

# Scaling real time search and analytics with elasticsearch

**Shay Banon**  
@kimchy

# elasticsearch

# elasticsearch

- real-time

# elasticsearch

- real-time
- distributed

# elasticsearch

- real-time
- distributed
- search

# elasticsearch

- real-time
- distributed
- search
- analytics

# how to use it?

# how to use it?



# how does it work?

# step 1:

## making text searchable

Shall I compare thee to a summer's day?  
Thou art more lovely and more temperate:  
Rough winds do shake the darling buds of May,  
And summer's lease hath all too short a date:  
Sometime too hot the eye of heaven shines,  
And often is his gold complexion dimmed,  
And every fair from fair sometime declines,  
By chance, or nature's changing course untrimmed:  
But thy eternal summer shall not fade,  
Nor lose possession of that fair thou ow'st,  
Nor shall death brag thou wander'st in his shade,  
When in eternal lines to time thou grow'st,  
    So long as men can breathe, or eyes can see,  
    So long lives this, and this gives life to thee.

Shall I compare thee to a summer's day?  
Thou art more lovely and more temperate:  
Rough winds do shake the darling buds of May,  
And summer's lease hath all too short a date:

**where content like**  
**"%darling%buds%"**

Nor lose possession of that fair thou ow'st,  
Nor shall death brag thou wander'st in his shade,  
When in eternal lines to time thou grow'st,  
So long as men can breathe, or eyes can see,  
So long lives this, and this gives life to thee.

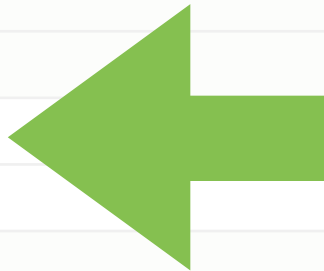
Shall I compare thee to a summer's day?  
Thou art more lovely and more temperate:  
Rough winds do shake the darling buds of May,  
And summer's lease hath all too short a date:  
Sometime too hot the eve of heaven shines.

# slow & inflexible

But thy eternal summer shall not fade,  
Nor lose possession of that fair thou ow'st,  
Nor shall death brag thou wander'st in his shade,  
When in eternal lines to time thou grow'st,  
So long as men can breathe, or eyes can see,  
So long lives this, and this gives life to thee.

Shall I compare thee to a summer's day?  
Thou art more lovely and more temperate:  
Rough winds do shake the darling buds of May,  
And summer's lease hath all too short a date:  
Sometime too hot the eye of heaven shines,  
And often is his gold complexion dimmed,  
And every fair from fair sometime declines,  
By chance, or nature's changing course untrimmed:  
But thy eternal summer shall not fade,  
Nor lose possession of that fair thou ow'st,  
Nor shall death brag thou wander'st in his shade,  
When in eternal lines to time thou grow'st,  
So long as men can breathe, or eyes can see,  
So long lives this, and this gives life to thee.

Term	Doc 1	Doc 2	Doc 3
breathe			
brings			
buds			
but			
by			
can			
...			
damasked			
darling			
date			
day			
deaf			
death			
declines			
delight			



**sorted list of  
unique terms**

[illegible]



Term	Doc 1	Doc 2	Doc 3
<b>breathe</b>			
<b>brings</b>			
<b>buds</b>			
<b>but</b>			
<b>by</b>			
<b>can</b>			
<b>...</b>			
<b>damasked</b>			
<b>darling</b>			
<b>date</b>			
<b>day</b>			
<b>deaf</b>			
<b>death</b>			
<b>declines</b>			
<b>delight</b>			

Term	Doc 1	Doc 2	Doc 3
breathe			
brings			
<b>buds</b>			
but			
by			
can			
...			
damasked			
<b>darling</b>			
date			
day			
deaf			
death			
declines			
delight			

# inverted index

# inverted index

- term frequencies

# inverted index

- term frequencies » relevance

# inverted index

- term frequencies » relevance
- text length

# inverted index

- term frequencies » relevance
- text length » doc weight

# inverted index

- term frequencies » relevance
- text length » doc weight
- term positions



# inverted index

- term frequencies » relevance
- text length » doc weight
- term positions » word proximity

# inverted index

- term frequencies » relevance
- text length » doc weight
- term positions » word proximity
- char offsets

# inverted index

- term frequencies » relevance
- text length » doc weight
- term positions » word proximity
- char offsets » highlighting

# inverted index

**not just for text**

# inverted index

numbers, dates, bools, enums  
geopoints, geoshapes, etc

# step 2:

## analytics

# for search

map values → doc\_ids

# for search

map values  $\rightarrow$  doc\_ids

# for analytics

map doc\_ids  $\rightarrow$  values



# uninvert the index

# uninvert the index

cache values in memory  
called "fielddata"

# uninvert the index

**data access from RAM  
very fast**

# on-the-fly analytics

**in the context of  
a user's query**

# on-the-fly analytics

**relevant analytics  
for each user**

# calculate metrics

count, min, max, sum, avg,  
percentiles, cardinality,  
stddev, variance, sum of squares

**grouped by**

**popular terms, significant terms,  
ranges, dates, geolocation, etc**

**grouped by**

**groups can**

**... contain subgroups**

**... which contain subgroups**

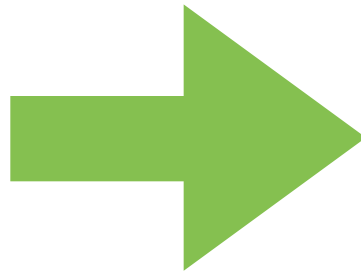
**etc**



**step 3:**

**building the inverted index**

# inverted index



# immutable



# immutable

- cache friendly



# immutable

- cache friendly
- reads from RAM



# immutable

- cache friendly
- reads from RAM
- fielddata never changes



# immutable

- cache friendly
- reads from RAM
- fielddata never changes
- compressible



# immutable

- cache friendly
- reads from RAM
- fielddata never changes
- compressible
- no locking





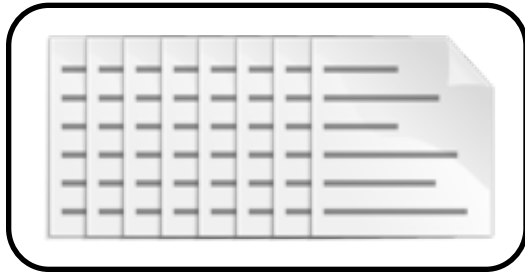
# but, immutable...



**step 4:**

**dynamic inverted index**

# in-memory buffer



## commit



# segment

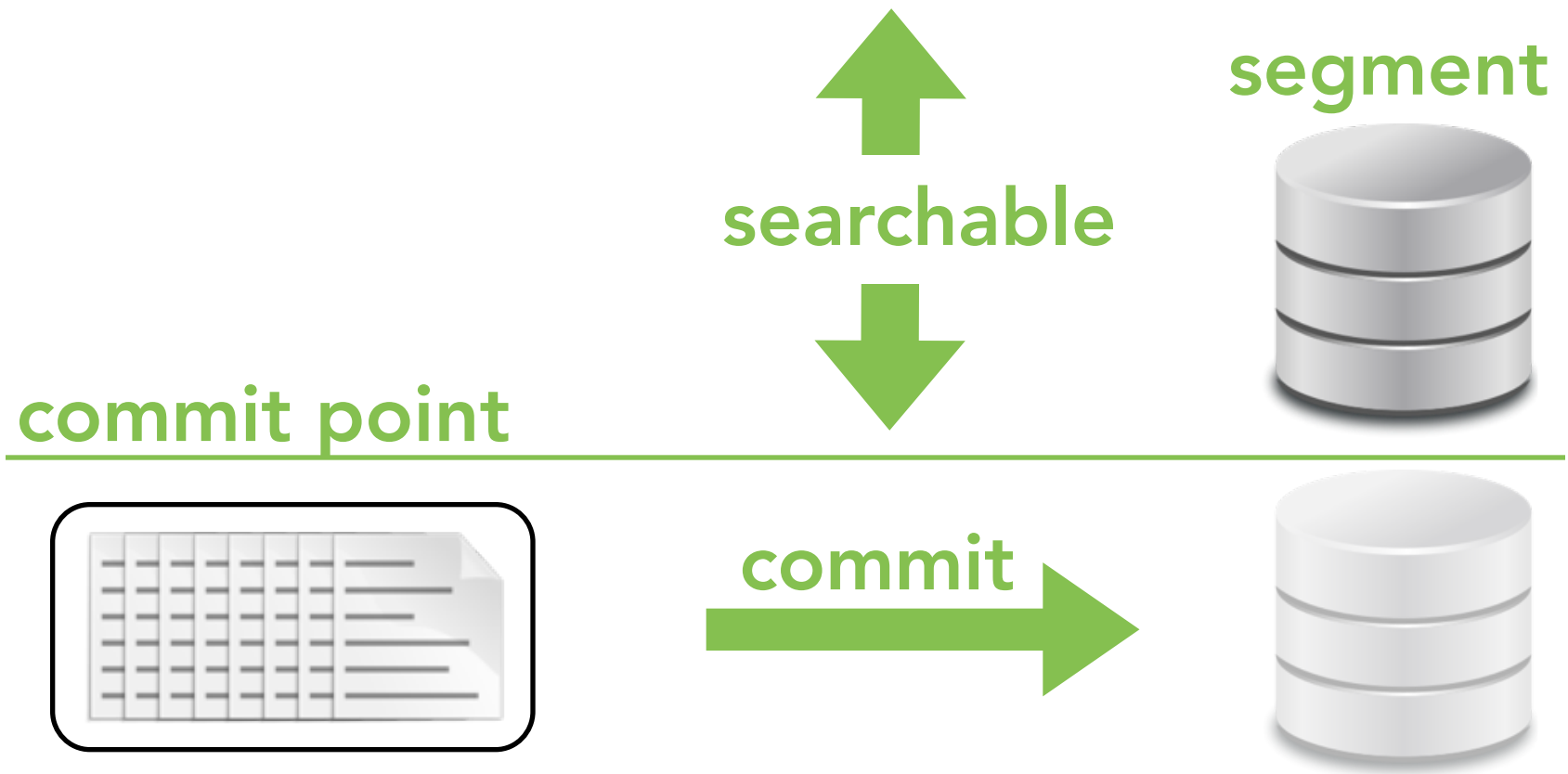


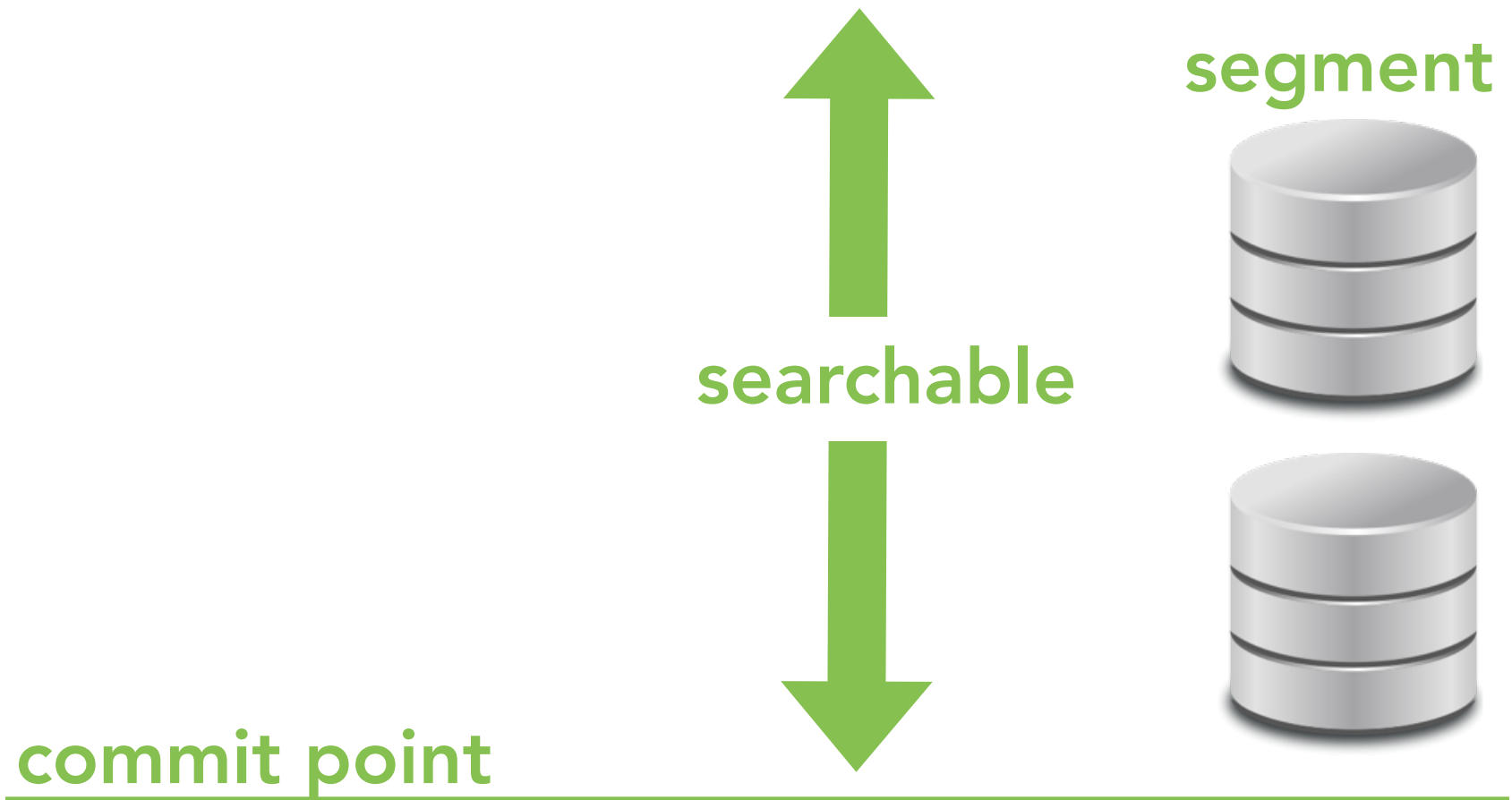
commit point

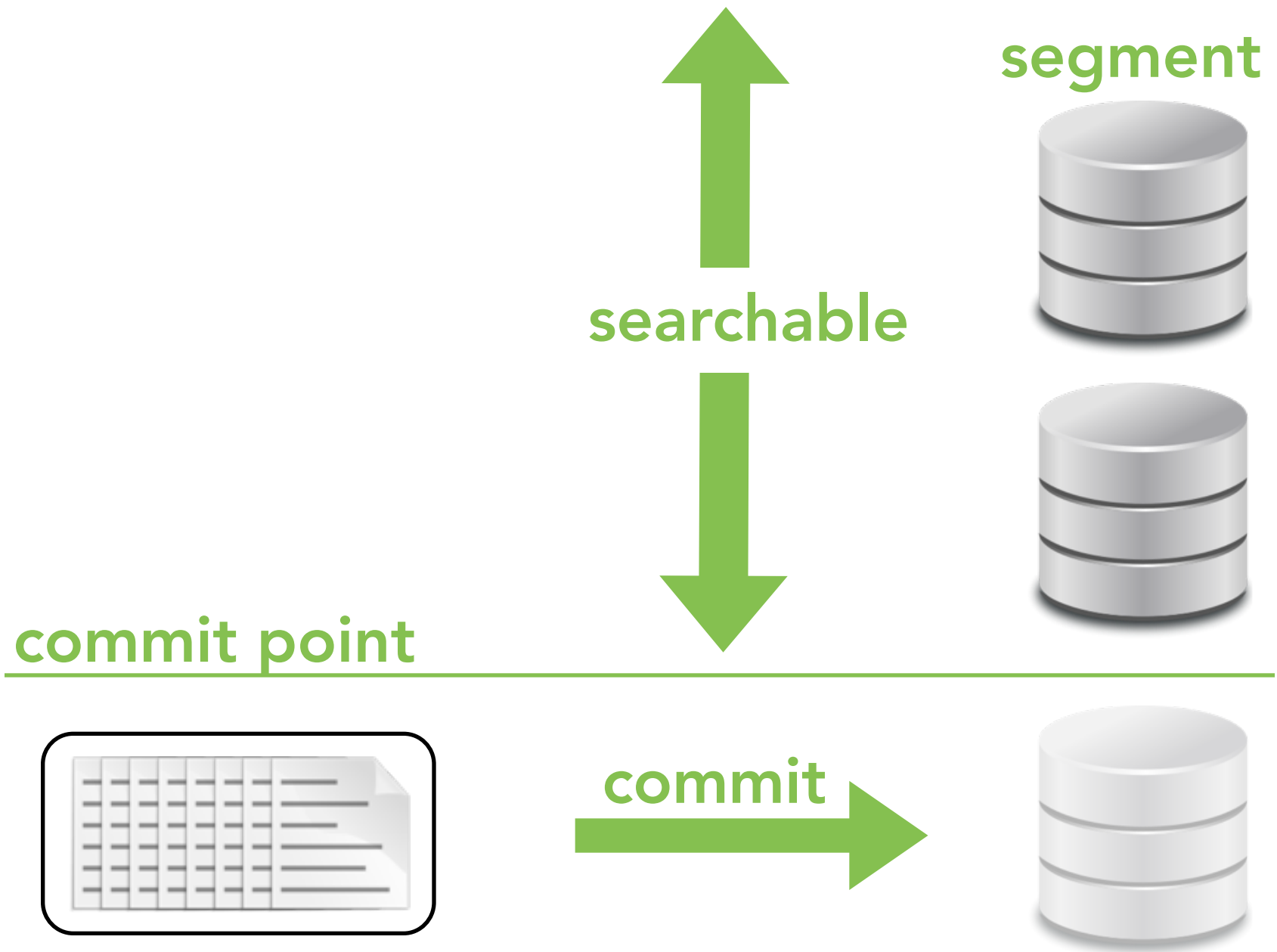
searchable

segment









segment



searchable

commit point

elasticsearch.



