INTEGRASI KUBERNETES DENGAN LOAD BALANCING DAN SKALABILITAS DALAM PENGEMBANGAN APLIKASI MANAJEMEN LINGKUNGAN BERBASIS ANDROID UNTUK PERUMAHAN KOMERSIL

SKRIPSI

Nama: Feri Winarta
NIM : 20201000041

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS BUDDHI DHARMA
TANGERANG
2025

INTEGRASI KUBERNETES DENGAN LOAD BALANCING DAN SKALABILITAS DALAM PENGEMBANGAN APLIKASI MANAJEMEN LINGKUNGAN BERBASIS ANDROID UNTUK PERUMAHAN KOMERSIL

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh Gelar Sarjana pada Program Studi Teknik Informatika

Nama: Feri Winarta

NIM : 20201000041

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS BUDDHI DHARMA

TANGERANG

2025

UNIVERSITAS BUDDHI DHARMA

LEMBAR PERNYATAAN KEASLIAN SKRIPSI

Yang bertanda tangan di bawah ini:

NIM : 20201000041
Nama : Feri Winarta
Jenjang Studi : Sastra 1
Program Studi : Teknik Informatika
Peminatan : Infrastructure & Security

Dengan ini saya menyatakan bahwa:

1. Skripsi ini adalah asli dan belum pernah diajukan untuk mendapat gelar akademik Sarjana atau kelengkapan studi, baik di Universitas Buddhi Dharma maupun di Perguruan Tinggi lainnya.
2. Skripsi ini saya buat sendiri tanpa bantuan dari pihak lain, kecuali arahan dosen pembimbing.
3. Dalam Skripsi ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dan dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan daftar pustaka.
4. Dalam Skripsi ini tidak terdapat pemalsuan (kebohongan). Seperti buku, artikel, jurnal, data sekunder, pengolahan data, dan pemalsuan tanda tangan dosen atau Ketua Program Studi Universitas Buddhi Dharma yang dibuktikan dengan keasliannya.
5. Lembar pernyataan ini, saya buat dengan sesungguhnya, tanpa paksaan dan apabila dikemudian hari atau pada waktu lainnya terdapat penyimpangan dan tidak benaran dalam pernyataan ini, saya bersedia menerima sanksi akademik berupa pencabutan gelar akademik yang telah saya peroleh karena Skripsi ini serta sanksi lainnya sesuai dengan peraturan dan norma yang berlaku.

Tangerang, 4 Februari 2025

Yang membuat pernyataan,

Feri Winarta

NIM : 20201000041

UNIVERSITAS BUDDHI DHARMA

LEMBAR PERSETUJUAN PUBLIKASI KARYA ILMIAH

Yang bertanda tangan di bawah ini:

NIM	: 20201000041
Nama	: Feri Winarta
Jenjang Studi	: Sastra 1
Program Studi	: Teknik Informatika
Peminatan	: Infrastructure & Security

Dengan ini menyetujui untuk memberikan ijin kepada pihak Universitas Buddhi Dharma, Hak Bebas Royalti Non – Eksklusif (*Non-exclusive Royalty-Free Right*) atas karya ilmiah kami yang berjudul: "*integrasi kubernetes dengan load balancing dan skalabilitas dalam pengembangan aplikasi manajemen lingkungan berbasis android untuk perumahan komersil*", beserta alat yang diperlukan.

Dengan Hak Bebas Royalti Non – Eksklusif ini pihak Universitas Buddhi Dharma berhak menyimpan, mengalih-media atau format-kan, mengelolanya dalam pangkalan data (*database*), mendistribusikannya, dan menampilkan atau mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari saya selama tetap mencantumkan nama saya sebagai penulis pertama atau pencipta karya ilmiah tersebut.

Saya bersedia untuk menanggung secara pribadi, tanpa melibatkan pihak Universitas Buddhi Dharma, segala bentuk tuntutan hukum yang timbul atas pelanggaran Hak Cipta dalam karya ilmiah saya ini. Demikian pernyataan ini saya buat dengan sebenarnya.

Tangerang, 4 Februari 2025

Yang membuat pernyataan,

Feri Winarta

NIM : 20201000041

UNIVERSITAS BUDDHI DHARMA
LEMBAR PENGESAHAN PEMBIMBING

INTEGRASI KUBERNETES DENGAN LOAD BALANCING DAN SKALABILITAS DALAM PENGEMBANGAN APLIKASI MANAJEMEN LINGKUNGAN BERBASIS ANDROID UNTUK PERUMAHAN KOMERSIL

Dibuat oleh:

NIM : 20201000041

Nama : Feri Winarta

Telah disetujui untuk dipertahankan di hadapan TIM Penguji Ujian Kompherensif

Program Studi Teknik Informatika

Peminatan Infrastructure & Security

Tahun Akademik 2024/2025

Disahkan oleh,

Tangerang, 4 Februari 2025

Pembimbing,

Rino S,Kom M.Kom.

NIDN: 0420058502

UNIVERSITAS BUDDHI DHARMA

LEMBAR PENGESAHAN KERJA SKRIPSI

INTEGRASI KUBERNETES DENGAN LOAD BALANCING DAN SKALABILITAS DALAM PENGEMBANGAN APLIKASI MANAJEMEN LINGKUNGAN BERBASIS ANDROID UNTUK PERUMAHAN KOMERSIL

Dibuat oleh:

NIM : 20201000041

Nama : Feri Winarta

Telah disetujui untuk dipertahankan di hadapan TIM Penguji Ujian Kompherensif

Program Studi Teknik Informatika

Peminatan Infrastructure & Security

Tahun Akademik 2024/2025

Disahkan oleh,

Tangerang, 4 Februari 2025

Dekan,

Dr. Yakub, M.Kom., M.M.

NIDN: 0304056901

Ketua Program Studi,

Hartana Wijaya, S.kom, M.Kom.

NIDN: 0412058102

UNIVERSITAS BUDDHI DHARMA

LEMBAR PENGESAHAN TIM PENGUJI

Nama	: Feri Winarta
NIM	: 20201000041
Fakultas	: Sains dan Teknologi
Judul Skripsi	: INTEGRASI KUBERNETES DENGAN LOAD BALANCING DAN SKALABILITAS DALAM PENGEMBANGAN APLIKASI MANAJEMEN LINGKUNGAN BERBASIS *ANDROID* UNTUK PERUMAHAN KOMERSIL

Dinyatakan **LULUS** setelah mempertahankan di depan Tim Penguji pada hari Selasa, 04 Februari 2025

	Nama Penguji:	**Tanda Tangan:**
Ketua Sidang	**: Susanto Hariyanto, S.Kom.,M.Kom** NIDN : 04281268601	.....
Penguji 1	**: Rudy Arijanto, S.Kom., M.Kom** NIDN : 0415077105	.....
Penguji 2	**: Rino, M.Kom** NIDN : 04200058502	.....

Mengetahui,

Dekan Fakultas Sains dan Teknologi

Dr. Yakub, M.Kom., M.M.

NIDN: 0304056901

KATA PENGANTAR

Puji dan Syukur kepada Tuhan Yang Maha Esa saya panjatkan, yang telah memberi hikmah dan kesehatan bagi saya sehingga saya dapat menyusun dan menyelesaikan karya ilmiah ini (Skripsi). Dengan judul "*INTEGRASI KUBERNETES DENGAN LOAD BALANCING DAN SKALABILITAS DALAM PENGEMBANGAN APLIKASI MANAJEMEN LINGKUNGAN BERBASIS ANDROID UNTUK PERUMAHAN KOMERSIL*".

Tujuan utama dari penyusunan karya ilmiah ini (Skripsi) adalah sebagai satu syarat untuk menyelesaikan program pendidikan Strata 1 (Sarjana) dengan Program Studi Teknik Informatika di Universitas Buddhi Dharma.

Dalam proses penyusunan karya ilmiah ini (Skripsi) saya banyak menerima masukan dan bantuan dari berbagai pihak, maka pada kesempatan kali ini saya akan menyampaikan banyak terimakasih yang sebesar – besarnya kepada :

1. Ibu. Dr. Limajatini, SE., MM., BKP, Selaku Rektor Universitas Buddhi Dharma.
2. Bapak. Dr. Yakub, S.Kom., M.Kom., M.M., Sebagai Dekan Fakultas Sains dan Teknologi Universitas Buddhi Dharma.
3. Bapak. Hartana Wijaya, S.Kom., M.Kom, Selaku Ketua Program Studi Teknik Informatika Universitas Buddhi Dharma.
4. Bapak. Rino, S.Kom., M.Kom.,Selaku Dosen Pembimbing saya yang telah membantu dan mendukung saya sehingga saya dapat menyelesaikan karya ilmiah ini.
5. Kedua Orang tua saya serta kelurga yang telah memberi dukungan kepada saya, sehingga saya berhasil menyelesikan karya ilmaih ini.
6. Serta Teman – teman saya yang selalu membantu serta memberi semangat kepada saya, sehingga saya berhasil menyelesikan karya ilmaih ini.

Saya sebagai penulis pan penyusun karya ilmiah ini (Skripsi) menyadari bahwa masih banyak kekurangan dalam penyusunan karya ilmah ini, baik yang disegaja maupun tidak disegaja. Maka dari itulah saya sebagai penulis memohon maaf atas segala kekurangan tersebut.

Dan saya sebagai penulis dan penyusun karya ilmiah ini (Skripsi) berharap semoga dengan selesainya penyusunan karya ilmiah ini dapat membantu serta bermanfaat bagi instansi, dan masyarakat luas. Tidak lupa saya ucapkan terimakasih banyak.

Tangerang, 4 Februari 2025

Penulis

INTEGRASI KUBERNETES DENGAN LOAD BALANCING DAN SKALABILITAS DALAM PENGEMBANGAN APLIKASI MANAJEMEN LINGKUNGAN BERBASIS *ANDROID* UNTUK PERUMAHAN KOMERSIL

Hal 160 + xix Halaman / 17 Tabel / 57 Gambar / 44 Daftar Pustaka

ABSTRAK

Penelitian ini berfokus pada pengembangan aplikasi manajemen lingkungan berbasis Android yang mengintegrasikan teknologi Kubernetes, load balancing, dan skalabilitas untuk perumahan komersil. Latar belakang penelitian ini didasari oleh kebutuhan akan sistem yang efisien dalam menangani laporan permasalahan lingkungan yang sering kali tidak terkelola dengan baik. Masalah yang dihadapi mencakup kesulitan warga dalam melaporkan isu lingkungan dan keterbatasan infrastruktur yang ada. Metode yang digunakan dalam penelitian ini meliputi perancangan sistem, pengembangan aplikasi menggunakan framework Flutter, serta penerapan Kubernetes untuk pengelolaan kontainer. Tujuan dari penelitian ini adalah untuk menciptakan aplikasi yang dapat meningkatkan responsivitas dan ketersediaan layanan, serta memudahkan warga dalam melaporkan isu lingkungan secara real-time. Hasil penelitian menunjukkan bahwa aplikasi yang dikembangkan mampu menangani lonjakan permintaan dengan baik, berkat penerapan load balancing dan horizontal scaling yang dilakukan oleh Kubernetes. Temuan ini mengindikasikan bahwa integrasi teknologi ini tidak hanya meningkatkan performa aplikasi, tetapi juga memberikan pengalaman pengguna yang lebih baik. Kesimpulan dari penelitian ini adalah bahwa penggunaan Kubernetes dalam pengembangan aplikasi manajemen lingkungan sangat efektif dan dapat menjadi model untuk aplikasi serupa di masa depan. Penelitian lebih lanjut diperlukan untuk mengeksplorasi fitur tambahan dan peningkatan sistem agar dapat memenuhi kebutuhan pengguna yang terus berkembang

Kata Kunci: Kubernetes, Load Balancing,Skalabilitas, Flutter, Manajemen Perumahan.

INTEGRATION OF KUBERNETES WITH LOAD BALANCING AND SCALABILITY IN THE DEVELOPMENT OF ANDROID-BASED ENVIRONMENTAL MANAGEMENT APPLICATIONS FOR COMMERCIAL HOUSING

Page 160 + xix Page / 17 Table / 57 Figures / 44 Blibiography

ABSTRACT

This research focuses on the development of an Android-based environmental management application that integrates Kubernetes technology, load balancing, and scalability for commercial housing. The background of this research is based on the need for an efficient system to handle environmental issue reports that are often poorly managed. The problems faced include difficulties for residents in reporting environmental issues and the limitations of existing infrastructure. The methods used in this research include system design, application development using the Flutter framework, and the implementation of Kubernetes for container management. The aim of this research is to create an application that can enhance responsiveness and service availability, as well as facilitate residents in reporting environmental issues in real-time. The results of the research show that the developed application is capable of handling spikes in demand effectively, thanks to the implementation of load balancing and horizontal scaling performed by Kubernetes. These findings indicate that the integration of this technology not only improves application performance but also provides a better user experience. The conclusion of this research is that the use of Kubernetes in the development of environmental management applications is highly effective and can serve as a model for similar applications in the future. Further research is needed to explore additional features and system enhancements to meet the evolving needs of users.

Keywords: Kubernetes, Load Balancing,Scalability, Flutter, Housing Management

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Perumahan adalah kumpulan rumah sebagai bagian dari permukiman,. baik perkotaan maupun perdesaan, yang dilengkapi dengan prasarana, sarana, dan utilitas umum sebagai hasil upaya pemenuhan rumah yang layak huni (UURI Nomor 1/2011 tentang Perumahan dan Kawasan Permukiman) (Dr. Ir. M. Amir Salipu et al., 2023). Perumahan pada umumnya memiliki struktur organisasi yang digunakan untuk mengatur dan menjalankan fungsionalitas perumahan, mulai dari ketua RW, ketua RT, sekretaris, bendahara, hingga bagian perlengkapan dan keamanan.

PT Sinar Solusi Informatika (NextG) adalah sebuah perusahan yang bergerak didalam bidang teknologi. Perusahaan ini merupakan sebuah *software house* yang mengembangkan aplikasi atau sistem sesuai permintaan. Seperti salah satu pemintaan RW 05 Bukit Golf Mediterania PIK yang memiliki permintaan pembuatan aplikasi untuk mengurus kebutuhan manajemen lingkungan.

RW 05 Bukit Golf Mediterania PIK yang memiliki divisi tambahan seperti *estate manager* yang bertugas mengelola lingkungan perumahan. Tugas-tugas tersebut meliputi perawatan taman, kebersihan lingkungan, kesehatan lingkungan, perawatan mekanikal dan elektrikal, perawatan *clubhouse*, pengendalian bangunan, perawatan kolam renang, pengurusan administrasi, dan permasalahan keamanan. Perumahan Bukit Golf Mediterania PIK merupakan salah satu perumahan mewah di kawasan Pantai Indah Kapuk, Jakarta Utara. RW 05 perumahan ini memiliki jumlah unit rumah sekitar 700 dan terdiri dari 4 Klaster, yaitu Klaster Akasia Golf, Klaster Cendana Golf, Klaster Ebony Golf, dan Klaster Damar Golf.

Sebagai sebuah RW yang besar dan padat penduduk, RW 05 Bukit Golf Mediterania PIK menghadapi berbagai permasalahan, salah satunya adalah permasalahan lingkungan. Permasalahan lingkungan yang sering terjadi di RW ini antara lain rumput taman yang sudah panjang, listrik mati, sampah berserakan, permasalahan keamanan, dan penanganan infrastruktur lainnya. Permasalahan lingkungan tersebut dapat mengganggu kenyamanan dan keamanan warga RW 05. Oleh karena itu, diperlukan solusi efektif untuk mengatasi permasalahan tersebut.

Sebelumnya, RW 05 telah menerapkan pelaporan permasalahan lingkungan melalui aplikasi perpesanan daring. Namun, cara ini masih tergolong manual dan memiliki beberapa kelemahan, seperti kesulitan bagi pekerja untuk mendapatkan lokasi terkini permasalahan, warga tidak mengetahui riwayat penyelesaian permasalahan, serta pengelola kesulitan melakukan penilaian hasil kerja pekerja karena tidak berbasis data langsung.

