BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Penelitian oleh (Jain, Bhargava, & Rajput, 2016) yang berjudul "*Role of Log Management in Information Security Compliance*" yang membahas tentang pengimplementasian *Log Management Framework* yang mengatasi semua kebutuhan dalam keamanan informasi yang berhubungan dengan log. Dimana dari penelitian tersebut, disimpulkan bahwa manajemen log memegang peranan penting dalam sebuah organisasi. Selain itu log juga sangat berguna dalam mengidentifikasi masalah keamanan, pelanggaran kebijakan, aktivitas penipuan, dan masalah operasional. Log juga sangat penting dalam melakukan audit, analisa forensik, investigasi internal, dan mengidentifikasi masalah jangka panjang.

Salah satu penelitian oleh (Daubner, 2018) yang berjudul "*Effective Computer Infrastructure Monitoring*" membahas tentang perancangan dan pengimplementasian sebuah solusi *monitoring* untuk infrastruktur komputer di *Institute of Computer Science (ICS)* di Universitas Masaryk. Dimana permasalahan yang terjadi yaitu infrastruktur computer di *ICS* saat itu tidak memiliki solusi pemantauan terpusat yang terpadu. Hasil penelitian mendapatkan kesimpulan bahwa solusi manajemen log menggunakan *Graylog Collector Sidecar* menjadi lebih efisien dan lebih mudah digunakan dibandingkan dengan beberapa manajemen *log* lainnya seperti *Elastic Stack* dan mempunyai fitur keamanan secara *default* dibandingkan dengan *Kibana*.

Penelitian dari (Sahoo, Jena, Chottray, & Pattnaiak, 2012) yang berjudul "*Syslog a Promising Solution to Log Management*" membahas tentang perancangan sentralisasi manajemen log. Hasil penelitian menyebutkan bahwa bahwa manajemen log memastikan file log disimpan secara terpusat dengan aman dalam jangka waktu yang tepat sehingga berguna dalam analisa log lebih lanjut ketika dibutuhkan.

1.2 Landasan Teori

Dalam proses penelitian, penulis tentu membutuhkan teori – teori pendukung sebagai landasan dalam menyelesaikan penelitian ini. Teori yang dipakai penulis terdiri dari :

2.2.1 Jaringan Komputer

Jaringan komputer adalah kumpulan dari dua atau lebih komputer yang saling terhubung dan berbagi sumber daya seperti *printer, server, hardware,* serta pertukaran data dan fasilitas komunikasi elektronik. Komputer di dalam suatu jaringan dapat terhubung dengan kabel, *line* telepon, gelombang radio, satelit, ataupun kabel fiber optik (Ardiantoro, Triyono, & Fatkhiyah, 2016).

Menurut (Thakor & Joshi, 2015), berdasarkan cakupan wilayahnya, jaringan komputer dapat dibagi menjadi tiga, antara lain :

1. Local Area Network

LAN adalah jaringan yang diapakai untuk berkomunikasi antar perangkat komputer, biasanya masih didalam wilayah yang kecil rumah, kantor, sekolah atau *airport*. *LAN* memungkinkan untuk membagi sumberdaya seperti file atau perangkat *hardware* yang mungkin dapat digunakan oleh lebih dari satu pengguna seperti

printer, dan lain-lain. Kelebihan dari *LAN* yaitu kita mendapatkan kecepatan yang tinggi dan tidak terlalu mahal. Selain itu perangkat kita lebih aman.

2. Wide Area Network

WAN mencakup area yang sangat luas seperti komunikasi antar kota atau negara. Contoh dari *WAN* yaitu internet. Kecepatan transfer datanya tergantung *ISP provider* dan juga berbagai lokasi. Kelebihan dari *WAN* yaitu dapat menghubungkan antar negara walapun secara geografis sangat jauh sehingga bisnis jarak jauh pun dapat terhubung dalam satu jaringan. Kekurangan dari *WAN* yaitu diperlukan keamanan sisten yang baik untuk mencegah pengguna asing masuk dan menggangu jaringan. Selain itu, biaya yang harus dikeluarkan juga tidak sedikit dan bisa sangat rumit dan mendapat kecepatan yang pelan.

