IJAI (Indonesian Journal of Applied Informatics)
Vol. 10 No. 1 Tahun 2025 pISSN: 2548-3846, eISSN: 2598-5981

Monitoring Kinerja *Virtual Machine* pada Lingkungan *Google Cloud Platform* dengan Notifikasi ke Media Sosial

Irna Widyaningsih[1]*, Abdul Haq[1], Tri Anggoro[1]
[1]Program Studi S1 Informatika, Universitas Nahdlatul Ulama Al Ghazali Cilacap, Indonesia
*Email: irnawidya24@gmail.com

Info Artikel

Kata Kunci :
monitoring, GCP, virtual machine, CPU usage, telegram.

Keywords :
monitoring, GCP, virtual machine, CPU usage, telegram.

Tanggal Artikel
Dikirim : 18 September 2025
Direvisi : 25 Desember 2025
Diterima : 30 Desember 2025

Abstrak

Virtual Machine (VM) merupakan elemen penting dalam *cloud computing* karena mendukung fleksibilitas dan efisiensi pengelolaan sumber daya. Namun, lonjakan penggunaan VM dapat menurunkan kinerja jika tidak terdeteksi cepat. Penelitian ini mengembangkan sistem monitoring pada *Google Cloud Platform* (GCP) dengan Grafana yang terintegrasi Telegram untuk peringatan dini otomatis. Prometheus digunakan sebagai pengumpul metrik, sedangkan Grafana menampilkan visualisasi, berfokus pada pemantauan CPU secara *real-time* di Google Compute Engine (GCE). Notifikasi dikirim melalui Telegram ketika penggunaan CPU melewati ambang batas. Pengujian menunjukkan rata-rata keterlambatan notifikasi hanya 1 detik, kecuali satu anomali 11 detik. Pada skenario Threshold Validation, terjadi satu alert dengan CPU maksimum 37% dan rata-rata 29%, sedangkan Long Hold menghasilkan tiga alert dengan rata-rata 23,3% sesuai konfigurasi interval. Hasil ini membuktikan sistem mampu memberikan notifikasi hampir *real-time*, menjaga konsistensi, dan mendukung deteksi dini baik pada beban singkat maupun berkepanjangan di infrastruktur GCP.

Abstract

Virtual Machines (VMs) are essential in cloud computing for flexibility and efficient resource management. However, sudden spikes in VM usage can degrade performance if not detected quickly. This study develops a monitoring system on Google Cloud Platform (GCP) using Grafana integrated with Telegram for automated early alerts. Prometheus collects metrics, while Grafana provides visualization, focusing on real-time CPU monitoring in Google Compute Engine (GCE). Alerts are sent via Telegram when CPU usage exceeds a set threshold. Testing shows an average notification delay of 1 second, except for a single 11-second anomaly. In the Threshold Validation scenario, one alert occurred with 37% maximum CPU and 29% average, while the Long Hold scenario produced three alerts with an average of 23.3%, following configured intervals. Results indicate the system delivers near real-time, consistent alerts and supports early detection under both short and sustained load conditions on GCP infrastructure.

1. PENDAHULUAN

Di era digitalisasi saat ini, teknologi *cloud computing* menjadi solusi utama bagi organisasi dalam memenuhi kebutuhan komputasi. Salah satu teknologi inti dalam *cloud computing* adalah virtualisasi, yang memungkinkan pembagian sumber daya fisik ke dalam beberapa lingkungan digital berupa *Virtual Machine* (VM). Setiap VM dapat menjalankan aplikasi secara independen, sehingga pemantauan kinerjanya menjadi krusial untuk menjaga stabilitas dan keandalan sistem. Monitoring diperlukan untuk memastikan performa layanan tetap optimal dan mendeteksi gangguan yang umumnya disebabkan oleh lonjakan atau keterbatasan sumber daya, terutama CPU, memori, dan disk [1]. Di antara ketiganya, CPU menjadi indikator utama penurunan kinerja karena seluruh aktivitas pemrosesan bergantung pada komponen ini. Panduan *Compute Engine Insights* dari *Google Cloud* menetapkan kondisi *HIGH_CPU_USAGE* ketika penggunaan CPU mencapai 83% atau lebih selama 90% waktu pemantauan [2], yang menjadi tanda awal risiko penurunan performa.

Tanpa sistem monitoring yang dilengkapi notifikasi otomatis, banyak organisasi kesulitan mendeteksi anomali secara dini. Gangguan sering kali baru diketahui setelah berdampak pada pengguna, sehingga menimbulkan *downtime* tidak terencana. Studi menunjukkan bahwa lebih dari 60% *downtime* disebabkan oleh keterlambatan deteksi *overload* CPU, dengan potensi kerugian finansial mencapai rata-rata USD 125.000 per jam pada perusahaan berskala menengah [3].

Beberapa penelitian sebelumnya telah mengembangkan sistem monitoring menggunakan Prometheus dan Grafana. Salah satu penelitian mengimplementasikan monitoring pada lingkungan Kubernetes, tetapi mekanisme *alerting*-nya masih terbatas [4]. Penelitian lain di Universitas Udayana menggunakan Prometheus dan Grafana untuk memantau metrik CPU, memori, dan disk, namun belum mengintegrasikan fitur notifikasi otomatis [5].

