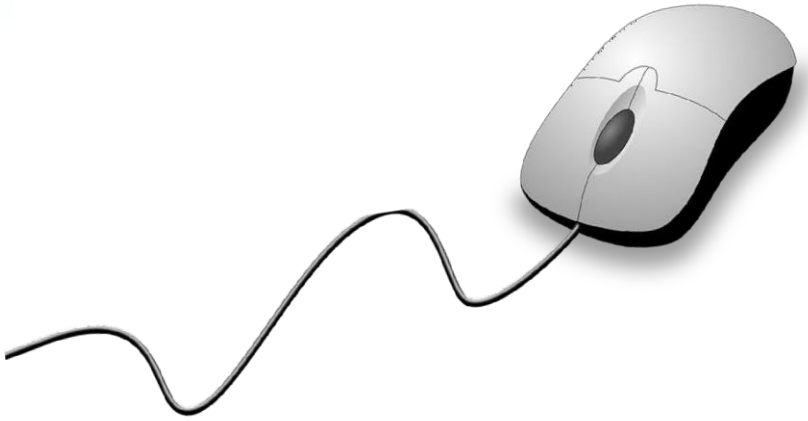
공개SW 솔루션 설치 & 활용 가이드

미들웨어 > WEB/WAS 서버



APACHE kafka

A distributed streaming platform

제대로 배워보자

How to Use Open Source Software

Open Source Software Installation & Application Guide

OPENUP

오픈소스 소프트웨어 통합지원센터

Open Source Software Support Center

CONTENTS

1. 개요

소개	• 2011년 LinkedIn에서 개발된 분산 메시징 시스템 • 2014년 아파치를 통해 오픈 소스화된 프로젝트에 편입됨		
주요기능	• Publisher Subscriber 모델 • 고가용성(High availability) 및 확장성(Scalability) • 디스크 순차 저장 및 처리(Sequential Store and Process in Disk) • 분산 처리(Distributed Processing)		
대분류	• 미들웨어	**소분류**	• WEB/WAS서버
라이선스형태	• Apache License v2.0	**사전설치 솔루션**	• 없음
		버전	• 2.3.0 (2019년 8월 기준)
특징	• 메시지를 메모리에 저장하지 않고 파일 시스템에 저장 • TCP 기반의 Protocol을 사용하여 오버헤드를 감소 • 컨슈머 그룹이라는 개념을 도입해서 큐와 발행/구독 모델을 모두 지원 • 스트리밍 플랫폼을 구성하는 노드에 문제가 생기더라도 데이터를 안전하게 저장		
개발회사/커뮤니티	• LinkedIn		
공식 홈페이지	• https://kafka.apache.org/		

2. 기능요약

- kafka 주요 기능

메시징 서버	• 다양한 시스템과 연결되어 높은 신뢰/확장성의 엔터프라이즈 메세징 시스템 • 배치 소비자를 지원하며, 온라인, 오프라인에 저지연율(Low latency)을 보장.
분산 시스템	• 메세지 분산 처리로 인하여 단일 시스템보다 높은 성능을 제공한다. • 생산자 중심적이며, 엄청난 이벤트 데이터을 파티셔닝하는데 기반을 둔다.
확장성	• 3대의 브로커로 시작해 수십대의 브로커로 확장 가능 ✓ 브로커 : 카프카 애플리케이션이 설치되어 있는 서버
안정성 & 고가용성	• 파일 시스템에 메시지를 저장하기 때문에 별다른 설정 없이도 데이터의 영속성(Persistence)가 보장. • 무중단 확장 가능
고성능 디자인	• 메시지를 메모리대신 파일 시스템에 쌓아두고 관리 • 메모리에 별도의 캐시를 구현하지 않고 OS의 페이지 캐시에 위임 후 미리 읽어 들여(readahead) 디스크 읽기 성능을 향상

3. 실행환경

- OS 플랫폼 종류에 따른 지원
  - Linux: 32-bit, 64-bit, and ARM.
  - Mac OS X: 32-bit and 64-bit.
  - Microsoft Windows: 32-bit and 64-bit.

