Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 11, No. 3, Juni 2025, hlm. 497-504 e-ISSN: 2528-6579

PENGEMBANGAN PEMBAGIAN BEBAN KERJA SERVER CLUSTER DOCKER SWARM BERBASIS RASPBERRY DENGAN METODE LOADBALANCER IP HASH SERTA TERAUTOMASI ANSIBLE

Fauzan Adzima Tohari[*1], Rakhmadhany Primananda[2], Nur Hazbiy Shaffan[3]

[1,2,3]Universitas Brawijaya, Malang
Email: [1]adzim520@student.ub.ac.id, [2]rakhmadhany@ub.ac.id, [3]nur.hazbiy@ub.ac.id
[*]Penulis Korespondensi

(Naskah masuk: 29 November 2024, diterima untuk diterbitkan: 18 Juni 2025)

Abstrak

Perkembangan teknologi saat ini mempercepat penyebaran informasi, hal ini meningkatkan traffic dan tekanan pada infrastruktur. Menghadapi ini, sistem yang reliable, availability, dan maintainability diperlukan. Dengan container dan Raspberry Pi sebagai server. Memberikan keunggulan selain 3 karakteristik sistem tersebut juga dari sumberdaya yang diperlukan. Raspberry Pi dapat terhubung ke jaringan LAN maupun nirkabel. Fokus penelitian ini pada perancangan dan implementasi sistem berbasis empat Raspberry Pi, tiga sebagai webserver Nginx, dan satu sebagai kontroler. Sistem ini menggunakan teknologi containerization di lingkungan DockerSwarm dan diintegrasikan Ansible untuk otomatisasi tugas. Tujuan penelitian ini adalah menciptakan sistem yang dapat disesuaikan dengan kebutuhan, memanfaatkan Raspberry Pi untuk efisiensi daya, dan memberikan manfaat dalam hal skalabilitas dan otomatisasi. Sistem ini menggunakan empat Raspberry Pi dengan algoritma IP HASH untuk pembagian beban kerja. Komponen utama mencakup input dari pengguna, pemrosesan beban kerja oleh load balancer, dan output melalui antarmuka web browser. Load balancer bertanggung jawab atas distribusi lalu lintas dan pemantauan status server untuk mengalihkan lalu lintas jika diperlukan. Pengujian menunjukkan system dengan load balancer lebih efisien dalam penggunaan CPU dengan perbandingan CPU 39.1% pada ws1 dengan loadbalancer dan 51.2% tanpa loadbalancer, 38.9% pada ws2 dengan loadbalancer dan 51.7% tanpa loadbalancer, serta 6.3% pada ws3 ketika loadbalancer mengalihkan request saat server 1 atau 2 mati dan efektif dalam mengalihkan request saat server bermasalah dengan persentase kegagalan melayani request sebesar 18% atau 18 request dari 1000 request sedangkan tanpa loadbalancer request tidak dapat dilayani ketika server yang dituju mati. Ini membuktikan reliable, availability, dan maintainability pada sistem ini terpenuhi.

Kata kunci: *Container, Raspberry, Loadbalancer, Nginx, IP Hash, Server, Ansible, DockerSwarm*

DEVELOPMENT OF SERVER CLUSTER LOAD BALANCING USING DOCKER SWARM BASED ON RASPBERRY PI WITH IP HASH METHOD AND ANSIBLE AUTOMATION

Abstract

Current technological advancements have accelerated the dissemination of information, increasing traffic and pressure on infrastructure. To handle this, reliable, available, and maintainable systems are required. Utilizing containers and Raspberry Pi as servers offers significant advantages, particularly in resource efficiency. Raspberry Pi can connect to networks via LAN cables or wirelessly. This research focuses on designing and implementing a system based on four Raspberry Pis three as Nginx web servers and one as a controller. The system employs containerization technology within a Docker Swarm environment and integrates Ansible for task automation. The goal is to create an adaptable system that leverages Raspberry Pi for energy efficiency, scalability, and automation benefits. The system uses the IP HASH algorithm for workload distribution, with key components including user input, workload processing by the load balancer, and output via a web browser interface. The load balancer handles traffic distribution and server status monitoring, redirecting traffic as needed. Testing shows that the system with a load balancer is more efficient in CPU usage 39.1% on ws1 with a load balancer and 51.2% without, 38.9% on ws2 with a load balancer and 51.7% without, and 6.3% on ws3 when redirecting requests during server 1 or 2 failures. It effectively redirects requests during server issues, with a failure rate of 18% (18 out of 1000 requests), while requests cannot be served when the targeted server is down without a load balancer. This demonstrates the system's reliability, availability, and maintainability.

