IMPLEMENTASI *ZEN LOAD BALANCER* PADA SERVER PROTOKOL BERBASIS *DOCKER CONTAINER*

PROJEK

Sebagai salah satu syarat untuk menyelesaikan studi di

Program Studi Teknik Komputer DIII

Oleh:

CHINDY LEANDA PUTRI

09040581923001

PROGRAM STUDI TEKNIK KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SRIWIJAYA

DESEMBER 2023

HALAMAN PENGESAHAN

PROJEK

IMPLEMENTASI *ZEN LOAD BALANCER* PADA SERVER PROTOKOL BERBASIS *DOCKER CONTAINER*

Sebagai salah satu syarat untuk penyelesaian studi di
Program Studi Teknik Komputer DIII

Oleh :

CHINDY LEANDA PUTRI **09040581923001**

Palembang, 29 Desember 2023

Pembimbing I,	**Pembimbing II,**
Ahmad Heryanto, S.Kom., M.T.	**Adi Hermansyah, M.T.**
NIP.198701222015041002	**NIK.161303300489001**

Mengetahui
Koordinator Program Studi Teknik Komputer,

Huda Ubaya, M.T.
NIP.198106162012121003

HALAMAN PERSETUJUAN

Telah diuji dan lulus pada :

Hari : Jumat

Tanggal : 29 Desember 2023

Tim penguji :

1. **Ketua : Dr. Ahmad Zarkasi, M.T.** _____
2. **Pembimbing I : Ahmad Heryanto, S.Kom., M.T.** _____
3. **Pembimbing II : Adi Hermansyah, M.T.** _____
4. **Penguji : Kemahyanto Exaudi,S.Kom., M.T.** _____

Mengetahui

Koordinator Program Studi Teknik Komputer

Huda Ubaya, M.T.

NIP 198106162012121003

SURAT PERNYATAAN

Yang bertanda tangan dibawah ini :

Nama	: Chindy Leanda Putri
NIM	: 09040581923001
Program Studi	: Teknik Komputer
Peminatan	: Teknik Komputer Jaringan
Judul	: Implementasi *Zen Load Balancer* Pada Server Protokol Berbasis *Docker Container*

Hasil Pengecekan Software iThenticate/Turnitin : 11%

Menyatakan bahwa Laporan Projek saya merupakan hasil karya sendiri dan bukan hasil penjiplakan/plagiat, apabila ditemukan unsur penjiplakan/plagiat dalam laporan projek ini, maka saya bersedia menerima sanksi akademik dari Universitas Sriwijaya sesuai dengan ketentuan yang berlaku.

Demikian, pernyataan ini saya buat dengan sebenarnya dan tidak ada paksaan oleh siapapun.

Palembang, 29 Desember 2023

Chindy Leanda Putri

NIM 09040581823001

HALAMAN PERSEMBAHAN

MOTTO

Hal yang tidak bisa kembali adalah waktu,
Jika menunda sesuatu, kita bisa kehilangan kesempatan yang ada,
Jangan sampai kita menyesal karena telah kehilangan sesuatu tersebut yaitu
"*Kesempatan*"
Maka gunakanlah waktumu sebaik mungkin..

Jangan biarkan kesulitan membuatmu gelisah,
. Karena bagaimanapun juga hanya dimalam yang paling gelap
bintang-bintang tampak bersinar lebih terang.
- Ali bin Abi Thalib –

"Mindset yang baik akan menghasilkan energi yang positif"
Kita punya kemampuan yang lebih, dari sekedar apa yang kita pikirkan.
-Chindy Leanda Putri-

PERSEMBAHAN

Dengan mengucap syukur alhamdulillah atas Allah Subhanahu wa Ta'ala, kupersembahkan karya kecil ini untuk…

Kedua orang tua tercinta

(Bapak Indra dan Ibu Masdalena)

Kedua saudaraku tersayang

(Cheza Leanda dan Chira Zifva ilcizava)

Almamater perjuangan

(Universitas Sriwijaya)

Desember 2023

KATA PENGANTA R

Segala puji dan syukur atas kehadiran Allah SWT, karena berkat rahmat dan karunia-Nyalah penulis dapat menyelesaikan penulisan projek akhir ini dengan judul "**IMPLEMENTASI *ZEN LOAD BALANCER* PADA SERVER BERBASIS *DOCKER CONTAINER*** ". Penulisan projek akhir ini dibuat dalam rangka memenuhi persyaratan untuk menyelesaikan pendidikan di Program Studi Teknik Komputer Fakultas Ilmu Komputer Universitas.

Pada kesempatan ini, penulis mengucapkan terima kasih kepada semua pihak yang telah membantu memberikan ide-ide masukan, membimbing, dan terus mendukung penulis dalam menyelesaikan Projek Akhir ini di antaranya :

1. Allah SWT yang telah memberikan hamba kesehatan, kemudahan, dan kelancaran sehingga hamba dapat menyelesaikan laporan projek Akhir ini dengan selesai.
2. Kedua orang tua serta keluarga yang telah memberikan dukungan dan do'a untuk kelancaran penyelesaian laporan projek akhir ini.
3. Bapak Ahmad Heryanto, S.Kom., M.T. dan Bapak Adi Hermansyah, M.T. selaku Dosen Pembimbing I dan II projek akhir, yang telah memberikan bimbingan, arahan baik dan semangat kepada penulis dalam menyelesaikan projek akhir.
4. Bapak Huda Ubaya, M.T. selaku Koordinator Program Studi Teknik Komputer Universitas Sriwijaya.

5. Bapak Kemahyanto Exaudi, S.Kom., M.T. selaku Dosen penguji sidang projek yang telah memberikan kritik dan saran serta ilmu yang bermanfaat sehingga laporan ini menjadi lebih baik.
6. Seluruh Dosen Program Studi Teknik Komputer, Fakultas Ilmu Komputer Universitas Sriwijaya.
7. Staff di Program Studi Teknik Komputer, khususnya Mba Faula selaku Admin yang telah membantu penyelesaian proses Administrasi.
8. Keluarga Besar Fakultas Ilmu Komputer Universitas Sriwijaya, bagian Akademik, Kemahasiswaan, Tata Usaha, Perlengkapan dan Keuangan.
9. Seluruh Pimpinan yang ada pada lingkungan Fakultas Ilmu Komputer, Universitas Sriwijaya.
10. Teruntuk teman-teman satu angkatan, khususnya Teknik Komputer Jaringan 2019. Semoga sehat dan sukses untuk kita semua.

Akhir kata penulis berharap semoga laporan projek akhir ini dapat bermanfaat bagi pembaca khususnya Mahasiswa Program Studi Teknik Komputer Fakultas Ilmu Komputer Universitas Sriwijaya. Semoga laporan projek akhir ini menjadi lebih baik di masa mendatang. Terima kasih.

Palembang, 29 Desember 2023

Chindy Leanda Putri

IMPLEMENTASI *ZEN LOAD BALANCER* PADA SERVER

PROTOKOL BERBASIS *DOCKER CONTAINER*

Oleh :

Chindy Leanda Putri
09040581923001

ABSTRAK

Load balancing adalah strategi yang diterapkan untuk mendistribusikan lalu lintas aplikasi secara merata ke sejumlah server backend, sehingga memastikan ketersediaan dan kinerja yang optimal. *Zen Load Balancer* adalah solusi open-source yang dapat digunakan unmtuk melaksanakan fungsi load balancing dengan mudah dan efisien. *Docker* adalah platform yang memungkinkan pengembang untuk mengemas, mendistribusikan, dan menjalankan aplikasi dalam *container*. Penggunaan *Zen Load Balancer* dalam lingkungan *Docker Container* dapat membantu mengatasi tantangan dalam manajemen lalu lintas yang kompleks dan meningkatkan skalabilitas aplikasi. Penelitian ini untuk memberikan pemahaman yang lebih baik tentang bagaimana Zen Load Balancer dapat dioptimalkan untuk meningkatkan kinerja *server* dalam lingkungan Docker container.

Kata Kunci: *Load balancing, Zevenet, Docker, container*, skalabilitas, ketersediaan.

IMPLEMENTATION OF ZEN LOAD BALANCER ON THE SERVER DOCKER CONTAINER BASED PROTOCOL

***By* :**

Chindy Leanda Putri
09040581923001

ABSTRACT

Load balancing is a strategy implemented to distribute application traffic evenly across a number of backend servers, thereby ensuring optimal availability and performance. Zen Load Balancer is an open-source solution that can be used to carry out load balancing functions easily and efficiently. Docker is a platform that allows developers to package, distribute, and run applications in containers. Using Zen Load Balancer in a Docker Container environment can help overcome challenges in complex traffic management and improve application scalability. This research is to provide a better understanding of how Zen Load Balancer can be optimized to improve server performance in a Docker container environment.

Keywords*: Load balancing, Zevenet, Docker, containers, scalability, availability.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada era digital saat ini, kebutuhan informasi semakin meningkat dengan seiringnya perkembangan teknologi yang mempengaruhi penggunaan jumlah layanan data. Semakin banyak *user* yang mengakses *server*, maka beban kerja pada *server* akan semakin berat dan bisa mengalami suatu kegagalan, hal ini disebabkan oleh jumlah permintaan data oleh *client* yang sangat banyak, sehingga *server* tidak mampu menghadapi lonjakan data tersebut dalam jangka waktu yang bersamaan, istilah ini biasa disebut *overload request*[1].

Load balancing dapat memproses pendistribusian data secara efisien, saat *server* atau aplikasi menerima *request* dari luar maka *load balancer* akan membagikan *traffic* tersebut secara merata ke beberapa *server* yang telah tersedia, otomatis penggunaan aplikasi akan berjalan dengan optimal, sehingga dapat memaksimalkan *throughput* dan mengatasi *overload* (kelebihan) beban pada salah satu koneksi[2][1].

Dalam Pengembangan Perangkat Lunak, teknologi *virtual container* saat ini semakin umum dan banyak diminati karena sifatnya yang lebih portabel hemat sumber daya dalam menjalankan aplikasi ke dalam unit yang dapat dipindahkan secara cepat dan bebas[3]. Salah satu *container* yang populer yaitu *docker*, layanan yang memiliki kemampuan dilingkungan terisolasi, untuk menjalankan aplikasi di dalam *Container.*

Perancangan suatu *load balancing* akan menggunakan *Zevenet load balancer* yang memiliki kinerja yang sesuai dengan kebutuhan untuk mengatur distribusi pengolahan ke beberapa *server*. Berdasarkan penjelasan tersebut maka Penulis tertarik untuk melakukan Penelitian lebih lanjut kedalam Projek dengan Judul "**Implementasi *Zen Load Balancer* Pada Server Protokol Berbasis *Docker Container"***

1.2 Rumusan Masalah

Berdasarkan pemaparan latar belakang yang dikemukakan diatas, maka perumusan masalah pada perancangan ini secara umum ialah:

1. Bagaimana mengimplementasikan *load balancing server* menggunakan metode *docker*?
2. Bagaimana meningkatkan skalabilitas dari *load balancing* pada *server*?

