Analisis Kinerja NGINX pada Lingkungan Simulasi Docker Tanpa Limit dan Dengan Limit Memory 64MB CPU 0.2

Andika Sastra Praja[1], M.Syawalludin[2], Faridatul Munawaroh[3]
[1,2,3]Manajemen Informatika, Institut Teknologi Dan Bisnis Bina Sriwijaya, Kota Palembang, Indonesia

Article Info

Article history:

Received Dec 22, 2025
Revised Feb 16, 2026
Accepted Feb 16, 2026

Keywords:

Container;
Docker;
Kinerja;
NGINX;
Sumber Daya.

ABSTRACT

Perkembangan teknologi containerisasi telah mendorong penggunaan web server berbasis container, seperti NGINX, dalam lingkungan pembelajaran dan simulasi sistem terdistribusi. Salah satu tantangan utama dalam penerapan container adalah pengelolaan sumber daya, khususnya memory dan CPU, yang dapat memengaruhi kinerja dan stabilitas layanan. Penelitian ini bertujuan untuk menganalisis kinerja NGINX sebagai image container pada lingkungan simulasi Docker menggunakan platform Play-With-Docker, dengan membandingkan kondisi tanpa pembatasan sumber daya dan kondisi dengan pembatasan memory sebesar 64 MB serta CPU sebesar 0.2 core. Penelitian dilakukan melalui simulasi praktikum yang melibatkan 20 mahasiswa dengan spesifikasi laptop yang berbeda-beda, sehingga merepresentasikan kondisi penggunaan nyata dalam lingkungan pendidikan.Metode penelitian yang digunakan adalah pendekatan eksperimental dengan dua skenario pengujian. Parameter yang dianalisis meliputi waktu startup container, waktu respons HTTP, penggunaan memory, penggunaan CPU, serta stabilitas container. Hasil penelitian menunjukkan bahwa NGINX container tanpa pembatasan sumber daya memiliki kinerja yang lebih optimal dengan waktu respons yang lebih cepat dan stabilitas yang tinggi. Sebaliknya, pada kondisi dengan pembatasan resource, terjadi peningkatan waktu startup dan waktu respons, serta munculnya throttling CPU dan kejadian out-of-memory pada sebagian pengujian, meskipun penggunaan sumber daya menjadi lebih efisien.Kesimpulan dari penelitian ini menunjukkan bahwa pembatasan memory dan CPU berpengaruh signifikan terhadap kinerja NGINX container. Meskipun performa menurun, skenario dengan limit resource memiliki nilai edukatif yang tinggi dalam membantu mahasiswa memahami konsep manajemen dan optimalisasi sumber daya pada lingkungan containerized. Dengan demikian, Play-With-Docker dinilai efektif sebagai media simulasi pembelajaran container berbasis NGINX.

Corresponding Author:

Andika Sastra Praja,
Manajemen Informatika,
Institut Teknologi Dan Bisnis Bina Sriwijaya, Kota Palembang, Indonesia,
Email: andika789099@gmail.com.

How to Cite: A. S. Praja, M. Syawalludin, and F. Munawaroh, "Analisis Kinerja NGINX pada Lingkungan Simulasi Docker Tanpa Limit dan Dengan Limit Memory 64MB CPU 0.2," Journal of Logic, Informatics, and Algorithm, vol. 1, no. 1, pp. 26–33, Feb. 2026.

1 PENDAHULUAN

Perkembangan teknologi informasi dan komunikasi pada skala global telah mengalami akselerasi yang sangat pesat dalam dua dekade terakhir, seiring meningkatnya kebutuhan masyarakat terhadap layanan digital yang cepat, andal, dan dapat diakses secara luas. [1] Hampir seluruh sektor, mulai dari pemerintahan, pendidikan, industri, hingga layanan publik, kini mengandalkan sistem berbasis web sebagai tulang punggung operasional dan penyediaan informasi. Kondisi ini menuntut infrastruktur server yang tidak hanya mampu menangani jumlah permintaan yang besar, tetapi juga memiliki efisiensi tinggi dalam penggunaan sumber daya komputasi. [2] Web server sebagai komponen inti dalam arsitektur sistem berbasis web memiliki peranan strategis dalam menjamin ketersediaan layanan, kecepatan akses, serta stabilitas sistem secara keseluruhan.Salah satu web server yang paling banyak digunakan dan dikaji dalam berbagai penelitian adalah NGINX, yang dikenal luas karena arsitekturnya yang bersifat event-driven dan non-blocking. [3] Arsitektur ini memungkinkan NGINX menangani koneksi dalam jumlah besar secara simultan dengan penggunaan memori dan CPU yang relatif lebih rendah dibandingkan web server konvensional lainnya. Keunggulan tersebut menjadikan NGINX banyak diadopsi sebagai web server, reverse proxy, maupun load balancer pada berbagai skala sistem, mulai dari aplikasi kecil hingga layanan berskala besar. Dengan karakteristik tersebut, NGINX menjadi objek penelitian yang relevan untuk dianalisis kinerjanya, khususnya ketika dijalankan dalam lingkungan modern yang mengedepankan efisiensi dan fleksibilitas sumber daya.[4]

