BAB II
LANDASAN TEORI

2.1. Tinjauan Pustaka

Untuk mendukung penelitian ini, penulis menggunakan beberapa *Literature* yang berkaitan dengan judul dan pokok bahasan pada penelitian. Adapun *Literature* yang dipergunakan dapat ditinjau pada Tabel 2.1.

Tabel 2.1. Literature Review

No. ***Literature***	**Penulis, Tahun**	**Judul**
Literature 01	(Dimas Setiawan Afis, Mahendra Data, Widhi Yahya. 2021)	"*Load Balancing Server Web* Berdasarkan Jumlah Koneksi Klien Pada *Docker Swarm*"
Literature 02	(Stefanus Eko Prasetyo, Agung Wijaya. 2021)	"Analisis *Load Balancing* Menggunakan *Docker Swarm*"
Literature 03	(Destriawan, Destriawan. 2021)	"Analisis dan Implementasi *Performance Load Balancing Web Server* pada *Nginx* menggunakan *Docker*"
Literature 04	(Fernando Mardi Nurzaman, Ferdi	"Analisis Perbandingan Performa *Load Balancer*

	Cahyadi, Muhamad Radzi Rathomi. 2022)	*Nginx* dan *Haproxy* pada *Docker*"
Literatur 05	(Stefanus Eko Prasetyo, Yulfan salimin. 2021)	"Analisis Perbandingan Performa *Web Server Docker Swarm* dengan *Kubernetes Cluster*"

2.1.1 Tinjauan Terhadap Literatur 01

Berdasarkan penalitian yang telah dilakukan oleh (Dimas Setiawan Afis, Mahendra ata, Widhi Yahya. 2021) dengan judul, "*Load Balancing* server web berdasarkan jumlah koneksi klien pada *Docker Swarm*". Melalui pengujian kami mengetahui apakah sistem yang dibuat memenuhi persyaratan dan kami mengetahui apakah mekanisme yang diterapkan dapat bekerja dengan baik atau tidak. Terdapat 3 tahap pengujian yaitu pengujian kinerja, pengujian *load balancing*, pengujian fungsionalitas sistem.

Uji kinerja membandingkan kinerja penyeimbang beban menggunakan metode transfer data baca dan melingkar. Tes *load balancing* digunakan untuk mengetahui apakah algoritma yang digunakan untuk *load balancing* bekerja dengan benar atau tidak. Tes penyeimbangan beban dilakukan dengan menganalisis paket permintaan yang masuk dengan *Wireshark*. Berdasarkan hasil yang diperoleh, *load balancing* menggunakan algoritma *round-robin* dapat mendistribusikan beban permintaan dengan benar, sedangkan algoritma *lessconn* tidak bisa.

Pengujian fungsionalitas sistem dilakukan dengan menguji mekanisme failover, yang merupakan mekanisme di *Docker Swarm*. Tes ini dilakukan oleh

salah satu *swarm host*. Hasil *throughput load index* yang menggunakan *least connection* dan algoritma *round robin* memiliki hasil yang berbeda. *Load balancer* yang menerapkan setidaknya algoritma *Conn* memberikan hasil kinerja yang lebih baik daripada algoritma *round-robin*, (Dimas Setiawan Afis, Mahendra Data, 2021).

2.1.2 Tinjauan Terhadap Literatur 02

Berdasarkan penelitian yang telah dilakukan oleh (Strfanus Eko Prasetyo, agung Wijaya. 2021) dengan judul, " Analisis *Load Balancing* Mengguanakan *Docker Swarm*". *Load balancing* adalah metode yang digunakan untuk memaksimalkan *throughput*, mengoptimalkan *traffic*, meminimalkan *latency*, dan menghindari kemacetan pada sebuah link. Eksperimen ini menggunakan wadah *Docker Swarm*, yang merupakan wadah virtualisasi beban seimbang. Variabel yang digunakan dalam pengujian *load balancing* ini adalah *response time*. Hasil yang diperoleh setelah pengujian menunjukkan bahwa *load balancing* yang disertakan dalam virtualisasi *swarm Container Docker* dapat bekerja dengan baik tanpa gangguan, (A. W. Stefanus Eko Prasetyo, 2021).

