elastic | The Search Analytics Company

The building blocks of Elasticsearch's massive scalability

Iraklis Psaroudakis

15th of April 2024 at codeweek.

Welcome

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
- How Elasticsearch scales out with shards
- Distributed searches & aggregations
- Shard recovery
- Cluster state
- Scaling with data streams
- Scaling with ILM
- Vision: scaling with Serverless

This talk contains personal views and is not officially endorsed

elastic

Meet Elastic — The Search Analytics 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

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

Elastic Search

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

Elasticsearch™ 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*

Trusted by Organizations Around the World

TECHNOLOGY	FINANCE	TELCO	CONSUMER	HEALTHCARE	PUBLIC SECTOR	AUTOMOTIVE / TRANSPORTATION	RETAIL
Adobe	BARCLAYS	orange	Uber	VITAS Healthcare	Lawrence Livermore National Laboratory	VOLVO Volvo Group	AutoZone
CISCO	ZURICH	dish media	Grab	UCLA Health	OAK RIDGE National Laboratory	Audi	The Home Depot
workday	USAA	COMCAST	Miles & More Lufthansa	Yale NewHaven Health	De Watergroep WATER. VANDAAG EN MORGEN.	JAGUAR LAND ROVER	ebay
Microsoft	Swift	verizon	ACTIVISION BLIZZARD	MAYO CLINIC	JPL Jet Propulsion Laboratory	BMW	Kroger
INGRAM MICRO	Postbank	T-Mobile	lyft	Pfizer	MENTAT COMPUTE OPTIMIZATION	Volkswagen	Walgreens

Elasticsearch

- Distributed, scalable, highly available, resilient search & analytics engine
- HTTP based JSON interface
- Flexibility (index time vs. query time)
- Based on Apache Lucene
- Much more than `grep` or SQL's `LIKE = '%quick%`
  - Ranked results (BM25, recency, popularity), fuzzy matching
  - Complex search expressions
  - Spell correction, Synonyms, Phrases, Stemming
- Timeseries, geospatial, ML, vector search

github.com/elastic/elasticsearch

elastic/**elasticsearch**

Free and Open, Distributed, RESTful Search Engine

java · search-engine · elasticsearch

Java · 65k · Updated 2 minutes ago

db-engines.com/en/ranking

Rank			DBMS	Database Model	Score	
Apr 2024	Mar 2024	Apr 2023			Apr 2024	Mar 2024
1.	1.	1.	Oracle	Relational, Multi-model	1234.27	+13.21
2.	2.	2.	MySQL	Relational, Multi-model	1087.72	-13.77
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	829.80	-16.01
4.	4.	4.	PostgreSQL	Relational, Multi-model	645.05	+10.15
5.	5.	5.	MongoDB	Document, Multi-model	423.96	-0.57
6.	6.	6.	Redis	Key-value, Multi-model	156.44	-0.56
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	134.78	-0.01
8.	8.	↓ 7.	IBM Db2	Relational, Multi-model	127.49	-0.26
9.	9.	↑ 12.	Snowflake	Relational	123.20	-2.18
10.	10.	↓ 9.	SQLite	Relational	116.01	-2.15

elastic

Inverted index

Document 1: “The quick brown fox jumped over the lazy dog”
Document 2: “Quick brown foxes leap over lazy dogs in summer”

Map: sorted tokens → documents

Quick	2
The	1
brown	1,2
dog	1
dogs	2
fox	1
foxes	2
in	2
jumped	1
lazy	1,2
leap	2
over	1,2
quick	1
summer	2
the	1

Queries

“lazy AND dog” → [1] AND [1,2] → [1]

“lazy OR dog” → [1] OR [1,2] → [1,2]
and a document can have a higher score (TF/IDF based)

~ Lucene segment

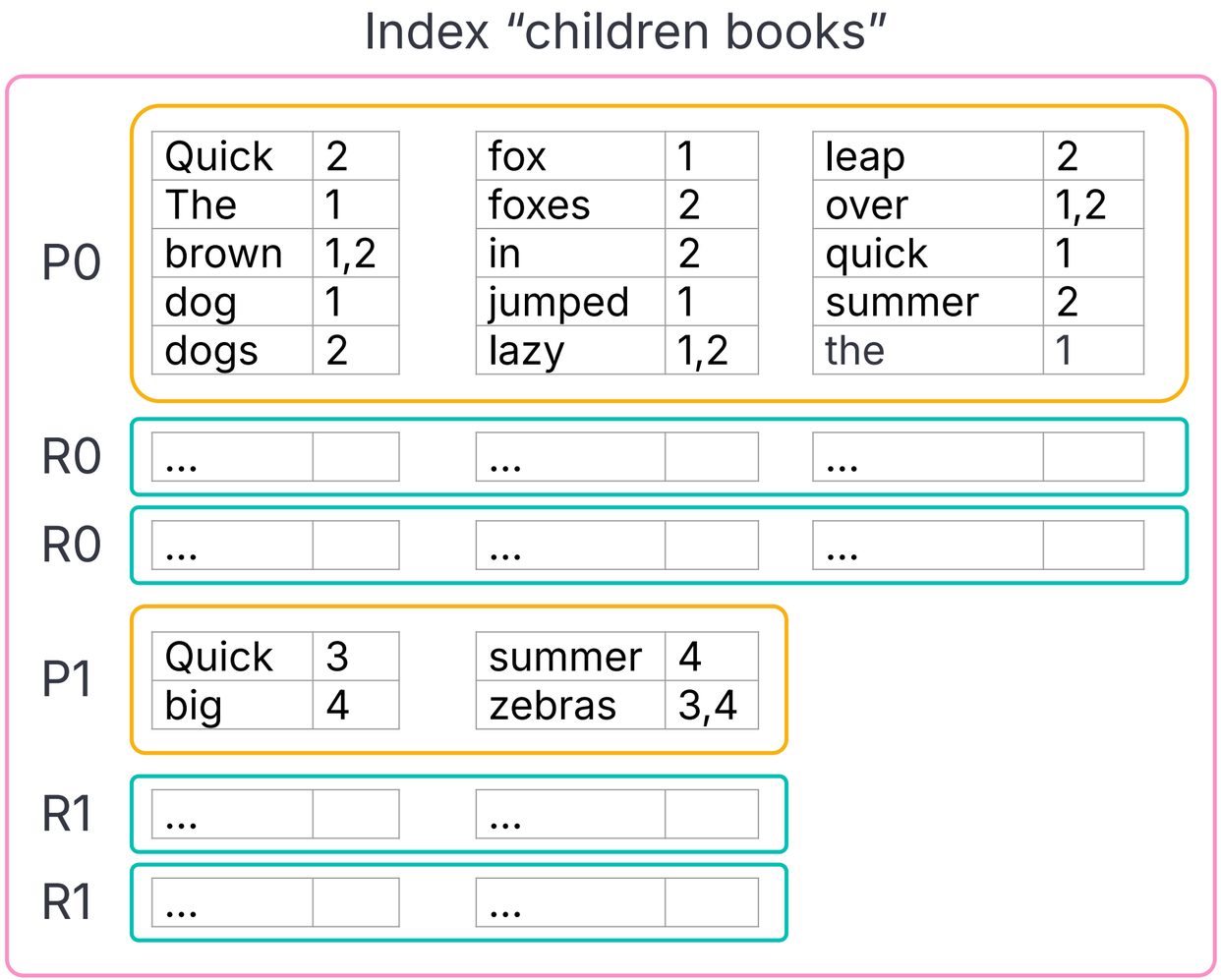
Credits: “Introduction into full-text search with distributed search engines” (2019), “Distributed search under the hood” (2021) by Alexander Reelsen

elastic

Elasticsearch scales with shards

An “index” consists of shards

- A shard is composed of Lucene segments
- **Primary** shards
  - Partitioning of data in the index
  - **Write**/ingest scalability
- **Replica** shard
  - Auto synced copy of a primary
  - **Read**/query scalability

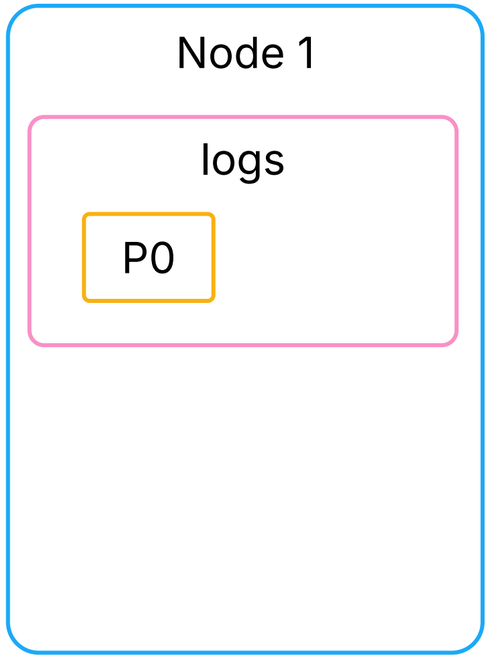


elastic

Scaling out to multiple machines

Primary and replica shards are distributed across the cluster

- In a balanced manner → ingestion and querying distribution
- Replicas are not collocated → fault tolerance

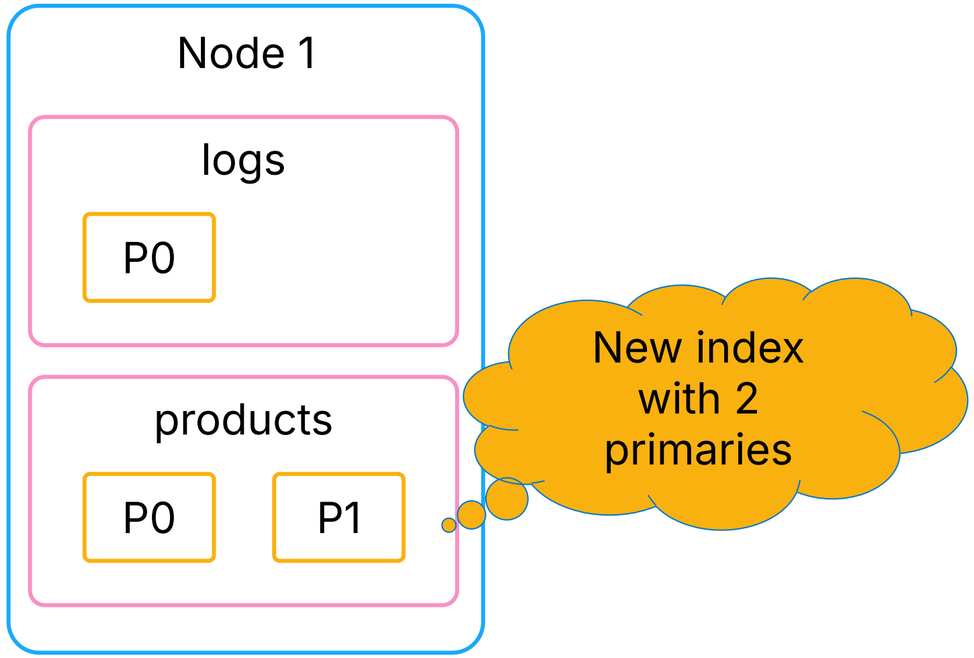


elastic

Scaling out to multiple machines

Primary and replica shards are distributed across the cluster

- In a balanced manner → ingestion and querying distribution
- Replicas are not collocated → fault tolerance

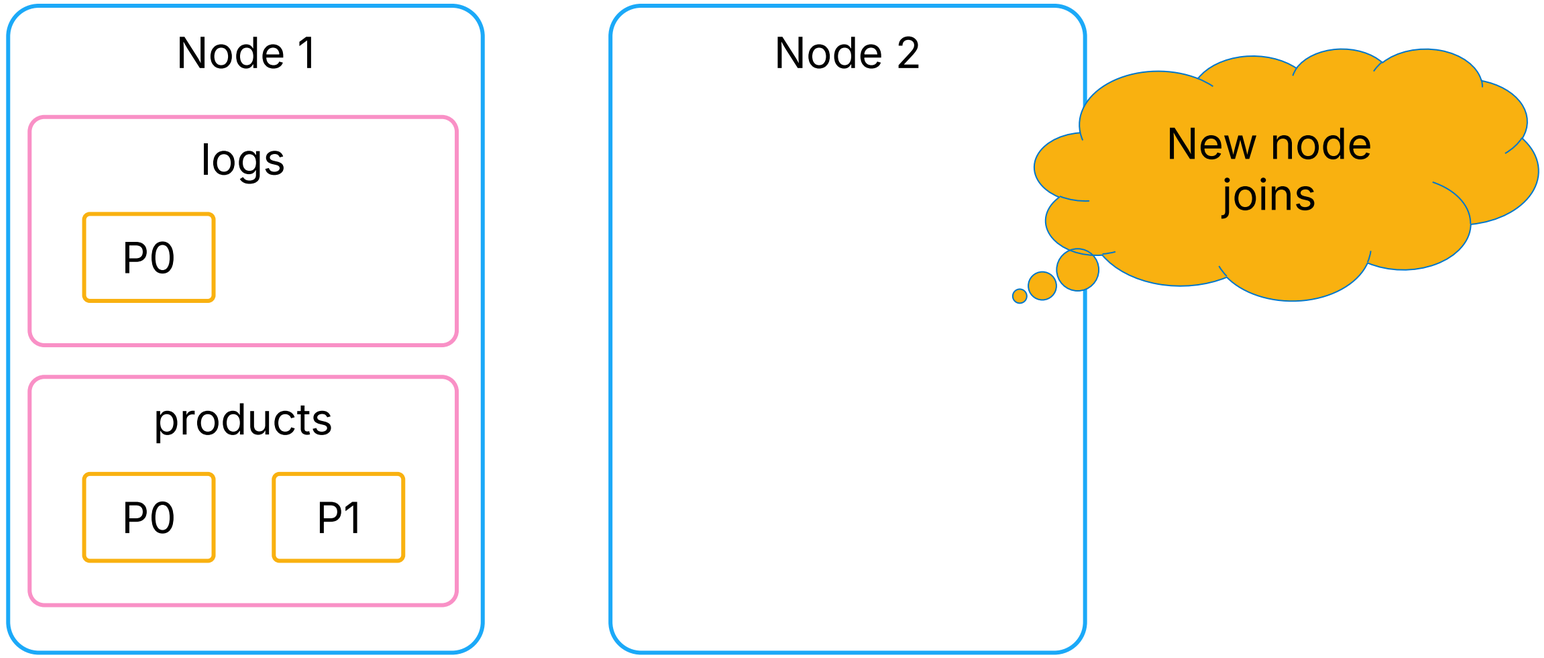


elastic

Scaling out to multiple machines

Primary and replica shards are distributed across the cluster

- In a balanced manner → ingestion and querying distribution
- Replicas are not collocated → fault tolerance