Memasuki era Society 5.0, paradigma pengembangan teknologi mengalami pergeseran dari sekadar digitalisasi menuju optimalisasi sistem yang berorientasi pada kebutuhan manusia dan keberlanjutan. Era ini menekankan integrasi antara dunia fisik dan dunia siber melalui pemanfaatan teknologi cerdas seperti komputasi awan, big data, kecerdasan buatan, serta virtualisasi. Dalam konteks tersebut, efisiensi penggunaan sumber daya menjadi isu yang sangat penting, mengingat meningkatnya jumlah layanan digital yang berjalan secara bersamaan dalam satu infrastruktur. [5] Teknologi container hadir sebagai solusi untuk mendukung kebutuhan tersebut dengan menyediakan lingkungan aplikasi yang ringan, terisolasi, dan mudah dikelola.Docker sebagai salah satu platform containerization yang paling populer memungkinkan pengembang dan administrator sistem untuk mengemas aplikasi beserta seluruh dependensinya ke dalam sebuah container yang portabel dan konsisten di berbagai lingkungan. Penerapan Docker tidak hanya mempercepat proses deployment, tetapi juga memudahkan pengujian dan simulasi konfigurasi sistem sebelum diimplementasikan pada lingkungan produksi.[6] Salah satu platform yang banyak digunakan untuk tujuan pembelajaran dan eksperimen adalah Play-With-Docker (PWD), yang menyediakan lingkungan Docker berbasis cloud tanpa memerlukan instalasi lokal. PWD menjadi sarana yang efektif untuk melakukan simulasi kinerja aplikasi berbasis container, termasuk pengujian berbagai konfigurasi sumber daya.

Meskipun Docker memberikan fleksibilitas dalam menjalankan aplikasi, pengelolaan sumber daya seperti memori dan CPU tetap menjadi aspek yang krusial. Dalam praktik nyata, container jarang dijalankan tanpa batasan resource, terutama pada lingkungan yang bersifat multi-tenant atau memiliki keterbatasan infrastruktur. Pemberian limit memory dan CPU bertujuan untuk mencegah satu container mendominasi sumber daya sistem dan mengganggu container lainnya. [7] Namun demikian, pembatasan tersebut juga berpotensi menurunkan performa aplikasi yang berjalan di dalam container, termasuk layanan web yang membutuhkan respons cepat dan stabilitas tinggi.Urgensi penelitian ini semakin meningkat seiring dengan meluasnya penggunaan NGINX sebagai image container dalam berbagai skenario, baik pada lingkungan pembelajaran, simulasi, maupun implementasi nyata berskala kecil. Meskipun NGINX dikenal sebagai web server yang ringan, kinerjanya tetap dipengaruhi oleh alokasi sumber daya yang diberikan. [8] Pembatasan memori yang terlalu kecil, seperti 64MB, serta pembatasan CPU hingga 0.2 core, dapat memengaruhi kemampuan server dalam menangani permintaan klien, mengelola proses, dan menjaga kestabilan layanan. Oleh karena itu, diperlukan kajian empiris untuk memahami sejauh mana pembatasan tersebut berdampak terhadap kinerja NGINX yang dijalankan dalam container.[9]

Berdasarkan kajian terhadap penelitian-penelitian sebelumnya, sebagian besar studi mengenai performa web server dan Docker container berfokus pada perbandingan antar web server, evaluasi performa container pada lingkungan produksi, atau analisis penggunaan sumber daya tanpa skenario pembatasan yang ketat. [10] Penelitian yang secara khusus membandingkan kinerja NGINX image container antara kondisi tanpa limit resource dan dengan limit memory serta CPU yang sangat terbatas, khususnya pada lingkungan simulasi Play-With-Docker, masih relatif jarang ditemukan. Kondisi ini menunjukkan adanya research gap yang perlu diisi melalui penelitian yang lebih spesifik dan terarah.Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk menganalisis kinerja NGINX (Image Container) pada lingkungan simulasi Docker Play-With-Docker dengan dua skenario konfigurasi sumber daya, yaitu tanpa pembatasan resource dan dengan pembatasan memory sebesar 64MB serta CPU sebesar 0.2 core. Analisis dilakukan untuk mengidentifikasi perbedaan kinerja yang muncul akibat penerapan limitasi resource, serta untuk mengevaluasi sejauh mana NGINX mampu mempertahankan performanya dalam kondisi sumber daya yang terbatas. Hasil penelitian ini diharapkan dapat memberikan kontribusi ilmiah berupa referensi bagi peneliti, mahasiswa, dan praktisi dalam merancang serta mengelola container NGINX secara lebih efisien dan optimal, khususnya pada lingkungan simulasi dan sistem dengan keterbatasan sumber daya.[11]

