Elasticsearch Serverless:
the Transition from Stateful to Stateless

Iraklis Psaroudakis, Principal Software Engineer
April 12th, Devoxx Greece 2025

Iraklis Psaroudakis

Principal Software Engineer
at Elastic, focusing on distributed

Previously:
Oracle Labs, graph-based analytics
PhD at EPFL, scaling up analytics
ECE degree at NTUA in Athens

Agenda:

- What is Elastic
- What is Elasticsearch
- Shared-nothing stateful architecture
- Serverless Elasticsearch
- New stateless architecture
- Walkthrough of how data is stored in stateful vs serverless Elasticsearch
- Batching data
- Autoscaling

This talk contains personal views and is not officially endorsed

elastic

Meet Elastic — The Search AI Company

Elastic helps everyone find answers that matter.
From all data. In real time. At scale.

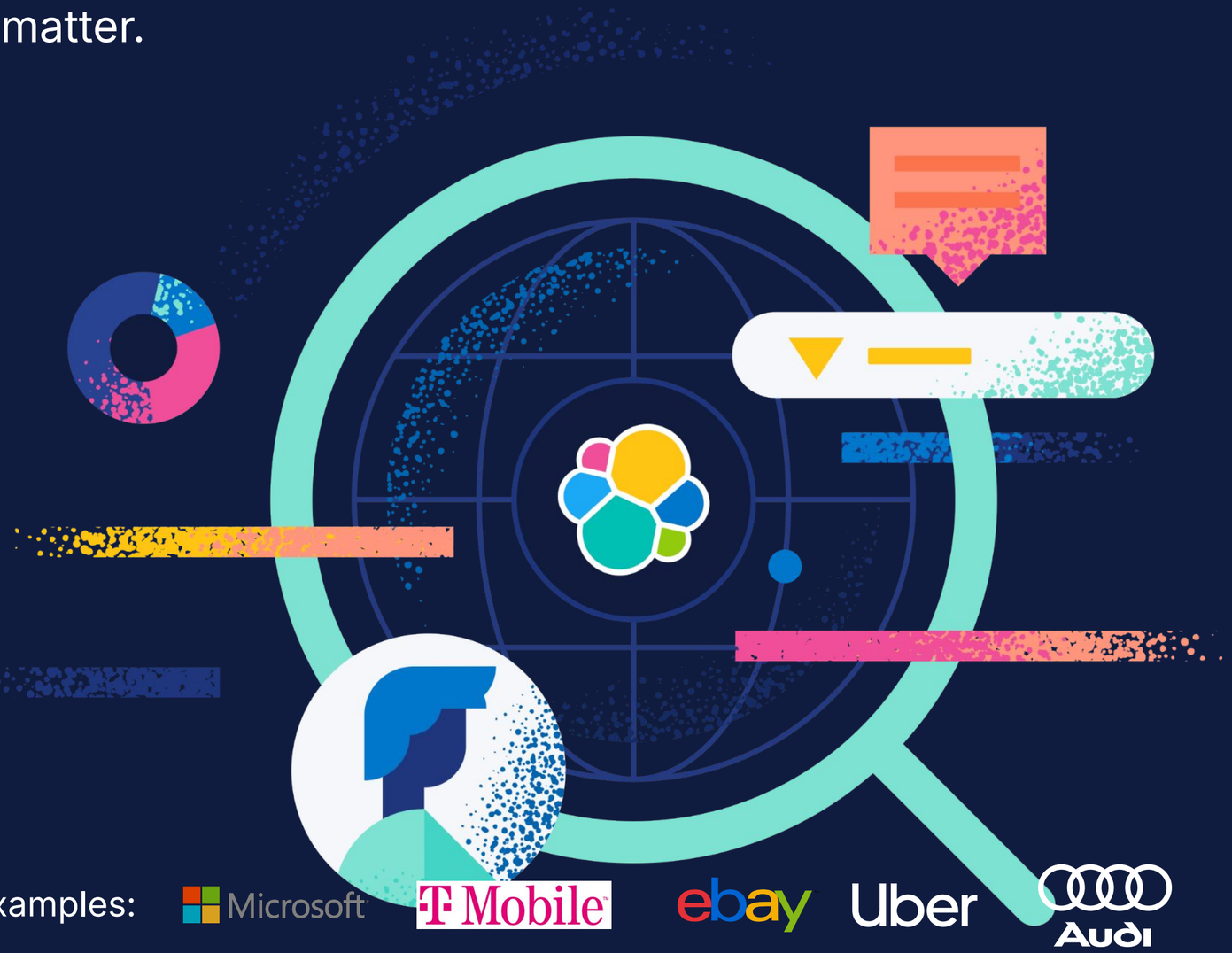
Founded in 2012
NYSE: ESTC

2800+
employees

40+
Countries with employees

4B+
downloads

54%
Of Fortune 500 companies
trust Elastic



Examples: Microsoft T-Mobile ebay Uber Audi

Data as of April 2023

Performance that Delivers Relevant Results in Real-time

Any Data, Any Source

- Databases
- Legacy Systems
- Applications
- SaaS Apps
- Web Services
- Files
- Public Cloud
- On-Premises

Out-of-the-Box Solutions

Elastic Observability

Logs, APM, Tracing, Metrics, Synthetics, Profiling, RUM

Elastic Security

SIEM, Endpoint, Cloud

Build Your Own

Elasticsearch

Generative AI Apps, Product Search, Workplace Search, Custom Search Apps

Search AI Platform

Ingest & Secure Storage

- Data Extraction
- Transformation / Normalization
- Enrichment
- Loading / Indexing
- Intelligent Data Storage
- Security / Governance

AI / ML & Search

- Full-Text / Vector Search
- Machine Learning
- Correlations
- Analytics & Aggregations
- Data Manipulation
- Federated Searches & Queries

Visualization & Automation

- Share & Collaborate
- Data Exploration
- Data Visualization
- Custom Dashboards
- 3rd Party Integrations
- Workflow Automation

Business Outcomes for Everyone

69% Improvement in customer and employee satisfaction

60% Reduction in risk

62% Reduction in revenue disruption

**Validated by third-party research*

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 Mar 2025	Rank Feb 2025	Rank Mar 2024	DBMS	Database Model
1.	1.	1.	Elasticsearch	Multi-model
2.	2.	2.	Splunk	Search engine
3.	3.	3.	Apache Solr	Search engine, Multi-model
4.	4.	4.	OpenSearch	Multi-model
5.	5.	↑ 6.	Sphinx	Search engine
6.	6.	↑ 7.	Algolia	Search engine
7.	7.	↑ 8.	Microsoft Azure AI Search	Search engine, Multi-model

elastic

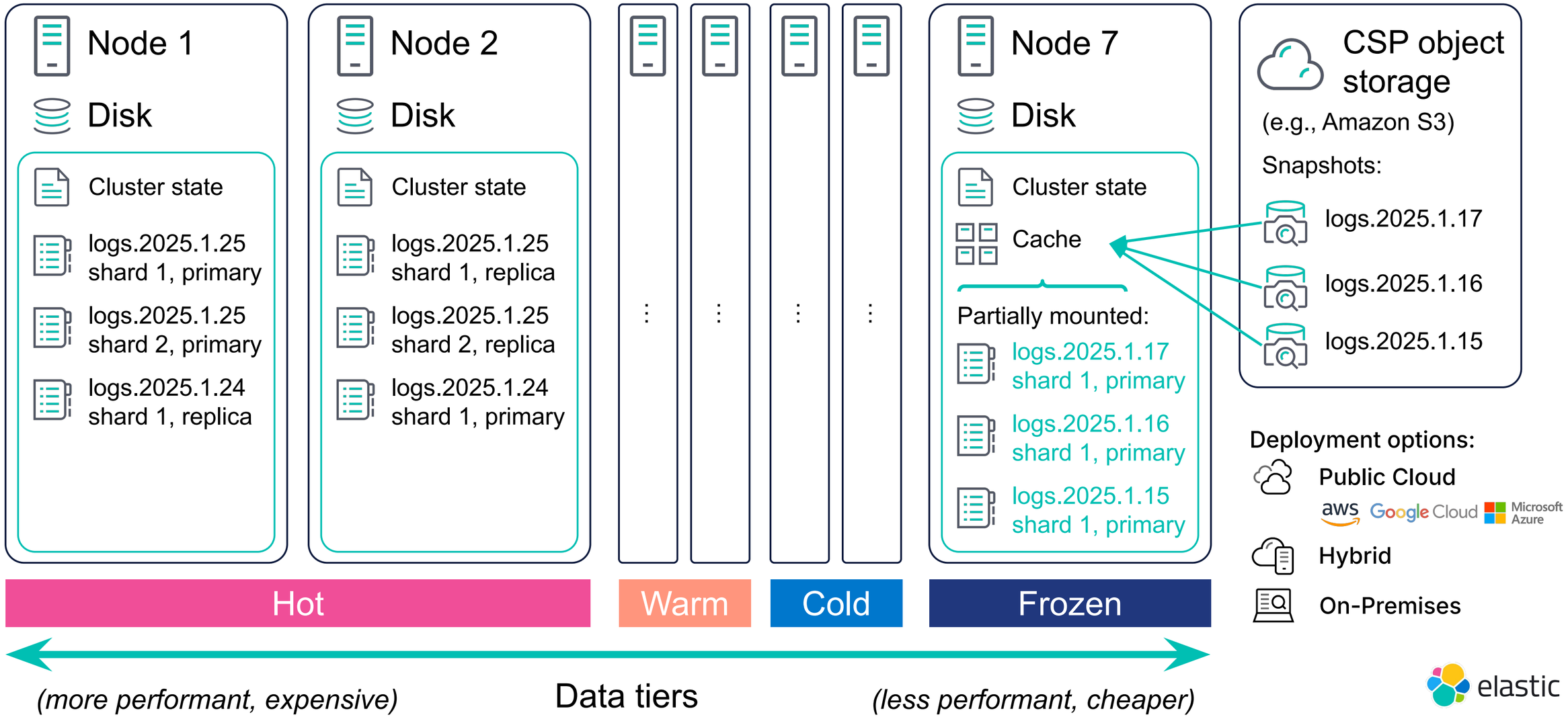
Elasticsearch in a nutshell

Devoxx 2023 talk: kingherc.com/archives/36

- An **index** comprises of Lucene segments (~ inverted indices)
- ES scales with index **shards** (primary & replica)
- Searches & aggregations across distributed shards
- Shards recover from disk & other nodes, and relocate
- **Cluster state** contains core metadata (e.g., shard assignments)
  - Master node updates it with a consensus protocol
- ES scales time-series data (e.g., logs) with data streams
  - Chain of older read-mostly indices and one write index
- Index Lifecycle Management (ILM) passes indices through data tiers
  - **Hot**, performant tier for recent updateable data
  - **Frozen**, cheaper, partial caching of static snapshots on an object store

