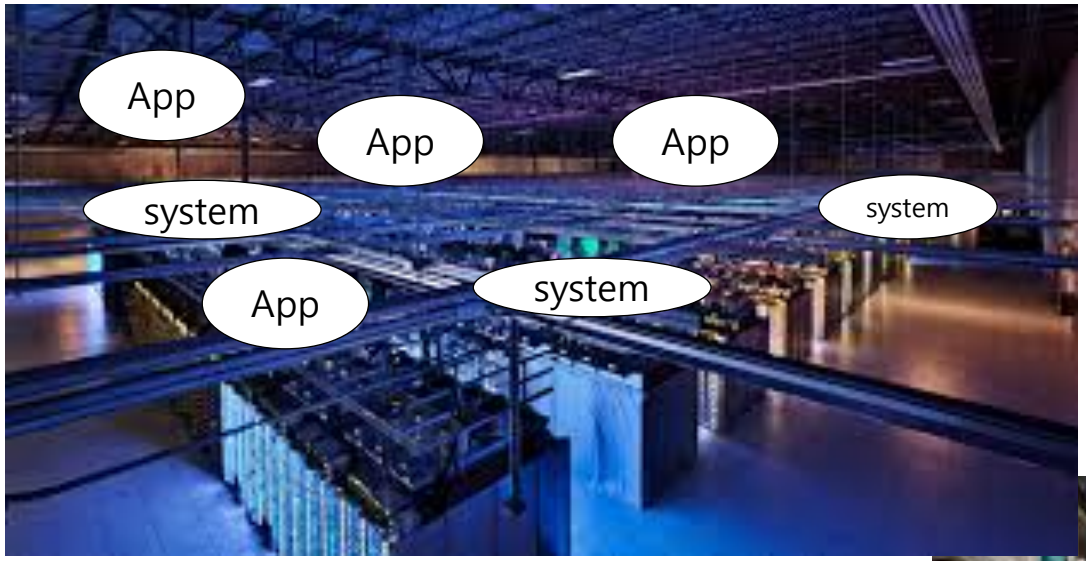
Operational Intelligence

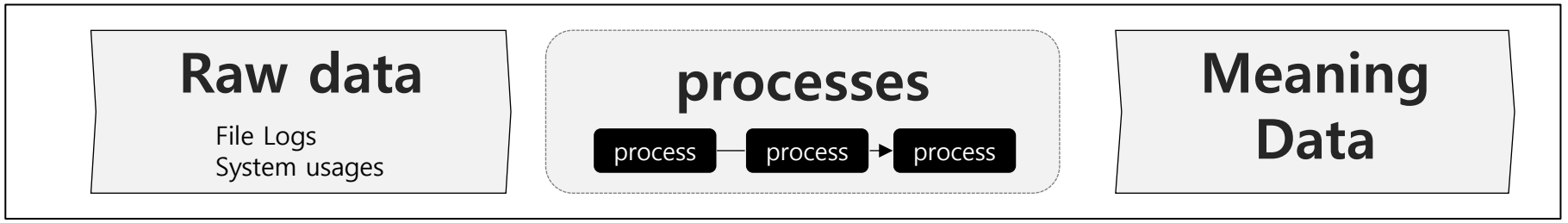
Written by Ted.Jung
ted.jung@elastic.co

Story – Current status



- ***Combine*** **objects**
- **Line-up processes *simply***
- ***Operate Proactive*****ly**

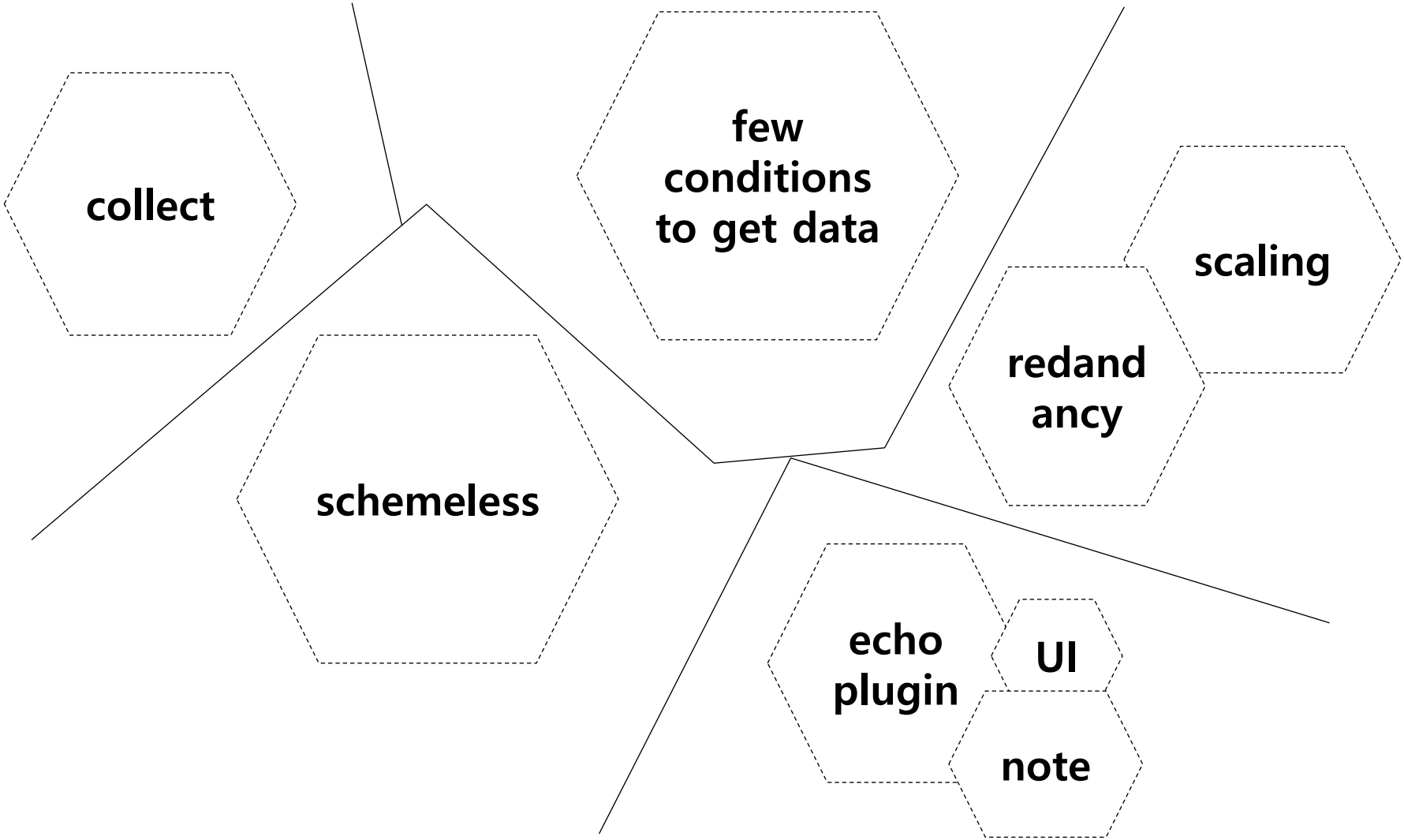
Story – pipelining



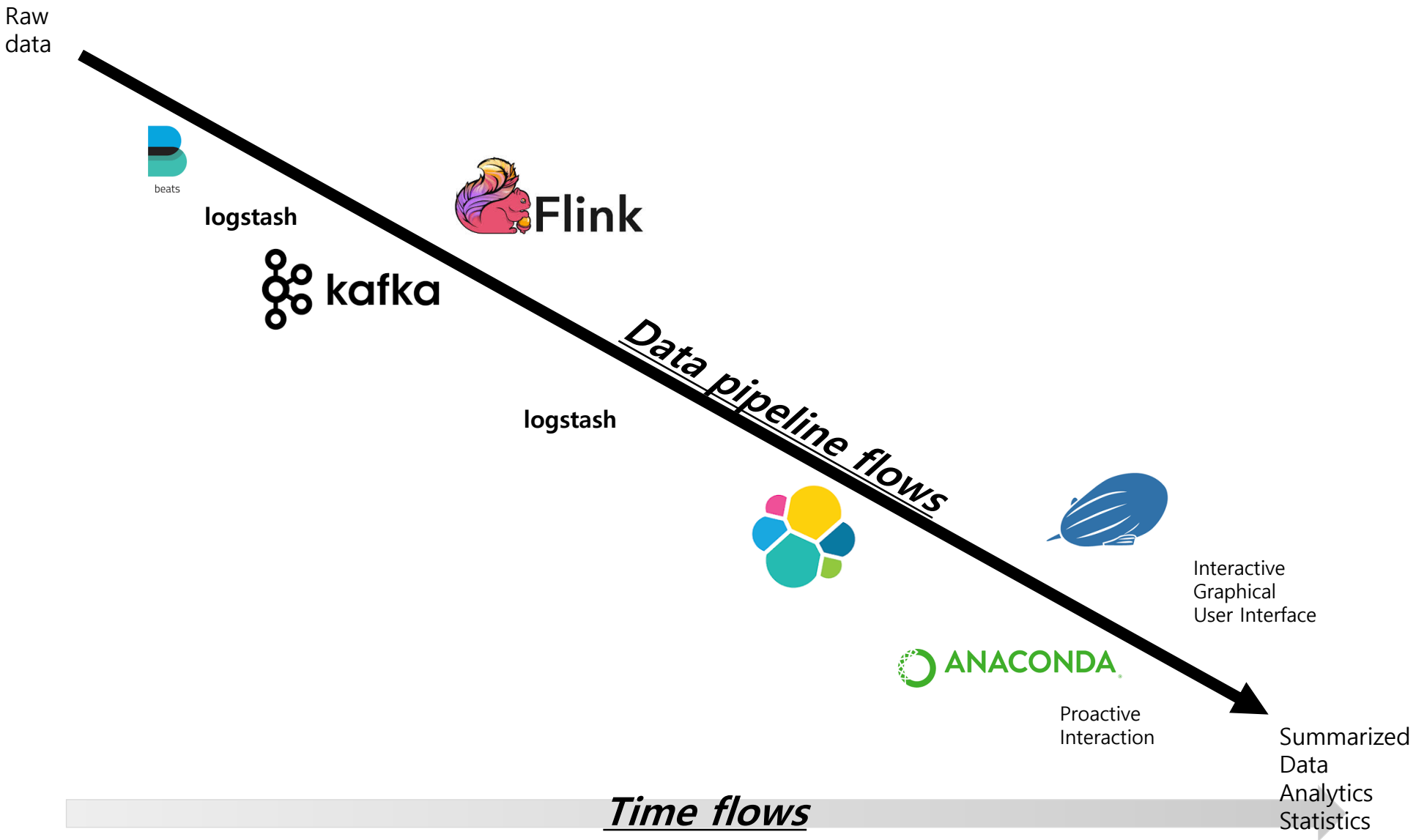
Which Data need to be collected?
How?
- Make a schema on DB
- Table with many attributes
- Normalization
- Insert
- Many Select Queries
- HTML with business logic

Changes (just add a new single attribute)

No options to improve?



Open source solutions



Current Image

What types of data?

Concerns?

⇒ Traditional, 3-Tiers application

Web (Nginx, Haproxy, Apache)

Was (Tomcat)

DB (MySQL, Redis, etc)

&

System (H/W-CPU, Memory, Disk, Network, Temperator, Fan)

Q) How to collect with which tool?

datas of which shape of format?

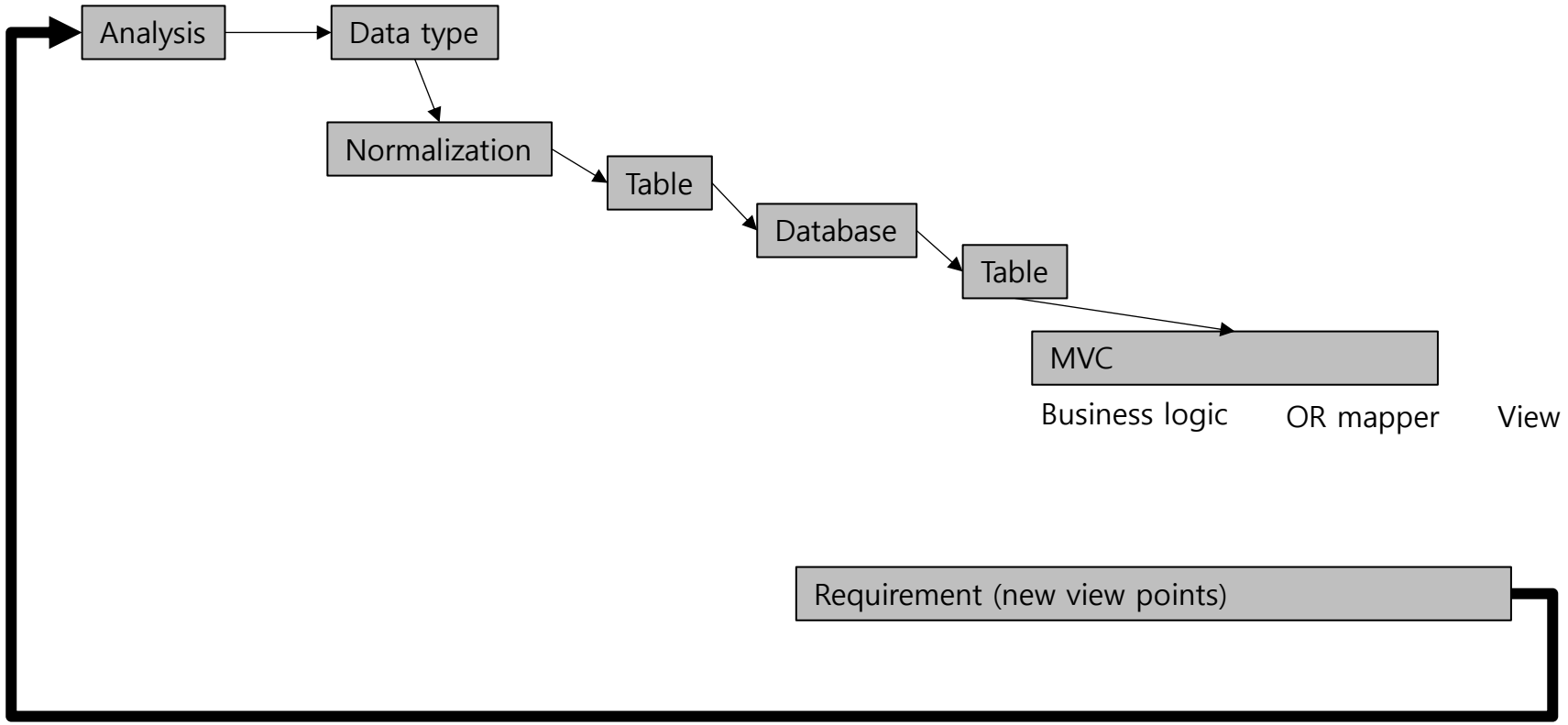
Q) How to store data to where?

Q) How to presents it? (just text or graphically by interaction)

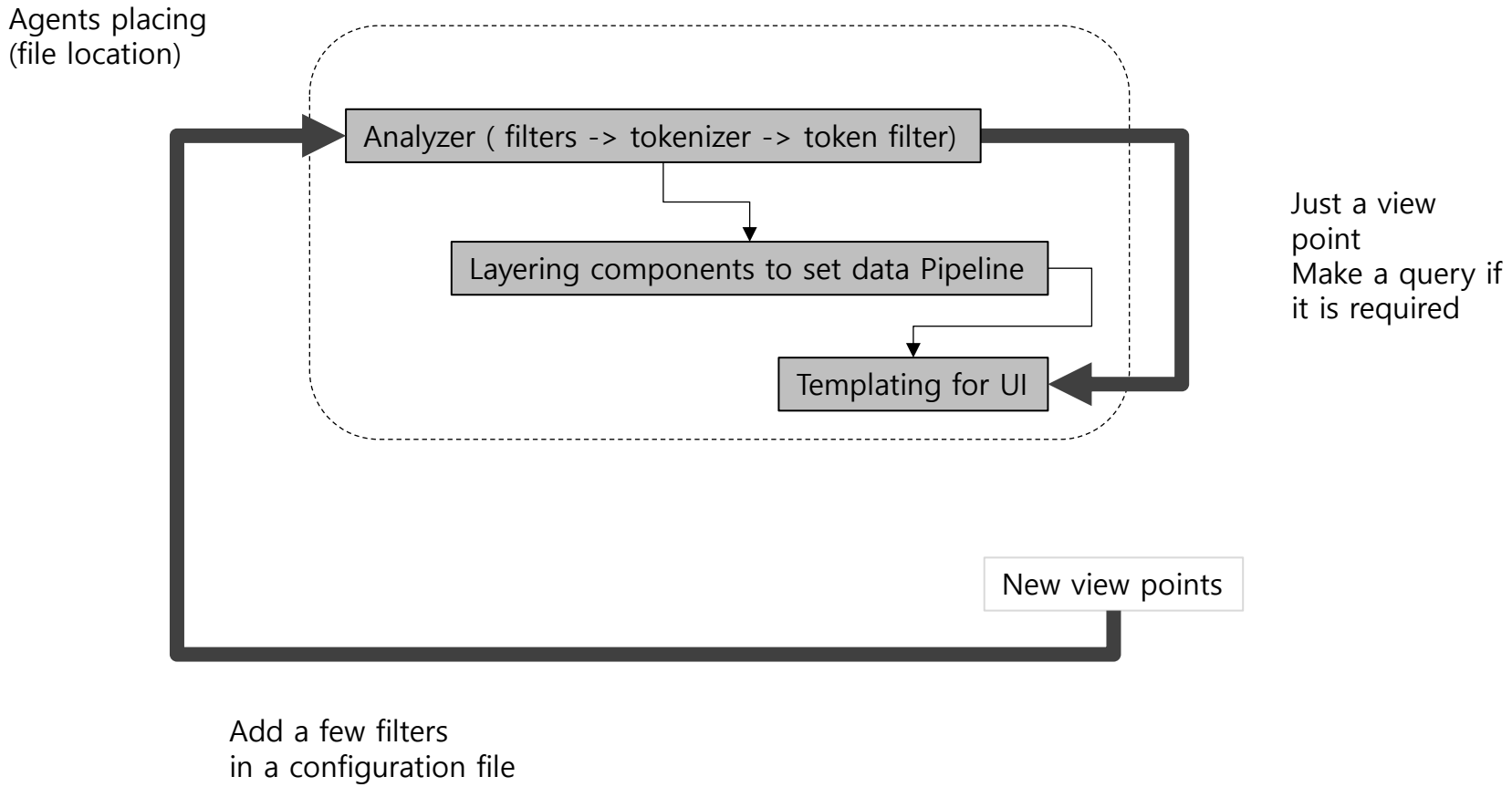
Portal v.s Enterprise Svc

Dynamic	Static
Flexible	Fixed
Results with a single keyword	Conditions
	

Traditional Processes



Another approach to replace the *traditional* with new differently



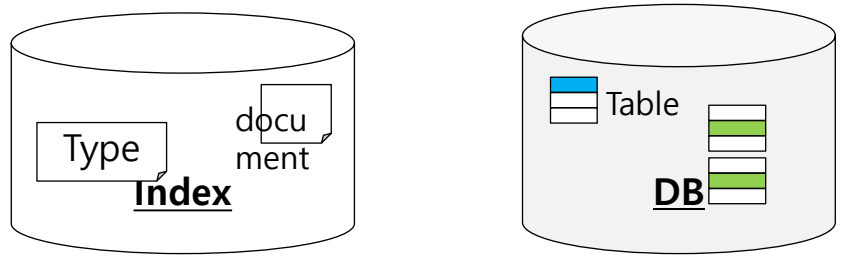
Dive Into (index)

Elasticsearch data storage

- Analogy

Relational DB	Elasticsearch
Databases	Indeces
Tables	Types
Rows	Documents
Columns	Fields
Schema	Mapping
Index	Everything is indexed
SQL	Query DSL

RDBMS Row Data .vs Search Index



Here is some changes
No more the type should be looked to a table in RDB.

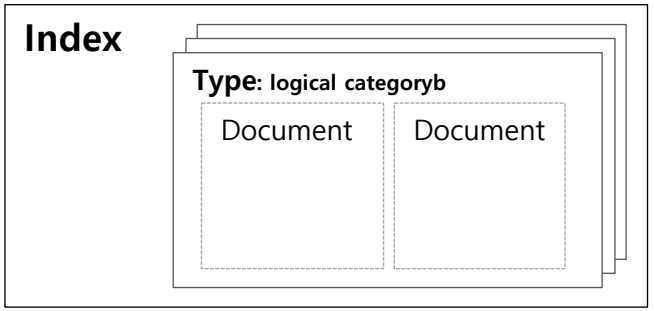
Dive Into (sharding)

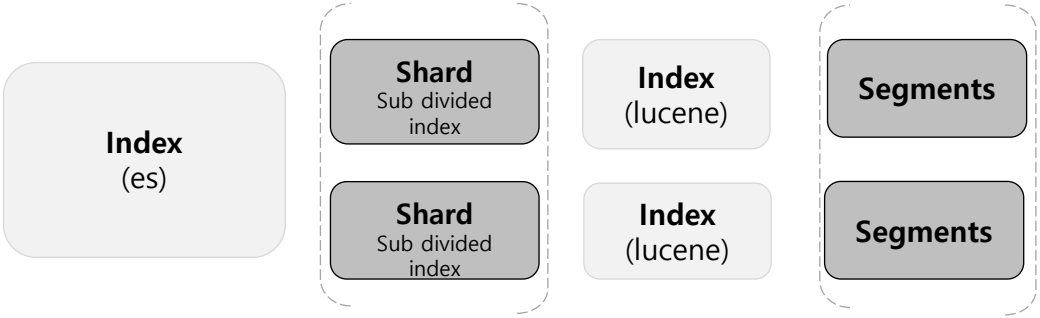
- The act of storing data in Elasticsearch is called *indexing*.
- In Elasticsearch, a document belongs to a *type*, and those types live inside an *index*.
- An index is just a logical namespace that points to one or more physical shards.
- Documents are stored and indexed in shards, but our applications don't talk to them directly. Instead, they talk to an index.

- Shards are allocated to nodes in your cluster. As your cluster grows or shrinks, It will be automatically migrated to shards between nodes by Elasticsearch so that the cluster remains balanced.

- Each shard has 0 or more replicas. A shard can be either a primary shard or a replica shard. (A replica shard == just a copy of a primary shard.)

- Each document in index belongs to a single primary shard. While there is no theoretical limit to the amount of data that a primary shard can hold, there is a practical limit based on hardware etc.





Elasticsearch Cluster

Elasticsearch Cluster

A group of nodes with the same cluster.name that are working together (share data and to provide failover and scale)

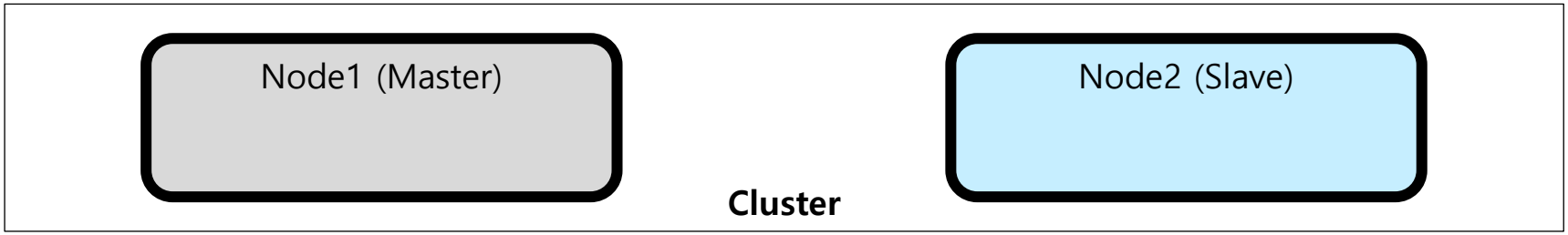
Elasticsearch Plugins

easily expand the functionality of ElasticSearch through plugins.
: bin/plugin –install <plugin name>

Elasticsearch Cluster

Roles of Nodes

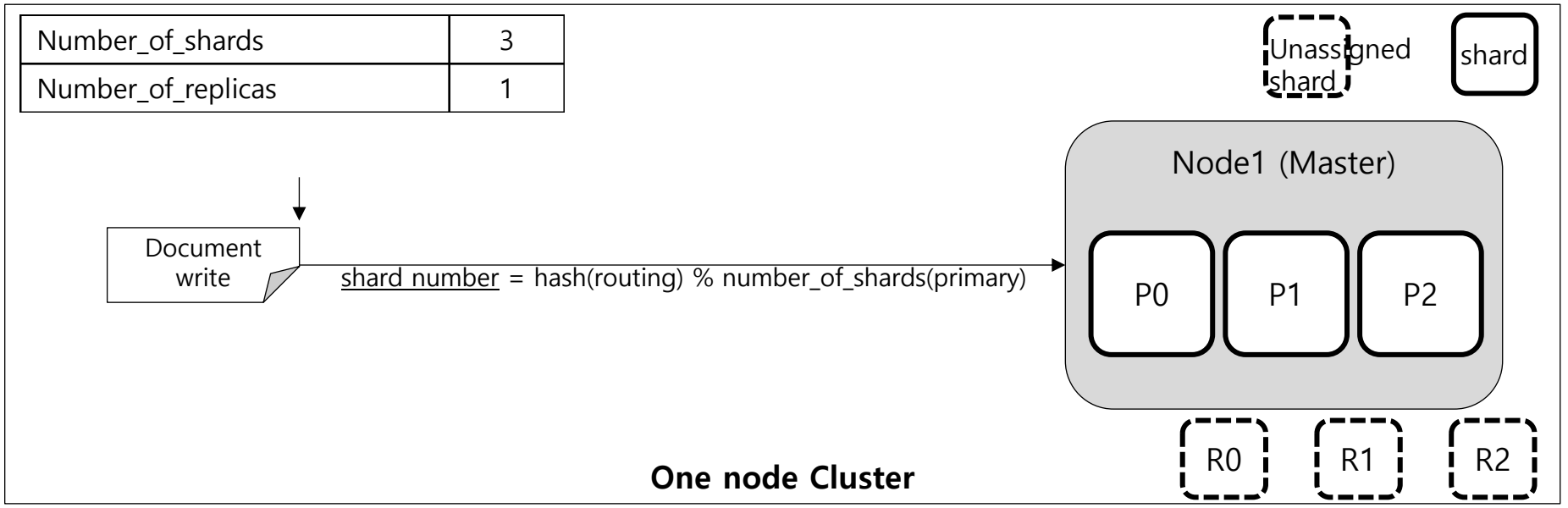
1) self-manages
2) load balance the location of all shards across all nodes.
3) the master node maintains the shard routing table
4) distributes a copy of the shard routing table to other nodes within the cluster.



Coordinating Node in ES cluster behaves as a "smart load balancer"

Request(Cluster level)	Request(Data related)	Coordinating node
Master node managing cluster-wide changes Index(creating/deleting)Node(adding/removing)	Data node	

Writes a Data



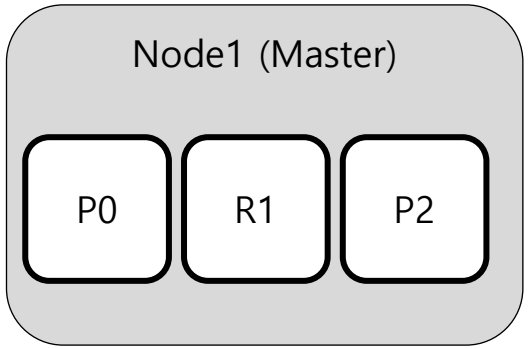
- Calculate the shard on which to store the document
  : Gives us the number of the shard where a particular document lives
  Default: (5 shards , 1 replica) / index

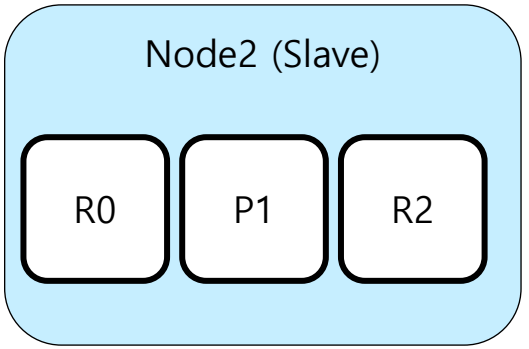
Status: “yellow”

```
"cluster_name": "elasticsearch",
"active_primary_shards": 3,
"active_shards": 3,
"relocating_shards": 0,
"initializing_shards": 0,
"unassigned_shards": 3
```

Writes a Data

Number_of_shards	3
Number_of_replicas	1



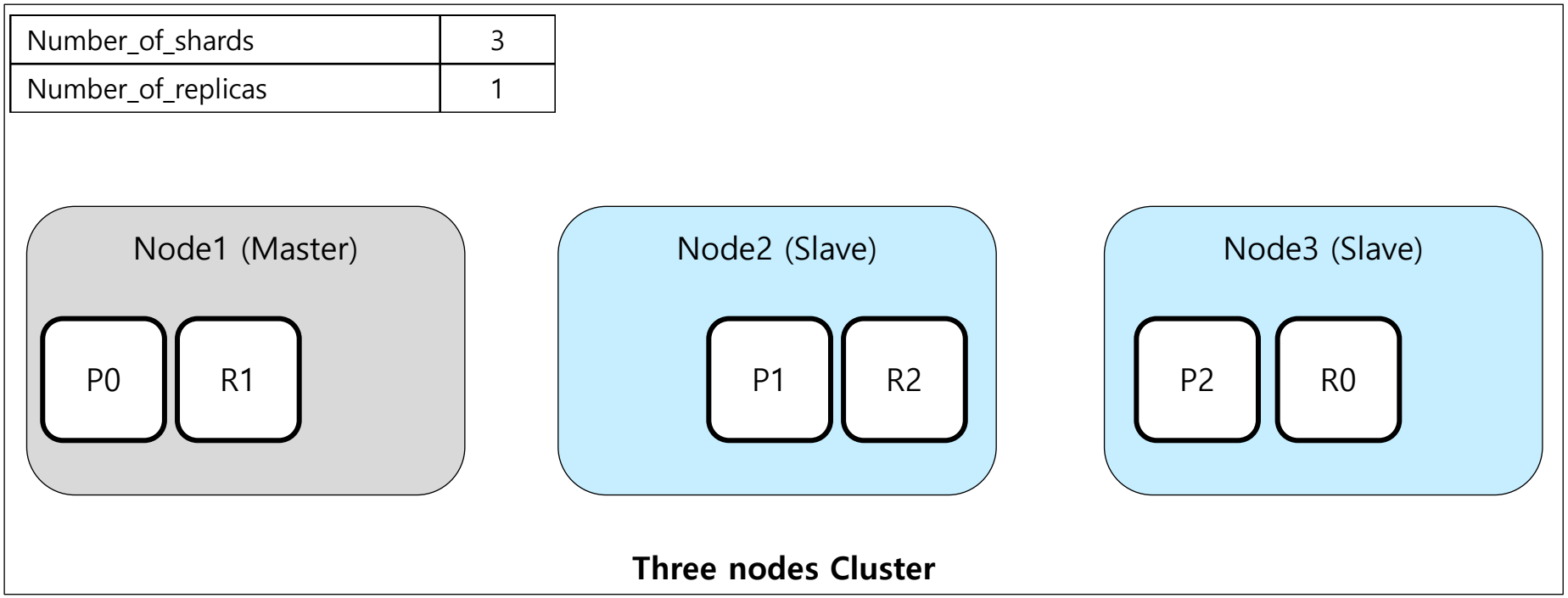


"cluster_name": "elasticsearch",
"status": "green",
"timed_out": false,
"number_of_nodes": 2,
"number_of_data_nodes": 2,
"active_primary_shards": 3,
"active_shards": 6,
"relocating_shards": 0,
"initializing_shards": 0,
"unassigned_shards": 0

Two nodes Cluster

Writes a Data

Number_of_shards	3
Number_of_replicas	1

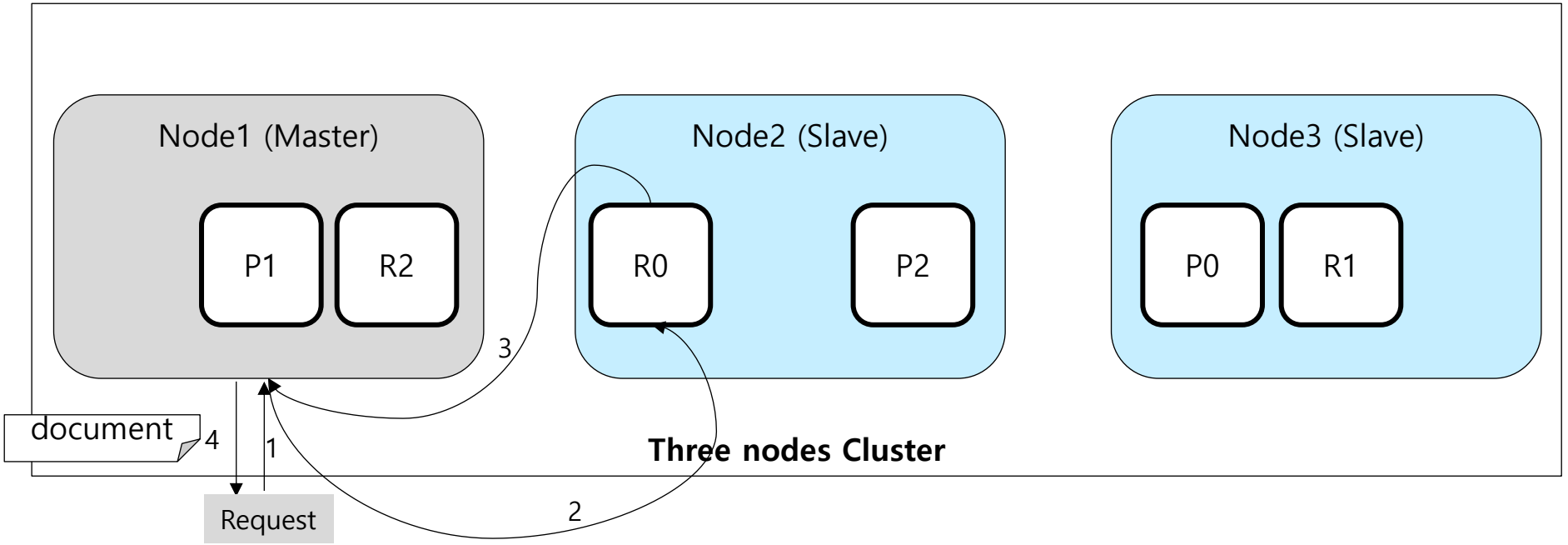


One shard each from Node 1 and Node 2 have moved to ~
~ the new Node 3 We have two shards per node instead of three

