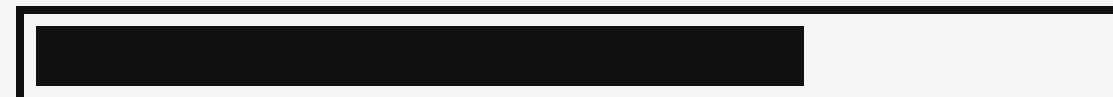




Event Logging story



© Sad & Fun

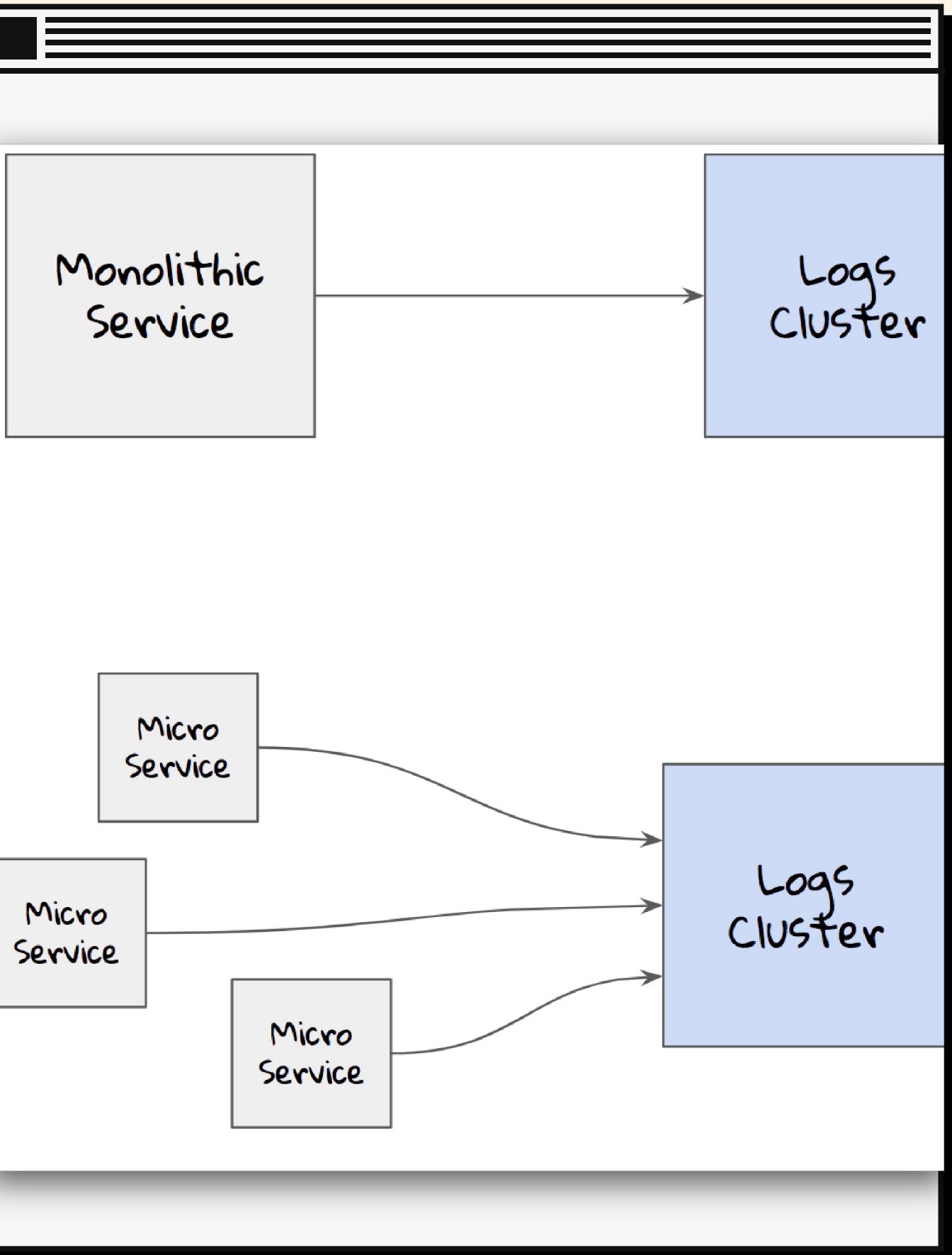


Once upon a time

There is a company who wants to adopt SRE practices.
And Logging - Tracing - Metrics came along to the picture

