Vol. 7, No. 1, April 2026, hal. 105-111

Jurnal Computer Science and Information Technology (CoSciTech)

p-ISSN: 2723-567X
e-ISSN: 2723-5661

http://ejurnal.umri.ac.id/index.php/coscitech/index

Penerapan Victoriametrics Sebagai *Timeseries Database* untuk Monitoring Klaster Kubernetes

Roby Yasir Amri[*1], Nungky Awang Chandra[2], Mohamad Yusuf[3]
Email: [1]41521120006@student.mercubuana.ac.id, [2]nungky_awang@mercubuana.ac.id, [3]mhd.yusuf@mercubuana.ac.id

[123]Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Mercu Buana

Diterima: 7 Januari 2026 | Direvisi: 21 Februari 2026 | Disetujui: 27 April 2026

Abstrak

Monitoring infrastruktur merupakan komponen penting dalam pengelolaan klaster Kubernetes, khususnya untuk memastikan ketersediaan layanan dan analisis performa sistem. Seiring meningkatnya kompleksitas dan skala infrastruktur, sistem monitoring dituntut mampu mengelola volume data metrik yang besar secara efisien. Penelitian ini bertujuan untuk menganalisis performa VictoriaMetrics sebagai time-series database dalam sistem monitoring Kubernetes dan membandingkannya dengan Prometheus berdasarkan penggunaan sumber daya. Metode penelitian yang digunakan adalah pendekatan kuantitatif dengan eksperimen benchmark pada tiga skenario beban, yaitu 500, 750, dan 1000 target host. Parameter yang dianalisis meliputi penggunaan CPU, memori, dan kapasitas penyimpanan. Hasil penelitian menunjukkan adanya perbedaan signifikan dalam efisiensi penggunaan sumber daya, di mana VictoriaMetrics mempertahankan penggunaan CPU sebesar 2–10% pada seluruh skenario, jauh lebih rendah dibandingkan Prometheus yang mencapai 12–24%. Pada penggunaan memori, VictoriaMetrics hanya memerlukan 21–27%, sedangkan Prometheus meningkat hingga 41–67%. Dari sisi penyimpanan, VictoriaMetrics menggunakan 5–13 GB, sementara Prometheus membutuhkan 13–45 GB. Temuan ini diharapkan dapat menjadi referensi bagi organisasi dalam menentukan solusi monitoring yang sesuai dengan kebutuhan dan skala infrastruktur Kubernetes.

Kata kunci: *VictoriaMetrics, Kubernetes, Monitoring, Time Series Database, Prometheus,*

Implementing victoriametrics as a time series database for kubernetes cluster monitoring

Infrastructure monitoring is a critical component in managing Kubernetes clusters, particularly for ensuring service availability and analyzing system performance. As the complexity and scale of infrastructure increase, monitoring systems are required to efficiently handle large volumes of metric data. This study aims to analyze the performance of VictoriaMetrics as a time-series database within Kubernetes monitoring systems and compare it with Prometheus based on resource usage. The research employs a quantitative approach with benchmark experiments conducted under three load scenarios: 500, 750, and 1000 target hosts. The analyzed parameters include CPU usage, memory consumption, and storage capacity. The results indicate significant differences in resource efficiency, where VictoriaMetrics maintains CPU usage between 2–10% across all scenarios, substantially lower than Prometheus, which reaches 12–24%. In terms of memory consumption, VictoriaMetrics requires only 21–27%, whereas Prometheus increases to 41–67%. For storage usage, VictoriaMetrics consumes 5–13 GB, while Prometheus requires 13–45 GB. These findings are expected to serve as a reference for organizations in selecting an appropriate monitoring solution that aligns with their Kubernetes infrastructure scale and requirements.