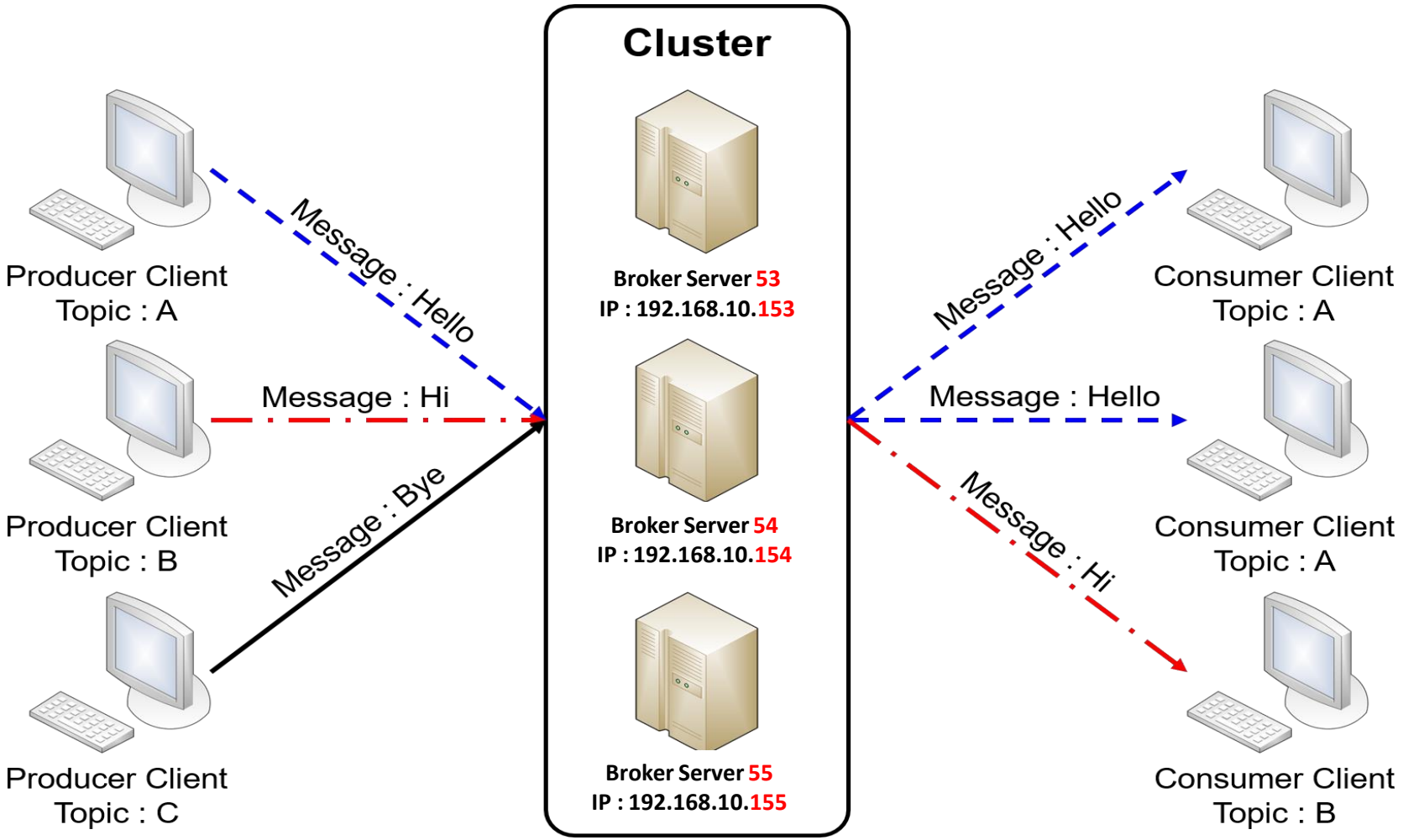
4. 설치 및 실행

세부 목차

1. kafka 클러스터 구성도
2. kafka 구성을 위한 사전 설정
3. kafka 서버 압축/설치 파일 다운로드
4. kafka 클러스터 - zookeeper 설정
5. kafka 클러스터 – kafka server 설정
6. kafka 클러스터 실행 –zookeeper-server
7. kafka 클러스터 실행 –kafka-server

4. 설치 및 실행

4.1 kafka 클러스터 구성도



4. 설치 및 실행

4.2 kafka 구성을 위한 사전 설정

1. 사용자 그룹 및 계정 생성

```
# groupadd –g 711 kafka
# useradd –g 711 –u 711 –m kafka
```

2. kafka cluster 구성을 위한 디렉토리 생성

```
# mkdir /data3
```

3. 생성한 디렉토리에 소유권 설정

```
# chown kafka:kafka /data3
```

4. 사용자 전환 및 해당 폴더로 이동

```
# su - kafka
# cd /data3
```

4. 설치 및 실행

4.3 kafka 서버 압축/설치 파일 다운로드

- OS Version : CentOS Linux release 7.6.1810 (Core)
- kafka Version : kafka-2.3.0
- kafka 공식 홈페이지 에서 다운

http://kafka.apache.org/downloads

<2019년 8월 10일 기준으로 최신버전이 2.3.0>

1. 소스 파일 다운로드

   # wget **http://mirror.navercorp.com/apache/kafka/2.3.0/kafka_2.12-2.3.0.tgz**

2. 소스 파일 압축 해제

   # tar -xvzf kafka_2.12-2.3.0.tgz

3. 해당 폴더로 이동

   # cd kafka_2.12-2.3.0/bin

4. 버전 확인

   # ./kafka-config.sh --version

```
[kafka@ha-twx-svr-01.novalocal /data3/kafka_2.12-2.3.0/bin]$ ./kafka-configs.sh --version
2.3.0 (Commit:fc1aaa116b661c8a)
[kafka@ha-twx-svr-01.novalocal /data3/kafka_2.12-2.3.0/bin]$
```

4. 설치 및 실행

4.4 kafka 클러스터 설정 -zookeeper

1. config/zookeep.properties에 클러스터 서버 정보 입력

```
# the directory where the snapshot is stored.
dataDir=/tmp/zookeeper
# the port at which the clients will connect
clientPort=2181
# disable the per-ip limit on the number of connections
maxClientCnxns=0

initLimit=1
syncLimit=3

server.53=192.168.10.153:2888:3888
server.54=192.168.10.154:2888:3888
server.55=192.168.10.155:2888:3888
```

```
dataDir=/tmp/zookeeper
clientPort=2181   maxClie
ntCnxns=0

initLimit=1
syncLimit=3

server.53=192.168.10.153:2888:3888
server.54=192.168.10.154:2888:3888
server.55=192.168.10.155:2888:3888
```

2. /tmp/zookeeper 폴더 생성 및 BrokerID 구성 파일 생성

2.1 # broker 53번 서버 (192.168.10.153)

```
# mkdir /tmp/zookeeper
# echo 53 > /tmp/zookeeper/myid
```

2.2 # broker 54번 서버 (192.168.10.154)

```
# mkdir /tmp/zookeeper
# echo 54 > /tmp/zookeeper/myid
```

2.3 # broker 55번 서버 (192.168.10.155)

```
# mkdir /tmp/zookeeper
# echo 55 > /tmp/zookeeper/myid
```

4. 설치 및 실행

4.5 kafka 클러스터 설정 –kafka-server

1. config/server.properties에 클러스터 서버 정보 입력

1.1 # broker 53번 서버 (192.168.10.153)

```
broker.id=53
listeners=PLAINTEXT://:9092  advertised.listeners=PLAINTEXT://192.168.10.153:9092
zookeeper.connect=192.168.10.153:2181, 192.168.10.154:2181, 192.168.10.154:2181
```

1.2 # broker 54번 서버 (192.168.10.154)

```
broker.id=54
listeners=PLAINTEXT://:9092  advertised.listeners=PLAINTEXT://192.168.10.154:9092
zookeeper.connect=192.168.10.153:2181, 192.168.10.154:2181, 192.168.10.154:2181
```

1.3 # broker 55번 서버 (192.168.10.155)

```
broker.id=55
listeners=PLAINTEXT://:9092  advertised.listeners=PLAINTEXT://192.168.10.155:9092
zookeeper.connect=192.168.10.153:2181, 192.168.10.154:2181, 192.168.10.154:2181
```

4. 설치 및 실행

4.6 kafka 클러스터 실행 –zookeeper-server

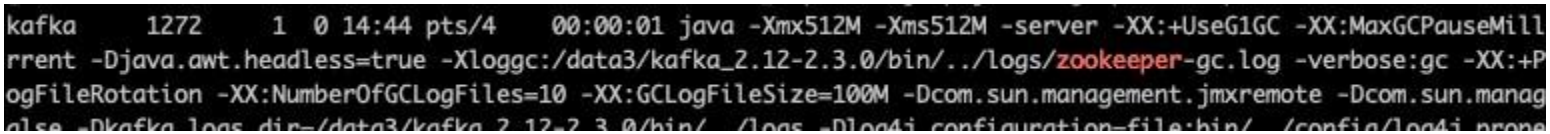
1. zookeeper-server 구동

    1. # broker 53/54/55번 서버 공통

```
# cd /data3/kafka_2.12-2.3.0
# sh bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
```

2. zookeeper-server 구동 확인 (공통)

    1. # broker 53/54/55번 서버 공통 – 프로세스 확인



kafka 1272 1 0 14:44 pts/4 00:00:01 java -Xmx512M -Xms512M ....

