



CONFLUENT

Confluent Platform 6.1 & Apache Kafka 2.7 New Features & Roadmap

Oracle CDC Source Connector Deep Dive

Feb. 24. 2021

HyunSoo Kim(hkim@confluent.io)

Senior Solutions Engineer



Confluent Platform 6.1 & Apache Kafka 2.7 New Features & Roadmap

Confluent 소개

The Rise of Event Streaming



Kafka & Confluent



Confluent는 이벤트 스트리밍을 개척했습니다

Kafka의 창시자가 Confluent를 설립

Kafka 소프트웨어 Commit의 80% 이상을 수행하고 있으며, Kafka에 대해 1백만 시간 이상의 기술 경험을 보유하고 있으며 5,000 개 이상의 클러스터를 운영

Confluent Platform은 Apache Kafka를 완성한 안전한 엔터프라이즈용 플랫폼

Confluent Cloud는 전 세계 유일의 멀티 클라우드, 완전 관리형 및 종량제 기반 이벤트 스트리밍 서비스

AWARDS



Forbes

Partner
of the Year
Google Cloud



Morgan Stanley



CTO Innovation
Award Winner

2019

JPMORGAN CHASE & CO.
Hall of Innovation

BANK OF AMERICA

Enterprise Technology
Innovation

Confluent Platform 6.1

New Features



Confluent Platform 6.1

최신 Apache Kafka 2.7 기반

2021년 2월 9일에 GA

Confluent Platform 6.1



DEVELOPER

개발자 생산성 극대화

다양한 언어를 통한 개발
Non-Java Clients | REST Proxy
Admin REST APIs

풍부한 Pre-built 에코시스템
Connectors | Hub | Schema Registry

이벤트 스트리밍 데이터베이스
ksqlDB

OPERATOR

대규모 운영시 효율적인 운영 기능

GUI-기반 관리 및 모니터링
Control Center

유연한 DevOps 자동화
Operator | Ansible

동적인 성능 및 유연성
Self-Balancing Clusters | Tiered Storage

ARCHITECT

운영 환경에 필수적인 기능

엔터프라이즈급 보안
RBAC | Secrets | Audit Logs

데이터 호환성
Schema Registry | Schema Validation

글로벌 탄력성
Multi-Region Clusters | Replicator
Cluster Linking

⌘ Apache Kafka

IMPROVED IN CP 6.1



Self-managed Software



Fully Managed Cloud Service



Enterprise
Support



Professional
Services

선택의 자유

커미터 중심의 전문성



Training



Partners



CP 6.1은 Kafka가 원활하게 실행되도록 더 간단한 작업으로 더 높은 가용성을 제공합니다



자동화 및 지속적 배포를 사용하여 다운 타임 감소

- MRC Automatic Observer Promotion
- ksqlDB Query Upgrades



운영을 단순화하고 능률적인 사용자 경험 제공

- MRC Automatic Observer Promotion
- C3 + Central Cluster Registry integration
- ksqlDB Multi-key Pull Queries



