searchgoose

Team 9
201411296 이선명
201411312 장하나
201711375 권혁규
201711413 이유진

2차구현

Index

1. Overview

- Searchgoose는 NoSQL 기반의 full text search를 지원하는 RESTful 분산 검색 엔진.
- Searchgoose는 문자열 검색과 분산(sharding)이 어렵다는 기존 relational database 의 문제점을 해결하여 빠른 속도와 유연한 데이터 핸들링을 제공함이 목표.
- Github에 소스가 공개된 open-source 프로젝트.

2. Focus of Project

- 검색엔진: 사용자가 원하는 정보를 찾기 위한 소프트웨어
- 구현하기가 상당히 번거로움
  - 형태소 분리
  - inverted index
  - index 를 이용한 검색
  - 등등 직접 구현하려면 해야하는 게 많다.

- 가장 중요하게 생각한 부분은 **개발자가 손쉽게 사용 가능**할 것.
  - 어떻게 하면 사용하기 쉬울지 고민을 거쳐서 http api를 설계
  - 개발자가 고민할 필요 없이 분산시스템 구현

- 반면 텍스트 데이터를 직접 인덱싱 하는 방법은 이용자에게 노출되지 않음
  - 이 부분은 외부 라이브러리인 bleve를 사용

3. Requirements

Functional Requirements

- R1. Cluster Service Discovery
- R2. Master Node Election
- R3. Index Create
- R4. Index Read
- R5. Index Delete
- R6. Document Insert
- R6.1. Document Insert With ID
- R7. Document Read
- R8. Document Delete
- R9. Document Search

Non-Functional Requirements

- cluster에 속한 각 node는 저장할 데이터를 fair하게 나누어가지며, 이를 통해 node의 부하를 줄이고 성능을 최대화한다.
- user request의 평균 응답 시간은 1초 이내로 한다.

3.1 ChangeLog of Requirements

1. **Document ID가 중복인 경우** (#T6-4 Document ID Error TEST)

   ✅ Error를 발생시키지 않고 해당 Document 내용을 overwrite 한다.

2. **Persistent Data를 특별히 관리하지 않음**

   ✅ 이전에 master/data 역할을 했어도 현재의 election 과정에는 영향을 미치지 않는다
   ✅ 이전에 생성한 index/document를 기억하지 않고 현재의 data를 새로 생성한다
   ✅ 이전에 연결되었던 노드들을 따로 기억하지 않고 새로 discovery 과정을 거친다

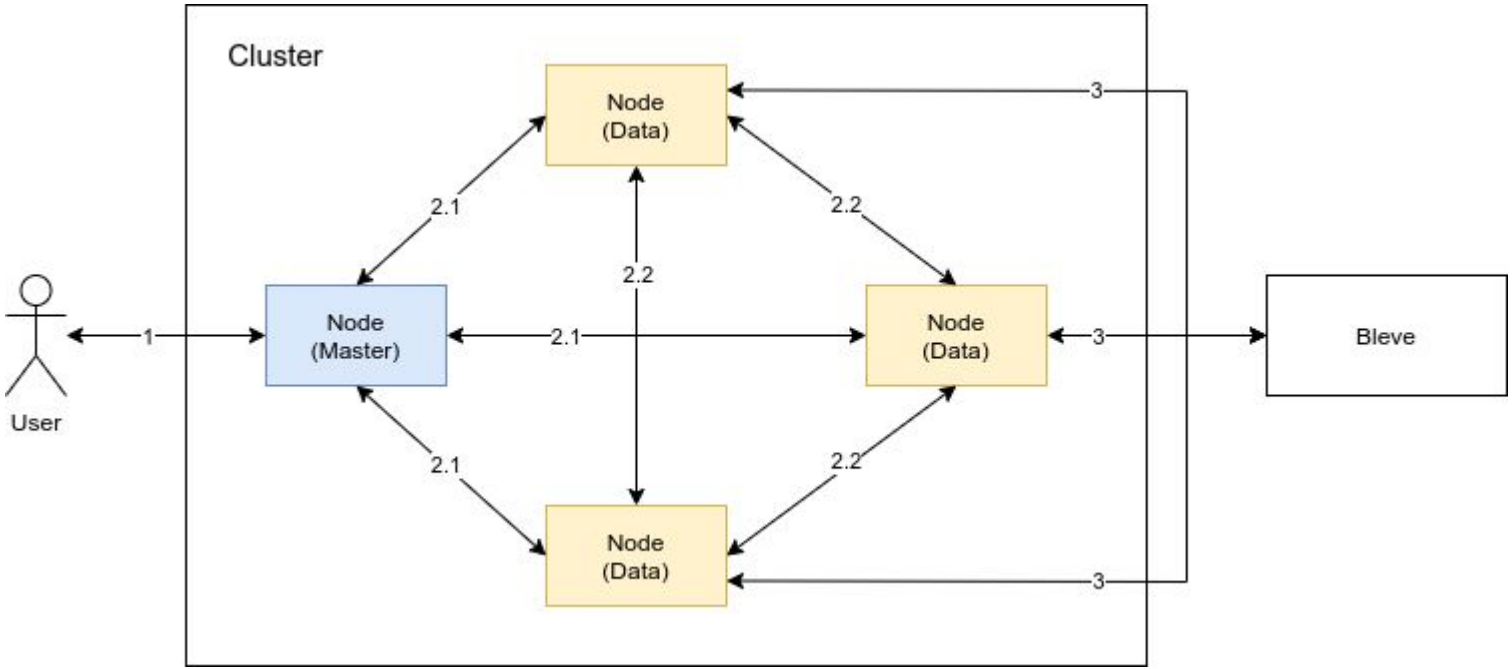
3. **Master node가 종료되면 새로운 election을 실행하지 않음**

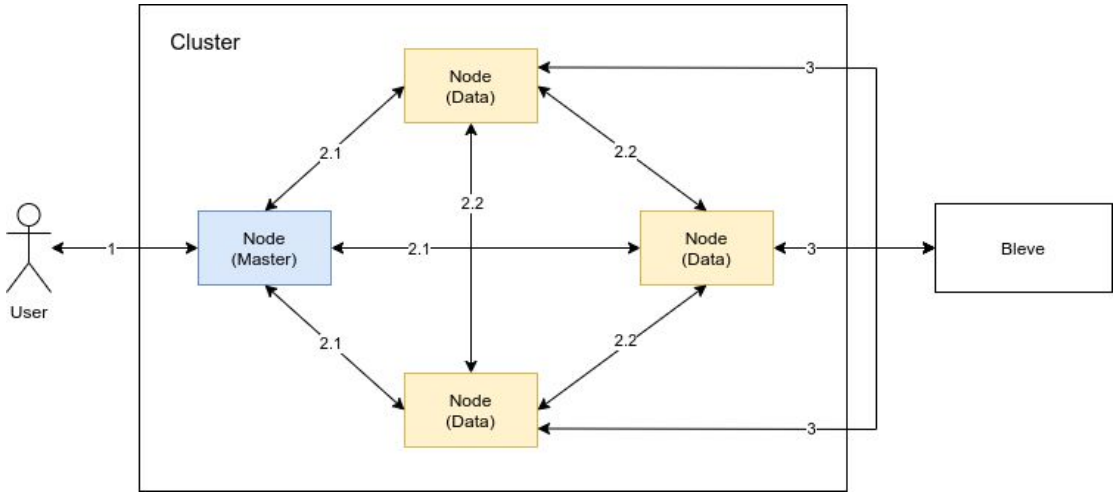
   ✅ 전체 클러스터를 종료하고 처음 부터 다시 discovery와 election을 거친다

4. **노드가 동시에 setup 되는 상황은 없다고 가정**

   ✅ 클러스터에 속한 단 하나의 노드라도 먼저 띄워지는 상황을 가정한다

4. 전체 Architecture





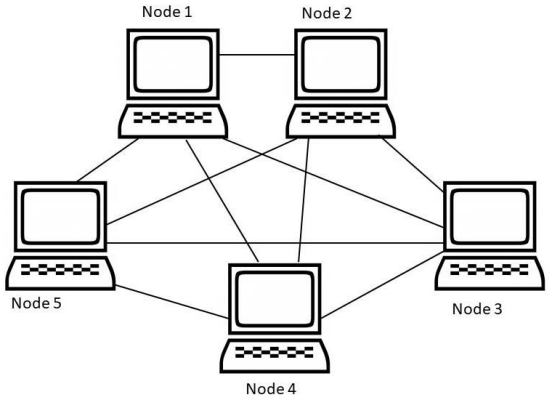
목표

- Hadoop등 프레임워크의 도움을 받지 않고 직접 분산시스템을 구현
- 즉, **Peer Finding**, **분산 합의**, **부하 분산**등을 모두 직접 고민

구현

- Request-Reply 기반
- 메세지 브로커 없음
- Leader - Follower architecture
- Sharding 개념의 부하 분산

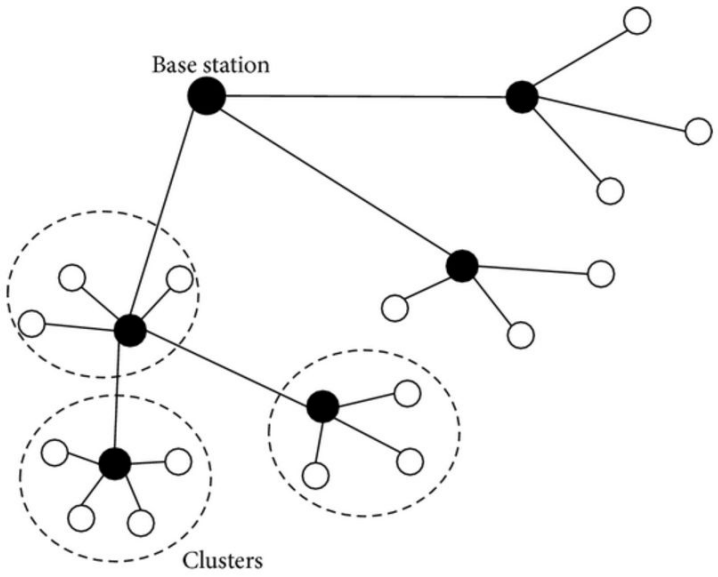
Peer Finding



Fully-connected mesh network 필요.

