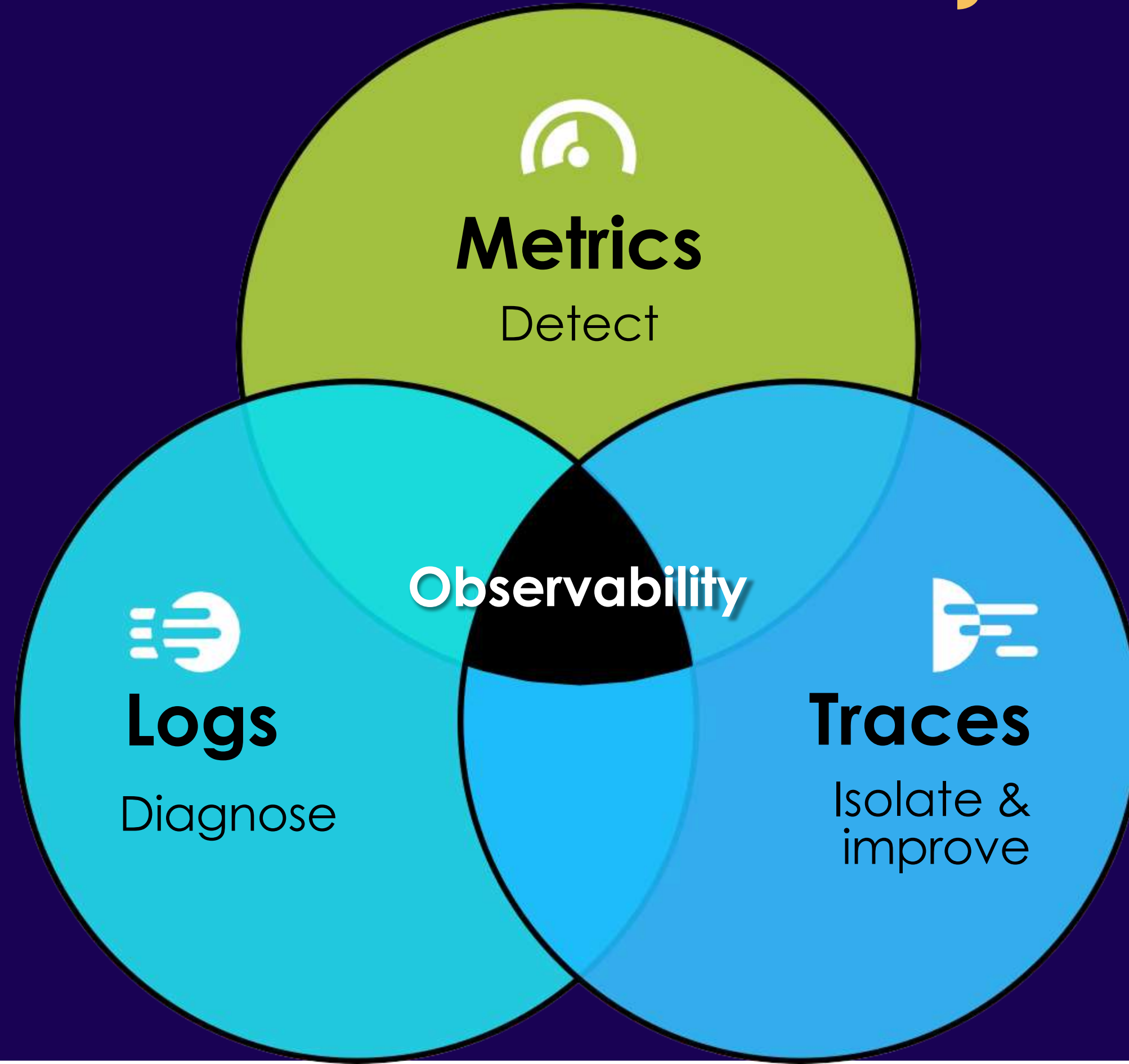


Getting started with ELK

Seshagiri Sriram - Aug 27/28 2025

The vision: unified observability



Observability

- Being able to measure “things” or witness state changes.
- Not useful if doing so alters behavior (significantly).
- **Measurement:** a single measurement of something
- **Metric:** something that you are measuring The version of deployed code
 - Total cost on Amazon services total bugs filed, bug backlog Total queries executed

Observability

- **Measurement Velocity** - The rate @ which measurement is taken
- **Perspective**
- **Visualization**
- **Trends**
- **Alerting**
- **MONITORING IS ALL OF THIS 😊**

A new Perspective

Monitoring		Observability
1	Says whether the System is Working or Not	Why its not working
2	Collects Metrics and Logs from a System	Actionable Insights gained from the Metrics
3	Failure Centric	Overall Behavior of the System
4	Is “the How” of something you do	Is “The Process” of something you have
5	I monitor you	You make yourself observable

Pillars of Observability

Logs/events



Immutable records of discrete events that happen over time

Metrics



Numbers describing a particular process or activity measured over intervals of time

Traces



Data that shows, for each invocation of each downstream service, which instance was called, which method within that instance was invoked, how the request performed, and what the results were

