Analisis dan Peningkatan Performa *Log File* Pada *Server* dengan Elk Stack

M. Nerdi Rafli [a,1], Emil Nafan [b,2,*], Eka Praja Wiyata Mandala [c,3]

[a] Teknik Informatika, Universitas Putra Indonesia "YPTK", Padang, Indonesia
[1]mnerdiraf20@gmail.com; [2]emilnafan@upiyptk.ac.id*; [3]ekaprajawm@upiyptk.ac.id
* Penulis Korespondensi

ABSTRAK

Komputer server adalah penyedia layanan dalam jaringan dan memastikan bahwa semua aktivitas komputasi dapat dicatat dan memiliki keluaran berupa *file log*. Berdasarkan pengamatan yang dilakukan pada sebuah kantor pemerintahan di Kota Padang teridentifikasi kesulitan *mengolah file log* dan masih memiliki banyak kekurangan dalam proses menganalisa *log*. Oleh karena itu, dibutuhkan sistem pengelolaan *file log* secara lengkap, seperti *management log* yang efektif untuk menampung masalah. Sistem yang diusulkan ini dapat mengevaluasi *log* jaringan berdasarkan fungsi untuk *log management*, *log collection*, *log transformation*, dan *log file*. Saat ini telah terdapat banyak alat yang digunakan dengan bermacam metode dalam mengumpulkan *log* dan menganalisis *log* untuk mendeteksi aktivitas jahat. Alat yang tersedia belum mengimplementasikan metode ELK Stack, yaitu kombinasi dari *Elasticsearch*, *Log*stash, dan Kibana. Arsitektur ELK ini menarik untuk diimplementasikan karena dapat membangun pembaruan yang lebih tertata dengan sedikit usaha dan fungsionalitas yang didapat lebih akurat. Dengan penerapan manajemen *log* ini, seorang administrator dapat membaca *log*, meningkatkan performa *log file*, mengidentifikasi masalah pada server, dan menganalisis pelaporan secara lebih mudah.

Riwayat Artikel
Diterima 18 Januari 2024
Diperbaiki 25 Maret 2024
Diterbitkan 27 Maret 2024

Kata Kunci
Computer Security
ELK Stack
Log File

1. Pendahuluan

Komputer *server* bertindak sebagai penyedia layanan dalam jaringan dan memastikan bahwa semua aktivitas yang terkait dengannya dicatat. *Log* adalah *file* yang mencantumkan peristiwa dan waktu yang terjadi di *server*. Memantau kinerja *server* dengan *log* akan meningkatkan daya tahan *server* dan tingkat layanan. Masalahnya adalah bahwa *log* yang direkam tersebar di beberapa tempat, dan karena tidak berbentuk tabel, sulit untuk ditangani secara langsung [1]. Perangkat *server* merupakan perangkat penting dari sistem dan berjalan 24 jam sehari. Jika *server* mengalami masalah atau diserang, sulit bagi administrator untuk mendeteksi akar penyebab dan penyebab masalah *server* karena banyaknya jumlah *log*. Hal ini sering terlihat, sehingga sulit bagi administrator menentukan keberadaan pertanyaan [2]. Masih dipertanyakan upaya mengelola *log* dan memproses sejumlah besar data *log*, menghasilkan informasi aktivitas terperinci dan memvisualisasikan data *log* agar mudah dipahami oleh pengguna.

Log File adalah salah satu prosedur penting dalam keamanan jaringan komputer. Namun, seiring berjalannya waktu, volume informasi *log* meningkat seiring dengan rentang waktu dikarenakan jumlah klien yang bertambah. Penganalisaan *log* yang ada belum bisa menampung itu semua. Untuk mengatasi masalah ini, sudah pernah diatasi dengan memanfaatkan penerapan *Big Data* seperti halnya *Hadoop Framework* yang telah umum digunakan sebagai aplikasi *Big Data*. Hasil diproses dengan kombinasi *MapReduce* sebagai segmen utama *Hadoop* untuk menganalisis *log* dan HDFS sebagai wadah informasi. Beberapa penelitian sebelumnya juga telah dilakukan dengan *tool* yang berbeda, bersama-sama dengan *Hadoop Framework* untuk tujuan penyelidikan. Kesimpulannya,

e-ISSN: 2809-3399, p-ISSN: 2338-5197 Vol. 12., No. 1, Februari 2024, pp. 1-9

studi tentang berbagai *Framework* dan sistem yang ada menggunakan *Hadoop* dan *Framework* lainnya memberikan pentingnya menganalisis *file log* untuk menarik wawasan. Juga berbagai teknik dapat digunakan untuk menganalisis *file log* dan mengekstrak wawasan yang dapat ditindaklanjuti dari *file log* [3]. Namun, penggunaan *Docker* juga dapat menampung beberapa keinginan *brainware* sebagai wadah untuk menerapkan sistem *container* juga dengan mudah menjalakan program.