# lucene commit

# lucene commit

- write new segment

# lucene commit

- write new segment
- write new commit point

# lucene commit

- write new segment
- write new commit point
- fsync

# lucene commit

- write new segment
- write new commit point
- fsync
- clear buffer

# lucene commit

- write new segment
- write new commit point
- fsync
- clear buffer
- reopen index

# lucene commit

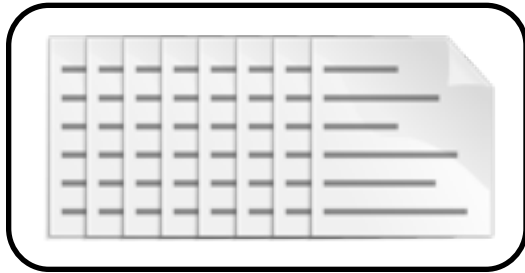
- write new segment
- write new commit point
- **fsync** ← **expensive!**
- clear buffer
- reopen index

# step 5:

## near real-time search



# in-memory buffer



## flush



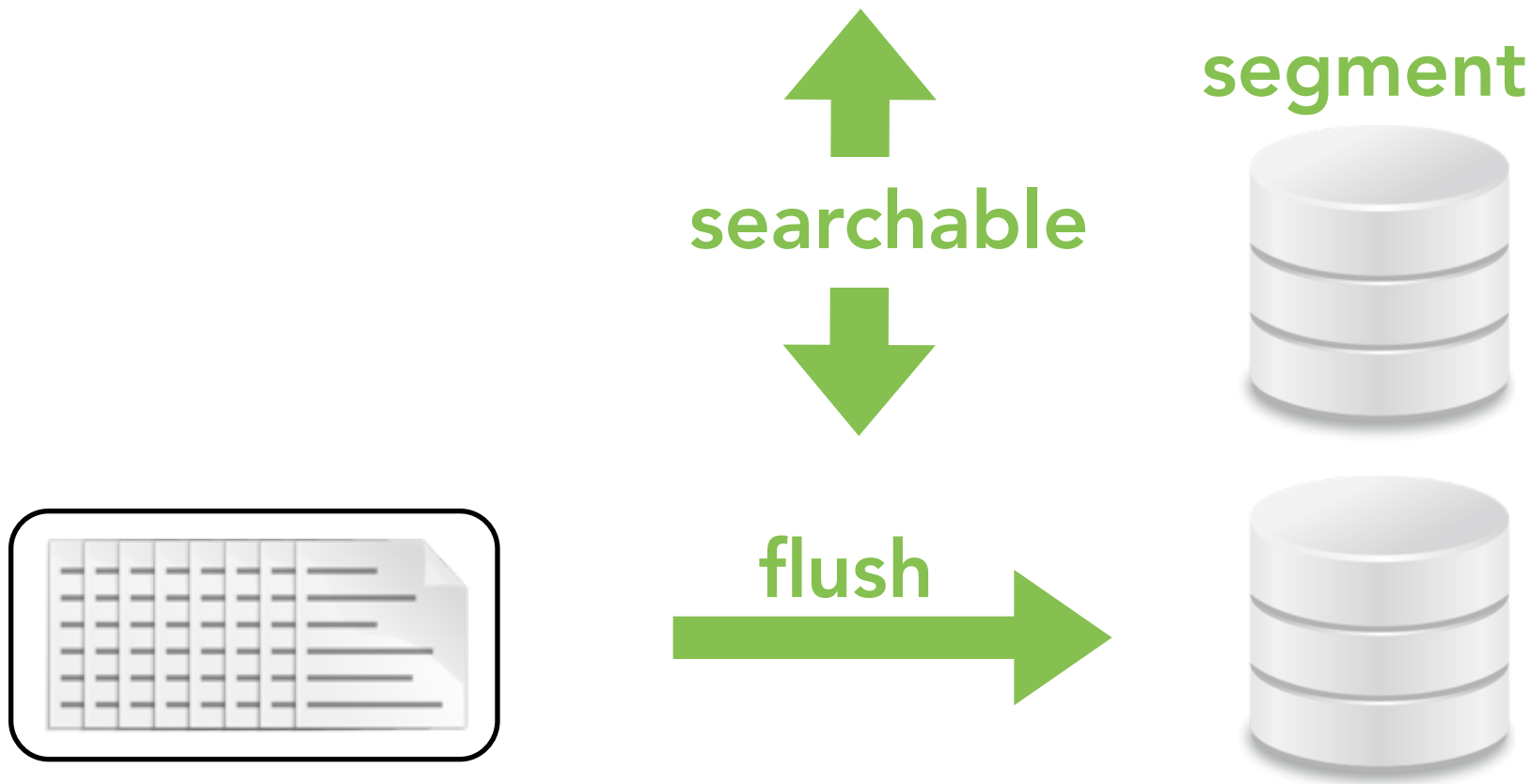
# segment



↑  
searchable  
↓

segment



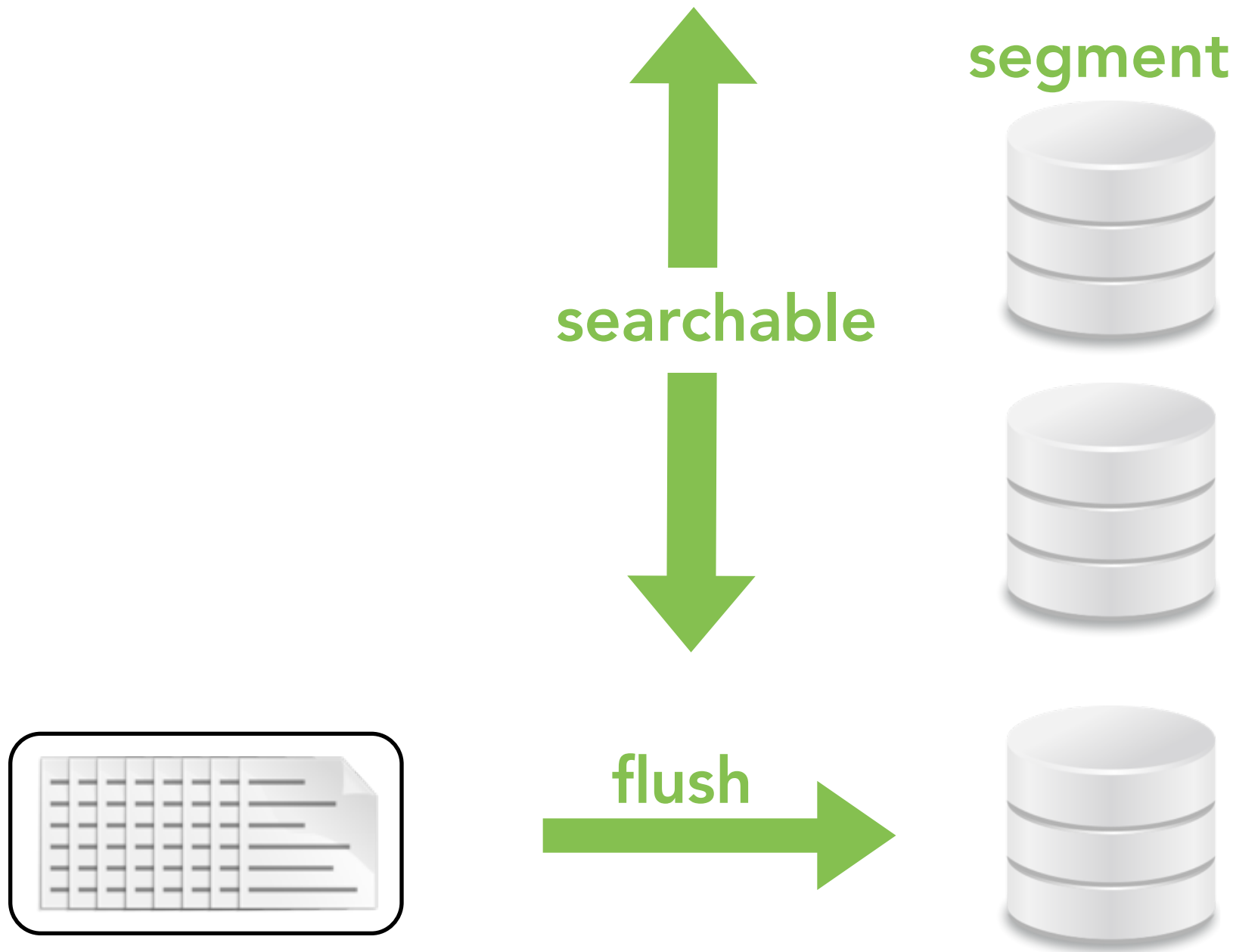


segment



searchable





segment



searchable



elasticsearch.

segment



commit

elasticsearch.

segment



searchable

commit point

elasticsearch.



# lucene flush

# lucene flush

- write new segment
- clear buffer
- reopen index

# lucene flush

- write new segment
- clear buffer
- reopen index
- **no fsync**

# lucene flush

- write new segment
- clear buffer
- reopen index
- **no fsync** → **lightweight**

but...

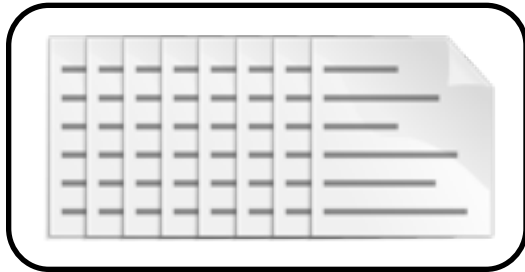
**data not safe until fsync'ed!**

**step 6:**  
**don't lose data**

**step 6:**

**don't lose data  
→ transaction log**

# in-memory buffer



## flush



# segment



## translog



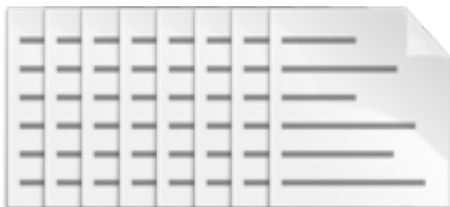


↑  
searchable  
↓

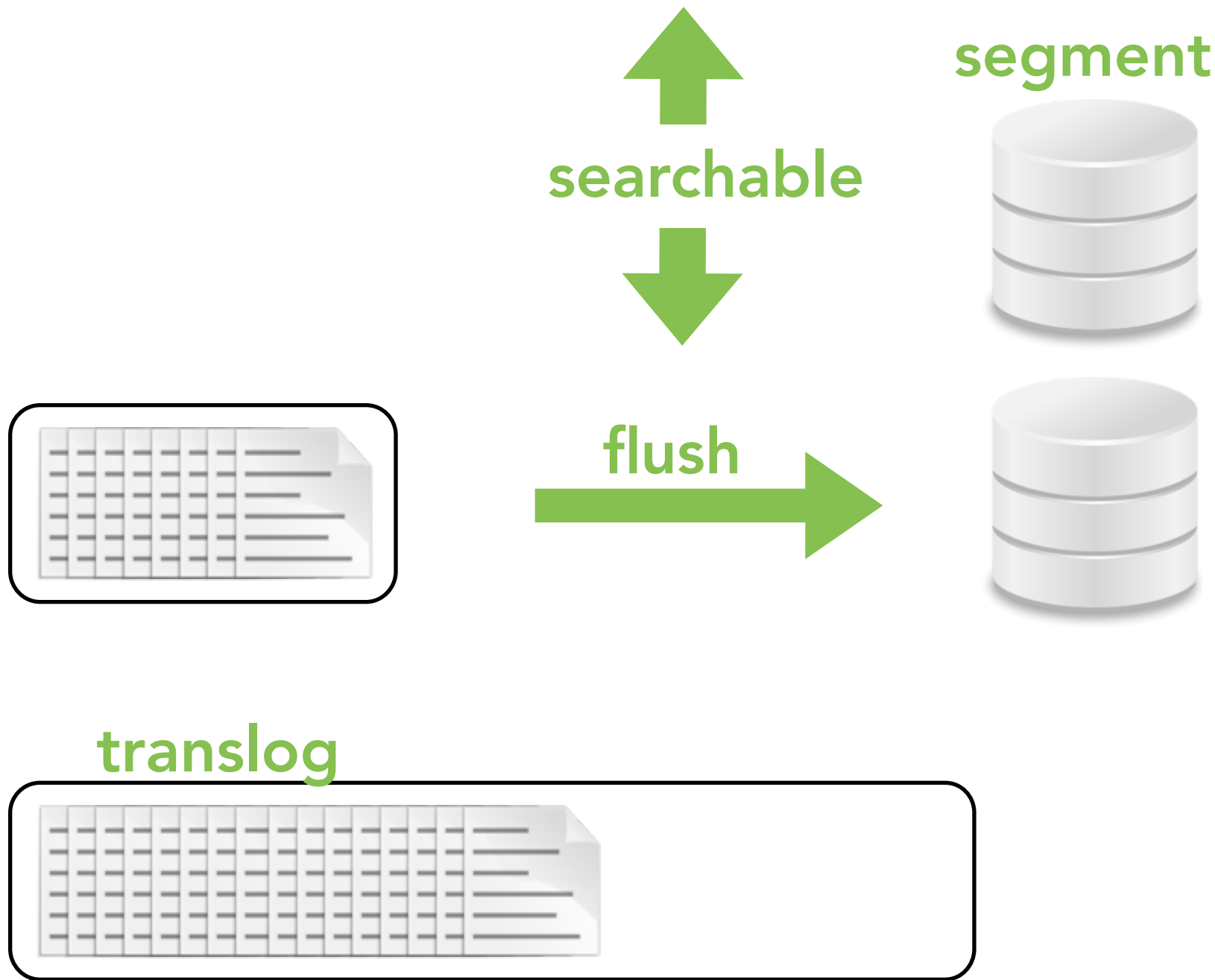
segment



translog



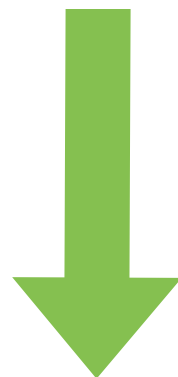
elasticsearch.



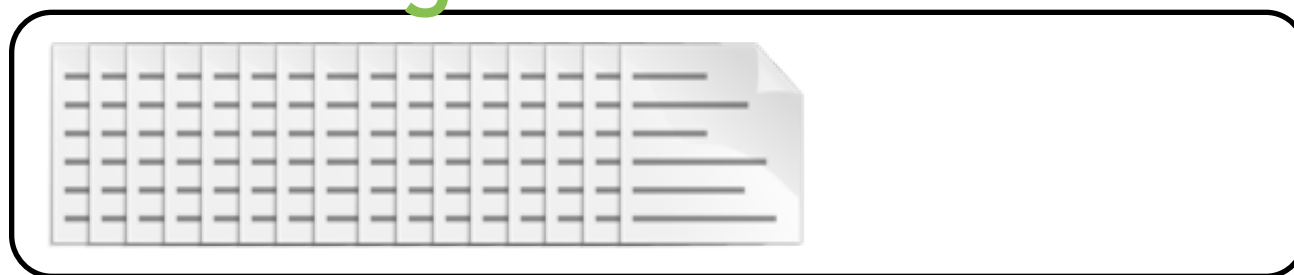
segment



searchable



translog



elasticsearch.

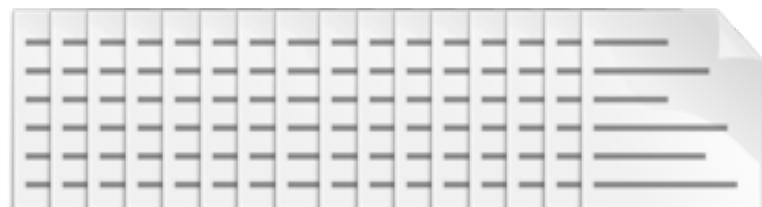
segment



commit



translog



elasticsearch.

segment

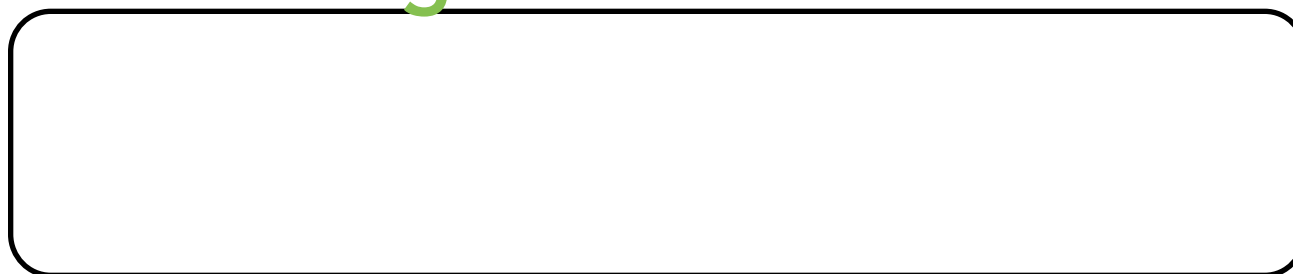


searchable

commit point

---

translog



elasticsearch.