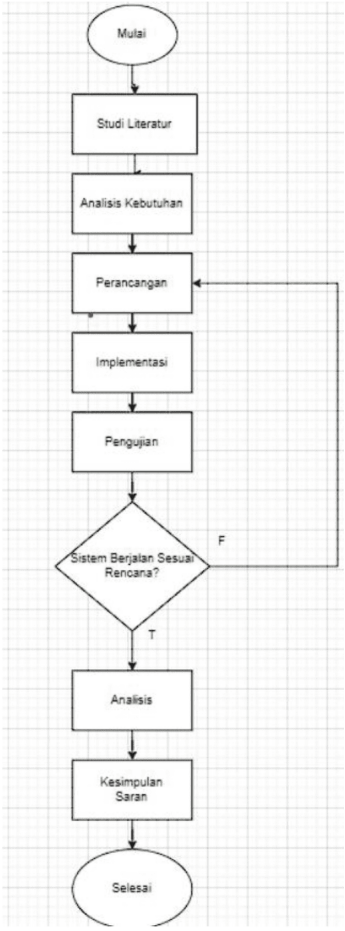
Keywords: *Container, Raspberry, Loadbalancer, Nginx, IP Hash, Server, Ansible, DockerSwarm*

1. PENDAHULUAN

Sebuah sistem yang baik memiliki 3 karakteristik yaitu harus handal, ketersediaan, dan harus maintainability. Handal berarti sistem harus memiliki kemampuan untuk melakukan fungsi yang diperlukan dalam kondisi tertentu untuk interval waktu tertentu, serta dapat memberikan layanan yang benar sesuai dengan fungsi, ketersediaan berarti sistem harus memiliki kemampuan untuk berada dalam keadaan untuk melakukan fungsi yang diperlukan, dalam kondisi tertentu, pada waktu tertentu, atau setelah waktu yang cukup berlalu, selama sumber daya eksternal yang diperlukan tersedia, serta harus dapat dimaintainability berarti sistem harus dapat kemampuan untuk dipertahankan, atau dikembalikan ke keadaan di mana ia dapat menjalankan fungsi yang diperlukan, dalam kondisi penggunaan dan pemeliharaan tertentu (Kishor S., Andrea Bobbio, 2017, p.6). menggunakan alamat IP sumber yang sama, koneksi akan tetap persisten (Anita Sumiati, Primantara Hari Trisnawan, Mochammad Ali Fauzi, 2020). Load balancer juga berperan penting dalam mengatasi permintaan dan menghindari "bottleneck" pada server yang bisa mengganggu pengalaman pengguna (Faris Muslim Azmi, Mahendra ,& Heru Nurwasito, 2019). Dari sisi high availability, system juga dituntut untuk selalu tersedia melayani request yang masuk dari pengguna, sehingga dibutuhkan docker pada cluster docker swarm memiliki kemampuan bawaan untuk mencapai high availability dengan menjalankan kontainer di seluruh host dalam cluster. Ini memastikan bahwa jika ada host yang mengalam masalah, aplikasi masih tetap berjalan di host lain (Yudi Hadiwandra T., Ferry Candra, 2020). Kemudian untuk meminimalkan effort dari administrator diperlukan Perubahan di bidang teknologi, seperti percepatan penyebaran informasi, kemudahan akses terhadap informasi, dan semakin banyaknya Masyarakat yang mulai melek teknologi, telah menyebabkan perubahan kenaikan dalam lalu lintas internet dan menempatkan tekanan besar pada infrastruktur server (Ahmad Firmansyah, 2017). Pada infrastruktur IT ada beberapa jenis hardware yang bisa digunakan sebagai server, salah satunya Raspberry, Raspberry Pi memberikan keunggulan salah satunya di sisi sumber daya yang kecil untuk sebuah komputer dan juga biaya yang dikeluarkan untuk Raspberry Pi ini juga jauh lebih terjangkau dari komputer konvensional serta dapat terhubung ke dalam jaringan baik dengan menggunakan koneksi LAN maupun melalui koneksi nirkabel (Albert Darmaliputra, Henry Hermawan, 2014). Agar system yang dirancang dengan Raspberry ini dapat memproses request semaksimal mungkin serta mengatur jumlah traffic yang masuk sesuai dengan sumber daya yang ada, dibutuhkan load balancer yang dapat mengoptimalkan distribusi lalu lintas di antara server yang ada dalam lingkungan cluster. Ini membantu menghindari overload pada server tertentu dan memastikan penggunaan sumber daya yang lebih merata (Faris Muslim Azmi, Mahendra ,& Heru Nurwasito, 2019). Metode IP Hash adalah algoritma load balancing yang mengambil source IP dari client untuk menghasilkan kunci hash yang unik. Kunci ini digunakan untuk mengalokasikan client ke server tertentu. Metode penyeimbangan beban ini dapat memastikan bahwa client diarahkan ke server yang sama dengan yang digunakan sebelumnya. Saat melakukan pengiriman Source IP Hash, fungsi Hash diterapkan pada alamat IP sumber dari permintaan yang masuk. Hal tersebut memastikan bahwa Ketika suatu koneksi melewati perangkat, selama itu ansible untuk mempermudah manajemen konfigurasi sehingga terpusat dari satu node, Ansible merupakan tool bersifat open source dan otomatisasi agentless yang dapat digunakan untuk melakukan manajemen konfigurasi, penyebaran dan orkestrasi TI (Ni Made Anggrena Yalestia Chandrawaty, I Putu Hariyadi, 2021). Penelitian ini akan membahas mengenai analisis fungsi load balancer yang berada pada layer transport untuk mengatur pendistribusian beban kerja pada suatu cluster berbasis Raspberry. Node pada suatu cluster yang mengalami overload jumlah request akan mengalami “bottleneck” atau request yang masuk tidak dapat diimbangi dengan kinerja server sehingga akan menyebabkan down dan terganggunya sebuah service atau layanan, terlebih jika dalam lingkungan server tersebut tidak di terapkan dockerswarm yang dapat mengarahkan request yang masuk Ketika sebuah server down dengan cara mengalihkan request ke server lain yang aktif. Oleh karena itu, fokus dari penelitian adalah menganalisis efektivitas beban kerja server menggunakan metode loadbalancer IP hash dalam lingkup docker swarm cluster berbasis Raspberry yang akan mencegah, dan menangani kasus “bottleneck” pada server, serta mengefisienkan kinerja dari sebuah server sehingga dapat melayani jumlah request yang masuk secara maksimal.