- 모든 노드가 서로 연결된 상태
- 노드끼리 데이터를 빠르게 주고 받기 위해서
- 각 노드가 다른 모든 노드와 TCP connection

Peer Finding

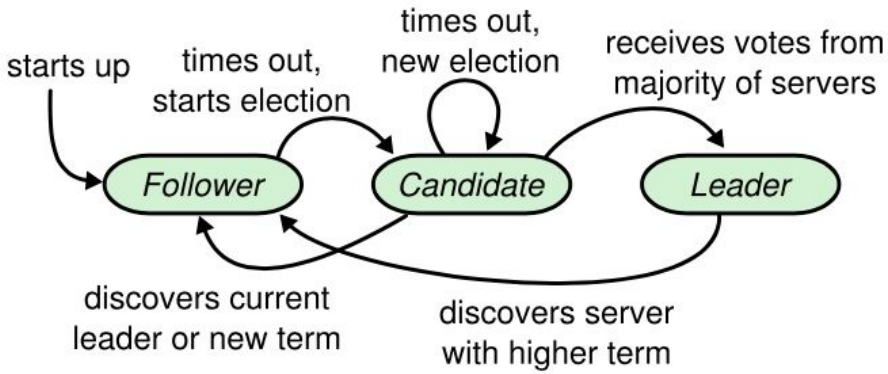


과정

1. 처음 설정된 Seed host들과 Connection
2. 1에서 맺은 Connection으로 해당 노드들 식별
3. 식별 후 각각의 노드는 원격 노드 리스트를 공유
4. 3의 리스트를 통해 아직 연결이 안 된 노드와 Connection
5. 2, 3, 4번 반복
6. 최종적으로 Cluster내의 Fully-connected mesh network를 형성

Peer Finding은 크게 어렵지 않음

분산 합의 - Raft algorithm



Raft algorithm distributed consensus protocol

- 분산 합의 알고리즘.
  클러스터의 노드 상태를 통일 시킨다.

Leader: 클러스터 상태(Cluster Status) 정보를 관리

Follower: 실제로 색인된 데이터를 저장

Candidate: 클러스터의 Leader를 선출(election) 하기 위한 상태.

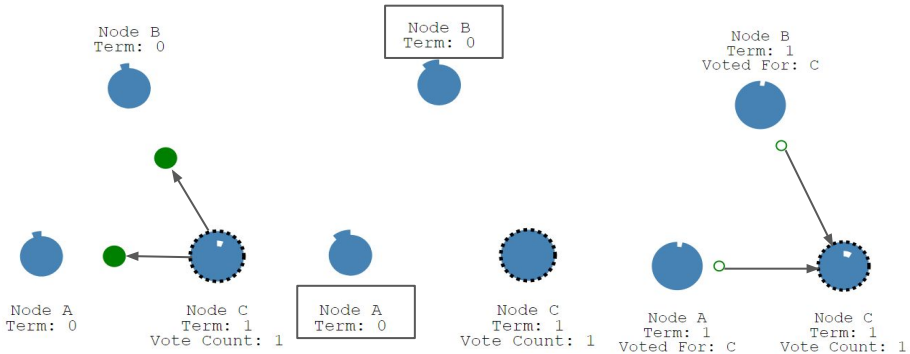
분산 합의 - Raft algorithm

직접 구현하려고 보니까 생긴 문제들

- 각 노드는 독립적 (중앙 통제 x).
- 다른 노드의 state를 알 방법이 없다.
- 각 노드에서도 Race condition 발생
- 그 어떤 상태에서도 동작이 원할해야 한다.

단순하게 구현해보자.

분산 합의 - Raft algorithm

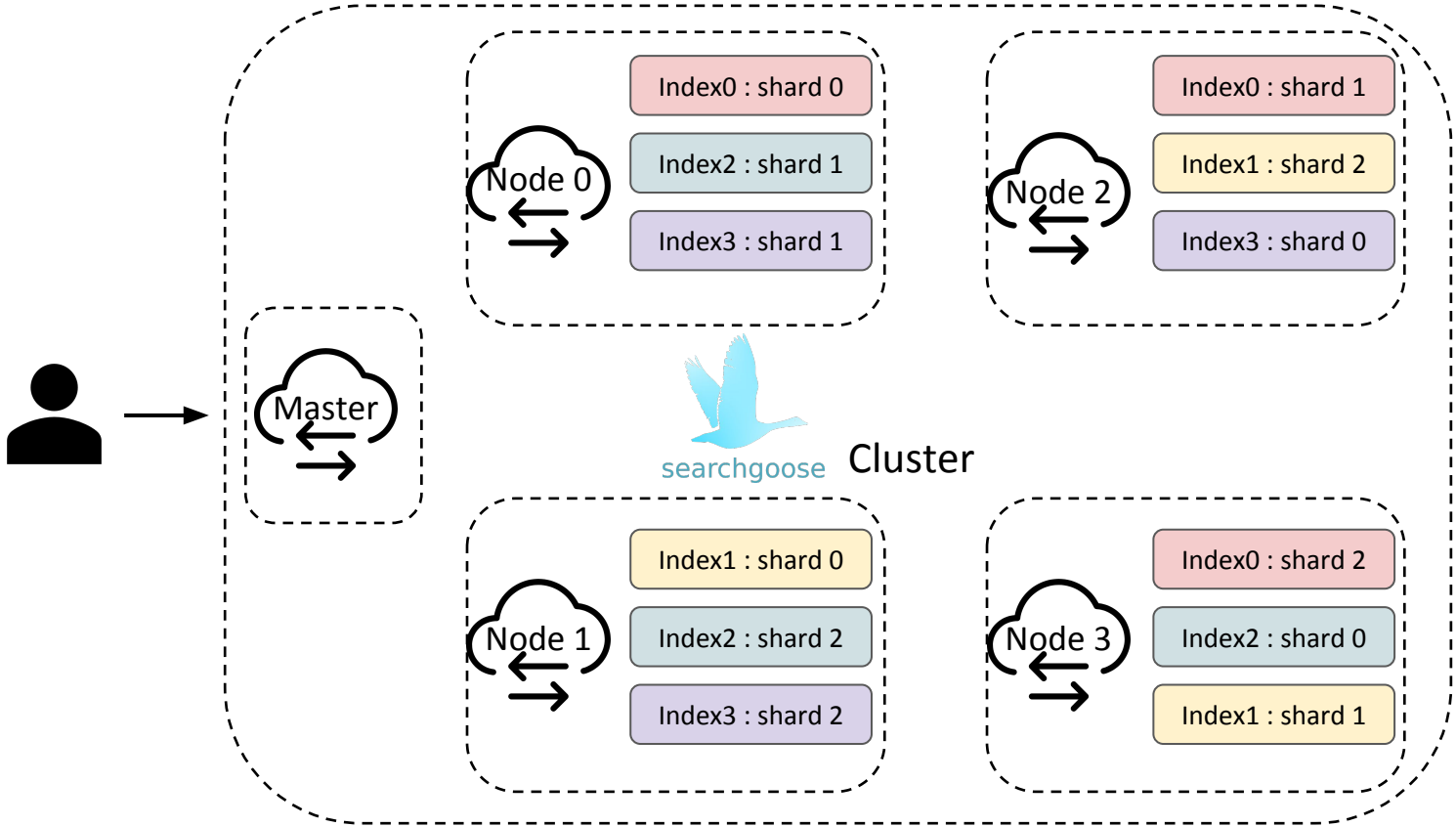


최대한 단순하게 구현한 결과

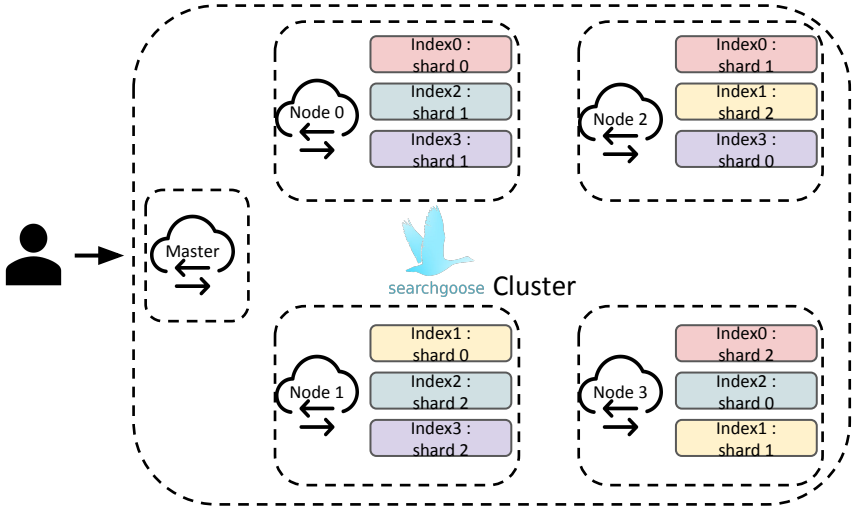
Election process in Searchgoose

1. 모든 노드가 Candidate 상태.
2. 각 노드에서 Join을 요청하는 request를 다른 노드들에게 전송
3. 2번 request를 받은 노드들은 Join을 동의하는 response 전송 (JoinResponse)
4. 가장 먼저 과반수의 JoinResponse를 받은 노드가 Leader로 선출
5. 자신이 Leader로 선출 되었음을 연결이 맺어진 다른 노드들에게 알린다 (Publish)