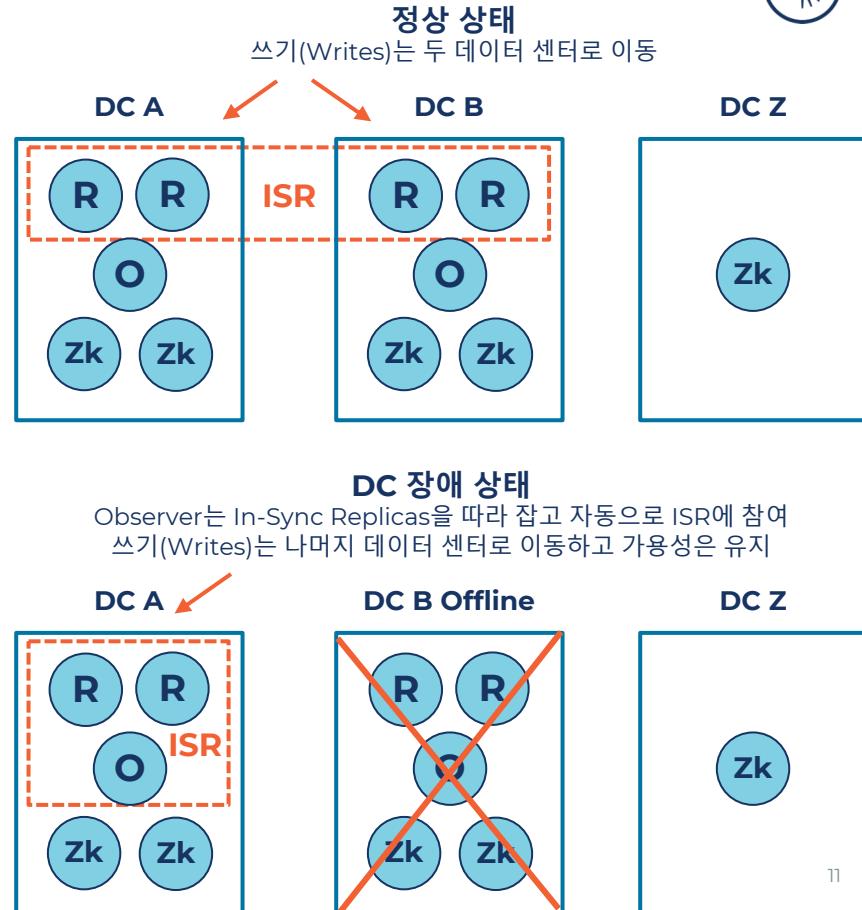
중앙 집중식 관리로 가시성 및 제어 향상

- C3 + Central Cluster Registry integration
- Self-Balancing Clusters Status API



2.5 Datacenters ; 4 Replicas ("R") + 2 Observers ("O")
Min ISR: 3 ; acks=all

Confluent는 Multi-Region Clusters에 대한 Automatic Observer Promotion을 통해 Kafka의 안정성을 향상시킵니다





Leader-Follower-Observer(Topic Partition)

```
kafka-topics --bootstrap-server localhost:9092 --describe
```

```
Topic: test-observers PartitionCount: 3 ReplicationFactor: 5 Configs: segment.bytes=1073741824,confluent.placement.constraints={"version":1,"replicas":[{"count":3,"constraints":{"rack":"us-west"}]},{"observers":[{"count":2,"constraints":{"rack":"us-east"}]}}
Topic: test-observers Partition: 0 Leader: 1 Replicas: 1,2,3,4,5 Isr: 1,2,3 Offline: Observers: 4,5
Topic: test-observers Partition: 1 Leader: 2 Replicas: 2,3,1,5,4 Isr: 2,3,1 Offline: Observers: 5,4
Topic: test-observers Partition: 2 Leader: 3 Replicas: 3,1,2,4,5 Isr: 3,1,2 Offline: Observers: 4,5
```



Why add MRC Automatic Observer Promotion?

운영자는 더 적은 수작업으로 자체 관리되는 Multi-Datacenter Kafka에 대해 더 높은 가용성을 달성

데이터 센터에 장애가 발생하거나 Broker가 오프라인 상태가 되면, 이제 Observer Replicas를 자동으로 승격하여 가용성에 미치는 영향을 최소화하면서 클러스터를 계속 실행할 수 있습니다.

더 간단한 구성으로 더 쉽게 높은 SLA를 달성하고 운영 부담을 줄임

특정 Topic 또는 전체 클러스터에 적용할 수 있는 간단한 구성으로 활성화합니다.



**Confluent는
In-Place ksqlDB
Query Upgrades를
통해
Streaming
Applications 중단을
최소화 합니다.**

ksqlDB를 사용하면 **Application** 가용성을
저하시키지 않고 **Query**를 쉽게 업데이트

Original Stream

Customer	Order ID
Jay	1
Sue	2
Fred	3
...	...

Updated Stream

Customer	Order ID	Payment
Jay	1	\$10
Sue	2	\$15
Fred	3	\$5
...



다운 타임없이 간단한
ksqlDB Stream/Table
업그레이드



Why add ksqlDB Query Upgrades?

운영환경에 배포된 중요한
스트리밍 애플리케이션에 대한
중단 최소화

비즈니스 요구 사항이 변경됨에 따른 Schema
진화를 지원하기 위해 In-Place Query
업그레이드를 수행합니다.

