PROCEEDINGS OF SCIENCE

A solution for secure use of Kibana and Elasticsearch in multi-user environment

Wataru Takase*
High Energy Accelerator Research Organization (KEK)
E-mail: wataru.takase@kek

Tomoaki Nakamura
High Energy Accelerator Research Organization (KEK)
E-mail: tomoaki.nakamura@kek.jp

Yoshiyuki Watase
High Energy Accelerator Research Organization (KEK)
E-mail: yoshiyuki.watase@kek.jp

Takashi Sasaki
High Energy Accelerator Research Organization (KEK)
E-mail: takashi.sasaki@kek.jp

Monitoring is indispensable to check status, activities, or resource usage of IT services. A combination of Kibana and Elasticsearch is used for monitoring in many places such as KEK, CC-IN2P3, CERN, and also non-HEP communities. Kibana provides a web interface for rich visualization, and Elasticsearch is a scalable distributed search engine. However, these tools do not support authentication and authorization features by default. There is no problem in the case of single-user environment. On the other hand, in the case of single Kibana and Elasticsearch services shared among many users, any user who can access Kibana can retrieve other's information from Elasticsearch. In multi-user environment, in order to protect own data from others or share part of data among a group, fine-grained access control is necessary.
The CERN cloud service group had provided cloud utilization dashboard to each user by Elasticsearch and Kibana. They had deployed a homemade Elasticsearch plugin to restrict data access based on a user authenticated by the CERN Single Sign On system. It enabled each user to have a separated Kibana dashboard for cloud usage, and the user could not access to other's one. Based on the solution, we propose an alternative one which enables user/group based Elasticsearch access control and Kibana objects separation. It is more flexible and can be applied to not only the cloud service but also the other various situations. We confirmed our solution works fine in CC-IN2P3. Moreover, a pre-production platform for CC-IN2P3 has been under construction.
We will describe our solution for the secure use of Kibana and Elasticsearch including integration of Kerberos authentication, development of a Kibana plugin which allows Kibana objects to be separated based on user/group, and contribution to Search Guard which is an Elasticsearch plugin enabling user/group based access control. We will also describe the effect on performance from using Search Guard.

International Symposium on Grids and Clouds 2017 -ISGC 2017-
5-10 March 2017
Academia Sinica, Taipei, Taiwan

*Speaker.

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).

PoS(ISGC2017)008

1. Introduction

For reliable operation of IT services, a regular monitoring of system status, user activities, and resource usage is essential. Elasticsearch[1] and Kibana[2] are open-source monitoring tools developed by Elastic. Elasticsearch is a distributed, RESTful, horizontally scalable full-text search and analytics engine based on Apache Lucene. Kibana provides a variety of ways to visualize Elasticsearch data on a web interface. On the interface, a user can define visualizations of data in Elasticsearch and can create dashboards through arranging and resizing the visualizations. These visualizations and dashboards are called Kibana objects and stored to an Elasticsearch *index* as well as monitoring data.

Elasticsearch and Kibana are used not only in High Energy Physics (HEP) community, such as KEK, CERN, CC-IN2P3 but also in the other fields, such as Facebook, GitHub, Stack Exchange, and so on. Although these tools provide a useful monitoring platform and are used in many places, they do not support authentication and authorization features by default. This means any user can access all data in Elasticsearch. However, in the case that data itself or Kibana objects should be shared with limited users or personalized in some extent, fine-gained access control is necessary.

In this paper, we provide a solution for secure use of Kibana and Elasticsearch in multi-user environment. This paper starts with an introduction of a solution of the CERN cloud group in Section 2. Section 3 describes our solution consisting of four steps. Section 4 describes performance degradation of secured Elasticsearch. Related work is shown in Section 5, and finally in Section 6, we provide a summary of the work.

2. Solution of the CERN cloud group

The CERN private cloud[3] has been in production since 2013. The cloud is based on OpenStack and configured to be integrated with the CERN network and authentication services. The cloud service enables a flexible provisioning of computing resources by utilizing virtual machines on demand for the CERN IT services, the batch service, the LHC experimental groups, and personal users. There are more than two thousands of registered users, and personal and group shared tenants on the cloud.

