PERANCANGAN ARSITEKTUR WEB SERVER MODERN MENGGUNAKAN PENDEKATAN MICROSERVICES BERBASIS DOCKER CONTAINER

Muhamad Kamaludin[1], Muhammad Khoirul Mauludi[2], Muhammad Amar Subhan[3], Muhamad Azhar Muzhaffar[4], Marganing Rifki Nugroho[5]
muhamad.kamaludin@raharja.info[1], khoirul.mauludi@raharja.info[2], muhammad.amar@raharja.info[3], muhamad.azhar@raharja.info[4], marganing.rifki@raharja.info[5]
Universitas Raharja

Abstrak

Perkembangan teknologi pengembangan perangkat lunak mendorong munculnya berbagai pendekatan baru dalam membangun sistem yang lebih fleksibel dan mudah dikembangkan. Salah satu pendekatan yang banyak digunakan saat ini adalah arsitektur microservices yang memecah aplikasi menjadi beberapa layanan kecil yang saling terhubung. Bersamaan dengan itu, teknologi container seperti Docker dimanfaatkan untuk mempermudah proses pembangunan, pengujian, serta distribusi aplikasi secara konsisten pada berbagai lingkungan sistem. Penelitian ini bertujuan untuk mengkaji penerapan arsitektur microservices dalam perancangan web server dengan memanfaatkan teknologi Docker container. Metode penelitian yang digunakan adalah studi literatur dengan menelaah berbagai sumber ilmiah yang berkaitan dengan konsep microservices, containerisasi, serta implementasinya pada sistem berbasis web. Hasil kajian menunjukkan bahwa penggunaan Docker container mampu mendukung proses pengembangan aplikasi yang lebih terstruktur, meningkatkan efisiensi pengelolaan sumber daya, serta mempermudah proses deployment layanan. Selain itu, isolasi antar container juga memberikan keuntungan dalam hal stabilitas dan pemeliharaan sistem. Berdasarkan hasil analisis tersebut, dapat disimpulkan bahwa penerapan arsitektur microservices yang dikombinasikan dengan teknologi container merupakan pendekatan yang efektif dalam merancang web server yang lebih skalabel dan mudah dikelola.

Kata Kunci: Microservices, Web Server, Docker Container, Arsitektur Perangkat Lunak, Containerization.

PENDAHULUAN

Perkembangan teknologi digital yang semakin pesat mendorong kebutuhan akan aplikasi web yang memiliki kinerja tinggi, responsif, serta mampu menangani peningkatan jumlah pengguna secara efektif. Oleh karena itu, organisasi dan perusahaan teknologi mulai mengadopsi berbagai pendekatan arsitektur perangkat lunak yang lebih fleksibel dan mudah dikembangkan. Salah satu pendekatan yang banyak digunakan saat ini adalah arsitektur microservices. Pendekatan ini memungkinkan sebuah aplikasi dipecah menjadi beberapa layanan kecil yang saling terhubung namun dapat dikembangkan, diuji, dan diimplementasikan secara terpisah. Dengan demikian, sistem menjadi lebih fleksibel, mudah dipelihara, serta dapat dikembangkan secara berkelanjutan sesuai kebutuhan.

Salah satu teknologi yang mendukung penerapan arsitektur microservices adalah Docker, yaitu sebuah platform sumber terbuka yang digunakan untuk mengotomatisasi proses pembangunan, distribusi, dan eksekusi aplikasi melalui mekanisme container. Melalui teknologi ini, setiap layanan dalam arsitektur microservices dapat dikemas ke dalam container yang bersifat terisolasi dan dilengkapi dengan seluruh dependensi yang diperlukan. Pendekatan tersebut memungkinkan aplikasi dijalankan secara konsisten pada berbagai lingkungan sistem, baik pada tahap pengembangan, pengujian, maupun implementasi di lingkungan produksi.