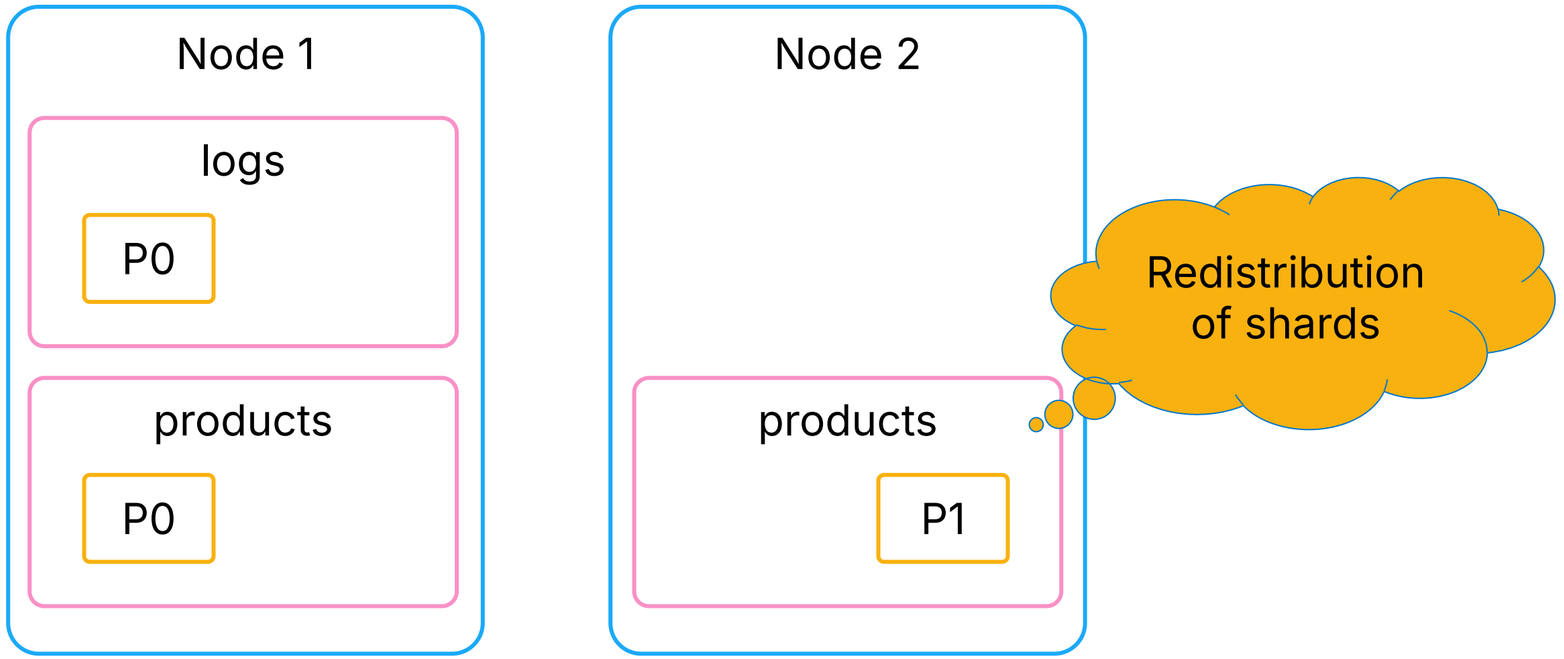


elastic

Scaling out to multiple machines

Primary and replica shards are distributed across the cluster

- In a balanced manner → ingestion and querying distribution
- Replicas are not collocated → fault tolerance

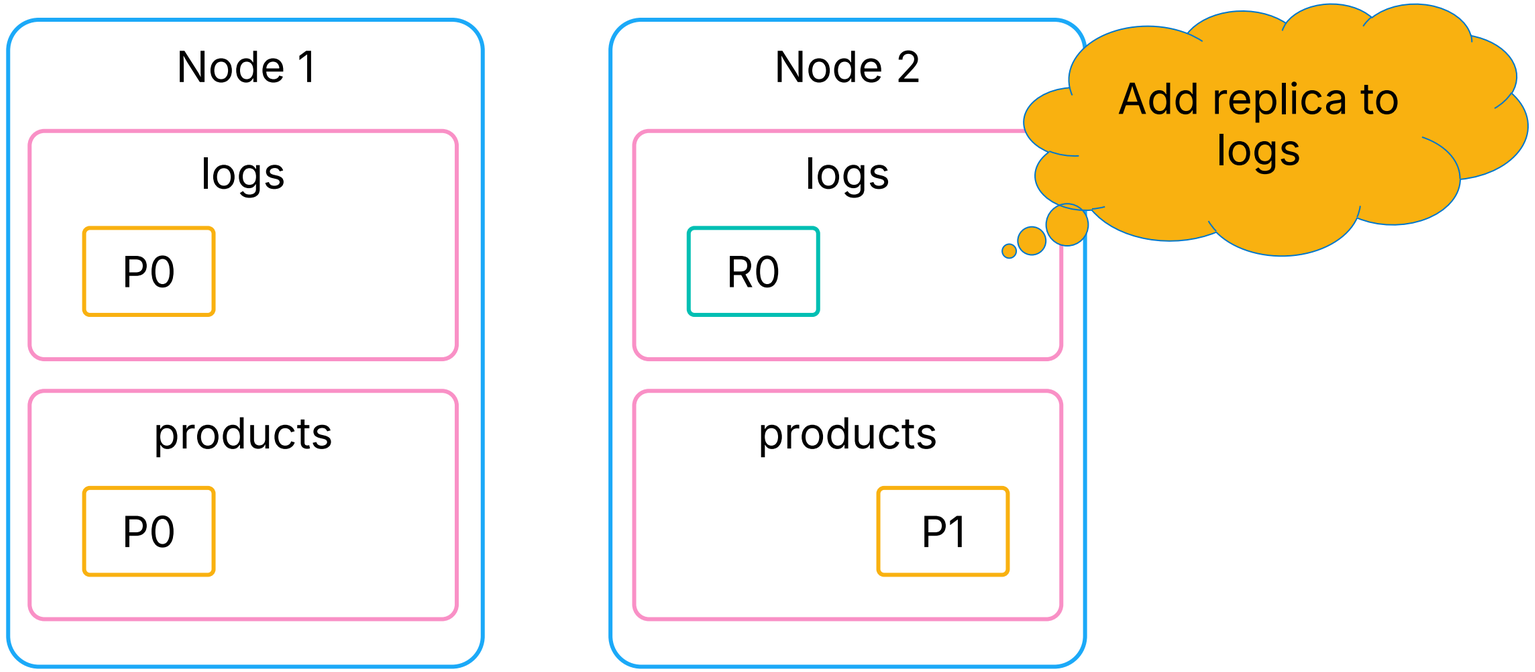


elastic

Scaling out to multiple machines

Primary and replica shards are distributed across the cluster

- In a balanced manner → ingestion and querying distribution
- Replicas are not collocated → fault tolerance

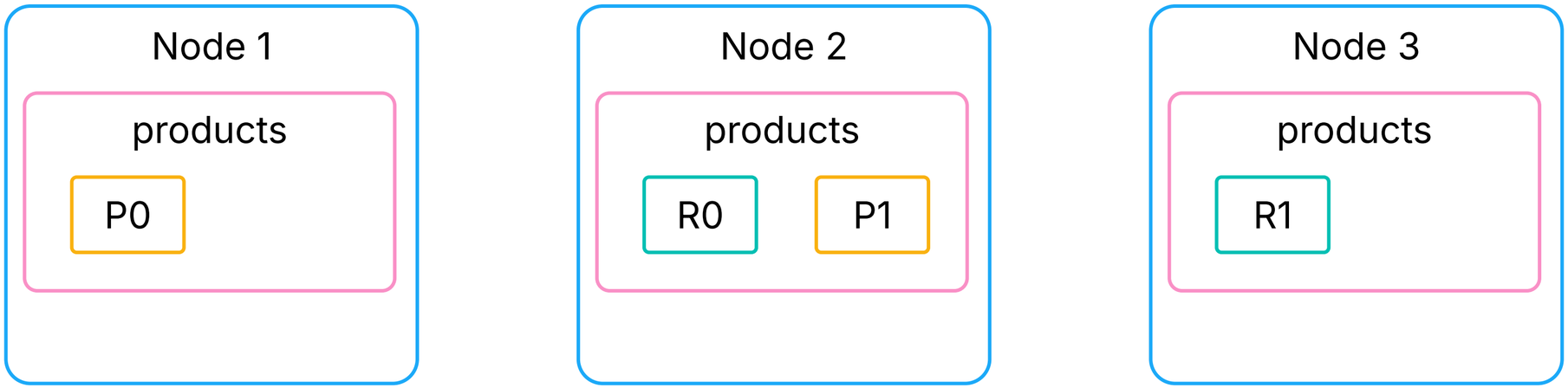


elastic

Distributed search

Two phase approach

- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)

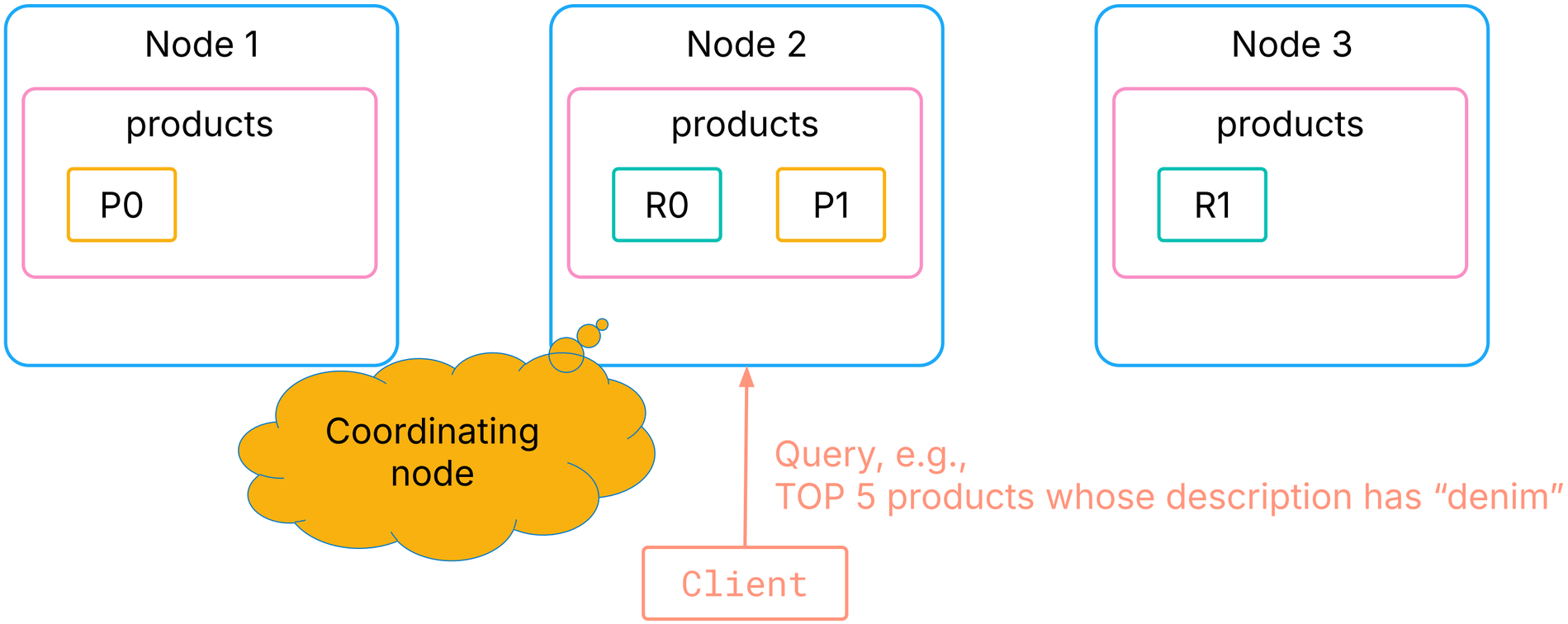


elastic

Distributed search

Two phase approach

- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)

