# Real-Time Data Processing with Apache Kafka: A Kubernetes-Native Approach

dojobits

Iliyan Petkov

# #whoami

dojobits

## Iliyan Petkov

iliyan-s-petkov

@Iliyan_petkov

@Iliyan-s-petkov

## Valentin Hristev

valentin-hristev

@vhristev

kubestronaut

aws certified Machine Learning Engineer ASSOCIATE

Prometheus CERTIFIED ASSOCIATE

https://dojobits.io/

## What we do

Professional services

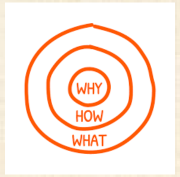
Management services

Cloud design and architecture

Trainings

# Our Goals:

Define the Problem

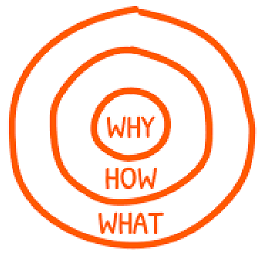


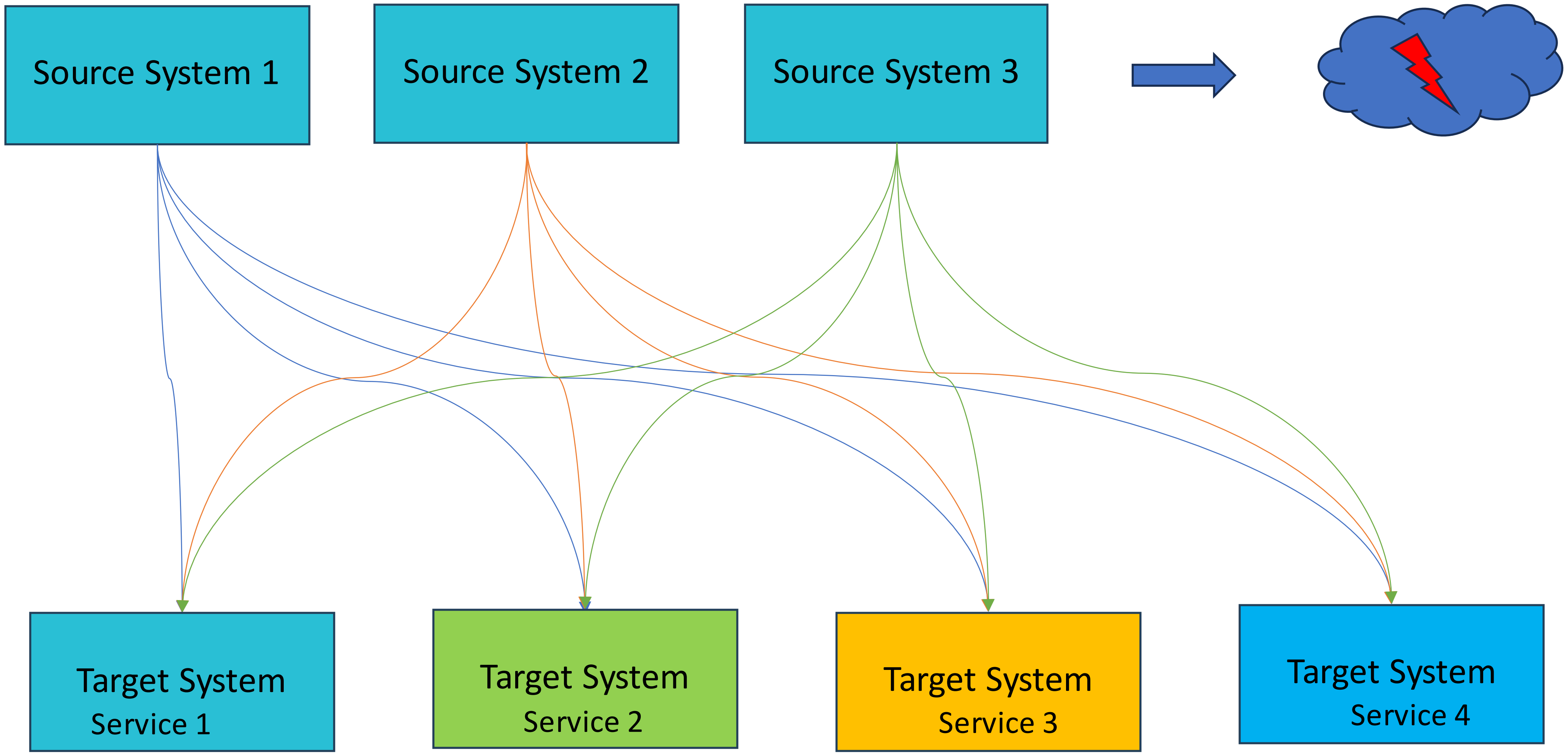
Apache Kafka Overview

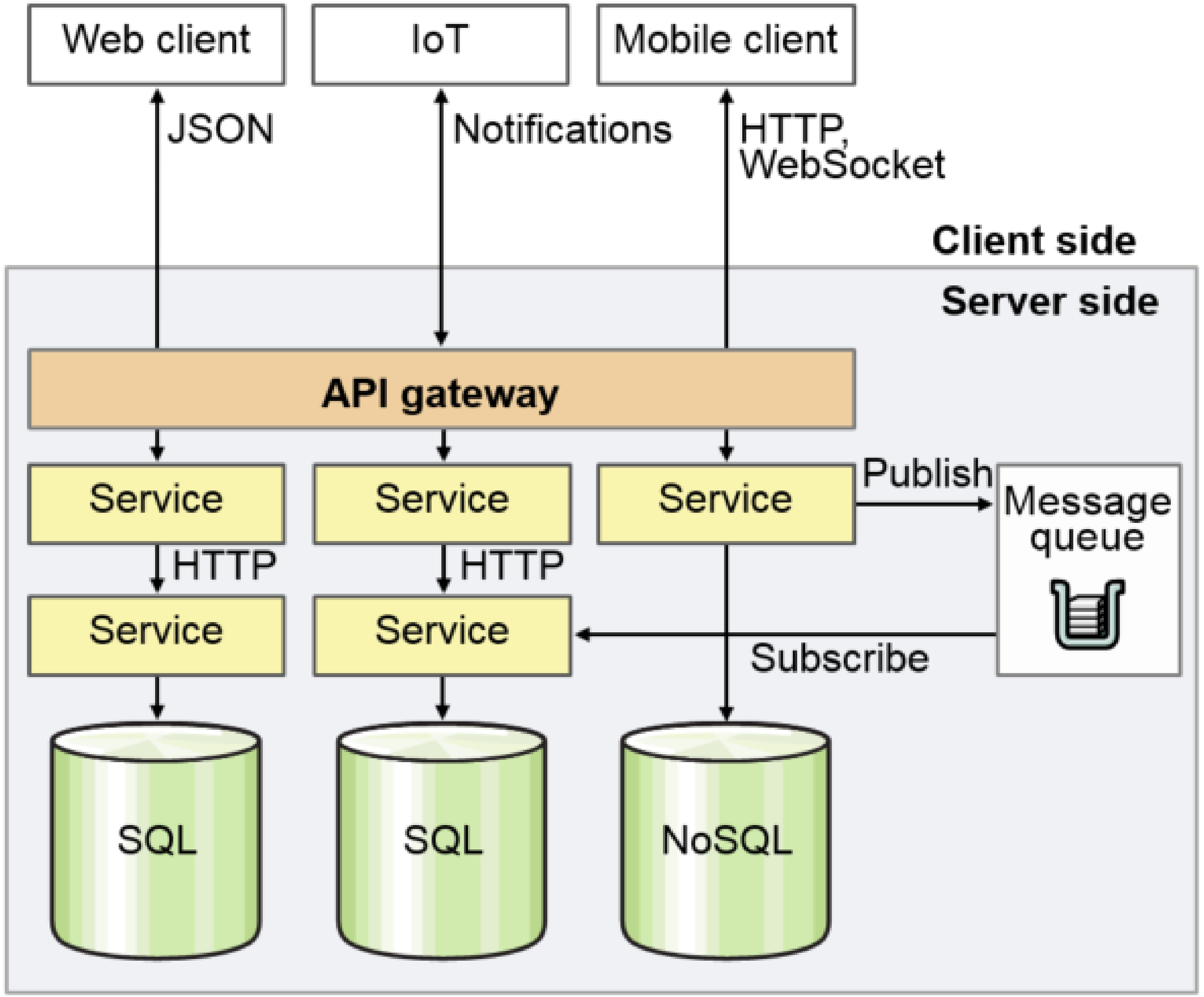
Running Apache Kafka on Kubernetes

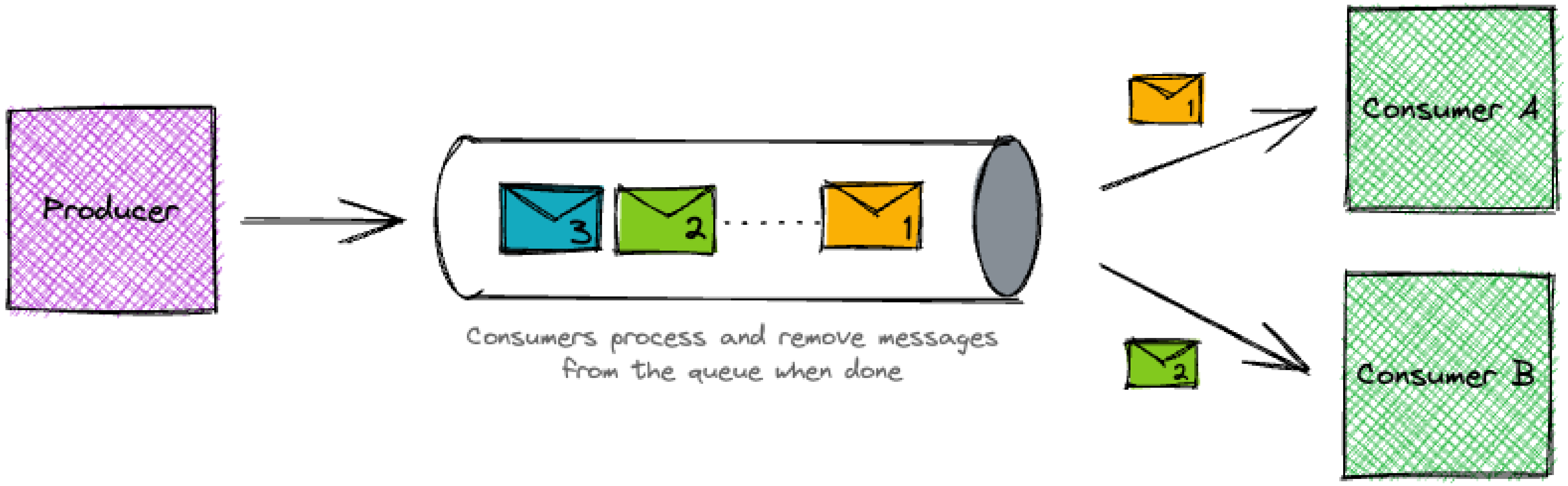
Never show a single line of JAVA code…

# Let’s start with Why?



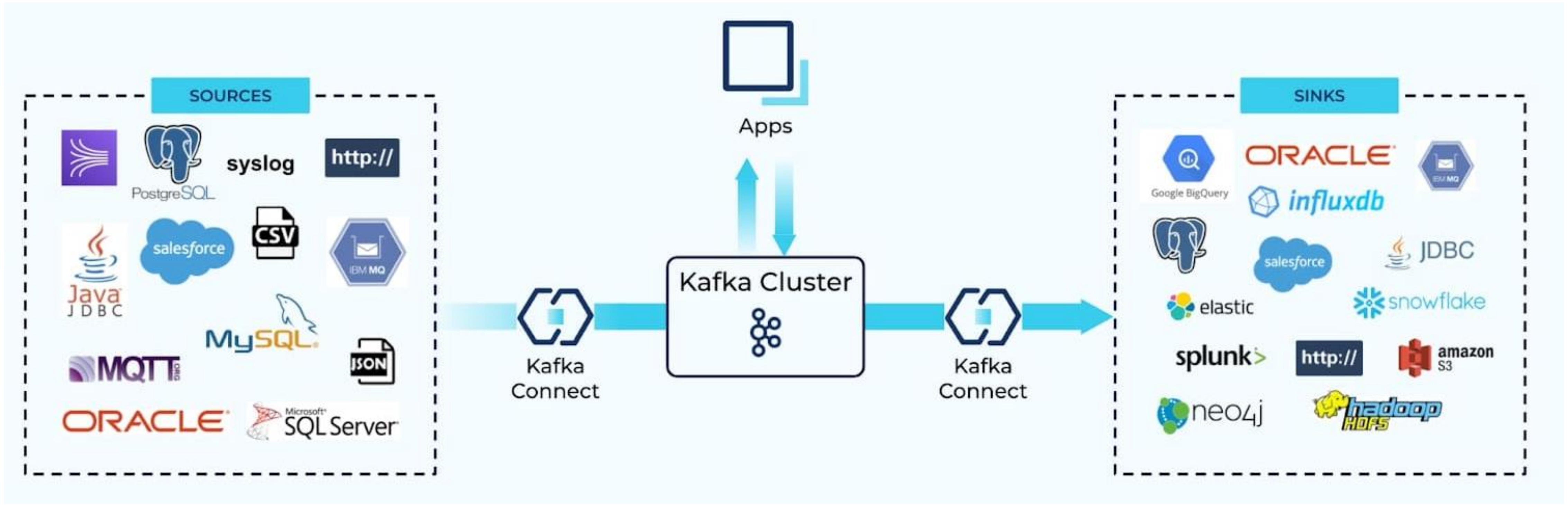

Source System 1
Source System 2
Source System 3
Target System Service 1
Target System Service 2
Target System Service 3
Target System Service 4


