
TECH
University
ebay
CHINA CENTER
OF EXCELLENCE
elastic

Elasticsearch practices at eBay

Pei Wang

Aug 2018

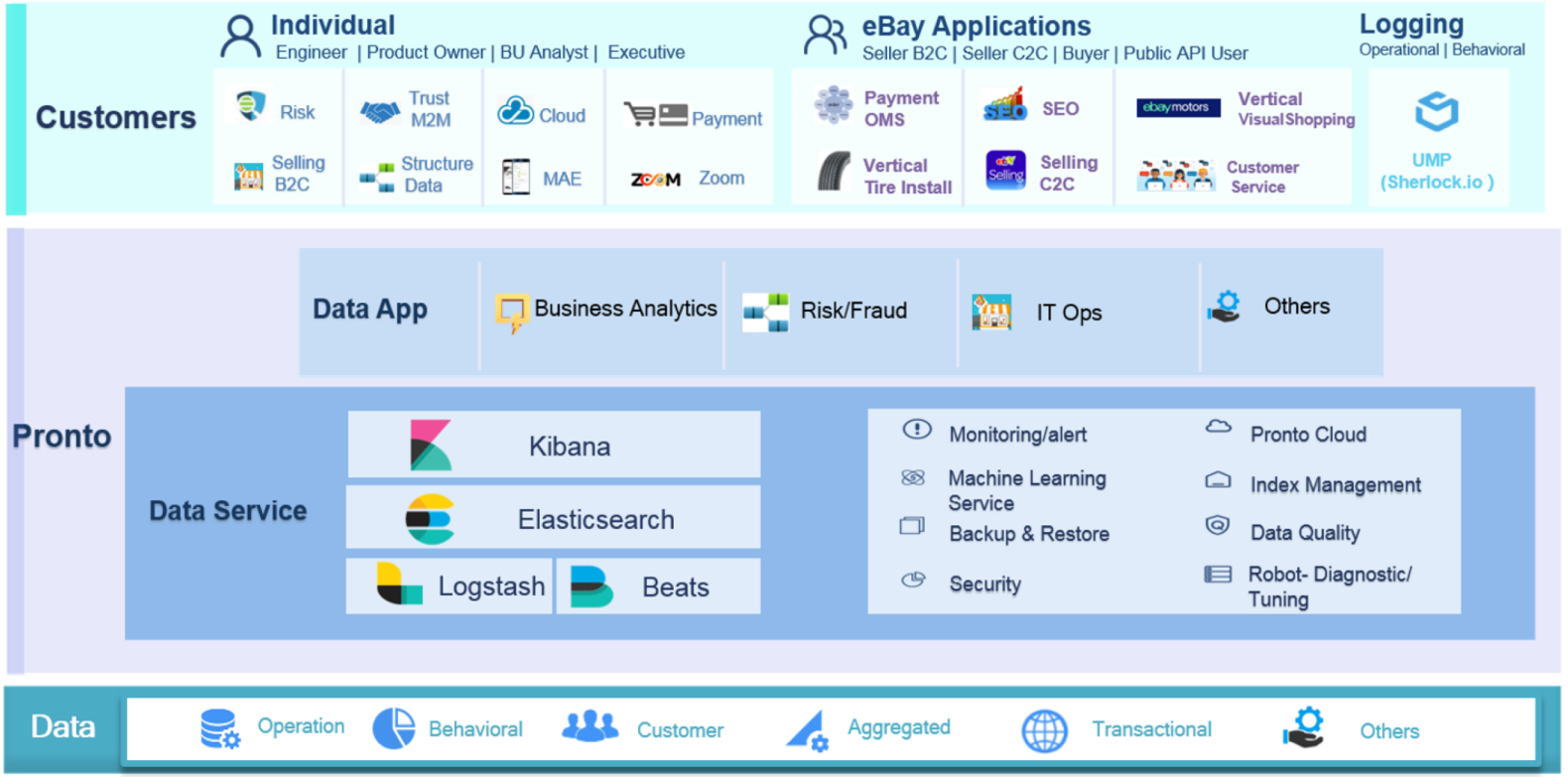
Agenda

- Elasticsearch in eBay tech stack
- Performance tuning
- Index management

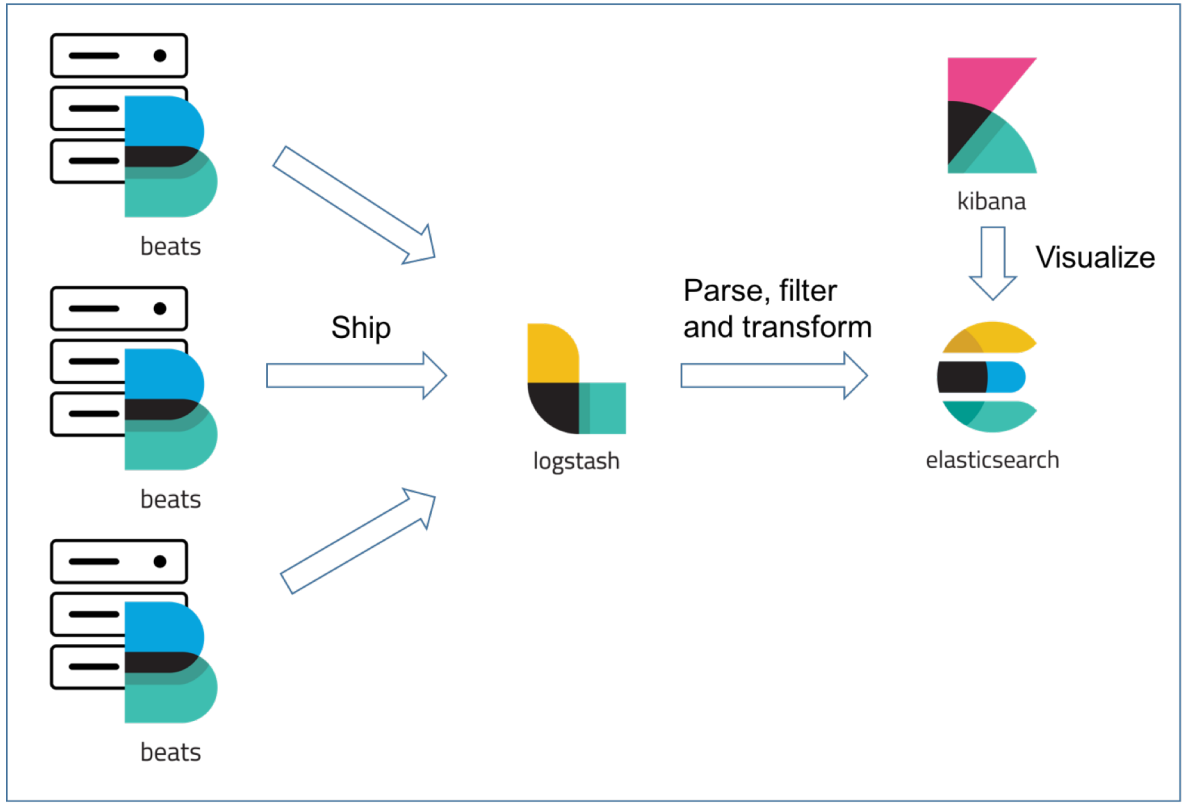
1

Elasticsearch in eBay tech stack

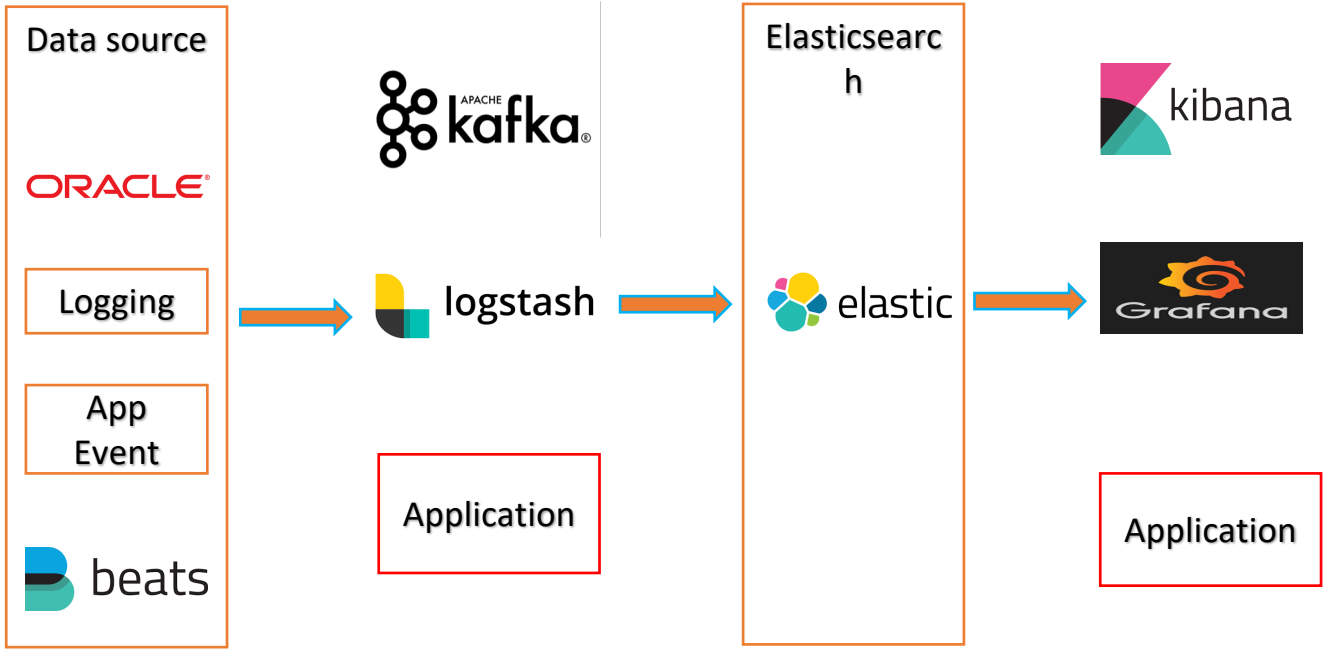
Elasticsearch in eBay tech stack



ELK + Beats


beats
beats
beats
Ship
logstash
Parse, filter
and transform
kibana
Visualize
elasticsearch

Elasticsearch in eBay tech stack



Agenda

- Elasticsearch in eBay tech stack
- Performance tuning
- Index management

2

Aggregation

Optimize Index Design

Let us consider such scenario

- there is an index which have one billion social media messages
- we have queries like below.

```
GET _search
{
  "query": {
    "bool": {
      "must": [
        { "match": { "title":   "Search"        }}
      ],
      "filter": [
        { "term":  { "region": "US" }},
        { "range": { "publish_date": { "gte": "2017-01-01" }}},
        { "match":{ "author_id": 10086}}
      ]
    }
  }
}
```

