Operating Elasticsearch at Scale!

Vikram Ramakrishnan & Aishwarya Sankaravadivel

June 2019 @ SRECON Asia/Pacific

Agenda

- Introductions
- Search Platform @ PayPal
- Elasticsearch Primer
- Making Elasticsearch truly Elastic
- Managing Elasticsearch @ Scale
- Elasticsearch in action!
- Q & A

Introductions

Vikram Ramakrishnan

Senior Manager,
Technology Platforms & Experience

15 years of industry experience with specific focus in the areas of Large scale distributed systems, Enterprise Search & Big Data

vikramakrishnan@paypal.com
https://www.linkedin.com/in/vikramakrishnan

Aishwarya Sankaravadivel

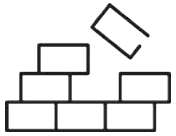
Senior Software Engineer
Technology Platforms & Experience

Passionate technologist with deep expertise in managing Large scale Elasticsearch deployments & Reactive systems

asankaravadivel@paypal.com
https://www.linkedin.com/in/aishwaryasankar/

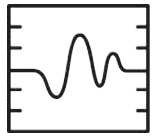
Search Platform @ PayPal

Fully Managed Search offering for PayPal Inc.



1500+ nodes

4+ PB of Data



20+ Billion Ingests / day

50+ Million Searches / day

Other Offerings

Custom Security Plugin

Custom Monitoring & Alerting

Elasticsearch Primer

Open Source

Released in 2010 and built on top of Apache Lucene

Distributed

Supports replication, sharding & routing

Schema Free

Supports JSON structured documents, detects types and makes them searchable.

Restful APIs

API driven and all actions can be performed via simple APIs using JSON over HTTP

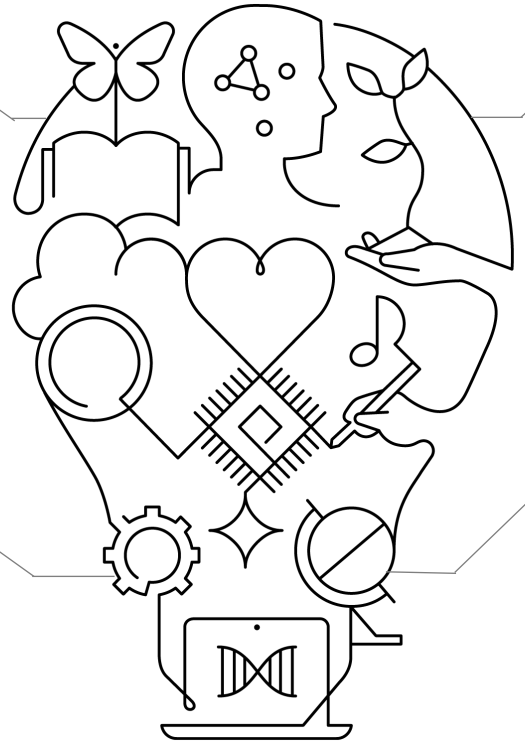
Elasticsearch Primer

(un)Structured Searches

Full text search, Faceted Navigation, Pagination

Fuzzy Searches

Autocomplete, ”Did you mean”, Suggestions.



Aggregations

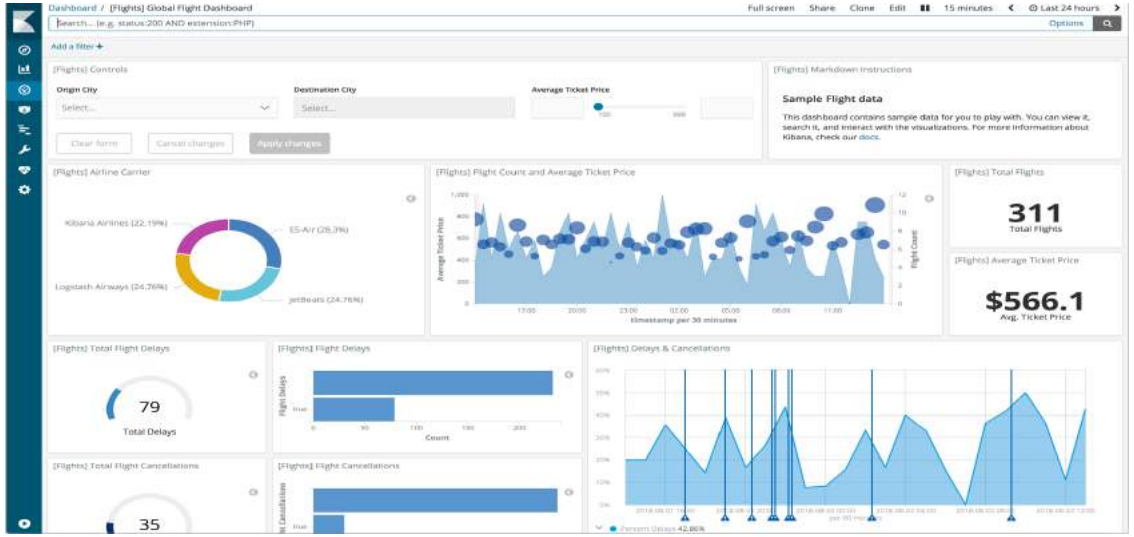
Bucketing, Metric, Pipelining

Powerful Visualizations

Tools like Kibana, Grafana to provide deep actionable insights

Elasticsearch Primer

What's in it for SREs?



Improved Observability

Realtime, Actionable Insights

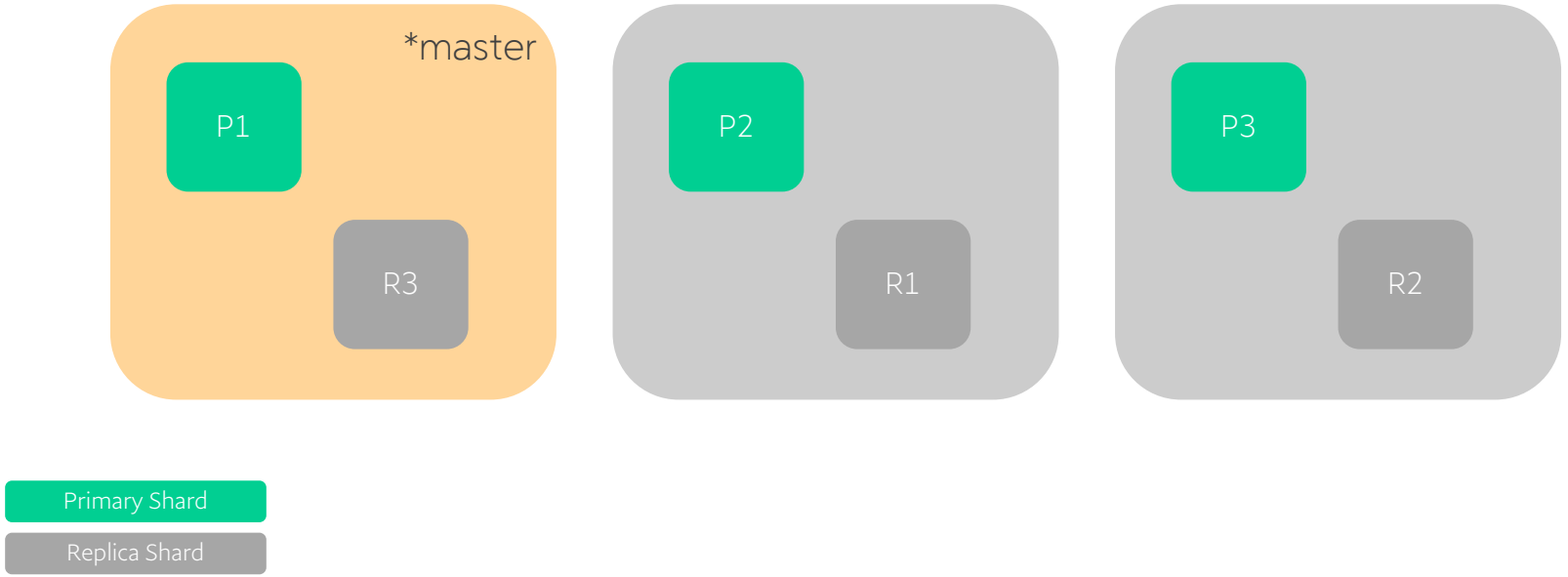
Elasticsearch Primer

Things to know before we dig deep...

