시스템 운영 아키텍쳐
Logstash 설정과 트러블 슈팅

NBT

cash slide

NBT

김정환

얼마전 까지는
제품길드 DevOPS

이제부터는
제품길드 Server Class

Contents

NBT

logstash

유용한 설정들

NBT

json codec

```
input {
  file {
    codec => "json"
    path => "user.log"
    sincedb_path => ".sincedb-user.log"
    sincedb_write_interval => 1
    stat_interval => 1
    type => "raw"
  }
}
```

custom patterns

- patterns 폴더 밑 파일에 정의

```
RAPPNAME \[(?<appname>[\w]+)\]
RTIMEFORMAT \[(?<date>.{29})\]
RLEVEL \[(?<level>(debug¦fatal¦error¦warn¦info)?)\]
RLOG #std_logger=\{(?<msg>.*)\}
```

- grok 설정이 간단해짐

```
grok {
  patterns_dir => [ "logstash-2.3.2/patterns" ]
  match => { "message" =>
    "%{RAPPNAME} %{RTIMEFORMAT} %{RLEVEL} %{RLOG}" }
}
```

fingerprint

```
filter {
  fingerprint {
    source => [ "nickname" ]
    method => "SHA1"
    target => "nickname"
  }
}
```

if else

```
output {
  if "_grokparsefailure" in [tags] {
    file { path => "log/error_grokparsefalure.txt" }
    stdout { codec => rubydebug }
  }
  else {
    elasticsearch {
      hosts => [ "127.0.0.1" ]
      index => "log-%{+YYYY-MM-dd}"
    }
  }
}
```

if else

```
if ![app] or ![app][sdk_version_code] {
   mutate {
     add_field => { "[app][sdk_version_code]"
=> 1 }
   }
}
```

drop

```
filter {
  if [app][version_code] == 10269 {
    if [name] =~ "image_load::" {
      drop { }
    }
  }
}
```

metadata

```
filter {
  ruby {
    init => "require 'time'"
    code => "event['@metadata']['shard'] =
      (Time.parse(event['@timestamp'].to_s).hour / 4).to_i"
  }
}

output {
  elasticsearch {
    hosts => ["127.0.0.1"]
    index => "log-%{+YYYY-MM-dd}_%{[@metadata][shard]}"
  }
}
```

debug

```
output {
  stdout {
    codec => rubydebug {
      metadata => true
    }
  }
}
```

mutate

```
filter {
  mutate {
    remove_field => [ "married" ]
    convert => { "[version_code]" =>
"integer" }
    rename => { "level" => "[@metadata]
[level]" }
  }
}
```

if else & metadata

```
filter {
  ruby {
    init => "require 'time'"
    code => "event['@metadata']['shard'] =
      (Time.parse(event['@timestamp'].to_s).hour / 4).to_i"
  }
}
```

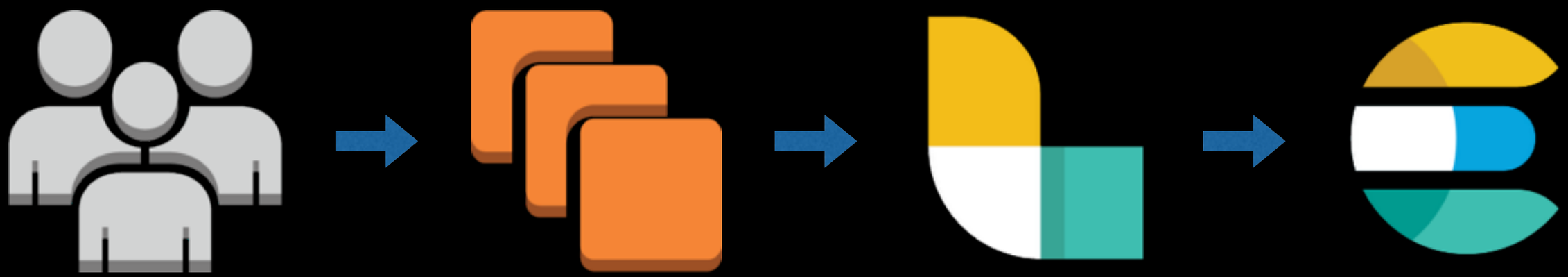
```
output {
  if "_grokparsefailure" in [tags] {
    file { path => "log/error_grokparsefalure.txt" }
    stdout { codec => rubydebug }
  }
  else {
    elasticsearch {
      hosts => [ "127.0.0.1" ]
      index => "log-%{+YYYY-MM-dd}_%{[@metadata][shard]}"
      document_type => "%{[@metadata][level]}"
    }
  }
}
```