...중간생략...

../libs/zkclient-0.11.jar:/data3/kafka_2.12-2.3.0/bin/../libs/zookeeper-3.4.14.jar:/data3/kafka_2.12-2.3.0/bin/../libs/zstd-jni-1.4.0-1.jar org.apache.zookeeper.server.quorum.QuorumPeerMain **config/zookeeper.properties**

2.2 # broker 53/54/55번 서버 공통 (로그 확인 – logs/zookeeper.out)

[2019-08-29 15:17:18,369] INFO Accepted socket connection from /192.168.10.153:5429 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2019-08-29 15:17:18,378] INFO Client attempting to establish new session at /192.168.10.153:5429 (**org.apache.zookeeper.server.ZooKeeperServer**)

4. 설치 및 실행

4.7 kafka 클러스터 실행 –kafka-server

1. kafka-server 구동

    1. # broker 53/54/55번 서버 공통

```
# cd /data3/kafka_2.12-2.3.0
# sh bin/kafka-server-start.sh -daemon config/server.properties
```

2. kafka-server 구동 확인 (공통)

    1. # broker 53/54/55번 서버 공통 (프로세스 확인)

```
[kafka@ha-twx-svr-01.novalocal /data3/kafka_2.12-2.3.0]$ ps -ef|grep -v "grep" | grep kafkaServer
kafka     5705     1  4 15:17 pts/0    00:00:05 java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPause
t -Djava.awt.headless=true -Xloggc:/data3/kafka_2.12-2.3.0/bin/../logs/kafkaServer-gc.log -verbose:gc
```

```
kafka     5705     1 11 15:17 pts/0    00:00:04 java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:MaxGCPauseMillis=20 -
XX:InitiatingHeapOccupancyPercent=35 ....

...중간생략...

...:/data3/kafka_2.12-2.3.0/bin/../libs/validation-api-2.0.1.Final.jar:/data3/kafka_2.12-2.3.0/bin/../libs/zkclient-   0.11.ja
r:/data3/kafka_2.12-2.3.0/bin/../libs/zookeeper-3.4.14.jar:/data3/kafka_2.12-2.3.0/bin/../libs/zstd-jni-1.4.0-1.jar   kafka.
Kafka config/server.properties
```

2.2 # broker 53/54/55번 서버 공통 (로그 확인 – logs/kafkaServer.out)

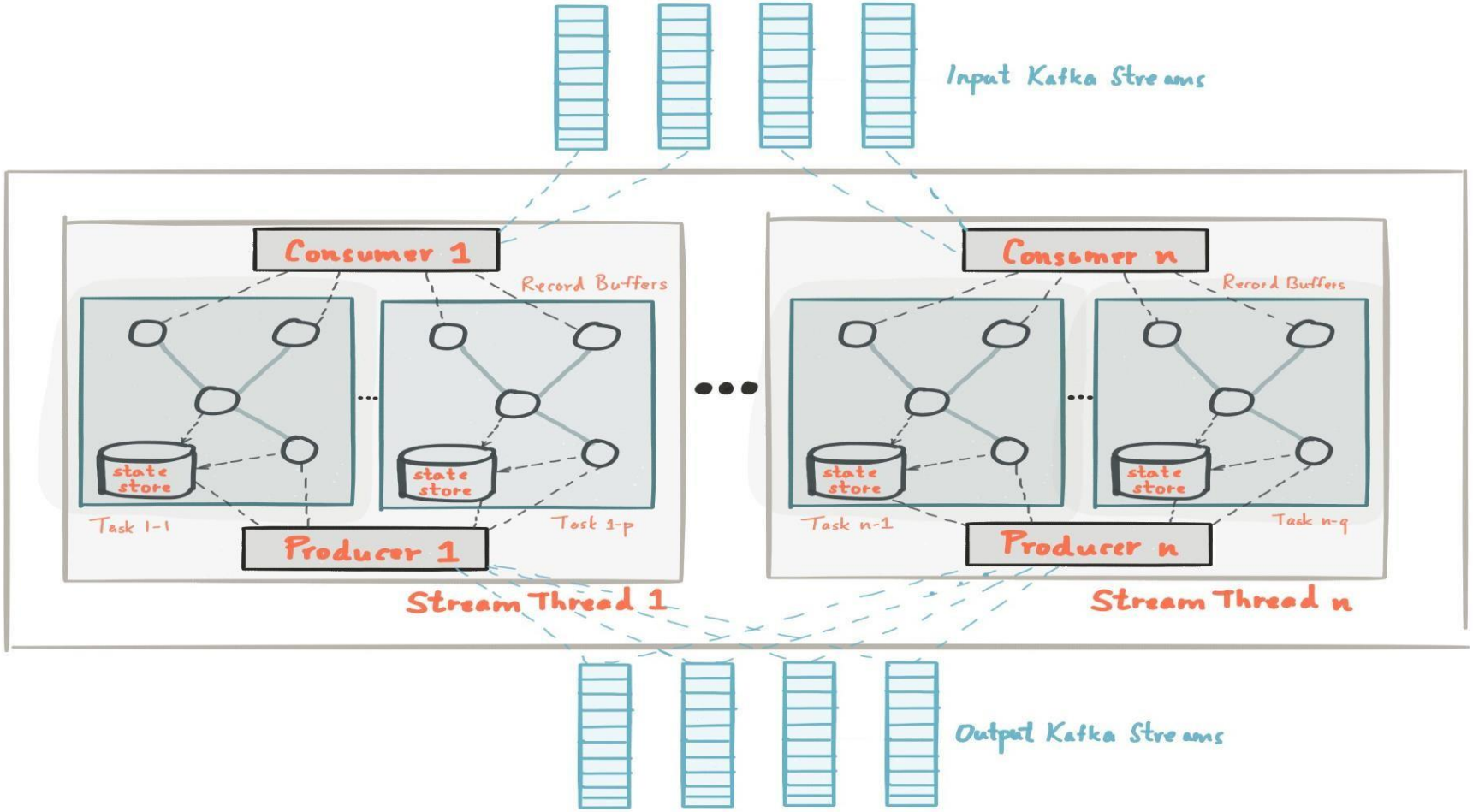
```
[2019-08-29 15:17:20,663] INFO Kafka version: 2.3.0 (org.apache.kafka.common.utils.AppInfoParser)
[2019-08-29 15:17:20,663] INFO Kafka commitId: fc1aaa116b661c8a (org.apache.kafka.common.utils.AppInfoParser)
[2019-08-29 15:17:20,663] INFO Kafka startTimeMs: 1567059440657 (org.apache.kafka.common.utils.AppInfoParser)
[2019-08-29 15:17:20,665] INFO [KafkaServer id=53] started (kafka.server.KafkaServer)
```