Web client
IoT
Mobile client
JSON
Notifications
HTTP, WebSocket
Client side
Server side
API gateway
Service
Service
Service
Publish
Message queue
HTTP
HTTP
Service
Service
Subscribe
SQL
SQL
NoSQL

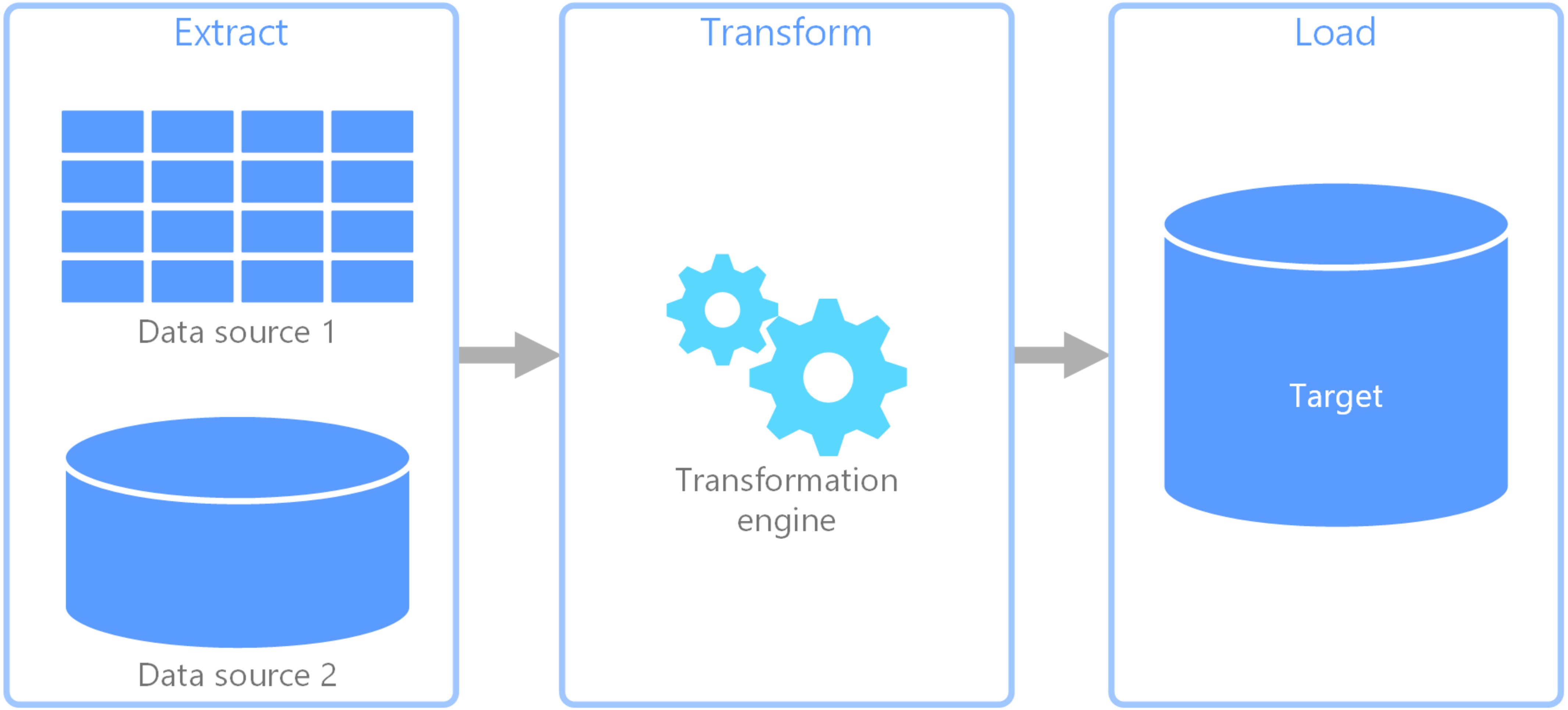

Producer
3
2
1
Consumer A
Consumer B
Consumers process and remove messages from the queue when done

## Queues

Consumers pull messages, process and remove



SOURCES
PostgreSQL
syslog
http://
salesforce
CSV
IBM MQ
Java JDBC
MySQL
MQTT.ORG
JSON
ORACLE
Microsoft SQL Server
Kafka Connect
Apps
Kafka Cluster
Kafka Connect
SINKS
Google BigQuery
ORACLE
influxdb
IBM MQ
salesforce
JDBC
elastic
snowflake
splunk>
http://
amazon S3
neo4j
hadoop HDFS


Extract
Data source 1
Data source 2
Transform
Transformation engine
Load
Target

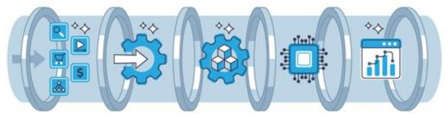
APACHE kafka™

## Overview

- Initially developed inside Linked-in
- Open-Source since 2011
- Donated to the Apache Software Foundation
- Event streaming platform
- Distributed
- Highly-available

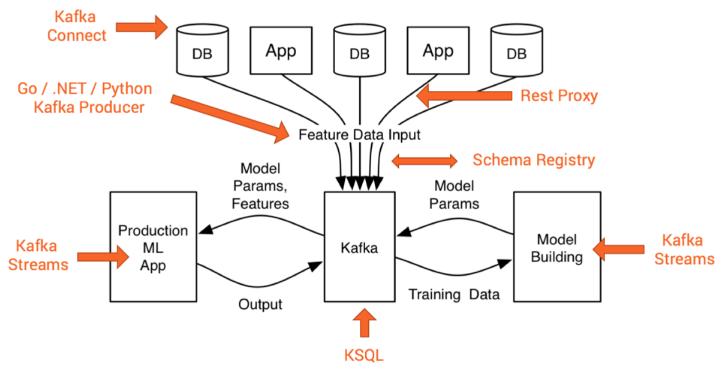
Real-time data processing

Stream processing



Building Data pipelines

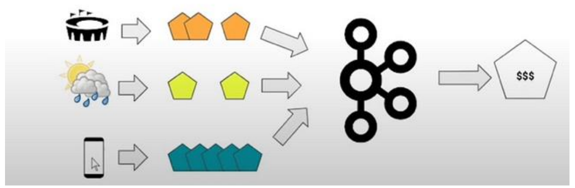
CYBER SECURITY

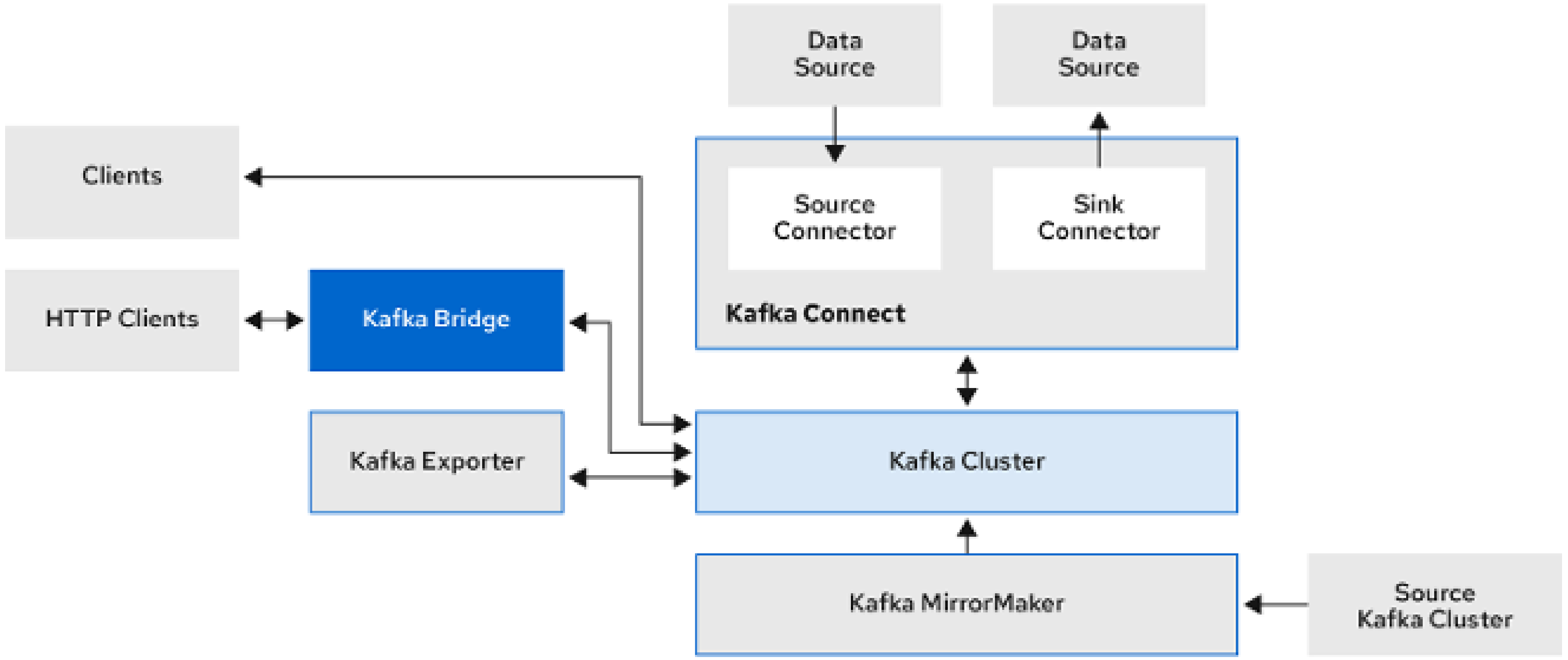

Kafka Connect
DB
App
DB
App
DB
Go / .NET / Python Kafka Producer
Rest Proxy
Feature Data Input
Schema Registry
Model Params, Features
Model Params
Kafka Streams
Production ML App
Kafka
Model Building
Kafka Streams
Output
Training Data
KSQL

