ElasticSearch
Under The Hood

What is Big Data ?

And how do you manage it ?

What are Distributed Systems ?

What are MPP systems ?

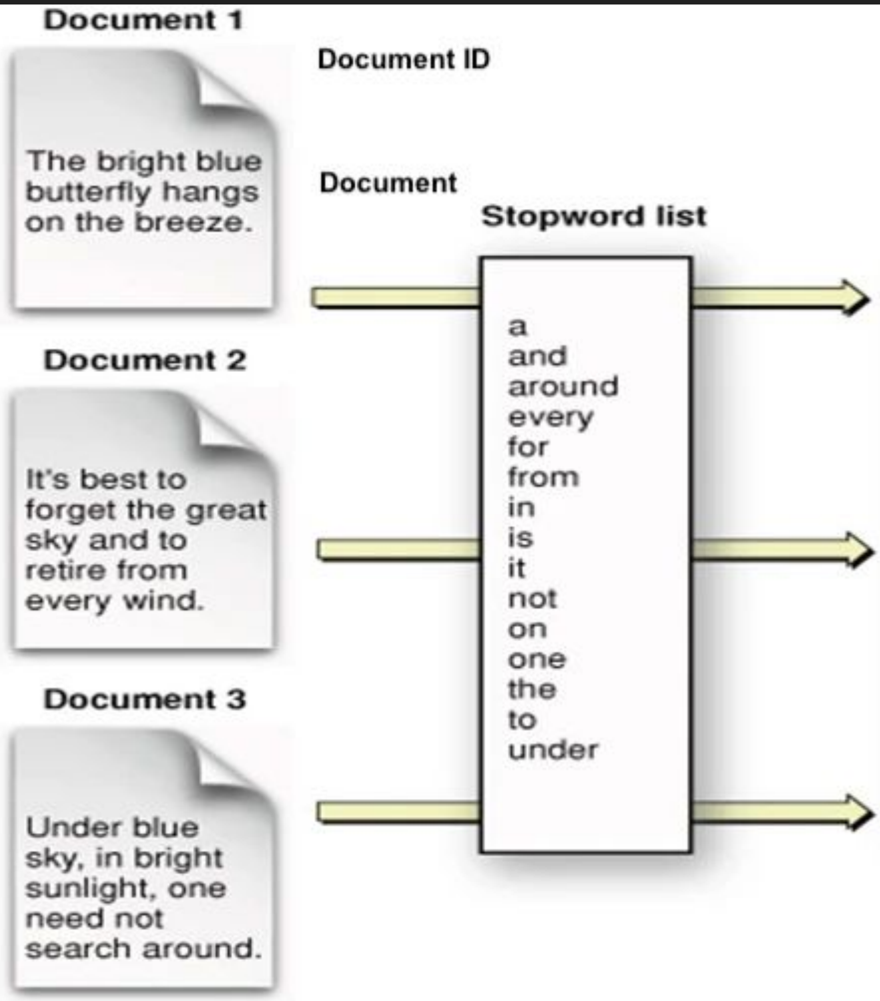
Monolithic Systems

- Shared Memory Architecture

Distributed Systems

- Shared Disk Architecture
- Shared Nothing Architecture

Inverted Index



Inverted index

ID	Term	Document
1	best	2
2	blue	1, 3
3	bright	1, 3
4	butterfly	1
5	breeze	1
6	forget	2
7	great	2
8	hangs	1
9	need	3
10	retire	2
11	search	3
12	sky	2, 3
13	wind	2

Term Ordinal — **Terms Dictionary** — **Postings List**

What is Lucene ?

Horizontally Scalable

Big Data

Shared Nothing Architecture

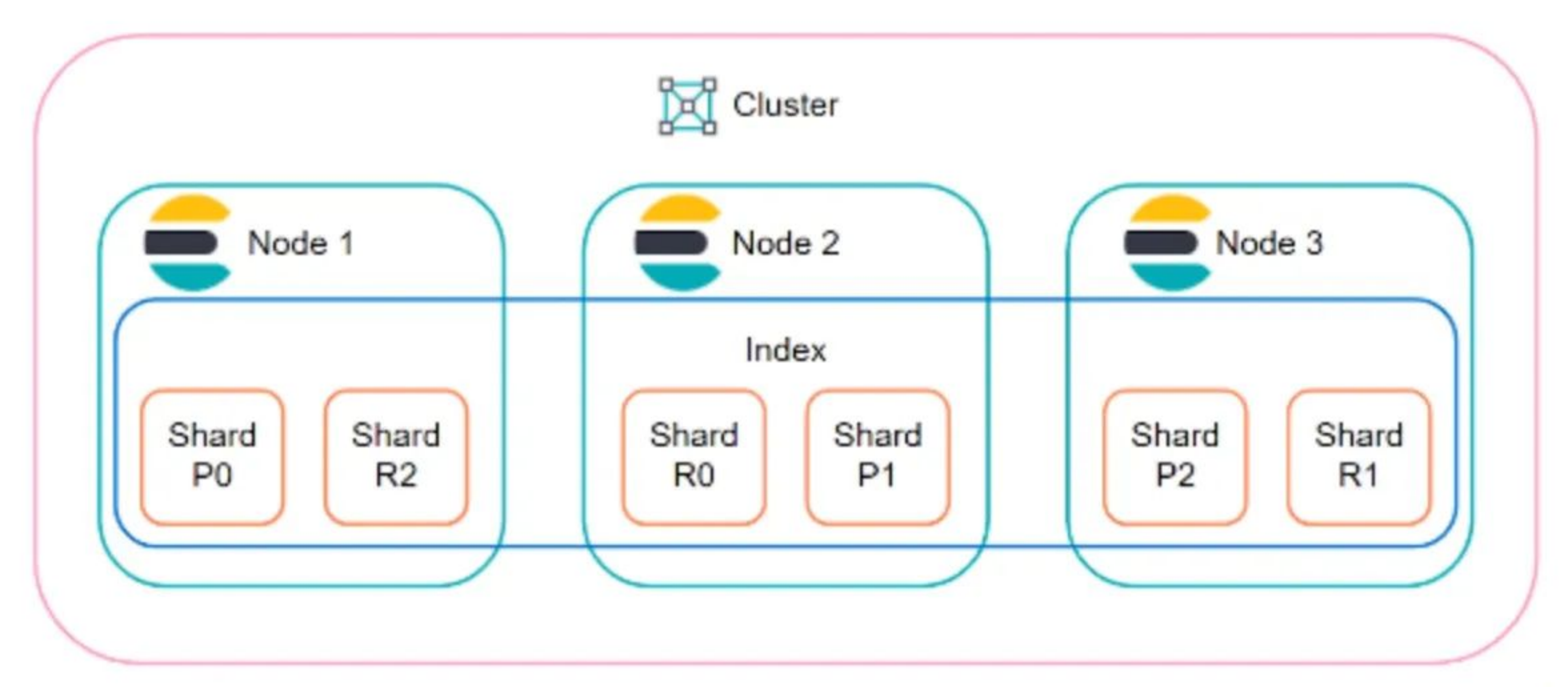
What is ElasticSearch ?