5. 기능소개

세부 목차

5. 기능소개

5.1 kafka 기본 Architecture

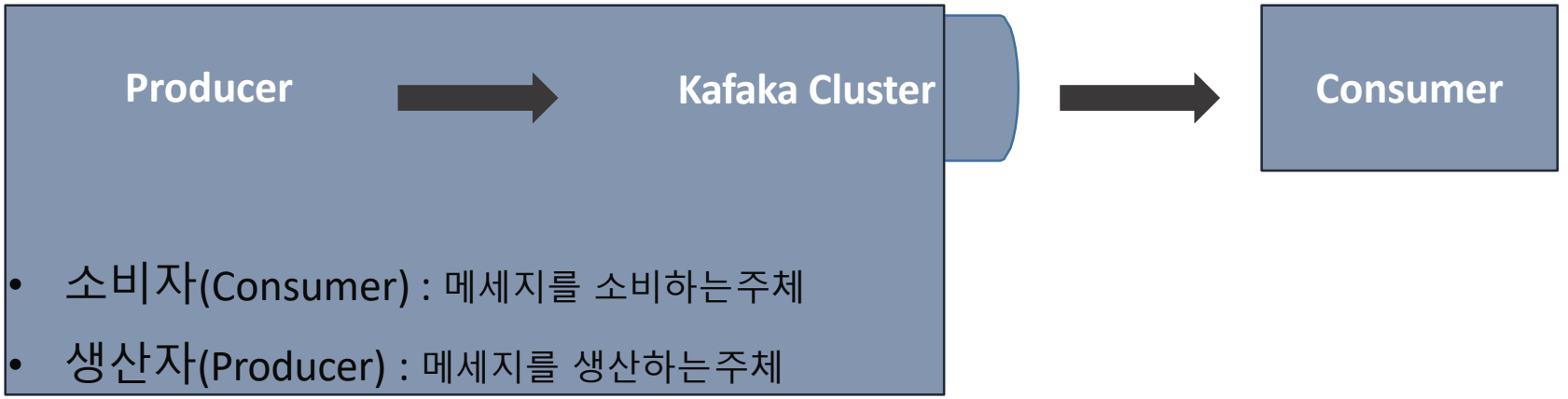


출처 : https://kafka.apache.org/22/documentation/streams/architecture

5. 기능소개

5.2 메세지(Message)

- 비동기 메시지 시스템
  - 메일을 보내면 메일 서버에 저장 메일을 받는 사람은 자신이 필요할 때 수신 하는 비동기 구조
  - 메일시스템과 동일 하게 Producer가 메시지를 보내면 Kafka Cluster에 저장 , Consumer가 필요시 메시지를 가져오는구조

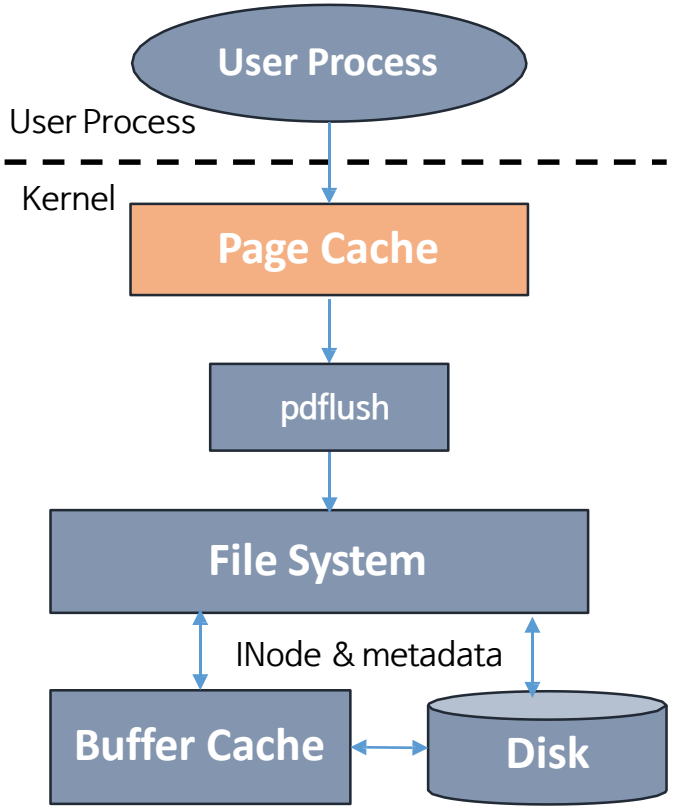


5. 기능소개

5.3 데이터 보관

- Disk-Based Retention(Persistent)
  - 다른 Message-system 과 다르게 파일시스템을 Message 주 저장소로사용
  - Memory에 별도의 Cache 없이 OS의 Page Cache 사용
  - Page Cache 로 read() 함수 호출 없이