1.3 Tujuan Penelitian

Adapun tujuan yang ingin dicapai dari pembuatan projek ini yaitu :

1. Untuk mengimplementasikan *load balancing server* menggunakan metode *docker.*
2. Untuk meningkatkan skalabilitas dari *load balancing* pada *server.*

1.4 Manfaat Penelitian

Adapun manfaat penelitian yang diharapkan ialah sebagai berikut:

1. Memberikan pemahaman yang lebih baik tentang cara mengimplementasikan *load balancing server* menggunakan metode *docker*
2. Meningkatkan kinerja *load balancing* pada *server*

1.5 Metode Penelitian

Adapun metodelogi penelitian yang digunakan pada penelitian ini sebagai berikut:

1. Metode Literatur

   Metode Literatur untuk memperoleh pemahaman yang komprehensif tentang *load balancing server* untuk metode *docker*, dan integrasi antar keduanya. Melalui studi literature, peneliti dapat mengumpulkan informasi tentang konsep, prinsip dan praktik terkait yang relevan dengan penelitian ini.

2. Perancangan Sistem

   Perancangan sistem ini melibatkan pemilihan versi yang sesuai penentuan konfigurasi yang diperlukan. Dan perancangan penggunaan *dokcer* untuk mengelola *container* yang akan menjalankan *server.*

3. Implementasi

   Implementasi ini melibatkan mengunduh dan menginstal perangkat lunak yang diperlukan untuk membuat file konfigurasi dan membangun, menjalankan *container docker.*

4. Pengujian dan Evaluasi

   Pengujian ini dapat melibatkan uji beban untuk mengukur performa *load balancer* dalam menangani lalu lintas web, pengujian *skalabilitas* untuk menguji kemampuan sistem dalam menangani peningkatan beban, dan

perbandingan dengan implementasi non-*docker* untuk mengevaluasi manfaat penggunaan *docker*.

5. Pembahasan dan Kesimpulan

Pembahasan melibatkan interpretasi temuan, perbandingan dengan studi literatur, dan penjelasan mengenai manfaat, keuntungan, dan tantangan yang ditemui selama penelitian. Kesimpulan ditarik berdasarkan analisis dan pembahasan hasil penelitian.

1.6 Sistematis Penulisan

Sistematika penulisan dalam laporan projek ini dikelompokkan dalam 5 bab dengan masing-masing pokok pembahasan yang telah disusun sebagai berikut:

BAB I PENDAHULUAN

Bab ini akan mengartikan perihal latar belakang masalah, rumusan masalah, tujuan, manfaat, dan metode perancangan dari teori implementasi *zen load balancer* pada *server* menggunakan metode *docker*.

BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan perihal sesuatu yang akan digunakan untuk dasar penulisan pada projek penelitian yaitu tentang implementasi *zen load balancer* pada *server* menggunakan metode *docker*.

BAB III METODELOGI PENELITIAN

Bab ini menjelaskan tentang metode penelitian yang digunakan, desain sistem yang akan di implementasikan, langkah-langkah implementasi secara rinci, pengujian serta evaluasi alat dan perangkat lunak yang akan digunakan.

BAB IV HASIL DAN PEMBAHASAN

Bab ini menjelaskan tentang hasil akhir pengujian yang dilakukan, pengumpulan data, analisis data, interprestasi hasil pengujian, dan diskusi tentang temuan penelitian, analisis data, perbandingan, manfaat, keuntungan dan tantangan yang ditemui dalam implementasi *zen load balancer* pada *server* menggunakan metode *docker*.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan tentang pengujian yang telah dilakukan dan serta berisikan saran-saran untuk dipraktikan pada perancangan selanjutnya.

BAB II

TINJAUAN PUSTAKA

2.1 Load Balancing

Load balancing merupakan sebuah konsep distribusi *traffic* jaringan atau permintaan aplikasi dalam sekolompok *server*. Dengan menggunakan teknik *load balancing* website dapat membagi beban kinerja *server* dengan stabil saat mengalami peningkatan *traffic*[4]. Melalui proses ini, layanan yang diakses dapat berjalan secara optimal dan tidak *overload* (kelebihan) beban pada salah satu jalur koneksi.

Load balancing juga bisa di katakan sebagai penggabungan dua buah jaringan atau lebih untuk di gabungkan ke dalam *router* dan di sambungkan ke *server* serta *client*. ketika sebuah *server* sedang diakses oleh para pengguna, maka *server* tersebut sedang terbebani karena harus melakukan proses permintaan kepada para penggunanya[5][6]. Jika penggunanya banyak maka prosesnya akan lama.

Peran utama *load balancing* adalah untuk mendistribusikan beban kerja secara merata didalam *server*, tujuannya untuk meningkatkan kinerja, ketersediaan, dan keandalan sistem komputer dan jaringan. Ketika pengguna mengirimkan permintaan ke aplikasi yang diakses melalui *load balancer*, permintaan ini akan diterima dan sistem akan mengeksekusi tugas ke *server* yang ada, jika *server* tersebut sudah hampir penuh, maka *load balancer* akan mendistribusikan ke *server* lain yang masih kosong[7].

2.1.1 Algoritma Load Balancing

1. *Round Robin Algoritma*

Round robin merupakan algoritma yang paling sederhana dan banyak digunakan oleh perangkat *load balancing.* Algoritma ini membagi secara bergiliran dan berurutan dari satu *server* ke *server* lain sehinga membentuk putaran.

2. *Ratio*

Ratio (rasio) sebenarnya merupakan sebuah parameter yang diberikan untuk masing-masing server yang akan dimasukkan kedalam sistem *load balancing.* Dari parameter *ratio* ini, akan dilakukan pembagian beban terhadap *server-server* yang diberi rasio. *server* dengan rasio terbesar diberi beban besar, begitu juga dengan *server* dengan rasio kecil akan lebih sedikit diberi beban.

3. *Fastest*

Algoritma yang satu ini melakukan pembagian beban dengan mengutamakan *server - server* yang memiliki respon yang paling cepat. *Server* di dalam jaringan yang memiliki respon paling cepat merupakan *server* yang akan mengambil beban pada saat permintaan masuk.

4. *Least Connection*

Algoritma *least connection* akan melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah *server*. *Server* dengan

pelayanan koneksi yang paling sedikit akan diberikan beban yang berikutnya akan masuk.

2.1.2 Fitur *Load Balancing*

Beberapa fitur yang ada pada baik *load balancer hardware* maupun *load balancer software*, yaitu:

1. *Asymmetric load* rasio dapat dibuat dengan menentukan koneksi yang menjadi primary yang dianggap paling baik backbonenya dan terbaik dalam path routingnya, jadi kita dapat membuat mesin untuk mencari best path determination dan routing yang terpendek dan terbaik untuk sampai ketujuan.
2. Aktivitas berdasarkan prioritas. Disaat *load* jaringan sedang peek, *server* akan dapat membagi aktivitas berdasarkan prioritas dan ke link cadangan.
3. Proteksi dari serangan DDoS. karena kita dapat membuat fitur seperti *SYN Cookies* dan *delayed-binding* (suatu metode di *back-end server* 16 pada saat terjadi proses *TCP handshake*) pada saat terjadi serangan *SYN Flood*.
4. Kompresi HTTP. Memungkinkan data untuk bisa mentransfer objek HTTP dengan dimungkinkannya penggunaan utilisasi kompresi gzip yang berada di semua *web browser* yang modern.
5. *TCP Buffering*. dapat membuat respon *buffer* dari *server* dan berakibat dapat memungkinkan task akses lebih cepat.

6. HTTP Caching. dapat menyimpan content yang static, dengan demikian request dapat di handel tanpa harus melakukan kontak ke web server diluar jaringan yang berakibat akses terasa semakin cepat.
7. *Content Filtering*. Beberapa *load balancing* dapat melakukan perubahan trafik pada saat dijalankan.
8. *HTTP Security*. beberapa sistem *Load Balancing* dapat menyembunyikan *HTTP error pages*, menghapus identifikasi *header server* dari respon HTTP, dan melakukan enkripsi *cookies agar user* tidak dapat memanipulasinya.
9. *Priority Queuing*. berguna untuk memberikan perbedaan prioritas *traffic* paket.
10. *Spam Filtering*. Spam istilah lainnya junk mail merupakan penyalahgunaan dalam pengiriman berita elektronik untuk menampilkan berita iklan dan keperluan lainnya yang mengakibatkan ketidaknyamanan bagi para pengguna *web*. Bentuk berita spam yang umum dikenal meliputi: spam surat elektronik, spam instant messaging, 17 spam Usenet newsgroup, spam mesin pencari informasi *web (web search engine spam*), spam blog, spam berita pada telepon genggam, spam forum Internet, dan lain lain. Spam ini biasanya datang bertubitubi tanpa diminta dan sering kali tidak dikehendaki oleh penerimany. [8]

Ada beberapa metode *Load Balancing* yang bisa diterapkan, antara lain:

1. Penyeimbangan Beban Berbasis Server: Mendistribusikan beban di antara beberapa *server* untuk memastikan setiap *server* menerima jumlah permintaan klien yang kira-kira sama.
2. Penyeimbangan Beban Berbasis Jaringan: Menggunakan perangkat jaringan khusus untuk mengelola distribusi lalu lintas di antara beberapa rute jaringan.
3. Penyeimbangan Beban Berbasis Aplikasi: Mendistribusikan permintaan berdasarkan jenis aplikasi atau protokol tertentu.
4. *Load Balancing* Berbasis DNS: Menggunakan DNS untuk mengarahkan klien ke alamat IP yang benar berdasarkan metrik *load balancing*[8].

2.2 *Zevenet* (*Zen Load Balancer*)

Zevenet adalah sebuah *load balancer* (pengimbang beban) dan aplikasi pengamanan jaringan yang digunakan untuk mengelola dan mendistribusikan lalu lintas jaringan dengan cerdas, sistem pengiriman dengan berbagai fitur yang mendukung beragam jenis service, Seperti HTTP *(Hypertext Transfer Protokol)* atau *HTTPS (HTTP Secure),* untuk aplikasi web serta layanan penyeimbangan beban TCP dan UDP[9].

