Manajemen Log Di Cluster Kubernetes Menggunakan Fluentd Dan Elasticsearch

I Gede Megantara
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
igedem@student.telkomuniversity.ac.id

Jafaruddin Gusti Amri Ginting
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
jafargustiamri@telkomuniversity.ac.id

Fauza Khair
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
fauzakhair@telkomuniversity.ac.id

Abstrak — Perkembangan teknologi digital membutuhkan infrastruktur Teknologi Informasi yang mumpuni, termasuk adopsi arsitektur mikroservis dan kontainerisasi yang mendorong penggunaan Kubernetes sebagai orkestrator kontainer. Kubernetes memudahkan pengelolaan dan penyebaran aplikasi kompleks, namun juga meningkatkan kompleksitas pengelolaan log dari berbagai kontainer. Log ini mencakup aktivitas aplikasi, performa sistem, dan pesan debugging yang penting untuk memahami kinerja aplikasi, mendeteksi masalah, dan memenuhi kebutuhan keamanan. Penelitian ini berfokus pada manajemen log terdistribusi di lingkungan Kubernetes menggunakan Fluentd dan Elasticsearch. Fluentd berperan sebagai agen pengumpul log yang efisien, sementara Elasticsearch menyediakan penyimpanan, pengindeksan, dan analisis log yang cepat. Kombinasi keduanya diharapkan meningkatkan efisiensi, ketersediaan, dan keamanan infrastruktur cloud yang kompleks. Metode penelitian mencakup implementasi dua skenario utama yaitu pengumpulan log secara manual menggunakan perintah kubectl logs pada satu pod dan 20 pod untuk mendapatkan performa, dan pengumpulan log terdistribusi menggunakan Fluentd yang terintegrasi dengan Elasticsearch di cluster Kubernetes. Evaluasi dilakukan berdasarkan waktu pencarian log, efisiensi pengumpulan log, dan kemudahan analisis menggunakan visualisasi dari Kibana. Hasil penelitian menunjukkan bahwa pengumpulan log menggunakan Fluentd dan Elasticsearch secara signifikan lebih efisien dibandingkan metode manual.

Kata Kunci : Elasticsearch, Fluentd, infrastruktur TI , kontainerisasi, Kubernetes, manajemen log, mikroservis

Abstract - The development of digital technology requires a capable Information Technology infrastructure, including the adoption of microservice architecture and containerization that encourages the use of Kubernetes as a container orchestrator. Kubernetes makes it easier to manage and deploy complex applications, but also increases the complexity of managing logs from various containers. These logs include application activity, system performance, and debugging messages that are important for understanding application performance, detecting problems, and meeting security needs. This study focuses on distributed log management in a Kubernetes environment using Fluentd and Elasticsearch. Fluentd acts as an efficient log collection agent, while Elasticsearch provides fast log storage, indexing, and analysis. The combination of the two is expected to improve the efficiency, availability, and security of complex cloud infrastructure. The research method includes the implementation of two main scenarios, namely manual log collection using the kubectl logs command on one pod and 20 pods to get performance, and distributed log collection using Fluentd integrated with Elasticsearch in a Kubernetes cluster. The evaluation was carried out based on log search time, log collection efficiency, and ease of analysis using visualization from Kibana. The results of the study show that log collection using Fluentd and Elasticsearch is significantly more efficient than the manual method. Keywords:

Containerization, Elasticsearch, Fluentd, IT infrastructure, Kubernetes, log management, microservices.

I. PENDAHULUAN

Perkembangan Teknologi yang sangat cepat di era digital saat ini tentu akan diperlukan infrastruktur dilayanan Teknologi Informasi. Dengan adanya pergeseran ke arah arsitektur mikroservis dan kontainerisasi, penggunaan kubernetes sebagai orkestrator kontainer telah berkembang pesat. Kubernetes memungkinkan pengelolaan dan penyebaran aplikasi yang kompleks dengan cepat dan efisien. Namun, pertumbuhan ini juga diikuti oleh meningkatnya kompleksitas log dari berbagai kontainer yang berjalan dalam cluster kubernetes [1]. Kubernetes merupakan sebuah platform open-source yang mampu melakukan management clustering container dalam jumlah besar atau bisa disebut sebagai cluster orchestration yang memiliki tugas melakukan penjadwalan, scaling, recovery dan monitoring container [2]. Klaster kubernetes memungkinkan organisasi untuk mudah mengelola, mendistribusikan, dan memperluas penyebaran aplikasi sesuai dengan kebutuhan. Dalam lingkungan klaster Kubernetes, load balancing memiliki peran penting dalam mendistribusikan lalu lintas pengguna secara merata dan mengoptimalkan sumber daya. Dalam suatu ekosistem Kubernetes, setiap kontainer yang berjalan menghasilkan log yang signifikan, mencakup aktivitas aplikasi, performa sistem, dan pesan debugging. Log ini merupakan sumber informasi penting yang mencerminkan kesehatan sistem, status operasional aplikasi, serta deteksi dini terhadap potensi masalah atau anomali. Dengan pertumbuhan skala dan kompleksitas lingkungan Kubernetes, manajemen log menjadi semakin krusial, terutama untuk memahami kinerja aplikasi, mendeteksi masalah yang muncul, dan memenuhi kebutuhan keamanan seperti audit, pelacakan aktivitas, dan kepatuhan terhadap standar tertentu. Namun, dengan pertumbuhan infrastruktur yang semakin besar dan dinamis, tantangan dalam mengelola log menjadi semakin kompleks. Setiap kontainer, pod, dan node di dalam kluster Kubernetes menghasilkan volume log yang besar dan terus bertambah seiring dengan peningkatan jumlah layanan atau aplikasi yang dikelola. Data log ini tidak hanya berasal dari aplikasi, tetapi juga mencakup aktivitas sistem, seperti komunikasi antar-pod, aktivitas jaringan, serta interaksi dengan penyimpanan atau sumber daya lainnya. Selain itu, log tersebut tersebar di berbagai lokasi dalam kluster, yang membuat pengelolaannya menjadi sulit jika dilakukan secara manual. Kendala lain yang muncul adalah bagaimana mengelola log secara efisien di tengah skala besar. Volume log yang besar memerlukan sistem penyimpanan yang andal serta mekanisme untuk memproses, mengindeks, dan