2 METODE PENELITIAN

Metode penelitian yang digunakan dalam penelitian ini adalah metode eksperimen kuantitatif, yang bertujuan untuk mengukur dan membandingkan kinerja NGINX sebagai image container secara objektif dan terukur berdasarkan data numerik

yang diperoleh dari hasil pengujian terkontrol. Pendekatan eksperimental kuantitatif dipilih karena memungkinkan pengamatan yang sistematis terhadap pengaruh perlakuan tertentu terhadap objek penelitian dalam kondisi yang relatif terkendali dan dapat direplikasi. Dalam penelitian ini, eksperimen dilakukan dengan dua skenario perlakuan yang berbeda, yaitu menjalankan NGINX image container tanpa pembatasan sumber daya serta menjalankannya dengan pembatasan sumber daya berupa limit memory sebesar 64 MB dan CPU sebesar 0.2 core. Kedua skenario tersebut diimplementasikan secara konsisten pada lingkungan simulasi berbasis cloud menggunakan platform Play-With-Docker (PWD), yang berperan sebagai media utama dalam pelaksanaan pengujian.[12] Platform ini dipilih karena mampu menyediakan lingkungan Docker secara instan, fleksibel, dan konsisten tanpa memerlukan penyediaan infrastruktur fisik secara mandiri, sehingga mendukung proses pengujian yang terstandarisasi dan meminimalkan potensi perbedaan hasil akibat variasi lingkungan. Sementara itu, image NGINX yang digunakan dalam penelitian ini merupakan image resmi yang diperoleh dari Docker Hub, dengan tujuan untuk menjamin keseragaman versi perangkat lunak, stabilitas sistem, serta konsistensi konfigurasi pada seluruh percobaan yang dilakukan.Dalam rancangan penelitian ini, variabel bebas ditetapkan sebagai konfigurasi sumber daya container, yang mencakup kondisi tanpa pembatasan sumber daya dan kondisi dengan pembatasan memory serta CPU. Variabel terikat berupa kinerja NGINX yang diukur melalui beberapa parameter kuantitatif utama, antara lain waktu respons layanan web, throughput dalam menangani permintaan akses, serta tingkat penggunaan memory dan CPU selama container beroperasi. Adapun variabel kontrol meliputi lingkungan pengujian, versi image NGINX yang digunakan, metode pengujian akses layanan, serta skenario pengujian yang dijaga tetap sama pada setiap percobaan, sehingga perbedaan hasil yang diperoleh dapat dikaitkan secara langsung dengan perlakuan yang diberikan.[13]

Tahapan eksperimen diawali dengan persiapan lingkungan Play-With-Docker, penarikan (pull) image NGINX dari repositori resmi Docker Hub, serta pembuatan dan eksekusi container pada masing-masing skenario konfigurasi yang telah ditentukan. Setelah container dijalankan, dilakukan verifikasi layanan untuk memastikan bahwa NGINX dapat berjalan dengan baik dan layanan web dapat diakses secara normal sebelum proses pengambilan data dilakukan. Selanjutnya, pengujian beban dilakukan dengan memberikan sejumlah permintaan akses secara berulang dalam periode waktu tertentu guna mensimulasikan kondisi penggunaan layanan web yang umum terjadi, di mana setiap skenario pengujian dijalankan lebih dari satu kali untuk memperoleh hasil yang lebih stabil, konsisten, dan representatif.[14] Data kinerja yang diperoleh dari seluruh tahapan pengujian dikumpulkan dalam bentuk nilai numerik dan dianalisis secara kuantitatif dengan membandingkan nilai rata-rata hasil pengujian pada kedua skenario perlakuan. Hasil analisis tersebut kemudian disajikan dalam bentuk tabel dan grafik untuk memudahkan proses interpretasi dan pembahasan. Melalui tahapan analisis ini, pada tahap akhir penelitian dapat ditarik kesimpulan yang komprehensif mengenai dampak pembatasan memory dan CPU terhadap kinerja NGINX image container pada lingkungan simulasi Docker menggunakan platform Play-With-Docker [15].

Gambar 1. Alur Metodelogi Penelitian

3 HASIL DAN PEMBAHASAN

Pengujian kinerja NGINX sebagai image container dilakukan pada lingkungan simulasi Play-With-Docker dengan melibatkan 20 mahasiswa sebagai partisipan praktikum yang secara aktif menjalankan dan mengamati perilaku container selama proses pengujian. Seluruh mahasiswa menggunakan perangkat laptop pribadi dengan spesifikasi yang berbeda-beda, baik dari segi kapasitas RAM, jenis dan generasi prosesor, maupun sistem operasi yang digunakan, sehingga menciptakan kondisi pengujian yang heterogen. Variasi spesifikasi perangkat tersebut mencerminkan kondisi riil lingkungan pembelajaran berbasis teknologi container di perguruan tinggi, di mana pengguna umumnya tidak memiliki standar perangkat keras yang seragam. Oleh karena itu, hasil pengujian yang diperoleh diharapkan mampu memberikan gambaran yang lebih representatif dan realistis terhadap penggunaan NGINX dalam konteks pendidikan, khususnya pada kegiatan praktikum dan simulasi berbasis container. Dalam penelitian ini, setiap mahasiswa menjalankan dua skenario pengujian yang telah ditentukan, yaitu menjalankan container NGINX tanpa pembatasan sumber daya serta menjalankan container NGINX dengan pembatasan sumber daya berupa memory sebesar 64 MB dan CPU sebesar 0.2 core, dengan menggunakan konfigurasi image NGINX yang sama pada kedua skenario tersebut. Penggunaan konfigurasi yang identik dimaksudkan untuk memastikan bahwa perbedaan hasil yang diperoleh benar-benar disebabkan oleh kebijakan pembatasan sumber daya, bukan oleh perbedaan konfigurasi aplikasi.

