

Analisis Perbandingan Performansi dan Keamanan Container Docker dalam Menjalankan Layanan WordPress dan Drupal

Badrid Maulana Ardi, Tomy Tri Sujaka, Kurniadin Abd Latif

Universitas Bumigora, Mataram, Indonesia

Correspondence : e-mail: badridardi48@gmail.com

Abstrak

Virtualisasi berbasis container menjadi populer karena kemampuannya yang ringan, efisien, dan mudah dikonfigurasi. Docker sebagai platform containerization memungkinkan pengemasan layanan dan dependensinya ke dalam satu unit yang portabel dan konsisten. Dalam konteks ini, pemilihan Content Management System (CMS) yang optimal untuk dijalankan dalam container menjadi penting, khususnya dalam hal performansi dan efisiensi sumber daya. Penelitian ini bertujuan untuk membandingkan performansi dan keamanan container Docker dalam menjalankan layanan WordPress dan Drupal. Metode yang digunakan adalah Network Development Life Cycle (NDLC) yang terdiri dari tahapan analisis, desain, dan simulasi. Dua layanan CMS yaitu WordPress dan Drupal dijalankan dalam container Docker dan dibandingkan performansi dan keamanannya. Uji coba dilakukan dengan mengukur response time, penggunaan CPU, penggunaan memory, dan kerentanan image. performansi diukur menggunakan Apache Benchmark, sedangkan kerentanan menggunakan Trivy. Pengujian performansi dilakukan sebanyak sepuluh kali pada setiap layanan. Hasil pengujian menunjukkan bahwa Drupal memiliki performa yang lebih baik dibandingkan WordPress. Rata-rata response time Drupal adalah 5,31 ms, jauh lebih cepat dibandingkan WordPress yang mencapai 104,93 ms. Dari sisi penggunaan sumber daya, WordPress mencatat rata-rata penggunaan CPU sebesar 179,62% dan memori sebesar 1,345 GiB, sedangkan Drupal hanya menggunakan CPU sebesar 67,79% dan memori rata-rata 175,206 MiB. Selain itu, image Drupal menunjukkan tingkat kerentanan yang lebih tinggi dibandingkan WordPress. Dengan demikian, Drupal menunjukkan performa yang lebih baik dari WordPress dalam hal response time, penggunaan CPU, dan memory saat dijalankan dalam container Docker. Namun dari sisi keamanan image, WordPress memiliki kerentanan lebih sedikit.

Kata kunci: Container, Docker, Drupal, WordPress.

Abstract

Container-based virtualization has become popular due to its lightweight, efficient, and easy-to-configure capabilities. Docker, as a containerization platform, allows the packaging of services and their dependencies into a single, portable and consistent unit. In this context, selecting the optimal Content Management System (CMS) to run in containers is crucial, particularly in terms of performance and resource efficiency. This study aims to compare the performance and security of Docker containers in running WordPress and Drupal services. The Network Development Life Cycle (NDLC) method is used, consisting of analysis, design, and simulation stages. Two CMS services, WordPress and Drupal, were run in Docker containers and their performance and security were compared. Tests were conducted by measuring response time, CPU usage, memory usage, and image vulnerabilities. Performance was measured using Apache Benchmark, while vulnerabilities were measured using Trivy. Performance testing was performed ten times on each service. The test results showed that Drupal performed better than WordPress. Drupal's average response time was 5.31 ms, significantly faster than WordPress's 104.93 ms. In terms of resource usage, WordPress recorded an average CPU usage of 179.62% and memory usage of 1,345 GiB, while Drupal only used 67.79% of the CPU and averaged 175,206 MiB of memory. Furthermore, Drupal images exhibited a higher level of vulnerabilities than WordPress. Thus, Drupal outperformed WordPress in terms of response time, CPU usage, and memory usage when running in a Docker container. However, in terms of image security, WordPress had fewer vulnerabilities.

