Implementasi Cluster Container dengan Docker Swarm pada Infrastruktur Microsoft Azure

Angelie Rachmadhani Aulia
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
angellraa@student.telkomuniversity.ac.id

Muhammad Iqbal
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
miqbal@telkomuniversity.ac.id

Agus Ganda Permana
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
gandapermana@telkomuniversity.ac.id

***Abstrak* — Kebutuhan akan layanan web server yang dapat diskalakan dan memiliki ketersediaan tinggi mendorong pemanfaatan teknologi orkestrasi container. Penelitian ini mengimplementasikan cluster container menggunakan Docker Swarm pada infrastruktur cloud Microsoft Azure dengan konfigurasi satu node manajer dan tiga node pekerja. Layanan NGINX dijalankan dalam container dan direplikasi untuk menguji skalabilitas, kinerja, serta ketersediaan. Pengujian dilakukan menggunakan Apache JMeter dengan tiga skenario replika (5, 10, dan 15) serta empat variasi beban permintaan (50, 100, 150, dan 200 request/detik). Parameter yang diukur meliputi response time, error rate, dan throughput. Hasil menunjukkan bahwa peningkatan replika dari 5 menjadi 15 mampu meningkatkan throughput rata-rata dari 93 menjadi 108 request/detik, dengan error rate tetap 0% meskipun beban mencapai 2000 request/detik. Response time juga relatif stabil di bawah 160 ms pada sebagian besar variasi beban. Selain itu, sistem mampu menjaga ketersediaan melalui pemindahan container otomatis saat terjadi perubahan status node, menunjukkan keandalan Docker Swarm dalam mengelola layanan berbasis container pada lingkungan cloud.**

***Kata kunci*— docker, swarm, replika, throughput, response time**

I. PENDAHULUAN

Kemajuan teknologi menuntut infrastruktur jaringan yang mampu menyediakan skalabilitas dan ketersediaan tinggi. Cloud Computing menjadi solusi dengan menyediakan akses data dan aplikasi melalui internet tanpa instalasi lokal [1]. Microsoft Azure merupakan salah satu platform cloud yang banyak digunakan karena fleksibilitas pengelolaan infrastruktur virtual. Namun, lonjakan permintaan pengguna dapat meningkatkan beban server, sehingga dibutuhkan solusi efisien untuk mempertahankan kinerja.

Teknologi container menawarkan lingkungan terisolasi yang ringan dan konsisten, sehingga proses deployment layanan web lebih cepat dibandingkan virtual machine [2]. Untuk skala besar, container memerlukan sistem orkestrasi guna memastikan ketersediaan dan meminimalkan kesalahan. Docker sebagai platform open source mampu mengotomatisasi deployment aplikasi [3], sementara Docker Swarm sebagai pengembangannya memungkinkan pengelolaan cluster container melalui konsep node manajer dan node pekerja, serta mendukung penambahan node dalam jumlah besar untuk pengujian berskala dan kondisi mendekati nyata [4].

II. KAJIAN TEORI

Kajian teori ini membahas konsep-konsep yang menjadi dasar penelitian, meliputi teknologi cluster container, infrastruktur cloud, dan pengujian performa layanan web.

A. Cluster Container dengan Docker Swarm

Artikel Cluster container merupakan sekumpulan container yang dikelola secara terdistribusi untuk menjalankan aplikasi beserta dependensinya dalam lingkungan terisolasi. Platform orkestrasi seperti Kubernetes dan Docker Swarm digunakan untuk mengatur interaksi, distribusi beban kerja, serta ketersediaan layanan [2].

Docker Swarm adalah alat orkestrasi open source yang mengelola dan menghubungkan node dalam sebuah cluster. Fitur utamanya meliputi keamanan, skalabilitas, pemeliharaan, dan kemampuan melakukan pemulihan otomatis jika terjadi kegagalan container. Arsitektur Docker Swarm terdiri dari manager node yang bertugas mengatur lalu lintas permintaan klien, serta worker node yang mengeksekusi layanan. Permintaan klien diarahkan ke manager node dan didistribusikan secara merata ke worker node melalui mekanisme internal load balancing [5].

B. Infrastruktur Microsoft Azure

Cloud computing merupakan model komputasi yang memungkinkan akses jaringan sesuai permintaan terhadap sumber daya komputasi yang dikelola penyedia layanan [6]. Web server berperan menerima permintaan dari klien, memprosesnya, dan mengirimkan respons berupa halaman web atau data melalui protokol HTTP/HTTPS [7].