# elasticsearch “refresh”

- lucene “flush”
- makes changes searchable
- lightweight

# elasticsearch “flush”

- lucene “commit”
- clears transaction log
- persists changes
- heavy

refresh every second



# near real-time search!

**near real-time search!**  
**near real-time analytics!**

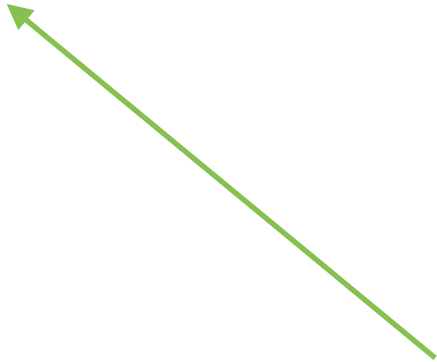


# too many segments

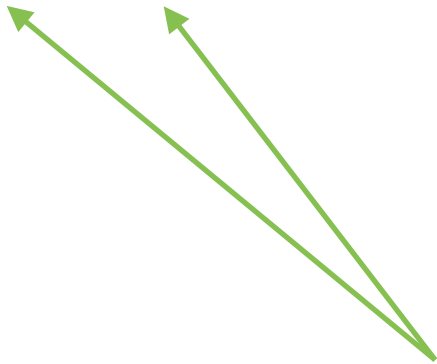
- slow searches
- poor term frequencies
- poor compression

# step 7:

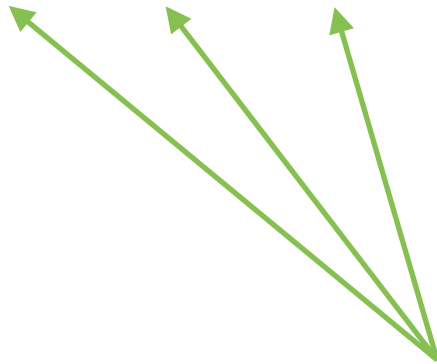
## reduce segments



**searchable**

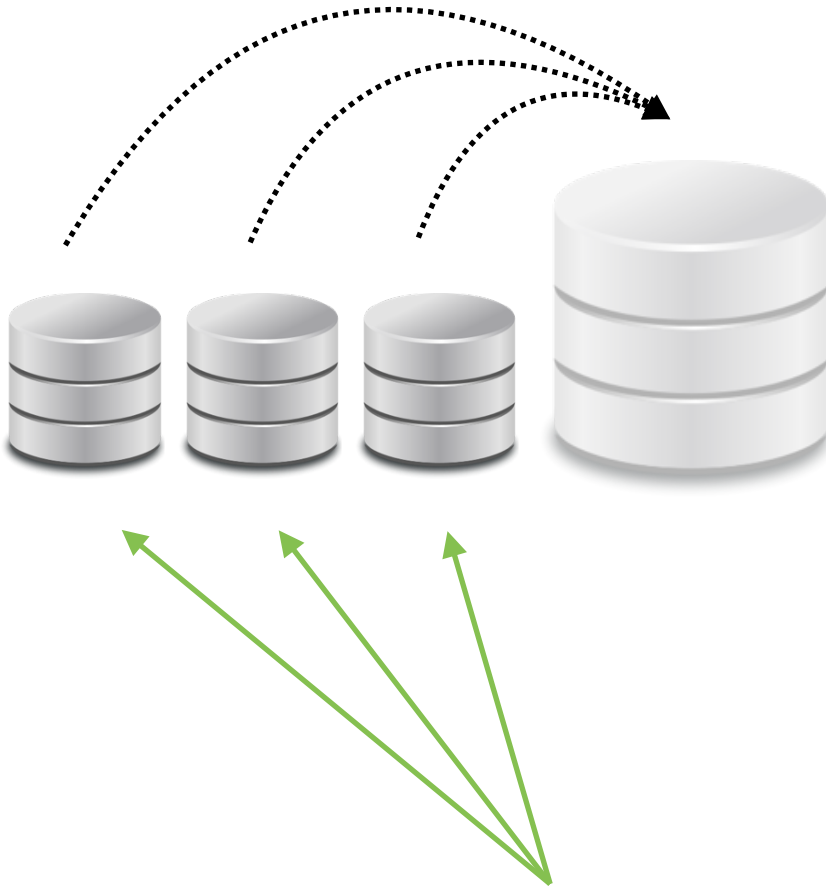


**searchable**

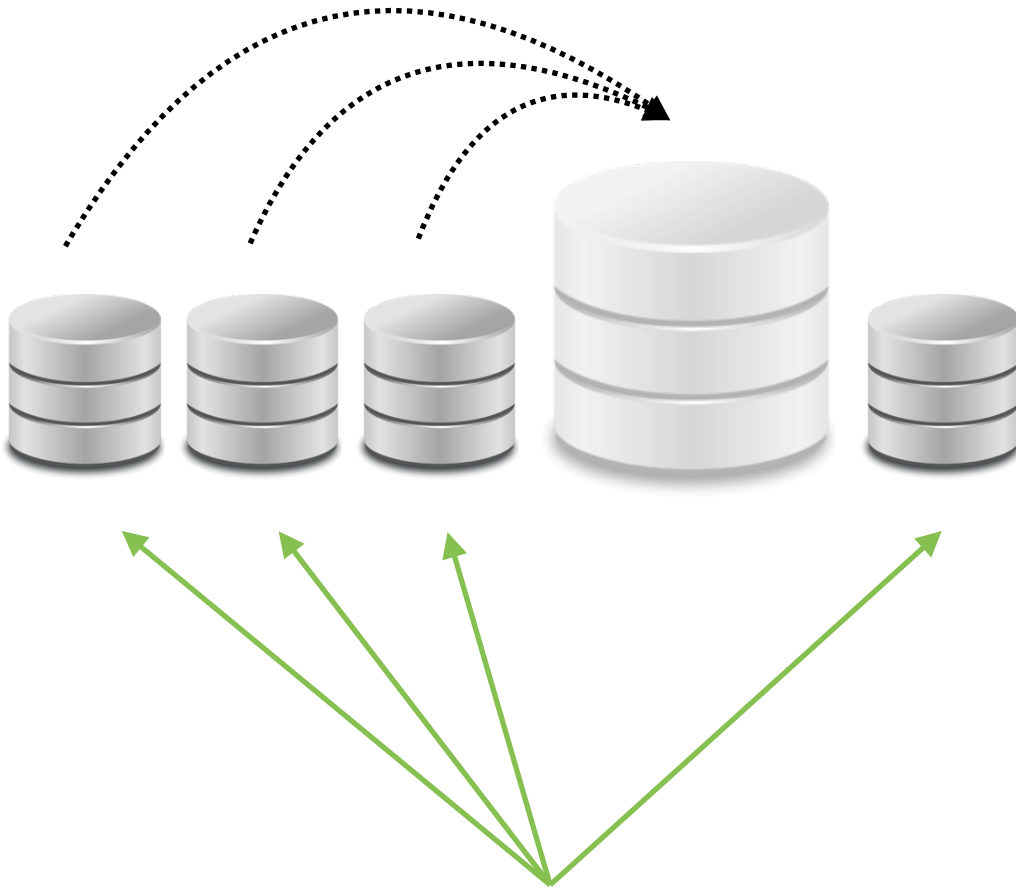


**searchable**

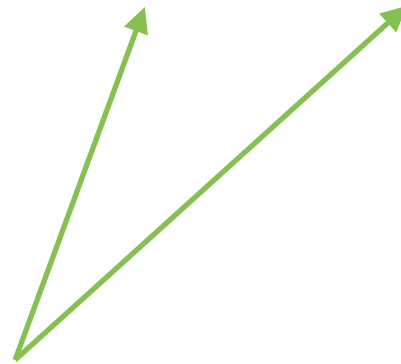




**searchable**



**searchable**



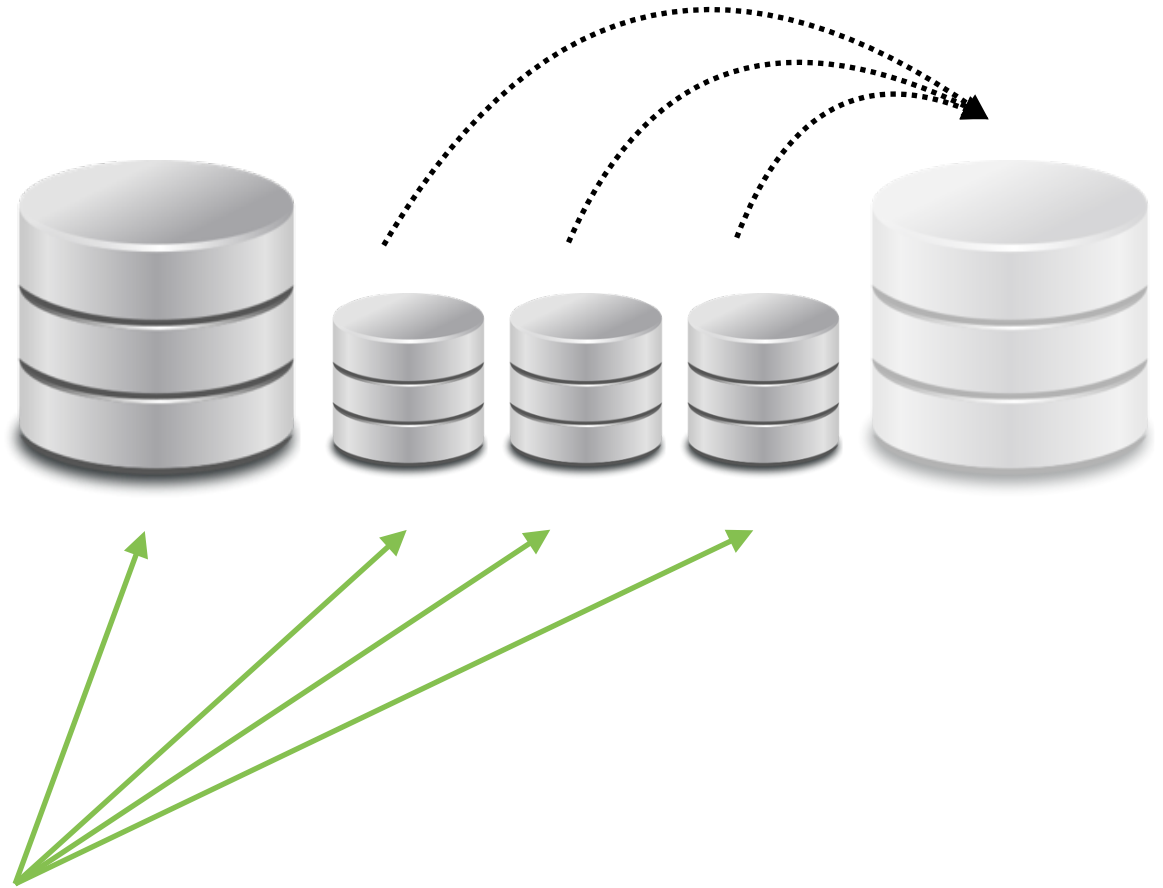
**searchable**



**searchable**



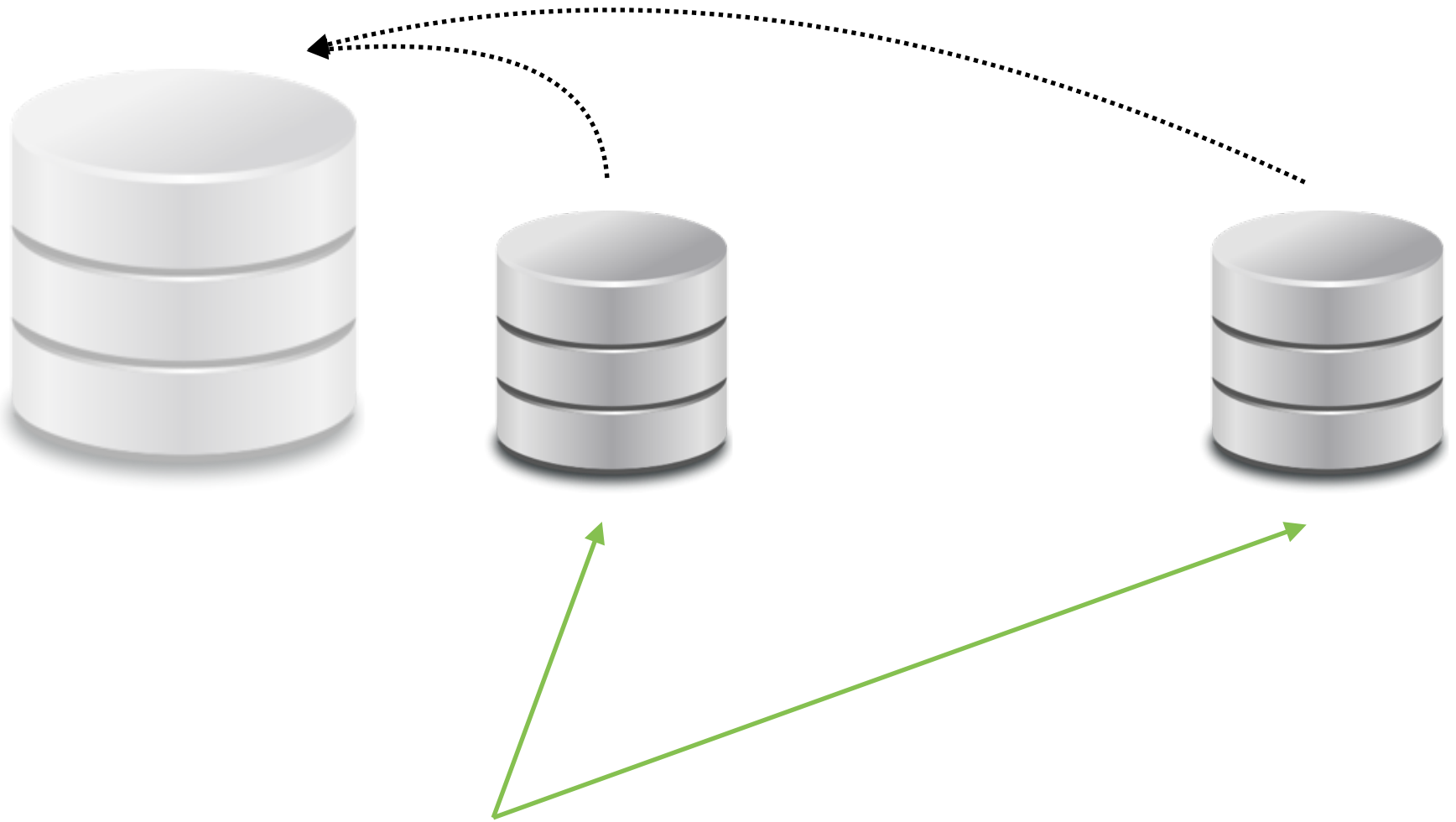
**searchable**



**searchable**

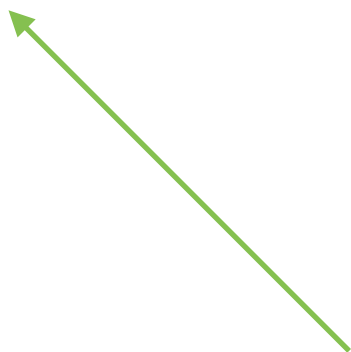


**searchable**



**searchable**





**searchable**

# merge process

- many small → one big
- removes deleted docs
- runs in background
- throttled

but...