전체 원래 Schema를 지정하지
않고 Stream 또는 Table의
Schema를 변경함

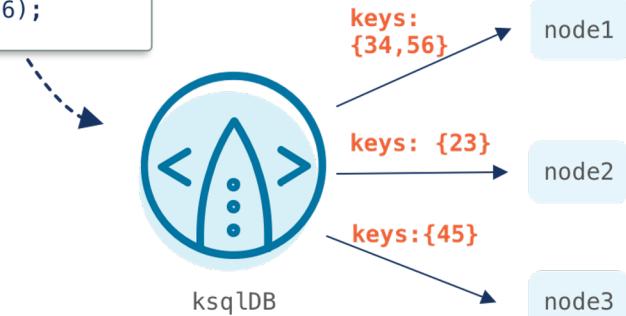
ALTER 구문을 사용하면 전체 Schema를 제공하지
않고도 하나 이상의 Columns을 추가할 수
있습니다.



Confluent는
ksqlDB Multiple Keys
Pull Queries를
사용하여 개발을
간소화합니다.

Single Key가 아닌 Multiple Keys에 대한 Pull Query

```
SELECT * FROM table  
WHERE key  
IN (23, 34, 45, 56);
```





Why add ksqlDB Multi-key Pull Queries?

Multiple Keys로 ksqlDB Table을 Query하여 개발자 생산성 향상

Materialized View의 상태를 가져 오기 위해 Single Key를 사용하는 검색으로 제한되지 않고 Multiple Keys 검색에 IN 을 사용하십시오.

사용 가능한 Query 패턴을 확장하여 ksqlDB 생태계를 강화

개발자는 원하는 Multiple Keys Pull Query 기능을 얻기 위해, 더이상 외부 저장소 / DB에서 Table을 생성할 필요가 없습니다.



Confluent는 C3를 Cluster Registry와 긴밀하게 통합하여 관리 작업을 간소화합니다

클러스터 이름을
Cluster Registry에
반영된 사용자
친화적인 이름으로
변경하십시오.

Cluster settings in C3

Change the cluster name

Cluster settings

General

Broker defaults

Self-balancing

Storage

Cluster name*

Kafka Raleigh

Cluster id

Km03YjMaT_mBJwG4IxwkaA



Hosts

SASL_SSL://kafka1:10091 SASL_SSL://kafka2:10092





Why add C3 + Cluster Registry integration?

클러스터 ID 정보를 기억할 필요가 없으므로 사용자 친화적인 사용자 경험을 제공

RBAC Role-Binding과 같은 일상적인 작업을 더 쉽게 만들기 위해 Kafka 환경 전체에서 사용할 수 있도록 친숙한 이름을 클러스터에 할당합니다.

클러스터 관리를 중앙 집중화하여 가시성과 제어를 개선하고 운영을 단순화

각 구성을 개별적으로 관리할 필요없이 중앙에서 여러 Kafka 클러스터에 대한 감사 로그 구성을 변경합니다.



Confluent의 Status API를 통해 Self-Balancing Cluster 작업을 추적하여 가시성을 높일 수 있습니다

Partition
Rebalancing의
진행 상황과
Broker 추가 또는
제거 상황을
쉽게 추적

Self-Balancing Status UI

Monitor the “add broker” status

The screenshot shows the Confluent Self-Balancing Status UI. On the left, there's a sidebar with clusters AL (Cluster 1), GR (Cluster 2), YO (Cluster 3), CL (Cluster 4), and Schemas. The main area shows a 'BROKERS OVERVIEW' titled 'Self-balancing'. It lists brokers with their IDs and statuses: broker.11 (Failed), broker.7 (In progress), broker.12 (In progress), broker.9 (Failed), broker.10 (Success), broker.6 (In progress), broker.8 (Failed), and broker.13 (Failed). Below this is a table of 'Add broker tasks' with columns for Broker ID, Status, Partition reassignment, and Broker shutdown. The table shows tasks for brokers 11, 7, 12, 9, 10, 6, 8, and 13.

Broker ID	Status	Partition reassignment	Broker shutdown
broker.11	Failed		
broker.7	In progress		
broker.12	In progress		
broker.9	Failed		Error found XYZ
broker.10	Success		
broker.6	In progress		
broker.8	Failed		Error found XYZ
broker.13	Failed		Error found XYZ



Why add Self-Balancing Clusters Status API?

어느 시점에서는 **Self-Balancing Clusters**가 수행하는 작업에 대한
가시성 향상

새로운 Status UI를 사용하면 Self-Balancing Clusters가 수행하는 작업을 알기 위해서 Metrics API를 활용할 필요가 없습니다.

Broker 추가 및 제거 상태를 확인하기 위한 API로 진행 상황을 쉽게 추적합니다.