Microsoft Azure adalah platform cloud yang menyediakan layanan Platform as a Service (PaaS) untuk membangun, mengelola, dan menghosting aplikasi web [8].

Microsoft Azure menawarkan fleksibilitas, skalabilitas, dan keamanan tinggi tanpa memerlukan pengelolaan perangkat keras secara langsung.

C. Response Time

Response time adalah waktu yang dibutuhkan server untuk merespons permintaan klien sejak permintaan dikirim hingga respons diterima [8].

D. Error Rate

Error rate merupakan persentase kegagalan permintaan selama pengujian, dihitung dari perbandingan antara jumlah permintaan gagal dan total permintaan [9].

E. Throughput

Jumlah permintaan yang diproses server per detik, diukur dalam satuan *request per second* (rps) atau *bits per second* [9].

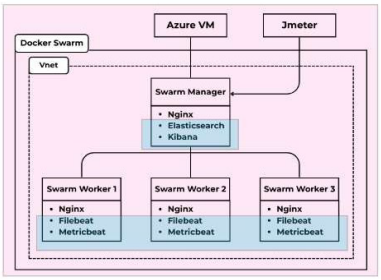
F. Apache JMeter

Apache JMeter adalah sebuah perangkat lunak yang digunakan untuk melakukan pengujian performa dengan mensimulasikan beban kerja pada server atau sistem [8].

III. PERANCANGAN SISTEM

Memberikan gambaran rancangan penelitian yang meliputi prosedur atau langkah-langkah penelitian, waktu penelitian, sumber data, cara perolehan data dan menjelaskan metode yang akan digunakan dalam penelitian.

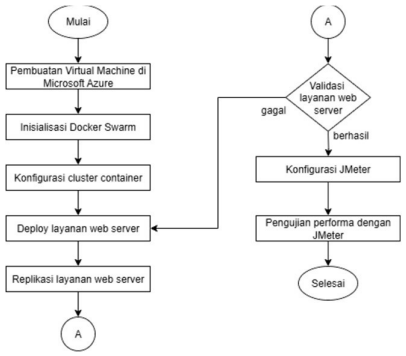
A. Arsitektur Sistem



GAMBAR 1
Arsitektur Sistem

Sistem penelitian ini dibangun pada infrastruktur cloud Microsoft Azure dengan konsep cluster container menggunakan Docker Swarm sebagai orkestrator. Empat virtual machine dikonfigurasi dalam satu virtual network, terdiri dari satu manager node dan tiga worker node. Layanan web server NGINX dijalankan secara terdistribusi melalui mekanisme replikasi untuk mendukung skalabilitas. Pengujian dilakukan menggunakan Apache JMeter yang diarahkan ke manager node untuk mengukur performa NGINX pada berbagai skenario jumlah replika dan variasi beban permintaan.

B. Alur Perancangan Sistem



GAMBAR 2
Alur Perancangan Sistem

Proses penelitian diawali dengan identifikasi masalah, penentuan perangkat lunak, dan perancangan arsitektur sistem. Selanjutnya dibuat empat virtual machine di Microsoft Azure dalam satu resource group dan virtual network, terdiri dari satu manager node dan tiga worker node, disertai konfigurasi network security group. Setelah itu, Docker diinstal pada seluruh node dan Docker Swarm diinisialisasi pada manager node.

Tiga worker node ditambahkan ke dalam cluster dan melakukan validasi koneksi. Layanan web server NGINX kemudian dideploy secara terdistribusi di seluruh node, dilanjutkan replikasi untuk menjalankan layanan pada masing-masing worker node. Validasi dilakukan dengan mengakses layanan melalui peramban. Apache JMeter kemudian dikonfigurasi untuk membuat skenario pengujian dengan variasi beban permintaan, dan pengujian dijalankan untuk memperoleh parameter response time, throughput, dan error rate.

IV. HASIL DAN PEMBAHASAN

A. Hasil Swarm

Cluster Container yang berhasil dibuat dapat divalidasi sehingga dapat terlihat anggota cluster yang terdiri dari satu manager node dan tiga worker node. Status dan peran dari seluruh node di dalam cluster dapat terlihat melalui proses validasi.

```
root@DockerManager:/home/swarm# docker node ls
ID                            HOSTNAME        STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
qtvmh6ydnh89nsbomuox5lmbp *   DockerManager   Ready     Active         Leader           28.3.3
reidkmmv4cpbqmlfxil8ad3fn     worker1         Ready     Active                          28.3.3
03x18s1cxzmss6qzl2k4mneim     worker2         Ready     Active                          28.3.3
uj97f7x8y5rw0lp07n9v3g4tq     worker3         Ready     Active                          28.3.3
```

