Kafka를 활용한 Elasticsearch 실무프로젝트 소개

2019. 3. 26

이은학

Agenda

01	02	03	04	05
We are	**Project Case**	**Audit**	**Privacy**	**Connect BigData ecosystem**
하는일	프로젝트 사례 - 활용 영역 - 설계 & 구축 - 아키텍처 - 데이터 파이프라인	사용자쿼리기록 - 남기는 법 - 문제점 - 해결책 - 활용	개인정보 비식별화 - Logstash Ruby Filter - 성명 - 카드번호, 주민번호 - 주소	빅데이터 에코시스템 연계 - ES <-> Hadoop 연계 - 명사 별도 적재

Data Solution Team

하는 일

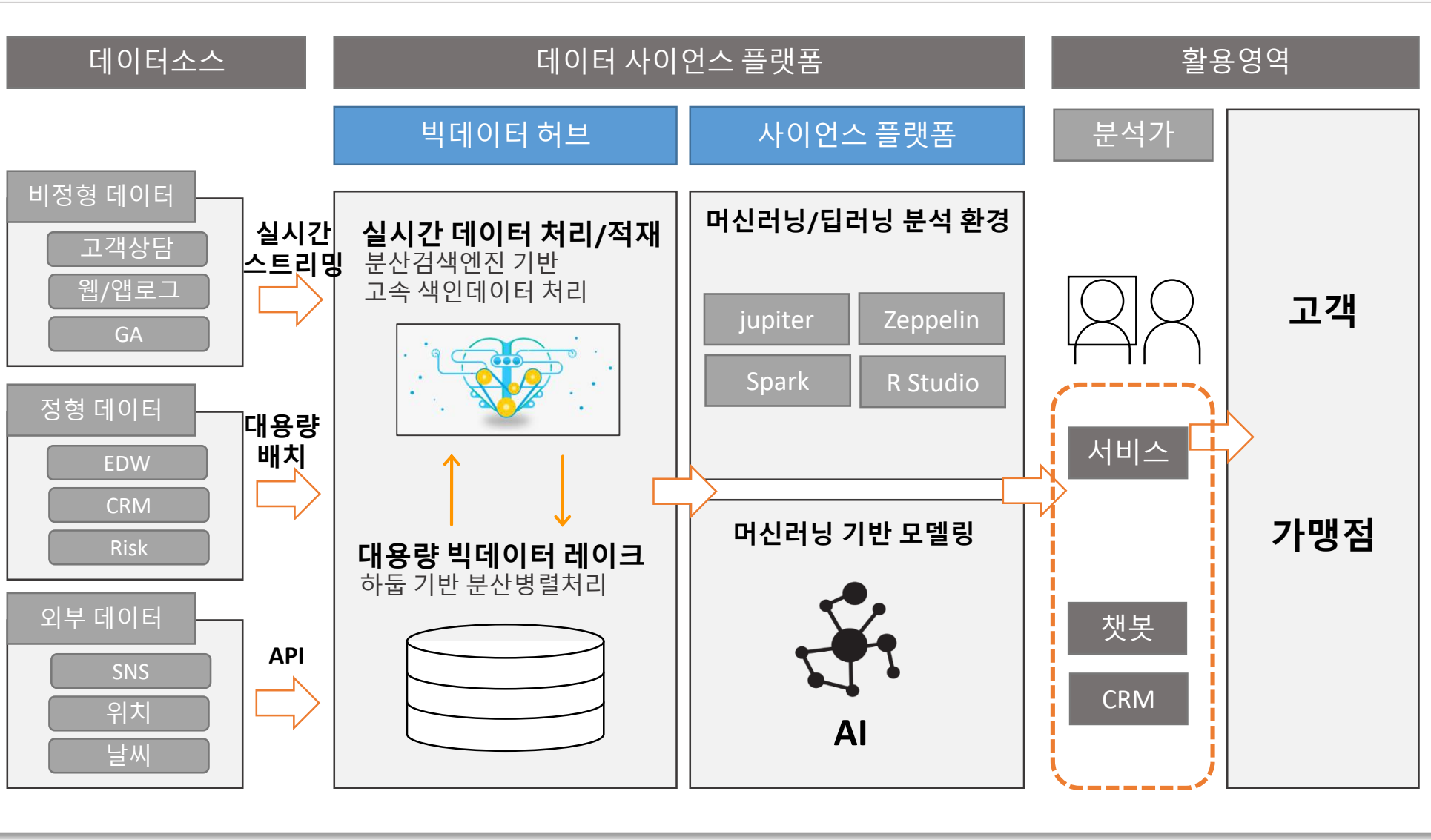
빅데이터 플랫폼 구축 및 활용솔루션 개발

Elasticsearch 클러스터 운영

EC Seoul 운영

* Seoul Region 의 Elasticsearch SaaS

프로젝트 개요


데이터소스
데이터 사이언스 플랫폼
활용영역
빅데이터 허브
사이언스 플랫폼
분석가
비정형 데이터
고객상담
웹/앱로그
GA
정형 데이터
EDW
CRM
Risk
외부 데이터
SNS
위치
날씨
실시간
스트리밍
대용량
배치
API
실시간 데이터 처리/적재
분산검색엔진 기반
고속 색인데이터 처리
대용량 빅데이터 레이크
하둡 기반 분산병렬처리
머신러닝/딥러닝 분석 환경
jupiter
Zeppelin
Spark
R Studio
머신러닝 기반 모델링
AI
서비스
챗봇
CRM
고객
가맹점