menganalisis data secara real-time. Dalam situasi ini, pendekatan tradisional yang mengandalkan pengumpulan log manual atau pencatatan lokal tidak lagi memadai. Setiap perubahan kecil dalam aplikasi dapat menghasilkan ribuan baris log baru, sehingga dibutuhkan solusi terpusat untuk mengintegrasikan dan mengelola log secara efisien. Dengan pertumbuhan skala dan kompleksitas lingkungan Kubernetes, manajemen log menjadi semakin krusial untuk memahami kinerja aplikasi, mendeteksi masalah, dan memenuhi kebutuhan keamanan. Namun, dengan pertumbuhan infrastruktur yang semakin besar dan kompleks, mengelola log (catatan aktivitas sistem dan aplikasi) di dalam lingkungan kubernetes menjadi semakin menantang. Setiap kontainer, pod, dan node dalam cluster kubernetes menghasilkan volume log yang besar, yang mencakup informasi penting tentang kinerja aplikasi, aktivitas pengguna, dan masalah sistem yang perlu diatasi [3]. Pengelolaan log di lingkungan kubernetes memerlukan solusi yang mampu mengumpulkan, menyimpan, dan menganalisis log dengan efisien dan efektif. Salah satu pendekatan yang telah banyak digunakan dalam mengatasi tantangan ini adalah dengan menggabungkan alat-alat khusus seperti Fluentd dan Elasticsearch [4]. Fluentd adalah agen pengumpul log open-source yang dirancang untuk beroperasi dalam lingkungan terdistribusi. Dengan kemampuannya untuk mengumpulkan log dari berbagai sumber yang tersebar di dalam cluster Kubernetes, Fluentd menjadi salah satu komponen kunci dalam infrastruktur manajemen log terdistribusi. Sementara itu, Elasticsearch adalah mesin pencarian dan analisis log yang kuat. Dengan kemampuannya untuk menyimpan, mengindeks, dan memungkinkan pencarian cepat terhadap volume data log yang besar, Elasticsearch menjadi solusi yang ideal untuk menyediakan wawasan yang mendalam tentang kinerja aplikasi dan infrastruktur di dalam lingkungan kubernetes [5]. Dalam konteks inilah penelitian tentang "Manajemen Log Terdistribusi di Cluster Kubernetes Menggunakan Fluentd dan Elasticsearch" menjadi relevan. Penelitian ini bertujuan untuk menginvestigasi dan mengembangkan solusi yang efektif untuk pengelolaan log di lingkungan kubernetes, dengan memanfaatkan potensi dari Fluentd dan Elasticsearch. Pada penelitian sebelumnya [6], telah dibuat monitoring pada kluster kubernetes menggunakan Prometheus dan Grafana, dimana Prometheus digunakan sebagai agen untuk mengambil data dari kluster Kubernetes. Prometheus digunakan sebagai agen untuk mengambil data dari kluster Kubernetes dan menyimpan metrik tersebut ke dalam basis data time-series yang mendukung visualisasi oleh Grafana. Kombinasi Prometheus dan Grafana menunjukkan hasil yang cukup efektif dalam memonitor performa aplikasi di lingkungan Kubernetes, terutama untuk memantau metrik seperti penggunaan CPU, memori, dan status pod. Namun, solusi ini terbatas pada monitoring metrik dan tidak secara khusus berfokus pada pengelolaan log aplikasi atau sistem, yang merupakan aspek penting untuk mendeteksi anomali, debugging, dan audit keamanan. Pada penelitian [7] monitoring log menggunakan logtash sebagai engine pengumpul data dimana pada logtash ini memiliki kekurangan dari sisi support plugin yang tidak terlalu luas dan penggunaan resources pada server yang tinggi sehingga akan membebani server. Logstash mampu memproses data log dengan pipeline yang dapat dikustomisasi, tetapi memiliki beberapa kekurangan signifikan. Salah satunya adalah dukungan plugin yang tidak terlalu luas, sehingga membatasi fleksibilitas dalam mengintegrasikan berbagai jenis sumber log. Selain itu, penggunaan sumber daya server oleh Logstash relatif tinggi, yang dapat menyebabkan overhead besar pada sistem, terutama di lingkungan dengan jumlah log yang masif. Hal ini dapat menghambat performa sistem utama, membuatnya kurang efisien untuk digunakan di lingkungan cloud yang membutuhkan efisiensi tinggi. Perbedaan penelitian ini dengan penelitian sebelumnya adalah pada penelitian ini agen untuk mengumpulkan log atau data dari server adalah fluentd dimana dengan fluentd yang mendukung plugin lebih luas dan resources server yang digunakan tidak terlalu tinggi. Dalam penelitian ini dilakukan dengan menguji implementasi Fluentd sebagai agen pengumpul log dan Elasticsearch sebagai sistem penyimpanan dan analisis log. Pengujian dilakukan dengan membandingkan pencarian log secara manual menggunakan perintah Kubernetes (kubectl logs) dan pencarian log menggunakan Fluentd yang diteruskan ke Elasticsearch. Pengukuran dilakukan untuk mengevaluasi latensi pengiriman log, akurasi waktu pengumpulan log, kelengkapan log, penggunaan sumber daya. Diharapkan penelitian ini dapat memberikan kontribusi signifikan dalam meningkatkan efisiensi, ketersediaan, dan keamanan infrastruktur cloud dalam menghadapi tuntutan yang semakin kompleks dan dinamis

II. KAJIAN TEORI

A. Linux

Linux server adalah komputer yang menjalankan sistem operasi Linux, yang terkenal karena stabilitas, keandalan, dan fleksibilitasnya. Server ini digunakan untuk berbagai keperluan, seperti hosting situs web, penyimpanan data, server aplikasi, dan layanan jaringan lainnya. Linux server dapat diimplementasikan di berbagai lingkungan, mulai dari pusat data perusahaan hingga layanan cloud publik dan privat.

B. Kubernetes

Kubernetes adalah platform open-source yang dirancang untuk mengotomatiskan deployment, skalabilitas, dan manajemen aplikasi berbasis kontainer. Dikembangkan oleh Google dan dirilis sebagai proyek open-source pada 2014, Kubernetes kini menjadi fondasi utama dalam infrastruktur cloud modern. Platform ini memungkinkan pengelolaan aplikasi dalam kontainer secara efisien dan skalabel, dengan fitur seperti load balancing, orkestrasi pod, dan rolling updates yang memastikan aplikasi tetap berjalan tanpa gangguan.

C. Log Management

Log Management adalah proses mengumpulkan, mengelola, menganalisis, menyimpan, dan memantau log dari sistem komputer, aplikasi, dan perangkat jaringan. Log ini mencatat berbagai aktivitas, seperti penggunaan sistem, perubahan konfigurasi, akses data, serta kejadian keamanan. Dengan menganalisis log, organisasi dapat mengidentifikasi dan menyelesaikan masalah operasional, mendeteksi pelanggaran kebijakan, serta menangani insiden keamanan.

D. Elasticsearch

Elasticsearch adalah mesin pencarian dan analisis log yang dirancang untuk menyimpan, mencari, dan menganalisis data dalam skala besar secara real-time. Sebagai sistem open-source, Elasticsearch sering digunakan dalam EFK dan ELK stack serta berfungsi sebagai database NoSQL yang berfokus pada pencarian data. Dalam strukturnya, indeks di Elasticsearch dapat dianggap sebagai database, types sebagai tabel, dokumen sebagai record, dan mapping sebagai skema tabel.