Keywords: *VictoriaMetrics, Kubernetes, Monitoring, Time Series Database, Prometheus*

1. PENDAHULUAN

Ketersediaan dan keandalan sistem Teknologi Informasi menuntut kemampuan monitoring infrastruktur secara real-time dan historis. Berdasarkan pengalaman penulis sebagai *Cloud Engineer* di sebuah perusahaan perbankan di Indonesia, ditemukan kendala pada sistem monitoring berbasis Prometheus, terutama ketika melakukan pengambilan metrik dalam rentang historis panjang (±30 hari) dan digunakan oleh banyak pengguna secara bersamaan. Kondisi ini menyebabkan performa menurun hingga terjadi kegagalan sistem (*down*), yang berdampak pada terhambatnya proses monitoring, analisis performa, serta penyusunan laporan bulanan (*Monthly Report*).

Sebagai ilustrasi, permintaan query metrik *container_memory_working_set_bytes* untuk periode empat minggu kerap menghasilkan Gateway Time-out, menunjukkan keterbatasan Prometheus dalam menangani *query* historis berskala besar. Ketidakmampuan mengakses metrik historis ini menyulitkan tim monitoring dan vendor dalam penyusunan laporan berkala, sehingga berisiko mengganggu pemenuhan *Service Level Agreement* (SLA). Dampaknya juga merambat pada proses administratif seperti keterlambatan validasi *invoice*, yang dapat mempengaruhi arus kas dan operasional vendor.

Dengan meningkatnya kompleksitas lingkungan Kubernetes, kebutuhan akan solusi monitoring yang lebih andal dan efisien menjadi semakin penting. Prometheus merupakan salah satu tools monitoring populer dalam ekosistem CNCF (*Cloud Native Computing Foundation*), pertumbuhan volume metrik pada klaster berskala besar mengungkap keterbatasannya dalam penyimpanan jangka panjang, performa *query*, dan skalabilitas. Untuk mengatasi masalah tersebut, studi ini mengeksplorasi penggunaan VictoriaMetrics sebagai alternatif. VictoriaMetrics adalah *time series database* (TSDB) berperforma tinggi dengan efisiensi penyimpanan dan kemampuan menangani data metrik dalam skala besar [1], yang telah diadopsi oleh perusahaan seperti Adidas dan Grammarly [2].

Gap Penelitian

1. *Compare*

   Penelitian sebelumnya oleh [3] serta [4] menunjukkan bahwa kombinasi Prometheus dan Grafana efektif untuk *monitoring real-time* pada server tradisional maupun klaster Kubernetes. Namun, kedua studi tersebut belum membahas performa Prometheus dalam menangani *query* metrik historis berskala besar, akses paralel oleh banyak tim, serta analisis konsumsi sumber daya pada beban produksi yang tinggi. Dengan demikian, aspek efisiensi Prometheus dalam skenario operasional berskala enterprise masih menjadi celah penelitian

2. *Synthesize*

   Sintesis dari penelitian terdahulu menegaskan bahwa Prometheus dan Grafana terbukti efektif untuk monitoring infrastruktur skala kecil hingga menengah, visualisasi *real-time*, dan kebutuhan alerting. Implementasinya relatif mudah karena dukungan komunitas yang luas. Namun, terdapat kekurangan pada eksplorasi performa jangka panjang Prometheus pada skenario *multi-user*, keterbatasan evaluasi penggunaan time-series database alternatif yang lebih efisien, serta kurangnya kajian terkait optimalisasi sumber daya pada sistem monitoring skala *enterprise*

3. *Summarize*

   Secara keseluruhan, penelitian sebelumnya masih berfokus pada penggunaan Prometheus sebagai komponen utama monitoring. Belum terdapat penelitian yang mengevaluasi VictoriaMetrics sebagai *backend time-series database* dalam konteks peningkatan performa jangka panjang dan efisiensi operasional. Oleh karena itu, penelitian ini mengisi celah tersebut dengan mengusulkan implementasi dan evaluasi VictoriaMetrics pada klaster Kubernetes sebagai alternatif yang lebih efisien dan skalabel.