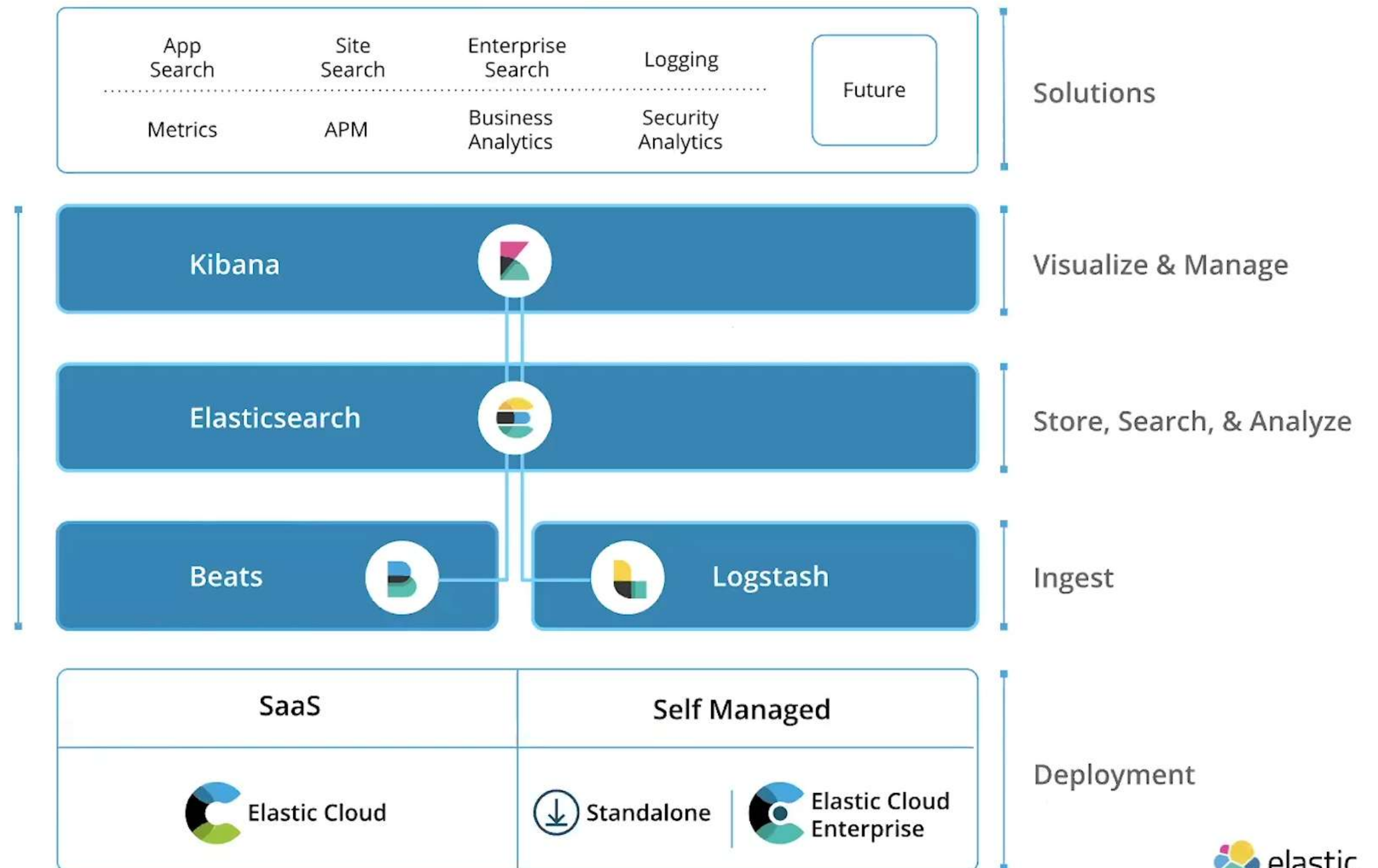
Logging with ELK

- ELK Stack is the leading open-source IT log management solution
- Centralized logging solution
- One of the most widely used stacks for processing log files and storing them as JSON documents.
- Extremely configurable, versatile, and scalable.
- Supports both simple and advanced operations.

What is the ELK Stack?

The ELK Stack is a collection of multiple open-source products, all developed, managed and maintained by [Elastic](https://www.elastic.co).

