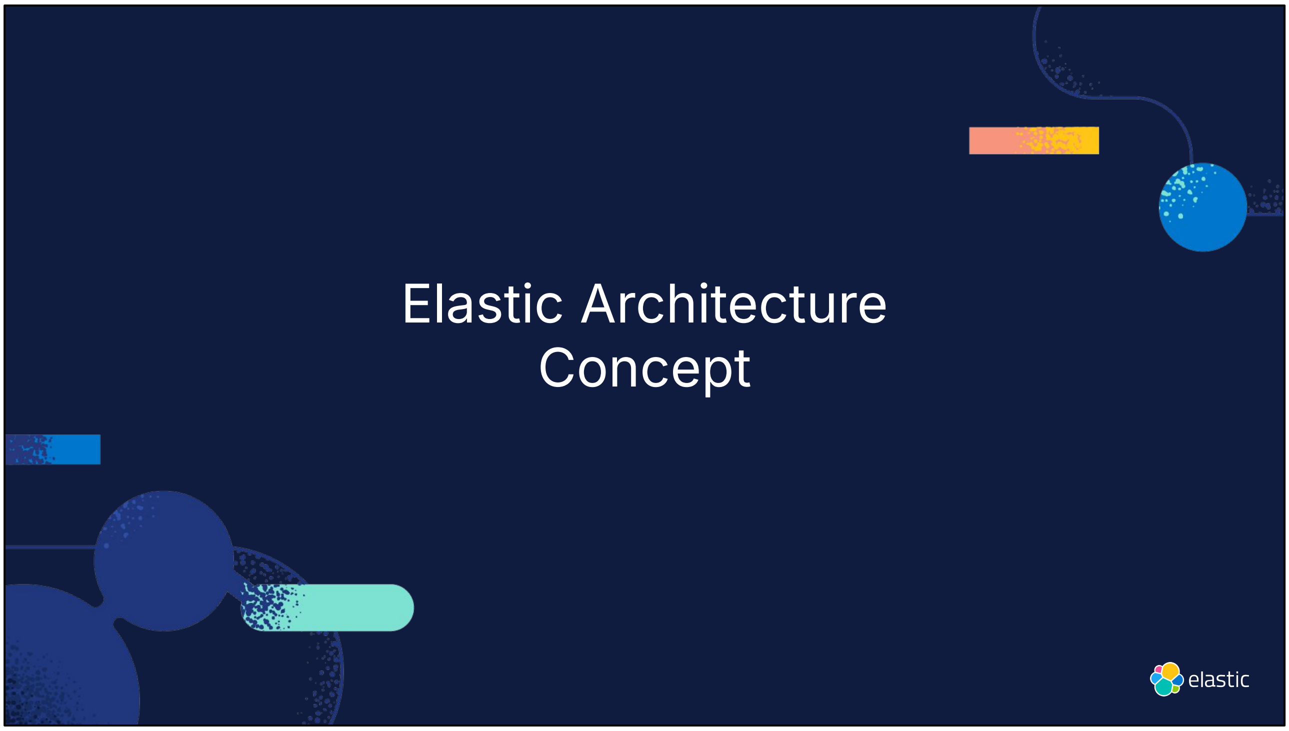
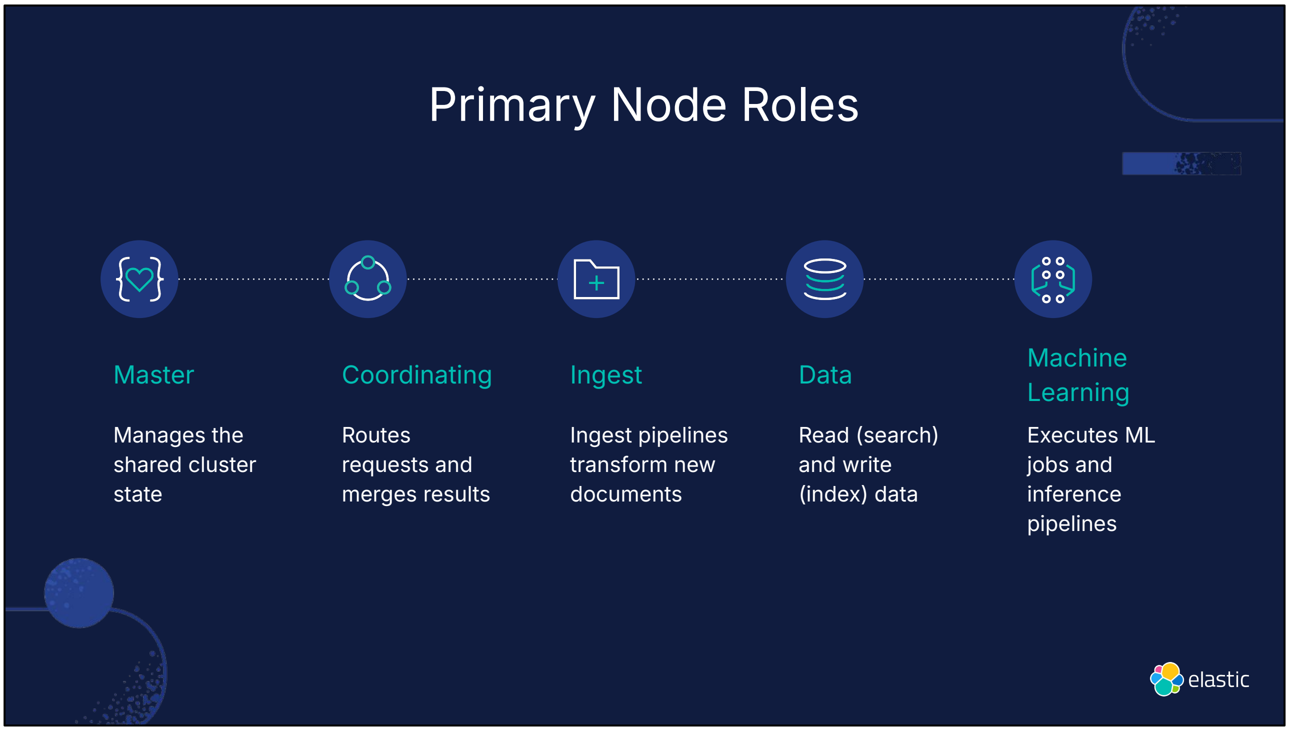


Elastic Architecture Concept

The Elasticsearch Cluster

Roles and definitions 📘

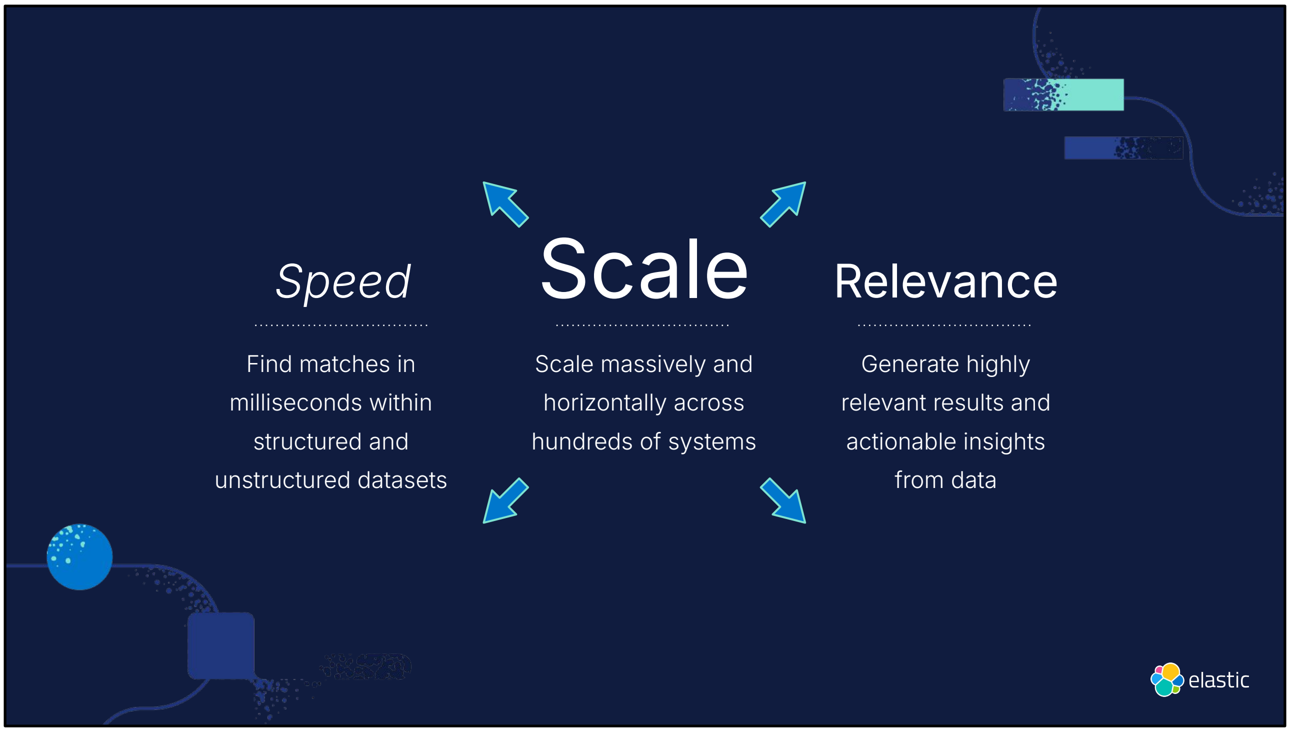
elastic



Primary Node Roles

Master	Manages the shared cluster state
Coordinating	Routes requests and merges results
Ingest	Ingest pipelines transform new documents
Data	Read (search) and write (index) data
Machine Learning	Executes ML jobs and inference pipelines

