Elastic Stack

also known as ELK

Overview and Usage

About me

- Saeed Rasooli
- github.com/ilius
- Go and Python. Mostly Back-end and desktop
- Open Source projects: StarCalendar, PyGlossary, AyanDict and several command line tools in Go
- saeed.gnu@gmail.com, saeedgnu@riseup.net

About me

- Currently work at ParsPooyesh
- Most of this slide was made at work, as a result of a R&D task.
- https://github.com/ParspooyeshFanavar

What is Elastic Stack?

- It’s a “Search Platform” according to its website:
  - https://www.elastic.co/elastic-stack/
- It’s comprised of
  - Elasticsearch: Distributed RESTful Search Engine
  - Logstash: transport and process your logs, events or other data
  - Kibana: Your window into the Elastic Stack
  - Beats: Lightweight shippers (in Go) for Elasticsearch & Logstash

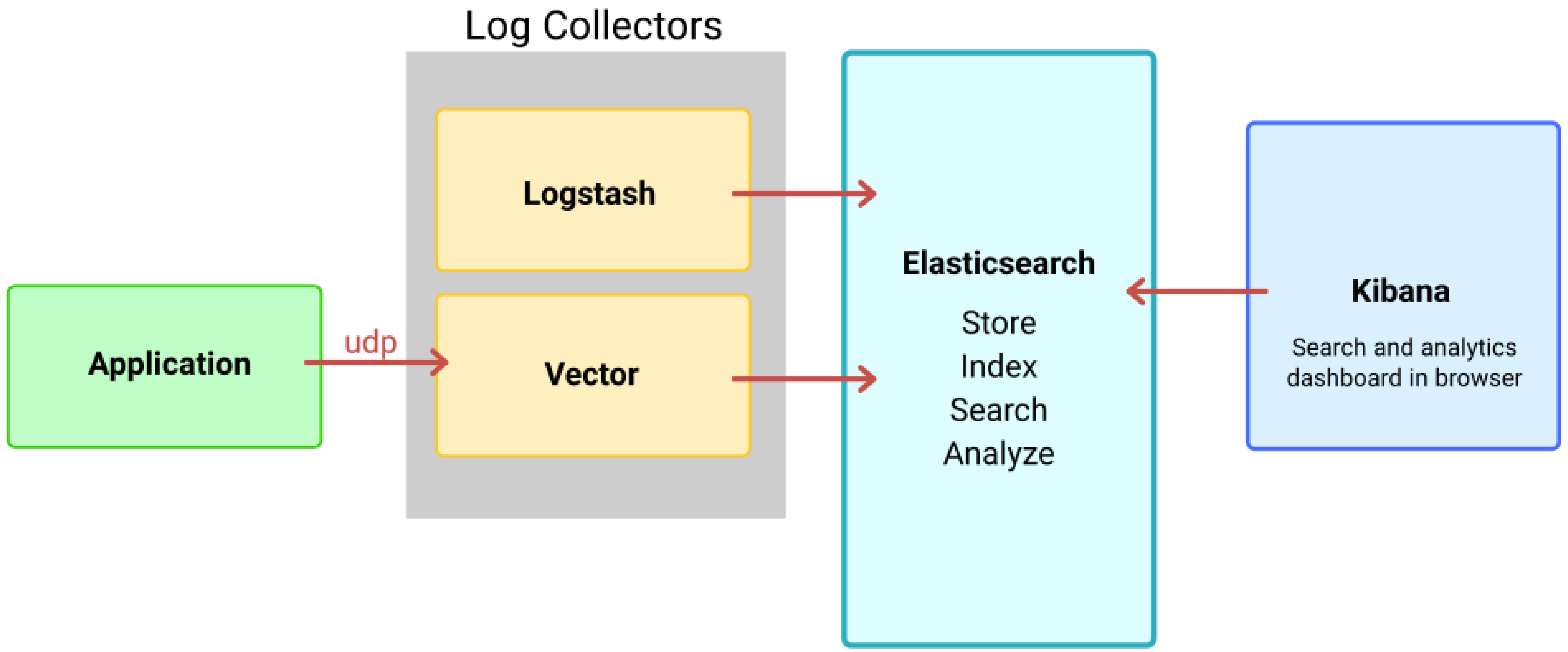
Why use Elastic Stack?