Docker adalah *platform* sumber terbuka yang mengotomatiskan penyebaran perangkat lunak dalam kontainer *Docker*. *Container* adalah *standalone package* yang berisi semua yang diperlukan untuk menjalankan perangkat lunak tertentu, sambil mengisolasi dari infrastruktur dasar yang menjalankannya. *Container* berbeda dari mesin *virtual* dalam hal mereka tidak berisi seluruh sistem operasi, tetapi hanya program yang berjalan di atas sistem operasi. Ini mengurangi *overhead*, menjadikan ringan, mengurangi memori sistem, disk, dan penggunaan sumber daya lainnya [4]. Oleh karena itu, sistem yang akan diimplementasikan nanti bisa dipasang pada *Docker* untuk menampung kemudahan yang ada dari *Docker*.

Penelitian ini menggambil lokasi di sebuah kantor pemerintahan yang dibentuk setelah otonomi daerah pada tahun 1999. Berdasarkan hasil pengamatan ditemukan bahwasanya kantor pemerintah ini masih kesulitan mengolah *file log* yang dihasilkan dan masih memiliki banyak kekurangan dalam proses menganalisis *log*. Penganalisaan *log* masih manual dan belum efektif pada segi pengelolaan *log*, terlebih dalam segi pendekatan *log file*-nya. Dari masalah tersebut mengakibatkan data *log* yang sudah dikumpulkan belum dapat dikelola dan dianalisis dengan baik untuk mendapatkan informasi lebih lanjut terkait upaya yang dapat dilakukan selanjutnya seperti menampilkan "peringatan" atau "*error*" terhadap sistem yang sedang berjalan, kebutuhan forensik, dan sebagainya. Hal ini tentunya berdampak pada tugas dan tanggung jawab operasional kantor pemerintahan dalam meningkatkan kualitas keamanan informasinya.

Keamanan informasi adalah praktik melindungi informasi dengan mengurangi risiko serangan dunia maya, dan biasanya mencakup pencegahan atau pengurangan kemungkinan akses data yang tidak sah/tidak pantas, penggunaan yang melanggar hukum, pemberitahuan rahasia, gangguan. Konsep keamanan informasi juga mencakup berbagai prosedur yang bertujuan untuk meminimalkan efek negatif dari insiden dan ancaman tersebut. Ancaman ini mungkin berasal dari perilaku manusia yang dapat menyebabkan kerusakan besar pada aset data organisasi. Dengan demikian, fokus utama keamanan informasi adalah perlindungan yang seimbang terhadap *confidentiality* (kerahasiaan), *integrity* (integritas) and *availability* (ketersediaan) data sambil mempertahankan penggunaan sistem secara efektif. Keamanan informasi adalah tentang pencapaian tiga tujuan utama yaitu kerahasiaan, integritas, dan ketersediaan melalui berbagai ukuran dan mekanisme. Risiko yang mengancam keamanan informasi tidak terbatas, tetapi banyak di antaranya disebabkan oleh perilaku manusia di dalam organisasi [5].

Sesuatu yang menarik, telah dirancang sistem manajemen *server log* menggunakan *ELK Stack* (*Elasticsearch, Logstash, Kibana*), yang dapat mengirim *log* dari *server*, mengumpulkan *log*, dan memvisualisasikan *log* [2]. *ELK Stack* adalah kombinasi dari *Elasticsearch* yang menerima data mentah untuk membentuk indeks yang berbeda, *Logstash* yang digunakan untuk mengumpulkan *log* dan *Kibana* digunakan untuk visualisasi. *ELK stack* sebagai pengelolaan *log* total, dan tugas setiap segmen *open source* dari *stack*. Ini adalah pengaturan investigasi *log* yang sempurna, berdasarkan campuran dari tiga instrumen *open source*: *Elasticsearch*, *Logstash*, dan *Kibana*. Ini mencoba untuk mengatasi setiap masalah dan kesulitan yang ditemukan di area lalu. *ELK* menggunakan tumpukan *opensource Elasticsearch* untuk pencarian dan penyelidikan informasi, *Logstash* untuk *logging* terkonsentrasi, yang menggabungkan pengiriman *log* dari berbagai *server*, peningkatan *log*, dan penguraian, serta *Kibana* untuk visualisasi informasi yang luar biasa. *ELK stack* saat ini dipelihara dan didukung secara aktif oleh perusahaan bernama *Elastic* [6].

Dari paparan masalah sebelumnya, maka kantor pemerintahan ini membutuhkan sebuah sistem pengelolaan *log* lengkap, seperti *management log* yang efektif untuk menampung masalah yang sedang terjadi. Diharapkan sistem tersebut dapat memonitor *log* jaringan, terkait fungsi untuk *log management*, *log collection*, *log transformation*, *log file*. Meskipun pada saat ini ada banyak alat yang digunakan dengan bermacam metode dalam mengumpulkan *log* dan menganalisis *log* tersebut

untuk mendeteksi aktivitas jahat, serta juga banyak alat komersial untuk memberikan hal yang sama secara lebih akurat. Maka akan diimplementasikan metode *ELK Stack*, yaitu kombinasi dari *Elasticsearch*, *Logstash*, dan *Kibana*. Alasan utama peneliti menggunakan arsitektur ini diharapkan dapat membangun pembaruan yang lebih tertata dengan sedikit usaha dan fungsionalitas yang didapat lebih akurat. Dengan diterapkannya manajemen *log* ini memudahkan *administrator* untuk membaca *log*, meningkatkan performa *log file*, mengidentifikasi masalah pada *server*, kemudian memiliki sistem pelaporan yang mudah dianalisis.

