JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 13 No. 3S1, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v13i3S1.8024

SISTEM MONITORING SERVER MENGGUNAKAN PROMETHEUS DAN GRAFANA DENGAN INTEGRASI SLACK DI PT. ATLAS LINTAS INDONESIA

Farhan Saory[1], Jajam Haerul Jaman[2], Carudin[3]

[1,2,3] Informatika, Universitas Singaperbangsa Karawang; Jl. HS.Ronggo Waluyo, Telukjambe Timur, Karawang, Jawa Barat 41361

Keywords:
Monitoring;
Prometheus;
Grafana;
Server.

Corespondent Email:
farhansaory33@gmail.com

Abstrak. Infrastruktur teknologi informasi yang andal sangat penting untuk menjaga kontinuitas layanan di perusahaan. PT. Atlas Lintas Indonesia belum memiliki sistem monitoring server yang mampu mendeteksi gangguan secara otomatis dan *real-time*, sehingga diperlukan solusi yang efisien. Penelitian ini bertujuan untuk mengimplementasikan sistem monitoring menggunakan Prometheus, Grafana, dan integrasi Slack untuk mendeteksi lonjakan penggunaan CPU, memori, dan *disk*. Metode penelitian yang digunakan adalah eksperimental dengan tahapan implementasi sistem, pengujian performa, dan analisis menggunakan *independent t-test*. Sistem diuji pada dua kelompok server: eksperimen (dengan monitoring) dan kontrol (tanpa monitoring). Hasil menunjukkan bahwa sistem monitoring berhasil mendeteksi anomali dengan lebih cepat dan memberikan notifikasi *real-time* melalui Slack. Terdapat perbedaan signifikan dalam waktu deteksi antara kedua kelompok, yang menunjukkan efektivitas sistem. Kesimpulannya, sistem ini mampu meningkatkan efisiensi pemantauan dan dapat mengurangi risiko *downtime* pada infrastruktur server perusahaan.

Abtsract. *Reliable IT infrastructure is essential for maintaining service continuity in companies. PT. Atlas Lintas Indonesia does not yet have a server monitoring system capable of detecting issues automatically and in real-time, thus requiring an efficient solution. This study aims to implement a monitoring system using Prometheus, Grafana, and Slack integration to detect spikes in CPU, memory, and disk usage. The research uses an experimental method, including system implementation, performance testing, and analysis using the independent t-test. The system was tested on two server groups: experimental (with monitoring) and control (without monitoring). Results showed that the monitoring system successfully detected anomalies faster and delivered real-time notifications via Slack. A significant difference in detection time was found between the two groups, indicating the system's effectiveness. In conclusion, the system improves monitoring efficiency and can reduce the risk of server infrastructure downtime.*

1. PENDAHULUAN

Seiring pesatnya perkembangan teknologi informasi, kebutuhan akan sistem monitoring server yang andal menjadi krusial, khususnya dalam menjamin stabilitas dan efisiensi operasional perusahaan. Keterlambatan dalam mendeteksi gangguan seperti kegagalan perangkat keras atau penggunaan sumber daya yang berlebihan dapat berdampak pada peningkatan waktu rata-rata deteksi dan perbaikan. Server sebagai pusat infrastruktur TI memerlukan sistem pemantauan yang mampu memberikan informasi kinerja secara *real-time*.

Sistem monitoring seperti Prometheus, Grafana, dan Slack dapat menjadi solusi untuk memantau kondisi server, menyajikan visualisasi metrik, serta mengirimkan notifikasi secara otomatis apabila terjadi anomali. Kombinasi ini terbukti efektif sebagaimana pada penelitian sebelumnya oleh Rahman *et al.* (2020), yang menunjukkan bahwa penerapan Prometheus dan Grafana berhasil memberikan peringatan dini terhadap gangguan sistem[14].

PT. Atlas Lintas Indonesia hingga saat ini masih mengandalkan metode manual yang kurang efisien. Oleh karena itu, penelitian ini bertujuan untuk mengimplementasikan sistem monitoring *real-time* berbasis Prometheus, Grafana, dan Slack di lingkungan PT. Atlas Lintas Indonesia

Penelitian ini difokuskan pada pemantauan metrik CPU, memori, dan kapasitas *disk* pada server tertentu yang ditentukan perusahaan. Manfaat yang diharapkan meliputi kontribusi terhadap pengembangan ilmu di bidang monitoring infrastruktur TI serta peningkatan efisiensi dan keandalan operasional bagi perusahaan dan tim IT dalam mendeteksi serta menangani masalah secara lebih cepat dan tepat.