Keywords: Container, Docker, Drupal, WordPress..

1. Pendahuluan

Teknologi virtualisasi yang berkembang dengan sangat pesat telah menjadikan virtualisasi berbasis *container* sebagai pilihan utama dalam pengembangan dan pengelolaan sistem operasi. *Container* adalah virtualisasi pada *level* sistem operasi di mana tiap proses atau aplikasi yang dijalankan di tiap *container* memiliki *kernel* yang sama[1]. Pada pengembangan dan penyebaran aplikasi *web modern*, efisiensi dan portabilitas yang ditawarkan *container* menjadi sangat penting. *Docker* adalah salah satu *platform* bersifat *open source* yang dibangun berdasarkan teknologi *container* dan ditujukan untuk membangun, mengemas, dan menjalankan aplikasi dimana pun dalam sebuah *container*[2]. Kemudahan penggunaan dan fleksibilitas yang ditawarkan menjadikan *Docker* sangat ideal untuk mengelola *web modern*, termasuk layanan *Content Management System (CMS)*. Layanan *CMS* seperti *WordPress* dan *Drupal* banyak digunakan dalam berbagai aplikasi web karena kemudahan pengelolaan konten yang mereka tawarkan. Menurut laporan [3] *Wordpress* menguasai 43.2% pasar *cms global* dan *platform CMS* yang paling banyak digunakan. Sedangkan menurut laporan [4] menunjukkan bahwa *Drupal* memiliki insiden keamanan jauh lebih sedikit dibandingkan *Wordpress*.

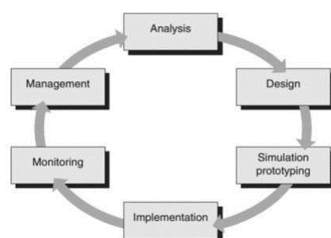
Beberapa penelitian yang berkaitan dengan *container* adalah penelitian yang dilakukan[5] membahas perbandingan kebutuhan *resource* dan tingkat independensi antara tiga teknologi dalam pengelolaan aplikasi *web server*, yaitu *single server*, virtualisasi, dan *Docker container*. Lalu penelitian yang dilakukan[6] membandingkan performansi tiga teknologi *container* populer *Docker*, *LXC*, dan *LXD* dalam menjalankan tiga jenis layanan, yaitu *web server*, *FTP server*, dan *mail server*. Penelitian dilakukan untuk mengukur efisiensi dan stabilitas masing-masing *container* dengan berbagai beban layanan. Lalu penelitian dari[7] membandingkan performa dan skalabilitas antara dua teknologi *container*, yaitu *Docker* dan *Kubernetes*, dalam mengelola layanan *WordPress* menggunakan metode *Horizontal Scaler*. Penelitian dilakukan dengan pendekatan *Action Research* dan menguji proses *scaling up* dan *scaling down*. Selanjutnya penelitian dari[8] membandingkan kinerja dari *container Docker* dan *Podman* sebagai *Container as a Service (CaaS)*. Penelitian ini membandingkan kinerja kedua *container* dari segi keamanannya terhadap serangan *cyber*.

Berbeda dengan penelitian sebelumnya tujuan penelitian ini adalah untuk menganalisis dan membandingkan performansi dan keamanan *Docker container* dalam menjalankan layanan *WordPress* dan *Drupal* menggunakan parameter seperti penggunaan *CPU*, *memory usage*, *response time*, dan kerentanan *image* dengan parameter jumlah dan kategori kerentanan seperti *low*, *medium*, *high*, dan *critical*. Untuk pengujian penelitian ini menggunakan *Apache Bench* dan *Trivy*. *Apache Bench* adalah program komputer baris perintah berulir tunggal yang digunakan untuk membandingkan *server web HTTP* [9]. Sedangkan *Trivy* merupakan alat pemindaian yang bertujuan untuk memindai kerentanan *container image* dan menggunakan *database Common Vulnerabilities and Exposures (CVE)* sebagai informasi dari kerentanan yang ditemukan pada *container image*[10].