Penerapan *alerting* otomatis melalui integrasi dengan Telegram telah dilakukan, tetapi penerapannya masih terbatas pada server atau jaringan tertentu. Salah satu penelitian membahas sistem monitoring server menggunakan Prometheus dan Grafana yang berhasil mengirimkan notifikasi *real-time* ke Telegram ketika layanan server bermasalah, sehingga memudahkan administrator memantau performa server dengan lebih efisiensi[6]. Penelitian lain menyoroti otomatisasi manajemen insiden dengan integrasi *alerting* ke sistem tiket. Sistem monitoring ini memungkinkan pembuatan tiket insiden secara otomatis berdasarkan alert dari Prometheus, sehingga mempercepat respons terhadap kegagalan sistem dan meningkatkan efisiensi manajemen insiden[7]. Penerapan *unified monitoring* pada arsitektur *microservices* menggunakan Prometheus dan Grafana menunjukkan efektivitas *alerting real-time*, tetapi masih terbatas pada lingkungan *microservices* tertentu tanpa memperhatikan resource server secara keseluruhan[8]. Monitoring server Linux *real-time* dengan Prometheus dan Grafana mampu memberikan pandangan menyeluruh terhadap performa, namun *alerting* otomatisnya belum dioptimalkan untuk berbagai jenis gangguan[9].

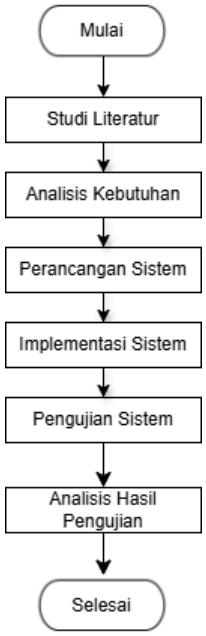
Penelitian sebelumnya masih memiliki keterbatasan. Sistem monitoring sering hanya fokus pada satu jenis layanan infrastruktur, sehingga pemantauan tidak menyeluruh. Integrasi notifikasi *real-time* belum diterapkan secara konsisten, terutama di lingkungan *microservices* atau *cloud* berskala besar. Menanggapi keterbatasan tersebut, penelitian ini merancang sistem monitoring yang tidak hanya menampilkan data kinerja secara visual, tetapi juga mengintegrasikan notifikasi otomatis melalui platform Telegram. Telegram dipilih karena mendukung integrasi dengan Grafana melalui Bot API dan banyak digunakan oleh adiministrator. Selain itu, penelitian ini menggabungkan pengujian beban menggunakan Apache JMeter terhadap layanan *web server* pada VM untuk memicu lonjakan CPU sebagai representasi beban kerja nyata di lingkungan *cloud*.

Fokus penelitian diarahkan pada pengujian efektivitas sistem monitoring dalam mendeteksi lonjakan CPU serta memberikan peringatan otomatis secara *real-time*. Penelitian dilakukan pada *Google Cloud Platform* (GCP) sebagai salah satu penyedia layanan *cloud* yang banyak digunakan. Dengan demikian, penelitian ini diharapkan menghasilkan sistem monitoring yang mampu menampilkan metrik kinerja sekaligus memberikan notifikasi *real-time* ketika terjadi lonjakan beban, sehingga mampu mengatasi keterbatasan penelitian sebelumnya yang belum mengintegrasikan mekanisme peringatan.

2. METODE PENELITIAN

Tahapan dalam penelitian proyek akhir ini dilakukan dengan melakukan perumusan masalah pada sistem monitoring VM, kemudian dilakukan studi literatur untuk mengidentifikasi kebutuhan sistem berupa teknologi dan skema yang akan diimplementasikan. Berdasarkan hasil studi literatur maka dilakukan perancangan sistem dan arsitektur perangkat yang digunakan. Tahapan selanjutnya melakukan instalasi perangkat lunak yang akan menunjang terlaksananya penelitian ini. Proses instalasi mencakup integrasi Grafana dengan Prometheus sebagai *data source*, pembuatan Alert Rule, serta pengaturan notifikasi melalui Telegram. Implementasi dijalankan pada tiga VM, yaitu VM Monitoring (Prometheus, Grafana), VM Target (Node Exporter, Nginx), dan VM Pengujian (Apache JMeter). Pengujian dilakukan secara fungsional untuk memverifikasi proses monitoring dan notifikasi, serta stress testing untuk menilai kecepatan dan stabilitas sistem dalam mendeteksi lonjakan beban

CPU. Tahap akhir berupa analisis hasil pengujian untuk mengevaluasi efektivitas sistem dalam memberikan peringatan *real-time* terhadap kondisi kritis. Dan diagram alir metode penelitian ditunjukkan pada Gambar 1.



Gambar 1. Diagram alir penelitian

Berikut penjelasan tahapan dari Gambar 1:

1) Studi Literatur
   Tahap ini dilakukan untuk mengumpulkan teori, penelitian terdahulu, serta referensi terkait sistem monitoring, *cloud computing*, dan metode pengujian yang relevan.

2) Analisis Kebutuhan
   Mengidentifikasi kebutuhan sistem baik dari sisi perangkat keras maupun perangkat lunak, termasuk spesifikasi VM, *tools* monitoring, serta media notifikasi yang akan digunakan.

3) Perancangan Sistem
   Pada tahap perancangan, kegiatan difokuskan pada penyusunan arsitektur sistem monitoring, perancangan alur kerja, serta perencanaan integrasi antar komponen seperti Prometheus, Grafana, Telegram, dan JMeter yang akan dijalankan di lingkungan *cloud*. Perangkat laptop tidak digunakan untuk menjalankan sistem monitoring, melainkan berfungsi sebagai perangkat kerja untuk mengelola layanan *cloud*, melakukan konfigurasi, serta mengakses dashboard monitoring secara *remote*.
   Laptop digunakan untuk menyusun desain arsitektur, membuat konfigurasi awal, menyiapkan *script*, serta melakukan manajemen VM di Google Cloud Platform melalui SSH dan antarmuka web. Oleh karena itu, spesifikasi laptop harus memadai agar proses pengelolaan *cloud*, editing konfigurasi, serta akses dashboard Grafana dapat berjalan lancar. Keterbatasan spesifikasi pada perangkat lokal dapat menghambat proses administrasi dan monitoring jarak jauh, meskipun eksekusi sistem sepenuhnya berjalan di sisi *cloud*.