elastic



Speed

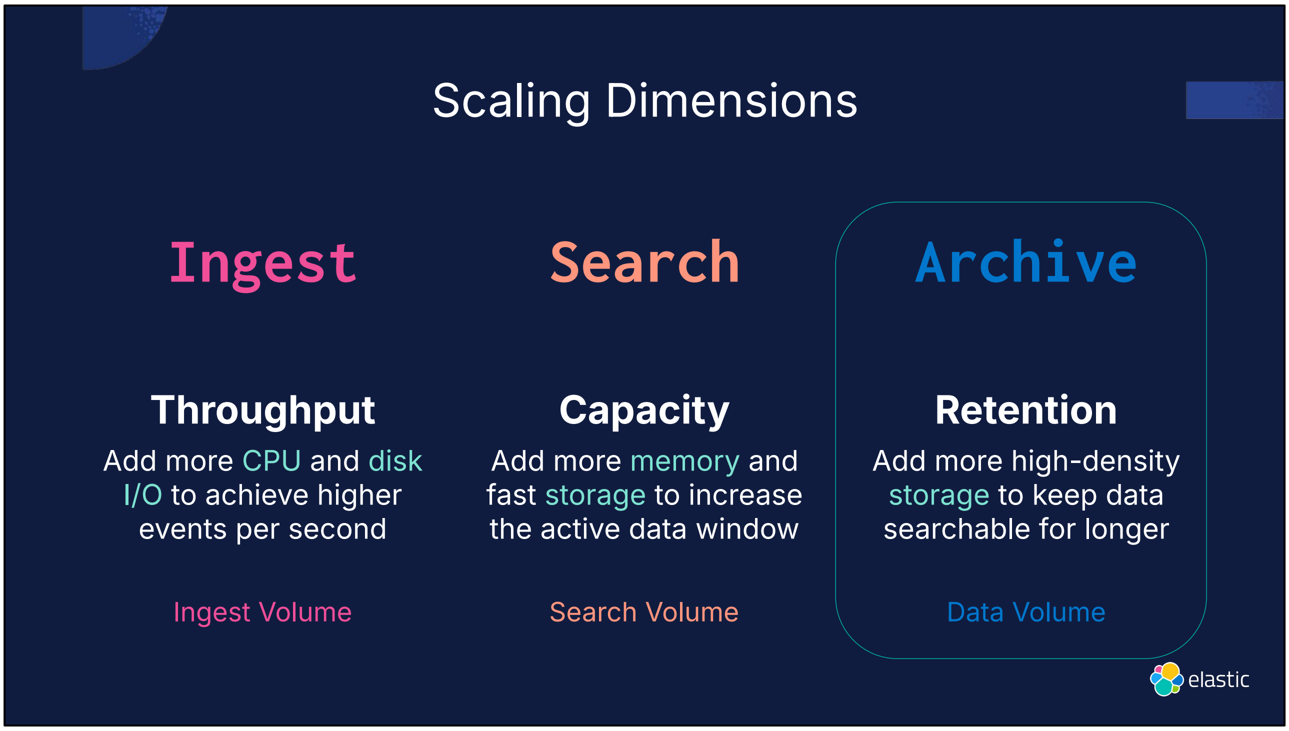
Find matches in milliseconds within structured and unstructured datasets

Scale

Scale massively and horizontally across hundreds of systems

Relevance

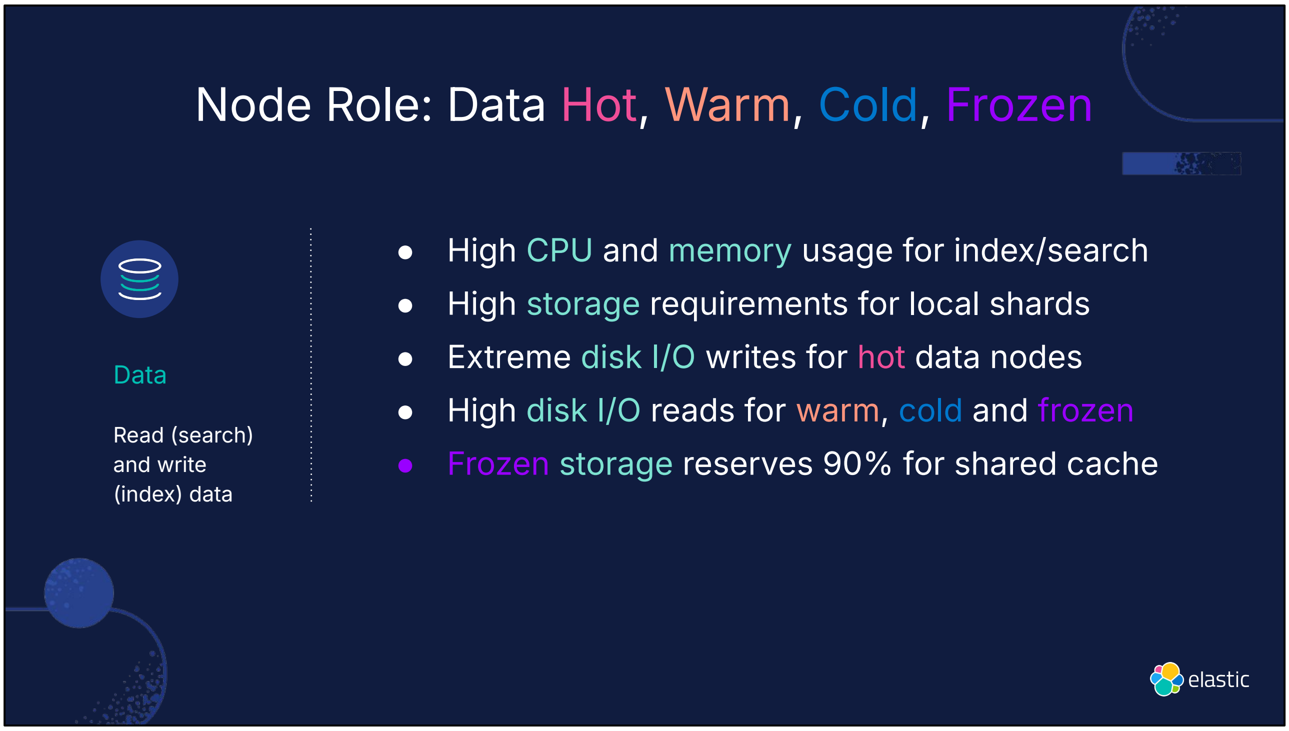
Generate highly relevant results and actionable insights from data



Scaling Dimensions

Ingest	Search	Archive
Throughput	**Capacity**	**Retention**
Add more CPU and disk I/O to achieve higher events per second	Add more memory and fast storage to increase the active data window	Add more high-density storage to keep data searchable for longer
Ingest Volume	Search Volume	Data Volume

elastic



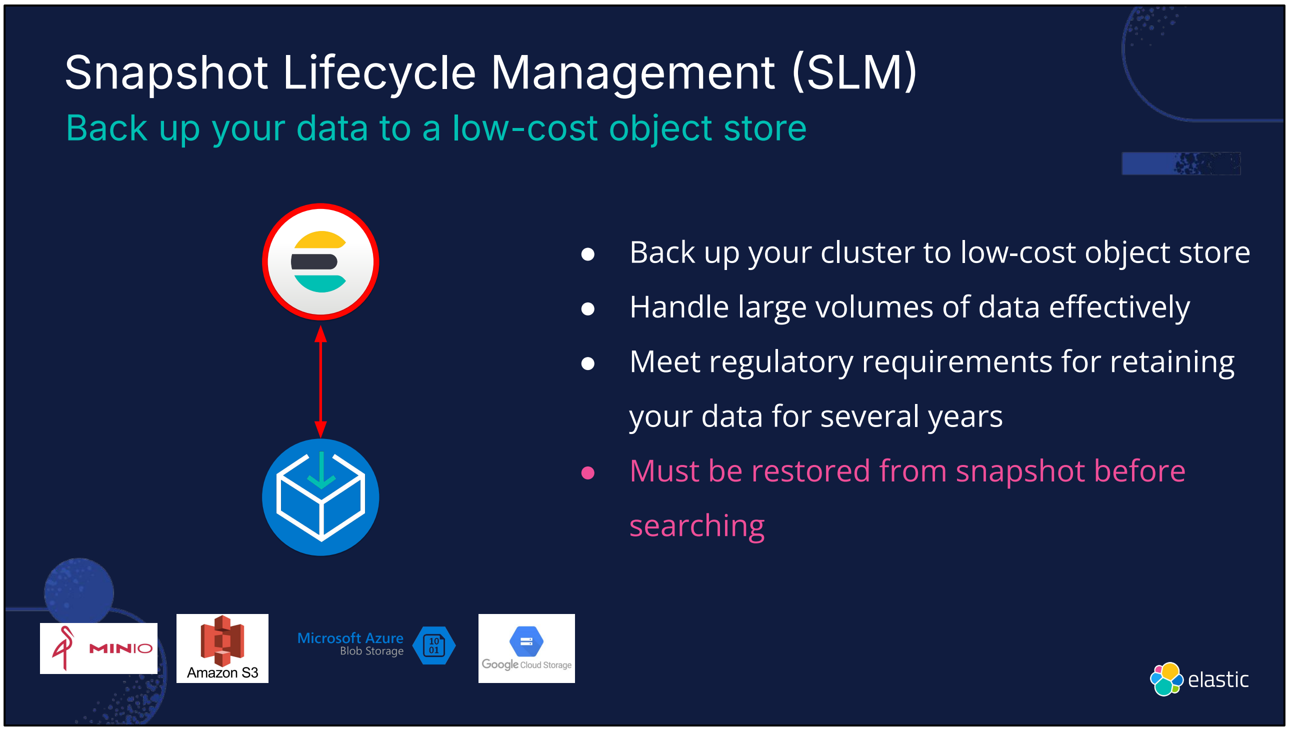
Node Role: Data Hot, Warm, Cold, Frozen

Data

Read (search) and write (index) data

- High CPU and memory usage for index/search
- High storage requirements for local shards
- Extreme disk I/O writes for hot data nodes
- High disk I/O reads for warm, cold and frozen
- Frozen storage reserves 90% for shared cache

