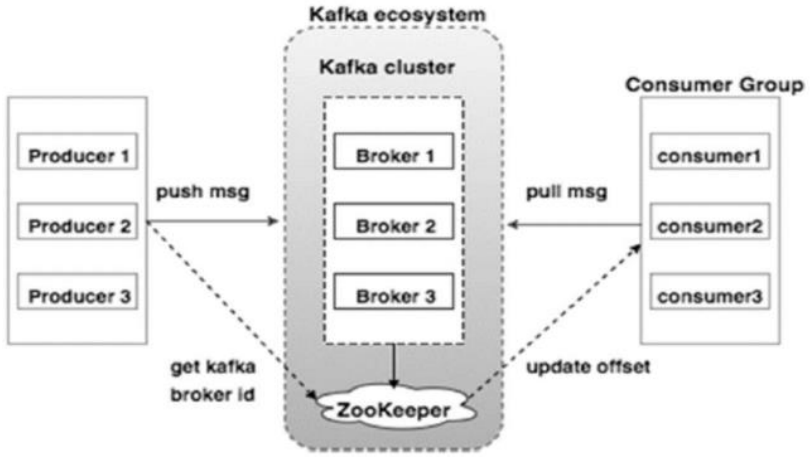
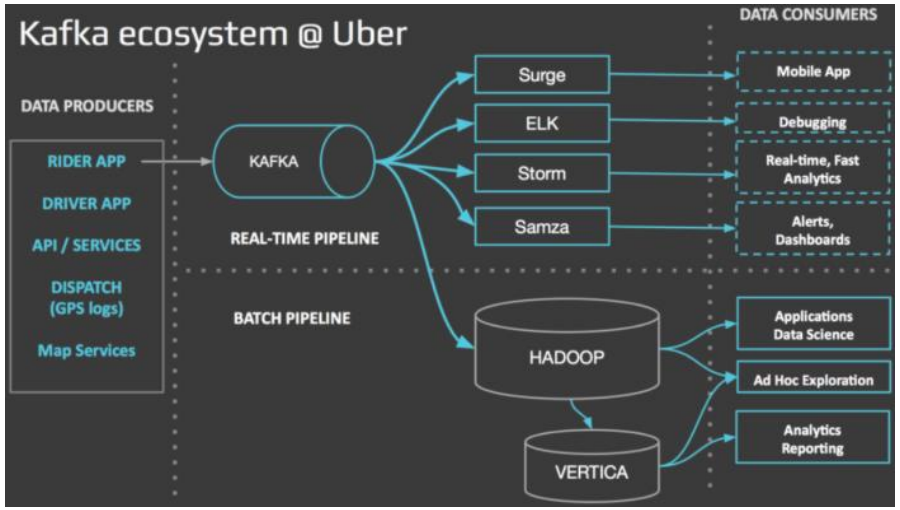
Kafka