Tabel 1. Spesifikasi *hardware* tahap perancangan

Perangkat	*CPU*	*RAM*	*Storage*	*Sistem Operasi*	*Keterangan*
Laptop	Intel Core i5-12450H	8GB	512 GB SSD	Windows 11	Digunakan untuk perancangan sistem, konfigurasi *cloud*, dan akses dashboard monitoring

4) Implementasi Sistem

Tahap implementasi dilakukan sepenuhnya pada Virtual Machine (VM) di Google Cloud Platform (GCP) sesuai dengan rancangan yang telah disusun sebelumnya. VM ini berperan sebagai lingkungan utama tempat seluruh komponen sistem monitoring dijalankan, termasuk Prometheus, Grafana, Telegram Bot, dan layanan *web server* yang akan diuji. Implementasi dilakukan pada lingkungan cloud agar mendekati kondisi operasional nyata dan memungkinkan pengujian performa secara objektif. Pada tahap ini, Prometheus dikonfigurasikan untuk mengumpulkan metrik performa VM, Grafana digunakan untuk menampilkan visualisasi monitoring, serta Telegram Bot diintegrasikan sebagai media notifikasi otomatis ketika terjadi lonjakan penggunaan sumber daya. Apache JMeter dijalankan untuk memberikan beban pada web server di dalam VM guna memicu peningkatan penggunaan CPU sebagai skenario uji. Seluruh proses ini berlangsung di *cloud*, sehingga performa sistem tidak bergantung pada spesifikasi perangkat lokal.

Tabel 2. Spesifikasi *hardware* tahap implementasi

Perangkat	*vCPU*	*RAM*	*Storage*	*Sistem Operasi*	*Region / Data Center*	*Keterangan*
Virtual Machine Google Cloud Platform	2 vCPU	8GB	50 GB	Ubuntu 22.04 LTS	Asia Tenggara	Lingkungan utama implementasi sistem monitoring

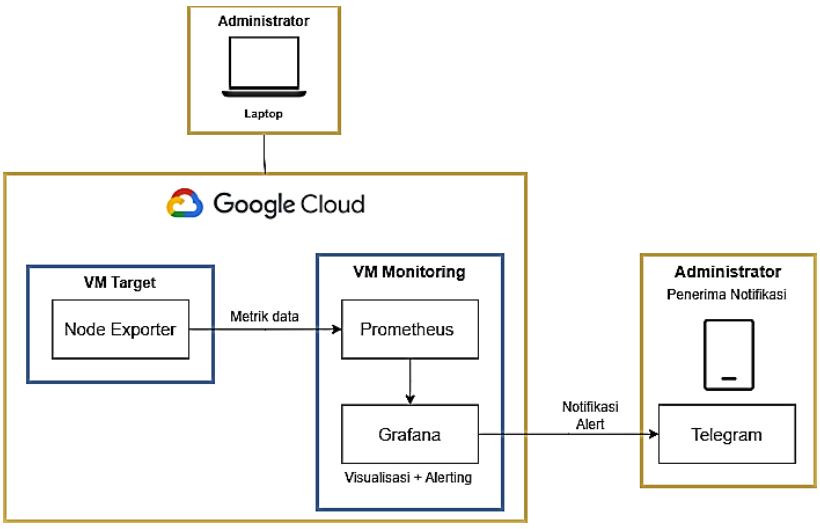
5) Pengujian Sistem

Menjalankan uji fungsionalitas dan *stress testing* untuk memverifikasi kemampuan sistem dalam memantau CPU *utilization* serta mengirim notifikasi otomatis melalui Telegram. Pengujian meliputi uji fungsionalitas dilakukan satu kali untuk memastikan seluruh komponen sistem terintegrasi dan berjalan sesuai perancangan. Stress testing dilakukan melalui beberapa skenario pengujian variasi beban dan durasi, di mana setiap skenario dijalankan satu kali dan menghasilkan beberapa kejadian *alert*akibat fluktuasi CPU. Pendekatn ini digunakan untuk mengevaluasi kecepatan respon dan konsistensi sistem dalam memicu *alert*, mengirim notifikasi, serta melakukan *recovery*.

6) Analisis Hasil Pengujian

Mengevaluasi data hasil uji untuk menilai efektivitas sistem dalam memberikan peringatan *real-time* terhadap lonjakan beban CPU.

Berdasarkan hasil analisis kebutuhan, dirancang arsitektur sistem monitoring yang terdiri dari komponen utama Prometheus, Grafana, Node Exporter, dan Telegram. Arsitektur ini digunakan sebagai acuan dalam proses implementasi pada lingkungan Google Cloud Platform.



Gambar 2. Arsitektur sistem monitoring

Arsitektur yang ditunjukkan pada Gambar 2 kemudian diimplementasikan melalui beberapa tahapan. Proses dimulai dengan (1) instalasi Node Exporter pada VM target untuk menyediakan metrik CPU, memori, dan disk. Selanjutnya, (2) Prometheus dipasang pada VM monitoring dengan konfigurasi prometheus.yml yang memuat alamat IP VM target agar dapat melakukan *scraping* metrik secara periodik. Kemudian, (3) Grafana diintegrasikan dengan Prometheus sebagai *data source* dan dibuat dashboard khusus untuk menampilkan metrik CPU *utilization* secara *real-time*. Tahap berikutnya adalah (4) pembuatan *alert rule* pada Grafana dengan menentukan ambang batas CPU *utilization* sehingga status akan berubah menjadi *Alerting* ketika nilai melewati threshold yang ditetapkan. Untuk mendukung notifikasi otomatis, (5) Telegram Bot dibuat dan dihubungkan ke Grafana melalui *notification channel* menggunakan token API. Dengan konfigurasi ini, setiap alert yang aktif akan langsung dikirim sebagai pesan peringatan ke Telegram. Pada tahap akhir, (6) dilakukan pengujian sistem yang mencakup uji fungsional dan *stress testing*. Uji fungsional memastikan Prometheus mampu melakukan *scraping*, Grafana menampilkan visualisasi dengan benar, serta notifikasi terkirim ke Telegram, sedangkan *stress testing* menggunakan Apache JMeter pada VM pengujian untuk memberikan beban ke *web server* (Nginx) sehingga memicu lonjakan CPU.