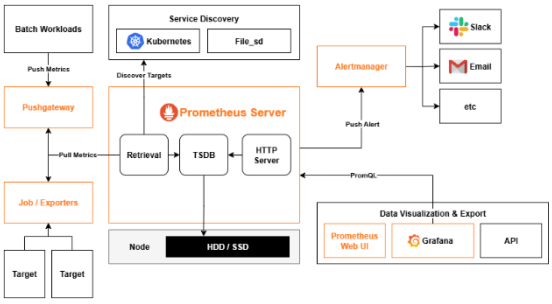
2. TINJAUAN PUSTAKA

2.1 Monitoring Server

Monitoring server merupakan proses pemantauan kondisi dan kinerja perangkat server secara terus-menerus. Menurut Aditya (2021), monitoring bertujuan untuk mengamati dan mengendalikan sistem sehingga administrator dapat merespons gangguan dengan cepat dan tepat [1]. Melalui monitoring, deteksi dini terhadap penggunaan sumber daya seperti CPU, memori, dan *disk* dapat dilakukan[3]. Salah satu alat yang umum digunakan dalam monitoring server adalah Prometheus.

2.2 Prometheus

Prometheus adalah perangkat lunak *open-source* untuk monitoring dan *alerting* yang berfungsi mengumpulkan metrik dari target yang telah ditentukan dalam format *time-series*. Prometheus memiliki *exporter* yang bertugas mengumpulkan metrik dari tiap target yang dimonitoring[2], salah satu yang popular adalah Node Exporter. Prometheus menggunakan bahasa *query* PromQL dan dapat mengirimkan peringatan melalui *alert manager* ke berbagai media notifikasi seperti Slack. Prometheus dapat mengumpulkan metrik melalui berbagai *exporter* yang dimilikinya[15]. Prometheus sendiri memiliki arsitektur yang dapat dilihat pada gambar 1.



Gambar 1. Arsitektur Prometheus

Gambar tersebut menggambarkan alur pengambilan data metrik, proses penyimpanan, pengelolaan notifikasi, serta integrasi dengan layanan visualisasi seperti Grafana.

2.3 Node Exporter

Node Exporter adalah komponen dalam ekosistem Prometheus yang berfungsi mengumpulkan metrik kinerja dari mesin atau server. Komponen ini diinstal pada setiap mesin target dan berjalan sebagai *daemon* service untuk memantau *resource* seperti CPU, memori, *disk*, dan jaringan. Pada sistem Linux, Node Exporter mengambil data dari direktori /proc dan /sys, lalu menyajikannya melalui *endpoint* HTTP pada port default 9100 dalam format *prometheus metrics* sehingga dapat diambil dan diolah oleh Prometheus.

2.4 Alert Manager

Alert Manager Adalah komponen yang bertugas untuk menangani notifikasi peringatan. Alert Manager menerima peringatan dari Prometheus Server dan mengirimkannya ke saluran notifikasi yang telah dikonfigurasi seperti email, Slack, PagerDuty dan lain-lain.

2.5 Grafana

Grafana adalah platform *open-source* untuk monitoring dan visualisasi data yang mendukung integrasi dengan berbagai sumber data seperti Prometheus, InfluxDB, MySQL,

dan lainnya. Grafana memungkinkan pembuatan *dashboard* interaktif untuk menampilkan metrik dalam bentuk grafik, tabel, maupun visualisasi lainnya[5]. Dengan fitur visualisasi yang kuat serta kemampuan integrasi dan *alerting*, Grafana membantu pengguna dalam menganalisis dan memahami performa system[11]. Platform ini dapat diakses melalui *browser* pada port 3000 dan memerlukan konfigurasi awal untuk koneksi ke sumber data yang digunakan.

2.6 Slack

Slack merupakan platform komunikasi berbasis *cloud* yang dirancang untuk meningkatkan efisiensi kerja tim melalui fitur pesan instan, saluran (*channel*) tematik, serta integrasi dengan berbagai layanan eksternal seperti Google Drive, Trello, GitHub, Prometheus, dan Grafana [7]. Integrasi dengan sistem monitoring dan alerting memungkinkan notifikasi dikirim secara *real-time*, sehingga mempermudah tim dalam merespons peringatan yang muncul.