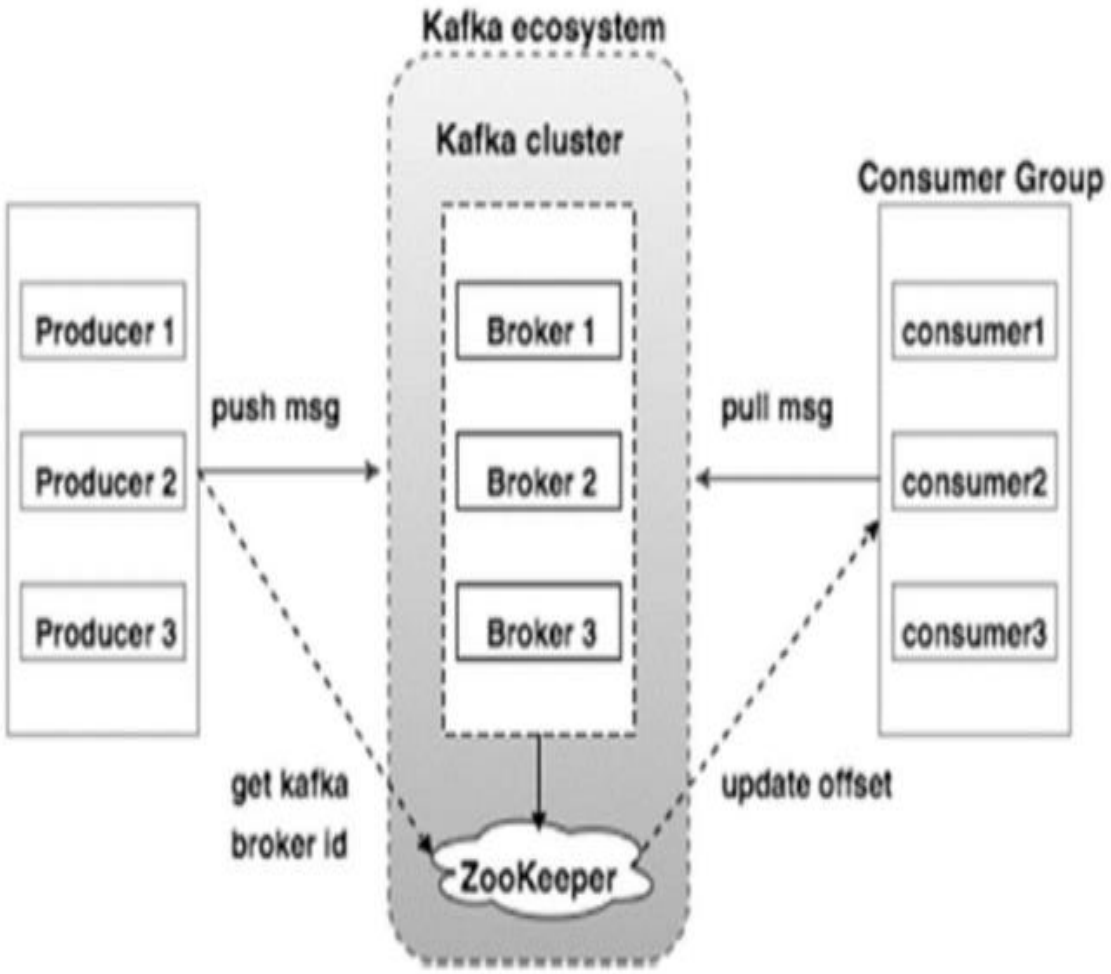




Kafka?

- 분산 비동기 메시징 시스템
- Linkedin에서 개발 되어 2011년 오픈 소스로 공개
- Pub/Sub 형태의 Model
- 디스크에 메시지 저장
  - 일반적인 메시징 시스템들은 Consumer가 메시지를 읽어가면 큐에서 바로 메시지 삭제
  - Kafka는 보관 주기 동안 디스크에 메시지를 저장 ⇒장애 발생시 다시 메시지 처리(Rewind)가능
- Multi Producer와 Multi Consumer로 구성 가능
- Zookeeper 사용
  - 클러스터 내의 broker에 대한 분산 처리 담당

Kafka



Producer(Publisher)

- 데이터 단위를 보내는 부분
- 특정 Topic의 메시지를 생성한 뒤 해당 메시지를 Broker에 전달

Consumer(Subscriber)

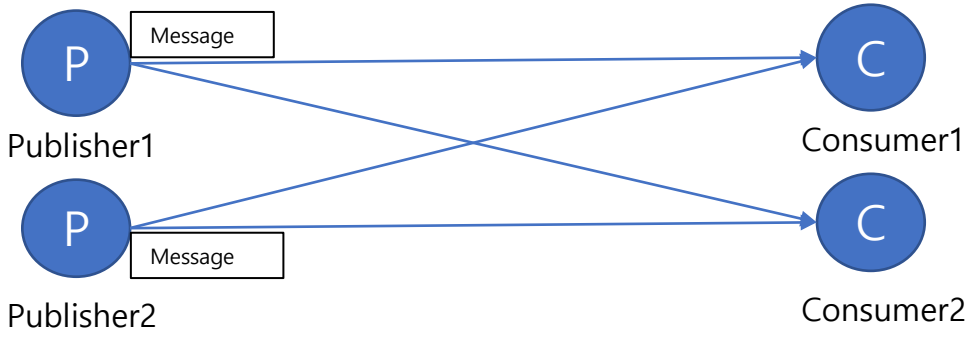
- Topic이라는 메시지 저장소에 저장된 데이터를 가져가는 부분
- 해당 Topic을 구독하는 Consumer들이 Broker에서 메시지를 가져가서 처리