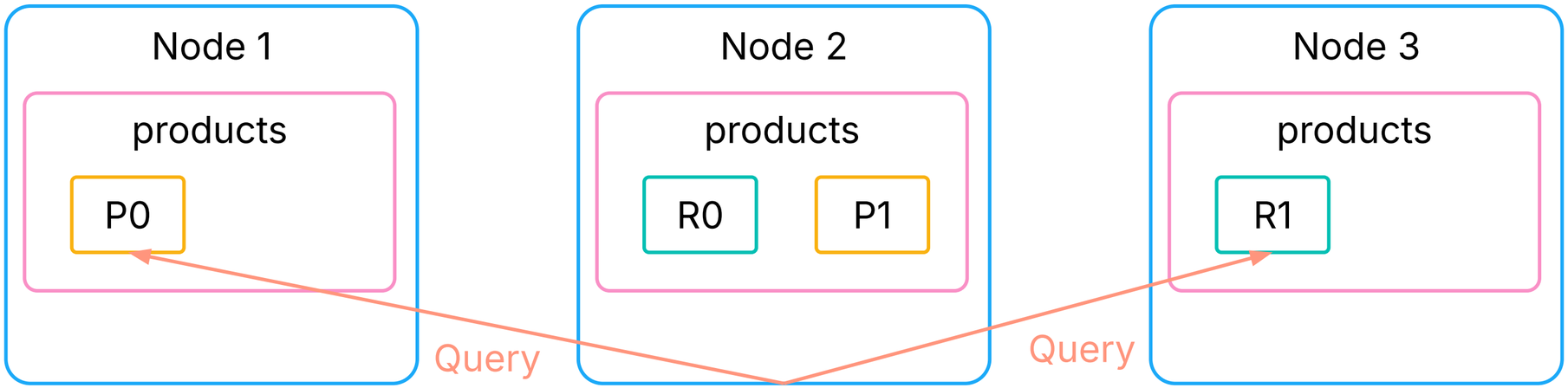


elastic

Distributed search

Two phase approach

- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)

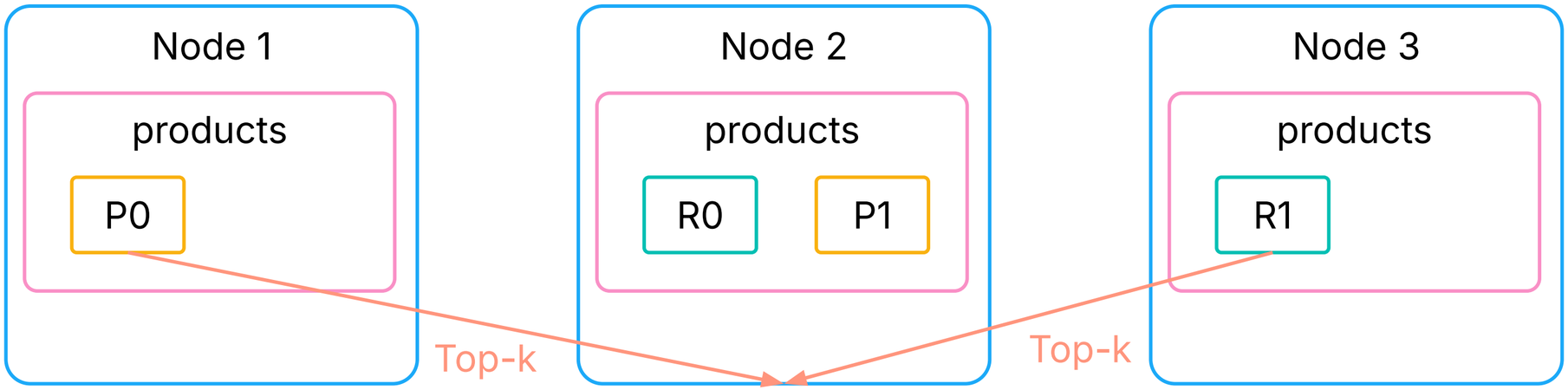


elastic

Distributed search

Two phase approach

- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)

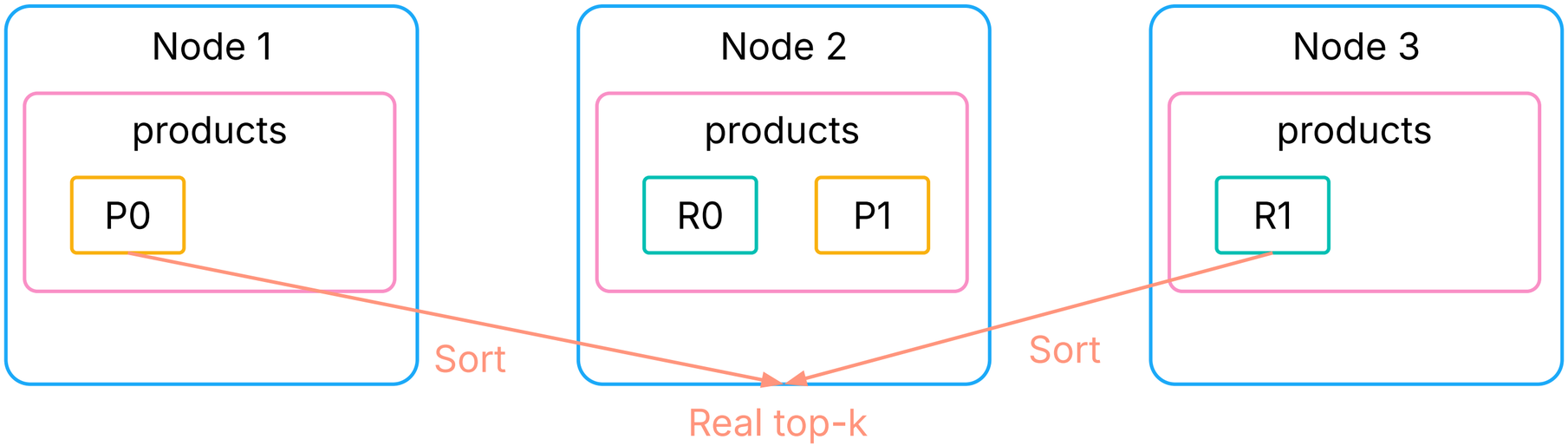


elastic

Distributed search

Two phase approach

- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)

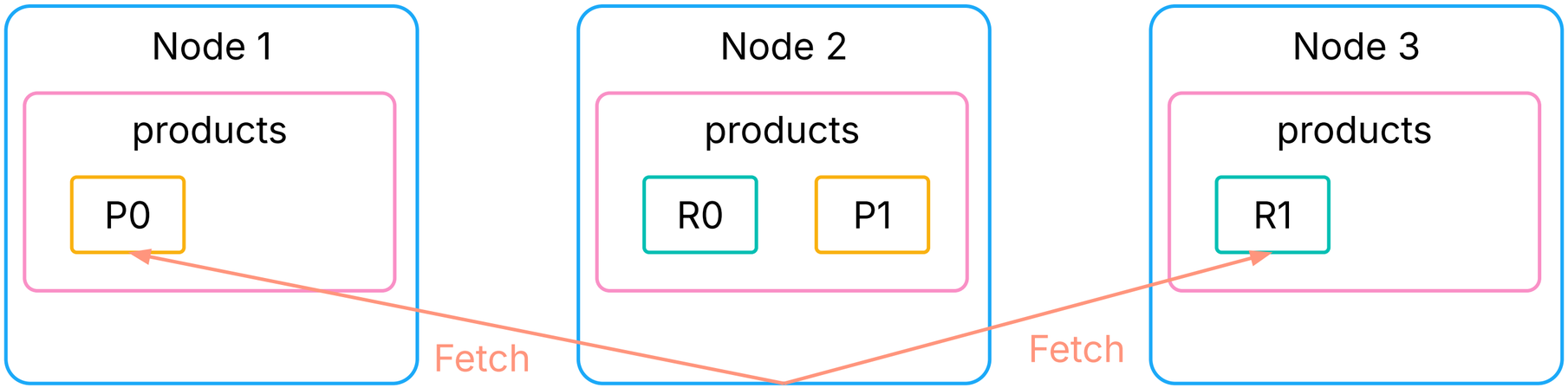


elastic

Distributed search

Two phase approach

- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)

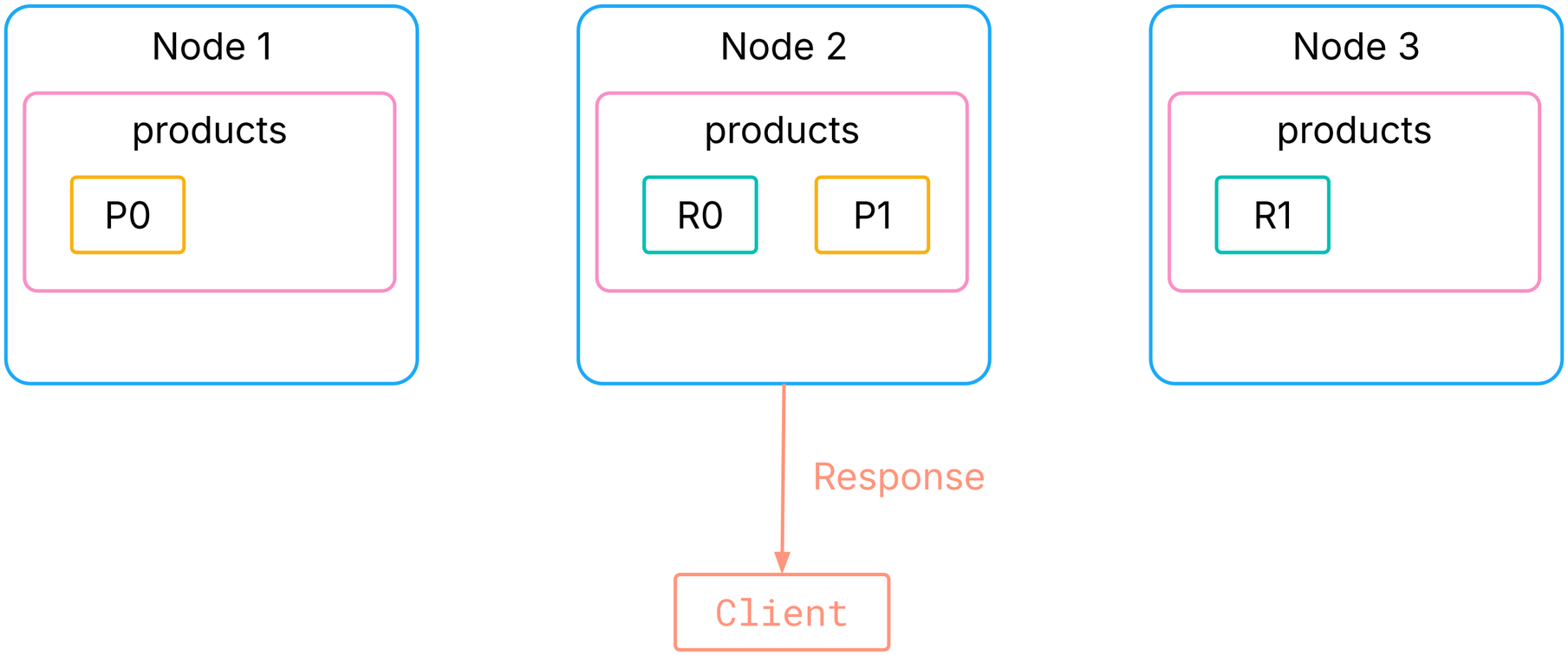


elastic

Distributed search

Two phase approach

- Query all shards, collect top-k hits, sort (on score) all results on coordinating node
- Create real top-k result set and fetch data (top-k instead of shards * top-k)

