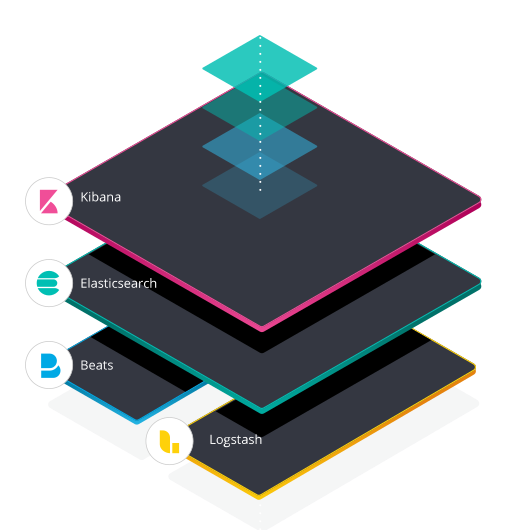
# ELK Stack

## What is the ELK Stack?

"ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana. Elasticsearch is a search and analytics engine. Logstash is a server‑side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch. Kibana lets users visualize data with charts and graphs in Elasticsearch.

The Elastic Stack is the next evolution of the ELK Stack.



## Companies/ Customers using ELK Stack

Netflix

Stack Overflow

LinkedIn

Fujitsu (OpenStack Cloud)

Accenture

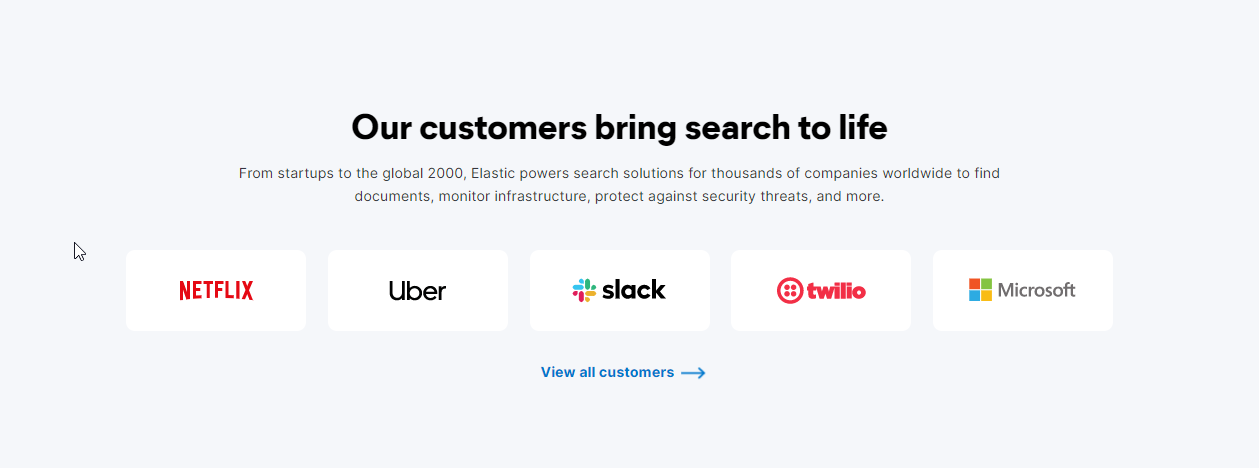
Tripwire

Medium

Swat.io

IFTTT

HipChat



## Define

### Elasticsearch

Elasticsearch is an open source, full-text search and analysis engine, based on the Apache Lucene search engine.

### Logstash

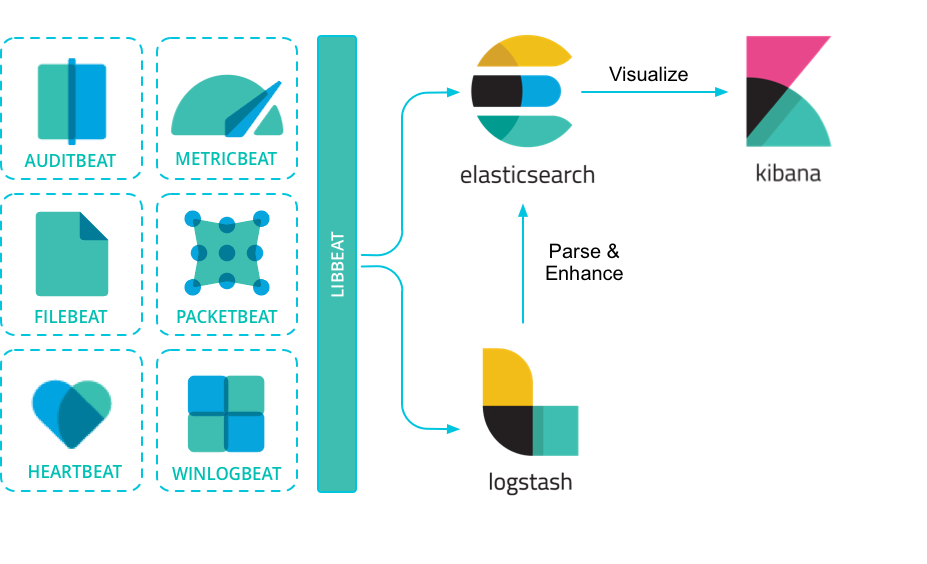
Logstash is a log aggregator that collects data from various input sources, executes different transformations and enhancements and then ships the data to various supported output destinations.

