

Tech planet 2016

# Elasticsearch를 이용한 Log기반 모니터링 & Log 검색 시스템 구축

SK플래닛 윤태형

Tech  
planet  
2016

## Contents

I. Log기반 모니터링 & Log검색 시스템 소개

II. 전체 아키텍처

III. Elasticsearch 트러블슈팅

# LOG기반 모니터링 & LOG검색 시스템 소개

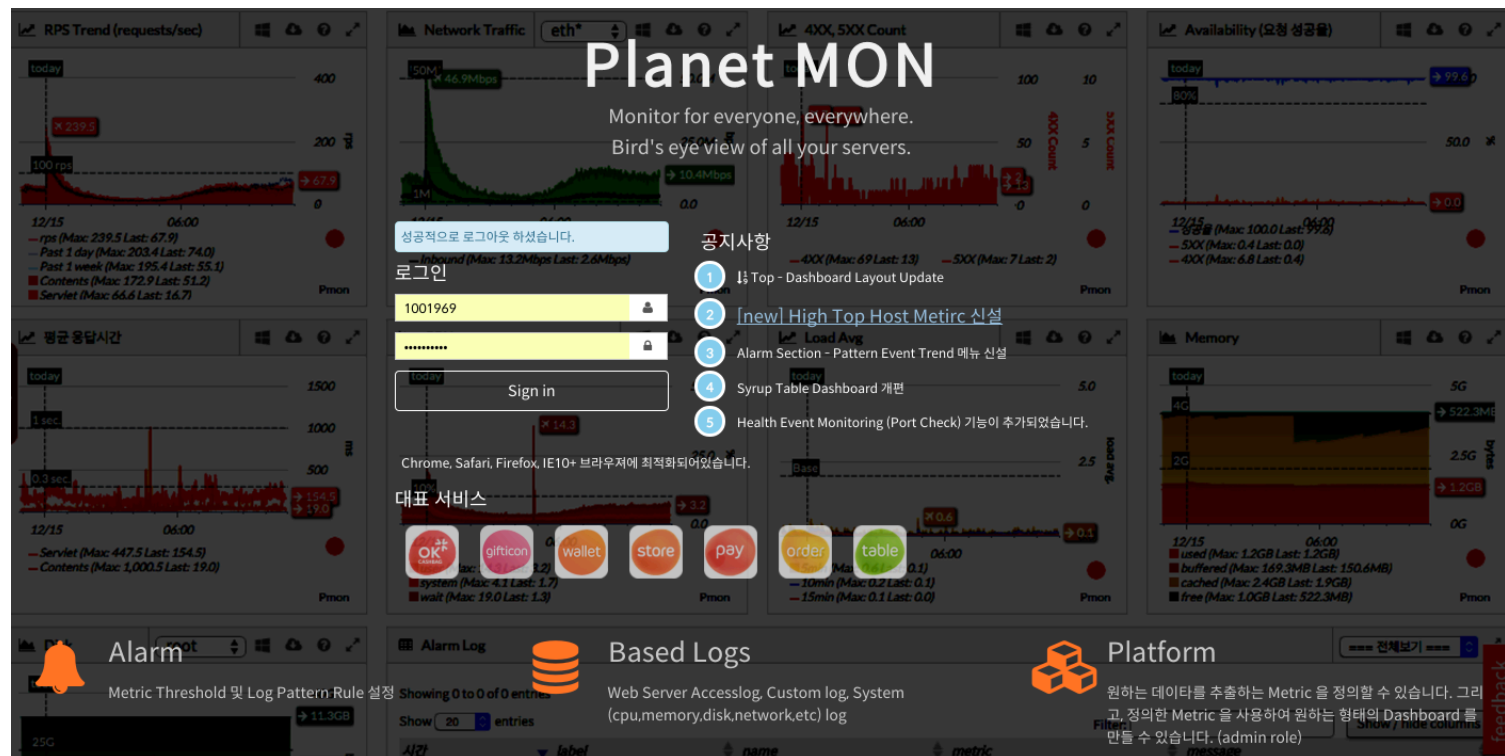
- PMON

- LMON

# PMON 개요

## Log 기반 모니터링 시스템

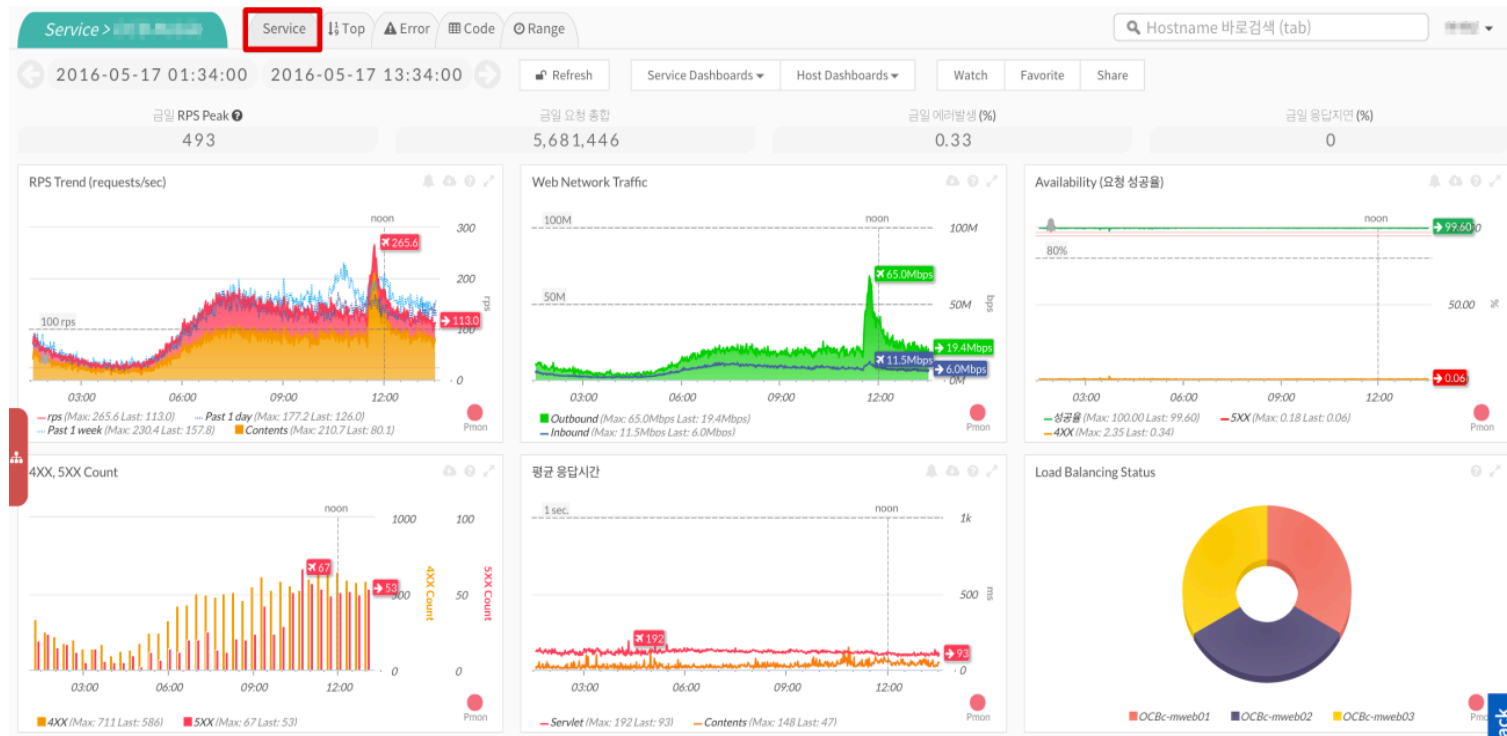
- Log를 수집, 분석하여 차트 & 대시보드 및 알람 제공
- 서비스 관리 기능을 제공하여 여러 서비스들을 통합 모니터링
- 반응형 웹 UI를 제공하여 다양한 디바이스 환경에서 접근