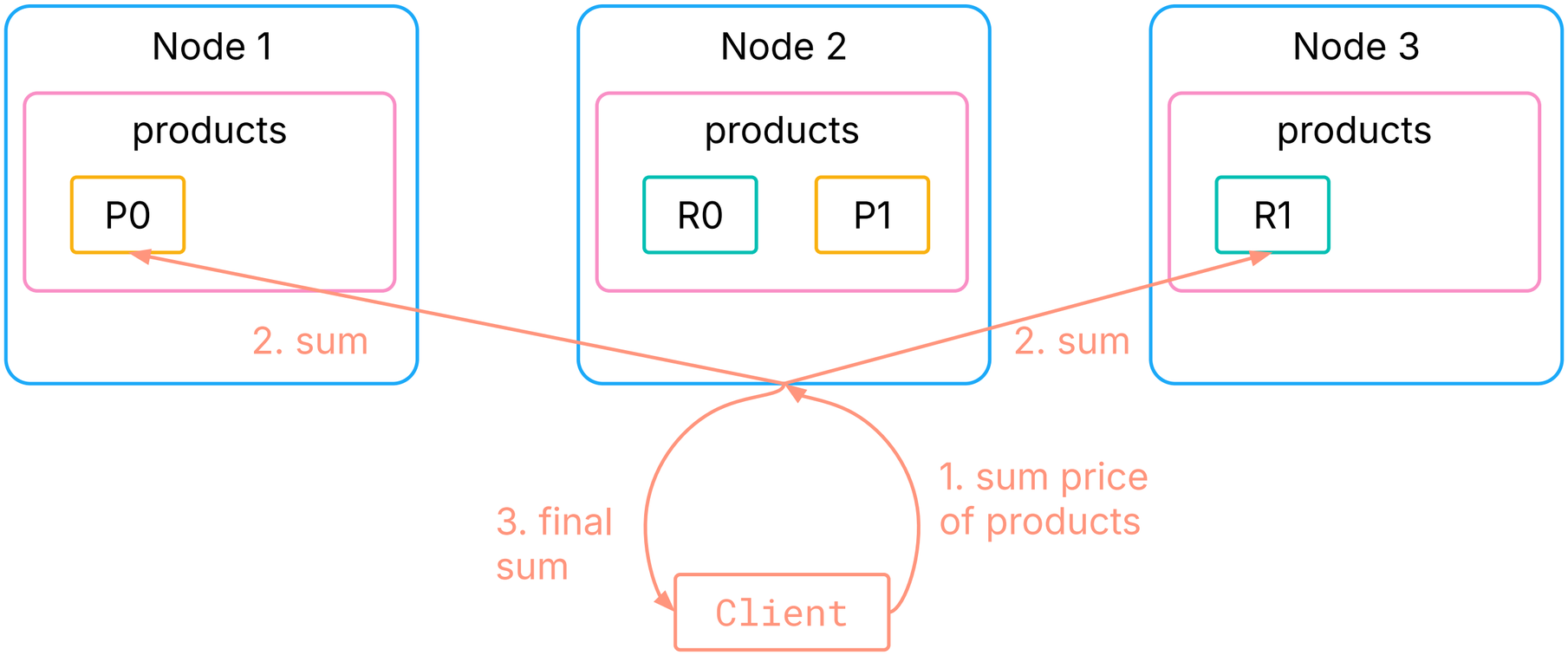


elastic

Distributed aggregations

Slice, dice and combine data to get insights

- Can be run on top of a result set of a query
- Some aggregations require your data to be central, e.g., cardinality → efficient estimations

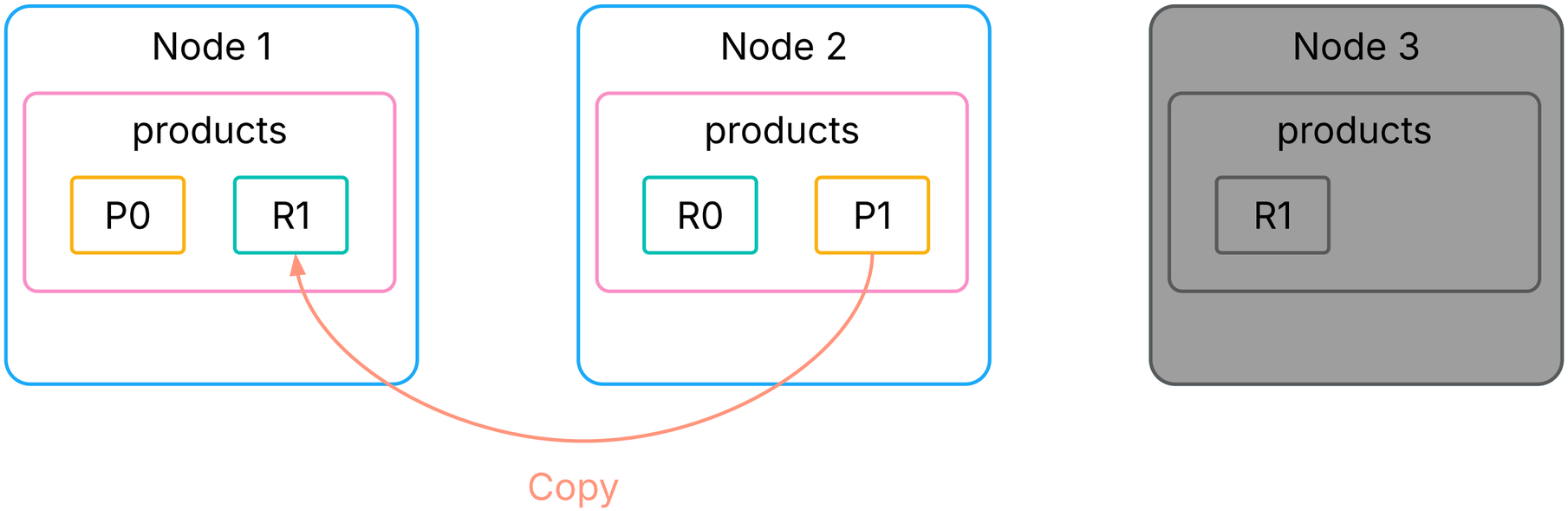


elastic

Shard recovery

What happens to shards when a node fails?

- If primary is lost → a replica is promoted to primary
- If replica is lost → it is copied from the primary to another node



elastic

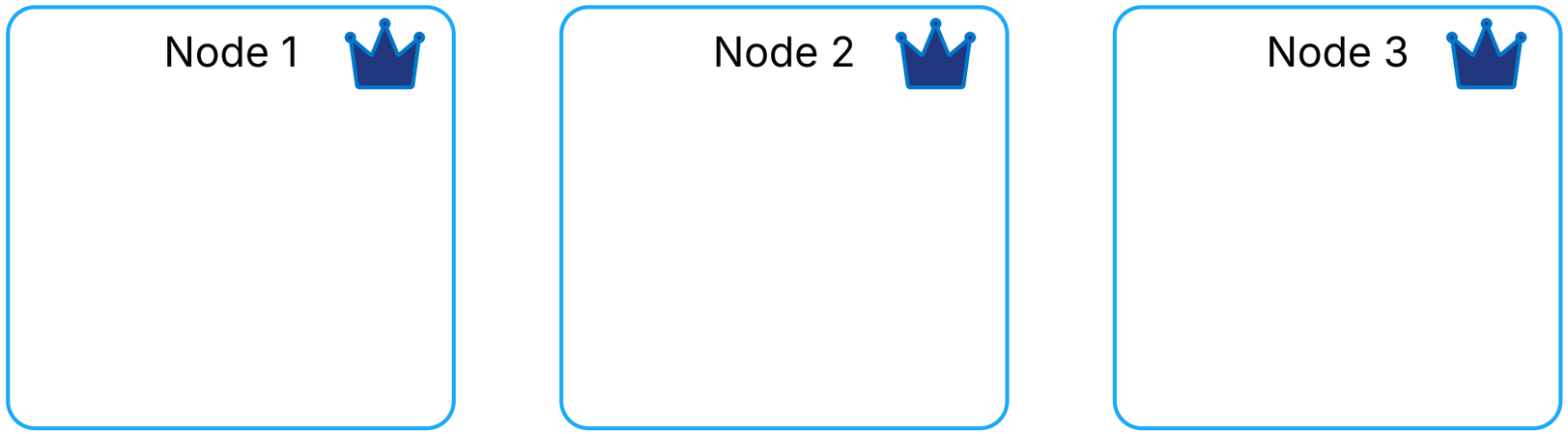
Cluster State

Cluster state

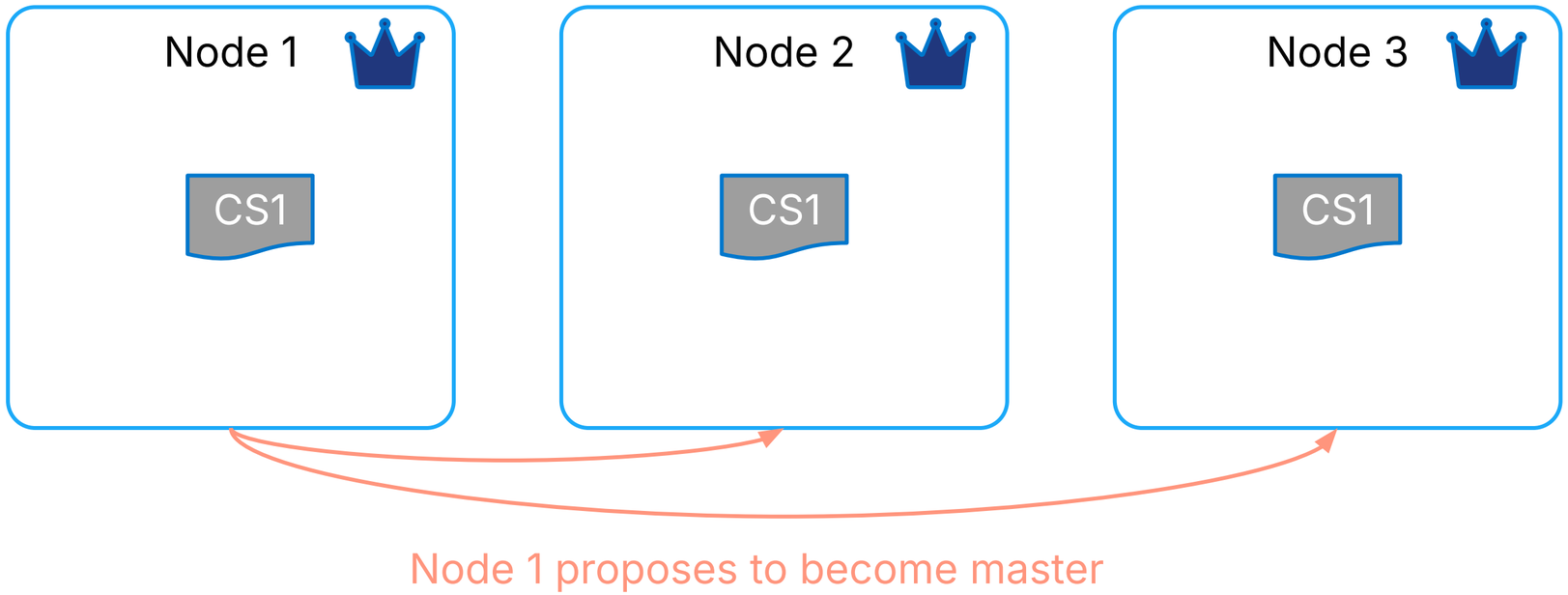
Contains the core information on the cluster membership and the indices

- One elected master node
  - Proposes updates of the cluster state based on events (e.g., node leaves, index created) and distributes it to all nodes
  - Decides data placement, where shards are moved and replicated
  - Node health checks
  - Not needed for reading/writing
- Consensus is used across a small set of master-eligible nodes
  - To establish the cluster state update proposals (quorum required)
  - To re-elect a master node on failure