BANKING

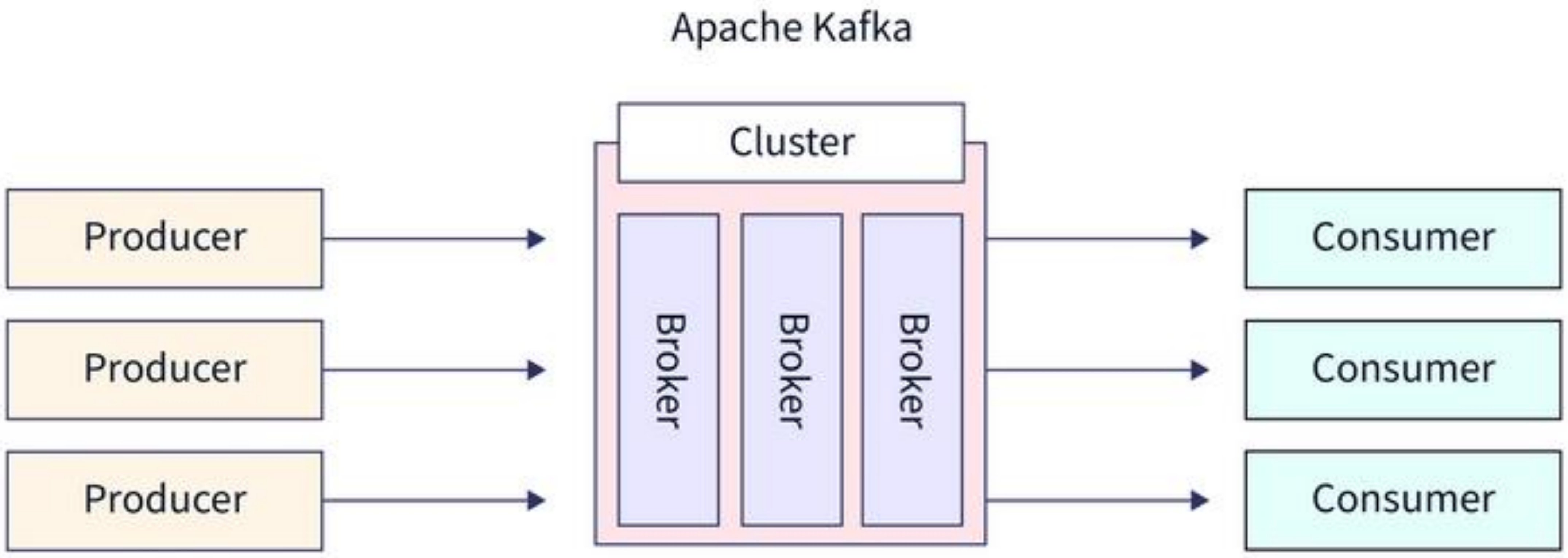
IOT

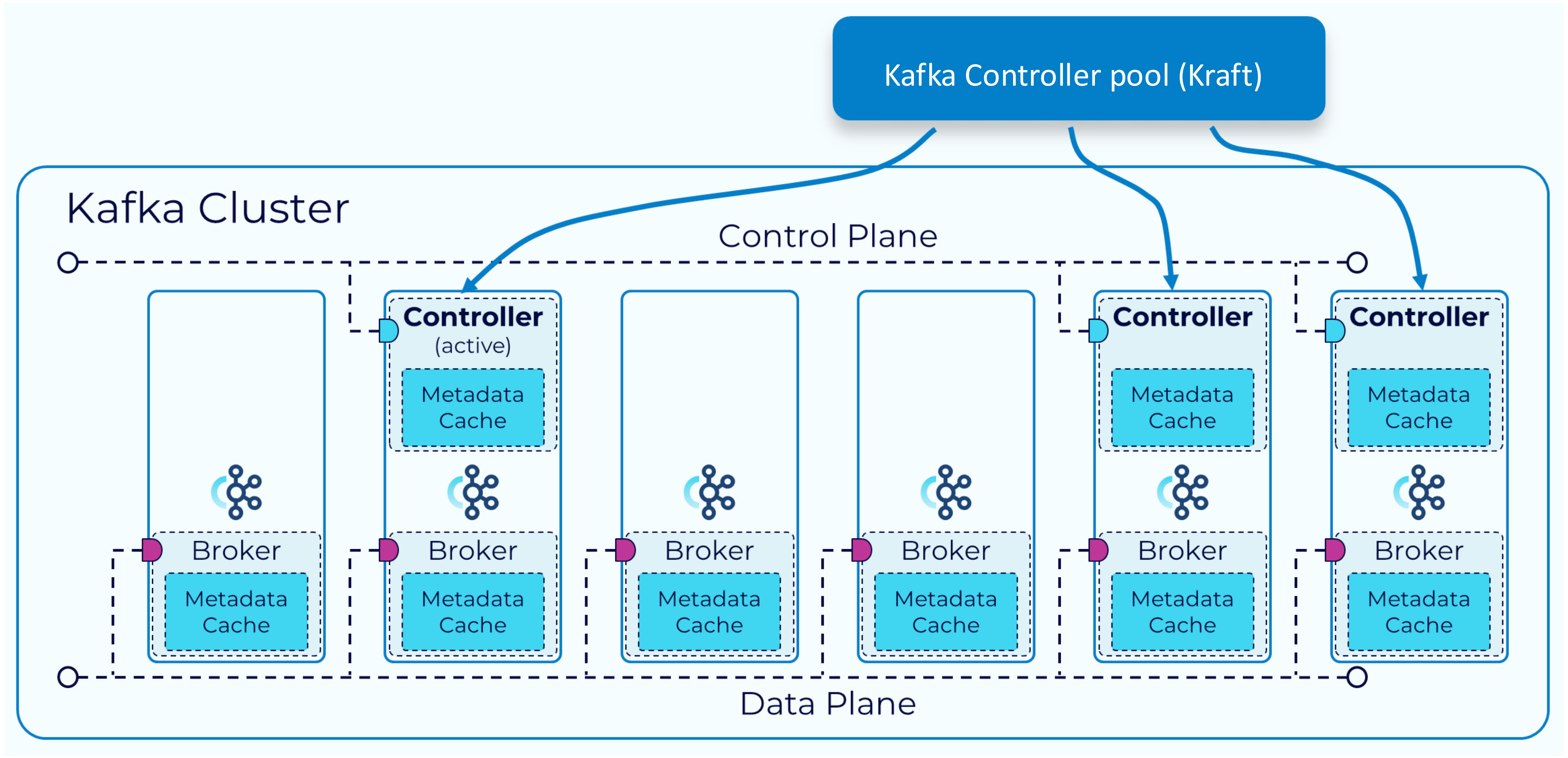
Uber

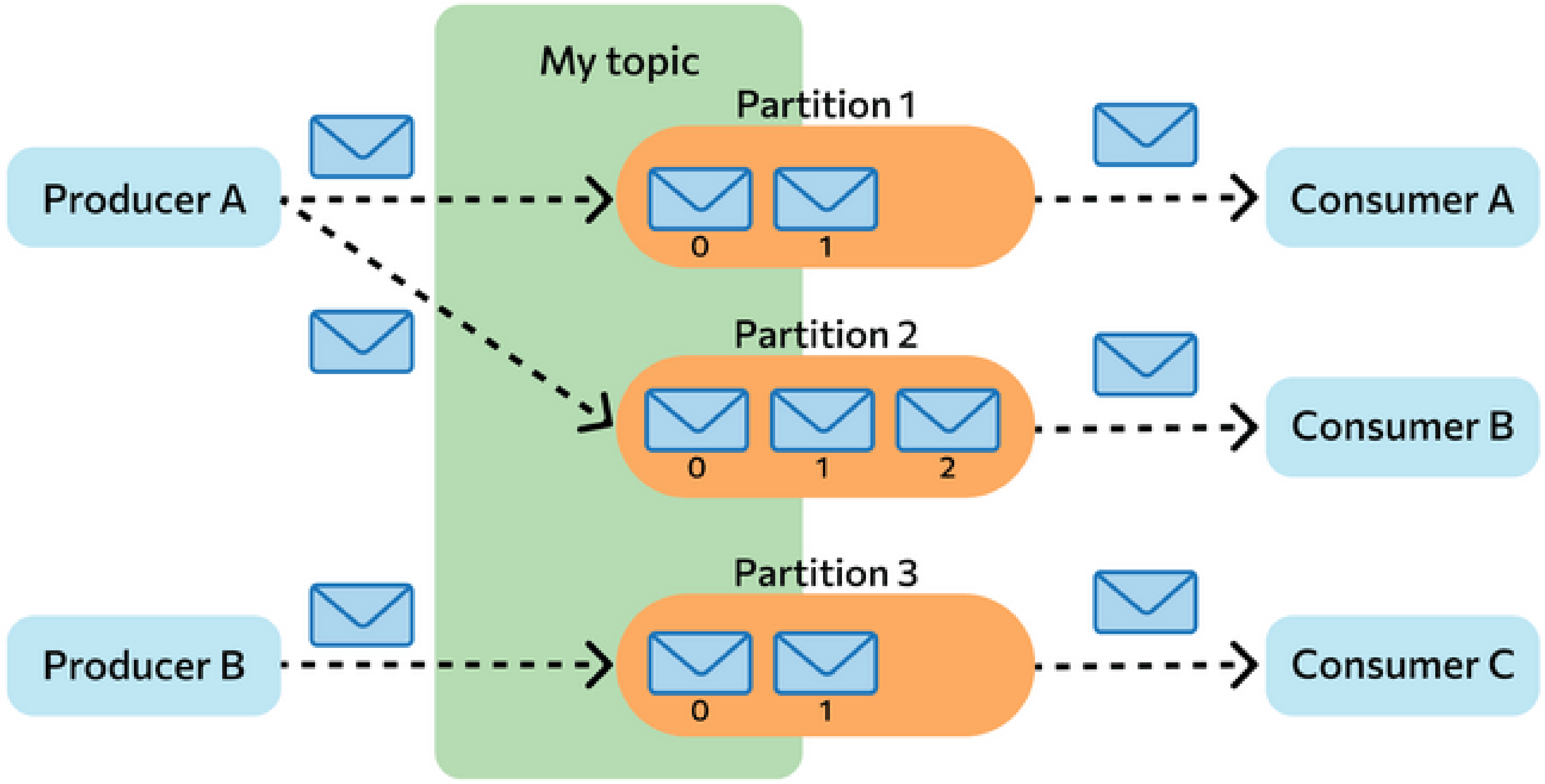

$$$

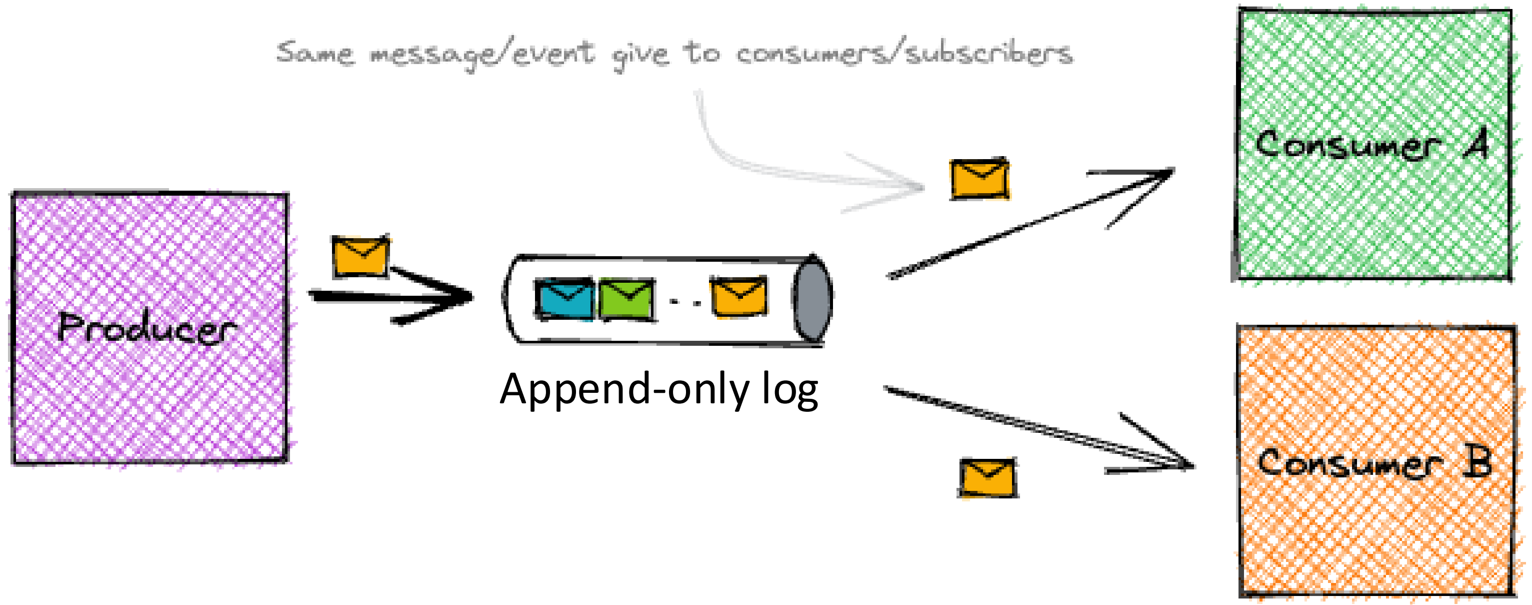


Source: [https://strimzi.io/docs/operators/latest/overview](https://strimzi.io/docs/operators/latest/overview)


Apache Kafka
Cluster
Producer
Producer
Producer
Broker
Broker
Broker
Consumer
Consumer
Consumer

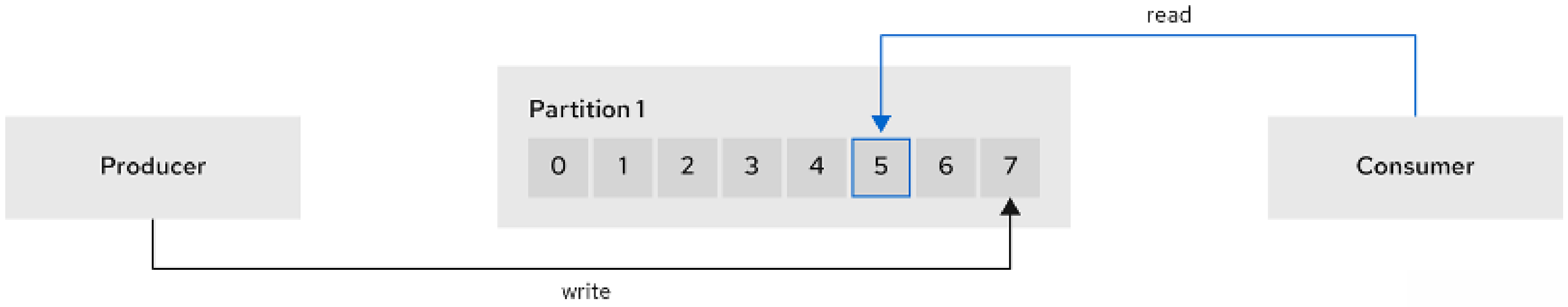

Kafka Controller pool (Kraft)
Kafka Cluster
Control Plane
Controller
(active)
Metadata Cache
Controller
Metadata Cache
Controller
Metadata Cache
Broker
Metadata Cache
Broker
Metadata Cache
Broker
Metadata Cache
Broker
Metadata Cache
Broker
Metadata Cache
Broker
Metadata Cache
Data Plane

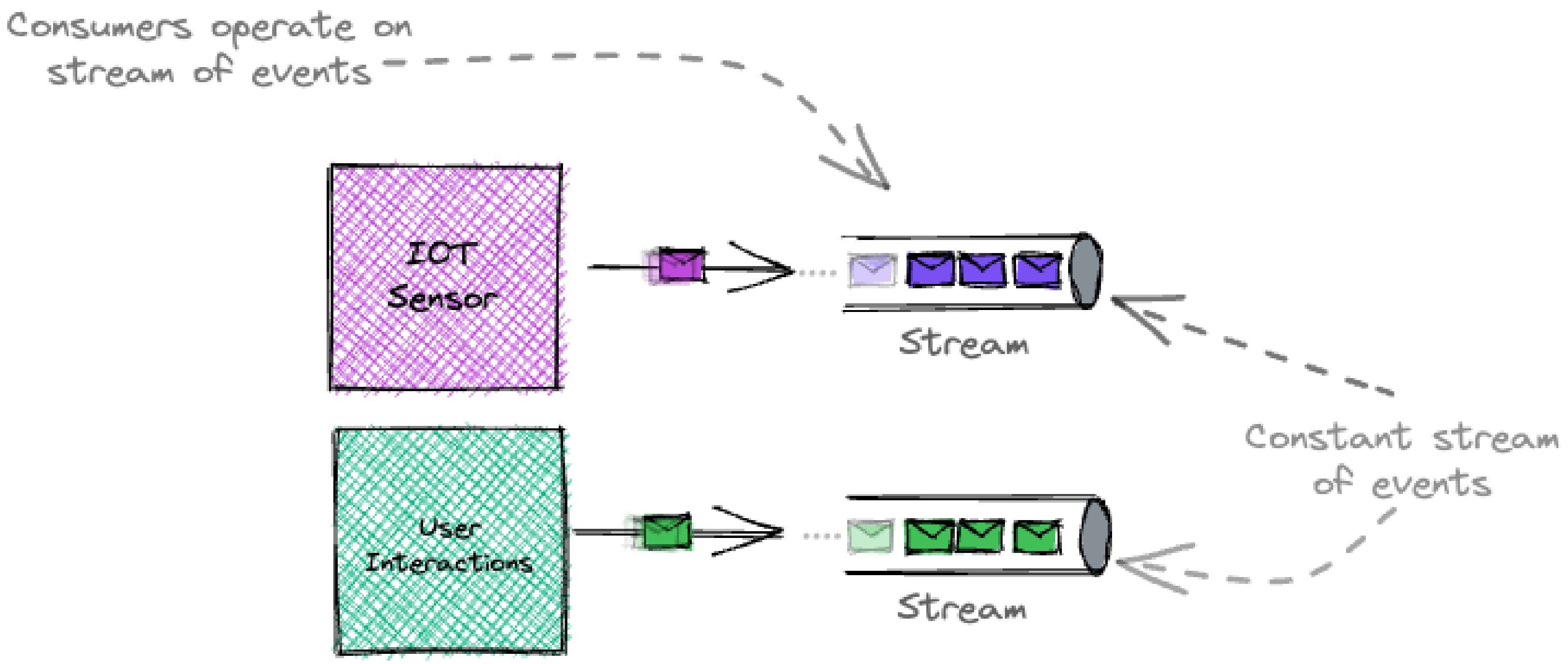

My topic
Partition 1
Partition 2
Partition 3
Producer A
Producer B
Consumer A
Consumer B
Consumer C
0
1
0
1
2
0
1



