International Journal of Information Technology (IJIT)
Volume 4, Issue 02, July-December 2023, pp. 1-9 Article ID: IJIT_4_02_001
Available online at https://iaeme.com/Home/issue/IJIT?Volume=4&Issue=2
Journal ID: 4573-3410, **DOI:** https://doi.org/10.17605/OSF.IO/F4UTS

SOLVING COMPLEXITY AND IMPROVING STORAGE EFFICIENCY IN MANAGING ELASTICSEARCH CLUSTERS USING KUBERNETES AND ECK OPERATORS

Amreth Chandrasehar
Informatica, CA, USA

ABSTRACT

Elasticsearch is a popular open-source tool based on Apache Lucene, widely used for search, storing large amounts of data. It is being used in many small startups to large enterprises for Full-text search, log analytics, application search, geospatial data analysis and very recently also used in SIEM. As business needs grows, the Elasticsearch cluster needs to be scaled and often, the operators of Elasticsearch face many complex issues to manage the cluster. In this article, solution is provided on how to reliably manage the clusters at petabyte scale. Also, key information on monitoring, alerting and autoscaling the clusters are provided for the operators of the clusters to easily manage large scale Elasticsearch clusters. This paper aims to provide key information to the administrators and users of Elasticsearch clusters to reliably use to handle large volumes of data and possibly used as a Data Lake. It will also help organizations to have better financial controls, operational efficiency and do more with less by adopting key solutions discussed in this paper.

Keywords: Elasticsearch, ELK, Elastic Stack, Observability, SIEM, Machine learning, Kubernetes, ECK Operator, Data Lake, Cost Control

Cite this Article: Amreth Chandrasehar, Solving Complexity and Improving Storage Efficiency in Managing Elasticsearch Clusters Using Kubernetes and Eck Operators, *International Journal of Information Technology (IJIT)*, 4(2), 2023, pp. 1-9 https://iaeme.com/Home/issue/IJIT?Volume=4&Issue=2

1. INTRODUCTION

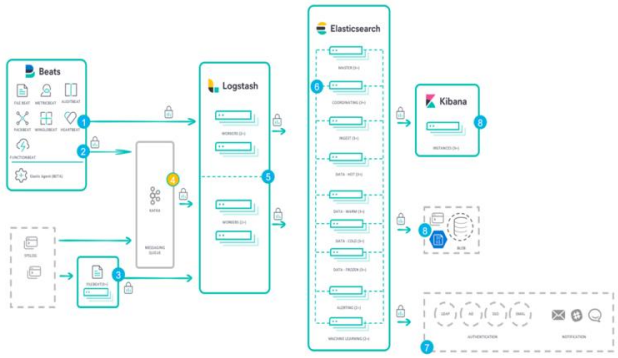
Elasticsearch administrators, operators face many challenges to maintain a large Elasticsearch cluster. Upgrades, certificate rotation, OS patching, high availability and fault tolerance of data or even enabling TLS encryption can be very complex task in environment with 100's of nodes. Misconfigured Elasticsearch clusters can cause data leaks, running obsolete or EOL versions as security fixes and upgrades cannot be done without taking a downtime or operators not willing to take the task as it is very complex.

The organizations are spending a lot of time and resources into managing Elasticsearch clusters when it is not optimally architected and deployed. This causes a lot of issues such as poor security, outages, downtimes during upgrades, unable to serve requests, increased latency and over provisioning of hardware which cost a lot of money. Determining the right configuration is key to ensure the cluster performs optimally.

2. INFORMATION ON CLUSTER COMPONENTS

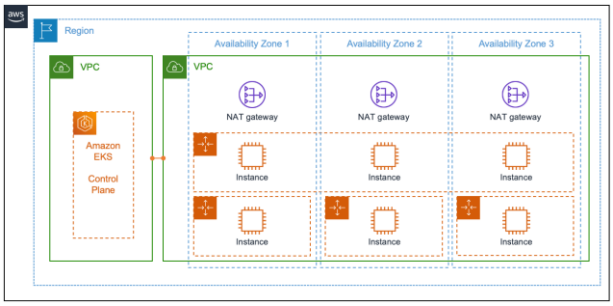
2.1. Elastic Stack

Elastic Stack comprises of Elasticsearch nodes (master, data, client and ingest), Logstash, Beats (file beat, metric beat, audit beat, winlog beat), Kibana, APM (server and agent) and Elastic Agent.