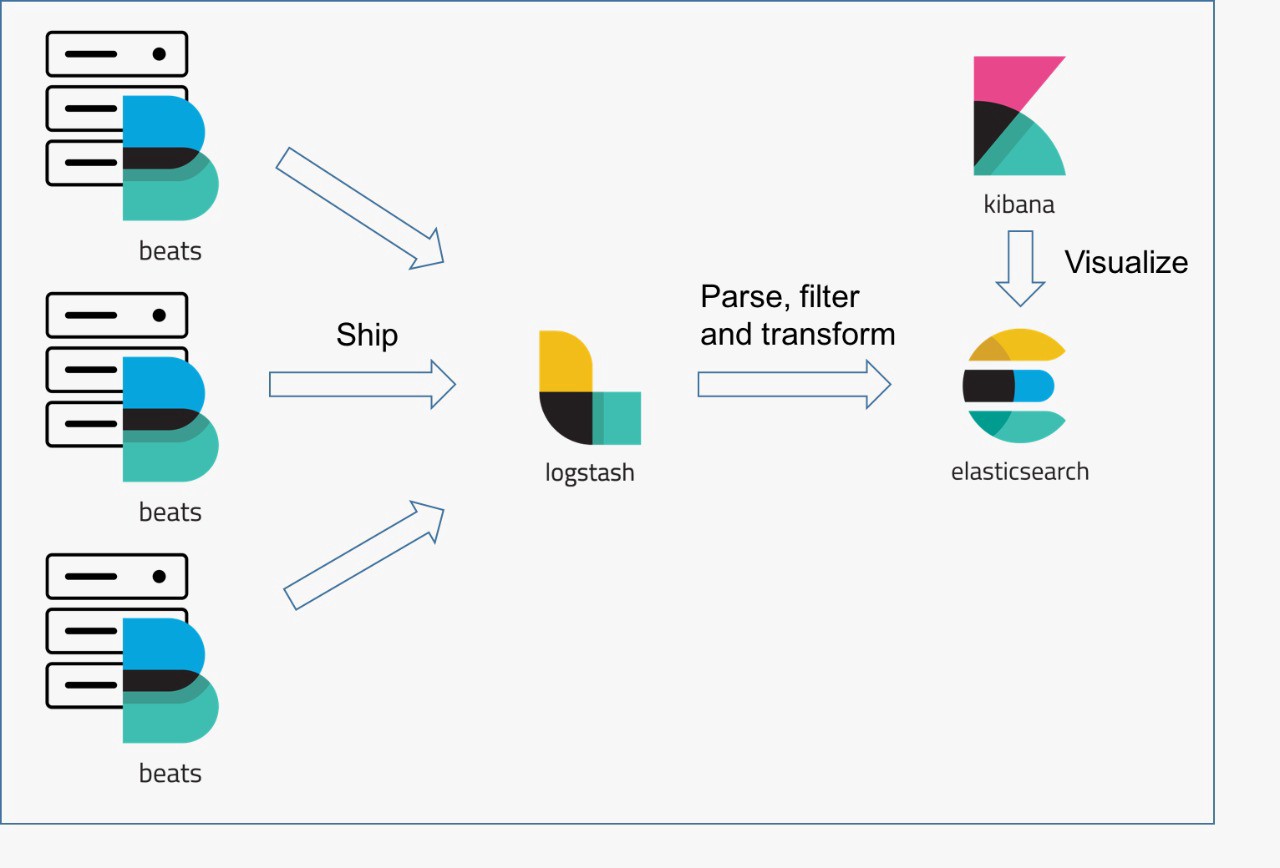
### Kibana

Kibana is a visualization layer that works on top of Elasticsearch, providing users with the ability to analyze and visualize the data.