- Cluster
- Node
- Document
- Index
- Field
- Mapping
- Shards
- Routing

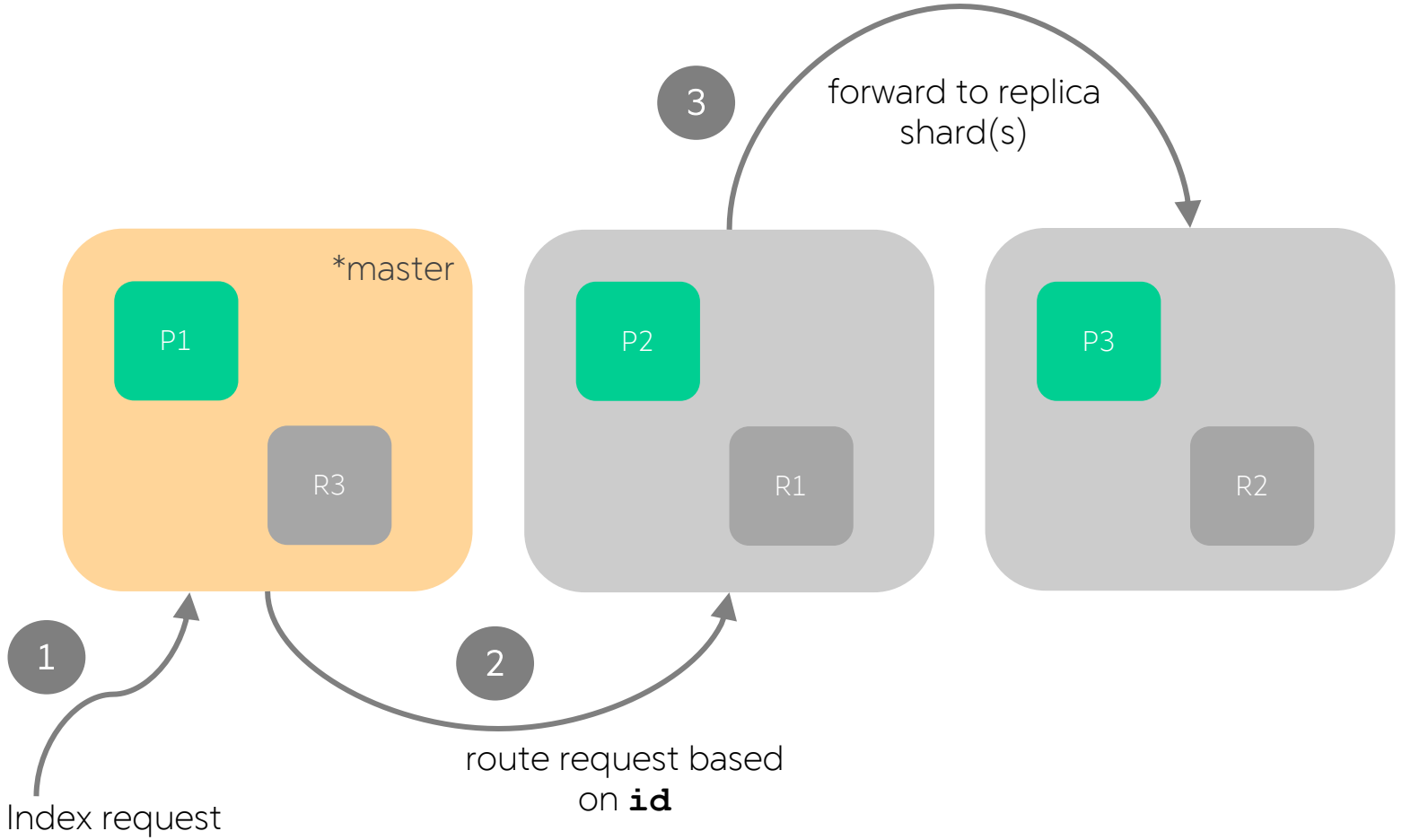
Elasticsearch Primer

Anatomy of an Elasticsearch cluster



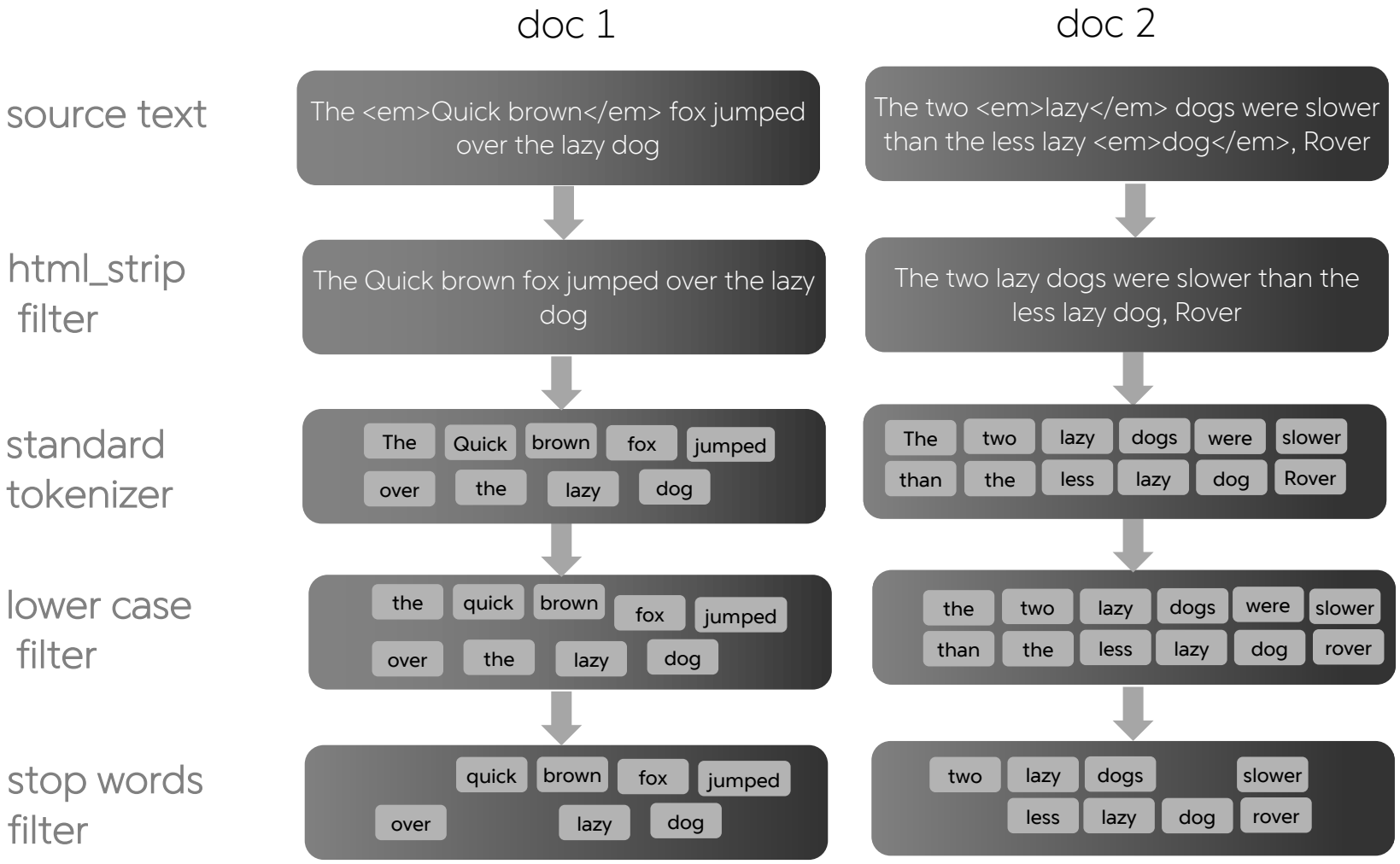
Elasticsearch Primer

Anatomy of an Indexing request



Elasticsearch Primer

Dissecting Indexing

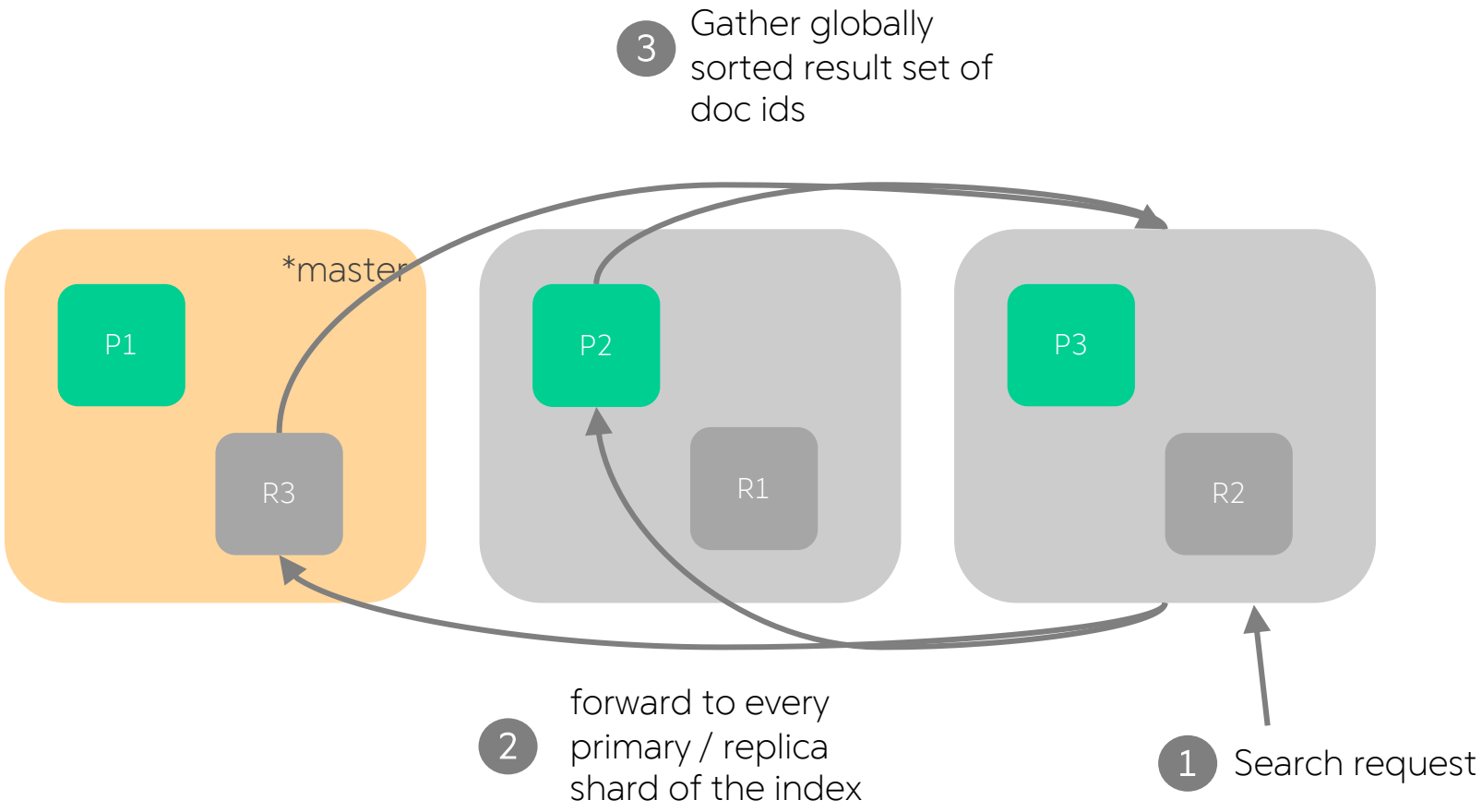


Inverted Index

Term	Document
quick	1
brown	1
fox	1
jumped	1
over	1
lazy	1, 2
dog	1, 2
two	2
dogs	2
slower	2
less	2
rover	2

Elasticsearch Primer

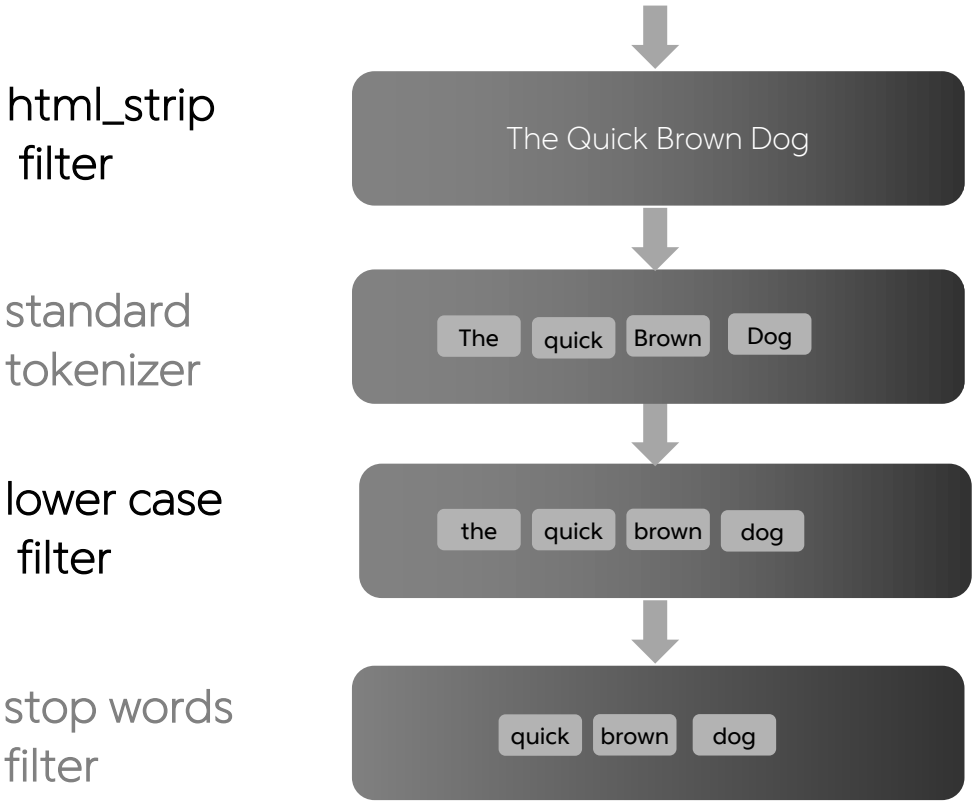