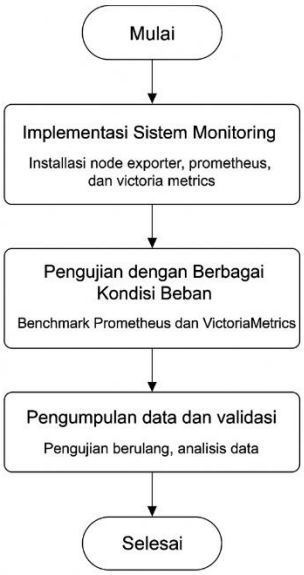
Tujuan Penelitian

Penelitian ini bertujuan untuk merancang serta mengimplementasikan sebuah sistem monitoring yang efektif dan efisien guna memantau kinerja dan kondisi infrastruktur *microservices* berbasis Kubernetes. Sistem tersebut dibangun melalui integrasi berbagai komponen pendukung, termasuk VictoriaMetrics sebagai *time series database* [5], Node Exporter sebagai pengumpul metrik, serta Grafana sebagai platform visualisasi, sehingga diharapkan mampu meningkatkan kapabilitas observabilitas dan keandalan lingkungan operasional.

2. METODE PENELITIAN

2.1. Tahapan Penelitian

Penelitian ini menggunakan pendekatan kuantitatif dengan metode eksperimen. Pendekatan kuantitatif dipilih karena penelitian ini berfokus untuk mengukur dan membandingkan performa sistem monitoring secara objektif dan terukur. Pendekatan ini didukung dengan pengujian berulang (*benchmarking*) [6], serta pengumpulan data metrik dari sistem monitoring itu sendiri.



Gambar 1 Tahapan Penelitian

Penelitian ini dilakukan kedalam tiga tahap utama, yaitu:

1. Implementasi Sistem Monitoring

   Melakukan instalasi dan konfigurasi Node Exporter, Prometheus, dan VictoriaMetrics pada lingkungan Kubernetes sebagai dasar pengujian performa.

2. Pengujian dengan Berbagai Kondisi Beban

   Melakukan pengujian benchmark pada Prometheus dan VictoriaMetrics dengan tiga scenario beban: 500, 750, dan 1000 host target, untuk mengukur stabilitas dan konsumsi resource seperti penggunaan CPU, memory, dan disk untuk melihat efektivitas pada kedua sistem

3. Pengumpulan Data dan Validasi

   Mengumpulkan hasil benchmark dan melakukan validasi melalui pengujian berulang guna memastikan konsistensi data sebelum dianalisis lebih lanjut

2.2. Teknik Pengumpulan Data

Penelitian ini menggunakan teknik:

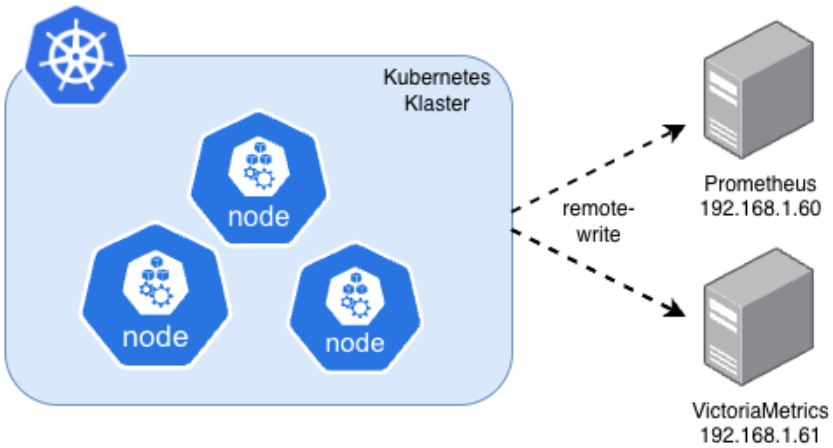
1. Studi Literatur dengan mengumpulkan data melalui kajian referensi akademik, dokumentasi resmi, dan hasil benchmark pihak ketiga untuk memahami performa serta efisiensi masing-masing *time-series database*.
2. Observasi langsung [5] dengan melakukan pengamatan terhadap sistem monitoring di lingkungan klaster Kubernetes, termasuk melakukan analisis error seperti *Gateway Timeout*, sebagai dasar identifikasi masalah dan pemilihan solusi alternatif.