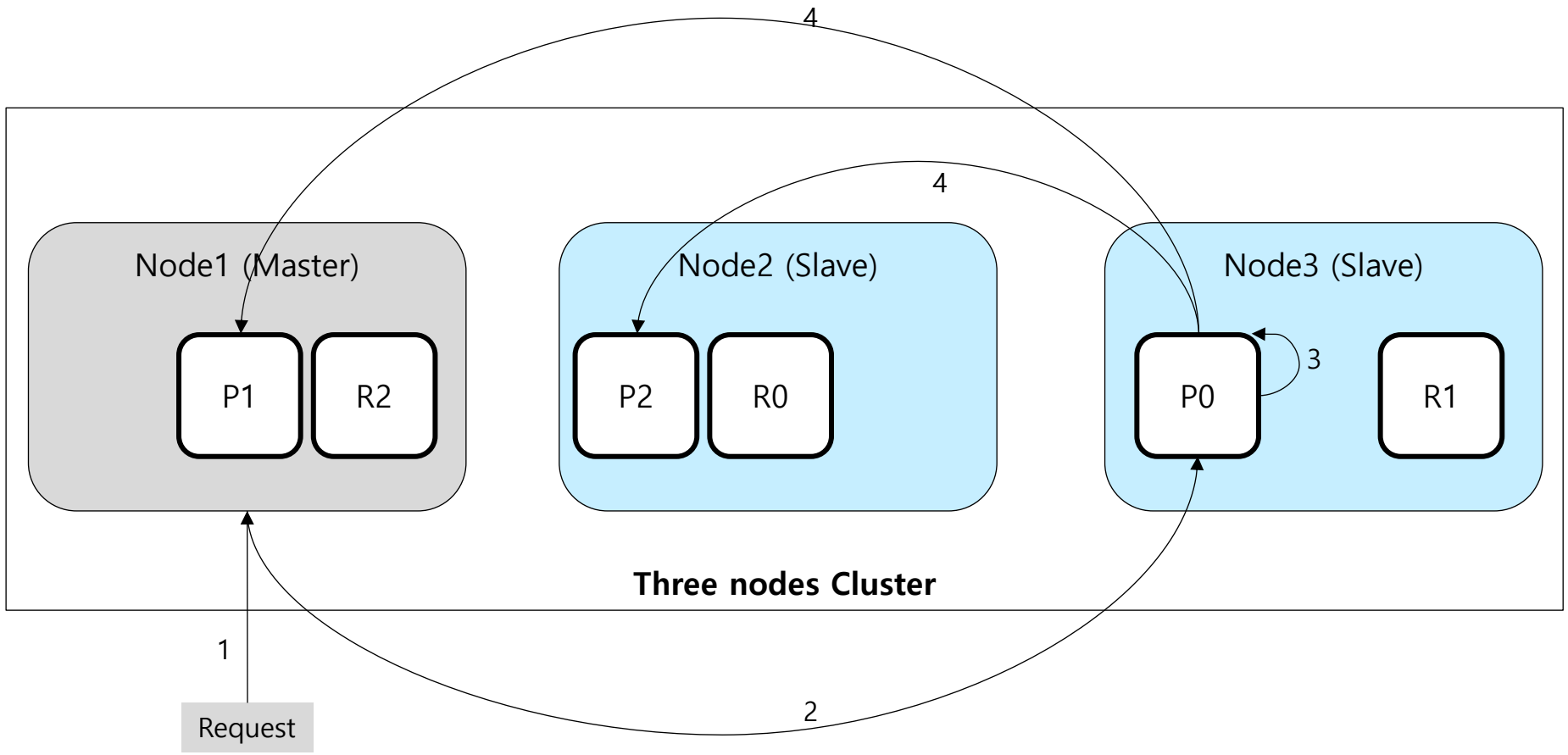
Retriving a Document

read requests(document retrieval)can be handled by a primary *or* a replica shard,

the more copies of data that you have, the more search throughput you can handle

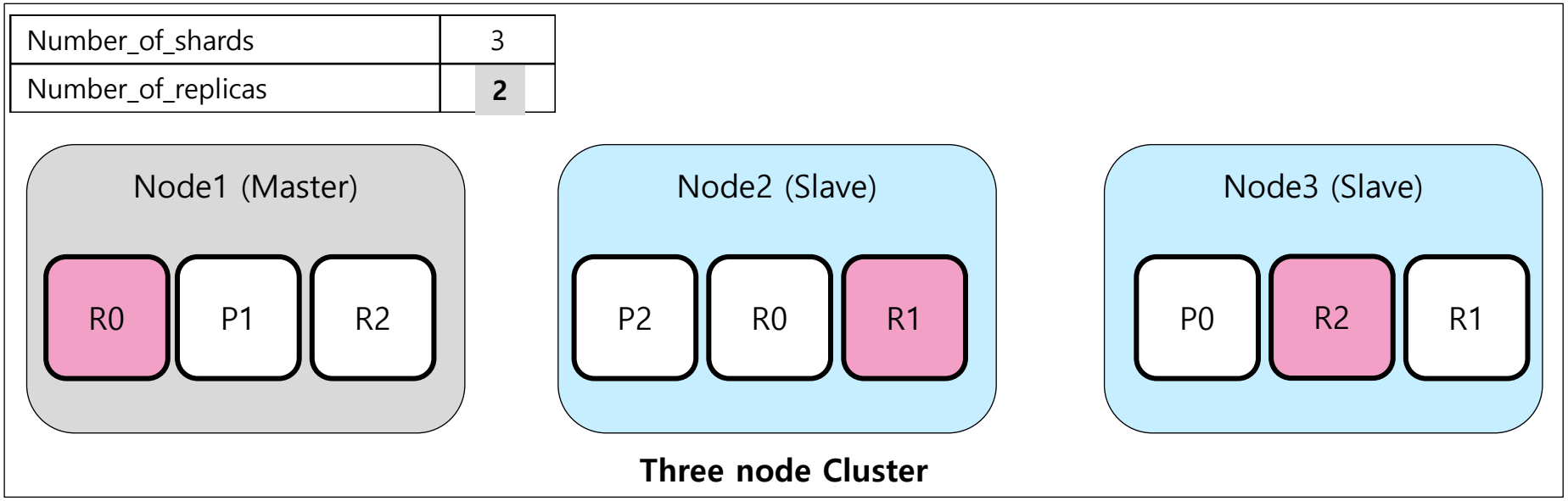


Updates to a Document



Replica changes

The number of replica shards can be change dynamically on a live cluster



Three Primaries
Six Replicas

Now we has nine shards

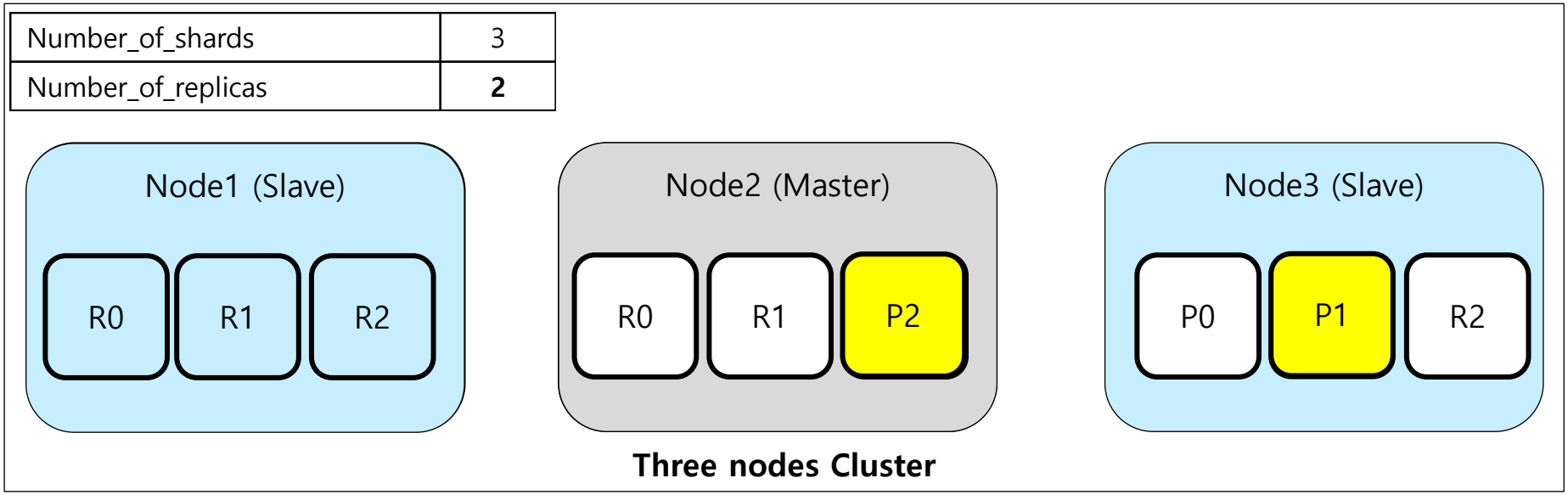
Can scale out to a total of nine nodes

Triple search performance

※ Can afford to lose two nodes without losing any data

Coping with Failure(HA)

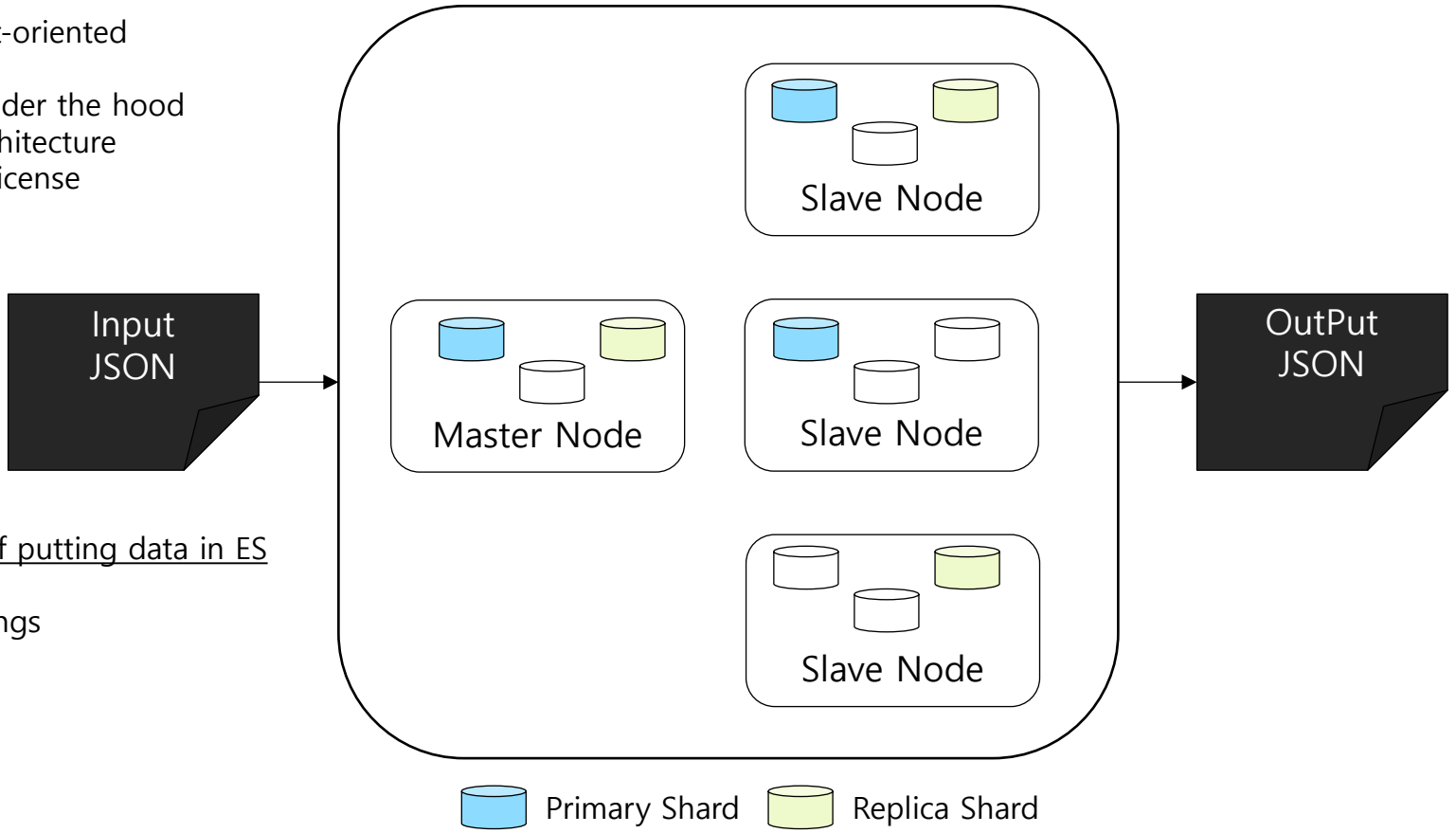
Number_of_shards	3
Number_of_replicas	**2**



- Kill the master node
- One of Nodes be elected as a new master(Node 2)
- Primary shard P1, P2 are missing
- Status: Red (not all primary are existing)
- Promote the replicas of these shards on Node 2 and Node 3 to be primaries
- Status: Yellow

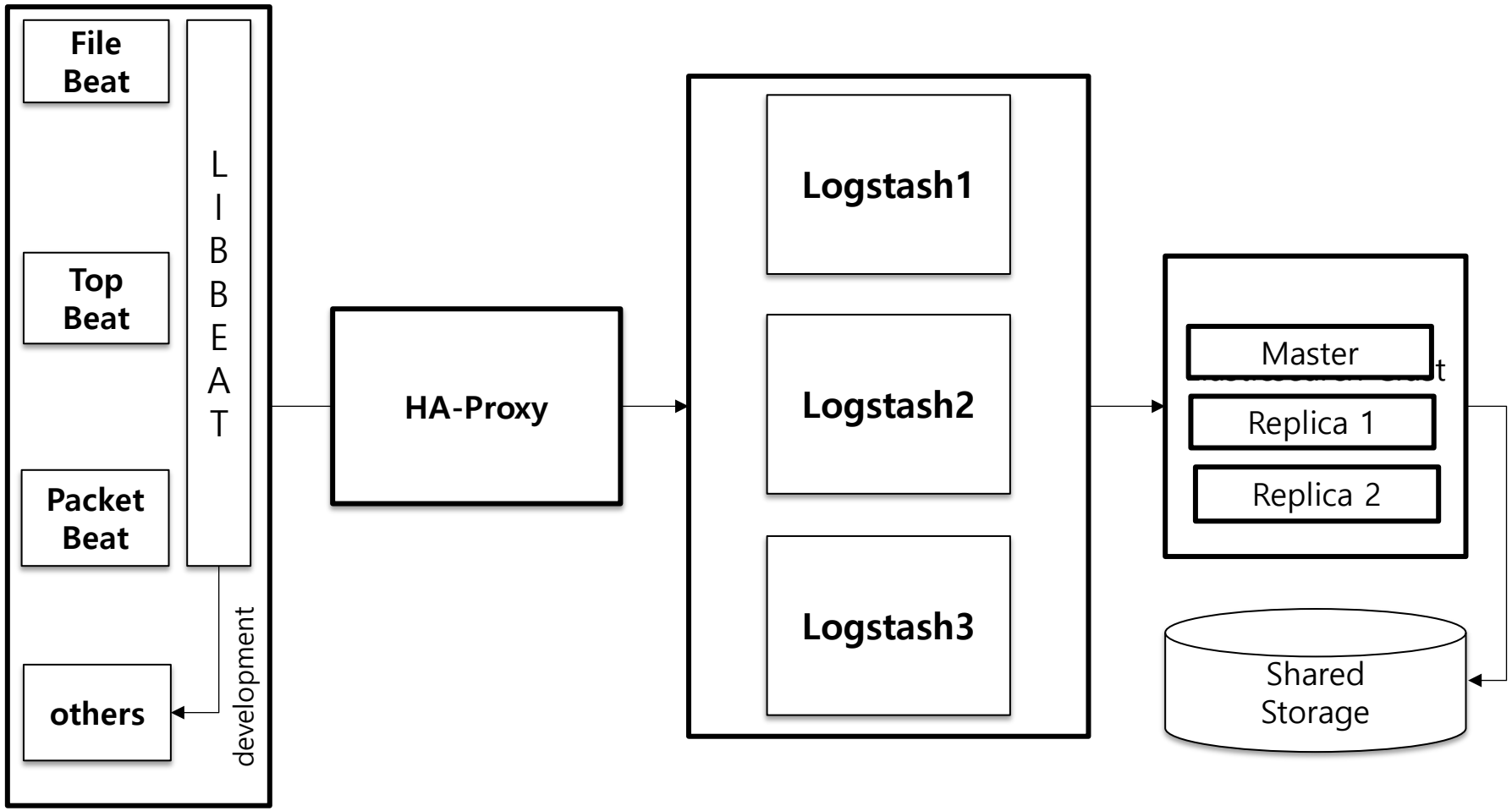
Elasticsearch Cluster

Sclable
Document-oriented
Rest&json
Lucene under the hood
Plugin architecture
Apache2 license

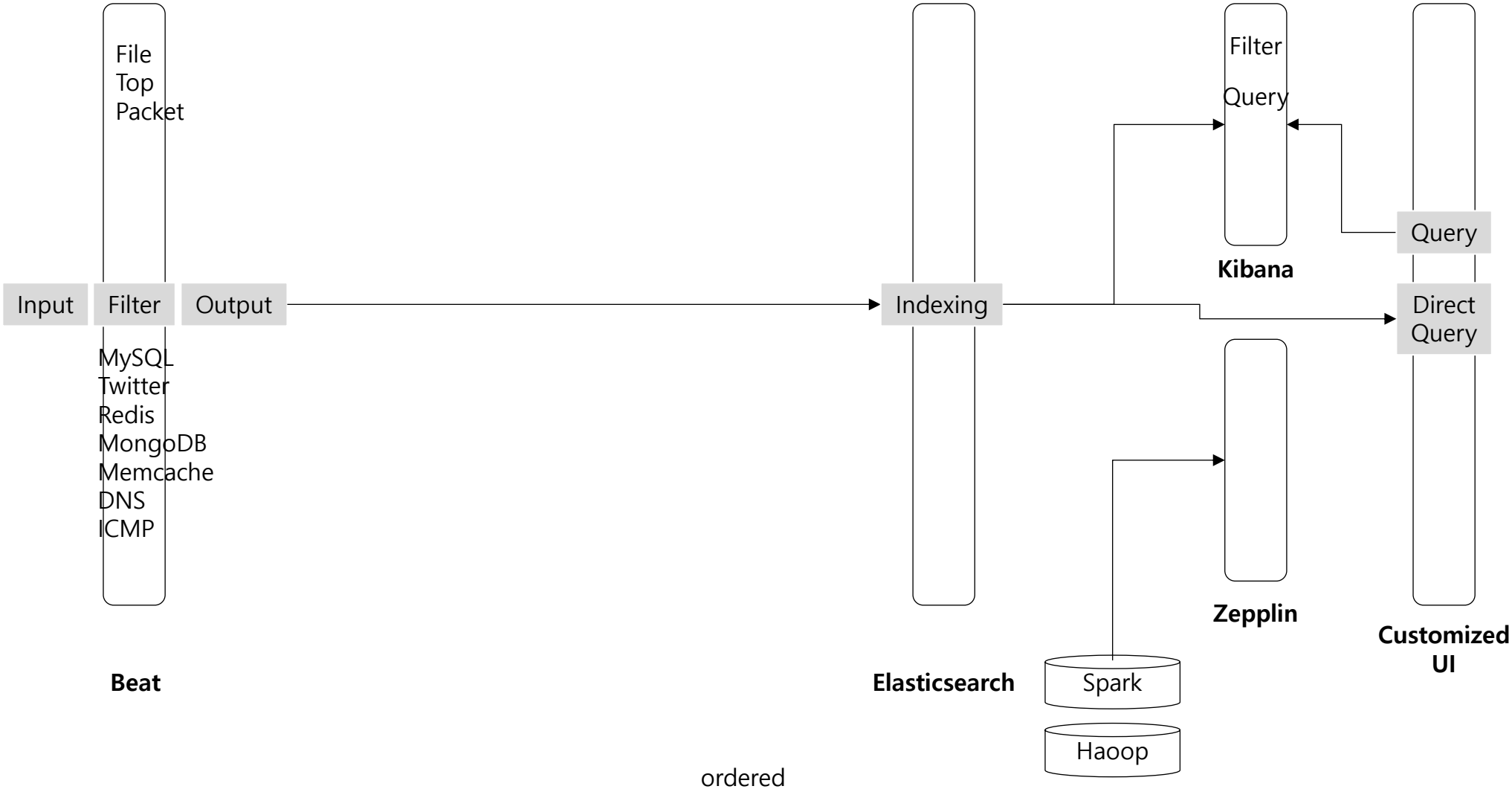


<u>The way of putting data in ES</u>
Index api
http bindings
Logstash
Rivers
Flume
Fluentd

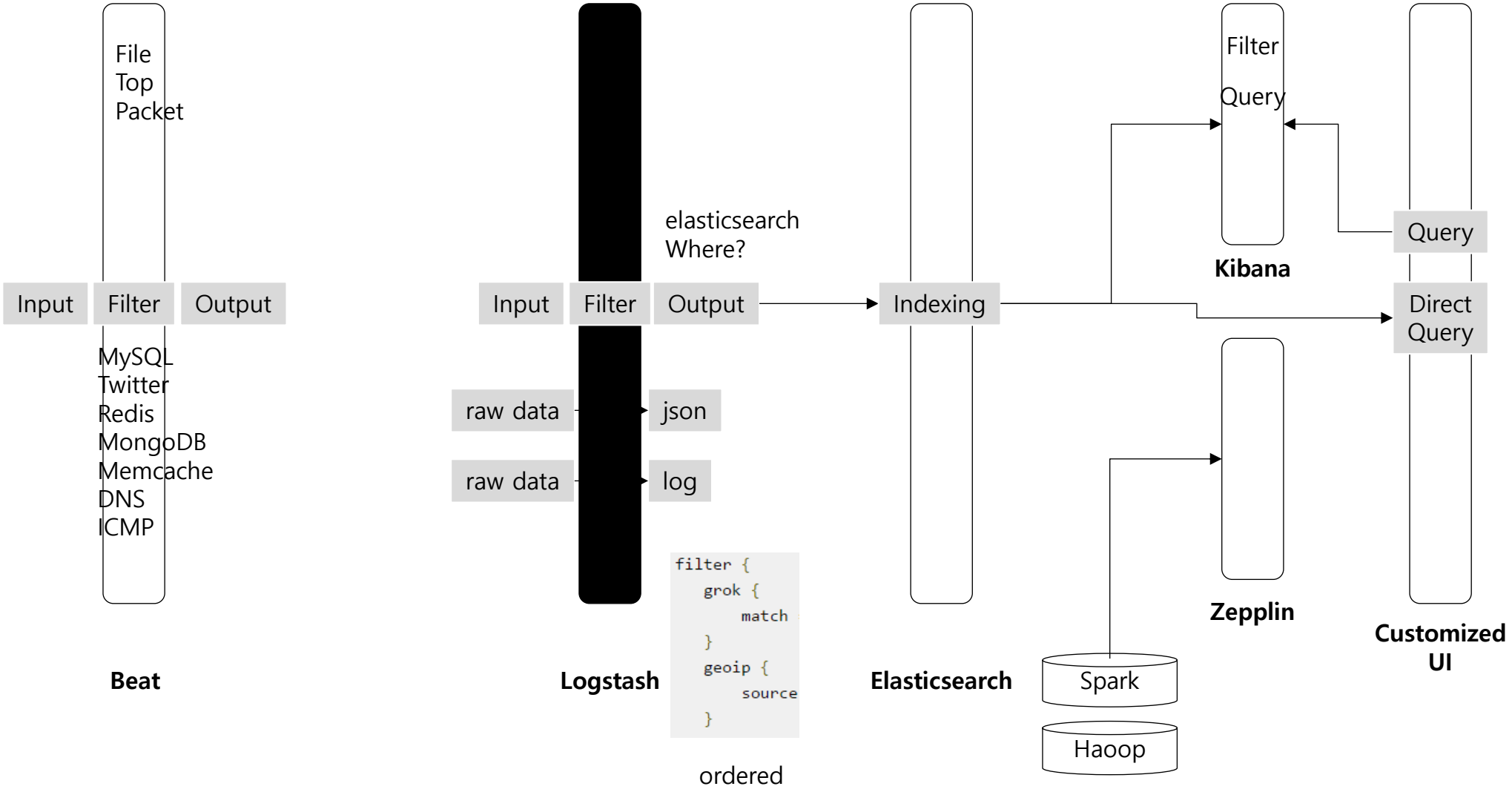
System Layout



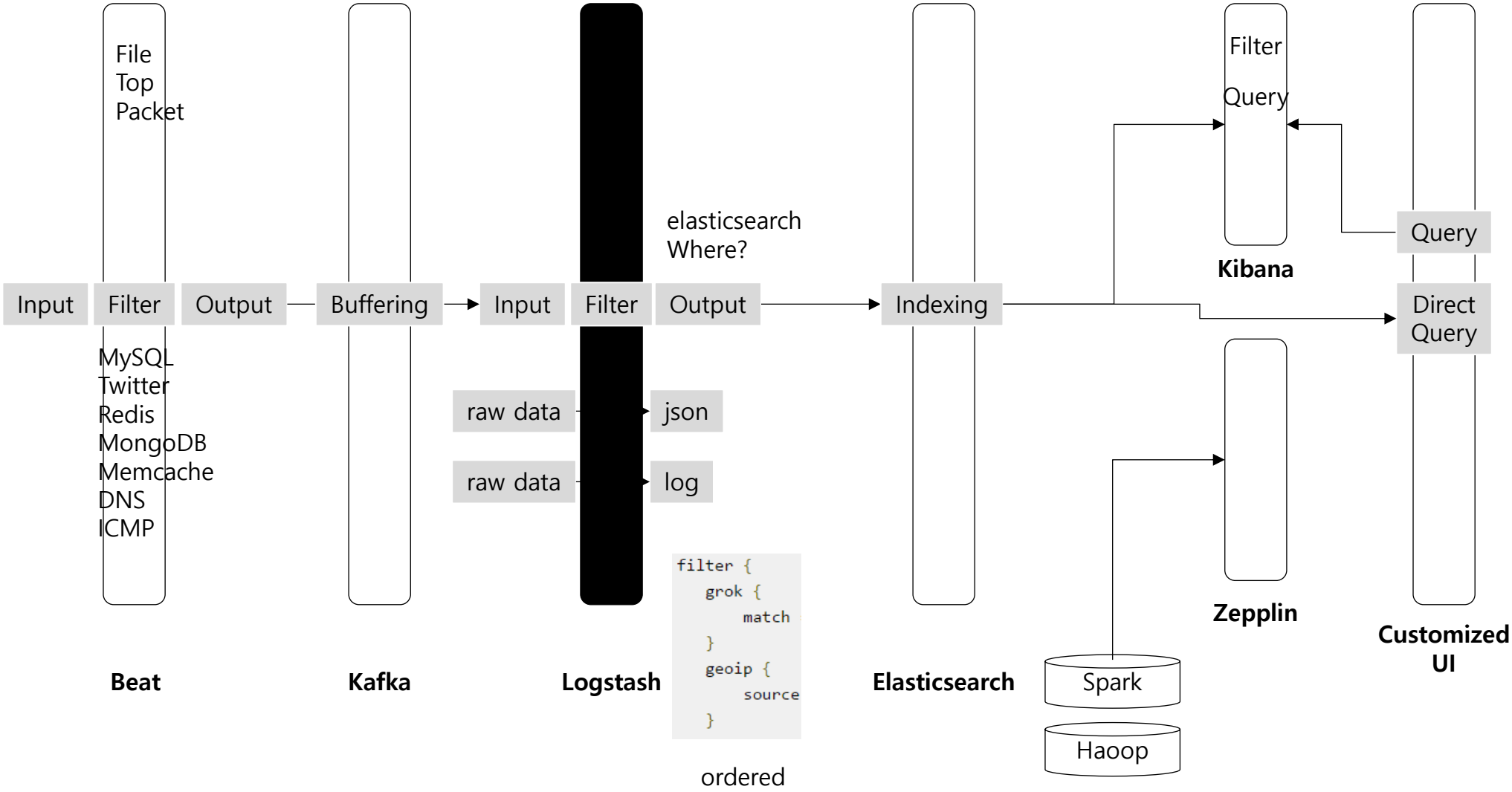
Data Pipeline(I)



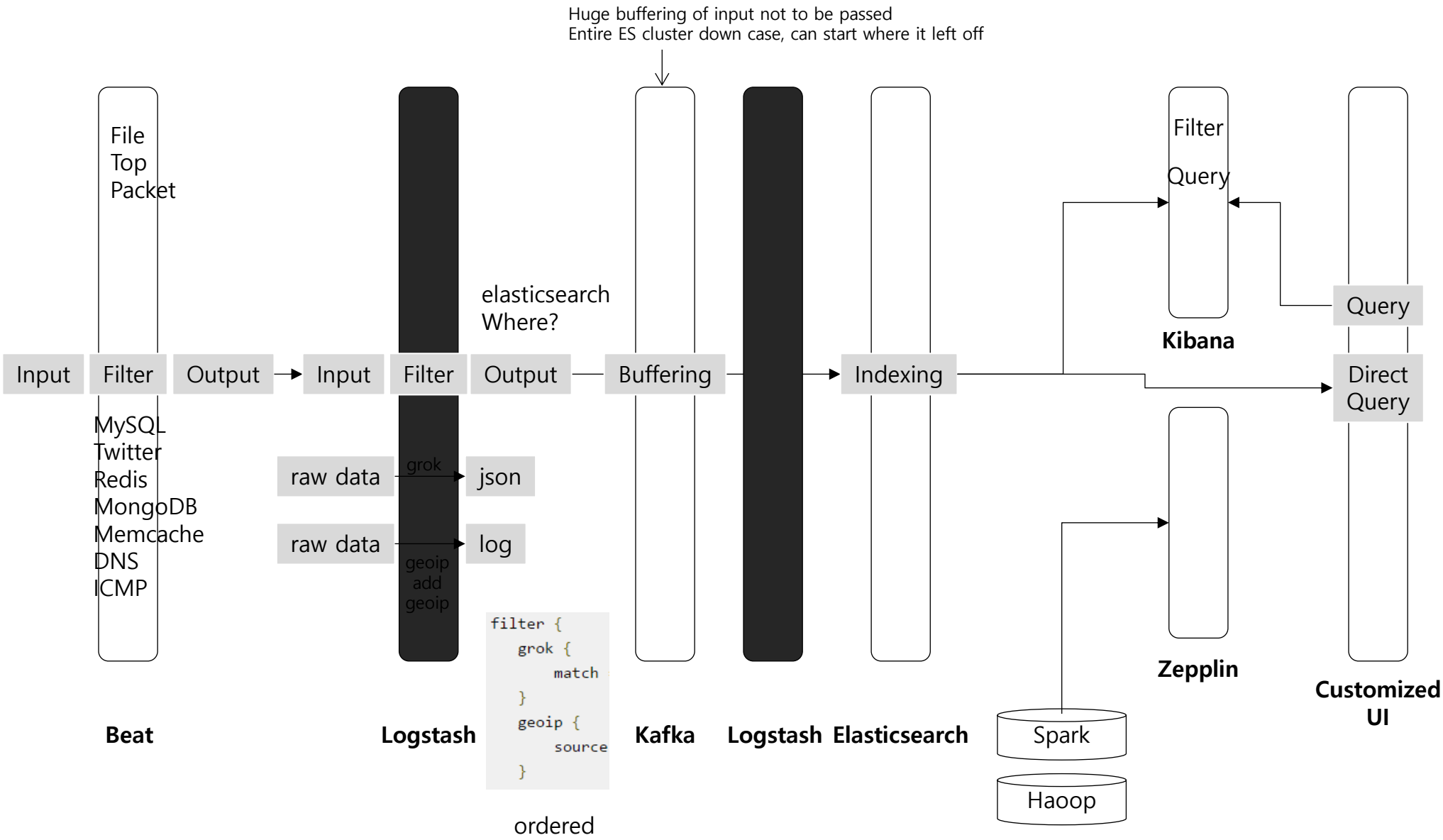
Data Pipeline(II)



Data Pipeline(III)

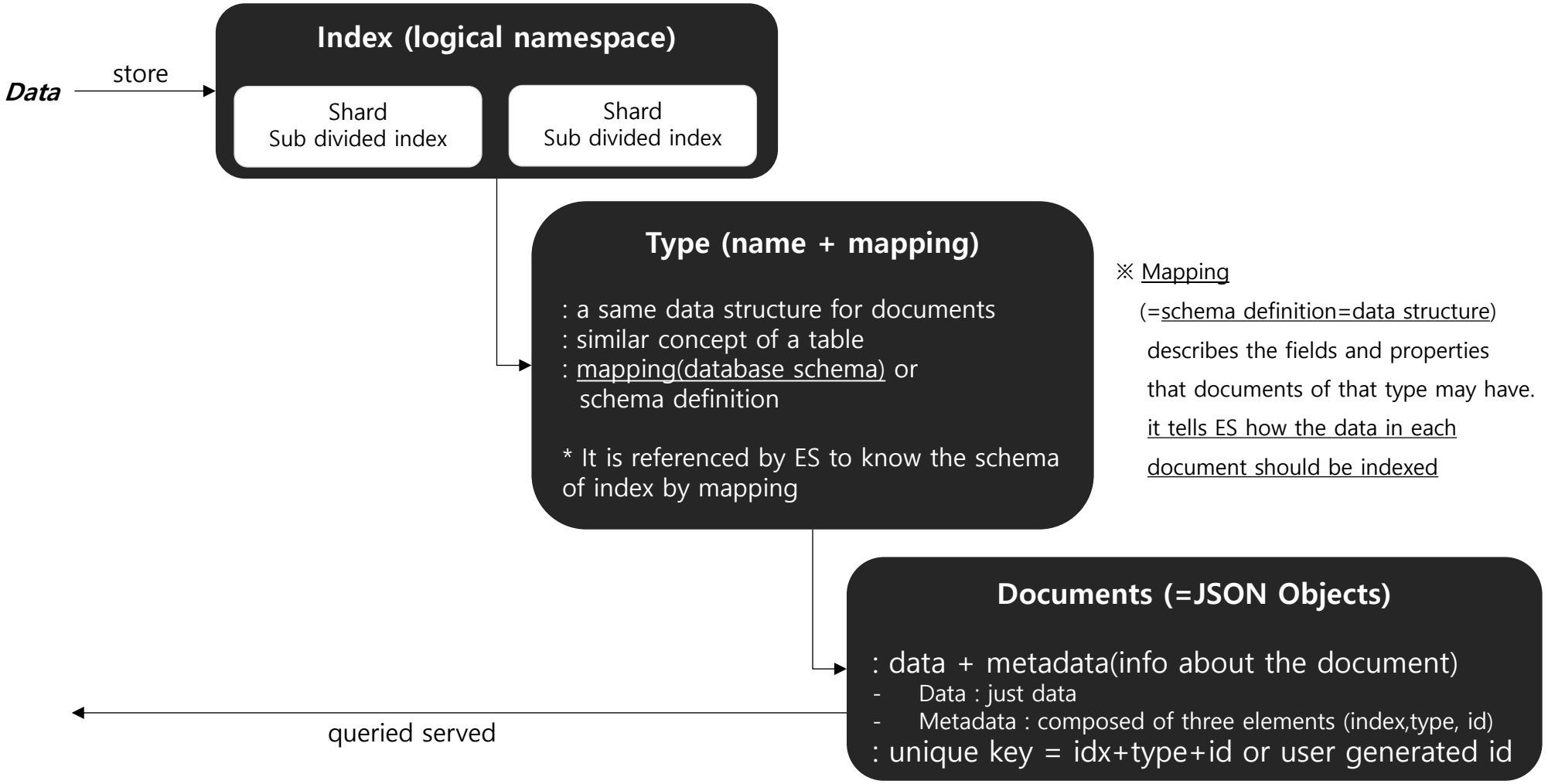


Data Pipeline(IV)



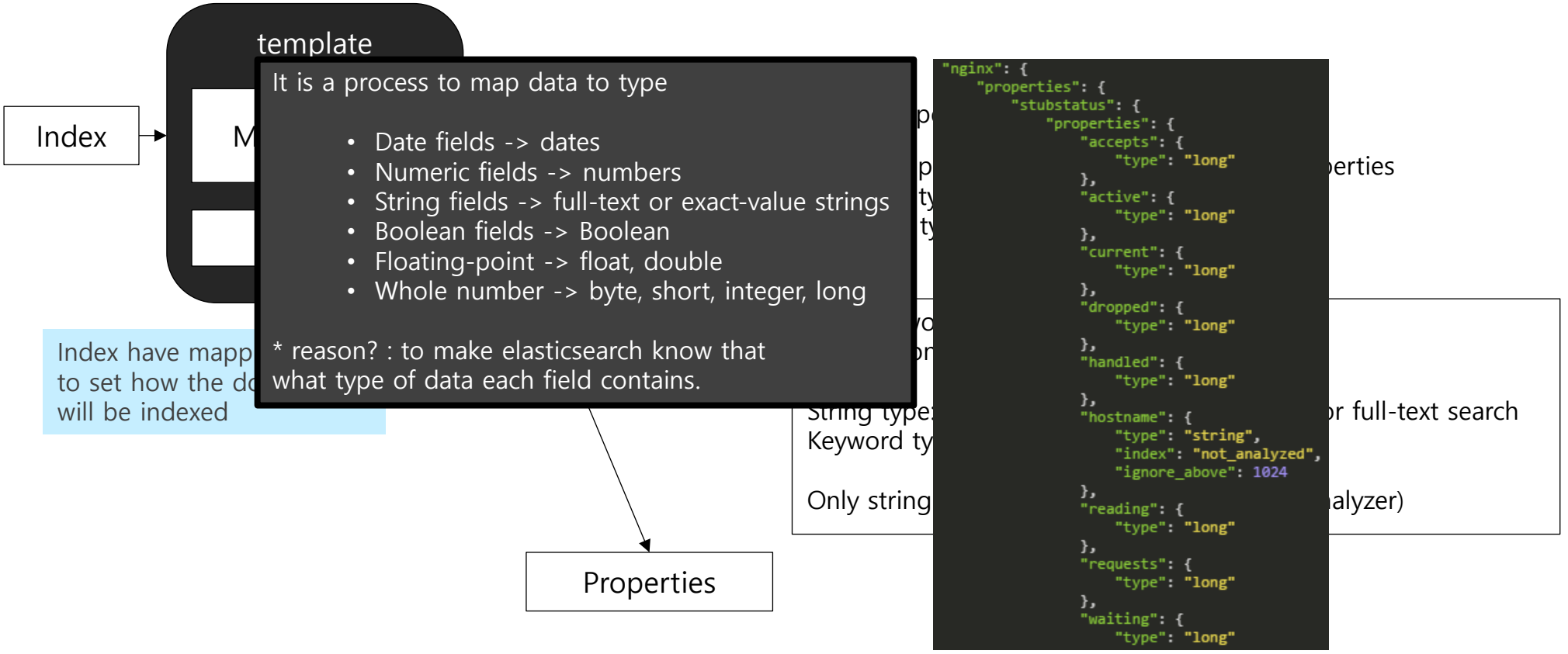
Index to Document?