2. Metode

Subjek pada penelitian ini adalah peningkatan performa *log file* menggunakan kombinasi *Docker* dengan pengimplementasian ELK Stack didalamnya. Pengujian yang dilakukan menggunakan *log server Apache* yang berasal dari objek penelitian sebagai sampel data. Keamanan informasi adalah tentang pencapaian tiga tujuan utama: kerahasiaan, integritas, dan ketersediaan melalui berbagai ukuran dan mekanisme. Risiko yang mengancam keamanan informasi tidak terbatas, tetapi banyak di antaranya disebabkan oleh perilaku manusia di dalam organisasi. Dengan demikian, fokus utama keamanan informasi adalah perlindungan yang seimbang terhadap *confidentiality* (kerahasiaan), *integrity* (integritas), dan *availability* (ketersediaan) data serta mempertahankan penggunaan sistem secara efektif [5].

2.1. Logging

Data log yaitu dokumen peristiwa yang terjadi pada saat tertentu dan ada untuk tujuan dan jumlah waktu yang telah ditentukan. Baik kondisi normal maupun tidak normal memerlukan *log* dan pemrosesan *log*, yang pertama untuk memantau operasi, statistik, dan melakukan analitik [4]. Dalam istilahnya *Logging* adalah kontrol keamanan informasi penting yang digunakan untuk mengidentifikasi, menanggapi, dan mencegah masalah operasional, insiden keamanan, pelanggaran kebijakan, aktivitas penipuan; mengoptimalkan performa sistem dan aplikasi; membantu dalam kegiatan pemulihan bisnis; dan, dalam banyak kasus, mematuhi undang-undang dan peraturan federal, negara bagian, dan lokal. Tujuan dari standar ini adalah untuk menentukan ekspektasi dan persyaratan *logging* untuk sistem teknologi informasi (TI) [7].

Salah satu pembagiannya disebut *log management*, yaitu proses penanganan setiap peristiwa *log* yang dihasilkan oleh semua aplikasi dan infrastruktur perangkat lunak tempat untuk menjalankannya. Proses ini melibatkan pengumpulan, dekomposisi, penyimpanan, analisis, pencarian, pengarsipan, dan pembuangan. *Log management* memiliki tujuan akhir menggunakan data untuk memecahkan masalah dan mendapatkan wawasan bisnis, sekaligus memastikan kepatuhan dan keamanan aplikasi dan infrastruktur [8].

2.2. *Log File*

Log file adalah teknik memperoleh pengetahuan dari file *log* yang berisi catatan peristiwa dalam sistem komputer. Aplikasi umum dari *log file* adalah untuk memperoleh informasi penting tentang masalah keamanan dan gangguan sistem, yang selanjutnya mengarah pada kemampuan untuk mengidentifikasi dan berpotensi menghentikan penyusup yang menyerang sistem. Analisis *log* adalah salah satu metode paling populer untuk mendeteksi ancaman ini [9]. *Log file* merupakan proses penting untuk menentukan *health system* dan menemukan penyebab kegagalan sistem. *Log* dihasilkan oleh aplikasi, perangkat, dan server dan berisi informasi penting tentang sistem. Ada beberapa teknik umum untuk menganalisis data *log* secara manual. Namun, akurasi dan efektivitas teknik ini terbatas. Metode umum untuk menganalisis file *log* besar adalah dengan mencari kata kunci untuk entri *log* [10].

Analisis *log* membantu mengoptimalkan atau men-*debug* performa sistem dan memberikan masukan penting pada kemacetan sistem. Memahami performa sistem seringkali tentang memahami penggunaan sumber daya dalam sistem. Misalnya, *log* server web dapat membantu menentukan performa layanan individual berdasarkan waktu respons, kode respons HTTP, dan lainnya [11].

