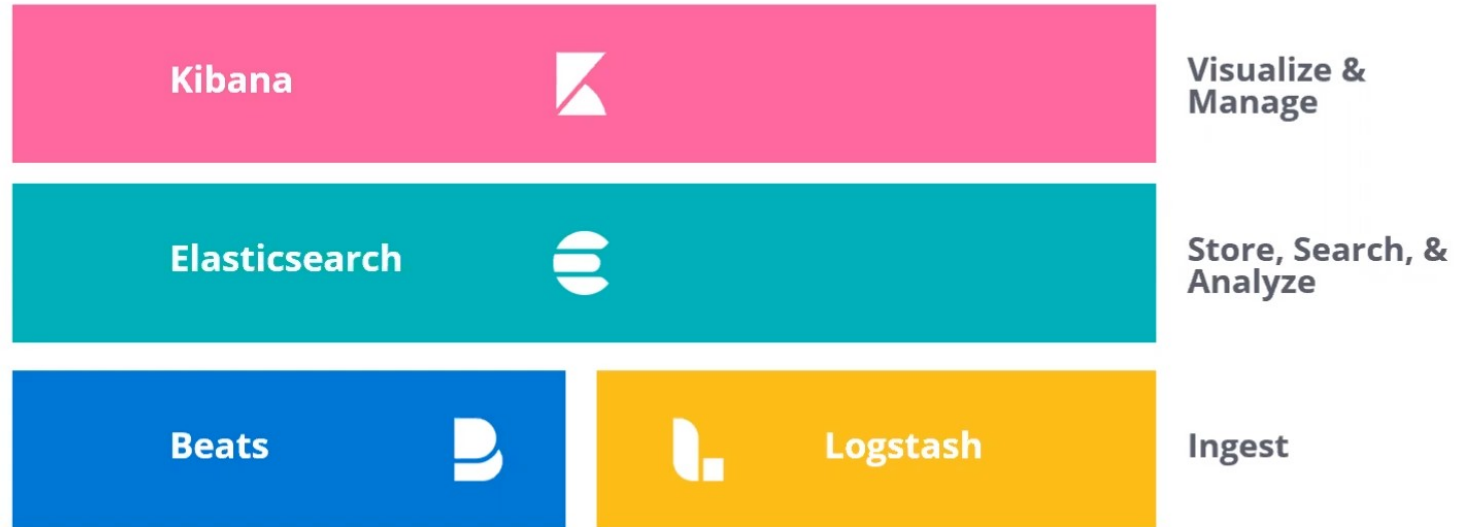


FileBeat / Logstash / Fluentd
+ Elasticsearch + Kibana

Elastic Stack

SOLUTIONS



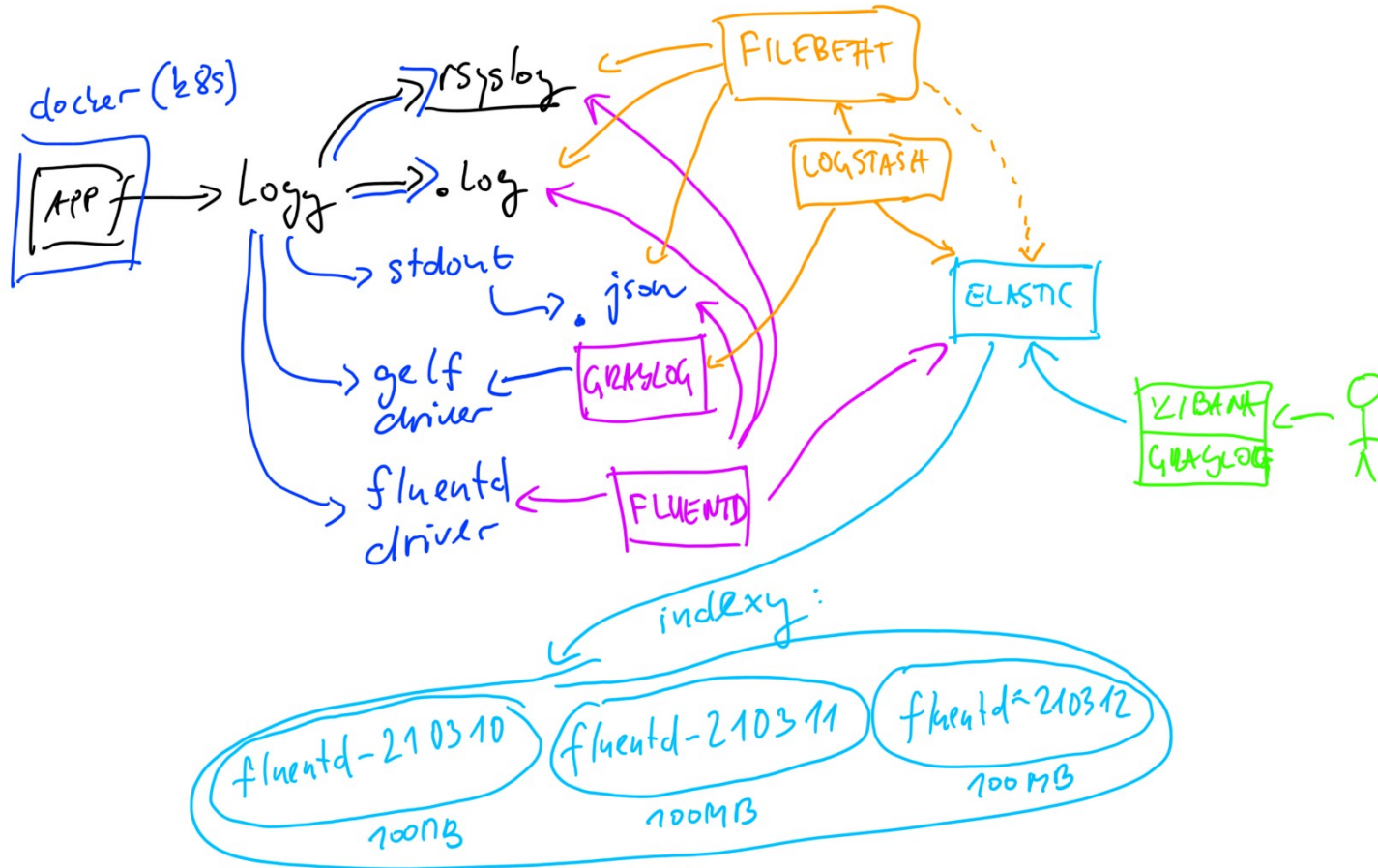
SaaS



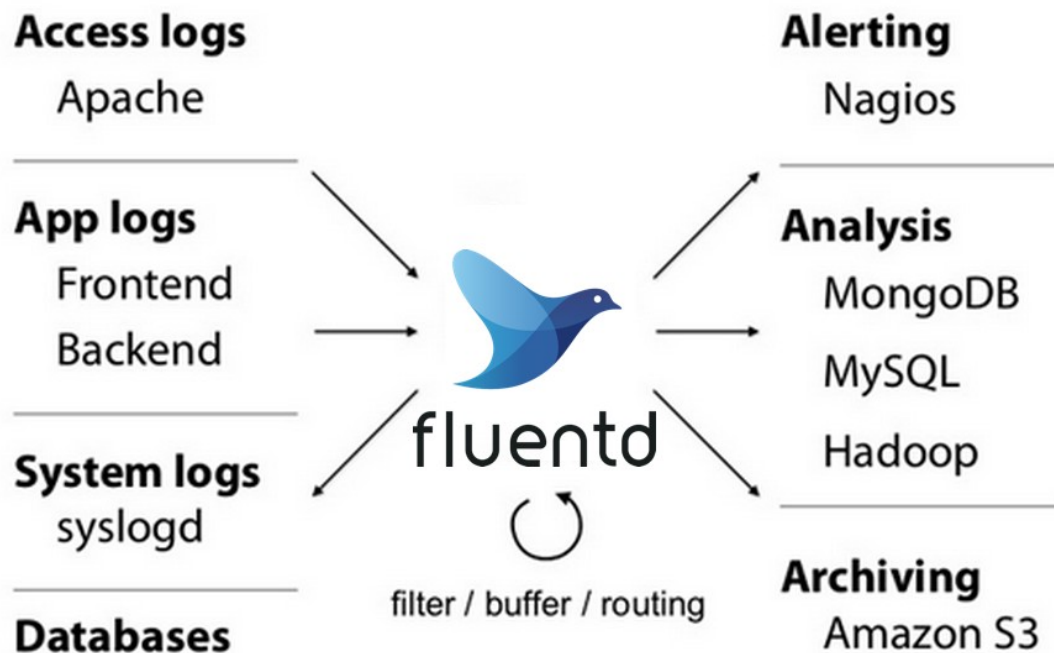
SELF-MANAGED



Elastic stack



Fluentd



Fluentd struktura conf. souboru

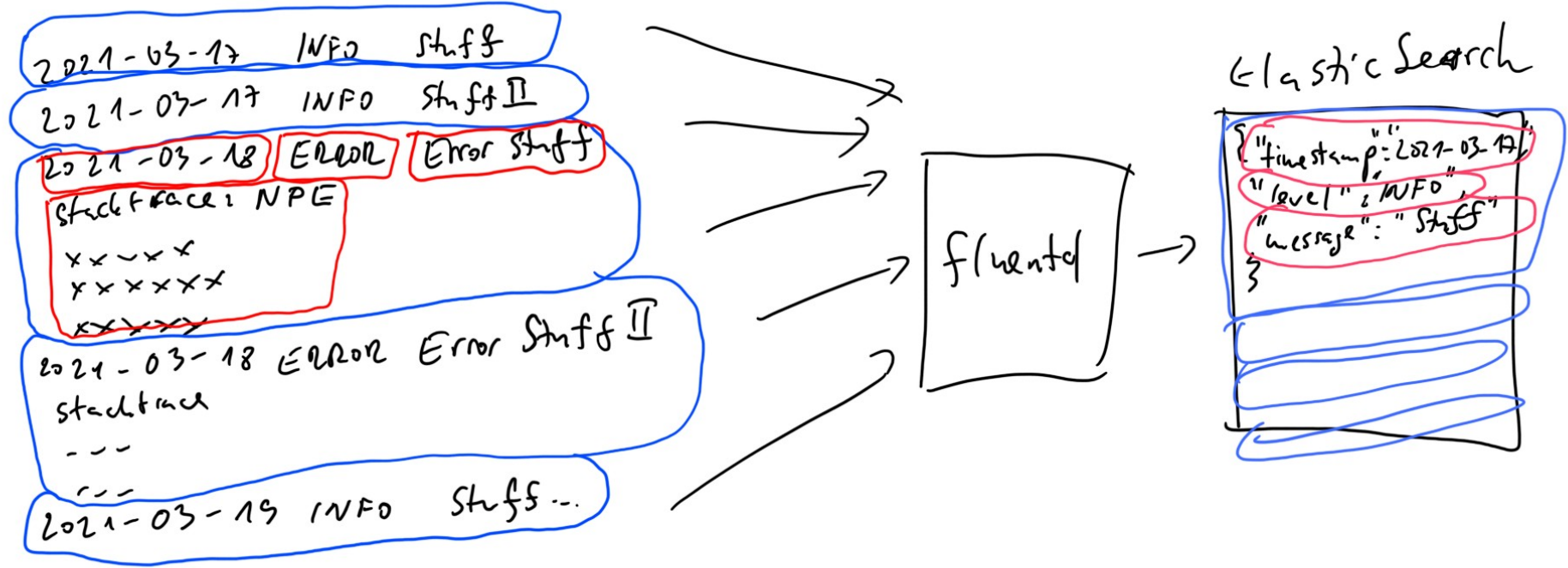
- `source` = input (vstup), může být větší množství vstupů
 - `parse` = v jakém formátu jsou vstupní data
- `filter` = filtrace a transformace vstupu, může být větší množství filtrů, na pořadí filtrů záleží!
- `match` = output (výstup), může být větší množství výstupů, na pořadí záleží!
 - <https://docs.fluentd.org/configuration/config-file#note-on-match-order>
 - `buffer` = bufferování dat při výstupu
 - `format` = formátování výstupu
- <https://docs.fluentd.org/configuration/config-file>

Umístění souboru: <https://docs.fluentd.org/configuration/config-file#config-file-location>

Fluentd buffer

- U bufferu je důležité nastavení:
 - flush_interval: Jak často se bude buffer vyprazdňovat (standardně 60s)
 - chunk_limit_size: Maximální velikost chunku
- Buffer je množina chunků.
- <https://kkc.github.io/2019/05/02/ultimate-note-for-tweaking-fluentd-aggregator/>

Fuentd multiline



Fluentd Tag & Label

- Každý vstup má:
 - Tag: String, ve kterém může být použita tečka, například: `myapp.access`
 - Time: Unix time
 - Record: JSON objekt
- Tagy se typicky používají v: `filter` & `match`, aby se příslušná konfigurace použila pouze pro konkrétní vstup.
- Label je podobný koncept jako tag a umožňuje “fine-grained” nastavení routování uvnitř konfigurace:
 - <https://docs.fluentd.org/configuration/config-file#5-group-filter-and-output-the-label-directive>
 - Existuje speciální label `@ERROR`, který umožňuje routovat Fluentd errorry někam jinam.

Fluentd parser & timestamp

- Často používané atributy:
 - time_key (klíč kde se nachází timestamp)
 - <https://docs.fluentd.org/configuration/parse-section#parse-parameters>
 - timezone (časová zóna)
 - <https://docs.fluentd.org/configuration/parse-section#time-parameters>
 - time_format (formát, ve kterém je čas v time_key)
 - 2021-03-30 17:23:35.018 ~ %Y-%m-%d %T.%L
 - Online tester formátů: <http://strftime.net/>

Fluentd @include

- Není zapotřebí mít jenom jeden fluent.conf soubor, ve kterém by byla veškerá konfigurace. Je možné mít konfiguraci ve více souborech a pomocí @include ji do fluent.conf připojit:
 - <https://docs.fluentd.org/configuration/config-file#6-reuse-your-config-the-include-directive>

Fluentd + Docker

- Docker obsahuje fluentd log driver, tudíž integrace je velice jednoduchá:
- `docker run --log-driver=fluentd --log-opt fluentd-tag=docker.appname`
 - <https://www.fluentd.org/guides/recipes/docker-logging>
- Pro transformaci log message se dá použít filter @type parser
- NEBO:
 - Aplikace může zapisovat logy do .log souboru a pak se pro jejich čtení dá použít multiline parser

Fluentd + Kubernetes

- <https://docs.fluentd.org/container-deployment/kubernetes>

Fluentd Grok parser

- Fluentd plugin pro parsování pomocí Grok formátu (nativní formát pro Logstash):
 - <https://github.com/fluent/fluent-plugin-grok-parser>
- Grok online debugger:
 - <https://grokdebug.herokuapp.com/>
- Další Grok debugger je v Kibana: Management → Dev Tools → Grok Debugger

Fluentd + multiline

