mempermudah pengaturan secara otomatis[5].

Karena dikembangkan langsung oleh Vmware, Pyvmomi mendukung berbagai akses fungsionalitas pada Vmware Vsphere. Selain itu, Pyvmomi sangat mudah digunakan dan memiliki beberapa contoh pemakaian. Pada tugas akhir ini, Pyvmomi digunakan sebagai Python SDK untuk mengatur Vmware Vsphere.

2.7 Proxmoxer

Proxmoxer adalah Python *wraper* untuk mengakses API Proxmox versi 2 melalui protokol HTTPS dan SSH. Modul ini dikembangkan oleh komunitas[6].

Proxmoxer sangat mudah digunakan dibanding modul yang lain. Selain itu, Proxmoxer mendukung berbagai akses fungsionalitas dalam manajemen Proxmox. Pada tugas akhri ini, Proxmoxer digunakan untuk modul komunikasi antara *middleware* dengan *server* Proxmox.

2.8 PHP

PHP adalah bahasa pemrograman *scripting* yang memiliki lisensi terbuka yang cocok untuk membuat aplikasi web dan dapat dimasukkan pada HTML. PHP dapat PHP dapat dijalankan pada sistem operasi Unix, Linux, Windows dan Mac Os. Untuk saat ini, PHP saat ini didukung berbagai web server, contohnya Apache, IIS dan lainnya. PHP juga mendukung pemrograman prosedural, pemrograman berbasis objek dan pengabungan dari keduanya. Selain menampilkan HTML, PHP juga dapat menampilkan gambar, PDF dan Flash yang dihasilkan secara cepat. PHP juga didukung oleh berbagai macam database seperti Mysql. Untuk berkomunikasi dengan servis yang lain, PHP mendukung berbagai protokol seperti LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM dan *raw socket*[7].

Dengan banyaknya dukungan web *server* terhadap PHP, pada tugas akhir ini, bahasa pemrograman PHP akan digunakan untuk membangun *interface* web.

2.9 Redis

Redis adalah perangkat lunak terbuka penyimpanan data dengan lisensi BSD yang digunakan sebagai wadah untuk menyimpan struktur data dalam *memory* yang digunakan sebagai basis data, cache dan message broker. Redis mendukung penyimpanan dengan berbagai tipe data seperti *strings*, *hashes*, *lists*, *sets*, *sorted sets*, *bitmaps*, *hpreloglogs* dan *geospatial indexes*. Selain itu, salah satu penggunaan Redis yang umum digunakan adalah sebagai *task queue*.

Dalam hal meningkatkan kinerjanya, Redis bekerja pada *in-memory dataset*. Data yang dikelola oleh Redis berada dalam *memory* sehingga proses membaca dan menulisnya akan cepat dan efisien, selain itu data tersebut bisa dijadikan *persistent*

dengan menyimpannya ke dalam *disk*[8].

Pada tugas akhir ini, Redis akan digunakan sebagai tempat *queue* yang akan diintegrasikan dengan Python Celery.

2.10 MySQL

MySQL adalah merupakan salah basis data resional terbuka yang awalnya dikembangkan oleh MySQL AB. Mulai tahun 2009, setelah Oracle Corporation mengakusisi Sun Microsystems, MySQL dikembangkan dan distribusikan oleh Oracle Corporation. MySQL tersedia sebagai basis data gratis di bawah lesensi *GNU General Public License* (GPL), tetapi juga tersedia lisensi komersial.

MySQL bekerja pada banyak *platform*, seperti Compaq Tru64, DEC OSF, FreeBSD, IBM AIX, HP-UX, Linux, Mac OS X, Novell NetWare, OpenBSD, QNX, SCO, SGI IRIX, Solaris (versions 8, 9 and 10) dan Microsoft Windows. MySQL juga menyediakan *source code* apabila MySQL dalam bentuk *binaries* tidak tersedia pada platform yang digunakan. MySQL menyediakan *API* untuk berbagai bahasa pemrograman seperti C, C++, Java, Perl, PHP, Ruby, Tcl dan lainnya.

MySQL juga menawarkan banyak jenis mekanisme untuk mengelola data, yang dikenal sebagai *storage engines*. MySQL telah lama mendukung beberapa *storage engines*, yaitu MyISAM (standar umum pada semua sistem operasi kecuali Windows), MEMORY (sebelumnya dikenal sebagai HEAP), InnoDB (standar umum pada Windows) dan MERGE. Versi 5 menambahkan mesin ARCHIVE, BLACKHOLE, CSV, FEDERATED dan EXAMPLES[9].

Pada tugas akhir ini, MySQL akan digunakan sebagai basis data penyimpanan data.

2.11 Algoritma *Analytical Hierarchy Process*

Analytical Hierarchy Process atau yang sering disebut AHP adalah teknik terstruktur untuk menangani keputusan yang kompleks berdasarkan matematika dan psikologi. AHP dikembangkan oleh Thomas L. Saaty pada tahun 1970an dan telah dipelajari dan disempurnakan secara intensif sejak saat itu. AHP menyediakan kerangka kerja yang komprehensif dan rasional untuk menyusun suatu masalah keputusan, untuk mewakili dan mengkuantifikasi elemen-elemennya, untuk menghubungkan elemen-elemen tersebut dengan tujuan keseluruhan dan untuk mengevaluasi solusi alternatif. AHP telah digunakan diseluruh dunia dalam bidang pemerintahan, bisnis, industri, kesehatan dan pendidikan. Awalnya prioritas ditetapkan sesuai dengan kepentingan untuk mencapai tujuan, setelah prioritas tersebut diturunkan untuk kinerja alternatif pada setiap kriteria, prioritas tersebut diturunkan berdasarkan penilaian berpasangan menggunakan penilaian, atau jatah pengukuran dari skala jika ada. AHP telah digunakan di banyak bidang karena kemampuan untuk menentukan peringkat pilihan sesuai dengan keefektifannya dalam mencapai tujuan yang bertentangan. Penelitian telah berhasil menggunakan AHP dalam memilih satu alternatif dari banyak alokasi sumber daya, peramalan, manajemen kualitas total, rekayasa ulang proses bisnis, penyebaran fungsi kualitas dan nilai skor yang berimbang. AHP adalah metode yang lebih baik, di mana parameter adalah kategori ke dalam sub parameter[10]. Skala umum pada AHP [11] ditunjukkan pada Tabel 2.1

Tabel 2.1: Daftar Skala Prioritas pada AHP

Tingkat Kepentingan	**Definisi**	**Penjelasan**
1	Sama Penting	2 faktor berkontribusi senilai terhadap objektif yang ada.
3	Sedikit Lebih Penting	salah satu faktor lebih penting sedikit dibandingkan faktor yang lain.
5	Lebih Penting	salah satu faktor lebih penting dibandingkan faktor yang lain.
7	Sangat Lebih Penting	salah satu faktor sangat lebih penting dibandingkan faktor yang lain.
9	Benar-benar Lebih Penting	Bukti yang mendukung salah satu faktor dari yang lain telah mencapai kemungkinan yang tertinggi.
2,4,6,8	Nilai Tengah	Nilai saat dimana dibutuhkan kompromi.

Untuk membuat keputusan dengan cara yang terorganisasi untuk menghasilkan prioritas, pengguna perlu menguraikan keputusan menjadi langkah-langkah berikut[12]:

1. Pengaturan Masalah dan Pemilihan Kriteria.
   Langkah pertama adalah menguraikan masalah pengambilan keputusan menjadi bagian-bagian

penyusunnya. Dalam bentuk yang paling sederhana.

2. Menetapkan kriteria prioritas dengan perbandingan berpasangan (pembobotan).
3. Untuk setiap pasangan kriteria, pengambil keputusan diperlukan untuk menjawab pertanyaan seperti ”Seberapa penting A ke B?” untuk menilai prioritas ”relatif” dari setiap kriteria. Tugas ini dilakukan dengan menetapkan bobot antara 1 dan 9 seperti yang ditunjukkan pada Tabel 2.1 terhadap kriteria yang lebih penting, sementara nilai timbal balik dari nilai ini akan diberikan kepada pasangan kriteria. Pembobotan ini kemudian akan dinormalisasi untuk menambah berat badan untuk setiap kriteria yang disebut Option Performance Matrix (OPM).
4. Perbandingan berpasangan dari pilihan pada setiap kriteria (penilaian).
5. Untuk setiap pasangan dalam setiap kriteria, pilihan yang lebih baik akan diberi nilai antara 1 dan 9, sementara pasangan opsi lainnya akan diberi nilai yang sama dengan nilai timbal balik dari nilai ini. Setiap nilai akan menunjukkan seberapa baik ”A” opsi untuk kriteria ”B”.
6. Terdapat tiga langkah untuk menghitung Weight of Criteria, yakni:
   (a) Hitung n *root of product*
   (b) Priority Vector = nilai dari *n root of product* / jumlah dari *3rd root of product*
   (c) Hitung jumlah nilai dari setiap kolom
7. Dapatkan skor keseluruhan untuk setiap opsi
   Pada langkah terakhir, nilai setiap opsi digabungkan dengan bobot kriteria untuk menghasilkan nilai keseluruhan untuk setiap opsi. Sejauh mana pilihan tersebut memenuhi kriteria akan diukur berdasarkan seberapa penting kriteria tersebut. Ini dilakukan dengan rumus:

$$\sum \text{(kriteria pembobotan * bobot OPM)}$$

Pada tugas akhir ini, Algoritma *Analytical Hierarchy Proses* akan digunakan sebagai algoritma pemilihan *server* terbaik untuk alokasi *virtual machine* baru.

(Halaman ini sengaja dikosongkan)

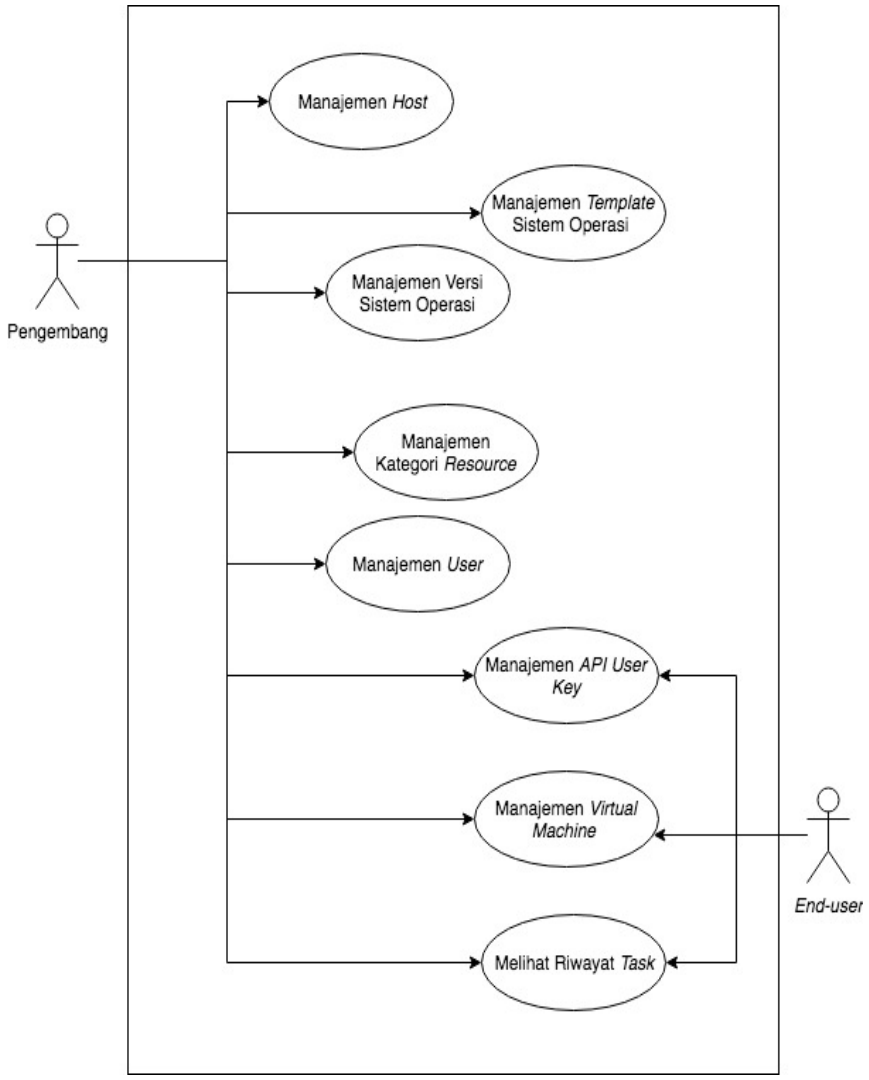
BAB III

DESAIN DAN PERANCANGAN

Pada bab ini dibahas mengenai analisis dan perancangan sistem.

3.1 Kasus Penggunaan

Terdapat dua aktor dalam sistem ini, yaitu pengembang (*administrator*) dan *end-user* (pengguna) dari aplikasi web yang dikelola oleh sistem. Diagram kasus penggunaan digambarkan pada Gambar 3.1.



Gambar 3.1: Diagram Kasus Penggunaan

Diagram kasus penggunaan pada Gambar 3.1 dideskripsikan masing-masing pada Tabel 3.1.

Tabel 3.1: Daftar Kode Kasus Penggunaan

Kode Kasus Penggunaan	**Nama Kasus Penggunaan**	**Keterangan**
UC-0001	Manajemen *Virtual Machine*.	Pengembang dan *end-user* dapat mangatur *virtual machine*.
UC-0002	Manajemen *API Secret Key*.	Pengembang dan *end-user* dapat mengatur *secret key* untuk akses dari aplikasi lain.
UC-0003	Melihat Riwayat *Task*.	Pengembang dan *end-user* dapat melihat *task* yang sedang berjalan atau *task* yang sudah dikerjakan.
UC-0004	Manajemen *Host*.	Pengembang dapat mengatur *host hypervisor* yang tersedia untuk menunjang sistem.
UC-0005	Manajemen *Template* Sistem Operasi.	Pengembang dapat mengatur template sistem operasi yang akan digunakan untuk pembuatan *virtual machine*.
UC-0006	Manajemen Versi Sistem Operasi.	Pengembang dapat mengatur versi sistem operasi.

Tabel 3.1: Daftar Kode Kasus Penggunaan

Kode Kasus Penggunaan	**Nama Kasus Penggunaan**	**Keterangan**
UC-0007	Manajemen Kategori *Resource*.	Pengembang dapat mengatur tipe *resource* untuk pembuatan *virtual machine*.
UC-0008	Manajemen *User*.	Pengembang dapat mengatur *user* yang dapat mengakses sistem.

3.2 Arsitektur Sistem

Pada sub-bab ini, dibahas mengenai tahap analisis dan kebutuhan bisnis dan desain dari sistem yang akan dibangun.

3.2.1 Desain Umum Sistem

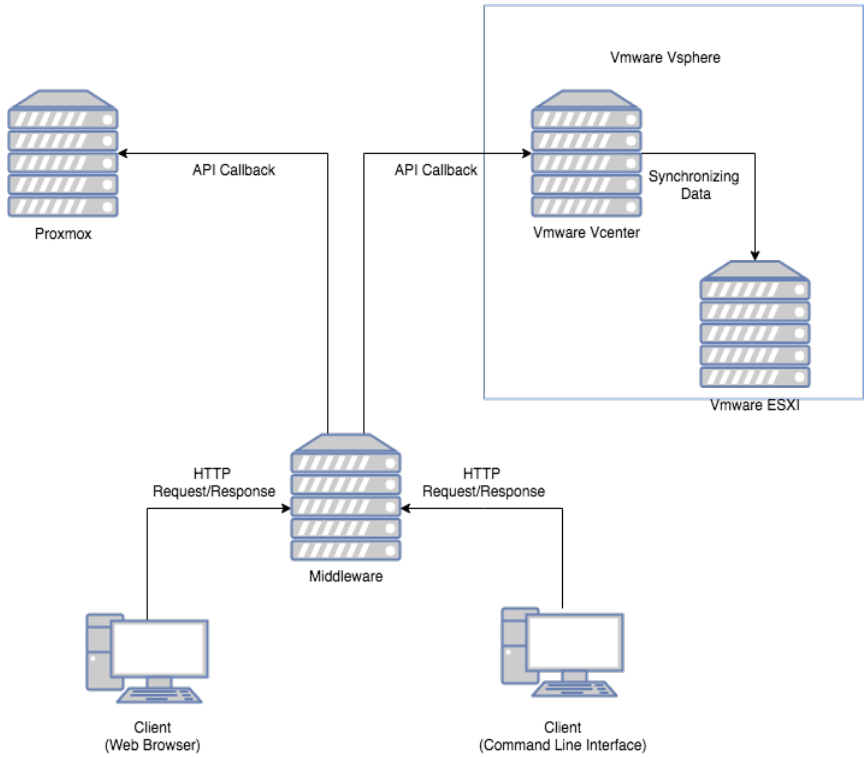
Sistem yang akan dibuat yaitu sistem yang dapat melakukan manajemen alokasi *virtual machine* pada *hypervisor* dalam lingkungan yang heterogen. Sistem yang dikembangkan oleh penulis mendukung alokasi *virtual machine* pada *hypervisor Proxmox* dan *Vmware Vsphere*. Pada saat alokasi *virtual machine* baru, sistem akan memilih server terbaik dengan memperhitungkan presentase ketersediaan sumber daya CPU, memori dan *Storage* dengan menggunakan algoritma *Analytic Hierarchy Process (AHP)*. Setelah menentukan *server* terbaik, sistem akan mengirimkan perintah-perintah alokasi *virtual machine* baru sesuai dengan *hypervisor* yang terinstall pada *server*.

Setiap permintaan alokasi maupun manajemen *virtual machine* akan ditampung pada *queue* terlebih dahulu.

Permintaan-permintaan yang ditampung pada *queue* tersebut akan dikerjakan oleh *worker* yang tersedia.

Sistem ini akan digunakan oleh pengguna, yaitu *end-user* dari aplikasi yang mana hanya bisa melakukan manajemen *manajemen virtual machine*, membuat *secret key* dan melihat riwayat *task*. Selain itu juga digunakan oleh pengembang, yaitu orang mengelola aplikasi. Pengembang dapat melakukan manajemen *host* atau *server*, manejemen *template* sistem operasi, manajemen pengguna *(user)* manajemen kategori *resource*, manajemen versi sistem operasi dan dapat melakukan perkerjaan yang dilakukan oleh pengguna *end-user*.