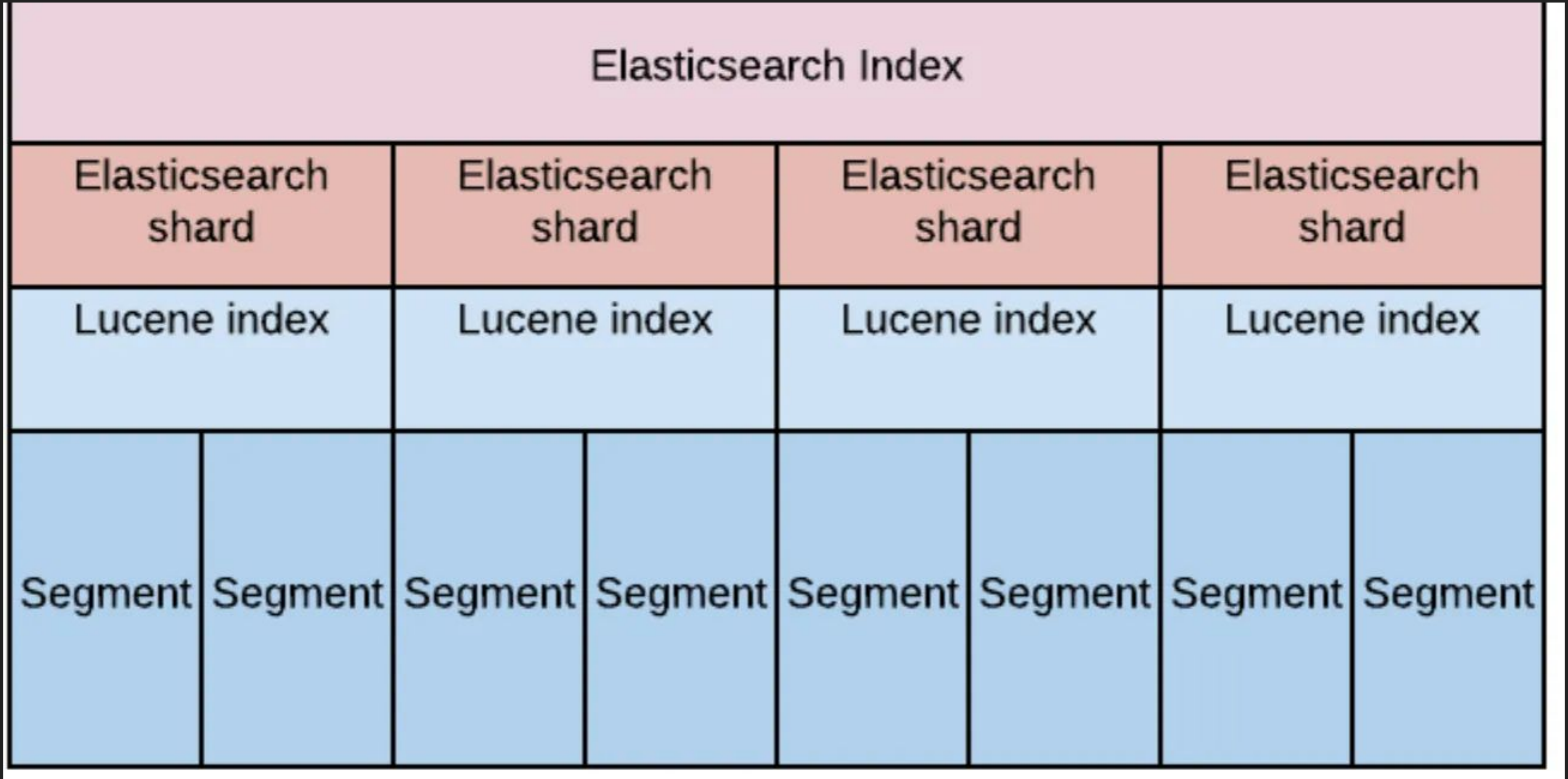
Search Engine

MPP

Distributed Database

Basic Data Spread Overview




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

We’re starting the bottom up

We will come back to ElasticSearch

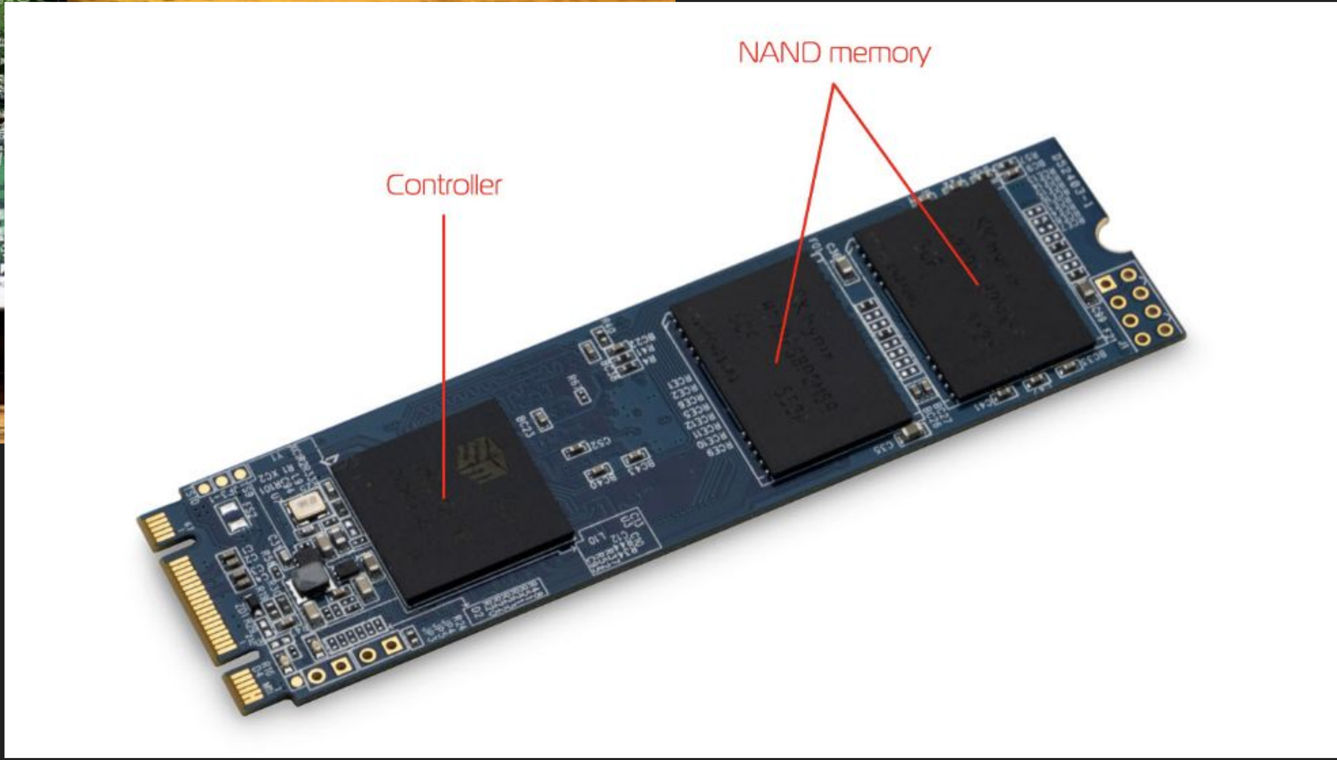
The Challenge Of Reliable Storage

Disks (both HDD/SSD)

The Challenge Of Reliable Storage

- Disks have **Disk Controllers**


K2FYA
C2FYA
15311020


NAND memory
Controller

The Challenge Of Reliable Storage

- Disks have **Disk Controllers** and they have **Caches**

The Challenge Of Reliable Storage

- Disks have **Disk Controllers** and they have **Caches INSIDE OF THEM**

The Challenge Of Reliable Storage

- Disks have **Disk Controllers** and they have **Caches INSIDE OF THEM**
- Disk IOPS Are Slow

```
Latency Comparison Numbers (~2012)
----------------------------------
L1 cache reference                           0.5 ns
Branch mispredict                            5   ns
L2 cache reference                           7   ns                      14x L1 cache
Mutex lock/unlock                           25   ns
Main memory reference                      100   ns                      20x L2 cache, 200x L1 cache
Compress 1K bytes with Zippy             3,000   ns        3 us
Send 1K bytes over 1 Gbps network       10,000   ns       10 us
Read 4K randomly from SSD*             150,000   ns      150 us          ~1GB/sec SSD
Read 1 MB sequentially from memory     250,000   ns      250 us
Round trip within same datacenter      500,000   ns      500 us
Read 1 MB sequentially from SSD*     1,000,000   ns    1,000 us    1 ms  ~1GB/sec SSD, 4X memory