# PMON 주요 기능

## Log 모니터링

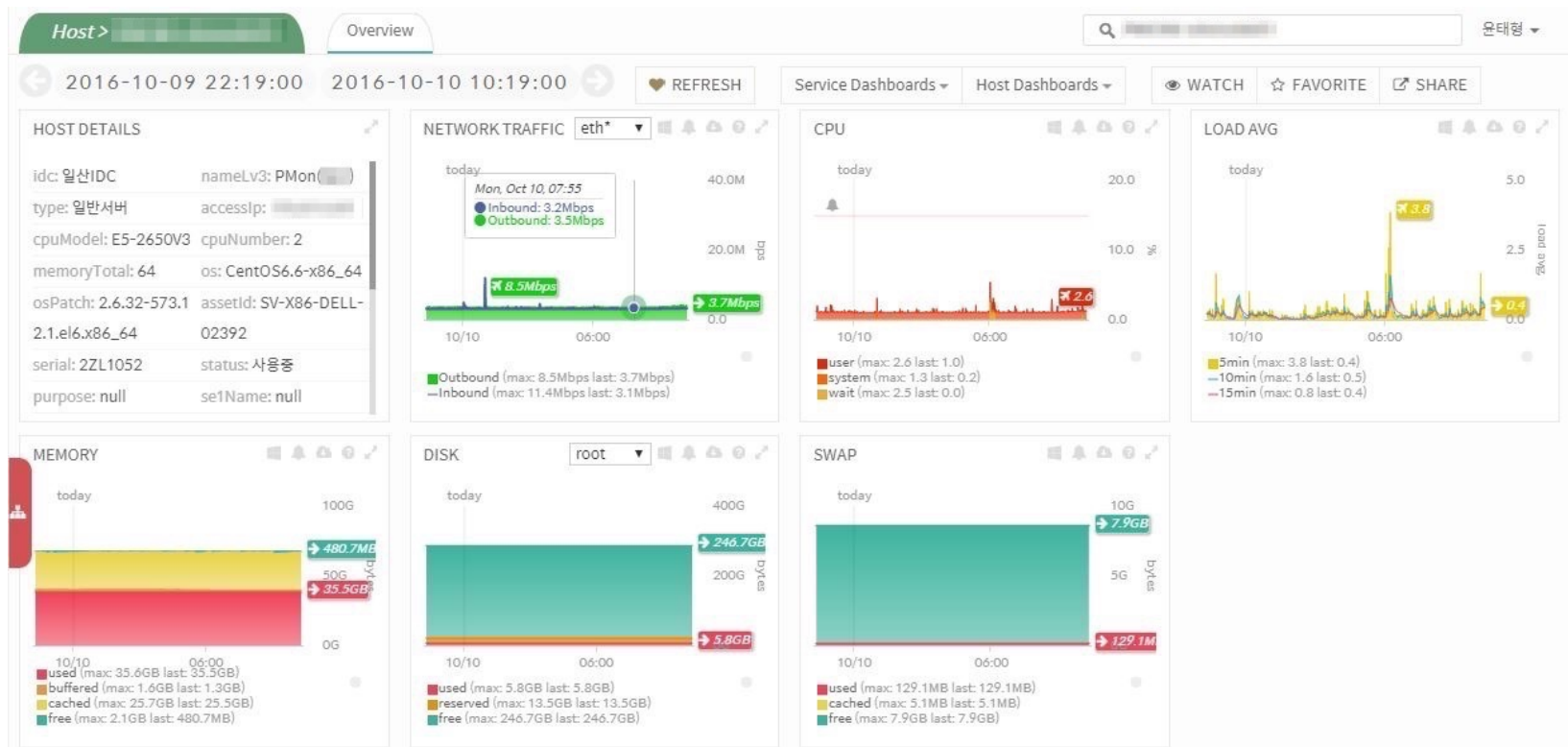
- Web Access Log 분석
  - Web 지표 차트, URL 분석 차트
- Log 패턴 분석
  - Log 패턴 발생 Count 차트



# PMON 주요 기능

## 시스템 모니터링

- CPU, Memory, Disk, Network, TCP Connection, ...
- Process/Port Check





# PMON 주요 기능

## 알람

- 차트 임계치 알람
- Log 패턴 알람
- 전송 채널
  - SMS
  - Email
  - Slack

WEB NETWORK TRAFFIC

### SYSTEM EVENT REGISTER

Name (\*)  
RPS Trend (requests/sec) (OCBc-portal)