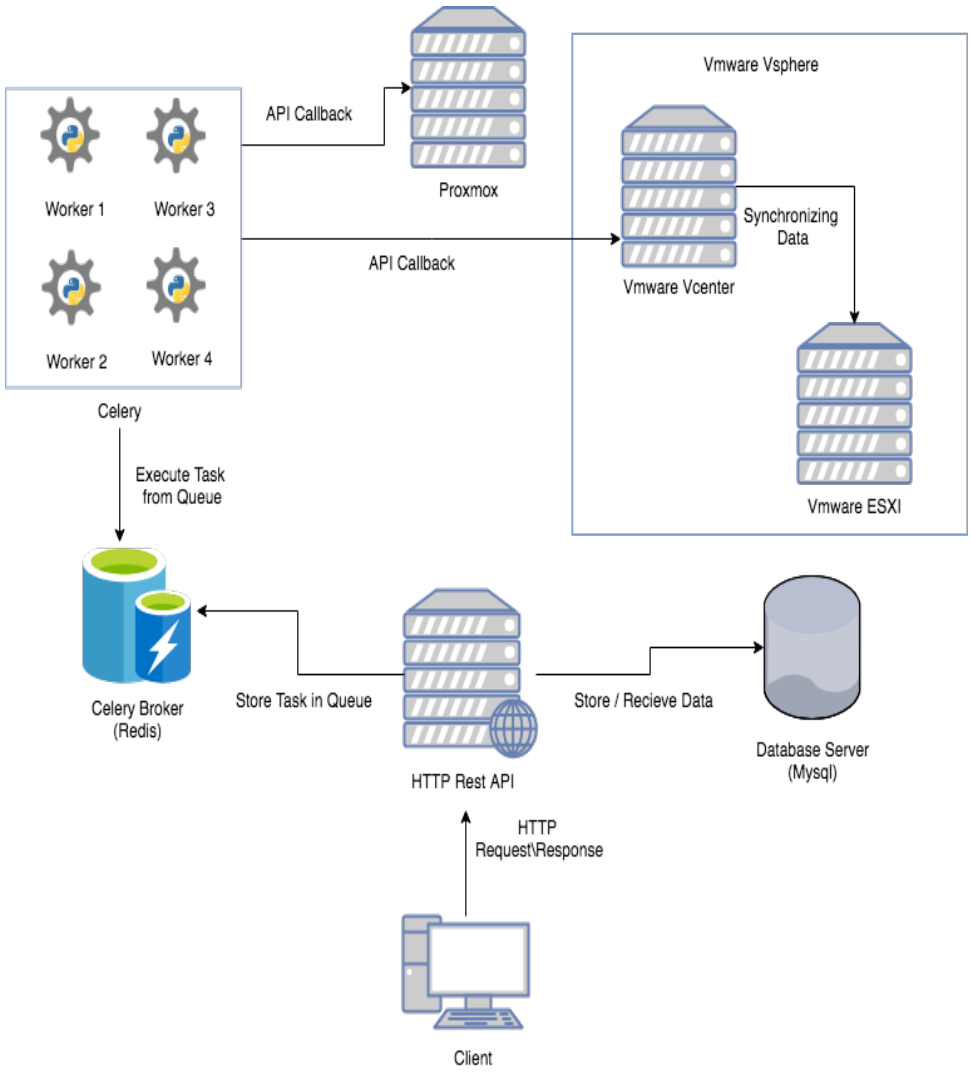
Sistem ini dapat diakses oleh pengguna maupun pengembang melalui aplikasi web dan *command line interface (CLI)*. Aplikasi web dan *command line interface (CLI)* akan meneruskan permintaan pada *middleware* yang merupakan inti penting pada sistem ini. *Middleware* berupa *webservice REST API*. Pada saat alokasi *virtual machine* baru, *middleware* berperan untuk memilih server terbaik dan menerjemahkan permintaan pengguna dalam bentuk permintaan yang dimengerti oleh *hypervisor* yang terinstal pada *server* terbaik. Penjelasan secara umum arsitektur sistem akan diuraikan pada Gambar 3.2.



Gambar 3.2: Desain Umum Sistem

3.2.2 Desain *Middleware*

Middleware digunakan sebagai penerjemaah permintaan dari pengguna agar dikenali oleh *hypervisor*. Selain itu *Middleware* juga bertugas untuk memilih *server* terbaik untuk alokasi *virtual machine*. *Middleware* terdari berbagi layanan yaitu, *HTTP Rest API*, basis data, *tasks queue* dan *worker* yang bertugas menjalankan pekerjaan yang disimpan di *queue*. Arsitektur *Middleware* dapat dilihat pada Gambar 3.3.

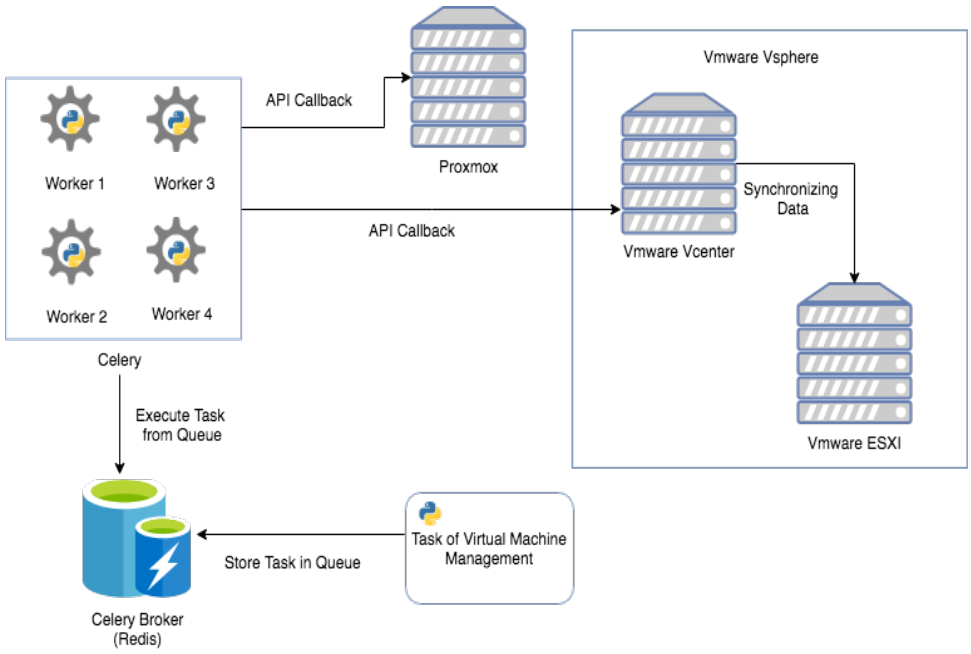


Gambar 3.3: Desain *Middleware*

HTTP Rest API dibangun menggunakan Flask yang sudah terintegrasi dengan modul Pyvmomi dan Proxmoxer. Modul Pyvmomi dan Proxmoxer digunakan sebagai penghubung antara *Middleware* dengan *hypervisor*. Basis data MySQL terhubung dengan *HTTP Rest APi* sebagai tempat penyimpanan data dari sistem, seperti data pengguna, data *host* yang sudah terinstal *hypervisor* dan sebagainya. Untuk mengeksekusi permintaan dari pengguna, *HTTP Rest API* terhubung Celery sebagai *task queue* dan sebagai *worker*. Celery terhubung dengan basis data Redis sebagai tempat penyimpanan *queue*.

3.2.3 Perancangan *Task Queue*

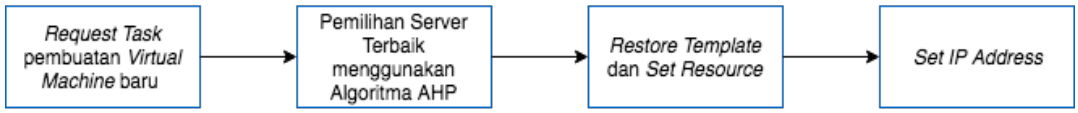
Pada sistem ini, akan banyak proses yang berjalan dalam jangka waktu yang panjang karena banyaknya eksekusi perintah didalamnya. Jika proses tersebut dipanggil melalui protokol HTTP, maka umpan balik yang diberikan menunggu semua proses yang berkaitan dengan penggunaan dari pengguna selesai semua. Hal tersebut membuat pengguna yang melakukan permintaan perlu menunggu sampai selesai dan hal tersebut tidak efisien. Untuk mengatasi hal tersebut, proses yang memerlukan banyak perintah, akan dimasukkan ke dalam sebuah *queue* atau yang bisa disebut sebagai *task queue*. Untuk *task queue* menggunakan modul Celery yang menggunakan basis data Redis sebagai wadah untuk menampung perintah atau fungsi yang akan dikerjakan dalam bentuk *queue*. Modul Celery dijalankan sebagai layanan tersendiri dan terintegrasikan dengan Flask. Pada modul Celery secara *default* menyediakan 4 *worker* yang digunakan untuk mengerjakan perintah atau fungsi yang ditampung pada *queue*. *Worker* tersebut akan menjalankan perintah secara bergantian sampai tidak ada lagi perintah atau fungsi yang berada di *queue*. Desain aristektur *Queue* dapat dilihat pada Gambar 3.4.



Gambar 3.4: Desain *Queue*

3.2.4 Perancangan Alokasi dan *Provisioning Virtual Machine* Baru

Alokasi dan *provisioning virtual machine* merupakan salah satu bagian dari fungsi sistem yang dibuat oleh penulis. Pengalokasian *virtual machine* baru pada *host atau server* yang tersedia melalui proses perhitungan algoritma AHP untuk menentukan *host atau server* terbaik. Setelah mendapatkan *server* terbaik, sistem akan mengambil informasi *server* terbaik pada basis data untuk mengetahui *hypervisor* yang ter-*install*. Untuk pengalokasian *virtual machine* baru pada Proxmox, sistem mengirimkan perintah melalui protokol SSH dan melalui modul Proxmoxer. Sedangkan untuk pengalokasian pada Vmware Vsphere, sistem mengirimkan perintah melalui modul Pyvmomi. Setelah berhasil me-*restore template virtual machine*, sistem akan mengatur IP *virtual machine* berdasarkan *hypervisor* dan sistem operasinya. Desain skenario alokasi *virtual machine* baru dapat dilihat pada Gambar 3.5.

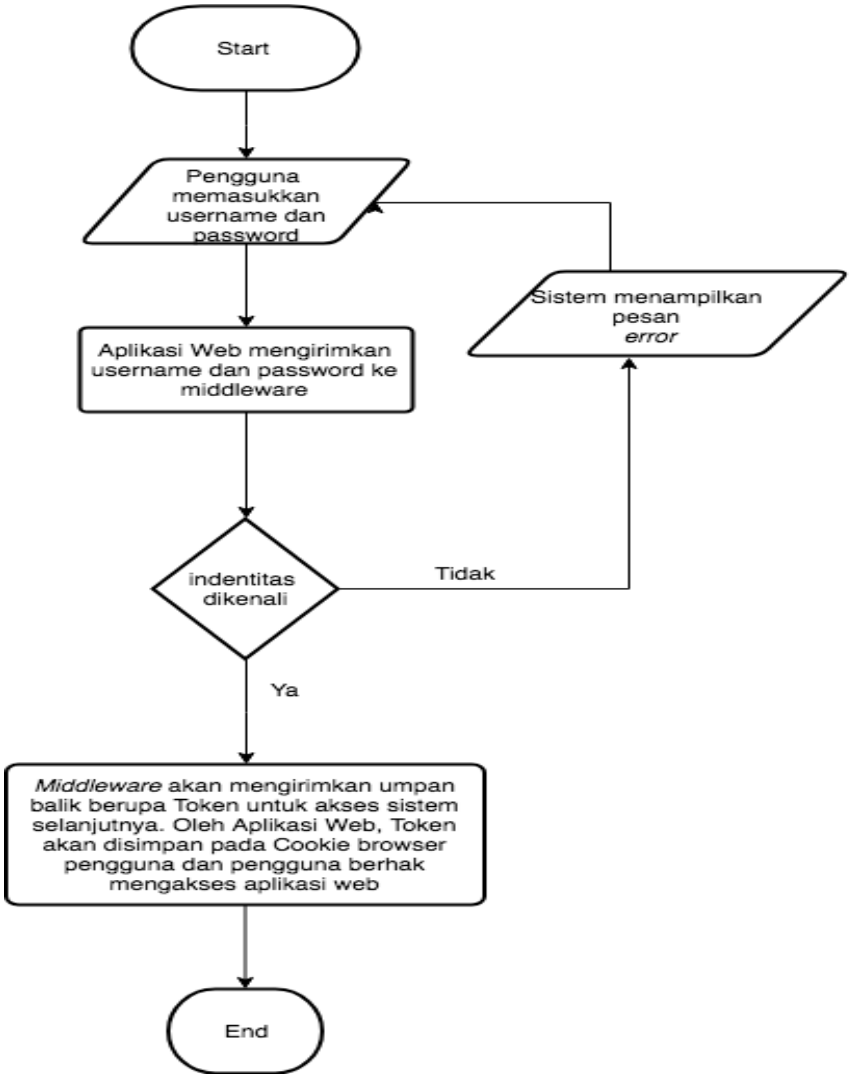


Gambar 3.5: Desain Skenario Alokasi dan *Provisioning Virtual Machine* Baru

3.2.5 Desain Web *Interface*

Pada sistem ini, pengguna dapat mengakses melalui dua *interface* yaitu web dan *command line interface*. *Interface* web pada sistem dibuat menggunakan kerangka kerja Laravel yang menggunakan bahasa pemrogramaan PHP. Web dapat diakses oleh *end-user* dan pengembang. Pada sistem ini web menjadi *interface* yang paling penting karena pengguna dapat melakukan pengaturan terhadap sistem seperti memanajemen host, memanajemen user, memanejemen *template*, memanajemen versi sistem operasi yang didukung, memanajemen kategori *resource* yang digunakan untuk menentukan *resource virtual machine*, memanajemen *virtual machine*, melihat riwayat pekerjaan dan memanajemen *API Secret Key*. Setiap permintaan pengguna melalui *interface* web akan dihubungkan langsung dengan *HTTP Rest API* sesuai *parameter* permintaan yang sudah ditentukan oleh *HTTP Rest API*. Umban Balik dari setiap permintaan pengguna, akan ditampilkan oleh web termasuk apabila *paramter* permintaan.

Untuk mengakses *interface* web, pengguna membutuhkan *akun* yang sudah terdaftar pada sistem. Akun hanya bisa ditambahkan oleh pengguna yang memiliki hak akses sebagai *administrator*.



Gambar 3.6: *Flowchart* Autentifikasi dan Otorisasi Pada *Interface* Web

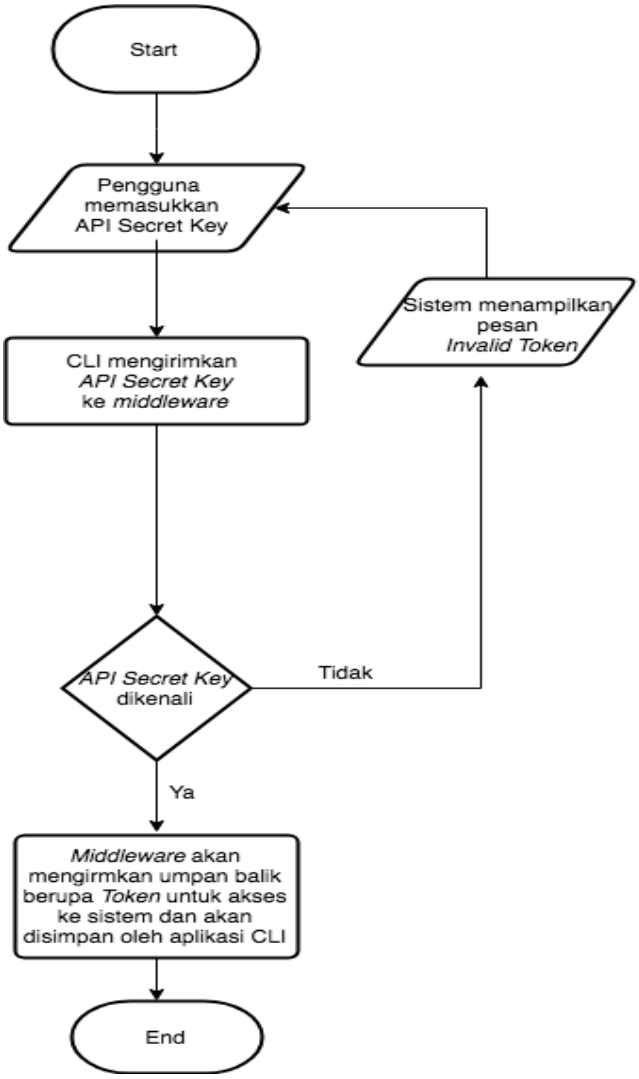
Berdasarkan *Flowchart* Autentifikasi dan Otorisasi Pada *Interface* Web **??**, pengguna awalnya memasukkan *username* dan *password* terlebih dahulu. Kemudian *interface* web akan mengirimkan *username* dan *password* ke *middleware*. Apabila identitas tidak dikenali, sistem akan menampilkan pesan *error* bahwa pengguna tidak dikenal. Apabila identitas dikenali, *middleware* akan mengirimkan umpan balik berupa *Token*. *Token* akan disimpan oleh *Interface* web pada *Cookie Browser* pengguna untuk akses sistem selanjutnya dan pengguna akan diarahkan ke halaman dasbor *Interface* web.

3.2.6 Desain *Command Line Interface*

Selain *interface* web, sistem ini dapat diakses melalui *Command Line Interface*. *Command Line Interface* dapat diinstal pada komputer pengguna. *Command Line Interface* dibuat menggunakan bahasa pemrograman Python. *End-user* dan pengembang hanya dapat melakukan berbagai pekerjaan saja, seperti melihat *virtual machine*, mematikan atau menyalakan *virtual machine*, menghapus *virtual machine*, melihat kategori *resource* dan melihat sistem operasi yang tersedia.

Setiap permintaan melalui *Command Line Interface* akan diteruskan ke *HTTP Rest API* sesuai *paramter* permintaan yang dibutuhkan. Untuk *Command Line Interface* menggunakan menggunakan *API Secret Key* yang sudah dibuat melalui *interface* web. *API Secret Key* digunakan untuk autentifikasi dan otorisasi hak akses pada sistem.

Berdasarkan *flowchart* autentifikasi dan otorisasi pada Gambar 3.7, pengguna awalnya mengatur *API Secret Key* terlebih dahulu. Kemudian aplikasi *command line interface* akan mengirimkan *API Secret Key* ke *Middleware* untuk proses autentifikasi dan otorisasi. Apabila *API Secret Key* tidak dikenali, maka *middleware* akan mengirim umpan balik yang berisi pesan *error*, *Invalid Token*. Apabila *API Secret Key* dikenali, maka *middleware* akan mengirimkan umpan balik berupa *Token* untuk mengakses sistem selanjutnya. *Token* akan disimpan oleh aplikasi CLI.



Gambar 3.7: *Flowchart* Autentifikasi dan Otorisasi Menggunakan *API Secret Key*

BAB IV

IMPLEMENTASI

Bab ini membahas implementasi sistem Pengendali Elastisitas secara rinci. Pembahasan dilakukan secara rinci untuk setiap komponen yang ada, yaitu: *middleware*, alokasi *virtual machine* baru, *interface* web dan *command line interface*.

4.1 Lingkungan Implementasi

Dalam mengimplementasikan sistem, digunakan beberapa perangkat pendukung sebagai ebrikut.

4.1.1 Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan sistem adalah sebagai berikut:

1. *Middleware*, *processor* Intel(R) Core(TM) i5 CPU @ 2.5 GHz dan Ram 4GB.
2. *Server* Proxmox A dengan IP 10.151.36.222, *processor* Intel(R) Xeon(R) CPU E3-1220 V2 @ 3.10GHz dan Ram 8GB.
3. *Server* Proxmox B dengan IP 10.151.38.100, *processor* Intel(R) Xeon(R) CPU E5507 @ 2.27GHz dan Ram 8GB.
4. *Server* Vmware ESXI A dengan IP 10.199.14.150, *processor* Intel(R) Xeon(R) CPU E5420 @ 2.50GHz dan Ram 4GB.
5. *Server* Windows Server 2016 dan terinstal Vmware Vcenter dengan IP 10.151.38.38, *processor* Intel(R) Core(TM) i7 CPU @ 2.50GHz dan Ram 16GB.

4.1.2 Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan adalah sebagai berikut:

- Sistem Operasi Mac OS High Sierra 10.13.4

- Proxmox
- Vmware EXSI
- Vmware Vcenter
- Windows Server 2016
- Python 2.7
- Redis
- MySQL
- Flask
- Celery
- Insomnia

4.2 Implementasi *Middleware*

Server Middleware merupakan *server* inti dari sistem. Pada *server* ini yang akan mengelola keseluruhan data dari sistem. Pada server ini, selain mengelola kesuluruhan data, *server middleware* bertindak untuk mengambil keputusan dalam alokasi *virtual machine* baru. Selain itu semua permintaan dari pengguna akan diterjemahkan dalam bentuk *parameter* yang dibutuhkan untuk meneruskan permintaan dari pengguna ke hypervisor. *Server middleware* memilik IP 10.151.36.4 dan dapat diakses menggunakan port 5000.