2.7 Independent t-Test

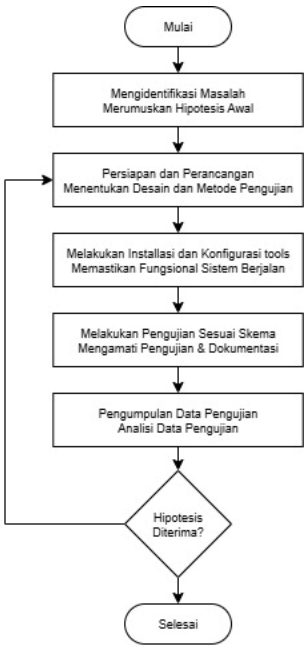
Independent t-Test adalah metode statistik parametrik yang digunakan untuk membandingkan rata-rata dua kelompok yang tidak saling berhubungan (independen) guna mengetahui apakah terdapat perbedaan yang signifikan di antara keduanya. Independent t-Test digunakan untuk menguji efektivitas sistem monitoring otomatis dibandingkan sistem monitoring konvensional, dengan membandingkan rata-rata waktu deteksi dan pengiriman notifikasi dari dua kelompok server.

2.8 SPSS Software

SPSS (*Statistical Package for the Social Sciences*) adalah perangkat lunak statistik yang digunakan untuk melakukan analisis data kuantitatif. SPSS mempermudah proses pengolahan data karena menyajikan output statistik secara visual dan interpretatif, serta menghasilkan nilai signifikansi (*p-value*) yang digunakan untuk pengambilan keputusan berdasarkan hipotesis penelitian. SPSS dipilih karena kemampuannya dalam mengolah data secara akurat dan efisien, serta kemudahan penggunaannya[6].

3. METODE PENELITIAN

Metode penelitian yang digunakan dalam penelitian ini adalah metode eksperimental, yang terdiri dari beberapa tahapan utama, yaitu identifikasi masalah dan perumusan hipotesis, desain eksperimen, implementasi sistem, pelaksanaan eksperimen, analisis data dan kesimpulan eksperimen. Dalam implementasinya, metode ini memiliki beberapa tahapan yang perlu dilakukan. Berikut merupakan tahapan yang dilakukan pada metode penelitian eksperimental pada gambar 2.



Gambar 2. Tahapan Penelitian

4. HASIL DAN PEMBAHASAN

4.1 Identifikasi Masalah & Perumusan Hipotesis

PT Atlas Lintas Indonesia masih menggunakan metode monitoring server secara konvensional melalui pengecekan manual dan alat bawaan sistem operasi. Cara ini memiliki keterbatasan, seperti keterlambatan dalam mendeteksi masalah, serta visualisasi data yang kurang efektif. Dengan meningkatnya kompleksitas sistem IT perusahaan, dibutuhkan metode monitoring yang lebih efisien dan proaktif. Oleh karena itu, penelitian ini mengusulkan implementasi monitoring berbasis Prometheus, Grafana, dan Slack sebagai solusi.

Berdasarkan identifikasi masalah di atas, hipotesis yang diajukan dalam penelitian ini adalah:

- H_0 (Hipotesis Nol): Tidak terdapat perbedaan efektivitas yang signifikan antara metode monitoring konvensional dengan metode berbasis Prometheus, Grafana dan Slack dalam memantau kinerja server di PT Atlas Lintas Indonesia.
- H_1 (Hipotesis Alternatif): Metode monitoring berbasis Prometheus, Grafana dan Slack terbukti lebih efektif dibandingkan dengan metode

4.2 Desain Eksperimen

Skema Penelitian

1) Spesifikasi

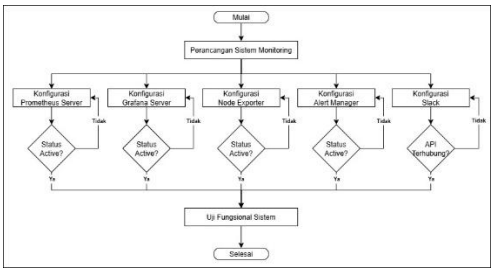
Dalam implementasinya sistem monitoring menggunakan 3 buah perangkat server fisik dengan 4 *virtual machine* yang akan diuji yang selanjutnya disebut dengan target monitoring. Server ini menjadi perbandingan efektivitas monitoring otomatis dengan monitoring konvensional. Berikut merupakan tabel spesifikasi server tersebut:

Tabel 1. Spesifikasi Server

Host Name	Ip Address	CPU	Mem	Disk	OS	Jenis
monitoring-srv	172.30.10.82	4 x E3-1225 v3 CPU	8GB	1TB	Ubuntu 20.04 LTS	Server Monitoring
atlindo-pve01	172.30.20.137	4 x E3-1225 v3 CPU	8GB	1TB	Proxmox 6.4 (Host)	Kelompok Eksperimen
webserver-pve01	172.30.20.139	1 (1 sockets, 1 cores)	2GB	100GB	Debian 8	Kelompok Eksperimen (VM)
dbserver-pve01	172.30.20.140	1 (1 sockets, 1 cores)	2GB	100GB	Debian 8	Kelompok Eksperimen (VM)
atlindo-pve02	172.30.30.170	4 x i3-3240 CPU	4GB	1TB	Proxmox 6.4 (Host)	Kelompok Kontrol
webserver-pve02	172.30.30.169	1 (1 sockets, 1 cores)	2GB	100GB	Debian 8	Kelompok Kontrol (VM)
dbserver-pve02	172.30.30.174	1 (1 sockets, 1 cores)	2GB	100GB	Debian 8	Kelompok Kontrol (VM)

2) Alur Konfigurasi

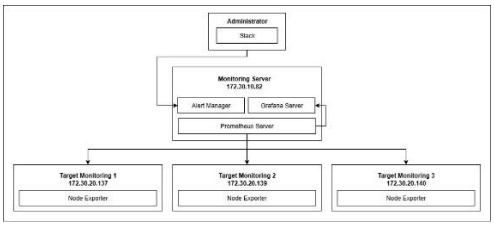
Berikut merupakan flowchart yang menggambarkan alur konfigurasi dan pengujian fungsional sistem monitoring. Proses dimulai dari tahap Mulai menuju Perancangan Sistem Monitoring hingga akhirnya siap untuk Uji Fungsional Sistem. Berikut merupakan gambar alur konfigurasi dari sistem yang dibuat.



Gambar 3. Alur Konfigurasi

3) Alur Kerja Sistem

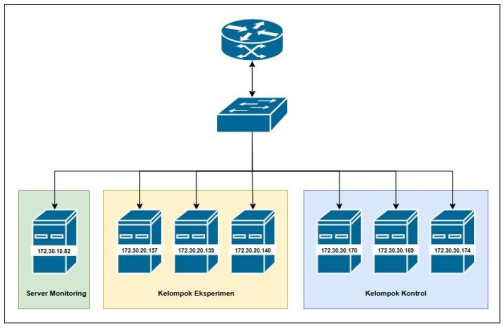
Berikut merupakan rangkaian alur kerja sistem monitoring yang digunakan. Alur ini menggambarkan secara *logic* bagaimana sistem saling terhubung.



Gambar 4. Alur Kerja Sistem

4) Desain Topologi

Penelitian ini menggunakan dua kelompok server, yaitu Kelompok Kontrol dan Kelompok Eksperimen, yang keduanya terhubung ke Server Monitoring. Topologi ini dirancang untuk membandingkan efektivitas metode monitoring konvensional dengan sistem berbasis Prometheus, Grafana, dan Slack.



Gambar 5. Desain Topologi

Skema Pengujian

1) Perfomance Testing

Pengujian performa dilakukan untuk menilai efektivitas dan responsivitas sistem

Prometheus dan Grafana dalam mendeteksi perubahan metrik serta mengirimkan notifikasi ke Slack. Pengujian mencakup:

- *Detection Response Testing*, yaitu mengukur waktu respon Prometheus dalam mendeteksi lonjakan metrik setelah pemicu dibuat.
- *Alert Delivery Time Testing*, yaitu mengukur waktu pengiriman notifikasi dari saat ambang batas metrik terpenuhi hingga *alert* diterima di Slack.

