APR 16TH, 2020

RAFAŁ BAJEK

EFFECTIVE LOG MANAGEMENT WITH ELASTICSEARCH

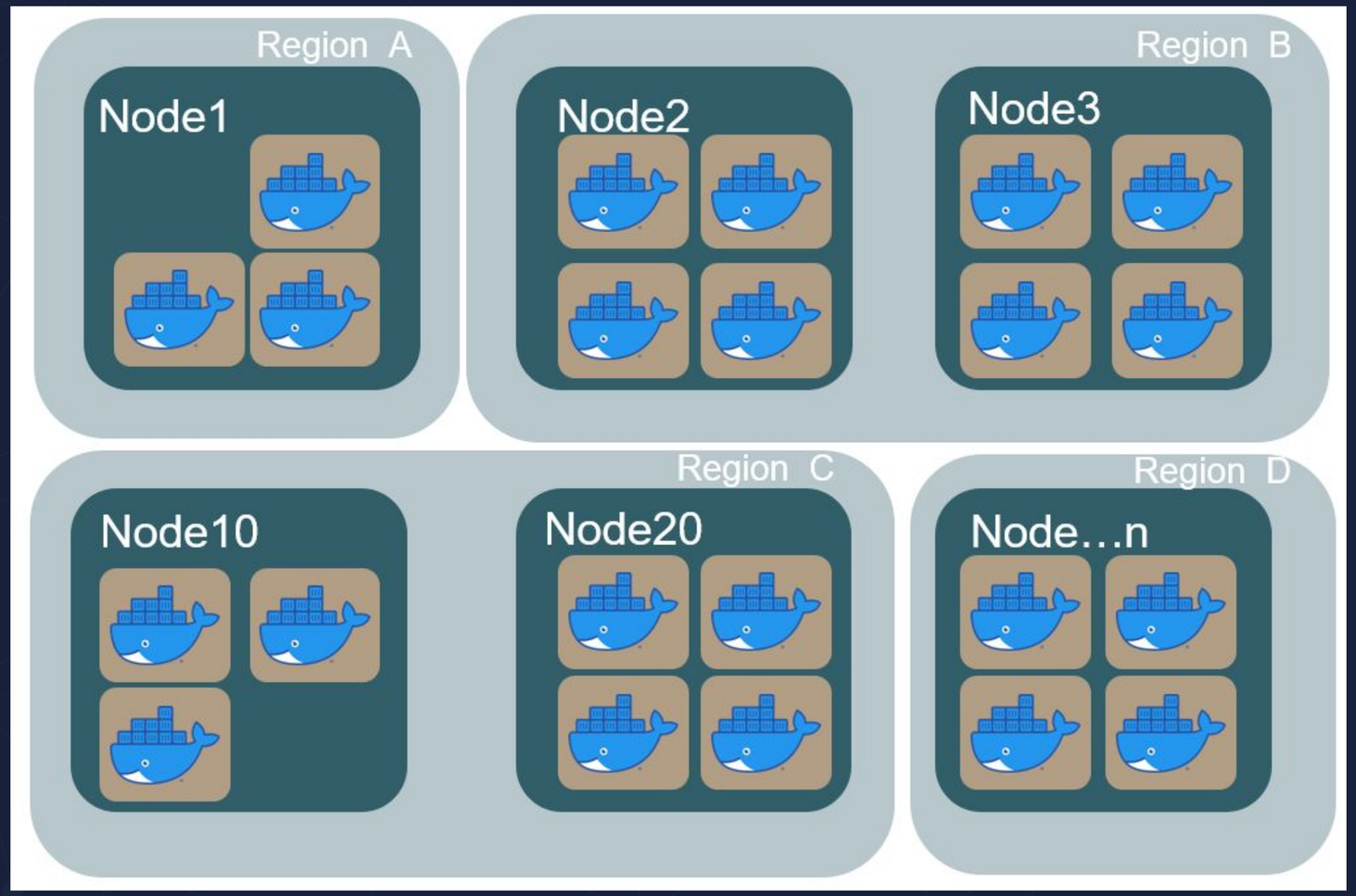
Cometari

INTRODUCTION

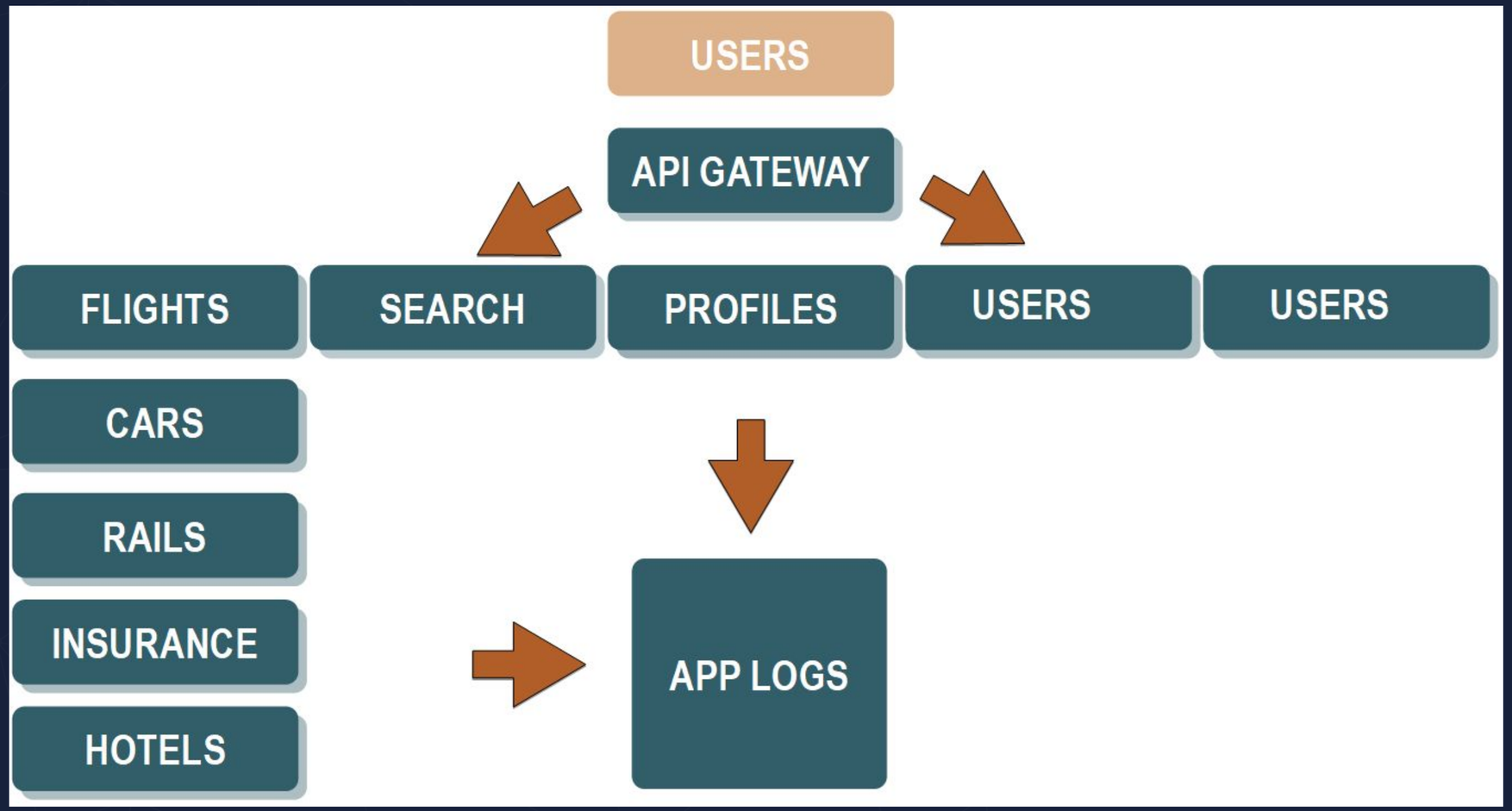
- I am the System Architect and System Engineer with more than 15 years experience
- Cometari is a solutions company implementing DevOps culture and providing consultancy, workshops and software services.
- Our expertise are DevOps, Distributed Systems, Elastic Stack - log analysis,
- We are deeply involved in the travel tech industry
- However our solutions go much further than just integrating travel API's.

My goal is to show you how to build an effective and efficient log management system based on Elasticsearch.