Broker

- Topic을 기준으로 메시지를 관리
- Producer가 전달 받은 메시지를 Topic 별로 분류 및 저장

PUB/SUB Model

일반적인 형태의 네트워크 통신



장점

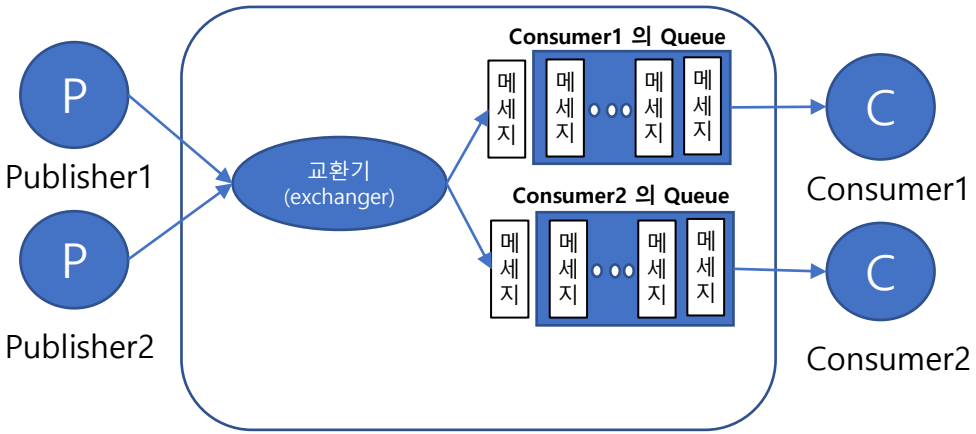
- 빠른 전송 속도
- 전송 결과를 신속하게 알 수 있음

단점

- 통신에 참여하는 개체에 문제가 생겼을 시 메시지를 보내는 쪽에서 대기 처리 등을 개별적으로 진행 해줘야 함
- 통신에 참여하는 개체가 많아질수록 확장성이 좋지 못함

PUB/SUB Model

PUB/SUB 형태의 네트워크 통신



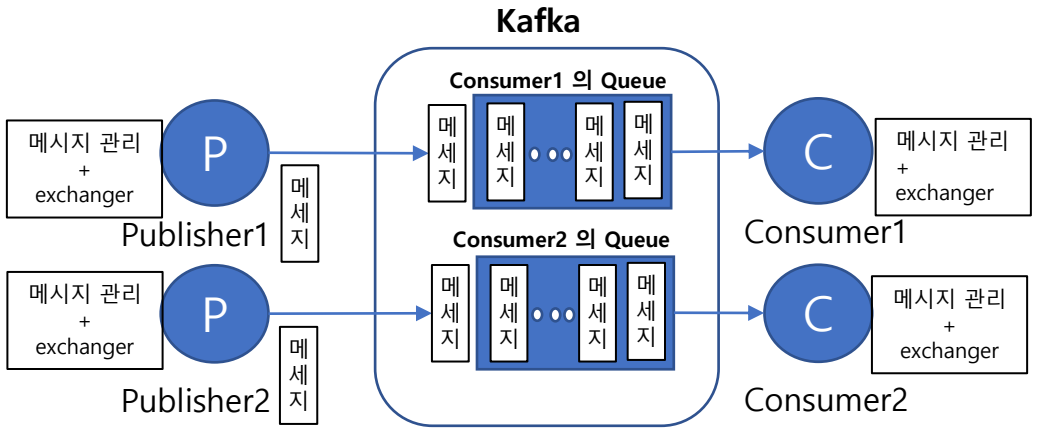
장점

- 개체가 하나 빠지거나 수신 불능 상태가 되었을 때에도, 메시징 시스템만 살아있으면 Producer에서 전달된 메시지가 유실 되지 않는다. 불능 상태의 개체가 다시 회복 되면 언제든지 메시지를 다시 가져 갈 수 있음
- 메시징 시스템 중심으로 운영되므로 확장성이 용이하다.
- 사용자가 직접 연결을 관장하는 것이 아니라,교환기의 룰에 따라 데이터가 전달되므로 메시지 데이터 유실의 염려가 없다.