4.2.1 Skema Basis Data Middleware Menggunakan MySQL

Untuk mengelola data yang ada pada sistem, dibutuhkan basis data sebagai tempat penyimpanannya, yaitu MySQL. MySQL *server* yang digunakan adalah versi 5.7.22. Data yang disimpan pada basis data adalah data *hypervisor*, data *host* yang sudah terinstal *hypervisor*, data sistem operasi yang didukung oleh sistem, data versi sistem operasi, data riwayat pekerjaan, data kategori *resource*, data *template* sistem operasi, data *users*, data *virtual machine*, data relasi kepemilikan *virtual machine* dengan pengguna dan data *API Secret Key*.

4.2.1.1 Tabel *Hypervisors*

Pada tabel *hypervisors* menyimpan data-data *hypervisor* yang didukung oleh sistem. Nama *hypervisor* tidak di-*hardcode* pada sistem agar dapat dilakukan pengembangan kedepannya. Berikut definisi tabel *hypervisors* pada Tabel 4.1.

Tabel 4.1: Tabel Hypervisors

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.
2	*name*	varchar(45)	Menunjukan nama *hypervisor* yang didukung oleh sistem.

4.2.1.2 Tabel *OS Distributions*

Pada tabel *os_distributions* menyimpan data-data distribusi sistem operasi yang didukung oleh sistem. Nama distribusi sistem operasi tidak di-*hardcode* pada sistem agar dapat dilakukan pengembangan kedepannya. Berikut definisi tabel *os_distributions* pada Tabel 4.2.

Tabel 4.2: Tabel OS Distributions

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.

Tabel 4.2: Tabel OS Distributions

No	Kolom	Tipe	Keterangan
2	*name*	varchar(45)	Menunjukan nama distribusi sistem operasi yang didukung oleh sistem.
3	logo	longtext	Menunjukan logo distribusi sistem operasi yang didukung oleh sistem. Disimpan dalam bentuk gambar yang sudah di-*endcode base64*.

4.2.1.3 Tabel *Request Categories*

Pada tabel *request_categories* menyimpan data-data kategori *resource* yang digunakan untuk menentukan *resource virtual machine*. Berikut definisi tabel *request_categories* pada Tabel 4.3.

Tabel 4.3: Tabel *Request Categories*

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.
2	*name*	varchar(45)	Menunjukan nama kategori *resource*.

Tabel 4.3: Tabel *Request Categories*

No	Kolom	Tipe	Keterangan
3	*storage*	varchar(45)	Menunjukan besarnya *storage* yang digunakan untuk pembuatan *virtual machine*.
4	*memory*	varchar(45)	Menunjukan besarnya *memory* yang digunakan untuk pembuatan *virtual machine*.
5	*core*	varchar(45)	Menunjukan besarnya *core cpu* yang digunakan untuk pembuatan *virtual machine*.

4.2.1.4 Tabel *Users*

Pada tabel *users* menyimpan data-data pengguna yang dapat mengakses sistem. Berikut definisi tabel *users* pada Tabel 4.4.

Tabel 4.4: Tabel *Users*

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.
2	*username*	varchar(45)	Menunjukan *username* pengguna yang digunakan untuk *login*.

Tabel 4.4: Tabel *Users*

No	Kolom	Tipe	Keterangan
3	*password*	varchar(100)	Menunjukan *password* pengguna yang digunakan untuk *login*.
4	*name*	varchar(45)	Menunjukan nama dari pengguna.
5	*is_admin*	int	Menunjukan jabatan dari pengguna. Apabila kolom *is_admin* tidak diisi maka akan bernilai 0.

4.2.1.5 Tabel *Hosts*

Pada tabel *hosts* menyimpan data-data *server* yang dapat digunakan oleh sistem untuk pembuatan *virtual machine*. Berikut definisi tabel *hosts* pada Tabel 4.5.

Tabel 4.5: Tabel *Hosts*

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.
2	*ip*	varchar(45)	Menunjukan ip *server*.
3	*username*	varchar(100)	Menunjukan *username* yang digunakan untuk masuk ke server.
4	*password*	varchar(200)	Menunjukan *password* yang digunakan untuk masuk ke server.

Tabel 4.5: Tabel *Hosts*

No	Kolom	Tipe	Keterangan
5	*hypervi-sors_id*	char(36)	Menunjukan *foreign key* dari tabel *hypevisors*.
6	*node_name*	varchar(45)	Menunjukan nama dari server pada *hypervisor*.
7	*is_active*	int	Menunjukan status aktif atau tidaknya sebuah *server*. Apabila kolom *is_active* tidak diisi maka akan bernilai 1.

4.2.1.6 Tabel OS

Pada tabel *os* menyimpan data-data versi sistem operasi yang didukung oleh sistem. Berikut definisi tabel *os* pada Tabel 4.6.

Tabel 4.6: Tabel OS

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.
2	*name*	varchar(45)	Menunjukan versi sistem operasi.
3	*kategori*	int	Menunjukan kategori sistem operasi. Apabila kolom *kategori* tidak diisi maka akan bernilai 1 yang berarti sistem operasi Linux.

Tabel 4.6: Tabel OS

No	Kolom	Tipe	Keterangan
4	*os_distri-butions_id*	char(36)	Menunjukan *foreign key* dari tabel *os_distributions*.

4.2.1.7 Tabel *Templates*

Pada tabel *templates* menyimpan data-data *template* sistem operasi yang didukung oleh sistem. Berikut definisi tabel *templates* pada Tabel 4.7.

Tabel 4.7: Tabel *Templates*

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.
2	*file*	varchar(45)	Menunjukan nama *template*.
3	*hypervi-sors_id*	char(36)	Menunjukan *foreign key* tabel *hypervisors*.
4	*os_id*	char(36)	Menunjukan *foreign key* tabel os.
5	*username*	varchar(45)	Menunjukan *username* sistem operasi pada template dengan hak akses *superuser template*.
5	*password*	varchar(100)	Menunjukan *username* sistem operasi pada template dengan hak akses *superuser template*.

Tabel 4.7: Tabel *Templates*

No	Kolom	Tipe	Keterangan
7	*user-name_default_user*	varchar(45)	Menunjukan *username* sistem operasi pada template untuk pengguna sistem.
8	*password_default_user*	varchar(45)	Menunjukan *password* sistem operasi pada template untuk pengguna sistem.

4.2.1.8 Tabel VMS

Pada tabel *vms* menyimpan data-data *virtual machine* yang terdaftar di sistem. Berikut definisi tabel *vms* pada Tabel 4.8.

Tabel 4.8: Tabel *VMS*

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.
2	*memory*	varchar(45)	Menunjukan *memory* yang diatur pada *virtual machine*.
3	*storage*	varchar(45)	Menunjukan *storage* yang diatur pada *virtual machine*.
4	*core*	varchar(45)	Menunjukan *core* yang diatur pada *virtual machine*.

Tabel 4.8: Tabel *VMS*

No	Kolom	Tipe	Keterangan
5	*ip*	varchar(45)	Menunjukan ip yang diatur pada *virtual machine*.
6	*mac address*	varchar(45)	Menunjukan *mac address virtual machine*.
7	*hypervi-sors_id*	char(36)	Menunjukan *foreign key* tabel *hypervisors*.
8	*unique_id*	varchar(255)	Menunjukan nama unik dari *virtual machine*.
9	*name*	varchar(45)	Menunjukan nama *virtual machine*.
10	*host_id*	char(36)	Menunjukan *foreign key* tabel *hosts*.
11	*created_at*	timestamp	Menunjukan waktu pembuatan *virtual machine*.
12	*request_cate-gories_id*	char(36)	Menunjukan *foreign key* tabel *request_categories*.
13	*os_id*	char(36)	Menunjukan *foreign key* tabel os.
14	*bridge_hyper-visor*	varchar(45)	Menunjukan nama *interface bridge* pada *hypervisor*.
15	*nid_hypervisor*	varchar(45)	Menunjukan nama *network identifier* pada *hypervisor*.
16	*random*	int	Menunjukan angka acak yang digunakan untuk identitas unik *virtual machine*.

4.2.1.9 Tabel *VM Owners*

Pada tabel *vm_owners* menyimpan data-data relasi antara *virtual machine* dengan pengguna yang terdaftar di sistem. Berikut definisi tabel *vms* pada Tabel 4.9.

Tabel 4.9: Tabel *VM Owners*

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.
2	*vms_id*	char(36)	Menunjukan *foreign key* tabel *hypervisors*.
3	*users_id*	char(36)	Menunjukan *foreign key* tabel *users*.
4	*is_real_owner*	int	Menunjukan status kepemilikan *virtual machine*. Apabila tidak diisi memiliki nilai 0.

4.2.1.10 Tabel *Tasks*

Pada tabel *tasks* menyimpan data-data riwayat pekerjaan yang pernah atau sedang berjalan pada sistem. Berikut definisi tabel *tasks* pada Tabel 4.10.

Tabel 4.10: Tabel *Tasks*

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.
2	*vms_id*	char(36)	Menunjukan *foreign key* tabel *hypervisors*.
3	*users_id*	char(36)	Menunjukan *foreign key* tabel *users*.
4	*status*	int	Menunjukan status pekerjaan. Status bernilai 0 apabila pekerjaan masih berjalan, bernilai 1 apabila perkerjaan selesai dan sukses dan bernilai -1 apabila pekerjaan gagal dikerjakan.
5	*start*	timestamp	Menunjukan waktu pekerjaan dimulai.
6	*end*	timestamp	Menunjukan waktu pekerjaan berakhir.
7	*description*	text	Menunjukan status pesan pekerjaan.
8	*celery_id*	varchar(45)	Menunjukan id pekerjaan pada *celery queue*.
9	*last_step*	int	Menunjukan langkah terakhir yang dikerjaan oleh sistem.
10	*action*	varchar(255)	Menunjukan nama pekerjaan yang dilakukan.

4.2.1.11 Tabel *Tokens*

Pada tabel *tokens* menyimpan data-data *API Secret Key*. Berikut definisi tabel *tokens* pada Tabel 4.11.

Tabel 4.11: Tabel *Tokens*

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.
2	*users_id*	char(36)	Menunjukan *foreign key* tabel *users*.
3	*token*	longtext	Menunjukan *API Secret Key*.
4	*name*	timestamp	Menunjukan nama *API Secret Key*.
5	*is_write*	int	Menunjukan hak akses *API Secret Key*. Ketika kolom *is_write* bernilai 1 (satu), maka *API Secret Key* dapat melakukan aksi pengubahan data.

4.2.1.12 Tabel IP *Addresses*

Pada tabel *ip_addresses* menyimpan data-data *pool* ip yang dapat digunakan pada suatu *server*. Berikut definisi tabel *ip_address* pada Tabel 4.12.

Tabel 4.12: Tabel *IP Addresses*

No	Kolom	Tipe	Keterangan
1	id	char(36)	Sebagai primary key pada tabel, nilai pada kolom ini ada dalam bentuk *Universally Unique Identifier (UUID)*.
2	*hosts_id*	char(36)	Menunjukan *foreign key* tabel *hosts*.
3	*ip*	varchar(45)	Menunjukan *API Secret Key*.
4	*netmask*	varchar(45)	Menunjukkan *netmask* suatu ip.
5	*gateway*	varchar(45)	Menunjukan *gateway* suatu ip.

4.2.2 Implementasi Autentifikasi dan Otorisasi pada *Middleware*

Untuk mengakses *middleware* diperlukan *token* yang dikirimkan pada *header* paket. *Token* didapatkan setelah *login* pada *middleware*. Terdapat dua mekanisme autentifikasi dan otorisasi yaitu dengan mengirimkan data *username* dan *password* atau menggunakan *API Secret Key* yang didapatkan melalui *interface* web.

4.2.2.1 Implementasi Autentifikasi dan Otorisasi dengan *username* dan *password*

Untuk melakukan autentifikasi dan otorisasi menggunakan *username* dan *password*, pengguna dapat mengakses pada *end-point* `/api/v1/auth/login/form` dengan menggunakan *Method Post*. *Parameter* yang dibutuhkan untuk mengakses

end-point tersebut adalah *username* dan *password*.

4.2.2.2 Implementasi Autentifikasi dan Otorisasi dengan *API Secret Key*

Untuk melakukan autentifikasi dan otorisasi menggunakan *API Secret Key*, pengguna dapat mengakses pada *end-point* `/api/v1/auth/login/token` dengan menggunakan *Method Post*. *Parameter* yang dibutuhkan untuk mengakses *end-point* tersebut adalah *token*.

4.2.2.3 Implementasi *Token* untuk Autentifikasi dan Otorisasi

Token dikirim oleh *middleware* dalam bentuk *response* dari permintaan autentifikasi dan otorisasi oleh pengguna yang berupa *JSON*. Bentuk *response* dapat dilihat pada Gambar 4.1.


```
{
        "data":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IjUxMTQxMDAwMjAiLCJ1c2VyX3Rva2VuIjpu
dWxsLCJ1YSI6Imluc29tbmlhLzUuMTYuMiIsImlzX2FkbWluIjoxLCJleHAiOjE1MjcyNjU4NzUsImlzX3ZpYV90b2
tlbiI6MCwiaXAiOiIxMjcuMC4wLjEiLCJpc192aWFfd2ViIjoxLCJpZCI6Ijg2YjRlNDYwLTE2MmEtMTFlOC1hNjRl
LWE4N2M3M2NjYTZkOSJ9.eqbi726YrgL82FTynf74JY3Tk8UiyzubBa6SYrJFwPk",
        "status": 200
}
```


Gambar 4.1: *Response Token* dari *Middleware*

Token dikirimkan oleh *middleware* pada *index JSON* dengan nama data dan dalam bentuk *JSON Web Token (JWT)*. Ketika *token* tersebut di-*decode*, maka *token* berisi detail pengguna yang sudah terautentifikasi dan terotorisasi.

4.2.3 Implementasi *Endpoint* pada *Middleware*

Untuk mempermudah manajemen sistem, penulis menyediakan *end-point* pada *middleware*.

4.2.3.1 *End-Point* Manajemen *Host*

Untuk manajemen *host*, terdapat lima operasi yang didapat dilakukan yaitu melihat daftar *host*, menambah *host*, melihat detail data *host*, mengubah data *host* dan menghapus *host*. Untuk detail *end-point* dapat dilihat pada Tabel 4.13.

Tabel 4.13: Tabel *End-point* Manajemen *Host*

No	*End-point*	*Method*	Keterangan
1	`/api/v1/hypervisor/<hypervisor_id>/host`	*GET*	Untuk melihat daftar *host* pada *hypervisor*.
2	`/api/v1/hypervisor/<hypervisor_id>/host/create`	POST	Untuk menambah *host*.
3	`/api/v1/hypervisor/<hypervisor_id>/host/<host_id>`	GET	Untuk melihat data *host*.
4	`/api/v1/hypervisor/<hypervisor_id>/host/<host_id>`	PUT	Untuk mengubah data *host*.

Tabel 4.13: Tabel *End-point* Manajemen *Host*

No	*End-point*	*Method*	Keterangan
5	`/api/v1/hypervisor/<hypervisor_id>/host/<host_id>`	DELETE	Untuk menghapus data *host*.

4.2.3.2 *End-Point* Manajemen Kategori *Resource*

Untuk manajemen kategori *resource*, terdapat lima operasi yang didapat dilakukan yaitu melihat daftar kategori *resource*, menambah kategori *resource*, melihat detail data kategori *resource*, mengubah data kategori *resource* dan menghapus kategori *resource*. Untuk detail *end-point* dapat dilihat pada Tabel 4.14.

Tabel 4.14: Tabel *End-point* Manajemen Kategori *Resource*

No	*End-point*	*Method*	Keterangan
1	`/api/v1/requestcategory`	*GET*	Untuk melihat daftar kategori *resource*.
2	`/api/v1/requestcategory/create`	POST	Untuk menambah kategori *resource*.
3	`/api/v1/requestcategory/<resource_id>`	GET	Untuk melihat data kategori *resource*.
4	`/api/v1/requestcategory/<resource_id>`	PUT	Untuk mengubah data kategori *resource*.

Tabel 4.14: Tabel *End-point* Manajemen Kategori *Resource*

No	*End-point*	*Method*	Keterangan
5	`/api/v1/requestcategory/<resource_id>`	DELETE	Untuk menghapus data kategori *resource*.

4.2.3.3 *End-Point* Manajemen Versi Sistem Operasi

Untuk manajemen versi sistem operasi, terdapat lima operasi yang didapat dilakukan yaitu melihat daftar versi sistem operasi, menambah versi sistem operasi, melihat detail data sistem operasi, mengubah data versi sistem operasi dan menghapus versi sistem operasi. Untuk detail *end-point* dapat dilihat pada Tabel 4.15.

Tabel 4.15: Tabel *End-point* Manajemen Versi Sistem Operasi

No	*End-point*	*Method*	Keterangan
1	`/api/v1/os`	*GET*	Untuk melihat daftar versi sistem operasi.
2	`/api/v1/os/create`	POST	Untuk menambah versi sistem operasi.
3	`/api/v1/os/<os_id>`	GET	Untuk melihat data versi sistem operasi.
4	`/api/v1/os/<os_id>`	PUT	Untuk mengubah data versi sistem operasi.
5	`/api/v1/os/<os_id>`	DELETE	Untuk menghapus data suatu versi sistem operasi.

4.2.3.4 *End-Point* Manajemen *Template* Sistem Operasi

Untuk manajemen *template* sistem operasi, terdapat lima operasi yang didapat dilakukan yaitu melihat daftar *template* sistem operasi, menambah *template* sistem operasi, melihat detail

data *template* sistem operasi, mengubah data *template* sistem operasi dan menghapus *template* sistem operasi. Untuk detail *end-point* dapat dilihat pada Tabel 4.16.

Tabel 4.16: Tabel *End-point* Manajemen *Template* Sistem Operasi

No	***End-point***	***Method***	**Keterangan**
1	`/api/v1/hypervisor/<hypervisor_id>/template`	*GET*	Untuk melihat daftar *template* sistem operasi.
2	`/api/v1/hypervisor/<hypervisor_id>/template/create`	POST	Untuk menambah *template* sistem operasi.
3	`/api/v1/hypervisor/<hypervisor_id>/template/<template_id>`	GET	Untuk melihat data *template* sistem operasi.
4	`/api/v1/hypervisor/<hypervisor_id>/template/<template_id>`	PUT	Untuk mengubah data *template* sistem operasi.
5	`/api/v1/hypervisor/<hypervisor_id>/template/<template_id>`	DELETE	Untuk menghapus data *template* sistem operasi.

4.2.3.5 *End-Point* Manajemen *User*

Untuk manajemen pengguna, terdapat lima operasi yang didapat dilakukan yaitu melihat daftar pengguna, menambah pengguna, melihat detail data pengguna, mengubah data pengguna dan menghapus pengguna. Untuk detail *end-point* dapat dilihat pada Tabel 4.17.

Tabel 4.17: Tabel *End-point* Manajemen *User*

No	*End-point*	*Method*	Keterangan
1	`/api/v1/user`	*GET*	Untuk melihat daftar pengguna.
2	`/api/v1/user/create`	POST	Untuk menambah pengguna.
3	`/api/v1/user/<user_id>`	GET	Untuk melihat pengguna.
4	`/api/v1/user/<user_id>`	PUT	Untuk mengubah pengguna.
5	`/api/v1/user/<user_id>`	DELETE	Untuk menghapus pengguna.