Block diagram showing the interworking between different Elastic Stack components [4]

In a large environment ingesting terabytes or petabytes of data everyday will require 100's of data nodes to store these large amounts of data. In a cloud environment, the nodes need to spread in multiple availability zones to ensure high availability, prevent data loss and also to meet compliance and regulatory requirements.



Cluster add-ons and application workloads share the data plane [5]

Maintaining large number of nodes needs more configurations, automations scripts, provisioning resources, monitoring, scaling, and ensuring high availability. Kubernetes, an open-source container orchestration platform, offers a powerful framework for automating these tasks. It also offers a declarative approach, ensuring desired cluster state and automatic healing of components. Kubernetes also gives unparalleled scalability to organizations with its auto scaling tools like VPA and HPA. Operations can grow on-demand during peak demand without having to set up and invest in the infrastructure. Kubernetes simply allots new resources to compensate. Conversely, Kubernetes can re-assign resources to other applications when demand dies down. This can help make your operation incredibly efficient by minimizing wasted resources.

AWS EKS blueprint [6]

3. HELM VS KUBERNETES OPERATOR

As Kubernetes provides great capabilities, we can run Elasticsearch clusters on Kubernetes to simplify the overall process of configuring, managing, and scaling Elasticsearch clusters. Also, Elasticsearch can be deployed on multiple cloud environments and on-premises, using the various automation features available on Kubernetes. There are now two ways to deploy Elasticsearch on Kubernetes, first is using HELM charts or second option is to use ECK to configure all required components.

Helm is a package manager designed specifically for Kubernetes and improves the management of the YAML manifests required to create Kubernetes projects. But when applications such as Elasticsearch requires detailed configurations, it becomes difficult to express in YAML files using Helm charts.

Kubernetes Operators are geared toward site reliability engineering teams to manage the complex runbooks that orchestrate the deployment of a complex application, along with automating mundane tasks for the Kubernetes platform. Kubernetes operators build on custom resources and controllers. In plain terms, an IT admin can define their resources in Kubernetes and publish logic -- via an operator -- that can handle CRUD operations for the custom resource. These custom resources can model a complex application or a standard templatized application; the operators maintain the resource's lifecycle. [2] Hence choosing Elastic supported ECK operator is a better choice to manage Elasticsearch clusters in Kubernetes.

ECK stands for Elastic Cloud on Kubernetes; it uses Kubernetes Custom Resource Definitions (CDRs) and provide native Kubernetes operator to make it easier to deploy and maintain Elasticsearch clusters. Beyond deployment, ECK also provides general management capabilities for Kubernetes. It can help with managing multiple Kubernetes clusters, upgrading the Elastic stack, monitoring, Configuration Management and Secret Handling, cluster capacity expansion and reduction, Dynamic Scaling and Resource Allocation, cluster configuration changes, backups, and dynamically expanding local storage (including Elastic Local Volume).

4. APPROACH TO CREATE A SCALABLE ELASTICSEARCH CLUSTER USING ECK OPERATOR.

Elastic Stack consists of Elasticsearch, Kibana, APM Server, Enterprise Search, Beats, Elastic Agent, Elastic Maps Server, and Logstash. With Elastic Cloud on Kubernetes (ECK), the Elastic Stack can be configured to deploy all the components and other critical operations can be streamlined, such as: Managing and monitoring multiple clusters, Scaling cluster capacity and storage, performing safe configuration changes through rolling upgrades, scheduling backups, securing clusters with TLS certificates and setting up hot-warm-cold architectures with availability zone awareness.

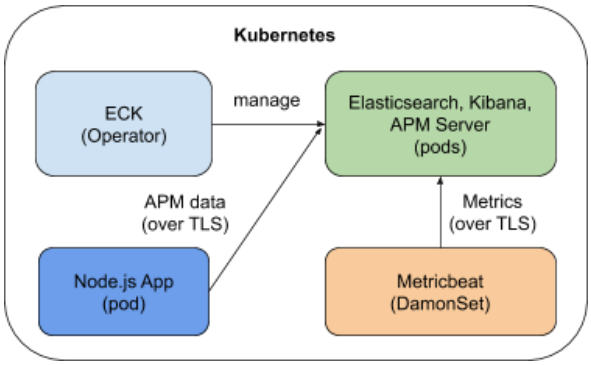


Image showing ECK operator managing Elastic components [7]

