Introducing KRaft: Kafka without Zookeeper

Lee Dongjin | dongjin@apache.org

오늘 할 이야기는… (1)

- “KRaft란 무엇이며, 왜 필요하며, 어떻게 사용할 수 있는가?”
  - “Zookeeper 없이 동작하는 Apache Kafka”

오늘 할 이야기는… (2)

- 배경
  - 필요성
- 개념 및 동작 방식
- 사용법
  - Bridge Release

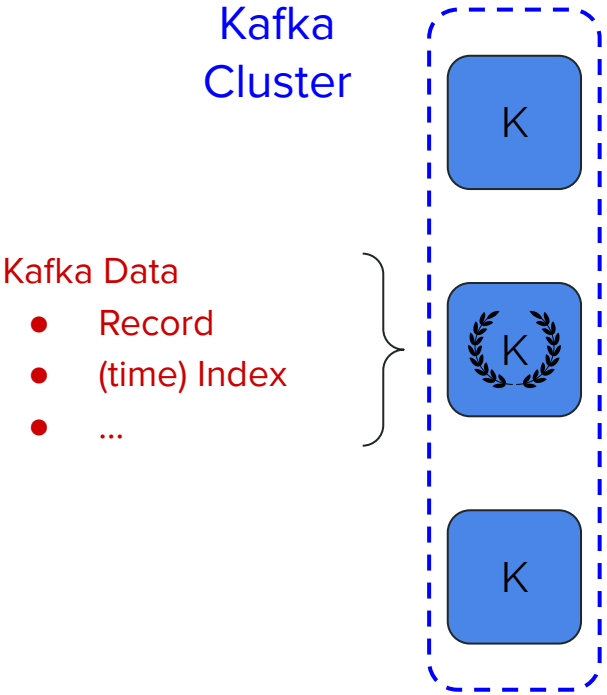
Introducing Myself …

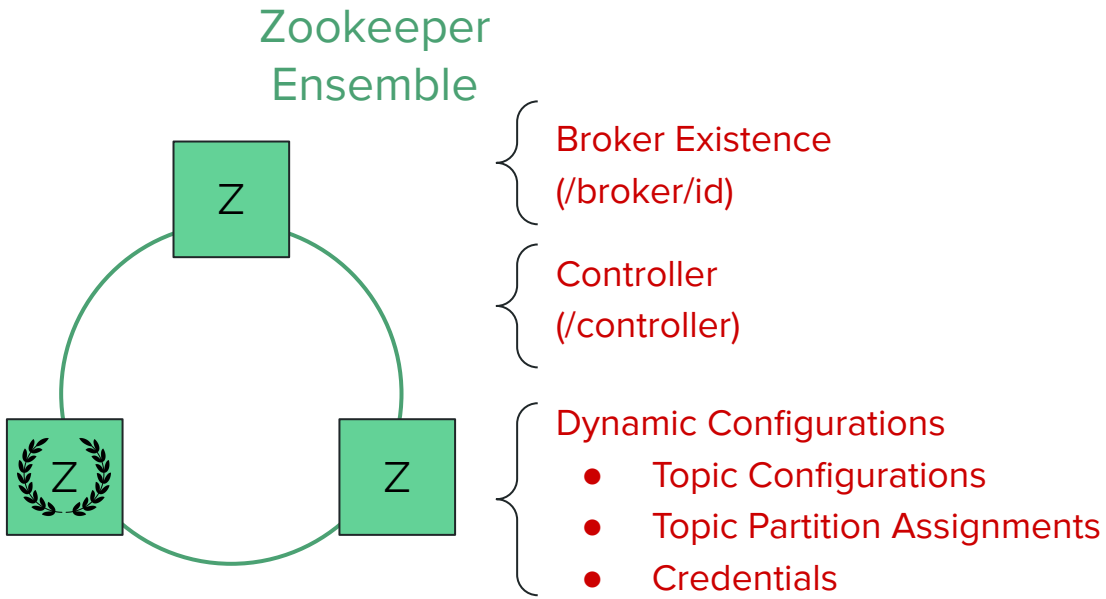
- Committer, Apache Software Foundation
- Apache Kafka Contributor
  - Compression
  - Log4j2 Support
  - etc…
- Apache Kafka @ Naver