Zevenet dirancang untuk meningkatkan kinerja, ketersediaan, dan keamanan aplikasi dan layanan yang di-host di berbagai *server* backend. Beberapa fitur dan fungsi utama *Zevenet* meliputi[1][10]:

1. *Load Balancing: Zevenet* dapat mendistribusikan lalu lintas jaringan ke berbagai *server backend*, memastikan beban yang merata di antara mereka. Ini membantu dalam meningkatkan kinerja dan skalabilitas aplikasi.
2. Pemantauan dan Pelaporan: *Zevenet* dilengkapi dengan alat pemantauan dan pelaporan yang memungkinkan administrator untuk melacak kinerja, lalu lintas, dan kesehatan *server-server backend* secara *real-time*.
3. Pemutusan SSL (SSL Offloading): *Zevenet* dapat menangani dekripsi SSL/TLS sehingga mengurangi beban *server backend* dan mempercepat pengiriman konten yang dienkripsi.
4. Konfigurasi melalui Antarmuka Web*: Zevenet* menyediakan antarmuka pengguna berbasis *web* yang mudah digunakan untuk mengelola konfigurasi *load balancing* dan pemantauan.
5. Keamanan: Selain *load balancing, Zevenet* juga memiliki fitur-fitur keamanan seperti firewall, IDS (*Intrusion Detection System*), dan IPS (*Intrusion Prevention System*) untuk melindungi jaringan kita dari ancaman keamanan.

2.2.1 Load Balancing pada *Zevenet*

Zevenet adalah salah satu perangkat lunak *open-source* yang digunakan untuk *load balancing*. *Zevenet* memiliki berbagai fitur yang berguna dalam mengelola *traffic* jaringan, termasuk algoritma *load balancing* yang dapat disesuaikan, pemantauan kesehatan *server* dan banyak lagi[11].

2.3 *Server* Protokol

Server adalah sistem komputer yang memproses permintaan dan mengirim data ke komputer. *Server* Protokol merupakan sebuah sistem yang mengatur proses pertukaran data antar komputer dan jaringan. *Protocol* juga bisa disebut sebagai media yang menghubungkan perangkat, yang akan saling komunikasi dan bertukar informasi. Ini berarti *server* tersebut menerima permintaan dari klien dan memberikan respon sesuai dengan protokol yang diikuti. Server Web (HTTP/HTTPS). Server web seperti Apache, Nginx, atau Microsoft IIS menjalankan Protokol HTTP atau HTTPS untuk mengirimkan halaman web, gambar, atau berkas-berkas lainnya kepada klien.

2.3.1. Apache Web *Server*

Apache web server adalah *open source* yang dibangun dan dikelola oleh *apache.org*. *Apache web server* merupakan *unix-based web server* yang awalnya dikembangkan berbasis kode pada NCSA HTTPD 1.3 kemudian diprogram ulang menjadi sebuah *web server* yang paling banyak digunakan saat ini [12]. *Apache* HTTP dikembangan oleh *apache oftware foundation* sebagai usaha untuk membangun *server* HTTP *open source* yang dapat digunakan untuk sistem operasi modern seperti UNIX dan *windows*. Tujuannnya adalah untuk memberikan keamanan, efisiensi, dan ekstensibilitas bagi pengembang aplikasi dan kompatibel dengan standar baku HTTP. Proyek ini pertama kali diluncurkan pada tahun 1995 kemudian menjadi yang terpopuler sejak april 1996. *Apache* HTTP fleksibel terhadap berbagai sistem operasi seperti *windows* 9x/NT/2000/XP/Vista ataupun Unix dan Linux [13].

Apache terdiri dari dua blok bangunan utama dengan bangunan akhir yang terdiri dari banyak blok bangunan kecil lainnya. Blok bangunan tersebut ialah *apache core* dan kemudian *modul apache* yang dalam arti memperluas inti *apache* [14].

Apache adalah sebuah nama *web server* yang bertanggung jawab pada *request-response* HTTP dan *logging* informasi secara detail. Selain itu, *apache* juga diartikan sebagai suatu *web server* yang kompak, modular, mengikuti standar protokol HTTP, dan tentu saja sangat digemari lebih dari 42% dari berbagai domain *website* yang ada di internet [15].

2.4 Docker

Docker adalah platform perangkat lunak yang digunakan untuk mengemas, mengirim, dan menjalankan aplikasi dalam wadah (*container*). Wadah adalah unit standar yang dapat membungkus semua elemen yang diperlukan untuk menjalankan sebuah perangkat lunak, termasuk kode, runtime, perpustakaan, dan variabel lingkungan[16]. *Docker* memungkinkan aplikasi untuk diisolasi dalam wadah, sehingga dapat dijalankan dengan konsisten di berbagai lingkungan, seperti lingkungan pengembangan, pengujian, dan produksi.

Docker adalah salah satu solusi populer untuk kontainerisasi. Beberapa fitur utama *doker* yang harus dipahami meliputi:

Tabel 2.1 Fitur Utama Docker

FITUR UTAMA *DOCKER*	
1. Image	*Docker image* adalah tample bawaan

	yang berisi sistem operasi, aplikasi, dan gambar. Gambar yang digunakan sebagai dasar untuk membuat wadah.
2. *Container*	*Container Docker* adalah yang dihasilkan dari image *docker*. *Container* berjalan dalam isolasi dan memiliki lingkungan runtime yang independen dari lingkungan host.
3. Portabilitas	*Container Docker* dapat dijalankan di berbagai lingkungan yang mendukung *Docker*, dari lingkungan pengembangan hingga produksi, tanpa perlu mengubah kode aplikasi.
4. Skalabilitas	*Docker* memungkinkan pengguna untuk dengan mudah menambah atau mengurangi jumlah *Container* sesuai dengan kebutuhan aplikasi, sehingga mendukung skalabilitas *horizontal.*

2.5 Container

Container adalah teknologi *software* yang mencakup aplikasi beserta dependensiasinya, seperti sistem operasi, konfigurasi dan *library,* didalam unit ini aplikasi berjalan secara independen dan terisolasi dari lingkungan host, aplikasi dan komponen lainnya dikelompokkan menjadi satu *server* dalam *Container,*

layanan dikelola secara konsistensi untuk meningkatkan efisiensi, tanpa kendala dengan aplikasi lain yang berjalan di sistem yang sama[3][16].

Dengan menggunakan *container*, aplikasi dapat dijalankan secara konsisten di berbagai lingkungan, dari pengembangan hingga produksi, tanpa perlu khawatir tentang perbedaan dalam konfigurasi sistem host. Berikut adalah beberapa konsep utama yang terkait dengan *container*:

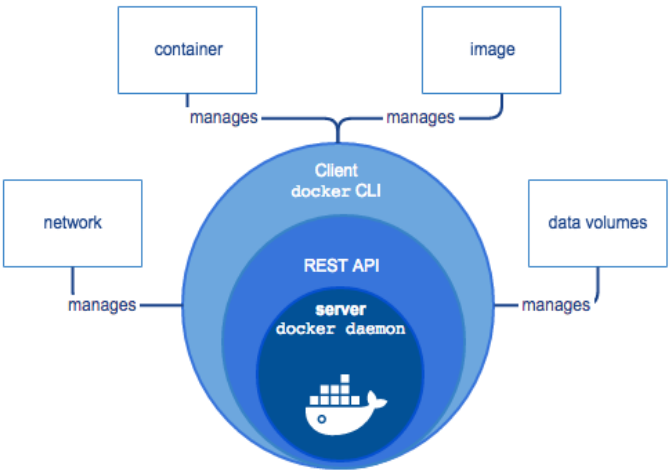
1. Isolasi: *Container* menggunakan teknologi isolasi seperti namespace dan groups pada sistem operasi Linux untuk memastikan bahwa aplikasi dalam *container* berjalan terpisah dari aplikasi lain dan tidak mempengaruhi atau berinteraksi dengan sistem host atau *container* lainnya.
2. Portabilitas: *Container* adalah unit yang dapat dipindahkan dengan mudah antara berbagai lingkungan. Ini berarti Anda dapat mengemas aplikasi Anda dalam wadah di lingkungan pengembangan dan dengan mudah menjalankannya di lingkungan produksi tanpa perubahan signifikan.
3. Kekekalan: Wadah bersifat immutabel, artinya setelah wadah dibuat, isinya tidak berubah. Jika perlu mengubah aplikasi atau konfigurasi, Anda hanya perlu membuat versi *container* baru.
4. *Docker*: adalah *platform* yang paling populer untuk mengelola dan menjalankan *container*. Ini menyediakan alat-alat untuk membuat, mengirim, dan mengelola *container* dengan mudah. *Docker* juga memiliki *docker* Hub, yang merupakan repositori publik berisi banyak image container yang dapat digunakan sebagai dasar untuk aplikasi Anda.
5. *Microservices*: *Container* sering digunakan dalam arsitektur mikroservis, di mana setiap komponen aplikasi dijalankan dalam *container* terpisah. Ini

memungkinkan skalabilitas, perbaikan isolasi, dan pemeliharaan independen dari berbagai bagian aplikasi.

Tabel 2.2 Keuntungan Container dengan Docker

KEUNTUNGAN *CONTAINER* DENGAN *DOCKER*	
1. Isolasi	Wadah diisolasi dari lingkungan host dan wadah lain, mengurangi konflik dan masalah ketergantungan.
2. Portabilitas	*Container* dapat dijalankan di berbagai lingkungan tanpa perubahan apa pun, sehingga mengurangi masalah konsistensi lingkungan.
3. Efisiensi	*Container* memiliki *overhead* yang lebih rendah dibandingkan dengan mesin *virtual* tradisional, memungkinkan penggunaan sumber daya yang lebih efisien.

Docker menggunakan arsitektur *client-server*. Dimana c*lient* dan *docker* berkomunikasi dengan daemon *docker*, yang melakukan suatu tindakan untuk membangun, menjalankan, dan mendistribusikan *container docker*. *client docker* dan *daemon* dapat berjalan pada sistem yang sama. C*lient docker* dan *daemon* berkomunikasi menggunakan REST API, melalui soket UNIX atau antarmuka jaringan[17].



