KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)
Terakreditasi Nomor 204/E/KPT/2022 | Vol. 6, No. 2, April (2025), pp. 421-435

# Implementasi Model *Rational Unified Process* (RUP) dalam Perancangan Infrastruktur *Private Cloud berbasis Web* dengan Pendekatan *Infrastructure as a Service* (IaaS)

*Anis Mirza[1], Yudi Irawan Chandra[2], Melani Dewi Lusita[3]*
*[1]Universitas Pamulang, Banten, Indonesia*
*[2,3]STMIK Jakarta STI&K, Jakarta Selatan, Indonesia*
*E-mail: dosen00289@unpam.ac.id[1], yirawanc@gmail.com[2], melanilusita@gmail.com[3]*

## *Abstract*

*The development of cloud computing technology has a significant impact on the management of information technology infrastructure, especially in the university environment. This research aims to design and build a private cloud infrastructure based on Infrastructure as a Service (IaaS) using OpenStack Zed at Djuanda University Bogor. The utilization of private cloud is intended to support the efficiency of computing resource management, internal data security, and flexibility in providing information technology services. The system development model used is the Rational Unified Process (RUP), which consists of four main phases: inception, elaboration, construction, and transition. Each phase in RUP is executed iteratively and aims to produce a system that is stable, flexible, and can be integrated well within the university environment. The implementation was carried out on a virtualized server environment based on Linux Ubuntu, with OpenStack Zed components including Nova, Neutron, Glance, Keystone, and Horizon. The private cloud development process also involves designing a web-based interface to facilitate users in accessing and managing computing services. The results of this research show that the developed system is able to provide IaaS services effectively, scalable, and improve efficiency in managing IT infrastructure at Djuanda University. With the RUP approach, system development becomes more structured and scalable, thus minimizing errors and speeding up the implementation process. This research is expected to be a reference for private cloud development in other educational institutions that have similar needs in managing IT infrastructure independently and securely.*

***Keywords:*** *Recommendation App, Complexion Shade, Make Up, RUP Model, Web*

## *Abstrak*

*Perkembangan teknologi komputasi awan (cloud computing) memberikan dampak signifikan dalam pengelolaan infrastruktur teknologi informasi, khususnya dalam lingkungan perguruan tinggi. Penelitian ini bertujuan untuk merancang dan membangun infrastruktur private cloud berbasis Infrastructure as a Service (IaaS) dengan menggunakan OpenStack Zed pada Universitas Djuanda Bogor. Pemanfaatan private cloud ini ditujukan untuk mendukung efisiensi pengelolaan sumber daya komputasi, keamanan data internal, serta fleksibilitas dalam penyediaan layanan teknologi informasi. Model pengembangan sistem yang digunakan adalah Rational Unified Process (RUP), yang terdiri dari empat fase utama: inception, elaboration, construction, dan transition. Setiap fase dalam RUP dijalankan secara iteratif dan bertujuan untuk menghasilkan sistem yang stabil, fleksibel, serta dapat diintegrasikan dengan baik dalam lingkungan universitas. Implementasi dilakukan pada lingkungan virtualized server berbasis Linux Ubuntu, dengan komponen OpenStack Zed yang meliputi Nova, Neutron, Glance, Keystone, dan Horizon. Proses pembangunan private cloud ini juga melibatkan perancangan antarmuka berbasis web untuk memudahkan pengguna dalam mengakses dan mengelola layanan komputasi. Hasil dari penelitian ini menunjukkan bahwa sistem*

ISSN: 2720-992X

*yang dikembangkan mampu menyediakan layanan IaaS secara efektif, dapat diskalakan, serta meningkatkan efisiensi dalam pengelolaan infrastruktur TI di Universitas Djuanda. Dengan pendekatan RUP, pembangunan sistem menjadi lebih terstruktur dan terukur, sehingga meminimalkan kesalahan dan mempercepat proses implementasi. Penelitian ini diharapkan menjadi acuan bagi pengembangan private cloud di institusi pendidikan lainnya yang memiliki kebutuhan serupa dalam mengelola infrastruktur TI secara mandiri dan aman.*

***Kata kunci:*** *private cloud, OpenStack Zed, Infrastructure as a Service, RUP Model, Web*

## 1. Pendahuluan

Universitas Djuanda Bogor (UNIDA) merupakan perguruan tinggi swasta di Bogor, Jawa Barat, yang berdiri sejak 1987. UNIDA memiliki komitmen mencetak generasi unggul dan berkarakter melalui integrasi nilai-nilai Islam dalam pendidikan. Dengan visi menjadi universitas unggul yang mencerdaskan kehidupan bangsa dan berstandar internasional, UNIDA fokus tidak hanya pada akademik, tetapi juga pengabdian masyarakat melalui penelitian dan inovasi.

Saat ini, UNIDA menghadapi tantangan dalam pengelolaan infrastruktur TI karena masih mengandalkan *VirtualBox* sebagai platform virtualisasi utama. Keterbatasan dalam skalabilitas, performa, dan keandalan menjadikan sistem kurang optimal dalam mendukung kebutuhan aplikasi yang terus berkembang. Sebagai solusi, migrasi ke infrastruktur private cloud dengan platform OpenStack menjadi alternatif strategis. OpenStack menawarkan keunggulan dalam fleksibilitas, skalabilitas, dan keandalan, yang sesuai untuk sistem yang terus tumbuh [1], [2], [3].

Dengan private cloud berbasis OpenStack, UNIDA dapat meningkatkan efisiensi penggunaan sumber daya, skalabilitas sistem, dan mengurangi beban administrasi melalui otomatisasi. Solusi ini juga memperkuat dukungan layanan akademik dan administratif dengan infrastruktur yang lebih tangguh. Berdasarkan latar belakang di atas, permasalahan yang dapat diidentifikasi meliputi: proses provisioning dan pengelolaan mesin virtual berjalan lambat karena terbatasnya skalabilitas, performa, dan keandalan serta infrastruktur berbasis Oracle *VirtualBox* kurang fleksibel dalam penyesuaian kapasitas secara cepat, yang dapat menyebabkan downtime dan menurunkan produktivitas. Perancangan infrastruktur private cloud berbasis IaaS dengan platform open source OpenStack diharapkan menjadi solusi atas permasalahan tersebut [4], [5], [6].

Berdasarkan hasil identifikasi terhadap permasalahan yang dihadapi, dirumuskan beberapa pertanyaan penelitian utama yang menjadi dasar dalam penyusunan solusi infrastruktur teknologi informasi. Pertanyaan pertama adalah bagaimana cara meningkatkan efisiensi dalam proses provisioning dan pengelolaan mesin virtual, khususnya dalam mengatasi keterbatasan dari sisi skalabilitas, performa, serta keandalan infrastruktur yang digunakan saat ini. Keterbatasan ini berdampak langsung terhadap kinerja sistem, khususnya dalam konteks pengembangan layanan digital yang semakin kompleks dan membutuhkan dukungan infrastruktur yang lebih adaptif dan dinamis. Pertanyaan kedua berkaitan dengan bagaimana meningkatkan fleksibilitas infrastruktur TI agar mampu menyesuaikan kapasitas secara cepat dan efisien, sehingga dapat mencegah terjadinya downtime dan memastikan kelancaran operasional akademik maupun administratif di lingkungan kampus. Berdasarkan pertanyaan-pertanyaan tersebut, penelitian ini bertujuan untuk menyusun solusi yang dapat mengatasi permasalahan tersebut melalui penerapan teknologi private cloud berbasis OpenStack. OpenStack dipilih karena merupakan platform open source yang mendukung penerapan model Infrastructure as a Service (IaaS), yang mampu menyediakan infrastruktur TI yang

skalabel, fleksibel, dan dapat diandalkan untuk memenuhi kebutuhan layanan aplikasi di lingkungan Universitas Djuanda Bogor [7], [8].