4.2.3.6 *End-Point* Manajemen *API Secret Key*

Untuk manajemen *API secret key*, terdapat lima operasi yang didapat dilakukan yaitu melihat daftar *API secret key*, menambah *API secret key*, melihat detail data *API secret key*, mengubah data *API secret key* dan menghapus *API secret key*. Untuk detail *end-point* dapat dilihat pada Tabel 4.18.

Tabel 4.18: Tabel *End-point* Manajemen *API Secret Key*

No	*End-point*	*Method*	Keterangan
1	`/api/v1/user/<user_id>/token`	*GET*	Untuk melihat daftar *API secret key*.
2	`/api/v1/user/<user_id>/token/create`	POST	Untuk menambah *API secret key*.
3	`/api/v1/user/<user_id>/token/<token_id>`	GET	Untuk melihat *API secret key*.
4	`/api/v1/user/<user_id>/token/<token_id>`	PUT	Untuk mengubah *API secret key*.
5	`/api/v1/user/<user_id>/token/<token_id>`	DELETE	Untuk menghapus *API secret key*.

4.2.3.7 *End-Point* Manajemen *Virtual Machine*

Untuk manajemen *API secret key*, terdapat 12 operasi yang didapat dilakukan yaitu melihat daftar *API secret key*, menambah *virtual machine*, melihat detail *virtual machine*, meningkatkan kategori *resource virtual machine*, menghapus *virtual machine*, menyalakan *virtual machine*, mematikan *virtual machine*, melihat pengguna *virtual machine*, membagikan *virtual machine* ke pengguna lain, menghapus pengguna dari hak kepemilikan *virtual machine*, melihat status *virtual machine* dan melihat riwayat *virtual machine*. Untuk detail *end-point* dapat dilihat pada Tabel 4.19.

Tabel 4.19: Tabel *End-point* Manajemen *Virtual Machine*

No	***End-point***	***Method***	**Keterangan**
1	`/api/v1/vm`	*GET*	Untuk melihat daftar *virtual machine*.
2	`/api/v1/vm/create`	POST	Untuk menambah *virtual machine*.
3	`/api/v1/vm/<vm_id>/status/start`	POST	Untuk menyalakan *virtual machine*.
4	`/api/v1/vm/<vm_id>/status/stop`	POST	Untuk mematikan *virtual machine*.
5	`/api/v1/vm/<vm_id>/owner`	GET	Untuk melihat daftar kepemilikan *virtual machine*.
6	`/api/v1/vm/<vm_id>`	DELETE	Untuk menghapus *virtual machine*.
7	`/api/v1/vm/<vm_id>/owner`	DELETE	Untuk menghapus pengguna dari kepemilikan *virtual machine*.
8	`/api/v1/vm/<vm_id>/owner`	POST	Untuk membagikan hak kepemilikan *virtual machine* ke pengguna lain.
9	`/api/v1/vm/<vm_id>/status/current`	GET	Untuk melihat status *virtual machine*.
10	`/api/v1/vm/<vm_id>`	GET	Untuk melihat detail *virtual machine*.
11	`/api/v1/vm/<vm_id>`	PUT	Untuk me-*resize* kategori *resource virtual machine*.

Tabel 4.19: Tabel *End-point* Manajemen *Virtual Machine*

No	*End-point*	*Method*	Keterangan
12	`/api/v1/ vm/<vm_id> /history`	GET	Untuk melihat riwayat *virtual machine*.

4.2.3.8 *End-Point* Melihat *Task*

Untuk melihat daftar *task*, pengguna dapat mengakses *end-point /api/v1/task* dengan *method GET*.

4.2.4 Implementasi Integrasi *HTTP Rest API* dengan Celery *Task Queque*

Pada sistem ini menggunakan *Task Queue* untuk mengoptimalkan performa. *Task Queue* dibangun menggunakan modul Python Celery yang terintegrasi dengan *HTTP Rest API* yang dibangun menggunakan Python Flask. Untuk melakukan instalasi Python Celery dapat dilihat pada Kode Sumber IV.1.

```
pip install celery
```

Kode Sumber IV.1: Perintah Instalasi Python Celery

Setelah melakukan instalasi Python Celery, langkah selanjutnya mengintegrasikan *object* Python Flask dengan Python Celery sebagai konfigurasi dasar. *Pseudocde* Untuk pengintegrasian dapat dilihat pada Kode Sumber IV.2.

```
FUNCTION make_celery(app)
  celery <- Celery(app.import_name,
    backend=app.config['result_backend'],
    broker=app.config['CELERY_BROKER_URL'])
  celery.conf.update(app.config)
  celery.conf.task_default_queue <- 'hypgen_queue
      '
  TaskBase <- celery.Task
  CLASS ContextTask(TaskBase):
    abstract <- True
    FUNCTION __call__(self, *args, **kwargs)
      with app.app_context()
        RETURN TaskBase__call__(self, *args, **
            kwargs)
    ENDFUNCTION
  ENDCLASS
  celery.Task <- ContextTask
  RETURN celer
ENDFUNCTION
```

Kode Sumber IV.2: *Pseudocode* Pengintegrasian Python Flask dan Python Celery

Setelah melakukan pengintegrasian, selanjutnya membuat *file Tasks.py* yang berisi daftar pekerjaan yang dikerjakan menggunakan Python Celery. *Pseudocode file Tasks.py* dapat dilihat pada Kode Sumber IV.3.

```
celery_worker <- make_celery(app)

@celery_worker.task(bind <- True)
FUNCTION create_vm(self, params <- {}):
  return status, message, params
ENDFUNCTION
```

Kode Sumber IV.3: *Pseudocode File Tasks.py*

Variabel *celery_worker* merupakan variabel hasil pengintegrasian Python Flask dengan Python Celery. Variabel tersebut dijadikan fungsi *decorator* agar fungsi dibawahnya dapat dikerjakan dengan Python Celery. Untuk cara pemanggilan fungsi dapat dilihat pada Kode Sumber IV.4.

```
FUNCTION create_vm()
  params <- getUserParams()
  Tasks.create_vm.delay(params)
ENDFUNCTION
```

Kode Sumber IV.4: Pseudocode Membuat *Virtual Machine* pada *File VM_Controller.py*

Untuk memanggil fungsi pada file *Tasks.py*, pada *controller* ditambahkan *method delay* agar saat pemanggilan fungsi tersebut dimasukkan ke dalam *Task Queue* dan *task* bersifat asinkronus. Python Celery berjalan diatas proses tersendiri. Untuk menjalankan Python Celery dengan menjalankan perintah yang dapat dilihat pada Kode Sumber IV.5.

```
celery worker –E ––app=app.Library.Tasks.
    celery_worker –Q hypgen_queue ––loglevel
    =DEBUG
```

Kode Sumber IV.5: Perintah Untuk Menjalankan Python Celery

4.2.5 Implementasi Manajemen *Virtual Machine*

Melalui *middleware* pengguna dapat membuat *virtual machine* baru, menyalakan *virtual machine*, mematikan *virtual machine*, meng-*resize resource virtual machine* dan menghapus *virtual machine* tanpa berinteraksi secara langsung dengan *hypervisor*. *Middleware* akan menerjemahkan permintaan pengguna dalam bentuk *parameter-parameter* yang dibutuhkan untuk terhubung ke *hypervisor* secara otomatis.

4.2.5.1 Implementasi Alokasi dan *Provisioning Virtual Machine* Baru

Untuk membuat *virtual machine*, pengguna mengirimkan *parameter* berupa nama *virtual machine*, jenis sistem operasi dan kategori *resource* melalui *end-point HTTP Rest API* yang sudah disediakan. *HTTP Rest API* akan menyimpan data *virtual machine* baru pengguna ke dalam basis data. Selain menyimpan data *virtual machine* baru, *HTTP Rest API* akan membuat *task* baru. *Pseudocode* dapat dilihat pada Kode Sumber IV.6.

```
1  FUNCTION create():
2    vm <- VM()
3    TRY
4      vm.save()
5    CATCH
6      RETURN "Error 500"
7    ENDTRY
8    task <- TaskModel()
9    TRY
10     task.save()
11   CATCH
12     RETURN "Error 500"
13   ENDTRY
14   params <- {"name":
         getNameVirtualMachineFromUser(),"last_step":
          1 , "os":getOsFromUser(), "request_category
         ":getResourceFromUser(), "vm": vm, "owner_vm
         ": getUserUniqueID(), "random": timestamp, "
         task": task}
15   Tasks.create_vm.delay(params)
16   RETURN {"status":200, "data": "Proses pembuatan
          virtual machine baru akan dimasukkan dalam
         antrian. Mohon tunggu sampai proses selesai"
         }
17 ENDFUNCTION
```

Kode Sumber IV.6: *Pseudocode* Alokasi *Virtual Machine* Baru pada *File VM_Controller.py*

Setelah menyimpan data *virtual machine* dan membuat *task* baru, selanjutnya *middleware* akan memanggil fungsi *create_vm* pada *file Tasks.py* melalui fungsi *delay* agar *task* tersebut disimpan pada *queue* dan berjalan secara asinkronus. Pada fungsi *create_vm*, *middleware* akan memanggil fungsi *create_vm* pada

class Hypervisor_Library. *Class Hypervisor_Library* bertugas untuk menghubungkan *middleware* dengan *hypervisor*.

Proses selanjutnya terjadi dalam fungsi *create_vm* pada *class Hypervisor_Library*, *middleware* akan melakukan *query* untuk mencari *server* yang tersedia berdasarkan sistem operasi. Sistem operasi memliki *template* disetiap *hypervisor* dan setiap *hypervisor* memiliki relasi ke *server*. Selanjutnya *middleware* akan melakukan *query* untuk mendapatkan data kategori *resource* untuk *resource virtual machine*. Setelah mendapatkan *server* yang tersedia dan data kategori *resource*, *middleware* memanggil fungsi *get_selected_host* untuk mendapatkan *server* terbaik dengan menggunakan algoritma AHP. Setelah mendapatkan data *server* terbaik, *middleware* akan menjalankan perintah *virtual machine* berdasarkan *hypervisor* dari *server* terbaik. Pseudocode fungsi *create_vm* pada *class Hypervisor_Library* dapat dilihat pada Kode Sumber IV.7.

```
1  FUNCTION create_vm(self, params <- {})
2    hosts <-  get_host_by_os(params["os"])
3    resource <-  get_data_request_category(params["
         request_category"])
4    selected_host <-  get_selected_host(hosts,
         params)
5
6    IF selected_host.hypervisor = "proxmox"
7    THEN
8      status, message, params <- Proxmox_Library().
           create_vm(params, selected_host)
9    ELSE IF selected_host.hypervisor = "vmware"
10   THEN
11     status, message, params <- Vsphere_Library().
           create_vm(params, selected_host)
12   ENDIF
13   RETURN status, message, params
14 ENDFUNCTION
```

Kode Sumber IV.7: *Pseudocode* Fungsi *create_vm* pada *class Hypervisor_Library*

Setelah *virtual machine* berhasil dibuat, langkah selanjutnya adalah mengatur IP pada *virtual machine*. Sistem hanya mampu mengatur IP pada sistem operasi Ubuntu dan Debian, dikarenakan berbeda sistem operasi berbeda pula cara mengatur IP. Untuk pengaturan *IP* pada *hypervisor* Proxmox, sistem akan melakukan pencarian IP berdasarkan *MAC Address virtual machine* dan *network identifier hypervisor*. Pseudocode fungsi mendapatkan IP *virtual machine* dapat dilihat pada Kode Sumber IV.8.

```
FUNCTION get_ip(self, mac_address, nid_hypervisor
    ):
host <- Selected_Host_Parser( selected_host).
    parse()
ssh <- Connect_SSH(host["ip"], host["username"],
    host["password"]).connect()
command <- "arp-scan -q -l --interface "+ params[
    "bridge_hypervisor"]+" = grep "+mac_address.
    lower()+" = awk '{print $1}'"
TRY:
  ssh_stdin, ssh_stdout, ssh_stderr <- ssh.
      exec_command(command, get_pty=True, timeout
      = 40)
  vm_ip <- ssh_stdout.readlines()
  IF len(vm_ip) = 0:
    raise Exception("Cannot find vm's ip")
  ENDIF
CATCH Exception as e:
  ssh.close()
  RETURN False, e, None
ENDTRY
ssh.close()
RETURN True, None, vm_ip

ENDFUNCTION
```

Kode Sumber IV.8: *Pseudocode* Fungsi *get_ip*

Setelah mendapatkan IP dari *virtual machine*, sistem akan mengatur IP *virtual machine* berdasarkan sistem operasinya melalui protokol SSH dengan mengubah *file* konfigurasi internet pada *virtual machine*. Pada *hypervisor* proxmox, ketika melakukan pengaturan IP pada sistem operasi Ubuntu dan Debian, sama-sama melakukan perintah dengan bantuan SSH ke

virtual machine.

Sedangkan pada *hypervisor* Vmware Vsphere, pengaturan IP pada sistem operasi Ubuntu dan Debian memiliki perbedaan mekanisme. Pada sistem operasi Ubuntu, Vmware Vpshere menyediakan *vmtools* yang digunakan untuk mempermudah pengaturan *virtual machine* tanpa harus melakukan *remote*. Pengaturan IP akan dilakukan pada saat proses *restore template virtual machine* sedang berjalan. Pseudocode proses *restore template* sistem operasi Ubuntu dapat dilihat pada Kode Sumber IV.9.

```
1 adaptermap = vim.vm.customization.AdapterMapping
    ()
2 adaptermap.adapter = vim.vm.customization.
    IPSettings()
3 adaptermap.adapter.ip = vim.vm.customization.
    FixedIp()
4 adaptermap.adapter.ip.ipAddress = self.params['ip
    ']
5 adaptermap.adapter.subnetMask = self.params['
    netmask']
6 adaptermap.adapter.gateway = self.params['netmask
    ']
7 globalip = vim.vm.customization.GlobalIPSettings(
    dnsServerList=['202.46.129.2', '202.46.129.3']
    )
```

Kode Sumber IV.9: *Pseudocode* Proses *Restore Template* Sistem Operasi Ubuntu pada Vmware Vsphere

Berbeda dengan mekanisme pengaturan IP pada sistem operasi Ubuntu, pengaturan IP pada sistem operasi Debian pada *hypervisor* Vmware Vsphere dilakukan pada saat proses *restore template* sistem operasi selesai dilakukan. Proses pengaturan IP dilakukan dengan cara melakukan perubahan file konfigurasi

internet dengan cara me-*remote virtual machine* dengan bantuan Pyvmomi. Pseudocode dapat dilihat pada Kode Sumber IV.10.

```
1  creds <- vim.vm.guest.NamePasswordAuthentication
       (
2  username= params['template_username'], password=
       AESHelper(AppConfig.default_secret_key).
       decrypt( params['template_password'])
3  ENDFUNCTION
4
5  pm <- content.guestOperationsManager.
       processManager
6  ps <- vim.vm.guest.ProcessManager.ProgramSpec(
7  programPath="/bin/sed",
8  arguments="-i 's/address 10.199.14.155/address "+
        params['ip']+"/g; s/gateway 10.199.14.1/
       gateway "+ params['gateway']+"/g; s/netmask 25
       5.255.255.0/netmask "+ params['netmask']+"/g'
       /etc/network/interfaces"
9  )
10 res <- pm.StartProgramInGuest(vm, creds, ps)
```

Kode Sumber IV.10: *Pseudocode* Proses Pengaturan IP pada Sistem Operasi Debian *Hypervisor* Vmware Vsphere

4.2.6 Implementasi Mematikan *Virtual Machine*

Untuk mematikan *virtual machine*, pengguna hanya perlu mengakses *end-point*. Pada *end-point* terdapat *path* yang akan diuraikan oleh *HTTP Rest API* sebagai *vm_id*. Selanjutnya *middleware* akan melakukan *query* untuk mendapatkan data-data mengenai *virtual machine* termasuk memvalidasi kepemilikan *virtual machine*.

Setelah mendapatkan data mengenai *virtual machine*, selanjutnya *middleware* akan membuat *task* baru. Kemudian data

virtual machine dan data *task* dijadikan parameter pemanggilan fungsi *stop_vm* pada *file Tasks.py* oleh *middleware* dengan menambah fungsi *delay* agar *task* tersebut disimpan pada *queue* dan berjalan secara asinkronus. Pada fungsi *stop_vm*, *middleware* akan memanggil fungsi *stop_vm* pada *class Hypervisor_Library*. Pseudocode dapat dilihat pada Kode Sumber IV.11.

```
1  FUNCTION stop(id)
2    vm <- VM.select().where(VM.id = id).first()
3    task <- TaskModel()
4    TRY
5      task.save()
6    CATCH
7      RETURN "Error 500"
8    ENDTRY
9    params <- {'task': task, 'vm': vm}
10   Tasks.stop_vm.delay(params)
11   RETURN {"status":200, "data": "Proses mematikan
         akan dimasukkan dalam antrian. Mohon tunggu
         sampai proses selesai"}
12 ENDFUNCTION
```

Kode Sumber IV.11: *Pseudocode* Mematikan *Virtual Machine* pada *VM_Controller*

Proses selanjutnya terjadi dalam fungsi *stop_vm* pada *class Hypervisor_Library*, *middleware* akan memeriksa jenis *hypervisor* pada data *virtual machine* dengan *parameter* untuk menentukan mekanisme mematikan *virtual machine*. Pseudocode fungsi *stop_vm* dapat dilihat pada Kode Sumber IV.12.

```
1  FUNCTION stop_vm(self, params <- {})
2    vm = params['vm']
3    IF vm.hypervisor = "proxmox"
4    THEN
5      status, message, params <- Proxmox_Library().
           stop_vm(params, params['vm']['hosts'])
6    ELSE IF vm.hypervisor = "vmware"
7    THEN
8      status, message, params <- Vsphere_Library().
           stop_vm(params, vm.hosts)
9    ENDIF
10   RETURN status, message, params
11 ENDFUNCTION
```

Kode Sumber IV.12: *Pseudocode* Fungsi *stop_vm* pada *class Hypervisor_Library*

4.2.7 Implementasi Menyalakan *Virtual Machine*

Untuk menyalakan *virtual machine*, pengguna hanya perlu mengakses *end-point*. Pada *end-point* terdapat *path* yang akan diuraikan oleh *HTTP Rest API* sebagai *vm_id*. Selanjutnya *middleware* akan melakukan *query* untuk mendapatkan data-data mengenai *virtual machine* termasuk memvalidasi kepemilikan *virtual machine*.

Setelah mendapatkan data mengenai *virtual machine*, selanjutnya *middleware* akan membuat *task* baru. Kemudian data *virtual machine* dan data *task* dijadikan parameter pemanggilan fungsi *start_vm* pada *file Tasks.py* oleh *middleware* dengan menambah fungsi *delay* agar *task* tersebut disimpan pada *queue* dan berjalan secara asinkronus. Pada fungsi *start_vm*, *middleware* akan memanggil fungsi *start_vm* pada *class Hypervisor_Library*. Pseudocode dapat dilihat pada Kode

Sumber IV.13.

```
1  FUNCTION start(id)
2    vm <- VM.select().where(VM.id = id).first()
3    task <- TaskModel()
4    TRY
5      task.save()
6    CATCH
7      RETURN "Error 500"
8    ENDTRY
9    params <- {'task': task, 'vm': vm}
10   Tasks.start_vm.delay(params)
11   RETURN {"status":200, "data": "Proses
         menyalakan akan dimasukkan dalam antrian.
         Mohon tunggu sampai proses selesai"}
12 ENDFUNCTION
```

Kode Sumber IV.13: *Pseudocode* Menyalakan *Virtual Machine* pada *VM_Controller*

Proses selanjutnya terjadi dalam fungsi *start_vm* pada *class Hypervisor_Library*, *middleware* akan memeriksa jenis *hypervisor* pada data *virtual machine* dengan *parameter* untuk menentukan mekanisme menyalakan *virtual machine*. Pseudocode fungsi *start_vm* dapat dilihat pada Kode Sumber IV.14.

```
1  FUNCTION start_vm(self, params <- {})
2    vm = params['vm']
3
4    IF vm.hypervisor = "proxmox"
5    THEN
6      status, message, params <- Proxmox_Library().
           start_vm(params, params['vm']['hosts'])
7    ELSE IF vm.hypervisor = "vmware"
8    THEN
9      status, message, params <- Vsphere_Library().
           start_vm(params, vm.hosts)
10   ENDIF
11
12   RETURN status, message, params
13 ENDFUNCTION
```

Kode Sumber IV.14: *Pseudocode* Fungsi *start_vm* pada *class Hypervisor_Library*

4.2.8 Implementasi Menghapus *Virtual Machine*

Untuk menghapus *virtual machine*, pengguna hanya perlu mengakses *end-point*. Pada *end-point* terdapat *path* yang akan diuraikan oleh *HTTP Rest API* sebagai *vm_id*. Selanjutnya *middleware* akan melakukan *query* untuk mendapatkan data-data mengenai *virtual machine* termasuk memvalidasi kepemilikan *virtual machine*.