(The Structure)



Mapping

Is the process of defining how a document, and the field contains, are stored and indexed
- Which string fields should be treated as full text fields
- Which fields contain numbers, dates, geolocation
- Custom rules to control the mapping for dynamically added fields



Mapping *(type)*

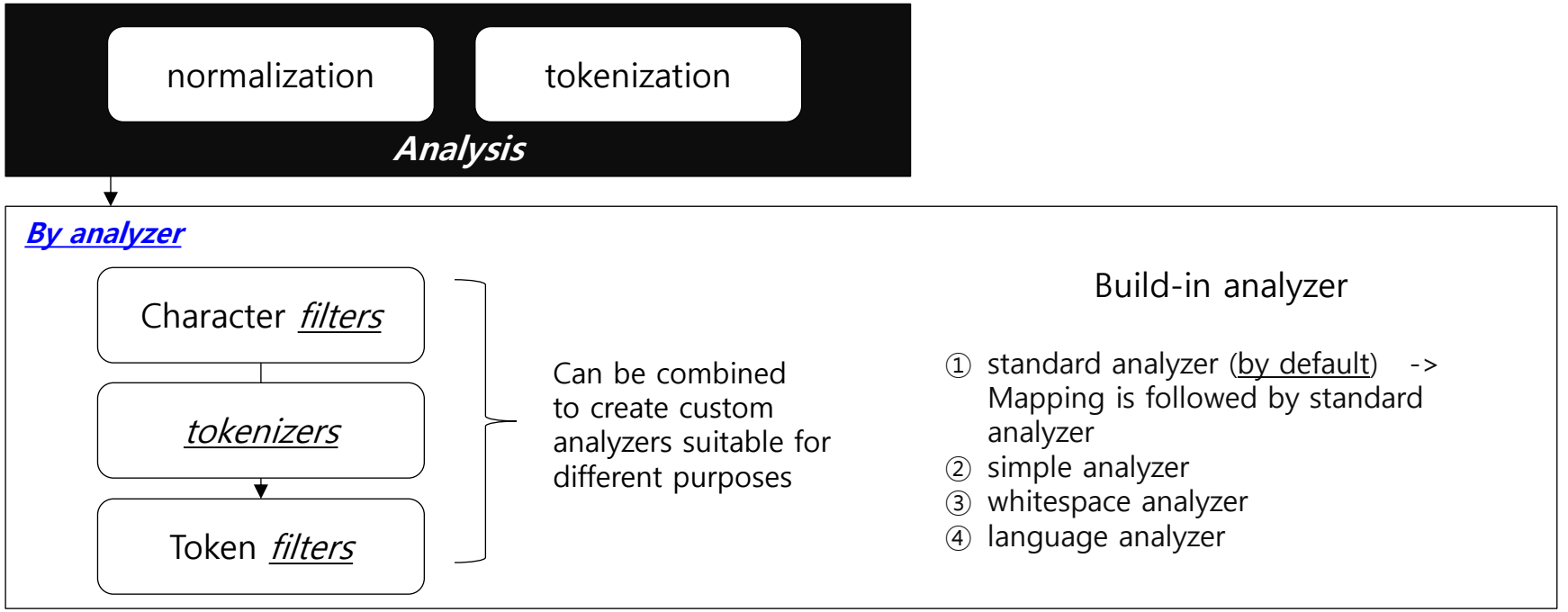
Field	type	description	analyzer
string	text	Full-text search (inverted)	Standard, English, French
	keyword	Sorting or Aggregation (not inverted)	
	date		
	long		
	double		
	boolean		
	ip		
	json(object, nested)		
	geo_point , geo_shape		

What is Analysis?

Combined Processes (***Normalization*** + ***Tokenization***)

※ Purpose: is **1)to make a highly efficient inverted index 2) for making a query precisely**

What is normalization? Lowercased word, a word be stemmed, synonyms of word
What is tokenization?

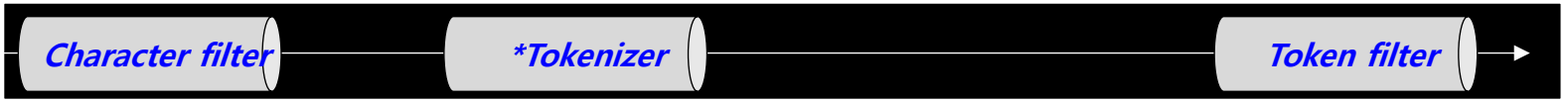


What is Analysis?
(Analyzer)

✓ Analyzer (= is a wrapper that combines three functions into a single package)
- ✓ Character filters :
- ✓ Tokenizers: whenever it encounters whitespace, punctuation
- ✓ token filters: change terms(lowercasing) / add terms / remove terms

✓ Standard analyzer (for full-text fields , it's a good choice for western languages)
- ✓ Standard tokenizer
- ✓ Standard token filter
- ✓ Lowercase toke filter
- ✓ Stop token filter (for Korean?...we need to see to support multi language)

```
PUT /spanish_docs
{
    "settings": {
        "analysis": {
            "analyzer": {
                "es_std": {
```



It used to tidy up a string before it has tokenized
- [html_strip character filter] to remove all HTML tags

Tokenizer breaks up the string into individual terms or tokens

Standard	Remove most punctuation
Keyword	the same string as it received
Whitespace	splits text on [whitespace]
pattern	split text on a matching [regular expression]
Edge ngram,letter,lowercase,ngram,thai,etc	

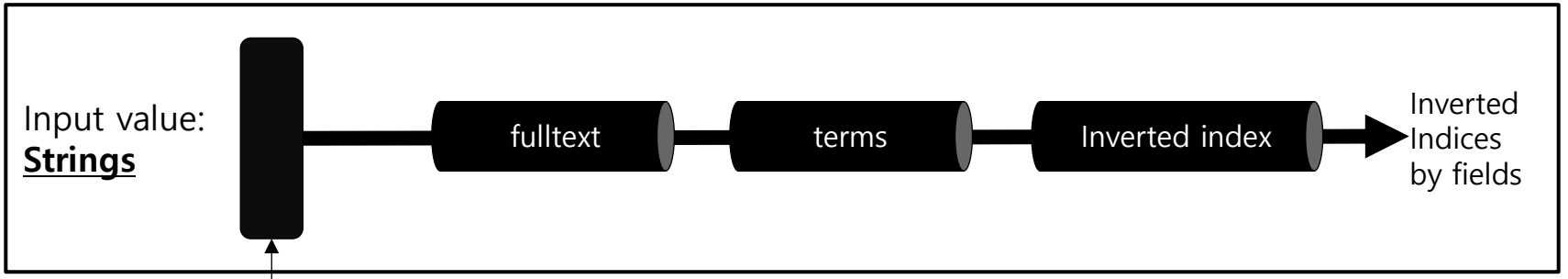
Change, add, remove tokens
- Lowercase
- Stop token filters
- Stemming token filters
- Asicc folding filter
- Ngram
- Edge_ngram

✓ Other Analyzers
- ✓ Standard analyzer (by default) -> Mapping is followed by standard analyzer
- ✓ Simple analyzer
- ✓ Whitespace analyzer
- ✓ Language analyzer

✓ Custom Analyzers

What is Analysis?

(example)



defined analyzer?
- Yes: it will be used
  or
- No: standard analyzer by default

Need to see which language
String, date, etc
How?

Mapping

In order to be able to treat date fields as dates
Numeric fields as numbers
String fields as full-text
Exact-value strings
So, ES needs to know what type of data each field contained
- Define the fields within a type
- Datatype for each field
- How the field should be handled by ES

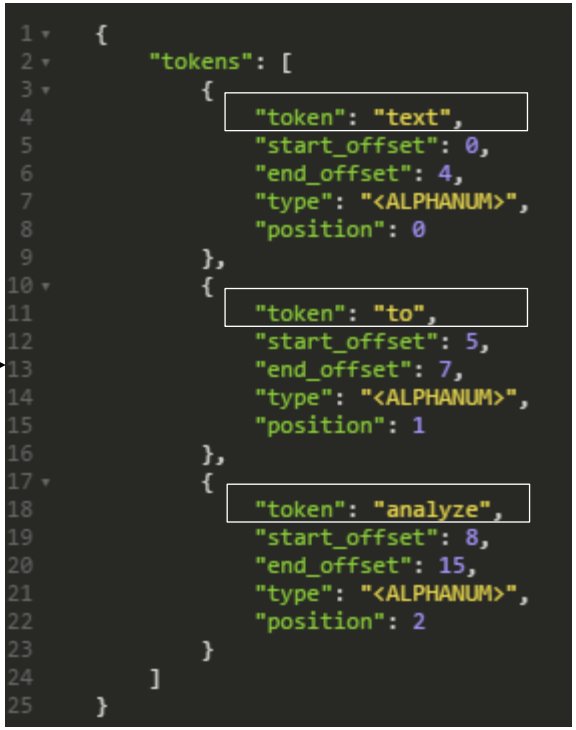
What is Analysis?

(output example by standard)

GET http://70.60.31.129:9201/_analyze Send

JSON Auth Query Headers (1)

```
{
    "analyzer": "standard",
    "text": "Text to analyze"
}
```



```
{
    "tokens": [
        {
            "token": "text",
            "start_offset": 0,
            "end_offset": 4,
            "type": "<ALPHANUM>",
            "position": 0
        },
        {
            "token": "to",
            "start_offset": 5,
            "end_offset": 7,
            "type": "<ALPHANUM>",
            "position": 1
        },
        {
            "token": "analyze",
            "start_offset": 8,
            "end_offset": 15,
            "type": "<ALPHANUM>",
            "position": 2
        }
    ]
}
```

To better understand what's going on

Can use analyze API to see how text is analyzed

"standard" is a default analyzer
Automatically configures it as full-text string

↓

Like to interpolate a different type of process with not standard analyzer

↓

It is a "Mapping" <- click

How to use analyzer?

① Create Index with Settings
② Use analyze API for checking that it works correctly
③ Need to tell Elasticsearch where to use it
Apply it to a string field with a mapping like in box #3

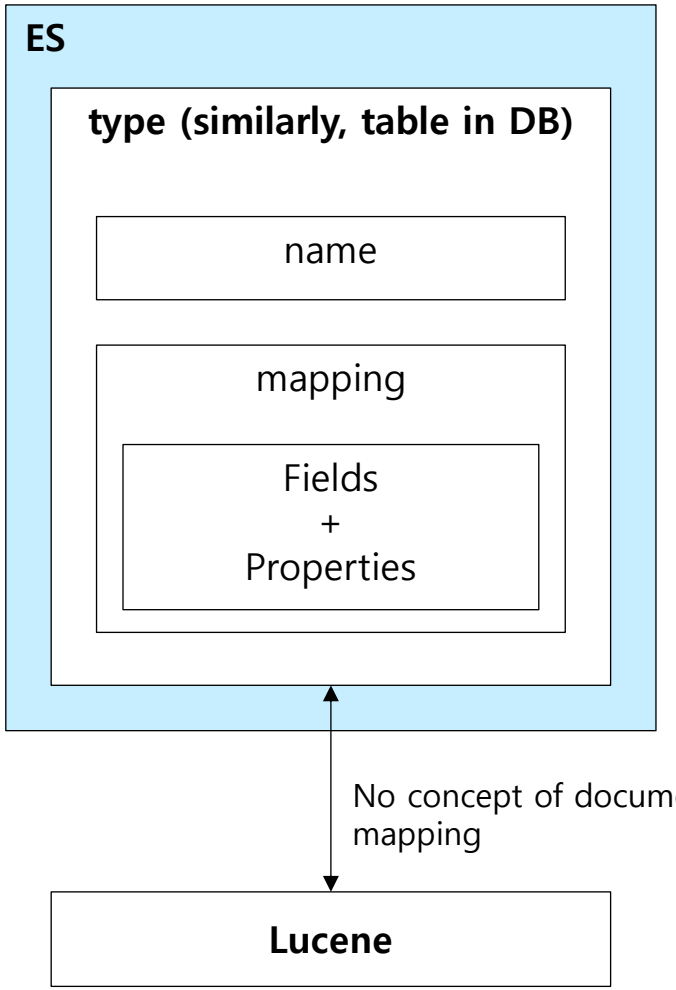

```
PUT /my_index                                          <format>
{
    "settings": {
        "analysis": {
            "char_filter": { ... custom character filters ... },
            "tokenizer":   { ...    custom tokenizers     ... },
            "filter":      { ...   custom token filters   ... },
            "analyzer":    { ...    custom analyzers      ... }
        }
    }
}
```

② GET /my_index/_analyze?analyzer=my_analyzer
{
The quick & brown fox
}

③ PUT /my_index/_mapping/my_type
{
"properties": {
"title": {
"type": "string",
"analyzer": "my_analyzer"
}
}
}

PUT /my_index ①
{
"settings": {
"analysis": {
"char_filter": {
"&_to_and": {
"type": "mapping",
"mappings": ["&=> and "]
}
},
"filter": {
"my_stopwords": {
"type": "stop",
"stopwords": ["the", "a"]
},
"synonym": {
"type": "synonym",
"synonyms": [
"albert => albert, al",
"allan => allan, al"
]
}
},
"analyzer" {
"my_analyzer": {
"type": "custom",
"char_filter": ["html_strip", "&_to_and"],
"tokenizer": "standard",
"filter": ["lowercase", "my_stopwords"]
}
}
}
}

How to use analyzer
(types and mappings)




```
PUT /my_index/_mapping/blogpost
{
  "properties": {
    "user": {
      "properties": {
        "name": { ❶
          "type": "string",
          "fields": {
            "raw": { ❷
              "type":  "string",
              "index": "not_analyzed"
            }
          }
        }
      }
    }
  }
}
```

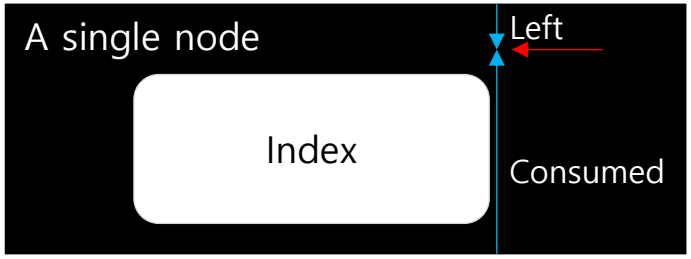
❶ ← Full-text search

❷ ← For grouping

When shards?

shards & replicas

An index can potentially store a large amount of data that can exceed the hardware limits of a single node. For example, a single index of a billion documents taking up 1TB of disk space may not fit on the disk of a single node or may be too slow to serve search requests from a single node alone



It's time to shard
& make replicas

What it means
that Index grows?

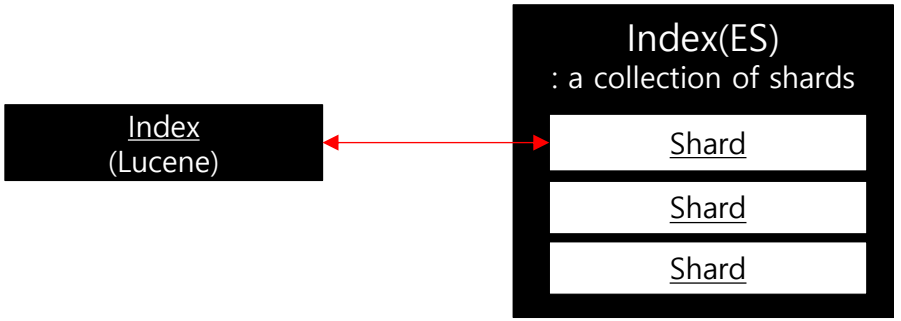
Can not afford to allocate a free space for storing data
More request as much as the size of data

Merits

- High Availability (Redundancy)
- Scale out search volume
- Throughput on all replicas in parallel

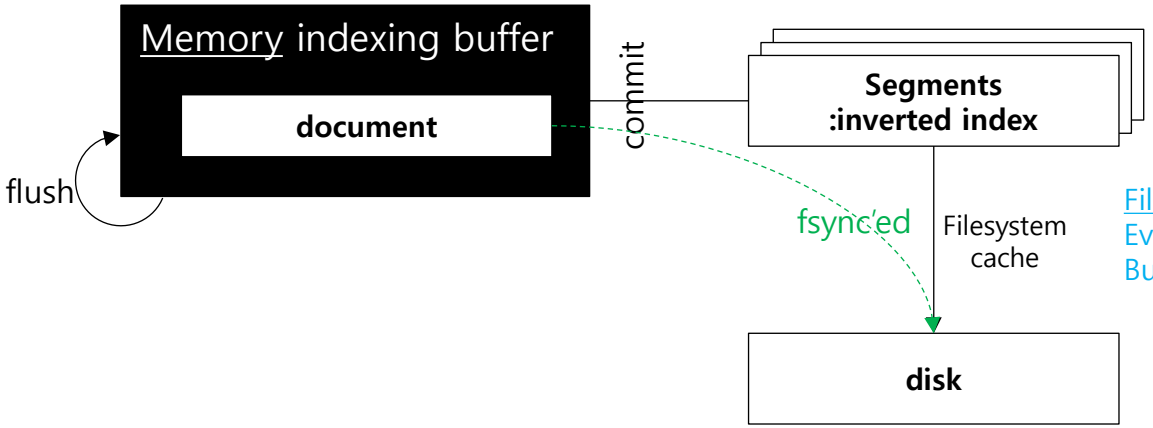
※ each index capacity
5 primary shards and 1 replica (10 shards per index)

What is sharding?

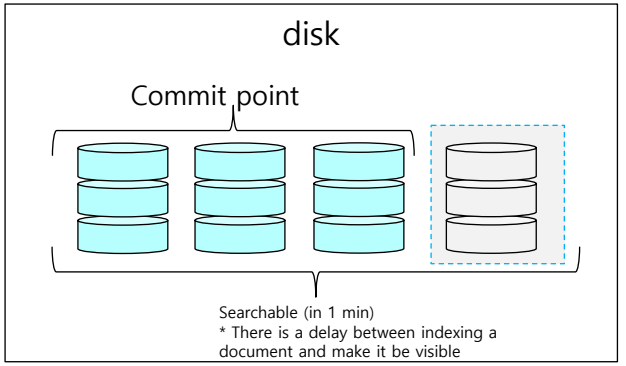


Every shard is refreshed automatically once every second
But, also can refresh with refresh API

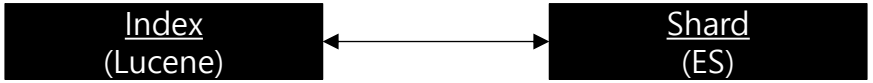
- POST /_refresh
- POST /blogs/_refresh



Filesystem cache (in 1 sec)
Eventually, it will be fsynced.
But, it is a searchable even not synced to disk though



What is sharding?



Refresh

Not all use cases require a refresh every second.
Able to set an interval for refresh

```
PUT /my_logs
{
        "settings": {
                "refresh_interval": "30s"
        }
}
```

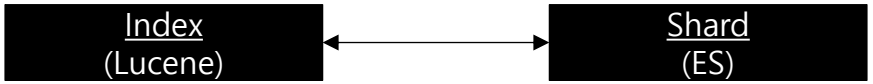
Can turn off for a big new index and on to enable it

POST /my_logs/_settings { "refresh_interval": -1 } → diable automatic refreshes (2m(min), 2(millisec))
POST /my_ogs/_settings { "refresh_inverval": "1s" } → refresh automatically every second

Flush (automatically, every 30 minutes)

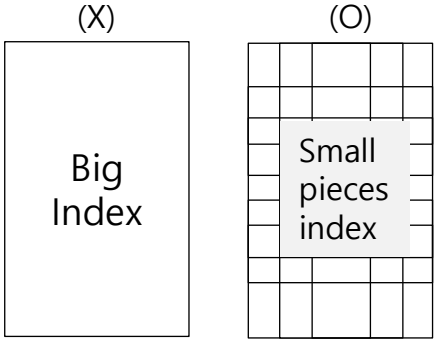
POST /blogs/_flush
POST /_flush?wait_for_ongoing

What is sharding?



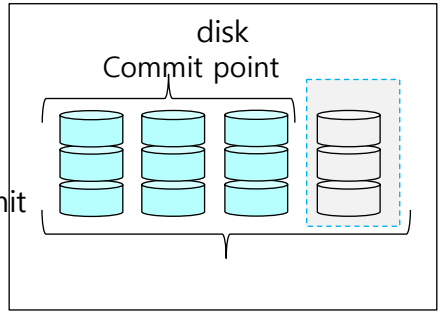
For the performance we can use filesystem cache before the index be fsync'ed.
But, where the server reach failure status before commit...we don't want to lose those either (real time search and data in filesystem cache)

What is the safety belt to secure those situation?
It is a translog or tansaction log, which records every operation in elasticsearch as it happens.

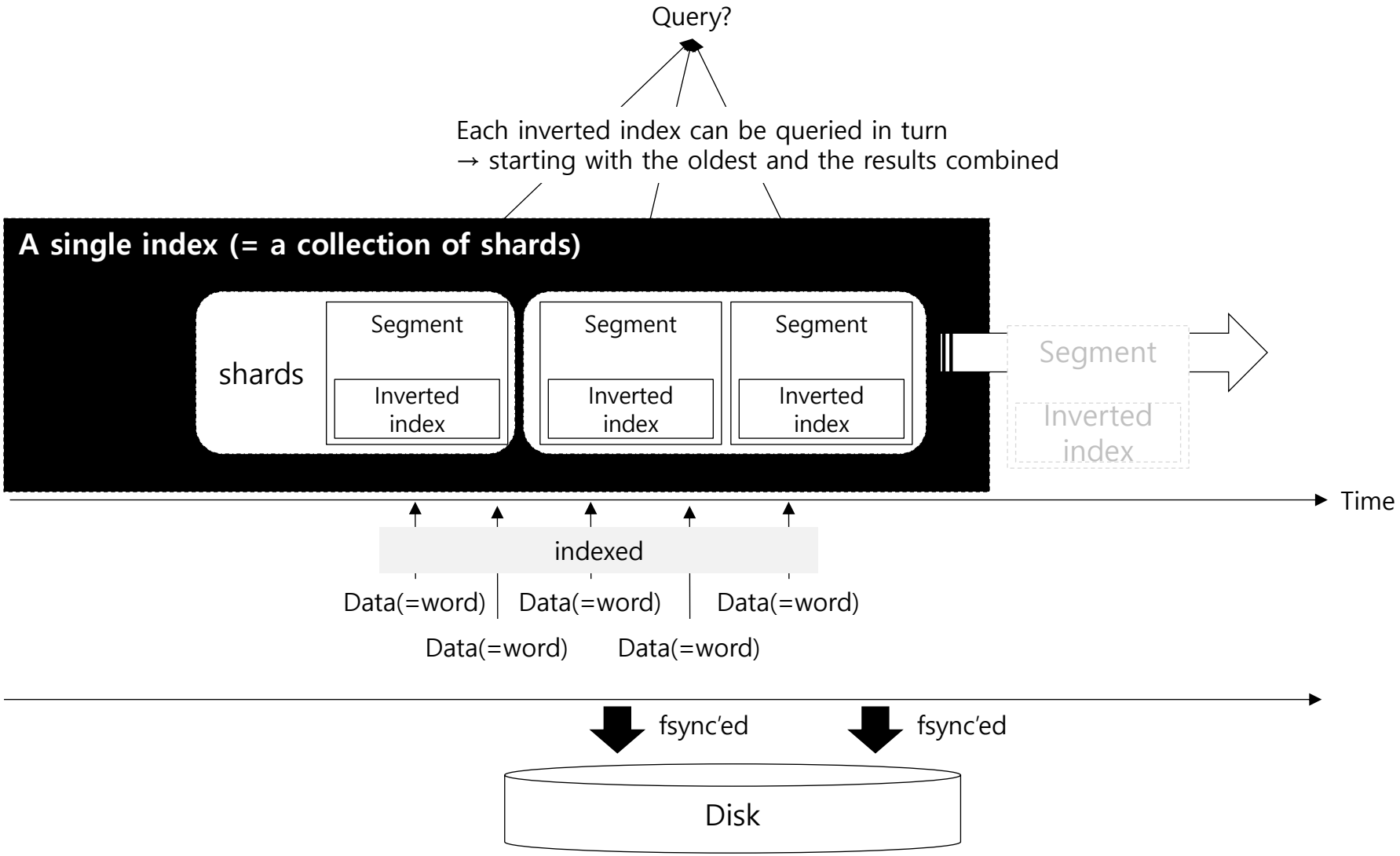


Index updatable?
Index is an immutable
Use more than one index
Dynamically updatable indices

If translog is getting to big?
- The index is flushed
- A new translog is created
- A full commit is performed
  and point is written to disk
- The filesystem cache is flushed with an fsync
- The old translog is deleted
- Redo a translog the time after the last commit



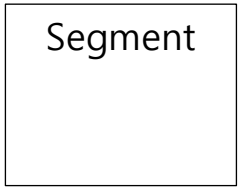
What is sharding?



**** Lucene introduced the concept of per-segment search***

What is sharding? (index management)

Deletes and Updates



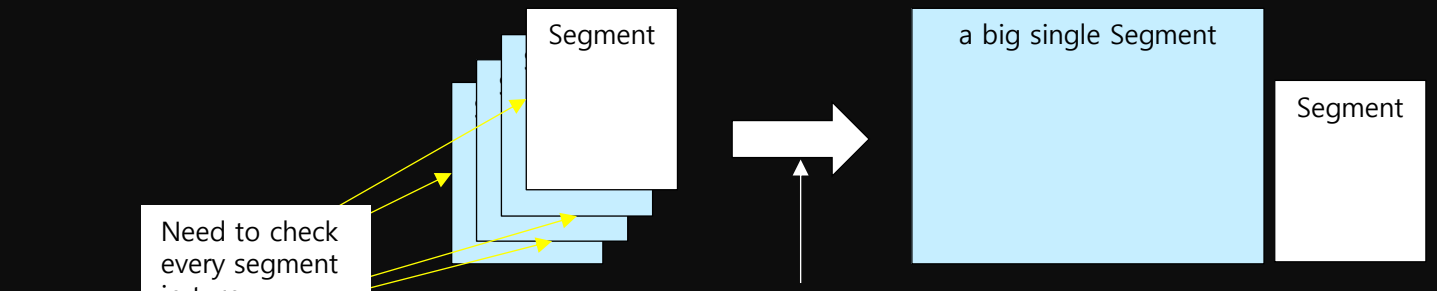
Segment is immutable, which is cannot be removed from old segments
no can older segments be updated to reflect a newer version of a document
Ever commit point includes a **.del** file that list which documents in which segments have been deleted.

When a document is "deleted", it is actually just marged as deleted in the .del file

A document that has been marked as deleted can still match a query,
But, it is removed from the results list before the final query results are returned.

What is sharding? (segment managing)

It's a backgroud process and happens automatically while indexing and searching

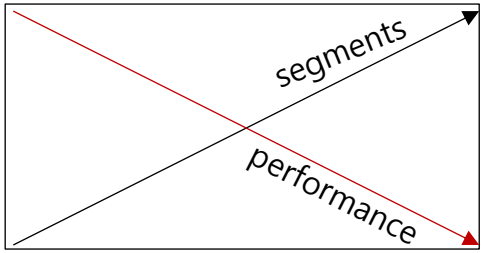


Old all *.del will be deleted and does not copied over to the new bigger segment.
And those files are purged from the filesystem.
It happens automatically while indexing and searching
It acts like a process compaction in Cassandra

Tunning is required

The more segments there are, the slower the search will be. (performance)
Each segment consumes file handles, memory, and CPU cycles.

So,
ES solves this problem by merging segments in the background.
Small segments are merged into bigger segments.



like a compaction of cassandra

Index management

1. Creating an index

Need to set to customize index

Settings	mappings

In order not to create index automatically…:
→ action.auto_create_index: false conf/elasticsearch.yml

2. Deleting an index

DELETE /my_index
DELETE /index_one,index_two or /index_*
DELETE /_all

*** Index Settings**

number_of_shards (number of primary shards-can't be changed)
number_of_replicas (number of replica shards-can be changed)

3. Configuring Analyzers

Analyzer

4. Re indexing

Index management

1. Query

PUT /website <- create a index named "website"

PUT /website/blog/123 { "A": "va1", "B": "val2"} <- id made by owner (version: 1, created: TRUE)
PUT /website/blog/123 { "A": "val3, "B": "val4"} <- will be replaced with a new version (version: 2, created: False)

POST /website/blog/ {" ": " ", " ": " " } <- will be made by system

GET /website/blog/123?_source=title,text <- can define only two fileds among many
GET /website/blog/123/_source <- can query to receive except for metadata
GET /website/blog/123?pretty <- can receive with the format "JSON"

HEAD /website/blog/123 <- just check whether the document exists (return 200 or 404)

Update API (can be used to make partial update to a document)

Pipeline(logstash)

Three stages for Logstash event processing

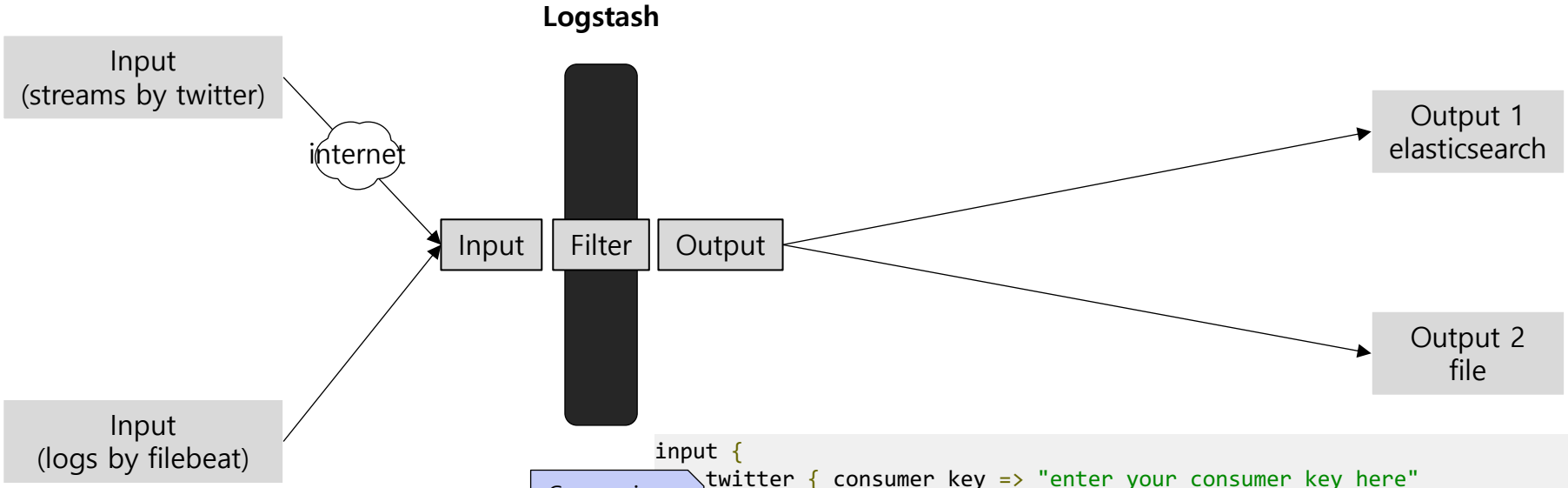
- inputs → capture stream, turn it into events (generate events)
  https://www.elastic.co/guide/en/logstash/5.0/input-plugins.html
- filters → process events (modify them)
  https://www.elastic.co/guide/en/logstash/5.0/filter-plugins.html#filter-plugins
- outputs → send events to ship to destination
- codecs → input or output filter (i.g, json, multiline)

grok	Supports 120 patterns
mutate	Transformation (Rename, remove, replace, modify)
drop	Drop debug events
clone	Make a copy of an event
geoip	Add information about geographical location of IP addresses (kibana)

PQ(persistent queue), DLQ(dead letter queue)

Pipeline(logstash)

(stitching together multiple input and output)




```
filebeat.prospectors:
- input_type: log
  paths:
        - /var/log/*.log
  fields:
        type:syslog
  output.logstash:
    hosts: ["localhost:5043"]
```

Consuming
-twitter
-beat

```
input {
    twitter { consumer_key => "enter_your_consumer_key_here"
        consumer_secret => "enter_your_secret_here"
        keywords => ["cloud"]
        oauth_token => "enter_your_access_token_here"
        oauth_token_secret => "enter_your_access_token_secret_here"
    }
    beats { port => "5043" }
}
output {
    elasticsearch {
        hosts => ["IP Address 1:port1", "IP Address 2:port2"]
    }
    file {
        path => "/path/to/target/file"
    }
}
```

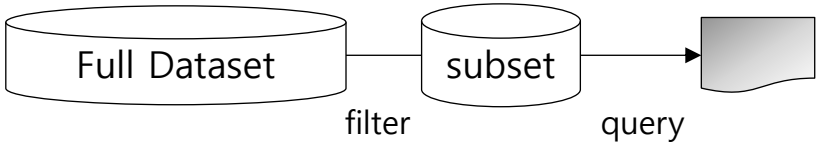
Indexing
-elasticsearch
-file

Query & Filter

What differences Query .vs Filter?