Untuk mengatasi permasalahan tersebut, salah satu solusi yang dapat diterapkan adalah pengembangan aplikasi manajemen lingkungan berbasis *Android*. Aplikasi ini dapat memberikan kemudahan bagi warga dalam melaporkan permasalahan lingkungan, sehingga permasalahan tersebut dapat segera ditangani. Dalam pengembangan aplikasi ini, kemudahan dalam pengembangan dan dukungan *cross-platform* menjadi pertimbangan utama. *Flutter* sendiri adalah sebuah *f*ram*ework* pemrograman dengan bahasa dart yang memudahkan pengembang untuk membuat aplikasi yang dapat berjalan di lebih dari satu jenis peralatan. Pada saat ini *flutter* dapat menggunakan aplikasi *mobile* (*IOS*, dan *android*), *web*, dan *desktop* (Handoyo et al., 2022).

Namun, seiring dengan berkembangnya aplikasi dan meningkatnya jumlah pengguna, diperlukan infrastruktur yang mampu menangani beban kerja yang tinggi

dan memastikan ketersediaan layanan secara terus-menerus. Masalah yang dihadapi adalah bahwa beban lalu lintas yang tinggi sering menyebabkan *server* mengalami *downtime* dan kelebihan beban (Ridha & Suhatman, 2022).

Untuk itu, integrasi *Kubernetes* dengan strategi *deployment* menggunakan *load balancing* dan *horizontal scaling* sangat penting dalam mendukung *high availability* dan skalabilitas aplikasi.

Kubernetes adalah kerangka kerja sumber terbuka yang dirancang untuk mengelola beban kerja aplikasi yang terkontainerisasi, yang menawarkan fitur-fitur seperti konfigurasi deklaratif dan otomatisasi. Kerangka kerja ini hadir dalam ekosistem yang luas dan berkembang pesat. Berbagai layanan, dukungan, dan alat yang terkait dengan *Kubernetes* dapat diakses dengan mudah. (Rina & Ridha, 2021).

Dengan menerapkan *Kubernetes*, aplikasi dapat di-*deploy* dalam bentuk kontainer yang terisolasi, sehingga memudahkan pengelolaan dan pengembangan lebih lanjut. Dengan menerapkan teknik *load balancing*, pelaku dapat mengalokasikan permintaan layanan yang masuk secara efektif ke sumber daya yang paling sesuai dan dapat diakses. Alokasi strategis ini meminimalkan potensi kemacetan dan mengoptimalkan *throughput* secara keseluruhan (Khoiru Sabila et al., 2024).

Dengan mengintegrasikan *Kubernetes* dalam pengembangan aplikasi manajemen lingkungan berbasis *Android* di Perumahan Bukit Golf Mediterania PIK, diharapkan dapat meningkatkan performa, skalabilitas, dan ketersediaan aplikasi. Penerapan *Kubernetes* juga memungkinkan pengelolaan infrastruktur yang lebih efisien dan fleksibel, sehingga dapat mendukung operasional aplikasi dalam jangka panjang.

Hal ini akan berdampak positif terhadap pelayanan kepada warga, karena permasalahan lingkungan dapat ditangani dengan lebih cepat dan efisien, serta informasi terkait penyelesaian permasalahan dapat diakses secara transparan. Integrasi *Kubernetes* dengan strategi *load balancing* dan *horizontal scaling* menjadi kunci dalam memastikan aplikasi dapat melayani permintaan pengguna dengan baik, tanpa mengalami *downtime* yang signifikan.

1.2 Identifikasi Masalah

Berdasarkan latar belakang dan penjelasan yang telah diberikan, maka identifikasi masalah yang dapat diidentifikasi adalah sebagai berikut:

1. Warga RW 05 harus membuat laporan permasalahan di lingkungannya masih menggunakan media perantara aplikasi perpesanan daring, sehingga sulit bagi pengelola lingkungan untuk menentukan lokasi masalah, serta dan memberikan update terkini riwayat penyelesaian permasalahan tersebut ke pelapor.
2. Infrastruktur saat ini tidak memadai untuk mendukung beban kerja yang meningkat. Tanpa dukungan *load balancing* dan arsitektur yang memastikan *high availability*, aplikasi rentan terhadap *downtime* dan kinerja yang menurun saat beban tinggi.

1.3 Ruang lingkup

Ruang lingkup penelitian ini dibatasi pada perancangan dan pembangunan sistem manajemen lingkungan penghuni perumahan berbasis *Android* di RW 05 perumahan bukit golf mediterania. Sistem ini akan mencakup fitur-fitur berikut:

1. Aplikasi dibuat menggunakan *framework flutter*
2. Aplikasi ini hanya dijalankan di *platform Android*.
3. Aplikasi menampilkan gambar, teks, dan grafik.
4. Database yang digunakan yaitu *MySql*
5. Bahasa pemrograman *back end* yang digunakan *PHP*

6. Fitur komplain masalah lingkungan dengan lokasi terkini menggunakan *gps*
7. Sistem *deployment* aplikasi menggunakan *Kubernetes* untuk pengelolaan *container*
8. Memanfaatkan fitur *load balancing*, *horizontal scaling*, dan fitur teknologi jaringan dari *Kubernetes*

1.4 Tujuan dan manfaat penulisan

1.4.1 Tujuan Penulisan

Adapun tujuan penulisan yang ingin dicapai :

1. Mengimplementasikan *framework Flutter* untuk pelaporan permasalahan lingkungan.
2. Mengintegrasikan *Kubernetes* dalam strategi *deployment* aplikasi untuk mendukung *load balancing* dan *horizontal scaling*

1.4.2 Manfaat Penulisan

1. Dengan implementasi *framework Flutter*, pengembang dapat membuat antarmuka pengguna yang seragam di berbagai perangkat *Android*. Hal ini membuatnya lebih mudah dipahami oleh pengguna, sehingga meminimalkan hambatan saat mereka melaporkan masalah lingkungan.
2. Dengan mengintegrasikan *Kubernetes* dalam strategi *deployment* aplikasi untuk mendukung *load balancing* dan *horizontal scaling*, aplikasi dapat menangani peningkatan beban kerja secara efisien. Hal ini memastikan performa aplikasi tetap optimal meskipun jumlah pengguna bertambah, sehingga pengalaman pengguna menjadi lebih baik tanpa gangguan kinerja.

1.5 Sistematika Penulisan

Sistematika penulisan adalah gambaran umum mengenai isi dari penelitian secara keseluruhan. Pokok pembahasan dalam skripsi ini dibagi menjadi 5 bab yang masing - masing terbagi menjadi beberapa sub bab guna agar dapat lebih dimengerti.

BAB I PENDAHULUAN

Berisikan pembahasan tentang latar belakang masalah, identifikasi masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Berisi pembahasan tentang gambaran umum teori terkait antara variabel dependen dan variabel independen, penelitian terdahulu beserta hasilnya, kerangka pemikiran dan hipotesis penelitian.

BAB III ANALISA MASALAH DAN PERANCANGAN APLIKASI

Berisi pembahasan tentang analisa kebutuhan, konstruksi algoritma atau metode, perancangan layar, menu, database, perancangan prototype.

BAB IV PENGUJIAN DAN IMPLEMENTASI

Berisi pembahasan tentang, pembahasan metode dan algoritma spesifikasi hardware dan software, tampilan Program (screenshot program), pengujian Aplikasi.

BAB V KESIMPULAN DAN SARAN

Berisi kesimpulan, implikasi, dan saran untuk penelitian selanjutnya yang akan dilakukan.

BAB II

LANDASAN TEORI

2.1 Integrasi *Kubernetes* Dengan *Load Balancing* Dan *Horizontal* Skalabilitas

2.1.1 Integrasi *Kubernetes*

a. Integrasi

Dalam Kamus Siswa Sekolah Menengah Pertama, istilah integrasi diartikan sebagai proses penyatuan sehingga terbentuk suatu kesatuan yang utuh dan utuh (Rusdiana, 2023)

Integrasi adalah cara menyatukan berbagai hal. Artinya, menyatukan berbagai bagian menjadi satu. Dalam pengertian ini, integrasi adalah tentang memadukan berbagai elemen untuk membentuk sesuatu yang kompleks, suatu keseluruhan yang lebih dari sekadar jumlah bagian-bagiannya. (Sunarti et al., 2021)

Berdasarkan penjelasan ahli diatas, dapat disimpulkan bahwa integrasi adalah proses atau cara untuk menyatukan berbagai elemen atau bagian sehingga terbentuk suatu kesatuan yang utuh dan kompleks. Integrasi melibatkan penyatuan elemen-elemen yang berbeda untuk membentuk suatu keseluruhan yang lebih dari sekadar penjumlahan bagian-bagian tersebut.

b. *Docker*

Docker merupakan sebuah aplikasi yang menggunakan teknologi open source, yang memungkinkan para pengembang atau siapa saja untuk membuat, menjalankan, menguji, dan meluncurkan aplikasi di dalam sebuah kontainer (Dwiyatno et al., 2020)

Docker adalah proyek *open-source* yang bertujuan untuk memudahkan pengembang dan administrator sistem dalam proses pembangunan, pengemasan, dan pembuatan aplikasi di berbagai lokasi menggunakan kontainer. Arsitekturnya mengadopsi model klien-server, di mana klien Docker mengirimkan permintaan kepada daemon Docker untuk melakukan pembangunan, distribusi, dan eksekusi kontainer (Subekti et al., 2024)

Berdasarkan pendapat ahli diatas maka dapat disimpulkan bahwa Docker adalah sebuah aplikasi yang memanfaatkan teknologi sumber terbuka, memungkinkan pengembang atau siapa pun untuk membuat, menjalankan, menguji, dan meluncurkan aplikasi di dalam kontainer. Selain itu, Docker juga merupakan proyek open-source yang dirancang untuk mempermudah pengembang dan administrator sistem dalam proses pembangunan, pengemasan, dan peluncuran aplikasi di berbagai tempat menggunakan kontainer.

c. *Kubernetes*

Kubernetes adalah alat untuk mengelola aplikasi yang dikontainerisasi. Alat ini bersifat sumber terbuka dan memungkinkan konfigurasi dan otomatisasi yang jelas. Alat ini hadir dalam ekosistem yang luas dan terus berkembang (Rina & Ridha, 2021).

Kubernetes berfungsi sebagai *platform* orkestrasi kontainer dan menawarkan kerangka kerja untuk memfasilitasi berbagai fungsinya. Arsitekturnya terdiri dari komponen-komponen penting, termasuk *Pod*, *Label*, dan *Selector* (Ridha & Suhatman, 2022)

Menurut para ahli diatas maka dapat disimpulkan bahwa *kubernetes* adalah sebuah alat sumber terbuka yang digunakan untuk mengelola aplikasi yang dikontainerisasi. Alat ini memungkinkan konfigurasi dan otomatisasi yang jelas, serta didukung oleh ekosistem yang luas dan terus berkembang

d. *Devops*

DevOps adalah suatu metodologi yang mengintegrasikan pengembangan (development) dan operasional (operations) untuk meningkatkan kolaborasi serta otomatisasi dalam proses pengembangan perangkat lunak. Tujuan utama dari pendekatan ini adalah untuk mempercepat peluncuran produk tanpa mengorbankan kualitas (Subekti et al., 2024).

Tujuan utama dari metode DevOps adalah untuk mempercepat pengiriman perangkat lunak, serta menjadi strategi yang efektif untuk mendapatkan perangkat lunak yang efisien sesuai dengan kebutuhan pengguna atau pengembang (Gulo & Simanjuntak, 2021).

Menurut pendapat ahli diatas maka dapat disimpulkan bahwa *DevOps* adalah metodologi yang menggabungkan pengembangan dan operasional untuk meningkatkan kolaborasi dan otomatisasi dalam proses pengembangan perangkat lunak. Tujuan utama dari pendekatan ini adalah untuk mempercepat peluncuran dan pengiriman produk tanpa mengorbankan kualitas, serta untuk menghasilkan perangkat lunak yang efisien sesuai dengan kebutuhan pengguna dan pengembang.

e. Repository

Repository adalah sebuah tempat penyimpanan digital yang digunakan untuk menyimpan data dalam bentuk teks, audio, dan video. Ini merupakan salah satu solusi untuk menghemat ruang penyimpanan serta mengoptimalkan proses pencarian data saat dibutuhkan (Reza et al., 2022).

Repository adalah sebuah tempat penyimpanan yang dirancang untuk menyimpan berbagai jenis program atau aplikasi yang telah dikembangkan dengan cara tertentu, sehingga dapat diakses dengan mudah melalui internet (Reza et al., 2022)

Berdasarkan pendapat para ahli tersebut, dapat disimpulkan bahwa repository adalah tempat penyimpanan digital yang berfungsi untuk menyimpan berbagai jenis data, termasuk teks, audio, dan video, serta program atau aplikasi yang telah dikembangkan. Repository memungkinkan akses yang mudah melalui internet, sehingga memfasilitasi kolaborasi dan berbagi informasi di antara pengguna dan pengembang.

f. Registri kontainer

Registri kontainer merupakan layanan yang menyimpan serta mendistribusikan gambar kontainer dan artefak yang berkaitan. Salah satu contoh registri kontainer publik adalah *Docker Hub*, yang berfungsi sebagai katalog umum untuk gambar kontainer *Docker* (Microsoft, 2024)

Registri kontainer memberikan kemudahan bagi para pengembang dalam proses penyimpanan, pembagian, dan pengelolaan gambar kontainer, termasuk di dalamnya gambar *Docker*. Dengan menggunakan layanan ini, pengembang dapat dengan mudah menyimpan berbagai versi gambar

kontainer yang mereka buat, sehingga memudahkan dalam pengelolaan dan pemeliharaan aplikasi (oracle, 2025).

Berdasarkan penjelasan diatas maka dapat disimpulkan Registri kontainer memainkan peran penting dalam ekosistem pengembangan perangkat lunak dengan menyediakan layanan untuk menyimpan dan mendistribusikan gambar kontainer serta artefak terkait. Contoh yang paling dikenal adalah *Docker Hub*, yang berfungsi sebagai katalog publik untuk gambar kontainer *Docker*. Dengan memanfaatkan registri kontainer, pengembang dapat dengan mudah menyimpan, membagikan, dan mengelola berbagai versi gambar kontainer, yang pada gilirannya meningkatkan efisiensi dalam pengelolaan dan pemeliharaan aplikasi

2.1.2 *Load Balancing*

Load balancing adalah metodologi strategis yang digunakan untuk mengalokasikan lalu lintas data dengan cara yang mendukung distribusi beban kerja yang merata di berbagai jaringan. Teknik ini bertujuan untuk meningkatkan pemanfaatan sumber daya dan meningkatkan kinerja sistem secara keseluruhan (Odhi Dwi Putra et al., 2020).

Dengan mengoptimalkan distribusi beban, *load balancing* bukan hanya meningkatkan kinerja jaringan saja, tapi juga memastikan layanan yang ada memiliki ketersedian yang tinggi serta mengurangi resiko kegagalan sistem (Khoiru Sabila et al., 2024).

Berdasarkan pendapat ahli diatas, *load balancing* merupakan strategi penting dalam mengelola lalu lintas data untuk memastikan distribusi beban kerja yang merata di jaringan. Ini tidak hanya meningkatkan kinerja sistem,

tetapi juga meningkatkan pemanfaatan sumber daya, memastikan ketersediaan layanan yang tinggi, dan mengurangi risiko kegagalan sistem.

2.1.3 *Horizontal* Skalabilitas

Skalabilitas adalah kemampuan suatu layanan untuk memperluas kapasitas sumber daya yang dimilikinya dalam menghadapi peningkatan beban kerja. Dengan kata lain, layanan yang memiliki skalabilitas yang baik dapat menyesuaikan diri secara efektif terhadap permintaan yang meningkat, baik melalui penambahan sumber daya secara vertikal misalnya, meningkatkan spesifikasi *server* maupun *horizontal* misalnya, menambah jumlah *server* atau meningkatkan jumlah *instance* (Tanuwijaya et al., 2021)

Skala *horizontal* merujuk pada kemampuan sebuah orkestrator atau *platform* manajemen kontainer untuk meningkatkan jumlah *instance* atau layanan dengan menambahkan lebih banyak *node* ke dalam klaster. Dengan cara ini, sistem dapat diperluas secara efisien untuk menangani permintaan yang lebih tinggi (Subhi Basit et al., 2023)

Dapat disimpulkan bahwa skalabilitas merupakan kemampuan krusial suatu layanan untuk menyesuaikan kapasitas sumber daya dalam menghadapi peningkatan beban kerja. Layanan yang *scalable* dapat melakukan penyesuaian baik secara vertikal, seperti meningkatkan spesifikasi *server*, maupun secara *horizontal*, dengan menambah jumlah *server* atau *instance*.

