What’s evolving in the Elasticsearch and Lucene

Berlin Buzzwords 2019
Simon Willnauer
June 18th, 2019

Queries

Query	Before	After	Improvement
Fuzzy	46 qps		
Phrase	4 qps		
Bool AND	9.3 qps		
Bool OR	3.3 qps		
Term	**33 qps**		

Query	Before	After	Improvement
Fuzzy	46 qps	59 qps	28%
Phrase	4 qps	7 qps	87%
Bool AND	9.3 qps	23.5 qps	247%
Bool OR	3.3 qps	9.8 qps	292%
Term	**33 qps**	**1,160 qps**	**3,700%**

MAGIC WAND

Weak-AND

"query" : "elasticsearch and lucene"

max_score(and) == 1
max_score(elasticsearch) == 3
max_score(lucene) == 5

Weak-AND

Min top-10 score	and (1)	elasticsearch (3)	lucene (5)
<=1	✓	✓	✓
> 1 and <= 4	✗	✓	✓
> 4 and <= 9	✗	✗	✓
> 9	✗	✗	✗

"aggs": { ... }

"track_total_hits": true

Weak-AND

```
"hits": {
  "total": 123456789,
  "hits": [ ... ]
}
```

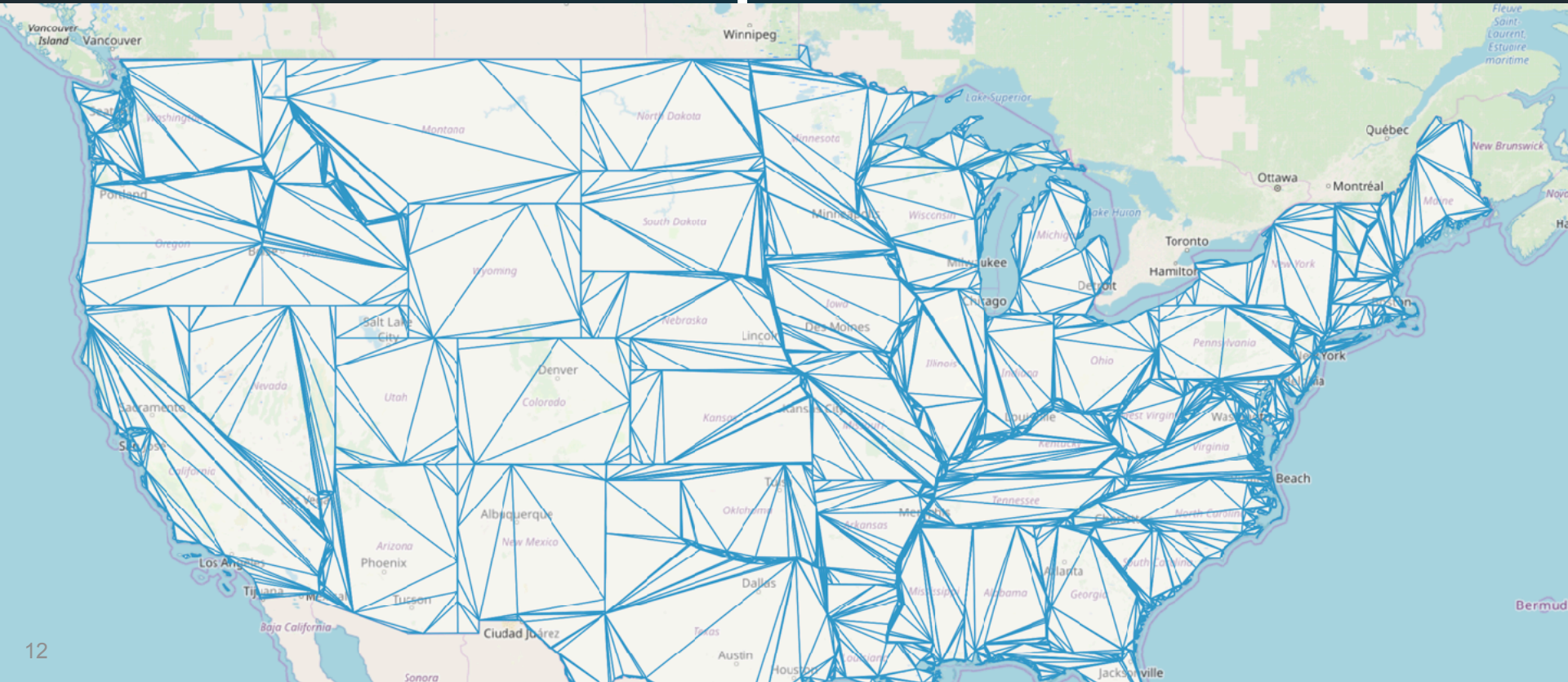
```
"hits": {
  "total": {
    "value": 10000,
    "relation": "gte"
  },
  "hits": [ ... ]
}
```

Geoshapes

BKD Trees

- v2.3: 1 dim, for numbers and dates
- v5.0: 2 dim, for geopoints
- v5.2: 2 dim, for number & date ranges
- v6.7: 7 dim, for geoshapes

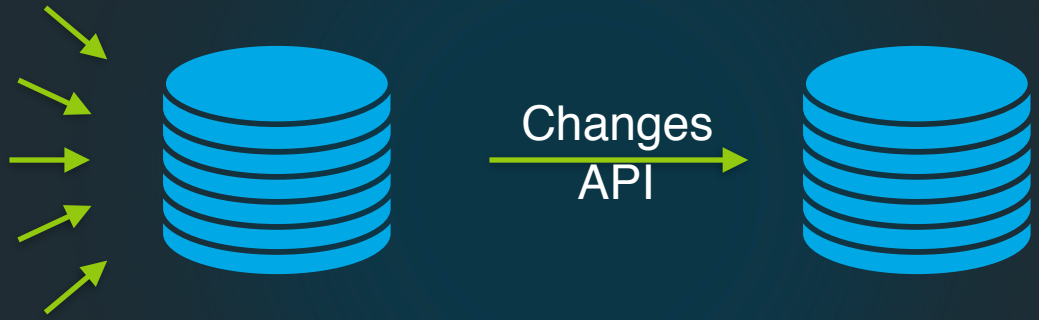
BKD Geoshapes



BKD Geoshapes

- Accurate to 1mm, vs 50m
- Index is 60% smaller
- Indexing 60% faster
- Queries 50% faster
- Plus BKD GeoPoints 80% faster indexing