Queries	Filters
relevance	boolean yes/no (=binary question)
Full text (unstructured)	Exact values (structured)
Not cached	Cached
Slower	Faster (dataset->filtered- >subset->query)
Filter first, then query remaining docs	

Filter (boolean, fast, cacheable)




```
GET /_search
{
    "query": {
        "filtered": {
            "query": {"match": { "title": "search"}},
            "filter": {"term": {"status": "active"}}
        }
    }
}
```

Kinds	Description	Objects	Examples
Term	Exact values	Dates, Booleans, not_analyzed exact-value string fields	{ "term": { "age": 26 }} { "term": { "date": "2014-09-01" }}

Filter (Boolean type)

```
"bool": {
    "must":  [<filters>],       AND
    "should":   [filters],      OR
    "must_not":  [<filters>]   NOT
}
```

```
{
    "filtered": {
        "query": {"match": { "title": "full text search" }},
        "filter"; {
            "bool": {
                "must":  {"range": {"created": {"gte": "now-1d/d"}}},
                "should": [
                    {"term": {"featured": true}},
                    {"term": {"starred": true}}
                ],
                "must_not": {"term": { "deleted": false}}
    }
}
```

Analytics

Aggregations dsl (domain specific language)
Differen types of aggregations

Buckets	Metrics
Terms	Stats
Date Historgram	Percentile
Filter	Cardinality
Range	Top hits
Nested	Scripted
Children	Max | Min | Avg
	Simple mathematical operations

Agg = Buckets + Calculated Metric

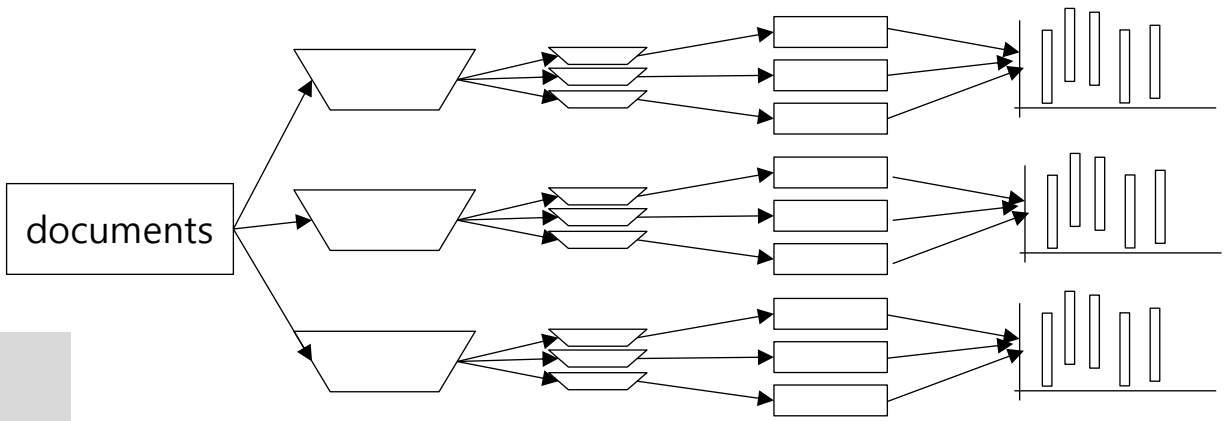
Buckets : Collection of documents that meet a certain criteria

Metrics : Statistics calculated on the documents in a bucket

Example>

Select count(color) ← metric
From table
Group By color ← bucket

Combining the two is an aggregation



Shard Partition Per Buckets | Sub Buckets | Metric(calc) Reduce | Results

Designed for speed and scale
Aggregates are computed in-memory
Aggregation can be composed
Near real-time response times
Trades accuracy for speed in some use cases
Single network round-trip
Single pass through the data on shards
Some aggregations are very expensive

Parse → Collect & Combine → Reduce

Modeling Your Data

How to fill up the gap between FlatWorld and RealWorld(RDB)

- Application-side joins
- Data denormalization
- *Nested objects*: *all entities live within the same document*
- Parent/child relationships: completely separate documents

Beats

Examples of operational data

Wired data

Packetbeat

Captures insights from
Network packets

-Copy traffic at OS or hardware level
-Is completely passive
-ZERO latency overhead
-Not in the request/response path,
Cannot break your application

System stats

Topbeat

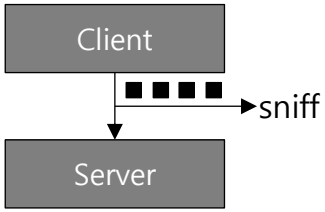
Like the Unix top command but
Sends the output periodically to
Elasticsearch. Also works on windows

System wide (system load,
cpu(core),memory
Per process: state name command
line pid
Disk usage: used , free, mounted
points

Logs

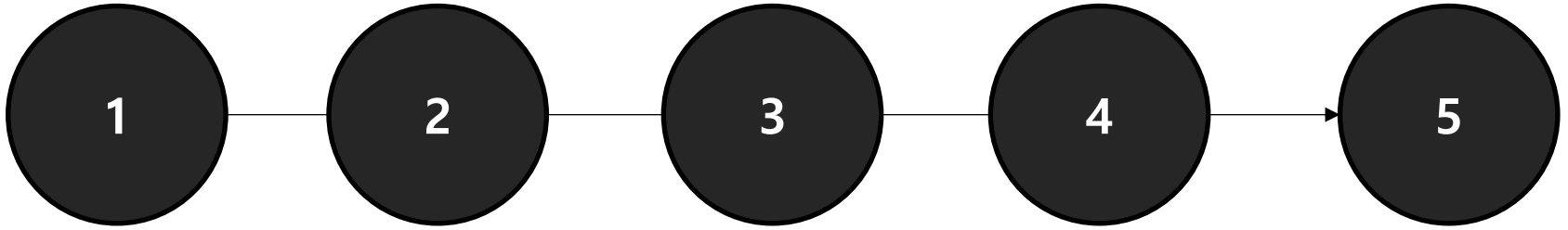
Filebeat
Winlogbeat

Like Forwards log lines to
Elasticsearch



Use cases
Security (Intrusion Detection)
Troubleshotting network issues
Troubleshotting applications
Performance analysis

Beats-packetbeat



Capture network traffic

Decodes network traffic

Corelates request With response in transactions

Extract Measurements like Response time, status

Group meta info Json objects to send to elasticsearch

It does all of these in real-time directly on the target servers

HTTP	Thrift-RPC	DNS	(+) addable
MySQL	Memcache	AMQP	
PostgreSQL	MongoDB	Redis	ICMP

Beats-packetbeat

Network devices	Packetbeat.interfaces.device: any * Select the network interface to sniff the data Any means all connected interfaces
Transaction protocols	Packetbeat.protocols.icmp [amqp, dns,http,memcache,mysql,pgsql,redis,mongodb,nfs] Port and types to listen
Output	ES output LS output

```
# Select the network interfaces to sniff the dat
# keyword to sniff on all connected interfaces.
interfaces:
  # On which device to sniff
  device: any

  # The maximum capture size of a single packet.
  snaplen: 1514

  # The type of the sniffer to use
  type: af_packet

  # The size of the sniffing buffer
  buffer_size_mb: 100
```

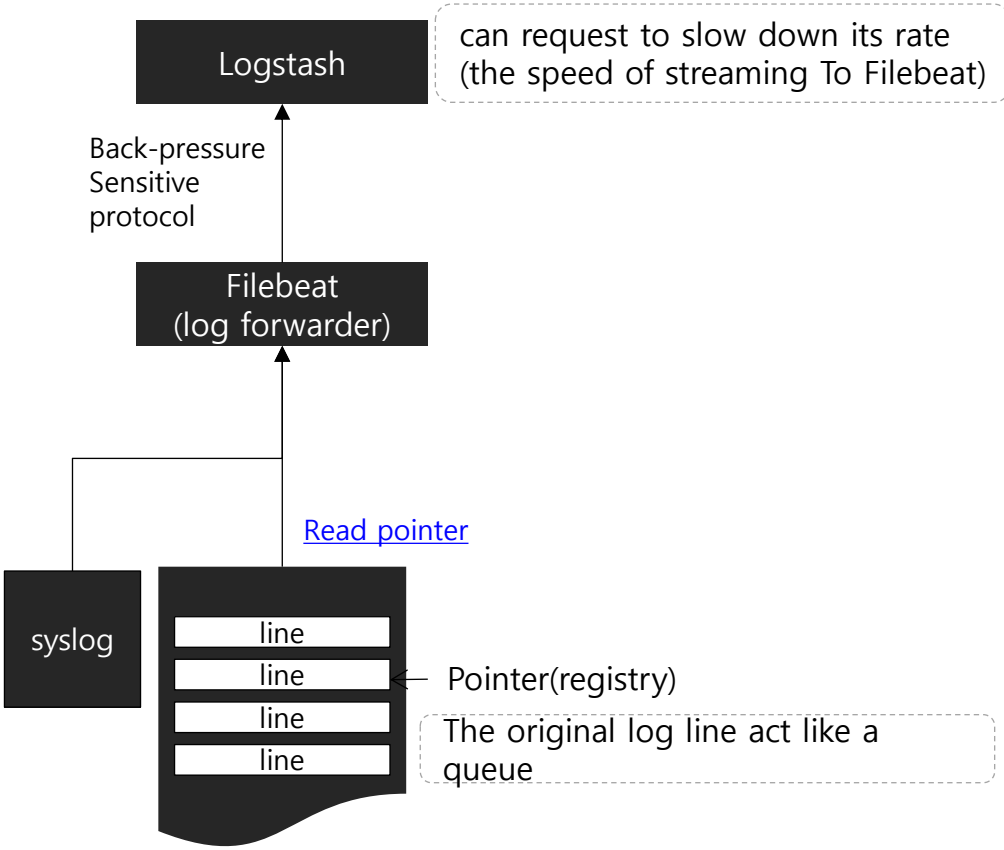
```
interfaces:
  device: eth0
```

```
protocols:
  http:

    # Configure the ports where to listen for HTTP traff
    # the http protocol by commenting the list of ports.
    ports: [80, 8080, 8000, 5000, 8002]
    send_all_headers: true
    send_response: true
    include_body_for: ["text/html"]
```

```
protocols:
  http:

    # Conf
    # the
    ports:
    send_a
    send_r
    includ

    # Unco
    # to H
    # The
    # This
    # sens
    # Only
    # hide

  mysql:

    # Conf
    # MySQ
    ports:

  pgsql:
```

Beats-filebeat

Never lose a log line



- Filebeat sends out unparsed log lines
- Use filters from Logstash to parse the log lines
- Flexible, with conditionals & custom filters
- Forward data to other systems using the Logstash output plugins
- Multiline features

Who parse it?

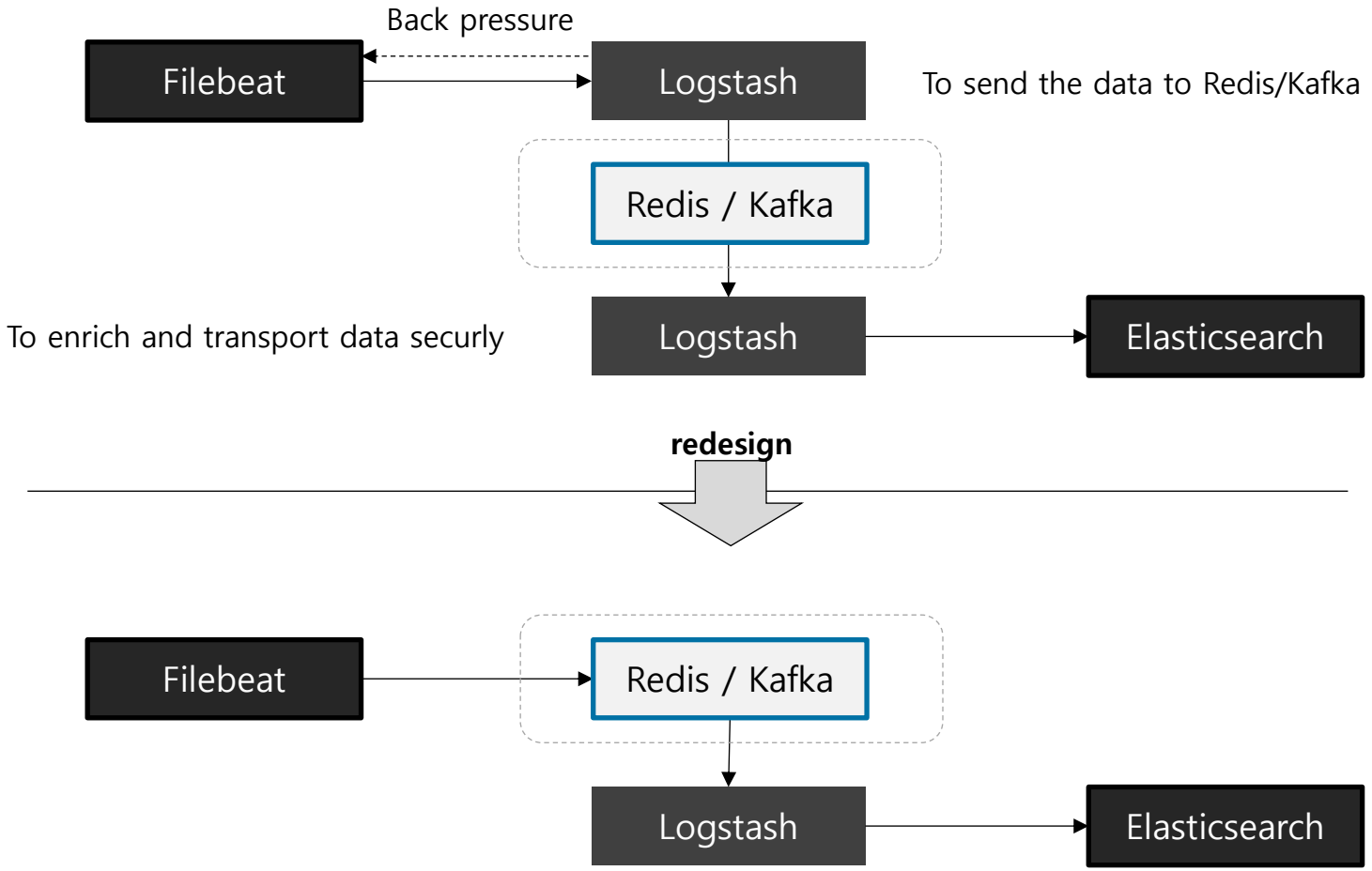
Two options:

1. via Logstash (filtering grok, mutate, etc(combine)) or
   Write my filter in Ruby
2. Ingest Node (= not through Logstash) ,Upcoming in 5.0

* It supports multiline process

Beats-filebeat

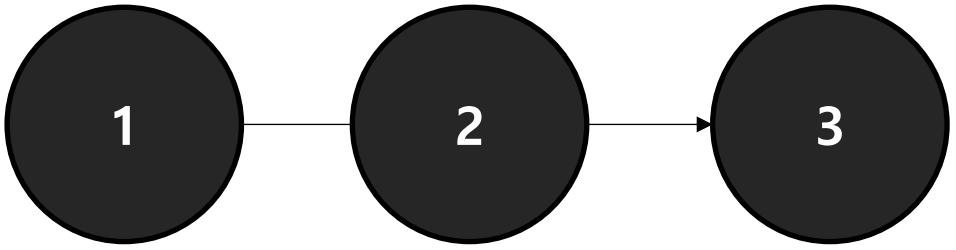
Never lose a log line



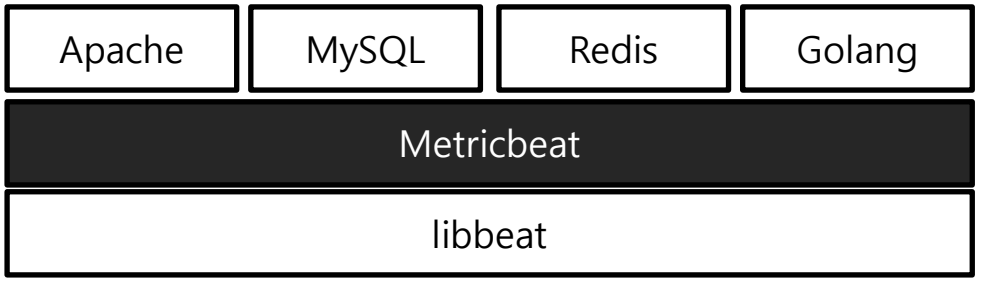
Module
-Apache2
-Auditd
-icinga
-Kafka
-Logstash
-MySQL
-Nginx
-Osquery
-PostgreSQL
-Redis
-System
-Traefik

Beats-metricbeat

To provide common infrastructure for all “metrics” related Beats
A module for each monitored system



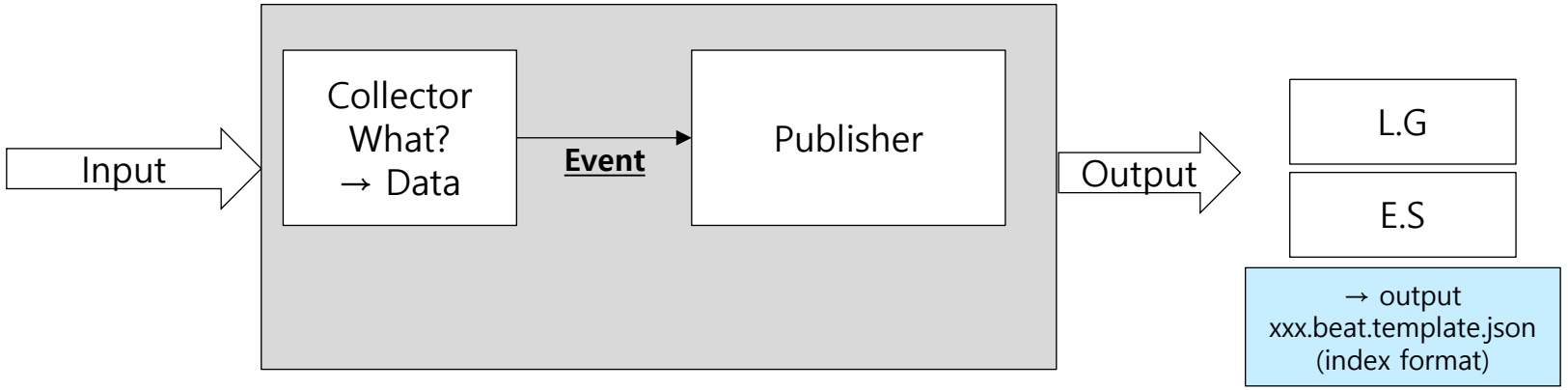
Same code same functionality => Matricbeats comes out



Beats-topbeat

1. Configuration management
2. Logging
3. Demonizing
4. Windows service handling
5. Data processing module(future) filtering/sampling

Implemented in "libbeat"



※ each beat use template for output
: /Topbeat options/General options/Filters(rule)/Output
- topbeat.template.json
- Packetbeat.template.json

※ be able to define many option at filters
in xxxbeat.yml file
include_filelds:
drop_fileds:

```
#filters:
#- include_fields:
#    fields: ["cpu"]
#- drop_fields:
#    fields: ["cpu.user", "cpu.system"]
#
# The following example drops the events that have the HTTP response code 200:
#
#filters:
#- drop_event:
#    equals:
#       http.code: 200
```

Drop event http.code=200

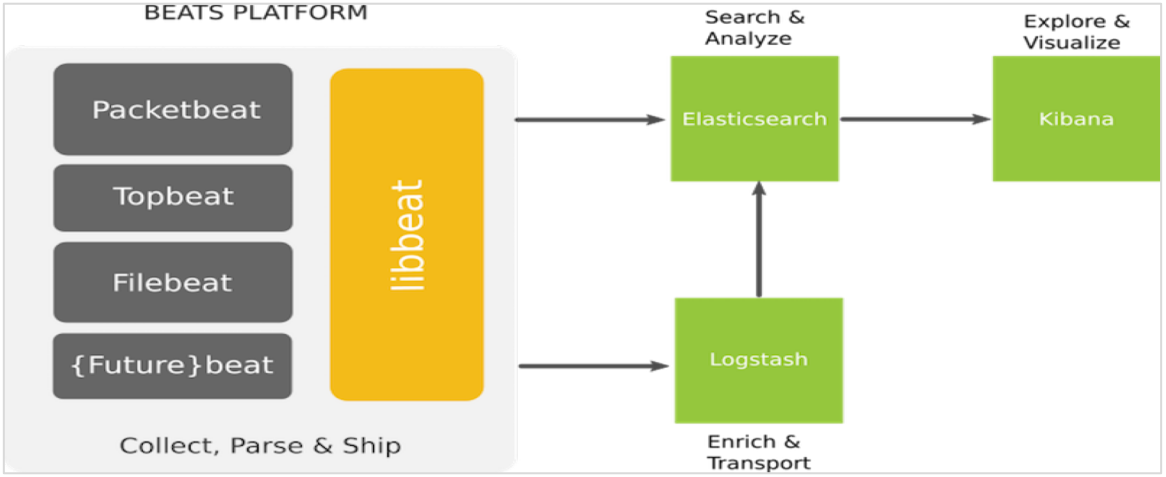
Beats-community beats

Apachebeat	Dockerbeat (per container)	Elasticbeat	Execbeat
Factbeat	Hsbeat	Httpbeat	Nagioscheckbeat
Nginxbeat	Phpfpmbeat	Pingbeat	Redisbeat
Unifiedbeat	Uwsgibeat		

Beats

Lightweight shippers for Elasticsearch & Logstash
Periodically ships system-wide and Per-process statistics from servers

Folder	Description	
libbeat	The Go framework for creating new Beats	Officially supported by ES
Topbeat	Like 'top' but inserting the data into Elasticsearch	
Packetbeat	Tap into your wire data	
Filebeat	Lightweight log forwarder to Logstash & Elasticsearch	
Winlogbeat	Sends Windows Event logs	
Install an agent on servers, which collecting of data is required		



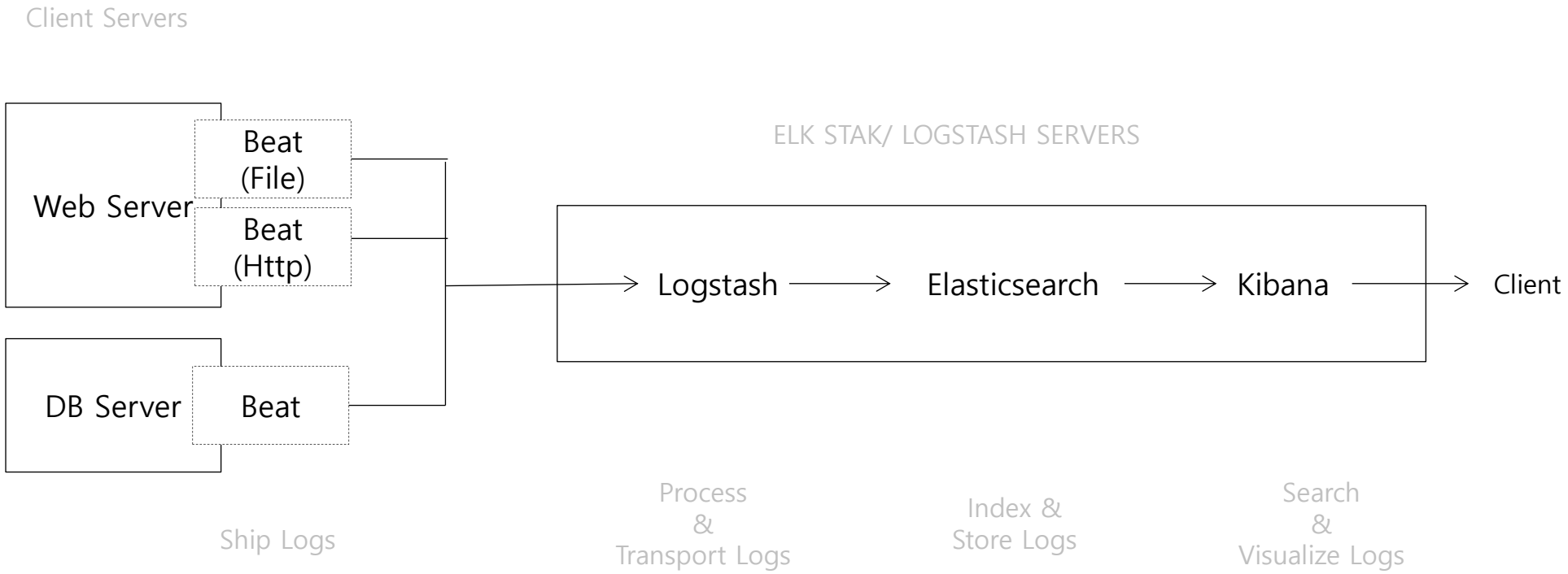
Ships log files from servers

Community Beats

apachebeat / dockerbeat /
jmxproxybeat /
imsensorbeat (linux sensor, cpu temp, fan speeds, voltages) /
mysqlbeat / nginxbeat /
redisbeat / twitterbeat /
udpbeat / pingbeat /
nagioscheckbeat / execbeat(shell cmds)

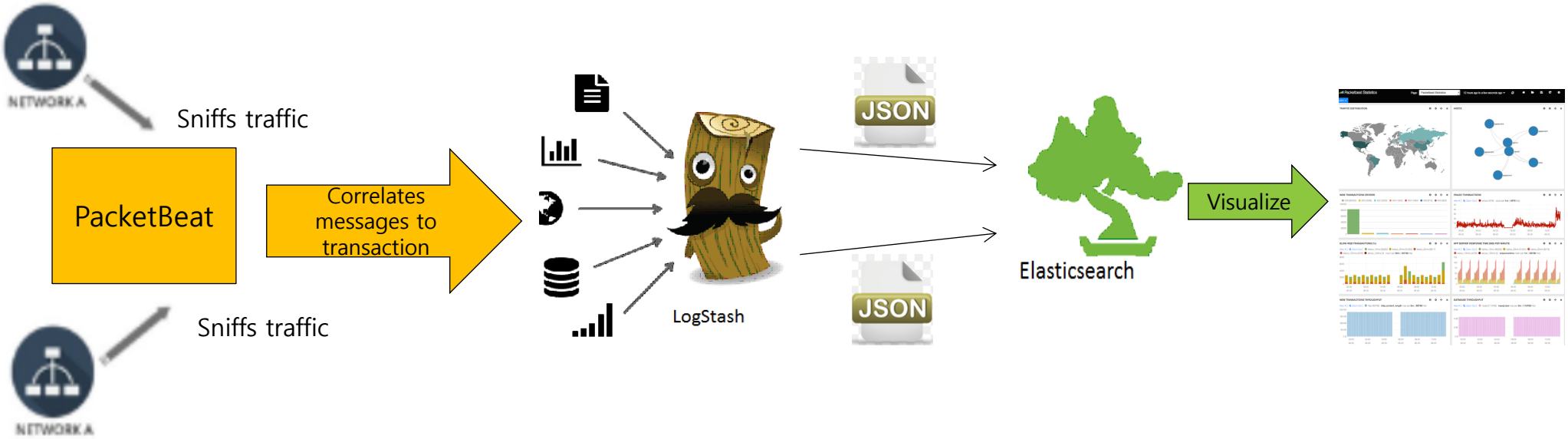
Beats

ELK stack setup has four main components:
Logstash: The server component of Logstash that processes incoming logs
Elasticsearch: Stores all of the logs
Kibana: Web interface for searching and visualizing logs, which will be proxied through Nginx
Beat: Installed on client servers that will send their logs to Logstash, Serves as a log shipping agent that utilizes *Packetbeat*, *Topbeat*, and *Filebeat* are a few examples of *Beats*



Beat - Packetbeat

- Sniffs the traffic between your servers, parses the application-level protocols on the fly, and correlates the messages into transactions.
- For each transaction, Packetbeat inserts a JSON document into Elasticsearch which can be further visualize in Kibana.



Beat - Packetbeat

- A real-time network packet analyzer that you can use with Elasticsearch to provide an *application monitoring and performance analytics system*.

Works by capturing the network traffic between your application servers, decoding the application layer protocols (HTTP, MySQL, Redis, and so on) correlating the requests with the responses, and recording the interesting *fields for each transaction.

- Packetbeat can help you easily notice issues with your back-end application, such as bugs or performance problems, and it makes troubleshooting them - and therefore fixing them - much faster.

- **Supported protocols from packetbeat**
  - ✓ **ICMP (v4 and v6)**
  - ✓ **DNS**
  - ✓ **HTTP**
  - ✓ **AMQP 0.9.1**
  - ✓ **Mysql**
  - ✓ **PostgreSQL**
  - ✓ **Redis**
  - ✓ **Thrift-RPC**
  - ✓ **MongoDB**
  - ✓ **Memcache**

- Refer Appendix

Beat - Packetbeat

- /etc/packetbeat/packetbeat.yml file is the main config file that keep the information about interface, protocols and ports for capturing the transactions. Also includes configuration to send data after capturing and processing.

- Two ways to deploy:
  - ✓ On dedicated servers, getting the traffic from mirror ports or tap devices. Less overhead on application servers, but requires dedicated networking gear
  - ✓ On existing application servers.

- Packetbeat has several options for traffic capturing:
  - **pcap**, which uses the libpcap library and works on most platforms, but it's not the fastest option.
  - **af_packet**, which uses memory mapped sniffing. This option is faster than libpcap and doesn't require a k ernel module, but it's Linux-specific. Optimal sniffing mode for both deployment options
  - **pf_ring**, which makes use of an ntop.org project. This setting provides the best sniffing speed, but it requi res a kernel module, and it's Linux-specific. Sniffing speed of Gigabits/s for dedicated servers for packetbeat.

Beat - Packetbeat

To configure Packetbeat, edit the /etc/packetbeat/packetbeat.yml

- Select the network interface from which to capture the traffic.

```
# Select the network interfaces to sniff the data. You can use the "any"
# keyword to sniff on all connected interfaces.
packetbeat.interfaces:
  device: any
```

- In the protocols section, configure the ports on which Packetbeat can find each protocol. If you use any non-standard ports, add them here.

```
packetbeat.protocols:
  dns:
    ports: [53]

    include_authorities: true
    include_additionals: true

  http:
    ports: [80, 8080, 8081, 5000, 8002]

  memcache:
    ports: [11211]

  mysql:
    ports: [3306]

  pgsql:
    ports: [5432]

  redis:
    ports: [6379]

  thrift:
    ports: [9090]

  mongodb:
    ports: [27017]
```

- Set the IP address and port where Packetbeat can find the Elasticsearch installation:

```
# Configure what outputs to use when sending the data collected by the beat.
# Multiple outputs may be used.
output:
  ### Elasticsearch as output
  elasticsearch:
    # Array of hosts to connect to.
     hosts: ["192.168.1.42:9200"]
```

Beat - Packetbeat

- Packetbeat is Go program, The current Go version used for development is Golang 1.6.2. Packetbeat use a variety of scripts based on python to generate configuration files and documentations.

- The way Packetbeat works is that both the kernel and the user space program map the same memory zone, and a simple circular buffer is organized in this memory zone. The kernel writes packets into the circular buffer, and the user space program reads from it. The poll system call is used for getting a notification for the first packet available, but the remaining available packets can be simply read via memory access.
  - Example:

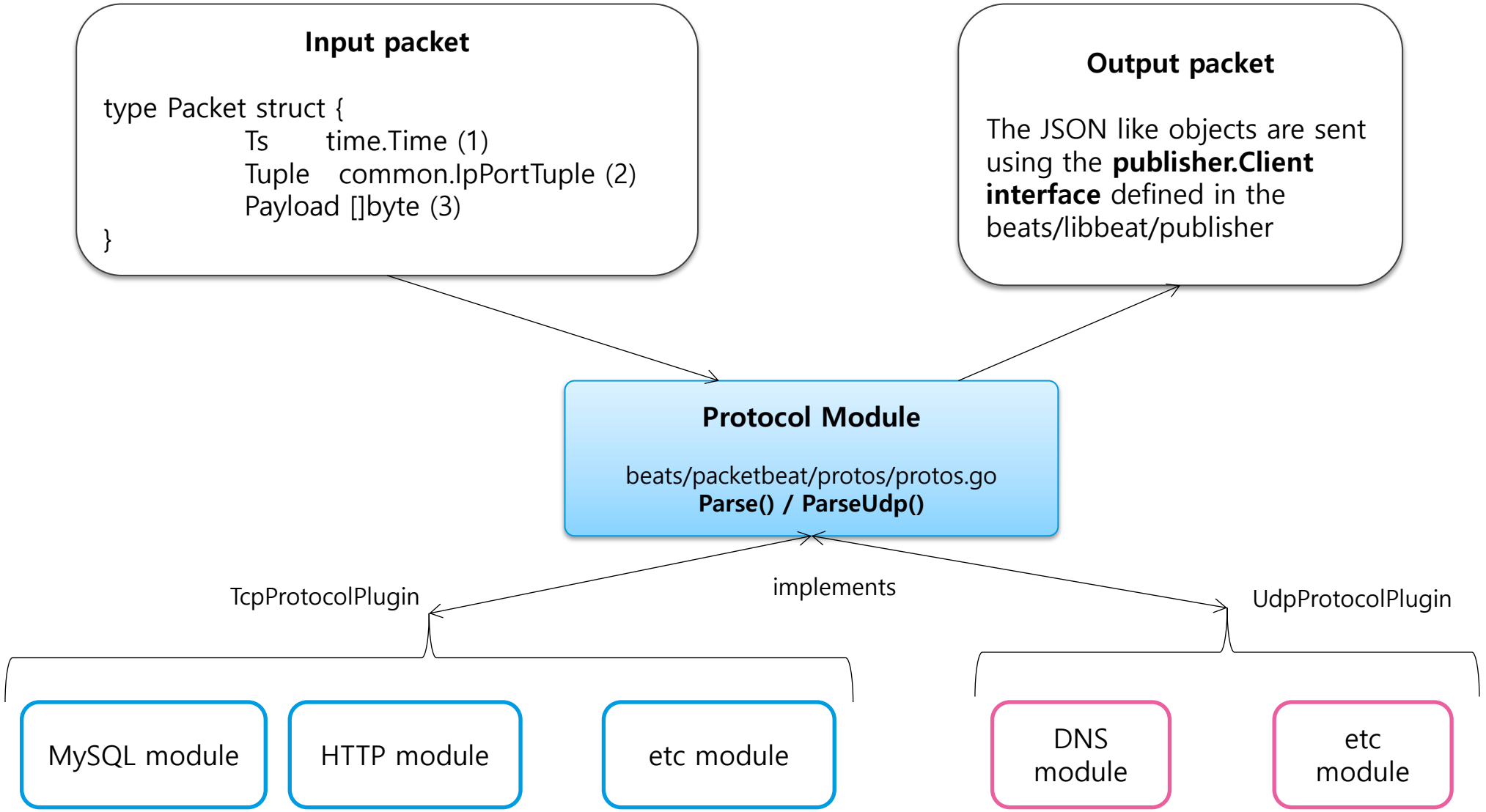
```
packetbeat.interfaces:
  device: eth0
  type: af_packet
  buffer_size_mb: 100
```

- Packetbeat's source code is split up in Go packages or modules.

- The protocol modules can be found in individual folders in the beats/packetbeat/protos directory of https://github.com/elastic/beats/tree/master/packetbeat

- At the high level, the protocols plugins receive raw packet data via the Parse() or ParseUdp() methods and produce objects that will be indexed into Elasticsearch. Parse() get called for every packets captured.

