SKRIPSI

PENGEMBANGAN DASBOR SISTEM PENCATATAN LOG SERVER MENGGUNAKAN ELASTICSEARCH-FLUENTD-KIBANA (EFK) STACK

Disusun Oleh:

N a m a : Rio Pradana Aji

NIM : 16523120

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2020

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN DASBOR SISTEM PENCATATAN LOG SERVER MENGGUNAKAN ELASTICSEARCH-FLUENTD-KIBANA (EFK) STACK

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Rio Pradana Aji

NIM : 16523120

Yogyakarta, 12 November 2020

Pembimbing,

Andhik Budi Cahyono, S.T., M.T.

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN DASBOR SISTEM PENCATATAN LOG SERVER MENGGUNAKAN ELASTICSEARCH-FLUENTD-KIBANA (EFK) STACK

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 12 November 2020

Tim Penguji

Ketua Penguji

Andhik Budi Cahyono, S.T., M.T. _____

Anggota 1

Ari Sujarwo, S.Kom., MIT. _____

Anggota 2

Erika Ramadhani, S.T., M.Eng. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Rio Pradana Aji

NIM : 16523120

Tugas akhir dengan judul:

PENGEMBANGAN DASBOR SISTEM PENCATATAN LOG SERVER MENGGUNAKAN ELASTICSEARCH-FLUENTD-KIBANA (EFK) STACK

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 12 November 2020

Rio Pradana Aji

HALAMAN PERSEMBAHAN

Alhamdulillah berkat rahmat dan kasih sayang Allah Subhanahu Wa Ta'ala saya dapat sampai di titik ini dan mampu menyelesaikan Tugas Akhir ini.

Tugas akhir ini saya persembahkan kepada kedua orang tua saya yang banyak berkorban dan menyayangi saya tanpa pamrih sedikitpun dari di hari di mana saya dilahirkan hingga saat ini. Kedua orang tersebut adalah Bapak Dinharjaya dan Ibu Erna Listiyani, terima kasih telah menyayangi saya dan terima kasih atas segala doa dan ridho yang telah kalian berikan kepada saya. Adik saya tercinta Dyah Ayu Pramesti Regita Putri dan kepada semua kucing-kucing saya yang tidak bisa saya sebutkan satu persatu baik yang masih hidup maupun yang sudah meninggal. Terima kasih kepada kalian semua saya bisa sampai pada titik ini.

HALAMAN MOTO

"Jahatnya Kemalasan" – Rio Pradana Aji

"Relax, you will graduate. You will get a job. You will find love. You have an entire life. Things take time. Just enjoy where you are now." – Anonymous

"…Maka sesungguhnya beserta kesulitan ada kemudahan, sesungguhnya beserta kesulitan itu ada kemudahan…." Al-Qur'an Surah Al-Insyirah ayat 5-6

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Alhamdulillah saya haturkan kepada Allah *Subhanahu wa ta'ala,* yang telah melimpahkan rahmat dan taufiq serta hidayat-Nya, sehingga Tugas Akhir yang berjudul "PENGEMBANGAN DASBOR SISTEM PENCATATAN LOG SERVER MENGGUNAKAN ELASTICSEARCH-FLUENTD-KIBANA (EFK) STACK (STUDI KASUS BADAN SISTEM INFORMASI UIVERSITAS ISLAM INDONESIA)" ini dapat terselesaikan. Shalawat serta salam tak lupa penulis haturkan kepada junjungan kita Nabi Muhammad *shalallahu 'alaihi wa salam.*

Penulisan tugas akhir ini bertujuan untuk memenuhi persyaratan mendapatkan gelar Sarjana di Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Adapun penjaluran yang penulis ambil ialah jalur Magang sehingga tugas akhir ini penulis kerjakan sembari melaksanakan magang di Badan Sistem Informasi yang bertempat di gedung rektorat UII.

Selama penulisan Tugas Akhir ini, penulis banyak menerima dukungan dan masukan sehingga dapat menyelesaikan laporan ini, oleh karena itu, penulis mengucapkan terima kasih setulusnya kepada:

1. Allah *Subhanahu wa ta'ala* karena tanpa izin-Nya penulis tidak akan mampu menyelesaikan laporan ini.
2. Bapak Dinharjaya dan Ibu Erna Listiyani selaku kedua orang tua serta Adik saya Dyah Ayu Pramesti Regita Putri yang selama ini selalu mendoakan dan memberi dukungan moral materi.
3. Bapak Andhik Budi Cahyono, S.T, M.T. selaku dosen pembimbing yang banyak membantu dan meluangkan waktu untuk membimbing, memeriksa, dan memberikan saran dalam penyusunan laporan ini.
4. Ibu Aridhanyati Arifin, S.T, M.Cs. selaku dosen pendamping akademik yang banyak membantu penulis melewati masa-masa perkuliahan. Terima kasih atas bimbingannya selama ini bu.
5. Mas Nashihun Amien, Mas Krismanto, Mas Ardhi Pratama, Mas Ikhwan Alfath, dan Feby Permatasari sebagai pembimbing lapangan dan rekan kerja yang telah bersedia meluangkan waktu untuk membimbing, mengarahkan, dan memberikan banyak masukan selama proses magang.
6. Teman teman kontrakan Bapak Muslimin yaitu Raden Mifthakhurozak Budi Nugraha, Nassharih Abdulloh, Aditya Putra Irawan, Aditya Mahavira, Firza Ikhwanda Halim,

Alexander Ramadhan Suratinoyo, Muhammad Yasin, Yoga Kosasih. Terima kasih kawan, tanpa kalian kuliah saya pasti sangat sepi.

7. Seluruh pihak yang tidak bisa saya sebutkan satu persatu. Terima kasih atas doa kalian yang telah kalian panjatkan untuk saya. Semoga kalian mendapatkan berkah, rezeki, dan mimpi yang ingin kalian capai.

Akhir kata penulis menyadari bahwa laporan ini masih jauh dari kata sempurna oleh karena itu kritik dan saran yang bersifat membangun akan penulis terima dengan senang hati dan semoga laporan ini bermanfaat bagi semua pihak yang memerlukan.

Wassalamualaikum Warahmatullahi Wabarakatuh

Yogyakarta, 12 November 2020

Rio Pradana Aji

SARI

Badan Sistem Informasi (BSI) yang ada di Universitas Islam Indonesia (UII) adalah sebuah badan yang bertugas untuk menyediakan layanan sistem informasi dan juga internet di lingkup kampus UII. Bertambahnya kebutuhan civitas academica UII akan layanan aplikasi yang disediakan oleh BSI menyebabkan semakin bertambahnya jumlah *server* aplikasi yang harus dikelola oleh BSI. *Server* tersebut berisi *log* yang digunakan oleh *SysAdmin* untuk memantau kondisi *server-server* yang dikelola BSI. Namun, banyaknya *log* di setiap *server* tersebut menyita waktu *SysAdmin* untuk mencari *log* yang berisikan *error* terlebih bila harus dilakukan pengecekan satu-persatu. Solusi dari masalah tersebut adalah dibuatnya sebuah *Centralized Log Management (CLM)* sebagai tempat penyimpanan semua *log* yang ada agar bisa di-*monitoring* oleh *SysAdmin*. CLM ini menggunakan teknologi Elasticsearch, Fluentd, Kibana (EFK) *Stack* sebagai penyimpan *log, aggregator*, visualisasi *log*. Selain EFK, CLM juga menggunakan Fluentbit sebagai *log collector*. Data *log* yang sudah diolah dengan EFK *Stack* dan Fluentbit tersebut selanjutnya akan divisualisasikan dalam dasbor sistem pencatalan *log* untuk memudahkan *SysAdmin* dalam membaca dan mencari data *log*. Hasil akhir dari penggunaan teknologi EFK *Stack* dan Fluentbit dalam pengembangan dasbor sistem pencatatan *log* adalah *SysAdmin* dapat menemukan *log* di *server* dengan cepat dan tepat. Serta mengurangi waktu perbaikan jika ditemukan kesalahan karena tidak harus mengecek *server* satu-persatu.

Kata kunci: EFK Stack, Fluentbit, Centralized Log Management.

GLOSARIUM

Log	file berisikan teks sederhana aktifitas yang dilakukan oleh *server* dalam waktu tertentu.
CLM	proses mengumpulkan data *log* dari berbagai sumber dan menggabungkan data yang dikumpulkan untuk dilakukan *monitoring*.
Fluentbit	*tool* untuk mengambil data *log* pada server
Fluentd	*tool* untuk mengirimkan data *log* ke berbagai tujuan penyimpanan data.
Elasticsearch	*storage* atau tempat penyimpanan data *log.*
Kibana	visualisasi untuk menampilkan data *log* secara terstruktur.
Node	mesin pekerja yang ada di kubernetes, bisa berupa fisik maupun virtual.
Pod	objek paling kecil di kubernetes yang biasanya berisikan satu atau lebih kontainer.
Kubernetes	sistem orkestra kontainer yang bertujuan untuk mengelola kontainer pada cluster server.
Docker	sebuah produk layanan virtualisasi yang menjalankan aplikasi pada secara terisolasi(kontainer) satu dengan yang lainnya.
Aggregator	perangkat untuk mengumpulkan data kemudian melakukan *filter* pada data tersebut.
Downstream	proses data menuju tujuan.
ConfigMap	objek API yang digunakan untuk menyimpan data *deployment* pada kubernetes.

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latarbelakang

Badan Sistem Informasi (BSI) adalah sebuah badan yang berfungsi sebagai penyedia, perancang, dan pengembang layanan sistem informasi yang ada di lingkungan Universitas Islam Indonesia (UII). Selain menangani pengembangan layanan sistem informasi BSI juga menangani jaringan internet yang digunakan di UII. Pengguna layanan BSI sendiri tidak hanya mahasiswa aktif saja namun dosen, karyawan, tenaga pendidikan, dan pemangku lainnya yang berada di lingkungan UII.

Layanan sistem informasi di BSI yang diberikan kepada warga UII saat ini telah berkembang dengan pesat. Sudah banyak layanan yang telah diberikan maupun yang sedang dikembangkan oleh pihak BSI. Namun pihak BSI masih kekurangan di sisi keamanan dan pelacakan *error* yang ada di *server*. Pelacakan *error* masih menggunakan cara lama yaitu mengecek *server* satu persatu. Sehingga mereka membutuhkan sebuah dasbor sistem pencatatan *log* atau *centralized log management* untuk melihat *log* server yang ada. Dalam membangun sebuah *centralized log management* diperlukan dasbor sistem pencatatan *log* yang mumpuni untuk mengelola banyaknya data *log* tersebut.

Dasbor sistem pencatatan *log* ini menggunakan teknologi Elasticsearch Fluentd Kibana (EFK) *stack* dan Fluentbit. Fluentbit yang merupakan versi ringan Fluentd berfungsi sebagai log *collector* yang akan di-*install* di *virtual machine server* pada layanan BSI atau *virtual machine server* yang ingin dimonitor *behavior*-nya. Fluentd sebagai log *aggregator* untuk melakukan *filtering* dan meneruskan data *log* dari berbagai sumber *log collector* yang ada di *server* BSI. Penggunaan kedua teknologi tersebut dapat dipelajari dengan panduan yang diberikan oleh perusahaan pengembang Fluentd. Selanjutnya data yang sudah masuk ke *aggregator* tersebut disimpan kembali *di server* yang memiliki sistem Elasticsearh dan Kibana. Elasticsearch digunakan untuk menampung semua data yang diperoleh dari *aggregator* sedangkan Kibana bertugas untuk memvisualisasikan data tersebut sehingga data tersebut bisa dibaca dengan normal. Tujuan akhir dari pengembangan dasbor sistem pencatatan *log* adalah mempermudah dan mempercepat kinerja *SysAdmin* dalam menemukan *error* di *server* dengan *centralized log management* tersebut tanpa harus mengecek *server* satu persatu serta menampilkan dasbor sistem yang memberikan informasi lebih detail mengenai server yang di-*monitoring* oleh *SysAdmin.*

Pengembangan dasbor sistem pencatatan *log* ini terhitung dimulai sejak awal Februari 2020 dan selesai serta siap untuk digunakan pada bulan Juni 2020. Dalam pengembangan sistem ini selain telah di-implementasikan pada beberapa *server* di BSI juga telah di-implementasikan di dalam Kubernetes *cluster*. Tujuan implementasi yang dilakukan di dalam Kubernetes selain memonitor *server* juga memonitor *traffic network* dan aplikasi yang berada di dalam Kubernetes *cluster* tersebut. Sehingga data yang dimonitor jauh lebih banyak dan lebih kompleks jika dibandingkan dengan *server* yang berada di luar Kubernetes *cluster*.

1.2 Ruang Lingkup Magang

Ruang lingkup dalam pengembangan dasbor sistem pencatatan *log* ini memiliki beberapa batasan, batasan tersebut dibuat untuk menghindari melencengnya penulisan TA ini. Batasan tersebut adalah sebagai berikut ini:

a. Stack yang digunakan adalah EFK stack.
b. Grafik chart menggunakan pie dan data tabel.
c. Fluentd sebagai *aggregator* dan Fluentbit sebagai *log collector*.
d. Elasticsearch sebagai indeks data dari Fluentd.
e. Dasbor menampilkan data *log syslog* untuk memonitor *up down* server perjam serta *behavior* server, *log ssh attack* untuk memonitor serangan/*attack* yang diterima server, dan *log nginx* untuk memonitor *up down* pada website yang ada di server.

1.3 Tujuan

Tujuan dari dikembangkannya dasbor sistem pencatatan *log* ini adalah sebagai berikut:

a. Dasbor sistem dapat memonitor *log* di setiap *virtual machine server* dengan menggunakan dasbor sistem pencatatan *log* tersebut.
b. Administrator/*Sysadmin* yang ada di tim Site Reliability Engineer (SRE) dapat dengan mudah mencari *error* di *server*.
c. Data log yang rumit dibaca sebelumnya dapat dibaca dengan lebih mudah melalui visualisasi tersebut.
d. Sebagai teknologi baru yang diimplementasikan oleh BSI UII.

1.4 Manfaat

Manfaat dari dikembangkannya dasbor log ini adalah sebagai berikut:

A. Mempermudah mengelola dan memanajemen data log.

B. Mempersingkat waktu dalam menemukan sumber *error* yang ada di *server*.
C. Log dapat terfilter oleh *aggregator* dan di visualisasi sesuai dengan kebutuhan administrator/*Sysadmin*.
D. Mengikuti arus perkembangan teknologi sehingga tidak tertinggal dalam menerapkan/mengimplementasi sebuah teknologi.

1.5 Sistematika Penulisan

Berisi susunan bab dan subbab pada keseluruhan laporan.

A. **BAB 1 PENDAHULUAN**

   Bab ini berisi tentang latar belakang ditulisnya laporan, ruang lingkup magang yang merupakan pekerjaan magang yang diterima kemudian dijadikan bahan TA, tujuan dari pekerjaan tersebut, manfaat yang di dapat dari pekerjaan tersebut, dan sistematika penulisan sebagai struktur inti laporan TA.

B. **BAB 2 DASAR TEORI**

   Bagian ini menjelaskan tentang sumber teori teori yang digunakan dalam pekerjaan yang dilakukan saat magang dan menuliskan laporan TA ini.

C. **BAB 3 PELAKSANAAN MAGANG**

   Bab ini menjelaskan terkait pelaksanaan magang yang telah dilakukan dan pekerjaan apa saja yang telah dilakukan selama magang di BSI UII.

D. **BAB 4 REFLEKSI PELAKSANAAN MAGANG**

   Bagian ini berisi tentang manfaat apa saja yang bisa diambil dari hasil magang di BSI UII selama kurang lebih 6 bulan.

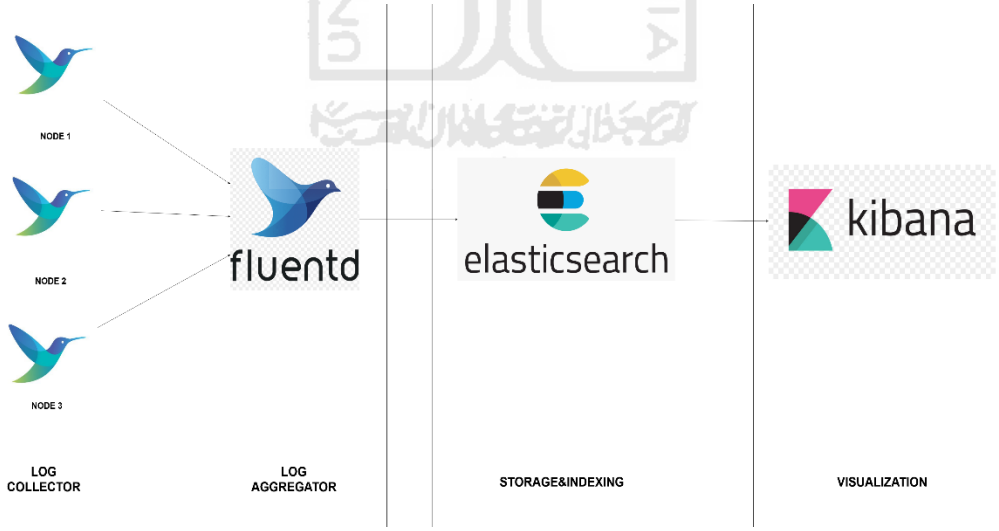
E. **BAB 5 KESIMPULAN DAN SARAN**

   Berisikan saran saran yang ditulis berdasarkan pengalaman yang telah dilalui selama magang di BSI UII.