Service (\*)  
OCB Web

Metric (\*)  
RPS Contents

Threshold (\*)  
>= 10 GAUGE (기본) 1분 연속

Direct  
Select users

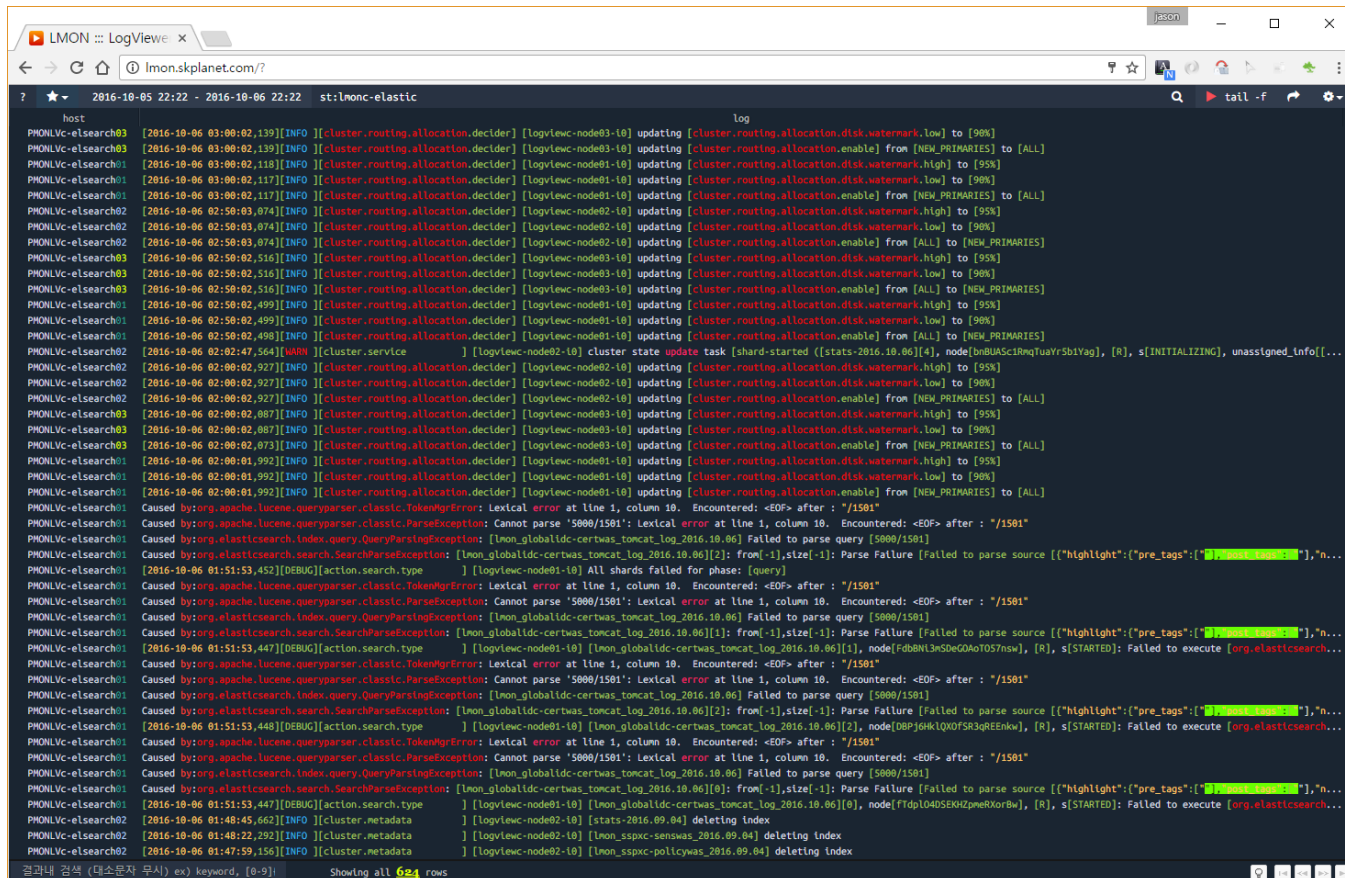
LDAP 에 등록되어 있는 직원이면, 바로 검색이 가능.  
혹은 이름:전화번호 형식으로 바로입력 가능 e.g. 홍길동;000-0000-0000

Group  
Select options

# LMON 개요

## Log 검색 & Tailing 시스템

- 실시간으로 여러 서버 Log 동시 검색 & Tailing
- Hot !!! - Log 검색 시간 대폭 단축



The screenshot shows the LMON LogView application interface. The top bar displays the application name 'LMON :: LogView' and the URL 'lmon.skplanet.com/'. The main area is divided into two panes. The left pane shows a list of log entries with columns for host, timestamp, and log message. The right pane shows a detailed view of a selected log entry, including the full log message and a stack trace. The log message is: '[2016-10-06 01:51:53.447][DEBUG][action.search.type] [logview-node01-10] All shards failed for phase: [query]'. The stack trace shows an exception: 'org.apache.lucene.queryparser.classic.ParseException: Lexical error at line 1, column 10. Encountered: <EOF> after: "/1501"'. The bottom status bar shows 'Showing all 684 rows'.