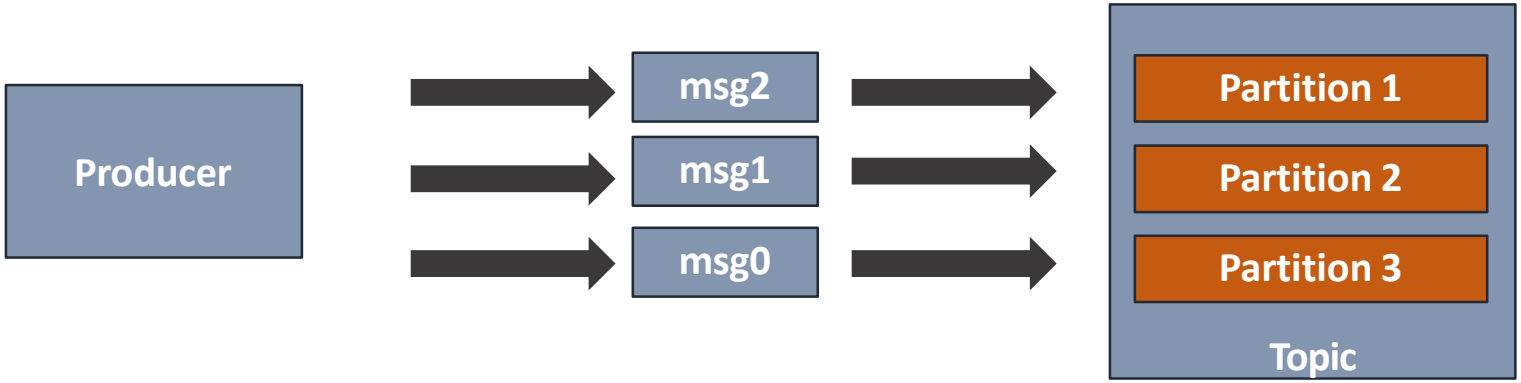
    직접 mapping된 주 메모리 참조




```
[root@kafka-01 ~]# vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 2966616   1052 3502016    0    0     1     5   10    3  0  0 99  0  0
[root@kafka-01 ~]# free -m
              total        used        free      shared  buff/cache   available
Mem:           7821        1503        2896          72        3420        5883
Swap:             0           0           0
```

- 성능 향상을 위한 Topic Partition 사용
  - Producer 가 Message 를 전송 할 때 1개의 Topic으로 Partitioning이 안되어 있을 경우 1개의 Message를 낼 때 1초가 걸린다면 3개의 Message를 보낼 때 3초가 소요

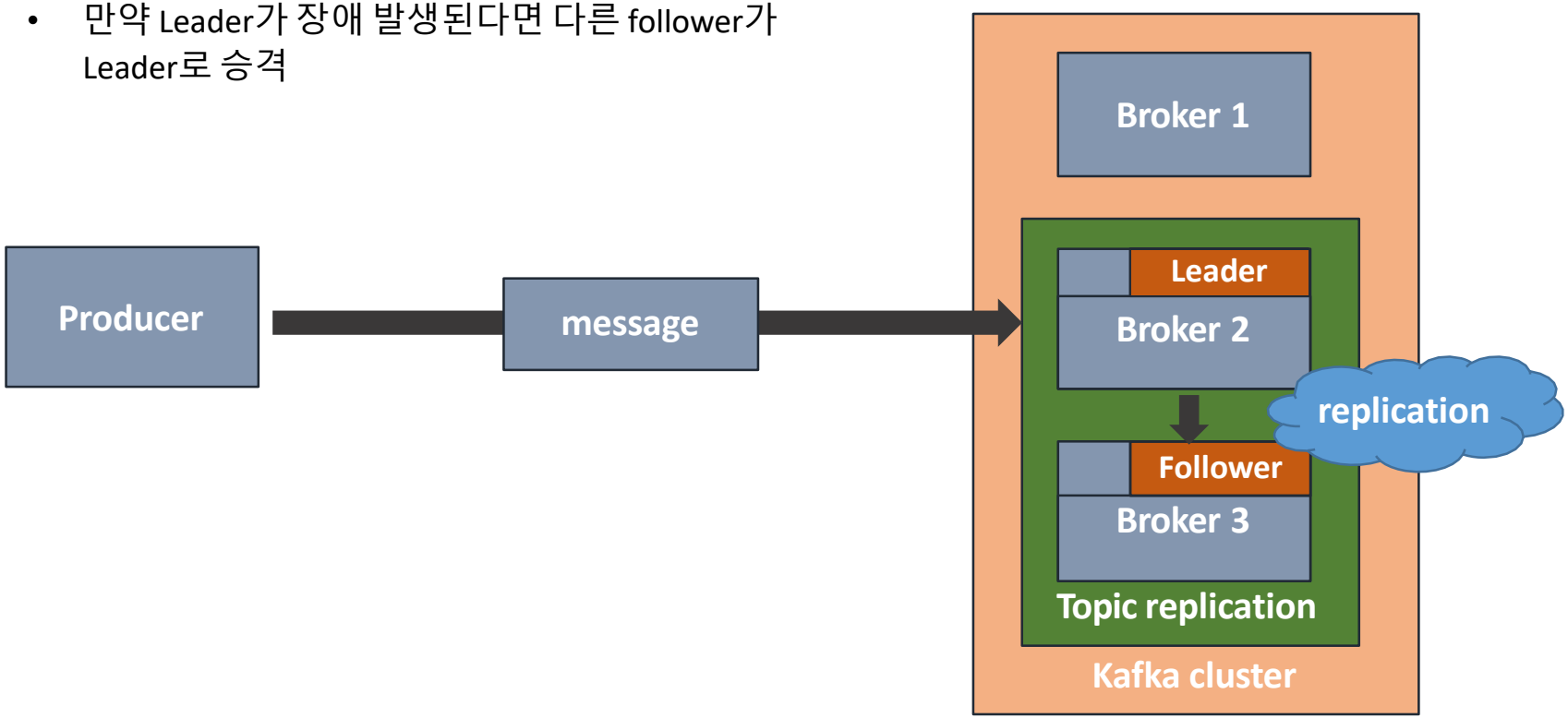
  - Partitioning 된다면 동일한 작업에 대하여 Partition 갯수에 비례 하여 처리 속도가 향상



