목차

[Introduction 1](#_Toc42103537)

[Zookeeper, Kafka 설치 및 설정. 2](#_Toc42103538)

[도메인 설정 3](#_Toc42103539)

[Zookeeper 기본 설정 3](#_Toc42103540)

[Kafka 기본 설정 4](#_Toc42103541)

[Zookeeper 기본 명령어 4](#_Toc42103542)

[Kafka 기본 명령어 4](#_Toc42103543)

[Kafka MongoDB Connect 생성 및 설정. 5](#_Toc42103544)

[Worker(Connect) configuration 5](#_Toc42103545)

[Plugin install (MongoDB) 6](#_Toc42103546)

[MongoDB, Kafka 테스트 6](#_Toc42103547)

[전체 Flow 7](#_Toc42103548)

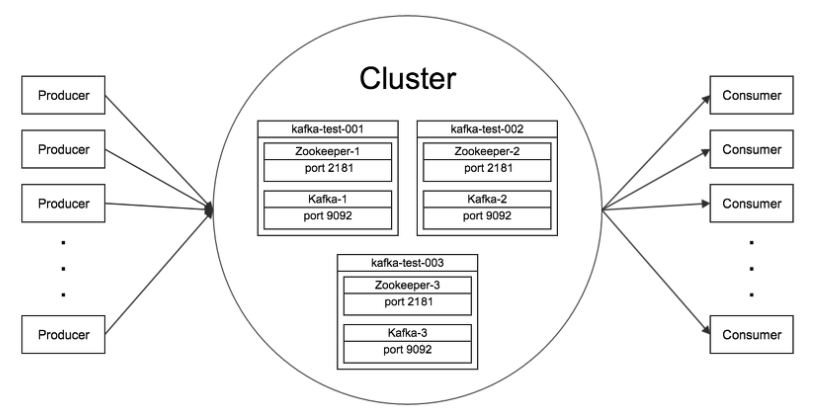
# **Introduction**

Kafka 에 대한 원론적인 접근은 많이 하고 있으므로, 여기서는 실질적으로 설치 및 설정, 실행하는 전체 흐름을 짚어본다.

이 전체 과정의 환경은 Docker 3개로 cluster 환경을 조성해 1개의 host에 3개의 docker를 띄워놨다. 그러므로 외부 ip와 내부 ip를 설정할 때 고민을 많이 했다.

다른 환경에서 환경을 조성할 경우에는 사용하는 host-name(/etc/hosts), Directory, Port을 깊게 고려해봐야한다.

전체적인 개념을 이해한 상태에서 시작했으며 모르는 부분은 (갓)구글과 카프카, 데이터 플랫폼의 최강자(고승범 저)를 참고하였습니다.



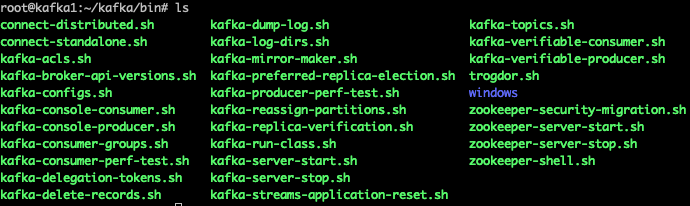
[클러스터 구축]

구성한 환경과 유사한 그림으로 별도의 zookeeper를 사용하고 port를 3개 모두 통일시켰다.

# **Zookeeper, Kafka 설치 및 설정.**

설치 과정은 공식 다운로드 사이트에 접속해서 설치.

1. Zookeeper, Kafka 설치.
2. 압축 해제 및 링크 설정.



[ Kafka 실행 파일 ]

Kafka 내부에는 다양한 실행파일들이 존재하며 헷갈릴 수도 있는 부분이 kafka의 실행 파일 중 내부에도 Zookeeper가 존재하고 있어 별도의 zookeeper 설치의 필요에 의문성을 느낄 수 있다.

* 별도 zookeeper 설치의 장점은 지금은 **한개의 장비에 환경을 구축하고 있기 때문에** 완벽하게 장점을 살렸다고 할 수 는 없지만 다른 장비에 zookeeper cluster를 설치해서 kafka cluster에 연결을 하는 것이 장점을 살릴 수 있다.

여기서는 별도의 Zookeeper cluster를 형성하였다.