elastic

Shared-nothing stateful architecture



elastic

Disadvantages of stateful

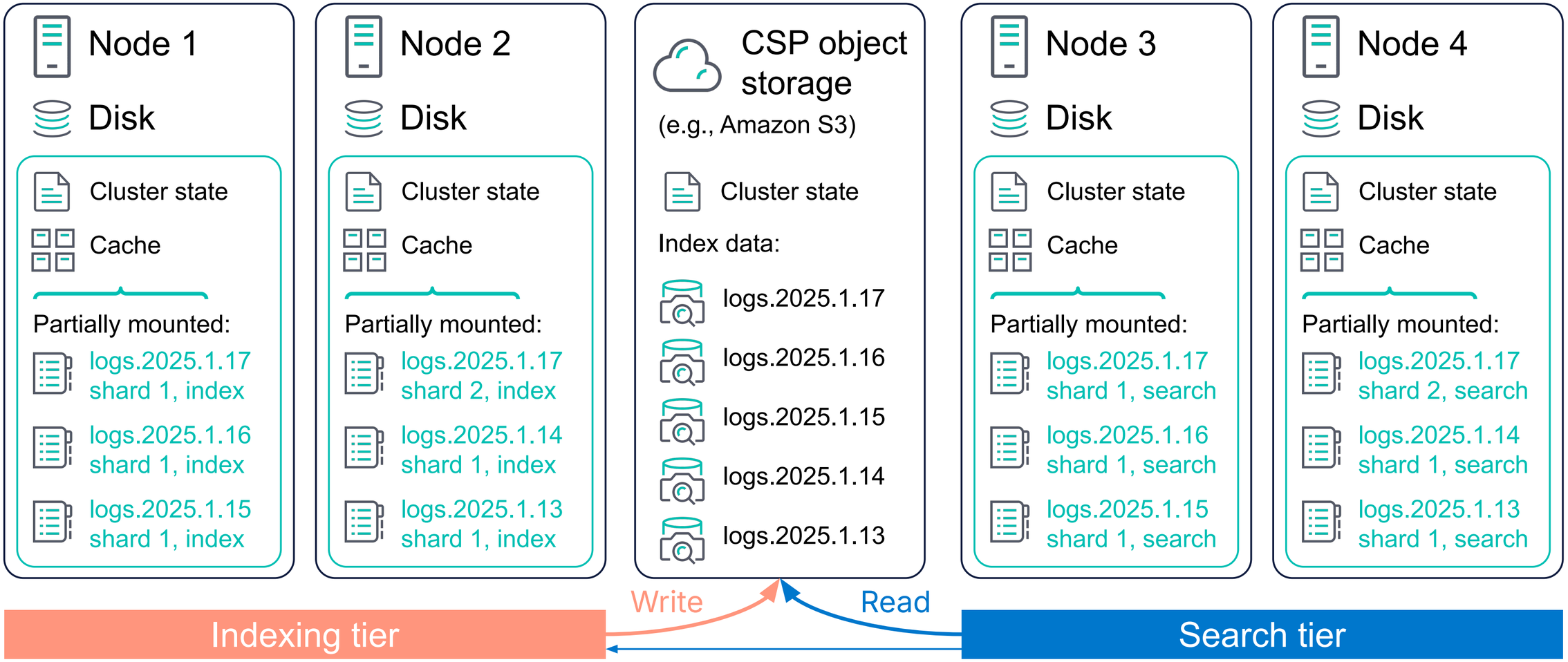
- User defines cluster for their performance
  - Considers datase & workload size, and H/W (RAM, CPU, disk)
- Provisioning CPU & RAM coupled with storage → underutilization
- CPU shared between ingestion & search workloads → overprovisioning
- User considers different data tiers and ILM policies
  - To balance cost, performance, and high availability
- Adding/removing nodes moves data w/ inter-node traffic → slow, expensive
  - Leads to infrequent upgrade cadence, lowering security (e.g., CVEs)
  - Admins need to deal with upgrade issues, e.g., BWC issues

Serverless Elasticsearch

- SaaS (software as a service)
- Exploits CSP: resilience & scalability over commodity H/W
- Stateless: decouples compute from storage
  - Offload data to an affordable highly available object store → no replicas
  - Partial caching to dynamically load data as required by ingestion or searches
- Same API & read-after-write semantics as stateful ES
- Pay-as-you-go model for data retention, indexing, searching
- Without worrying about cluster specifics, sizing, upgrading, HA
- 2 simplified data tiers: (a) indexing, (b) search
  - Autoscaling based on workload

https://www.elastic.co/cloud/serverless, https://www.elastic.co/blog/elastic-cloud-serverless-pricing-packaging

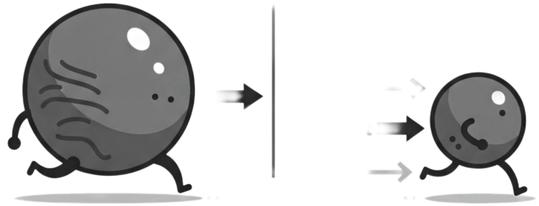
New stateless architecture



https://www.elastic.co/blog/elastic-serverless-architecture, https://www.elastic.co/search-labs/blog/stateless-your-new-state-of-find-with-elasticsearch

elastic

Thin shards

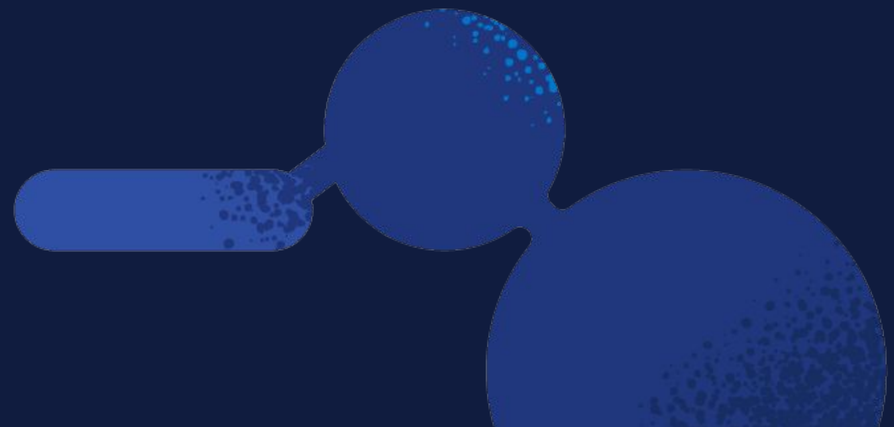


- Loaded with metadata until data is indexed/searched
- Indexing generates files, uploads them and deletes them when unused
- Searching loads from object store only the required data into the cache
- Indexing & searching virtually limitless data
- Not dependent on disk, not restricted by disk capacity → stateless
  - Durability is guaranteed by the object store
- Shard relocation (and recovery) is very fast
  - Indexing shard has a hand-off phase to transition workload
  - Removing/adding nodes is quick → autoscaling
  - Transparent frequent rolling upgrades to a cluster

https://www.elastic.co/search-labs/blog/thin-indexing-shards-elasticsearch-serverless

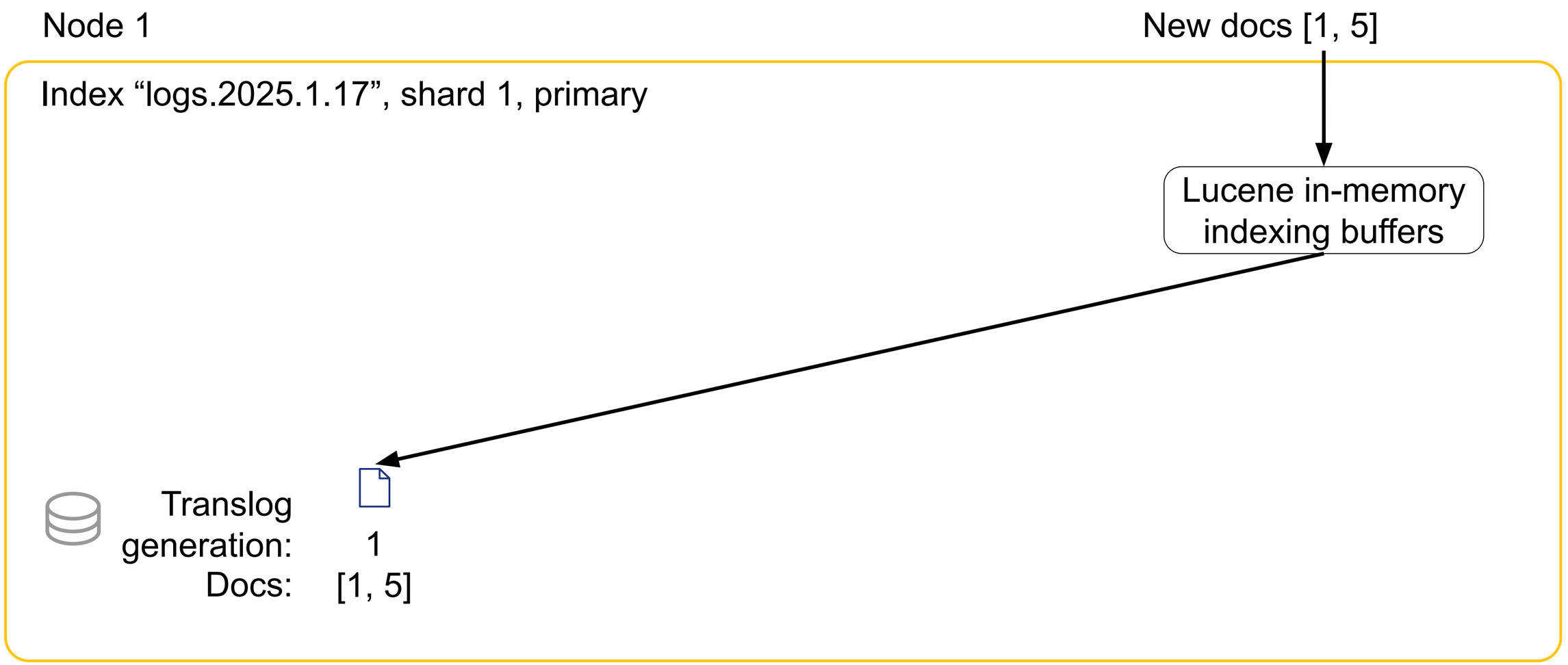
Stateful example

To understand data inside a shard



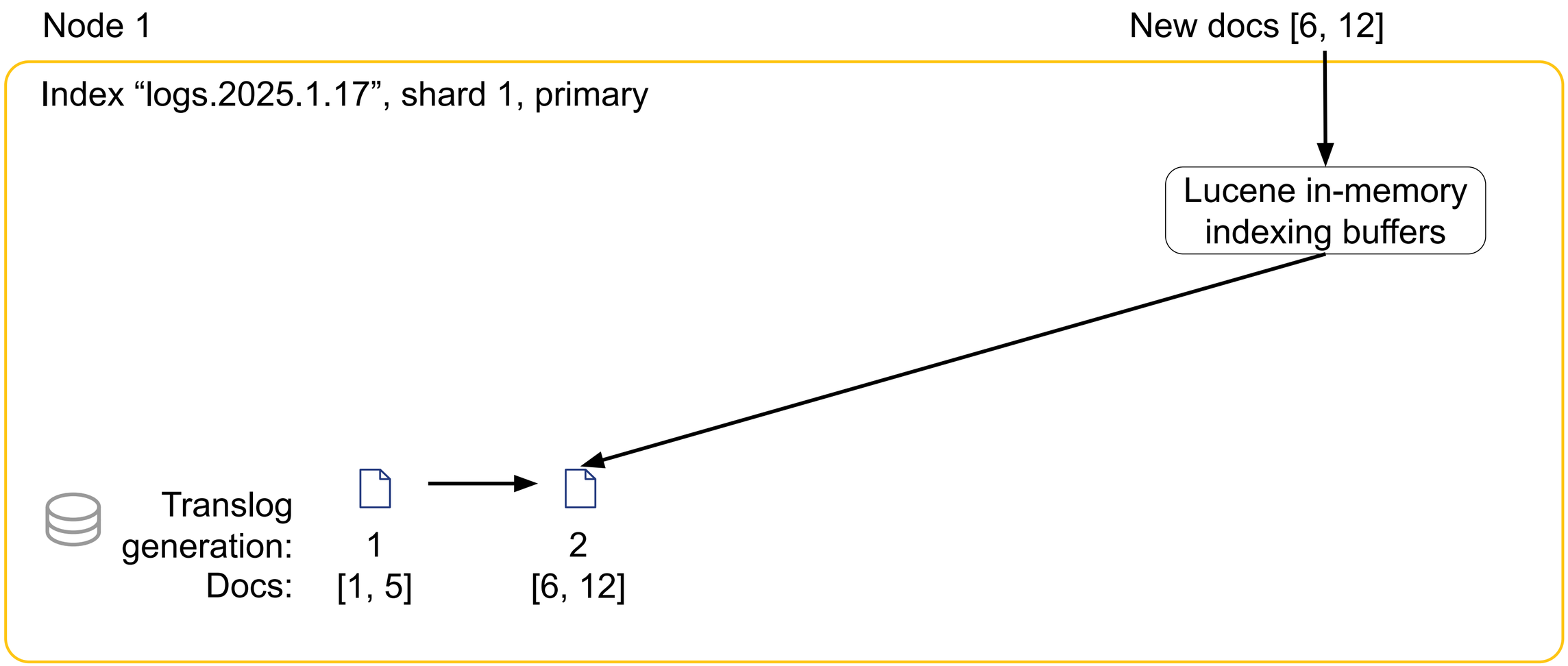
elastic

Data inside a shard (stateful)



