# 

Graphical user interface, text, application

Description automatically generated

# Introduction

Elasticsearch is an open-source distributed analytics and search engine built on top of Apache Lucene. It provides a JSON-based REST API to access Lucene indexes and functions and adds a distributed system on top of Lucene to support cluster-based data storage and search functionality.

Elasticsearch supports numerous data aggregation and filtering tasks at search time in a single request, which makes it suitable for data analytics, monitoring, anomaly detection, data mining, text analysis, and even complex machine learning operations. Combining state-of-the-art data search with data analytics makes Elasticsearch great for the era of AI/ML and Big Data.

Elasticsearch is built for high performance with large pools of semi-structured and unstructured data. Among other things, this is achieved by efficient index design with dedicated, optimized data structures for each field type. For example, text fields use efficient inverted indexes, and numeric and geo fields are modeled as BKD trees.

Also, the platform provides such features as thread-pooling, queues, node/cluster monitoring API, data monitoring API, cluster management, etc

Elasticsearch data is stored in indexes—collections of documents similar to ‘databases’ in the relational database model. Each index can be partitioned into ‘shards,’ which may be thought of as index slices behaving like standalone indexes.

The motivation behind shards is to enable a distributed storage of data when an index is too big to run on a single node. With shards, one can split an index into multiple parts and distribute them evenly across the cluster.

Diagram

Description automatically generated

# Technical Design

## **Cluster Design Considerations**

### Avoid SPOF:

Designing for failure means running cluster nodes and be ready to lose any node in data center without service interruption.

Therefore, we recommend design it on Kubernetes, which makes it easier to deploy, scale, and manage Elasticsearch clusters at a large scale

### Kubernetes:

We can configure nodes for Autoscaling Horizontally or vertically by butting thresholds  
  
APP:  
We can configure pods for Autoscaling Horizontally or vertically by butting thresholds

Which will reflect while upgrade or updates and keep app up and running

Components:

(Eligible) master nodes control the cluster.

Http nodes: to run your queries to.

Data nodes: the place data is stored, obviously.

### Resilient Elasticsearch Cluster:

To maintain resilience of the cluster at least

- 3 Master-eligible nodes

-2 Data nodes

\*( At Least 2 copies of each shard ( one primary and replica )

### General Sizing

#### Sharding

It depends on how much data used

-1 primary shard by default Max shards per index (1024 )

\*Number of shards ( sourceData+Buffer) \* ( 1+10% index ) / Desired Shard size

Ex:

150 GB of Data

Number of shards = (150+0)\*1.10 /25 = 7

Number of shards ( with 100% grows in year ) = ( 150+150) )\*1.10 /25 =13 shards

#### Storage

Mini Storage required = ( source Data )\* (1+No of replicas ) \*1.45

#### Resources

Node resources = 2vCPU, 8GB Memory for every 100 GB of storage

### **Picnic Sizing**

Chart

Description automatically generated with medium confidence

Master and LB Resources:

CPU: 25m

Memory: 256Mi

Data node:

CPU: 25m

Memory: 2Gi

# **Technical Specifications and Implementation**

## OS considerations

Disable Swapping

Swapping involves transferring memory pages from the running memory (RAM) into swap files stored on the hard

drives. Swapping out running memory is appropriate for many applications, but is not a good default choice for

Elasticsearch. Being much slower than RAM, swap memory can compromise Elasticsearch cluster performance in

heavy I/O tasks and garbage collection. Therefore, it’s recommended to disable swap memory for Elasticsearch.

### File System Cache

A file system cache size is one of the main performance factors for Elasticsearch. File system cache enables faster

access to data for frequent reading operations and allows chaining a large amount of I/O operations. For expert

users, it’s possible to preload the content of “hot” index files into the file system cache upon starting Elasticsearch

thereby improving performance by a great margin. It’s important to make sure that at least half the memory of the

machine running Elasticsearch is dedicated to the file system cache.

### Other File and Memory Settings

Some other useful file and memory tweaks for Elasticsearch are related to file descriptors, Java heap size, and

number of threads.

File descriptors. It’s a good practice to increase the limit on the number of open file descriptors for the user

running Elasticsearch to 65,536 or higher.

Threads. Elasticsearch instances should be able to create at least 4096 threads for optimal performance on

heavy tasks.

Java heap size. As Elasticsearch runs in Java, the Java heap size allows controlling how much RAM Java Virtual

Machine (JVM) has access to. Although the larger heap size the better, usually it’s not recommended to set it

above 50% of the RAM because Elasticsearch may need memory for other tasks and a host’s OS can become

slow

## Kubernetes Cluster

Running on AWS EKS cluster which owned by run team

Text

Description automatically generated

## Kubernetes Packaging

Using Helm for installing Elasticsearch cluster and its components

#wget <https://get.helm.sh/helm-v3.4.1-linux-amd64.tar.gz>

## Add the Bitnami Repository and Deploy the Elasticsearch Chart

#helm repo add bitnami <https://charts.bitnami.com/bitnami>

## Install Charts

#helm install elasticsearch --set master.replicas=3,coordinating.service.type=LoadBalancer bitnami/elasticsearch

## Monitoring

#kubectl get pods until getting ready 1/1

## Test the Connection

#kubectl port-forward svc/elasticsearch-master 9200 #curl localhost:9200 ( from another terminal )

## Check pods info :

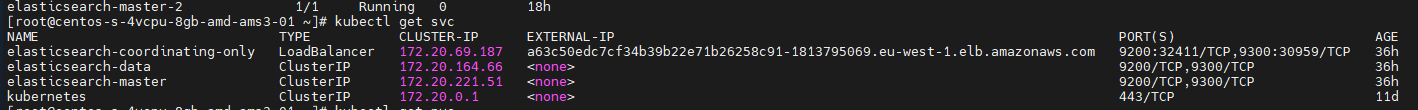
#kubectl get pods -o wide

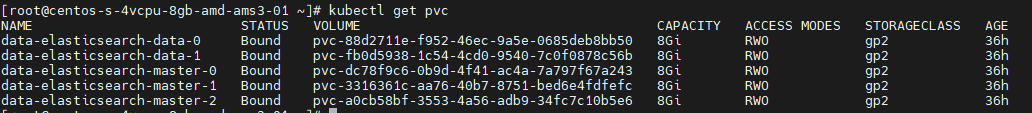
## Check cluster services

#kubectl get svc

Text

Description automatically generated





### Administration:

We can configure roles and cluster roles depends on teams and members