이 실행 파일 중 **connect-\*.sh** , **kafka-console-\*.sh** , **kafka-topics.sh** , **kafka-server-\*.sh**

4가지 실행 파일을 주로 사용하게 된다.

\*\* 설정은 내가 한 환경을 중심으로 설명한다.

## 도메인 설정

/etc/hosts 파일에 cluster node들의 static ip 와 host name 을 등록해준다.

나같은 경우에는 아래와 같이 등록해주었다.

172.28.0.1 kafka1

172.28.0.2 kafka2

172.28.0.3 kafka3

172.28.0.4 mongodb

## Zookeeper 기본 설정

**위치 - <path to zookeeper>/conf**

# zoo\_sample.cfg 를 zoo.cfg로 복사

cp zoo\_sample.cfg zoo.cfg

zoo.cfg 파일 수정.

# 이 부분이 많은 블로그들에서 잘못 말하고 있는데 dataDir 경로에는 호스트마다 번호를 갖고있 는 myid 파일을 갖고 있어야 한다.

Ex) host\_1 의 경로 설정이 dataDir = /data 이면 /data/myid 에 1 이 작성되어야 한다.

dataDir = <path to dir>

# zoo.cfg에 cluster를 구성할 zookeeper들의 domain을 등록해준다.

## 여기서 kafka1, kafka2, kafka3은 zookeeper의 hostname을 나타낸다.

### 2888은 zookeeper 서버들의 동기를 위한 포트(zookeeper 간의 통신 채널)

3888은 zookeeper 클러스터에서 leader를 선출하는 포트.

echo server.1=kafka1:2888:3888 >> /root/zookeeper/conf/zoo.cfg

echo server.2=kafka2:2888:3888 >> /root/zookeeper/conf/zoo.cfg

echo server.3=kafka3:2888:3888 >> /root/zookeeper/conf/zoo.cfg

## Kafka 기본 설정

**위치 - <path to kafka>/config/**

server.properties (kafka server configuration 파일) 수정

# log를 담을 directory 설정.

mkdir /data1 /data2

(60번째 줄) log.dirs = /data1, /data2

# listener는 내부, advertised.listeners는 외부에서 쓰인다고 많이들 설명하는데,

# 여기선 같은 host에서 접근하다보니 port를 9092로 고정해줬다.

(31번째 줄) listeners=PLAINTEXT://:9092

echo advertised.listeners=PLAINTEXT://${host\_name}:9092

# zookeeper 와 설정해주는 부분으로 host\_name:port로 해주며 cluster 이다보니 리스트 형태로 넣었다. 여기서 twitter는 Znode로 zookeeper에서 선행적으로 만들어주어야한다.

(123번째 줄) zookeeper.connect=kafka1:2181,kafka2:2181,kafka3:2181/twitter

## Zookeeper 기본 명령어

## Kafka 기본 명령어

# 토픽 리스트 확인.

bin/kafka-topics.sh --list --zookeeper kafka1:2181,kafka2:2181,kafka3:2181/twitter

# console producer 실행

bin/kafka-console-producer.sh --broker-list localhost:9092 --topic topicname

# console consumer 실행

bin/kafka-console-consumber.sh –bootsrap-server localhost:9092 –topic topicname

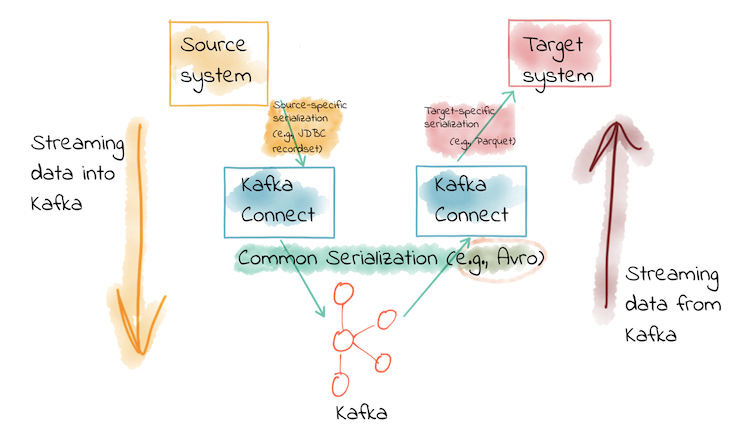
# topic 생성

bin/kafka-topics.sh --create --zookeeper kafka1:2181,kafka2:2181,kafka3:2181/twitter --replication-factor 3 --partitions 1 --topic connect-configs

# **Kafka MongoDB Connect 생성 및 설정.**

Kafka connect는 Kafka의 component로 Consumer API와 Producer API로 대체할 수 있지만

긴밀한 연결을 위해 사용된다.