# Pub/Sub

Publish messages/events to many subscribers
Each subscriber gets copy of event to process

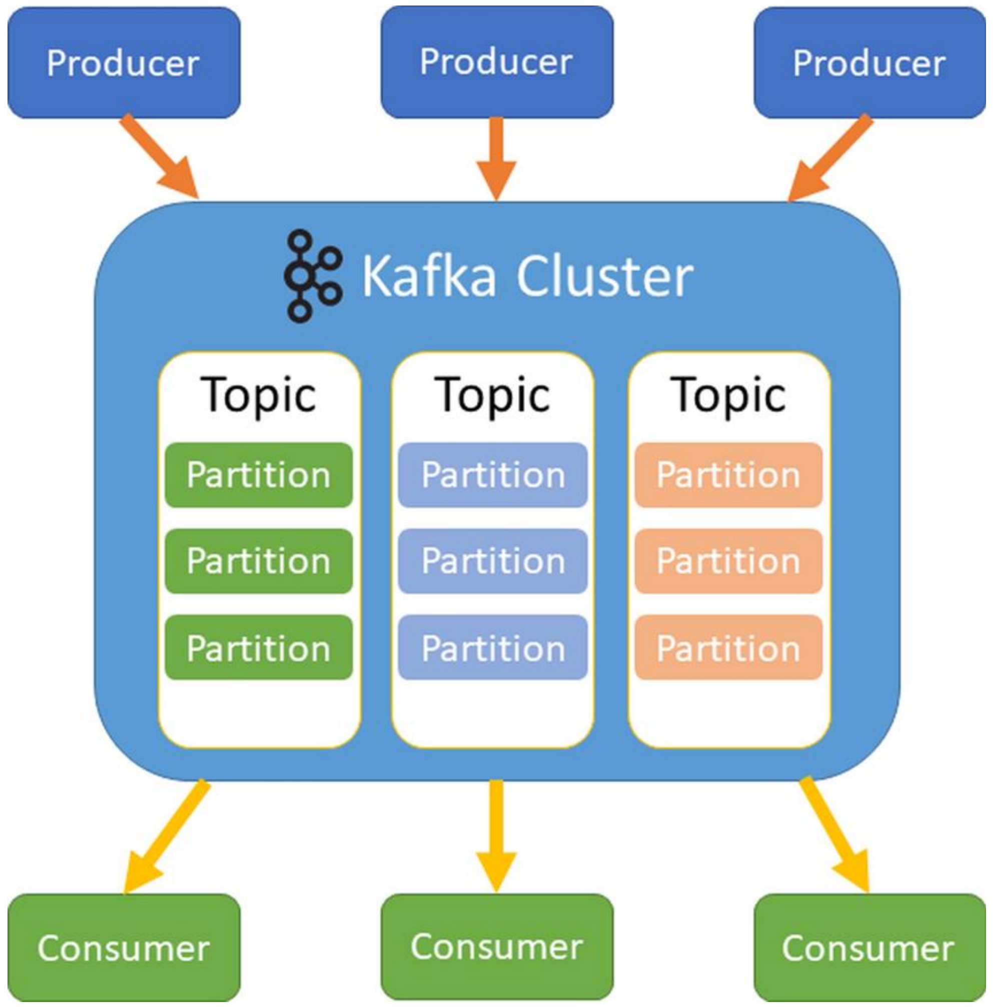

read
Partition 1
Producer
0
1
2
3
4
5
6
7
Consumer
write

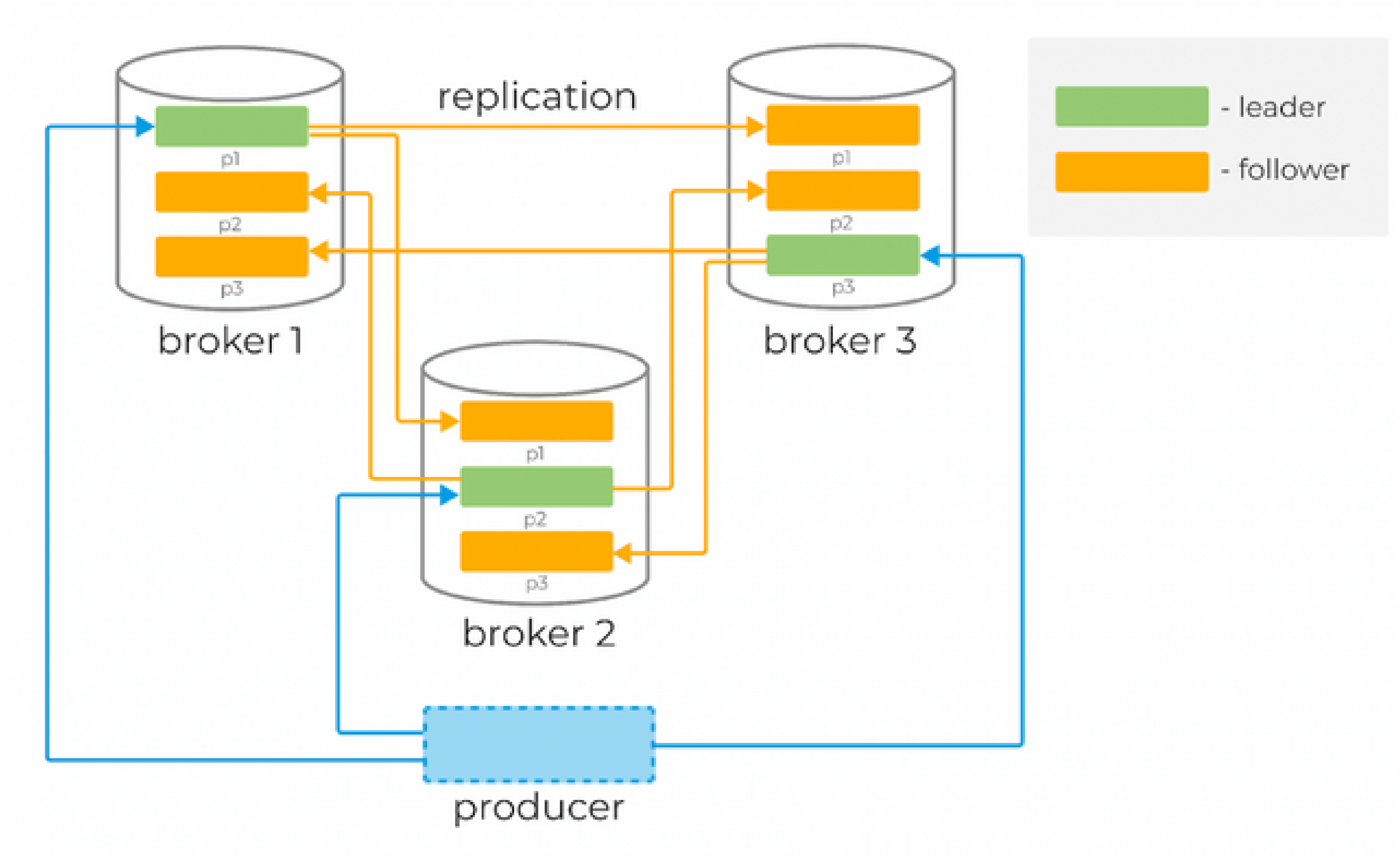

Consumers operate on stream of events
IOT Sensor
Stream
User Interactions
Stream
Constant stream of events