Anatomy of a search request.



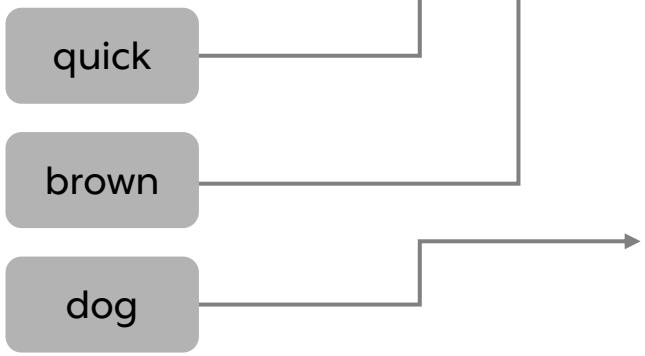
Elasticsearch Primer

Dissecting a search request.

Search for "*The quick Brown Dog*"



Inverted Index



Term	Document
quick	1
brown	1
fox	1
jumped	1
over	1
lazy	1, 2
dog	1, 2
two	2
dogs	2
slower	2
less	2
rover	2

Elasticsearch Primer

Installation

```
$ curl -L -O https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-{VERSION}.tar.gz

$ tar -xvf elasticsearch-{VERSION}.tar.gz

$ cd elasticsearch-{VERSION}/bin

$ ./elasticsearch
```