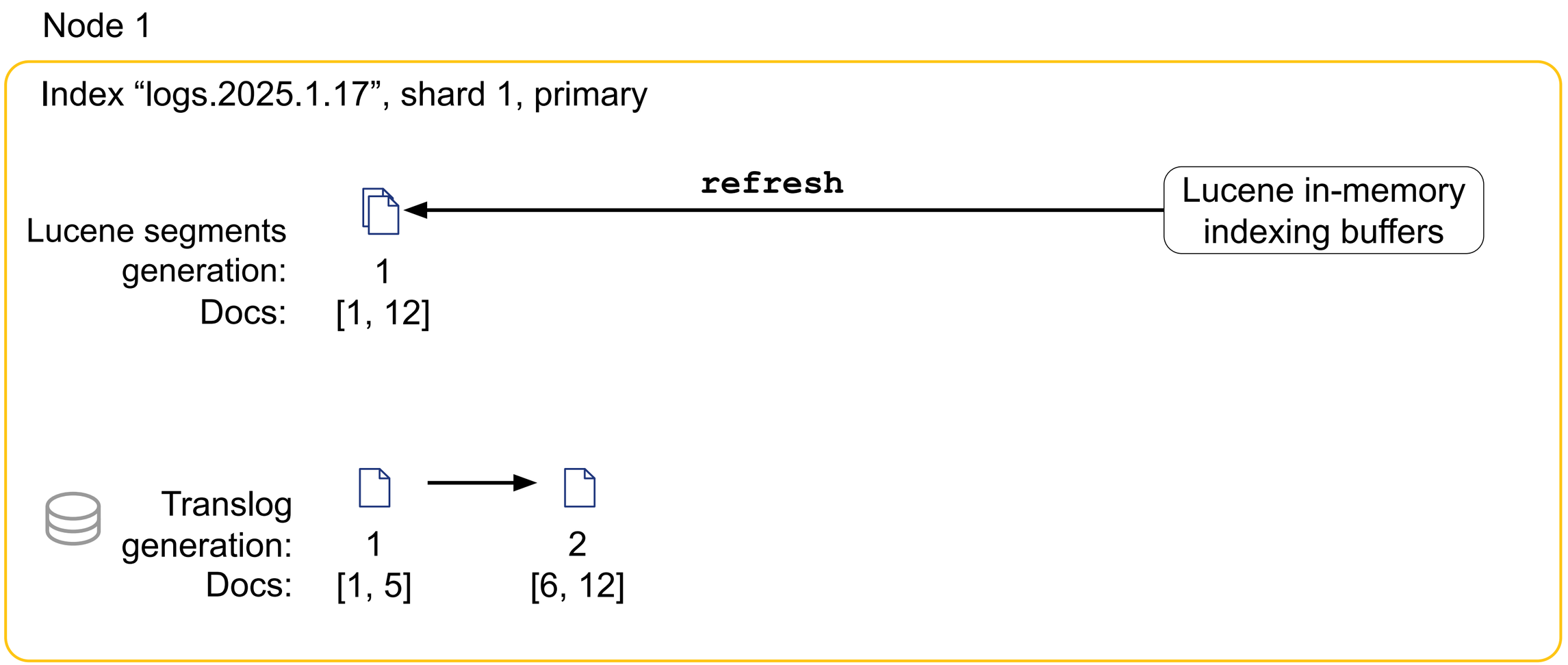
elastic

Data inside a shard (stateful)



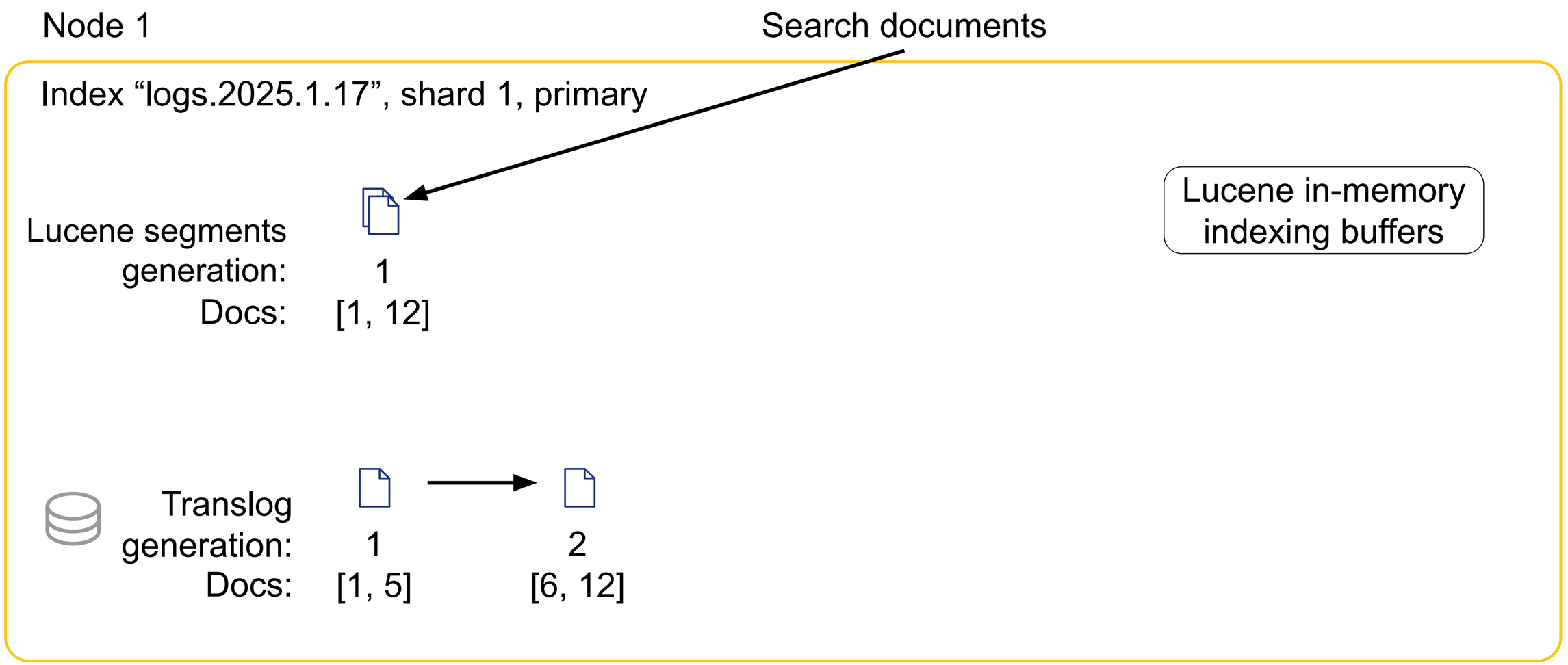
elastic

Data inside a shard (stateful)



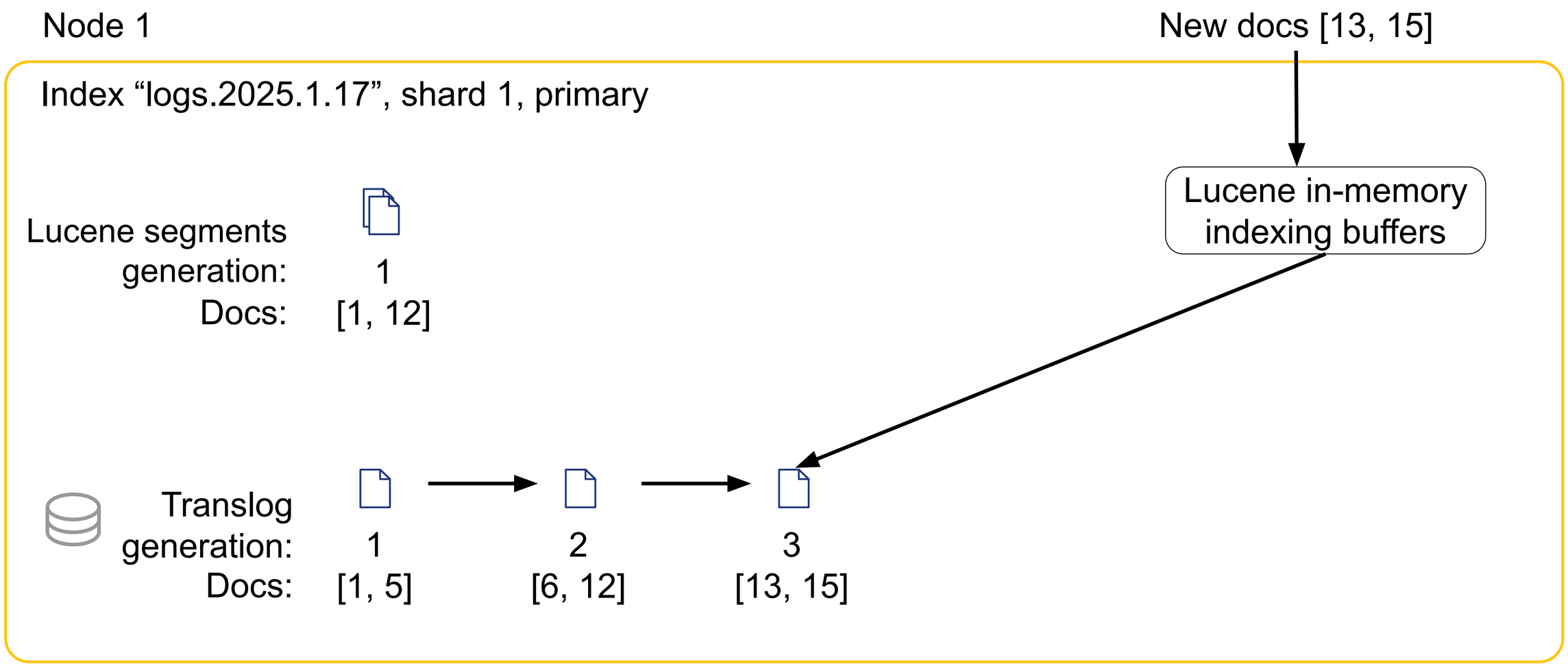
elastic

Data inside a shard (stateful)



elastic

Data inside a shard (stateful)