"Any wonder it broke down" by Brian Snelson is licensed under CC BY 2.0

sometimes you  
need another truck

# step 8:

## scale out, not up

# shard your data

# shard your data

## transparent in elasticsearch



# many segments



# many segments → one shard



many segments → one shard



many shards



many segments → one shard



many shards → one index



**"node"**

**running instance of elasticsearch**

**≈ one server**

**"shard"**

**bucket of data**

**physical worker unit**

**lives on one node**

**“index”**

**logical namespace**

**points to one or more shards**

# "index"

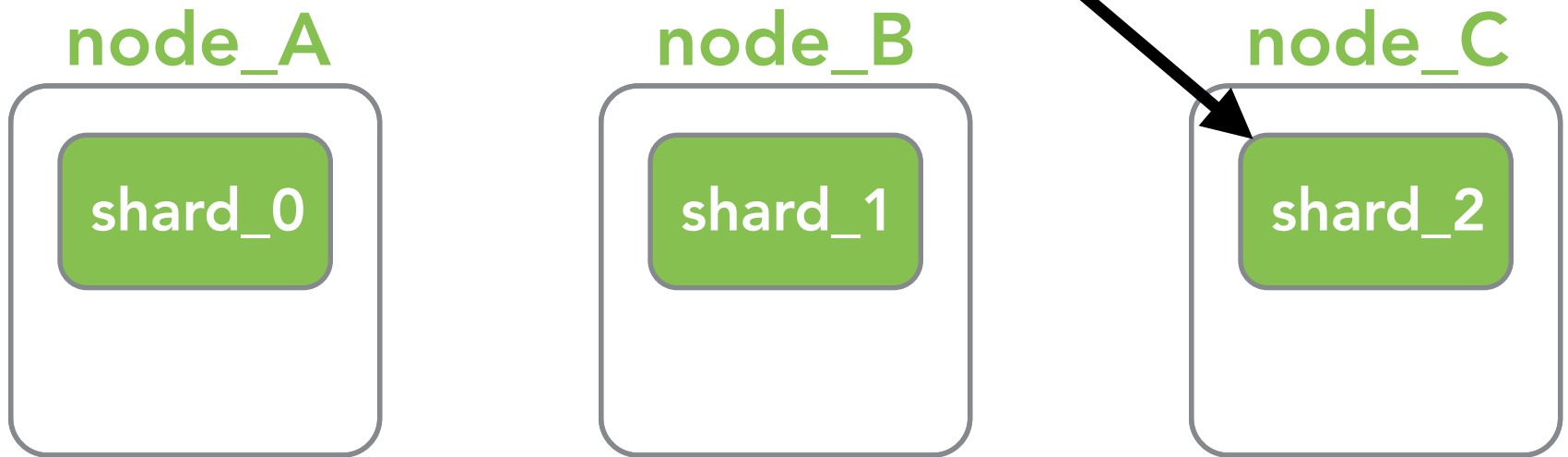
**logical namespace**

**points to one or more shards**

```
shard = hash(_id) % no_of_shards
```

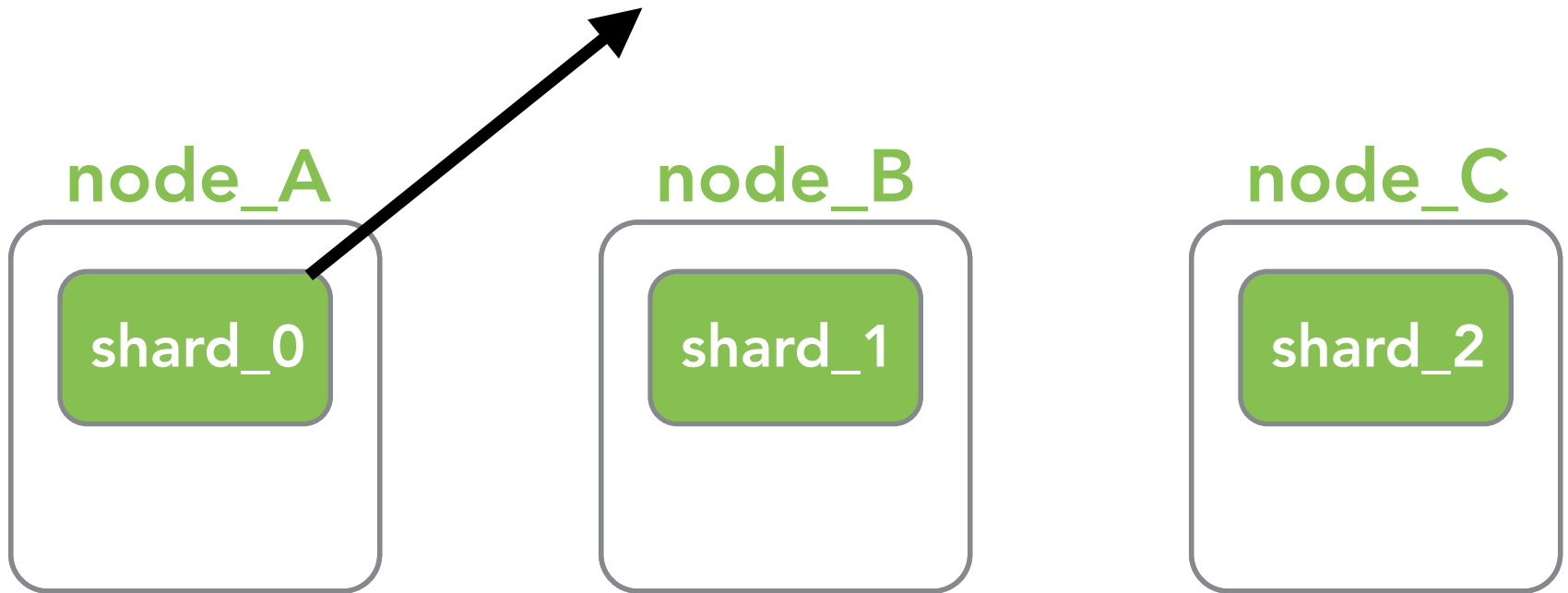


# PUT doc\_id:1



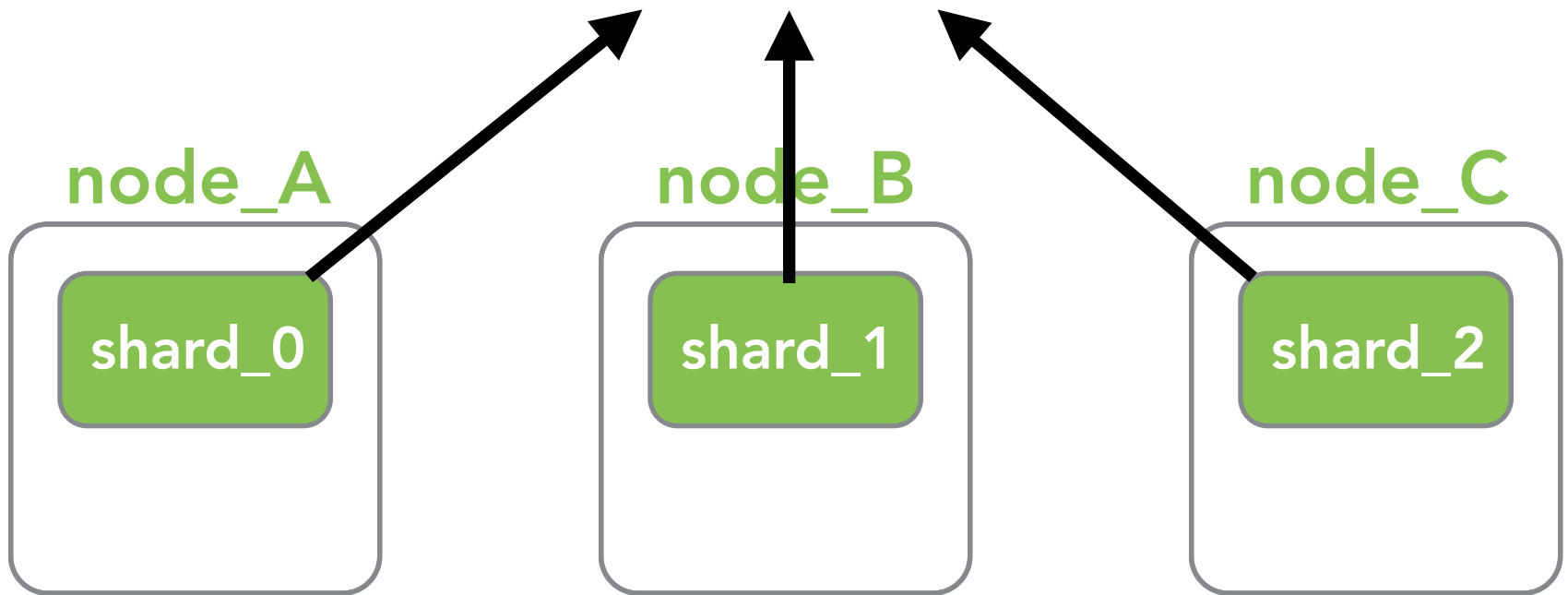
$\text{hash}(1) \% 3 \Rightarrow \text{shard\_2}$

# GET doc\_id:2



$\text{hash}(2) \% 3 \Rightarrow \text{shard\_0}$

# Search all docs



~~$$\text{shard} = \text{hash}(\_id) \% \text{no\_of\_shards}$$~~

# step 9:

## scaling elastically

# start small

node\_A

shard\_0

shard\_1

shard\_2

# add more nodes

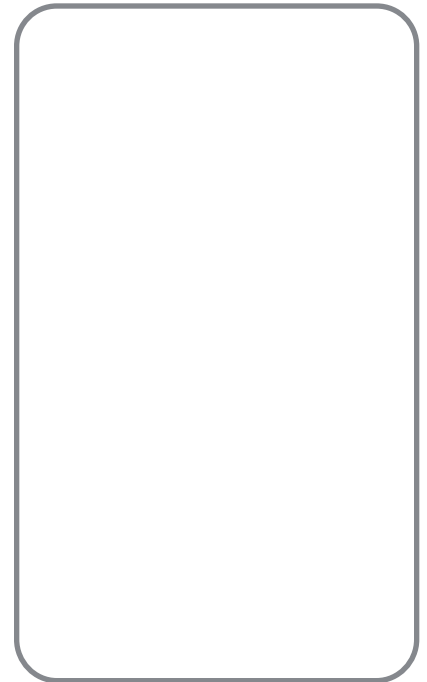
node\_A



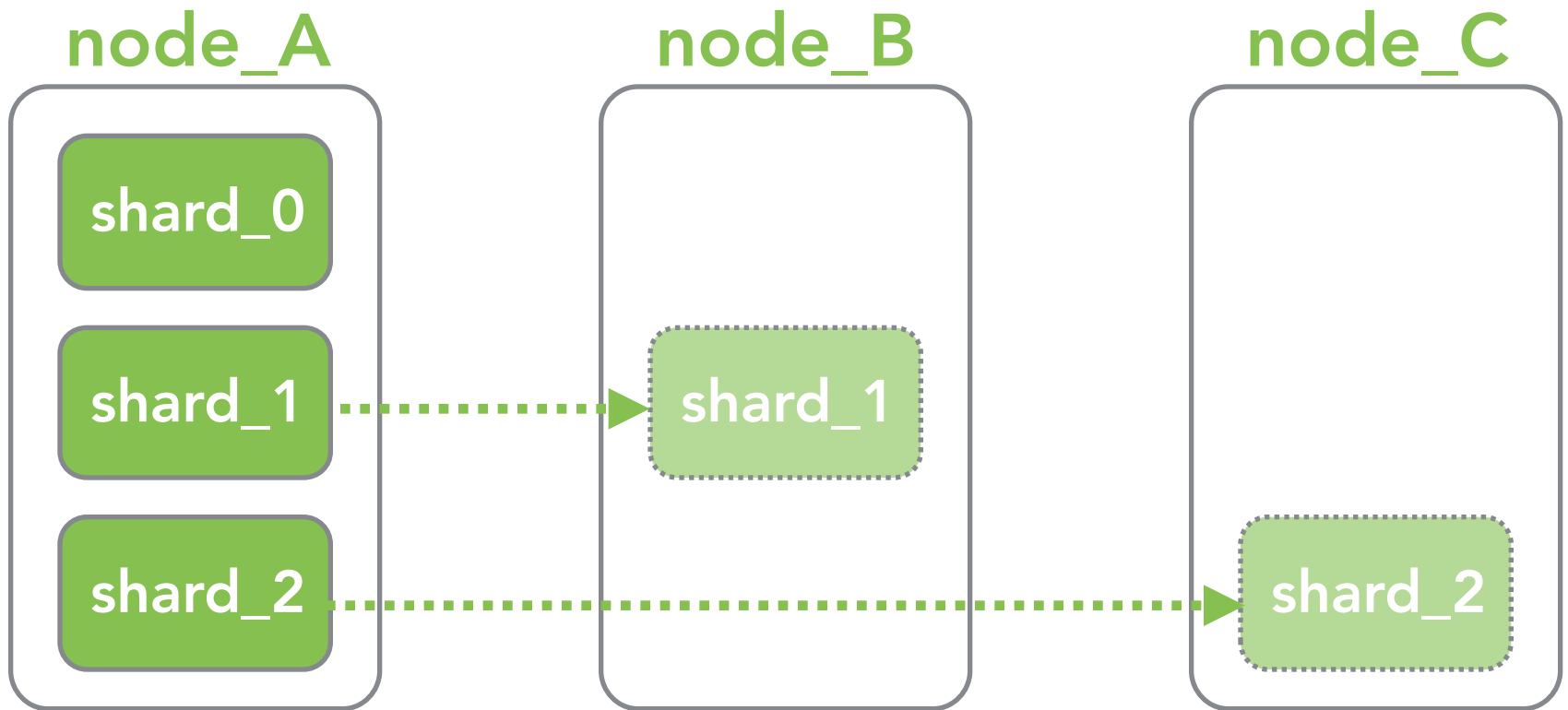
node\_B



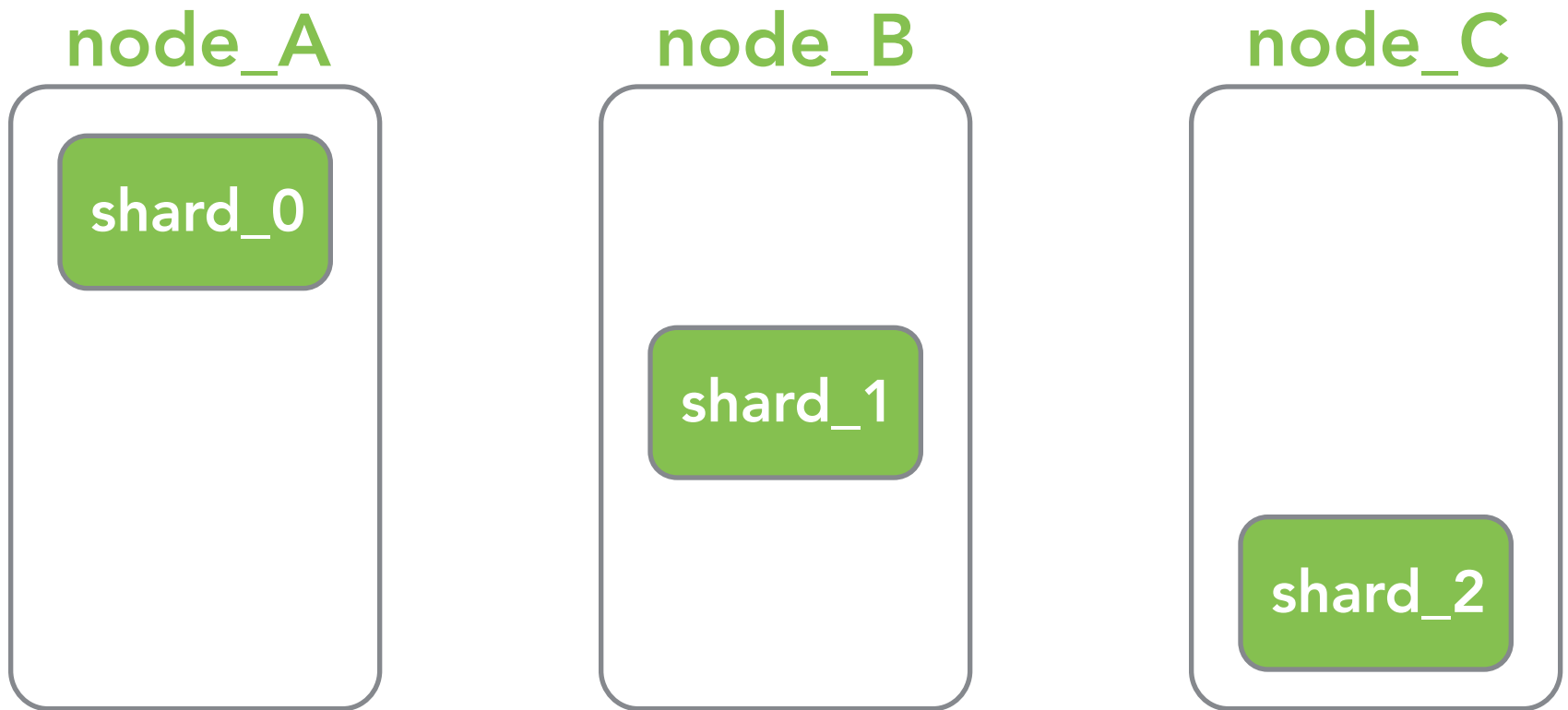
node\_C



# shards migrate

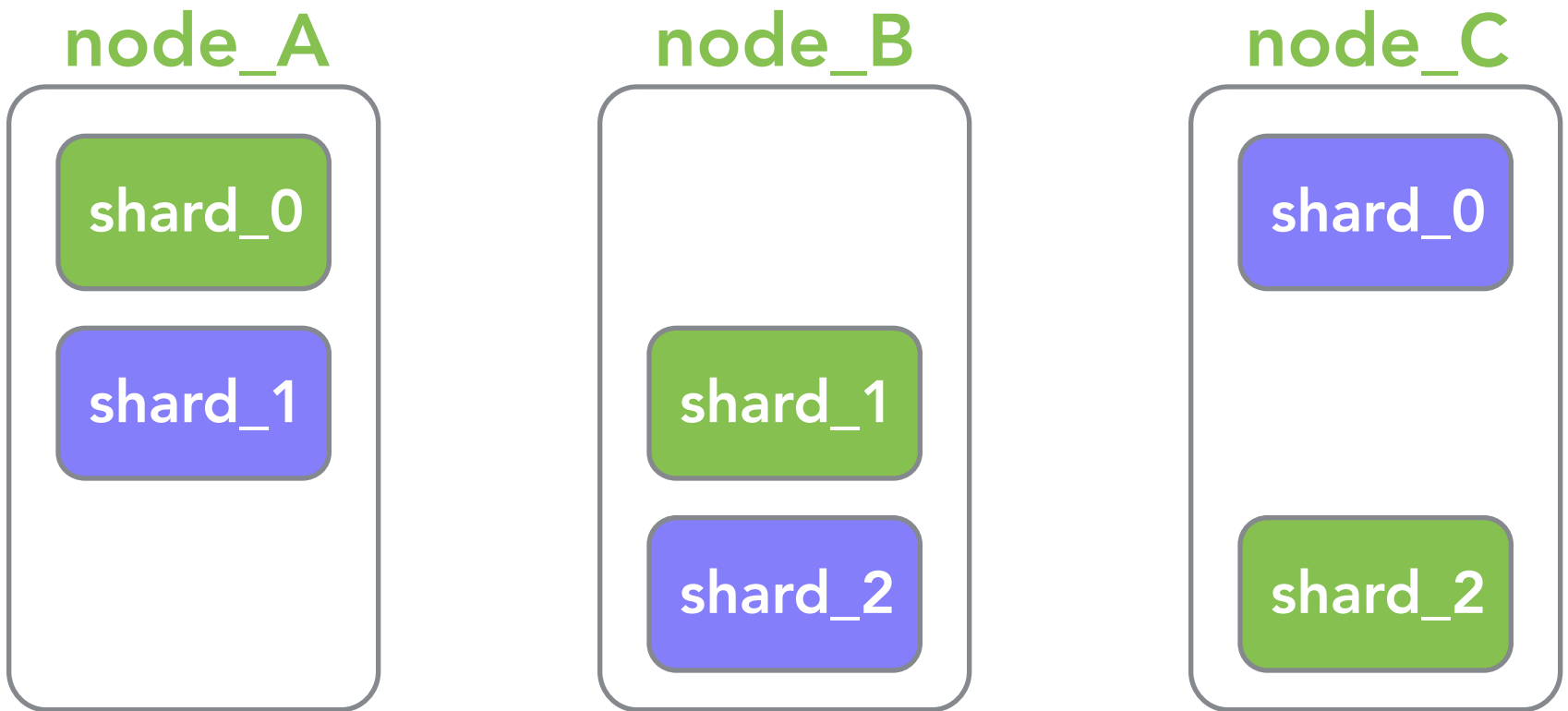


# rebalanced





# add new index



but...

but...