Disk seek                           10,000,000   ns   10,000 us   10 ms  20x datacenter roundtrip
Read 1 MB sequentially from disk    20,000,000   ns   20,000 us   20 ms  80x memory, 20X SSD
Send packet CA->Netherlands->CA    150,000,000   ns  150,000 us  150 ms

Notes
-----
1 ns = 10^-9 seconds
1 us = 10^-6 seconds = 1,000 ns
1 ms = 10^-3 seconds = 1,000 us = 1,000,000 ns

Credit
------
By Jeff Dean:               http://research.google.com/people/jeff/
Originally by Peter Norvig: http://norvig.com/21-days.html#answers
```

The Challenge Of Reliable Storage

- Have **Disk Controllers** and they have **Caches INSIDE OF THEM**
- Disk IOPS Are Slow
- Most databases rely on filesystems and not use disks directly

The Challenge Of Reliable Storage

- Have **Disk Controllers** and they have **Caches INSIDE OF THEM**
- Disk IOPS Are Slow
- Most databases rely on filesystems and not use disks directly

The Challenge Of Reliable Storage

- Have **Disk Controllers** and they have **Caches INSIDE OF THEM**
- Disk IOPS Are Slow
- Most databases rely on filesystems and not use disks directly
- And filesystems have their own metadata to maintain

```
#touch file.txt
#
#stat file.txt
  File: file.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 10303h/66307d   Inode: 16961354    Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/ec2-user)   Gid: ( 1000/ec2-user)
Access: 2024-07-13 05:22:35.308253378 +0530
Modify: 2024-07-13 05:22:35.308253378 +0530
Change: 2024-07-13 05:22:35.308253378 +0530
 Birth: 2024-07-13 05:22:35.308253378 +0530
#
```

Which Means..