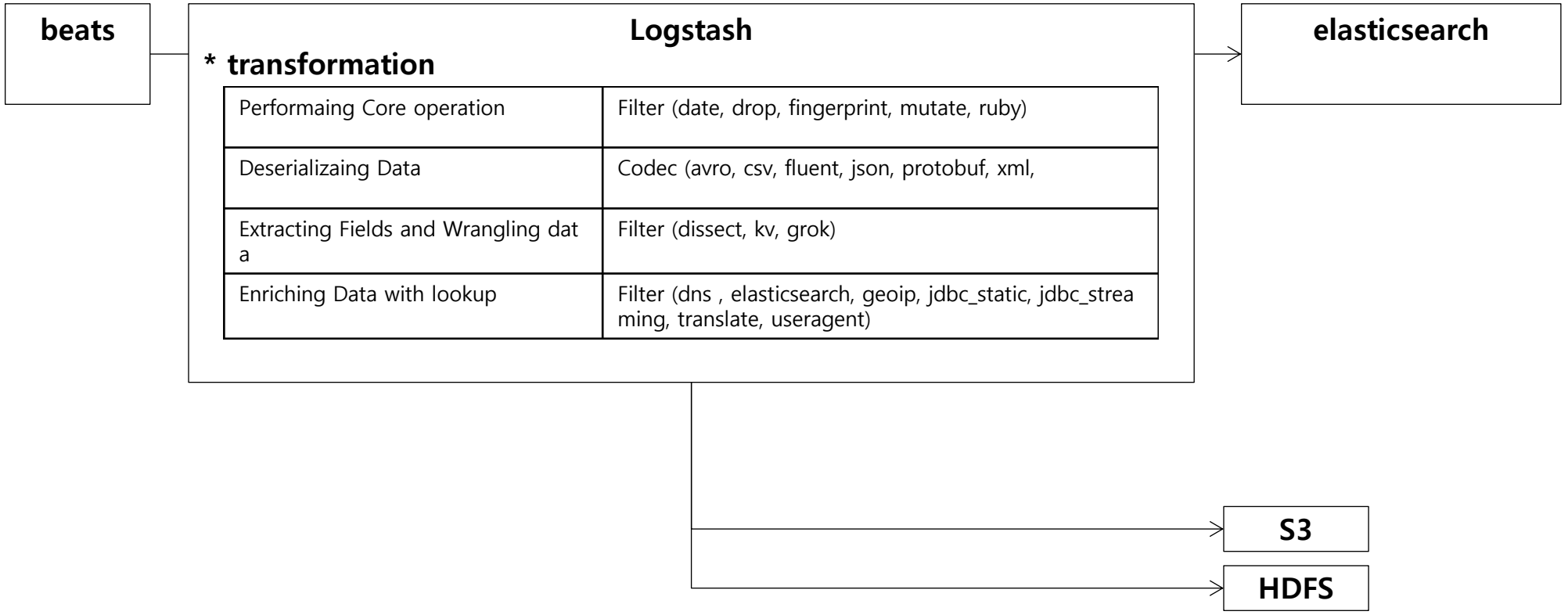
Beat - Packetbeat



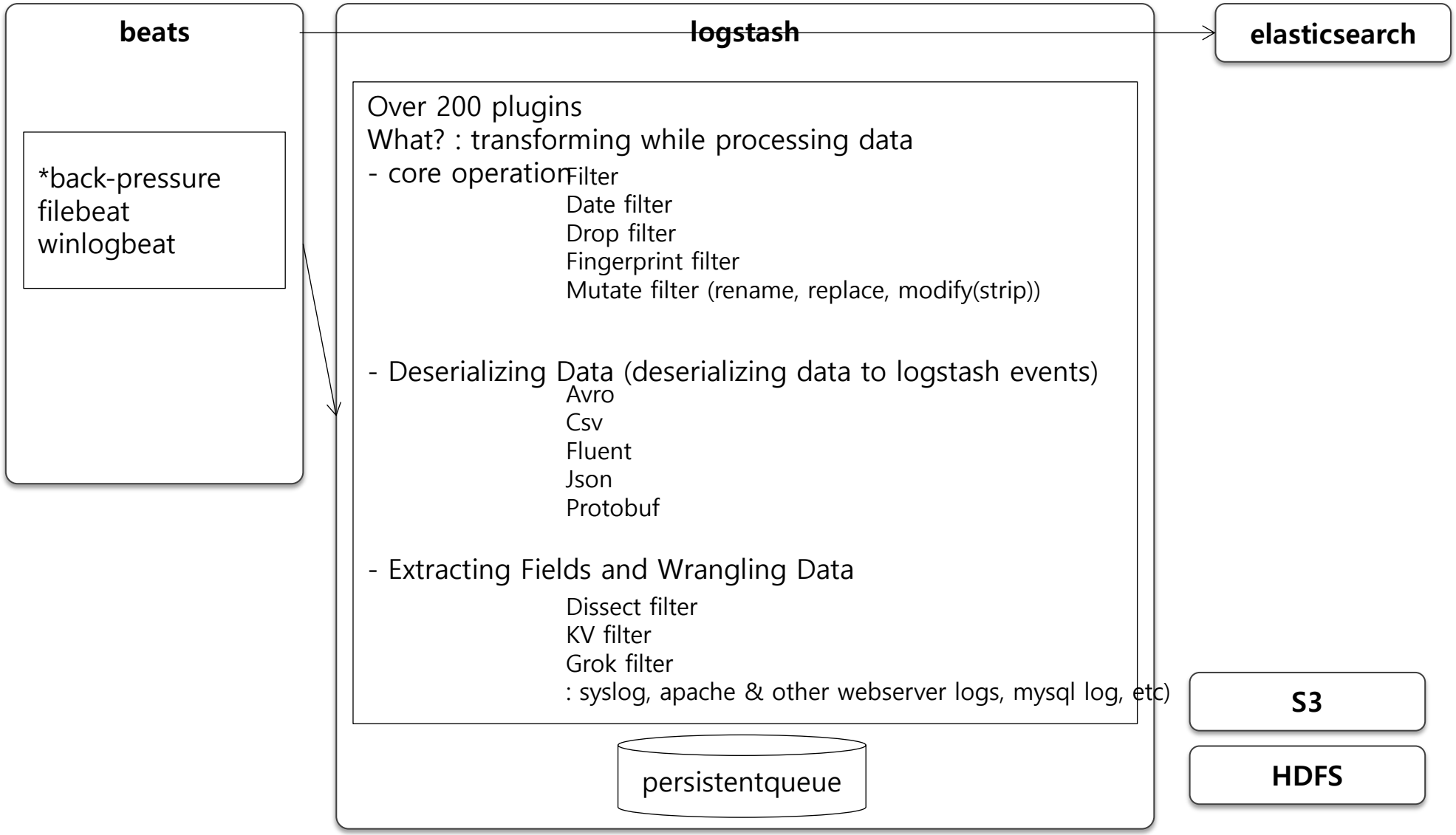
Use-cases

Type of Resources	Description
File	* Internal Systems Host/Event logs (e.g. Windows, Linux, Server, Switch, Storage) • Tracking new behavior of users requests (e.g. app generated log) • Measuring of customer response • Near Real Time Tracking of Events (Who tries to open my admin page) • Malicious behavior (Security, Fraudulent activity) • Anti-virus logs • Forensics on the machine • Investigation • Backend Information Factory • NRE on-demand interactive data mining • System resource monitoring (e.g. Storage Volume Usage) • Web farm health check (e.g. Cellwe)
Process	* System process (ex: top process-Mem,CPU) be able to integrated with monitoring system
Packet	* Network protocol (currently, HTTP)
Analytics	* Connect speedy search with Big Data Analytics can be packaged & integrated with system for analytics

Logstash


beats
Logstash
* transformation
Performaing Core operation
Filter (date, drop, fingerprint, mutate, ruby)
Deserializaing Data
Codec (avro, csv, fluent, json, protobuf, xml,
Extracting Fields and Wrangling dat a
Filter (dissect, kv, grok)
Enriching Data with lookup
Filter (dns , elasticsearch, geoip, jdbc_static, jdbc_strea ming, translate, useragent)
elasticsearch
S3
HDFS

Logstash


beats
*back-pressure
filebeat
winlogbeat
logstash
Over 200 plugins
What? : transforming while processing data
- core operation
Filter
Date filter
Drop filter
Fingerprint filter
Mutate filter (rename, replace, modify(strip))
- Deserializing Data (deserializing data to logstash events)
Avro
Csv
Fluent
Json
Protobuf
- Extracting Fields and Wrangling Data
Dissect filter
KV filter
Grok filter
: syslog, apache & other webserver logs, mysql log, etc)
persistentqueue
elasticsearch
S3
HDFS

Use-cases

✓ Application Monitoring

Get notified when:

- App response time exceeds SLA
- # of active users spikes or plummets
- User activity drops in a specific geo

✓ Real-Time Tracing

- Issue and errors or events

✓ External Interfacing

- Monitoring module as a service

✓ Optimization and Capacity Planning

- Is my current usage healthy?
- What is the growth rate in ingestion, search, total docs, or memory usage?
- How many nodes will I need in 6 months?
- What's the impact of my application change?

✓ Behavior Analysis

What other endpoints did a known malicious IP also visit?

✓ IP Operational Analytics

Get notified when:

- 404 errors > 100 in last 5 minutes
- Free disk space drops below 5%

✓ Security Analytics

Get notified when:

- > 5 failed logins from same IP in 15 min
- Unknown process consumes a large amount of memory
- Anomalous outbound data transfer

✓ Root Cause Analysis

- Why are certain nodes not responding?
- Why is my cluster not ingesting more data?
- Why is one node busier than the others?

Who use this stack?

LinkedIn

More than 30 ELK clusters (across multiple DCs)
Cluster with 50+ billion docs (50+TB) each
Daily indices average 3+ billion docs, 3+TB
15,000 indices
~1,000 shards per cluster

Wikipedia

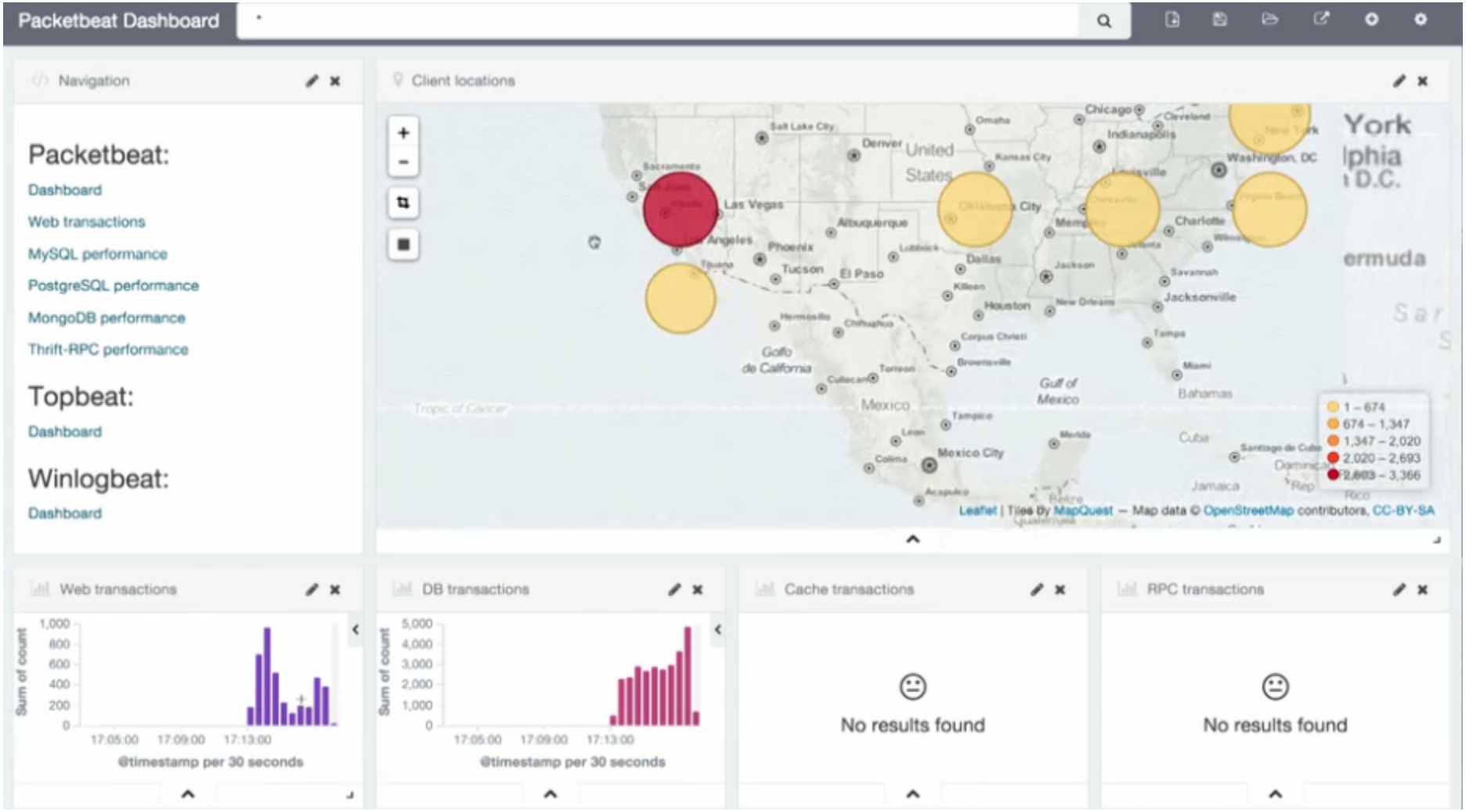
Uses Elasticsearch to provide full-text search with highlighted search Snippets.

GitHub

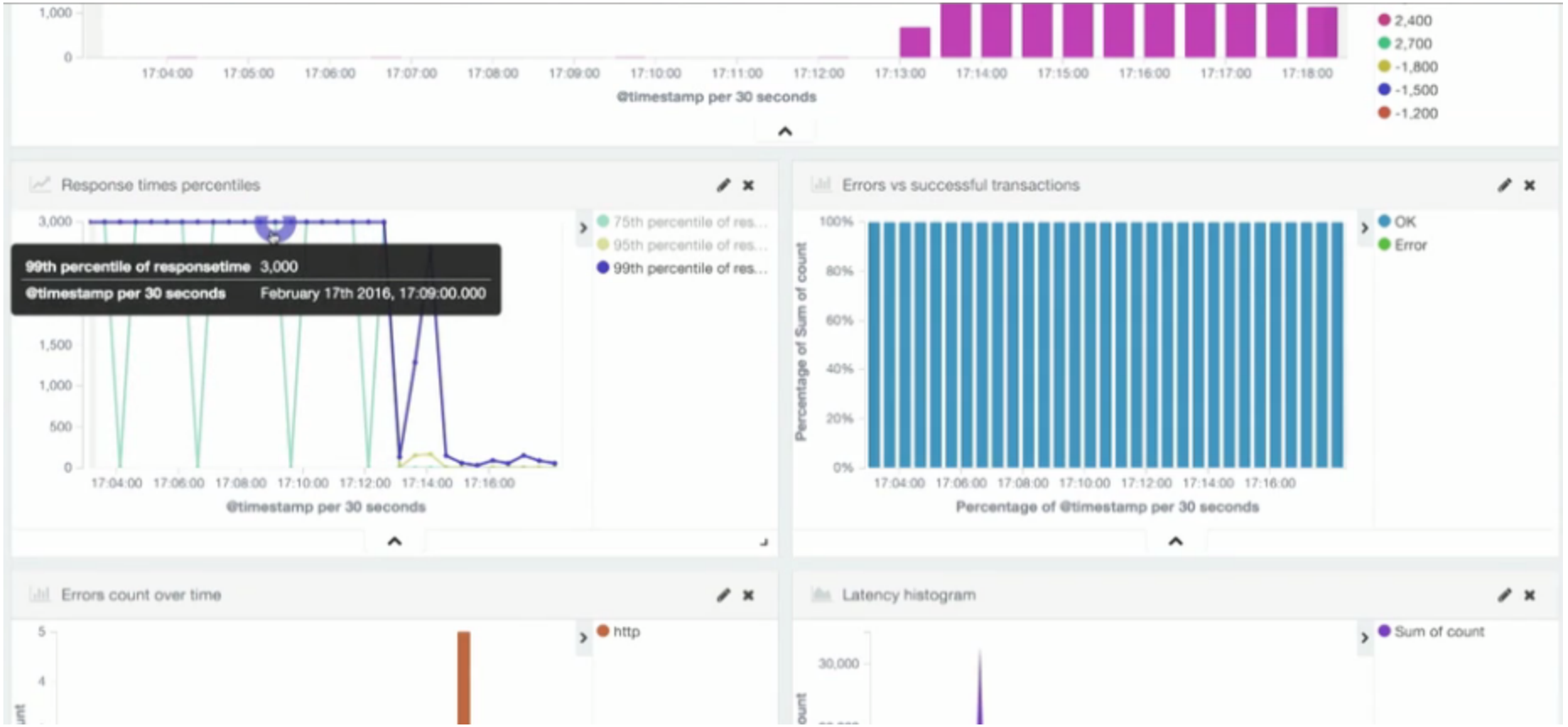
GitHub uses Elasticsearch to query 130 billion lines of code.

E-bay

Example Screen

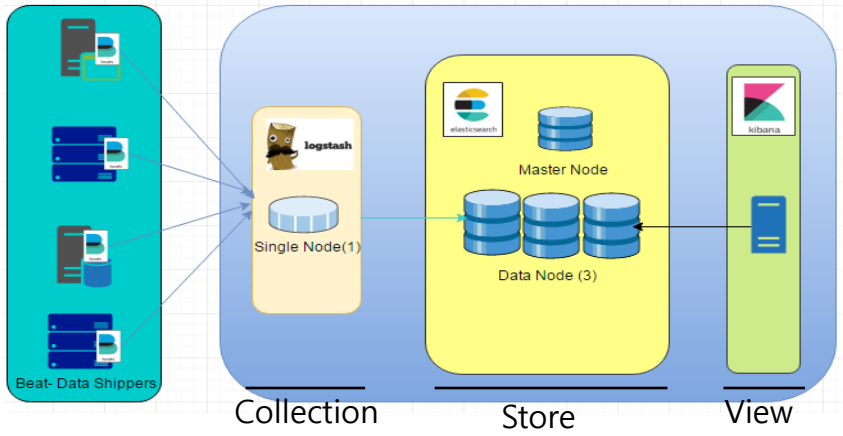


Example Screen

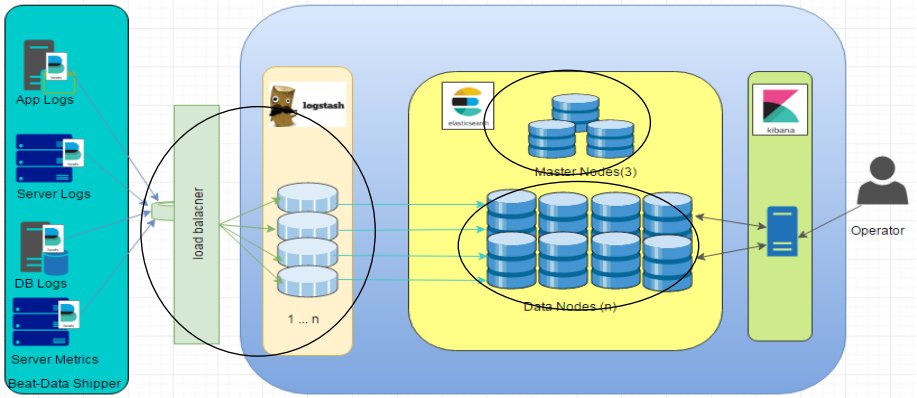


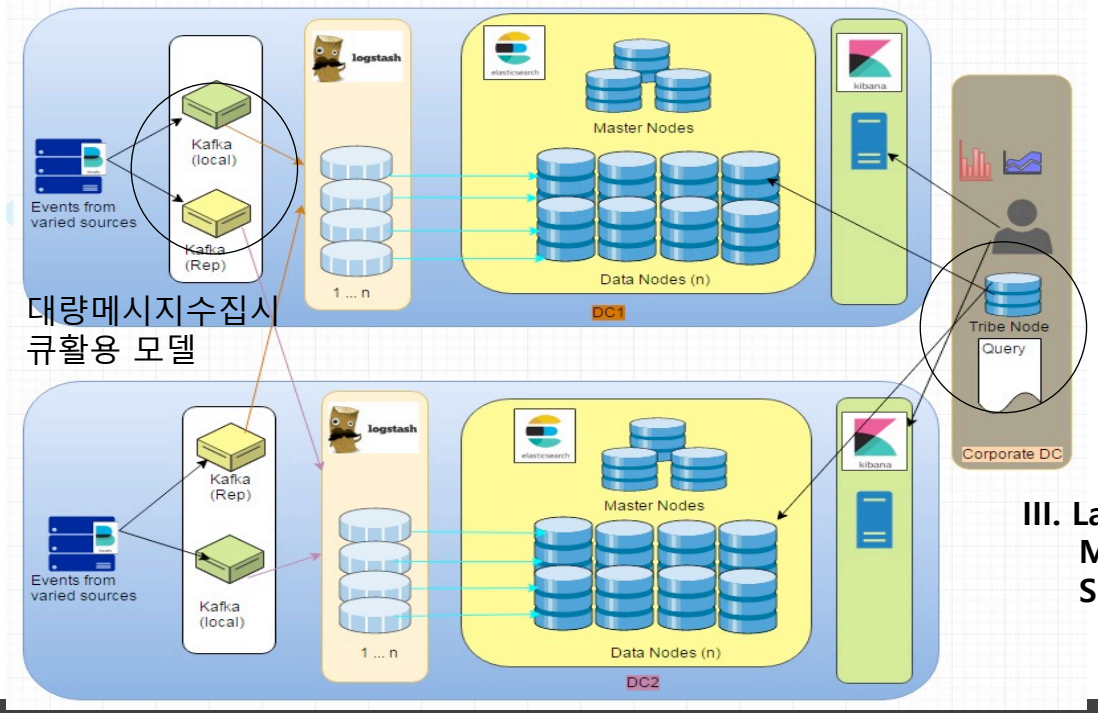
Attachments

I. Basic Model



II. Mid-Scale Model





III. Large Scale or Multi Region Support Model

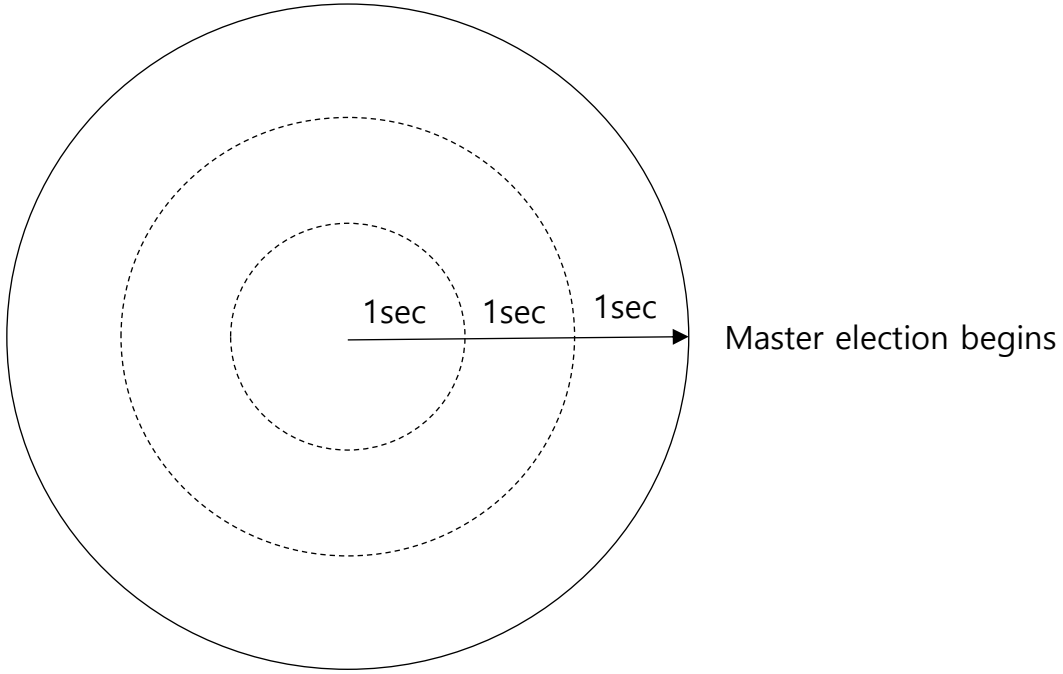
Master election

discovery.zen.ping_timeout: 3
No failure on master, but network is slow..then increase this number

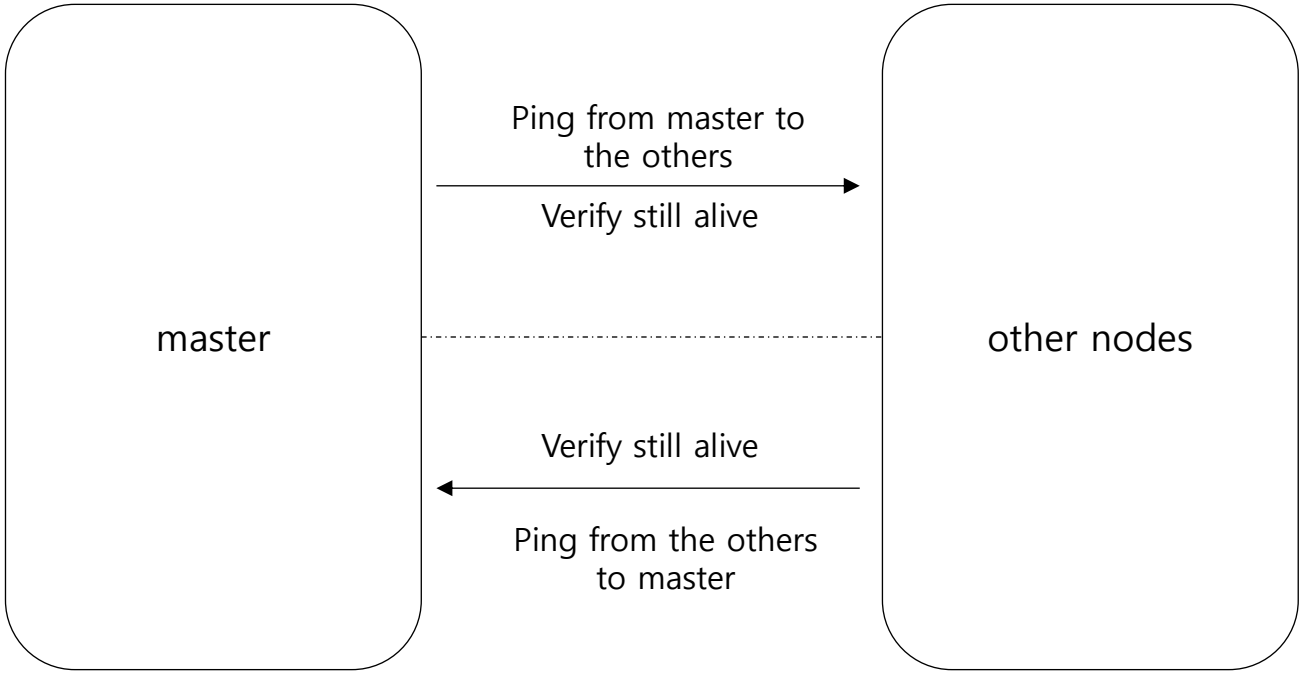
How the role be given?

- master
- data node
- ingest
- coordinator

node.master	true or false
node.data	false or true

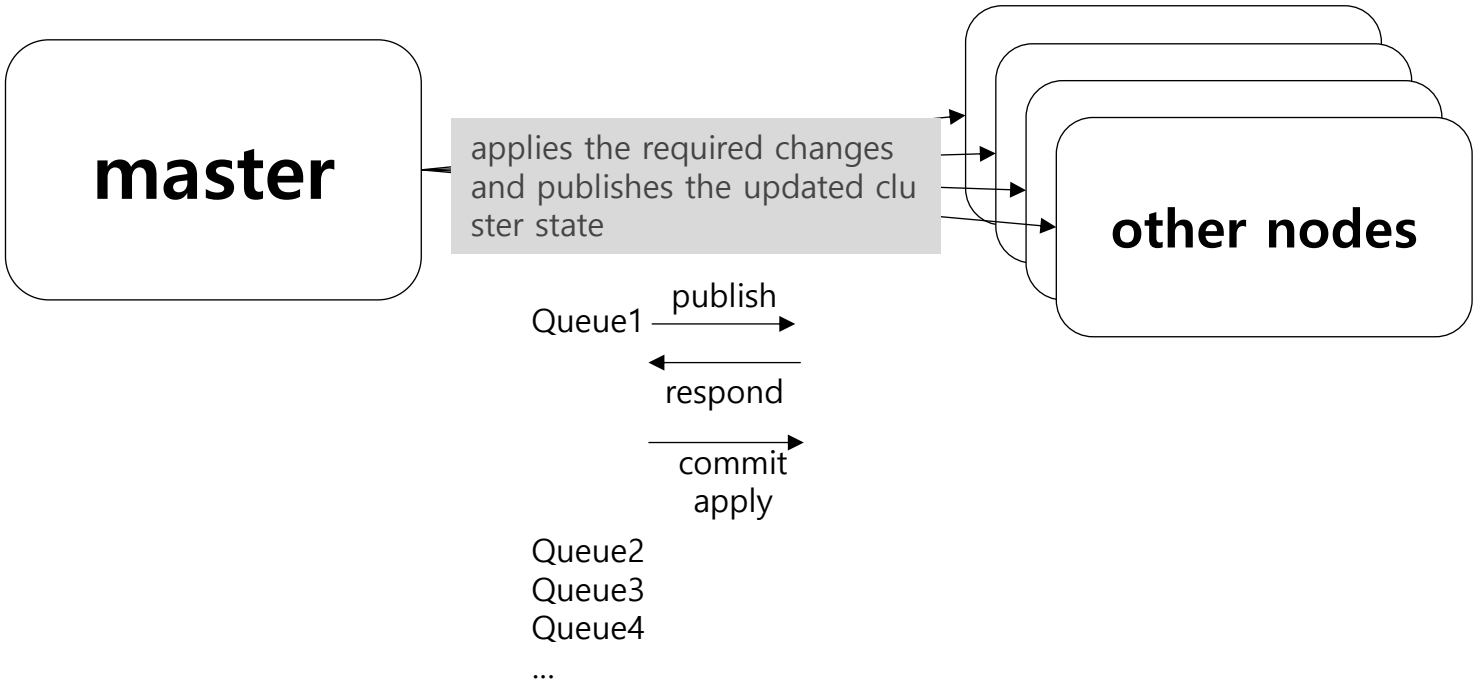


Fault detection




```
If we enable
: discovery.zen.fd.(ping_interval, ping_timeout, ping_retries)
```

Who’s goanna update cluster status?



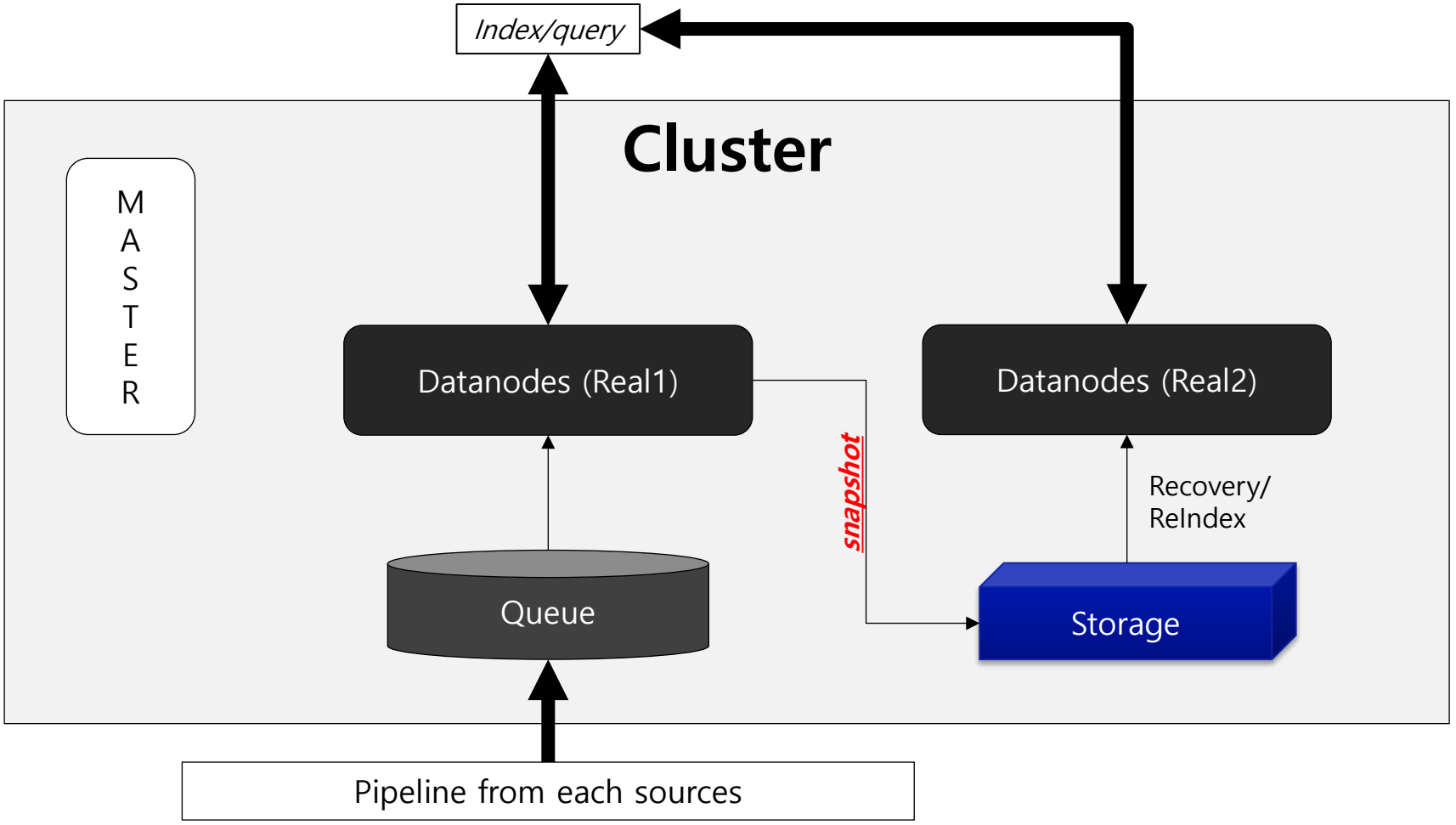
Important system settings

Setting	What
JVM heap size	Set the minimum heap size (Xms) and maximum heap size (Xmx) to be equal to each other Set Xmx to no more than 50% of your physical RAM More memory = more caching but, garbage collection pause
Disable swapping	sudo swapoff –a sysctl value vm.swappiness is set to 1
Increase file descriptor	set ulimit -n 65536 as root before starting Elasticsearch, or set nofile to 65536 in /etc/security/limits.conf
Sufficient virtual memory	sysctl -w vm.max_map_count=262144 update the vm.max_map_count setting in /etc/sysctl.conf
Sufficient thread	nproc to 4096

https://www.elastic.co/guide/en/elasticsearch/reference/6.1/bootstrap-checks.html