elastic



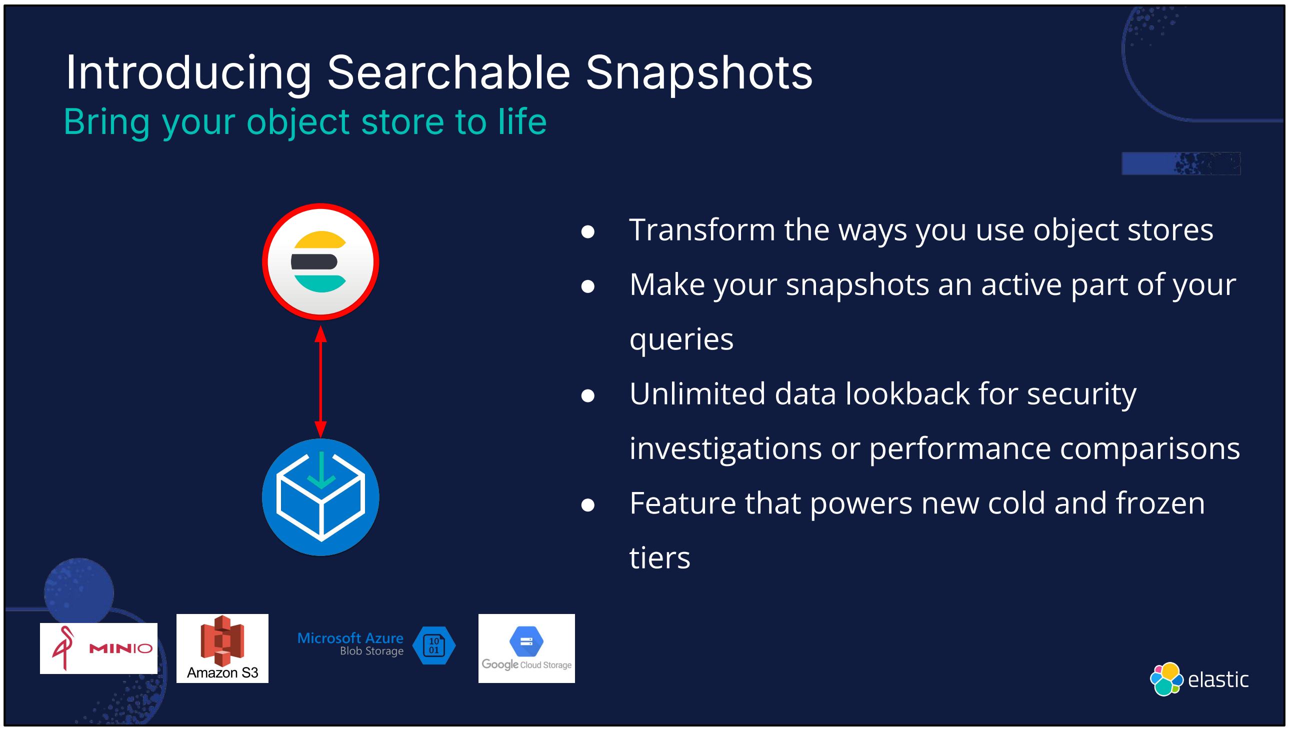
Snapshot Lifecycle Management (SLM)

Back up your data to a low-cost object store

- Back up your cluster to low-cost object store
- Handle large volumes of data effectively
- Meet regulatory requirements for retaining your data for several years
- Must be restored from snapshot before searching

MINIO
Amazon S3
Microsoft Azure
Blob Storage
Google Cloud Storage
elastic

We have, of course, had support for snapshots for a while now. Snapshot lifecycle management is a great way to back up your data into S3 or any other low-cost object store. You're able to successfully manage the large volumes of data that are continuously being ingested into your clusters without having to break your bank accounts. You can retain your data for the long haul without having to store it all on expensive, high performant hardware where only your most frequently accessed data ineeds to be located. You can further meet the regulatory and compliance requirements for data retention this way, whether it's 2 or 10 years, or something in between. The issue to-date, however, has been that these snapshots aren't searchable. You have to restore the data first into your higher-cost hardware before it can be accessed, which affects both the user experience and time-to-value.



Introducing Searchable Snapshots

Bring your object store to life

- Transform the ways you use object stores
- Make your snapshots an active part of your queries
- Unlimited data lookback for security investigations or performance comparisons
- Feature that powers new cold and frozen tiers

MINIO
Amazon S3
Microsoft Azure Blob Storage
Google Cloud Storage
elastic

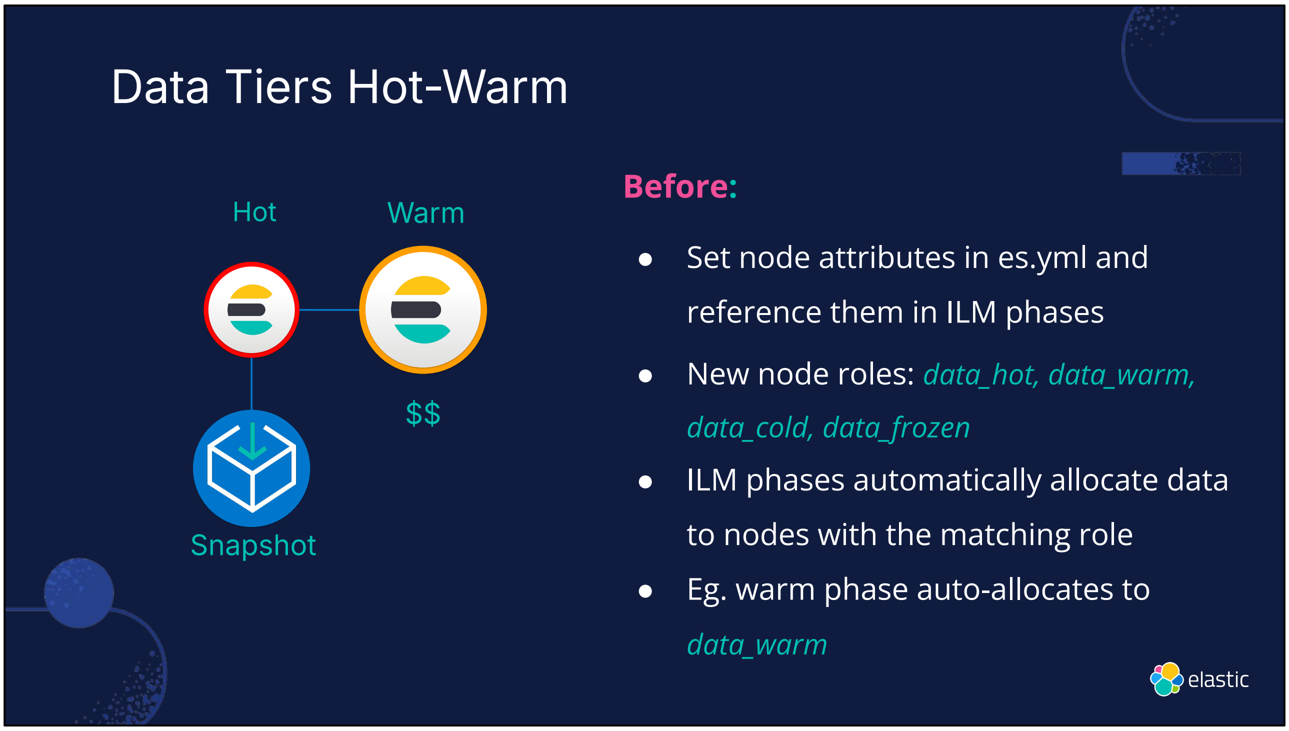
So we're introducing searchable snapshots, an exciting new capability that literally brings your object store to life. Your snapshots can now be directly queried and can become an active part of your searches, and that opens up a slew of new possibilities and use cases for you. Whether it's looking up historical data for audits, security investigations or performance comparisons, searchable snapshots has you covered. Searchable snapshots are the underlying, fundamental capability that powers new, lower-cost data tiers that we're adding to the Elastic stack to help our users more effectively balance performance, cost and data retention needs. And, like all features we build, there are APIs to directly control how searchable snapshots load, manage, and search data from your object store.

Data Tiers

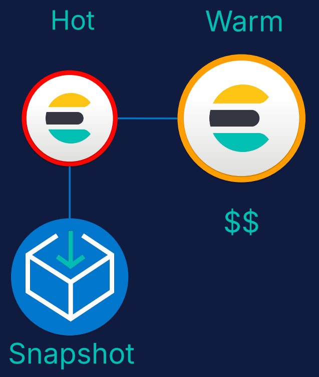
Use
data tiers
to store everything
and search it when
you need to

elastic

1. **Use frozen data tiers to store everything and search it when you need to**
   a. The frozen tier, powered by searchable snapshots, leverages cost-effective object storage such as Amazon S3, Google Cloud Storage, and Microsoft Azure Blob Storage to decrease data storage costs. The frozen tier is fully integrated with autoscaling, allowing customers to seamlessly scale in order to store massive amounts of data. It is designed to be so cost-effective that customers never need to delete data. They can save it all, and search it when they need to.