2.3. Dasar Teori Penunjang

1. Kubernetes merupakan platform *open-source* yang digunakan untuk melakukan manajemen *workloads* aplikasi yang dikontainerisasi, serta menyediakan konfigurasi dan otomatisasi secara deklaratif. Kubernetes berada di dalam ekosistem yang besar dan berkembang cepat. *Service*, support Kubernetes tersedia secara meluas. [7]
2. Arsitektur *microservices* atau yang biasa disebut microservice mengacu pada gaya arsitektur untuk mengembangkan aplikasi. Microservice memungkinkan aplikasi berukuran besar dipisahkan menjadi bagian-bagian independent yang

lebih kecil, dengan setiap bagian memiliki area tanggung jawab sendiri. Untuk melayani satu permintaan pengguna, aplikasi berbasis *microservice* dapat memanggil banyak *microservice* internal untuk membuat responsnya

3. *Time Series Database* merupakan sistem penyimpanan khusus menangani data *time-series*. Data *time series* adalah serangkaian poin data yang diindeks oleh waktu, *Time Series* hanyalah pengukuran atau kejadian yang dilacak, dipantau, di downsampled, dan dikumpulkan dari waktu ke waktu.
4. VictoriaMetrics merupakan solusi *time-series* database terdistribusi yang cepat, hemat sumber daya dan kompatibel dengan Prometheus. Selain bersifat *OpenSource*, VictoriaMetrics juga menyediakan versi *Enterprise* yang sangat cocok digunakan untuk perusahaan yang membutuhkan support atau dukungan langsung dari teknisi VictoriaMetrics itu sendiri. VictoriaMetrics dirancang untuk dapat mengatasi keterbatasan Prometheus dalam skala besar dan kebutuhan akan penyimpanan data historis secara efisien.[8]
5. Prometheus merupakan perangkat lunak pemantauan dan peringatan sistem yang bersifat *opensource* awalnya dibuat di *SoundCloud*. Sejak dimulai pada 2012, banyak perusahaan dan organisasi yang telah mengadopsi Prometheus dan memiliki banyak komunitas pengembang dan pengguna yang sangat aktif. Sekarang proyek ini menjadi proyek *opensource* yang berdiri sendiri dan dikelola secara independent dari perusahaan manapun. Untuk menekankan hal ini, dan untuk memperjelas struktur tata kelola proyek, Prometheus bergabung dengan *Cloud Native Computing Foundation* pada tahun 2016 sebagai proyek kedua yang dihosting, setelah Kubernetes [9]
6. Topologi Benchmark



Gambar 2 Topologi Benchmark

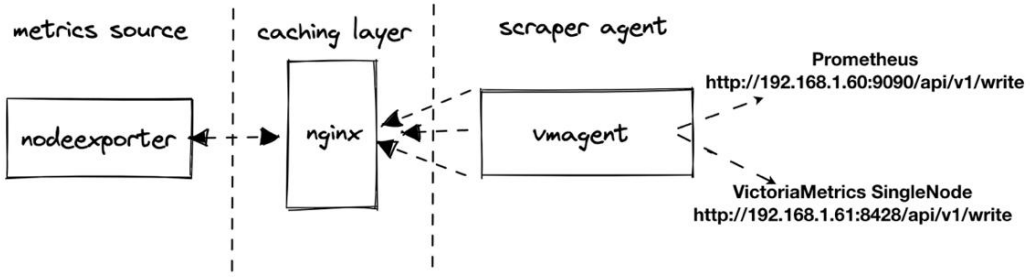
Topologi pada penelitian ini terdiri dari satu klaster Kubernetes yang berfungsi sebagai lingkungan pengujian, serta dua virtual machine terpisah untuk Prometheus (192.168.1.60) dan VictoriaMetrics (192.168.1.61). Klaster Kubernetes menjalankan tool dari repositori [10] untuk menghasilkan beban simulasi berupa sejumlah target *metrics* yang kemudian dikumpulkan oleh VMAgent di dalam klaster. Hasil pengumpulan data tersebut dikirimkan melalui mekanisme *remote write* secara bersamaan ke VM Prometheus dan VM VictoriaMetrics. Dengan konfigurasi ini, kedua sistem menerima beban yang identik sehingga dapat dibandingkan performanya dalam hal penggunaan sumber daya, kecepatan pemrosesan, dan efisiensi penyimpanan data *time series*.