Because log files are:

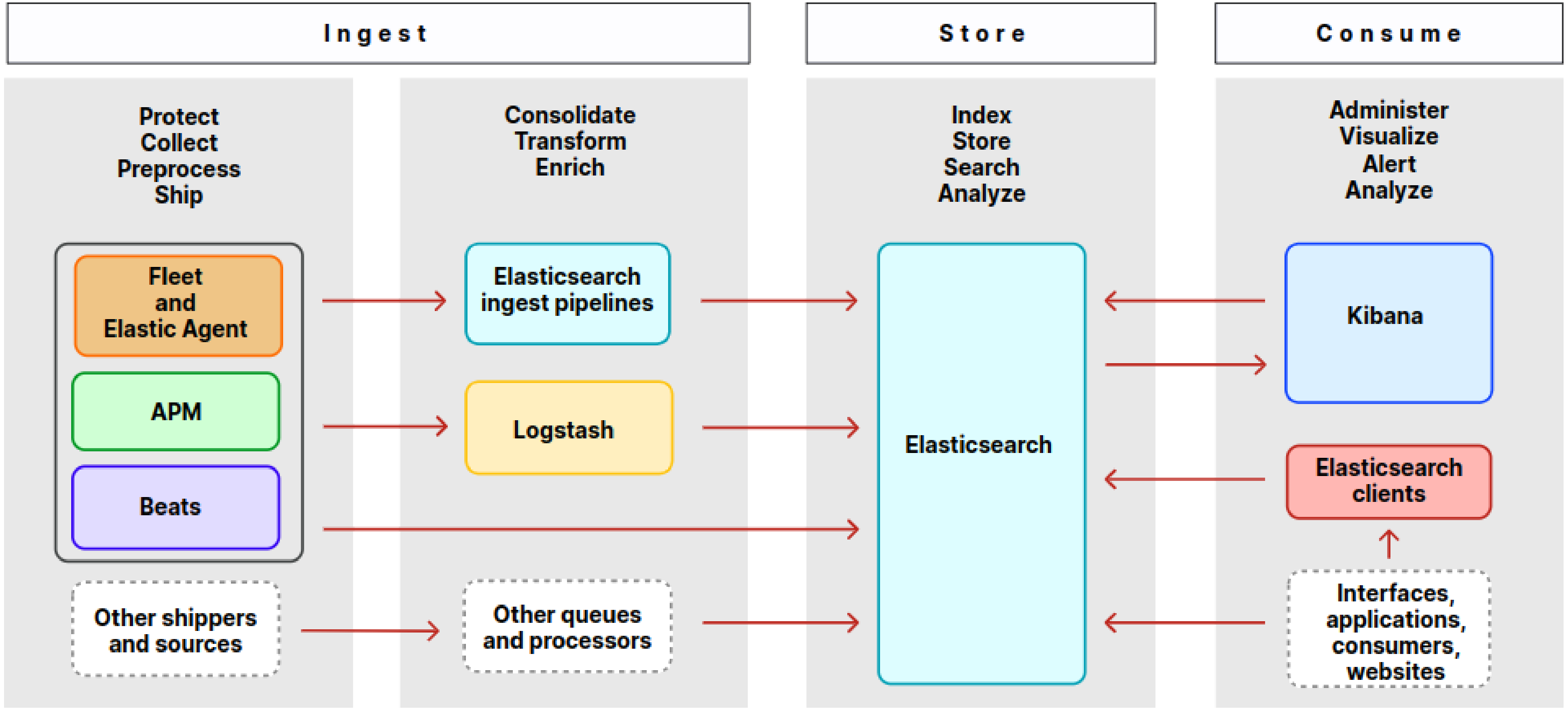
- Not centralized or container-friendly
  - Mount external storage for each instance of app.
  - Each instance writes it's own log files.
  - Must observe free space for each instance (or share it)
- Not easy to filter and/or aggregate by level / component.
- Not easy / fast to search (no index).
- Need command line access to view / monitor logs.