Penerapan teknologi container melalui Docker memberikan sejumlah keunggulan, seperti portabilitas yang tinggi, penggunaan sumber daya yang lebih efisien, serta

kemudahan dalam pengelolaan sistem. Dibandingkan dengan mesin virtual tradisional, container memiliki ukuran yang lebih ringan dan waktu eksekusi yang lebih cepat karena memanfaatkan kernel sistem operasi yang sama tanpa mengurangi tingkat isolasi antar layanan. Kondisi ini menjadikan Docker sebagai solusi yang efektif dalam menjalankan layanan microservices yang membutuhkan efisiensi sumber daya serta kemampuan beradaptasi terhadap perubahan sistem secara cepat.

Dalam proses perancangan web server, pemanfaatan arsitektur microservices berbasis container memberikan berbagai keuntungan. Setiap layanan dapat dikembangkan dan dikelola secara independen sehingga proses pembaruan sistem dapat dilakukan tanpa memengaruhi keseluruhan aplikasi. Selain itu, sistem juga dapat dengan mudah ditingkatkan skalabilitasnya melalui penambahan atau pengurangan jumlah container sesuai dengan kebutuhan beban kerja. Proses pengelolaan layanan juga dapat didukung oleh platform orkestrasi seperti Kubernetes yang berfungsi untuk mengatur deployment, pemantauan, serta pengelolaan siklus hidup container secara otomatis.

Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk merancang serta mengimplementasikan web server dengan memanfaatkan arsitektur microservices berbasis container. Penelitian ini juga berfokus pada analisis manfaat dan tantangan dalam penerapan teknologi tersebut, serta melakukan evaluasi terhadap kinerja dan skalabilitas sistem yang dihasilkan. Dengan adanya penelitian ini, diharapkan dapat memberikan kontribusi dalam pengembangan pemahaman mengenai penerapan teknologi containerisasi dalam pembangunan aplikasi web modern.

METODOLOGI

Penelitian ini menggunakan metode studi literatur dengan pendekatan kualitatif untuk memperoleh pemahaman yang komprehensif mengenai berbagai konsep, pendekatan, serta hasil penelitian yang berkaitan dengan perancangan web server menggunakan arsitektur microservices berbasis container. Pendekatan kualitatif dipilih karena metode ini memungkinkan peneliti melakukan kajian secara lebih mendalam terhadap berbagai sumber informasi dan pemikiran para peneliti sebelumnya mengenai penerapan teknologi microservices dalam pengembangan sistem berbasis web.

Proses pengumpulan data dilakukan dengan menelaah berbagai sumber referensi yang relevan, seperti artikel jurnal ilmiah, buku akademik, serta publikasi penelitian yang tersedia pada basis data ilmiah terpercaya. Pemilihan literatur dilakukan secara selektif dengan mempertimbangkan kesesuaian topik penelitian, kualitas metode yang digunakan dalam penelitian sebelumnya, serta kredibilitas penulis dan penerbit. Melalui proses seleksi tersebut, diperoleh sumber-sumber literatur yang dianggap mampu memberikan landasan teoritis yang kuat bagi penelitian ini.

Selanjutnya, data yang diperoleh dari berbagai referensi dianalisis menggunakan teknik analisis tematik, yaitu dengan mengidentifikasi, mengelompokkan, serta menafsirkan berbagai tema utama yang berkaitan dengan penerapan arsitektur microservices dan pemanfaatan teknologi container seperti Docker dalam perancangan web server. Melalui proses analisis tersebut, diharapkan dapat diperoleh pemahaman yang lebih sistematis mengenai konsep, manfaat, serta tantangan yang berkaitan dengan implementasi teknologi tersebut dalam pengembangan sistem berbasis web.