3. Metropolitan Area Network

MAN adalah sebuah jaringan komputer yang besar biasanya mencakup antar kota atau kampus besar yang berdiameter kurang lebih 5-50 KM. *MAN* mungkin dioperasikan dan dimiliki oleh satu organisasi namun dapat digunakan oleh banyak organisasi dan perorangan. Contoh *MAN* salah satunya yaitu parusahaan jaringan TV kabel dan perusahaan jaringan telepon yang menyediakan pelayanan berkecepatan tinggi untuk pelanggan. Kelebihan *MAN* terletak pada *bandwith*-nya. *MAN* menyediakan rangka yang bagus untuk jaringan yang lebih besar seperti *WAN*. Sumberdaya yang dibagi di *MAN* dapat digunakan dengan cepat. Kekurangannya membutuhkan kabel yang banyak dari satu tempat ke tempat lain, dan juga harganya yang mahal.

2.2.2 *IP Address*

IP Address adalah sebuah metode dalam mengalamatkan sebuah perangkat yang termasuk dalam jaringan komputer yang terdiri dari deretan suatu angka. Contoh *IP address* adalah 192.168.1.1 atau 10.0.0.2. Pada suatu jaringan komputer, seluruh komputer didaftarkan dengan alamat-alamat IP yang unik antara satu yang lainnya untuk mencegah kemungkinan terjadinya kesalahan. IPv4 yang mana merupakan IP yang paling banyak digunakan diciptakan pada awal tahun 1980 dan melayani komunitas internet global lebih dari tiga dekade. IPv4 mempunyai kapasitas lebih dari empat milyar alamat IP. Namun kapasitas tersebut terbatas dan dapat habis seiring perkembangan teknologi. Hanya 3,7 milyar alamat IPv4 yang dapat digunakan oleh perangkat biasa, sisanya digunakan untuk protokol spesial seperti IP Multicasting (Singh A. K., 2015).

Menurut (Tedyyana & Kurniati, 2016), *IP address* dibagi menjadi dua macam, yakni *public IP address* dan *private IP address*. *Public IP address* mendaftarkan semua komputer yang terhubung ke internet dimana setiap IP yang didaftarkan unik. Sehingga tidak ada dua komputer yang memiliki *IP address* yang sama di internet dan memungkinkan setiap komputer untuk terhubung secara online satu sama lain dan bertukar data. Pengguna tidak mempunyai kontrol atas IP publik yang terhubung ke komputer karena *IP address* tersebut didaftarkan oleh ISP (*Internet Service Provider*) saat komputer terhubung di gateway internet. Sebuah IP publik dapat berupa statik atau dinamis. IP Statik tidak akan berubah dan dipakai khususnya untuk hosting halaman web atau layanan internet. Sedangkan IP dinamik dipilih dari kumpulan IP yang tersedia dan berubah setiap kali terhubung ke internet. *Private IP* address adalah IP yang termasuk dalam salah satu dari range *IP private* seperti *LAN*. *IP private* didaftarkan untuk komputer yang berada di jaringan private seperti rumah, dan

sekolah yang mana memungkinkan komputer di dalam jaringan untuk berkomunikasi antara satu dengan yang lain.

2.2.3 Topologi Jaringan

Menurut (Bisht & Singh, 2015), topologi jaringan adalah suatu studi atau aturan dalam menghubungkan kumpulan komputer atau elemen-elemen jaringan lain secara fisik dan bagaimana pola hubungan antar komponen yang terhubung melalui media atau peralatan jaringan seperti pemasangan kabel, server, dan *switch/router*.

