David Pilato
Developer | Evangelist
elastic

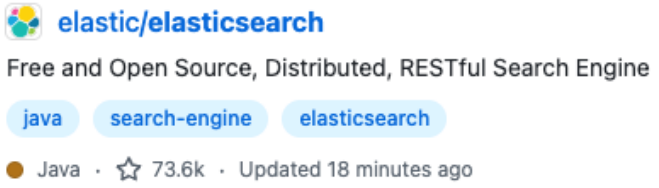
do MORE with stateLESS Elasticsearch

Sponsors
elastic
DEEZER

Elasticsearch

- Distributed, scalable, highly available, resilient search & analytics engine
- HTTP based JSON interface
- Based on Apache Lucene
- Not only grep or SQL's LIKE = '%quick%
  - Ranked results (BM25, recency, popularity), fuzzy matching
  - Complex search expressions
  - Spelling, Synonyms, Phrases, Stemming
- Timeseries, geospatial
- Vector search, (G)AI/ML, RAG search apps

github.com/elastic/elasticsearch



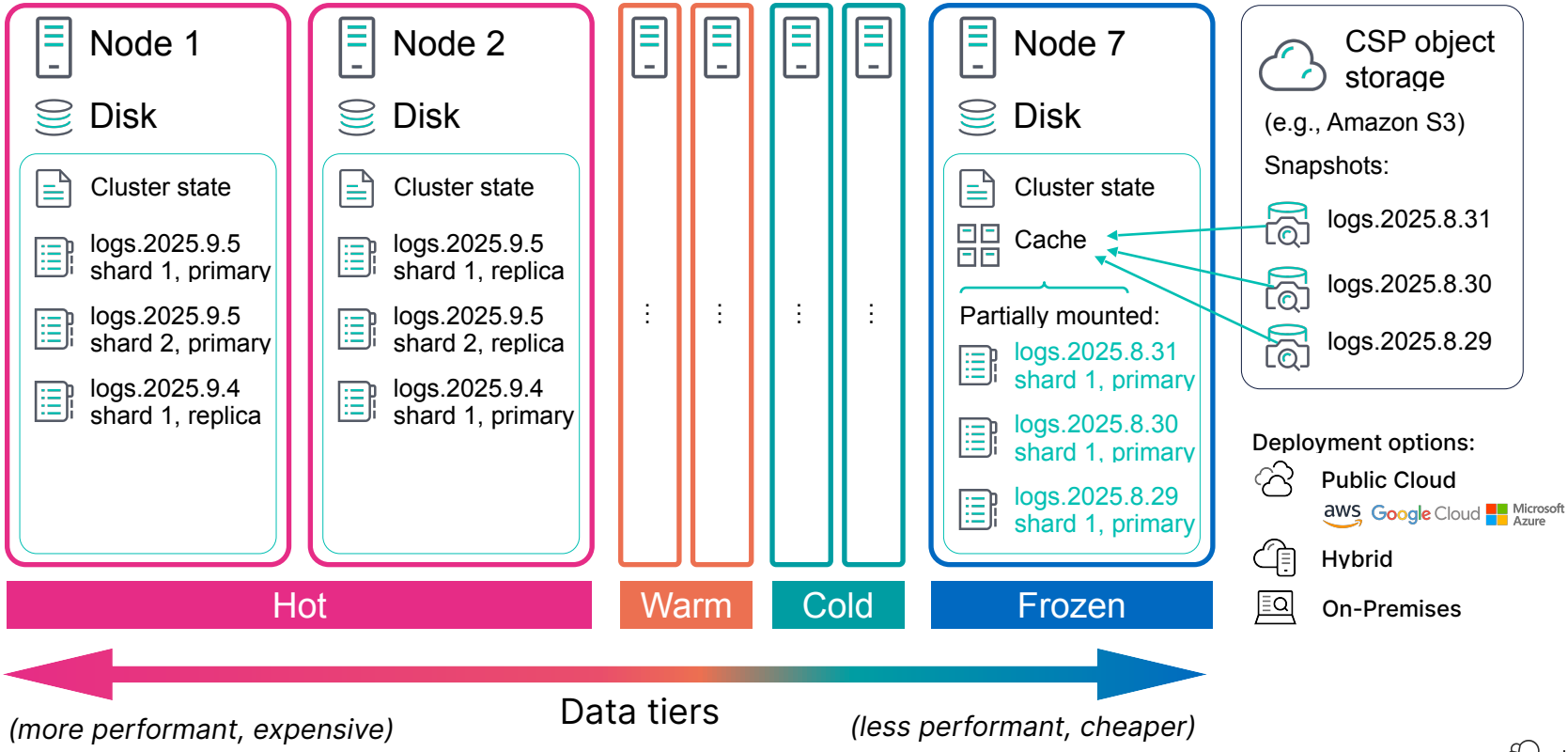
db-engines.com/en/ranking/search+engine

Rank Oct 2025	Rank Sep 2025	Rank Oct 2024	DBMS	Database Model	Score Oct 2025
1.	1.	1.	Elasticsearch	Multi-model	116.67
2.	2.	2.	Splunk	Search engine	73.87
3.	3.	3.	Apache Solr	Search engine, Multi-model	35.31
4.	4.	4.	OpenSearch	Multi-model	19.30
5.	5.	5.	Algolia	Search engine	6.80
6.	↑ 7.	↑ 7.	Microsoft Azure AI Search	Search engine, Multi-model	6.00
7.	↓ 6.	↓ 6.	Sphinx	Search engine	5.94

Elasticsearch

- Cluster
- Nodes
- Index
- Shards
- Segments
- Cluster state
- Data streams
- Index Lifecycle Management (ILM)