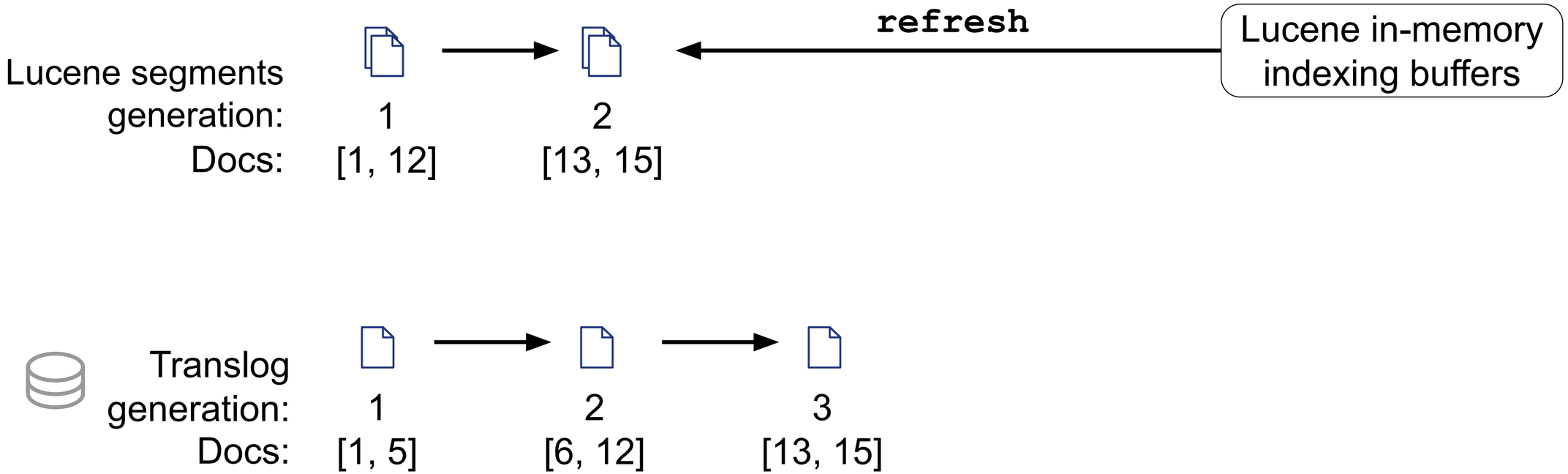


elastic

Data inside a shard (stateful)

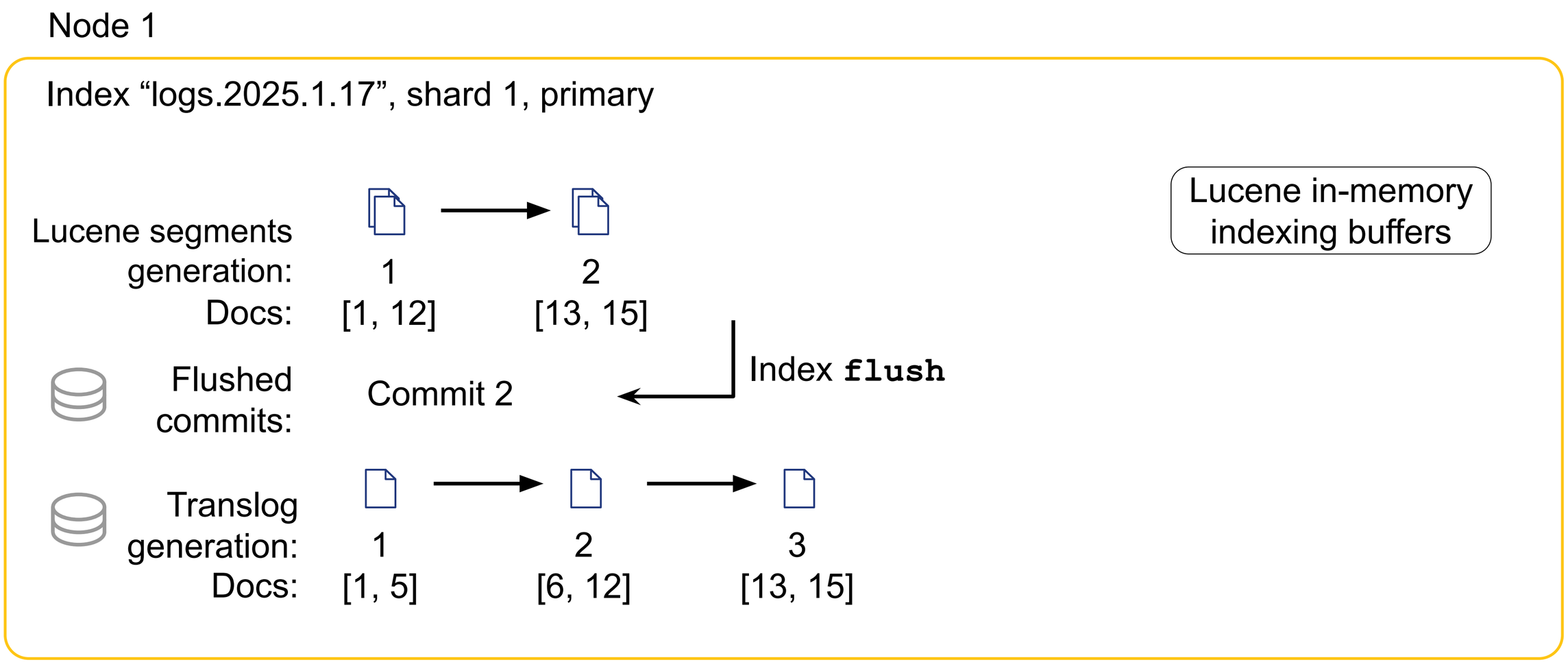
Node 1

Index “logs.2025.1.17”, shard 1, primary



elastic

Data inside a shard (stateful)



Data inside a shard (stateful)

Node 1

Index “logs.2025.1.17”, shard 1, primary

Lucene segments generation: 1 2

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

Lucene in-memory indexing buffers

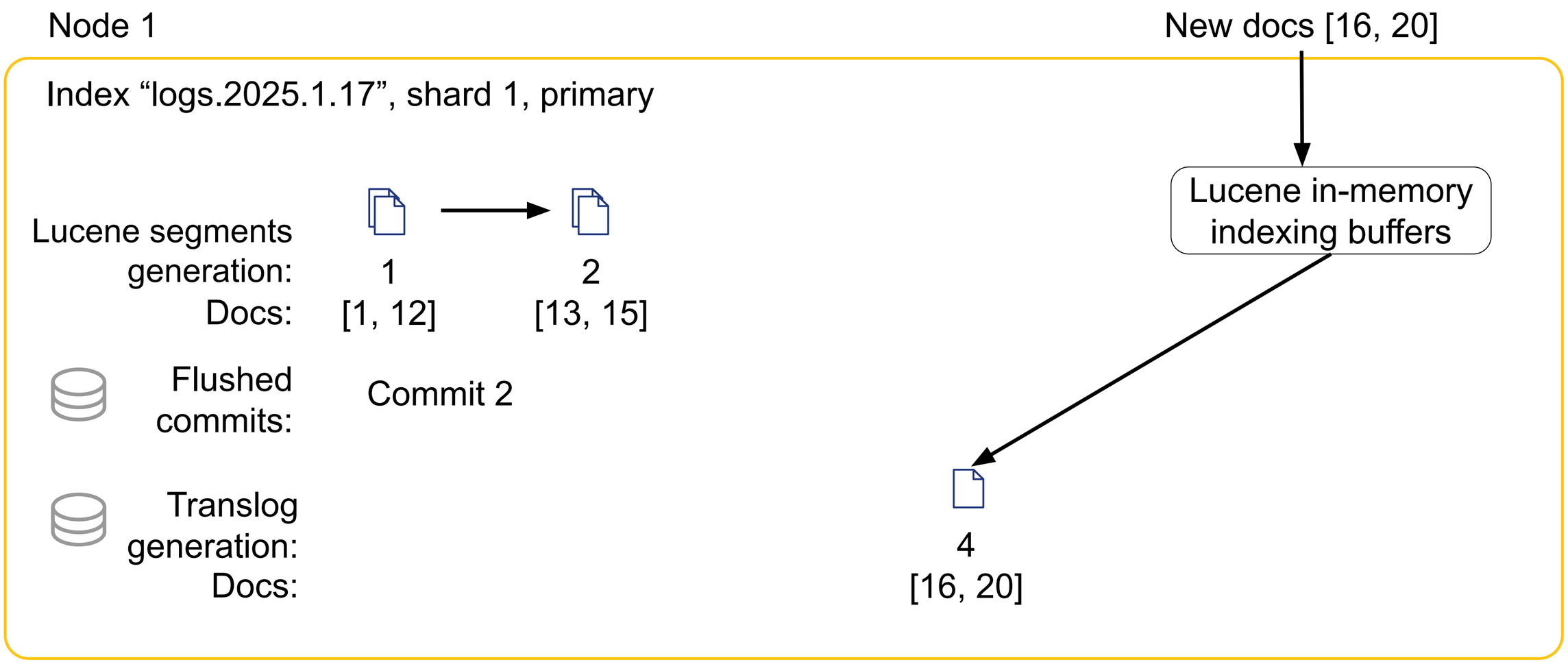
Flushed commits: Commit 2

Translog generation:

Docs:

elastic

Data inside a shard (stateful)

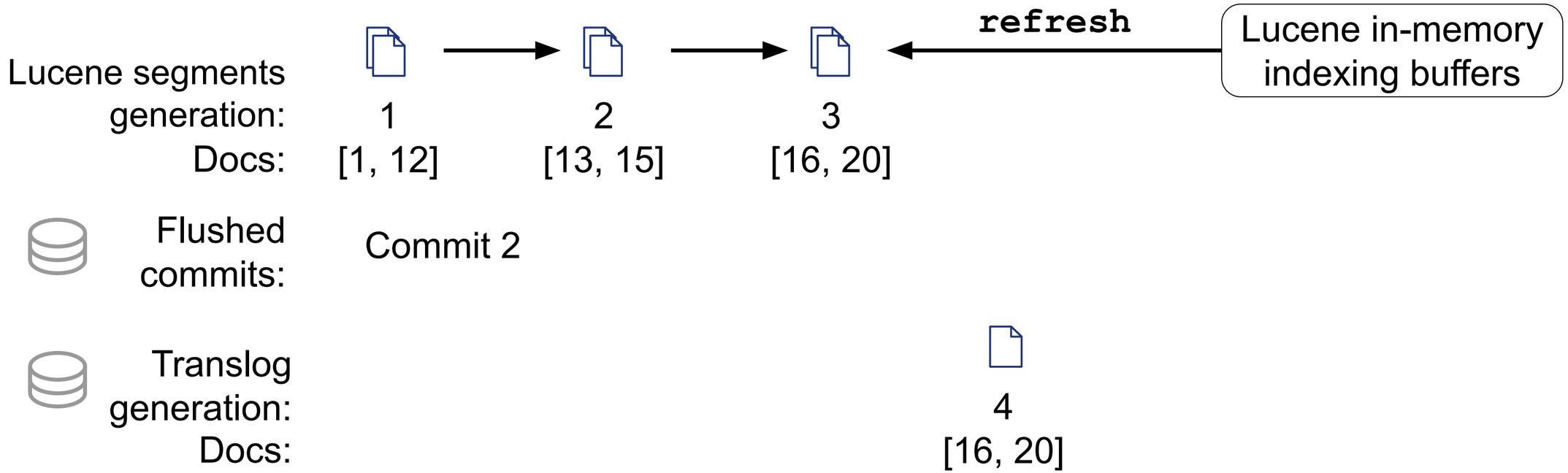


elastic

Data inside a shard (stateful)