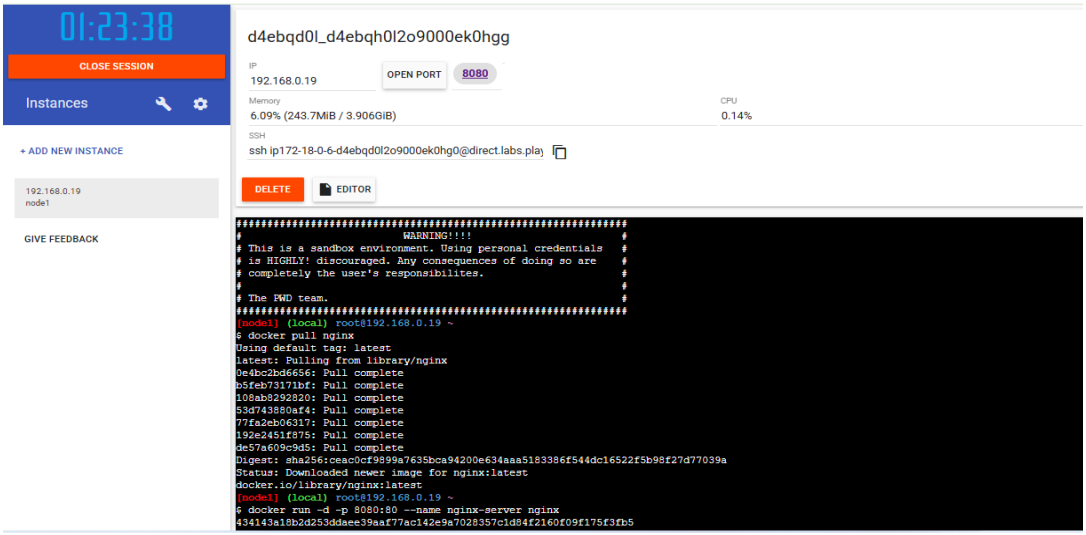
Hasil pengujian menunjukkan bahwa pada skenario tanpa pembatasan sumber daya, NGINX mampu berjalan dengan sangat stabil dan responsif pada seluruh percobaan yang dilakukan oleh mahasiswa. Proses deployment container berlangsung cepat, layanan web dapat diakses tanpa kendala yang berarti, serta waktu respons relatif singkat baik saat diakses melalui peramban web maupun melalui pengujian permintaan HTTP sederhana secara berulang. Penggunaan sumber daya pada kondisi ini bersifat dinamis, di mana container dapat memanfaatkan memori dan CPU sesuai kebutuhan aplikasi tanpa mengalami pembatasan yang signifikan, sehingga mendukung kinerja layanan yang optimal. Meskipun mahasiswa menggunakan laptop dengan spesifikasi yang berbeda-beda, perbedaan performa yang dirasakan relatif kecil dan tidak memengaruhi keberhasilan pengujian secara keseluruhan. Temuan ini menunjukkan bahwa teknologi container mampu mengisolasi aplikasi dari variasi lingkungan host secara efektif, selama sumber daya yang tersedia tidak dibatasi secara ketat. Namun demikian, pada beberapa perangkat dengan spesifikasi yang lebih rendah, terutama laptop dengan kapasitas RAM terbatas, penggunaan container tanpa limit berpotensi meningkatkan beban sistem host, khususnya ketika mahasiswa menjalankan beberapa container secara bersamaan atau melakukan aktivitas praktikum lain secara paralel. Kondisi ini mengindikasikan bahwa meskipun container dikenal ringan, ketersediaan sumber daya pada perangkat host tetap menjadi faktor yang perlu diperhatikan dalam perencanaan dan pelaksanaan praktikum berbasis container.

Pada skenario dengan pembatasan memory sebesar 64 MB dan CPU sebesar 0.2 core, terjadi penurunan kinerja NGINX yang cukup terlihat dibandingkan skenario tanpa limit. Meskipun container masih dapat dijalankan dan mampu melayani permintaan HTTP dasar, waktu startup container cenderung lebih lambat dan respons server terasa menurun ketika dilakukan akses berulang. Pembatasan memory menyebabkan NGINX bekerja pada kapasitas minimum, sehingga pada beberapa percobaan, terutama ketika mahasiswa melakukan perubahan konfigurasi atau menjalankan proses tambahan di dalam container, muncul kondisi tekanan memori yang berujung pada throttling bahkan penghentian container secara otomatis. Pembatasan CPU juga berdampak pada kemampuan NGINX dalam menangani permintaan secara simultan, di mana peningkatan latensi dan penurunan throughput mulai terlihat ketika dilakukan simulasi akses secara berulang atau bersamaan.