Melalui penelitian ini, diharapkan dapat diperoleh hasil yang memberikan wawasan mengenai perbedaan performansi dan keamanan *Docker* dalam menjalankan layanan *WordPress* dan *Drupal* serta bagaimana *Docker* mampu mendukung layanan *CMS* dengan performansi yang efisien dan aman.

2. Metode Penelitian

Penelitian ini menggunakan *metode Network Development Life Cycle (NDLC)* merupakan metode untuk mengembangkan atau merancang sistem jaringan komputer dan memungkinkan pemantauan terhadap sistem yang sedang dirancang atau dikembangkan agar dapat diketahui kinerjanya[11].



Gambar 1. Metode NDLC

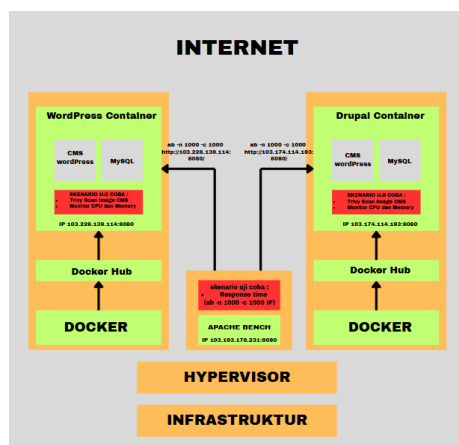
Penelitian ini menggunakan tiga tahapan yaitu analisis, desain, dan simulasi. Pada tahap analisis, dilakukan studi literatur untuk mengidentifikasi kebutuhan sistem. Tahap desain mencakup perancangan topologi, rancangan pengalamatan *IP*, serta alur kerja sistem uji coba. Tahap simulasi mencakup instalasi sistem, konfigurasi *container*, dan pelaksanaan uji performansi dan keamanan.

2.1 Tahap Analisis

Tahapan ini mengidentifikasi kebutuhan sistem. Data dikumpulkan melalui studi literatur untuk memahami konsep dan kekurangan penelitian sebelumnya. Analisis dilakukan terhadap performa dan keamanan *container Docker* dengan fokus pada efisiensi sumber daya, keamanan, dan kestabilan saat menjalankan layanan CMS.

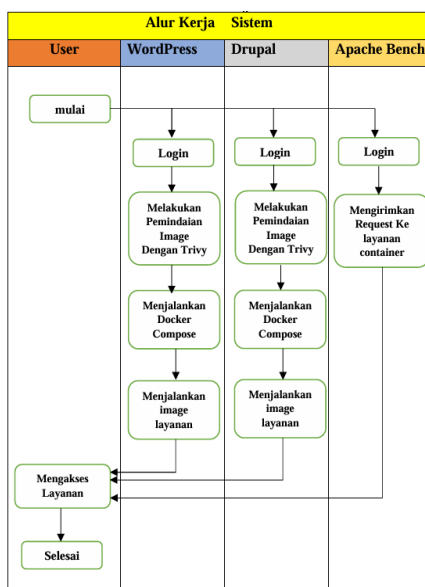
2.2 Tahap Desain

Pada tahap ini dilakukan perancangan topologi serta pengembangan alur kerja sistem yang akan diterapkan dalam skenario uji coba. Uji coba ini melibatkan pemasangan, penjalanan, dan penggunaan layanan CMS *WordPress* dan *Drupal*.



Gambar 2. Rancangan Jaringan Uji Coba

Rancangan jaringan untuk skenario uji coba akan disimulasikan secara virtual terdiri dari tiga VM utama,. VM pertama digunakan untuk menjalankan *container Docker* layanan *WordPress*, VM kedua digunakan untuk menjalankan *container Docker* layanan *Drupal* dan VM ketiga digunakan sebagai pengujian menggunakan *Apache Bench*. Seluruh VM menjalankan sistem operasi *Ubuntu Server 22.04.2 LTS*.



