Jurnal RISTIKA – *Jurnal Riset Teknologi Informasi, Manajemen, dan Multimedia*
Ristika., 1(2) 86-97
Februari 2026

Studi Eksperimental terhadap Permasalahan Akses Localhost pada Docker Desktop Berbasis Windows

Kiagus Muhamad Bayu ivandra[1*], Muhammad Ridho Ardiansyah[2], Bella Paramita[3]

[1]Manajemen Informatika, Institut Teknologi Dan Bisnis Bina Sriwijaya, Kota Palembang,Indonesia
[2]Sistem Informasi, Institut Teknologi Dan Bisnis Bina Sriwijaya, Kota Palembang,Indonesia
[3]Teknik Komputer, Institut Teknologi Dan Bisnis Bina Sriwijaya, Kota Palembang,Indonesia
[1]bayuivandrak@gmail.com, [2]ridho.ard@gmail.com, [3]bellaparamita88@gmail.com

Received : 21 Desember 2025
Received in revised from: 31 Januari 2026
Accepted : 8 Februari 2026

Abstract – Docker has become a widely adopted containerization platform due to its ability to provide consistency, portability, and efficiency in application development and deployment processes. Despite its advantages, the implementation of Docker on Windows-based environments frequently encounters technical challenges, particularly related to the inability to access containerized applications through localhost, even when containers are running properly. This issue often leads to confusion among developers and hinders local testing and debugging activities. Therefore, this study aims to experimentally investigate the underlying causes and effective solutions for localhost access failures in Docker Desktop running on the Windows operating system. This research employs an experimental method by deploying a containerized web server as a case study and systematically observing the behavior of localhost access under various configuration scenarios. The experimental stages include container execution, port mapping configuration, localhost access testing via web browsers, log observation, and iterative troubleshooting. Several technical factors were examined, including port mapping conflicts, firewall restrictions, Docker daemon stability, and the behavior of Docker's internal networking in a Windows Subsystem for Linux 2 (WSL2) environment. The experimental results indicate that localhost access failures are not caused by a single factor but arise from the interaction of multiple system components. Incorrect or conflicting port usage, restrictive firewall settings, and instability in Docker daemon services were identified as dominant factors contributing to access failure. Furthermore, the additional virtualization layer introduced by Docker Desktop and WSL2 increases network complexity, which can disrupt request forwarding from the host system to containers. The study demonstrates that applying a structured troubleshooting approach—such as using alternative ports, adjusting firewall permissions, and restarting Docker services—effectively restores stable localhost access. Overall, this study provides a comprehensive technical understanding of localhost access issues in Docker Desktop on Windows and offers practical insights that can serve as a reference for developers, students, and practitioners in optimizing container-based development environments.

Keywords: Docker Desktop, Localhost Access, Container Networking, Windows Operating System, Port Mapping, Troubleshooting.

Pendahuluan

Transformasi digital yang berlangsung secara masif dalam beberapa tahun terakhir telah mendorong perubahan mendasar pada paradigma pengembangan dan pengelolaan sistem perangkat lunak. Organisasi, institusi pendidikan, dan industri teknologi dituntut untuk menghasilkan aplikasi yang tidak hanya fungsional, tetapi juga fleksibel, mudah dipelihara, serta mampu beradaptasi dengan cepat terhadap perubahan kebutuhan pengguna dan lingkungan komputasi. Dalam konteks tersebut, pendekatan tradisional berbasis instalasi manual dan konfigurasi sistem yang bergantung pada spesifikasi perangkat keras tertentu mulai ditinggalkan karena dinilai kurang efisien dan rentan terhadap permasalahan kompatibilitas (Afrizal and Prihanto, 2022).

Sebagai respons terhadap tantangan tersebut, teknologi virtualisasi berkembang pesat dan melahirkan konsep kontainerisasi sebagai pendekatan yang lebih ringan dan efisien dibandingkan virtualisasi konvensional. Kontainer memungkinkan aplikasi dijalankan dalam lingkungan terisolasi yang mencakup seluruh dependensi yang dibutuhkan, tanpa harus memuat sistem operasi secara penuh. Salah satu platform kontainerisasi yang paling luas digunakan saat ini adalah Docker. Docker menawarkan mekanisme pengemasan aplikasi yang konsisten melalui *image* dan *container*, sehingga aplikasi dapat dijalankan secara seragam di berbagai lingkungan komputasi, mulai dari mesin pengembang hingga server produksi (Dwiyatno, Saleh and Gustiawan, 2020).

Popularitas Docker tidak terlepas dari berbagai keunggulan yang ditawarkannya, seperti efisiensi penggunaan sumber daya, kemudahan replikasi lingkungan aplikasi, serta percepatan proses pengembangan dan *deployment*. Docker juga mendukung penerapan arsitektur modern seperti microservices, yang memungkinkan aplikasi dibangun dalam komponen-komponen kecil dan independen sehingga lebih mudah dikembangkan dan dipelihara (Fathurrahman and Darmizal, 2024). Selain itu, Docker banyak dimanfaatkan dalam pengembangan web server, sistem penyimpanan terdistribusi, mail server, hingga infrastruktur berketahanan tinggi (*high availability*) (Putra, Fitri and Iskandar, 2020; Ridla et al., 2025; Irawan, 2024).

Dalam konteks pendidikan dan penelitian, Docker juga berperan penting sebagai sarana pembelajaran dan eksperimen teknologi jaringan serta sistem terdistribusi. Penggunaan Docker memungkinkan mahasiswa dan peneliti untuk membangun lingkungan praktikum yang terstandarisasi, mudah direplikasi, dan tidak bergantung pada konfigurasi perangkat keras tertentu (Nugraha, Data and Siregar, 2018). Dengan demikian, Docker tidak hanya berfungsi sebagai alat teknis, tetapi juga sebagai media pendukung penguatan kompetensi digital di bidang teknologi informasi.

Meskipun demikian, di balik berbagai keunggulan tersebut, penerapan Docker tidak selalu berjalan tanpa kendala, khususnya ketika digunakan pada sistem operasi Windows. Berbeda dengan Linux yang merupakan lingkungan *native* Docker, sistem operasi Windows memerlukan lapisan virtualisasi tambahan untuk menjalankan kernel Linux, umumnya melalui Windows *Subsystem for* Linux 2 (WSL2) atau *Hyper*-V. Keberadaan lapisan ini menambah kompleksitas arsitektur sistem, terutama pada aspek jaringan dan komunikasi antar komponen, yang berpotensi menimbulkan permasalahan teknis yang tidak mudah diidentifikasi oleh pengguna (Aldiwidianto and Prismana, 2023).

Salah satu permasalahan teknis yang paling sering ditemui dalam penggunaan Docker di Windows adalah kegagalan akses aplikasi melalui alamat localhost, meskipun container berada dalam kondisi berjalan (*running*). Permasalahan ini biasanya ditandai dengan munculnya pesan kesalahan pada peramban web seperti *Connection Refused* atau *This site can't be reached*. Kondisi tersebut menimbulkan kebingungan karena secara logis aplikasi seharusnya dapat diakses selama *container* aktif dan port telah dipetakan. Namun, dalam praktiknya, kegagalan akses *localhost* sering kali disebabkan oleh faktor teknis yang saling berkaitan dan tidak bersifat tunggal.

