hawksearch / FORUM

Elasticsearch - Big Brother of Lucene

Distributed Search Execution

Introductions

Shreyas Kamath
Chief Technology Officer

Mitch Kresca
Director of Implementations

Agenda

01. Overview

02. Query Phase

03. Fetch Phase

04. Search Options

05. Scroll

Overview

01.

Overview

- Quick Introduction to Hawksearch with Elasticsearch
- Elasticsearch is a distributed, JSON-based search and analytics engine designed for horizontal scalability, maximum reliability, and easy management.
- Apache Lucene™ is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.

Overview

Elasticsearch provides us a convenient layer over Lucene. Each shard in Elasticsearch is a separate Lucene instance. To summarize:

- Elasticsearch is built over Lucene and provides a JSON based REST API to refer to Lucene features.
- Elasticsearch provides a distributed system on top of Lucene. A distributed system is not something Lucene is aware of or built for. Elasticsearch provides this abstraction of distributed structure.
- Elasticsearch provides other supporting features like thread-pool, queues, node/cluster monitoring API, data monitoring API, Cluster management, etc.

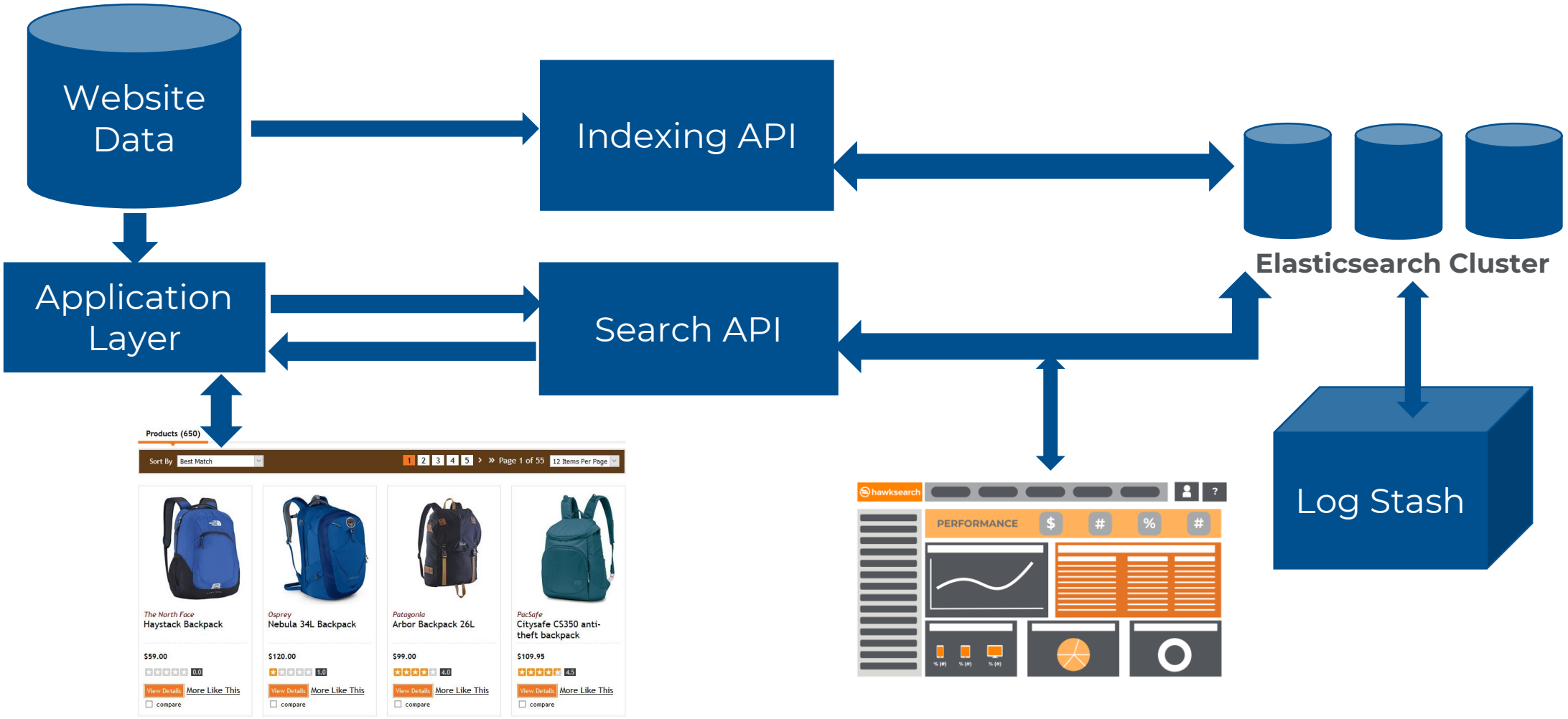
Distributed Searching and Indexing is necessary for