운영자는 Partition Rebalancing이 성공적으로 진행되고 있는지, 수집된 Metrics으로 인해 속도가 느려지는지, 취소되었는지 또는 다시 시작되었는지 추적할 수 있습니다.

Apache Kafka 2.7

Highlight Features

Apache Kafka 2.7 Highlight Features



Confluent Platform 6.1은 Apache Kafka의 최신버전 2.7 기반입니다.

- **KIP-497: Add inter-broker API to alter ISR** - Controller에 Partition Leader 및 ISR의 상태를 업데이트하는 독점 기능을 제공하는 새로운 "AlterIsr" API가 추가되었습니다. 이 새로운 API의 가장 큰 장점은 메타 데이터 요청이 항상 최신 상태를 반영한다는 것입니다. 이 API의 추가는 ZooKeeper를 제거하고 KIP-500을 완료하는 과정에서 중요한 진전입니다.
- **KIP-595: Addition of Core Raft Implementation** - Core Consensus(핵심 합의) 프로토콜을 포함하는 별도의 "Raft"모듈이 추가되었습니다. Controller(Partitions 및 Replicas의 상태를 관리하는 Kafka 클러스터의 Broker)와의 통합이 완료될 때까지 Raft 구현의 성능을 테스트하는 데 사용할 수 있는 Standalone(독립형) 서버가 있습니다.
- **KIP-450: Sliding window aggregations in the DSL** - Kafka Streams는 Session Windows, Tumbling Windows 그리고 Hopping windows을 제공했습니다. **KIP-450를 통해 Sliding Windows를 추가하여, Kafka Streams는 이제 Sliding Aggregations을 수행하는 보다 효율적인 방법을 제공합니다.**

KIP-500: Replace ZooKeeper with a Self-Managed Metadata Quorum

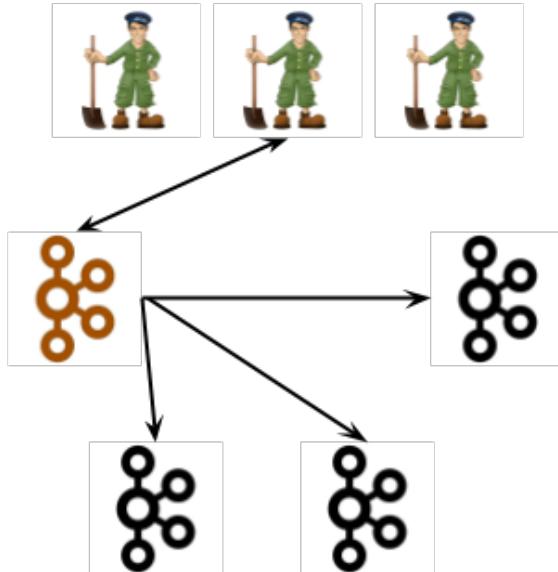
What is KIP?



KIP = Kafka Improvement Proposal

<https://cwiki.apache.org/confluence/display/KAFKA/Kafka+Improvement+Proposals>

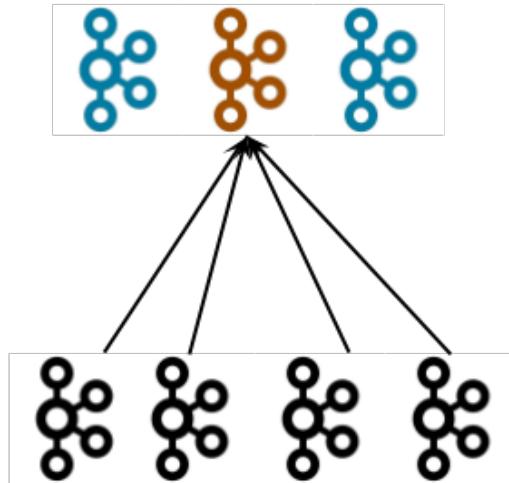
KIP-500: Replace ZooKeeper with a Self-Managed Metadata Quorum



- 실제로는 Controller 이외의 다른 Broker는 ZooKeeper와 통신할 수 있습니다. 따라서 실제로 각 Broker에서 ZooKeeper 까지 선을 그려야 합니다. 그러나 그렇게 많은 선을 그리면 다이어그램을 읽기가 어렵습니다.
- 이 다이어그램에서 표현되지 않은 또 다른 문제는 외부 Command Line 도구 및 유ти리티가 Controller의 개입없이 ZooKeeper의 상태를 수정할 수 있다는 것입니다.
- 이러한 문제로 인해 Controller의 메모리 상태가 실제로 ZooKeeper의 영구 상태를 반영하는지 여부를 알기가 어렵습니다.