Elasticsearch Primer

Installation

```
[~]$ curl http://localhost:9200
{
  "name" : "LM-MAA-26500559_bloodseeker-masterdata",
  "cluster_name" : "bloodseeker",
  "cluster_uuid" : "gy1tSWy5RNCkAoIAxprpWA",
  "version" : {
    "number" : "6.2.4",
    "build_hash" : "ccec39f",
    "build_date" : "2018-04-12T20:37:28.497551Z",
    "build_snapshot" : false,
    "lucene_version" : "7.2.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Now, how do we

scale this for the

enterprise?

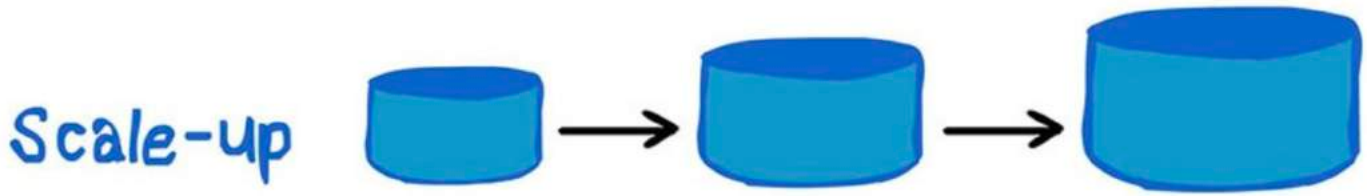

Scaling Elasticsearch!

Scaling Elasticsearch!

SCALABILITY?

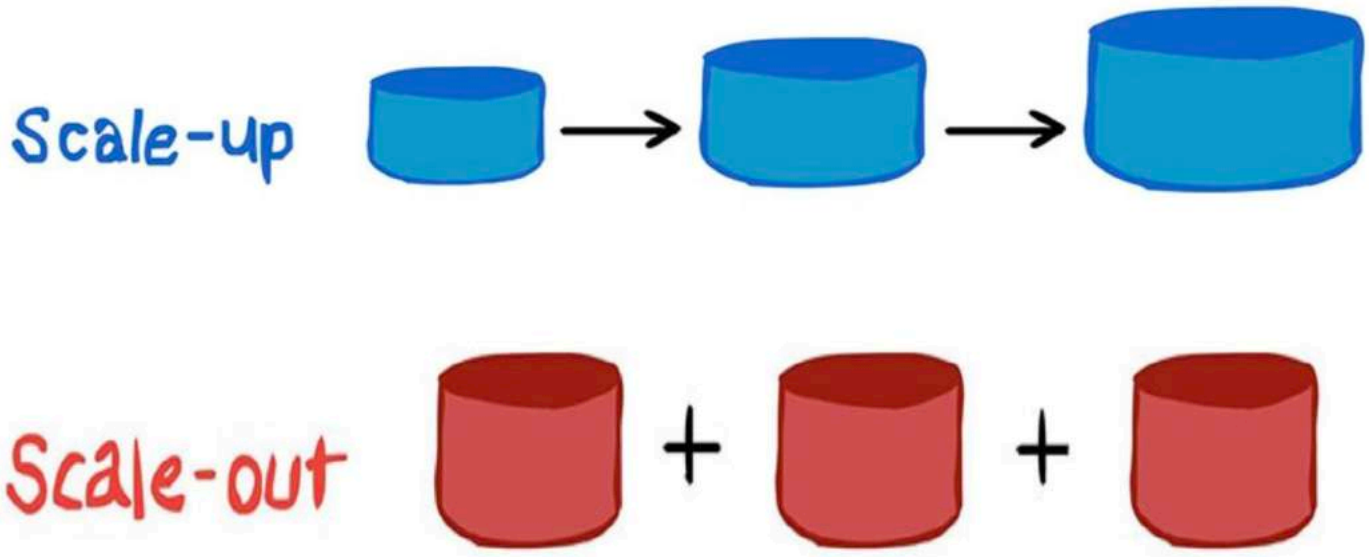
Scaling Elasticsearch!

- Scale Up

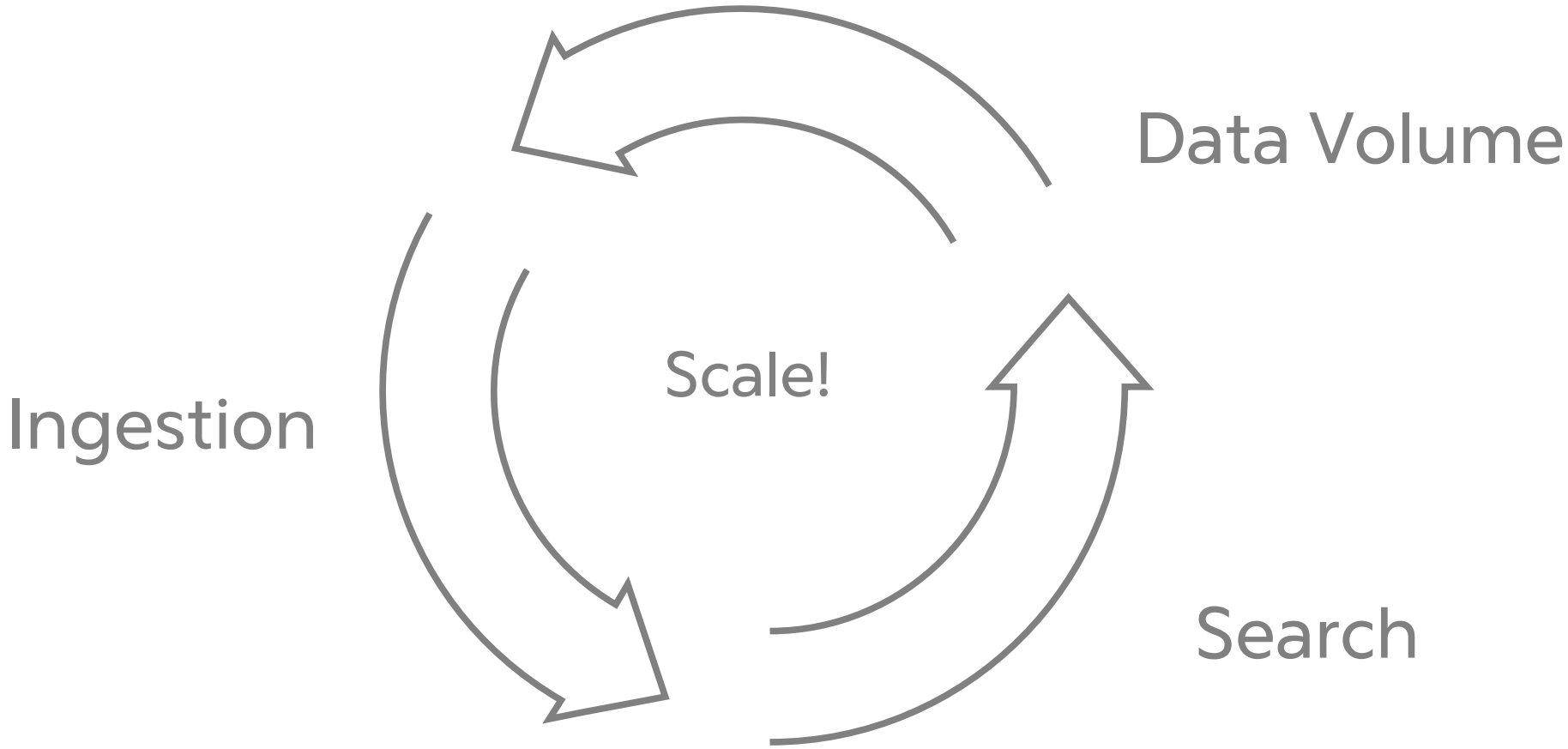


Scaling Elasticsearch!

- Scale Up
- Scale Out



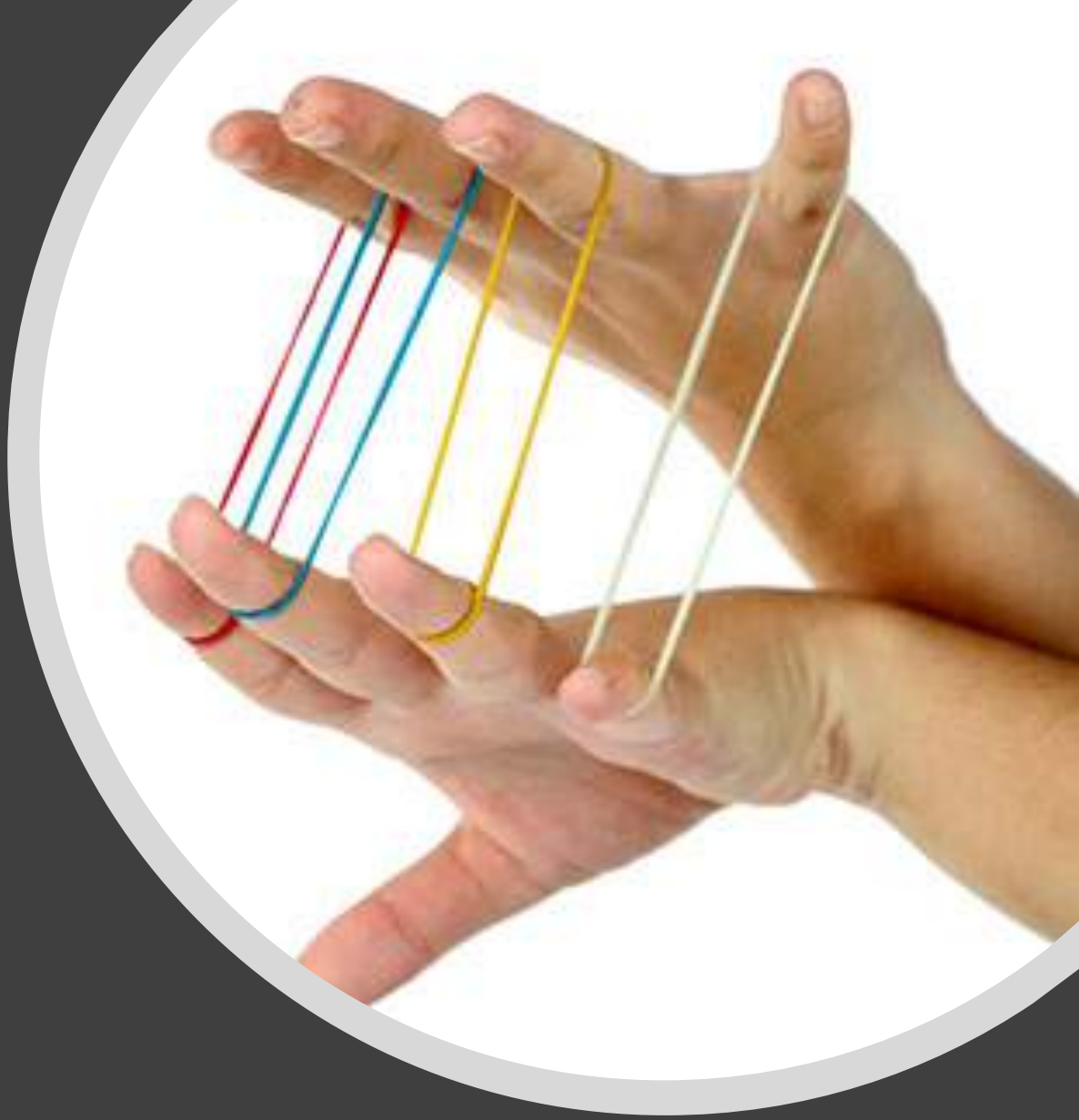
Scaling Elasticsearch!



Scaling Elasticsearch!

- Memory / CPU / IO / Network Issues
- Sub-optimal Queries
- Mapping Explosion
- Undersized / Oversized shards

Is Elasticsearch truly elastic?



Scaling Elasticsearch!

Problem#1 : Low Ingestion Throughput

Need to process ‘X’ million / minute additionally.

Learnings

- `_routing` for Bulk requests
- `index.refresh_interval`
- `number_of_shards, total_shards_per_node`
- `index.translog.durability:async`

Scaling Elasticsearch!

Problem #2 : Node instabilities due to Mapping Explosion and Sparse Fields

Scaling Elasticsearch!

Problem #2 : Node instabilities due to Mapping Explosion and Sparse Fields

- Mapping Explosion

	C1	C2	C3	C4	C5	.	.	.	.	.	.	.	.	.	.	.	.	.	C2000
R1																			
R2																			
R3																			

Scaling Elasticsearch!

Problem #2 : Node instabilities due to Mapping Explosion and Sparse Fields

- Mapping Explosion
- Sparse fields

	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9	C 10	C 11	C 12	C 13	C 14	C 15	C 16	C 17	C 18
R1	A 1			A 2		A 3	A 4			A 5		A 8	A 9		A 10			A 11
R2		A 1			A 2						A 3		A 4			A 5		
R3			A 1			A 2		A 3			A 4	A 5	A 6	A 7		A 9		

Scaling Elasticsearch!

Problem #2 : Node instabilities due to Mapping Explosion and Sparse Fields

Learnings:

- Keep a watch on mapping.
- Split indices.
- Date based indices works best for logs monitoring!

20% disk space

13% search latency

Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

Example Queries:

#1 Group By operation on a Id field

