

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi dalam beberapa tahun terakhir mendorong perubahan yang signifikan terhadap cara organisasi maupun individu membangun dan mengelola infrastruktur aplikasi. Adopsi teknologi *cloud computing* yang dipadukan dengan penerapan filosofi DevOps telah menciptakan kebutuhan akan sistem yang mampu menyediakan layanan secara cepat, fleksibel, dan mudah dikelola. Kondisi tersebut mendorong pemanfaatan teknologi kontainerisasi sebagai salah satu solusi untuk meningkatkan efisiensi pengelolaan aplikasi. Salah satu teknologi yang banyak digunakan adalah Docker, yaitu platform yang memungkinkan aplikasi beserta seluruh dependensinya dikemas ke dalam sebuah *container* sehingga dapat dijalankan secara konsisten pada berbagai lingkungan komputasi.

Dibandingkan dengan pendekatan virtualisasi berbasis *Virtual Machine* (VM), Docker menawarkan mekanisme yang lebih ringan karena tidak memerlukan sistem operasi terpisah pada setiap instans. Teknologi ini memanfaatkan kernel sistem operasi yang sama sehingga penggunaan sumber daya komputasi, seperti CPU dan RAM, menjadi lebih efisien. Selain memberikan efisiensi penggunaan sumber daya, Docker juga mampu meningkatkan kecepatan proses penyediaan layanan dan mempermudah proses distribusi aplikasi. Penelitian yang dilakukan oleh Bhardwaj et al. (2025) menunjukkan bahwa *container* berbasis Docker memiliki performa yang lebih baik dibandingkan *virtual machine* pada pengujian CPU, *disk I/O*, dan *server throughput*. Hasil penelitian tersebut juga memperlihatkan bahwa Docker menghasilkan *throughput* yang lebih tinggi serta latensi yang lebih rendah, sehingga mampu memanfaatkan sumber daya perangkat keras secara lebih optimal dibandingkan pendekatan virtualisasi tradisional (Bhardwaj et al., 2025).

Meskipun demikian, pemanfaatan Docker dalam lingkungan *cloud hosting* masih menghadapi beberapa kendala pada proses pengelolaannya. Sebagian besar

aktivitas administrasi, seperti pembuatan *container*, konfigurasi layanan, maupun proses *deployment*, masih dilakukan melalui *Command Line Interface* (CLI) yang membutuhkan pemahaman terhadap konfigurasi server dan berbagai perintah terminal. Kondisi tersebut dapat menyulitkan pengguna yang belum memiliki pengalaman dalam administrasi sistem sekaligus meningkatkan kemungkinan terjadinya kesalahan konfigurasi. Penelitian yang dilakukan oleh Amin et al. (2025) menunjukkan bahwa penerapan otomatisasi pada proses *deployment* mampu mengurangi waktu pengerjaan hingga 98,67% dibandingkan dengan metode manual (Amin et al., 2025).

Pengembangan antarmuka tersebut memerlukan teknologi yang mampu mendukung performa aplikasi secara optimal. Oleh karena itu, penelitian ini memanfaatkan Next.js sebagai *framework* pada sisi *frontend* dan Node.js pada sisi *backend*. Node.js memiliki mekanisme *event-driven* dan *non-blocking I/O* yang memungkinkan aplikasi menangani banyak proses secara bersamaan dengan penggunaan sumber daya yang efisien. Sementara itu, Next.js menyediakan dukungan terhadap Server-Side Rendering (SSR) dan Static Site Generation (SSG) sehingga mampu meningkatkan kecepatan pemuatan halaman sekaligus memberikan pengalaman pengguna yang lebih baik (A Jartarghar et al., 2022).

Penelitian ini dilaksanakan secara mandiri oleh penulis mulai dari proses identifikasi kebutuhan, perancangan antarmuka pengguna, implementasi sistem, hingga pengujian dan evaluasi. Seluruh proses pengembangan memanfaatkan perangkat lunak berbasis *open source* dengan lingkungan pengembangan yang dikonfigurasi secara mandiri. Pendekatan tersebut memberikan fleksibilitas dalam proses implementasi sekaligus memungkinkan setiap tahapan pengembangan dievaluasi dan disempurnakan secara berkelanjutan sesuai dengan prinsip yang diterapkan pada metode Incremental.