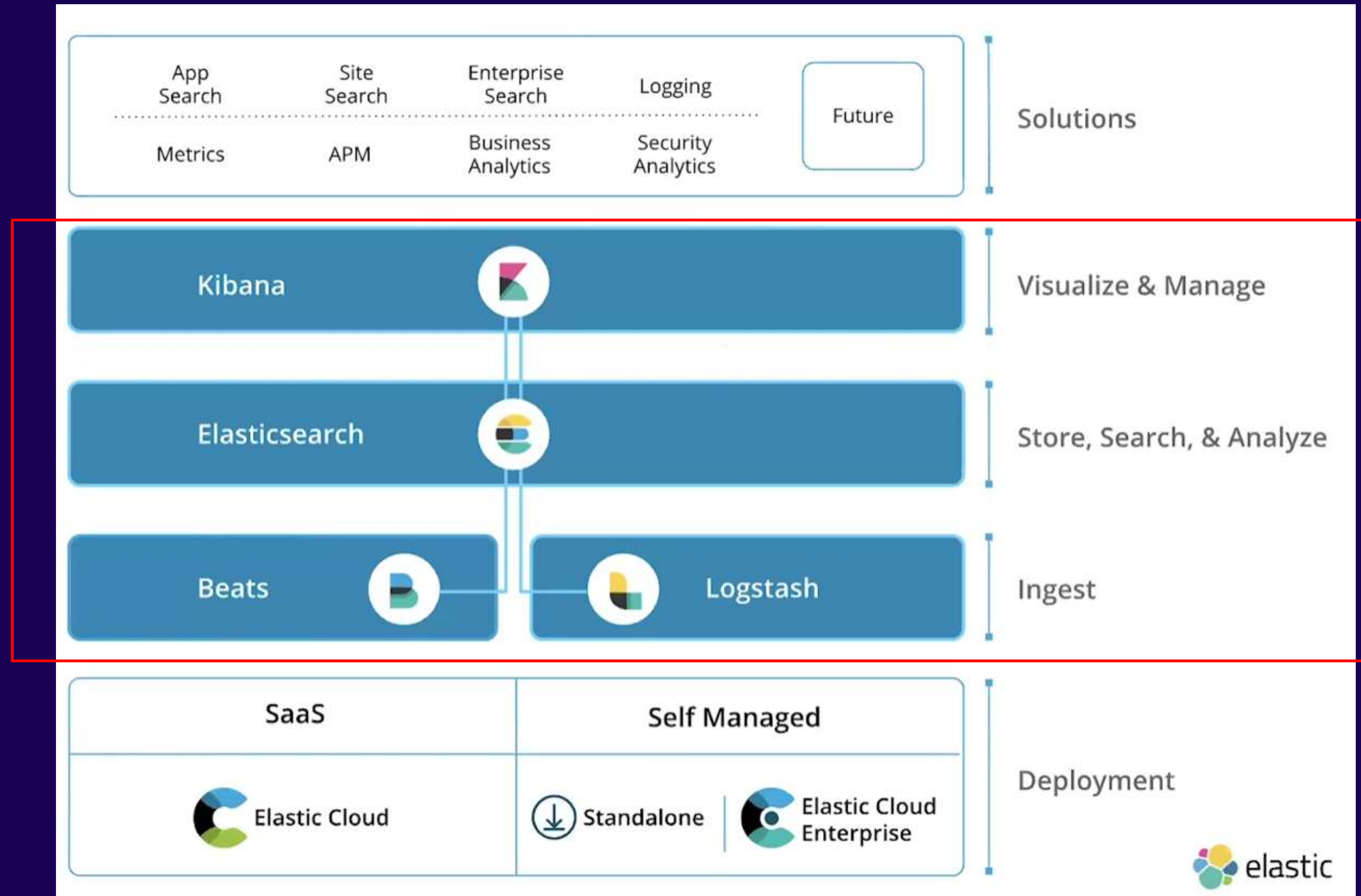

Elastic Stack



What is the ELK Stack?

However, we only need to use the following modules for this solution.

- **Beats**: agents to ship log data
- **Logstash**: processing log data
- **Elasticsearch**: storing log data
- **Kibana**: UI for visualizing



Elasticsearch Basic Concepts

- Index
 - An index contains one or multiple types
 - An Elasticsearch index is an independent chunk of document and stored on the disk in the same set of files
- Type
 - A type can be thought of as a table in a RDB
 - A type has one or more documents
- Document
 - A document is normally a JSON representation of your data

Terminology

Relation Databases

- Database
- Table
- Row
- Column
- Schema



Elasticsearch

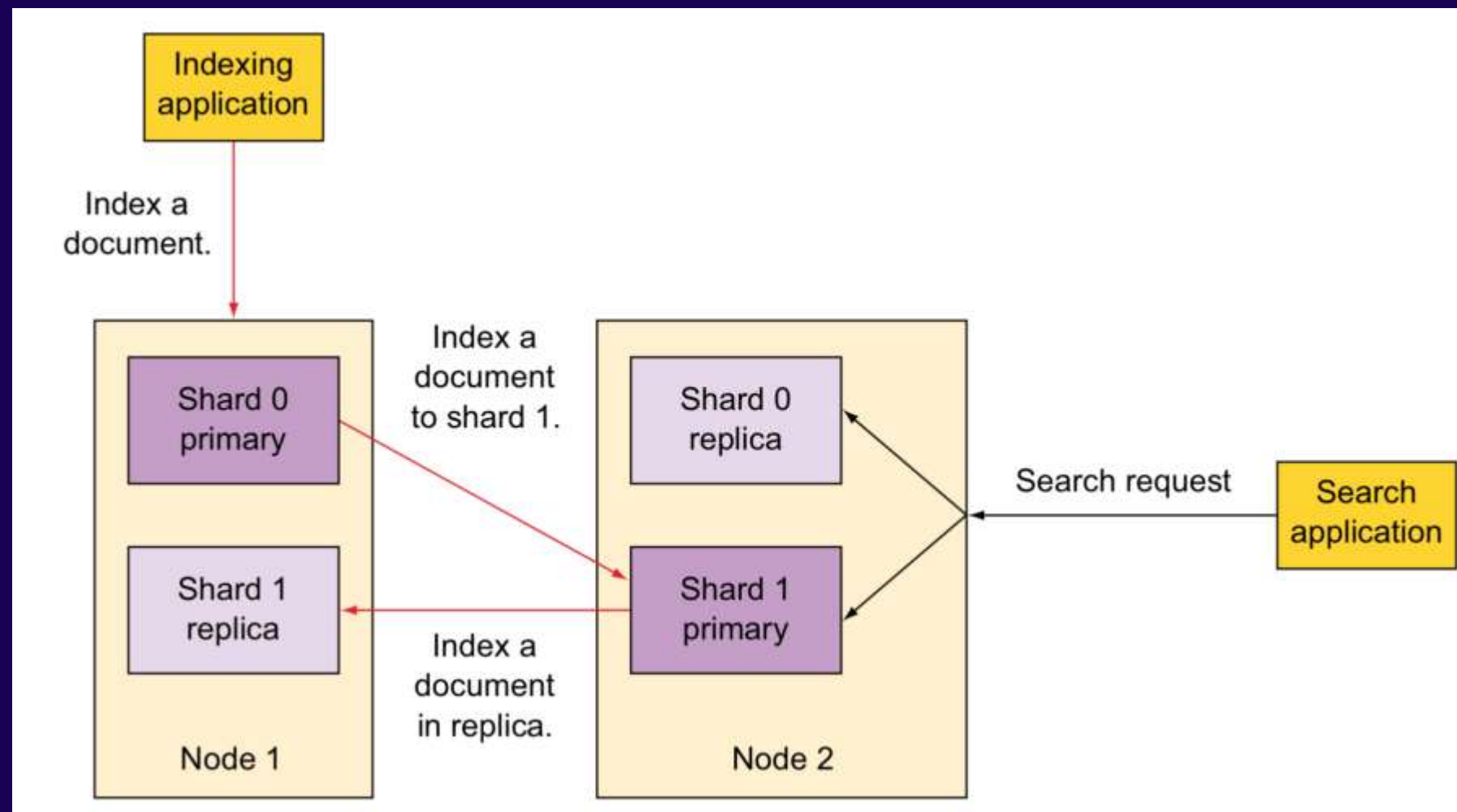
- Index
- Type
- Document
- Fields
- Mapping