부하 분산 - Sharding



부하 분산 - Sharding

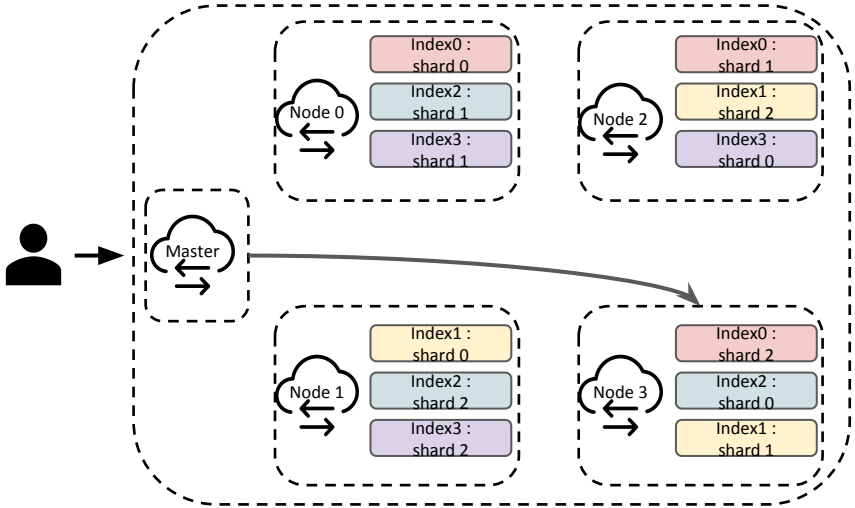


각 노드가 데이터를 최대한 균등하게 분산.

Rule

- 각 노드가 Shard를 균등하게
- 각 노드가 데이터를 균등하게
- 데이터가 어디에 있어도 찾는 것이 가능
- 검색 시에는 모든 데이터에서 질의

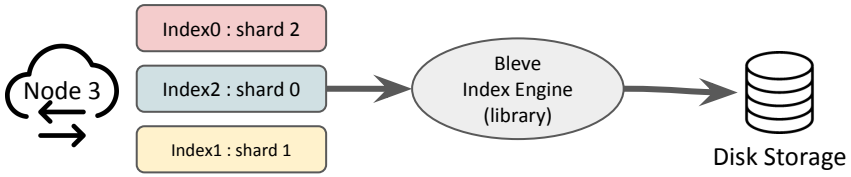
부하 분산 - Sharding



Solution

1. Shard 배치 시 들고 있는 Shard가 적은 순으로 배치
2. 데이터에 UUID 부여,
   특정 알고리즘으로 배치할 Shard id 계산,
   데이터 forwarding
3. 데이터 조회시에도 UUID 알고리즘 계산을 통해 같은 Shard에서 가져오는 것이 가능
4. 검색시에는 모든 Shard에서 질의해서 가져온다.

부하 분산 - Sharding



Bleve Index Engine
- 데이터 삽입
- 데이터 조회
- 데이터 삭제
- 검색

데이터 관리는 각 follower 노드에서.

- 형태소 분석, inverted-index, full-text search가 이루어져야 한다.
- 사용자에게 노출되는 부분은 아님.
- 외부 라이브러리 bleve의 도움을 받아서 처리

부하 분산 - Sharding

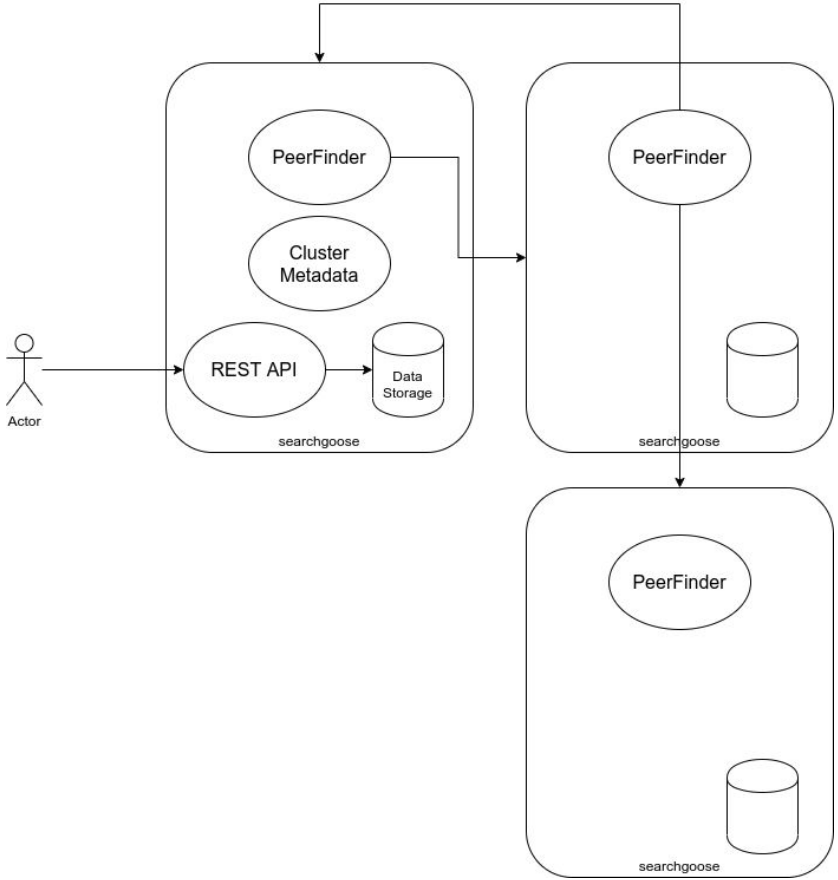
아이디어는 좋았으나

요청이 많아질 때 생긴 문제점

- 네트워크 버퍼문제
  (패킷이 4kb가 넘으면 끊어서 읽게 된다.)
- 패킷 length를 같이 보내게 했더니 동시성 이슈 발생
  (읽기/쓰기 시 race condition)

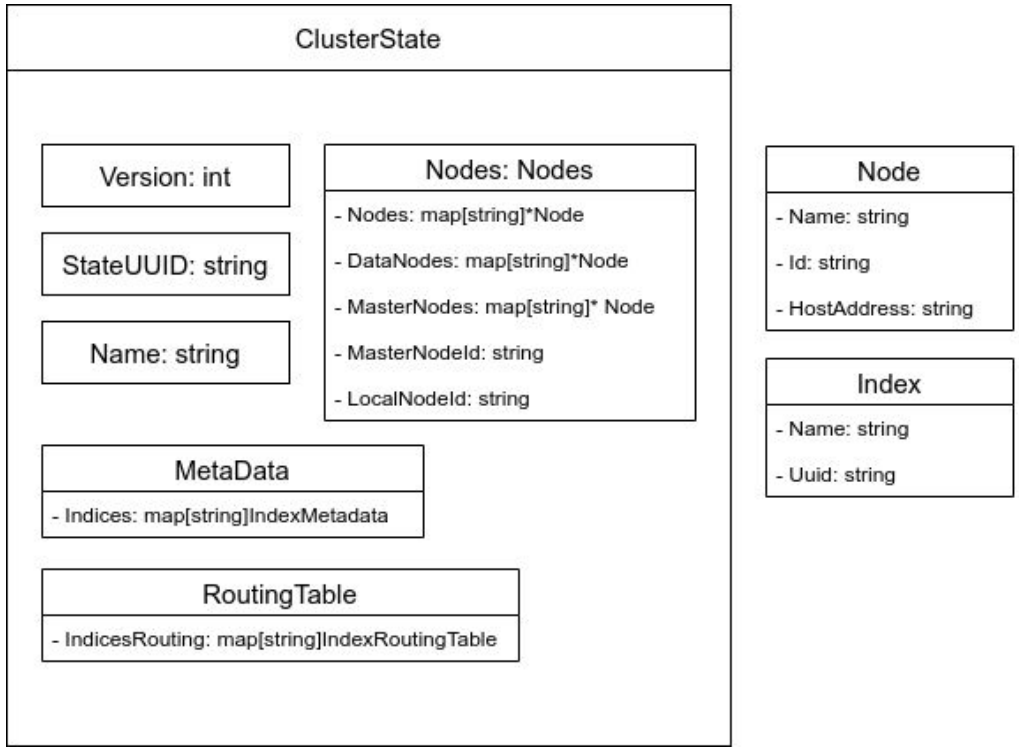
- Lock으로 해결했더니 전체 throughput이 Lock 이전에 비해 15%정도 하락
- 이후 해결해야할 과제로 미뤄둠