Permasalahan akses *localhost* pada Docker Desktop di Windows dapat dipengaruhi oleh berbagai aspek, antara lain ketidakstabilan layanan Docker daemon, kesalahan konfigurasi pemetaan *port* (*port mapping*), konflik penggunaan port dengan layanan lain pada sistem *host*, pembatasan lalu lintas jaringan oleh *firewall* Windows, serta konfigurasi jaringan virtual yang diterapkan oleh Docker Desktop dan WSL2 (Rokiman, Azindani and Akhsani, 2025). Kompleksitas ini menjadikan proses penelusuran kesalahan (*troubleshooting*) tidak selalu sederhana, terutama bagi pengguna yang belum memiliki pemahaman mendalam mengenai mekanisme kerja jaringan Docker.

Dalam pengembangan aplikasi berbasis web, akses localhost merupakan tahap krusial dalam proses pengujian dan validasi aplikasi sebelum diterapkan ke lingkungan produksi. Ketika akses ini gagal, pengembang tidak dapat memastikan bahwa aplikasi berjalan sesuai dengan spesifikasi yang diharapkan. Kondisi tersebut tidak hanya menghambat proses *debugging*, tetapi juga berpotensi memperpanjang waktu pengembangan dan meningkatkan risiko kesalahan pada tahap implementasi. Dalam lingkungan pendidikan, permasalahan ini bahkan dapat mengalihkan fokus pembelajaran dari pemahaman konsep inti kontainerisasi ke persoalan konfigurasi teknis yang kompleks.

Sejumlah penelitian terdahulu telah membahas pemanfaatan Docker dalam berbagai konteks, seperti implementasi high availability dan load balancing (Putra, Fitri and Iskandar, 2020; Rifiera and Nurwarsito, 2022), analisis perbandingan orkestrasi menggunakan Kubernetes dan Docker Swarm (Aldiwidianto and Prismana, 2023), serta penerapan Docker dalam arsitektur *microservices* (Fathurrahman and Darmizal, 2024). Penelitian lain juga menyoroti aspek keamanan Docker melalui analisis kerentanan serta monitoring celah keamanan pada *cluster container* (Astriani, 2021; Bediatra, 2023).

Namun demikian, sebagian besar penelitian tersebut menempatkan Docker sebagai solusi yang telah stabil dan matang, sehingga fokus kajian diarahkan pada optimalisasi performa, skalabilitas, dan keamanan sistem. Pembahasan mengenai permasalahan teknis dasar yang sering muncul pada tahap awal penggunaan, khususnya kegagalan akses localhost di lingkungan Windows, masih relatif terbatas. Kajian yang ada umumnya bersifat deskriptif dan tidak disusun dalam kerangka eksperimen yang sistematis, sehingga sulit dijadikan rujukan akademik yang aplikatif.

Selain itu, penelitian yang membahas Docker dalam konteks pendidikan dan pengembangan sistem informasi lebih menekankan pada manfaat dan implementasi fungsional sistem, seperti pengembangan sistem informasi berbasis web dan peningkatan literasi digital (Taqwiym and Satria, 2025a; Taqwiym and Satria, 2025b), tanpa menguraikan secara mendalam kendala teknis yang sering dihadapi dalam proses pengembangan menggunakan Docker di lingkungan Windows. Padahal, kendala teknis tersebut memiliki dampak langsung terhadap keberhasilan pengembangan dan kualitas sistem yang dihasilkan.

Berdasarkan telaah literatur tersebut, dapat diidentifikasi adanya *research gap* yang jelas, yaitu minimnya penelitian yang secara khusus dan sistematis membahas penyebab dan solusi *error* akses *localhost* pada Docker Desktop di sistem operasi Windows melalui pendekatan eksperimen langsung. Belum banyak penelitian yang mengintegrasikan analisis konfigurasi Docker, mekanisme jaringan virtual WSL2, pemetaan port, dan pengaruh *firewall* Windows dalam satu kerangka kajian yang utuh dan komprehensif. Kesenjangan ini menunjukkan perlunya penelitian yang tidak hanya bersifat konseptual, tetapi juga berbasis pada pengujian empiris di lingkungan nyata.

Selain itu, terdapat celah penelitian pada aspek validasi solusi teknis. Banyak solusi yang beredar di komunitas pengguna Docker masih bersifat asumtif dan tidak diuji secara konsisten dalam berbagai skenario konfigurasi. Akibatnya, pengguna sering kali mencoba berbagai solusi secara acak tanpa memahami akar permasalahan yang sebenarnya. Kondisi ini menegaskan pentingnya penelitian yang menguji solusi secara berulang dan terdokumentasi, sehingga dapat memberikan panduan teknis yang reliabel dan dapat direplikasi.

Oleh karena itu, penelitian ini dilakukan dengan tujuan untuk mengidentifikasi dan menganalisis secara mendalam faktor-faktor penyebab terjadinya error akses localhost pada penggunaan Docker di sistem operasi Windows. Penelitian ini menggunakan metode eksperimen dengan menjalankan *container* web sebagai studi kasus, kemudian mengamati perilaku akses jaringan melalui *localhost* serta menganalisis aspek teknis yang meliputi status Docker daemon, konfigurasi pemetaan port, pengaturan jaringan Docker Desktop dan WSL2, serta pengaruh firewall Windows. Setiap indikasi kesalahan ditelusuri melalui pemeriksaan status *container* dan analisis log Docker untuk menemukan akar permasalahan yang terjadi.

Dengan pendekatan tersebut, penelitian ini diharapkan mampu memberikan kontribusi ilmiah berupa pemahaman yang komprehensif dan aplikatif mengenai mekanisme jaringan Docker di lingkungan Windows. Selain itu, hasil penelitian ini diharapkan dapat menjadi referensi teknis yang bermanfaat bagi pengembang, mahasiswa, dan praktisi teknologi informasi dalam mengatasi permasalahan akses localhost secara efektif dan sistematis. Secara lebih luas, penelitian ini juga berkontribusi terhadap penguatan literasi digital teknis dan optimalisasi pemanfaatan teknologi kontainer dalam pengembangan sistem informasi yang andal dan berkelanjutan.

Metode

Penelitian ini menggunakan metode eksperimen dengan pendekatan deskriptif-analitis untuk mengidentifikasi, menganalisis, dan memvalidasi penyebab serta solusi teknis terhadap permasalahan error akses localhost pada penggunaan Docker di sistem operasi Windows. Metode eksperimen dipilih karena permasalahan yang diteliti bersifat teknis, kontekstual, dan sangat dipengaruhi oleh konfigurasi sistem, sehingga memerlukan pengujian langsung pada lingkungan nyata agar diperoleh hasil yang akurat dan dapat dipertanggungjawabkan secara ilmiah.