Setelah mendapatkan data mengenai *virtual machine*, selanjutnya *middleware* akan membuat *task* baru. Kemudian data *virtual machine* dan data *task* dijadikan parameter pemanggilan fungsi *delete_vm* pada *file Tasks.py* oleh *middleware* dengan menambah fungsi *delay* agar *task* tersebut disimpan pada *queue* dan berjalan secara asinkronus. Pada fungsi *delete_vm*,

middleware akan memanggil fungsi *delete_vm* pada *class Hypervisor_Library*. Pseudocode dapat dilihat pada Kode Sumber IV.15.

```
1  FUNCTION delete_vm(id)
2    vm <- VM.select().where(VM.id = id).first()
3    task <- TaskModel()
4    TRY
5      task.save()
6    CATCH
7      RETURN "Error 500"
8    ENDTRY
9    params <- {'task': task, 'vm': vm}
10   Tasks.delete_vm.delay(params)
11 RETURN {"status":200, "data": "Proses
     pengahapusan virtual machine akan dimasukkan
     dalam antrian. Mohon tunggu sampai proses
     selesai"}
12 ENDFUNCTION
```

Kode Sumber IV.15: *Pseudocode* Menghapus *Virtual Machine* pada *VM_Controller*

Proses selanjutnya terjadi dalam fungsi *delete_vm* pada *class Hypervisor_Library*, *middleware* akan memeriksa jenis *hypervisor* pada data *virtual machine* dengan *parameter* untuk menentukan mekanisme menghapus *virtual machine*. Pseudocode fungsi *delete_vm* dapat dilihat pada Kode Sumber IV.16.

```
FUNCTION delete_vm(self, params <- {})
    vm = params['vm']
    IF vm.hypervisor = "proxmox"
    THEN
      status, message, params <- Proxmox_Library
          ().delete_vm(params, params['vm']['hosts
          '])
    ELSE IF vm.hypervisor = "vmware"
    THEN
      status, message, params <- Vsphere_Library
          ().delete_vm(params, vm.hosts)
    ENDIF
    RETURN status, message, params
ENDFUNCTION
```

Kode Sumber IV.16: *Pseudocode* Fungsi *delete_vm* pada *class Hypervisor_Library*

4.2.9 Implementasi *Resize Resource Virtual Machine*

Untuk melakukan *resize resource virtual machine*, pengguna mengakses *end-point* dengan mengirimkan parameter *request_category_id* sebagai kategori *resource* yang akan diatur pada *virtual machine*. Pada *end-point* terdapat *path* yang akan diuraikan oleh *HTTP Rest API* sebagai *vm_id*. Selanjutnya *middleware* akan melakukan *query* untuk mendapatkan data-data mengenai *virtual machine*, memvalidasi kepemilikan *virtual machine*, dan melakukan *query* untuk mendapatkan data kategori *resource* yang dikirim oleh pengguna.

Setelah mendapatkan data mengenai *virtual machine*, selanjutnya *middleware* akan membandingkan kategori *resource* yang dikirim oleh pengguna dengan kategori *resource* yang sudah diimplementasikan pada *virtual machine*. Apabila sama,

middleware akan mengirim umpan balik berupa kode *HTTP* 400. Setelah itu *middleware* akan membuat *task* baru. Kemudian data *virtual machine* dan data *task* dijadikan parameter pemanggilan fungsi *resize_vm* pada *file Tasks.py* oleh *middleware* dengan menambah fungsi *delay* agar *task* tersebut disimpan pada *queue* dan berjalan secara asinkronus. Pada fungsi *resize_vm*, *middleware* akan memanggil fungsi *resize_vm* pada *class Hypervisor_Library*. Pseudocode dapat dilihat pada Kode Sumber IV.17.

```
1  FUNCTION vm_resize(id):
2    vm <- VM.select().where(VM.id = id).first()
3    request_category <- Request_Category.select().
       where(Request_Category.id =
       getUserRequestParamter('request_category_id
       ')).first()
4    task <- TaskModel()
5    TRY
6      task.save()
7    CATCH Exception as e:
8      RETURN 'Error 500'
9    ENDTRY
10   params <- {"vm" : vm, "task" : task, "
       request_category": request_category}
11   Tasks.resize_vm.delay(params)
12   RETURN {"status":200, "data": "Proses resize vm
       akan dimasukkan dalam antrian. Mohon tunggu
       sampai proses selesai"}
13 ENDFUNCTION
```

Kode Sumber IV.17: *Pseudocode Resize Resource Virtual Machine* pada *VM_Controller*

Proses selanjutnya terjadi dalam fungsi *resize_vm* pada *class Hypervisor_Library*, *middleware* akan memeriksa jenis

hypervisor pada data *virtual machine* dengan *parameter* untuk menentukan mekanisme *resize resource virtual machine*. Pseudocode fungsi *resize_vm* dapat dilihat pada Kode Sumber IV.18.

```
1  FUNCTION resize_vm(self, params <- {})
2  vm = params['vm']
3
4  IF vm.hypervisor = "proxmox"
5  THEN
6  status, message, params <- Proxmox_Library().
       resize_vm(params, params['vm']['hosts'])
7  ELSE IF vm.hypervisor = "vmware"
8  THEN
9  status, message, params <- Vsphere_Library().
       resize_vm(params, vm.hosts)
10 ENDIF
11
12 RETURN status, message, params
13 ENDFUNCTION
```

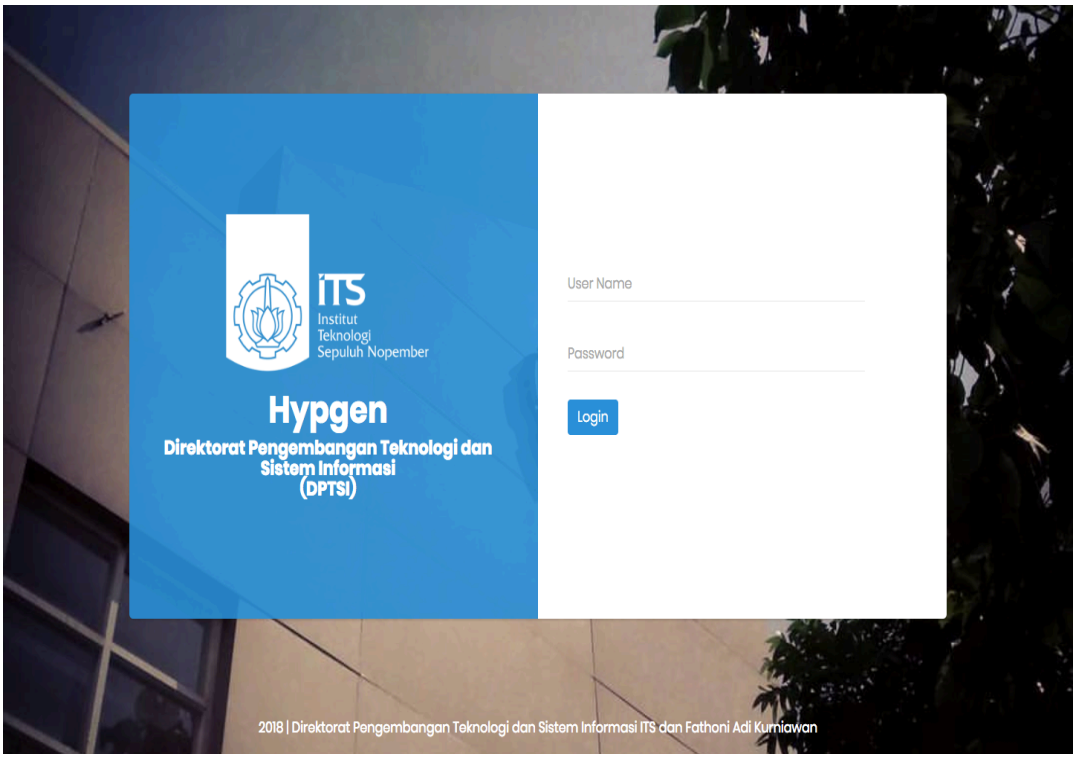
Kode Sumber IV.18: *Pseudocode* Fungsi *resize_vm* pada *class Hypervisor_Library*

4.3 Implementasi *Interface* Web

Untuk mengakses sistem, terdapat dua buah *interface* yang bisa digunakan yaitu *interface* web dan *command line interface*. *Interface* web dapat diakses melalui *port 9090*.

4.3.1 Implementasi Autentifikasi dan Otorisasi pada *Interface* Web

Untuk melakukan autentifikasi dan otorisasi pada *interface* web, pengguna mengisi *form username* dan *password* pada halaman *login*. Data *username* dan *password* dikirim ke *end-point middleware*. *Middleware* akan mengirim umpan balik berupa *token* yang digunakan untuk autentifikasi dan otorisasi. Selanjutnya *token* akan disimpan pada *cookie*.Tampilan halaman *login* dapat dilihat pada Gambar 4.2



Gambar 4.2: Implementasi Halaman Login pada *Interface* Web

Token akan dikirim oleh *interface* web pada *header* permintaan dari pengguna saat pengguna mengakses halaman pada *interface* web.

4.3.2 Implementasi *End-point* pada *Interface* Web

Untuk mengakses *interface* web, penulis menyediakan *end-point* yang dapat digunakan untuk mengatur sistem maupun untuk melakukan manajemen pada *virtual machine*. Untuk detail *End-point* dapat dilihat di Tabel 4.20.

Tabel 4.20: Tabel *End-point* pada *Interface* Web

No	*End-point*	*Method*	Keterangan
1	`/`	GET	Merupakan halaman *root*, ketika pengguna mengakses *end-point* maka akan dialihkan ke `/login`.
2	`/dashboard`	GET	Merupakan halaman dasbor.
3	`/host`	GET	Halaman untuk melihat *host* yang tersedia.
4	`/host/create`	POST	*End-point* untuk memproses *form* data *host* baru dan pada *end-point* ini, *interface* web akan mengirimkan data tersebut ke *middleware*.
5	`/host/create`	GET	Halaman mengisi *form* data *host* baru.

Tabel 4.20: Tabel *End-point* pada *Interface* Web

No	***End-point***	***Method***	**Keterangan**
6	`/host/<hypervisor_id>/<host_id>`	DELETE	*End-point* untuk menghapus *host* terpilih. *Interface* web akan mengirimkan *vm_id* dan *hypervisor_id* dari hasil penguraian *end-point* ke *middleware*.
7	`/host/<hypervisor_id>/<host_id>`	PUT	*End-point* untuk memproses perubahan data *host* terpilih. *Interface* web akan mengirimkan data ke *middleware*.
8	`/host/<hypervisor_id>/<host_id>/edit`	GET	Halaman untuk mengisi *form* perubahan data *host*.
9	`/login`	GET	Halaman login.
10	`/login`	POST	*End-point* untuk memproses permintaan *login* pengguna, Pada *end-point* ini, *interface* web akan mengirimkan data *username* dan *password* ke *middleware* untuk autentifikasi dan otorisasi.
11	`/logout`	GET	*End-point* untuk keluat dari sistem.

Tabel 4.20: Tabel *End-point* pada *Interface* Web

No	***End-point***	***Method***	**Keterangan**
12	`/os`	POST	*End-point* untuk memproses *form* data versi sistem operasi baru dan pada *end-point* ini, *interface* web akan mengirimkan data tersebut ke *middleware*.
13	`/os`	GET	Halaman untuk melihat versi sistem operasi yang tersedia.
14	`/os/create`	GET	Halaman mengisi *form* data versi sistem operasi baru.
15	`/os/<os_id>`	DELETE	*End-point* untuk menghapus versi sistem operasi terpilih. *Interface* web akan mengirimkan *os_id* dari hasil penguraian *end-point* ke *middleware*.
16	`/os/<os_id>`	PUT	*End-point* untuk memproses perubahan data versi sistem operasi terpilih. *Interface* web akan mengirimkan data ke *middleware*.
17	`/os/<os_id>/edit`	GET	Halaman untuk mengisi *form* perubahan data versi sistem operasi.

Tabel 4.20: Tabel *End-point* pada *Interface* Web

No	***End-point***	***Method***	**Keterangan**
18	`/resource`	GET	Halaman untuk melihat kategori *resource* yang tersedia.
19	`/resource`	POST	*End-point* untuk memproses data kategori *resource* baru dan pada *end-point* ini, *interface* web akan mengirimkan data tersebut ke *middleware*.
20	`/resource/create`	GET	Halaman mengisi *form* data kategori *resource* baru.
21	`/resource/<resource_id>`	DELETE	*End-point* untuk menghapus kategori *resource* terpilih. *Interface* web akan mengirimkan *resource_id* dari hasil penguraian *end-point* ke *middleware*.
22	`/resource/<resource_id>`	PUT	*End-point* untuk memproses perubahan data kategori *resource* terpilih. *Interface* web akan mengirimkan data ke *middleware*.
23	`/resource/<resource_id>/edit`	GET	Halaman untuk mengisi kategori *resource* perubahan data *host*.

Tabel 4.20: Tabel *End-point* pada *Interface* Web

No	***End-point***	***Method***	**Keterangan**
24	`/task`	GET	Halaman untuk melihat riwayat *task*.
25	`/template`	GET	Halaman untuk melihat *template* sistem operasi yang tersedia.
26	`/template/create`	GET	Halaman mengisi *form* data *template* sistem operas barui.
27	`/template/create`	POST	*End-point* untuk memproses data versi sistem operasi baru dan pada *end-point* ini, *interface* web akan mengirimkan data tersebut ke *middleware*.
28	`/template/<hypervisor_id>/<template_id>`	PUT	*End-point* untuk memproses perubahan data *template* sistem operasi terpilih. *Interface* web akan mengirimkan data ke *middleware*.
28	`/template/<hypervisor_id>/<template_id>`	DELETE	*End-point* untuk menghapus *template* sistem operasi terpilih. *Interface* web akan mengirimkan *template_id* dan *hypervisor_id* dari hasil penguraian *end-point* ke *middleware*.

Tabel 4.20: Tabel *End-point* pada *Interface* Web

No	*End-point*	*Method*	Keterangan
29	`/template/ <hypervisor_ id> /<template_ id>/edit`	GET	Halaman untuk mengisi *form* perubahan data *template* sistem operasi.
30	`/token`	POST	*End-point* untuk memproses data *API Secret Key* baru dan pada *end-point* ini, *interface* web akan mengirimkan data tersebut ke *middleware*.
31	`/token`	GET	Halaman untuk melihat *API Secret Key* yang tersedia.
32	`/token/create`	GET	Halaman mengisi *form* data *API Secret Key* versi sistem operasi baru.
33	`/token/ <token_id>`	DELETE	*End-point* untuk menghapus *API Secret Key* terpilih. *Interface* web akan mengirimkan *token_id* dari hasil penguraian *end-point* ke *middleware*.

Tabel 4.20: Tabel *End-point* pada *Interface* Web

No	***End-point***	***Method***	**Keterangan**
34	`/token/ <token_id>`	PUT	*End-point* untuk memproses perubahan data *API Secret Key* terpilih. *Interface* web akan mengirimkan data ke *middleware*.
35	`/token/ <token_id> /edit`	GET	Halaman untuk mengisi *form* perubahan data *API Secret Key*.
36	`/user`	GET	Halaman untuk melihat pengguna yang terdaftar pada sistem.
37	`/user`	POST	*End-point* untuk memproses data pengguna baru dan pada *end-point* ini, *interface* web akan mengirimkan data tersebut ke *middleware*.
38	`/user/create`	GET	Halaman mendaftarkan pengguna pada sistem.
39	`/user/<user_ id>`	PUT	*End-point* untuk memproses perubahan data pengguna terpilih. *Interface* web akan mengirimkan data ke *middleware*.

Tabel 4.20: Tabel *End-point* pada *Interface* Web

No	***End-point***	***Method***	**Keterangan**
40	`/user/<user_id>`	DELETE	*End-point* untuk menghapus pengguna terpilih. *Interface* web akan mengirimkan *user_id* dari hasil penguraian *end-point* ke *middleware*.
41	`/user/<user_id>/edit`	GET	Halaman untuk mengisi *form* perubahan data pengguna.
42	`/vm`	POST	*End-point* untuk memproses data *virtual machine* baru dan pada *end-point* ini, *interface* web akan mengirimkan data tersebut ke *middleware*.
43	`/vm/create`	GET	Halaman untuk membuat *virtual machine* baru.
44	`/vm/<vm_id>/destory`	GET	Halaman konfirmasi penghapusan *virtual machine*.
45	`/vm/<vm_id>/history`	GET	Halaman untuk melihat riwayat *task virtual machine*.
46	`/vm/<vm_id>/owner`	GET	Halaman melihat pemilik *virtual machine*.

Tabel 4.20: Tabel *End-point* pada *Interface* Web

No	***End-point***	***Method***	**Keterangan**
47	`/vm/<vm_id> /owner`	POST	*End-point* untuk memproses data pemilik *virtual machine* baru dan pada *end-point* ini, *interface* web akan mengirimkan data tersebut ke *middleware*.
48	`/vm/<vm_id> /owner/revoke`	POST	*End-point* untuk memproses penghapusan pemilik *virtual machine* dan pada *end-point* ini, *interface* web akan mengirimkan data tersebut ke *middleware*.
49	`/vm/<vm_id> /owner/revoke`	GET	Halaman untuk memilih pengguna yang akan dihapus dari kepemilikan *virtual machine*.
50	`/vm/<vm_id> /resize`	GET	Halaman untuk memilih perubahan kategori *resource* pada *virtual machine*.
51	`/vm/<vm_id> /resize`	POST	*End-point* untuk memproses perubahan *resource virtual machine* dan pada *end-point* ini, *interface* web akan mengirimkan data tersebut ke *middleware*.

Tabel 4.20: Tabel *End-point* pada *Interface* Web

No	***End-point***	***Method***	**Keterangan**
52	`/vm/<vm_id> /start`	POST	*End-point* untuk menyalakan *virtual machine*.
53	`/vm/<vm_id> /stop`	POST	*End-point* untuk mematikan *virtual machine*.
54	`/vm/<vm_id>`	GET	Halaman untuk melihat data *virtual machine*.
55	`/vm/<vm_id>`	DELETE	*End-point* untuk menghapus *virtual machine* terpilih. *Interface* web akan mengirimkan *vm_id* dari hasil penguraian *end-point* ke *middleware*.

4.4 Implementasi *Command Line Interface*

Selain dapat diakses melalui *interface* web, sistem juga dapat diakses mennggunakan *Command Line Interface*. *Command Line Interface* dibangun menggunakan bahasa pemrograman Python.