2.2 Pengembangan Aplikasi Manajemen Lingkungan

2.2.1 Pengembangan Aplikasi

Salah satu tujuan utama dalam pengembangan aplikasi adalah untuk menghasilkan aplikasi yang memenuhi kebutuhan dan harapan pengguna.

Hal ini mencakup pengetahuan mendalam tentang siapa pengguna aplikasi, apa yang dibutuhkan, dan bagaimana aplikasi dapat memberikan solusi bagi permasalahan dan kebutuhan pengguna. Memprioritaskan kebutuhan pengguna akan membuat aplikasi lebih relevan bagi pengguna (Magdalena et al., 2024).

Pengembangan aplikasi juga memperhatikan aspek yang memberikan pengalaman pengguna yang berkesan dan menarik. Hal ini termasuk merancang antarmuka pengguna yang responsif, intuitif, dan menarik (Magdalena et al., 2024).

Berdasarkan pendapat ahli diatas, maka disimpulkan bahwa pengembangan aplikasi harus fokus pada pemenuhan kebutuhan dan harapan pengguna. Ini melibatkan pemahaman yang mendalam tentang pengguna, kebutuhan mereka, dan bagaimana aplikasi dapat memberikan solusi yang efektif. penting untuk menciptakan pengalaman pengguna yang menarik melalui desain antarmuka yang responsif dan intuitif.

2.2.2 Manajemen Lingkungan

Manajemen lingkungan muncul di sekitar tahun 1970an sebagai sarana untuk memecahkan masalah, dimana di dalamnya terdapat panduan praktis yang biasanya resmi dari suatu lembaga atau negara (Utomo et al., 2021)

Manajemen lingkungan adalah proses yang menekankan interaksi antara manusia dan lingkungan. Proses ini bertujuan untuk mengidentifikasi kebutuhan lingkungan, serta memahami aspek sosial, ekonomi, dan teknologi yang mendesak yang harus ditangani untuk memenuhi kebutuhan tersebut. Selain itu, manajemen lingkungan juga mencakup penentuan pilihan atau

langkah-langkah yang paling efektif untuk memenuhi kebutuhan lingkungan dengan mempertimbangkan semua aspek tersebut (Utomo et al., 2021)

Berdasarkan pendapat ahli tersebut, maka manajemen lingkungan, yang muncul sekitar tahun 1970-an, berfungsi sebagai alat untuk mengatasi masalah lingkungan. Proses ini melibatkan pemahaman mengenai interaksi antara manusia dan lingkungan serta identifikasi kebutuhan lingkungan. Selain itu, manajemen lingkungan juga memperhatikan aspek sosial, ekonomi, dan teknologi yang mendesak, dan berfokus pada pengembangan langkah-langkah efektif untuk memenuhi kebutuhan tersebut

2.3 Berbasis *Flutter Dan PHP*

2.3.1 Flutter

Flutter adalah alat pengembangan perangkat lunak buatan *Google* yang memungkinkan pengembang membuat aplikasi untuk berbagai sistem operasi seperti *Android*, *iOS*, *Web*, *Linux*, dan *MacOS* hanya dengan satu kode sumber (Ariestiandy et al., 2023)

Flutter menyediakan serangkaian widget yang sepenuhnya dapat disesuaikan untuk membangun antarmuka pengguna asli, termasuk perpustakaan *Desain Material* yang elegan dan *widget Cupertino*. Ini memungkinkan pembuatan UI dengan cepat tanpa kehilangan status pada *emulator*, simulator, dan perangkat keras *iOS* serta *Android* (Nelly Sofi & Riza Dharmawan, 2022).

Dengan kedua pendapat ahli diatas maka dapat disimpulkan bahwa *Flutter* merupakan alat pengembangan perangkat lunak buatan *Google*, memungkinkan pengembang membuat aplikasi untuk berbagai sistem

operasi seperti *Android*, *iOS*, *Web*, *Linux*, dan *MacOS* dengan menggunakan satu kode sumber.

2.3.2 PHP

PHP yang merupakan singkatan dari Hypertext Preprocessor adalah bahasa pemrograman sisi server yang saat ini banyak digunakan terutama dalam pengembangan situs web dinamis. Dalam beberapa aspek tertentu dari pembuatan situs web, penggunaan bahasa pemrograman PHP sangat penting, contohnya untuk memproses data yang dikirimkan oleh pengunjung situs (Zulfa & Wanda, 2023)

PHP dapat digunakan untuk mengembangkan berbagai jenis aplikasi web, mulai dari halaman web yang sederhana hingga aplikasi yang lebih kompleks yang memerlukan koneksi ke database. Saat ini, PHP telah mendukung banyak jenis database, dan kemungkinan dukungan ini akan terus berkembang. (Dewi & Putra, 2022)

Berdasarkan pendapat para ahli diatas maka dapat disimpulkan bahwa PHP, atau *Hypertext Preprocessor*, merupakan bahasa pemrograman sisi server yang sangat populer dan banyak digunakan dalam pengembangan situs web dinamis. Penggunaan PHP sangat krusial dalam berbagai aspek pembuatan situs, terutama dalam memproses data yang dikirimkan oleh pengunjung. Selain itu, PHP memiliki fleksibilitas yang tinggi, memungkinkan pengembang untuk menciptakan berbagai jenis aplikasi web, mulai dari yang sederhana hingga yang kompleks yang memerlukan koneksi ke database.

2.4 Load Test

Load Test adalah salah satu jenis pengujian kinerja yang dirancang untuk mengevaluasi bagaimana sistem berfungsi ketika sejumlah besar pengguna virtual melakukan interaksi secara bersamaan. Pengujian ini dilakukan selama periode waktu tertentu untuk mengamati respons sistem dalam menghadapi beban yang tinggi (Arlinta Christy Barus et al., 2022)

Load test berfungsi sebagai pengujian kinerja dan memastikan bahwa perangkat lunak beroperasi dengan baik. Metode ini digunakan untuk mengukur waktu respons dalam berbagai kondisi. Hasil dari *load test* dapat dianalisis untuk mengidentifikasi masalah performa yang mungkin terjadi pada aplikasi (Fansha et al., 2021)

Berdasarkan pendapat para ahli diatas maka dapat disimpulkan bahwa *Load Test* merupakan metode penting dalam pengujian kinerja yang bertujuan untuk mengevaluasi bagaimana sistem beroperasi ketika dihadapkan pada interaksi simultan dari sejumlah besar pengguna virtual. Pengujian ini tidak hanya dilakukan untuk mengamati respons sistem dalam kondisi beban tinggi, tetapi juga untuk memastikan bahwa perangkat lunak berfungsi dengan baik. Dengan mengukur waktu respons dalam berbagai kondisi, hasil dari *load test* dapat memberikan wawasan yang berharga untuk mengidentifikasi dan menganalisis masalah performa yang mungkin muncul pada aplikasi.

2.5 Pengujian Skalabilitas

Pengujian skalabilitas merupakan metode pengujian perangkat lunak non-fungsional yang bertujuan untuk mengevaluasi seberapa baik aplikasi dan infrastruktur beroperasi saat menghadapi perubahan beban kerja, baik yang meningkat maupun menurun. Dengan mengidentifikasi ketahanan pada sisi server dan penurunan kinerja di sisi klien, organisasi dapat mengoptimalkan biaya infrastruktur serta menciptakan pengalaman pengguna yang lebih baik (Ahmad, n.d.)

Berdasarkan pendapat diatas maka dapat disimpulkan bahwa pengujian skalabilitas sangat relevan dan mencerminkan pentingnya aspek ini dalam pengembangan perangkat lunak modern. Pengujian skalabilitas tidak hanya berfokus pada kemampuan aplikasi untuk menangani beban kerja yang meningkat, tetapi juga pada kemampuannya untuk beradaptasi saat beban kerja menurun. Ini menunjukkan bahwa pengujian skalabilitas adalah proses yang komprehensif dan dinamis.

2.6 Prometheus dan Grafana

2.6.1 Prometheus

Prometheus adalah perangkat lunak open-source yang digunakan untuk pemantauan dan peringatan sistem, awalnya dikembangkan di SoundCloud. Sejak peluncurannya pada tahun 2012, banyak perusahaan dan organisasi telah mengadopsi Prometheus, dan kini memiliki komunitas pengembang serta pengguna yang sangat aktif (Rahman et al., 2020)

Prometheus dibuat untuk memantau kinerja dan status sistem secara real-time dengan mengumpulkan metrik dari berbagai komponen sistem, seperti penggunaan server. Prometheus memiliki kemampuan untuk mengumpulkan dan menyimpan data metrik, yang memungkinkan pengguna untuk menganalisis perilaku sistem (Rasyidi & Pratama, 2024)

Berdasarkan pendapat ahli diatas maka dapat disimpulkan bahwa Prometheus merupakan perangkat lunak open-source yang dirancang untuk pemantauan dan peringatan sistem, yang awalnya dikembangkan di SoundCloud. Sejak diluncurkan pada tahun 2012, Prometheus telah diadopsi oleh banyak perusahaan dan organisasi, serta memiliki komunitas pengembang dan pengguna yang aktif. Fungsionalitas utama Prometheus adalah kemampuannya untuk memantau kinerja dan status sistem secara real-time dengan mengumpulkan metrik dari berbagai komponen, termasuk penggunaan server.

2.6.2 Grafana

Grafana adalah perangkat lunak open-source yang digunakan untuk memvisualisasikan dan menganalisis data melalui grafik dan dashboard interaktif. Dalam konteks ilmiah, Grafana merupakan alat yang umum digunakan dalam pemantauan sistem dan analisis data. Grafana memungkinkan pengguna untuk mengintegrasikan dan memvisualisasikan data dari berbagai sumber, termasuk basis data dan sistem pemantauan seperti Prometheus (Rasyidi & Pratama, 2024)

Grafana adalah perangkat lunak open-source yang digunakan untuk visualisasi dan analisis data. Grafana memungkinkan pengguna untuk memvisualisasikan, mengatur peringatan, dan menjelajahi metrik yang disimpan. Alat ini berfungsi untuk mengubah data dari basis data time series (TSDB) menjadi grafik dan visualisasi yang menarik (Rahman et al., 2020)

Berdasarkan pendapat ahli diatas maka dapat disimpulkan bagwa grafana adalah perangkat lunak open-source yang dirancang untuk memvisualisasikan dan menganalisis data melalui grafik dan dashboard interaktif. Dalam bidang ilmiah, Grafana sering digunakan untuk pemantauan sistem dan analisis data. Perangkat ini memungkinkan pengguna untuk mengintegrasikan dan memvisualisasikan data dari berbagai sumber, termasuk basis data dan sistem pemantauan seperti Prometheus.

2.7 Perancangan sistem dan arsitektur aplikasi

2.7.1 Desain dan Arsitektur Aplikasi Berbasis *Mobile*

Desain struktur aplikasi manajemen lingkungan penghuni haruslah mempertimbangkan beberapa aspek yang penting sehingga memungkinkan pengguna merasakan pengalaman yang baik, kinerja yang optimal, integrasi teknologi yang efektif

Salah satu tujuan utama dalam pengembangan aplikasi manajemen lingkungan adalah untuk menghasilkan aplikasi yang memenuhi kebutuhan dan harapan pengguna. Hal ini mencakup pengetahuan mendalam tentang siapa pengguna aplikasi, apa yang dibutuhkan, dan bagaimana aplikasi dapat memberikan solusi bagi permasalahan dan kebutuhan pengguna. Memprioritaskan kebutuhan pengguna akan membuat aplikasi lebih relevan bagi pengguna (Magdalena et al., 2024)

Pengembangan aplikasi manajemen lingkungan juga memperrhatikan aspek yang memberikan pengalaman pengguna yang berkesan dan menarik. Hal ini termasuk merancang antarmuka pengguna yang responsif, intuitif, dan menarik (Magdalena et al., 2024)

Memastikan aplikasi manajemen lingkungan mempunyai kualitas dan kinerja yang baik. Dengan mematuhi beberapa aspek-aspek seperti stabilitas, keamanan, integrasi dengan teknologi lainnya (Magdalena et al., 2024)

Untuk membentuk sebuah aplikasi manajemen lingkungan yang terintegrasi, artinya dapat berupa proses menghubungkan sistem yang sifatnya individu atau kelompok (Hamidin, 2017). Oleh karena itu integrasi layanan *google maps* yang dipadukan dengan *gps* amatlah dibutuhkan guna menunjang kebutuhan pengambilan

lokasi terkini pengguna yang memungkinkan fitur aplikasi manajemen lingkungan dapat berjalan dengan semestinya.

Pengujian aplikasi manajemen lingkungan menjadi peran sentral dalam memastikan aplikasi yang dikembangkan memenuhi standar dalam hal fungsionalitas, keamanan, keandalan dan kepuasan pengguna (Magdalena et al., 2024). Pengujian pada aplikasi manajemen lingkungan mencakup pengujian unit, pengujian integrasi, dan pengujian sistem langsung ke pengguna. Pengujian unit yang digunakan dalam manajem lingkungan dilakukan di bagian *front end* dan *back end* serta pengujian integrasi hanya dilakukan di satu tempat di *front end*. Pengujian langsung kepengguna memastikan bahwa aplikasi sudah sesuai dengan fungsionalitas dan kebutuhan (Magdalena et al., 2024)

Aplikasi manajemen lingkungan yang sudah jadi kemudian dioperasikan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada Langkah sebelumnya, perbaikan implementasi unit sistem dan peningkatan fitur sebagai kebutuhan baru (Sanubari et al., 2020)

2.7.2 Komponen-komponen utama yang terlibat dalam sistem aplikasi manajemen perumahan

1. Fitur peduli lingkungan (komplain)

   Sistem ini mengelola proses pembuatan komplain laporan warga, penerimaan laporan, proses kerja laporan, hingga penyelesaian laporan

2. Fitur status peduli lingkungan

   Menampilkan daftar laporan komplain yang sudah dibuat oleh setiap warga serta detail setiap laporan termasuk lokasi, status laporan, hasil kerja, serta penilaian langsung hasil kerja

3. Fitur status ipl

Fitur ini menampilkan riwayat pembayaran ipl yang dilakukan warga tiap bulannya

4. Fitur informasi warga & umum

Fitur ini menampilkan setiap informasi yang valid dan disajikan oleh pengelola ke setiap warga RW 05.

5. Fitur permintaan khusus

Fitur ini berfungsi bagi warga untuk melakukan permintaan khusus yang sudah ditentukan pengelola, seperti yang sedang tersedia saat ini adalah permintaan khusus pengambilan plastik kantong sampah.

6. Fitur *dashboard estate manager*

Fitur ini menampilkan grafik laporan komplain yang diterima setiap divisi kontraktor

2.7.3 UML (*Unified Modeling Language*)

UML adalah dasaran bagi pengembang perangkat lunak untuk mendefinisikan dari *requirement*, membuat analisa & desain dan menggambarkan arsitektur dalam pemrograman yang berorientasi pada objek (Josi, 2017, 52)

Uml bisa berfungsi sebagai jembatan dalam menkoneksikan beberapa aspek dari sistem (Bintana et al., 2020). Dengan demikian semua anggota tim yang bekerja dalam proyek yang sama akan mempunyai gambaran yang sama dengan sistem yang akan dibuat

2.7.4 *Use Case* Diagram

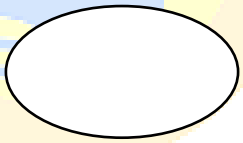
Use case adalah wujud penggambaran persyaratan sebuah sistem yaitu sistem apa yang seharusnya digunakan (Rachmat Destriana et al.,

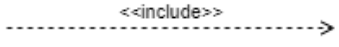
2021). Ini digunakan untuk menunjukan interaksi antar satu atau lebih aktor dalam sistem yang akan dibuat. Diagram ini sangat penting dalam memodelkan perilaku suatu sistem (Indah Purnama Sari, 2021).