Attachments

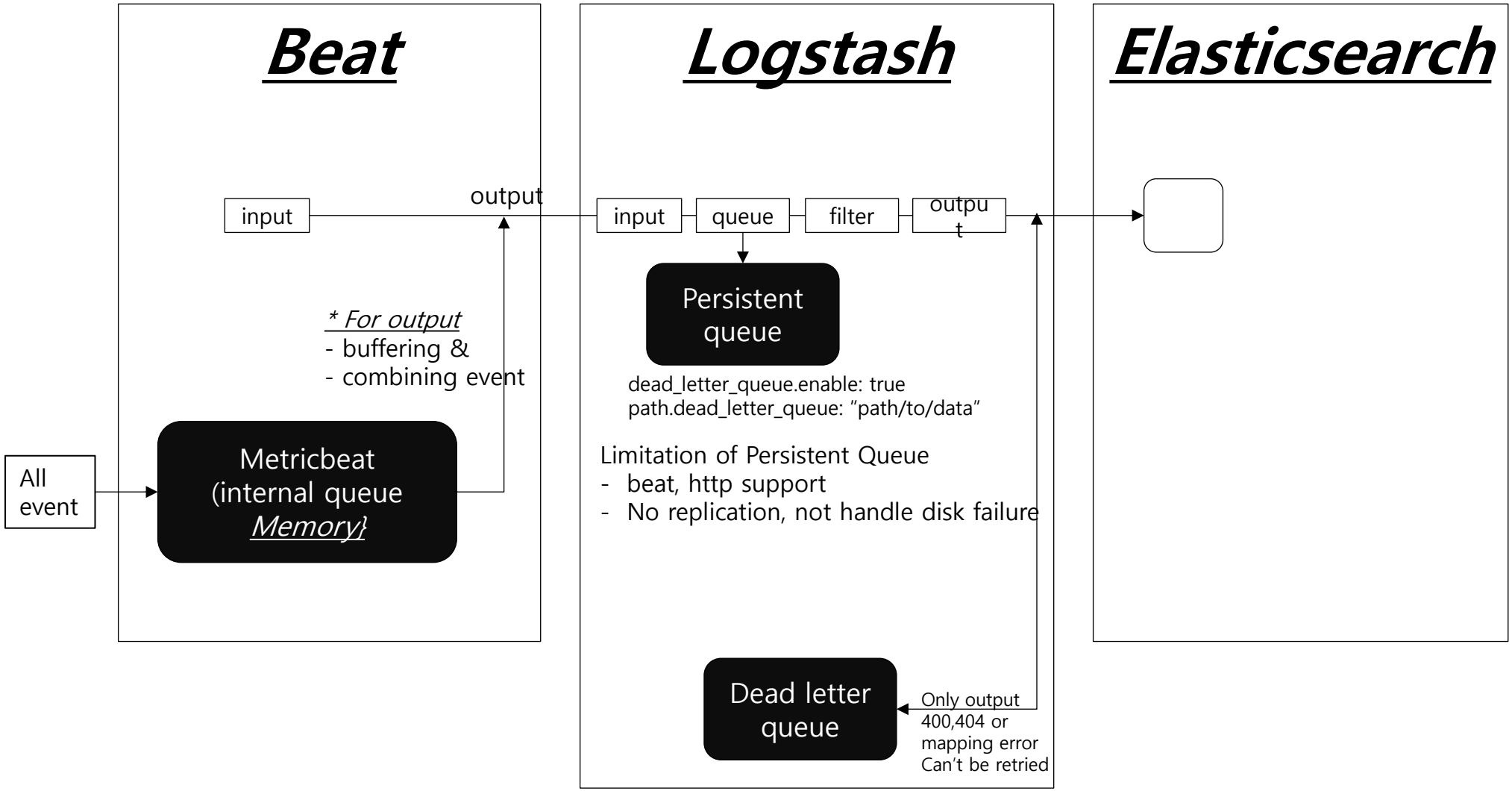
DR/HA


Index/query
Cluster
MASTER
Datanodes (Real1)
Datanodes (Real2)
snapshot
Recovery/
ReIndex
Queue
Storage
Pipeline from each sources

Hot/Warm Architecture

1.	Tag the Nodes	node.attr.my_node_type:hot
2.	Configure the Hot Data on each Index	index.routing.allocation.require.my_node_type:hot
3.	Interval	1 month later
4.	Move Older Shards to Warm	Index.routing.allocation.require.my_node_type:warm

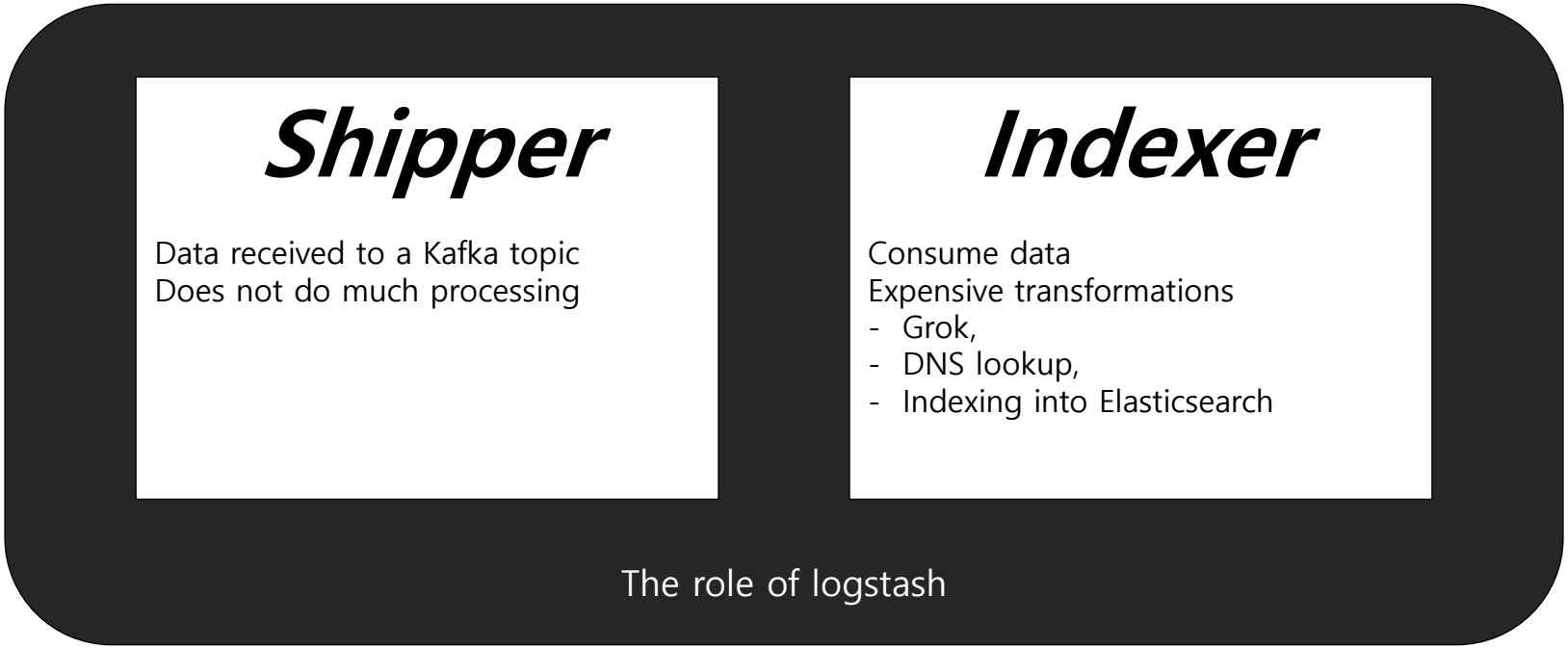
Queue


Beat
Logstash
Elasticsearch
input
output
input
queue
filter
output
Persistent queue
* For output
- buffering &
- combining event
dead_letter_queue.enable: true
path.dead_letter_queue: "path/to/data"
All event
Metricbeat (internal queue Memory}
Limitation of Persistent Queue
- beat, http support
- No replication, not handle disk failure
Dead letter queue
Only output 400,404 or mapping error Can't be retried

Logstash pipeline

Logstash stages for a scalability

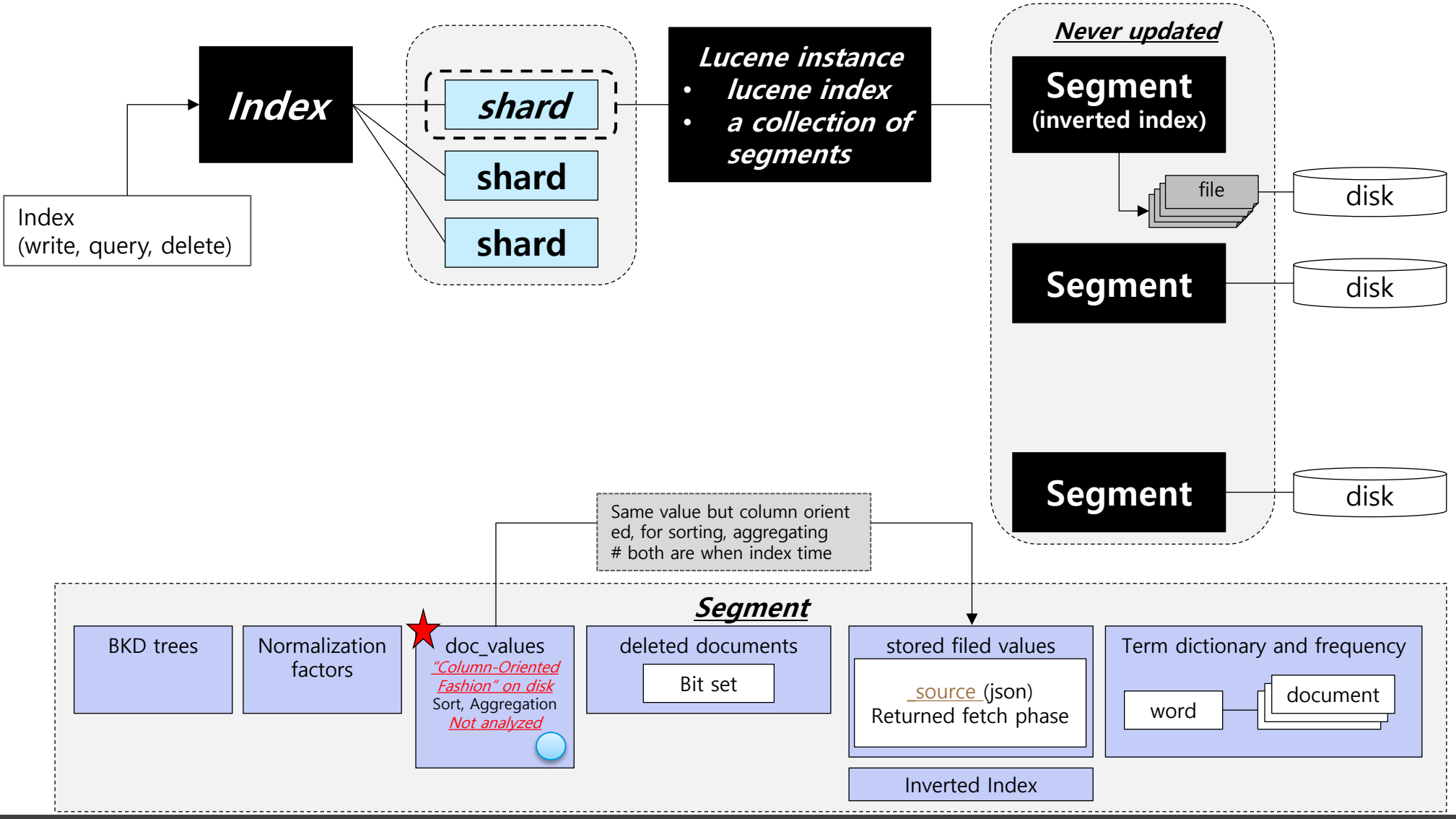
AS...



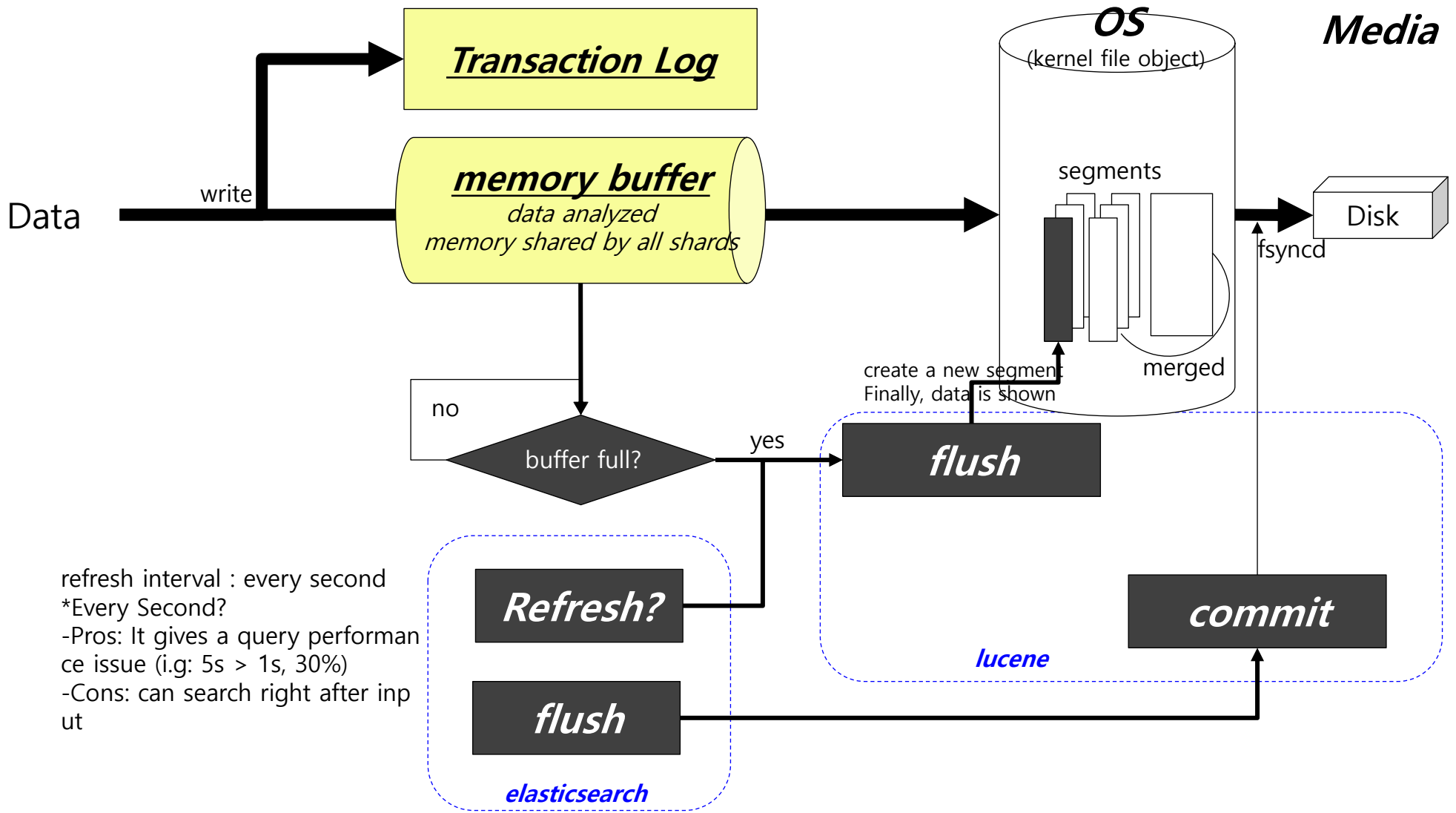
Field modeling (mapping)

No sorting or aggregation? Set "doc_values": false to save disk space

Anatomy of index



Refresh & Fresh



refresh interval : every second
*Every Second?
-Pros: It gives a query performance issue (i.g: 5s > 1s, 30%)
-Cons: can search right after input

Cluster Level Shard Allocation

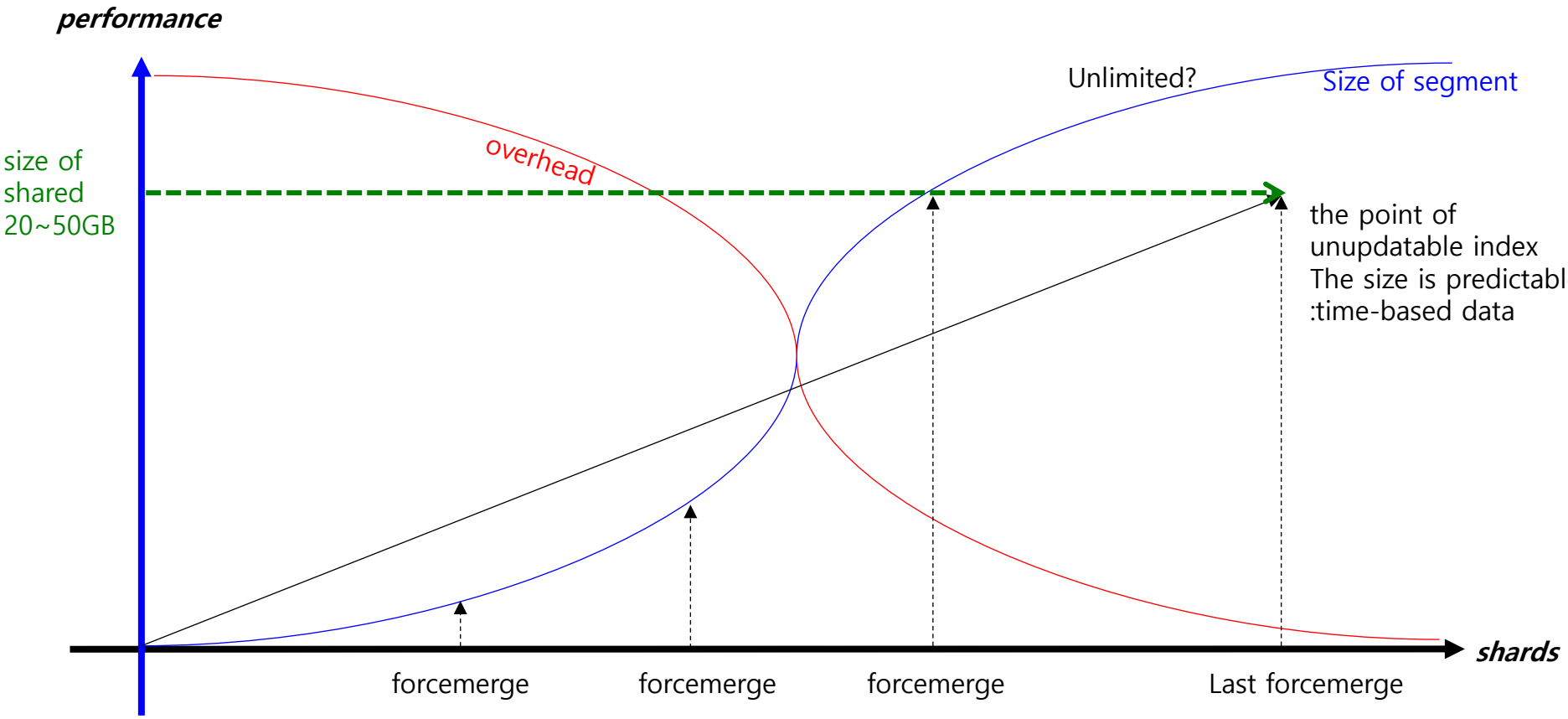
When does it happens?
1. Replica allocation
2. Rebalancing
3. Nodes are added or removed
4. Initial recovery

Cluster level	shard *allocation* settings	Allocation & Recovery
	shard *rebalancing* settings	To control Rebalancing of shards across the cluster
	shard *balancing* heuristics	Extra supplement options ,which can be used.
Disk level	Shard allocate or relocate	Watermark
Allocation Awareness		disk, hosts, network, switch, rack
Shard allocation filtering		Include, require, exclude

balancing algorithm

Shard & Segment

Reindex? When
Primary & Replica

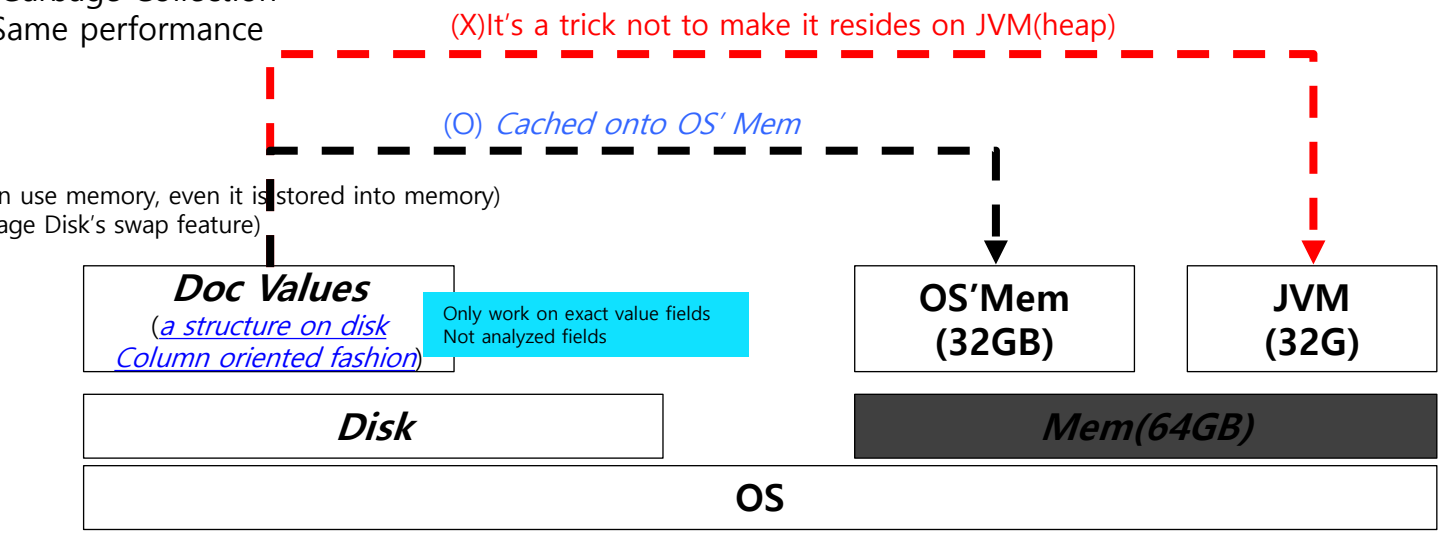


- #Shard(20~25)/Heap(1GB)
- #700 Shards => 32GB Node
- 1.4TB Disk (then, we need to think....4TB, 8TB, 16TB?)

Doc Values

What Benefit?

- Smaller Heap (it does not use JVM Heap)
- Smaller Garbage Collection
- Finally, Same performance



- Performance (can use memory, even it is stored into memory)
- Scalability (leverage Disk's swap feature)

Note) why doc valus come out?

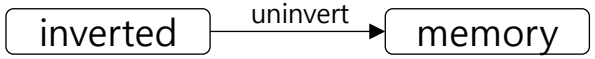
Doc Values vs. Inverted Index

:uninvert the inverted index
good for searching
@_search time
but, some action(sorting)
go into heap?
it has to be uninvert
lots of memory be used

=> Run out of memory then go into out of memory

remember,
field('text') has to be analyzed..so that reason, there is no doc values
field('keyword') has doc values. so, it is good for sorting and aggregating

Comparison (fielddata vs. doc_values)

Text field	*Keyword field*	*where*
Fielddata (In-memory data structure) : when aggregation, sorting, in a script : uninvert inverted index **It use high memory when loading high cardinality text fileds**  **Condition, enable fielddata on text field(PUT)**		Memory(JVM)
no doc value	***doc value*** (sorting) : on-disk data structure, column oriented structure	
searching	***aggregation, sorting***	Both required Then, use both
index time	query time	When

_Source

What is source?

- Source contains a original JSON document body that was passed at index time.
- Source does not indexed. (thus is not searchable)
- It returned when executing fetch requests
- It can be disabled "mappings._doc._source.enabled=false" but,...think about pros & cons

Able to (Pros)

- Reduce space

Proposal1: why don't you use compression option?

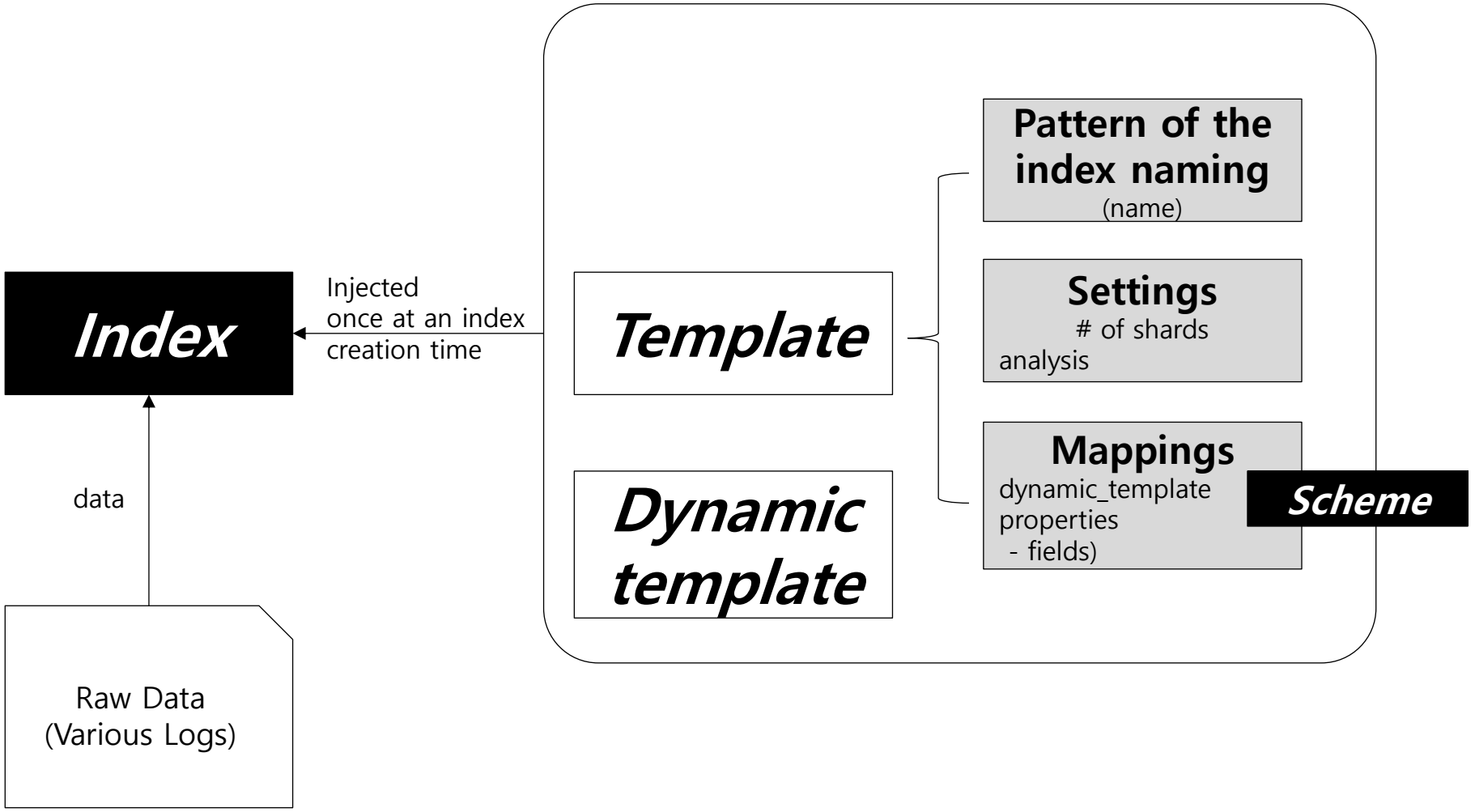
Proposal2: small document consist only of numbers,dates,keywords, no update, no highlighting

Proposal3: prune contents of the _source after document has been indexed (searchable(=indexed), just not stored to _source)

Not able to (Cons)

- update, update_by_query
- ***Reindex***
- On the fly highlighting
- Debug query
- Repair index corruption automatically

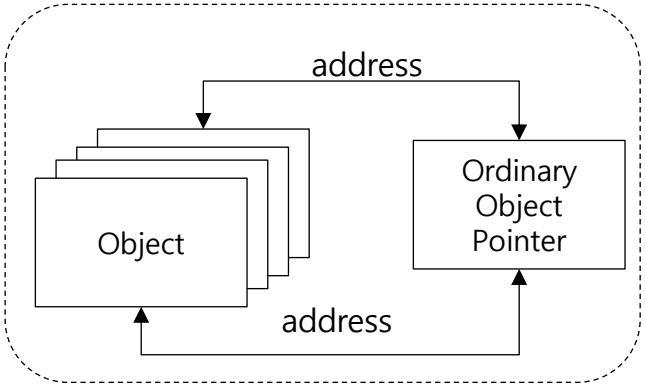
Index & Template



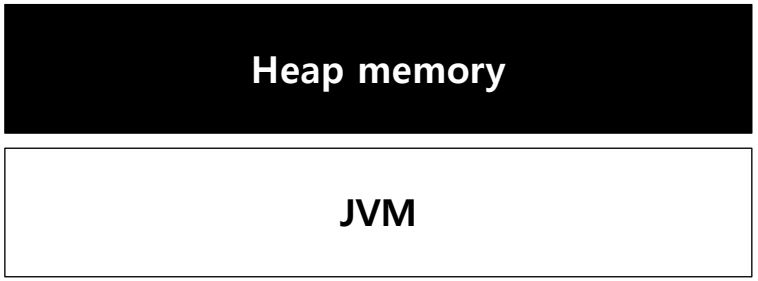
ES & Java heap

Why heap does not recommended more than 32GB?

- JVM handle heap memory where is the place objects are living.
  - 2^32 = 4GB memory (max), if mem over > 4GB? Then it use COOP
  - 2^64 = 18EB (more memory, more computation)



Address to access
object in memory



In genera, OOP used
If (heap)object size > 4GB, then COOP be used
If (heap)object size > 32GB, the 64bit OOP be used
Compressed ordinary object object)
For this reason, 32GB heap is recommended

Too much small > OOM
Too much big > Garbage Collection

ES & Java heap

Too Small	*Too Large*
Reduced throughput from constant garbage collection pauses Reduce the number of operations Constant short pauses -> reduce the number of indexing operations and queries per second Reduce the memory available for indexing buffers, caches, memory-hungry features like aggregations and suggesters	Long latency spikes from full-heap garbage collections User requests will sometimes see unacceptably-long response times. Long pause is indistinguishable from node that is un reachable because it is hung or isolated from the cluster. No code executing(no read, write) Long garbage collection pause no-master -> master election Reallocating the paused node's assigned shards. -> increase network traffic , increase disk I/O Finally, instability

ES & Java Heap

Use half of the total memory or a little bit lesser

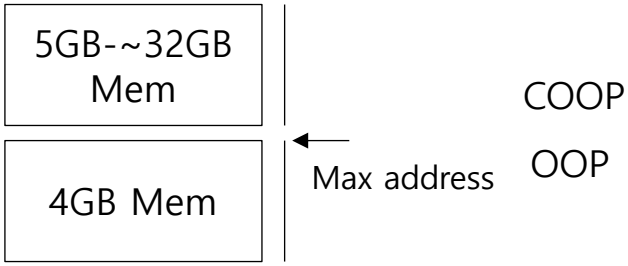
Assign 32GB Memory force to use Compressed OOP, pl, check the limit and possibility for COOP
>java –Xmx32766m –XX:+PrintFlagsFinal 2>/dev/null | grep UseCompressedOops

Check the memory be able to started from Zero Based

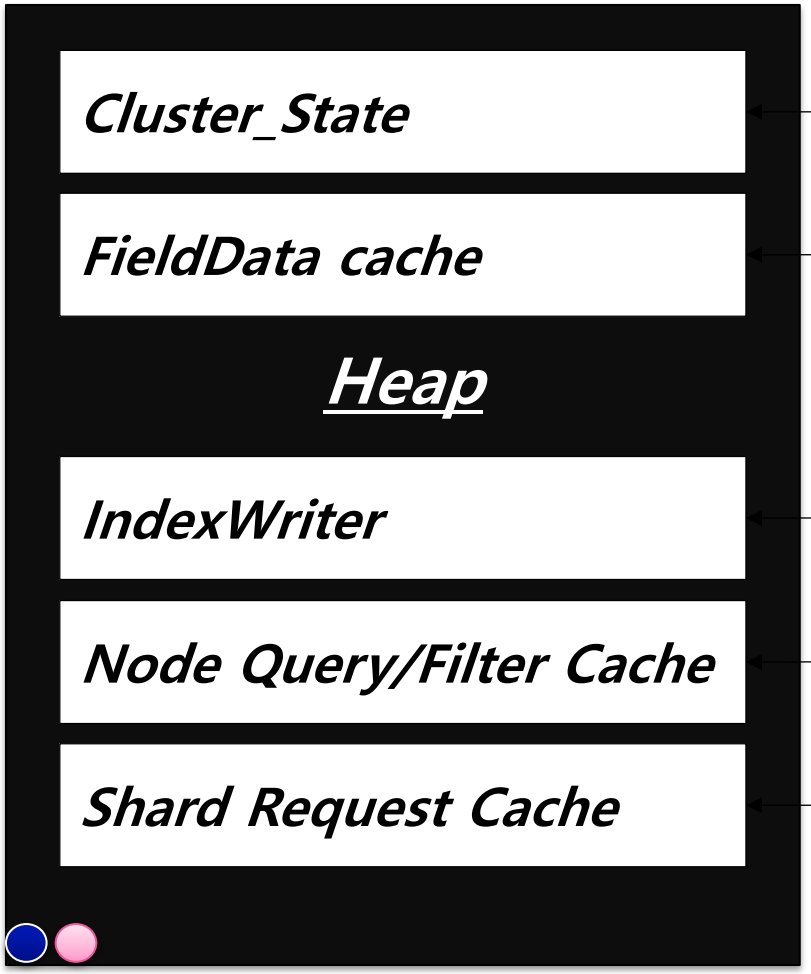
JVM store address of each objects to be structured OOP. (max 4GB of 32bits)
There is another concept called COOP(Compressed Ordinary Objects Pointer), which is memory points to address objects where in Heap memory to extend the addressing.
Can extends address thru COOP 8 times bigger than general OOP
So,

OOP	4GB pointer
COOP	Capability32GB

Recommended not to use 64bit



Which things are loaded into Heap Memory?