Pemanfaatan Docker dalam lingkungan *cloud* memberikan berbagai keuntungan dibandingkan pendekatan virtualisasi tradisional. Dengan mekanisme *containerization*, aplikasi dapat dijalankan secara terisolasi tanpa memerlukan sistem operasi terpisah pada setiap instance sehingga penggunaan sumber daya menjadi lebih efisien. Selain itu, Docker mempermudah proses *deployment*,

menjaga konsistensi lingkungan pengembangan hingga produksi, serta meningkatkan portabilitas aplikasi ketika dipindahkan antar server. Karakteristik tersebut menjadikan Docker banyak digunakan sebagai fondasi dalam pembangunan infrastruktur cloud modern.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah di paparkan, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana merancang arsitektur antarmuka pengguna berbasis *web* yang mendukung otomatisasi proses penyediaan layanan *cloud hosting* berbasis kontainer menggunakan teknologi Next.js dan Node.js?
2. Bagaimana mengembangkan mekanisme interaksi *real-time* antara pengguna dan lingkungan kontainer melalui implementasi protokol *WebSocket* pada antarmuka berbasis web?
3. Bagaimana menerapkan metode pengembangan perangkat lunak *Incremental* dalam proses pembangunan sistem untuk mendukung evaluasi bertahap dan minimalisasi risiko kegagalan sistem?

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. Menghasilkan antarmuka pengguna berbasis *web* yang mampu mengotomatisasi proses penyediaan layanan *cloud hosting* berbasis kontainer dengan memanfaatkan *framework* Next.js dan Node.js.
2. Menghasilkan fitur *web terminal* berbasis *browser* yang memungkinkan interaksi langsung dengan lingkungan kontainer secara *real-time* melalui implementasi protokol *WebSocket*.
3. Menghasilkan sistem pengelolaan *cloud hosting* berbasis kontainer yang dikembangkan menggunakan metode *incremental* untuk mendukung proses pengembangan yang terstruktur, terukur, dan mudah dievaluasi pada setiap tahapan.

1.4 Batasan Masalah

Agar penelitian ini lebih terfokus dan tidak menyimpang dari tujuan yang telah ditetapkan, maka diberikan batasan masalah sebagai berikut:

1. Sistem otomatisasi ini dikembangkan khusus untuk pengelolaan Docker Container, bukan untuk manajemen *Virtual Machine* (VM).
2. *Environment* atau server yang digunakan dalam penelitian ini berbasis sistem operasi Linux (Ubuntu/Debian).
3. Otomatisasi pembuatan kontainer hanya mendukung penggunaan Dockerfile (tidak mencakup orkestrasi kompleks seperti Kubernetes).
4. Fitur *Web Terminal* dibatasi untuk menjalankan perintah melalui docker exec ke dalam kontainer milik pengguna, bukan untuk mengontrol sistem operasi server utama (*host*).
5. Pembatasan sumber daya (*resource limiting*) dibatasi pada parameter CPU dan RAM sebagai representasi utama dalam mekanisme alokasi dan pengendalian sumber daya kontainer.
6. Sistem tidak membahas aspek keamanan jaringan tingkat lanjut seperti perlindungan terhadap serangan DDoS atau penetrasi *firewall* yang kompleks.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan kegunaan yang luas, baik secara teoritis maupun praktis bagi berbagai pihak yang terkait. Adapun manfaat dari penelitian ini adalah sebagai berikut:

1. Bagi Ilmu Pengetahuan

Secara teoritis, penelitian ini berkontribusi pada pengembangan metodologi perangkat lunak, khususnya dalam penerapan metode incremental pada sistem manajemen infrastruktur cloud. Selain itu, penelitian ini menjadi referensi ilmiah mengenai integrasi Next.js, Node.js, dan *Docker* dalam

pengembangan sistem otomatisasi *deployment* berbasis web, serta landasan bagi penelitian lanjutan di bidang *Container as a Service* (CaaS).