More Writes..

More Writes..

Per Unit of Data..

The Challenge Of Reliable Storage

- Disks have **Disk Controllers** and they have **Caches INSIDE OF THEM**
- Disk IOPS Are Slow
- Most databases rely on filesystems and not use disks directly
- And filesystems have their own metadata to maintain
- These filesystems are mounted in operating systems
- And operating systems have **THEIR OWN Page Cache**

—------------ Databases
—------------ Operating System (with it’s cache)
—------------ Filesystem
—------------ Disks (with it’s Caches) (outside the control of OS)

Cache Writing Policies

- Write-Back
- Write-Through

Linux WriteBack Mechanism

```
# ps -ef | grep writeback
root         25     2  0 Jun28 ?        00:00:00 [writeback]
```

Linux Background Kernel Thread

How does a database guarantee a write ?

Linux Write-Through Mechanism

fsync() and fdatasync()

Linux Syscalls

How does it help in guaranteeing a write ?

fsync() and fdatasync()

CRUD

C - Create
R - Read
U - Update
D - Delete

Databases need to handle a lot of CUD..

And each CUD can result in multiple writes..

How ?

If fsync() and fdatasync() are synchronous operations, and disk writes take long

Then how do databases perform fast with the high CUD operations

Without compromising on efficiency and reliability.. ?

Write Ahead Log

Different storage solutions refer WAL by different names

- Write Ahead Log (WAL) in Postgresql in Ceph SDS, RabbitMQ
- Binlog in MySQL
- Transaction Log in ElasticSearch, MongoDB, Neo4j
- Journals in most filesystems (xfs, ext4)
- Append Only File (AOF) in Redis
- Commit Log in Cassandra
- Redo Log in Oracle

Problems solved for CUD

fsync() and fdatasync() solved reliability

WAL solved efficiency in writes

But what about efficiently organising writes ?

Let’s talk about storing the data on disk for efficient reads..

The common ways data is organised

LSM

B-Tree

A word about indexing

Back To ElasticSearch..