2.1.3 Tinjauan Terhadap Literatur 03

Berdasarkan penelitian yang telah dilakukan oleh (Destriawan, 2021) yang berjudul, "Analisis dan Implementasi *Performance Load Balancing* Web Server pada *Nginx* menggunakan *Docker*", Penulis mencakup penyeimbangan muatan antara memecah server web yang kelebihan beban, merancang pengalaman pengguna yang optimal, dan menangani permintaan melalui berbagai layanan dengan sumber daya yang efisien. Poin utama dari *load balancing* adalah *load*

balancing dinamis antara node sesuai dengan kebutuhan *klien*. Saat ini, beberapa aplikasi memiliki jutaan pengguna, karena pengguna terlalu banyak, server sulit ditangani, 2 begitu banyak pengembang aplikasi membutuhkan komputasi terdistribusi, dan teknologi *load balancing* adalah salah satu solusinya. Secara umum, arsitektur saat ini hanya berfokus pada perluasan kemampuan server web menggunakan server *back-end* tunggal.

Maka, penelitian ini mengimplementasikan dua model *load balancing* yang berbeda yaitu *cluster* dan replikasi di *Docker*. Keuntungan *Docker* adalah banyak aplikasi web dapat dikelola lebih efisien dibandingkan dengan mesin virtual. Secara umum, *Docker* menggunakan fitur sistem operasi *Linux* melalui proses daemonnya untuk membuat wadah yang mudah diakses dan persisten. *Docker* menawarkan beberapa keunggulan untuk perangkat lunak. Kemungkinan untuk merepresentasikan tugas mesin virtual dan memfasilitasi bekerja dengan berbagai *prototipe* perangkat lunak menggunakan gambar *nginx* dan file terkait yang tersedia dalam wadah dan diisolasi, (Destriawan, 2021).

2.1.4 Tinjauan Terhadap Literatur 04

Pada penelitian yang dilakukan oleh (Fernando Mardi Nurzaman, Ferdi Cahyadi, Muhamad Radzi Rathomi. 2022) yang berjudul Analisis Perbandingan Performa *Load Bancer Nginx* dan *Haproxy* pada *Docker*, Penulis mengajukan pertanyaan tentang kinerja yang baik dari *Nginx* dan *Haproxy* selama penyeimbangan muatan dengan *Docker machine* yang terhubung melalui *Docker Swarm*. Ini diimplementasikan dengan *Virtual Private Server (VPS)* dan *Docker Swarm* sebagai koneksi antara mesin *Docker* menggunakan *Nginx* dan *Haproxy* sebagai penyeimbang beban.

Penelitian ini dilakukan dengan menguji kinerja *load balancer Nginx* dan *Haproxy* di *Docker* menggunakan alat dari *Webtreestools* bernama *Httperf*. Serta membandingkan penyeimbang beban *Nginx* dan *Haproxy* dengan parameter yang diamati pada komputer tradisional seperti kinerja, waktu *respons*, CPU. penggunaan dan penggunaan penyimpanan. Analisis yang dilakukan hanya pengujian performa dari hasil pengujian performa, *response time* dan parameter CPU, (Mardi Nurzaman et al., 2022).

2.1.5 Tinjauan Terhadap Literatur 05

Pada penelitian yang dilakaukan oleh (Stefanus Eko Prasetyo, Yulfan salimin. 2021) dengan judul Analisis Perbandingan Performa *Web Server Docker Swarm* dengan *Kubernetes Cluster*, panalitian ini dilakukan mengukur respon server dalam berbagai kondisi *load*. Parameter pengukuran respon server yang digunakan dalam load testing ini adalah sebagai berikut:

1. *Response Time:* Jumlah total waktu yang diperlukan untuk menanggapi permintaan layanan. *Response time* terbagi menjadi tiga, *Average* -waktu rata-rata dari serangkaian hasil, Min -waktu terpendek yang *client* tunggu agar server merespon, Max -waktu terlama yang *client* tunggu agar server merespon.
2. *Standard Deviation:* Kestabilan *response time*, bisa juga dikatakan dengan bagaimana rata-rata waktu respon tersebar.
3. *Error:* Persentase permintaan yang tidak terpenuhi
4. *Throughput*: Jumlah permintaan yang diproses per detik

5. CPU *Utilization:* Penggunaansumber daya CPU komputer, atau jumlah pekerjaan yang ditangani oleh CPU pada untuk menanggapi permintaan layanan, (Y. salimin Stefanus Eko Prasetyo, 2021).

