

Druid Monitoring Web Application

Metatron Project

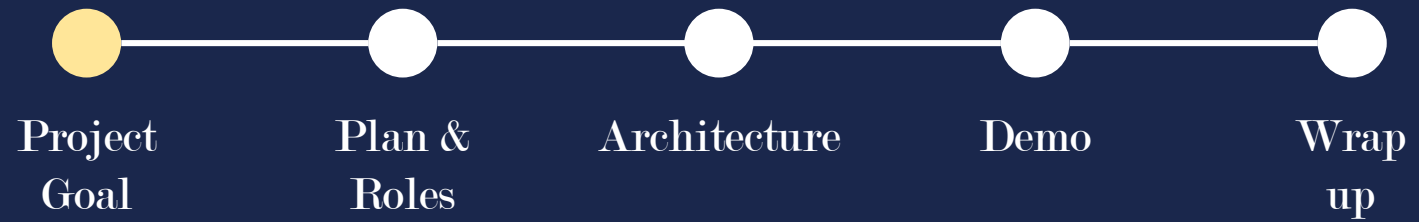
T WorX 1기

강준후, 김수연, 김진영, 오영택

Contents

1. Our Goal
2. Plan & Roles
3. Project Architecture
4. Demonstration
5. Wrap up

Our Goal



⇒ Druid Monitoring web application

(1) Necessity: Inefficient monitoring on distributed platform

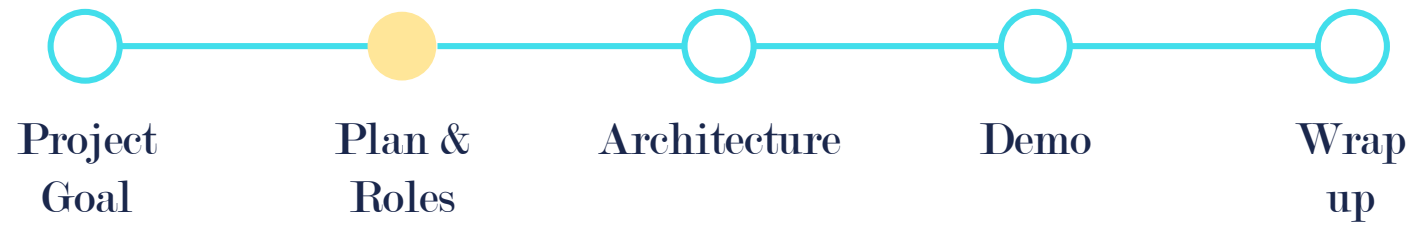
(2) Expectation

- ✓ Efficiency Increase
- ✓ User Friendly

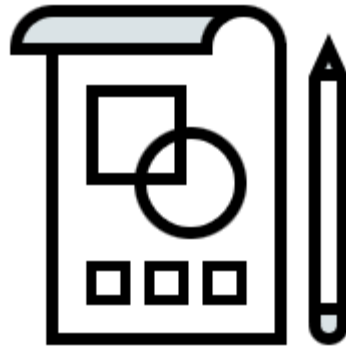
Why Druid Monitoring?

- * **Druid:** Scalable distributed data store
 - ✓ **Difficulties on:** diagnosing each server & nodes
 - ✓ **Lack:** Druid does not have cluster monitoring feature
- ∴ **Need:** Monitoring tool for Druid

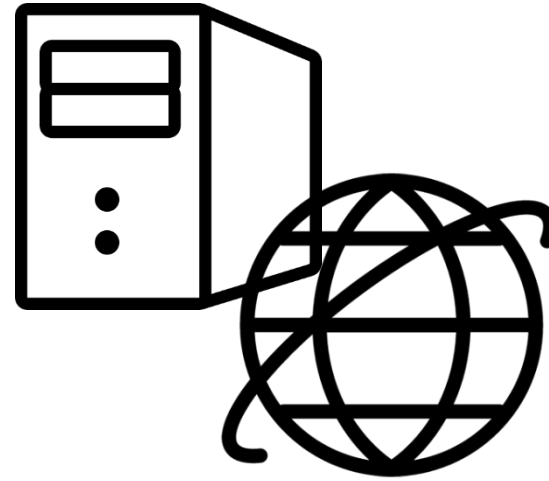
Project Milestone



Druid & Metatron
Analysis



Planning



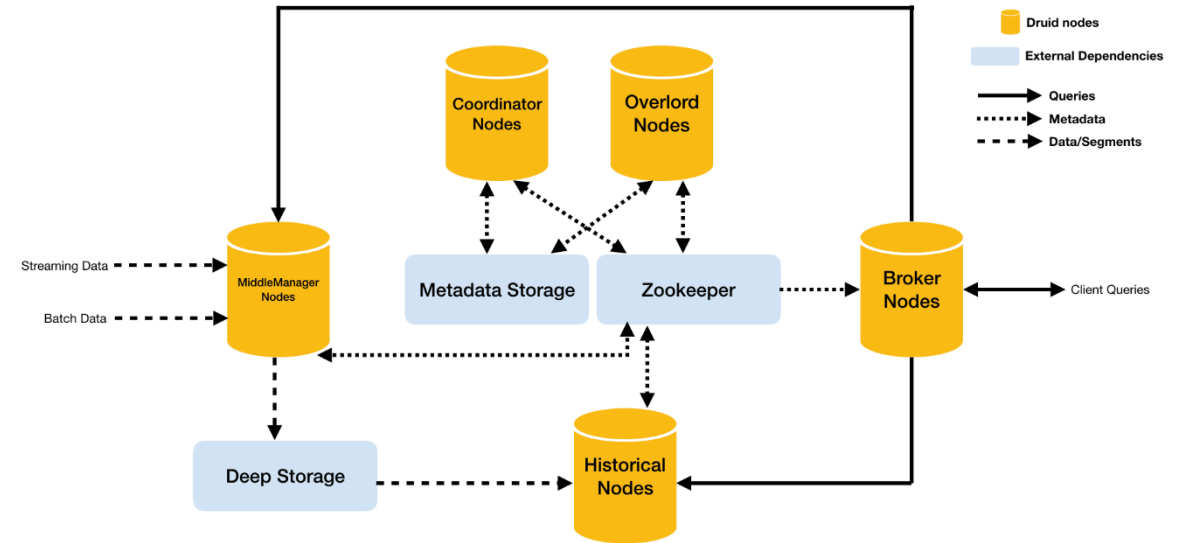
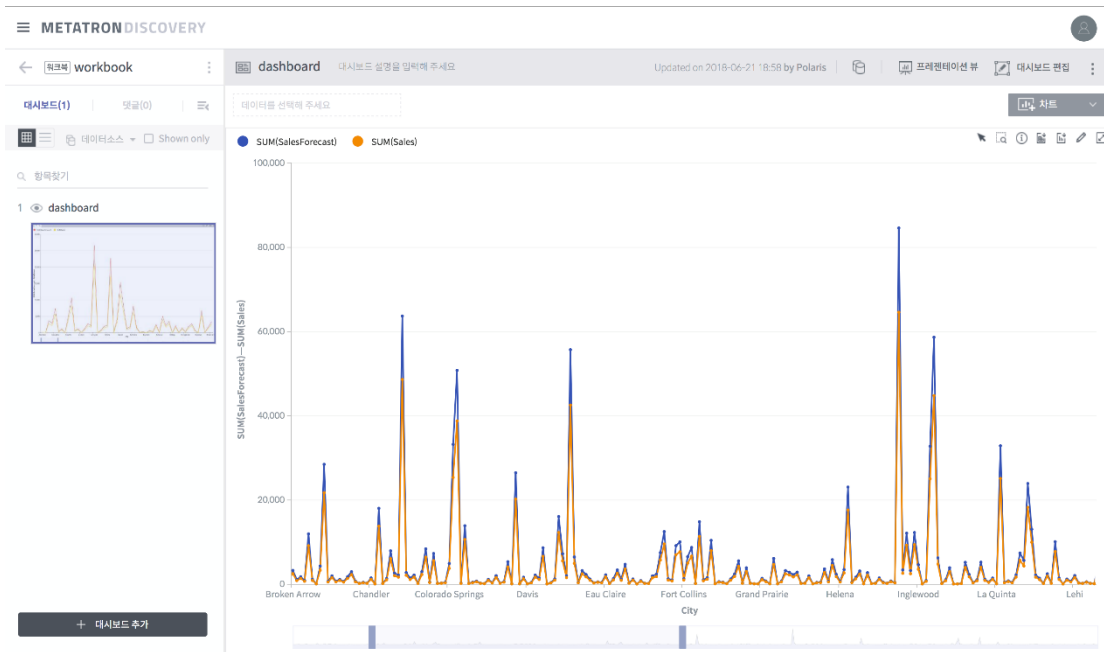
Development