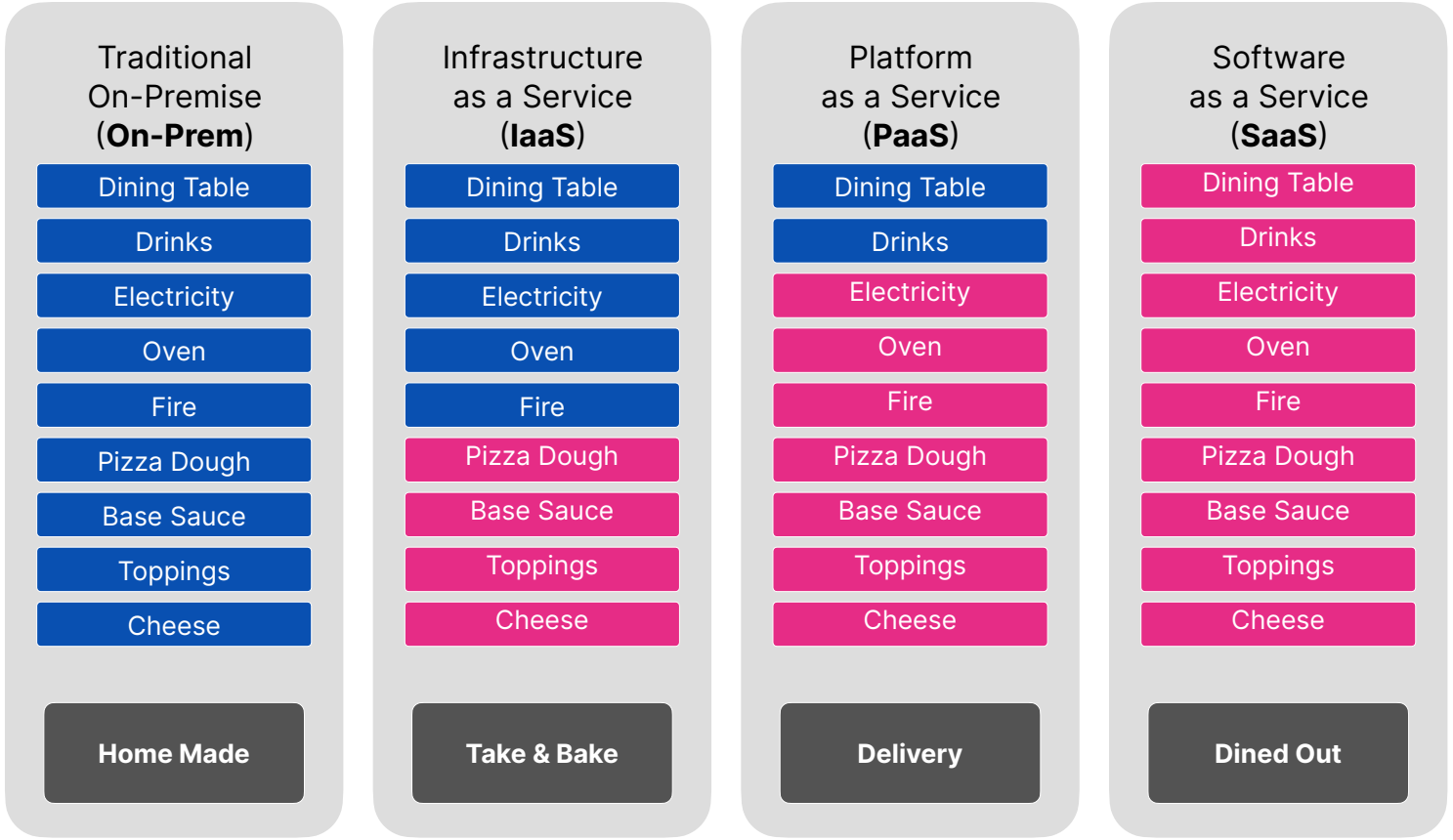
Shared-nothing stateful architecture



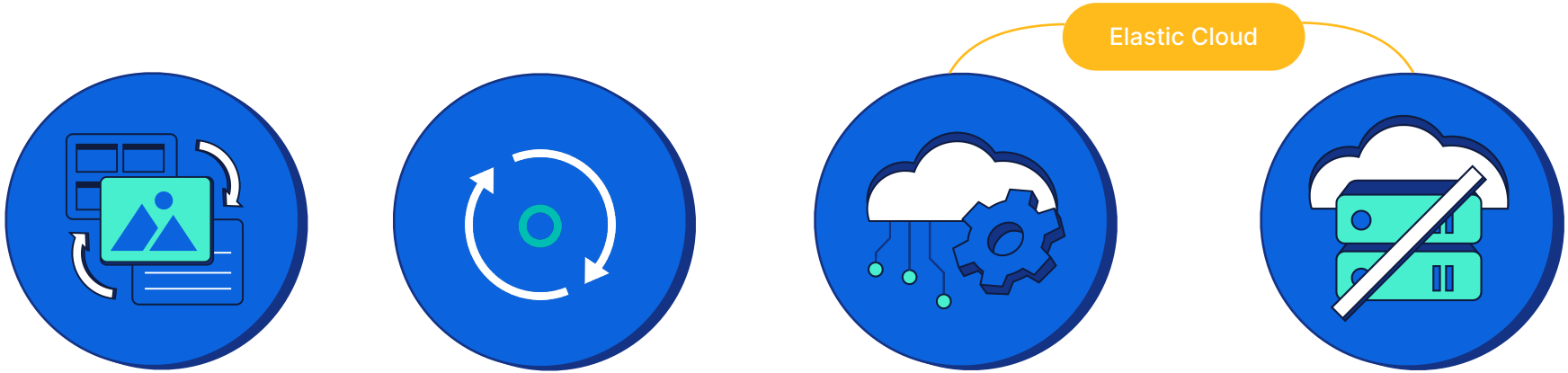
Disadvantages of stateful

- **User** defines cluster (RAM, CPU, disk)
- CPU & RAM **coupled** with storage
- CPU **shared** by ingestion & search
- Needs to think of **data tiers**
- **Adding/removing nodes** moves data

Pizza as a Service



Introducing **Elastic Cloud Serverless**



Elastic Self Managed

Download and
run it yourself

Elastic Cloud on K8s

We give the
orchestration to you

Elastic Cloud Hosted

We orchestrate
the infrastructure

Elastic Cloud Serverless

We manage
everything

Control

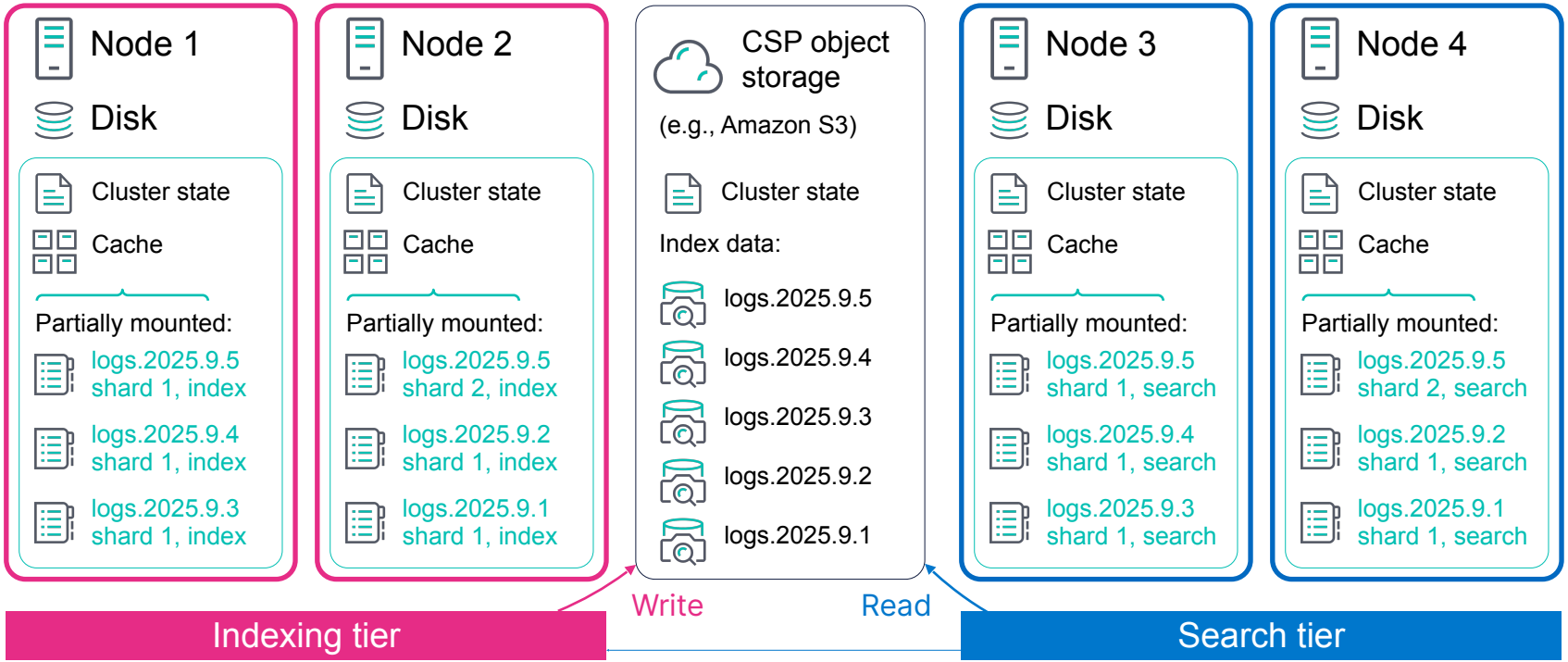
Simplicity

elastic

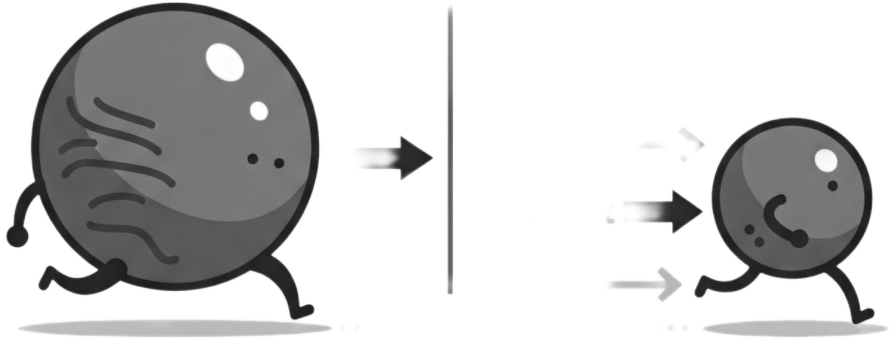
Serverless Elasticsearch

- SaaS
- Exploits **Cloud Service Provider**
- Stateless: **decouples compute from storage**
- **Same API** as stateful Elasticsearch
- **Pay-as-you-go** model (retention, indexing, searching)
- **No more cluster specifics**, sizing, upgrading, HA
- 2 simplified data tiers: **indexing**, **search**

New stateless architecture



(Super) Thin shards

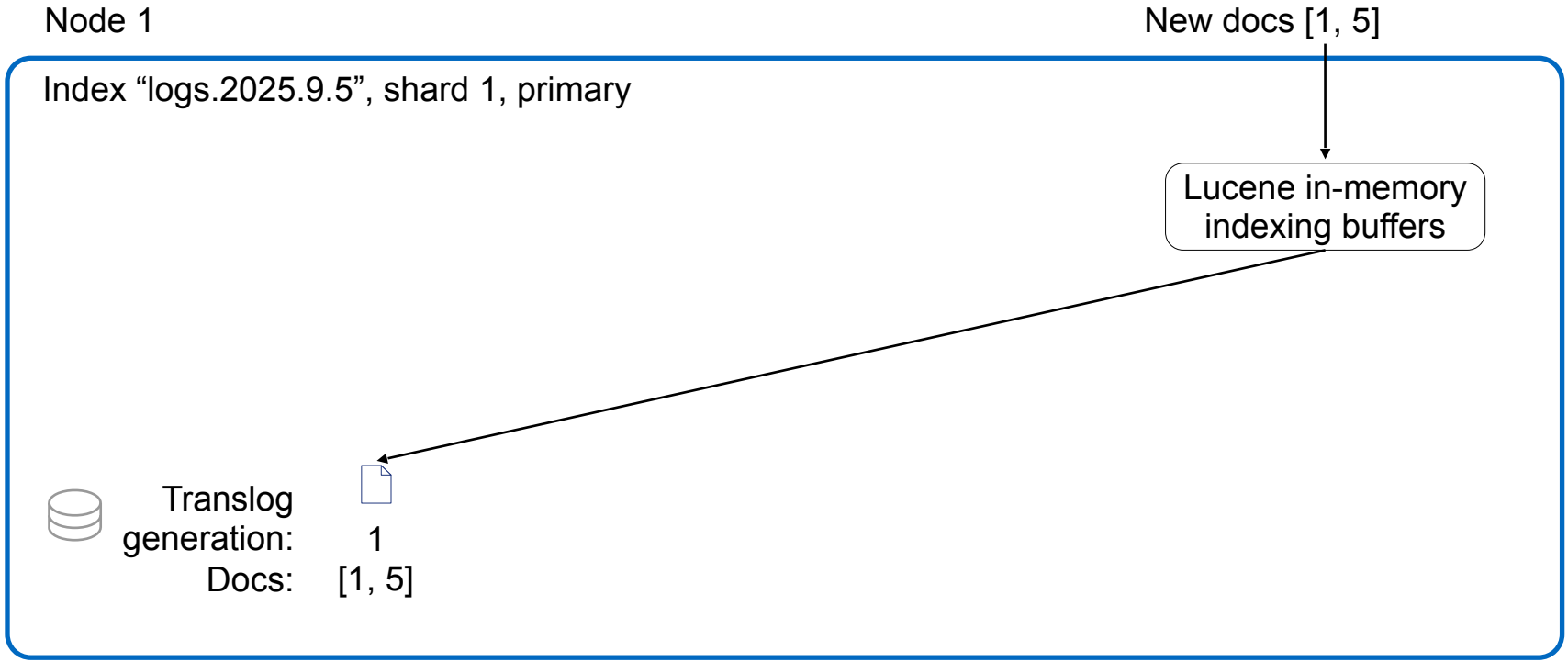


- Loaded **with metadata** until data is indexed/searched
- Indexing generates files, uploads them and deletes them when unused
- Searching loads from object store only the required data into the cache
- Indexing & searching virtually **limitless data**
- **Not dependent on disk**, not restricted by disk capacity
- Shard relocation (and recovery) is very fast
- **WIP**: **Super Thin Indexing Shards**