- Fluentd multiline parser funguje pouze se vstupem typu tail (data se čtou ze souboru).
- Fluentd + Docker / k8s + Java aplikace & multiline support:
 - Buď zapisovat data do souboru (pak se použije multiline parser), nebo je možné použít fluent concat plugin & parser plugin:
 - <https://arnoldgalovics.com/java-and-spring-boot-multiline-log-support-for-fluentd-efk-stack/>

Testování regulárních výrazů

- Užitečný online nástroj pro testování regulárních výrazů:
 - <https://regexr.com/>

Docker & Elastic & Kibana

docker-compose.yml:

```
version: '3.7'
services:
  elastic:
    image: elasticsearch:7.3.0
    ports:
      - "9200:9200"
      - "9300:9300"
    environment:
      discovery.type: single-node
  kibana:
    image: kibana:7.3.0
    ports:
      - "5601:5601"
    environment:
      ELASTICSEARCH_HOSTS: http://elastic:9200
```


ES Terminologie

- Cluster: Skupina ES uzlů
- Uzel (Node): Java proces, ve kterém běží ES
- Index: Skupina shardů, které dohromady tvoří datové úložiště
- Shard: Lucene index, kde jsou fyzicky uložena data
- Segment: Immutable část Lucene indexu. Segment se vytvoří jakmile chceme něco do Lucene indexu zapsat.

- <https://fdv.github.io/running-elasticsearch-fun-profit/003-about-lucene/003-about-lucene.html>

- Document: Konkrétní záznam (record), který můžeme do ES uložit / z ES získat

Elasticsearch Index							
Elasticsearch shard		Elasticsearch shard		Elasticsearch shard		Elasticsearch shard	
Lucene index		Lucene index		Lucene index		Lucene index	
Segment	Segment	Segment	Segment	Segment	Segment	Segment	Segment

Elasticsearch & shardy

- Od ES 7 je out-of-the-box 1 primary shard a 1 replika pro každý index (dříve to bylo 5 shardů a také byla 1 replika pro každý index).
- Větší množství replik je pro zrychlení dotazů (dotazy se mohou vykonávat v jakékoli replice) a failover. Počet replik je možné po vytvoření indexu zvyšovat i snižovat.
- Větší množství Shardů je pro zrychlení zápisů. Nepříjemné je, že počet shardů se definuje při vytváření indexu a není možné ho za běhu lehce změnit.
- <https://qbox.io/blog/optimizing-elasticsearch-how-many-shards-per-index>

Elasticsearch

- Elasticsearch obsahuje indexy (v principu index ~ tabulka v relační databázi).
- V indexech se nacházejí dokumenty (dokument ~ řádek v tabulce v relační databázi). Dokumenty jsou obvykle JSON soubory.


Vytvoření indexu (create)

PUT <http://localhost:9200/movies>

```
{
  "mappings" : {
    "properties" : {
      "name" : {
        "type" : "text"
      }
    }
  }
}
```

Poznámka: Při vytváření indexu se mappings nemusí specifikovat, body může být prázdné. Anebo může obsahovat settings, kde se specifikuje počet shardů a replik:

```
{
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 2
  }
}
```




Tady budou 3 shardy a každý z nich bude mít 2 repliky

Template

- Index může být také vytvořený na základě šablony (template):
 - <https://www.elastic.co/guide/en/elasticsearch/reference/current/index-templates.html>

Elasticsearch indexy

- Tímto způsobem lze lehce získat seznam všech indexů:
 - GET http://localhost:9200/_cat/indices
 - GET http://localhost:9200/_cat/indices?v  verbose
 - Plus názvy sloupců
- Nastavení indexu (vrací mimo jiné počet shardů a replik):
 - GET http://localhost:9200/movies/_settings
- Shardy:
 - http://localhost:9200/_cat/shards

Přidání záznamu (insert)