프로젝트내 Elastic Stack 활용 영역

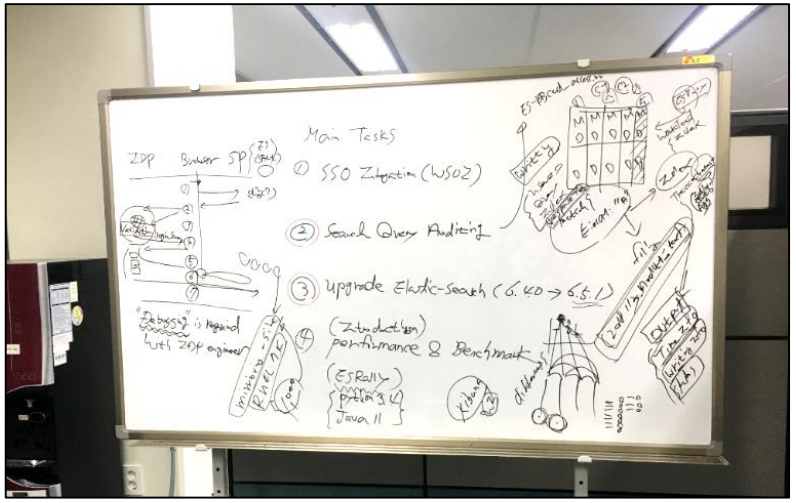
- **비정형데이터 분석**
  - Google Analytics
  - 고객 상담데이터(STT) ➡ * RESTFul 한글 형태소 분석기(Nori)
  - 웹/앱 로그

- **3rd Party Solution APM**
  - Docker Host
  - Docker Containers

  분석도구
  (R, Jupiter, zeppelin 등)

