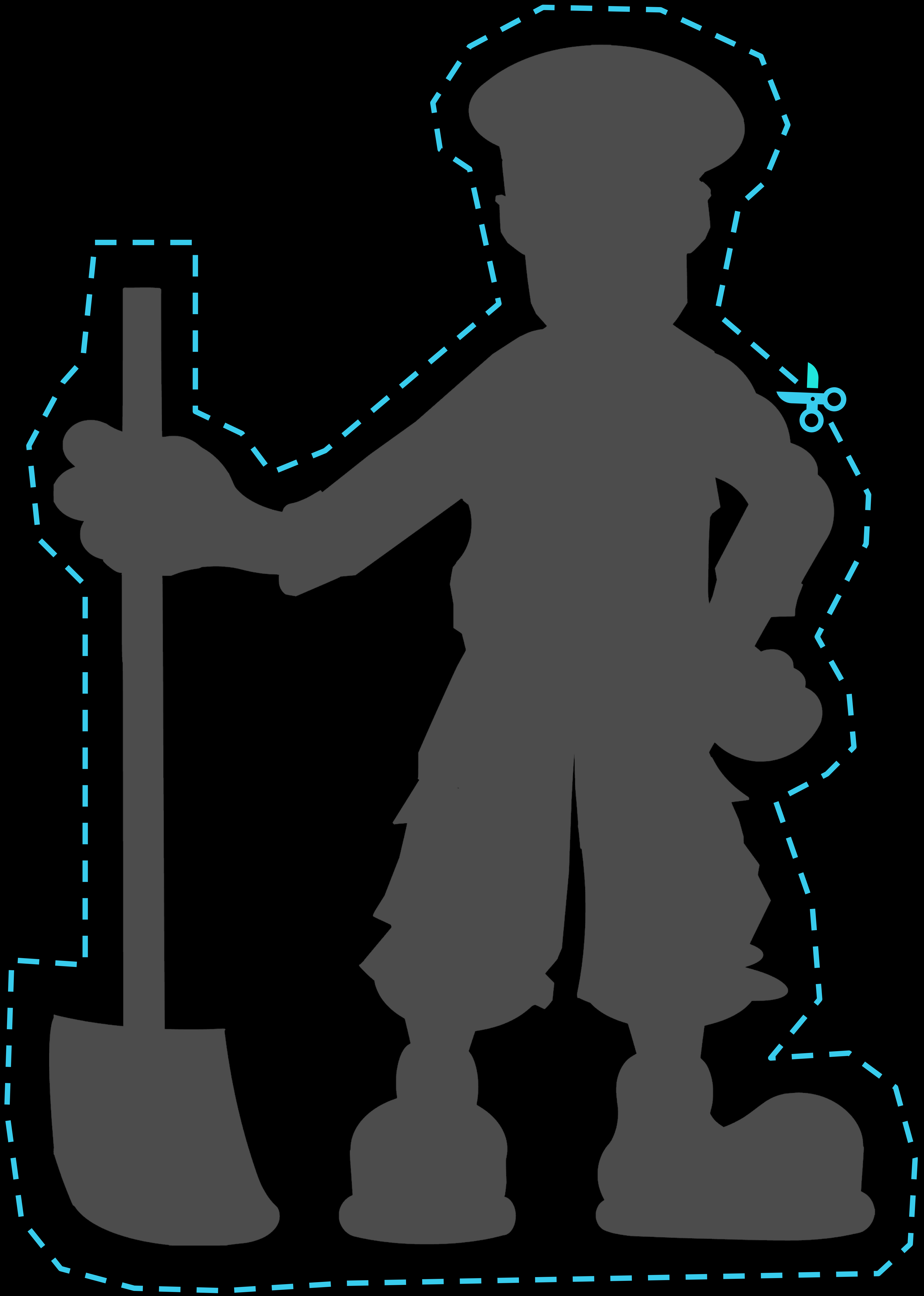
CONFLUENT

# Kafka without Zookeeper

## You can go your way

@gamussa | @confluentinc

ZOO



Streams

- ★gamov.dev/twitter
- ★gamov.dev/telegram
- ★gamov.dev/youtube

# Agenda

## What we are going to talk about

# Agenda

## What we are going to talk about

- Brief history of Kafka

# Agenda

## What we are going to talk about

- Brief history of Kafka
- What's Zookeeper?

# Agenda

## What we are going to talk about

- Brief history of Kafka
- What's Zookeeper?
    - What Does it Do:

# Agenda

## What we are going to talk about

- Brief history of Kafka
- What’s Zookeeper?
    - What Does it Do:
        - For clients

# Agenda

## What we are going to talk about

- Brief history of Kafka
- What's Zookeeper?
  - What Does it Do:
    - For clients
    - For brokers

# Agenda

## What we are going to talk about

- Brief history of Kafka
- What's Zookeeper?
  - What Does it Do:
    - For clients
    - For brokers
- Brave New World - without Zookeeper

# Agenda

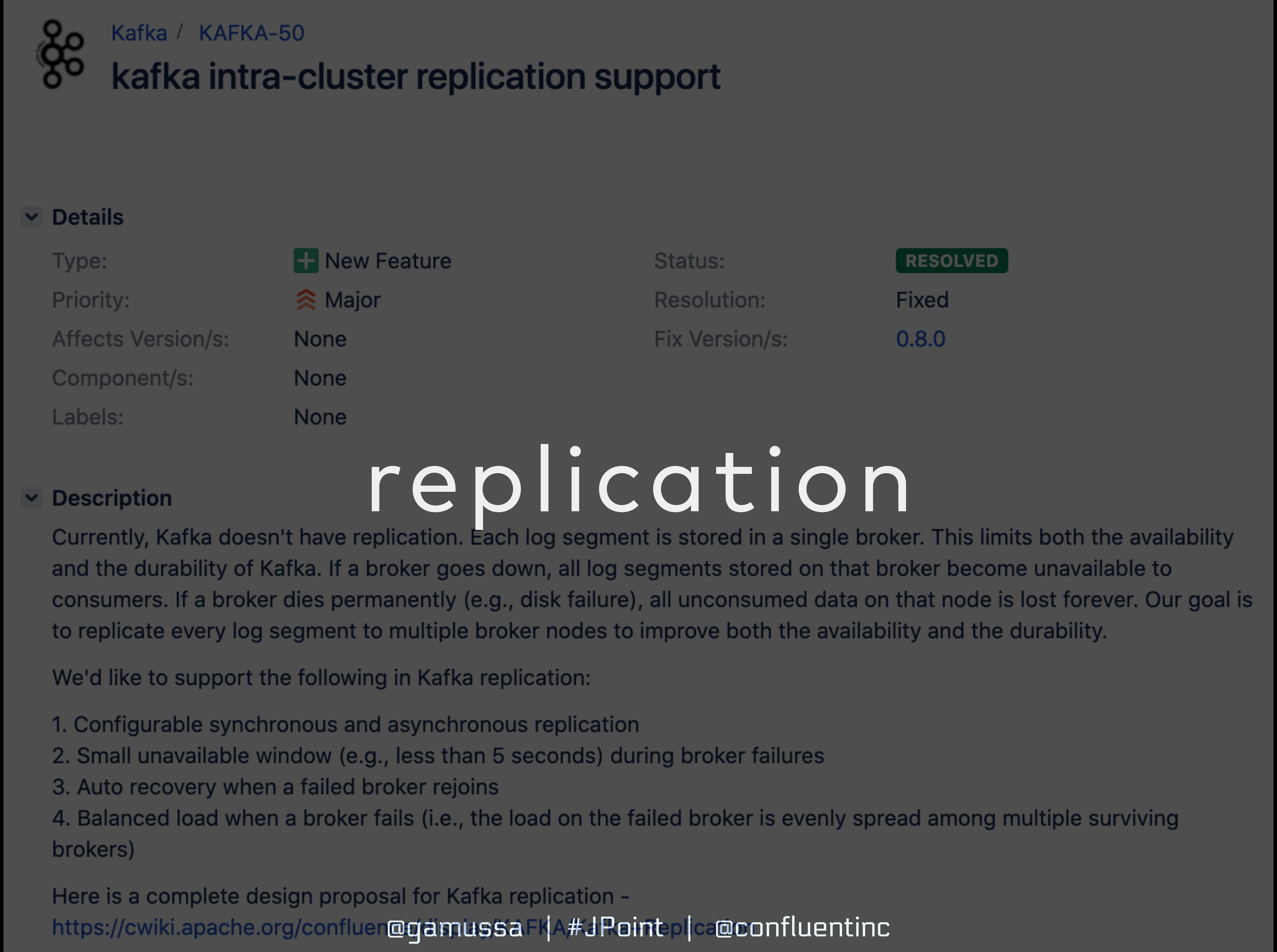
## What we are going to talk about

- Brief history of Kafka
- What's Zookeeper?
    - What Does it Do:
        - For clients
        - For brokers
- Brave New World - without Zookeeper
- Demo 🔥

# Kafka Past and Future

# replication

# replication



Kafka / KAFKA-50

## kafka intra-cluster replication support

**Details**

| | | | |
|---|---|---|---|
| Type: | New Feature | Status: | RESOLVED |
| Priority: | Major | Resolution: | Fixed |
| Affects Version/s: | None | Fix Version/s: | 0.8.0 |
| Component/s: | None | | |
| Labels: | None | | |

**Description**

Currently, Kafka doesn't have replication. Each log segment is stored in a single broker. This limits both the availability and the durability of Kafka. If a broker goes down, all log segments stored on that broker become unavailable to consumers. If a broker dies permanently (e.g., disk failure), all unconsumed data on that node is lost forever. Our goal is to replicate every log segment to multiple broker nodes to improve both the availability and the durability.

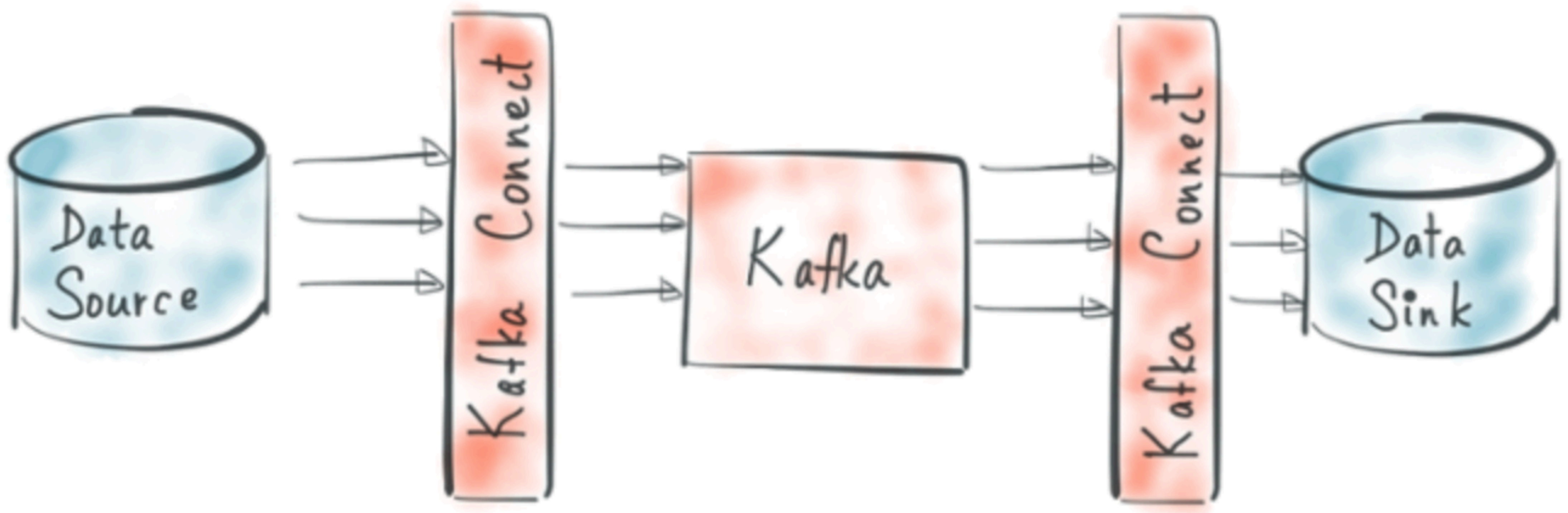
We'd like to support the following in Kafka replication:

1. Configurable synchronous and asynchronous replication
2. Small unavailable window (e.g., less than 5 seconds) during broker failures
3. Auto recovery when a failed broker rejoins
4. Balanced load when a broker fails (i.e., the load on the failed broker is evenly spread among multiple surviving brokers)

Here is a complete design proposal for Kafka replication -
https://cwiki.apache.org/confluence/display/KAFKA/Kafka+Replication

## Kafka Connect

Kafka has become a tremendously popular system to enable streaming data flow between external systems to unlock data siloes and to exchange data in real-time. And indeed the open source community has written numerous connectors, such as Camus, to integrate Kafka with other systems. But unfortunately for users every such integration tool looks very different, and most don't attempt to solve all the problems that need to be addressed for reliable large-scale data ingestion. Users have been forced to understand and operate many different one-off integration tools as their data infrastructure and systems proliferate. Furthermore, many of these one-off tools do not offer high availability or adequate scalability. This presents difficulties for the adoption of Kafka for data integration purposes. To address this situation, Kafka 0.9 adds support for a new feature called *Kafka Connect* (those who follow the open source discussions closely might have heard it by it's working name "Copycat").



# Introducing Kafka Streams: Stream Processing Made Simple

JAY KREPS

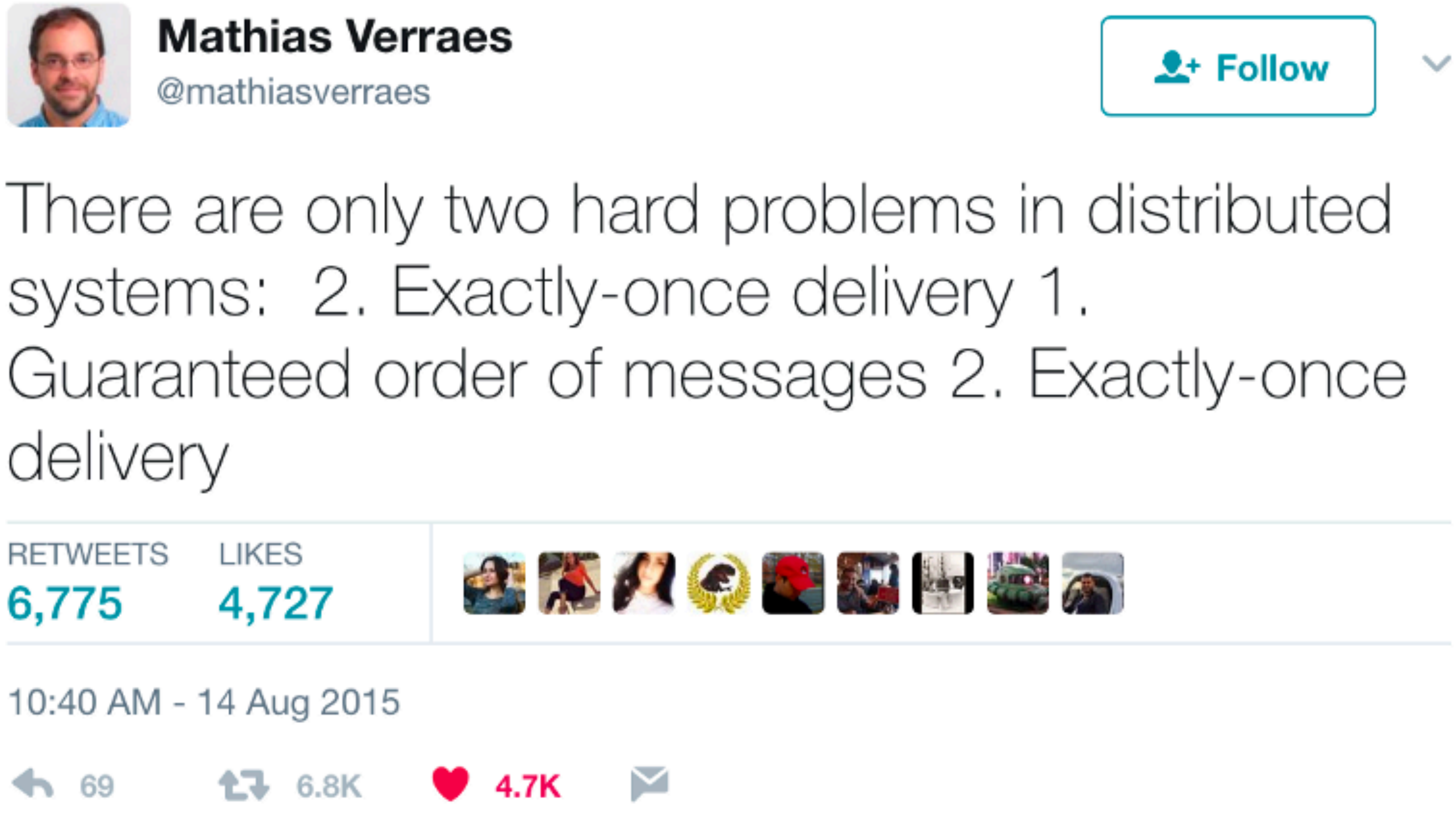
MARCH 10, 2016

I'm really excited to announce a major new feature in Apache Kafka v0.10: Kafka's Streams API. The Streams API, available as a Java library that is part of the official Kafka project, is the easiest way to write mission-critical, real-time applications and microservices with all the benefits of Kafka's server-side cluster technology.

> Note
>
> The latest documentation on Apache Kafka's Streams API is always available at https://kafka.apache.org/documentation/streams/

A stream processing application built with Kafka Streams looks like this:

```
import org.apache.kafka.common.serialization.Serdes;
import org.apache.kafka.streams.KafkaStreams;
import org.apache.kafka.streams.StreamsConfig;
import org.apache.kafka.streams.kstream.KStream;
import org.apache.kafka.streams.kstream.KStreamBuilder;
import org.apache.kafka.streams.kstream.KTable;
```



Mathias Verraes
@mathiasverraes

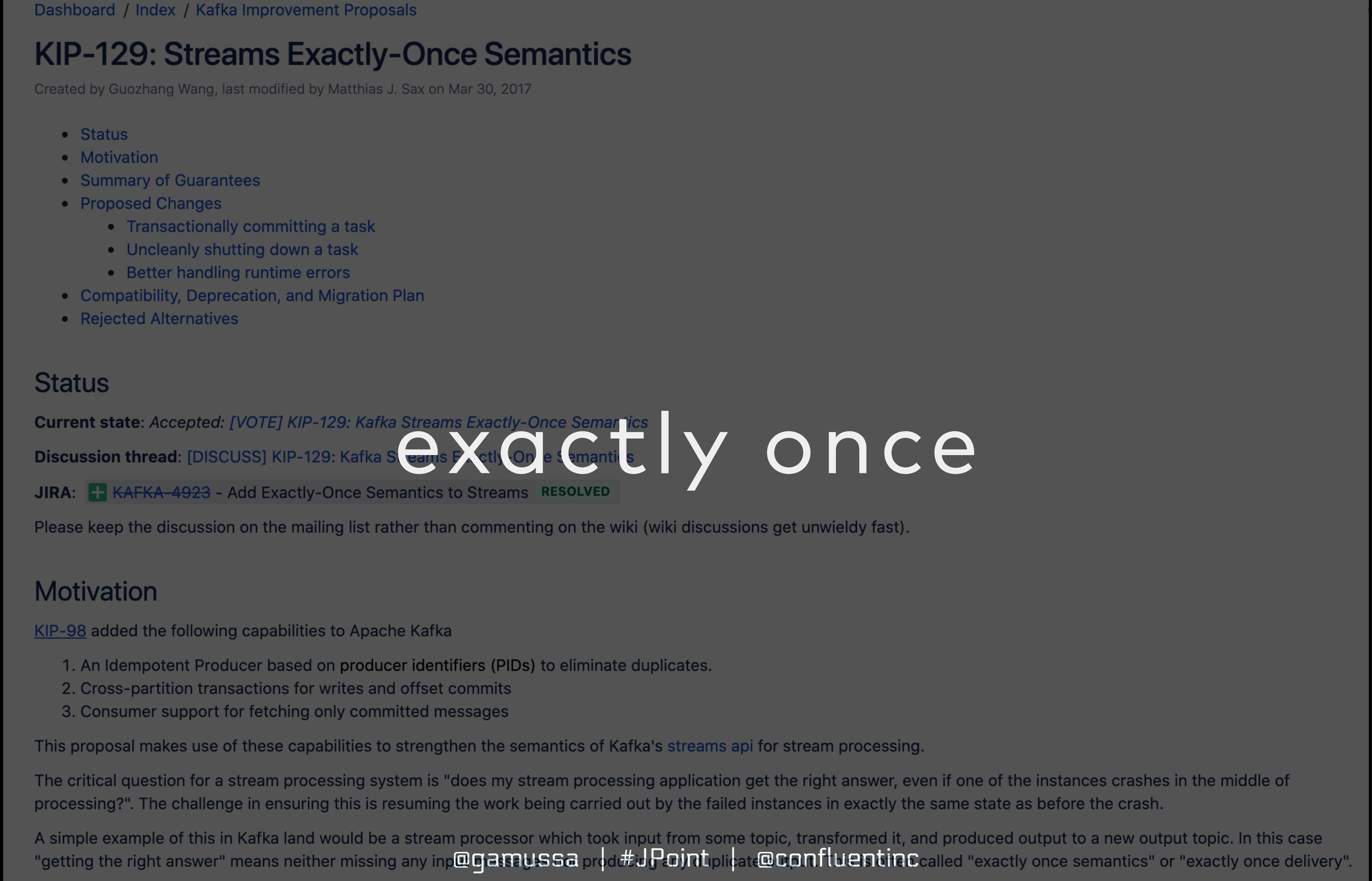
There are only two hard problems in distributed systems:  2. Exactly-once delivery 1. Guaranteed order of messages 2. Exactly-once delivery

RETWEETS 6,775
LIKES 4,727

10:40 AM - 14 Aug 2015

# exactly once

# exactly once



Dashboard / Index / Kafka Improvement Proposals

## KIP-129: Streams Exactly-Once Semantics

Created by Guozhang Wang, last modified by Matthias J. Sax on Mar 30, 2017

- Status
- Motivation
- Summary of Guarantees
- Proposed Changes
    - Transactionally committing a task
    - Uncleanly shutting down a task
    - Better handling runtime errors
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives

## Status

**Current state**: *Accepted: [VOTE] KIP-129: Kafka Streams Exactly-Once Semantics*

**Discussion thread**: [DISCUSS] KIP-129: Kafka Streams Exactly-Once Semantics

**JIRA**: ~~KAFKA-4923~~ - Add Exactly-Once Semantics to Streams RESOLVED

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

## Motivation

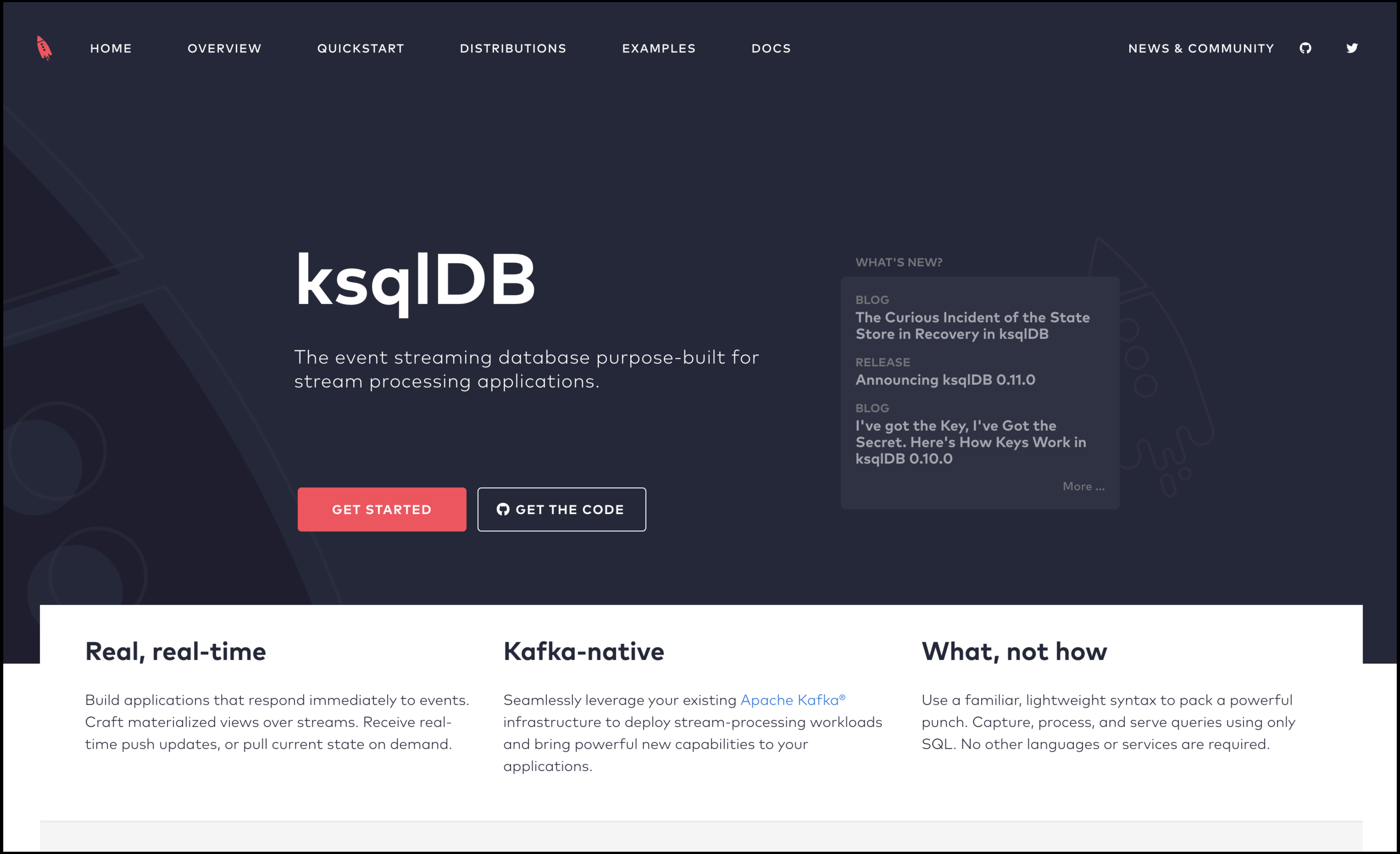
KIP-98 added the following capabilities to Apache Kafka

1. An Idempotent Producer based on **producer identifiers (PIDs)** to eliminate duplicates.
2. Cross-partition transactions for writes and offset commits
3. Consumer support for fetching only committed messages

This proposal makes use of these capabilities to strengthen the semantics of Kafka's streams api for stream processing.

The critical question for a stream processing system is "does my stream processing application get the right answer, even if one of the instances crashes in the middle of processing?". The challenge in ensuring this is resuming the work being carried out by the failed instances in exactly the same state as before the crash.

A simple example of this in Kafka land would be a stream processor which took input from some topic, transformed it, and produced output to a new output topic. In this case "getting the right answer" means neither missing any input messages or producing any duplicate output. This is often called "exactly once semantics" or "exactly once delivery".



# ksqlDB

The event streaming database purpose-built for stream processing applications.

WHAT'S NEW?

BLOG
The Curious Incident of the State Store in Recovery in ksqlDB

RELEASE
Announcing ksqlDB 0.11.0

BLOG
I've got the Key, I've Got the Secret. Here's How Keys Work in ksqlDB 0.10.0

## Real, real-time

Build applications that respond immediately to events. Craft materialized views over streams. Receive real-time push updates, or pull current state on demand.

## Kafka-native

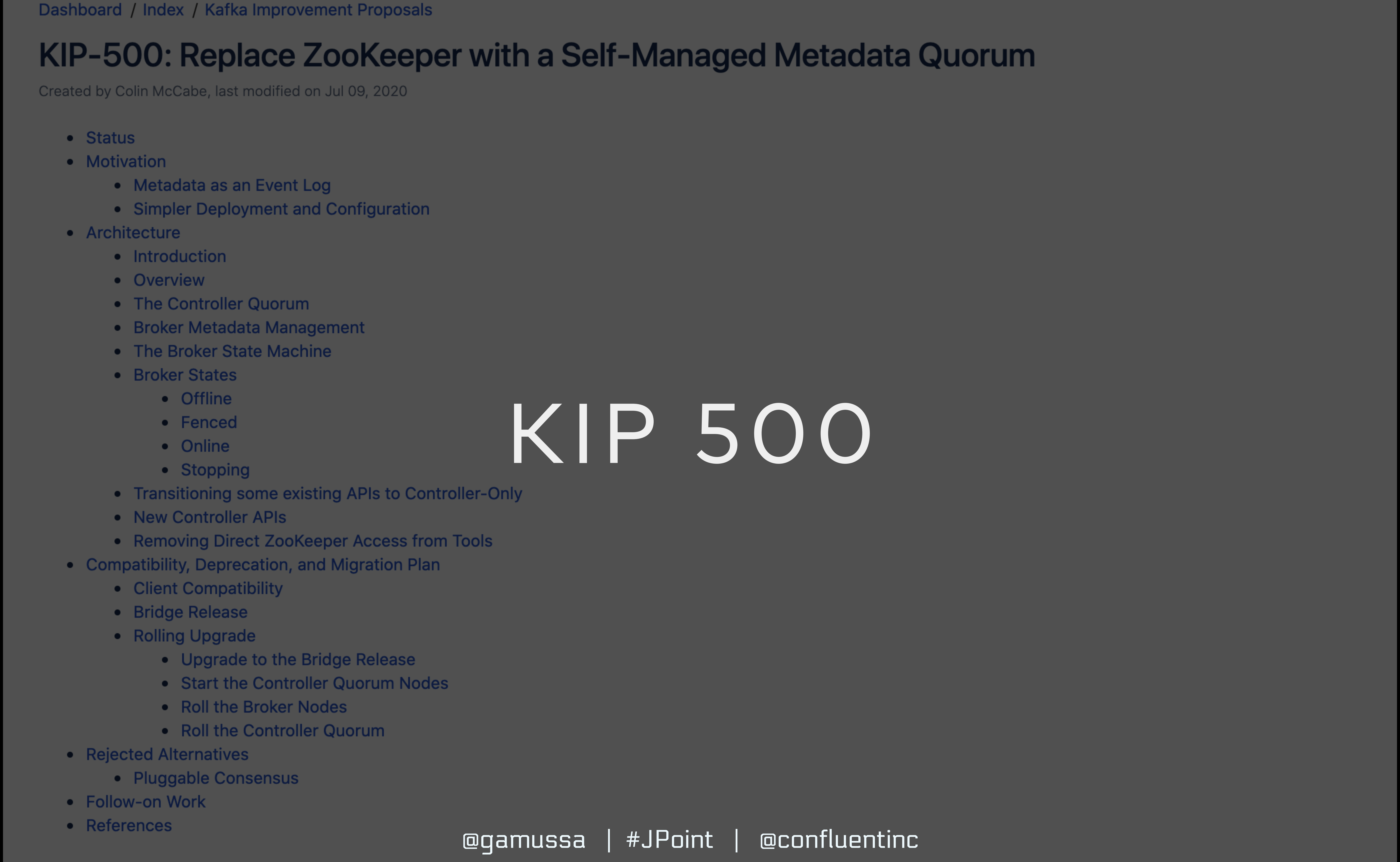
Seamlessly leverage your existing Apache Kafka® infrastructure to deploy stream-processing workloads and bring powerful new capabilities to your applications.