HASIL DAN PEMBAHASAN

Hasil Penelitian

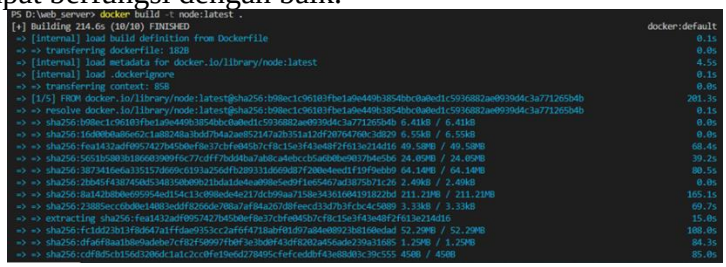
Dalam rangka memastikan bahwa aplikasi web yang dibangun menggunakan teknologi container dapat berjalan dengan baik, dilakukan serangkaian pengujian untuk

memverifikasi fungsi sistem serta menilai kinerja yang dihasilkan. Proses pengujian dilakukan melalui dua lingkungan utama, yaitu pada host yang menjalankan container serta pada komputer client yang terhubung ke sistem tersebut. Melalui tahapan pengujian ini, peneliti dapat mengetahui apakah sistem web server yang telah dirancang mampu beroperasi sesuai dengan konfigurasi yang telah ditetapkan. Selain itu, pengujian ini juga bertujuan untuk mengevaluasi stabilitas serta performa layanan yang dijalankan menggunakan teknologi container seperti Docker. Hasil dari pengujian tersebut kemudian dianalisis untuk memastikan bahwa seluruh komponen sistem dapat berfungsi secara optimal.

Hasil Pengujian Fungsional

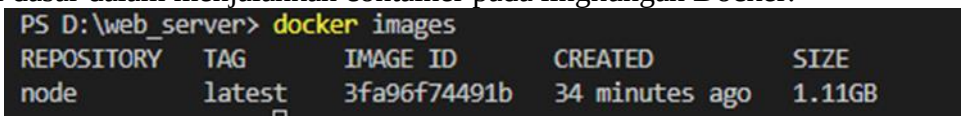
Setelah proses perancangan sistem, instalasi perangkat lunak, serta konfigurasi lingkungan server selesai dilakukan, tahap selanjutnya adalah implementasi sistem yang telah dibangun. Pada tahap ini, seluruh komponen yang telah dikonfigurasi dijalankan untuk memastikan bahwa layanan web server dapat beroperasi sesuai dengan rancangan awal. Dalam teknologi container, sebuah image digunakan sebagai dasar untuk menjalankan container yang berisi aplikasi dan seluruh dependensi yang dibutuhkan.

Oleh karena itu, pada tahap implementasi ini peneliti melakukan proses pembuatan image aplikasi berbasis Node.js dengan memanfaatkan file konfigurasi Dockerfile. Dockerfile digunakan untuk mendefinisikan langkah-langkah yang diperlukan dalam membangun image aplikasi sehingga container dapat dijalankan secara otomatis. Setelah proses pembangunan image selesai, container kemudian dijalankan untuk mengoperasikan layanan aplikasi yang telah dikembangkan. Seluruh container yang telah dibuat oleh peneliti dijalankan sesuai dengan konfigurasi yang telah ditentukan untuk memastikan bahwa sistem dapat berfungsi dengan baik.



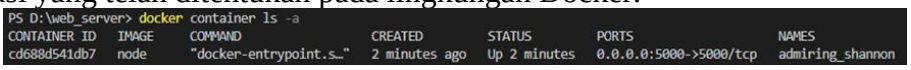
Gambar 1. Instalasi Docker Image

Gambar tersebut menunjukkan proses pembuatan dan instalasi image yang dilakukan menggunakan konfigurasi pada Dockerfile. Pada tahap ini, sistem membangun image yang berisi aplikasi beserta seluruh dependensi yang diperlukan sehingga dapat digunakan sebagai dasar dalam menjalankan container pada lingkungan Docker.