2. METODE PENELITIAN

Pada penelitian ini, tipe penelitian yang akan digunakan adalah tipe penelitian implementatif dan bersifat pengembangan. Penelitian implementatif merujuk pada mengimplementasikan jenis hasil penelitian desain yang dan pengembangan yang sudah dilakukan secara nyata berupa produk baik perangkat lunak maupun perangkat keras untuk menyelesaikan suatu permasalahan, dengan dasar kajian teori yang sudah pernah diteliti oleh penelitian terdahulu. Pada penelitian implementatif yang bersifat pengembangan ini bertujuan untuk mengembangkan hasil penelitian terdahulu dengan membedakan beberapa faktor seperti metode, arsitektur dan juga data yang digunakan pada penelitian sebelumnya. Pada penelitian ini berfokus pada untuk mengetahui efektivitas atau pengaruh load balancer dengan beberapa metode menggunakan Raspberry untuk bisa membagi beban kerja secara baik.



Gambar 1. Diagram Penelitian

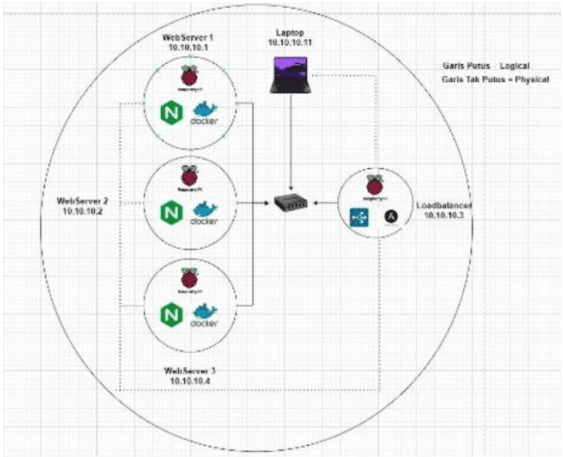
Langkah penelitian ini akan dilakukan secara berurutan sesuai dengan yang ditampilkan pada diagram penelitian, Penelitian dimulai dari Studi literatur bertujuan untuk melakukan Analisa penelitian terdahulu yang terkait dan mengevaluasi kelemahan atau kekurangan dari penelitian serta untuk memperdalam pemahaman terkait teori-teori yang ada, juga dengan bukti yang digunakan sebagai dasar hipotesis penelitian ini. Selanjutnya tahapan penelitian ini dilanjutkan dengan perancangan dan implementasi sistem bertujuan untuk memberi gambaran mengenai visualisasi sistem atau secara topologi dan sistem ketika sudah dibuat. Setelah sistem dirancang maka Langkah selanjutnya yaitu pengujian pada sistem memiliki tujuan untuk mendapatkan informasi-informasi yang dibutuhkan mengenai verifikasi dan validitas suatu system mengacu pada tahap yang telah dibuat sebelumnya. Kemudian penelitian diakhiri oleh tahap Pengambilan kesimpulan merupakan tujuan akhir dari penelitian ini. Kesimpulan merupakan hasil praktik dari hipotesis yang merupakan hasil teori.

2.1. Teknik Pengambilan Data dan Pembagian Beban

Teknik pengambilan data pada penelitian yang berjudul Pengembangan Pembagian Beban Kerja Server Cluster Docker Swarm Berbasis Raspberry Dengan Metode Loadbalancer IP HASH Serta Terautomasi Ansible ini menggunakan tools generator request yaitu apache benchmark, ini adalah sebuah tools yang akan melakukan request berjumlah sesuai dengan kebutuhan peneliti, sedangkan untuk pembagian beban pada penelitian ini akan menggunakan algoritma IP HASH pada node controller.

3. PERANCANGAN DAN IMPLEMENTASI

Perancangan sistem merupakan tahapan mengenai gambaran umum sistem yang akan dirancang sebagai dasar untuk proses selanjutnya. Pada bagian perancangan dan implementasi, akan dijelaskan mengenai rancangan sistem secara keseluruhan, yang mencakup perancangan perangkat keras dan diagram alur perangkat lunak. Sementara itu, implementasi sistem akan melibatkan pengimplementasian perangkat keras dan juga perangkat lunak.