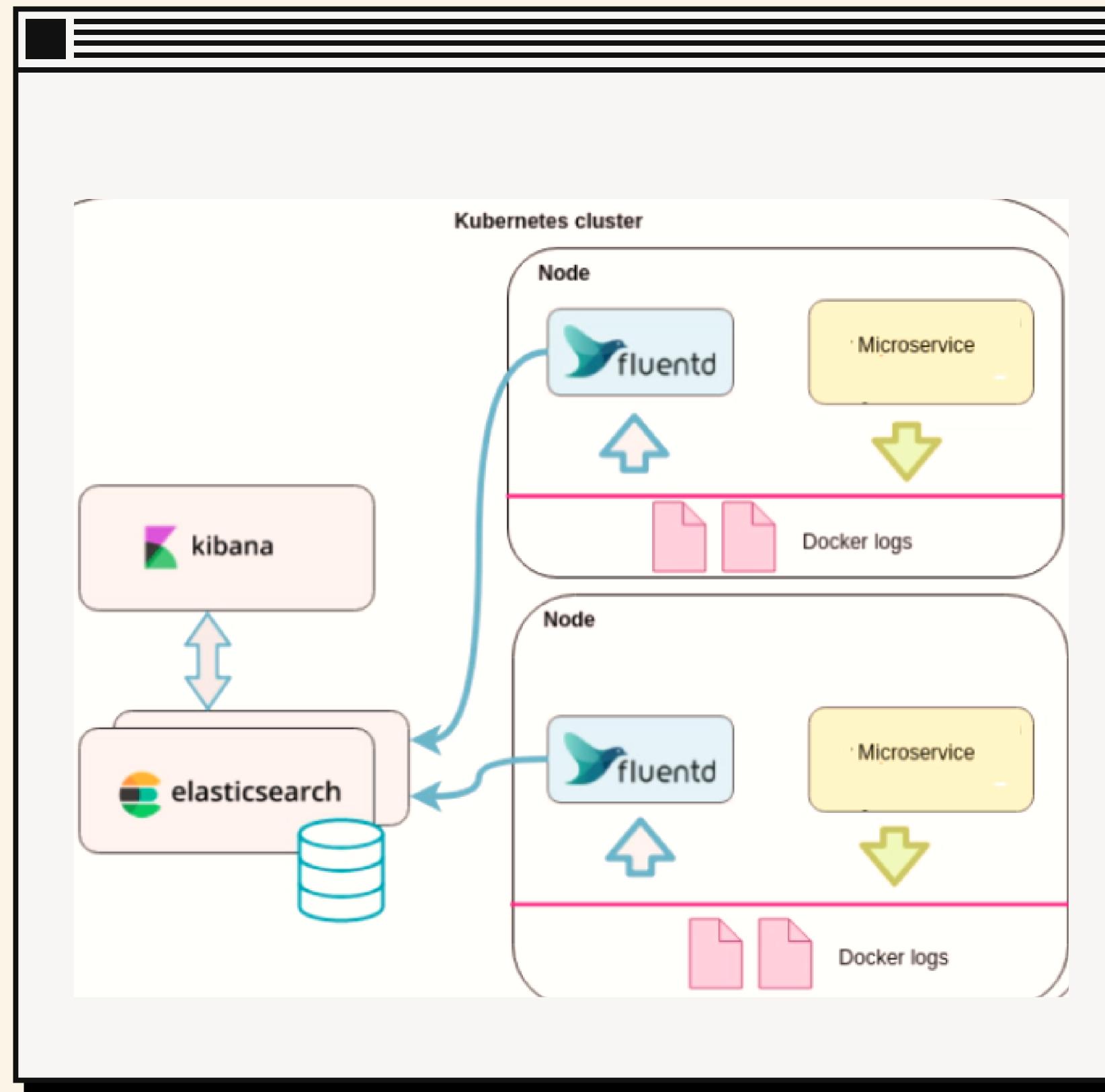
Good news is their architecture is quite simple