Gambar 2. Docker Images

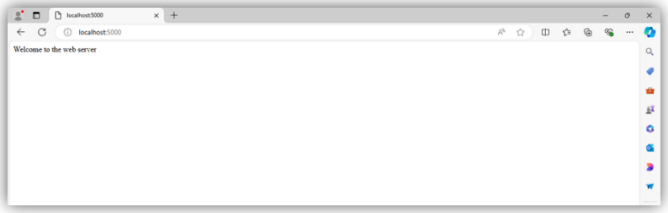
Gambar tersebut menampilkan hasil proses pembangunan (build) image aplikasi berbasis Node.js yang dibuat menggunakan konfigurasi Dockerfile. Image tersebut dibangun secara lokal tanpa melakukan pengunduhan langsung dari repositori resmi Docker, sehingga seluruh komponen aplikasi dan dependensinya disusun sesuai dengan konfigurasi yang telah ditentukan pada lingkungan Docker.



Gambar 3. Docker Container

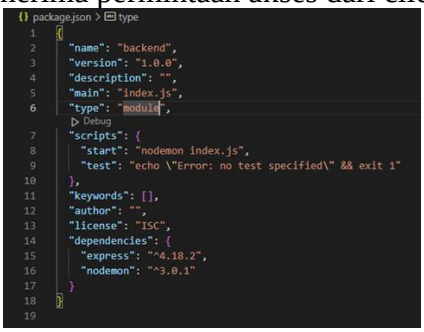
Gambar tersebut memperlihatkan container yang telah berhasil dibuat dari image yang sebelumnya dibangun. Pada tahap ini, container dijalankan dengan konfigurasi

tertentu seperti penentuan nama container, pengaturan port, serta parameter lain yang diperlukan agar aplikasi dapat berjalan dengan baik pada lingkungan Docker.



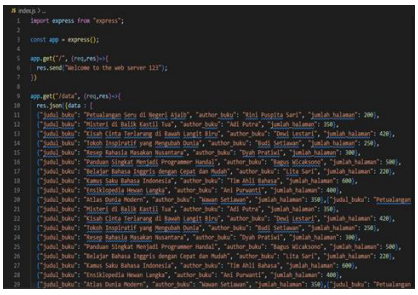
Gambar 4. Tampilan Web Server

Gambar tersebut menampilkan halaman web server berbasis Node.js yang berhasil diakses melalui browser pada komputer client. Hal ini menunjukkan bahwa container yang dijalankan pada lingkungan Docker telah berfungsi dengan baik dan mampu menjalankan layanan web server serta menerima permintaan akses dari client melalui jaringan.

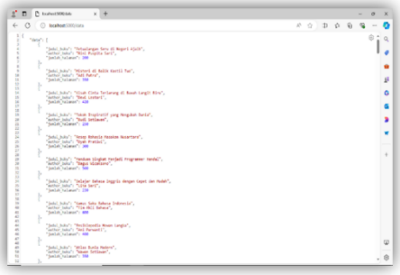


Gambar 5. Konfigurasi Web Server

Gambar tersebut menunjukkan proses konfigurasi pada aplikasi berbasis Node.js yang dilakukan agar web server dapat mengenali serta mengeksekusi berkas utama aplikasi, yaitu index.js. Konfigurasi ini memungkinkan server memproses permintaan dari pengguna dan menampilkan hasilnya melalui layanan web yang dijalankan di dalam lingkungan Docker.



Gambar 6. Tampilan File Index.js



Gambar 7. Tampilan Web Server Node js

Gambar pertama menunjukkan isi dari file index.js yang digunakan sebagai skrip utama untuk menjalankan aplikasi berbasis Node.js pada web server. File tersebut berfungsi agar server dapat membaca serta memproses kode JavaScript yang digunakan dalam aplikasi. Sementara itu, gambar kedua memperlihatkan hasil pengujian web server yang telah dijalankan. Pengujian dilakukan melalui komputer client dengan menggunakan browser untuk memastikan bahwa web server Node.js telah berjalan dengan baik dan dapat menampilkan aplikasi JavaScript secara normal.

Hasil Pengujian Web Server

Pada tahap ini dilakukan proses pengujian untuk memastikan bahwa seluruh fungsi pada web server dapat berjalan dengan baik. Pengujian dilakukan dengan melakukan pemantauan sistem serta uji coba menjalankan dan menghentikan aplikasi guna memastikan layanan berfungsi sebagaimana mestinya. Proses pengujian dilakukan melalui

beberapa tahapan, baik dari sisi host Docker maupun dari komputer client yang terhubung dengan host tersebut. Melalui tahapan tersebut dapat diamati interaksi yang terjadi antara client dan server. Adapun langkah-langkah serta hasil dari pengujian web server yang telah dilakukan dijelaskan sebagai berikut.