Gambar 3. Alur Kerja Sistem

Adapun beberapa proses kerja dalam sistem ini adalah sebagai berikut:

1. User akan menScan *image* layanan yang akan digunakan menggunakan *Trivy*.
2. User menjalankan *file Docker Compose* untuk mengunduh *image* layanan dari *Docker Hub* dan sekaligus mengatur konfigurasi layanan pada *container Docker*, termasuk pembuatan *database* masing-masing.

3. Setelah proses *Docker Compose* selesai, *image Wordpress* dan *Drupal* yang telah terkonfigurasi akan dijalankan secara otomatis pada masing-masing *container*.
4. *Container Docker* yang menjalankan layanan *Wordpress* akan menyajikan layanan melalui alamat IP 103.226.139.114:8080.
5. *Container Docker* yang menjalankan layanan *Drupal* akan menyajikan layanan melalui alamat IP 103.174.114.193:8080.
6. *User* dapat mengakses layanan *Wordpress* dan *Drupal* dengan membuka masing-masing alamat IP *container* melalui *web browser*.
7. Selanjutnya, *user* akan melakukan pengujian performa pada kedua *container* menggunakan *Apache Bench* dengan menjalankan beberapa skenario uji coba performa.
8. Proses selesai.

Dalam mendukung rancangan uji coba yang telah dibuat maka diperlukan perangkat keras dan perangkat lunak yang dapat mendukung implementasi dari rancangan yang dibuat.

1. Kebutuhan Perangkat Keras
 - A. Kebutuhan Perangkat Keras pada *Host container Docker* yang menjalankan layanan *Wordpress*.
 - 1) *Hardisk* : 20 GB
 - 2) *Processor* : 2
 - 3) *Memory* : 20 GB
 - B. Kebutuhan Perangkat Keras pada *Host container Docker* yang menjalankan layanan *Drupal*.
 - 1) *Hardisk* : 20 GB
 - 2) *Processor* : 2
 - 3) *Memory* : 20 GB
 - C. Kebutuhan Perangkat Keras pada *Apache Bench*
 - 1) *Hardisk* : 20 GB
 - 2) *Processor* : 2
 - 3) *Memory* : 20 GB
2. Kebutuhan Perangkat Lunak
 - A. *VirtualBox Graphical User Interface Version 6.1.34* untuk membangun virtualisasi sistem uji coba.
 - B. Sistem Operasi *Ubuntu Server 22.04.2 LTS* digunakan sebagai sistem operasi pada *container Docker*.
 - C. *Container Docker* versi 28.1.1 digunakan sebagai *container* yang menjalankan layanan *Wordpress:latest*.
 - D. *Container Docker* versi 28.1.1 digunakan sebagai *container* yang menjalankan layanan *Drupal:9.5*.
 - E. *Docker compose* digunakan untuk mengelola dan menyederhanakan konfigurasi *CMS*.
 - F. *MySQL* sebagai database management system pada layanan *Wordpress* dan *Drupal*.
 - G. *Microsoft Edge browser* untuk mengakses layanan *Wordpress* dan *MySQL*.
 - H. *Apache Benchmark* untuk mengukur performa *CPU*, *memory* dan *response time* pada saat *container Docker* menjalankan layanan *Wordpress* dan *Drupal*.
 - I. *Trivy* untuk menguji kerentanan *image CMS*.