```
{
      "aggs": {
            "ids": {
                  "terms": {
                        "field": ”emailId"
                  }
            }
      }
}
```

```
[{
    "key": ”a@xxx.com",
    "doc_count": 1
}, {
    "key": ”b@yyy.com",
    "doc_count": 1
}, {
    "key": ”c@zzz.com",
    "doc_count": 1
}……]
```

Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

Example Queries:

#2 Retrieve all documents from Elasticsearch

```
_search?scroll=1m
{
      "query": {
            "match_all": {}
      }
}
```

```
{
    "scroll": "1m",
    "scroll_id" :
      "cXVlcnlUaGVuRmV0Y2T="
}
```

Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

Example Queries:

#3 Multiple levels of aggregations

```
{
    "aggs": {
        "date": {
                "aggs": {
                        "customer": {
                                "aggs": {
                                        "device": {
                                                "aggs": {
                                                        "type": {
                                                                "aggs": {
                                                                        "action":{
                                ...                                             }
}
```

Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

Example Queries:

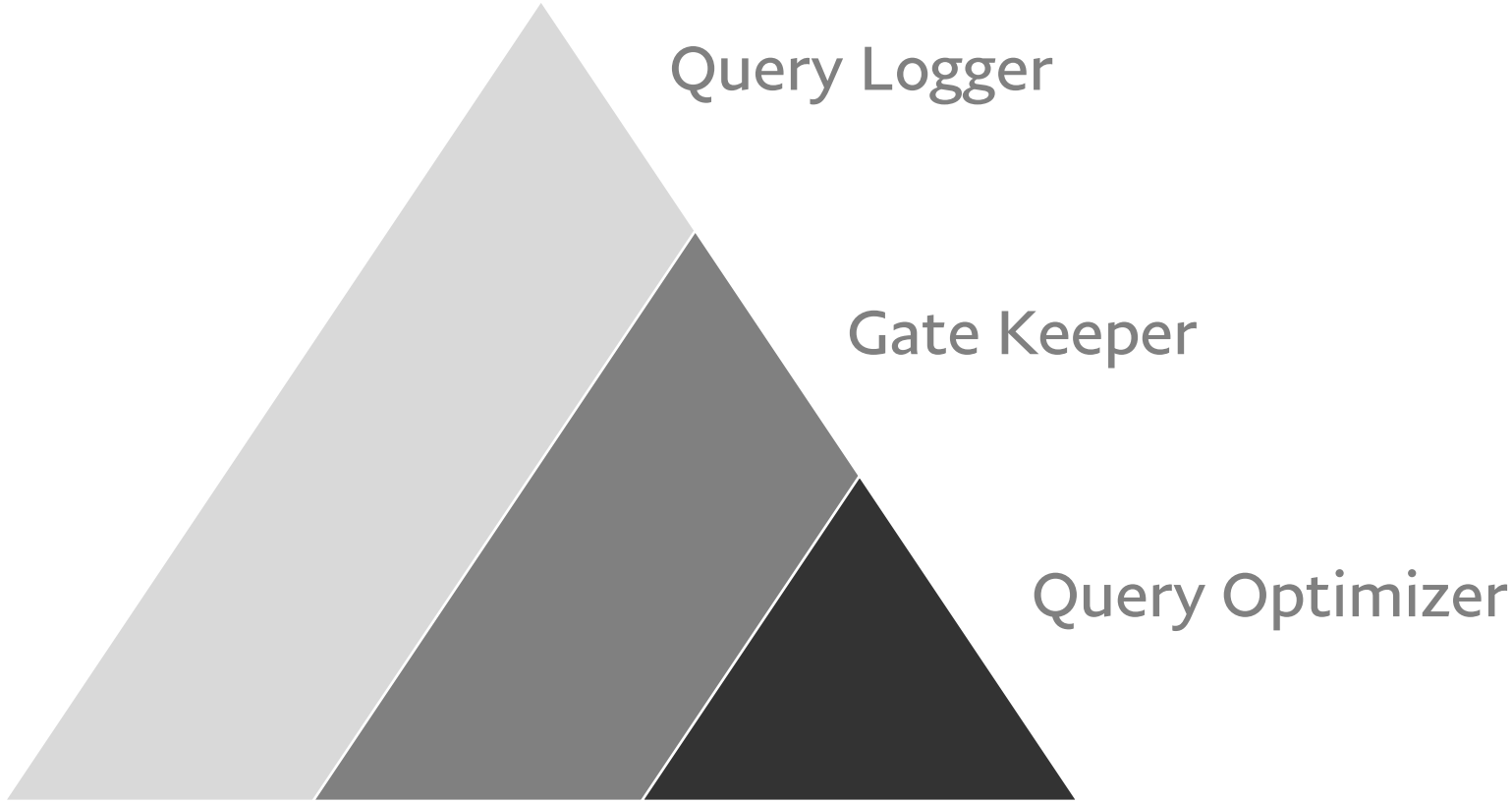
#4 Wildcard / Regex Queries