POST http://localhost:9200/movies/_doc

```
{  
  "name" : "Batman"  
}
```

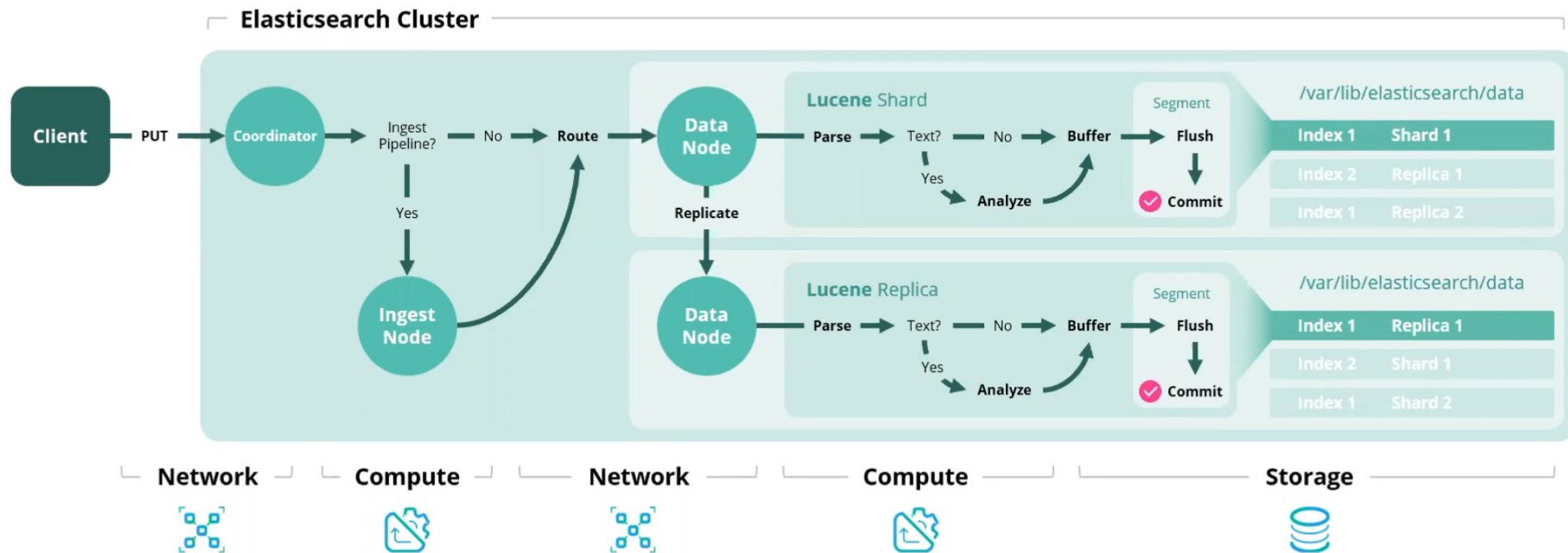
Poznámka: V odpovědi serveru se nachází atribut `_id`.
Dřív to bývalo číslo, nyní to je například: "mo71NngB_-oJ5tVYwmi0"

Nyní funguje: http://localhost:9200/movies/_doc/{HODNOTA_ID}

Poznámka: Také je k dispozici Bulk API (pomocí něj je možné dělat INSERT, UPDATE i DELETE:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html>

Insert operation



Změna záznamu (update)

- Každý dokument má atribut `_version`. Jakmile se provede update stávajícího záznamu, pak se vytvoří nový dokument s incrementovanou hodnotou `_version` a starý dokument je označen ke smazání. Elasticsearch ho někdy později automaticky smaže (je to podobný princip jako Gargage Collection). Záznamy jsou immutable.

```
PUT http://localhost:9200/movies/_doc/mo71NngB_-oJ5tVYwmi0/_update
```

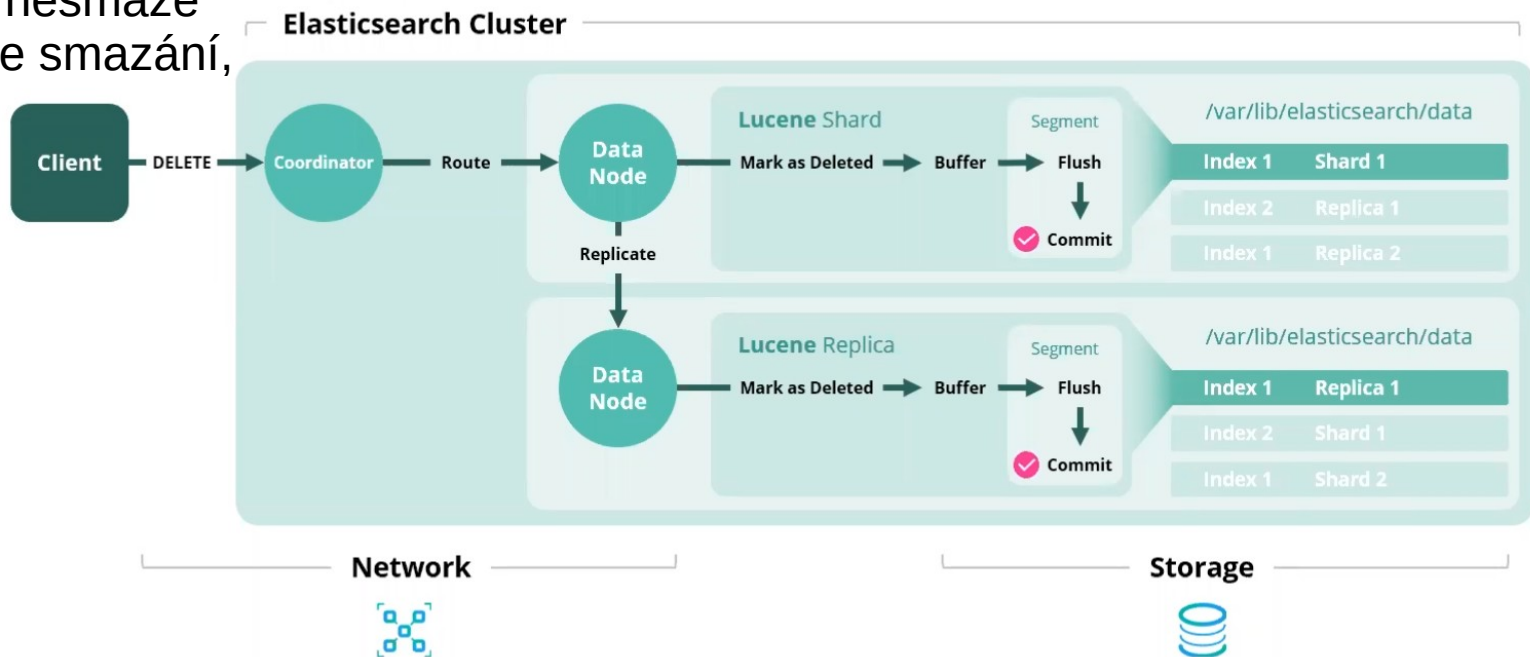
```
{
  "doc" : {
    "name" : "Batman 2"
  }
}
```

Smazání záznamu (delete)

- Smazání záznamu je triviální:

DELETE http://localhost:9200/movies/_doc/mo71NngB_-oJ5tVYwmi0

Poznámka: ES záznam nesmaže ihned, ale označího ho ke smazání, tudíž se místo na disku neuvolní ihned, ale až někdy později.



Vyhledání záznamu (search)

GET http://localhost:9200/movies/_search

```
{
  "query": {
    "match": {
      "name": "Batman"
    }
  }
}
```

Nebo zjednodušeně (“URI Search”) tímto způsobem:

GET http://localhost:9200/movies/_search?q=name:Batman

Nevýhoda: Před posláním na server se musí provést URL encoding (browser to provede za nás, ale jindy na to nesmíme zapomenout), takže je zapotřebí poslat toto:

http://localhost:9200/movies/_search?q=name%3ABatman

Hodí se to zejména pro jednoduché experimentování

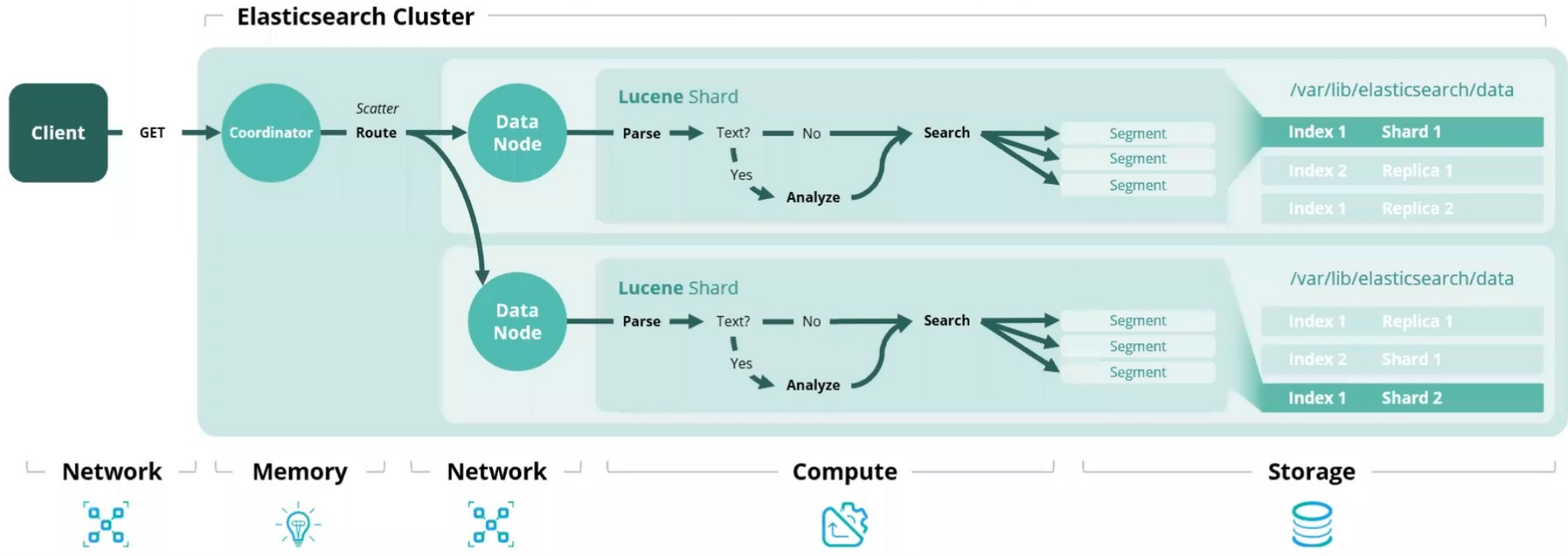
Search

- Vrátí všechny záznamy v indexu:

```
GET movies/_search
```

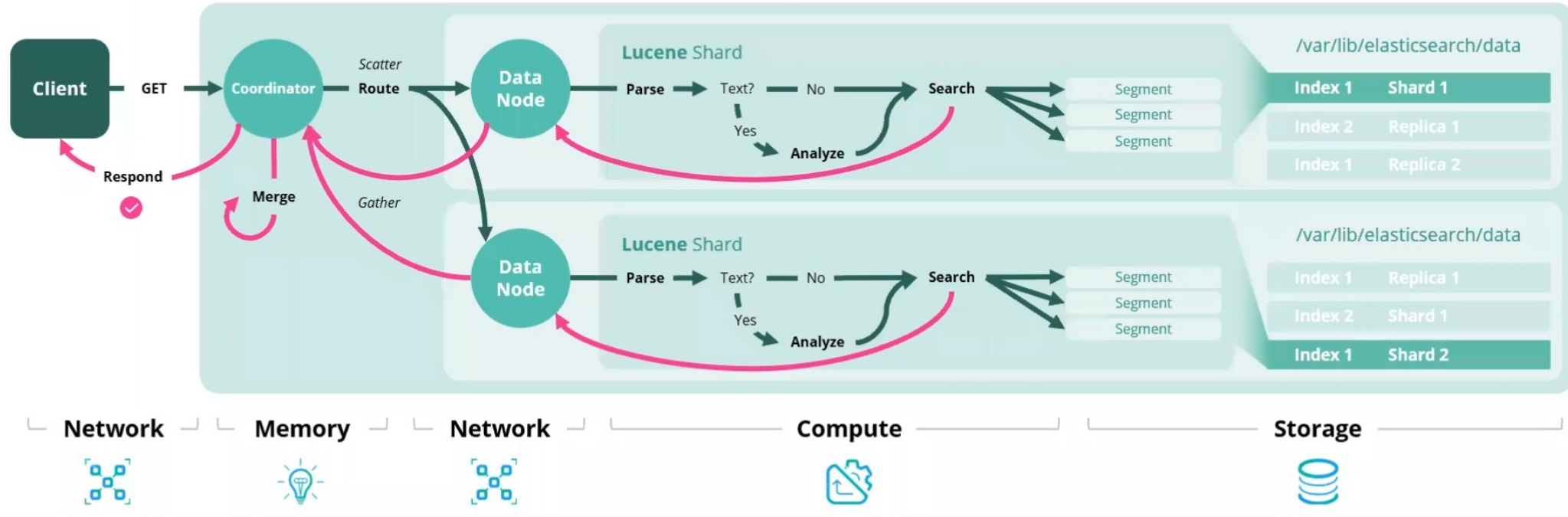
```
{  
  "query": {  
    "match_all": { }  
  }  
}
```

Search I.



Search II.

Elasticsearch Cluster



Query Benchmarking: Rally

- Rychlost vyhledávání závisí na mnoha proměnných. Pokud byste potřebovali optimalizovat rychlost vyhledávání, tak se vám bude hodit Rally:
 - <https://esrally.readthedocs.io/en/stable/>

Elasticsearch & storage effectivity

- Záznamy v ES jsou ve formátu JSON. Ten je méně “ukecaný” než například XML, ale více “ukecaný” než CSV formát, jehož varianty se používají při logování.
- Z toho je zřejmé, že například log data budou na disku v ES zabírat více místa než v raw podobě. Jaká bude tato “neefektivita” záleží na povaze dat.
- Při indexování dat jsou zejména texty indexované dvakrát: jednou samotná hodnota a podruhé pro full-text search.
- Na druhou stranu ES provádí kompresi dat, jejíž efektivita záleží na tom, co reálně v datech je, ale obvykle se tím ušetří 20 – 30%.
- Data jsou navíc replikované v tzv. replica shardech, kde se nachází kompletní kopie primárního shardu (pro zvýšení fault-tolerance a zrychlení vyhledávání v datech).

Elasticsearch SQL

- Od Elasticsearch 6.3 je možné dělat dotazy pomocí SQL:
 - <https://www.elastic.co/guide/en/elasticsearch/reference/current/sql-getting-started.html>

Docker & Filebeat & Elasticsearch

- Jak rozchodit Docker & Filebeat & Elasticsearch:
 - <https://medium.com/@bcoste/powerful-logging-with-docker-filebeat-and-elasticsearch-8ad021aecd87>
 - <https://www.sarulabs.com/post/5/2019-08-12/sending-docker-logs-to-elasticsearch-and-kibana-with-filebeat.html>

Konfigurace lifecycle management I.

- http://localhost:5601/app/kibana#/management/elasticsearch/index_lifecycle_management/policies
 - Nastavení rollover indexů, mazání starých indexů
 - Nastavení v Kibana
- Nebo se dá použít Curator (ale ILM je jednodušší):
 - <https://www.elastic.co/guide/en/elasticsearch/client/curator/current/index.html>

Poznámka: Fluentd elasticsearch plugin v současnosti obsahuje nativní podporu pro ILM:
https://github.com/ukenu/fluent-plugin-elasticsearch#enable_ilm

Konfigurace lifecycle management II.

- Low-level způsob konfigurace ILM je následujícím způsobem:

```
PUT _ilm/policy/datastream_policy
{
  "policy": {
    "phases": {
      "hot": {
        "actions": {
          "rollover": {
            "max_size": "50GB",
            "max_age": "30d"
          }
        }
      },
      "delete": {
        "min_age": "90d",
        "actions": {
          "delete": {}
        }
      }
    }
  }
}
```

```
PUT _template/datastream_template
{
  "index_patterns": ["datastream-*"],
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1,
    "index.lifecycle.name": "datastream_policy",
    "index.lifecycle.rollover_alias": "datastream"
  }
}
```

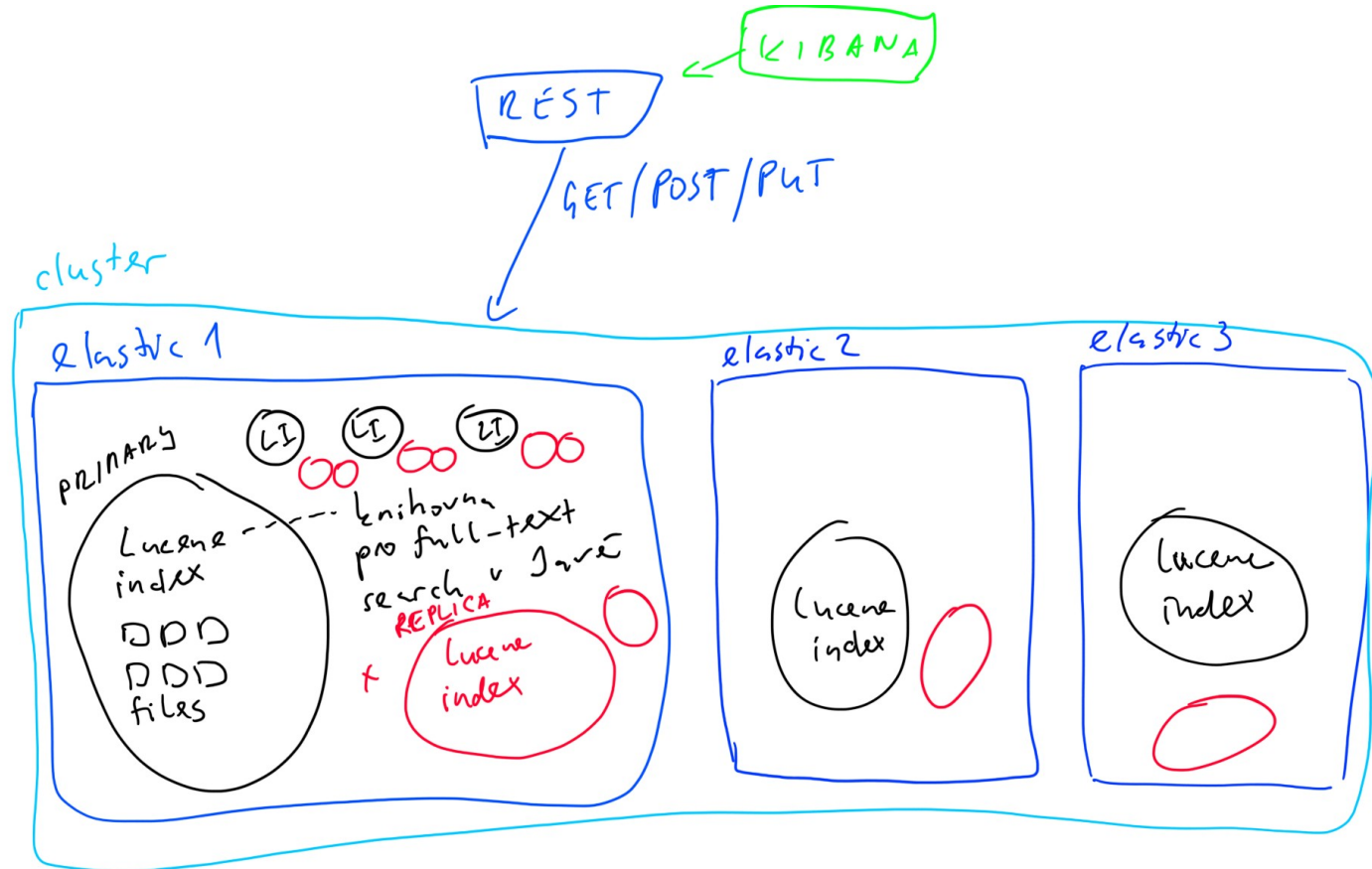
Index Alias

- V Elasticsearch je možné mít nejenom samotné indexy, ale také alias indexů:
 - <https://www.elastic.co/guide/en/elasticsearch/reference/6.8/indices-aliases.html>