# LMON 주요 기능

---

## 여러 서버 Log 동시 검색

- Keyword 검색
- And/Or/Not 검색
- Like 검색
- Regular Expression 검색

## 여러 서버 Log 동시 Tailing

Tech planet 2016

# 전체 아키텍처

# Reference Architecture

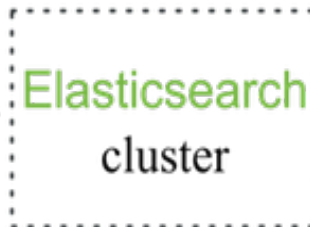
---

## ELK (Elasticsearch - Logstash - Kibana)

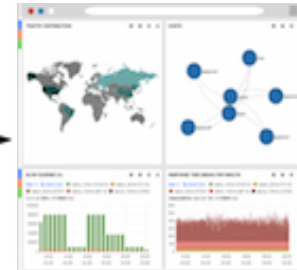
**Collection &  
Transport**



logstash

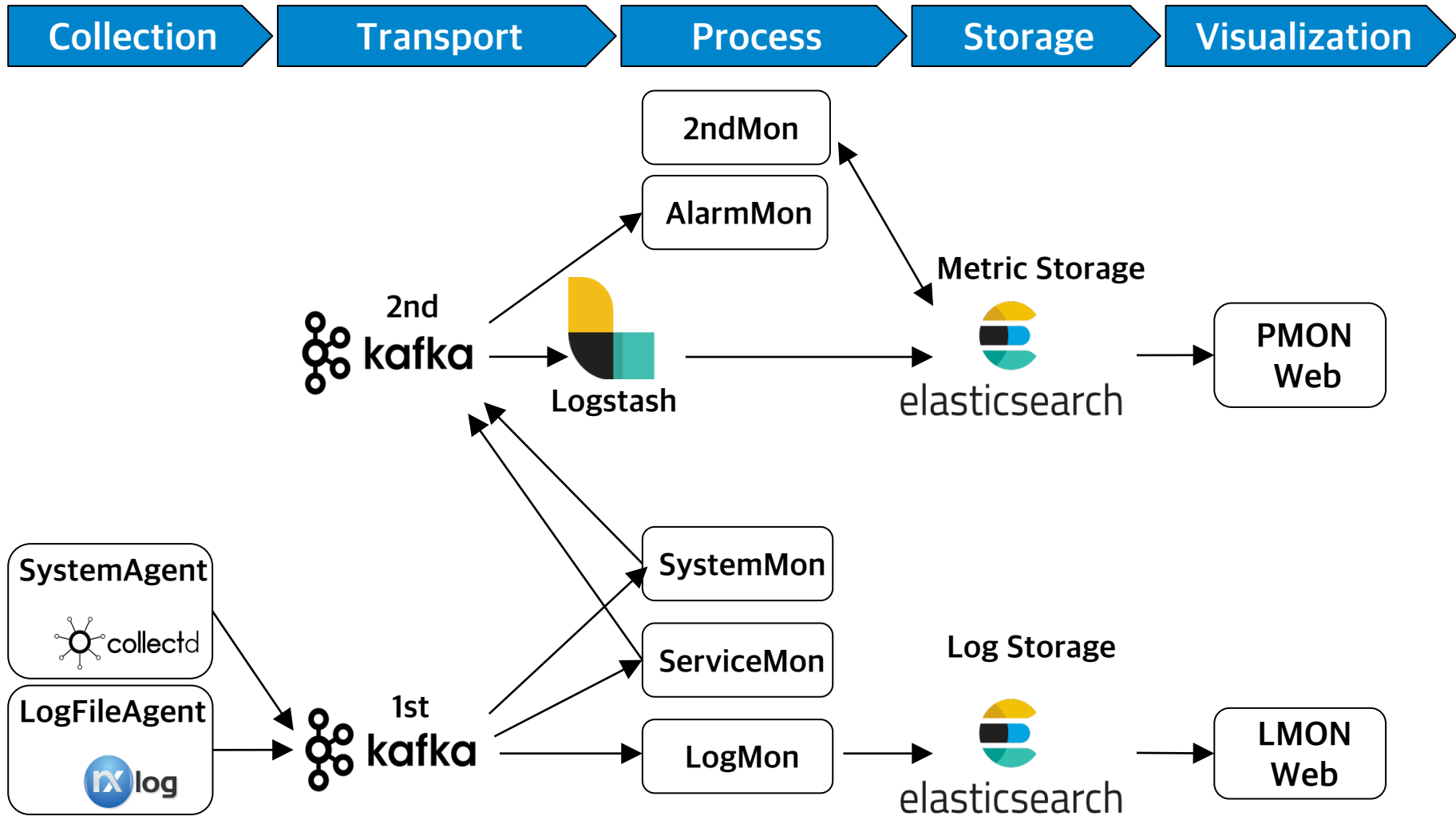


**Visualization**



Kibana

# 전체 아키텍처



# 전체 아키텍처

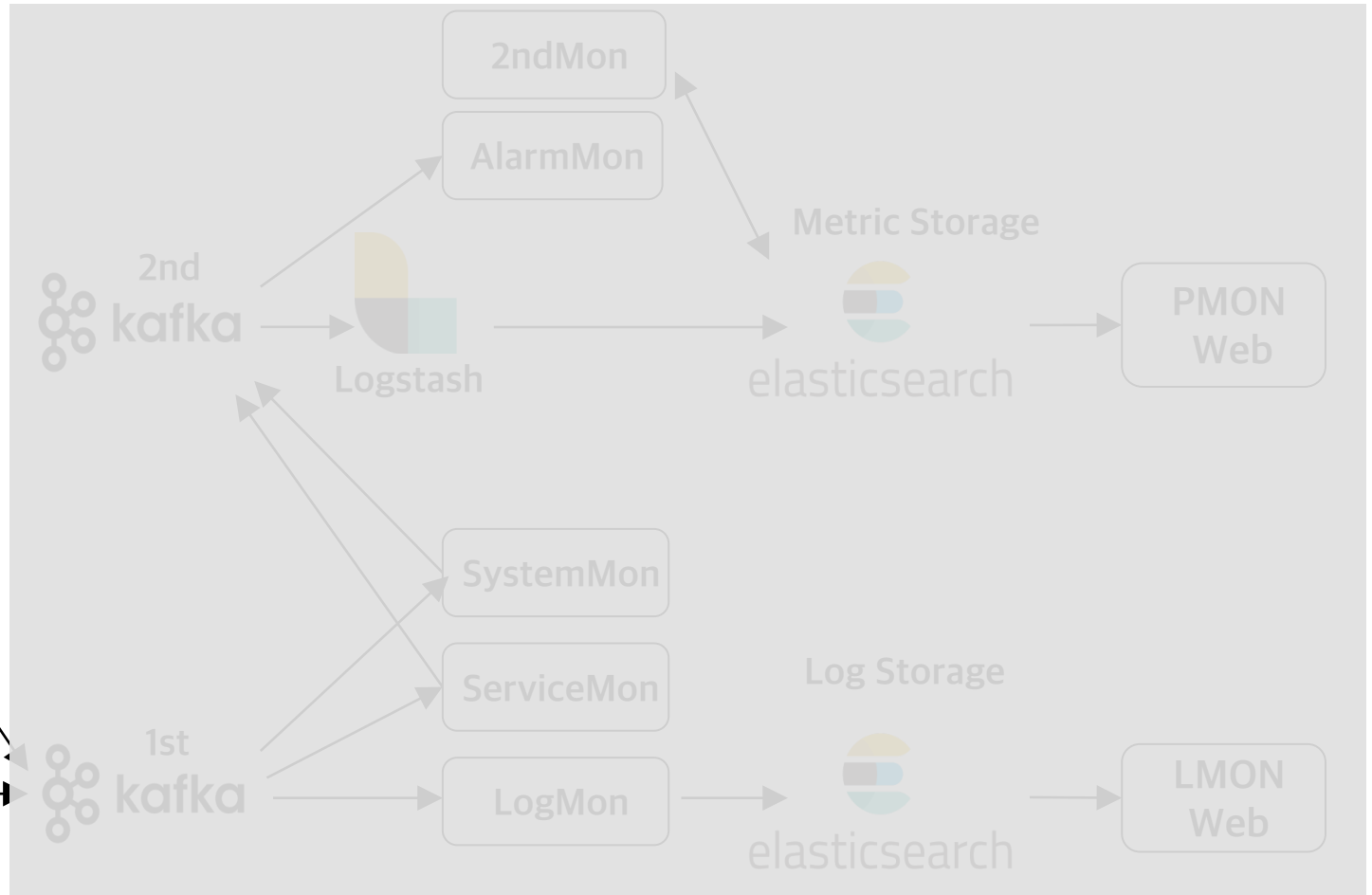
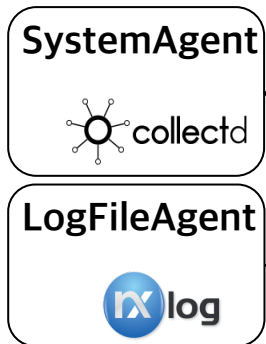
Collection