They had provided cloud utilization dashboards to each user by Elasticsearch and Kibana. On the dashboard, a user could check a usage history of personal tenant such as a number of active virtual machines, used CPU cores, RAMs, storage spaces. However, a user could also access others' dashboards which contain private information. Moreover, anyone could modify and delete all data in Elasticsearch by default. To solve the situation, they developed an Elasticsearch plugin to protect user's dashboard from others. By the plugin, each user could access only information of personal tenants belonging to. Figure 1 shows an overview of their solution. A web server runs in front of Kibana and Elasticsearch to authenticate a user by the CERN Single Sign On system. After the authentication, the server passes user's request to Kibana. The developed plugin intercepts a query from Kibana. Then it appends user-specific query condition based on username and cloud tenant ID that user belongs. For example, there is a user named *user01* who belongs to *tenant01* whose tenant ID is *123456*. If Kibana sends a query requested by user01, the plugin appends

PoS(ISGC2017)008

Table 1: Comparison between the CERN's and our solutions.

	CERN's solution	Our solution
Authentication	Shibboleth	Kerberos
Elasticsearch plugin	Homemade	Search Guard
Elasticsearch access control	User based; document level	User/group based; index, type, document,operation level
Kibana object separation	Per user	Per user/group enabled by a developed Kibana plugin

tenant_id=123456 query condition. Finally, Elasticsearch receives the modified query and returns only results matched the query.

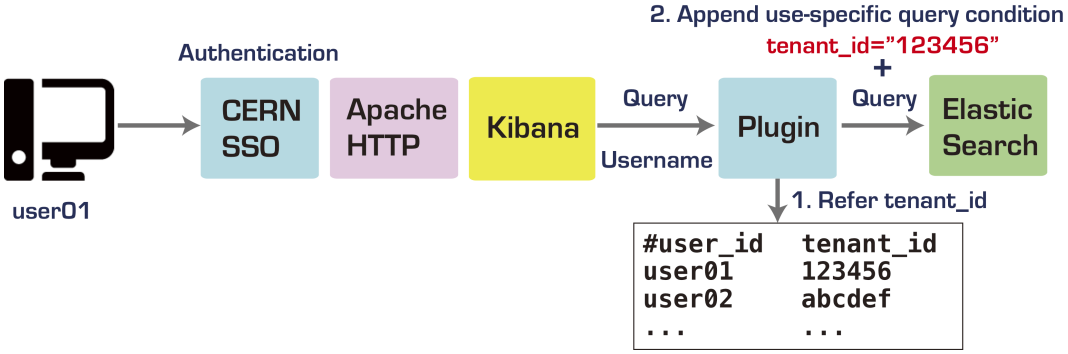

Figure 1: Overview of the CERN's solution.

3. Proposed solution

Although the CERN's solution enables access control based on user, all the data in Elasticsearch is required to have tenant ID information for the filtering. Furthermore, the solution is only for their cloud service, and it is difficult to adapt to the other use cases.

Based on their work, we propose a more flexible solution. It enables fine-grained access control on Elasticserch and Kibana objects separation based on user and groups. Table 1 compares the CERN's and our solutions. Ours integrates Kerberos 5 for authentication. Search Guard plugin for Elasticsearch security provides more flexible, various ways of access control than the plugin developed by the CERN cloud group. In addition, a developed Kibana plugin enables each user or group to have a tenant so that Kibana objects are saved to separated location from others.

Figure 2 shows an overview of our solution. Kibana and Elasticsearch run behind of a web server. Therefore these services only allow access from authenticated user. The developed Kibana plugin generates available Kibana tenant list based on user and LDAP groups. A user can switch a tenant on a web interface depending on the situation. All requests going to Elasticsearch are intercepted by Search Guard plugin. The plugin enables user and LDAP groups based access control on Elasticsearch. Apache Flume is a tool which collects system logs and metrics from

PoS(ISGC2017)008

servers and stores them to Elasticsearch. Our Flume patch enables to push data to Search Guard-enabled Elasticsearch over SSL/TLS connection.