```
{
      "query": {
            "wildcard": {
                  "field": "*foo"
            }
      }
}
```

Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

Learnings



Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

Learnings : Query Logger

- Logging in Elasticsearch

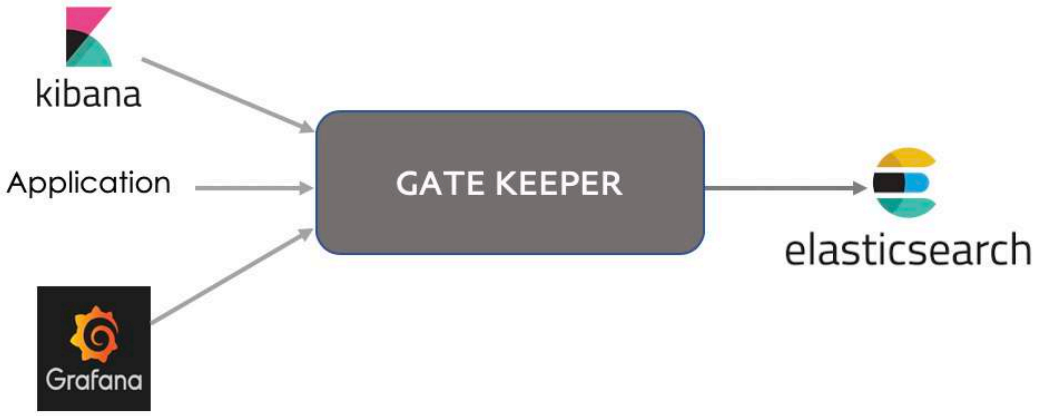
  Is `index.search.slowlog.*` suffice?

- Distinct Query Logger

Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

Learnings : Gate Keeper

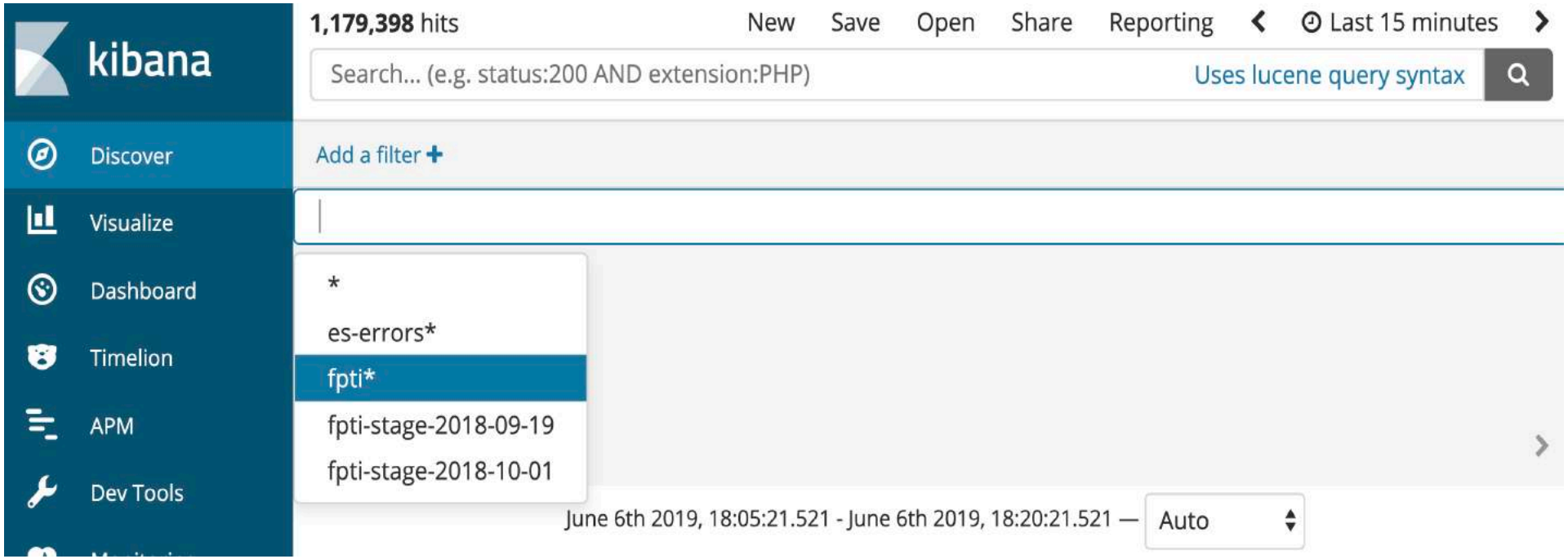


- Intercept and Act.
- Proactively prevent incidents in LIVE.
- Example: Time range > 48 hours | Nested Aggregations level is >10

Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

Learnings : Query Optimizer



Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

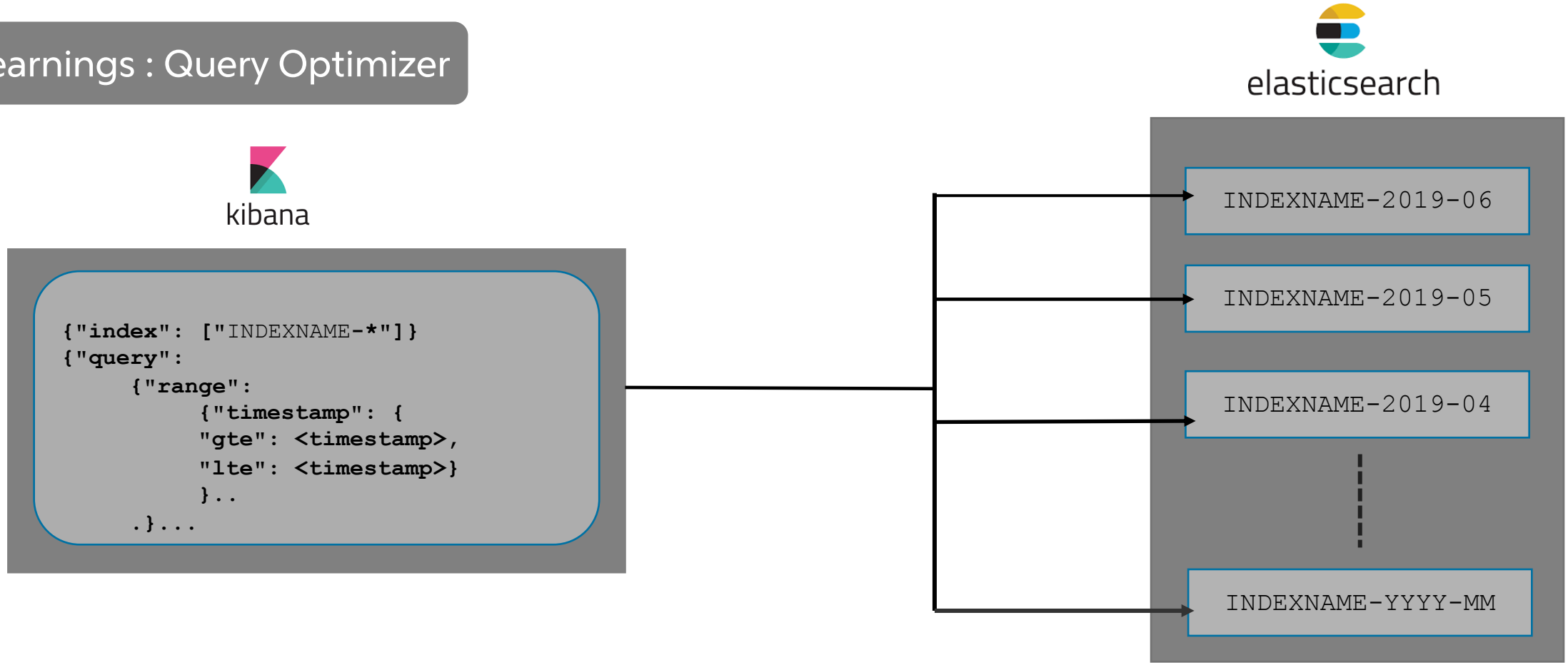
Learnings : Query Optimizer