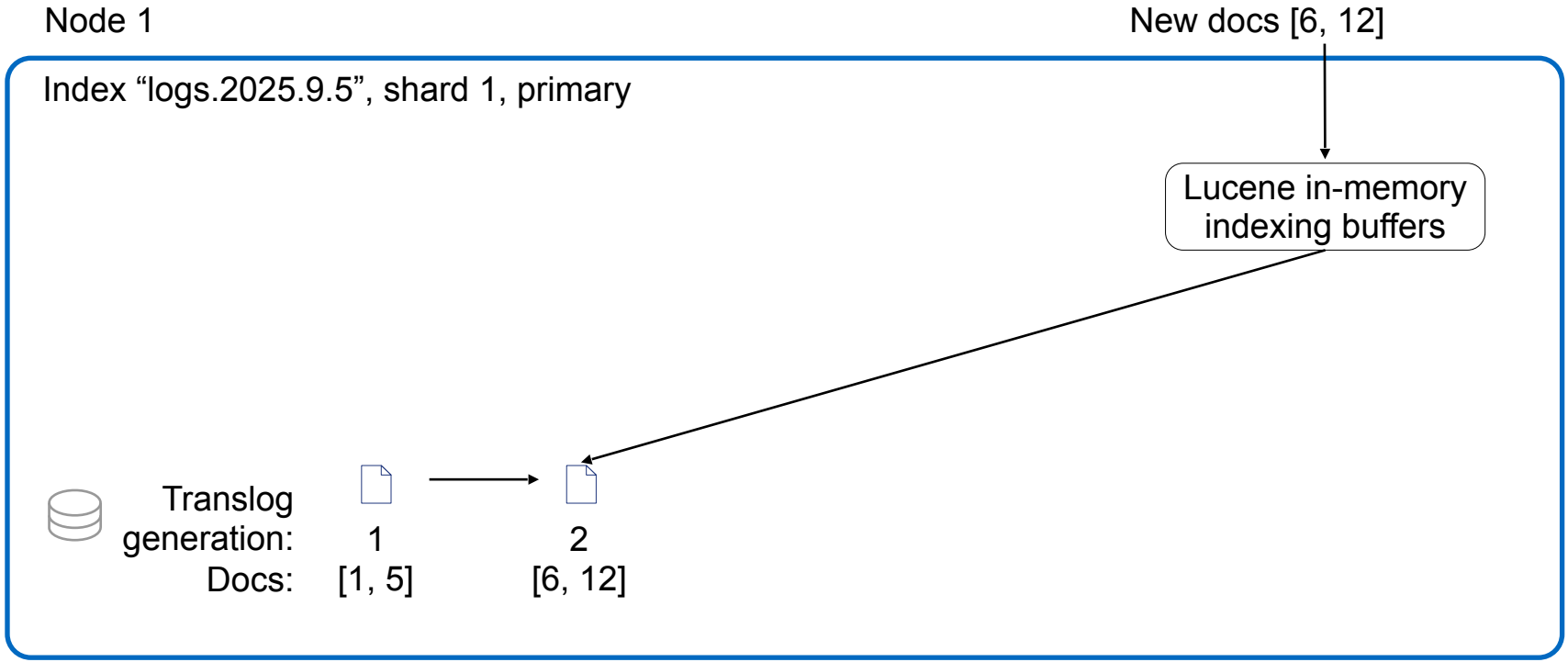
Stateful example

data inside a shard

Data inside a shard (stateful)



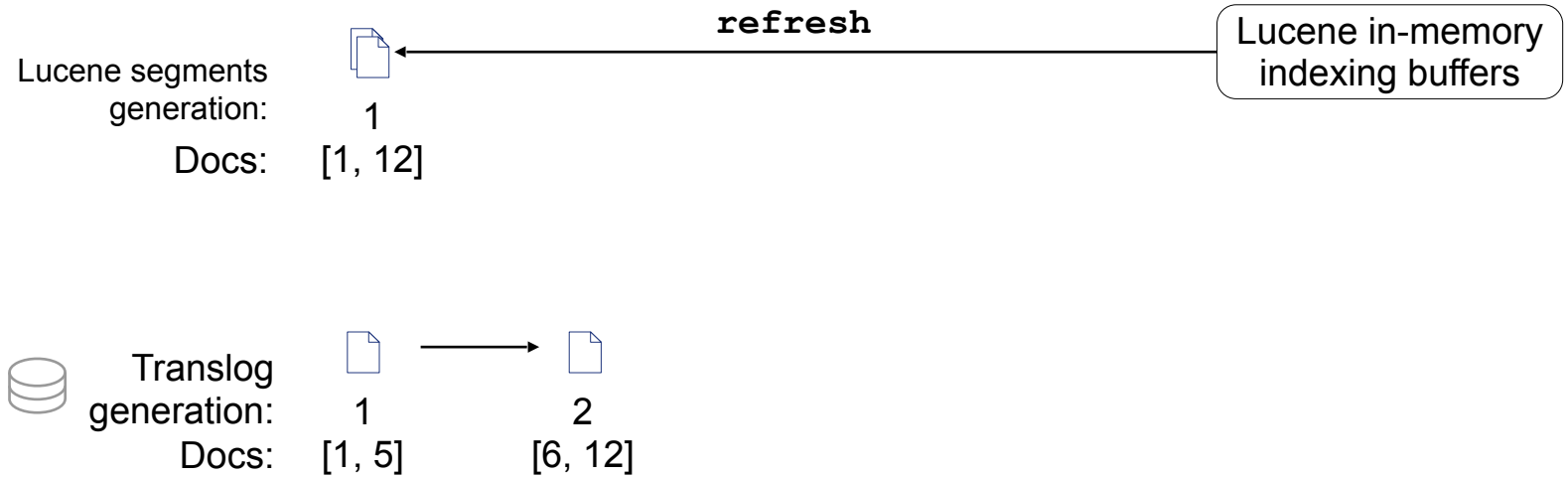
Data inside a shard (stateful)



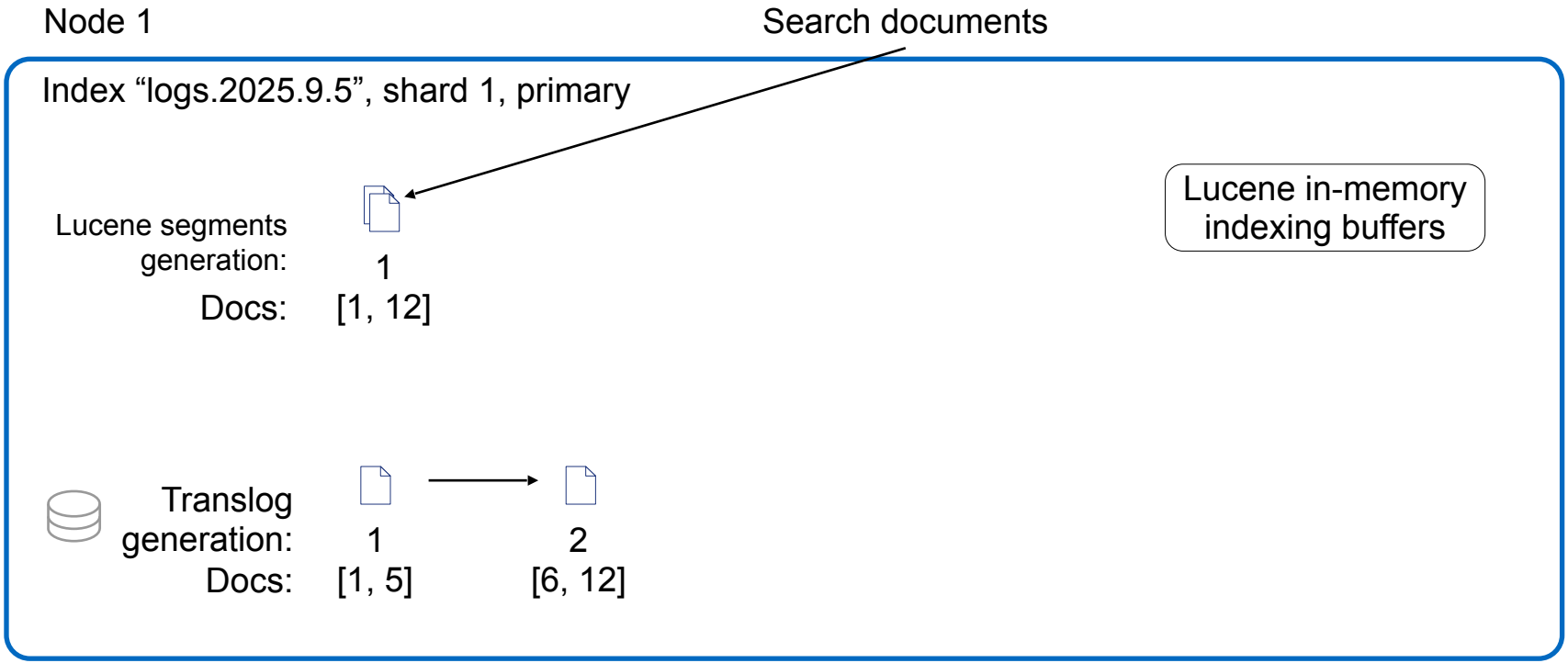
Data inside a shard (stateful)