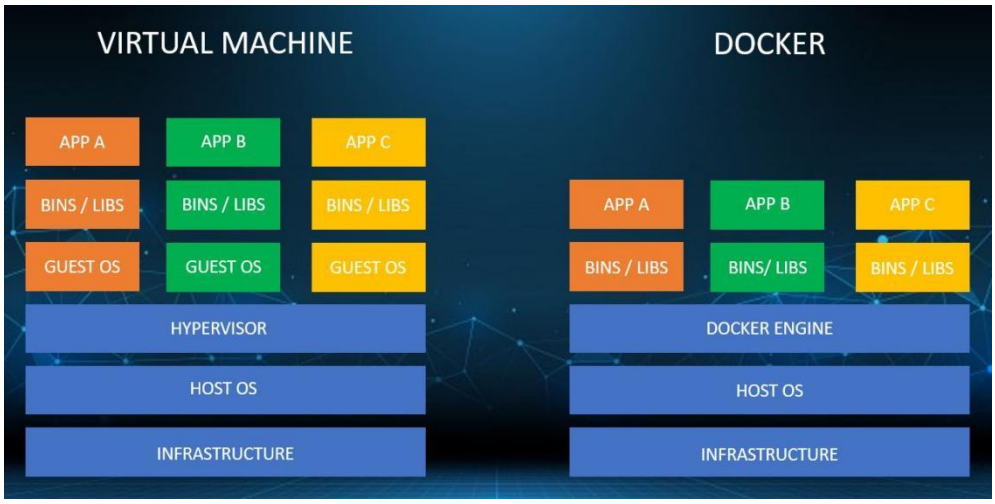
Gambar 2.1 Server Docker daemon

Docker sendiri memiliki banyak komponen dalam menjalankan proses dan tugasnya. Masing-masing komponen pada *docker* itu saling ketergantungan, sehingga jika ada komponen yang tidak tersedia hal itu tentu saja akan *Docker* images adalah sebuah komponen *docker* yang berupa template dasar yang bersifat *read only*. Tamplate ini sebenarnya adalah sebuah OS atau OS yang telah di-install berbagai aplikasi.

Docker images berfungsi untuk membuat *docker Container*, dengan hanya satu *docker* images dapat membuat banyak *docker Container*.

Docker Container merupakan sebuah *images* yang dapat dikemas dan bersifat *read-write*, *Container* berjalan diatas images. *Docker Container* juga bisa dikatakan sebagai sebuah folder, dimana *Container* ini dibuat dengan menggunakan *docker Container*. Setiap *docker Container* disimpan maka akan terbentuuk layer baru tepat di atas *Docker* images atau base images diatasnya.

Perbedaan *docker* dengan *virtual machine* adalah *Container* dalam *docker* memiliki sumber daya yang terisolasi sama dan manfaat alokasi seperti *virtual machine* tetapi perbedaan arsitektur yang berbeda memungkinkan *Container* untuk menjadi lebih portable dan efisien .*Virtual machine* membutuhkan banyak ruang untuk *guest OS* atau *Operating System*. Sedangkan *docker* tidak membutuhkan banyak ruang untuk *operating* sistem. Karena *operating* sistem pada *docker* dapat digunakan untuk menjalankan beberapa *container* bersama-sama[6].



(a) (b)

Gambar 2.2 Perbedaan antara *Docker* dan *Virtual machine:* (a) Virtual machine dan (b) Docker Container

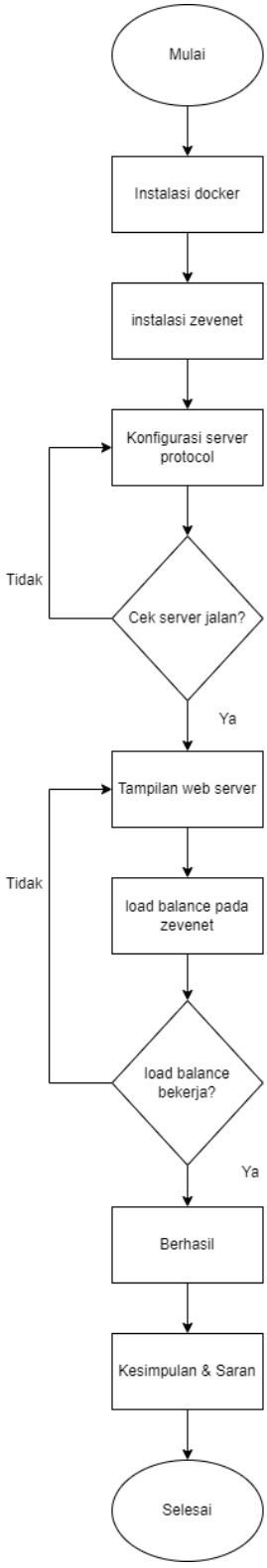
BAB III

METODELOGI PENELITIAN

3.1 Kerangka Kerja Penelitian

Kerangka kerja penelitian merupakan tahapan perancangan penelitian yang akan dilakukan. Dalam penelitian yang sedang dilakukan, kerangka kerja penelitian dibuat menggunakan *flowchart* sebagai alur pengerjaan. *Flowchart* ini memudahkan peneliti ketika memulai penelitian.

Pada tahap pertama penelitian dilakukan instalasi *docker* sebagai *platform* perangkat lunak untuk mengembangkan, menguji, dan men*deploy* aplikasi menggunakan teknologi kontainer. Kemudian dilakukannya instalasi *zevenet* sebagai *load balancer*. Setelah semua terinstalasi maka konfigurasi *server* protokol dilakukan apabila *server* berjalan maka *web server* akan tertampil. Akan tetapi apabila *server* tidak berjalan maka konfigurasi akan kembali dilakukan. *Load balance* yang dilakukan pada *zevenet* apabila selesai dapat ditarik kesimpulan dan saran selama penelitian berlangsung.



Gambar 3.1 Flowchat penelitian

3.2 Perancangan Penelitian

Kegiatan penelitian projek akhir ini dilakukan di Laboratorium Jaringan Komputer dan Komunikasi Data (JARKOM) Fakultas Ilmu Komputer Universitas Sriwijaya. Sistem yang dibangun terdiri dari beberapa tahapan yang meliputi perancangan topologi, identifikasi kebutuhan perangkat keras, identifikasi kebutuhan perangkat lunak, instalasi dan konfigurasi sistem, skenario pengujian, dan skenario pengambilan data.

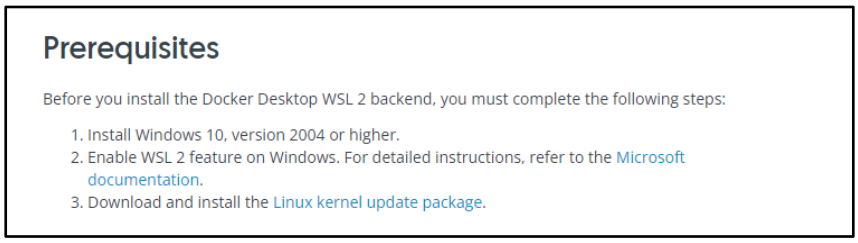
3.3 Persiapan Instalasi

Persiapan instalasi adalah menjelaskan langkah-langkah instalasi sebelum melakukan implementasi termasuk konfigurasi *load balancing server* dan instalasi *docker*. Sebelum melakukan implementasi *load balancing server* menggunakan *docker*, kita perlu melakukan persiapan lingkungan yang mencakup *load balancing server* menggunakan *Zevenet* yang telah dikonfigurasi sesuai dengan perintah resmi *website zevenet*.

Sebelum pengujian dilakukan langkah yang diambil ialah melakukan instalasi *docker desktop* pada windows dengan membuka *browser* lalu *install* aplikasi *docker windows desktop.*

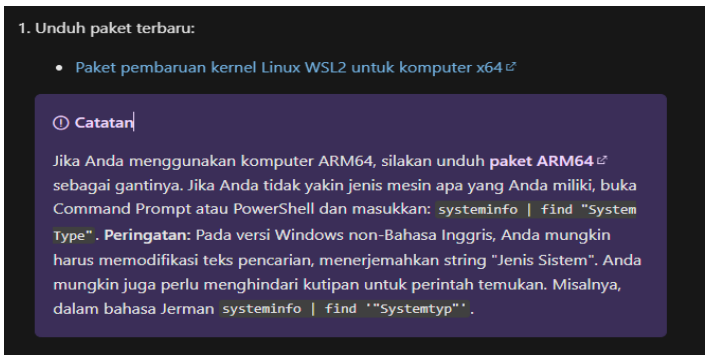
Gambar 3.2 Instalasi *Docker* Desktop

Langkah berikutnya adalah melakukan instalasi *software* WSL 2 *backend* untuk menjalankan aplikasi *docker*. Untuk melakukan instalasi dibutuhkan beberapa persyaratan yang harus dimiliki di dalam *server* atau komputer yang digunakan saat akan melakukan pengujian. Gambar 3.3 berikut adalah persyaratan yang harus dimiliki.



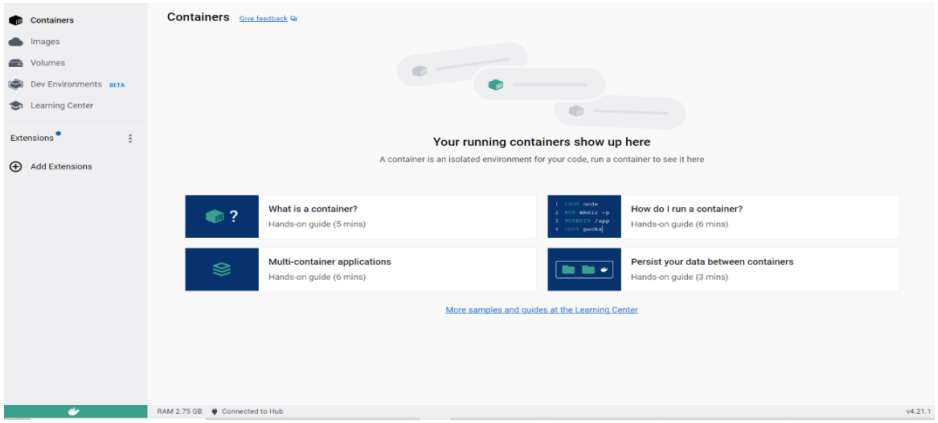
Gambar 3.3 Syarat *download file* WSL 2 *Backend*

Setelah syarat pada gambar 3.3 terpenuhi maka WSL 2 *backend* dapat di*download* pada laptop untuk digunakan dalam pengujian dengan mengklik *Linux Karnel Update Package*. Kemudian akan muncul perintah seperti gambar 3.4 dibawah ini. Lalu klik pada bagian "Paket pembaruan *kernel Linux* WSL2 untuk komputer x64".