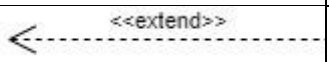
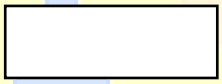
Didalam *use case* diagram terdapat beberapa komponen seperti aktor, *use case*, subjek (sistem). Sistem adalah semua elemen yang terlibat dalam konteks penggunaan tertentu di mana sistem tersebut beroperasi. Pelaku adalah pihak-pihak yang terlibat dalam interaksi dengan sistem, termasuk pengguna dan entitas lainnya. Kasus penggunaan merupakan deskripsi perilaku sistem yang diinginkan. Contoh kasus penggunaan menggambarkan bagaimana perilaku yang sesuai dengan kasus penggunaan tersebut terwujud dalam interaksi sistem (Rachmat Destriana et al., 2021).

Simbol-simbol yang tedapat dalam *use case* dapat dilihat pada table.

Tabel 2.1 Use Case

NO	Nama	Simbol	Keterangan
1.	*Use Case*		• Digambarkan dengan ellipsis *horizontal* • Nama *use case* menggunakan kata kerja
2.	Aktor		• Menggambarkan orang system / *external* entitas yang menyediakan atau menerima informasi

			• Merupakan lingkungan luar dari sistem • Nama aktor menggunakan kata benda • Aktor utama Digambar pada pojok kiri atau dari diagram
3.	Asosiasi		• Menggambarkan bagaimana aktor berinteraksi dengan *use case* • Bukan menggambarkan aliran data / informasi
4.	Generalisasi		• Gambarkan generalisasi antara *use case* atau antara aktor dengan panah tertutup yang mengarah dari *child* ke *parent*
5.	Relasi Include		• Hubungan antara dua *use case* untuk menunjukan adanya perilaku *use case* yang

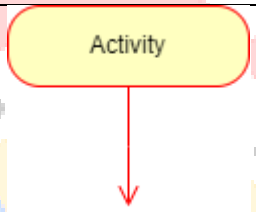
			dimasukan ke dalam perilaku dari base *use case* • Tandah panah terbuka harus tearah ke sub *use case*
6.	Relasi Extend		• Perluasan dari *use case* lain • Tanda panah terbuka harus terarah ke *use case* lain
7.	Boundary Boxes		• Untuk memperlihatkan Batasan sistem dengan lingkungan luar sistem

2.7.5 *Activity* Diagram

Diagram Aktivitas membahas aspek dinamis dari suatu sistem, menjadi kunci dalam menggambarkan fungsi sistem dan menyoroti bagaimana kontrol mengalir antara objek-objek yang terlibat (Indah Purnama Sari, 2021). Simbol-simbol yang digunakan dalam activity diagram dapat dilihat dalam table

Tabel 2.2 activity diagram

No	Nama	Simbol	Keterangan
1.	Start		• Menjelaskan awal proses kerja dalam *activity* diagram

			• Hanya ada satu simbol start
2.	End		• Menandai kondisi akhir dari suatu aktivitas dan mempresentasikan penyelesaian semua arus proses • Bisa lebih dari satu simbol *end*
3.	Activity		• Menunjukan kegiatan yang membentuk proses dalam diagram
4.	Join		• Menggabungkan dua atau lebih aktivitas bersamaan dan menghasilkan hanya satu aktivitas yang terjadi dalam satu waktu

2.7.6 *Class* Diagram

Class Diagram adalah spesifikasi yang, saat diinstansiasi, akan menciptakan objek dan menjadi elemen utama dalam pengembangan dan desain berbasis objek (Prof. Dr. Ir. Riri Fitri Sari & Utami, 2021).

Class Diagram menggambarkan kumpulan kelas, antarmuka, serta kolaborasi dan hubungan di antara elemen-elemen tersebut. Diagram ini memvisualisasikan

bagaimana kelas-kelas dan antarmuka saling berinteraksi dan berhubungan dalam suatu sistem Selain itu, *Class* Diagram menekankan pada aspek desain statis dari sistem, menunjukkan struktur dan organisasi komponen-komponen perangkat lunak tanpa memperhatikan perilaku dinamisnya. Diagram ini membantu dalam perencanaan dan dokumentasi desain sistem berbasis objek, serta mempermudah pemahaman dan komunikasi antar anggota tim pengembang.

Simbol-simbol yang digunakan dalam *class* diagram

Tabel 2.3 Class Diagram

NO	Simbol	Nama	Penjelasan
1.		Asosiasi	Relasi antar *class* dengan makna umum, biasanya disertai dengan *multiplicity*
2.		Asosiasi berarah	Relasi antar *class* dengan makna *class* yang satu digunakan oleh *class* yang lain, disertai dengan *multiplicity*
3.		Generalisasi	Relasi antar *class* dengan makna generalisasi-generalisasi
4.		Dependecy	Relasi antar *class* dengan makna ketergantungan antar *class*
5.		Agregasi	Relasi antar *class* dengan makna semua bagian (*whole-part*)

2.8 Pengujian *Blackbox Testing*

2.8.1 Pengujian

Pengujian adalah proses penting dalam pengembangan perangkat lunak yang dilakukan untuk menjalankan program dengan tujuan menemukan kesalahan atau bug

yang mungkin ada. Proses ini tidak hanya sekadar menjalankan kode, tetapi juga menganalisis hasilnya. Sebuah kasus uji yang baik dirancang untuk memiliki kemungkinan tinggi dalam mendeteksi kesalahan yang belum terungkap (Permatasari, 2020).

Tujuan utama dari pengujian adalah untuk merancang serangkaian pengujian yang secara sistematis dapat mengidentifikasi berbagai jenis kesalahan dengan efisiensi waktu dan usaha yang optimal. Hal ini sangat penting untuk meningkatkan kualitas perangkat lunak dan meminimalkan risiko kesalahan yang dapat merugikan pengguna (Permatasari, 2020).

Berdasarkan pendapat ahli diatas, maka pengujian merupakan tahap krusial dalam pengembangan perangkat lunak, yang bertujuan untuk mendeteksi kesalahan atau bug dalam program. Selain itu, pengujian yang dirancang dengan baik memiliki potensi tinggi untuk mengidentifikasi kesalahan yang belum terungkap, serta harus dilakukan dengan efisiensi waktu dan usaha.

2.8.2 Pengujian *Blacbox*

Metode *Black Box Testing* adalah pengujian yang berfokus pada kebutuhan fungsional dari perangkat lunak. Dengan metode ini, penguji dapat menentukan serangkaian kondisi masukan yang valid dan memastikan bahwa keluaran yang dihasilkan sesuai dengan spesifikasi perangkat lunak (Fitriani Dwi Ramadhani, 2022)

Blackbox Testing merupakan jenis pengujian yang melibatkan pengamatan terhadap hasil eksekusi perangkat lunak menggunakan data uji dan evaluasi fungsi-fungsinya. Metode ini diibaratkan seperti memandang sebuah kotak hitam, di mana kita hanya bisa melihat bagian luar tanpa mengetahui isi di dalamnya (Cahyani et al., 2020).

Tujuan pengujian ini adalah untuk memastikan bahwa sistem yang dihasilkan memenuhi kebutuhan dan dapat digunakan dengan baik. Metode *Black Box Testing* yang diterapkan bertujuan untuk menampilkan pesan kesalahan pada aplikasi jika terjadi kesalahan, serta mendeteksi fungsi yang tidak tepat atau hilang saat data dimasukkan (Fitriani Dwi Ramadhani, 2022)

2.9 Penelitian Yang Relevan

2.9.1 Jurnal Nasional

a. Penerapan Aplikasi Pelayanan Desa Berbasis *Mobile* dengan Konsep *Smart Village* di Desa Pegantenan, Kecamatan Pegantenan, Kabupaten Pamekasan

Penelitian pertama berjudul "Penerapan Aplikasi Pelayanan Desa Berbasis *Mobile* dengan Konsep *Smart Village* di Desa Pegantenan, Kecamatan Pegantenan, Kabupaten Pamekasan" diterbitkan dalam Jurnal Pengabdian Masyarakat Berkemajuan dengan p-ISSN: 2614-5251 dan e-ISSN; 2614-526X Volume 4, Nomor 1, 646 – 652 Pada tahun 2020. Penulisnya adalah Yanuar Risah Prayogi dan kawan-kawan.

Penelitian ini bertujuan untuk mengembangkan aplikasi layanan desa berbasis *mobile* di Desa Pegantenan guna memfasilitasi layanan pemerintah dan meningkatkan komunikasi antara pemerintah dan masyarakat. Penelitian dilakukan di Desa Pegantenan, Kecamatan Pegantenan, Kabupaten Pamekasan, menggunakan metodologi yang melibatkan pengumpulan data melalui survei, wawancara langsung, serta pengumpulan data desa seperti data populasi, informasi keuangan desa, dan catatan korespondensi.

Penelitian ini memiliki kekuatan dalam menerapkan konsep Desa Cerdas untuk meningkatkan transparansi dan efisiensi layanan pemerintah.

implementasi aplikasi ini berhasil menjembatani kesenjangan antara pemerintah dan penduduk, serta berkontribusi pada terwujudnya program Desa Cerdas dan tata kelola desa yang lebih progresif dan transparan.

b. Analisa Penerapan *Server Deployment* Menggunakan *Kubernetes* untuk Menghindari *Single Point of Failure*

Penelitian kedua berjudul "Analisa Penerapan *Server Deployment* Menggunakan *Kubernetes* untuk Menghindari *Single Point of Failure*" ditertbitkan dalam jurnal JINTEKS (Jurnal Informatika Teknologi dan Sains) dengan e-ISSN: 2686-3359 Vol. 3 No. 1, Februari 2021, hlm. 267 – 271. Penulisnya adalah Lilik Widyawati dan kawan-kawan.

Penelitian ini bertujuan untuk menganalisis penerapan *web server* Nginx dan FTP *Server* Fauria menggunakan *Kubernetes* untuk menciptakan sistem terdistribusi dengan tingkat ketersediaan tinggi dan memastikan toleransi kesalahan pada saat terjadi kegagalan *server*.

Hasil uji coba menunjukkan bahwa penerapan *Kubernetes* efektif dalam menjaga ketersediaan sistem, di mana ketika salah satu atau seluruh *node* dalam sistem mengalami masalah, *Kubernetes* secara otomatis melakukan *self-healing* dengan membentuk *container* baru untuk memastikan fungsionalitas tetap terjaga. Mekanisme *failover* juga terbukti berjalan dengan baik, memungkinkan sistem untuk beroperasi tanpa gangguan meskipun ada gangguan pada *node* tertentu.

c. Sistem Informasi Kependudukan Warga: Studi Kasus Perumahan Tegallega Permai Kabupaten Bogor

Penelitian ketiga berjudul "Sistem Informasi Kependudukan Warga: Studi Kasus Perumahan Tegallega Permai Kabupaten Bogor" diterbitkan

dalam jurnal JIKA (Jurnal Informatika) dengan p-ISSN: 2549-0710 dan e-ISSN: 722-2713 Volume 7 Nomor 1, 63-70 Pada Februari 2023, ditulis oleh Ivan Nur Amanda dan kawan-kawan.

Penelitian ini bertujuan untuk mengembangkan sistem informasi guna mengelola data warga di perumahan Tegallega Permai, memanfaatkan teknologi berbasis IT.

Melalui metode prototipe yang diterapkan dalam penelitian, sistem informasi yang dikembangkan berhasil memberikan kemampuan bagi administrator untuk memantau populasi dan pergerakan penduduk serta meningkatkan layanan kepada penduduk. Hasil penelitian menunjukkan keberhasilan pengembangan sistem yang efektif dalam meningkatkan efisiensi administrasi di kawasan tersebut.

d. Sistem Informasi Antar Warga 'SI-ANWAR' sebagai Solusi Bermasyarakat di Perumahan Tigaraksa Berbasis *Web*

Penelitian keempat berjudul "Sistem Informasi Antar Warga 'SI-ANWAR' sebagai Solusi Bermasyarakat di Perumahan Tigaraksa Berbasis *Web*" yang ditulis oleh Ardhi Abdurrahman Ghozi, Muhammad Taufik Hidayat, dan Khoiru Rohman yang diterbitkan dalam Jurnal Universitas Muhammadiyah Jakarta dengan p-ISSN: 2089-0265 dan e-ISSN: 2598-3016 Volume 11, Nomor 1, halaman 45-52, pada tahun 2020.

Penelitian ini bertujuan untuk memberikan akses informasi digital dan *real-time* kepada warga, serta mengatasi kurangnya transparansi dalam pelaporan keuangan. Dengan adanya sistem ini, diharapkan warga dapat memperoleh informasi yang jelas dan melaporkan masalah kepada ketua RT dengan cara yang lebih efisien.

Hasil dari penelitian ini menekankan bahwa perancangan aplikasi "SI-ANWAR" merupakan langkah efektif menuju peningkatan akses informasi, efisiensi pelaporan, dan kontrol terhadap kondisi lingkungan di perumahan. Aplikasi ini berhasil menghadirkan solusi nyata terhadap masalah yang ada di lingkungan perumahan

e. **Implementation *Autoscaling Container Web Server* using *Kubernetes Promox-Based* on *Server University* of Darussalam Gontor**

Penelitian kelima berjudul "*Implementation Autoscaling Container Web Server* using *Kubernetes Proxmox Based on Server University of* Darussalam Gontor" diterbitkan oleh Jurnal Rekayasa Sistem dan Industri dengan e-ISSN: 2579-914 dan 2 p-ISSN: 2356-0843 Volume 6 Nomor 01 Pada tahun 2019 yang ditulis oleh Imron Rosyadi dan kawan-kawan.

Penelitian ini bertujuan untuk mengimplementasikan *virtualisasi server* dengan Proxmox dan autoscaling menggunakan *Kubernetes*, untuk meningkatkan performa *web server* dan menangani lonjakan permintaan yang tinggi pada *server* Universitas Darussalam Gontor.

Hasil penelitian menunjukkan bahwa penerapan *virtualisasi* dan autoscaling berhasil meningkatkan kinerja *web server* berdasarkan tiga parameter, yaitu *Throughput*, yang mencapai 4494.34 KB/s; *Response Time*, meningkat sebesar 14.46 *requests*/detik; dan *CPU Usage*, yang lebih efisien dengan pengurangan 13.01%

f. **Penerapan Aplikasi Sistem Informasi Forkomling Pada Perumahan Berkala Asri**

Penelitian keenam berjudul "Penerapan Aplikasi Sistem Informasi Forkomling Pada Perumahan Berkala Asri" diterbitkan dalam Jurnal

Pengabdian kepada Masyarakat Nusantara (JPkMN) dengan e-ISSN: 2745-4053, pada Volume 5, Nomor 1, halaman 983-989, pada tahun 2024 Penelitian ini ditulis oleh Jijon Raphita Sagala dan kawan-kawan.

Penelitian ini bertujuan untuk menerapkan Aplikasi Sistem Informasi Forkomling di Perumahan Bekala Asri guna meningkatkan keamanan, kebersihan, dan proses pengumpulan biaya.

Hasil penelitian menunjukkan bahwa implementasi aplikasi sistem informasi Forkomling di Perumahan Bekala Asri berhasil dilakukan dengan efektif, yang memungkinkan pengelolaan keamanan, kebersihan, pengutipan iuran, dan informasi lainnya secara efisien

g. *High Availability Service* dengan *Multiple Master* pada *Kubernetes Cluster* Menggunakan *Virtualisasi Kernel Based Virtual Machine* (*KVM*)

Penelitian ketujuh ini berjudul "*High Availability Service* dengan *Multiple Master* pada *Kubernetes Cluster* Menggunakan *Virtualisasi Kernel Based Virtual Machine* (*KVM*)" diterbitkan dalam jurnal *Proceeding Applied Business and Engineering Conference* dengan E-ISSN: 2776-2343 Volume 10, halaman 17-19, pada November 2022. Penelitian ini ditulis oleh Nabila Firdha Aisyah dan Muhammad Arif Fadhly Ridha

Penelitian ini bertujuan untuk menguji implementasi *high availability* pada *server* menggunakan teknologi *cluster* dengan *HAProxy* sebagai *load balancer*, serta mengukur kinerja *server* dalam menangani beban lalu lintas dengan mengaplikasikan *virtualisasi KVM* dan *Kubernetes*.