Current

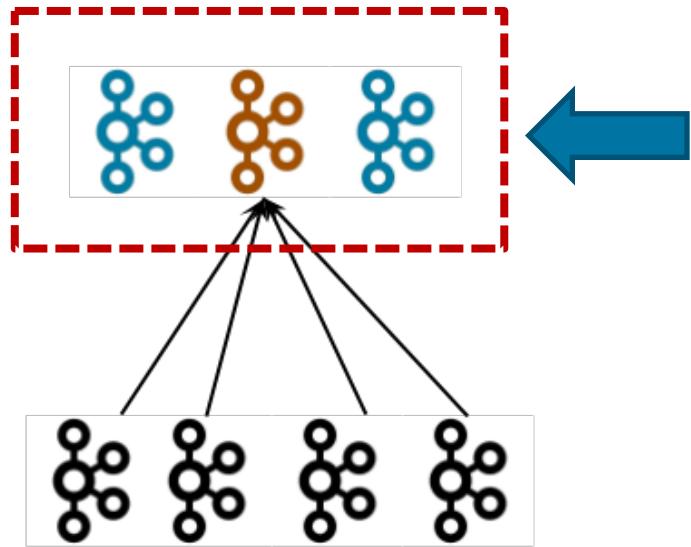
KIP-500: Replace ZooKeeper with a Self-Managed Metadata Quorum



- 제안된 아키텍처에서는 3 개의 Controller 노드가 3 개의 ZooKeeper 노드를 대체합니다.
- Controller 노드들은, 주황색으로 표시된, 메타 데이터 파티션의 단일 Leader를 선택합니다. Controller가 Broker에 업데이트를 푸시하는 대신 Broker는 이 Leader에서 메타 데이터 업데이트를 가져옵니다.
- Controller 프로세스는 Broker 프로세스와 논리적으로 분리되어 있지만 물리적으로 분리될 필요는 없습니다. 경우에 따라 Broker 프로세스와 동일한 노드에 Controller 프로세스의 일부 또는 전체를 배포하는 것이 합리적일 수도 있습니다.

Proposed

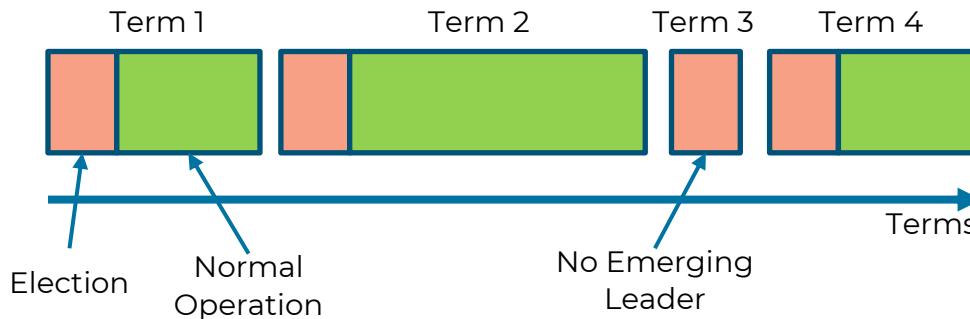
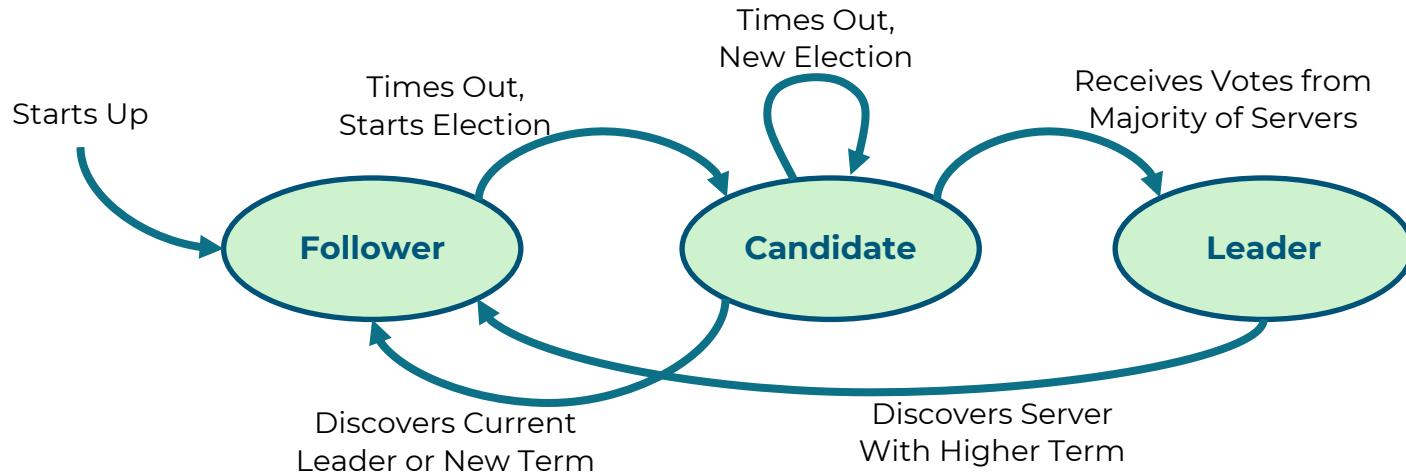
KIP-595: Addition of Core Raft Implementation