E. Fluentd

Fluentd adalah agen pengumpul log open-source yang dikembangkan oleh Treasure Data. Alat ini berfungsi untuk mengumpulkan, memproses, dan meneruskan log dari berbagai sumber ke berbagai tujuan, termasuk Elasticsearch dan AWS S3. Dengan lebih dari 500 plugin yang tersedia, Fluentd mampu melakukan parsing, transformasi, dan routing data log secara fleksibel.

F. Kibana

Kibana adalah sistem front-end open-source yang berfungsi sebagai antarmuka untuk mencari, memvisualisasikan, dan mengelola data yang diindeks dalam Elasticsearch. Dikembangkan sejak 2013 dalam komunitas Elastic, Kibana kini menjadi portal utama bagi Elastic Stack, memungkinkan pengguna untuk memonitor dan mengamankan klaster Elasticsearch.

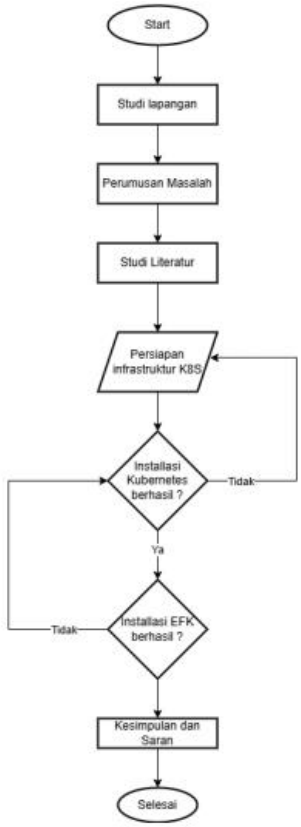
Kibana bekerja dengan membaca serta mengolah visualisasi dari indeks Elasticsearch, termasuk data yang dikumpulkan oleh alat seperti Logstash, Beats, Fluentd, dan Fluent Bit. Fitur utamanya meliputi dashboard interaktif, pembuatan bagan dan grafik beragam, serta kemampuan analisis data log secara real-time. Pengguna dapat menampilkan data dalam bentuk pie chart, bar chart, heatmap, serta menggunakan alat seperti Canvas untuk presentasi dinamis dan Lens untuk visualisasi yang lebih intuitif.

G. Kluster Manajemen

Klasterisasi adalah proses mengelompokkan beberapa komputer atau server agar bekerja bersama sebagai satu kesatuan guna meningkatkan ketersediaan, skalabilitas, dan kinerja sistem. Dalam klaster, setiap node dapat beroperasi secara mandiri tetapi tetap terkoordinasi untuk memberikan layanan yang stabil dan efisien.

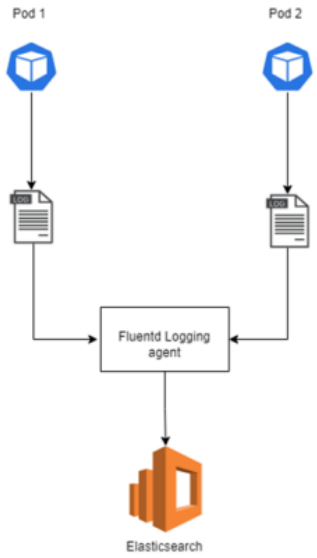
B. METODE

A. Diagram Alir Pengerjaan



Gambar diatas menunjukkan diagram alur penelitian yang terdiri dari beberapa tahap, dimulai dari studi lapangan. Tahap ini dilakukan untuk mengidentifikasi permasalahan yang ada, terutama terkait dengan pesatnya perkembangan teknologi microservices yang sering kali tidak diimbangi dengan implementasi sistem manajemen log yang baik. Permasalahan ini dapat berdampak pada kesulitan dalam memantau performa dan mengidentifikasi masalah di lingkungan microservices. Setelah studi lapangan, dilakukan perumusan masalah untuk merinci isu utama yang ingin diselesaikan melalui penelitian ini. Tahap berikutnya adalah studi literatur untuk mengumpulkan informasi dari berbagai referensi yang relevan terkait sistem manajemen log, teknologi Kubernetes, dan stack EFK (Elasticsearch, Fluentd, dan Kibana). Selanjutnya adalah tahap persiapan infrastruktur K8S yang mencakup penyediaan perangkat keras atau virtualisasi, konfigurasi jaringan, dan kebutuhan lainnya untuk mendukung instalasi sistem. Apabila infrastruktur telah siap, dilakukan instalasi cluster Kubernetes. Jika instalasi Kubernetes berhasil, penelitian dilanjutkan dengan instalasi stack EFK untuk memungkinkan pengumpulan, pemrosesan, dan visualisasi log di dalam cluster Kubernetes. Jika terdapat kendala pada salah satu tahap instalasi, penelitian akan kembali ke tahap sebelumnya untuk memperbaiki permasalahan hingga berhasil. Tahap terakhir dari alur penelitian adalah penyusunan kesimpulan dan saran. Pada tahap ini, peneliti akan mengevaluasi hasil instalasi dan implementasi sistem, serta memberikan rekomendasi untuk pengembangan dan optimalisasi sistem manajemen log di masa depan. Diagram ini menggambarkan alur penelitian yang sistematis untuk memastikan keberhasilan implementasi sistem log berbasis Kubernetes dan EFK

B. Blok Diagram Sistem Logging Fluentd



Gambar diatas menggambarkan blok diagram sistem manajemen log pada Kubernetes yang dirancang untuk penelitian ini. Sistem ini menggunakan Fluentd sebagai agen pengumpul log untuk mengelola log dari berbagai aplikasi yang berjalan di dalam setiap pod Kubernetes. Setiap pod menghasilkan log sistem dan log aplikasi, seperti log status layanan, informasi error, dan aktivitas lainnya. Fluentd, yang diinstal sebagai agen pada setiap pod, berfungsi untuk mengumpulkan, memproses, dan meneruskan log dari pod-pod tersebut. Setelah dikumpulkan oleh agen Fluentd, log dikirim ke Elasticsearch, yang bertugas untuk menyimpan log tersebut dalam format yang terstruktur dan mudah diakses. Elasticsearch juga memungkinkan pengguna untuk menjalankan query yang kompleks terhadap data log yang tersimpan, sehingga mendukung analisis data log secara mendalam. Data log yang telah diindeks kemudian dapat divisualisasikan menggunakan Kibana, yang memungkinkan pengguna untuk membuat berbagai jenis grafik, dashboard interaktif, dan laporan berdasarkan log yang telah terkumpul. Diagram ini mencerminkan alur kerja sistem manajemen log yang efisien, mulai dari pengumpulan log di tingkat pod Kubernetes hingga penyimpanan dan visualisasi log di Elasticsearch dan Kibana. Dengan pendekatan ini, sistem tidak hanya mendukung pengelolaan log secara 22 real-time, tetapi juga mempermudah proses pemantauan, analisis performa, dan identifikasi masalah pada cluster Kubernetes.