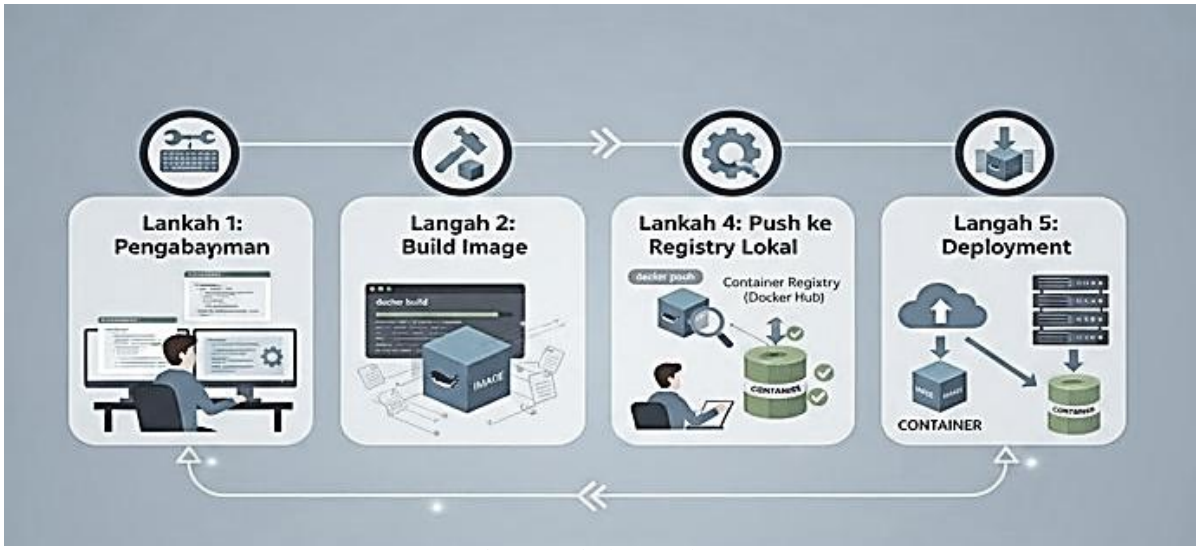
Pendekatan ini memungkinkan peneliti untuk mengamati perilaku sistem secara empiris, mengidentifikasi hubungan sebab-akibat antara konfigurasi Docker dan kegagalan akses localhost, serta menguji efektivitas solusi teknis melalui penerapan dan pengujian berulang. Dengan demikian, metode eksperimen dianggap paling relevan untuk menjawab tujuan penelitian yang berfokus pada analisis permasalahan teknis dan troubleshooting berbasis praktik.

Lingkungan dan Objek Penelitian

Lingkungan penelitian dibangun pada sistem operasi Windows dengan memanfaatkan Docker Desktop sebagai platform utama kontainerisasi. Docker dijalankan menggunakan Windows *Subsystem for* Linux 2 (WSL2) sebagai mesin virtual yang menyediakan kernel Linux, sehingga memungkinkan *container* berbasis Linux berjalan pada sistem Windows. Pemilihan lingkungan ini didasarkan pada fakta bahwa konfigurasi tersebut merupakan

konfigurasi yang paling umum digunakan oleh pengembang dan mahasiswa saat ini, sekaligus menjadi sumber utama munculnya permasalahan akses localhost.

Objek penelitian berupa container web server yang dijalankan menggunakan image resmi Nginx dari Docker *Registry*. Nginx dipilih karena merupakan web server yang ringan, stabil, dan banyak digunakan sebagai studi kasus dalam pengembangan aplikasi berbasis Docker. Penggunaan web server sebagai objek penelitian memungkinkan pengujian akses localhost dilakukan secara langsung melalui peramban web, sehingga gejala kesalahan dapat diamati dengan jelas dan terukur.



Gambar 1 alur kerja Docker

Tahapan Penelitian

Pelaksanaan penelitian dilakukan melalui beberapa tahapan yang terstruktur dan sistematis, sebagai berikut:

1. Persiapan Lingkungan Pengujian
   Tahap awal penelitian meliputi instalasi Docker Desktop pada sistem operasi Windows, pengaktifan WSL2, serta konfigurasi awal Docker agar dapat berjalan dengan optimal. Pada tahap ini juga dilakukan verifikasi status Docker daemon untuk memastikan bahwa layanan Docker berjalan dengan normal sebelum eksperimen dimulai.
2. Pembuatan dan Eksekusi *Container*
   Setelah lingkungan siap, image web server Nginx diunduh dari Docker *Registry* menggunakan perintah docker *pull*. Selanjutnya, *container* dijalankan menggunakan perintah docker *run* dengan berbagai skenario pemetaan *port* (*port mapping*), baik menggunakan port standar maupun port alternatif. Tujuan tahap ini adalah untuk mengamati bagaimana konfigurasi pemetaan port memengaruhi akses localhost dari sistem *host*.
3. Pengujian Akses Localhost
   Pengujian akses dilakukan melalui peramban web dengan mengakses alamat localhost atau localhost:port sesuai dengan konfigurasi pemetaan port yang diterapkan. Pada tahap ini dicatat apakah aplikasi dapat diakses dengan normal atau muncul pesan kesalahan seperti *Connection Refused* atau *Not Found*. Hasil pengujian ini menjadi indikator awal adanya permasalahan akses jaringan.
4. Analisis Kesalahan dan Observasi Sistem
   Apabila terjadi kegagalan akses, langkah selanjutnya adalah melakukan analisis kesalahan dengan memanfaatkan perintah-perintah Docker seperti docker ps untuk memeriksa status *container* dan docker logs untuk mengamati pesan log yang dihasilkan. Selain itu, dilakukan pemeriksaan terhadap kemungkinan konflik penggunaan port pada sistem *host* serta pengaruh firewall Windows terhadap lalu lintas jaringan Docker.
5. Penerapan dan Pengujian Solusi Teknis
   Berdasarkan hasil analisis kesalahan, berbagai solusi teknis diterapkan secara bertahap, antara lain dengan melakukan *restart* Docker daemon, mengubah konfigurasi pemetaan port, memberikan izin akses pada *firewall*

Windows, serta membangun ulang *container* apabila diperlukan. Setiap solusi yang diterapkan diuji kembali melalui akses localhost untuk memastikan efektivitasnya dalam mengatasi permasalahan.

6. Validasi dan Dokumentasi Hasil
   Seluruh proses pengujian dan penerapan solusi didokumentasikan secara sistematis, termasuk kondisi awal, langkah yang dilakukan, dan hasil yang diperoleh. Validasi dilakukan dengan mengulangi pengujian pada beberapa skenario konfigurasi untuk memastikan bahwa solusi yang ditemukan bersifat konsisten dan stabil.

Teknik Pengumpulan dan Analisis Data

Data dalam penelitian ini dikumpulkan melalui observasi langsung terhadap perilaku sistem, analisis log Docker, serta hasil pengujian akses localhost. Data yang diperoleh bersifat kualitatif dan teknis, berupa status container, pesan kesalahan, log sistem, serta respons peramban web. Analisis data dilakukan dengan mengkaji keterkaitan antara konfigurasi sistem dan gejala kesalahan yang muncul, kemudian menarik kesimpulan mengenai faktor penyebab dan solusi yang paling efektif.

