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

Rio Pradana Aji
Fakultas Teknologi Industri
Universitas Islam Indonesia
Yogyakarta, Indonesia
16523120@students.uii.ac.id

Andhik Budi Cahyono
Fakultas Teknologi Industri
Universitas Islam Indonesia
Yogyakarta, Indonesia
andhikbudi@uii.ac.id

Abstract*— 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* yang berisi *log* ini tentunya membantu kinerja *SysAdmin* dalam memantau kondisi *server-server* yang dikelola BSI. Namun, banyaknya *log* di setiap *server* tersebut menyita waktu *SysAdmin* mencari *log* yang berisikan *error* terlebih bila harus dilakukan pengecekan server satu-persatu. Solusi dari masalah tersebut adalah dibuatnya sebuah *Centralized Log* yang menyimpan semua *log* yang ada agar bisa dimonitoring oleh *SysAdmin*. *Centralized Log* ini menggunakan teknologi Elasticsearch, Fluentd, Kibana (EFK) *Stack* sebagai penyimpan *log, aggregator*, visualisasi log dan juga Fluentbit sebagai *log collector*. Data 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 pengembangan dasbor sistem pencatatan *log* ini adalah *SysAdmin* dapat menemukan *log* di *server* dengan cepat dan tepat serta mengurangi waktu perbaikan jika ditemukan kesalahan.

Keywords—Elasticsearch, Fluentd, Kibana, Centralized log, Dasbor Sistem.

I. PENDAHULUAN

Badan Sistem Infromasi (BSI) secara umum bertugas mengawal perencanan, pengembangan, operasi, serta layanan sistem dan tekno*log*i informasi di lingkungan Universitas Islam Indonesia (UII) dengan jumlah pengguna yang mencapai kurang lebih 30.000 pengguna yang terdiri dari mahasiswa, dosen tenaga pendidikan, dan para pemangku kepentingan lainnya [1]. Layanan yang diberikan oleh BSI sendiri terus bertambah menyesuaikan kebutuhan civitas academica baik itu karyawan ataupun mahasiswa. Sebagian layanan berupa pengembangan sistem informasi atau perangkat lunak lainnya untuk mendukung kebutuhan bisnis organisasi. Semua sistem informasi baik *desktop* maupun berbasis web tersebut membutuhkan sebuah *server* untuk berjalan dan di dalam *server* tersebut terdapat sebuah *log* yang sangat membantu kinerja *SysAdmin* dalam memonitor keadaan *server* untuk dilakukan pemeliharaan *server*. Selain memonitor, tugas lain dari *SysAdmin* adalah melakukan pebaikan apabila terjadi *error* pada sebuah *server* atau layanan yang sedang berjalan. Perbaikan ini tentunya diharapkan diselesaikan dengan waktu singkat mengingat layanan tersebut digunakan oleh banyak orang baik luar maupun dalam lingkungan UII. Maka dari itu solusi untuk mempersingkat waktu yang diperlukan oleh *SysAdmin* adalah membuat sebuah dasbor *log* yang terpusat sehingga memudahkan pekerjaan *SysAdmin* untuk mencari *error log* yang ada serta tidak harus membuang waktu dengan mencari *log* satu persatu pada tiap *server* yang ada.

Dasbor *log* adalah sebuah sistem informasi yang bertujuan untuk menampilkan data *log* yang ada pada sebuah *server* [2]. Dasbor *log* juga dapat dikembangkan menjadi sebuah dasbor *log* terpusat yang menampilkan semua data *server* dan menggabungkannya pada satu titik tertentu. Dasbor *log* yang mencatat *log* secara terpusat ini sedang dibutuhkan oleh Badan Sistem Informasi (BSI) yang ada di Universitas Islam Indonesia (UII).. Sistem dasbor *log* terpusat ini menggunakan tekno*log*i Elasticsearch, Fluentd, Kibana stack (EFK stack) serta tekno*log*i tambahan yaitu Fluentbit.