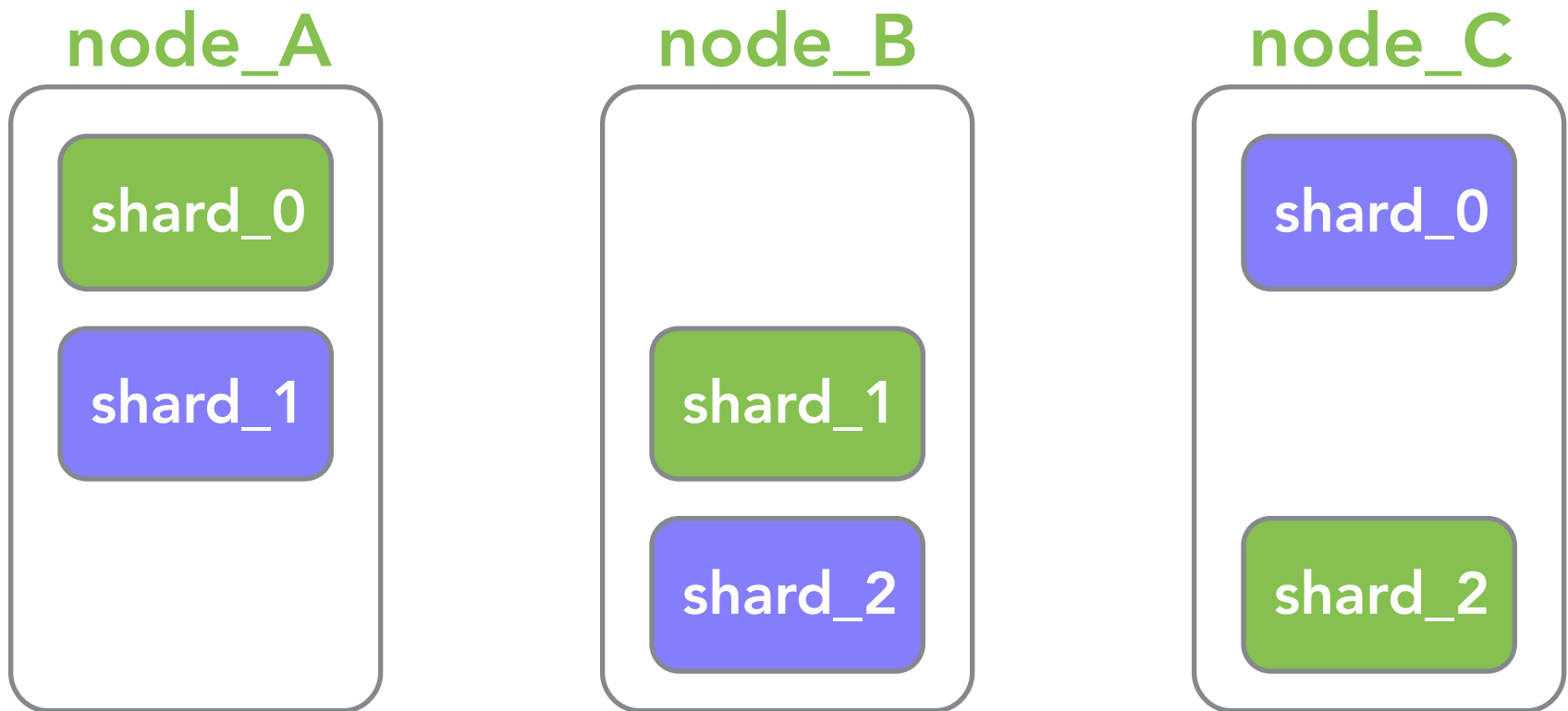
**more hardware?**

but...