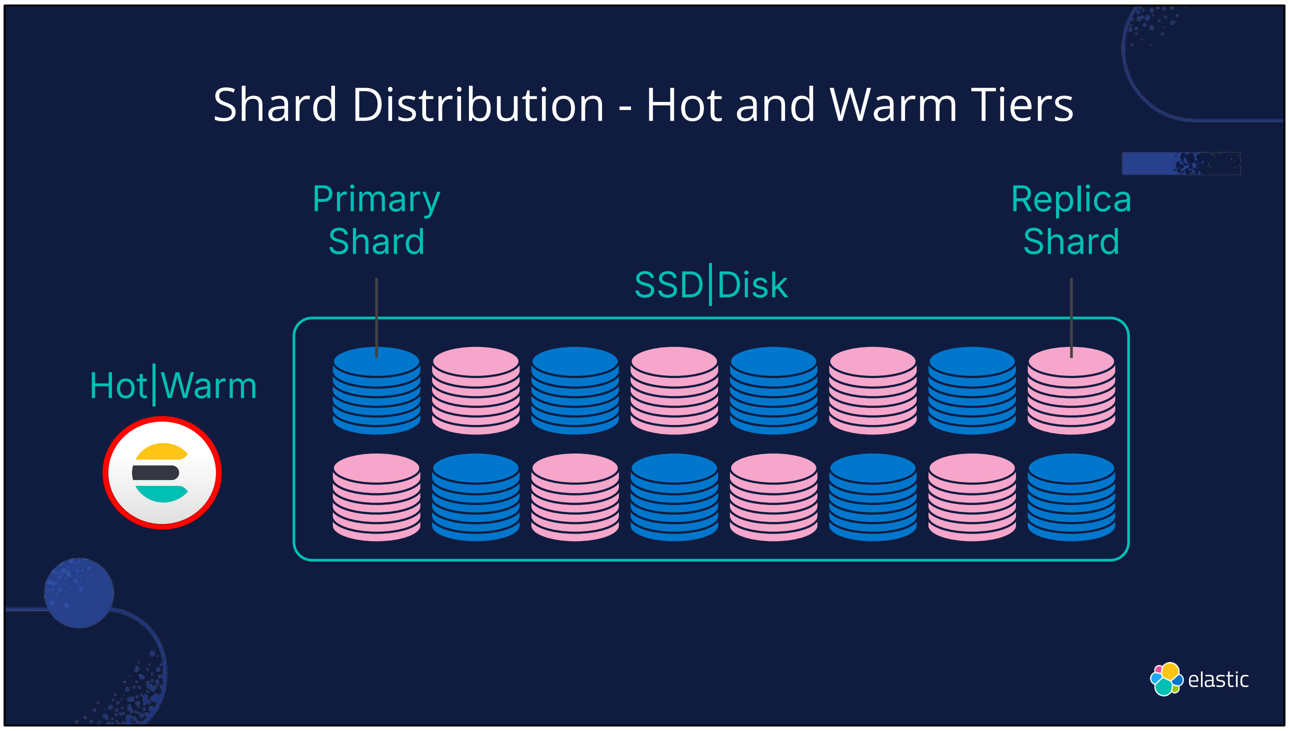
Data Tiers Hot-Warm



Before:

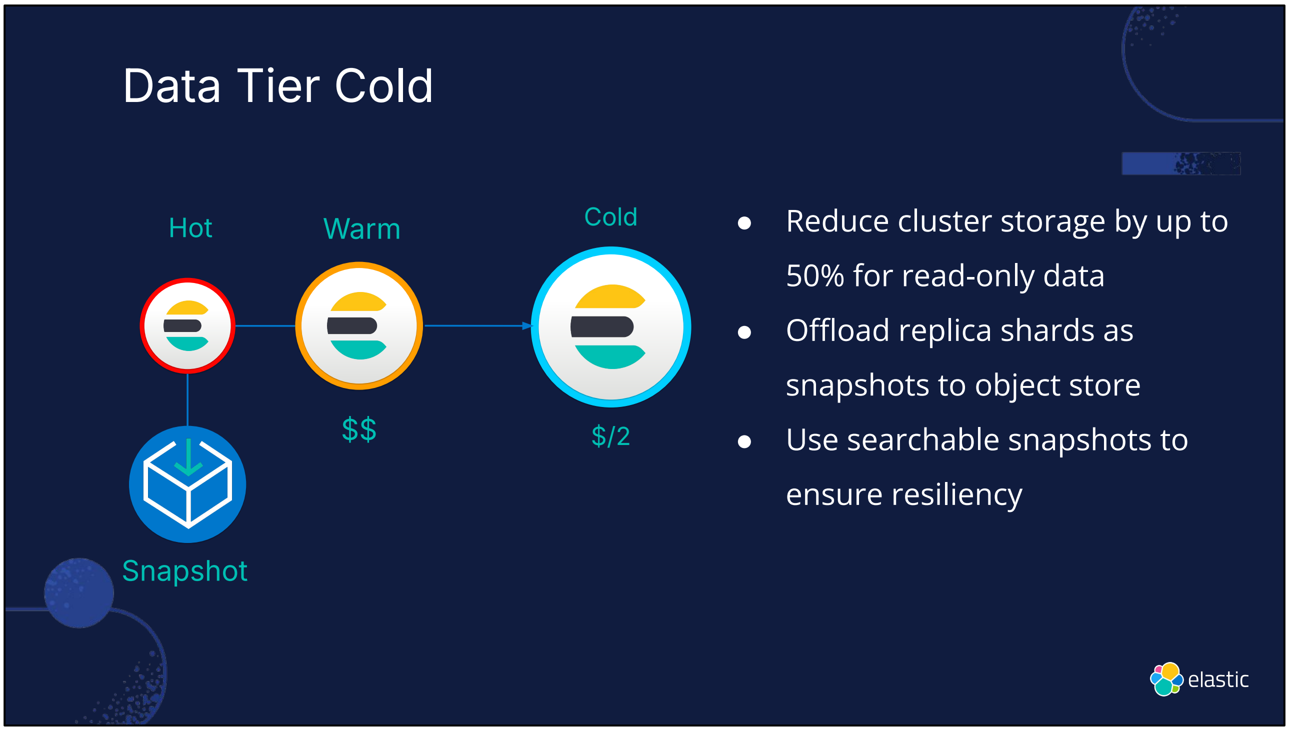
- Set node attributes in es.yml and reference them in ILM phases
- New node roles: *data_hot, data_warm, data_cold, data_frozen*
- ILM phases automatically allocate data to nodes with the matching role
- Eg. warm phase auto-allocates to *data_warm*

Let's first quickly cover what we've done with data tiers as a whole in 7.10. We have, of course, had support for data tiers for a while. Index lifecycle management provides some conventions to make it easy to manage data across hot and warm nodes, where hot is generally for fast machines with SSDs while warm is for lower cost machines with spinning disks. There is some manual work, though, involved in setting up your node attributes and referencing them appropriately in the ILM phases. In 7.10, we've made it much easier by introducing a new node role that you can assign your nodes to - hot, warm or cold, with frozen coming in the future. Now the ILM phases automatically allocate data to the nodes with the matching roles. This makes the job of admin users much easier, and it formalizes the whole data tier definition and associated flow of the data down the temperature gradient. Based on your ILM policy, data will first be kept in the hot tier for, say, 7 days before moving to the warm tier, where it's kept for 30 days, and so on.



Shard Distribution - Hot and Warm Tiers

Here’s some more detail on how it works. In the hot and warm tiers, we have a replica for every shard of data, and the replicas are stored locally on the same kind of hardware as the primary shards. In the hot tier, that’s typically on SSD’s and in the warm tier it’s on spinning disk.