배경: Apache Kafka에서의 Zookeeper (1)

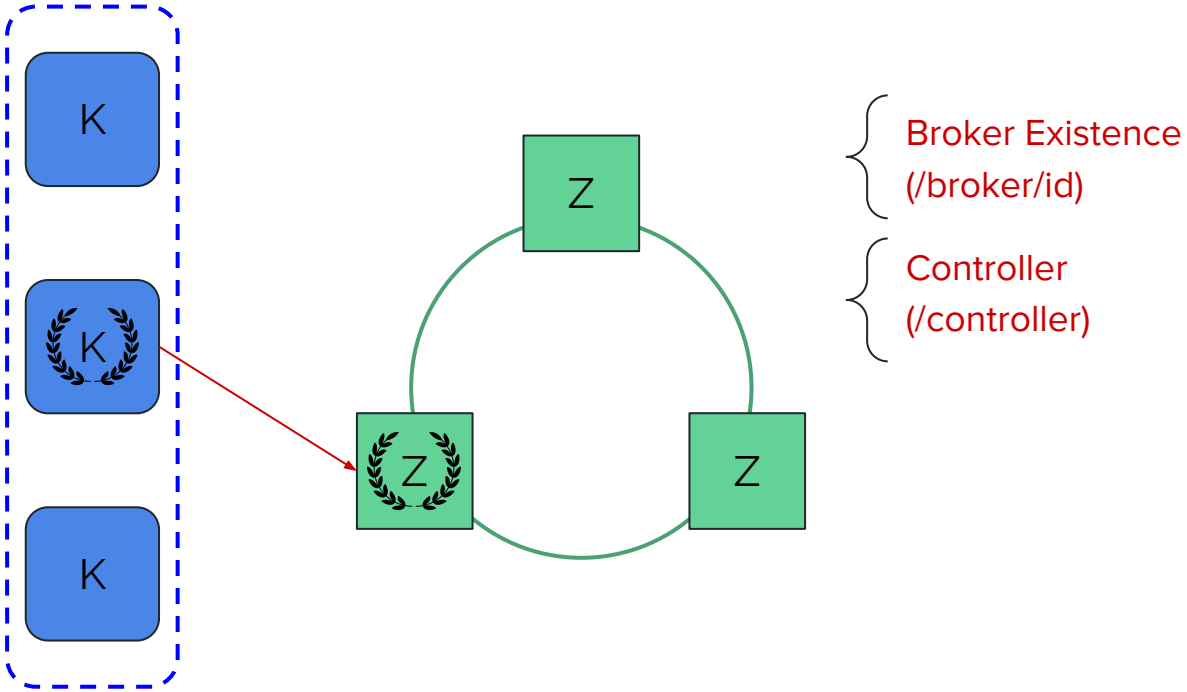
- Zookeeper Ensemble
  - 동적 metadata
- Kafka Cluster
  - Record Data
- Kafka 설치의 전제 조건

배경: Apache Kafka에서의 Zookeeper (2)