단점

- 직접 통신을 하지 않기 때문에 메시지가 정확하게 전달되었는지 확인하려면 코드가 복잡해 짐
- 통신 중간에 메시징 시스템이 있기 때문에 속도가 빠르지 않음

Kafka System



Kafka의 간단한 아키텍처

Producer는 새로운 Message를 Kafka로 전송

Producer에 보낸 Message를 Kafka에 Consumer Queue(Kafka에서는 Topic)에 저장

Consumer 는 Kafka sever에 접속하여 새로운 Message를 가져 감

기존 메시징 시스템과의 차이점

- 메시지 교환 전달의 신뢰성 관리를 Producer와 Consumer쪽으로 넘김
- 부하가 많이 걸리는 교환기 기능 역시 Consumer가 만들 수 있게 함
- 메시징 시스템 내에서의 작업량을 줄임=> 메시징 전달 성능에 집중 시킴

Kafka 특징

비동기 메시지 시스템

- 메일을 보내면 메일 서버에 저장. 메일을 받는 사람은 자신이 필요할 때 수신하는 비동기 구조
- 메일 시스템과 동일하게 Producer가 메시지를 보내면 Kafka Cluster에 저장, Consumer가 필요할 때 메시지를 가져오는 구조

Kafka 특징

Producer와 Consumer 역할 분리

- 하나의 Topic에 대하여 여러 Producer가 Message를 전송하고 여러 Consumer가 Message를 소비하는 구조로 중앙 집중형 구조로 구성
- 확장성
  - Kafka Cluster를 통하여 확장이 어느 정도 용이하게 설계됨
- High Performance

Zookeeper

Zookeeper : 분산 어플리케이션을 위한 분산 관리 서비스

◆ **분산 어플리케이션을 위한 코디네이터**

◆ **분산 시스템**
- 복수의 컴퓨터가 네트워크를 통해 통신하며 하나의 목적을 위해 서로 간에 상호작용 하는 것
  다수의 컴퓨터가 마치 하나인 것처럼 동작하는 시스템

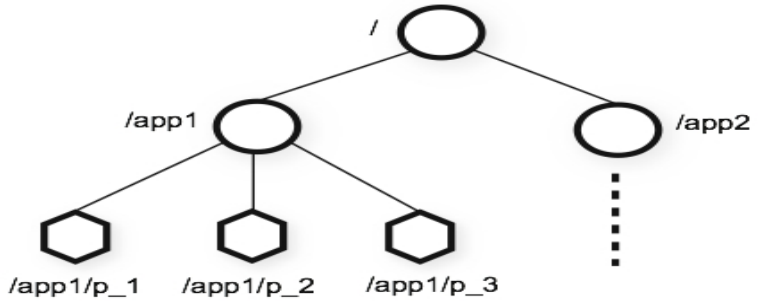
◆ **Zookeeper 사용 용도**
- 설정 관리(Configuration management)
  ⇒클러스터의 설정 정보를 최신으로 유지하기 위한 조율 시스템으로 사용됨
- 클러스터 관리(Cluster management)
  ⇒ 클러스터의 서버가 추가되거나 제외될 때 그 정보를 클러스터 안 서버들이 공유하는데 사용
- 리더 채택(Leader selection)
  ⇒ 다중 어플리케이션 중에서 어떤 노드를 리더로 선출할 지를 정하는 로직을 만드는데 사용
  주로 복제된 여러 노드 중 연산이 이루어지는 하나의 노드를 택하는 데 사용
- 락, 동기화 서비스(Locking and synchronization service)
  ⇒ 클러스터에 쓰기 연산이 빈번할 경우 경쟁상태에 들어갈 가능성이 커짐 -> 데이터 불일치 발생
  클러스터 전체에 대상을 동기화해(락을 검) 경쟁상태에 들어간 경우를 사전에 방지

