EPJ Web of Conferences **245**, 07021 (2020) https://doi.org/10.1051/epjconf/202024507021
CHEP 2019

Large Elasticsearch cluster management

Pablo Saiz[1,*] and *Ulrich* Schwickerath[1,**]

[1]CERN Esplanade des Particules 1, 1217 Meyrin Switzerland

Abstract. The Centralised Elasticsearch Service at CERN runs the infrastructure to provide Elasticsearch clusters for more than 100 different use cases. This contribution presents how the infrastructure is managed, covering the resource distribution, instance creation, cluster monitoring and user support. The contribution will present the components that have been identified as critical in order to share resources and minimise the amount of clusters and machines needed to run the service. In particular, all the automation for the instance configuration, including index template management, backups and visualisation settings, will be explained in detail.

1 Introduction

Elasticsearch [1] is a non-SQL storage solution, optimized for fast searches in text documents. Thanks to its documentation and easy setup, it is relatively easy to deploy a testing instance on a single node. To take advantage of the features offered by the system, the standard configuration is to install a cluster, that offer security, data replication, parallel access to the data and high availability.

The CERN centralised Elasticsearch Service was created in 2016. The goal was to concentrate all the independent Elasticsearch clusters that had been created, and unify them, with the idea of:

- Consolidating the resources needed for the clusters.
- Ensuring that all clusters are installed, configured and maintained following the security recommendations.
- Creating a support unit that can advise users on the best strategies to use the Elasticsearch components.
- Reducing the amount of effort and personnel needed to maintain all clusters.

After one year of prototyping the structure of clusters and its deployment, it became a production service in 2017. Since then, its usage has continued to increase, with more clusters, more users and more resources. At the time of writing this article, there are more than thirty clusters, a hundred and fifty instances and four hundred nodes. At the same time, after the service became production, the amount of human effort needed for the maintenance of the clusters has even decreased, thanks to the automation of recurrent tasks, like for example rolling minor upgrades, Elasticsearch restarts or kernel upgrades.

*e-mail: pablo.saiz@cern.ch
**e-mail: ulrich.schickerath@cern.ch

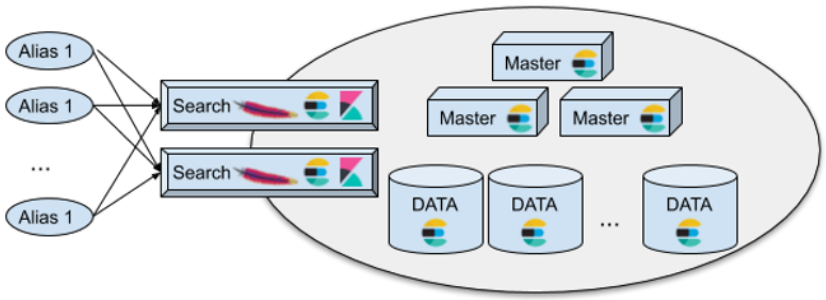

Figure 1. Cluster structure

This paper will present how the service is handled. The next chapter describes the structure of the Elasticsearch clusters provided by the service, including the different components that are installed. The next chapter will describe the tools used for monitoring all the clusters. After that, the paper presents the way customers request clusters, and the options offered for configuration and maintenance. Finally, a recap of the main lessons learned will be given, followed by the summary of this article.

2 Cluster structure

The structure of the Elasticsearch clusters provided by the CERN Elasticsearch service, and in particular the security setup, has been described in detail in [2]. As a summary, all the clusters have a common structure, with three types of nodes: data, master and search nodes. Clusters do differ on the number and size of nodes. Clusters might also have some specific settings. For instance, some of the clusters are visible from a different network used only by the accelerators. Other clusters provide a two-tier level of storage, with a first tier of SSD nodes for recent data, and a second tier with file servers with spinning media for longer term storage.