Hasil penelitian menunjukkan bahwa *throughput* tertinggi yang dicapai adalah 1330,6 *requests* per detik pada 500 klien. Selain itu, pengujian

menemukan penggunaan *CPU* terendah sebesar 0,07% dan penggunaan memori terendah sebesar 24% pada *server* utama dalam keadaan *standby*.

h. Analisis Dan Perancangan Sistem Informasi Lingkungan Perumahan Dengan Metode *Waterfall* Berbasis *Web* Pada Perumahan Puri Adika Bogor

Penelitian kedelapan ini berjudul "Analisis dan Perancangan Sistem Informasi Lingkungan Perumahan dengan Metode *Waterfall* Berbasis *Web* pada Perumahan Puri Adika Bogor" diterbitkan dalam jurnal Biner: Jurnal Ilmu Komputer, Teknik dan Multimedia dengan ISSN: 2988-3814, Volume 1, Nomor 6, halaman 1390-1402, pada Februari 2024.

Penelitian ini ditulis oleh Erwin Wahyudi, yang bertujuan untuk menganalisis dan merancang sistem informasi pengelolaan lingkungan perumahan menggunakan metode *Waterfall* berbasis *web*.

Hasil penelitian menunjukkan bahwa sistem informasi yang dirancang sukses diimplementasikan, memungkinkan pengolahan data terkait keamanan, kebersihan, pengutipan iuran, dan informasi lainnya dilakukan secara efektif di Perumahan Puri Adika Bogor. Penelitian ini menghasilkan sistem informasi berbasis *web* yang memungkinkan akses efisien kepada data kependudukan, keuangan, dan aktivitas perumahan bagi warga dan pengurus.

i. Implementasi *High Availability Cluster Web Server* Menggunakan *Virtualisasi Container Docker*

Penelitian kesembilan ini berjudul "Implementasi *High Availability Cluster Web Server* Menggunakan *Virtualisasi Container Docker*" diterbitkan dalam jurnal media informatika budidarma dengan P-ISSN 2614-

5278 dan E-ISSN 2548-8368, Volume 4, Nomor 1, Halaman 9-13 pada tahun 2020.

Penelitian ini ditulis oleh Muhammad Aldi Aditia Putra dan kawan-kawan yang bertujuan untuk menerapkan *load balancing* untuk mengurangi beban trafik di *web server* dengan menggunakan metode *load balancing* algoritma *round robin* dan *least connections* serta membandingkannya dengan *single server*.

Hasil penelitian menunjukkan bahwa sistem *load balancing Haproxy* dengan algoritma *least connections* lebih unggul dibandingkan algoritma *round robin*. Nilai *request* per-detik untuk *least connection* mencapai 2607.141 req/s dengan *throughput* sebesar 9.25 MB/s, sedangkan untuk *round robin* tercatat 2807.171 req/s dan *throughput* 9.30 MB/s.

j. Sistem Informasi Pengaduan Pamsimnas Perumahan Podosugih Kota Pekalongan

Penelitian kesepuluh ini berjudul "Sistem Informasi Pengaduan Pamsimnas Perumahan *Pod*osugih Kota Pekalongan" diterbitkan dalam Jurnal Minfo Polgan dengan p-ISSN: 2089-9424 dan e-ISSN: 2797-3298, Volume 12, Nomor 2, halaman 771-776, pada Juni 2023.

Penelitian ini ditulis oleh Fenilinas AdiArtanto, yang bertujuan untuk mengembangkan Sistem Informasi Pengaduan PAMSIMAS berbasis *Android* di Perumahan *Pod*osugih, Kota Pekalongan, serta meningkatkan kualitas layanan bagi pelanggan melalui pengujian fungsionalitas sistem menggunakan metode *blackbox testing* untuk memastikan kinerja yang optimal**.**

Hasil penelitian menunjukkan keberhasilan dalam desain dan implementasi sistem informasi yang berfungsi untuk merampingkan proses pengelolaan pengaduan, meningkatkan transparansi, dan memperbaiki komunikasi antara penyedia layanan dan masyarakat terkait sistem pasokan air di PAMSIMAS Perumahan *Pod*osugih.

2.9.2 Jurnal Internasional

a. Efficient Resource Utilization in Kubernetes: A Review of Load Balancing Solutions

Penelitian kesebelas ini berjudul "*Efficient Resource Utilization in Kubernetes: A Review of Load Balancing Solutions*" diterbitkan oleh International Journal of Innovative Research in Engineering and Management (IJIREM) dengan e-ISSN: 2350-0557, Volume 10, Halaman 44-48, Pada Dessember 2023.

Penelitian ini ditulis oleh Indrani Vasireddy dan kawan-kawan. Tujuan dari penelitian ini adalah untuk mengeksplorasi berbagai strategi penyeimbangan beban dalam *Kubernetes* dan untuk memberikan gambaran menyeluruh mengenai teknik-teknik yang ada, tantangan, serta praktik terbaik yang dapat diadopsi.

Hasil penelitian menunjukkan bahwa pemahaman tentang penyeimbangan beban di lingkungan orkestrasikan kontainer dinamis adalah semakin penting, dengan fokus pada *trade-off* antara kecepatan respons, *throughput*, dan adaptabilitas untuk memenuhi berbagai beban kerja

b. Traffic-Aware Horizontal Pod Autoscaler in Kubernetes-Based Edge Computing Infrastructure

Penelitian ke dua belas ini berjudul "*Traffic-Aware Horizontal Pod Autoscaler in Kubernetes-Based Edge Computing Infrastructure*" diterbitkan oleh IEEE Access. dengan e-ISSN: 2169-3536, Volume 10, Halam 18969-18977, pada Februari 2022.

Penelitian ini ditulis oleh Le Hoang Phuc dan kawan-kawan. Tujuan dari penelitian ini adalah untuk meningkatkan kualitas layanan *Internet of Things (IoT)* di lingkungan komputasi tepi dengan mengadopsi pengautoscalaan *pod* yang peka terhadap lalu lintas (THPA), yang mengatasi kelemahan dari *Horizontal Pod Autoscaler* (KHPA) yang tidak mempertimbangkan beban lalu lintas.

Hasil penelitian menunjukkan bahwa THPA mampu meningkatkan waktu respons rata-rata dan *throughput* aplikasi IoT sekitar 150% dibandingkan dengan KHPA, yang mengindikasikan pentingnya pengelolaan sumber daya yang lebih baik berdasarkan distribusi lalu lintas jaringan di masing-masing *node*.

c. ***Load-Balancing of Kubernetes-Based Edge Computing Infrastructure Using Resource Adaptive Proxy***

Penelitian ke tiga belas ini berjudul "*Load-Balancing of Kubernetes-Based Edge Computing Infrastructure Using Resource Adaptive Proxy*" Diterbitkan dalam jurnal Sensors, dengan e-ISSN: 1424-8220, Volume 22, Halaman 2869, pada April 2022.

Penelitian ini ditulis oleh Quang-Minh Nguyen dan kawan-kawan. Penelitian ini bertujuan untuk mengatasi masalah yang dihadapi oleh sistem *load balancing* default di *Kubernetes* (*kube-proxy*) yang menyebabkan

keterlambatan dan menurunkan *throughput* aplikasi di lingkungan *edge computing*.

Jurnal ini menyajikan konsep "*Resource Adaptive Proxy* (RAP)" yang mengawasi status sumber daya dari setiap *pod* dan latensi jaringan antar *node* untuk melakukan pengambilan keputusan *load-balancing* yang lebih baik. Evaluasi menunjukkan bahwa RAP secara signifikan meningkatkan *throughput* dan mengurangi latensi dibandingkan dengan mekanisme *load-balancing default kubernetes*, terutama dalam pengaturan dengan banyak permintaan terkonsentrasi dan terdistribusi.

d. Automated Scaling and Load Balancing in Kubernetes for High-Volume Data Processing

Penelitian ke empat belas ini berjudul "*Automated Scaling and Load Balancing in Kubernetes for High-Volume Data Processing*" diterbitkan dalam jurnal ESP Journal of Engineering & Technology Advancements dengan ISSN 2583-2646, Volume 2, Halaman 23-48 pada Maret 2022.

Penelitian ini ditulis oleh Anirudh Mustyala dan Karthik Allam, Penelitian ini bertujuan untuk mengeksplorasi kemampuan *Kubernetes* dalam mengelola pemrosesan data volume tinggi melalui otomatisasi skala dan penyeimbangan beban, serta untuk menunjukkan bagaimana fitur-fitur ini dapat meningkatkan kinerja dan keandalan aplikasi.

Hasil penelitian menunjukkan bahwa *Kubernetes* efektif dalam menangani beban kerja yang fluktuatif dengan mengadaptasi sumber daya secara otomatis. Ini mengurangi *overhead* operasional dan memaksimalkan pemanfaatan sumber daya, sehingga organisasi dapat mencapai kinerja yang lebih baik dan keandalan yang lebih tinggi dalam proses data berbasis *cloud*.

e. ***Kubernetes and Docker Load Balancing: State-of-the-Art Techniques and Challenges***

Penelitian ke lima belas ini berjudul "*Kubernetes and Docker Load Balancing: State-of-the-Art Techniques and Challenges"* diterbitkan dalam International Journal of Innovative Research in Engineering and Management (IJIREM) dengan e-ISSN: 2350-0557, Volume 10, Halaman 49-52 pada Desember 2023.

Penelitian ini ditulis oleh Indrani Vasireddy dan kawan-kawan. Penelitian ini bertujuan untuk mengeksplorasi teknik *load balancing* yang mutakhir dalam konteks *Kubernetes* dan *Docker*, serta tantangan yang dihadapi oleh sistem ini dalam mengelola beban kerja *containerized*.

Hasil penelitian menunjukkan bahwa *Kubernetes* lebih unggul dibandingkan *Docker* Swarm dalam hal *load balancing*, dengan efisiensi yang lebih tinggi dalam distribusi sumber daya, skalabilitas, dan pengelolaan aplikasi. Penelitian ini juga mengidentifikasi berbagai strategi *load balancing* yang dapat meningkatkan kinerja aplikasi dalam lingkungan terdistribusi.

BAB III

METODOLOGI PENELITIAN

3.1 Tinjauan Umum Perusahaan

3.1.1 Sejarah Perusahaan

PT Sinar Solusi Informatika, yang lebih dikenal dengan nama NextG, didirikan atas kesepakatan Bapak Lie Po Kong yang memiliki visi untuk membentuk sebuah perusahaan teknologi yang responsif terhadap kebutuhan akan solusi informasi inovatif di Indonesia.

Proses pendirian perusahaan ini dilatar belakangi oleh tuntutan pasar yang berkembang pesat, yang mendorong keinginan untuk menciptakan suatu entitas yang dapat bersaing di industri teknologi modern.

Perusahaan ini resmi berdiri pada tahun 2022 dan berlokasi strategis di Jalan Kenanga No. 05, Tangerang, Banten, Indonesia. PT Sinar Solusi Informatika berada di bawah naungan Sinar Group, sebuah kelompok usaha yang sudah mapan dan terkenal, yang juga mencakup beberapa entitas lain seperti CV Sinar Mutiara, PT Simut Sakti, dan CV Charlie SPS.

Sinar Group telah memiliki reputasi yang kuat dalam industri sablon printing, dan dengan pembentukan PT Sinar Solusi Informatika, mereka berkomitmen untuk memperluas layanan yang ditawarkan dengan memanfaatkan kemajuan dalam teknologi digital

Bapak Lie Po Kong berperan sebagai Direktur Utama yang visioner dan memiliki pemahaman mendalam mengenai dinamika dunia bisnis digital. Dengan pengetahuan dan pengalaman yang dimilikinya, beliau memfokuskan perusahaan pada pengembangan solusi teknologi yang dapat memfasilitasi kebutuhan bisnis modern.

PT Sinar Solusi Informatika berkomitmen untuk memimpin bisnis di sekitar dunia teknologi informasi. Dengan spesialisasi dalam pengembangan software, perusahaan ini menyediakan berbagai solusi canggih seperti program *Enterprise Resource Planning (ERP),* pembuatan *landing page*, aplikasi *mobile*, dan berbagai *software inovatif* lainnya.

Keberadaan produk-produk tersebut menunjukkan dedikasi perusahaan dalam mendukung kebutuhan digitalisasi bisnis modern, serta memberikan nilai tambah yang signifikan bagi klien-klien mereka.

Dengan tim yang berpengalaman dan berkomitmen, PT Sinar Solusi Informatika terus memperluas portofolio layanannya dan berinovasi dalam menciptakan solusi yang tidak hanya memenuhi harapan klien, tetapi juga membawa dampak positif terhadap industri teknologi di Indonesia.

Perusahaan ini bertekad untuk menjadi salah satu pemimpin dalam bidang teknologi informasi di kawasan, dengan harapan dapat memberikan kontribusi lebih banyak dalam era digital yang sedang berkembang pesat ini.

3.1.2 Struktur Gambaran Organisasi

a. Direktur Utama / Pimpinan

Direktur utama perusahaan memiliki peran yang sangat penting dalam menentukan keberhasilan atau kegagalan perusahaan, terutama dalam bidang teknologi informasi. posisi ini bertanggung jawab atas pengambilan keputusan strategis yang dapat memengaruhi kinerja perusahaan serta inovasi dan efektivitas operasional di sektor IT (Kristiawan, 2022)

Salah satu karakteristik penting dari seorang direktur utama, terutama di bidang teknologi informasi, adalah latar belakang keahlian yang

relevan. Direktur utama dengan pengalaman dan keahlian dalam IT memiliki pemahaman yang mendalam tentang tantangan dan peluang yang dihadapi oleh perusahaan dalam era digital, yang berkontribusi pada kualitas berbagai laporan dan keputusan strategis perusahaan (Kristiawan, 2022).

Tugas Direktur/pimpinan adalah sebagai berikut:

1. Seluruh dewan atau komite eksekutif memberikan visi dan ide-ide kreatif di tingkat tertinggi, serta memimpin rapat umum dengan tujuan untuk memastikan penerapan tata tertib, keadilan, dan kesempatan bagi semua untuk berpartisipasi. Mereka juga bertanggung jawab untuk mengatur alokasi waktu untuk setiap agenda, menentukan urutan masalah, memandu diskusi menuju kesepakatan, serta menjelaskan dan merangkum tindakan dan kebijakan yang diambil.
2. Berfungsi sebagai wakil organisasi dalam interaksinya dengan pihak luar, serta berperan penting dalam menentukan susunan dewan dan sub-komite, untuk mencapai keselarasan dan efisiensi. Membuat keputusan sesuai dengan delegasi dari bawahannya atau dalam situasi tertentu yang dianggap perlu, yang diambil dalam rapat-rapat perusahaan.

b. Manajer

Manajer adalah individu yang berperan dalam pengelolaan aspek-aspek tertentu dalam perusahaan dan tidak dapat bekerja secara mandiri. Mereka memerlukan kerjasama dan kontinuitas di berbagai bidang untuk mencapai tujuan organisasi (Rohani et al., 2022).

Tugas seorang manajer adalah untuk memantau kinerja tim secara keseluruhan dan menyampaikan hasil pemantauan tersebut kepada karyawan. Sebuah bisnis dapat beroperasi secara efisien jika didukung oleh faktor-faktor yang berkontribusi terhadap keberhasilan usaha tersebut (Rohani et al., 2022)

Berdasarkan dua pengertian tersebut, dapat disimpulkan bahwa seorang manajer memainkan peran vital dalam pengelolaan dan pengawasan tim dalam perusahaan. Manajer tidak hanya bertanggung jawab untuk memantau kinerja tim, tetapi juga mengandalkan kerjasama dan kontinuitas di berbagai bidang untuk mencapai tujuan organisasi

c. Pengurus proyek

Pengurus projek adalah orang yang memiliki tanggung jawab untuk memastikan bahawa projek yang mereka kelola diselesaikan tepat waktu, sesuai dengan standar kualiti, dan dalam anggaran yang telah disepakati, sehingga dapat memberikan kepuasan kepada semua pihak yang berkepentingan dalam projek tersebut (Utomo et al., 2021) (Alam et al., 2021)

Selain itu, pengurus proyek juga merupakan salah satu elemen yang dapat menentukan keberhasilan atau kegagalan sebuah proyek. Ini disebabkan oleh fakta bahwa semua anggota tim akan selalu menjadikan pengurus proyek sebagai pedoman dalam melaksanakan tugas mereka. Jika pengurus projek memberikan arahan yang tidak akurat, hal ini dapat mengakibatkan projek tidak dapat diselesaikan sesuai dengan jadwal yang telah ditetapkan (Alam et al., 2021).