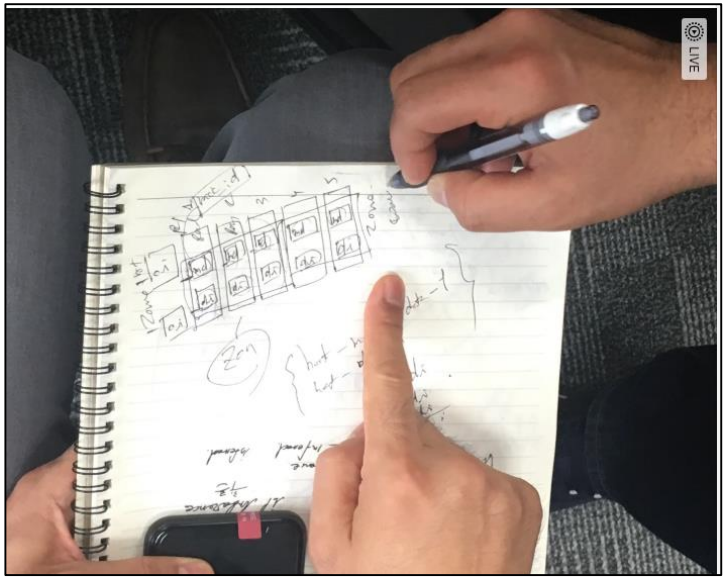
  - Apache Server Error

  Wiki 등

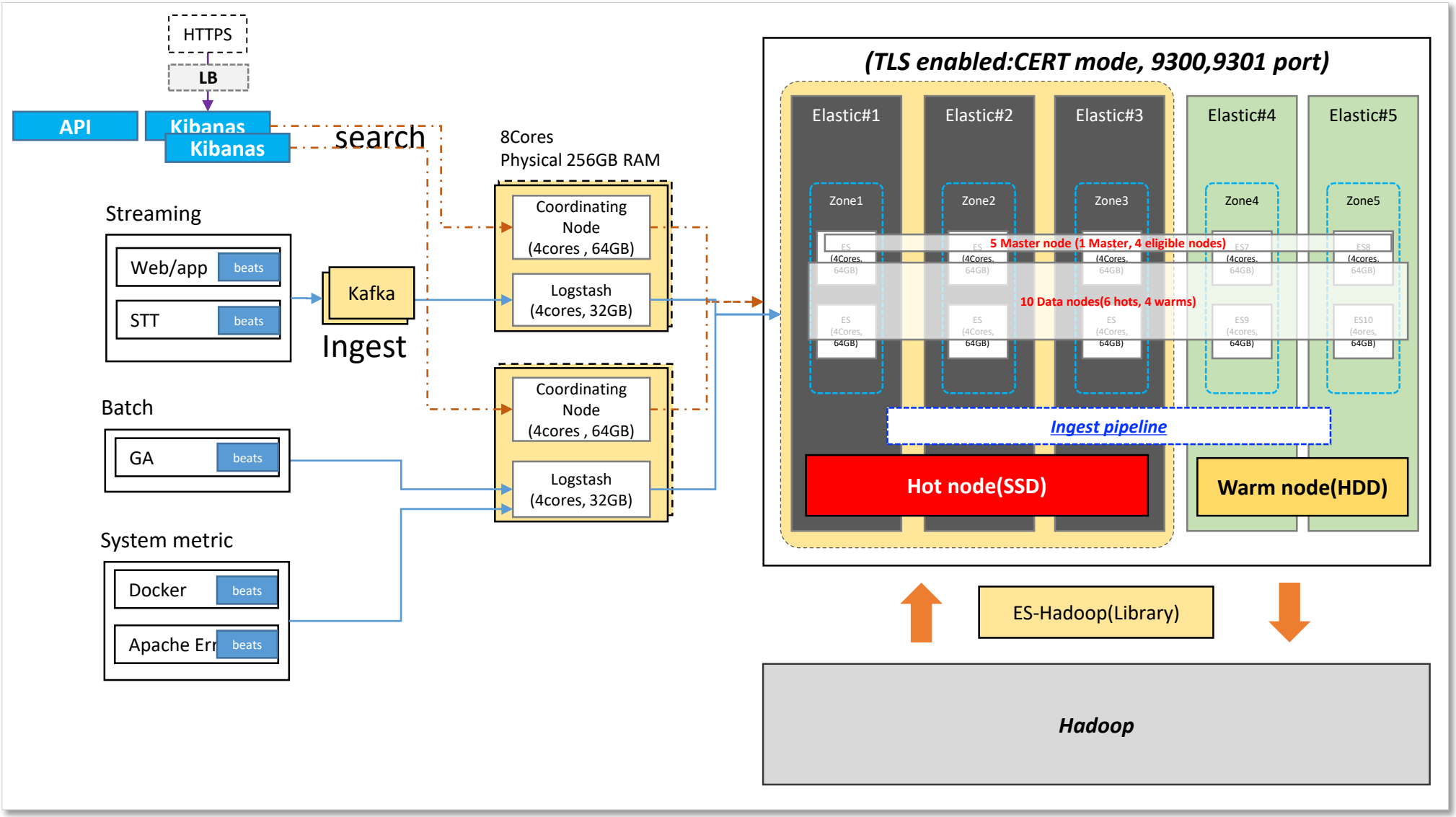


클러스터 설계 방향

- Index Lifecycle 을 고려한 Hot/Warm 아키텍쳐
  (HA를 위해 Index가 동일 서버에 저장되지 않도록)
- 각 영역의 노드별로 확장 가능한 아키텍쳐
  (필요한 경우 각 영역에 노드 추가 용이)
- 가능한 단순한 아키텍쳐 지향
  (유지보수 편의)