BAB II
DASAR TEORI

2.1 EFK Stack + Fluentbit

EFK merupakan singkatan dari 3 proyek *open source*: Elasticsearch, Fluentd, dan Kibana. Elasticsearch adalah mesin pencari dan analitik. Fluentd untuk menerima, membersihkan, dan mengurai data log. Kibana memungkinkan pengguna untuk memvisualisasikan data dengan bagan dan grafik di Elasticsearch. Mengapa menggunakan EFK *stack ?*. Ketika Data terus mengalir ke sistem, data tersebut dapat dengan cepat menjadi banyak dan kadaluarsa. Saat data tersebut semakin bertambah banyak, sistem analitik dapat melambat, menghasilkan wawasan yang lambat, kemungkinan akan menjadi masalah bisnis yang serius. Solusi dari masalah tersebut adalah EFK *stack* yang mempermudah dan mempercepat untuk mencari dan menganalisis kumpulan data yang besar (Communications, 2019). EFK stack juga dapat digunakan bersamaan dengan sebuah *tools* bernama Fluentbit sebagai log *collector* di setiap server yang ingin di monitoring. Pada **Gambar 2.1** adalah arsitektur yang umumnya digunakan dalam pengembangan dasbor sistem pencatatan *log*:



Gambar 2.1 Arsitektur EFK stack dan Fluentbit

Pada **Gambar 2.1** bahwa masing-masing *tools* yang ada di EFK stack ini memiliki peran yang berkaitan satu dengan lainnya. Mulai dari Fluenbit berfungsi sebagai kolektor data *log* di *server-server* yang ada. Fluentd sebagai Aggregator yang mem-*parse* dan melakukan *filtering* data

log. Elasticsearch sebagai *database* yang melakukan *indexing* data *log*. Terakhir ada Kibana yang melakukan peran paling krusial yaitu sebagai visualisasi dan pembuatan dasbor sistem dari data *log* yang telah diambil dan diolah. Berikut adalah jabaran lebih lengkap terhadap fungsi dari masing-masing *tools* tersebut:

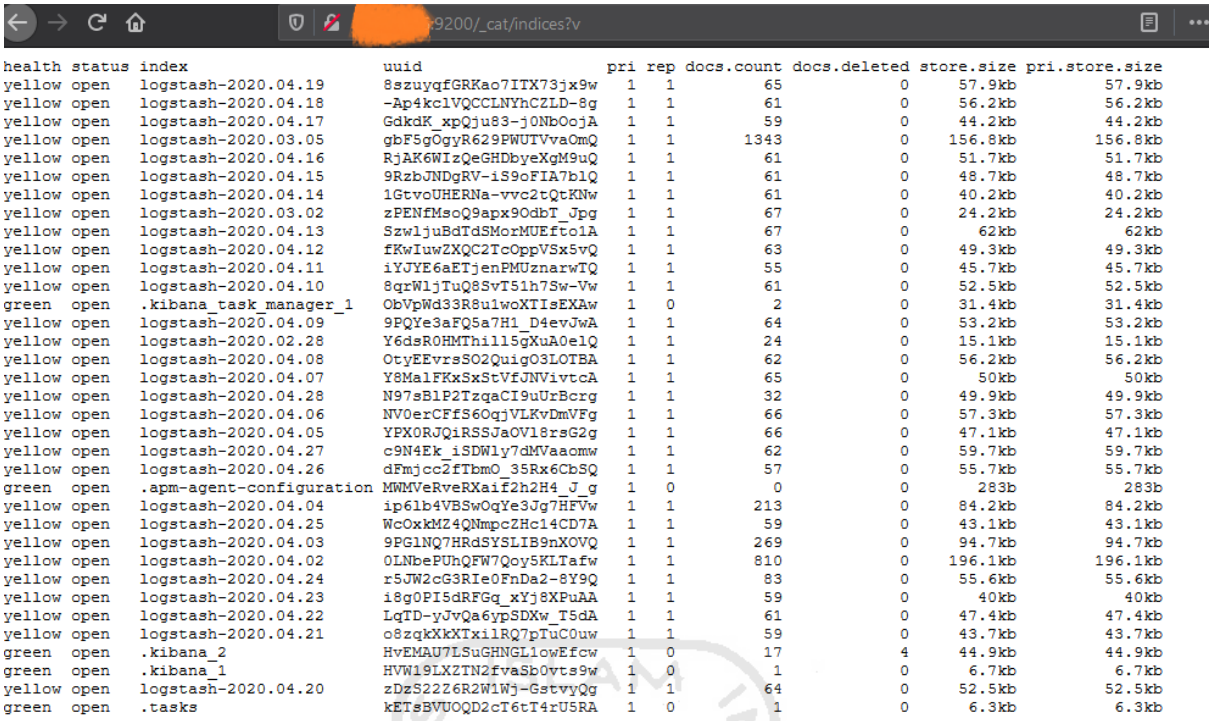
2.1.1. Elasticsearch

Elasticsearch adalah mesin pencari dan analitik terdistirbusi untuk semua basis data, termasuk tekstual, numerik, geospasiail, tersturktur, dan tidak terstruktur (Elastic, what is elasticsearch, 2020). Elasticsearch bersifat *open source* dan biasa digunakan pada pengembangan EFK *stack* dan ELK *stack*. Elasticsearch juga merupakan *database* noSQL yang berfokus pada *search engine database*. Sifatnya terbilang cukup unik karena di Elasticsearch kita bisa mengasumsikan indeks sebagai sebuah *database*, *types* sebagai *table*, dokumen sebagai *record* atau *row*, dan *mapping* sebagai skema tabel (Arslan, 2016). Pada **Gambar 2.2** terlihat bahwa wujud dari Elasticsearch yang sudah ter-*install* pada sebuah *server*:

```
root@visualisasi-poc:/home/visualisasi# curl http://127.0.0.1:9200
{
  "name" : "node-masterefk",
  "cluster_name" : "visualisasiefk",
  "cluster_uuid" : "nRFqbIFGSRe6cEjeDhfcIA",
  "version" : {
    "number" : "7.6.0",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "7f634e9f44834fbc12724506cc1da681b0c3b1e3",
    "build_date" : "2020-02-06T00:09:00.449973Z",
    "build_snapshot" : false,
    "lucene_version" : "8.4.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

Gambar 2.2 Cluster Elasticsearch di sebuah Server

Elasticsearch bekerja dalam beberapa tahapan. Pertama, Elasticsearch menerima *data flow* dari aggregator atau sumber lain yang mengirimkan data *log* tersebut (baik berupa *log*, *metric* sistem, dan web aplikasi). Kedua, data di *ingest* atau tepatnya terjadi *Data Ingestion* di Elasticsearch yaitu memproses data *log* tersebut (*parse*, *normalized*, dan *enriched*). Tahap terakhir adalah *indexing* terhadap data *log* ke dalam Elasticsearch. Setelah data *log* masuk ke Elasticsearch maka *user* dapat melakukan *query* kompleks terhadap data *log* mereka dan mengambil ringkasan data *log* untuk ditampilkan di Kibana (Elastic, what is elasticsearch, 2020). **Gambar 2.3** menunjukkan data log yang sudah masuk ke dalam Elasticsearch:



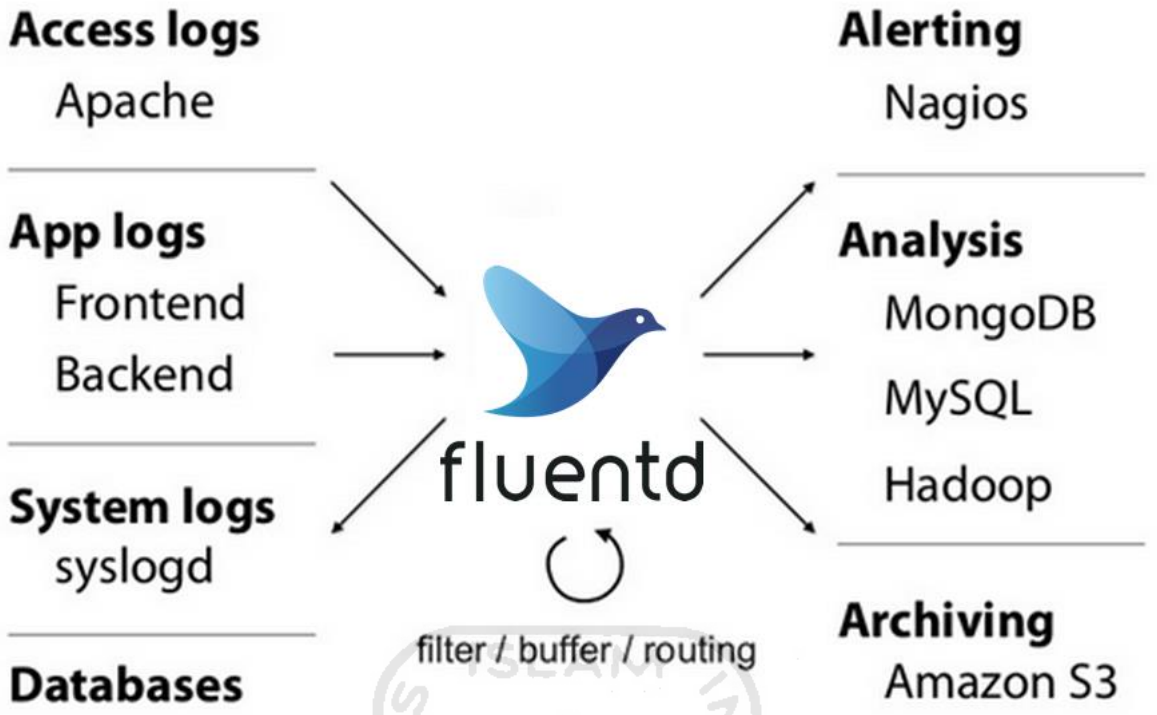
```
                                                  9200/_cat/indices?v
health status index                           uuid                   pri rep docs.count docs.deleted store.size pri.store.size
yellow open   logstash-2020.04.19           8szuyqfGRKao7ITX73jx9w   1   1         65            0     57.9kb         57.9kb
yellow open   logstash-2020.04.18           -Ap4kclVQCCLNYhCZLD-8g   1   1         61            0     56.2kb         56.2kb
yellow open   logstash-2020.04.17           GdkdK_xpQju83-j0NbOojA   1   1         59            0     44.2kb         44.2kb
yellow open   logstash-2020.03.05           gbF5gOgyR629PWUTVvaOmQ   1   1       1343            0    156.8kb        156.8kb
yellow open   logstash-2020.04.16           RjAK6WIzQeGHDbyeXgM9uQ   1   1         61            0     51.7kb         51.7kb
yellow open   logstash-2020.04.15           9RzbJNDgRV-iS9oFIA7blQ   1   1         61            0     48.7kb         48.7kb
yellow open   logstash-2020.04.14           1GtvoUHERNa-vvc2tQtKNw   1   1         61            0     40.2kb         40.2kb
yellow open   logstash-2020.03.02           zPENfMsoQ9apx9OdbT_Jpg   1   1         67            0     24.2kb         24.2kb
yellow open   logstash-2020.04.13           SzwljuBdTdSMorMUEftolA   1   1         67            0       62kb           62kb
yellow open   logstash-2020.04.12           fKwIuwZXQC2TcOppVSx5vQ   1   1         63            0     49.3kb         49.3kb
yellow open   logstash-2020.04.11           iYJYE6aETjenPMUznarwTQ   1   1         55            0     45.7kb         45.7kb
yellow open   logstash-2020.04.10           8qrWljTuQ8SvT51h7Sw-Vw   1   1         61            0     52.5kb         52.5kb
green  open   .kibana_task_manager_1        ObVpWd33R8u1woXTIsEXAw   1   0          2            0     31.4kb         31.4kb
yellow open   logstash-2020.04.09           9PQYe3aFQ5a7H1_D4evJwA   1   1         64            0     53.2kb         53.2kb
yellow open   logstash-2020.02.28           Y6dsR0HMThill5gXuA0elQ   1   1         24            0     15.1kb         15.1kb
yellow open   logstash-2020.04.08           OtyEEvrsSO2QuigO3LOTBA   1   1         62            0     56.2kb         56.2kb
yellow open   logstash-2020.04.07           Y8MalFKxSxStVfJNVivtcA   1   1         65            0       50kb           50kb
yellow open   logstash-2020.04.28           N97sBlP2TzqaCI9uUrBcrg   1   1         32            0     49.9kb         49.9kb
yellow open   logstash-2020.04.06           NV0erCFfS6OqjVLKvDmVFg   1   1         66            0     57.3kb         57.3kb
yellow open   logstash-2020.04.05           YPX0RJQiRSSJaOVl8rsG2g   1   1         66            0     47.1kb         47.1kb
yellow open   logstash-2020.04.27           c9N4Ek_iSDWly7dMVaaomw   1   1         62            0     59.7kb         59.7kb
yellow open   logstash-2020.04.26           dFmjcc2fTbmO_35Rx6CbSQ   1   1         57            0     55.7kb         55.7kb
green  open   .apm-agent-configuration      MWMVeRveRXaif2h2H4_J_g   1   0          0            0      283b           283b
yellow open   logstash-2020.04.04           ip6lb4VBSwOqYe3Jg7HFVw   1   1        213            0     84.2kb         84.2kb
yellow open   logstash-2020.04.25           WcOxkMZ4QNmpcZHc14CD7A   1   1         59            0     43.1kb         43.1kb
yellow open   logstash-2020.04.03           9PGlNQ7HRdSYSLIB9nXOVQ   1   1        269            0     94.7kb         94.7kb
yellow open   logstash-2020.04.02           0LNbePUhQFW7Qoy5KLTafw   1   1        810            0    196.1kb        196.1kb
yellow open   logstash-2020.04.24           r5JW2cG3RIe0FnDa2-8Y9Q   1   1         83            0     55.6kb         55.6kb
yellow open   logstash-2020.04.23           i8g0PI5dRFGq_xYj8XPuAA   1   1         59            0       40kb           40kb
yellow open   logstash-2020.04.22           LqTD-yJvQa6ypSDXw_T5dA   1   1         61            0     47.4kb         47.4kb
yellow open   logstash-2020.04.21           o8zqkXkXTxilRQ7pTuC0uw   1   1         59            0     43.7kb         43.7kb
green  open   .kibana_2                     HvEMAU7LSuGHNGL1owEfcw   1   0         17            4     44.9kb         44.9kb
green  open   .kibana_1                     HVW19LXZTN2fvaSb0vts9w   1   0          1            0      6.7kb          6.7kb
yellow open   logstash-2020.04.20           zDzS22Z6R2W1Wj-GstvyQg   1   1         64            0     52.5kb         52.5kb
green  open   .tasks                        kETsBVUOQD2cT6tT4rU5RA   1   0          1            0      6.3kb          6.3kb
```

Gambar 2.3 Data Log yang ada di dalam Elasticsearch

Elasticsearch menyimpan data *log* dalam format JSON. Struktur data Elasticsearch disebut dengan *inverted index* yaitu memungkinkan pencarian *full text* secara cepat. Pada saat *indexing* data *log*, Elasticsearch menyimpan data tersebut dan melakukan *inverted index* sehingga data dapat di cari secara *realtime* (Elastic, what is elasticsearch, 2020). Adapun penelitian yang dilakukan oleh (Sartika, 2020) menggunakan Elasticsearch sebagai *query* data untuk mengolah data Portal Pengembangan dan Pembinaan Sumber Daya Manusia (PPSDM) yang dimiliki oleh Lembaga Kebijakan Pengadaan Barang/Jasa Pemerintah (LKPP). Dalam menangani masalah monitoring dan evaluasi data pada saat terjadinya sertifikasi dan pelatihan yang ada di Portal PPSDM latensi data yang terjadi berkurang selama 2,5 detik.

2.1.2. Fluentd

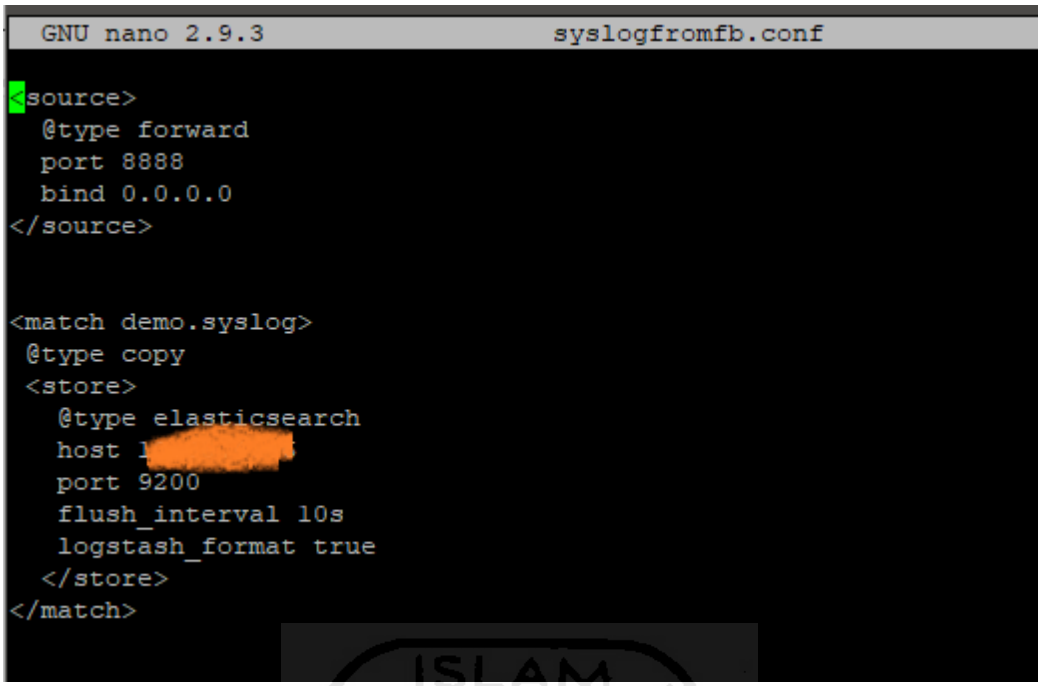
Fluentd merupakan *open source data collector* yang berfungsi menyatukan berbagai jenis *log* dari semua macam-macam *log* yang ada (Fluentd, Main pages, 2020). *Tools* yang dikembangkan oleh perusahaan Treasure Data ini bersifat *open source* dan sampai saat ini telah banyak digunakan oleh berbagai perusahaan untuk menangani masalah log yang ada. Sifat dari Fluentd juga dapat sebagai Aggregator yaitu memungkinkan Fluentd melakukan *filtering* data *log* yang masuk sebelum mengarahkan data tersebut ke sebuah *Storage*, dalam kasus kali ini adalah Elasticsearch. **Gambar 2.4** menjelaskan bagaimana cara kerja Fluentd secara umumnya:



Gambar 2.4 Proses kerja Fluentd (Fluend, 2020)

Data yang disimpan di Fluentd berformat JSON sebisa mungkin karena *downstream* data *processing* lebih mudah dan fleksibel. *Plugins* yang ada di Fluentd juga terbilang cukup banyak yaitu lebih dari 500 *plugins* dan terus bertambah dengan seiringnya waktu melalui bantuan perusahaan pengembang maupun user yang memakai Fluentd. Saat ini telah digunakan lebih dari 2000 perusahaan hingga di *deploy* di 50000 server dan terus bertambah. Selain hal tersebut Fluentd juga memerlukan sumberdaya yang sedikit saat digunakan dan juga memiliki kelebihan *memory file* berbasis *buffering* sebagai tindakan pencegahan untuk kehilangan data (Fluentd, architecture, 2020).

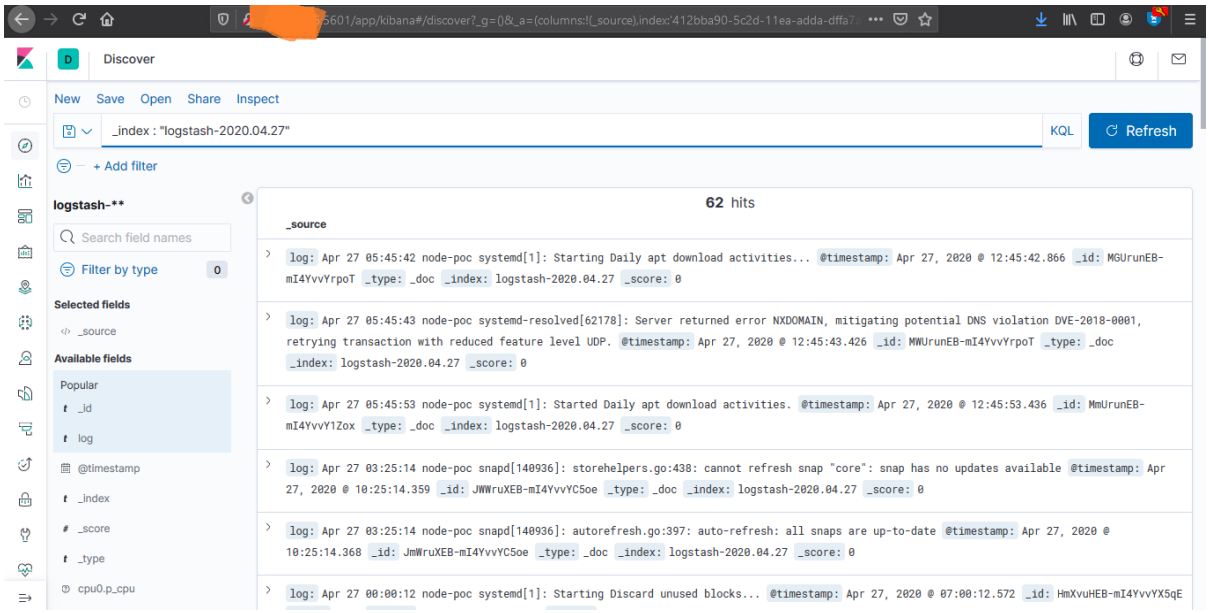
Pada sisi konfigurasi memiliki kelebihan yaitu sangat simpel dan mudah dipahami bagi pemula yang baru menggunakan teknologi ini. Fluentd dapat bertindak sebagai Aggregator yang berfungsi untuk mengarahkan data log dari *log collector* Fluentbit ke Elasticsearch dan memberikan *filtering* bila diperlukan atau dapat bertindak sebagai *log collector* saja. **Gambar 2.5** adalah contoh konfigurasi file Fluentd yang ada:



Gambar 2.5 Contoh konfigurasi Fluentd

2.1.3. Kibana

Kibana merupakan *open source* sistem *front end* yang berada di atas Elastic-stack dan menyediakan pencarian dan visualisasi data yang telah di-*indexing* di dalam Elasticsearch. Kibana juga bertindak sebagai *user interface* yang bertujuan untuk memonitor, mengelola, dan mengamankan sebuah Elastic *cluster/stack*. Dikembangkan mulai dari tahun 2013 di komunitas Elastic hingga sekarang Kibana telah menjadi jendela bagi Elastic stack itu sendiri dan menjadi sebuah portal bagi *user* dan perusahaan (Elastic, what is kibana, 2020). **Gambar 2.6** adalah contoh installasi Kibana dan memuat beberapa data log yang telah masuk ke dalam Elasticsearch:



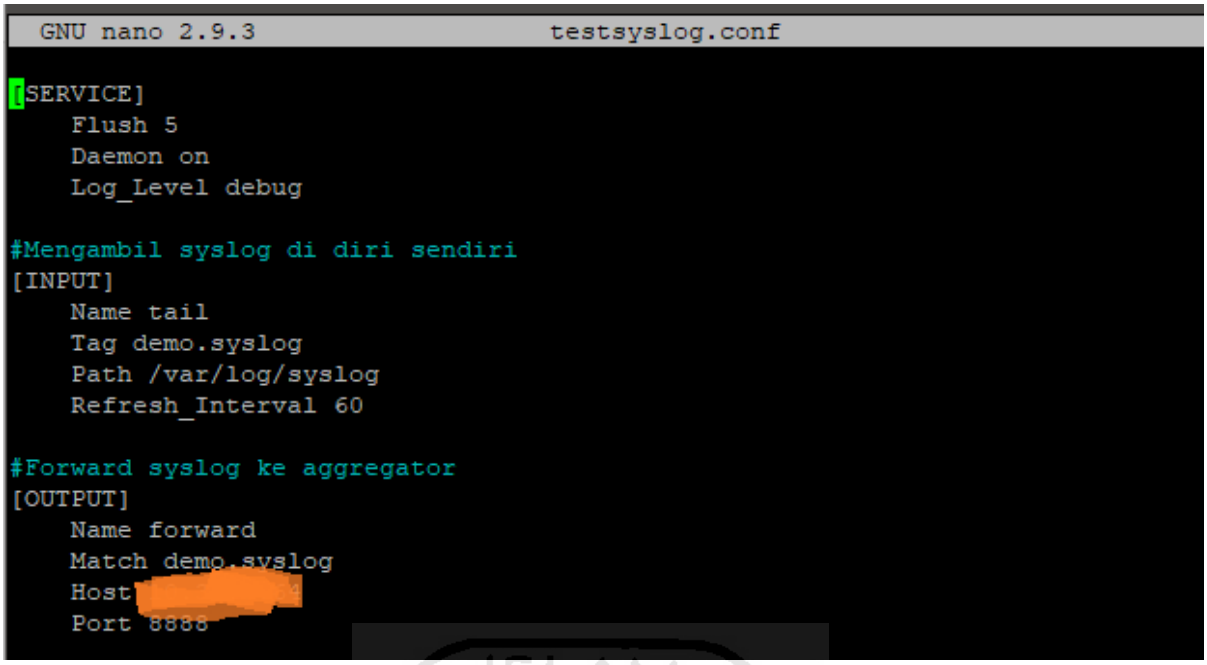
Gambar 2.6 Data Log yang ditampilkan Kibana

Kibana bekerja dengan cara melihat dan mengolah sebuah visualisasi indeks dari Elasticsearch maupun dari berbagai indeks lainnya. Indeks terbentuk bisa dari Logstash, Beat, Fluentd, ataupun Fluentbit. Indeks berasal dari Fluentd bertindak sebagai aggregator yang mendapatkan data *log* dari Fluentbit sebagai *log collector*. *Interface* Kibana sendiri mengijinkan user untuk mengolah data *log* di Elasticsearch dan kemudian memvisualisasikan data *log* tersebut sesuai dengan bagan yang tersedia.

Penelitian yang dilakukan oleh (Wicaksono, 2020) menggunakan Kibana sebagai visualisasi data yang digunakan untuk melihat kualitas *internet mobile broadband* dari berbagai macam Internet Service Provider (ISP) untuk pengguna yang ada di daerahnya tersebut. Adapun penelitian yang dilakukan oleh (Erwinsyah, 2020) bertempat di BSI juga menggunakan Kibana sebagai visualisasi data *log* sehingga memudahkan proses monitoring untuk *log server* di BSI tersebut.

2.1.4. Fluentbit

Fluentbit merupakan *open source platform* yang digunakan sebagai *log processor* atau *log collector* untuk mengambil data *log* dari berbagai sumber. Kemudian menggabungkan log-log tersebut untuk dikirimkan ke berbagai lokasi. Penjelasan mudahnya Fluentbit merupakan versi mini dari Fluentd dan cocok dipakai di *environment* yang memiliki kapasitas terbatas. **Gambar 2.7** adalah sebuah contoh dari konfigurasi Fluentbit untuk mengambil data *log syslog*.



Gambar 2.7 Contoh konfigurasi Fluentbit

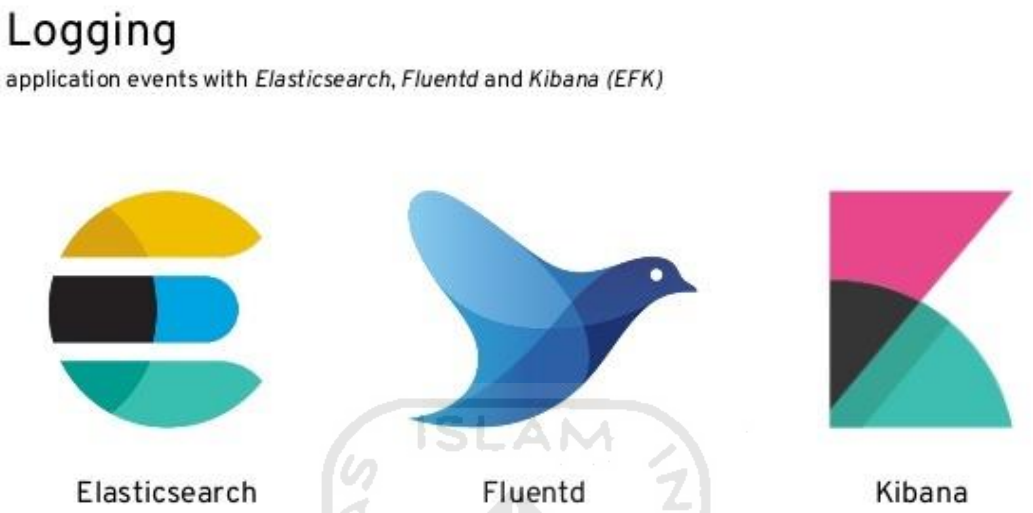
Perbedaan mendasar lainnnya antara Fluentbit dan Fluentd ialah Fluentbit memakai Bahasa C, *lightweight*/ringan sekali karena hanya memakan sekitar 450KB *memory*, dan tidak bisa digunakan sebagai aggregator. Fluentbit digunakan di *server* untuk mengambil data *log* yang diperlukan. Hal lainnya yang perlu diperhatikan adalah kombinasi Fluentd dan Fluentbit ini sangat populer digunakan dalam kluster kubernetes.

2.2 Centralized Log Management

Setiap aplikasi, sistem, atau *service* yang berjalan pada *server* biasanya menghasilkan *log* secara otomatis. *Log* ini sangat penting bagi seorang *SysAdmin*, karena dengan *log* tersebut seorang *SysAdmin* dapat mengetahui bagaimana aplikasi, sistem, *service*, dan *server* tesebut bekerja saat ini maupun di masa lampau. Dengan menggunakan *log* maka seorang *SysAdmin* dapat menemukan *error* yang ada di aplikasi, sistem, *service*, ataupun *sever*. Namun hal tersebut sangat memakan waktu bagi seorang *SysAdmin* untuk mencari *error log* di antara banyaknya log-log pada *server*, *service,* sistem, dan aplikasi yang ada. Untuk mengatasi masalah tersebut *SysAdmin* membutuhkan *Centralized Log* untuk memonitor seluruh *server* dengan mudah.