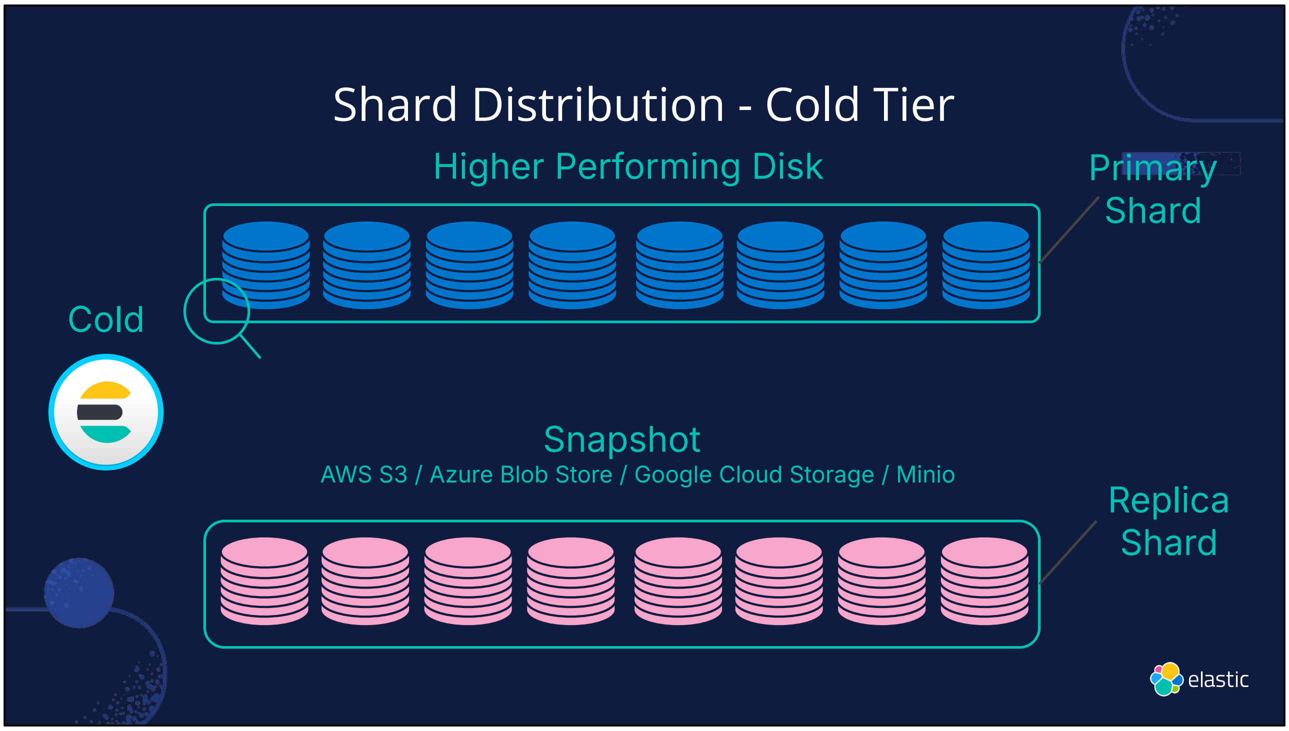


Data Tier Cold

Hot
Warm
Cold
$$
$/2
Snapshot
elastic

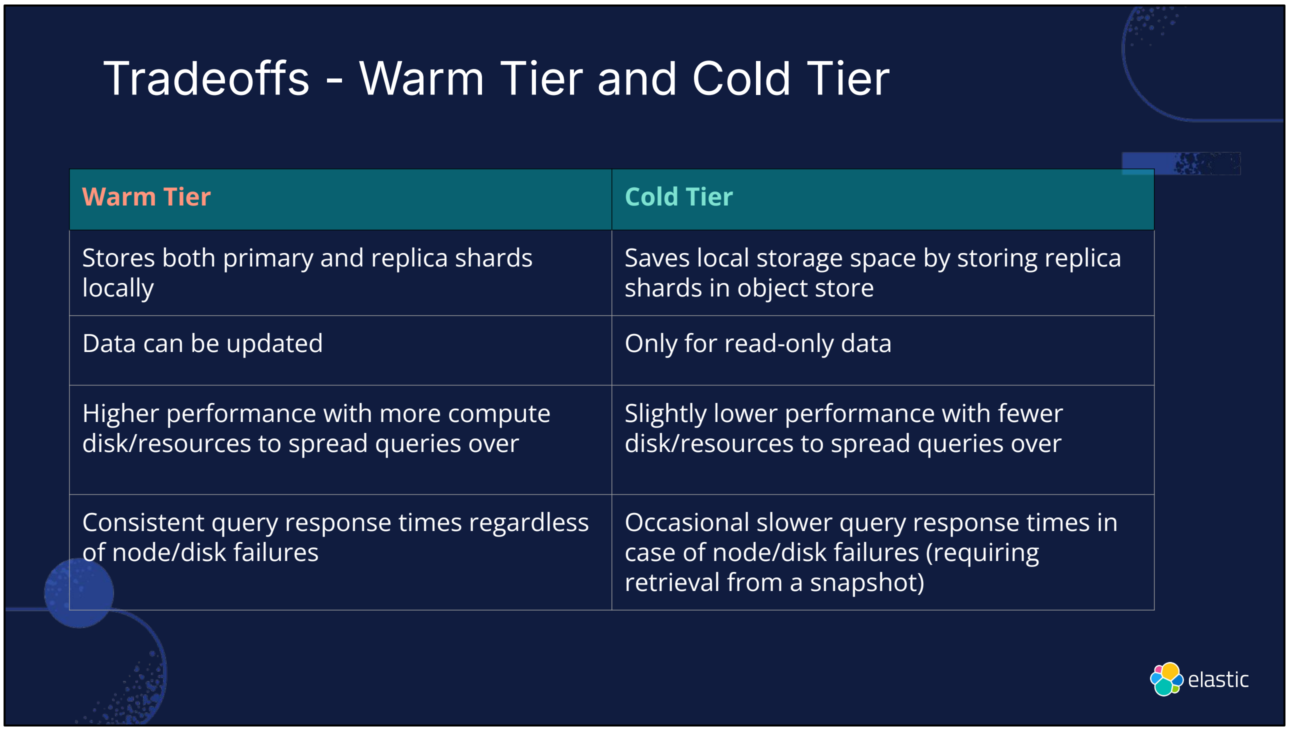
- Reduce cluster storage by up to 50% for read-only data
- Offload replica shards as snapshots to object store
- Use searchable snapshots to ensure resiliency

The hot and warm tiers are already there. And now, we're thrilled to be announcing a new, lower cost storage option in 7.10. It's called the cold tier, a natural progression after hot and warm. Whether by age or different criteria, different kinds of data have different levels of need to be searched - by frequency, concurrency, speed, volume and so on. Data that's accessed less frequently and has become read-only can easily migrate to the cold tier, where we're reducing storage costs by up to 50% compared to the warm tier. How did we do this? By offloading replica shards as snapshots to an object store like S3. Then how do we ensure resiliency, you may ask? That's where searchable snapshots comes in. We're using it to automatically retrieve data from the replica shards whenever a primary shard fails.



Shard Distribution - Cold Tier

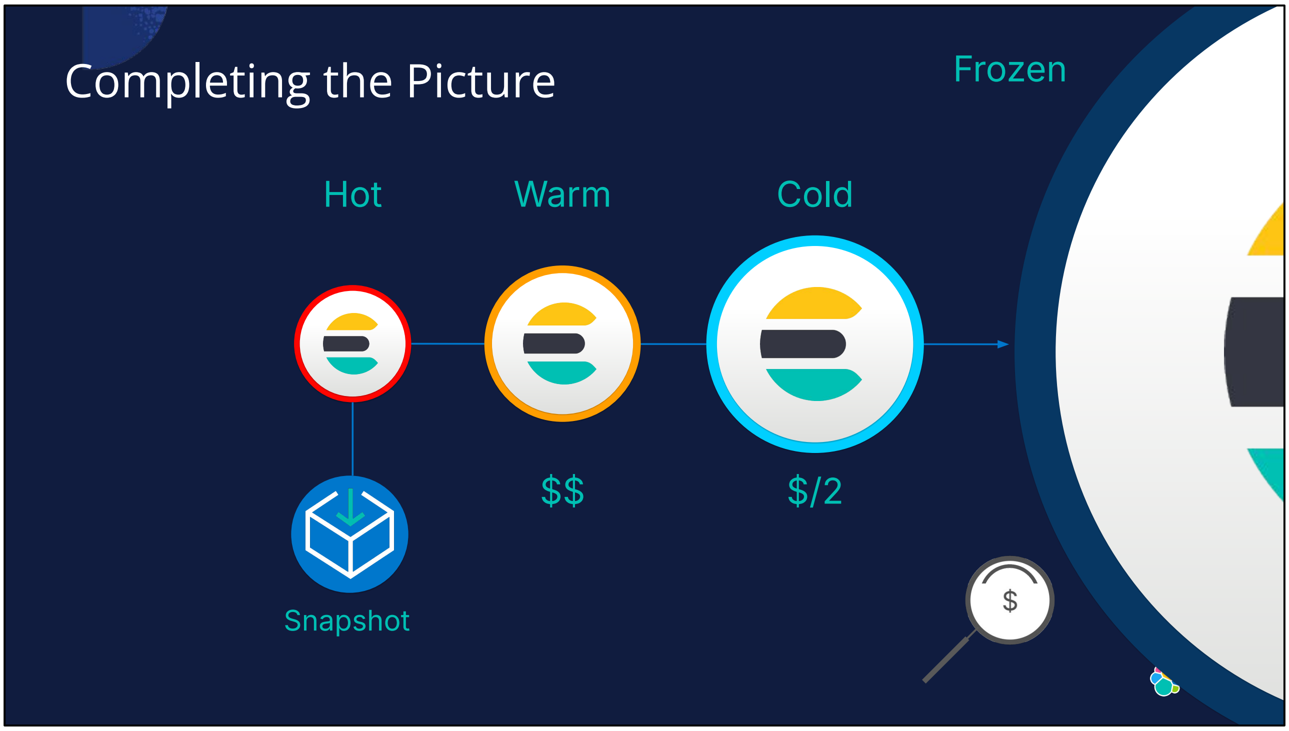
In the cold tier, what we've done is to separate replica shards from their primary counterparts. Your primary shards are still running on local disk as in your warm tier, but the replicas are now stored as snapshots in S3 or a similar object store. So we've doubled the capacity of your cold nodes, effectively halving your storage costs in comparison to the warm tier. And all with very little impact to search performance, as your primary data is still stored locally. In the case of a machine failure, we will automatically recover for you using the replica shard in S3. And because of deep improvements that we've made in Lucene and Elasticsearch, we can begin searches against those shards within seconds, long before a restore from snapshot to build the local cached copy could even complete.



Tradeoffs - Warm Tier and Cold Tier

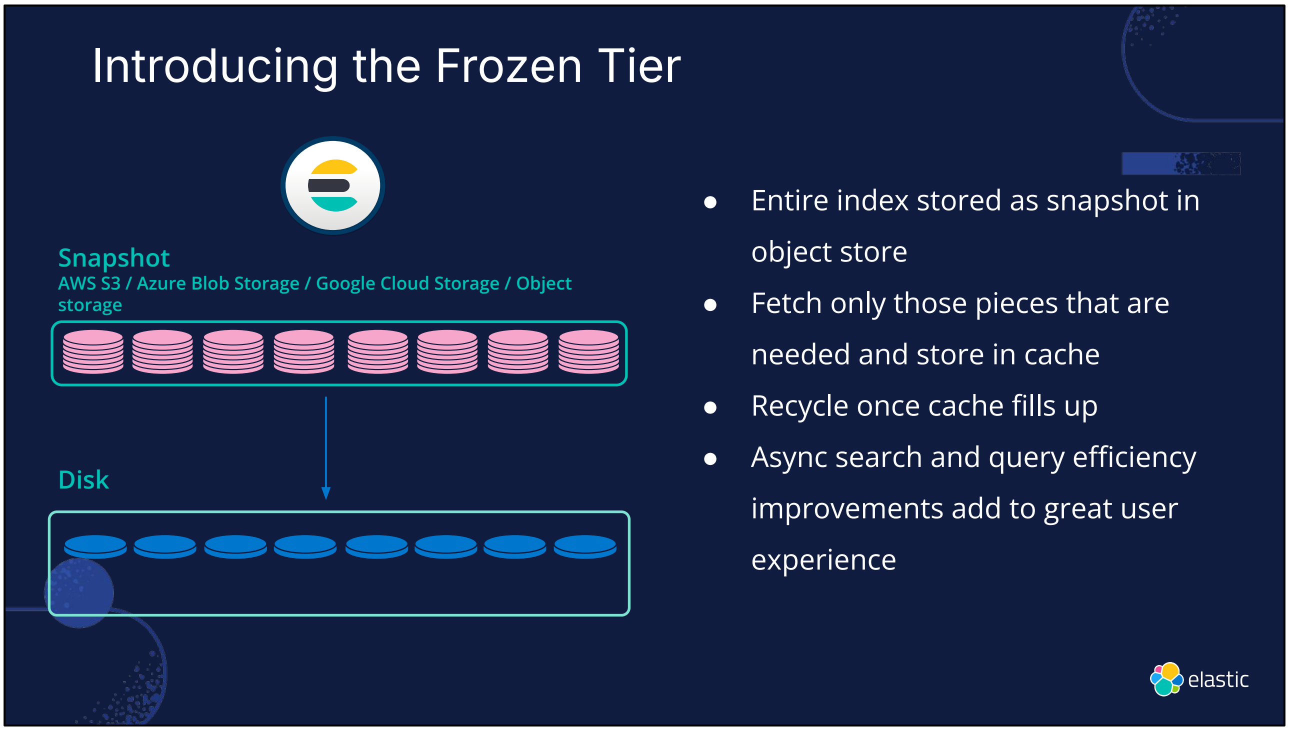
Warm Tier	Cold Tier
Stores both primary and replica shards locally	Saves local storage space by storing replica shards in object store
Data can be updated	Only for read-only data
Higher performance with more compute disk/resources to spread queries over	Slightly lower performance with fewer disk/resources to spread queries over
Consistent query response times regardless of node/disk failures	Occasional slower query response times in case of node/disk failures (requiring retrieval from a snapshot)

Great, so then why use the warm tier anymore at all? Well, as with everything there are tradeoffs. The cold tier, while it reduces your local storage needs considerably, can only store read-only data. And while impact to search performance is low, there's still an impact as there's fewer disk resources to spread your queries over. Most importantly, response times may not always be consistent. When there's a local primary shard failure, data does have to be retrieved from your snapshot. In the warm tier, a primary shard failure will result in an immediate failover to your local replica with no difference in response time. That said, the cold tier does offer great storage savings over your hot and warm tiers for the right kind of data.