4. 전체 Architecture - General



- User는 REST API에 접근,
  각 노드의 Disk Storage에 저장된 데이터를 불러 온다.
- 각 노드는 PeerFinder로 다른 노드를 찾는다.
- 클러스터 정보를 저장하는 Cluster Metadata (ClusterState) 존재

4. 전체 Architecture - Cluster State

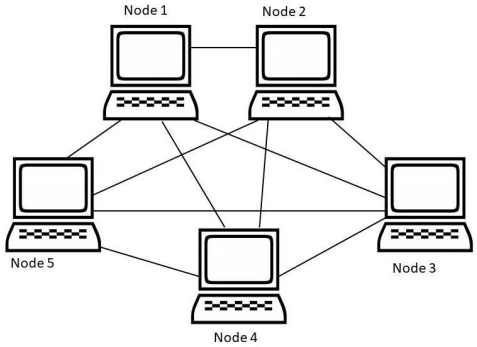


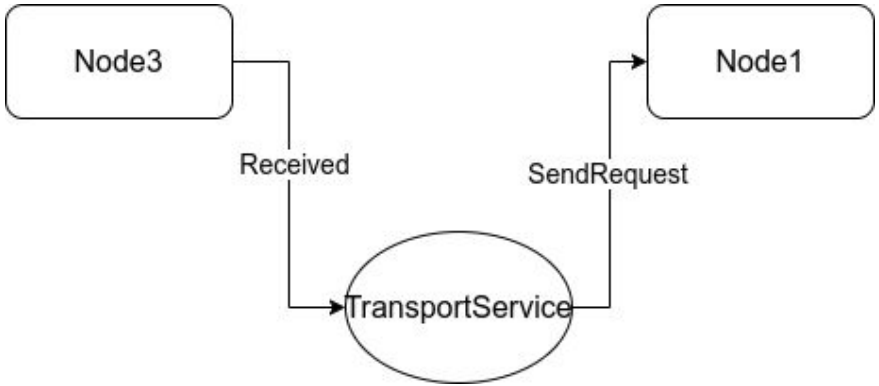
ClusterState

cluster는 cluster에 속한 모든 node와 index의 정보를 가진 ClusterState struct를 가지고 있다.

Nodes: cluster에 속한 모든 node의 정보
Metadata: cluster가 관리하는 모든 index의 정보
RoutingTable: cluster가 관리하는 모든 [index-shard-datanode: matching pair의 정보

4. 전체 Architecture - Network Transport

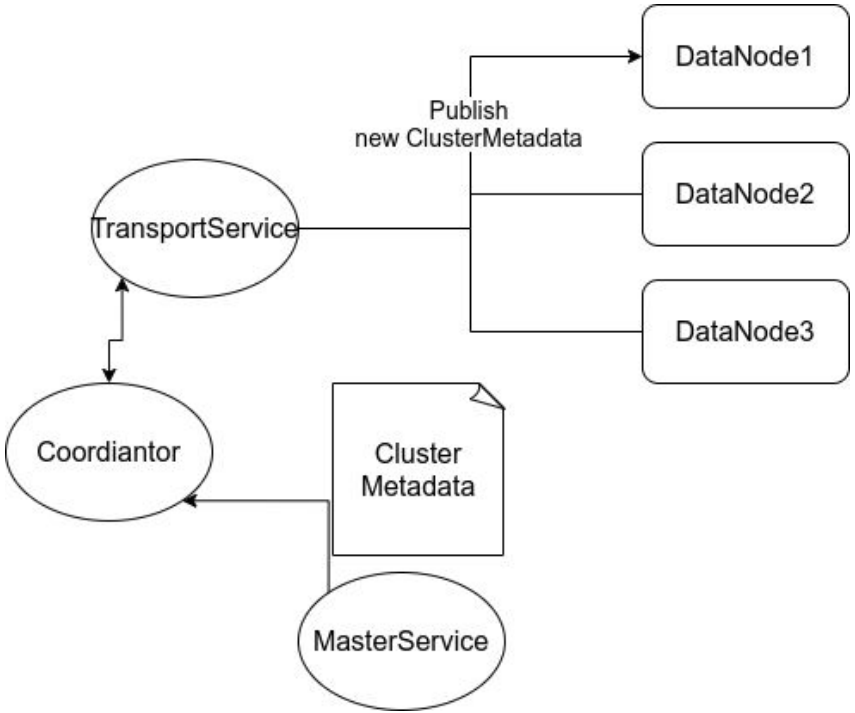




TransportService

- 노드 간 데이터 전송 / 수신을 담당, Connection 관리
- 다른 노드로 Request 전송 시 TransportService 호출
- 다른 노드로부터 Request 수신시 type에 따라 handler 호출

4. 전체 Architecture - Send message to other nodes



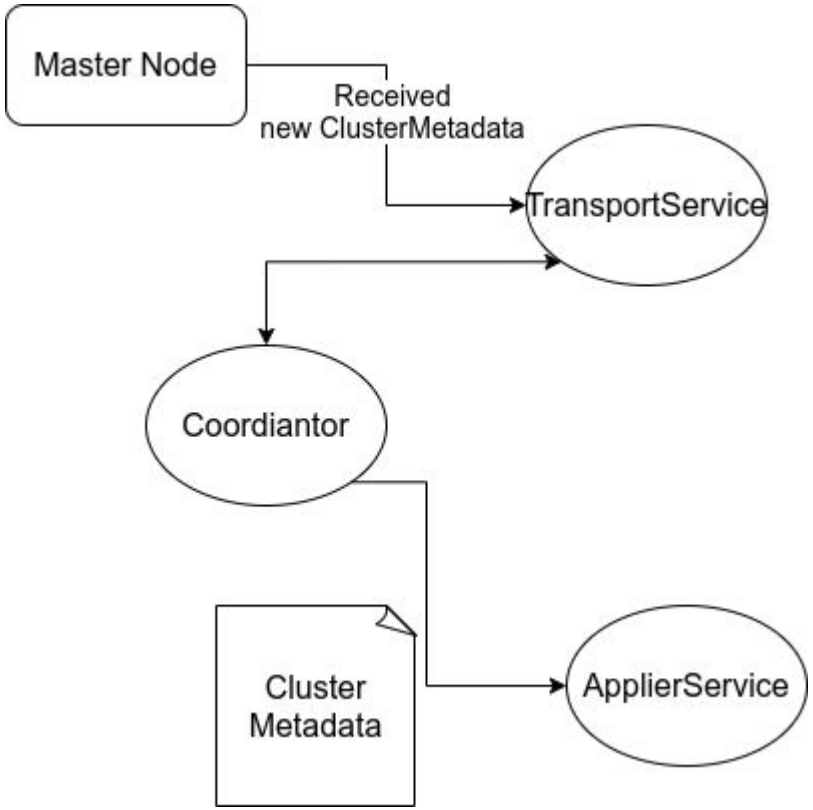
MasterService

- ClusterState 변경 담당
- Leader node 만 호출 가능
- ClusterState가 변경사항이 있을경우 Coordinator 를 호출해 다른 노드에게 전파