개인적으로 connect를 가장 잘나타내는 그림으로 connect에 schema registry을 연동시켜 공통된 스키마 지정을 지원한다.

ELK Stack의 Logstash, Fluentd 등이 서로 다른 시스템 간의 브리지 역할을 이미 하고 있지만, Connect는 실시간 데이터 전달을 할때 이미 처리되거나 실패한 데이터를 추적한다거나, 분산처리, 작업 배포 등에 강점을 갖는다.

우리가 해볼 것은 Kafka – Kafka connect – Sink – MongoDB

Kafka로부터 데이터를 받아 MongoDB로 저장하는 것이다.

## Worker(Connect) configuration

Ref.

* <https://docs.confluent.io/3.2.0/connect/userguide.html#prerequisites>
* <https://www.confluent.io/blog/getting-started-mongodb-connector-for-apache-kafka-and-mongodb/>
* <https://docs.mongodb.com/kafka-connector/master/kafka-sink-data-formats/>
* <https://swalloow.github.io/kafka-connect>

실행파일로는 bin/connect-\*.sh 을 사용하고, 설정파일로는 config/connect.properties 를 생성해서 사용한다.

echo bootstrap.servers=kafka1:9092,kafka2:9092,kafka3:9092 >> connect.properties

echo **plugin.path=/root/kafka/plugins** >> connect.properties

echo internal.key.converter=org.apache.kafka.connect.json.JsonConverter >> connect.properties echo internal.value.converter=org.apache.kafka.connect.json.JsonConverter >> connect.properties

echo internal.key.converter.schemas.enable=false >> connect.properties

echo internal.value.converter.schemas.enable=false >> connect.properties

echo offset.storage.file.filename=/tmp/connect.offsets >> connect.properties

echo offset.storage.topic=offset-test >> connect.properties

echo \#rest.host.name=connect >> connect.properties

echo \#rest.port=8083 >> connect.properties

echo group.id=test >> connect.properties

echo config.storage.topic=config-test >> connect.properties

echo config.storage.replication.factor=1 >> connect.properties

echo status.storage.topic=status-test >> connect.properties

## Plugin install (MongoDB)

여기선 MongoDB를 연결하다보니 MongoDB Connect plugin을 설치하고 Path를 잡아줬다.

* 여기서 문제가 생긴게 Confluent의 connect plugin 와 Apache 의 connect plugin 구별을 잘해야하며, 사용 목적에 따라 Sink와 Source를 구분할 수 있어야한다.
* 나 같은 경우에는 이상한 plugin 파일을 다운로드 받아와서 요청을 날리면 ClassDenotFound exception을 일으켰다. 압축 해제한 폴더를 집어넣음으로 해결했다.

## MongoDB, Kafka 테스트

Kafka/config/connect.properties 파일에 따라 topic 생성을 먼저 해준다.

# config.storage.topic=connect-configs

bin/kafka-topics.sh --create --zookeeper kafka1:2181,kafka2:2181,kafka3:2181/twitter --replication-factor 3 --partitions 1 --topic connect-configs

# offset.storage.topic=connect-offsets

bin/kafka-topics.sh --create --zookeeper kafka1:2181,kafka2:2181,kafka3:2181/twitter --replication-factor 3 --partitions 50 --topic connect-offsets

# status.storage.topic=connect-status

bin/kafka-topics.sh --create --zookeeper kafka1:2181,kafka2:2181,kafka3:2181/twitter --replication-factor 3 --partitions 10 --topic connect-status

* Connect 실행.

bin/connect-distributed.sh config/connect.properties

* curl 날리기.

curl -X PUT http://kafka1:8083/connectors/sink-mongodb-users/config -H "Content-Type: application/json" -d ' {

"connector.class":"com.mongodb.kafka.connect.MongoSinkConnector",

"tasks.max":"1",

# Kafka의 topic

"**topics**":"newuser",

# 연결할 MongoDB uri, Database, collection 입력.

* Database, collection은 생성을 해놓지 않아도 자동으로 생성되며,

연결한 mongodb 서버의 외부 접속을 허용해줘야 한다.