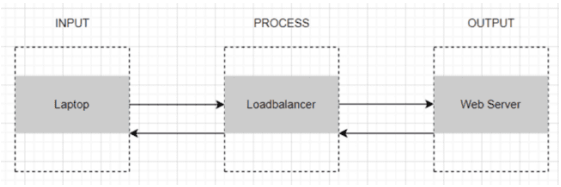
Gambar 2. Desain Perancangan Sistem

Pada gambar desain perancangan sistem diatas dapat dilihat pada sistem ini terdiri dari empat Raspberry Pi, dengan tiga di antaranya berperan sebagai server web menggunakan Nginx, dan satu sebagai kontroler. Sistem ini menggunakan teknologi containerization di lingkungan Docker Swarm dan diintegrasikan dengan Ansible untuk otomatisasi tugas administratif yang akan diimplementasikan pada kontroler. Ansible akan dikonfigurasi untuk melakukan zip pada image server 1, 2, dan 3 yang sudah dibuat dari Dockerfile kemudian akan melakukan pendistribusian image, pengesktrakan image hingga menghapus kembali file agar menghindari menumpuknya file tidak terpakai di server. Docker swarm berperan untuk manajemen container, kemudian loadbalancer berperan sebagai pembagi beban yang akan diteruskan kepada server dan juga melakukan ketersediaan server dan melakukan pengalihan ketika server yang dituju mati.

3.1. Perancangan Perangkat Keras

Diagram Blok adalah sebuah diagram yang terdiri dari kotak-kotak (blok) yang digunakan untuk

menjelaskan suatu proses kerja dalam ilmu teknik. Diagram ini mencakup berbagai komponen seperti kotak dan garis yang saling terhubung untuk menggambarkan cara suatu sistem beroperasi. Fungsi utama dari diagram blok adalah sebagai panduan untuk menjelaskan cara kerja sebuah sistem yang akan dibuat, serta sebagai rancangan awal dari system yang akan dikembangkan. Diagram blok system dalam penelitian ini dapat dilihat pada Gambar 3 berikut ini.



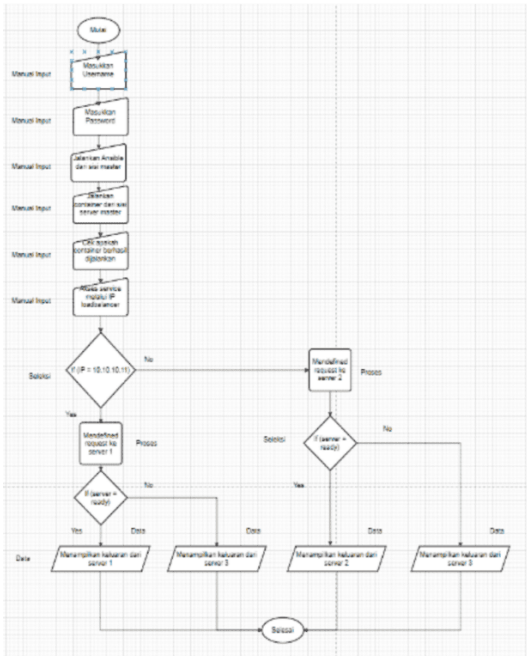
Gambar 3. Diagram blok system

Pada gambar di atas, di sebelah kiri terdapat komponen penginput data yang berasal dari laptop, Permintaan yang diperoleh dari laptop tersebut akan masuk ke dalam sistem atau server, kemudian akan diproses untuk mengarahkan hasil permintaan tersebut ke server terkait sesuai dengan algoritma yang telah ditetapkan pada file di direktori site enabled pada Nginx. Setelah pemrosesan selesai dan permintaan telah mendapatkan alamat server yang dituju, tahap terakhir adalah meneruskan permintaan tersebut ke server terkait untuk mengakses isi dari webserver tersebut. Semua perangkat terhubung melalui jaringan LAN atau konektivitas WIFI.

3.2. Perancangan Perangkat Lunak

Perancangan perangkat lunak pada sistem akan terbagi menjadi 3 yaitu: perancangan task pada ansible, perancangan algoritma pada loadbalancer, serta perancangan isi pada webserver. Pada proses perancangan task pada ansible, hal yang dilakukan yaitu menentukan serta menkonfigurasi hal-hal atau task apa saja yang akan diautomasi oleh ansible, dalam hal ini yaitu hal yang berkaitan dengan distribusi image antar server yang akan digunakan untuk image pada docker container, lalu pada tahapan kedua yaitu perancangan algoritma loadbalancer, pada tahap ini algoritma yang digunakan yaitu IP HASH serta menambahkan beberapa algoritma seperti pengecekan server status, dan melakukan pengalihan server jika server yang dituju down. Flowchart pada sistem ini dapat dilihat pada gambar 5.

Berikut merupakan penjelasan pada flowchart sistem. Pertama Langkah kerja sistem dimulai dengan memberi daya pada sistem atau menyalakan sistem, setelah sistem menyala, langkah kedua yaitu memasukan username dan password yang sudah disetting pada sistem, setelah berhasil masuk ke dalam sistem, langkah berikutnya yaitu menjalankan ansible pada sisi master untuk mengautomasi