Pendekatan analisis dilakukan secara deskriptif-analitis dengan membandingkan hasil pengujian sebelum dan sesudah penerapan solusi. Dengan cara ini, perubahan perilaku sistem dapat diamati secara jelas dan dijadikan dasar dalam penarikan kesimpulan.

Hasil

Fokus utama penelitian adalah mengamati secara langsung proses terjadinya error akses *localhost* pada *container* Docker, menelusuri faktor penyebabnya, serta mengevaluasi efektivitas solusi teknis yang diterapkan. Seluruh hasil disusun berdasarkan urutan pengambilan data yang sistematis, dimulai dari kondisi awal lingkungan pengujian, proses eksekusi *container*, pengamatan gejala kesalahan, hingga validasi hasil setelah penerapan solusi.

Kondisi Awal Lingkungan Pengujian

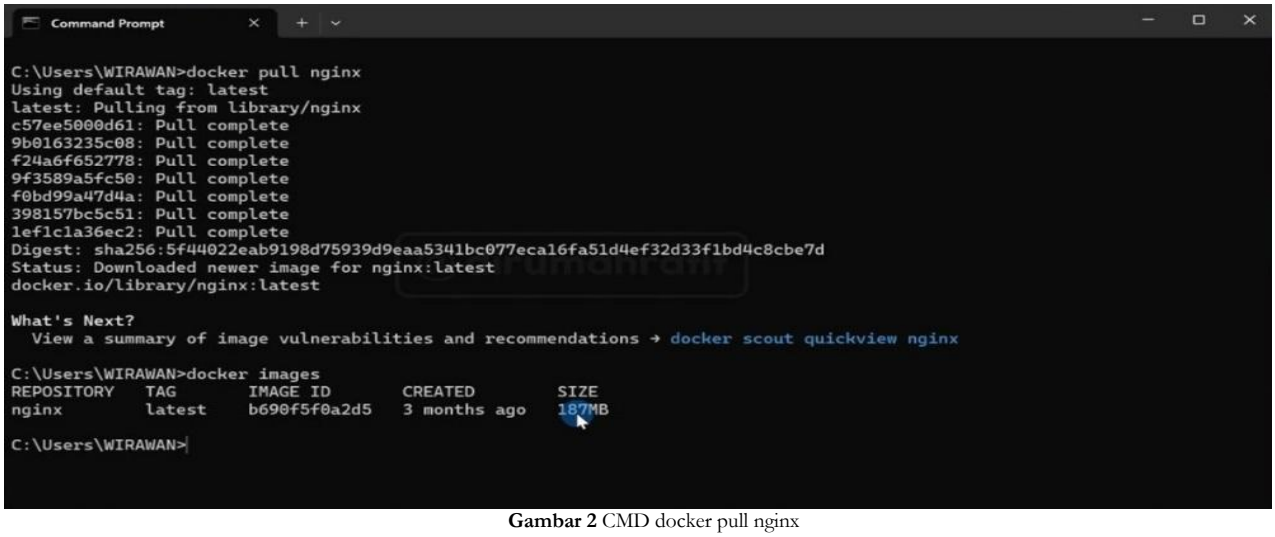
Pada tahap awal, Docker Desktop berhasil dijalankan pada sistem operasi Windows dengan konfigurasi *default* menggunakan WSL2 sebagai backend. Layanan Docker daemon terverifikasi berjalan normal tanpa adanya pesan kesalahan pada antarmuka Docker Desktop. Kondisi ini menunjukkan bahwa secara umum Docker telah siap digunakan dan tidak terdapat indikasi gangguan pada tahap instalasi maupun inisialisasi awal sistem.

Selanjutnya, dilakukan pengujian awal untuk memastikan bahwa lingkungan terminal dapat berkomunikasi dengan Docker daemon. Perintah untuk menampilkan daftar container dijalankan dan menghasilkan *output* yang sesuai, meskipun belum terdapat container aktif pada tahap ini. Hasil tersebut menjadi dasar bahwa komunikasi antara Docker client dan Docker daemon berjalan dengan baik sebelum eksperimen dimulai.

Proses Pengunduhan dan Penyimpanan Image

Tahap berikutnya adalah pengunduhan *image* web *server* yang akan digunakan sebagai objek pengujian. Proses ini dilakukan dengan menarik image resmi Nginx dari Docker *Registry*. Selama proses pengunduhan, sistem menampilkan informasi mengenai lapisan-lapisan image yang diunduh secara bertahap hingga seluruh proses selesai tanpa *error*. Keberhasilan tahap ini menunjukkan bahwa koneksi jaringan Docker ke *registry* eksternal berjalan normal serta *image* berhasil tersimpan di sistem lokal.

Setelah proses pengunduhan selesai, dilakukan verifikasi terhadap image yang tersedia pada sistem. Hasil verifikasi menunjukkan bahwa *image* Nginx telah tersimpan dengan benar dan siap digunakan untuk pembuatan *container*. Pada tahap ini, belum ditemukan indikasi permasalahan yang berpotensi menyebabkan *error* akses *localhost*.

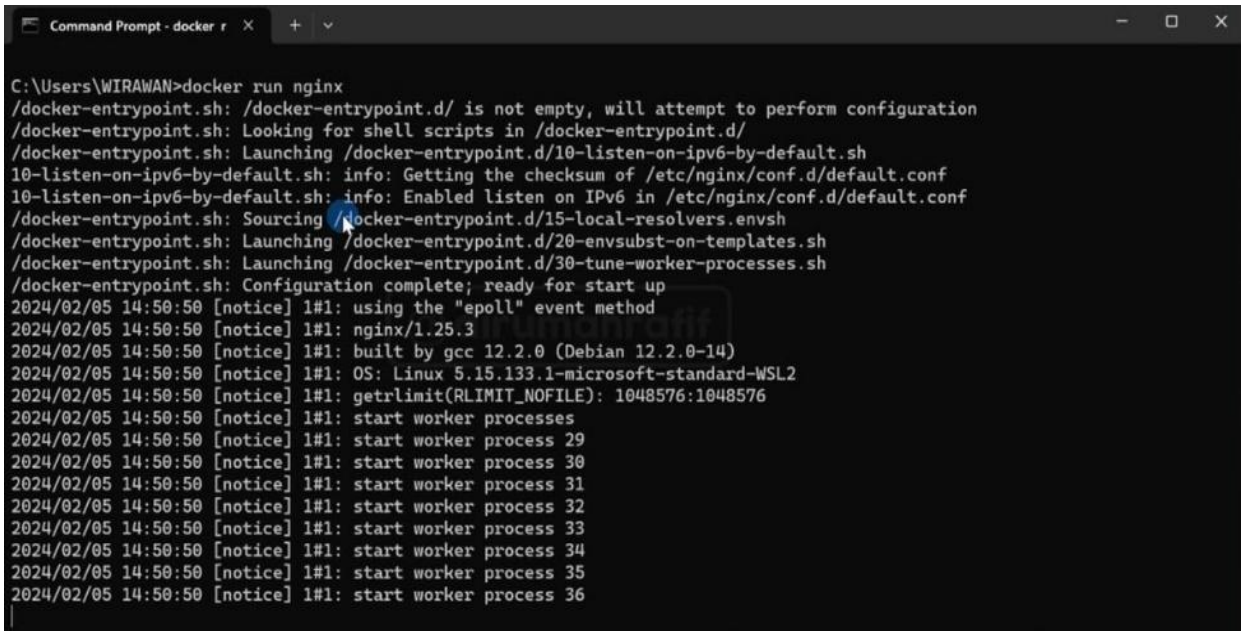


Gambar 2 CMD docker pull nginx

Eksekusi Container Tanpa Pemetaan Port

Container pertama dijalankan menggunakan konfigurasi standar tanpa melakukan pemetaan port antara *container* dan sistem *host*. Setelah perintah eksekusi dijalankan, *container* terdeteksi berjalan dalam kondisi aktif. Log *container* menunjukkan bahwa layanan web server di dalam *container* berhasil diinisialisasi dan tidak mengalami kesalahan saat proses startup.