Berikut merupakan landasan teori dari penelitian ini:

2.1 *Cloud Computing*

Cloud computing merupakan teknologi jaringan berbasis internet yang menyediakan sumber daya komputasi secara daring[10]. Teknologi ini digunakan dalam penelitian karena mampu menyediakan sumber daya secara fleksibel dan skalabel tanpa memerlukan infrastruktur fisik. Keunggulan *cloud computing* terletak pada kemudahan pengelolaan, efisiensi biaya, serta kemampuan penyesuaian sumber daya sesuai kebutuhan sistem monitoring[11].

2.2 *Google Cloud Platform*

Google Cloud Platform merupakan layanan *cloud computing* yang dikelola oleh Google untuk memenuhi kebutuhan komputasi secara daring[12]. GCP digunakan dalam penelitian ini karena menyediakan layanan Compute Engine yang memungkinkan pembuatan dan pengelolaan *virtual machine* dengan konfigurasi yang fleksibel. Keunggulan GCP terletak pada tingkat reliabilitas, availability, serta dukungan terhadap implementasi sistem monitoring berbasis *cloud*[13], [14].

2.3 *Monitoring*

Monitoring adalah proses pengumpulan dan analisis data secara rutin untuk mengetahui status dan performa suatu sistem secara berkelanjutan[15]. Monitoring digunakan dalam penelitian ini untuk memantau kondisi sumber daya sistem secara *real-time.* Keunggulan sistem monitoring adalah kemampuannya dalam mendeteksi gangguan lebih dini serta membantu pengelolaan dan optimalisasi sumber daya seperti CPU, memori, dan jaringan[16].

2.4 *Prometheus*

Prometheus merupakan alat *open-source* yang umum dirancang untuk pencatatan dan penyimpanan data dalam format deret waktu (*time-series*)[5]. Prometheus digunakan karena memiliki mekanisme *pull-based* yang efisien serta mudah diintegrasikan dengan exporter. Keunggulan Prometheus terletak pada kemampuannya dalam mengumpulkan dan menyimpan metrik performa sistem secara *real-time*, termasuk melalui penggunaan Node Exporter untuk pemantauan tingkat sistem operasi.

2.5 *Grafana*

Grafana merupakan platform *open-source* yang digunakan untuk visualisasi data metrik dari berbagai sumber[8]. Grafana digunakan karena mendukung integrasi langsung dengan Prometheus dan menyediakan dashboard interaktif. Keunggulan Grafana adalah tersedianya fitur *alerting* yang memungkinkan pengiriman notifikasi otomatis ketika metrik mencapai ambang batas tertentu.

2.6 *Telegram*

Telegram merupakan platform komunikasi digital yang mendukung penggunan bot melalui Bot API[17]. Telegram digunakan sebagai media notifikasi karena mampu mengirimkan pesan secara *real-time*. Keunggulan Telegram terletak pada kemudahan integrasi, kecepatan pengiriman pesan, serta dukungan lintas platform, sehingga efektif digunakan dalam sistem monitoring berbasis notifikasi otomatis[18], [19].

2.7 *Apache Jmeter*

Apache JMeter merupakan aplikasi *open-source* yang digunakan untuk pengujian performa dan beban sistem[20]. JMeter digunakan dalam penelitian ini untuk menghasilkan beban terukur pada sistem guna menguji respons monitoring. Keunggulan JMeter adalah kemampuannya mensimulasikan banyak permintaan secara parallel sehingga efektivitas sistem monitoring dalam mendeteksi lonjakan dan mengirim notifikasi dapat dievaluasi [24].

3. HASIL DAN PEMBAHASAN

3.1 *Pengujian fungsional*

Pengujian fungsional bertujuan untuk memastikan bahwa sistem monitoring berjalan sesuai fungsi yang dirancang. Fokus pengujian meliputi verifikasi koneksi dan scraping data metrik oleh Prometheus, visualisasi *real-time* pada Grafana, pemicu *alert* berdasarkan ambang batas CPU *usage*, serta pengiriman notifikasi otomatis ke Telegram.

Tabel 3. Hasil pengujian fungsional sistem monitoring

Komponen Pengujian	*Deskripsi*	*Hasil*
Scraping Prometheus	Verifikasi Prometheus berhasil mengumpulkan metrik dari VM target	Berhasil, data termonitor
Status Node Exporter	Cek status endpoint Node Exporter terdaftar dan “UP” di Prometheus	Terhubung dan aktif
Konfigurasi prometheus.yml	Alamat IP VM target dan port Node Exporter telah ditambahkan di konfigurasi	Terdeteksi
Visualisasi Dashboard Grafana	Panel Grafana menampilkan metrik CPU, Memori dan Disk secara *real-time*	Data tampil dengan benar
Integrasi Data Source Grafana	Grafana berhasil menarik data dari Prometheus sebagai data source utama	Terintegrasi
Trigger Alert > 20 %	Uji ambang batas alert pada CPU Utilization dengan kondisi beban	Alert aktif setelah > 2 menit
Status Alert Grafana	Status alert berubah dari Normal menjadi Alerting saat threshold terlampaui	Terkirim < 5 detik
Format Pesan Telegram	Pesan alert mencantumkan nama alert, nilai CPU, dan status dengan jelas	Format sesuai
Ketersediaan Layanan	Layanan Prometheus dan Grafana tetap berjalan pasca reboot VM monitoring	Tetap aktif tanpa konfigurasi ulang
Scrape Interval Konsisten	Interval pengambilan data sesuai konfigurasi (15 detik per scrape)	Stabil dan sinkron