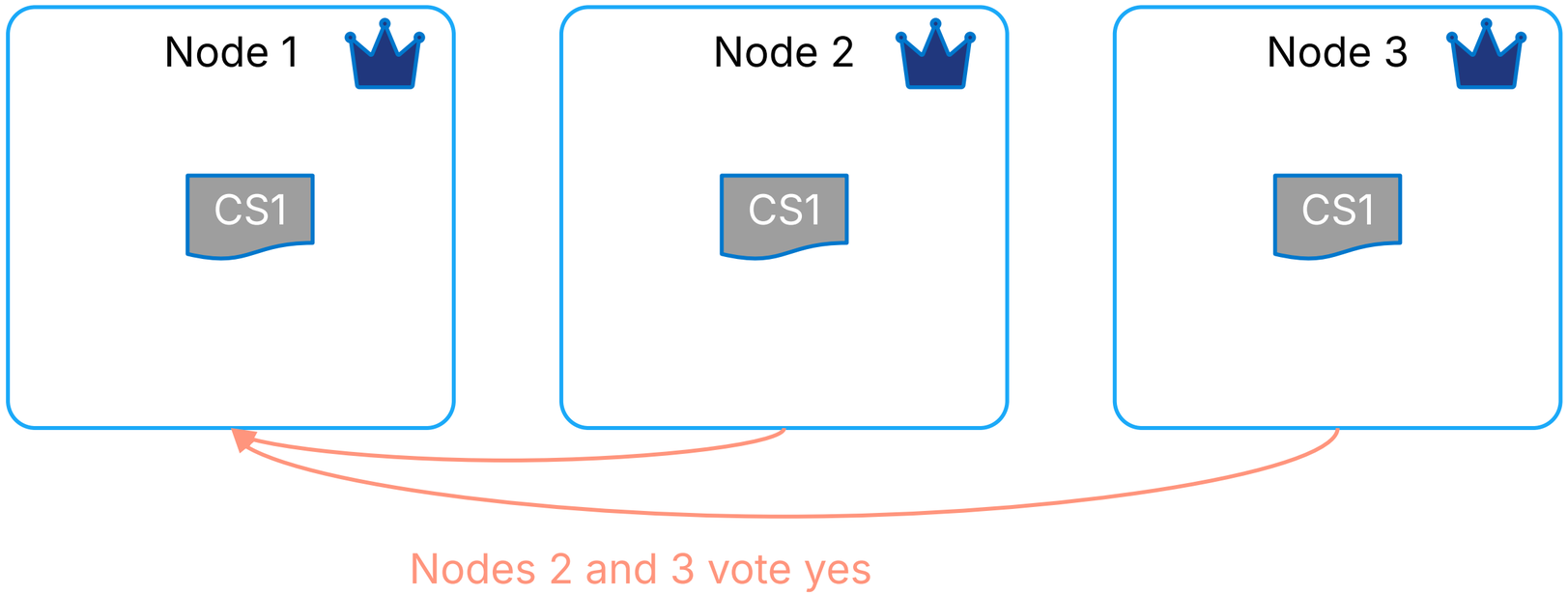
Cluster state – bootstrapping



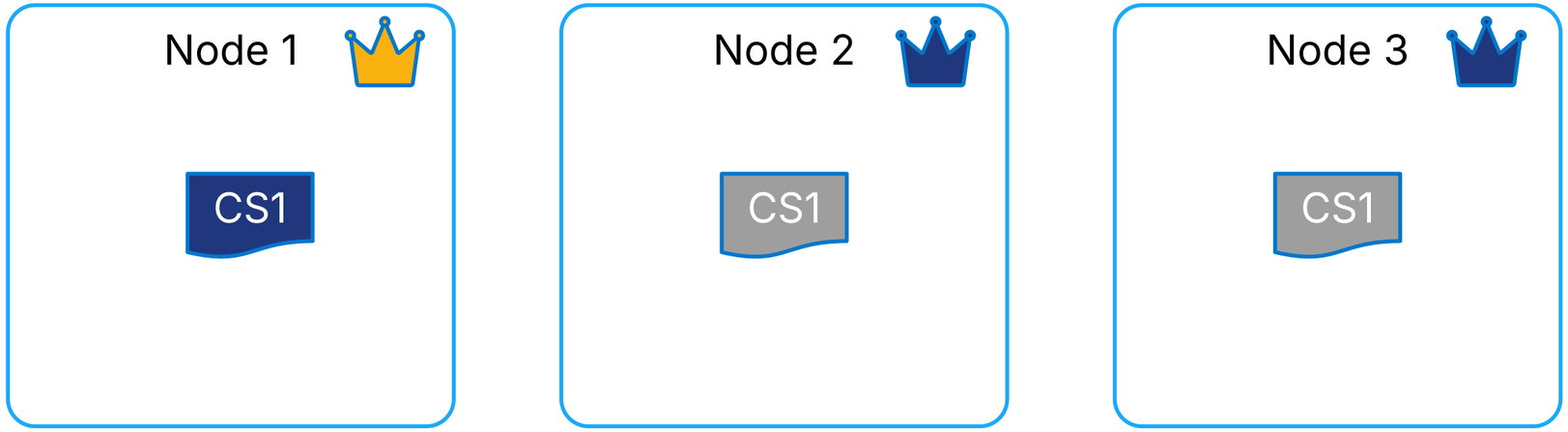
Cluster state – election



Cluster state – election

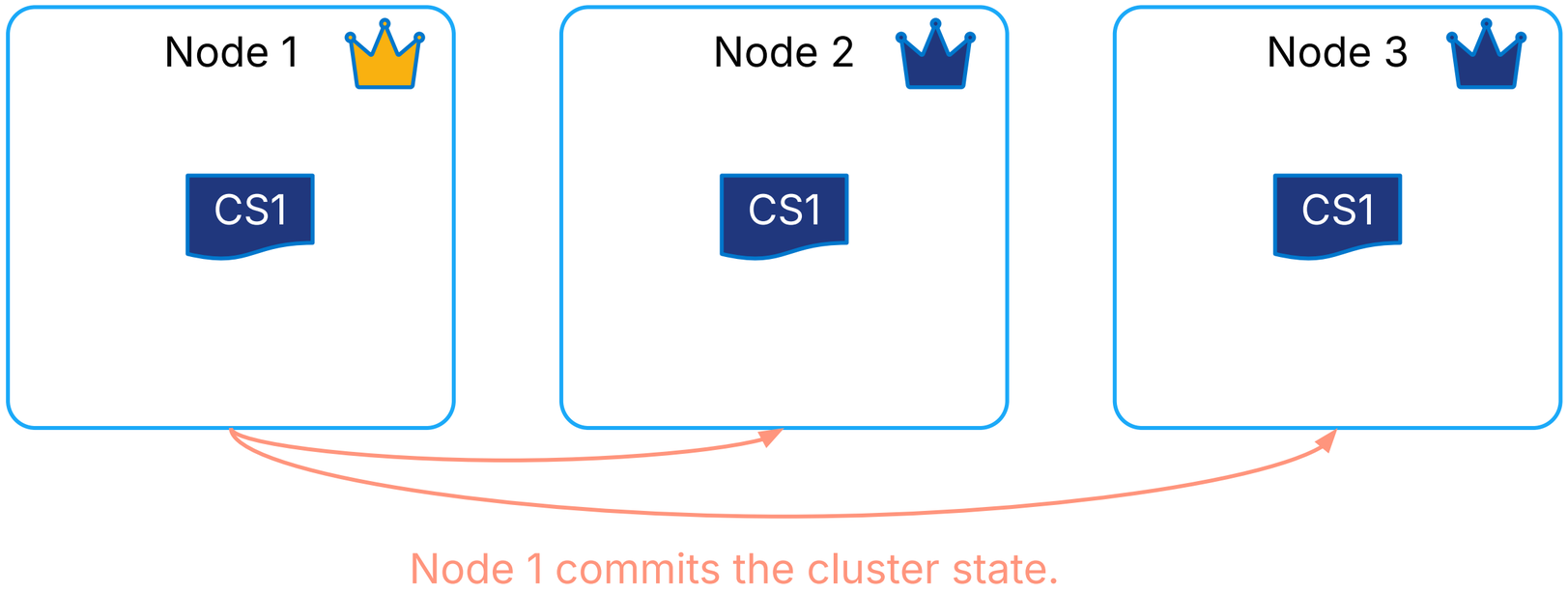


Cluster state – election

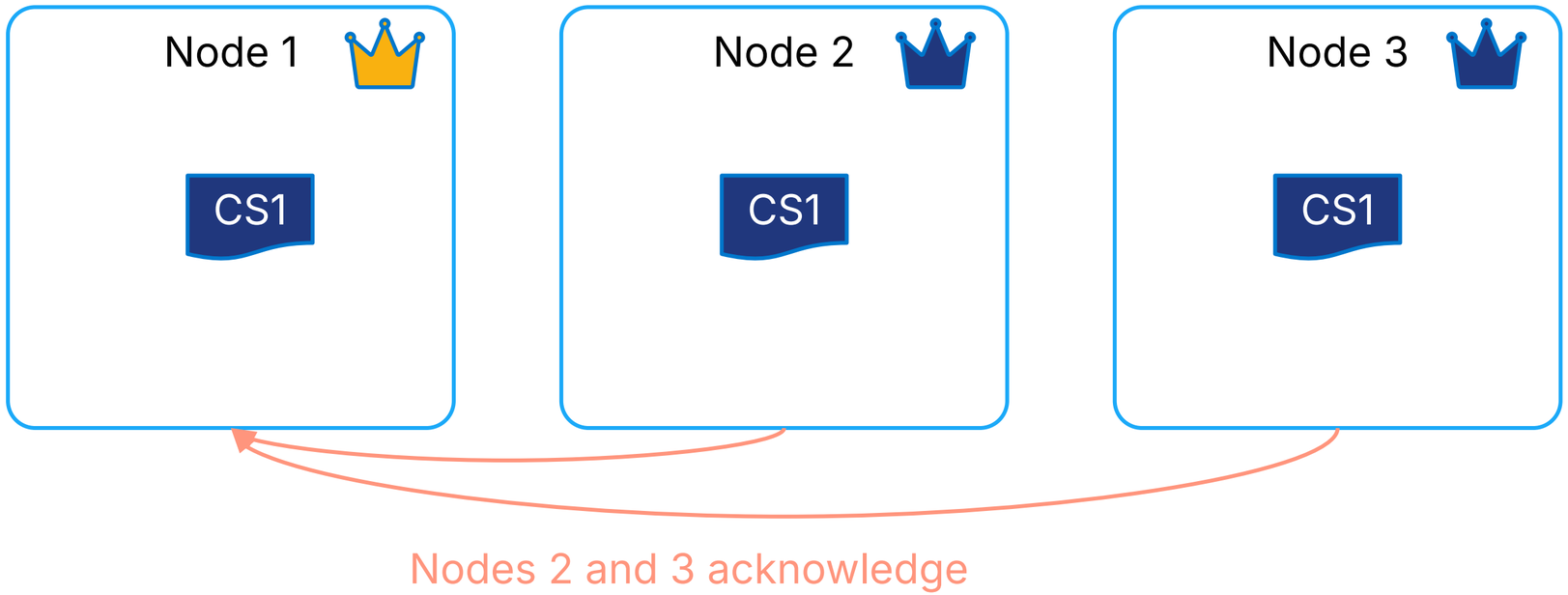


Node 1 becomes master

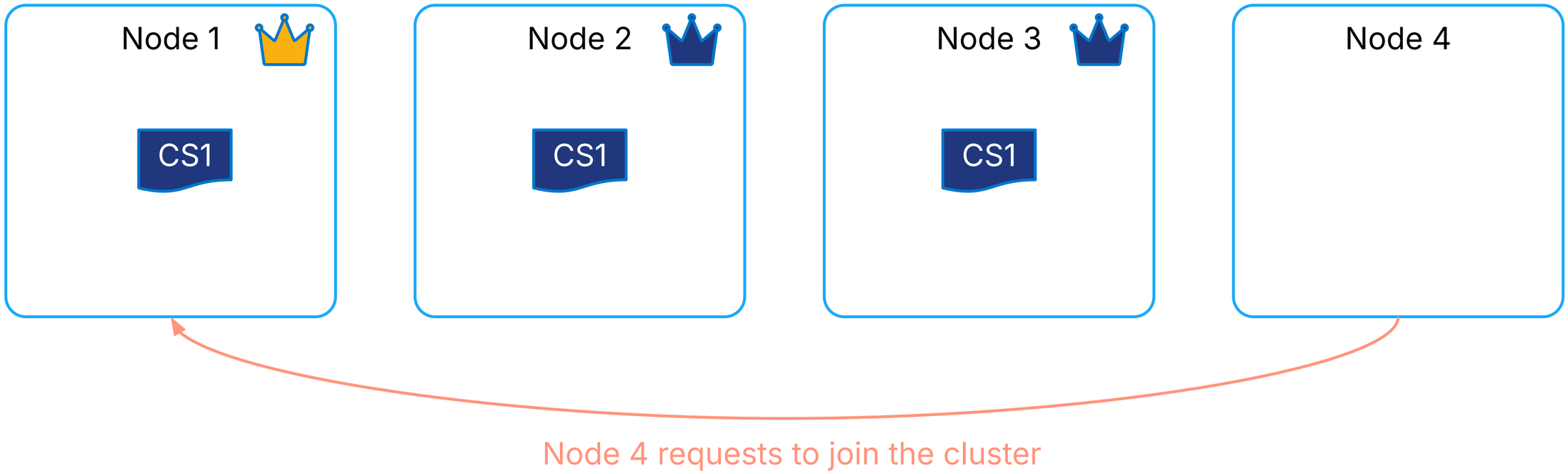
Cluster state – election



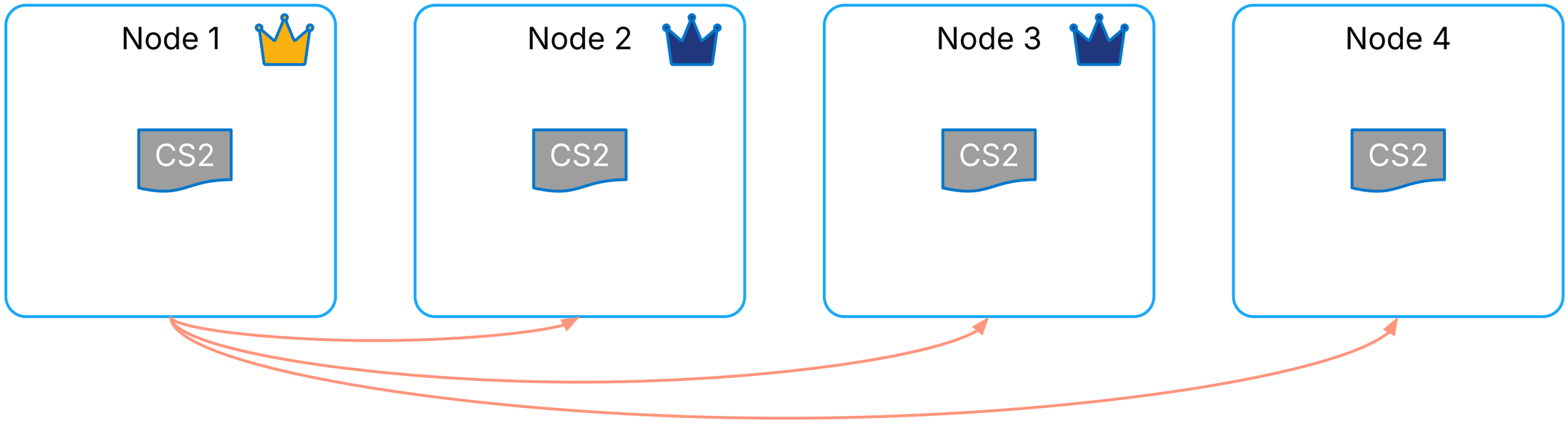
Cluster state – election



Cluster state – node joins

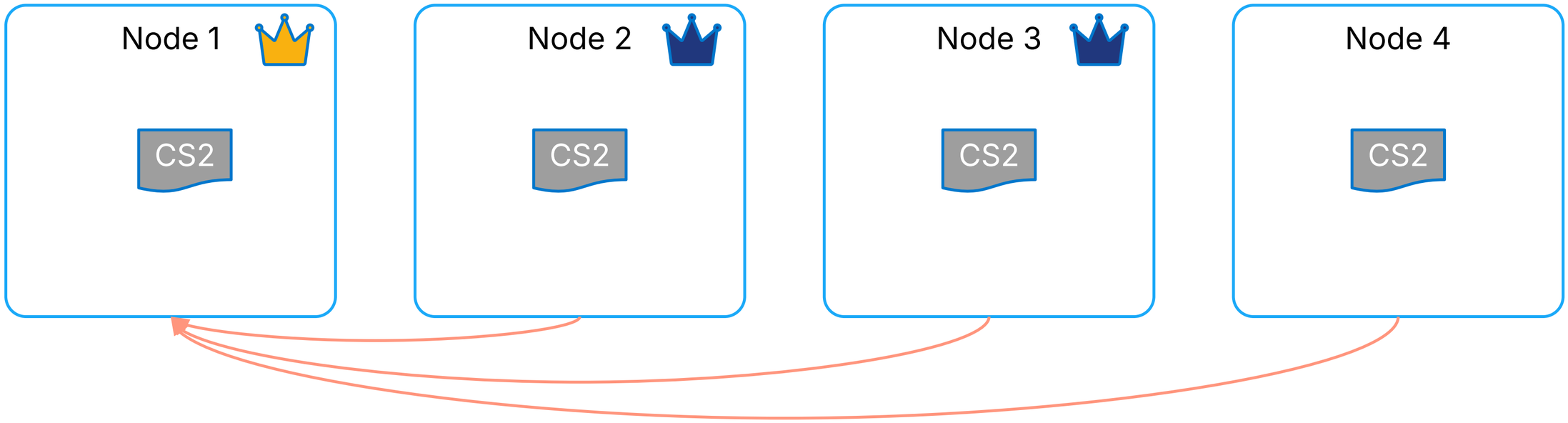


Cluster state – node joins



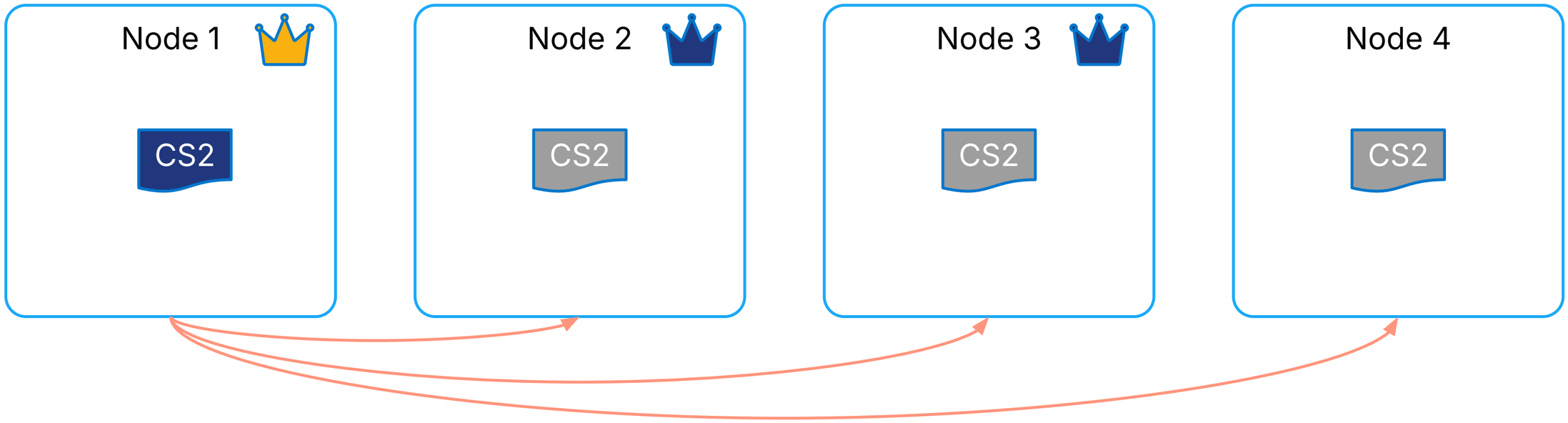
elastic

Cluster state – node joins

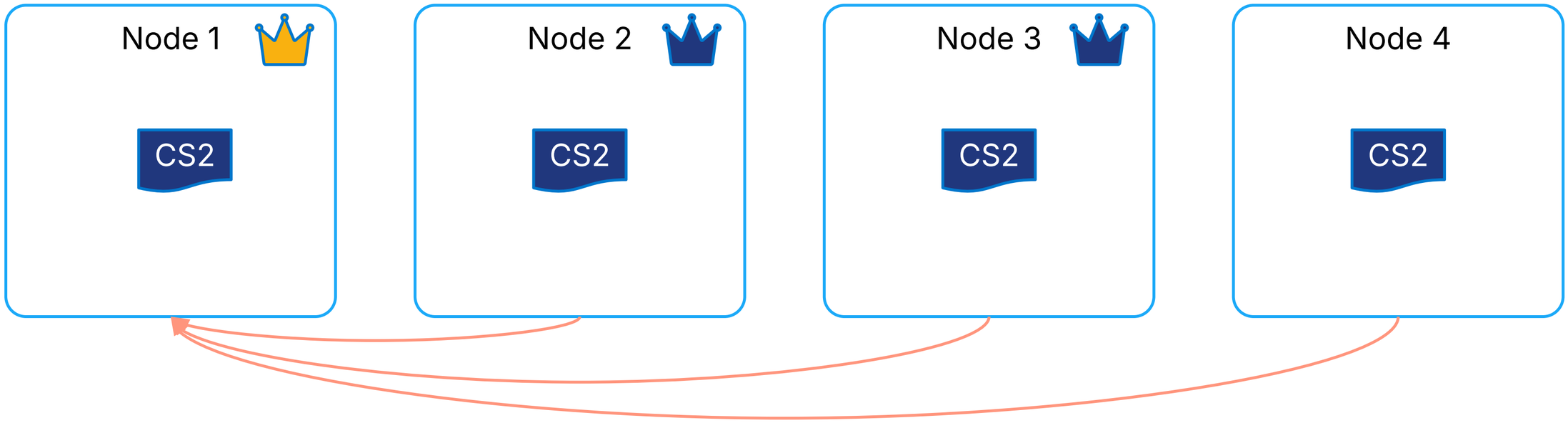


elastic