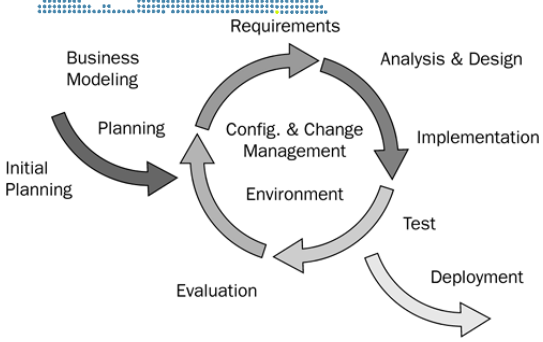
Penelitian ini memiliki ruang lingkup yang dibatasi pada perancangan infrastruktur private cloud berbasis IaaS dengan menggunakan platform OpenStack versi Zed. Fokus utama diarahkan pada penggunaan OpenStack sebagai platform virtualisasi yang mampu menggantikan sistem sebelumnya yang masih bergantung pada teknologi virtualisasi konvensional seperti Oracle *VirtualBox*. Dalam konteks ini, pembahasan penelitian difokuskan pada pengembangan solusi IaaS yang difungsikan sebagai wadah virtualisasi untuk menjalankan berbagai aplikasi kampus, baik untuk keperluan akademik maupun operasional internal. Tujuan utama dari penelitian ini adalah untuk menerapkan OpenStack sebagai solusi strategis dalam membangun infrastruktur private cloud yang lebih efisien, skalabel, dan fleksibel. Di samping itu, penelitian ini juga bertujuan untuk mengoptimalkan alokasi sumber daya melalui pemanfaatan fitur manajemen yang tersedia dalam OpenStack, sehingga visibilitas dan kontrol terhadap penggunaan sumber daya dapat ditingkatkan. Penggunaan alat-alat otomatisasi dan orkestrasi yang terintegrasi dalam platform ini diharapkan dapat mengurangi beban kerja manual tim IT, mempercepat proses deployment mesin virtual, serta meningkatkan efisiensi operasional secara keseluruhan. Dengan demikian, hasil penelitian ini diharapkan dapat menjadi kontribusi nyata dalam pengembangan sistem infrastruktur TI yang tangguh dan adaptif di lingkungan Universitas Djuanda, sekaligus sebagai referensi bagi institusi pendidikan lainnya yang ingin mengadopsi solusi serupa.

## 2. Metodologi Penelitian

Penelitian ini menggunakan metode Rekayasa Perangkat Lunak (RPL) dengan pendekatan model Rational Unified Process (RUP) untuk merancang dan membangun infrastruktur private cloud berbasis Infrastructure as a Service (IaaS) menggunakan platform OpenStack Zed [9], [10]. Model RUP dipilih karena mampu memberikan pendekatan iteratif dan terstruktur dalam proses pengembangan sistem, terutama untuk proyek berskala besar yang membutuhkan dokumentasi lengkap serta proses validasi yang sistematis. Penelitian dilakukan di Universitas Djuanda Bogor sebagai studi kasus, dengan tujuan menyediakan layanan cloud internal berbasis web yang dapat dimanfaatkan oleh civitas akademika secara efisien dan terpusat.

Model RUP terdiri dari empat fase utama, yaitu: Inception, Elaboration, Construction, dan Transition seperti terlihat pada Gambar 1. Pada fase Inception, dilakukan identifikasi kebutuhan dasar sistem dan studi kelayakan untuk memastikan bahwa pengembangan infrastruktur cloud sesuai dengan kebutuhan institusi. Fase Elaboration difokuskan pada perancangan arsitektur sistem dan penyusunan use case, termasuk pemetaan komponen OpenStack seperti Nova, Neutron, Glance, dan Horizon dalam infrastruktur cloud. Fase Construction meliputi implementasi dan konfigurasi OpenStack Zed di lingkungan virtual menggunakan server lokal dan jaringan internal kampus, serta pengujian fungsionalitas layanan IaaS seperti pembuatan instance, alokasi IP, dan manajemen storage. Terakhir, fase Transition mencakup penyebaran sistem kepada pengguna akhir, pelatihan, dan dokumentasi teknis [11], [12], [13].

Seluruh tahapan didukung oleh pemanfaatan teknologi berbasis web untuk mengakses layanan cloud melalui dashboard Horizon. Pengujian sistem dilakukan menggunakan metode black box testing untuk memastikan setiap fungsi berjalan sesuai skenario yang telah dirancang. Hasil dari penelitian ini diharapkan dapat menjadi solusi implementatif dalam pengembangan infrastruktur IT kampus berbasis cloud yang mandiri dan scalable [14].




**Gambar 1.** Bagan Model Rational Unified Process (RUP)

**2.1. Karakteristik Utama dari Model Rational Unified Process (RUP):**

Model RUP memiliki sejumlah ciri khas yang membedakannya dalam pengembangan perangkat lunak:

a) Berbasis Objek: Pendekatan RUP menitikberatkan pada pemodelan sistem sebagai kumpulan objek yang saling berinteraksi, mengikuti paradigma pemrograman berorientasi objek.
b) Pendekatan Iteratif dan Inkremental: Proses pengembangan dalam RUP dilakukan secara bertahap dan berulang. Setiap siklus menghasilkan tambahan fitur baru yang memperluas fungsionalitas sistem secara progresif.
c) Tahapan Pengembangan: RUP terbagi ke dalam empat fase utama, yaitu:
    1. *Inception (Permulaan):* Menentukan visi proyek, tujuan, dan batasannya.
    2. *Elaboration (Pengembangan Awal):* Melakukan analisis kebutuhan dan desain sistem secara mendalam.
    3. *Construction (Pembangunan):* Menyusun kode dan mengintegrasikan komponen sistem.
    4. *Transition (Penerapan):* Menyerahkan sistem yang telah dibangun kepada pengguna akhir.
d) Disiplin dan Produk Kerja: Dalam RUP, proses diklasifikasikan ke dalam sejumlah disiplin seperti analisis bisnis, manajemen proyek, pengujian, serta pemeliharaan. Setiap disiplin menghasilkan dokumen atau model yang disebut artefak.
e) Fokus pada Arsitektur: RUP memberikan perhatian besar terhadap rancangan arsitektur perangkat lunak. Struktur sistem dibangun sejak awal dan dikembangkan secara berkelanjutan sepanjang proyek.
f) Manajemen Risiko: RUP memungkinkan identifikasi dan mitigasi risiko sejak fase awal pengembangan guna meningkatkan keberhasilan proyek.
g) Dokumentasi Lengkap: Proses ini menghasilkan dokumentasi yang komprehensif, mencakup kebutuhan sistem, desain teknis, serta hasil pengujian.