Perbandingan antara kedua skenario menunjukkan bahwa kebijakan pembatasan sumber daya memiliki pengaruh langsung terhadap kinerja dan stabilitas NGINX container. Pada kondisi tanpa limit, NGINX mampu memberikan performa yang konsisten dan relatif stabil, sedangkan pada kondisi dengan limit memory dan CPU, performa server menjadi lebih terbatas dan sensitif terhadap peningkatan beban kerja. Meskipun demikian, penggunaan limit resource tetap memberikan manfaat dari sisi efisiensi dan pembelajaran, karena mahasiswa dapat memahami secara langsung dampak pengelolaan sumber daya terhadap performa aplikasi berbasis container. Hal ini relevan dengan praktik penerapan container pada lingkungan produksi modern yang menuntut efisiensi dan optimalisasi penggunaan sumber daya.

Variasi spesifikasi laptop mahasiswa tidak menunjukkan pengaruh yang signifikan terhadap performa container pada skenario tanpa pembatasan sumber daya, yang menegaskan bahwa Play-With-Docker mampu menyediakan lingkungan simulasi yang relatif seragam. Namun, pada skenario dengan pembatasan sumber daya, perangkat dengan spesifikasi lebih rendah cenderung mengalami penurunan performa sistem host lebih cepat, terutama ketika menjalankan lebih dari satu container secara bersamaan. Temuan ini menunjukkan bahwa meskipun container bersifat ringan, keterbatasan perangkat keras host tetap menjadi faktor yang perlu diperhatikan dalam kegiatan praktikum berbasis container.

Secara keseluruhan, hasil dan analisis penelitian ini menunjukkan bahwa NGINX sangat layak digunakan sebagai web server berbasis container dalam lingkungan simulasi pendidikan. Penggunaan tanpa pembatasan sumber daya lebih sesuai untuk demonstrasi performa maksimal dan stabilitas layanan, sedangkan penggunaan dengan pembatasan memory dan CPU efektif untuk menanamkan pemahaman mengenai manajemen sumber daya container. Dengan demikian, simulasi menggunakan Play-With-Docker tidak hanya mendukung kegiatan praktikum secara teknis, tetapi juga memberikan wawasan konseptual yang penting mengenai pengelolaan dan optimasi sumber daya dalam arsitektur containerized.

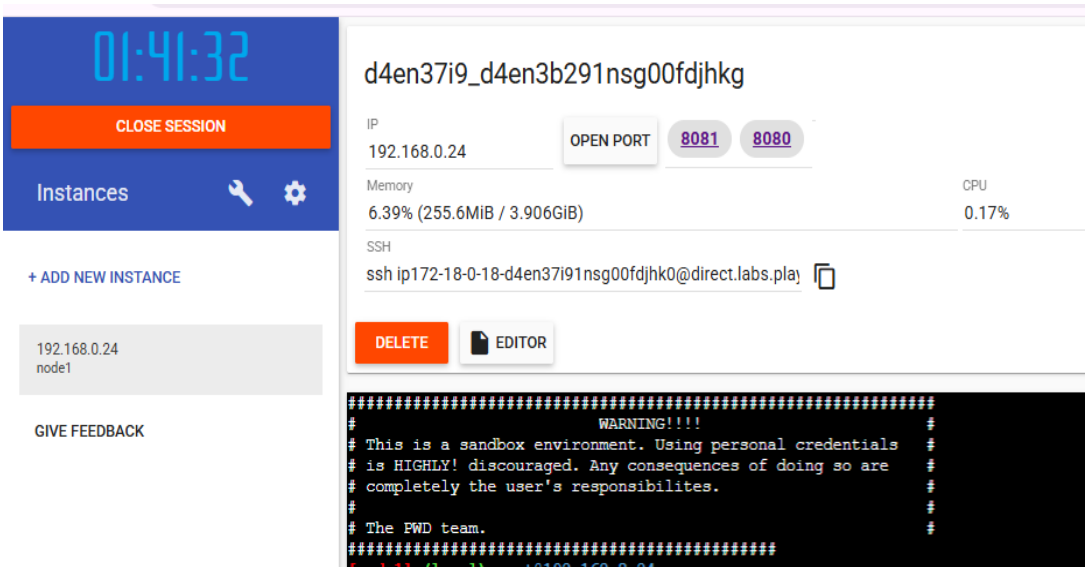


Gambar 2. Menjalankan Nginx Tanpa Limit

Gambar 2 memperlihatkan proses menjalankan container NGINX pada lingkungan simulasi Play-With-Docker tanpa penerapan pembatasan sumber daya (resource limit). Pada tampilan terminal terlihat bahwa image NGINX berhasil di-pull dari repositori Docker dan dijalankan menggunakan perintah Docker standar tanpa menyertakan parameter pembatasan memory maupun CPU. Kondisi ini memungkinkan container NGINX untuk memanfaatkan sumber daya sistem secara penuh dan fleksibel sesuai dengan kebutuhan operasional aplikasi web server yang dijalankan. Antarmuka Play-With-Docker juga menampilkan informasi penggunaan sumber daya secara real-time, di mana konsumsi memory dan CPU masih berada pada tingkat rendah pada saat awal eksekusi container, yang mengindikasikan bahwa NGINX memiliki karakteristik ringan dan efisien pada kondisi beban awal.

