전문 검색과 분석을 위한 Elasticsearch 서버

Ch 1. 일래스틱서치 클러스터 시작

아꿈사 스터디
정민철

들어가기 전

- Elasticsearch 최신 버전 => 1.7.1
- 책에서 다루는 버전 => 1.0.0
- 변경된 부분은 https://www.elastic.co/downloads/past-releases 참고
- 1장에서 영향받은 부분: dynamic scripting disabled https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scripting.html#enable-dynamic-scripting

전문 검색

- SQL DB: 확장성 결여, 유연성 낮음, 언어분석 결여

- Apach Lucene
  - 전문 검색기능을 포함한 라이브러리 제공
  - 빠름, 확장성 제공, 다양한 언어 분석 기능 제공

루씬 아키텍처

- 용어정리
  - 다큐먼트: 주요 자료 전달 도구, 하나이상의 필드로 이루어짐
  - 필드: 이름, 값으로 구성된 다큐먼트의 구성요소
  - 키워드: 단어를 표현하는 검색단위
  - 토큰: 필드의 텍스트에 출현한 키워드의 상태정보

- 역색인: 색인에 포함된 키워드를 다큐먼트에 매핑하는 자료구조

- 검색: 질의 키워드를 색인된 키워드와 일치시키는 작업

- Elasticsearch Server 1.0 (document 1)
- Mastering Elasticsearch (document 2)
- Apache Solr 4 Cookbook (document 3)

Term	Count	Document
1.0	1	<1>
4	1	<3>
Apache	1	<3>
Cookbook	1	<3>
Elasticsearch	2	<1>, <2>
Mastering	1	<2>
Server	1	<1>
Solr	1	<3>

- 입력자료 분석
  - 토큰 추출기: 텍스트를 토큰으로 분리
  - 토큰 스트림: 토큰 추출기 작업 결과
  - 토큰 필터: 토큰 스트림에 담긴 토큰 처리
  - 문자 매퍼: 분석되지 않은 텍스트를 대상으로 동작

- 색인과 질의
  - 질의 과정에서도 분석이 일어남
  - 색인이 질의 키워드와 일치해야 함

- 점수와 질의 관련성
  - 점수: 다큐먼트가 질의와 얼마나 잘 맞는지
  - TF/IDF(Term Frequency/Inverse Document Frequency) 점수 매커니즘 사용

Elasticsearch란?

- Shay Banon이 시작한 오픈소스 검색 서버 프로젝트
  => 2010년 2월에 공개
- Apache Lucene 기반 검색 서버
- Elasticsearch is a distributed RESTful search engine built for the cloud
  - 분산 기능 제공
  - 실시간 처리 기능 제공
  - RESTful API 제공
  - 전문 검색(full-text search) 엔진