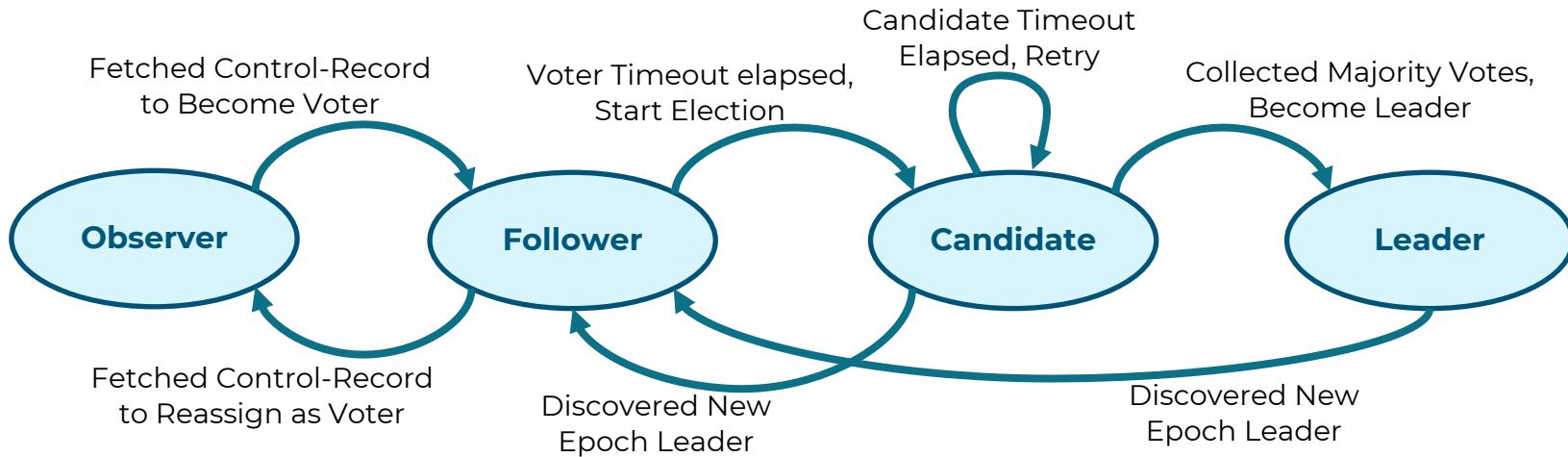
Controller 노드들에서 Leader를 선출하는데
사용되는 핵심 알고리즘