Selain itu, pada gambar ditunjukkan bahwa layanan NGINX berhasil diakses melalui port yang telah dibuka, yang menandakan bahwa web server dapat beroperasi secara normal tanpa mengalami kegagalan layanan atau kesalahan konfigurasi. Keberhasilan pembukaan port dan tidak ditemukannya pesan kesalahan pada proses deployment menunjukkan bahwa lingkungan simulasi Play-With-Docker mampu menyediakan platform yang stabil untuk menjalankan container tanpa pembatasan sumber daya. Kondisi ini juga memberikan gambaran bahwa pada skenario tanpa limit, NGINX memiliki kemampuan untuk menangani permintaan layanan web secara responsif, terutama pada tahap awal pengujian dan akses tunggal.

Gambar ini digunakan sebagai representasi visual dari skenario pengujian tanpa pembatasan resource dalam penelitian ini, yang selanjutnya dijadikan sebagai acuan pembanding terhadap skenario pengujian dengan pembatasan memory dan CPU. Dengan adanya perbandingan tersebut, pengaruh kebijakan pembatasan sumber daya terhadap kinerja, stabilitas, dan respons layanan NGINX dapat dianalisis secara lebih komprehensif. Oleh karena itu, Gambar 2 tidak hanya berfungsi sebagai dokumentasi proses eksperimen, tetapi juga sebagai bukti empiris awal bahwa NGINX container dapat berjalan secara stabil dan efisien pada lingkungan simulasi Docker ketika tidak dikenakan batasan sumber daya.



Gambar 3. Menjalankan Nginx Dengan Limit Memory + CPU

.

Gambar 3 menunjukkan skenario pengujian kedua, yaitu menjalankan container NGINX dengan penerapan pembatasan sumber daya berupa limit memory dan CPU pada lingkungan simulasi Play-With-Docker. Pada gambar tersebut ditampilkan tahapan eksekusi yang dilakukan, dimulai dari perintah untuk menjalankan container NGINX dengan parameter --memory=64m dan --cpus=0.2, yang secara eksplisit membatasi alokasi memori maksimum sebesar 64 MB serta penggunaan CPU hingga 0.2 core. Penerapan parameter ini bertujuan untuk mensimulasikan kondisi keterbatasan sumber daya yang umum dijumpai pada lingkungan komputasi berbasis container, khususnya pada arsitektur microservices dan sistem dengan efisiensi resource tinggi. Selanjutnya, pada langkah pengecekan status container menggunakan perintah docker ps, terlihat bahwa container NGINX berhasil dijalankan meskipun berada dalam kondisi sumber daya yang dibatasi. Hal ini menunjukkan bahwa NGINX masih mampu beroperasi secara fungsional pada alokasi resource minimum. Akses layanan melalui browser pada port yang telah ditentukan juga berhasil dilakukan, yang menandakan bahwa web server tetap dapat melayani permintaan HTTP dasar meskipun berada pada kondisi keterbatasan memory dan CPU. Namun, dibandingkan dengan skenario tanpa limit, respons layanan pada kondisi ini cenderung lebih lambat, terutama ketika dilakukan akses secara berulang.

Pada tahap akhir, pemantauan penggunaan sumber daya dilakukan menggunakan perintah docker stats, sebagaimana ditunjukkan pada gambar. Hasil pemantauan memperlihatkan bahwa penggunaan memory berada mendekati batas maksimum yang ditetapkan, sementara penggunaan CPU cenderung mencapai nilai tinggi relatif terhadap batas 0.2 core. Kondisi ini mengindikasikan adanya tekanan sumber daya (resource pressure) yang berpotensi memengaruhi stabilitas dan kinerja container. Oleh karena itu, Gambar 3 merepresentasikan secara visual dampak penerapan pembatasan memory dan CPU terhadap operasional NGINX container, sekaligus menjadi dasar analisis perbandingan dengan skenario tanpa pembatasan sumber daya pada penelitian ini.

Untuk memberikan gambaran yang lebih terstruktur dan komprehensif mengenai perbedaan kinerja NGINX container pada kedua skenario pengujian, yaitu tanpa pembatasan sumber daya dan dengan pembatasan memory serta CPU, hasil simulasi yang diperoleh dirangkum dalam bentuk tabel. Tabel ringkasan hasil analisis ini menyajikan perbandingan parameter kinerja utama yang diamati selama pengujian, sehingga memudahkan pembaca dalam memahami dampak penerapan kebijakan pembatasan sumber daya terhadap performa, stabilitas, dan efisiensi penggunaan resource pada lingkungan simulasi Play-With-Docker.