Berikut ini merupakan table spesifikasi dan detail yang akan digunakan dalam proses pengetesan benchmark:

Tabel 2.1 Detail *Virtual Machines*

Name	IP	Resources			OS
		Memory (GB)	CPU (Core)	Disk (GB)	
Prometheus	192.168.1.60	4	4	80	Ubuntu 22.04.5 LTS
VictoriaMetrics	192.168.1.61	4	4	80	Ubuntu 22.04.5 LTS

7. Benchmarking



Gambar 3 Cara Kerja Benchmark
Referensi: https://victoriametrics.com/blog/remote-write-benchmark/

Topologi pada gambar 2.2 menunjukkan alur pengumpulan dan pengiriman *metrics* menggunakan vmagent sebagai scraper agent. Proses dimulai dari node exporter sebagai sumber metrics yang menyediakan data performa node. Data tersebut melewati nginx yang berfungsi sebagai *caching layer* untuk mengurangi beban akses langsung ke sumber data dan meningkatkan efisiensi pengambilan *metrics*. Selanjutnya, vmagent melakukan scrape terhadap metrics dari nginx dan mengirimkannya melalui mekanisme *remote write* ke dua tujuan sekaligus, yaitu Prometheus di http://192.168.1.60:9090/api/v1/write dan VictoriaMetrics *SingleNode* di http://192.168.1.61:8428/api/v1/write. Dengan arsitektur ini, proses pengumpulan data dapat berlangsung lebih stabil dan hasilnya dapat digunakan untuk membandingkan performa kedua sistem penyimpanan time series tersebut. [11]

3. HASIL DAN PEMBAHASAN

Proses benchmark menggunakan Prometheus-benchmark dianalisis secara komprehensif untuk mengevaluasi kinerja sistem monitoring. Pembahasan mencakup tahapan instalasi, analisis performa pasca-instalasi, pelaksanaan proses benchmark, perbandingan pemanfaatan sumber daya setelah pengujian, serta evaluasi manfaat yang diperoleh berdasarkan hasil yang dihasilkan. Pendekatan ini memungkinkan penilaian yang sistematis terhadap efektivitas dan efisiensi masing-masing platform monitoring dalam menangani beban kerja pada berbagai skala.

3.1. Pengujian dengan 500 Target

Pada pengujian awal, dilakukan simulasi terhadap 500 host yang secara paralel mengirimkan data metrik ke Prometheus dan VictoriaMetrics melalui *remote write*. Dengan ±1.230 metrik per node_exporter, jumlah *active time series* mencapai sekitar 615.000, dengan ingestion rate ±61.500 sampel/detik pada interval scrape 10 detik.

Hasil benchmark menunjukkan bahwa Prometheus menggunakan sumber daya lebih tinggi dibandingkan VictoriaMetrics. Penggunaan CPU pada Prometheus mencapai 12%, sedangkan VictoriaMetrics hanya 2%. Konsumsi memori Prometheus sebesar 41%, sementara VictoriaMetrics 21%. Pada sisi penyimpanan, Prometheus memerlukan 13 GB, sedangkan VictoriaMetrics hanya 5 GB. Temuan ini memperlihatkan perbedaan efisiensi arsitektur kedua sistem dalam pengelolaan pemrosesan data dan penyimpanan.

3.2. Pengujian dengan 750 Target

Pengujian kedua mensimulasikan 750 host dengan total ±922.500 *active time series*, menghasilkan *ingestion rate* sekitar 92.250 sampel/detik. Pengujian ini bertujuan mengevaluasi skalabilitas dan efisiensi penggunaan sumber daya pada peningkatan jumlah target.