Pada pekerjaan ini EFK stack digunakan untuk mengolah dan memonitor data *log* yang terdapat pada sebuah *server*/*node*. Selain EFK stack digunakan juga tekno*log*i bernama Fluentbit. Penjelasan dari 4 tekno*log*i yang digunakan adalah sebagai berikut. Fluentbit merupakan turunan dari Fluentd yang berfungsi sebagai pengumpul *log*/*log collector*. Kemudian semua *log* yang telah diambil dari berbagai *server* tersebut diteruskan ke Fluentd. Fungsi Fluentd yaitu sebagai *aggregator* untuk melakukan *filtering* data *log* [3]. Setelah data di- *filtering* kemudian Fluentd meneruskan data tersebut untuk disimpan di Elasticsearch yang berfungsi sebagai penyimpanan/*storage* dan kemudian dilakukan *indexing*. Setelah data terindeks maka tugas dari Kibana adalah memvisualisasikan data tersebut agar dapat terbaca dengan baik.

Makalah ini ditulis dengan tujuan untuk memaparkan hasil pengembangan dasbor sistem pencatatal *log server* menggunakan EFK stack dan Fluentbit , berikut adalah struktur pembahasan dari masing masing bab yang ditulis. Bab I berisikan pokok masalah yang akan diselesaikan dalam

makalah ini, Bab II berisikan teori-teori pendukung untuk menyelesaikan makalah ini, Bab III berisikan proses dalam menyelesaikan sistem ini, Bab IV berisikan hasil dan pembahasan pada sistem yang sudah dibuat, Bab V berisikan kesimpulan yang bisa diambil dalam menyelesaikan sistem tersebut dan saran yang bisa digunakan ke depannya untuk meningkatkan kinerja sistem yang telah dibuat.

II. DASAR TEORI DAN STUDI PUSTAKA

A. *ELASTICSEARCH*

Elasticsearch adalah sebuah sistem yang berfungsi sebagai *database* yang biasa digunakan dalam EFK stack dan ELK stack yang ada. Elasticsearch sendiri merupakan *database* noSQL yang berfokus pada *search engine database* yang ada dan sifatnya terbilang cukup unik karena di Elasticsearch kita bisa mengamsumsikan indeks sebagai sebuah *database*, *types* sebagai *table*, dokumen sebagai *record* atau *row*, dan *mapping* sebagai skema tabel [4].

Cara kerja dari Elasticsearch adalah setelah menerima *data flow* dari sebuah *aggregator* atau sumber yang mengirimkan data tersebut (baik berupa *log*, *metric* sistem, dan web aplikasi). Kemudian data di- *ingest* atau lebih tepatnya terjadi *Data Ingestion* di Elasticsearch yang memproses data tersebut (*parse, normalized*, dan *enriched*) sebelum kemudian dilakukan *indexing*/dimasukkan ke dalam Elasticsearch. Setelah data masuk ke dalam Elasticsearch maka *user* dapat melakukan *query* kompleks terhadap data mereka dan mengambil ringkasan data untuk ditampilkan di Kibana [5].

Adapun penelitian yang dilakukan oleh [6] 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 sehingga latensi data yang terjadi berkurang selama 2,5 detik.

B. *FLUENTD*

Fluentd adalah sebuah sistem/*service* yang berfungsi sebagai *data collector* yang menyatukan berbagai jenis *log* dari semua macam-macam *log* yang ada [7]. Sistem 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* yang memungkinkan Fluentd melakukan *filtering* data *log* yang masuk sebelum mengarahkan data tersebut ke sebuah *Storage*, dalam kasus kali ini adalah Elasticsearch.

Data yang disimpan di Fluentd berformat JSON sebisa mungkin karena *downstream* data *processing* lebih mudah dan fleksibel dengan menggunakan format JSON. Plugins yang ada di Fluentd juga terbilang cukup banyak yaitu lebih dari 500 plugin dan terus bertambah dengan seiringnya waktu dari bantuan perusahaan pengembang maupun user yang memakai Fluentd, saat ini telah digunakan lebih dari 2000 perusahaan dan di *deploy* di 50000 *server* dan terus bertambah. Selain hal tersebut Fluentd juga membutuhkan sumberdaya yang sedikit saat digunakan dan juga memiliki kelebihan *memory file* berbasis *buffering* sebagai tindakan pencegahan untuk kehilangan data [8].