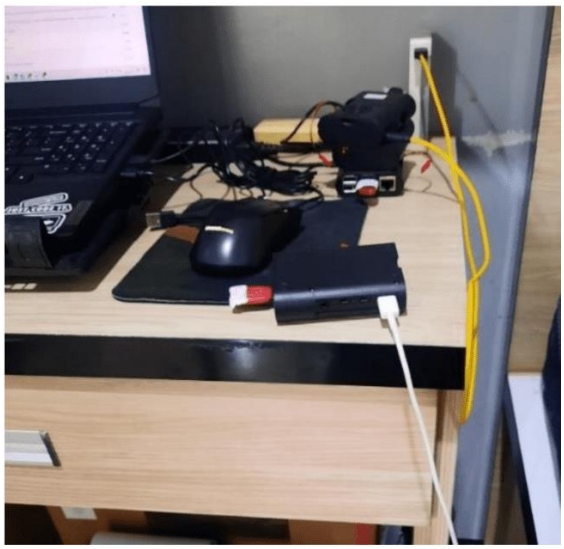
Gambar 5. Flowchart Sistem

Langkah zip image, distribusi image, melakukan load pada file zip yang berisi image docker, lalu menghapus file sampah. Setelah image berhasil diload disetiap perangkat, jalankan container dari sisi master yang ditujukan untuk slave yang ada di dalam lingkungan docker swarm, kemudian melakukan pengecekan untuk memastikan container berjalan semestinya. Setelah container berhasil berjalan, user bisa mengakses ip dari perangkat controller, setelah controller mendapat request, request yang masuk akan mengalami proses sesuai dengan algoritma yang sudah ditetapkan, dan akan diteruskan ke server yang sesuai dengan algoritma.

3.3. Implementasi Sistem

Implementasi sistem adalah penerapan dari sistem yang telah dikembangkan sesuai dengan rancangan yang telah dibuat sebelumnya. Pada bagian ini, akan ditunjukkan hasil perakitan sistem dari berbagai komponen yang telah disebutkan sebelumnya. Hasil perakitan perangkat keras dapat dilihat pada Gambar 6.

Dari gambar di atas dapat terlihat hasil perakitan sistem terdiri dari beberapa komponen perangkat keras. Kotak berwarna hitam yang terhubung kabel LAN merupakan Raspberry PI sebagai server, dalam penelitian ini menggunakan 4 server yang terdiri dari 4 Raspberry PI, 1 buah server berperan sebagai loadbalancer dan juga controller, sedangkan 2 buah server lain berperan sebagai webserver utama, dan 1 lainnya berperan sebagai webserver cadangan untuk mengantisipasi terjadinya kondisi down pada server utama.



Gambar 6. Keseluruhan Alat

Pada penelitian ini, pernghubung antar komponen bisa menggunakan kabel LAN seperti pada gambar dengan menggunakan 1 komponen tambahan yaitu hub maupun melalui jaringan nirkabel atau wifi, pada setiap Raspberry server, membutuhkan 1 buah flashdisk yang berfungsi untuk tempat penginstalan OS pada server, juga berfungsi sebagai tempat penyimpanan pada server, setiap Raspberry pun dapat diberi daya melalui listrik pada umumnya maupun melalui powerbank jika terjadi hal-hal darurat, dan dihubungkan menggunakan kabel type-c.

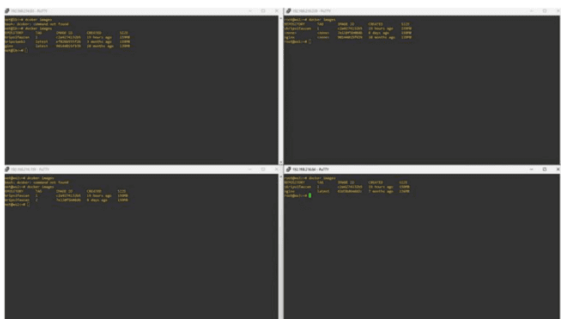
4. PENGUJIAN DAN ANALISIS

4.1. Pengujian Fungsionalitas Sistem

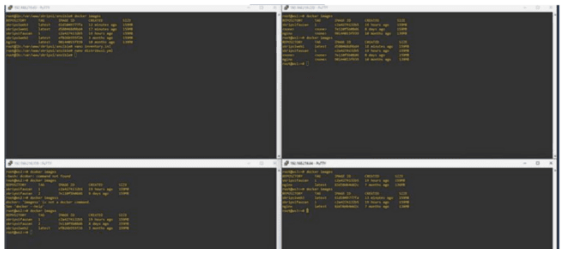
Pengujian fungsionalitas Sistem bertujuan untuk melakukan pengujian guna menilai fungsionalitas dari sistem yang telah dirancang dan diimplementasikan agar dapat berjalan sesuai dengan fungsi yang telah ditetapkan.

4.1.1. Pengujian Pendistribusian Image

Pengujian pendistribusian image adalah untuk memastikan apakah sistem yang dikembangkan dapat menjalankan fungsi otomatisasi pendistribusian image melalui satu node, yaitu swarm manager, dan dapat melakukan load image dari file yang sudah didistribusikan, proses pendistribusian image dapat dilihat pada gambar 7 hingga 8 dibawah ini.



Gambar 7. Image Awal Setiap Node

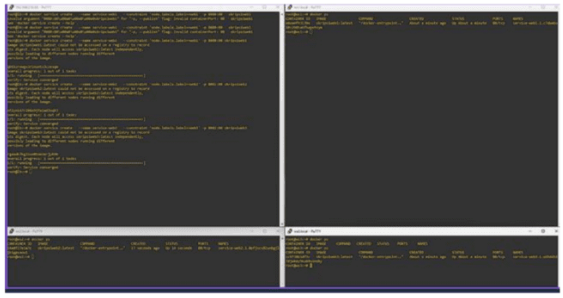


Gambar 8. Image Telah Berhasil Terdistribusi

Dari gambar diatas terlihat ansible telah berhasil melakukan tugas untuk mendistribusikan Image dari swarm manager atau master ke worker-worker yang ada pada lingkungan swarm, ansible berhasil melakukan penyalinan file image berekstensi .tar hingga meload image tersebut hingga bisa digunakan.