- **Centralized Logging**: The Elastic Stack provides a centralized repository for storing logs from various sources, making it easier to collect, manage, and analyze log data.
- **Real-time Analysis**: Logs are ingested and indexed in real time, allowing for immediate analysis and monitoring of system activity.
- **Advanced Search and Analytics**: Elasticsearch's powerful search capabilities enable users to quickly find specific log entries and perform complex data analysis.
- **Visualization and Dashboards**: Kibana provides a user-friendly interface for visualizing log data in dashboards, charts, and graphs, making it easier to identify trends and patterns.
- **Scale and Reliability**: The Elastic Stack is designed to handle large volumes of log data and can be scaled horizontally to meet growing demands.

How we use it


Log Collectors
Logstash
Vector
Application
udp
Elasticsearch
Store
Index
Search
Analyze
Kibana
Search and analytics dashboard in browser

Components of the Elastic Stack



APM: Application Performance Monitoring

Beats

A collection of lightweight shippers (data pipelines)

- Written in Go
- Licensed under the Apache License version 2.0
- **https://github.com/elastic/beats**

- **`libbeat`**: Go library containing the common packages for all the Beats. It is Apache licensed and actively maintained by the Elastic team.

Beat	Description
Audit beat	Collects your Linux audit framework data and monitors the integrity of your files
File beat	Tails and ships log files
Function beat	Reads and ships events from serverless infrastructure
Heart beat	Pings remote services for availability
Metric beat	Fetches sets of metrics from the operating system and services
Packet beat	Monitors the network and applications by sniffing packets
Winlog beat	Fetches and ships Windows Event logs
Osquery beat	Runs Osquery and manages interraction with it

What is Elasticsearch NOT?

- It is designed as a “Search Engine”, not a “Database”
- It’s not meant to be used as a primary store.
- Want to use it as a NoSQL Database? Read this post first:
  - https://www.elastic.co/blog/found-elasticsearch-as-nosql
  - “Elasticsearch is commonly used in addition to another database. A database system with stronger focus on constraints, correctness and robustness, and on being readily and transactionally updatable”

Log collectors

	Language	GH Stars	Last commit
`logstash`	Java, Ruby	13.6k	2023
`vector`	Rust	14.2k	2023
`fluentd`	Ruby	12.1k	2023
`fluent-bit`	C, C++	4.8k	2023
`gogstash`	Go	104	2023
`logcool`	Go	164	2017
`protologbeat`	Go	28	fork: 2022
`logpeck`	Go	23	2020

Log collectors: performace

Test	Vector	FluentBit	Logstash	FluentD	Filebeat	Splunk UF	Splunk HF
TCP to Blackhole	86.0	64.4	40.6	27.7	n/a	n/a	n/a
File to TCP	76.7	35.0	3.1	26.1	7.8	40.1	39.0
Regex Parsing	13.2	20.5	4.6	2.6	n/a	n/a	7.8
TCP to HTTP	26.7	19.6	2.7	1.0	n/a	n/a	n/a
TCP to TCP	69.9	67.1	10.0	3.9	5.0	70.4	7.6
Average	54.5	41.3	12.2	12.3			

Source: Vector’s Github: https://github.com/vectordotdev/vector/

Splunk: US company, server is closed-source: https://en.wikipedia.org/wiki/Splunk

Log collectors: my choice

I chose **Vector**. Why?

- Startup time: much lower than Logstash
- Better performance (better than/comparable with Fluent Bit: C++)
- Documentation: very clean, organized, complete
- Errors on mis-config: nicer and more readable
- Runtime-safety of Rust (even better than Go or Java)
- Easier to build/compile than C/C++
- So many stars on Github (even more than Logstash)

Data streams

A data stream lets you store append-only time series data across multiple indices while giving you a single named resource for requests.

Data streams are well-suited for logs, events, metrics, and other continuously generated data.

(from elastic.co)