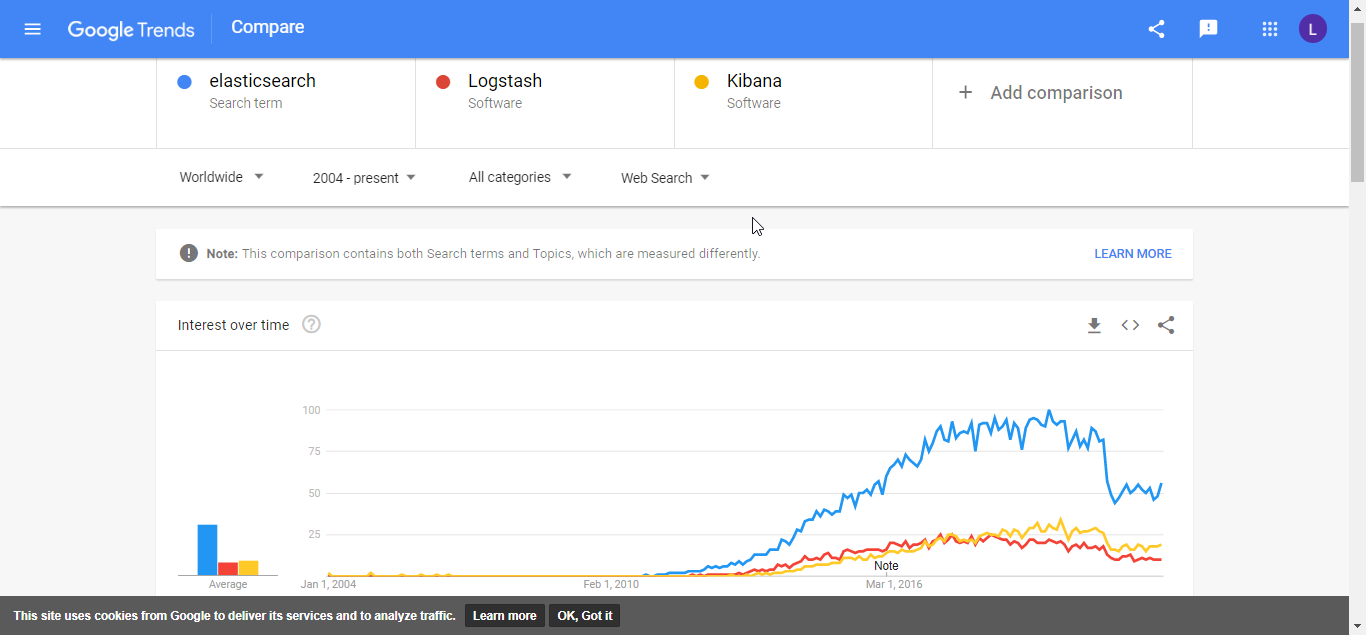
### Beats

Beats are lightweight agents that are installed on edge hosts to collect different types of data for forwarding into the stack.





## Google Trends of ELK Stack



## Modern log management and analysis

Modern log management and analysis solutions include the following key capabilities:

**Aggregation** – the ability to collect and ship logs from multiple data sources.

**Processing** – the ability to transform log messages into meaningful data for easier analysis.

**Storage** – the ability to store data for extended time periods to allow for monitoring, trend analysis, and security use cases.

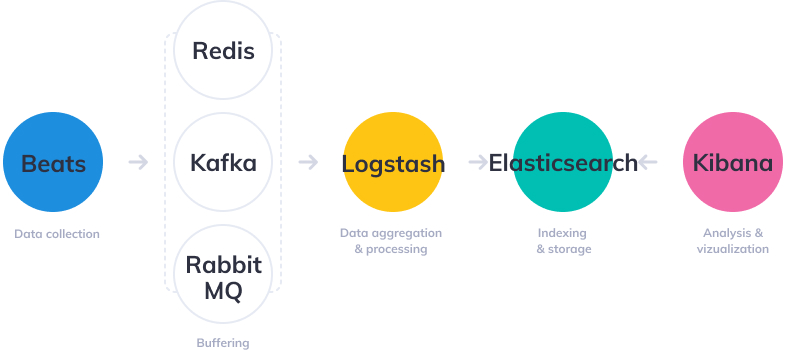
**Analysis** – the ability to dissect the data by querying it and creating visualizations and dashboards on top of it.