Menurut (Pandya, 2013), ada beberapa tipe dasar topologi jaringan, yaitu :

1. Topologi *Bus*

Pada topologi ini, kumpulan komputer dihubungkan dari sebuah kabel jaringan tunggal yang disebut *bus* yang bertindak sebagai *backbone.* Keuntungan dalam menggunakan topologi ini yaitu mudahnya dalam proses implementasi dan pengembangan, lebih murah karena menggunakan sedikit kabel untuk menghubungkan seluruh komputer, dan mudah digunakan. Sedangkan kekurangannya apabila trafik jaringan sedang tinggi maka akan berdampak pada kecepatan jaringan tersebut, Apabila terjadi kegagalan pada kabel utama maka akan mematikan seluruh transmisi.

2. **Topologi *Star***

Pada topologi ini, *switch* atau *hub* bertindak sebagai pusat dalam menghubungkan seluruh komponen jaringan. Setiap pengguna atau perangkat tidak terhubung antara satu dengan yang lainnya secara langsung. Kelebihan

menggunakan topologi *star* yaitu kegagalan dalam jaringan dapat dengan mudah terdeteksi dan apabila terjadi kegagalan dalam suatu terminal maka terminal lain tidak akan terganggu. Untuk kelemahannya, topologi ini sangat mahal untuk diimplementasikan dan keseluruhan jaringan sangat bergantung pada *hub/switch*.

3. Topologi *Ring*

Pada topologi ini, kabel jaringan menghubungkan satu node ke node lainnya sampai semua node terhubung membentuk sebuah lingkaran atau *ring*. Topologi ini menyediakan performa yang tinggi dan mudah untuk dikembangkan. Kelemahan dari topologi ini yaitu sangat sulit dalam troubleshooting apabila terjadi masalah dan juga harga implementasinya yang mahal.

4. **Topologi *Mesh***

Topologi *mesh* merupakan topologi yang mana memiliki koneksi *point-to-point* antar node. Topologi ini memiliki kelebihan yaitu menyediakan keamanan dan privasi setiap komputer dan masalah jaringan menjadi sangat mudah didiagnosa. Kelemahannya yaitu kebutuhan kabel dan *port input/output* yang sangat tinggi.

5. **Topologi *Tree***

Topologi pohon merupakan topologi yang mana hanya terdapat satu rute diantara dua node di jaringan. Kelebihan dari topologi ini yaitu mudahnya dalam pengelolaan dan penjagaan dan mudah dalam mendeteksi

error. Kekurangan topologi ini yaitu membutuhkan banyak kabel dan apabila *hub* pusat terjadi masalah maka keseluruhan jaringan terputus.

2.2.4 Ubuntu

Sistem operasi *Ubuntu* mempunyai 2 versi yaitu versi *desktop* dan versi *server*. *Ubuntu* merupakan turunan dari *Debian Linux*. Di *Ubuntu*, tersedia dua tipe *GUI* yaitu *GNOME* dan *KDE*. Di dalam *ubuntu desktop* tersedia aplikasi *Open office, web browser*, aplikasi pesan, *text* dan *graphic editor*, dan *game* secara *default* (U, B, & D, 2016).

2.2.5 Java

Java merupakan salah satu bahasa pemrograman yang dapat dijalankan di banyak sistem operasi. Karena kelebihannya tersebut *java* dikenal juga dengan bahasa pemrograman *multiplatform*. Selain itu *Java* juga pemrograman berorientasi objek dan memiliki *library* yang lengkap. *Java* diciptakan oleh James Gosling saat masih berada di Sun Microsystem. Saat ini *Java* menjadi salah satu bahasa pemrograman yang paling banyak digunakan yang mana *Java* sejatinya adalah *upgrade* dari Bahasa C++ (Fridayanthie & Charter, 2016).

2.2.6 MongoDB

MongoDB adalah *database noSQL* yang berbasis C++ dan *open source* yang dirilis pada tahun 2009. Konsep dari *MongoDB* adalah dokumen sebagai unit dasar untuk menyimpan data. Sebuah dokumen *MongoDB* mendukung dan meyediakan berbagai tipe struktur data terdiri dari nomor, tanggal, *array*, dan lain-lain (Latta, 2013).