Zookeeper

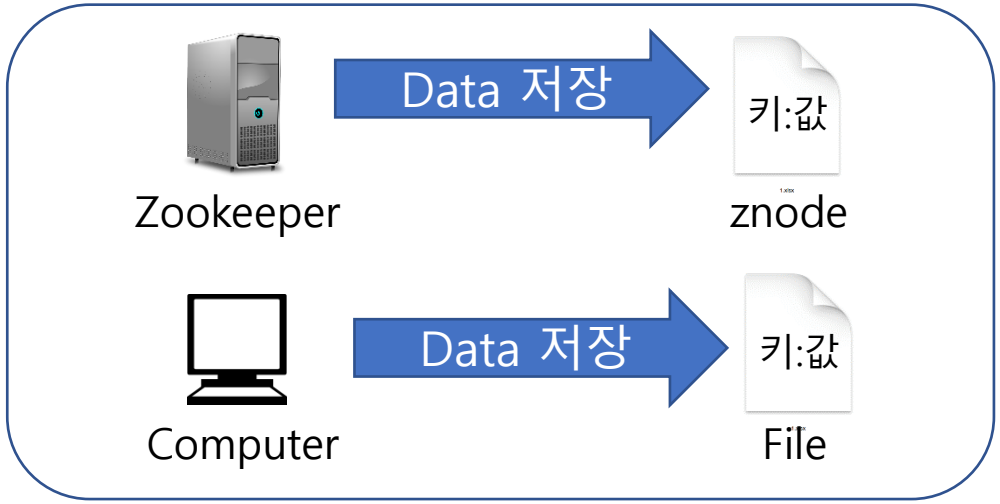
◆ **특징 – 자체적으로 클러스터링 제공, 장애에도 데이터 유실 없이 fail over/fail back이 가능**