Storage Layers in ElasticSearch

- Segment Files
- Shards
- Index
- Alias
- Data Streams

Components in the ElasticSearch storage

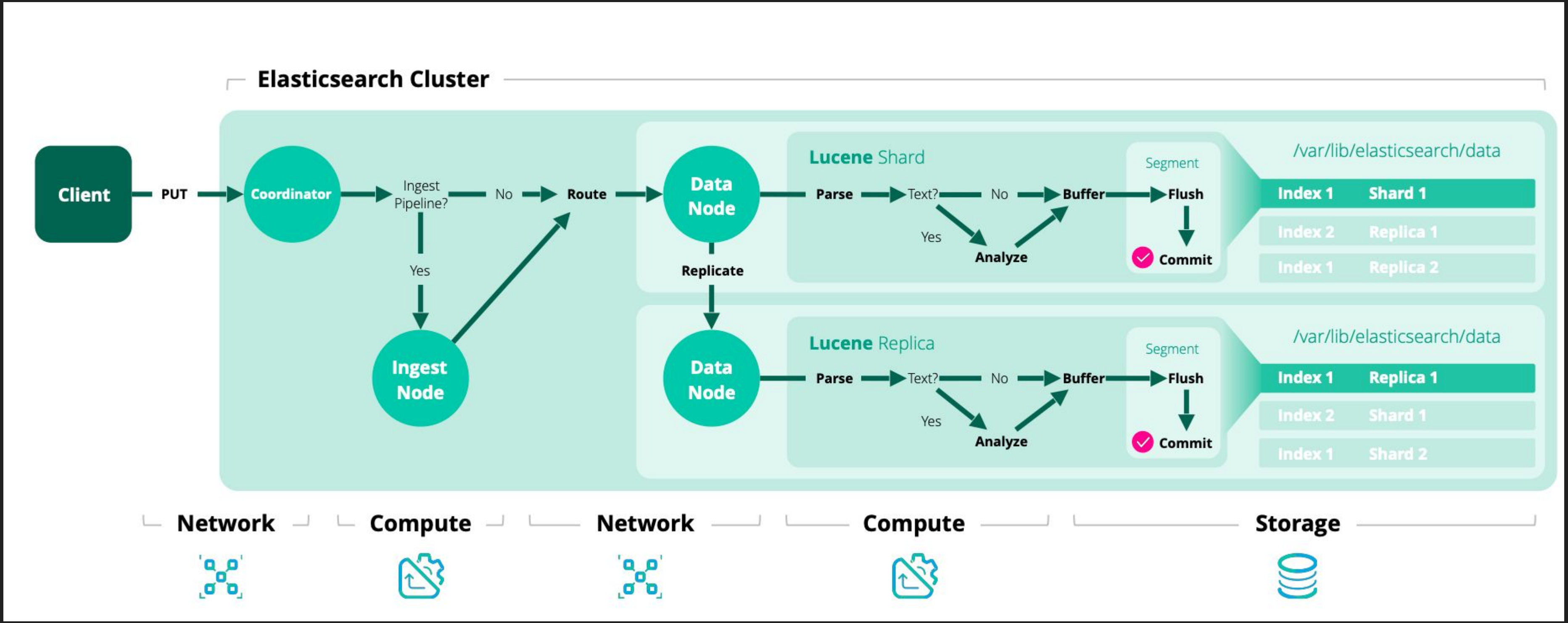
- In-Memory Buffer
- Segment Files

1. Committed Segment Files
2. Searchable Segment Files

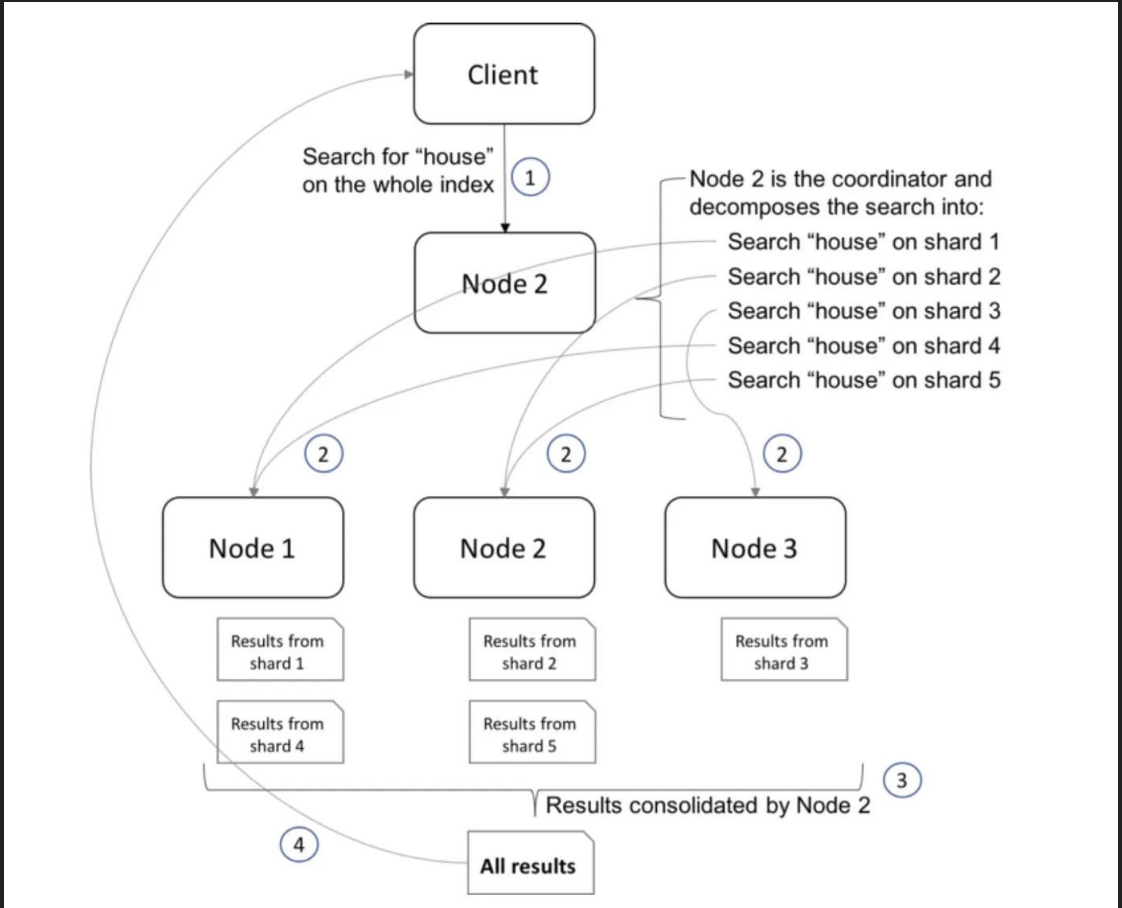
- Segment Files Merge