3.2 *Stress Testing*

3.2.1 *Pengujian Kecepatan*

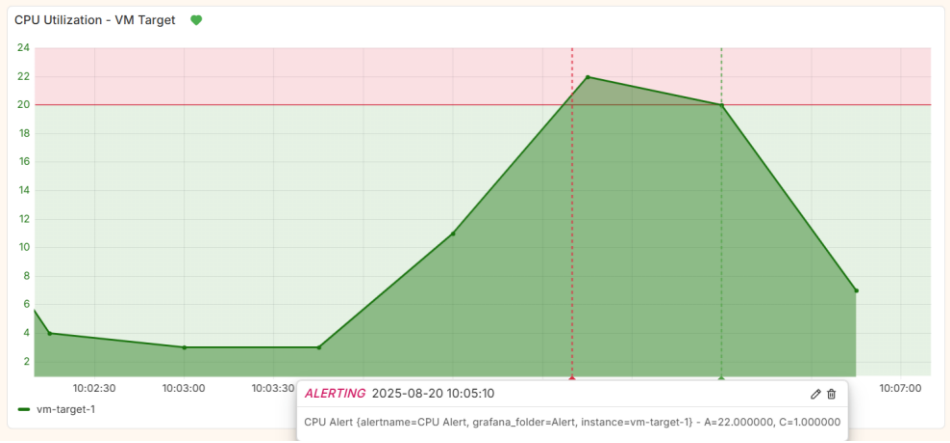
Pengujian fungsional bertujuan untuk memastikan bahwa sistem monitoring berjalan sesuai fungsi yang dirancang. Fokus pengujian meliputi verifikasi koneksi dan scraping data metrik oleh Prometheus, visualisasi *real-time* pada Grafana, pemicu *alert* berdasarkan ambang batas CPU *usage*, serta pengiriman notifikasi otomatis ke Telegram. Pengujian dilakukan dengan variasi jumlah threads 60–120 dan durasi 120–150 detik menggunakan JMeter untuk memicu CPU melewati ambang batas 20%.

Selisih waktu antara status Alerting di Grafana dan notifikasi di Telegram dicatat sebagai indikator kecepatan sistem. Ringkasan hasil pengujian ditunjukkan pada Tabel 2.

Tabel 1. Hasil pengujian kecepatan

No	*Threads*	*Duration*	*Waktu*	
			Grafana	*Telegram*
1	60	120s	04:14:10	04:14:11
2	70	120s	20:33:00	20:33:11
3	80	120s	20:43:50	20:43:51
4	100	150s	04:51:00	04:51:01
5	120	150s	10:25:20	10:25:21

Selanjutnya, Gambar 3 menunjukkan tampilan status *Alerting* pada Grafana saat CPU usage melewati ambang batas 20%, sedangkan Gambar 4 memperlihatkan notifikasi otomatis yang dikirim melalui Telegram. Kedua bukti visual ini mengonfirmasi kecepatan dan akurasi sistem dalam merespons lonjakan beban. Pada Gambar 3 memperlihatkan grafik CPU *usage* pada VM target yang meningkat hingga melewati ambang batas 20%. Perubahan status ditandai dengan garis merah (*Alerting*) pada dashboard Grafana, yang menunjukkan sistem berhasil mendeteksi lonjakan beban.



Gambar 3. Status alert Grafana

Pada Gambar 4 menampilkan pesan peringatan otomatis yang dikirim oleh sistem ke aplikasi Telegram saat CPU *usage* melampaui ambang batas. Notifikasi berisi detail nilai CPU aktual serta tautan menuju dashboard Grafana, sehingga administrator dapat segera melakukan verifikasi.

Gambar 4. Notifikasi Telegram

3.2.2 *Pengujian Stabilitas*

Pengujian stabilitas dilakukan untuk memastikan sistem monitoring mampu memberikan peringatan secara konsisten, baik pada kondisi beban singkat maupun beban yang berlangsung lama. Pada tahap ini digunakan JMeter sebagai load generator dengan konfigurasi sesuai skenario pengujian. Rincian parameter uji ditunjukan pada Tabel 3.

Tabel 2. Parameter uji JMeter

Skenario Uji	*Threads*	*Ramp-up*	*Duration*
Threshold Validation	60	30s	180s
Long Hold (10 menit)	120	30s	600s

Pengujian stabilitas dilakukan untuk memastikan sistem monitoring mampu memberikan peringatan secara konsisten, baik pada kondisi beban singkat maupun beban yang berlangsung lama. Pada tahap ini digunakan JMeter sebagai *load generator* dengan konfigurasi sesuai skenario pengujian. Rincian parameter uji ditunjukan pada Tabel 3.