```
{"index": ["INDEXNAME-*"]}
{"query":
   {"range":
      {"timestamp": {
      "gte": <timestamp>,
      "lte": <timestamp>}
      }..
   .}...
}
```

Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

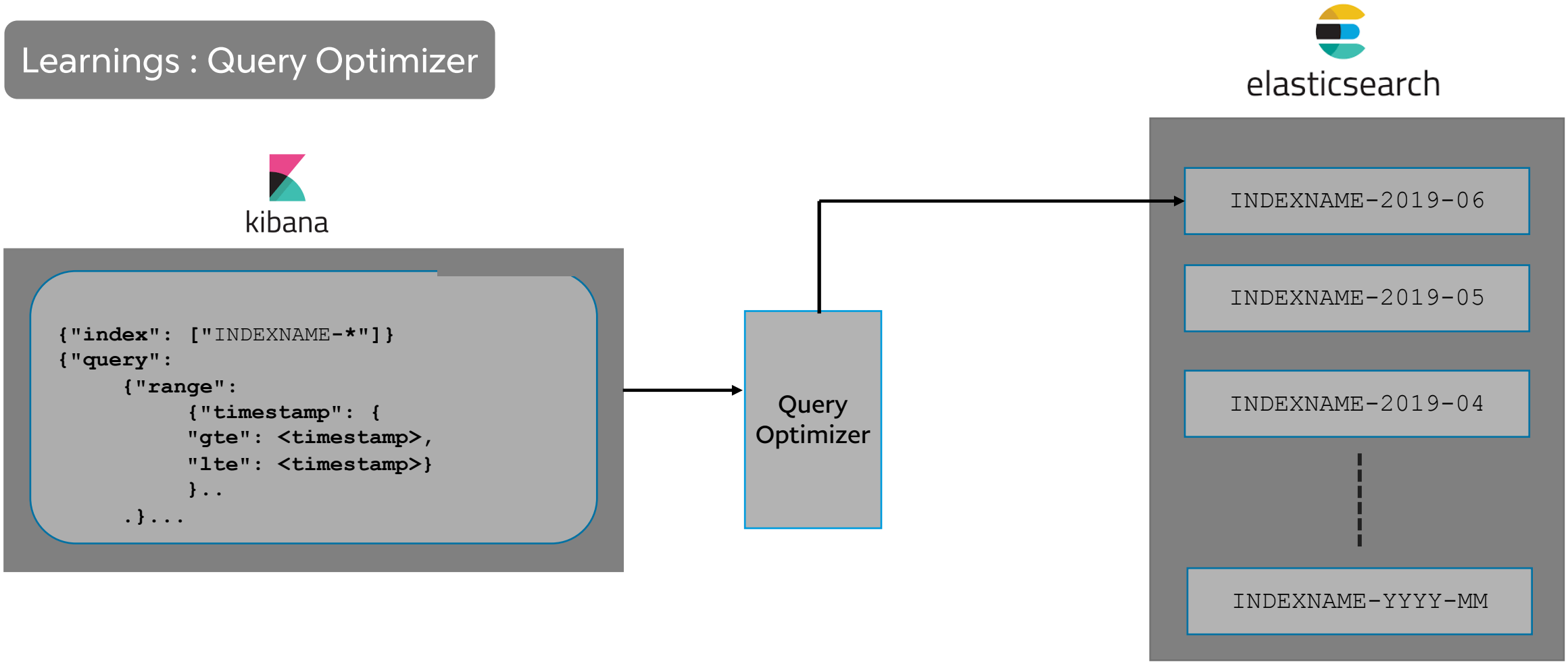
Learnings : Query Optimizer



Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

Learnings : Query Optimizer



Scaling Elasticsearch!

Problem #3 : Expensive/sub optimal queries – A threat to availability!

Learnings : Query Optimizer

- Reduce the number of indices(shards)that are being queried.
- Reorder the queries for reducing the scan margin *

⬇ Search latency by 25% and we observed better availability ☺

Scaling Elasticsearch!

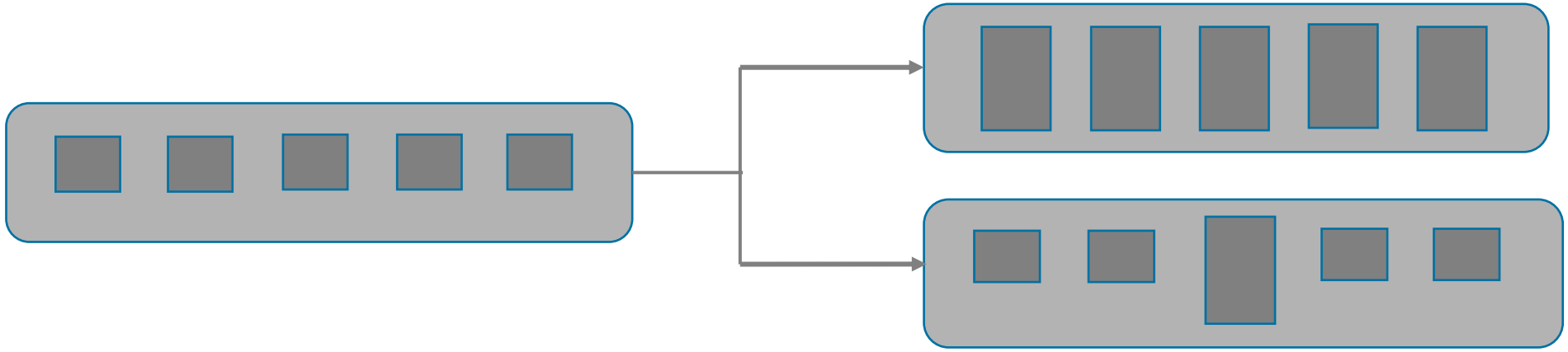
Problem #4: Oversized Shards

When?

- Shards assigned per index is incorrect.
- Skewness in the values of the routing key.

Learnings

- Anticipate the future and assign the shards.
- Split
- Re-index



Managing Elasticsearch!

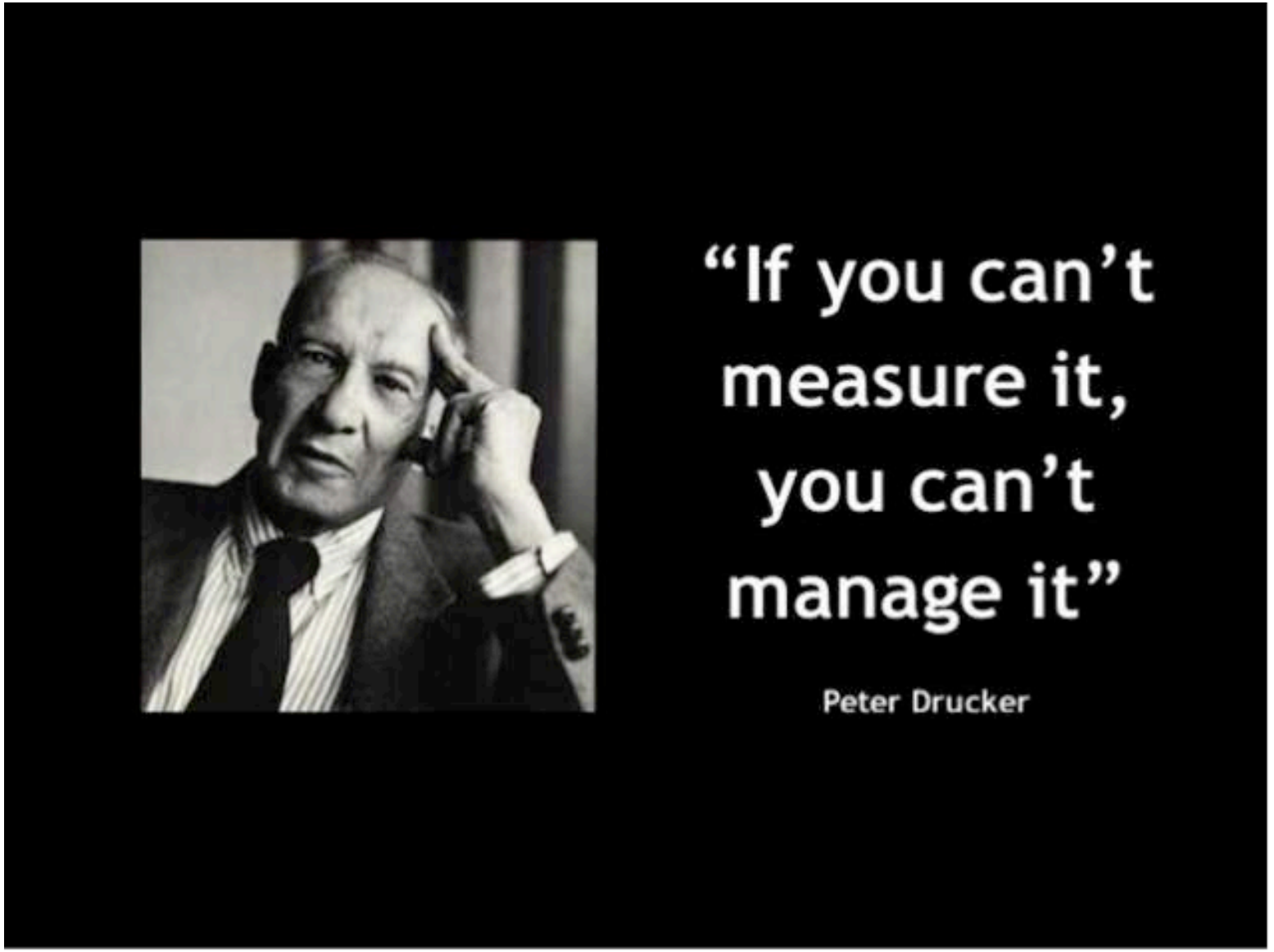
Managing Elasticsearch!

- Monitoring your clusters
- Securing Elasticsearch
- Upgrading Clusters

How do we police the Police?

When Availability, is the top priority!

Monitoring your clusters!

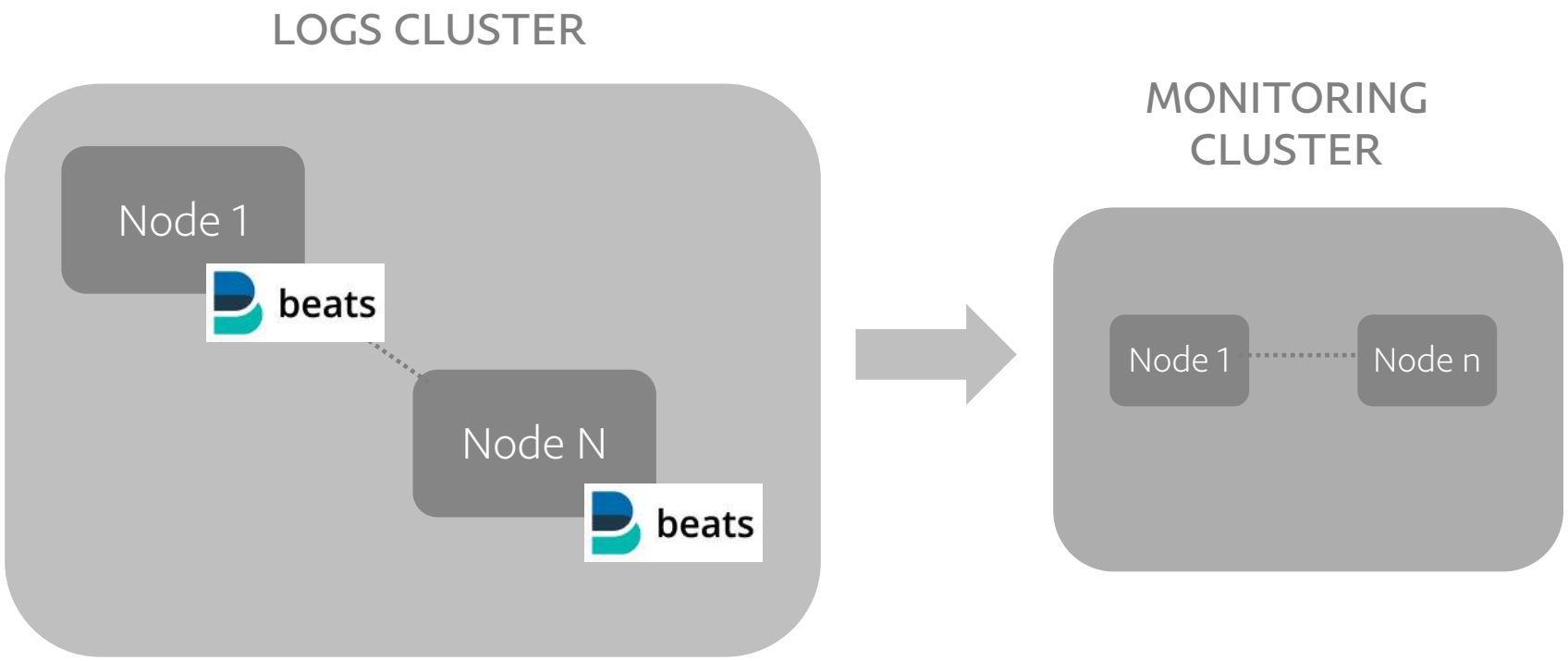


Managing Elasticsearch!

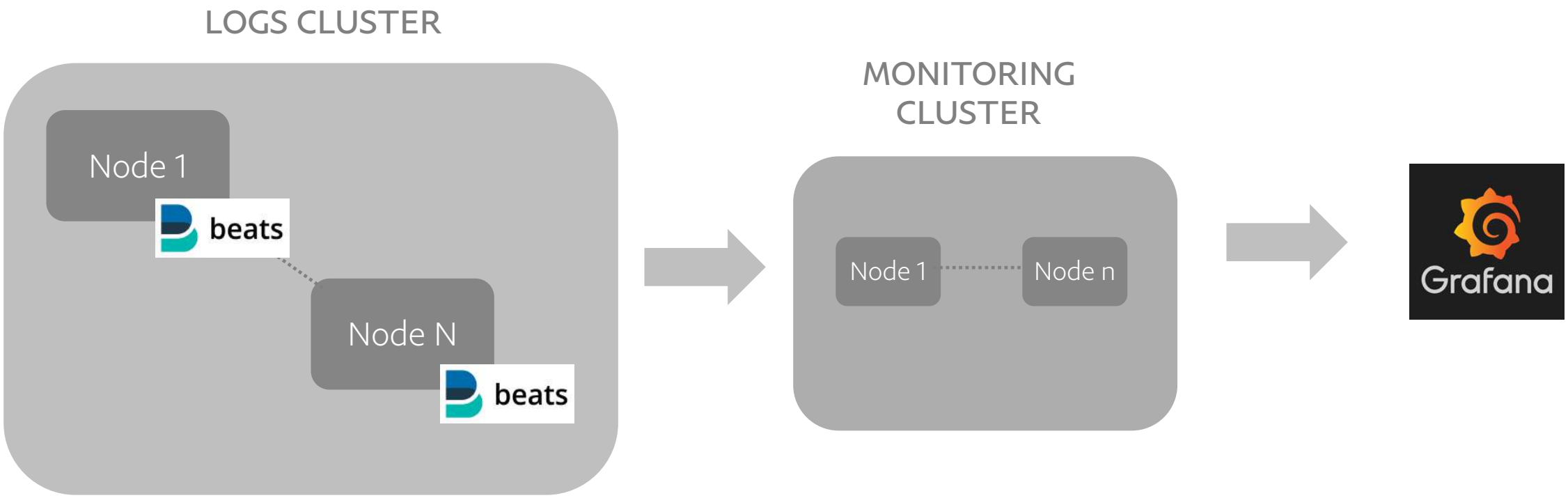
Collect Node stats from **every** node

- Index — `field data, docs, merge, refresh`
- Shard — `type, state, docs`
- Thread Pool — `type, queue, rejected, max`
- JVM — `heap_used, gc collection count and time, buffer pools`