GAMBAR 3
Anggota Swarm Cluster

B. Hasil Deployment dan Replikasi Sistem Layanan Web

Layanan web server NGINX direplikasi sebanyak enam replika yang tersebar di dalam container yang ada pada seluruh node berdasarkan distribusi internal Docker Swarm.

```
root@DockerManager:/home/swarm# docker service ps nginx_web
ID             NAME          IMAGE          NODE          DESIRED STATE   CURRENT STATE               ERROR     PORTS
ylu97bryx8ud   nginx_web.1   nginx:latest   worker1       Running         Running about a minute ago
mdm6luuf50c0   nginx_web.2   nginx:latest   worker2       Running         Running 23 seconds ago
hyh0peng1412   nginx_web.3   nginx:latest   worker3       Running         Running 23 seconds ago
c1avbrh3046t   nginx_web.4   nginx:latest   worker3       Running         Running 23 seconds ago
4iwxgm800c1p   nginx_web.5   nginx:latest   worker1       Running         Running 35 seconds ago
uhg0wonbeu5s   nginx_web.6   nginx:latest   DockerManager Running         Running 26 seconds ago
root@DockerManager:/home/swarm#
```

GAMBAR 4

Hasil dan Lokasi Replika Layanan Web Server

C. Hasil Pengujian High Availability

Pengujian high availability dilakukan dengan cara mengubah status dari salah satu worker node menjadi 'drain' sehingga node tidak dapat mengerjakan atau menerima layanan. Pada pengujian ini, masing-masing node diubah status menjadi 'drain' untuk diuji dengan diaktifkan kembali setelah selesai dilakukan pengujian.

TABEL 1

Hasil Pengujian Drain Node worker1

Container	Node Awal	Node Baru
nginx_web.1	worker1	DockerManager
nginx_web.2	worker2	worker2
nginx_web.3	worker3	worker3
nginx_web.4	worker3	worker3
nginx_web.5	worker1	worker2
nginx_web.6	DockerManager	DockerManager

TABEL 2

Hasil Pengujian Drain Node worker2

Container	Node Awal	Node Baru
nginx_web.1	DockerManager	DockerManager
nginx_web.2	worker2	worker1
nginx_web.3	worker3	worker3
nginx_web.4	worker3	worker3
nginx_web.5	worker2	worker1
nginx_web.6	DockerManager	DockerManager

TABEL 3

Hasil Pengujian Drain Node worker3

Container	Node Awal	Node Baru
nginx_web.1	DockerManager	DockerManager
nginx_web.2	worker1	worker1
nginx_web.3	worker3	worker2
nginx_web.4	worker3	worker2
nginx_web.5	worker1	worker1
nginx_web.6	DockerManager	DockerManager

TABEL 4

Hasil Pengujian Drain Node DockerManager

Container	Node Awal	Node Baru
nginx_web.1	DockerManager	worker3
nginx_web.2	worker1	worker1
nginx_web.3	worker2	worker2
nginx_web.4	worker2	worker2
nginx_web.5	worker1	worker1
nginx_web.6	DockerManager	worker3

Dari hasil pengujian tersebut dapat diketahui bahwa container akan secara otomatis berpindah ke node baru Ketika node Lokasi awal diubah statusnya menjadi 'drain'.

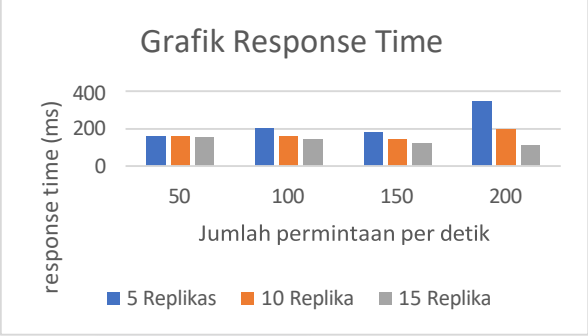
D. Hasil Pengujian Skalabilitas

Pengujian skalabilitas dilakukan dengan melakukan simulasi beban melalui Apache Jmeter dengan skenario variasi pengujian berupa variasi replika (5, 10, 15 replika) dan variasi beban permintaan per detik (50, 100, 150, 200 per detik).