2. Bagi Institusi (Kampus)

Penelitian ini diharapkan menjadi referensi bagi mahasiswa Informatika, khususnya di bidang *Cloud Computing* dan *Full-stack Development*. Selain itu, hasil penelitian ini dapat mencerminkan kompetensi lulusan dalam penerapan teknologi modern seperti *Docker* dan NestJS.

3. Bagi Pengguna (Praktis)

Secara praktis, penelitian ini menawarkan solusi untuk mengoptimalkan proses *deployment* melalui otomatisasi, sehingga lebih cepat dan meminimalkan kesalahan konfigurasi. Selain itu, fitur *Web Terminal* memudahkan akses dan pemantauan kontainer langsung melalui *browser* tanpa bergantung pada akses SSH manual.

4. Bagi Peneliti (Diri Sendiri)

Penelitian ini menjadi sarana untuk menerapkan pengetahuan teoritis ke dalam solusi nyata. Selain itu, penelitian ini meningkatkan kemampuan teknis dalam pengelolaan lingkungan Linux, pengembangan *backend* dengan NestJS, serta pemahaman arsitektur jaringan pada teknologi kontainerisasi sebagai bekal memasuki dunia industri TI.

1.6 Jadwal Penelitian

NO	Kegiatan	Mar	Apr	Mei	Jun	Jul	Agu
1.	Penyusunan Proposal						
2.	Seminar Proposal						
3.	Pengumpulan Data & Analisis Kebutuhan Sistem						
4.	Perancangan & Pengembangan Aplikasi (<i>Incremental</i>)						
5.	Pengujian Aplikasi & Penyusunan Laporan						
6.	Seminar Hasil Penelitian						
7.	Sidang Akhir						

Tabel 1. 1 Tabel Jadwal Penelitian

1.7 Sistematika Penulisan

Sistematika penulisan laporan penelitian ini disusun secara sistematis yang terbagi ke dalam beberapa bab sebagai berikut:

BAB I PENDAHULUAN

Bab ini menguraikan latar belakang mengenai kebutuhan efisiensi dalam pengelolaan *cloud hosting*, identifikasi masalah terkait kompleksitas manajemen kontainer Docker, tujuan penelitian, batasan masalah, serta manfaat penelitian. Bab ini juga memuat kebaruan penelitian (*novelty*) yang membedakan platform ini dengan penelitian sebelumnya

BAB II LANDASAN TEORI

Bab ini menyajikan teori dan literatur yang menjadi landasan penelitian, mencakup konsep dasar *cloud computing*, teknologi kontainerisasi Docker, arsitektur *Server-Side Rendering* pada Next.js, serta penggunaan Node.js untuk integrasi sistem. Selain itu, bab ini memuat tabel *State of the Art* (SOTA) sebagai hasil tinjauan dari penelitian terdahulu.

BAB III ANALISIS DAN PERANCANGAN SISTEM

Bab ini menjelaskan tahapan pengembangan sistem menggunakan Metode *Incremental*. Di dalamnya diuraikan mengenai perangkat keras dan perangkat lunak yang digunakan, prosedur perancangan sistem otomatisasi, hingga jadwal pelaksanaan penelitian yang direncanakan.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi hasil implementasi sistem yang telah dirancang pada bab sebelumnya serta pembahasan mengenai fitur-fitur yang telah berhasil dikembangkan. Pada bab ini dijelaskan implementasi antarmuka pengguna, implementasi basis data, implementasi infrastruktur Docker, serta hasil pengujian sistem yang dilakukan untuk memastikan seluruh fungsi berjalan sesuai dengan kebutuhan yang telah ditentukan. Selain itu, bab ini juga membahas analisis hasil

pengujian terhadap fitur deployment aplikasi, monitoring container, dan web terminal yang menjadi bagian utama dari sistem.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang diperoleh berdasarkan hasil perancangan, implementasi, dan pengujian sistem yang telah dilakukan. Kesimpulan disusun untuk menjawab tujuan penelitian yang telah ditetapkan pada bab pendahuluan. Selain itu, bab ini juga memuat saran yang dapat digunakan sebagai bahan pengembangan sistem pada penelitian selanjutnya, baik dari aspek fitur, performa, keamanan, maupun pengembangan teknologi yang digunakan dalam platform.