Changes API


Changes
API

Zen 2

minimum_master_nodes: 2

minimum_master_nodes: 2

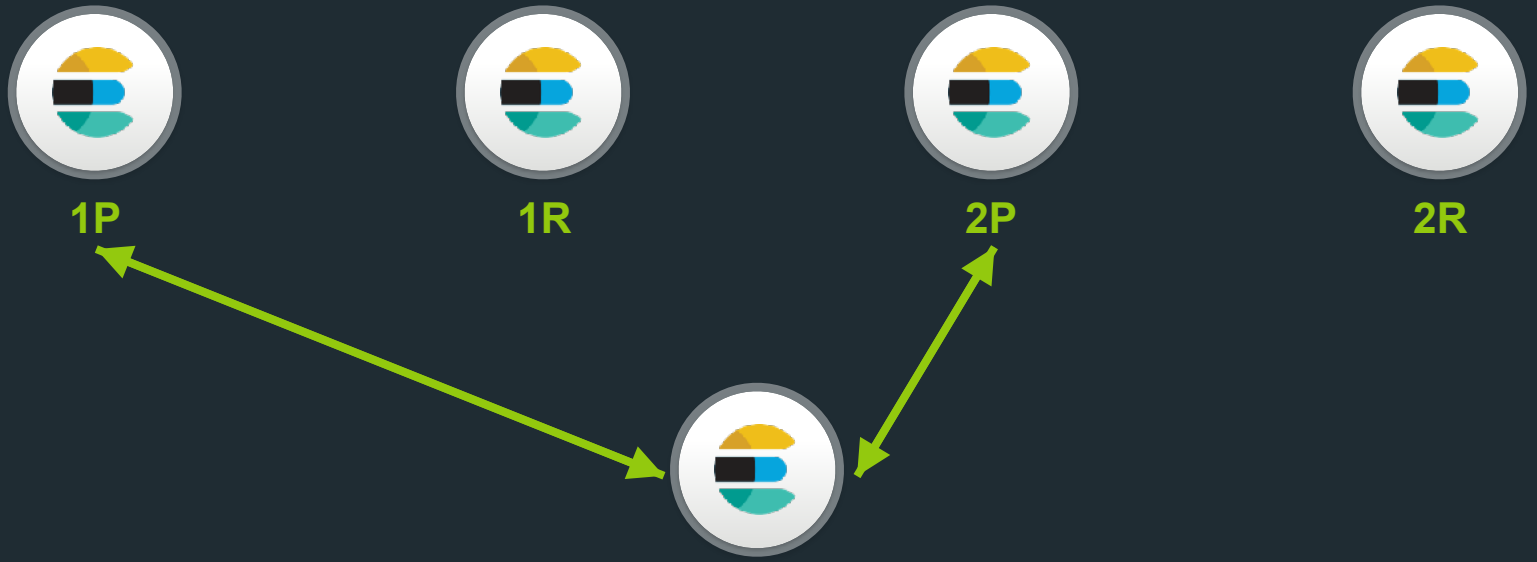
minimum_master_nodes: 2

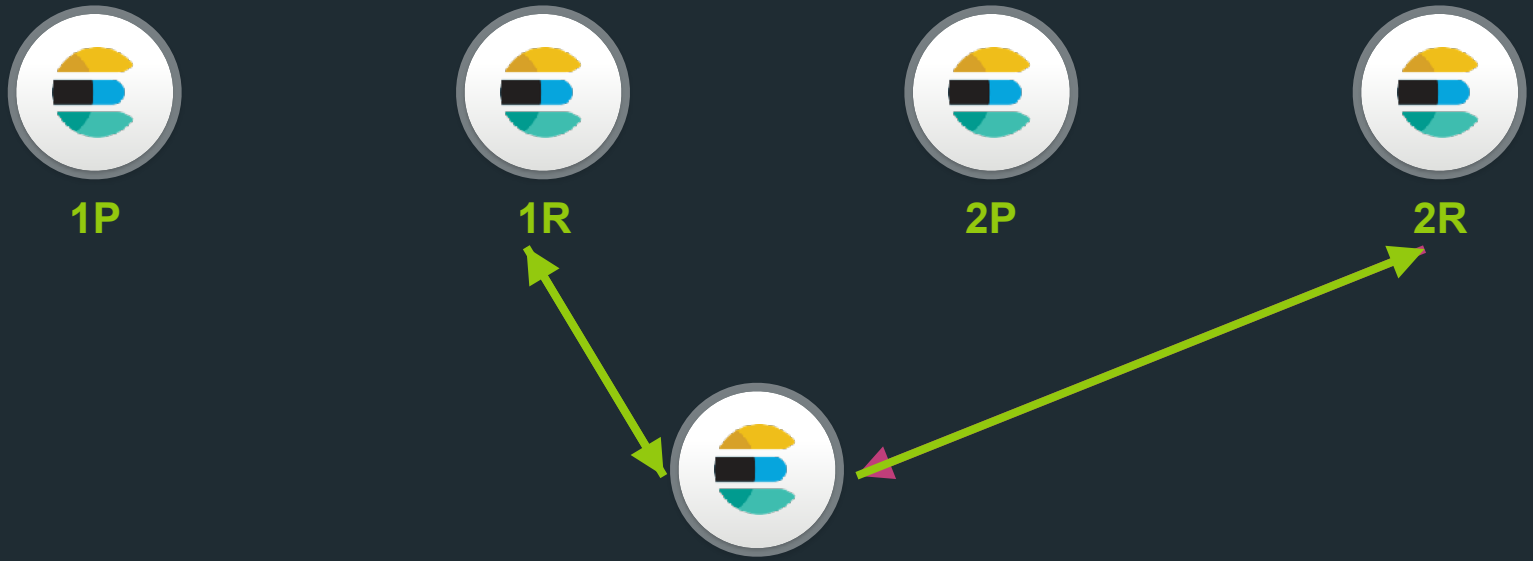
minimum_master_nodes: 1

~~minimum_master_nodes: 1~~

cluster.initial_master_nodes

Adaptive Replica Selection


1P
1R
2P
2R


1P
1R
2P
2R