Optimize Index Design

- **Organize your index by date**
  - Cases: Logging / Monitoring / Event
- **Organize your index by field, like region**
  - Cases: Query with **enumerable** filter field
- **Use routing key**
  - Distribute docs with same routing key to same shard
  - Cases: Query with **non-enumerable** filter field

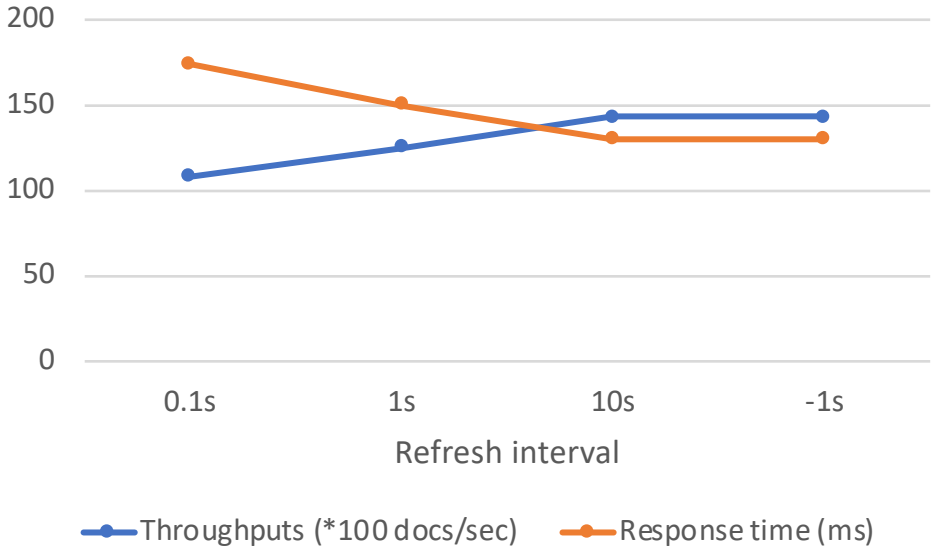
```
GET _search
{
  "query": {
    "bool": {
      "must": [
        { "match": { "title":   "Search"        }}
      ],
      "filter": [
        { "term":  { "region": "US" }},
        { "range": { "publish_date": { "gte": "2017-01-01" }}},
        { "match":{ "author_id": 10086}}
      ]
    }
  }
}
```

Optimize Index Design

- **Set mapping explicitly**
  - The default mapping may not fit your case
- **Make shards distributed evenly across nodes**
  - Nodes have more shards than others may became the bottle neck
- **Avoid imbalanced sharding if docs are indexed with routing key or user-defined ID**
  - User-defined ID should be random enough.
  - Imbalanced routing key distribution could cause imbalanced sharding.