- Performance
- Redundancy
- Handle complex query model and volume of data

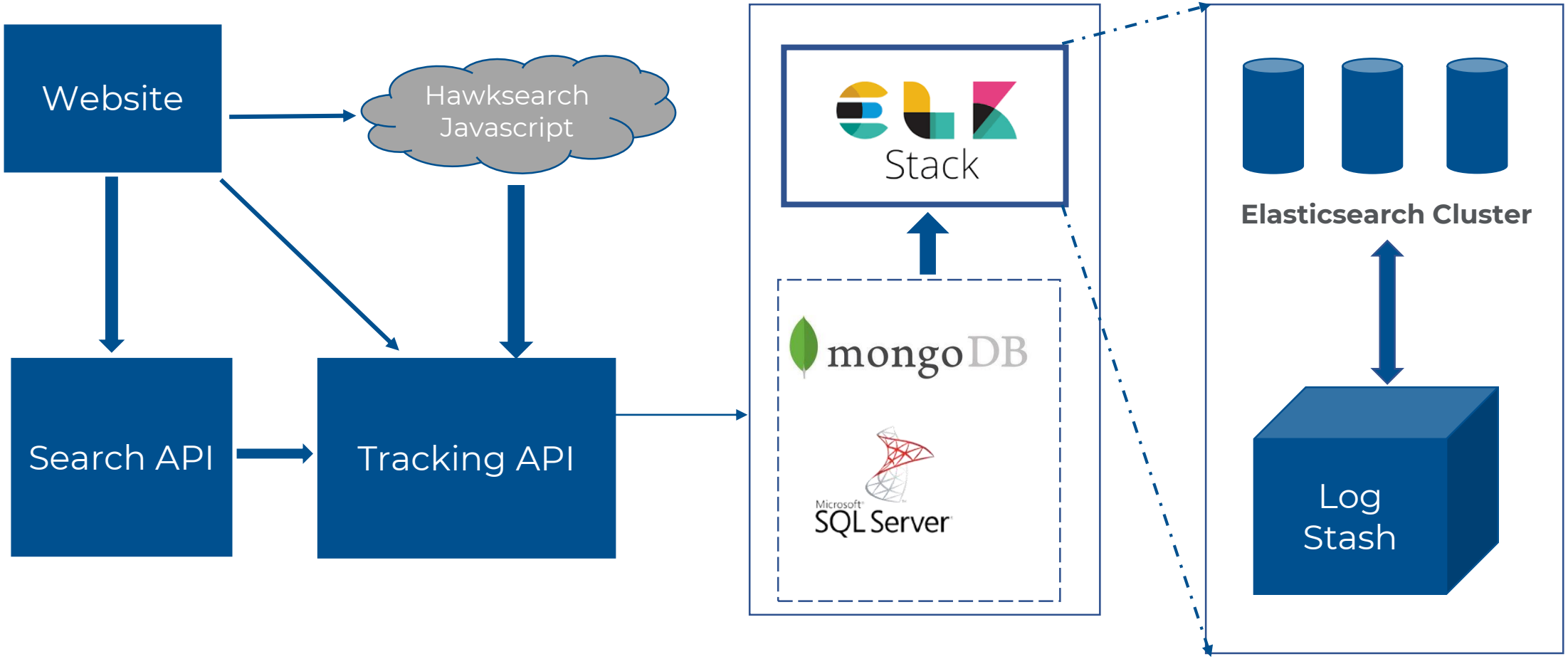
Architecture Overview

02.