Namun, ketika dilakukan pengujian akses melalui peramban web dengan menggunakan alamat localhost, halaman web tidak dapat diakses dan peramban menampilkan pesan penolakan koneksi. Pada tahap ini, kondisi *container* yang berjalan normal tetapi tidak dapat diakses menjadi indikasi awal adanya kesenjangan antara status internal *container* dan akses jaringan dari sistem *host*. Data ini menunjukkan bahwa keberadaan *container* dalam kondisi running tidak serta-merta menjamin bahwa layanan di dalamnya dapat diakses dari luar *container*.

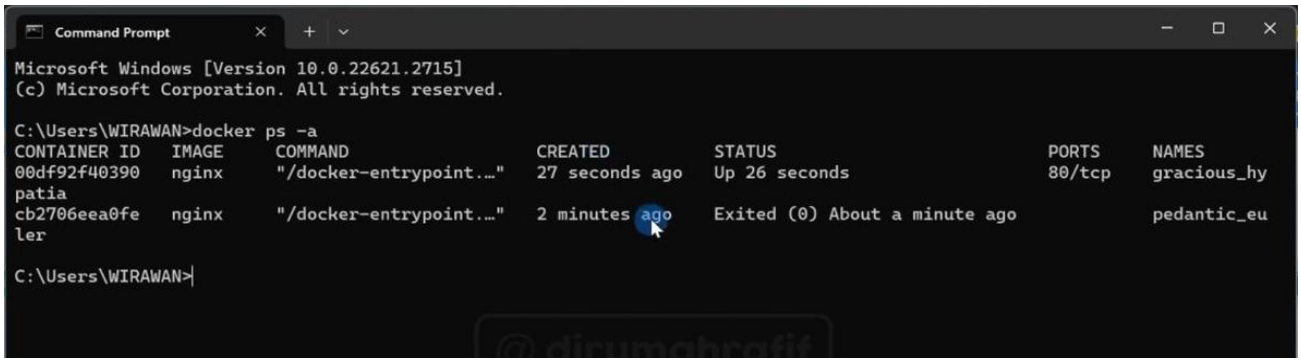


Gambar 3 Menjalankan server web Nginx di dalam container

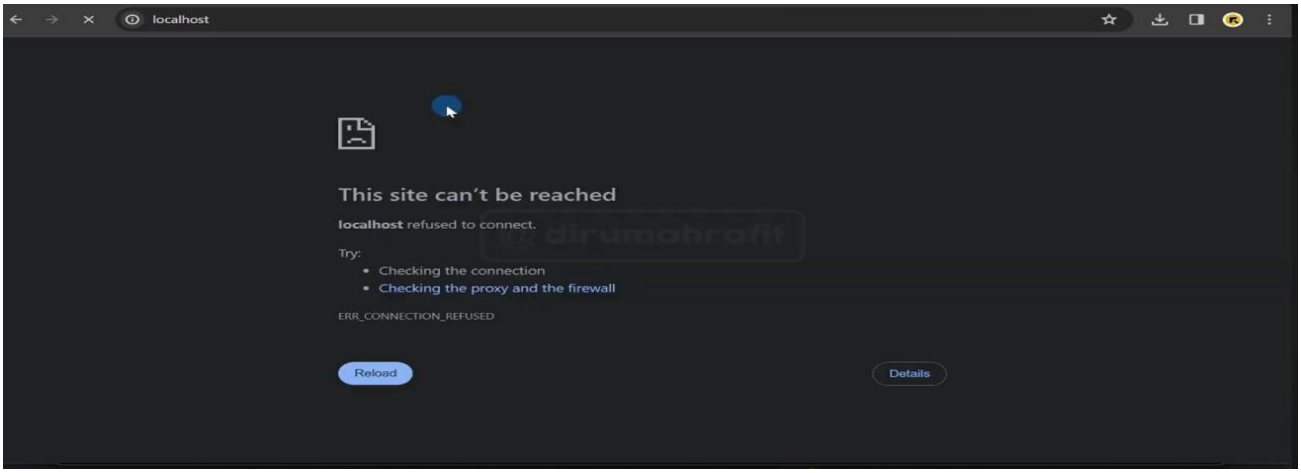
Pengamatan Status Container dan Log Sistem

Untuk menelusuri penyebab kegagalan akses tersebut, dilakukan pemeriksaan status *container* menggunakan perintah yang menampilkan daftar *container* aktif. Hasil pemeriksaan menunjukkan bahwa container masih berjalan tanpa adanya restart atau kegagalan proses. Selanjutnya, dilakukan pengamatan terhadap log container untuk memastikan bahwa tidak terjadi kesalahan internal pada layanan web *server*.

Log yang dihasilkan menunjukkan bahwa server web telah berjalan dan siap menerima koneksi pada port internal *container*. Tidak ditemukan pesan kesalahan yang mengindikasikan kegagalan layanan. Berdasarkan data ini, dapat disimpulkan bahwa penyebab kegagalan akses tidak berasal dari aplikasi di dalam *container*, melainkan dari mekanisme akses jaringan antara sistem *host* dan *container*.



Gambar 4 Setelah selesai unduh ke container



Gambar 5 localhost menolak untuk terhubung

Pengujian Container dengan Pemetaan Port Standar

Tahap selanjutnya adalah menjalankan *container* dengan konfigurasi pemetaan port standar, yaitu memetakan *port* internal layanan web ke port yang sama pada sistem *host*. *Container* kembali dijalankan dan terdeteksi aktif tanpa kendala. Informasi pemetaan port juga ditampilkan secara eksplisit pada daftar *container*, yang menunjukkan bahwa port telah dibuka dan diarahkan ke *container*.

Namun, hasil pengujian akses localhost pada konfigurasi ini masih menunjukkan kegagalan. Peramban kembali menampilkan pesan kesalahan yang serupa dengan pengujian sebelumnya. Data ini mengindikasikan bahwa meskipun pemetaan port telah diterapkan, masih terdapat faktor lain yang menghambat komunikasi jaringan antara *host* dan *container*.

Identifikasi Konflik Penggunaan Port

Untuk memperdalam analisis, dilakukan pemeriksaan terhadap kemungkinan konflik penggunaan *port* pada sistem *host*. Pada tahap ini, ditemukan bahwa *port* standar yang digunakan oleh layanan web juga berpotensi digunakan oleh layanan lain pada sistem Windows. Meskipun konflik tersebut tidak selalu ditampilkan secara eksplisit oleh Docker, keberadaannya dapat menyebabkan permintaan jaringan tidak diteruskan ke container sebagaimana mestinya.

