Jurnal RISTIKA – *Jurnal Riset Teknologi Informasi, Manajemen, dan Multimedia*
Ristika., 1(2) 64-72
Februari 2026

Analisis Kinerja NGINX sebagai Image Container pada Lingkungan Simulasi Docker Play-With-Docker dengan Pembatasan Memory dan CPU

Andika Sastra Praja[1]*, Muhammad Syawalludin[2], Faridatul Munawaroh[3]

[1,2,3]Manajemen Informatika, Institut Teknologi Dan Bisnis Bina Sriwijaya, Kota Palembang, Indonesia
[1]andika789099@gmail.com, [2]ulusyawall@gmail.com, [3]kha.dijah6731@gmail.com

Received : 16 Desember 2025
Received in revised from: 31 Januari 2026
Accepted : 6 Februari 2026

Abstract – The rapid adoption of containerization technologies has increased the use of lightweight web servers in both production and educational environments. NGINX is widely recognized for its high performance and efficient resource utilization, making it suitable for container-based deployment. However, resource constraints applied to containers may significantly affect service performance and stability. This study aims to analyze the performance of NGINX as an image container in a Docker simulation environment using the Play-With-Docker platform, focusing on the impact of memory and CPU limitations. An experimental quantitative approach was employed by comparing two testing scenarios: NGINX containers without resource limitations and containers with constrained resources, specifically 64 MB of memory and 0.2 CPU cores. The experiments were conducted in a simulated learning environment involving 20 participants using heterogeneous devices. Performance evaluation parameters included container startup time, HTTP response time, memory usage, CPU utilization, and container stability. The results indicate that NGINX containers without resource constraints achieve faster startup times, lower HTTP response latency, and higher service stability. In contrast, containers with memory and CPU limitations experience increased startup time, higher response latency, CPU throttling, and occasional out-of-memory events, although resource usage becomes more efficient. These findings demonstrate a clear trade-off between performance and resource efficiency. This study concludes that Play-With-Docker is an effective platform for simulating container-based web server performance and provides significant educational value in understanding resource management and optimization in containerized environments.

Keywords: NGINX, Docker, Play-With-Docker, Resource Limitation, Container Performance.

Pendahuluan

Perkembangan teknologi informasi dan komunikasi dalam dua dekade terakhir telah mengalami akselerasi yang sangat signifikan, seiring dengan meningkatnya kebutuhan masyarakat terhadap layanan digital yang cepat, andal, dan selalu tersedia. Hampir seluruh sektor kehidupan modern, mulai dari pemerintahan, pendidikan, industri, hingga layanan publik, kini bergantung pada sistem berbasis web sebagai tulang punggung operasional dan penyediaan informasi. Ketergantungan ini menuntut infrastruktur server yang tidak hanya mampu menangani permintaan dalam jumlah besar, tetapi juga memiliki efisiensi tinggi dalam pemanfaatan sumber daya komputasi (Potdar et al., 2020). Web server sebagai komponen inti dalam arsitektur sistem informasi berbasis web memegang peranan strategis dalam menjamin kualitas layanan, kecepatan akses, serta stabilitas sistem secara keseluruhan.

Seiring dengan meningkatnya kompleksitas aplikasi dan jumlah pengguna, tantangan dalam pengelolaan infrastruktur server juga semakin besar. Infrastruktur konvensional berbasis mesin fisik maupun virtual machine sering kali menghadapi keterbatasan dalam hal skalabilitas, efisiensi penggunaan sumber daya, serta fleksibilitas pengelolaan (Potdar et al., 2020; Anand and Jebaseeli, 2019). Kondisi ini mendorong munculnya paradigma baru dalam pengembangan dan pengelolaan sistem, yaitu pemanfaatan teknologi virtualisasi ringan melalui *container*. Teknologi *container* memungkinkan aplikasi dijalankan dalam lingkungan terisolasi dengan *overhead* yang jauh lebih rendah dibandingkan *virtual machine*, sehingga lebih efisien dalam penggunaan memori dan CPU (Boettiger, 2015).

Docker merupakan salah satu *platform containerization* yang paling banyak diadopsi secara global, baik dalam lingkungan pengembangan, pengujian, maupun produksi. Docker memungkinkan pengembang dan administrator sistem untuk mengemas aplikasi beserta seluruh dependensinya ke dalam sebuah image yang portabel dan konsisten di berbagai lingkungan komputasi (Boettiger, 2015). Berbagai penelitian menunjukkan bahwa container berbasis Docker memiliki performa yang lebih baik dan konsumsi sumber daya yang lebih

efisien dibandingkan dengan pendekatan virtualisasi tradisional, khususnya dalam konteks layanan web dan aplikasi server (Potdar et al., 2020; Anand and Jebaseeli, 2019). Oleh karena itu, Docker menjadi teknologi yang sangat relevan untuk dikaji, baik dari sisi performa teknis maupun dari sisi penerapannya dalam lingkungan nyata.