Completing the Picture

Let’s talk for a few minutes about what’s coming next. Beyond 7.10, we’re aiming to complete the picture by taking things a step further. With the frozen tier, we’re storing data only as snapshots and only in the object store. The frozen tier opens up new use cases by providing the ability for our users to directly run queries on their historical data or data that has otherwise been backed up for retention purposes.

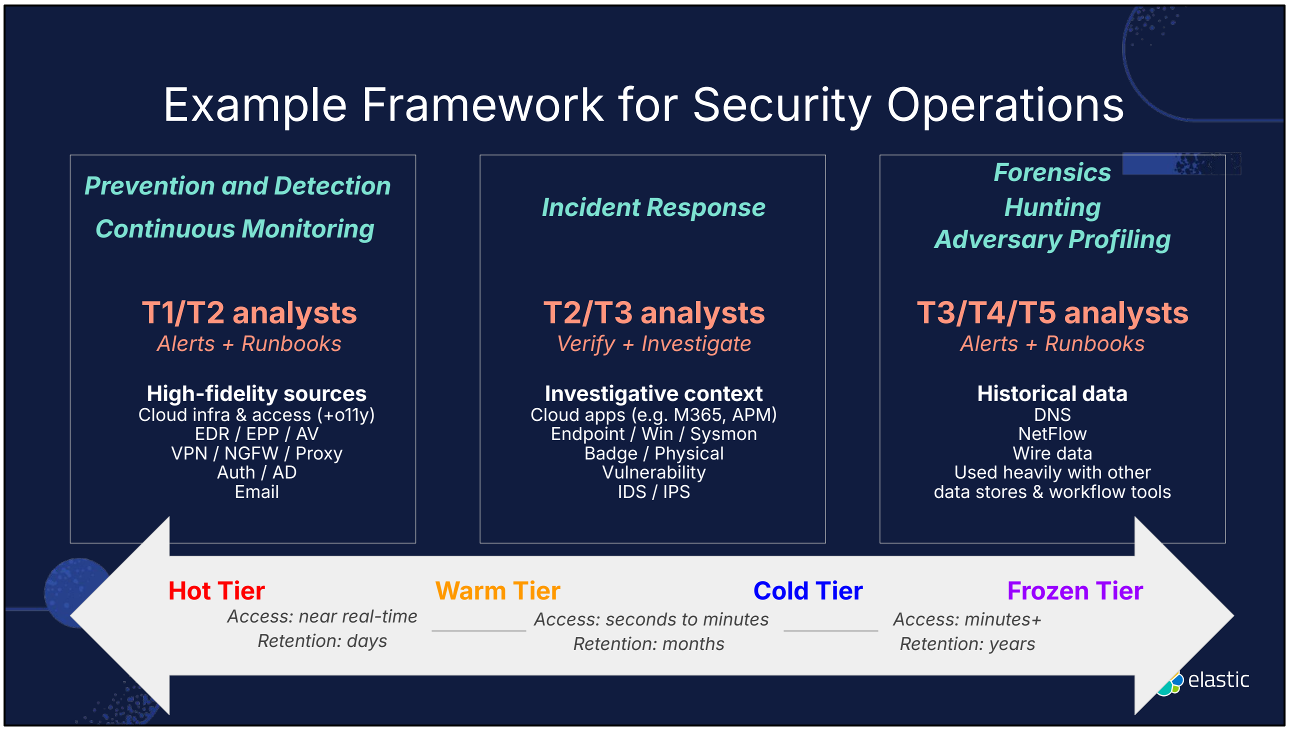


Introducing the Frozen Tier

Snapshot
AWS S3 / Azure Blob Storage / Google Cloud Storage / Object storage
Disk

- Entire index stored as snapshot in object store
- Fetch only those pieces that are needed and store in cache
- Recycle once cache fills up
- Async search and query efficiency improvements add to great user experience

What's neat about the frozen tier is that it's also powered by searchable snapshots. When you run a query against the frozen tier, you're fetching only those pieces that you need from the object store and storing it locally in a cache. If you need to run queries against that same data again, you just run against the cache for optimal performance. As the cache builds up with time, once it runs out of space we just recycle the old data for new data as more data is fetched from the object store. The other neat thing is that we've built up a collection of other features in the Elastic stack to provide the best possible experience around these slow running queries - be it async search or improvements to query efficiency such as skipping unmatched indices and block-max WAND for text search.



Example Framework for Security Operations

Prevention and Detection
Continuous Monitoring

T1/T2 analysts
Alerts + Runbooks

High-fidelity sources
Cloud infra & access (+o11y)
EDR / EPP / AV
VPN / NGFW / Proxy
Auth / AD
Email

Incident Response

T2/T3 analysts
Verify + Investigate

Investigative context
Cloud apps (e.g. M365, APM)
Endpoint / Win / Sysmon
Badge / Physical
Vulnerability
IDS / IPS

Forensics
Hunting
Adversary Profiling

T3/T4/T5 analysts
Alerts + Runbooks

Historical data
DNS
NetFlow
Wire data
Used heavily with other
data stores & workflow tools

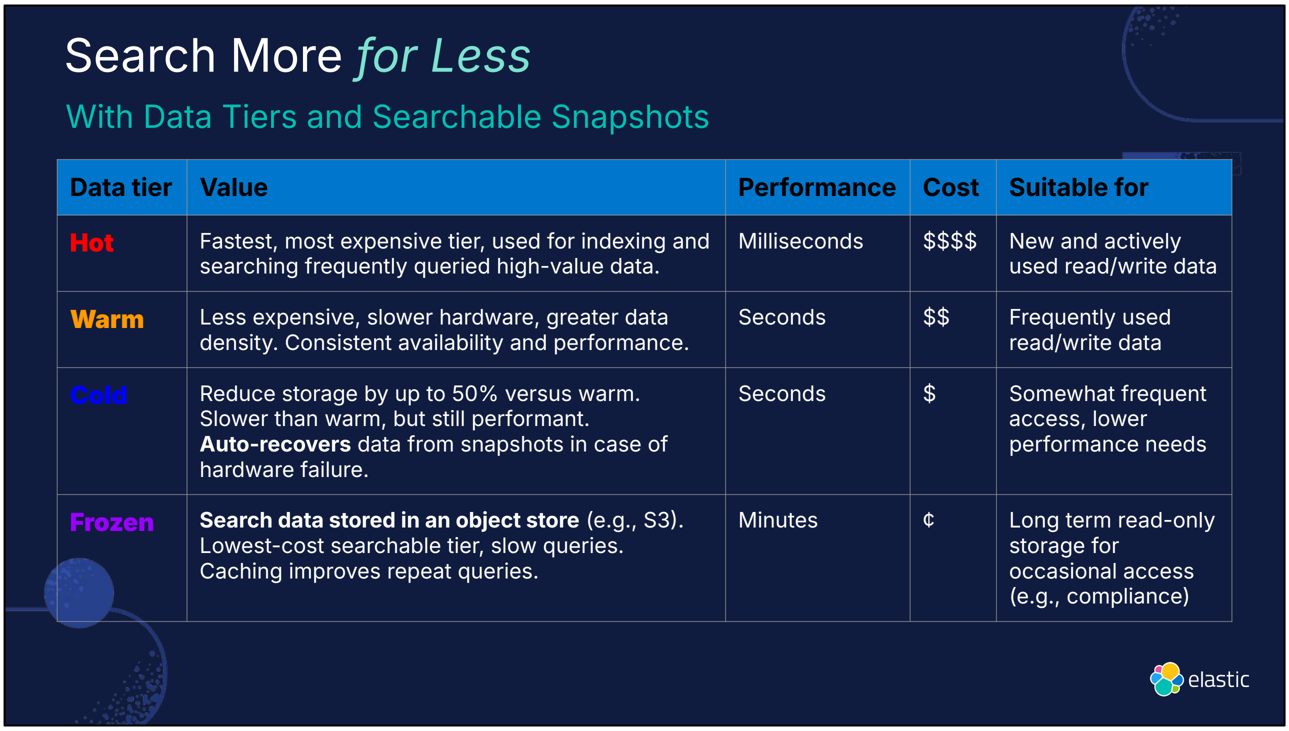
Hot Tier
Access: near real-time
Retention: days