Hawksearch with Elasticsearch



Hawksearch Reporting Stack



Benefits

03.

Benefits

Document-oriented
Elasticsearch is document-oriented. It stores real world complex entities as structured JSON documents and indexes all fields by default, with a higher performance result.

Speed
Elasticsearch is able to execute complex queries extremely fast. It caches almost all of the structured queries commonly used as a filter for the result set and executes them only once. This saves the time parsing and executing the query improving the speed.

Scalability
Elasticsearch is distributed by nature and can easily scale horizontally providing the ability to extend resources and balance the loading between the nodes in a cluster.

Structured search
Elasticsearch is schema free, it accepts JSON documents, as well as tries to detect the data structure, index the data, and make it searchable.

Benefits

Data record
Elasticsearch records any changes made in transactions logs on multiple nodes in the cluster to minimize the chance of data loss.

Query Fine Tuning
Elasticsearch has a powerful JSON-based DSL, which allows for constructing complex queries and fine tuning them to receive the precise results from a search. It provides also a way of ranking and grouping results.

Restful API
Elasticsearch is API driven, actions can be performed using a simple Restful API.

Distributed approach
Indices can be divided into shards, with each shard able to have any number of replicas. Routing and rebalancing operations are done automatically when new documents are added.

Multi-Tenancy
Handling documents that require access controls for multiple users

Indexing API

04.

Lucene vs Elastic

elasticsearch	Lucene
• Data Feeds are optional • Partial Updates are optional and are performed in real-time VIA and API • Up to “X” previous copies of the index are stored and can be switched VIA the API • Can create inactive indexes • Can clone an index – Launching in Q2	• Data Feeds are required • Partial Updates are optional but still take time to merge into the index and replicate out to the web servers • Two Versions of index Stored

Create New Index

- POST - api/indexing/create
- Optional Suffix

```
{
    "Suffix": "test-addingcolorfamily"
}
```

- Response

```
{
  "IndexName": "myengine.20190123.153931.test-addingcolorfamily"
}
```

Index New Items

- POST - api/indexing/index-items
- {

"IndexName": "myengine.20190123.075859",

"Items": [

{

"id": ["123"],

"Sku": ["987654"],

"title": ["Rain Jacket"],

"price": ["39.99"],

"saleprice": ["32.99"], "image":["http://www.mysite.com/images/Thumbnail/987654.jpg"],

"url_detail":["http://www.mysite.com/jackets/mens/987654"],

"brand" : ["Acme"],

"color" : ["Orange"]

}]

}

Getting Current Index

- POST - api/indexing/get-current

```
{
    "IndexName": "myengine.20190123.074248"
}
```

Setting Current Index

- POST - api/indexing/set-current

```
{
    "IndexName": "myengine.20190123.074248"
}
```

Updating Current Items

POST - api/indexing/index-items

- Body is the same as when indexing a new item
- If the item does not exist it will be added
- If the item exists it will be updated
- Single attributes updates are possible
- Cloning an index

Delete Items

- POST - api/indexing/delete-items

```
{
  "IndexName": "myengine.20190123.131548",
  "Ids": ["Item_72452", "Item_72655"]
}
```

Delete Index

- POST - api/indexing/delete-index

```
{
    "IndexName": "myengine.20190123.090452"
}
```

- If the current index is the index that you are trying to delete an error will be thrown.

Search API

05.

Lucene vs Elastic

elasticsearch	Lucene
• Method – POST • api/search, api/autocomplete • JSON in Body • Response is JSON Based • Designed to be implemented with a Full API integration, or through a JS integration using React or Angular JS	• Method – GET • /sites/%%clientname%% • Parameters in the URL • Response can be HTML, JSON, XML, or a combination of any of them.