Jurnal Sarjana Teknik Informatika
Vol. 12., No. 1, Februari 2024, pp. 1-9

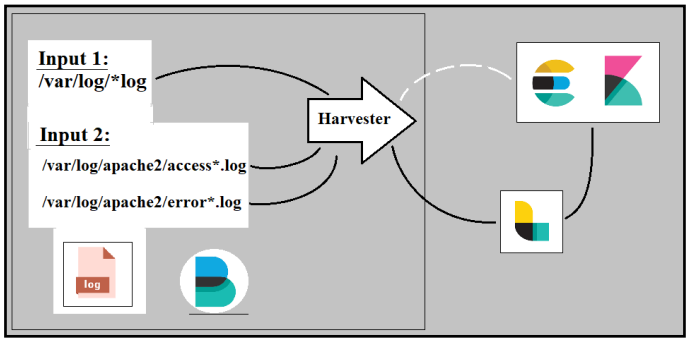
2.3. *ELK Stack*

Elasticsearch adalah mesin pencari dan analitik terdistribusi *opensource* yang bekerja dengan semua jenis data. Ini adalah *database* dan mesin pencari NoSQL yang akan digunakan untuk menyimpan dan mencari data. Ia dikenal dengan basic REST APIs dan properti skalabilitasnya. *Elasticsearch* adalah integran utama dari ELK (Elastic Stack). Ini dianggap sebagai salah satu mesin pencari terbaik yang mampu menangani data secara terstruktur dan tidak terstruktur. Penganalisis memainkan peran penting dalam pembangunan indeks data. Alat analisis yang tepat dapat meningkatkan akurasi indeks secara signifikan [12].

Logstash adalah saluran informasi yang memungkinkan upaya pengumpulan, penguraian, dan penyelidikan berbagai macam informasi dan kejadian terorganisir dan tidak terstruktur yang dihasilkan di berbagai kerangka kerja. Ini memberikan modul untuk berinteraksi dengan berbagai jenis sumber informasi dan tahapan, dan dimaksudkan untuk mahir memproses *log*, peristiwa, dan alokasi sumber informasi tidak terstruktur ke dalam berbagai hasil dengan penggunaan modul, khususnya catatan, atau *Elasticsearch* [6].

Kibana adalah tahap representasi informasi resmi *Apache 2.0 opensource*, yang membantu dalam menggambarkan segala jenis informasi terorganisir dan tidak terstruktur yang disimpan dalam catatan *Elasticsearch*. *Kibana* sepenuhnya ditulis dalam HTML dan JavaScript. Ini menggunakan kemampuan cocok dan luar biasa dari *Elasticsearch* yang ditemukan melalui RESTful API-nya untuk menunjukkan desain yang luar biasa bagi *client*. *Kibana* menjalankan tugasnya dengan mengungkap informasi melalui histogram yang cantik, peta geografis, *pie outlines*, *chart*, *tabel*, dan lainnya. *Kibana* menjadikan informasi yang sangat banyak. Antarmuka berbasis programnya yang sederhana memungkinkan dengan cepat membuat dan berbagi dashboard dinamis yang menunjukkan perubahan pada pertanyaan *Elasticsearch* secara bertahap [6].

Filebeat adalah *plugin Logstash* yang bertindak sebagai *proxy* di *server* asal, mengirim data ke ELK Stack. *Filebeat* menggantikan plugin *Logstash* lama, forwarder *Logstash* atau *lumberjack*. Di ELK Stack, *log* dari beberapa server aplikasi dikirim ke induk *Log*stash melalui pengirim *Log*stash [2].



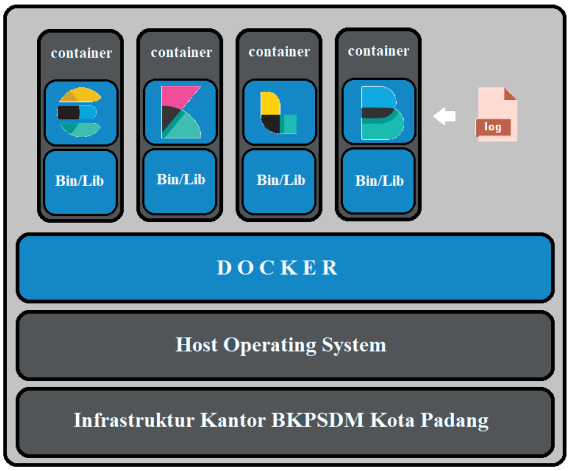
Gambar 1. Skema Pengiriman Filebeat Agent

2.4. *Docker*

Docker adalah *platform* sumber terbuka yang mengotomatiskan penyebaran perangkat lunak dalam kontainer *Docker*. *Container* adalah *standalone package* yang berisi semua yang diperlukan untuk menjalankan perangkat lunak tertentu, sambil mengisolasi dari infrastruktur dasar yang menjalankannya [4]. *Docker* sering disebut teknologi mesin kontainer tingkat tinggi berdasarkan LXC (Linux Container), dikembangkan dengan *open source* oleh *DotCloud*, yang merupakan solusi arus utama untuk proses virtualisasi. LXC adalah teknologi virtualisasi kernel yang menyediakan virtualisasi ringan untuk isolasi sumber daya dan proses [13].