4.4.1 Implementasi Autentifikasi dan Otorisasi pada *Command Line Interface*

Untuk mengakses sistem melalui *Command Line Interface*, pengguna harus membuat *API Secret Key* pada *interface* web. *API Secret Key* barus diatur terlebih dahulu pada aplikasi. Ketika pengguna menjalankan aplikasi dengan perintah tertentu, *API Secret Key* akan dikirim ke *middleware* terlebih dahulu untuk mendapatkan *token*. *API Secret Key* juga membatasi hak akses

dari pengguna. Terdapat dua kategori yaitu *write* dan *read only*. Untuk mengatur *API Secret Key* dapat dilihat pada Kode Sumber IV.19.

```
hypgen auth <API Secret Key>
```

Kode Sumber IV.19: Perintah Untuk Mengatur *API Secret Key*

4.4.2 Implementasi Manajemen *Virtual Machine* pada *Command Line Interface*

Pada aplikasi *Command Line Interface* terdapat *parameter* untuk melakukan manajemen pada sistem. *Parameter* yang tersedia dapat dilihat pada Tabel 4.21.

Tabel 4.21: Tabel *Parameter* pada *Command Line Interface*

No	***Parameter***	**Keterangan**
1	*hypgen ps*	Untuk melihat status *virtual machine*.
2	*hypgen config*	Untuk mengatur dan melihat konfigurasi *Command Line Interface*. Pada *parameter* ini, pengguna dapat mengatur alamat *HTTP Rest API*
3	*hypgen show resource*	Untuk melihat kategori *resource* yang tersedia.
4	*hypgen show resource*	Untuk melihat kategori *resource* yang tersedia.
5	*hypgen vm add*	Untuk membuat *virtual machine* baru.
6	*hypgen vm rm <vm_id>*	Untuk menghapus *virtual machine* tertentu.

Tabel 4.21: Tabel *Parameter* pada *Command Line Interface*

No	***Parameter***	**Keterangan**
7	*hypgen vm start <vm_id>*	Untuk menyalakan *virtual machine* tertentu.
8	*hypgen vm stop <vm_id>*	Untuk mematikan *virtual machine* tertentu.

(Halaman ini sengaja dikosongkan)

BAB V

PENGUJIAN DAN EVALUASI

5.1 Lingkungan Uji Coba

Lingkungan pengujian menggunakan komponen-komponen yang terdiri dari: satu server *middleware*, dua server *proxmox*, satu server *Vmware ESXI*, satu server menggunakan Windows Server 2016 sebagai *Vmware Vcenter* dan Komputer Penguji. Untuk melakukan pengujian performa, penulis membuat script Python untuk melakukan permintaan sejumlah skenario pengujian.

Spesifikasi untuk setiap komponen yang digunakan ditunjukkan pada Tabel 5.1.

Tabel 5.1: Spesifikasi Komponen

No	Komponen	Perangkat Keras	Perangkat Lunak
1	Middleware	4 core processor, 4GB RAM	Python 2.7
2	Proxmox A	4 core processor, 8GB RAM	*Hypervisor* Proxmox
3	Proxmox B	4 core processor, 8GB RAM	*Hypervisor* Proxmox
4	VMware ESXI A	4 core processor, 4GB RAM	*Hypervisor* Vmware ESXI
5	Windows Server	8 core processor, 16GB RAM	Windows Server 2016 dan *Vmware Vcenter for Windows*
6	Komputer penguji	4 core processor, 8GB RAM	Ubuntu, Insomnia, Python 2.7

5.2 Skenario Uji Coba

Uji coba akan dilakukan untuk mengetahui keberhasilan sistem yang telah dibangun. Skenario pengujian dibedakan menjadi 2 bagian, yaitu:

- **Uji Fungsionalitas**
  Pengujian ini didasrkan pada fungsionalitas yang disajikan sistem.
- **Uji Performa**
  Pengujian ini untuk menguji ketahanan sistem terhadap sejumlah permintaan ke aplikasi secara bersamaan sejumlah pengguna yang meminta *virtual machine* baru. Pengujian dilakukan dengan melakukan *benchmark* pada sistem.

5.2.1 Skenario Uji Coba Fungsionalitas

Uji fungsionalitas dibagi menjadi 4, yaitu uji mengelola sistem menggunakan *Rest Client* untuk mengakses *HTTP Rest API* secara langsung, mengelola sistem menggunakan *interface* web dan mengelola *virtual machine* menggunakan *Command Line Interface* dan distribusi alokasi *virtual machine*.

5.2.1.1 Uji Fungsionalitas Mengelola Sistem Menggunakan *Rest Client*

Pada *middleware* terdapat *HTTP Rest API* yang menjadi gerbang untuk mengakses sistem. Pengujian dilakukan dengan menggunakan aplikasi *Rest Client* dengan cara mengakses *(*end-point) serta mengirimkan *parameter* yang dibutuhkan pada *HTTP Rest API* secara langsung. Rancangan pengujian dan hasil yang diharapkan ditunjukkan pada Tabel 5.2.

Tabel 5.2: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil Harapan
1	Autentifikasi	Melakukan permintaan autentifikasi dan otorisasi dengan mengirimkan *username* dan *password*.	*HTTP Rest API* dapat mengirimkan umpan balik berupa *token*.
		Melakukan permintaan autentifikasi dan otorisasi dengan mengirimkan *HTTP Rest API*.	*HTTP Rest API* dapat mengirimkan umpan balik berupa *token*.
2	*Host*	Menambahkan server baru.	Data server baru dapat disimpan pada basis data sistem.
		Melihat daftar server yang tersedia.	Pengguna dapat melihat daftar server yang tersedia pada sistem.
		Melihat data server terpilih.	Pengguna dapat melihat detail data server yang dipilih.

Tabel 5.2: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil Harapan
		Memperbaharui data server terpilih.	Pengguna dapat melakukan perubahan data pada server yang dipilih.
		Menghapus data server terpilih.	Pengguna menghapus data server yang dipilih.
3	*OS*	Menambahkan versi sistem operasi yang didukung oleh sistem.	Data versi sistem operasi baru dapat disimpan pada basis data sistem.
		Melihat daftar versi sistem operasi yang tersedia	Pengguna dapat melihat daftar versi sistem operasi yang tersedia pada sistem.
		Melihat data versi sistem operasi terpilih.	Pengguna dapat melihat detail data versi sistem operasi yang dipilih.

Tabel 5.2: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil Harapan
		Memperbaharui data versi sistem operasi terpilih.	Pengguna dapat melakukan perubahan data pada versi sistem operasi yang dipilih.
		Menghapus data versi sistem operasi terpilih.	Pengguna menghapus data versi sistem operasi yang dipilih.
4	*Template*	Menambahkan *template* sistem operasi yang didukung oleh sistem.	Data *template* sistem operasi baru dapat disimpan pada basis data sistem.
		Melihat daftar *template* sistem operasi yang tersedia.	Pengguna dapat melihat daftar *template* sistem operasi yang tersedia pada sistem.
		Melihat data *template* sistem operasi terpilih.	Pengguna dapat melihat detail data *template* sistem operasi yang dipilih.

Tabel 5.2: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil Harapan
		Memperbaharui data *template* sistem operasi terpilih.	Pengguna dapat melakukan perubahan data pada *template* sistem operasi yang dipilih.
		Menghapus data *template* sistem operasi terpilih.	Pengguna menghapus data *template* sistem operasi yang dipilih.
5	Kategori *Resource*	Menambahkan kategori *resource* untuk pilihan *resource virtual machine*.	Data kategori *resource* baru dapat disimpan pada basis data sistem.
		Melihat daftar kategori *resource* yang tersedia.	Pengguna dapat melihat daftar kategori *resource* yang tersedia pada sistem.
		Melihat data kategori *resource* terpilih.	Pengguna dapat melihat detail kategori *resource* yang dipilih.

Tabel 5.2: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil Harapan
		Memperbaharui data kategori *resource* terpilih.	Pengguna dapat melakukan perubahan data pada kategori *resource* yang dipilih.
		Menghapus data kategori *resource* terpilih.	Pengguna menghapus data kategori *resource* yang dipilih.
6	*User*	Mendaftarkan pengguna pada sistem.	Data pengguna baru dapat disimpan pada basis data sistem.
		Melihat daftar pengguna yang terdaftar pada sistem.	Pengguna dapat melihat daftar pengguna yang terdaftar pada sistem.
		Melihat data pengguna terpilih.	Pengguna dapat melihat detail pengguna yang dipilih.
		Memperbaharui data pengguna terpilih.	Pengguna dapat melakukan perubahan data pada pengguna yang dipilih.

Tabel 5.2: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil Harapan
		Menghapus data pengguna terpilih.	Pengguna menghapus data pengguna yang dipilih.
7	*API Secret Key*	Membuat *API Secret Key* baru.	Data *API Secret Key* baru dapat disimpan pada basis data sistem.
		Melihat daftar *API Secret Key* yang tersedia pada pengguna tertentu.	Pengguna dapat melihat daftar *API Secret Key* yang tersedia.
		Melihat data *API Secret Key* terpilih.	Pengguna dapat melihat detail *API Secret Key* yang dipilih.
		Memperbaharui data *API Secret Key* terpilih.	Pengguna dapat melakukan perubahan data pada *API Secret Key* yang dipilih.
		Menghapus data *API Secret Key* terpilih.	Pengguna menghapus data *API Secret Key* yang dipilih.

Tabel 5.2: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil Harapan
8	*Virtual Machine*	Membuat *virtual machine* baru.	Data *virtual machine* baru dibuat dan data terkait *virtual machine* tersimpan pada basis data sistem.
		Melihat daftar *virtual machine* yang tersedia pada pengguna tertentu.	Pengguna dapat melihat daftar *virtual machine* yang tersedia.
		Melihat detail data *virtual machine* terpilih.	Pengguna dapat melihat detail data *virtual machine* yang dipilih.
		Mengubah kategori *resource virtual machine* terpilih.	Pengguna dapat melakukan perubahan data pada kategori *resource virtual machine* yang dipilih.
		Menghapus *virtual machine* terpilih.	Pengguna dapat menghapus *virtual machine* yang dipilih.

Tabel 5.2: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil Harapan
		Mematikan *virtual machine* terpilih.	Pengguna dapat mematikan *virtual machine* yang dipilih.
		Menyalakan *virtual machine* terpilih.	Pengguna dapat menyalakan *virtual machine* yang dipilih.
		Membagikan hak pengelolaan *virtual machine* kepada pengguna lain.	Pengguna dapat membagikan hak pengelolaan *virtual machine* kepada pengguna yang dipilih.
		Melihat daftar pengguna yang memiliki hak pengelolaan *virtual machine*.	Pengguna dapat melihat daftar pengguna yang memiliki hak pengelolaan *virtual machine*.
		Menghapus pengguna terpilih yang dari hak pengelolaan *virtual machine*.	Pengguna dapat menghapus pengguna terpilih dari hak pengelolaan *virtual machine*.

Tabel 5.2: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil Harapan
		Melihat riwayat *task* yang dilakukan pengguna terhadap *virtual machine*.	Pengguna dapat melihat daftar riwayat *task* yang dilakukan dirinya sendiri maupun pengguna lain terhadap *virtual machine*.
		Melihat status *virtual machine* terpilih.	Pengguna dapat melihat status *virtual machine* yang dipilih.
9	*Task*	Melihat daftar *task*	Pengguna dapat melihat daftar *task* yang dilakukan oleh pengguna yang sedang *login*

5.2.1.2 Uji Fungsionalitas Mengelola Sistem Menggunakan *Interface* Web

Pengujian fungsionalitas selanjutnya, penulis menguji fitur-fitur pengelolaan sistem melalui *interface* web. Rancangan pengujian dan hasil yang diharapkan ditunjukkan pada Tabel 5.3.

Tabel 5.3: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil Harapan
1	Autentifikasi	Melakukan permintaan autentifikasi dan otorisasi dengan mengirimkan *username* dan *password*.	Pengguna dapat masuk pada sistem dan melihat halaman dasbor.
2	*Host*	Menambahkan server baru.	Data server baru dapat diteruskan oleh *interface* web ke middleware dan disimpan pada basis data sistem.
		Melihat daftar server yang tersedia.	Pengguna dapat melihat daftar server yang tersedia pada sistem.
		Memperbaharui data server terpilih.	Pengguna dapat melakukan perubahan data pada server yang dipilih.
		Menghapus data server terpilih.	Pengguna dapat menghapus data server yang dipilih.

Tabel 5.3: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil Harapan
3	*OS*	Menambahkan versi sistem operasi yang didukung oleh sistem.	Data versi sistem operasi baru dapat disimpan pada basis data sistem.
		Melihat daftar versi sistem operasi yang tersedia	Pengguna dapat melihat daftar versi sistem operasi yang tersedia pada sistem.
		Memperbaharui data versi sistem operasi terpilih.	Pengguna dapat melakukan perubahan data pada versi sistem operasi yang dipilih.
		Menghapus data versi sistem operasi terpilih.	Pengguna dapat menghapus data versi sistem operasi yang dipilih.
4	*Template*	Menambahkan *template* sistem operasi yang didukung oleh sistem.	Data *template* sistem operasi baru dapat disimpan pada basis data sistem.

Tabel 5.3: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil Harapan
		Melihat daftar *template* sistem operasi yang tersedia.	Pengguna dapat melihat daftar *template* sistem operasi yang tersedia pada sistem.
		Melihat data *template* sistem operasi terpilih.	Pengguna dapat melihat detail data *template* sistem operasi yang dipilih.
		Memperbaharui data *template* sistem operasi terpilih.	Pengguna dapat melakukan perubahan data pada *template* sistem operasi yang dipilih.
		Menghapus data *template* sistem operasi terpilih.	Pengguna dapat menghapus data *template* sistem operasi yang dipilih.
5	Kategori *Resource*	Menambahkan kategori *resource* untuk pilihan *resource virtual machine*.	Data kategori *resource* baru dapat disimpan pada basis data sistem.

Tabel 5.3: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil Harapan
		Melihat daftar kategori *resource* yang tersedia.	Pengguna dapat melihat daftar kategori *resource* yang tersedia pada sistem.
		Memperbaharui data kategori *resource* terpilih.	Pengguna dapat melakukan perubahan data pada kategori *resource* yang dipilih.
		Menghapus data kategori *resource* terpilih.	Pengguna data menghapus data kategori *resource* yang dipilih.
6	*User*	Mendaftarkan pengguna pada sistem.	Data pengguna baru dapat disimpan pada basis data sistem.
		Melihat daftar pengguna yang terdaftar pada sistem.	Pengguna dapat melihat daftar pengguna yang terdaftar pada sistem.
		Memperbaharui data pengguna terpilih.	Pengguna dapat melakukan perubahan data pada pengguna yang dipilih.

Tabel 5.3: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil Harapan
		Menghapus data pengguna terpilih.	Pengguna dapat menghapus data pengguna yang dipilih.
7	*API Secret Key*	Membuat *API Secret Key* baru.	Data *API Secret Key* baru dapat disimpan pada basis data sistem.
		Melihat daftar *API Secret Key* yang tersedia pada pengguna tertentu.	Pengguna dapat melihat daftar *API Secret Key* yang tersedia.
		Memperbaharui data *API Secret Key* terpilih.	Pengguna dapat melakukan perubahan data pada *API Secret Key* yang dipilih.
		Menghapus data *API Secret Key* terpilih.	Pengguna menghapus data *API Secret Key* yang dipilih.
8	*Virtual Machine*	Membuat *virtual machine* baru.	Data *virtual machine* baru dibuat dan data terkait *virtual machine* tersimpan pada basis data sistem.

Tabel 5.3: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil Harapan
		Melihat detail data *virtual machine* terpilih.	Pengguna dapat melihat detail data *virtual machine* yang dipilih.
		Mengubah kategori *resource virtual machine* terpilih.	Pengguna dapat melakukan perubahan data pada kategori *resource virtual machine* yang dipilih.
		Menghapus *virtual machine* terpilih.	Pengguna dapat menghapus *virtual machine* yang dipilih.
		Mematikan *virtual machine* terpilih.	Pengguna dapat mematikan *virtual machine* yang dipilih.
		Menyalakan *virtual machine* terpilih.	Pengguna dapat menyalakan *virtual machine* yang dipilih.
		Membagikan hak pengelolaan *virtual machine* kepada pengguna lain.	Pengguna dapat membagikan hak pengelolaan *virtual machine* kepada pengguna yang dipilih.

Tabel 5.3: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil Harapan
		Melihat daftar pengguna yang memiliki hak pengelolaan *virtual machine*.	Pengguna dapat melihat daftar pengguna yang memiliki hak pengelolaan *virtual machine*.
		Menghapus pengguna terpilih yang dari hak pengelolaan *virtual machine*.	Pengguna dapat menghapus pengguna terpilih dari hak pengelolaan *virtual machine*.
		Melihat riwayat *task* yang dilakukan pengguna terhadap *virtual machine*.	Pengguna dapat melihat daftar riwayat *task* yang dilakukan dirinya sendiri maupun pengguna lain terhadap *virtual machine*.
		Melihat status *virtual machine* terpilih.	Pengguna dapat melihat status *virtual machine* yang dipilih.
9	*Task*	Melihat daftar *task*	Pengguna dapat melihat daftar *task* yang dilakukan oleh pengguna yang sedang *login*

Tabel 5.3: Skenario Uji Fungsionalitas Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil Harapan
10	*Dashboard*	Melihat daftar *virtual machine* yang tersedia	Pengguna dapat melihat daftar *virtual machine* yang tersedia.

5.2.1.3 Uji Fungsionalitas Mengelola *Virtual Machine* Menggunakan *Command Line Interface*

Pengujian fungsionalitas selanjutnya, penulis menguji fitur-fitur pengelolaan *virtual machine* melalui *Command Line Interface*. Rancangan pengujian dan hasil yang diharapkan ditunjukkan pada Tabel 5.4.

Tabel 5.4: Skenario Uji Fungsionalitas Mengelola *Virtual Machine* Menggunakan *Command Line Interface*

No	Menu	Uji Coba	Hasil Harapan
1	Autentifikasi	Mengatur *API Secret Key* pada *Command Line Interface*	Pengguna dapat mengatur *API Secret Key* agar terautentifikasi dan terotentifikasi *Command Line Interface*.
2	Config	Mengatur konfigurasi *Command Line Interface*	Pengguna dapat mengatur konfigurasi dasar *Command Line Interface*.

Tabel 5.4: Skenario Uji Fungsionalitas Mengelola *Virtual Machine* Menggunakan *Command Line Interface*

No	Menu	Uji Coba	Hasil Harapan
3	Show	Melihat daftar versi sistem operasi yang tersedia	Pengguna dapat melihat daftar versi sistem operasi yang tersedia.
		Melihat kategori *resource* yang tersedia	Pengguna dapat melihat kategori *resource* yang tersedia.
4	Status	Melihat status *virtual machine*	Pengguna dapat melihat status *virtual machine*.
5	VM	Membuat *virtual machine* baru	Pengguna dapat melihat status *virtual machine*.
		Mematikan *virtual machine*	Pengguna dapat mematikan *virtual machine* melalui *Command Line Interface*.
		Menyalakan *virtual machine*	Pengguna dapat menyalakan *virtual machine* melalui *Command Line Interface*.

Tabel 5.4: Skenario Uji Fungsionalitas Mengelola *Virtual Machine* Menggunakan *Command Line Interface*

No	Menu	Uji Coba	Hasil Harapan
		Menghapus *virtual machine*	Pengguna dapat menghapus *virtual machine* melalui *Command Line Interface*.