Our solution can replace the CERN's solution and moreover be adaptable for the other various use cases by securing Kibana and Elasticsearch. The monitoring group in CC-IN2P3, Lyon, France, have tried to centralize their Elasticsearch and have investigated the way of access control on Elasticsearch. We collaborated with them under TYL-FJPPL (Toshiko Yuasa Laboratory France-Japan Particle Physics Laboratory) Comp_03 project, and a prototype of our solution worked well in CC-IN2P3 in February 2016. Also, a production platform has been under construction.

The following sub-sections describe each part of our solution.

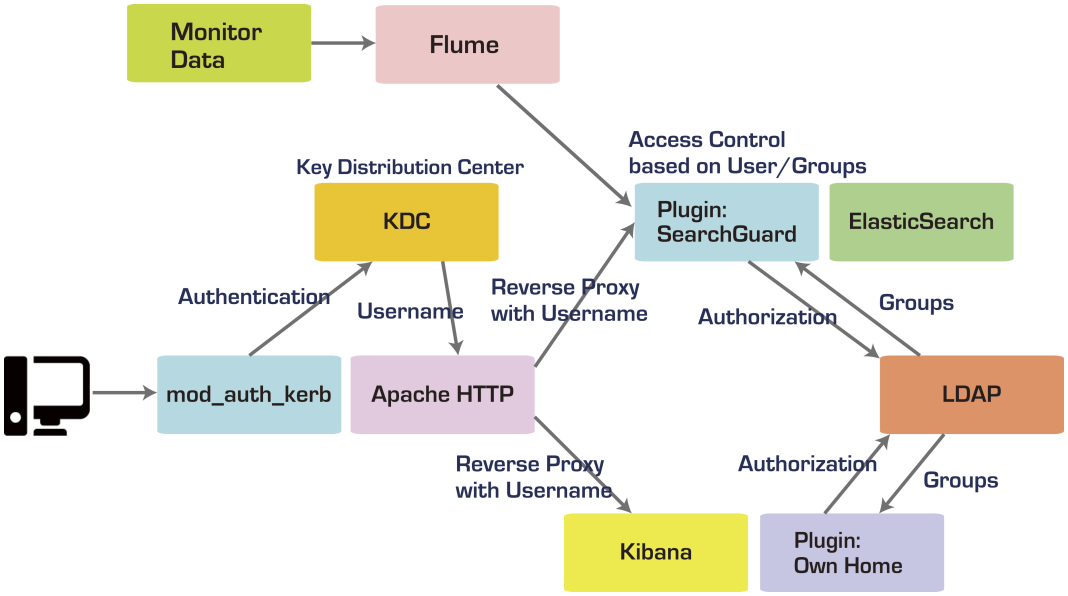

Figure 2: Overview of our solution.

3.1 Integration of Kerberos authentication

For the first step, we set up a web server using Apache HTTP in front of Kibana and Elasticsearch. The server is an only entry point to access the both services. We configured to use Kerberos 5 authentication on the server. The other authentication method is adaptable such as Basic, LDAP, Shibboleth by changing authentication module of Apache HTTP. After authentication, the web server set an authenticated username to HTTP request header and then passes a request to the behind service so that Kibana or Elasticsearch recognizes who requested.

3.2 Development of a Kibana plugin for multi-teancy

In the case of single Kibana instance shared among many users and groups, all Kibana objects (searches, visualizations, dashboards) are stored to the same Elasticsearch index which is called *Kibana index*. This means any user can access, modify, and also delete all objects in the index. In multi-user environment, to protect own objects from others or share part of objects among a group, Kibana index separation is one of the solutions.

PoS(ISGC2017)008

One idea is preparing Kibana instances as many as users. Each user has a dedicated Kibana instance configured to have a different Kibana index from the other instances. However, this idea does not scale. In the case that there are thousands of Kibana users, the same amount of Kibana instances is necessary.