Dalam konteks layanan web, web server merupakan komponen utama yang menentukan kualitas dan keandalan sistem. Salah satu web server yang paling populer dan banyak digunakan saat ini adalah NGINX. NGINX dikenal memiliki arsitektur event-driven dan non-*blocking*, yang memungkinkan server menangani ribuan koneksi secara simultan dengan penggunaan sumber daya yang relatif rendah. Karakteristik ini menjadikan NGINX sangat sesuai untuk dijalankan dalam lingkungan *containerized* dan *cloud computing* (Ekaputra and Affandi, 2023; Wahanani and Idhom, 2020). Dengan karakteristik tersebut, NGINX banyak diadopsi sebagai web server, reverse proxy, maupun load balancer dalam berbagai skala sistem.

Memasuki era Society 5.0, paradigma pengembangan teknologi tidak lagi hanya berfokus pada digitalisasi, tetapi juga pada optimalisasi sistem yang berorientasi pada kebutuhan manusia, keberlanjutan, dan efisiensi sumber daya (Megantara et al., 2022). Dalam konteks ini, infrastruktur teknologi dituntut untuk mampu memberikan layanan yang optimal meskipun dijalankan pada sumber daya yang terbatas. Teknologi *container* hadir sebagai solusi yang mendukung efisiensi tersebut dengan menyediakan lingkungan aplikasi yang ringan, fleksibel, dan mudah dikelola (Alam et al., 2024).

Meskipun Docker menawarkan berbagai keunggulan, pengelolaan sumber daya seperti memori dan CPU tetap menjadi aspek yang krusial, terutama pada lingkungan multi-tenant dan sistem berbasis cloud. Dalam praktik nyata, container jarang dijalankan tanpa pembatasan sumber daya. Penerapan limit memory dan CPU bertujuan untuk mencegah satu container menggunakan sumber daya secara berlebihan dan mengganggu container lainnya (Kwon and Lee, 2020). Namun demikian, pembatasan sumber daya juga berpotensi menurunkan performa aplikasi yang berjalan di dalam container, khususnya layanan web yang membutuhkan respons cepat dan stabil (Cahyaningrum and Widiasari, 2020).

Berbagai penelitian terdahulu telah mengkaji performa container Docker dalam beragam skenario. Potdar et al. (2020) membandingkan performa Docker container dengan virtual machine dan menemukan bahwa container memiliki keunggulan signifikan dalam hal efisiensi dan kecepatan. Wahanani and Idhom (2020) menganalisis performansi Docker, LXC, dan LXD pada web server serta layanan jaringan lainnya, dan menunjukkan bahwa container mampu memberikan performa yang kompetitif. Anand and Jebaseeli (2019) juga membuktikan bahwa web server berbasis Docker memiliki latency yang lebih rendah dibandingkan dengan web server berbasis virtual machine. Namun, sebagian besar penelitian tersebut dilakukan pada kondisi sumber daya yang relatif longgar.

Di sisi lain, isu keamanan dan stabilitas container juga menjadi perhatian penting. Penelitian oleh Kwon and Lee (2020) serta Jain et al. (2021) menyoroti potensi kerentanan pada Docker image dan pentingnya manajemen container yang tepat. Selain itu, Cahyaningrum and Widiasari (2020) menunjukkan bahwa performa container dapat terpengaruh oleh kondisi eksternal seperti serangan malware. Elmenreich et al. (2019) menekankan pentingnya reproducibility dalam simulasi berbasis Docker, terutama untuk memastikan bahwa hasil pengujian dapat direplikasi dan digunakan sebagai referensi yang valid.

Dalam konteks pendidikan tinggi, Docker semakin banyak dimanfaatkan sebagai media pembelajaran dan praktikum. Alam et al. (2024) menunjukkan bahwa Docker efektif digunakan dalam sistem monitoring dan pengontrolan di laboratorium cerdas. Ramsari and Ginanjar (2022) menegaskan bahwa integrasi cloud computing dan container mendukung fleksibilitas serta efisiensi layanan web. Penggunaan Docker dalam lingkungan pendidikan memungkinkan mahasiswa memahami konsep infrastruktur modern secara langsung melalui praktik, tanpa memerlukan perangkat keras dengan spesifikasi tinggi.