2.3 Tahap Simulasi

Pada tahapan ini terdapat 4 (empat) skenario pengujian yaitu:

1. Pengujian kerentanan *image*. Pengujian ini dilakukan menggunakan *Trivy* untuk memindai *image Wordpress* dan *Drupal*. Hasil pemindaian akan menunjukkan jumlah dan tingkat kerentanan seperti *unknow*, *low*, *medium*, *high*, *critical*.
2. Pembuatan *image*. Tahapan ini berisi proses pembuatan dan konfigurasi *image* menggunakan *docker-compose*. Proses ini mencakup pengunduhan *image* dari *Docker Hub*, penyesuaian konfigurasi layanan (seperti *port*, *volume*, dan *environment variable*), serta pengujian awal untuk memastikan *container* dapat berjalan dengan baik.
3. Penggunaan layanan. Setelah *container* berhasil dibuat dan dijalankan, layanan *WordPress* dan *Drupal* digunakan sebagaimana mestinya untuk memastikan bahwa layanan berjalan normal, stabil, dan dapat diakses dengan baik.
4. Pengujian performansi. Pengujian ini dilakukan untuk mengukur performa *container* selama *CMS* dijalankan. Parameter yang diuji meliputi penggunaan *CPU* dan *Memory* untuk melihat seberapa besar

sumber daya sistem yang digunakan pada saat layanan aktif serta *response time* untuk menguji seberapa cepat server merespons.

3. Hasil dan Pembahasan

3.1. Hasil Uji Coba Kerentanan *Image* Menggunakan *Trivy*

Berdasarkan skenario hasil uji coba kerentanan *image* menggunakan *Trivy*, terdapat beberapa perbedaan yang signifikan antara jumlah dan tingkat keparahan kerentanan pada *image WordPress:latest* dan *Drupal:9.5*.

Tabel 1. Hasil Uji Kerentanan *Image*

Tingkat	Kerentanan <i>Image</i>	
	<i>WordPress:latest</i>	<i>Drupal:9.5</i>
Unknown	2	10
Low	473	453
Medium	557	2508
High	83	701
Critical	2	20
Total	1117	3629

Terlihat hasil *scan* jumlah kerentanan dari semua tingkat keparahan menggunakan *trivy* dimana jumlah kerentanan *image Drupal:9.5* memiliki jumlah total 3692 kerentanan lebih banyak dibandingkan *image WordPress:latest* yang hanya memiliki 1117 total kerentanan.

Kategori kerentanan kritis (*CRITICAL*), *Drupal* pada gambar memiliki 20 kerentanan kritis, sementara *WordPress* hanya memiliki 2 kerentanan kritis. Hal ini menunjukkan bahwa dari segi tingkat ancaman paling berbahaya, *image Drupal* lebih berisiko. Kategori *high* dan *medium* juga menunjukkan lebih tinggi jumlah kerentanan pada *Drupal*, dengan masing-masing 701 dan 2.508 dibandingkan dengan *WordPress* yang memiliki 83 dan 557 kerentanan pada kategori yang sama. Ini menandakan bahwa potensi eksploitasi pada *Drupal* lebih besar. Namun *WordPress* memiliki jumlah kerentanan *low* yang lebih tinggi dibandingkan *Drupal*, yaitu 473 berbanding 453. Sedangkan pada kategori *unknown*, yang mencakup kerentanan dengan tingkat keparahan belum teridentifikasi juga lebih banyak ditemukan pada *Drupal* 10 kerentanan dibandingkan *WordPress* 2 kerentanan.

3.2. Hasil Uji Pemasangan *Image Container*

Berdasarkan Skenario hasil uji coba pemasangan *image container*, kedua layanan ini tidak memiliki perbedaan yang signifikan.