The developed plugin named Own Home[4] adds multi-tenancy feature to single Kibana instance. The plugin enables a user to have own personal Kibana index so that objects the user created are stored to separate location from others. Furthermore, a group shared Kibana index can be provided. A user can switch Kibana index depending on his/her use cases. Available Kibana index list is generated based on username, LDAP groups, and file based definition.

Figure 3 shows a screen-shot of the plugin. A current selected Kibana index is stored in HTTP session. When a user switches Kibana index by clicking one of available Kibana indices, the plugin updates session to newly selected one.

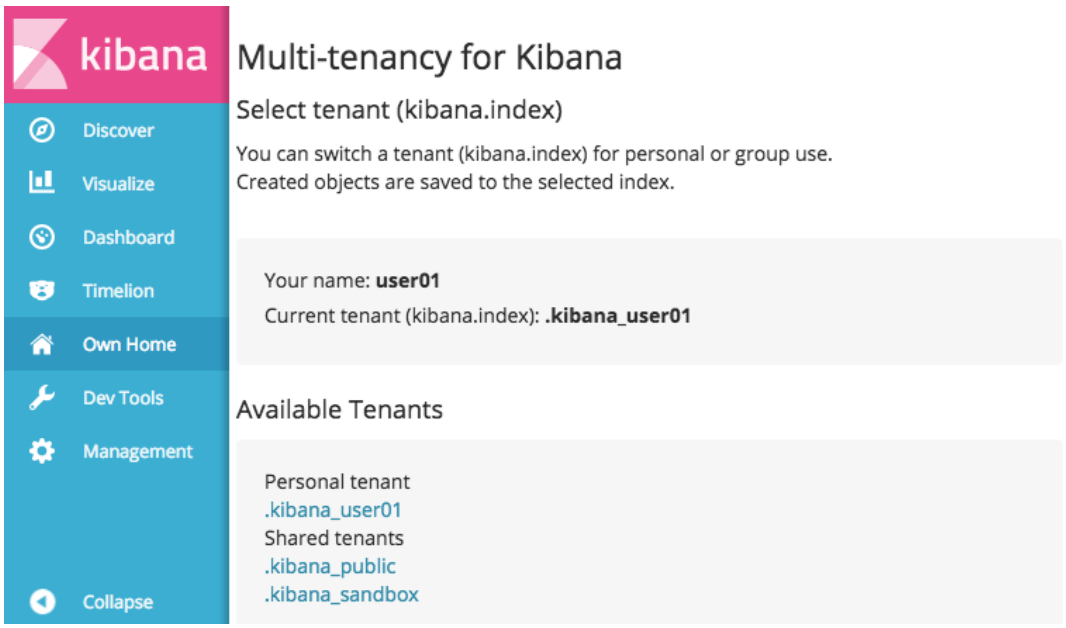

Figure 3: Screen-shot of Own Home.

Figure 4 shows how the plugin works. The plugin launches a proxy server on the same host of Kibana instance at the time of initialization. At first, a user selects a Kibana index from available index list on the web interface. Although Kibana always requests Elasticsearch to store Kibana objects into the single Kibana index, the proxy server intercepts the requests, then replaces the Kibana index with user's selected one which comes from HTTP session.

3.3 Configuration and contribution to Search Guard

The developed Kibana plugin just separates Kibana index and stores Kibana objects to a different location based on user/group. For the next step, it is necessary to set index level access control on Elasticsearch. For example, only *user01* can access *user01*'s Kibana index, or users belonging to *group01* can access *group01*'s index.

PoS(ISGC2017)008

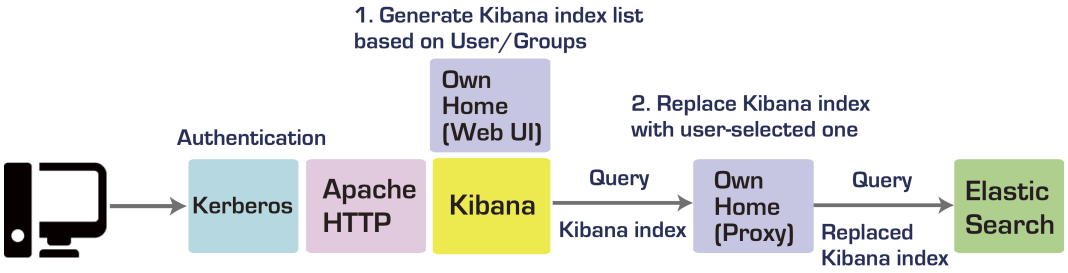

Figure 4: Workflow of Kibana index replacement.