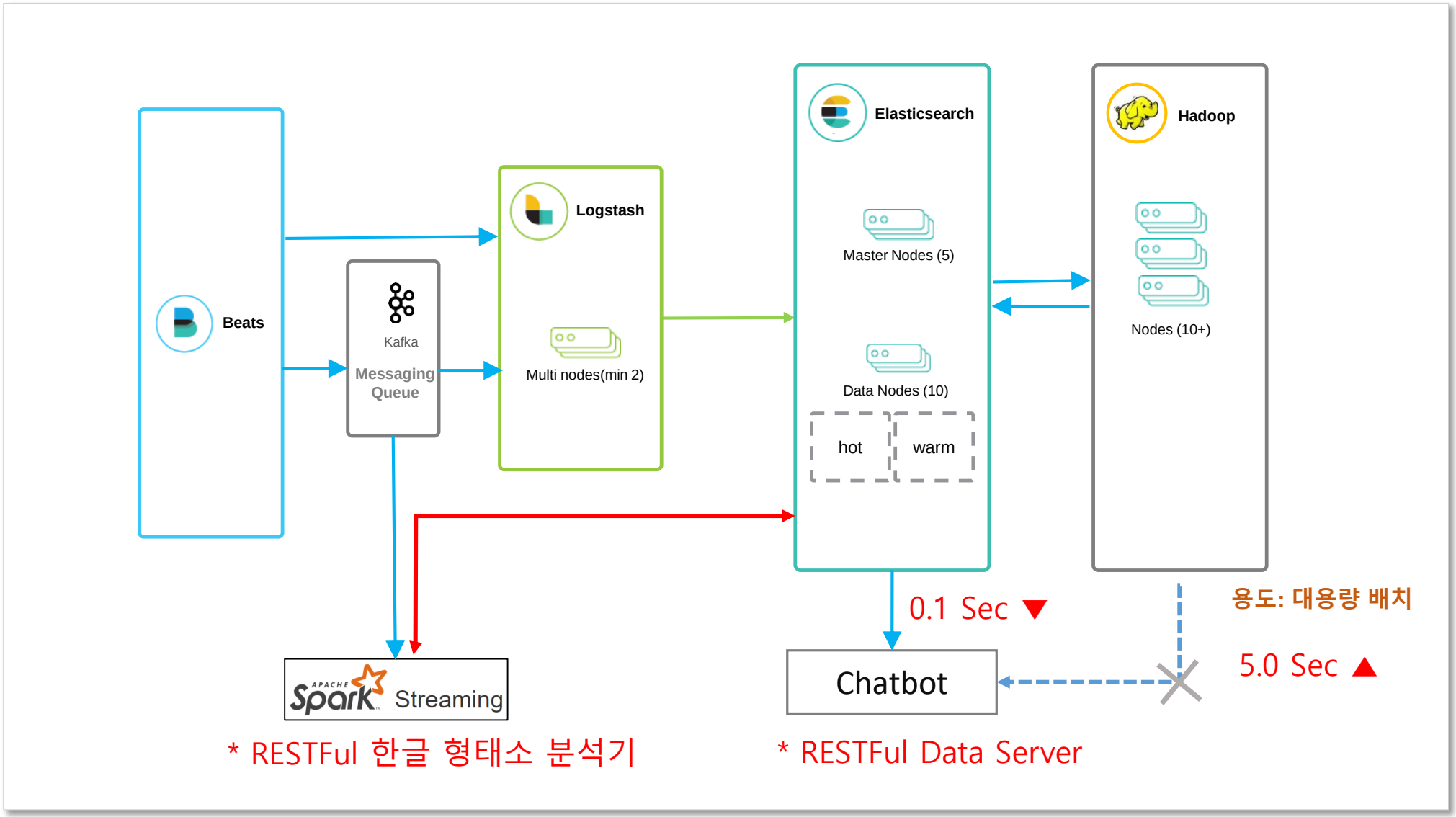


Architecture



kafka KRU

Data Pipeline



이슈 해결 과정

프로젝트 진행시 발생한 이슈들의 해결 과정

Audit – 사용자쿼리 기록

요구사항 : 모든 사용자의 쿼리는 이력 관리가 필요함(금융감독원 감사)

이슈 : Query, UserID, @timestamp 동시에 남아야 함

* 6.5.1 부터 UserID 를 남길 수 있음
(2018.11)

Audit – 기록 방법

- elasticsearch.yml

```
xpack.security.audit.enabled: true
xpack.security.audit.outputs: [~~index~~, logfile]

xpack.security.audit.logfile.events.ignore_filters:
  example1:
    users: ["kibana", "admin_user", "elastic"]
```

❶ logfile로 남겨야 userID가 남음

❷ 제외할 ID설정(이 부분 없을 시 10node 기준 일 아무 이벤트 없어도 160GB 발생)

*클러스터 노드 간의 통신도 기록

Audit – 적재 방법

- audit1.conf

```
filter {
  json {
    source => "message"
  }
  if [message] !~ "url.path" {
    drop {}
  } else {
    mutate {
      split => { "url.path" => "/" }
    }
    if [url.path][1] == "_xpack" {
      if [url.path][2] != "sql" {
        drop {}
      }
      else if [url.path][1] == "_aliases" { drop {} }
      else if [url.path][1] == "_mapping" { drop {} }
      else if [url.path][1] == "_template" { drop {} }
      mutate { join => { "url.path" => "/"} }
    }
    mutate {
      join => { "url.path" => "/" }
    }
  }
  mutate { remove_field => ["message" ] }
}
```

```
input {
  beats {
    port => 5046
    type => json
  }
}

output {
  elasticsearch {
    hosts => ["https://127.0.0.1:9200"]
    index => “audit-%{+yyyy.MM.dd}"
    user => “*******"
    password => “********"
    ssl => true
    ssl_certificate_verification => false
  }
}
```

정책에 맞게 필터링

Audit – 활용 사례(1)

- 건수 확인

```
curl -XGET http://localhost:9200/sampledata-2019/_count
```

```
{
  "count" : 300000
}
```

- 조회(페이징)

```
curl -XGET "http://localhost:9200/sampledata-2019/_search" -H
'Content-Type: application/json' -d'
{
  "sort": [
    {
      "@timestamp": {
        "order": "asc"
      }
    }
  ],
  "from": 0, #(10, 20, 30, 40 ...)
  "size": 10
}'
```

```
{
  "took" : 56,
  "timed_out" : false,
  "_shards" : {...},
  "hits" : {
    "total" : 300000,
    "max_score" : null,
    "hits" : [
      {...},
      {...},
      {...},
      ...
    ]
  }
}
```

Audit – 활용 사례(2)

- 페이징 사용시 주의사항

```
{
  "error": {
    "root_cause": [
      {
        "type": "query_phase_execution_exception",
        "reason": "Result window is too large, from + size must be less than or equal to: [10000] but was [300000].
See the scroll api for a more efficient way to request large data sets.
This limit can be set by changing the [index.max_result_window] index level setting."
      }
    ],
...
}
```

- 설정: max_result_window

```
curl -XPUT "http://localhost:9200/basic-logstash-2016/_settings" -H 'Content-Type: application/json' -d'
{
  "index": {
    "max_result_window": 300000
  }
}‘
```

- 해제 : max_result_window

```
curl -XPUT "http://localhost:9200/basic-logstash-2016/_settings" -H 'Content-Type: application/json' -d'
{
  "index": {
    "max_result_window": null
  }
}‘
```

개인정보 비식별화

요구사항 : 개인정보는 반드시 비식별처리 되어야 한다.