Model RUP dapat disesuaikan dengan kebutuhan setiap proyek, dan telah menjadi fondasi penting dalam metodologi pengembangan berbasis objek modern.

**2.2. Kelebihan RUP:**

a) Memberikan akses mudah ke pengetahuan inti bagi tim pengembang.
b) Memberikan panduan penggunaan UML secara optimal.
c) Mendukung pendekatan iteratif dalam pengembangan.
d) Fleksibel dalam menerima modifikasi proses.
e) Memfasilitasi kontrol perubahan sistem selama pengembangan.
f) Mendukung pelaksanaan *test case* menggunakan Rational TestManager.

**2.3.Kekurangan RUP:**

Metode ini lebih cocok diterapkan pada pengembangan perangkat lunak yang berbasis objek dan terintegrasi dengan pemodelan UML.

# 3. Hasil dan Pembahasan

## 3.1. Planning and Requirement

Dalam penerapan penggunaan sistem yang dibuat, digunakan perangkat keras dan perangkat lunak antara lain :

1. Perangkat Keras

   Hardware merupakan sarana fisik untuk menghasilkan data, program dan keluaran. Spesifikasi perangkat keras yang digunakan sebagai pendukung server dalam menjalankan aplikasi ini yaitu :

   a. *Server on-premise*: minimal 2 node, node pertama sebagai *controller node* dan node kedua sebagai *compute node*.
   b. *Procesessor*: minimal *processor* dengan CPU 8 core.
   c. RAM: minimal RAM dengan ukuran 16 GB.
   d. *Storage*: minimal 512 GB
   e. *Networking*: 2 *Interface* NIC, NIC Pertama sebagai komunikasi antar komponen *OpenStack* dan akses *Dashboard OpenStack*, NIC kedua sebagai *bridge Openvswitch*.

2. Perangkat lunak

   Dalam pengembangan aplikasi ini menggunakan perangkat lunak dengan spesifikasi sebagai berikut:

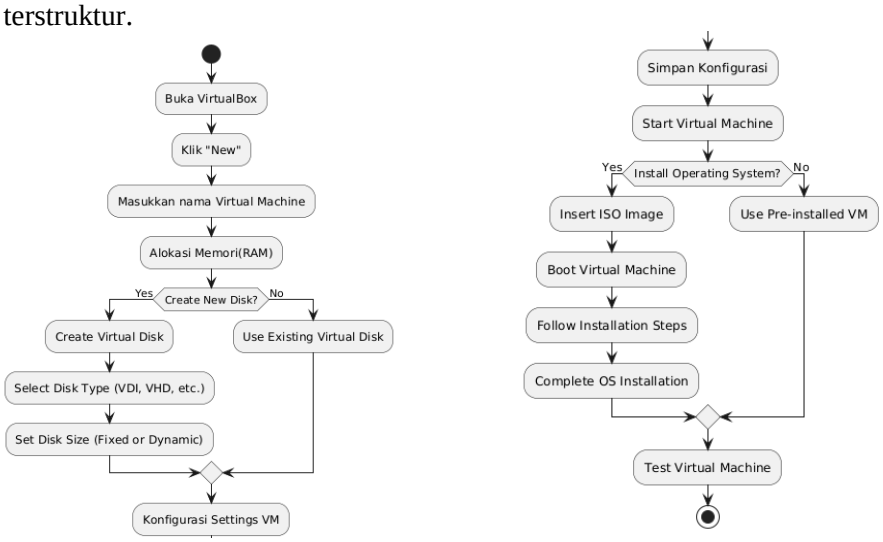
   a. Sistem Operasi: *Linux* berbasis *CentOS* Stream 8 sebagai platform utama.
   b. *Hypervisor*: KVM (*Kernel-based Virtual machine*) untuk virtualisasi.
   c. *OpenStack* sebagai platform utama untuk manajemen sumber daya *cloud*.

3. Komponen OpenStack:

   a. *Controller Node*: Mengelola layanan inti OpenStack, seperti Nova (*Compute*), *Neutron* (*Networking*), Glance (*Image Service*), dan Keystone (*Identity Service*).
   b. *Compute Node*: Bertanggung jawab untuk menjalankan *Virtual machine*.
   c. *Storage*: Menggunakan *Cinder* (*Block Storage*) untuk penyimpanan data instance.
   d. *Networking*: Menggunakan Neutron untuk koneksi jaringan *Virtual machine*.

## 3.2. *Analisys and Design*

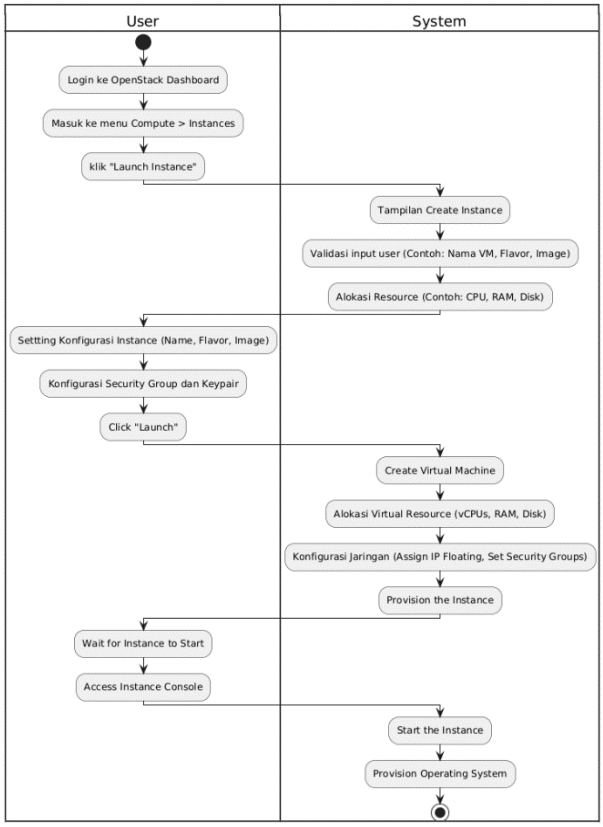
Infrastruktur virtualisasi yang berjalan saat ini memanfaatkan *VirtualBox* sebagai platform utama untuk menyediakan lingkungan virtual. *VirtualBox* digunakan untuk menjalankan beberapa *virtual machine* (VM) yang berfungsi sebagai server aplikasi dan database. Berikut adalah Analisa sistem berjalan proses pembuatan *virtual machine* di *VirtualBox*. Gambar 2 menjelaskan aktivitas pengguna dalam membuat *virtual machine* secara terstruktur.