Warm Tier
Access: seconds to minutes
Retention: months

Cold Tier

Frozen Tier
Access: minutes+
Retention: years

elastic

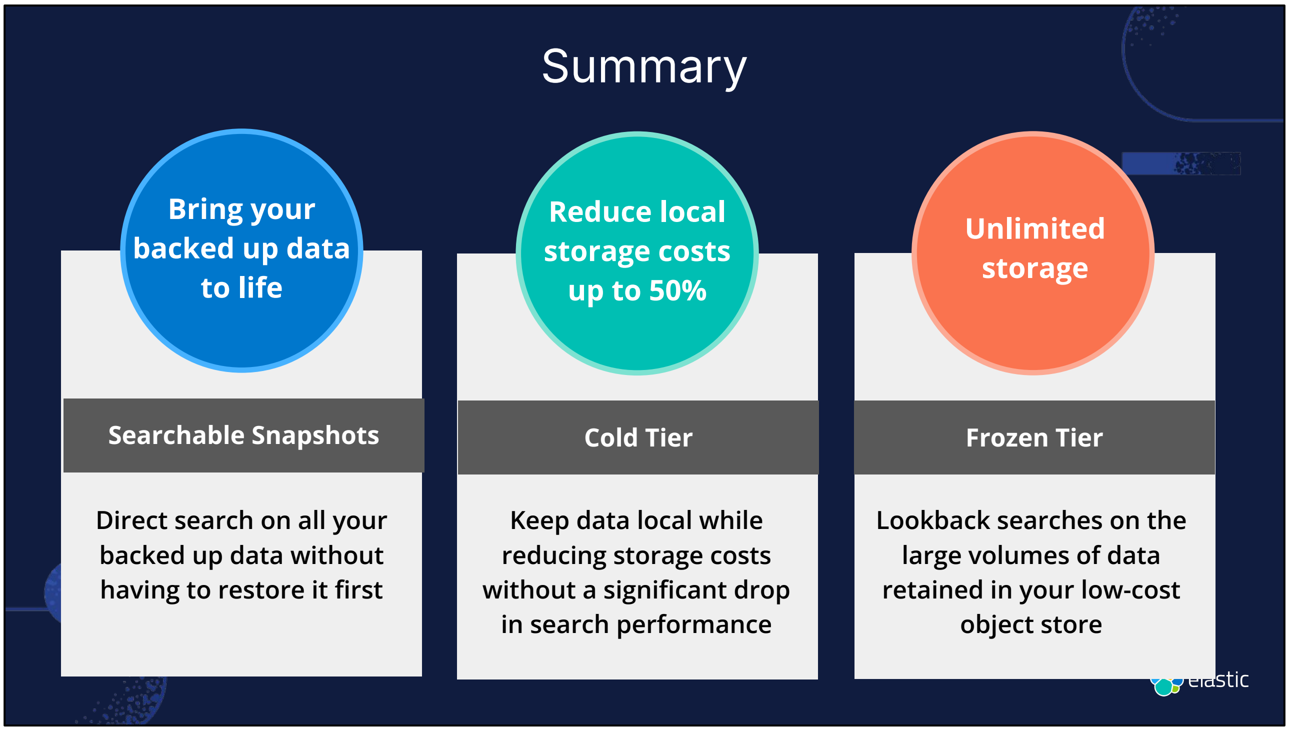


Search More *for Less*

With Data Tiers and Searchable Snapshots

Data tier	Value	Performance	Cost	Suitable for
Hot	Fastest, most expensive tier, used for indexing and searching frequently queried high-value data.	Milliseconds	$$$$	New and actively used read/write data
Warm	Less expensive, slower hardware, greater data density. Consistent availability and performance.	Seconds	$$	Frequently used read/write data
Cold	Reduce storage by up to 50% versus warm. Slower than warm, but still performant. **Auto-recovers** data from snapshots in case of hardware failure.	Seconds	$	Somewhat frequent access, lower performance needs
Frozen	**Search data stored in an object store** (e.g., S3). Lowest-cost searchable tier, slow queries. Caching improves repeat queries.	Minutes	¢	Long term read-only storage for occasional access (e.g., compliance)

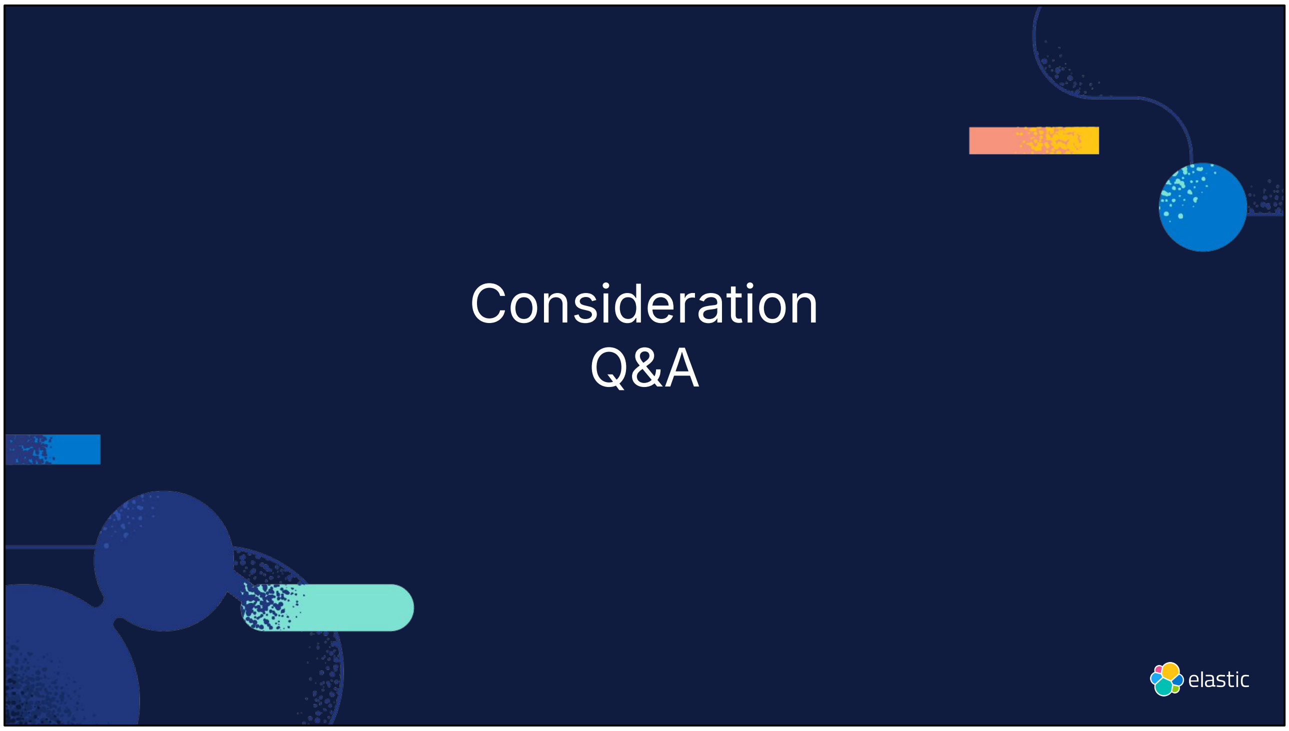
elastic



Summary

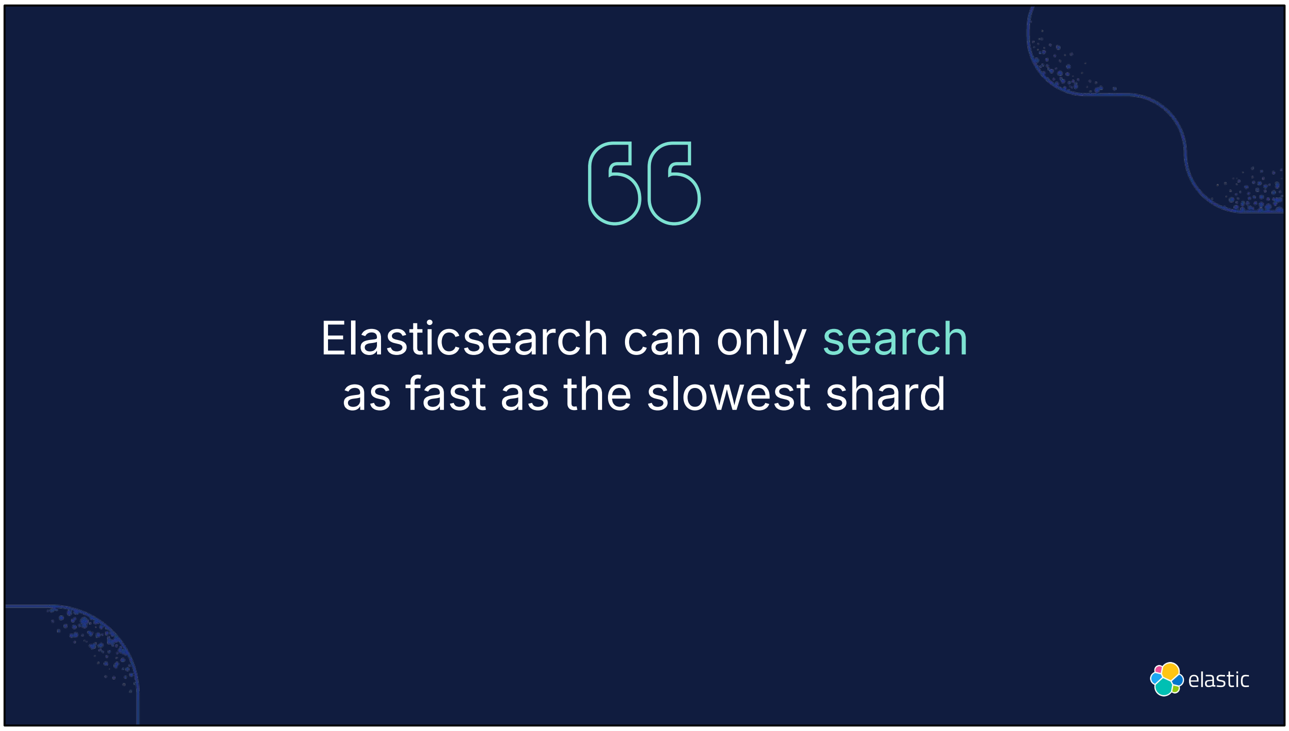
Searchable Snapshots	Cold Tier	Frozen Tier
Bring your backed up data to life	Reduce local storage costs up to 50%	Unlimited storage
Direct search on all your backed up data without having to restore it first	Keep data local while reducing storage costs without a significant drop in search performance	Lookback searches on the large volumes of data retained in your low-cost object store

This is a summary of what we've just covered. Searchable snapshots, the underlying capability that brings your backed up data to life with direct search access, and the two new tiers it powers. The cold tier which can halve your local storage costs by offloading your replica shards to an object store without a significant drop in performance, and the frozen tier which stores your data entirely in the object store. It's the right set of capabilities to provide our users with to more effectively balance cost, performance and their data retention needs. No more restores, no more rehydration. So many new use cases opened up, be it in observability, security or enterprise search. Query years of archived data without having to go through the slow and costly process of restoring indices from snapshots. Arm threat hunters with lookbacks on years of high-volume security data. Search across all your application content and historical workplace records without breaking the bank. That's what this is all about. And, with that, let's talk about how we want to message all of this.