4.1.2. Pengujian Menjalankan Container

Pengujian menjalankan container bertujuan untuk memastikan apakah sistem yang dikembangkan mampu menjalankan fungsi untuk menjalankan sebuah layanan pada server tertentu yang ada dalam lingkup swarm melalui satu node, yaitu swarm manager. proses menjalankan container dapat dilihat pada gambar dibawah ini.

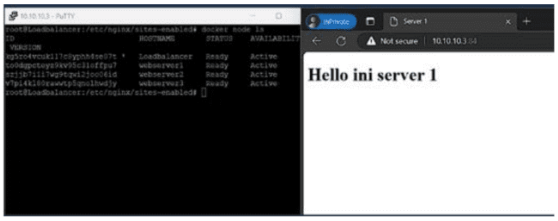


Gambar 9.Service Telah Berhasil Dideploy

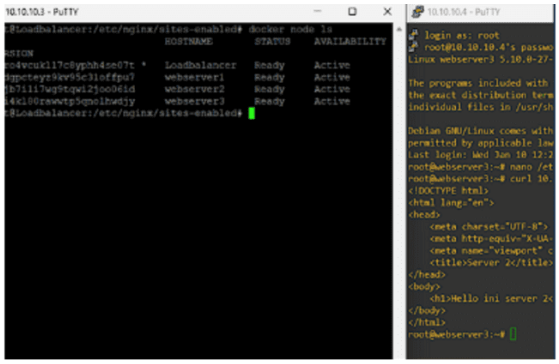
Pada gambar di atas terlihat bahwa sistem telah berhasil mendeploy sebuah service dari node controller kemudian diarahkan kepada node 1, 2, dan dengan masing- masing service yaitu web1, web2, dan
web3, terlihat juga service berhasil berjalan.

4.1.3. Pengujian Loadbalancer

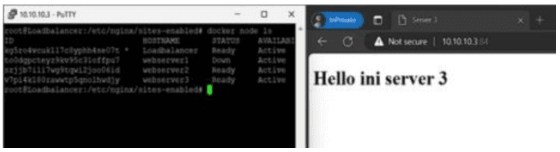
Pengujian loadbalancer bertujuan untuk menentukan apakah sistem yang dikembangkan dapat menjalankan algoritma IP Hash sesuai dengan konfigurasi yang ditetapkan pada konfigurasi nginx untuk membagi request atau beban kerja yang masuk. proses menjalankan container dapat dilihat pada gambar dibawah ini.



Gambar 10. sistem berhasil mengalihkan request dari IP 10.10.10.11 ke server 1



Gambar 11. sistem berhasil mengalihkan request dari IP selain 10.10.10.11 ke server 2



Gambar 12. sistem berhasil mengalihkan Ketika server 1 dan 2 down

Gambar diatas menunjukkan bahwa system sudah berhasil memberikan tampilan sesuai dengan konfigurasi pada loadbalancer ketika ada request, kemudian sistem juga telah berhasil mengalihkan request yang masuk ke node 3 ketika node 1 atau 2 mengalami down.

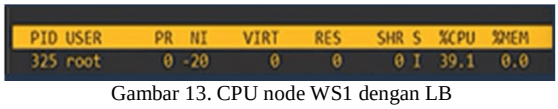
4.2. Pengujian Kinerja Sistem

Pengujian kinerja sistem bertujuan untuk menguji ketahanan kinerja dan sejauh mana system yang menerapkan load balancer dapat bertahan Ketika diberi beban yang banyak, serta untuk mengetahui jumlah request yang berhasil dilayani oleh sistem. Selain itu, pada tahap ini juga akan membandingkan kinerja sistem yang menggunakan load balancer dengan sistem yang tidak menggunakan load balancer, dengan tujuan untuk mengevaluasi pengaruh dari algoritma load balancer yang digunakan.

4.2.1. Pengujian Apache Benchmark

Pengujian Apache Benchmark adalah pengujian yang dilakukan untuk mengetahui sejauh mana dan seberapa banyak sistem dapat melayani permintaan dalam jumlah banyak serta melihat seberapa banyak request yang berhasil dilayani dan yang tidak dapat dilayani atau terbuang.

4.2.1.1. Pengujian kinerja Menggunakan Apache Benchmark pengujian dengan jumlah request 1000 dengan user 10.10.10.11 (mengarah ke server 1)



Gambar 13. CPU node WS1 dengan LB

Gambar 14. CPU node WS1 Tanpa LB

Pada gambar pengujian di atas terlihat Ketika sistem diberikan request sebanyak 1000 request melalui loadbalancer dan diakses oleh IP laptop peneliti, hasil menunjukkan bahwa sistem dapat melayani semua request berdasarkan parameter Complete Request yaitu 1000 yang dimana berarti seluruh request dapat dilayani, kemudian cpu dari server yang diakses yaitu ws1 juga menunjukkan bahwa request yang melewati loadbalancer mendapat cpu usage lebih kecil dibanding tanpa loadbalancer yaitu 39.1% sedangkan tanpa loadbalancer yaitu 51.2%.