## What, not how

Use a familiar, lightweight syntax to pack a powerful punch. Capture, process, and serve queries using only SQL. No other languages or services are required.

# KIP 500

# KIP 500



Dashboard / Index / Kafka Improvement Proposals

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

Created by Colin McCabe, last modified on Jul 09, 2020

- Status
- Motivation
    - Metadata as an Event Log
    - Simpler Deployment and Configuration
- Architecture
    - Introduction
    - Overview
    - The Controller Quorum
    - Broker Metadata Management
    - The Broker State Machine
    - Broker States
        - Offline
        - Fenced
        - Online
        - Stopping
    - Transitioning some existing APIs to Controller-Only
    - New Controller APIs
    - Removing Direct ZooKeeper Access from Tools
- Compatibility, Deprecation, and Migration Plan
    - Client Compatibility
    - Bridge Release
    - Rolling Upgrade
        - Upgrade to the Bridge Release
        - Start the Controller Quorum Nodes
        - Roll the Broker Nodes
        - Roll the Controller Quorum
- Rejected Alternatives
    - Pluggable Consensus
- Follow-on Work
- References

**KIP-455**: Create an Administrative API for Replica Reassignment

**KIP-497**: Add inter-broker API to alter ISR

**KIP-543**: Expand ConfigCommand's non-ZK functionality

**KIP-555**: Deprecate Direct Zookeeper access in Kafka Administrative Tools

**KIP-589**: Add API to update Replica state in Controller

**KIP-590**: Redirect Zookeeper Mutation Protocols to The Controller

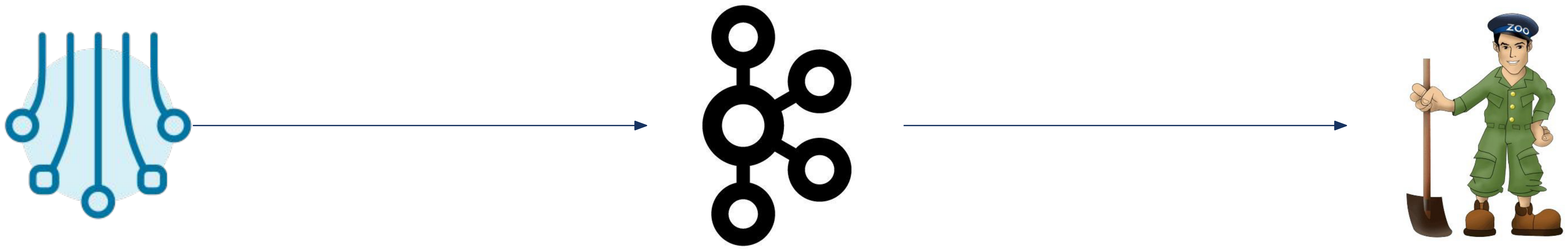
**KIP-595**: A Raft Protocol for the Metadata Quorum

**KIP-631**: The Quorum-based Kafka Controller

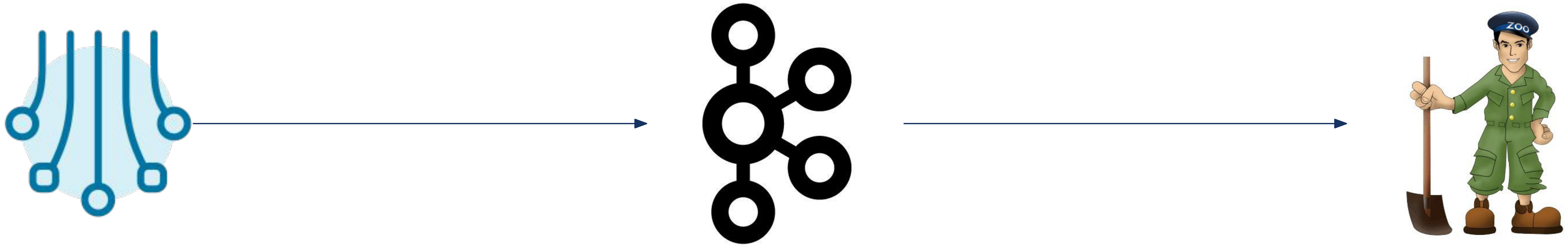
# How Kafka uses ZooKeeper

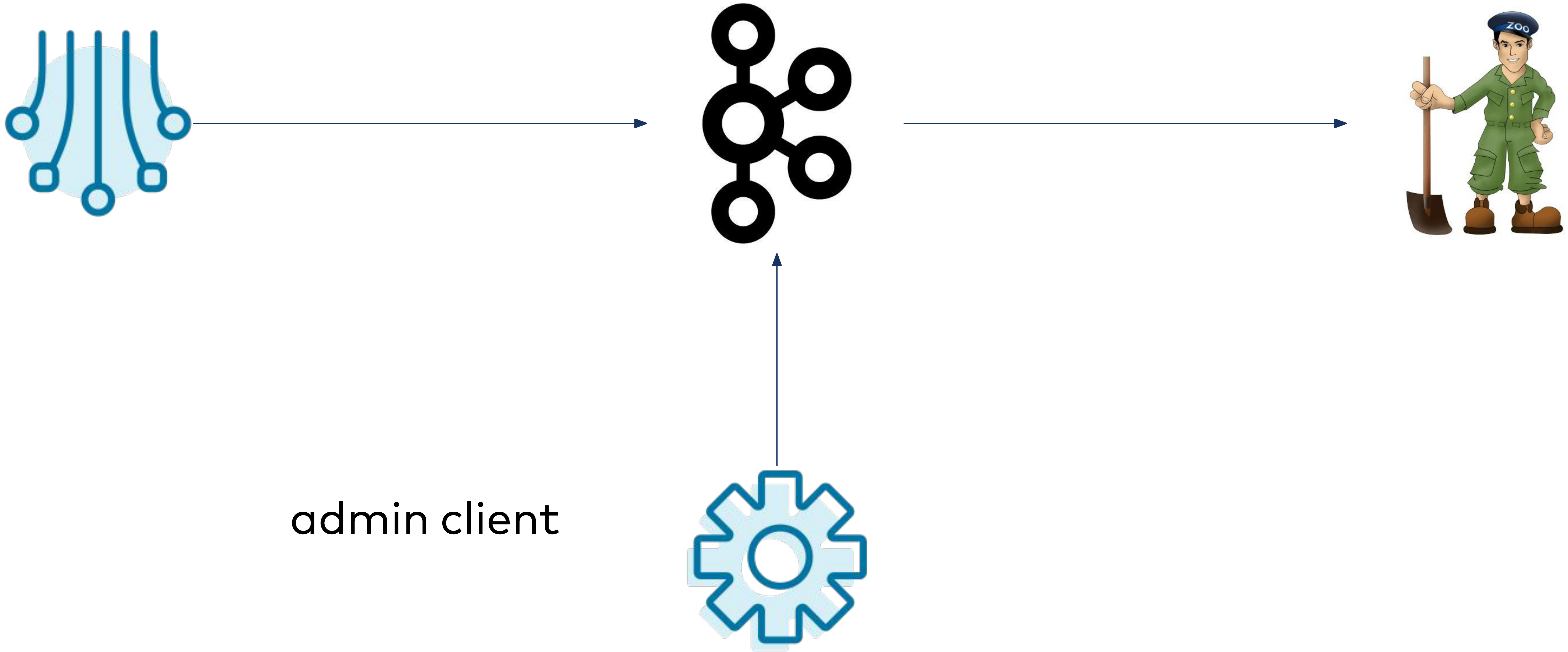
# Clients

ZOO

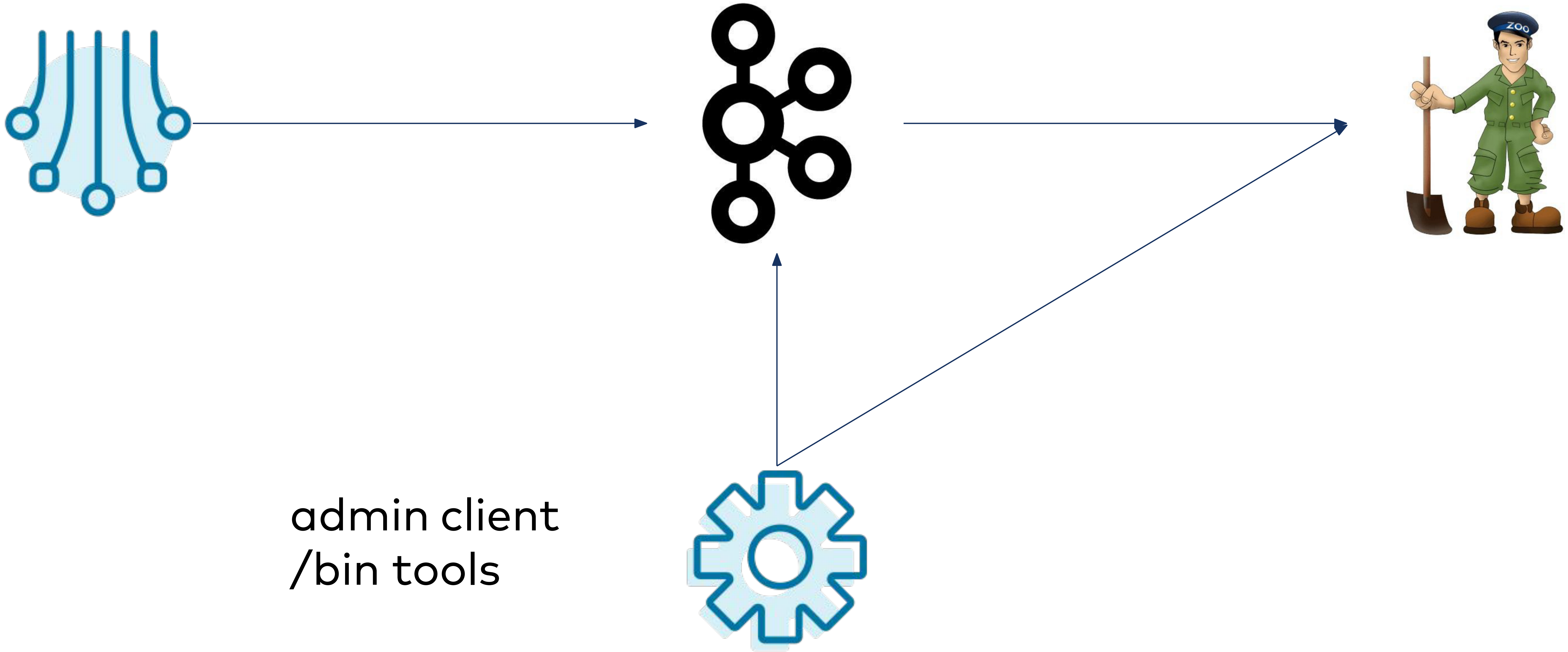

ZOO

producer
consumer
streams
connect

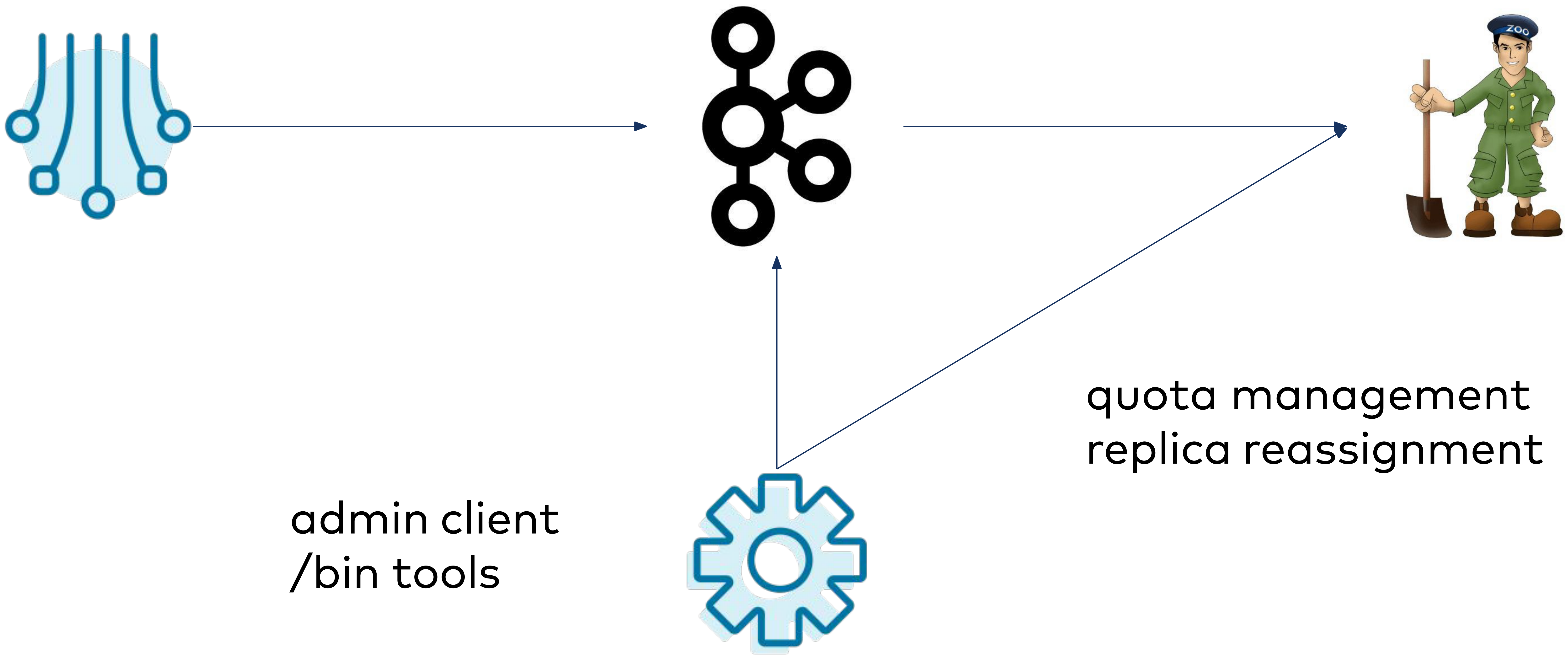




admin client

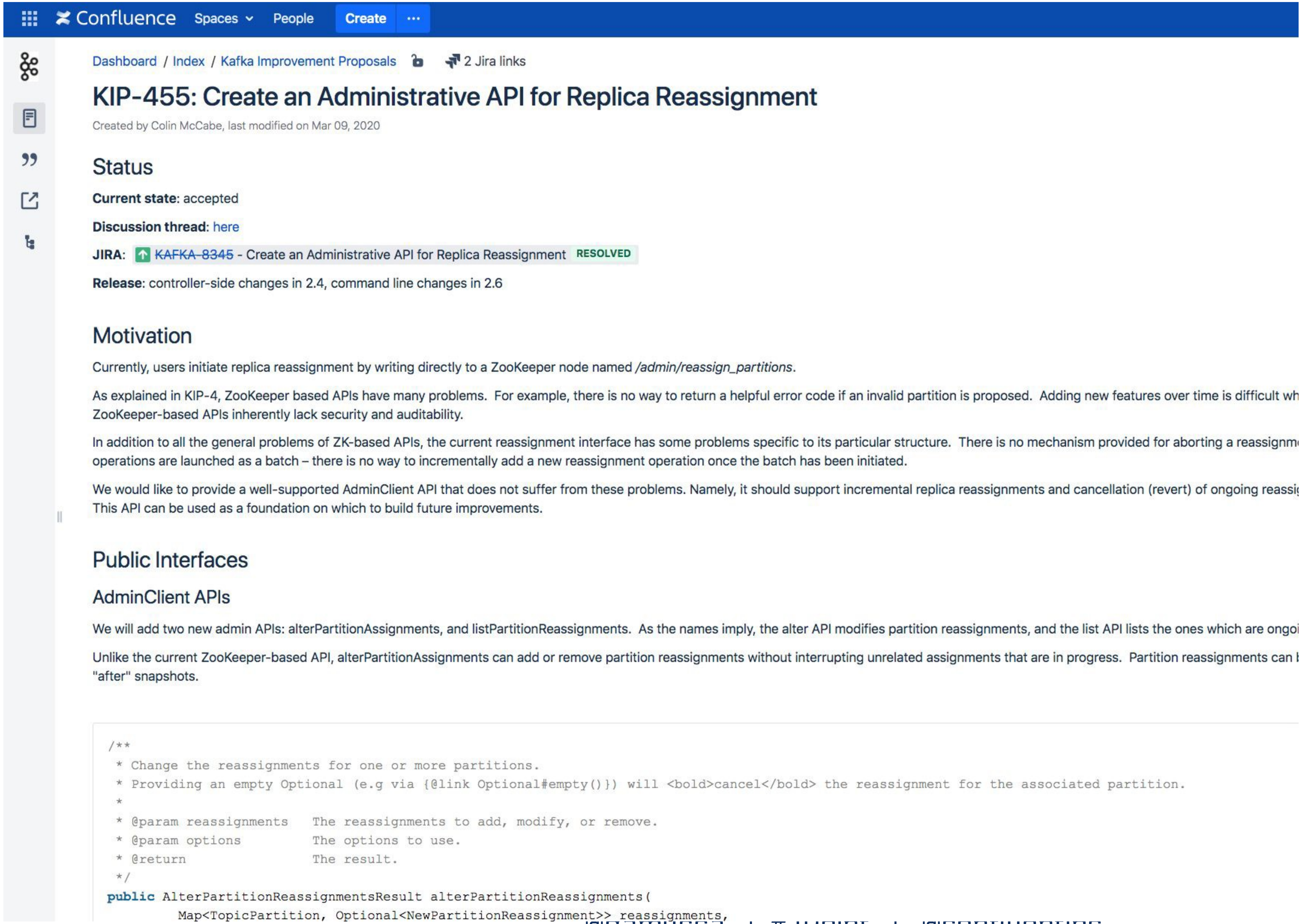


admin client
/bin tools



quota management
replica reassignment

admin client
/bin tools



Dashboard / Index / Kafka Improvement Proposals 2 Jira links

# KIP-455: Create an Administrative API for Replica Reassignment

Created by Colin McCabe, last modified on Mar 09, 2020

## Status

**Current state**: accepted

**Discussion thread**: here

**JIRA**: ~~KAFKA-8345~~ - Create an Administrative API for Replica Reassignment RESOLVED

**Release**: controller-side changes in 2.4, command line changes in 2.6

## Motivation

Currently, users initiate replica reassignment by writing directly to a ZooKeeper node named */admin/reassign_partitions*.

As explained in KIP-4, ZooKeeper based APIs have many problems. For example, there is no way to return a helpful error code if an invalid partition is proposed. Adding new features over time is difficult wh
ZooKeeper-based APIs inherently lack security and auditability.

In addition to all the general problems of ZK-based APIs, the current reassignment interface has some problems specific to its particular structure. There is no mechanism provided for aborting a reassignm
operations are launched as a batch – there is no way to incrementally add a new reassignment operation once the batch has been initiated.

We would like to provide a well-supported AdminClient API that does not suffer from these problems. Namely, it should support incremental replica reassignments and cancellation (revert) of ongoing reassi
This API can be used as a foundation on which to build future improvements.

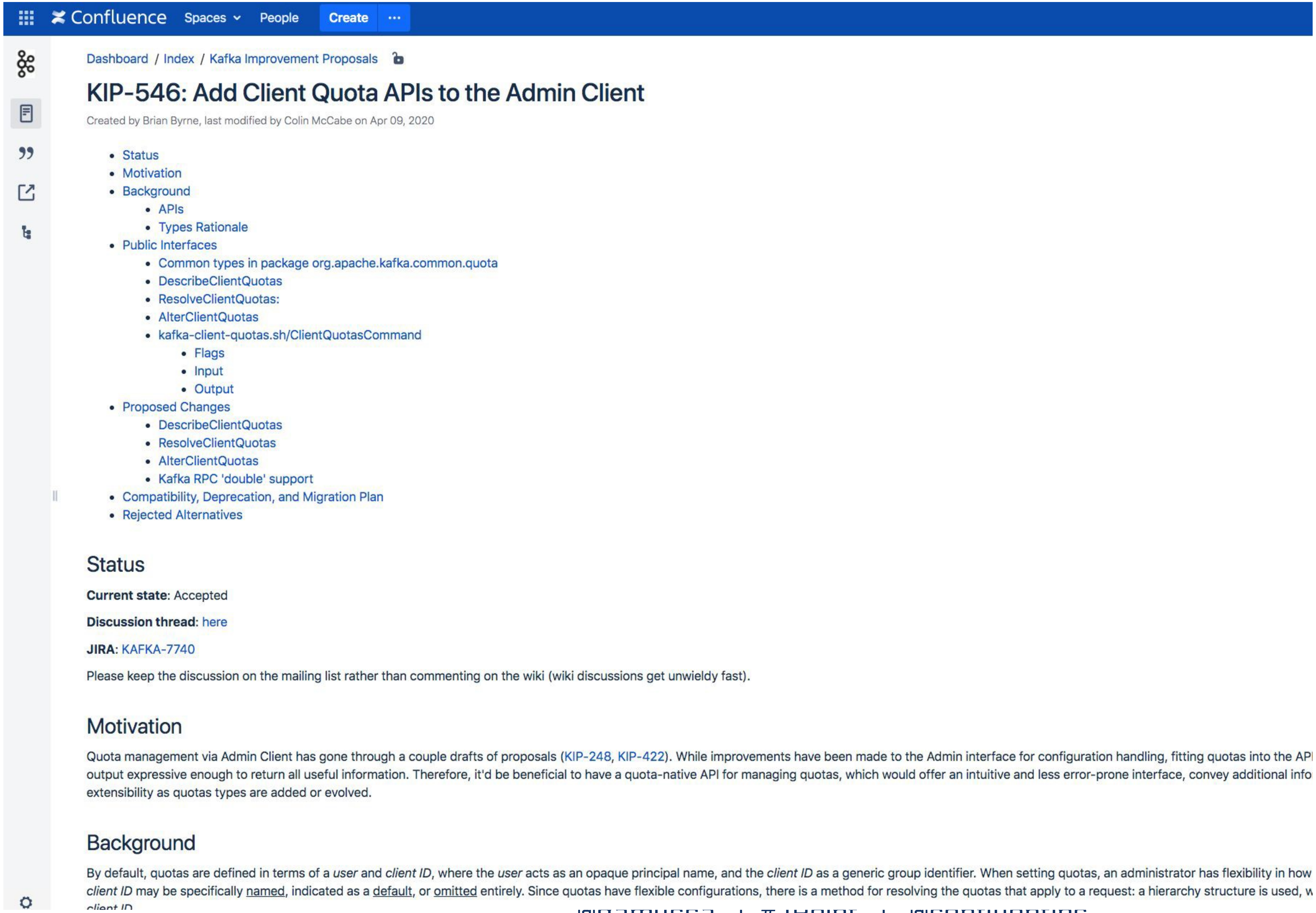
## Public Interfaces

### AdminClient APIs

We will add two new admin APIs: alterPartitionAssignments, and listPartitionReassignments. As the names imply, the alter API modifies partition reassignments, and the list API lists the ones which are ongoi

Unlike the current ZooKeeper-based API, alterPartitionAssignments can add or remove partition reassignments without interrupting unrelated assignments that are in progress. Partition reassignments can b
"after" snapshots.

```
/**
 * Change the reassignments for one or more partitions.
 * Providing an empty Optional (e.g via {@link Optional#empty()}) will <bold>cancel</bold> the reassignment for the associated partition.
 *
 * @param reassignments   The reassignments to add, modify, or remove.
 * @param options         The options to use.
 * @return                The result.
 */
public AlterPartitionReassignmentsResult alterPartitionReassignments(
        Map<TopicPartition, Optional<NewPartitionReassignment>> reassignments,
```



Dashboard / Index / Kafka Improvement Proposals

# KIP-546: Add Client Quota APIs to the Admin Client

Created by Brian Byrne, last modified by Colin McCabe on Apr 09, 2020

## Status

**Current state**: Accepted

**Discussion thread**: here

**JIRA**: KAFKA-7740

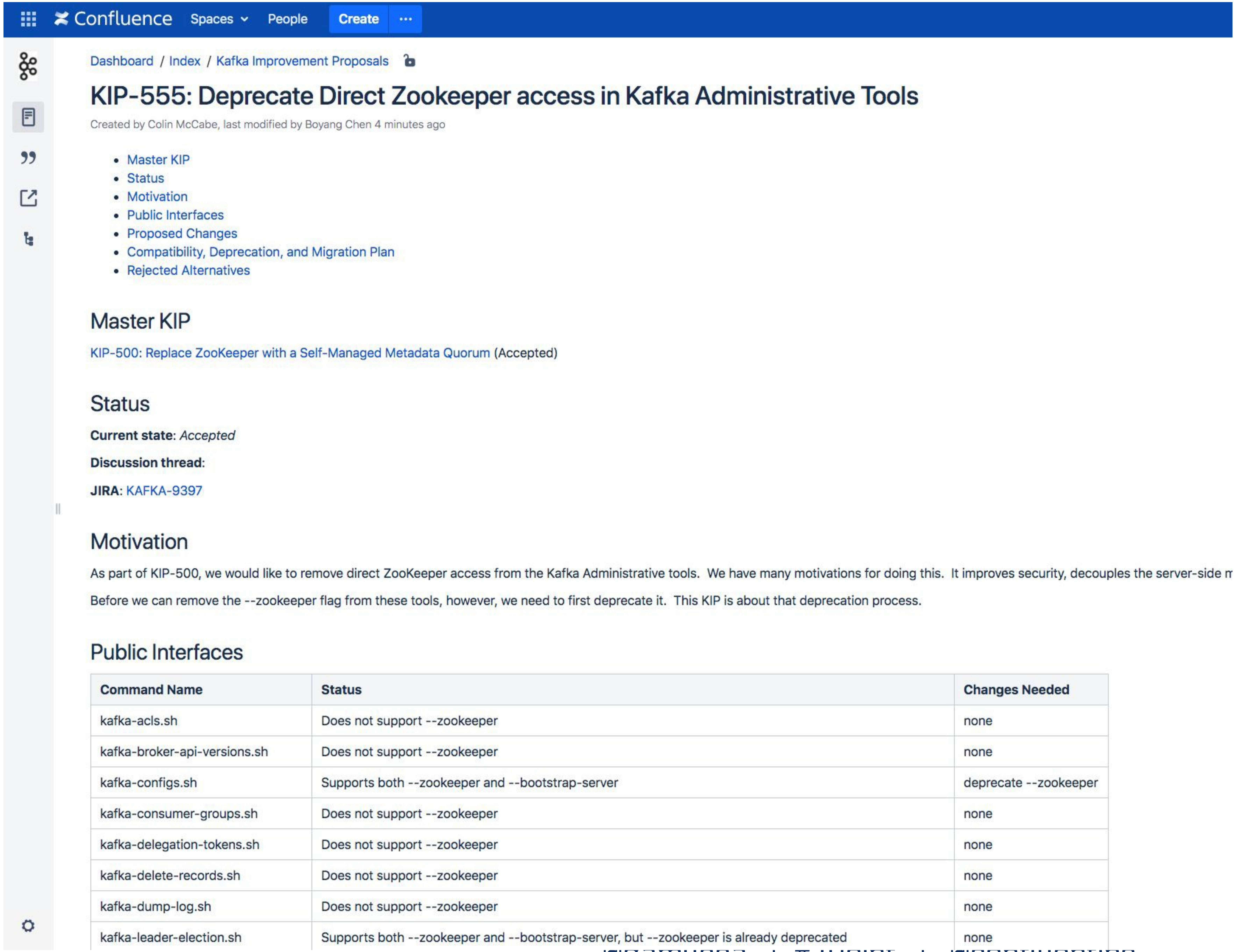
Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

## Motivation

Quota management via Admin Client has gone through a couple drafts of proposals (KIP-248, KIP-422). While improvements have been made to the Admin interface for configuration handling, fitting quotas into the API output expressive enough to return all useful information. Therefore, it'd be beneficial to have a quota-native API for managing quotas, which would offer an intuitive and less error-prone interface, convey additional info extensibility as quotas types are added or evolved.