Berdasarkan temuan ini, dilakukan perubahan strategi dengan menggunakan *port* alternatif pada sistem *host* untuk memetakan layanan web di dalam *container*. *Container* kemudian dijalankan ulang dengan konfigurasi pemetaan *port* baru yang berbeda dari *port* standar.

Hasil Pengujian dengan Pemetaan Port Alternatif

Setelah *container* dijalankan dengan *port* alternatif, dilakukan kembali pengujian akses melalui peramban web dengan menyertakan nomor *port* yang telah ditentukan. Pada tahap ini, hasil pengujian menunjukkan perubahan yang signifikan. Halaman *web default* dari layanan web server berhasil ditampilkan secara normal, yang menandakan bahwa komunikasi jaringan antara sistem *host* dan *container* telah berjalan dengan baik.

Keberhasilan ini menjadi data penting yang menunjukkan bahwa konflik penggunaan *port* merupakan salah satu faktor utama penyebab *error* akses *localhost*. Selain itu, hasil ini juga menunjukkan bahwa pemilihan *port* yang tepat pada sistem *host* memiliki pengaruh langsung terhadap keberhasilan akses layanan *container*.

Pengaruh Firewall Windows terhadap Akses Localhost

Meskipun akses berhasil pada konfigurasi tertentu, penelitian ini juga menemukan bahwa pada beberapa kondisi, *firewall* Windows berperan dalam membatasi lalu lintas jaringan Docker. Ketika pengaturan *firewall* diperketat, akses localhost kembali mengalami gangguan meskipun pemetaan *port* telah benar. Setelah dilakukan penyesuaian izin akses pada *firewall*, layanan kembali dapat diakses dengan normal.

Data ini menunjukkan bahwa *firewall* Windows merupakan faktor eksternal yang signifikan dalam menentukan keberhasilan akses *localhost*. Ketidaksesuaian pengaturan *firewall* dapat menyebabkan *container* berjalan tanpa error tetapi tetap tidak dapat diakses dari sisi klien.

Stabilitas Docker Daemon dan Jaringan Internal

Selain faktor *port* dan *firewall*, penelitian ini juga mengamati bahwa pada kondisi tertentu Docker daemon mengalami ketidakstabilan jaringan internal. Hal ini ditandai dengan kondisi di mana konfigurasi yang sebelumnya berhasil tiba-tiba tidak lagi dapat diakses. Setelah dilakukan *restart* Docker daemon, layanan kembali berjalan normal tanpa perubahan konfigurasi tambahan.

Temuan ini menunjukkan bahwa layanan Docker daemon berperan sebagai komponen sentral dalam mekanisme jaringan *container*. Ketika layanan ini mengalami gangguan, seluruh pemetaan jaringan dapat terdampak meskipun *container* dan konfigurasi lainnya tidak berubah.

Validasi dan Pengulangan Pengujian

Untuk memastikan konsistensi hasil, seluruh skenario pengujian diulang beberapa kali dengan urutan yang sama. Hasil pengujian menunjukkan pola yang konsisten, yaitu:

1. *Container* dapat berjalan normal tanpa jaminan akses *localhost*.
2. Pemetaan *port* merupakan syarat utama, tetapi tidak selalu cukup.
3. Konflik *port*, *firewall*, dan stabilitas Docker daemon menjadi faktor dominan penyebab kegagalan akses.

4. Kombinasi pemetaan port yang tepat, izin *firewall* yang sesuai, dan layanan Docker daemon yang stabil menghasilkan akses *localhost* yang konsisten.

Pembahasan

Pembahasan ini bertujuan untuk menafsirkan temuan penelitian secara analitis dengan mengaitkannya secara langsung terhadap celah penelitian (*research gap*) yang telah diidentifikasi pada bagian pendahuluan. Fokus pembahasan tidak hanya pada pemaparan kembali hasil eksperimen, tetapi pada penjelasan rasional mengenai mekanisme terjadinya error akses localhost pada Docker di sistem operasi Windows, serta kontribusi temuan penelitian ini dalam memperkaya kajian ilmiah terkait penggunaan Docker pada lingkungan non-native.

Hasil penelitian menunjukkan bahwa status *container* yang berada dalam kondisi aktif (*running*) tidak serta-merta menjamin ketersediaan layanan aplikasi melalui localhost. Temuan ini mengindikasikan adanya perbedaan antara kondisi operasional internal *container* dan kemampuan akses jaringan dari sistem *host*. Perbedaan tersebut menjadi aspek penting yang belum banyak dibahas dalam kajian terdahulu, khususnya pada konteks Docker Desktop berbasis Windows. Dengan demikian, penelitian ini mempertegas bahwa pemahaman terhadap mekanisme jaringan Docker perlu melampaui indikator status container semata.

Salah satu temuan utama penelitian ini adalah bahwa pemetaan *port* (*port mapping*) merupakan prasyarat penting, namun bukan satu-satunya faktor penentu keberhasilan akses *localhost*. Dalam beberapa skenario pengujian, pemetaan *port* yang telah dikonfigurasi dengan benar masih belum menghasilkan akses yang berhasil. Kondisi ini menunjukkan bahwa pendekatan penyelesaian masalah yang hanya berfokus pada konfigurasi *port* bersifat tidak komprehensif. Temuan tersebut secara langsung mengisi celah penelitian yang selama ini cenderung menyederhanakan permasalahan akses Docker sebagai isu konfigurasi port semata.

Aspek konflik penggunaan *port* pada sistem operasi Windows juga menjadi faktor signifikan yang teridentifikasi dalam penelitian ini. Berbeda dengan lingkungan Linux native, sistem operasi Windows memungkinkan berbagai layanan berjalan secara paralel dengan alokasi port yang dinamis. Konflik ini tidak selalu terdeteksi secara eksplisit oleh Docker, sehingga dapat menyebabkan kegagalan akses tanpa disertai pesan kesalahan yang jelas. Temuan ini memberikan kontribusi baru terhadap pemahaman mengenai karakteristik lingkungan Windows yang memengaruhi perilaku jaringan Docker dan selama ini kurang mendapatkan perhatian dalam kajian akademik.

Selain konflik *port*, penelitian ini menegaskan peran penting firewall Windows dalam menentukan keberhasilan akses *localhost*. Hasil pengujian menunjukkan bahwa kebijakan firewall yang ketat dapat membatasi lalu lintas jaringan Docker, meskipun *container* dan konfigurasi port telah disiapkan dengan benar. Kondisi ini memperjelas bahwa mekanisme keamanan pada sistem host memiliki pengaruh langsung terhadap operasional *container*. Dengan demikian, analisis permasalahan Docker di Windows perlu mencakup evaluasi kebijakan keamanan *host* sebagai bagian integral dari proses troubleshooting.