C. *KIBANA*

Kibana adalah sebuah *open source* aplikasi *front end* yang berada di atas Elastic-stack dan menyediakan pencarian dan visualisasi data yang telah di *index* 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 [9].

Pada penelitian yang dilakukan oleh [10] 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 [11] yang berkebetulan bertempat di BSI juga menggunakan Kibana sebagai visualisasi data *log* yang ada sehingga memudahkan proses monitoring untuk *log server* yang ada di BSI tersebut.

D. *FLUENTBIT*

Fluentbit adalah sebuah *open source* platform yang digunakan sebagai *log processor* atau *log forwarder* untuk mengambil data *log* dari berbagai sumber yang ada kemudian mengerimkannya ke lokasi yang sudah ditentukan dan dalam kasus kali ini adalah Fluentd [12]. Lebih mudahnya Fluentbit merupakan versi mini dari Fluentd yang cocok dipakai di *environment* yang membutuhkan kapasitas terbatas.

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

E. *CENTRALIZED LOG*

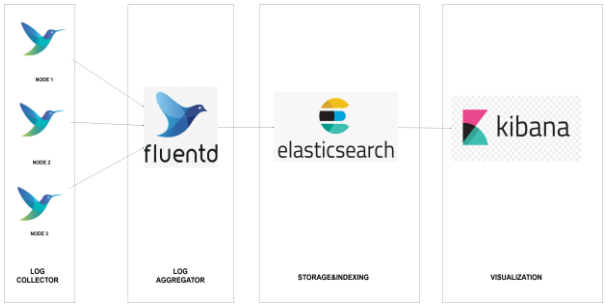
Centralized Log atau sering disebut dengan *Centralized Log Management* (CLM) adalah sebuah tipe sistem *logging* solusi yang menggabungkan semua data *log* yang dimiliki dan kemudian di arahkan menuju sebuah pusat yang mudah diakses dan memiliki UI yang mudah digunakan. *Centralized Log* dibuat untuk memudahkan pekerjaan seorang *engineer*. CLM tidak hanya menyediakan fitur bagi *engineer* untuk mengambil data saja, selain fitur tersebut CLM juga menggabungkan, menganalisis, dan memberikan sebuah gambaran informasi dengan cepat dan jelas [13].

III. METODO*LOG*I

Dalam pengerjaan makalah ini telah ditentukan langkah-langkah yang harus diambil berdasarkan kebutuhan pekerjaan yang telah didiskusikan oleh pengampu kepentingan sistem ini. Hasil dari langkah-langkah yang diambil adalah sebagai berikut : Perancangan Arsitektur Centralized *Log*, Instalasi dan konfigurasi Fluentbit, Instalasi dan konfigurasi Fluentd, Instalasi Elasticsearch, Instalasi dan konfigurasi Kibana

A. *PERANCANGAN ARSITEKTUR CENTRALIZED LOG*

Setelah melakukan diskusi dengan Tim Site Reliability Engineer (SRE) yang ada di BSI maka hasil arsitektur yang disepakati dan akan dilakukan pengembangan adalah sebagai **Gambar 3.1**.



Gambar 3.1 Arsitektur Centralized *Log*

Adapun penjelasan yang bisa dijabarkan berdasarkan arsitektur **Gambar 3.1** Arsitektur Centralized Log adalah sebagai berikut:

- Fluentbit - digunakan sebagai pengumpul *log* (*log collector*) yang ada di setiap *server* dan kemudian meneruskan semua *log* yang sudah diambil ke Fluentd.
- Fluentd - sebuah *aggregator* yang digunakan sebagai penerima *log* dari berbagai *log collector* yang ada, dalam kasus kali ini adalah Fluentbit.
- Elasticsearch - database yang digunakan untuk menyimpan data *log* yang sudah dikirimkan oleh Fluentd dan sebagai query *log* untuk Kibana.
- Kibana – adalah sebuah sistem untuk melakukan visualisasi terhadap *log* yang sudah tersimpan di dalam Elasticsearch, data *log* yang tersimpan di Elasticsearch di-query oleh Kibana untuk kemudian ditampilkan dengan bentuk chart.