5.2.1.4 Uji Fungsionalitas Distribusi Alokasi *Virtual Machine*

Pengujian fungsionalitas yang terakhir adalah pengujian distribusi alokasi *virtual machine* baru yang diminta oleh pengguna. Setiap permintaan alokasi *virtual machine* baru, sistem akan menghitung server terbaik menggunakan algoritma AHP. Skenario pengujian, dengan melakukan permintaan alokasi *virtual machine* dengan jumlah *concurrent user* 5, 10, dan 15. Pada pengujian ini, penulis menggunakan *worker* sebanyak 6. Komputer pengujian akan menjalakan *script* Python dengan memasukkan parameter jumlah *concurrent user* sesuai skenario pengujian.

5.2.2 Skenario Uji Coba Performa

Uji performa dilakukan dengan menggunakan *script* Python yang mensimulasikan permintaan pengguna untuk melakukan akses secara bersamaan ke aplikasi. *Script* Python akan mengakses *end-point* pembuatan *virtual machine* pada *HTTP Rest API* dengan *parameter* jumlah *concurrent user* yang ditentukan oleh penulis.

Pengujian dilakukan dengan melakukan permintaan alokasi *virtual machine* dengan jumlah *concurrent user* 5, 10, dan 15

sebagai *parameter* pertama. Selain jumlah *concurrent user*, jumlah *worker* menjadi *parameter* pengujian. Jumlah *worker* yang digunakan adalah 4, 6, dan 8. Skenario pengujian dapat dilihat pada Tabel 5.5.

Tabel 5.5: Skenario Uji Performa

No	Jumlah *Worker*	*Concurrent User*
1	4	5
		10
		15
2	6	5
		10
		15
3	8	5
		10
		15

Pengujian *request* ini bertujuan untuk mengukur kemampuan dari *middleware* dalam menangani permintaan alokasi *virtual machine* baru. Keberhasilan permintaan dari pengguna, tidak diukur berdasarkan waktu umpan balik dari *middleware* tetapi diukur kemampuan *middleware* dalam menangani *task* alokasi *virtual machine* baru yang diberikan pengguna sejumlah skenario yang sudah ditentukan. Pada pengujian performa, terdapat 2 pengujian yaitu pengujian terhadap kecapatan menganangi *success task* dari pengguna dan pengujian terhadap keberhasilan *request* dari pengguna.

5.2.2.1 Uji Performa Kecepatan Menangani *Success Task*

Pengujian dilakukan dengan mengukur rata-rata waktu yang diperlukan untuk menyelesaikan *success task* yang dilakukan

oleh komputer penguji. Waktu yang diukur adalah rata-rata dari total waktu yang dibutuhkan dalam alokasi *virtual machine* pertama sampai dengan yang terakhir pada setiap *hypervisor*.

5.2.2.2 Uji Performa Keberhasilan *Request*

Pengujian dilakukan dengan menghitung jumlah persentase kegagalan dari *request* yang dikirimkan selama skenario dijalankan.

5.3 Hasil Uji Coba dan Evaluasi

Berikut dijelaskan hasil uji coba dan evaluasi berdasarkan skenario yang telah dijelaskan pada subbab 5.2.

5.3.1 Uji Fungsionalitas

Berikut dijelaskan hasil pengujian fungsionalitas pada sistem yang dibangun.

5.3.1.1 Uji Mengelola Sistem Menggunakan *Rest Client*

Pengujian dilakukan sesuai dengan skenario yang dijelaskan pada subbab 5.2.1.1 dan pada Tabel 5.2. Hasil pengujian seperti tertera pada Tabel 5.6.

Tabel 5.6: Hasil Uji Coba Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil
1	Autentifikasi	Melakukan permintaan autentifikasi dan otorisasi dengan mengirimkan *username* dan *password*.	Berhasil

Tabel 5.6: Hasil Uji Coba Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil
		Melakukan permintaan autentifikasi dan otorisasi dengan mengirimkan *HTTP Rest API*.	Berhasil
2	*Host*	Menambahkan server baru.	Berhasil
		Melihat daftar server yang tersedia.	Berhasil
		Melihat data server terpilih.	Berhasil
		Memperbaharui data server terpilih.	Berhasil
		Menghapus data server terpilih.	Berhasil
3	*OS*	Menambahkan versi sistem operasi yang didukung oleh sistem.	Berhasil
		Melihat daftar versi sistem operasi yang tersedia	Berhasil
		Melihat data versi sistem operasi terpilih.	Berhasil
		Memperbaharui data versi sistem operasi terpilih.	Berhasil

Tabel 5.6: Hasil Uji Coba Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil
		Menghapus data versi sistem operasi terpilih.	Berhasil
4	*Template*	Menambahkan *template* sistem operasi yang didukung oleh sistem.	Berhasil
		Melihat daftar *template* sistem operasi yang tersedia.	Berhasil
		Melihat data *template* sistem operasi terpilih.	Berhasil
		Memperbaharui data *template* sistem operasi terpilih.	Berhasil
		Menghapus data *template* sistem operasi terpilih.	Berhasil
5	Kategori *Resource*	Menambahkan kategori *resource* untuk pilihan *resource virtual machine*.	Berhasil
		Melihat daftar kategori *resource* yang tersedia.	Berhasil

Tabel 5.6: Hasil Uji Coba Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil
		Melihat data kategori *resource* terpilih.	Berhasil
		Memperbaharui data kategori *resource* terpilih.	Berhasil
		Menghapus data kategori *resource* terpilih.	Berhasil
6	*User*	Mendaftarkan pengguna pada sistem.	Berhasil
		Melihat daftar pengguna yang terdaftar pada sistem.	Berhasil
		Melihat data pengguna terpilih.	Berhasil
		Memperbaharui data pengguna terpilih.	Berhasil
		Menghapus data pengguna terpilih.	Berhasil
7	*API Secret Key*	Membuat *API Secret Key* baru.	Berhasil
		Melihat daftar *API Secret Key* yang tersedia pada pengguna tertentu.	Berhasil
		Melihat data *API Secret Key* terpilih.	Berhasil

Tabel 5.6: Hasil Uji Coba Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil
		Memperbaharui data *API Secret Key* terpilih.	Berhasil
		Menghapus data *API Secret Key* terpilih.	Berhasil
8	*Virtual Machine*	Membuat *virtual machine* baru	Berhasil
		Melihat daftar *virtual machine* yang tersedia pada pengguna tertentu.	Berhasil
		Melihat detail data *virtual machine* terpilih.	Berhasil
		Mengubah kategori *resource virtual machine* terpilih.	Berhasil
		Menghapus *virtual machine* terpilih.	Berhasil
		Mematikan *virtual machine* terpilih.	Berhasil
		Menyalakan *virtual machine* terpilih.	Berhasil
		Membagikan hak pengelolaan *virtual machine* kepada pengguna lain.	Berhasil

Tabel 5.6: Hasil Uji Coba Mengelola Sistem Menggunakan *Rest Client*

No	Menu	Uji Coba	Hasil
		Melihat daftar pengguna yang memiliki hak pengelolaan *virtual machine*.	Berhasil .
		Menghapus pengguna terpilih yang dari hak pengelolaan *virtual machine*.	Berhasil
		Melihat riwayat *task* yang dilakukan pengguna terhadap *virtual machine*.	Berhasil
		Melihat status *virtual machine* terpilih.	Berhasil
9	*Task*	Melihat daftar *task*	Berhasil

Sesuai dengan skenario uji coba yang diberikan pada Tabel 5.2, hasil uji coba menunjukkan semua skenario berhasil ditangani.

5.3.1.2 Uji Fungsionalitas Mengelola Sistem Menggunakan *Interface* Web

Sesuai dengan skenario pengujian yang dilakukan pada *interface* web. Pengujian dilakukan dengan menguji setiap menu pada *interface* web. Hasil uji coba dapat dilihat pada Table 5.7. Semua skenario yang direncanakan berhasil ditangani.

Tabel 5.7: Hasil Uji Coba Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil
1	Autentifikasi	Melakukan permintaan autentifikasi dan otorisasi dengan mengirimkan *username* dan *password*.	Berhasil
2	*Host*	Menambahkan server baru.	Berhasil
		Melihat daftar server yang tersedia.	Berhasil
		Memperbaharui data server terpilih.	Berhasil
		Menghapus data server terpilih.	Berhasil
3	*OS*	Menambahkan versi sistem operasi yang didukung oleh sistem.	Berhasil
		Melihat daftar versi sistem operasi yang tersedia	Berhasil
		Memperbaharui data versi sistem operasi terpilih.	Berhasil
		Menghapus data versi sistem operasi terpilih.	Berhasil

Tabel 5.7: Hasil Uji Coba Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil
4	*Template*	Menambahkan *template* sistem operasi yang didukung oleh sistem.	Berhasil
		Melihat daftar *template* sistem operasi yang tersedia.	Berhasil
		Melihat data *template* sistem operasi terpilih.	Berhasil
		Memperbaharui data *template* sistem operasi terpilih.	Berhasil
		Menghapus data *template* sistem operasi terpilih.	Berhasil
5	Kategori *Resource*	Menambahkan kategori *resource* untuk pilihan *resource virtual machine*.	Berhasil
		Melihat daftar kategori *resource* yang tersedia.	Berhasil
		Memperbaharui data kategori *resource* terpilih.	Berhasil

Tabel 5.7: Hasil Uji Coba Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil
		Menghapus data kategori *resource* terpilih.	Berhasil
6	*User*	Mendaftarkan pengguna pada sistem.	Berhasil
		Melihat daftar pengguna yang terdaftar pada sistem.	Berhasil
		Memperbaharui data pengguna terpilih.	Berhasil
		Menghapus data pengguna terpilih.	Berhasil
7	*API Secret Key*	Membuat *API Secret Key* baru.	Berhasil
		Melihat daftar *API Secret Key* yang tersedia pada pengguna tertentu.	Berhasil
		Memperbaharui data *API Secret Key* terpilih.	Berhasil
		Menghapus data *API Secret Key* terpilih.	Berhasil
8	*Virtual Machine*	Membuat *virtual machine* baru.	Berhasil.

Tabel 5.7: Hasil Uji Coba Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil
		Melihat detail data *virtual machine* terpilih.	Berhasil
		Mengubah kategori *resource virtual machine* terpilih.	Berhasil
		Menghapus *virtual machine* terpilih.	Berhasil
		Mematikan *virtual machine* terpilih.	Berhasil
		Menyalakan *virtual machine* terpilih.	Berhasil
		Membagikan hak pengelolaan *virtual machine* kepada pengguna lain.	Berhasil
		Melihat daftar pengguna yang memiliki hak pengelolaan *virtual machine*.	Berhasil
		Menghapus pengguna terpilih yang dari hak pengelolaan *virtual machine*.	Berhasil
		Melihat riwayat *task* yang dilakukan pengguna terhadap *virtual machine*.	Berhasil

Tabel 5.7: Hasil Uji Coba Mengelola Sistem Menggunakan *Interface* Web

No	Menu	Uji Coba	Hasil
		Melihat status *virtual machine* terpilih.	Berhasil
9	*Task*	Melihat daftar *task*	Berhasil
10	*Dashboard*	Melihat daftar *virtual machine* yang tersedia	Berhasil

5.3.1.3 Uji Fungsionalitas Mengelola *Virtual Machine* Menggunakan *Command Line Interface*

Sesuai dengan skenario pengujian yang dilakukan pada aplikasi *Command Line Interface*. Pengujian dilakukan dengan menguji setiap parameter/menu pada *Command Line Interface*. Hasil uji coba dapat dilihat pada Table 5.8. Semua skenario yang direncanakan berhasil ditangani.

Tabel 5.8: Hasil Uji Coba Mengelola *Virtual Machine* Menggunakan *Command Line Interface*

No	Menu	Uji Coba	Hasil Harapan
1	Autentifikasi	Mengatur *API Secret Key* pada *Command Line Interface*	Berhasil
2	Config	Mengatur konfigurasi *Command Line Interface*	Berhasil
3	Show	Melihat daftar versi sistem operasi yang tersedia	Berhasil

Tabel 5.8: Hasil Uji Coba Mengelola *Virtual Machine* Menggunakan *Command Line Interface*

No	Menu	Uji Coba	Hasil
		Melihat kategori *resource* yang tersedia	Berhasil
4	Status	Melihat status *virtual machine*	Berhasil
5	VM	Membuat *virtual machine* baru	Berhasil
		Mematikan *virtual machine*	Berhasil
		Menyalakan *virtual machine*	Berhasil
		Menghapus *virtual machine*	Berhasil

5.3.1.4 Uji Fungsionalitas Distribusi Alokasi *Virtual Machine*

Sesuai dengan skenario pengujian distribusi alokasi *virtual machine*. Pengujian dilakukan dengan mengganti *parameter concurrent user* untuk mengetahui distribusi alokasi *virtual machine*.

Kondisi awal ketersedian sumber daya *server* dapat dilihat pada Tabel 5.9. Data dalam bentuk persentase ketersediaan sumber daya.

Tabel 5.9: Persentase Kondisi Awal Ketersediaan Sumber Daya pada Server

No	*Server*	CPU (%)	*Memory* (%)	*Storage* (%)
1	Vmware ESXI A	99.86	64.34	85.97

Tabel 5.9: Persentase Kondisi Awal Ketersediaan Sumber Daya pada Server

No	***Server*** (%)	**CPU** (%)	***Memory*** (%)	***Storage*** (%)
2	Proxmox A	99.19	54.89	75.33
3	Proxmox B	99.89	91.85	99.72

Setelah dilakukan pengujian, hasil distribusi alokasi *virtual machine* dapat dilihat pada Tabel 5.10.

Tabel 5.10: Hasil Uji Coba Fungsionalitas Distribusi Alokasi *Virtual Machine*

No	Jumlah *User*	*VM* yang Teralokasi pada *Server*		
		Proxmox A	Proxmox B	Vmware ESXI A
1	5	0	5	0
2	10	0	10	0
3	15	0	0	15

Setelah dilakukan pengujian, persentase kondisi akhir ketersediaan sumber daya dapat dilihat pada Tabel 5.11.

Tabel 5.11: Persentase Kondisi Akhir Ketersediaan Sumber Daya pada Server

No	***Server*** (%)	**CPU** (%)	***Memory*** (%)	***Storage*** (%)
1	Vmware ESXI A	99.13	60.24	81.83
2	Proxmox A	99.19	52.44	75.28
3	Proxmox B	98.89	75.63	97.09

5.3.2 Hasil Uji Performa

Seperti yang sudah dijelaskan pada subbab 5.2 pengujian performa dilakukan dengan melakukan akses ke aplikasi dengan sejumlah pengguna secara bersama-sama. Pengujian dilakukan dengan melakukan permintaan alokasi *virtual machine* dengan jumlah *concurrent user* 5, 10, dan 15 sebagai *parameter* pertama. Selain jumlah *concurrent user*, jumlah *worker* menjadi *parameter* pengujian. Jumlah *worker* yang digunakan adalah 4, 6, dan 8.

Jumlah *worker* dan *concurrent user* merupakan *parameter* yang digunakan untuk melihat performa sistem. Setiap penggantian jumlah *concurrent user*, *virtual machine* yang sudah dialokasikan tidak dihapus terlebih dahulu. Sedangkan untuk penggantian jumlah *worker*, *virtual machine* yang sudah dialokasikan sebelumnya akan dihapus terlebih dahulu agar sumber daya pada server tidak habis dan untuk meringankan beban server.

Pada pengujian ini, *virtual machine* yang akan dialokasikan memiliki spesifikasi sebagai berikut, sistem operasi menggunakan Ubuntu, *memory* sebesar 512MB, *CPU cores* sebanyak 1 *core* dan *storage* sebesar 21GB.

Hasil pengujian menggunakan skenario pada Tabel 5.5, didapatkan distribusi alokasi *virtual machine* yang dapat dilihat pada Tabel 5.12.

Tabel 5.12: Hasil Uji Coba Distribusi Alokasi *Virtual Machine* Menggunakan Skenario Uji Performa

No	Jumlah *Worker*	Jumlah *User*	*VM* yang Teralokasi pada *Server*		
			Proxmox A	Proxmox B	Vmware ESXI A
1	4	5	0	5	0
		10	0	10	0

		15	0	8	7
2	6	5	0	5	0
		10	0	10	0
		15	0	0	15
3	8	5	0	5	0
		10	0	10	0
		15	0	15	0

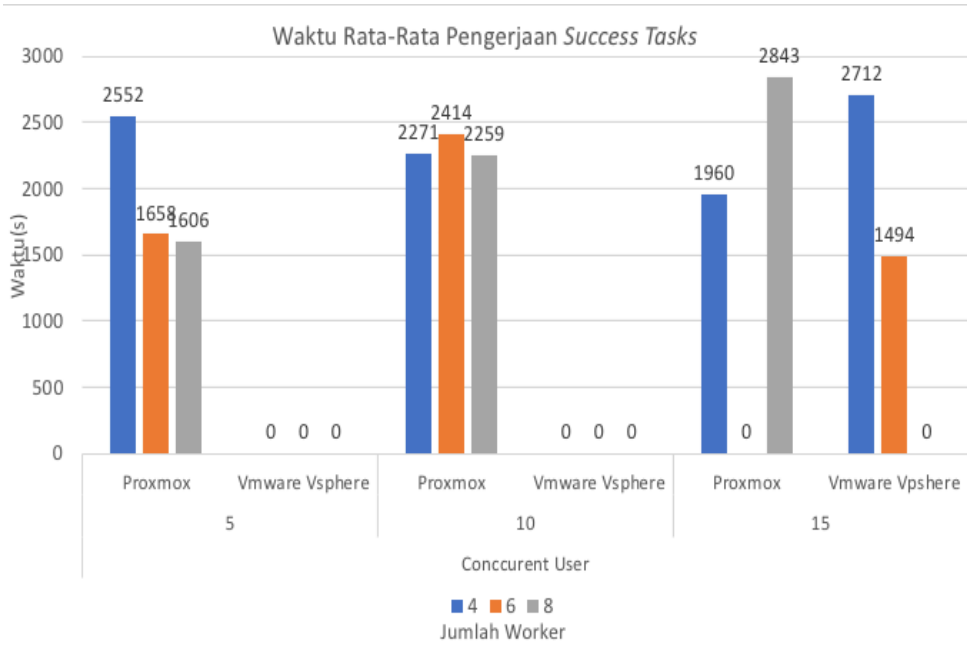
Hasil distribusi alokasi *virtual machine* dapat mempengaruhi hasil uji coba performa sistem.

5.3.2.1 Uji Performa Kecepatan Menangani *Success Task*

Dari hasil uji coba kecepatan menangani *success tasks*, dapat dilihat pada Table 5.13 dalam satuan detik.

Tabel 5.13: Hasil Uji Coba Waktu Rata-Rata Kecepatan Menangani *Success Tasks*

No	Jumlah *Worker*	Jumlah *Concurrent User*					
		5		10		15	
		Prox-mox (s)	Vmware Vsphere (s)	Prox-mox (s)	Vmware Vsphere (s)	Prox-mox (s)	Vmware Vsphere (s)
1	4	2552	0	2271	0	1960	2712
2	6	1658	0	2414	0	0	1494
3	8	1606	0	2259	0	2843	0



Gambar 5.1: Grafik Waktu Rata-Rata Pengerjaan *Success Task*

Pada hasil uji coba, lama waktu setiap *task* dikerjakan dihitung dari selisih waktu *task* selesai dikerjakan dengan waktu *task* dibuat. Waktu tunggu *task* dikerjakan sangat mempengaruhi lama *task* dikerjakan. Semakin lama suatu *task* tersimpan pada *queue*, semakin lama pula waktu pengerjaan *task* yang dibutuhkan.