**Gambar 2.** Analisa Sistem Berjalan *Create VM di VirtualBox*

Sebagai solusi alternatif untuk menggantikan infrastruktur *VirtualBox*, maka penulis mencoba untuk merancang bangun infrastruktur Private Cloud menggunakan OpenStack. OpenStack merupakan pilihan yang lebih sesuai untuk membangun infrastruktur private cloud berbasis Infrastructure as a Service (IaaS). OpenStack adalah platform open-source yang menyediakan berbagai layanan untuk membangun dan mengelola infrastruktur cloud, dan sangat cocok untuk organisasi yang membutuhkan skalabilitas, manajemen terpusat, serta pengelolaan sumber daya cloud secara otomatis.

Berdasarkan Sistem yang berjalan, penulis dapat membuat sistem usulan yang baru untuk mempercepat proses provisioning *Virtual machine* tanpa harus menginstal Sistem Operasi terlebih dahulu, terlihat pada Gambar 3.

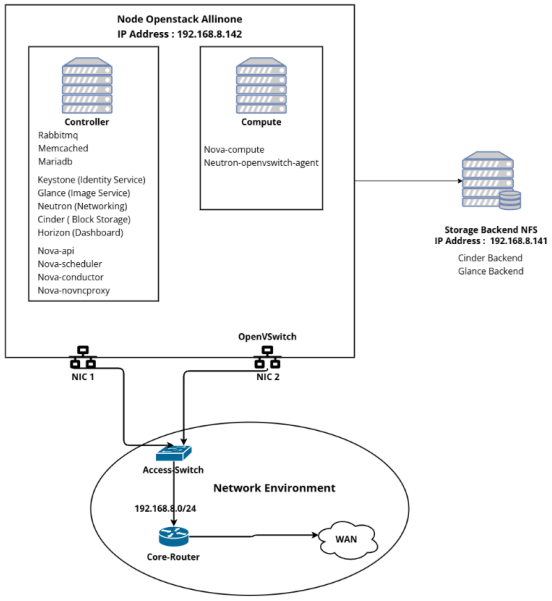



**Gambar 3.** Analisa Sistem Usulan *Create Virtual machine* di *OpenStack*

Arsitektur infrastruktur cloud yang dirancang menggunakan OpenStack Zed mencakup beberapa komponen utama yang saling terintegrasi untuk menyediakan layanan Infrastructure as a Service (IaaS). Komponen utama dalam arsitektur ini meliputi Nova (Compute), yang bertanggung jawab dalam penyediaan dan pengelolaan *virtual machine* (VM), Cinder (Block Storage) untuk pengelolaan penyimpanan *virtual machine*, Glance (Image Service) untuk pengelolaan image VM, Neutron (Networking) bertugas untuk pengaturan jaringan virtual, Keystone (Identity Service) menangani manajemen identitas dan otorisasi pengguna, dan Horizon (Dashboard) berfungsi sebagai antarmuka grafis untuk memudahkan pengguna dan administrator dalam mengelola infrastruktur.

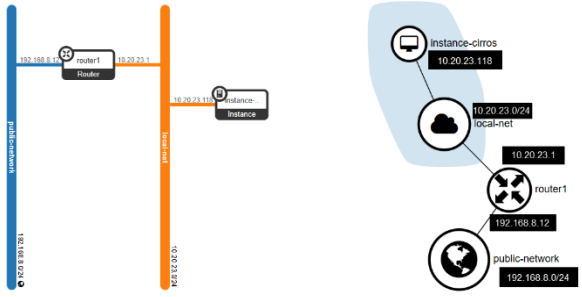
Perancangan topologi OpenStack pada skenario ini menggunakan dua node utama, yaitu satu node all-in-one dan satu node NFS server seperti terlihat pada Gambar 4. Node all-in-one bertindak sebagai pusat kendali yang mengintegrasikan seluruh layanan OpenStack, termasuk keystone, nova, neutron, glance, cinder, dan horizon, dalam satu server. Node ini juga berfungsi sebagai compute node untuk menjalankan instance. Sementara itu, node NFS server digunakan secara khusus sebagai backend penyimpanan

untuk layanan glance dan cinder. Layanan glance akan memanfaatkan NFS server untuk menyimpan image instance, sedangkan cinder menggunakan NFS server untuk menyediakan volume block storage yang dapat digunakan oleh instance OpenStack. Kedua node ini dihubungkan melalui jaringan internal menggunakan management network untuk komunikasi antar layanan, serta public network untuk akses eksternal ke instance dan dashboard. Topologi ini dirancang untuk memaksimalkan efisiensi penggunaan sumber daya sekaligus menyediakan backend penyimpanan yang terpusat dan dapat diakses dengan cepat oleh layanan OpenStack.




**Gambar 4.** Topologi *OpenStack*

Perancangan topologi jaringan (Neutron) dalam environment OpenStack pada skenario ini menggunakan konfigurasi sederhana dengan satu jaringan lokal (local-net), floating IP untuk akses eksternal atau publik, dan sebuah router untuk menghubungkan local-net dengan public-net. Local-net berfungsi sebagai jaringan internal di mana instance-instance OpenStack berkomunikasi satu sama lain secara privat. Public-net, di sisi lain, digunakan untuk menyediakan akses ke jaringan eksternal atau internet. Router Neutron berperan sebagai penghubung antara local-net dan public-net, memungkinkan instance di jaringan lokal untuk mengakses jaringan eksternal dengan bantuan NAT (Network Address Translation). Untuk akses publik, setiap instance yang membutuhkan konektivitas eksternal akan diberikan floating IP dari subnet public-net, terlihat pada Gambar 5. Topologi ini dirancang untuk kesederhanaan sekaligus menjaga fleksibilitas dan keamanan komunikasi antar jaringan.

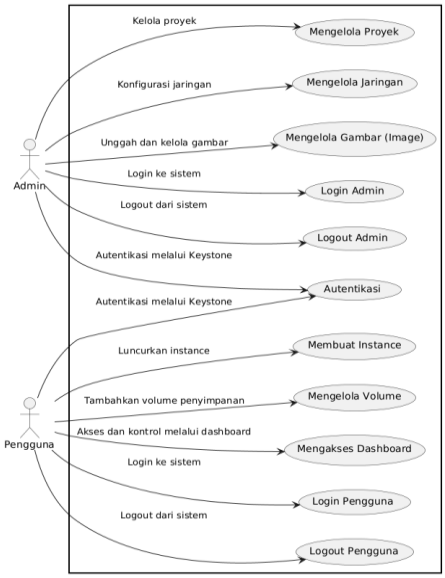



**Gambar 5.** Topologi Jaringan *(Neutron) Instance OpenStack*

Perancangan Unified Modeling Language (UML) bertujuan untuk mengGambarkan secara visual arsitektur, proses, dan interaksi antar komponen di dalam platform OpenStack. Rancangan UML diantaranya : *Use case diagram*, activity diagram, *Sequence Diagram*, class diagram.

**1. *Use case diagram***

*Use case diagram* mengGambarkan interaksi antara aktor (pengguna atau sistem eksternal) dengan sistem yang sedang dianalisis atau dikembangkan. Diagram ini membantu untuk mengGambarkan fungsionalitas sistem dari sudut pandang pengguna, mengGambarkan apa yang dapat dilakukan oleh pengguna dan sistem untuk memenuhi tujuan tertentu.