Elasticsearch Basic Concepts

- Cluster
 - A cluster has one or more nodes. Clusters are identified by their names. (default cluster name= elasticsearch)
- Node
 - A node is an instance of Elasticsearch
 - Multiple nodes can join the same cluster. With a cluster of multiple nodes, the same data can be spread across multiple servers.
- Shard
 - An index can be divided into shards.
 - There are two types of shards
 - primary
 - replica

Elasticsearch Basic Concepts

- Node types
 - **Master node:** in charge of cluster management
 - **Data node:** Data nodes hold data and perform CRUD, search, and aggregations.
 - **Ingest node:** for pre-processing documents before indexing
 - **Machine learning node**

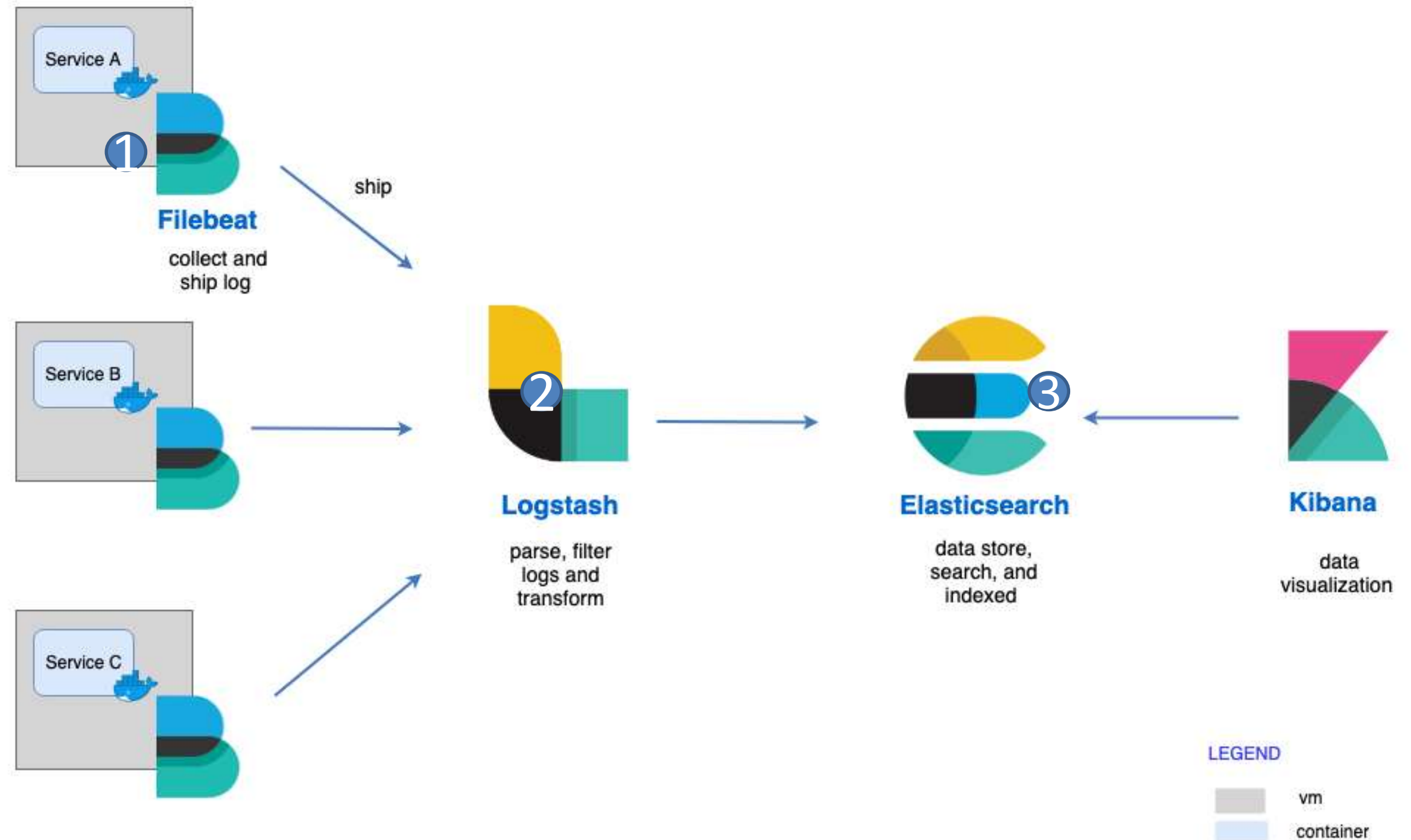


Each document, when indexed, is first added to the primary shard and then to one or more replica shards

Log Management Architecture Design

Data processing flow:

1. Filebeat will ship all the logs to Logstash
2. Logstash transform data format and output to Elasticsearch
3. view logs on Kibana and query by specific field