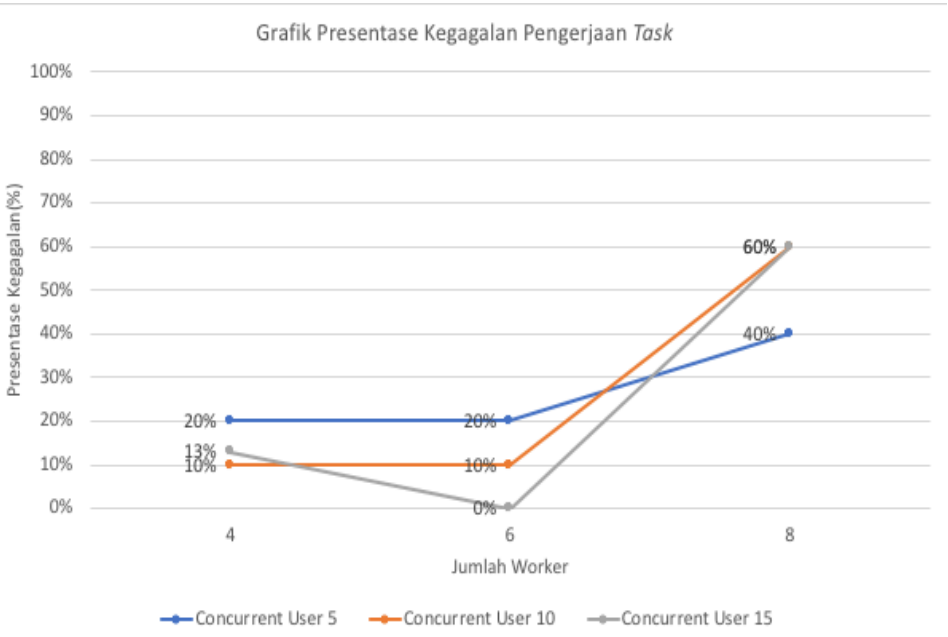
Selain terpengaruhi oleh *queue*, lama waktu penerjaan *task* juga dipengaruhi jenis *hypervisor* yang terpilih karena setiap *hypervisor* memiliki mekanisme alokasi tersendiri. Ketika *parameter* jumlah *worker* yang diatur adalah 6 dengan *concurrent user* 15, semua *virtual machine* teralokasikan pada Vmware Vsphere sehingga waktu lebih cepat dibanding dengan pengujian lainnya. Untuk alokasi Proxmox membutuhkan waktu alokasi *virtual machine* lebih lama daripada *hypervisor* Vmware Vsphere dikarenakan harus mengirimkan *template* sistem operasi ke *server* terbaik terlebih dahulu.

5.3.2.2 Uji Performa Keberhasilan *Request*

Pada uji coba ini, dilakukan perhitungan jumlah persentase kegagalan dari *request* yang dikirimkan selama skenario dijalankan. Hasil pengujian dapat dilihat pada Tabel 5.14.

Tabel 5.14: *Error Ratio Request*

No	Jumlah *Worker*	Jumlah *Error Ratio* Setiap *Concurrent User* (%)		
		5	10	15
1	4	20	10	13
2	6	20	10	0
3	8	40	60	60



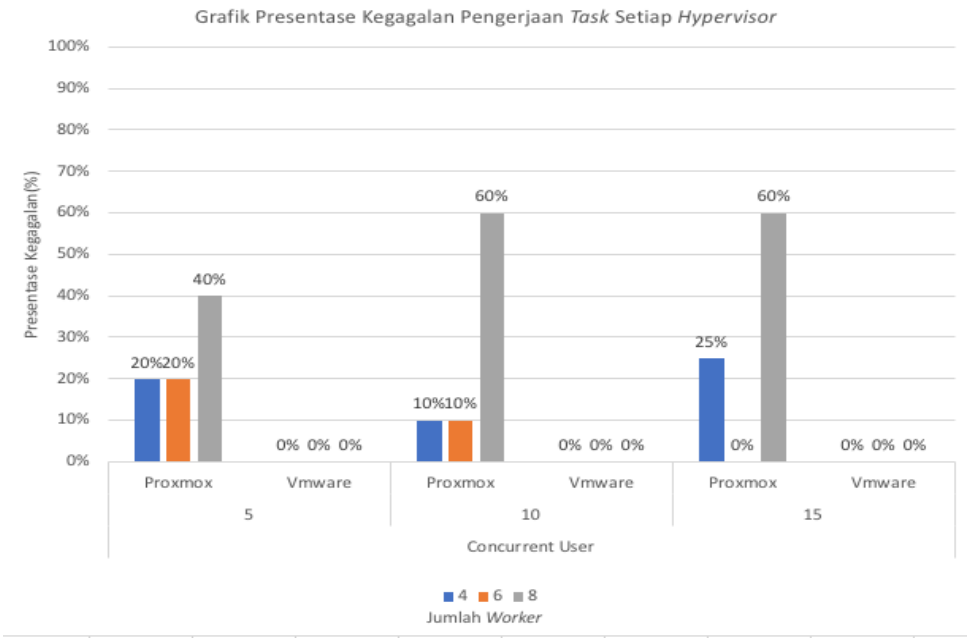
Gambar 5.2: Grafik Presentase Kegagalan Pengerjaan *Task*

Dari hasil uji coba terlihat bahwa terjadi *trend* kenaikan presentase kegagalan. Presentase kegagalan terbesar terjadi pada skenario jumlah *concurrent user* 15 dengan jumlah *worker* 8 dan skenario jumlah *concurrent user* 10 dengan jumlah *worker* 8.

Untuk skenario jumlah *concurrent user* 15 dengan jumlah *worker* 8, sebanyak 9 *request* yang gagal diselesaikan oleh sistem pada saat alokasi *virtual machine*. Sedangkan untuk skenario jumlah *concurrent user* 10 dengan jumlah *worker* 8, sebanyak 6 *request* yang gagal diselesaikan oleh sistem pada saat alokasi *virtual machine*.

Tabel 5.15: *Error Ratio Request* Setiap *Hypervisor*

No	Jumlah *Worker*	Jumlah Concurrent User					
		5		10		15	
		Prox-mox (%)	Vmware Vsphere (%)	Prox-mox (%)	Vmware Vsphere (%)	Prox-mox (%)	Vmware Vsphere (%)
1	4	20	0	10	0	25	0
2	6	20	0	10	0	0	0
3	8	40	0	60	0	60	0



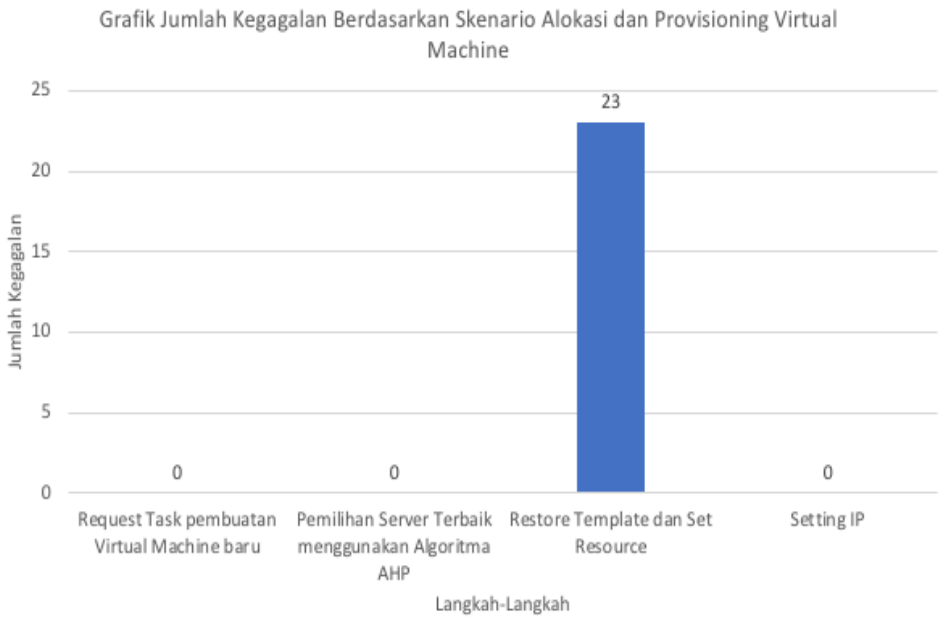
Gambar 5.3: Grafik Presentase Kegagalan Pengerjaan *Task* Setiap *Hypervisor*

Jika dilihat dari distribusi alokasi *virtual machine*, presentase kegagalan pada setiap *hypervisor* dapat dilihat pada Tabel 5.15. *Hypervisor* Proxmox sering terjadi kegagalan pada saat alokasi *virtual machine*. Namun pada skenario dengan jumlah *concurrent user* 15 dan jumlah *worker* 6, tidak terjadi kegagalan sama sekali pada *hypervisor* Proxmox. Hal tersebut dikarenakan semua *virtual machine* dialokasikan pada *hypervisor* Vmware Vsphere.

Berdasarkan hasil uji coba, pada saat alokasi *virtual machine* di *hypervisor* Vmware Vsphere, tidak terjadi kegagalan sama sekali dengan melihat hasil skenario jumlah *concurrent user* 15 dengan jumlah *worker* 4 dan skenario jumlah *concurrent user* 15 dengan jumlah *worker* 6.

Tabel 5.16: Jumlah Kegagalan Berdasarkan Langkah-Langkah Skenario Alokasi dan *Provisioning Virtual Machine*

No	**Langkah-Langkah**	**Jumlah *Task* yang Gagal**
1	*Request Task* pembuatan *Virtual Machine* baru	0
2	Pemilihan *Server* Terbaik Menggunakan Algoritma AHP	0
3	*Restore Template* dan *Set Resource*	23
4	*Setting IP*	0



Gambar 5.4: Grafik Presentase Kegagalan Berdasarkan Skenario Alokasi dan *Provisioning Virtual Machine*

Berdasarkan semua skenario hasil pengujian, jika dilihat dari skenario alokasi dan *provisioning virtual machine* baru. Semua kegagalan terjadi pada langkah *restore template* dan *set resource*. Terdapat dua penyebab terjadinya kegagalan. Pertama disebabkan karena *request timeout* pada saat melakukan *restore template* sebanyak 19 *task* dan yang kedua disebabkan *server* tidak me-*response* perintah dari *middleware* sebanyak 4 *task*.

BAB VI

PENUTUP

Bab ini membahas kesimpulan yang dapat diambil dari tujuan pembuatan sistem dan hubungannya dengan hasil uji coba dan evaluasi yang telah dilakukan. Selain itu, terdapat beberapa saran yang bisa dijadikan acuan untuk melakukan pengembangan dan penelitian lebih lanjut.

6.1 Kesimpulan

Dari proses perencangan, implementasi dan pengujian terhadap sistem, dapat diambil beberapa kesimpulan berikut:

1. Sistem dapat melakukan manajemen alokasi *virtual machine* pada lingkungan *hypervisor* yang heterogen. *Hypervisor* yang didukung oleh sistem adalah *Vmware Vsphere* dan *Proxmox*.
2. Sistem dapat membagi distribusi alokasi *virtual machine* baru pada *server* yang tersedia dengan algoritma *Analytical Hierarchy Process*.
3. Sistem dapat melakukan *Provisioning* sampai proses pengaturan IP berdasarkan *hypervisor* dan sistem operasi.
4. Sistem dapat diakses oleh pengguna melalui *interface* web dan *command line interface*.
5. Dari hasil pengujian performa, semakin banyak *worker* yang digunakan sangat rawan terjadinya kegagalan alokasi *virtual machine* pada *hypervisor* Proxmox.

6.2 Saran

Berikut beberapa saran yang diberikan untuk pengembangan lebih lanjut:

1. Untuk mempercepat waktu alokasi pada *hypervisor* Proxmox, diperlukan *storage area network* sebagai tempat menyimpan *file template* sistem operasi sehingga saat

alokasi *virtual machine* baru, *middleware* tidak perlu mengirimkan file *template* terlebih dahulu.

2. Untuk memperbanyak dukungan terhadap sistem operasi, untuk pengaturan IP dapat dilakukan dengan mekanisme *IP Floating*.

DAFTAR PUSTAKA

[1] N. Jain dan S. Choudhary, "Overview of Virtualization in Cloud Computing," in *Colossal Data Analysis and Networking (CDAN), Symposium on*, 2012.

[2] "General Python FAQ," 3 Mei 2018. [Daring]. Tersedia pada: https://docs.python.org/3/faq/general.html#what-is-python. [Diakses: 3 Mei 2018].

[3] "A simple framework for building complex web applications," 3 Mei 2018. [Daring]. Tersedia pada: https://pypi.org/project/Flask/. [Diakses: 3 Mei 2018].

[4] "Celery," 21 Mei 2018. [Daring]. Tersedia pada: http://www.celeryproject.org/. [Diakses: 21 Mei 2018].

[5] "Vmware vsphere Python SDK," 3 Mei 2018. [Daring]. Tersedia pada: https://pypi.org/project/pyvmomi/. [Diakses: 3 Mei 2018].

[6] "Python Wrapper for the Proxmox 2.x API (HTTP and SSH)," 3 Mei 2018. [Daring]. Tersedia pada: https://pypi.org/project/proxmoxer/. [Diakses: 3 Mei 2018].

[7] "What can PHP do?" 3 Mei 2018. [Daring]. Tersedia pada: https://secure.php.net/manual/en/intro-whatcando.php. [Diakses: 3 Mei 2018].

[8] "Redis," 10 April 2017. [Daring]. Tersedia pada: https://redis.io/. [Diakses: 10 April 2017].

[9] W. J. Gilmore, "Beginning PHP and MySQL From Novice to Professional," *Apress*, vol. 4th, hal. 477–480, 2010.

[10] F. Mohammad, V. Yadav, dan others, "Automatic decision making for multi-criteria load balancing in cloud environment using AHP," in *Computing, Communication & Automation (ICCCA), 2015 International Conference on*. IEEE, 2015, hal. 569–576.].

[11] T. L. Saaty, “Decision making with the analytic hierarchy process,” *International journal of services sciences*, vol. 1, no. 1, hal. 83–98, 2008.

[12] R. M. Ijtihadie, B. J. Santoso, D. Fablius, dan I. D. P. A. Nusawan, “Rancang bangun sistem penentuan keputusan untuk distribusi penyedian kontainer dengan multi kriteria secara dinamis,” 2017, hal. 198–199.].

LAMPIRAN A

INSTALASI PERANGKAT LUNAK

Instalasi Pustaka Python

Dalam pengembangan sistem ini, digunakan berbagai pustaka pendukung. Pustaka pendukung yang digunakan merupakan pustaka untuk bahasa pemrograman Python. Berikut adalah daftar pustaka yang digunakan dan cara pemasangannya:

- Python Dev
  $ `sudo apt-get install python-dev`
- Flask
  $ `sudo pip install Flask`
- Pyvmomi
  $ `sudo pip install pyvmomi`
- Proxmoxer
  $ `sudo pip install proxmoxer`
- MySQLd
  $ `sudo apt-get install python-mysqldb`
- Redis
  $ `sudo pip install redis`
- Celery
  $ `sudo pip install celery`

Pemasangan Redis

Redis dapat dipasang dengan mempersiapkan kebutuhan pustaka pendukungnya. Pustaka yang digunakan adalah `build-essential` dan `tcl8.5`. Untuk melakukan pemasangannya, jalankan perintah berikut:

$ `sudo apt-get install build-essential`
$ `sudo apt-get install tcl8.5`

Setelah itu unduh aplikasi Redis dengan menjalankan perintah `wget http://download.redis.io/releases/redis-stable.tar.gz`.

Setelah selesai diunduh, buka file dengan perintah berikut:
$ `tar xzf redis-stable.tar.gz && cd redis-stable`

Di dalam folder `redis-stable`, bangun Redis dari kode sumber dengan menjalankan perintah `make`. Setelah itu lakukan tes kode sumber dengan menjalankan `make test`. Setelah selesai, pasang Redis dengan menggunakan perinah `sudo make install`. Setelah selesai melakukan pemasangan, Redis dapat diaktifkan dengan menjalankan berkas bash dengan nama `install_server.sh`.

Untuk menambah pengaman pada Redis, diatur agar Redis hanya bisa dari *localhost*. Untuk melakukannya, buka file `/etc/redis/6379.conf`, kemudian cari baris `bind 127.0.0.1`. Hapus komen jika sebelumnya baris tersebut dalam keadaan tidak aktif. Jika tidak ditemukan baris dengan isi tersebut, tambahkan pada akhir berkas baris tersebut.

Menjalankan Aplikasi *Middleware*

Untuk menjalankan *middleware*, jalankan perintah sebagai berikut:
$ *python server.py*
Setelah menjalankan perintah tersebut, *middleware* dapat diakses menggunakan port 9000.

Menjalankan *Celery Worker*

Untuk menjalankan *celery worker*, jalankan perintah perintah sebagai berikut:
$ *celery worker -E –app=app.Library.Tasks.celery_worker -c 8 -Q hypgen_queue –loglevel=DEBUG*
Pada perintah diatas, terdapat tiga *parameter* penting yaitu, *–app* yang digunakan sebagai *parameter* variabel objek python *celery*, *parameter -c* yang digunakan sebagai *parameter* jumlah *worker*

dan *parameter -Q* sebagai nama *queue* yang sudah diatur pada konfigurasi *middleware*.

Instalasi *Interface* Web

Web interface dikembangkan menggunakan *framework* Laravel. Untuk melakukan instalasi jalankan perintah berikut:
$ *composer install*
$ *php artisan:key generate*

Instalasi *Command Line Interface*

Command Line Interface dibuat menggunakan bahasa pemrograman Python. Untuk menggunakan *command line interface*, pengguna diharuskan melakukan *compile* dengan menjalankan perintah berikut pada folder kode sumber *command line interface*:
$ *pip install –editable .*

(Halaman ini sengaja dikosongkan)

LAMPIRAN B

KODE SUMBER

File Environment Middleware

```
HYPGEN_DEFAULT_DB_DRIVER=mysql
HYPGEN_DEFAULT_DB_HOST=localhost
HYPGEN_DEFAULT_DB_USER=root
HYPGEN_DEFAULT_DB_PASSWORD=password
HYPGEN_DEFAULT_DB_DATABASE=hypgen
HYPGEN_DEFAULT_DB_PORT=3306
HYPGEN_RESOURCES_FOLDER=resources/views

HYPGEN_ENV=development

REDIS_URL=redis://localhost:6379

SECRET_KEY=secret
```

Kode Sumber B.1: File Environment Middleware (.env)

File Environment Interface Web

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:uLobOEiG30dPktmQxjdSFuW/
    dll1O2cePkIzY7MDxBY=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
```

```
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret

BROADCAST_DRIVER=log
CACHE_DRIVER=file
SESSION_DRIVER=file
SESSION_LIFETIME=120
QUEUE_DRIVER=sync

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${
    PUSHER_APP_CLUSTER}"


JWT_SECRET=secret
```

Kode Sumber B.2: File Environment Interface Web (.env)

BIODATA PENULIS

Fathoni Adi Kurniawan, akbrab dipanggil Thoni lahir pada tanggal 4 Maret 1996 di kabupaten Klaten, Jawa Tengah. Penulis merupakan seorang mahasiswa yang sedang menempuh studi di Departemen Informatika Institut Teknologi Sepuluh Nopember. Memiliki hobi antara lain mendengarkan musik dan mencoba *tool-tool* IT yang baru. Selama menempuh pendidikan di kampus, penulis juga aktif dalam organisasi kemahasiswaan, antara lain Staff Departemen Media Informasi (Medfo) Himpunan Mahasiswa Teknik Computer-Informatika pada tahun ke-2. Pernah menjadi staff National Programming Contest Schematics tahun 2015 dan dan pengembang web Schematics 2016. Selain itu penulis pernah menjadi asisten dosen di mata kuliah Sistem Operasi, Jaringan Komputer dan Komputasi Awan.