Salah satu platform yang banyak digunakan untuk tujuan pembelajaran dan simulasi adalah *Play-With-Docker* (PWD). Platform ini menyediakan lingkungan Docker berbasis cloud yang dapat diakses secara instan melalui browser, sehingga sangat sesuai digunakan dalam kegiatan praktikum. Namun demikian, kajian empiris yang secara khusus menganalisis kinerja aplikasi berbasis container pada platform PWD, terutama dengan pembatasan sumber daya yang ketat, masih relatif terbatas.

Berdasarkan kajian literatur tersebut, dapat diidentifikasi adanya celah penelitian terkait analisis kinerja NGINX sebagai image container pada kondisi pembatasan memory dan CPU yang ekstrem dalam lingkungan simulasi *Play-With-Docker*. Sebagian besar penelitian sebelumnya lebih menitikberatkan pada perbandingan teknologi atau pengujian pada lingkungan produksi dengan sumber daya yang relatif memadai. Selain itu,

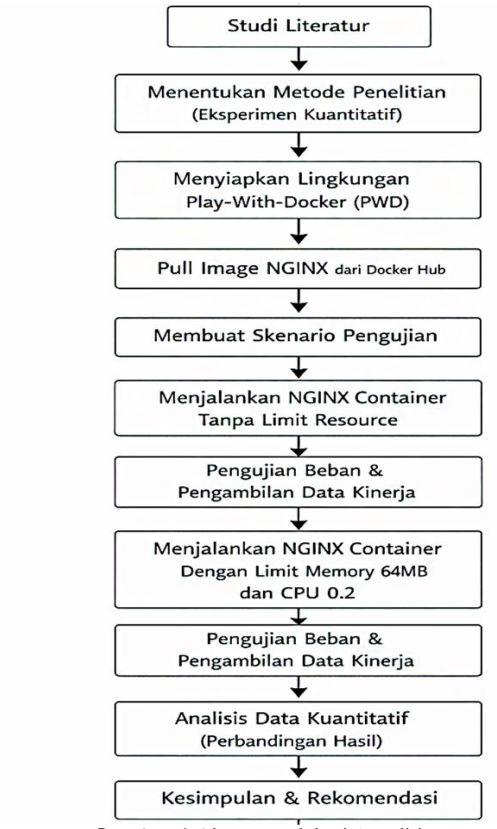
penelitian dengan konteks pendidikan yang melibatkan perangkat pengguna yang heterogen masih jarang dilakukan.

Oleh karena itu, penelitian ini bertujuan untuk menganalisis kinerja NGINX sebagai *image container* pada lingkungan simulasi Docker menggunakan platform *Play-With-Docker* dengan dua skenario, yaitu tanpa pembatasan sumber daya dan dengan pembatasan memory sebesar 64 MB serta CPU sebesar 0.2 core. Analisis dilakukan terhadap parameter kinerja utama seperti waktu *startup container*, waktu respons HTTP, penggunaan memori, penggunaan CPU, dan stabilitas container. Hasil penelitian ini diharapkan dapat memberikan kontribusi ilmiah sebagai referensi empiris serta kontribusi praktis dalam perancangan pembelajaran dan pengelolaan *container* berbasis NGINX secara efisien dan realistis.

Metode

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 peneliti untuk mengamati secara sistematis pengaruh perlakuan tertentu terhadap objek penelitian dalam kondisi yang relatif terkendali dan dapat direplikasi (Potdar et al., 2020; Elmenreich et al., 2019). Melalui pendekatan ini, perbedaan kinerja yang muncul akibat variasi konfigurasi sumber daya dapat dianalisis secara lebih akurat dan komprehensif.

Penelitian ini dilaksanakan pada lingkungan simulasi berbasis cloud menggunakan platform *Play-With-Docker* (PWD), yang menyediakan lingkungan Docker instan tanpa memerlukan instalasi lokal. Pemilihan PWD sebagai media eksperimen didasarkan pada kemampuannya dalam menyediakan lingkungan *container* yang konsisten dan mudah diakses, sehingga mendukung pelaksanaan simulasi dan praktikum secara efektif, khususnya dalam konteks pendidikan tinggi (Ramsari and Ginanjar, 2022; Alam et al., 2024). Selain itu, penggunaan PWD juga mendukung prinsip reproducibility dalam penelitian berbasis container, karena konfigurasi dan skenario pengujian dapat dijalankan kembali dengan kondisi yang serupa (Elmenreich et al., 2019).