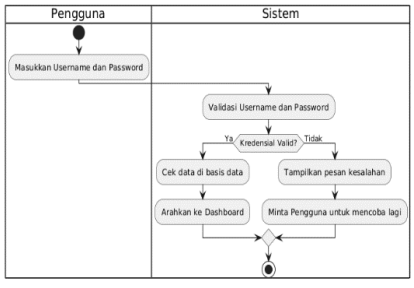



**Gambar 6.** *Use case diagram* Aplikasi *OpenStack*

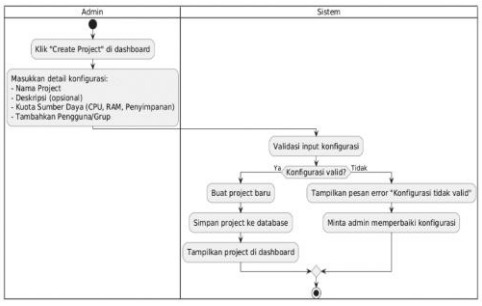
Gambar 6 menampilkan *Use case diagram* yang terdapat pada sistem OpenStack ini. Secara default, pada platform OpenStack terdapat dua peran yaitu admin dan user. Masing-masing dari dua peran ini, memiliki fungsi yang berbeda sesuai dengan kebutuhan sistem.

**2. *Activity Diagram***

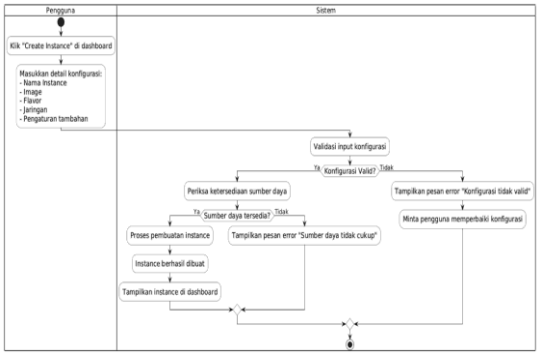
*Activity diagram* adalah *diagram* yang menjelaskan alur proses sistem secara prosedural. Ini memungkinkan untuk mengevaluasi kemungkinan adanya lebih dari satu jalur yang terbentuk dan berjalan secara bersamaan. PengGambaran aktivitas diagram dimulai dari *node* awal dan berakhir di *node* akhir.