Quality Assurance

Druid & Metatron Analysis

Plan & Roles



Competitors Analysis

Plan &
Roles

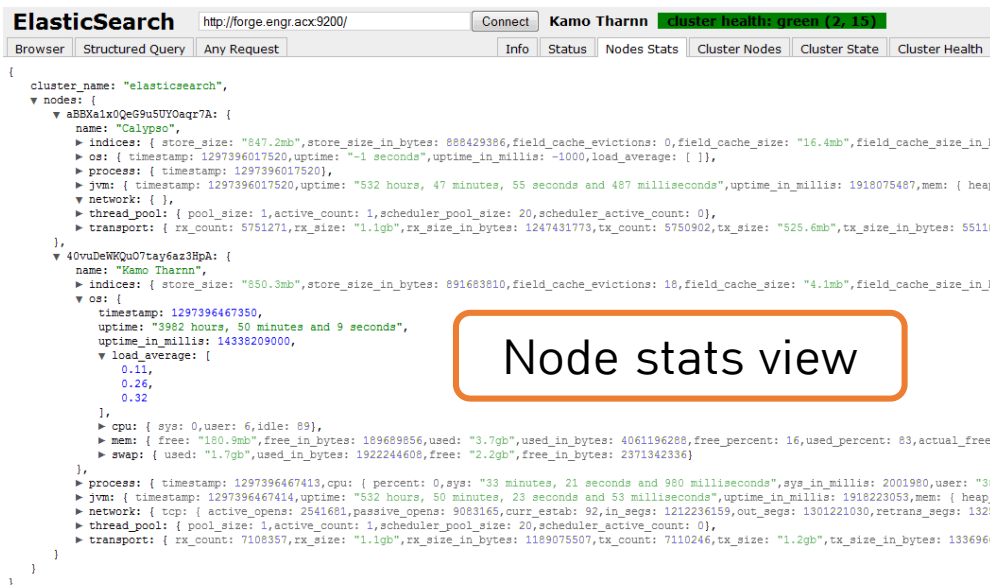
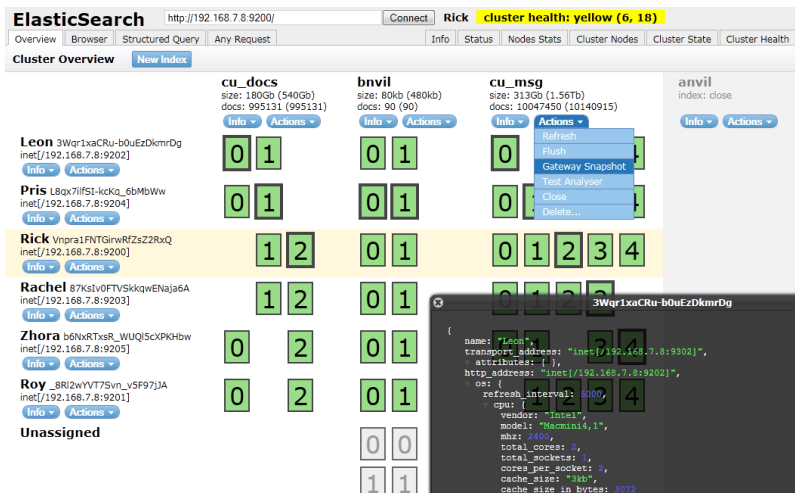
1. Elasticsearch-head, Elasticsearch-kopf (deprecated)
2. Cerebro
3. Elastic X-pack Monitoring



ES-head:

a web front end for an Elasticsearch cluster

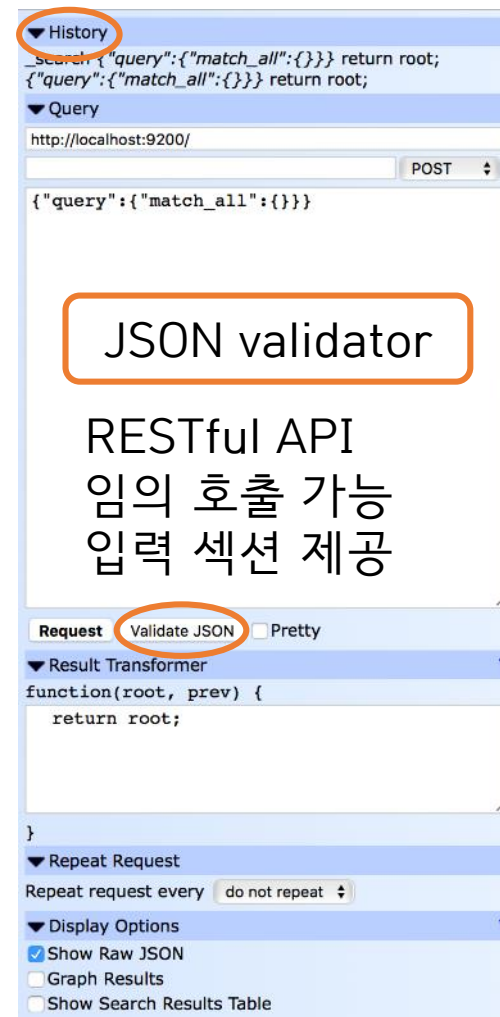
Cluster Overview



Node stats view

Index와 node 관련 작업 가능

- 클러스터 조회 두 가지 검색 interface: JSON or 표 형식
- 클러스터의 상태 보여줌



JSON validator

RESTful API
임의 호출 가능
입력 섹션 제공

*kopf = 'head' in German

ES-kopf*

web administration tool for Elasticsearch

The screenshot displays the ES-kopf web interface. At the top, a navigation bar includes 'KOPF' (circled in orange), 'cluster', 'rest', 'aliases', 'analysis', 'percolator', 'warmers', and 'snapshot'. Below this, a summary bar shows '6 nodes', '5 indices', '80 shards', '0 unassigned', '4155 docs', and '814.93KB'. The main area is divided into two panels. The left panel, labeled 'Nodes' (in a blue box), lists nodes like 'Anti-Lay', 'Boneyard', 'Contemplator', 'Lady Jacqueline Falsworth-Crichton', 'Matt Murdock', and 'Ogress', each with a status bar and a 'show settings' dropdown menu. The right panel, labeled 'Indices' (in a blue box), shows a table of indices with columns for index name, shards, docs, and size. A callout box labeled 'Refresh time interval' points to the 'refresh index' option in the dropdown menu. Another callout box labeled 'Shard와 node별 분배 상태 오버뷰' (Overview of shard and node distribution status) points to the main index table.

Refresh time interval

Indices

Nodes

1. 새 index 생성
2. Shard 분배 비활성화
3. 클러스터 설정
4. 클러스터 진단 옵션

Shard와 node별 분배 상태 오버뷰

- Elasticsearch의 클러스터 상태 시각적 확인
- Indices에 대한 상세 정보 접근 가능

KOPF	cluster	rest	aliases	analysis	percolator	warmup	repository
KOPF	cluster	rest	aliases	analysis	percolator	warmup	repository
KOPF	cluster	rest	aliases	analysis	percolator	warmup	repository
KOPF	cluster	rest	aliases	analysis	percolator	warmup	repository

Cluster Health in 4 colors

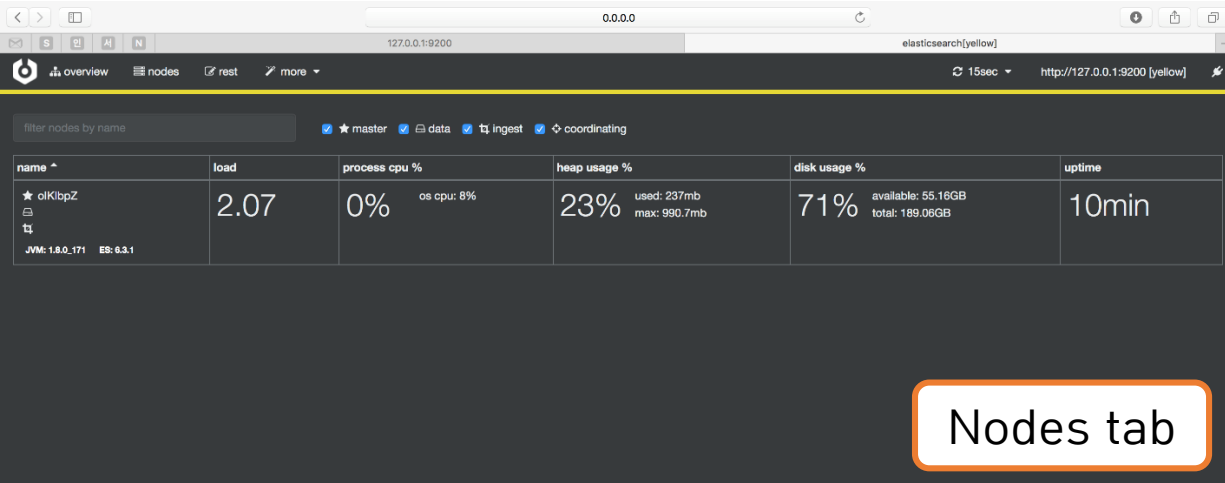
Cerebro: upgrade version of kopf

Overview

Refresh 기능

Display shard stats

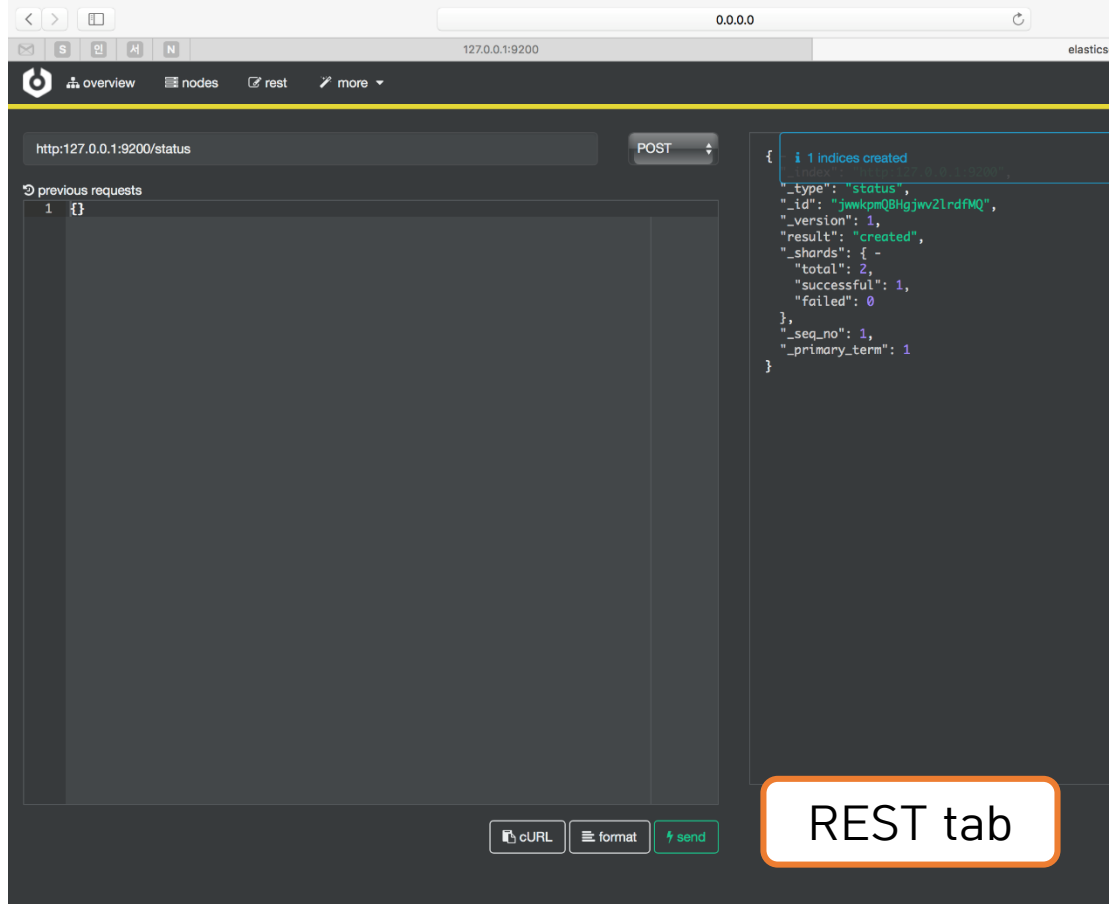
Cerebro



The screenshot shows the 'Nodes' tab in the Cerebro interface. At the top, there's a filter bar with 'filter nodes by name' and checkboxes for roles: master, data, ingest, and coordinating. Below this is a table with columns: name, load, process cpu %, heap usage %, disk usage %, and uptime. The table contains one entry for a node named 'oiklpZ'. A red box highlights the 'Nodes tab' label at the bottom right.