Tabel 1. Ringkasan Hasil Analisis Kinerja NGINX Container

Parameter Kinerja	Tanpa Limit Resource	Limit Memory 64MB & CPU 0.2
Jumlah Pengguna Uji	20 mahasiswa	20 mahasiswa
Waktu Startup Container (ms)	± 300–500 ms	± 800–1200 ms
Waktu Respons HTTP Rata-rata (ms)	± 40–70 ms	± 150–300 ms
Penggunaan Memory Rata-rata (MB)	± 90–130 MB	± 55–64 MB
Penggunaan CPU Rata-rata (%)	± 10–25 %	± 80–100 % (dibatasi)
Respons Akses Berulang	Stabil	Latensi meningkat
Akses Simultan (request berturut-turut)	Lancar	Terjadi antrian
Throttling CPU	Tidak ada	Terjadi
OOM / Container Restart	0 %	± 10–15 % pengujian
Stabilitas Container	Tinggi	Sedang
Pengaruh Spesifikasi Laptop	Rendah	Sedang
Kesesuaian Praktikum	Sangat baik	Baik (edukatif)

Berdasarkan hasil ringkasan yang disajikan pada Tabel 1, terlihat perbedaan kinerja yang cukup signifikan antara NGINX container yang dijalankan tanpa pembatasan sumber daya dan yang dijalankan dengan pembatasan memory 64 MB serta CPU 0.2 core. Pada skenario tanpa limit, waktu startup container berada pada kisaran 300–500 ms dengan waktu respons HTTP rata-rata sebesar 40–70 ms, yang menunjukkan bahwa NGINX mampu memberikan layanan web yang cepat dan stabil ketika memiliki akses sumber daya yang lebih fleksibel. Sebaliknya, pada skenario dengan pembatasan resource, waktu startup meningkat hingga 800–1200 ms dan waktu respons HTTP rata-rata naik menjadi 150–300 ms, yang mengindikasikan adanya penurunan performa akibat keterbatasan alokasi memory dan CPU.

Dari sisi penggunaan memory, Tabel 1 menunjukkan bahwa container tanpa limit menggunakan memory rata-rata sebesar 90–130 MB, sedangkan container dengan limit berada pada kisaran 55–64 MB, mendekati batas maksimum yang ditetapkan. Kondisi ini menunjukkan bahwa pembatasan memory berhasil meningkatkan efisiensi penggunaan sumber daya, namun sekaligus menempatkan container pada kondisi tekanan memori yang lebih tinggi. Hal tersebut tercermin dari munculnya kejadian throttling CPU serta container restart akibat out-of-memory pada sekitar 10–15% pengujian dengan limit resource, sementara pada skenario tanpa limit tidak ditemukan kejadian serupa.

Selain itu, berdasarkan Tabel 1, penggunaan CPU pada container tanpa limit relatif rendah, yaitu sekitar 10–25%, yang memungkinkan NGINX menangani permintaan akses secara berulang maupun simultan dengan lancar. Sebaliknya, pada skenario dengan pembatasan CPU 0.2 core, penggunaan CPU sering berada pada kisaran 80–100% dari batas yang ditentukan, sehingga menyebabkan terjadinya antrian permintaan dan peningkatan latensi ketika dilakukan simulasi akses berturut-turut. Dampak pembatasan ini juga terlihat dari tingkat stabilitas container yang menurun dari kategori tinggi menjadi sedang, terutama pada perangkat dengan spesifikasi laptop yang lebih rendah.

Secara keseluruhan, analisis berdasarkan Tabel 1 menegaskan bahwa pembatasan memory dan CPU pada NGINX container berpengaruh langsung terhadap waktu respons, stabilitas, dan kemampuan menangani beban akses. Meskipun performa menurun, skenario dengan limit resource tetap memberikan nilai edukatif yang tinggi karena mampu memperlihatkan secara nyata trade-off antara efisiensi penggunaan sumber daya dan kualitas layanan. Oleh karena itu, hasil ini mendukung penggunaan kedua skenario secara komplementer dalam kegiatan praktikum, di mana konfigurasi tanpa limit digunakan untuk menunjukkan performa optimal, sementara konfigurasi dengan limit digunakan untuk memahami konsep manajemen dan optimalisasi sumber daya pada lingkungan containerized.

4 KESIMPULAN

Berdasarkan tujuan penelitian yang telah diuraikan pada bab Pendahuluan, penelitian ini bertujuan untuk menganalisis kinerja NGINX sebagai image container pada lingkungan simulasi Docker menggunakan platform Play-With-Docker, baik pada kondisi tanpa pembatasan sumber daya maupun dengan pembatasan memory dan CPU. Hasil penelitian yang dipaparkan pada bab Hasil dan Analisis menunjukkan bahwa tujuan tersebut telah tercapai, di mana perbedaan kinerja NGINX container pada kedua skenario dapat diidentifikasi dan dianalisis secara jelas. Dengan demikian, terdapat kesesuaian antara permasalahan yang dirumuskan, tujuan penelitian, serta hasil yang diperoleh.Hasil analisis menunjukkan bahwa NGINX container tanpa pembatasan sumber daya mampu memberikan kinerja yang lebih optimal, ditunjukkan oleh waktu startup dan waktu respons HTTP yang lebih cepat serta tingkat stabilitas container yang tinggi. Sebaliknya, penerapan pembatasan memory sebesar 64 MB dan CPU sebesar 0.2 core terbukti memberikan dampak signifikan terhadap penurunan performa, berupa peningkatan latensi, keterbatasan dalam menangani akses simultan, serta munculnya kondisi throttling dan out-of-memory pada sebagian pengujian. Meskipun demikian, pembatasan sumber daya tersebut juga menunjukkan peningkatan efisiensi penggunaan resource dan memberikan gambaran nyata mengenai trade-off antara performa dan efisiensi dalam lingkungan containerized. Selain menjawab tujuan utama penelitian, hasil ini juga memberikan implikasi praktis dalam konteks pembelajaran. Simulasi menggunakan Play-With-Docker terbukti efektif sebagai media praktikum untuk memperkenalkan konsep containerisasi, kinerja web server, dan manajemen sumber daya kepada mahasiswa, meskipun digunakan pada perangkat dengan spesifikasi yang beragam. Hal ini menunjukkan bahwa platform tersebut layak diterapkan sebagai sarana pembelajaran berbasis praktik di lingkungan pendidikan tinggi.Sebagai prospek pengembangan, penelitian selanjutnya dapat memperluas skenario pengujian dengan variasi batasan sumber daya yang lebih beragam, jumlah pengguna yang lebih besar, serta penggunaan alat benchmarking yang lebih komprehensif. Selain itu, penelitian lanjutan juga dapat mengkaji perbandingan kinerja NGINX dengan web server lain atau menerapkan simulasi pada lingkungan orkestrasi container seperti Kubernetes. Dengan pengembangan tersebut, hasil penelitian diharapkan dapat memberikan kontribusi yang lebih luas terhadap pengembangan dan penerapan teknologi container di bidang pendidikan maupun industry.