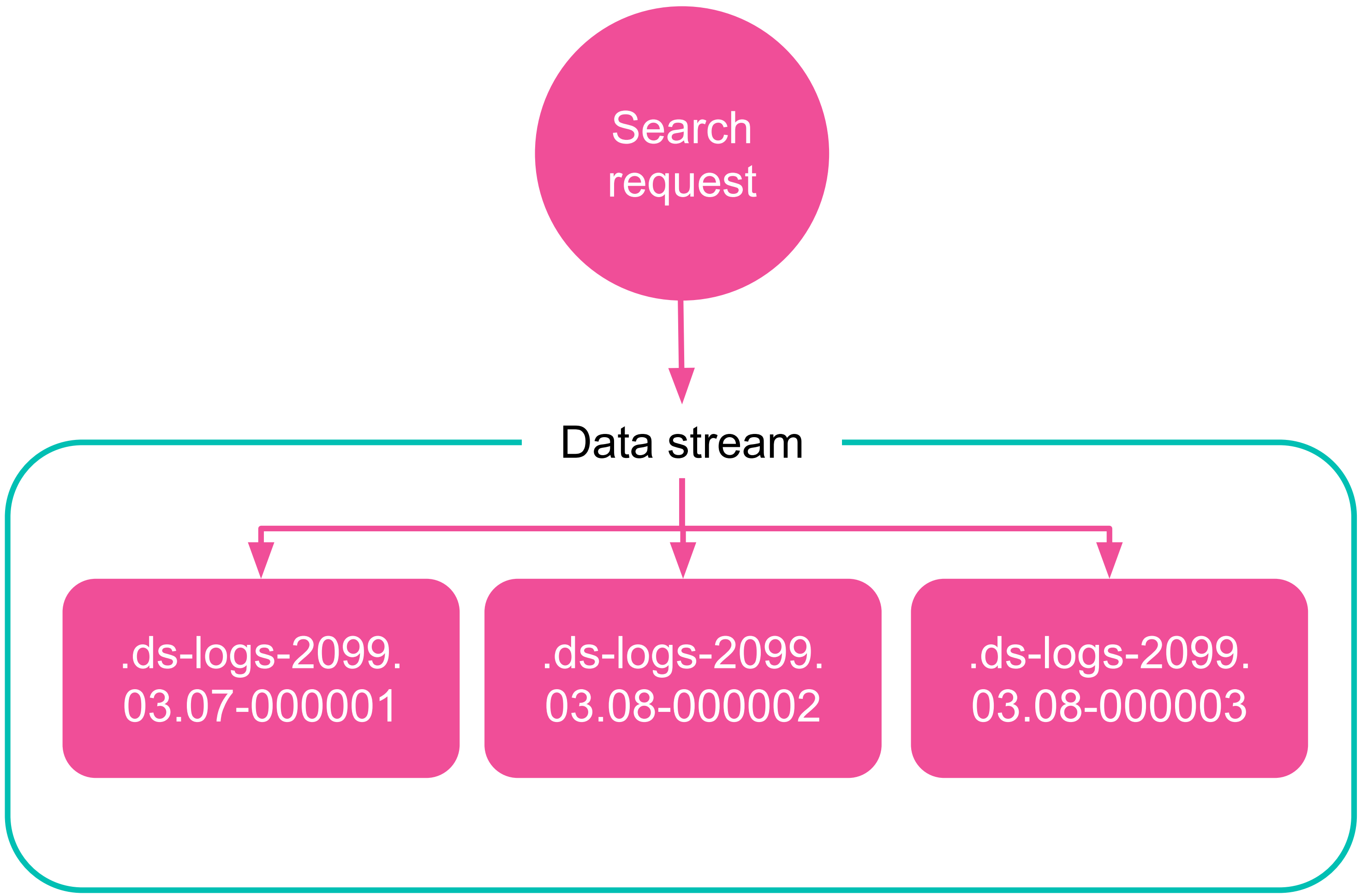
Data streams: indices

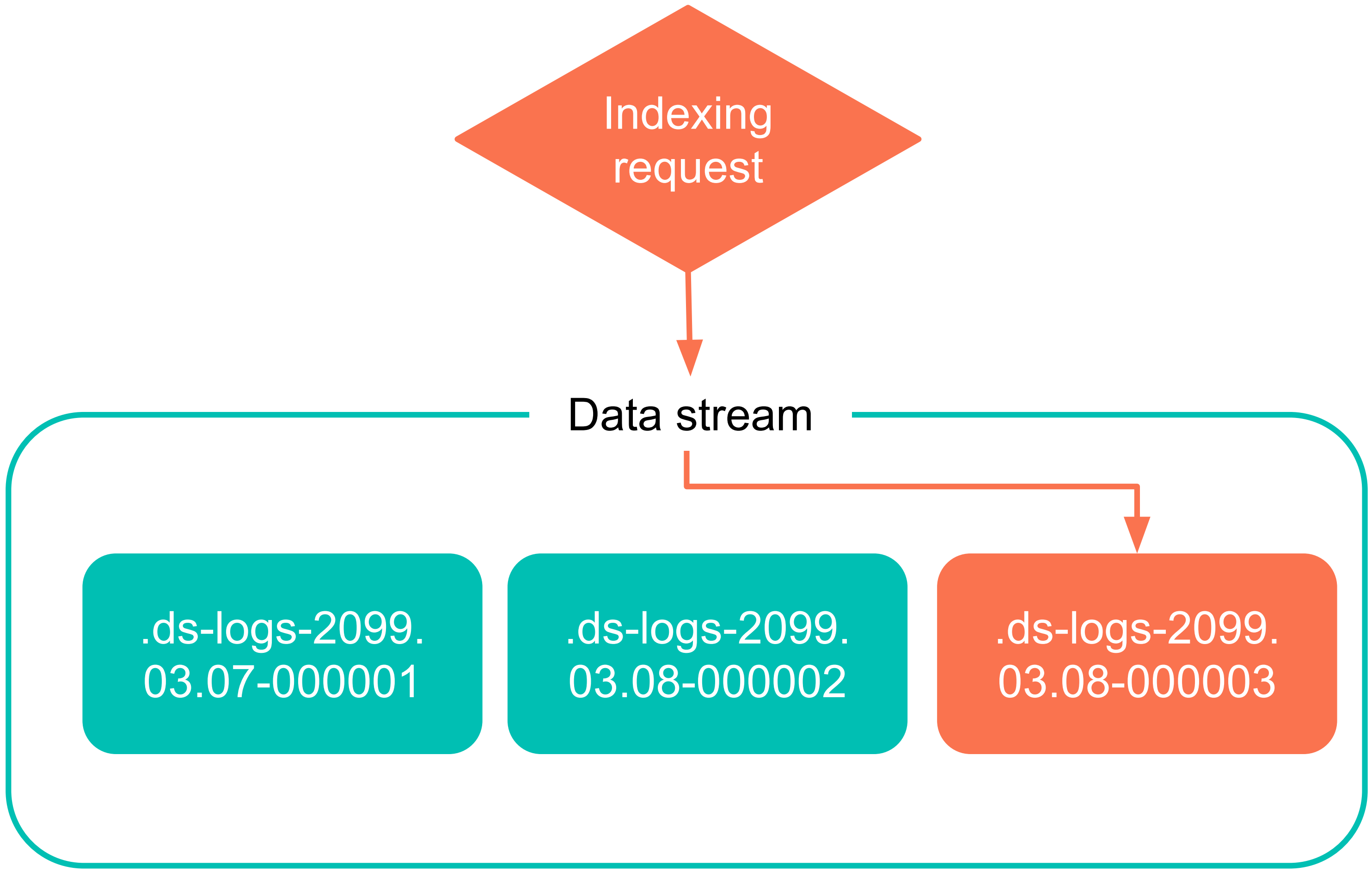
Data stream automatically creates (backing) indices with this naming pattern:

```
.ds-{data_stream}-{date}-{generation}
```

For example:

```
.ds-logs-myapp-auditlog-2023.09.03-000001
.ds-logs-myapp-auditlog-2023.09.04-000002
.ds-logs-myapp-auditlog-2023.09.04-000003
```


Search request
Data stream
.ds-logs-2099.03.07-000001
.ds-logs-2099.03.08-000002
.ds-logs-2099.03.08-000003


Indexing request
Data stream
.ds-logs-2099.03.07-000001
.ds-logs-2099.03.08-000002
.ds-logs-2099.03.08-000003

Data streams: examples

Some Examples:

Log Type	**Data Stream Name**
Requests	`logs-myapp-request`
Audit Log	`logs-myapp-auditlog`
Email	`logs-myapp-email`
SMS	`logs-myapp-sms`
Internal logs	`logs-myapp-log`

Index template

A data stream requires a matching “index template”.

For data streams, the index template configures the stream’s backing indices as they are created.

An index template is a way to tell Elasticsearch how to configure an index when it is created.