Centralized Log atau sering disebut dengan *Centralized Log Management* (CLM) adalah tipe sistem *logging* solusi untuk menggabungkan semua data *log* yang dimiliki. Selanjutnya *log* tersebut di arahkan menuju sebuah pusat *log* yang memiliki UI untuk dilakukan visualisasi. *Centralized Log Management* dibuat untuk memudahkan pekerjaan seorang *SysAdmin*. CLM tidak hanya menyediakan fitur bagi *SysAdmin* untuk mengambil data *log* saja, selain fitur tersebut CLM

juga menggabungkan, menganalisis, dan memberikan gambaran informasi dengan cepat dan jelas (Morgan, 2016). **Gambar 2.8** merupakan contoh arsitektur CLM yang menggunakan Elasticsearch, Fluentd, dan Kibana:



Gambar 2.8 Contoh salah satu Arsitektur dari CLM

Selain menggunakan EFK *stack* di atas dapat ditambahkan juga *log collector* yaitu Fluentbit untuk menunjang kinerja EFK *stack* dalam pengembangan CLM. CLM juga membiarkan seorang *SysAdmin* mengelola data mereka secara bebas dan seefisien mungkin. *SysAdmin* memiliki kemampuan untuk mengakses data *log* secara instan menggunakan CLM tersebut daripada menunggu beberapa jam atau hari dengan mencari secara manual. Keuntungan lainnya adalah perusahaan yang menggunakan CLM tersebut akan lebih dinamis, menguntungkan, aman (Morgan, 2016).

Penelitian yang dilakukan oleh (Gunawan, 2018) menggunakan NoSql untuk mengukur *respon time* pada database berbasis *document stored.* Hasil penelitian menunjukkan *query read* pada NoSql *database* memiliki response yang cepat dibanding *query* untuk proses *create, update,* dan *delete. Query update* data pada Nosql *database* memiliki *response times* yang paling lama dibanding *query* untuk proses *create, read* dan *delete*. Sedangkan *query* untuk proses *delete* data pada Nosql *database* memiliki *response times* yang lebih cepat dibanding *query create* data. Penelitian yang dilakukan oleh (Lei, et al., 2020) menggunakan teknologi EFK stack untuk memonitor kinerja baterai dan memaksimalkan kemampuan baterai dalam kendaraan listrik. Hasil

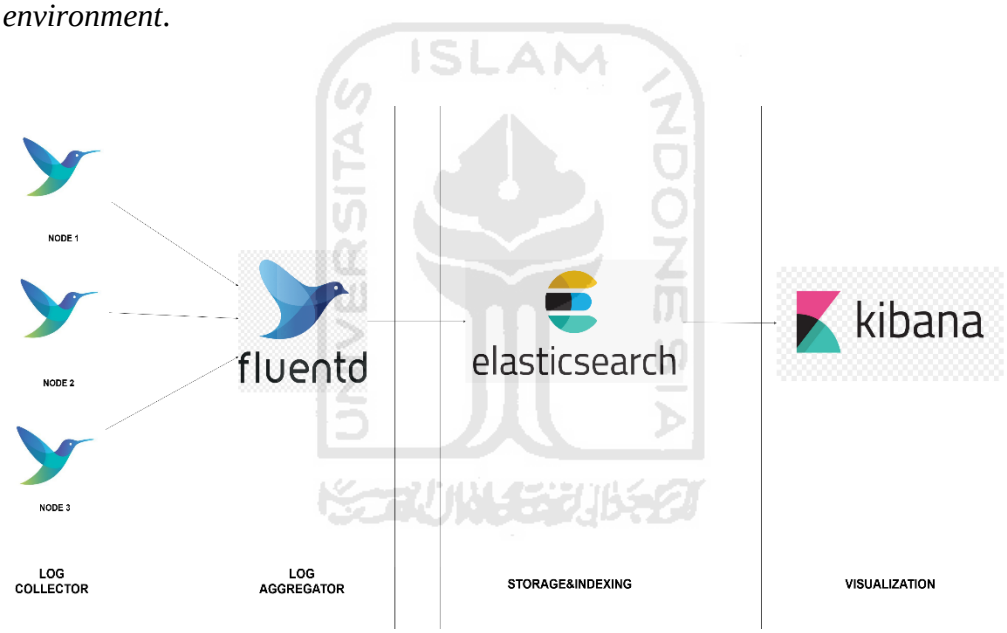
akhir penelitian menunjukkan kelayakan, keandalan, serta presisi penggunaan baterai dilihat dari Algoritma EFK dapat memenuhi standar nasional pada kendaraan listrik.

Penelitian yang dilakukan oleh (Yugitama, Rachman, & Sulistyo, 2020) mengenai dasbor log adalah dengan membuat dasbor untuk memonitor sebuah Honeypot. Dasbor log melihat data serangan yang ditujukan pada server Honeypot tersebut dan melakukan otomatisasi laporan melalui aplikasi Telegram. Hasil dari penelitian tersebut adalah sistem berhasil menampilkan visualisasi dan otomatisasi laporan sesuai dengan data yang terdapat pada log pada server berupa alamat IP penyerang, port target beserta layanannya, waktu serangan dan *binary* malware. Penelitian yang dilakukan oleh (J, Dubey, B, Rao, & Rao, 2018) menunjukkan visualisasi data menggunakan Elasticsearch dan Kibana pada Platfrom Github Repository bertujuan untuk melihat cerita dari berbagai *code* yang diambil dari *Repository* tersebut serta melihat bagaimana Github *Repository* bekerja. Hasil akhir dari penelitian tersebut menunjukkan bahwa Github telah menjadi salah satu tempat *hosting code* paling familiar yang dipakai individu maupun organisasi. Dari beberapa data yang diambil dan beberapa *research question* (RQ) yang telah divisualisasikan menunjukkan sebuah model prediktif dibangun di atas data-data yang dikumpulkan Github dapat menjadi ruang lingkup untuk penelitian lebih lanjut di masa yang akan datang.

BAB III
PELAKSANAAN MAGANG

3.1 Manajemen Proyek

Hal pertama yang dilakukan ketika mengembangkan CLM ini adalah membuat arsitektur aplikasinya. Arsitektur aplikasi ini bertujuan untuk memudahkan pengembangan aplikasi agar terarah, terstruktur, dan terencana. Proses perencanaan arsitektur ini dibuat bersama tim *Site Realibility Engineer* (SRE) BSI. **Gambar 3.1** merupakan hasil diskusi antara pemagang dengan Tim SRE BSI. Selain menetapkan arsirtektur CLM, hasil lain dari diskusi adalah menentukan dua cara pengembangan CLM yaitu dikembangkan di luar kubernetes *environment* dan di dalam kubernetes *environment*.



Gambar 3.1 Hasil diskusi Arsitektur EFK stack dan Fluentbit

Penjelasan arsitektur CLM pada **Gambar 3.1** adalah sebagai berikut:

1. Fluentbit digunakan sebagai pengumpul *log (log collector)* di setiap server dan kemudian meneruskan semua *log* yang sudah diambil ke Fluentd.
2. Fluentd sebagai *aggregator* yang menerima *log* dari berbagai *log collector*(Fluentbit).
3. Elasticsearch digunakan untuk menyimpan data *log* yang sudah dikirimkan oleh Fluentd dan sebagai *query log* untuk Kibana.

4. Kibana digunakan untuk visualisasi terhadap *log* yang telah tersimpan di dalam Elasticsearch. Data *log* yang tersimpan di Elasticsearch di-*query* oleh Kibana untuk ditampilkan dengan bentuk *chart* yang tersedia.

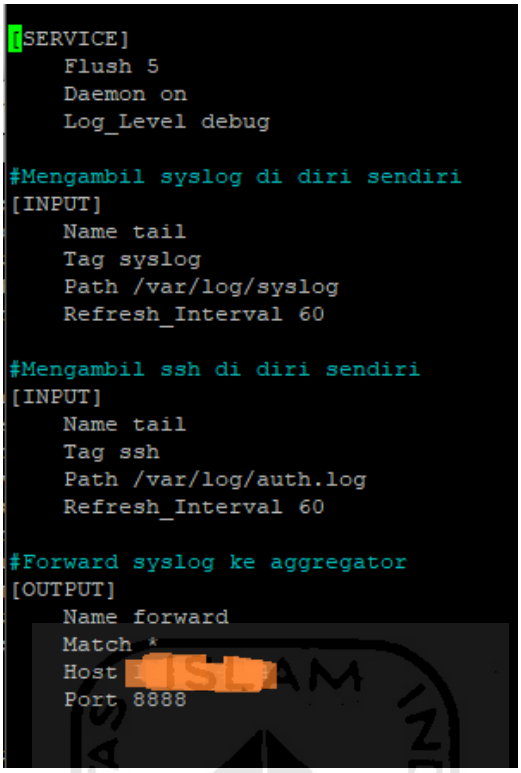
3.2 EFK stack dan Fluentbit di luar Kubernetes Environment

Saat mengembangkan CLM di luar Kubernetes *environment* kita dapat menentukan *server* mana saja yang perlu di-*monitoring* dan *log* mana saja yang perlu diambil oleh Fluentbit. Sedangkan untuk Fluentd sebagai *aggregator* cukup menyediakan satu *server* saja dan *server* ini bisa diletakkan bersamaan dengan Elasticsearch dan Kibana. Namun, untuk pengembangan kali ini *server* Fluentd terpisah dengan *server* Elasticsearch dan Kibana. Arsitektur CLM di luar Kubenetes environment ini sama dengan **Gambar 3.1**

3.2.1 Konfigurasi Fluentbit

Langkah awal yang dilakukan dalam pengembangan di luar Kubernetes *environment* ini yaitu menentukan *server* mana saja yang perlu dilakukan monitoring dan melakukan installasi Fluentbit di dalam *server* tersebut. Fluentbit di sini bertindak sebagai *log collector* yang bertugas untuk men-*tail log* yang telah dipilih di dalam konfigurasi file Fluentbit. Sebelum melakukan konfigurasi Fluentbit, terlebih dahulu dilakukan installasi aplikasi pendukung pada *server* yaitu: Compiler GCC atau clang, Cmake, Flex dan Bison, dan Fluentbit versi terbaru.

Setelah memastikan empat aplikasi pendukung di atas ter-install, dilanjutkan dengan mengkonfigurasi file Fluentbit untuk mengambil data *log* yang diinginkan. Kemudian melalui file konfigurasi tersebut Fluentbit mengirimkan data *log* ke Fluentd sebagai *aggregator* yang ada. Contoh file konfigurasi untuk mengambil data sys*log* dan ssh auth *log* di sebuah *server* dapat dilihat pada **Gambar 3.2**.



Gambar 3.2 Konfigurasi file Fluentbit

Berikut adalah penjelasan isi komponen file konfigurasi pada **Gambar 3.2**:

Service] berguna sebagai tingkah laku dari file konfigurasi tersebut. ***Flush*** digunakan sebagai tanda data dikirimkan oleh konfigurasi dan diberi indikator 5 menunjukkan setiap 5 detik data akan diteruskan ke ***Output*** tujuan. ***Daemon on*** digunakan agar konfigurasi dapat berjalan di latar belakang *server*. Terakhir ***Log_Level*** berguna untuk memberikan informasi jenis data *log* yang diambil (dalam hal ini debug mencakup *info* dan *error log*).

[***Input***] berguna sebagai informasi *log* apa yang akan diambil pada sebuah server. ***Name*** digunakan sebagai identifikasi dan bagaimana tingkah laku Fluentbit. Pada contoh kali ini digunakan *tail* yaitu *plugins* untuk membaca *log* terbaru secara *realtime.* ***Tag*** digunakan sebagai penanda ***Input*** agar dapat dibaca oleh ***Match*** pada ***Output***. ***Path*** merupakan lokasi *log* yang akan dibaca. Terakhir ***Refresh_Interval*** sebagai waktu data *log* dibaca oleh Fluentbit(dalam hal ini setiap 60 detik Fluentbit akan meng-*update log* baru dari data *log* yang ada di ***PATH)***.

[***Output***] berguna sebagai tujuan kemana data *log* akan dikirimkan oleh Fluentbit. ***Name*** digunakan sebagai identifikasi dan bagaimana tingkah laku Fluentbit. Pada contoh kali ini digunakan *forward* yaitu *plugins* yang digunakan oleh Fluentbit dan Fluentd untuk berkomunikasi satu dengan lainnya. ***Match*** adalah penanda yang digunakan untuk menangkap ***Tag*** dan simbol * digunakan untuk menangkap semua ***Tag*** yang ada. ***Host*** digunakan sebagai tujuan data *log* yang

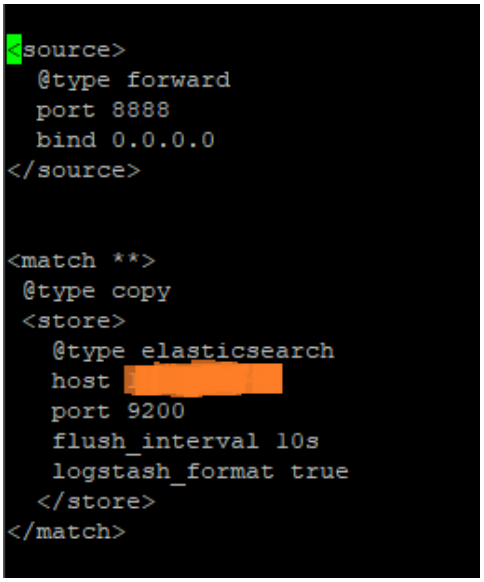
akan dikirimkan dan dalam kasus kali ini adalah IP address dari Fluentd yang bertindak sebagai *Aggregator*. ***Port*** digunakan sebagai *port* untuk menerima data *log* tersebut.

Setelah melakukan konfigurasi file tersebut saatnya untuk menjalankan perintah agar file konfigurasi dapat berjalan dengan perintah *Fluent-bit -c namakonfigurasi.conf.* Saat perintah dijalankan, Fluentbit akan berjalan di latar belakang server. Untuk menghentikan file konfigurasi tersebut maka cukup dengan perintah *kill -9 idkonfigurasi.* Maka otomatis file konfigurasi yang berjalan di latar belakang akan berhenti dengan sendirinya dan Fluentbit berhenti mengirimkan data *log* ke Fluentd.

3.2.2 Konfigurasi Fluentd

Langkah selanjutnya setelah meng-*install* Fluentbit adalah melakukan installasi Fluentd sebagai *aggregator*. *Aggregator* berfungsi untuk meneruskan *log* yang didapatkan dari Fluentbit menuju ke berbagai *storage* tujuan sesuai dengan file konfigurasi yang ada. Dalam pengembangan kali ini *storage* tujuan adalah Elasticsearch meskipun tidak dipungkiri dalam konfigurasi file tersebut data *log* yang datang dari Fluentbit bisa diteruskan ke *storage* tujuan yang lainnya. Sama seperti Fluentbit yaitu sebelum mengkonfigurasi file untuk *log aggregator* dilakukan terlebih dahulu installasi aplikasi pendukung yaitu: Ruby Gem, Fluentd, dan Plugins Fluentd untuk elasticsearch.

Saat tiga aplikasi pendukung di atas sudah ter-*install*, Fluentd dapat digunakan sebagai *log aggregator* untuk menerima semua *log collector*. Pada **Gambar 3.3** merupakan contoh file konfigurasi Fluentd sebagai *log aggregator*.



Gambar 3.3 Konfigurasi file Fluentd

Berikut merupakan penjelasan isi komponen file konfigurasi pada **Gambar 3.3**:

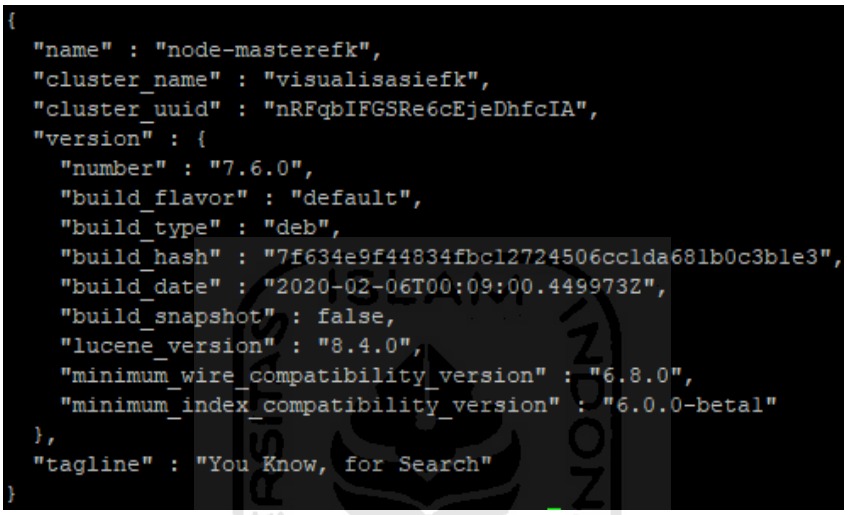
<***Source***> digunakan sebagai penerima untuk menyaring data *log* Fluentbit yang sudah *stream* ke Fluentd. Penggunaan ***@type forward*** sebagai *plugins* penghubung antara Fluentbit dan Fluentd yang ada. ***Port*** adalah keterangan *port* mana yang dibuka untuk menerima data *log* tersebut. Terakhir ***bind*** adalah sebagai pengunci untuk mengikat *ip address* mana yang boleh diterima oleh Fluentd dan ***0.0.0.0*** menunjukkan semua *ip address* diterima oleh Fluentd.

<***match ****> digunakan sebagai tujuan data *log* yang sudah masuk ke dalam Fluentd akan diteruskan. ***@type copy*** digunakan sebagai *plugins* Fluentd untuk menyalin data *log* tersebut. Kemudian digunakan <***store***> untuk menyimpan data yang sudah di-*copy* tadi. Di dalam <***store***> terdapat ***type*** yaitu *elasticsearch* sebagai *plugins* untuk menyimpan data *log* ke Elasticsearch. ***Host*** digunakan sebagai alamat Elasticsearch yang ada. ***Port*** digunakan sebagai *port* yang dibuka dan diijinkan masuk ke Elasticsearch. ***Flush_interal*** digunakan sebagai waktu untuk meneruskan data *log* dari Fluentd ke Elasticsearch tersebut. Terakhir ***logstash_format*** digunakan untuk mengubah format data *log* menjadi *logstash* agar dapat dibaca oleh Elasticsearch.

Langkah berikutnya adalah menjalankan perintah agar file konfigurasi tersebut dapat berjalan. Perintahnya hampir sama dengan Fluentbit yaitu *Fluentd -c namakonfigurasi.conf.* Ketika menjalankan perintah tersebut Fluentd akan berjalan di latar belakang server. Untuk menghentikan konfigurasi tersebut maka cukup dengan perintah *kill -9 idkonfigurasi.* Lalu secara otomatis konfigurasi yang berjalan di latar belakang akan berhenti dengan sendirinya dan Fluentd berhenti mengirimkan data log ke Elasticsearch.

3.2.3 Konfigurasi Elasticsearch

Ketika konfigurasi Fluend selesai dilakukan, dilanjutkan dengan menyiapkan sebuah *storage/*penyimpanan untuk menampung log-log yang terkirim tersebut. Penyimpanan pada pengembangan ini adalah Elasticsearch yang sepaket dengan visualisasinya yaitu Kibana. Konfigurasi yang dilakukan pada sisi Elasticsearch cukup mudah karena hanya perlu melakukan installasi dan pemberian nama *cluster* di *server* yang akan menjadi tempat *storage* data *log* ini. **Gambar 3.4** menunjukkan hasil konfigurasi Elasticsearch di *server* yang ada.



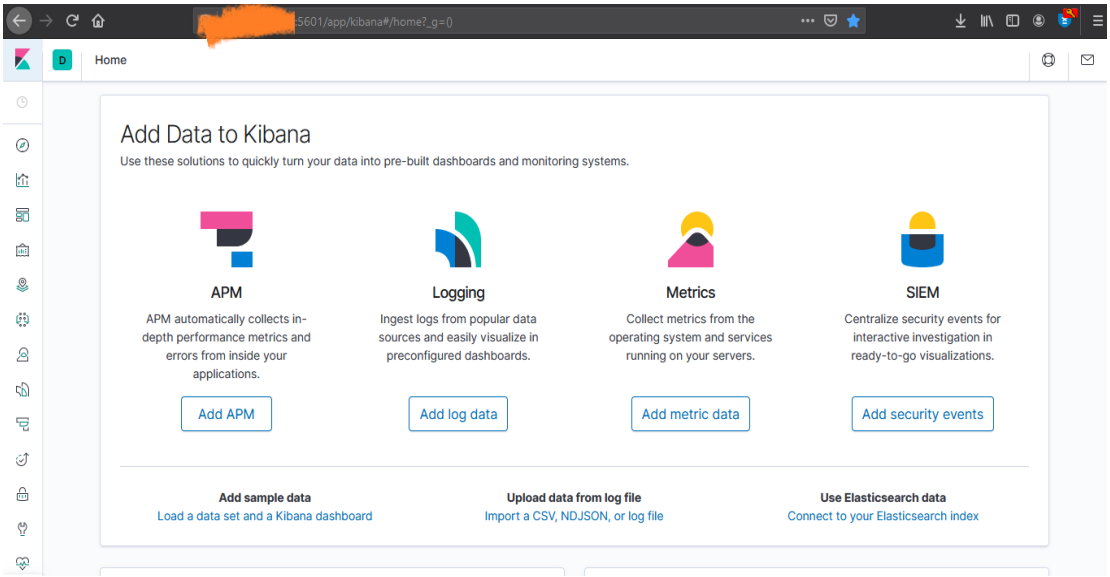
```
{
  "name" : "node-masterefk",
  "cluster_name" : "visualisasiefk",
  "cluster_uuid" : "nRFqbIFGSRe6cEjeDhfcIA",
  "version" : {
    "number" : "7.6.0",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "7f634e9f44834fbc12724506cc1da681b0c3b1e3",
    "build_date" : "2020-02-06T00:09:00.449973Z",
    "build_snapshot" : false,
    "lucene_version" : "8.4.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```