4.1. Considerations before starting to provision the environments

- Sizing of Elasticsearch cluster - Ingest volume, number of read requests to factor replication and parsing the data.
- Choose right infrastructure resources - Instance types, network throughput, block storage for hot tier and Object storage for frozen tier and backup. In AWS environments, i Series instance types are usually used as it provides instance storage, but Graviton nodes such as m7g and r7g instance types can also be used with storage provisioned using EBS volumes
- Provision right amount of disk storage, this can be scaled up as needed. Once the disk has been provisioned, it cannot be scaled down due to limitations of EBS volumes from AWS and other cloud vendors.
- Choose right Kubernetes components such as CNI, EBS CSI driver, cluster autoscaler, loadbalancer controller, kubeproxy, CoreDNS and its versions
- Consider GitOps to provision Kubernetes, ECK operator and Elastic Stack. Tools such as Flux and ArgoCD can be used to manage the resources effectively without manual interventions. This is very much required to manage large Elasticsearch clusters
- For high availability, consider using at least 2 availability zones. Using 3 AZs can become expensive due to Inter-AZ network costs. Noncritical or non-production clusters can be deployed in single AZ to eliminate Inter-AZ data transfer costs.
- Elasticsearch nodes consist of Master, data, ingest and client nodes. For high availability, consider using 3 Master nodes, dedicated client nodes to handle traffic from APIs and Kibana, data and ingest nodes can be coupled together to save cost.
- Configure MTLS to secure transfer of data from client to server.

4.2. Best practices and cost savings configurations post Elasticsearch cluster is provisioning

Below are the best practices to be followed to operate a large scale Elasticsearch cluster. If rightly followed, this can save millions of dollars in infrastructure costs and scale the cluster to ingest petabytes of data.

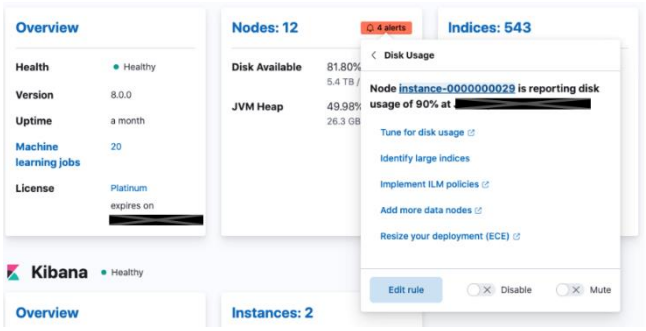
- To transform data, use Kibana ingest pipelines or use dissect filter in filebeat. Logstash consumes a lot of resources, avoiding data transformation in logstash can save cost and maintenance tasks.
- Drop unused logs to save storage, network costs
- Configure backup at least once a day to object storage like S3 to restore data in case of cluster failure
- Use streaming service such as Kafka to handle backpressure from Elasticsearch and to avoid data loss, restore logs quickly in case of cluster failure.
- Configure more than 1 primary shard for indexes with higher ingestion rate. This will avoid performance bottleneck
- Ensure 1 primary shard per index is configured, this will ensure traffic is evenly distributed on all hot/ingest nodes.
- Use LZ4 compression supported by Elasticsearch to optimally store logs
- Use hot and frozen tiers. If possible, avoid warm and cold tiers as it leads to log of Inter-AZ network costs
- Remove unused fields. Elastic provides field usage stats in indexes. This can be used to automate removal of unused fields automatically.
- Use Synthetic source for TSDB and for logs that are less used. This will save up to 65% in disk storage.
- Based on usage, purchase RI or savings plan to reduce compute costs.
- Use Private endpoints or VPC Peering or Transit gateway between VPCs to reduce up to 75% network costs if NAT gateway is used.
- Use API Gateway if access to Elasticsearch API is provided. This will help to throttle requests and save from cluster failures.
- Setup dedicated monitoring cluster to store audit logs and to monitor all Elasticsearch clusters in one place
- Use searchable snapshots feature to provision frozen tier. This will help to search logs beyond data available in hot tier.

5. MAINTENANCE OF ELASTICSEARCH CLUSTERS USING ECK OPERATOR

Cluster operations are necessary to ensure Elastic Stack, ECK and Kubernetes is running with latest versions to take advantage of new features and more importantly to secure the cluster. The worker nodes should ensure it is running in latest OS version, this will require rotation of nodes every month or at least once a quarter.