2.2. Keaslian Penelitian

Adapun beberapa hal yang menjadi pembeda antara penelitia yang dilakukan penulis dengan penelitian yang sudah dilakukan sebelumnya. Penelitian ini berfokus pada bagaimana performa dari *Load Balancer Nginx* dalam melakukan *load Balancer* menggunakan *Docker Contrainer* pada *Website Company Profile.*

2.3. *Load Balancing*

pengertian *load balancing* adalah suatu proses penyaluran trafik ke dalam sejumlah server. Sehingga, dengan menggunakan *load balancing*, kinerja jaringan pun akan menjadi jauh lebih stabil. Hal ini dikarenakan beban trafik tersebut tidak hanya ditanggung oleh salah satu server saja, melainkan terbagi secara merata. Adapun tujuan utama *load balancing* adalah untuk mengoptimalkan kinerja sekaligus mengatasi berbagai gangguan pada database, aplikasi, maupun situs web kamu.Penggunaan *load balancing* adalah hal yang sangat tepat, memiliki website dengan trafik jaringan cukup tinggi. Sebab, *load balancing* akan membantu server kamu tetap merespon cepat dan juga terhindar dari kendala atau *overloading*, (Hosting, 2017).

2.4. *Nginx*

Nginx adalah web server dengan performa yang andal dan mempunyai beberapa fitur canggih lain yang mudah dikonfigurasi. Alhasil, *Nginx* mampu

membuat *website* Anda lebih *powerful* dan canggih. Pada awal munculnya, *Nginx* hanya dipakai untuk server HTTP saja. Seiring perkembangan teknologinya, sekarang web server *Nginx* juga dipakai sebagai HTTP *cache, load balancer* (HTTP, TCP, dan UDP), dan server *proxy* (IMAP, POP3, dan SMTP). Selain kemampuan di atas, web server *Nginx* juga dapat berjalan di berbagai macam sistem operasi, seperti Linux, Mac OS X, HP-UX, BSD Varian, dan Solaris. Web server *Nginx* adalah web server yang dipakai di berbagai perusahaan besar, di antaranya *Atlassian, Intuit, T-Mobile, GitLab, Microsoft, Google, Adobe, LinkedIn, Facebook, Twitter, Apple*, dan masih banyak lainnya, (K, 2019).

2.5. *Docker*

Docker adalah platform untuk mengembangkan, mengirim, dan menjalan aplikasi yang mulai berkembang pada tahun 2013. *Docker* memungkinkan pengembang perangkat lunak untuk memisahkan aplikasi dari infrastruktur sehingga pengembang dapat mengirimkan perangkat lunak dengan cepat. *Docker* dapat membantu pengembang untuk mengelola infrastruktur teknis dengan cara yang sama seperti saat pengembang mengelola aplikasi. Pemanfaatan metode *Docker* untuk pengiriman, pengujian, dan penerapan kode dengan cepat. Selain itu, pengembang dapat secara signifikan mengurangi penundaan antara penulisan kode dan proses menjalankannya dalam produksi perangkat lunak Secara umum, *Docker* adalah proyek open *source* yang menyediakan platform terbuka untuk para *website developer* atau pengembang agar dapat membangun,mengemas, dan menjalankan aplikasi di berbagai lokasi dan bertindak sebagai sebuah *Container* yang ringan. *Docker* memiliki fungsi utama dalam penyederhanaan konfigurasi yang dibangun berdasarkan teknologi *Container*, (Raharja, 2021).

2.6. *Company Profile*

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).

2.7. *Port*

Dalam protokol jaringan TCP/IP, sebuah porta adalah mekanisme yang mengizinkan sebuah komputer untuk mendukung beberapa sesi koneksi dengan komputer lainnya dan program di dalam jaringan. Porta dapat mengidentifikasikan aplikasi dan layanan yang menggunakan koneksi di dalam jaringan TCP/IP (Santoso et al., 2008).

