Elasticsearch index management for PaaS style logging system

Jaeik Lee, Qin Tang

NAVER

Agenda

- Introduction to NELO
- Elasticsearch in NELO: Phase 1
- Problem of Phase 1
- Elasticsearch in NELO: Phase 2
- Index Manager in Details

NAVER

Introduction to NELO

What is NELO?

- Forwarding logs: SDKs
- Collecting logs: HTTP/HTTPS/Thrift/Syslog
- Real-time/Scheduled alerts
- Crash log desymbolicating(deobfuscating)
- Webapp with Kibana dashboard
- OpenAPI for custom use cases

Design Considerations 1: Heterogeneous Logs

Platforms & Frameworks

Mobile

Desktop

Applications

Design Considerations 2: Scale

490

Node

9 instances

11 clusters

7.6T

Documents

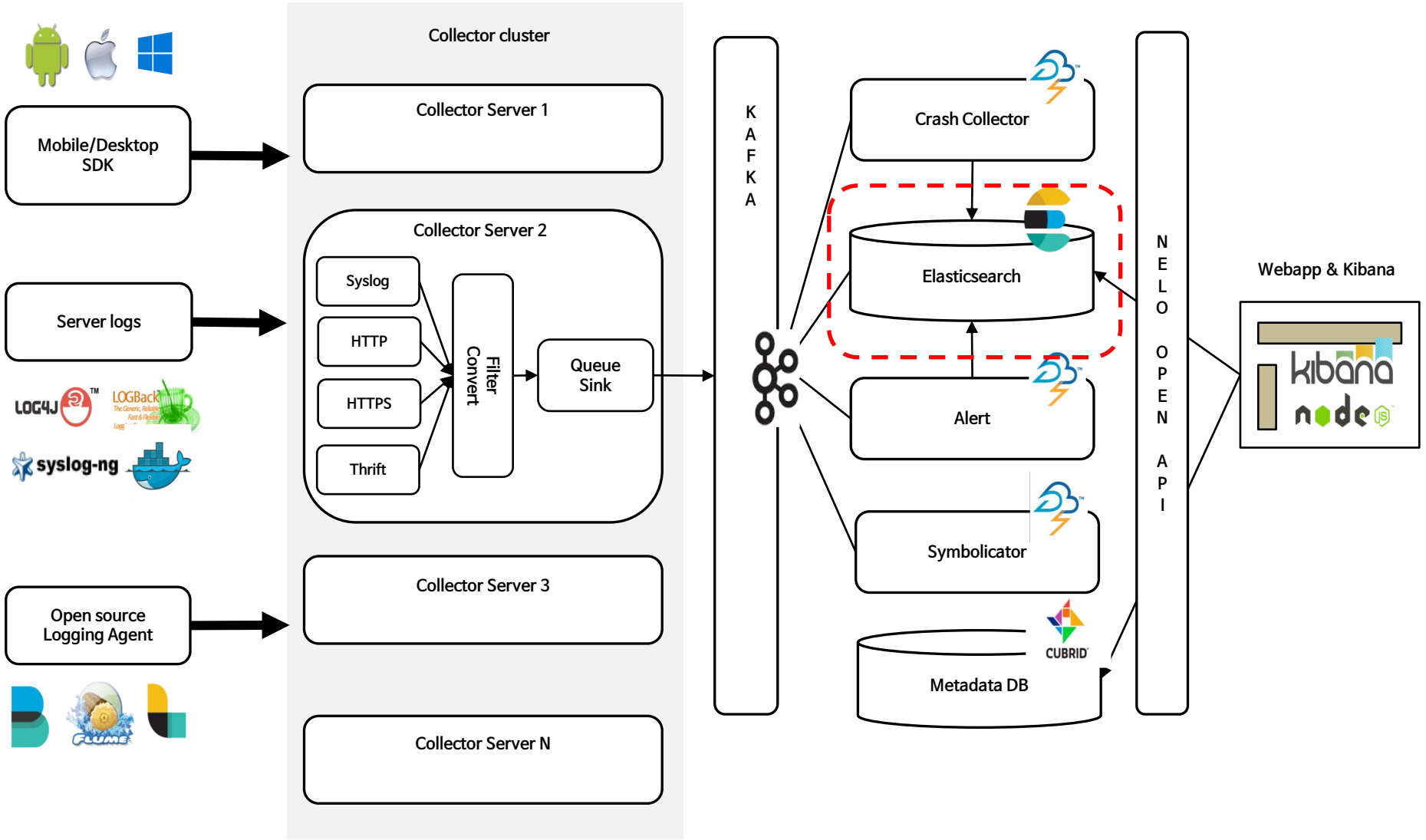
Total number of logs

1.6PB

Size

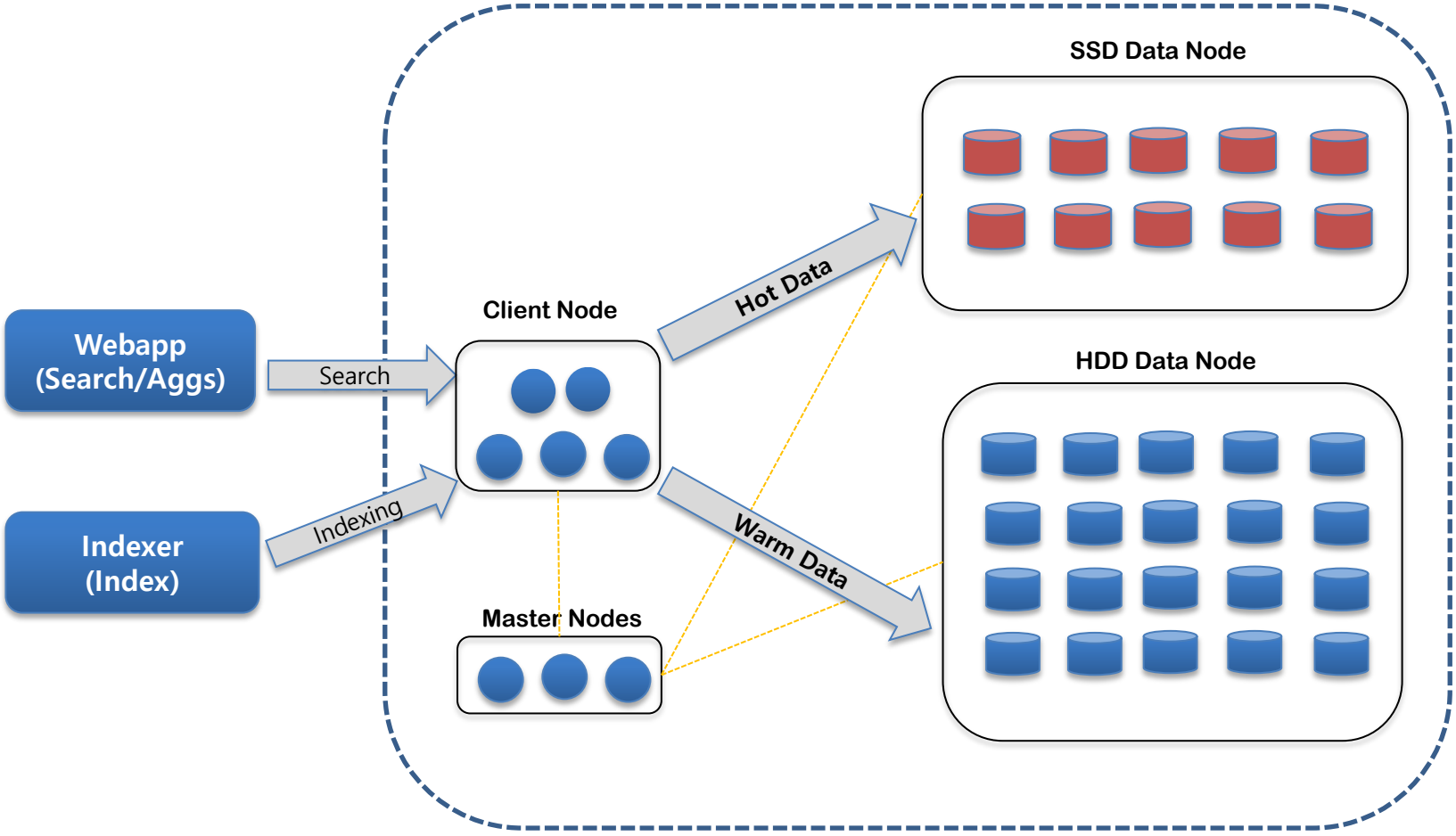
Total size of logs

Architecture of NELO



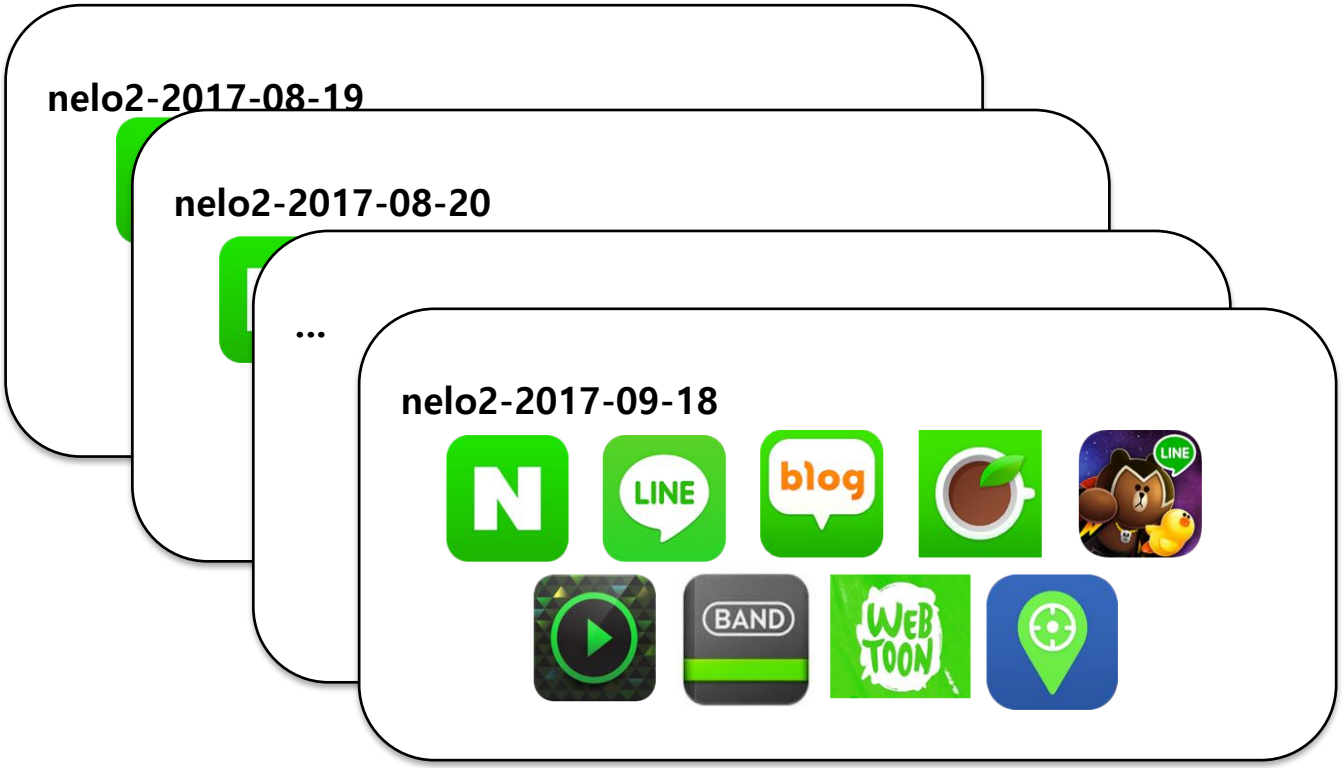