Node 1

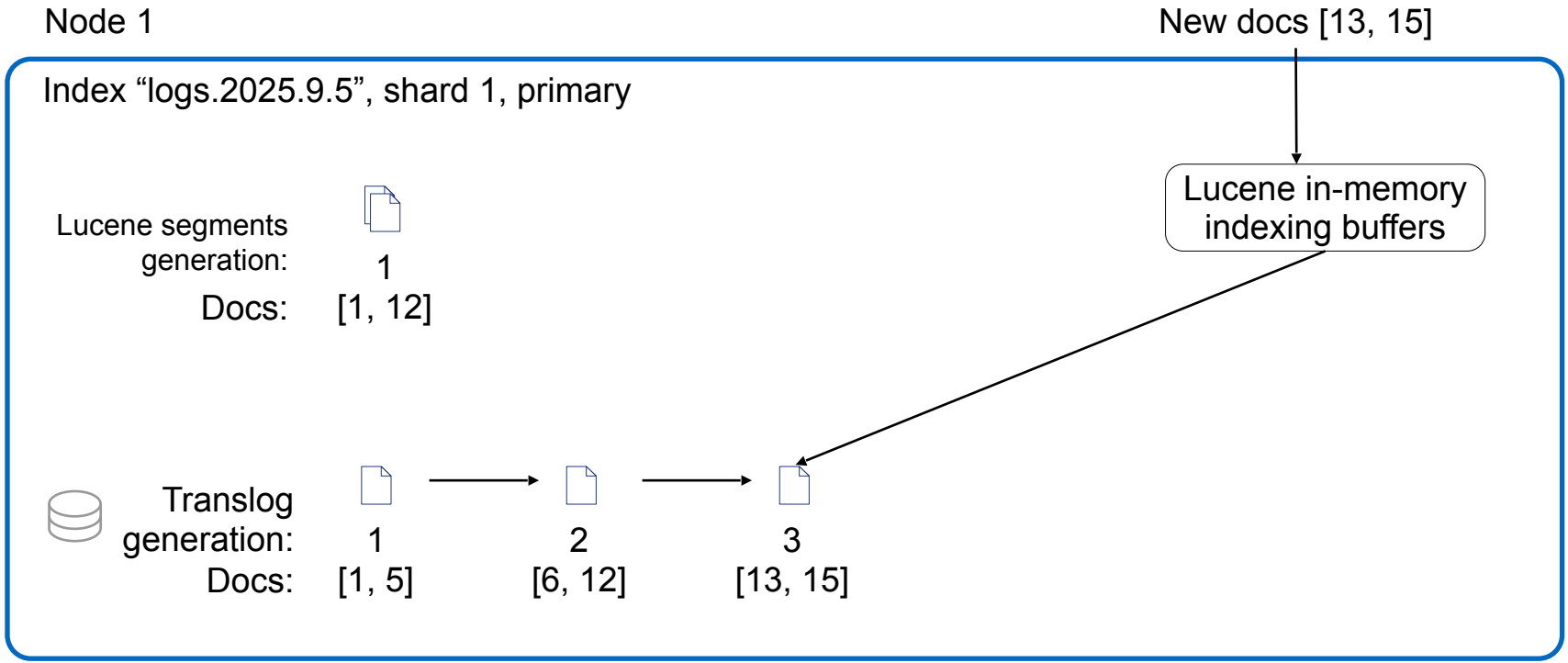
Index “logs.2025.9.5”, shard 1, primary



Data inside a shard (stateful)



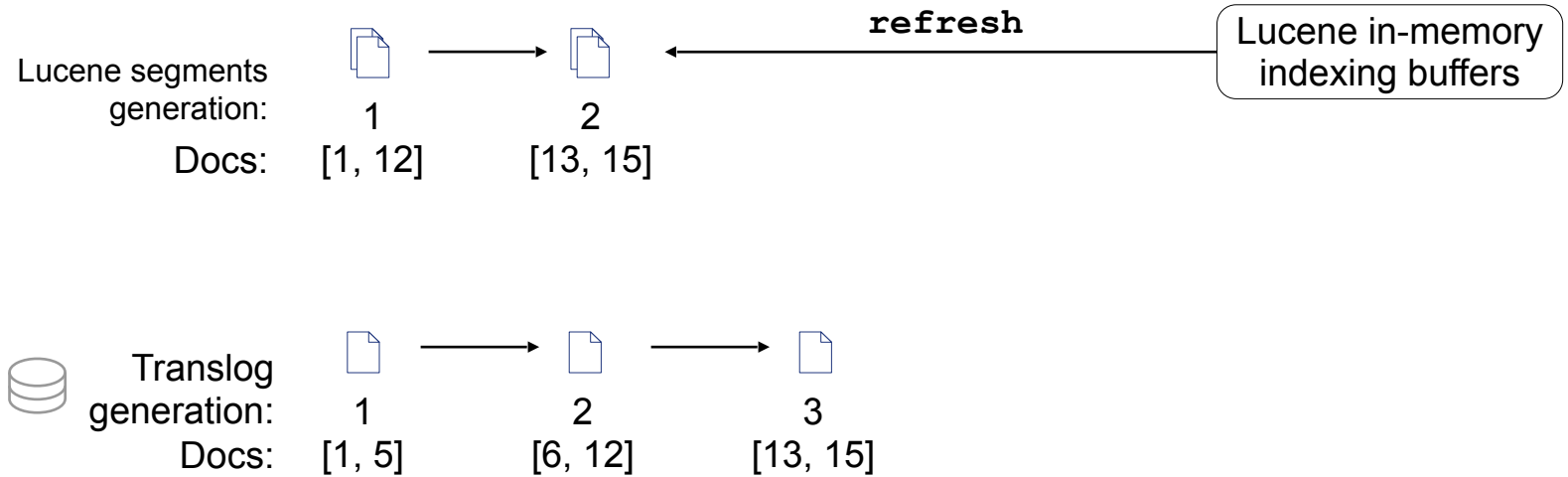
Data inside a shard (stateful)



Data inside a shard (stateful)

Node 1

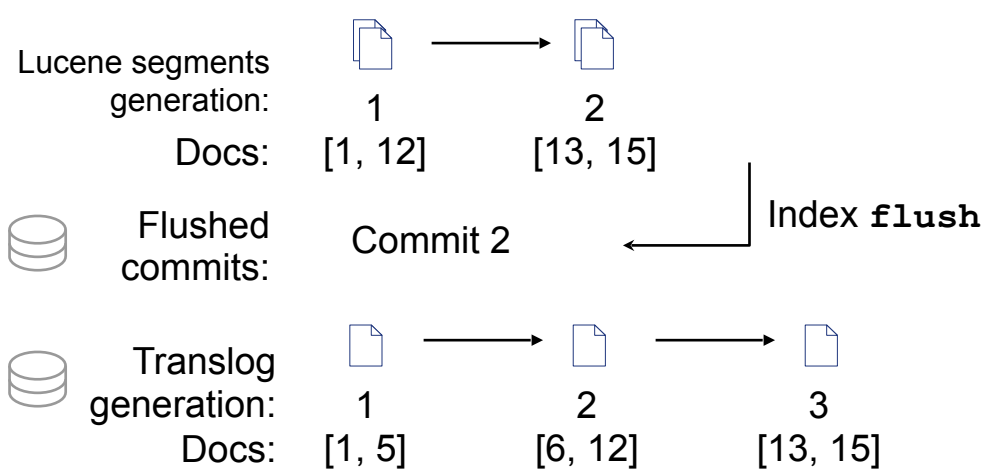
Index “logs.2025.9.5”, shard 1, primary



Data inside a shard (stateful)

Node 1

Index “logs.2025.9.5”, shard 1, primary

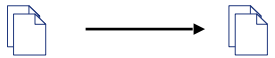


Lucene in-memory indexing buffers

Data inside a shard (stateful)

Node 1

Index “logs.2025.9.5”, shard 1, primary



Lucene segments generation: 1 2

Docs: [1, 12] [13, 15]

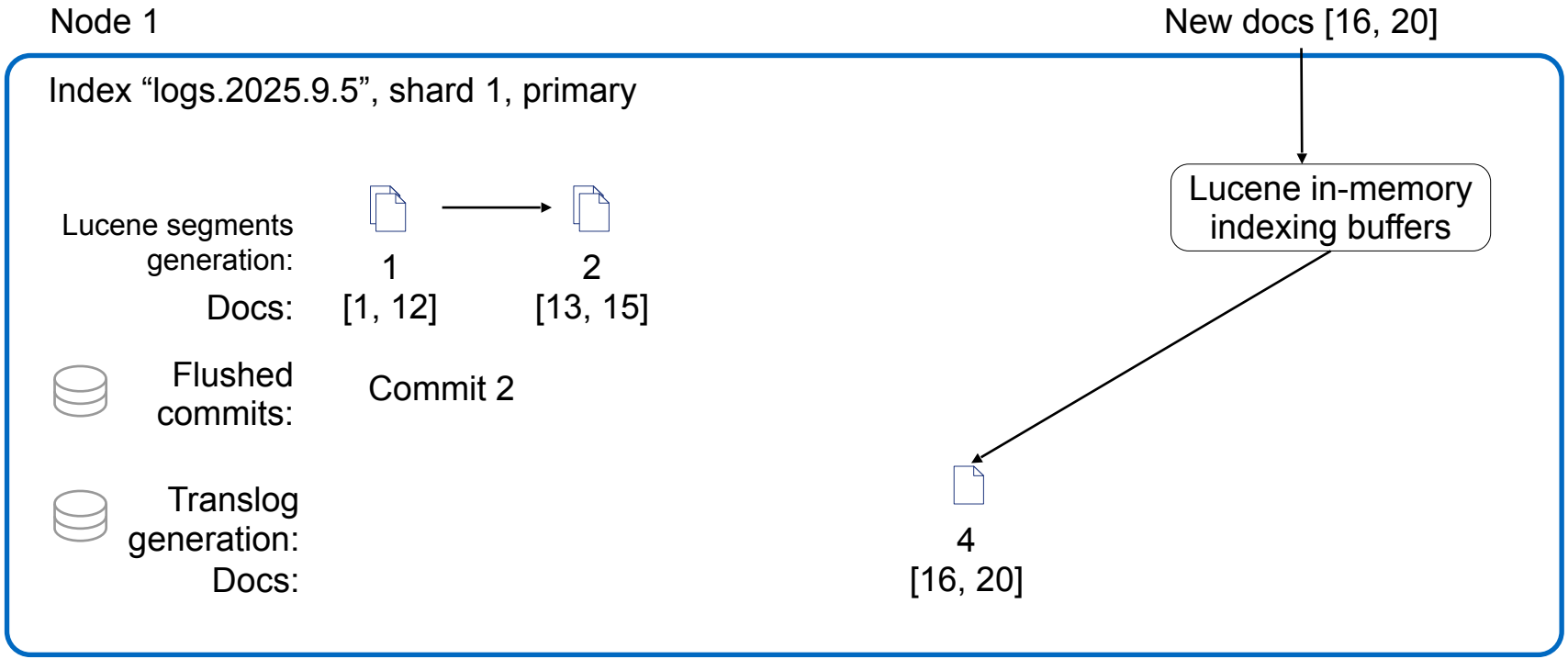
Lucene in-memory indexing buffers

Flushed commits: Commit 2

Translog generation:

Docs:

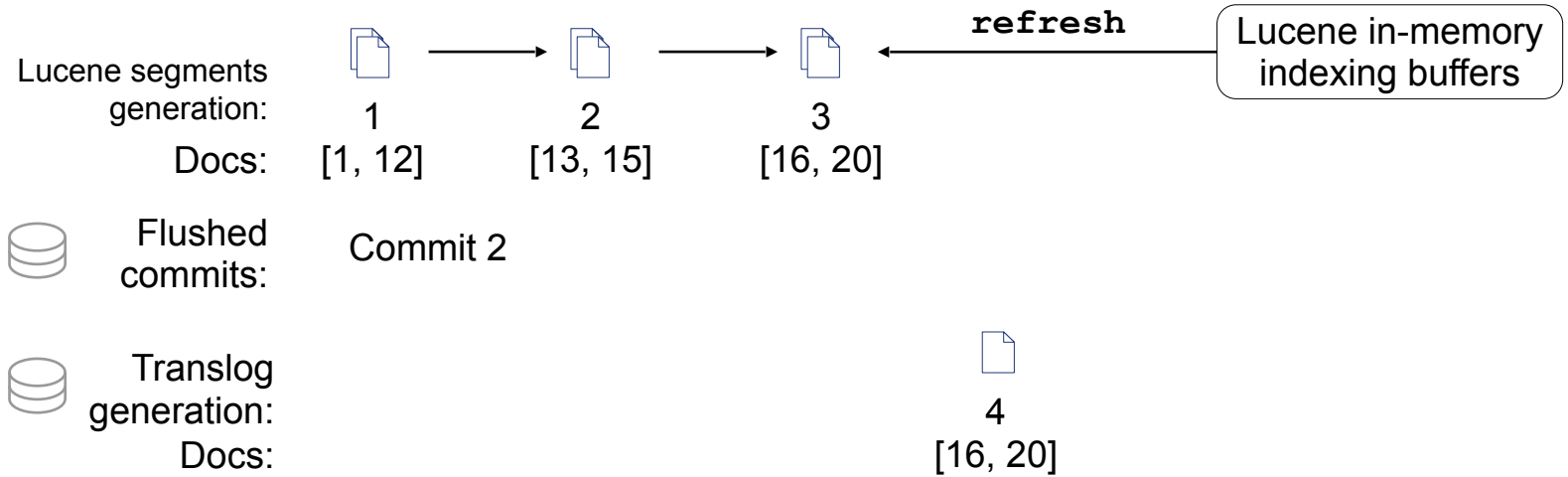
Data inside a shard (stateful)



Data inside a shard (stateful)

Node 1

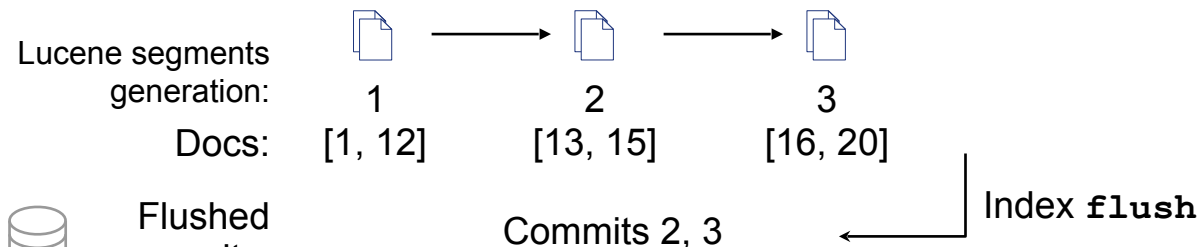
Index “logs.2025.9.5”, shard 1, primary



Data inside a shard (stateful)

Node 1

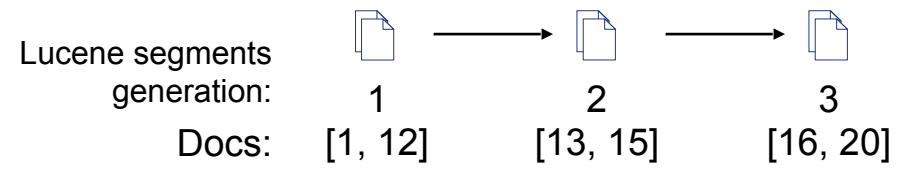
Index “logs.2025.9.5”, shard 1, primary



Data inside a shard (stateful)

Node 1

Index “logs.2025.9.5”, shard 1, primary



Searchable **refreshed** segments

Flushed commits:

Commits 2, 3

Flushed commits (on disk)

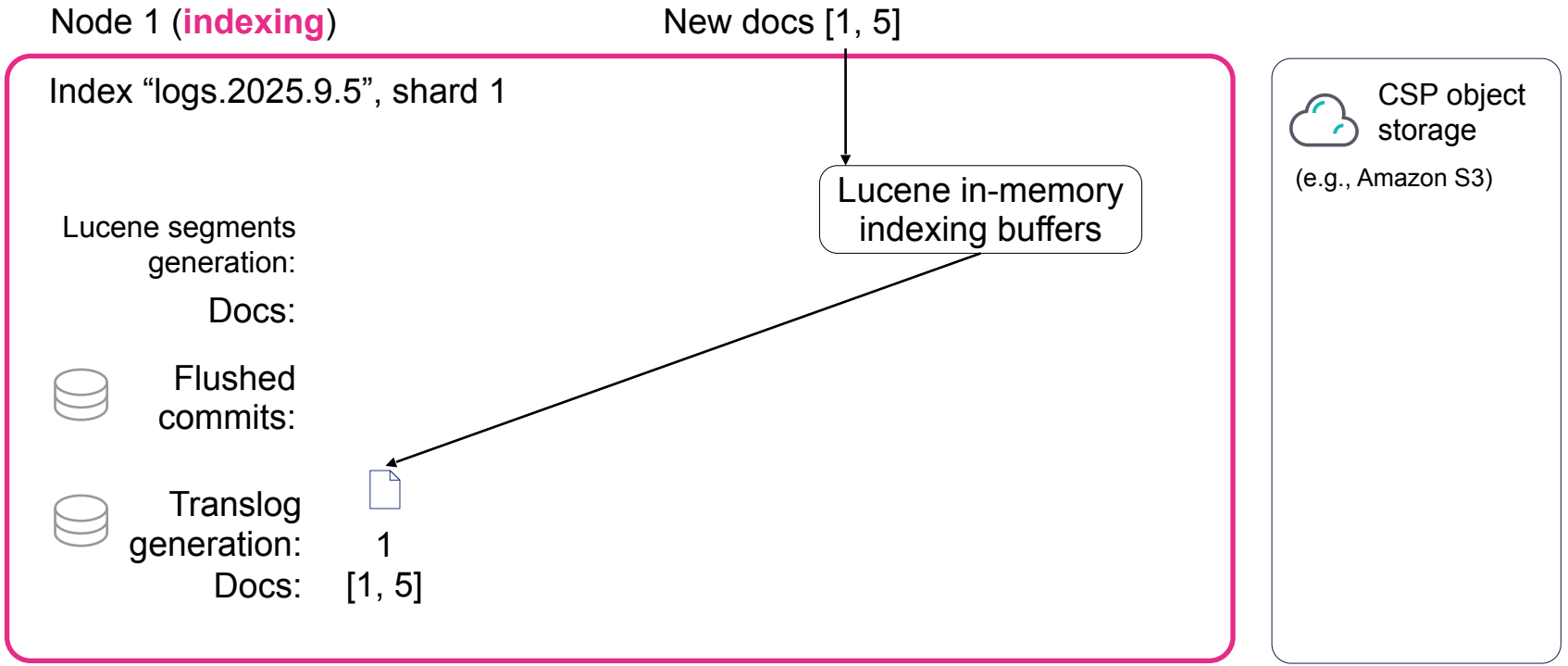
Translog generation:
Docs:

Non-flushed operations (on disk) for recovery

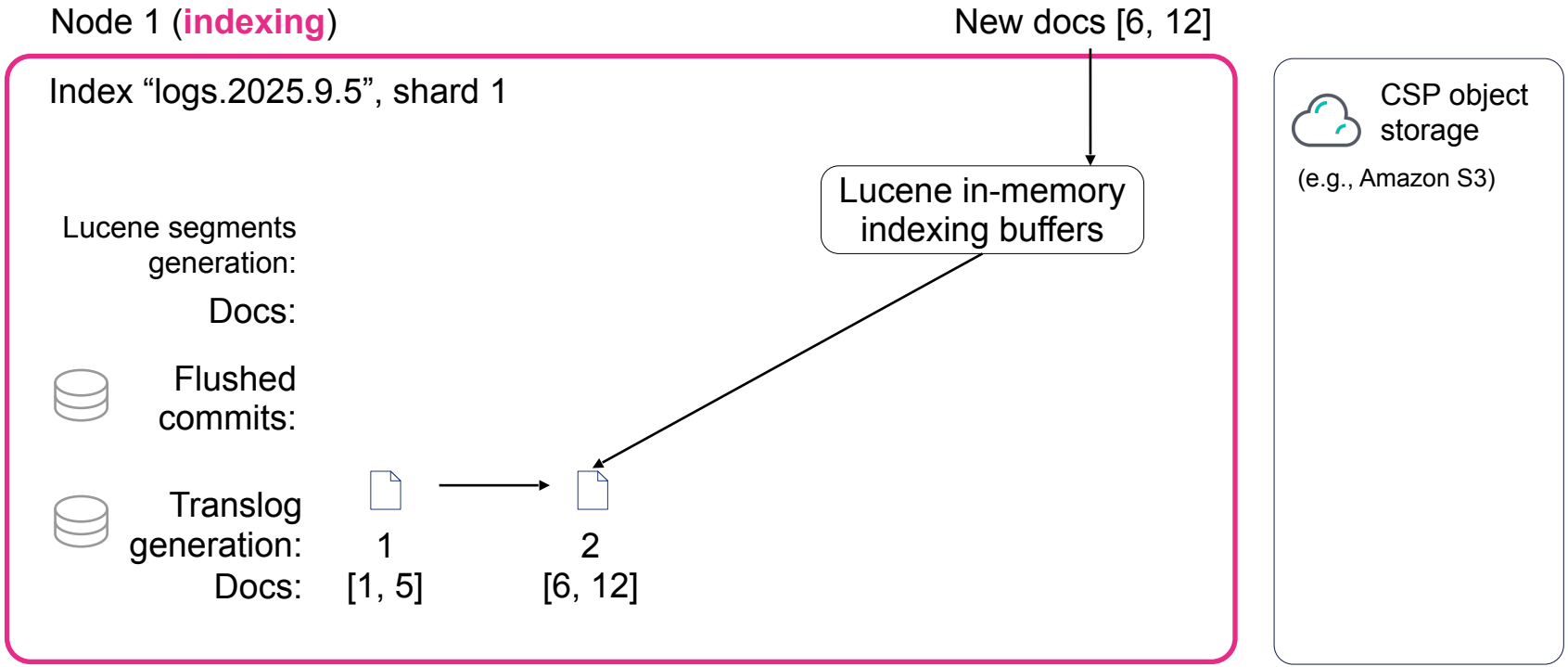
Stateless example

offloading shard data to an object store

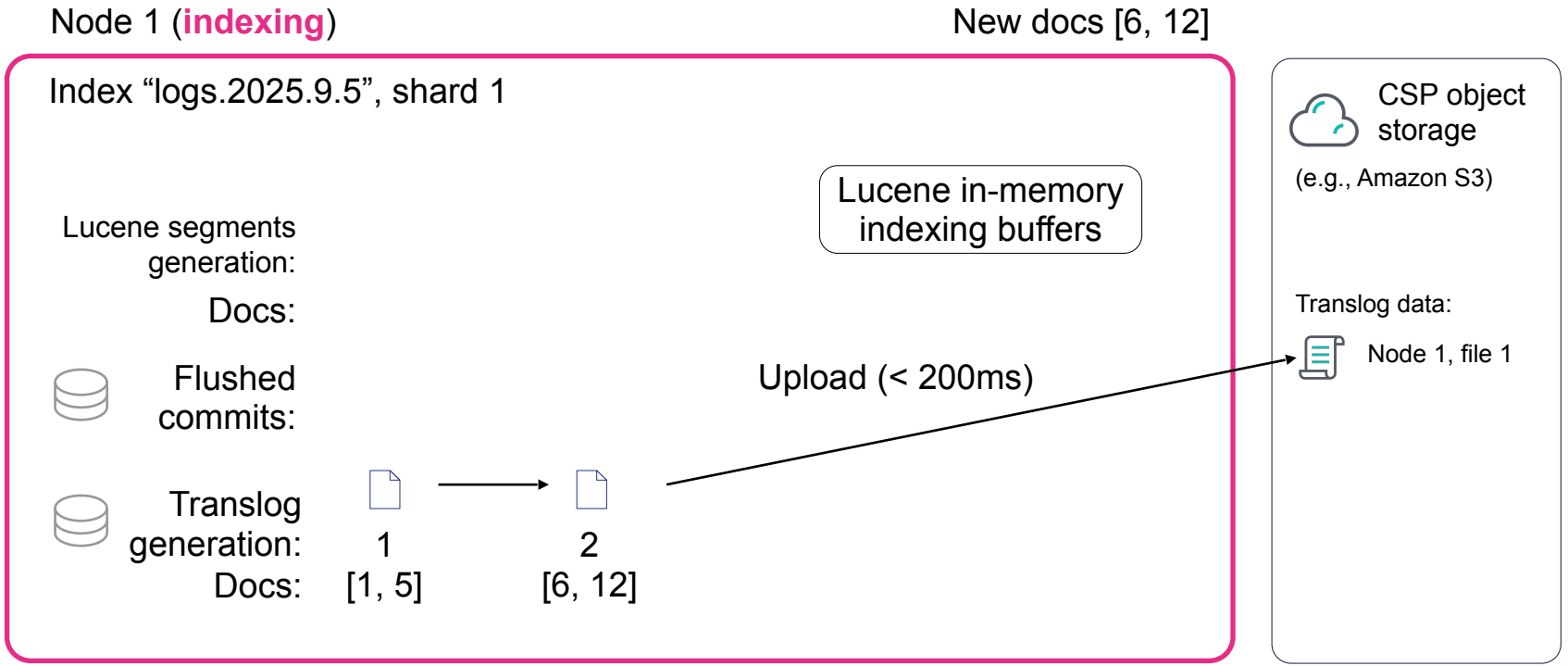
Offloading index data to the object store (stateless)



Offloading index data to the object store (stateless)



Offloading index data to the object store (stateless)



Offloading index data to the object store (stateless)

Node 1 (**indexing**)

Acknowledged writes

Index “logs.2025.9.5”, shard 1

Lucene in-memory indexing buffers

Lucene segments generation:

Docs:

Flushed commits:

Translog generation:

Docs:

1 [1, 5]

2 [6, 12]

CSP object storage

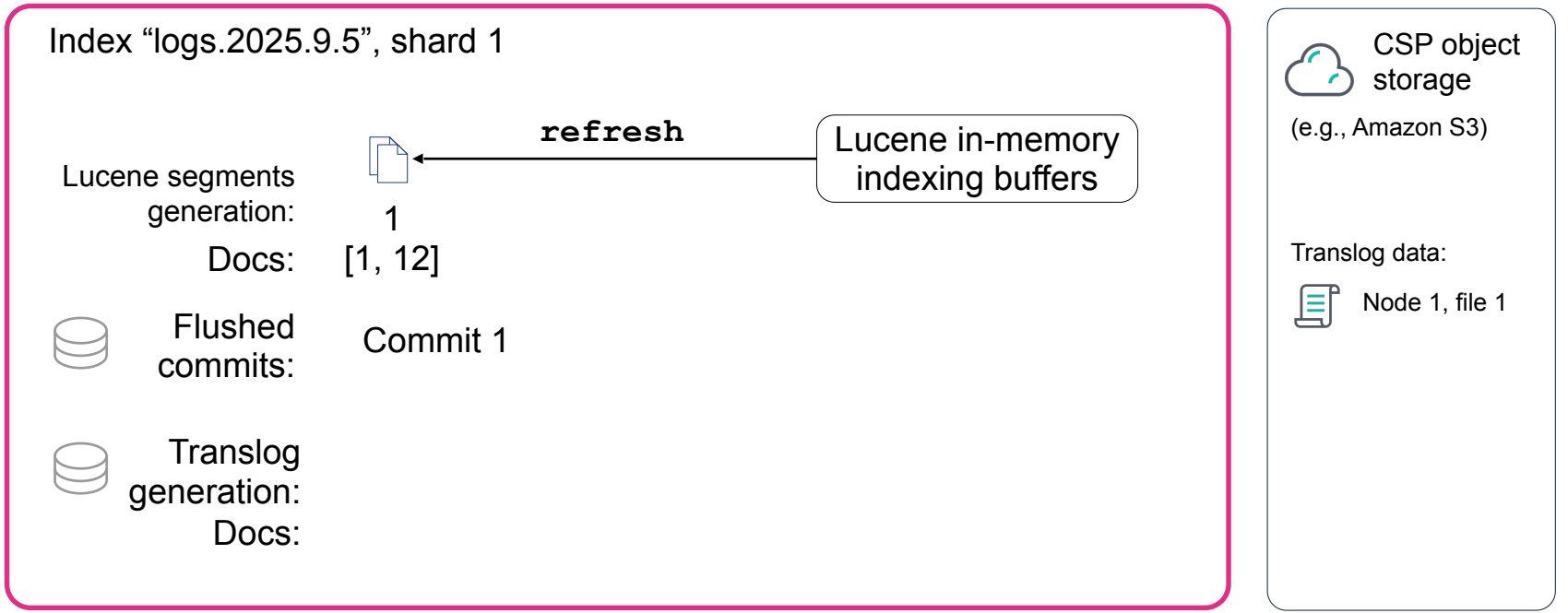
(e.g., Amazon S3)

Translog data:

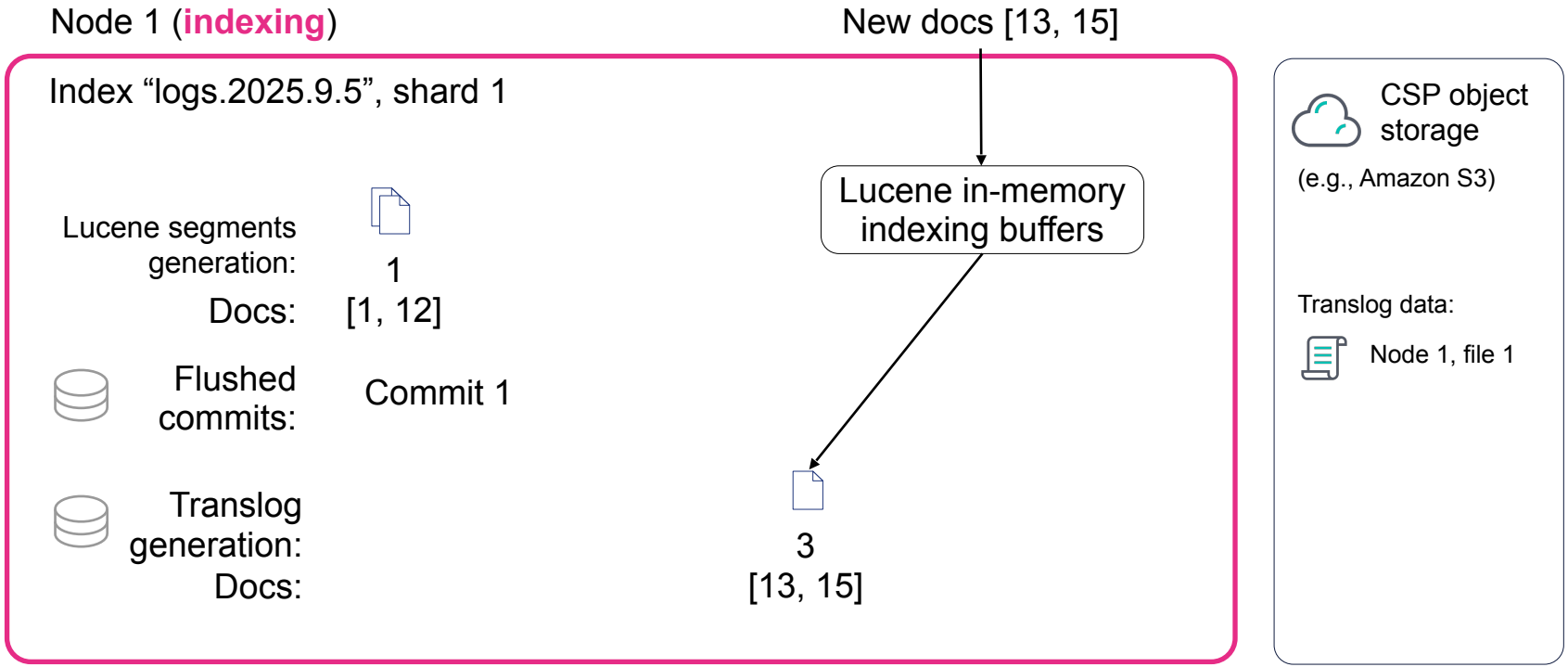
Node 1, file 1

Offloading index data to the object store (stateless)

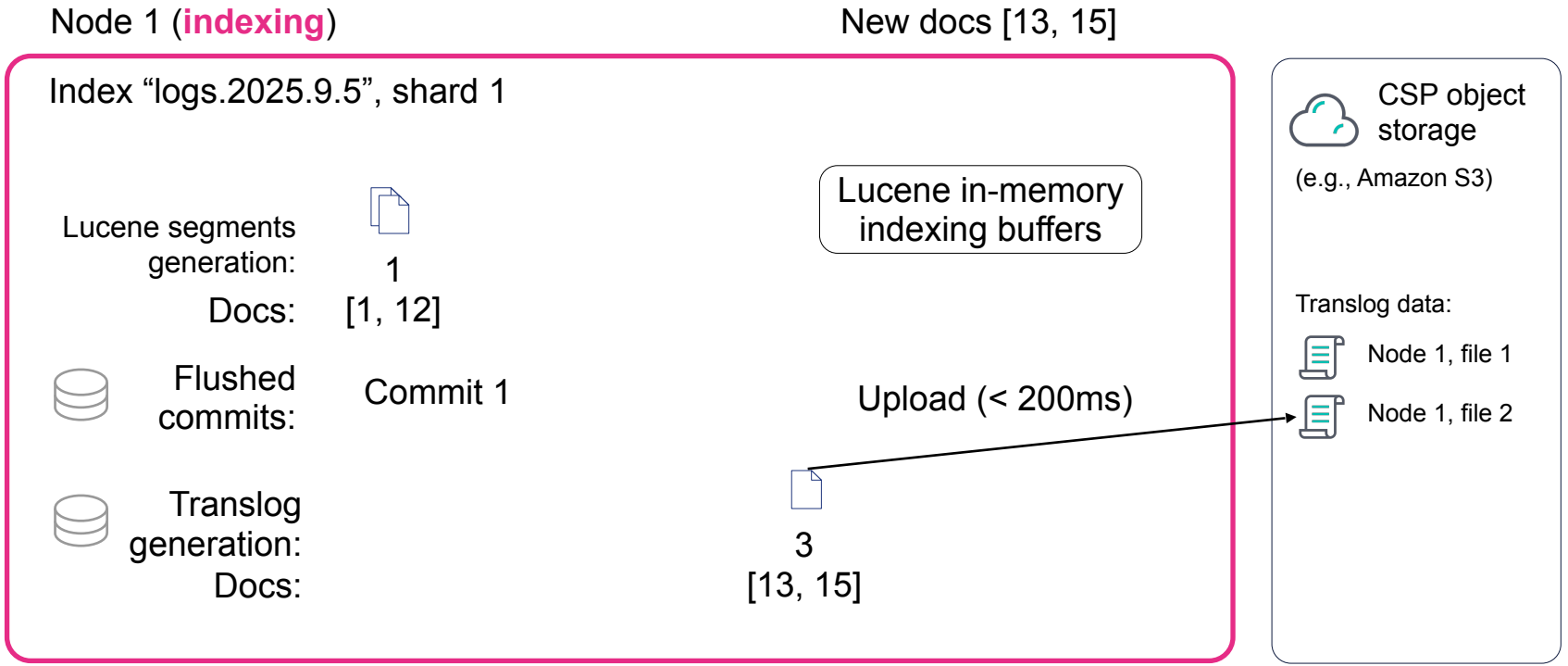
Node 1 (**indexing**)



Offloading index data to the object store (stateless)



Offloading index data to the object store (stateless)



Offloading index data to the object store (stateless)

Node 1 (**indexing**)

Acknowledged writes

Index “logs.2025.9.5”, shard 1

Lucene segments generation: 1

Docs: [1, 12]

Lucene in-memory indexing buffers

Flushed commits: Commit 1

Translog generation: 3

Docs: [13, 15]

CSP object storage

(e.g., Amazon S3)

Translog data:

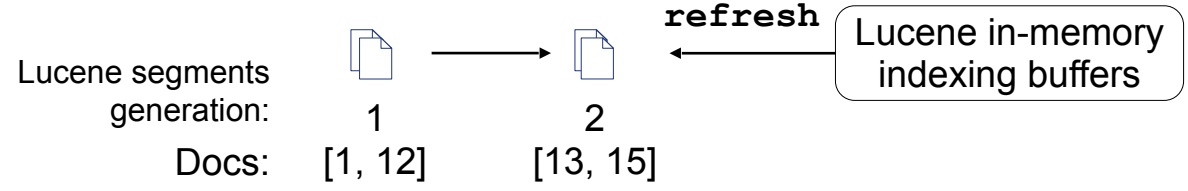
Node 1, file 1

Node 1, file 2

Offloading index data to the object store (stateless)

Node 1 (**indexing**)

Index “logs.2025.9.5”, shard 1



CSP object storage

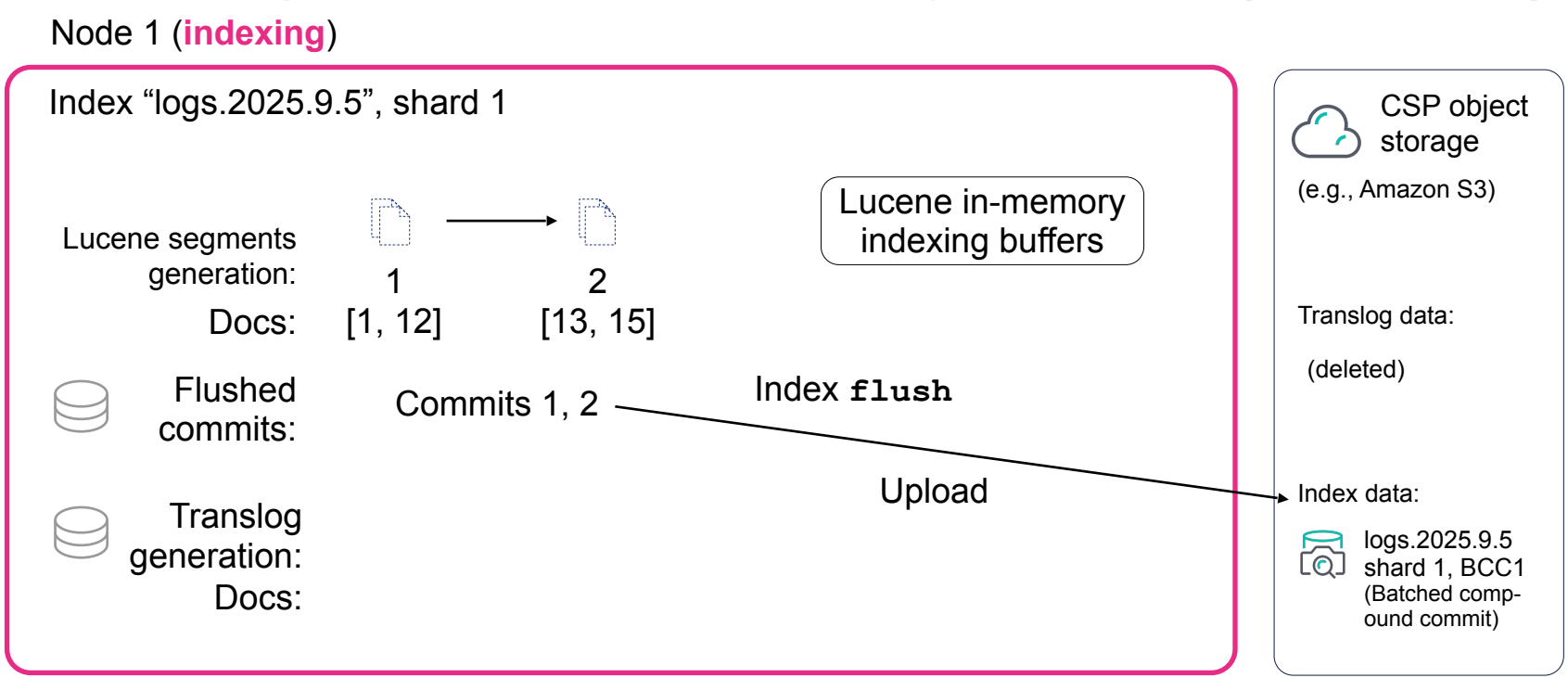
(e.g., Amazon S3)

Translog data:

Node 1, file 1

Node 1, file 2

Offloading index data to the object store (stateless)