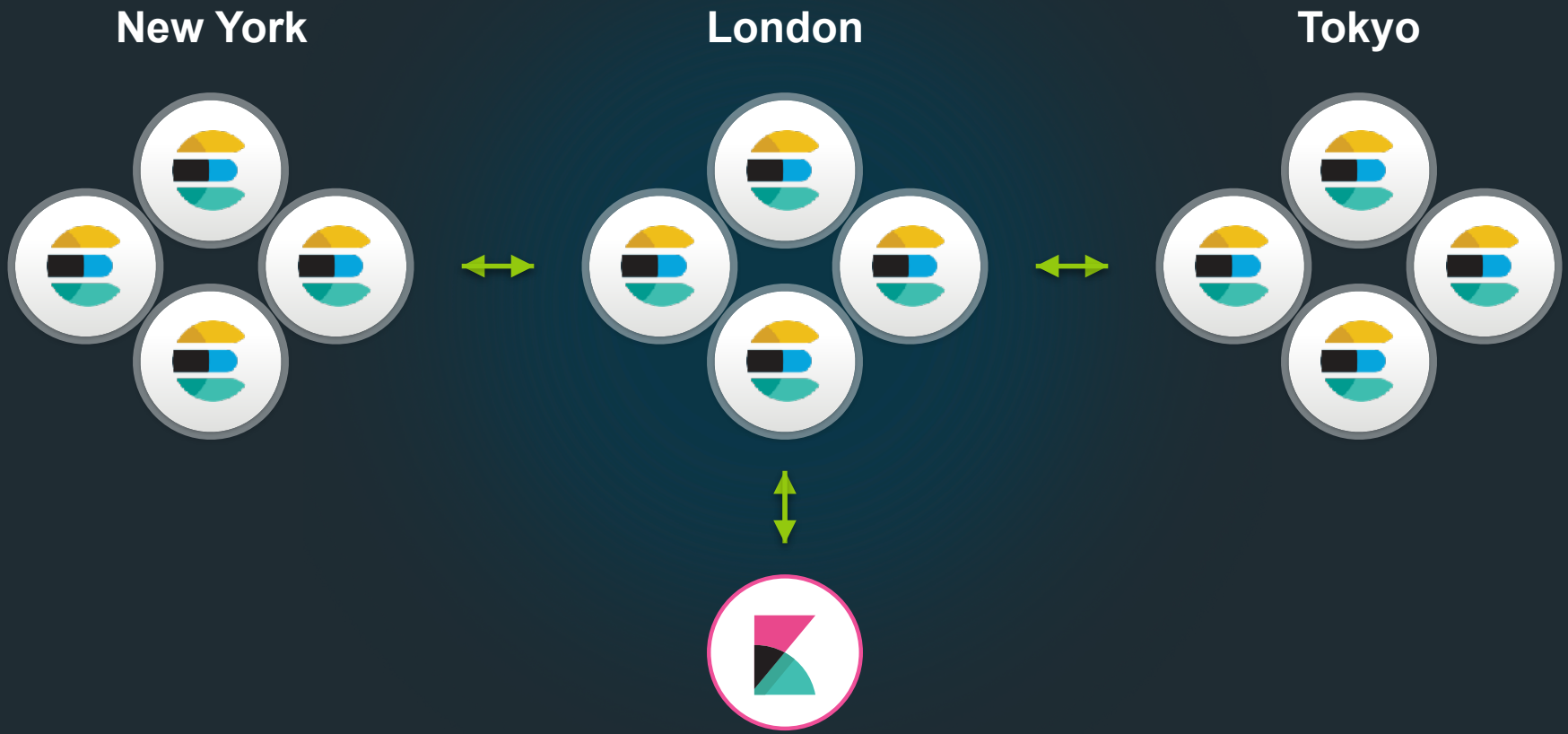
1P

1R

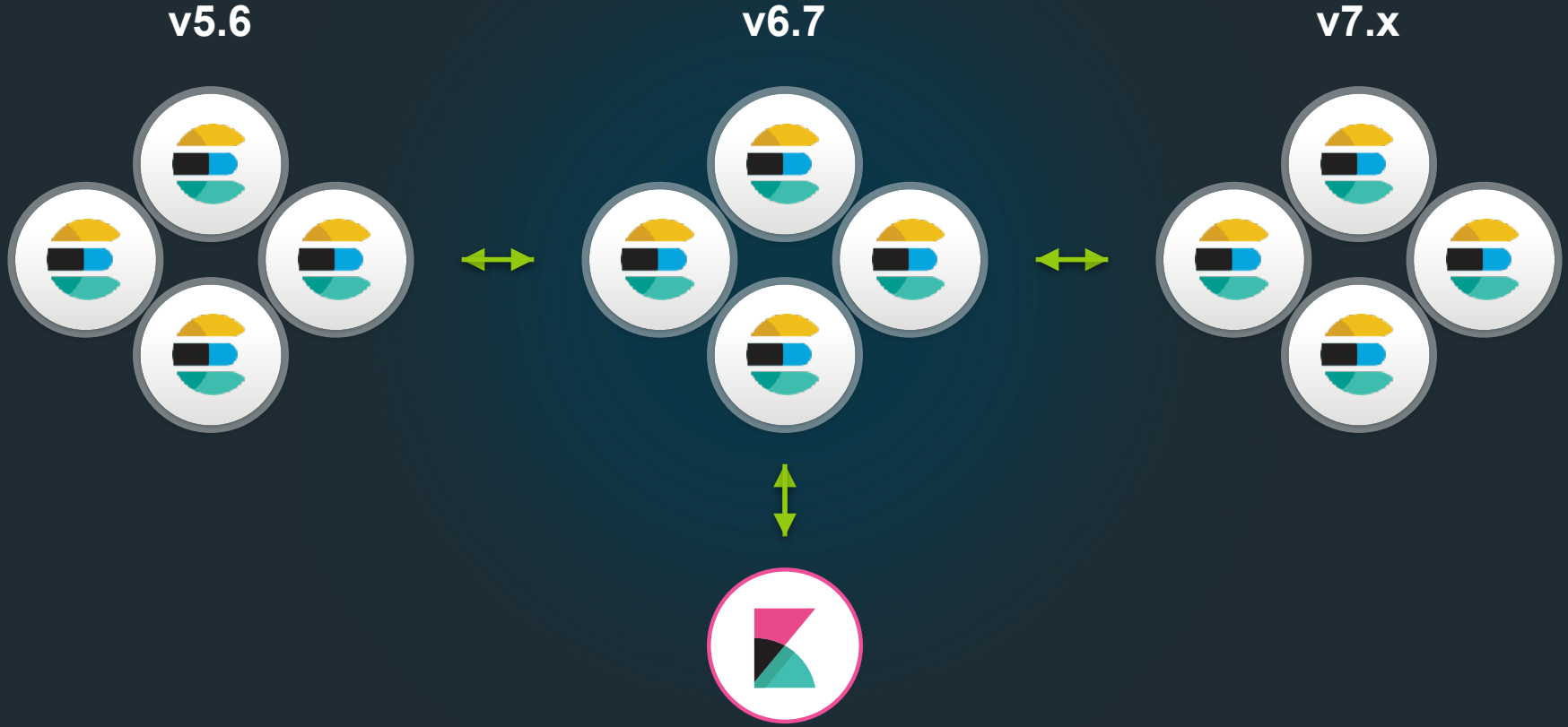
2P

2R

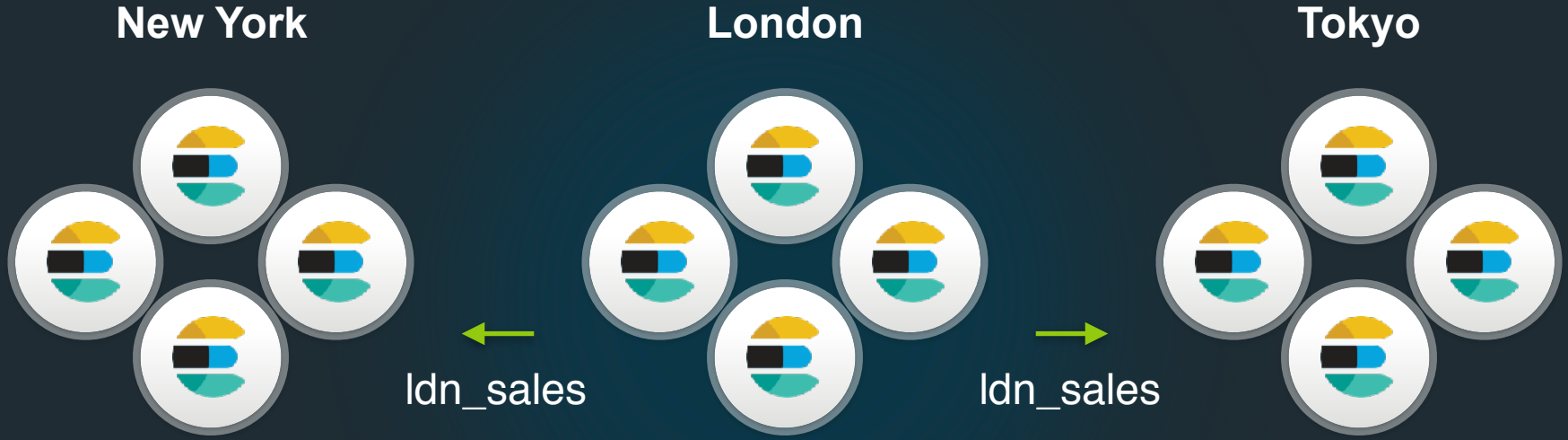
Cross Cluster Search

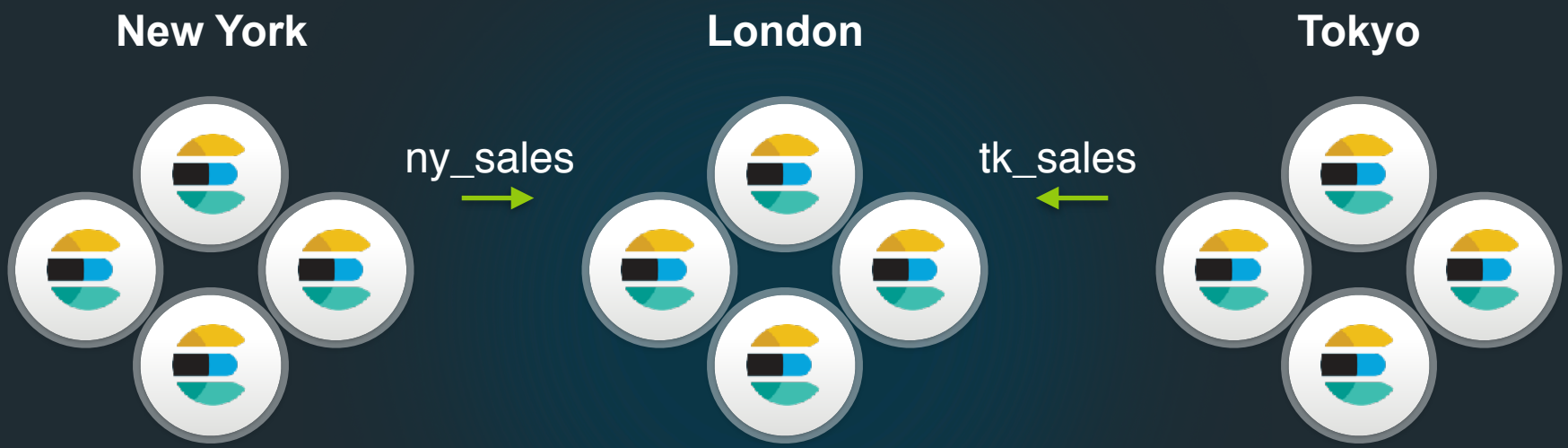

New York
London
Tokyo

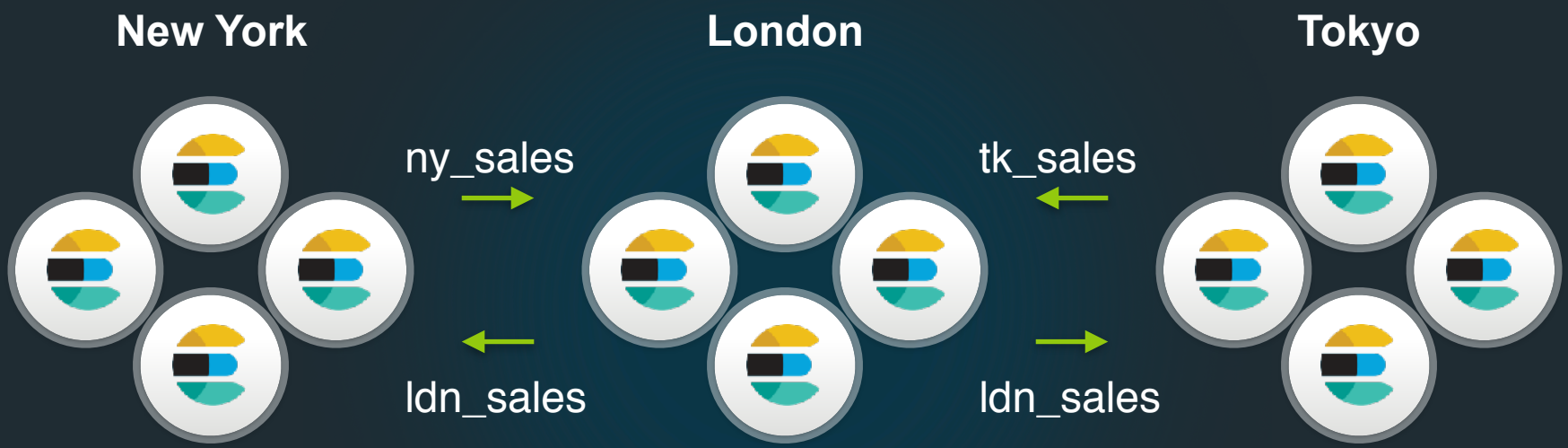
Three Major Versions



Cross Cluster Replication


New York
London
Tokyo