Since there are numerous moving parts, it is recommended to use GitOps model to provision and update the configuration files. ECK will ensure the pods are rotated one at time, applying the configurations automatically. This will avoid manual interventions and cluster failures as ECK will perform failure recovery as needed.

Monitoring setup is very important to ensure the operators and SRE personals are alerted if there are any cluster failures. ECK automatically provides monitoring of Elastic Stack, below is an example from monitoring cluster.



Sample view from monitoring cluster and Kibana alerts [3]

Alerts that need to be configured for operating clusters successfully are:

a. SSL Cert Expiry – To ensure client, server certs are rotated before expiry and avoid log loss
b. URL and to track uptime of endpoints
c. Synthetic monitoring to automatically replicate usage patterns
d. Cluster Alerts - Disk Usage, Heap usage, Cluster State, pending tasks, relocation shards, index not rotated, no new documents ingested in indexes and thread pool rejection
e. Kubernetes Alerts – Pod not running, daemon set rollout stuck, job failure, node not joined cluster

6. USING TECHNIQUES TO OPTIMIZE DISK STORAGE

Below are techniques to optimize Disk storage.

- Use default Filebeat module mappings
- Reducing the number of fields which are not required. This information can be got by using field usage stats API “GET /my-index-000001/_field_usage_stats”
- Use best_compression instead of LZ4 compressin which comes by default.
- Remove _source field if not required. This comes at an expense of fields not discoverable at search, repair index automatically if corrupt

For example, if you're expecting to ingest 5 TB of structured log data per day and store it for 30 days, you're looking at a difference between 83 and 168 TB in total storage needs when comparing the mappings with minimum vs. maximum storage needs. Depending on other factors which will help define how much data you can host on each node while maintaining reasonable query performance, this could mean 20-30 extra nodes. And that's not even considering replication. [8]

7. STORAGE OPTIMIZATION RESULTS

For these tests, file containing 280000 sample Apache2 access log records were indexed using the Filebeat Apache2 module.

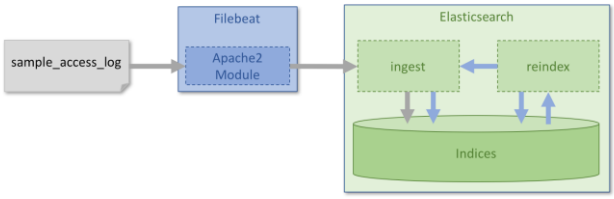


Figure representing how filebeat module is used to ingest logs optimally

Once all the test indices had been created, we used the force merge API to reduce all indices down to a single segment per shard. This helps to reduce disk space required to store the data and also to ensure the comparison between index sizes is fair as all the indices have been optimized to the same level.

_all field and _default mapping has been removed in 7.0, the test result has been replaced with _source as it provides similar results.

Test	Reduced field count	Best compression enabled	_source field disabled	Raw data size (MB)	Enriched and indexed data size (MB)	Size compared to raw data (%)	Saving vs Elasticsearch defaults (%)	Saving vs Filebeat defaults (%)
1	No	No	No	60.194	95.360	158.4	19.7	N/A
2	Yes	No	No	60.194	78.886	131.1	33.5	17.3
3	No	Yes	No	60.194	76.575	127.2	35.5	19.7
4	No	No	Yes	60.194	70.475	117.1	40.6	26.1
5	Yes	Yes	Yes	60.194	47.838	79.5	59.7	49.8

The full results of all the tests are shown in the table above [9]

Summary of the test results:

- The 280000 sample records with default dynamic Elasticsearch mappings took up 118.713 MB bytes on disk while the optimized index took up only 95.360 MB bytes. That is a substantial 19.7% space saving.
- If we instead compare how much space these enriched events take up on disk compared to the raw data (60.194 MB), we can see that the index with default Filebeat module settings takes up 158.4% of the raw data size.
- Reducing the number of fields this way resulted in a space saving of 17.3% compared to the default Filebeat modules index. Full results will be presented at the end of this blog post.
- Using best_compression made a significant impact on disk space and saved 17% saving compared to the default compression. This additional compression does however not come for free, and uses up more CPU, particularly during indexing, which can negatively impact indexing performance. It may therefore not be the right option for all users, which is why it is not used by default.
- To evaluate the impact disabling the _source field can have, we reindexed the data into a test index with standard Filebeat module mappings, apart from the _all field being disabled. This resulted in an impressive 33.3% saving compared to Filebeat defaults.
- As the various optimizations affect each other, the total gain is less than the sum of the individual tests, but still resulted in a reduction in storage size of almost 50% compared to the defaults. Comparing this to the raw file size shows that the indexed data only takes up around 80% of the raw size. If disk space is critical and you want to be even more aggressive, there is however still room for further improvement. [9]