Bad news the simplicity is not always

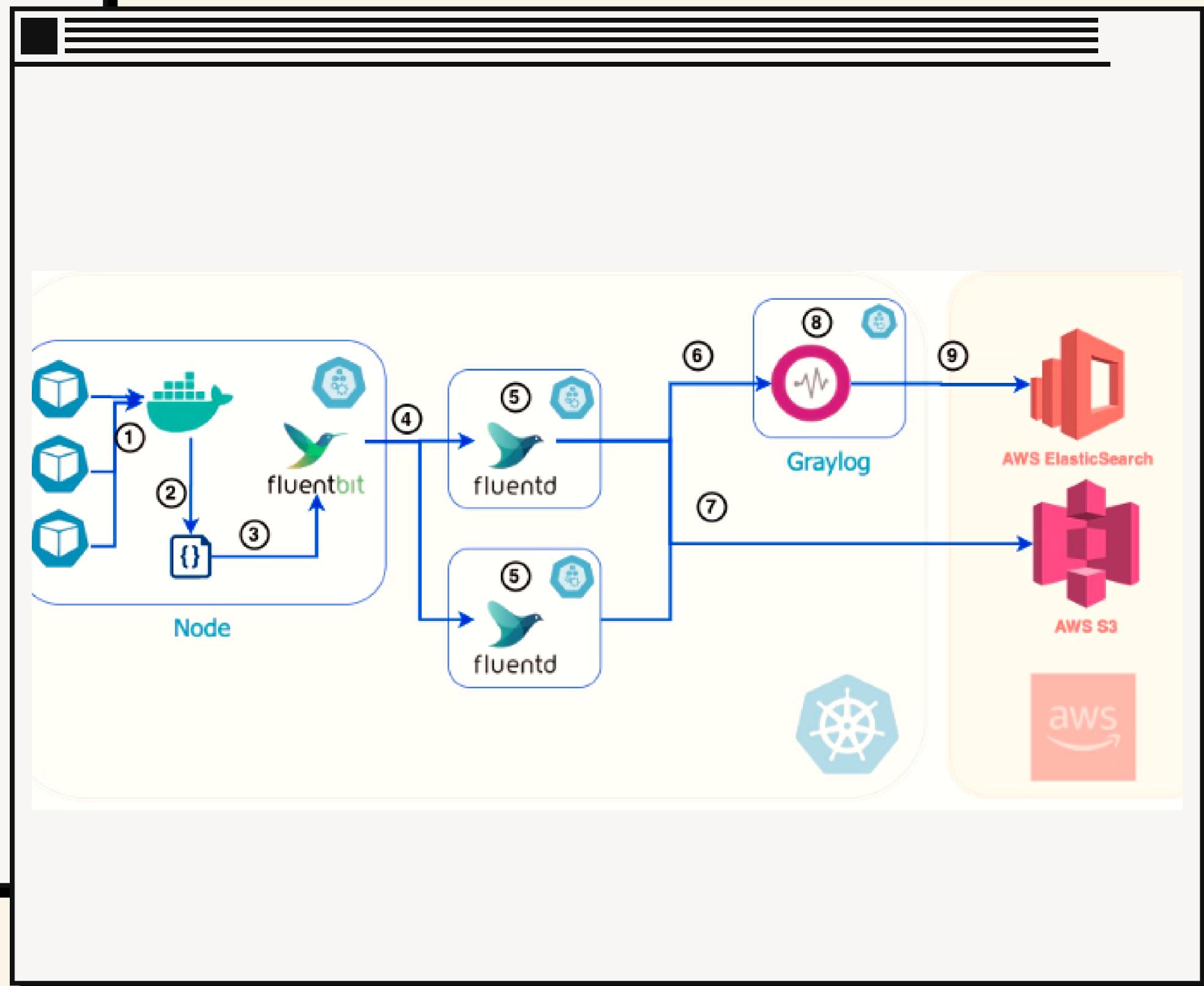


- How to normalise / standardise all the log events ?
- How to track all operations which are happening in side your logging pipeline ?
- How to distribute the core configuration to every engineers ?
- How to prevent the applications who try to abuse your logging pipeline ?