Tabel 3. Hasil pengujian stabilitas

Skenario Uji	*Status*	*Waktu*	*CPU Usage (%)*
Threshold Validation	*Trigger*	8:36:00	22
	Notifikasi (1)	8:36:16	28
	Notifikasi (2)	8:37:21	37
	Recover	8:37:50	19
	Normal	8:38:50	18
Long Hold (10 menit)	*Alerting*	8:58:00	22
	Notifikasi (1)	8:58:16	28
	Notifikasi (2)	8:59:21	28
	Recover	8:59:40	20
	Alerting	9:02:20	23
	Notifikasi	9:02:36	23
	Recover	9:03:00	20
	Alerting	9:03:10	21
	Recover	9:03:30	20
	Normal	9:03:40	19

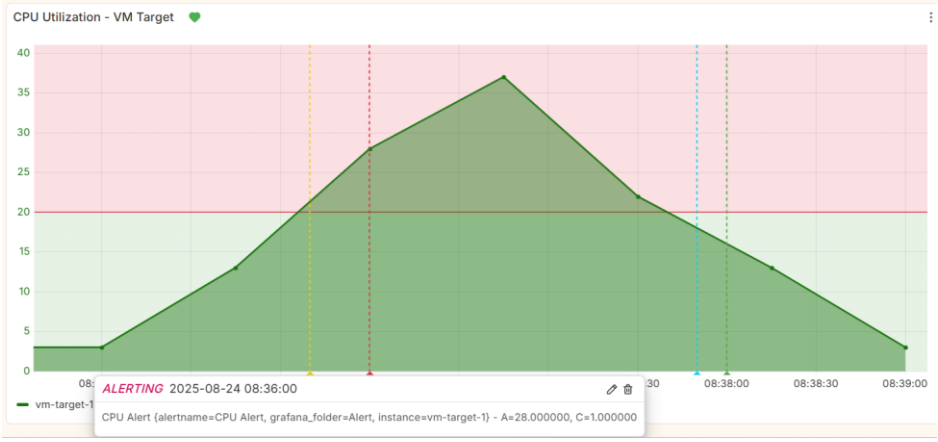
Pengujian menunjukkan sistem mengirim notifikasi sesuai konfigurasi, yaitu pending period 20 detik, *repeat interval* 1 menit, dan *keep firing* 10 detik. Pada kondisi beban tinggi, *alert* aktif, notifikasi terkirim ke Telegram, lalu sistem melakukan *recovery* saat CPU turun di bawah ambang batas. Tabel 5 Rekapitulasi Hasil Pengujian Sistem Monitoring.

Tabel 4. Rekapitulasi hasil pengujian sistem monitoring

Skenario Uji	*Jumlah Alert*	*CPU Max (%)*	*CPU Min (%)*	*Rata-rata CPU*
Threshold Validation	1 kali	37	22	29
Long Hold (10 menit)	3 kali	28	21	23.3

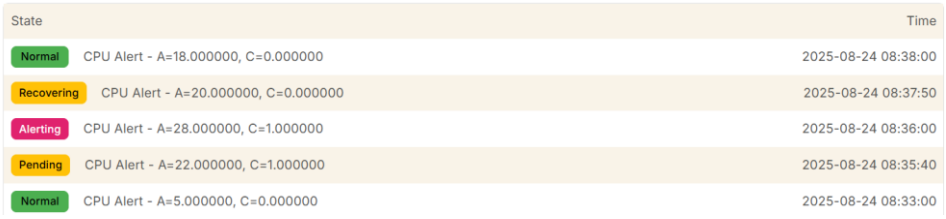
Berdasarkan Tabel 12, pada skenario *Threshold Validation* terjadi satu *alert* dengan CPU rata-rata 29% dan puncak 37%, menunjukkan sistem mampu mendeteksi lonjakan singkat. Pada skenario Long Hold (10 menit) tercatat tiga *alert* dengan CPU rata-rata 23,3%, menandakan mekanisme *alerting* tetap konsisten meski beban lebih rendah. Hasil ini menunjukkan sistem stabil dalam mendeteksi alert baik pada beban singkat maupun lama, dengan status *alert* terlihat aktif di Grafana dan kembali normal saat CPU turun.

Pada skenario uji *threshold validation* menghasilkan tampilan dashboard seperti pada Gambar 5.



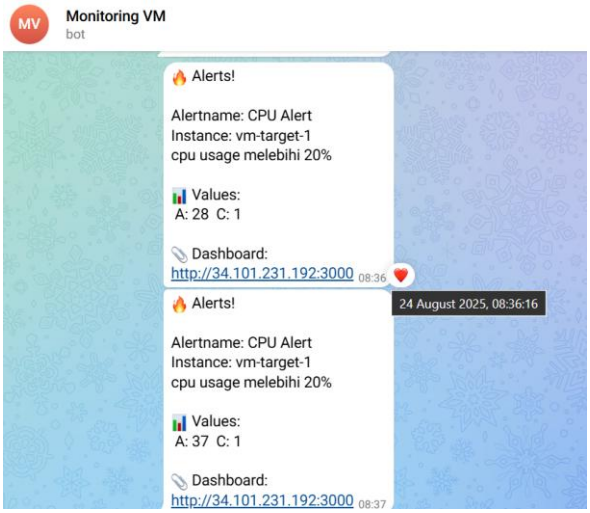
Gambar 1. Dashboard Threshold Validation

Kemudian jika pada Dashboard sudah terdapat status "alerting" maka seharusnya masuk ke history alert *rule* seperti pada Gambar 6 berikut.



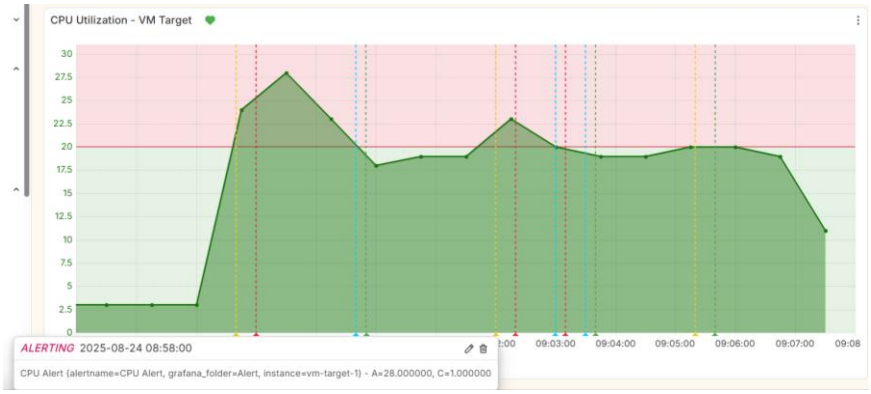
Gambar 2. History Alert Rule

Selain terdapat Riwayat, sama seperti uji kecepatan sebelumnya. Uji stabilitas *threshold validation* juga mendapat notifikasi. Kali ini mendapat dua notifikasi karena hasil pengujian memenuhi evaluasi selama dua kali. Berikut tampilan notifikasi ada pada Gambar 7 yang menampilkan dua notifikasi yang diterima sesuai konfigurasi *pending period* dan *repeat interval*.