Coordinator

- 분산 합의 알고리즘 구현체.
- Leader election 담당 및 ClusterState 변경시 전파

4. 전체 Architecture - Receive messages from another node



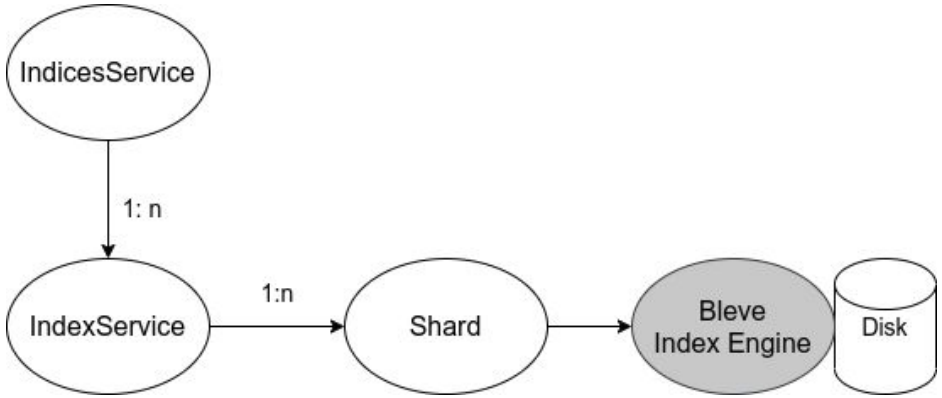
Coordinator

- Master Node로부터 변경된 ClusterState 수신 시 적용

ApplierService

- 변경된 ClusterState 를 해당 노드에 적용
- 인덱스 및 샤드 생성 등

4. 전체 Architecture - Index and shard data management



IndicesService

- 해당 노드에서 관리할 Index들을 담당

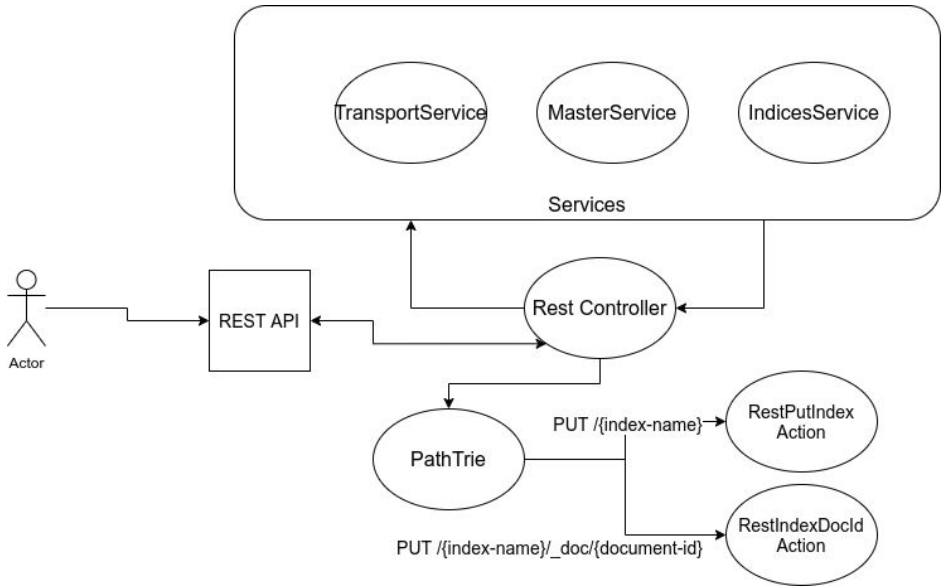
IndexService

- 해당 노드에서 관리할 Index의 Shard를 담당

Shard

- 데이터를 Disk Storage에 넣는 역할 담당
- Bleve 라이브러리 호출

4. 전체 Architecture - REST APIs



HTTP request path 패턴은 여러 종류

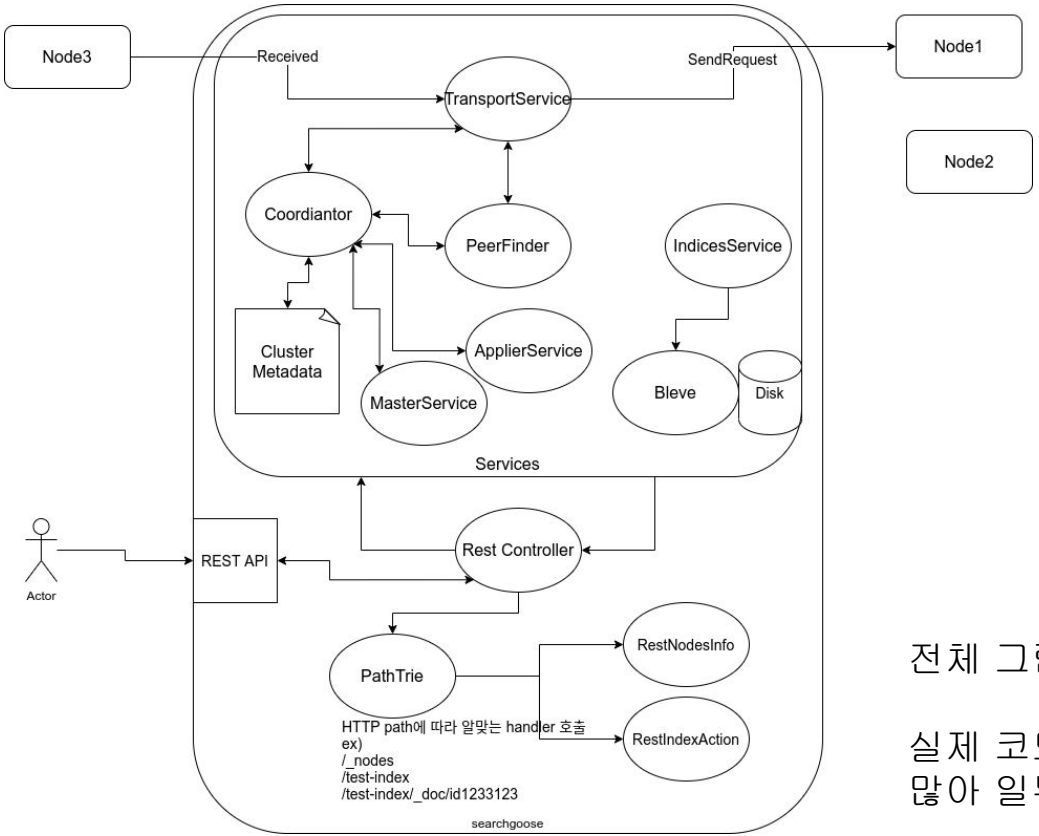
- /{index}
- /{index}/_doc
- /{index}/_doc/{id}
- /{index}/_search

해당 패턴에 맞는 Handler를 호출해야함

Rest Controller

- 여러 Request Handler 관리
- Path pattern 에 맞는 handler 호출

4. 전체 Architecture - Total



전체 그림.

실제 코드상에는 클래스가 더 많아 일부는 생략되었습니다.

5. System Test Cases & Results

Test Name	Description	P/F
T1-1. Config File TEST	config.yaml 파일에서 seed host를 정상적으로 읽어올 수 있는지 확인	P
T1-2. Peer Finding TEST	한 cluster 안에 속한 모든 host가 정상적으로 fully-connected mesh network를 형 성할 수 있는지 확인	P
T1-3. Seed Host Error TEST	config 파일에 잘못된 주소의 seed host가 있다면 error를 발생시킬 수 있는지 확인	P
T2-1. Vote Request TEST	vote request & response를 정상적으로 발신 및 수신 가능한지 확인	P
T2-2. Publish TEST	master node 선출, cluster state struct 생성 및 초기화, publish request 발신 및 수 신이 정상적으로 가능한지 확인	P
T3-1. Index Create Master Node TEST	master node가 정상적으로 cluster state 정보를 변경하고, data node에게 변경된 cluster state를 전달할 수 있는지 확인	P
T3-3. Index Create Format Error TEST	data node가 정상적으로 master node로부터 변경사항을 전달받아 적절한 행동을 하고, 그 결과를 알릴 수 있는지 확인	P
T3-4. Index Name Error TEST	Index name이 중복인 경우 error를 발생시킬 수 있는지 확인	P
T4-1. Index Read TEST	특정 index의 정보를 정상적으로 조회 가능한지 확인	P

5. System Test Cases & Results

Test Name	Description	P/F
T4-2. Index Read Format Error TEST	지정된 index read request format을 준수하지 않은 경우 error를 발생시킬 수 있 는지 확인	P
T5-1. Index Delete Master Node TEST	master node가 정상적으로 cluster state 정보를 변경하고, data node에게 변경된 cluster state를 전달할 수 있는지 확인	P
T5-2. Index Delete Data Node TEST	data node가 정상적으로 master node로부터 변경사항을 전달받아 적절한 행동을 하고, 그 결과를 알릴 수 있는지 확인	P
T5-3. Index Delete Format Error TEST	지정된 index delete request format을 준수하지 않은 경우 error를 발생시킬 수 있 는지 확인	P
T6-1. Document Insert Master Node TEST	master node가 정상적으로 document를 저장할 적절한 shard를 찾고 data node 에게 request를 전달할 수 있는지 확인	P
T6-2. Document Insert Data Node TEST	data node가 document를 정상적으로 삽입할 수 있는지 확인	P
T6-3. Document Insert Format Error TEST	지정된 document insert request format을 준수하지 않은 경우 error를 발생시킬 수 있는지 확인	P
T6-4. Document ID Error TEST	document ID가 중복인 경우 error를 발생시킬 수 있는지 확인	P
T7-1. Document Read Master Node TEST	master node가 정상적으로 document가 저장된 shard를 찾고 data node에게 request를 전달할 수 있는지 확인	P
T7-2. Document Read Data Node TEST	data node가 document를 정상적으로 읽을 수 있는지 확인	P

5. System Test Cases & Results

Test Name	Description	P/F
T7-3. Document Read Format Error TEST	지정된 document read request format을 준수하지 않은 경우 error를 발생시킬 수 있는지 확인	P
T7-4. Document Read Not Exist Error TEST	해당 document가 존재하지 않는 경우 error를 발생시킬 수 있는지 확인	P
T8-1. Document Delete Master Node TEST	Master node가 정상적으로 document가 저장된 shard를 찾고 data node에게 request를 전달할 수 있는지 확인	P
T8-2. Document Delete Data Node TEST	Data node가 document를 정상적으로 삭제할 수 있는지 확인	P
T8-3. Document Delete Format Error TEST	지정된 document delete request format을 준수하지 않은 경우 error를 발생시킬 수 있는지 확인	P
T8-4. Document Delete Not Exist Error TEST	해당 document가 존재하지 않는 경우 error를 발생시킬 수 있는지 확인	P
T9-1. Document Search Master Node TEST	Master node가 data node에게 정상적으로 search request를 전달하고, 결과를 취 합할 수 있는지 확인	P
T9-2. Document Search Data Node TEST	Data node가 정상적으로 master node로부터 request를 수신하고, 문서 검색 후, 결과를 master node에게 알릴 수 있는지 확인	P
T9-3. Document Search Format Error TEST	지정된 document Search request format을 준수하지 않은 경우 error를 발생시킬 수 있는지 확인	P

5.1 Success Criteria Check

2.2. Success Criteria

- success 의 기준은 priority 가 primary 인 모든 use case 의 test case 를 pass 하는 것으로 한다.
- 본 프로젝트의 경우 9 개의 use case 모두 primary 이므로, 총 29 개의 test case 중 29 개를 pass 한 경우 success 이다.