Logging with ELK

As data size increases and there is need for securing, we use other tools like

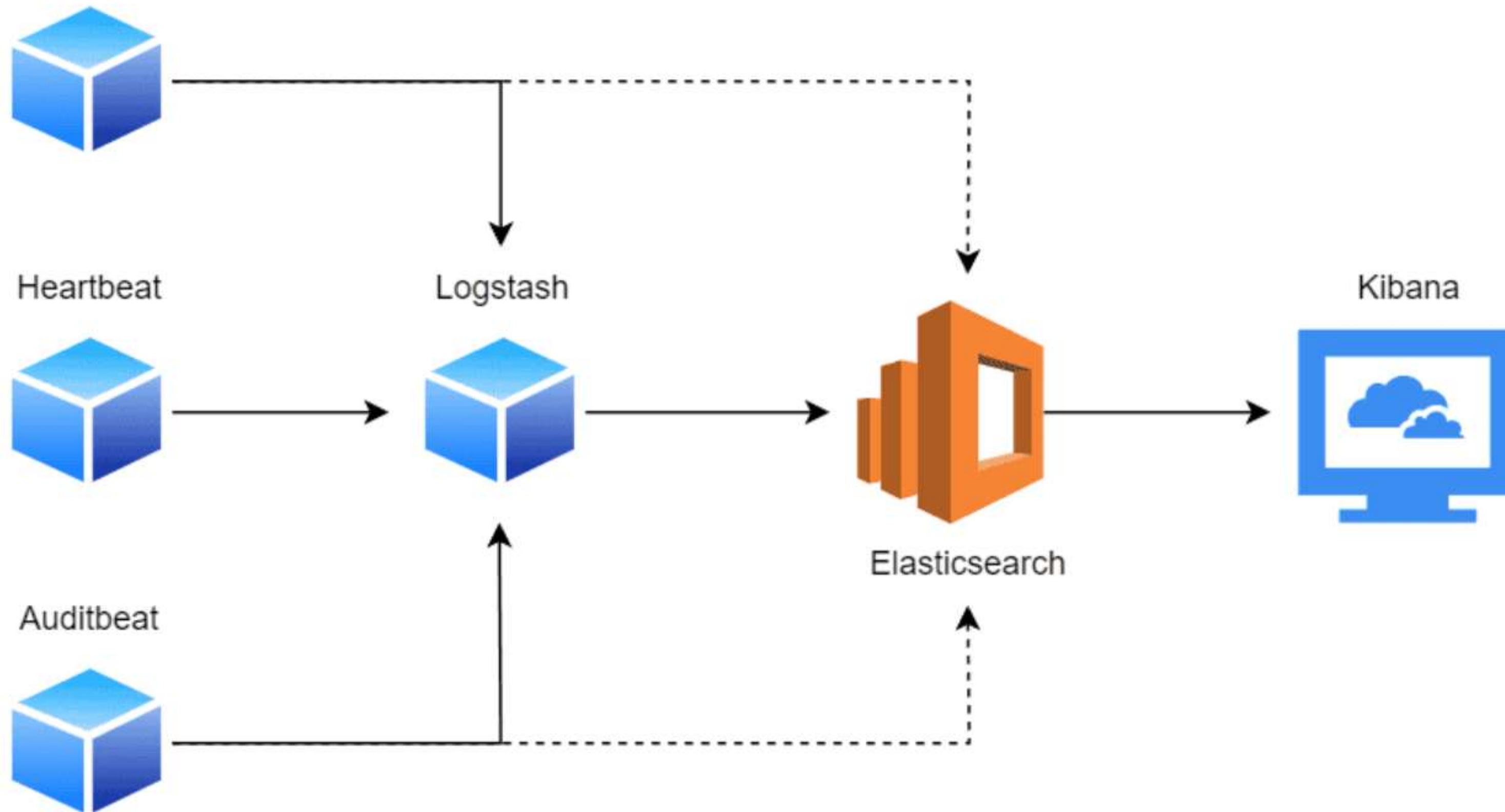
- Kafka

- Redis

- NGINX (for securing access)

Kafka, in particular, is used for its excellent streaming facilities

Logging with ELK - Summary



Implementation Details

Installing Elastic Stack using
Docker Compose

```
version: '2.2'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.0.0
    container_name: es01
    environment:
      - node.name=es01
      - cluster.initial_master_nodes=es01
      - cluster.name=docker-cluster
      - bootstrap.memory_lock=true
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - esdata01:/usr/share/elasticsearch/data
    ports:
      - 9200:9200
    networks:
      - esnet
  kibana:
    container_name: kibana
    image: docker.elastic.co/kibana/kibana:7.0.0
    environment:
      ELASTICSEARCH_HOSTS: http://es01:9200
    ports:
      - 5601:5601
    networks:
      - esnet
  logstash:
    container_name: logstash
    image: docker.elastic.co/logstash/logstash:7.0.0
    volumes:
      - /tmp/pipeline:/usr/share/logstash/pipeline/
      - /tmp/input:/tmp/input
    networks:
      - esnet
```

Implementation Details

- Filebeat

- is a lightweight log data shipper for local files
- monitors all the logs in the log directory and forwards to Logstash
- configuration:

```
filebeat.inputs:  
- type: log  
  enabled: true  
  paths:  
    - /tmp/log/api.log  
  multiline:  
    pattern: '(^.+Exception: .+)|(^\\s+at .+)|(^\\s+... \\d+ more)  
    negate: false  
    match: after  
  fields:  
    index: api  
    doc: spring-boot  
output.logstash:  
  hosts: ["localhost:5044"]
```

Because Java stack traces consist of multiple lines, so we need consolidate these lines into a single event in Filebeat

Logstash output sends events directly to Logstash by TCP

Implementation Details

- Logstash
 - An open source, server-side data processing pipeline that ingests data from a multitude of sources, transforms it, and then sends it to your favorite “stash.”
 - configuration:

The Logstash event processing pipeline has three stages:

1. input

2. filters
(optional)