ldn_sales
ldn_sales


New York
London
Tokyo
ny_sales
tk_sales


New York
London
Tokyo
ny_sales
ldn_sales
tk_sales
ldn_sales

Cross Cluster Replication

Follower indices | Auto-follow patterns

A follower index replicates a leader index on a remote cluster.

Create a follower index

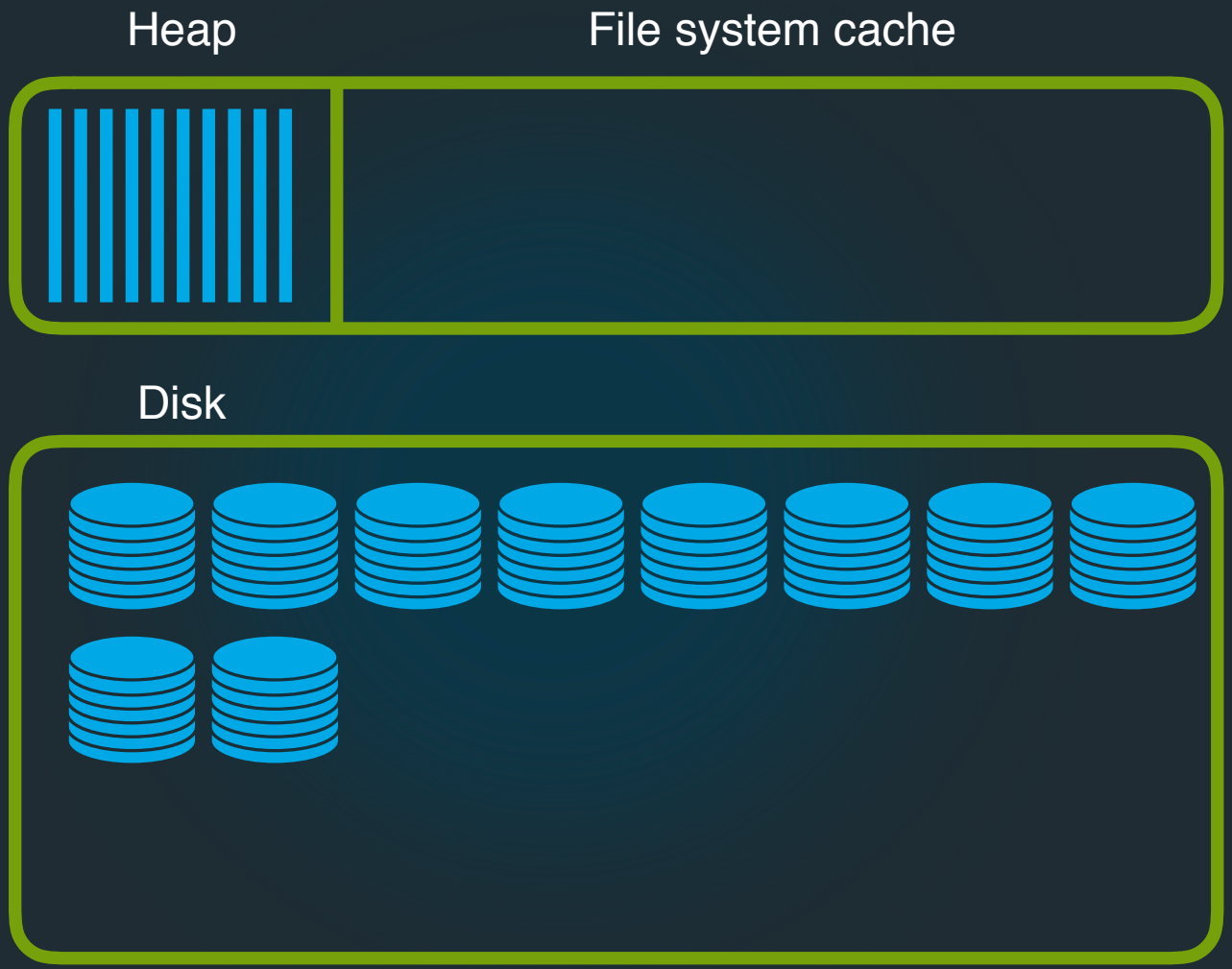
Search...