License! (more at the end)

Elasticsearch is dual-licensed since 2021:

- Elastic License: Proprietary, Source-available
- Server Side Public License (SSPL): Proprietary, Source-available
    - Introduced by MongoDB Inc. in 2018
    - Strong Copyleft, more copyleft than GNU Aferro GPL (AGPL)
    - Rejected by Open Source Initiative (called “fauxpen”)
    - Rejected by Debian, Fedora, RHEL
    - No comments from Free Software Foundation!
- Now Elasticsearch is just like MongoDB!

Kibana → Discover

Uses Kibana Query Language (KQL)

KQL only filters data, and has no role in aggregating, transforming, or sorting data.

Examples:

- `@timestamp < now-2w`
- `response_status >= 500`
- `logger_name: "django.server" and level_no > 30`

Kibana → Discover

More KQL Examples:

- http.request.method: (GET or POST or DELETE)
- remote_ip: 172.30.0.0/16 and object_type: "Branch"
- user:{ first: "John" and last: "Doe" }
- user.names:{ first: "John" and last: "Doe" }

Vector configuration

Components:

- Sources (inputs)
- Transforms (filters / modifiers)
- Sinks (outputs)
- Global options

https://vector.dev/docs/reference/configuration/

Vector config example

Format: TOML (Tom's Obvious, Minimal Language)

File: /etc/vector/vector.toml
Mount it as volume in docker-compose.yml

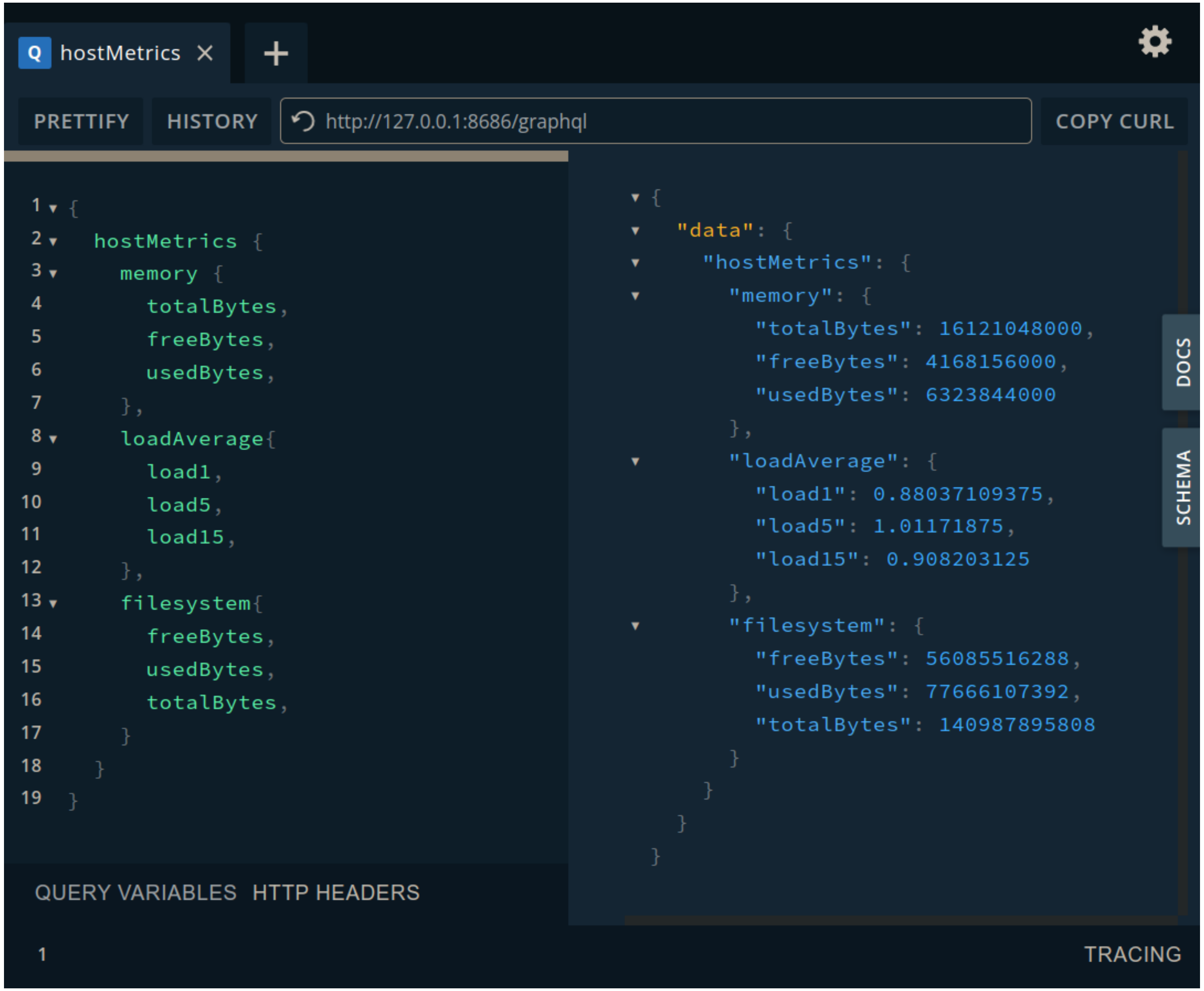
```
[sources.udp_input]
type = "socket"
address = "0.0.0.0:5958"
mode = "udp"
decoding.codec = "bytes"`
```

```
[transforms.json_parser]
type = "remap"
inputs = ["udp_input"]
drop_on_error = false
source = '''
parsed, err = parse_json(.message)
if err == null {
    . |= object!(parsed)
} else {
    log("Failed to parse json from udp_input", level: "error")
}
'''
```

```
[sinks.elastic_output]
type = "elasticsearch"
inputs = ["json_parser"]
healthcheck = false
api_version = "v7"
endpoints = ["http://elasticsearch:9200"]
auth.user = "${VECTOR_ELASTIC_USERNAME}"
auth.password = "${VECTOR_ELASTIC_PASSWORD}"
auth.strategy = "basic"
mode = "data_stream"
data_stream.dataset = "{{ tags[0] }}"
data_stream.namespace = "{{ type }}"
```