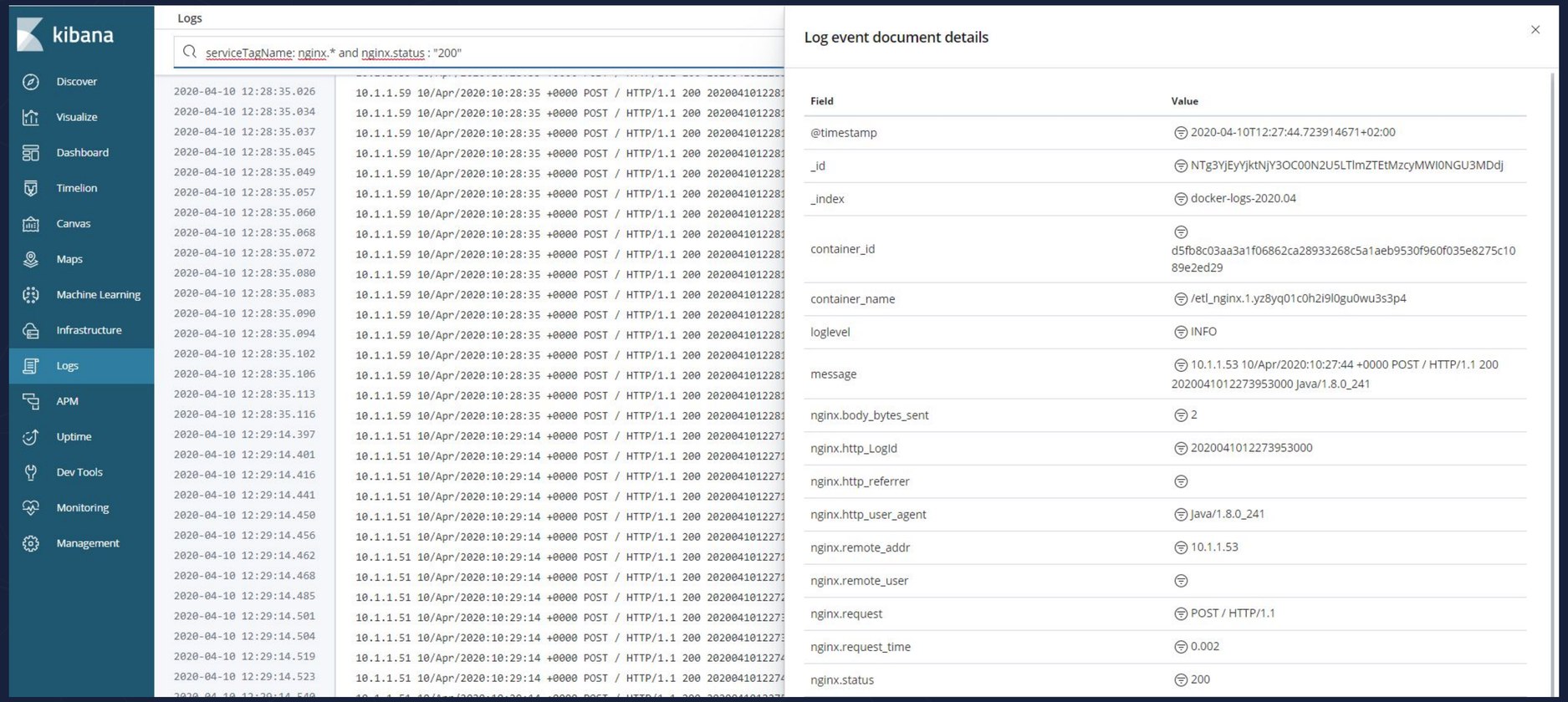
CHALLENGES OF LOG ANALYSIS TODAY



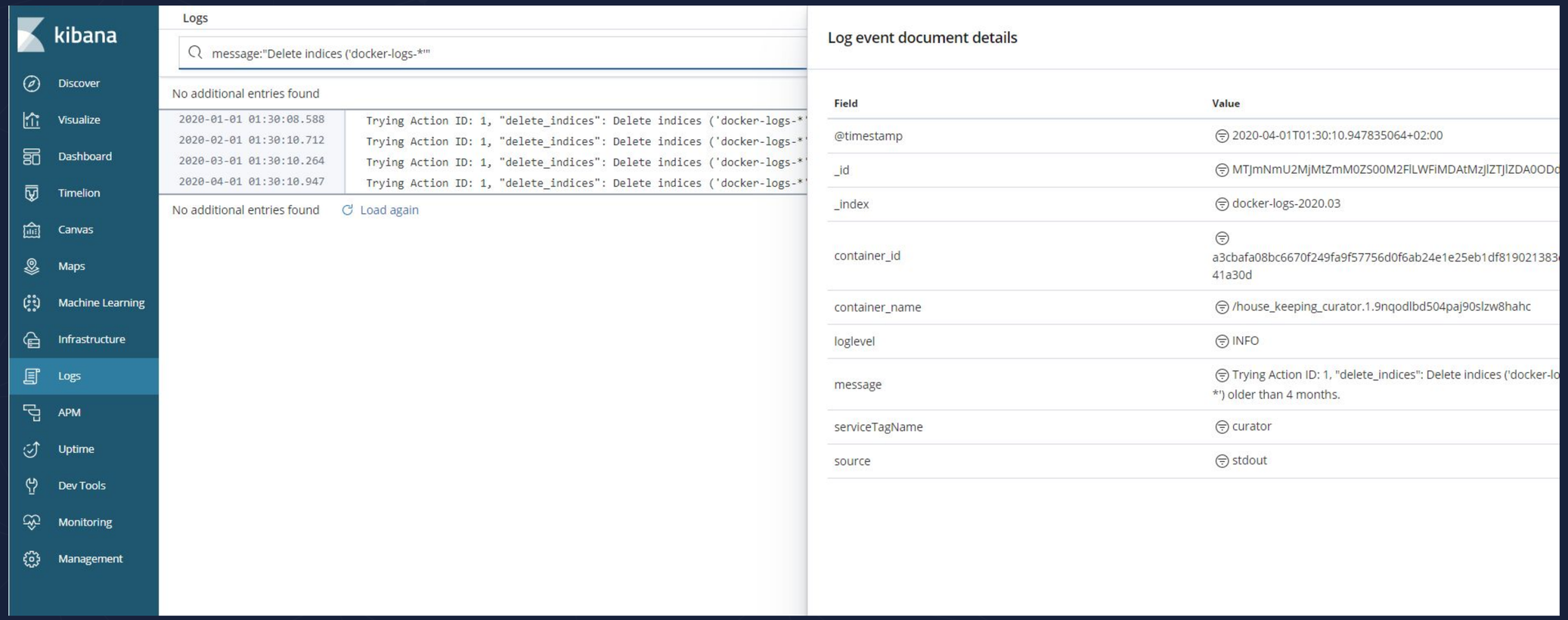
CHALLENGES OF LOG ANALYSIS TODAY



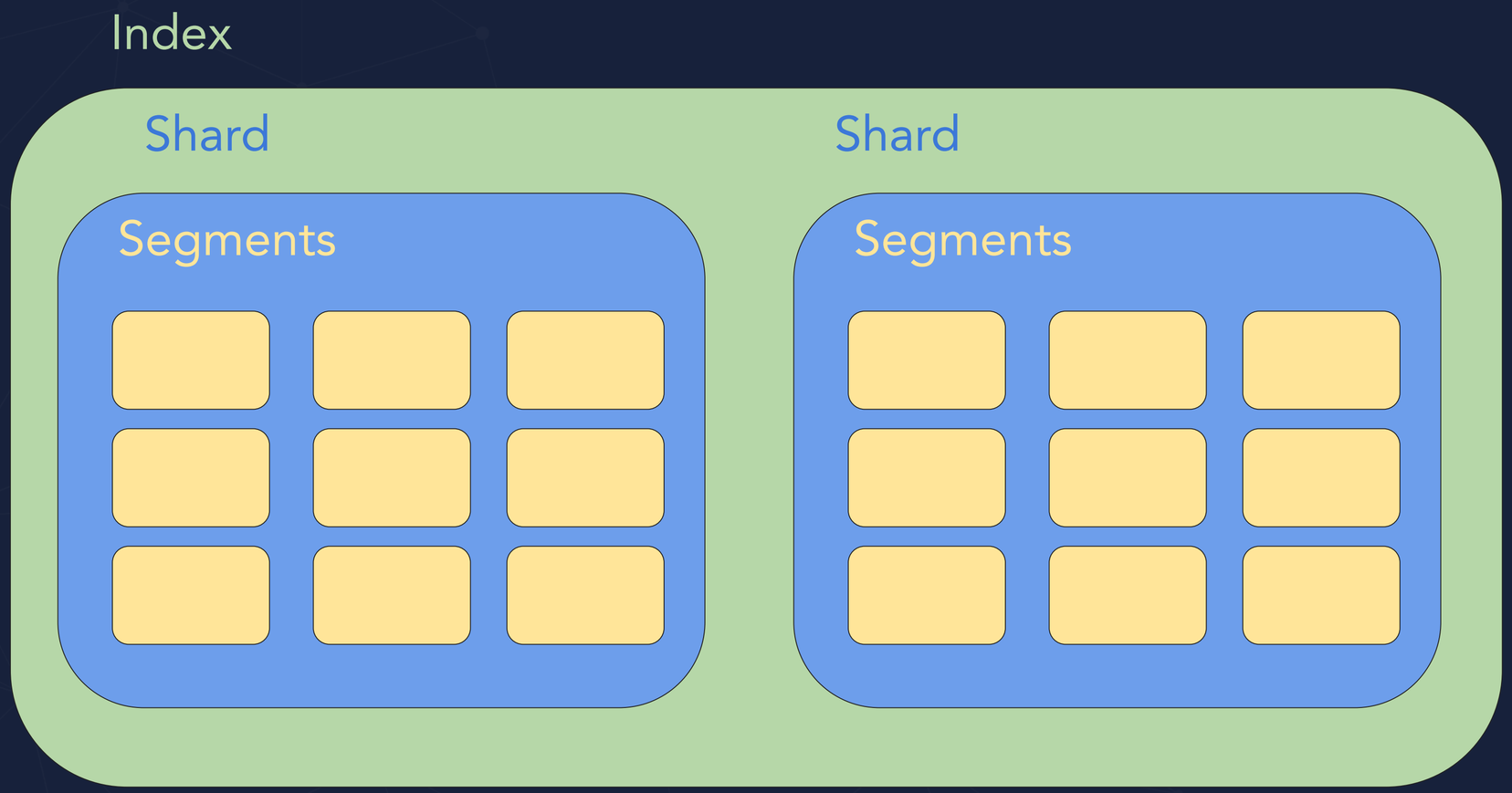
CHALLENGES OF LOG ANALYSIS TODAY



CHALLENGES OF LOG ANALYSIS TODAY



INDICES, SHARDS, SEGMENTS



Logs - good practises

- Normalize your logs (e.g. use the JSON structure)
- Use the same time zone in logs (within the entire services)
- Take care about the sensitive data
- Use the correlation identifiers (very useful in distributed systems)
- Log only important/useful data

PLANNING CLUSTER AND DESIGNING INDICES

- What your logs look like
- The logs size: more or less constant or various
- Optimal size of shard
- How many primary shards/replicas?
- Indices: daily, weekly, monthly, fixed size?
- How you are going to search your data?
- Index templates
- Aliases

Index template

- refresh_interval
- disable dynamic mapping
- use the right data type

Schema for common logs

Field name	data type	mandatory
@timestamp	date	true
message	text	true
serviceTagName	text/keyword	true
loglevel	keyword	true
traceId	keyword	false
spanId	keyword	false
<specific_service>	JSON object	false

Index template

```
{
    "index_patterns": ["docker-logs-*"],
    "version": 1,
    "settings": {
        "number_of_shards": 1,
        "number_of_replicas": 0,
        "refresh_interval": "10s"
    },
    "mappings": {
        "_doc": {
            "dynamic": "false",
            "properties": {
                "@timestamp": {
                    "type": "date"
                },
                "message": {
                    "type": "text"
                },
                "serviceTagName": {
                    "type": "text",
                    "fields": {
                        "keyword": {
                            "type": "keyword"
                        }
                    }
                },
                "loglevel": {
                    "type": "keyword"
                },
                "nginx": {
                    "properties": {
                        "http_LogId": {
                            "type": "text",
                            "fields": {
                                "keyword": {
                                    "type": "keyword"
                                }
                            }
                        },
                        "status": {
                            "type": "keyword"
                        }
                    }
                }
            }
        }
    },
    "aliases": {
        "docker-logs": {}
    }
}
```