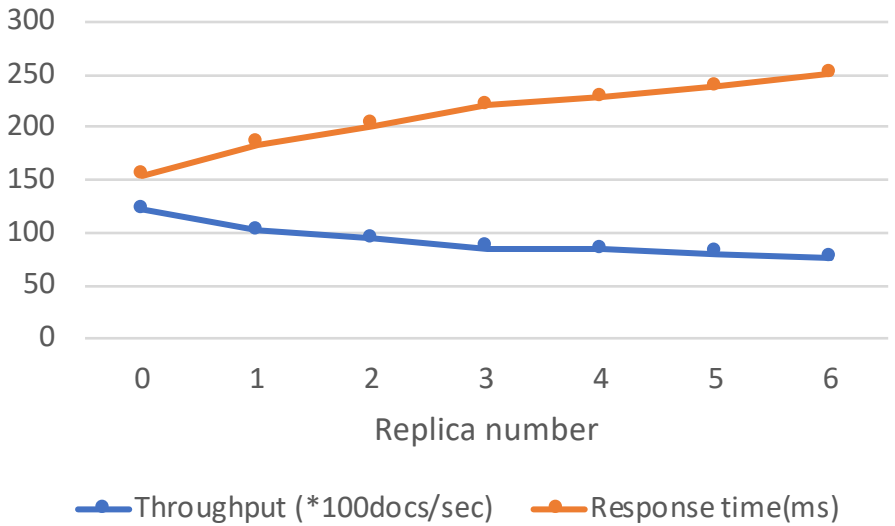
Index performance tuning

- **Increase shard number if node count > (1+ replica) * shard number**
  - Scale out, distribute data into more nodes
- **Increase refresh interval**
  - ElasticSearch create a segment every time refresh event happen. Increase interval would reduce segment count and merge cost.
  - Documents are not available for search until refresh



Index performance tuning

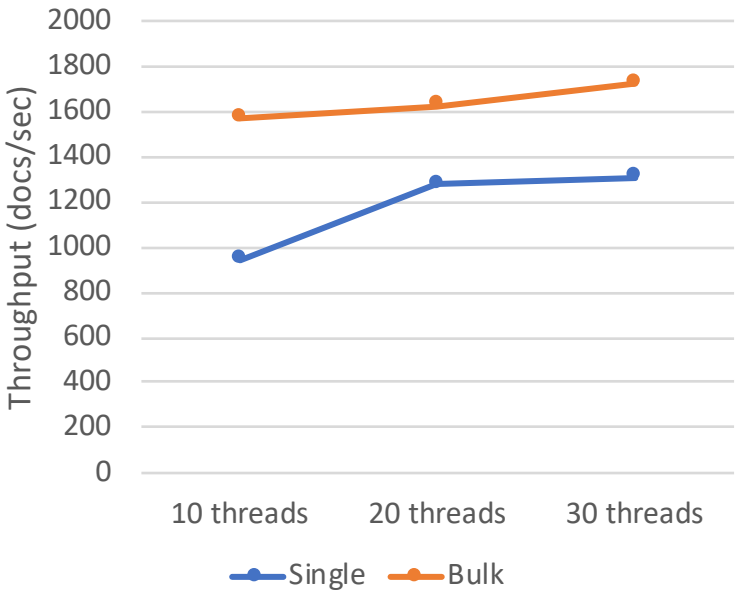
- **Use auto generated IDs if possible**
  - ElasticSearch auto generate ID algorithm can reduce the duplicate ID check and version check cost.
- **Reduce replica number**
  - ElasticSearch need to write primary shard and all replica shards for every index request
  - Replica number should not be 0, otherwise you will have data loss risk