The vast majority of the nodes are virtual machines running on OpenStack [3]. This way, the nodes can be sized according to their needs. The only exceptions are the physical nodes used as file servers with spinning disks for clusters that require large amounts of data. This will be explained in more detailed in the section about data nodes. All the nodes are managed by Puppet [4].

2.1 Nodes

There are three main types of nodes, as depicted in Figure 1.

2.1.1 Search nodes

The search nodes are the entry point to the clusters. They do not provide storage, and they are the only nodes with open ports to access the cluster. The authentication and authorization is handled here. These nodes run three services:

- Elasticsearch instance, listening on a local port, and neither providing storage, nor master services. Elasticsearch contains the ReadOnlyRest plugin [5], which handles the authorization. It is worth noting that the rest of the Elasticsearch nodes do not have any authorization plugin installed.
- Kibana [6] visualization on top of Elasticsearch, listening on a local port.
- Apache reverse lookup service, with virtual hosts depending on the alias. Apache is configured to offer multiple authentication mechanisms, and, if the authentication is successful, it forwards the call to Elasticsearch and Kibana, where the authorization will take place. By default, there are three authentication mechanisms provided: user/password, Kerberos and the Single Sign On (SSO). Some setups include as well ip-based authentication.

2.1.2 Master nodes

The master nodes are responsible for the management of the cluster. Only one of the masters is active at any point in time. The others are used when the active master is not responsive. At that moment, they have an election system to promote a new active master. This promotion requires that more than half of the master nodes are connected, thus avoiding split-brain issues. For these reasons, the clusters provided by the Elasticsearch system have three master nodes, located on different availability zones. These nodes require little disk space, memory and CPU power.

2.1.3 Data nodes

The last type of nodes are the ones storing the data. These nodes do require fast storage and enough memory, since they will do the bulk of the operations of the cluster. According to the documentation of Elasticsearch, the optimal amount of memory that should be allocated to Elasticsearch is 32 GB, since it is the maximum that can be used still with short pointers. As mentioned above, most of the nodes are virtual machines. The physical nodes hosting them are 40 core, 128 GB of RAM and 3.6 TB of disk space. These nodes are split into 20-core virtual machines, or even 10-core virtual machines for smaller clusters.

Two of the clusters supported by the Centralised Elasticsearch Service need to store a large amount of data, in the order of a hundred terabytes. The use case is to keep a terabyte of daily logs over the last three months. Once the daily indices are written, there are no further updates. From that moment, it only requires read access. The cost of providing the full cluster on SSD is too high, so the structure that is used for these two clusters is a hot-warm architecture, as described in [7]. In order to optimise memory allocations, the file servers were configured to have multiple docker containers, each one with a fraction of the storage available on the node.

2.2 Plugins

The required Elasticsearch and Kibana plugins are packaged into RPMs and installed with Puppet. The following list presents some of them. The first two are compulsory, since they provide part of the security of the system. The last two plugins have been developed at CERN.

- ReadOnlyRest [5]: provides the authorization on Elasticsearch. It is configured specifying the actions that users are allowed to do on different indices. The latest versions provide document level security.
- OwnHome [8]: Kibana plugin that allows to have multiple decoupled customers on a single instance of Kibana.

- Kibana logout button [9]: icon to Kibana to stop the current session.
- Kibana relational filter [10]: adds a visualization to Kibana, which allows to add filters based on different indices.

3 Monitoring

3.1 Internal service monitoring

One of the first clusters created by the service was dedicated to the performance monitoring of all the other clusters. This was done for two reasons:

- Centralizing the monitoring of all the clusters, having a single dashboard that can show the status of all the clusters.
- Gaining experience on the management and usage of the Elasticsearch components. This cluster was deployed using the same tools that are used for all the others. In addition to Kibana and Elasticsearch, this cluster also requires the configuration of Logstash [11] and Filebeat [12] to collect the data. Thanks to this, the service managers have the opportunity to become familiar with the rest of the Elastic stack.