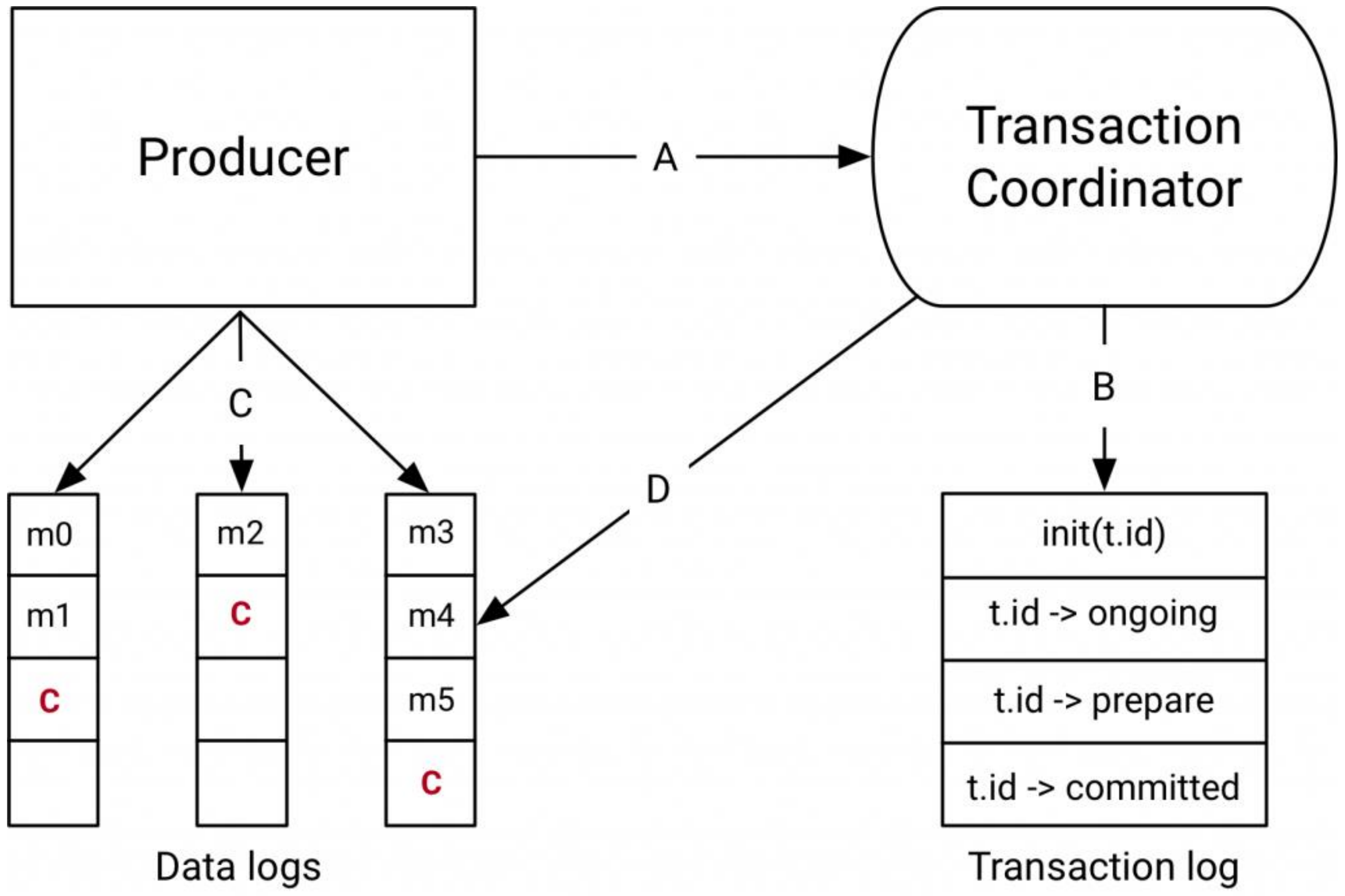
# Streams

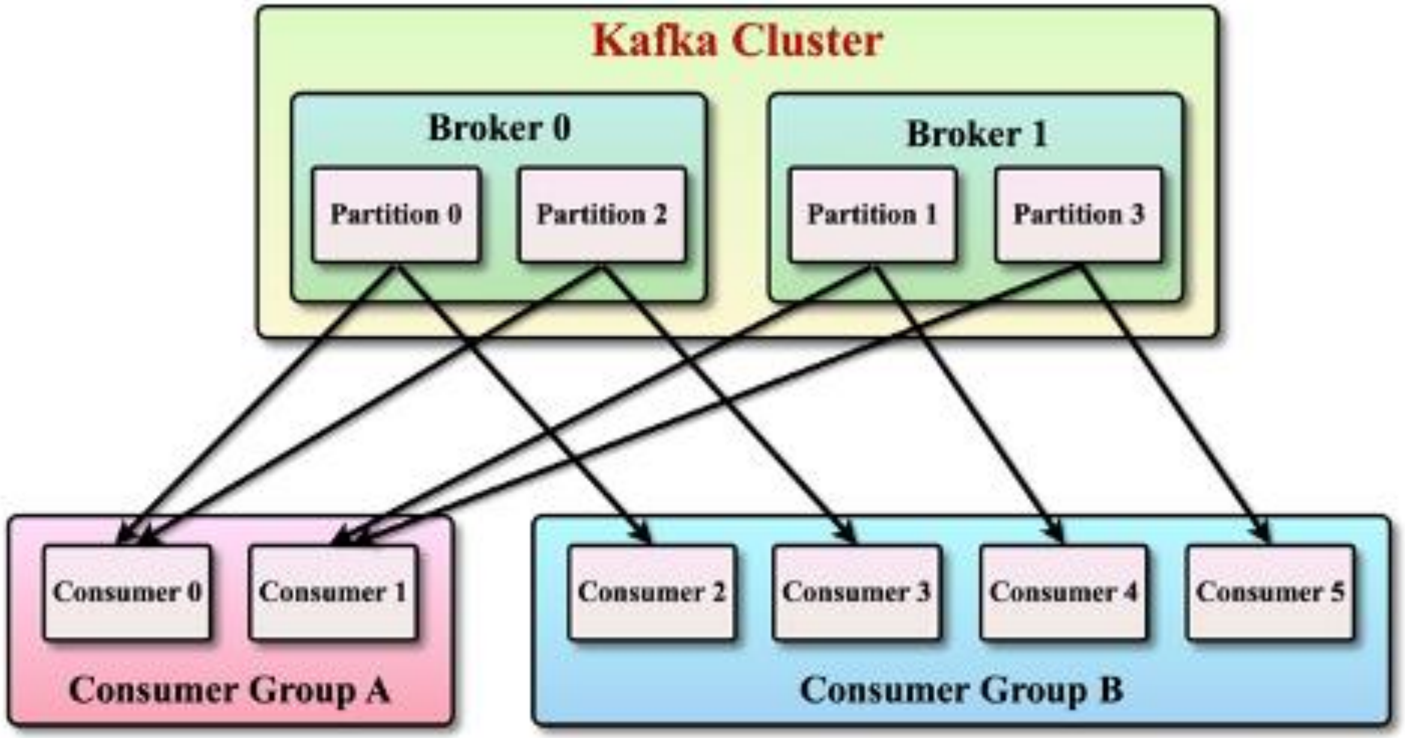
Unbounded series of events

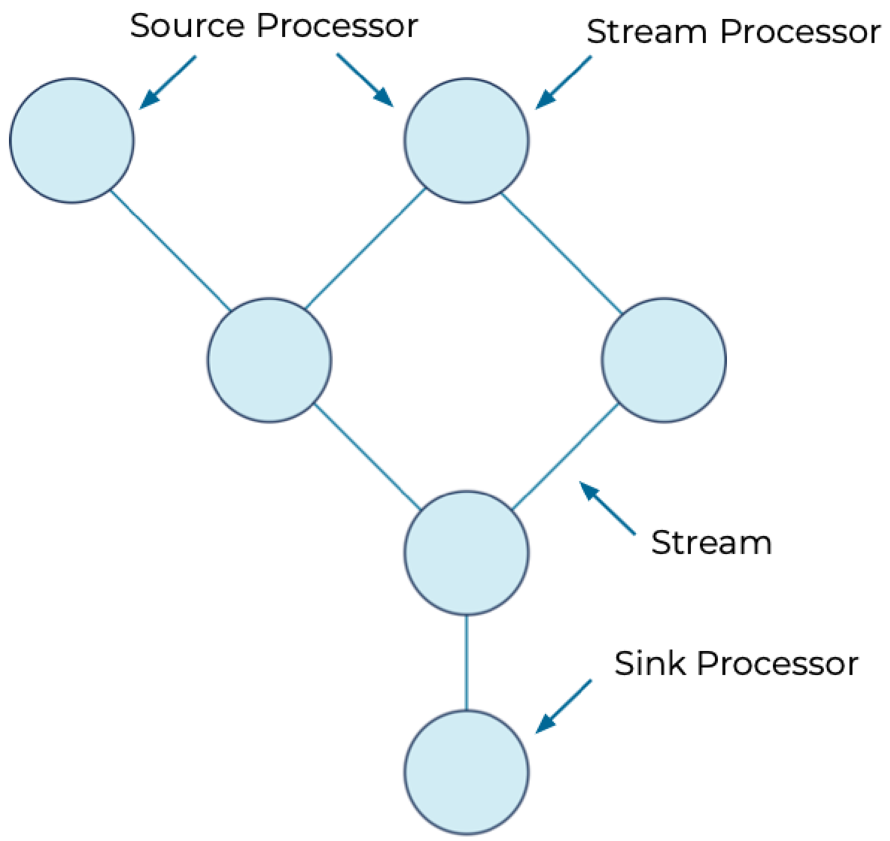
Common examples include user click streams / IOT / transactions

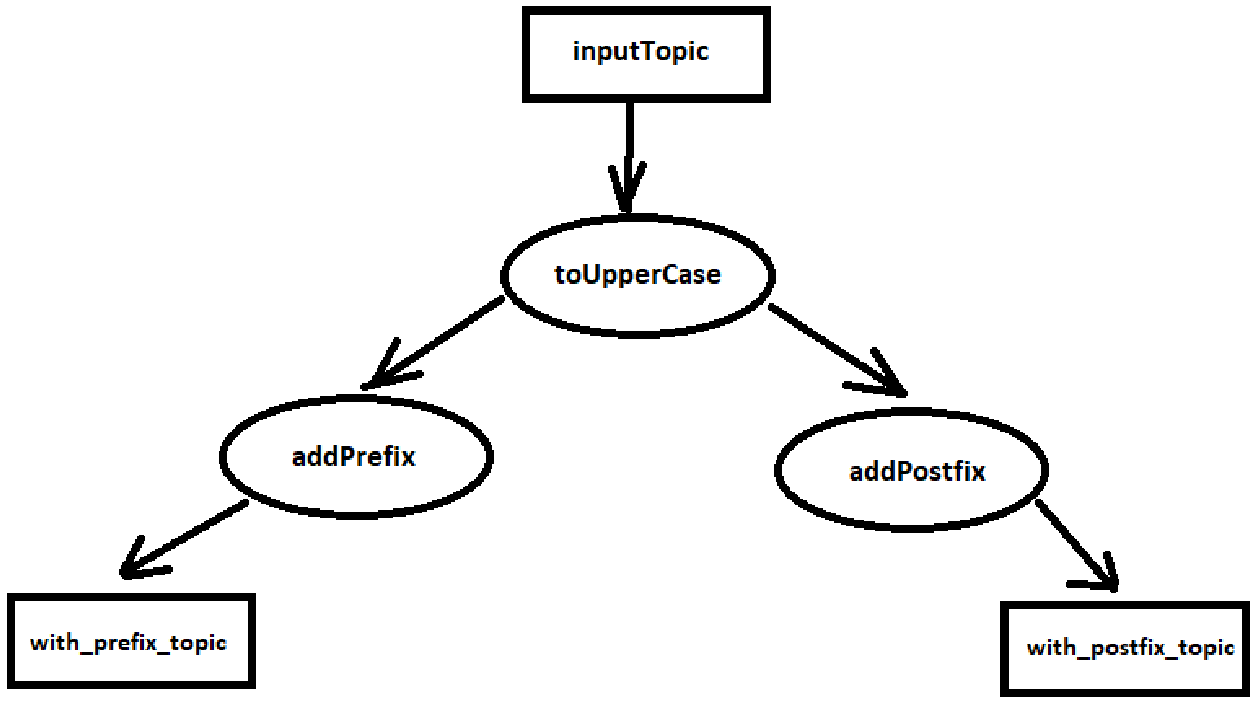

Producer
Producer
Producer
Kafka Cluster
Topic
Partition
Partition
Partition
Topic
Partition
Partition
Partition
Topic
Partition
Partition
Partition
Consumer
Consumer
Consumer

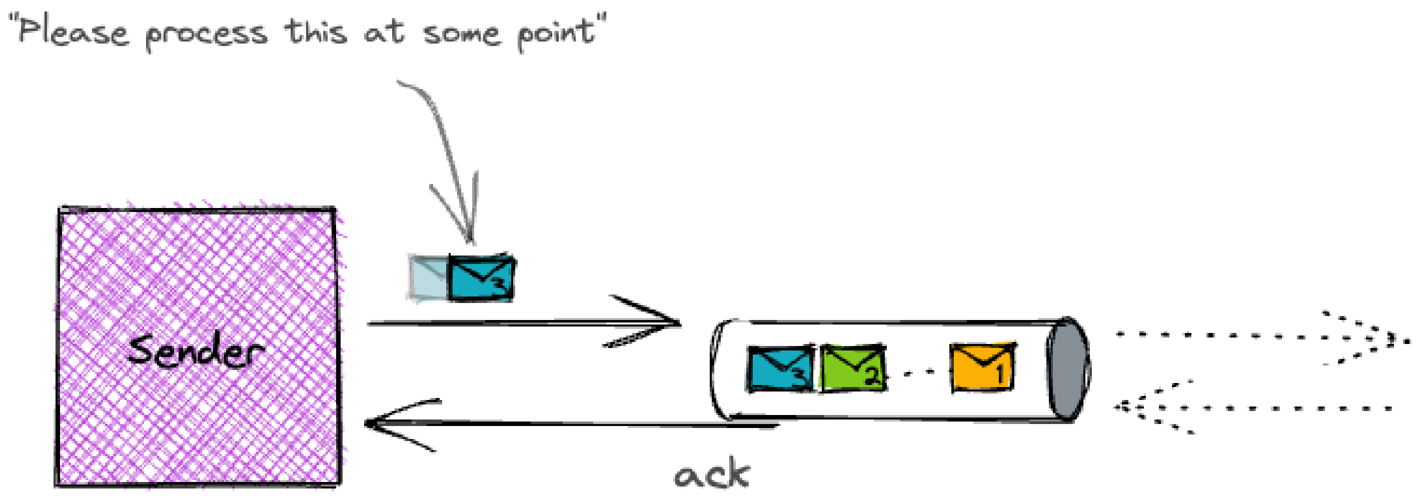

replication
- leader
- follower
p1
p2
p3
broker 1
p1
p2
p3
broker 3
p1
p2
p3
broker 2
producer


Producer
A
Transaction Coordinator
B
C
D
m0
m1
C
m2
C
m3
m4
m5
C
init(t.id)
t.id -> ongoing
t.id -> prepare
t.id -> committed
Data logs
Transaction log


Kafka Cluster
Broker 0
Broker 1
Partition 0
Partition 2
Partition 1
Partition 3
Consumer 0
Consumer 1
Consumer 2
Consumer 3
Consumer 4
Consumer 5
Consumer Group A
Consumer Group B