Gambar 1 Alur Metodelogi Penelitian

Rancangan penelitian ini menggunakan dua skenario perlakuan utama yang diterapkan pada *container* NGINX. Skenario pertama adalah menjalankan NGINX *image container* tanpa pembatasan sumber daya, di mana *container* dapat memanfaatkan memori dan CPU sistem secara dinamis sesuai kebutuhan aplikasi. Skenario kedua adalah menjalankan NGINX *image container* dengan pembatasan sumber daya, yaitu limit memory sebesar 64 MB dan limit CPU sebesar 0.2 core. Penerapan pembatasan ini dilakukan menggunakan parameter bawaan Docker untuk mensimulasikan kondisi keterbatasan sumber daya yang umum dijumpai pada lingkungan *multi-tenant* dan sistem berbasis container modern (Kwon and Lee, 2020).

Image NGINX yang digunakan dalam penelitian ini merupakan image resmi yang diperoleh dari repositori Docker Hub. Penggunaan *image* resmi bertujuan untuk menjamin keseragaman versi perangkat lunak, stabilitas sistem, serta menghindari potensi perbedaan performa akibat variasi konfigurasi atau modifikasi image pihak ketiga (Jain et al., 2021). Dengan demikian, seluruh perbedaan hasil pengujian dapat dikaitkan secara langsung dengan perlakuan pembatasan sumber daya yang diterapkan.

Variabel Penelitian

Dalam penelitian ini, variabel bebas ditetapkan sebagai konfigurasi sumber daya *container*, yang terdiri atas dua kondisi, yaitu tanpa pembatasan sumber daya dan dengan pembatasan memory serta CPU. Variabel terikat adalah kinerja NGINX *container* yang diukur melalui beberapa parameter kuantitatif utama, meliputi waktu startup *container*, waktu respons HTTP, penggunaan memori, penggunaan CPU, serta stabilitas *container* selama pengujian. Sementara itu, variabel kontrol meliputi lingkungan pengujian, versi image NGINX, metode akses layanan web, serta skenario pengujian yang dijaga tetap sama pada setiap percobaan, sehingga hasil yang diperoleh dapat dibandingkan secara objektif (Anand and Jebaseeli, 2019; Wahanani and Idhom, 2020).

Subjek dan Lingkungan Pengujian

Pengujian dilakukan dengan melibatkan 20 mahasiswa yang berperan sebagai partisipan praktikum. Setiap mahasiswa menggunakan perangkat laptop pribadi dengan spesifikasi perangkat keras dan sistem operasi yang berbeda-beda. Variasi spesifikasi perangkat ini mencerminkan kondisi nyata lingkungan pembelajaran di perguruan tinggi, di mana pengguna umumnya tidak memiliki perangkat dengan konfigurasi yang seragam. Pendekatan ini bertujuan untuk memperoleh hasil yang lebih representatif dan relevan dengan konteks penggunaan Docker dan NGINX dalam kegiatan praktikum pendidikan (Megantara et al., 2022).

Meskipun perangkat host yang digunakan bersifat heterogen, lingkungan pengujian *container* tetap diseragamkan melalui penggunaan *platform Play-With-Docker*. Dengan demikian, pengaruh variasi perangkat keras *host* terhadap kinerja *container* dapat diminimalkan, dan fokus analisis tetap diarahkan pada dampak pembatasan sumber daya container itu sendiri (Elmenreich et al., 2019).

Prosedur Pengujian

Tahapan eksperimen diawali dengan persiapan lingkungan *Play-With-Docker*, termasuk pembuatan sesi Docker dan penarikan (*pull*) *image* NGINX dari repositori resmi Docker Hub. Setelah image berhasil diunduh, *container* NGINX dijalankan sesuai dengan skenario pengujian yang telah ditentukan. Pada skenario tanpa pembatasan sumber daya, *container* dijalankan menggunakan perintah Docker standar tanpa parameter limit. Pada skenario dengan pembatasan sumber daya, *container* dijalankan dengan parameter pembatasan *memory* dan CPU yang telah ditetapkan sebelumnya.

Setelah *container* berhasil dijalankan, dilakukan verifikasi layanan untuk memastikan bahwa NGINX dapat beroperasi secara normal dan layanan web dapat diakses melalui port yang telah dibuka. Selanjutnya, pengujian beban dilakukan dengan memberikan permintaan HTTP secara berulang dalam periode waktu tertentu untuk mensimulasikan kondisi penggunaan layanan web yang umum terjadi. Setiap skenario pengujian dijalankan lebih dari satu kali guna memperoleh hasil yang lebih stabil dan konsisten (Potdar et al., 2020).