Index performance tuning

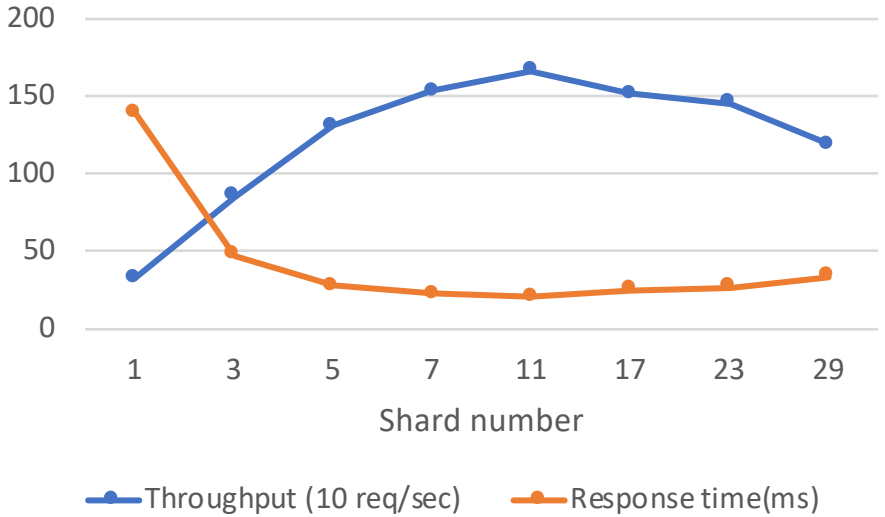
Client side

- **Use bulk request**
- **Use multiple threads/workers**



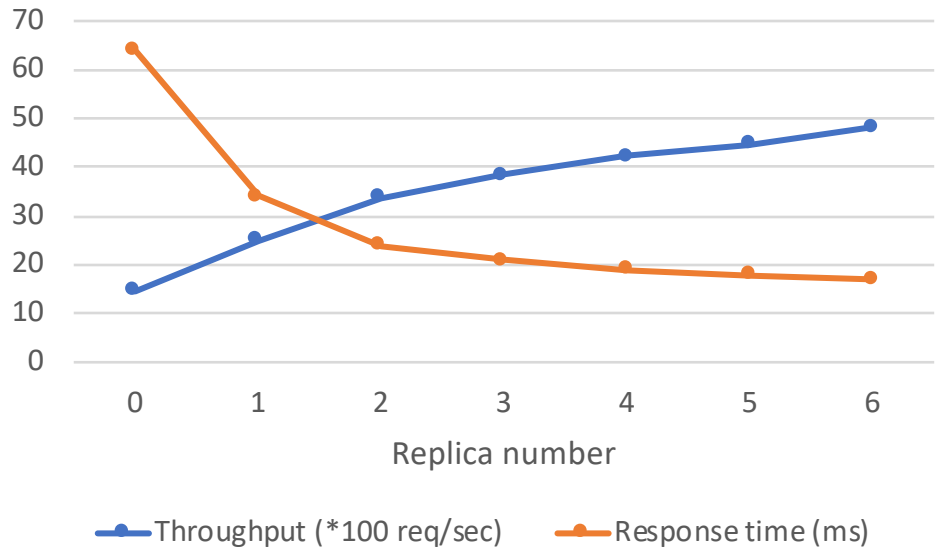
Search performance tuning

- **Choose suitable shard number**
  - Too small shard number will make search unable to scale out.
  - Too big shard number will hurt performance too.
  - Shard size should not exceed 30-50GB
  - Notes, in this test, every shard has an exclusive node.



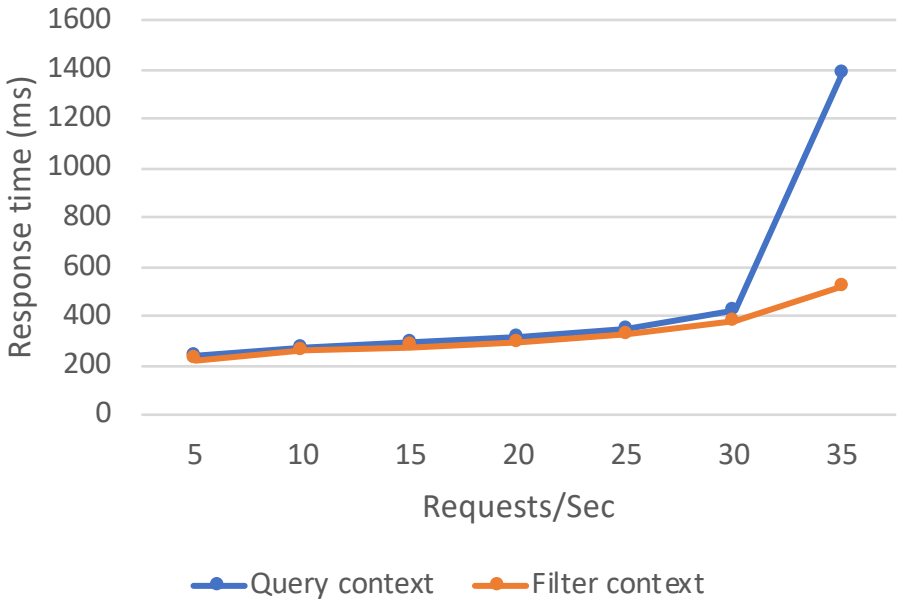
Search performance tuning

- **Increase replica number**
  - Search can be performed on either a primary or replica shard.
  - Will decrease indexing performance
  - Notes, in this test, every shard has an exclusive node.



Search performance tuning

- **Use filter context instead of query context if possible**
  - ElasticSearch do not need to calculate relevancy score for filter context
  - Filtered result can be cached



Search performance tuning

- **Node query cache**
  - Only cache queries used in filter context.
  - ElasticSearch used bit set mechanism to cache filter results.
  - ElasticSearch only enable node query cache for segments have more than 10000 or 3% documents, whichever is larger.

```
GET index_name/_stats?filter_path=indices.**.query_cache
{
  "indices": {
    "index_name": {
      "primaries": {
        "query_cache": {
          "memory_size_in_bytes": 46004616,
          "total_count": 1588886,
          "hit_count": 515001,
          "miss_count": 1073885,
          "cache_size": 630,
          "cache_count": 630,
          "evictions": 0
        }
      },
      "total": {
        "query_cache": {
          "memory_size_in_bytes": 46004616,
          "total_count": 1588886,
          "hit_count": 515001,
          "miss_count": 1073885,
          "cache_size": 630,
          "cache_count": 630,
          "evictions": 0
        }
      }
    }
  }
}
```

Search performance tuning

- **Shard request cache**
  - Only cache request size:0, like aggregate, suggestions or hit.totals.
  - Use payload body as cache key, so that the payload must be the same, even the JSON body order.
  - Do not use now or new Date() to build request body since it will make payload change every time, round your date time.
  - Cache are invalidated when refresh happen and the data in the shard has actually changed.

```
GET index_name/_stats?filter_path=indices.**.request_cache
{
  "indices": {
    "index_name": {
      "primaries": {
        "request_cache": {
          "memory_size_in_bytes": 0,
          "evictions": 0,
          "hit_count": 541,
          "miss_count": 514098
        }
      },
      "total": {
        "request_cache": {
          "memory_size_in_bytes": 0,
          "evictions": 0,
          "hit_count": 982,
          "miss_count": 947321
        }
      }
    }
  }
}
```

Search performance tuning

- **Retrieve only necessary fields**
  - Use ”_source” or “stored_fileds” to let ElasticSearch only return necessary fields
- **Reduce documents count in response if possible**

```
GET /perf_test_result/_search
{
  "size": 5,
  "_source": [
    "job_name",
    "create_at"
  ]
}
```


```
"hits": {
  "total": 12338,
  "max_score": 1,
  "hits": [
    {
      "_index": "perf_test_result",
      "_type": "REQUEST",
      "_id": "AVwvObQplyIgCso4VHVs",
      "_score": 1,
      "_source": {
        "job_name": "single_shard_ingest_test",
        "create_at": "2017-05-22T08:15:29.703Z"
      }
    },
```

Search performance tuning

- **Sort by “_doc” explicitly if do not care about the document order in response**
  - ElasticSearch use the “_score” field to sort by score as default.
  - Use “sort”: “_doc” to let ElasticSearch return hits by index order
  - Especially helps when scrolling

```
GET /perf_test_result/_search
{
  "sort":["_doc"]
}
```

Search performance tuning

- **Avoid searching stop words**
  - Stop words like “a" and "the” may cause the query hit results count to explode.
  - Use stop word filter
  - Refine query, use “the AND fox” if you really meant to search “the” word.
  - If some words are frequently used in your documents but not in the default list, use cutoff_frequency to specify which terms are stop words.

```
GET /_search
{
    "query": {
        "match" : {
            "message" : {
                "query" : "to be or not to be",
                "cutoff_frequency" : 0.001
            }
        }
    }
}
```