Cluster state – node joins

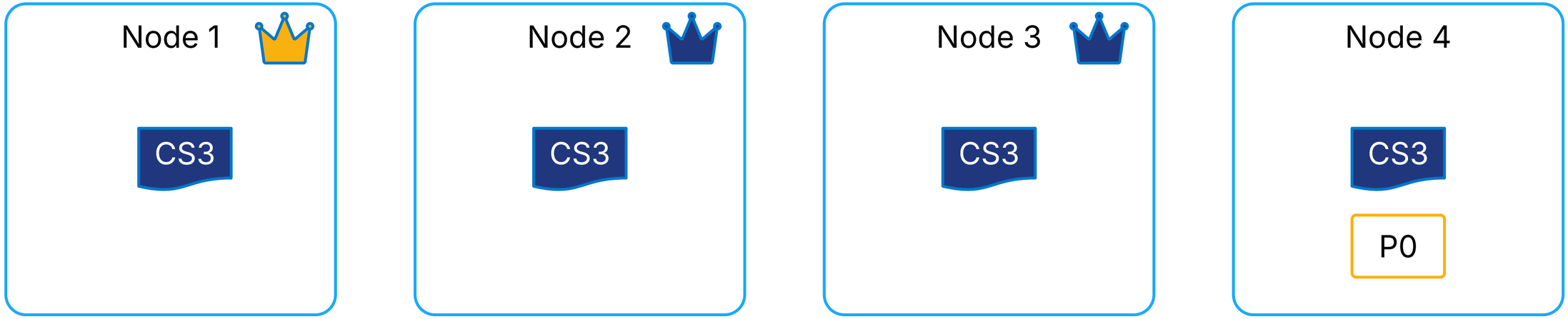


Cluster state – node joins



elastic

Cluster state – index created (fast forwarded)



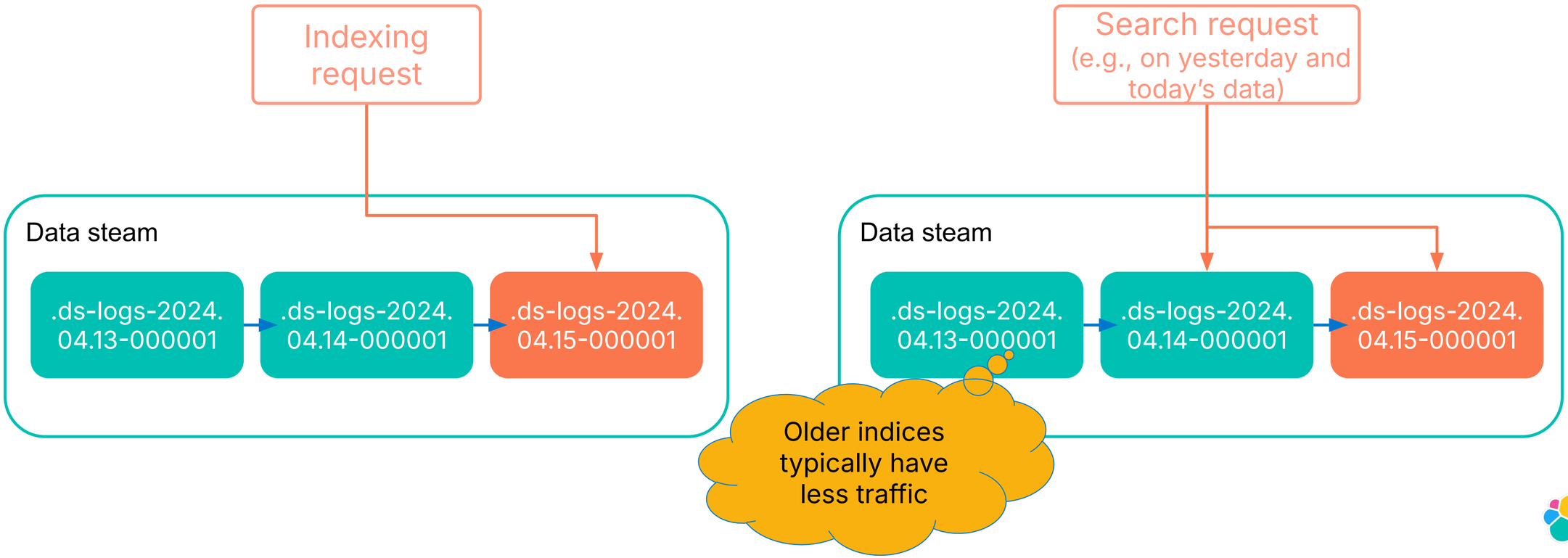
elastic

Scaling with Data Streams and ILM

Scaling with data streams

Store append-only time series data across multiple indices

- Gives a single named resource for requests
- Rollover indices based on age and/or size

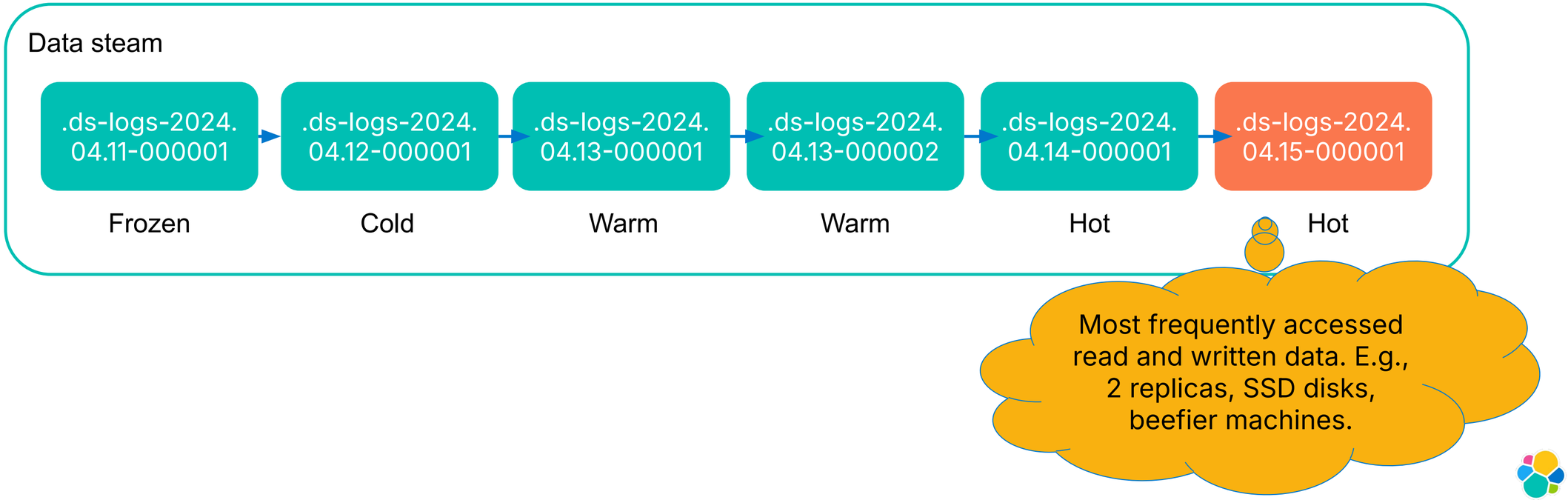


elastic

Scaling with Index Lifecycle Management

Tiers of data nodes with different cost/performance characteristics

- Automatic rollover of data streams through the data tiers via a lifecycle policy