C. PENGUMPULAN DAN PENGOLAHAN DATA

A. Implementasi Fluentd pada Kubernetes

Proses implementasi dimulai dengan pengaturan cluster Kubernetes sebagai wadah untuk aplikasi berjalan, termasuk konfigurasi namespace khusus efkmonitoring untuk memisahkan log pipeline dari workload lainnya. Fluentd diinstal sebagai agen pengumpulan log yang bertugas membaca log dari pod, node, dan aplikasi, kemudian memformatnya sesuai kebutuhan sebelum diteruskan ke Elasticsearch. Selanjutnya, Elasticsearch dideploy untuk menyimpan data log dalam bentuk dokumen yang dapat diindeks dan dicari dengan cepat. Sistem ini diperkuat dengan Kibana sebagai antarmuka visualisasi untuk menganalisis data log yang tersimpan. Seluruh komponen dikonfigurasi untuk berjalan dalam mode yang saling terintegrasi, memungkinkan log dari berbagai sumber dapat diteruskan, diproses, dan disajikan dalam format yang mendukung troubleshooting dan analisis performa aplikasi. Proses validasi dilakukan untuk memastikan sistem berjalan sesuai ekspektasi, dengan pengujian yang mencakup pengiriman log dari Fluentd ke Elasticsearch, ketersediaan data di Elasticsearch, dan visualisasi log di Kibana. Pada gambar 4.1 terlihat pod dari fluentd yang berhasil dideploy pada cluster Kubernetes.

```
[root@worker k8s-logging-efk]# kubectl get po -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-77d59654f4-qqlc5   1/1     Running   0          7d17h
calico-node-2fk24                          1/1     Running   0          7d17h
coredns-cb4864fb5-26zjs                    1/1     Running   0          7d17h
coredns-cb4864fb5-ts45j                    1/1     Running   0          7d17h
etcd-worker                                1/1     Running   0          7d17h
fluentd-vcq41                              1/1     Running   0          7d17h
kube-apiserver-worker                      1/1     Running   0          7d17h
kube-controller-manager-worker             1/1     Running   0          7d17h
kube-proxy-ks569                           1/1     Running   0          7d17h
kube-scheduler-worker                      1/1     Running   0          7d17h
```

Setelah memastikan fluentd berjalan pada cluster Kubernetes, selanjutnya memastikan Elasticsearch dan Kibana juga berjalan pada cluster Kubernetes. Pada gambar tersebut dapat dilihat untuk Elasticsearch dan Kibana berjalan dengan normal pada 31 cluster.

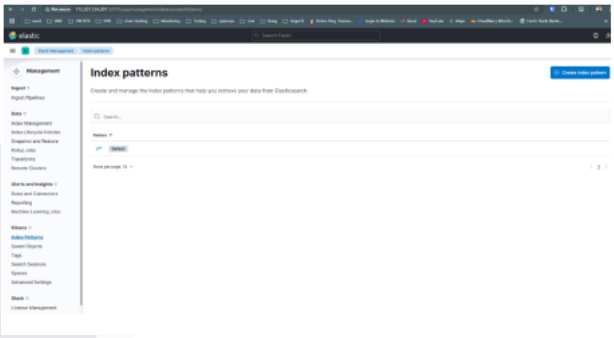
```
NAME                                READY   STATUS    RESTARTS   AGE
pod/elasticsearch-master-0          1/1     Running   0          7d18h
pod/kibana-kibana-7757997f5f-64pbq  1/1     Running   0          7d18h

NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/elasticsearch-master        ClusterIP   10.109.164.98   <none>        9200/TCP,9300/TCP   7d18h
service/elasticsearch-master-headless   ClusterIP   None        <none>        9200/TCP,9300/TCP   7d18h
service/kibana-kibana               NodePort    10.104.98.154   <none>        5601:32076/TCP      7d18h

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/kibana-kibana       1/1     1            1           7d18h

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/kibana-kibana-7757997f5f  1         1         1       7d18h
```

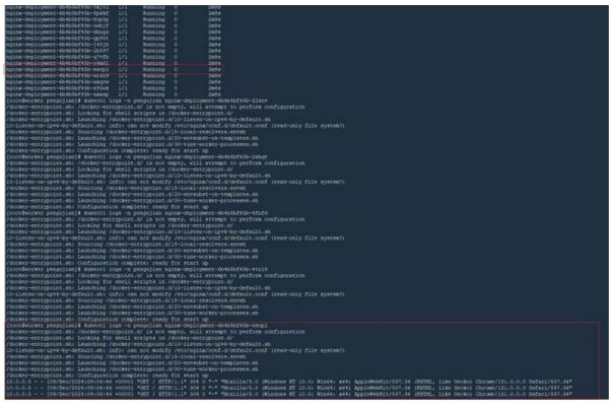
Setelah semua service berjalan dengan normal mulai dari Fluentd, Elasticsearch, dan Kibana. Selajutnya dilakukan akses dashboard Kibana untuk melihat log hasil dari fluentd. Untuk melakukan akses pada Kibana dapat menggunakan port yang sebelumnya dibuka dengan type NodePort yaitu http://IP_cluster:32076 maka akan muncul halaman dashboard dari Kibana seperti pada gambar 4.3 yang menampilkan halaman dari Kibana.



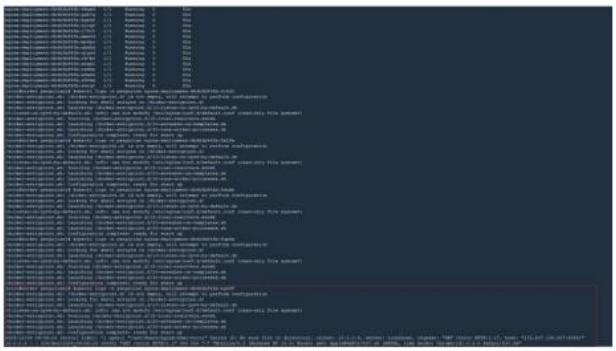
B. Hasil Skenario Pengujian Melakukan Pencarian Log Manual

Pengujian ini dilakukan dengan mengambil log langsung dari pod di Kubernetes menggunakan perintah kubectl logs. Data yang diperoleh mencakup informasi penting, seperti timestamp, level log (INFO, ERROR, DEBUG), dan pesan log. Pada gambar 4.4 menunjukan access log dari pod yang menjalankan web sederhana, untuk jumlah pod yang dideploy adalah 1 buah pod. Pada gambar 4.5 menunjukan error log dari pod dengan jumlah pod adalah 1.

Dalam melakukan pengambilan log pada aplikasi dimana hanya ada 1 buah pod masih bisa dilakukan dengan cepat dan sesuai dengan waktu aplikasi diakses. Selanjutnya adalah dengan meningkatkan jumlah replica pod menjadi 20 buah dan melakukan pengambilan log secara manual dengan cara melakukan eksekusi command kubectl logs nama-pod sebanyak 20 kali, dikarenakan jumlah pod yang tersedia adalah 20 replica. Dari total 20 replica yang tersedia pada Kubernetes dengan melakukan pengambilan loga secara manual untuk mendapatkan akses log yang sesuai diperlukan 16 kali eksekusi command kubectl logs nama-pod pada kluster Kubernetes. Pada gambar 4.6 menunjukan dari total 20 replica untuk mendaptkan akses log maka perlu melakukan cek satusatu pada semua replica sampai akses log berhasil ditemukan pada pod ke 16.