## Designing Stack for Your environment

For a small-sized development environment, the classic architecture will look as follows:



However, for handling more complex pipelines built for handling large amounts of data in production, additional components are likely to be added into your logging architecture, for resiliency (Kafka, RabbitMQ, Redis) and security (nginx):



## Elasticsearch

GET /

GET /\_all

## Logstash

Filter plugin

[kv](https://www.elastic.co/guide/en/logstash/current/plugins-filters-kv.html) - Parses key-value pairs

[grok](https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html) - Parses unstructured event data into fields

JSON - Parses JSON events

### Multiple Configuration files in Single

#### Method 1

input {

   file {

      path => "BatchData\Batch\_Raw\_Data.csv"

      tags => [ "batchdata" ]

      start\_position => "beginning"

   }

}

output {

   if "batchdata" in [tags] {

      elasticsearch {

         action => "index"

         index => "IndexName"

      }

   }

}

#### Method 2

<https://www.elastic.co/guide/en/logstash/current/multiple-pipelines.html>

<https://www.elastic.co/guide/en/logstash/current/pipeline-to-pipeline.html>

- pipeline.id: my-pipeline\_1

path.config: "/etc/path/to/p1.config"

pipeline.workers: 3

- pipeline.id: my-other-pipeline

path.config: "/etc/different/path/p2.cfg"

queue.type: persisted

Lo

### Logstash kubernetes Configuration

#### Json

filter {

      json {

          source => "message"

      }

    }

#### KeyValue

    filter {

      kv {

      }

    }

#### If condition for Array

    filter {

        if "json" in [tags] {

          json {

              source => "message"

          }

        }

        if "keyvalue" in [tags] {

          kv {

          }

        }

    }

#### App Key Value

if "json" in [tags] {

          json {

              source => "message"

          }

        }

        if "keyvalue" in [tags] {

          kv {

          }

        }

        if "appkeyvalue" in [tags] {

            grok {

                pattern\_definitions => {

                  "CUS\_PATH" => "\b\w+(/[[[:alnum:]]\_%!$@:.,+~-]\*)+"

                }

                match => [

                "message", "%{TIMESTAMP\_ISO8601:timeStamp}\|%{LOGLEVEL:logLevel}\|(?:%{WORD:appName})\|P%{NUMBER:processId}\|T%{NUMBER:thread\_id}\|%{CUS\_PATH:fileName}\|%{WORD:funcName}\|L%{NUMBER:lineNum}\|(?:%{WORD:correlationId})?\|%{GREEDYDATA:msg}"

                ]

            }

            if [msg] {

              kv{

                source => "msg"

              }

            }

        }

New Version 2

if "json" in [tags] {

          json {

              source => "message"

          }

        }

        if "keyvalue" in [tags] {

          kv {

          }

        }

        if "appkeyvalue" in [tags] {

            grok {

                pattern\_definitions => {

                  "DEFAULT\_PATH" => "(default\[#[0-9]+\]|%{WORD})"

                }

                match => [

                "message", "%{TIMESTAMP\_ISO8601:timeStamp}\|%{LOGLEVEL:logLevel}\|(?:%{WORD:appName})\|P%{NUMBER:processId}\|T%{NUMBER:thread\_id}\|%{PATH:fileName}\|%{WORD:funcName}\|L%{NUMBER:lineNum}\|%{DEFAULT\_PATH:correlationId}\|%{GREEDYDATA:msg}"

                ]

            }

            if [msg] {

              kv{

                source => "msg"

              }

            }

        }

## FileBeat

### Autodiscover

<https://logz.io/blog/what-is-autodiscover-filebeat/>

more

### Filebeat Modules with Docker & Kubernetes

<https://xeraa.net/blog/2020_filebeat-modules-with-docker-kubernetes/>

### Filebeat - Load Balance to Logstash



Filebeat kubernetes Configuration

<https://www.elastic.co/guide/en/beats/filebeat/6.8/defining-processors.html#conditions>

#### Input

##### All Container log in namespace

filebeat.inputs:

    - type: container

      paths:

        - /var/log/containers/\*.log

      processors:

        - add\_kubernetes\_metadata:

            host: ${NODE\_NAME}

            matchers:

            - logs\_path:

                logs\_path: "/var/log/containers/"

##### Image Name [One Container log]

filebeat.autodiscover:

      providers:

        - type: kubernetes

          templates:

            - condition:

                equals:

                  kubernetes.container.image: "ledison/applogger:json1.0"

              config:

                - type: container

                  paths:

                    - /var/log/containers/\*-${data.kubernetes.container.id}.log

##### Adding Tag using Filebeat [One Container log using kubernetes.container.image]

    filebeat.autodiscover:

      providers:

        - type: kubernetes

          templates:

            - condition:

                equals:

                  kubernetes.container.image: "ledison/applogger:json1.0"

              config:

                - type: container

                  paths:

                    - /var/log/containers/\*-${data.kubernetes.container.id}.log

                  tags: ["json"]

                  fields:

                    cus\_app\_id: "applogger\_filebeat"

                  fields\_under\_root: true

                  processors:

                  - add\_tags:

                      tags: "web"

                      target: "environment"

###### tags [adding value tags]

tags: ["json"]

"tags": [

"json",

"beats\_input\_codec\_plain\_applied"

],

###### Field [Adding tag]

                  fields:

                    cus\_app\_id: "applogger\_filebeat"

                  fields\_under\_root: true

"cus\_app\_id": "applogger\_filebeat",

###### Processors [Adding tag]

                  processors:

                  - add\_tags:

                      tags: "web"

                      target: "environment"

"environment": [

"web"

],

##### Deployment Name [One Container log]

filebeat.autodiscover:

      providers:

        - type: kubernetes

          templates:

            - condition:

                equals:

                  kubernetes.deployment.name: "applogger"

              config:

                - type: container

                  paths:

                    - /var/log/containers/\*-${data.kubernetes.container.id}.log

##### Three Type Application Format – Json, Keyvalue, AppKeyValue

- type: kubernetes

          templates:

            - condition:

                equals:

                  kubernetes.deployment.name: "apploggerjson"

              config:

                - type: container

                  paths:

                    - /var/log/containers/\*-${data.kubernetes.container.id}.log

                  tags: ["json"]

        - type: kubernetes

          templates:

            - condition:

                equals:

                  kubernetes.deployment.name: "apploggerkeyvalue"

              config:

                - type: container

                  paths:

                    - /var/log/containers/\*-${data.kubernetes.container.id}.log

                  tags: ["keyvalue"]

        - type: kubernetes

          templates:

            - condition:

                equals:

                  kubernetes.deployment.name: "apploggerappkeyvalue"

              config:

                - type: container

                  paths:

                    - /var/log/containers/\*-${data.kubernetes.container.id}.log

                  tags: ["appkeyvalue"]

##### Regular Expression

- type: kubernetes

          templates:

            - condition:

                regexp:

                  kubernetes.deployment.name: "cpaas.\*"

              config:

                - type: container

                  paths:

                    - /var/log/containers/\*-${data.kubernetes.container.id}.log

                  tags: ["json"]

All Container and One namespace adding tags

***hints.enabled: true***

filebeat.autodiscover:

      providers:

        - type: kubernetes

**hints.enabled: true**

          templates:

            - condition:

                equals:

                  kubernetes.namespace: "cpaas-demo-number-mgt"

              config:

                - type: container

                  paths:

                    - /var/log/containers/\*-${data.kubernetes.container.id}.log

                  tags: ["appkeyvalue"]

***kubernetes.namespace: "cpaas-demo-number-mgt"***

**Conditions**

<https://www.elastic.co/guide/en/beats/filebeat/current/defining-processors.html#conditions>

### CONFIGURATION-AUTODISCOVER-HINTS

important

<https://www.elastic.co/guide/en/beats/filebeat/7.15/configuration-autodiscover-hints.html>

#### Output

##### Output to Elasticsearch

output.elasticsearch:

      hosts: ['${ELASTICSEARCH\_HOST:elasticsearch}:${ELASTICSEARCH\_PORT:9200}']

##### Output to Logstash

output.logstash:

      hosts: ['logstash-service:5044']

## LOG Format

**Machine and human readability**

Formatting structures your logs. Structuring, in turn, helps both machines and humans read the data more efficiently.

<https://logz.io/blog/logging-best-practices/>

<https://kubernetes.io/blog/2020/09/04/kubernetes-1-19-introducing-structured-logs/>

### Json Format

### Key Value Format

2017-07025 17:02:12 level=error message="connection refused" service="listener" thread=125 customerid=776622 ip=34.124.233.12 queryid=45

### Timestamp

ISO8601 format

#### Example 1:

Consider this log message:

12-19-17 13:40:42:000 login failed.