8. FUTURE WORK

Elastic has released many new compression techniques such as 70% metrics storage savings with TSDS enabled integrations, synthetic source, disabling dynamic field mapping, disabling doc_values and excluding fields from _source. Each of these new techniques if rightly tested and enabled will offer much more significant savings. Future paper will focus on more of these techniques to save cost for organizations and to store lot more data as the need to ingest logs and metrics continues to rise.

9. CONCLUSION

Managing Elasticsearch clusters at scale can be a complex undertaking, but with the power of Kubernetes and ECK Operators, organizations can simplify the process significantly. This white paper explored the challenges faced in managing Elasticsearch clusters, introduced Kubernetes and ECK Operators as a solution, and provided a step-by-step guide to streamline the management process. By adopting this approach, organizations can achieve higher levels of automation, scalability, and resilience while reducing operational overhead and ensuring optimal Elasticsearch cluster performance.

Using the compression techniques can save upto 80% of raw file, this is significant as without any of these configurations, it will result is using a lot of hardware to store logs and can cost millions of dollars in hardware and licensing costs.

Users and administrators should always look at ways to optimize the logs created, ingested and stored in Elasticseach clusters. The information shared in this paper can help to reduce operational overhead and save organizations a lot of money and invest the savings in other critical areas.

REFERENCES

[1] Elasticsearch on Kubernetes: Step-by-step guides to run ELK on the most popular k8s platforms. Available: https://portworx.com/elasticsearch-kubernetes/

[2] Deepak Singh Dhami, When to use Kubernetes operators vs. Helm charts, Published: 01 Sep 2021. Available: https://www.techtarget.com/searchitoperations/tip/When-to-use-Kubernetes-operators-vs-Helm-charts?Offer=abt_pubpro_AI-Insider

[3] A simplified stack monitoring experience in Elastic Cloud on Kubernetes, 16 Sep 202. Available: https://www.elastic.co/blog/a-simplified-stack-monitoring-experience-in-elastic-cloud-on-kubernetes

[4] Migrate an ELK Stack to Elastic Cloud on AWS. Available: https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/migrate-an-elk-stack-to-elastic-cloud-on-aws.html

[5] Dirk Michel, On Amazon EKS and cluster add-ons, 5 Jul 2022. Available: https://medium.com/@micheldirk/on-running-amazon-eks-cluster-add-ons-29fae658e902

[6] Kevin Coleman, Apoorva Kulkarni, Mikhail Shapirov, and Vara Bonthu, Bootstrapping clusters with EKS Blueprints, 20 APR 2022. Available: https://aws.amazon.com/blogs/containers/bootstrapping-clusters-with-eks-blueprints/

[7] Adam Quan, Getting started with Elastic Cloud on Kubernetes: Deployment, 04 Sep 2019. Available: https://www.elastic.co/blog/getting-started-with-elastic-cloud-on-kubernetes-deployment

[8] Brandon Mensing, Peter Kim, Part 2.0: The true story behind Elasticsearch storage requirements, 15 Sep 2015. Available: https://www.elastic.co/blog/elasticsearch-storage-the-true-story-2.0

[9] Christian Dahlqvist, Filebeat modules, access logs and Elasticsearch storage requirements, 19 Jun 2019. Available: https://www.elastic.co/blog/filebeat-modiles-access-logs-and-elasticsearch-storage-requirements

Citation: Amreth Chandrasehar, Solving Complexity and Improving Storage Efficiency in Managing Elasticsearch Clusters Using Kubernetes and Eck Operators, International Journal of Information Technology (IJIT), 4(2), 2023, pp. 1-9

https://doi.org/10.17605/OSF.IO/F4UTS

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/IJIT/VOLUME_4_ISSUE_2/IJIT_4_02_002.pdf

editor@iaeme.com