Elasticsearch in NELO: Phase 1

Cluster Architecture V1.0



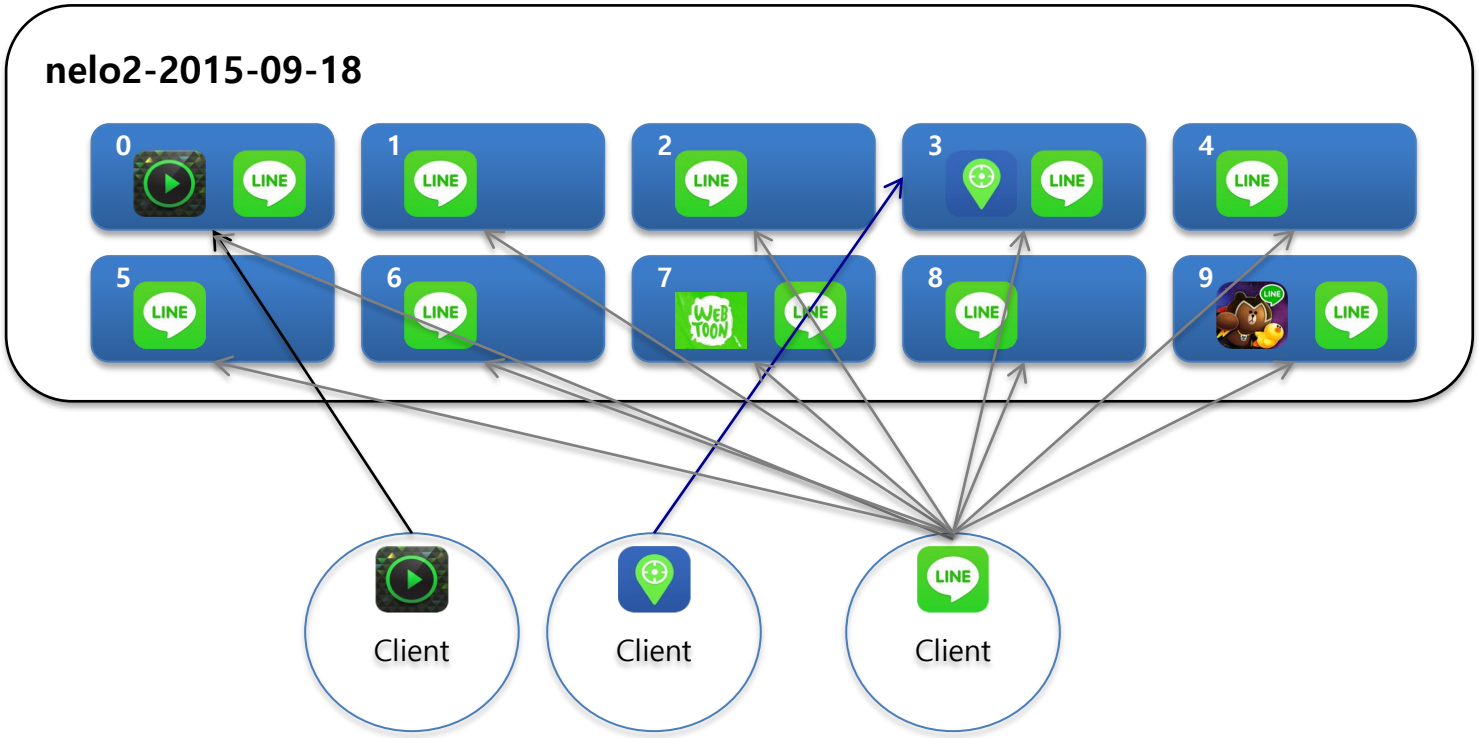
Index Model V1.0

- 1 Index per day → daily index lifecycle management
- One Index: All projects
- One Project : One Mapping
- Various retention time according to the instances (1 M, 3M, 2Y, 5Y)



Searching/Indexing with Routing V1.0

- Use custom routing both in indexing and searching
  - Small project: store only in one shard (custom routing: project name)
  - Big project: distribute logs over all shards (default routing)



Problems

Problem 1: Mapping explosion

- More projects created, more mapping created

  3,000 projects → 3,000 mappings (6 MB per day)

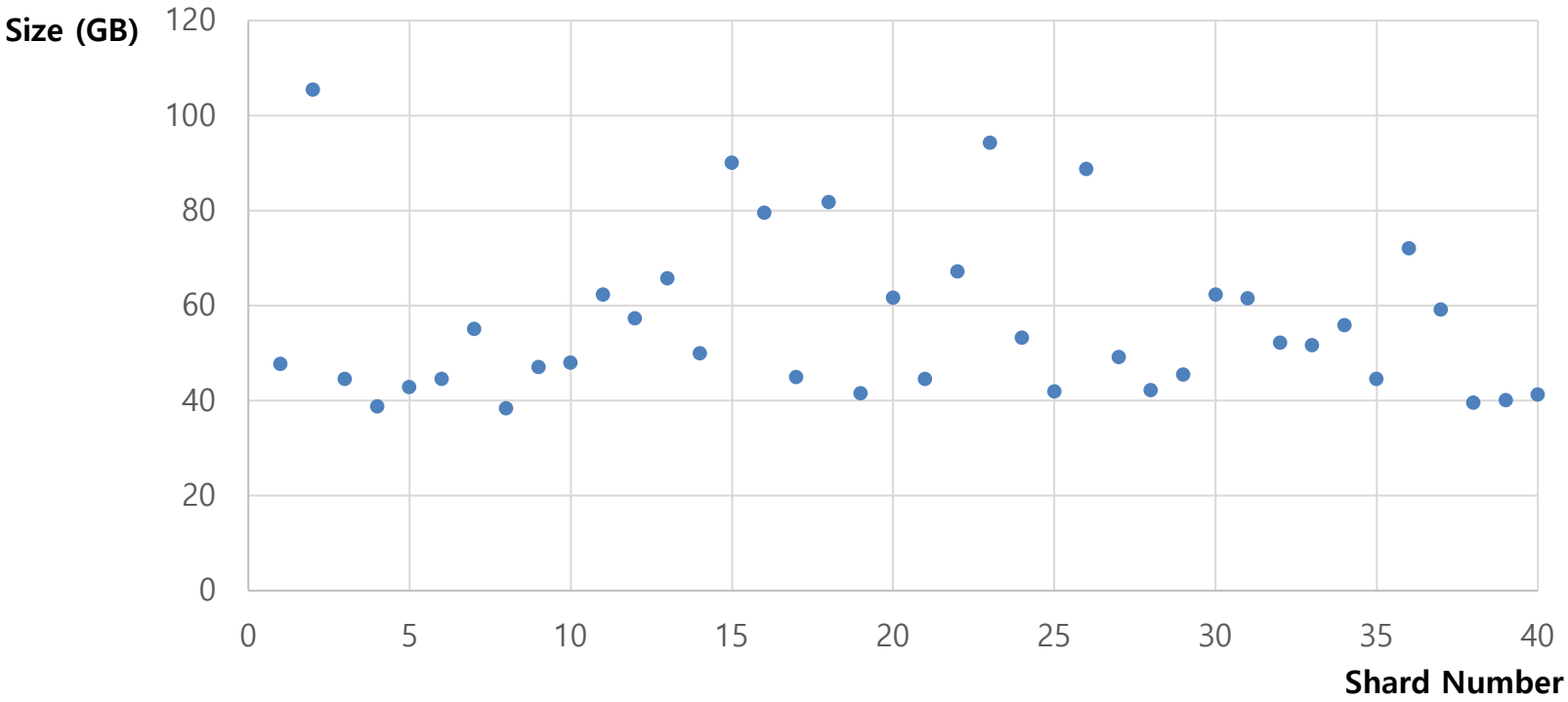
- Elasticsearch synchronize the mapping of an index among all nodes
- Sometimes entire cluster is blocked by update mapping event

```
[2017-05-30 21:36:57,773][WARN ][cluster.service ] [elastic09.nelo2]
cluster state update task [put-mapping [naver-project],put-mapping
[naver-project]] took 5.1m above the warn threshold of 30s
```

Problem 2: Imbalance of shard size

- Skewed shard due to Routing

  Utilization of node resources is not fair



Problem 3: Impact of big projects

- Big Projects: less than 10 %

  Sending more than 80% of logs

  → Sharing shards

  → Sharing resources

  → Affecting the performance of small projects

Problem 4: Retention time

- Requirement for custom retention time
- Not easy to support custom retention time due to sharing same indices among all projects

Problem 5: Type conflict

- Requirement of custom type
- From Elasticsearch 2.x, even if mapping is different, type of a field with same name in same index should have same type.