**more hardware?**

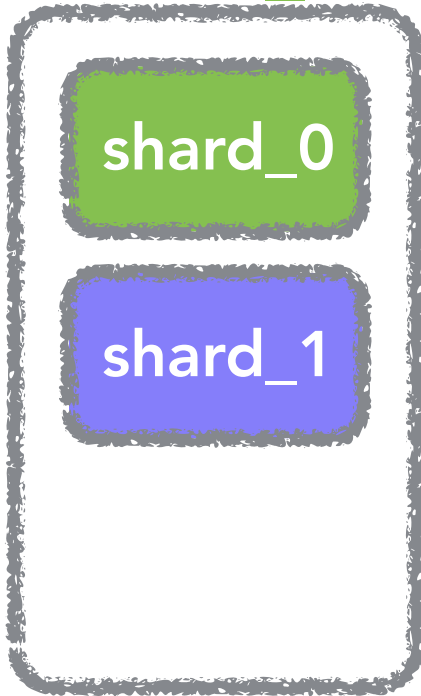
**more hardware failure**

# at 3am on sunday...



# boom!

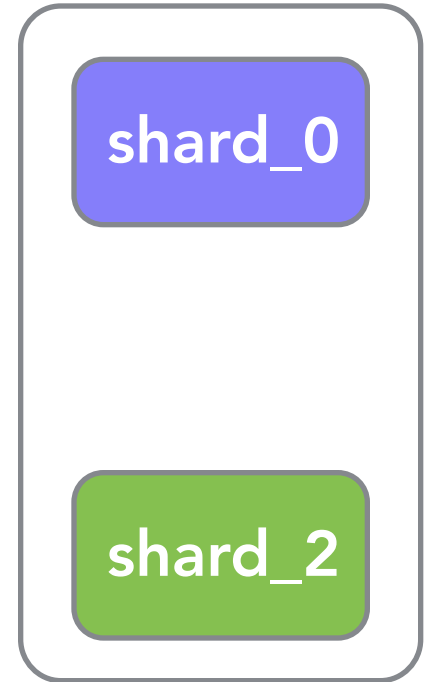
node\_A



node\_B



node\_C



# step 10:

## add redundancy

for every shard

**...make a copy**



**"primary shard"**

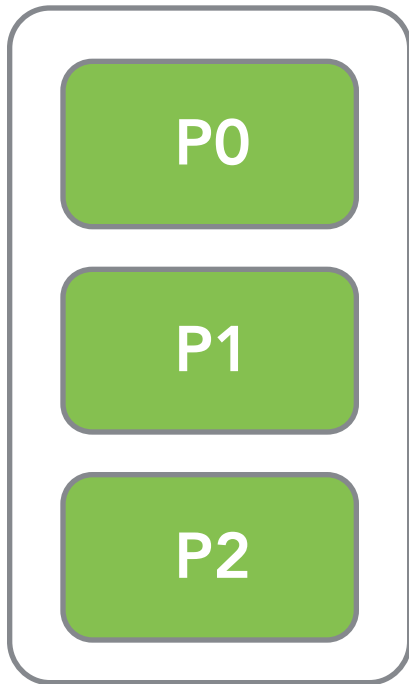
**main shard**

**“replica shard(s)”**

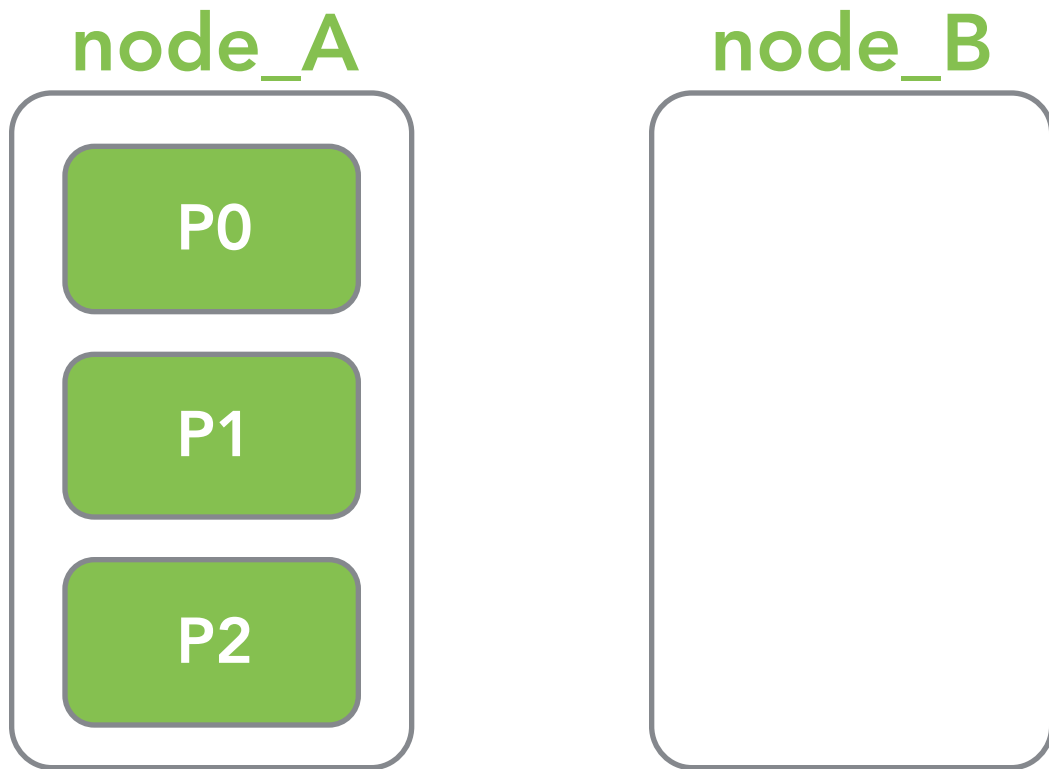
**copy of primary shard**

# one node

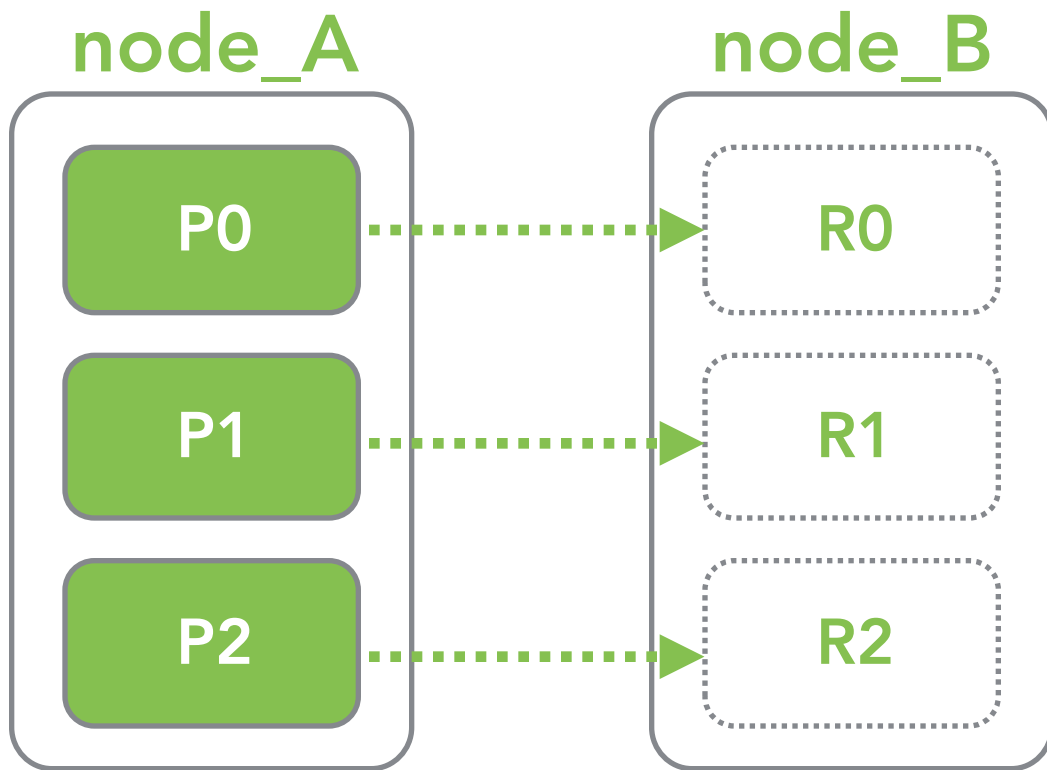
node\_A



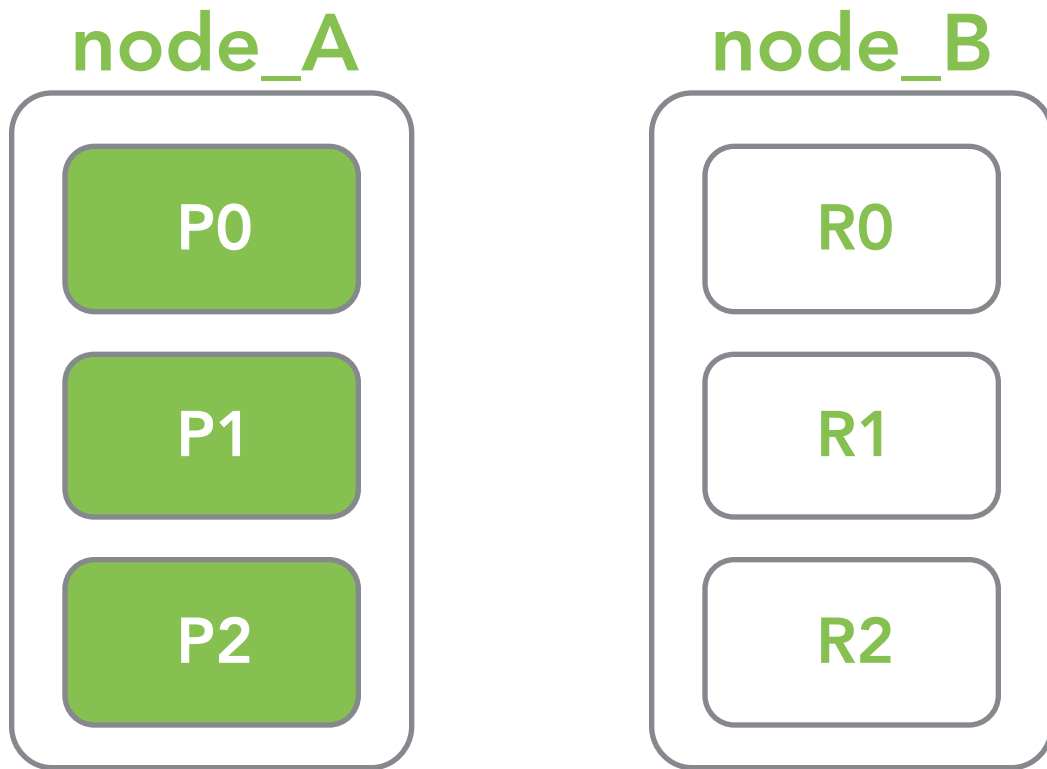
# add a node



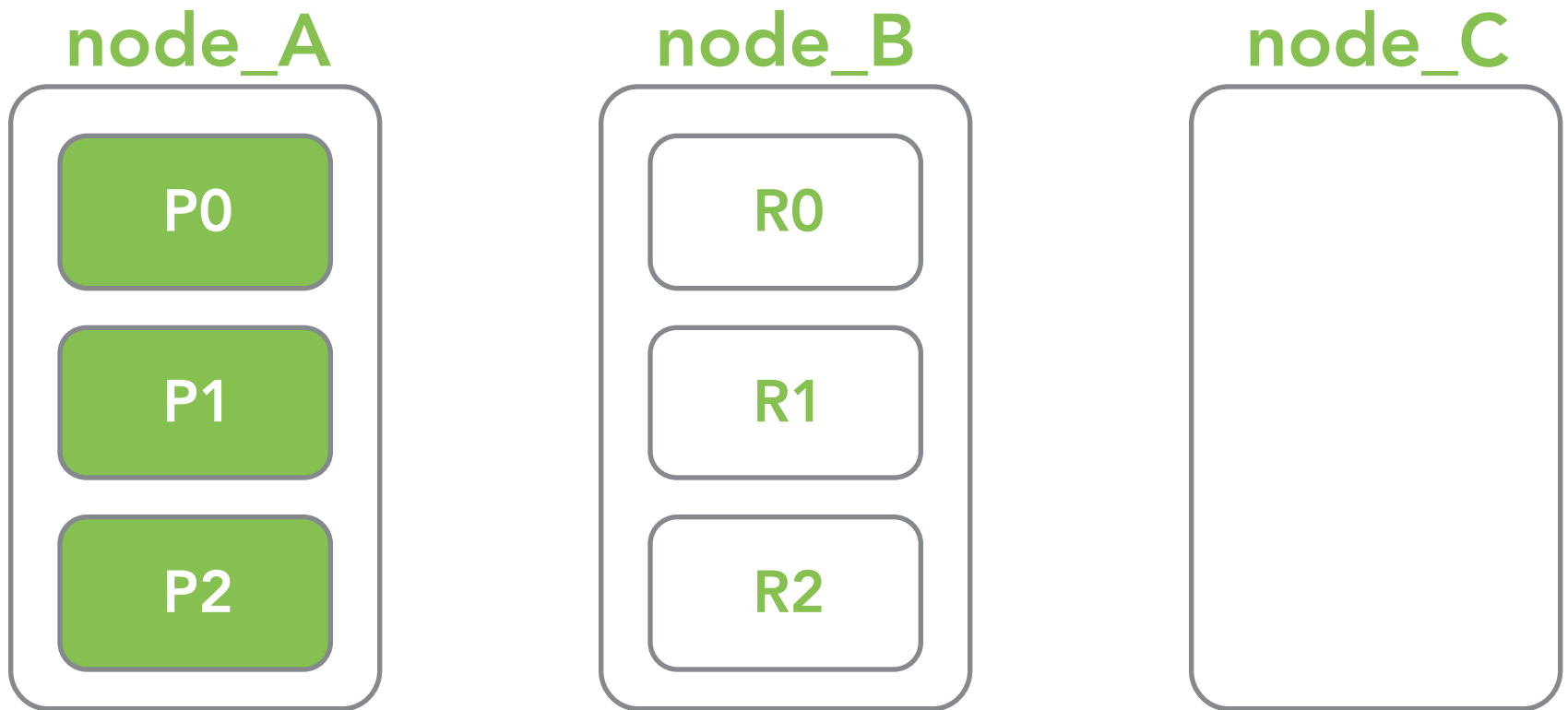
# add a node



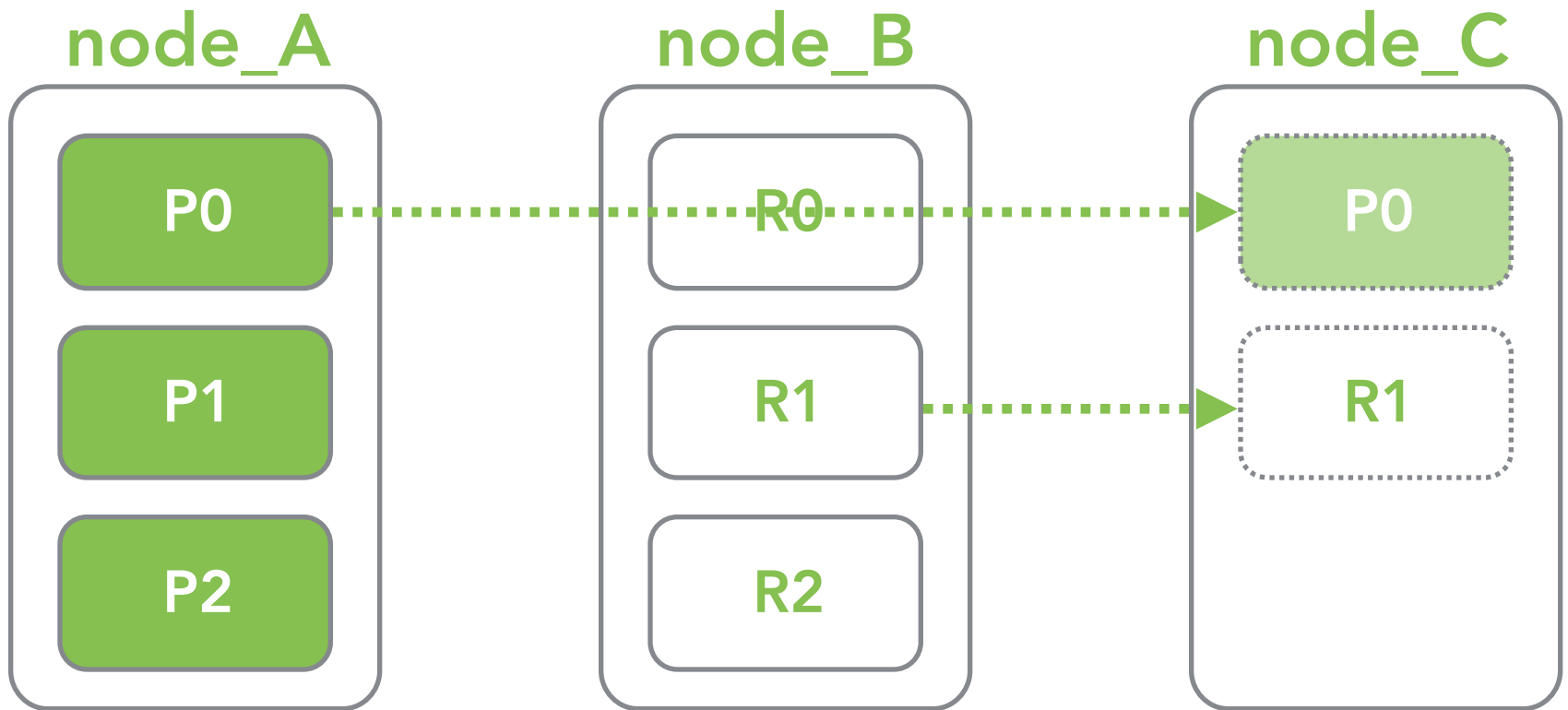
# redundancy



# add a node

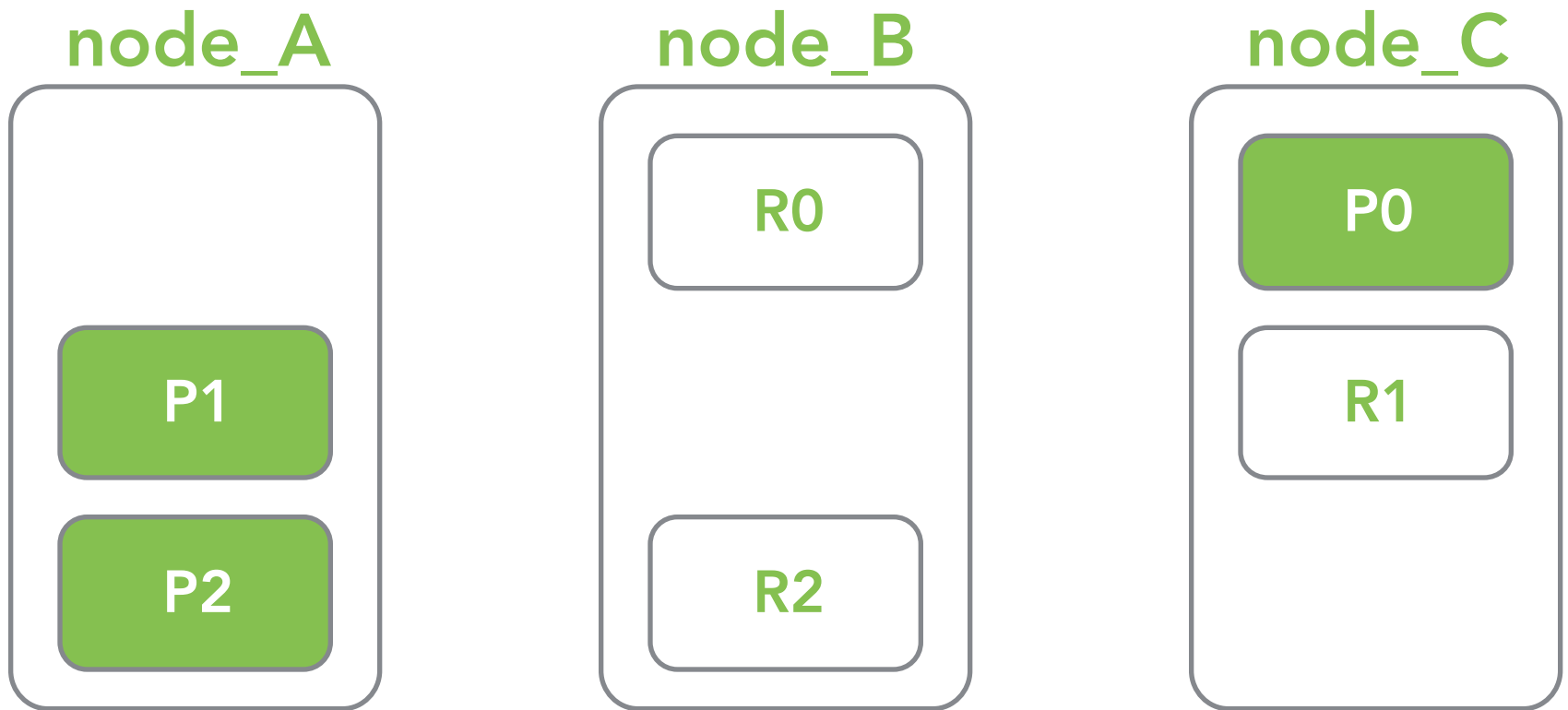


# add a node

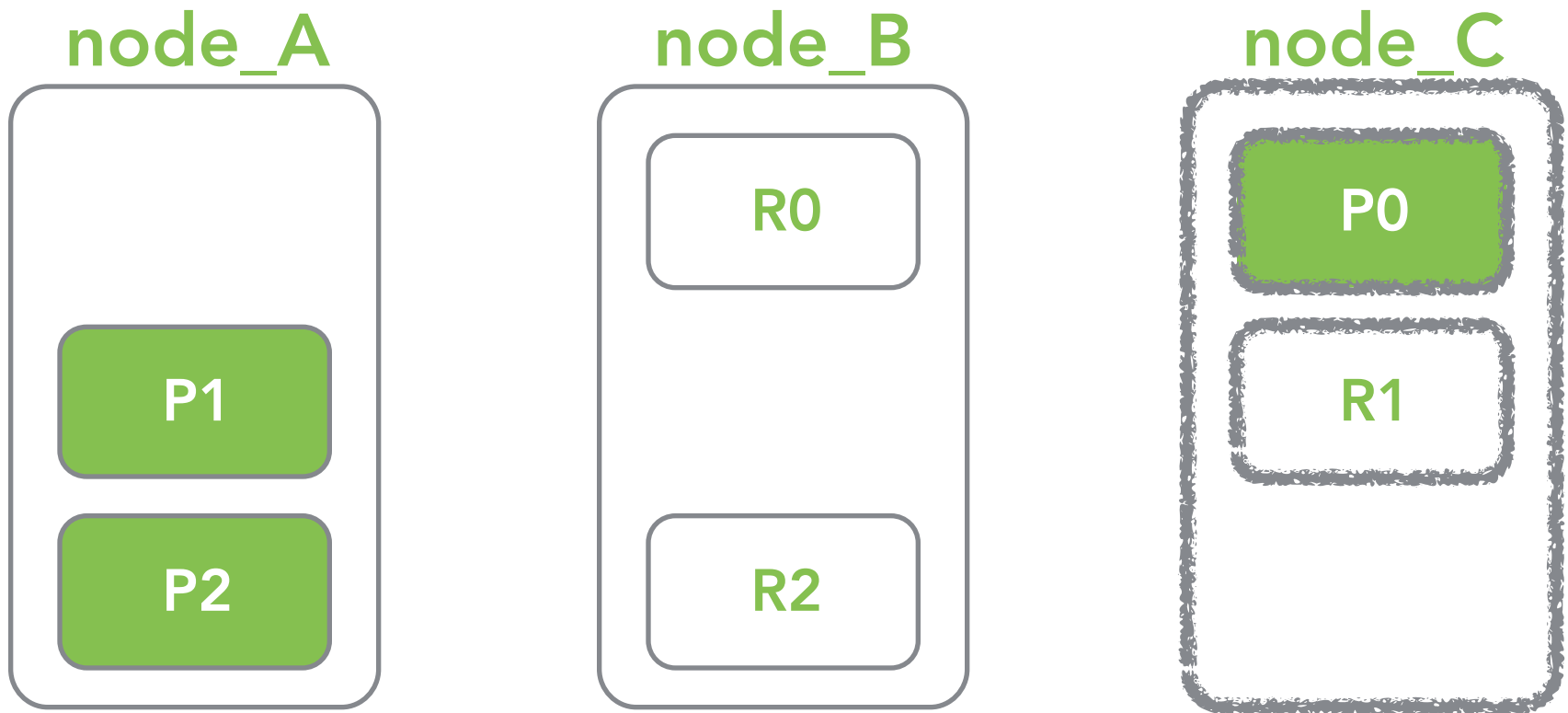




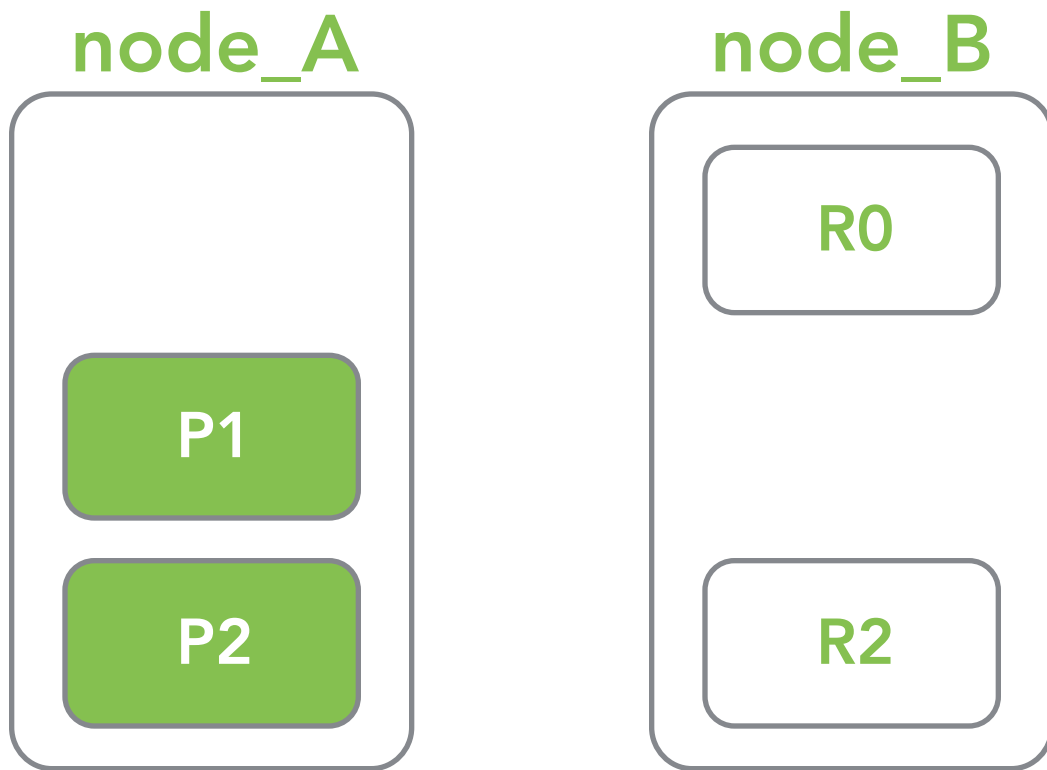
# rebalanced



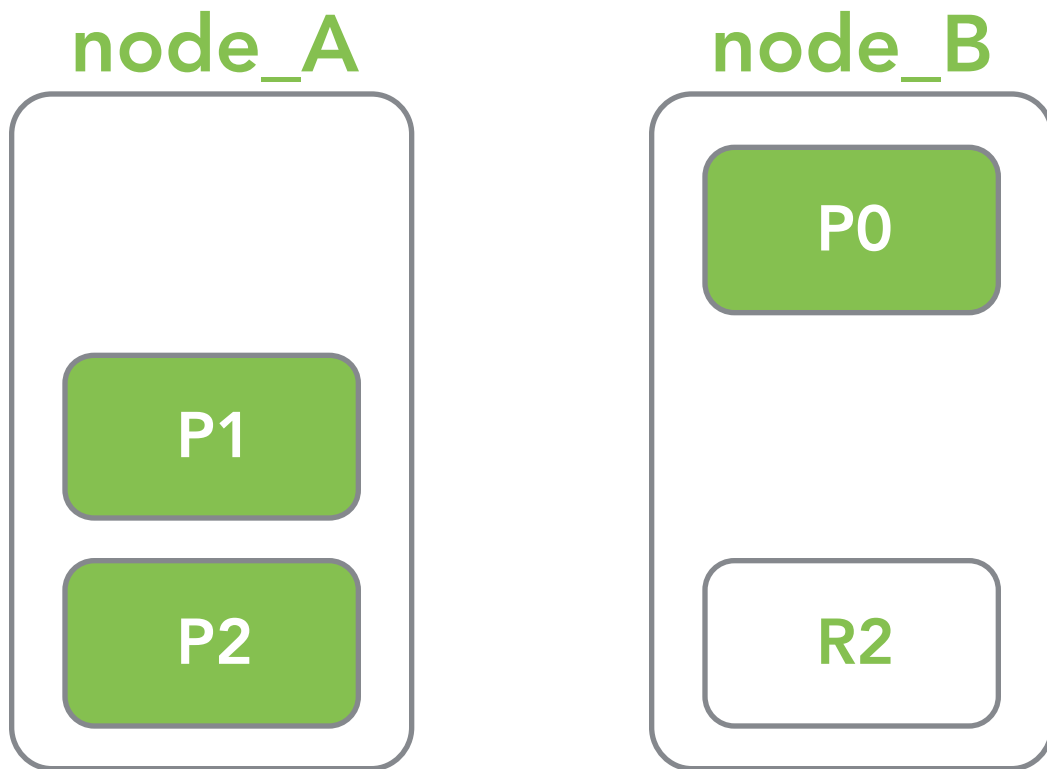
# lose a node



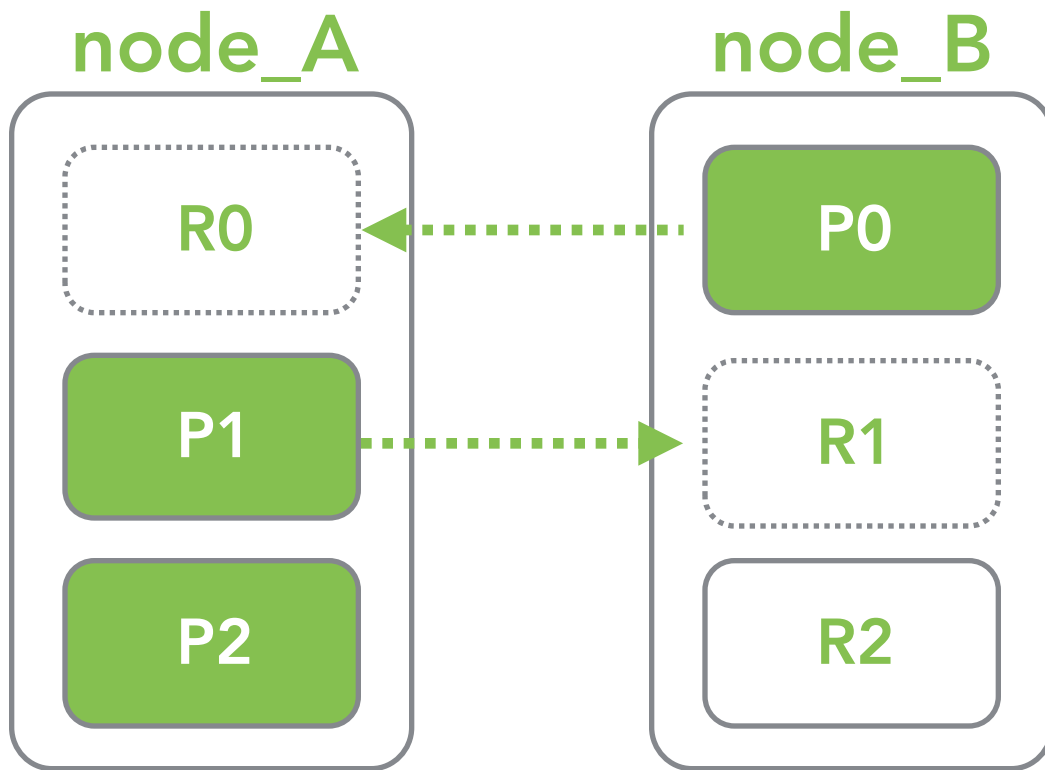
# replica $\Rightarrow$ primary



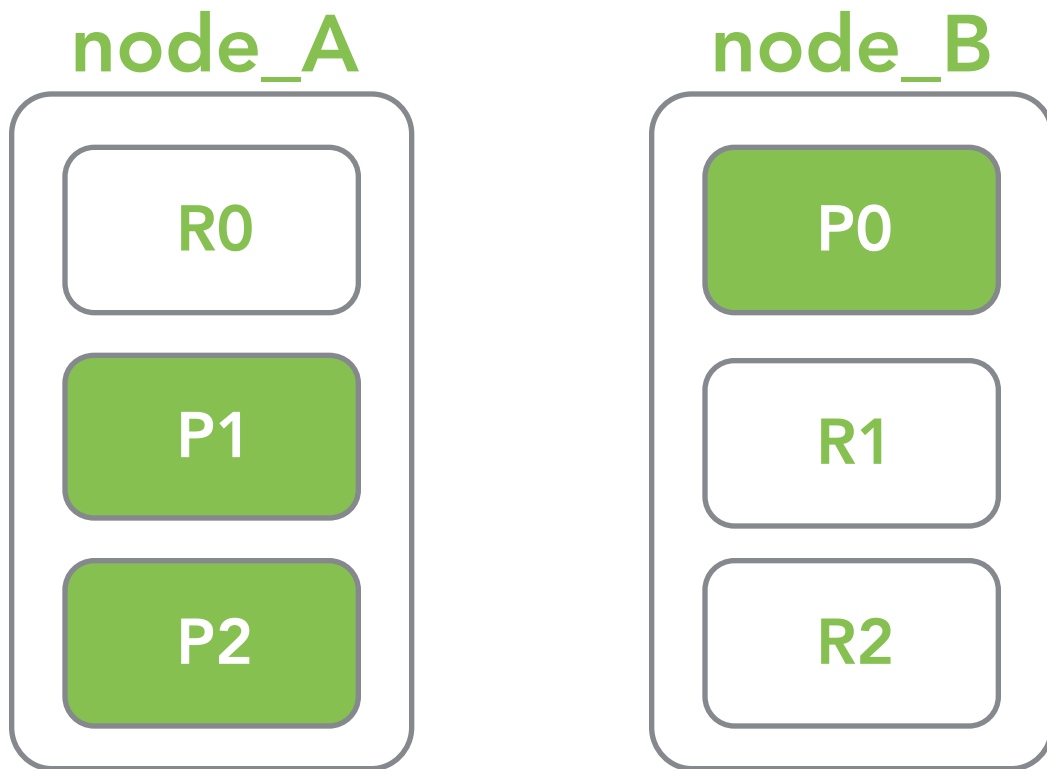
# replica $\Rightarrow$ primary



# allocate replicas



# rebalanced



# primary shard

- just a role
- receives doc changes first
- forwards new doc to replicas in parallel
- number of primaries fixed

# replica shard

- copy of primary shard
  - serves read/search requests
  - number of replicas can be changed
  - more replicas → more read throughput
- \*if you have more hardware\***



but...

but...