MongoDB menggunakan JSON sebagai format penyimpanan data di *database* dan dapat digunakan untuk menyimpan data terstruktur maupun tidak terstruktur. *MongoDB* juga

mendukung *query* untuk *database* sehingga dapat dijadikan alternatif RDBMS (Wu, Huang, & Lee, 2015).

MongoDB menyediakan 2 kebijakan replika untuk persiapan ketika terjadi kegagalan sistem dan *data loss* yaitu *master-slave* dan *replica-set*. Pendekatan *master-slave* meningkatkan kinerja dan kehandalan sistem dengan cara menyimpan replika yang dihasilkan di *master node* disimpan di *slave node*. Sedangkan pendekatan *replica-set* terdiri dari dua atau lebih salinan data. Ketika master node mengalami masalah, *MongoDB* secara otomatis memilih *master* alternatif diantara *node* lain dan menditribusikan replika baru untuk replika yang hilang. *MongoDB* menyediakan manajemen yang mudah lewat fungsi *AutoSharding*, yang mana mengizinkan pengguna untuk memisahkan data dan menyimpannya di *shard* (Kim, Cui, & Lee, 2015).

2.2.7 Elasticsearch

Elasticsearch merupakan mesin pencari berbasis *full-text* yang handal dan menyediakan kemampuan untuk melakukan pencarian dokumen berbeda dengan cepat dan *realtime*. *Elasticsearch* bersifat *open source* dan menggunakan *platform Java*. *Elasticsearch* merupakan *upgrade* dari *Apache Lucene* yang mana membawa beberapa kelebihan dibanding *Lucene* seperti *API* yang lebih sederhana, kemudahan dalam penggunaan operasional dan terdapat fitur *cluster* dan *replica* (Taware & Shaikh, 2017).

Elasticsearch memungkinkan pengguna untuk mencari, menganalisis dan menjelajahi data yang berada di dalamnya. *Elasticsearch* mencari data di *field* untuk menemukan dokumen dan menampilkan hasil yang paling relevan terlebih dahulu. Relevansi dokumen yang ditampilkan sangat akurat mengingat *Elasticsearch* menggunakan

model *Boolean* dalam pencarian dokumen. Segera setelah dokumen yang cocok dengan suatu *query* ditemukan, *Elasticsearch* menghitung jumlah *query* tersebut dan menggabungkannya dengan *query* lain yang cocok (Kalyani & Mehta, 2017).

Terdapat beberapa komponen dasar *Elasticsearch* yang perlu diketahui untuk memahami bagaimana cara *Elasticsearch* bekerja (Singh, Kumar, Singhal, & Dubey, 2018). Komponen-komponen dasar *Elasticsearch* tersebut antara lain:

1. Index

Index merupakan kumpulan dari beberapa dokumen yang mempunyai karakteristik yang sama. Sebuah *index* dapat memiliki 0 atau lebih *shard* atau *replica*.

2. Document

Dokumen merupakan kumpulan dari data atau informasi dasar yang dapat di *index*. Dokumen menggunakan format JSON yang merupakan satu-satunya format yang didukung *Elasticsearch.*

3. Type and Mapping

Setiap dokumen di *Elasticsearch* mempunyai tipe tersendiri. *Type* bisa sangat berguna untuk memisahkan data yang sama namun tidak identik. Sedangkan *mapping* bertindak sebagai skema *database* yang menggambarkan properti yang dimiliki oleh tipe dokumen tertentu.