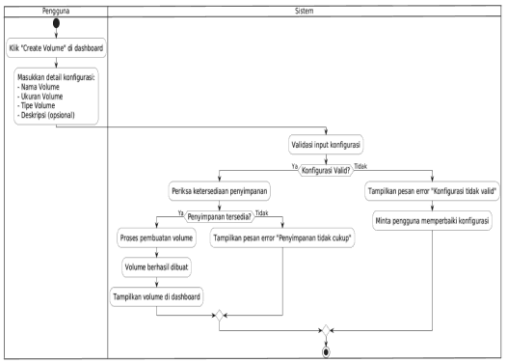

**Gambar 7.** *Activity Diagram Login*



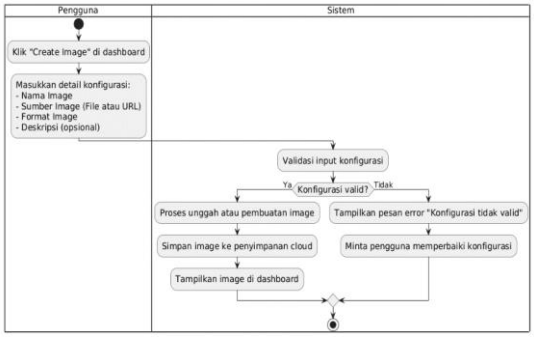

**Gambar 8.** *Activity Diagram* membuat *Project*



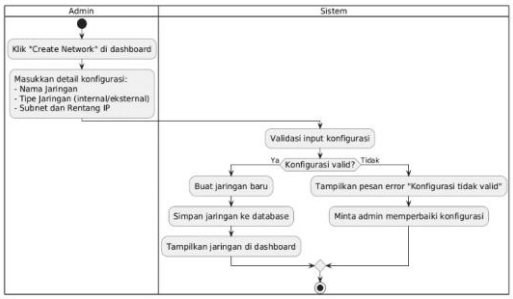

**Gambar 9.** *Activity Diagram* membuat *Instance*




**Gambar 10.** *Activity Diagram* membuat *Instance*



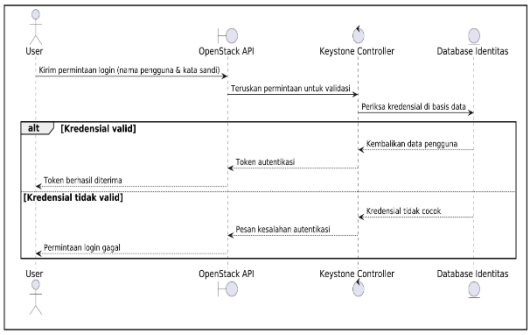

**Gambar 11.** *Activity Diagram* membuat *Image*

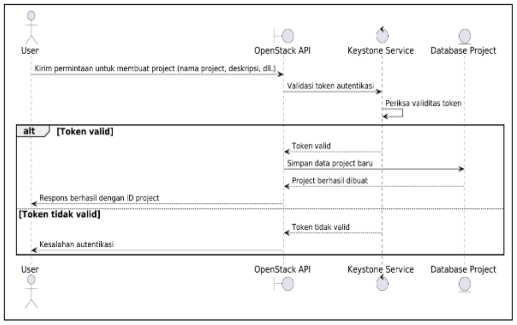


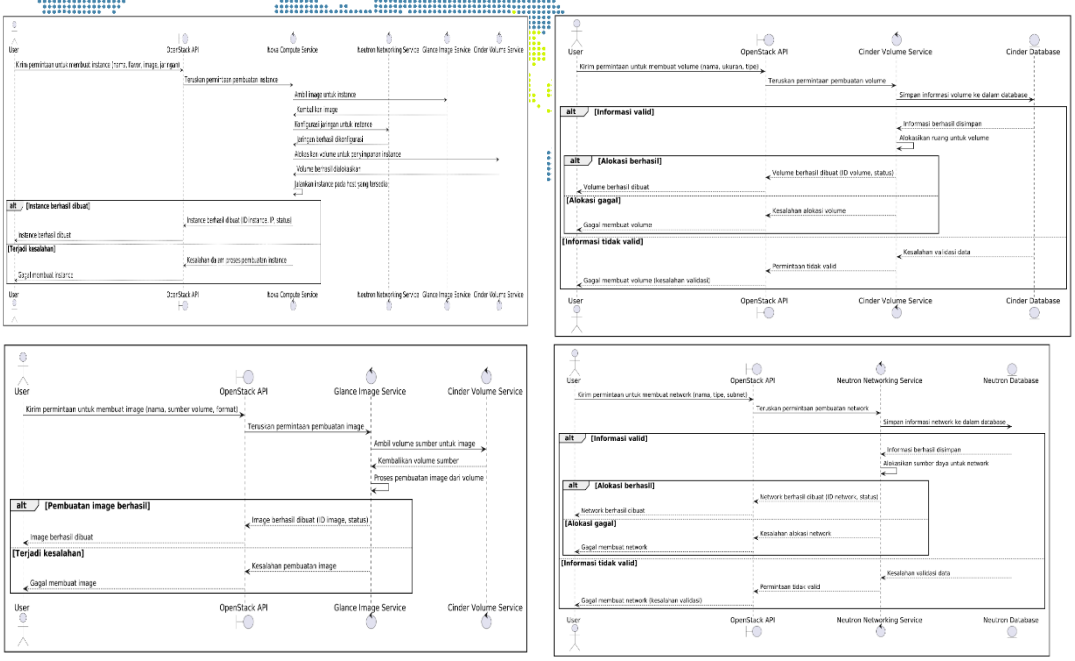

**Gambar 12.** *Activity Diagram* membuat *Network*

**2. *Sequence Diagram***

*Sequence Diagram* adalah diagram yang memvisualisasikan bagaimana objek-objek dalam sistem berinteraksi satu sama lain melalui pesan dalam urutan waktu tertentu. Diagram ini berfokus pada urutan kronologis dari komunikasi antara objek-objek tersebut, termasuk pengguna (actor), komponen sistem, dan subsistem lainnya. Gambar 13 menjelaskan *Sequence Diagram* pada platform OpenStack yang terdiri dari *Sequence Diagram Login*, *Sequence Diagram* membuat *Project*, *Sequence Diagram* membuat *Instance*, *Sequence Diagram membuat Volume*, *Sequence Diagram* membuat *Image dan Sequence Diagram* membuat *Network.*



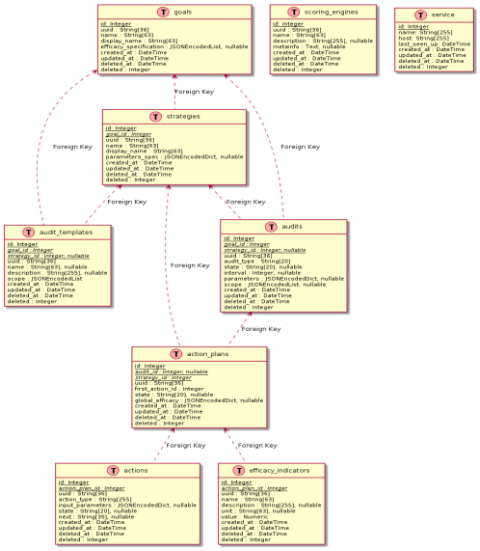






**Gambar 13** *Sequence Diagram* Aplikasi**.**

### 3. *Class Diagram*

*Class Diagram* adalah diagram yang merepresentasikan struktur statis dari sistem yang mewakili objek-objek dalam sistem dan hubungan pada setiap objek. Diagram ini digunakan untuk memodelkan tampilan struktur sistem yang mencakup kemampuan untuk membawa data dan menjalankan tindakan, serta sebagai dasar untuk pengembangan sistem perangkat lunak lebih lanjut, terlihat pada Gambar 14.




**Gambar 14.** *Class Diagram*

### 3.3. Implementation

Antarmuka (*User Interface*) adalah bagian dari sistem atau perangkat lunak yang berfungsi sebagai perantara antara pengguna dengan sistem tersebut, memungkinkan pengguna untuk berinteraksi dan memberikan perintah kepada sistem secara mudah dan intuitif. Berikut adalah tampilan antarmuka (*user interface*) pada *platform private cloud OpenStack.*

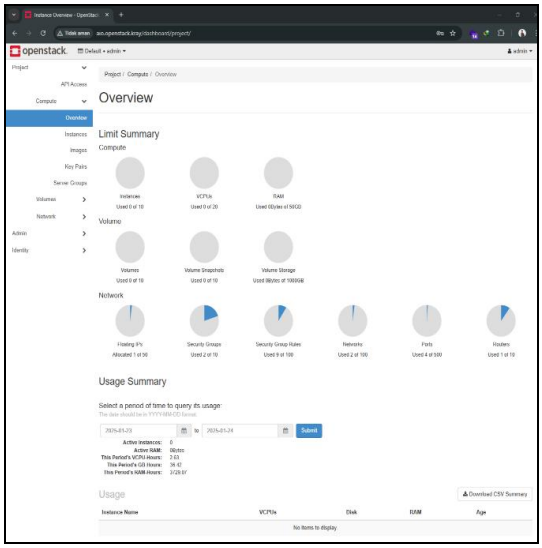
**a. Halaman *Login***

Gambar 15 menjelaskan halaman awal pada *Platform OpenStack*. Halaman *login* di gunakan user untuk *login* ke dalam *Dashboard* dari sistem *OpenStack*. Pengguna admin atau *member* diharuskan untuk menginput *username* dan *password* yang sudah terdaftar di aplikasi. Halaman *login* pada *OpenStack* memastikan pengguna dapat mengakses *platform OpenStack* dengan cepat dan aman.


**Gambar 15.** Tampilan Halaman *Login*

**b. Halaman *Dashboard***

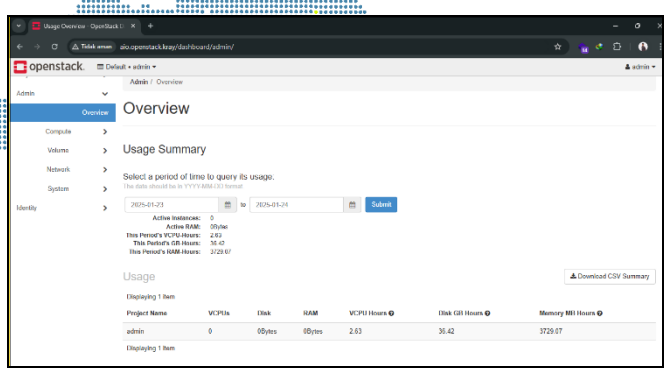
Gambar 16 menjelaskan antarmuka untuk mengelola infrastruktur *cloud* dengan berbagai fitur. Pada bagian *header*, terdapat menu untuk mengelola akun, dan informasi project yang sedang di gunakan. Di sisi kiri, menu navigasi utama mencakup *Overview* untuk melihat ringkasan penggunaan sumber daya, *Compute* untuk mengelola *instance virtual*, *image*, dan *key pair*, *Network* untuk mengatur topologi jaringan, *router*, dan *floating IP*, serta *Volumes* untuk mengelola *storage block*, *Backups* dan *Snapshots*. Area utama menampilkan detail sesuai menu yang dipilih, seperti tabel *instance* atau grafik sumber daya.