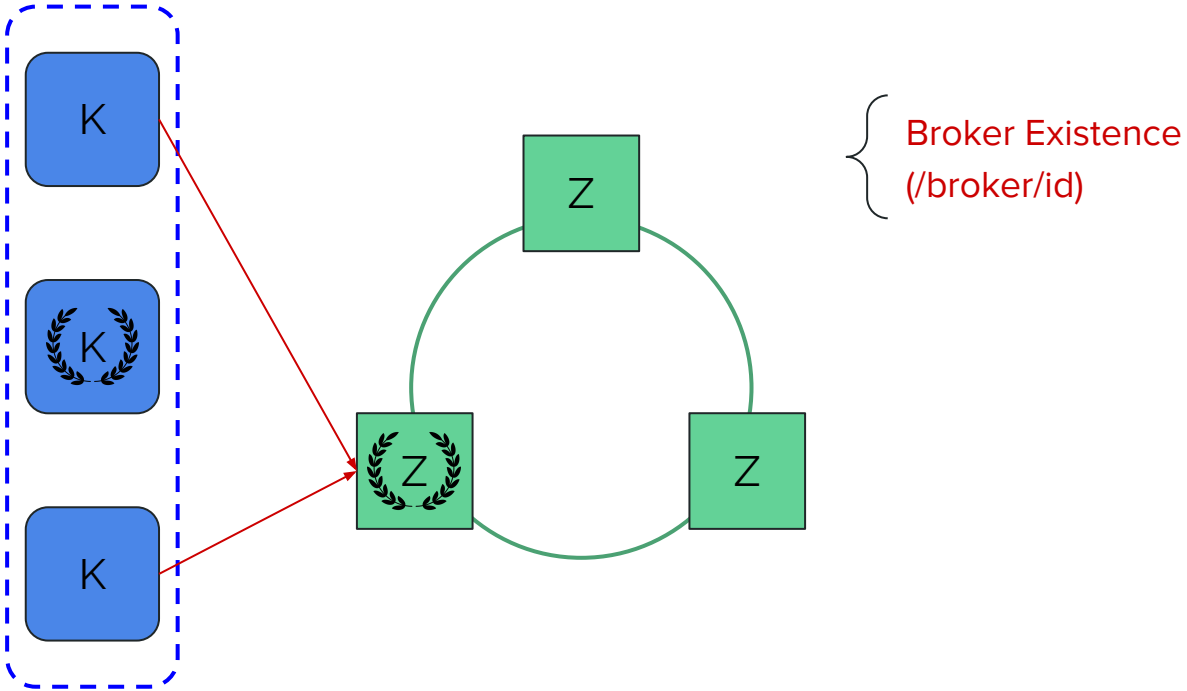




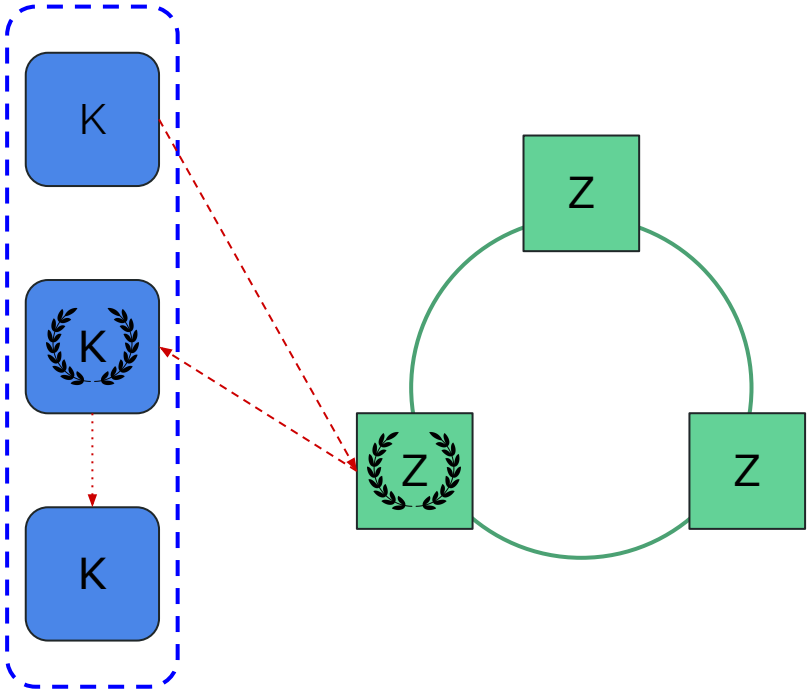
배경: Apache Kafka에서의 Zookeeper (3)



배경: Apache Kafka에서의 Zookeeper (4)



배경: Apache Kafka에서의 Zookeeper (5)



Dynamic Configurations

- Topic Configurations
- Topic Partition Assignments
- Credentials

문제

- Inconsistency
  - Zookeeper vs. Controller vs. Broker
- Performance
  - Controller Change
    - **Freezing**
  - Who becomes Controller?
    - **RANDOM**
- Cost
  - Zookeeper Instance?
  - Maintenance?
- etc
  - Scalability
  - A ➔ B vs. B ➔ A
  - A ➔ B ➔ C vs. A ➔ C