3. outputs

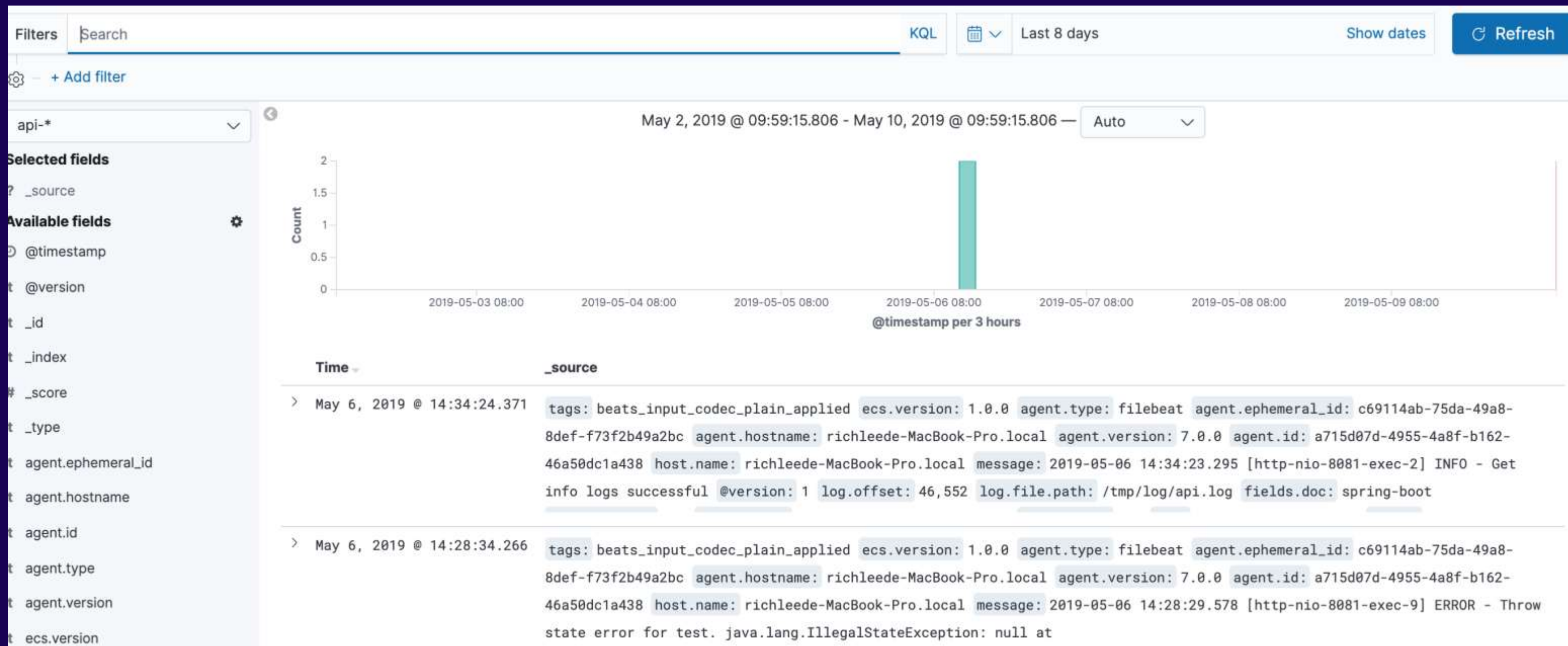
```
input {
  beats {
    port => "5044"
  }
}

filter {
  #If log line contains tab character followed by 'at' then we will tag that entry as stacktrace
  if [message] =~ "\tat" {
    grok {
      match => ["message", "^(\\tat)"]
      add_tag => ["stacktrace"]
    }
  }
}

output {
  stdout { codec => rubydebug }
  elasticsearch {
    hosts => [ "localhost:9200" ]
    index => "%{[fields][index]}-%{+YYYY.MM.dd}"
  }
}
```

Implementation Details

- Kibana
 - The Discover page allows you to perform different types of searches on the log data.



Advanced Design for Large Amounts of Logs

- Context:
 - When the service increases, the amount of data in the log will become larger
 - Following a production incident, logs can suddenly surge
- Problem:
 - The receiving end (Logstash or Elasticsearch) will be the main bottleneck.
 - Suddenly surge logs will overwhelm your logging infrastructure.
- Solution:
 - To protect Logstash and Elasticsearch against such data bursts, users deploy buffering mechanisms to act as message brokers.