ES Best Practices & Resources

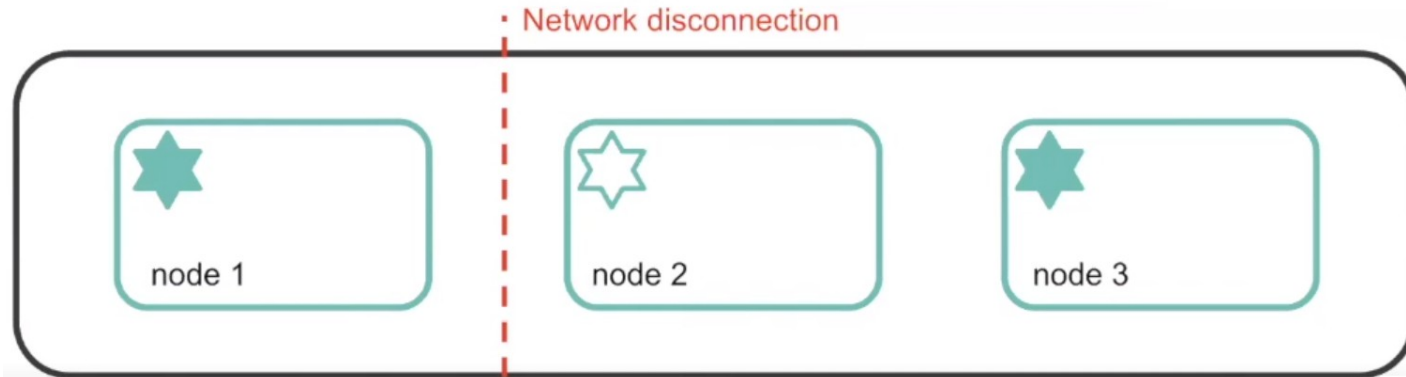
- Storage:
 - Best practice je používat maximálně RAID 0, ES provádí replikaci dat na uzly, tudíž je vyšší RAID zbytečná režie.
 - Nepoužívat NAS (SMB apod.) kvůli vysoké latenci
- Memory:
 - Je doporučeno používat maximálně 64GB RAM, přičemž z toho 32GB alokovat pro JVM heap (zbytek se použije pro cache):
 - https://www.elastic.co/guide/en/elasticsearch/guide/current/hardware.html#_memory
- CPU:
 - Jak hodně CPU prostředků potřebujeme záleží na tom, co s ES děláme
- Network:
 - Pokud není latence mezi jednotlivými uzly moc vysoká, tak s tím není problém.

Elasticsearch cluster



ES Cluster Best Practices

- Split Brain problem:
 - V jedné chvíli může být pouze jeden master node
 - Best practice je mít 3 master-eligible nody (master-eligible node je node, který může být master) & nastavené `minimum_master_nodes = 2`
 - Jinak může nastat Split Brain problem: Node 1 může mít problém se sítí a může se nastavit jako master. Jakmile problém se sítí zmizí, tak jsou v clusteru 2 master nody a to je “Split Brain” problem:



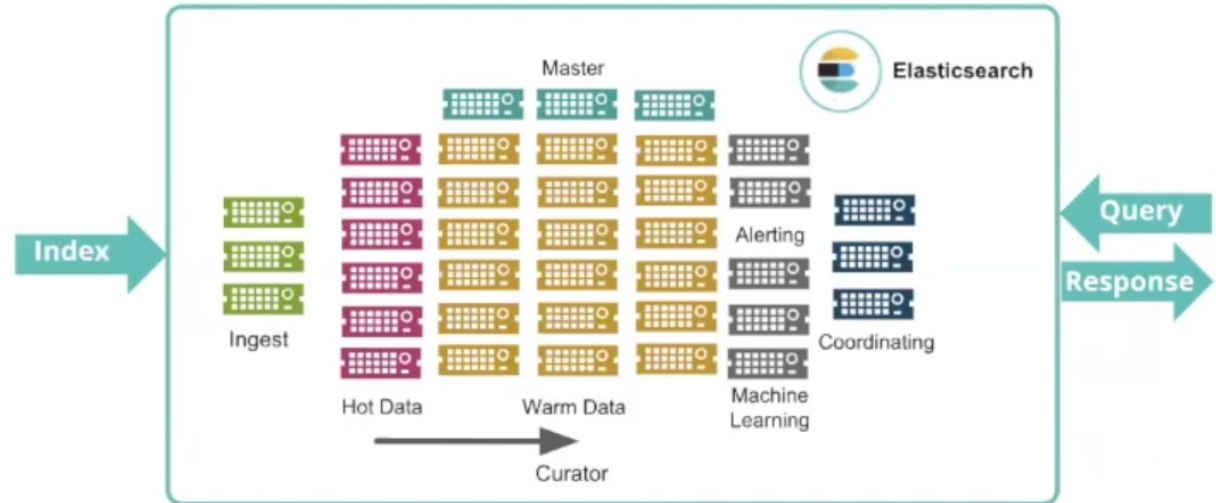
Hot & Warm nodes

- V případě, že bychom měli velký cluster, pak má význam u některých uzlů nastavit, že jsou “Hot” nebo “Warm”, přičemž “Hot” uzly budou na SSD (drahé, ale rychlé) a “Warm” budou na HDD (levné, ale pomalé).
- Nejnovější data (například logy za poslední den) se ukládají do “Hot” uzlů a starší data se z těch “Hot” uzlů pomocí nastavení ILM / Curator přesouvají do “Warm” uzlů.

Poznámka: Jestli je uzel hot nebo warm se nastavuje v elasticsearch.yml:

```
bin/elasticsearch -Enode.attr.data=hot  
bin/elasticsearch -Enode.attr.data=warm  
bin/elasticsearch -Enode.attr.data=cold
```

<https://www.elastic.co/blog/implementing-hot-warm-cold-in-elasticsearch-with-index-lifecycle-management>



Typy uzlů

<https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html>

Role	Description	Resources			
		Storage	Memory	Compute	Network
Data	Indexes, stores, and searches data	Extreme	High	High	Medium
Master	Manages cluster state	Low	Low	Low	Low
Ingest	Transforms inbound data	Low	Medium	High	Medium
Machine Learning	Processes machine learning models	Low	Extreme	Extreme	Medium
Coordinator	Delegates requests and merges search results	Low	Medium	Medium	Medium

Poznámky:

- Ingest uzly (dělají transformaci dokumentu před jeho indexací) se dají nahradit Logstash
- Všechny uzly mají automaticky roli “Coordinator”
- Machine Learning je placená nadstavba
- Nejdůležitější jsou tedy Data & Master uzly

Clustered Elasticsearch sharding

- <https://www.objectrocket.com/blog/elasticsearch/clustered-elasticsearch-best-practices/>

Nastavení filebeat & stacktrace (multiline)

- <https://www.elastic.co/guide/en/beats/filebeat/master/multiline-examples.html>

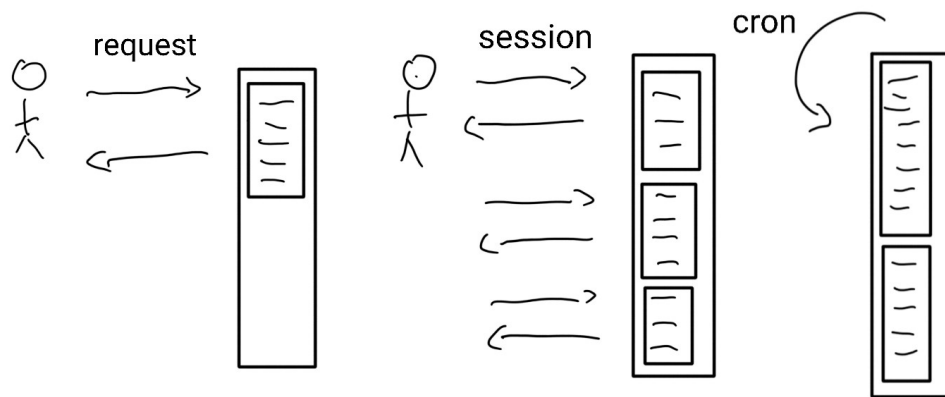
Tip: Kibana LIKE QUERY

- <https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-wildcard-query.html>

```
GET /_search
{
  "query": {
    "wildcard": {
      "user.id": {
        "value": "ki*y",
        "boost": 1.0,
        "rewrite": "constant_score"
      }
    }
  }
}
```

Tip: MDC (Mapped Diagnostic Context)

- Při logování více microservice nebo u reaktivních microservice je užitečné do logu přidat MDC (Mapped Diagnostic Context):
 - <https://dzone.com/articles/mdc-better-way-of-logging-1>
 - <https://spring.io/projects/spring-cloud-sleuth>
- Jinak pro analýzu logů obecně (včetně nebo i bez MDC) je velice užitečný například ELK stack (ElasticSearch & Kibana).
 - Nebo Graylog:
 - <https://www.graylog.org/>
 - Nebo Loki:
 - <https://grafana.com/oss/loki/>



Tip: Exclude Docker Container

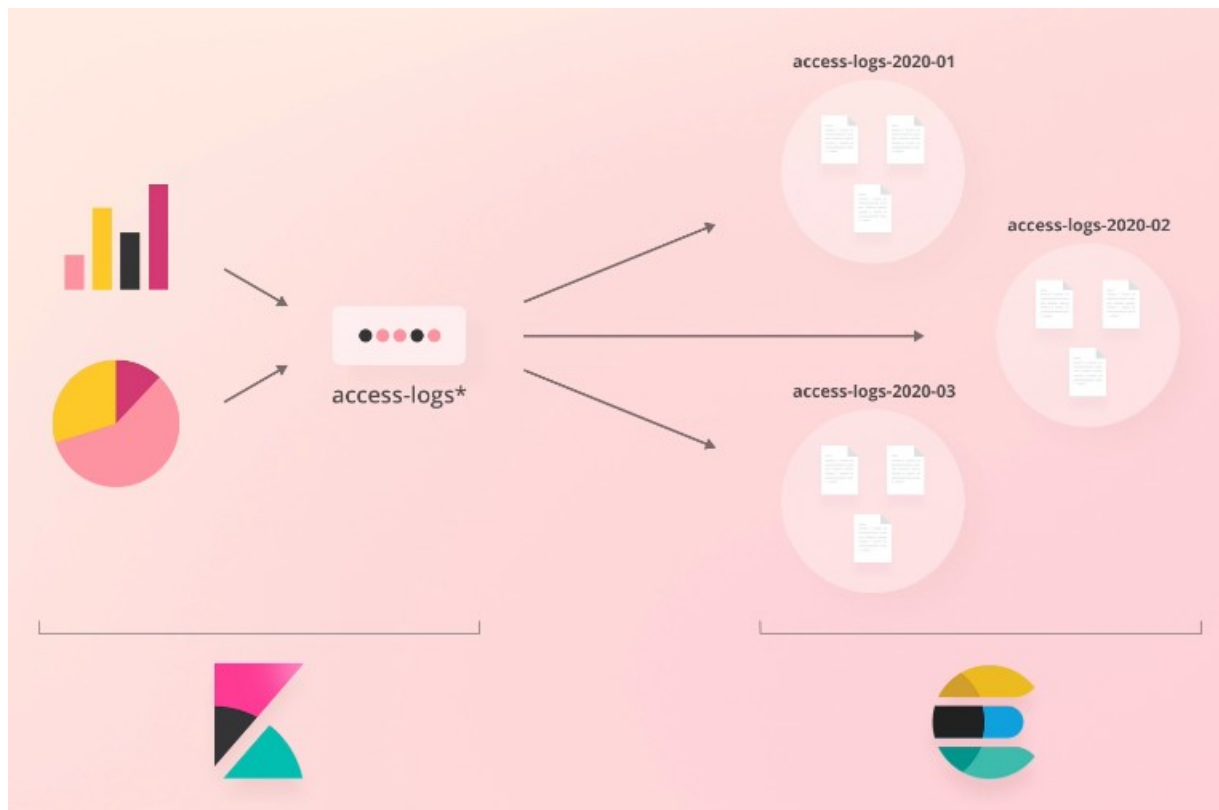
- Když se čtou logy z Docker JSONu:
 - <https://www.sarulabs.com/post/5/2019-08-12/sending-docker-logs-to-elasticsearch-and-kibana-with-filebeat.html>
- Pak je možné provést exclude Docker kontejneru ve Filebeat tímto způsobem:
 - https://www.reddit.com/r/elastic/comments/9jt7a8/filebeat_docker_logs_how_can_you_exclude/