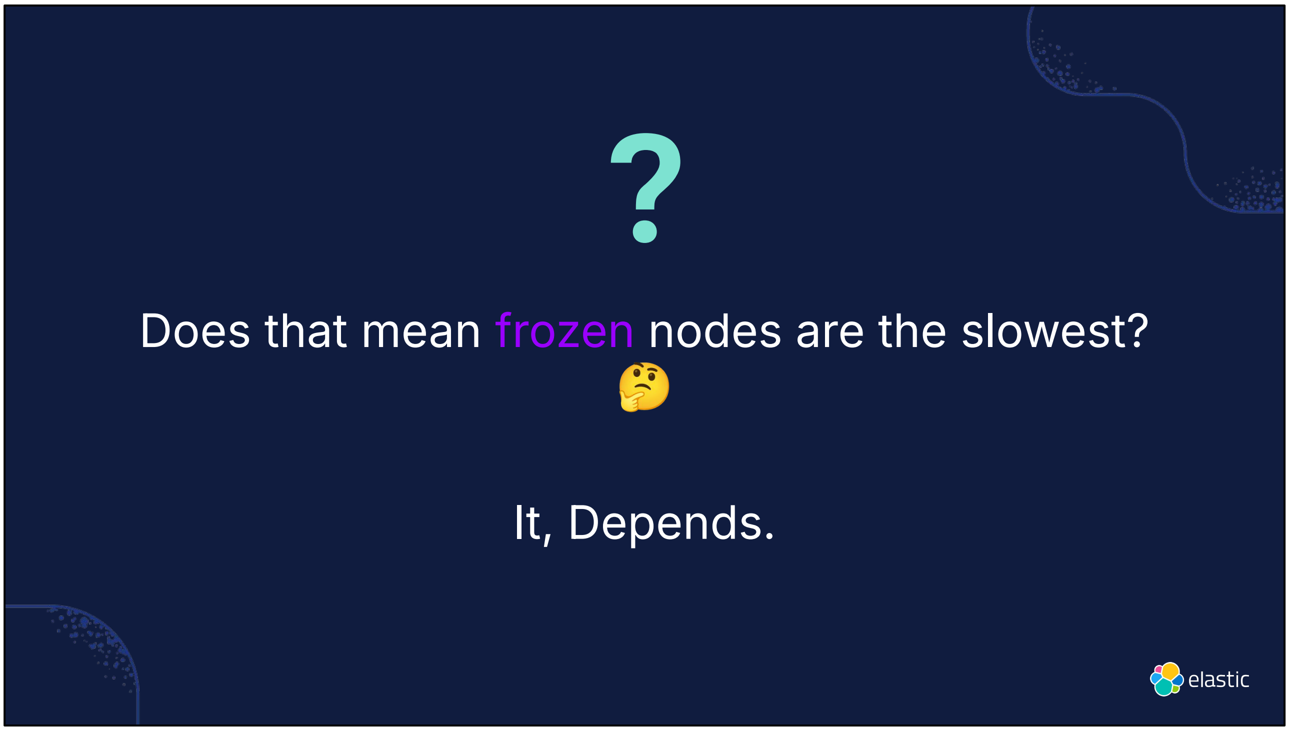
Consideration Q&A

elastic



Elasticsearch can only search
as fast as the slowest shard

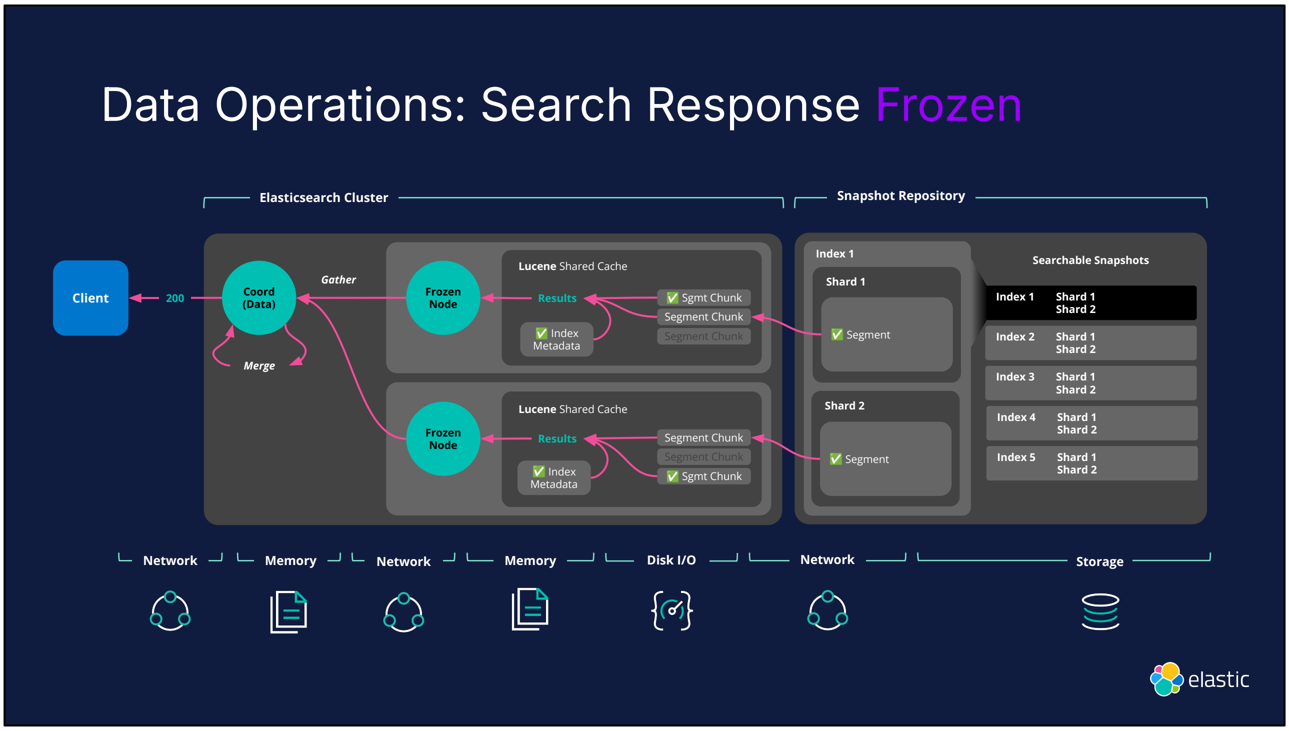
elastic



?

Does that mean frozen nodes are the slowest?
🤔

It, Depends.



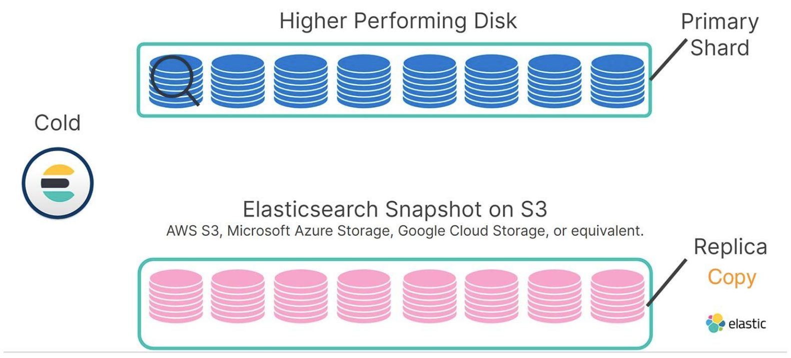
Data Operations: Search Response Frozen

Frozen nodes can resolve request in three distinct ways:

1. By only using the index metadata
2. By landing on a segment cache hit
3. By pulling a segment chunk from the repository

With Searchable Snapshot in Cold tier

- Primary Shard on disk
- Replica data copy on object store (AWS S3, etc)


Higher Performing Disk
Primary Shard
Cold
Elasticsearch Snapshot on S3
AWS S3, Microsoft Azure Storage, Google Cloud Storage, or equivalent.
Replica Copy
elastic

With Searchable Snapshot in Cold tier - 2

- Per manual
    - By default, searchable snapshot indices have no replicas.

- Why?
    - Replica is used for keeping high availability
    - But for searchable snapshot, in cold tier, the data copy is stored in i.e. S3
    - And S3 storage can guarantee the reliability (*)

(*) Per manual, the blob storage offered by all major public cloud providers typically offers very good protection against data loss or corruption. If you manage your own repository storage then you are responsible for its reliability.

elastic

With Searchable Snapshot in Cold tier - 3

- Index name
  - Indices managed by ILM are prefixed with **“restored-”** when *fully mounted*.

- Why it’s called “fully mounted”?
  - Because it “Loads a full copy of the snapshotted index’s shards on node-local storage within the cluster.” (manual)

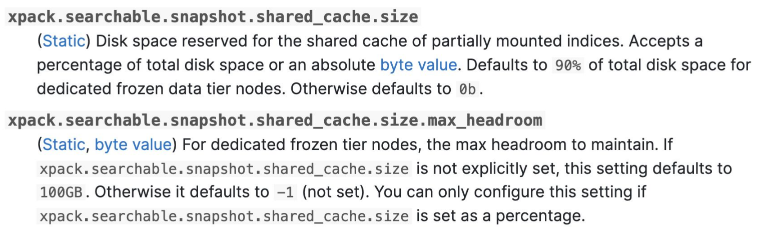
With Searchable Snapshot in Frozen tier

- Shard data on object store i.e. S3
- Cache on local disk

With Searchable Snapshot in Frozen tier - 2

- No real shard data are stored on local disk
  - Index size shows 0b on disk

- 2 parameters for using cache on local disk



`xpack.searchable.snapshot.shared_cache.size`
(Static) Disk space reserved for the shared cache of partially mounted indices. Accepts a percentage of total disk space or an absolute byte value. Defaults to `90%` of total disk space for dedicated frozen data tier nodes. Otherwise defaults to `0b`.

`xpack.searchable.snapshot.shared_cache.size.max_headroom`
(Static, byte value) For dedicated frozen tier nodes, the max headroom to maintain. If `xpack.searchable.snapshot.shared_cache.size` is not explicitly set, this setting defaults to `100GB`. Otherwise it defaults to `-1` (not set). You can only configure this setting if `xpack.searchable.snapshot.shared_cache.size` is set as a percentage.

With Searchable Snapshot in Frozen tier - 3

- Index name
  - Indices managed by ILM are prefixed with **“partial-”** when *partially mounted*.

- Why it’s called “partially mounted”?
  - Because it “Uses a local cache containing only recently searched parts of the snapshotted index’s data.” (manual)

elastic