Gambar 3.4 Download file WSL 2 Backend

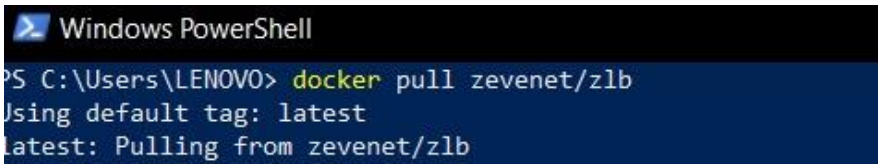
File WSL2 yang telah di*download* kemudian di klik dan di *install* agar aplikasi *docker* dapat dijalankan dan tidak terjadi *eror* saat melakukan pengujian. Setelah semua *software* telah terinstal maka aplikasi *docker* akan terbuka dengan tampilan *desktop* seperti gambar 3.5 dibawah ini.



Gambar 3.5 Tampilan Aplikasi *Docker* Desktop

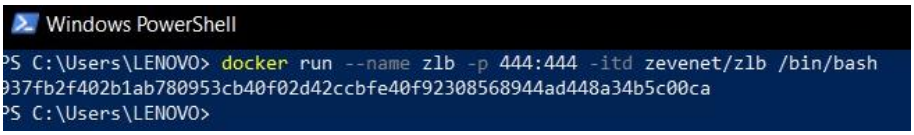
Setelah melakukan instalasi *docker* proses berikutnya adalah melakukan instalasi dan konfigurasi *zevenet*. Memastikan bahwa komputer yang digunakan tidak memiliki *software zevenet* merupakan langkah pertama yang harus dilakukan sebelum instalasi.

Proses instalasi *zevenet* dilakukan dengan melakukan konfigurasi pada *Windows PowerShell*. Konfigurasi tersebut terdiri dari *download image* ZLB dan menjalankan serta memastikan bahwa *zevenet* telah berjalan. *Command download image* ZLB dapat dilihat pada gambar 3.6 dibawah ini.



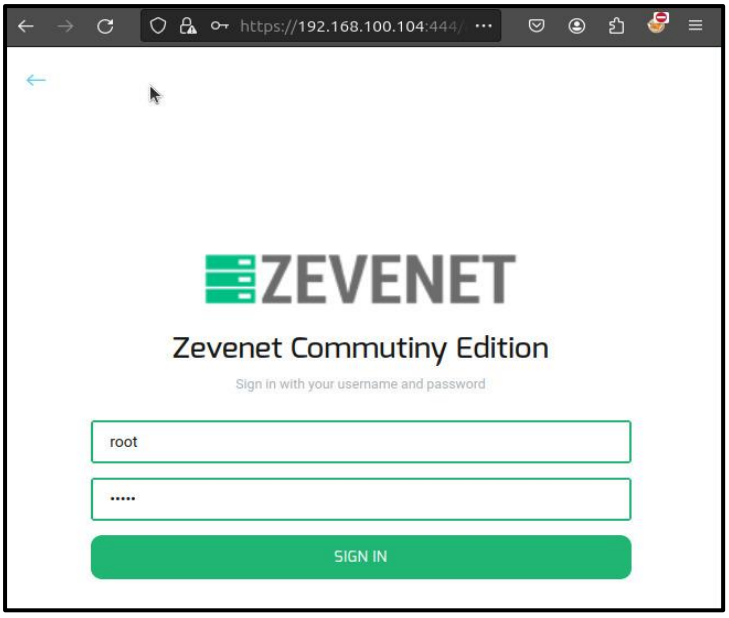
Gambar 3.6 Command download image ZLB

Setelah *image* ZLB ter*download* maka *container* baru dapat dimulai dengan mengekspor *port* GUI untuk menjalankan *container zevenet* melaui *Windows PowerShell*.



Gambar 3.7 Command membuat container zevenet

Apabila proses yang telah dijelaskan telah dilakukan dan selesai maka akses *zevenet* dapat dilakukan. *Zevenet* diakses menggunakan alamat *web* https://[host_public_ip atau domain]:444 dimana *host_public_ip* atau domain diisi dengan *ip public*. Kemudian tampilan halaman *log in zevenet* akan tertampil seperti gambar 3.8 dibawah ini.



Gambar 3.8 Tampilan halaman *log in zevenet*

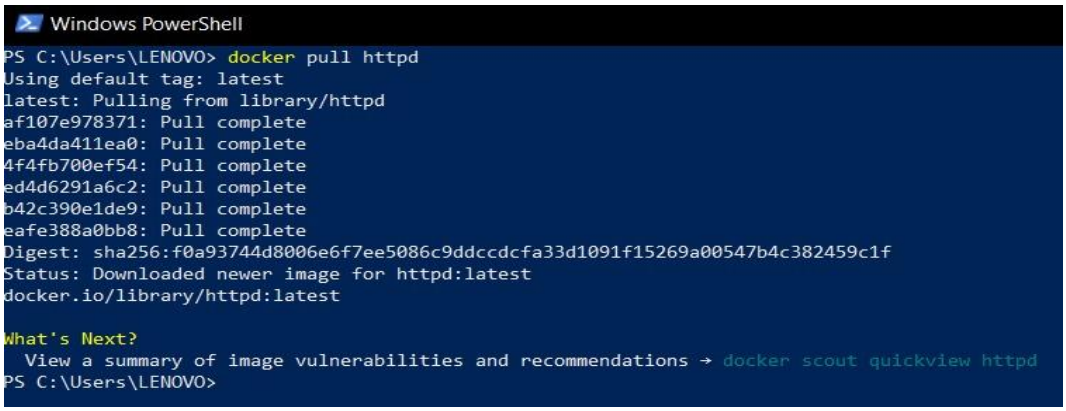
Pada halaman *log in* dimasukkan nama pengguna dan kata sandi yang telah dibuat yaitu *root* dan admin untuk sandi.

3.4 *Web Server* Protokol

Implementasi *load balancing* untuk *server* protokol menggunakan *docker* adalah pendekatan yang efektif untuk mendistribusikan *traffic* antara beberapa *instance server* yang berjalan di *container*.

Persiapan *Container Server*, pertama perlu membuat *docker* image kustom untuk *server* protokol dengan konfigurasi yang sesuai dengan kebutuhan. Buat *dockerfile* dengan intruksi yang diperlukan, seperti menyalin konfigurasi *server* kustom dan file situs web ke dalam image.

Bangun image menggunakan perintah *docker* untuk membangun image *server* kustom dari *dockerfile.* setelah itu buat *container server*, setelah image *server* kustom dibangun buat beberapa *container server* berdasarkan image tersebut dengan perintah *docker run.*



Gambar 3.9 Command Web Server docker

Perintah pada gambar 3.9 diatas merupakan perintah yang digunakan untuk mengunduh *image* resmi *Apache* HTTP *server* dari *docker hub* dan menyimpannya secara lokal. *Docker hub* sendiri digunakan sebagai repositori *image docker* dan *image* httpd menyediakan lingkungan *apache* HTTP *server* yang siap digunakan dalam kontainer *docker*.

Setelah dilakukannya *command* pada gambar 3.9 kemudian pembuatan *docker* kontainer dibuat. Pembuatan kontainer dilakukan dengan memasukkan *command run* seperti gam 3.10 dibawah ini.

Gambar 3.10 Membuat *Container Docker*

Pada Gambar 3.10 perintah *docker* run untuk membuat dan memulai *container Docker,* opsi ‘-d’ akan menjalankan *Container* dalam mode latar belakang, ‘-p 80:80’ akan meneruskan port 80 dari host ke port 80 dalam *Container*, sehingga dapat mengakses *server* protokol dari *browser* di port 80.

3.5 Hasil dan Pembahasan

Hasil dan Pembahasan diperoleh dari tahapan pengujian yang telah dilakukan pada *server* Protokol dengan menerapkan metode *Docker*. Data yang diambil pada Skenario pengujian sebelumnya akan dijelaskan pada bab selanjutnya, dengan menjelaskan proses pengujian pertama, kedua, sehingga akan mendapatkan kesimpulan dan saran.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Implementasi *Zen Load Balance Server* Protokol Pada *Docker*

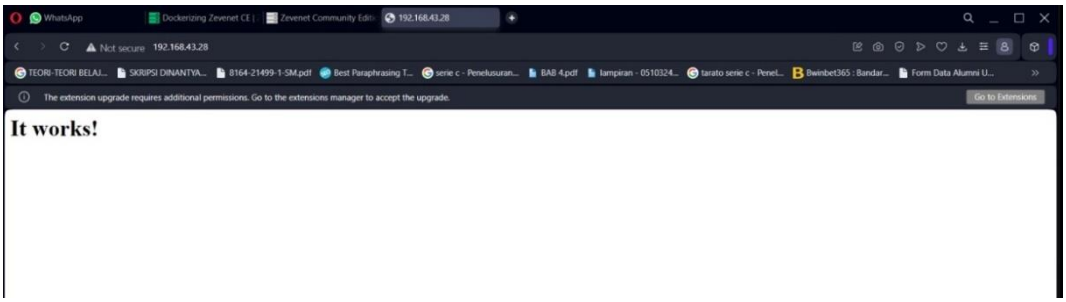
Implementasi *server* ke dalam lingkungan *docker* melibatkan pembuatan citra *docker* untuk *server* dan membangun *container* bedasarkan citra tersebut. Pembuatan tersebut berdasarkan beberapa langkah pembuatan yaitu membuat *dockerfile* untuk *image server* dan verifikasi container *server* kemudian grafik *load balancing* dapat dilihat pada salah satu fitur yang ada pada *docker* atau pun *zevenet.*

Dockerfile adalah file teks yang berisi instruksi tentang bagaimana *Docker* harus membangun image *Container*. Membuat *Docker*file untuk *server* yang akan mendefinisikan lingkungan dan konfigurasi yang diperlukan untuk menjalankan *server* protocol.

Setelah *Container server* berjalan, dapat memverifikasi apakah web *server* protokol dapat diakses melalui *browser* dengan membuka alamat IP atau nama domain *server* pada port 80. Misalnya, jika *server* memiliki alamat IP *127.0.0.0/80,* dapat membuka *browser* dan mengakses *http://127.0.0.0/80* untuk melihat halaman *server*. Dengan demikian, *server* telah diintegrasikan dalam lingkungan *docker* dan berjalan sebagai *Container* terisolasi.