Tabel 2. Hasil Uji Pemasangan *Image*

Uji Coba	Pemasangan <i>Image</i>	
	<i>WordPress:latest</i>	<i>Drupal:9.5</i>
Image Version	Latest	9.5
Docker Compose	473	Berhasil
Durasi Pemasangan	557	42,47 S
Ukuran <i>Image</i>	83	603 MB
MySQL Terintegrasi	2	Berhasil
Layanan Berjalan	1117	Berhasil
Ip Layanan		103.174.114.193:8080

Hasil pengujian skenario pemasangan *image container* menunjukkan bahwa *Docker Compose* berhasil mengelola proses *deployment* layanan *WordPress* dan *Drupal* secara efektif. Pada *WordPress*, *image wordpress:latest* berhasil diunduh dan dijalankan dengan waktu pemasangan 44,13 detik, sementara *Drupal* menggunakan *image drupal:9.5* dengan waktu pemasangan 42,47 detik. Keduanya dikonfigurasi melalui *docker-compose.yml* yang mencakup integrasi dengan *database MySQL* versi 5.7. *WordPress* memiliki ukuran *image* sebesar 703 MB, *Drupal* 603 MB, dan *MySQL* 501 MB. Perbedaan ukuran *image* berpengaruh terhadap durasi pemasangan serta kebutuhan ruang *disk*. Hasil ini membuktikan bahwa *Docker Compose* mampu melakukan *deployment* layanan CMS secara otomatis, stabil, dan efisien, termasuk dalam pengelolaan *image* berukuran besar serta integrasi layanan pendukung seperti *database*.

3.3 Hasil Uji Coba Menggunakan Layanan

Tabel 3. Hasil Uji Menggunakan Layanan

Uji Coba	Menggunakan Layanan	
	<i>WordPress:latest</i>	<i>Drupal:9.5</i>
Instalasi Via Web	Berhasil	Berhasil
Login Admin	Berhasil	Berhasil
Membuat Postingan	Berhasil	Berhasil
Postingan Muncul	Berhasil	Berhasil

Data Hilang Setelah Restart	Tidak	Tidak
-----------------------------	-------	-------

Hasil pengujian menunjukkan bahwa layanan *WordPress* dan *Drupal* dapat berjalan secara stabil di dalam lingkungan *container Docker*. Instalasi *WordPress* melalui IP 103.226.139.114:8080 berhasil hingga ke tahap login admin dan pembuatan konten (teks dan gambar), yang tetap tersimpan meskipun *container di-restart*. Hal serupa juga terjadi pada *Drupal* yang diakses melalui IP 103.174.114.193:8080, di mana proses instalasi, login, serta penambahan konten berjalan normal dan data tetap utuh setelah restart.

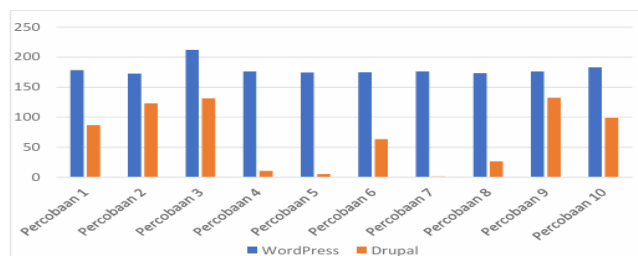
3.4 Hasil Uji Coba Performansi

3.4.1 Penggunaan CPU dan Memory

Pada skenario penggunaan *CPU* dan *Memory*, kedua layanan pada *container Docker* akan dilakukan pengiriman *request* sebanyak 1000 request sebanyak 10 kali.

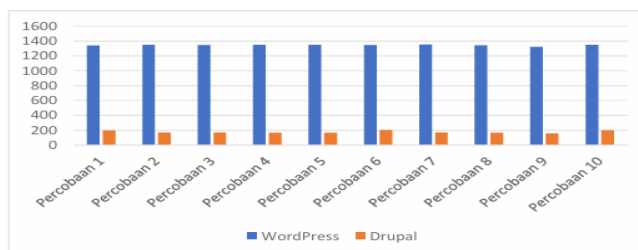
Tabel 4. Hasil Uji Penggunaan CPU dan Memory