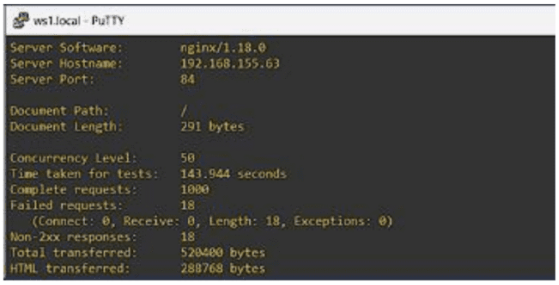
4.2.1.2. Pengujian kinerja Menggunakan Apache Benchmark Pengujian dengan jumlah request 1000 dengan user selain 10.10.10.11 (mengarah ke server 2)

Gambar 15. CPU node WS2 dengan LB

Gambar 16. CPU node WS2Tanpa LB

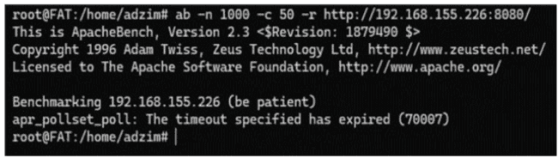
Pada gambar pengujian di atas terlihat Ketika sistem diberikan request sebanyak 1k melalui loadbalancer dan diakses oleh selain IP laptop peneliti hasil menunjukkan bahwa sistem dapat melayani semua request berdasarkan parameter Complete Request yaitu 1000 yang dimana berarti seluruh request dapat dilayani, kemudian cpu dari server yang diakses yaitu ws2 juga menunjukkan bahwa request yang melewati loadbalancer mendapat cpu usage lebih kecil dibanding tanpa loadbalancer yaitu 38.9% sedangkan tanpa loadbalancer yaitu 51.7%.

4.2.1.3. Pengujian kinerja Menggunakan Apache Benchmark Pengujian dengan jumlah request 1k ke server 2 ketika server 2 mati (dialihkan ke server 3)



Gambar 17. Pengujian request 1k lb ke server 3

Gambar 18. CPU node WS3 dengan LB

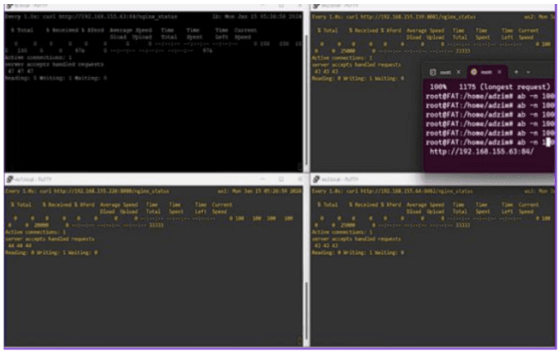


Gambar 19. Pengujian request 1k no lb ke server 3

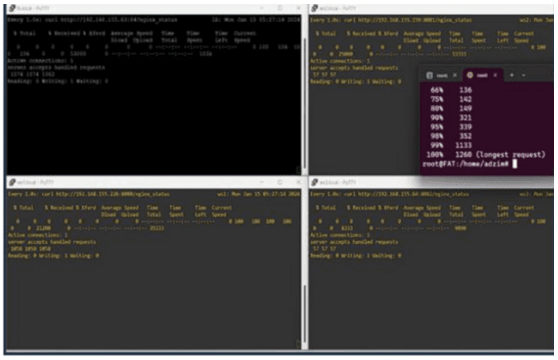
Pada gambar pengujian di atas terlihat Ketika system diberikan request sebanyak 1000 melalui loadbalancer dan diakses oleh IP laptop peneliti hasil menunjukkan bahwa terdapat 18 request yang tidak dapat dilayani dikarenakan ketika server mati, setiap request akan dilakukan cek server status oleh loadbalancer yang menyebabkan timeout pada beberapa request, sedangkan apabila tanpa loadbalancer, maka semua request tidak dapat dilayani, sedangkan untuk cpu usagenya berada di angka 6.3%.

4.2.2. Pembagian Beban Load

Pembagian beban load adalah subbab yang bertujuan untuk mengetahui jumlah beban yang diterima server ketika dilakukan loadtest, hal ini juga berguna untuk mengetahui apakah skema IP HASH ini berjalan atau tidak, Jika beban yang di hasilkan oleh satu Alamat IP diteruskan ke satu server tertentu menunjukkan algoritma yang di buat berhasil diterapkan. Pembuktian beban load dapat dilihati pada gambar-gambar dibawah ini yang menunjukan perubahan pada parameter server accepts handled request.



Gambar 20. Pengujian request 1k no lb ke server 3



Gambar 21. Pengujian request 1k no lb ke server 3

Pada gambar diatas menunjukkan total 1000 request berhasil dialihkan semua ke server 1 dibuktikan oleh perubahan parameter server accepts handled requests dikarenakan dengan algoritma ini request yang masuk akan diteruskan kepada server yang sama sesuai dengan algoritma IP HASH.