Gambar 3.4 *Cluster* Elasticsearch yang telah dibuat

Adapun **Gambar 3.4** merupakan Elastic Cluster yang telah dibuat dan memberi informasi mengenai: Nama *cluster*, kapan *cluster* dibuat, versi *cluster* yang telah dibuat, dan lain sebagainya.

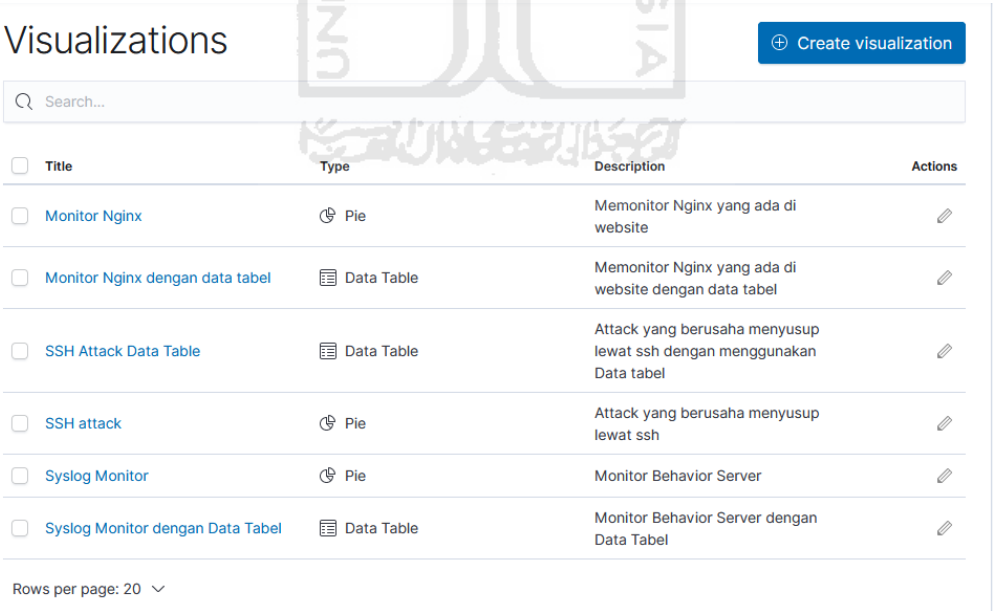
3.2.4 Konfigurasi Kibana

Setelah Elasticsearch selesai terinstall, langkah selanjutnya adalah installasi Kibana sebagai visualisasi dari pengembangan CLM ini. Konfigurasi di Kibana juga tidak terlalu rumit dan sama dengan konfigurasi yang telah dilakukan di Elasticsearch. Hanya perlu untuk meng-*install* Kibana pada *server* yang akan dijadikan tempat untuk visualisasi data *log* ini. **Gambar 3.5** merupakan hasil dari installasi yang dilakukan yaitu halaman awal Kibana.



Gambar 3.5 Halaman awal Kibana

Sesudah memastikan setiap komponen (Fluentbit, Fluentd, Elasticsearch, dan Kibana) berjalan dengan lancar dan memeriksa *stream log* yang masuk ke Kibana langkah selanjutnya yang dilakukan yaitu melakukan visualisasi. Terdapat tiga visualiasi yang dibuat dan dari ketiga visualisasi tersebut masing masing dibuat dua tipe data. **Gambar 3.6** merupakan hasil dari beberapa visualisasi yang sudah dibuat:

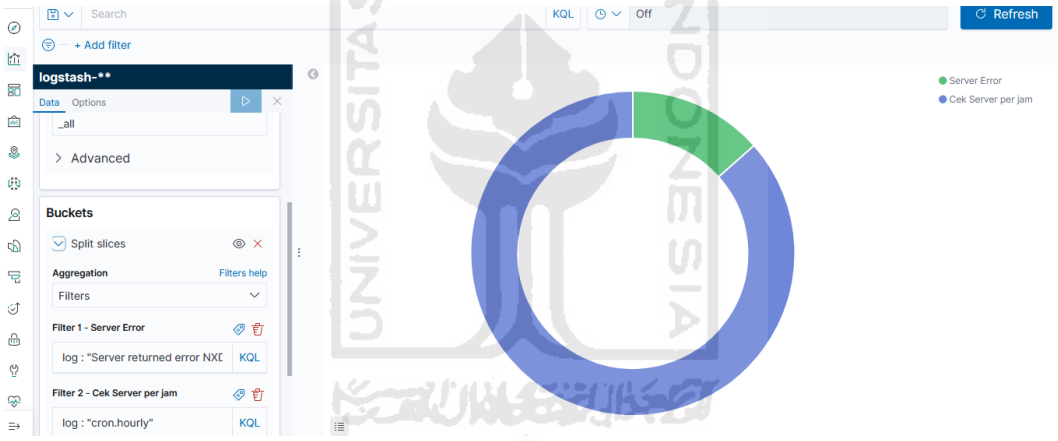


Gambar 3.6 Hasil visualisasi yang telah dibuat

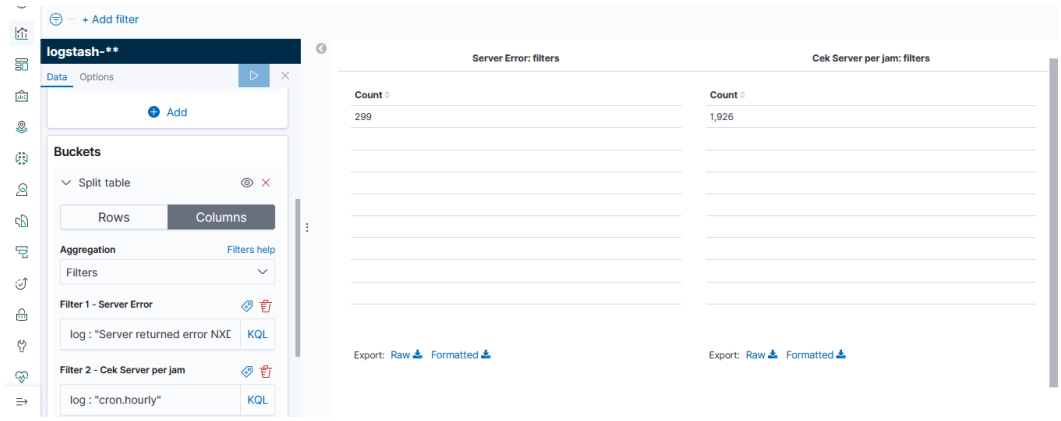
3.2.5 Hasil Visualisasi dan Dasbor Sistem Pencatatan Log

Pada saat selesai melakukan konfigurasi yang diperlukan untuk mengambangkan CLM(Fluentbit, Fluentd, Elasticsearch, Kibana) langkah terakhir yaitu membuat Visualisasi dan dasbor sistem pencatatan *log*/CLM. Sebelum membuat dasbor hal pertama yang dilakukan adalah visualisasi *log* satu per satu terlebih dahulu. Visualisasi ini akan dibagi menjadi tiga yaitu: Visualisasi *log* Syslog, Visualisasi *log* SSH, dan Visualisasi *log* Nginx. Bentuk selanjutnya dari masing-masing visualisasi tersebut adalah *pie chart* dan *data table*. Pemilihan grafik *pie chart* pada *syslog* digunakan untuk membandingkan seberapa sering server dalam kondisi *up* dengan kondisi *down*, untuk *ssh attack* digunakan untuk melihat seberapa banyak serangan yang dilakukan dibandingkan dengan sukses masuk ke dalam server, terakhir pada *nginx* digunakan untuk melihat perbandingan ketika *webserver* sedang *up* dan *down*. Sedangkan *data table* digunakan untuk menghitung seluruh log yang masuk pada ketiga visualisasi tersebut.

Pertama adalah visualisasi *log* syslog sebagaimana **Gambar 3.7** dan **Gambar 3.8**:



Gambar 3.7 Visualisasi *syslog* berbentuk *pie chart*

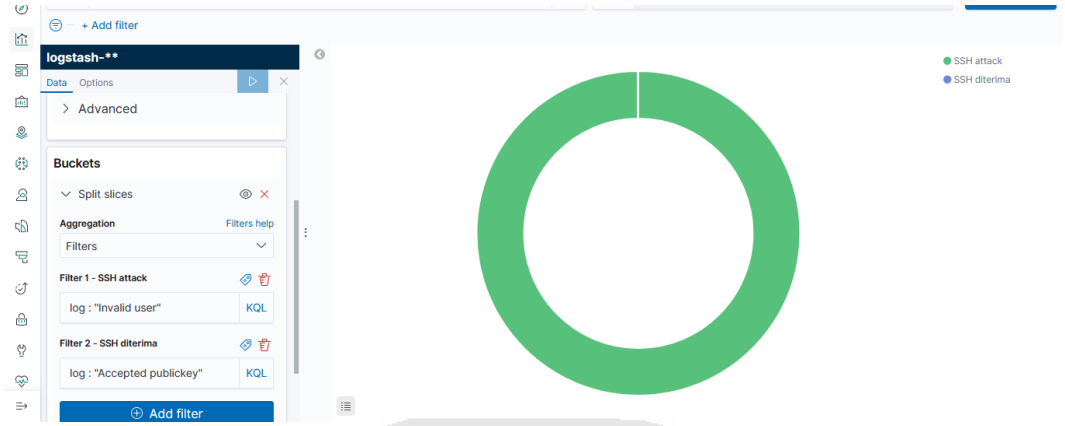


Gambar 3.8 Visualisasi *syslog* berbentuk *data table*

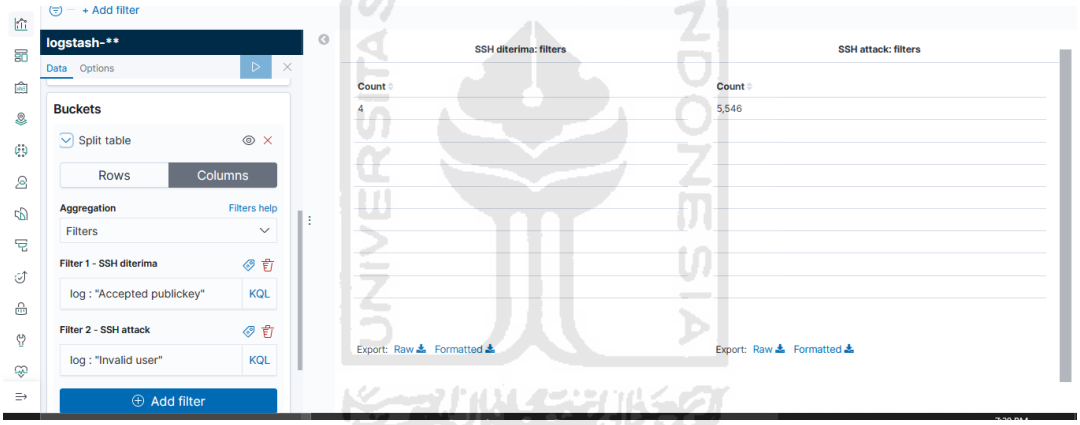
Terlihat pada **Gambar 3.7** dan **Gambar 3.8** visualisasi syslog ini bertujuan untuk memonitor *behavior server* yang ada. Filter *cek server perjam* adalah *log cron hourly* yang bertugas

memonitor apakah server *up* setiap jamnya. Sedangkan filter *server error* bertugas memonitor seberapa sering server *down*. Visualisasi dibuat dua dengan model *pie chart* dan *data table*.

Selanjutnya adalah visualisasi mengenai *log* SSH di sebuah server sebagaimana terlihat pada **Gambar 3.9** dan **Gambar 3.10**:



Gambar 3.9 Visualisasi *SSH Attack* berbentuk *pie chart*

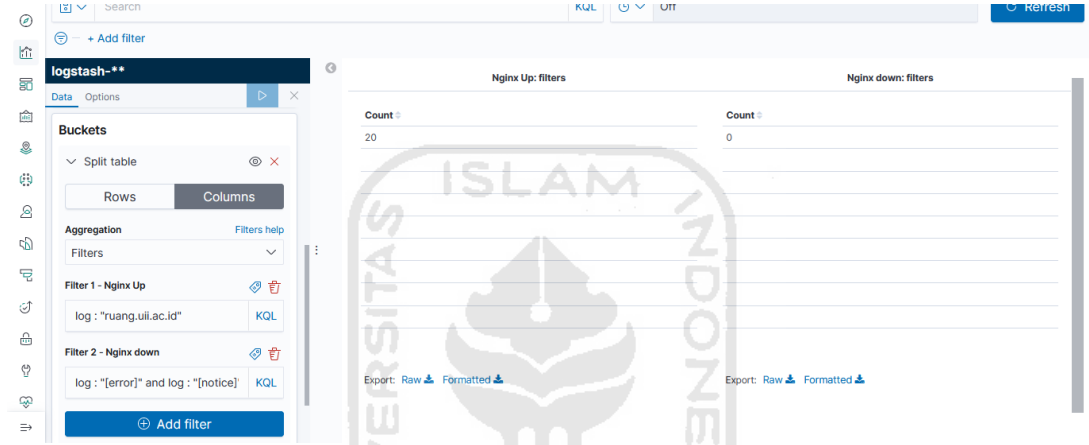


Gambar 3.10 Visualisasi *SSH Attack* berbentuk *data table*

Terlihat pada **Gambar 3.9** dan **Gambar 3.10** visualisasi *log* SSH bertujuan untuk memonitor jumlah *ssh attack/password brute force* yang dilakukan terhadap server tersebut dan melihat jumalah *ssh login key* yang berhasil masuk. Kedua hal tersebut dicek juga dengan menggunakan fitur filter *accepted publickey* dan *invalid user*. Visualisasi ini juga dibuat dengan dua model yaitu *pie chart* dan *date table*.

Terakhir adalah visualisasi mengenai *log* Nginx pada sebuah *website/web server* yang terlihat pada **Gambar 3.11** dan **Gambar 3.12** berikut:

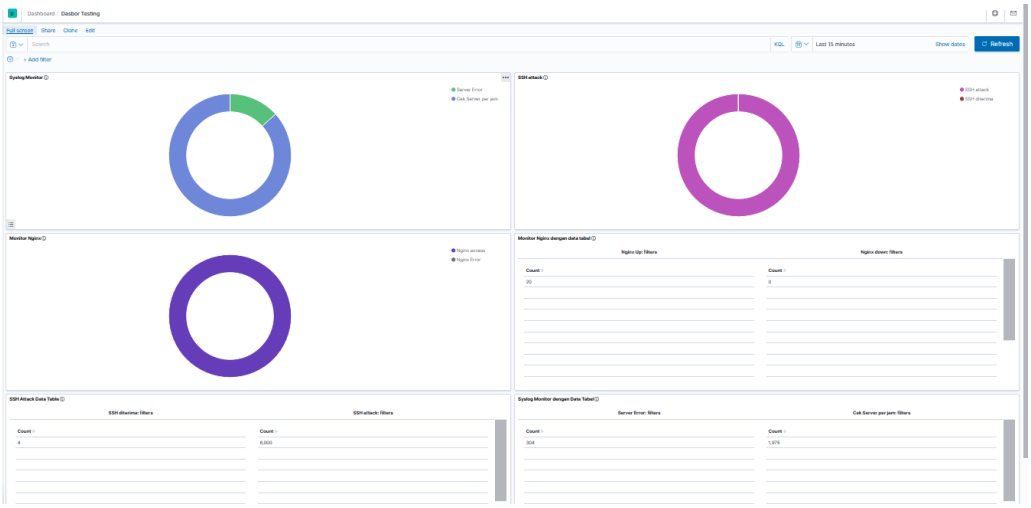
Gambar 3.11 Visualisasi *Nginx* berbentuk *pie chart*



Gambar 3.12 Visualisasi *Nginx* berbentuk *data table*

Terlihat pada **Gambar 3.11** dan **Gambar 3.12** visualiasi *log* Nginx ini bertujuan untuk memonitor *up* dan *down* sebuah *website* yang ada. Filter yang digunakan *Nginx up* adalah untuk mengecek keadaan server saat *up* dan filter *nginx down* untuk mengecek keadaan server saat *down*. Namun dalam proses monitoring ini hingga saat ini belum menunjukkan *website* tersebut dalam kondisi *down/nginx down.* Visualisasi ini juga dibuat dengan dua model yaitu *pie chart* dan *date table*.