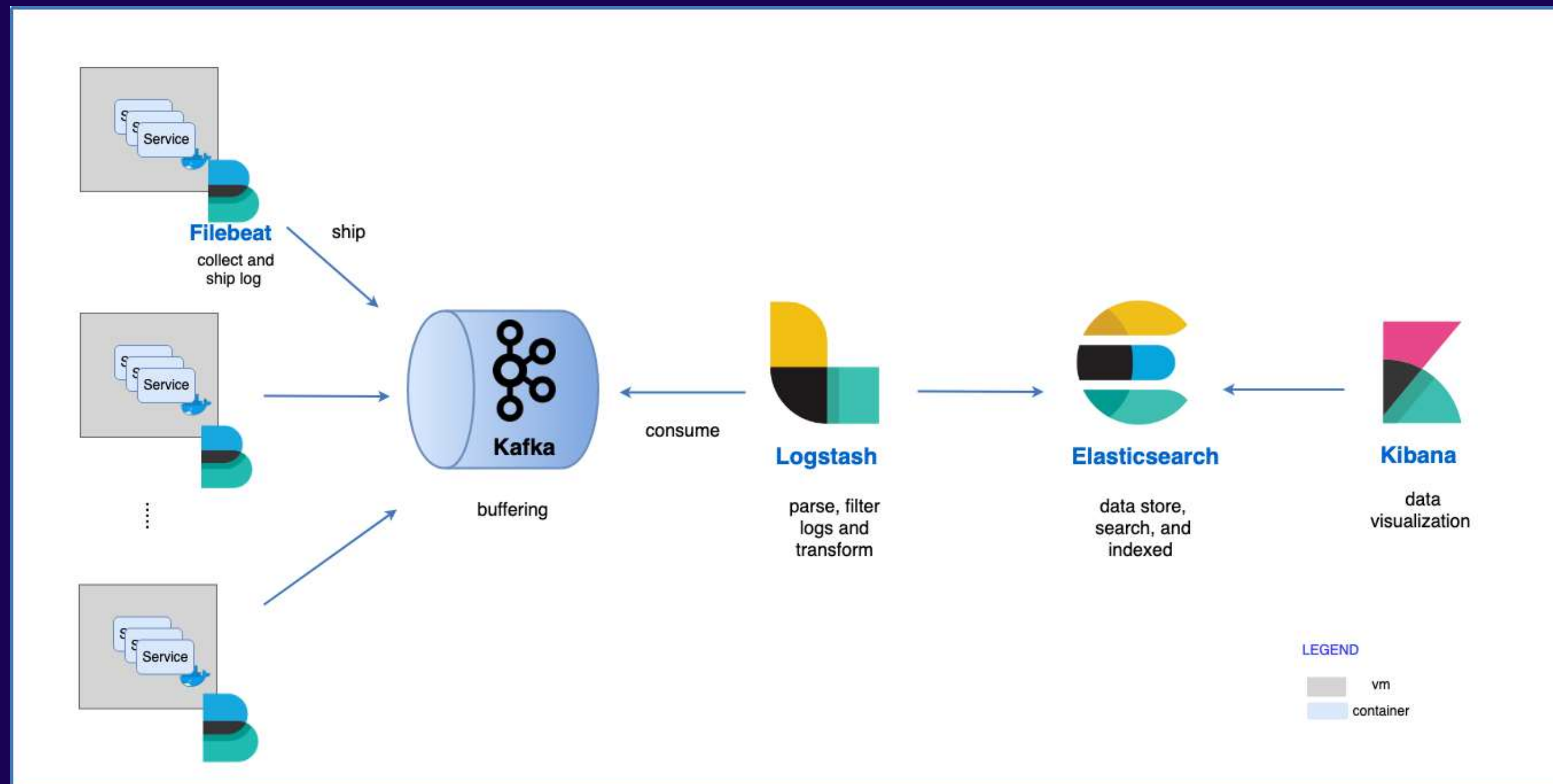
Advanced Design for Large Amounts of Logs

- Apache Kafka is a Publish-Subscribe messaging system, originated at LinkedIn in 2009, open sourced in 2011. Apache Kafka has some characteristics such as:
 - Kafka was designed to be distributed inherently. This makes Kafka be very easy to scale out.
 - High throughput, high performance
 - Guarantee the fault-tolerant in term of machine failure.



Advanced Design for Large Amounts of Logs

- Apache Kafka is the most common broker solution deployed together the ELK Stack.
- Usually, Kafka is deployed between the shipper and the indexer, acting as an entrypoint for the data being collected



Advanced Design for Large Amounts of Logs

- Logstash config: aggregates the data from the Kafka topic, processes it and ships to Elasticsearch.

```
input {
  kafka {
    bootstrap_servers => "localhost:9092"
    topics => "apache"
  }
}

filter {
  grok {
    match => { "message" => "%{COMBINEDAPACHELOG}" }
  }
  date {
    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
  geoip {
    source => "clientip"
  }
}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
  }
}
```

Logstash Kafka input plugin to define the Kafka host and the topic we want Logstash to pull from.

Advanced Design for Large Amounts of Logs

- Filebeat config: collects logs and forwards them to a Kafka topic.

```
filebeat.inputs:  
- type: log  
  enabled: true  
  paths:  
    - /var/log/apache2/access.log  
output.kafka:  
  codec.format:  
    string: '%{[@timestamp]} %{[message]}'  
  hosts: ["localhost:9092"]  
  topic: apache  
  partition.round_robin:  
    reachable_only: false  
  required_acks: 1  
  compression: gzip  
  max_message_bytes: 1000000
```

forward the data to Kafka server
and the relevant topic

Querying Data

- Elasticsearch Query DSL

- Get mapping field of index

```
curl -XGET 'http://localhost:9200/api-2019.05.06?pretty'
```

- Get document by id

```
curl -XGET 'http://localhost:9200/api-2019.05.06/_doc/sxXZi2oB-PgPWI1Y9CI1?pretty=true'
```

- URI Search

```
curl -XGET 'http://localhost:9200/api-2019.05.06 \
/_search?q=message:ERROR&from=0&size=10&sort=@timestamp:desc&pretty'
```


Querying Data

- Elasticsearch Query DSL
 - Request Body Search

```
curl -H "Content-Type: application/json" \
-XGET 'http://localhost:9200/api-2019.05.06/_search?pretty' -d \
'{
  "query" : {
    "match" : { "message": "ERROR"}
  }
}'
```

- Count query

```
curl -H "Content-Type: application/json" \
-XGET 'http://localhost:9200/api-2019.05.06/_doc/_count?pretty' -d \
'{
  "query" : {
    "term" : {"message" : "ERROR"}
  }
}'
```


Querying Data

- Elasticsearch Query DSL
 - Validate DSL

```
curl -H "Content-Type: application/json" \
-XGET 'http://localhost:9200/api-2019.05.06/_doc/_validate/query?pretty' -d \
'{
  "query" : {
    "term" : {"message" : "ERROR"}
  }
}'
```

result:

```
{
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "failed" : 0
  },
  "valid" : true
}
```

Querying Data

- Kibana
 - is the front end for Elasticsearch and provides visualizations for data, that can be used to search, view, and analyze data.
- Kibana User Interface
 - Discover
 - Visualize
 - Dashboard
- Kibana Query Language (KQL)
 - Release from version 6.3
 - If a default field is not set these terms will be matched against all fields

Querying Data

- Kibana Query Language (KQL)

- Match query

- message:"info"

- Token analyzer query

- message:"info error"

- Multiple values query

- message:(info or error)

- Wildcard query

- container.name: *api

- Field exist querie

- fields.doc:*

Logging with ELK - Summary

Generally, a single Elasticsearch cluster aggregates data from several Logstash instances. For example, multiple Logstash instances deployed on different nodes will collect data and feed it to the central Elasticsearch cluster for storage. Usually, these nodes are runtime environments like Kubernetes nodes, VMs, IoT devices, servers, and appliances.