2) Uji Statistik Independent t-Test

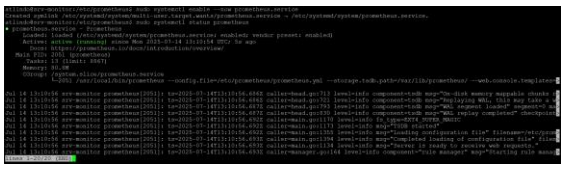
Uji Statistik (*Independent t-Test*) dilakukan terhadap data metrik yang telah dikumpulkan untuk menganalisis perbedaan antara server kontrol dan server eksperimen setelah penerapan sistem monitoring.

4.3 Implementasi Sistem

Sub-bab ini menyajikan hasil implementasi teknis dari desain arsitektur sistem monitoring yang telah diuraikan pada bagian metodologi. Implementasi sistem secara keseluruhan melibatkan integrasi antara server pengumpul metrik Prometheus, server visualisasi data Grafana, dan komponen notifikasi melalui Alertmanager yang terhubung dengan layanan Slack. Pembahasan akan mendetailkan konfigurasi, *deployment* pada lingkungan uji, hingga verifikasi keberhasilan integrasi setiap komponen utama.

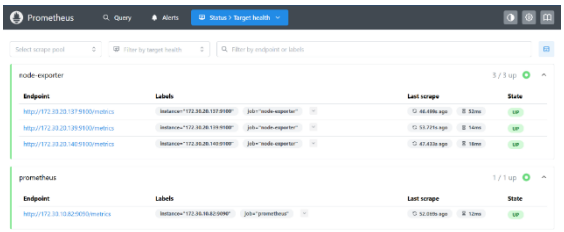
Implementasi Prometheus & Exporters

Instalasi Prometheus Server dilakukan pada server khusus, dan dikonfigurasi untuk berjalan sebagai layanan (*daemon*) melalui file *prometheus.service* guna memastikan stabilitas operasional. Verifikasi status menunjukkan bahwa layanan Prometheus aktif (*running*) dan beroperasi pada port 9090.



Gambar 6. Status Active Prometheus Server

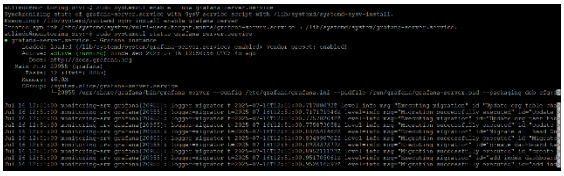
Konfigurasi inti terletak pada file prometheus.yml, di mana Node Exporter dari tiga server target telah didefinisikan sebagai endpoint scraping metrik. Verifikasi pada Web UI Prometheus di menu Status > Targets menunjukkan bahwa semua target terdeteksi dan memiliki status UP, menandakan keberhasilan pengumpulan data metrik.



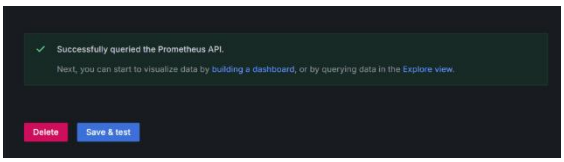
Gambar 7. Target Monitoring

Implementasi Grafana

Setelah layanan Grafana berhasil diinstal dan diaktifkan di port 3000, langkah krusial adalah mengintegrasikannya dengan Prometheus. Konfigurasi *data source* berhasil dilakukan dengan menunjuk URL server Prometheus, dan menghasilkan status koneksi berhasil (*successfully queried*).