## Background

By default, quotas are defined in terms of a *user* and *client ID*, where the *user* acts as an opaque principal name, and the *client ID* as a generic group identifier. When setting quotas, an administrator has flexibility in how *client ID* may be specifically named, indicated as a default, or omitted entirely. Since quotas have flexible configurations, there is a method for resolving the quotas that apply to a request: a hierarchy structure is used, w *client ID*



Dashboard / Index / Kafka Improvement Proposals

# KIP-555: Deprecate Direct Zookeeper access in Kafka Administrative Tools

Created by Colin McCabe, last modified by Boyang Chen 4 minutes ago

- Master KIP
- Status
- Motivation
- Public Interfaces
- Proposed Changes
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives

## Master KIP

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

## Status

**Current state**: *Accepted*

**Discussion thread**:

**JIRA**: KAFKA-9397

## Motivation

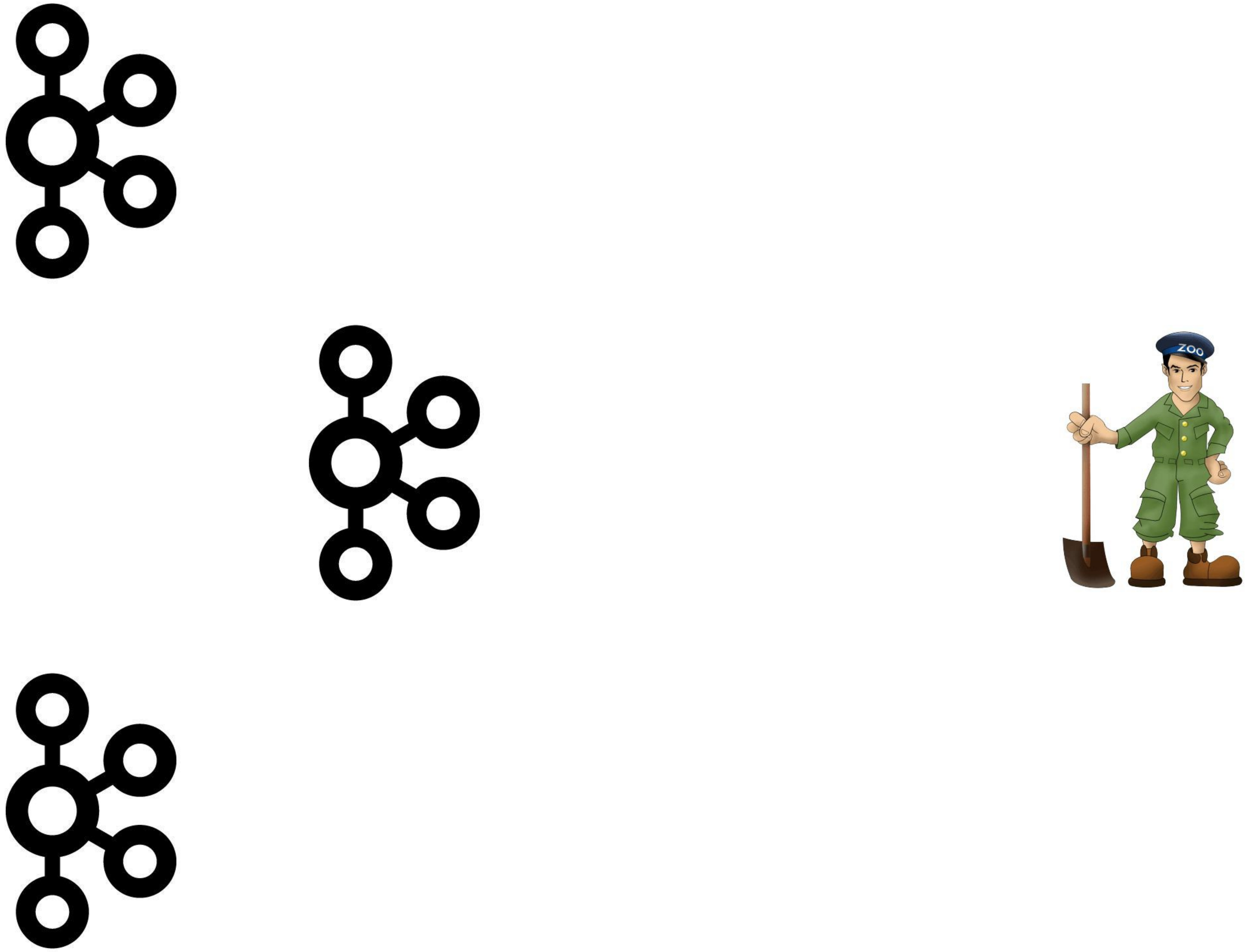
As part of KIP-500, we would like to remove direct ZooKeeper access from the Kafka Administrative tools. We have many motivations for doing this. It improves security, decouples the server-side m

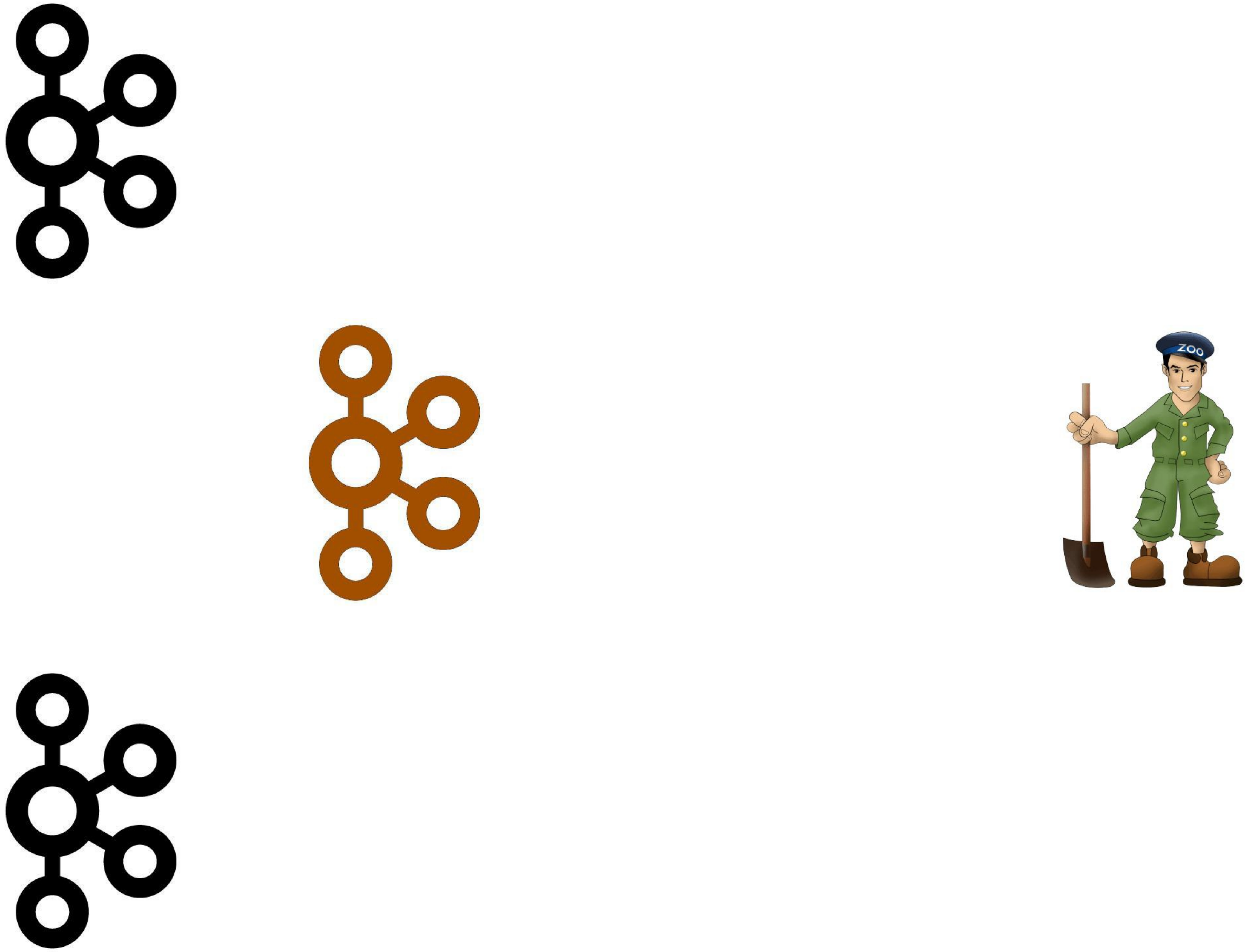
Before we can remove the --zookeeper flag from these tools, however, we need to first deprecate it. This KIP is about that deprecation process.

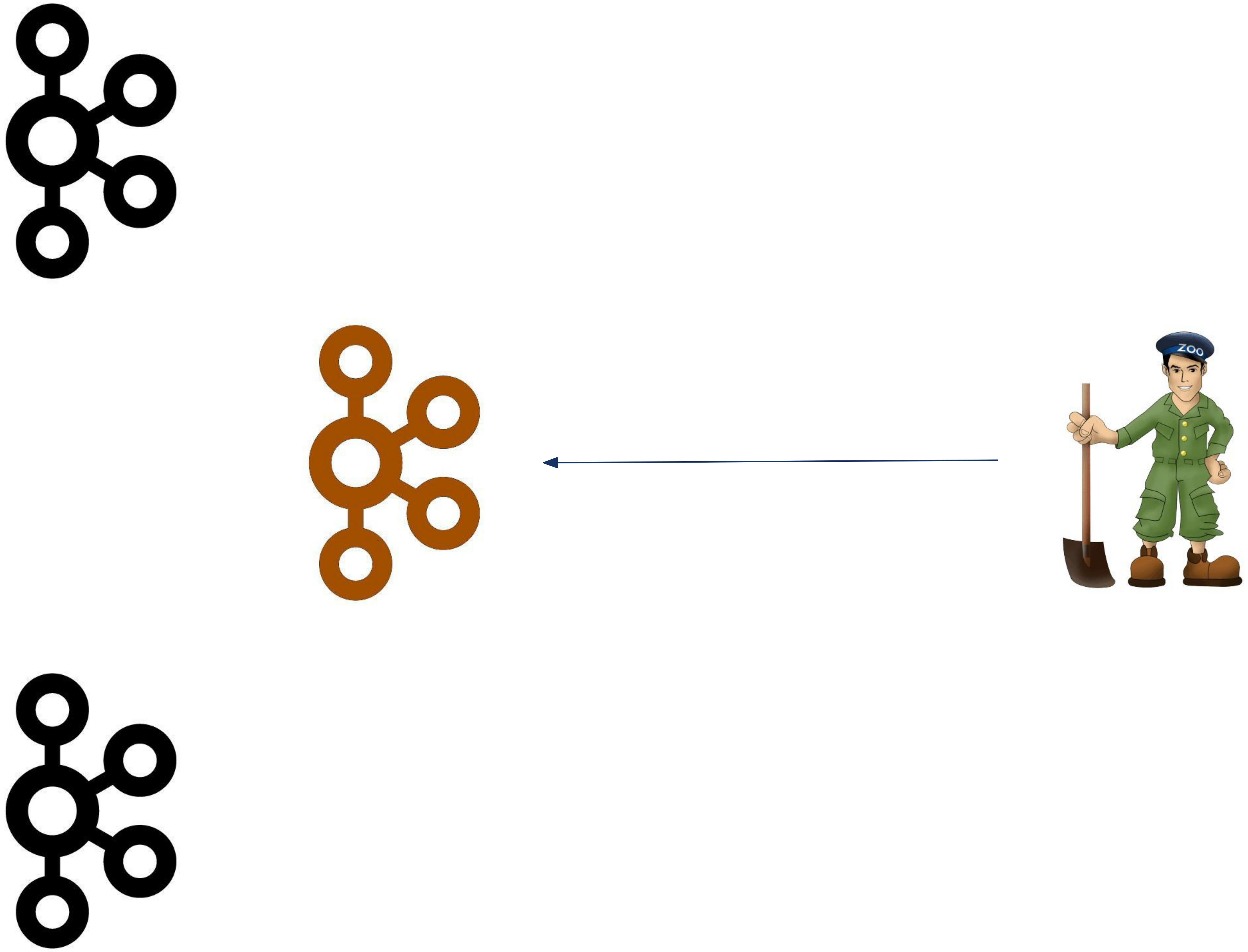
## Public Interfaces

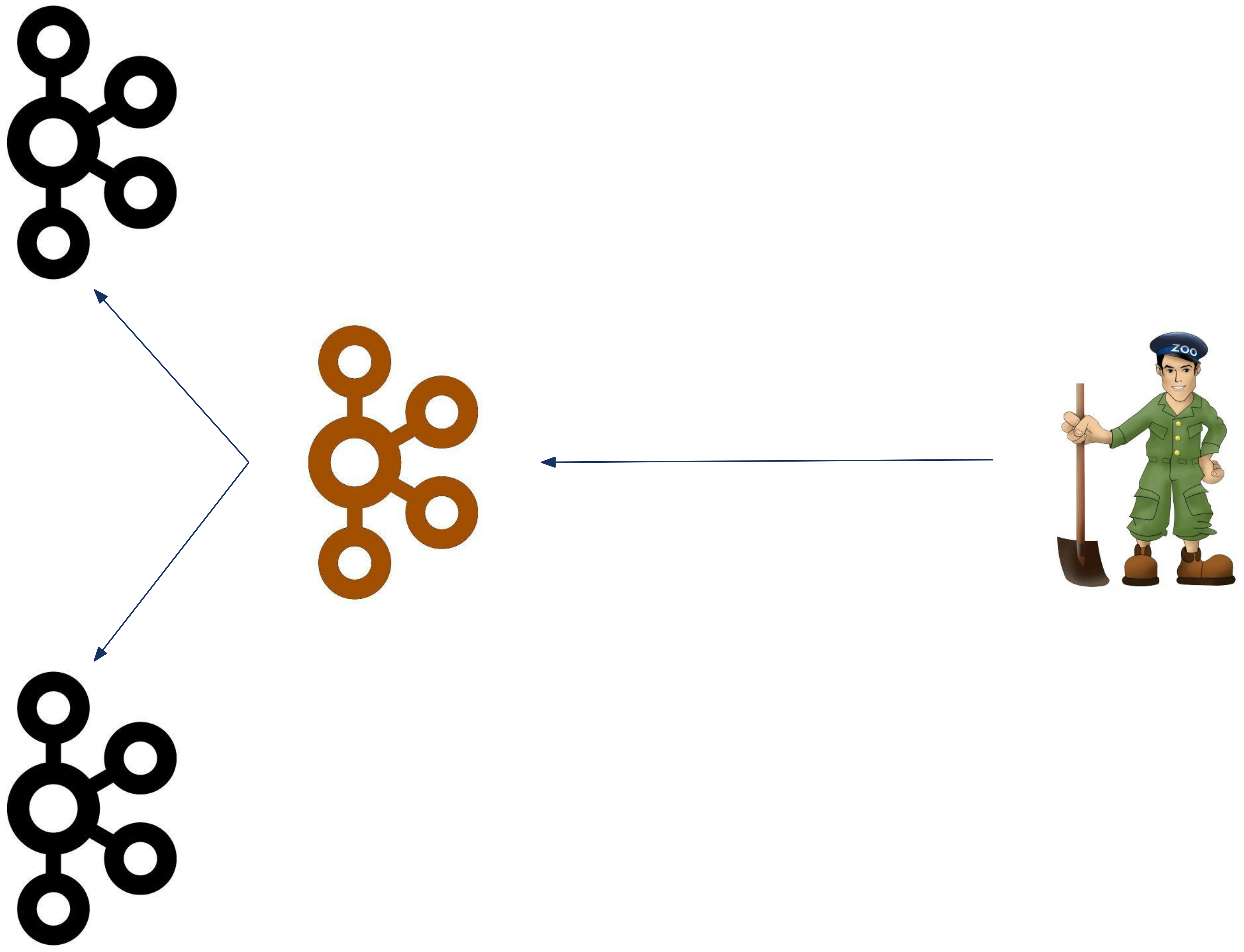
| Command Name | Status | Changes Needed |
|---|---|---|
| kafka-acls.sh | Does not support --zookeeper | none |
| kafka-broker-api-versions.sh | Does not support --zookeeper | none |
| kafka-configs.sh | Supports both --zookeeper and --bootstrap-server | deprecate --zookeeper |
| kafka-consumer-groups.sh | Does not support --zookeeper | none |
| kafka-delegation-tokens.sh | Does not support --zookeeper | none |
| kafka-delete-records.sh | Does not support --zookeeper | none |
| kafka-dump-log.sh | Does not support --zookeeper | none |
| kafka-leader-election.sh | Supports both --zookeeper and --bootstrap-server, but --zookeeper is already deprecated | none |

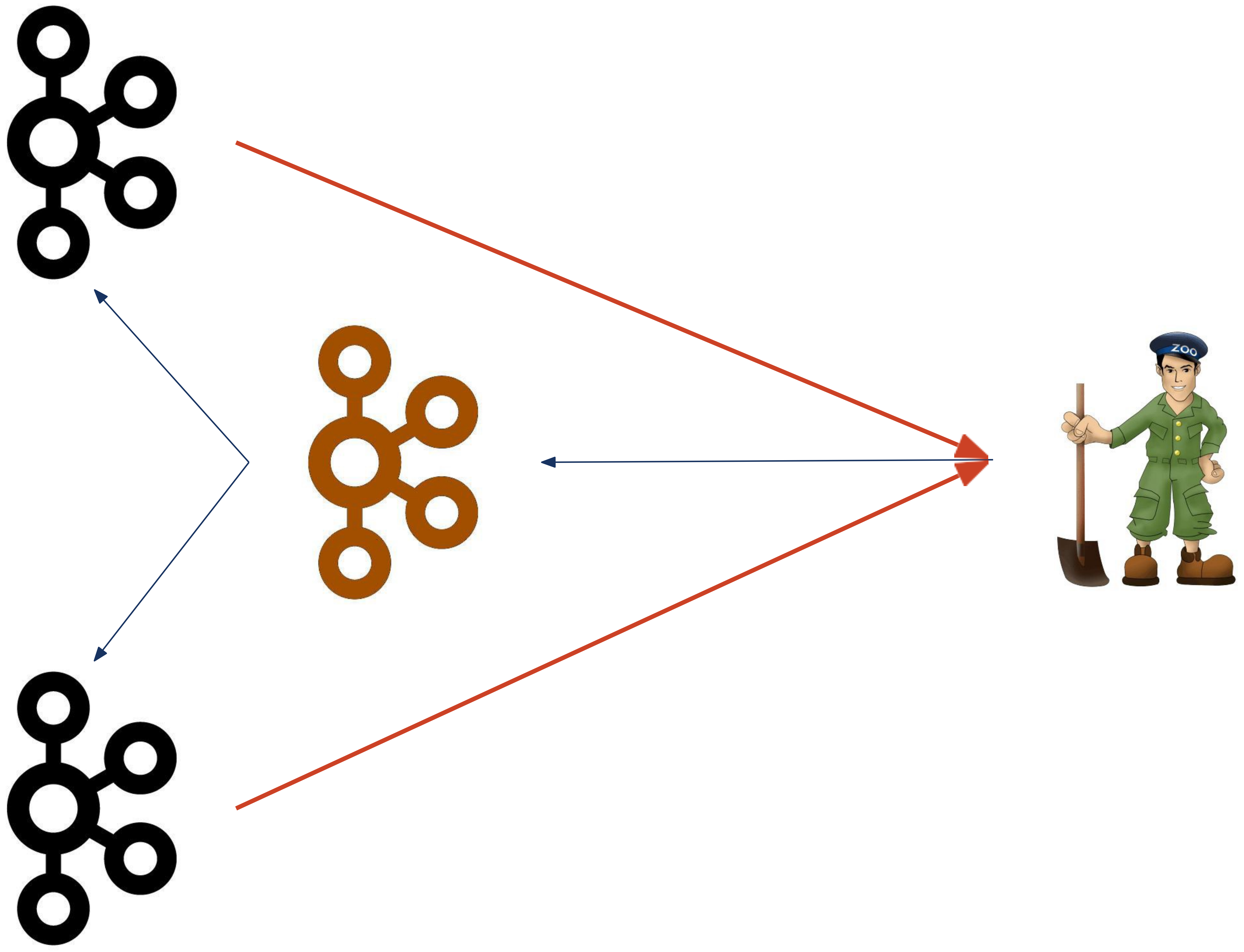
# Brokers

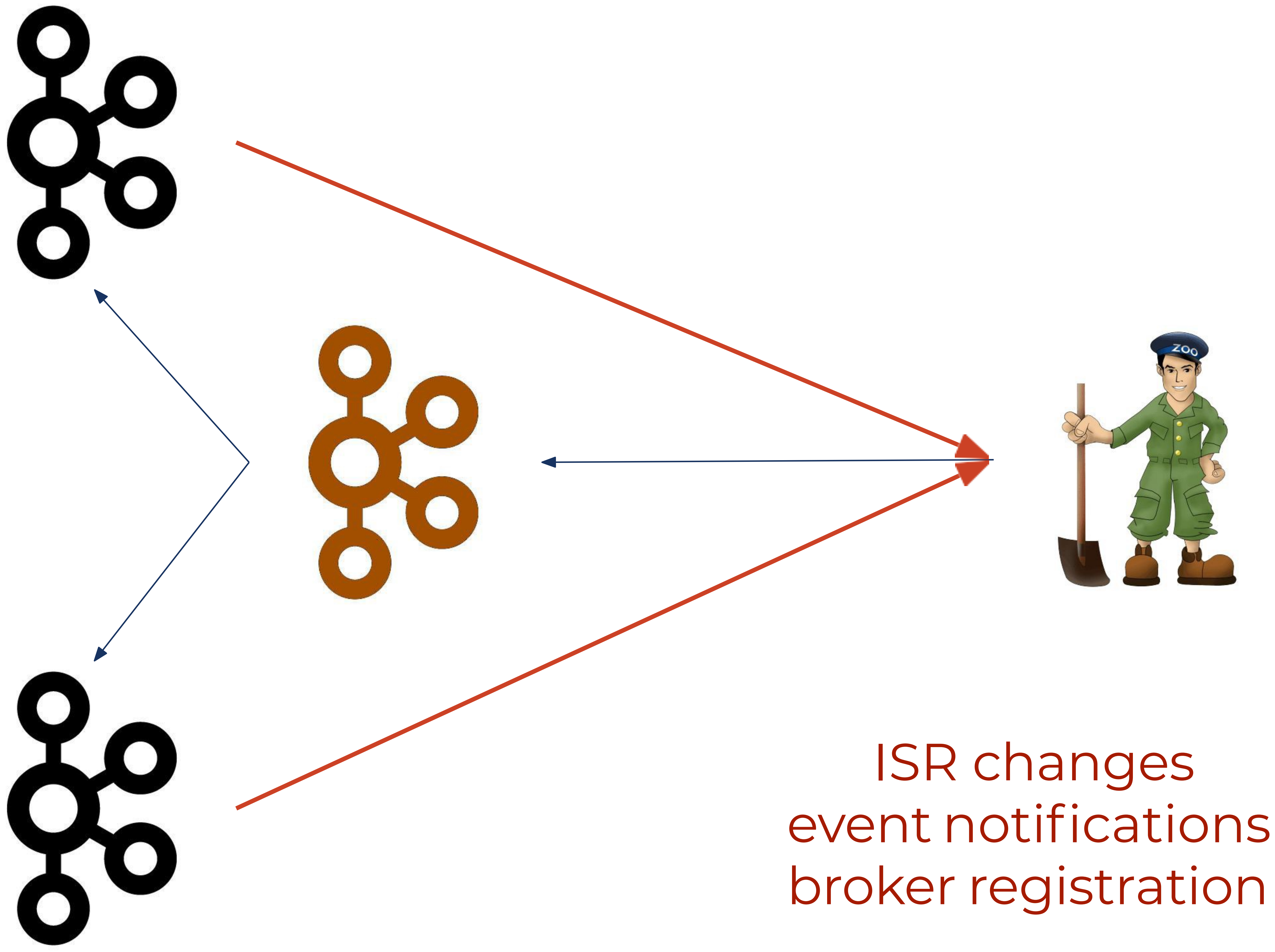

ZOO


ZOO

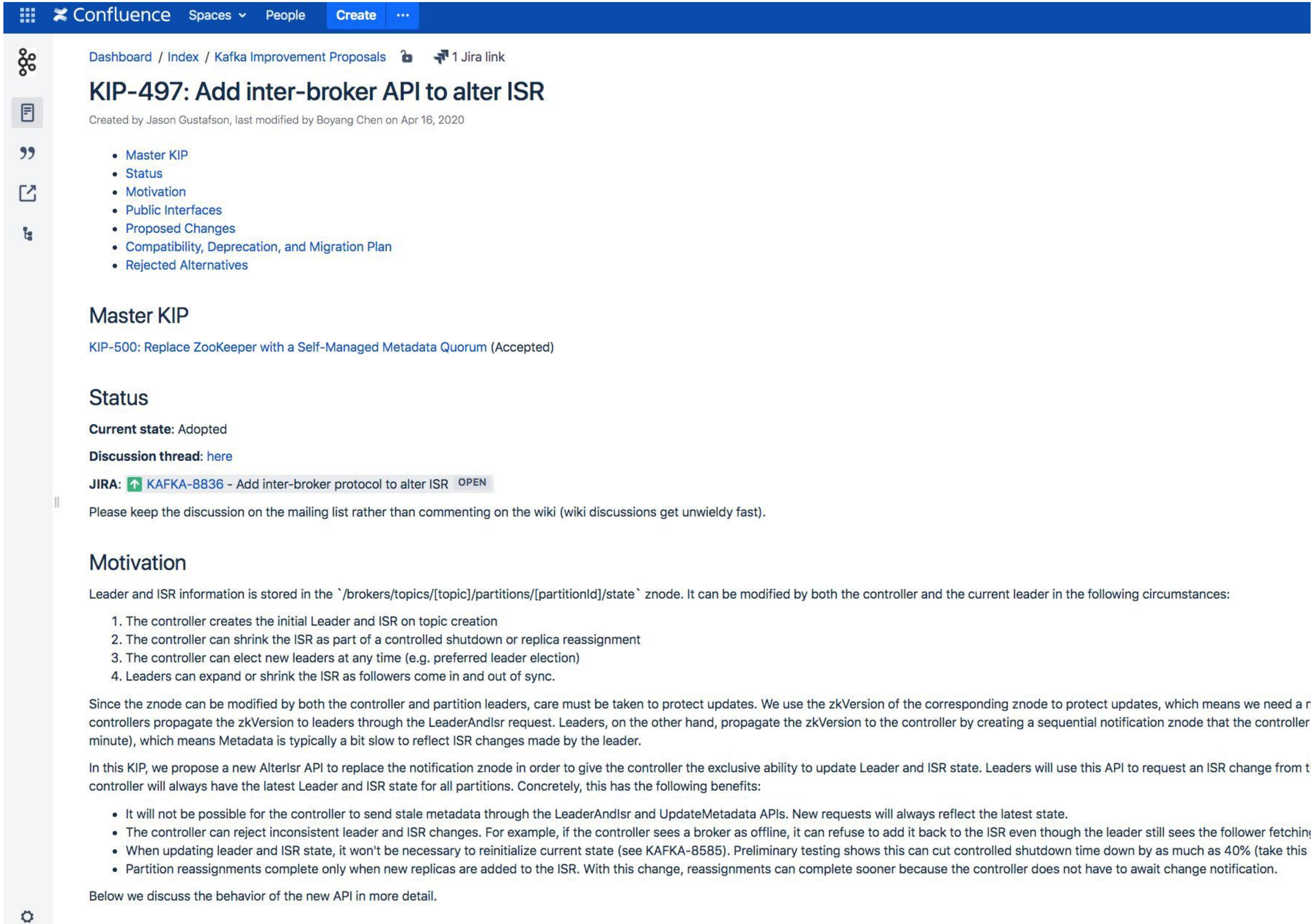

ZOO


ZOO


ZOO



ISR changes
event notifications
broker registration



Dashboard / Index / Kafka Improvement Proposals 1 Jira link

# KIP-497: Add inter-broker API to alter ISR

Created by Jason Gustafson, last modified by Boyang Chen on Apr 16, 2020

## Master KIP

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

## Status

**Current state**: Adopted

**Discussion thread**: here

**JIRA**: KAFKA-8836 - Add inter-broker protocol to alter ISR OPEN

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

## Motivation

Leader and ISR information is stored in the `/brokers/topics/[topic]/partitions/[partitionId]/state` znode. It can be modified by both the controller and the current leader in the following circumstances:

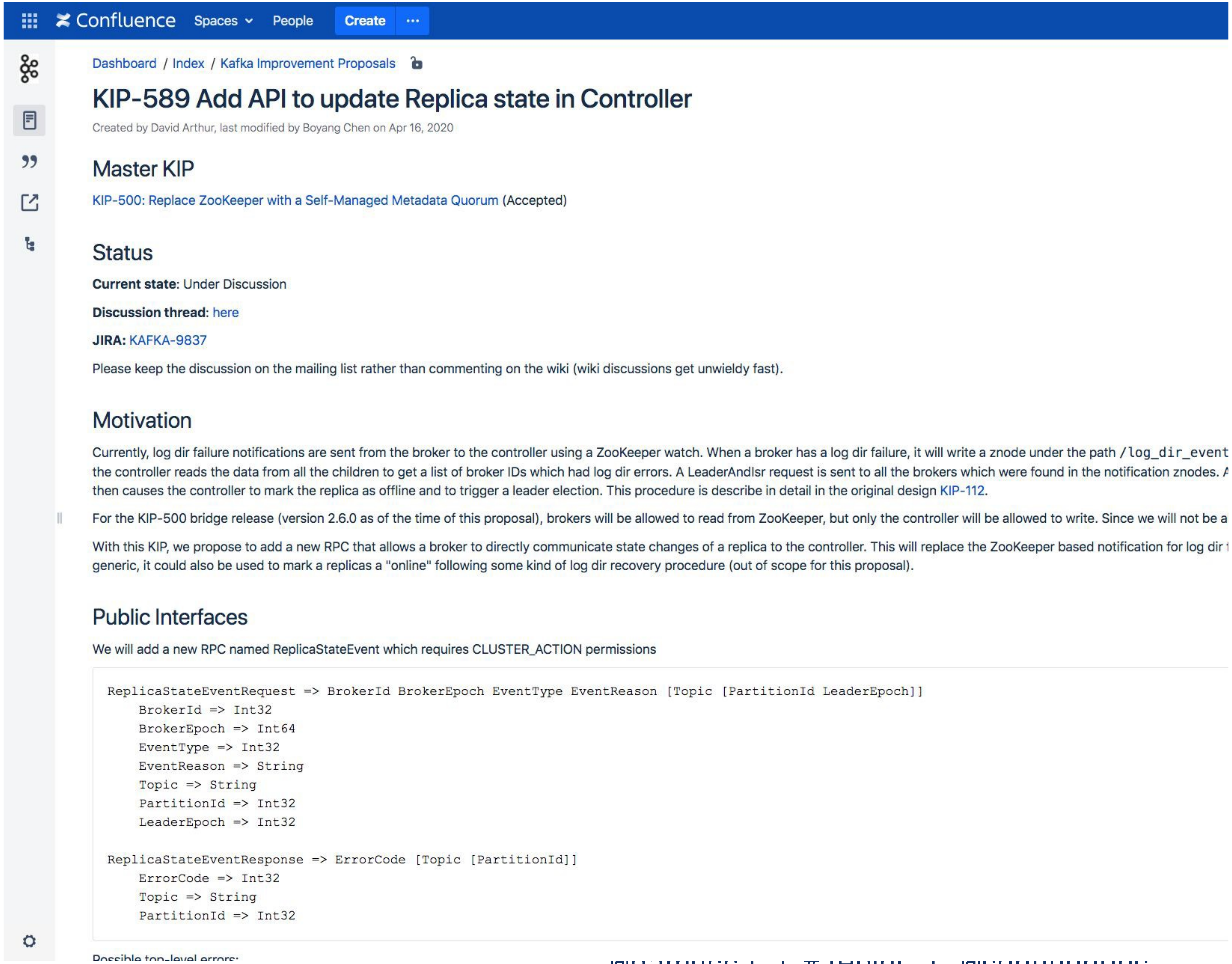
1. The controller creates the initial Leader and ISR on topic creation
2. The controller can shrink the ISR as part of a controlled shutdown or replica reassignment
3. The controller can elect new leaders at any time (e.g. preferred leader election)
4. Leaders can expand or shrink the ISR as followers come in and out of sync.

Since the znode can be modified by both the controller and partition leaders, care must be taken to protect updates. We use the zkVersion of the corresponding znode to protect updates, which means we need a r controllers propagate the zkVersion to leaders through the LeaderAndIsr request. Leaders, on the other hand, propagate the zkVersion to the controller by creating a sequential notification znode that the controller minute), which means Metadata is typically a bit slow to reflect ISR changes made by the leader.

In this KIP, we propose a new AlterIsr API to replace the notification znode in order to give the controller the exclusive ability to update Leader and ISR state. Leaders will use this API to request an ISR change from t controller will always have the latest Leader and ISR state for all partitions. Concretely, this has the following benefits:

- It will not be possible for the controller to send stale metadata through the LeaderAndIsr and UpdateMetadata APIs. New requests will always reflect the latest state.
- The controller can reject inconsistent leader and ISR changes. For example, if the controller sees a broker as offline, it can refuse to add it back to the ISR even though the leader still sees the follower fetchin
- When updating leader and ISR state, it won't be necessary to reinitialize current state (see KAFKA-8585). Preliminary testing shows this can cut controlled shutdown time down by as much as 40% (take this
- Partition reassignments complete only when new replicas are added to the ISR. With this change, reassignments can complete sooner because the controller does not have to await change notification.

Below we discuss the behavior of the new API in more detail.



Dashboard / Index / Kafka Improvement Proposals

# KIP-589 Add API to update Replica state in Controller

Created by David Arthur, last modified by Boyang Chen on Apr 16, 2020

## Master KIP

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

## Status

**Current state**: Under Discussion

**Discussion thread**: here

**JIRA:** KAFKA-9837

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

## Motivation

Currently, log dir failure notifications are sent from the broker to the controller using a ZooKeeper watch. When a broker has a log dir failure, it will write a znode under the path `/log_dir_event`
the controller reads the data from all the children to get a list of broker IDs which had log dir errors. A LeaderAndIsr request is sent to all the brokers which were found in the notification znodes. A
then causes the controller to mark the replica as offline and to trigger a leader election. This procedure is describe in detail in the original design KIP-112.