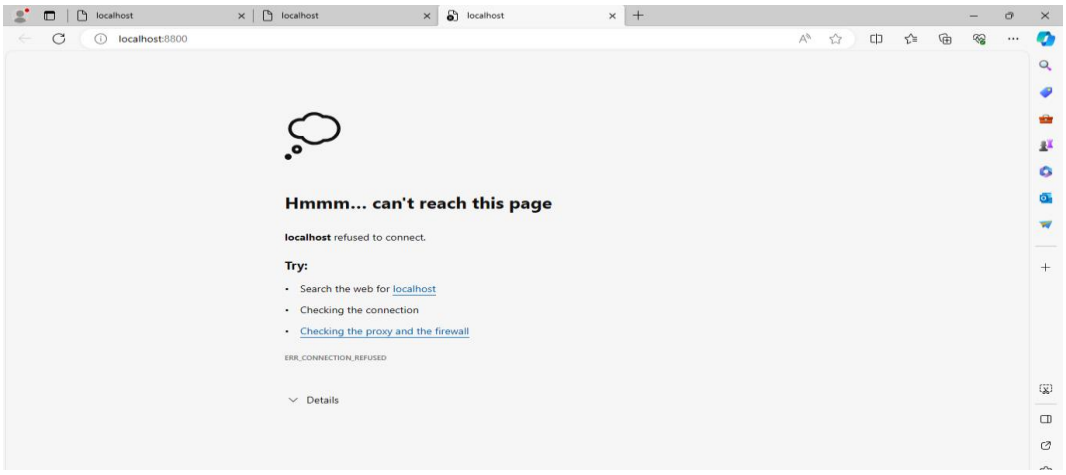
Hal selanjutnya yang dilakukan adalah membuka membuka *browser* menggunakan *IP Network* untuk masuk *ke web server Apache* baru dengan port host yang sama dengan *container http://localhost:*80.

Gambar 4.1 dibawah ini tampilan dari *container server* yang berjalan didalam *container docker*. Apabila *server* dihentikan maka tidak bisa diakses melalui *web browser* seperti gambar 4.1 berikut.



Gambar 4.1 Tampilan *Web Server*

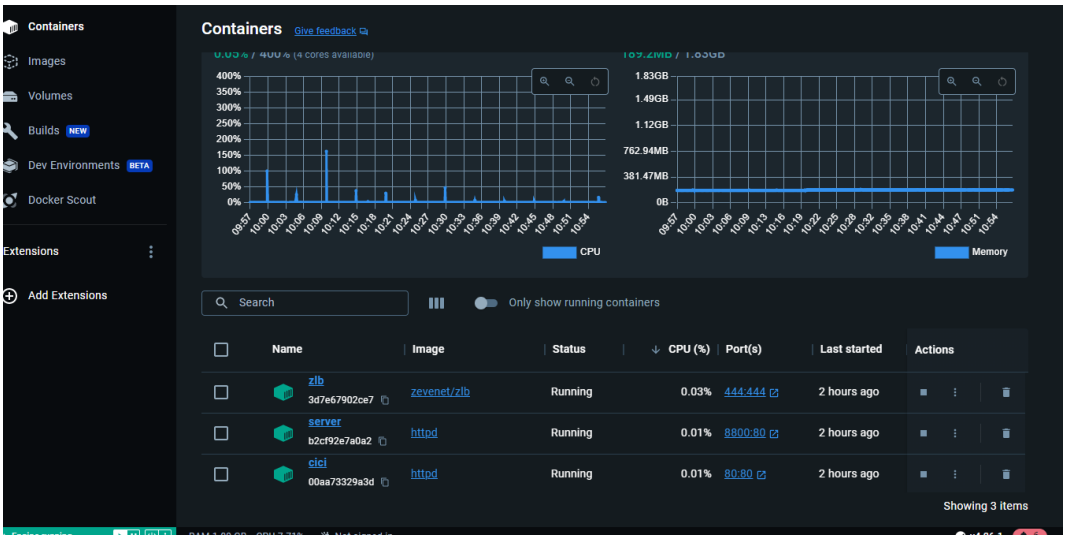
Tampilan gambar 4.1 adalah tampilan yang muncul ketika *web server* dibuka adalah *It works!* yang menandakan bahwa *server* telah berjalan dengan baik pada mesin pencari seperti *chrome*, *bing*, *mozilla firefox*, dan sejenisnya sedangkan tampilan pada gambar 4.2 ketika *server* tidak berjalan adalah *hmmm… can't reach this page*.



Gambar 4.2 *Server* tidak berjalan

Pada *container docker*, *server* yang berjalan dan tidak berjalan tertampil dengan keterangan detail mengenai *port*, status, nama *server*, penggunaan data

CPU, dan *action* untuk memberhentikan status dari *container server* yang sedang berjalan. Tampilan *container docker* tersebut dapat dilihat pada gambar 4.3 berikut.



Gambar 4.3 Container *server* yang berjalan

Gambar 4.3 memperlihatkan isi kontainer yang sedang berjalan dimana apabila salah satu *server* dimatikan maka *web* tidak dapat dibuka atau dijalankan seperti gambar 4.2 sebelumnya.

Gambar 4.4 Grafik *zevenet*

Gambar 4.4 merupakan tampilan grafik *load balancing* pada *zevenet*. Kedua grafik tersebut merupakan hasil *daily* dan *weekly load balancing* pengujian. Setiap warna menunjukkan perbedaan menit yang terambil oleh *zevenet* yaitu warna abu-abu untuk 15 menit terakhir, warna hijau muda untuk 5 menit terakhir dan hijau muda untuk menit terakhir. Grafik pada gambar 4.4 merupakan grafik penggunaan *server* sebelum dilakukannya *load balancing* dimana hasil grafik menunjukkan penggunaan data yang tidak rata sehingga dapat terjadi *traffic* jaringan selama *server* berjalan. Grafik yang telah dilakukan *load balancing* menunjukkan data yang dipakai sama rata seperti gambar 4.5 berikut ini.

Gambar 4.5 Grafik *load balancing*

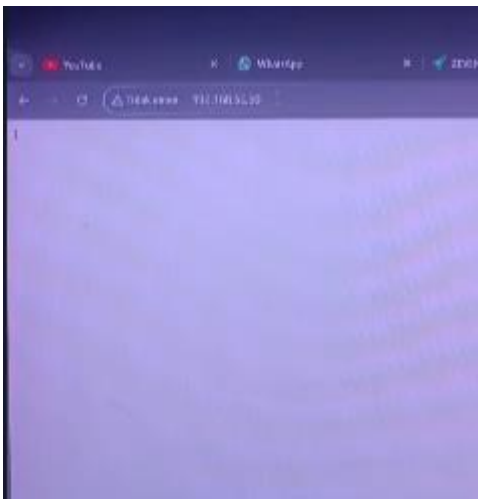
Berdasarkan grafik pada Gambar 4.5 diatas warna hijau menunjukkan beban *server,* bahwa *server* telah ter*load balance,* dimana tidak adanya beban berlebih pada *server* ketika dijalankan karena penggunaan dari data telah tersalurkan sama rata pada setiap *server* telah terbagi bebannya.

4.2 Hasil performasi

Pada pengujian performarsi *load balancer zevenet* menggunakan algoritma *fastest* dimana algoritma ini melakukan pembagian beban dengan mengutamakan *server - server* yang memiliki respon yang paling cepat. *Server* di dalam jaringan yang memiliki respon paling cepat merupakan *server* yang akan mengambil beban pada saat permintaan masuk.

Pengujian dilakukan dengan mengganti *weight* pada *backend farm* kedua server untuk menguji kesempatan waktu masing-masing *server* terbuka. Pengujian tersebut menggunakan *weight* 1 banding 3 dan 1 banding 2 dimana 1 merupakan *weight server* 1 sedangkan 3 dan 2 merupakan *weight server* 2.

Berdasarkan *weight* tersebut server 1 tidak membutuhkan kesempatan membuka server lebih cepat atau didahulukan sedangkan server 2 membutuhkan waktu lebih lama berkisar 18 detik untuk membuka seperti gambar dibawah ini.



Gambar 4.6 *Server* 2 me*load*

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari pengujian dan hasil implementasi web server *Zevenet* menggunakan metode *Docker* untuk mengatasi lalu lintas web. Bedasarkan penelitian dan pengujian yang dilakukan, diperoleh beberapa kesimpulan sebagai berikut:

1. Implementasi web server menggunakan *Zevenet* dengan metode *Docker* dapat memberikan manfaat dalam mengatasi lalu lintas web.
2. Mengidentifikasi dan mengimplementasikan konfigurasi *Zevenet* dan *Docker* dengan memastikan penggunaan sumber daya yang efisien dan penanganan *Traffic* yang stabil.

5.2 Saran

Bedasarkan kesimpulan diatas, berikut beberapa saran yang dapat diberikan untuk pengembangan lebih lanjut atau penelitian terkait:

1. Menyelidiki dan membandingkan performa *Zevenet* dalam *Docker* dengan solusi *Container*isasi lainnya, seperti Kubernetes atau *Docker* Swarm. Mengidentifikasi kelebihan dan kekurangan masing-masing platform dapat membantu dalam memilih solusi yang tepat.
2. Meneliti dan mengoptimalkan konfigurasi *Zevenet* dalam *Docker* untuk meningkatkan performa dan efisiensi lebih lanjut. Melakukan pengaturan yang tepat untuk sumber daya seperti CPU, Memori, dan Bandwidth jaringan dapat membantu dalam memaksimalkan kinerja server.

DAFTAR PUSTAKA

[1] P. A. Ghifary *et al.*, “Analisis Performansi Load Balancer Menggunakan Zevenet dan Haproxy pada Tiga Web Server Load Balancer Performance Analysis Using Zevenet and Haproxy on Three Web Servers.”

[2] A. R. Sofyan, S. Dwi, and Y. Kusuma, “Implementasi Load Balancing Web Server menggunakan Haproxy pada Virtual Server Direktorat SMK Kemendikbudristek,” vol. 6, pp. 9669–9682, 2022.

[3] S. Dwiyatno, E. Rakhmat, and O. Gustiawan, “Implementasi virtualisasi server berbasis docker container,” vol. 7, no. 2, pp. 165–175, 2020.

[4] A. Info, “Instalasi Serta Konfigurasi HAproxy Sebagai Load Balancing Web Server,” pp. 23–35, 2023.

[5] B. A. B. Iii and L. Teori, “No Title,” pp. 13–34.

[6] M. Rexa, M. Bella, M. Data, and W. Yahya, “Implementasi Load Balancing Server Web Berbasis Docker Swarm Berdasarkan Penggunaan Sumber Daya Memory Host,” vol. 3, no. 4, pp. 3478–3487, 2019.

[7] S. D. Riskiono and D. Darwis, “Peran Load Balancing Dalam Meningkatkan Kinerja Web Server di Lingkungan Cloud,” vol. 8, no. 2, pp. 1–8, 2020.

[8] D. K. Hakim, J. K. Riyanto, and A. Fauzan, “Pengujian Algoritma Load Balancing pada Virtualisasi Server (Testing the Load Balancing Algorithm on Server Virtualization),” vol. 16, no. 1, pp. 33–41, 2019.