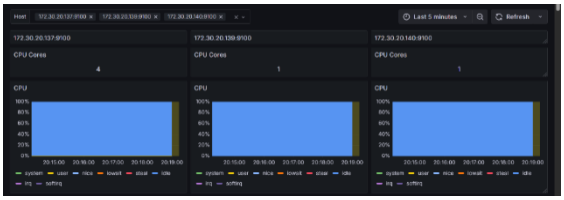


Gambar 8. Status *Active* Grafana Server

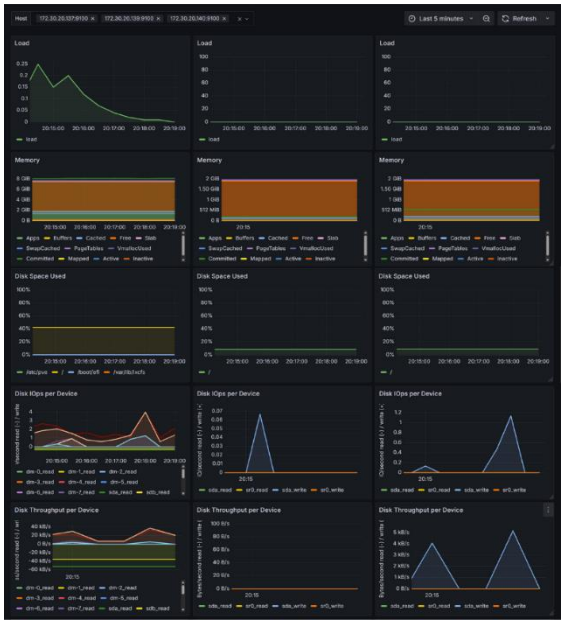


Gambar 9. Koneksi *Data Source*

Perancangan *dashboard* dilakukan menggunakan PromQL untuk memvisualisasikan data yang diambil Prometheus. Gambar 10 dan 11 menyajikan *dashboard* utama yang berhasil menampilkan metrik *real-time* dari server target, termasuk parameter krusial seperti Persentase Penggunaan CPU, penggunaan memori, *disk* I/O, dan beban rata-rata (*load average*).



Gambar 10. Dashboard Resource CPU



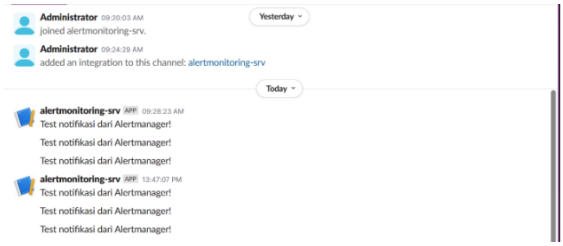
Gambar 11. Dashboard Grafana Keseluruhan

Implementasi Alertmanager

Aturan peringatan seperti HighCpuUsage dan HighMemoryUsage telah diimplementasikan dan dimuat oleh Prometheus. Untuk mekanisme notifikasi, Alertmanager dikonfigurasi melalui alertmanager.yml untuk merutekan semua peringatan ke layanan eksternal Slack.

Uji Notifikasi Slack

Integrasi dengan Slack dilakukan melalui Incoming Webhook URL yang dimasukkan ke dalam konfigurasi Alertmanager. Pengujian yang disimulasikan (misalnya, membuat beban CPU tinggi) membuktikan bahwa Alertmanager berhasil mengirimkan notifikasi *real-time* ke *channel* Slack (*#alertmonitoring-srv*) saat status peringatan FIRING dan mengirim notifikasi pemulihan (RESOLVED) saat kondisi kembali normal.



Gambar 12. Notifikasi Slack

4.4 Pelaksanaan Eksperimen

Setelah memastikan fungsional sistem monitoring berjalan dengan baik, maka tahap selanjutnya adalah melakukan tes performa sistem monitoring dengan menggunakan metode performance testing. Jenis pengujian ini meliputi Pengujian Respon Deteksi (*Detection Response Testing*) dan Pengujian Waktu Pengiriman Peringatan (*Alert Delivery Time Testing*).

Skema pengujian dilakukan dengan melakukan penambahan beban buatan pada server-server yang diuji dengan bersamaan dengan 2 (dua) kali pengujian. Berikut merupakan data selisih waktu pengiriman notifikasi dari kedua kelompok server yang diuji.

Tabel 2. Selisih Waktu Deteksi - Notifikasi

No	Selisih Waktu	
	Kontrol	Eksperimen
1	16.27m	08.28m
2	13.32m	09.12m
3	24.03m	13.28m
4	18.23m	13.34m
5	18.31m	11.24m
6	16.23m	18.33m

4.5 Analisis Data

Uji Normalitas Data

Sebelum melakukan *t-test*, dilakukan uji normalitas menggunakan metode Shapiro-Wilk karena jumlah sampel < 50. Uji normalitas dengan Shapiro-Wilk menunjukkan bahwa data kedua kelompok terdistribusi normal (p > 0,05). Berikut hasil uji normalitas data menggunakan SPSS.