Proposed

Raft Algorithm – 노드들의 State 및 Term



KIP-595: Addition of Core Raft Implementation



quorum.voters configuration parameter

Index / Term 용어 대신에
Offset / Epoch 용어 사용



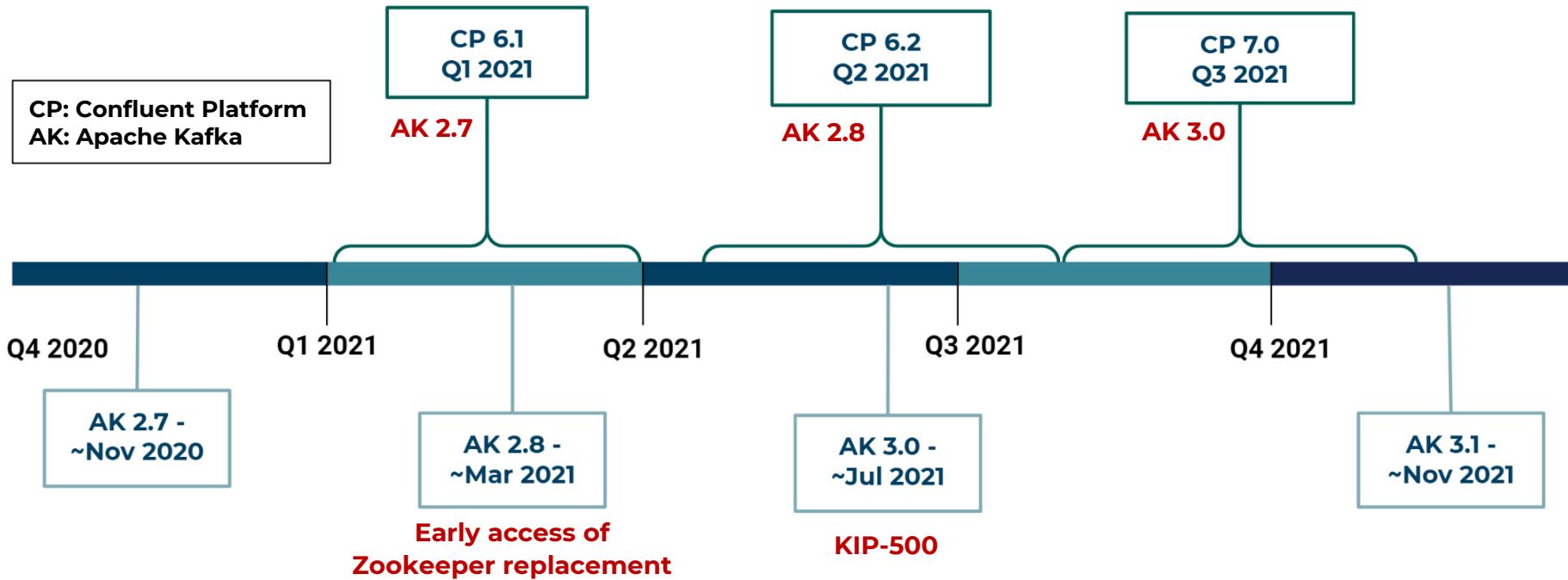
Leader-Follower-Observer(Metadata Quorum)

```
> bin/kafka-metadata-quorum.sh --describe
ClusterId:           SomeClusterId
LeaderId:            0
LeaderEpoch:         15
HighWatermark:       234130
MaxFollowerLag:     34
MaxFollowerLagTimeMs: 15
CurrentVoters:      [0, 1, 2]
```

```
> bin/kafka-metadata-quorum.sh --describe replication
ReplicaId  LogEndOffset   Lag    LagTimeMs  Status
0          234134        0      0          Leader
1          234130        4      10         Follower
2          234100        34     15         Follower
3          234124        10     12         Observer
4          234130        4      15         Observer
```

Confluent Platform & Apache Kafka Roadmap

Confluent Platform & Apache Kafka Roadmap



This subjects to change.



Oracle CDC Source Connector



Confluent Oracle CDC Source Connector 2021년 2월 16일에 GA



Premium Connectors for Confluent Platform

데이터 시스템 및 애플리케이션의 데이터를 Kafka 및 Confluent Platform에 빠르고 안정적으로 통합



차별화되고 비용 효율적인 대안

고비용의 레거시 데이터 벤더(예 : Oracle, IBM 등) 및 전문 솔루션 공급 업체의 비용이 많이 드는 가격 모델에서 고 가치 데이터를 비용 효율적으로 오프로드합니다.



복잡하고 가치가 높은 데이터 시스템과 빠르고 안정적으로 통합

사전 구축 및 인증된 Connector를 활용하여 복잡하고 가치 있는 데이터 시스템, 애플리케이션 및 기록 시스템을 Kafka에 원활하고 안정적으로 통합합니다.