Hasil benchmark menunjukkan bahwa Prometheus mencatat penggunaan CPU sebesar 14%, sedangkan VictoriaMetrics berada pada 10%. Penggunaan memori Prometheus mencapai 50%, jauh lebih tinggi dibandingkan VictoriaMetrics yang hanya 23%. Dari sisi penyimpanan, Prometheus membutuhkan 25 GB, sedangkan VictoriaMetrics memerlukan 10 GB. Perbedaan ini mengindikasikan bahwa VictoriaMetrics tetap lebih efisien ketika beban sistem meningkat.

3.3. Pengujian dengan 1000 Target

Pada pengujian ketiga, beban ditingkatkan menjadi 1.000 host atau sekitar 1,23 juta *active time series*. Laju pengumpulan data mencapai ±123.000 sampel/detik. Sebanyak 1% time series mengalami label churn setiap 10 menit untuk mensimulasikan dinamika beban operasional.

Hasil yang diperoleh menunjukkan perbedaan signifikan antara kedua platform. VictoriaMetrics mencatat penggunaan CPU sebesar 4%, sedangkan Prometheus mencapai 24%. Dalam penggunaan memori, VictoriaMetrics berada pada angka 27%, sementara Prometheus mencapai 67%. Dari segi penyimpanan, VictoriaMetrics memerlukan sekitar 13 GB, sedangkan

Prometheus membutuhkan hingga 45 GB. Hasil ini menegaskan bahwa VictoriaMetrics lebih efisien dalam mengelola data skala besar, terutama pada lingkungan dengan beban tinggi dan tingkat churn yang meningkat.

4. KESIMPULAN

Berdasarkan hasil pengujian benchmark dengan tiga scenario beban (500, 750, dan 1000 target *host*), didapatkan hasil sebagai berikut:

4.1. Penggunaan CPU (%)

Tabel 4.1 Tabel Penggunaan CPU

Jumlah Target	Prometheus	VictoriaMetrics
500	12%	2%
750	14%	10%
1000	24%	4%

Tabel penggunaan CPU menunjukkan bahwa VictoriaMetrics secara konsisten lebih efisien dibandingkan Prometheus pada seluruh skenario pengujian. Pada 500 hingga 1000 target, Prometheus mengalami peningkatan konsumsi CPU yang signifikan, sementara VictoriaMetrics tetap stabil pada tingkat yang jauh lebih rendah. Hal ini mengindikasikan bahwa VictoriaMetrics memiliki arsitektur pemrosesan yang lebih ringan dan optimal untuk beban besar.

4.2. Penggunaan Memori (%)

Tabel 4.2 Tabel Penggunaan Memori

Jumlah Target	Prometheus	VictoriaMetrics
500	41%	21%
750	50%	23%
1000	67%	27%

Tabel penggunaan memori memperlihatkan bahwa Prometheus menggunakan memori hampir dua kali lipat dibandingkan VictoriaMetrics pada semua tingkat beban. Kenaikan jumlah target secara langsung meningkatkan konsumsi memori Prometheus, sedangkan VictoriaMetrics mempertahankan penggunaan memori yang relatif lebih rendah dan stabil. Temuan ini menunjukkan bahwa VictoriaMetrics lebih efisien dalam manajemen *in-memory time series*.

4.3. Penggunaan Penyimpanan (GB)

Tabel 4.3 Tabel Penggunaan Penyimpanan

Jumlah Target	Prometheus	VictoriaMetrics
500	13 GB	5 GB
750	25 GB	10 GB
1000	45 GB	13 GB

Tabel penggunaan penyimpanan memperlihatkan perbedaan signifikan pada efisiensi kompresi dan manajemen data. Prometheus memerlukan ruang penyimpanan lebih besar pada seluruh skenario, dengan peningkatan yang tajam pada beban 1000 target. Sebaliknya, VictoriaMetrics menggunakan ruang disk yang jauh lebih kecil, menggambarkan efektivitas mekanisme kompresi dan struktur penyimpanan yang lebih optimal.