자료구조의 핵심개념

- 색인: 논리적인 자료를 저장하는 논리공간
  - 더 작은 조각으로 나뉘어질 수 있음
  - 여러 서버에 색인 분산 가능

- 다큐먼트: 여러필드 포함, 필드마다 타입 지정
  - 구조가 고정되어있지 않음
  - 다큐먼트 타입과 관련된 유일한 식별자 요구

- 다큐먼트 타입: 단일 색인에서 객체를 쉽게 구분하게 해줌
  - 동일한 프로퍼티에 대해 다른 타입을 설정할 수 없음

- 매핑: 다큐먼트의 필드에 대한 정보를 저장
  - 타입마다 적절한 분석 제공

일래스틱서치 핵심 개념

- 노드와 클러스터: 서로 협력하는 여러 서버에서 동작가능
  - 노드: 클러스터를 구성하는 각 서버
  - 클러스터: 서로 협력하는 서버 집합

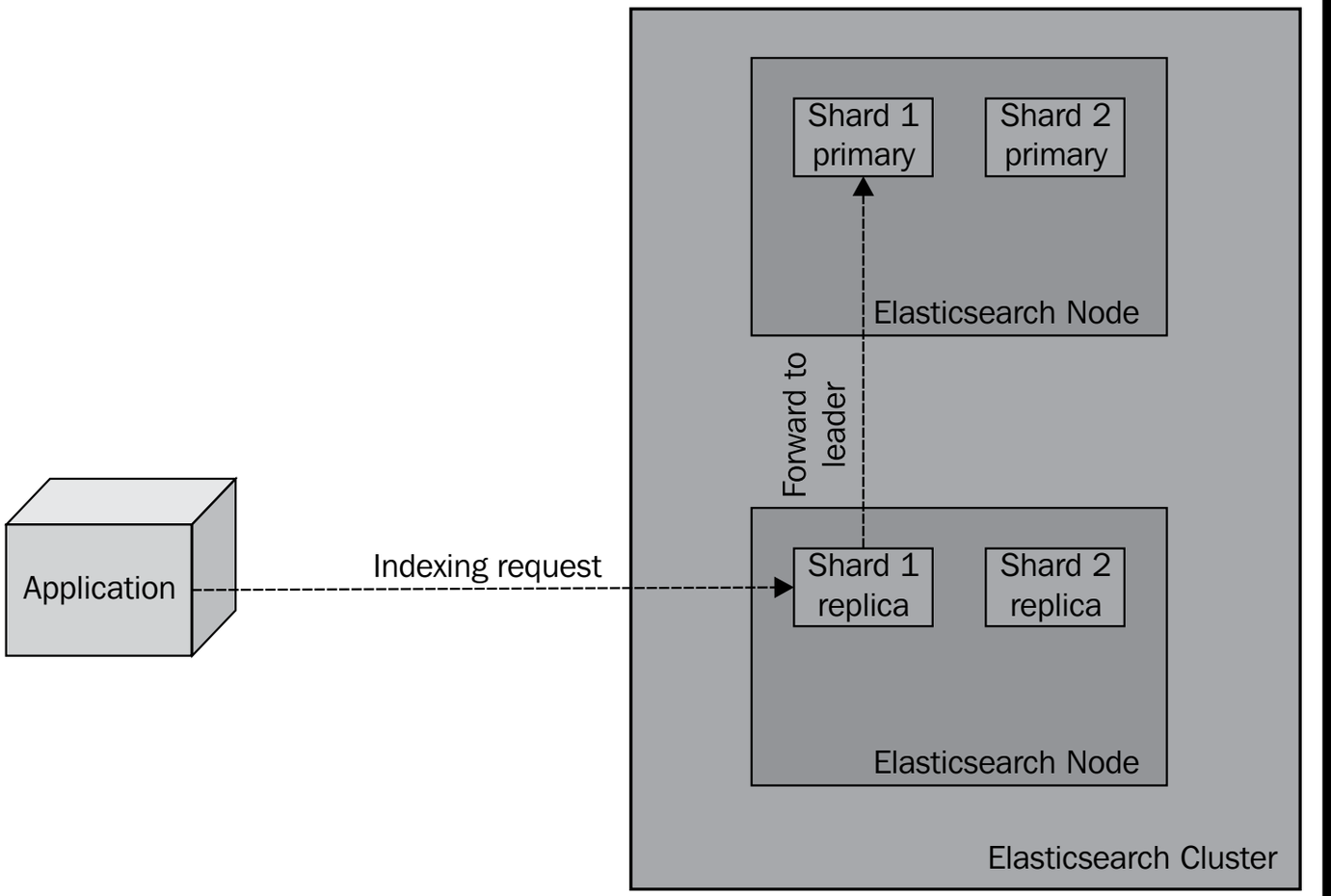
- 샤드: 자료를 더 작은 부분으로 나눠 여러 서버에 저장
  - 단일 노드에 다큐먼트를 모두 저장할 자원이 없는 경우
  - 질의를 유효한 샤드로 보내 결과를 병합함
  - 색인 속도 증가