Ref & Use Case	Test Case No.	Priority
R1. Cluster Service Discovery	T1-1, T1-2, T1-3	primary
R2. Master Node Election	T2-1, T2-2	primary
R3. Index Create	T3-1, T3-2, T3-3, T3-4	primary
R4. Index Read	T4-1, T4-2	primary
R5. Index Delete	T5-1, T5-2, T5-3	primary
R6. Document Insert	T6-1, T6-2, T6-3	primary
R6.1. Document Insert With ID	T6-4	primary
R7. Document Read	T7-1, T7-2, T7-3, T7-4	primary
R8. Document Delete	T8-1, T8-2, T8-3, T8-4	primary
R9. Document Search	T9-1, T9-2, T9-3	primary

STP에 명시된 Success Criteria에 의하면,
총 29개의 test case 중
29개를 pass하였으므로,
success이다.

5.2 Test Issues - Bug fix

1. **동시성 이슈(Race Condition)**

   ❌ 평소에는 빈번히 일어나지 않는 버그이며 리퀘스트를 동시에 여러 번 처리하는 테스트 과정에서 발생

   ✅ 원자성이 필요한 부분 ➔ 네트워크 버퍼
   💡 Connection 맺을 때 Read/Write 하는 부분을 Mutex Lock/Unlock을 통해 해결

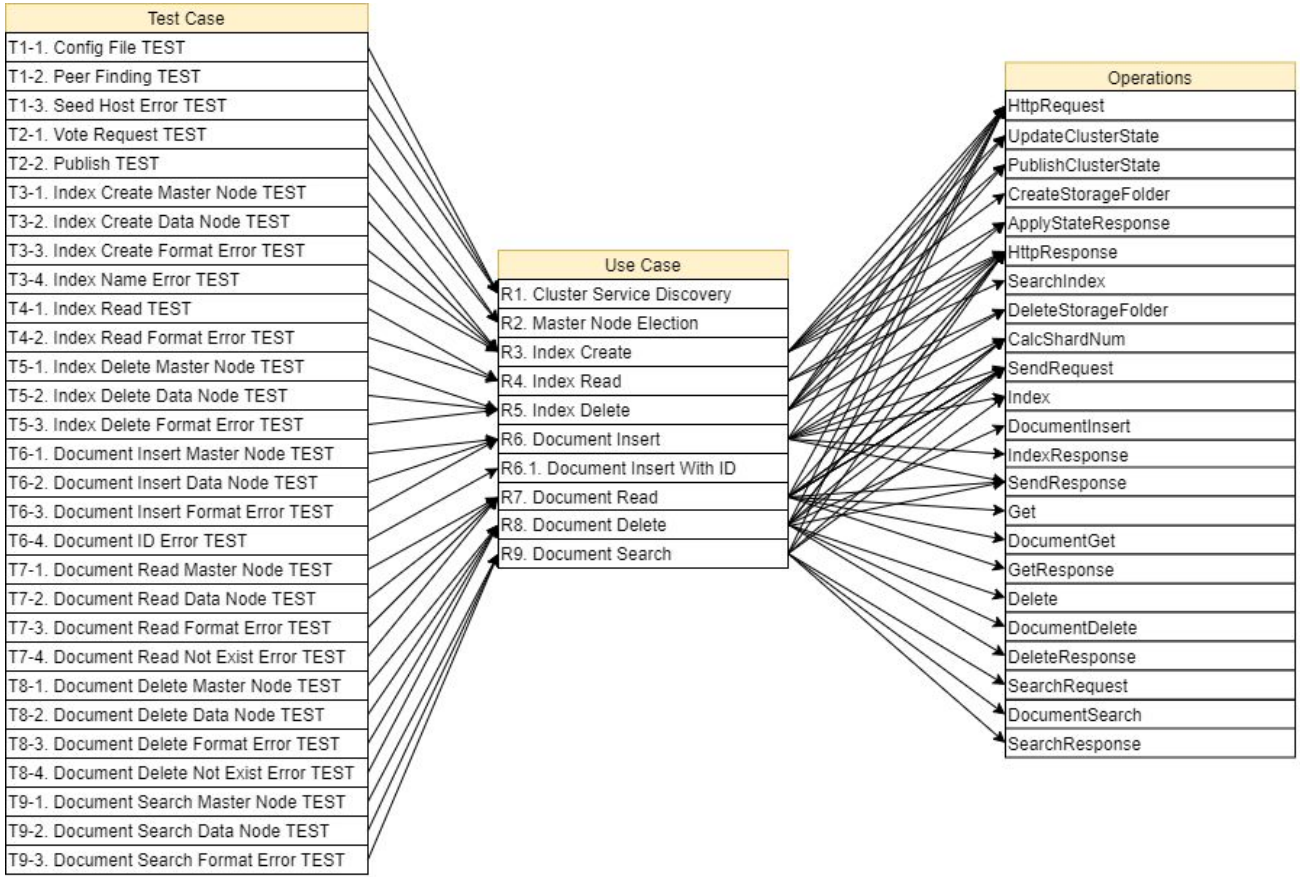
   ✅ 원자성이 필요한 부분 ➔ 노드들의 Connection을 관리하는 Map
   💡 Map에 새로운 노드들의 Connection을 추가하고 삭제할 때 Mutex Lock/Unlock을 통해 해결
   💡 Write 의 경우 동일한 노드의 Connection을 overwrite 할 수 있도록 처리

2. **Election이 끝나기 전에 HTTP 채널을 여는 경우**

   ❌ Single 모드에서는 발생하지 않고 Clustering 모드에서 발생하는 버그
   ❌ Cluster 안에 있는 노드들끼리 Election이 끝나기 전에 스스로를 master 라 생각하고 HTTP 채널을 연다

   ✅ 해당 순서를 보장하고 제어할 필요가 있음
   💡 Goroutine 실행 시 Election이 끝났을 때 Go Channel을 통해 signal을 보내고
   💡 해당 Channel에 signal이 올 때까지 holding