- Tombstones

Data Flow Overview

Write Path Overview



Read Path Overview



Write Path Encounters

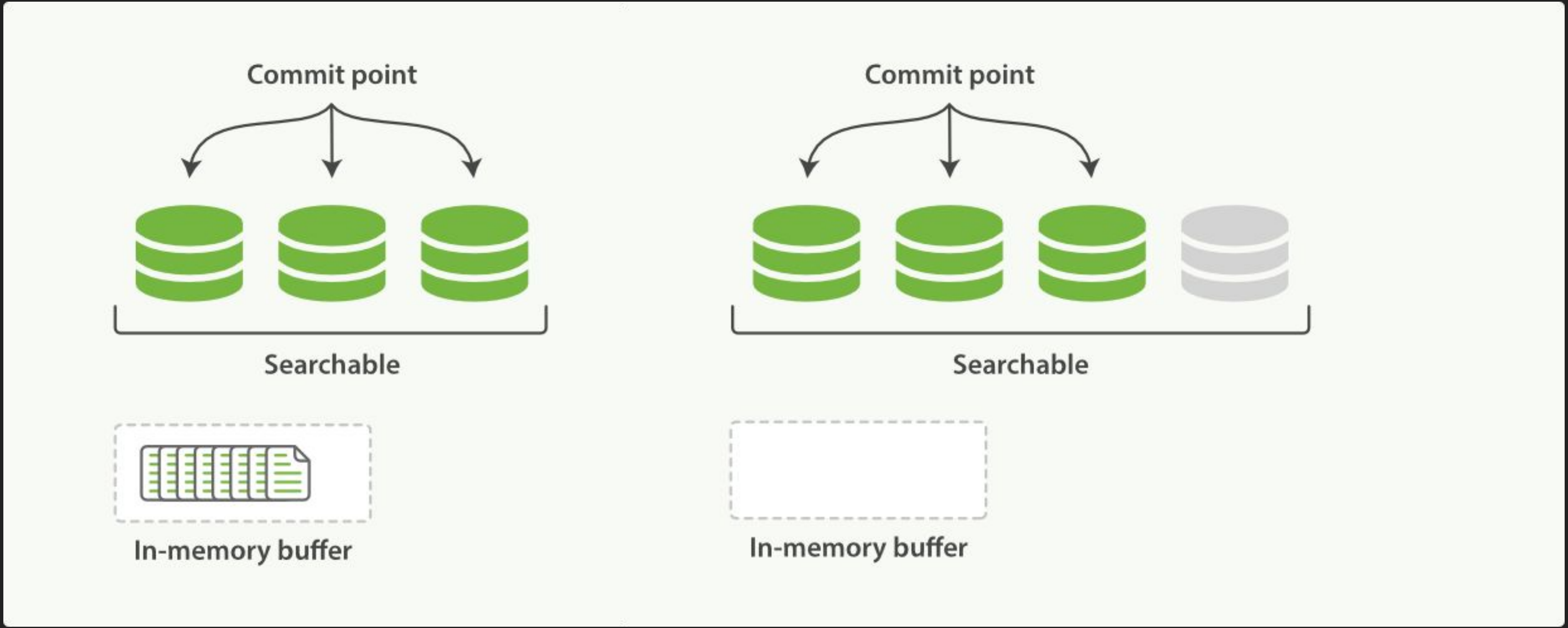
Refresh

- Can be thought of as a lightweight fsync
- Happens every second by default in ElasticSearch
- Flushes the in-memory buffers to create an in-memory segment and allow it to be searched