4. Node

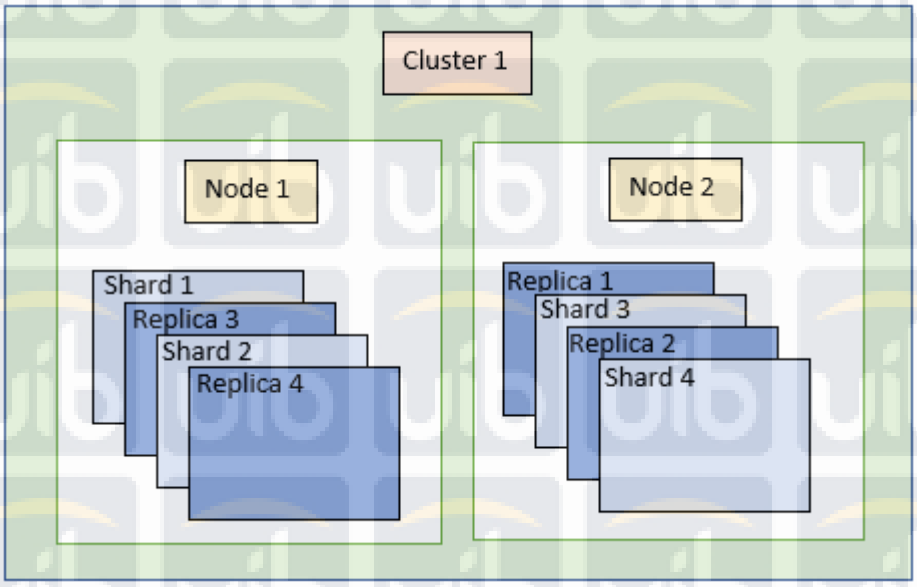
Node merupakan *server* tunggal yang merupakan bagian dari *cluster*. Sebuah *node* dapat dikonfigurasi untuk dimasukkan ke dalam *cluster* tertentu berdasarkan namanya.

5. Cluster

Cluster merupakan kumpulan dari satu atau lebih *node* yang menyimpan seluruh data dari *node* tersebut dan menyediakan kemampuan untuk *searching* dan *indexing*. *Cluster* diidentifikasi dengan *ID* nama yang unik.

6. Shard dan Replica

Elasticsearh menyebarkan data di *cluster* dalam beberapa penyimpanan yang disebut *shard*. *Elasticsearch* membagi data ke dalam beberapa *shard* secara otomatis dan mengikatnya dalam bentuk *index*. Sedangkan *replica* merupakan salinan dari *shard*. Alasan utama mengapa kita perlu menggunakan *replica* karena *replica* menyediakan ketersediaan data apabila terjadi kegagalan dalam *node* atau *shard* dengan catatan *shard* dan *replica* harus berada di dalam *node* yang berbeda. Untuk penejelasan lebih lanjut, silahkan lihat Gambar 2.1.



Gambar 2.1 Komponen dasar *Elasticsearch*

Menurut (Gupta & Nair, 2016), Alur kerja *Elasticsearch* terdiri dari tiga langkah, antara lain:

1. Proses Start-Up

Ketika *node elasticsearch* mulai berjalan, *node* tersebut akan mencari *node* lainnya di dalam *cluster* yang sama dan membuat *node* tersebut berhubungan. Dan salah satu *node* yang dianggap layak akan dijadikan *master node*. *Master node* bertanggung jawab dalam mengelola status *cluster* serta proses pendaftaran *shard* ke dalam *node* sebagai reaksi dalam perubahan topologi *cluster*.

2. Query Data

Query API merupakan bagian besar dari *Elasticsearch API*. Proses *query* tersebut dapat dibagi menjadi 2 fase, yaitu *scatter phase* dan *gather phase*. *Scatter phase* yaitu meng*query* seluruh *shard* yang relevan di dalam *index*. Sedangkan *gather phase* yaitu mengumpulkan hasil dari *shard* yang relevan, menggabungkannya, mensorting, memproses dan mengembalikan kepada *client.*

3. Index Data

Ada beberapa cara untuk mengirim data ke *Elasticsearch*. Salah satu cara termudah yaitu menggunakan *Index API,* yang mana memungkinkan kita untuk mengirim satu dokumen ke *index* tertentu. Cara kedua memungkinkan untuk mengirim banyak dokumen menggunakan *bulk API* dan *bulk UDP*. Perbedaan dari kedua metode ini adalah tipe koneksinya dimana *UDP* mengirimnya dengan protocol datagram tanpa koneksi yang mana lebih cepat namun tidak terlalu dapat diandalkan.