Search Guard[5] is an Elasticsearch plugin for security developed by Floragunn. The software is open-source under Apache license. It provides flexible REST/Transport layer access control based on user/group, indices, types, documents, or cluster operations. Moreover, Elasticsearch node-to-node connections are encrypted for security. Search Guard supports multiple authentication and authorization backends such as Basic, Kerberos, proxy-based authentication, and LDAP authorization. In the case of our solution, authentication has been done at the web server in front of Elasticsearch. Hence, proxy-based authentication is selected as an authentication method. Also, LDAP authorization is configured for access control based on user and LDAP groups. An Elasticsearch admin can define access control list and push it to a highly secured Elasticsearch index. When a user accesses to data in Elasticsearch, Search Guard checks the user's permission by the list.

In addition, we have contributed to Search Guard community by proposing some code patches for more flexible configuration. All of the patches have been merged into the upstream code. The following describes one of the patches. In the case that each user has own Kibana index and each index allows access only from the owner, an admin has to define permissions for every user. Furthermore, whenever a new user is registered, the admin has to add permission and update the access control list. Our patch enables to use username variable in access control list, so access is controlled dynamically accessed user basis.

3.4 Development of an Apache Flume patch for SSL/TLS connection

Apache Flume[6] is one of the tools to push monitoring data to Elasticsearch. It enables to collect, aggregate, and move a large amount of data from many different sources to a centralized data store. Flume provides several kinds of *sinks* which put collected data to an external repository like HDFS. *Flume Elasticsearch sink* is one of them enabling to push collected data to Elasticsearch via plain text connection. However, the sink request is refused by Search Guard because Search Guard encrypts Elasticsearch connections and does not allow non-SSL/TLS requests. Our Flume patch enables to support SSL/TLS connection, and we confirmed the patched Flume could push data to Search Guard-enabled Elasticsearch.

4. Evaluation of Search Guard-enabled Elasticsearch performance

The previous section described securing Elasticsearch, and this section looks into its impact on performance. We measured Elasticsearch performance degradation caused by Search Guard

PoS(ISGC2017)008

Table 2: Part of data structure of the geographical dataset used in Rally.

Field	Type	Description
name	string	Name of geographical point
country_code	string	ISO-3166 2-letter country code
population	long	Population
longitude	double	Longitude
latitude	double	Latitude

using Rally[7]. Rally is a benchmarking tool for Elasticsearch developed by Elastic. It measures indexing throughput, query latency, aggregation latency, stats latency, and so on. Rally provides a few default benchmark scenarios, and also a user can define customized one.

In this paper, we used one of the default scenarios named *geonames*. In the scenario, Rally downloads geographical dataset (see Table 2) from a data source, then indexes the documents, total 2.8 GB, using eight client threads against a target Elasticsearch cluster.

We prepared two physical machines of AMD Opteron 6212 2.6 GHz 8 cores and 8 GB of RAMs with CentOS 7 and set up an Elasticsearch cluster on top of them. One of the machines also serves as a web server, Kerberos Key Distribution Center, LDAP server in the case of Search Guard-enabled environment. We used Rally 0.3.1, and Elasticsearch and Search Guard versions are 2.3.4. Then normal and Search Guard-enabled Elasticsearch performances were compared. For normal Elasticsearch measurement, Rally accesses Elasticsearch REST API directory. On the other hand, for Search Guard-enabled environment, Rally accesses a front end web server(see Figure 5).