Search performance tuning

- **Avoid using script query to calculate hits in flight.**
  - Script query like below is quite time-consuming.
  - Should consider add extra fields in indexing phase if you have a lot of script query

```
{
  "query": {
    "script" : {
        "script" : {
          "inline": "doc['job_name.keyword'].value.startsWith('ebay')"
        }
    }
  }
}
```

```
{
  "query": {
    "match": {
      "job_name_prefix.keyword": "ebay"
    }
  }
}
```

Search performance tuning

- **Avoid wildcard query**
  - wildcard query like below is quite time-consuming, like script query, especially for leading wildcard query like “*ebay”
  - Should consider add extra fields in indexing phase if you have a lot of wildcard query

```
{
  "query": {
    "wildcard": {
      "job_name.keyword": "ebay*"
    }
  }
}
```

```
{
  "query": {
    "match": {
      "job_name_prefix.keyword": "ebay"
    }
  }
}
```

Agenda

- Elasticsearch in eBay tech stack
- Performance tuning
- Index management

3

Index management

Index Lifecycle Management Service

What it can do

- High availability index lifecycle management with user-friendly web portal
- One-stop management portal for multiple clusters and multiple tenants

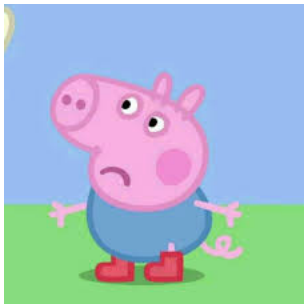
Why we build it

- Help customers manage ES data through its full lifecycle
- Let important business applications survive from peak days
- Avoid improper ES usage.

Bad design

All data in one index

10 million

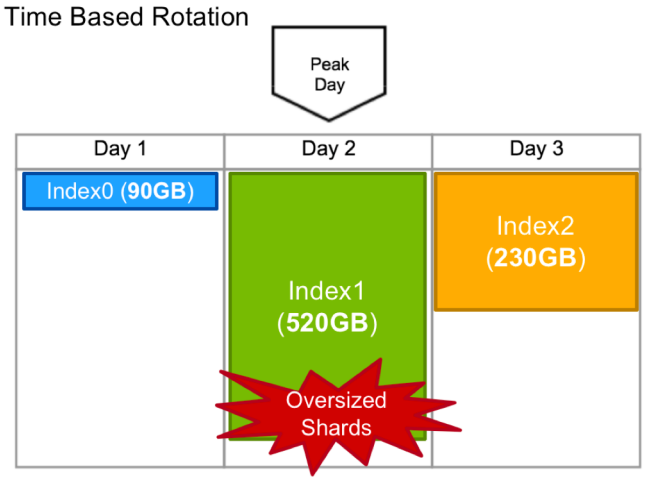


As time increase

1 billion

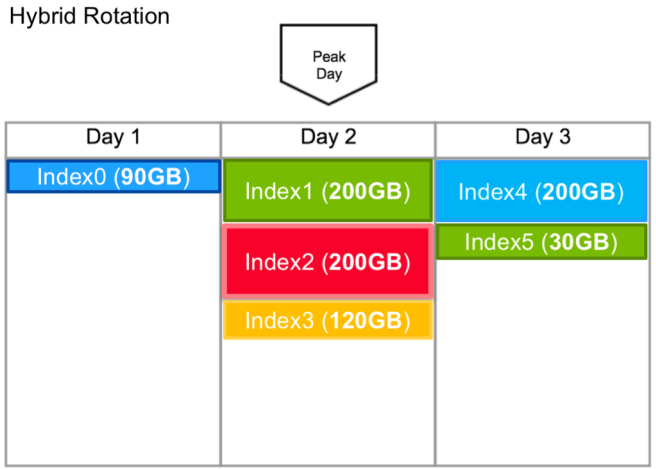
Time Based Creation

- Create new index according to timestamp automatically
- Distribute document base on timestamp
- Delete expired indices automatically



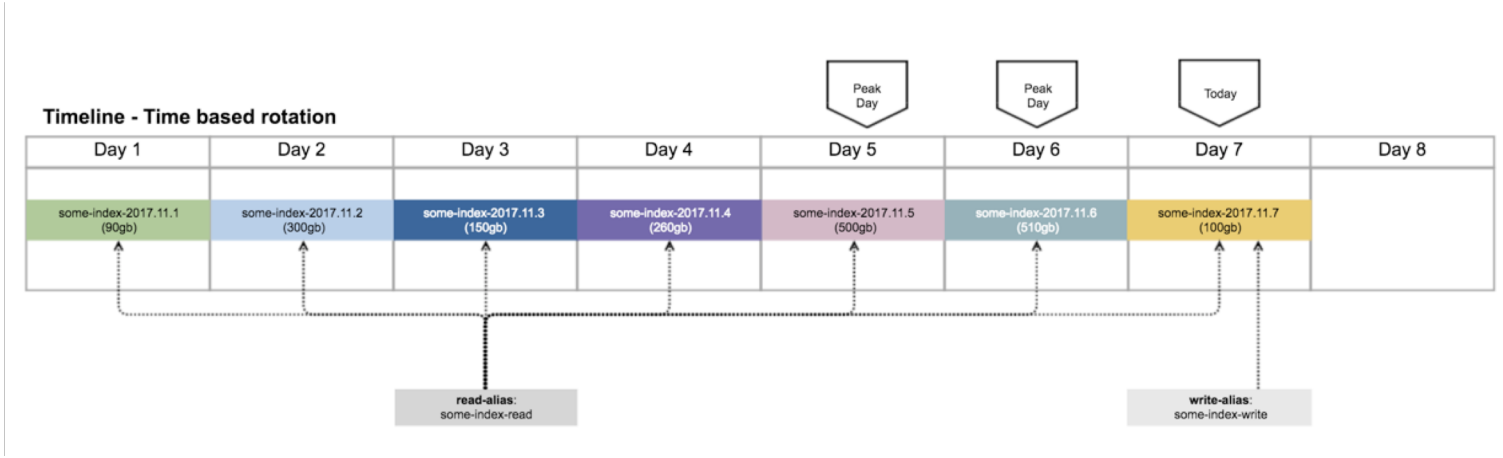
Hybrid Rotation

- Besides time rotation, Hybrid solution take disk space into consideration
- Create and manage indices automatically