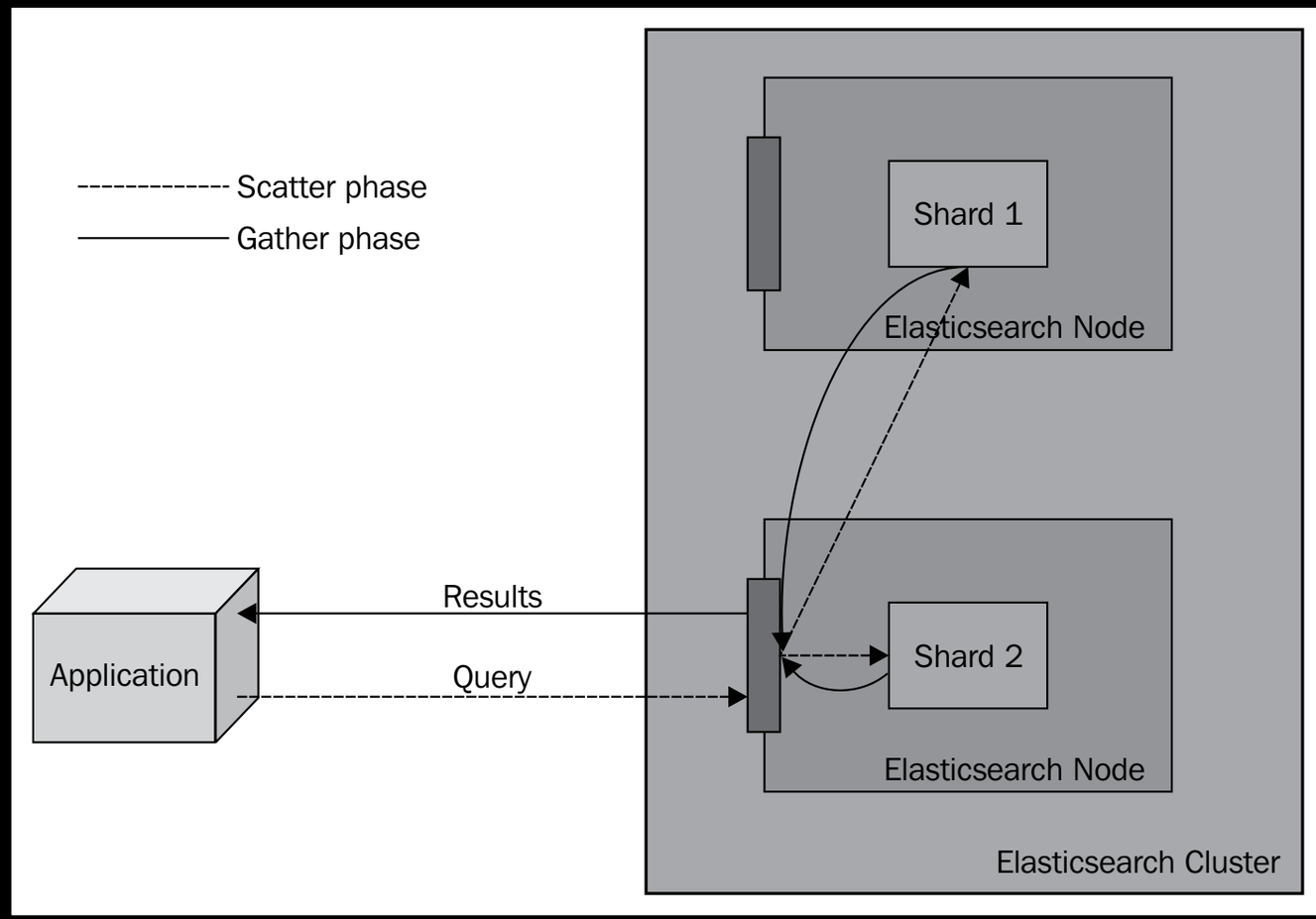
- 레플리카: 샤드의 동일한 복사본
  - 질의 처리량을 늘리거나 고가용성 목적
  - 주 샤드에 문제가 생기면 레플리카를 주 샤드로 승격

- 게이트웨이: 클러스터 상태 관리

색인과 검색

- 새로운 다큐먼트 추가
  - 클러스터는 목표색인 명세 및 노드에 다큐먼트 전송
  - 노드는 다큐먼트를 저장할 샤드 결정
  - 다큐먼트의 유일한 식별자를 사용해 위치할 샤드 계산
- 검색요청 수행
  - 색인을 담은 샤드가 위치한 모든 노드에 쿼리 전달
  - 질의와 일치하는 다큐먼트에 대한 최소정보 요청(식별자, 점수) => 분산과정
  - 집계노드가 결과를 정렬 후 필요한 다큐먼트 정보 요청 => 취합과정
  - 위 과정에서 샤드와 레플리카 사이에 자동으로 부하 분산


Shard 1
primary
Shard 2
primary
Elasticsearch Node
Forward to
leader
Application
Indexing request
Shard 1
replica
Shard 2
replica
Elasticsearch Node
Elasticsearch Cluster