이슈 : 비정형 테이터의 특성상 개인정보의 식별범위 모호

- 예문

> 안녕하세요 홍길동 고객님 용산구 갈월동에 거주하시나요?
> 제 카드 번호는 삼사오칠 다시 구구팔사 칠육오삼....

- 패턴

> 성명 : XXX 회원님, XXX 님
> 카드번호, 주민번호 : 연속되는 숫자(영|일|이|삼|사|오|육|칠|팔|구|십|열)

- 치환

> 주소 : 행정구역 단어 20,000 여건
> 서울특별시, 강남구, 테헤란로, 갈라빌딩, ...

로직처리 가능

개인정보 비식별화 – Ruby Filter(1)

- stt.rb(ruby 필터 작성)

```
...

def filter(event)
      ratogtxt = event.get('ratogtxt')
      ratogtxt = ratogtxt&.gsub(/\s[가-힣]+\s고객님/) { |match| ("*" * match.length) + "고객님" }
      ratogtxt = ratogtxt&.gsub(/\s[가-힣]+\s님/) { |match| ("*" * match.length) + "님" }

      ratogtxt = ratogtxt.gsub(/\s[공|영|일|이|삼|사|오|육|칠|팔|구|십|열|\s]{4, 13}/) { |match| "*" * match.length }

      replacements = [
         ["...도로명...", "*"],
         ["가가로", "*"],
         ["충남", "*"],
         ["충청북도", "*"],
         ["충북", "*"]
      ]

      replacements.each { |replacement| ratogtxt&.gsub!(replacement[0], replacement[1]) }

      event.set('ratogtxt', ratogtxt)
      return [event]

End
...
```

개인정보 비식별화 – Ruby Filter(2)

- stt.conf(logstash 적용)

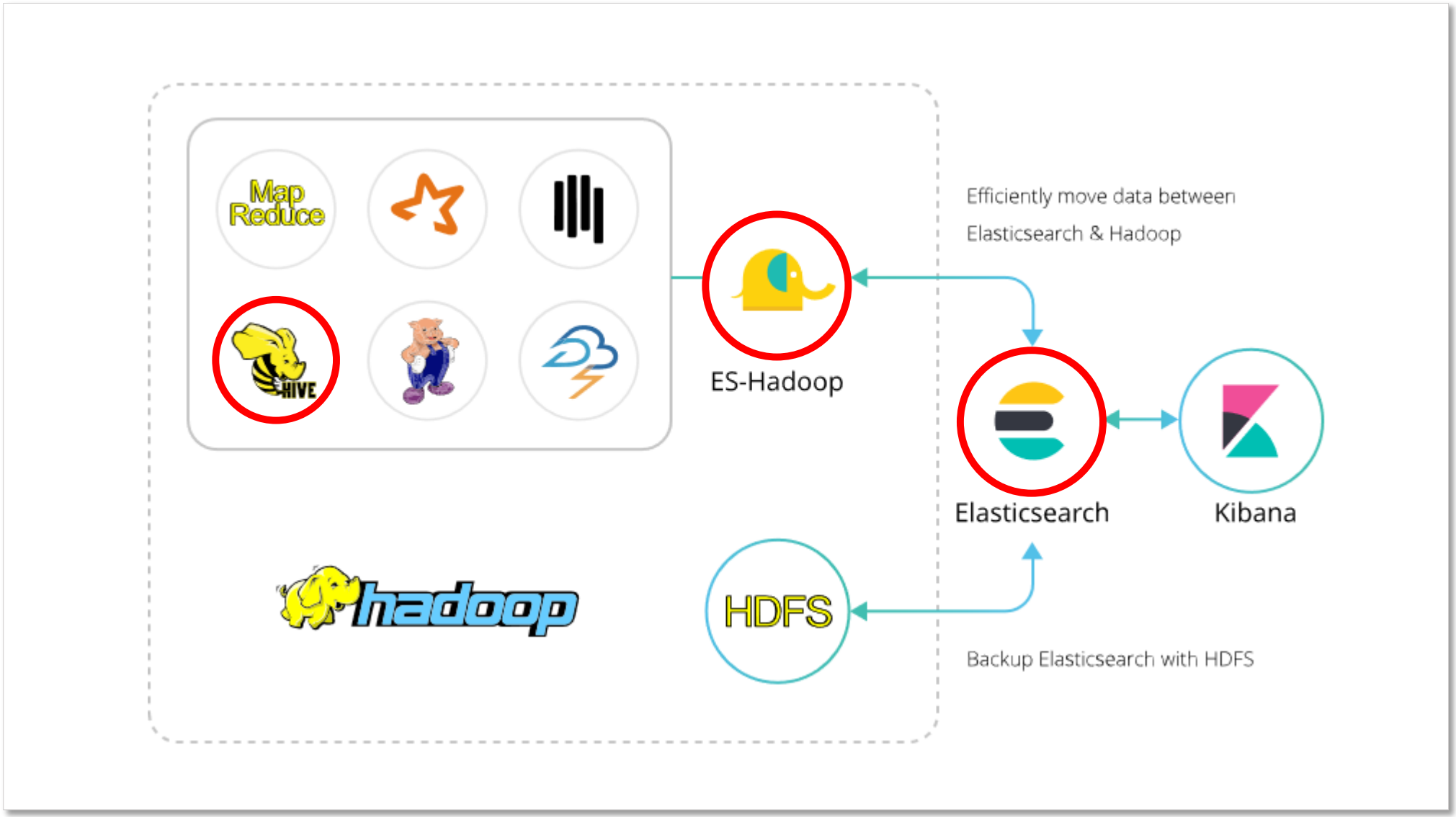
```
input {
  stdin {}
}

filter {
  csv {
    columns => ["ratogtxt", "sub"]
    separator => ","
  }
}

filter {
  ruby {
    path => "/home/ec2-user/logstash-6.5.4/ruby/stt.rb"
    script_params => { "param1" => "" }
  }
}

output {
  stdout {}
}
```