Container	Percobaan	CPU	Memory
WordPress	1	178.10%	1,343 Gib
	2	172.58%	1.351 Gib
	3	212.20%	1,347 Gib
	4	176.36%	1.350 Gib
	5	174.04%	1.349 Gib
	6	174.71%	1.347 Gib
	7	175.96%	1353 Gib
	8	173.40%	1.344 Gib
	9	176.06%	1.323 Gib
	10	182.78%	1.348 Gib
Drupal	1	86.63%	194.5 Mib
	2	122.89%	168.6 Mib
	3	131.30%	168.6 Mib
	4	10.39%	165.7 Mib
	5	4.92%	165.8 Mib
	6	63.47%	201.1 Mib
	7	1.13%	166.7 Mib
	8	26.22%	165.8 Mib
	9	132.12%	155.4 Mib
	10	98.85%	199.4 Mib



Gambar 4. Grafik Penggunaan CPU

Pada *container Docker WordPress* terlihat pemakaian *CPU* terendah terlihat pada percobaan ke-2 yaitu sebesar 172.58% sedangkan untuk pemakaian *CPU* tertinggi terlihat pada percobaan ke-3 sebesar 212.20%. Untuk pemakaian *memory*, pemakaian terendah terlihat pada percobaan ke-9 yaitu 1.323 *Gib*, sedangkan untuk pemakaian tertinggi terlihat pada percobaan ke-4 sebesar 1.35 *Gib*.



Gambar 5. Grafik Penggunaan Memory

Pada *container Docker Drupal* ntuk pemakaian *CPU* tertinggi terlihat pada percobaan ke-9 yaitu sebesar 132.12%. Untuk pemakaian *memory*, pemakaian terendah terlihat pada percobaan ke-9 yaitu sebesar 155.4 *Mib*, sedangkan untuk pemakaian tertinggi terlihat pada percobaan ke-6 yaitu sebesar 201.1 *Mib*.

3.4.2 Response Time

Pada skenario *response time*, layanan *WordPress* dan *Drupal* pada *container Docker* akan dibebankan pengiriman *request* sebanyak 1000 request pada saat yang bersamaan. Setiap percobaan pengiriman *request* akan dilakukan sebanyak 10 kali.

Tabel 5. Hasil Uji Response Time

Container	Percobaan	Response Time (ms)
WordPress	1	161.084 ms
	2	108.173 ms
	3	106.818 ms
	4	89.757 ms
	5	113.747 ms
	6	96.900 ms
	7	103.872 ms
	8	101.378 ms
	9	82.002 ms
	10	77.559 ms
Drupal	1	7.302 ms
	2	6.137 ms
	3	4.769 ms
	4	4.640 ms
	5	4.494 ms
	6	5.356 ms
	7	4.362 ms
	8	4.341 ms
	9	5.163 ms
	10	6.165 ms



Gambar 6. Grafik Response Time

Rata-rata *response time WordPress* adalah 104,129 ms, dengan waktu tercepat 77,559 ms dan terlama 161,084 ms. Sementara itu, *Drupal* mencatat rata-rata 5,243 ms, dengan waktu tercepat 4,341 ms dan terlama 7,302 ms. Perbedaan ini kemungkinan dipengaruhi oleh arsitektur internal, optimasi *default*, dan kebutuhan *resource* masing-masing *CMS*.

4. Kesimpulan

Berdasarkan hasil pengujian, *Drupal* menunjukkan performa lebih baik dibandingkan *WordPress* dalam lingkungan *Docker*. *Drupal* memiliki rata-rata *response time* sebesar 5,31 ms, penggunaan *CPU* 67,79%, dan penggunaan *memory* sebesar 175 *MiB*. Sebaliknya, *WordPress* mencatat rata-rata *response time* sebesar 104,93 ms, penggunaan *CPU* 179,62%, dan penggunaan *memory* sebesar 1,345 *GiB*. Namun, dari sisi keamanan, *image WordPress* memiliki 1.117 kerentanan (termasuk 2 kritis), lebih sedikit dibandingkan *image Drupal* yang memiliki 3.692 kerentanan (dengan 20 kritis). Dengan demikian, *Drupal* unggul dalam hal performansi, sedangkan *WordPress* lebih baik dari segi keamanan *image*.