name ^	load	process cpu %	heap usage %	disk usage %	uptime
★ oiklpZ JVM: 1.8.0_171 ES: 6.3.1	2.07	0% os cpu: 8%	23% used: 237mb max: 990.7mb	71% available: 55.16GB total: 189.06GB	10min

Nodes tab



The screenshot shows the 'REST' tab in the Cerebro interface. It features a REST client with a URL bar containing 'http:127.0.0.1:9200/status', a method dropdown set to 'POST', and a large text area for the request body. Below the text area are buttons for 'cURL', 'format', and 'send'. On the right, the JSON response is displayed. A red box highlights the 'REST tab' label at the bottom right.

http:127.0.0.1:9200/status POST

previous requests

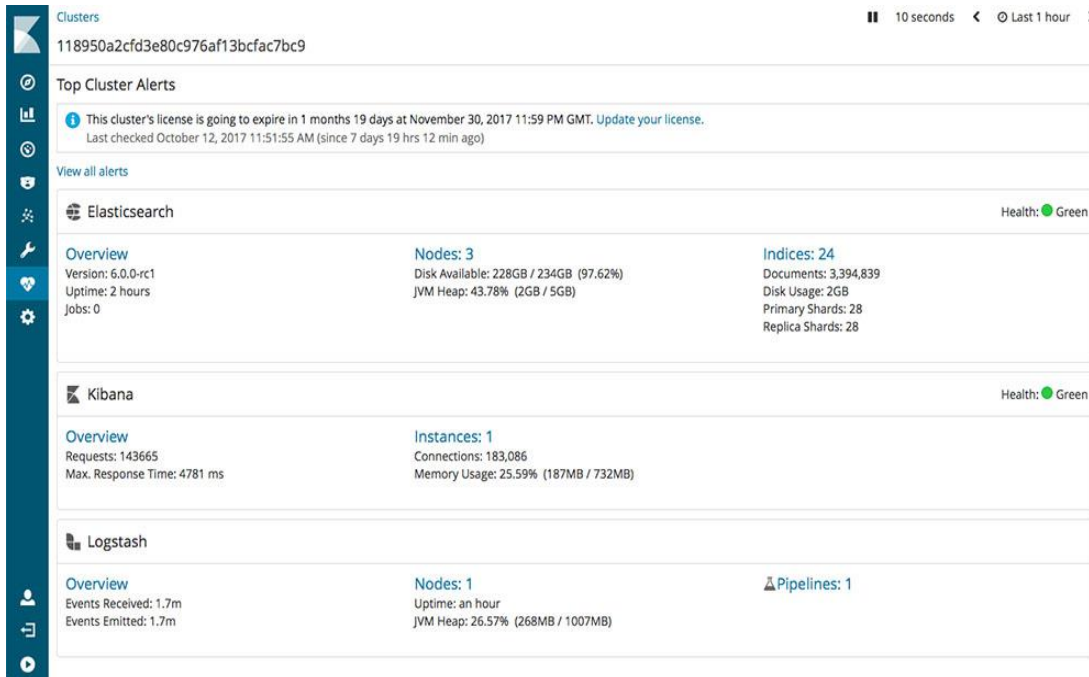
1 {}

```
{
  "type": "status",
  "_id": "jwkkpmQBHgJwv2lrdFMQ",
  "_version": 1,
  "result": "created",
  "_shards": {
    "total": 2,
    "successful": 1,
    "failed": 0
  },
  "_seq_no": 1,
  "_primary_term": 1
}
```