B. *KONFIGURASI FLUENT BIT*

Sebelum melakukan konfigurasi untuk Fluentbit sebagai *log collector*, dilakukan terlebih dahulu sebuah installasi aplikasi pendukung pada *server* yaitu : Compiler GCC atau clang, Cmake, Flex dan Bison, Fluentbit versi terbaru

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

```
[SERVICE]
    Flush 5
    Daemon on
    Log_Level debug

#Mengambil syslog di diri sendiri
[INPUT]
    Name tail
    Tag syslog
    Path /var/log/syslog
    Refresh_Interval 60

#Mengambil ssh di diri sendiri
[INPUT]
    Name tail
    Tag ssh
    Path /var/log/auth.log
    Refresh_Interval 60

#Forward syslog ke aggregator
[OUTPUT]
    Name forward
    Match *
    Host [redacted]
    Port 8888
```

Gambar 3.2 Konfigurasi Fluentbit

Adapun penjelasan isi komponen dari konfigurasi tersebut adalah sebagai berikut :

[***Service***] berguna sebagai tingkah laku yang konfigurasi file tersebut. Penjelasn di dalam [***Service***] adalah : ***Flush*** digunakan sebagai tanda data dikirimkan oleh konfigurasi dan dalam ini diberi indikator 5 menunjukkan setiap 5 detik data akan diteruskan ke ***Output*** tujuan, ***Daemon on*** digunakan agar konfigurasi dapat berjalan di latar belakang *server*, dan terakhir ***Log_Level*** berguna untuk menunjukkan data *log* yang diambil itu bersifat apa(dalam hal ini debug mencakup *info* dan *error log*).

[***Input***] berguna sebagai informasi *log* apa yang akan diambil di sebuah server. Penjelasan di dalam [***Input]*** adalah : ***Name*** digunakan sebagai identifikasi dan bagaimana tingkah laku Fluentbit ini dan pada contoh kali ini digunakan *tail* yaitu membaca *log* terbaru secara *realtime*, ***Tag*** digunakan sebagai penanda ***Input*** agar dapat dibaca oleh ***Match*** pada ***Output***, ***Path*** merupakan lokasi *log* yang akan dibaca, dan 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. Penjelasan di dalam [***Output***] adalah : ***Name*** digunakan sebagai identifikasi dan bagaimana tingkah laku Fluentbit ini dan 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* yang menerima data *log* tersebut.

C. KONFIGURASI FLUENTD

Sama seperti Fluentbit yaitu sebelum melakukan sebuah konfigurasi untuk *log aggregator*, maka dilakukan terlebih dahulu meng-install aplikasi pendukung yaitu : Ruby Gem, Fluentd, Plugins Fluentd untuk elasticsearch

Setelah meng-install tiga aplikasi pendukung di atas maka Fluentd sudah bisa digunakan sebagai *log aggregator* yang berfungsi sebagai penerima semua *log collector* yang berjalan. **Gambar 3.3** adalah contoh konfigurasi Fluentd sebagai *log aggregator.*

```
<source>
  @type forward
  port 8888
  bind 0.0.0.0
</source>


<match **>
 @type copy
 <store>
   @type elasticsearch
   host [redacted]
   port 9200
   flush_interval 10s
   logstash_format true
  </store>
</match>
```

Gambar 3.3 Konfigurasi Fluentd

Adapun isi dari komponen konfigurasi tersebut adalah sebagai berikut :

<***Source***> digunakan sebagai penerima untuk menyaring data *log* Fluentbit yang sudah *stream* ke Fluentd, digunakan ***@type forward*** sebagai *plugins* penghubung antara Fluentbit dan Fluentd yang ada, ***Port*** adalah keterangan *port* mana yang dibuka untuk menerima *log* data tersebut, dan terakhir ***bind*** adalah sebagai pengunci untuk mengikat *ip address* mana yang boleh 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 yang kemudian digunakan <***store***> untuk menyimpan data yang sudah di-*copy* tadi, di dalam <***store***> terdapat ***type*** lagi yaitu *elasticsearch* sebagai *plugin* yang digunakan untuk menyimpan data *log* tersebut 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 tersebutm dan terakhir ***logstash_format*** digunakan untuk merubah format data *log* tersebut agar dapat dibaca oleh Elasticsearch.