Offloading index data to the object store (stateless)

Node 2 (**search**)

Index “logs.2025.9.5”, shard 1

Lucene segments generation:

Docs:

1 [1, 12]

2 [13, 15]

Flushed commits:

Commits 1, 2

Translog generation:

Docs:

Lucene in-memory indexing buffers

Node Cache

Partially mounted:

CSP object storage

(e.g., Amazon S3)

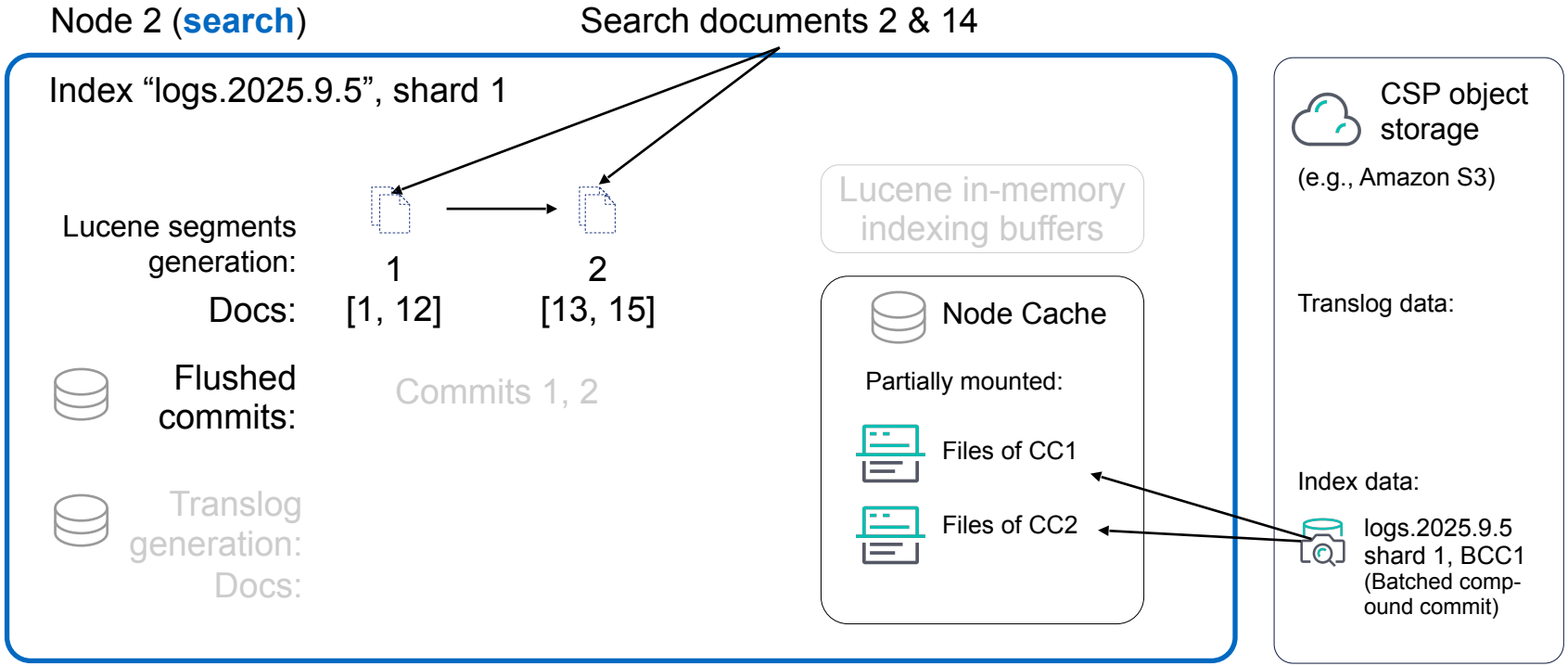
Translog data:

Index data:

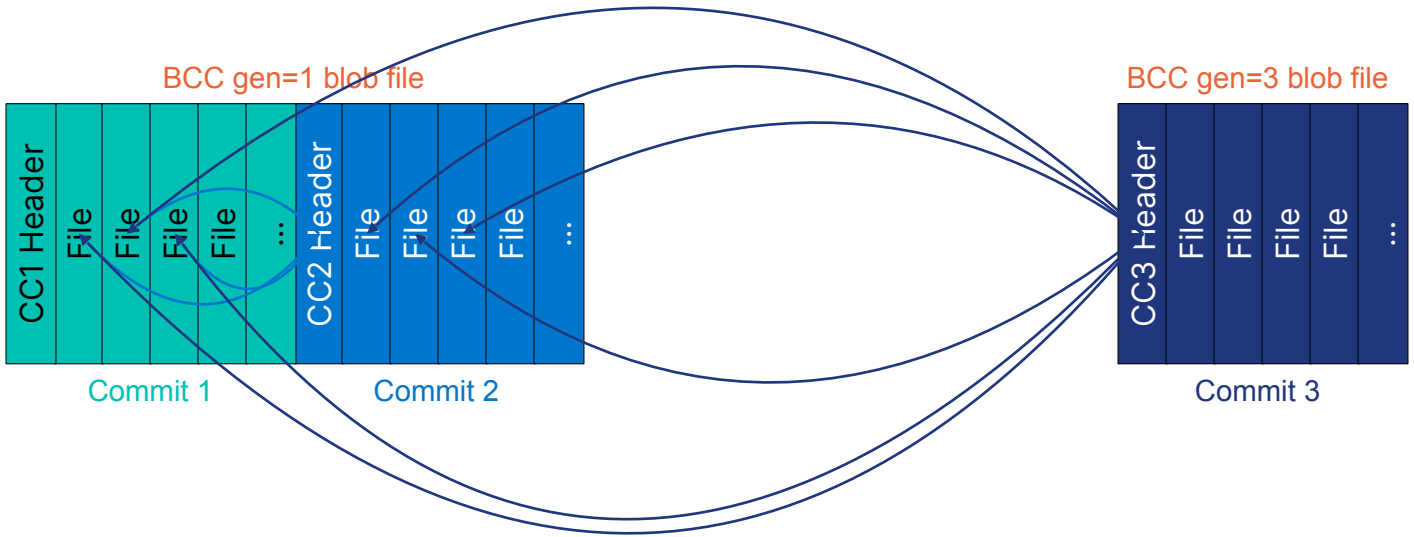
logs.2025.9.5 shard 1, BCC1 (Batched comp-ound commit)

elastic

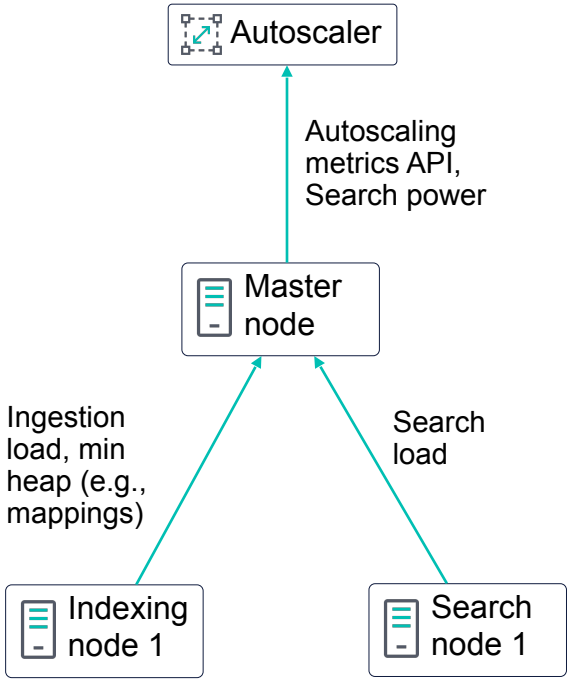
Offloading index data to the object store (stateless)



Batched Compound Commit



Autoscaling

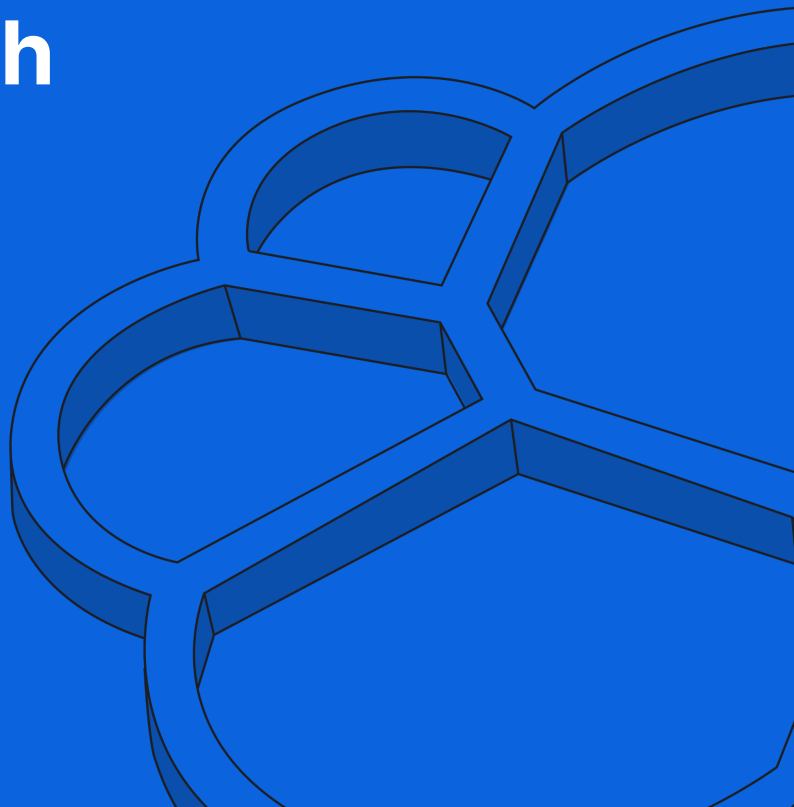


Limitations

- Links to external resources in Kibana is not supported
- Refresh rate non adjustable to less than x per minute
- No dashboard export as PNG/PDF (no headless browser)
- Subset of APIs only
- Can not define the number of shards

Do MORE with StateLESS Elasticsearch

David Pilato



elastic | The Search AI Company