Tabel 3. Uji Normalitas Data

Tests of Normality

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
Jenis Kelompok		Statistic	df	Sig.	Statistic	df	Sig.
Dalam Menit	Kontrol	.269	6	.199	.911	6	.445
	Eksperimen	.217	6	.200*	.931	6	.588

*. This is a lower bound of the true significance.
a. Lilliefors Significance Correction

Group Statistic

Berdasarkan hasil Group Statistics, rata-rata waktu notifikasi pada kelompok kontrol adalah 17,73 menit dengan standar deviasi 3,58, sedangkan pada kelompok eksperimen rata-rata waktu notifikasi hanya 12,27 menit dengan standar deviasi 3,63. Hasil ini menunjukkan bahwa sistem monitoring berbasis Prometheus, Grafana, dan Slack mampu mempercepat waktu deteksi dan pengiriman notifikasi dibandingkan metode konvensional.

Tabel 4. Group Statistic

Group Statistics					
	Jenis Kelompok	N	Mean	Std. Deviation	Std. Error Mean
Dalam Menit	Kontrol	6	17.7317	3.58067	1.46180
	Eksperimen	6	12.2650	3.62745	1.48090

Independent t-Test

Berdasarkan hasil Independent Samples t-Test, diperoleh nilai signifikansi (Sig. 2-tailed) sebesar 0,001 ($p < 0,05$). Hal ini menunjukkan bahwa terdapat perbedaan yang signifikan antara kelompok kontrol dan eksperimen dalam hal waktu notifikasi. Dengan demikian, hipotesis alternatif (H_1) diterima, yaitu sistem monitoring berbasis Prometheus, Grafana, dan Slack lebih efektif dibandingkan metode konvensional dalam mendeteksi dan mengirimkan notifikasi gangguan pada server.

Tabel 5. Independent t-Test

Independent Samples Effect Sizes					
				95% Confidence Interval	
		Standardizer[a]	Point Estimate	Lower	Upper
Dalam Menit	Cohen's d	3.60414	1.517	.181	2.797
	Hedges' correction	3.90588	1.400	.167	2.581
	Glass's delta	3.62745	1.507	.014	2.915

4.6 Kesimpulan Eksperimen

Hasil penelitian menunjukkan adanya perbedaan yang signifikan antara metode monitoring konvensional dengan sistem berbasis Prometheus, Grafana, dan Slack. Kelompok kontrol memiliki rata-rata waktu notifikasi sebesar 17,73 menit, sedangkan kelompok eksperimen hanya 12,27 menit. Perbedaan ini juga diperkuat oleh hasil Independent Samples t-Test dengan nilai $p = 0,001$ ($p < 0,05$), yang menandakan bahwa sistem monitoring yang diusulkan mampu memberikan deteksi dan notifikasi lebih cepat dibandingkan metode konvensional.

Percepatan waktu notifikasi ini disebabkan oleh kemampuan Prometheus dalam melakukan *scraping* metrik secara *real-time*, yang kemudian divisualisasikan melalui Grafana dan diteruskan ke Slack sebagai media notifikasi instan. Dengan alur ini, administrator tidak lagi bergantung pada pengecekan manual, sehingga respon terhadap gangguan dapat dilakukan lebih dini.

5. KESIMPULAN DAN SARAN

Sistem monitoring server berbasis Prometheus, Grafana, dan Slack terbukti lebih efektif dibandingkan metode konvensional. Hasil uji menunjukkan waktu notifikasi kelompok eksperimen lebih cepat (12,27 menit) dibandingkan kelompok kontrol (17,73 menit) dengan perbedaan signifikan ($p < 0,05$). Implementasi ini meningkatkan efisiensi monitoring dan membantu mengurangi potensi *downtime* pada server perusahaan.

Saran pengembangan yang dapat dilakukan adalah:

- Fokus monitoring dapat diperluas ke *service-level* seperti *web server*, *database*, *cron job* atau aplikasi perusahaan lainnya.
- Integrasi dengan *platform* orkestrasi atau *automation tool* untuk mendukung otomatisasi dan *self-healing*.
- Perluasan jumlah sampel dan variasi beban kerja untuk meningkatkan validitas hasil di lingkungan produksi nyata.

UCAPAN TERIMA KASIH

Dukungan dan bantuan yang diberikan oleh berbagai pihak telah menjadi faktor penentu dalam kelancaran dan keberhasilan penelitian ini. Oleh karena itu, penulis ingin menyampaikan penghargaan dan rasa terima kasih yang setinggi-tingginya atas kontribusi serta partisipasi yang berharga tersebut.