엔지니어링 리소스 및 비용 절감

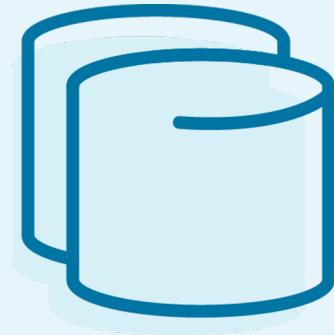
최고의 전문가가 구축한 Kafka-native Connector는 Kafka와 즉시 통합되어, 12 개월 이상의 매우 복잡한 통합을 설계, 구축, 테스트 및 유지하는데 드는 리소스를 줄일 수 있습니다.



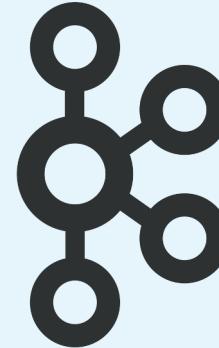


실시간 Use Case를 가능하게 하려면 기업은 레거시
시스템을 최신 기술에 연결해야 합니다

Databases

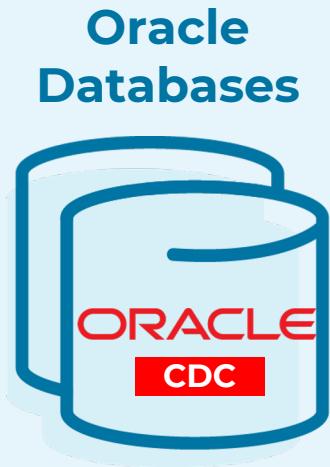


Kafka





Oracle DB는 Kafka에 연결하는 데 필요한 가장 일반적인 관계형 데이터베이스 중 하나입니다



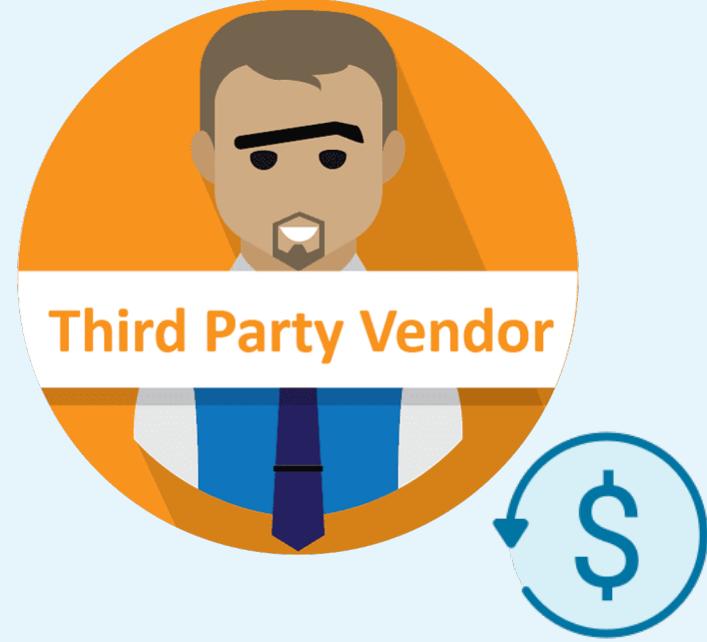
Kafka



레거시 시스템의 데이터를 잠금 해제하여 보다 혁신적인 실시간 고객 경험을 제공합니다



레거시 시스템을 Kafka에 연결할 때, 두 가지 접근 방식 중 하나를 사용합니다





Oracle CDC Source Premium Connector

Oracle Database에서 Confluent Platform / Kafka로 데이터를 비용 효율적이고 안정적으로 오프로드할 수 있습니다.

개발자가 각 Oracle 테이블의 매우 중요한 변경 이벤트를 별도의 Kafka Topic으로 캡처하여 최신 애플리케이션을 구축할 수 있습니다.



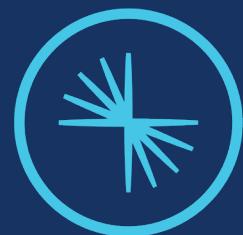
높은 가치의 Oracle DB 변경 이벤트를 캡처하는 데 드는 엄청난 라이선스 비용을 줄이거나 방지함(예 : Oracle GoldenGate)



최신 데이터 시스템과 레거시 데이터를 활용하여 실시간 Use Case 및 애플리케이션 (예 : RT 인벤토리, RT 사기 감지, DWH 현대화)을 가능하게 함



매우 복잡한 Oracle CDC 통합 및 연결을 설계, 구축, 테스트 및 유지 관리하여, 평균적으로 12 ~ 24 개월의 엔지니어링 비용 절감



CONFLUENT