Teknik Pengumpulan dan Analisis Data

Data kinerja yang dikumpulkan dalam penelitian ini berupa data kuantitatif yang mencakup waktu *startup container*, waktu respons HTTP, penggunaan memori, penggunaan CPU, serta kejadian *throttling* atau penghentian *container* akibat *out-of-memory*. Pemantauan penggunaan sumber daya dilakukan menggunakan perintah dan utilitas

bawaan Docker, seperti *docker stats*, yang memberikan informasi penggunaan *resource* secara *real-time* (Wahanani and Idhom, 2020).

Data yang diperoleh dari seluruh pengujian kemudian dianalisis secara deskriptif kuantitatif dengan membandingkan nilai rata-rata dan kecenderungan kinerja pada kedua skenario perlakuan. Hasil analisis disajikan dalam bentuk tabel dan visualisasi untuk memudahkan interpretasi dan pembahasan. Melalui proses analisis ini, dapat ditarik kesimpulan mengenai sejauh mana pembatasan memory dan CPU memengaruhi kinerja serta stabilitas NGINX *image container* pada lingkungan simulasi *Play-With-Docker*.

Hasil

Pengujian kinerja NGINX sebagai *image container* dilaksanakan pada lingkungan simulasi *Play-With-Docker* dengan melibatkan 20 mahasiswa sebagai partisipan praktikum. Setiap mahasiswa menjalankan *container* NGINX pada dua skenario pengujian, yaitu tanpa pembatasan sumber daya dan dengan pembatasan sumber daya berupa memory sebesar 64 MB serta CPU sebesar 0.2 *core*. Seluruh pengujian dilakukan menggunakan *image* NGINX yang sama untuk memastikan bahwa perbedaan kinerja yang diamati hanya disebabkan oleh konfigurasi sumber daya yang diterapkan.

Gambar 2 Web Play With Docker

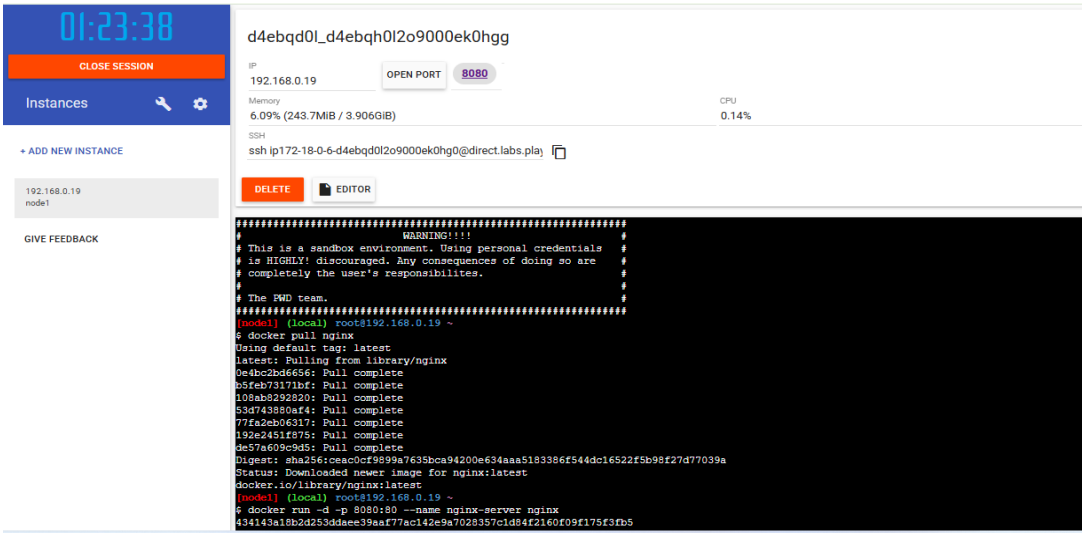
Hasil Pengujian Tanpa Pembatasan Sumber Daya

Pada skenario tanpa pembatasan sumber daya, hasil pengujian menunjukkan bahwa *container* NGINX dapat dijalankan dengan sangat stabil pada seluruh percobaan. Waktu *startup container* berada pada kisaran ±300–500 ms, yang menunjukkan proses inisialisasi *container* berlangsung cepat dan efisien. Setelah *container* aktif, layanan web dapat diakses tanpa kendala yang berarti, baik melalui peramban web maupun melalui pengujian permintaan HTTP secara berulang.

Waktu respons HTTP rata-rata pada skenario ini berada pada kisaran ±40–70 ms. Nilai tersebut menunjukkan bahwa NGINX mampu memberikan respons yang cepat dan konsisten terhadap permintaan klien. Pada pengujian akses berulang maupun akses berturut-turut, tidak ditemukan peningkatan latensi yang signifikan. Hal ini mengindikasikan bahwa NGINX memiliki kemampuan yang baik dalam menangani beban akses ringan hingga sedang ketika tidak dibatasi oleh alokasi sumber daya.