The monitoring information gathered in this cluster contains the log files of the services (Apache, Elasticsearch, Kibana and Logstash), multiple metrics provided by the Elasticsearch interface (status of the cluster, number and size of indices and shards, thread pool, garbage collector, ...) and accounting information of the requests and usage of the clusters.

On top of this internal monitoring cluster, several aliases have been created that filter the information and present to the users dashboards and visualizations containing only their clusters.

3.2 Anomaly detection

The monitoring system presented here follows up a high number of metrics over a high number of clusters. It is a challenge to present a single dashboard with the status of all the clusters. To tackle this, a machine learning based anomaly detection system has recently been put in place to make it easier to identify any issues with the clusters, and detect them as soon as possible.

The system is based on an unsupervised LSTM [13] based network. One day of data of selected input features from the internal monitoring system is used to predict the current state for each cluster. This is then compared to the actual values of the input features. The mean squared error between the prediction and the actual measurement is used as anomaly score. The features with the largest contribution to the anomaly score can be used to classify the type of the anomaly.

While still being actively developed and improved, this system has already been very successful in spotting issues before users became aware of them.

4 Cluster creation

The process to create new clusters and instances has been streamlined. First of all, the user fills in a form with the desired parameters, like the name of the cluster, size, permissions and version of Elasticsearch. If the requests seems reasonable, the service managers decide if it can be added as a new use case in one of the existing clusters, or if a new cluster has to be created. The former has the advantage that the resources have already been allocated. The

latter has to start by installing a new cluster (three masters, two search and at least three data nodes).

Once the cluster has been created, the next action is to create a new DNS alias that will point to its search nodes. At this point, the permissions that are allowed through the alias are also defined, including the groups that can access Kibana and Elasticsearch, with the possibility of dividing them among users with read access, or users with read-write access.

If the users prefer to connect with an username and password, the credentials will be sent to them in an encrypted email.

Finally, a Gitlab repository for the user is also created, where they can further configure their cluster. The settings that can be configured are described in the next section.

4.1 Cluster settings

The customer receives a Gitlab repository to configure some parameters of the cluster. Among these parameters are:

- The index templates that apply to newly created indices. Users are only allowed to put templates that apply to their own indices. If the template applies to other indices, it will receive an unauthorized return. At the same time, one standing issue is that all the templates of a cluster are stored on the same place, and, if two users of the same cluster specify the same template name, they will overwrite each other. At the moment, this item has not been addressed, and the service counts on the users to upload templates with unique names.
- Curator rules [14]. Elastic provides a set of tools to maintain indices. Using curator, old indices can be closed, deleted or moved to a different node type. It can be used as well to modify the aliases for indices. Thanks to this, it is very easy to provide, for instance, an index with a suffix of '_today' that contains the latest data, without having to re-index the data.
- Landing page for Kibana. The user can define which will be the default dashboard that Kibana should open.

5 Lessons learned

During these years, different setups and configurations have been examined. This is a list of some of the most important lessons that have been learned while running the service:

- The speed of the cluster is usually limited by the slowest data nodes. This means that clusters with heterogeneous data nodes get usually limited by the slowest nodes. For this reason, all the data nodes within the same cluster have the same specs.
- The data nodes need fast access to storage. Spinning media can be a bottle neck. An SSD cache in front of the spinning media helps the situation for specific use cases. This requires additional scheduling policies to ensure that fresh data goes to the fast cache, and old data gets migrated to slower disks. A setup where the full data is on SSD only is of course better and simpler, but also more expensive.
- The performance of the cluster does not scale linearly with the number of nodes. Increasing the number of nodes increases significantly the management tasks for the master, and the recovery time in case of issues or interventions.
- Similarly, a high number of indices and shards increases the complexity of the cluster. Since the amount of indices and shards are defined by the users, this has to be discussed and followed up with them.