Tabel 1. Pengujian Web Server

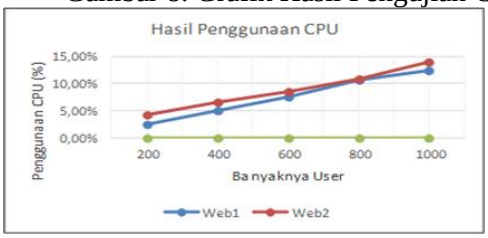
No.	Kebutuhan Fungsional	Hasil
1	Peneliti menambahkan image baru dari sebuah aplikasi ke server	OK
2	Peneliti membuat *container* baru dari sebuah image yang telah ditambahkan dari docker file	OK
3	Peneliti menjalankan dan menghentikan *container* yang telah dibuat	OK
4	Peneliti menghapus *container* yang telah dibuat	OK
5	Peneliti berhasil mengkonfigurasikan docker metrics dan node exporter ke prometheus	OK
6	Peneliti berhasil menghubungkan prometheus ke dashboard grafana	OK

Hasil Penggunaan CPU

Tabel 2. Hasil Penggunaan CPU

User	Nama *Container*	
Request	Web 1	Web 2
200	2,50%	4,18%
400	4,98%	6,48%
600	7,51%	8,59%
800	10,63%	10,82%
1000	12,36%	13,87%

Gambar 8. Grafik Hasil Pengujian CPU



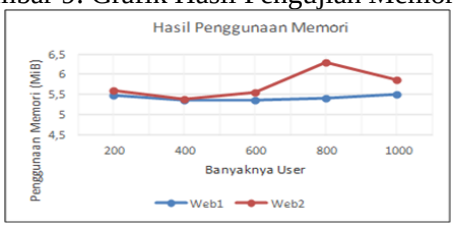
Pengukuran penggunaan CPU dilakukan dengan mencatat tingkat pemakaian CPU tertinggi pada setiap proses pengujian yang melibatkan jumlah pengguna yang berbeda pada masing-masing container yang sedang berjalan. Data yang dikumpulkan berasal dari statistik kinerja web server selama proses pengujian berlangsung, khususnya ketika pengujian penggunaan CPU sedang dilakukan.

Hasil Penggunaan Memori

Tabel 3. Hasil Penggunaan Memori

User	Nama *Container*	
Request	Web 1	Web 2
200	5.492 MiB	5.598 MiB
400	5.367 MiB	5.383 MiB
600	5.367 MiB	5.560 MiB
800	5.398 MiB	6.299 MiB
1000	5.508 MiB	5.876 MiB

Gambar 9. Grafik Hasil Pengujian Memori



Pengukuran penggunaan memori dilakukan dengan mencatat nilai pemakaian memori tertinggi pada setiap tahap pengujian yang menggunakan variasi jumlah pengguna pada masing-masing container yang sedang berjalan. Data tersebut diperoleh dari statistik kinerja web server selama proses pengujian berlangsung.

KESIMPULAN

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, penerapan web server menggunakan teknologi Docker container menunjukkan kemampuan dalam mengoptimalkan pemanfaatan sumber daya server, baik dari sisi perangkat keras maupun perangkat lunak. Teknologi container memungkinkan aplikasi dijalankan dalam lingkungan yang terisolasi dan konsisten sehingga dapat dioperasikan pada berbagai sistem

tanpa mengalami perbedaan konfigurasi yang signifikan. Selain meningkatkan efisiensi penggunaan sumber daya, pendekatan ini juga memberikan kemudahan bagi pengembang dalam proses pengembangan, pengujian, dan distribusi aplikasi. Dengan memanfaatkan Docker container, aplikasi dapat dijalankan secara lebih stabil pada berbagai lingkungan sistem tanpa harus melakukan penyesuaian konfigurasi yang kompleks.