[9] I. K. Dewi, “Implementasi Load Balancer menggunakan Zevenet dengan Database Distributed Redis serta Perbandingan Performa MongoDB dan ArangoDB berdasarkan Respons Time,” no. January, 2021.

[10] T. Akhir, “ANALISIS PERFORMANSI LOAD BALANCER TIGA WEB SERVER LOAD BALANCER PERFORMANCE ANALYSIS USING ZEVENET AND HAPROXY ON THREE WEB SERVERS TIGA WEB SERVER LOAD BALANCER PERFORMANCE ANALYSIS USING ZEVENET AND HAPROXY ON THREE WEB SERVERS,” 2022.

[11] L. B. Seiring and S. Baik, “BAB I,” pp. 7–9.

[12] A. Y. Chandra, “Analisis Performansi Antara Apache & Nginx Web Server Dalam Menangani Client Request,” *J. Sist. dan Inform.*, vol. 14, no. 1, pp. 48–56, 2019, doi: 10.30864/jsi.v14i1.248.

[13] A. Jiwandono, “Analisa Perbandingan Kinerja Web Server Apache, Nginx, Dan Litespeed Dengan Menggunakan Metode Stress Test,” pp. 1–78, 2021.

[14] I. F. Irza, Z. Zulhendra, and E. Efrizon, “Analisis Perbandingan Kinerja Web Server Apache dan Nginx Menggunakan Httperf Pada Portal Berita (Studi Kasus beritalinux.com),” *Voteteknika (Vocational Tek. Elektron. dan Inform.*, vol. 5, no. 2, 2017, doi: 10.24036/voteteknika.v5i2.8489.

[15] I. K. S. Satwika and K. N. Semadi, “Perbandingan Performansi Web Server Apache Dan Nginx Dengan Menggunakan Ipv6,” *SCAN - J. Teknol. Inf. dan Komun.*, vol. 15, no. 1, pp. 10–15, 2020, doi: 10.33005/scan.v15i1.1847.

[16] B. A. B. Ii, D. Teori, and D. A. N. Tinjauan, “No Title,” pp. 3–5.

[17] Y. Cahyaningrum and I. R. Widiasari, “Analisis Performa Container Berplatform Docker atas Serangan Malicious Software (Malware) Artikel Ilmiah Analisis Performa Container Berplatform Docker atas Serangan Malicious Software (Malware),” no. 672015031, 2019.

LAMPIRAN

Lampiran 1 SK TA

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI

UNIVERSITAS SRIWIJAYA
FAKULTAS ILMU KOMPUTER
Jalan Palembang – Prabumulih Km. 32 Indralaya Ogan Ilir Kode Pos 30662
Telepon (+62711) 379249, Pos-el : humas@ilkom.unsri.ac.id

KEPUTUSAN
DEKAN FAKULTAS ILMU KOMPUTER UNIVERSITAS SRIWIJAYA
Nomor : 0346/UN9.FIK/TU.SK/2023

TENTANG
PENGANGKATAN PEMBIMBING PROJEK
MAHASISWA PROGRAM STUDI TEKNIK KOMPUTER
FAKULTAS ILMU KOMPUTER UNIVERSITAS SRIWIJAYA

DEKAN FAKULTAS ILMU KOMPUTER UNIVERSITAS SRIWIJAYA

MENIMBANG :
a. Bahwa untuk kelancaran pembimbingan dan pembuatan Projek mahasiswa Program Studi Teknik Komputer Fakultas Ilmu Komputer Universitas Sriwijaya perlu ditetapkan dosen Pembimbing Projek;
b. Bahwa sehubungan dengan butir a di atas, dipandang perlu menerbitkan Surat Keputusan sebagai landasan hukumnya.

MENGINGAT :
1. Undang-Undang No. 20 tahun 2003 tentang Sistem Pendidikan Nasional (Lembaran Negara Republik Indonesia tahun 2003 No. 78);
2. Undang-Undang No. 12 Tahun 2012 tentang Pendidikan Tinggi (Tambahan Lembaran Negara Republik Indonesia No. 5336);
3. Peraturan Pemerintah No. 42 tahun 1960 Jo No. 60 tahun 1999 tentang Pendirian Universitas Sriwijaya;
4. Keputusan Menristekdikti No.12 Tahun 2015 tentang Organisasi dan Tata Kerja Universitas Sriwijaya;
5. Keputusan Menristekdikti No.334/M/KP/XI/2015 tentang Pengangkatan Rektor Universitas Sriwijaya;
6. Surat Keputusan Rektor Universitas Sriwijaya No. 0633/UN9/SK/BUK/KP/2023 tentang Pemberhentian Pelaksana Tugas (Plt) Dekan Fakultas Ilmu Komputer Universitas Sriwijaya dan Pengangkatan Dekan Fakultas Ilmu Komputer Universitas Sriwijaya Pengganti Antar Waktu Masa Jabatan Tahun 2020-2024.

MEMUTUSKAN

Menetapkan : KEPUTUSAN DEKAN FAKULTAS ILMU KOMPUTER UNIVERSITAS SRIWIJAYA TENTANG PENGANGKATAN PEMBIMBING PROJEK MAHASISWA PROGRAM STUDI TEKNIK KOMPUTER FAKULTAS ILMU KOMPUTER UNIVERSITAS SRIWIJAYA.

KESATU : Mengangkat Dan Menugaskan Saudara:
1. Ahmad Heryanto, M.T.
2. Adi Hermansyah, M.T.

Sebagai Pembimbing Projek Dari:

Nama	:	Chindy Leanda Putri;
NIM	:	09040581923001;
Program Studi	:	Teknik Komputer;
Peminatan	:	Teknik Komputer Jaringan;
Judul Projek	:	Implemetasi Zen Load Balancer pada Server Protokol berbasis Docker Container.

KEDUA : Semua Biaya Yang Timbul Akibat Adanya Keputusan Ini Dibebankan Kepada Anggaran DIPA Universitas Sriwijaya Nomor : SP DIPA- 023.17.2.677515/2023 Tanggal 30 November 2022.

KETIGA : Keputusan Ini Berlaku Sejak Tanggal Ditetapkan Sampai Dengan Tanggal **7 Februari 2024**. Dengan Ketentuan Bahwa Segala Sesuatu Akan Diubah Dan Atau Diperbaiki Sebagaimana Mestinya Apabila Ternyata Terdapat Kekeliruan.

Ditetapkan Di Indralaya
Pada Tanggal 7 Agustus 2023
DEKAN,

PROF. DR. ERWIN, S.Si., M.Si.
NIP 197101291994121001

Lampiran 2 Kartu Konsultasi Pembimbing I

KARTU KONSULTASI

Nama : Chindy Leanda Putri
NIM : 09040581923001
Program Studi : Teknik Komputer
Jenjang : DIII
Judul Projek : Implementasi Zen Load Balancer pada Server Protokol Berbasis Docker Container
Pembimbing 1 : Ahmad Heryanto, S.Kom., M.T.

No	Tanggal	Hasil Konsultasi/Komentar	Paraf Pembimbing 1
1	05/10/2023	Revisi BAB 1 Pendahuluan 1. Masukan Jurnal penelitian terdahulu sebanyak 3-5 penelitian 2. Gunakan mendelay untuk menyadur kalimat dari penelitian terdahulu	
2	10/10/2023	Sudah Baik, silahkan lanjut kebagian berikutnya	
3	10/10/2023	Revisi bab II TINJAUAN PUSTAKA 1. masukkan sumber setiap paragrafnya	
4	15/10/2023	Tambahkan Gambar	
5	20/10/2023	Perbaiki penulisan Gambar	
6	25/10/2023	Sudah baik, silahkan lanjut ke bagian berikutnya.	
7	02/10/2023	BAB III METODE PENELITIN (ACC) 1. Masukkan Topologi	
8	26/10/2023	Revisi BAB IV HASIL DAN PEMBAHASAN	
9	02/11/2023	Lakukan percobaan	
10	10/11/2023	Revisi Tambahan 1. Setiap gambar dan table harus disebut di dalam kalimat. 2. Setiap gambar dan table yang digunakan buat penjelasannya	
11	20/11/2023	Revisi Load Balancing	
12	02/12/2023	Sudah Baik, silahkan lanjutkan ke bagian berikutnya	
13	10/12/2023	Revisi Bab V KESIMPULAN DAN SARAN 1. Tambahkan kesimpulan berupa informasi codingan. 2. Kesimpulan apakah sudah menjawab tujuan penelitian.	
14	15/12/2023	Sudah Baik, silahkan jadikan draf laporan TA jika bagian sudah direvisi	