elasticsearch

시스템 운영 아키텍쳐

NBT

구분

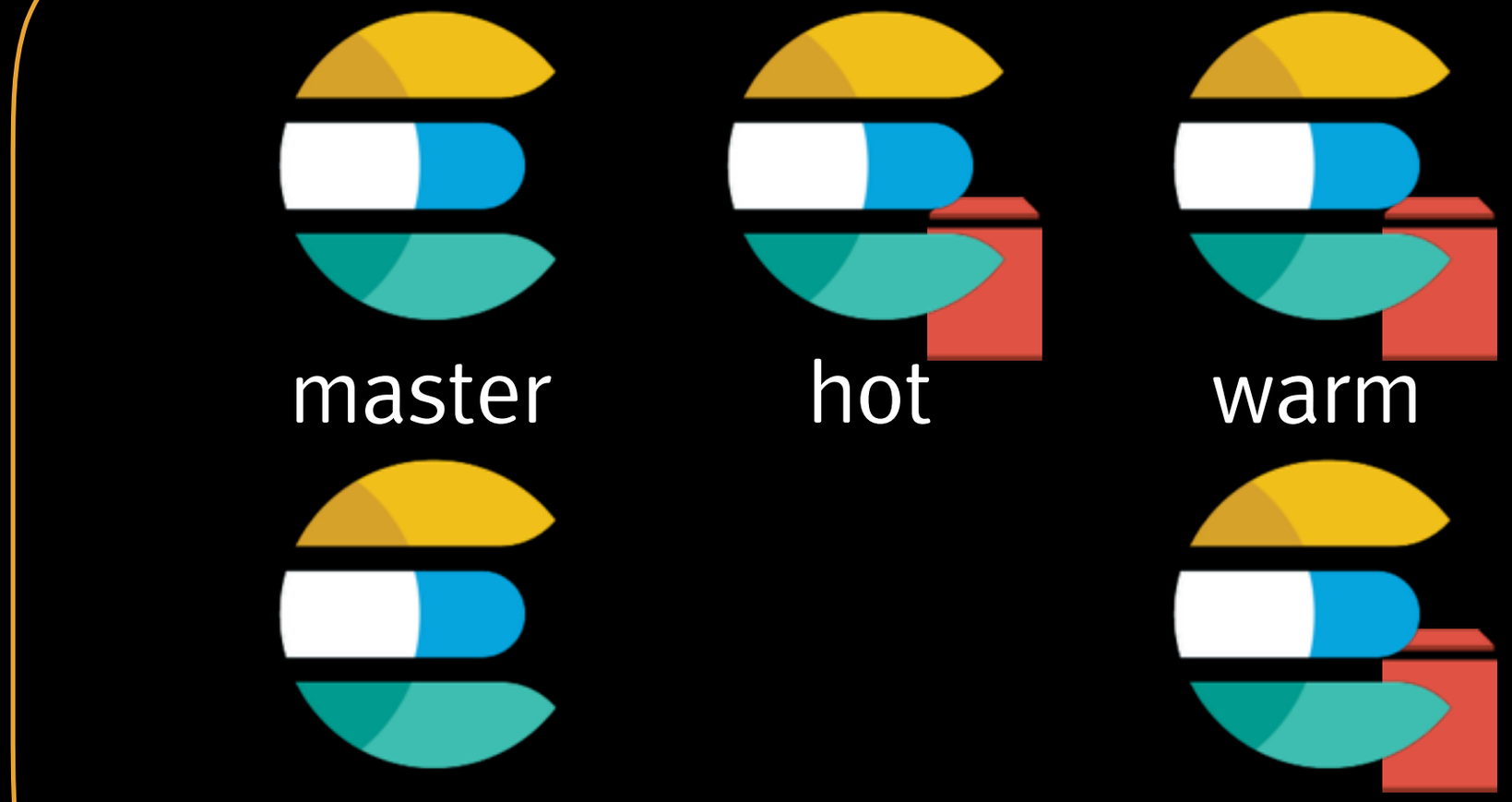


user log logstash elasticsearch

사용자 행동 데이터 저장

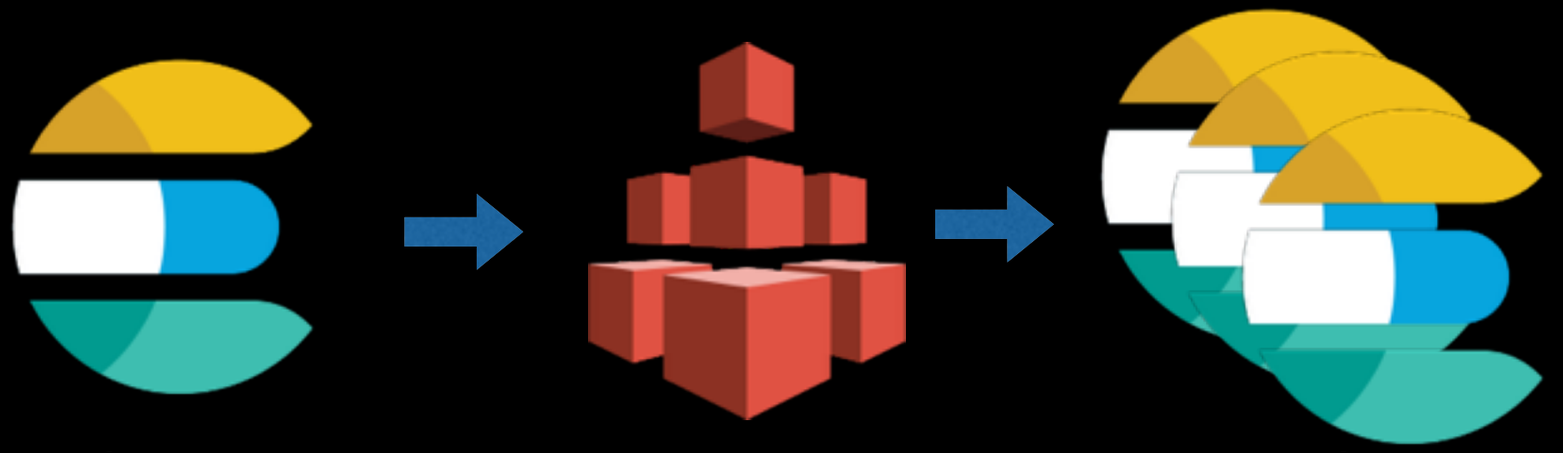
file + logstash + elasticsearch

NBT



관리, 저장, 조회 노드 분리

Master Node, Client Node, Data Node

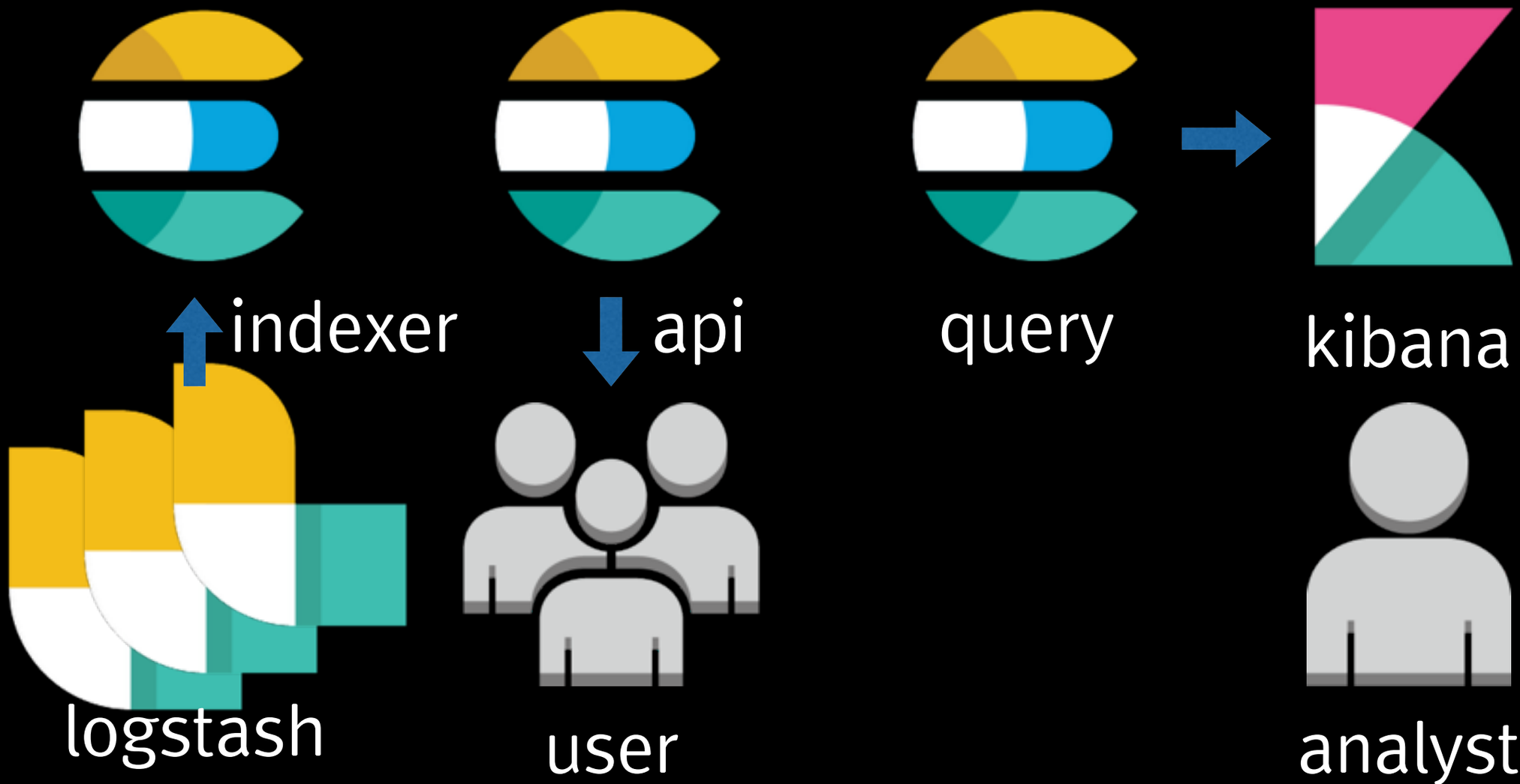


jenkins

갱신, 보관 노드 분리

hot , warm architecture

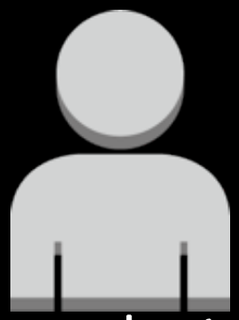
NBT



저장, 서비스, 분석 노드 분리