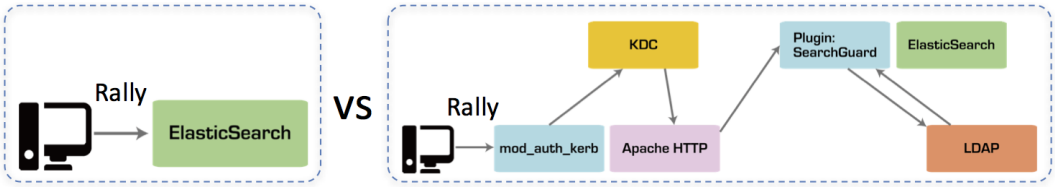

Figure 5: Elasticsearch performance comparison by Rally.

We measured indexing throughput and several kinds of query latencies. Figure 6-13 show the throughput or latency distributions of the measured benchmark result. The blue and red correspond to the normal and Search Guard-enabled Elasticsearch results respectively.

Figure 6 shows distributions of indexing throughput. Rally repeated to push 5000 documents per request and stored 8.6 million documents in total. The test measured the time from storing a document to making it searchable. Figure 7-13 show query latency results and 1000 queries were executed in each of the measurements. Distributions of seven kinds of query latency results are shown respectively. Table 3 shows descriptions of each of the queries.

Comparisons of medians between normal and Search Guard-enabled results are shown in Table 4. We observed 20% of performance degradation of index throughput in the case of using Search

PoS(ISGC2017)008

Table 3: Descriptions of queries.

Query	Description
Default query	Searches all documents.
Term query	Searches documents whose *country_code* equal to *AT*.
Phrase query	Searches documents whose *name* contain *Sankt Georgen*.
Aggregation query with/without caching	Sums up *population* grouped by *country_code* with/without caching.
Scroll query	Searches all documents, and fetches the results 1000 docs per page 25 times.
Expression query	Searches all documents, and evaluates each doc by calculation using *population*, *elevation*, and *latitude*.

Guard. Moreover, regarding the query latencies, there are certain differences depending on queries. Overhead of each query except the scroll is estimated as around 60 to 80 ms by the medians in Table 4, and overhead of the scroll query is about 130 ms. These degradations are caused by Kerberos authentication, reverse proxy, LDAP lookup, and Search Guard operations.

Normal Elasticsearch cannot be shared among users who have different purposes because of lack of access control feature. Secured Elasticsearch enables to provide a centralized service which can cover various use cases instead of launching Elasticsearch instance for each case. It leads to reduce maintenance cost, simplify system structure, and use limited compute resource effectively. However, it causes some performance deterioration. This paper provides a criterion of the degradation in secured Elasticsearch environment. In the case that a user requires higher indexing throughput and lower latency than Search Guard-enabled environment, setting up his/her dedicated Elasticsearch cluster is one of the solutions instead of using shared Elasticsearch among others.