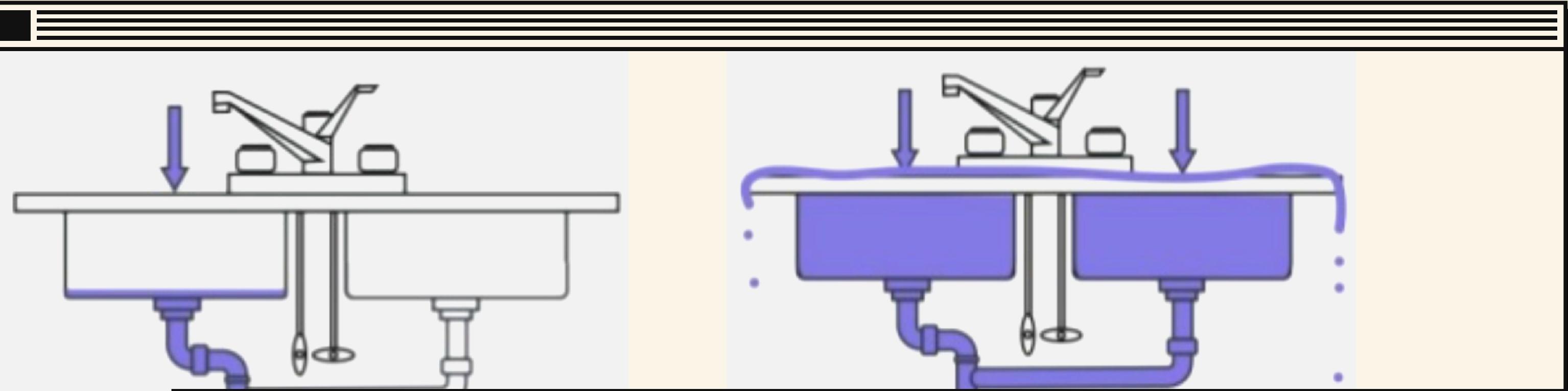
Leverage from logging application to the platform is a challenge