Dari sisi penggunaan sumber daya, container NGINX pada skenario tanpa limit memanfaatkan memory rata-rata sebesar ±90–130 MB, dengan penggunaan CPU berada pada kisaran ±10–25%. Penggunaan sumber daya yang bersifat dinamis ini memungkinkan NGINX menyesuaikan konsumsi *resource* sesuai dengan kebutuhan operasionalnya. Meskipun mahasiswa menggunakan perangkat laptop dengan spesifikasi yang beragam, perbedaan performa yang dirasakan relatif kecil dan tidak memengaruhi keberhasilan pengujian secara keseluruhan.

Stabilitas *container* pada skenario tanpa pembatasan sumber daya tergolong sangat tinggi. Selama proses pengujian, tidak ditemukan kejadian *throttling* CPU maupun penghentian *container* akibat kehabisan memori. Seluruh mahasiswa dapat menyelesaikan pengujian tanpa mengalami kegagalan layanan. Namun demikian, pada beberapa perangkat dengan spesifikasi lebih rendah, penggunaan *container* tanpa limit berpotensi meningkatkan beban sistem host, terutama ketika beberapa *container* dijalankan secara bersamaan. Kondisi ini menunjukkan bahwa meskipun *container* bersifat ringan, ketersediaan sumber daya pada sistem *host* tetap menjadi faktor pendukung yang perlu diperhatikan.



Gambar 3 Menjalankan Nginx Tanpa Limit

Hasil Pengujian Dengan Pembatasan Memory 64 MB dan CPU 0.2 Core

Pada skenario kedua, yaitu dengan pembatasan sumber daya, hasil pengujian menunjukkan adanya penurunan kinerja yang cukup signifikan dibandingkan dengan skenario tanpa limit. Waktu *startup container* meningkat menjadi ±800–1200 ms, yang mengindikasikan bahwa proses inisialisasi *container* membutuhkan waktu lebih lama akibat keterbatasan alokasi memori dan CPU.

Waktu respons HTTP rata-rata pada skenario ini berada pada kisaran ±150–300 ms. Peningkatan latensi ini semakin terasa ketika dilakukan akses secara berulang atau berturut-turut. Pada kondisi tertentu, permintaan HTTP harus menunggu antrian sebelum diproses, sehingga respons layanan menjadi lebih lambat dibandingkan skenario tanpa pembatasan sumber daya.

Penggunaan memory pada skenario dengan limit berada pada kisaran ±55–64 MB, mendekati batas maksimum yang ditetapkan. Kondisi ini menunjukkan bahwa NGINX beroperasi pada kapasitas memori minimum yang tersedia. Penggunaan CPU juga cenderung tinggi, berada pada kisaran ±80–100% dari batas 0.2 core yang diberikan. Akibatnya, terjadi throttling CPU pada sebagian besar pengujian, terutama ketika terjadi peningkatan jumlah permintaan secara bersamaan.

Selain itu, pada sekitar ±10–15% pengujian, ditemukan kejadian penghentian container akibat kehabisan memori (*out-of-memory*). Kejadian ini umumnya terjadi ketika mahasiswa melakukan perubahan konfigurasi atau menjalankan proses tambahan di dalam *container*. Meskipun demikian, secara umum *container* masih dapat dijalankan dan mampu melayani permintaan HTTP dasar selama beban akses tidak terlalu tinggi.

Stabilitas *container* pada skenario dengan pembatasan sumber daya berada pada kategori sedang. *Container* tetap berfungsi, namun lebih sensitif terhadap peningkatan beban kerja dan perubahan kondisi operasional. Variasi spesifikasi laptop mahasiswa juga lebih berpengaruh pada skenario ini, di mana perangkat dengan kapasitas RAM dan performa *prosesor* yang lebih rendah cenderung mengalami penurunan performa sistem *host* lebih cepat, terutama ketika menjalankan lebih dari satu *container* secara bersamaan.

Gambar 4 Menjalankan Nginx Dengan Limit Memory dan CPU

Analisis Perbandingan Kinerja Antar Skenario

Perbandingan hasil pengujian antara kedua skenario menunjukkan bahwa pembatasan sumber daya memiliki pengaruh langsung terhadap kinerja dan stabilitas NGINX container. Pada skenario tanpa pembatasan sumber daya, NGINX mampu memberikan performa yang optimal dengan waktu startup dan waktu respons yang relatif singkat serta stabilitas layanan yang tinggi. Sebaliknya, penerapan pembatasan *memory* dan CPU menyebabkan peningkatan latensi, penurunan throughput, serta meningkatnya risiko terjadinya *throttling* dan penghentian *container*.