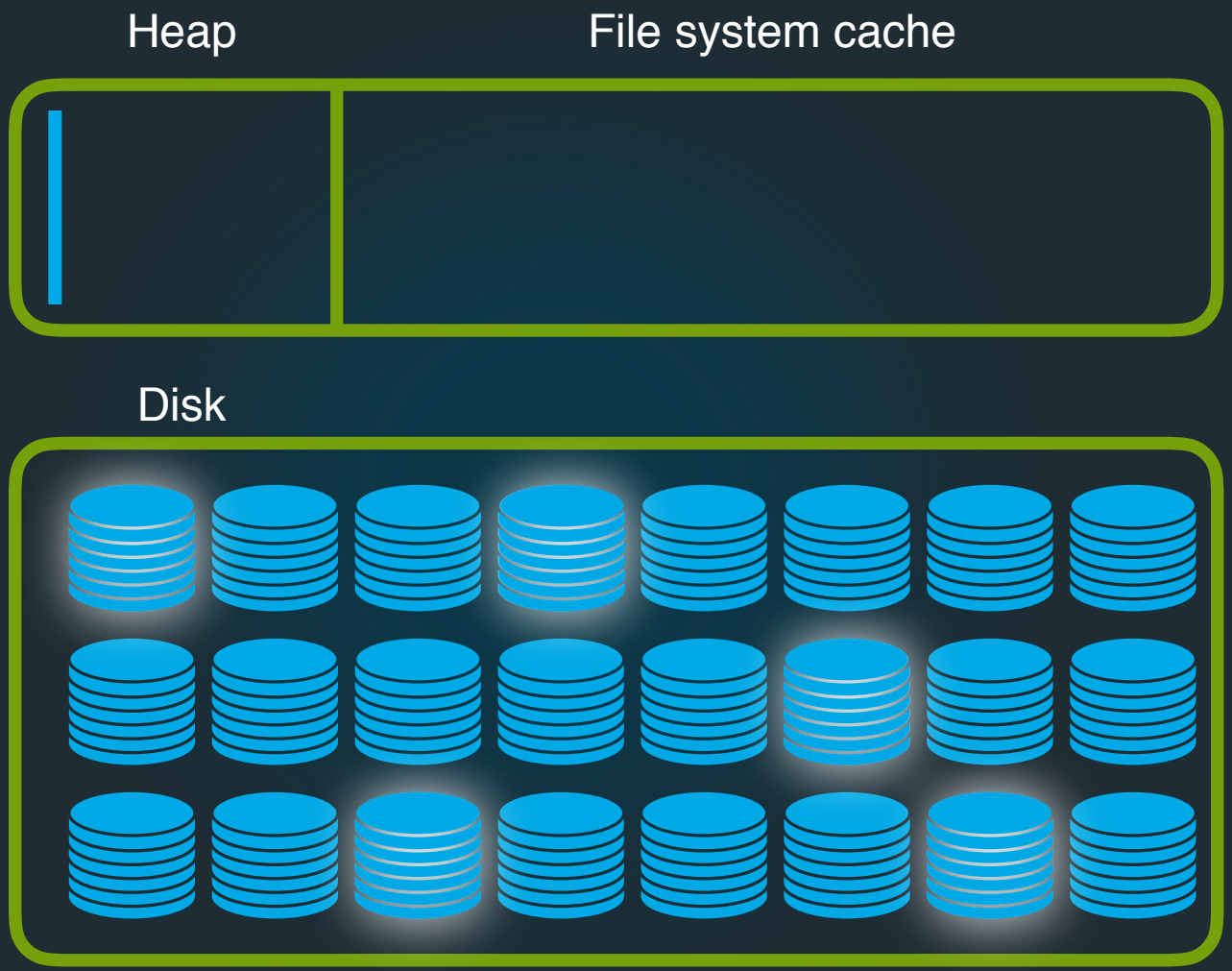
Name	Status	Remote cluster	Leader index	Actions
london_ny_sales	Active	NewYork	ny_sales	▫▫▫
london_tokyo_sales	Active	Tokyo	tokyo_sales	▫▫▫