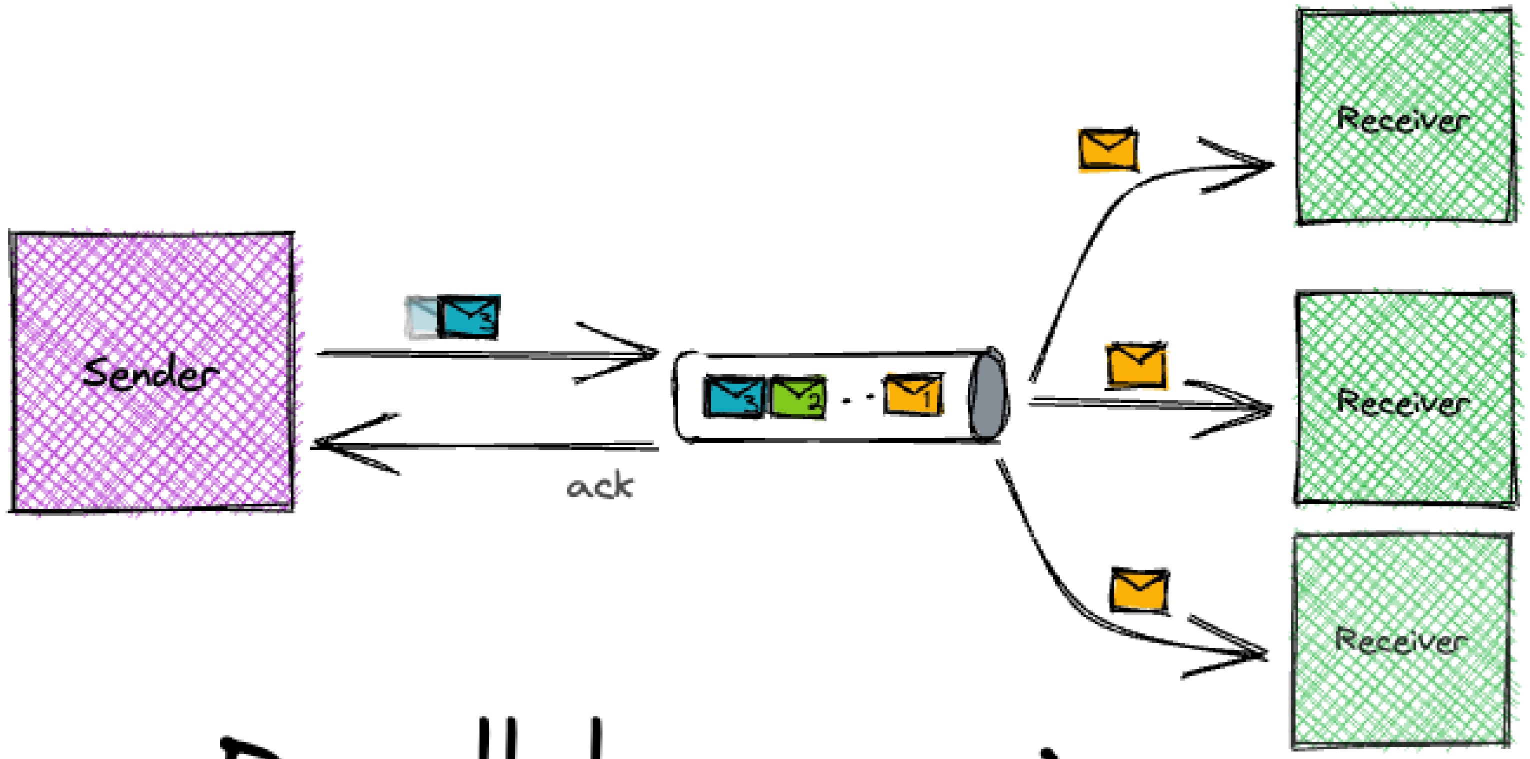

Source Processor
Stream Processor
Stream
Sink Processor


inputTopic
toUpperCase
addPrefix
addPostfix
with_prefix_topic
with_postfix_topic



## Reducing knowledge of systems

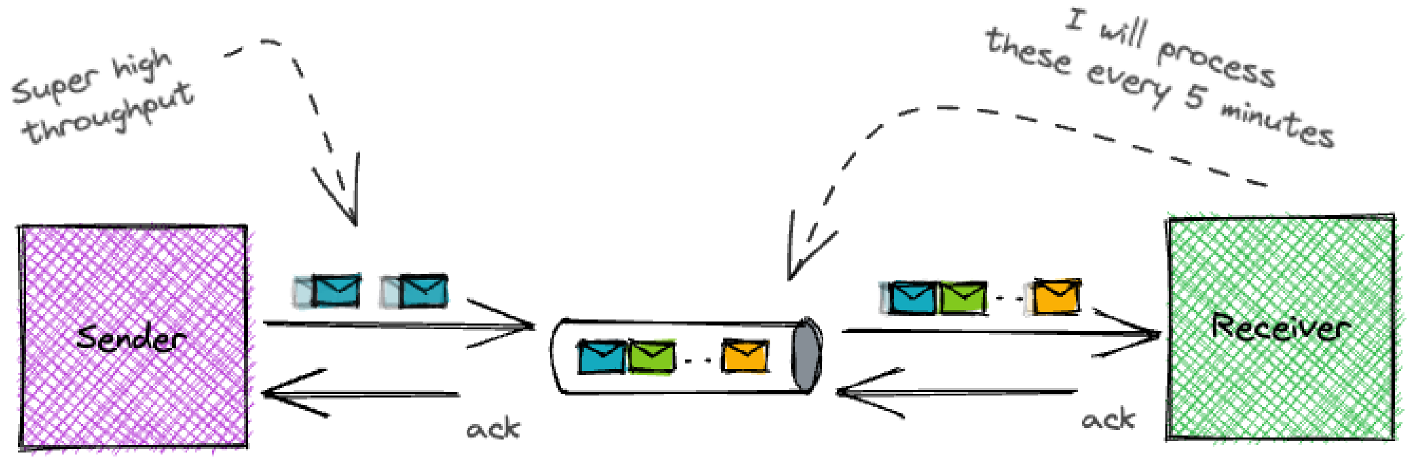
Using messages/events allows us to be decoupled.
This can reduce senders/producers info about the consumers.



## Parallel processing

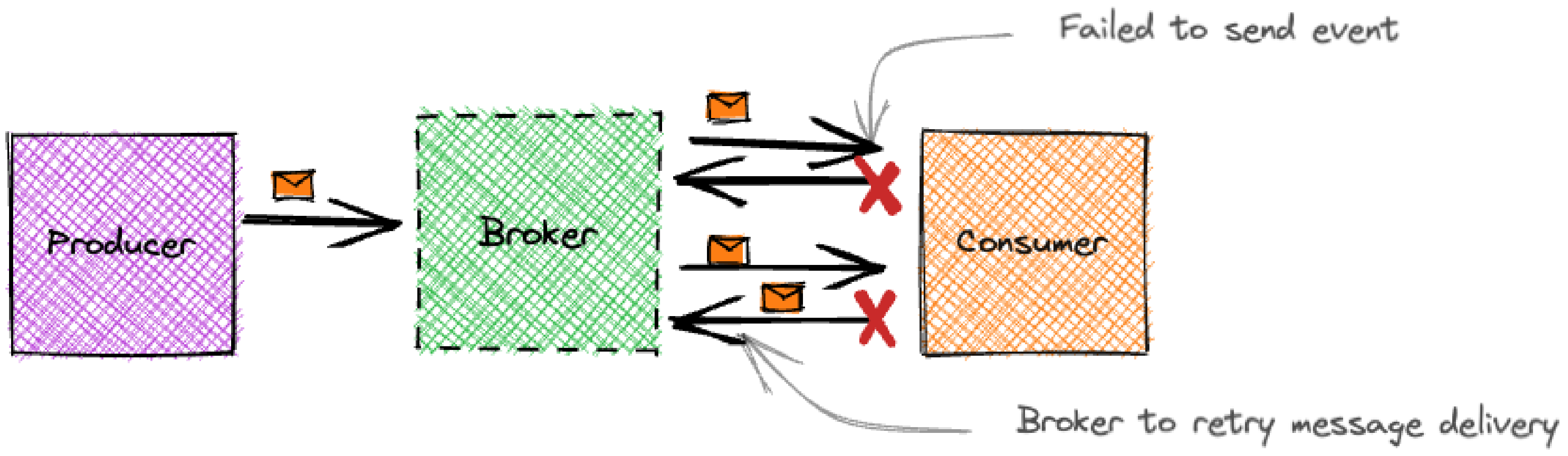
With Pub/Sub notify downstream consumers
Scale as downstream consumers process in parallel

Problems that Kafka Message Brokers solve



## Reduce pressure off downstream consumers

Protect services getting overloaded downstream.
Patterns to consume events at own ingestion rate.