Saran

Penelitian ini masih memiliki sejumlah keterbatasan yang perlu diperhatikan untuk pengembangan penelitian selanjutnya. Oleh karena itu, beberapa saran yang dapat dipertimbangkan untuk meningkatkan kualitas penelitian di masa mendatang antara lain sebagai berikut:

1. Melakukan analisis lebih mendalam untuk menentukan metode yang paling efektif dalam mengoptimalkan kinerja Docker container pada berbagai skenario penggunaan.
2. Mengkaji lebih lanjut aspek keamanan pada implementasi Docker container guna memastikan sistem yang dibangun memiliki tingkat perlindungan yang memadai.
3. Mengembangkan integrasi Docker dengan berbagai teknologi pendukung, seperti Kubernetes, pipeline CI/CD, serta layanan berbasis cloud, sehingga proses pengembangan dan operasional aplikasi dapat berlangsung lebih efisien dan terkelola dengan baik.

DAFTAR PUSTAKA

Adinta, F., & Neforawati, I. (2019). "Rancang Bangun Aplikasi Chatting Berbasis Web Menggunakan Docker". JOISIE (Journal Of Information Systems And Informatics Engineering), Vol. 1 (No.1), 28- 34

Afandi, A. T. (2020). Network Monitoring Berbasis Docker Container. Implementasi Network Monitoring Sistem Menggunakan Libernms Berbasis Dcoker Container, vol. 13, no. 01, (2021)

Alauddin, M. F. (2017). "Implementasi Virtual Data Center Menggungakan Linux Container Berbasis Docker dan SDN". Doctoral dissertation, Institut Teknologi Sepuluh Nopember, Vol. 6 (No.2), A440-A442.

Alauddin, M. F. (2017). Virtual Data Center. Implementasi Virtual Data Center Mengunakan Linux Container Berbasis Docker Dan Sdn, Vol. 6, No. 2 (2017), 2337-3520

Aziz, A. (2020). Webserver Nginx Dengan Lighttpd. Analisa Perbandingan Kinerja Webserver Nginx Dengan Lighttpd Untuk Kebutuhan Manajemen Web,

Bik, M. F. (2017). Pengelolan Banyak Aplikasi. Implementasi Docker Untuk Penglolaan Banyak Aplikasi, Vol 7 no 2 Tahun 2017, 46-50

Fs Ariadi, C. I. (2020). Docker Container Sebagai Teknologi. Penerapan Docker Container Sebagai Teknologi Ramah Skalabilitas Dibanding Teknik Virtualisasi Untuk Membangun Website Dubuntu 18.04 LTS, vol 8 no2 (2020): vol 8 no 2 Desember 2020

K, Arun. B, Vinutha. & B, Vinayaditya. (2019). "Real Time Monitoring Of Servers With Prometheus And Grafana For High Availability". International Research Journal of Engineering and Technology (IRJET), Vol. 6 (No.4) 5093-5096.

Saddiq, H. (2021). Load Balancing Webserver Menggunakan Container. Implementasi Dan Pengukuran Performasi Load Balacing Web Server Menggunakan Container, 20-35.

Santosa, M. W. I., Primananda, R., & Yahya, W (2018). "Implementasi Load Balancing Server Basis Data Pada Virtualisasi Berbasis Kontainer". Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, Vol. 2 (No.12), 6908-6914.

Sardi, E. S. (2017). Virtualisasi Container Dengan Docker. Implementasi Teknik Virtualisasi Container Dengan Docker Untuk Pengelolaan Aplikasi Web Dinas Komunikasi Dan Informatika Kota Payakumbuh

Sumbogo, Y. T., Data, M., & Siregar, R. A. (2018). "Implementasi Failover Dan Autoscaling Kontainer Web Server Nginx Pada Docker Menggunakan Kubernetes". Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, Vol. 2 (No.12), 6849-6854.