For the KIP-500 bridge release (version 2.6.0 as of the time of this proposal), brokers will be allowed to read from ZooKeeper, but only the controller will be allowed to write. Since we will not be a

With this KIP, we propose to add a new RPC that allows a broker to directly communicate state changes of a replica to the controller. This will replace the ZooKeeper based notification for log dir f
generic, it could also be used to mark a replicas a "online" following some kind of log dir recovery procedure (out of scope for this proposal).

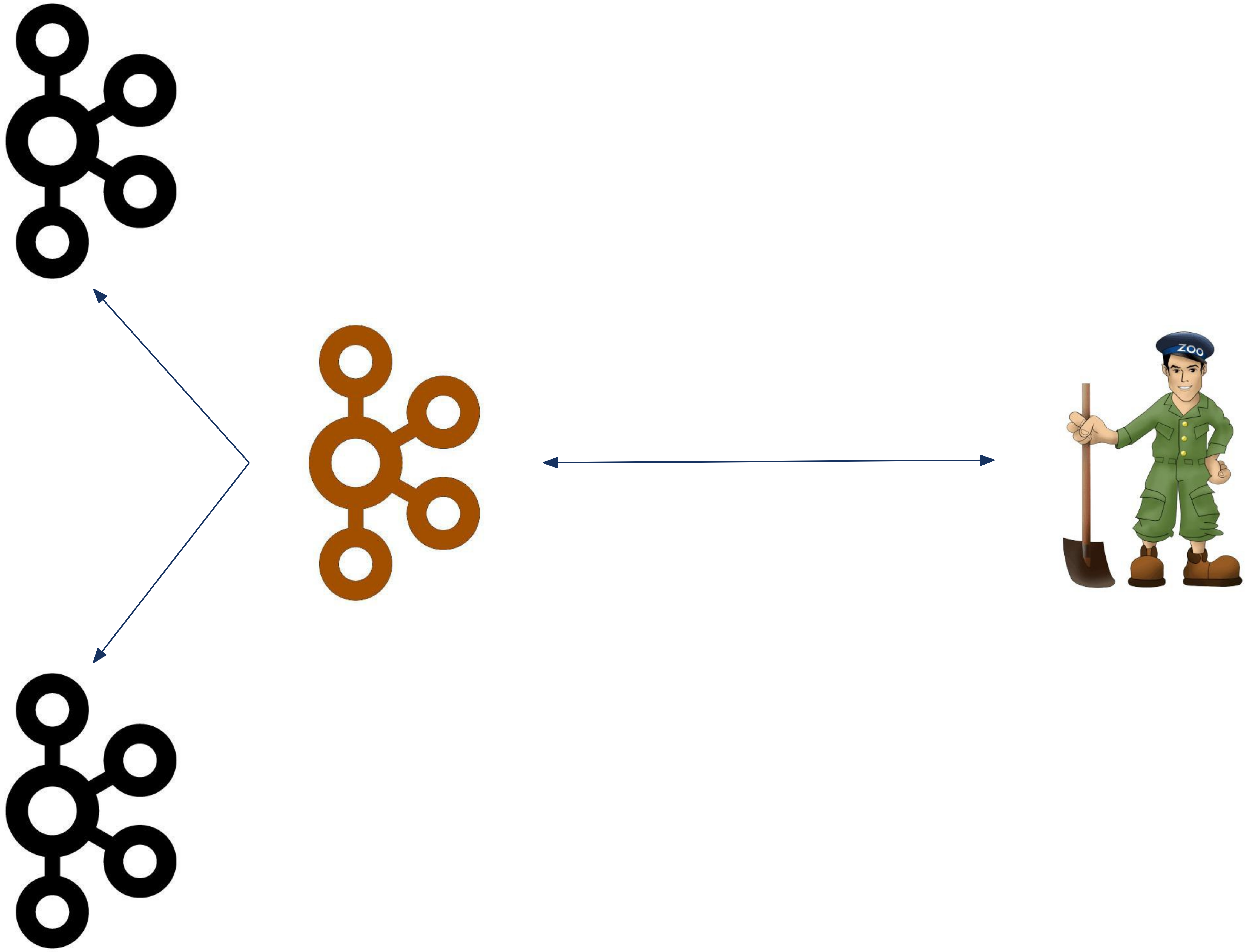
## Public Interfaces

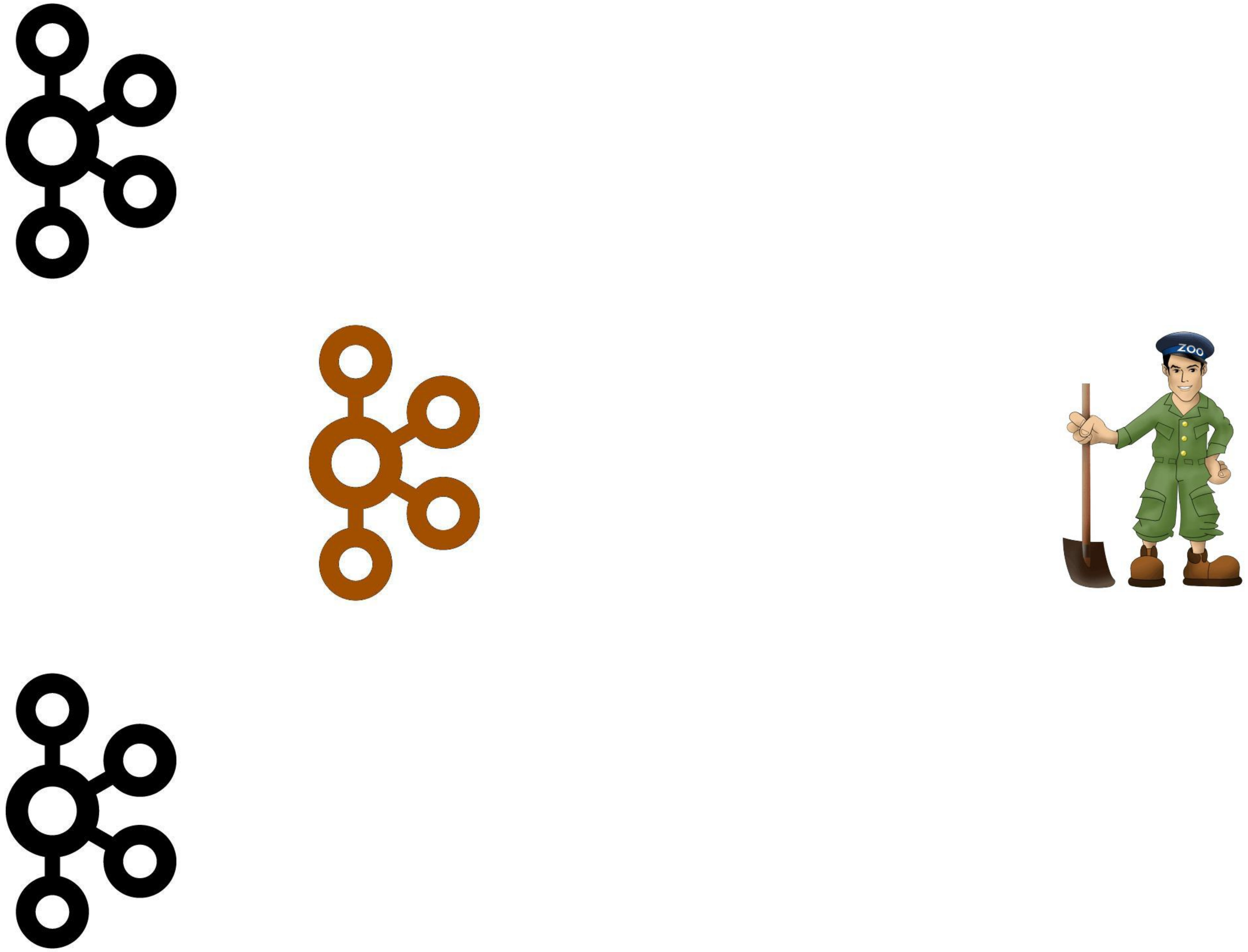
We will add a new RPC named ReplicaStateEvent which requires CLUSTER_ACTION permissions

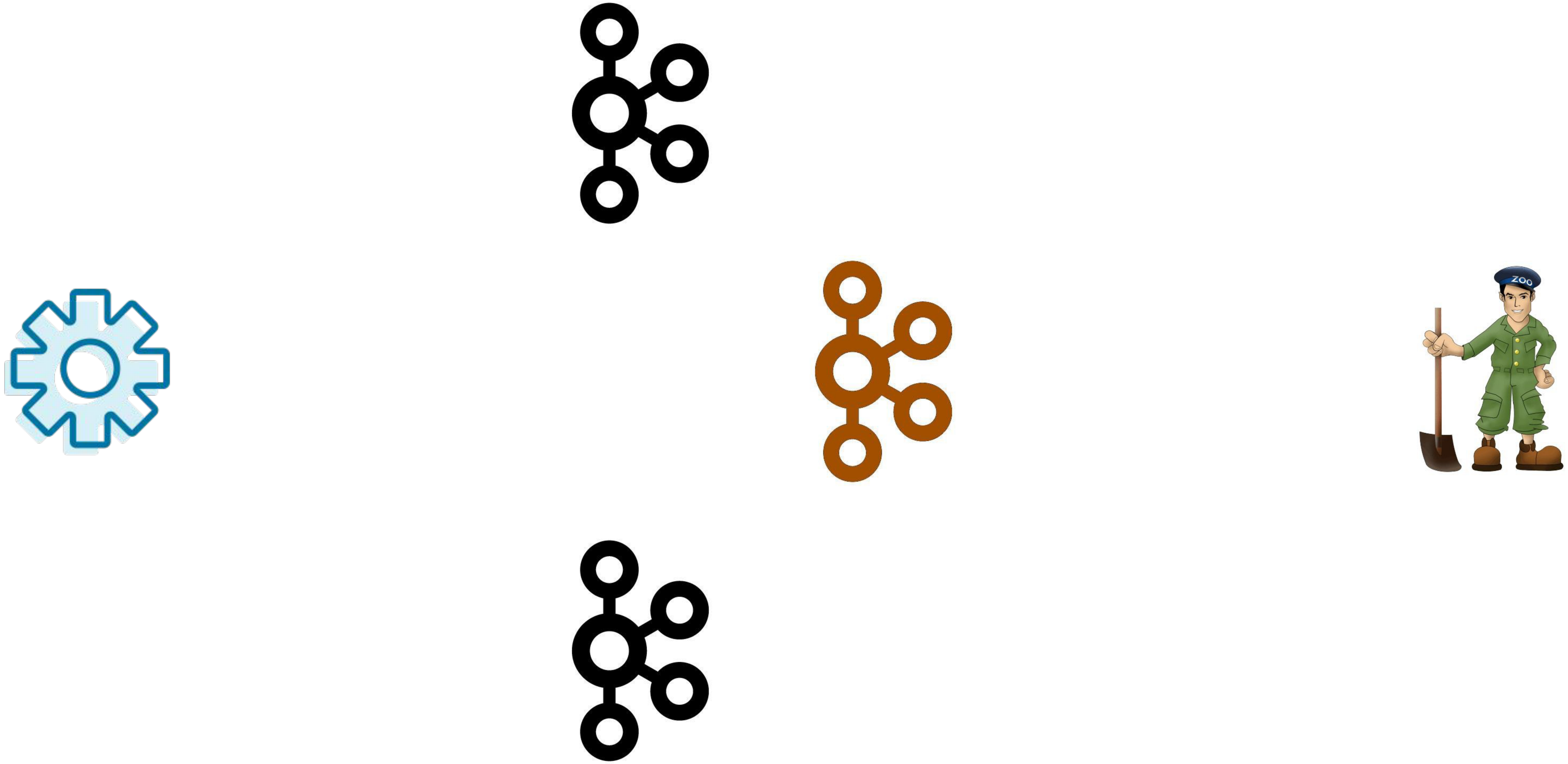
```
ReplicaStateEventRequest => BrokerId BrokerEpoch EventType EventReason [Topic [PartitionId LeaderEpoch]]
    BrokerId => Int32
    BrokerEpoch => Int64
    EventType => Int32
    EventReason => String
    Topic => String
    PartitionId => Int32
    LeaderEpoch => Int32

ReplicaStateEventResponse => ErrorCode [Topic [PartitionId]]
    ErrorCode => Int32
    Topic => String
    PartitionId => Int32
```

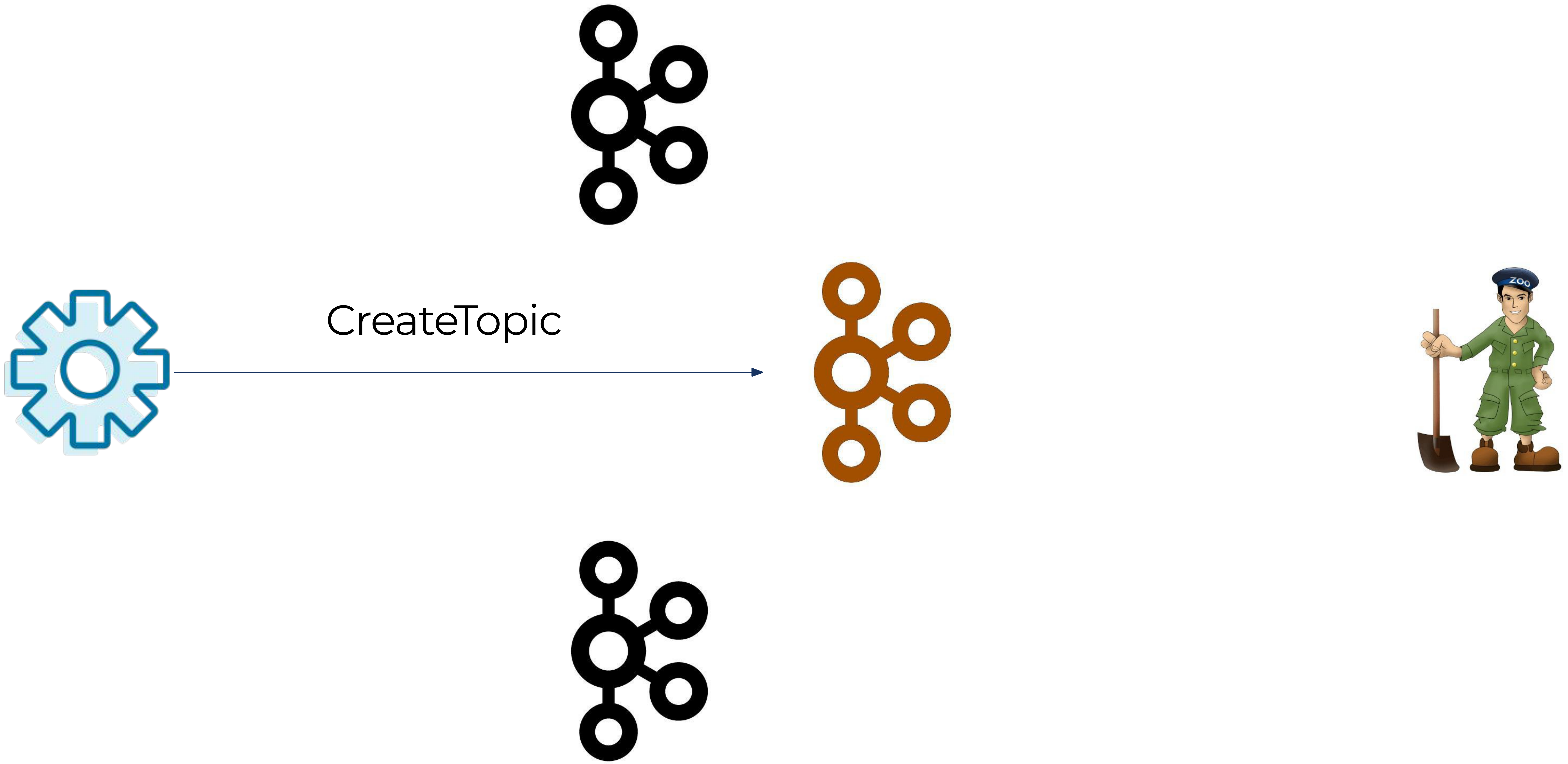
Possible top-level errors:

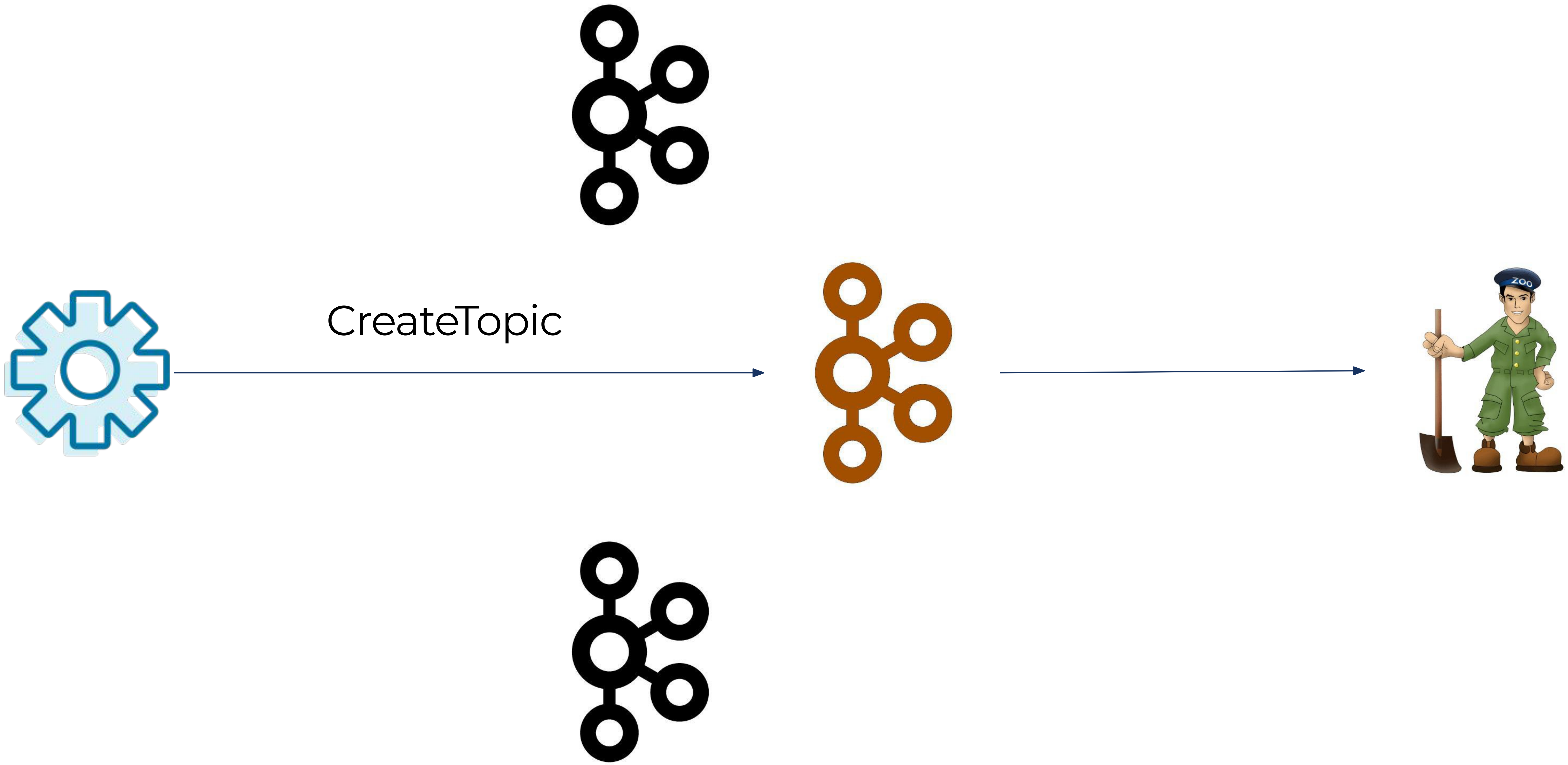
# Client Compatibility

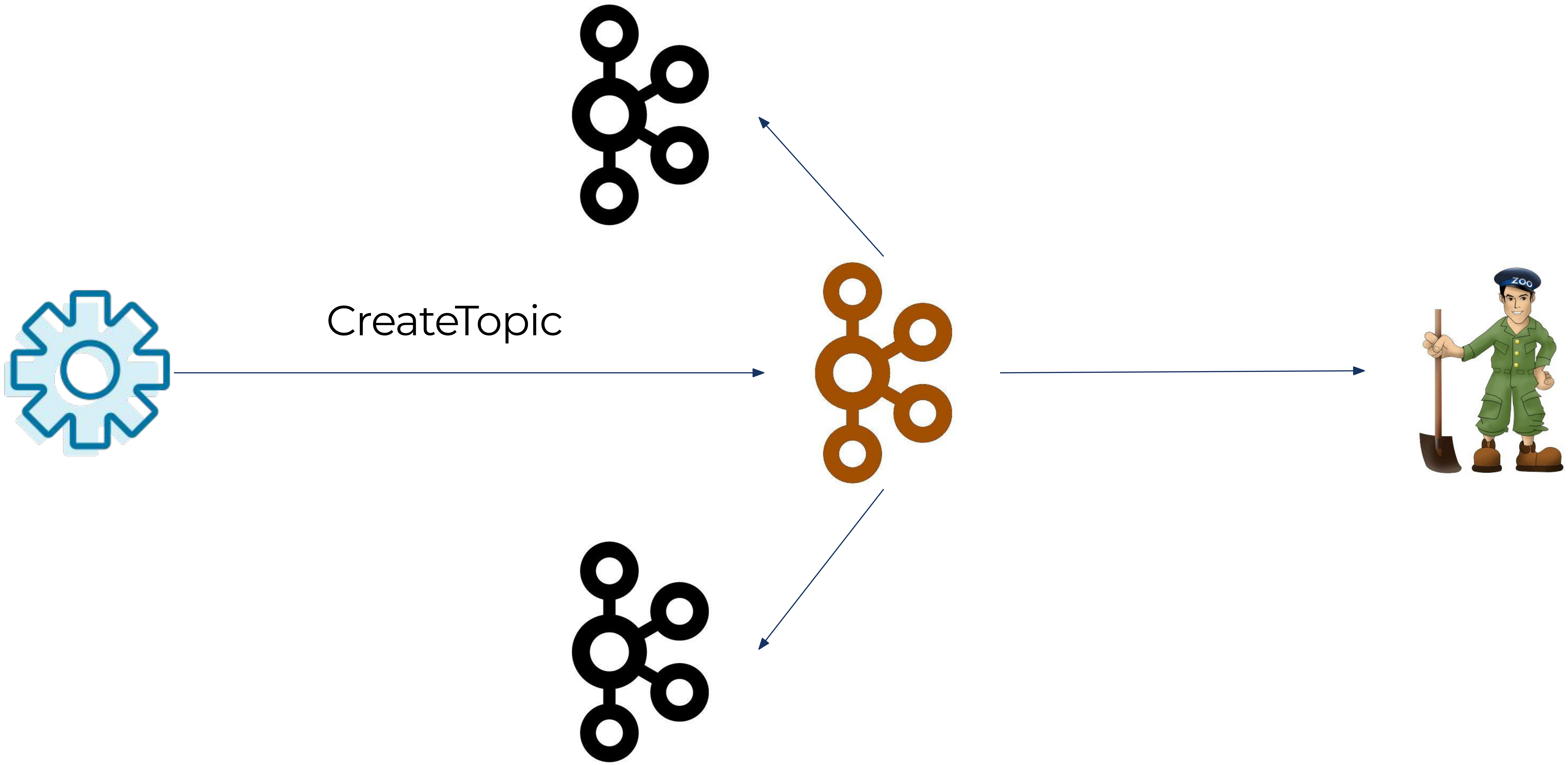

ZOO

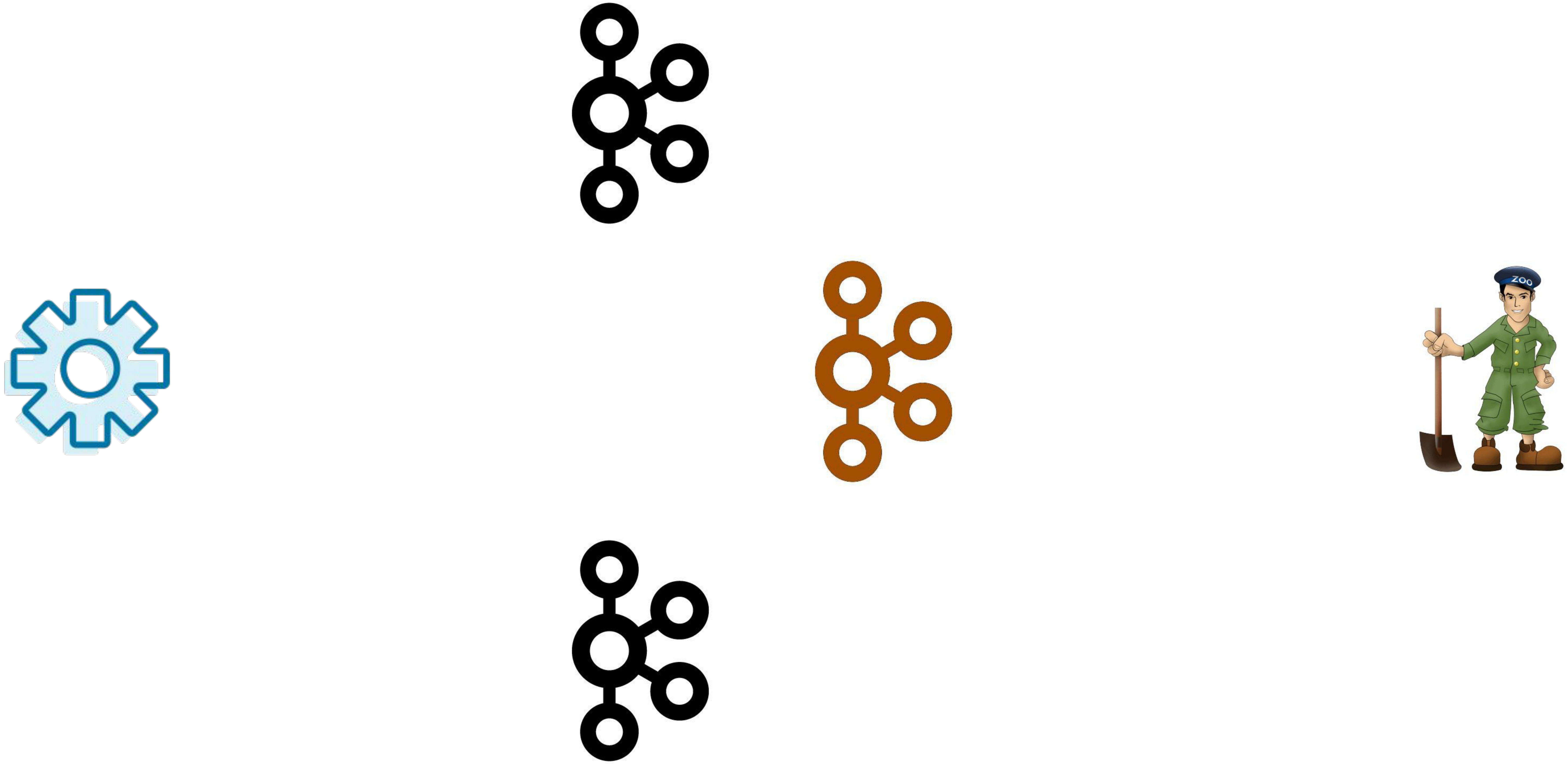

ZOO

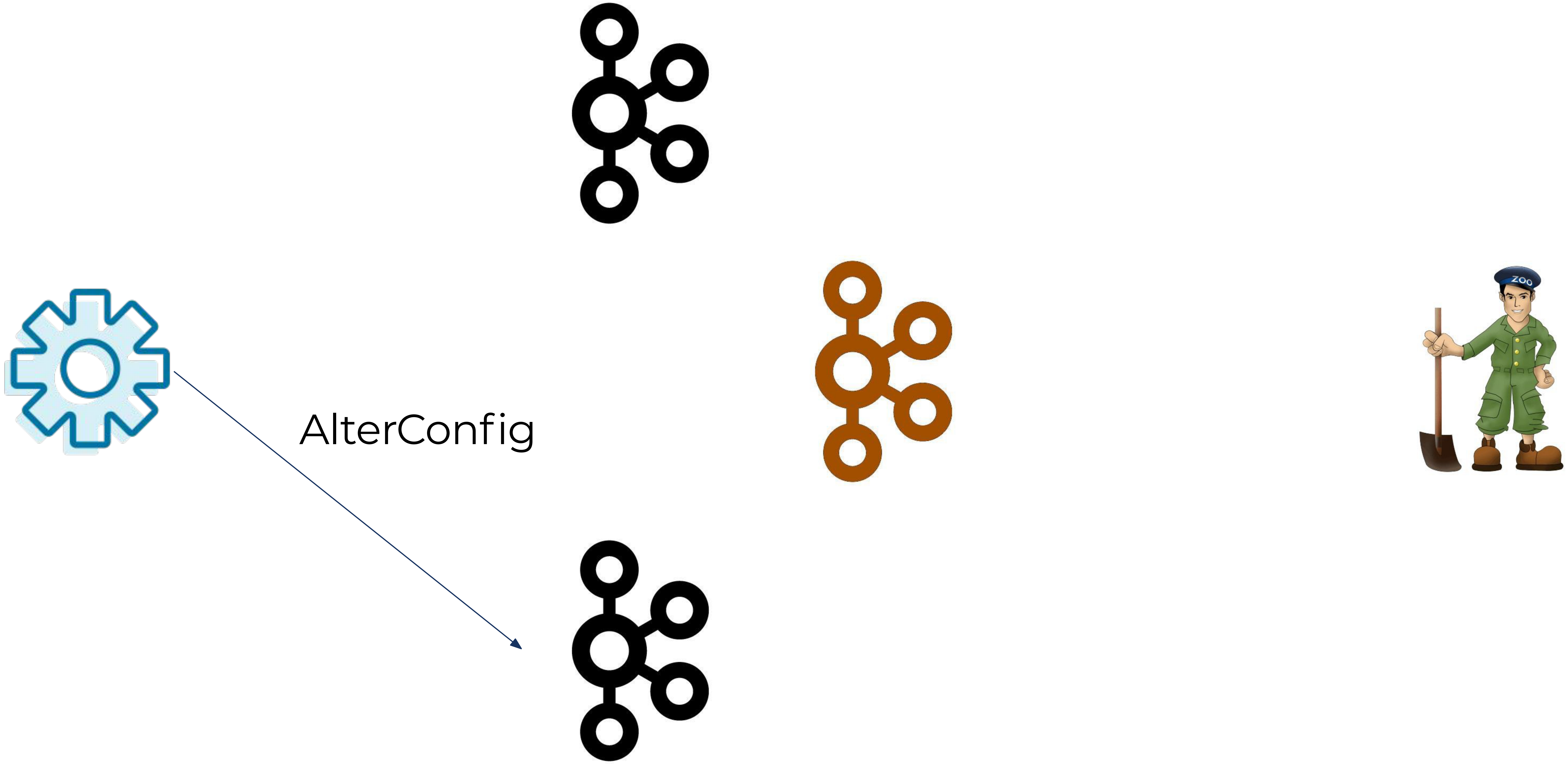


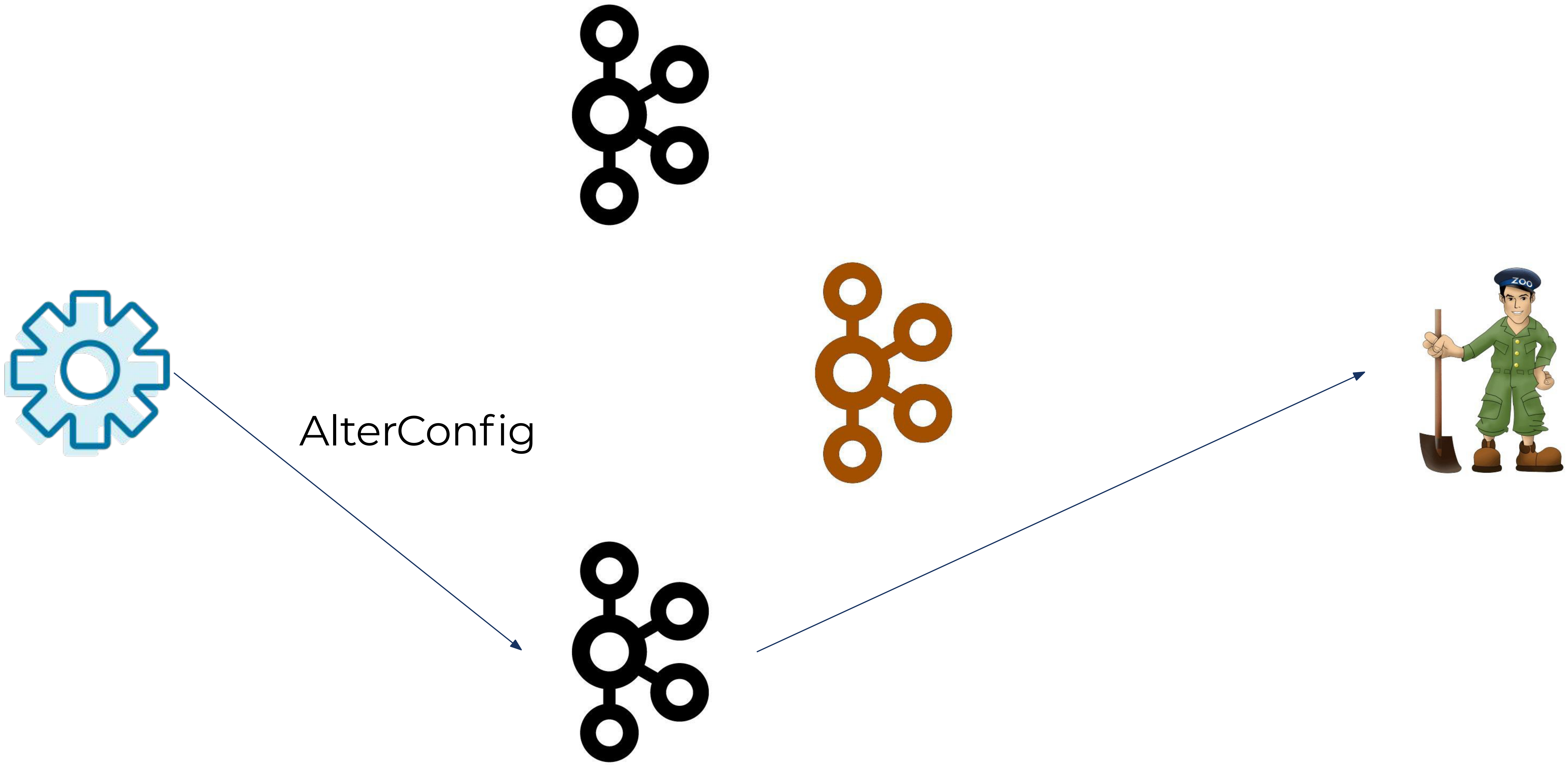

CreateTopic

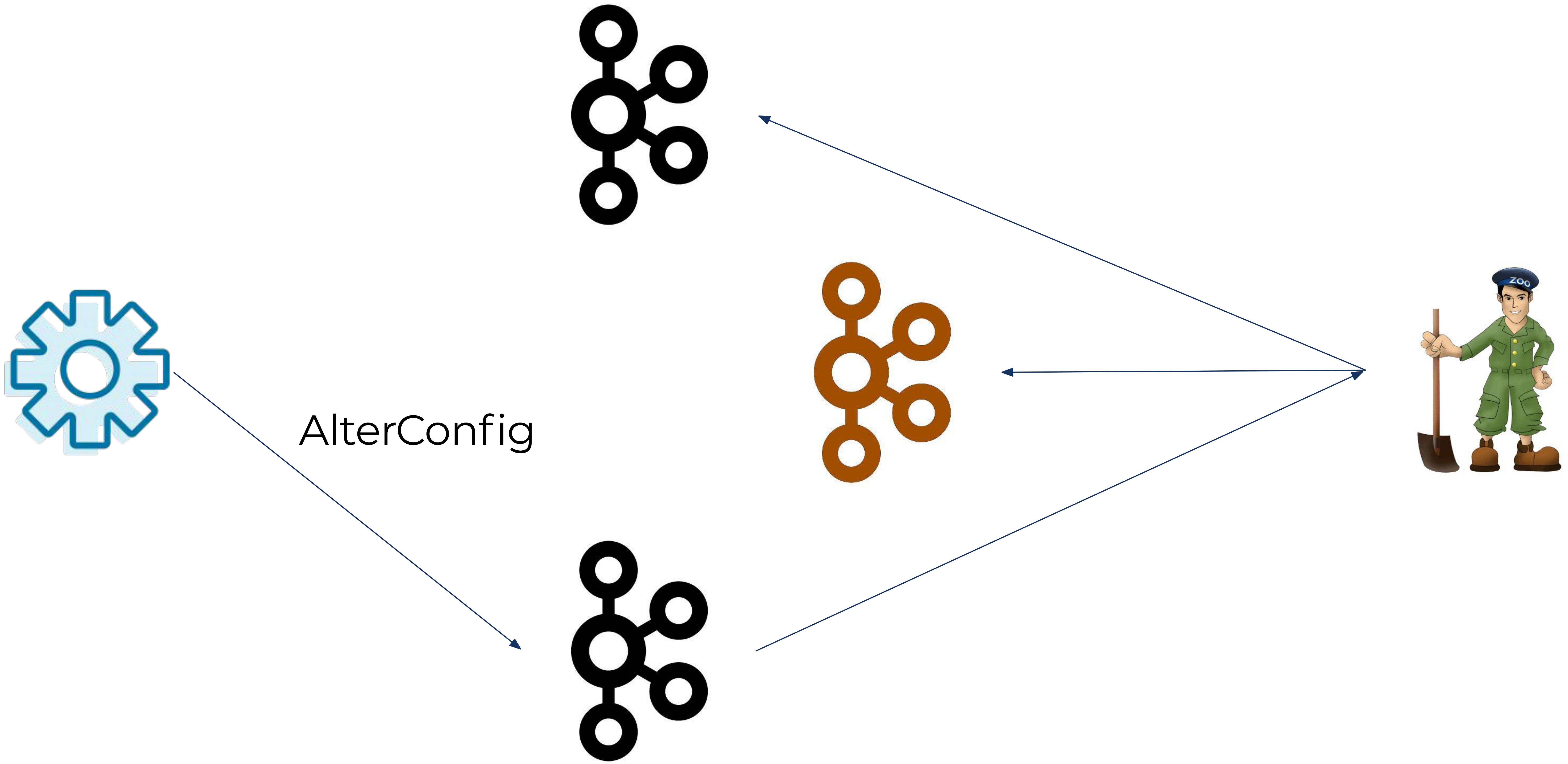

CreateTopic
ZOO

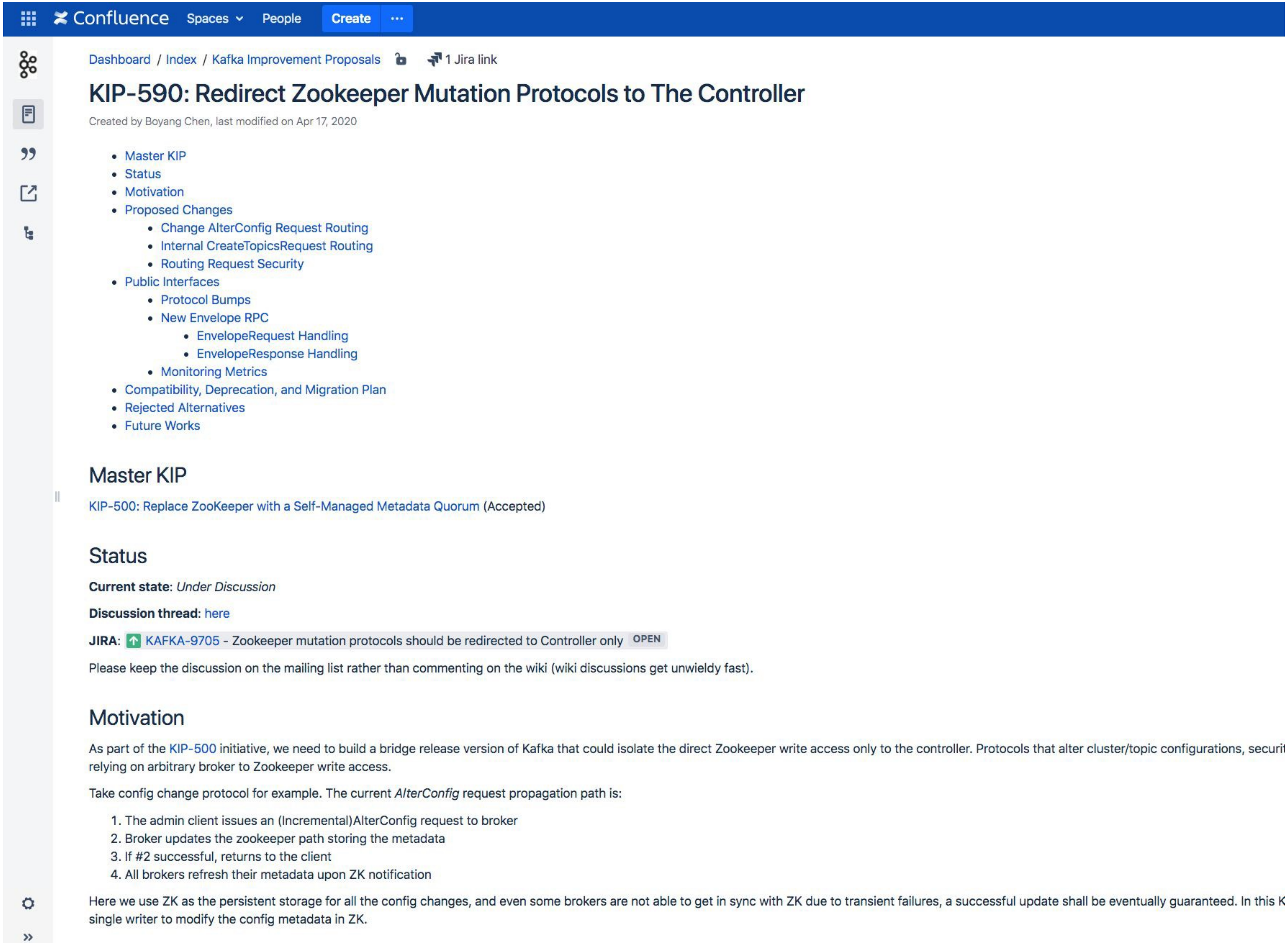

CreateTopic


ZOO


AlterConfig
ZOO


AlterConfig
ZOO


AlterConfig
ZOO



# KIP-590: Redirect Zookeeper Mutation Protocols to The Controller

Created by Boyang Chen, last modified on Apr 17, 2020

## Master KIP

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

## Status

**Current state**: *Under Discussion*

**Discussion thread**: here

**JIRA**: KAFKA-9705 - Zookeeper mutation protocols should be redirected to Controller only OPEN

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

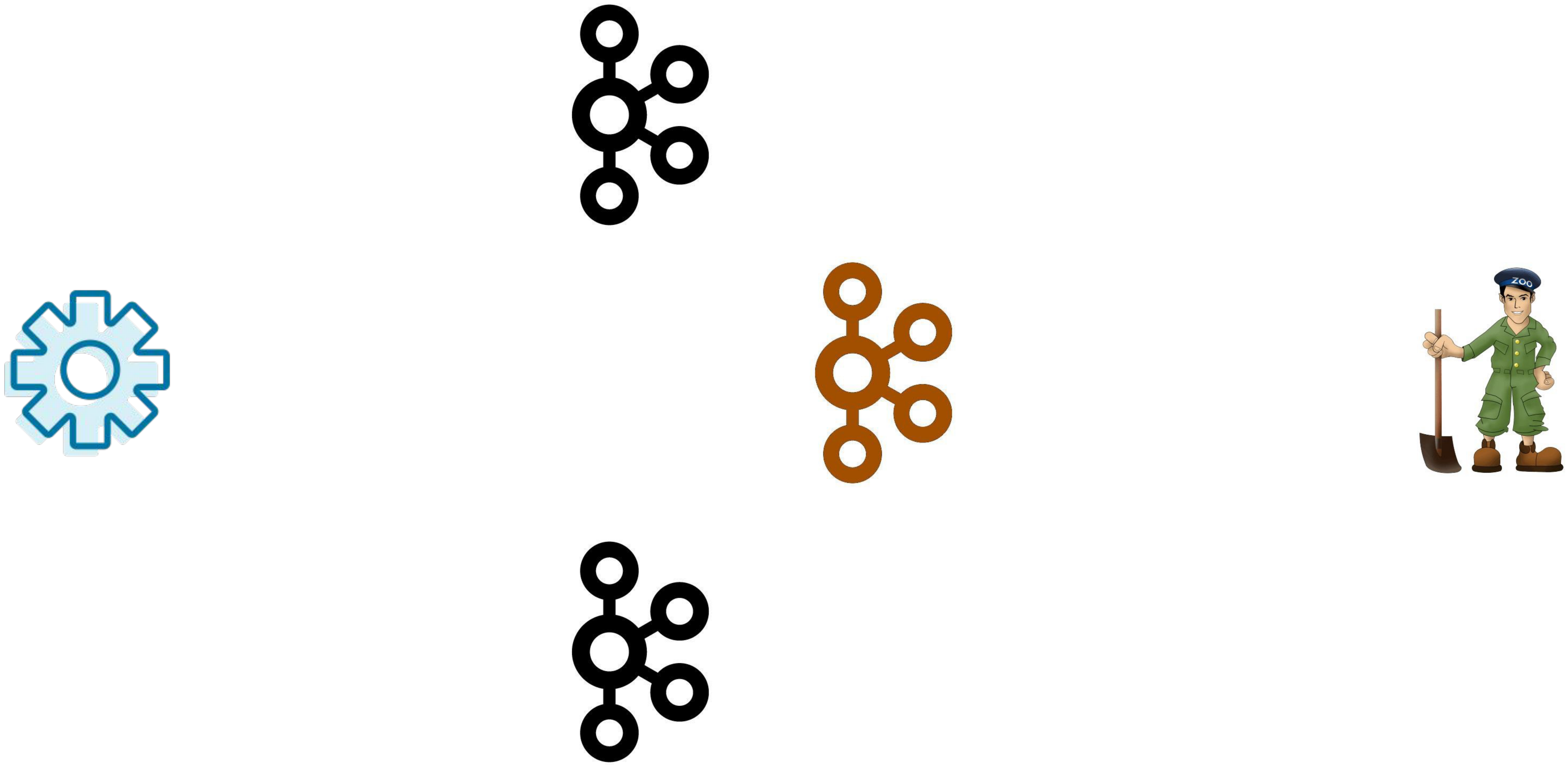
## Motivation

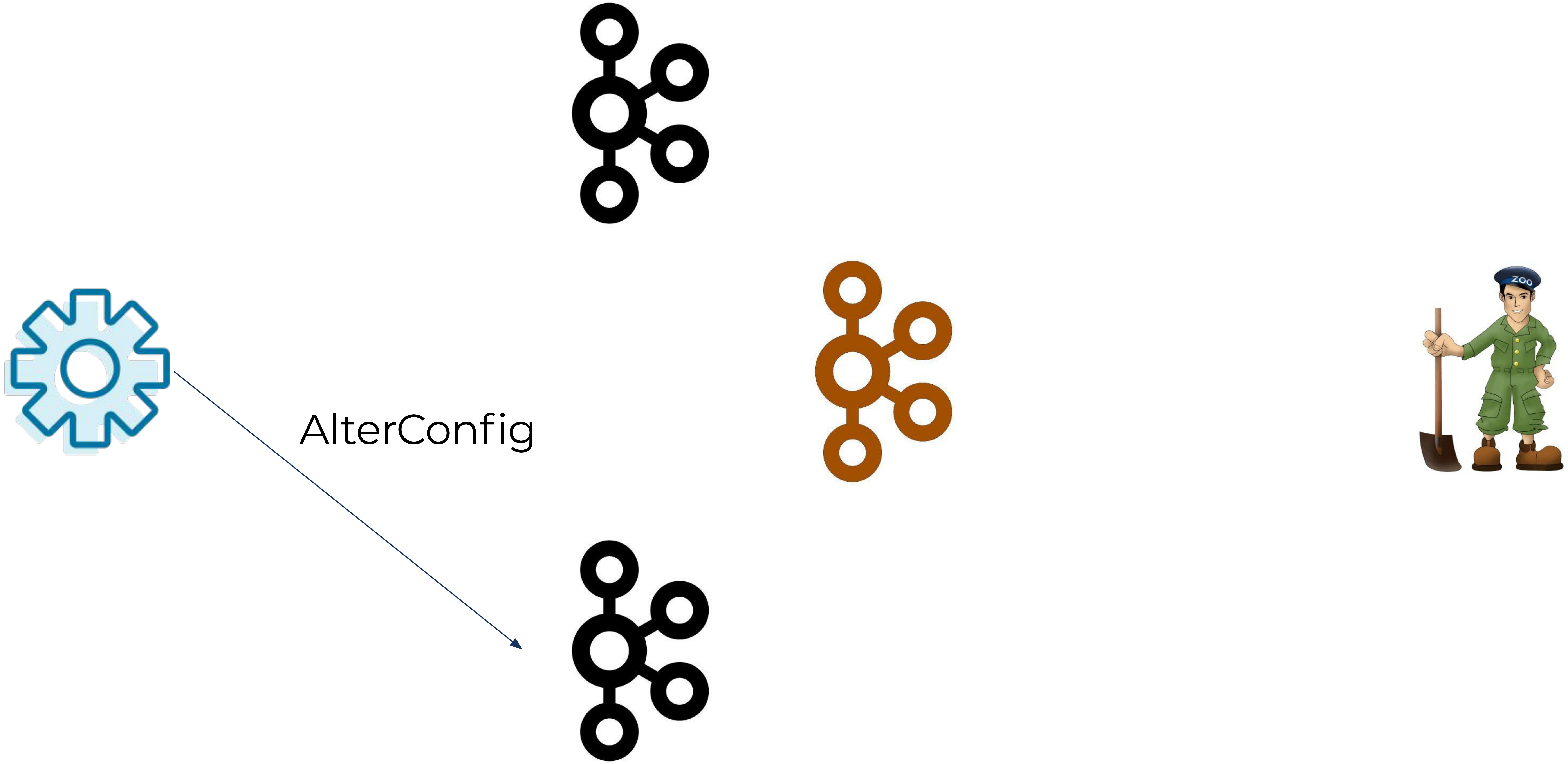
As part of the KIP-500 initiative, we need to build a bridge release version of Kafka that could isolate the direct Zookeeper write access only to the controller. Protocols that alter cluster/topic configurations, securit
relying on arbitrary broker to Zookeeper write access.

Take config change protocol for example. The current *AlterConfig* request propagation path is:

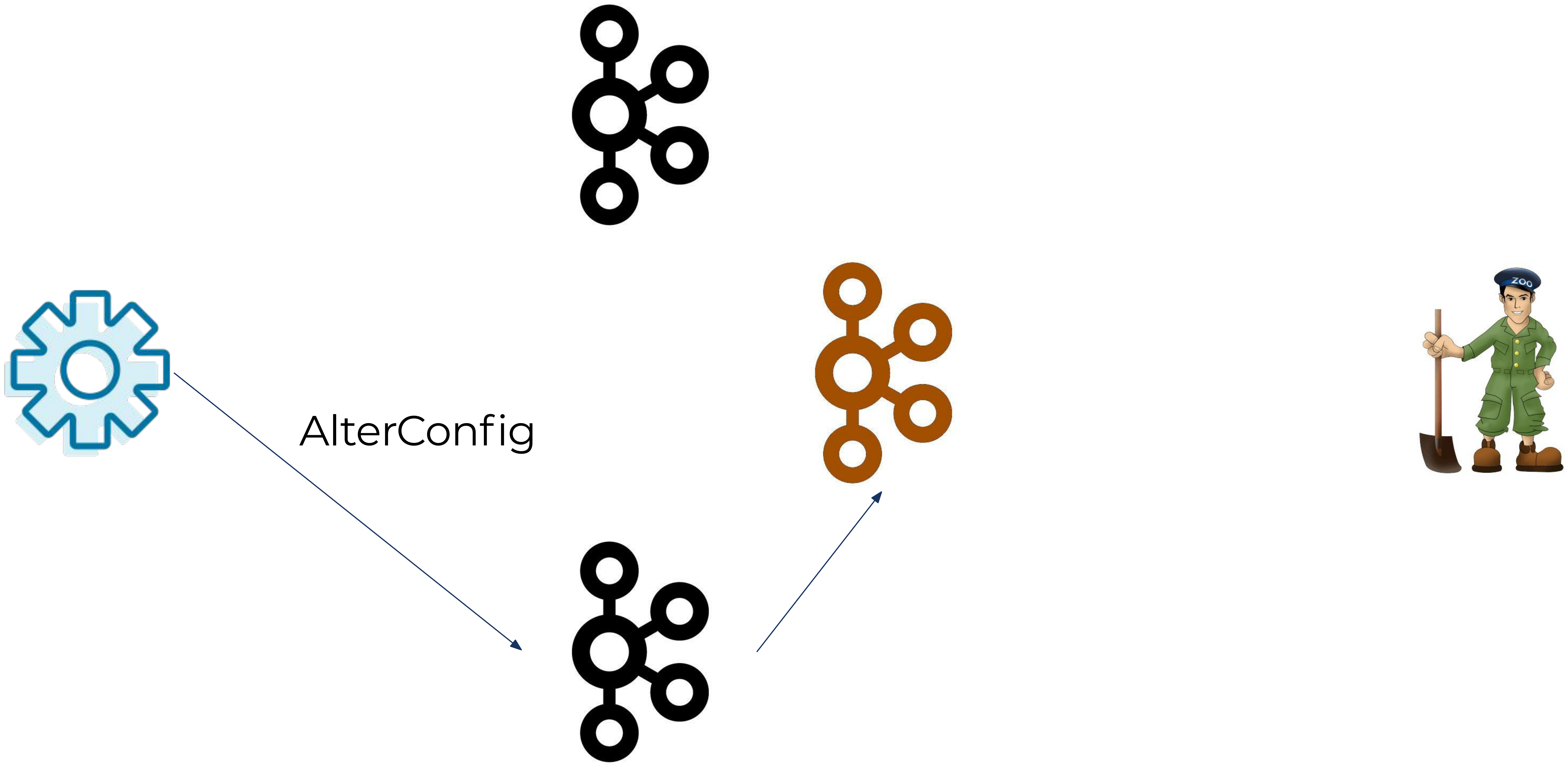
1. The admin client issues an (Incremental)AlterConfig request to broker
2. Broker updates the zookeeper path storing the metadata
3. If #2 successful, returns to the client
4. All brokers refresh their metadata upon ZK notification

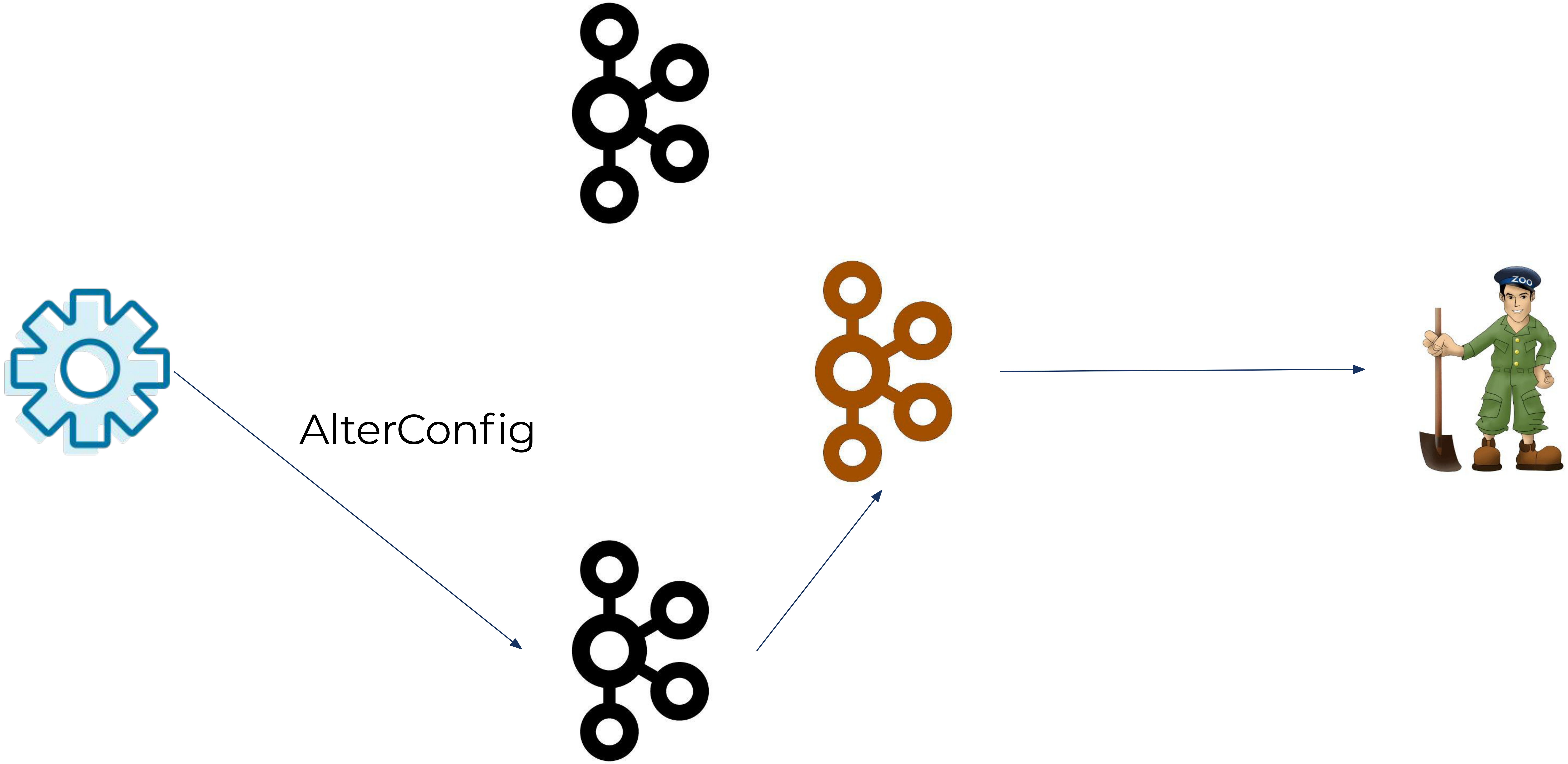
Here we use ZK as the persistent storage for all the config changes, and even some brokers are not able to get in sync with ZK due to transient failures, a successful update shall be eventually guaranteed. In this K
single writer to modify the config metadata in ZK.