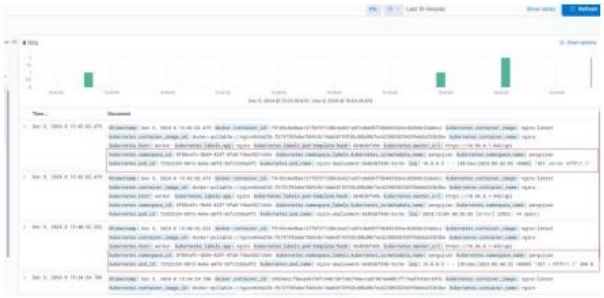


Selanjutnya melakukan hal yang sama pada saat melakukan pengambilan error logs pada aplikasi dengan jumlah replica sebanyak 20. Dari total 20 replica untuk mendapatkan error log dari aplikasi diperoleh pada pod kedelapan dimana pada saat user melakukan akses ke aplikasi maka yang melayani request dari user adalah salah satu pod dari total 20 replica pod yang tersedia. Pemilihan ini dilakukan secara acak maka dari itu untuk melakukan pengambilan log secara manual kita harus melakukan pengecekan pada semua pod yang tersedia. Seperti pada pengambilan error log pada gambar 4.7 didapatkan hasil error log berada pada pod kedelapan.

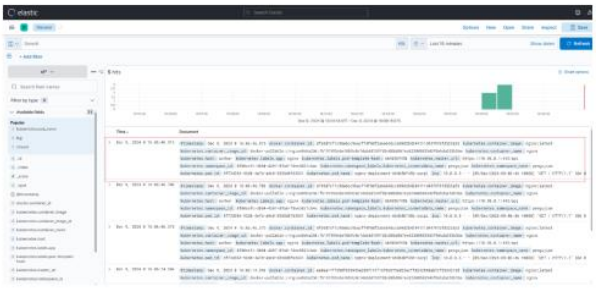


C. Hasil Skenario Pengujian Melakukan Pengambilan Log dari Dashboard Kibana

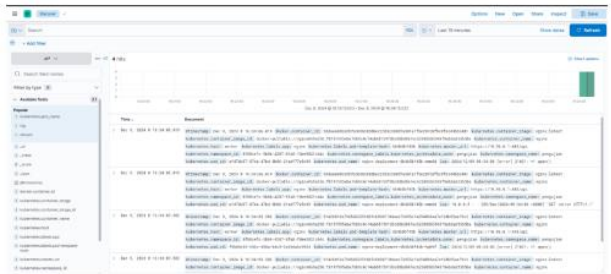
Pada pengujian pertama dilakukan dengan membuat 1 replica untuk pod. Terlihat pada gambar akses log dapat terlihat dan sesuai dengan waktu akses aplikasi.



Dengan menggunakan fluentd untuk mengumpulkan log tidak perlu melakukan eksekusi command secara manual pada pod. Fluentd akan secara otomatis mengirimkan log dari pod ke elasticsearch dan ditampilkan pada dashboard Kibana. Pada dashboard Kibana juga kita bisa melihat jenis dari log yang ditmpilkan seperti akses log dan juga error log yang dibedakan dengan tipe dari log yaitu stdout untuk standar output akses log dan stderr untuk standar error log. Selanjutnya adalah melakukan pengujian pengambilan log dengan meningkatkan jumlah replica menjadi 20. Dengan meningkatkan jumlah replica pada aplikasi tidak akan berpengaruh pada proses pengambilan loga dengan fluentd dikarenakan fluentd akan secara otomatis mengirimkan log ke elasticsearch dan ditampilkan pada dashboard Kibana. Pada gambar 4.9 merupakan tampilan dari pengambilan log dengan jumlah replica 20



Kemudian melakukan pengambilan error log pada jumlah replica 20 dapat dilihat pada gambar 4.10 untuk error log juga sama seperti akses log dimana tidak perlu melakukan pengecekan pada masing-masing pod melainkan fluentd secara otomatis mengirimkan log ke elasticsearch dan di tampilkan pada dashboard kibana.



D. ANALISIS DAN PEMBAHASAN

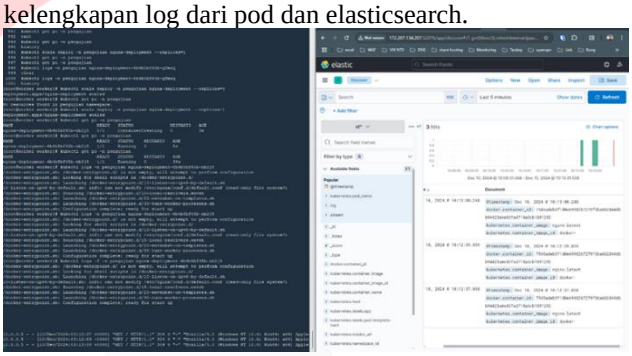
A. Pengujian Efektivitas Pengumpulan Log dengan Fluentd

Hasil Pengujian efektivitas pengumpulan log dengan Fluentd dilakukan untuk mengukur kinerja dan keandalan Fluentd dalam mengelola log dari cluster Kubernetes. Dalam pengujian ini, hasil yang diperoleh berdasarkan berbagai parameter yang telah ditentukan sebelumnya, yaitu latensi

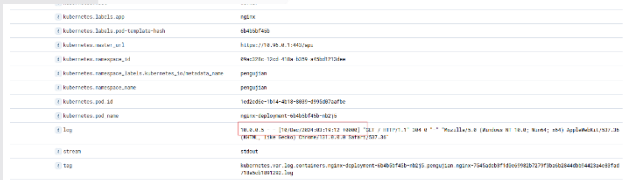
pengiriman log, akurasi waktu pengumpulan log, kelengkapan log, penggunaan sumber daya.

No	Pengujian	Deskripsi	Metode Pengujian	Hasil Pengujian	Ket
1	Kelengkapan log	Memastikan semua log yang dihasilkan *pod* dikumpulkan oleh Fluentd.	Membandingkan log manual dengan log di Elasticsearch.	Lengkap	Sesuai
2	Akurasi Waktu	Memastikan *timestamp* log sesuai antara log pada pod dan log di Elasticsearch.	Membandingkan *timestamp* log manual dari pod dan log yang dikirimkan fluentd.	*Timestamp* sama	Sesuai
3	Latensi Pengumpulan Log	Waktu yang dibutuhkan untuk log muncul di Elasticsearch setelah dibuat.	Mengukur selisih waktu pembuatan dan penerimaan log.	Kurang dari 1 detik	Sesuai
4	Volume Log	Memastikan Fluentd dapat menangani log dengan volume besar.	Menghasilkan log besar secara *stress testing*.	Jumlah logs 1000 per menit	Sesuai
5	Keberhasilan Visualisasi di Kibana	Memastikan log dapat divisualisasikan dengan baik di Kibana.	Memeriksa log di Kibana setelah diterima Elasticsearch.	Data visualisasi lengkap.	Sesuai

Berdasarkan tabel didapatkan bahwa pada pengujian efektivitas dari sisi kelengkapan log sudah sesuai dengan log yang berada pada pod sama dengan log yang diterima oleh elasticsearch. Pada gambar 5.1 adalah hasil dari perbadingan kelengkapan log dari pod dan elasticsearch.