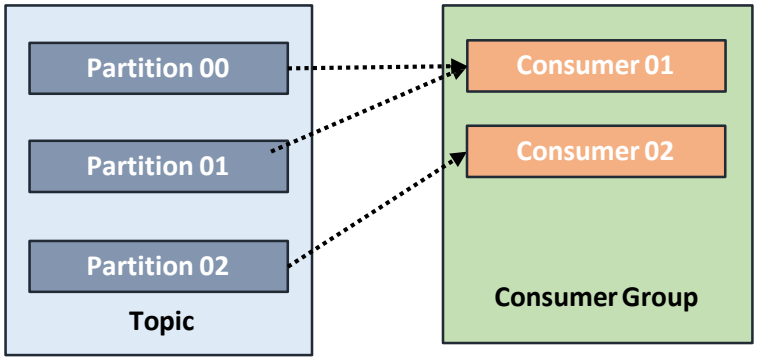
5. 기능소개

5.5 Topic Replication

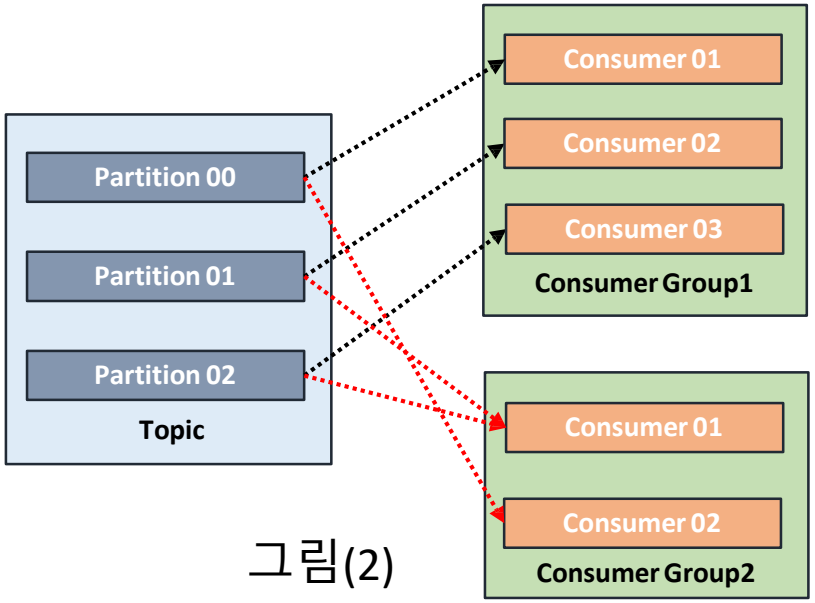
- 고가용성을 위하여 topic에 대한 replication(복제) 기능
  - Kafka cluster 는 Replication을 할 경우 broker들 간에 Leader를 구성
  - 구성 leader는 모든 Read/Write 가 발생
  - Leader는 follower라는 replication 대상으로 message Replication
  - 만약 Leader가 장애 발생된다면 다른 follower가 Leader로 승격



- 빠른 메세지 응답 및 데이터 손실 방지 기능
  - Kafka에서는 Cosumer Group 을 통하여 각각 Partition에 대하여 Consumer를 지정 하여 그룹화 함으로 빠른 속도를 보장 함과 함께 데이터 손실을 방지
  - Topic에 포함된 3개의 Partion에 대하여 하나의 Cosumer Group 에 2개의 consumer가 동작/분산하여 데이터를수신
  - 기존의 그림 (1)에서와 같이 하나의 Consumer Group 에서 처리 하던 정보를 동일하게 다른 쪽에서 사용하기 위해서 Consumer Group을 추가하여 활용 하는 것도 Kafka의 장점



그림(1)



그림(2)

6. 활용예제

세부 목차

1. Topic 생성
2. Topic 삭제
3. kafka 메세지 송수신
4. kafka – java api 를 활용한 메세지 송수신
5. kafka 모니터링

6. 활용예제

6.1 Topic 생성

- Data 요청을 위한 topic 생성 및 확인

1. test topic 생성

```
[kafka@ha-twx-svr-01.novalocal /data3/kafka_2.12-2.3.0]$ bin/kafka-topics.sh --create --zookeeper
192.168.10.153:2181 --replication-factor 1 --partitions 1 --topic test
Created topic test.
```

2. topic 생성 확인 (리스트 확인)

```
[kafka@ha-twx-svr-01.novalocal /data3/kafka_2.12-2.3.0]$ bin/kafka-topics.sh --list --zookeeper
192.168.10.153:2181
test
```

6. 활용예제

6.2 Topic 삭제

- topic 삭제 및 확인

1. topic 리스트 확인

```
[kafka@ha-twx-svr-01.novalocal /data3/kafka_2.12-2.3.0]$ bin/kafka-topics.sh --list --zookeeper
192.168.10.153:2181
test
```

2. test topic 삭제

```
[kafka@ha-twx-svr-01.novalocal /data3/kafka_2.12-2.3.0]$ bin/kafka-topics.sh --delete --zookeeper
192.168.10.153:2181 --topic test
Topic test is marked for deletion.
Note: This will have no impact if delete.topic.enable is not set to true.
```

3. topic 삭제 확인 (리스트 확인)

```
[kafka@ha-twx-svr-01.novalocal /data3/kafka_2.12-2.3.0]$ bin/kafka-topics.sh --list --zookeeper
192.168.10.153:2181
```

6. 활용예제

6.3 kafka 메세지 송수신

- kafka 테스트 Topic을 통한 메세지 송신 및 수신 확인

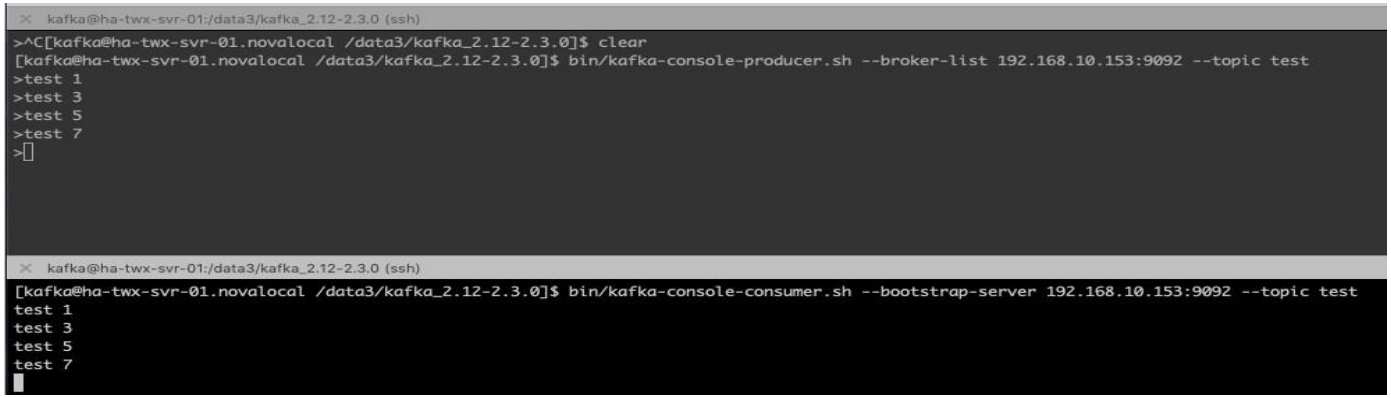
1. console-producer를 통한 메세지 송신

```
[kafka@ha-twx-svr-01.novalocal /data3/kafka_2.12-2.3.0]$ bin/kafka-console-producer.sh --broker-list
192.168.10.153:9092 --topic test
>test 1
```