Querying – Search Request


```
{
   "ClientGuid" : "dfb51ea2ea1a42e4a53d4e67f7f6847b",
   "Keyword" : "jack",
   "FieldOverride" : ["itemname", "isonsale"],
   "SortBy" : "saleprice",
   "FacetSelections": {
      "pricerange": ["1,25"],
      "Brand": [
                  "Columbia Sportswear",
                  "Patagonia"
      ],
      "zip_postal_code_range": ["ba3,bl6"]
   },
   "ClientData":{
                  "VisitorId" : "2F87556F-AA2F-438E-A52C-AFF4B7E10EB5",
                  "Custom" : {"some key" : "some value"},
                  "HttpTrueClientIp" : "68.72.70.2",
                  "UserAgent" : "some agent",
                  "Source" : ""
   }
}
```

Querying – Landing Page Request


```
{
   "ClientGuid" : "dfb51ea2ea1a42e4a53d4e67f7f6847b",
   "Custom URL" : "/product-test-elastic",
          "ClientData":{
          "VisitorId" : "2F87556F-AA2F-438E-A52C-AFF4B7E10EB5",
          "Custom" : {"some key" : "some value"},
          "HttpTrueClientIp" : "68.72.70.2",
          "UserAgent" : "some agent",
  "Source" : ""
          }
}
```

Querying – Response Differences


```
{
        "DocId": "Item_119547",
        "Score": 775.7318,
        "Document": {
                "image": [
                        Womens-Trabagon-Rain-Jacket-Watermelon.jpg"
                ],
                "itemname": [
                        "Women's Trabagon Rain Jacket"
                ]
        }
}
```

Querying – Response Differences

Facets Object


```
"Values": [
        {
            "Label": "Columbia Sportswear",
            "Value": "Columbia Sportswear",
            "Count": 39,
            "Selected": false
        }
]
```

Querying – Autocomplete Request


```
{
   "ClientGuid" : "dfb51ea2ea1a42e4a53d4e67f7f6847b",
   "Keyword" : "jack",
   "Type" : "Product",
   "ProductCount" : "2",
   "DisplayFullResponse" : "true",
   "FieldOverride" : ["itemname", "isonsale"],
   "ClientData":{
            "VisitorId" : "2F87556F-AA2F-438E-A52C-AFF4B7E10EB5",
            "Custom" : {"some key" : "some value"},
            "HttpTrueClientIp" : "68.72.70.2",
            "UserAgent" : "some agent",
      "Source" : ""
   }
}
```

Querying – Autocomplete Response


```
"Document": {
        "itemname": [
            "Jack Rabbit SL2 Footprint"
        ],
        "isonsale": [
            "Yes"
        ]
}
```

Querying – Autocomplete Response

"Categories": [
{
"Value": "Women » Jackets",
"Url": "http://demo.hawksearch.net?department_nest=Jackets_6"
},
",
"Popular": [
{
"Value": "jacket",
"Url": "http://demo.hawksearch.net?keyword=jacket"
}

Distributed Querying

06.

The Problem

- Documents can exist on any shard in the cluster.
- Fetching the documents is only half the story.
- Lucene’s single index approach doesn’t have this issue

The Elasticsearch Solution

- Two Phased
  - Query Phase – Finding all the Matching Documents and creating a Single Sorted List of Documents
  - Fetch Phase- Enriching the paginated results with the field data in the response