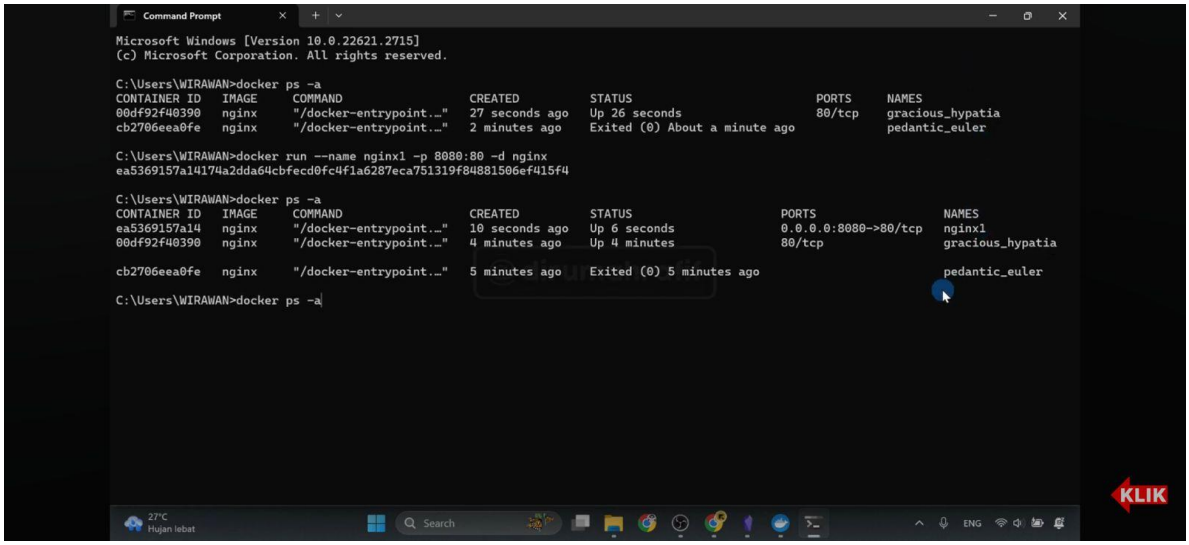
Penelitian ini juga mengidentifikasi bahwa stabilitas Docker daemon dan jaringan internal Docker Desktop memiliki peran sentral dalam mekanisme komunikasi antara sistem *host* dan *container*. Pada beberapa kondisi, gangguan pada layanan Docker daemon menyebabkan kegagalan akses localhost meskipun tidak terjadi perubahan konfigurasi. Temuan ini menunjukkan bahwa permasalahan jaringan Docker tidak selalu bersifat statis dan dapat dipengaruhi oleh kondisi runtime layanan. Oleh karena itu, pemeliharaan stabilitas layanan Docker daemon menjadi faktor penting dalam menjamin konsistensi akses aplikasi berbasis *container*.

Interaksi antara Docker Desktop, WSL2, dan sistem operasi Windows turut memperkuat kompleksitas permasalahan yang diteliti. Penelitian ini menunjukkan bahwa lapisan virtualisasi yang disediakan oleh WSL2 menambah jalur komunikasi jaringan yang harus dilalui sebelum permintaan dari peramban mencapai *container*. Gangguan pada salah satu lapisan tersebut berpotensi menyebabkan kegagalan akses *localhost* tanpa memberikan indikasi kesalahan yang mudah dikenali oleh pengguna. Temuan ini mengisi kekosongan kajian yang sebelumnya belum menguraikan keterkaitan antar lapisan sistem secara mendalam.

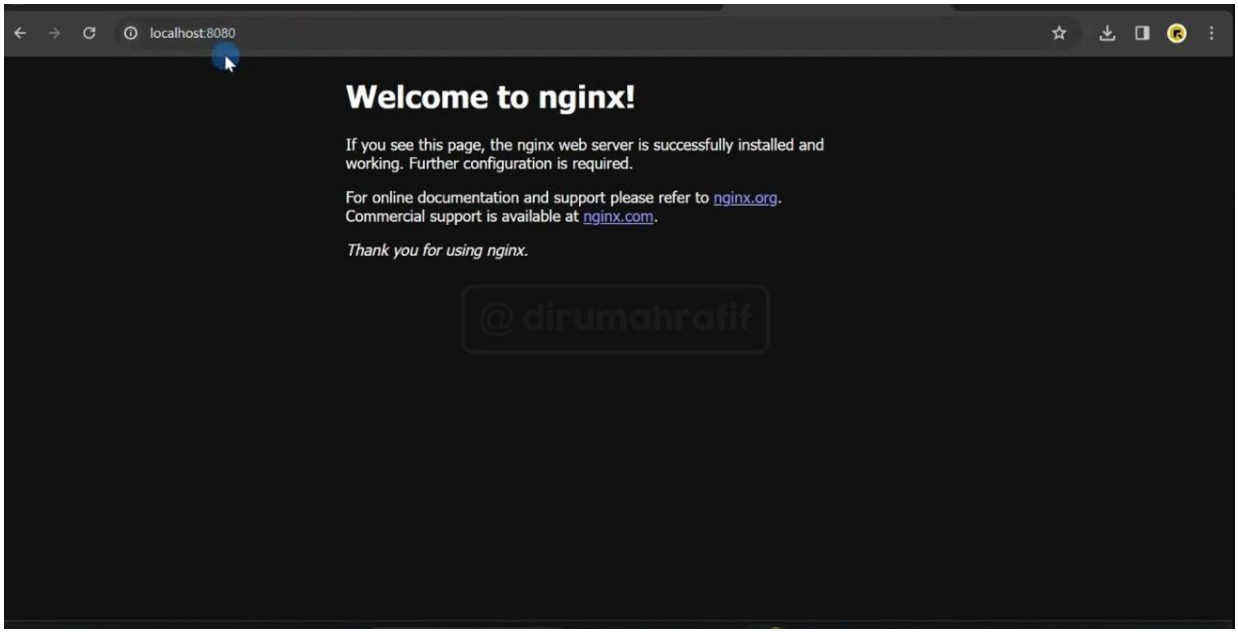
Dari sudut pandang metodologis, penelitian ini memberikan kontribusi dengan menerapkan pendekatan eksperimen terkontrol yang disertai dengan pengujian berulang dan dokumentasi sistematis. Pendekatan ini memungkinkan identifikasi pola penyebab dan solusi yang konsisten, sehingga hasil penelitian tidak bersifat subjektif atau berdasarkan pengalaman individual semata. Dengan demikian, temuan yang dihasilkan memiliki tingkat reliabilitas yang lebih tinggi dan dapat direplikasi pada lingkungan yang serupa.

Implikasi praktis dari temuan penelitian ini menunjukkan bahwa penyelesaian error akses localhost pada Docker di Windows memerlukan pendekatan yang holistik dan terintegrasi. Solusi teknis seperti penggunaan port alternatif terbukti efektif, namun harus diiringi dengan penyesuaian kebijakan *firewall* serta verifikasi stabilitas Docker daemon. Pendekatan yang terfragmentasi berpotensi menghasilkan solusi sementara yang tidak berkelanjutan. Oleh karena itu, pemahaman menyeluruh terhadap ekosistem Docker Desktop di Windows menjadi kunci utama dalam mengatasi permasalahan secara efektif.