Not very insightful, right? But how about:

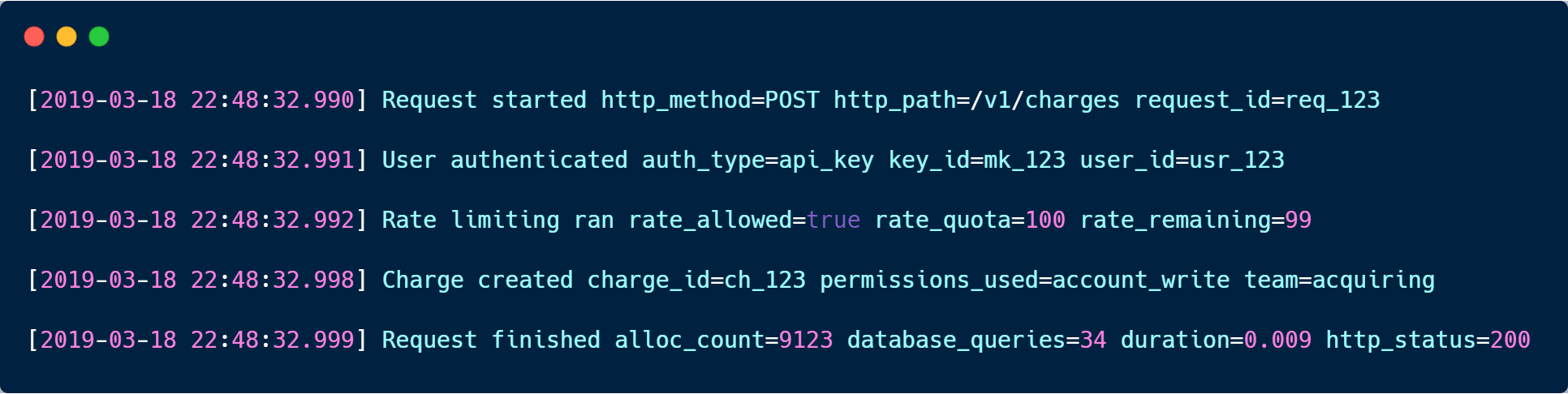
12-19-17 13:40:42:000 userId=23 action=login status=failure

Copy

In logging, context is all. Adding contextual information to your log messages creates a story and allows you, and any other party in your organization, to more easily understand and analyze the data.

#### Example 2:





### Kubernetes Structured Logs [KV or JSON ] [Go lang]

#### Using Structured Logs

We've added two new methods to the klog library: InfoS and ErrorS. For example, this invocation of InfoS:

klog.InfoS("Pod status updated", "pod", klog.KObj(pod), "status", status)

will result in this log:

I1025 00:15:15.525108 1 controller\_utils.go:116] "Pod status updated" pod="kube-system/kubedns" status="ready"

Or, if the **--logging-format=json** flag is set, it will result in this output:

{

**"ts"**: 1580306777.04728,

**"msg"**: "Pod status updated",

**"pod"**: {

**"name"**: "coredns",

**"namespace"**: "kube-system"

},

**"status"**: "ready"

}

### Logging format (filebeat module)

The logging format is generally the same for each logging output. The one exception is with the syslog output where the timestamp is not included in the message because syslog adds its own timestamp.

Each log message consists of the following parts:

* Timestamp in ISO8601 format
* Level
* Logger name contained in brackets (Optional)
* File name and line number of the caller
* Message
* Structured data encoded in JSON (Optional)

Below are some samples:

2017-12-17T18:54:16.241-0500 INFO logp/core\_test.go:13 unnamed global logger

2017-12-17T18:54:16.242-0500 INFO [example] logp/core\_test.go:16 some message

2017-12-17T18:54:16.242-0500 INFO [example] logp/core\_test.go:19 some message {"x": 1}

### Canonical Log Lines

A lightweight and stack agnostic operational technique for easy visibility into production systems.