2. console-consumer를 통한 메세지 수신 확인

```
[kafka@ha-twx-svr-01.novalocal /data3/kafka_2.12-2.3.0]$ bin/kafka-console-consumer.sh --bootstrap-server
192.168.10.153:9092 --topic test
test 1
```

3. 메세지 송수신 테스트 (리스트 확인)

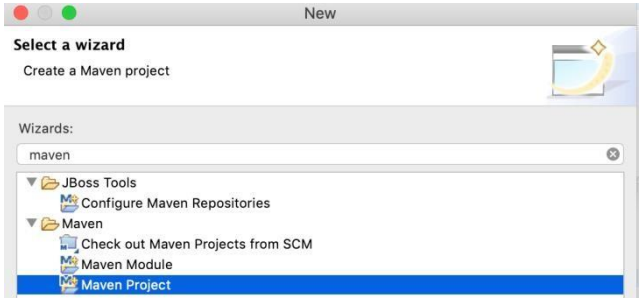


6. 활용예제

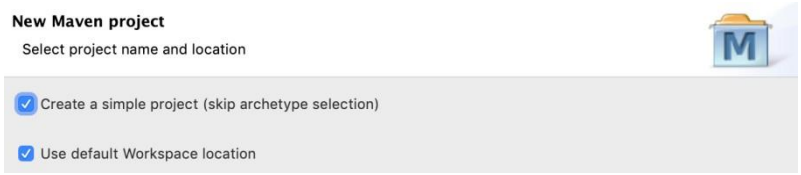
6.4 kafka – java api 를 활용한 메세지 송수신 (1/5)

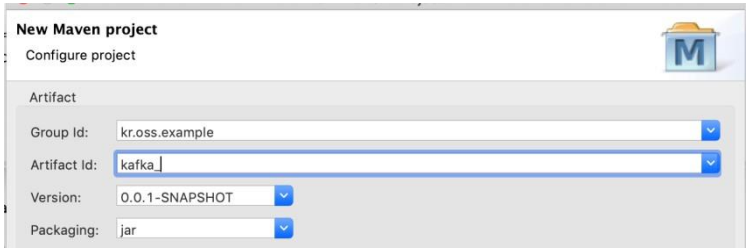
- Eclipse를 이용한 Maven Java Project 구성

1. maven project wizard 선택



2. maven project 생성





6. 활용예제

6.4 kafka – java api 를 활용한 메세지 송수신 (2/5)

- Eclipse를 이용한 Maven Java Project 구성

3. pom.xml 구성

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>kr.oss.example</groupId>
  <artifactId>kafka</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.apache.kafka</groupId>
      <artifactId>kafka_2.12</artifactId>
      <version>2.3.0</version>
    </dependency>
    <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-api</artifactId>
       <version>1.7.5</version>
    </dependency>
    <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-log4j12</artifactId>
       <version>1.7.5</version>
    </dependency>
  </dependencies>
</project>
```

6. 활용예제

6.4 kafka – java api 를 활용한 메세지 송수신 (3/5)

- Eclipse를 이용한 Maven Java Project 구성

4. ConsumerExample Class

```
package kr.oss.example.kafka.consumer;

import org.apache.kafka.clients.consumer.ConsumerRecord; i
mport org.apache.kafka.clients.consumer.ConsumerRecords; i
mport org.apache.kafka.clients.consumer.KafkaConsumer; im
port java.util.Arrays;
import java.util.Properties;

public class ConsumerExample {  publ
ic static void main(String[] args) {
      Properties props = new Properties();
      // 환경 변수 설정
      props.put("bootstrap.servers", "192.168.10.153:9092"); // kafka host 및 server 설정
      props.put("session.timeout.ms", "10000");           // session 설정   props.put("group
      .id", "test");                                      // topic 설정
      props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer"); // key deserializer  prop
      s.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer"); // value deserializer

      KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(props); // consumer 생성
      consumer.subscribe(Arrays.asList("test"));      // topic 설정

      while (true) { // 계속 loop를 돌면서 producer의 message를 띄운다.
         ConsumerRecords<String, String> records = consumer.poll(500);  for
         (ConsumerRecord<String, String> record : records) {
            String s = record.topic();
            if ("test".equals(s)) {
               System.out.println(record.value());
            } else {
               throw new IllegalStateException("get message on topic " + record.topic());
            }
         }
      }
   }
}
```

6. 활용예제

6.4 kafka – java api 를 활용한 메세지 송수신 (4/5)

- Eclipse를 이용한 Maven Java Project 구성

5. ProducerExample Class

```
package kr.oss.example.kafka.producer;

import java.util.Map; impo
rt java.util.Properties;

import org.apache.kafka.clients.producer.KafkaProducer; i
mport org.apache.kafka.clients.producer.ProducerRecord;

public class ProducerExample {

   public static void main(String[] args) throws Exception {

      Properties props = new Properties();
      // kafka host 및 server 설정
      props.put("bootstrap.servers", "192.168.10.153:9092,192.168.10.154:9092,192.168.10.155:9092");
      props.put("acks", "all");                          // 카프카로부터 확인을 기다리지 않습니다. props.put
      ("block.on.buffer.full", "true");                  // 버퍼링 할 때 사용할 수 있는 전체 메모리의 사용여부
      props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer"); // serialize 설정  pr
      ops.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer"); // serialize 설정

      // producer 생성
      KafkaProducer<String, String> producer = new KafkaProducer<String, String>(props);