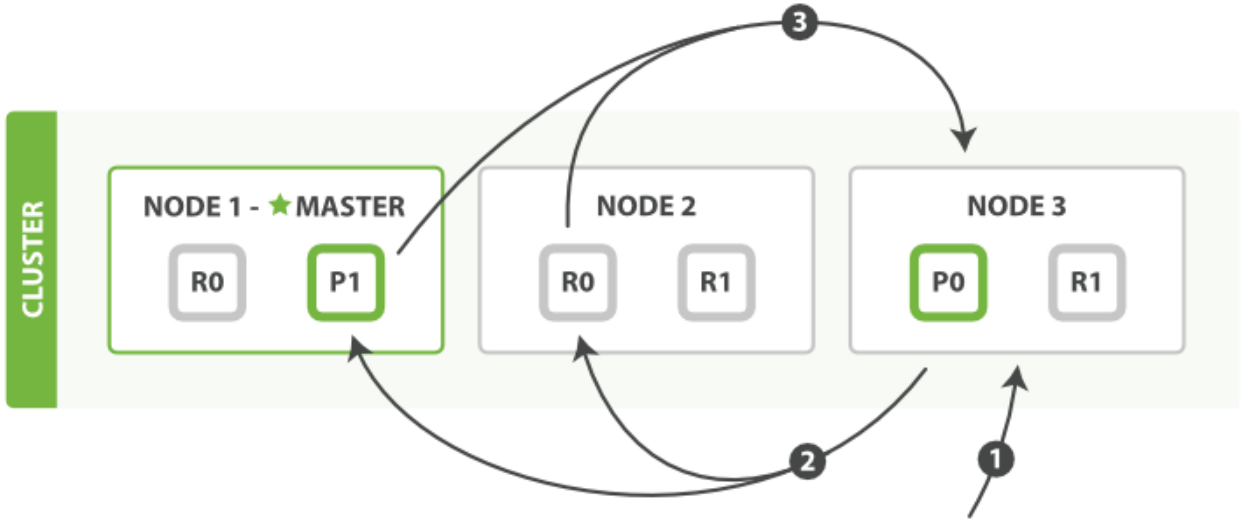
Query Phase

- Query is broadcast to a shard copy of every shard in the index
- Builds a sorted list that holds the matching documents
- Priority List

Query Phase

1. Request is sent from the master to Node 3
   - Node 3 becomes the coordinating node and creates empty priority list

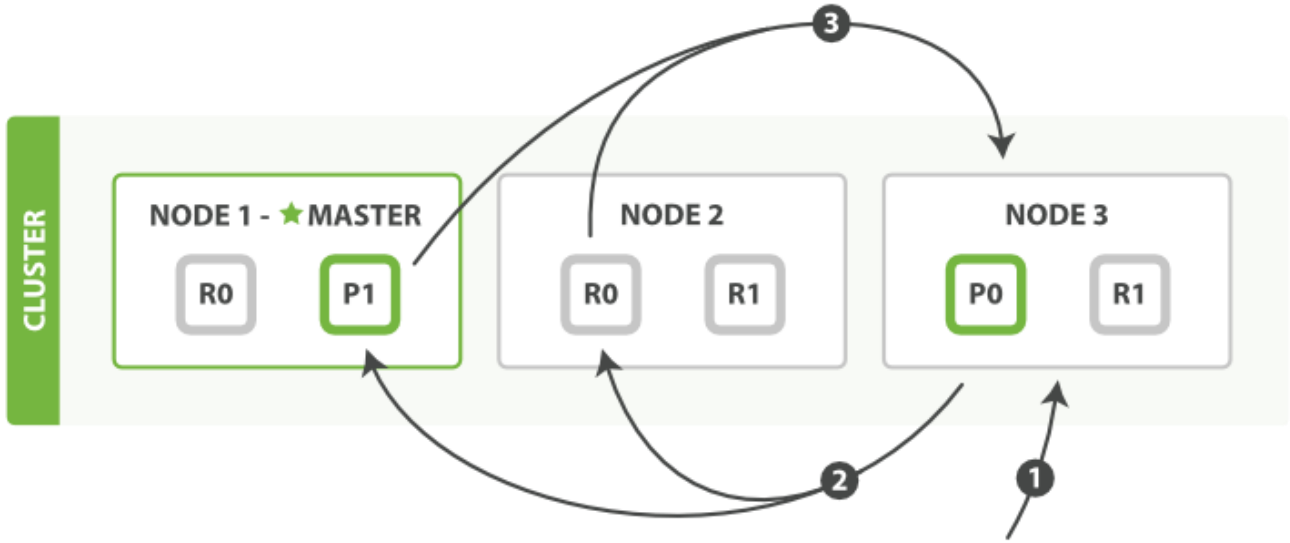
2. Node 3 then forwards the request to a copy of every shard in the index – Node 2