Gambar 3. Notifikasi Telegram Uji Threshold Validation

Pada Gambar 8 Grafik CPU *usage* menunjukkan beberapa siklus alerting selama 10 menit, dengan status alert aktif saat CPU berada di atas *threshold*.



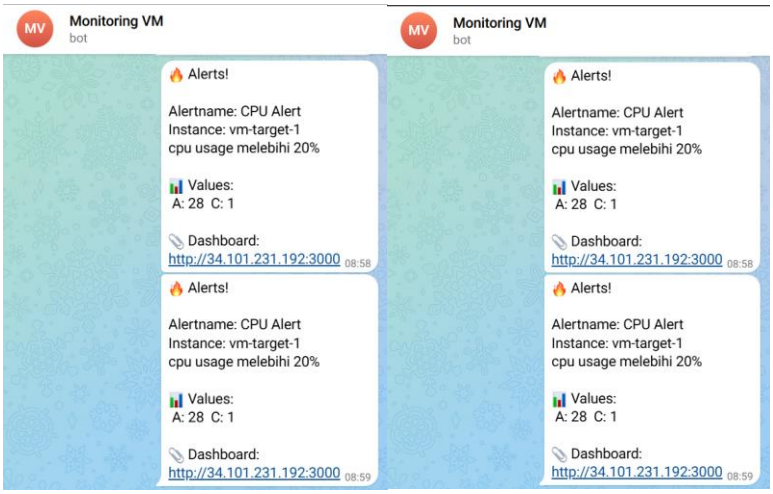
Gambar 4. Dashboard Long Hold

Pada Gambar 9 Menampilkan urutan perubahan status alert sesuai fluktuasi CPU: Normal → Pending → Alerting → Recovering → Normal, berulang beberapa kali.

Normal	CPU Alert - A=19.000000, C=0.000000	2025-08-24 09:03:40
Recovering	CPU Alert - A=20.000000, C=0.000000	2025-08-24 09:03:30
Alerting	CPU Alert - A=21.000000, C=1.000000	2025-08-24 09:03:10
Recovering	CPU Alert - A=20.000000, C=0.000000	2025-08-24 09:03:00
Alerting	CPU Alert - A=23.000000, C=1.000000	2025-08-24 09:02:20
Pending	CPU Alert - A=23.000000, C=1.000000	2025-08-24 09:02:00
Normal	CPU Alert - A=19.000000, C=0.000000	2025-08-24 08:59:50
Recovering	CPU Alert - A=20.000000, C=0.000000	2025-08-24 08:59:40
Alerting	CPU Alert - A=28.000000, C=1.000000	2025-08-24 08:58:00
Pending	CPU Alert - A=22.000000, C=1.000000	2025-08-24 08:57:40
Normal	CPU Alert - A=13.000000, C=0.000000	2025-08-24 08:54:20

Gambar 5. Riwayat Alert Rule Long Hold

Notifikasi periodik yang dikirim selama status alert aktif, mengikuti konfigurasi *pending period*, *repeat interval*, dan *group wait* ditunjukkan pada Gambar 10.



Gambar 6. Notifikasi Telegram Long Hold

4. KESIMPULAN

Berdasarkan hasil implementasi dan pengujian, sistem monitoring berbasis Prometheus, Grafana, dan Telegram Bot yang dibangun pada infrastruktur Google Cloud Platform memberikan kinerja sesuai tujuan penelitian. Kesimpulan utama adalah:

1. Sistem berhasil memantau penggunaan CPU pada VM secara *real-time*, menampilkan data melalui dashboard Grafana, dan mengirimkan notifikasi otomatis ke Telegram ketika CPU *usage* melebihi ambang batas 20% yang telah ditentukan.
2. Semua komponen berjalan optimal. Prometheus konsisten melakukan scraping metrik, Grafana menyajikan visualisasi *real-time*, dan notifikasi Telegram terkirim rata-rata kurang dari 5 detik setelah status alert aktif. Format pesan yang dikirim mencantumkan informasi metrik dan tautan dashboard.
3. Pengujian *stress* menunjukkan selisih rata-rata sekitar 1 detik antara status alert di Grafana dan notifikasi di Telegram, membuktikan sistem hampir *real-time*. Hanya terjadi satu anomali *delay* hingga 11 detik pada skenario 70 threads, akibat mekanisme evaluasi metrik, bukan kegagalan sistem.
4. Sistem tetap stabil saat CPU berada di atas *threshold* untuk periode lama. Pada skenario Long Hold, rata-rata CPU usage mencapai 23,3%, notifikasi terkirim sesuai konfigurasi repeat interval tanpa menimbulkan alert storm, dan proses pemulihan berlangsung normal saat CPU kembali di bawah 20%.