Docker memungkinkan untuk "membangun, menerbitkan, dan menjalankan aplikasi apa pun di mana saja". Ini telah berjalan jauh dalam waktu yang sangat singkat dan sekarang dianggap sebagai cara standar untuk mengatasi salah satu aspek perangkat lunak yang paling mahal, yaitu penyebaran. Sebelum munculnya *Docker*, jalur pengembangan sering kali melibatkan kombinasi berbagai

teknologi yang digunakan untuk mengelola pergerakan perangkat lunak seperti mesin virtual, alat manajemen konfigurasi, sistem manajemen paket, dan jaringan dependensi perpustakaan yang kompleks. Semua alat ini perlu dikelola dan dipelihara oleh insinyur profesional, dan sebagian besar memiliki cara konfigurasi yang unik [14].



Gambar 2. Skema Sistem ELK Stack Dengan Docker

3. Hasil dan Pembahasan

3.1. Analisis Data

Berdasarkan masalah dan data yang didapat dari objek penelitian, pengelolaan *log* membutuhkan beberapa program yang akan dijalankan. Pilihan ELK Stack dikombinasikan dengan *Docker* untuk tujuan peningkatan kualitas dan keuntungan yang dihasilkan nanti. Untuk pengujian, peneliti menggunakan *log* Apache Kantor BKPSDM Kota Padang. Apache *log* yang didapatkan berupa access *log* dan error *log* yang dihasilkan dalam beberapa bulan terakhir. Maka dari itu perlu dilakukan perancangan terlebih dahulu sehingga *log* dapat dianalisa nantinya.

Log Akses berisi informasi tentang permintaan yang masuk ke server web. Informasi ini dapat mencakup halaman apa yang dilihat orang, status keberhasilan permintaan, dan berapa lama waktu yang dibutuhkan permintaan untuk merespons. Akses *log* pada Kantor BKPSDM Kota Padang terlihat seperti Gambar 3.

```
.
.
182.4.69.173 - - [06/Jun/2022:06:52:08 +0700]
"POST /konektor_absensi/api/cek_distance HTTP/1.1" 200 783 "-" "Dalvik/2.1.0
(Linux; U; Android 9; RMX1941 Build/PPR1.180610.011)"
::1 - - [06/Jun/2022:06:52:09 +0700]
"OPTIONS * HTTP/1.0" 200 - "-" "Apache/2.4.6 (CentOS) PHP/5.6.40
(internal dummy connection)"
36.69.11.184 - - [06/Jun/2022:06:52:09 +0700]
"GET /konektor_absensi/api/check_version HTTP/1.1" 200 202 "-" "Dalvik/2.1.0
(Linux; U; Android 12; M2101K9AG Build/SKQ1.210908.001)"
182.4.70.11 - - [06/Jun/2022:06:52:09 +0700]
"POST /konektor_absensi/api/cek_distance HTTP/1.1" 200 136 "-" "Dalvik/2.1.0
(Linux; U; Android 7.1.2; Redmi 4X MIUI/V11.0.2.0.NAMMIXM)"
103.108.22.13 - - [06/Jun/2022:06:52:09 +0700]
"POST /konektor_absensi/api/cek_distance HTTP/1.1" 200 134 "-" "Dalvik/2.1.0
(Linux; U; Android 8.1.0; CPH1909 Build/O11019)"
36.69.11.184 - - [06/Jun/2022:06:52:09 +0700]
"GET /konektor_absensi/api/biodata_pegawai?nip=199403152019022001 HTTP/1.1" 200 368 "-" "Dalvik/2.1.0
(Linux; U; Android 12; M2101K9AG Build/SKQ1.210908.001)"
103.144.222.70 - - [06/Jun/2022:06:52:09 +0700] "OPTIONS * HTTP/1.1" 200 - "-" "-"
36.69.11.184 - - [06/Jun/2022:06:52:09 +0700]
"GET /konektor_absensi/api/biodata_pegawai?nip=199403152019022001 HTTP/1.1" 200 368 "-" "Dalvik/2.1.0
(Linux; U; Android 12; M2101K9AG Build/SKQ1.210908.001)"
182.4.69.114 - - [06/Jun/2022:06:52:09 +0700]
"POST /konektor_absensi/api/cek_distance HTTP/1.1" 200 136 "-" "Dalvik/2.1.0
(Linux; U; Android 9; SM-J400F Build/PPR1.180610.011)"
::1 - - [06/Jun/2022:06:52:10 +0700]
"OPTIONS * HTTP/1.0" 200 - "-" "Apache/2.4.6 (CentOS) PHP/5.6.40
(internal dummy connection)"
182.4.70.11 - - [06/Jun/2022:06:52:10 +0700]
"POST /konektor_absensi/api/cek_metode HTTP/1.1" 200 100 "-" "Dalvik/2.1.0
(Linux; U; Android 7.1.2; Redmi 4X MIUI/V11.0.2.0.NAMMIXM)"
.
```

Gambar 3. Bentuk Access Log Kantor BKPSDM Kota Padang

Selain Access *log* juga terdapat yang namanya Error *log*, *file* merupakan format yang sama namun perbedaannya adalah informasi yang terdapat didalamnya.