빅데이터 에코시스템 연계 – Elasticsearch-Hadoop



빅데이터 에코시스템 연계 – Elasticsearch-Hadoop 사례

Elasticsearch

Hadoop

- 작업절차

STEP1 : hive 테이블 생성

STEP2 : stage 테이블 삭제

STEP3 : stage 테이블 생성  포인터

STEP4 : stage 테이블 데이터 hive 테이블 입력

STEP5 : stage 테이블 삭제

빅데이터 에코시스템 연계 – Elasticsearch-Hadoop 사례

ES > Hadoop : create_weblog.hql

```
set hive.exec.dynamic.partition.mode=nonstrict;
set hive.exec.dynamic.partition=true;
create external table if not exists eshadoop.weblog (
                    datetime string,
                    ip string,
                    extension string,
                    response string,
                    agent string,
                    bytes           bigint,
                    url string,
                    os string,
                    tags array<string>
)
partitioned by (
                     p_year string,
                    p_month string,
                    p_date string,
                    p_hour string
)
stored as parquet
location "s3://eshadoop-demo/weblog";
```

STEP1 : hive 테이블 생성

STEP2 : stage 테이블 삭제

STEP3 : stage 테이블 생성  포인터

STEP4 : stage 테이블 데이터 hive 테이블 입력

STEP5 : stage 테이블 삭제

빅데이터 에코시스템 연계 – Elasticsearch-Hadoop 사례

ES > Hadoop : drop_weblog_stg.hql

```
drop table if exists eshadoop.weblog_stg;
```

STEP1 : hive 테이블 생성

STEP2 : stage 테이블 삭제

STEP3 : stage 테이블 생성

포인터

STEP4 : stage 테이블 데이터 hive 테이블 입력

STEP5 : stage 테이블 삭제

빅데이터 에코시스템 연계 – Elasticsearch-Hadoop 사례

ES -> Hadoop : create_weblog_stg.hql

```
create external table eshadoop.weblog_stg (
  datetime string,  ip string, extension string, response string, agent string, bytes bigint, url string, os string, tags array<string>
)
STORED BY 'org.elasticsearch.hadoop.hive.EsStorageHandler'
TBLPROPERTIES (
  'es.nodes' = '172.31.15.196',
  'es.nodes.wan.only' = 'true',
  'es.resource' = 'weblog-2019.02.21/log',
  'es.query' = '?q=*',
  'es.index.auto.create' = 'no',
  'es.index.read.missing.as.empty' = 'yes',
  'es.field.read.empty.as.null' = 'yes',
  'es.mapping.date.rich' = 'false',
  'es.mapping.names' = 'datetime:@timestamp, ip:ip, extension:extension, response:response, agent:agent,
                        bytes:bytes, url:url, os:machine.os, tags:@tags'
);
```

INDEX 지정

STEP1 : hive 테이블 생성

STEP2 : stage 테이블 삭제

STEP3 : stage 테이블 생성 포인터

STEP4 : stage 테이블 데이터 hive 테이블 입력

STEP5 : stage 테이블 삭제

빅데이터 에코시스템 연계 – Elasticsearch-Hadoop 사례

ES > Hadoop : insert_overwrite_weblog.hql

```
set hive.exec.dynamic.partition.mode=nonstrict;
insert overwrite table eshadoop.weblog
partition(p_year, p_month, p_date, p_hour)
select
                datetime,
                ip,
                extension,
                response,
                agent,
                bytes,
                url,
                os,
                tags,
                substr(datetime, 1, 4),
                substr(datetime, 6, 2),
                substr(datetime, 9, 2),
                substr(datetime, 12, 2)
from eshadoop.weblog_stg;
```