KRaft: Kafka without Zookeeper (1) – 개요

- Zookeeper를 사용하지 않음
  - Kafka Proces에 metadata를 저장
- Push 방식 (X) Pull 방식 (O)
  - Controller가 metadata 변경을 인지하고 control message를 보냄 (X)
  - Metadata가 구독 가능한 log의 형식으로 저장 (O)
    - Broker는 metadata 토픽을 구독해 오는 방식으로 동작
- 대격변
  - 용어, 아키텍처 전체가 다 바뀜

KRaft: Kafka without Zookeeper (2) - Controller

- Pre-KRaft
  - Broker 중 하나
- Post-KRaft
  - Metadata를 저장하는 역할을 맡은 Kafka Process
    - Broker
    - Controller
      - Active Controller
  - RAFT 알고리즘 사용
    - 조금 다름

KRaft: Kafka without Zookeeper (3) - Log

- Pre-KRaft
  - Kafka Broker에 저장된 record 뭉치
  - Metadata를 저장한 Log는 Zookeeper 내부에 감춰져서 보이지 않음
    - 구독 불가
- Post-KRaft
  - Broker Log
  - Controller Log

KRaft: Kafka without Zookeeper (4) - Log

	Broker Log	Controller Log
목적	Record 저장	Metadata 저장
위치	Broker	Controller
파티셔닝	1개 이상	1개
복제	Follower Replica (Broker)	Follower Controller

쓰기	• Asynchronous
	  ○ Page Cache
		• Synchronous
		  ○ fsync

읽기	• min.insync.replica 이상 Replication이 완료된 이후 • Leader Broker	• 과반 이상의 Controller에 Replication이 완료된 이후 • Active Controller

KRaft: Kafka without Zookeeper (5) - topic (1)

- Pre-KRaft
  - Internal Topics
    - __consumer_offsets, __transaction_state
  - Topics
    - Reserved Topics
      - _schema, connect-config, connect-status, ⋯

KRaft: Kafka without Zookeeper (5) - topic (2)

- Post-KRaft
  - Metadata topic
    - __cluster_metadata
  - Internal Topics
    - __consumer_offsets, __transaction_state
  - Topics
    - Reserved Topics
      - _schema, connect-config, connect-status, ⋯

사용법 (1)

- 클러스터 uuid 생성
  - bin/kafka-storage.sh random-uuid
- 로그 저장 공간 포맷
  - bin/kafka-storage.sh format -t {cluster-uuid} -c ./config/kraft/server.properties
- Pre-KRaft에서는 자동으로 수행되던 작업들

사용법 (2)

```
# The id of the broker. This must be set to a unique integer for each broker.

broker.id=0

# Zookeeper connection string (see zookeeper docs for details).

zookeeper.connect=localhost:2181

# listener configuration

listeners=PLAINTEXT://:9092
```

Pre-KRaft 설정

```
# The role of this server. Setting this puts us in KRaft mode

process.roles=broker,controller

# The node id associated with this instance's roles

node.id=1

# The connect string for the controller quorum

controller.quorum.voters=1@localhost:9093

# listener configuration

listeners=PLAINTEXT://:9092,CONTROLLER://:9093
```

Post-KRaft 설정

Bridge Release

- Zookeeper를 사용하던 Kafka Cluster를 KRaft 기반으로 이전시키기 위한 릴리즈
  - 하방 호환성
  - Post-KRaft Controller를 시동 과정에서 Zookeeper에 저장된 메타데이터를 전부 긁어옴
    - /controller 노드 봉쇄
  - Pre-Kraft Broker
    - /broker/id 에 등록
    - Active Controller가 제어 신호를 보내 줌
  - Post-KRaft Broker
    - Post-KRaft 모드로 동작
- 3.1.0 현재 작업중
  - KRaft 기능 자체가 experimental 한 기능

정리

- Zookeeper 기반 Apache Kafka
  - Zookeeper가 metadata를 제어
  - ‘옥상옥(屋上屋)’ 구조
  - 성능 문제
- Raft 기반 Apache Kafka
  - Kafka가 직접 metadata를 제어
- 사용법
  - Bridge Release

질문?