Vector API

- GET /health
- POST /graphql
- GET /playground



```
{
  hostMetrics {
    memory {
      totalBytes,
      freeBytes,
      usedBytes,
    },
    loadAverage{
      load1,
      load5,
      load15,
    },
    filesystem{
      freeBytes,
      usedBytes,
      totalBytes,
    }
  }
}
```

```
{
  "data": {
    "hostMetrics": {
      "memory": {
        "totalBytes": 16121048000,
        "freeBytes": 4168156000,
        "usedBytes": 6323844000
      },
      "loadAverage": {
        "load1": 0.88037109375,
        "load5": 1.01171875,
        "load15": 0.908203125
      },
      "filesystem": {
        "freeBytes": 56085516288,
        "usedBytes": 77666107392,
        "totalBytes": 140987895808
      }
    }
  }
}
```

Run It Yourself (RIY?)

Github repository with (almost) ready-to use configurations and scripts (needs some initial setup)

https://github.com/ParspooyeshFanavar/docker-elk

- Install docker and docker-compose
- Copy '.env.default' file to '.env'
- Edit '.env' (change passwords etc)
  - Don't put funny characters in passwords (bug in Bash script)
    - Best to use long alphanumerics (alphabet + numbers)
- Run: **`make init`**
- Run: **`make build`** and **`make up`**

License: why?

- Microsoft and Google were paying Elastic for their managed service
- Amazon launched it’s own managed service without paying Elastic, even using Elastic’s trademarks with no authorization. Basically taking advantage of it being Open Source. So Elastic changed it’s license!
- This does not effect other users / companies in practice (specially in Iran)
- SSPL is "Extreme Copyleft": those who offer the software to third-parties **as a service** must release the entirety of their source code, including all software, APIs, tools and other software that would be required for a user to run an instance of the service themselves.

License: Amazon!

- Amazon forked Elasticsearch (called Opensearch) under Apache license, and is trying to attract Open Source contributors (like free employeees), by acting like defenders of Open Source, and fish from the muddy water they helped create.
- Elasticsearch is not in the same boat as MongoDB who had the same story with Amazon! (and Amazon forked it under name of DocumentDB).
- Amazon keeps using both forks the same way! Even getting contributors from Open Source community!

License: Lessons

Lessons that we can learn:

- CLAs (Contributor License Agreement) allow the company to re-license your code without your consent to suit their needs.
- It’s always hard to pick a side between the bad and the worse.
- Big Tech normally steals (ideas, codes etc) from smaller companies as well as Open Source community!
- Cloud service providers challenge the fundumental concepts of Open Source and Free Software.