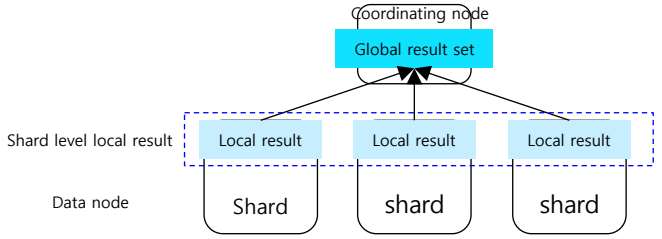
The status of cluster

_source
(holds the field value,
use this cache while computing aggregation)
Static setting(=dynamic), max 60% of heap by default

Writing Index (newly indexed docs)

If a doc matches a filter context
(shared by all shards)

Caches results of query by each shard



Performance tuning

Indexing Buffer	Default to 10% of the node heap (i.g, 2GB of 16GB Heap), is shared by all shard * indices.memory.index_buffer_size: 5%

Hardware Recommendations

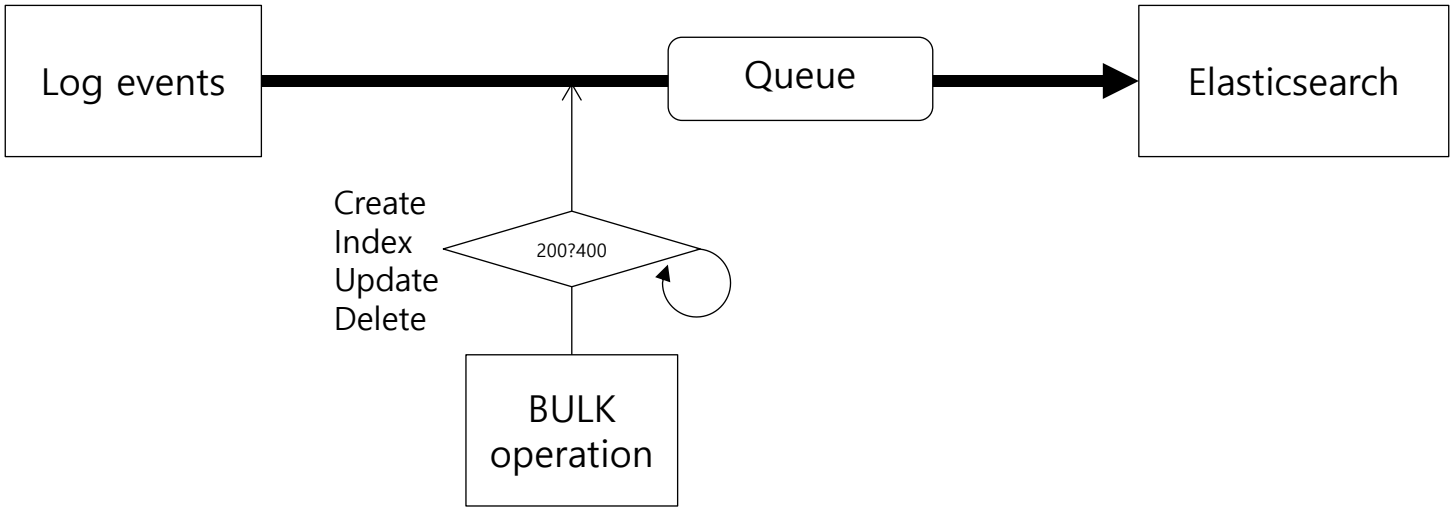
<table>
<tr><th>Items</th><th>Spec</th><th>Description</th></tr>
<tr><td>Box</td><td>X86</td><td>General X86 boxes</td></tr>
<tr><td>CPU</td><td>2 ~ 8 cores
32cores (production)</td><td>Modern processor
Choose more cores than faster CPUs</td></tr>
<tr><td>Mem</td><td>• Min: 16GB
• Best: 32GB
• Ideal: 64GB
More sorting and aggregation? 64GB</td><td>8GB memory does not recommended
Do not over 32GB for Java Heap
Give half your memory to Lucene
ElasticSearch 32GB, Rest of other Lucene
<table><tr><td>32(ES)</td><td>Rest(Lucene)</td></tr></table></td></tr>
<tr><td>Disks</td><td>SSD
(still question?
How many disks/node)
Size : 2TB</td><td>To improve query and indexing performance

Pl, check disk I/O scheduler
cfq is used by default. It is efficient to support spindle disk. But not for SSD.
No recommendation Network attached storage
http://www.mimul.com/pebble/default/2012/05/12/1336793032150.html
<table><tr><td>cfq</td><td>spinning disk</td><td></td></tr><tr><td>deadline</td><td>SSD</td><td>Raid0 DBMS</td></tr><tr><td>noop</td><td>SSD</td><td></td></tr></table></td></tr>
<tr><td>Network</td><td>Bandwidth(10GbE)</td><td>Do not cluster across data centers even if the DCs are colocated</td></tr>
</table>

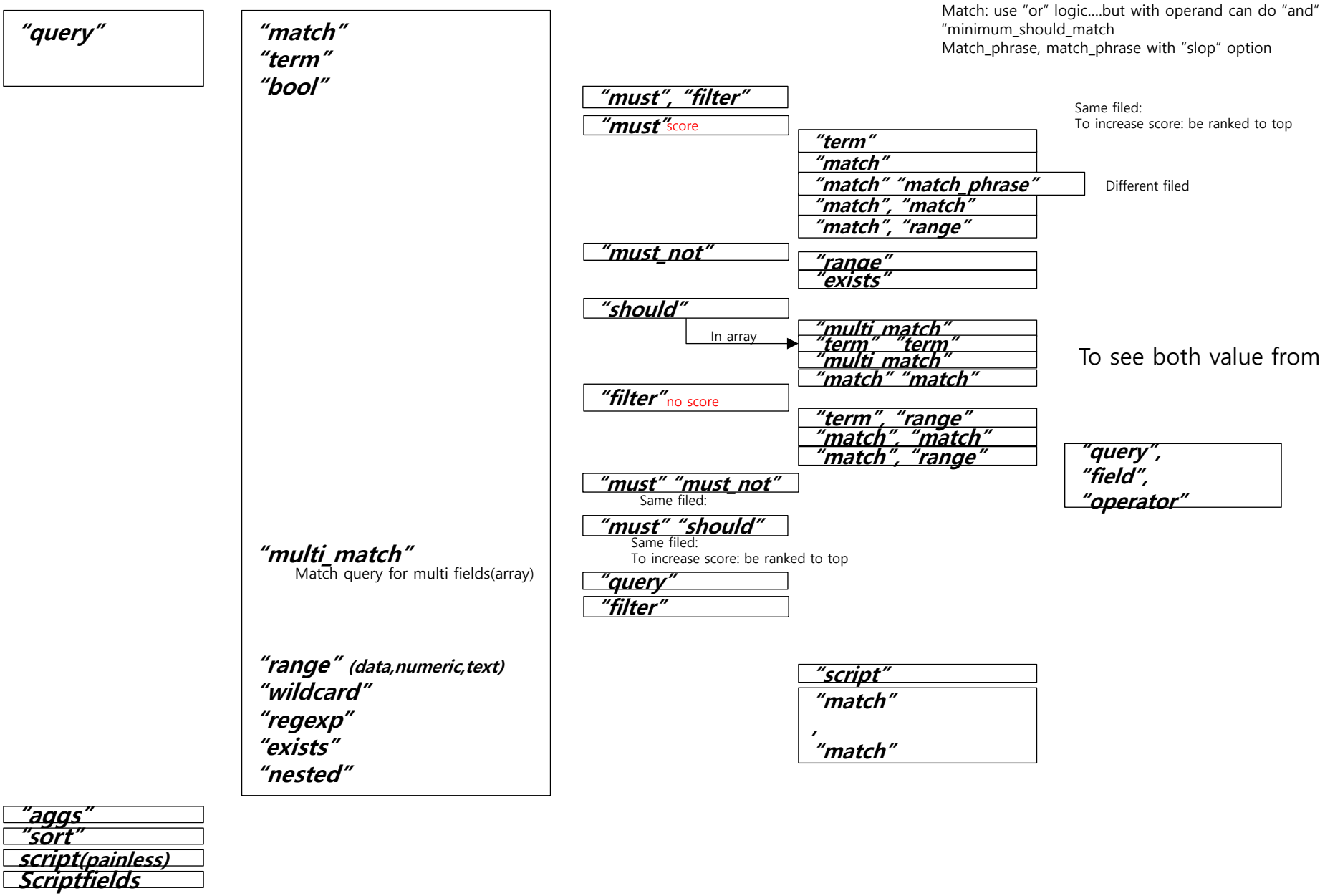
Role of Nodes

Node Type	Role	Description	Options
Master			node.master: true node.data: false node.ingest: false node.ml: false xpack.ml.enabled: true
Data			node.master: false node.data: true node.ingest: false node.ml: false
Ingest	Pre-process documents b efore actual document	1. define pipeline pipeline-> specifies a series of processors description processors blablabla processors blablabla	node.master: false node.data: false node.ingest: true search.remote.connect: false node.ml: false
Logstash			
Kibana			
Machine Learning			node.master: false node.data: false node.ingest: false search.remote.connect: false node.ml: true xpack.ml.enabled: true
Coordinating			node.master: false node.data: false node.ingest: false search.remote.connect: false node.ml: false

Bulk


Log events
Queue
Elasticsearch
Create
Index
Update
Delete
200?400
BULK
operation

Query Structures(I)



Query Structures(II)

"query"

"match"
"term"
"bool"

"must"
"filter"
"must"
"should"
"must_not"

"match"
"match", "match"

"range"
"exists"

"multi_match"

"should"

"query"
"filter"

"multi_match"

"query",
"field",
"operator"

"script"

"match"
,
"match"

term vs. terms
Indexed vs. not analyzed
able to refer to other indexes

"multi_match"

"range"
"wildcard"
"regexp"
"exists"
"nested"

"aggs"

"bucket"
(range, terms)

"metrics"
(sum, avg, min, max, stats, cardinality)

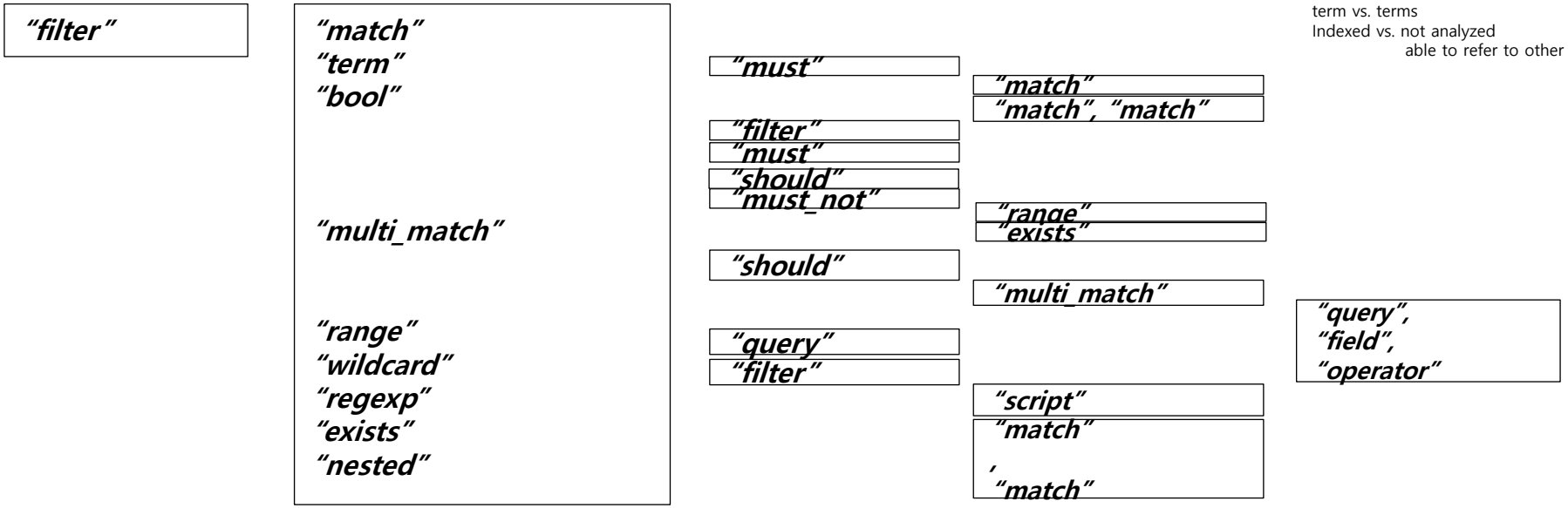
"Combining (bucket + metric)"
"range + avg)

"date_range"

Buckets("terms")
The result of terms agg does not exactly accurate cause the result from shards be backed by top of each shards which means some data from a shard does not be returned even the value has high than other value of different shard

* Cardinality: unique count

Query Structures(III)



term vs. terms
Indexed vs. not analyzed
able to refer to other indexes

Query

Bool query

must	Query with score
filter	Query without score
should	
must_not	Filter context (=no score)

Term level Query

term query

Exact term	Inverted index. (boost option to give higher relevance)

terms query

Exact term	Not analyzed

exist query

Exact term	Not analyzed

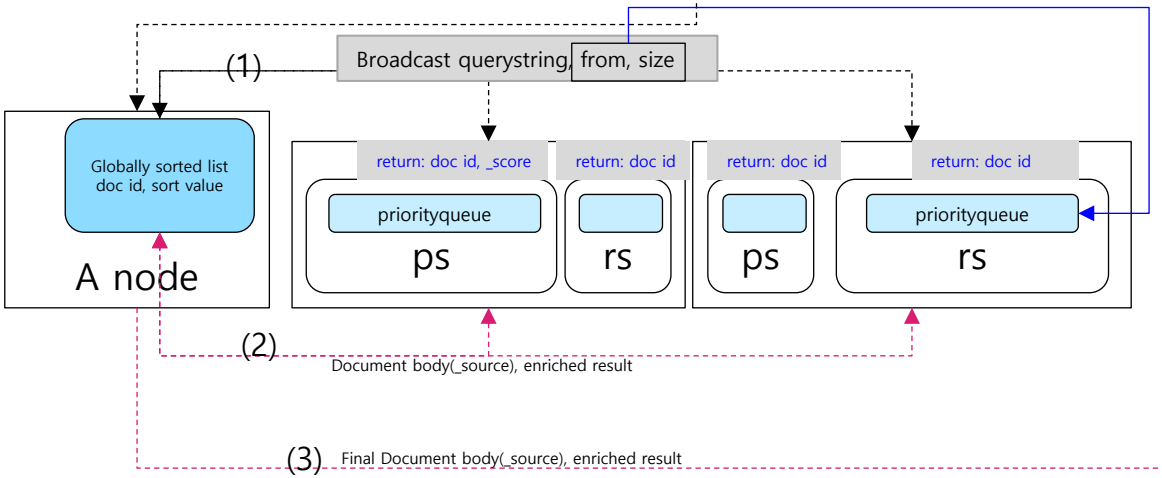
Do you know the type of search?

Search phase

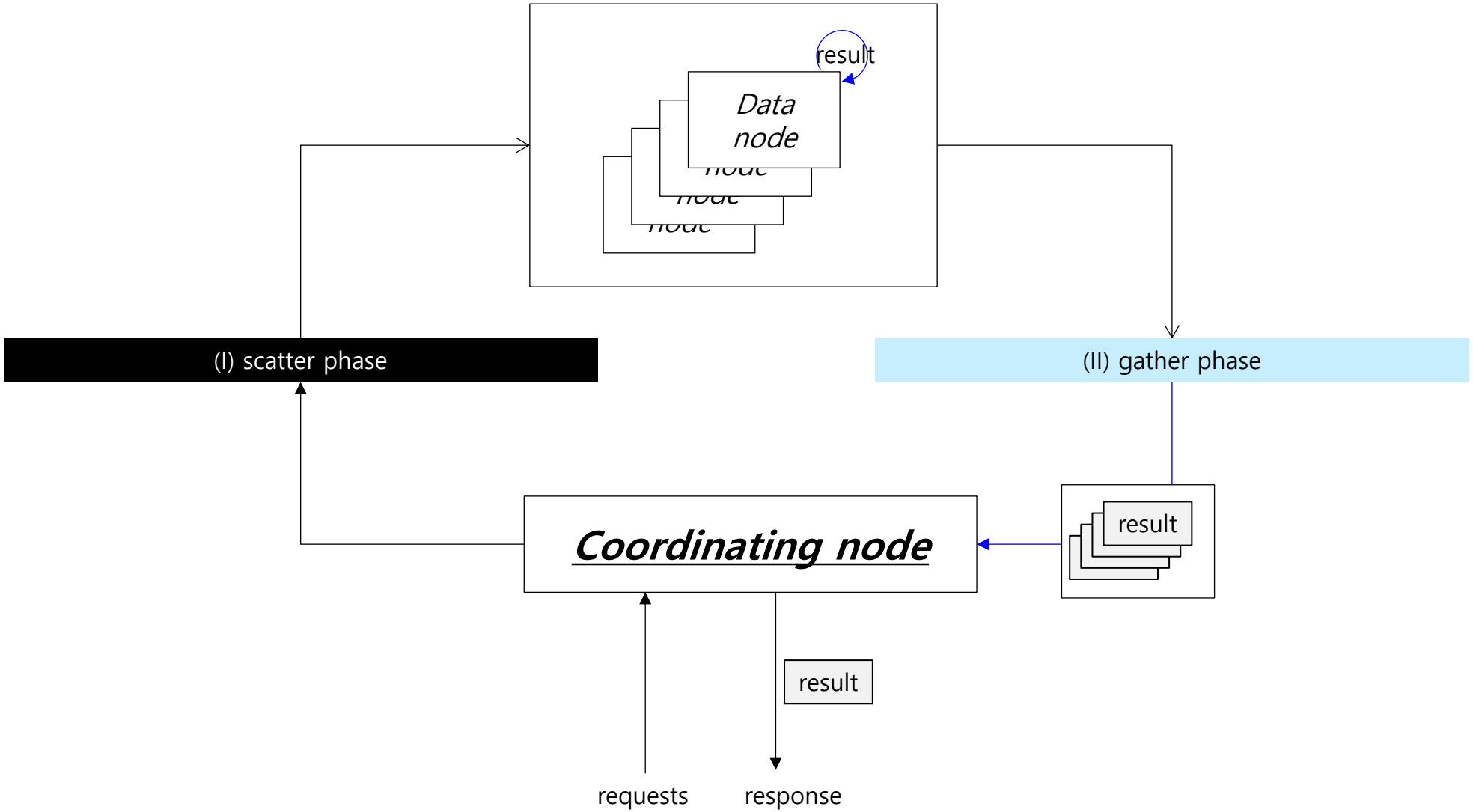
count	Only a query phase (no need search results)
query_and_fetch	Query phase Fetch phase
dfs_query_then_fetch	
scan	

Phase1-query

Phase2-fetch



Coordinating node

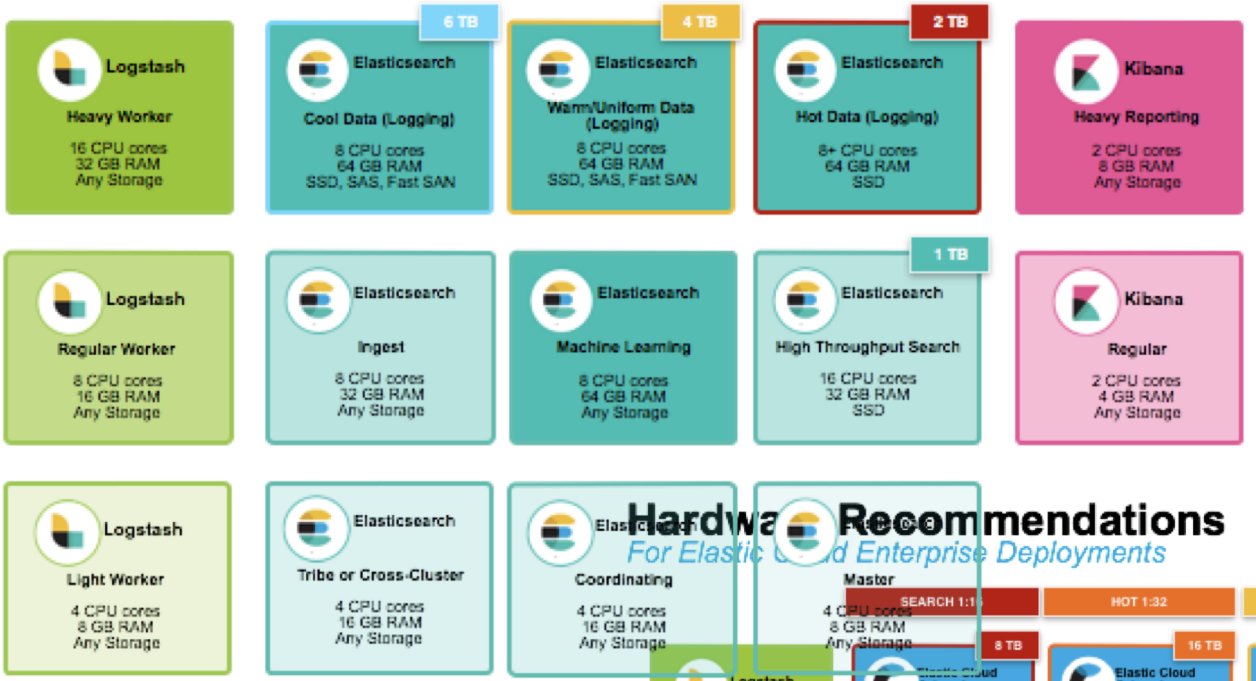


Sizing

CPU	Heavy indexing Complex analysis chains ES scripting Logstash filters
Memory	Sorting large datasets Complex aggregations
Disk I/O	Heavy indexing loads Logstash file input Logstash file output Lucene merges

Hardware Recommendations

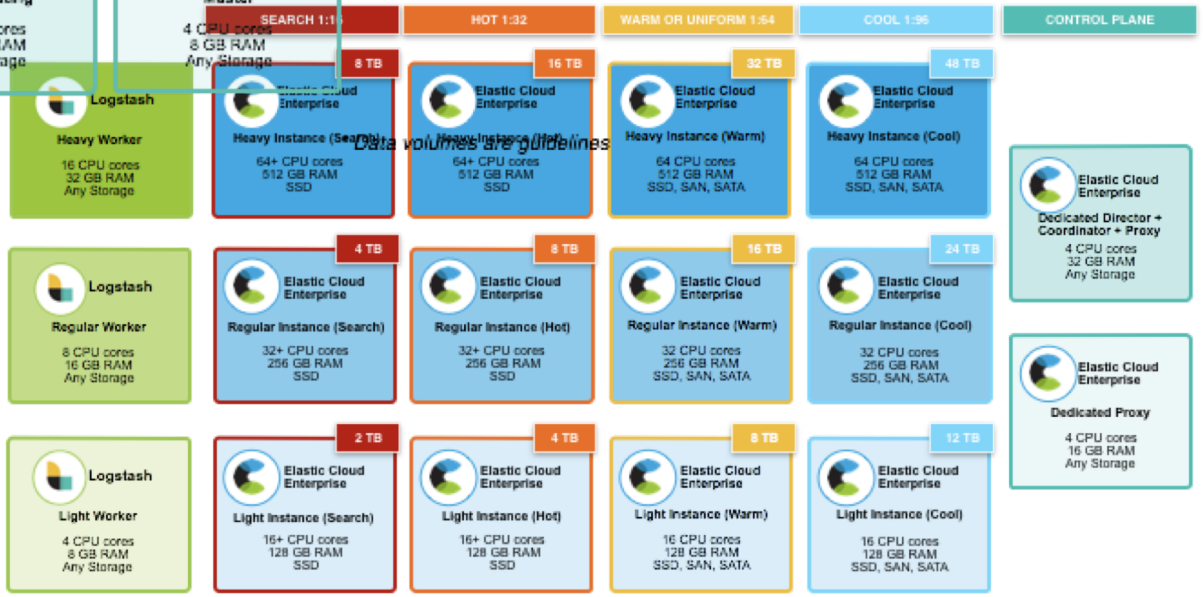
For On-Premise or IaaS Cloud Deployments



Hardware Recommendations

For Elastic Cloud Enterprise Deployments

- *Data volumes are guidelines*
- *Multi data tiers available from ECE 1.2*



Use cases

1) Coinbase

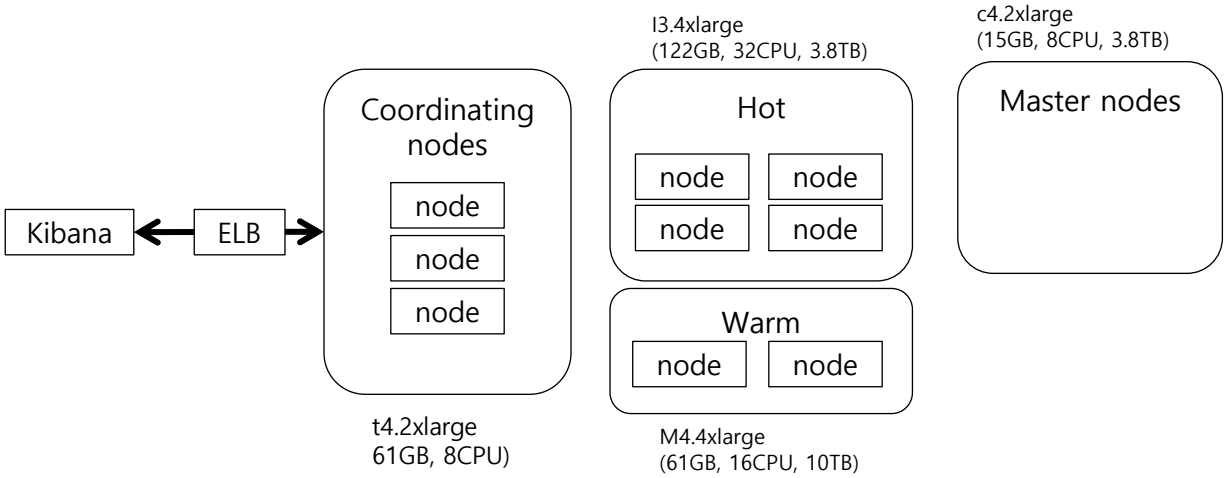
Peak Index Volume

912,739,256 documents/hr (1.1T)

Peak Search Volume

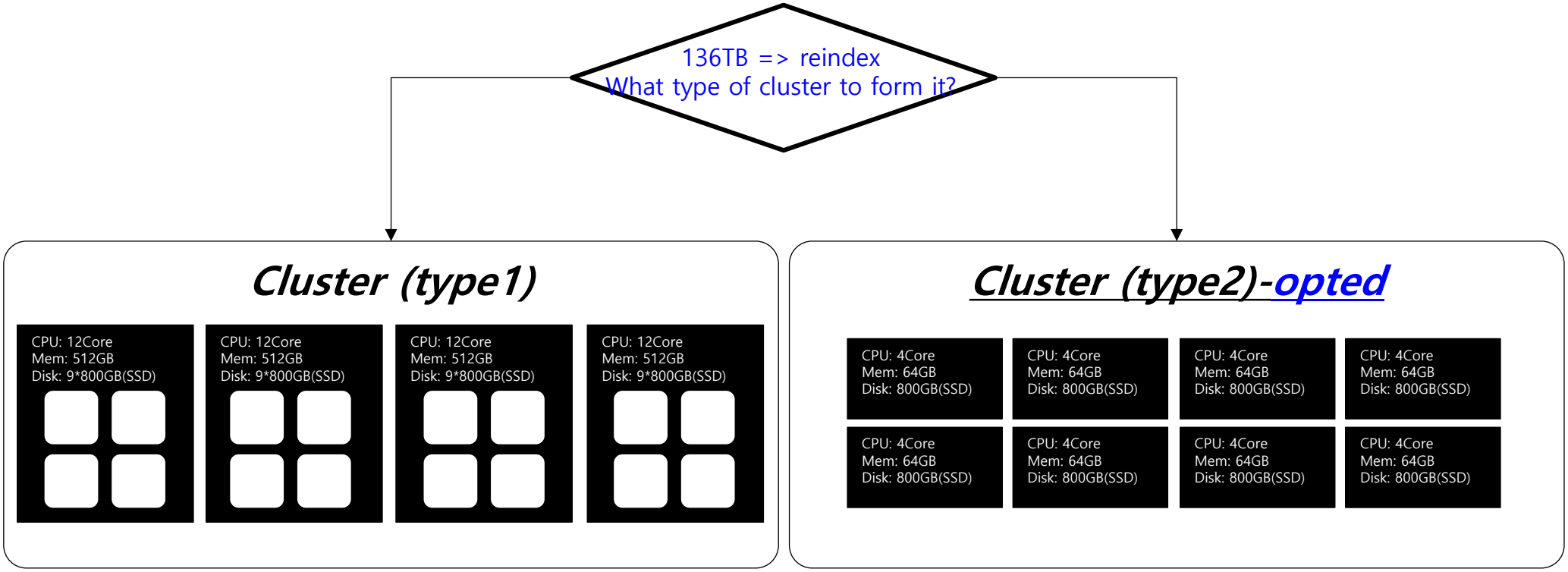
1,500 queries/s

Reliability is critical



Use cases

2) Synthesio



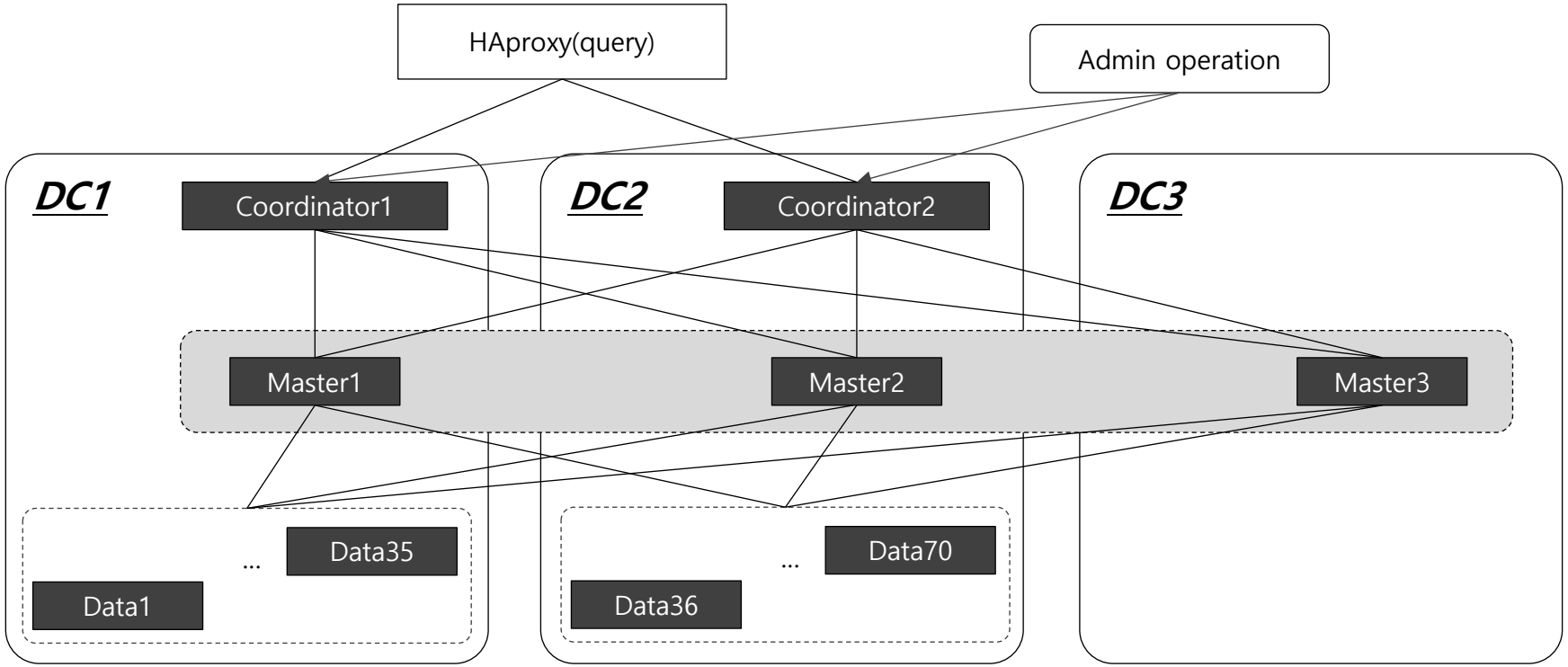
Scalable much easier
75 physical machines are running

Use cases

Deployment architecture at Synthesio

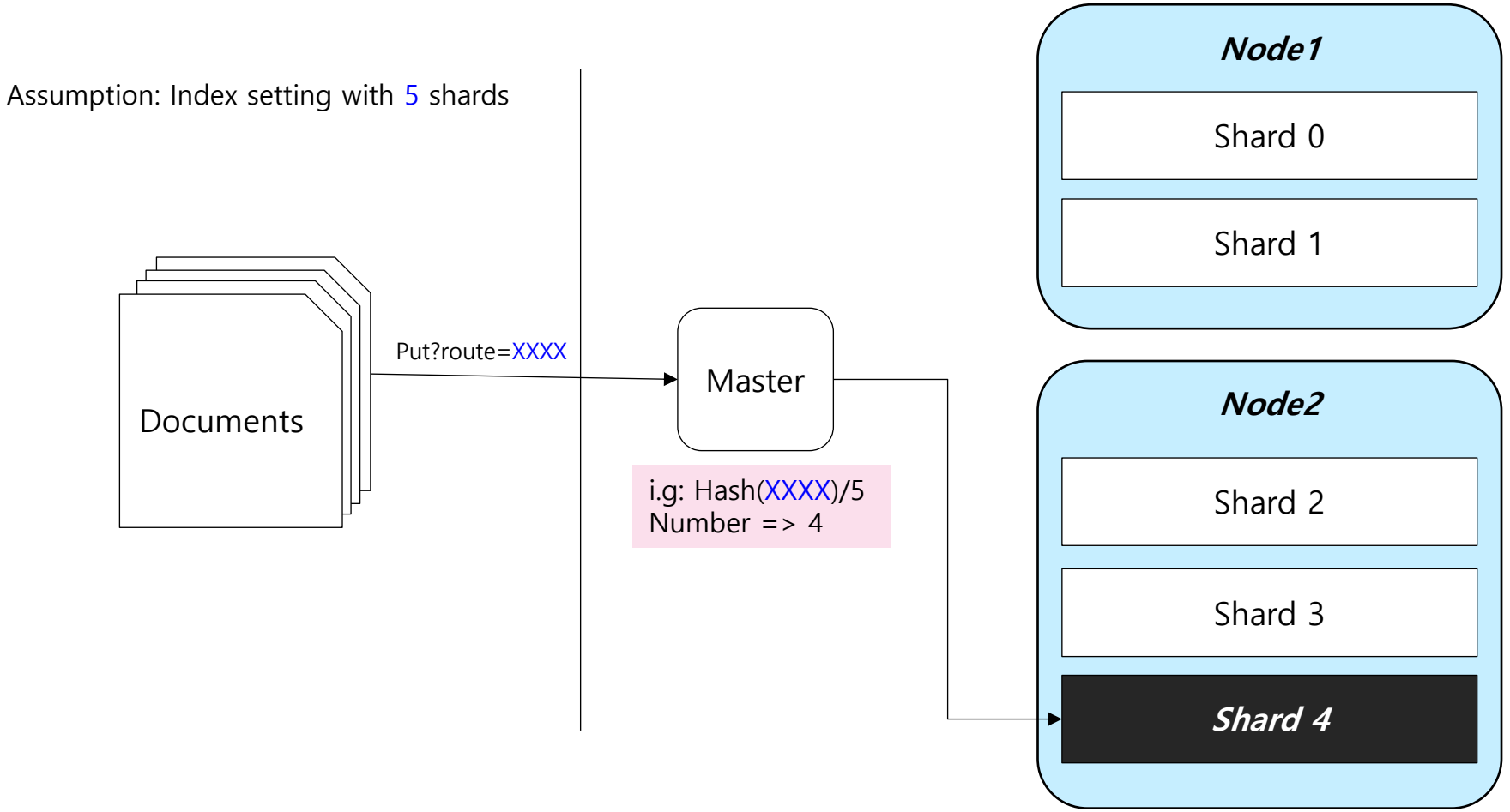
CPU: Quad Xeon D-1521 2.40GHz
Mem: 64GB
Disk: 4*800GB(SSD) – RAID-0
Filesystem: XFS

Total Storage Size: 218,75TB
4,68TB Memory
(2.39 TB Heap memory)

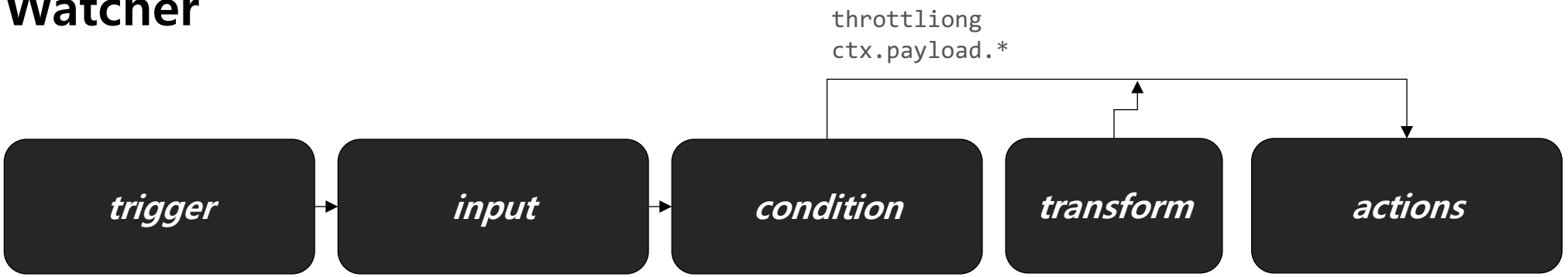


What is Routing?