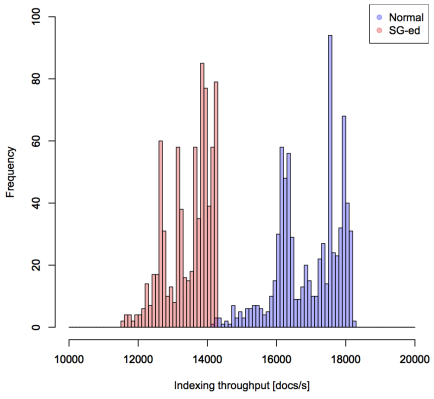

Figure 6: Distributions of indexing throughput

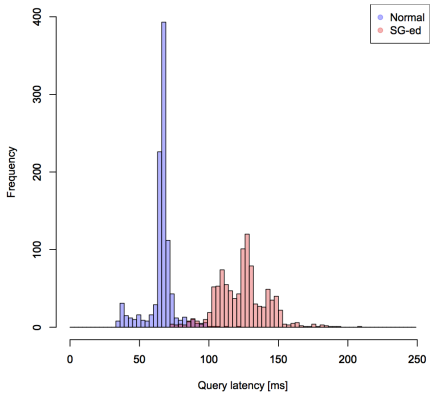

Figure 7: Distributions of default query latency

PoS(ISGC2017)008

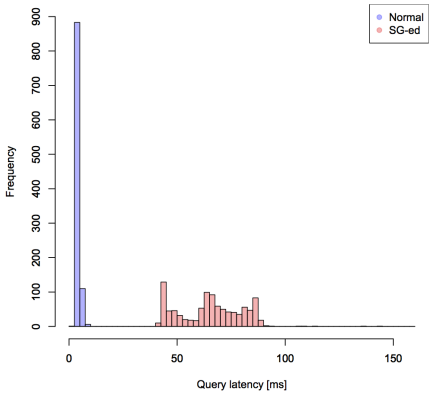

Figure 8: Distributions of term query latency

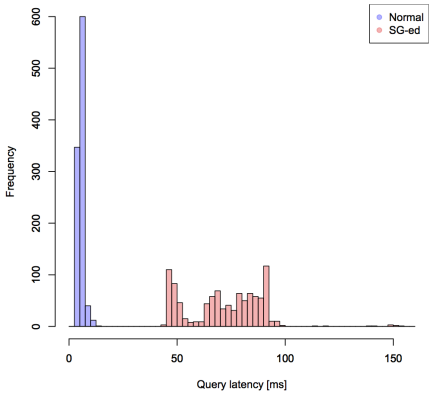

Figure 9: Distributions of phrase query latency

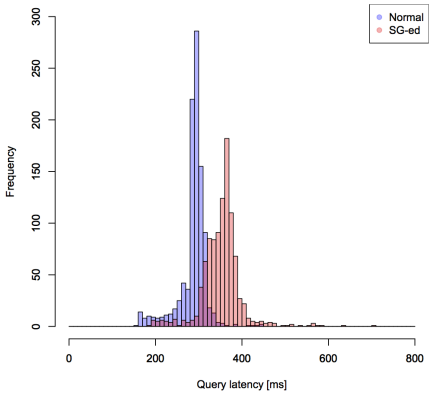

Figure 10: Distributions of aggregation without caching query latency

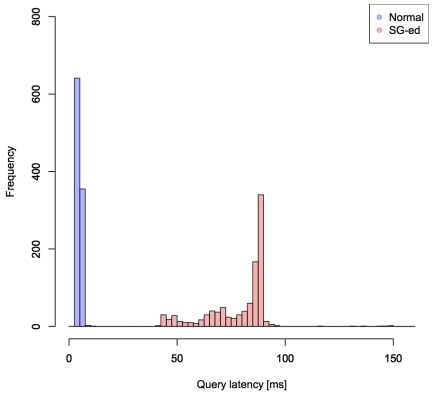

Figure 11: Distributions of aggregation with caching query latency

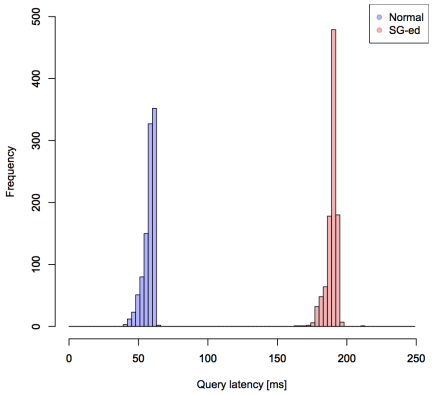

Figure 12: Distributions of scroll query latency

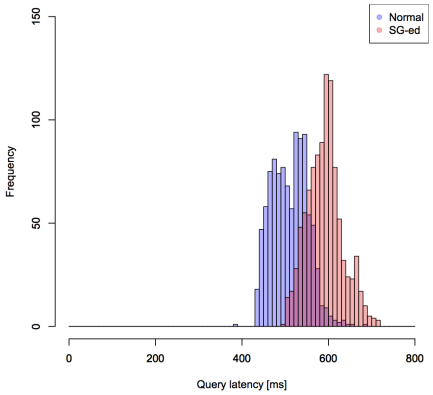

Figure 13: Distributions of expression query latency

PoS(ISGC2017)008

Table 4: Comparisons of medians between nomal and Search Guard-enabled Elasticsearch.