Tugas pengurus proyek :

1. Menyusun rencana proyek yang meliputi jadwal waktu, distribusi sumber daya, dan anggaran untuk memastikan bahwa setiap elemen proyek dapat terlaksana dengan baik.
2. Mengkoordinasikan tim, termasuk pengembang, desainer, dan pemangku kepentingan lainnya, untuk memastikan kerja sama yang efektif dan komunikasi yang baik
3. Memantau kemajuan proyek secara bergilir untuk memastikan bahwa semua tahap diselesaikan sesuai dengan rencana. Ini juga termasuk mengelola risiko dan masalah yang mungkin muncul selama proses berlangsung.
4. Menjaga komunikasi yang terus-menerus dengan pemangku kepentingan, termasuk klien dan manajemen, untuk memastikan bahwa semua pihak memahami perkembangan proyek dan harapan yang ada
5. Membuat laporan berkala tentang status proyek, termasuk pencapaian penting dan tantangan yang dihadapi, serta mendokumentasikan semua proses untuk referensi di masa depan
6. Memastikan bahwa produk akhir memenuhi standar kualitas yang ditentukan, termasuk pengujian sistem dan evaluasi untuk memastikan bahwa semua fungsionalitas berjalan dengan baik
7. Mengelola proses penutupan proyek, termasuk merencanakan tahap transisi kepada tim operasional dan mengevaluasi kinerja proyek untuk mendapatkan pelajaran yang bisa dipelajari untuk proyek mendatang

d. UI/UX

UI (*User Interface*) dan *UX* (*User Experience*) merupakan salah satu kemajuan teknologi yang memanfaatkan media digital dan internet untuk merancang produk yang dapat ditampilkan dan digunakan dengan baik, serta meningkatkan kenyamanan dan kemudahan pengguna saat menggunakan produk atau layanan tersebut (Haida Dafitri et al., 2023)

UI (*User Interface*) lebih menekankan pada interaksi pengguna melalui desain yang menarik. Umumnya, proses UI dilakukan setelah *UX* (*User Experience*) selesai, dengan menentukan desain untuk tata letak, logo, warna, tipografi, dan elemen lainnya (Haida Dafitri et al., 2023).

Sementara itu, *UX* (user experience) merujuk pada desain yang bertujuan untuk meningkatkan kepuasan pengguna aplikasi melalui pengalaman yang menyenangkan dan fungsionalitas yang ditawarkan selama interaksi antara pengguna atau pengunjung dengan produk (Haida Dafitri et al., 2023)

Dapat disimpulkan bahwa peranan pegawai UI/*UX* sangat krusial dalam proses perancangan produk digital. Pegawai UI/*UX* bertanggung jawab untuk menciptakan antarmuka (UI) yang menarik dan intuitif, serta pengalaman pengguna (*UX*) yang menyenankan dan fungsional

e. *Progammer*

*Prog*ram*mer* merupakan suatu jenis profesi yang bertujuan untuk mengembangkan sistem dengan memanfaatkan bahasa pemrograman. Seseorang yang memiliki kemampuan dalam menulis kode (*syntax*) dan merancang sistem dapat disebut sebagai *prog*ram*mer* (Dicoding, 2021)

f. Devops

DevOps merujuk pada perpaduan antara Development (Pengembangan) dan Operations (Operasional), yang menciptakan lingkungan kolaboratif di mana tim pengembangan dan tim operasional bekerja sama. Dengan menerapkan *DevOps*, kesenjangan antara pengembangan aplikasi dan pengelolaan infrastruktur dapat diatasi, menghasilkan sinergi yang kuat di antara tim.

Dalam pendekatan tradisional, yang sering dikenal sebagai pengembangan *waterfall*, tim pengembangan dan operasional berfungsi secara terpisah dan saling tergantung. Namun, dengan adanya *DevOps*, kolaborasi antara kedua tim dapat berlangsung dengan lebih harmonis (Biznetgio, 2023)

3.2 Identifikasi Kebutuhan

3.2.1 Kebutuhan Perangkat Lunak

Tabel 3.1 Kebutuhan perangkat lunak

Jenis	Keterangan
Sistem operasi	Sistem operasi *windows 10* atau yang terbaru dan sistem operasi *mac OS* 10.15 (*Catalina*) atau yang terbaru, *Linux*: *Linux* 64-bit apa pun (misalnya *Ubuntu, Debian, Fedora*, dll.).
Kode editor	*Visual Studio Code*
Integrated Development Environment (IDE)	*Android* studio
Software development kit	*Flutter* SDK
Alat pengembang tambahan	*Emulator android* ataupun handphone *android*
Alat Baris Perintah	*Terminal*, *shell* atau *command prompt*
Bahasa pemrograman	*Dart, PHP*

3.2.2 Kebutuhan Perangkat Keras

Tabel 3.2 Kebutuhan perangkat keras

Jenis	Keterangan
Komputer atau laptop dengan spesifikasi	1. Minimum 4 *core prosesor* x86_64, direkomendasikan 8 *core* atau lebih 2. *Minimum* 8 GB RAM, direkomendasikan 16 GB atau lebih. 3. Minimum 20 GB ruang kosong, direkomendasikan 52 GB atau lebih. 4. Minimum WXGA (1366 x 768), direkomendasikan FHD (1920 x 1080) atau lebih. 5. GPU yang kompatibel dengan OpenGL ES 3.0.
Smartphone *android* fisik dengan spesifikasi	1. *Prosesor*: Minimal *Quad-core* 1.5 GHz, direkomendasikan *Octa-core* 2.0 GHz atau lebih 2. *Memori*: Minimal 4 GB RAM, direkomendasikan 6 GB RAM atau lebih 3. Ruang penyimpanan: Minimal 32 GB, direkomendasikan 64 GB atau lebih 4. Resolusi layar: Minimal 720 x 1280, direkomendasikan 1080 x 1920 atau lebih 5. Sistem operasi: *Android* 6.0 atau lebih baru

Tabel 3.3 Kebutuhan perangkat keras untuk Kubernetes

Keterangan	Jenis
CPU	2 *Core* untuk *control panel*
RAM	2 GB
Penyimpanan	20 GB

Tabel 3.4 Kebutuhan tambahan

Keterangan	Jenis
Integrasi dengan layanan lain	*Google maps SDK for android* dari *google cloud*

3.3 Pembahasan Metode Yang Digunakan

3.3.1 Pengertian *Kubernetes* & *Load balancing*

Kubernetes adalah alat untuk mengelola aplikasi yang dikontainerisasi. Alat ini bersifat sumber terbuka dan memungkinkan konfigurasi dan otomatisasi yang jelas. Alat ini hadir dalam ekosistem yang luas dan terus berkembang (Rina Noviana, 2022).

Kubernetes berfungsi sebagai *platform* orkestrasi kontainer dan menawarkan kerangka kerja untuk memfasilitasi berbagai fungsinya. Arsitekturnya terdiri dari komponen-komponen penting, termasuk *Pod*, *Label*, dan *Selector* (Ridha & Suhatman, 2022)

Kubernetes adalah *platform* sumber terbuka yang digunakan untuk mengelola beban kerja aplikasi yang berada dalam kontainer, serta menawarkan pengaturan dan otomatisasi dengan pendekatan deklaratif (*Kubernetes*, 2024)

Kubernetes memanfaatkan proses penyebaran melalui kontainer, yang sangat membantu dalam pengelolaan penyebaran dibandingkan dengan metode tradisional atau *virtualisasi*. Evolusi dalam dunia penyebaran juga dapat dilihat melalui ilustrasi di bawah ini.

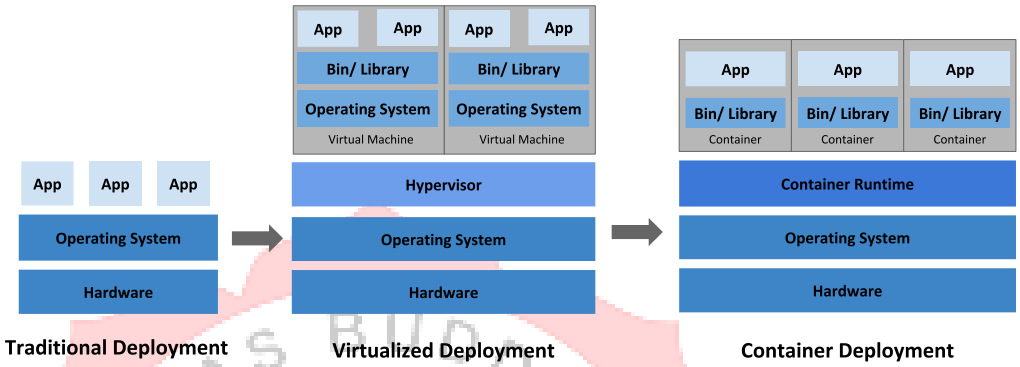
Load balancing adalah metodologi strategis yang digunakan untuk mengalokasikan lalu lintas data dengan cara yang mendukung distribusi beban kerja yang merata di berbagai jaringan. Teknik ini bertujuan untuk meningkatkan pemanfaatan sumber daya dan meningkatkan kinerja sistem secara keseluruhan (Odhi Dwi Putra et al., 2020)

Dengan mengoptimalkan distribusi beban, *Load balancing* bukan hanya meningkatkan kinerja jaringan saja, tapi juga memastikan layanan

yang ada memiliki ketersedian yang tinggi serta mengurangi resiko kegagalan sistem (Khoiru Sabila et al., 2024)

3.3.2 Konteks Riwayat *Kubernetes*

Berikut gambaran perkembangan *deployment* dari masa ke masa



Gambar 3.1 Era penyebaran dari masa ke masa

a. Era *deployment* tradisional

Pada awalnya, organisasi menjalankan aplikasi di *server* fisik. Tidak ada cara untuk menetapkan batasan sumber daya untuk aplikasi yang berjalan di *server* tersebut, yang mengakibatkan masalah dalam alokasi sumber daya.

Sebagai contoh, jika beberapa aplikasi dijalankan pada *server* fisik, satu aplikasi terkadang bisa mengonsumsi sebagian besar sumber daya, sehingga kinerja aplikasi lain terganggu.

Solusinya adalah dengan menjalankan setiap aplikasi pada *server* fisik yang terpisah. Namun, pendekatan ini tidak dapat diperluas karena sumber daya tidak dimanfaatkan secara optimal, dan biaya pemeliharaan banyak *server* fisik menjadi tinggi untuk organisasi.

b. Era penyebaran *virtual*

Sebagai solusinya, konsep *virtualisasi* diperkenalkan. *Virtualisasi* memungkinkan Anda untuk menjalankan beberapa Mesin *Virtual* (VM) di dalam satu *CPU server* fisik. Dengan *virtualisasi*, aplikasi dapat diisolasi di antara VM, memberikan tingkat keamanan yang lebih tinggi, karena data dari satu aplikasi tidak dapat diakses secara langsung oleh aplikasi lainnya.

Virtualisasi juga meningkatkan pemanfaatan sumber daya pada *server* fisik dan meningkatkan skalabilitas, karena aplikasi dapat dengan mudah ditambahkan atau di*upgrade*, sehingga mengurangi biaya untuk perangkat keras dan lainnya. Dengan memanfaatkan *virtualisasi*, Anda dapat menyajikan sekumpulan sumber daya fisik sebagai kluster mesin *virtual* yang dapat digunakan sesuai kebutuhan.

Setiap VM berfungsi sebagai mesin lengkap yang memiliki semua komponen yang diperlukan, termasuk sistem operasinya sendiri, berjalan di atas perangkat keras yang telah ter*virtualisasi*.

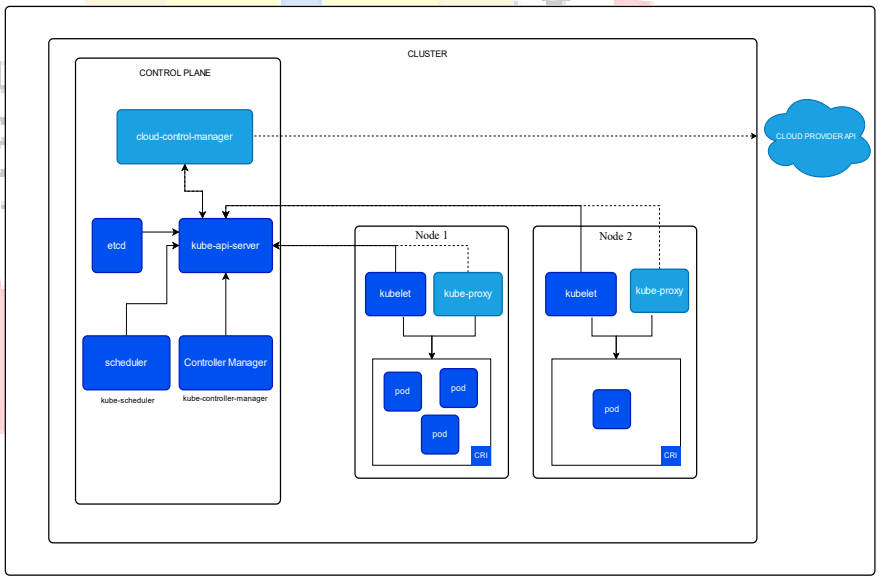
c. Era penyebaran Kontainer

Kontainer serupa dengan Mesin *Virtual* (VM), namun memiliki tingkat isolasi yang lebih rendah dalam berbagi Sistem Operasi (OS) di antara aplikasi. Hal ini membuat kontainer lebih ringan. Seperti halnya VM, kontainer memiliki sistem berkasnya sendiri dan berbagi sumber daya seperti *CPU*, memori, ruang proses, dan lainnya. Karena terpisah dari infrastruktur yang mendasarinya, kontainer dapat dengan mudah dipindahkan antara *cloud* dan distribusi OS

3.3.3 Komponen Arsitektur *Kubernetes*

Klaster *Kubernetes* terdiri dari satu bidang kontrol dan sejumlah mesin pekerja, yang dikenal sebagai *node*, yang menjalankan aplikasi dalam kontainer. Setiap klaster membutuhkan minimal satu *node* pekerja untuk mengoperasikan *Pod*.

Node pekerja berfungsi sebagai tuan rumah bagi *Pod*, yang merupakan elemen dari beban kerja aplikasi. Bidang kontrol bertugas mengelola *node* pekerja dan *Pod* yang terdapat dalam klaster. Dalam konfigurasi produksi, bidang kontrol umumnya dijalankan di beberapa komputer, dan klaster biasanya dapat menjalankan beberapa *node* untuk memastikan toleransi terhadap kesalahan dan tinggi ketersediaan.



Gambar 3.2 komponen arsitektur Kubernetes

Secara garis besar komponen arsitektur *Kubernetes* dipecah menjadi *control plane* dan *node worker*

a. Komponen *control plane*

Komponen dalam bidang kontrol bertanggung jawab untuk mengambil keputusan strategis mengenai klaster, seperti

penjadwalan, serta untuk mendeteksi dan menangani peristiwa yang terjadi dalam klaster, contohnya, menginisiasi klaster baru.

1. *Server* API

   adalah komponen dalam bidang kontrol *Kubernetes* yang menyediakan akses ke API *Kubernetes*. *Server* API bertindak sebagai antarmuka depan untuk bidang kontrol *Kubernetes*

2. *Etcd*

   Penyimpanan nilai kunci yang konsisten dan sangat handal berfungsi sebagai penyimpanan pendukung *Kubernetes* untuk seluruh data klaster.

3. *Kube scheduler*

   Komponen dalam bidang kontrol yang memantau *Pod* yang baru dibuat tanpa penugasan *node*, kemudian memilih *node* untuk mengeksekusinya.

4. *Kube controller manager*

   Kube Controller manager berfungsi untuk mengelola berbagai kontroler yang bertanggung jawab atas pengawasan dan pengelolaan keadaan sistem *Kubernetes*. Ia menjalankan kontroler untuk mendeteksi perubahan dalam status klaster, seperti menambah atau menghapus *Pod*, dan memastikan bahwa keadaan nyata klaster sesuai dengan keadaan yang diinginkan oleh pengguna

b. Komponen *node*

Komponen *node* beroperasi di setiap *node*, mengelola *Pod* yang aktif dan menyediakan lingkungan eksekusi untuk *Kubernetes*.