DAFTAR PUSTAKA

[1] Aditya, R., Pranatawijaya, V. H., & Putra, P. B. A. A. (2021). Rancang Bangun Aplikasi Monitoring Kegiatan Menggunakan Metode Prototype. *Journal of Information Technology and Computer Science*, *1*(1), 47-57.

[2] Albi, M. N. (2025). Rancang Bangun Teknologi Otomatisasi Ssh Pada Proxy Server

Menggunakan Framework Django Untuk Sistem Monitoring Cache Proxy. *Jurnal Informatika dan Teknik Elektro Terapan*, *13*(3).

[3] Amirudin, M. Z., Fahmi, R., Utami, E., & Mustafa, M. S. (2021). Evaluasi Penggunaan Prometheus dan Grafana Untuk Monitoring Database Mongodb. *Jurnal Informatika Polinema*, *7*(2), 43-50.

[4] Badri, R. I. (2021). Media Pembelajaran Pengenalan Komponen Komponen Komputer Menggunakan Adobe Flash CS6. *Engineering and Technology International Journal*, *3*(01), 83-96.

[5] Dira, S. R., & Ridha, M. A. F. (2023). Monitoring Kubernetes Cluster Menggunakan Prometheus dan Grafana. *ABEC Indonesia*, 345-351.

[6] Ediputra, K., Zulhendri, Z., & Hidayat, A. (2025). Pemanfaatan SPSS dalam Pengolahan Data dan Nilai di Tingkat SMP. *Dedikasi: Jurnal Pengabdian Pendidikan dan Teknologi Masyarakat*, *3*(1), 01-06.

[7] Firmansyah, Y. C., Winarno, W. W., & Pramono, E. (2019). Analisis Teknologi Virtual Mesin Proxmox dalam Rangka Persiapan Infrastruktur Server. *Jurnal Informa: Jurnal Penelitian dan Pengabdian Masyarakat*, *5*(3), 69-72.

[8] Irmawan, A. T., & Maulany, R. (2024). Implementation of proxmox server monitoring system with laravel and vue. js. *Jurnal Mantik*, *7*(4), 3971-3980.

[9] Juantoro, A., & Ratama, N. (2022). Sistem Notifikasi Monitoring Server Pada BOT Telegram Menggunakan Cronjob Berbasis Web (Studi Kasus: PT. Ekanuri Group). *OKTAL: Jurnal Ilmu Komputer dan Sains*, *1*(12), 2346-2351.

[10] Kusdinar, Y. I., & Widiastuti, N. (2020). Membangun Pola Komunikasi Berbantuan Teknologi Komunikasi "Slack". *Media Komunikasi FPIPS*, *19*(2), 62-72.

[11] Kusuma, G. Y., & Oktiawati, U. Y. (2022). Application performance monitoring system design using OpenTelemetry and Grafana Stack. *Journal of Internet and Software Engineering*, *3*(1), 26-35.

[12] Prabowo, M. I. A., Shadrii, Y. N., Ikhsan, B. A., Faizal, M., & Andaniari, P. L. (2025). Sistem Monitoring Server dan Website CV. Technos Studio Menggunakan Laravel Scheduller dan Telegram Bot. *Jurnal Pustaka Data (Pusat Akses Kajian Database, Analisa Teknologi, dan Arsitektur Komputer)*, *5*(1), 205-213.

[13] Rahma, A., Indriyani, F., & Sandi, T. A. A. (2023). Perancangan Dan Implementasi Monitoring Perangkat Server Menggunakan Zabbix Pada PT. Rizki Tujuh Belas Kelola. *Jurnal INSAN Journal of Information System Management Innovation*, *3*(2), 85-95.

[14] Rahman, D., Amnur, H., & Rahmayuni, I. (2020). Monitoring server dengan Prometheus dan Grafana serta notifikasi Telegram. *JITSI: Jurnal Ilmiah Teknologi Sistem Informasi*, *1*(4), 133-138.

[15] Rawoof, F. M., Tajammul, M., & Jamal, F. (2022). On-Premise Server Monitoring with Prometheus and Telegram Bot. *Interantional Journal Of Scientific Research In Engineering and Management*, *6*(04).