- Os & Process — `CPU Load, Memory, File descriptors`

Managing Elasticsearch!



Managing Elasticsearch!



Managing Elasticsearch!

COLLECT | SHIP | ANALYZE | **ALERT** | ACT

Securing Elasticsearch!

Why do we need security?

In-House Guard Plugin

- End to End TLS/SSL
- Authentication & Authorization

Security features, free in Elasticsearch?

- Audit Logging
- Third party integrations with LDAP/SAML

Upgrading Elasticsearch!

It’s tough only when you are in <5.X versions!

Must do checks:

- Are queries backward compatible?
- Mapping changes between versions?

Approach 1: Parallel cluster with dual ingestion

Approach 2: Re-index from remote


Demo

```
"hits": {
  "total": 11748840,
  "max_score": 1,
  "hits": [
    {
      "_index": "demo",
      "_type": "file",
      "_id": "WwUcRWsBqsrfqUax3C-c",
      "_score": 1,
      "_source": {
        "correlationId": "FgPjzpga6r",
        "applicationName": "application-13962406",
        "api": "v1/tenant/demo/test-13962406"
      }
    },
    {
      "_index": "demo",
      "_type": "file",
      "_id": "XgUcRWsBqsrfqUax3C-c",
      "_score": 1,
      "_source": {
        "correlationId": "7l1oefSR5U",
        "applicationName": "application-13962409",
        "api": "v1/tenant/demo/test-13962409"
      }
    },
    {
      "_index": "demo",
      "_type": "file",
      "_id": "ZAUcRWsBqsrfqUax3C-c",
      "_score": 1,
      "_source": {
        "correlationId": "mdOu4JwvgJ",
        "applicationName": "application-13962415",
        "api": "v1/tenant/demo/test-13962415"
```

Q & A