2.2.8 Manajemen Log

Log merupakan data yang menampilkan informasi tentang proses, pemakaian, kegagalan, dan kinerja suatu sistem. *File log* merekam data yang sedang terjadi ketika system berjalan. Data yang direkam adalah berupa *event* seperti aktivitas pengguna, status eksekusi program, pemakaian *resource* sistem, dan perubahan data (Reddy, Kumar, & B, 2014).

Log sudah ada sejak komputer diciptakan. Dengan munculnya system komputer modern dan layanan *cloud*, manajemen *log* menjadi sangat penting dalam menyediakan stabilitas, kinerja, dan keamanan aplikasi. *Log* biasanya dipakai untuk memecahkan suatu masalah, namun sekarang *log* menyediakan berbagai fungsi di suatu organisasi, seperti mengoptimalkan kinerja sistem dan jaringan, merekam aktivitas pengguna, dan menyediakan data yang sangat berguna untuk menginvestigasi aktivitas jahat. Salah satu contoh manfaat *log* yaitu *log* memegang peranan penting dalam keamanan. *Log* dapat mengetahui sistem program yang *crash*, kegagalan dalam *login*, dan *antivirus* dapat mengetahui suatu *malware* dari lognya (Hrvola, 2018).

Menurut (Sahoo, Jena, Chottray, & Pattnaiak, 2012), macam-macam *log* berdasarkan sumbernya dapat dibedakan menjadi tiga, antara lain:

1. *software log*, sebagian besar organisasi memakai beberapa tipe *software* keamanan berbasis jaringan atau berbasis *host* untuk mendeteksi aktivitas berbahaya, melindungi sistem dan data. Oleh karena itu, *software* keamanan merupakan sumber utama dalam *log* keamanan komputer.

2. *Operating systems log*, contohnya sistem operasi di *server, workstation, printer*, dan perangkat jaringan seperti *router* dan *switch*. *Log* yang diambil biasanya yang berisi informasi yang berhubungan dengan keamanan.
3. *Applications log*, contohnya seperti *email server, web server, browser, file server*, dan *database server*. Semua aplikasi tersebut menghasilkan berbagai jenis *log*.

Manajemen *log* adalah suatu solusi dalam menangani *log* dalam jumlah besar. Manajemen *log* yang baik dapat meningkatkan kualitas layanan, keamanan, dan efisensi dari perangkat yang dipantau. Dengan menggunakan manajemen *log*, administrator mendapat notifikasi apabila terjadi masalah, seperti *error report* dari *harddisk drive* (Sudha & Kumar, 2015).

Menurut (Agrawal & Makwana, 2015), beberapa tujuan dari manajemen log yaitu mengoleksi log dari semua sumber log seperti syslog, winlog, dan lain-lain. Membuat pencarian log dan memulihkan log yang sudah diarsipkan menjadi cepat dan fleksibel serta mengidentifikasi penyimpangan dalam aplikasi, database, system dan perangkat secara realtime.

2.2.9 Filebeat

Filebeat adalah bagian dari *Beats*, yaitu *log collector open source* yang mem*forward log* dari *server* ke *server* pusat. *Filebeat* merupakan sebuah upgrade dari *Logstash Forwarder*. Tujuan *filebeat* yaitu untuk memonitor setiap *file log* dan mengirimkan pesan *log* tersebut ke target yang diinginkan (Turnbull, 2017).