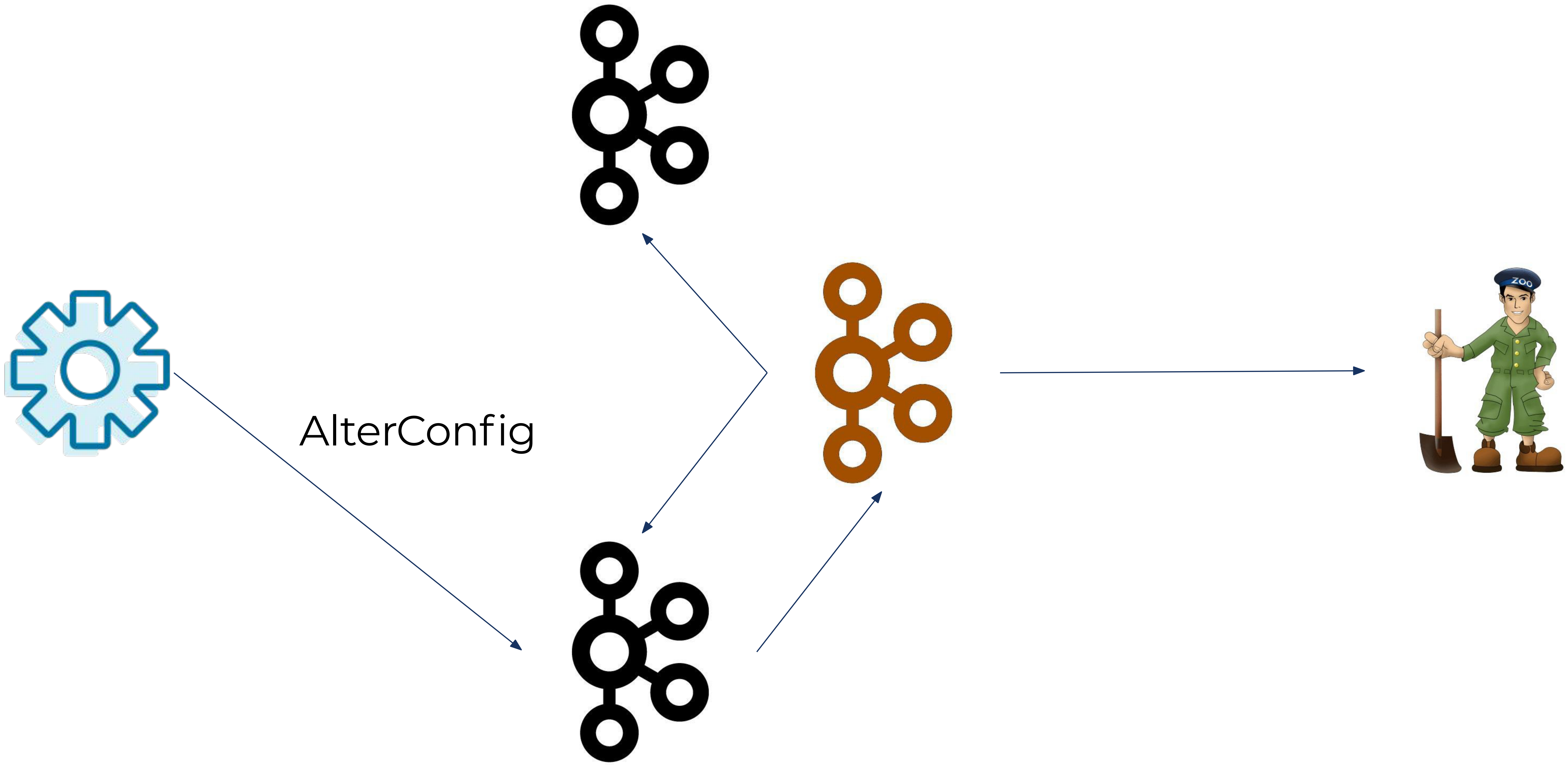

ZOO



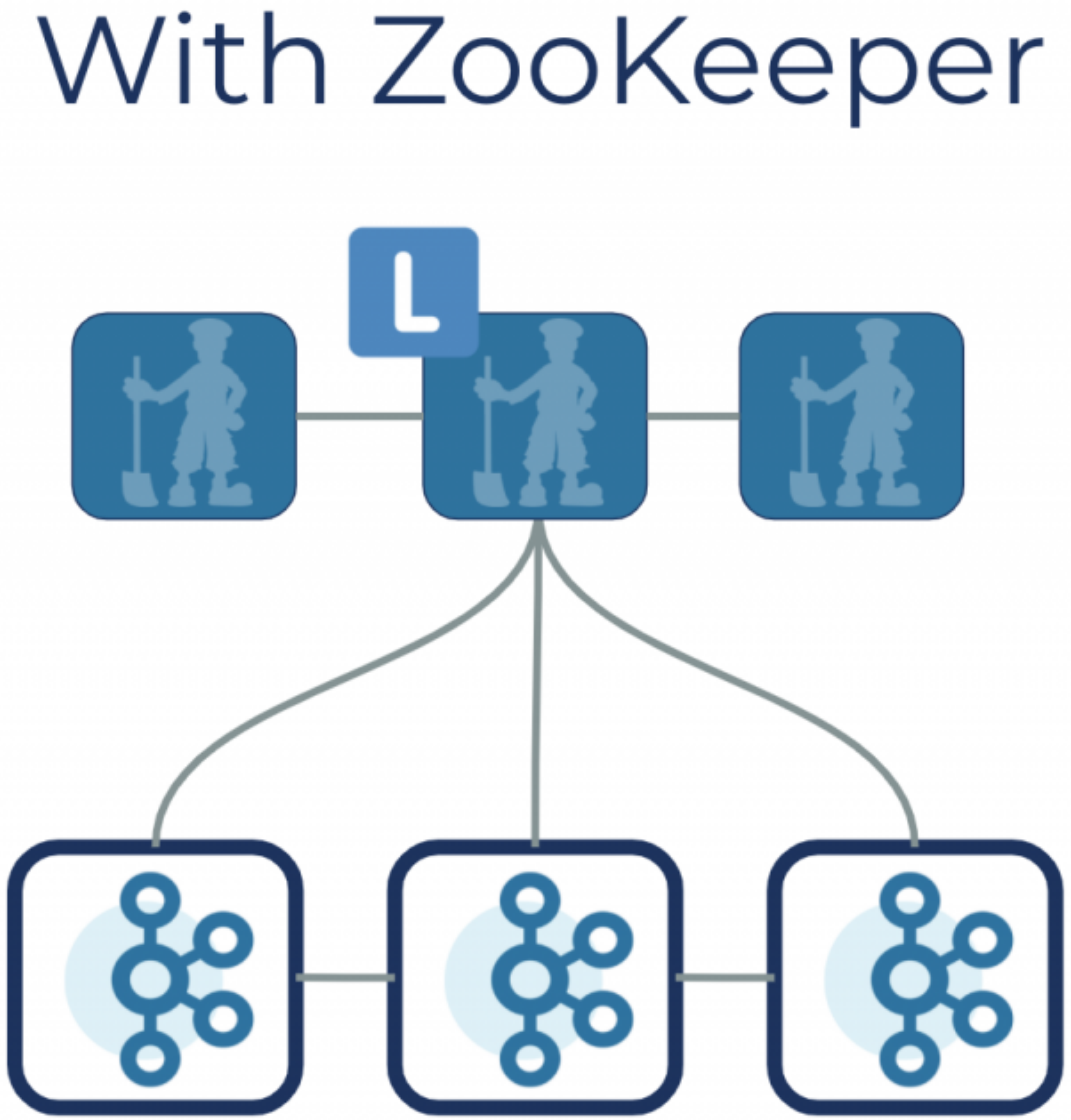
AlterConfig

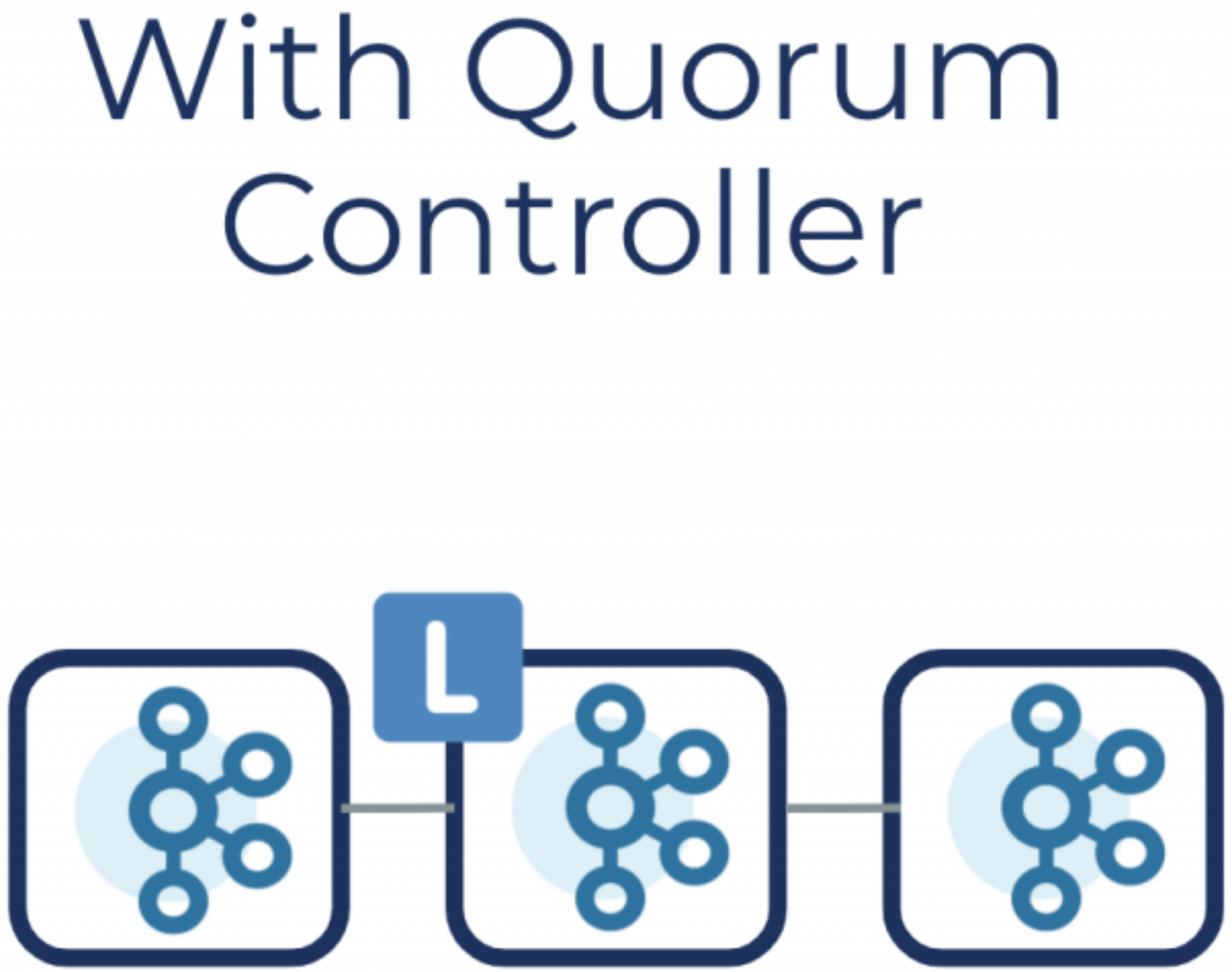

AlterConfig
ZOO


AlterConfig


AlterConfig

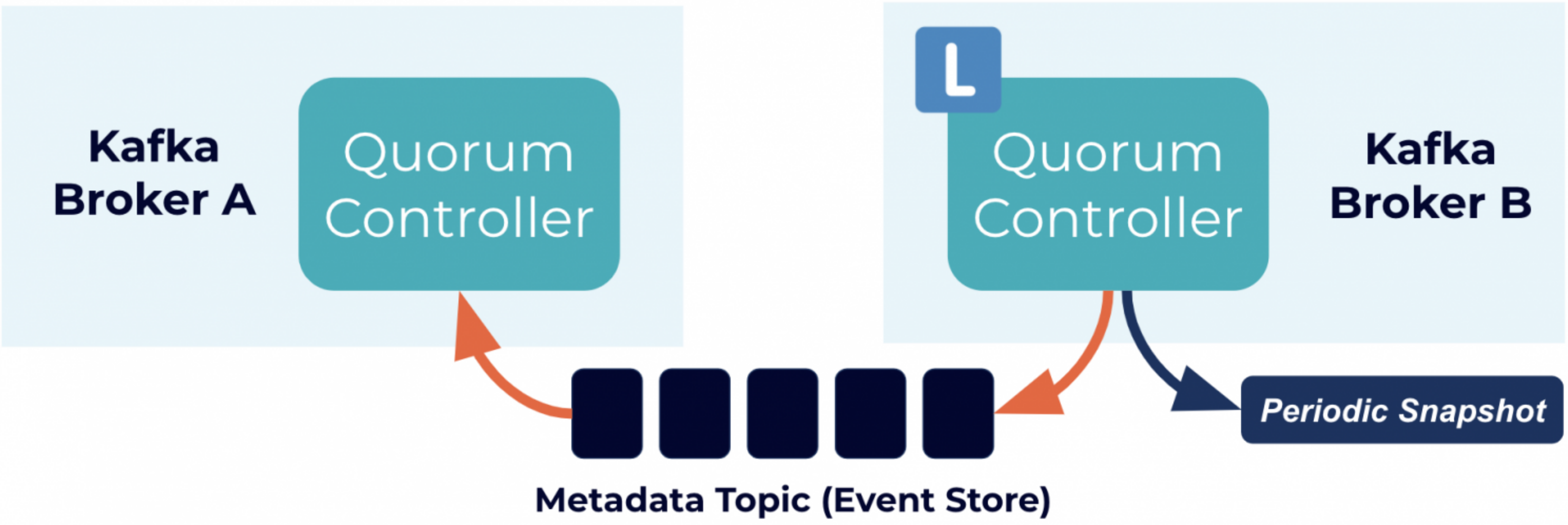
# KIP-500: Current State

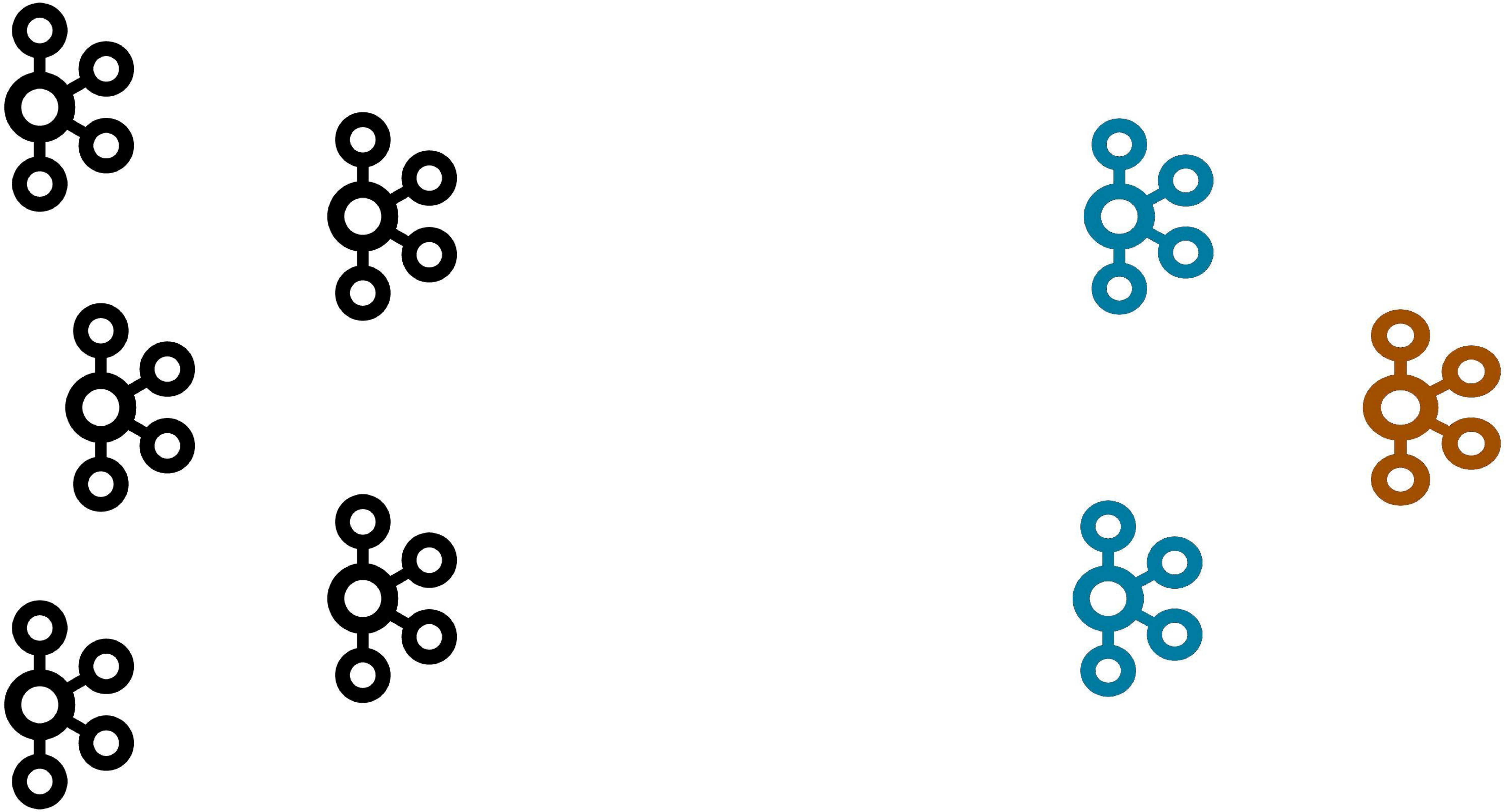


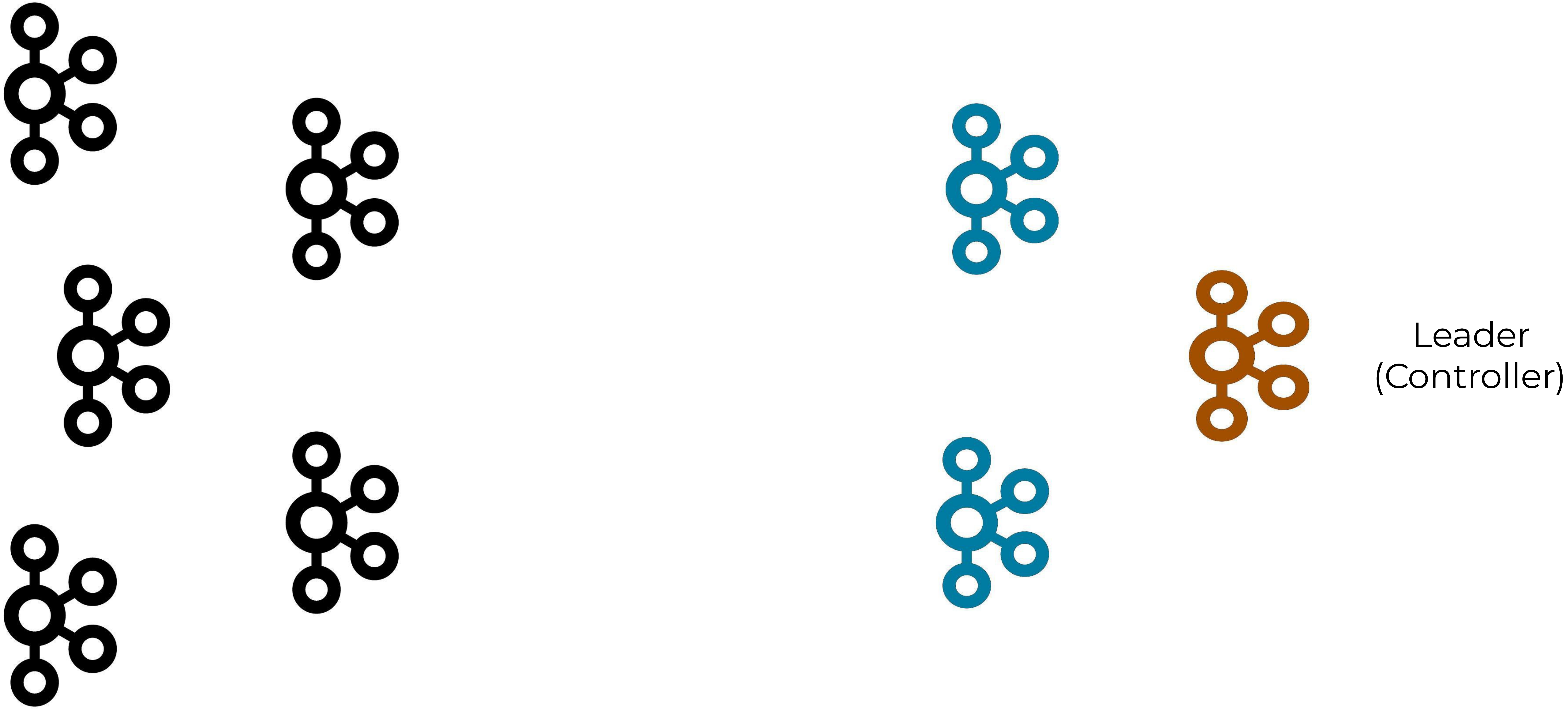


L denotes quorum leader

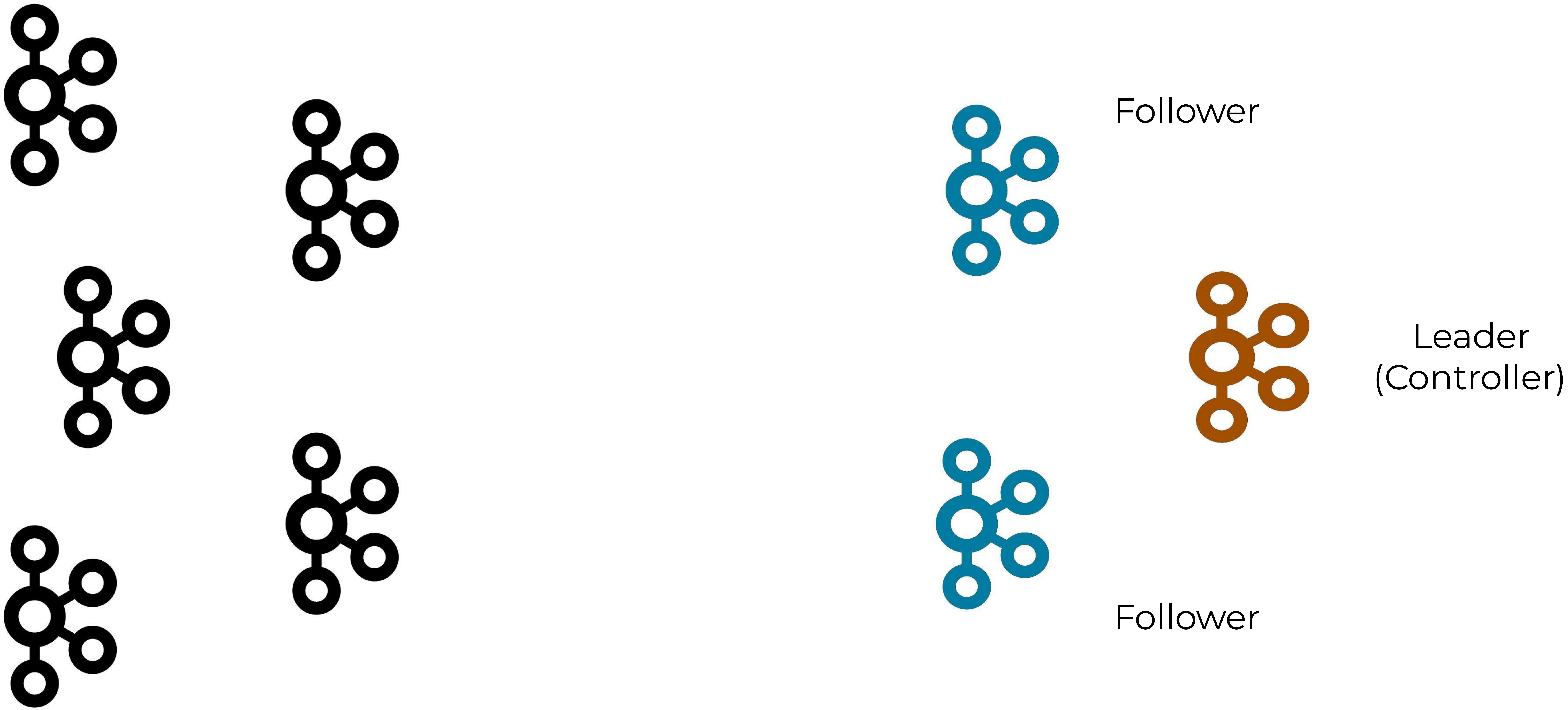
# Metadata Quorum


Kafka Broker A
Quorum Controller
L
Quorum Controller
Kafka Broker B
Periodic Snapshot
Metadata Topic (Event Store)





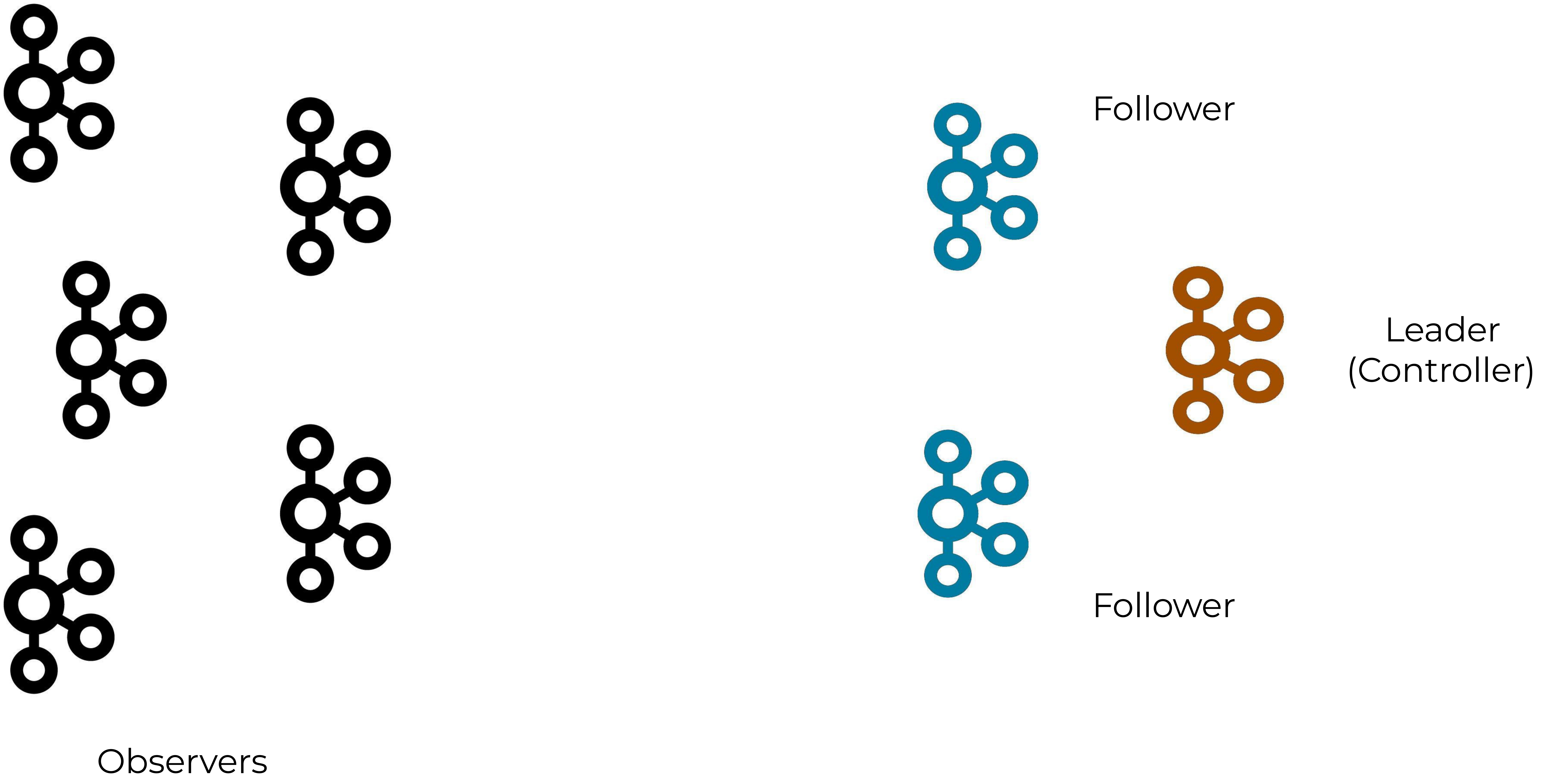
Leader
(Controller)

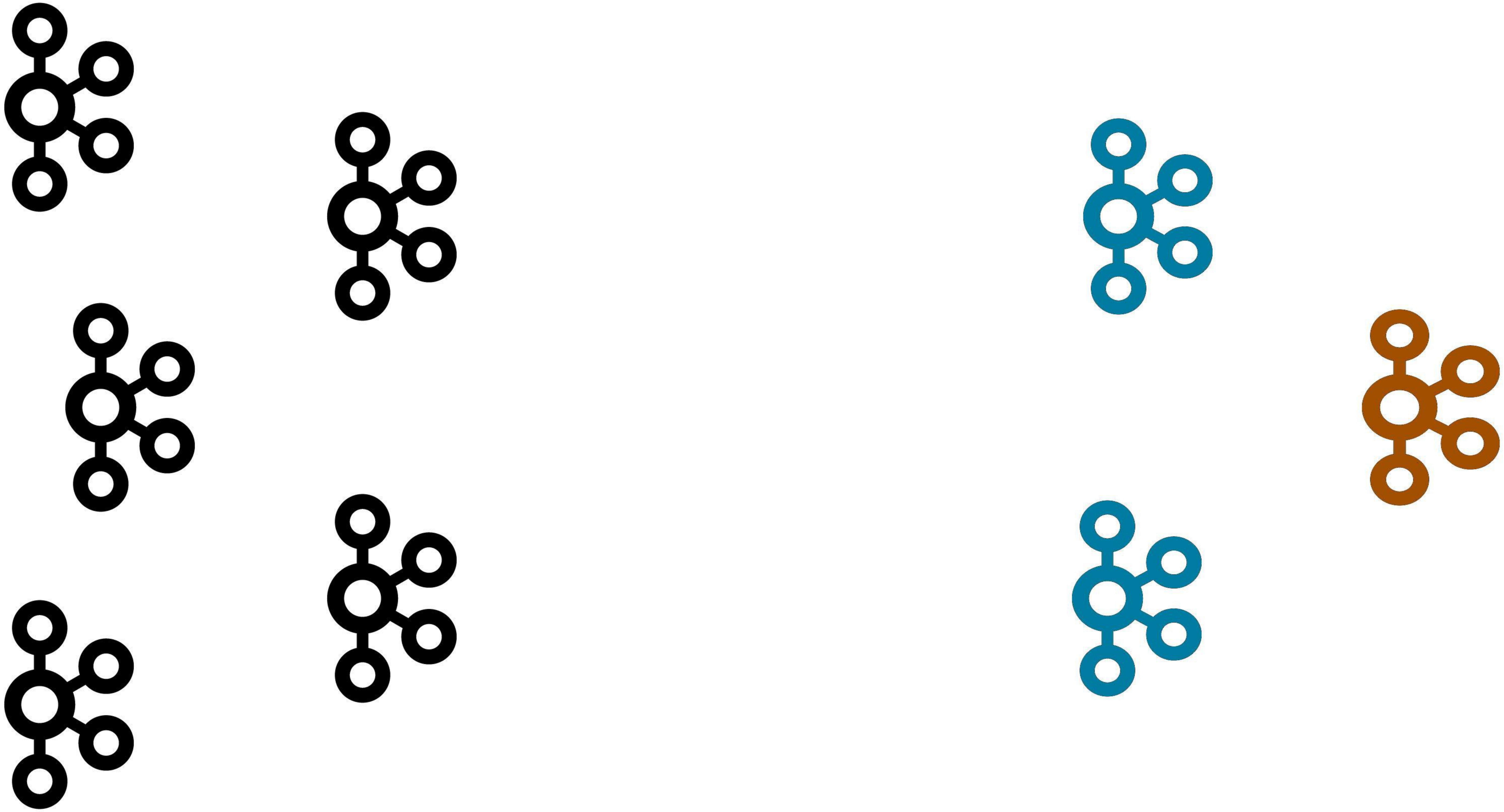


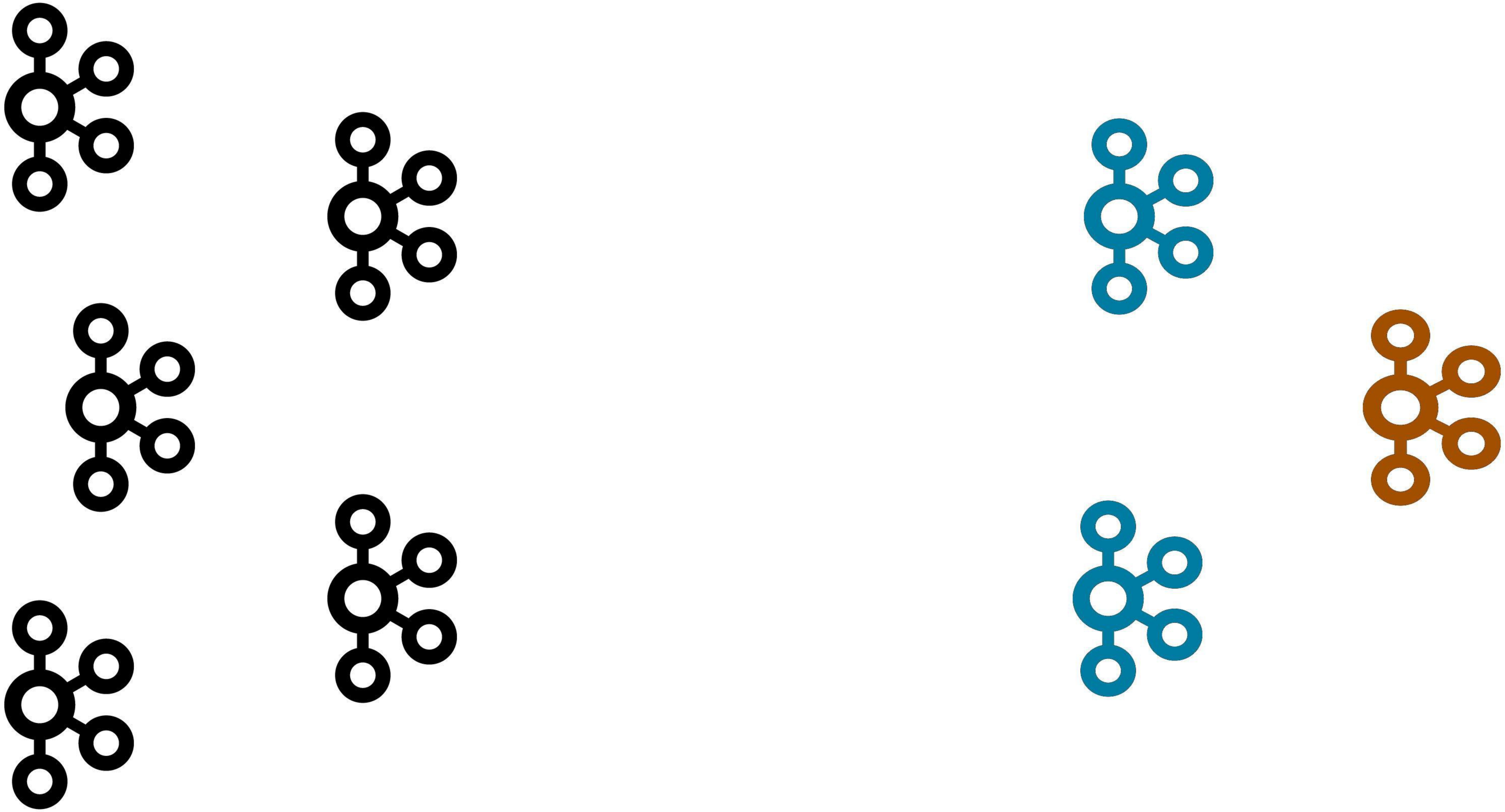
Follower

Leader
(Controller)