Client Node

전체 동작



정리

용도별 구분

- master , master-sub + Jenkins
- indexer , query + kibana , api
- fast
- box-click , box-storage , box-mission

데이터의 흐름

- master 노드는 전체 cluster 를 관리함
- log는 logstash 를 통해서 index 생성 노드로 전송
- hot 노드는 계속 변경되는 index 데이터를 갱신
- jenkins 는 시간에 따른 index 의 관리를 담당
- 생성된 snapshot data 는 warm, cold 노드로 복원
- 복원된 데이터는 alias 조정을 통해 조회 대상을 조정함

사전 질문

색인할 로그가 100G 리소스 할당 기준?

- 노드, 메모리 용량은 어느정도로 하고, 샤드 크기는 어느정도가 좋은지에 대한 질문들, 퍼포먼스 튜닝에 대한 질문들
- 그때 그때 다르기 때문에 정답을 드릴 수가 없습니다.

저의 경험

- 1일에 6개로 나눠보니 그럭저럭 괜찮음
- hot, warm 구조 분리는 필수
- elasticsearch 의 공식 가이드 문서를 최대한 적용
- index 의 data field 가 적절하게 구성되어 있는지?

아키텍처 운영 결과 개선 방향?

- 마스터 노드 분리
- hot , warm 분리
- 데이터 분석용 노드와 서비스용 노드를 분리하여 운영
- 로그성 데이터의 경우 거의 cold storage 인데 메모리를 항상 할당 하고 있어야 함
  - 오래 걸리는 응답이더라도 스왑이던 필요시 켜는 방법등을 구상할수는 없을지?

logstash 5.1.1 플러그인?