  → Mapping type of new fields are conflicted with existing fields

Question

Do we need to create separate indices for every project?

3000 projects→ 3000 indices→ more than 3000 shards per days

It's not scalable.

Elasticsearch in NELO: Phase 2

NAVER

Improvements in V 2.0

- Multi-cluster within one instance of NELO

  → keep smaller size of meta data

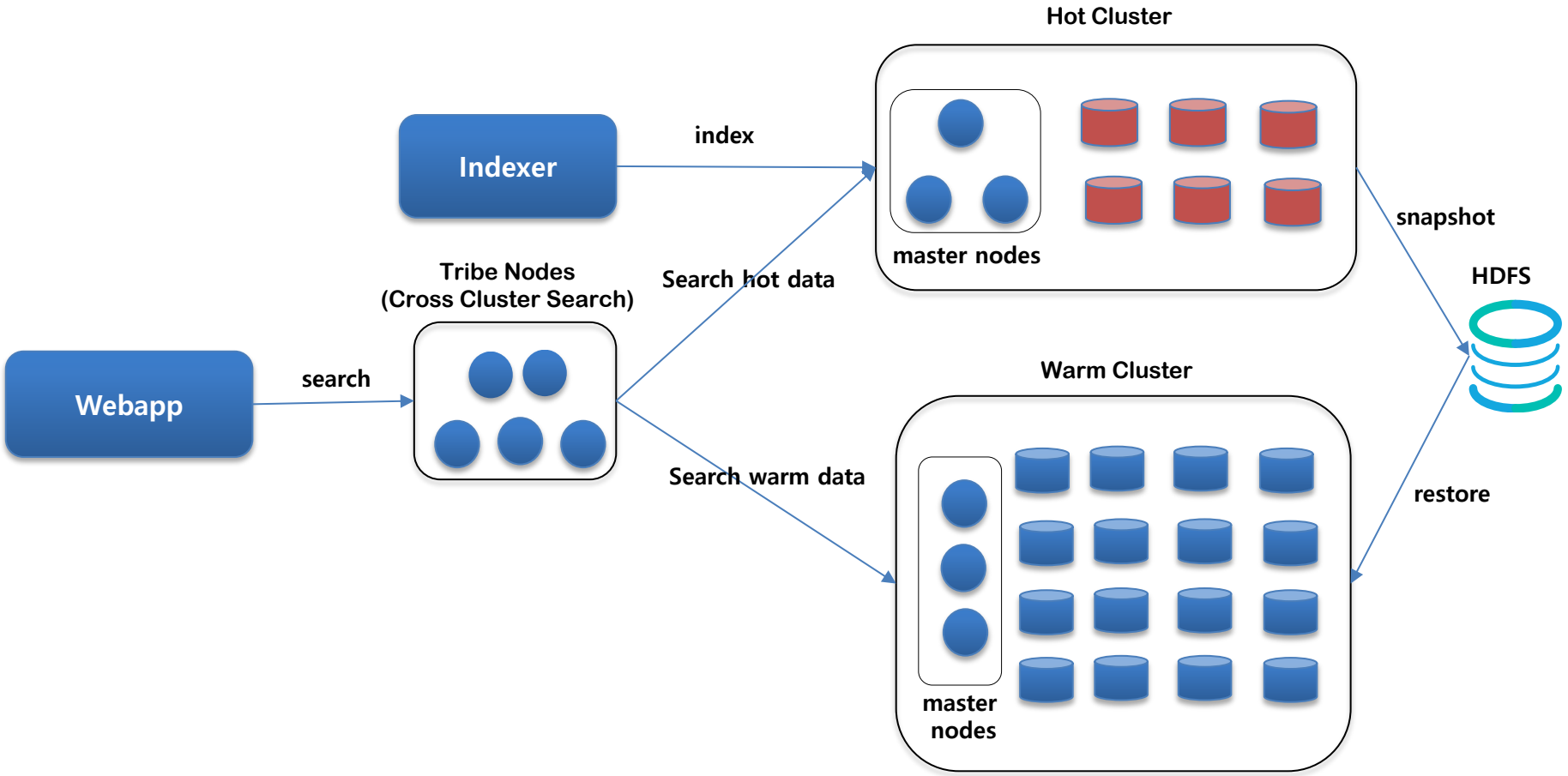
- Multi-Indices per day

  → supporting custom mapping, custom retention time

- Dynamic estimation of the number of shards

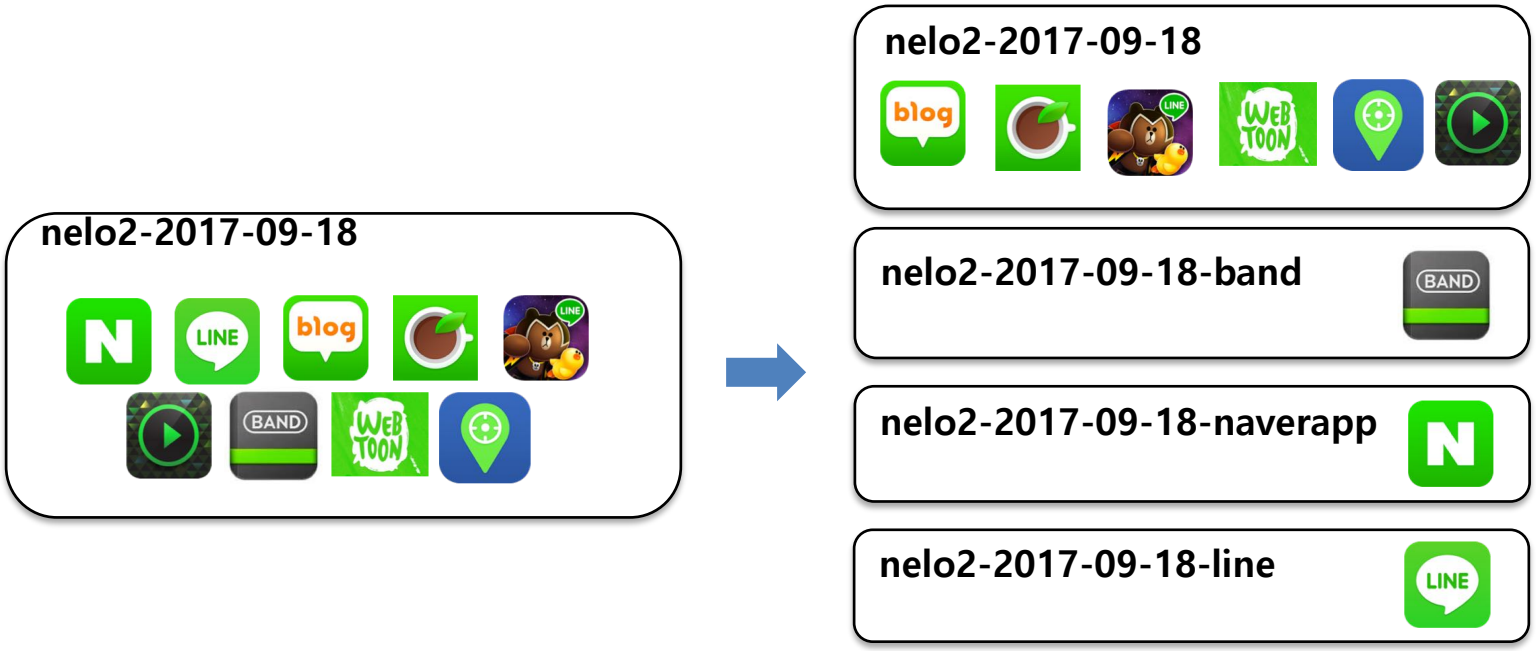
  → keep optimal number of shards