Transport

Process

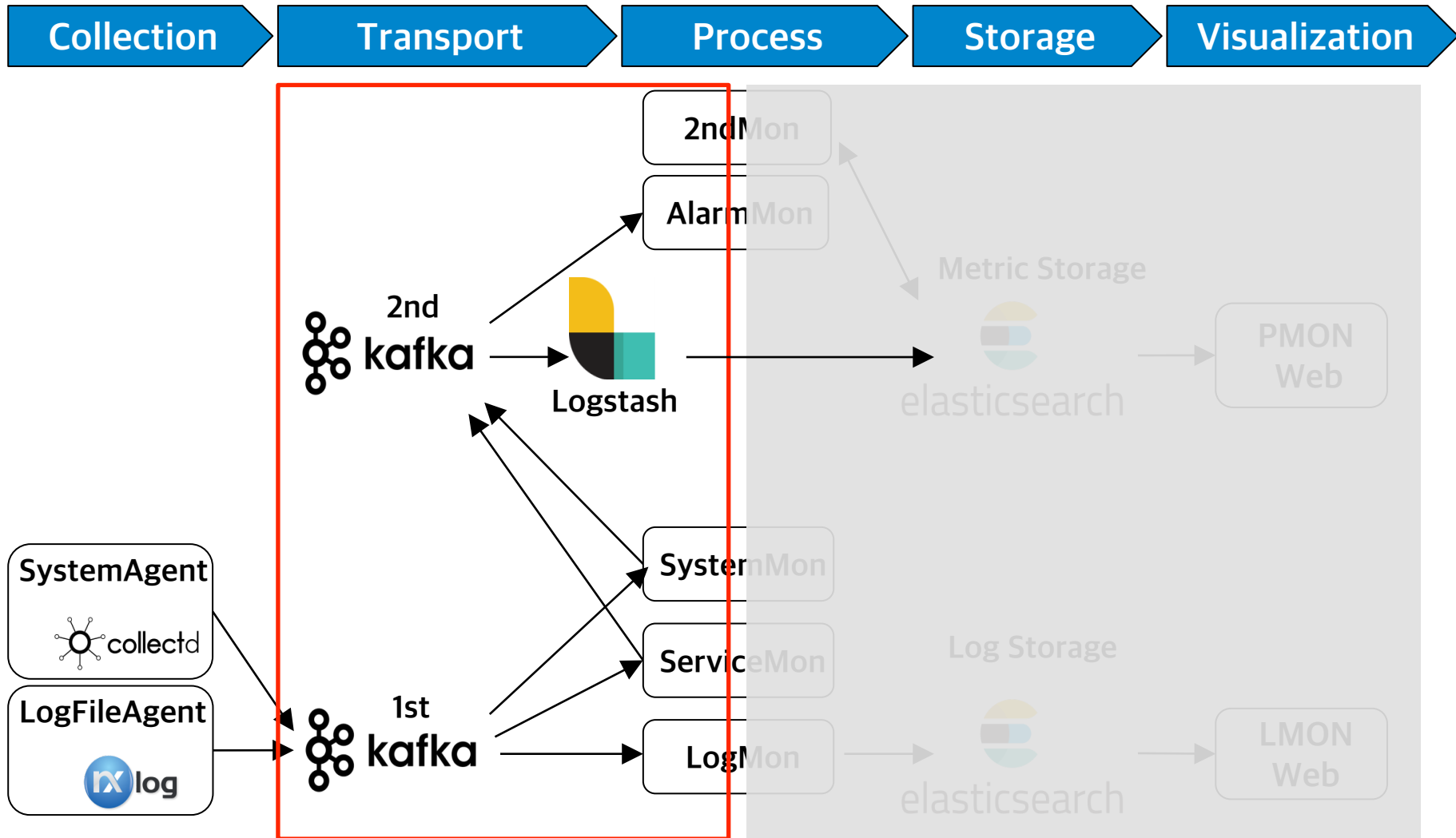
Storage

Visualization



수천대의 서버에서 System Metric과 Log File 데이터를 수집하는 Agent 관리

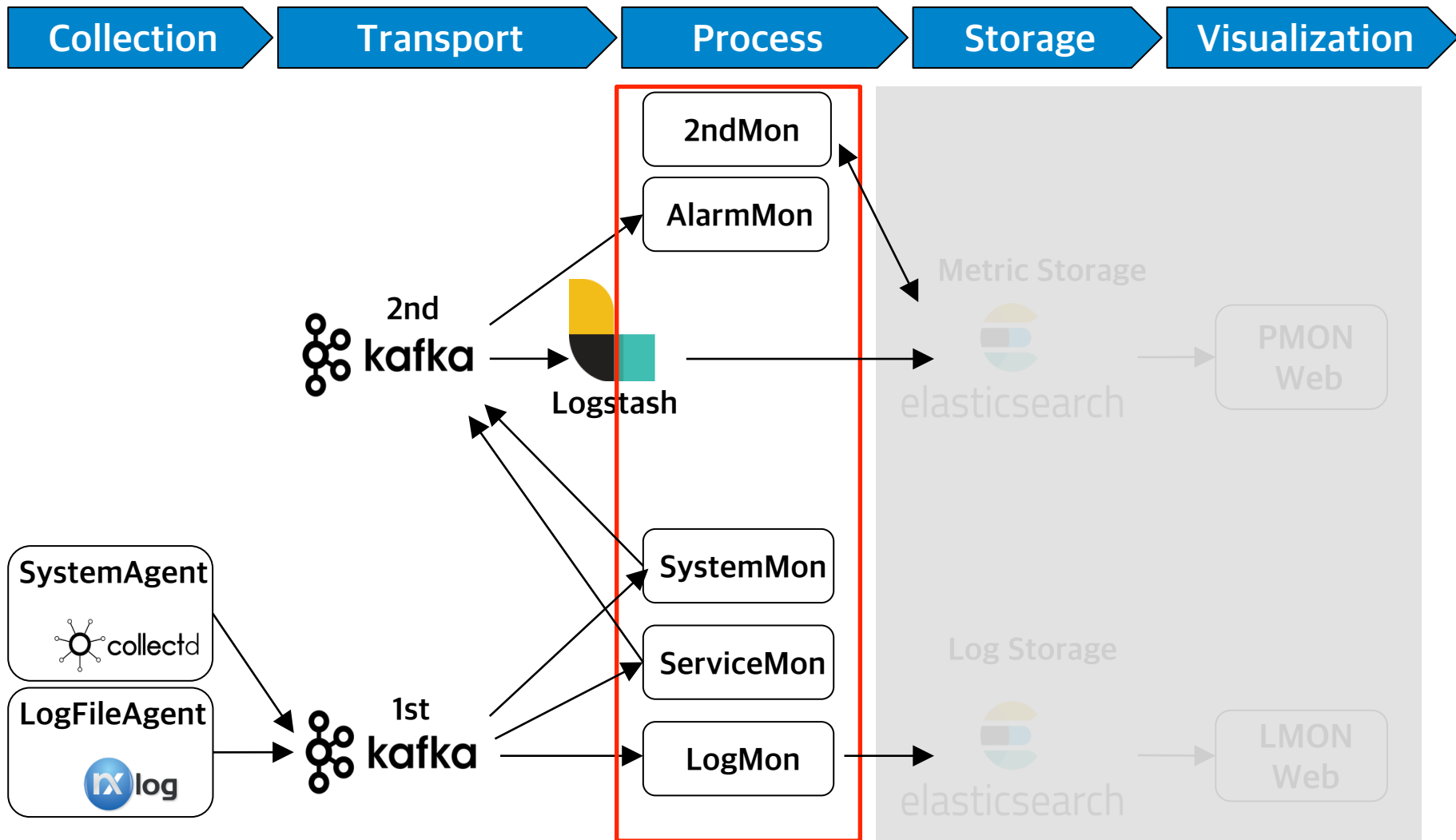
# 전체 아키텍처



Kafka와 Logstash를 활용한 안정적인 데이터 전송 및 보관



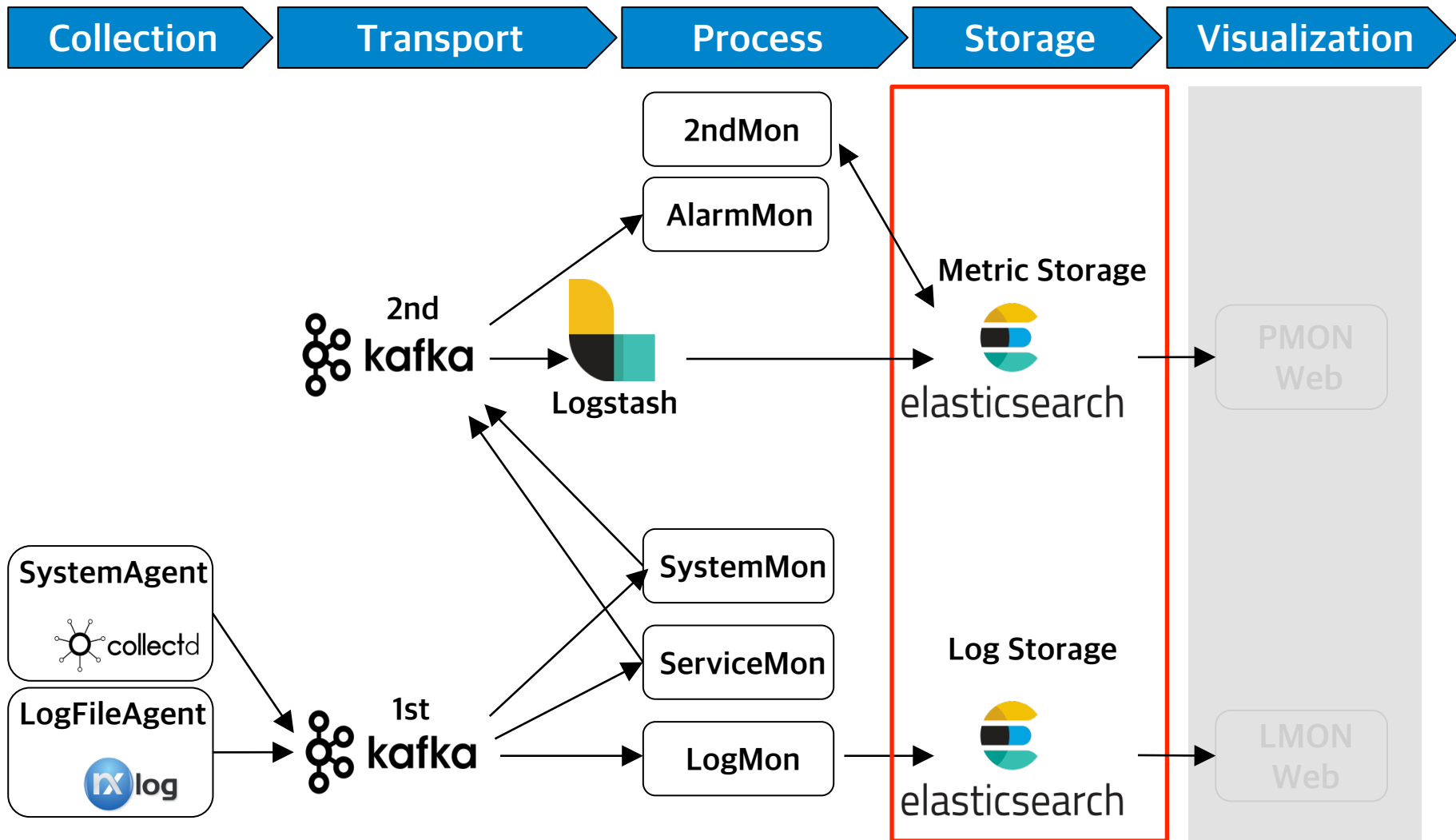
# 전체 아키텍처



적정한 기술을 사용한 Log Stream Analytics 서버 개발

- <http://readme.skplanet.com/?p=13110>

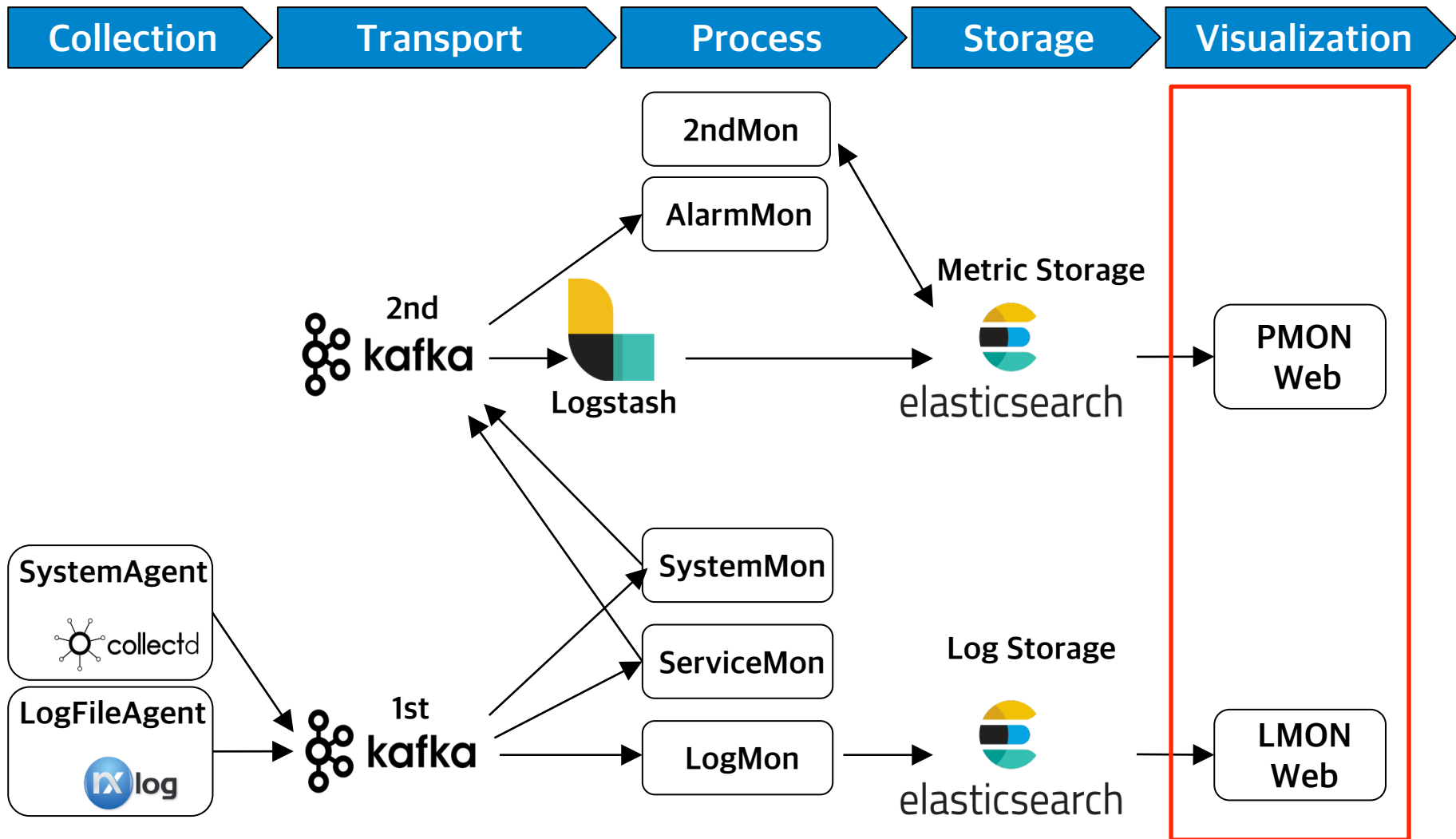
# 전체 아키텍처



Elasticsearch를 Timeseries DB와 같은 Metric Data 저장소로 사용

Elasticsearch를 검색 가능한 Log 데이터 저장소로 사용

# 전체 아키텍처



Metric Data기반으로 화려하고 다양한 그래프와 Dashboard를 동적으로 생성  