1. *Kubelet*

   Agen yang beroperasi di setiap simpul dalam klaster ini menjamin bahwa wadah tetap berjalan di dalam *Pod*. *Kubelet* menerima serangkaian *PodSpec* yang diberikan melalui berbagai cara dan memastikan bahwa kontainer yang dicantumkan dalam *PodSpec* tersebut berfungsi dengan baik. *Kubelet* tidak mengelola kontainer yang tidak diciptakan oleh *Kubernetes*.

2. *Kube proxy*

   Kube-proxy adalah *proxy* jaringan yang beroperasi di setiap simpul dalam klaster, mengimplementasikan bagian dari konsep layanan *Kubernetes*. *Kube-proxy* mengatur aturan jaringan di *node*, yang memungkinkan komunikasi jaringan ke *Pod* Anda dari sesi jaringan baik di dalam maupun di luar klaster

3. *Container runtime*

   Komponen fundamental yang memberdayakan *Kubernetes* untuk menjalankan kontainer secara efektif. Komponen ini bertanggung jawab untuk mengelola eksekusi dan siklus hidup kontainer dalam lingkungan *Kubernetes*.

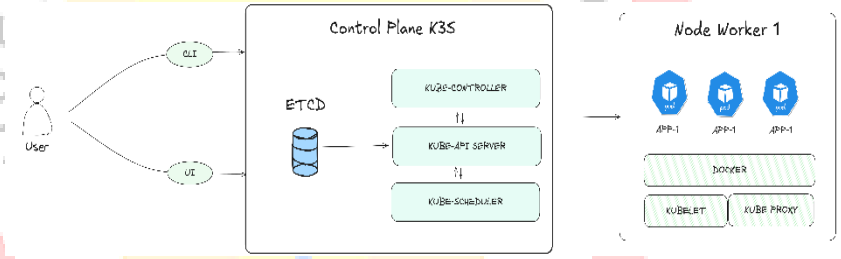
3.3.4 Instalasi *Kubernetes* dengan *K3S*

Ada banyak cara dalam melakukan penginstallan *Kubernetes*, salah satu cara tercepat dan *lightweight* adalah menggunakan *K3S*. *K3s* merupakan distribusi ringan dari *Kubernetes*, yang dirancang untuk memberikan kemudahan dalam pengelolaan kontainer dengan sumber daya yang minimal. *K3s* dikembangkan oleh *Rancher Labs* dan bersertifikat oleh *Cloud Native Computing Foundation* (*CNCF*)

a. Spesifikasi yang akan digunakan

Node	*Prosesor*	RAM
Control plane	4 Core	4 GB
Agent	2 Core	2 GB

b. Gambaran arsitektur yang akan diterapkan



Gambar 3.3 Rancangan Arsitektur kubernetes

c. Instalasi *Kubernetes* dengan *K3S*

Instalasi untuk lingkungan *control plane*

```
export K3S_TOKEN=SECURE@FERITOKEN
curl -sfL https://get.k3s.io | sh -s -- server –disable traefik –disable servicelb --docker
```

Instalasi untuk lingkungan *node worker*

```
export K3S_TOKEN=SECURE@FERITOKEN
curl -sfL https://get.k3s.io | sh -s -- agent --docker --server http://ip:6443
```

3.3.5 Penggunaan dan konfigurasi file *Kubernetes*

Penerapan file konfigurasi *Kubernetes* merupakan langkah penting dalam mengelola dan mendefinisikan sumber daya dalam kluster *Kubernetes*. File konfigurasi ini ditulis dalam format YAML dan memungkinkan pengguna untuk menetapkan spesifikasi yang jelas untuk berbagai objek, seperti *Pod*, layanan, dan *deployment*.

Sebagai contoh, berikut adalah konfigurasi untuk mendefinisikan sebuah *Pod* bernama "*nginx*". Dalam file ini, ditentukan versi API yang digunakan (api*Version*: v1), jenis objek (*kind*: *Pod*), serta metadata yang mencakup nama *Pod*.

Pada bagian spesifikasi (*spec*), diatur kontainer yang akan dijalankan, di mana kontainer dengan nama "*nginx*" menggunakan *image* "*nginx*:1.14.2" dan membuka *port* 80 untuk komunikasi. Konfigurasi ini memungkinkan pengguna untuk dengan mudah menyebarkan aplikasi berbasis kontainer di dalam kluster *Kubernetes* secara konsisten dan teratur.

```
apiVersion: v1
kind: Pod
metadata:
    name: nginx
spec:
   containers:
   - name: nginx
     image: nginx:1.14.2
     ports:
     - containerPort: 80
```

Setelah membuat file konfigurasi ini langkah selanjutnya kita dapat langsung membuat *Pod* dengan file ini, dengan cara mengetikan baris perintah

Kubectl apply -f (lokasi file berada disertai dengan *extension*nya)

3.3.6 Mengakses kedalam *Pod*

Dengan mengikuti langkah diatas kita berhasil membuat *Pod* dan berhasil mendeploy *Pod nginx* kedalam kluster *Kubernetes*. Akan tetapi *Pod* masih terisolasi dari luar dan belum bisa diakses.

Agar *Pod* bisa diakses dari luar, penggunaan *service* dengan tipe *load balancer* akan diterapkan. Secara singkat *service* ini berfungsi sebagai suatu abstraksi untuk mengekspos grup *Pod* ke suatu jaringan

Berkas manifest service

```
apiVersion: v1
kind: Service
metadata:
 name: nginx-service
spec:
 type: NodePort
 selector:
  app.kubernetes.io/name: nginx
 ports:
  - port: 80
    targetPort: 80
    nodePort: 30007
```

3.3.7 *Deployment* dengan banyak *instance* dan penggunaan *load balancer*

Deployment dalam *Kubernetes* diartikan sebagai objek yang mengelola dan mengatur penyebaran aplikasi dengan suatu grup atau keadaan yang dinginkan

Dengan penerapan objek ini dipastikan aplikasi dapat menggunakan skenario replica untuk mendukung kebutuhan *high avaibility* disertai dengan *load balancer*.

Berikut contoh dari file konfigurasi *Kubernetes* dalam penggunaan objek *deployment* dengan *load balancer*.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-service
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
```

```
        image: nginx:1.14.2
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: my-app
```

3.3.8 Pros and cons *deployment* tradisional dibanding *Kubernetes*

*Deploymen t*tradisional bisa dibilang sebagai *single deployment*. Hal Ini memiliki artian sebagai aplikasi yang menyala hanya di satu *server*. Aplikasi mengambil secara penuh kepemilikan *resource server*, cara ini sangat tidak efisien. Bagaimana *server down* ? aplikasi tidak akan bisa digunakan sampai tim devops mengtasi permasalahan ini. Bagaimana jika spek saat ini tidak bisa mendukung kebutuhan *request* yang tiba tiba melonjak? Tim *devops* harus melakukan *upgrade server*.

Setelah melakukan *upgrade server*, frekuensi penggunaan aplikasi tetap tidak bisa ditentukan. Saat penggunaan aplikasi sedang sedikit tentu

dengan spek yang sudah tinggi sangat lah sia-sia jika menggunakan layanan *cloud* berbasi *subscribe* tentu cara ini tidak efektif dan membuat biaya aplikasi sangat tinggi.

Jika penggunaan aplikasi semakin banyak, *server* akan selalu di*upgrade* baik skala *horizontal* atau *vertical*. akan tetapi langkah ini semua tetap dilakukan secara manual dan sangat merepotkan bagi tim *devops*.bagaimana jika *server down*, *server* harus segera di *reboot* dan ini semua dilakukan secara manual.

Menghadirkan solusi otomasi *deployment* aplikasi dengan *Kubernetes*. Setiap perusahan yang menerapkan *Kubernetes* dalam lingkungan produksi tentu akan sangat terbantu. Aplikasi bisa menerima *request* pengguna secara dinamis.

Dengan bantuan *horizontal scalling* dan *load balancer*. Aplikasi dapat disebarkan dibanyak tempat sesuai dengan status *server* saat ini. Misal saja aplikasi A menyala dengan 2 *Pod* dimana disetiap *Pod* mampu menampung 100 *request* jika ditotalkan dengan kondisi ini dapat dipastikan 200 *request* dapat ditangani.

Misal saja saat kondisi *cpu Pod* berada di 80% buat 1 *Pod* lagi jika 2 *Pod* sudah masing-masing mencapai 80% maka 2 *Pod* baru akan dibuat dengan kondisi ini maka total *request* yang dapat ditampung sekitar 400 *request*.

Saat ada *Pod* yang mati, dengan bantuan *Kubernetes* maka akan secara otomatis membuat *Pod* baru. Tentu ini mempermudah tim devops karena semua sudah diotomasi.

3.4 Algoritma Yang Digunakan

Input:

- ServiceCount: Jumlah layanan yang akan dikelola
- Replicas: Array 1D mewakili jumlah replika untuk setiap layanan
- AppConfigurations: Konfigurasi spesifik setiap aplikasi

Output:

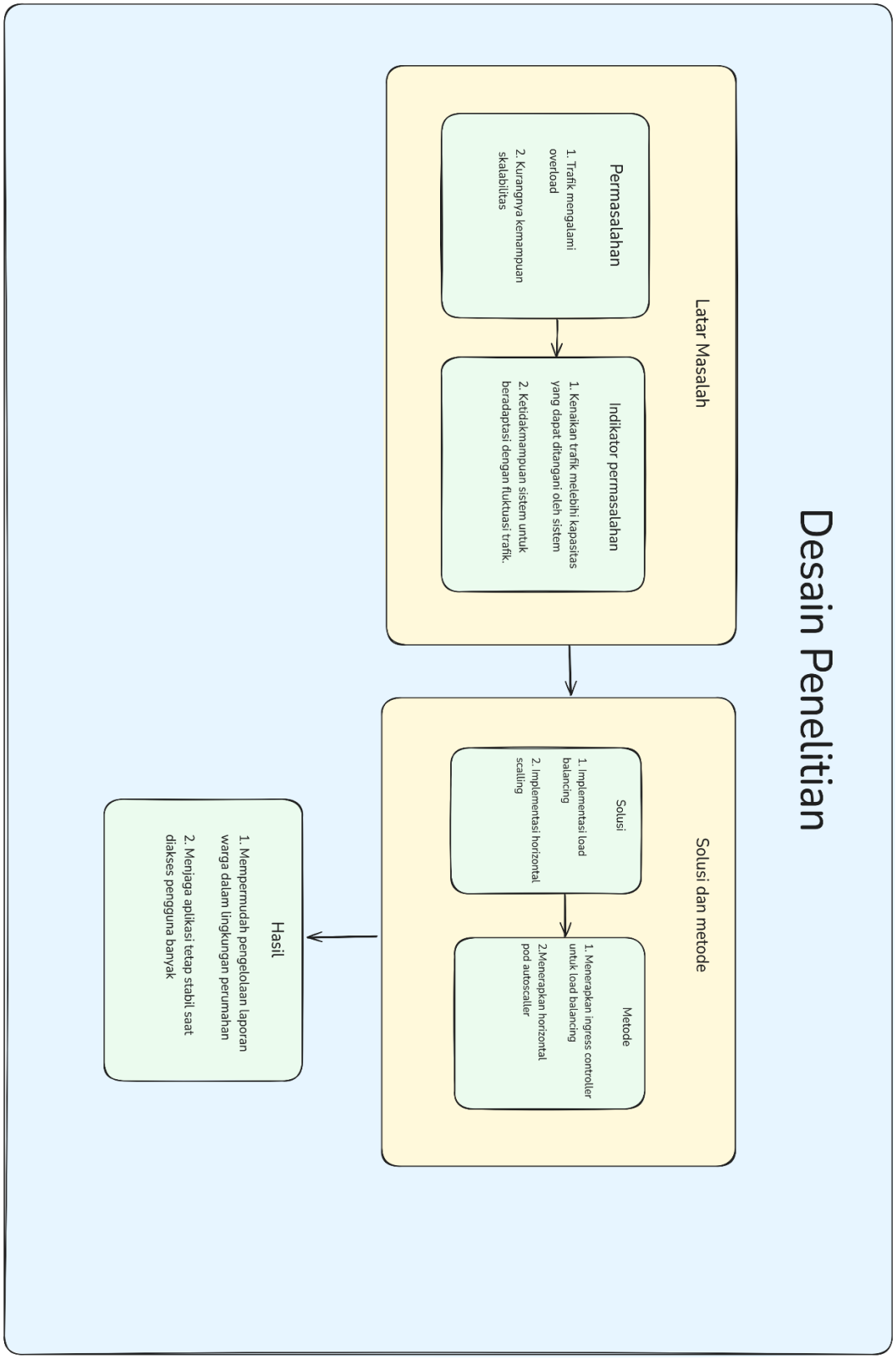
- Status: Status akhir dari deployment dan scaling aplikasi

```
1: function SetupKubernetesCluster()
2:     Initialize Kubernetes Control Plane
3:     Initialize Worker Node
4:     Configure High Availability (HA) settings
5:     return "Cluster Ready"
6: end function

7: function DeployApplications(ServiceCount, AppConfigurations)
8:     for service in 1 to ServiceCount do
9:         // Buat deployment untuk service
10:        DeploymentName = "Deployment-Service" + service
11:        Create Kubernetes Deployment with DeploymentName and AppConfigurations[service]
12:        Set Replicas = Replicas[service]
13:        Apply Deployment
14:    end for
15:    return "Applications Deployed"
16: end function