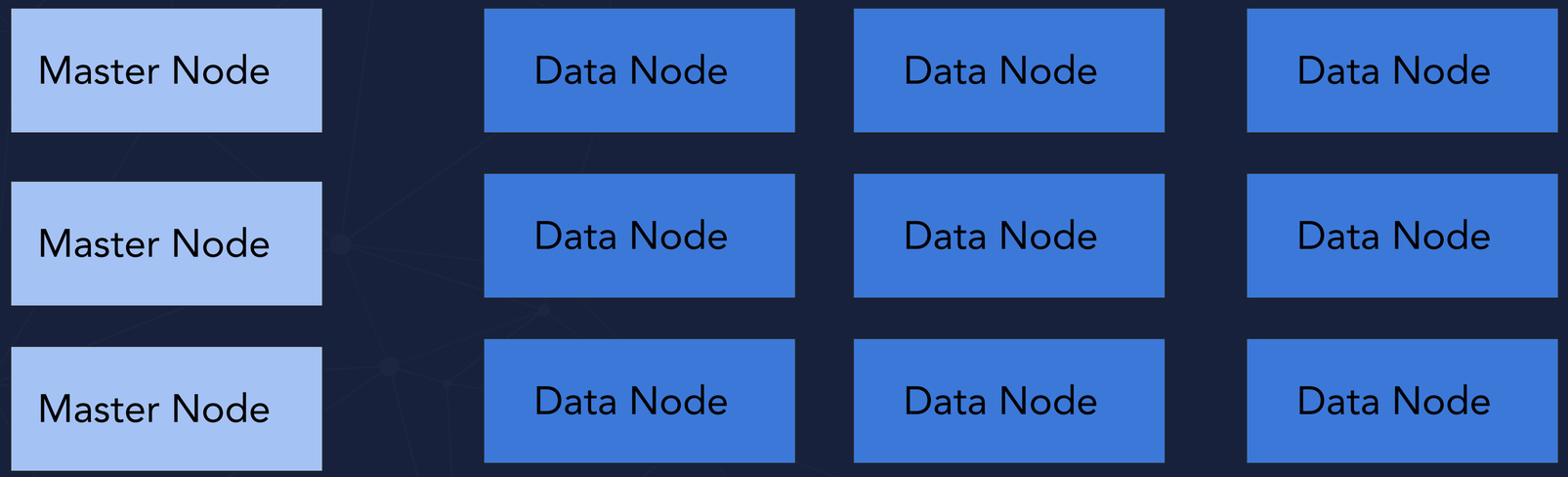
How to start?

- create a initial version of your index (based on planning)
- start indexing your data
- start monitoring size of your shards, resources, cluster
- check the queries/aggregations speed
- tune your cluster/indices/settings
- test and validate your solution!

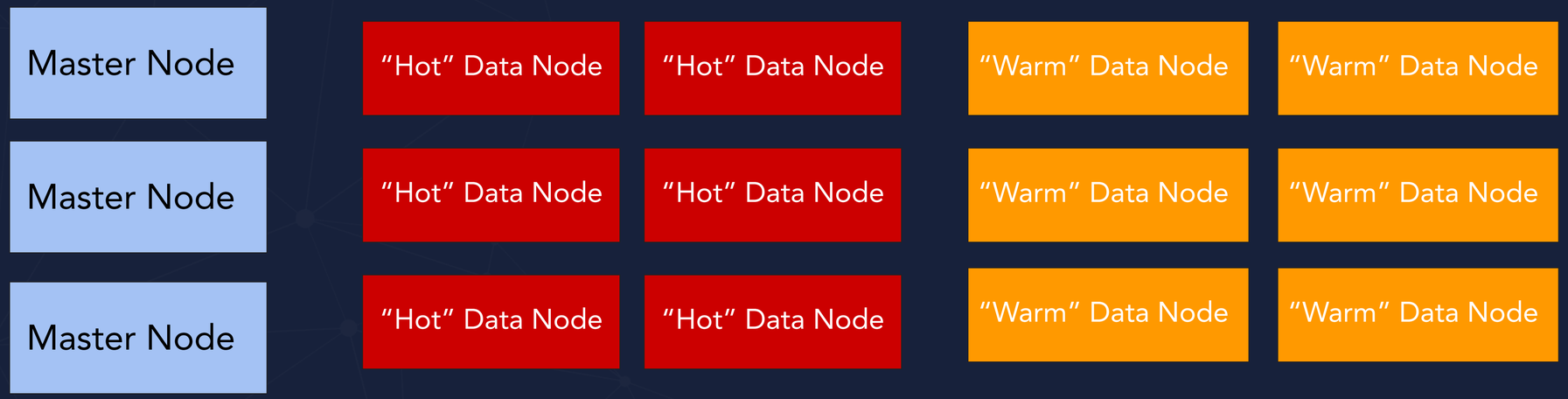
Start using your first configuration on production, monitor your system and tune the settings if necessary. Your cluster should live and should be constantly improved.