      // message 전달
      for (int i = 0; i < 5; i++) {
         String v = "hello-"+i;
         producer.send(new ProducerRecord<String, String>("test", v));
      }

      producer.flush();
      producer.close();
   }
}
```

6. 활용예제

6.4 kafka – java api 를 활용한 메세지 송수신 (5/5)

- Eclipse를 이용한 Maven Java Project 구성

6. 송수신 테스트

1. ConsumerExample 클래스 실행

```
Console
ConsumerExample [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_171.jdk/Contents/Home/bin/java (Aug 30, 2019, 1:26:25 PM)
2019-08-30 13:26:25,815 DEBUG Metrics - Added sensor with name fetch-latency
2019-08-30 13:26:25,815 DEBUG Metrics - Added sensor with name records-lag
2019-08-30 13:26:25,815 DEBUG Metrics - Added sensor with name records-lead
2019-08-30 13:26:25,820  INFO AppInfoParser - Kafka version: 2.3.0
2019-08-30 13:26:25,820  INFO AppInfoParser - Kafka commitId: fc1aaa116b661c8a
2019-08-30 13:26:25,820  INFO AppInfoParser - Kafka startTimeMs: 1567139185818
2019-08-30 13:26:25,822 DEBUG KafkaConsumer - [Consumer clientId=consumer-1, groupId=test] Kafka consumer initialized
2019-08-30 13:26:25,822  INFO KafkaConsumer - [Consumer clientId=consumer-1, groupId=test] Subscribed to topic(s): test
```

6.2 ProducerExample 클래스 실행

```
Console
ProducerExample (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_171.jdk/Contents/Home/bin/java (Aug 30, 2019, 1:32
2019-08-30 13:32:34,398  WARN ProducerConfig - The configuration 'block.on.buffer.full' was supplied
2019-08-30 13:32:34,398 DEBUG Sender - [Producer clientId=producer-1] Starting Kafka producer I/O th
2019-08-30 13:32:34,399 DEBUG NetworkClient - [Producer clientId=producer-1] Initialize connection t
2019-08-30 13:32:34,400 DEBUG NetworkClient - [Producer clientId=producer-1] Initiating connection t
2019-08-30 13:32:34,400  INFO AppInfoParser - Kafka version: 2.3.0
2019-08-30 13:32:34,400  INFO AppInfoParser - Kafka commitId: fc1aaa116b661c8a
2019-08-30 13:32:34,400  INFO AppInfoParser - Kafka startTimeMs: 1567139554398
2019-08-30 13:32:34,403 DEBUG KafkaProducer - [Producer clientId=producer-1] Kafka producer started
```

6.3 결과 확인

```
log4j:WARN No appenders could be found for logger (org.apache.kafka.clients.consumer.ConsumerConfig).
log4j:WARN Please initialize the log4j system properly.
hello0
hello1
hello2
hello3
hello4
```

6. 활용예제

6.5 kafka 모니터링

- kafka에는 JMX 인터페이스의 MBean(Java Managed Bean)을 통해서 모니터링이 가능

1. kafka Offset 이라는 툴을 통한 모니터링 (http://quantifind.github.io/KafkaOffsetMonitor/)

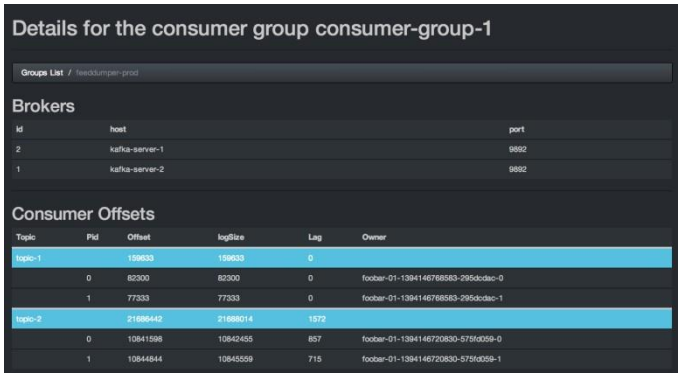
   1. 파일 다운로드

   wget https://github.com/quantifind/KafkaOffsetMonitor/releases/download/v0.2.1/KafkaOffsetMonitor-assembly-0.2.1.jar

   2. 구동

```
java -cp KafkaOffsetMonitor-assembly-0.2.1.jar \ c
om.quantifind.kafka.offsetapp.OffsetGetterWeb \
--zk 192.168.10.153 \
--port 8080 \
--refresh 10.seconds \
--retain 2.days
```

1.3. Brower 접속 (http://install_host:8080)



7. FAQ

Q kafka를 통해서 로그를 수집할 수 있나요?

A kafka는 자체적으로는 로그 수집을 할 수 있지는 않습니다.
다른 Producer 솔루션(filebeat, metricbeat 등)과 연계를 하여 데이터를 보관 및 전달를 할 수 있습니다.

Q kafka는 한대의 서버에 Cluster 구성이 가능한가요?

A 동일한 물리적인 서버에 kafka 서버를 여러대 구성한 후 Cluster로 구성해서 사용이 가능합니다.

하지만, 시스템 페이지 Cache 등 성능 향상을 위해 별도의 서버들로 구성하는 것이 좋습니다.

7. FAQ

Q Apache Kafka와 Apache Storm의 차이점은 무엇입니까?

&

A Apache Kafka는 많은 양의 데이터를 처리 할 수 있는 분산환경 메시징시스템으로 한쪽 끝에서 다른 쪽 끝으로 메시지를 전달합니다.

Apache Storm은 필요한 실시간 조작 및 데이터 계산을 수행하는 분산 실시간 컴퓨팅 시스템입니다.

즉, Apache Kafka를 통해서 데이터를 전달, 보관을 하면서 Apache Storm을 통해서 데이터의 분석/계산/조작을 수행한다고 보시면 됩니다.

8. 용어정리

용어	설 명
Broker	Kafka를 구성하는 서버 consumer와 producer를 연결하는 중개자 역활
Topic	Data가 저장되는 곳 producer (생산자) 가 발생한 메세지들의 보관 장소
Producer	생산자로써 메세지를 생산 중계자(Broker)에게 데이터를 전달
Consumer	소비자로써 메세지를 소비 중계자(Broker)가 가지고 있는 topic에서 메세지를 수신
Consumer-Group	메세지 소비자 묶음 단위 (n consumers)
Zookeeper	kafka를 안정적으로 운영하기 위한 Coorination Service
Partition	Topic이 복사(replicated)되어 나뉘어지는 단위

Open Source Software Installation & Application Guide