각 서버의 Log를 통합하여 거의 실시간으로 검색 & Tailing

# SK플래닛 Log 처리 현황

---

## 사용 서비스



...

## 수집 서버 & 파일 수

- System 데이터 수집 서버 - 7,000여대
- Log 수집 서버 - 1,600여대
- Log 수집 파일 수 - 7,000여개

## 1일 Log 수집량

- 30~50 억 Lines (3~5 Billion Lines)
- 2~3TB / 일

Tech planet 2016

# Elasticsearch 트러블슈팅

# PMON & LMON Elasticsearch Cluster

---

## Elasticsearch Cluster

- Version : 1.4.5 → 1.7.5
- Time-based Index
  - (ex) host-2016.10.17, host-hourly-2016.10.17, host-daily-2016.10

## Metric Storage



- 매일 Index 10여개 생성 (3개월 보관)
- Shard = 5 ~ 10, Replica = 1

## Log Storage



- 매일 Index 약 200여개 생성 (1개월 보관)
- Shard = 5 ~ 30, Replica = 1



# 문제와 위기

---

## Log Size가 증가함에 따라

- 많은 문제들
  - 잦은 Elasticsearch Cluster 장애
  - 낮에도 새벽에도 전화 연락
- 여러 차례 위기들
  - Elasticsearch 운영이 너무 힘든데...
  - 계속 사용할 수 있을까 ?

현재는 ?

# 트러블슈팅 리스트

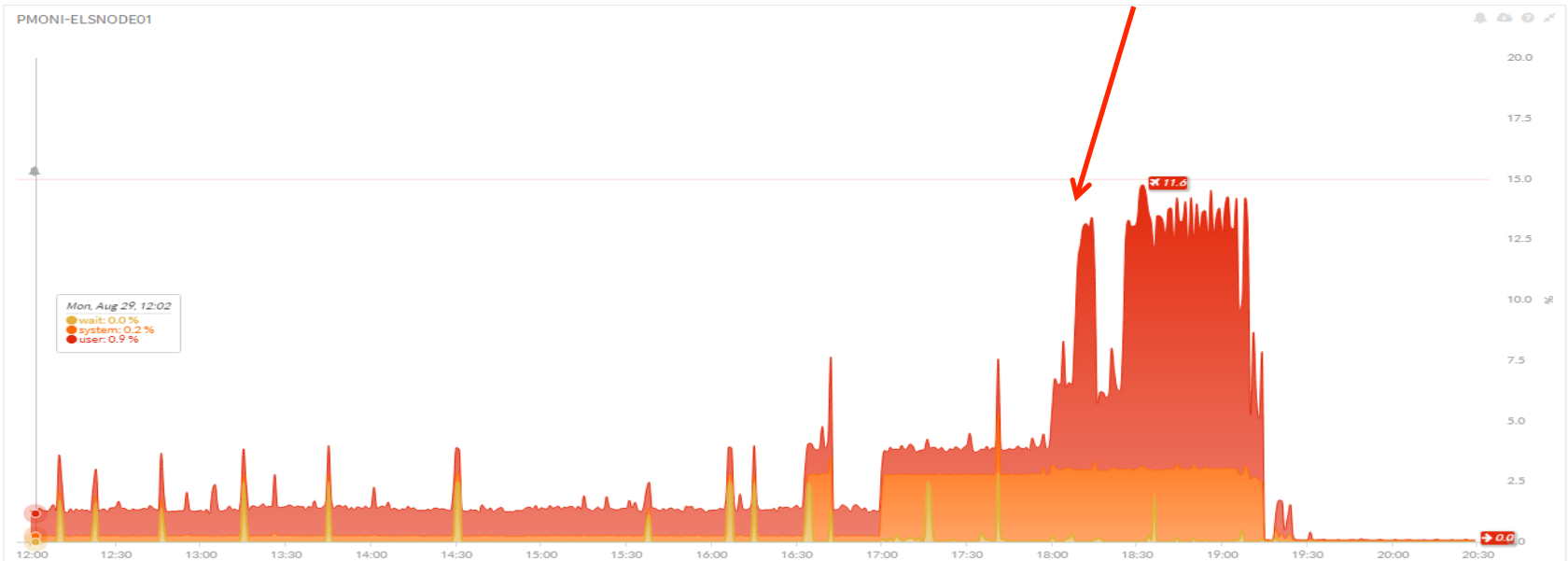
---

1. Heap Memory Shortage
2. Heavy Indexing
3. 첫번째 신데렐라 현상
4. 두번째 신데렐라 현상
5. Over 32G Heap Memory

# 1. Heap Memory Shortage (1)

## 증상

- 특정 Node의 Heap Memory 사용율 95% 이상
- GC (Garbage Collection) 계속 발생. 하지만 ...
- Elasticsearch Query Timeout. UI 반응 늦음.
- 정상화 잘 안되어 문제 지속. 강제 Kill & Restart.