Follower


Follower
Leader
(Controller)
Follower
Observers

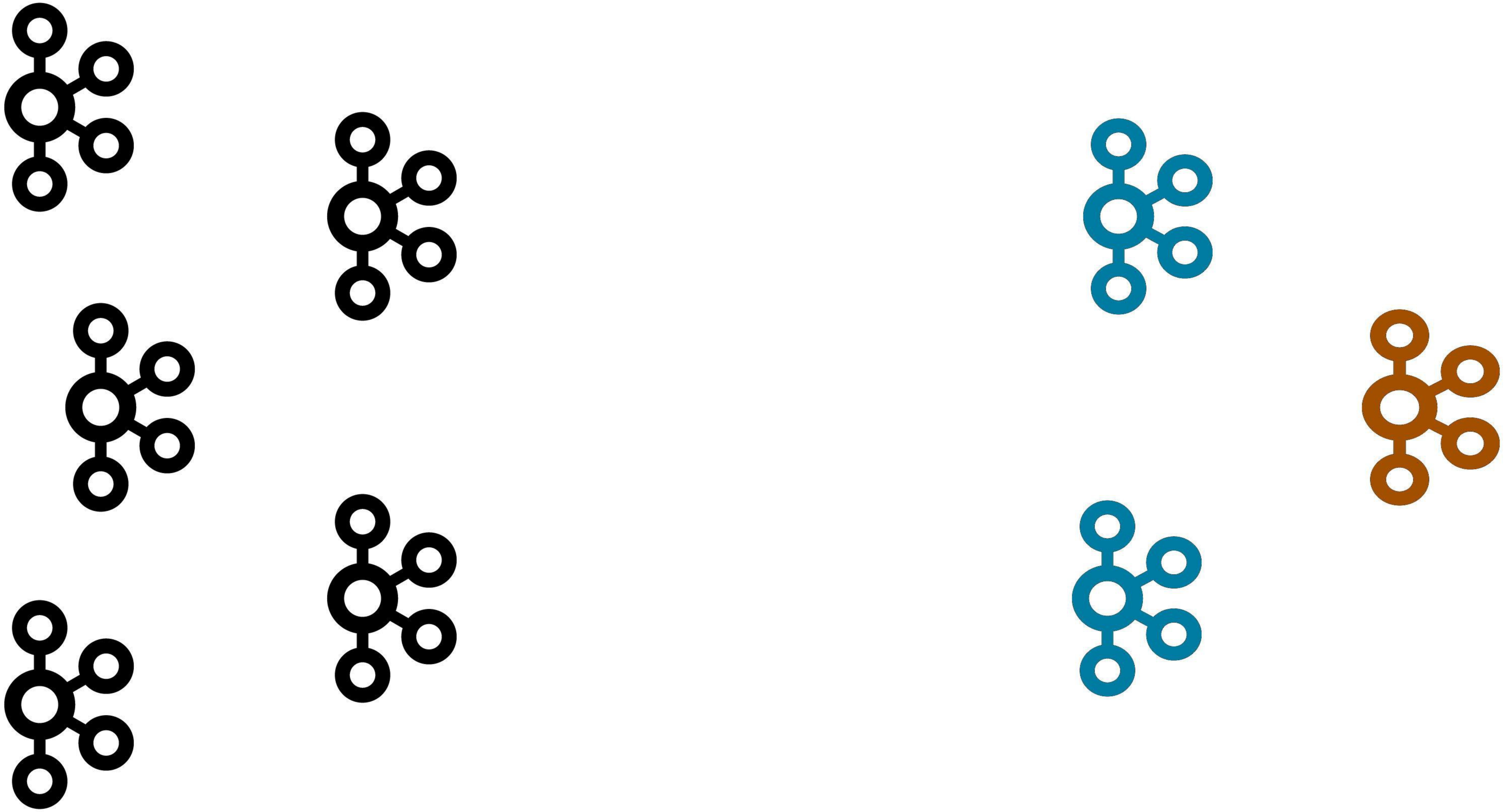




Data Nodes

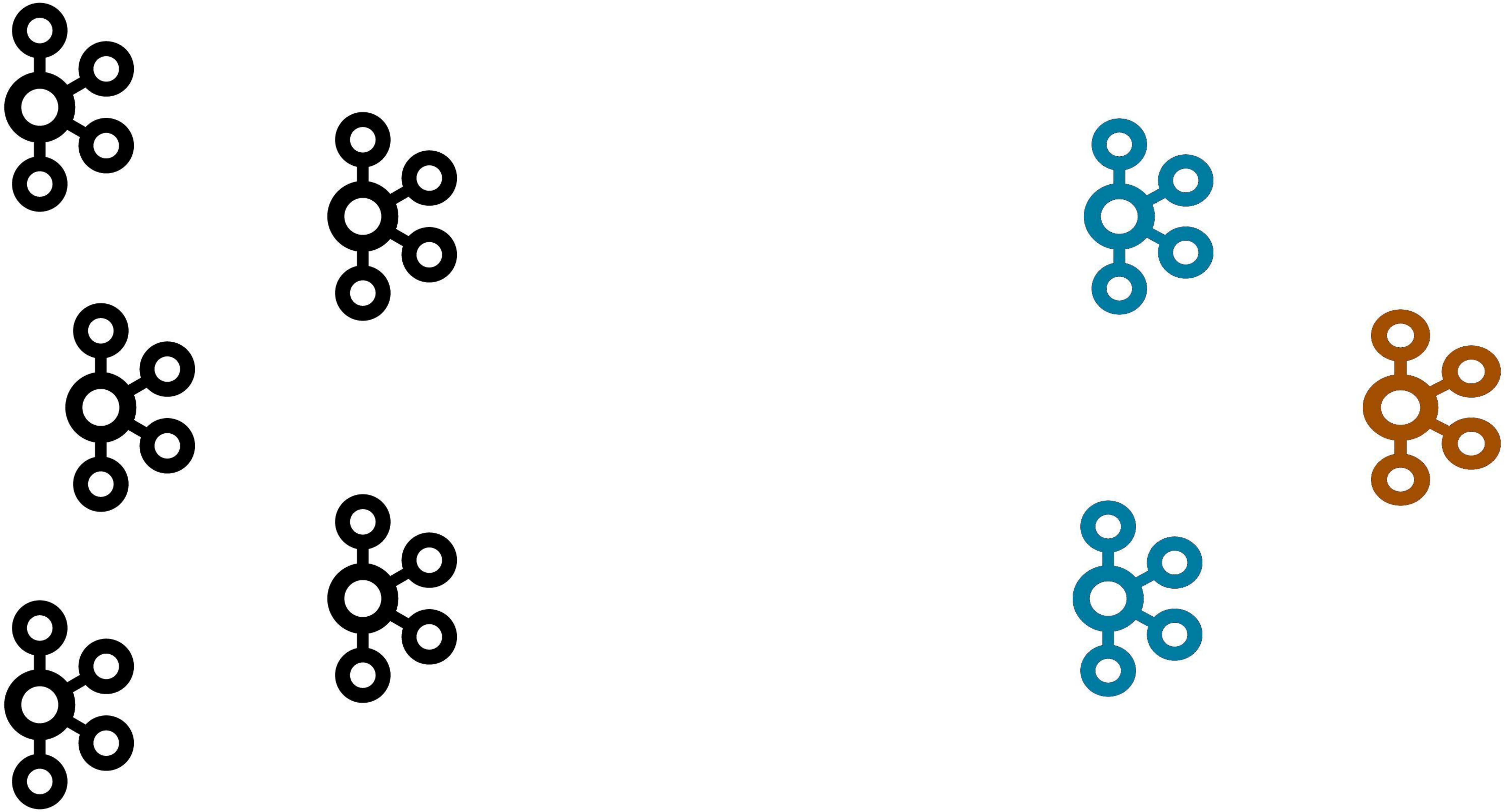
Metadata Nodes

# Dedicated Deployment



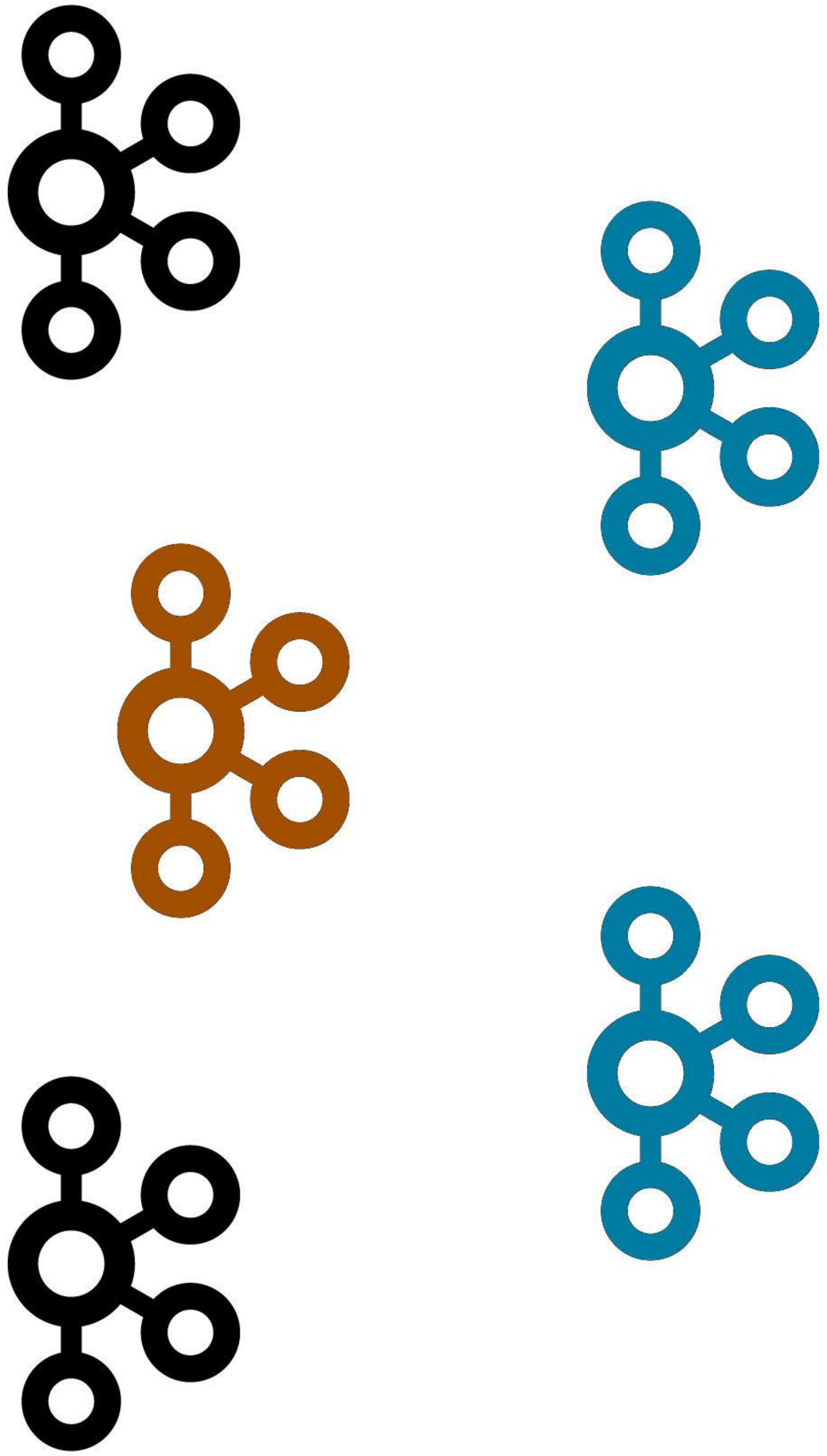
Data Nodes

Metadata Nodes

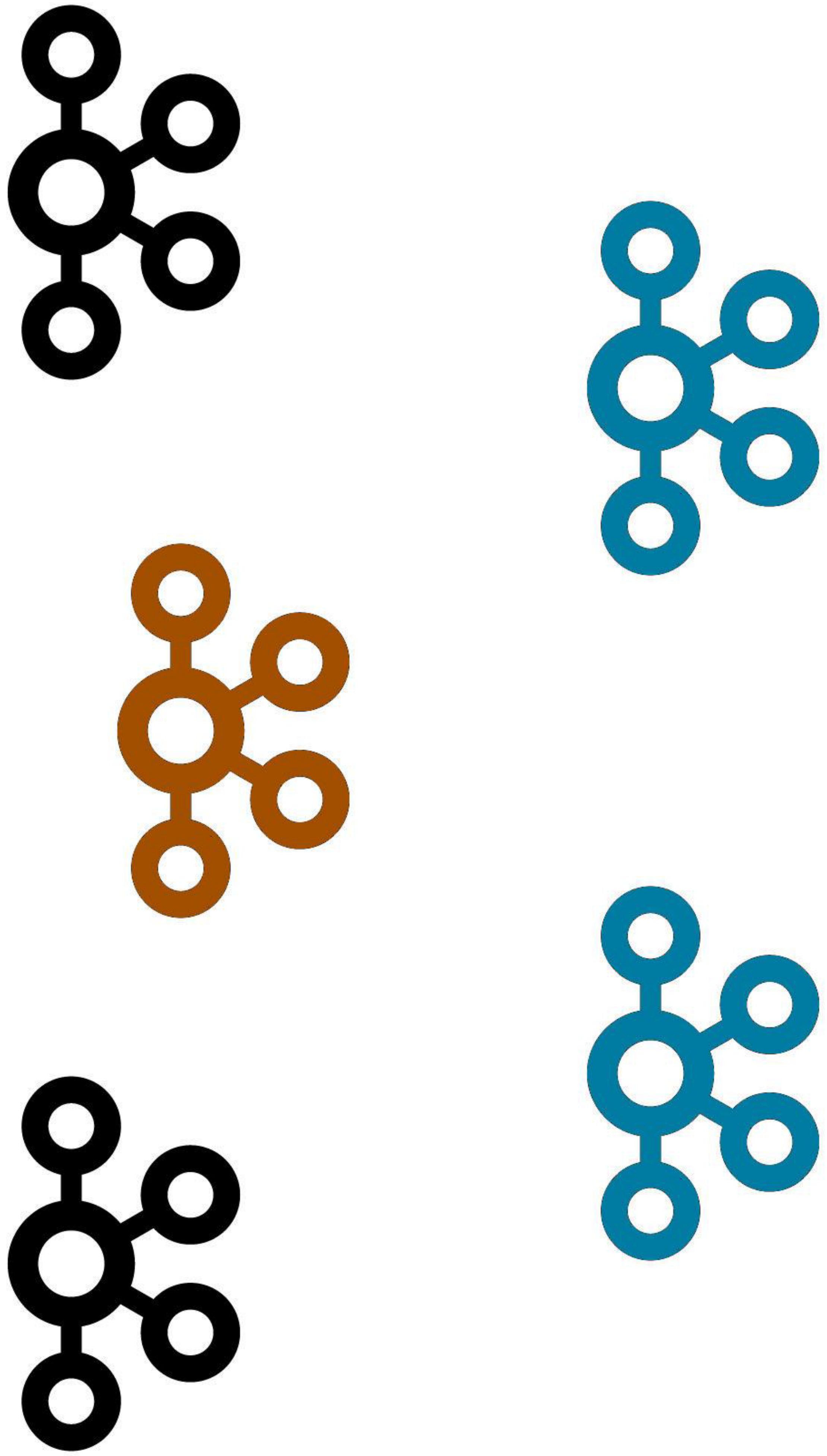


Data Nodes

Metadata Nodes

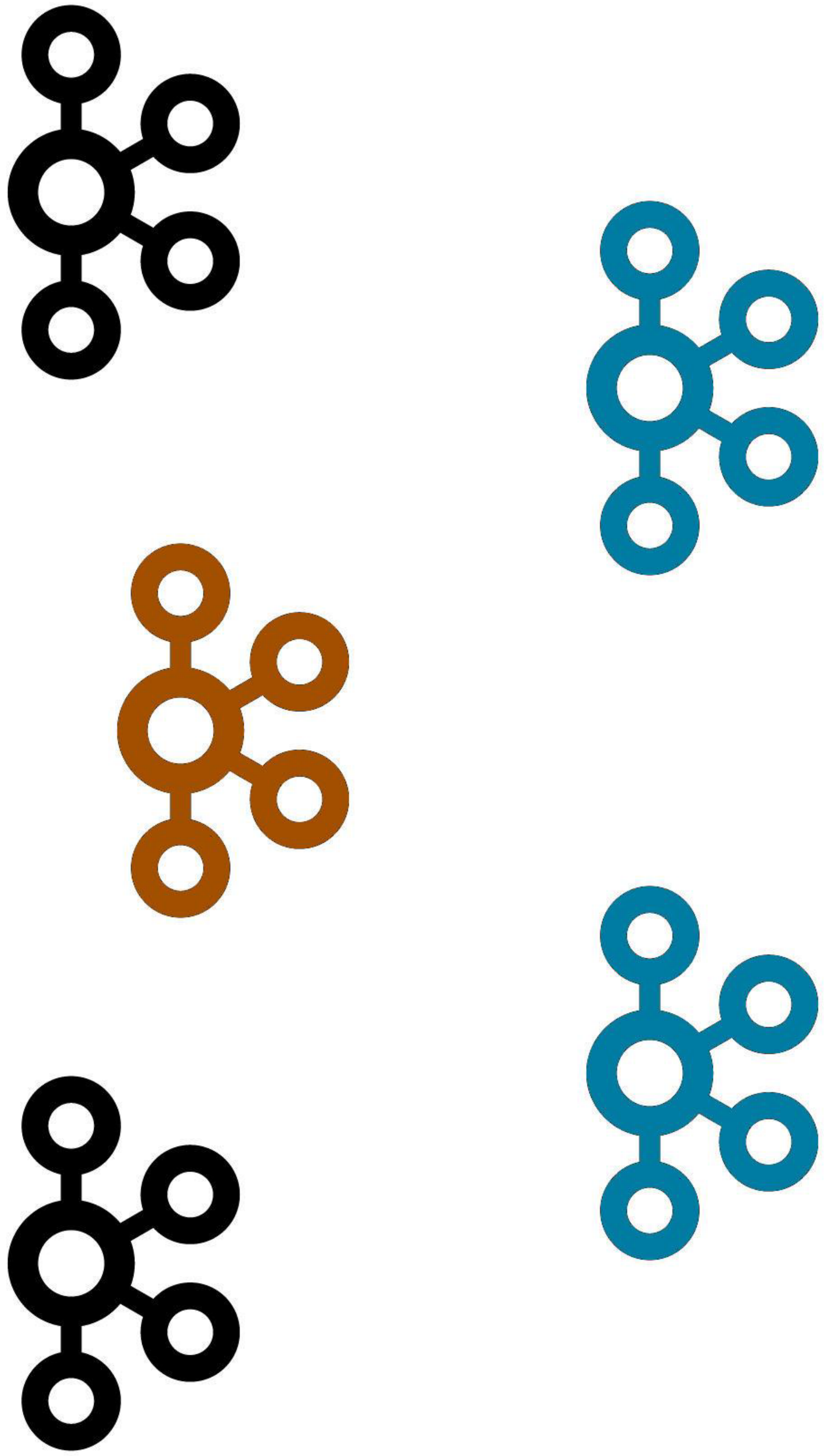


Data Nodes



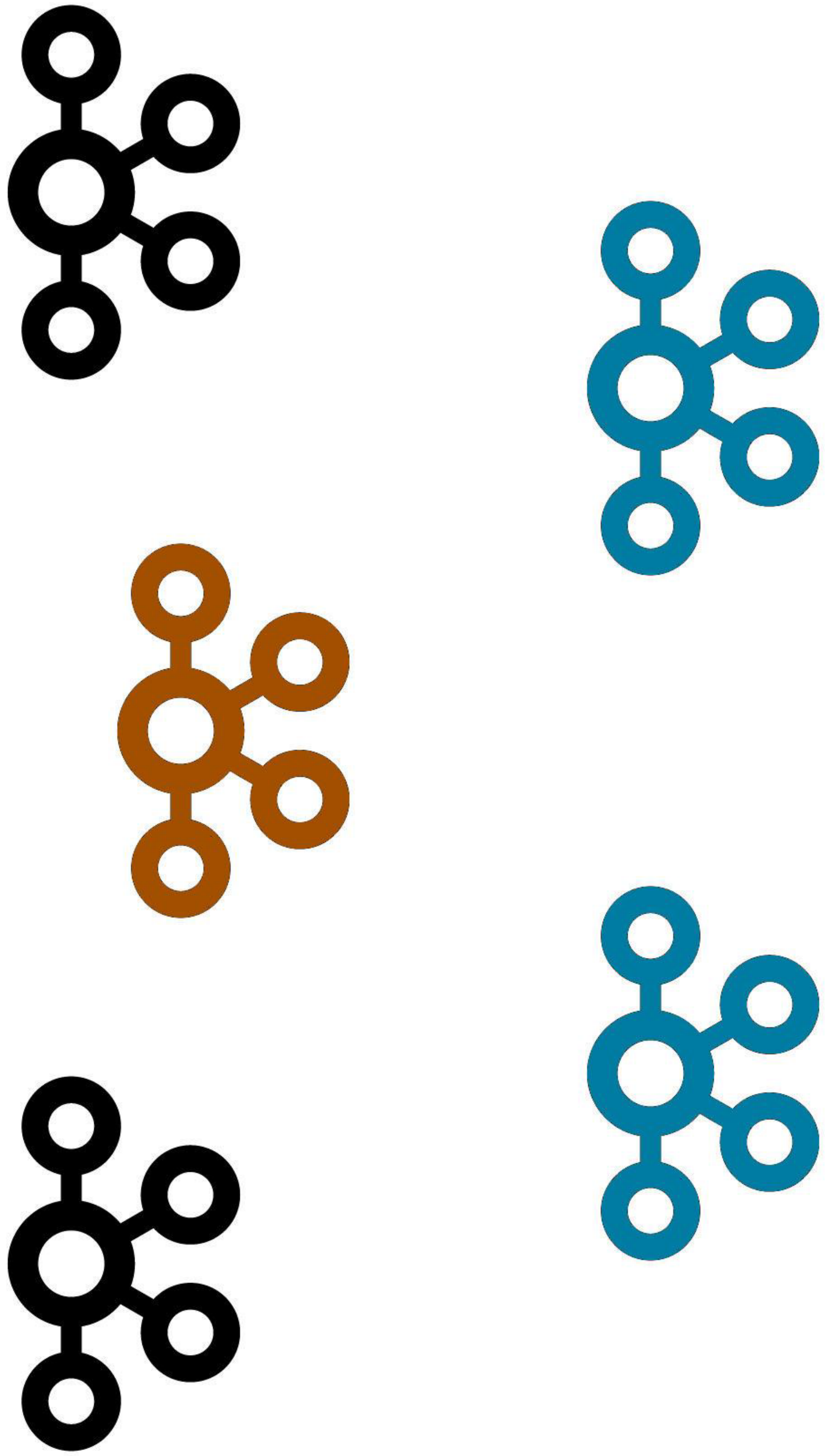
Data/Metadata
Nodes

# Mixed Deployment

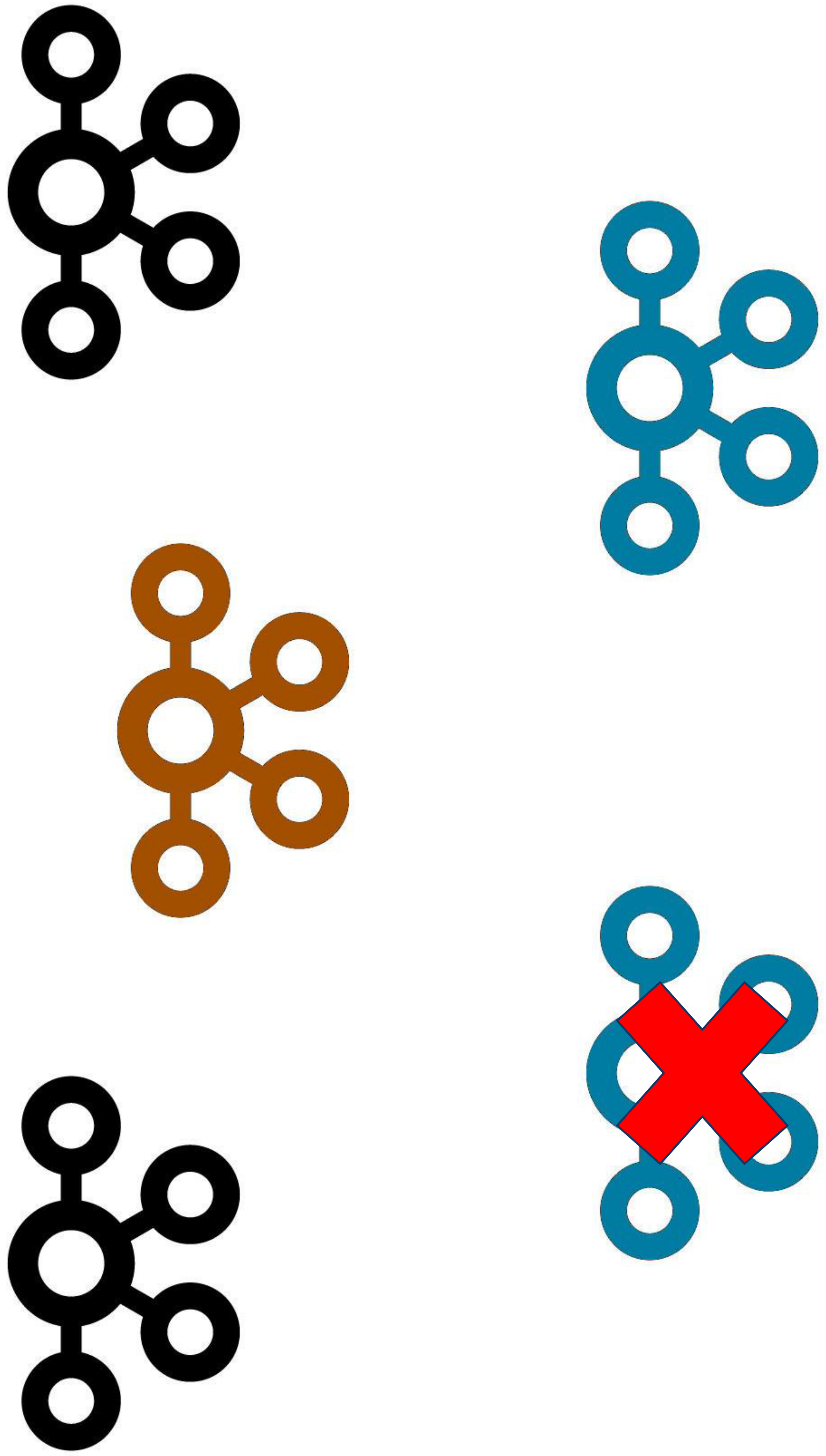


Data/Metadata
Nodes

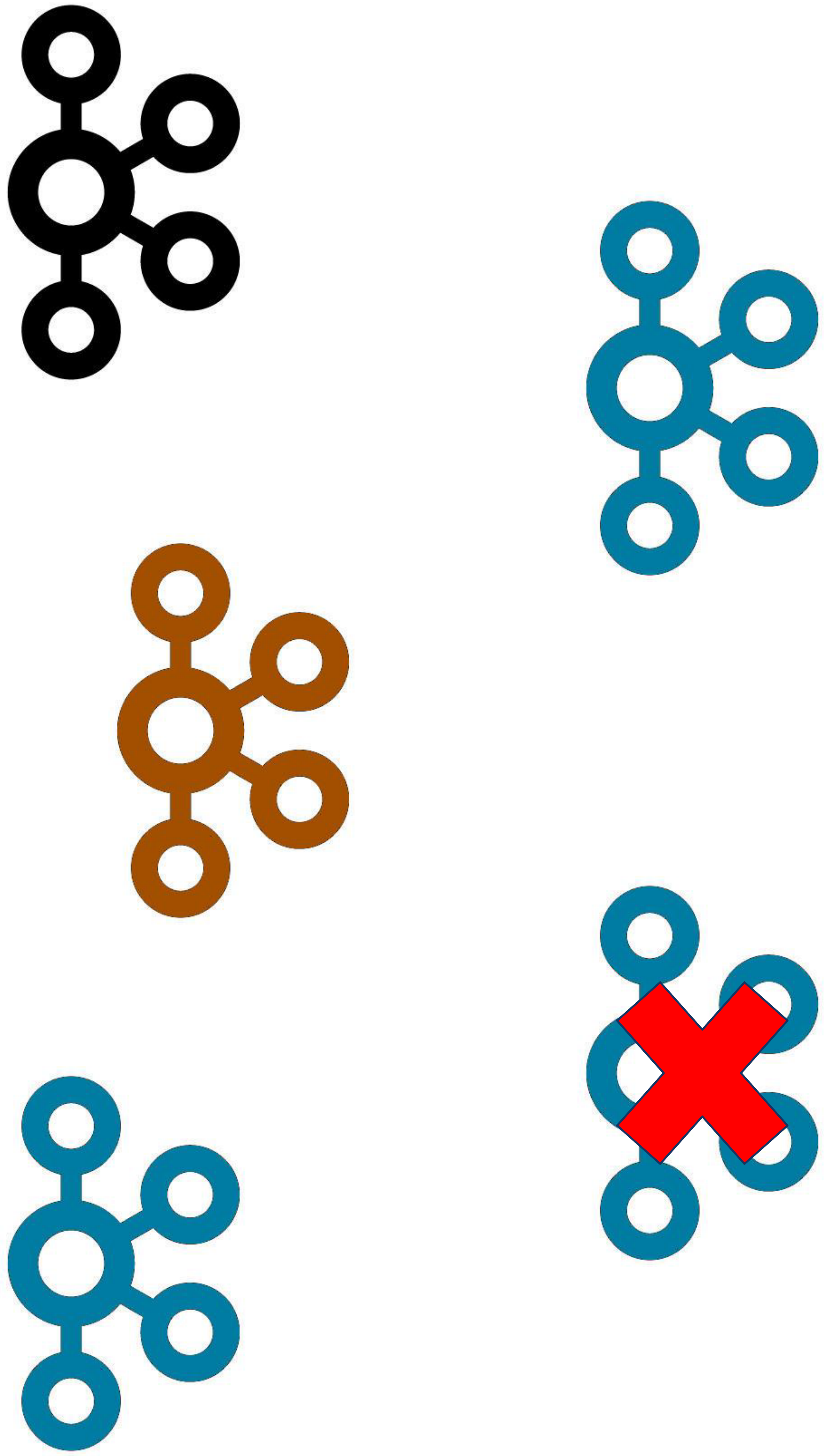
# Mixed Deployment



# Mixed Deployment

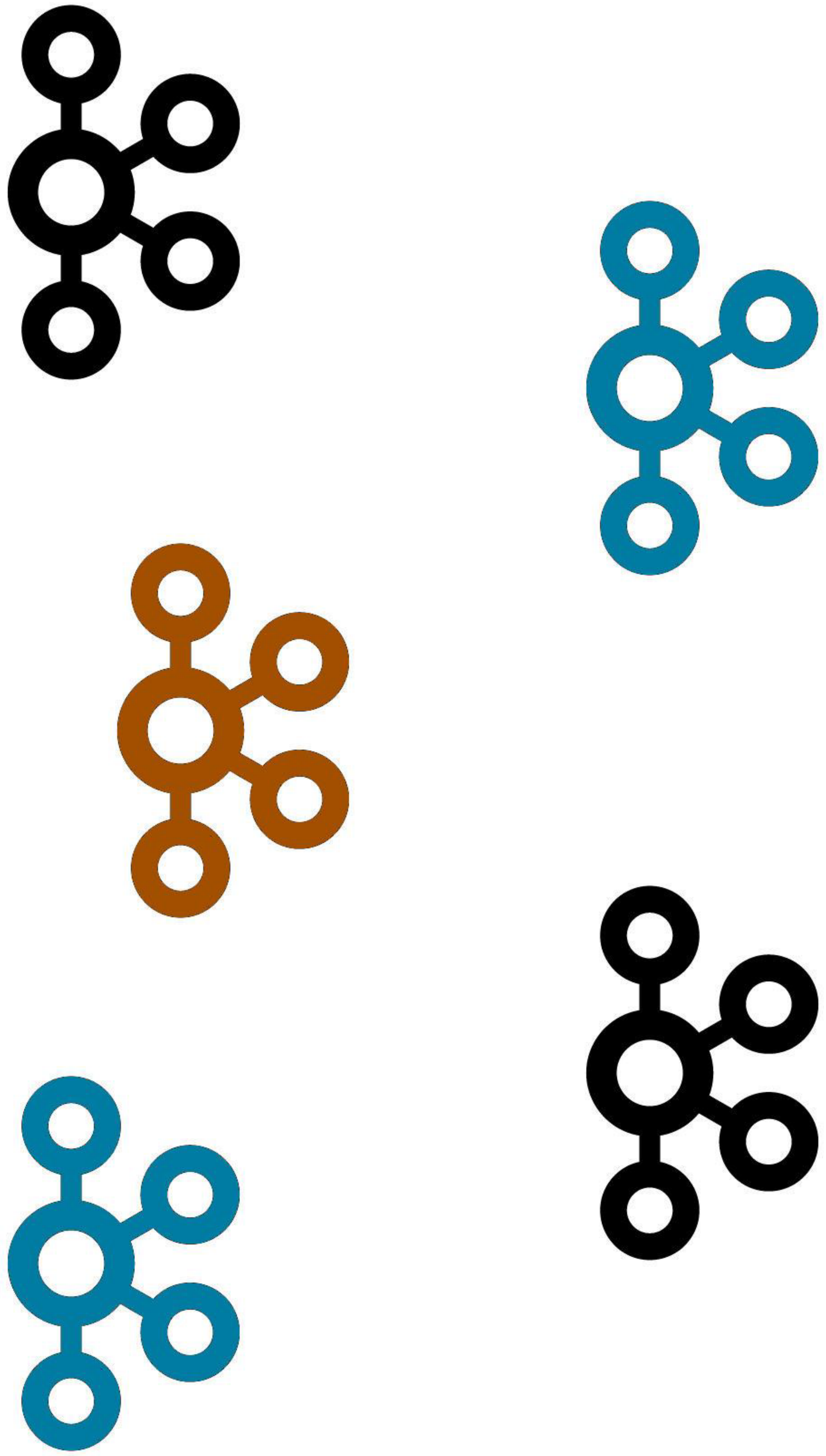


# Mixed Deployment

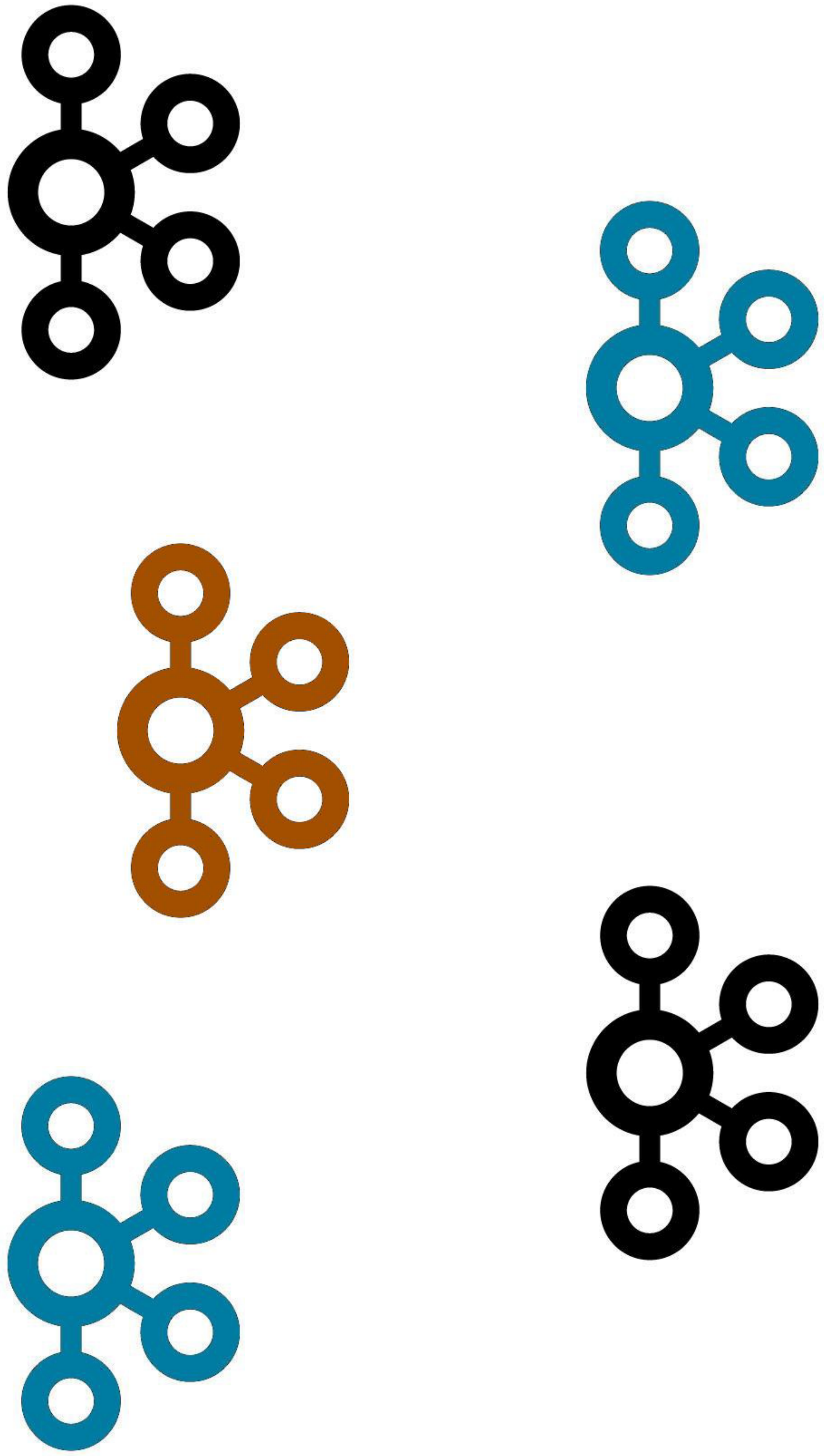


Observer
Promotion

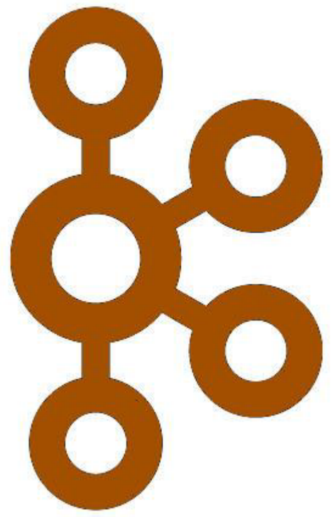
# Mixed Deployment



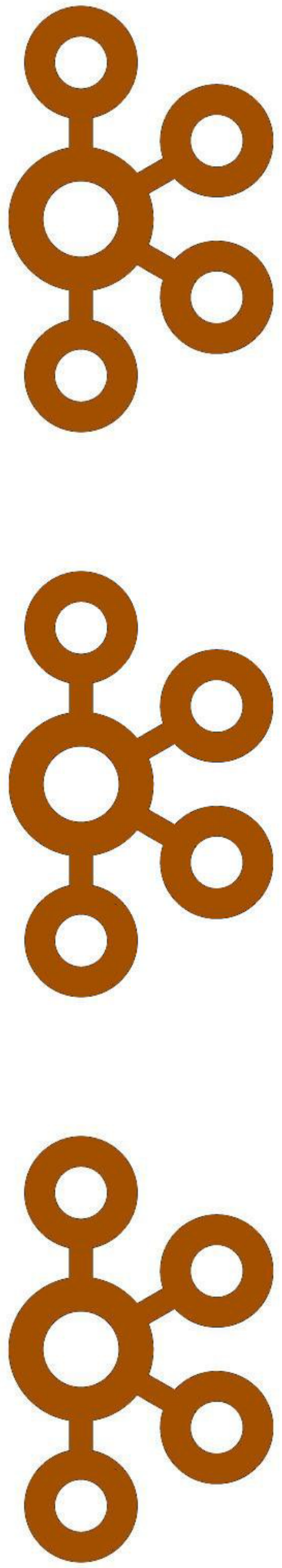
Observer
Promotion



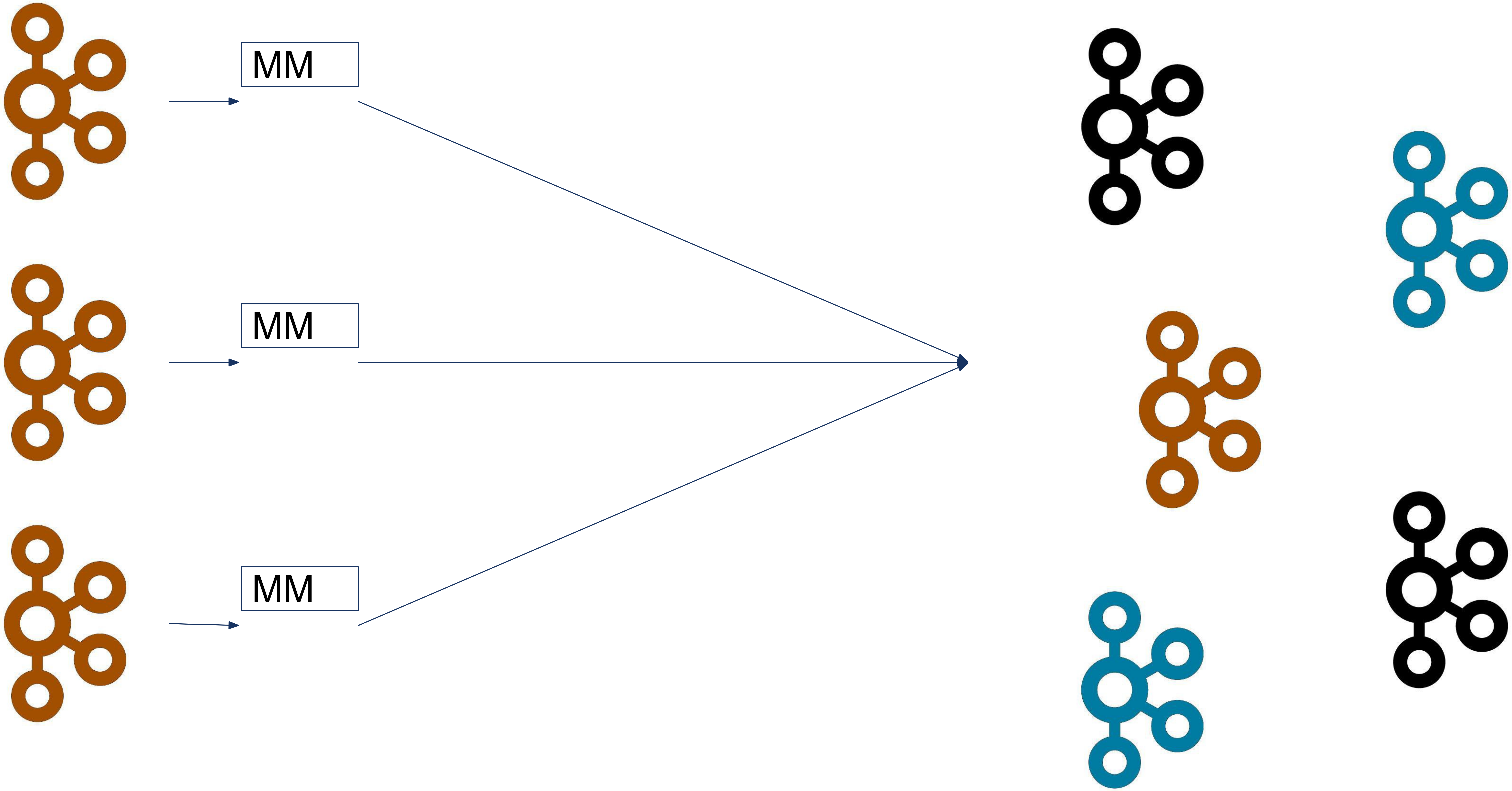
# Single Node Deployment



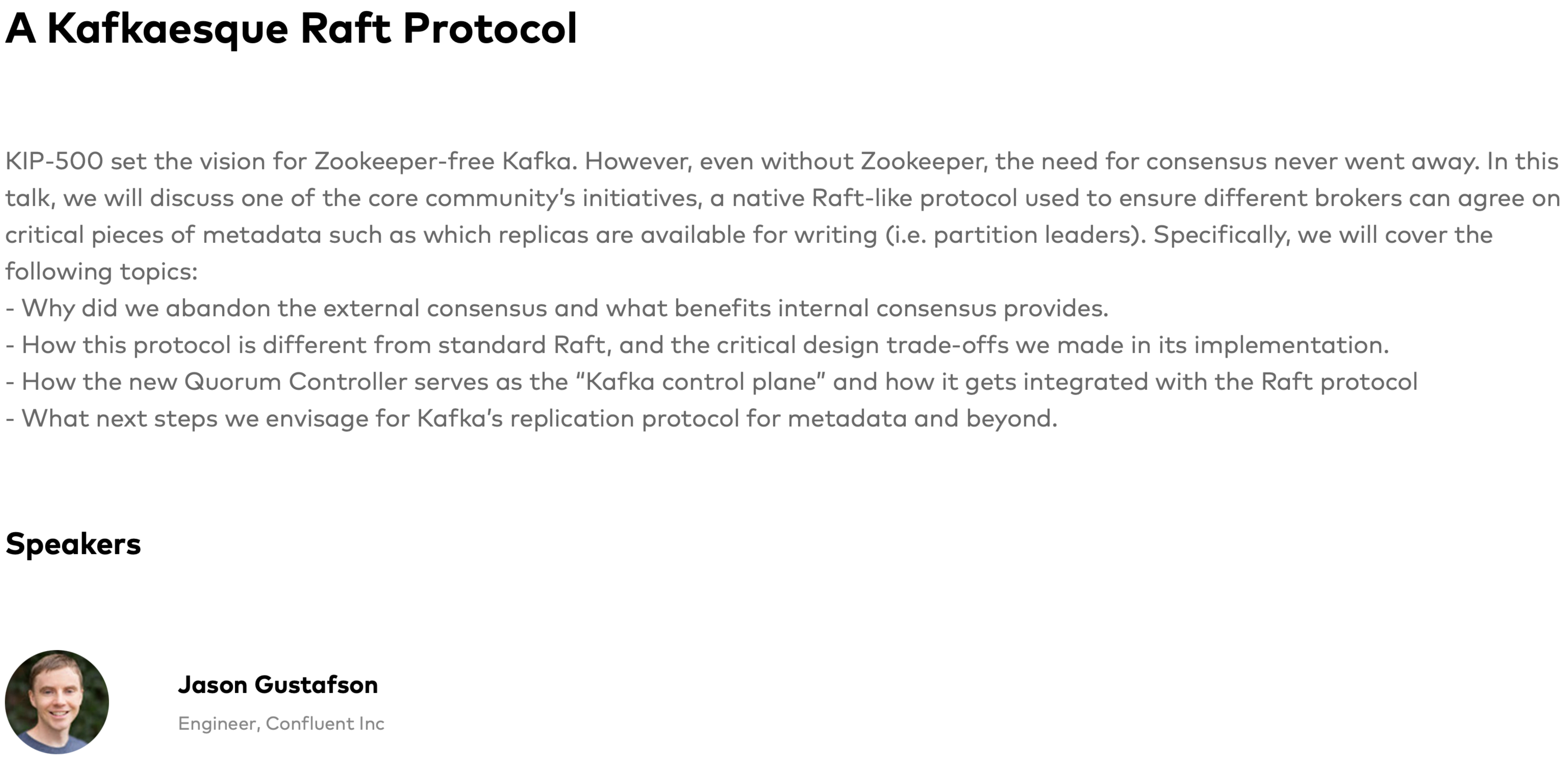
# Single Node Deployment



# Single Node Deployment



# KRaft: Kafkaesque Raft



# A Kafkaesque Raft Protocol

KIP-500 set the vision for Zookeeper-free Kafka. However, even without Zookeeper, the need for consensus never went away. In this talk, we will discuss one of the core community's initiatives, a native Raft-like protocol used to ensure different brokers can agree on critical pieces of metadata such as which replicas are available for writing (i.e. partition leaders). Specifically, we will cover the following topics:

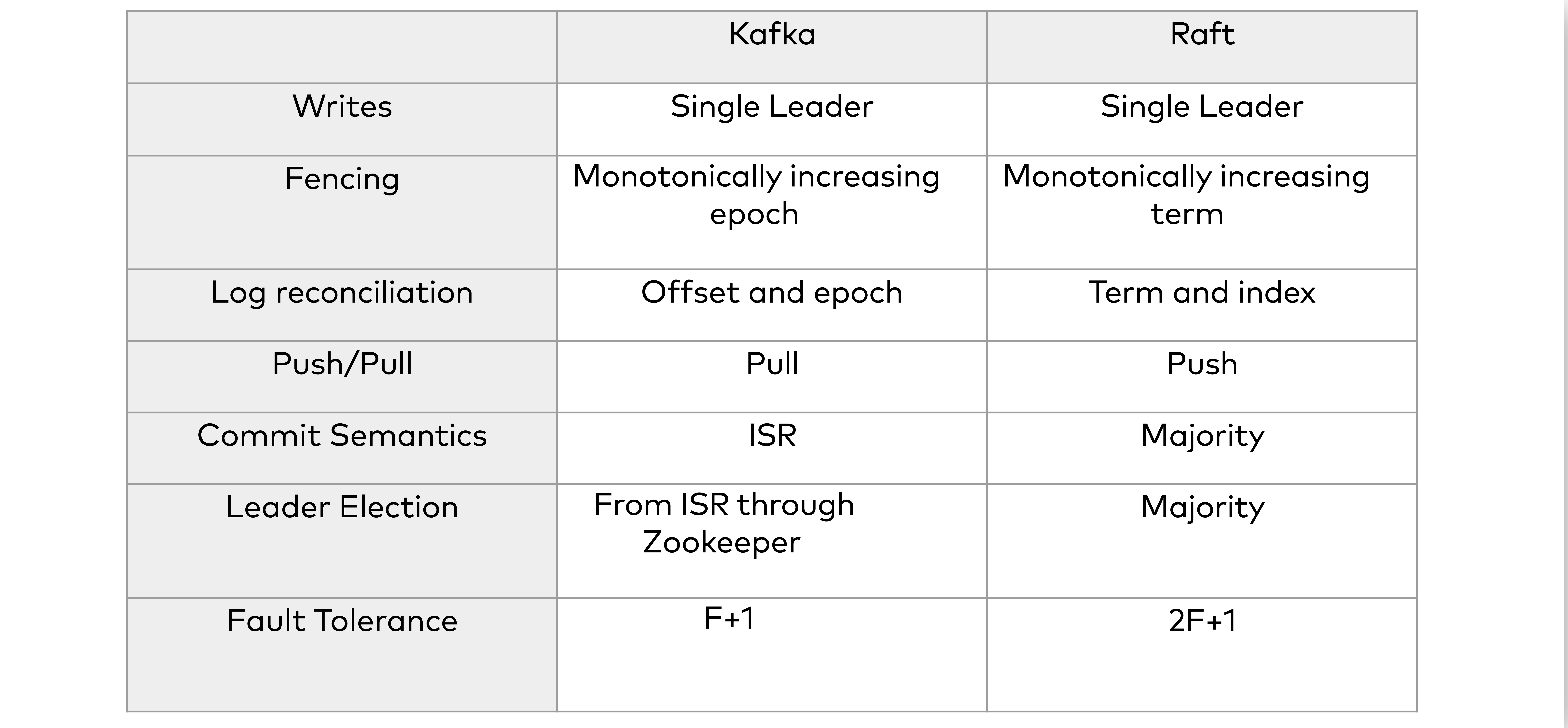
- Why did we abandon the external consensus and what benefits internal consensus provides.
- How this protocol is different from standard Raft, and the critical design trade-offs we made in its implementation.
- How the new Quorum Controller serves as the “Kafka control plane” and how it gets integrated with the Raft protocol
- What next steps we envisage for Kafka's replication protocol for metadata and beyond.

## Speakers

**Jason Gustafson**

Engineer, Confluent Inc

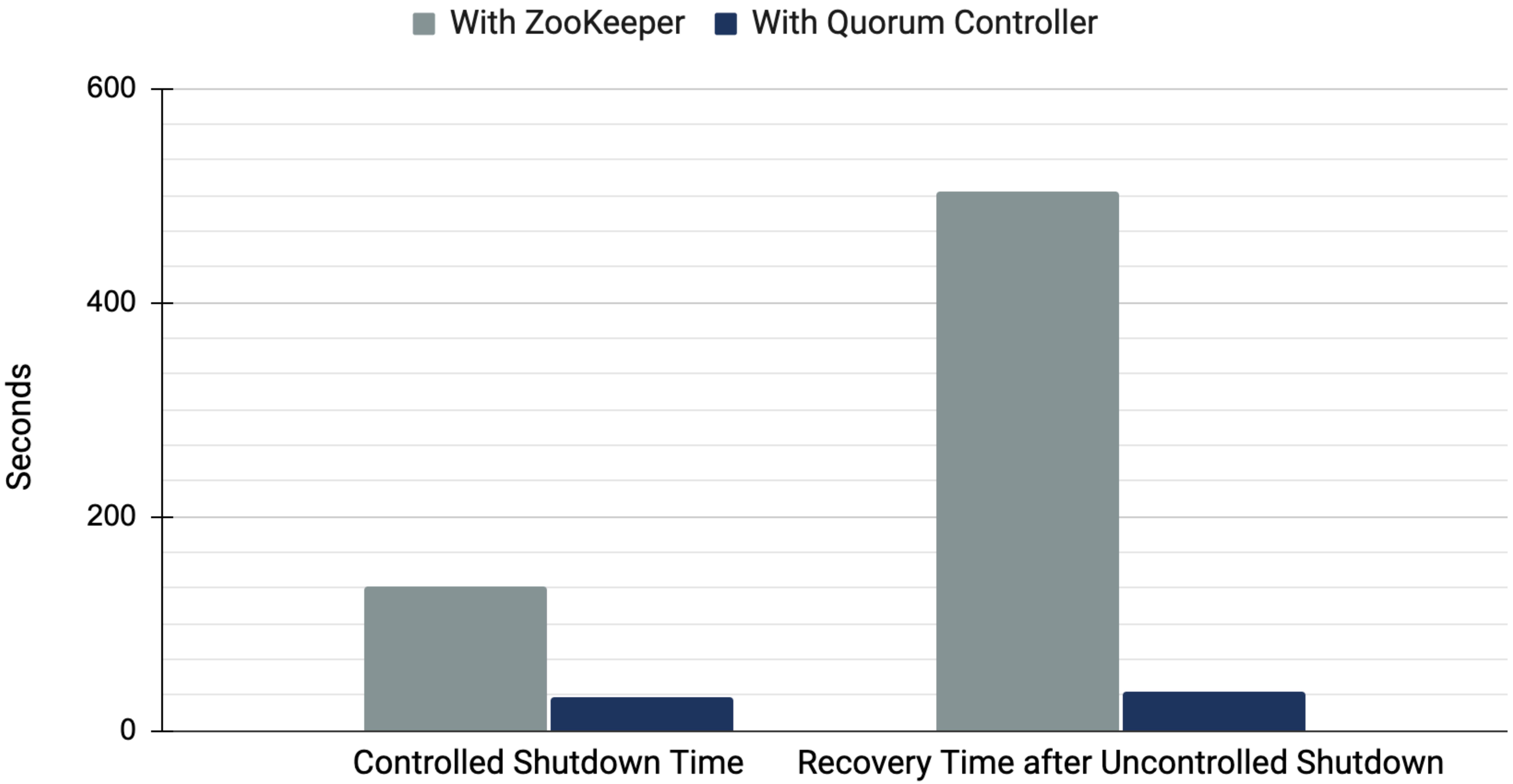
https://www.kafka-summit.org/sessions/a-kafkaesque-raft-protocol



| | Kafka | Raft |
|---|---|---|
| Writes | Single Leader | Single Leader |
| Fencing | Monotonically increasing epoch | Monotonically increasing term |
| Log reconciliation | Offset and epoch | Term and index |
| Push/Pull | Pull | Push |
| Commit Semantics | ISR | Majority |
| Leader Election | From ISR through Zookeeper | Majority |
| Fault Tolerance | F+1 | 2F+1 |

Timed Shutdown Operations In Apache Kafka with 2 Million Partitions

*Faster is better*



# STOP! Demo time!

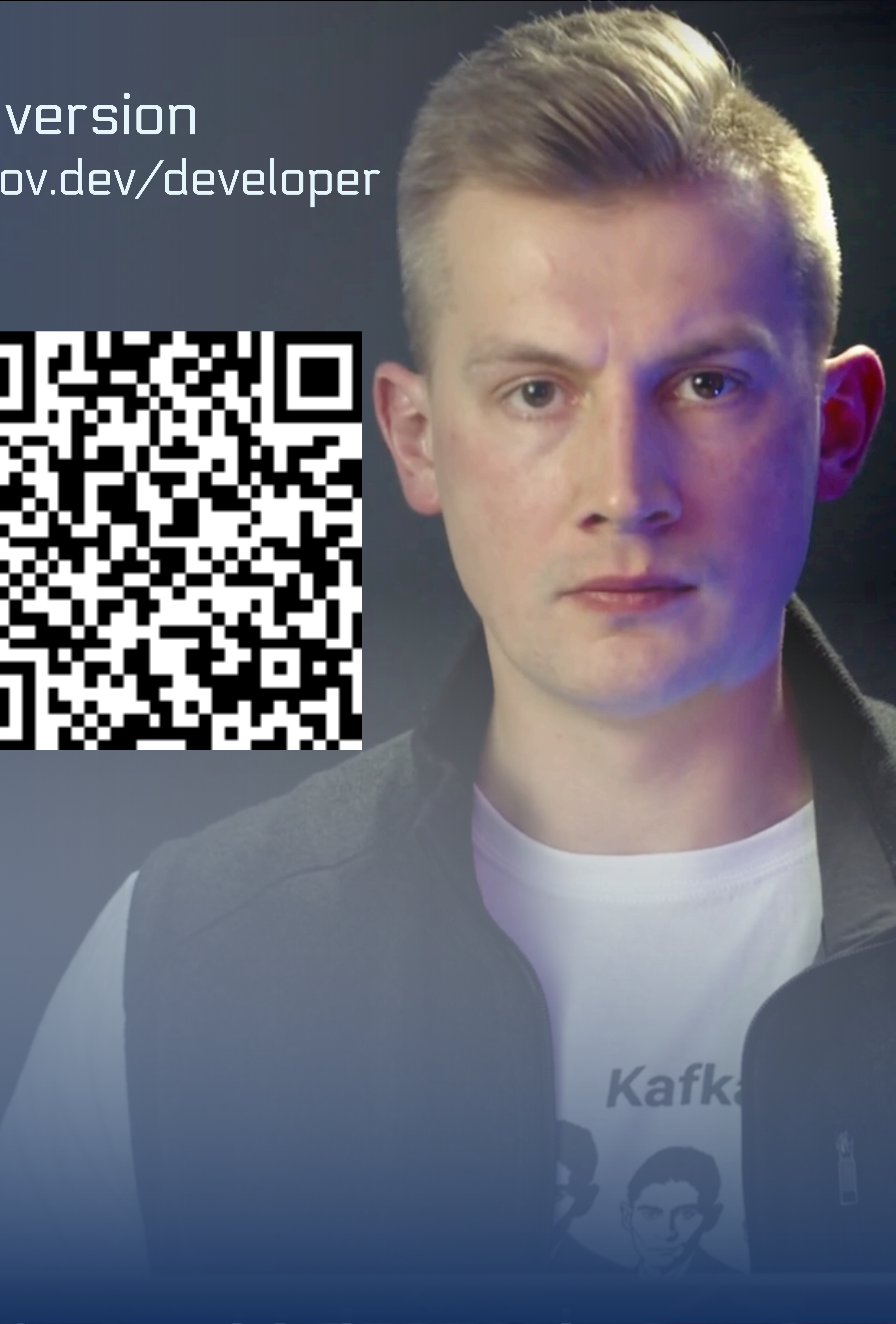
CONFLUENT

# Want to learn more about Kafka?

# Learn Kafka.

Watch full version
https://gamov.dev/developer

## Start building with Apache Kafka at Confluent Developer.



Confluent Developer
developer.confluent.io

# As Always
# Have A Nice Day

CONFLUENT