Tip: Low disk watermark exceeded on ...

- Disk musí být zaplněn méně než z 85%, jinak Elasticsearch nebude vytvářet shardy:
 - <https://www.elastic.co/guide/en/elasticsearch/reference/7.8/modules-cluster.html#disk-based-shard-allocation>
 - <https://stackoverflow.com/questions/33369955/low-disk-watermark-exceeded-on>

Kibana Index Patterns

- Indexy jsou často strukturovány tak, že je jeden index pro data za jeden den. K vyhledávání ve více takových indexech se v Kibana používají index patterns, které mají podporu pro wildcard (hvězdička: *):



KQL (Kibana Query Language)

```
url.path:"/brands" and (http.response.status_code:404 or http.response.status_code:500)
```



```
{
  "query": {
    "bool": {
      "filter": [
        {
          "match_phrase": {
            "url.path": "/brands"
          }
        },
        {
          "bool": {
            "should": [
              {
                "match": {
                  "http.response.status_code": 404
                }
              },
              {
                "match": {
                  "http.response.status_code": 500
                }
              }
            ]
          }
        }
      ]
    }
  }
}
```

KQL: text search

products fischer



```
{
  "query": {
    "bool": {
      "filter": [
        {
          "multi_match": {
            "type": "best_fields",
            "query": "products fischer",
            "lenient": true
          }
        }
      ]
    }
  }
}
```

multi_match je nadstavba nad “match query” a umožňuje vyhledávat ve více atributech (fields) dokumentu

Hledá se ve všech atributech dokumentu:
<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html#type-best-fields>

Nehledá se přesný výraz, ale hledají se tyto dvě slova. Mohou být klidně i v opačném pořadí. (To je vlastnost match query). A ve výsledném dokumentu může být i jenom jedno z těchto slov.

Umožňuje hledat řetězce v numeric attributech (není tak striktní s typy):
<https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-query.html#match-field-params>

KQL: přesný výraz

"products fischer"



```
{
  "query": {
    "bool": {
      "filter": [
        {
          "multi_match": {
            "type": "phrase",
            "query": "products fischer",
            "lenient": true
          }
        }
      ]
    }
  }
}
```

Když se hledaný text dá do uvozovek, pak se hledá přesný výraz. Dokument musí obsahovat text "products fischer".

KQL: hledání v konkrétním atributu

```
<field name><operator><value>
```

Šablona výrazu

```
http.response.status_code:404
```

Příklad

```
{
  "query": {
    "match": {
      "http.response.status_code": 404
    }
  }
}
```

KQL: hledání v konkrétním atributu

```
user_agent.original.text:mozilla
```



```
{
  "query": {
    "match": {
      "user_agent.original.text": "mozilla"
    }
  }
}
```

Volný text

```
user_agent.original.text:"mozilla iphone"
```



```
{
  "query": {
    "match_phrase": {
      "user_agent.original.text": "mozilla iphone"
    }
  }
}
```

Přesný výraz

Range operator

>

>=

<

<=

`http.response.status_code < 300`



```
{
  "query": {
    "range": {
      "http.response.status_code": {
        "lt": 300
      }
    }
  }
}
```

`hour_of_day >= 18`



```
{
  "query": {
    "range": {
      "hour_of_day": {
        "gte": 18
      }
    }
  }
}
```

Boolean operátor

```
http.response.status_code >= 200 and http.response.status_code < 300
```



```
{
  "query": {
    "bool": {
      "filter": [
        {
          "range": {
            "http.response.status_code": {
              "gte": 200
            }
          }
        },
        {
          "range": {
            "http.response.status_code": {
              "lt": 300
            }
          }
        }
      ]
    }
  }
}
```


Boolean operátor

`http.response.status_code:404 or http.response.status_code:500`



```
{
  "query": {
    "bool": {
      "should": [
        {
          "match": {
            "http.response.status_code": 404
          }
        },
        {
          "match": {
            "http.response.status_code": 500
          }
        }
      ]
    }
  }
}
```

NEBO:

`http.response.status_code:(404 or 500)`



```
{
  "query": {
    "bool": {
      "should": [
        {
          "match": {
            "http.response.status_code": 404
          }
        },
        {
          "match": {
            "http.response.status_code": 500
          }
        }
      ]
    }
  }
}
```

Závorky

“and” operátor má přednost před “or”. Klasickým způsobem můžeme použít závorky:

```
url.path: "/brands" and http.response.status_code:404 or http.response.status_code:500
```

```
url.path: "/brands" and (http.response.status_code:404 or http.response.status_code:500)
```

Boolean operator: negace

not http.response.status_code:200



```
{
  "query": {
    "bool": {
      "must_not": [
        {
          "match": {
            "http.response.status_code": 200
          }
        }
      ]
    }
  }
}
```

http.response.status_code:200 and not (url.path:"/brands" or url.path:"/brands/fischer")



```
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "http.response.status_code": 200
          }
        }
      ],
      "must_not": [
        {
          "match_phrase": {
            "url.path": "/brands"
          }
        },
        {
          "match_phrase": {
            "url.path": "/brands/fischer"
          }
        }
      ]
    }
  }
}
```

Existence atributu

`http.request.referrer:*`



```
{
  "query": {
    "exists": {
      "field": "http.request.referrer"
    }
  }
}
```

Tato query vrátí všechny dokumenty, které obsahují atribut (field): "http.request.referrer")

Hodnoty s hvězdičkou (wildcard)

`url.path:/brands*`



```
{
  "query": {
    "query_string": {
      "fields": [
        "url.path"
      ],
      "query": "\\ /brands*"
    }
  }
}
```

Najde všechny dokumenty, ve kterých field "url.path" začíná na "/brands"

Hledání ve více fieldech

`user_agent.original:mozilla or user_agent.original.text:mozilla`

NEBO:

`user_agent.original*:mozilla`



```
{
  "user_agent": {
    "properties": {
      "original": {
        "type": "keyword",
        "fields": {
          "text": {
            "type": "text",
            "norms": false
          }
        }
      }
    }
  }
}
```

Agregace I.

- ES má 3 typy agregací:
 - **Metric:** Výsledkem jsou metriky, jako sum, min, max, average atd. z hodnot ve fieldech
 - **Bucket:** Seskupuje dokumenty do košíků (buckets, bins) na základě hodnot ve fieldech
 - **Pipeline:** Provádí operace na výsledcích agregací

Příklad:

GET kibana_sample_data_logs/_search

```
{
  "aggs": {
    "preneseno_dat": {
      "sum": {
        "field": "bytes"
      }
    }
  }
}
```



Ve výsledku bude (dole):

```
"aggregations" : {
  "preneseno_dat" : {
    "value" : 7.9725689E7
  }
}
```

Agregace II.

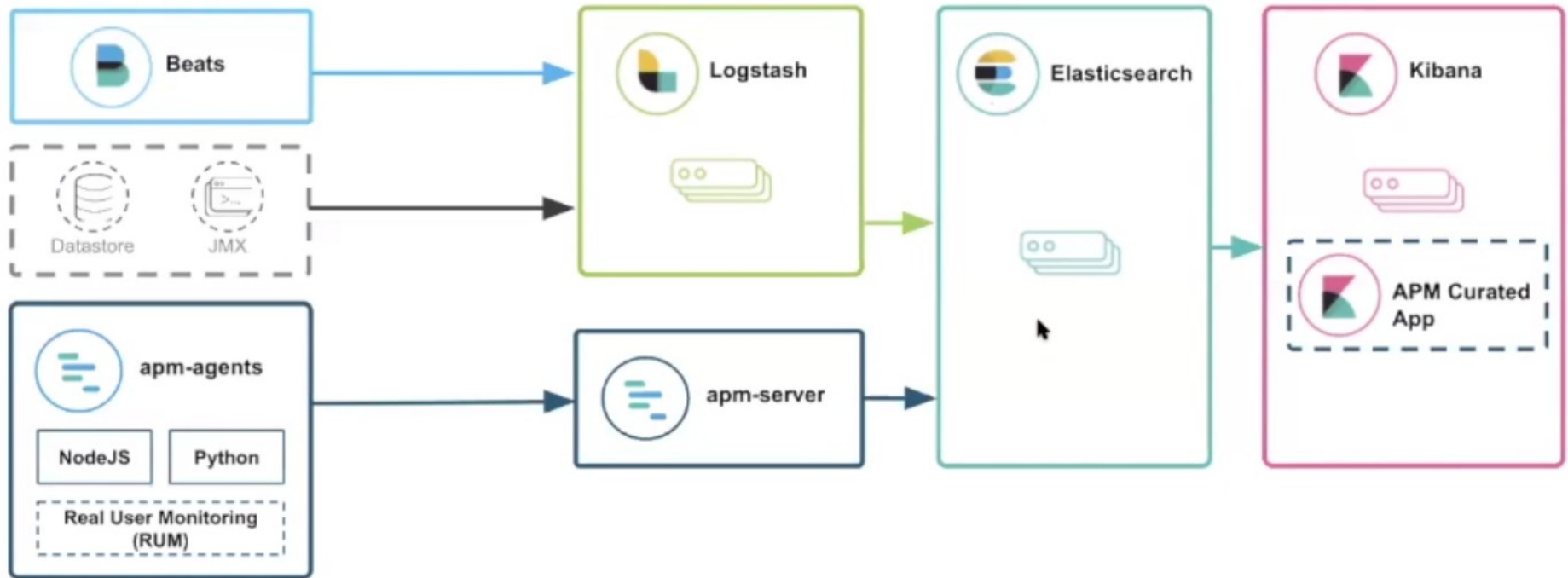
- V jednom požadavku může být více agregací:

```
GET kibana_sample_data_logs/_search
```