2.8. *Apache JMeter*

Apache JMeterTM adalah aplikasi *open source* berbasis Java yang dapat dipergunakan untuk *performance test*. Bagi seorang *QA Engineer JMeter* bisa digunakan untuk melakukan *load/stress testing* Web *Application*, *FTP Application* dan *Database server test.* JMeter bisa dijalankan dengan 2 cara, dengan GUI atau non-GUI (Command line). Untuk *beginner* lebih baik saya sarankan menggunakan

cara yang pertama. Mudah dan tanpa melakukan *scripting* tertentu. Tinggal membuat Test Plan, mengisikan berapa *thread & sample* yang akan diujicobakan, *running* dan menganalisa hasil/report (Yogi, 2018).

2.9. *VSCode*

Visual Studio Code adalah kode editor sumber yang dikembangkan oleh *Microsoft* untuk Windows, *Linux* dan *macOS.* Ini termasuk dukungan untuk debugging, kontrol git yang tertanam dan GitHub, penyorotan sintaksis, penyelesaian kode cerdas, *snippet*, dan *refactoring* kode. Ini sangat dapat disesuaikan, memungkinkan pengguna untuk mengubah tema, pintasan *keyboard*, preferensi, dan menginstal ekstensi yang menambah fungsionalitas tambahan (Agustini & Kurniawan, 2019).

2.10. *Prometheus*

Prometheus adalah *toolkit* pemantauan dan peringatan sistem sumber terbuka yang awalnya dibuat di *SoundCloud.* Sejak dimulai pada 2012, banyak perusahaan dan organisasi telah mengadopsi *Prometheus,* dan proyek ini memiliki komunitas pengembang dan pengguna yang sangat aktif. Sekarang proyek *open source* mandiri dan dikelola secara independen dari perusahaan mana pun. Untuk menekankan hal ini, dan untuk memperjelas struktur tata kelola proyek, *Prometheus* bergabung dengan *Cloud Native Computing Foundation* pada 2016 sebagai proyek yang dihosting kedua, setelah *Kubernetes* (Febriana & Jakarta, 2020).

Prometheus memiliki komponen utama yang disebut Prometheus server. Sebagai layanan *monitoring*, *Prometheus* server memantau hal tertentu. Dalam

kegiatan *monitoring*, hal-hal yang *dimonitoring* oleh *Prometheus* disebut Target. Target dapat merujuk ke berbagai hal, bisa berupa server tunggal atau target bisa berupa *endpoint* yang melalui HTTP,HTTPS, DNS,TCP dan ICMP. *Prometheus* server mengambil data target pada interval yang telah ditentukan untuk mengumpulkan *metric* dari target tertentu dan menyimpannya dalam *timeseries database* (Febriana & Jakarta, 2020).

2.11. *Alert Manager*

Alert Manager adalah biner tunggal yang menangani peringatan yang dikirim oleh server *prometheus* dan memberi tahu pengguna akhir melalui E-mail, *slack*, atau alat lainnya . *Alert Manager* memiliki tugas untuk memberikan notifikasi ke *grafana*, jika terjadi sesuatu hal yang melebihi batas yang dapat membuat *server down.*

2.12. *Metrik*

Metrik merupakan sebuah mekanisme yang berguna untuk meningkatkan kualitas dari produk *software* dan juga untuk menentukan cara yang paling tepat untuk membantu praktisi dan peneliti. Selain itu juga sebagai kompleksitas *database* sehingga ketahanan dan keberlangsungan *database* bisa terjaga (Wulandari et al., 2020).

2.13. *Nginx Exsporter*

Nginx Exporter berfungsi untuk memperlihatkan beberapa metrik melalui halaman stub_status. *Nginx Exporter* bekerja untuk *Prometheus* dengan mengambil metrik dari *nginx*, mengonversi metrik menjadi jenis metrik *Prometheus* yang

sesuai, dan terakhir memaparkannya melalui server HTTP untuk dikumpulkan oleh *Prometheus.*

2.14. *Node Exsporter*

Node Exporter memiliki fungsi yang sama dengan *Nginx Exporter* yaitu memaparkan berbagai macam metrik terkait perangkat keras dan kernel. Perbedaan dari *Node Exporter* ini mengambil *metrik* secara umum atau keseluruhan berbeda dengan *Nginx Exporter* yang dapat mengambil metrik *Nginx* dengan lebih detail dan terfokus.