Daftar Pustaka

- [1] S. Dwiyatno, E. Rachmat, A. P. Sari, and O. Gustiawan, "Implementasi Virtualisasi Server Berbasis

- Docker Container,” *PROSISKO J. Pengemb. Ris. dan Obs. Sist. Komput.*, vol. 7, no. 2, pp. 165–175, 2020, doi: 10.30656/prosisko.v7i2.2520.
- [2] M. R. Alamsyah, H. I. Wahanani, and M. Idhom, “Infrastruktur Private Cloud Storage Berbasis,” *J. Inform. dan Sist. Inf.*, vol. 2, no. 2, pp. 122–131, 2021.
- [3] “WordPress Market Share, Statistics, and More,” WordPress.com. [Online]. Available: <https://wordpress.com/blog/2025/04/17/wordpress-market-share/>
- [4] IBM, “Drupal versus WordPress.” [Online]. Available: <https://www.ibm.com/id-id/think/topics/drupal-wordpress>
- [5] H. Afrizal and A. Prihanto, “Analisis Kebutuhan Resource Dan Independensi Antara Teknologi Single Server, Virtualisasi Dan Container,” *J. Informatics Comput. Sci.*, vol. 4, no. 01, pp. 26–33, 2022, doi: 10.26740/jinacs.v4n01.p26-33.
- [6] H. E. Wahanani and M. Idhom, “Analisis Performansi Container Docker, LXC dan LXD Pada Web Server, FTP Server dan Mail Server,” *Pros. Semin. Nas. Inform. Bela Negara*, vol. 1, pp. 45–49, 2020, doi: 10.33005/santika.v1i0.13.
- [7] N. Destarina, E. S. Negara, E. Supratman, and M. Ulfa, “Implementation Docker and Kubernetes Scaling Using Horizontal Scaler Method for Wordpress Services,” vol. 8, no. 4, pp. 2192–2196, 2024.
- [8] C. Mukmin, T. Naraloka, and Q. H. Andriyanto, “ANALISIS PERBANDINGAN KINERJA LAYANAN CONTAINER AS A SERVICE (CAAS) Studi Kasus : Docker dan Podman,” *Kumpul. J. Ilmu Komput.*, vol. 08, no. 2, pp. 152–161, 2021.
- [9] A. Amarulloh, K. Kurniasih, and M. Muchlis, “ANALISIS PERBANDINGAN PERFORMA WEB SERVICE REST MENGGUNAKAN FRAMEWORK LARAVEL, DJANGO, DAN Node JS PADA APLIKASI BERBASIS WEBSITE,” *J. Tek. Inform.*, vol. 9, no. 1, pp. 14–19, 2023.
- [10] A. A. Kroons and C. Dewi, “Pengembangan Dashboard Trivy Berbasis Website Menggunakan React Js Dan Golang,” *J. Indones. Manaj. Inform. dan Komun.*, vol. 4, no. 3, pp. 1037–1049, 2023, doi: 10.35870/jimik.v4i3.295.
- [11] U. A. Ahmad, R. E. Saputra, and Y. Pangestu, “Perancangan Infrastruktur Jaringan Komputer Menggunakan Fiber Optic Dengan Metode Network Development Life Cycle (Ndlc) Design of Computer Network Infrastructure Using Optical Fiber With Network Development Life Cycle (Ndlc) Method,” *Peranc. Infrastruktur Jar. Komput. Menggunakan Fiber Opt. Dengan Metod. Netw. Development Life Cycle Des. Comput. Netw. Infrastruct. Using Opt. Fiber With Netw. Dev. Life Cycle Method*, vol. 8, no. 6, pp. 12066–12079, 2021.