Elasticsearch Cluster Architecture V 2.0



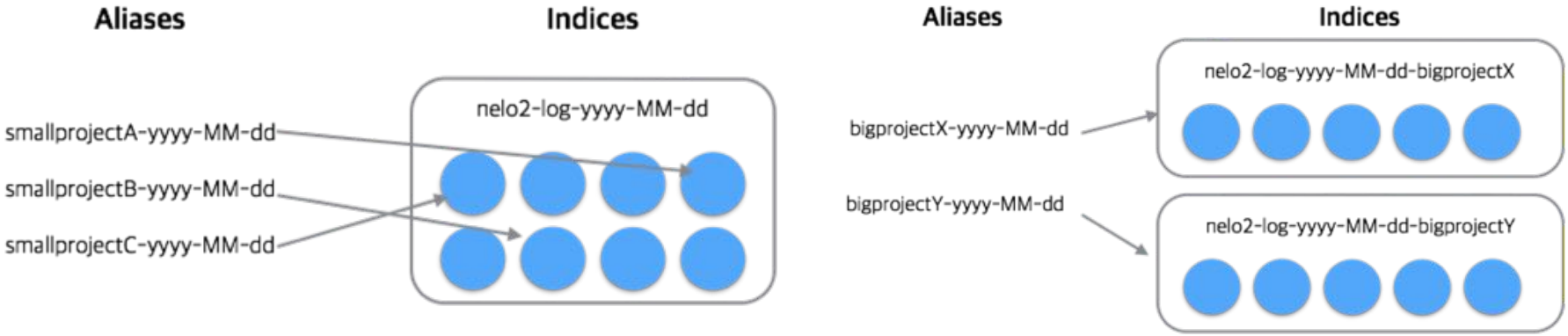
Index Model 2.0

- Separate the indices
  - For small projects, stored in common indices
  - For big projects, stored in dedicated indices



Indexing/Searching V 2.0

- Use aliases no matter a project is indexed either in common or dedicated index.
  - Alias name: <project name>-yyyyMMdd
  - Index name
    - Common: nelo2-log-yyyy-MM-dd
    - Dedicated: nelo2-log-yyyy-MM-dd-<project name>

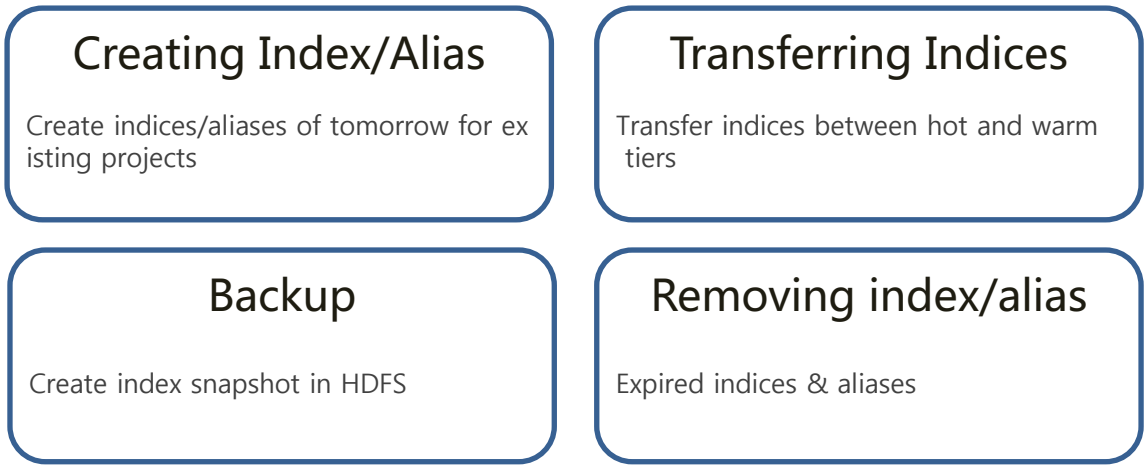


Index Manager in Details

What Index Manager is?