Node 1

Index “logs.2025.1.17”, shard 1, primary

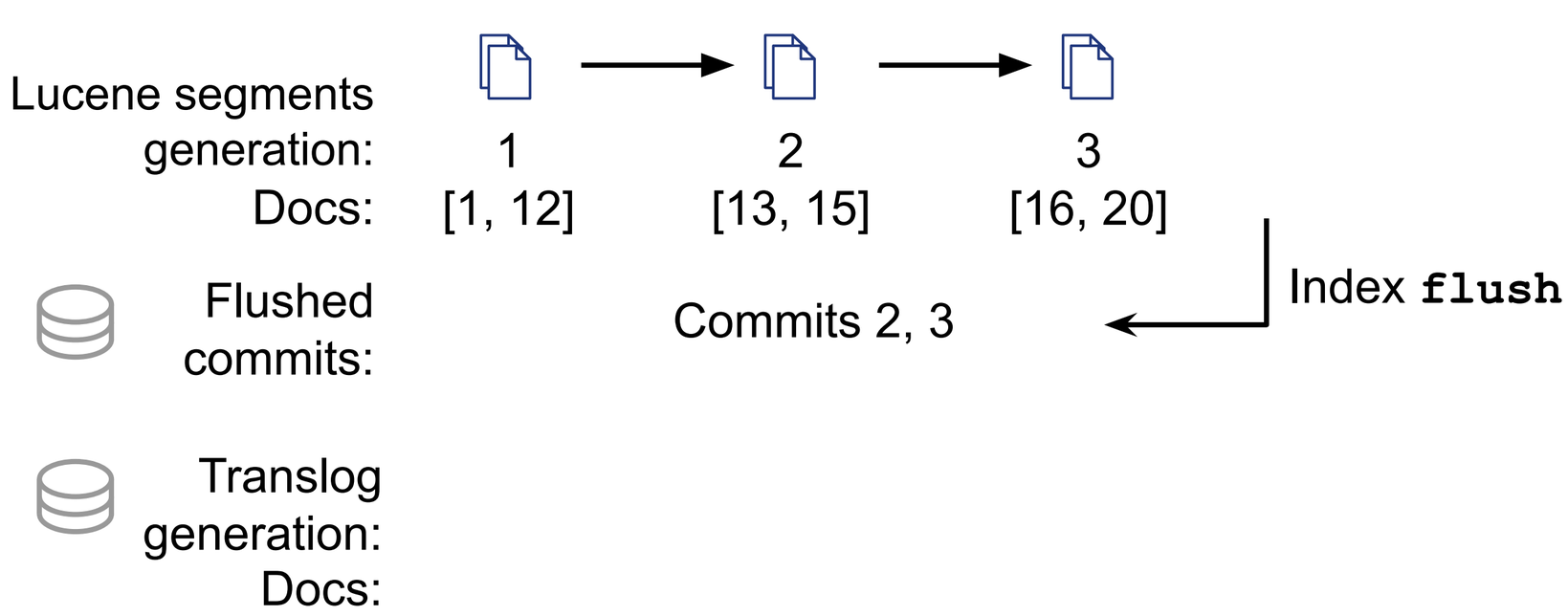


elastic

Data inside a shard (stateful)

Node 1

Index “logs.2025.1.17”, shard 1, primary

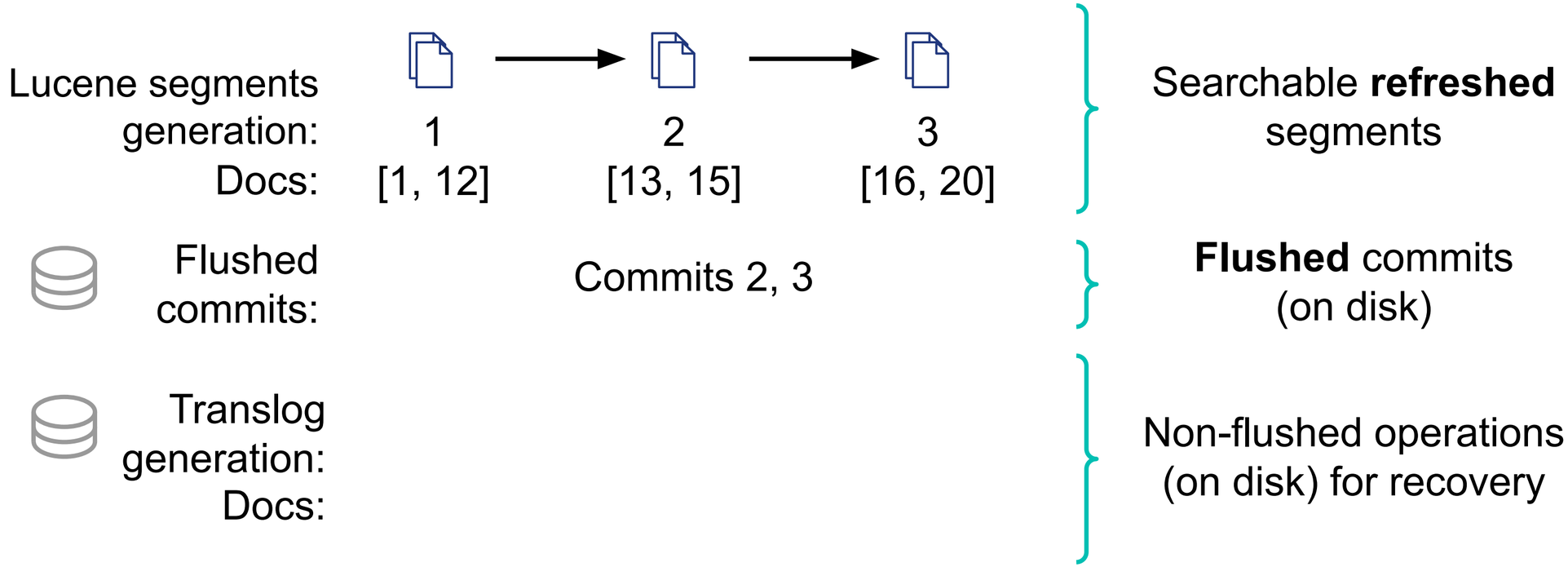


elastic

Data inside a shard (stateful)

Node 1

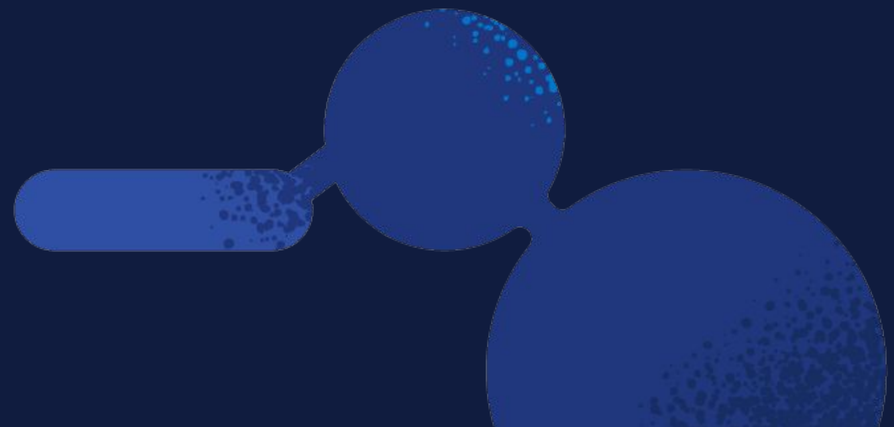
Index “logs.2025.1.17”, shard 1, primary