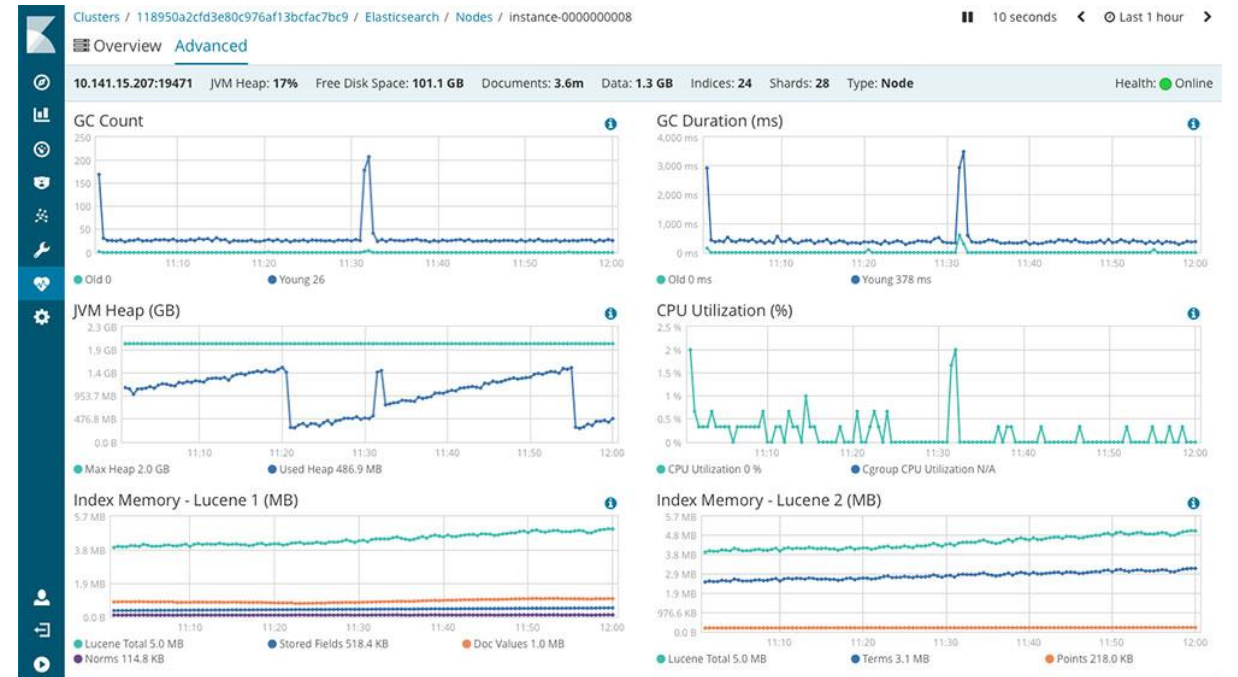
REST tab

Elastic X-pack: Monitoring

Clusters



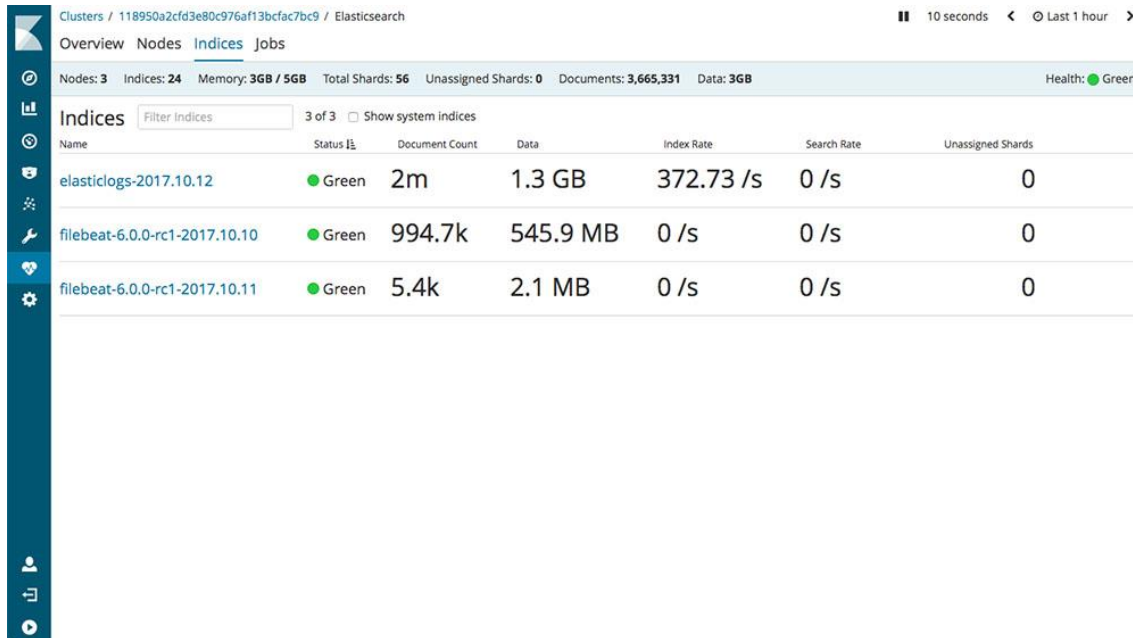
Node



- Elastic Stack의 실행 상태, 성능 지속적 확인
- 클러스터 상태 한 눈에 보기
- Alert: 클러스터 상태, 라이선스 만료, metrics의 변화에 대한 알림

Elastic X-pack: Monitoring

Indices



Logstash*



*Logstash: 데이터 ingestion, 변환 및 보관 기능

기획서

Plan & Roles



Druid Client Console
for

기능

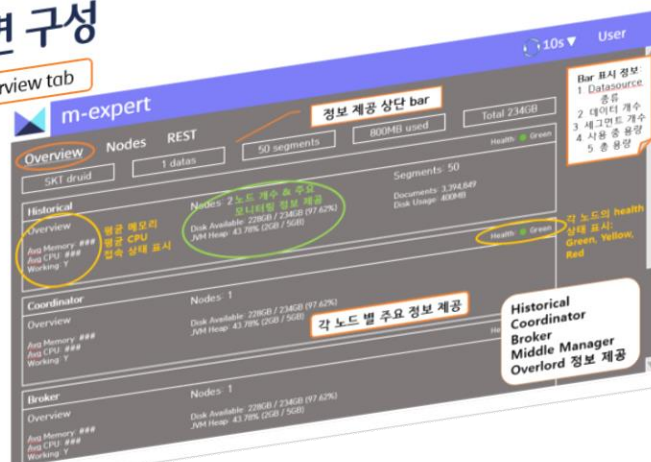
Node Status Monitoring
(1) Node Status Monitoring

- Overview : 전체 시스템 상
- Node Status diagram : 노드
- 상세 Node Status : 개별 노드

1.목표 2.제품 분석 3.기능 4.화면 구성

화면 구성

1.Overview tab



기능 명세서

Plan & Roles

분류	항목	세부 항목	기간	완료 일자	상태	설명
대분류	소분류	세부 항목	Release 1	18.07.20	100%	대분류 소분류 세부 항목
모니터링	General	새로고침	Release 1			노드 상태 갱신 주기를 드롭다운 형식으로 제공한다. (5, 10, 15, 30초, 1분.)
		클러스터 상태 표시	Release 1			우측 상단에 지속적으로 클러스터 상태 표시.
	노드별 모니터링	Overview tab 시각화	Release 1			노드 그룹별 전체적인 상태 표시.
		Nodes tab 시각화	Release 1			노드 종류별로 묶어서 개별 노드의 전체적인 상태를 표시.
		개별 노드 상태 상세정보 시각화	Release 1			Nodes tab에서 사용자가 선택한 노드의 상세정보 창으로 이동, 요약정보를 포함한 상세정보를 시각화.
	이력 관리	Druid 내 저장된 로그 그래프 형식 제공	Release 1			Druid에 저장된 로그 기록을 사용자에게 그래프 형식으로 보여준다. 이때 사용자가 조회하고 싶은 기간을 설정할 수 있다.
	상단 정보 bar	데이터소스 개수, segment 개수, 용량 used, 총 용량 정보 표시	Release 1			#(n) datas / # segments / # MB or GB used / Total # GB 의 정보를 표시해준다.
REST API	API Listing	API 색인 기능과 즐겨찾기 기능	Release 2			API 정보 검색 기능 탑재. 주로 쓰는 건 즐겨찾기로 등록 가능하도록.
		이력 관리 기능 (TBD)	Release 2			API 사용 이력을 기록하고 자주 사용하는 API를 보여준다.
		JSON Validate	Release 2			해당 JSON를 보냈을 때 나오는 결과값 출력.(202,404)
		노드별 API 목록 제공	Release 2			노드별 사용 가능한 API 목록을 리스트로 제공하여, 클릭 시 쿼리 발생을 돕는다.
		JSON sample 및 Description 제공	Release 2			쿼리 실행에 필요한 JSON 파일의 기본 양식 및 주석으로 사용자가 선택한 쿼리에 대한 Description 제공 .
Datasource	Segment Display		Release 3			Node 내 저장된 Segment 정보 조회