D. KONFIGURASI ELASTICSEARCH

Konfigurasi yang dilakukan pada sisi Elasticsearch cukup mudah karena hanya perlu melakukan installasi dan pemberian nama *cluster* saja di *server* yang akan menjadi tempat penyimpanan/*storage* data *log* ini. **Gambar 3.4** menunjukkan hasil konfigurasi Elasticsearch di *server* yang ada.

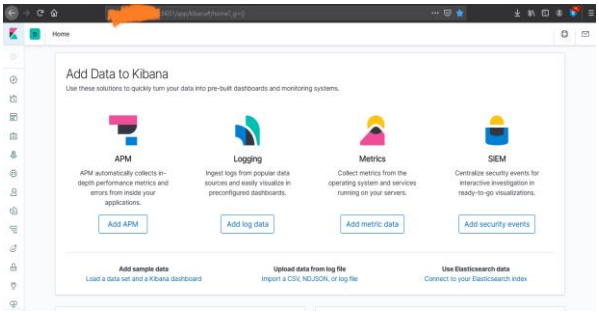

```
{
  "name" : "node-masterefk",
  "cluster_name" : "visualisasiefk",
  "cluster_uuid" : "nRFqbIFGSRe6cEjeDhfcIA",
  "version" : {
    "number" : "7.6.0",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "7f634e9f44834fbc12724506cc1da681b0c3b1e3",
    "build_date" : "2020-02-06T00:09:00.449973Z",
    "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

Adapun hasil gambar diatas merupakan Elastic Cluster yang sudah dibuat dan memberi informasi mengenai : Nama *cluster*, kapan *cluster* dibuat, versi *cluster* yang telah dibuat, dan lain sebagainya.

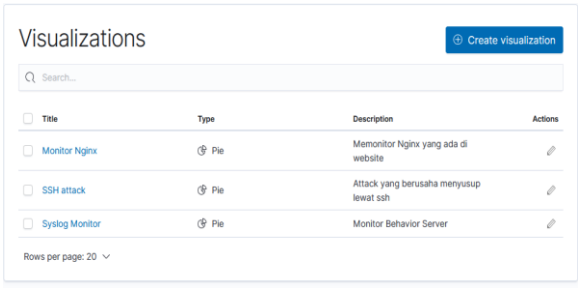
E. KONFIGURASI KIBANA

Konfigurasi di Kibana juga tidak terlalu rumit dan sama dengan konfigurasi yang telah dilakukan di Elasticsearch yaitu hanya perlu untuk meng-*install* Kibana pada *server* yang akan dijadikan tempat untuk visualisasi data *log* ini **Gambar 3.5** adalah hasil dari installasi yang dilakukan, halaman awal Kibana



Gambar 3.5 Hasil Installasi Kibana

Setelah itu dilakukan pembuatan visualisasi satu persatu untuk *log* yang telah masuk ke dalam Elasticsearch, 3 indeks ini adalah hasil visualisasi yang telah dibuat **Gambar 3.6** :



Gambar 3.6 3 Visualisasi yang telah dibuat

Hasil dan gambaran lebih jelas dari 3 visualisasi tersebut dan juga hasil dari dasbor yang telah dibuat akan disajikan pada Bab selanjutnya yaitu Bab IV Hasil dan Pembahasan.

IV. HASIL DAN PEMBAHASAN

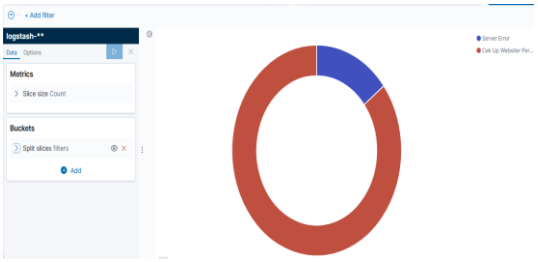
Setelah melakukan installasi dan konfigurasi yang diperlukan dalam membuat *Centralized Log* ini dapat diambil berbagai manfaat dari EFK stack dan Fluentbit ini yaitu:

Fluentbit sebagai *log collector* yang dapat mengambil berbagai *log* yang diinginkan serta dapat meneruskannya ke berbagai tujuan yang hanya membutuhkan seminimalnya satu konfigurasi saja.

Fluentd sebagai *log aggregator* yang menerima semua data *log* dari berbagai sumber kemudian memasukkan semua data tersebut ke Elasticsearch yang bertindak sebagai penyimpanan *log* juga hanya membutuhkan seminimalnya satu konfigurasi saja.