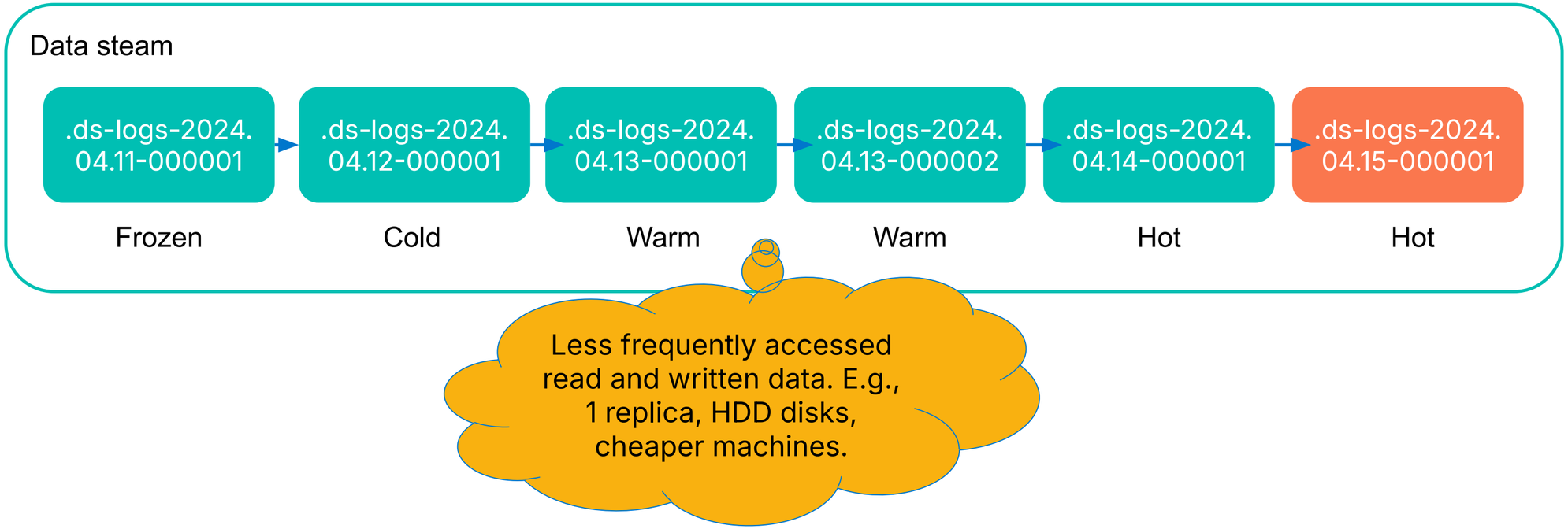


elastic

Scaling with Index Lifecycle Management

Tiers of data nodes with different cost/performance characteristics

- Automatic rollover of data streams through the data tiers via a lifecycle policy

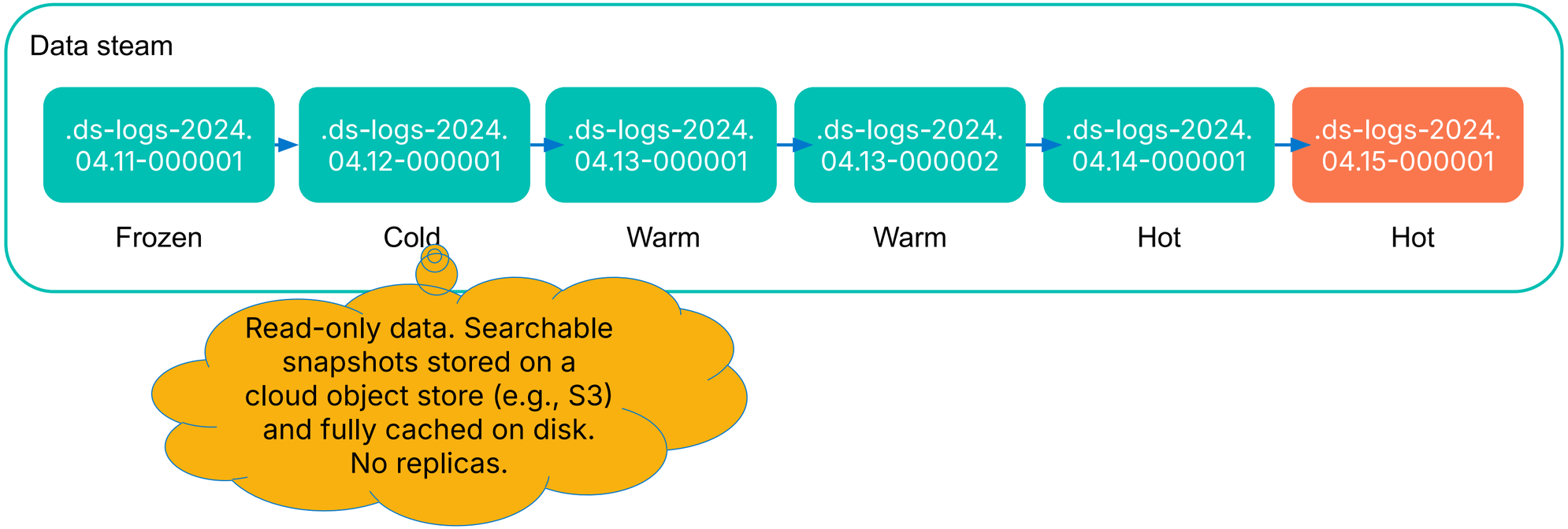


elastic

Scaling with Index Lifecycle Management

Tiers of data nodes with different cost/performance characteristics

- Automatic rollover of data streams through the data tiers via a lifecycle policy

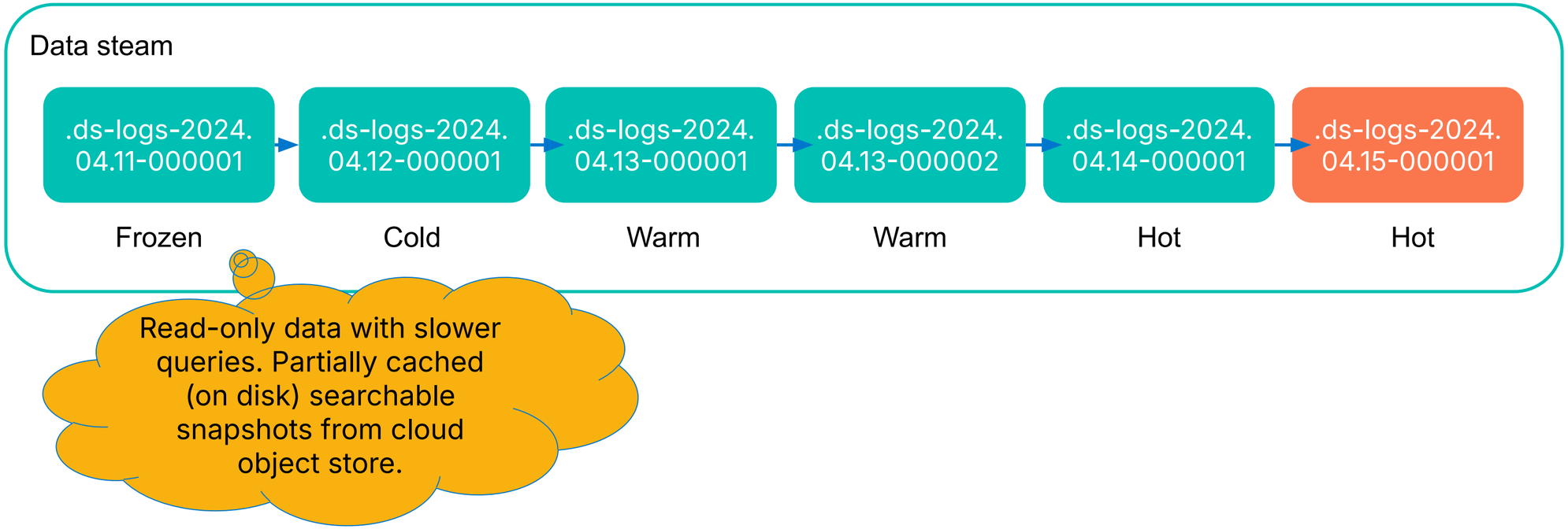


elastic

Scaling with Index Lifecycle Management

Tiers of data nodes with different cost/performance characteristics

- Automatic rollover of data streams through the data tiers via a lifecycle policy



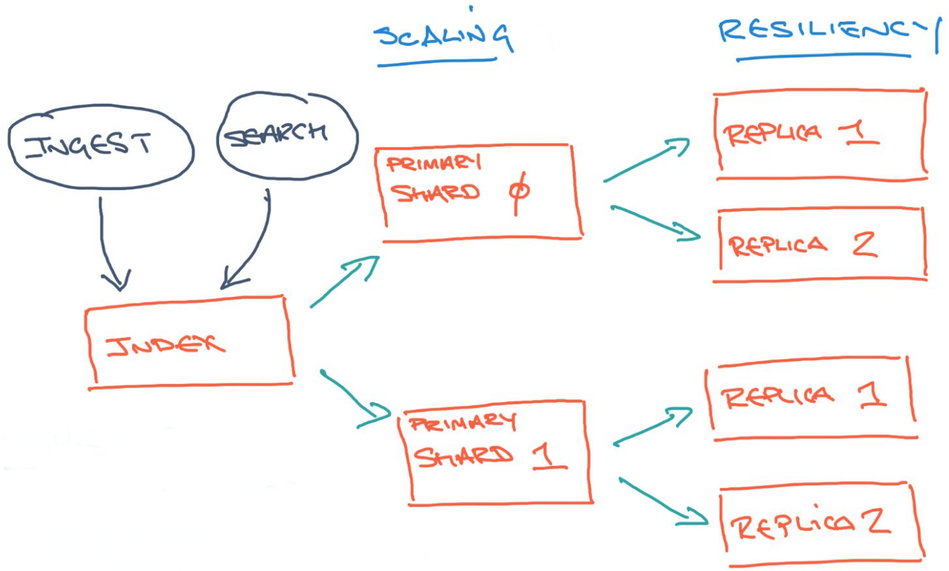
elastic

Vision: Scaling with Stateless

Summary of challenges so far

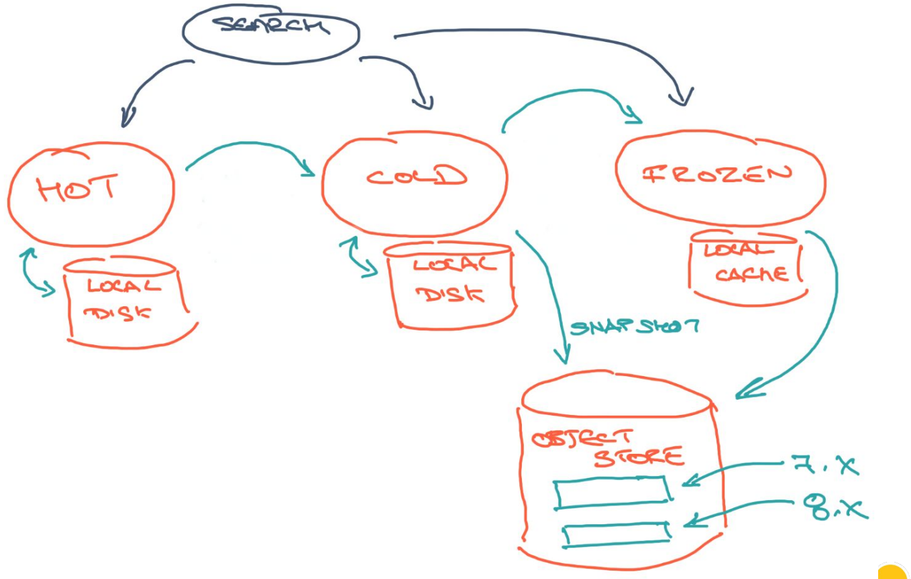
Data Management

- Cost of Ingestion (CPU, storage)
- Multiple redundant data copies
- Shared CPU for Ingest and Search (leads to overprovisioning)



ILM tiers

- Complex to understand cost & performance trade offs
- Lots of data movement between tiers
- Propagation delay between tiers