◆ **데이터 모델**

-디렉토리 구조기반으로 znode라는 데이터 저장 객체를 제공
이 객체에 데이터를 넣고 빼는 기능만을 제공

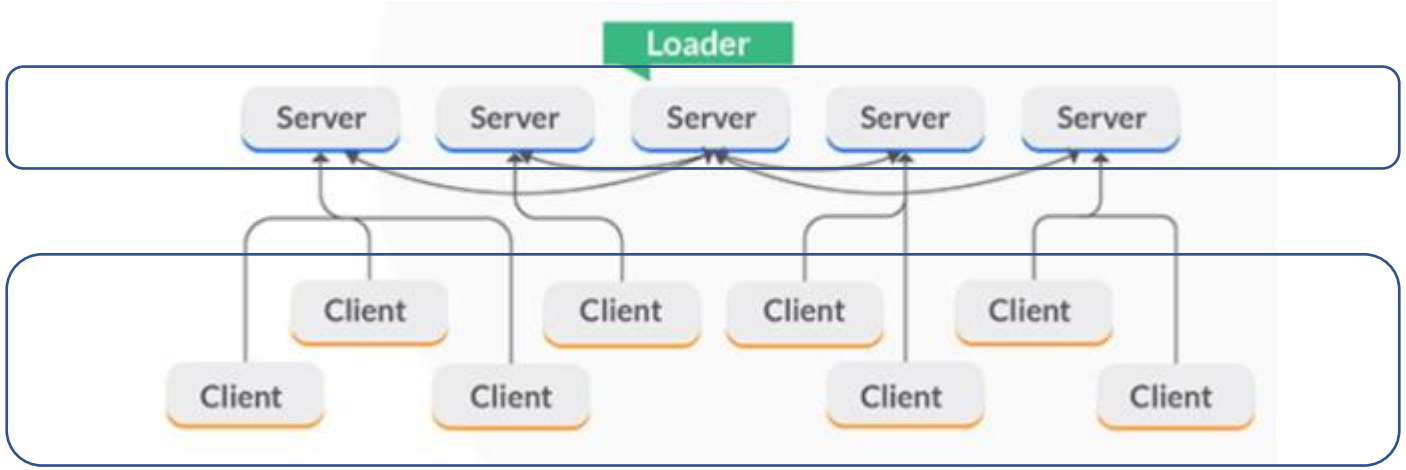


Zookeeper 계층형 구조



znode와 컴퓨터 파일 비교

Zookeeper



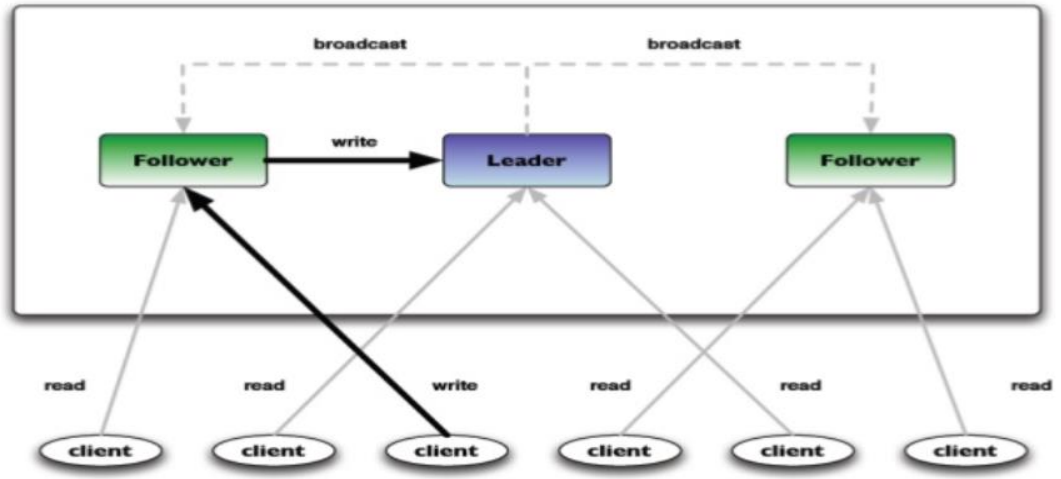
앙상블(클러스터)
-> 홀수로 구성

분산 어플리케이션

Zookeeper Service 구성도

- ◆ Zookeeper 서비스는 복수의 서버에 복제 된다.
- ◆ 모든 서버 장치는 데이터의 사본을 메모리에 저장한다.
- ◆ 서비스 기동(Startup)시 리더가 선출된다.
- ◆ 클라이언트들은 하나의 Zookeeper서버에 TCP/IP로 연결을 실행하고 유지한다.
- ◆ 클라이언트는 모든 Zookeeper 서버에서 읽을 수 있으며,
  리더를 통해 쓸 수 있되 과반수 서버의 승인(합의)이 필요하다.

Zookeeper



Znode 읽기/쓰기

- 조회 요청은 클라이언트가 연결한 Zookeeper 서버 내에서 처리된다.
- 쓰기 요청은 리더로 전달되며, 클라이언트로 정상 응답하기 전에 과반수 이상의 서버에서 쓰기가 완료되어야 한다.

Zookeeper

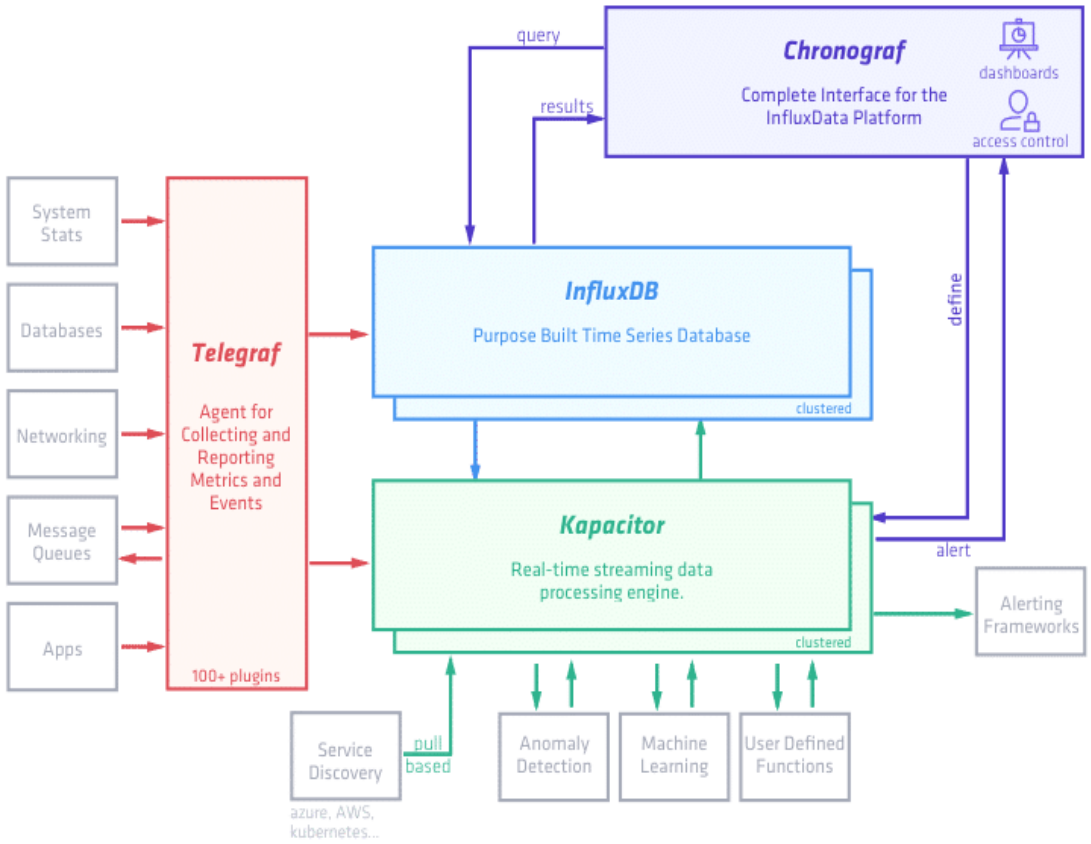
Zookeeper 앙상블 3대 구성 후 노드 1대 다운 : 과반수 유지 되므로 서비스 가능