### Json Format of ELK

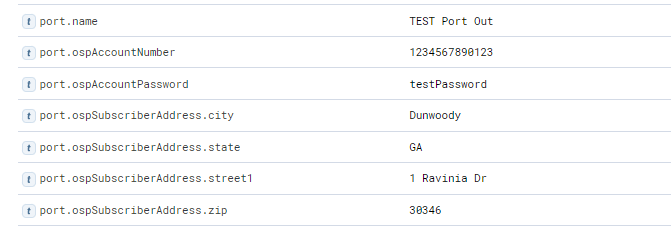
Write Log:

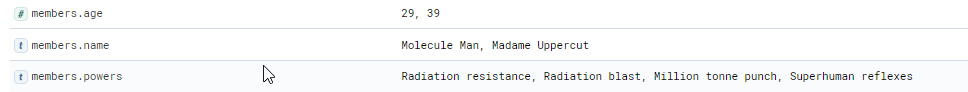






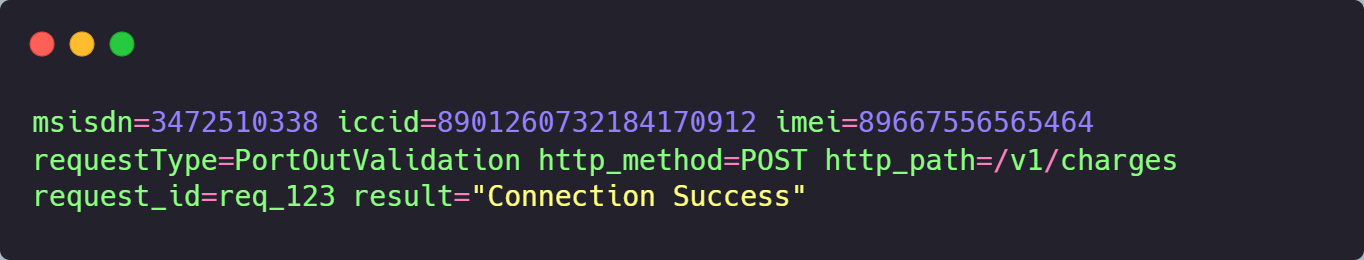


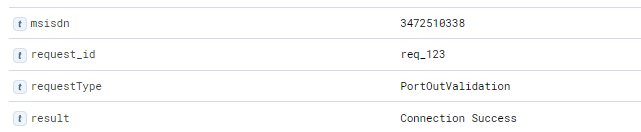




Store as Json Format

### KeyValue Format of ELK









SIEM [Security Information and Event Management]

https://www.elastic.co/siem/

## Kubernetes logs

### Logging Architecture

<https://kubernetes.io/docs/concepts/cluster-administration/logging/>

<https://codersociety.com/blog/articles/kubernetes-logging>

<https://sematext.com/guides/kubernetes-logging/>

#### Writing log

Likewise, container engines are designed to support logging. The easiest and most adopted logging method for containerized applications is **writing to standard output and standard error streams**.

Cluster-level logging architectures require a separate backend to store, analyze, and query logs**. Kubernetes does not provide a native storage solution for log data.** Instead, there are **many logging solutions** that integrate with Kubernetes. The following sections describe how to handle and store logs on nodes.

The Docker container engine redirects those two streams(stdout and stderr streams) to [a logging driver](https://docs.docker.com/engine/admin/logging/overview), which is configured in Kubernetes to write to a file in JSON format.

By default, if a container **restarts, the kubelet keeps one terminated container with its logs**. If a pod is evicted from the node, all corresponding containers are also evicted, along with their logs.

**Kubernetes is not responsible** for rotating logs, **but rather a deployment tool should set up** a solution to address that.