# 1. Heap Memory Shortage (2)

---

## 해결 1 - Heap Memory 증가

- export ES\_MAX\_MEM=16g      충분하겠지?
- export ES\_MAX\_MEM=24g      좀 많이 쓰는데!
- export ES\_MAX\_MEM=31g      너무 많네...
- export ES\_MAX\_MEM=32668m    31.9G

### Elasticsearch Recommendation

- 50% Memory → Heap Memory
- Other 50% → Free (OS disk cache → Used by Lucene)
- Don't cross 32G (Compressed OOPS)

# 1. Heap Memory Shortage (3)

---

## 해결 2 - Heap Memory 사용 최소화

- Big Problem - Field Data Cache (1.X)
  - Sorting/Aggregation Query시 사용되는 Cache. Query 조건과 상관 없이 Index내 필요 Field의 모든 Value들을 Field Data Cache로 로딩
  - Default : Unbounded, Not evicted
- Field Data Cache 과다 사용 해결
  - Field Data Cache 사용 제한 : Size 설정 - 1G, Evicted
  - Field Data Cache 사용 최소화 : Field에 Doc Value 설정

### Doc Value - Disk 기반 Data Structure

- 성능 저하 10~25%
- Field Data Cache 미사용으로 Heap Memory 사용량을 현저히 줄임

# 참고 - Doc Value 설정

## Elasticsearch 1.X

- Field별 Field Data Cache 사용량

```
$ curl 'http://localhost:9200/_cat/fielddata?v&fields=*
```

id	host	ip	node	total	@timestamp	user	...
iExRFXn1Qw	esnode01	192.168.1.2	esnode01	128mb	24.8kb	124mb	...

- Index 생성 시 Field Mapping에서 설정

```
"mappings":{  
  "cpu":{  
    "properties":{  
      "user":{"type":"float", "doc_values":true},  
      "system":{"type":"float", "doc_values":true},
```

## Elasticsearch 2.X

- Doc Value Default



# 1. Heap Memory Shortage (4)

---

## 해결 3 - Heap Memory 사용 최소화

- Query 기간 제한
- 장기간 데이터 Query가 필요한 경우
  - Hourly, Daily Metric 데이터 생성

## 2. Heavy Indexing - LMON (1)

---

### 증상

- 사내 사용자 대상으로 Query로 인한 CPU 사용율은 상대적으로 낮음
- 대부분의 CPU 사용율은 Indexing (Inserting) 때문에 발생
- 처리 데이터 증가로 CPU 사용율 상당폭 증가

### 원인

- 데이터 Indexing시 2회 Indexing함
  - each field
  - \_all field : Special catch-all field

## 2. Heavy Indexing - LMON (2)

---

### 해결

- Index 생성시 Field Mapping에서 \_all disable 설정

```
"mappings":{  
  "log":{  
    "_all": {  
      enabled: false  
    },  
    "properties":{  
      ...  
    }  
  }  
}
```

- 결과적으로 CPU 사용율 기존 대비 약 60~70%로 감소

### 3. 첫번째 신데렐라 현상 (1)

---

#### 증상

- LMON에서 Index는 최초 데이터 입력 시 자동 생성
- 자정에 Index 100여개 생성
- 자정 이후 약 30분간 Indexing 에러가 상당히 많이 발생

### 3. 첫번째 신데렐라 현상 (2)

---

#### 원인

- 1개 Index 생성 시간 : 5~10초 (전체 Shard 갯수에 따라 다름)
- Index 생성은 상당한 비용이 소요되는 작업

#### 해결

- 다음날 Index를 사전에 생성해 두기 위해 Index Create Script 배치 수행

## 4. 두번째 신데렐라 현상 (1)

---

### 증상

- Cluster에 신규 Node 추가 후 자정에 신규 Node에서 Heap Memory 부족 발생
- Index를 생성하면 신규 Node로 Shard가 집중 배치



## 4. 두번째 신데렐라 현상 (2)

---

### 원인

- Elasticsearch Shard 분배 정책
  - Node당 전체 Shard 개수 비슷하게 유지
  - 신규 추가 Node는 Shard 개수가 작으므로 Index를 생성하면 신규 추가 Node로 Shard가 집중 배치됨
  - 자정 이후 신규 추가 Node로 Indexing 부하가 집중됨

### 해결

- Index 생성시 Node당 Shard 개수 제한

```
$ curl -XPUT 'http://localhost:9200/host-2016.10.17' -d '{  
  "settings" : {  
    "number_of_shards" : 5,  
    "number_of_replicas" : 1,  
    "index.routing.allocation.total_shards_per_node" : 2  
  }  
}'
```

## 5. Over 32G Heap Memory (1)

---

### 증상

- 신데렐라 현상으로 인한 31G Heap Memory 부족
- Node들의 CPU 사용율은 비교적 낮음

### 고민

*Oh My God !*



*어쩌라고 ?*

### 해결 - 잘못된 길

- 48G Heap Memory (LMON은 심지어 64G)

## 5. Over 32G Heap Memory (2)

---

### 증상

- Long GC (Garbage Collection)
  - 1분 30초 ~ 2분 (Stop The World)
- Node가 Cluster에서 Remove됨
- 수일 간격으로 1~2회 발생

### 다시 고민

- CMS GC → G1 GC 로 변경 ?
- 32G 미만-Heap Memory 부족, 32G 이상-Long GC
- 아~ 좋은 해결 방법이 없을까 ?

## 5. Over 32G Heap Memory (3)

---

### 해결

1. Heap Memory 31G 로 낮춤
2. Elasticsearch Upgrade 1.4.5 → 1.7.5
  - 보다 효율적인 Memory 사용
3. Master, Data Node 분리
  - Cluster가 아주 안정화됨
  - 전체 Node의 CPU 사용율 감소
4. 1개 서버 내 2개 Elasticsearch Node 기동
  - `cluster.routing.allocation.same_shard.host: true`

# 참고 - Configuration

---

## Elasticsearch

- elasticsearch.yml 설정

```
bootstrap.mlockall: true
indices.fielddata.cache.size: 1gb
cluster.routing.allocation.same_shard.host: true
```

## Linux

- /etc/security/limits.conf 설정

```
elastic-dev soft memlock unlimited      # max mlockall size
elastic-dev hard memlock unlimited      # max mlockall size
elastic-dev soft nofile 655360          # max fd count
elastic-dev hard nofile 655360          # max fd count
```

# 참고 - Configuration

---

## fd (file descriptor) count 확인

```
$ curl 'localhost:9200/_cat/nodes?v&h=host,fdc,fdm'  
host      fdc  fdm  
es-node01 57329 655360  
es-node02 57380 655360  
es-node03 58760 655360
```

# Summary - Elasticsearch 운영 가이드

---

1. fd count & bootstrap.mlockall 체크
2. Time-based Index 사용 → Index 사전 생성
3. Memory 64G 이상 → 31G Heap Memory 사용
4. 처리 데이터 증가
  - Master, Data, Query Node 분리
  - Field Data Cache Size 설정
  - Field Data Cache 사용 최소화 (Doc Value)
  - Query 기간 제한
5. Scale-out
  - 서버 내 2개 Elasticsearch Node 기동
  - 서버 추가
6. Elasticsearch 2.X → Marvel 모니터링. 무료.

Tech planet 2016

**Thank you**

**Email : [th.yun@sk.com](mailto:th.yun@sk.com)**