Monitoring 지표 선정

Plan & Roles

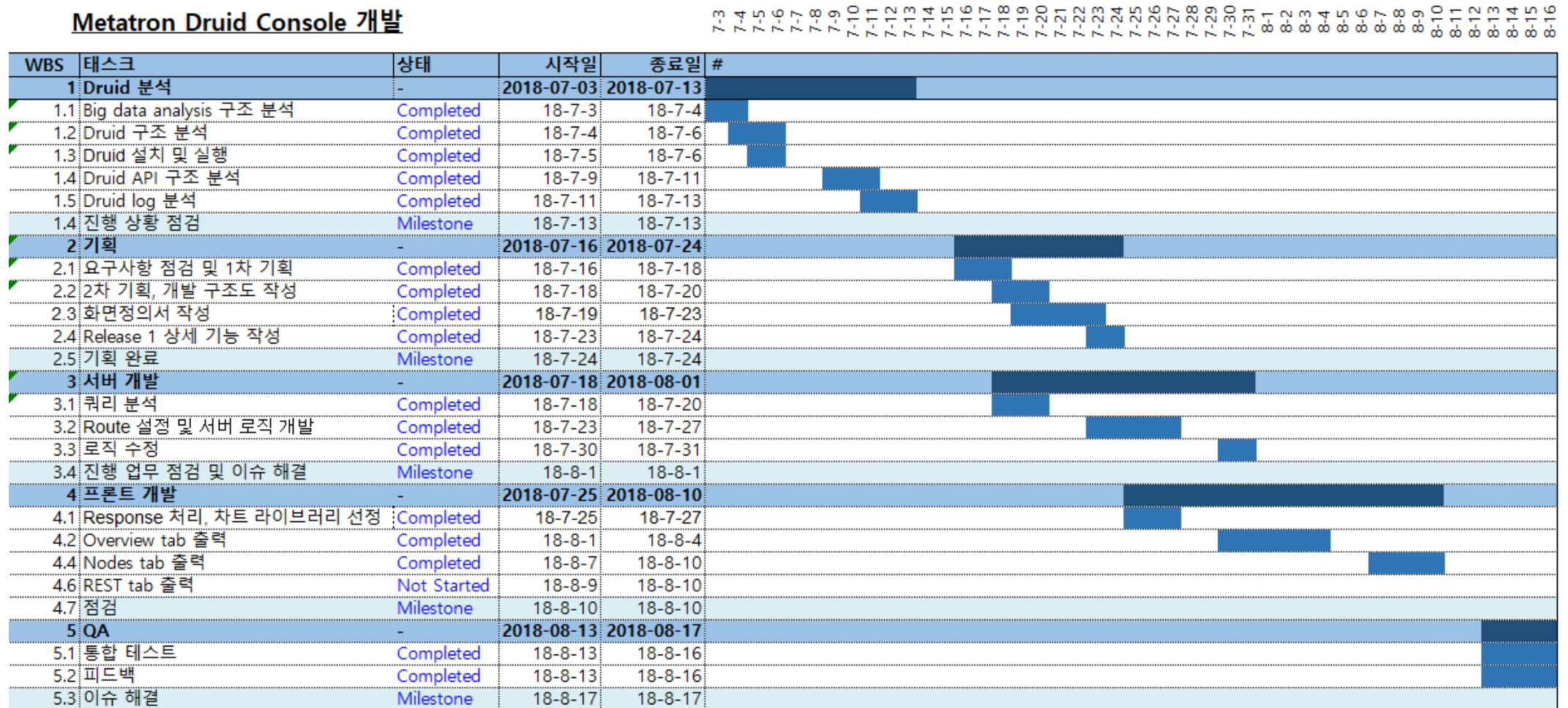
노드	metric	기타	overview 출력 여부	상세 정보 출력 여부	현재 사용 가능 여부	필요한 도움
Coordinator	placeholder		Y_공통	Y_공통		
	jvm/bufferpool/capacity		Y_공통	Y_공통		
	jvm/bufferpool/count		Y_공통	Y_공통		
	jvm/bufferpool/used		Y_공통	Y_공통		
	jvm/gc/mem/capacity		Y_공통	Y_공통		
	jvm/gc/mem/init		Y_공통	Y_공통		
	jvm/gc/mem/max		Y_공통	Y_공통		
	jvm/gc/mem/used		Y_공통	Y_공통		
	jvm/mem/committed		Y_공통	Y_공통		
	jvm/mem/init		Y_공통	Y_공통		
	jvm/mem/max		Y_공통	Y_공통		
	jvm/mem/used		Y_공통	Y_공통		
	jvm/pool/committed		Y_공통	Y_공통		
	jvm/pool/init		Y_공통	Y_공통		
Broker	정체 모름		?	Y_공통		
	jetty/numOpenConnections		Y_공통	Y_공통		
	jvm/bufferpool/capacity		Y_공통	Y_공통		
	jvm/bufferpool/count		Y_공통	Y_공통		
	jvm/bufferpool/used		Y_공통	Y_공통		
	jvm/gc/mem/capacity		Y_공통	Y_공통		
	jvm/gc/mem/init		Y_공통	Y_공통		
	jvm/gc/mem/max		Y_공통	Y_공통		
	jvm/gc/mem/used		Y_공통	Y_공통		
	jvm/mem/committed		Y_공통	Y_공통		
	jvm/mem/init		Y_공통	Y_공통		
	jvm/mem/max		Y_공통	Y_공통		
	jvm/mem/used		Y_공통	Y_공통		
	jvm/pool/committed		Y_공통	Y_공통		
	jvm/pool/init		Y_공통	Y_공통		
	jvm/pool/max		Y_공통	Y_공통		
	jvm/pool/used		Y_공통	Y_공통		
	query/cache/delta/errors	적 cache 관리 여부 파악	Y	Y		
	query/cache/delta/evictions		?	Y	N	QueryCountStatsMonitor 추가
	query/cache/delta/hitRate		Y	Y	N	QueryCountStatsMonitor 추가
	query/cache/delta/hits	보내는 시간	Y	Y	N	QueryCountStatsMonitor 추가
	query/cache/delta/misses	전체 완료 시간	Y	Y	N	QueryCountStatsMonitor 추가
	query/node/time		Y	Y		
	query/time	QueryCountStatsMonitor	Y	Y		
	query/success/count	QueryCountStatsMonitor				
	query/failed/count	QueryCountStatsMonitor				
	query/interrupted/count					