5. KESIMPULAN DAN SARAN

Setelah melalui berbagai tahapan penelitian, termasuk perancangan, implementasi, dan pengujian, diperoleh beberapa kesimpulan. Pertama, load balancer berhasil membagi beban sesuai konfigurasi Nginx, dengan persentase pembagian beban yang persisten sebesar 100% pada satu server. Kedua, pengujian kinerja menggunakan Apache Benchmark menunjukkan bahwa request melalui node load balancer memiliki penggunaan CPU yang lebih rendah dibandingkan dengan sistem tanpa load balancer. Ketiga, saat server tujuan mengalami down, sistem melakukan pengecekan status server dan mengarahkan request ke server lain jika statusnya tidak siap. Namun, sistem mengalami penurunan performa saat server down, dengan kegagalan melayani 1,8% dari 1000 request atau 18 request. Penelitian ini menunjukkan hasil yang baik meskipun ada beberapa kekurangan. Untuk penelitian selanjutnya, beberapa saran dapat dipertimbangkan. Pertama, peningkatan perangkat keras untuk memastikan kenyamanan pengguna dan kehandalan sistem yang lebih baik. Kedua, menambahkan proses pengiriman data atau backend untuk menguji kelayakan Raspberry Pi untuk layanan yang lebih berat dan mengintegrasikannya dengan CI/CD seperti GitLab-CI atau penjadwalan seperti cronjob. Ketiga, menambah cluster agar dapat menjalankan beberapa layanan secara bersamaan di berbagai cluster. Terakhir, memperbaiki algoritma agar lebih efisien dan dapat melayani semua request ketika salah satu server down.

DAFTAR PUSTAKA

STEFANUS, EKO, P., YULFAN, SALIMIN, 2021. Analisis Perbandingan Performa Web Server Docker Swarm dengan Kubernetes Cluster. S1. Universitas Internasional Batam.

FAJAR, RAHAYU, I. M., MOHAMMAD, RACHMAN, W., TERIS, E. W., 2020. Analisis Kepuasan Optimalisasi Kinerja Jaringan Berbasis Load Balancing dan Wifi Offload Menggunakan Uji T Paired (Studi Kasus pada UPN Veteran Jakarta). S2. Institut Teknologi Nasional Malang.

AGUNG, N. M., RIKIE, K., 2016. Analisis Kinerja Penerapan Container untuk Load Balancing Web Server Pada Raspberry PI. S1. STKIP PGRI Tulungagung.

SAMPURNA, D. R., DONARYA, P., 2020 Analisis Perbandingan Server Load Balancing dengan Haproxy & Nginx dalam Mendukung Kinerja Server E-Learning. S1. Universitas Teknokrat Indonesia.

YUDI, H. T., FERI, C., 2021. High Availability Server Using Raspberry Pi 4 Cluster and Docker Swarm. S1. Universitas Riau.

RYAN, A. P., ASPRIANTI, P. S., 2019. Implementasi Cluster Server Pada Raspberry Pi Dengan Metode Load Balancing. S1. Universitas Komputer Indonesia.

LALU, F. I. A., ADHITYA, B., WIDHI, Y., 2018. Implementasi Load Balancer Berdasarkan Server Status pada Arsitektur Software Defined Network (SDN). S1. Universitas Brawijaya.

NAIK, N. 2016. Building A Virtual System of Systems Using Docker Swarm in Multiple Clouds.

NAIK, N. 2016. Migrating from Virtualization to Dockerization in the Cloud: Simulation and Evaluation of Distributed Systems. 2016 IEEE 10th International Symposium on the Maintenance and Evolution of Service Oriented and Cloud-Based Environments.

DIMAS, S. A., MAHENDRA, D., WIDHI, Y., 2019. Load Balancing Server Web Berdasarkan Jumlah Koneksi Klien Pada Docker Swarm. S1. Universitas Brawijaya.

AGUS AAN JIWA P., 2014. Pengembangan Lab Komputer Sederhana Berbasis Jaringan Multipoint Menggunakan Switch Sebagai Sarana Penunjang Proses Pembelajaran. S1. Universitas Pendidikan Ganesha Singaraja Indonesia.

FARIS MUSLIM A., MAHENDRA D., HERU N., 2019. Perbandingan Kinerja Haproxy dan Zevenet Dalam Pengimplementasian Multi Service Load Balancing. S1. Universitas Brawijaya.

KISHOR TRIVEDI S., ANDREA B., 2017. Reliability and Availability Engineering Modeling, Analysis, and Applications. Shaftesbury Road: Cambridge University Press.

NI MADE ANGGRENA YALESTIA C., I PUTU H., 2021. Implementasi Ansible Playbook untuk Mengotomatisasi Manajemen Konfigurasi VLAN Berbasis VTP dan Layanan DHCP. S1. Universitas Bumigora.

ANITA, S., PRIMANTARA, H. T., MOCHAMMAD, A. F., 2020. Implementasi Load Balancing Web Server dengan Algoritma Source IP Hash pada Software Defined Network (SDN). S1. Universitas Brawijaya.

ALBERT, D., HENRY, H., 2014. Pembuatan Web Server Berbasis Raspberry PI untuk Kontrol Lampu dan AC. S1. Universitas Surabaya.

INDRA, M., Implementasi Raspberry Pi 4 Sebagai Server E-Learning in Jurnal Media Aplikom, Vol. 13 No. 2, Purwokerto: STIKOM Yos Sudarso Publisher. 2021, pp. 1-14.