Query Phase

3. Each shard returns a priority queue of doc IDs to the coordinating node – Node 3

4. Node 3 merges all results into its own priority queue to create a global priority queue.

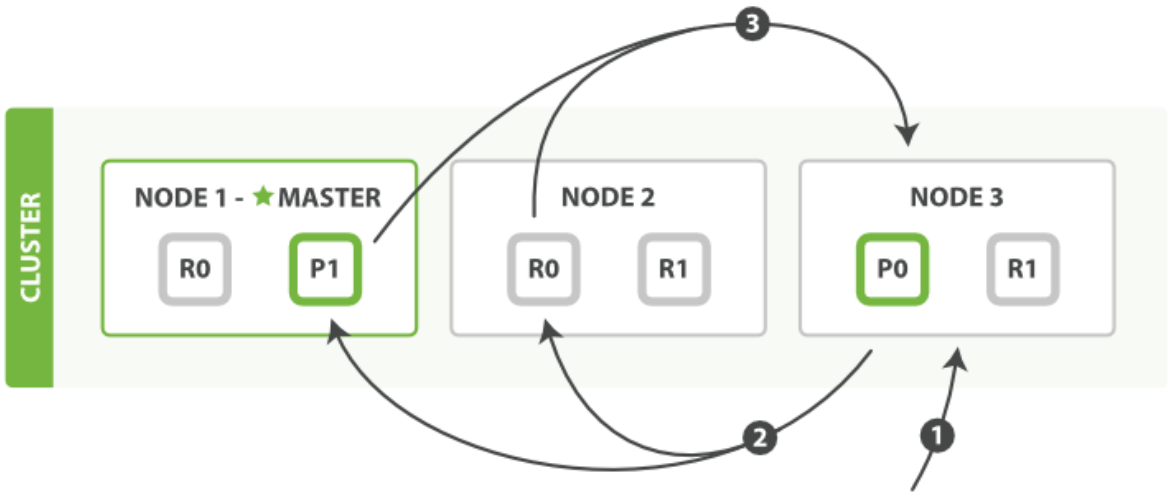


Fetch Phase

- Query Phase identifies and sorts the documents
- Fetch Phase retrieves the documents

Fetch Phase

1. Coordinating Node (Node 3) identifies which documents need to be fetched
   - Query Phase identifies all documents that match
   - Fetch Phase only needs to query the items that need to be returned in the response

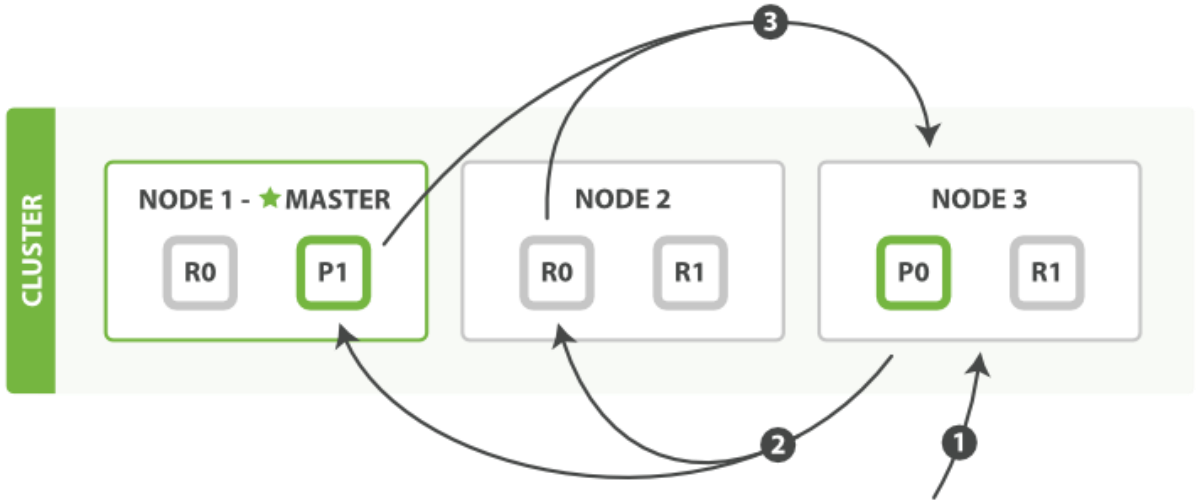


Fetch Phase

2. Node 3 issues multi request to shards in index.

3. Each shard loads the fields for each document in the local hard and enriches them

4. Coordinating node assembles them in the correct order and sends the response back to the client



Performance Benefits

- Indexes are stored on Disk
  - Finding hits is limited to the IO speed on a single server
  - Distributed Queries utilize IO Speed of multiple nodes in the cluster
- Communication with Shards is through the network, allowing some of the limited IO load to be transferred to the network.