2.15. *Grafana*

Grafana adalah analitik sumber terbuka *multi-platform* dan aplikasi web visualisasi interaktif. Ini memberikan bagan, grafik, dan peringatan untuk web saat terhubung ke sumber data yang didukung. Grafana juga berfungi sebagai alat visualisasi dalam bentuk grafik dari data rangkuman yang diberikan oleh *promethous.*

Untuk melengkapi system *monitoring* ini diperlukan visualisasi. *Grafana* dapat membuat visualisasi dari data yang telah diproses. Hasil penelitian ini menunjukan *device* yang terhubung didalam jaringan dapat ditampilkan *Grafana* (Febriana & Jakarta, 2020).

2.16. *Makefile*

Makefile adalah file yang berisi serangkaian arahan yang digunakan oleh alat otomatisasi make *build* untuk menghasilkan target / sasaran. File deskripsi

(*makefile*) berisi deskripsi hubungan antar file, dan perintah yang perlu dijalankan untuk memperbarui target untuk mencerminkan perubahan dalam prasyaratnya. Setiap spesifikasi, atau aturan, harus terdiri dari target, prasyarat opsional, dan perintah opsional yang harus dijalankan ketika prasyarat lebih baru daripada target (Ramadhika Dwi Poetra, 2019).

2.17. *Vegeta*

Vegeta adalah alat pengujian beban HTTP yang ditulis dalam *Go* yang dapat digunakan sebagai perintah dalam antarmuka baris perintah atau sebagai Pustaka. Program menguji bagaimana perilaku aplikasi berbasis HTTP ketika banyak pengguna mengaksesnya pada saat yang sama, dengan menghasilkan beban latar belakang dari permintaan GET. *Vegeta* digunakan untuk menghasilkan jumlah permintaan yang konstan dan berkelanjutan per detik untuk mengetahui berapa lama suatu layanan dapat mempertahankan beban puncak sebelum menurun kinerjanya.

2.18. Metode Experimen

Metode eksperimental memberikan peluang bagi individu dan Suatu kelompok yang percobaannya sengaja direncanakan dan dirancang untuk membuktikan kebenaran suatu teori dengan mengikuti jalur yang teratur dan mempraktikkannya dan sistematis. (Rismawati, Ratman dan Dewi, 2006). Penelitian Eksperimental dilakukan untuk mengetahui efek pengobatan atau perawatan topik penelitian (Cahya, 2013). Selain itu, prosedur eksperimental dilakukan untuk membuat informasi yang diperlukan parameter yang ditentukan.

Prosedur eksperimental dilakukan dengan cara tertentu Melakukan tes langsung di media (Hasim dan Riadi, 2013).

Sifat-sifat metode eksperimen (Rismawati, Ratman dan Dewi, 2006) sebagai berikut:

1) Metode untuk membuat percobaan, pengamatan dan kesimpulan sesuatu untuk dicoba.
2) Metode pengembangan pengetahuan.
3) Metode yang membantu dalam pemrosesan informasi secara aktif.
4) Metode pembelajaran yang mengarah pada lingkungan belajar sebagai suatu permasalahan Teknologi.
5) Metode pemecahan masalah ilmiah.

2.19. Metode Pengujian

Pada tahap ini metode yang dilakukan pada penelitian ini menggunakan teknik kualitatif berdasarkan aplikasi yang digunakan untuk menguji performa *load balancer nginx* menggunakan *Docker* pada sebuah Menurut Moleong (2005:6), penelitian kualitatif adalah penelitian yang bermaksud untuk memahami fenomena tentang apa yang dialami oleh subjek penelitian misalnya perilaku, persepsi, motivasi, tindakan, dll secara *holistic*, dan dengan cara deskripsi dalam bentuk kata-kata dan bahasa, pada suatu konteks khusus yang alamiah dan dengan memanfaatkan berbagai metode alamiah (Moleong, 2005).