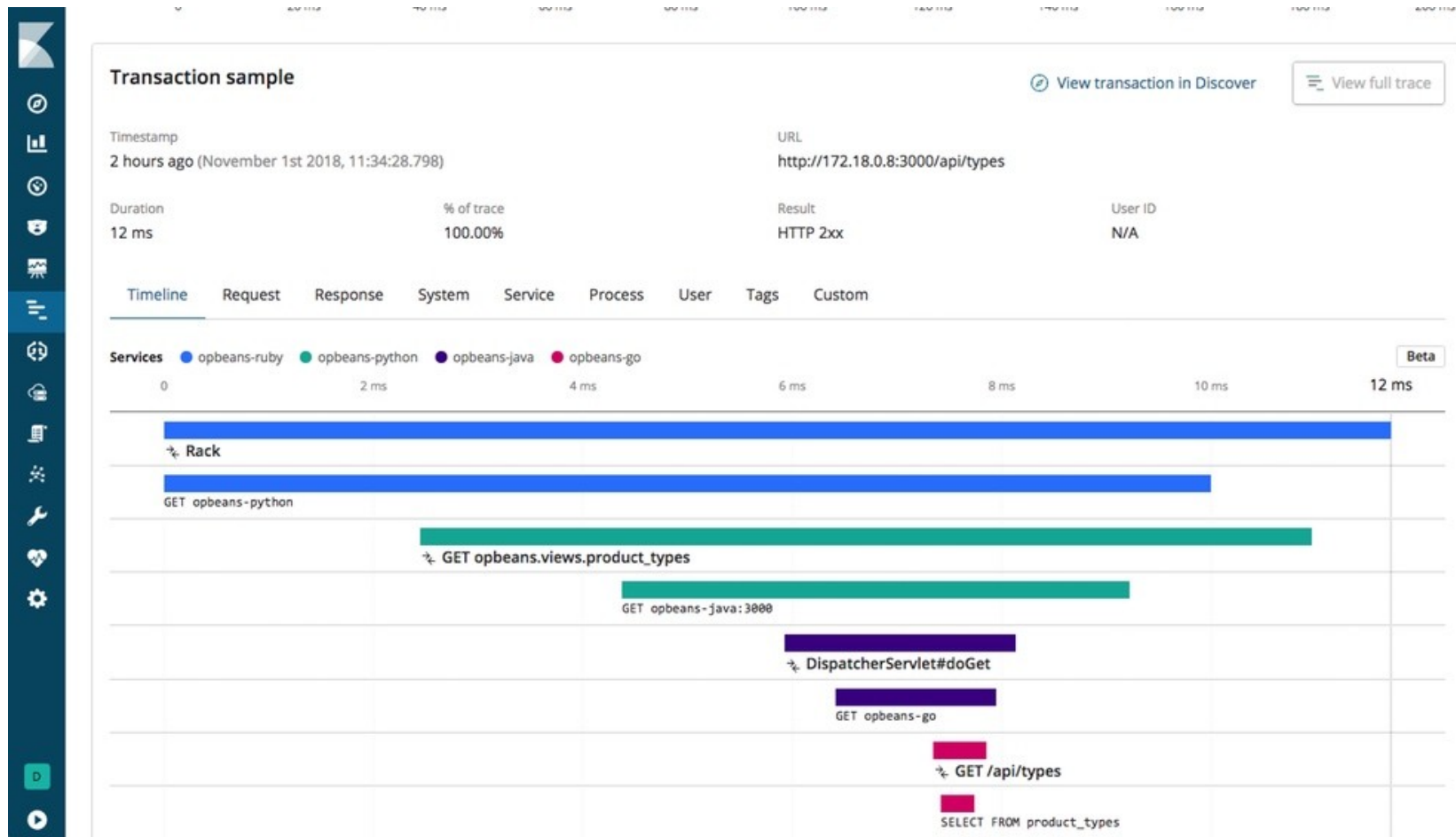
```
{  
  "aggs": {  
    "preneseno_dat": {  
      "sum": {  
        "field": "bytes"  
      }  
    },  
    "pocet_zaznamu" : {  
      "value_count": {  
        "field": "bytes"  
      }  
    }  
  }  
}
```


Kibana APM I.

- Kibana má APM:

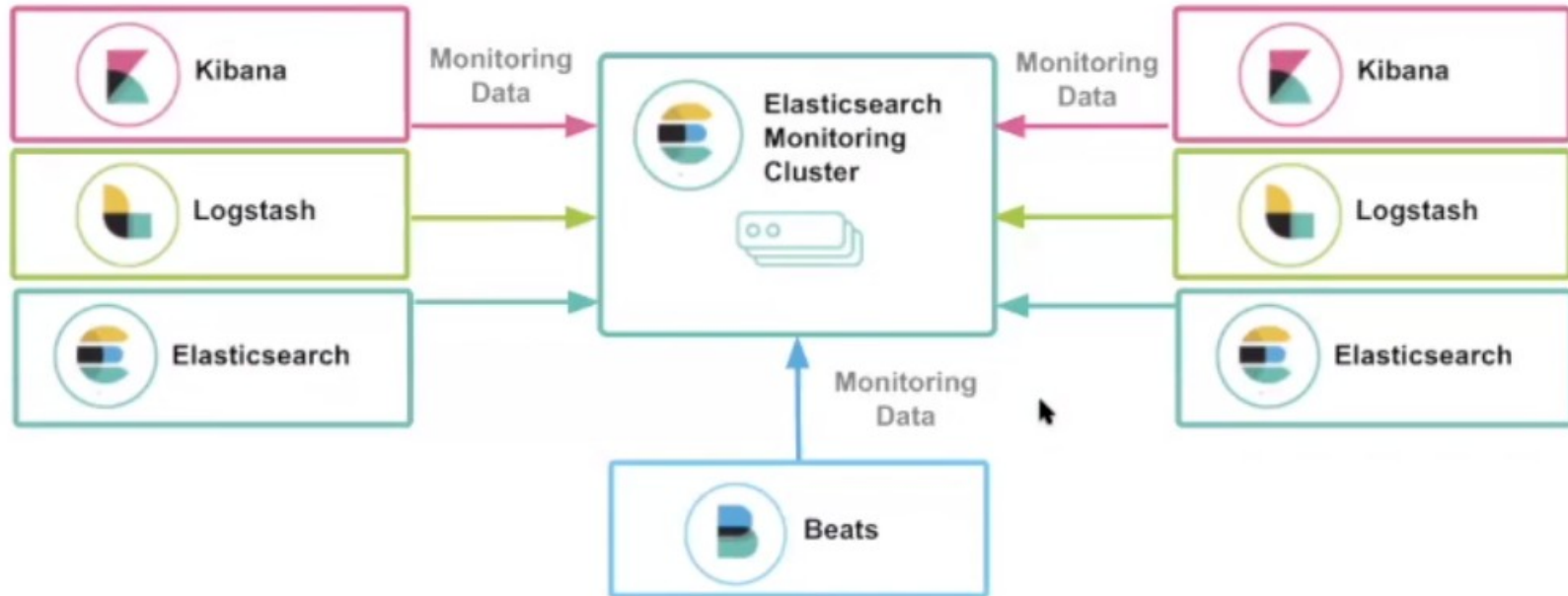


Kibana APM II.