Jurnal Sarjana Teknik Informatika
Vol. 12., No. 1, Februari 2024, pp. 1-9

```
.
.
[Sun Jun 05 06:40:30.902432 2022] [core:error] [pid 9749] [client 193.106.191.48:48432]
AH00126: Invalid URI in request POST /cgi-bin/.%2e/.%2e/.%2e/.%2e/bin/sh HTTP/1.1
[Sun Jun 05 11:11:52.194857 2022] [:error] [pid 14667] [client 222.216.102.177:40216]
PHP Notice:  Undefined index: log_path in /var/www/html/system/core/Log.php on line 127
[Sun Jun 05 11:11:52.194920 2022] [:error] [pid 14667] [client 222.216.102.177:40216]
PHP Notice:  Undefined index: log_path in /var/www/html/system/core/Log.php on line 127
[Sun Jun 05 11:11:52.194979 2022] [:error] [pid 14667] [client 222.216.102.177:40216]
PHP Warning:  mkdir(): Invalid path in /var/www/html/system/core/Log.php on line 131
[Sun Jun 05 11:11:52.195015 2022] [:error] [pid 14667] [client 222.216.102.177:40216]
PHP Notice:  Undefined index: log_threshold in /var/www/html/system/core/Log.php on line 138
[Sun Jun 05 11:11:52.195029 2022] [:error] [pid 14667] [client 222.216.102.177:40216]
PHP Notice:  Undefined index: log_threshold in /var/www/html/system/core/Log.php on line 142
[Sun Jun 05 11:31:56.048201 2022] [:error] [pid 12298] [client 101.0.38.184:20445]
PHP Notice:  Undefined index: log_path in /var/www/html/system/core/Log.php on line 127
[Sun Jun 05 11:31:56.048260 2022] [:error] [pid 12298] [client 101.0.38.184:20445]
PHP Notice:  Undefined index: log_path in /var/www/html/system/core/Log.php on line 127
[Sun Jun 05 11:31:56.048300 2022] [:error] [pid 12298] [client 101.0.38.184:20445]
PHP Warning:  mkdir(): Invalid path in /var/www/html/system/core/Log.php on line 131
[Sun Jun 05 11:31:56.048325 2022] [:error] [pid 12298] [client 101.0.38.184:20445]
PHP Notice:  Undefined index: log_threshold in /var/www/html/system/core/Log.php on line 138
[Sun Jun 05 11:31:56.048338 2022] [:error] [pid 12298] [client 101.0.38.184:20445]
PHP Notice:  Undefined index: log_threshold in /var/www/html/system/core/Log.php on line 142
.
.
```

Gambar 4. Bentuk Error *Log* Kantor BKPSDM Kota Padang

Log Error, berisi informasi tentang kesalahan yang ditemui server web saat memproses permintaan, seperti saat file hilang dan sebagainya. Error *log* Apache memungkinkan menemukan dan menyelesaikan masalah terkait server web. Error *log* melaporkan kesalahan yang berhubungan dengan, Konfigurasi, Eksekusi, File, Izin, Proksi, dan cache sesi. Misalnya, jika server melaporkan sejumlah besar kesalahan yang berbeda, ini mungkin merupakan indikasi bahwa ada masalah mendasar dengan sistem. Error *log* pada Kantor BKPSDM Kota Padang terlihat seperti Gambar 4.

3.2. Analisa Perancangan

Docker-compose adalah alat untuk mendefinisikan dan menjalankan *container*. Konfigurasi layanan aplikasi dilakukan dengan membuat file YAML. Perintah *compose*, lalu buat dan mulai semua layanan dari file YAML. Untuk *Docker Compose*, pertama-tama tentukan *Dockerfile* agar lingkungan aplikasi dapat direproduksi di mana saja. Kemudian, buat file "docker-compose.yml" dengan mendefinisikan semua layanan untuk aplikasi. Terakhir, jalankan seluruh aplikasi dengan perintah "docker-compose up" [15]. Contoh pembuatan konfigurasi docker-compose seperti Gambar 5. Maka tahap perancangan ELK Stack mengikuti konfigurasi yang ditentukan selanjutnya.

```
version: '3'
  services:
    web:
      build: .
      ports:
      - "5000:5000"
      volumes:
      - .:/code
      - logvolume01:/var/log
      links:
      - redis
    redis:
      image: redis
  volumes:
    logvolume01: {}
```