elastic

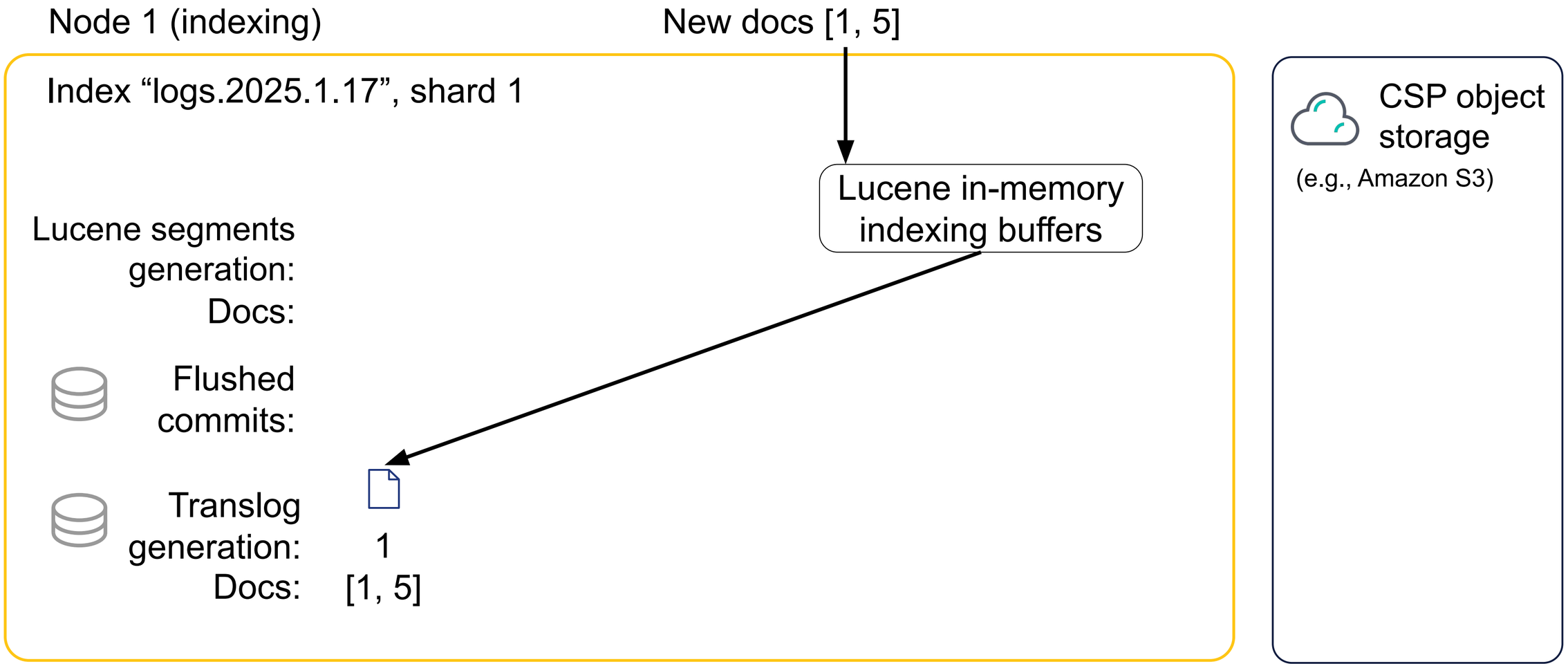
Stateless example

How we offload shard data to the object store



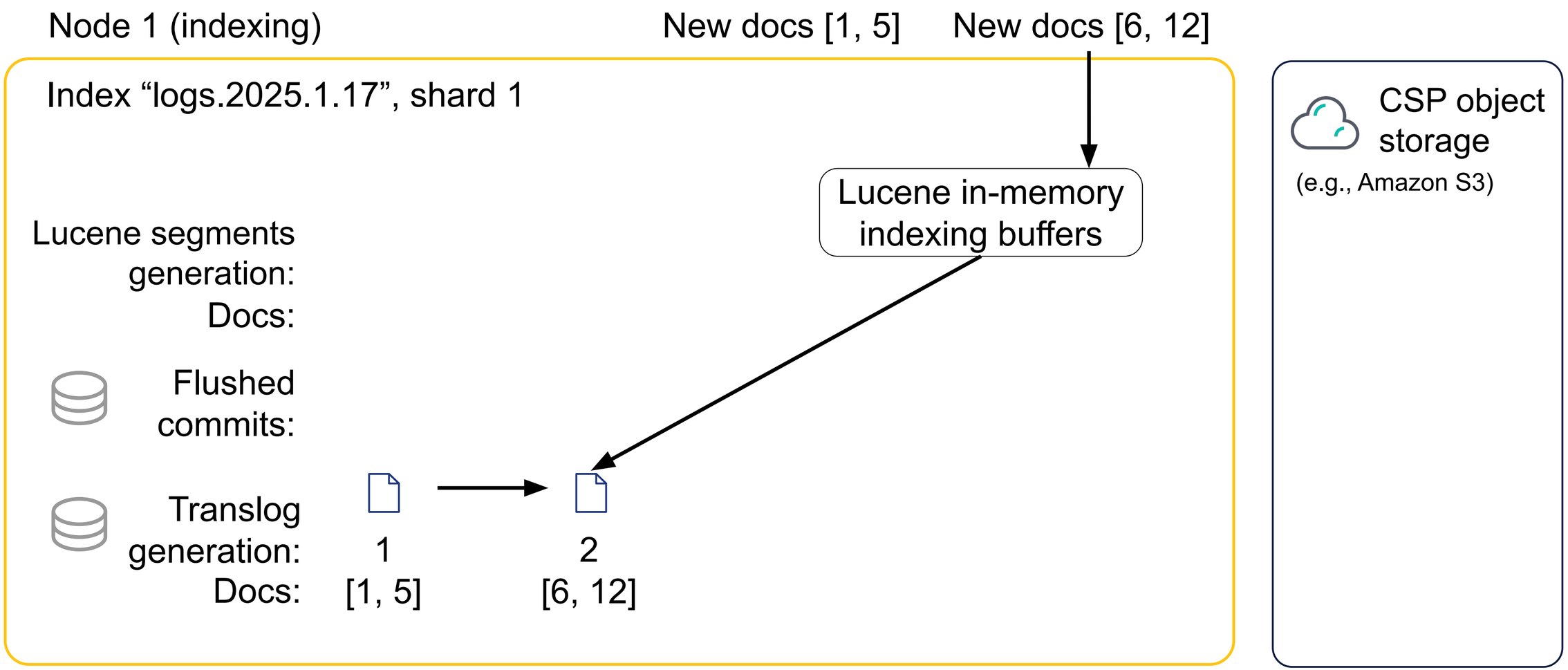
elastic

Offloading index data to the object store (stateless)



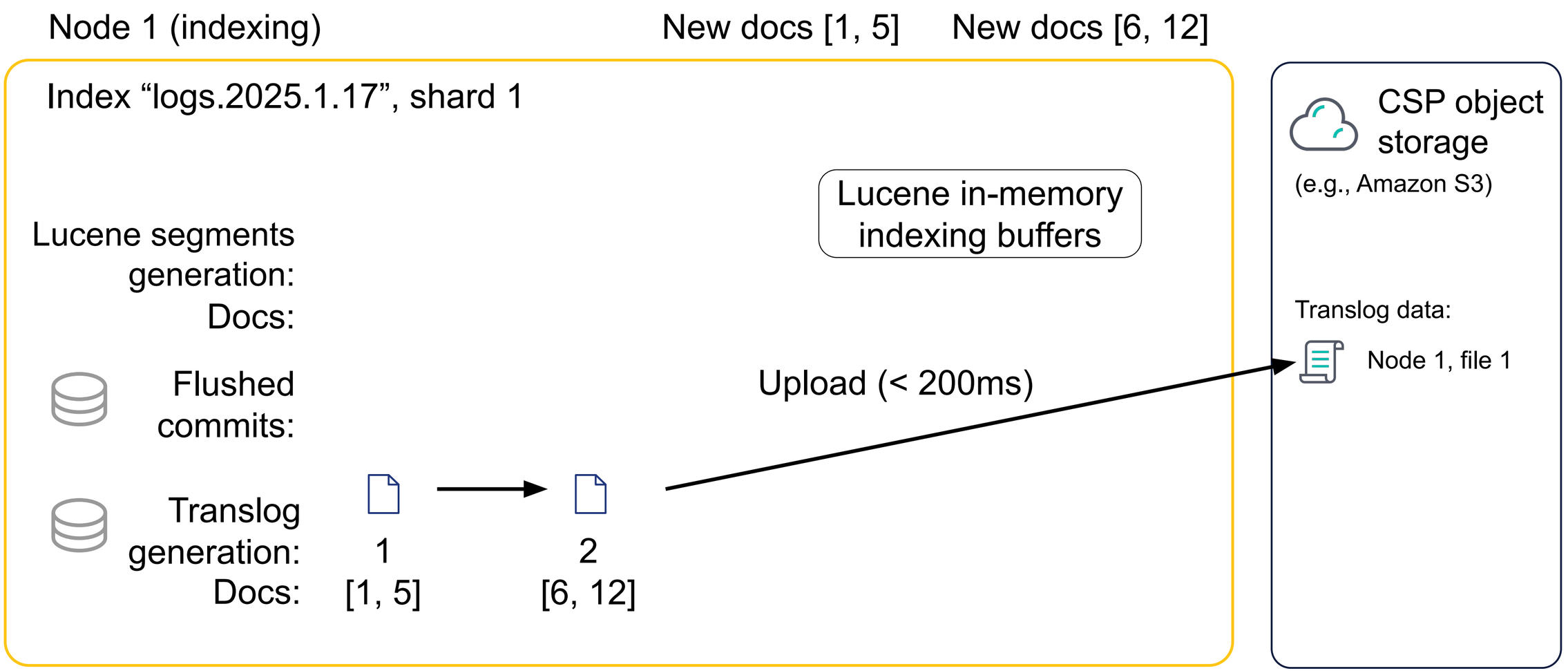
elastic

Offloading index data to the object store (stateless)

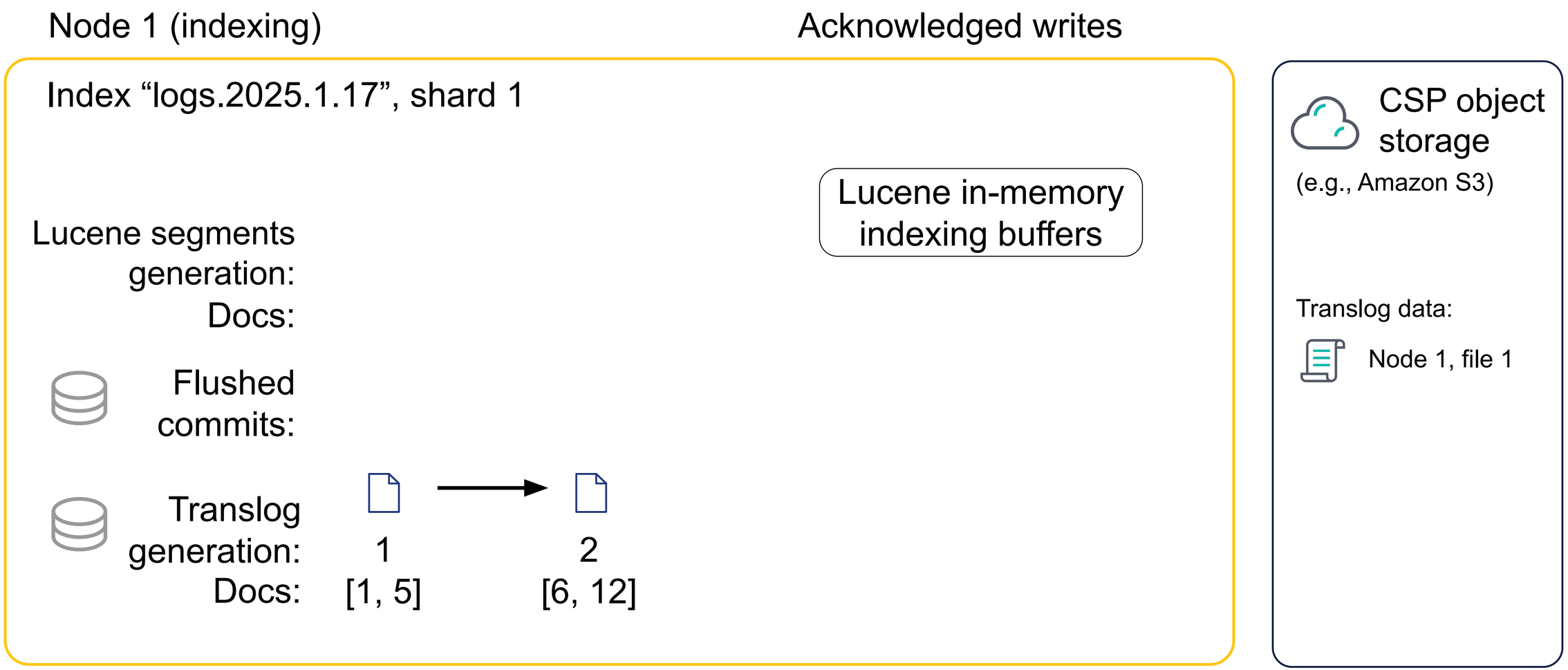


elastic

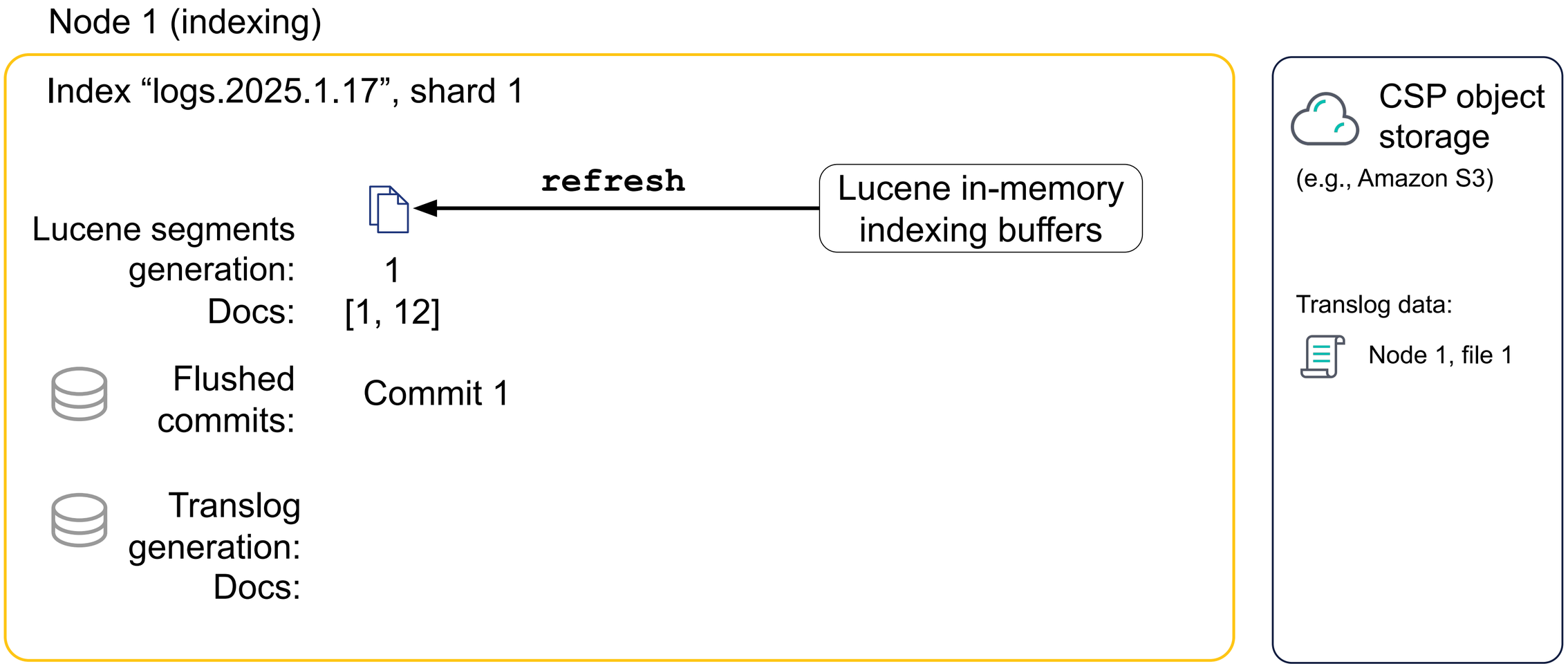
Offloading index data to the object store (stateless)