- Providing a full isolation of users is very difficult. The clusters offered by the CERN Centralised Elasticsearch system do have isolation on the indices that are visible, ensuring that users cannot access each others data. At the same time, the users do share the same boxes, and there are no restrictions or CPU quotas per user. If one of them has a heavy load on the cluster, all the other users of the same cluster will be affected.
- Upgrades on clusters used by different users can be complicated. For mayor releases, instead of updating clusters, the service managers chose to create a second cluster, so that users could migrate at different moments. At the same time, this usually requires copying data from one cluster to another, which can be time consuming.
- The behaviour of clusters can change drastically depending on the user patterns, and it should be closely monitored.

6 Future work

A recent change in the field has been the creation of the open distro Elasticsearch [15], an open-source branch that took the open components from Elastic, and build on top of that the other commercial Elastic components: monitoring and alarming, authentication and authorization, and SQL interface. Switching from the current scenario, where the system is based on open-source components maintained by different groups, to the OpenDistro, where everything is maintained by the same team, is very appealing. At the same time, it would imply an important change of the deployment and configuration. The benefits of this solution should be evaluated, and compared with the current structure.

The anomaly detection system is being further developed and optimised, identifying the key source metrics that can indicate anomalies in the cluster, thus making it easier to understand the cause of issues in less time.

7 Summary

This paper has presented the CERN Elasticsearch centralised service. Four years after being created, the service currently provides more than a hundred and fifty endpoints over thirty shared clusters, with more than four hundred nodes. The paper described how the clusters are deployed, maintained and monitored. Given the limited number of personnel dedicated to maintaining the service, extra effort has been put in automating recurrent tasks and monitoring. On top of that, an artificial intelligence system to detect anomalies has been deployed and it is constantly being improved.

References

[1] Elasticsearch [software], versions 5.5.1, 5.6.9, 6.2.4. (accessed 2020-03-10)

[2] U. Schwickerath, P. Saiz, Z. Toteva, EPJ Web Conf. **214** *08032* (2019)

[3] OpenStack Open Source Cloud Computing Software [software], version Ocata. Available from *https://openstack.org*, [accessed 2020-03-10)

[4] Puppet Labs: IT Automation Software for System Administrators [software], version 4.9.4, 2018. Available from *http://puppetlabs.com*, (accessed 2020-03-10)

[5] ReadonlyREST: Security for Elasticsearch and Kibana [software], version 2.18.1. Available from *https://github.com/sscarduzio/elasticsearch-readonlyrest-plugin*, (accessed 2020-03-10)

[6] Kibana: Your Window into the Elastic Stack [software], versions 5.5.2, 5.6.9, 6.2.4. Available from *https://www.elastic.co/products/kibana*, (accessed 2020-03-10)

[7] "Hot-Warm" Architecture in Elasticsearch 5.x, available from *https://www.elastic.co/blog/hot-warm-architecture-in-elasticsearch-5-x*, (accessed 2020-03-10)

[8] Kibana Own Home plugin by Wataru Takase [software], versions v5.5.2, v5.6.9, v6.2.4, Available from *https://github.com/wtakase/kibana-own-home*,(accessed 2020-03-10)

[9] Kibana logout plugin [sotfware], available from https://github.com/cernops/kibana-logout-plugin, (accessed 2020-03-10)

[10] Kibana filter plugin [software], available from https://github.com/cernops/kibana-relational-filter, (accessed 2020-03-10)

[11] Logstash [software], available from https://www.elastic.co/downloads/logstash, (accessed 2020-06-24)

[12] Filebeat [software], available from https://www.elastic.co/downloads/beats, (accessed 2020-06-24)

[13] S. Hochreiter, J. Schmidhuber, LONG SHORT-TERM MEMORY, NEURAL COMPUTATION **9(8)**, *1735-1780* (1997)

[14] Curator Reference, available from *https://www.elastic.co/guide/en/elasticsearch/client/curator/5.8/index.html*, (accessed 2020-03-10)

[15] OpenDistro for elasticsearch, available from *https://opendistro.github.io/for-elasticsearch*, (accessed 2020-03-10)