Gambar 5. Contoh Docker-compose.yml

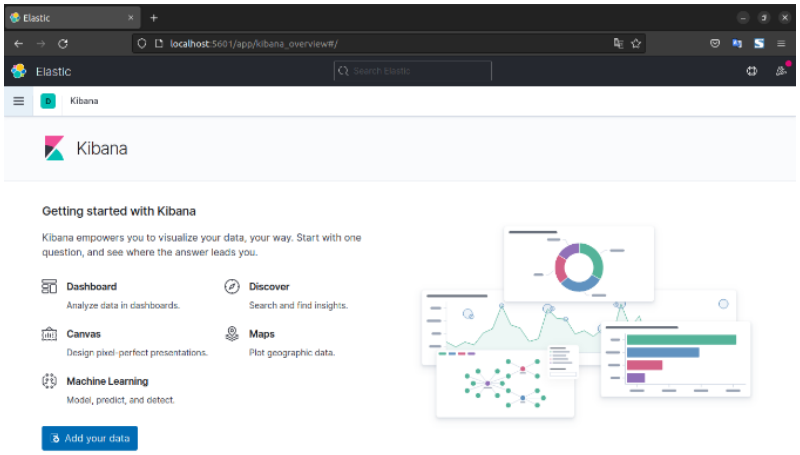
Contoh penjalanan hasil perancangan konfigurasi docker-compose seperti Gambar 6. Ini terjadi pada terminal sistem operasi. Apabila selesai merancang ELK Stack, maka akan muncul seperti Gambar 6.

```
$ docker-compose up
Creating dockercompose_server_1...
Creating dockercompose_client_1...
Attaching to dockercompose_server_1, dockercompose_client_1
client_1 | Received: Hello, world
client_1 |
client_1 | Received: Hello, world
client_1 |
```

Gambar 6. Eksekusi Docker-compose.yml

3.3. Analisis Sistem

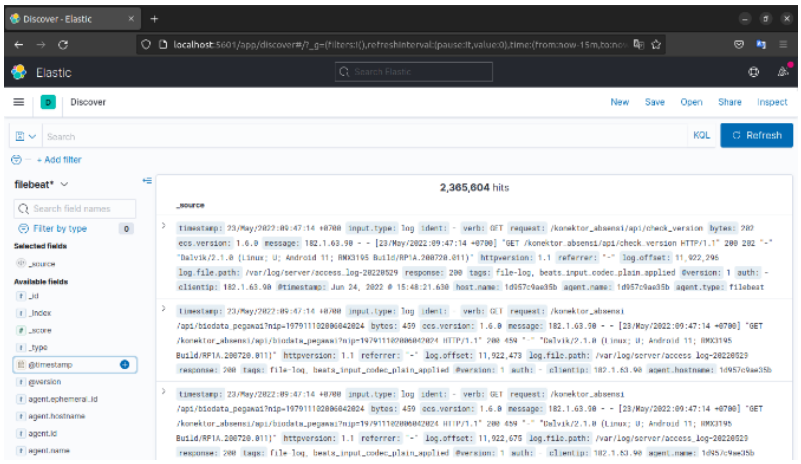
Apabila semua komponen ELK Stack yang dibutuhkan sudah berjalan, dan juga mengingat pendekatan yang dilakukan dalam penelitian ini adalah dengan menggunakan sekumpulan program *opensource* yang memiliki fungsi masing-masing. Maka secara singkat, *log file* akan diserap agen, ini adalah bagian yang bertugas mengirimkan data dari sisi yang lain dari sisi inti, bagian ELK Stack ini bernama *Beats*, lalu ia mengirimkan *log* ke bagian inti ELK Stack, seperti ke alamat *ip Logstash*. *Logstash* berfungsi mem-filter *log*, kemudian ini mengirim ke alamat *ip Elasticsearch* yang berkerja sama dengan *Kibana* dalam pemvisualisasian saat seluruh sistem berjalan pada *container Docker*. Maka tampak depan sistem ELK Stack yaitu Kibana dapat dikunjungi pada "localhost:5061" seperti Gambar 7.



Gambar 7. Tampilan Halaman Kibana

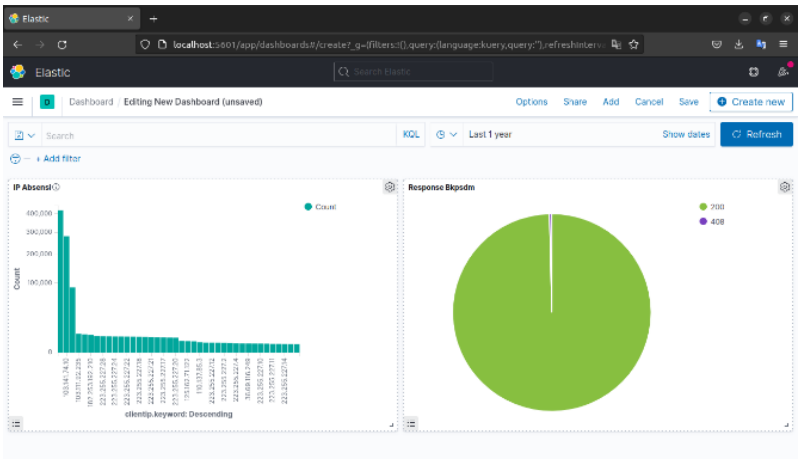
3.4. Pengujian Sistem

Pengujian ELK Stack berikut menggunakan data yang didapat dari Kantor BKPSDM Kota Padang. Data yang didapat berupa *log Apache* yang berasal dari *server* absensi karyawan Kantor BKPSDM Kota Padang. Apabila telah mengikuti tahap memasukkan *log* ke sistem ELK Stack untuk dianalisa, Maka hasil dapat dilakukan pencarian cepat pada Kibana Discover seperti Gambar 8.