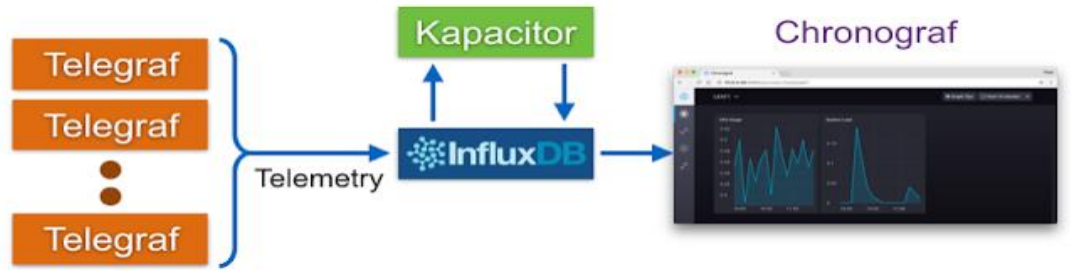
Zookeeper 앙상블 3대 구성 후 노드 2대 다운 : 과반수 유지되지 않기 때문에 서비스 불가능

Zookeeper 앙상블 5대 구성 후 노드 2대 다운 : 과반수 유지 되므로 서비스 가능

Zookeeper 앙상블과 장애 시나리오

TICK Stack





TICK Stack

- 다양한 DBMS의 데이터 수집 기능
- 커스터마이징 가능(Go언어)
- 변경 가능한 Component

Telegraf

- Go 언어로 쓰여진 Agent
- 주로 수집, 처리에 쓰기 위해 사용
- 다양한 Input, Output Plugin을 지원

InfluxDB

- Go 언어로 쓰여진 시계열(Time Series) Data에 특화된 Database
- Client Interface – REST 기반 Wrapper library
- Query Language – SQL Like(InfluxQL)