Secara keseluruhan, pembahasan ini menegaskan bahwa penelitian telah berhasil mengisi celah penelitian yang ada dengan menghadirkan analisis komprehensif mengenai *error* akses *localhost* pada Docker di sistem operasi Windows. Temuan penelitian ini memperluas pemahaman mengenai karakteristik jaringan Docker pada lingkungan non-native dan memberikan dasar konseptual serta teknis yang kuat bagi pengembang, praktisi, dan akademisi. Dengan demikian, penelitian ini memberikan kontribusi yang relevan baik secara teoritis maupun praktis dalam pengembangan dan pemanfaatan teknologi kontainerisasi.



Gambar 6 Hasil membuat portnya



Gambar 7 Hasil setelah cek browser

Kesimpulan

Berdasarkan hasil penelitian dan pembahasan yang telah dilakukan, dapat disimpulkan bahwa permasalahan *error* akses *localhost* pada penggunaan Docker di sistem operasi Windows merupakan isu teknis yang bersifat kompleks dan tidak disebabkan oleh satu faktor tunggal. Meskipun *container* berada dalam kondisi aktif (*running*), akses terhadap layanan aplikasi di dalam *container* tidak selalu dapat dilakukan secara langsung melalui *localhost*. Temuan ini menunjukkan adanya perbedaan antara kondisi operasional internal *container* dan mekanisme akses jaringan dari sistem host, khususnya pada lingkungan Docker Desktop berbasis Windows.

Hasil penelitian membuktikan bahwa pemetaan *port* (*port mapping*) merupakan prasyarat utama untuk memungkinkan komunikasi antara sistem *host* dan *container*, namun tidak bersifat determinan secara tunggal. Konflik penggunaan *port* pada sistem operasi Windows, pembatasan lalu lintas jaringan oleh *firewall*, serta ketidakstabilan layanan Docker daemon terbukti memiliki pengaruh signifikan terhadap keberhasilan akses localhost. Selain itu, kompleksitas jaringan yang muncul akibat penggunaan Windows Subsystem for Linux 2 (WSL2) turut memperbesar potensi terjadinya kegagalan akses meskipun konfigurasi dasar telah diterapkan dengan benar.

Penelitian ini juga menunjukkan bahwa solusi terhadap permasalahan error akses *localhost* tidak dapat dilakukan secara parsial atau instan. Pendekatan troubleshooting yang efektif memerlukan analisis menyeluruh terhadap konfigurasi *container*, sistem *host*, serta layanan pendukung Docker. Penerapan solusi seperti penggunaan *port* alternatif, penyesuaian kebijakan *firewall* Windows, dan pemulihan layanan Docker daemon secara terstruktur terbukti mampu mengembalikan akses localhost secara konsisten dan stabil.

Secara keseluruhan, penelitian ini memberikan kontribusi berupa pemahaman teknis yang lebih komprehensif mengenai mekanisme jaringan Docker pada lingkungan Windows serta menawarkan panduan penanganan permasalahan akses localhost yang aplikatif dan dapat direplikasi. Temuan penelitian ini diharapkan dapat menjadi referensi bagi pengembang, mahasiswa, dan praktisi teknologi informasi dalam mengoptimalkan penggunaan Docker pada sistem operasi Windows, sekaligus mendukung pengembangan aplikasi berbasis kontainer yang lebih andal dan efisien.

Daftar Pustaka

Dwiyatno, Saleh, E.R. and Gustiawan, O. (2020) 'Implementasi virtualisasi server berbasis Docker container', Jurnal PROSISKO, 7(2).

Putri, C.L. (2023) Implementasi Zen Load Balancer pada server protokol berbasis Docker container. Skripsi.

Putra, M.A.A., Fitri, I. and Iskandar, A. (2020) 'Implementasi high availability cluster web server menggunakan virtualisasi container Docker', Jurnal Media Informatika Budidarma, 4(1), pp. 9–16.

Aldiwidianto, W. and Prismana, I.G.L.P.E. (2023) 'Analisis perbandingan high availability pada web server menggunakan orchestration tool Kubernetes dan Docker Swarm', Journal of Informatics and Computer Science (JINACS), 5(2), pp. 138–148.

Ridla, A.A., et al. (2025) 'Implementasi MinIO sebagai storage dalam aplikasi LMS guru berbasis Docker', Journal of Computing and Informatics Research, 5(1), pp. 401–409.

Fathurrahman, Y. and Darmizal, T. (2024) 'Implementasi arsitektur microservices dan teknologi Docker pada aplikasi tracer alumni', Jurnal Sistem Informasi dan Teknologi, 6(2).

Astriani, T. (2021) 'Analisa kerentanan pada vulnerable Docker menggunakan scanner OpenVAS dan Docker Scan dengan acuan standar NIST 800-115', JATISI, 8(4), pp. 2041–2050.

Rifiera, S.N. and Nurwarsito, H. (2022) 'Implementasi load balancing dan failover pada proses migrasi container Docker', Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 6(5), pp. 2025–2033.

Tuara, H.A., Maridyah, N.A. and Khaerudin (2021) 'Implementasi CDN (Content Delivery Network) menggunakan Cloudflare terintegrasi dengan Docker container', Journal of Mechatronic and Electrical Engineering, 1(1), pp. 42–51.

Megantara, R.A., et al. (2022) 'Pengembangan dan implementasi Docker untuk memaksimalkan utilitas server universitas pada masa Covid-19', Jurnal, pp. 48–54.

Bediatra, M.R. (2023) Pengembangan dashboard untuk monitoring celah keamanan pada cluster Kubernetes menggunakan metode container images scanning. Skripsi. Universitas Islam Indonesia.

Irawan, A. (2024) Pembangunan mail server menggunakan Docker dan Mailu pada virtual private server (VPS) berbasis Linux. Skripsi. Program Studi Teknik Informatika.

Rokiman, Azindani, A. and Akhsani, I. (2025) 'Implementasi web server CasaOS menggunakan Docker dan Cloudflare Tunnels pada Linux Ubuntu', Informatic and Electronic Technology Journal, 1(1), pp. 9–15.

Afrizal, H. and Prihanto, A. (2022) 'Analisis kebutuhan resource dan independensi antara teknologi single server, virtualisasi, dan container', Journal of Informatics and Computer Science (JINACS), 4(1), pp. 26–33.

Nugraha, I.A., Data, M. and Siregar, R.A. (2018) 'Perancangan laboratorium komputer virtual mandiri untuk praktikum jaringan komputer dasar menggunakan Docker', Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 2(11), pp. 4442–4448.

Taqwiym, A. and Satria, R.R. (2025) 'Analysis and Development of a Web-Based Flower Board Rental Information System with the PIECES Approach', RISTIKA – Jurnal Riset Teknologi Informasi, Manajemen, dan Multimedia, 1(1), pp. 24–31.

Taqwiym, A. and Satria, R.R. (2025) 'Peningkatan Literasi Digital Hukum melalui Pelatihan Penggunaan PowerPoint untuk Analisis dan Presentasi Kasus', TIARA (Teknologi Informasi dan Rekayasa Aplikasi), 1(1), pp. 37–44.