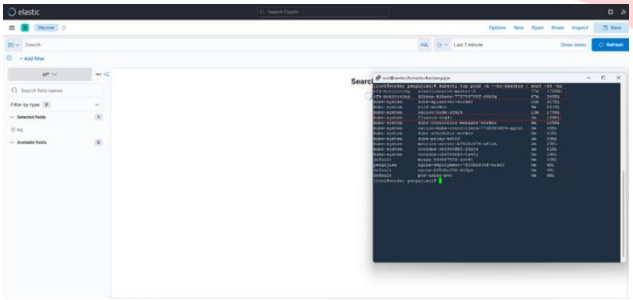


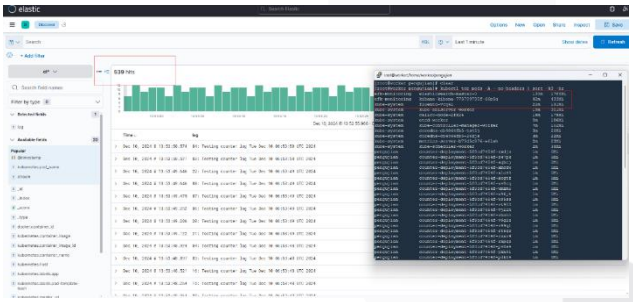
Kemudian pada pengujian akurasi waktu dilakukan pengujian berdasarkan pada kesesuaian *timestamp* pada log di pod dan log yang diterima oleh elasticsearch. Pada pengujian ini dapat dilihat pada gambar 5.2 yang menunjukan *timestamp* yang sesuai antara pod dan elasticsearch. Dilakukan juga pengujian untuk latensi pengiriman log dari fluentd yang mana latensi untuk pengiriman adalah kurang dari 1 detik.

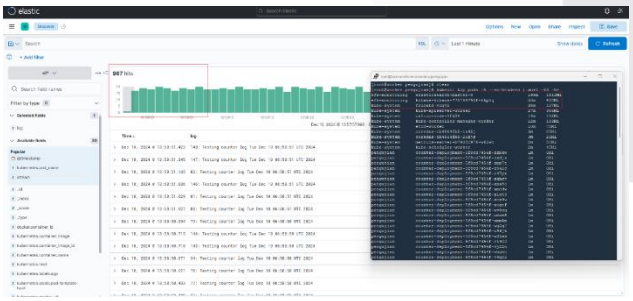


Pada pengujian volume log dengan menggunakan metode *stress test* dimana pengujian dilakukan dengan membuat 100 pod yang akan melakukan generate log secara terus menerus untuk dikirimkan oleh fluentd ke elasticsearch. Untuk perbandingan volume log yang dapat di*handle* oleh fluentd disini menggunakan perbadingan sebelum dilakukan *stress test* didapatkan hasil load atau resource yang digunakan oleh

fluentd adalah sebesar 2 mili core CPU dan 134 MB memory pada elasticsearch menggunakan CPU 77 mili core dengan memory 1795 MB, sedangkan untuk kibana menggunakan CPU sebesar 67 mili core dengan memory 348 MB terlihat pada gambar 5.4. Setelah dilakukan penambahan jumlah pod menjadi 50 seperti pada gambar 5.5 didapatkan load pada CPU yang digunakan oleh fluentd adalah sebesar 22 mili core dan memory 131 MB. Elasticsearch menggunakan CPU sebesar 130 mili core dengan memory 1788 MB dan kibana menggunakan CPU sebesar 42 mili core dengan memory 433 MB. Terlihat bahwa dengan melakukan penambahan jumlah pod yang digunakan untuk *generate* log penggunaan dari sisi resource juga meningkat. Kemudian untuk meningkatkan volume log pada cluster jumlah pod ditambahkan menjadi 100 pods dan diperoleh hasil seperti pada gambar 5.6 resource CPU yang digunakan oleh fluentd adalah 38 mili core dengan memory 127 MB, elasticsearch menggunakan CPU 196 mili core dengan memory 1512 MB dan kibana menggunakan 53 mili core dengan memory 428 MB. Terjadi penaikan pada resource yang digunakan namun tidak memperngaruhi fluentd untuk melakukan pengiriman log ke elasticsearch.

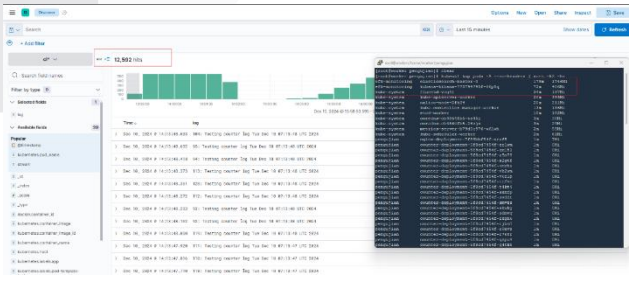




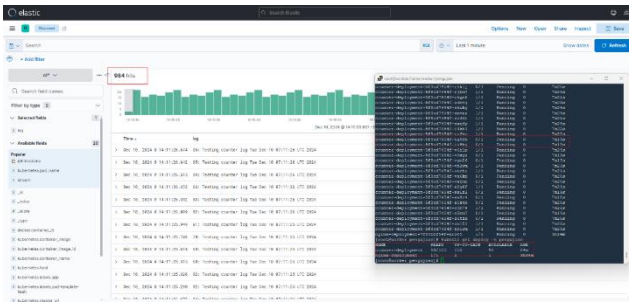


Pada pengujian juga didapatkan hasil dimana fluentd bisa menangani log sebanyak 12.592 dalam waktu 15 menit ditunjukan pada gambar 5.7 dan resource yang digunakan masih relatif aman. Dalam pengujian yang sudah dilakukan maksimal pods yang dapat dibuat adalah 100 dikarenakan node pada cluster hanya bisa maksimal untuk menangani 100 pods secara bersamaan terlihat pada gambar 5.8 dimana pod tidak dapat dibuat lebih dari 100.