Routing = which shard? = hash(document id) % number_of_primary_shards



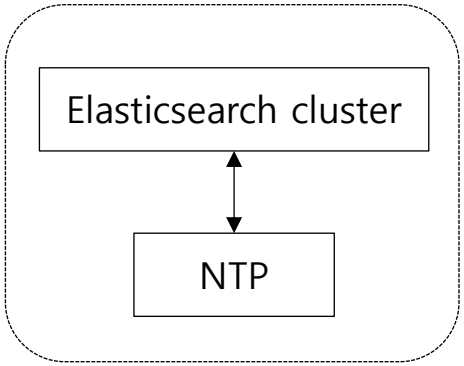
Watcher



- schedule
  - Hourly
  - Daily
  - Weekly
  - Monthly
  - Yearly
  - Cron
  - interval
- simple
- search
- http
- chain
- always
- never
- compare
- script

https://www.elastic.co/guide/en/x-pack/6.2/condition-script.html

- email
- webhook
- index
- logging
- hipchat
- slack
- jira
- etc



Roll~ & Scroll

Rollover	
RollUP	
Scroll	Can be used to retrieve large numbers of results from a single search request. (i.g *Cursor* on a traditional database) Note, use it carefully in real time user requests. Usecase: reindex

Shard Allocation

Shard awareness
Shard filtering
Shard overallocation. (when cluster grows…then it scale out without down time.)

Index allocation

I'm an index
Logs-2018-07

Put logs-2018-07
{
"settings": {***Index.routing.allocation***.require.node_type": "**hot**"}
}
→Dynamic setting

Put logs-2018-07/_settings
{
Index.routing.allocation.require.node_type": "**cold**"
}
What this means? Move index from hot to cold

※ if the condition didn't meet, then shard allocation does not work, which was planned.

Tag Nodes
Elasitcsearch.yml

Node1
node.attr.rack=rack1
node.attr.size=big
Node.attr.node_type=**hot**

hot: indexing only
h/w: ssd

Node2
node.attr.rack=rack2
node.attr.size=mid
Node.attr.node_type=**warm**

warm: read only
h/w: spindle

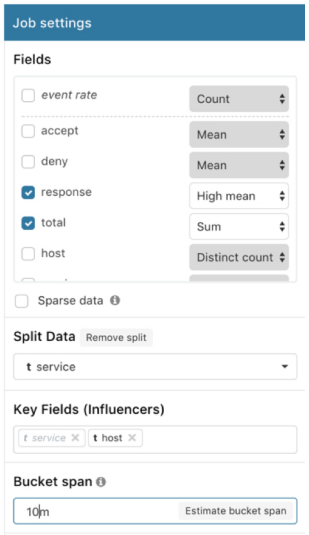
Node3
node.attr.rack=rack3
node.attr.size=small
Node.attr.node_type=**cold**

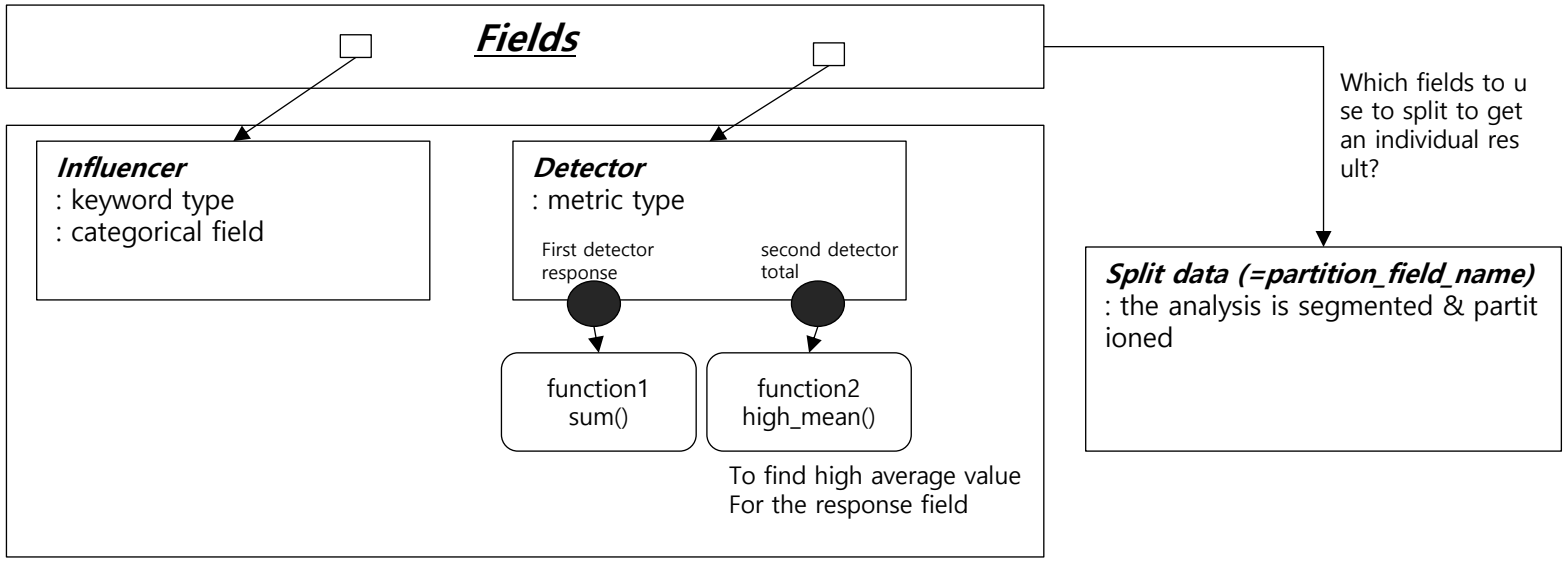
cold: just store

? Shrink
split

ML

	Field type	
Metric	long, float	detector
Field	text	influencer
Field_name		detector



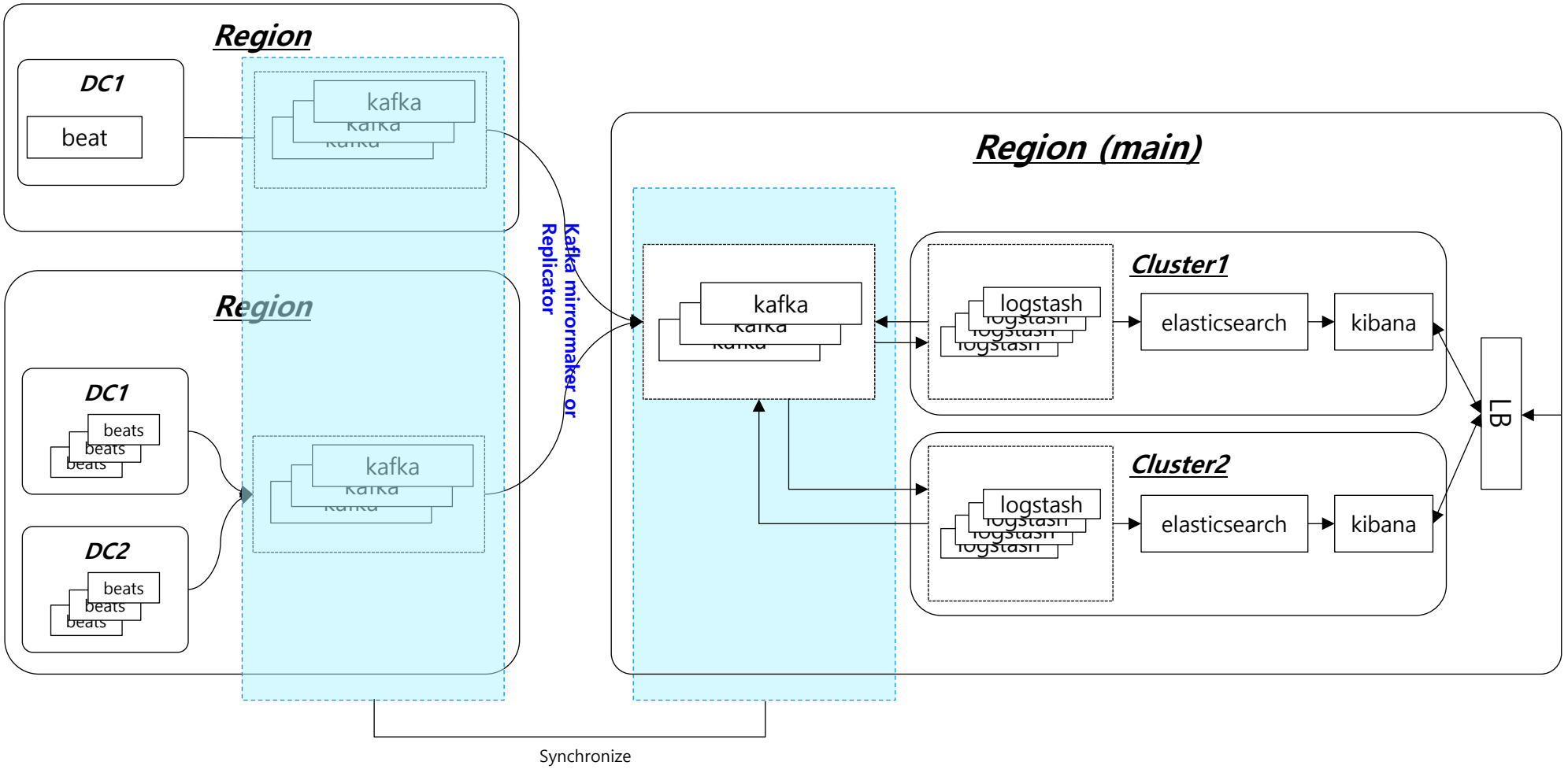


Tunning

memory	Give memory to the filesystem cache
Faster hardware	Disk (SSD) Local storage EBS? Then, use provisioned IOPS
Document modeling	Documents should be modeled so that search-time operations are as cheap as possible. Avoid joins, nested several time slower, parent-child: hundreds of times slower Denormalizing documents
Search as few fields as possible	The more fields, toe slower it is Search for multi-fields? Then, use copy_to at index time (consume more space)
Mappings	Sometimes…mapping filed as keyword like ISBN instead integer and long
Pre-index Data	
Avoid scripts	To use script…then painless and expressions engine.
Search rounded dates	https://www.elastic.co/guide/en/elasticsearch/reference/6.3/tune-for-search-speed.html#_document_modeling
Force-merge	Read only indices (i.g: time-based indices to a single segment) Do not force merge indices that aare still being written to – leave merging to bg merge process
Warm up global ordinals	
Warm up filesystem cache	
Map identifiers as keyword	Better performant range query while keyword fields are better at term query
Cache utilization	
Replicas might help with throug hput	Fewer shard…reuse cache highly

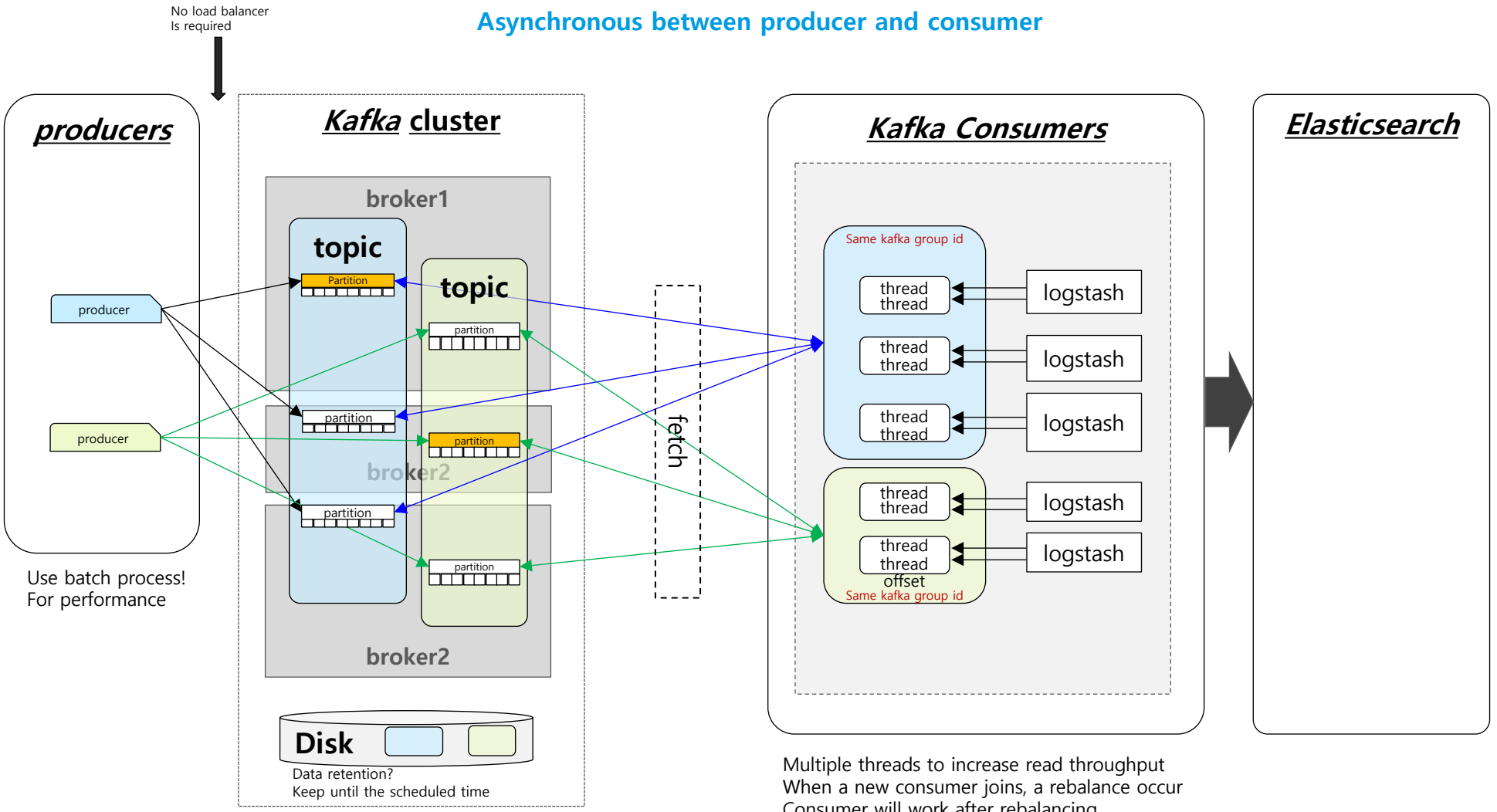
Architecture Diagram

DC to DC



Architecture Diagram

With Big Queue



topic	Logical concept for grouping of partitions
partition	Data store, scalable horizontally

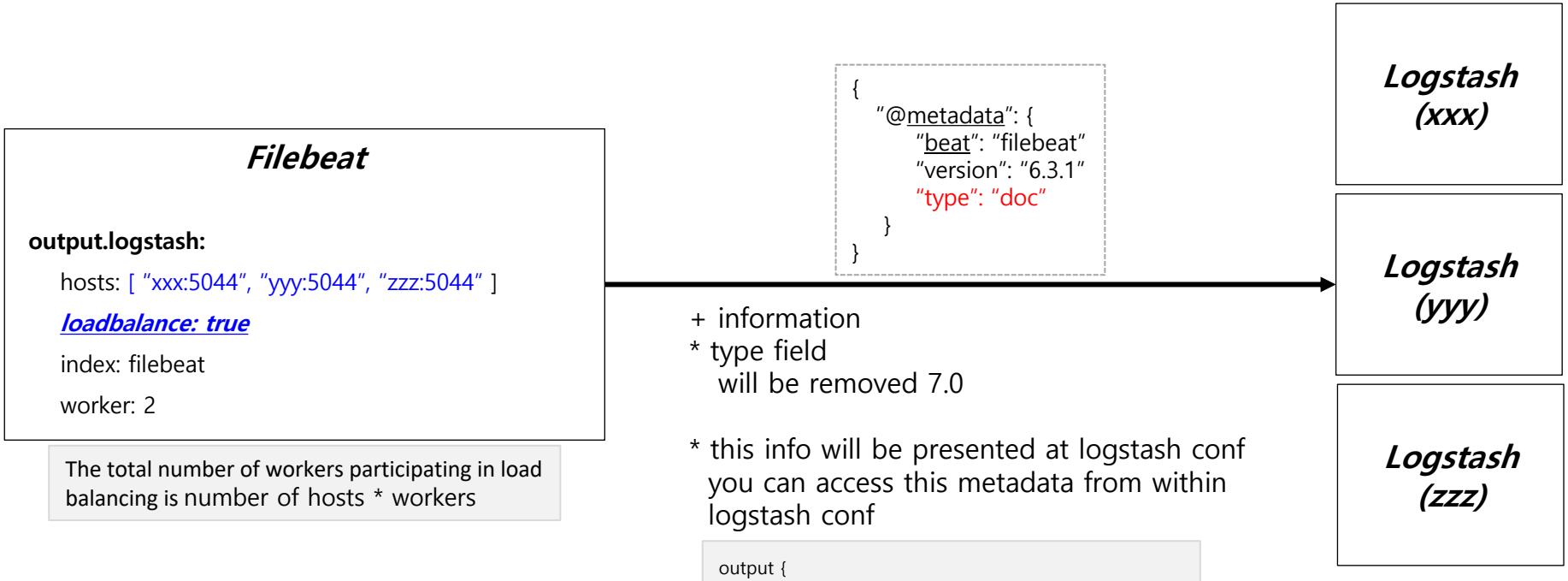
Architecture Diagram

With Hot-Warm

Architecture Diagram

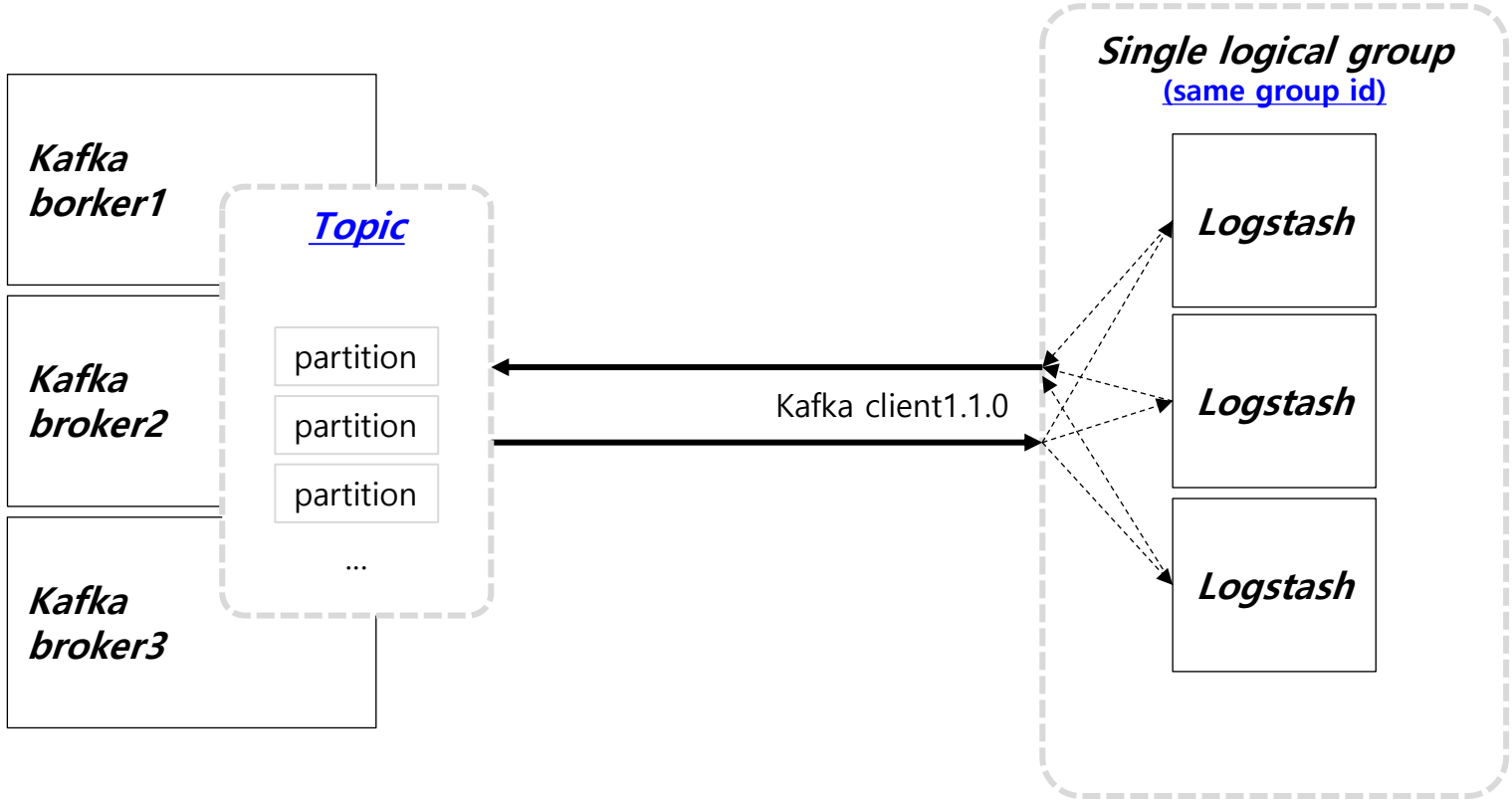
Architecture Diagram

Redis, Logstash, and Elasticsearch outputs. The Kafka output handles load balancing internally.

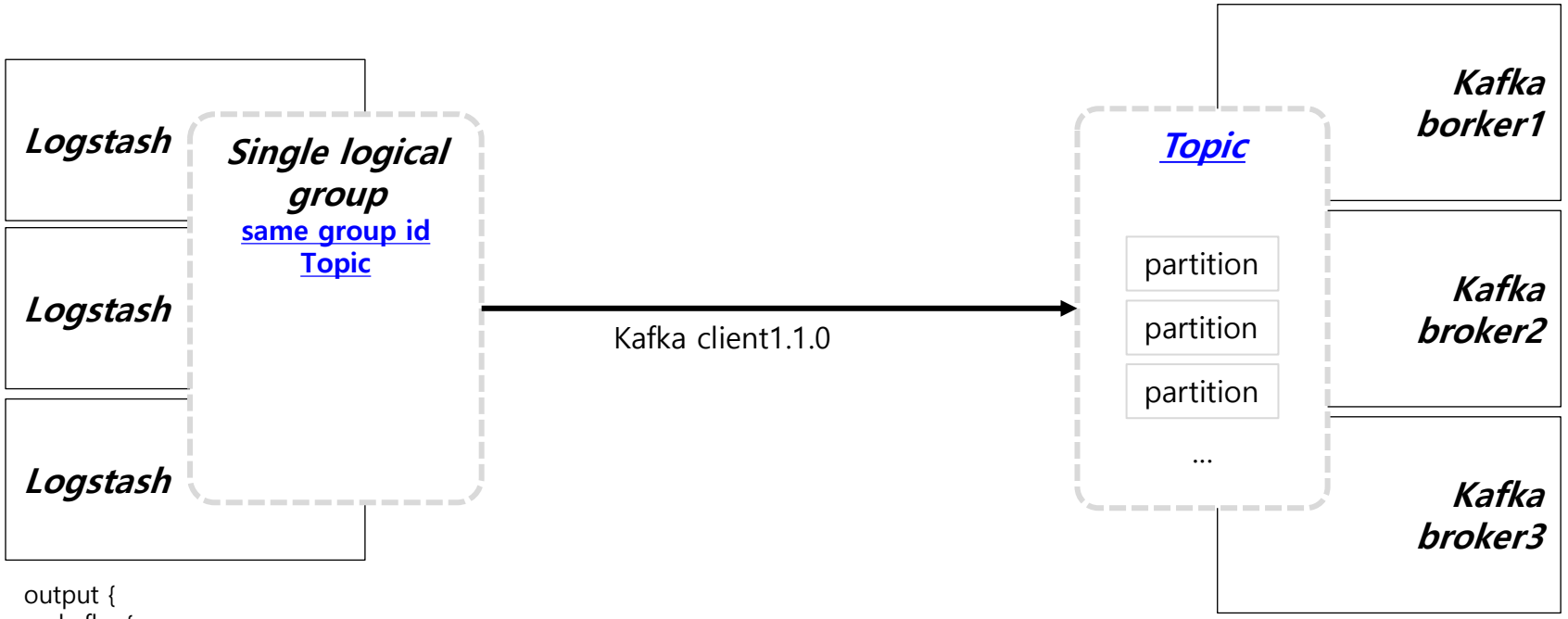


bulk_max_size: maximum numb
er of events to bulk in a single
logstash request

Architecture Diagram



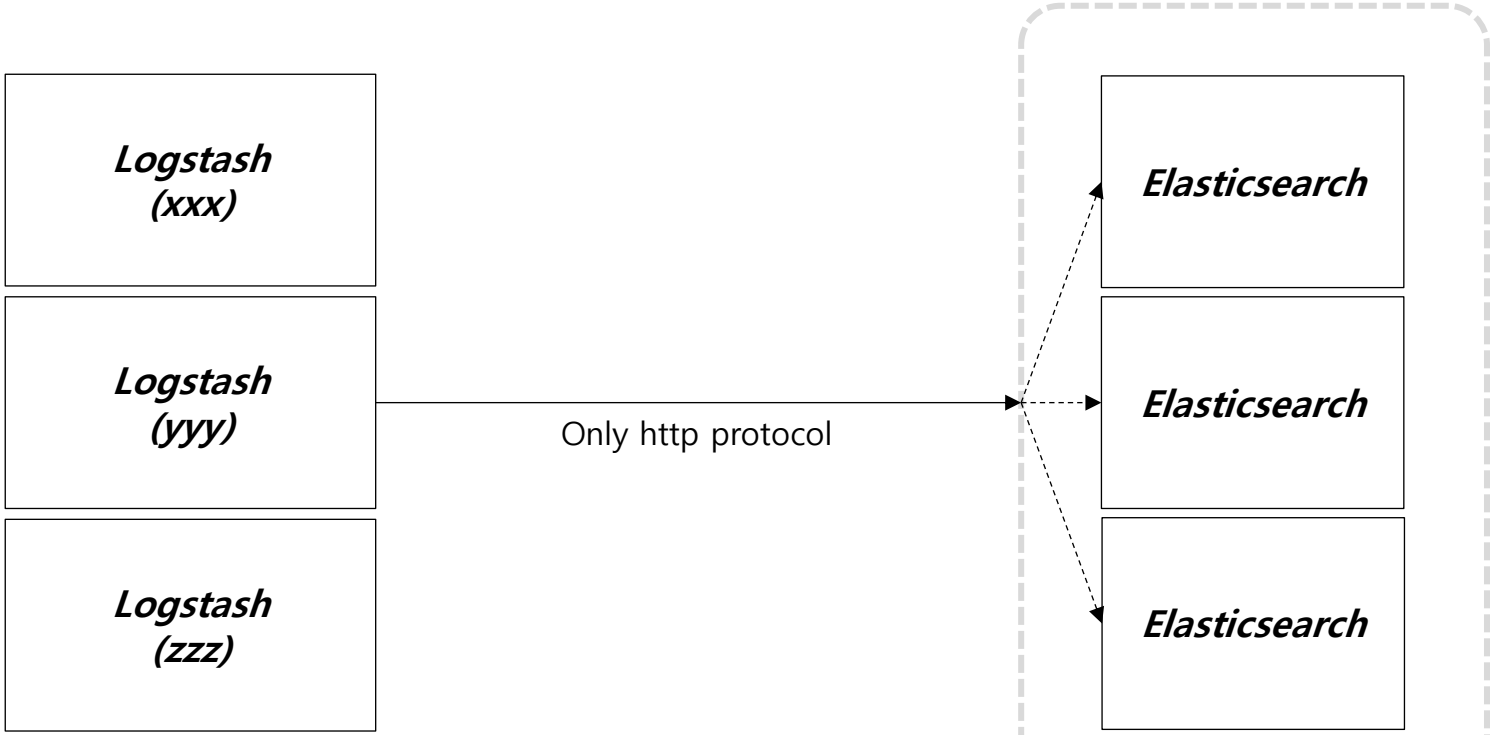
Architecture Diagram




```
output {
    kafka {
        codec => json. (* want the full content of you events to be sent as json?)
        topic_id => "my_topic"
        bootstrap_servers => [ "broker1:9092", "broker2:9092", "broker3:9092" ]
        retries => [do not set]
    }
}
```

Broker's list or VIP

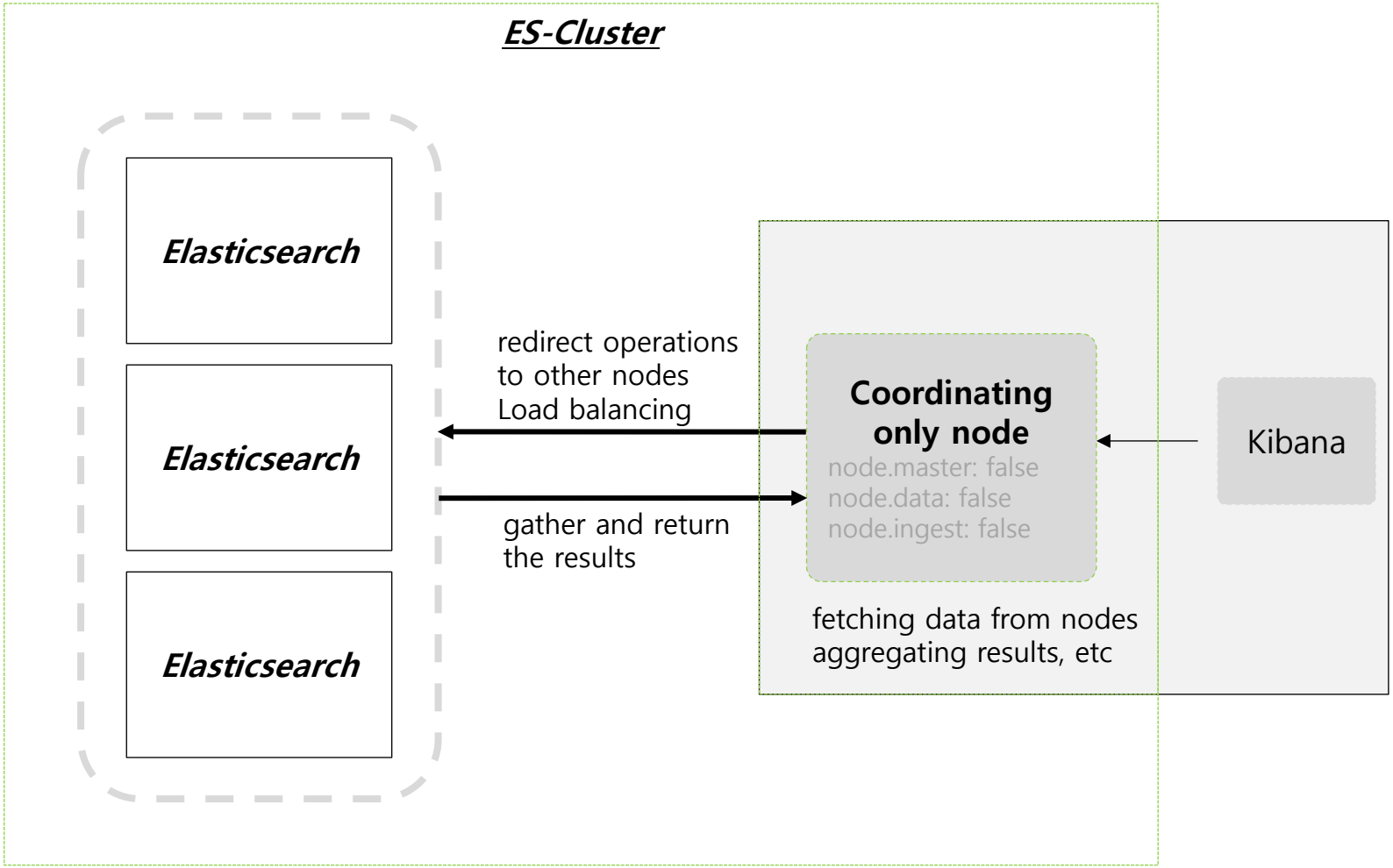
Architecture Diagram




```
output {
    elasticsearch {
        id => "my_id"
        script => "ctx._source.message = params.event.get('message')"
        hosts => [xxx:9200, yyy:9200, zzz:9200]
    }
}
```

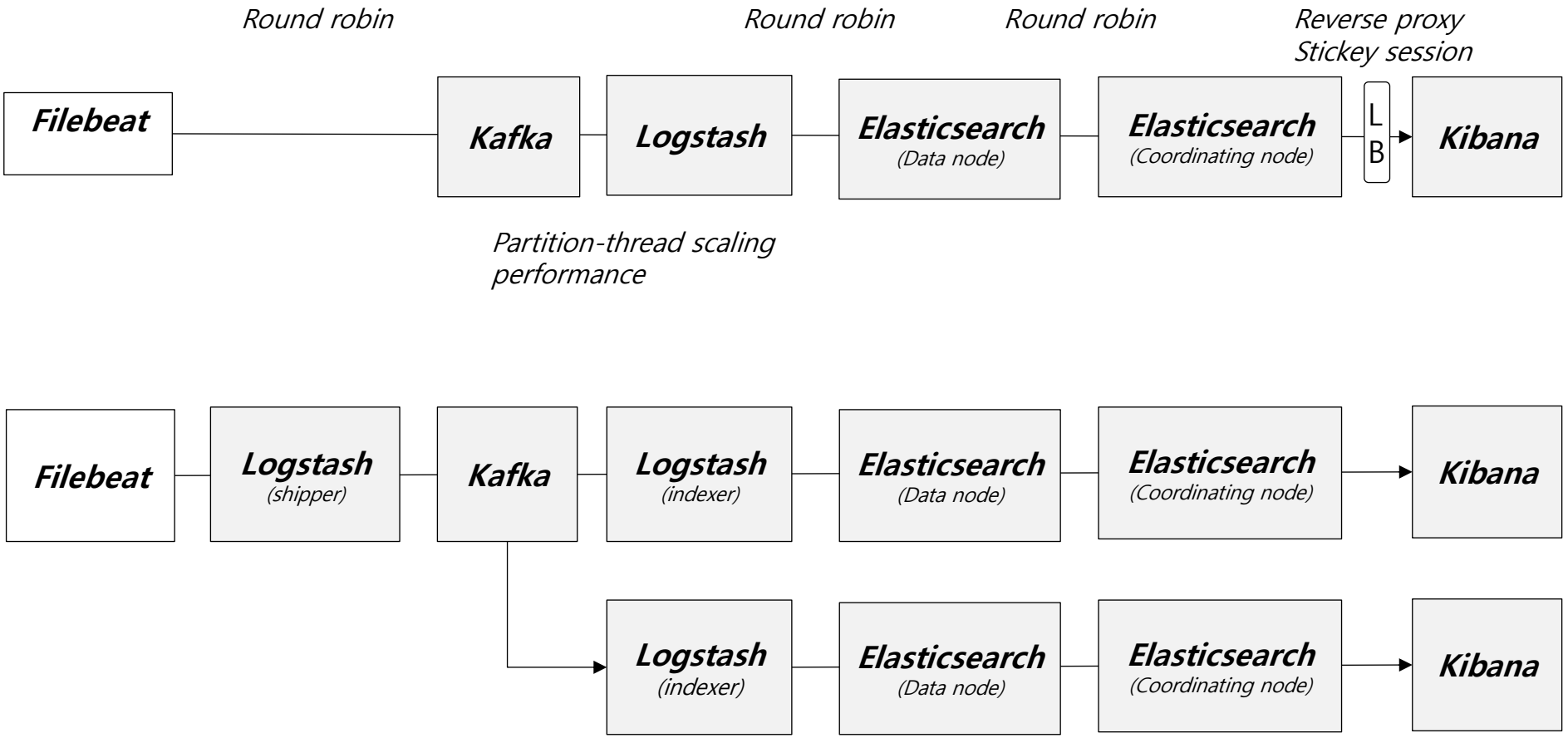
Load balance requests

Architecture Diagram



Architecture Diagram

End to End Pipeline



Queue of Logstash