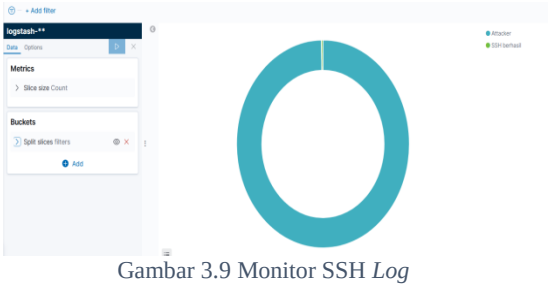
Kibana sebagai visualisasi yang tidak terbatas hanya dengan satu visualisasi saja untuk memonitor berbagai jenis *log* yang diinginkan dan dapat diskostumisasi dengan mudah dan sesuai dengan kebutuhan yang ada.

Hal di atas tidaklah mungkin dilakukan tanpa bantuan Elasticsearch sebagai tempat penyimpanan yang handal karena mampu menangangi ratusan *log* yang masuk setiap detiknya dan dapat bekerja dengan baik berbarengan dengan Kibana sebagai visualisasi data yang ada di dalam Elasticsearch tersebut. **Gambar 3.7**, **Gambar 3.8**, **Gambar 3.9**, dan **Gambar 3.10** adalah hasil visualisasi dan dasbor tersebut.

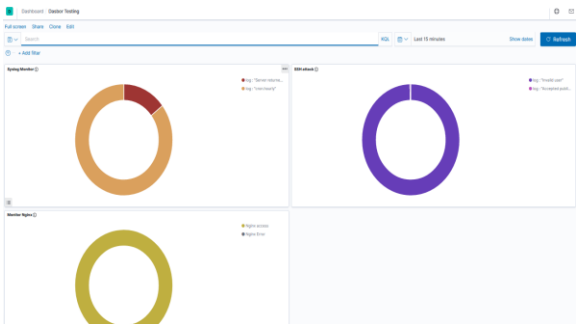


Gambar 3.7 Monitor Sys*log*

Gambar 3.8 Monitor Nginx Website



Gambar 3.9 Monitor SSH *Log*



Gambar 3.10 Dasbor Monitor 3 Visualisasi

Setelah dilakukannya penerapan EFK stack dan Fluentbit ini dapat disimpulkan pula bahwa hal ini membantu *SysAdmin* dalam meminimalisir pekerjaan yang dilakukan tanpa harus mengecek *server* satu-persatu sebagaimana telah dilakukan perbandingan hasil dari penerapan ini dapat dilihat pada **Gambar 3.11** dan **Gambar 3.12**.

Gambar 3.11 Cek *Log* secara Manual

Gambar 3.12 Cek *Log SSH Login* menggunakan Visualisasi

Hasil dari perbandingan 2 Gambar di atas menunjukkan bahwa dasbor monitor sangat membantu kinerja *SysAdmin* dalam mencari *log* yang bermasalah atau *server* yang sedang butuh dilakukan *maintenance* dengan cepat tanpa perlu membuang waktu mencari *log server* satu-persatu dikarenakan telah terdapat *log* visuaslisasi sebagai pengganti dari melihat *log* secara manual dengan masuk ke *server*.

V. KESIMPULAN DAN SARAN

A. *Kesimpulan*

Pengembangan dasbor sistem pencatatan *log* menggunakan EFK stack dan Fluentbit ini mampu membantu kinerja *SysAdmin* dalam memonitor *server* yang dikelola oleh BSI. Adanya dasbor tersebut mampu mengurangi waktu dan meningkatkan keakuratan pencarian log yang berisi kesalahan sehingga memudahkan penangan kesalahan pada server. Selain itu pengecekan kesalahan tidak perlu dilakukan secara manual, satu persatu, lewat *command line* di *server*.

B. *Saran*

Pengembangan yang dilakukan dengan EFK stack dan Fluentbit stack dalam pekerjaan dan makalah ini dapat dikembangkan lebih lanjut guna memaksimalkan penggunaan dasbor pada sistem yang telah dibuat. Beberapa saran tersebut antara lain:

- Menggunakan tekno*log*i kubernetes sehingga dalam meng-install Fluentbit dapat dilakukan dengan *daemonset* (otomatis).
- Membuat konfigurasi Fluentbit agar otomatis berjalan kembali setelah *server* berhenti mendadak/restart sendiri.
- Memaksimalkan penggunaan Fluentbit dengan mengambil beberapa *log* lain dan memaksimalkan fitur filter yang ada di Fluentd.
- Mengelompokkan *log* berdasarkan *server* yang dimiliki dalam konfigurasi Kibana.

VI. Referensi

[1] B. S. Informasi, "bsi.uii.ac.id," [Online]. Available: https://bsi.uii.ac.id/sekilas-bsi/. [Accessed 20 May 2020].

[2] Elastic, "elastic.co," [Online]. Available: https://www.elastic.co/guide/en/kibana/current/dashboard.html. [Accessed 20 05 2020].

[3] D. Berman, "*log*z.io," 25 03 2020. [Online]. Available: https://*log*z.io/b*log*/fluentd-vs-fluent-bit/. [Accessed 20 05 2020].

[4] M. Arslan, "codepolitan.com," CodePolitan, 23 Desember 2016. [Online]. Available: https://www.codepolitan.com/pengenalan-singkat-elasticsearch. [Accessed 20 Mei 2020].

[5] Elastic, "elastic.co," Elasticsearch, [Online]. Available: https://www.elastic.co/what-is/elasticsearch. [Accessed 20 Mei 2020].

[6] E. P. Sartika, "PEMANFAATAN ELASTICSEARCH *LOG*STASH KIBANA UNTUK PENGELOLAAN DAN VISUALISASI DATA PORTAL PENGEMBANGAN DAN PEMBINAAN SUMBER DAYA MANUSIA (PORTAL PPSDM)," *Informatic Engineering Universitas Islam Indonesia,* 2020.

[7] Fluentd, "fluentd.org," Cloud native Computing Foundation, [Online]. Available: https://www.fluentd.org/. [Accessed 20 Mei 2020].

[8] Fluentd, "fluentd.org," Cloud Native Computing Foundation, [Online]. Available: https://www.fluentd.org/architecture. [Accessed 20 Mei 2020].

[9] Elastic, "elastic.co," Elasticsearch, [Online]. Available: https://www.elastic.co/what-is/kibana. [Accessed 20 Mei 2020].

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

[11] Y. B. Erwinsyah, "KONSOLIDASI DAN VISUALISASI *LOG* SERVER BSI UII MENGGUNAKAN ELK STACK," *Informatic Engineering Universitas Islam Indonesia,* 2020.

[12] Fluentbit, "fluentbit.io," Treasure Data, [Online]. Available: https://fluentbit.io/. [Accessed 20 Mei 2020].

[13] J. Morgan, "missioncloud.com," Mission Cloud Services Inc, 31 Mei 2016. [Online]. Available: https://www.missioncloud.com/b*log*/what-is-centralized-*log*-management-clm. [Accessed 20 Mei 2020].