Offloading index data to the object store (stateless)

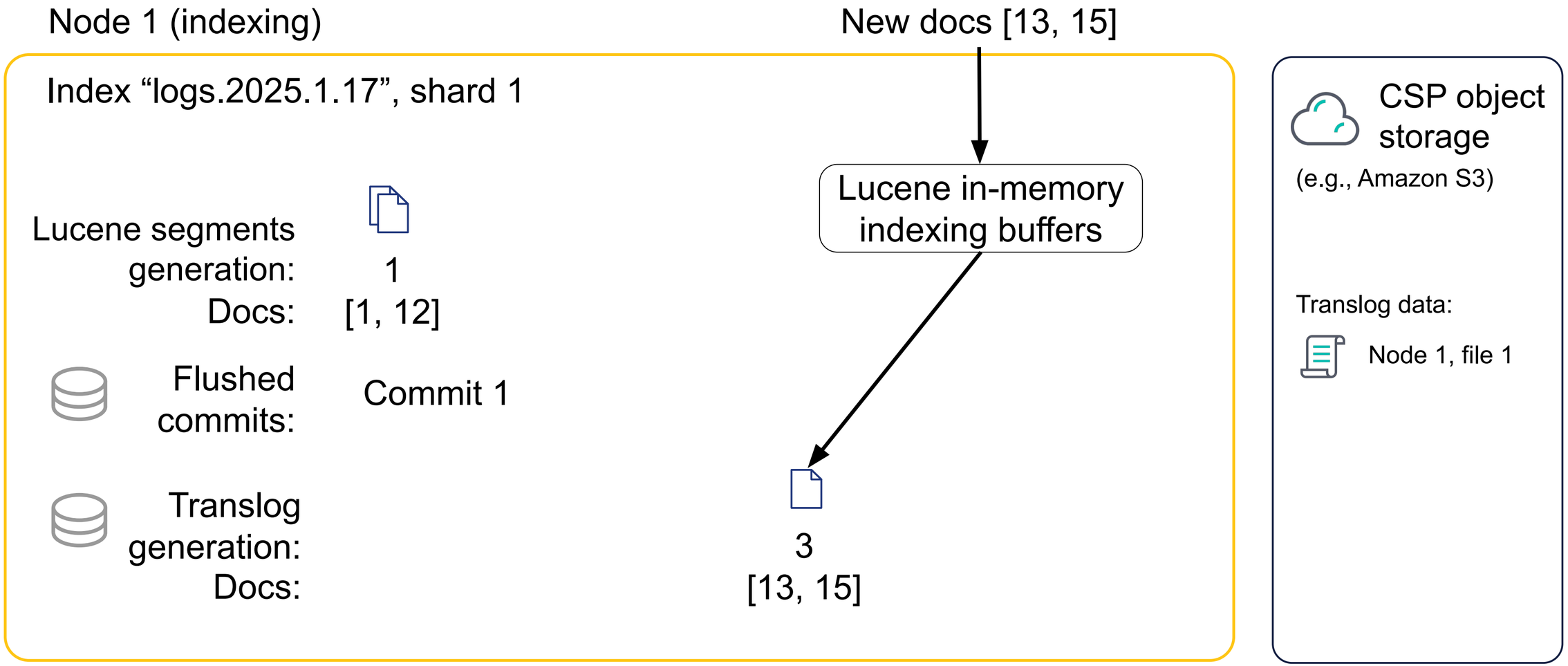


Offloading index data to the object store (stateless)

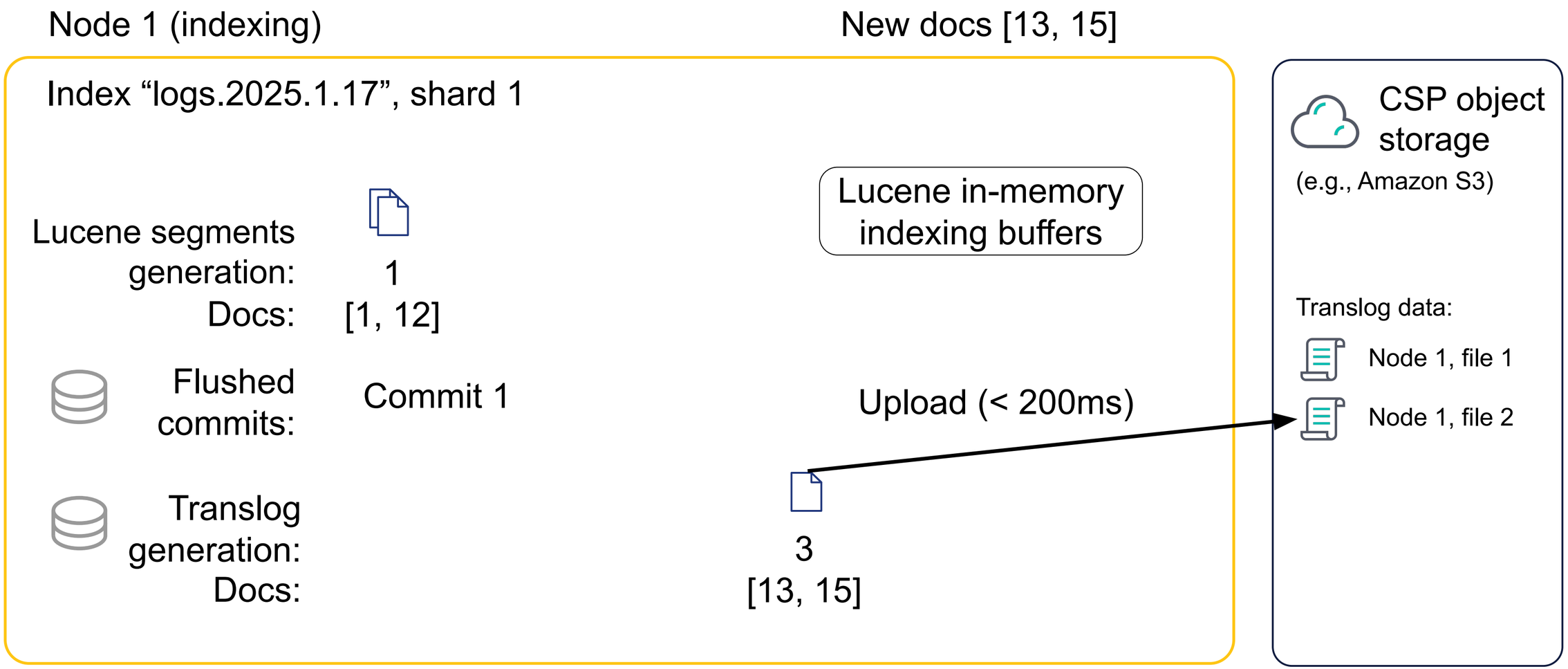


elastic

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.2025.1.17”, 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)

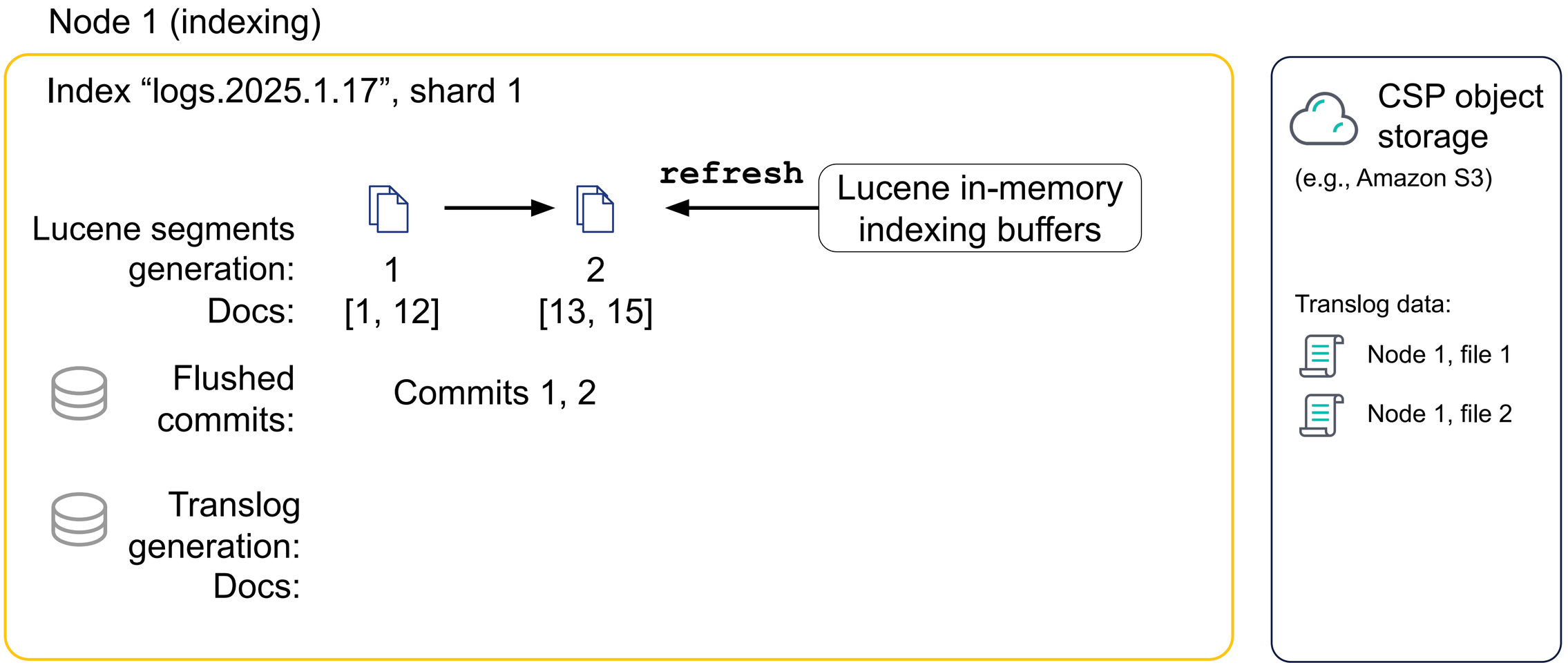
Translog data:

Node 1, file 1

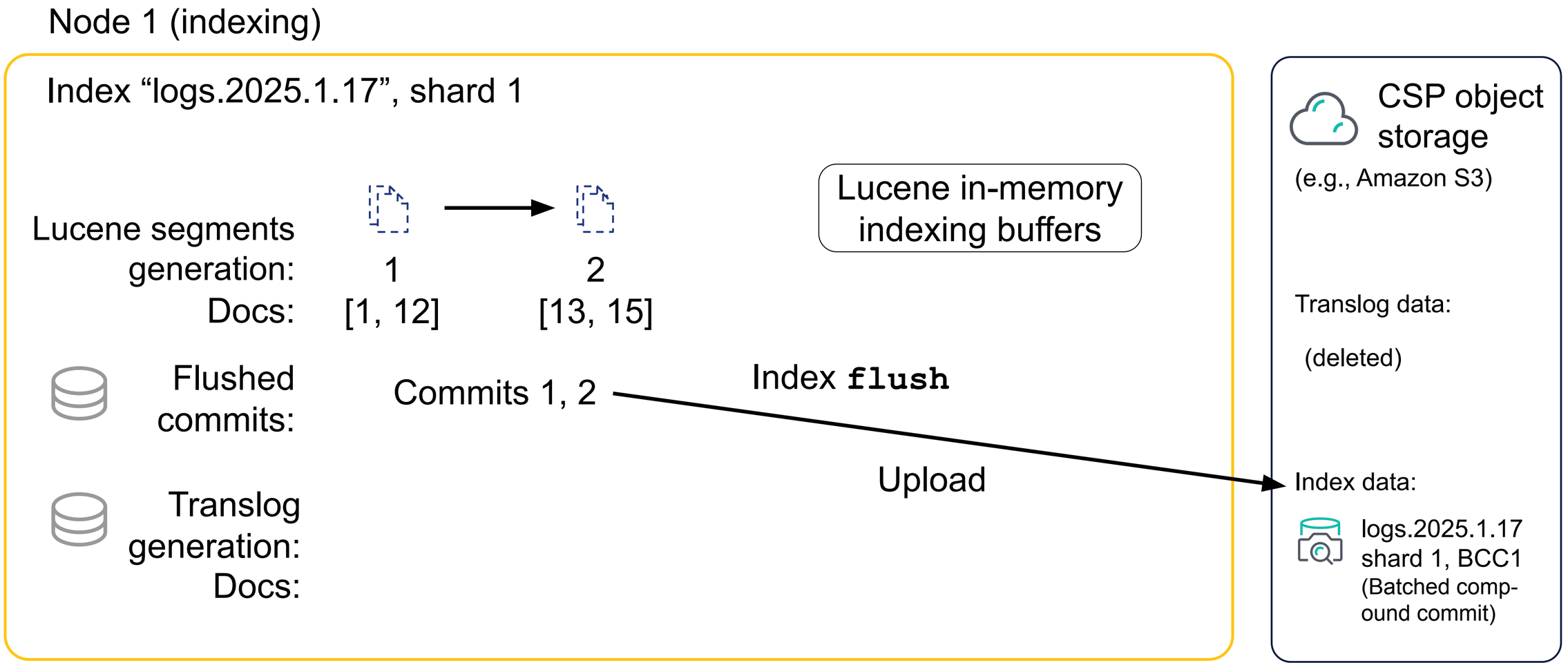
Node 1, file 2

elastic

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 2 (**search**)

Index “logs.2025.1.17”, shard 1

Lucene segments generation: 1 2

Docs: [1, 12] [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)

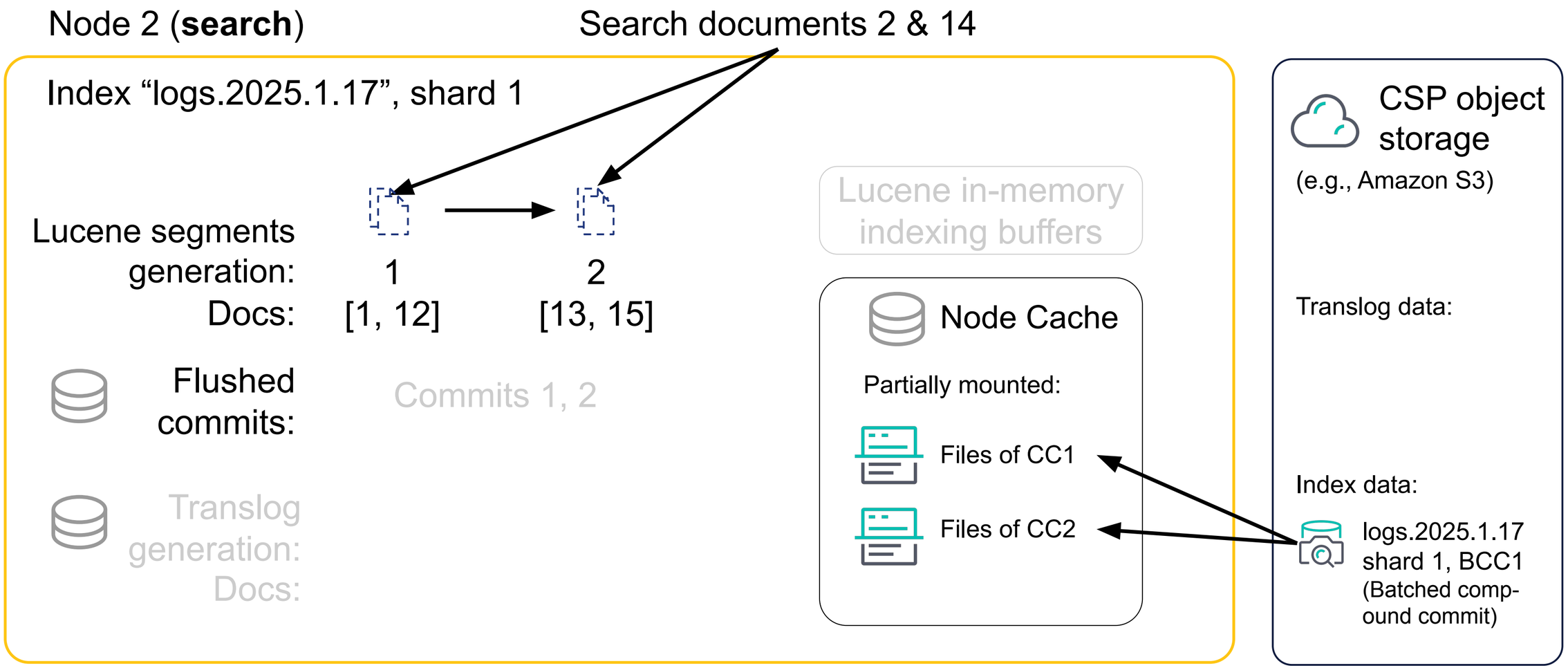
Translog data:

Index data:

logs.2025.1.17 shard 1, BCC1 (Batched comp-ound commit)

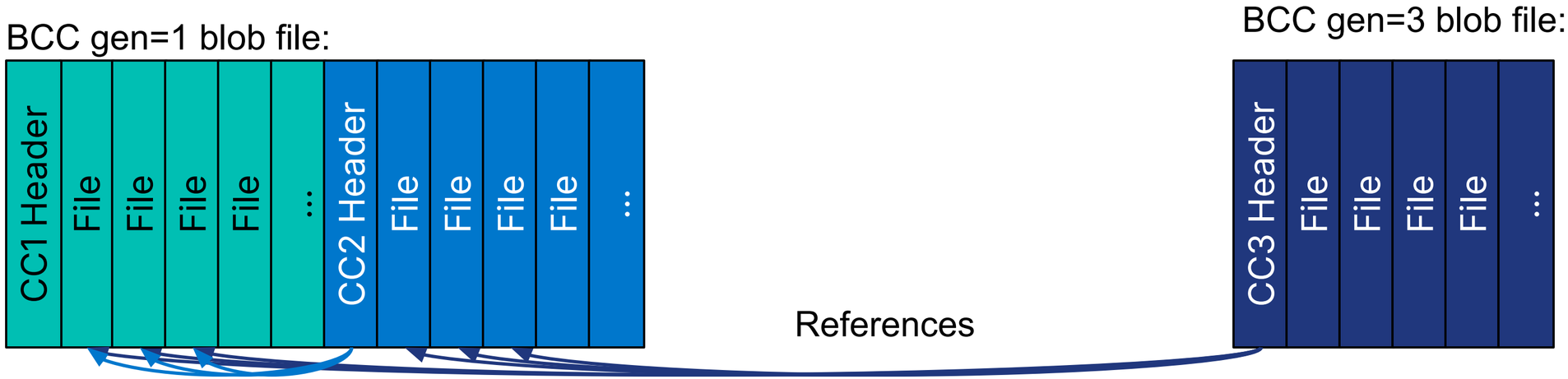
elastic

Offloading index data to the object store (stateless)



elastic

Batched Compound Commit



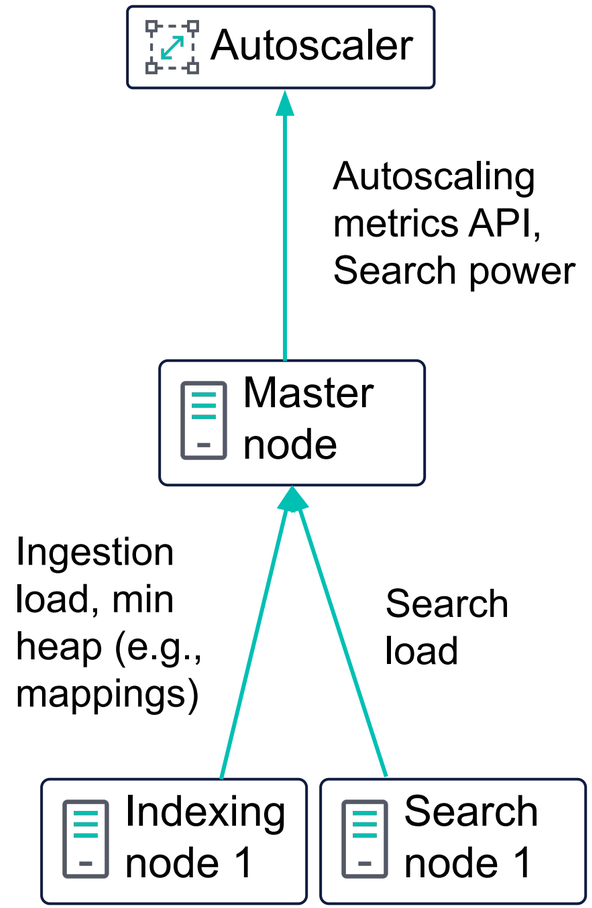
- Minimizes PUT/GETs to the CSP to only BCC-level blob
- To search refreshed, but not yet uploaded data, a search node may request data from the last “virtual” (being built) BCC on the indexing node.

https://www.elastic.co/search-labs/blog/elasticsearch-refresh-costs-serverless

elastic

Autoscaling

- Autoscaler adjust cluster size up/down to match search and ingestion load
- Nodes report:
  - Estimated minimum heap memory
  - CPU load & queuing, reflects I/O
- Search power (user range setting):
  - #VCUs (fraction of RAM, CPU, disk cache) allocated to boosted data (new time-based docs, and non-time-based docs) → responsivesness
- Autosharding data streams



https://www.elastic.co/search-labs/blog/elasticsearch-ingest-autoscaling, https://www.elastic.co/search-labs/blog/elasticsearch-serverless-tier-autoscaling, https://www.elastic.co/pricing/serverless-search, https://www.elastic.co/search-labs/blog/datastream-autosharding-serverless

elastic

Summary

- Elasticsearch is an unparalleled scalable search & analytics engine
- ES has been stateful, relying on disk durability
- New **Serverless ES** is stateless; durability on CSP **object store**
  - Showed how we upload BCCs and translogs cost-efficiently
- Just two **tiers, indexing and search** → independent workloads
- Partial caching enables indexing and searching limitless data
- Thin shards that can relocate/recover fast
  - Frequent automatic hassle-free **upgrades**, better security
  - Workload-based **autoscaling**
- Users **just use the same API**
- Pay-as-you-go model for data retention, indexing, searching

Community, culture and careers

- Start a trial at cloud.elastic.co
- Register for an upcoming event at elastic.co/events
- Vibrant community
  - Community portal → www.elastic.co/community
  - Elastic Community on Slack → ela.st/slack
  - Community videos → ela.st/community-youtube
  - Discussion forums → discuss.elastic.co
  - Elastic Contributor Program (e.g., earn training) → elastic.co/community/contributor
  - Community events & groups across the globe → community.elastic.co
  - Newsletter
- Careers → elastic.co/about/careers
  - Elastic Source Code, remote, 2800+ employees across 40+ countries
- Subscribe for next meetup.com/greece-elastic event

elastic

Thank You!

Iraklis Psaroudakis
www.kingherc.com