VictoriaMetrics terbukti lebih efisien dibandingkan Prometheus dalam pemanfaatan CPU, memori dan penyimpanan. VictoriaMetrics menunjukan penggunaan *resource* yang secara konsisten lebih rendah dan stabil pada seluruh scenario, sedangkan Prometheus mengalami peningkatan konsumsi yang cukup signifikan, khususnya pada memori dan disk. Kondisi ini mengindikasikan bahwa arsitektur VictoriaMetrics lebih optimal untuk menangani volume metrik yang besar sekaligus pemrosesan secara parallel.

Selain itu, VictoriaMetrics juga lebih andal dalam menyimpan data historis jangka panjang dan mengeksekusi query yang kompleks tanpa mengalami penurunan kinerja yang berarti, sehingga lebih sesuai untuk kebutuhan monitoring di lingkungan Kubernetes dengan skala besar. Berdasarkan temuan tersebut, VictoriaMetrics dapat direkomendasikan sebagai basis data time-series yang lebih efisien, stabil, dan mudah diskalakan untuk mendukung sistem monitoring modern.Temuan ini menegaskan bahwa risiko tinggi umumnya terjadi pada aset yang memiliki prioritas tinggi, terutama yang mengandung data sensitif atau mendukung layanan inti aplikasi.

DAFTAR PUSTAKA

[1] "Open Source Time Series Monitoring Tools & Solutions." Accessed: Dec. 23, 2025. [Online]. Available: https://victoriametrics.com/products/open-source/index.html
[2] "VictoriaMetrics: Case studies and talks." Accessed: Dec. 23, 2025. [Online]. Available: https://docs.victoriametrics.com/victoriametrics/casestudies/
[3] B. Rasyidi and F. Pratama, "Sistem Monitoring Server di PT. XYZ Media Indonesia Berbasis Grafana dan Prometheus," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, no. 4, pp. 1456–1465, Sep. 2024, doi: 10.57152/MALCOM.V4I4.1546.
[4] S. R. Dira, M. Arif, and F. Ridha, "Monitoring Kubernetes Cluster Menggunakan Prometheus dan Grafana," *Proceeding Applied Business and Engineering Conference*, no. November, 2022.
[5] Muhamad Satibi Mulya, I. Yustiana, and I. Lucia Khrisma, "Rancang Bangun Sistem Keamanan dan Monitoring Kendaraan Berbasis IoT dan Mobile Apps," *Jurnal CoSciTech (Computer Science and Information Technology)*, vol. 3, no. 2, pp. 58–65, Aug. 2022, doi: 10.37859/coscitech.v3i2.3934.
[6] A. Abdurrafiq Sujana, I. Yustiana, and A. Sujjada, "INTEGRASI QDRANT VECTOR DATABASE DAN DEEPSEEK AI UNTUK CHATBOT OTOMATIS PADA APLIKASI E-COMMERCE," *Jurnal CoSciTech (Computer Science and Information Technology)*, vol. 6, no. 2, pp. 311–318, Sep. 2025, doi: 10.37859/coscitech.v6i2.9668.
[7] "What Is Kubernetes? | Google Cloud." Accessed: Apr. 25, 2025. [Online]. Available: https://cloud.google.com/learn/what-is-kubernetes#what-is-kubernetes
[8] "Open Source Time Series Monitoring Tools & Solutions." Accessed: Dec. 23, 2025. [Online]. Available: https://victoriametrics.com/products/open-source/index.html
[9] "Overview | Prometheus." Accessed: May 31, 2025. [Online]. Available: https://prometheus.io/docs/introduction/overview/
[10] "GitHub - VictoriaMetrics/prometheus-benchmark: Benchmark for Prometheus-compatible systems." Accessed: Nov. 03, 2025. [Online]. Available: https://github.com/VictoriaMetrics/prometheus-benchmark
[11] "Benchmarking Prometheus-compatible time series databases." Accessed: Dec. 23, 2025. [Online]. Available: https://victoriametrics.com/blog/remote-write-benchmark/