DAFTAR PUSTAKA

[1] A. M. Potdar, D. G. Narayan, S. Kengond, and M. M. Mulla, “Performance Evaluation of Docker Container and Virtual Machine,” *Procedia Comput. Sci.*, vol. 171, no. 2019, pp. 1419–1428, 2020, doi: 10.1016/j.procs.2020.04.152.

[2] M. R. Alamsyah, H. I. Wahanani, and M. Idhom, “Infrastruktur Private Cloud Storage Berbasis,” *J. Inform. dan Sist. Inf.*, vol. 2, no. 2, pp. 122–131, 2021.

[3] P. L. Docker, “1* , 2 1,2,” pp. 1063–1072, 2025.

[4] A. R. Ekaputra and A. S. Affandi, “Pemanfaatan layanan cloud computing dan docker container untuk meningkatkan kinerja aplikasi web,” *J. Inf. Syst. Appl. Dev.*, vol. 1, no. 2, pp. 138–147, 2023, doi: 10.26905/jisad.v1i2.11084.

[5] R. A. Megantara, F. Alzami, R. Anggi, and D. Puji, “Memaksimalkan Utilitas Server Universitas Pada Masa Covid-19,” no. April, pp. 48–54, 2022.

[6] S. Alam, S. Lestari, A. K. Fauziyyah, and D. Dzulfikar, “Implementasi Docker Container untuk Sistem Monitoring dan Pengontrolan Peralatan Listrik di Laboratorium Cerdas,” *JELIKU J. Elektron. Ilmu Komput. Udayana*, vol. 12, no. 3, p.

697, 2024, [Online]. Available: https://ojs.unud.ac.id/index.php/jlk/article/view/109156

[7] S. Kwon and J. H. Lee, "DIVDS: Docker Image Vulnerability Diagnostic System," *IEEE Access*, vol. 8, pp. 42666–42673, 2020, doi: 10.1109/ACCESS.2020.2976874.

[8] V. Jain, B. Singh, M. Khenwar, and M. Sharma, "Static Vulnerability Analysis of Docker Images," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1131, no. 1, p. 012018, 2021, doi: 10.1088/1757-899x/1131/1/012018.

[9] Y. Cahyaningrum and I. R. Widiasari, "Analisis Performa Container Berplatform Docker atas SeranganMalicious Software (Malware)," *J. Buana Inform.*, vol. 11, no. 1, pp. 47–54, 2020, doi: 10.24002/jbi.v11i1.3279.

[10] H. E. Wahanani and M. Idhom, "Analisis Performansi Container Docker, LXC dan LXD Pada Web Server, FTP Server dan Mail Server," *Pros. Semin. Nas. Inform. Bela Negara*, vol. 1, pp. 45–49, 2020, doi: 10.33005/santika.v1i0.13.

[11] W. Elmenreich, P. Moll, S. Theuermann, and M. Lux, "Making simulation results reproducible-Survey, guidelines, and examples based on gradle and docker," *PeerJ Comput. Sci.*, vol. 2019, no. 12, pp. 1–27, 2019, doi: 10.7717/peerj-cs.240.

[12] N. Ramsari and A. Ginanjar, "Implementasi Infrastruktur Server Berbasis Cloud Computing Untuk Web Service Berbasis Teknologi Google Cloud Platform," *Conf. Senat. STT Adisutjipto Yogyakarta*, vol. 7, 2022, doi: 10.28989/senatik.v7i0.472.

[13] C. Boettiger, "An introduction to Docker for reproducible research, with examples from the R environment," 2014, doi: 10.1145/2723872.2723882.

[14] F. O. I. Nvestigations and S. Wilford, "Precept : a F Ramework for E Thical D Igital," pp. 1–18, 2025.

[15] A. Anand and A. Nisha Jebaseeli, "Performance Comparison between VM Based Webserver and Docker Container Webserver," *Asian J. Comput. Sci. Technol.*, vol. 8, no. S2, pp. 28–30, 2019, doi: 10.51983/ajcst-2019.8.s2.2030.