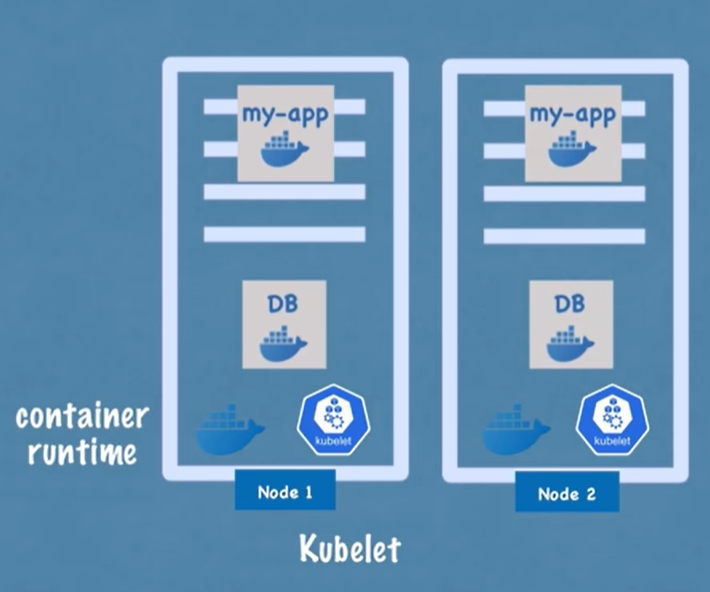
### Deploy Method for logrotate

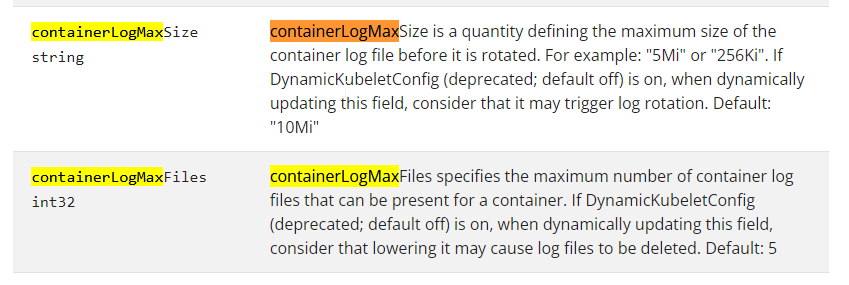
1. Kubelet Service
2. kube-up.sh

Cluster-level logging architectures

#### Kubelet Service

Two kubelet parameters containerLogMaxSize and containerLogMaxFiles in kubelet config file.





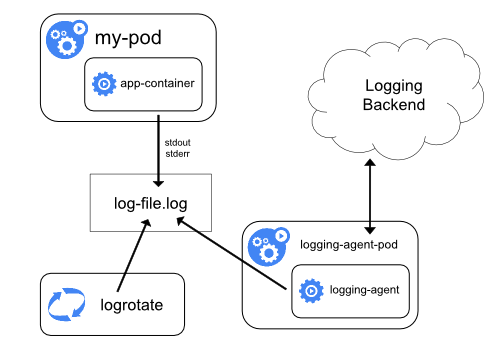
kube-up.sh

Can configure to be rotated by the logrotate tool daily or once the size exceeds 100MB.

### Cluster-level logging architectures

#### Type of logging model

* Using a node logging agent
* Using a sidecar container with the logging agent
* Sidecar container with a logging agent
* Exposing logs directly from the application
* Use a node-level logging agent that runs on every node.
* Include a dedicated sidecar container for logging in an application pod.
* Push logs directly to a backend from within an application



logging agent – Filebeat using DaemonSet.

### **sidecar container**

## Logger libraries

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Format** | | **Sink** | | **AsynLog** | **Performance** |  |
|  |  | **Key-Value** | **Json** | **Console** | **Socket** |  |  |  |
| Spdlog |  | NO | NO | YES | YES | YES |  |  |
| Log4cxx |  | NO | NO |  |  |  | Slow performance | |
| Loguru |  | NO | NO |  |  |  |  |  |
| Blackhole |  | YES | YES | YES | YES | YES |  | Develop by 3 ago, now no issue support Key value - all should be string |
| Structlog |  |  |  |  |  |  |  |  |
| Glog |  | NO | NO |  |  |  |  |  |

<https://github.com/3Hren/blackhole>

<https://github.com/takeshibaconsuzuki/structlog>

Level 1

grok {

                pattern\_definitions => {

                  "DEFAULT\_PATH" => "(default\[#[0-9]+\]|%{WORD})"

                }

                match => [

                "message", "%{TIMESTAMP\_ISO8601:timeStamp}\|%{LOGLEVEL:logLevel}\|(?:%{WORD:appName})\|P%{NUMBER:processId}\|T%{NUMBER:thread\_id}\|%{PATH:fileName}\|%{WORD:funcName}\|L%{NUMBER:lineNum}\|%{DEFAULT\_PATH:correlationId}\|%{GREEDYDATA:msg}"

                ]

            }

Problem

2023-03-31T10:53:01.722276+0000|INFO|NumberAnalysis|P6|T3038412869919268771|/number-analysis/src/Main.cpp|operator()|L50|default[#1]|Request Handler method="GET" uriPath="/numbers/active-numbers"

2023-03-01T14:23:11.607438+0000|INFO|SMS-STATUS-MANAGER|P7|T7563049970348772459|/Service/src/main.cpp|main|L29|MSM\_INTERNAL|Starting Service VERSION="0.0.0"

**Solution**

%{TIMESTAMP\_ISO8601:timeStamp}\|%{LOGLEVEL:logLevel}\|(?:%{DATA:appName})\|P%{NUMBER:processId}\|T%{NUMBER:thread\_id}\|%{PATH:fileName}\|%{DATA:funcName}\|L%{NUMBER:lineNum}\|%{DATA:correlationId}\|%{GREEDYDATA:msg}