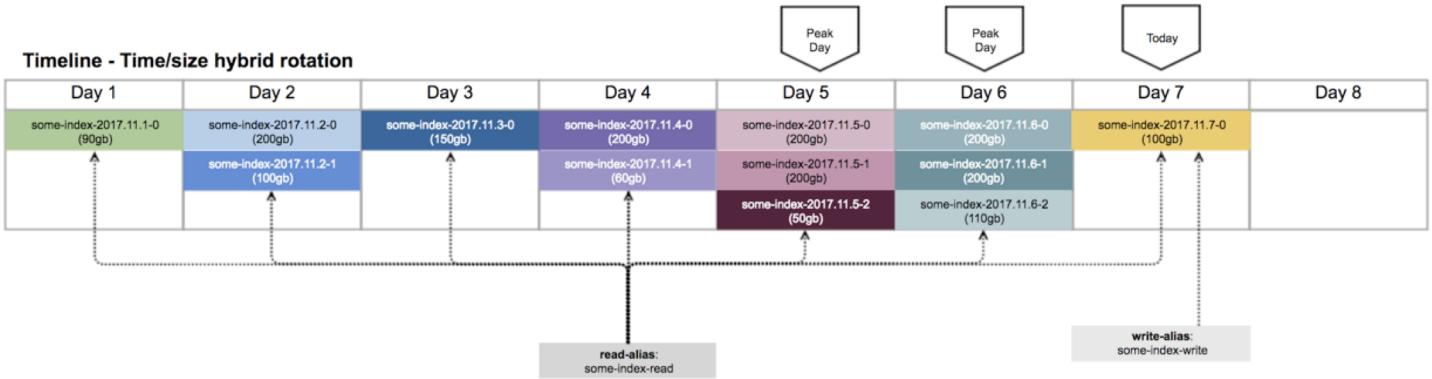
Index alias

- Minimize customer's effort
- Use Index alias to route user request to correct indices

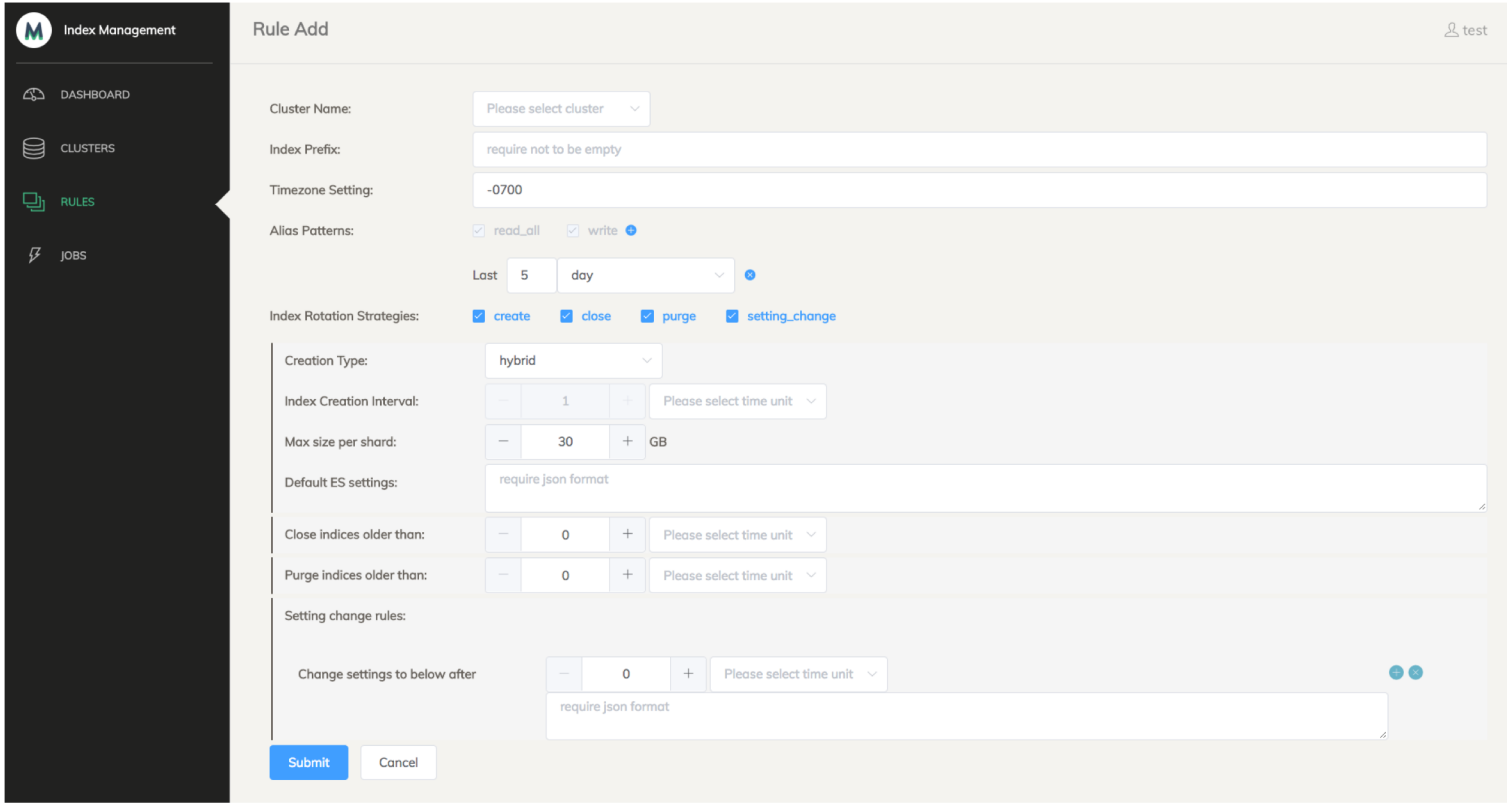


Index alias

- Minimize customer's effort
- Use Index alias to route user request to correct indices