Meskipun demikian, pembatasan sumber daya juga memberikan manfaat dari sisi efisiensi penggunaan *resource*. Pada skenario dengan limit, penggunaan *memory* dapat ditekan hingga mendekati batas yang ditetapkan, sehingga lebih hemat dibandingkan skenario tanpa limit. Hal ini menunjukkan adanya *trade-off* yang jelas antara performa dan efisiensi sumber daya pada lingkungan *containerized*.

Dari sudut pandang pembelajaran, hasil pengujian ini memiliki nilai edukatif yang tinggi. Skenario tanpa pembatasan sumber daya sesuai digunakan untuk memperlihatkan performa optimal dan stabilitas NGINX sebagai web server berbasis *container*. Sementara itu, skenario dengan pembatasan sumber daya efektif digunakan untuk menanamkan pemahaman mengenai manajemen dan optimalisasi sumber daya, serta dampak kebijakan *resource* limit terhadap kualitas layanan.

Secara keseluruhan, hasil dan analisis menunjukkan bahwa NGINX sangat layak digunakan sebagai web server berbasis container dalam lingkungan simulasi pendidikan. *Play-With-Docker* terbukti mampu menyediakan lingkungan pengujian yang relatif seragam dan mudah diakses, sehingga mendukung pelaksanaan praktikum berbasis *container*. Dengan demikian, kedua skenario pengujian dapat digunakan secara komplementer untuk mencapai tujuan pembelajaran yang berbeda, baik dari sisi performa teknis maupun dari sisi pemahaman konsep manajemen sumber daya.

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)

Kesimpulan

Penelitian ini bertujuan untuk menganalisis kinerja NGINX sebagai *image container* pada lingkungan simulasi Docker menggunakan *platform Play-With-Docker* dengan membandingkan dua skenario konfigurasi sumber daya, yaitu tanpa pembatasan *resource* dan dengan pembatasan *memory* sebesar 64 MB serta CPU sebesar 0.2 *core*. Berdasarkan hasil pengujian dan analisis yang telah dilakukan, dapat disimpulkan bahwa konfigurasi sumber daya memiliki pengaruh yang signifikan terhadap kinerja dan stabilitas NGINX *container*.

Pada skenario tanpa pembatasan sumber daya, NGINX container menunjukkan kinerja yang optimal dan stabil. Waktu *startup container* relatif cepat, waktu respons HTTP rendah, serta kemampuan menangani akses berulang dan berturut-turut berjalan dengan baik tanpa menimbulkan peningkatan latensi yang berarti. Penggunaan *memory* dan CPU bersifat dinamis sesuai kebutuhan layanan, dan tidak ditemukan kejadian *throttling* maupun penghentian *container* akibat kehabisan sumber daya. Kondisi ini menunjukkan bahwa NGINX sangat andal ketika dijalankan dalam lingkungan *container* dengan ketersediaan *resource* yang fleksibel.

Sebaliknya, pada skenario dengan pembatasan *memory* dan CPU, kinerja NGINX *container* mengalami penurunan yang cukup signifikan. Waktu startup dan waktu respons HTTP meningkat, terutama ketika terjadi akses berulang atau peningkatan beban kerja. Pembatasan sumber daya menyebabkan penggunaan *memory* dan CPU berada mendekati batas maksimum, sehingga memicu terjadinya throttling CPU dan pada sebagian pengujian mengakibatkan penghentian *container* akibat kondisi *out-of-memory*. Meskipun demikian, NGINX tetap dapat beroperasi dan melayani permintaan HTTP dasar selama beban akses berada pada tingkat rendah hingga sedang.

Perbandingan kedua skenario tersebut menunjukkan adanya *trade-off* yang jelas antara performa dan efisiensi penggunaan sumber daya. Skenario tanpa pembatasan sumber daya lebih unggul dari sisi kinerja dan stabilitas layanan, sedangkan skenario dengan pembatasan sumber daya lebih efisien dalam penggunaan *resource* namun mengorbankan kualitas layanan. Temuan ini menegaskan pentingnya pengelolaan dan penentuan alokasi sumber daya yang tepat dalam penerapan *container*, khususnya untuk layanan web berbasis NGINX.

Dari sudut pandang pendidikan, hasil penelitian ini memiliki nilai edukatif yang tinggi. Penggunaan *Play-With-Docker* sebagai media simulasi terbukti efektif dalam mendukung kegiatan praktikum berbasis *container*, meskipun digunakan pada perangkat dengan spesifikasi yang beragam. Skenario tanpa limit dapat dimanfaatkan untuk mendemonstrasikan performa maksimal NGINX, sementara skenario dengan limit *resource* dapat digunakan untuk memperkenalkan konsep manajemen sumber daya, efisiensi, serta dampak keterbatasan *resource* terhadap kinerja aplikasi.