No	Jumlah Pod	Fluentd Resource Usage	Elasticsearch Resource Usage	Kibana Resource Usage	Volume Log yang Ditangani	Catatan
1	1 Pod	CPU: 2 mCore Memory: 134 MB	CPU: 77 mCore Memory: 1795 MB	CPU: 67 mCore Memory: 348 MB	160 log dalam 15 menit	Pengujian awal tanpa stress test.
2	50 Pods	CPU: 22 mCore Memory: 131 MB	CPU: 130 mCore Memory: 1788 MB	CPU: 42 mCore Memory: 433 MB	8085 log dalam 15 menit	Penambahan jumlah pod menyebabkan peningkatan resource secara signifikan.
3	100 Pods	CPU: 38 mCore Memory: 127 MB	CPU: 196 mCore Memory: 1512 MB	CPU: 53 mCore Memory: 428 MB	12.592 log dalam 15 menit	Fluentd tetap stabil menangani log dengan peningkatan volume tanpa mengalami bottleneck.
4	>100 Pods	-	-	-	-	-



Pada Gambar 5.8 menunjukan bahwa kluster Kubernetes tidak bisa melakukan create pod lebih dari 100 hal ini dikarenakan dari spesifikasi server yang tidak bisa melayani pod lebih dari 100 buah pod.



B. Hasil Analisis Latensi Pengiriman Log

Pada pengujian latensi yang dilakukan pada sistem log menggunakan fluentd di kluster Kubernetes dilakukan dengan cara melakukan pengukuran waktu akses dari user dan waktu log diterima oleh elasticseacrh. Untuk mendapatkan nilai latensi dilakukan perhitungan selisih antara waktu log diterima oleh elasticsearch dengan waktu log saat user akses. Tabel 5.3 merupakan tabel dari latensi pengiriman log pada fluentd.

No Pengujian	Latensi (ms)	Keterangan
Pengujian 1	221 ms	Log terkirim
Pengujian 2	330 ms	Log terkirim
Pengujian 3	291 ms	Log terkirim
Pengujian 4	228 ms	Log terkirim
Pengujian 5	232 ms	Log terkirim
Pengujian 6	221 ms	Log terkirim
Pengujian 7	207 ms	Log terkirim
Pengujian 8	219 ms	Log terkirim
Pengujian 9	220 ms	Log terkirim
Pengujian 10	226 ms	Log terkirim
Pengujian 11	234 ms	Log terkirim
Pengujian 12	209 ms	Log terkirim
Pengujian 13	215 ms	Log terkirim
Pengujian 14	215 ms	Log terkirim
Pengujian 15	221 ms	Log terkirim
Pengujian 16	206 ms	Log terkirim
Pengujian 17	208 ms	Log terkirim
Pengujian 18	213 ms	Log terkirim
Pengujian 19	201 ms	Log terkirim
Pengujian 20	191 ms	Log terkirim
Rata-rata latensi	225,4 ms	

Berdasarkan tabel 5.3, nilai latensi pengiriman log bervariasi dengan rentang antara 191 ms hingga 330 ms. Dari hasil pengujian ini, ditemukan bahwa latensi tertinggi terjadi pada pengujian ke-2 dengan waktu 330 ms, sedangkan latensi terendah terjadi pada pengujian ke-20 dengan waktu 191 ms. Secara keseluruhan, rata-rata latensi pengiriman log dalam sistem ini adalah 225,4 ms. Dari hasil analisis ini, dapat disimpulkan bahwa sistem Fluentd yang digunakan dalam kluster Kubernetes memiliki performa yang cukup baik dalam mengirimkan log ke Elasticsearch dengan latensi yang relatif stabil. Meskipun terdapat beberapa fluktuasi dalam hasil pengukuran, mayoritas pengiriman log terjadi dalam rentang 200-230 ms, yang menunjukkan bahwa sistem mampu menangani log dengan cepat dan efisien. Dengan hasil pengujian ini, sistem Fluentd pada Kubernetes dapat dikatakan cukup responsif dalam menangani log secara real-time. Namun, untuk meningkatkan performa lebih lanjut, beberapa optimasi dapat dilakukan, seperti tuning konfigurasi buffer Fluentd, menyesuaikan parameter flush interval, serta memastikan infrastruktur jaringan memiliki bandwidth yang cukup untuk menangani lalu lintas log dalam jumlah besar.

VI. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan dari hasil implementasi dan pengujian yang telah dilakukan terkait manajemen log di cluster kubernetes menggunakan fluentd dan elasticsearch maka didapatkan kesimpulan sebagai berikut :

1. Implementasi Fluentd sebagai agen pengumpulan log pada cluster Kubernetes berhasil dilakukan, dengan konfigurasi khusus pada namespace efk-monitoring. Fluentd mampu membaca, memformat, dan mengirimkan log dari cluster kubernetes ke Elasticsearch, yang kemudian dapat diakses melalui dashboard visualisasi Kibana.
2. Pada pengujian yang telah dilakukan didapatkan bahwa dengan menggunakan fluentd memberikan peluang bagi system administrator dalam mengelola infrastruktur kubernetes dengan lebih mudah. Integrasi Fluentd dengan Elasticsearch dan visualisasi log menggunakan Kibana berjalan dengan baik. Pengguna dapat dengan mudah mencari, memfilter, dan menganalisis log menggunakan Kibana, sehingga meningkatkan efisiensi troubleshooting dan monitoring sistem. Dengan menggunakan fluentd dalam melakukan pencarian error dan akses log terhadap pod dengan jumlah replica lebih dari 10 didapatkan bahwa fluentd memberikan hasil dengan lebih cepat dimana dari jumlah 10 pod tidak perlu melakukan eksekusi command kubectl logs pada pod yang ada.
3. Berdasarkan pengujian yang telah dilakukan terhadap efektivitas pengumpulan log dengan fluentd. Terdapat 5 kategori pengujian yang dilakukan yaitu kelengkapan log, akurasi waktu, latensi pengumpulan log, volume log, dan keberhasilan visualisasi log dengan kibana. Pada pengujian kelengkapan log didapatkan hasil dimana log yang ditampilkan pada pod sesuai dengan log yang ditampilkan pada elasticsearch dan tidak ada log yang hilang. Dari sisi akurasi waktu memastikan *timestamp* log pada pod sesuai dengan yang ditampilkan di kibana. Lantensi penerimaan log oleh elasticsearch juga tidak lebih dari 1 sekon dan dapat menampilkan log secara

real time. Kemudian untuk pengujian volume log yang besar menggunakan *stress-test* dengan menggunakan jumlah pod mulai dari 20 kemudian ditambahkan menjadi 50 pod dan maksimal menggunakan 100 pod untuk melakukan *stress test.* Pada saat melakukan *stress test* flluentd masih bisa menangani log dengan baik dan mengirimkan pada elasticsearch yang kemudian ditampilkan pada kibana. Terjadi peningkatan pada resource CPU namun tidak terlalu signifikan yakni untuk menangani 12,592 logs dalam waktu 15 menit fluentd hanya menggunakan 38 mili core CPU dan 127 megabytes memory untuk menangani logs tersebut. Kemudian semua log yang berhasil diterima oleh elasticsearch ditampilkan pada dashboard kibana.