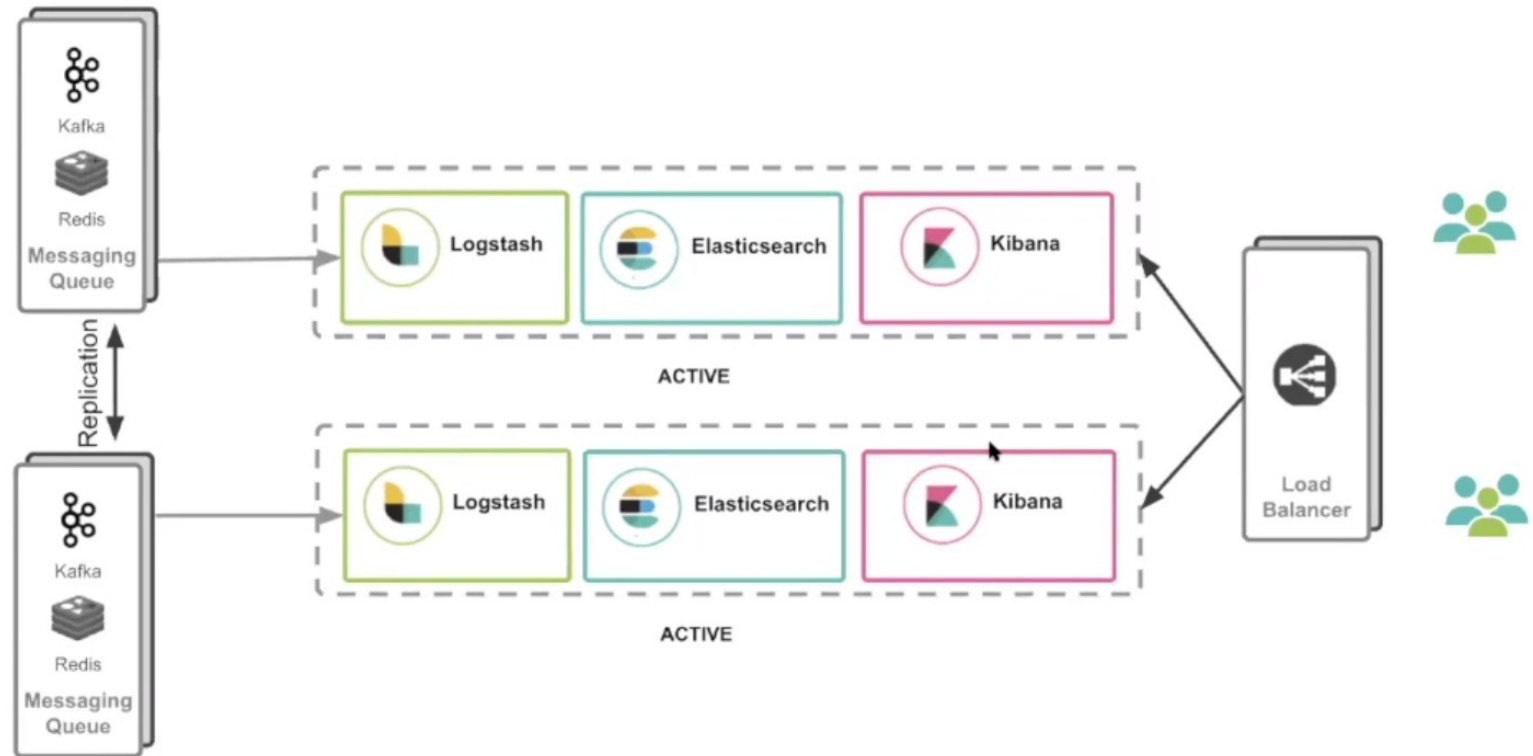
Beats (Monitoring)

- Pomocí Beats je možné monitorovat samotný Elasticsearch cluster, Kibanu, nebo cokoli jiného. Je doporučeno mít samostatný monitorovací cluster, který je oddělený od hlavního ES clusteru tak, aby když hlavní ES cluster vypadne, tak aby stále fungoval ten monitorovací:

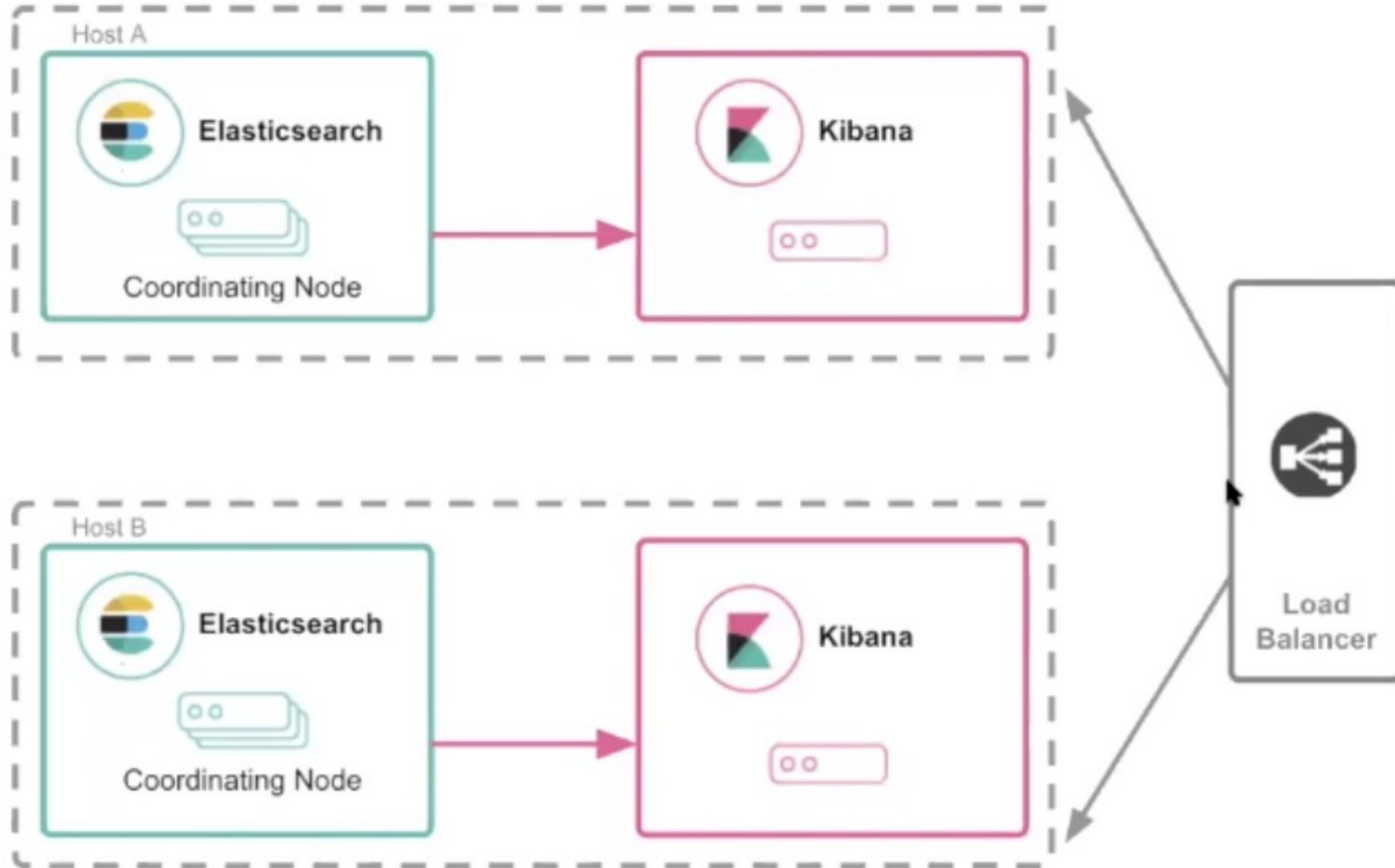


ES & větší množství datových center

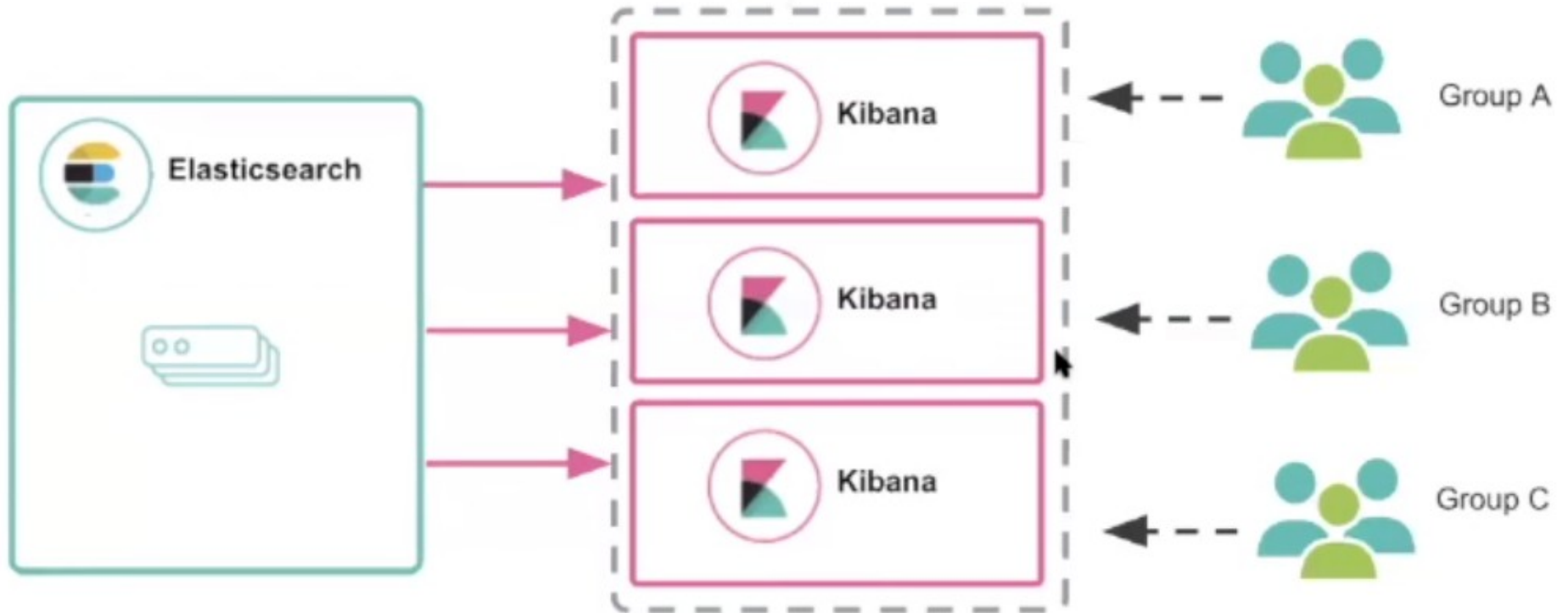
- V případě většího množství datových center (typicky také oddělených geograficky) je best practice, aby v každém datovém centru byl separátní ES cluster:



Kibana HA I.



Kibana HA II.



Security

- Od 6.8.0 a 7.1.0 je základní securita v Elasticsearch zdarma:
 - <https://www.elastic.co/blog/security-for-elasticsearch-is-now-free>
- Zdarma je:
 - TLS (HTTPS)
 - Vytváření a management uživatelů
 - Role-based access control pro přístup ke cluster API a indexům

Co je v ELK free / paid:

- <https://www.elastic.co/subscriptions>

Elasticsearch cheatsheet

- <https://elasticsearch-cheatsheet.jolicode.com/>