N번 실행시 중복 방지

STEP1 : hive 테이블 생성

STEP2 : stage 테이블 삭제

STEP3 : stage 테이블 생성

포인터

STEP4 : stage 테이블 데이터 hive 테이블 입력

STEP5 : stage 테이블 삭제

빅데이터 에코시스템 연계 – Elasticsearch-Hadoop 사례

ES > Hadoop : drop_weblog_stg.hql

```
drop table if exists eshadoop.weblog_stg;
```

STEP1 : hive 테이블 생성

STEP2 : stage 테이블 삭제

STEP3 : stage 테이블 생성

포인터

STEP4 : stage 테이블 데이터 hive 테이블 입력

STEP5 : stage 테이블 삭제

빅데이터 에코시스템 연계 – Elasticsearch-Hadoop 사례

Hadoop

Elasticsearch

- 작업절차

STEP1 : stage 테이블 삭제

STEP2 : stage 테이블 생성  포인터

STEP3 : hive 테이블 데이터 stage 테이블 입력

STEP4 : stage 테이블 삭제

빅데이터 에코시스템 연계 – Elasticsearch-Hadoop 사례

Hadoop > ES : drop_applog_stg.hql

```
drop table if exists eshadoop.applog_stg;
```

STEP1 : stage 테이블 삭제

STEP2 : stage 테이블 생성

포인터

STEP3 : hive 테이블 데이터 stage 테이블 입력

STEP4 : stage 테이블 삭제

빅데이터 에코시스템 연계 – Elasticsearch-Hadoop 사례

Hadoop > ES : create_applog_stg.hql

```
create external table if not exists eshadoop.applog_stg (
  datetime string, ip string, extension string, response string, agent string, bytes bigint, url string, os string, tags array<string>
)
STORED BY 'org.elasticsearch.hadoop.hive.EsStorageHandler'
TBLPROPERTIES (
  'es.nodes' = '172.31.15.196',
  'es.nodes.wan.only' = 'true',
  'es.resource' = 'applog-2015.05.18/log',
  'es.query' = '?q=*',
  'es.index.auto.create' = 'yes',
  'es.index.read.missing.as.empty' = 'yes',
  'es.field.read.empty.as.null' = 'yes',
  'es.mapping.date.rich' = 'false',
  'es.mapping.names' = '
                    datetime:@timestamp,
                    ip:ip,
                    extension:extension,
                    response:response,
                    agent:agent,
                    bytes:bytes,
                    url:url,
                    os:machine.os,
                    tags:@tags'
);
```

INDEX 지정

STEP1 : stage 테이블 삭제

STEP2 : stage 테이블 생성 포인터

STEP3 : hive 테이블 데이터 stage 테이블 입력

STEP4 : stage 테이블 삭제

빅데이터 에코시스템 연계 – Elasticsearch-Hadoop 사례

Hadoop > ES : insert_overwrite_applog_stg.hql

```
insert overwrite table eshadoop.applog_stg
select
                datetime,
                ip,
                extension,
                response,
                agent,
                bytes,
                url,
                os,
                tags
from eshadoop.applog
where p_year = '2015'
   and p_month = '05'
   and p_date = '18';
```

STEP1 : stage 테이블 삭제

STEP2 : stage 테이블 생성  포인터

STEP3 : hive 테이블 데이터 stage 테이블 입력

STEP4 : stage 테이블 삭제

빅데이터 에코시스템 연계 – Elasticsearch-Hadoop 사례

Hadoop > ES : drop_applog_stg.hql

```
drop table if exists eshadoop.applog_stg;
```

STEP1 : stage 테이블 삭제

STEP2 : stage 테이블 생성

포인터

STEP3 : hive 테이블 데이터 stage 테이블 입력

STEP4 : stage 테이블 삭제

빅데이터 에코시스템 연계 – 명사 별도 적재사례(1)

요구사항 : 적재된 한글 문장 데이터 중 명사만 별도의 field로 적재가 필요

이유 : Nori 형태소 분석 결과를 타 빅데이터 시스템에서의 활용
REST Call 을 하는 것이 번거로우니 Hive 테이블 컬럼상에 적재 해줬으면 함

빅데이터 에코시스템 연계 – 명사 별도 적재사례(2)

아이디어 1 : INDEX 데이터는 날짜별로 쌓인다(stt-2019.02.21)

아이디어 2 : Logstash를 통해 ReIndex 가 가능하다

아이디어 3 : Logstash는 Cron 형태의 배치도 가능하다

- 처리과정

STEP1 : INDEX 조회시 _source, 토큰 동시조회

STEP2 : 동일 INDEX에 _source, 토큰 업데이트

STEP3 : hive 적재

빅데이터 에코시스템 연계 – 명사 별도 적재사례(3)