Rows per page: 20

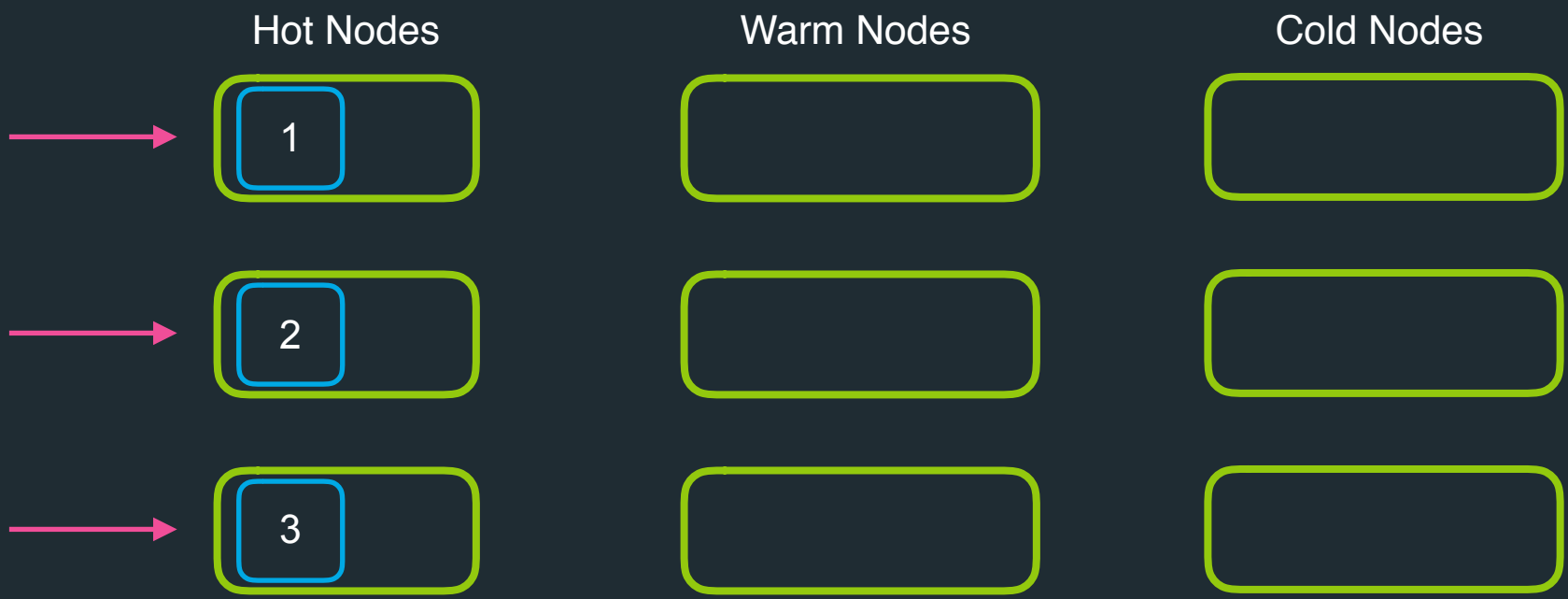
elastic

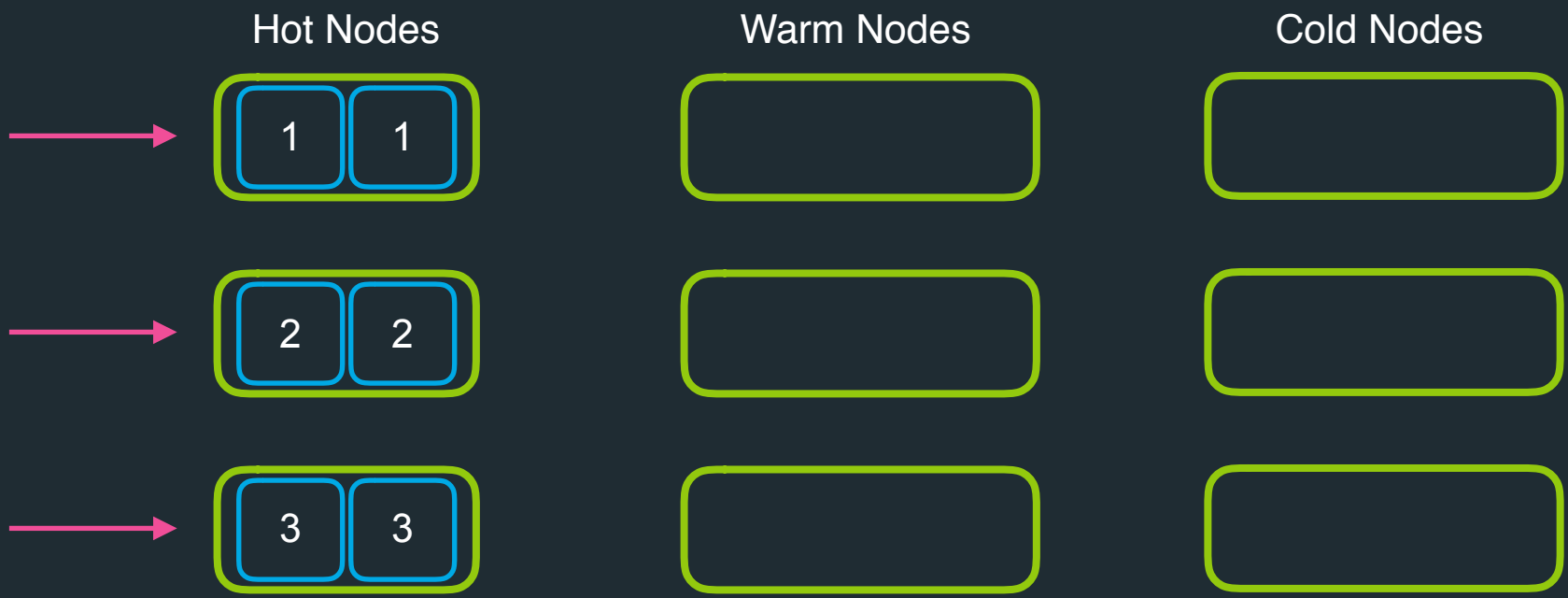
Frozen Indices

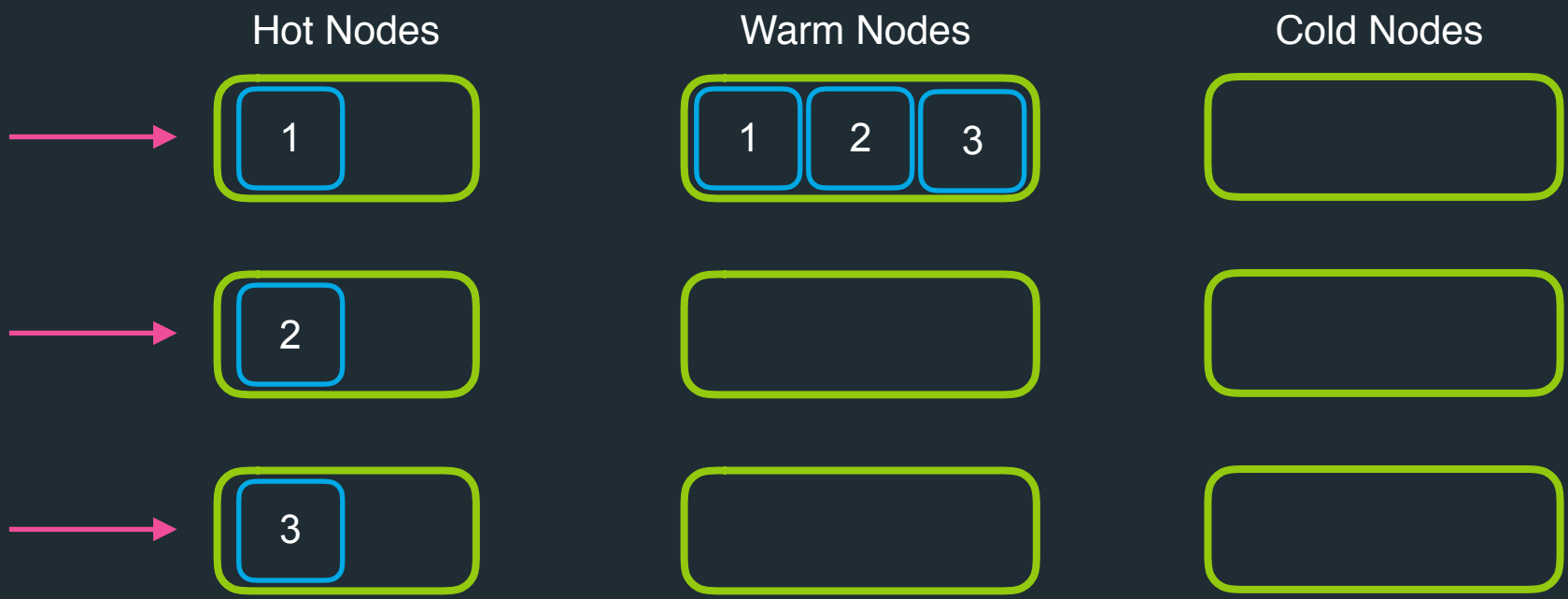

Heap
File system cache
Disk

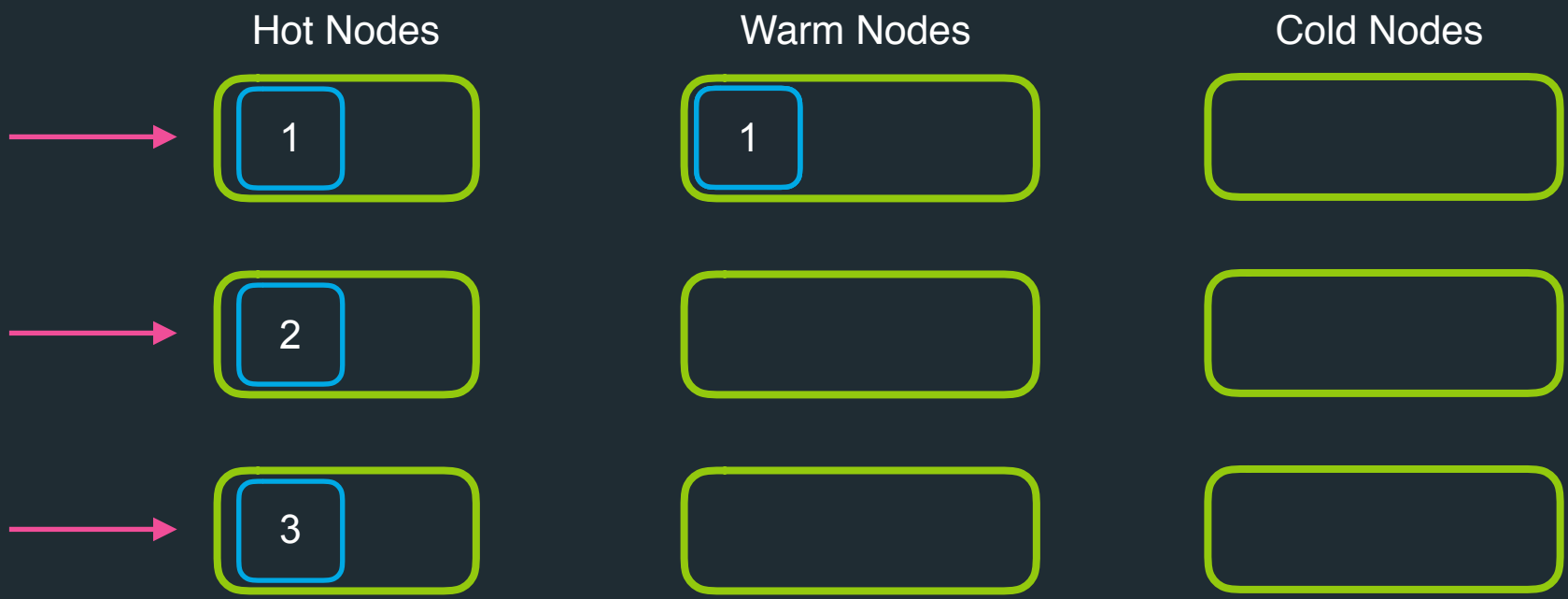

Heap
File system cache
Disk

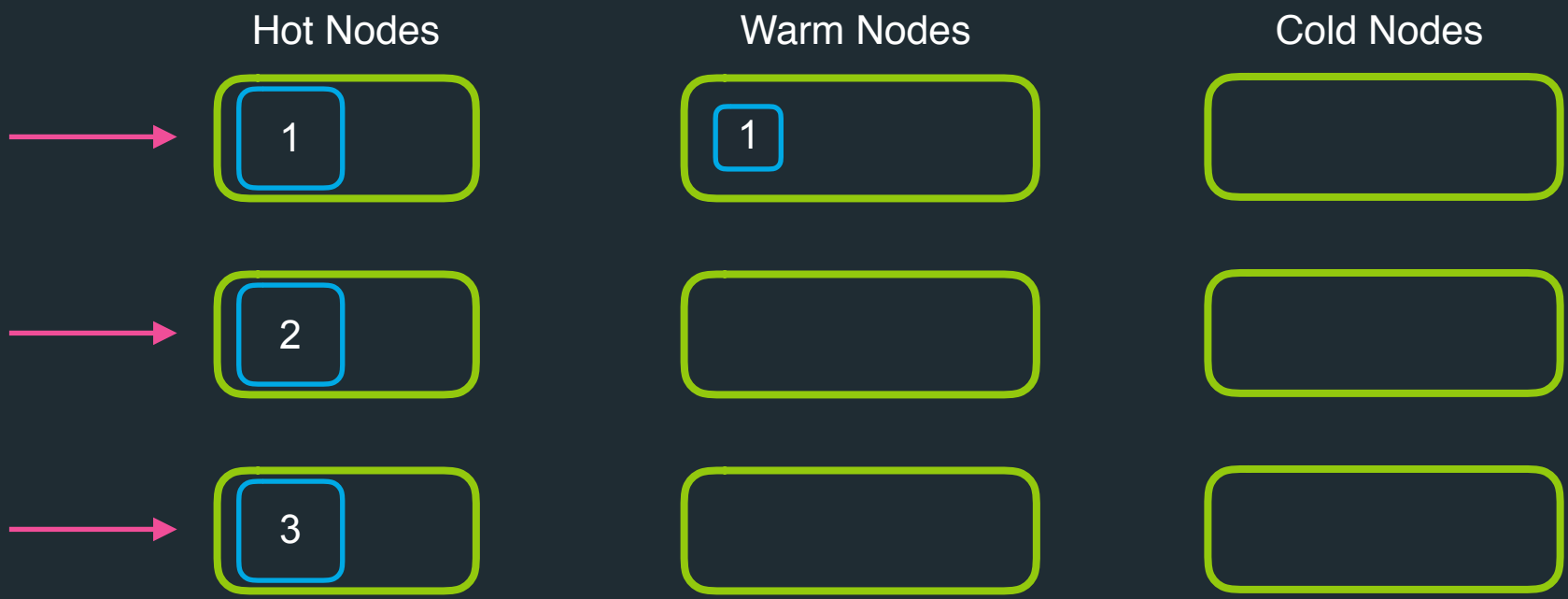
Index Lifecycle Management

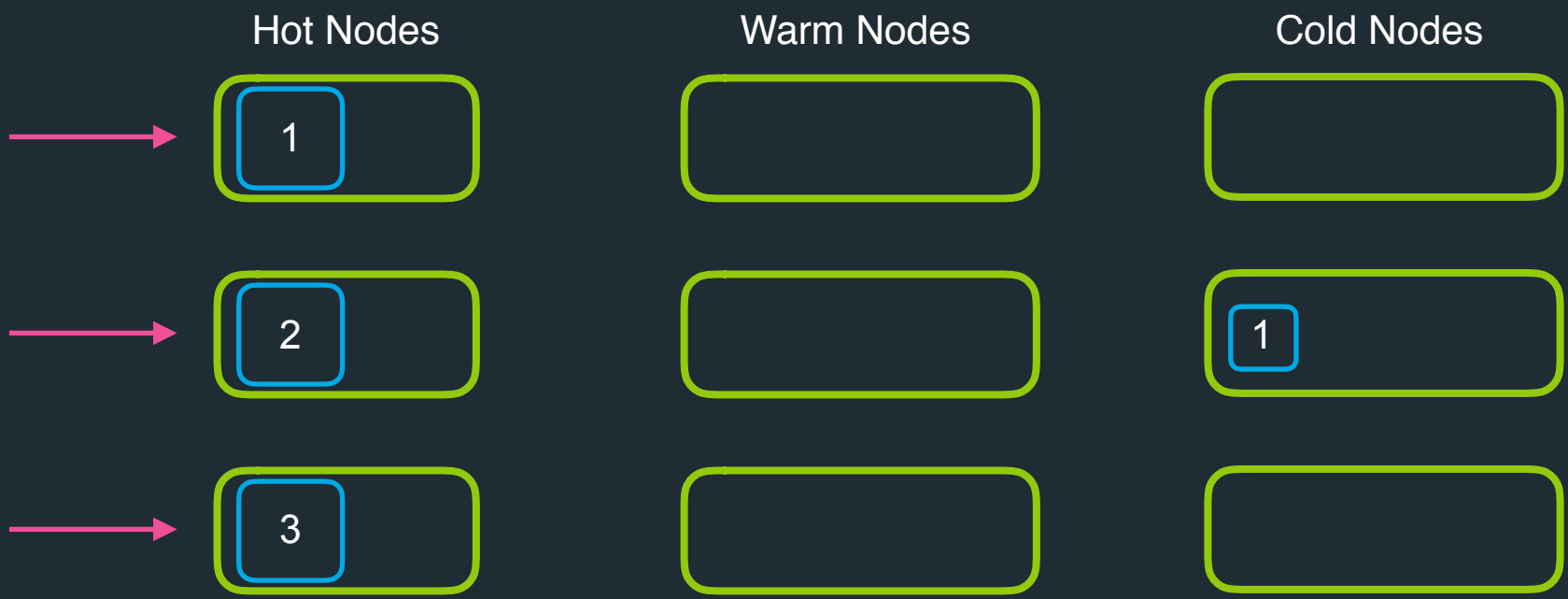

Hot Nodes
Warm Nodes
Cold Nodes
1
2
3

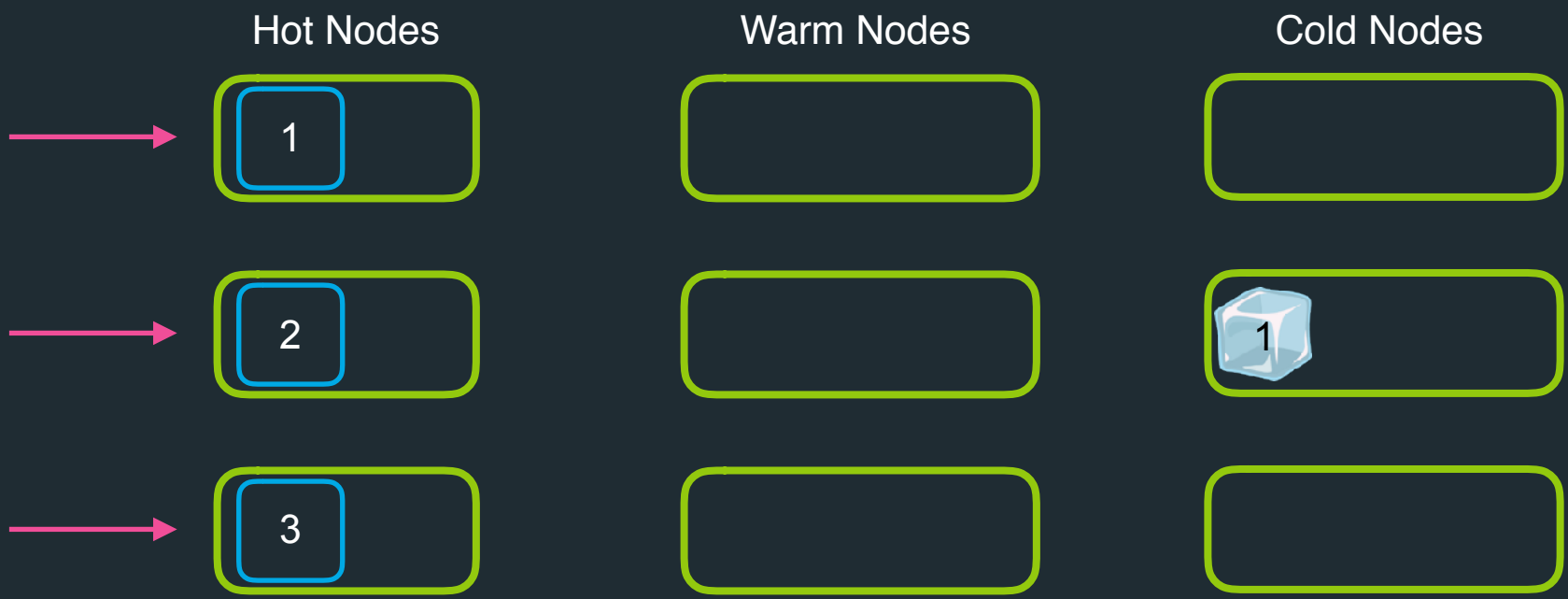

Hot Nodes
Warm Nodes
Cold Nodes
1
1
2
2
3
3

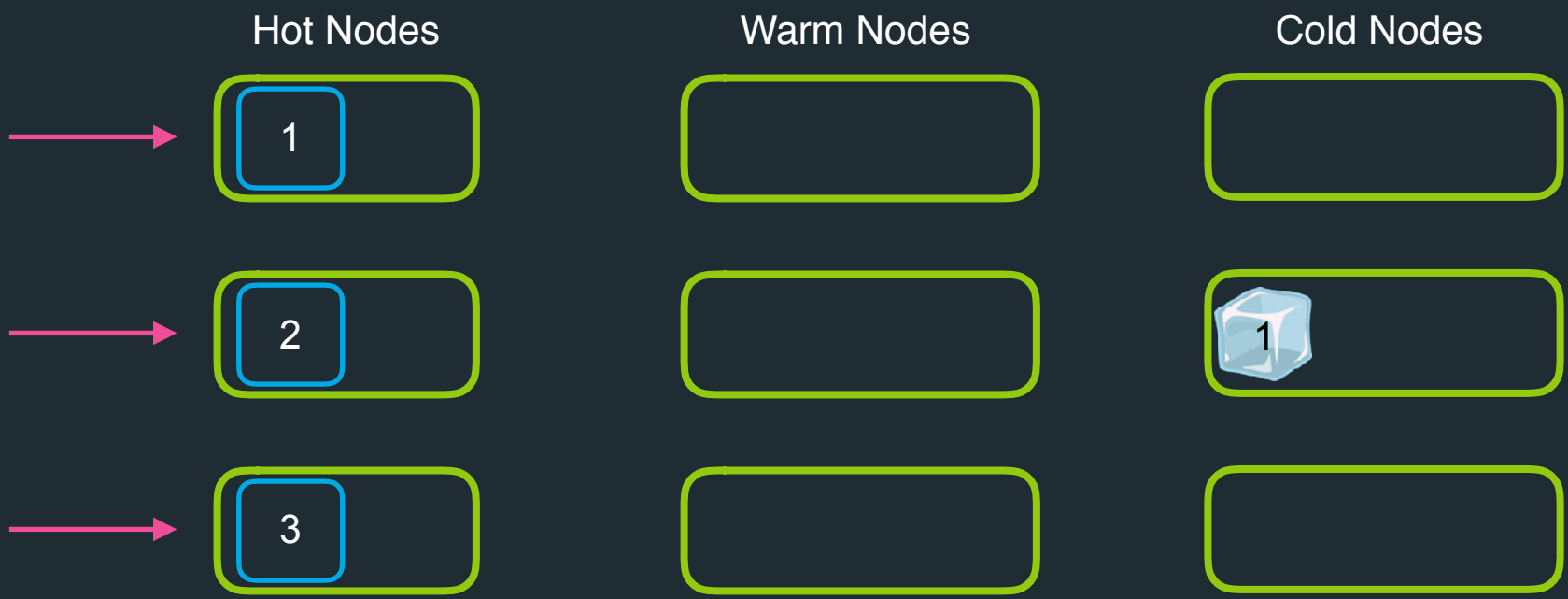

Hot Nodes
Warm Nodes
Cold Nodes
1
2
3
1
2
3


Hot Nodes
Warm Nodes
Cold Nodes
1
1
2
3


Hot Nodes
Warm Nodes
Cold Nodes
1
2
3
1


Hot Nodes
Warm Nodes
Cold Nodes
1
2
3
1


Hot Nodes
Warm Nodes