Upgrade Process

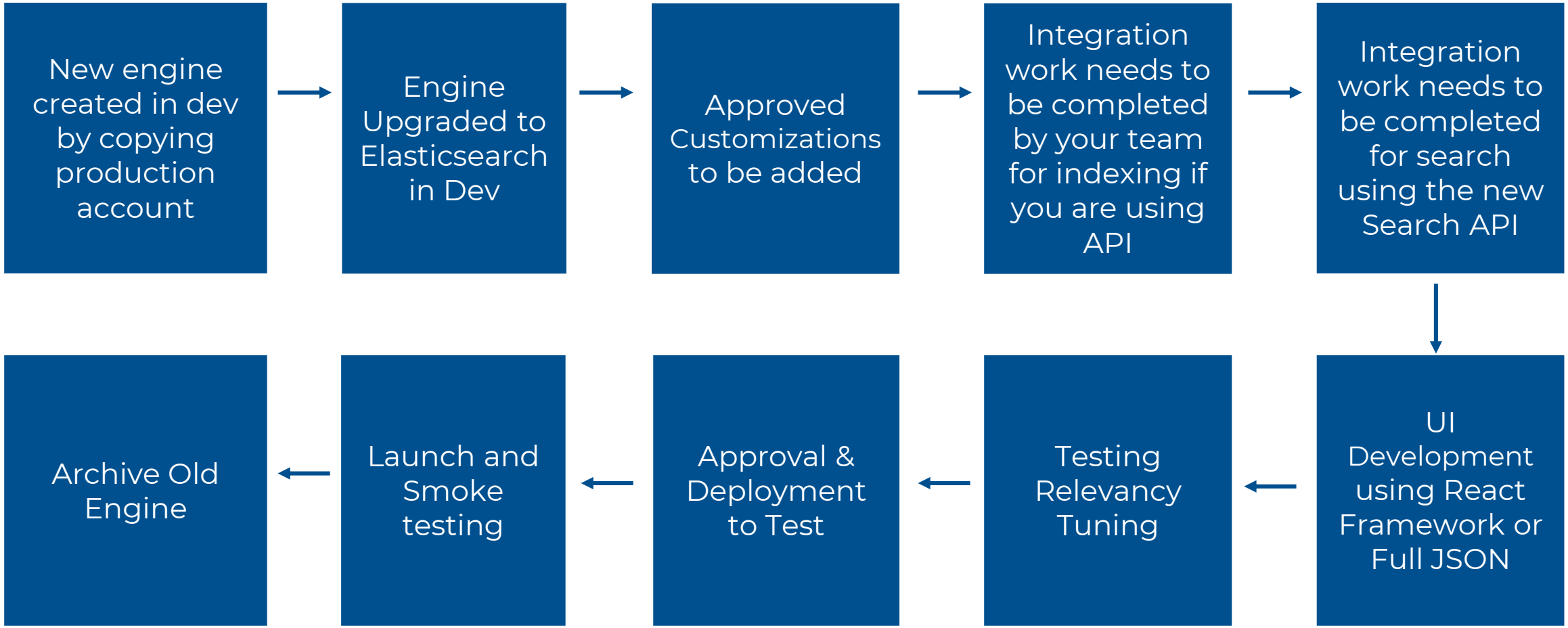
07.

Upgrade Process - What to Know before you upgrade

"Base Upgrade for Elasticsearch is free and included as part of the SaaS monthlies. The customizations are what we scope for"

- Hawksearch will provide you a list of all customizations on your site outside of what is covered by the SaaS product
- Review API documentation for indexing and Search
- Review provided functionality and confirm if any of the requirements are outdated or can be handled by the new tools Hawksearch offers
- Get on a scoping call with the team to identify and talk through the customizations so final estimate for the customizations can be provided
- Decide if you want to choose the API for indexing or continue to provide the feeds
- Agree on the scope and sign off. If there are no customizations you will be just signing off on the base proposal to proceed with upgrade at a "0" $ estimate

Upgrade Process



Any Questions?

Thank You!

Questions?
Contact your account manager or email marketing@hawksearch.com

#HawksearchForum