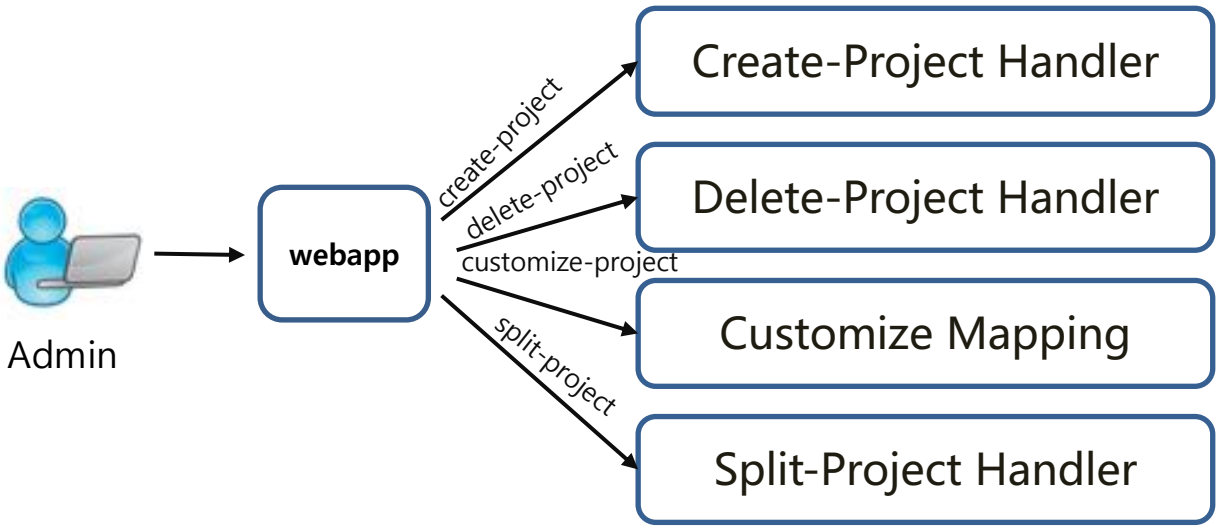
- Manage life cycle of indices and aliases
  - create/delete daily indices and aliases
  - allocate indices in either hot or warm tier
  - handle instant project creation/delete event