"**connection.uri**":"mongodb://mongodb:27017",

"database":"BigBoxStore",

"collection":"users",

# kafka에서 받은 데이터를 converting 하는 방법 설정.

여기서 schema registry도 함께 적용가능

"key.converter":"org.apache.kafka.connect.json.JsonConverter",

"key.converter.schemas.enable":false,

"value.converter":"org.apache.kafka.connect.json.JsonConverter",

"value.converter.schemas.enable":false

}'

curl -X PUT http://kafka1:8083/connectors/sink-mongodb-users/config -H "Content-Type: application/json" -d ' {

"connector.class":"com.mongodb.kafka.connect.MongoSinkConnector",

"tasks.max":"1",

"topics":"stream\_data\_1",

"connection.uri":"mongodb://mongodb:27017”,

"database":"twitter",

"collection":"data\_1",

"key.converter":"org.apache.kafka.connect.json.JsonConverter",

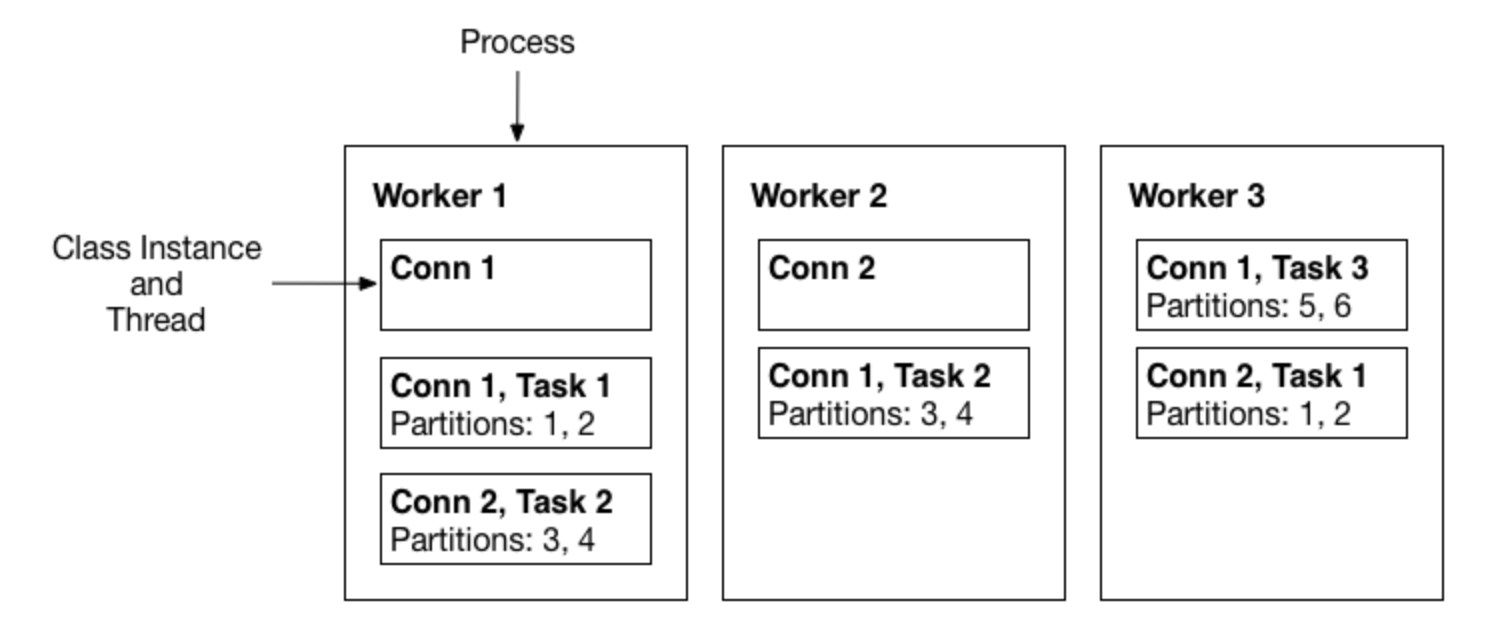
"key.converter.schemas.enable":false,

"value.converter":"org.apache.kafka.connect.json.JsonConverter",

"value.converter.schemas.enable":false

}'

## 전체 Flow



3대 Zookeeper 서버 실행 – 3대 Kafka 서버 실행 – 3대 Kafka Connect 실행 – 한곳에 MongoDB 연결.