B AB I
PENDAHULUAN

1.1 Latar Belakang

Meningkatnya teknologi komputer dan jaringan, serta permintaan akan informasi di internet, menyebabkan lalu lintas di situs web meningkat. Ini menyebabkan situs web mengalami kelebihan beban, dan salah satu cara untuk mengatasi masalah tersebut adalah dengan mengelompokkan server bersama untuk bekerja sebagai satu sistem. Salah satu jenis *clustering* adalah *load balancing*. *Load balancing* adalah cara untuk membagi pekerjaan antara beberapa server sehingga tidak ada satu server pun yang kelebihan beban. Ini dapat membantu meningkatkan kecepatan situs web, dan menghindari masalah dengan waktu respons. *Load balancing* menggunakan aplikasi *open source* seperti *Nginx* dan *Docker* dapat memiliki fitur yang berbeda. Nginx sebagai load balancer merupakan salah satu web server dan *load balancer* yang populer.

Dalam konteks ini, *Nginx* digunakan sebagai *load balancer* untuk mendistribusikan lalu lintas *website company profile* ke beberapa server *backend*. *Nginx* dapat membantu meningkatkan performa dan kehandalan dengan membagi beban lalu lintas secara merata. *Docker* adalah platform *open-source* yang memungkinkan pengemasan, pengiriman, dan menjalankan aplikasi dalam wadah yang terisolasi. Dalam kasus ini, *Docker* digunakan untuk mengelola dan menjalankan *instance nginx load balancer* serta server *backend* dengan cepat dan mudah. analisis performa *load balancer* melibatkan evaluasi terhadap kinerja *nginx* dalam membagi beban lalu lintas dan mengoptimalkan *throughput, latency,* dan

waktu respons. Pemantauan performa dapat melibatkan pengukuran metrik seperti jumlah permintaan per detik, waktu respons rata-rata, penggunaan sumber daya (CPU, RAM), dan sebagainya.

Analisis juga dapat melibatkan evaluasi skalabilitas sistem, yaitu kemampuan *load balancer* untuk menangani lonjakan lalu lintas dengan efektif. Dalam skenario ini, penggunaan *Docker* memudahkan replikasi dan penambahan *instance nginx* atau server *backend* sesuai kebutuhan. Evaluasi keandalan *load balancer* melibatkan pemantauan *uptime* dan deteksi kegagalan. *Docker* dapat memfasilitasi manajemen failover dan penggantian instansi yang gagal dengan cepat, meminimalkan dampak pada pengguna akhir. Analisis performa akan mencakup peninjauan konfigurasi *nginx* dan *Docker* yang digunakan untuk mengelola *load balancing*. Hal ini meliputi pengaturan *nginx*, strategi penyeimbangan beban, protokol komunikasi, dan manajemen skala *Docker*.

Nginx adalah perangkat lunak server web sumber terbuka gratis yang dapat melakukan lebih dari sekadar melayani halaman web HTTP. Itu dapat bertindak sebagai *proxy*, penyeimbang beban, dan server email. Ini sangat cepat, dan dapat menangani banyak lalu lintas. Untuk menyiapkan penyeimbang muatan menggunakan *nginx*, anda perlu menggunakan beberapa mesin *Docker* (Mardi Nurzaman et al., 2022). *Docker* adalah layanan yang menyediakan kemampuan untuk mengemas dan menjalankan aplikasi di lingkungan terpisah yang disebut wadah. Dari kelebihan-kelebihan yang telah dijelaskan di atas, dapat disimpulkan bahwa *Docker* dapat membantu meningkatkan produktivitas *developer* saat membuat *software* berkualitas tinggi (Setiawan, 2021).

Company Profile adalah gambaran umum mengenai perusahaan dan biasanya bertujuan untuk memberi tahu kepada audiens terkait produk atau layanan yang ditawarkan. Tidak hanya itu, komponen ini biasanya berisi tentang kisah mulainya suatu perusahaan, visi-misi, serta memberi tahu kepada audiens terkait alasan menjual produk atau layanan. Secara luas, *company profile* bertujuan untuk memberi tahu keberadaan sebuah perusahaan dengan informasi-informasi terperinci. Dengan demikian, *company profile* dapat meningkatkan *awareness* pelanggan serta menarik perhatiannya, tujuan dari dibuatnya *company profile* adalah untuk memberi tahu informasi terkait perusahaan kepada audiens atau pengunjung, (Adieb, 2022).

1.2 Rumusan masalah

1. Bagaimana kinerja *load balancer nginx* dalam membagi beban lalu lintas ke server *backend*?
2. Bagaimana tingkat latensi (waktu respons rata-rata) dari *load balancer nginx*?

1.3 Batasan masalah

Fokus pembahasan pada penelitian ini adalah:

1. Analisis hanya akan dilakukan terhadap penggunaan *load balancer nginx* dengan menggunakan *Docker*, dan tidak melibatkan teknologi liad *balancing* lainnya.
2. Analisis hanya akan dilakukan pada *website company profile*, dan tidak melibatkan *website* lain atau aplikasi yang berbeda.

3. Analisis performa hanya akan dilakukan pada sisi server dan infrastuktur *website*, dan tidak akan mengevaluasi performa aplikasi atau *website* secara keseluruhan.
4. Analisis akan fokus pada peningkatan performa dan ketersediaan *website*, dan tidak akan mengevaluasi aspek keamanan atau keandalan sistem secara detail.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah untuk menganalisis performa *load balancer nginx* menggunakan *Docker* pada *website company profile*. Beberapa tujuan dari penelitian ini adalah:

1. Menentukan efektivitas penggunaan *Docker* dalam konfigurasi *load balancer nginx* untuk *website company priofiel*.
2. Menganalisis kinerja *load balancer nginx* dalam mengelola lalu lintas situs web, termasuk waktu tanggap, waktu *loading*, dan kecepatan akses.
3. Memberikan pandian dan saran kepada pengembang dan *administrator* sistem yang ingin mengoptimalkan performa *load balancer nginx* pada lingkungan *Docker* untuk *website company profile*.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini antara lain:

1. Meningkatkan performa situs web, dengan menganalisis performa *load balancer nginx* pada lingkungan *Docker*, organisasi dapat mengidentifikasi

masalah dalam konfigurasi mereka dan melakukan perbaikan untuk meningkatkan performa situs web.

2. Memperbaiki pengalaman pengguna, dengan meningkatkan performa situs web, pengalaman pengguna dapat ditingkatkan. Hal ini dapat meningkatkan loyalitas pengguna dan meingkatkan konversi pada situs web.
3. Meningkatkan efesiensi pemeliharaan konfigurasi *load balancer nginx* dalam kontrainer *Docker* dapat lebih mudah dipelihara daripada konfigurasi *load balancer* tradisional. Dalam *scenario Docker*, pemeliharaan adapat dilakukan pada tingkat kontrainer, sehingga memungkinkan pemeliharaan yang lebih efesien dan isolasi masalah.