TABEL 4

Hasil Response Time

Jumlah Permintaan	Jumlah Replika		
	5	**10**	**15**
50	156.5ms	155.4ms	154.2ms
100	203.4ms	159.6ms	140.8ms
150	178.8ms	141.6ms	123.6ms
200	345.5ms	195.9ms	110.8ms



GAMBAR 5

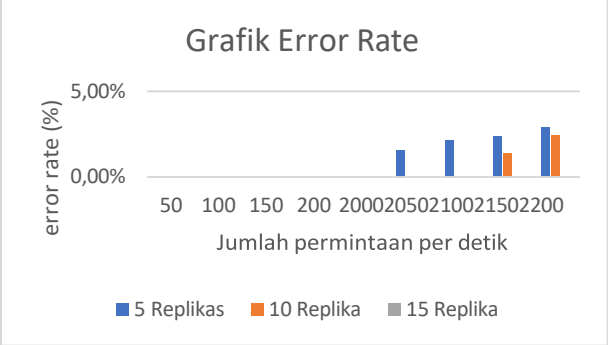
Grafik Response Time

Pada beban 50 permintaan per detik, response time turun dari 156,5 ms (5 replika) menjadi 155,4 ms (10 replika) dan 154,2 ms (15 replika). Pada beban 100 permintaan per detik, terjadi penurunan dari 203,4 ms menjadi 140,8 ms. Beban 150 permintaan per detik menghasilkan penurunan dari 178,8 ms menjadi 123,6 ms. Penurunan paling signifikan terjadi pada beban 200 permintaan per detik, yaitu dari 345,5 ms menjadi 110,8 ms (15 replika). Hasil ini menunjukkan bahwa penambahan replika mampu meningkatkan kecepatan respon sistem, terutama saat beban permintaan tinggi.

TABEL 5

Hasil Error Rate

Jumlah Permintaan	Jumlah Replika		
	5	**10**	**15**
50	0.00%	0.00%	0.00%
100	0.00%	0.00%	0.00%
150	0.00%	0.00%	0.00%
200	0.00%	0.00%	0.00%
2000	0.00%	0.00%	0.00%
2050	1.56%	0.00%	0.00%
2100	2.13%	0.00%	0.00%
2150	2.38%	1.36%	0.00%
2200	2.91%	2.43%	0.00%

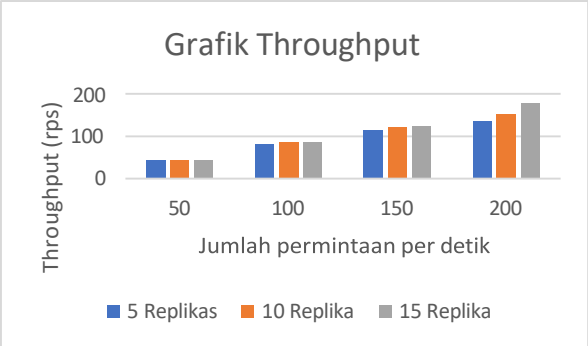


GAMBAR 6
Grafik Error Rate

Berdasarkan tabel tersebut, dapat diketahui bahwa sistem stabil dan dapat menangani permintaan dengan baik dengan beban mencapai 2000 permintaan per detik.

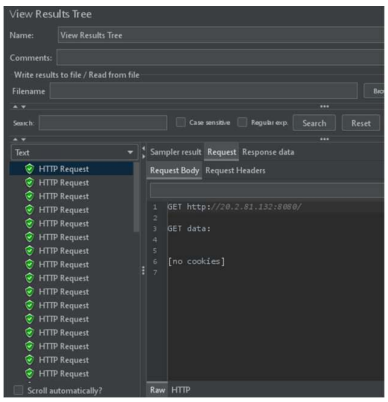
TABEL 6
Hasil Throughput

Jumlah Permintaan	Jumlah Replika		
	5	10	15
50	44.02rps	44.11rps	44.49rps
100	81.72rps	86.09rps	86.84rps
150	114.61rps	121.26rps	124.59rps
200	135.43rps	153.03rps	177.62rps



GAMBAR 7
Grafik Throughput

Hasil pengujian menunjukkan bahwa throughput meningkat seiring bertambahnya jumlah replika, terutama pada beban tinggi. Pada 50 permintaan per detik, throughput relatif sama di kisaran 44 rps. Perbedaan mulai terlihat pada 100 permintaan per detik, dengan peningkatan dari 81,72 rps (5 replika) menjadi 86,84 rps (15 replika). Pada 150 permintaan/detik, throughput naik dari 114,61 rps menjadi 124,59 rps, dan efek paling signifikan terjadi pada 200 permintaan/detik, yaitu dari 135,43 rps menjadi 177,62 rps. Hal ini menunjukkan bahwa penambahan replika meningkatkan kapasitas pemrosesan sistem, khususnya pada beban permintaan yang tinggi.