This is the old

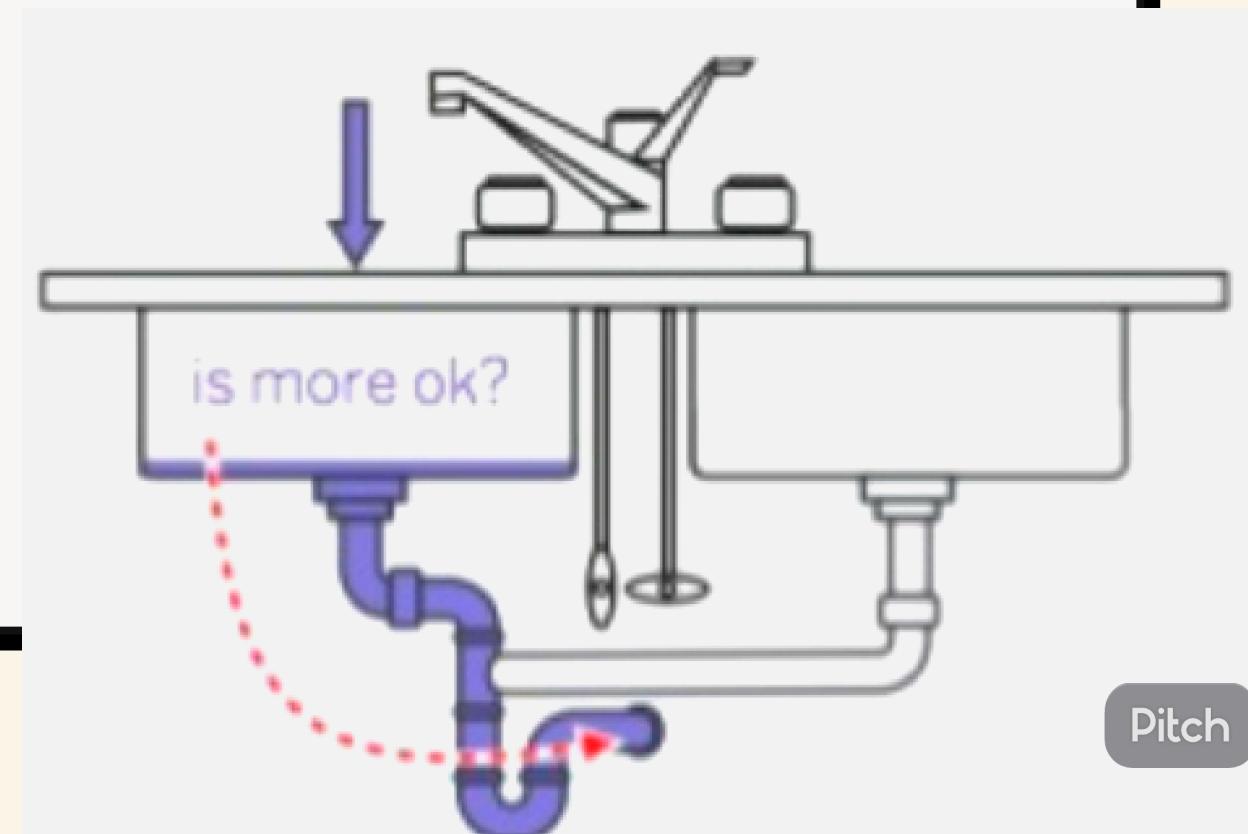


This is the new

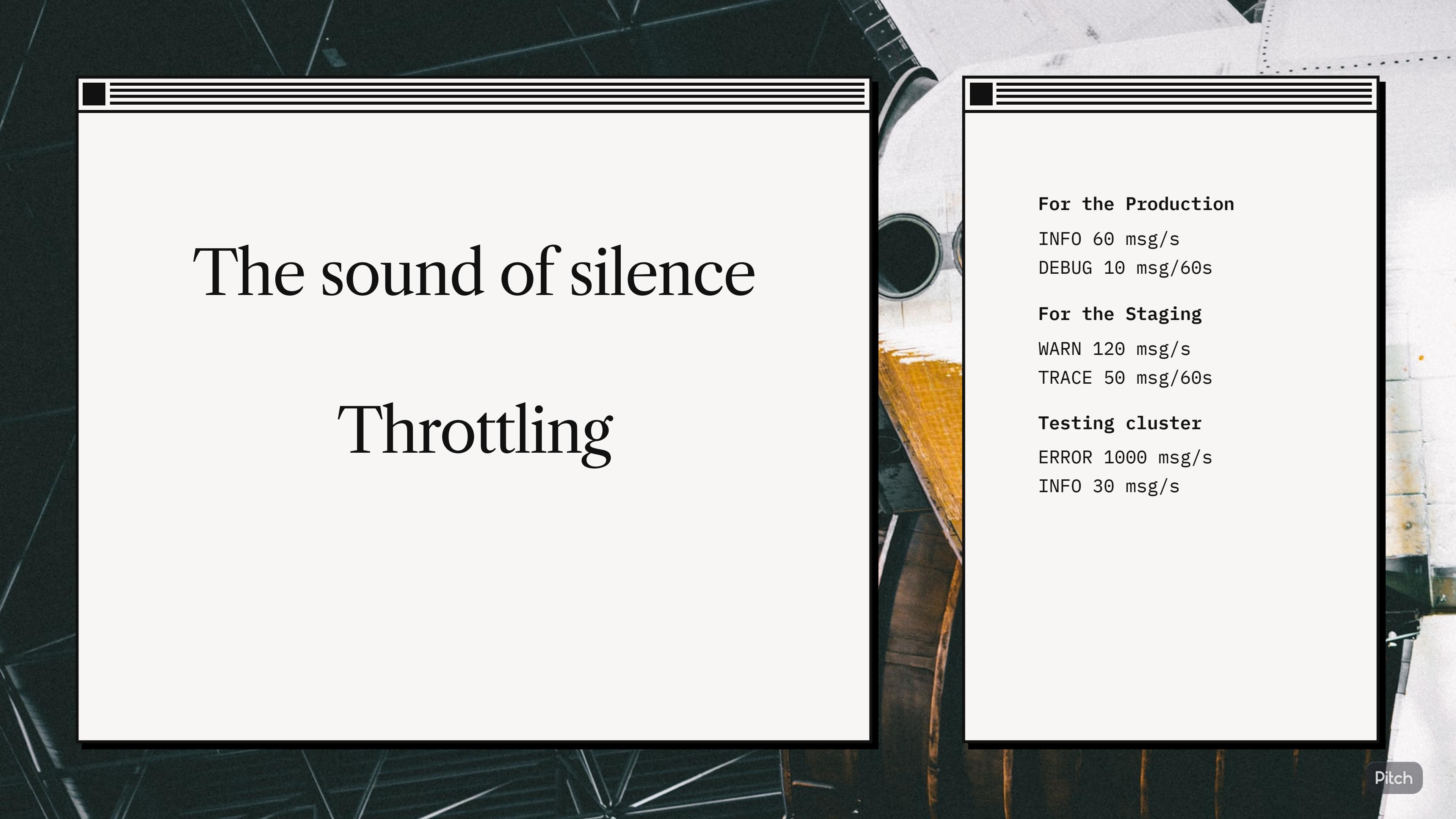


BackPressure mechanism

The main reason to use FluentBit is BackPressure mechanism which FluentD itself didn't have.