**who controls all this?**

# step 11:

## the master node



"Master Yoda" by [Gonzalo Martín](#) is licensed under [CC BY-SA 2.0](#)

# "node"

## running instance of elastic search

# "node"

## running instance of elastic search

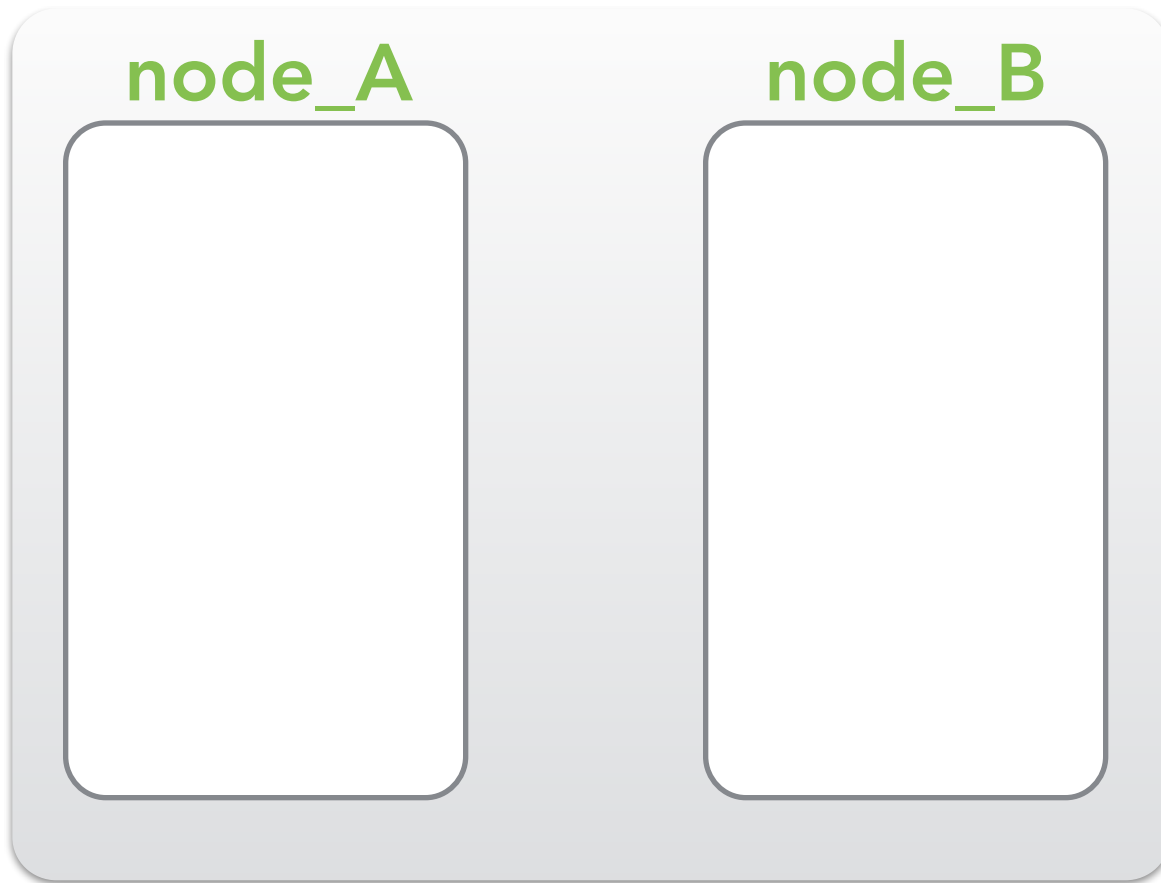
node\_A



# "cluster"

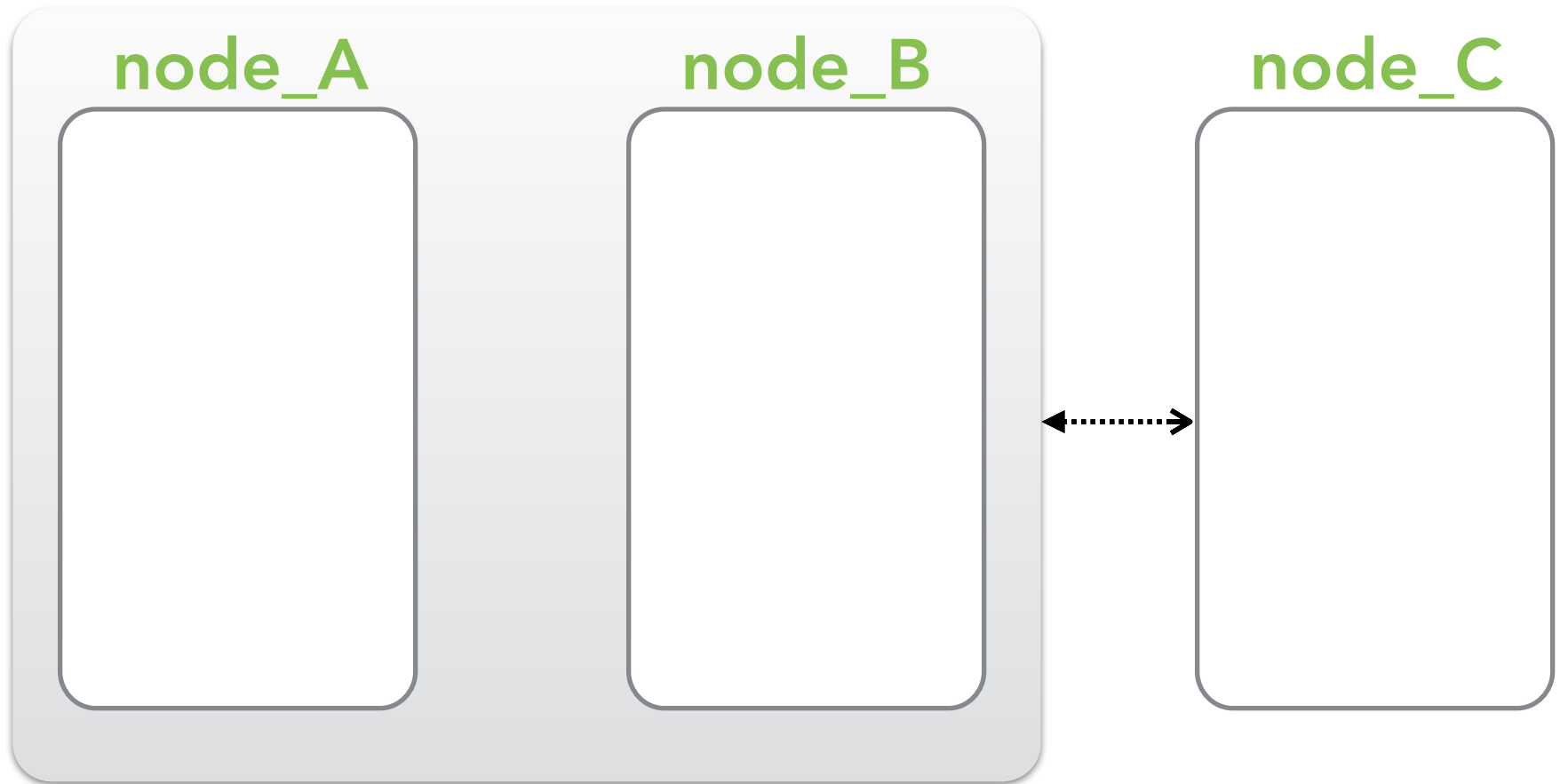
**one or more nodes  
with same cluster name  
working together**

# "cluster"





# discover a cluster with multicast/unicast



# discover a cluster with multicast/unicast

node\_A

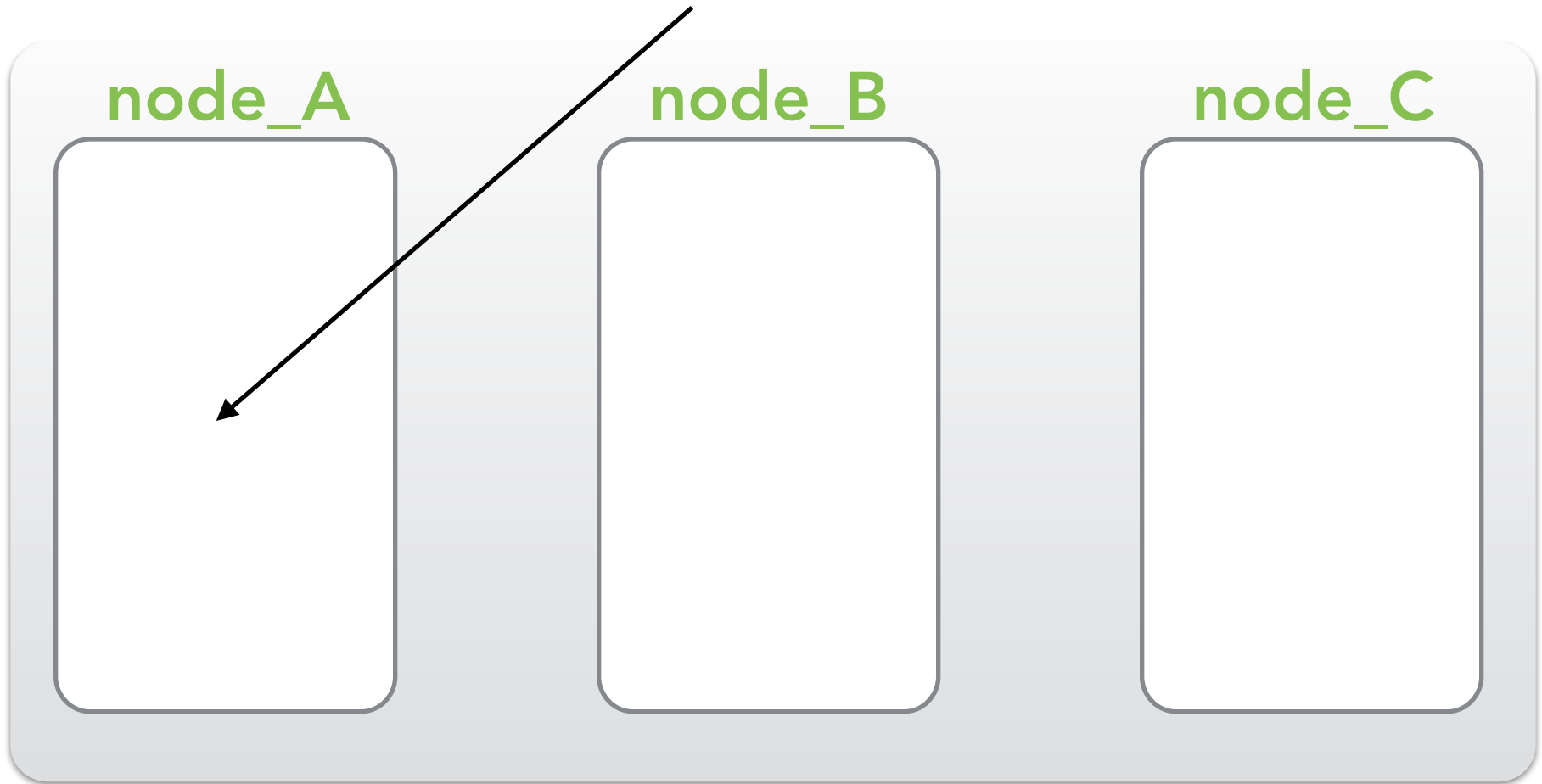


node\_B

node\_C

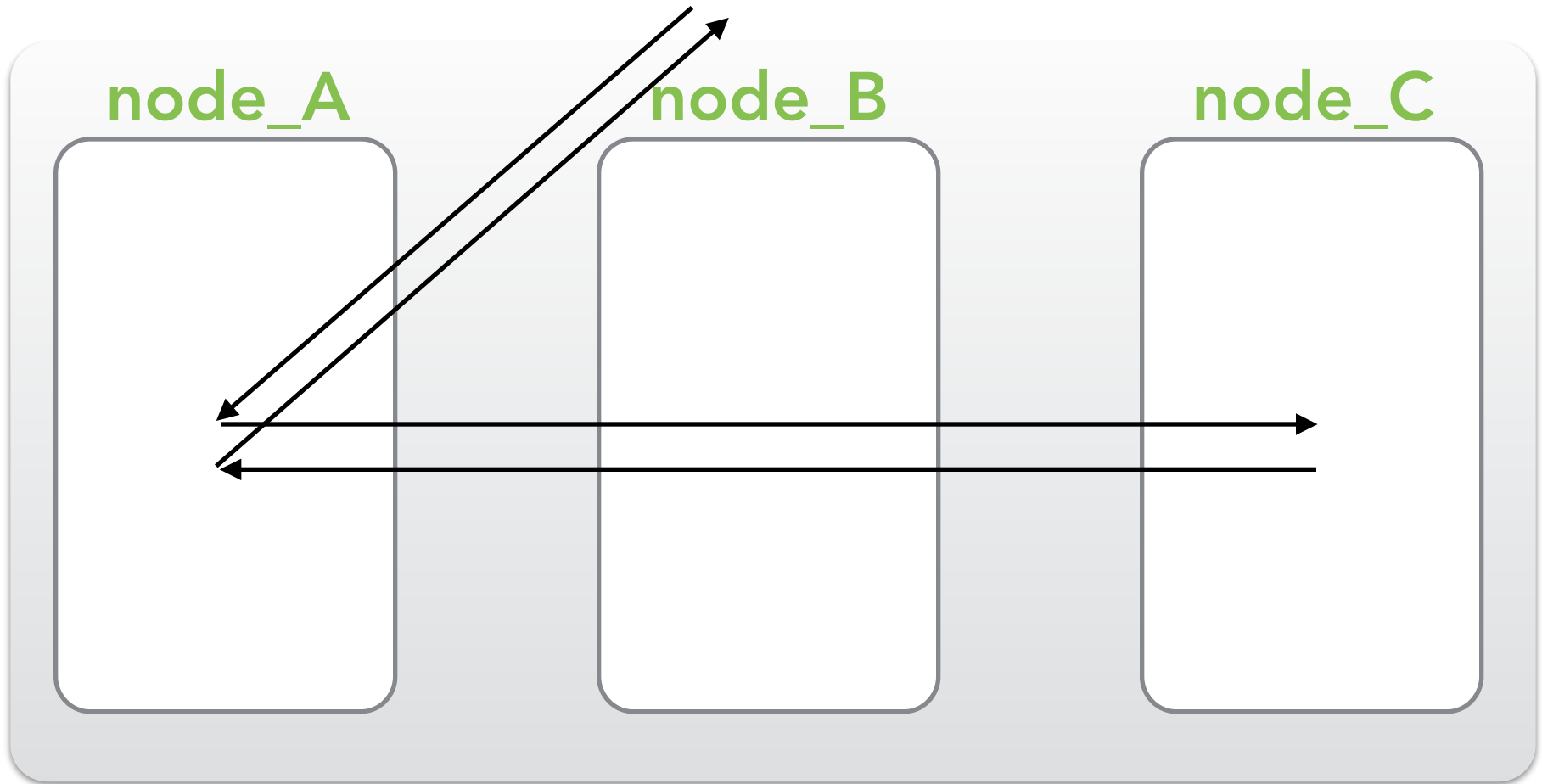
# request routing

send request to any node



# request routing

forwards to correct node



# how?

how?