GAMBAR 8
Hasil View Results Tree

Error rate 0% berarti semua permintaan berhasil diproses tanpa kegagalan, namun hal ini tidak secara otomatis membuat throughput sama persis dengan jumlah permintaan yang dikirim per detik. Hal ini dapat terjadi karena adanya delay atau latency dalam pemrosesan, baik di sisi jaringan maupun aplikasi, sehingga sebagian request yang dikirim pada detik tertentu baru selesai diproses pada detik berikutnya. Akibatnya, walaupun semua permintaan berstatus sukses (terbukti dari View Result Tree yang seluruhnya hijau), distribusi waktu penyelesaiannya tidak merata tepat dalam satu detik, sehingga throughput yang diukur per detik menjadi lebih rendah dari target pengiriman.

V. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, dan pengujian, sistem cluster container menggunakan Docker Swarm pada infrastruktur Microsoft Azure berhasil dibangun dengan konfigurasi satu manager node dan tiga worker node, yang mampu mendukung pengelolaan layanan secara terdistribusi dan skalabel. Pengujian dilakukan pada layanan web server berbasis NGINX dengan variasi jumlah replika (5, 10, dan 15) dan beban permintaan (50, 100, 150, dan 200 permintaan per detik) menggunakan Apache JMeter, dengan parameter response time, error rate, dan throughput. Hasil pengujian menunjukkan bahwa Docker Swarm mampu menjaga ketersediaan layanan melalui konsep replikasi, serta memberikan peningkatan kinerja seiring penambahan replika. Rata-rata throughput meningkat dari 93,95 rps menjadi 108,885 rps, sementara error rate tetap 0% meskipun beban mencapai 2000 permintaan per detik. Selain itu, response time cenderung stabil di bawah 160 ms pada sebagian besar skenario. Dengan demikian, penambahan replika terbukti dapat meningkatkan kapasitas pemrosesan, menjaga response time tetap stabil, dan mempertahankan error rate yang rendah meskipun beban permintaan meningkat.

REFERENSI

[1] S. N. Putri and M. A. F. Ridha, "IMPLEMENTASI CLUSTERED CONTAINER DENGAN DOCKER SWARM," *9th Applied Business and Engineering Conference,* vol. 9, pp. 201-208, 2021.

[2] M. A. Nugroho and C. Subiyantoro, "ANALISIS CLUSTER CONTAINER PADA KUBERNETES DENGAN INFRASTRUKTUR GOOGLECLOUD PLATFORM," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika),* vol. 03, no. 02, pp. 84-93, 2018.

[3] S. E. Prasetyo and Y. Salimin, "Analisis Perbandingan Performa Web Server Docker Swarm dengan Kubernetes Cluster," *Conference on Management, Business, Innovation, Education and Social Science,* vol. 1, no. 1, pp. 826-833, 2021.

[4] W. Aldiwidianto and I. G. L. E. Prismana, "Analisis Perbandingan High Availability Pada Web Server Menggunakan Orchestration Tool Kubernetes Dan Docker Swarm," *Journal of Informatics and Computer Science,* vol. 05, no. 01, pp. 138-148, 2023.

[5] N. Singh, Y. Hamid, S. Juneja, G. Srivastava, G. Dhiman, T. R. Gadekallu and M. A. Shah, "Load balancing and service discovery using Docker Swarm for microservice based big data applications," *Journal of Cloud Computing: Advances, System and Applications,* vol. 12, no. 4, pp. 1-9, 2023.

[6] A. S. Manalu, I. M. Siregar, N. J. Panjaitan and H. Sugara, "RANCANG BANGUN INFRASTRUKTUR CLOUD COMPUTING DENGAN," *Jurnal TEKINKOM,* vol. 4, no. 2, pp. 303-311, 2021.

[7] E. Nurmiati, "ANALISIS DAN PERANCANGAN WEB SERVER PADA HANDPHONE," *STUDIA INFORMATIKA: JURNAL SISTEM INFORMASI,* vol. 5, no. 2, pp. 1-17, 2012.

[8] A. C. Barus, J. Harungguan and E. Manulu, "PENGUJIAN API WEBSITE UNTUK PERBAIKAN PERFORMANSI," *Journal of Applied Technology and Informatics,* vol. 1, no. 3, pp. 14-21, 2021.

[9] G. Y. Kusuma and U. Y. Oktiawati, "Perancangan Sistem Monitoring Performa Aplikasi Menggunakan," *Journal of Internet and Software Engineering (JISE),* vol. 3, no. 1, pp. 26-35, 2022.