Sebagai pengembangan ke depan, penelitian selanjutnya dapat memperluas variasi pembatasan sumber daya, meningkatkan jumlah dan intensitas beban pengujian, serta menggunakan alat benchmarking yang lebih komprehensif. Selain itu, kajian lanjutan juga dapat dilakukan dengan membandingkan NGINX dengan web server lain atau menerapkan pengujian pada lingkungan *orkestrasi container* seperti Kubernetes. Dengan pengembangan tersebut, diharapkan pemahaman mengenai kinerja dan pengelolaan *container* dapat semakin mendalam dan aplikatif, baik dalam konteks pendidikan maupun implementasi nyata.

Daftar Pustaka

Alam, S., Lestari, S., Fauziyyah, A.K. and Dzulfikar, D. (2024) 'Implementasi Docker container untuk sistem monitoring dan pengontrolan peralatan listrik di laboratorium cerdas', JELIKU: Jurnal Elektronika Ilmu Komputer Udayana, 12(3), p. 697. Available at: https://ojs.unud.ac.id/index.php/jlk/article/view/109156

Alamsyah, M.R., Wahanani, H.E. and Idhom, M. (2021) 'Infrastruktur private cloud storage berbasis', Jurnal Informatika dan Sistem Informasi, 2(2), pp. 122–131.

Anand, A. and Jebaseeli, A.N. (2019) 'Performance comparison between VM-based webserver and Docker container webserver', Asian Journal of Computer Science and Technology, 8(S2), pp. 28–30. https://doi.org/10.51983/ajcst-2019.8.s2.2030

Boettiger, C. (2014) 'An introduction to Docker for reproducible research, with examples from the R environment', ACM SIGOPS Operating Systems Review, 49(1), pp. 71–79. https://doi.org/10.1145/2723872.2723882

Cahyaningrum, Y. and Widiasari, I.R. (2020) 'Analisis performa container berplatform Docker atas serangan malicious software (malware)', Jurnal Buana Informatika, 11(1), pp. 47–54. https://doi.org/10.24002/jbi.v11i1.3279

Ekaputra, A.R. and Affandi, A.S. (2023) 'Pemanfaatan layanan cloud computing dan Docker container untuk meningkatkan kinerja aplikasi web', Journal of Information System Application and Development, 1(2), pp. 138–147. https://doi.org/10.26905/jisad.v1i2.11084

Elmenreich, W., Moll, P., Theuermann, S. and Lux, M. (2019) 'Making simulation results reproducible: survey, guidelines, and examples based on Gradle and Docker', PeerJ Computer Science, 2019(12), pp. 1–27. https://doi.org/10.7717/peerj-cs.240

Jain, V., Singh, B., Khenwar, M. and Sharma, M. (2021) 'Static vulnerability analysis of Docker images', IOP Conference Series: Materials Science and Engineering, 1131(1), p. 012018. https://doi.org/10.1088/1757-899X/1131/1/012018

Kwon, S. and Lee, J.H. (2020) 'DIVDS: Docker image vulnerability diagnostic system', IEEE Access, 8, pp. 42666–42673. https://doi.org/10.1109/ACCESS.2020.2976874

Megantara, R.A., Alzami, F., Anggi, R. and Puji, D. (2022) 'Memaksimalkan utilitas server universitas pada masa Covid-19', Prosiding Seminar Nasional, pp. 48–54.

Potdar, A.M., Narayan, D.G., Kengond, S. and Mulla, M.M. (2020) 'Performance evaluation of Docker container and virtual machine', Procedia Computer Science, 171, pp. 1419–1428. https://doi.org/10.1016/j.procs.2020.04.152

Ramsari, N. and Ginanjar, A. (2022) 'Implementasi infrastruktur server berbasis cloud computing untuk web service berbasis teknologi Google Cloud Platform', Conference on Senat STT Adisutjipto Yogyakarta, 7. https://doi.org/10.28989/senatik.v7i0.472

Wahanani, H.E. and Idhom, M. (2020) 'Analisis performansi container Docker, LXC dan LXD pada web server, FTP server dan mail server', Prosiding Seminar Nasional Informatika Bela Negara, 1, pp. 45–49. https://doi.org/10.33005/santika.v1i0.13

Wilford, S. (2025) 'Precept: a framework for ethical digital investigations', pp. 1–18.