Selesai melakukan konfigurasi terhadap ketiga komponen *log*, hal yang selanjutnya dilakukan adalah membuat dasbor sistem pencatatan *log* agar mempermudah proses *monitoring* yang dilakukan oleh *SysAdmin.* **Gambar 3.13** merupakan hasil dasbor sistem pencatatan *log* yang telah dibuat.



Gambar 3.13 Hasil dasbor dari beberapa visualisasi yang sudah dibuat

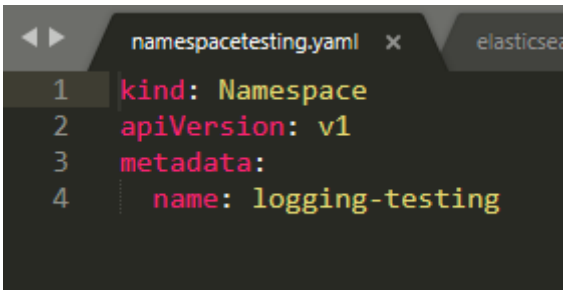
Dasbor sistem pencatatan *log* pada dasarnya memuat beberapa visuaslisasi yang telah dibuat kemudian digabungkan oleh Kibana. Sebelum menggunakan dasbor sistem pencatatan *log*, *SysAdmin* masih melakukan pengecekan *server* secara satu persatu melalui *console server*. Sesudah dasbor sistem berhasil dibuat, *SysAdmin* menggunakan dasbor sistem pencatatan *log* untuk melakukan monitoring pada server. Kesimpulan yang dapat diambil adalah pengembangan di luar Kubernetes *environtment* untuk dasbor sistem pencatatan *log* telah berhasil dijalankan.

3.3 EFK stack dan Fluentbit di dalam Kubernetes Environment

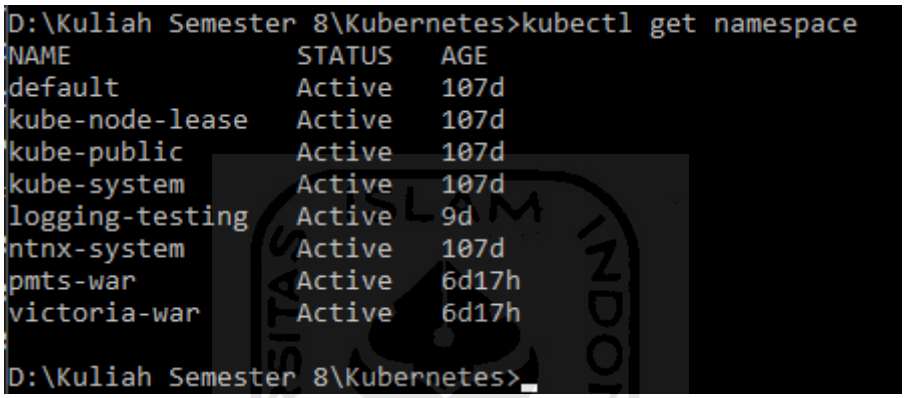
Perlu diperhatikan saat mengembangkan CLM di dalam kubernetes *environment* tidak perlu menentukan server mana yang perlu di-*monitoring* dan *log* mana saja yang perlu diambil oleh Fluentbit. Hal ini telah dilakukan secara otomatis oleh *Daemonset* yang ada di Kubernetes, Fluentd, Elasticsearch, dan Kibana juga demikian tidak perlu ditentukan di mana akan di-*install*. Karena hal tersebut sudah dilakukan oleh Kubernetes. Kita cukup membuat file konfigurasi untuk keempat Aplikasi tersebut di sebuah file yang berekstensi .yaml. Kemudian Kubernetes melakukan pembuatan *pod*, *service*, dll untuk keempat aplikasi tersebut. Arsitektur dari pengembangan kali ini juga sama dengan **Gambar 3.1**.

Perbedaan mendasar dari pengembangan di luar Kubernetes *environment* yaitu Fluentbit ter-*install* di semua Node yang telah ditentukan di dalam konfigurasi file. Namun di Kubernetes *environment* cara meng-*install* tidak dimulai dari Fluentbit dahulu melainkan dimulai dari Elasticsearch, Kibana, Fluentd, dan yang terakhir Fluentbit. Sebelum memulai konfigurasi dibuat terlebih dahulu *namespace* yang bertujuan untuk mengisolasi keempat komponen tadi dan sebagai ujicoba sebelum memasuki tahap *deployment*. **Gambar 3.14** merupakan contoh file konfigurasi

dari *namespace* yang telah dibuat dan **Gambar 3.15** merupakan hasil *namespace* yang telah masuk di Kubernetes.



Gambar 3.14 *Namespace* di Kubernetes



Gambar 3.15 *Namespace* telah berada di Kubernetes

3.3.1. Konfigurasi Elasticsearch

Konfigurasi selanjutnya setelah membuat *namespace* yaitu Elasticsearch. Ada dua langkah dalam menjalankan Elasticsearch di Kubernetes. Pertama membuat file konfigurasi *service* Kubernertes seperti **Gambar 3.16** berfungsi sebagai alat komunikasi antar *service* yang berjalan yaitu *service* Kibana dan Fluentd. Kedua membuat file konfigurasi *Statefulset* seperti **Gambar 3.17** dan **Gambar 3.18** berfungsi sebagai *image docker* yang dijalankan di dalam Kubernetes. *Image docker* inilah wadah Elasticsearch tersebut yang berfungsi untuk menyimpan data *log* yang datang dari Fluentd.

elasticsearch_service.yaml

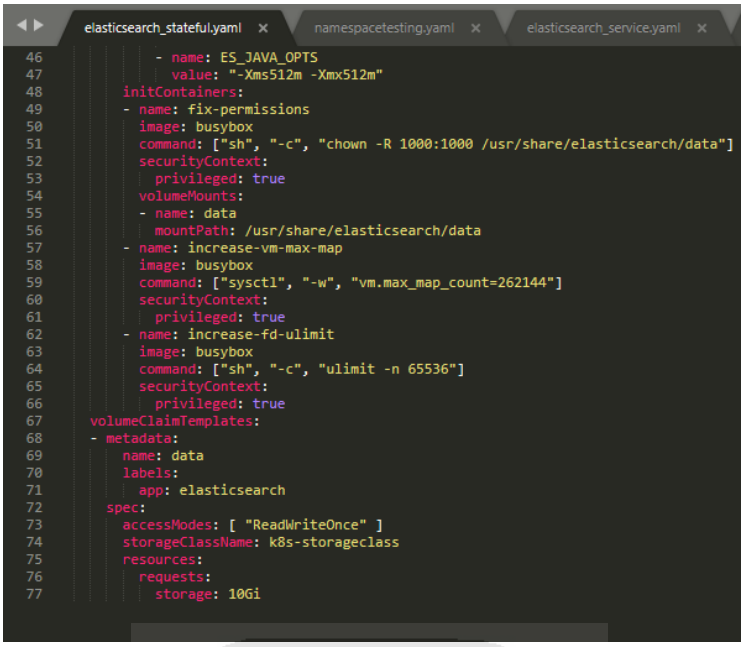
```yaml
kind: Service
apiVersion: v1
metadata:
  name: elasticsearch
  namespace: logging-testing
  labels:
    app: elasticsearch
spec:
  selector:
    app: elasticsearch
  clusterIP: None
  ports:
    - port: 9200
      name: rest
    - port: 9300
      name: inter-node
    - port: 5601
      name: kibana
```

Gambar 3.16 *Service* untuk Elasticsearch

elasticsearch_stateful.yaml

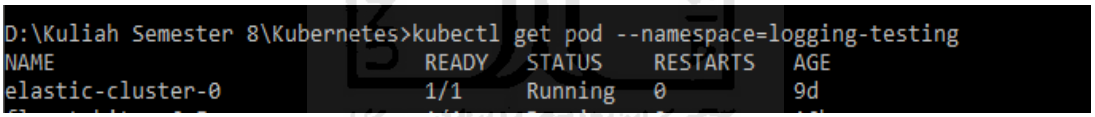
```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: elastic-cluster
  namespace: logging-testing
spec:
  serviceName: elasticsearch
  replicas: 1
  selector:
    matchLabels:
      app: elasticsearch
  template:
    metadata:
      labels:
        app: elasticsearch
    spec:
      containers:
      - name: elasticsearchtesting
        image: docker.elastic.co/elasticsearch/elasticsearch:7.2.0
        resources:
            limits:
              cpu: 1000m
            requests:
              cpu: 100m
        ports:
        - containerPort: 9200
          name: rest
          protocol: TCP
        - containerPort: 9300
          name: inter-node
          protocol: TCP
        volumeMounts:
        - name: data
          mountPath: /usr/share/elasticsearch/data
        env:
          - name: cluster.name
            value: k8s-testing-logs
          - name: node.name
            valueFrom:
              fieldRef:
                fieldPath: metadata.name
          - name: discovery.seed_hosts
            value: "elastic-cluster-0.elasticsearchtesting"
          - name: cluster.initial_master_nodes
            value: "elastic-cluster-0"
```

Gambar 3.17 *Image docker* untuk *Elasticsearch* bagian satu



Gambar 3.18 *Image docker* untuk Elasticsearch bagian dua

Sesudah membuat konfigurasi file seperti **Gambar 3.16**, **Gambar 3.17**, dan **Gambar 3.18**. dilanjutkan dengan menjalankan perintah *kubectl create -f namafilekonfigurasi.yaml*. Tunggu hingga muncul informasi bahwa *pod* telah dibuat dan masuk di dalam Kubernetes. Hasil dari *pod* tersebut akan terlihat dengan menjalankan perintah *kubectl get pod –namespace=logging-testing* seperti **Gambar 3.19** berikut:



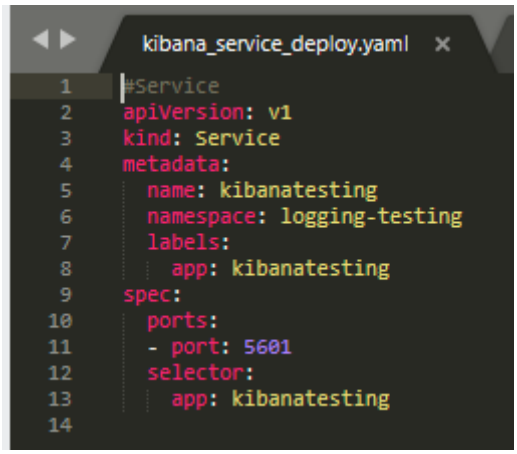
Gambar 3.19 *Pod* Elasticsearch yang telah berjalan di Kubernetes

Elasticsearch telah ter-*install* di dalam Kubernetes *environment* dan siap digunakan untuk menerima data *log* dari Fluentd. Elasticsearch yang digunakan dalan pengembangan ini hanya satu *pod* saja dikarenakan masih dalam tahap ujicoba. Namun, bila sudah masuk *deployment* disarankan untuk menambah jumlah *pod* sebagai *backup* dengan menambah beberapa baris konfigurasi di file konfigurasi Elasticsearch.

3.3.2. Konfigurasi Kibana

Saat Elasticsearch dipastikan sudah berjalan pada Kubernetes langkah selanjutnya adalah menjalankan Kibana. Seperti Elasticsearch ada 2 hal yang perlu dilakukan sebelum menjalankan Kibana pada Kubernetes. Pertama membuat konfigurasi file Kibana sebagai *service* seperti **Gambar 3.20** untuk melakukan komunikasi dengan *service* Elasticsearch dalam mengambil data

log untuk dibuat visualisasi. Kedua adalah membuat konfigurasi *image docker* seperti **Gambar 3.21** sebagai wadah Kibana tersebut di dalam Kubernetes.



Gambar 3.20 *Service* untuk Kibana

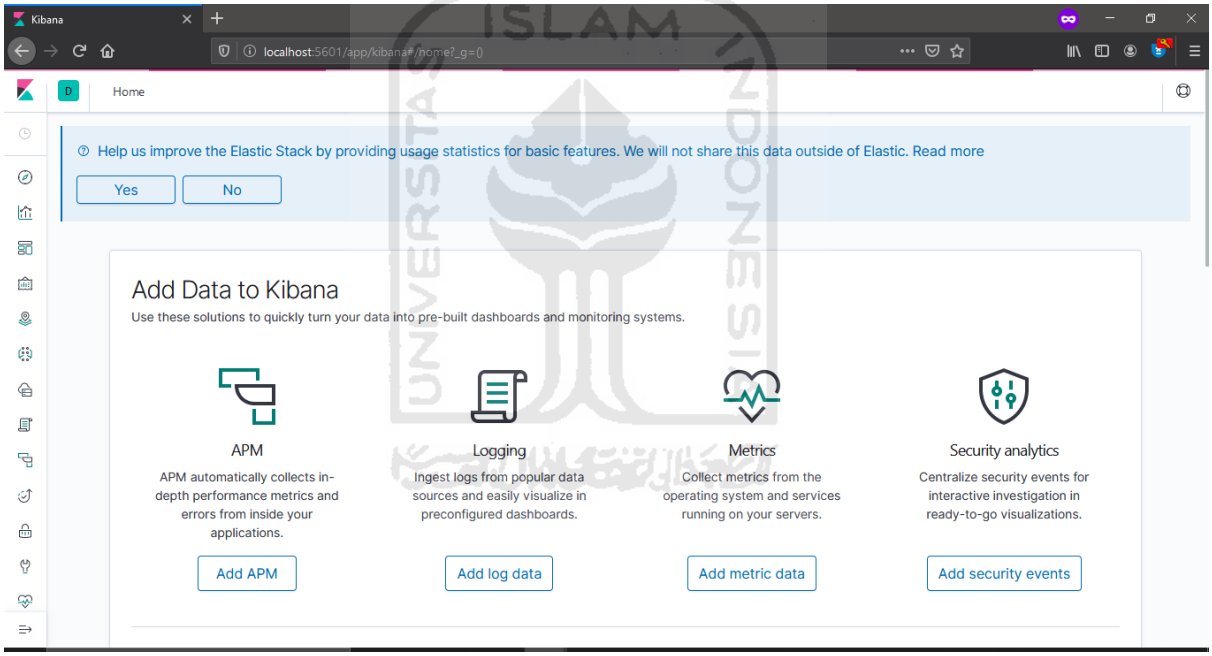
Gambar 3.21 *Image docker* untuk Kibana

Ketika sudah memastikan file konfigurasi seperti **Gambar 3.20** dan **Gambar 3.21**. Langkah berikutnya sama seperti Elasticsearch yaitu menjalankan perintah *kubectl create -f namafilekonfigurasi.yaml*. untuk men-*deploy* konfigurasi file tersebut ke Kubernetes. Kemudian cek kembali dengan perintah *kubectl get pod –namespace=logging-testing.* **Gambar 3.22** adalah hasil Kibana yang telah masuk dan berjalan di dalam Kubernetes.

```
D:\Kuliah Semester 8\Kubernetes>kubectl get pod --namespace=logging-testing
NAME                              READY   STATUS    RESTARTS   AGE
elastic-cluster-0                 1/1     Running   0          9d
fluent-bit-mm6v5                  1/1     Running   0          17h
fluent-bit-nx84g                  1/1     Running   0          17h
fluent-bit-rg4l5                  1/1     Running   0          15h
fluent-bit-tn8wt                  1/1     Running   0          17h
fluentd-5b7848cb4c-fhd6p          1/1     Running   0          21h
kibanatesting-699d7cfbd8-k66hf    1/1     Running   0          9d
```

Gambar 3.22 *Pod* Kibana yang telah masuk di Kubernetes

Kibana telah ter-*install* di dalam Kubernetes *environment* dan siap digunakan untuk visualisasi data *log*. Untuk mengecek Kibana dapat digunakan jalankan perintah *kubectl port-forward kibanatesting-699d7cfbd8-k66hf 5601:5601 --namespace=logging-testing*. Kemudian cek dengan mengetikkan *localhost:5601* di *browser* dan tunggu hingga Kibana muncul sepeti **Gambar 3.23**. Setelah muncul maka Kibana siap digunakan untuk melakukan visualisasi data *log*.