노드	metric	기타	overview 출력 여부	상세 정보 출력 여부	현재 사용 가능 여부	필요한 도움
Overlord	placeholder		Y_공통	Y_공통		
	jvm/bufferpool/capacity		Y_공통	Y_공통		
	jvm/bufferpool/count		Y_공통	Y_공통		
	jvm/bufferpool/used		Y_공통	Y_공통		
	jvm/gc/mem/capacity		Y_공통	Y_공통		
	jvm/gc/mem/init		Y_공통	Y_공통		
	jvm/gc/mem/max		Y_공통	Y_공통		
	jvm/gc/mem/used		Y_공통	Y_공통		
	jvm/mem/committed		Y_공통	Y_공통		
	jvm/mem/init		Y_공통	Y_공통		
	jvm/mem/max		Y_공통	Y_공통		
	jvm/mem/used		Y_공통	Y_공통		
	jvm/pool/committed		Y_공통	Y_공통		
	jvm/pool/init		Y_공통	Y_공통		
Middlemanager	jvm/pool/max		?	Y		
	jvm/pool/used		?	Y		
	segment/added/bytes	Indexing Service	Y	placeholder		
	segment/txn/success	Indexing Service	Y	Y		
	task/run/time	Indexing Service	Y	Y		
		p/p+th = 성공률?	Y	Y		
	ingest/events/processed		?	placeholder		
	ingest/events/throwAway		Y_공통	Y_공통		
	ingest/events/unparseable		Y_공통	Y_공통		
	ingest/handoff/count		Y_공통	Y_공통		
			Y_공통	Y_공통		
			Y_공통	Y_공통		
			Y_공통	Y_공통		
			Y_공통	Y_공통		
Historical	jvm/bufferpool/capacity		Y_공통	Y_공통		
	jvm/bufferpool/count		Y_공통	Y_공통		
	jvm/bufferpool/used		Y_공통	Y_공통		
	jvm/gc/mem/capacity		Y_공통	Y_공통		
	jvm/gc/mem/init		Y_공통	Y_공통		
	jvm/gc/mem/max		Y_공통	Y_공통		
	jvm/gc/mem/used		Y_공통	Y_공통		
	jvm/mem/committed		Y_공통	Y_공통		
	jvm/mem/init		Y_공통	Y_공통		
	jvm/mem/max		Y_공통	Y_공통		
	jvm/mem/used		Y_공통	Y_공통		
	jvm/pool/committed		Y_공통	Y_공통		
	jvm/pool/init		Y_공통	Y_공통		
	jvm/pool/max		Y	Y		
	jvm/pool/used		Y	Y		
	query/cache/delta/errors		Y	Y		
	query/cache/delta/evictions		?	?		
	query/cache/delta/hitRate		?	?		
	query/cache/delta/hits		Y	?		
	query/cache/delta/misses		Y	?		
	query/cache/delta/timeouts		Y	?		
	query/cache/delta/errors		Y	?		
	query/cache/total/hitRate		Y	?		
	query/cache/total/misses		Y	?		
	query/cache/total/timeouts		Y	?		
	query/cpu/time		Y	??	N	QueryCountStatsMonitor 추가
	query/segment/time		Y	Y	N	QueryCountStatsMonitor 추가
	query/segmentAndCache/time		Y	Y	N	QueryCountStatsMonitor 추가
	query/time		Y	Y	N	HistoricalMetricsMonitor 추가
	query/wait/time		Y	Y	N	HistoricalMetricsMonitor 추가
	query/success/count	QueryCountStatsMonitor	Y	Y	N	HistoricalMetricsMonitor 추가
	query/failed/count	QueryCountStatsMonitor	Y	Y	N	HistoricalMetricsMonitor 추가
	query/interrupted/count	QueryCountStatsMonitor	Y	Y	N	HistoricalMetricsMonitor 추가
	segment/max	HistoricalMetricsMonitor	Y	Y		
	segment/used	HistoricalMetricsMonitor	Y	Y		
	segment/usedPercent	HistoricalMetricsMonitor	Y	Y		
	segment/count	HistoricalMetricsMonitor	Y	Y		

WBS

Plan & Roles

Metatron Druid Console 개발



Role (Analysis, Planning)

Plan &
Roles



AND



metatron

Druid & metatron Analysis

All members

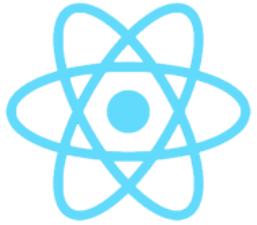


Competitors Analysis

김진영 김수연

Role (Development)