**every node knows where  
every document is**

# cluster state

**every node knows where  
every document is**

# cluster state

## cluster level information



# cluster state

**cluster level information**

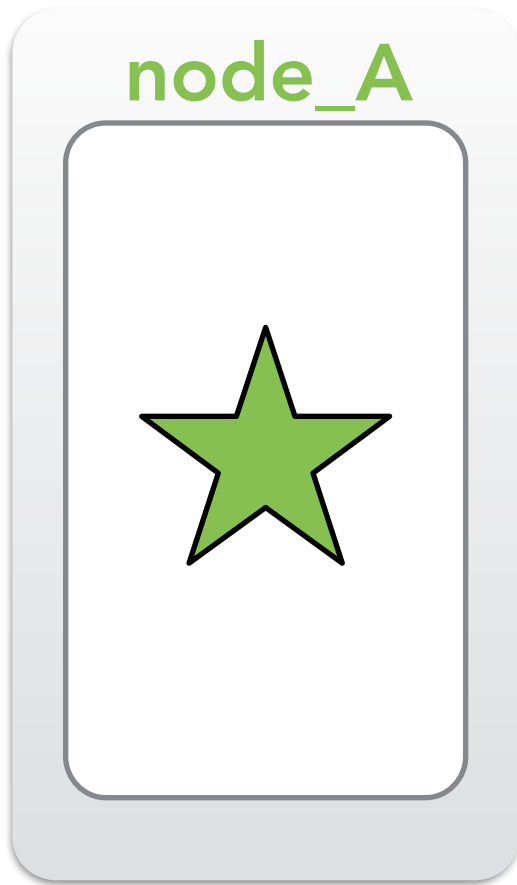
**indices  $\Leftrightarrow$  shards  $\Leftrightarrow$  nodes**

# cluster state

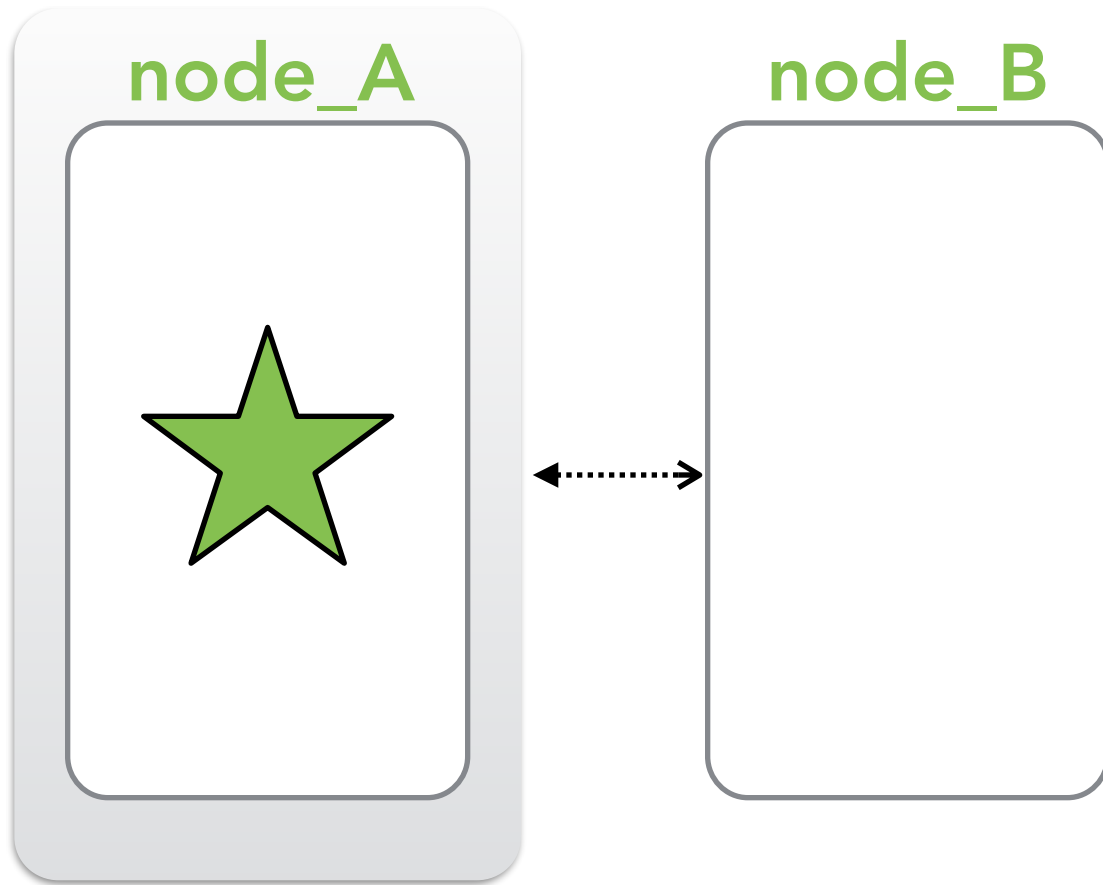
**can only be updated by  
the master node**

# master node

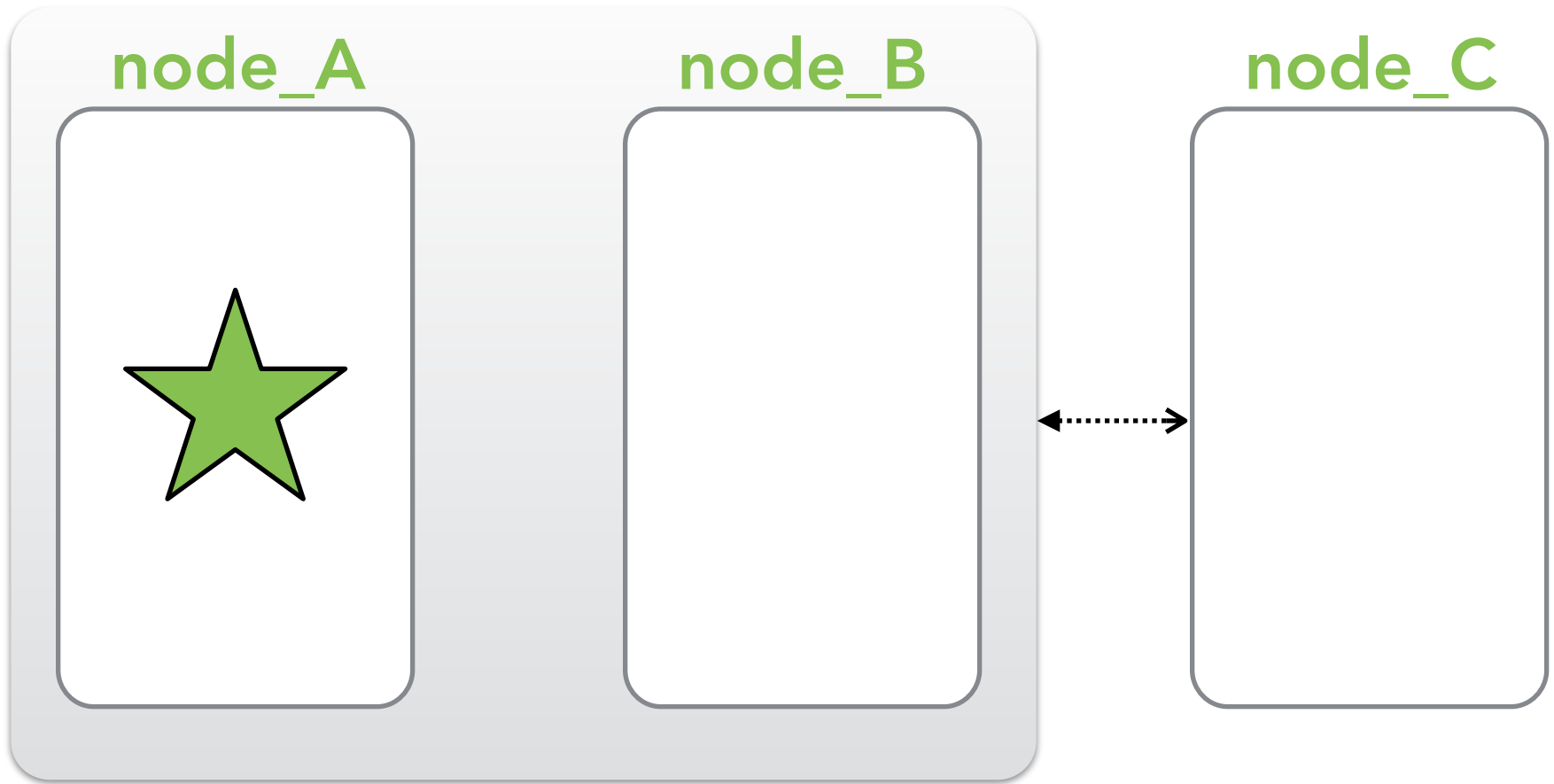
## elected when cluster forms



# master node

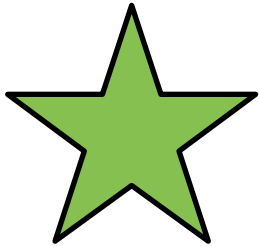


# master node



# master node

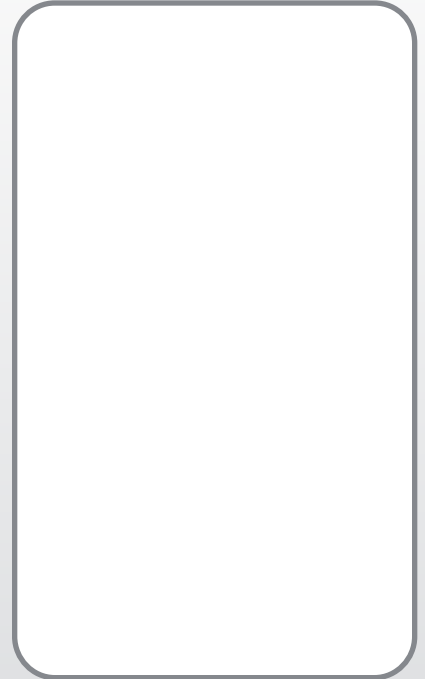
node\_A



node\_B



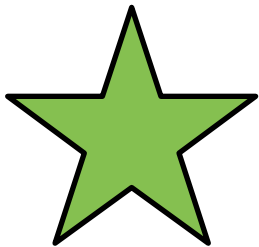
node\_C



# master node

## just a role

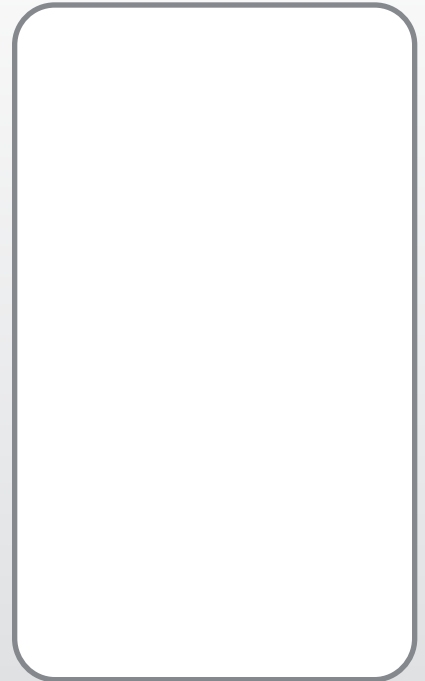
node\_A



node\_B



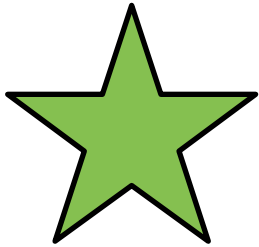
node\_C



# master node

## re-elected if master fails

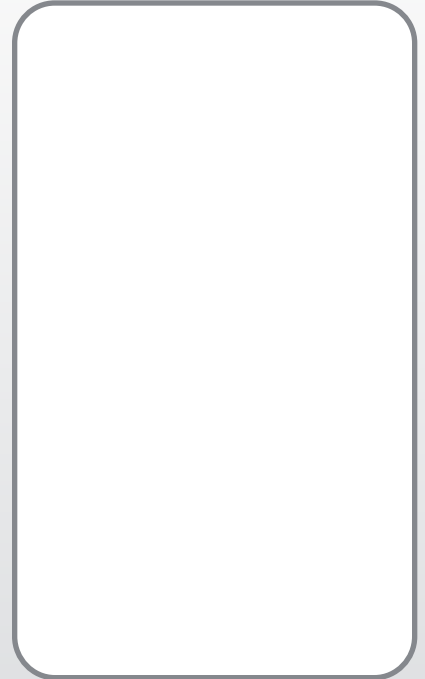
node\_A



node\_B



node\_C

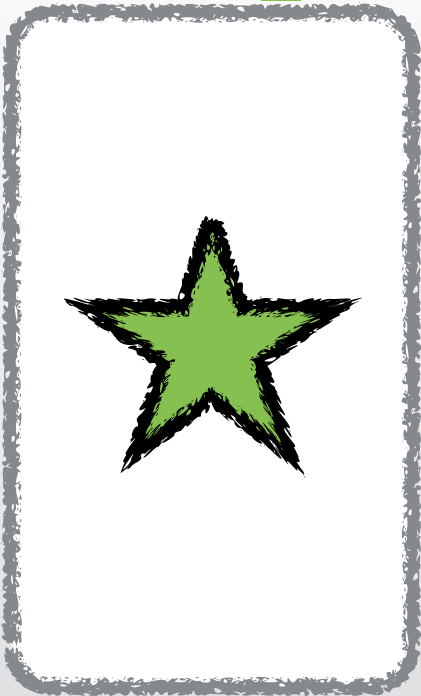




# master node

## re-elected if master fails

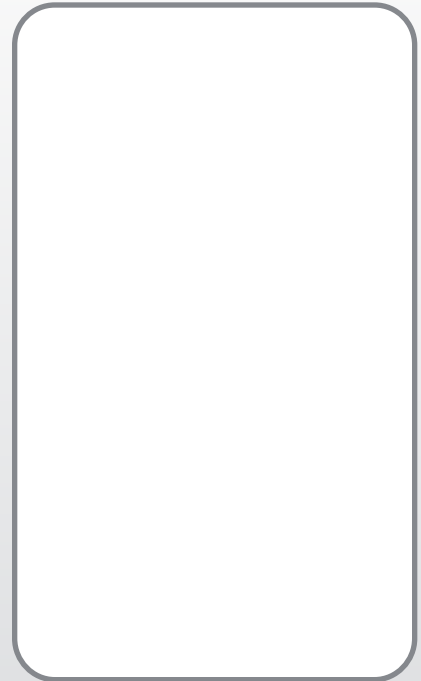
node\_A



node\_B

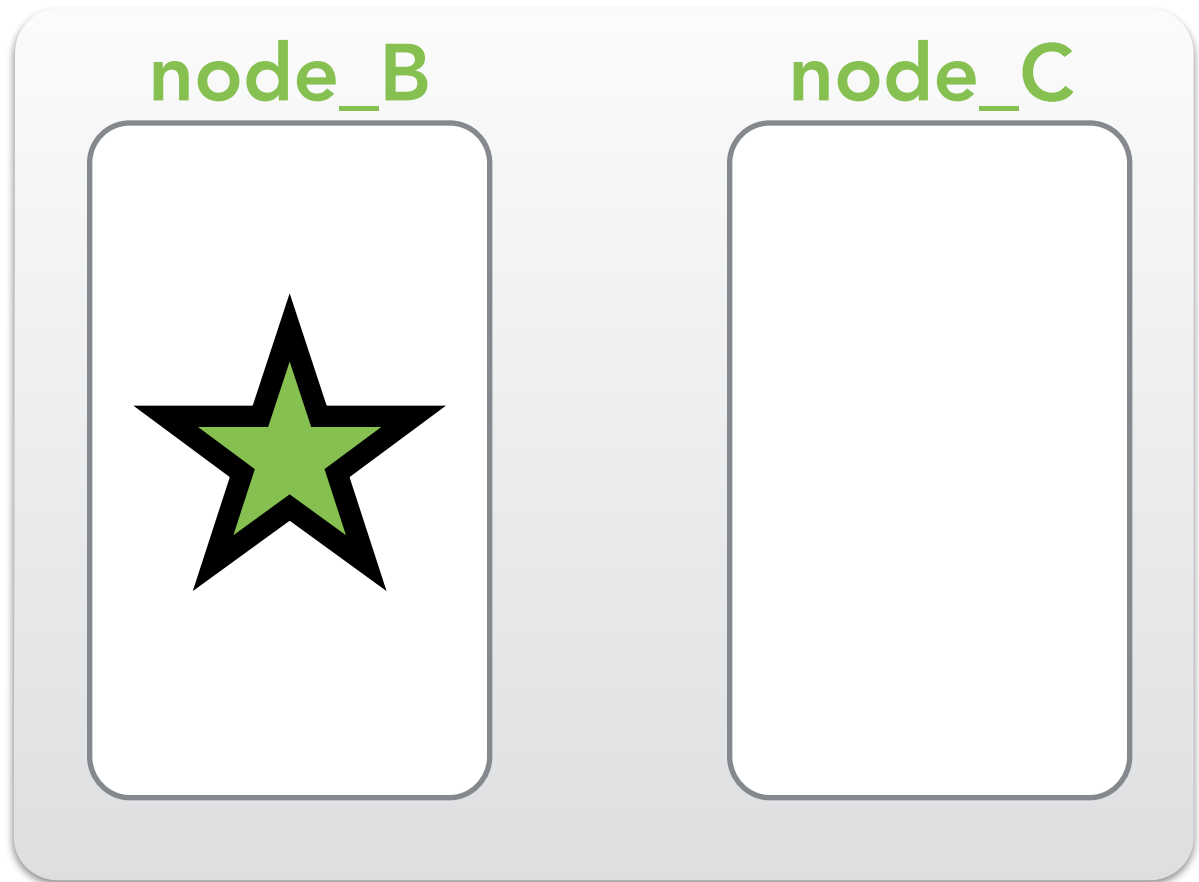


node\_C



# master node

## re-elected if master fails



# master node

**only manages  
cluster level changes**

# master node


**not doc-level  
get/put/search**

# the result?

# **distributed real-time search & analytics**

**which works in the same way  
on your laptop...**





**...as on your  
1,000 node cluster**



# who is using it?



WIKIPEDIA  
The Free Encyclopedia

- full text search
- highlighted search snippets
- ***search-as-you-type***
- ***did-you-mean*** suggestions



- **combine visitor logs with social network data**
- **real-time feedback to editors**



- combines full text search with geolocation
- uses ***more-like-this*** to find related questions and answers

# GitHub



- **search repositories, users, issues, pull requests**
- **search 130 billion lines of code**
- **track all alerts, events, logs**



- **index and analyse  
5TB of log data every day**

thank you

@kimchy

[elasticsearch.org/downloads](http://elasticsearch.org/downloads)

[elasticsearch.com/support](http://elasticsearch.com/support)

[elasticsearch.com/jobs](http://elasticsearch.com/jobs)