**Gambar 16.** Halaman *Dashboard*

**c. Halaman *Admin***

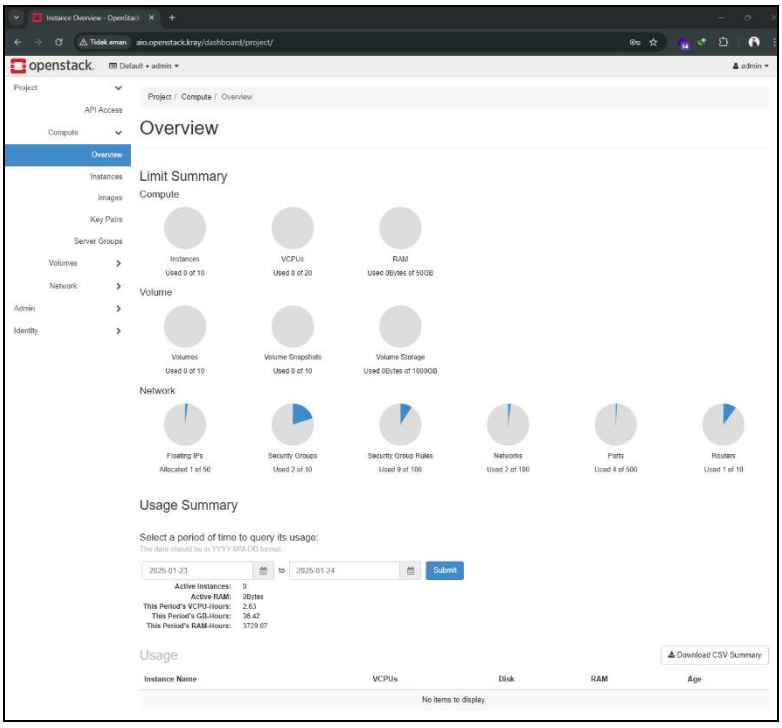
Halaman *Admin* pada *dashboard OpenStack* memberikan akses kepada pengguna dengan hak admin untuk mengelola, memantau, dan melakukan konfigurasi terhadap infrastruktur *cloud* secara menyeluruh terlihat pada Gambar 17.




**Gambar 17.** Halaman *Admin*

**d. Halaman *Project***

Halaman *Project* pada *dashboard OpenStack* memungkinkan pengguna untuk mengelola sumber daya dalam proyek, termasuk *instance*, *volume*, *image*, dan jaringan. Halaman ini memberikan kontrol penuh untuk memastikan proyek berjalan dengan efisien sesuai kebutuhan infrastruktur *cloud* terlihat pada Gambar 18.




**Gambar 18.** Halaman *Project*

### 3.4. Pengujian Sistem

Pengujian sistem adalah proses yang dilakukan untuk mengevaluasi kinerja dan kesesuaian sistem secara keseluruhan. Proses ini memastikan bahwa semua komponen sistem bekerja sama dengan baik dan sesuai dengan spesifikasi dan kebutuhan. Setelah perangkat lunak diintegrasikan dan siap digunakan secara bersamaan, pengujian sistem dilakukan. Proses ini menguji fungsi utama, kinerja, keandalan, keamanan, dan kompatibilitas sistem dalam berbagai kondisi operasional. Pengujian sistem membantu menemukan dan memperbaiki kesalahan yang mungkin tidak terlihat pada tahap pengembangan sebelumnya dan memastikan bahwa sistem berjalan dengan baik dalam situasi nyata tanpa gangguan, terlihat pada Tabel 1.

**Tabel 1.** Pengujian dengan *Blackbox Testing*

| **Pengujian Data Benar** | | | |
|---|---|---|---|
| **Data yang dimasukkan** | **Data yang diharapkan** | **Pengamatan** | **Kesimpulan** |
| *Username* : (*username*_benar)<br>*Password* : (*password*_benar) | *Login* berhasil, masuk ke *dashboard OpenStack* | Sistem menampilkan *dashboard* utama tanpa pesan error | sesuai |
| Pada menu *instance* klik *create instance*, pengguna diminta untuk mengisi parameter untuk membuat *instance* | Instance berhasil dibuat dan berjalan dalam status *active* | Sistem menampilkan status *active* tanpa *error* | sesuai |
| Klik pada *instance* yang sudah dibuat, lalu *edit* data *instance* | Menampilkan *instance* yang sudah di *edit* | Sistem menampilkan status *active* tanpa *error* | sesuai |
| Klik pada *instance* yang sudah dibuat, lalu *delete instance* | Menampilkan *instance* yang sudah di hapus | Sistem menampilkan halaman dengan *instance* yang sudah terhapus | sesuai |
| Pada menu *image*, pengguna diminta untuk mengisi parameter untuk membuat *image* | Image berhasil dibuat dalam status *active* | Sistem menampilkan status *active* tanpa *error* | sesuai |
| Klik pada *image* yang sudah dibuat, lalu *edit image* | Menampilkan *image* yang sudah di *edit* | Sistem menampilkan status *active* tanpa *error* | sesuai |
| Klik pada *image* yang sudah dibuat, lalu *delete image* | Menampilkan *image* yang sudah dihapus | Sistem menampilkan halaman dengan *image* yang sudah terhapus | sesuai |
| Pada menu *image*, pengguna diminta untuk mengisi parameter untuk membuat *image* | Image berhasil dibuat dalam status *active* | Sistem menampilkan status *active* tanpa *error* | sesuai |
| Klik pada *image* yang sudah dibuat, lalu *edit image* | Menampilkan *image* yang sudah di *edit* | Sistem menampilkan status *active* tanpa *error* | sesuai |
| Klik pada *image* yang sudah dibuat, lalu *delete image* | Menampilkan *image* yang sudah dihapus | Sistem menampilkan halaman dengan *image* yang sudah terhapus | sesuai |
| Pada menu *Network*, pengguna diminta untuk mengisi parameter untuk membuat n*etwork* | *Network* berhasil dibuat dalam status *active* | Sistem menampilkan status *active* tanpa *error* | sesuai |
| Klik pada *network* yang sudah dibuat, lalu *edit network* | Menampilkan *network* yang sudah di *edit* | Sistem menampilkan status *active* tanpa *error* | sesuai |
| Klik pada *network* yang sudah dibuat, lalu *delete network* | Menampilkan *network* yang sudah dihapus | Sistem menampilkan halaman dengan *network* yang sudah terhapus | sesuai |
| **Pengujian Data Salah** | | | |
| *Username* : (*username*_benar)<br>*Password* : (*password*_salah) | Gagal *login*, pesan *error*: "*Invalid credentials*" | Sistem menampilkan pesan *error* sesuai | sesuai |
| Pada menu *instance* parameter yang diisi tidak sesuai atau tidak di isi | Menampilkan pesan *error* | Sistem menampilkan *button launch instance* tidak dapat di klik | sesuai |
| Pada menu *image* parameter yang diisi tidak sesuai atau tidak di isi | Menampilkan pesan error | Sistem menampilkan *button create image* tidak dapat di klik | sesuai |
| Pada *volume* parameter *size volume* tidak di isi | Menampilkan pesan *error* | Sistem menampilkan pesan *error* pada kolom input *size volume* | sesuai |
| Pada menu *network* parameter yang diisi tidak sesuai atau tidak di isi | Menampilkan pesan *error* | Sistem menampilkan pesan error "*This field is required*" pada kolom input | Pada menu *network* parameter yang diisi tidak sesuai atau tidak di isi |

Berdasarkan hasil pengujian *blackbox* terhadap sistem infrastruktur private cloud berbasis OpenStack, dapat disimpulkan bahwa sistem berjalan sesuai dengan fungsinya. Pengujian dengan data yang benar menunjukkan bahwa fitur utama seperti login, pembuatan, pengeditan, dan penghapusan *instance*, *image*, dan *network* dapat digunakan dengan baik tanpa munculnya kesalahan. Setiap aksi yang dilakukan menghasilkan respons yang sesuai, seperti status *active* atau penghapusan objek secara langsung.