Gambar 3.23 Tampilan awal Kibana

3.3.3. Konfigurasi Fluentd

Pastikan Elasticsearch dan Kibana telah berjalan di dalam Kubernetes. Tahap selanjutnya adalah menjalankan Fluentd sebagai aggregator yang berfungsi menerima *log-log* yang datang dari Fluentbit. File konfigurasi Fluentd kali ini sudah mencakup konfigurasi *service, image docker, dan configmap* yang akan berjalan di Kubernetes. **Gambar 3.24**, **Gambar 3.25**, dan **Gambar 3.26** merupakan hasil dari konfigurasi file Fluentd.

```yaml
#### Fluentd Kubernetes Service
apiVersion: v1
kind: Service
metadata:
  name: fluentd
  namespace: logging-testing
spec:
  ports:
  - port: 24284
    protocol: TCP
  selector:
    app: fluentd
  type: ClusterIP
```

Gambar 3.24 *Service* untuk Fluentd

```yaml
#### Fluentd Kubernetes Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: fluentd
  namespace: logging-testing
  labels:
    app: fluentd
spec:
  selector:
    matchLabels:
      app: fluentd
  template:
    metadata:
      labels:
        app: fluentd
    spec:
      containers:
      - name: fluentd-testing
        image: fluent/fluentd-kubernetes-daemonset:v1-debian-elasticsearch
        env:
          - name:  FLUENT_ELASTICSEARCH_HOST
            value: elastic-cluster-0.logging-testing
          - name:  FLUENT_ELASTICSEARCH_PORT
            value: "9200"
        volumeMounts:
        - name: config-volume
          mountPath: /fluentd/etc
      volumes:
      - name: config-volume
        configMap:
          name: fluentd
---
#### Fluentd Kubernetes Service
```

Gambar 3.25 *Image docker* untuk Fluentd

fluentd.yaml | kibana_deploy.yaml | kibana_service_depl

```yaml
#### Custom configuration for Fluentd
kind: ConfigMap
apiVersion: v1
metadata:
  name: fluentd
  namespace: logging-testing
data:
  fluent.conf: |-
    <source>
      @type forward
      bind 0.0.0.0
      port 24284
    </source>

    <match *>
        @type elasticsearch
        host elasticsearch.logging-testing.svc.cluster.local
        port 9200
        flush_interval 10s
        logstash_format true
    </match>
---
```

Gambar 3.26 *ConfigMap* untuk Fluentd

Fungsi dari *configmap* Pada **Gambar 3.26** merupakan *behavior*/aksi yang dilakukan oleh Fluentd. Penjelasan lebih lanjut dari *configmap* tersebut ialah sebagai berikut:

<***source***> berfungsi untuk menerima data *log* yang telah dikirimkan oleh Fluentbit. ***@type*** berjenis ***forward*** sebagai *plugins* untuk sarana komunikasi antara Fluentbit dan Fluentd. Sedangkan ***bind 0.0.0.0*** bertujuan untuk mendengarkan semua IP address dari Fluentbit. Terakhir ***port 24284*** adalah port yang dibuka untuk menerima *stream log* dari berbagai IP address yang telah di *bind* tersebut.

<***match***> berfungsi untuk melanjutkan data *log* yang diterima ke tujuan penyimpanan. Tujuan tersebut adalah Elasticsearch dengan menggunakan perintah ***@type Elasticsearch*** mewakili *plugins* Elasticsearch tersebut. Nama ***host*** tersebut adalah nama *service* dari Elasticsearch sebagai tujuan dari data *log* disimpan. ***Port*** merupakan *port* Elasticsearch untuk menerima data *log*. ***Flush_interval*** berguna sebagai jarak setiap berapa detik sekali data *log* akan dikirimkan ke Elasticsearch. Terakhir ***logstash_format*** untuk mengubah data *log* menjadi format *logstash* agar dapat terbaca oleh Elasticsearch dan Kibana yang digunakan dalam visualisasi.

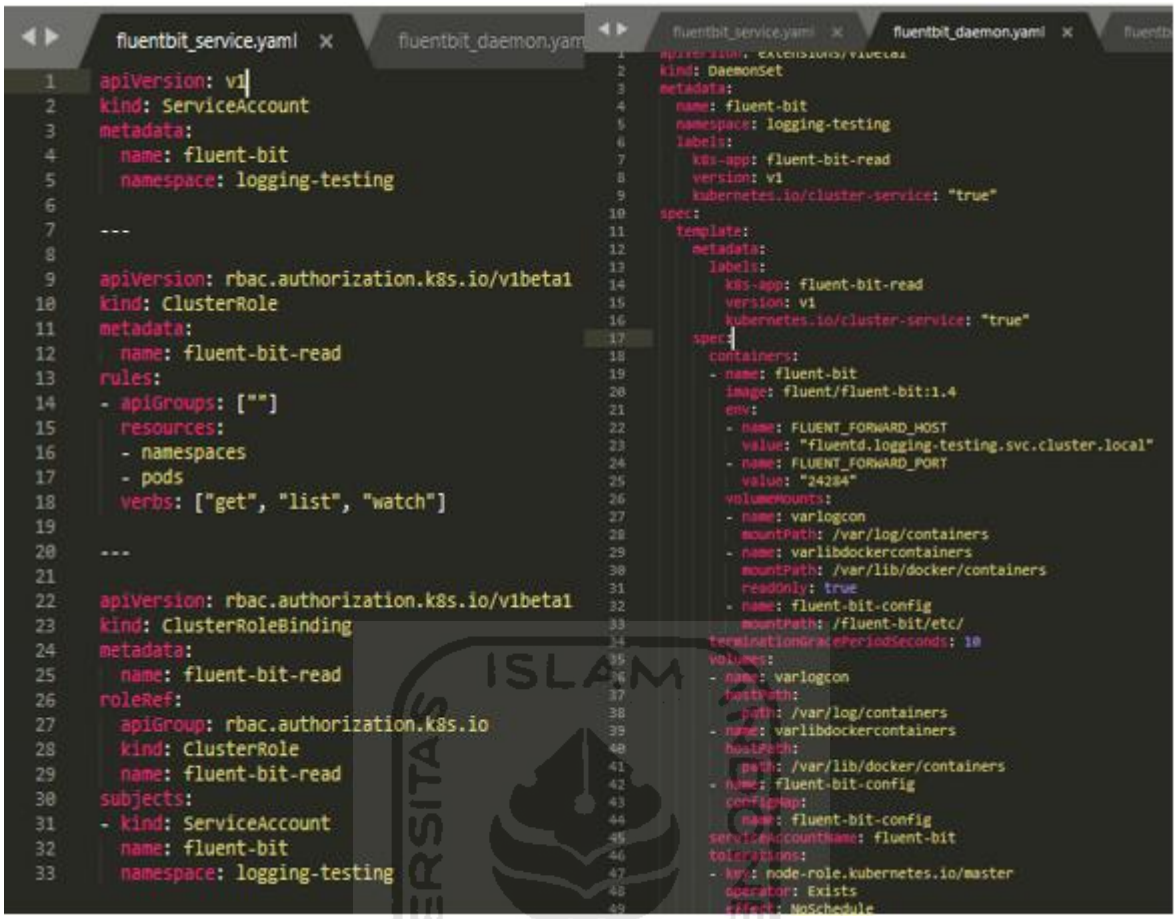
Cek Fluentd yang berjalan di Kubernetes dengan mengetikkan perintah *kubectl get pod –namespace=logging-testing* seperti **Gambar 3.27**. Fluentd sudah berjalan dan siap untuk menerima data *log* dari Fluentbit.

```
D:\Kuliah Semester 8\Kubernetes>kubectl get pod --namespace=logging-testing
NAME                                READY   STATUS    RESTARTS   AGE
elastic-cluster-0                   1/1     Running   0          9d
fluent-bit-mm6v5                    1/1     Running   0          19h
fluent-bit-nx84g                    1/1     Running   0          18h
fluent-bit-rg4l5                    1/1     Running   0          17h
fluent-bit-tn8wt                    1/1     Running   0          19h
fluentd-5b7848cb4c-fhd6p            1/1     Running   0          23h
kibanatesting-609d7cfbd8-k66hf      1/1     Running   0          9d
```

Gambar 3.27 *Pod* Fluentd yang telah masuk di Kubernetes

3.3.4. Konfigurasi Fluentbit

File konfigurasi terakhir yang dilakukan adalah Fluentbit. Fluentbit di sini digunakan sebagai *log collector* yang secara otomatis terbuat sesuai dengan jumlah *Node* di Kubernetes. Ada 3 file konfigurasi yang harus dimasukkan ke dalam Kubernetes. Pertama adalah file konfigurasi *service* seperti **Gambar 3.28** berfungsi untuk berkomunikasi dengan Fluentd. Kedua adalah file konfigurasi *daemonset* seperti **Gambar 3.28** berfungsi sebagai *image docker* Fluentbit dan bertugas untuk men-*duplicate* Fluentbit bila ada penambahan *Node*. Terakhir adalah file konfigurasi *configmap* seperti **Gambar 3.29** berfungsi sebagai *behavior* yang dilakukan Fluentbit.



Gambar 3.28 *Service* untuk Fluentbit dan *Image docker* untuk Fluentbit

Gambar 3.29 *ConfigMap* untuk Fluentbit

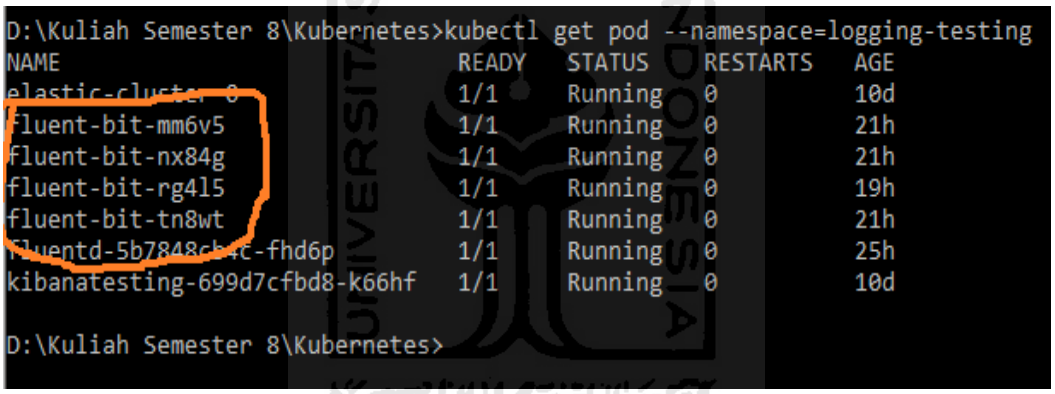
Berikut adalah penjelasan dari **Gambar 3.29** *configmap* file konfigurasi Fluentbit tersebut:

[***Service***] berguna sebagai keseluruhan fungsi dari konfigurasi Fluentbit tersebut. ***Flush*** berfungsi untuk setiap sekian detik maka data *log* yang di-*monitoring* dikirimkan ke Fluentd. ***Log_Level*** berfungsi sebagai jenis *log* yang dikirimkan oleh Fluentbit. ***Daemon*** menunjukkan apakah Fluentbit harus berjalan di latar belakang atau tidak. ***Parser_File*** digunakan untuk mengubah struktur *log* yang tidak beraturan menjadi beraturan.

[***Input***] berguna untuk menunjukkan data *log* mana saja yang akan di-*monitoring*. ***Name tail*** bertujuan agar Fluentbit melakukan proses *tailing* pada file *log*. ***Tag*** adalah pasangan dari ***Match*** yang ada di [***Output***] dan digunakan sebagai penanda dari [***Input***]. ***Path*** adalah lokasi di mana file *log* yang akan di-*monitoring*. ***Parser*** digunakan untuk merubah struktur *log* tersebut. ***DB*** adalah database *log* tersebut. ***Mem_Buf_Limit*** adalah batasan *memory* yang dipakai Fluentbit dalam mengambil data *log*. ***Skip_Long_Lines*** digunakan untuk mengabaikan file *log* yang terlalu panjang. ***Refresh_Interval*** digunakan untuk mengecek setiap sekian detik apakah ada *log* baru.

[***Output***] digunakan sebagai tujuan kemana *log* akan dikirimkan. ***Name forward*** merupakan *plugins* dari Fluentd dan Fluentbit untuk saling berkomunikasi dan sebagai tanda Fluentbit maupun Fluentd akan mengirimkan data *log* tersebut ke Fluend maupun Fluentbit. ***Match*** adalah pasangan dari ***Tag*** pada [***Input***] dan digunakan untuk mengambil *log* apa saja yang ada di ***Tag*** [***Input***]. ***Host*** adalah tujuan kemana data *log* yang telah diambil tersebut dikirimkan dan dalam kasus ini nama *service* Fluentd. ***Port*** adalah *port* yang dibuka oleh Fluentd untuk menerima data *log*. ***Retry_limit*** digunakan untuk mengulangi mengirimkan data *log*. ***Time_as_Integer*** digunakan untuk men-*set timestamp* yang ada menjadi integer.

Selesai ketiga file konfigurasi tersebut dijalankan kemudian cek dengan perintah seperti sebelumnya yaitu *kubectl get pod –namespace=logging-testing*. Bila sudah dan hasil dari perintah tersebut sama dengan **Gambar 3.30** artinya Fluentbit sudah siap digunakan. Seterusnya Fluentbit sudah dapat berjalan secara otomatis termasuk menambah *pod* Fluentbit baru saat munculnya *Node* baru yang ada di Kubernetes sehingga tinggal memonitoring data *log* lewat Kibana.



Gambar 3.30 *Pod* Fluentbit yang telah masuk di Kubernetes

Namun untuk saat ini terdapat masalah yang timbul yaitu *log-log* yang dikirimkan Fluentbit dan diteruskan oleh Fluentd ke Elasticsearch belum dapat dibuat visualisasinya oleh Kibana. Hal ini terjadi karena masih sulitnya untuk memahami *log* Kubernetes yang masuk. Karena pada *log-log* tersebut masih terdapat kesalahan dalam pengambilan *log* dan tidak sesuai dengan konfigurasi yang telah dibuat. Hal yang tak kalah penting adalah *log* tidak ter-*update* secara berkala. Hanya sekali *flush* saja berbeda halnya dengan *log* yang di luar Kubernetes *environment* selalu meng-*update log* secara berkala. Jadi, kesimpulan yang dapat diambil dari pengembangan dasbor sistem pencatatan *log* di dalam Kubernetes *environtment* adalah dasbor sistem belum dapat ditampilkan dan pengembangan ini masih perlu dilakukan riset lagi.

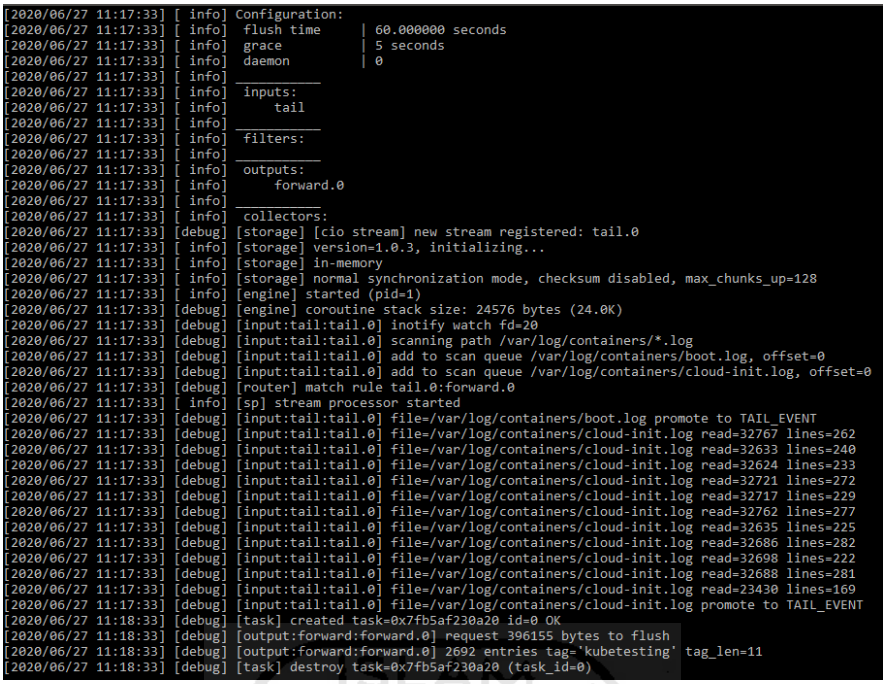
BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Hasil Pemanfaatn EFK Stack dan Fluentbit

Proses pengembangan dan uji coba dasbor sistem pencatatan *log* menggunakan EFK stack dan Fluenbit telah selesai. Pada pengembangan tersebut baik di dalam maupun di luar Kubernetes *environment* memberikan banyak informasi dan manfaat antara lain:

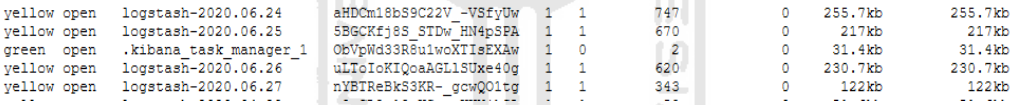
- Fluentbit dapat menangani ratusan log yang dikirimkan ke tujuan dengan waktu yang telah kita tentukan (*flush* pada konfigurasi file). Dalam satu konfigurasi file dapat ditentukan lebih dari satu *log* yang akan diambil. Tujuan pengiriman *log* pun tidak terbatas hanya satu tujuan saja. Fluentbit dapat melakukan duplikasi diri saat terdapat *Node* baru di Kubernetes.
- Fluentd yang merupakan versi lebih tinggi dari Fluentbit mampu menangani ratusan *log* yang datang. Fluentd dapat mendistribusikan *log* tersebut ke berbagai lokasi tujuan (tidak terbatasi hanya 1 tujuan) sesuai dengan konfigurasi file. Kemampuan *filter* Fluentd juga dapat diandalkan ketika ingin melakukan beberapa modifikasi pada *log* yang tiba sebelum mengirimkannya ke lokasi tujuan.
- Elasticsearch sebagai tempat *storage* yang handal karena dapat menampung ratusan *log* yang masuk baik dari *log collector(Fluentbit)* maupun *log aggregator(Fluentd).* Dalam Kubernetes *environment* dapat di *deploy* lebih dari satu *pod* sebagai *back-up* saat terjadi *error* pada salah satu *pod*.
- Kibana sebagai visualisasi dasbor sistem pencatatan *log* dapat membaca ribuan *log* dari Elasticsearch. Kibana memiliki berbagai macam jenis visualisasi yang tersedia. Selain visualisasi, Kibana juga menyediakan fitur dasbor yang menggabungkan beberapa visualisasi tersebut sehingga dapat di-*monitoring* secara bersamaan.