Gambar 8. Tampilan Kibana Discover

Kemudian *log* yang masuk juga bisa dikelola pada *Dashboard Kibana*. Seperti dibuatkan visual statistik dari informasi yang diperlukan. Maka hasil visualisasi dapat dilihat seperti Gambar 9.



Gambar 9. Dashboard BKPSDM

4. Kesimpulan

Dengan ELK Stack dapat mengatasi masalah pengelolaan *log* yang terjadi pada Kantor pemerintahan di Kota Padang. Dibuktikan dengan simulasi pengujian *log* server absensi karyawan yang telah dilakukan berhasil mengalokasikan *log* ke dalam sistem ELK Stack, kemudian mengolah dan menganalisa *log* untuk mendapatkan informasi. ELK Stack memiliki pengaruh yang signifikan terhadap peningkatan kualitas pengelolaan *log*. Dibuktikan dengan penelitian yang telah dilakukan dalam mengobservasi masalah *log* di kantor, ternyata sistem ELK Stack memberikan berbagai solusi dalam penyelesaian masalah *log* yang ada. ELK Stack meningkatan performa *log file* kantor. Dibuktikan dengan seluruh keunggulan komponen sistem yang diuji, kemurahannya, kemudahannya, dan fiturnya yang mumpuni, yang memang tujuan sistem dikhususkan untuk pengelolaan dan penganalisaan *log file*.

Daftar Pustaka

[1] C. Tarigan, V. J. L. Engel, and D. Angela, "Sistem Pengawasan Kinerja Jaringan Server Web Apache dengan *Log Management* System ELK (*Elasticsearch*, *Log*stash, Kibana)," *J. Telemat.*, pp. 7–14, 2018, [Online]. Available: https://journal.ithb.ac.id/telematika/article/view/218

[2] P. H. Putra, "Implementasi *Log Management* Server Menggunakan Elk (Elastic Implementasi *Log Management* Server Menggunakan Elk (Elastic Search , *Log*stash Dan Kibana) Stack Pada Server Web Snort Di Pt . Xyz," *J. Inform. Sunan Kalijaga*, vol. 4, no. April, pp. 1–8, 2020.

[3] R. Y, A. Kothakota, and S. S, "A study on *log file* techniques," vol. 6, no. 04, pp. 2014–2017, 2019.

[4] M. Ruokola, "Centralized *log management* Miikka Ruokola Bachelor ' s thesis," no. May, 2017.

[5] A. Y. El-Bably, "Overview of the Impact of Human Error on Cybersecurity based on ISO/IEC 27001 Information Security *Management*," *J. Inf. Secur. Cybercrimes Res.*, vol. 4, no. 1, pp. 95–102, 2021, doi: 10.26735/wlpw6121.

[6] P. P. Bavaskar, O. Kemker, and A. K. Sinha, "a Survey on: '*Log File* With Elk Stack Tool,'" *Int. J. Res. Anal. Rev.*, 2019, [Online]. Available: https://ssrn.com/abstract=3677845

[7] IEEE, "Standard for Information Techno*log*y," no. May, pp. 3–6, 2017.

[8] A. Kosasih, "Designing Automation System Based on *Log Management* for Bank XYZ's Data Center," *Int. J. Emerg. Trends Eng. Res.*, vol. 8, no. 5, pp. 1721–1727, 2020, doi: 10.30534/ijeter/2020/37852020.

[9] J. Svacina *et al.*, "On Vulnerability and Security *Log file*: A Systematic Literature Review on Recent Trends," *ACM Int. Conf. Proceeding Ser.*, pp. 175–180, 2020, doi: 10.1145/3400286.3418261.

[10] Z. Zamanian, "Anomaly Detection in System *Log* Files Using Machine Learning," pp. 1–92, 2019.

[11] Bharvi Dixit, *Elasticsearch: A Complete Guide*. Published by Packt Publishing Ltd., 2017. [Online]. Available: https://www.packtpub.com/

[12] V. K. Et. al., "Twego Trending: Data Analytics Based Search Engine Using *Elasticsearch*," *Turkish J.*

Comput. Math. Educ., vol. 12, no. 1S, pp. 246–251, 2021, doi: 10.17762/turcomat.v12i1s.1764.

[13] H. Ju, J. Wang, E. Zhu, X. Zhang, and F. Zheng, "Design Scheme of a Docker Container File Isolation against Computer Virus Spreading," *Math. Probl. Eng.*, vol. 2022, pp. 1–6, 2022, doi: 10.1155/2022/5348370.

[14] A. H. S. Ian Miell, *Docker in Practice Second Edition*, Second edi. Manning Publications Co. All rights reserved. No, 2019.

[15] M. Shrestha, "Exploring Docker Implementation With WORDPRESS," 2019.