Mengetahui,
Koordinator Program Studi
Teknik Komputer,

Huda Ubaya, M.T.
NIP 198106162012121003

Lampiran 3 Kartu Konsultasi Pembimbing II

KARTU KONSULTASI

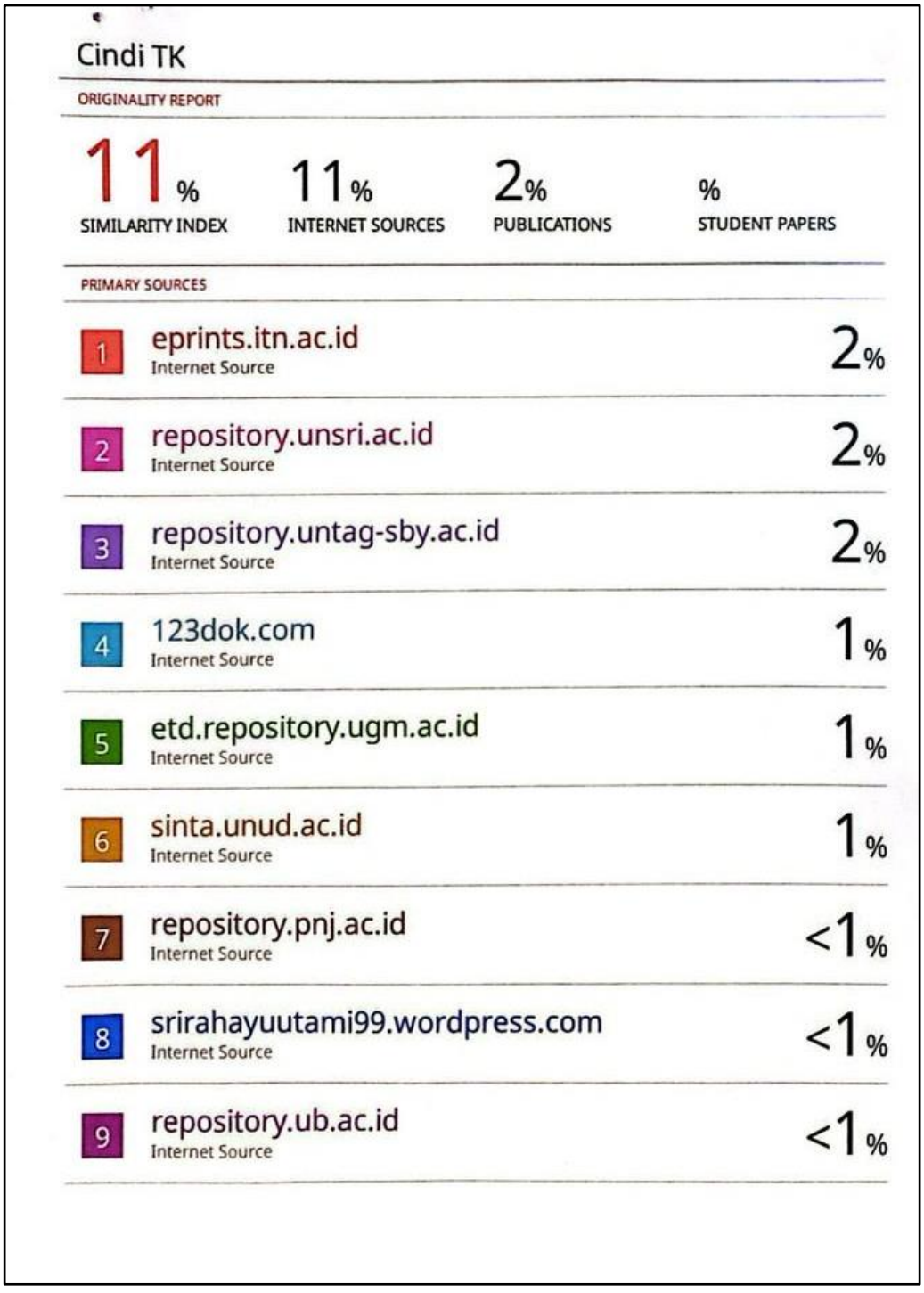
Nama : Chindy Leanda Putri
NIM : 09040581923001
Program Studi : Teknik Komputer
Jenjang : DIII
Judul Projek : Implementasi Zen Load Balancer pada Server Protokol Berbasis Docker Container
Pembimbing 1 : Adi Hermansyah, M.T.

No	Tanggal	Hasil Konsultasi/Komentar	Paraf Pembimbing 2
1	26/08/2023	Revisi BAB 1 Pendahuluan 1. Masukan Jurnal penelitian 2. Tambahkan Rumusan Masalah	
2	02/09/2023	Sudah Baik, silahkan lanjut kebagian berikutnya	
3	03/09/2023	Revisi bab II TINJAUAN PUSTAKA 1. Tambahkan Konsep Docker	
4	10/09/2023	Tambahkan Gambar	
5	20/09/2023	Perbaiki penulisan Gambar	
6	31/09/2023	Sudah baik, silahkan lanjut ke bagian berikutnya.	
7	02/10/2023	BAB III METODE PENELITIN (ACC)	
8	26/10/2023	Revisi BAB IV HASIL DAN PEMBAHASAN	
9	02/11/2023	Lakukan percobaan	
10	10/11/2023	Revisi Tambahan 1. Setiap gambar dan table harus disebut di dalam kalimat. 2. Setiap gambar dan table yang digunakan buat penjelasannya	
11	20/11/2023	Revisi Load Balancing	
12	02/12/2023	Sudah Baik, silahkan lanjutkan ke bagian berikutnya	
13	10/12/2023	Revisi Bab V KESIMPULAN DAN SARAN 1. Tambahkan kesimpulan berupa informasi codingan. 2. Kesimpulan apakah sudah menjawab tujuan penelitian.	
14	15/12/2023	Sudah Baik, silahkan jadikan draf laporan TA jika bagian sudah direvisi	15/1/2024

Mengetahui,
Koordinator Program Studi
Teknik Komputer,

Huda Ubaya, M.T.
NIP 198106162012121003

Lampiran 4 Hasil Pengecekan Turnitin



Cindi TK

ORIGINALITY REPORT

11%	11%	2%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	eprints.itn.ac.id Internet Source	2%
2	repository.unsri.ac.id Internet Source	2%
3	repository.untag-sby.ac.id Internet Source	2%
4	123dok.com Internet Source	1%
5	etd.repository.ugm.ac.id Internet Source	1%
6	sinta.unud.ac.id Internet Source	1%
7	repository.pnj.ac.id Internet Source	<1%
8	srirahayuutami99.wordpress.com Internet Source	<1%
9	repository.ub.ac.id Internet Source	<1%

Lampiran 5 Surat Rekomendasi Ujian Projek Pembimbing I

SURAT REKOMENDASI UJIAN PROJEK

Pembimbing projek memberikan rekomendasi kepada:

Nama	: Chindy Leanda Putri
NIM	09040581923001
Jurusan	: Sistem Komputer
Program Studi	: Teknik Komputer
Peminatan	: Teknik Kompuer Jaringan
Judul Projek	: Implementasi Zen Load Balancer Pada Server Protokol Berbasis Docker Container

Mahasiswa tersebut telah memenuhi persyaratan dan dapat mengikuti ujian projek pada tahun 2023/2024

Palembang, 04 Januari 2024
Pembimbing I,

Ahmad Heryanto, S.Kom., M.T.
NIP 197801222015041002

Lampiran 6 Surat Rekomendasi Ujian Projek Pembimbing II

SURAT REKOMENDASI UJIAN PROJEK

Pembimbing projek memberikan rekomendasi kepada:

Nama	: Chindy Leanda Putri
NIM	09040581923001
Jurusan	: Sistem Komputer
Program Studi	: Teknik Komputer
Peminatan	: Teknik Kompuer Jaringan
Judul Projek	: Implementasi Zen Load Balancer Pada Server Protokol Berbasis Docker Container

Mahasiswa tersebut telah memenuhi persyaratan dan dapat mengikuti ujian projek pada tahun 2023/2024

Palembang, 04 Januari 2024
Pembimbing II,

Adi Hermansyah, M.T
NIK. 1613033004890001

Lampiran 7 Surat Perbaikan Ujian Projek Penguji

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
UNIVERSITAS SRIWIJAYA
FAKULTAS ILMU KOMPUTER
PROGRAM STUDI TEKNIK KOMPUTER
Jalan. Srijaya Negara Kampus Unsri Bukit Besar Palembang Kode Pos 30139
Telepon (+62711) 379249, Pos-el : humas@ilkom.unsri.ac.id

PERBAIKAN UJIAN PROJEK

Nama Mahasiswa	: Chindy Leanda Putri
NIM	: 09040581923001
Program Studi	: Teknik Komputer
Peminatan	: Teknik Komputer Jaringan
Hari / Tanggal	: Kamis / 11 Januari 2024
Waktu	: 10.30 s.d 11.00
Judul Projek	: Implemetasi Zen Load Balancer pada Server Protokol berbasis Docker Container
Pembimbing I	: Ahmad Heryanto, M.T.
Pembimbing II	: Adi Hermansyah, M.T

Perbaikan/Saran :

Revisi laporan dan bentuk ✓

Jangka Waktu Perbaikan :

Telah diperbaiki sesuai dengan saran dan koreksi tim penguji ujian komprehensif.

No.	Nama Penguji	Status Penguji	Tanda Tangan
1.	Kemahyanto Exaudi, M.T.	Penguji	

18/01/24

Palembang, 11 Januari 2024

Koordinator Program Studi
Teknik Komputer,

Huda Ubaya, M.T.
NIP 198106162012121003

Lampiran 8 Surat Perbaikan Ujian Projek Pembimbing I

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
UNIVERSITAS SRIWIJAYA
FAKULTAS ILMU KOMPUTER
PROGRAM STUDI TEKNIK KOMPUTER
Jalan. Srijaya Negara Kampus Unsri Bukit Besar Palembang Kode Pos 30139
Telepon (+62711) 379249, Pos-el : humas@ilkom.unsri.ac.id

PERBAIKAN UJIAN PROJEK

Nama Mahasiswa	: Chindy Leanda Putri
NIM	: 09040581923001
Program Studi	: Teknik Komputer
Peminatan	: Teknik Komputer Jaringan
Hari / Tanggal	: Kamis / 11 Januari 2024
Waktu	: 10.30 s.d 11.00
Judul Projek	: Implemetasi Zen Load Balancer pada Server Protokol berbasis Docker Container
Pembimbing I	: Ahmad Heryanto, M.T.
Pembimbing II	: Adi Hermansyah, M.T

Perbaikan/Saran :

1. Perbaiki Laporan
2. Lengkapi Bab 4

Jangka Waktu Perbaikan :

Telah diperbaiki sesuai dengan saran dan koreksi tim penguji ujian komprehensif.

No.	Nama Penguji	Status Penguji	Tanda Tangan
1.	Ahmad Heryanto, M.T.	Pendamping I (Pembela I)	

Palembang, 11 Januari 2024

Koordinator Program Studi
Teknik Komputer,

Huda Ubaya, M.T.
NIP 198106162012121003

Lampiran 9 Surat Perbaikan Ujian Projek Pembimbing II

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN,
RISET DAN TEKNOLOGI
UNIVERSITAS SRIWIJAYA
FAKULTAS ILMU KOMPUTER
PROGRAM STUDI TEKNIK KOMPUTER
Jalan. Srijaya Negara Kampus Unsri Bukit Besar Palembang Kode Pos 30139
Telepon (+62711) 379249 , Pos-el : humas@ilkom.unsri.ac.id

PERBAIKAN UJIAN PROJEK

Nama Mahasiswa	: Chindy Leanda Putri
NIM	: 09040581923001
Program Studi	: Teknik Komputer
Peminatan	: Teknik Komputer Jaringan
Hari / Tanggal	: Kamis / 11 Januari 2024
Waktu	: 10.30 s.d 11.00
Judul Projek	: Implemetasi Zen Load Balancer pada Server Protokol berbasis Docker Container
Pembimbing I	: Ahmad Heryanto, M.T.
Pembimbing II	: Adi Hermansyah, M.T
Perbaikan/Saran	**:**

Jangka Waktu Perbaikan :

Telah diperbaiki sesuai dengan saran dan koreksi tim penguji ujian komprehensif.

No.	Nama Penguji	Status Penguji	Tanda Tangan
1.	Adi Hermansyah, M.T	Pendamping II (Pembela II)	

Palembang, 11 Januari 2024

Koordinator Program Studi
Teknik Komputer,

Huda Ubaya, M.T.
NIP 198106162012121003