- 저도 아직 안해봐서…

HA 구성 best case

- master , master-sub 반드시 독립
- 데이터를 전송하는 노드와
  데이터를 질의 하는 노드 분리
- 각 type 별 3ea 이상의 node 구성
- 스토리지 용량은 최소 1대 이상의 장애를 대비해서 산정
- 실제 index 가 아닌 alias 를 사용하여 긴급 장애에 대처

HA 구성

- 저장 공간 80% 이상 사용하면 replication 을 멈춤
- split brain 을 고려하여 3ea 를 권장,
  하지만 동일 네트워크(VPC) 등에서 3ea 까지?
- snapshot 은 daily 로 저장
- alias 관리를 잘 하면 장애 대처시 매우 유연함

cashslide Mobile Media

캐시슬라이드는 전세계 최초로 잠금화면 미디어 영역을 개척한 서비스입니다

잠금화면에서
최적화된노출&클릭형태

좌우슬라이딩

와이드한화면/상하슬라이딩

1.5억
일간 노출량

잠금화면에서컨텐츠와광고를
이미지/영상소재로 커뮤니케이션

"전면화면에서컨텐츠도보고"

이미지

영상

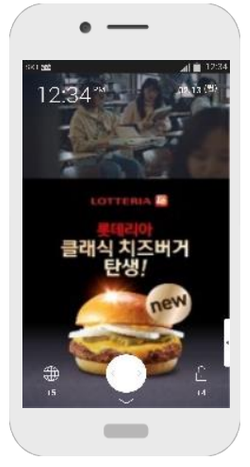
이미지

영상

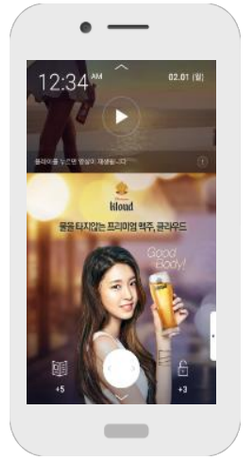
"전면화면에서광고도보고"

이미지

영상



이미지+영상



이미지+영상

elasticsearch 문제 상황

- snapshot restore 시 디스크 80% 이상인 경우 restore 안됨
- JVM 의 old Generation 이 차오르는 것은 반드시 모니터링 필요
- 디스크의 85% 이상인 경우 shard 재 분배가 이뤄지지 않음
- snapshot 에 있는 index 들을 삭제 해도 부하가 있거나 장애가 있으면 raw 데이터가 삭제 되지 않는 경우가 있음
- 시간 단위로 index 생성시 max file open 주의
- refresh_interval 의 의미 주의
- 같은 box type 의 경우 버전이 반드시 같아야 함 : v2.2.0 과 v2.2.1 이 shard rebalance 가 일어나지 않고 있었음

반드시 적용할 것 들

- elasticsearch port 방화벽 적용하여 접속 차단하기
- JVM GC memory 상황 확인하기
  - jstat –gcutil `ps –ef | grep java | egrep –v grep | grep elasticsearch | awk –F" " '{print $2}'` 5000
- bootstrap.mlockall : 시작시 메모리 할당 부터 하고 시작하기
- action.destructive_requires_name 모든 인덱스 삭제 방지
- 어떤 index 가 restore 중인지 확인하는 명령어
  - curl –s 'http://localhost:9200/_cluster/state' | jq '.restore'

logstash 문제 상황

처음 시작시

- 최소 1번 완전히 돌때까지는 sincedb 가 업데이트 안 됨 : 4.x.x 기준으로 경험
- 의외로 XXX 일 이전에 대한 문제 경험을 많이 겪음
- stdout 에 meta debug 까지 찍는 것을 추천

운영중 발생한 상황

- output 이 2개일 때 1개가 멈추면 다른 1개도 멈춤
- json 파싱에 실패하면 멈춰 버림 : skip 옵션?
- 설정 파일이 공유되어 logstash 를 2개 띄워야 할 때 좀 난감 : supervisor 사용
- logstash 의 성능을 높여야 할 때 : 병렬처리가 어려움

THANK YOU

NLBT

주소 : 서울시 서초구 남부순환로 347길 101 조이빌딩
전화번호 : 1522-1859