ARCHITECTURES OF DEPLOYMENTS

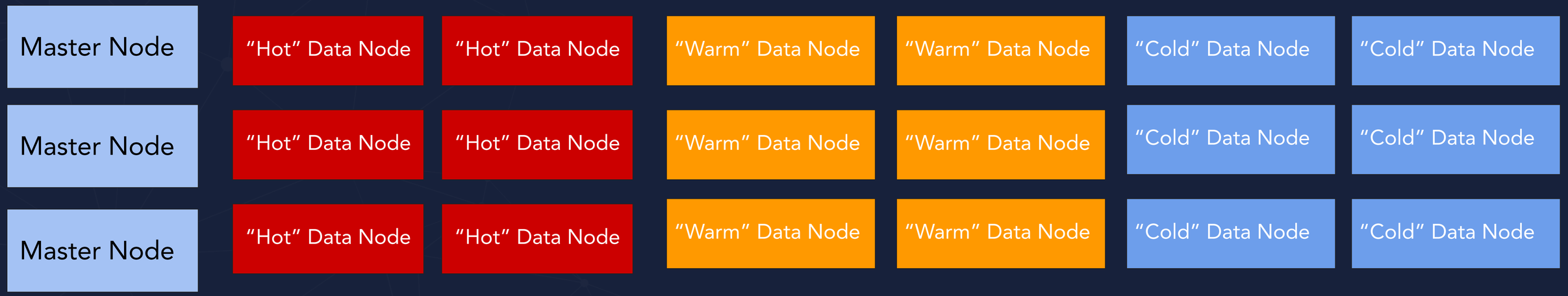
Uniform architecture



Hot-warm architecture



Hot-warm-cold architecture



BUILDING AN EFFICIENT LOG SYSTEM AT SCALE - FOR FREE!

Split your indices to “hot” and “warm”

- “hot” indices - the latest ones: writable and searchable
- “warm” - read-only, contains older logs
- the right amount of the primary shards (more for “hot”, less for “warm”)
- splitting based on “writing time range”

Shrink the shards

- move one copy of shards to a single node

```
PUT index_name/_settings
{
    "index.blocks.write": true,
    "index.routing.allocation.require._name": "some_node_name"
}
```

- do the shrink

```
POST index_name/_shrink/index_name-warm
{
    "settings": {
        "number_of_shards": 1,
        "number_of_replicas": 0,
        "codec": "best_compression",
        "index.blocks.write": true,
        "index.routing.allocation.require._name": null
    }
}
```

Force a merge of segments

```
POST index_name-warm/_forcemerge?max_num_segments=1
```

Increase the number of replicas

```
PUT index_name-warm/_settings
{
    "number_of_replicas": 1
}
```

Switch aliases

```
POST _aliases
{
  "actions": [
    {
      "remove": {
        "index": "index_name",
        "alias": "logs-view"
      }
    },
    {
      "add": {
        "index": "index_name-warm",
        "alias": "logs-view"
      }
    }
  ]
}
```

Remove the "hot" indices

```
DELETE index_name
```

CURATOR - MANAGING TOOL

CURATOR -

- crontab_jobs
- curator.yml
- delete_docker_logs_indices_action.yml
- delete_indices_action.yml
- delete_missing_logs_indices_action.yml
- rollover_missing_logs_indices_action.yml
- shrink_force_marge_indices_action.yml

CURATOR -

```yaml
actions:
  1:
    action: delete_indices
    description: >-
       Delete indices ('docker-logs-*') older than 4 months.
    options:
      ignore_empty_list: True
      continue_if_exception: True
      disable_action: False
    filters:
    - filtertype: pattern
      kind: prefix
      value: docker-logs-
      exclude:
    - filtertype: age
      source: name
      direction: older
      timestring: '%Y.%m'
      unit: months
      unit_count: 4
```

SUMMARY

- Take time for good planning
- Normalize your logs
- Use the same time zone for all logs (UTC is more than welcome)
- You must know your logs
- The right balance between the number and size of shards
- Choose the right solution architecture
- Monitor your cluster (shard size, heap space size, etc.)
- Automate the managing your logs (e.g. curator)
- Don't forget to add alerting!

RAFAL.BAJEK@COMETARI.COM

THANK YOU

Cometari