Scheduled & Real-time



Scheduled

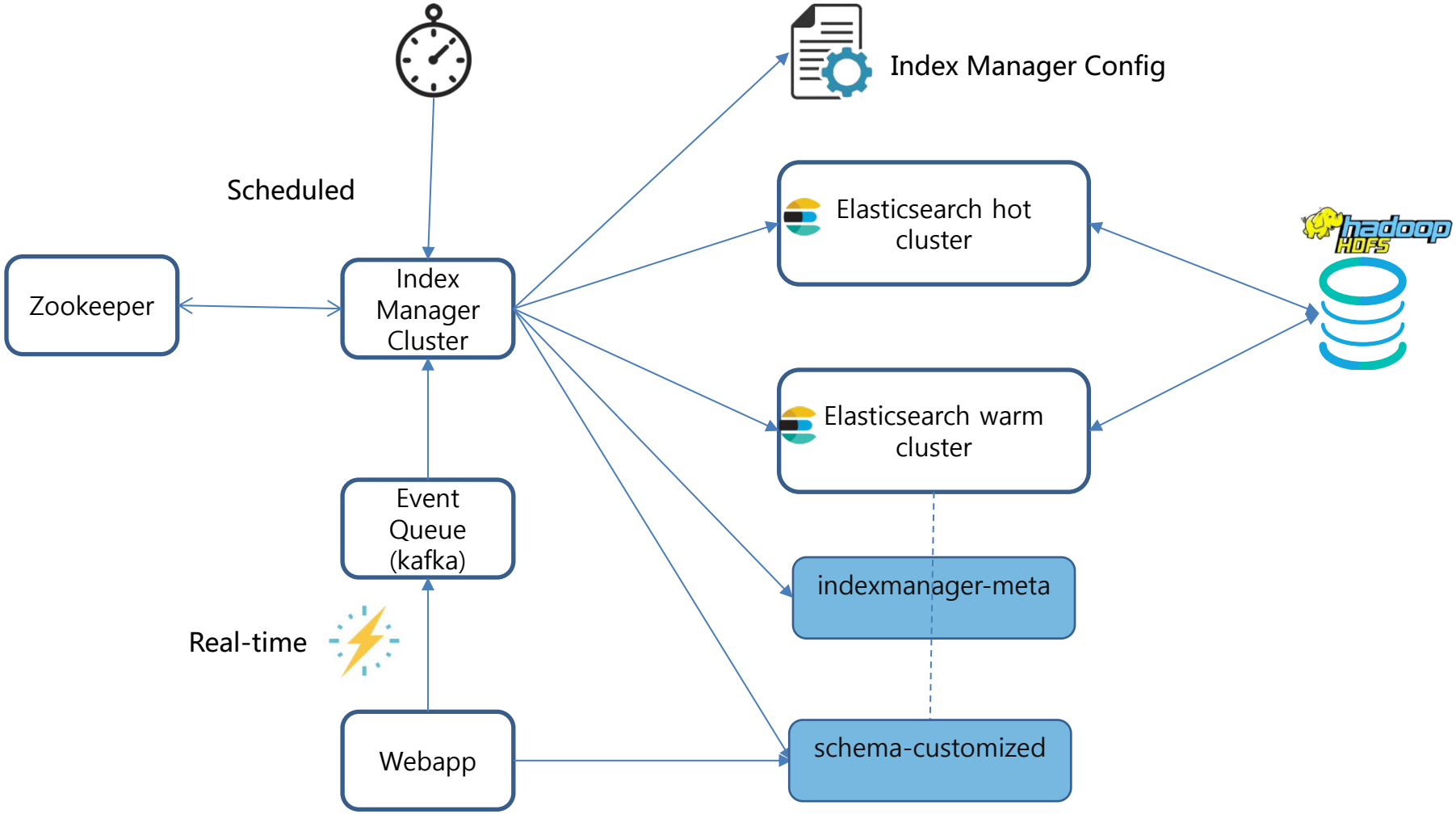
Maintaining
Existing projects



Real-time

Handling instant
events for projects

Architecture with Index manager



Creating Index/Alias

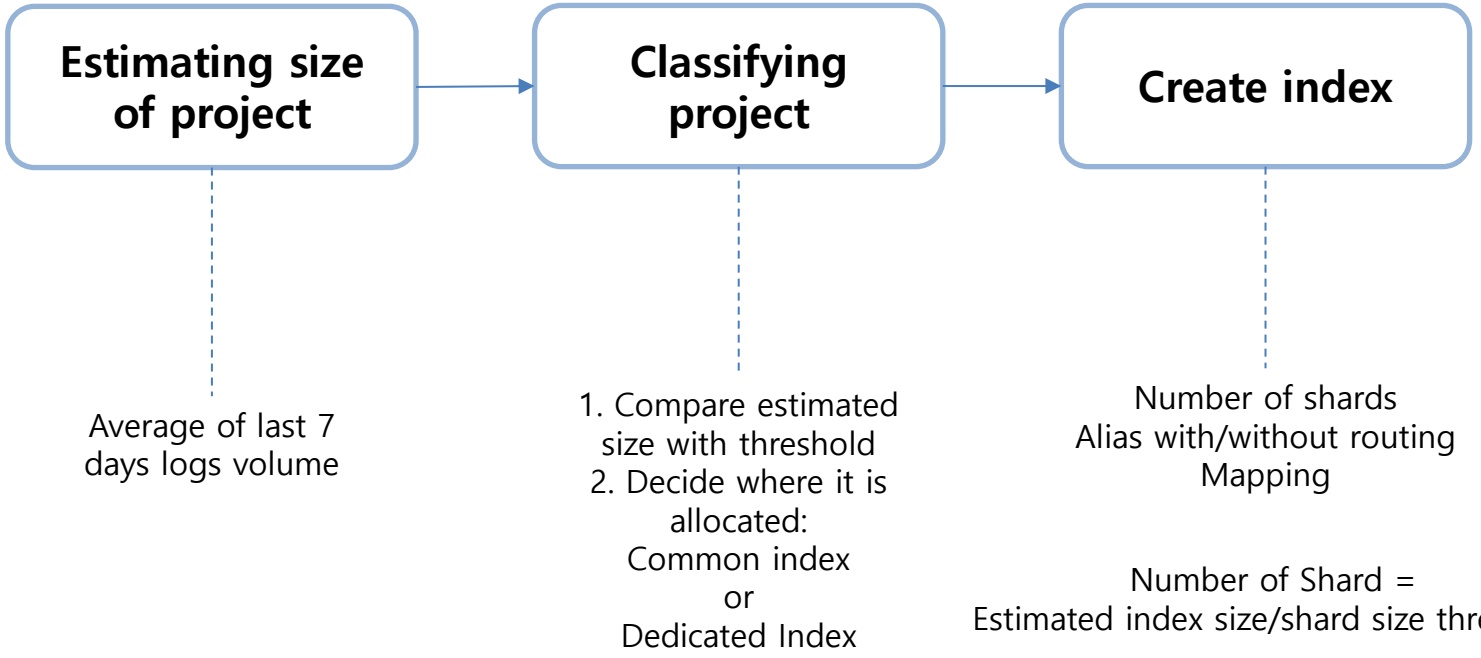
Big or Small Project?

What's the proper shard number for a index?

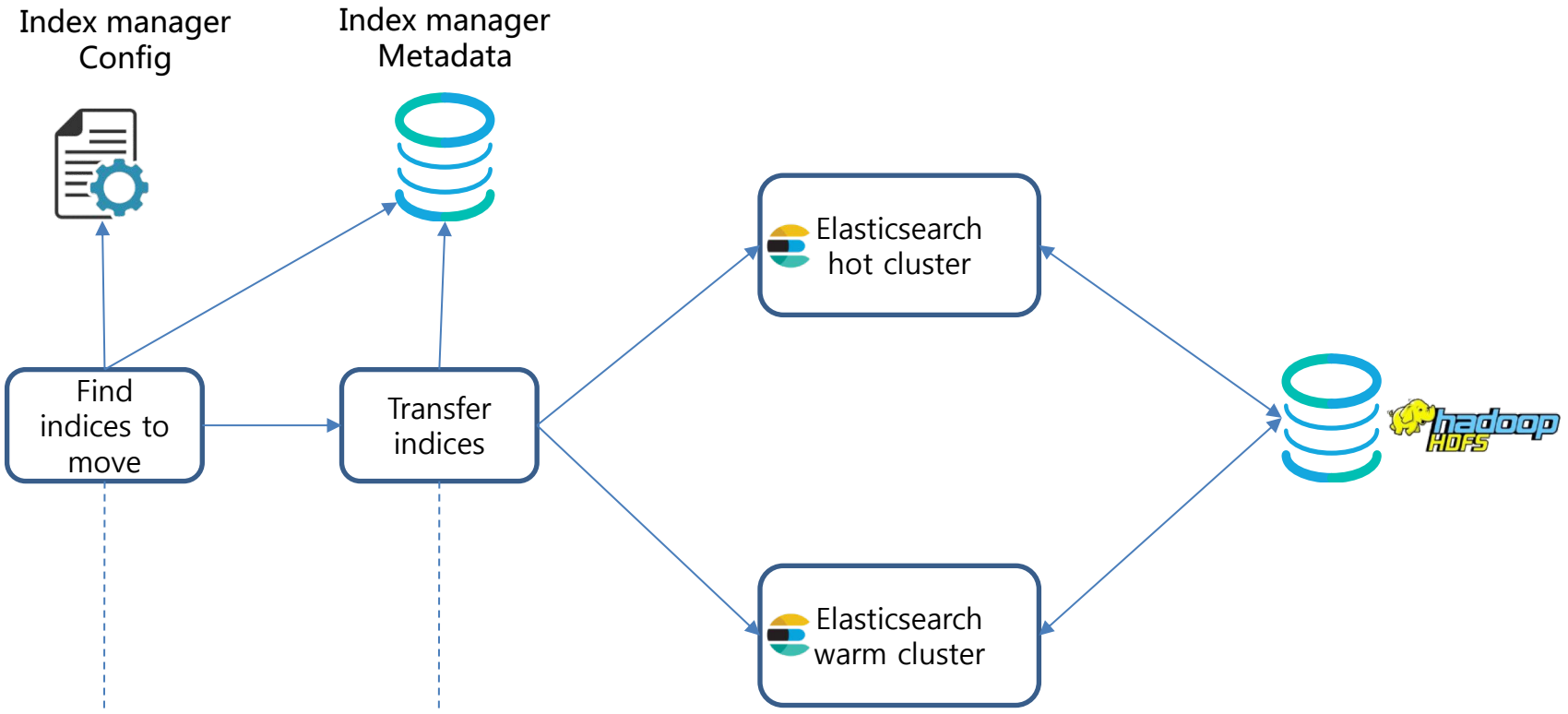
Routing or no routing for an alias in common index?