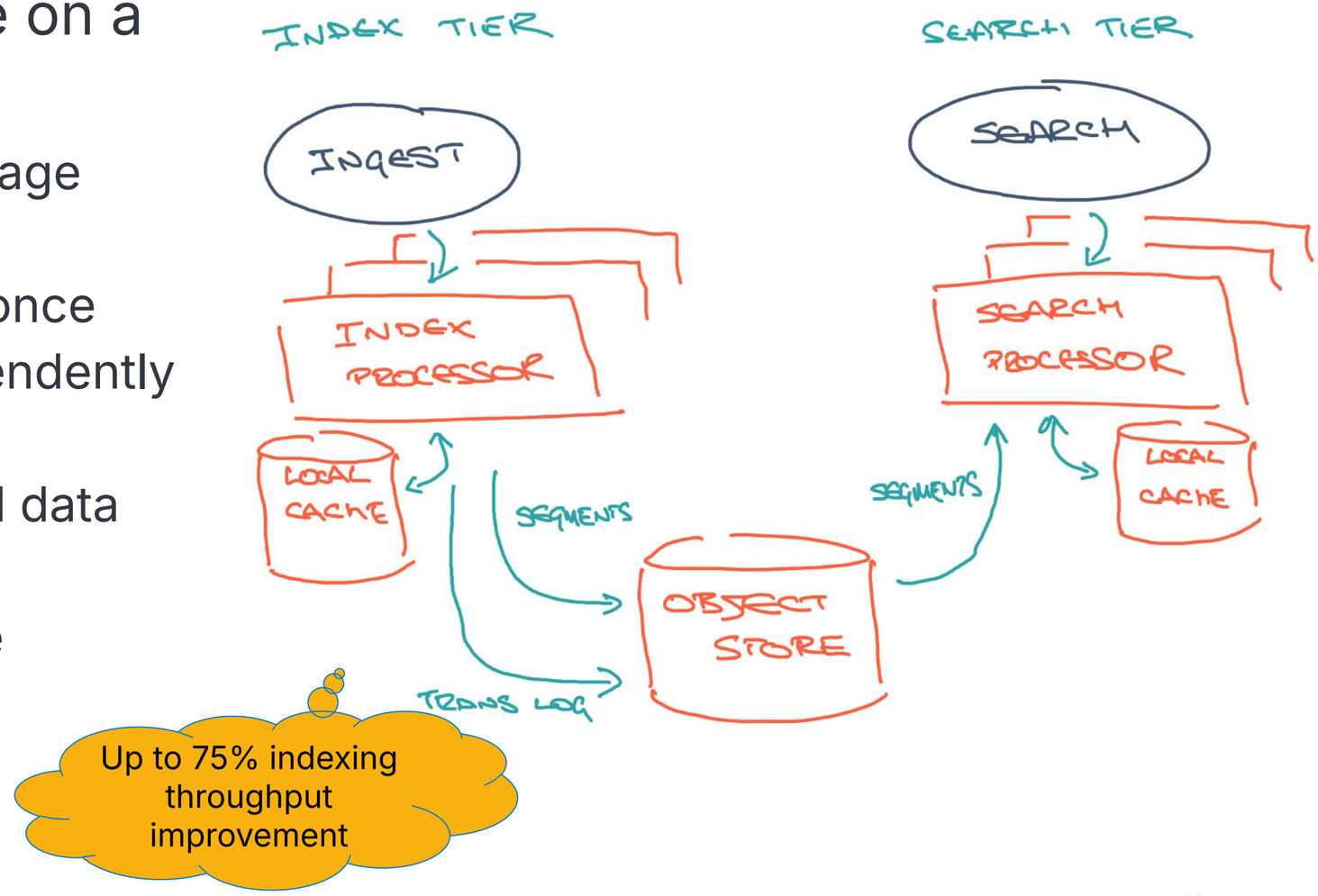
Credits: "The Journey to Serverless Elasticsearch", ElasticON Amsterdam 2022

elastic

Vision: Stateless

Store indices or cluster state on a cloud object store

- Separate Compute from Storage
- Separate Ingest from Search
  - Incur cost of Ingestion once
- Size & Scale each tier independently
- Object store for resilience
  - Elimination of replicated data
- Limitless Storage
- Local cache for performance
- Simpler administration



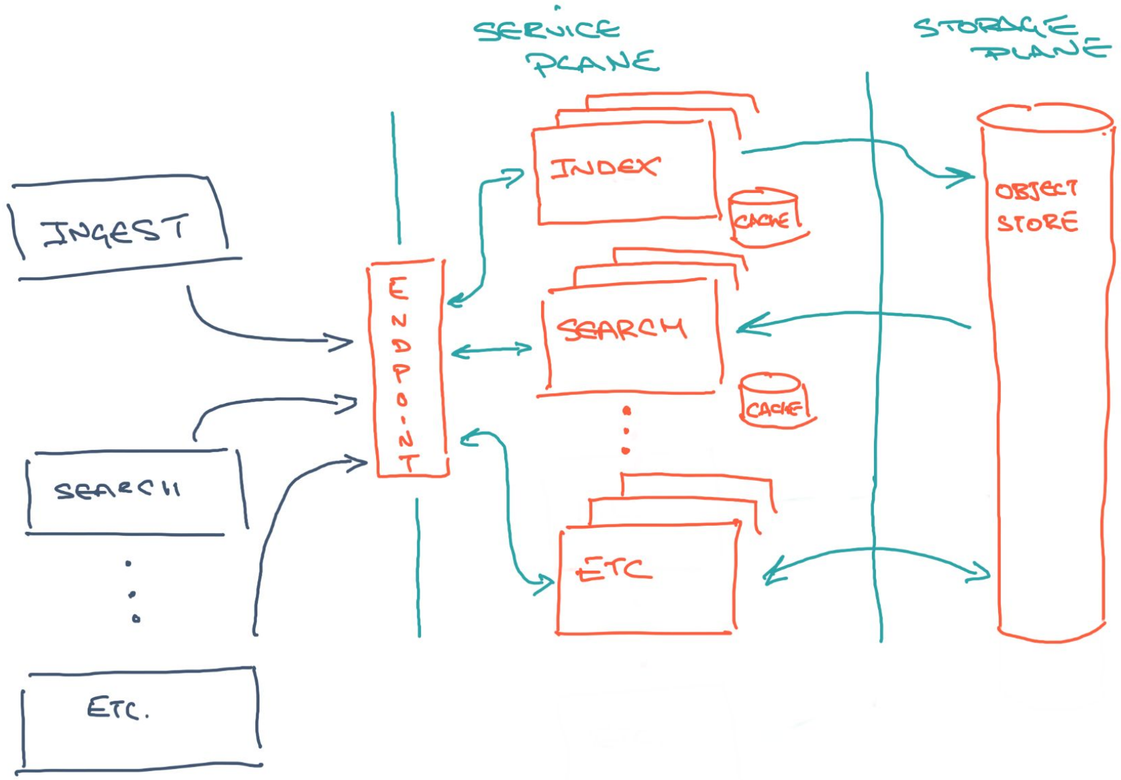
Credits: https://www.elastic.co/blog/stateless-your-new-state-of-find-with-elasticsearch

elastic

Vision: Serverless

Fully managed

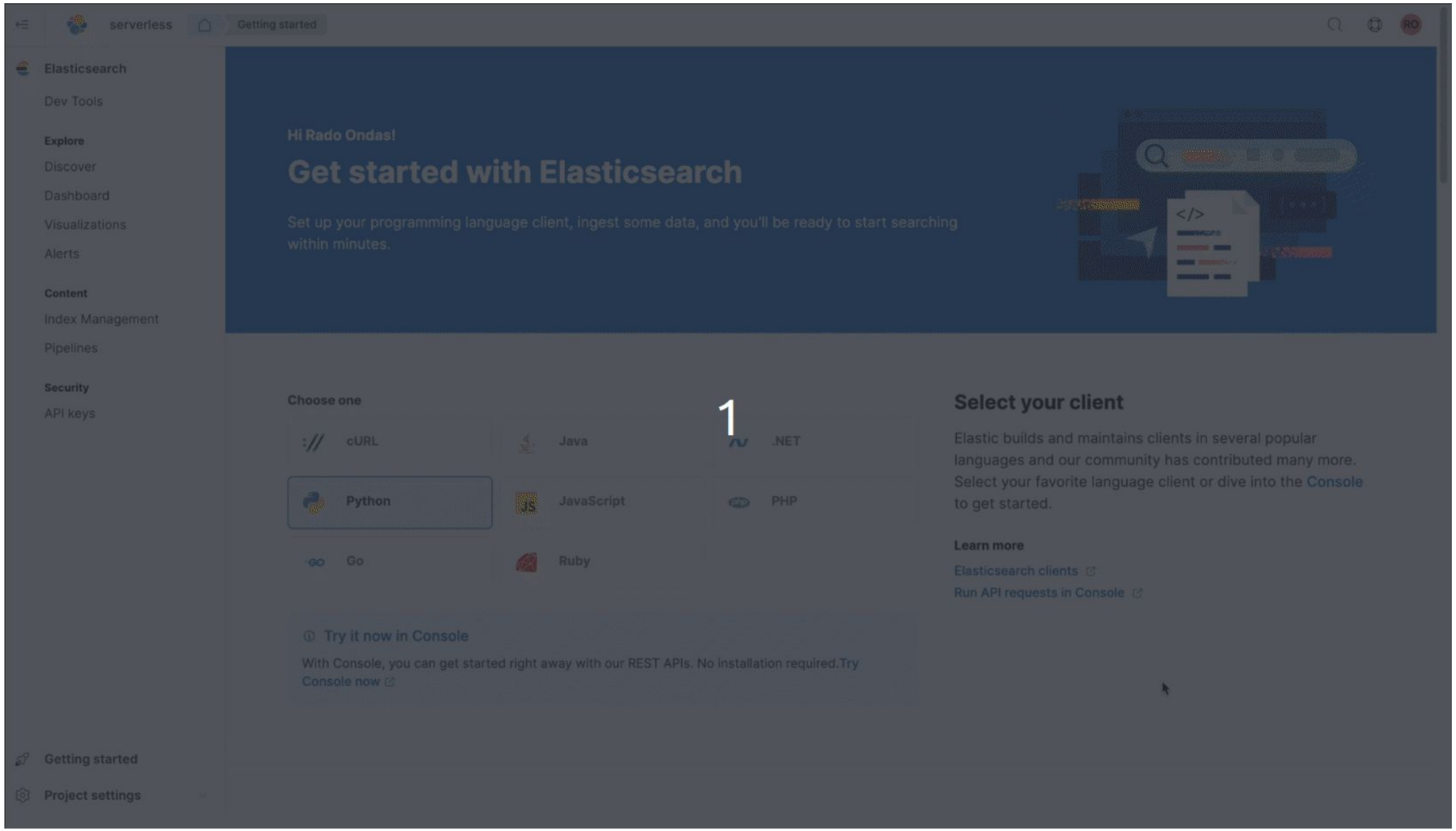
- Use solutions' APIs directly, it just works
- Elasticity of Compute & Storage
- Versionless
- Autoscaling - with back to zero
- Autohealing
- K8 control plane
- Multi-tenancy



Credits: "The Journey to Serverless Elasticsearch", ElasticON Amsterdam 2022

elastic

Elasticsearch serverless experience sneak peek



Credits: https://www.elastic.co/blog/stateless-your-new-state-of-find-with-elasticsearch

Summary

- Elasticsearch is an unparalleled scalable search & analytics engine:
  - Scaling Lucene instances with multiple shards
  - Distributed searches and aggregations over multiple nodes
  - Master node directs the cluster state updates, via an efficient consensus protocol
  - Time based data streams are scaled with Index Lifecycle Management
  - Future: serverless vision based on keeping state on cloud object store
- Examples of scalability
  - Adobe (2018): 400 VMs, 10B docs, 600q/sec, 6000docs/sec
  - Elastic Internal Observability Clusters (2022): 207 clusters (through Cross-Cluster Search), 1.2 trillion docs, 300TB events/day, 4 cloud providers (53 regions)
  - Serve more with Serverless, Elastic Stack & Cloud: Start from AWS in 3 clicks, learn about Elastic's serverless vision, Benchmark-driven optimizations, A new era for cluster coordination in Elasticsearch, Autoscale your Elastic Cloud data and machine learning nodes, How many shards should I have in my Elasticsearch cluster?

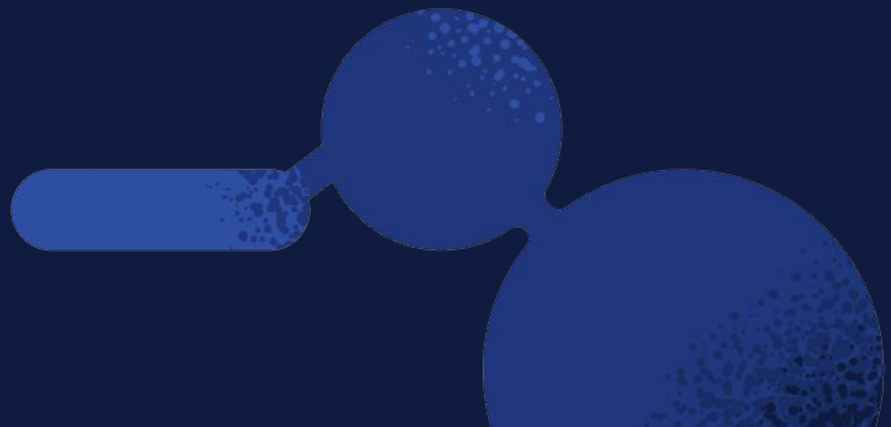
Community, culture and careers

- 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. E.g., AI-ready vector search with exact match and approximate kNN.
  - Elastic Search Labs → elastic.co/search-labs. E.g., Integrate with ChatGPT.
- Careers → elastic.co/about/careers
  - Elastic Source Code, remote, 2800+ employees across 40+ countries
- Subscribe for next meetup.com/greece-elastic events

elastic

Thank you!
Questions?

Iraklis Psaroudakis
www.kingherc.com



elastic