Flush

- An actual fsync to merge the segments and flush the merged segments on to disk

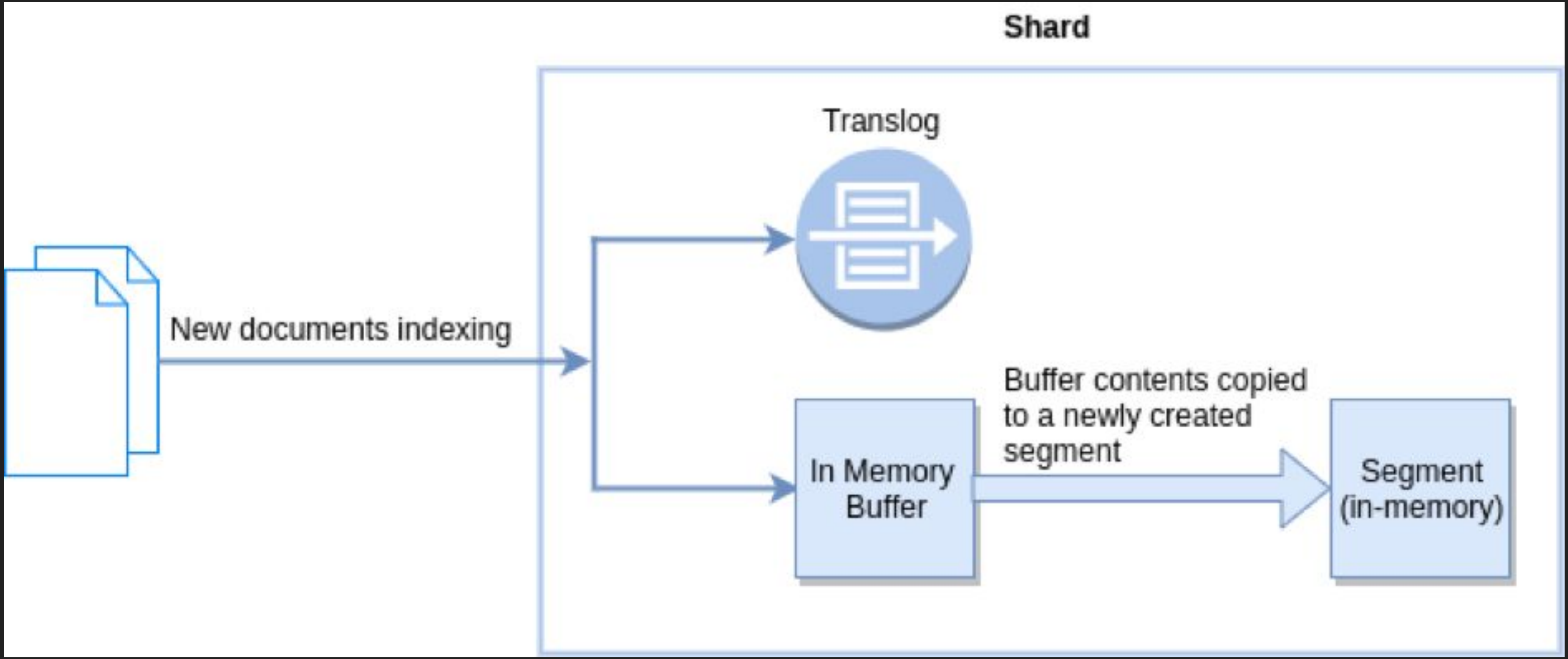
Refresh And Flush



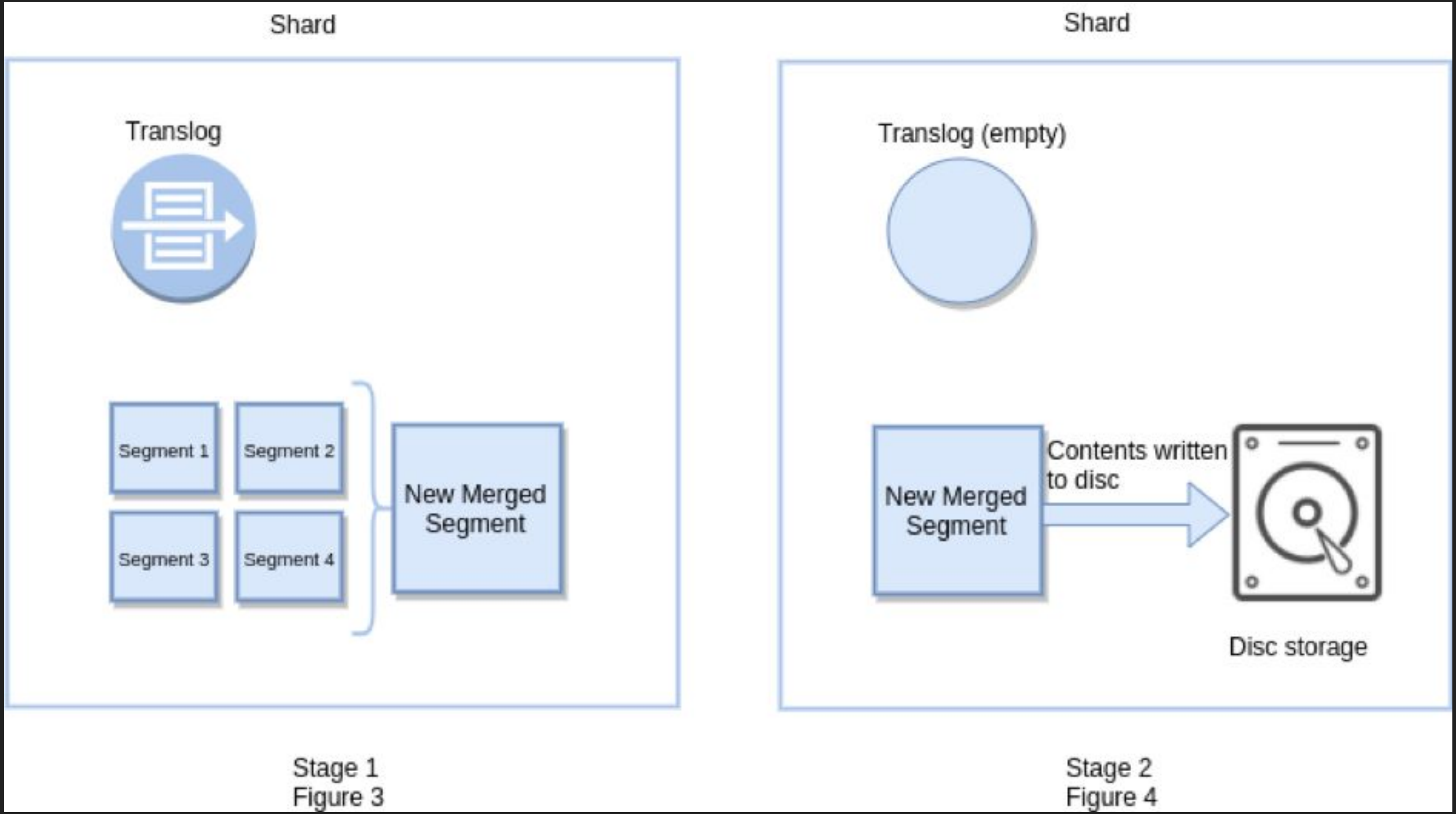
Yet Another Diagrams to Explain

Refresh And Flush

What Refresh Does



What Flush Does



Index and Indexing are not the same..

Understand the context..

Types of Indexing a field can have

- Text (for full-text search)
- Keyword (for exact match)