Pitch



The sound of silence

Throttling

For the Production

INFO 60 msg/s
DEBUG 10 msg/60s

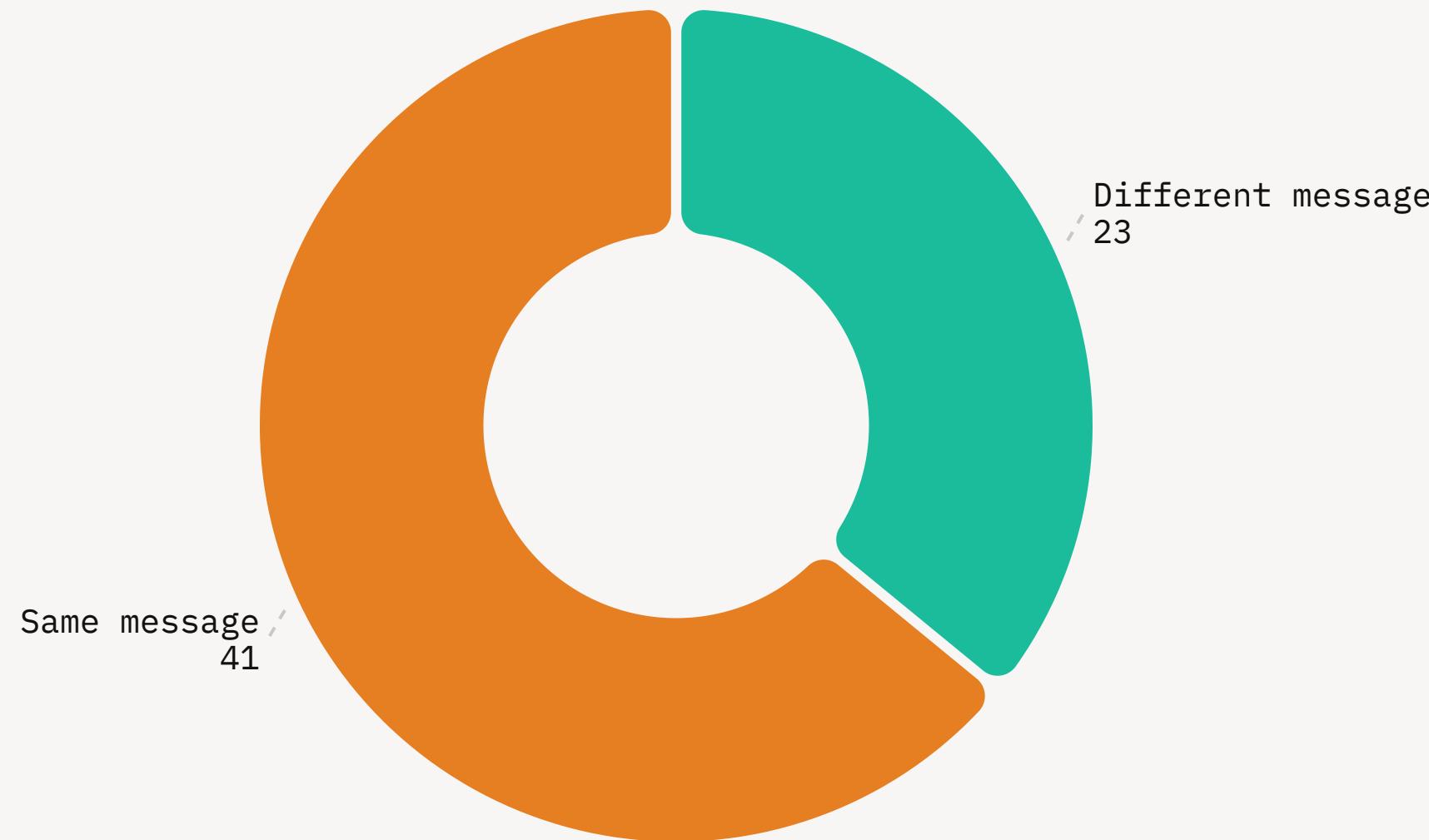
For the Staging

WARN 120 msg/s
TRACE 50 msg/60s

Testing cluster

ERROR 1000 msg/s
INFO 30 msg/s

Sampling



Mechanism

Suppress same messages

```
<match foo.*>
```

@type	suppress
interval	10
num	2
attr_keys	host,message

```
</match>
```

```
2020-11-22T11:22:33 foo.info
{"id":1,"host":"web01","message":"error!!"}
2020-11-22T11:22:34 foo.info
{"id":2,"host":"web01","message":"error!!"}
* 2020-11-22T11:22:35 foo.info
{"id":3,"host":"web01","message":"error!!"}
* 2020-11-22T11:22:36 foo.info
 {"id":4,"host":"web01","message":"error!!"}
2020-11-22T11:22:45 foo.info
 {"id":9,"host":"web01","message":"error!!"}
```

TRACING THE MESSAGE SIZE TO GET MORE VISIBILITY on STORAGE

Integrate Flow Counter & Prometheus plugin to achieve this



Distributed configuration

- FluentD introduces include syntax
 - <system>
 - @include teamA.conf
 - @include teamB.conf
 - @include teamC.conf
 - </system>

- Enable FluentD RPC
 - <system>
 - rpc_endpoint
 - 127.0.0.1:24444
 - </system>
 - \$ curl
 - http://127.0.0.1:24444/api/plugins.flushBuffer
 - s
 - {"ok":true}

- Utilise ConfigMap & MountPoint
 - kubectl get cm
 - NAME
 - fluentd-teamA.conf
 - fluentd-teamB.conf
 - fluentd-teamC.conf
 - DATA AGE
 - 1 8d
 - 1 8d
 - 1 8d

- Engineers now have full visibility on the logging pipeline

Log fields standardisation

Name	Format	Example	Description
Customer_Id	String	Alice-120	🐒 Monkey
Order_Id	Integer	12385910	🦏 Rhino
Trace_Id	Numeric	9292.14	🦩 Flamingo
Upstream_Host	String	http://checkout.teama.svc.cluster.local	🦄 Unicorn

Document all fields info to allow developers get references easier.

What if they don't follow ?

```
{"type":"mapper_parsing_exception","reason":"failed to parse field [duration] of type [float] in document with id 'da519231-1aaf-11eb-bb08-b22e566b37e4','caused_by":{"type":"number_format_exception","reason":"For input string: \"3.196501964s\""}}
```



ElasticSearch well known issue

FluentBit filtering by



The filter will check each log field to see if it matches the data type in our list. If it matches, it will leave it as is.
If it does not match, it will either just force/cast the value to a string value, or in case of a required numeric field it will try to parse it to a number

- The weakness thing is the strongest

FluentD Multiple Workers is ready to use

<system>

 workers 10

</system>