Secara keseluruhan, konfigurasi sistem monitoring ini terbukti optimal, responsif, dan andal untuk mendukung deteksi dini serta respons cepat terhadap anomali CPU di lingkungan produksi. Sistem memiliki potensi untuk dikembangkan lebih lanjut dengan memperluas pemantauan tidak hanya pada CPU, tetapi juga memori, disk dan ketersediaan jaringan. Selain itu, penyempurnaan dashboard Grafana serta penerapan otomatisasi instalasi dan pemisahan VM dapat meningkatkan kemudahan pengelolaan, keamanan, dan kinerja sistem secara keseluruhan, sehingga sistem tidak hanya memantau kondisi secara *real-time* tetapi juga mampu mendukung pengambilan Keputusan yang lebih cerdas dan proaktif.

DAFTAR PUSTAKA

[1] A. Marchenko and D. Shchemelinin, "Development of an Accessibility Testing System for the Virtual Machine Deployment Service in the Cloud," *Proc. of Telecommunication Universities*, vol. 9, no. 3, pp. 68–73, July 2023, doi: 10.31854/1813-324X-2023-9-3-68-73.

[2] Google Cloud, "View and Understand VM Instance Insights," Compute Engine Documentation. Accessed: Aug. 24, 2025. [Online]. Available: https://cloud.google.com/compute/docs/instances/view-and-understand-vm-insights

[3] K. Fitzgerald, "Radically Reduce Downtime and Data Loss with SaaS-based Disaster Recovery," CIO. Accessed: Aug. 24, 2025. [Online]. Available: https://www.cio.com/article/410036/radically-reduce-downtime-and-data-loss-with-saas-based-disaster-recovery.html

[4] P. B.C., H. Maddirala, and S. M., "Implementing an Effective Infrastructure Monitoring Solution with Prometheus and Grafana," *IJCA*, vol. 186, no. 38, pp. 7–15, Sept. 2024, doi: 10.5120/ijca2024923873.

[5] A. P. Putra, G. Sukadarmika, and D. M. Wiharta, "Model Utilisasi dan Visualisasi Resource Menggunakan Prometheus dan Grafana Untuk Pengelolaan Server di Universitas Udayana," *JTE*, vol. 22, no. 2, p. 305, Jan. 2024, doi: 10.24843/MITE.2023.v22i02.P19.

[6] M. Bajpai, "Automating Monitoring and Incident Management with Prometheus, Grafana, and Google Cloud Pub/Sub," *IJSR*, vol. 11, no. 1, pp. 1673–1675, Jan. 2022, doi: 10.21275/SR24829151754.

[7] G. Y. Kusuma and U. Y. Oktiawati, "Application Performance Monitoring System Design Using Opentelemetry and Grafana Stack," *Journal of Internet and Software Engineering*, vol. 3, no. 1, Art. no. 1, Nov. 2022, doi: 10.22146/jise.v3i1.5000.

[8] Y. Jani, "Unified Monitoring for Microservices: Implementing Prometheus and Grafana for Scalable Solutions," *JAIMLD*, vol. 2, no. 1, pp. 848–852, Mar. 2024, doi: 10.51219/JAIMLD/yash-jani/206.

[9] M. D. Elradi, "Prometheus & Grafana: A Metrics-focused Monitoring Stack," *Journal of Computer Allied Intelligence(JCAI, ISSN: 2584-2676)*, vol. 3, no. 3, pp. 28–39, June 2025, doi: 10.69996/jcai.2025015.

[10] D. Gustian, Y. Fitrisia, W. Novayani, and S. Purwantoro, "Implementasi Automation Deployment pada Google Cloud Compute VM Menggunakan Terraform," *ISI*, vol. 8, no. 1, p. 50, June 2023, doi: 10.35314/isi.v8i1.3095.

[11] M. Kondoj, H. Langi, Y. Putung, and V. Lengkong, "Performance Analysis of Cloud Computing Based E-Commerce Server Using PROXMOX Virtual Environment," in *Proc. 5th Int. Conf. Appl. Sci. Technol. Eng. Sci. (iCAST-ES)*, SCITEPRESS, 2022, pp. 741–745. doi: 10.5220/0011876000003575.

[12] "Manfaat Google Cloud Compute Engine," Elitery. Accessed: May 02, 2025. [Online]. Available: https://elitery.com/articles/manfaat-google-cloud-compute-engine/

[13] Praveen Borra, "A Survey of Google Cloud Platform (GCP): Features, Services, and Applications," *IJARSCT*, vol. 4, no. 3, pp. 191–199, June 2024, doi: 10.48175/IJARSCT-18922.
[14] N. Ramsari and A. Ginanjar, "Implementasi Infrastruktur Server Berbasis Cloud Computing Untuk Web Service Berbasis Teknologi Google Cloud Platform," in *Conference SENATIK STT Adisutjipto Yogyakarta*, Mar. 2022, pp. 169–182. doi: 10.28989/senatik.v7i0.472.
[15] Z. Nurrifa'at and M. N. Dasaprawira, "Pengembangan Aplikasi Monitoring PKL dengan Firebase Menggunakan Metode Agile (Studi Kasus: Fakultas Fmikom Unugha)," vol. 8, no. 3, 2024.
[16] P. K. G. Pandian, "Effective Resource Management In Virtualized Environments," vol. 1, no. 7, 2023.
[17] "Bots: An Introduction for Developers," Telegram. Accessed: May 02, 2025. [Online]. Available: https://core.telegram.org/bots
[18] F. Fitriansyah, "Penggunaan Telegram sebagai Media Komunikasi dalam Pembelajaran Online," *Jurnal Humaniora*, vol. 20, no. 2, pp. 111–117, 2020.
[19] N. C. Dewi, T. Sutabri, and F. Putrawansyah, "Analisis Penyadapan pada Telegram dengan Network Forensic," *JIKO (Jurnal Informatika dan Komputer)*, vol. 7, no. 2, pp. 183–190, Sept. 2023, doi: 10.26798/jiko.v7i2.789.