Fluentbit tidak hanya melakukan *tailing* pada *log*, tetapi juga bisa mengambil sebuah file yang berisi informasi. Proses pengambilan informasi pada file ini cukup dengan mengganti ***PATH*** di file konfigurasi Fluentbit dengan nama file yang akan diambil. Proses *tailing* yang dilakukan Fluentbit dapat dilihat pada **Gambar 4.1**:



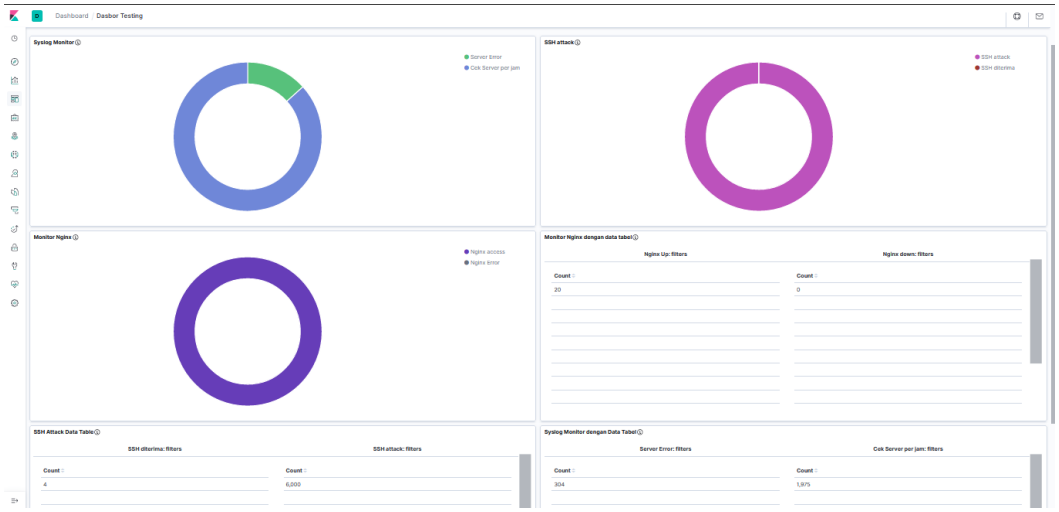
Gambar 4.1 Proses *Tailing log* yang dilakukan Fluentbit

Elasticsearch menerima ratusan *log* yang dikirimkan oleh Fluentd setiap hari. *Log* ini tersimpan di Elasticsearch dan menunggu untuk dilakukan visualisasi oleh Kibana. Pada **Gambar 4.2** menunjukkan *log* yang tersimpan di Elasticsearch.

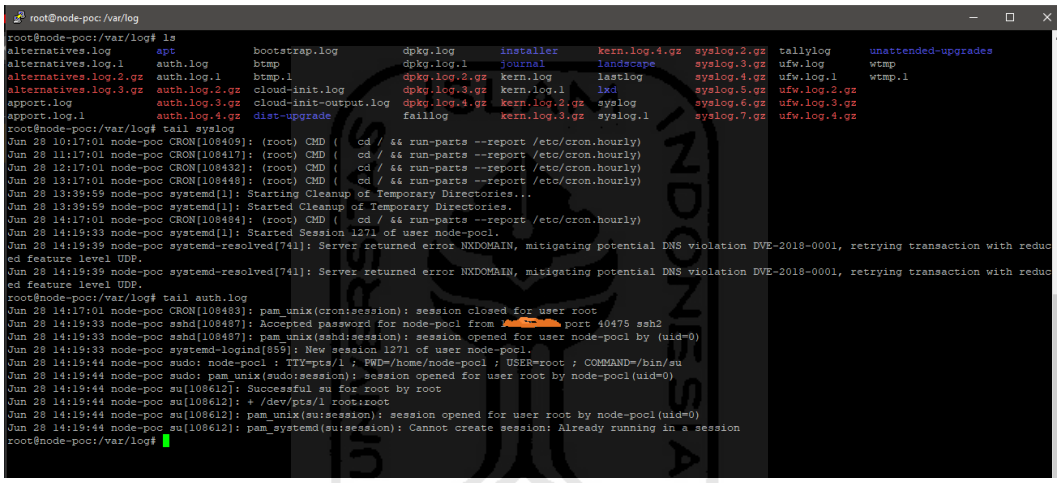


Gambar 4.2 *Log* yang tersimpan di Elasticsearch

Selain manfaat dan informasi yang telah disebutkan sebelumnya, hasil dari dasbor sistem pencatatan *log* tentunya sangat membantu kinerja SysAdmin. Hal ini terbukti dengan adanya dasbor sistem pencatatan *log* seperti pada **Gambar 4.3** membantu SysAdmin untuk melihat *log* dengan lebih mudah dan jelas daripada harus mengecek *log* tersebut satu per satu di dalam *server* seperti pada **Gambar 4.4**.



Gambar 4.3 Gambaran dasbor saat *Sysadmin* melakukan *monitoring*



Gambar 4.4 Cek *Log* secara manual

Selama masa pengembangan dan uji coba dasbor sistem pencatatan *log* menggunakan EFK stack dan Fluentbit, banyak memberikan penemuan-penemuan baru. Terdapat sebuah kejelasan berupa kelebihan dan kekurangan pada implementasi EFK stack dan Fluentbit baik di dalam maupun di luar Kubernetes *environment*.

4.1.1 Kelebihan dan Kekuragan di luar Kubenertes

Pengembangan dan ujicoba yang dilakukan pada dasbor sistem pencatatan *log* menggunakan EFK stack dan Fluentbit di luar Kubernetes *environment* terdapat kelebihan dan kekurangan, antara lain:

- Kelebihannya adalah *Log* yang di-*monitoring* terlihat jelas letak dan kegunaan dari *log* tersebut. Server yang di-*monitoring* teridentifikasi dengan jelas. Proses visualisasi yang mudah karena *log* yang terbaca sudah di identifikasi (cth: *log syslog*, *log ssh*, *log nginx*, dsb).

- Kekurangannya adalah proses instalasi semua komponen EFK stack dan Fluentbit dilakukan secara manual. Bila *server* yang di-instalasi Fluentbit mengalami *down/error* maka harus diaktifkan secara manual. Saat menambah *server* baru, Fluentbit harus di-*install* lagi di *server* baru tesebut.

4.1.2 Kelebihan dan Kekurangan di dalam Kubernetes

Pengembangan dan ujicoba yang dilakukan pada dasbor sistem pencatatan *log* menggunakan EFK stack dan Fluentbit di dalam Kubernetes *environment* terdapat kelebihan dan kekurangan, antara lain:

- Kelebihannya adalah tidak perlu menginstall komponen EFK stack dan Fluentbit secara satu persatu karena sudah otomatis dibuat oleh Kubernetes melalui konfigurasi file berekstensi *yaml*. Apabila terdapat *pods* yang *down* Kubernetes akan me-*restart pod* tersebut secara otomatis. Jika terdapat penambahan *Node,* Fluentbit secara otomatis akan menduplikat diri untuk memonitor keseluruhan *Node* baru tersebut.
- Kekurangannya adalah *logs* yang diambil masih belum dapat teridentifikasi milik *log* mana (cth: apakah *log syslog* atau yang lainnya). Proses visualisasi yang rumit dikarenakan *log* yang terbaca masuk kategori *log* universal di Kubernetes. Identifikasi Path *log* yang sulit di dalam *Node* Kubernetes tersebut.

4.2 Perbandingan Solusi dengan Alternatif lainnya

Selain menggunakan teknologi Fluentbit dan Fluentd dalam pengembangan sistem dasbor pencatatan *log* dapat digunakan teknologi lainnya. Adapun Elasticsearch dan Kibana belum ditemukan pengganti yang dapat digunakan dalam mengembangkan dasbor sistem pencatatan *log* tersebut. Solusi dan alternatif lain tersebut adalah sebagai berikut:

- Mengganti Fluentd menjadi Logstash sebagai aggregator.
- Mengganti Fluentbit menjadi Filebeats yang merupakan turunan Logstash sebagai *log collector.*
- Menghilangkan Fluentbit dan menjadikan Fluentd sebagai *log collector* dan *log aggregator.*
- Menghilangkan Fluentd dan menjadikan Fluentbit sebagai *log collector* saja.

4.3 Hasil dari Magang pada Kualitas Diri

Setelah menjalani magang selama lebih dari 6 bulan dan mengerjakan berbagai pekerjaan yang diberikan dapat diambil refleksi pada kualitas pribadi yang dijabarkan dalam poin-poin berikut:

- **Wawasan mengenai dunia *Security***. Selama magang yang dilakukan 6 bulan lebih pada tim *Site Realibility Engineer* (SRE), banyak hal yang dapat dipelajari dari sisi *Security* dalam sebuah infrastruktur sistem informasi, jaringan dan banyak lainnya terkait dunia IT. Pekerjaan yang diberikan selama magang tidak lepas dari keterkaitannya dengan *Security* sehingga membutuhkan waktu ekstra untuk mempelajari dan menyesuaikan dengan keadaan yang ada.
- **Belajar berkomunikasi dengan baik**. Tentunya selama magang dilakukan komunikasi dengan pihak-pihak pendukung dan instasi terkait, termasuk menjaga hubungan dengan rekan kerja satu tim dan ilmu baru yang didapatkan melalui mereka. Belajar untuk berani berdiskusi dan bertanya saat sedang mengadakan rapat mingguan yang dilaksanakan oleh tim SRE. Hal tersebut sangat membantu dalam pengembangan kepribadian untuk menjadi lebih baik.
- **Mengetahui keadaan dunia kerja**. Sebagai mahasiswa yang belum pernah menginjakkan kaki di dalam dunia kerja kesempatan untuk magang ini merupakan ajang untuk mengetahui hal apa saja yang terjadi di dunia kerja. Mulai dari regulasi organisasi yang perlu dipatuhi, bekerja secara tim, hingga berhadapan dengan berbagai macam orang yang memiliki kepribadian yang berbeda-beda. Hal ini merupakan sebuah pengalaman sekaligus pelajaran yang sangat bermanfaat.
- **Manajemen waktu terhadap pekerjaan**. Selama magang tentunya mendapatkan beban pekerjaan dan juga tidak ketinggalan kewajiban sebagai mahasiswa yang wajib diselesaikan. Pada kondisi itu manajemen waktu sangat berharga dan harus dimanfaatkan sebaik-baiknya. Supaya waktu tidak terbuang sia-sia dengan menelantarkan pekerjaan dan kewajiban yang ada, setiap 1 minggu sekali dilakukan rapat untuk menentukan pekerjaan selama 1 atau 2 minggu kedepannya. Pada akhir minggu tersebut dilakukan *review* agar kinerja kedepannya lebih baik.
- **Meningkatkan kemampuan diri**. Setelah melalui berbagai macam pekerjaan yang ada di tempat magang, tidak lengkap rasanya bila tidak ada perubahan pada diri dan kemampuan yang dimiliki. Tentunya selama magang, banyak hal yang telah dipelajari untuk meningkatkan kualitas diri agar menjadi pribadi yang lebih baik dari sebelum dimulainya

magang. Kemampuan yang terus diasah setiap hari dan pemaksaan dalam mempelajari hal baru dengan cepat telah meningkatkan kualitas pada bagian pengetahuan dan kemampuan yang dimiliki.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pengembangan dasbor sistem pencatatan *log / Centralized Log Management* menggunakan teknologi EFK *stack* dan Fluentbit untuk membantu kinerja *SysAdmin* sudah berhasil. Kesimpulan yang dapat diambil antara lain:

1. Semua *virtual machine server* telah ter-*install* Fluentbit sehingga dapat dilakukan *monitoring* pada setiap *server* menggunakan dasbor sistem pencatatan *log.*
2. Dasbor sistem pencatatan *log* tersebut mampu mengurangi waktu dan meningkatkan keakuratan pencarian kesalahan *log* pada server yang dilakukan oleh *SysAdmin.*
3. Pengecekan kesalahan dan data *log* tidak perlu dilakukan secara manual menggunakan *command line* di server secara satu persatu melainkan menggunakan visualisasi dasbor yang dimiliki Kibana.
4. Telah mengimplementasikan teknologi baru di ruang lingkup BSI UII. Namun, pengembangan dan ujicoba dasbor sistem pencatatan *log* ini berhasil dengan syarat EFK *stack* dan Fluentbit berada di luar Kubernetes *environment*. Pengembangan dan ujicoba pada Kubernetes *environment* belum dikatakan berhasil karena terdapat beberapa masalah pada pengambilan *log* dan visualisasi data. Selain itu, pengembangan dasbor sistem pencatatan *log* di luar Kubernetes *environement* sebaiknya dilakukan dengan skala *server* yang sedikit dan dilakukan oleh seorang *Basic Engineer*. Berbeda dengan pengujian di dalam Kubernetes *environment* yang dapat menangani skala *server* lebih besar/ratusan *Node.* Pengembangan di dalam Kubernetes *environment* tersebut harus dilakukan oleh *Expert Engineer.*

5.2 Saran

Pengembangan dan ujicoba dasbor sistem pencatatan *log* ini dapat dikembangkan lebih lanjut untuk mengoptimasi beberapa hal yang kurang pada dasbor sistem pencatatan *log* tersebut. Optimasi yang dapat dilakukan antara lain:

- Menambahkan beberapa *log* lainnya yang akan di moniroting pada pengembangan di luar Kubernetes *environment.*

- Menambahkan beberapa visualisasi lainnya sebagai penunjang dasbor pada pengembangan di luar Kubernetes *environment.*
- Melakukan riset dan ujicoba lebih lanjut terhadap pengembangan di dalam Kubernetes *environment.*

DAFTAR PUSTAKA

Arslan, M. (2016, Desember 23). *pengenalan singkat elasticsearch*. Diambil kembali dari codepolitan.com: https://www.codepolitan.com/pengenalan-singkat-elasticsearch

Communications, Y. (2019, Desember 06). *Medium*. Diambil kembali dari Medium.com: https://medium.com/@yobitelcommunications/efk-kubernetes-stack-in-cloud-native-way-scope-about-retail-industry-77c526402a27

Elastic. (2020). *what is elasticsearch*. Diambil kembali dari elastic.co: https://www.elastic.co/what-is/elasticsearch

Elastic. (2020). *what is kibana*. Diambil kembali dari elastic.co: https://www.elastic.co/what-is/kibana

Erwinsyah, Y. B. (2020). KONSOLIDASI DAN VISUALISASI LOG SERVER BSI UII MENGGUNAKAN ELK STACK. *Informatic Engineering Universitas Islam Indonesia*.

Fluend. (2020). *www.fluentd.org*. Diambil kembali dari www.fluentd.org/architecture: https://www.fluentd.org/images/fluentd-architecture.png

Fluentd. (2020). *architecture*. Diambil kembali dari fluentd.org: https://www.fluentd.org/architecture

Fluentd. (2020). *Main pages*. Diambil kembali dari fluentd.org: https://www.fluentd.org/

Gunawan, R. (2018). PENGUKURAN QUERY RESPON TIME PADA NOSQL DATABASE BERBASIS DOCUMENT STORED. *Jurnal Siliwangi Vol.4. No.2, 2018 Seri Sains dan Teknologi*, 1.

J, M. K., Dubey, S., B, B., Rao, D., & Rao, D. (2018). Data Visualization on GitHub Repository Parameters Using Elastic Search and Kibana. *International Conference on Trends in Electronics and Informatics (ICOEI)* (hal. 1-3). Tirunelveli, India: IEEE.

Lei, G., Luo, X., Cai, L., Gao, L., Dai, N., Xu, Q., & Tan, Z. (2020). Research on smart EFK algorithm for electric vehicle battery packs management system. *Research Article*, 257-262.

Morgan, J. (2016, Mei 31). *what is cenralized log management clm*. Diambil kembali dari missioncloud.com: https://www.missioncloud.com/blog/what-is-centralized-log-management-clm

Sartika, E. P. (2020). PEMANFAATAN ELASTICSEARCH LOGSTASH KIBANA UNTUK PENGELOLAAN DAN VISUALISASI DATA PORTAL PENGEMBANGAN DAN

PEMBINAAN SUMBER DAYA MANUSIA (PORTAL PPSDM). *Informatic Engineering Universitas Islam Indonesia*.

Wicaksono, H. I. (2020). VISUALISASI KUALITAS INTERNET MOBILE BROADBAND MENGGUNAKAN ELK STACK (Studi Kasus di Daerah Jambidan, Banguntapan, Bantul). *Informatic Engineering Universitas Islam Indonesia*.

Yugitama, R., Rachman, P. P., & Sulistyo. (2020). EFISIENSI MONITORING HONEYPOT DENGAN MENGGUNAKAN VISUALISASI DAN OTOMATISASI LAPORAN LOG SERANGAN . *Jurnal IT*, 1.

LAMPIRAN