Scatter phase
Gather phase
Shard 1
Elasticsearch Node
Application
Results
Query
Shard 2
Elasticsearch Node
Elasticsearch Cluster

클러스터 설치와 구성

- 일래스틱 서치 설치
  - Java 설치: Java 6 이상, Java 7 권장
  - 일래스틱 서치 다운로드: Link
  - 설치할 장소에 압축 해제
  - 리눅스에서는 바이너리 패키지로 설치 가능

- 일래스틱 설치 구성
  - 시작이 쉬움: 합리적인 기본값과 자동 설정값 제공
  - 설정파일: config/elasticsearch.yml
  - 로깅설정파일: config/logging.yml
  - 실행 중 설정값 변경 가능(cluster.name, node.name 제외)

- 일래스틱 설치 구성(2)
  - 동일 클러스터 이름으로 구성된 노드는 하나의 클러스터로 통합 시도함
  - node.name을 지정하지 않을 경우 자동으로 이름 선택
  - 자동 이름은 재시동 과정에서 변경될 수 있음

- logging.yml
  - 얼마나 많은 정보를 로그에 저장할지 명세
  - 로그파일 정의
  - 주기적으로 새 파일을 생성하기 위한 규칙 정의

- elasticsearch.yml
  - 운영체제 조율: file descriptor (32000개)
  - JVM heap memory 제약(ES_HEAP_SIZE)

디렉터리 구조

디렉터리	설명
bin	인스턴스 구동 및 플러그인 관리 스크립트
config	구성 파일
lib	사용하는 라이브러리
data	모든 자료 저장
logs	이벤트와 오류에 대한 정보파일 저장
plugins	설치된 플러그인 저장
work	임시파일 저장

- 일래스틱 서치 실행
  - 실행파일: bin/elasticsearch
  - port 9200: HTTP REST API 포트
  - port 9300: 클러스터 내부, 자바 클라이언트 통신 포트
  - 다른 디렉터리에서 실행하면 자동으로 클러스터 형성

- REST API
  - JSON 형식으로 결과 받음
  - curl을 이용한 쿼리: curl -XGET localhost:9200/
  - 노드 정보 얻기: localhost:9200/
  - 클러스터 상태 점검: /_cluster/health?pretty

- 일래스틱서치 중단
  - 콘솔에서: Ctrl + c
  - kill signal
  - REST API: POST /_cluster/nodes/_shutdown
  - 단일노드 중단: POST /_cluster/nodes/[node id]/_shutdown
  - 노드 id 얻기: GET /_cluster/state/nodes

- 시스템 서비스로 설치
  - 패키지로 설치한 경우: 이미 되어 있음
  - 서비스 wrapper 이용: Link
  - sudo bin/service/elasticsearch install

REST API로 자료 처리

- REST API
  - REpresentational State Transfer
  - 모든 요청은 주소의 일부가 지정하는 실체가 있는 객체로 전달
  - /books/1/chapter/6

- REST API의 CRUD
  - Create: PUT, 객체 생성
  - Retrieve: GET, 현재 상태 얻기
  - Update: POST, 객체 변경
  - Delete: DELETE, 객체 제거

- NoSQL DB를 사용하는 방법과 유사

일래스틱서치에 자료 저장

- 새로운 다큐먼트 생성
  - JSON으로 다큐먼트 표현

```
{ “id": "1",
   "title": "New version of Elasticsearch released!",
   "content": "Version 1.0 released today!",
   "priority": 10,
   "tags": ["announce", "elasticsearch", “release"] }
```

  - PUT /blog/article/1 -d ‘[JSON]’
  - 응답

```
{ “_index”:"blog",   “_type":"article",
  “_id”:"1",              “_version”:1,
  “created”: true }
```

  - 자동 식별자 생성: POST /blog/article/ -d ‘[JSON]’

- 다큐먼트 인출
  - GET /blog/article/1
  - 성공 시:

{ "_index" : “blog", "_type" : "article",
"_id" : “1", "_version" : 1,
"exists" : true,
"_source" : {[document 내용]}
}

- 실패 시:

{ "_index" : “blog", "_type" : "article",
"_id" : “9999", "exists" : false }

- 다큐먼트 갱신
  - 복잡한 작업
    1. 다큐먼트를 가져와 _source 필드에서 자료 얻기
    2. 예전 다큐먼트 삭제
    3. 변경사항을 _source 필드에 적용
    4. 새로운 다큐먼트 색인
  - POST /blog/article/1/_update -d ‘{
    “script”: “ctx._source.content = \”new content\”” }’
    => 변경 대상만 보내면 됨
  - 응답: { "_index" : “blog", "_type" : "article",
    "_id" : “1", "_version" : 2 }
    => 기사 내용과 버젼 정보 변경
  - 아직 없을 경우 기본값 지정 가능: {“upsert”: { “counter”:0} }
  - 동적 스크립팅 켜야 동작함(1.6.X 부터 보안 문제로 disable)

- 다큐먼트 삭제
  - DELETE /blog/article/1
  - 응답: { “found”: true, "_index" : “blog",
    "_type" : “article", "_id" : “1", "_version" : 2 }

- 버전
  - 다큐먼트가 추가/변경/삭제 될 때 버젼을 증가시킴
  - 낙관적인 잠금(optimistic locking) 구현에 도움
  - 특정 버젼 조건에서만 삭제
    DELETE /library/book/1?version=1
    주어진 버젼과 다를 경우 오류 반환
  - 외부 시스템 제공 버젼 사용
    PUT /library/book/1?version=1234 -d [JSON]

URI 요청 질의를 사용한 검색

- 샘플 데이터
  - POST /books/es/1
  {"title":"ElasticsearchServer", "published": 2013}
  - POST /books/es/2
  {"title":"Mastering Elasticsearch", "published": 2013}
  - POST /books/solr/1
  {"title":"Apache Solr 4 Cookbook", "published": 2012}

- 매핑 결과 확인: GET /books/_mapping

- URI 요청
  - 특정 색인/타입 검색: GET /books/_search, GET /books/es/_search
  - 여러 색인 검색: GET /books,clients/_search
  - 모든 색인 검색: GET /_search

- 질의 응답
  - GET /books/_search?q=title:elasticsearch
  {
  "took" : 4, "timed_out" : false,
  "_shards" : { "total" : 5, "successful" : 5, "failed" : 0 },
  "hits" : { "total" : 2,   "max_score" : 0.625,
    "hits" : [
      { "_index" : "books", "_type" : "es","_id" : "1", "_score" : 0.625, "_source" : {"title":"Elasticsearch Server", "published": 2013}},
      {"_index" : "books", "_type" : "es","_id" : "2", "_score" : 0.19178301, "_source" : {“title":"Mastering Elasticsearch", "published": 2013}}] }
  }

- 질의 분석
  - URI요청 질의는 query string으로 매핑되어 분석
  - 색인분석 API: 분석 과정 진행을 보여줌

  GET /books/_analyze?field=title -d ‘Elasticsearch server’
  - 응답

  { "tokens" : [
    {"token" : "elasticsearch", "start_offset" : 0,
     "end_offset" : 13, "type" : "<ALPHANUM>",
     "position" : 1 },
    {"token" : "server", "start_offset" : 14,
     "end_offset" : 20, "type" : "<ALPHANUM>",
     "position" : 2}]
  }

- URI 질의 문자열 매개변수
  - 각 매개변수는 & 문자로 결합

- 예제쿼리
  GET /books/_search?q=published:
  2013&df=title&explain=true&default_operation=AND

- q
  - 일치하는 다큐먼트를 찾기위한 질의
  - Lucene 질의구문 사용

- df
  - 기본 검색 필드 명세
  - 기본 값으로 all 필드를 사용

- Analyzer: 질의 분석에 사용될 분석기 이름 정의

- default_operator
  - 질의에 사용될 기본 부울 연산자 명세
  - 기본값은 OR

- explain: true일 경우 결과의 다큐먼트마다 질의 해설 정보 추가

- field: 반환될 필드 명세, 기본값은 _source

- sort
  - 독자적인 정렬방식 명세
  - sort=published:desc : published 필드를 기준으로 오름차순

- timeout
  - 검색 타임아웃 설정
  - 해당 시점까지 수집한 결과 반환

- 결과 윈도우: size, from
  - size: 반환될 최대 크기
  - from: 결과가 반환해야하는 다큐먼트 시작 위치

- search_type: 검색 타입 => 3장에서...

- lowecase_expanded_term: 확장 키워드 소문자 변환

- analyze_wildcard: 와일드 카드 질의와 접두어 분석