However, another approach to data collection is through the use of Beats. Beats are lightweight single-purpose agents that ship data to a central Logstash instance. Nonetheless, they can also send the data directly to an Elasticsearch cluster, but that's not recommended.

Afterward, the aggregated data can be accessed through a Kibana instance, which provides a real-time interface for monitoring and data visualizations.

Installation on Kubernetes

Steps

- Install Elastic Search and Service
- Configure Logstash to point to Elastic Search
- Configure Logstash to get from a Beat Application
- Install Logstash
- Install Kibana and Service

Document and Scripts Available on GitHub

³¹

<https://github.com/SeshagiriSriram/Devops.git>

Management Tools

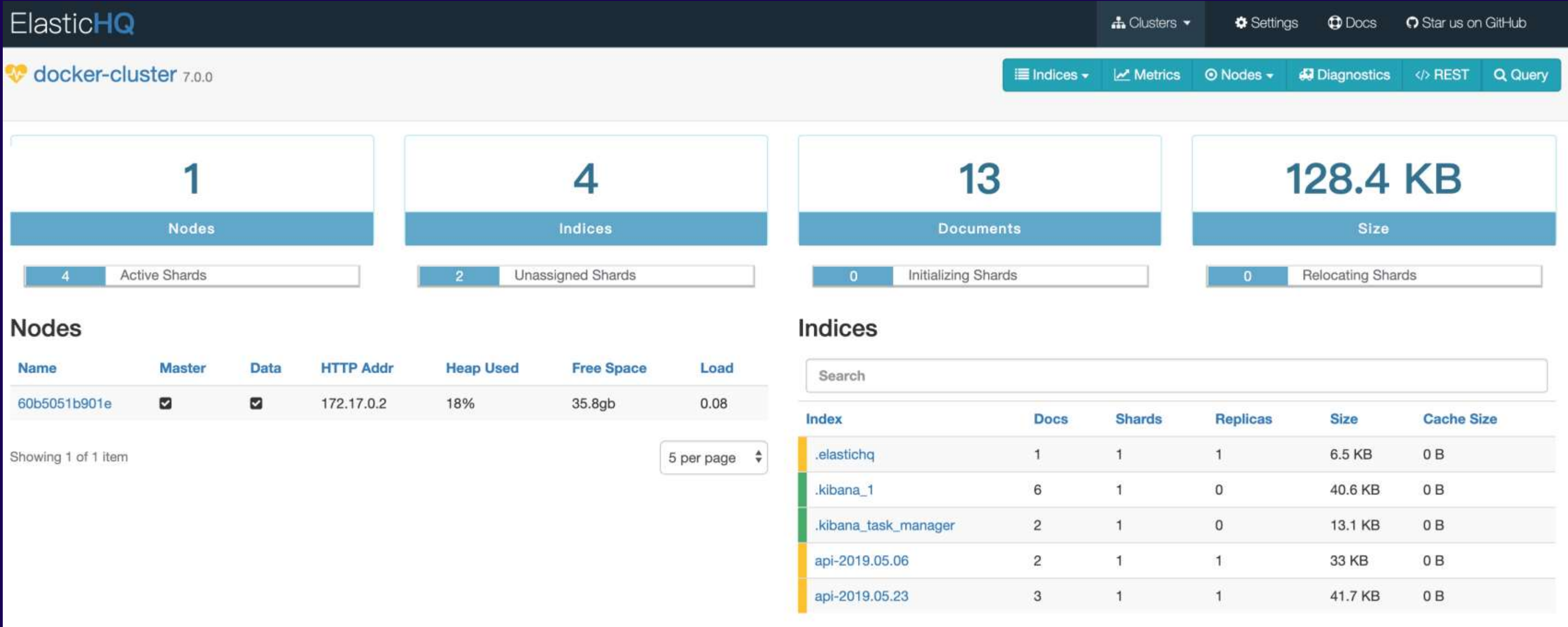
Elasticsearch Admin GUI Tools:

ElasticHQ : as a monitoring and management platform for Elasticsearch clusters.

Cerebro: is a elasticsearch web admin tool built using Scala, Play Framework, AngularJS and Bootstrap.


Management Tools

ElasticHQ



Management Tools

Cerebro



overviewnodesrestmore

15sechttp://10.137.98.249:9200 [yellow]

docker-cluster

1 nodes

5 indices

8 shards

14 docs

131.81KB






filter indices by name or aliases

closed (0)

.special (3)

filter nodes by name

1-2 of 2

<div><div></div></div>	api-2019.05.06 shards: 1 * 2 docs: 2 size: 32.19KB	api-2019.05.23 shards: 1 * 2 docs: 3 size: 40.69KB				
<div><div> 3 unassigned shards show only affected indices</div></div>	<div>0</div>	<div>0</div>				
<div><div><div>★ 60b5051b901e 📄 172.17.0.2 🔍</div><div>heapdiskcpuload</div></div></div>	<div>0</div>	<div>0</div>				



Thank You!