Sementara itu, pengujian dengan data yang salah menunjukkan bahwa sistem mampu memberikan validasi dan pesan kesalahan yang tepat. Sistem berhasil menolak login dengan kredensial yang salah serta menampilkan notifikasi kesalahan. Pada input yang tidak lengkap atau salah, seperti di menu *instance*, *image*, *volume*, dan *network*, sistem mencegah proses berlanjut dan menampilkan peringatan yang sesuai. Hal ini membuktikan bahwa sistem memiliki mekanisme validasi yang baik untuk mencegah

kesalahan pengguna. Secara keseluruhan, sistem telah memenuhi aspek keandalan dan ketepatan fungsi, serta siap digunakan untuk mendukung layanan dan operasional di lingkungan Universitas Djuanda Bogor.

## 4. Kesimpulan

Penelitian ini berhasil merancang dan membangun infrastruktur private cloud berbasis Infrastructure as a Service (IaaS) menggunakan platform OpenStack Zed, dengan pendekatan pengembangan perangkat lunak Rational Unified Process (RUP) berbasis web. Studi kasus yang diangkat, yaitu Universitas Djuanda Bogor, menjadi representasi institusi pendidikan tinggi yang membutuhkan solusi infrastruktur TI yang efisien, terukur, dan fleksibel. Melalui pendekatan RUP yang terdiri dari empat tahapan utama — inception, elaboration, construction, dan transition — seluruh proses perancangan, pembangunan, hingga implementasi sistem dilakukan secara sistematis dan terstruktur.

Penerapan OpenStack Zed terbukti mampu menyediakan layanan virtualisasi sumber daya komputasi secara optimal, termasuk pengelolaan instansi *virtual machine*, jaringan, dan penyimpanan data. Sistem ini memungkinkan administrator universitas untuk melakukan deployment layanan TI dengan lebih cepat dan efisien. Selain itu, antarmuka berbasis web yang dikembangkan memudahkan pengguna dalam melakukan konfigurasi dan pengelolaan layanan cloud tanpa memerlukan interaksi langsung dengan command line interface (CLI).

Sebagai saran, pengembangan sistem dapat terus ditingkatkan dengan menambahkan fitur monitoring real-time, pengamanan berlapis melalui autentikasi dua faktor (2FA), serta integrasi dengan sistem informasi akademik yang sudah berjalan di institusi. Selain itu, pelatihan teknis untuk tim TI internal sangat disarankan guna memastikan keberlanjutan pemeliharaan dan pengembangan sistem cloud yang telah dibangun. Ke depan, model penerapan ini dapat direplikasi oleh institusi lain yang memiliki kebutuhan serupa dalam pengembangan infrastruktur digital mandiri berbasis cloud. Dengan demikian, hasil penelitian ini tidak hanya bermanfaat secara lokal, namun juga berpotensi untuk diterapkan secara lebih luas dalam dunia pendidikan tinggi di Indonesia.

## Daftar Pustaka

[1] P. Krishnan, K. Jain, A. Aldweesh, P. Prabu, and R. Buyya, “OpenStackDP: a scalable network security framework for SDN-based OpenStack cloud infrastructure,” *J. Cloud Comput.*, vol. 12, no. 1, p. 26, Feb. 2023, doi: 10.1186/s13677-023-00406-w.

[2] T. Rosado and J. Bernardino, “An overview of openstack architecture,” in *Proceedings of the 18th International Database Engineering & Applications Symposium*, in IDEAS ’14. New York, NY, USA: Association for Computing Machinery, Jul. 2014, pp. 366–367. doi: 10.1145/2628194.2628195.

[3] “OpenStackDP: a scalable network security framework for SDN-based OpenStack cloud infrastructure | Journal of Cloud Computing | Full Text.” Accessed: Apr. 14, 2025. [Online]. Available: https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-023-00406-w

[4] O. Khedher and C. D. Chowdhury, *Mastering OpenStack*. Packt Publishing Ltd, 2017.

[5] A. Shrivastwa, S. Sarat, K. Jackson, C. Bunch, E. Sigler, and T. Campbell, *OpenStack: Building a Cloud Environment*. Packt Publishing Ltd, 2016.

[6] “Penerapan Model Simulasi Oracle *VirtualBox* Pada Kompetensi Sistem Operasi Di Smk Hidayah Semarang | Multimatrix.” Accessed: Apr. 14, 2025. [Online]. Available: https://jurnal.unw.ac.id/index.php/mm/article/view/273

[7] S. Bhardwaj, L. Jain, and S. Jain, "Cloud Computing: A Study Of Infrastructure As A Service (Iaas)," 2010.
[8] M. Santana, "Infrastructure as a Service (IaaS)," in *Cloud Computing Security*, 2nd ed., CRC Press, 2020.
[9] N. Dwivedi, D. Katiyar, and G. Goel, "A Comparative Study of Various Software Development Life Cycle (SDLC) Models," *Int. J. Res. Eng. Sci. Manag.*, vol. 5, no. 3, Art. no. 3, Mar. 2022.
[10] Y. I. Chandra, D. R. Irawati, and K. Rokoyah, "Rancang Bangun Aplikasi Pola Asuh Orang Tua Terhadap Anak Menggunakan Model Big Bang Berbasis Android," *Ikraith-Inform.*, vol. 6, no. 3, Art. no. 3, Nov. 2022, doi: 10.37817/ikraith-informatika.v6i3.2203.
[11] L. V. Manzoni and R. T. Price, "Identifying extensions required by RUP (rational unified process) to comply with CMM (capability maturity model) levels 2 and 3," *IEEE Trans. Softw. Eng.*, vol. 29, no. 2, pp. 181–192, Feb. 2003, doi: 10.1109/TSE.2003.1178058.
[12] H. Mohd *et al.*, "A secured e-tendering model based on rational unified process (RUP) approach: inception and elaboration phases," *Int. J. Supply Chain Manag. IJSCM*, vol. 5, no. 4, Art. no. 4, Dec. 2016.
[13] T. K. Tia, I. Nuryasin, and M. Maskur, "Model Simulasi Rational Unified Process (RUP) Pada Pegembagan Perangkat Lunak," *J. Repos.*, vol. 2, no. 4, Art. no. 4, 2020, doi: 10.22219/repositor.v2i4.30511.
[14] M. Sudarma, S. Ariyani, and P. A. Wicaksana, "Implementation of the Rational Unified Process (RUP) Model in Design Planning of Sales Order Management System," *INTENSIF J. Ilm. Penelit. Dan Penerapan Teknol. Sist. Inf.*, vol. 5, no. 2, Art. no. 2, Aug. 2021, doi: 10.29407/intensif.v5i2.15543.