Understanding how the R in the CRUD

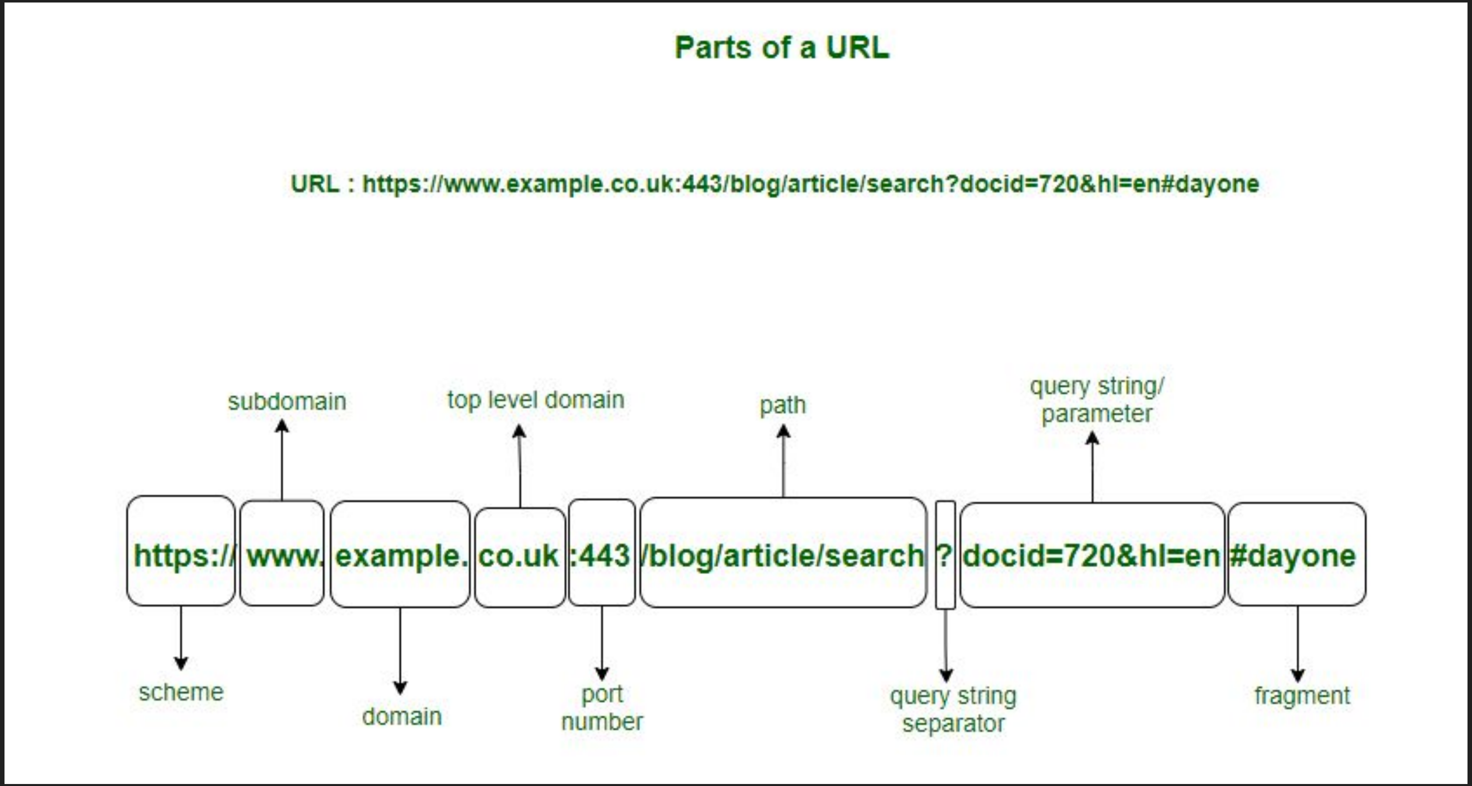
Works in ElasticSearch

The power of vocalisation

```
                QUERIES                  ROUTES                        DIVIDES              WRITES / READS
ES Client   ------------------->  Alias  ----------------->  RW / R Index  ------------------> SHARDS ------------------------> SEGMENTS
```

Gossip Protocol

Structure of a URL



Examples of a few HTTP APIs

GET /_cat/indices

```
# curl https://192.168.1.10:9200/_cat/indices?v
health status index                            uuid                   pri rep docs.count docs.deleted store.size pri.store.size
green  open   biz.live.data.2024.04.04-000024 fCmpWN2ySt6zZisx9Qj0vA   2   1          7            0    258.5kb        258.5kb
green  open   biz.live.data.2024.05.04-000025 Zo27FF_VT5-WqiOmBa-DsA   2   1          9            0    293.7kb        293.7kb
green  open   biz.live.data.2024.06.03-000026 IIq6DbwZTbKdRg3g9IqMcw   2   1         44            0    582.8kb        582.8kb
green  open   biz.live.data.2024.07.03-000027 WX3GgeLUQ_qwzPzCZQEgRg   2   1         19            0    949.8kb        474.9kb
```

GET /_cat/aliases

```
# curl https://192.168.1.10:9200/_cat/aliases?v&s=alias,index
alias              index                             filter routing.index routing.search is_write_index
biz.live.data      biz.live.data.2024.04.04-000024   -      -             -              false
biz.live.data      biz.live.data.2024.05.04-000025   -      -             -              false
biz.live.data      biz.live.data.2024.06.03-000026   -      -             -              false
biz.live.data      biz.live.data.2024.07.03-000027   -      -             -              true
```

GET /_cat/nodes

```
# curl https://192.168.1.10:9200/_cat/nodes?v
ip           heap.percent ram.percent cpu load_1m load_5m load_15m node.role master name
192.168.1.10           57          93  12    1.18    1.12     1.01 mdi       *      node-1
192.168.1.11           53          94  7     0.47    0.69     0.77 mdi       -      node-2
192.168.1.12           54          91  5     0.16    0.27     0.35 mdi       -      node-3
192.168.1.13           55          90  7     0.34    0.21     0.27 di        -      node-4
192.168.1.14           52          90  6     0.74    0.74     0.42 di        -      node-5
```

GET /_cat/allocation

```
# curl http://192.168.1.10:9200/_cat/allocation?v
shards disk.indices disk.used disk.avail disk.total disk.percent host         ip           node
  1063        118gb   120.2gb    179.7gb        1tb           83 192.168.1.10 192.168.1.10 node-1
  1061      132.9gb   136.3gb    163.5gb        1tb           85 192.168.1.11 192.168.1.11 node-2
  1061      153.6gb   157.4gb    142.4gb        1tb           87 192.168.1.12 192.168.1.12 node-3
  1063      122.6gb   127.1gb    172.7gb        1tb           84 192.168.1.13 192.168.1.13 node-4
  1063      199.8gb   103.2gb    196.6gb        1tb           82 192.168.1.14 192.168.1.14 node-5
```

GET /_cluster/health

```
# curl http://192.168.1.10:9200/_cluster/health?pretty
{
  "cluster_name" : "es-cluster-1",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 5,
  "number_of_data_nodes" : 5,
  "active_primary_shards" : 5310,
  "active_shards" : 10622,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

Other helpful APIs in ElasticSearch

- GET /_cat/recovery?v&active_only=true
- GET /_cat/thread_pools?v
- GET /_cluster/settings?flat_settings=true&include_defaults=true
- GET /_cat/shards?v
- GET /_cat/segments?v

A word about managing data at large scale

In ElasticSearch

Tiered Storage, Searchable Snapshots
Index Priorities, ILM, SLM policies

Signoff

@GypsyCosmonaut

https://www.linkedin.com/in/gypsycosmonaut/