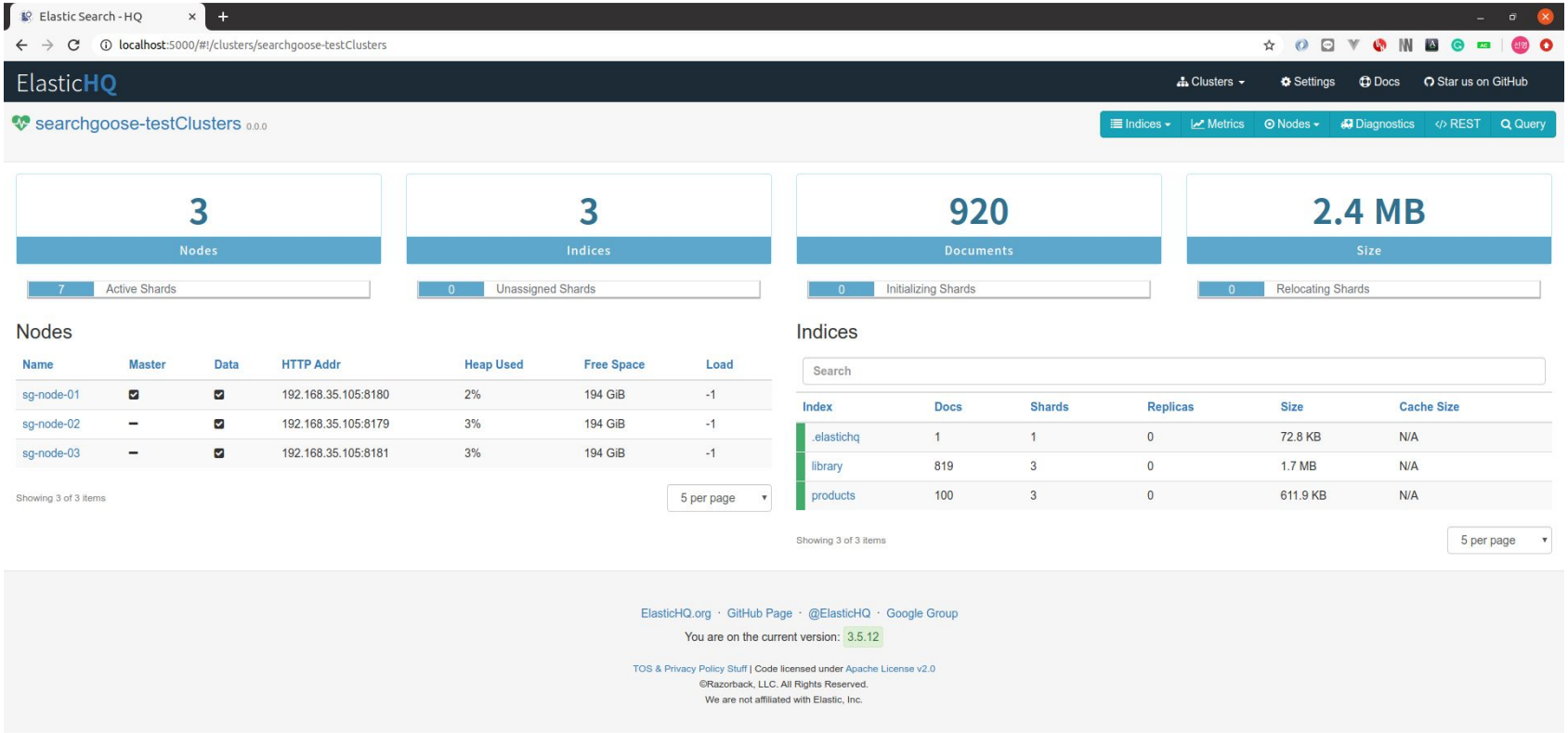
6. Traceability



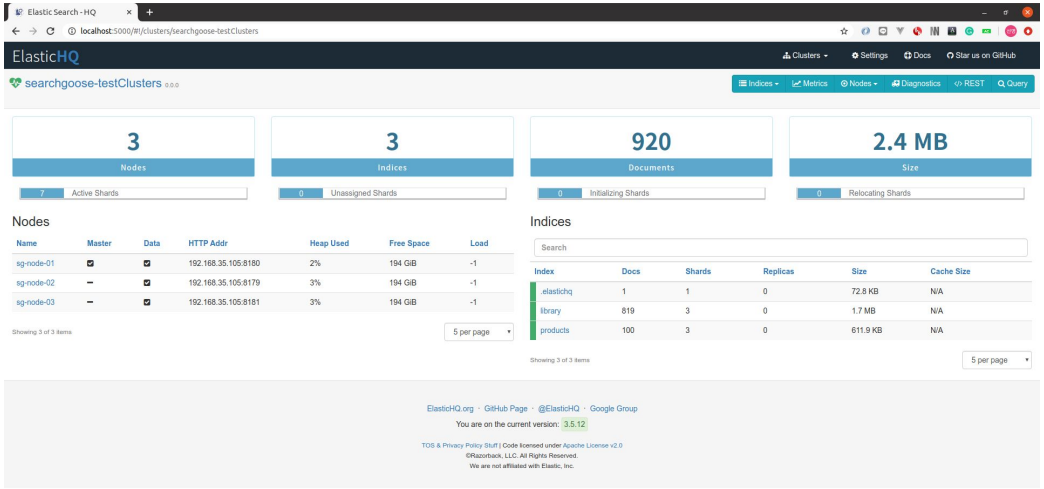
7. Demo

1. 클러스터 모니터링 웹뷰
2. 영문 소설 검색 서비스
3. 커머스 상품 검색 서비스

7.1 클러스터 모니터링 웹뷰



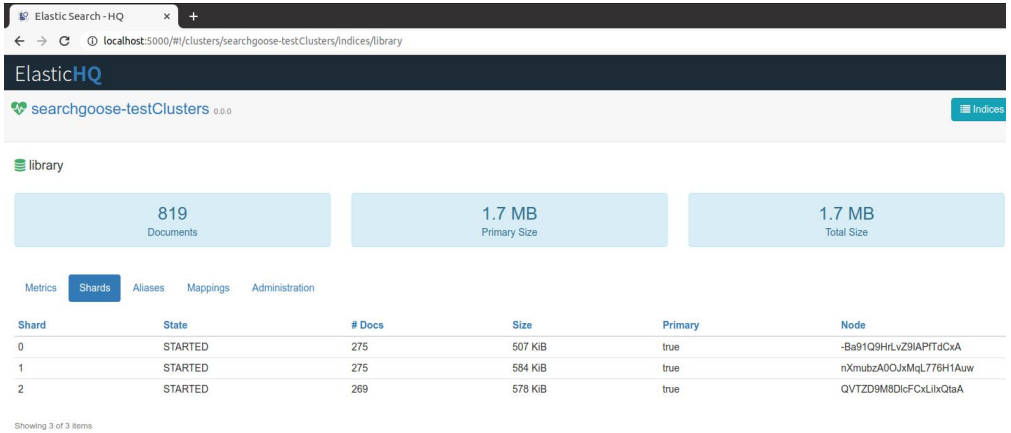
7.1 클러스터 모니터링 웹뷰



Searchgoose 클러스터 상태를 보여주는 웹뷰 도구입니다.

현재 클러스터에 참여하는 노드, 인덱스 정보 및 데이터 용량, document 갯수 등을 확인 할 수 있습니다.

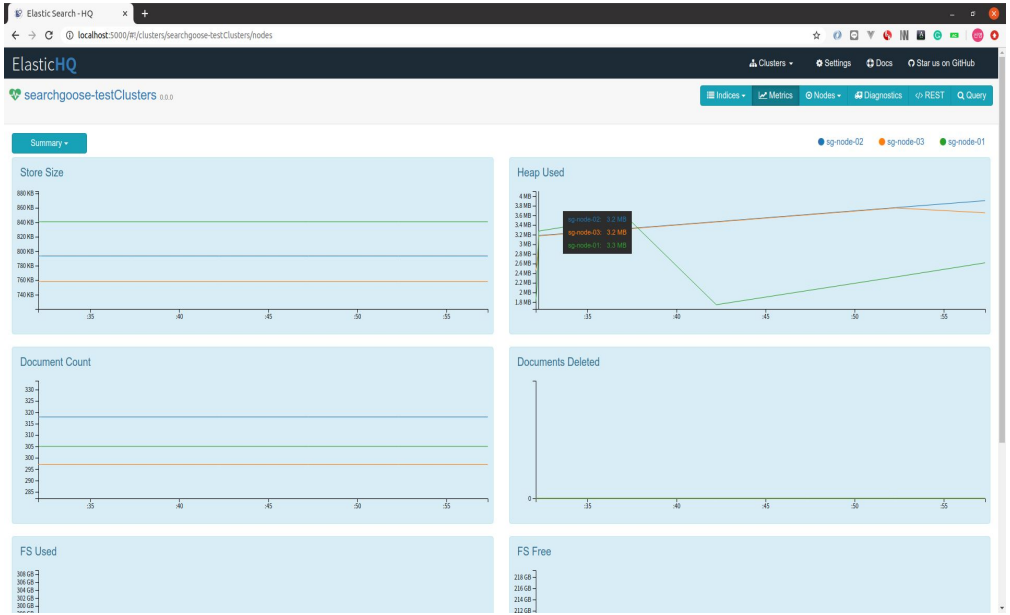
7.1 클러스터 모니터링 웹뷰



본 도구를 통해 생성된 인덱스의 정보를 더 자세히 들여다 볼 수 있는데,

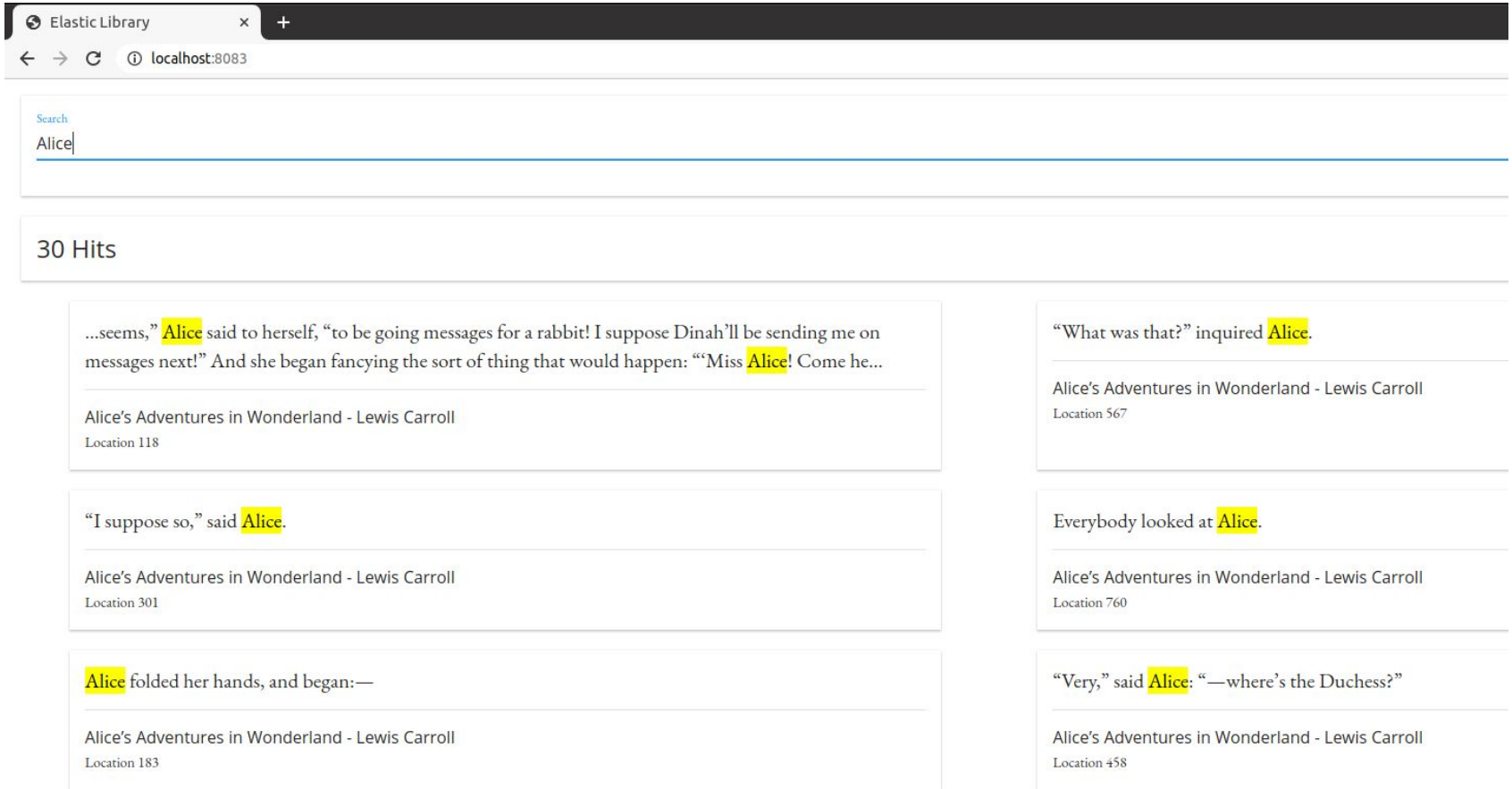
각 샤드별로 데이터가 얼마나 분산되어 저장되는지 등을 확인 할 수 있습니다.