Plan &
Roles



React

Front

김진영 강준후



Publisher

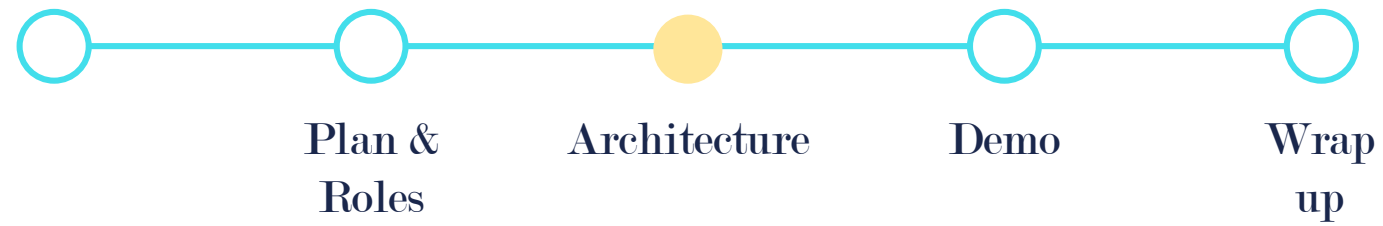
김수연

django

Server

오영택

Project Architecture



**Distributed
streaming platform**

Handles
log data stream
of druid

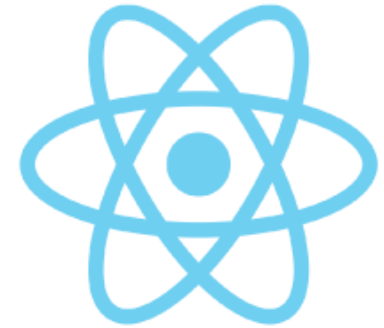


Target system



Backend

Queries data
to druid
& **Returns** it
to React



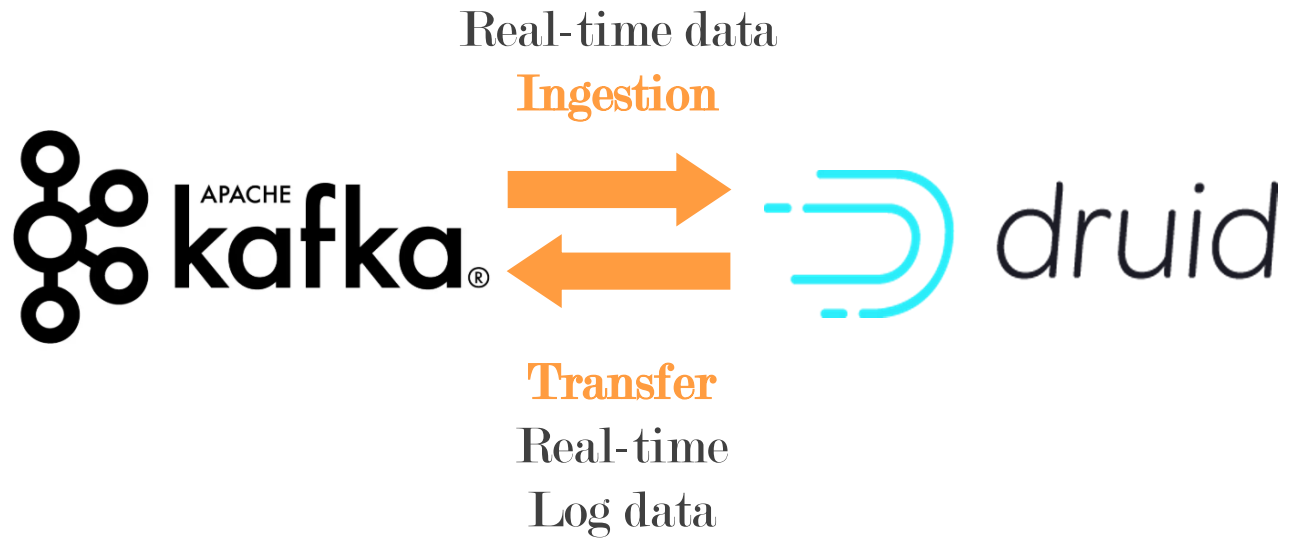
REACT

Frontend

Requests data
to Django
+ creates
User Views

Project Architecture: Kafka & Druid

Architecture



Project Architecture: Druid & Django



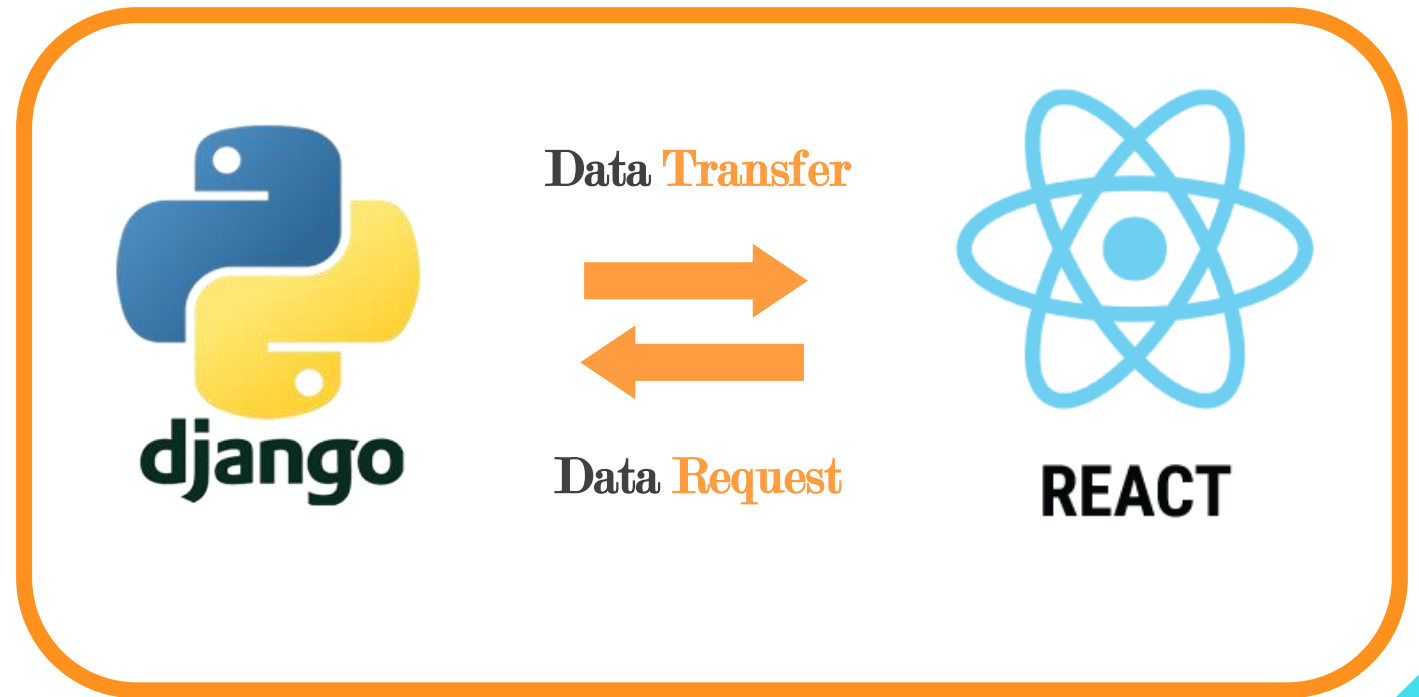
Data **Transfer**



Data **Request**
through Query



Project Architecture: Django & React.js



Git_Version Control

Architecture

README.md

druid-monitoring-project

prerequisite

- python 3.6.x
- pip3
- virtualenv
- nodejs higher than 8.x.x

how to install

- clone the project
- virtualenv {virtualevne_name}
- source {/virtualevne_name}/bin/activate
- pip install -r requirement.txt
- npm install -f

how to run

- npm run dev
- python manage.py {ip}:{port}



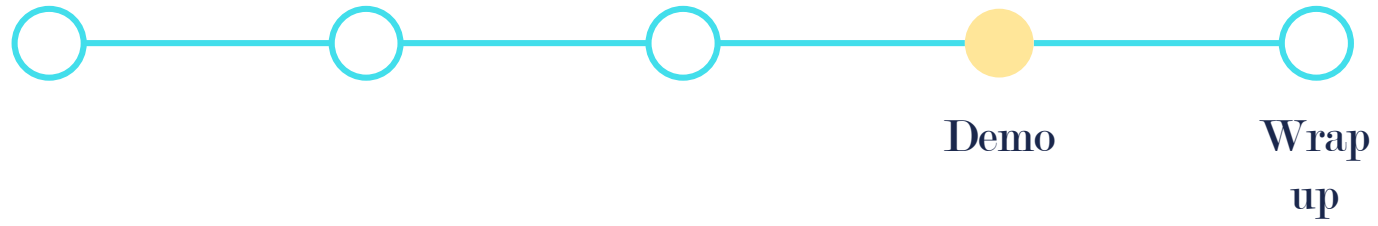
[Code](#) [Issues 0](#) [Pull requests 1](#) [Projects 0](#) [Wiki](#) [Insights](#)

No description, website, or topics provided.

[43 commits](#) [3 branches](#) [0 releases](#) [3 contributors](#)

Branch: master [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

pouder-Man final		Latest commit 552b054 2 days ago
.idea	final	2 days ago
data	final	2 days ago
django_and_react	final	2 days ago
frontend	final	2 days ago
.babelrc	final	2 days ago
.gitignore	final	2 days ago
README.md	final	2 days ago
init.sh	final	2 days ago
macarons.js	final	2 days ago
manage.py	final	2 days ago
nohup.out	final	2 days ago
package.json	final	2 days ago
requirement.txt	final	2 days ago
webpack.config.js	final	2 days ago



Demonstration



Wrap
up

Wrap up
소감