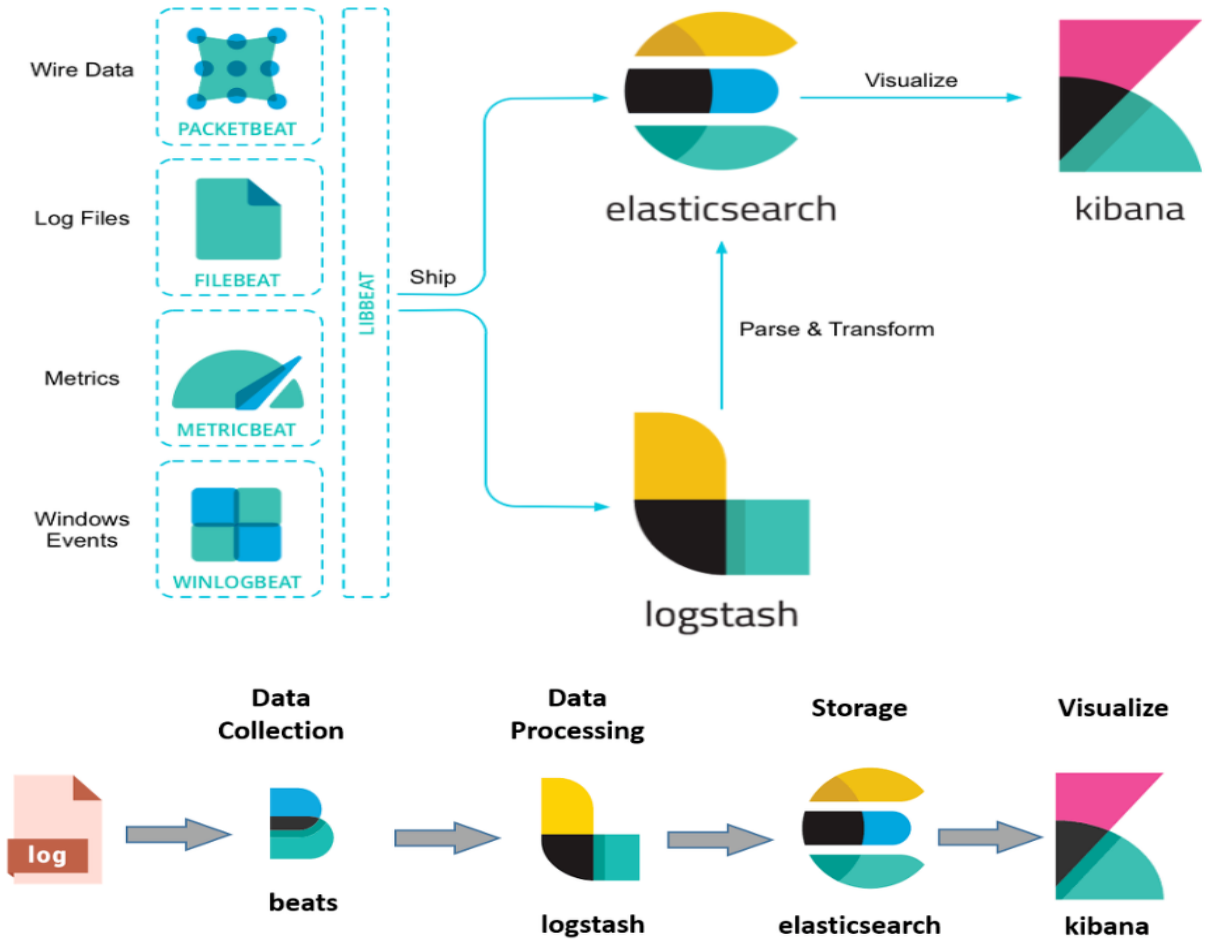
Chronograf

- Data 시각화 툴
- Grafana로 대체 가능

Kapacitor

- Real-Time 스트리밍 데이터 전송 및 Alert
- SPC 연산을 위해 사용

ELK Stack



ELK Stack

- Elasticsearch, Logstash 및 Kibana, 이 오픈 소스 프로젝트 세 개의 머리글자
- Data 검색 및 분석에 용이

Beats

- 단말 장치의 데이터를 Logstash 및 Elasticsearch로 전송하는 경량 수집기용 플랫폼

Elasticsearch

- 분산 시스템으로 수평적인 확장성, 최고의 안정성 및 간편한 관리를 위해 설계된 JSON문서 기반의 검색 및 분석 엔진
- 분산형 RESTful 검색 및 분석 엔진

Kibana

- 데이터 시각화
- Elastic Stack의 모든 기능을 구성 및 관리할 수 있는 확장 가능한 UI도구

Logstash

- 확장 가능한 플러그인 에코 시스템으로 구성된 동적 데이터 수집 파이프 라인
- 다양한 소스에서 동시에 데이터를 수집하고 변환하여 자주 사용하는 Stash 보관소로 보냅니다.