B. Saran

Saran yang dapat diterapkan pada pengembangan penelitian selanjutnya yang memiliki topik seperti pada skripsi ini sebagai berikut :

1. Untuk pengembangan lebih lanjut dalam monitoring log pada kubernetes menggunakan fluentd dari sisi elasticsearch dan kibana bisa dilakukan deploy atau installasi diluar cluster kubernetes sehingga dalam pengembangan monitoring log tidak hanya terbatas pada cluster kubernetes.
2. Untuk pengembangan selanjutnya juga diharapkan dapat mengintegrasikan sistem notifikasi berbasis event (seperti alert) menggunakan Slack, Telegram, atau email untuk mempercepat respons terhadap permasalahan yang terdeteksi di log.
3. Dalam penelitian selanjutnya diharapkan dapat menggunakan cluster yang lebih besar dari sisi spesifikasi server sehingga dalam pengujian *stress-test* bisa lebih besar untuk menguji skalabilitas sistem.

REFERENSI

[1] Bayu Agung Prakoso, Unan Yusmaniar Oktiawati, "Analisis Perbandingan Kinerja Container Network Interface Flannel dan Cilium sebagai Interface Utama pada Multus CNI dalam Jaringan Klaster Kubernetes," *Journal of Internet and Software Engineering (JISE),* vol. 5, p. 99, 2024.

[2] C. Melendez, " Getting Started with Kubernetes," 2022.

[3] B. S. Ach Izalul Haq, "Analisis Perbandingan Performa Metode ELK Stack dan Grafana Loki Pada Honeypot Server," *Jumal SISFOKOM (Sistem Informasi dan Komputer),* vol. 10, pp. 376-385, 2020.

[4] Vlad-Andrei Zamfir, Mihai Carabas, Costin Carabas, Nicolae Tapus, "Systems monitoring and big data analysis using the Elasticsearch system," *Intemational Conference on Control Systems and Computer Science,* p. 188, 2019.

[5] M. Nerdi Rafli, Emil Nafan, Eka Praja Wiyata Mandala, "Analisis dan Peningkatan Performa Log File Pada Server dengan Elk Stack," *Jurnal Sarjana Teknik Informatika,* vol. 12, Januari 2024.

[6] Salma Rachman Dira, Muhammad Arif Fadhlv Ridha, "Monitoring Kubernetes Cluster Menggunakan Prometheus dan Grafana," *Proceeding Applied Business and Engineering Conference,* p. 350, 2022.

[7] Bayu, Putu Napoleon Krishna, "Implementasi Server Log Monitoring System menggunakan Elastic Stack," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* vol. 6, p. 1822, April 2022.

[8] Oktiawati, Guntoro Yudh Kusumal. Unan Yusmaniar, "Perancangan Sistem Monitoring Performa Aplikasi Menggunakan Opentelemetry dan Grafana Stack," *Journal of Internet and Software Engineering (JISE),* vol. 3, November 2022.

[9] Putra, Yuri Chandra Tri, "Implementasi Arsitektur Microservice Pada Aplikasi Web Pengajaran Agama Islam Home Pesantren," 2020.

[10] T. R. Maitimu, "Perancangan dan Implementasi WebServer Clustering dengan Skema Load Balance Menggunakan Linux Virtual Server Via NAT," *Jurnal Teknologi Informasi-Aiti,* vol. 5, 2008.

[11] T. K. Authors, "Why you need Kubernetes and what it can do," 2024.

[12] A. Fauzi, "Sistem Manajemen Dan Visualisasi Syslog Perangkat Jaringan Komputer Pada Ict Universitas Diponegoro Berbasis Elk Stack," *Jurnal Sistem Komputer,* vol. 2, 2020.

[13] M. Arslan, "Pengenalan Singkat Elasticsearch".

[14] Eddy Tungadi, Meylanie Olivya, Suwesti Akbar, "Analisis Kinerja Elasticsearch Pada Proses Query Data," *Prosiding Seminar Nasional Komunikasi dan Informatika,* pp. 37-41, 2019.

[15] A. Gupta, "Common Architecture Patterns with Fluentd and Fluent Bit," 3 Desember 2020.

[16] Elastic, "What is Elasticsearch ?," 2019.

[17] Elang Putra Sartika, Andhik Budi Cahvono, "Implementasi Elasticsearch Logstash Kibana Stack pada Sistem Portal Pengembangan dan Pembinaan Sumber Daya Manusia," *Tugas akhir Fakultas Teknologi Industri Universitas Islam Indonesia Yogyakarta.*

[18] D. P. Sampurna Dadi Riskiono, "Analisis Metode Load Balancing Dalam Meningkatkan Kinerja Website E-Learning," *Jumai Teknoinfo,* vol. 14, pp. 22-26, 2020.

[19] Fajar Zuhroni, Adian Fatchur Rochim, Eko Didik Widianto, "Analisis Performansi Layanan Kluster Server Menggunakan Penyeimbang Beban Dan Virtualbox," *Jurnal Teknologi dan Sistem Komputer,* vol. 3, 2015.

[20] A. Ilmi Barokah, "Analisis Perbandingan Serverless Computing Pada Google Cloud Platform," *Jurnal Teknlogi Informatika dan Komputer MH. Thamrin,* vol. 7, September 2021.

[21] H. A. Toga Aldila Cinderatama, "Pemanfaatan Docker Swarm Sebagai Kolaborator Private Dan Public Cloud Untuk Implementasi Scalable Virtualisasi," *Seminar Nasional Inovasi Teknologi UN PGRI Kediri*, 2017.

[22] Y. S. Stefanus Eko Prasetyo, "Analisis Perbandingan Performa Web Server Docker Swann dengan Kubernetes Cluster," *Conference on Management, Business, Innovation, Education and Social Science*, vol. I, pp. 825-830, 2021.

[23] Vinandita Ayu Kinanti, Muhammad Iqbal, Candra Mahendra Putra, "Perancangan Infrastruktur Kubernetes Untuk Aplikasi Data Center Infrastructure Management (Dcim) Studi Kasus Pt. Pelayaran Nasional Indonesia (Pelni)," *e-proceeding Of Applied Science*, vol. 10, p. 948, 4 Agustus 2024.

[24] Harshali Bobde, Avantika Aglawe, Shruti Lakhamapure, Dhanashri Ukey, Prof. Komal Dhakate, "Log Alert System Server Log Recognition and Alert System," *International Journal of Trend in Scientific Research and Development (IJTSRD)*, vol. 8, no. 6, 2024.

[25] Diki Taufik Gurohman, Bekti Maryuni Susanto, Agus Hariyanto, Ery Setiyawan Jullev, Atmadji, Mukhamad Angga Gumilang, Ely Antika, Nanik Anita Mukhlisoh, "Penerapan Horizontal Pod Autoscaler dan Redis Cluster Berbasis Kubernetes untuk Meningkatkan Performa Website Elearning," *SKANIKA: Sistem Komputer dan Teknik Informatika*, vol. 7, pp. 224-235, Juli 2024.