- Index 생성

```
PUT nori_sample
{
 "settings": {
  "index": {
   "analysis": {
    "tokenizer": {
     "nori_user_dict": {
      "type": "nori_tokenizer",
      "decompound_mode": "mixed",
      "user_dictionary": "userdict_ko.txt"
     }
    },
    "analyzer": {
     "my_analyzer": {
      "type": "custom",
      "tokenizer": "nori_user_dict",
      "filter": ["unnoun_filter","lowercase"],
      "fielddata": true
     }
    },
    "filter": {
     "unnoun_filter": {
      "type": "nori_part_of_speech",
      "stoptags": ["E", "IC", "J", "MAG",
```

Lucene Part of speech tags 참조

```
"MM", "SP", "SSC", "SSO", "SC","SE", "XPN",
"XSA", "XSN", "XSV", "UNA", "NA", "VSV",
"VA", "E"]
     }
    }
   }
  }
 },
 "mappings": {
  "_doc" : {
   "properties": {
    "msg": {
     "type": "text",
     "analyzer": "my_analyzer",
     "fielddata": true
    }
   }
  }
 }
}
```

품사

토큰조회

빅데이터 에코시스템 연계 – 명사 별도 적재사례(4)

- 문장 입력

```
POST /nori_sample/_doc
{
   "msg" : "일래스틱서치 명사의 별도적재 예제입니다.“
}
```

- 문장 분석

```
GET /nori_sample/_analyze
{
   "analyzer": "my_analyzer",
   "text": "일래스틱서치 명사의 별도적재 예제입니다.“
}
```

```
{
  "tokens" : [
    {
      "token" : "일래스틱",
      "start_offset" : 0,
      "end_offset" : 4,
      "type" : "word",
      "position" : 0
    },
    {
      "token" : "서치",
      ...
    },
    {
      "token" : "명사",
      ...
    },
    {
      "token" : "별도",
      ...
    },
...
  ]
}
```

빅데이터 에코시스템 연계 – 명사 별도 적재사례(5)

- 형태소 분석 키워드 동시 조회

```
GET /nori_sample/_doc/_search
{
 "_source": "*",
 "query": {"match_all": {}},
 "script_fields": {
  "searchkeyword": {
   "script": {
    "lang": "painless",
    "source": "doc[params.field].values",
    "params": {"field": "msg"}
   }
  }
 }
}
```

```
{
 ...,
 "hits" : {
  "total" : 2, "max_score" : 1.0,
  "hits" : [
   {
    "_index" : "nori_sample",
    "_type" : "_doc",
    "_id" : "ISSx6mgBARmSfZozzHTU",
    "_score" : 1.0,
    "_source" : {
     "msg" : "일래스틱서치 명사의 별도 적재 이슈를 해
결 하자. "
    },
    " fields " : {
     " searchkeyword " : [ " 일래스틱서치 " , " 명사",
      "별도","적재", "이슈","활용"]
    }
   },
   {},
   {},
   ...
  ]
 }
}
```

빅데이터 에코시스템 연계 – 명사 별도 적재사례(6)

- Logstash를 통한 문서 update

```
input {
  elasticsearch {
    hosts => "$REPLACE_SOURCE_HOST_ADDR$"
    index => "<$REPLACE_SOURCE_INDEX_NAME$-{now/d-
1d{YYYY.MM.dd}}>"
    query => '{ "_source": "*", "query": { "match_all": {} },
"script_fields": { "terms": { "script": { "lang": "painless", "source":
"doc[params.field].values", "params": { "field" :
"$REPLACE_FIELD_KEY$" } } } } }'
    docinfo => true
    docinfo_fields => ["fields", "_index", "_type", "_id"]
    docinfo_target => "meta"
    schedule => "0 1 * * *"
  }
}

filter {
   mutate {
     copy => { "[meta][fields][terms]" => "terms" }
     copy => { "[meta][_index]" => "[@metadata][_index]" }
     copy => { "[meta][_type]" => "[@metadata][_type]" }
     copy => { "[meta][_id]" => "[@metadata][_id]" }
     remove_field => ["meta"]
   }
}
```

```
output {
  stdout {}
  elasticsearch {
    hosts => ["$REPLACE_TARGET_HOST_ADDR$"]
    index => "$REPLACE_TARGET_INDEX_NAME$-
%{[@metadata][_index]}"
    document_id => "%{[@metadata][_id]}"
    document_type => "%{[@metadata][_type]}"
    user => "$REPLACE_USERNAME$"
    password => "$REPLACE_PASSWORD$“
    document_id => "%{[@metadata][_id]}"
    doc_as_upsert => true
    action => "update"
  }
}
```

Github

https://github.com/leftwing871/elasticontour2019seoul

leftwing871@gmail.com

참고

Audit:
https://www.elastic.co/guide/en/elastic-stack-overview/current/auditing.html

Ruby filter plugin:
https://www.elastic.co/guide/en/logstash/current/plugins-filters-ruby.html

ES-Hadoop:
https://www.elastic.co/guide/en/elasticsearch/hadoop/current/hive.html

Lucene POS.Tag
http://lucene.apache.org/core/7_6_0/analyzers-nori/org/apache/lucene
/analysis/ko/POS.Tag.html

THANK YOU