	Normal Elasticsearch	Search Guard enabled Elasticsearch	Perfomance degradation
Indexing throughput [docs/s]	17076	13659	3417 (20%)
Default query latency [ms]	67.0	125.0	58.0
Term query latency [ms]	3.8	65.5	61.7
Phrase query latency [ms]	5.4	73.4	68.0
Aggregation without caching query latency [ms]	292.1	357.4	65.3
Aggregation with caching query latency [ms]	4.5	86.2	81.7
Scroll query latency [ms]	58.9	190.3	131.4
Expression query latency [ms]	510.3	592.0	81.7

5. Related work

Bagnasco et al. [8] set up a web server in front of Kibana in order to authenticate and authorize a user at the web server level. However, They did not mention about access control on Elasticsearch. Furthermore, all Kibana objects are accessible from authenticated users just by setting up a proxy server.

There is another Elasticsearch plugin for security named X-Pack Security (formerly known as Shield)[9] maintained by Elastic. It provides IP filtering, authentication, authorization, node encryption, and auditing. The plugin is not only for Elasticsearch but also well integrated with Kibana. On Kibana interface, an admin can define users and roles for access control. The plugin is a commercial product and also 30 days free trial license is available. It provides access restriction features almost the same as Search Guard. However, Search Guard provides additional features such as OpenSSL support, Kerberos support, HTTP Proxy authentication support, JSON Web token support. On Kibana side, X-Pack Security does not provide multi-tenancy feature such our developed Kibana plugin supports. Therefore, in the case of using X-Pack Security, all users still can access others Kibana objects.

6. Summary

For secure use of Kibana and Elasticsearch, this paper provides a solution which enables user/group based access restriction and Kibana object separation in multi-user environment. A web server authenticates a user and passes requests to backends. A developed Kibana plugin allows a user to switch Kibana index depending on the situation such as personal use or group shared use. Search Guard enables user/group based access control on Elasticsearch, and we have contributed to Search Guard community for more flexible configuration. A developed patch for Apache Flume enables SSL/TLS connection so that Flume pushes data to Search Guard-enabled Elasticsearch. We measured the effect on the performance of Search Guard environment. The result shows that

PoS(ISGC2017)008

indexing throughput performance is degraded 20%, the overhead of each query except the scroll is estimated as around 60 to 80 ms, and the scroll query overhead is about 130 ms. It would be acceptable in an interactive use of monitoring system.

References

[1] "Elasticsearch: Restful,distributed search & analytics | elastic." `https://www.elastic.co/products/elasticsearch`.

[2] "Kibana: Explore, visualize, discover data | elastic." `https://www.elastic.co/products/kibana`.

[3] T. Bell, B. Bompastor, S. Bukowiec, J. C. Leon, M. K. Denis, J. van Eldik, M. F. Lobo, L. F. Alvarez, D. F. Rodriguez, A. Marino, B. Moreira, B. Noel, T. Oulevey, W. Takase, A. Wiebalck, and S. Zilli, *Scaling the cern openstack cloud*, in *Journal of Physics: Conference Series, vol. 664, no. 2, Clouds and Virtualization*, 2015.

[4] "wtakase/kibana-own-home: Multi-tenancy for kibana." `https://github.com/wtakase/kibana-own-home`.

[5] "Search guard | security for elasticsearch." `https://floragunn.com/searchguard`.

[6] "Welcome to apache flume." `https://flume.apache.org`.

[7] "elastic/rally: Macrobenchmarking framework for elasticsearch." `https://github.com/elastic/rally`.

[8] S. Bagnasco, D. Berzano, A. Guarise, S. Lusso, M. Masera, and S. Vallero, *Monitoring of iaas and scientific applicationf on the cloud using the elasticsearch ecosystem*, in *Journal of Physics: Conference Series, vol. 608*, 2015.

[9] "Security: Enterprise security for elasticsearch | elastic." `https://www.elastic.co/products/x-pack/security`.

PoS(ISGC2017)008