7.1 클러스터 모니터링 웹뷰

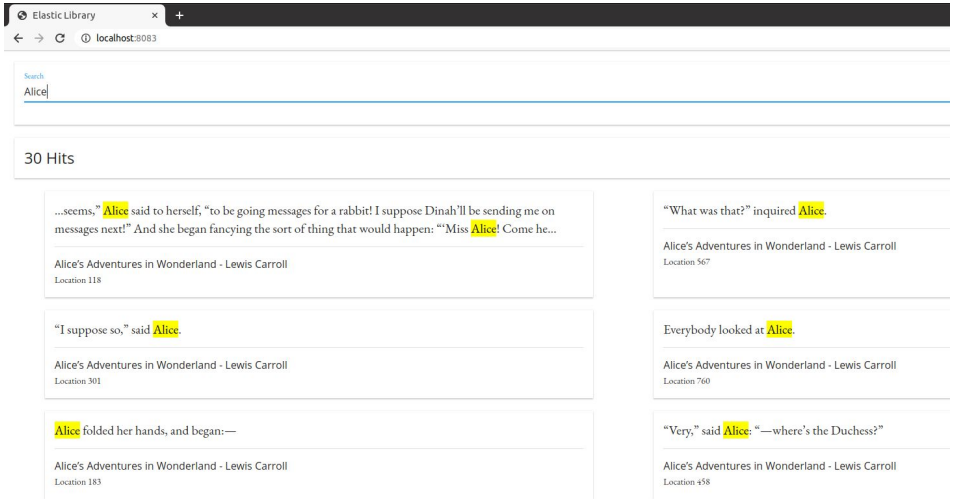


현재 클러스터에 참여하고 있는 각 노드의 상태 지표등을 확인할 수도 있습니다.

7.2 영문 소설 검색 서비스



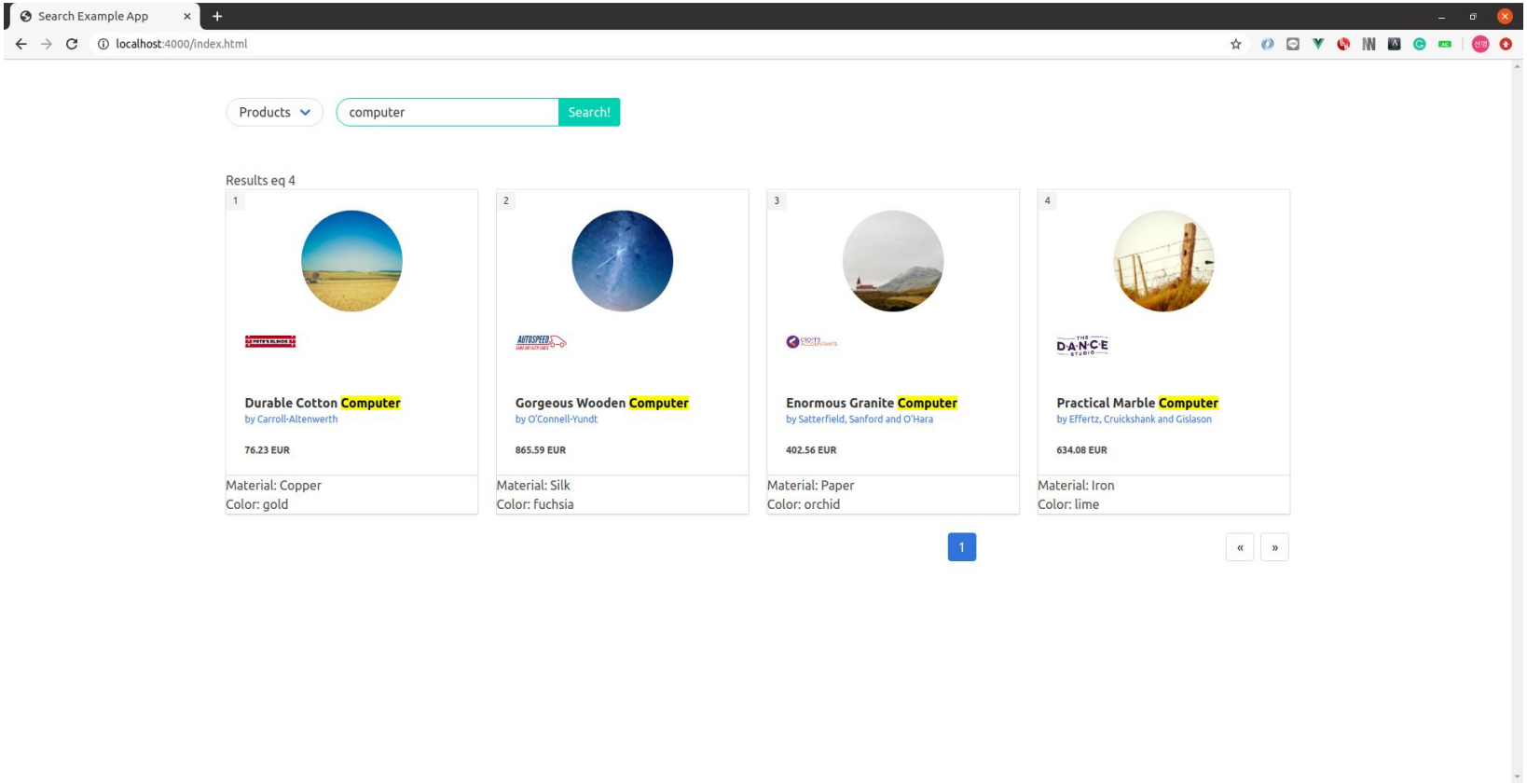
7.2 영문 소설 검색 서비스



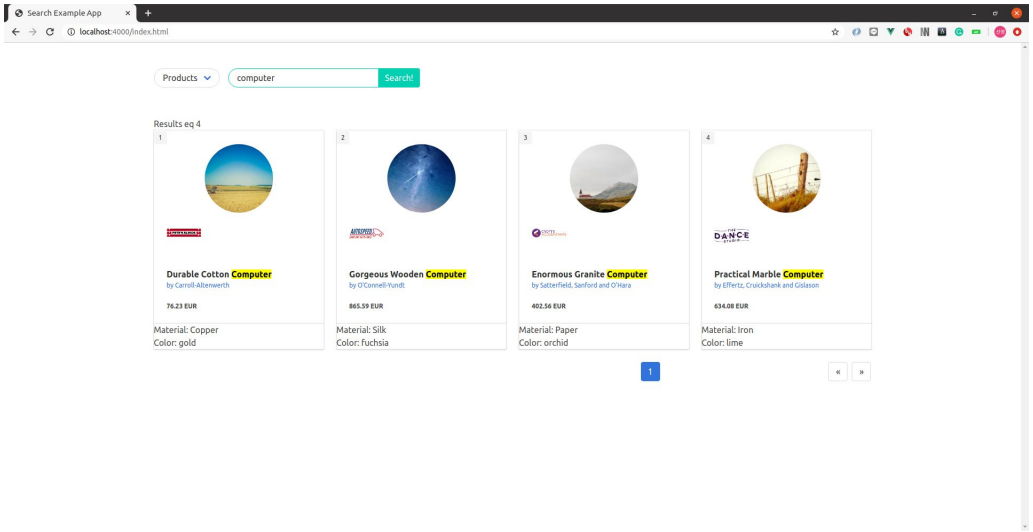
Searchgoose에 미리 영문 소설 데이터를 인덱싱 해두고, 검색기능을 연동해 만든 영문 소설 검색 서비스입니다.

왼쪽 이미지에선 Alice in wonderland 소설 데이터를 인덱싱 해두고,
"Alice" 라는 키워드로 검색시 나온 결과입니다.

7.3 커머스 상품 검색 서비스



7.3 커머스 상품 검색 서비스



커머스서비스의 필수 기능인 상품 검색 기능을 Searchgoose로 연동한 커머스 상품 검색 서비스입니다.

왼쪽 이미지에선 가상으로 커머스 데이터를 만들어서 Searchgoose에 인덱싱해 둔 뒤 “computer” 라는 키워드로 검색한 결과입니다.

Thank you