Estimated Size and Threshold

Creating Index/Alias



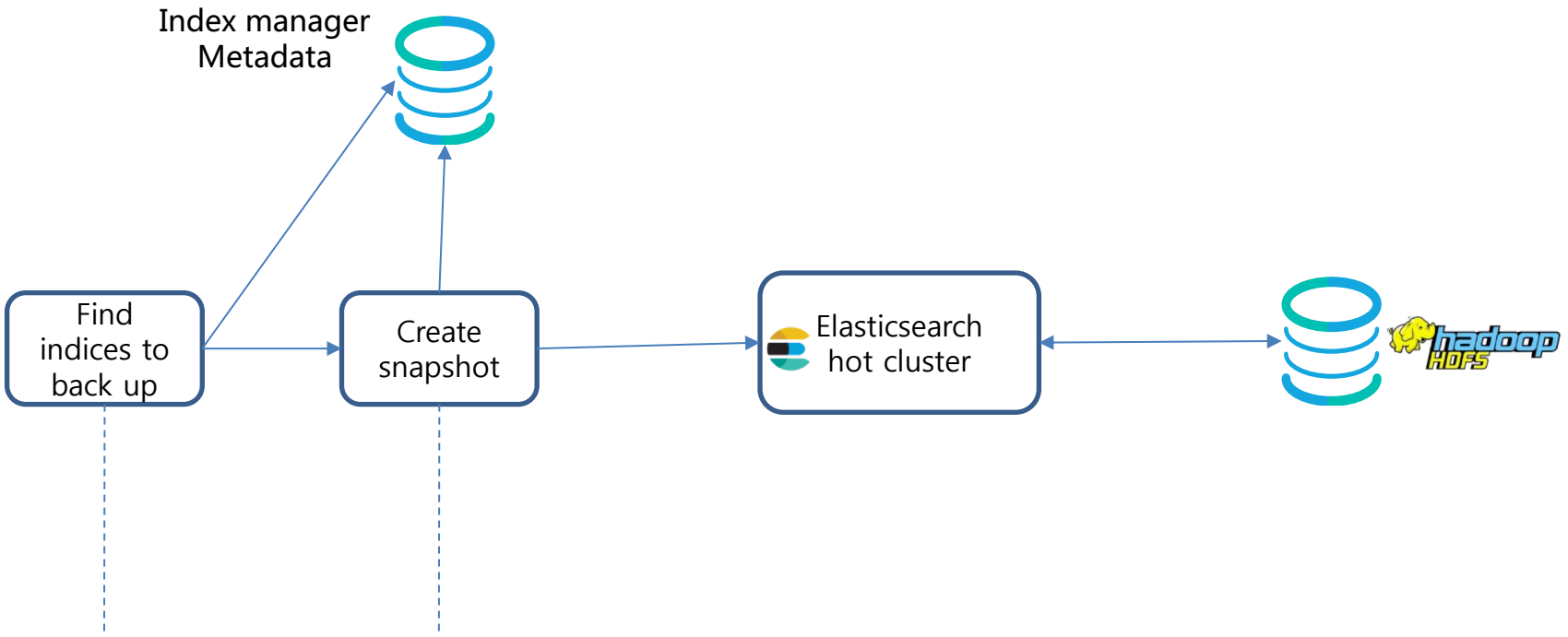
Transferring Indices



1. Check retention time of index in cluster
2. Check transferred index at last time
3. Decide which indices to move

1. Create snapshot
2. Restore index
3. Delete Index & snapshot
4. Update metadata

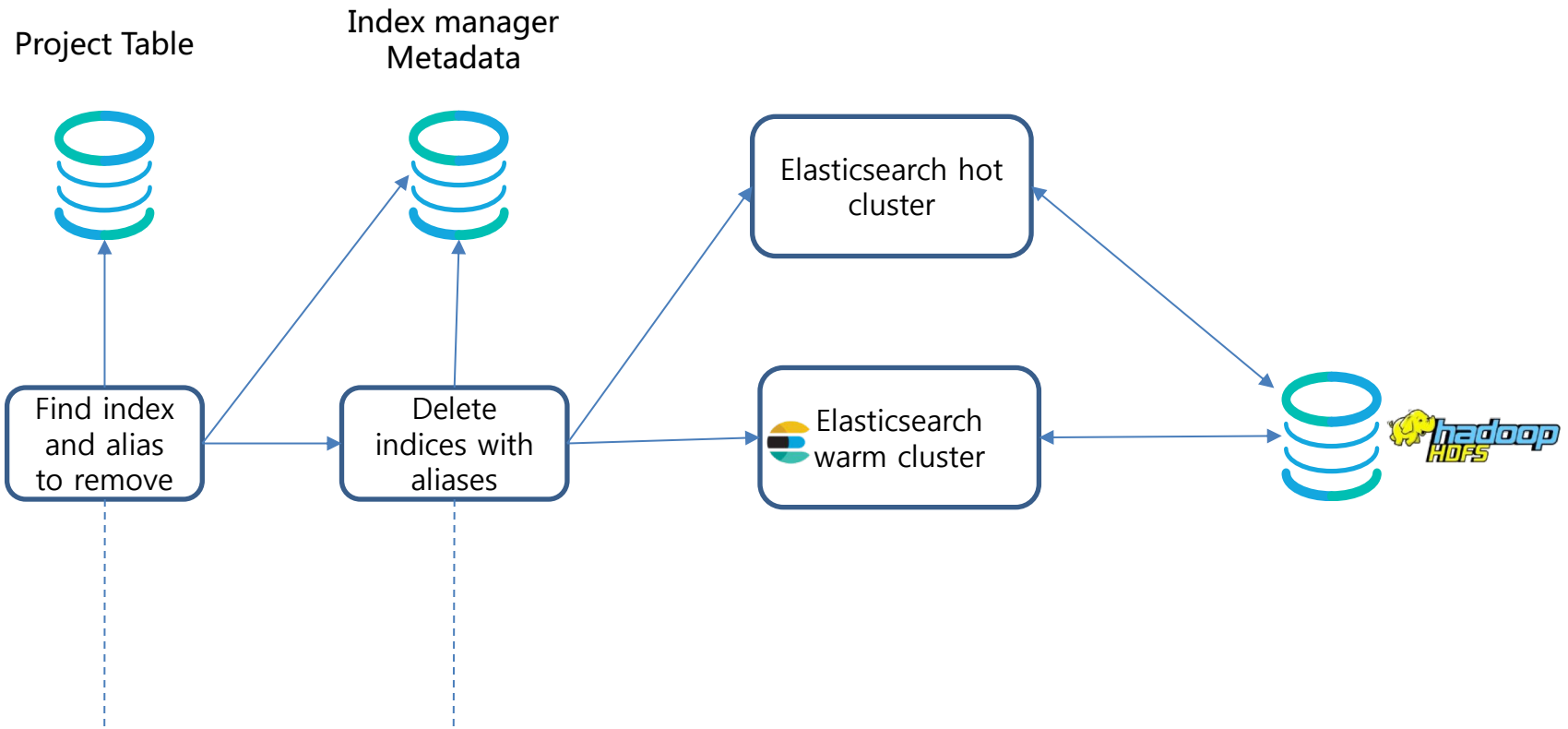
Backup



1. Check snapshotted index at last time
2. Decide which indices to backup

1. Create snapshot
2. Update metadata

Removing indices/aliases



1. Check the removed index at last time
2. Decide which indices to remove

1. Delete indices, aliases, snapshots
2. Update metadata

-

End of Document

-

Thank You.

-