Filebeat bekerja dengan cara menyiapkan satu atau lebih *prospector*, yang mana setiap *prospector* mengelola satu set *harvester*. Sebuah *prospector* mempunyai tipe yang menentukan jenis input apa yang digunakan. Sebuah *prospector* memantau satu atau beberapa *path local file*.

```
- type: log
  paths:
    - /var/log/*.log
    - /var/log/messages.txt
```

Gambar 2.2 *Prospector File*

Berdasarkan Gambar 2.2, *Prospector* tersebut memantau *folder* /var/log. Sebuah *harvester* akan dibuat untuk setiap *file* yang sesuai yang ditemukan di dalam *path folder* tersebut. *File* tersebut dapat berupa *messages.txt* atau *file* apapun yang berekstensi *.log*. *Harverster* membaca setiap baris dari *file* tersebut, lalu mengirimkannya ke *output* yang sudah ditentukan. *Filebeat* menyimpan *registry* dari status *file* untuk melacak berapa banyak baris yang sudah dibaca disetiap *file*. *Filebeat* menjamin bahwa setiap pesan diterima setidaknya satu kali dengan cara mencoba untuk mengirim pesan secara berulang-ulang sampai diterima oleh *output*. Ketika *filebeat* dimatikan, ada beberapa pesan yang sudah dikirim, namun tidak diterima oleh *output*. Pesan tersebut akan dikirim lagi ketika *filebeat* dijalankan, yang mana berpotensi menghasilkan *file* duplikat di *output*. *Filebeat* tidak dapat menyaring pesan duplikat yang terdapat di dalam *file log* dan mem*forward* setiap pesan tidak peduli pesan tersebut memiliki identitas yang unik atau tidak. Hal tersebut baru dapat diatasi setelah pesan berada di *logging pipeline* (Norrby, 2018).

2.2.10 Graylog

Graylog adalah *platform* manajemen *log* yang bersifat *open source* untuk mengumpulkan, *indexing*, dan analisis berbagai tipe data secara *realtime*. Tujuan dari *Graylog* adalah memberikan kemudahan dalam pekerjaan dan kemampuan untuk bekerja dengan *log entries* yang berjumlah besar. *Graylog* mengumpulkan *data log* dari beberapa *source* dan menampilkan data tersebut dengan singkat, *interface* yang sederhana dan menampilkan visualisasi data yang kuat untuk mengambil langkah selanjutnya (Phaltane, Nahar, & Garge, 2014).

Graylog bergantung pada dua buah *software*, yaitu *Elasticsearch* dan *MongoDB*. *Elasticsearch* digunakan sebagai tempat penyimpanan untuk *log* dan juga sebagai mesin pencari. Sedangkan *MongoDB* digunakan sebagai tempat penyimpanan pendukung untuk meyimpan konfigurasi dan *metadata*, bukan untuk log (Murínová, 2015).

Salah satu fitur *Graylog* adalah *Streams*. *Streams* merupakan sebuah mekanisme untuk mengambil *log entries* secara *realtime*. *Streams* memungkinkan untuk mengirim sebuah *subset* dari *log entries* ke sistem yang berbeda lewat *output plugin* dan untuk membuat sebuah peringatan. *Graylog* tidak mempunyai solusi yang didedikasikan untuk *log collection*. Namun *Graylog* mempunyai *Collector Sidecar* yang merupakan konfigurasi manajemen sistem yang ringan untuk *collector* seperti *nxlog*, *winlogbeat* dan *filebeat* (K & R, 2015).

2.2.11 Log Collector

Log Collector adalah sebuah pengirim *log* yang dipasang pada sisi *client*. *Log collector* menyediakan solusi dalam membangun manajemen *log* dengan cara

mengumpulkan *log* yang berasal dari *client* sehingga banyak *log* dapat dikumpulkan secara terpusat. *Log collector* digunakan untuk mengatasi kompleksitas pengguna dalam mengorganisir dan mengambil *file log* dari *server* yang banyak. *Log collector* menyimpan semua datanya dalam HDFS (*Hadoop Distributed File System*) di beberapa *cluster* komputer (Likhita & Sahoo, 2016).