17: function ConfigureLoadBalancer(ServiceCount)
18:    for service in 1 to ServiceCount do
19:        LoadBalancerServiceName = "LoadBalancer-Service" + service
```

Penjelasan pseudocode

a. Input terdiri dari 3 jumlah layanan (*ServiceCount*), array jumlah *replica* (replicas), dan konfigurasi aplikasi (*AppConfigurations*). Jumlah *node* yang ditetapkan tidak tuliskan secara eksplisit hal ini karena hanya ada *control plane* dan satu *worker node*.
b. Fungsi Setup *Kubernetes Cluster* hanya menginisialisasi satu *control plane* dan satu *worker node*. Ini berguna pada pembuatan arsitektur *cluster*. Tanpa banyak konfigurasi *HA* yang kompleks.
c. *Deploy Applications*, Configure *LoadBalancer*, dan *Enable Autoscaling*, masing masing fungsi ini adalah *object* yang sudah ada di *Kubernetes* yang masing-masing menjalankan dukungan *HA*.

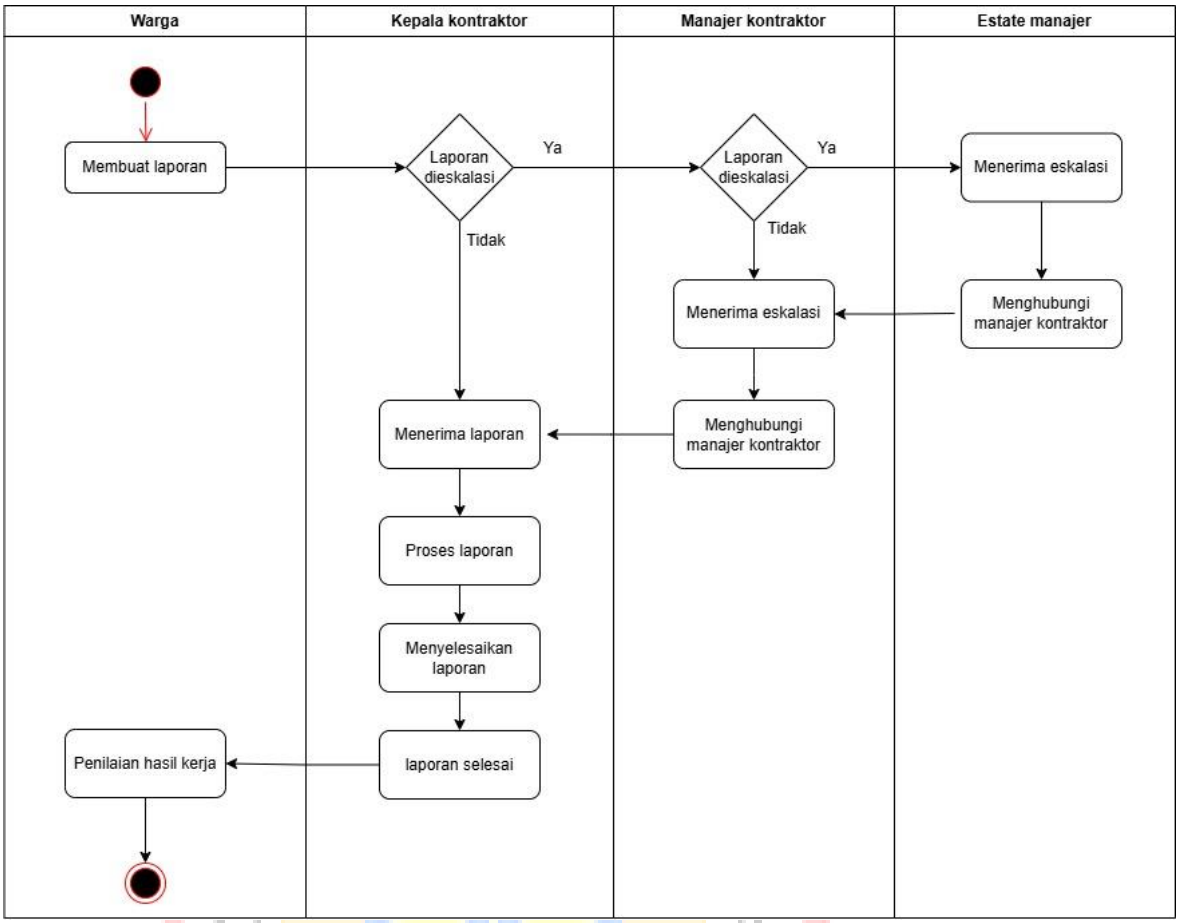
3.5 Kerangka Pemikiran



Gambar 3.3 Desain penelitian

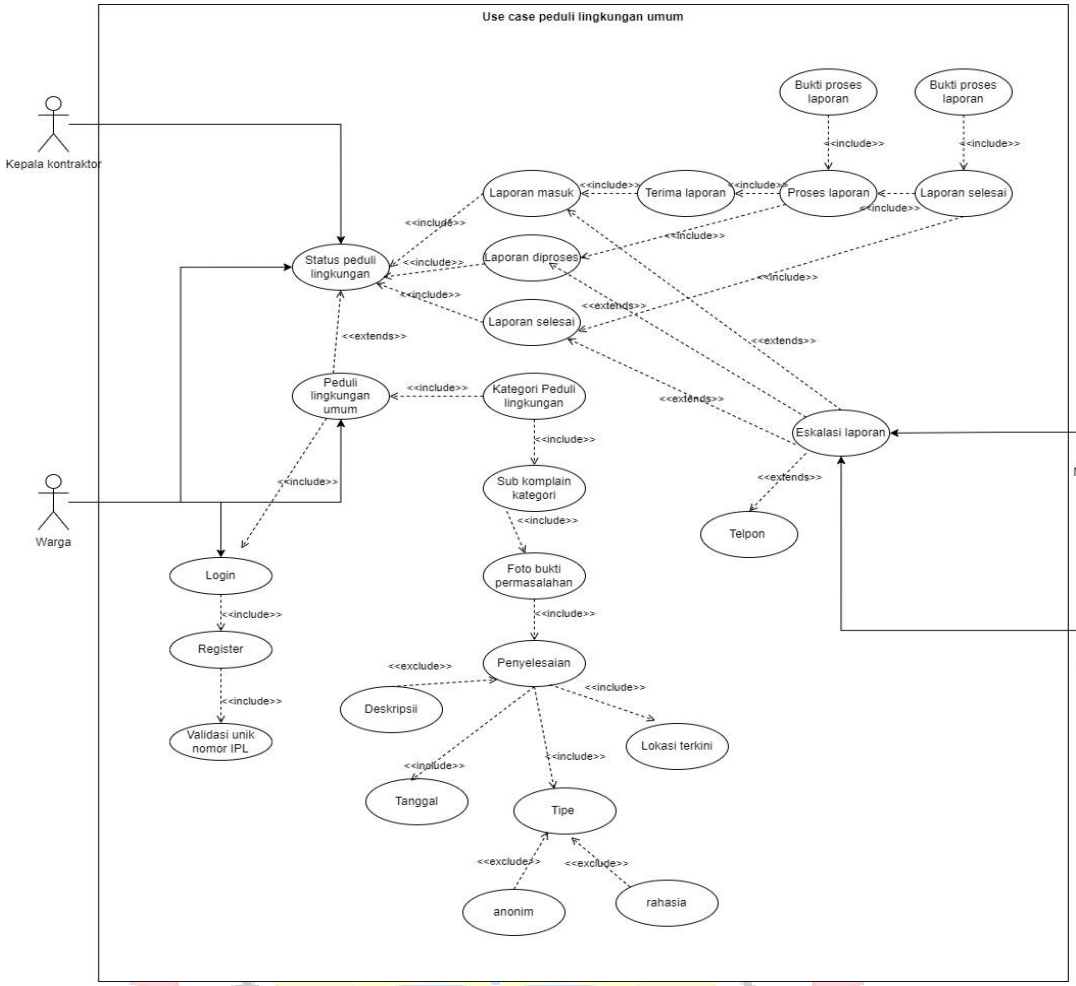
3.6 Perancangan UML

3.6.1 Activity Diagram



Gambar 3.4 Rancang Activity Diagram

3.6.2 *Use case* diagram

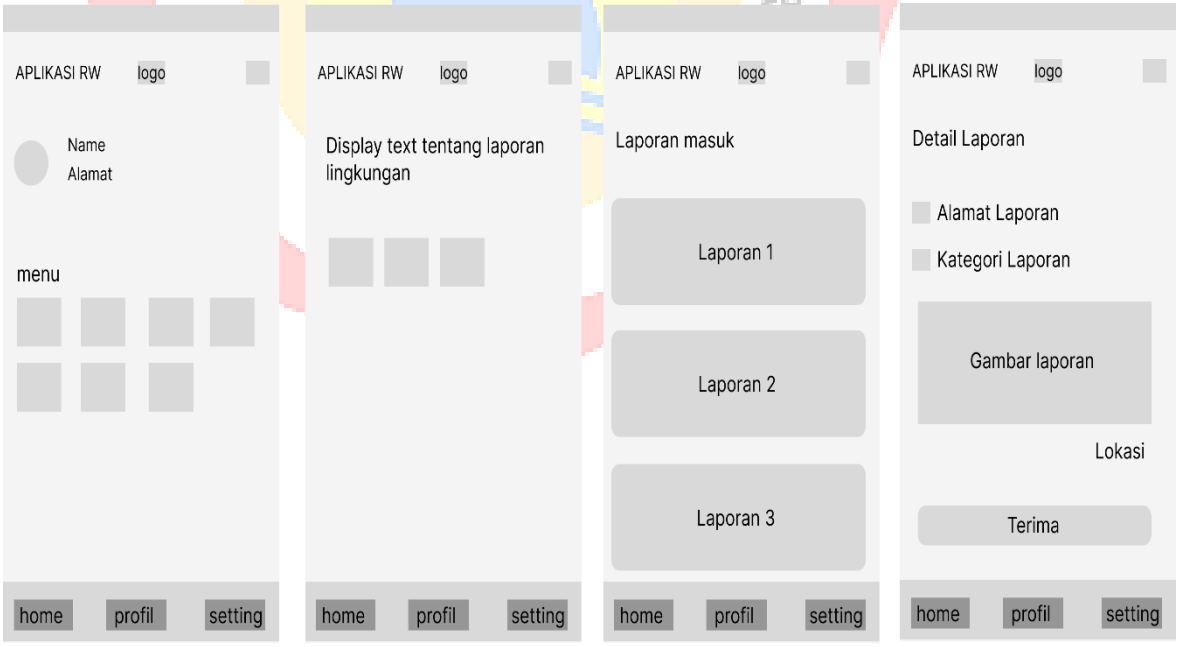


Gambar 3.5 Rancangan Use Case Diagram

3.7 Perancangan Layar, Menu, *Database*

3.7.1 Perancangan Layar Aplikasi

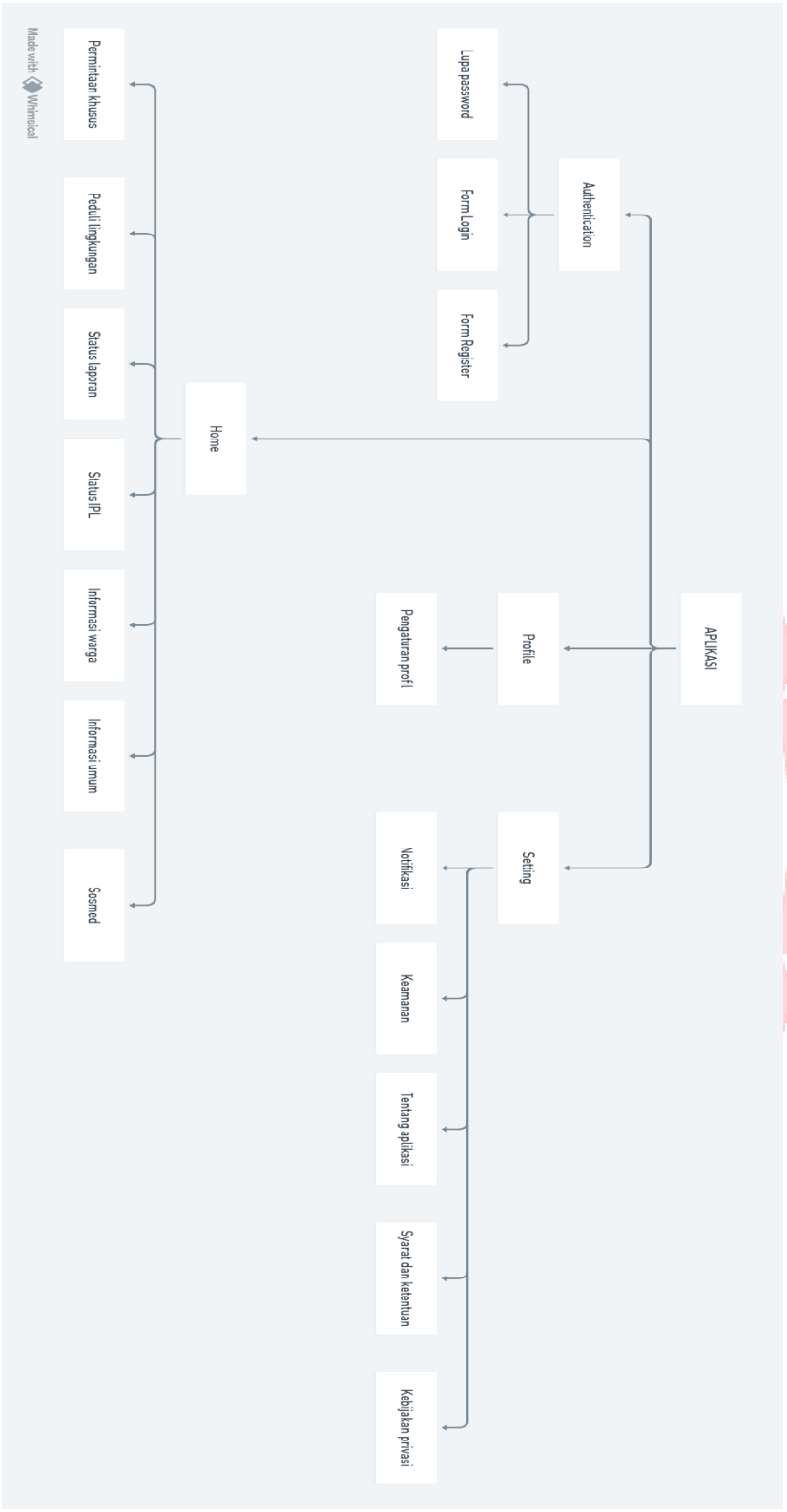
Gambar 3.6 Perancangan laporan aplikasi



Gambar 3.7 Perancangan daftar laporan

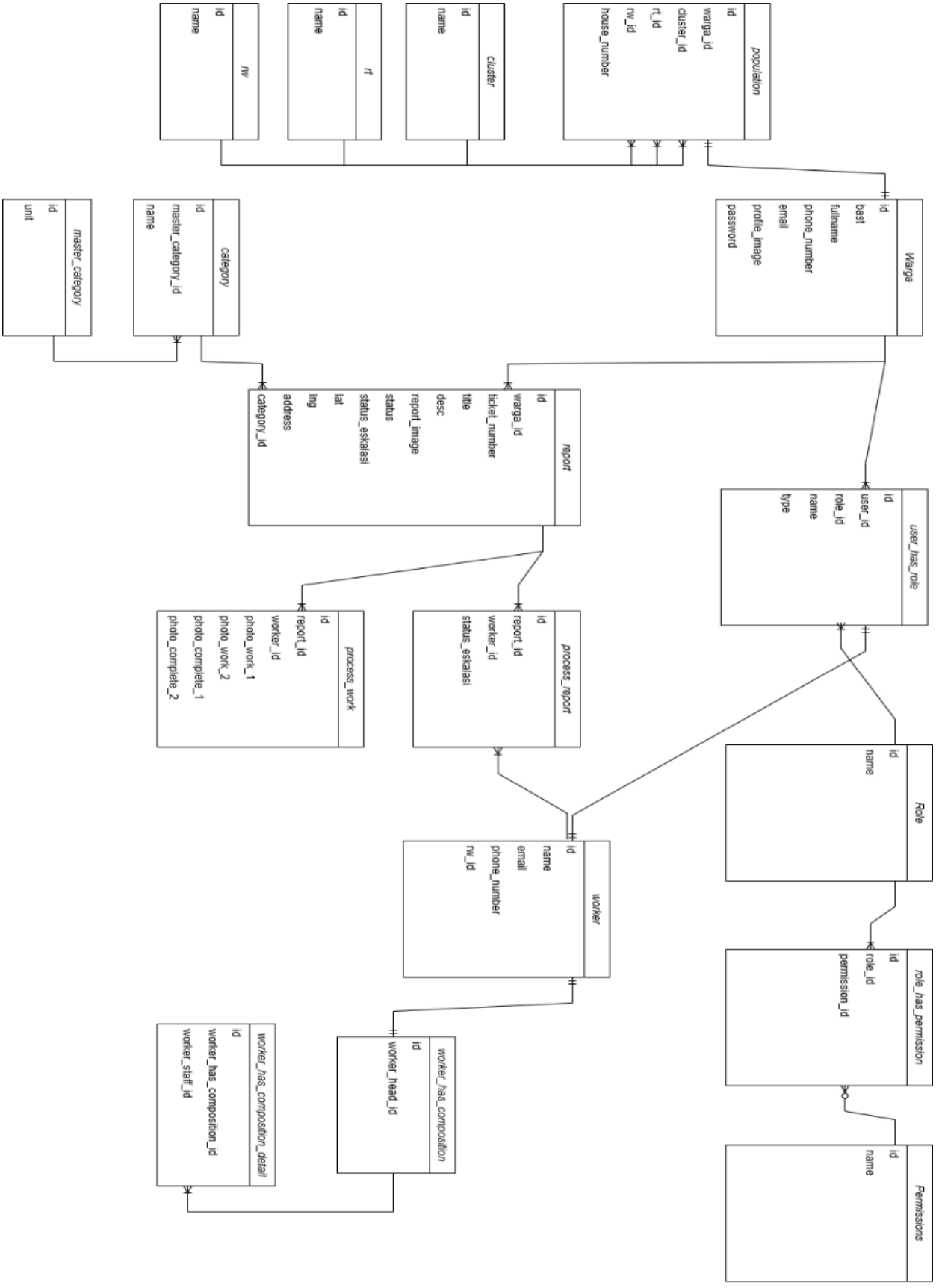
Gambar 3.8 Perancangan pengerjaan laporan

3.7.2 Perancangan Menu



Gambar 3.9 Perancangan Menu

3.7.3 Perancangan Database



Gambar 3.10 Perancangan ERD

3.8 Jadwal Penelitian

Tabel 3.5 Jadwal Penelitian

Jenis Kegiatan	September			Oktober				November				Desember			
	Minggu Ke														
	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Studi Literatur															
Identifikasi Masalah															
Perancangan Sistem															
Penginstalan dan penyiapan *Kubernetes*															
Pemograman Aplikasi															
Pemrograman *Kubernetes*															
Pengujian *Load balancing*															
Pengujian High *Avaibility*															
Dokumentasi Skripsi Bab 1- bab 5															