Cold Nodes
1
2
3
1


Hot Nodes
Warm Nodes
Cold Nodes
1
2
3
1

Index lifecycle management

Select a template — 2 Configure a policy — 3 Review and save

Select or create a policy

An index lifecycle policy is a blueprint for transitioning your data over time. You can create a new policy or edit an existing policy and save it with a new name.

Existing policies: my_policy5 | Create new policy

Edit policy my_policy5

Configure the phases of your data and when to transition between them.

Hot phase

This phase is required. Your index is being queried and actively written to. You can optimize this phase for write throughput.

Enable rollover

If true, rollover the index when it gets too big or too old. The alias switches to the new index. Learn more

Maximum index size: 3 gigabytes

Maximum age: days

Warm phase

Nanosecond Timestamps

date:
2019/01/01 12:34:56.001

date_nanos:
2019/01/01
12:34:56.000000001

Full Text Search

index_prefixes:

qu, qui, quic, quick
br, bro, brow, brown
fo, fox, foxe, foxes

index_phrases:

the_quick
quick_brown
brown_fox
fox_jumped
jumped_over
over_the
the_lazy
lazy_dog

match_phrase:

“quick brown fox”

match_phrase_prefix:

“quick brown f*”

Intervals Query

the quick brown fox
jumped over the
lazy dog

quick…fox

the quick brown fox
jumped over the
lazy dog

quick…fox, “lazy dog”

the quick brown fox
jumped over the
lazy dog

(quick…fox, “lazy dog”)~5

the quick brown fox
jumped over the
lazy dog

Thank you, questions?