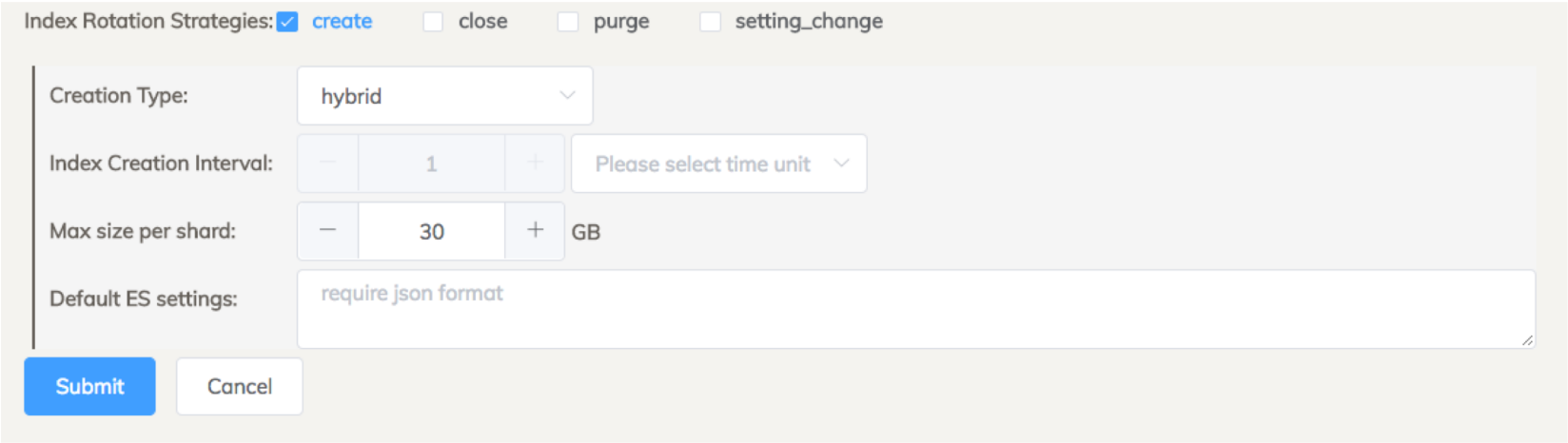
Rule Overview



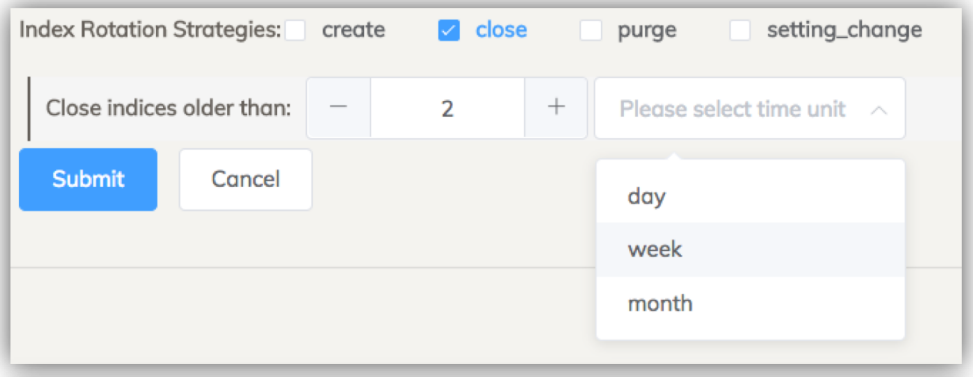
ebay CHINA CENTER OF EXCELLENCE

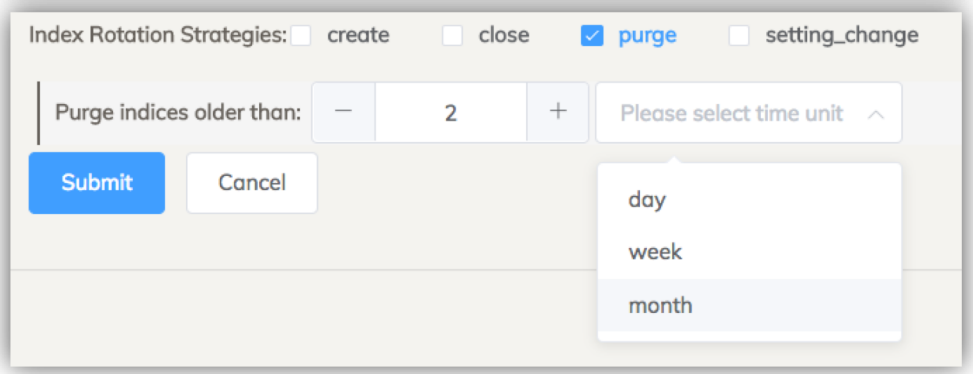
TECH University

Action: Index Creation

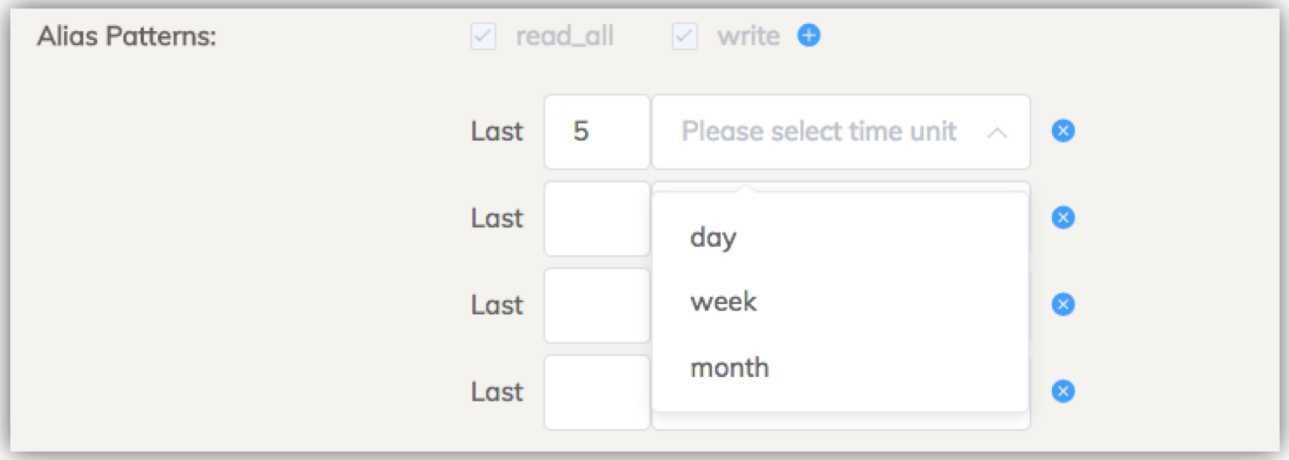


Action: Index Close and Delete

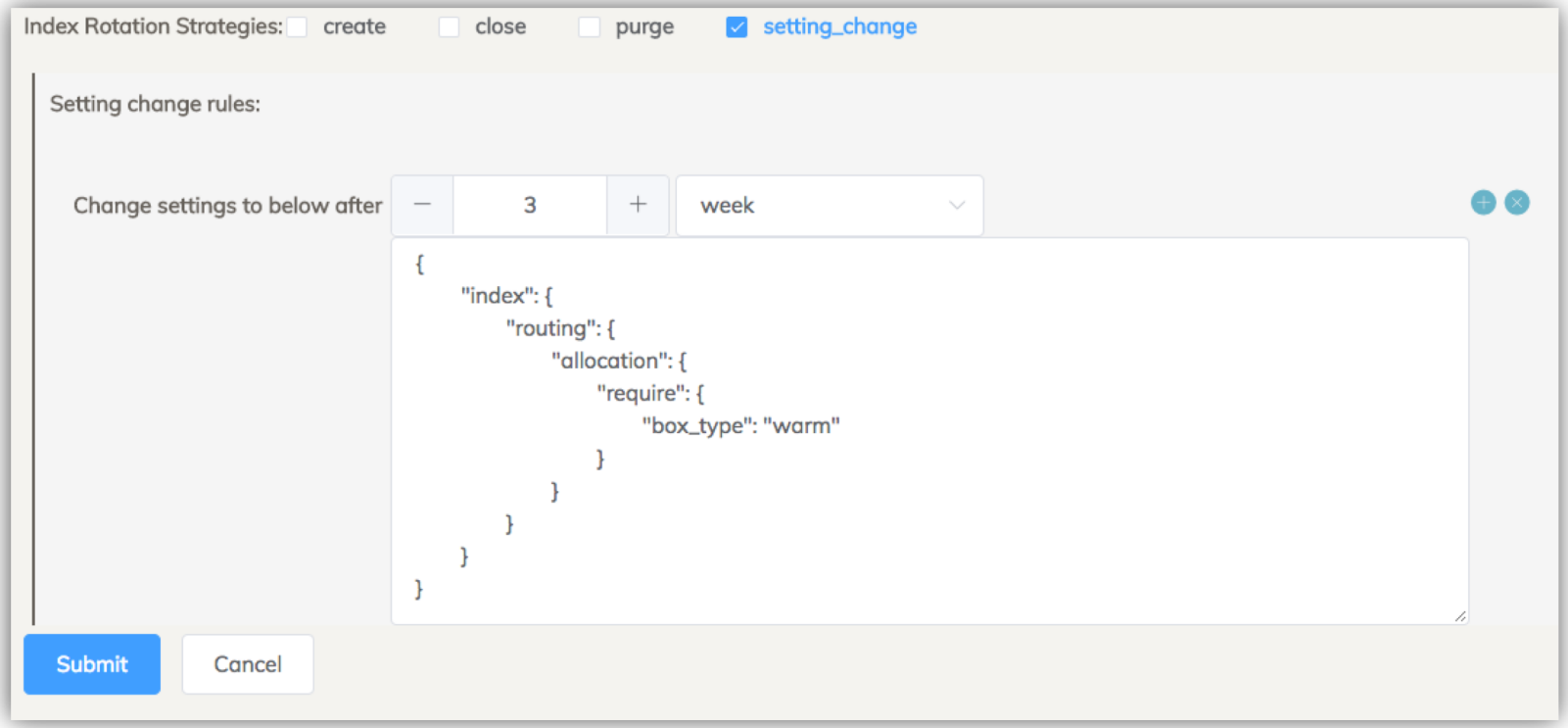




Alias Rotation



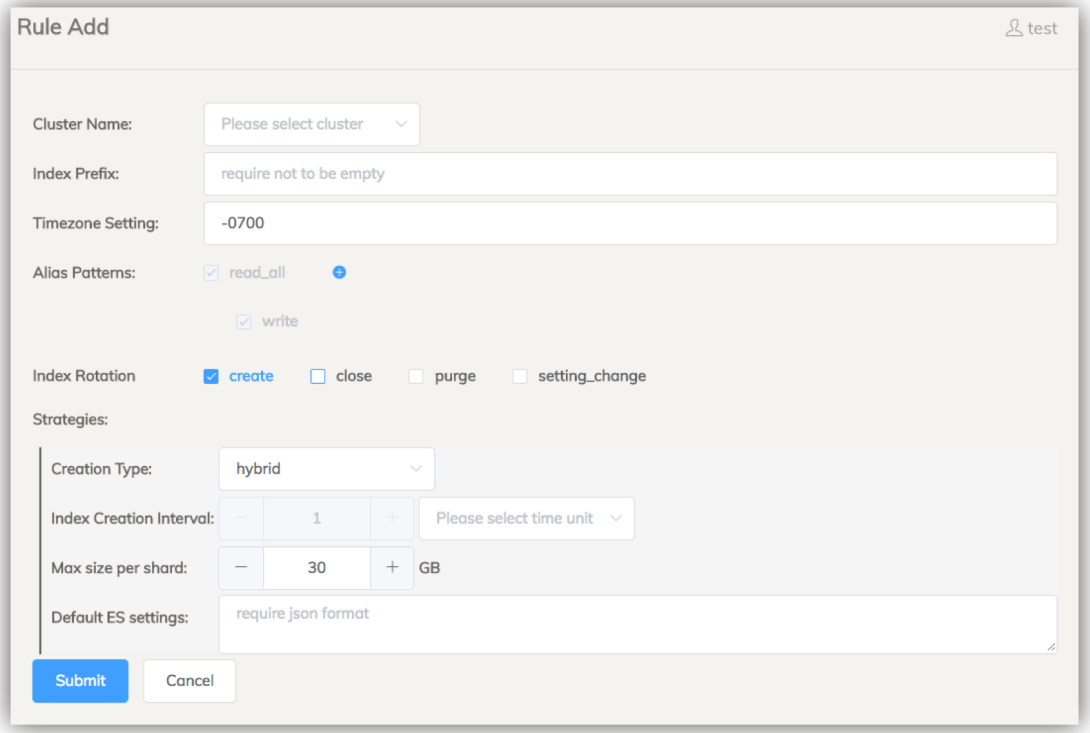
Action: Index Settings



Feature Summary

Business Specifications

- Index Creation
  - Condition :Time, Size, Shard-Size
  - Supported time unit : hour, day, week, month
  - Supported size unit : GB
- Index Close
- Index Deletion
- Index Setting Change
  - Condition :Time
  - Supported time unit : hour, day, week, month
  - Use case: hot-warm, replica number, etc.
- Alias Rotation



Thanks

Wang, Pei

Dai, Catherine

Hu, Hongsi