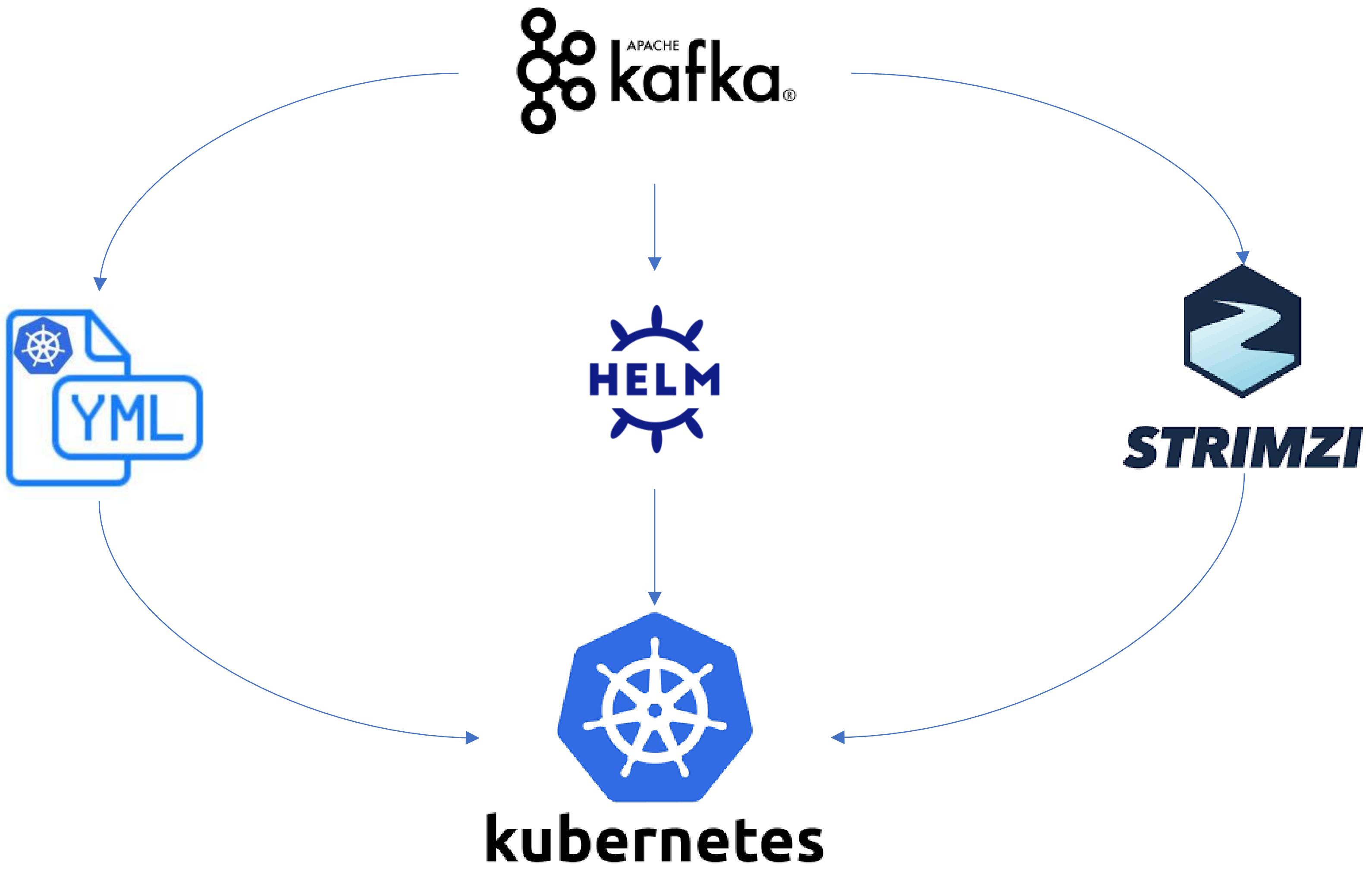


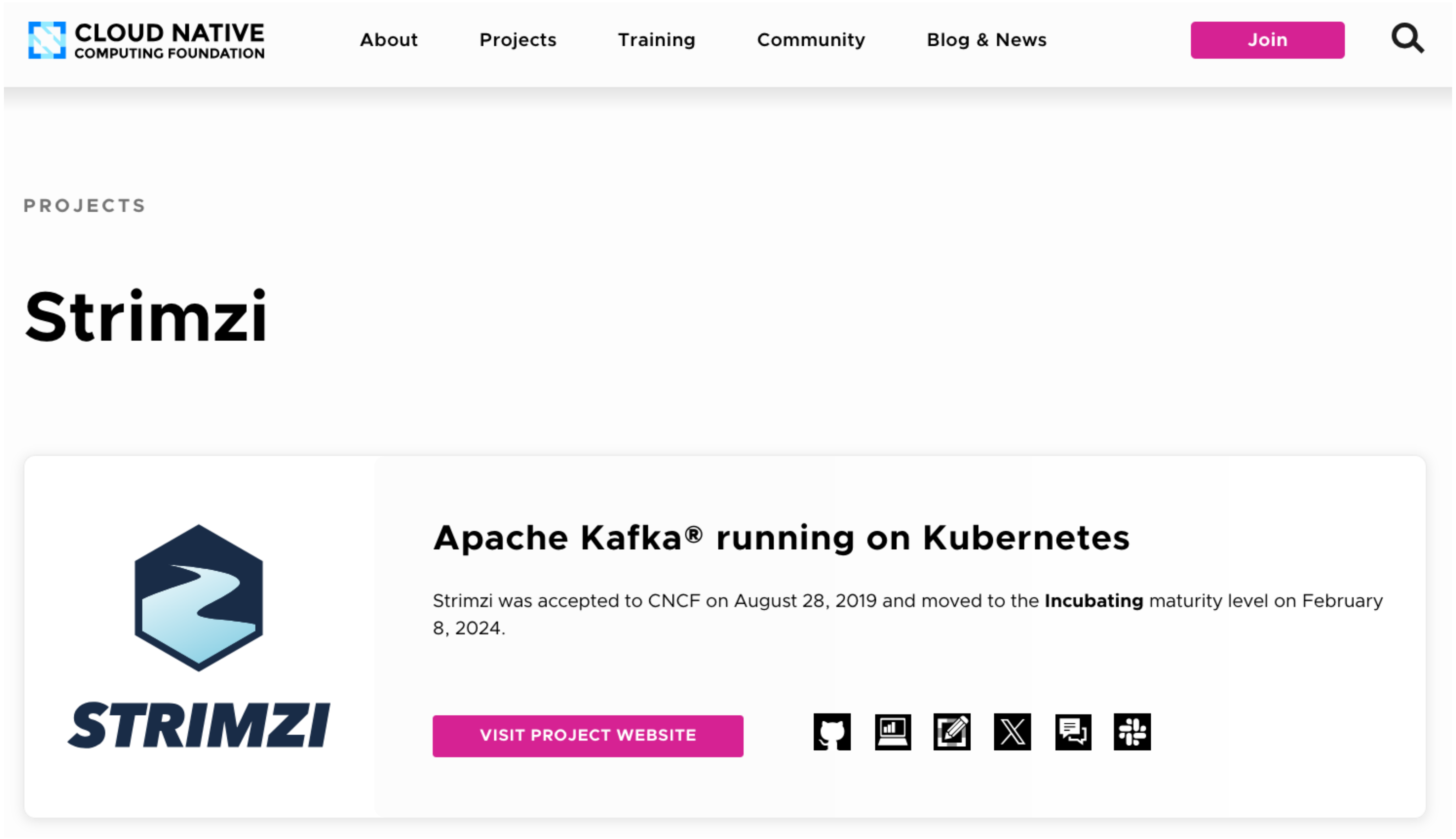
## Prevent messages/data being lost

Messaging pattern is highly scalable
Many consumers can process queue

## Apache Kafka on Kubernetes

# Running Apache Kafka on Kubernetes





CLOUD NATIVE COMPUTING FOUNDATION

About Projects Training Community Blog & News Join

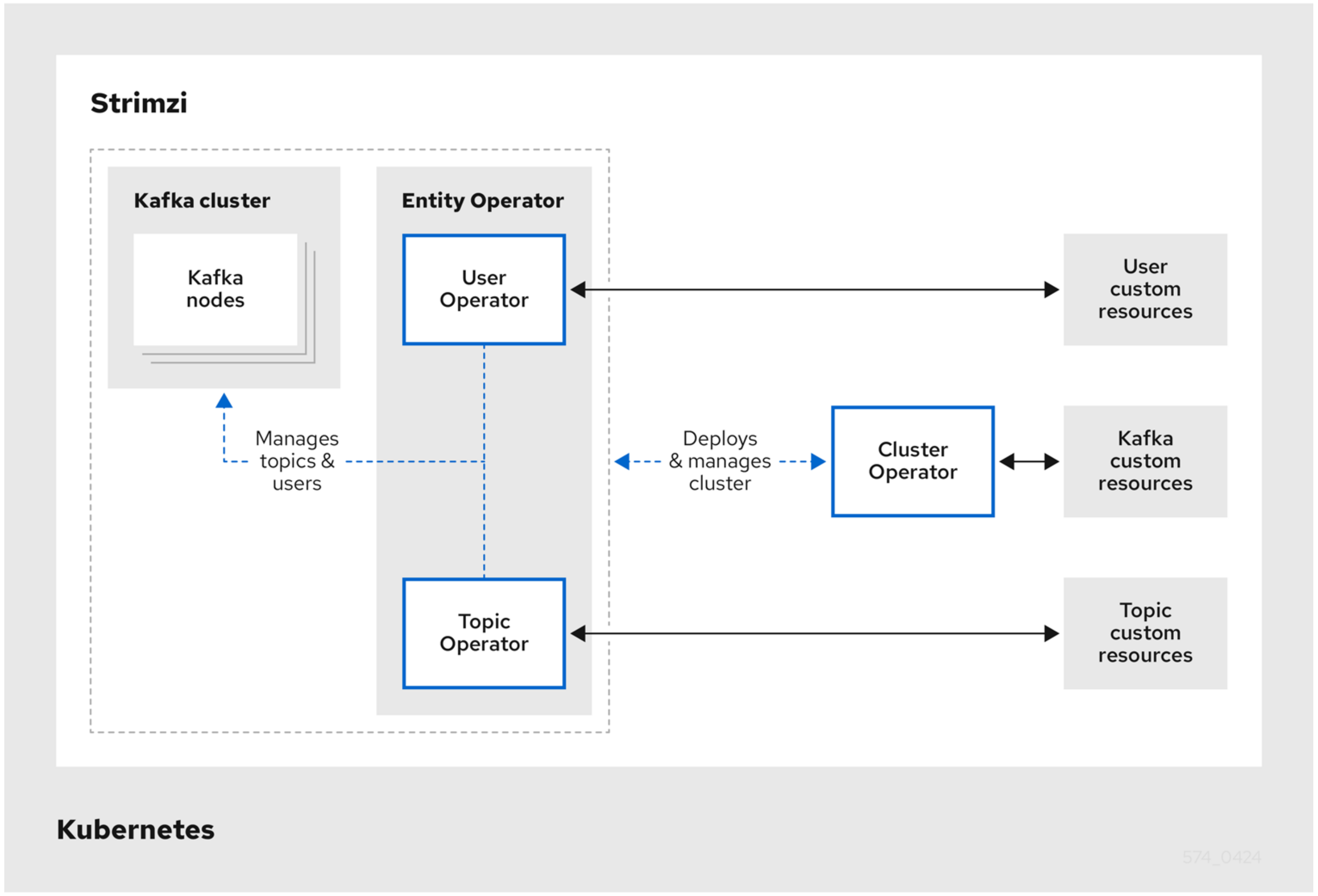
PROJECTS

Strimzi

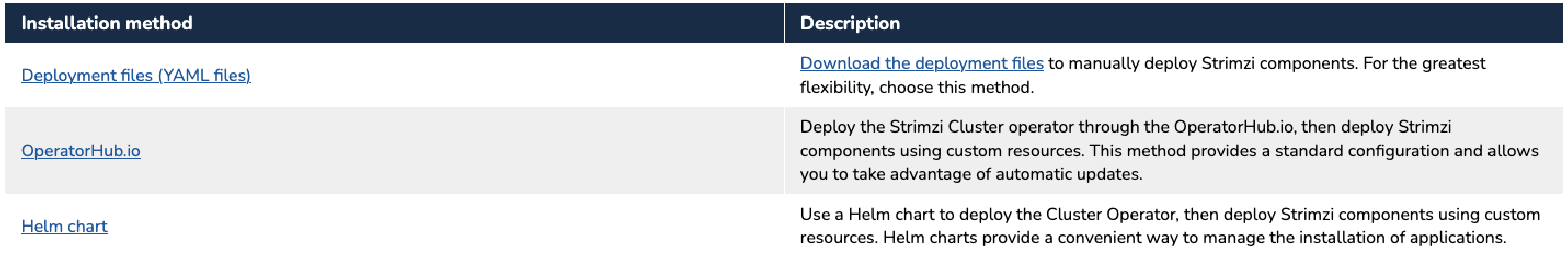
Apache Kafka® running on Kubernetes

Strimzi was accepted to CNCF on August 28, 2019 and moved to the **Incubating** maturity level on February 8, 2024.

VISIT PROJECT WEBSITE


Strimzi
Kafka cluster
Kafka nodes
Entity Operator
User Operator
Topic Operator
Manages topics & users
Deploys & manages cluster
Cluster Operator
User custom resources
Kafka custom resources
Topic custom resources
Kubernetes

# Strimzi Kafka Deployment Options



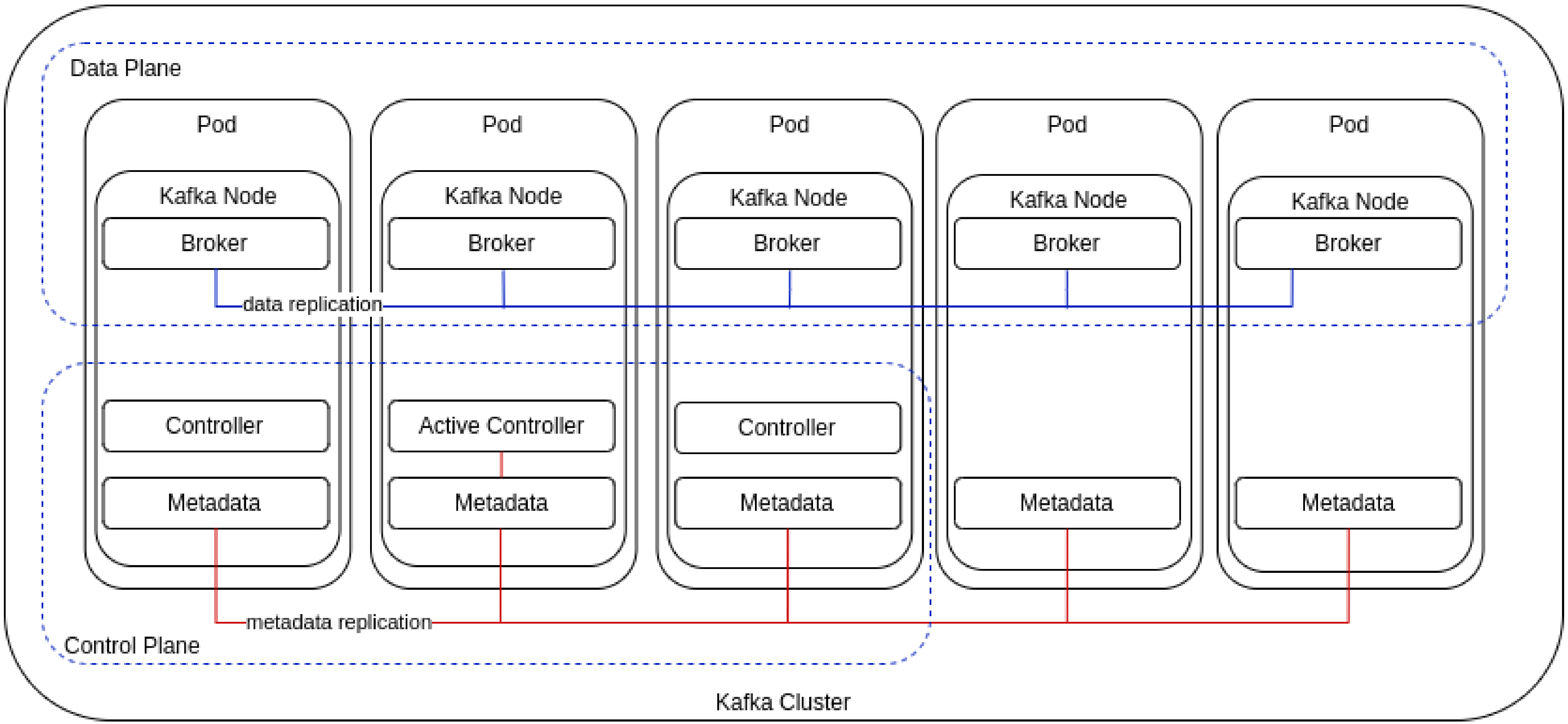
| Installation method | Description |
|---|---|
| Deployment files (YAML files) | Download the deployment files to manually deploy Strimzi components. For the greatest flexibility, choose this method. |
| OperatorHub.io | Deploy the Strimzi Cluster operator through the OperatorHub.io, then deploy Strimzi components using custom resources. This method provides a standard configuration and allows you to take advantage of automatic updates. |
| Helm chart | Use a Helm chart to deploy the Cluster Operator, then deploy Strimzi components using custom resources. Helm charts provide a convenient way to manage the installation of applications. |

```
$ helm repo add strimzi https://strimzi.io/charts
$ helm repo update
```

```
$ helm install strimzi-kafka-operator strimzi/strimzi-kafka-operator \

--version 0.45.0 -n kafka --create-namespace \

-f ./strimzi-operator/strimzi-kafka-values.yaml
```

```
watchAnyNamespace: true

dashboards:
  enabled: true
  namespace: monitoring
```

Custom Helm Values


Data Plane
Pod
Pod
Pod
Pod
Pod
Kafka Node
Kafka Node
Kafka Node
Kafka Node
Kafka Node
Broker
Broker
Broker
Broker
Broker
data replication
Controller
Active Controller
Controller
Metadata
Metadata
Metadata
Metadata
Metadata
metadata replication
Control Plane
Kafka Cluster

```
kubectl api-resources|grep strimzi

strimzipodsets          sps    core.strimzi.io/v1beta2     true    StrimziPodSet
kafkabridges            kb     kafka.strimzi.io/v1beta2    true    KafkaBridge
kafkaconnectors         kctr   kafka.strimzi.io/v1beta2    true    KafkaConnector
kafkaconnects           kc     kafka.strimzi.io/v1beta2    true    KafkaConnect
kafkamirrormaker2s      kmm2   kafka.strimzi.io/v1beta2    true    KafkaMirrorMaker2
kafkamirrormakers       kmm    kafka.strimzi.io/v1beta2    true    KafkaMirrorMaker
kafkanodepools          knp    kafka.strimzi.io/v1beta2    true    KafkaNodePool
kafkarebalances         kr     kafka.strimzi.io/v1beta2    true    KafkaRebalance
kafkas                  k      kafka.strimzi.io/v1beta2    true    Kafka
kafkatopics             kt     kafka.strimzi.io/v1beta2    true    KafkaTopic
kafkausers              ku     kafka.strimzi.io/v1beta2    true    KafkaUser
```

```yaml
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
 name: meet-kafka-cluster
 namespace: meet-kafka-cluster
 annotations:
  strimzi.io/node-pools: enabled
  strimzi.io/kraft: enabled
```

```yaml
spec:
 kafka:
  config:
   offsets.topic.replication.factor: 3
   transaction.state.log.replication.factor: 3
   transaction.state.log.min.isr: 2
   default.replication.factor: 3
   min.insync.replicas: 2
  listeners:
   - name: plain
    port: 9092
    type: internal
    tls: false
   - name: tls
    port: 9093
    type: internal
    tls: true
  version: 3.9.0
```

```yaml
 metricsConfig:
    type: jmxPrometheusExporter
    valueFrom:
     configMapKeyRef:
      name: kafka-metrics
      key: kafka-metrics-config.yml
 entityOperator:
  topicOperator: {}
  userOperator: {}
 cruiseControl: {}
```

```yaml
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaNodePool
metadata:
  name: xs-nodepool
  namespace: meet-kafka-cluster
  labels:
    strimzi.io/cluster: meet-kafka-cluster
spec:
  replicas: 3
  roles:
    - controller
    - broker
  storage:
    type: jbod
    volumes:
      - id: 0
        type: persistent-claim
        size: 1Gi
        deleteClaim: false
```

Persistent storage types:

- persistent-claim - for a single persistent volume
- Jbod - for multiple persistent volumes in a Kafka cluster

Ephemeral storage types:

- ephemeral - uses emptyDir Volumes

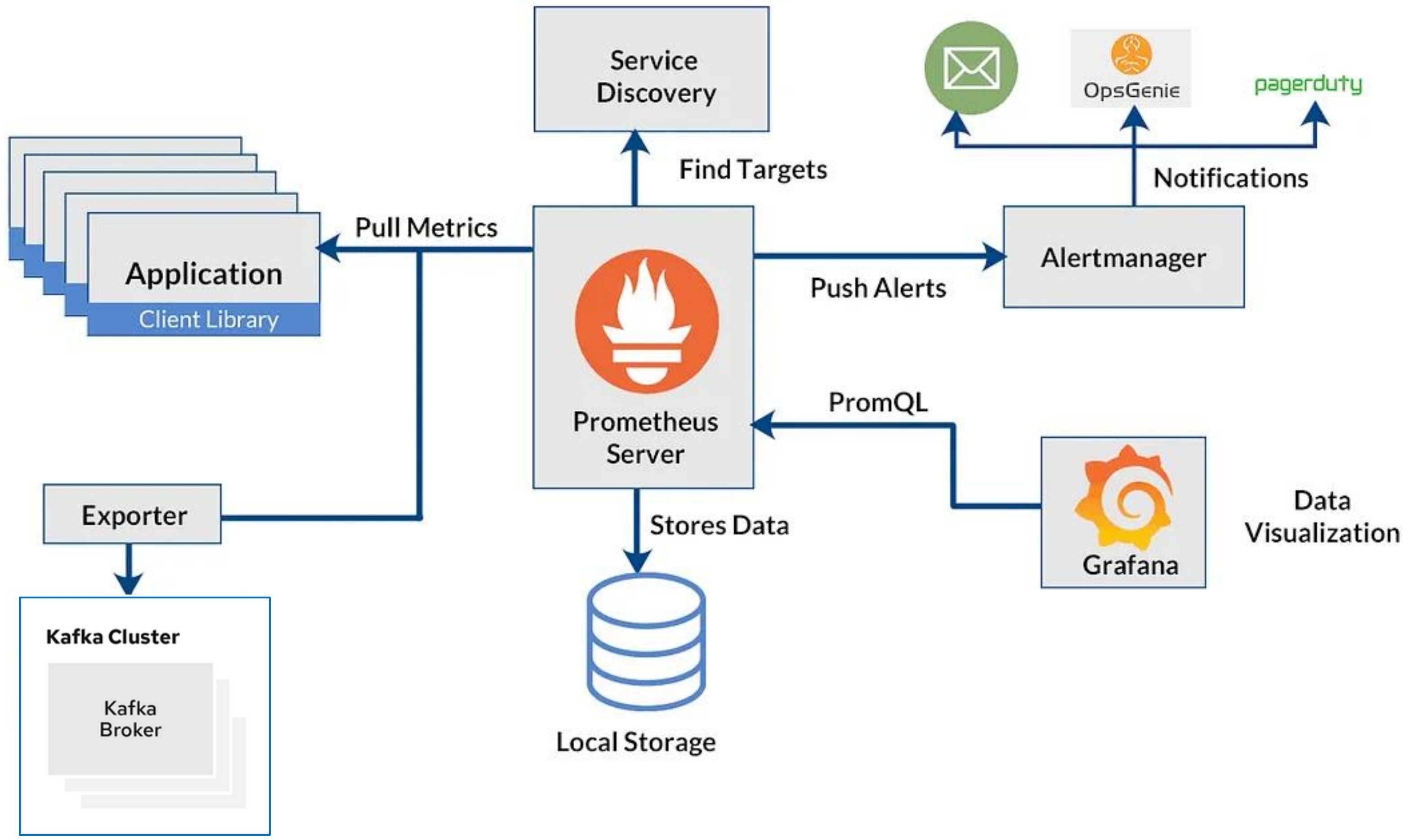
Kafka Node Roles:

- Controller - Manage cluster metadata and state
- Broker - Manage message data
- Dual-Role - All-in-one

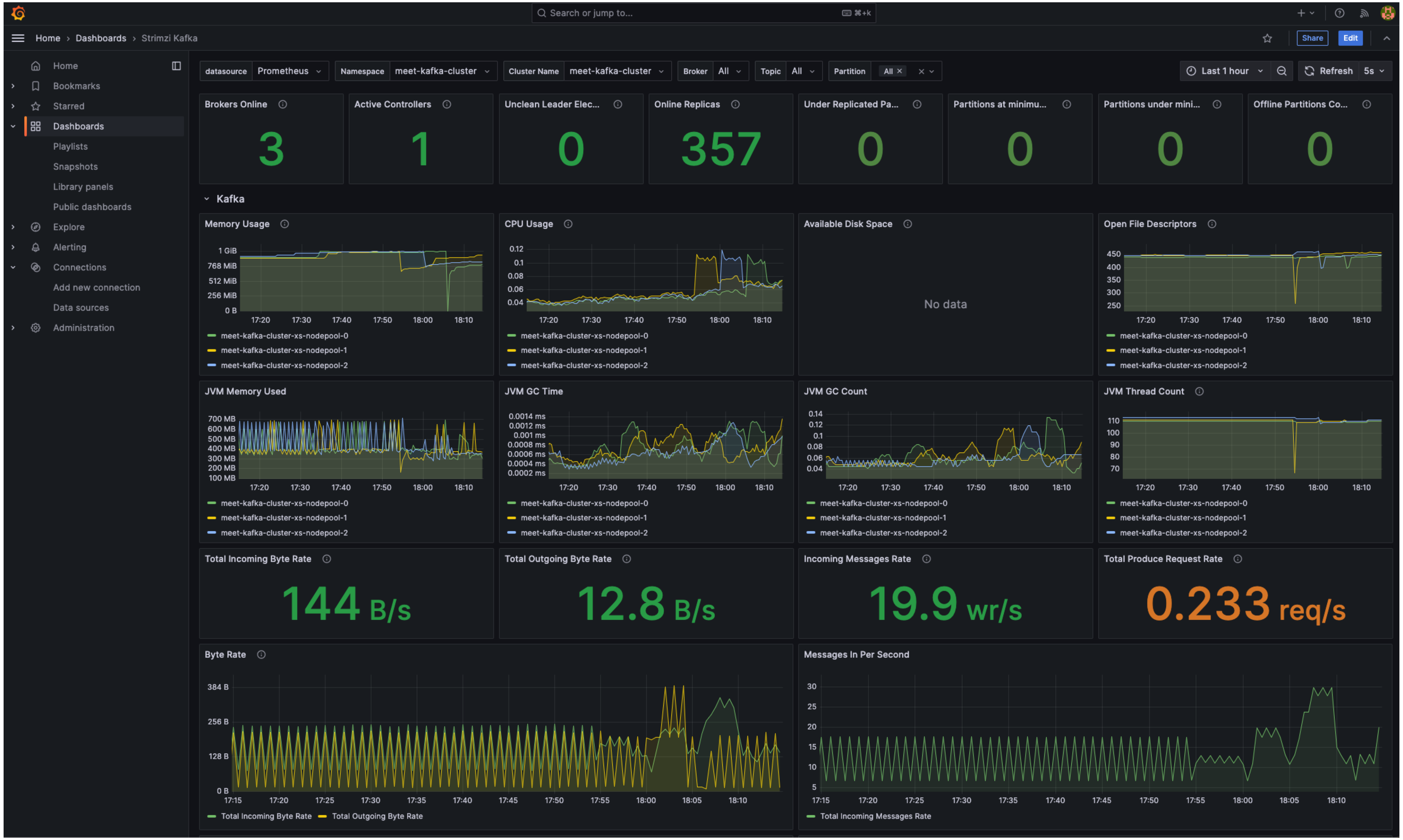
https://strimzi.io/docs/operators/latest/full/deploying.html
#con-config-storage-kraft-str

# Monitoring Apache Kafka on Kubernetes

# Monitoring Kafka



```
❯ kubectl -n monitoring get cm -l grafana_dashboard=1 |grep strimzi

strimzi-cruise-control                          1      3d23h
strimzi-kafka                                   1      3d23h
strimzi-kafka-bridge                            1      3d23h
strimzi-kafka-connect                           1      3d23h
strimzi-kafka-exporter                          1      3d23h
strimzi-kafka-mirror-maker-2                    1      3d23h
strimzi-kafka-oauth                             1      3d23h
strimzi-kraft                                   1      3d23h
strimzi-operators                               1      3d23h
strimzi-zookeeper
```

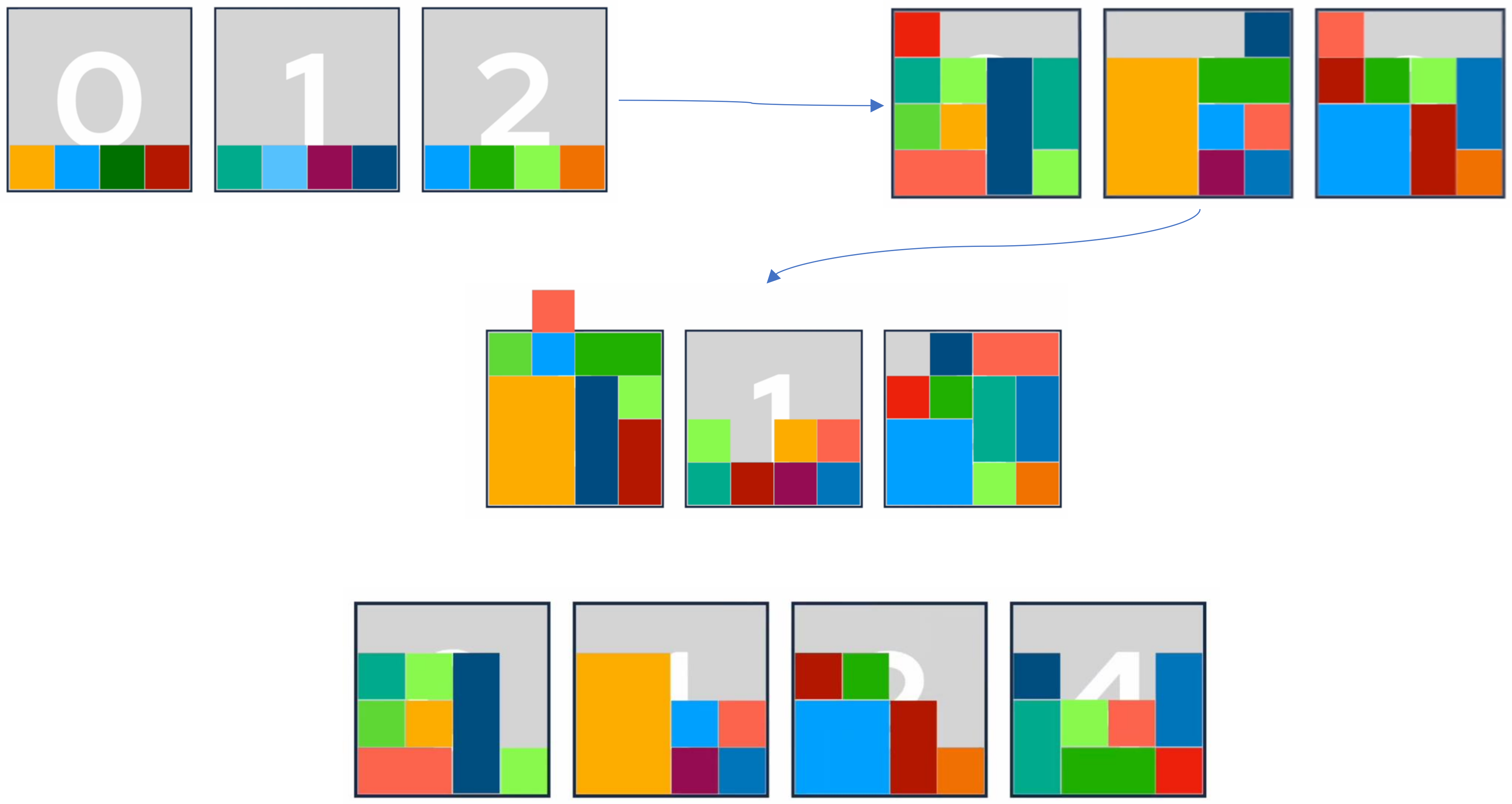


datasource: Prometheus
Namespace: meet-kafka-cluster
Cluster Name: meet-kafka-cluster
Broker: All
Topic: All
Partition: All
Last 1 hour
Refresh 5s

Brokers Online: 3
Active Controllers: 1
Unclean Leader Elec...: 0
Online Replicas: 357
Under Replicated Pa...: 0
Partitions at minimu...: 0
Partitions under mini...: 0
Offline Partitions Co...: 0

Total Incoming Byte Rate: 144 B/s
Total Outgoing Byte Rate: 12.8 B/s
Incoming Messages Rate: 19.9 wr/s
Total Produce Request Rate: 0.233 req/s

## Broker balancing



## Kafka Cluster Balancing

- What is the Optimal partitions/replica Distribution?
  - Rack Awareness?
  - Distribution of Leader replicas
  - Optimal Network, CPU, storage, or RAM utilization?
- Manual or automated optimizations?

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaRebalance
metadata:
  name: meet-cluster-rebalance
  labels:
    strimzi.io/cluster: meet-kafka-cluster
# Use the default Cruise Control optimization goals
spec: {}
```

CruizeControl CRD

```
NAME                      CLUSTER              TEMPLATE   STATUS
meet-cluster-rebalance   meet-kafka-cluster              ProposalReady
```

```
Name:         meet-cluster-rebalance
Namespace:    meet-kafka-cluster
Labels:       strimzi.io/cluster=meet-kafka-cluster
Annotations:  <none>
API Version:  kafka.strimzi.io/v1beta2
Kind:         KafkaRebalance
Metadata:
  Creation Timestamp:  2025-01-20T21:30:18Z
  Generation:          1
  Resource Version:    580148
  UID:                 d836fa04-4715-4f48-b68c-84dd0b6dff37
Spec:
Status:
  Conditions:
    Last Transition Time:  2025-01-20T21:30:18.754838993Z
    Status:                True
    Type:                  ProposalReady
```

```
Observed Generation:    1
  Optimization Result:
    After Before Load Config Map:  meet-cluster-rebalance
    Data To Move MB:               0
    Excluded Brokers For Leadership:
    Excluded Brokers For Replica Move:
    Excluded Topics:
    Intra Broker Data To Move MB:        0
    Monitored Partitions Percentage:     100
    Num Intra Broker Replica Movements:  0
    Num Leader Movements:                3
    Num Replica Movements:               0
    On Demand Balancedness Score After:   89.4347095948149
    On Demand Balancedness Score Before:  89.4347095948149
    Provision Recommendation:
    Provision Status:                RIGHT_SIZED
    Recent Windows:                  1
…
```

## Best Practices

- Install the Strimzi operator in a separate namespace from the Kafka cluster and other Kafka components it manages
  - Ensure clear separation of resources and configurations.
  - Avoid the issues associated with installing multiple Strimzi operators in a Kubernetes cluster
- Use a single Strimzi operator to manage all your Kafka instances within a Kubernetes cluster.
- Update the Strimzi operator and the supported Kafka version as often as possible
  - to reflect the latest features and enhancements.

## Best Practices

- By default, a single replica of the Cluster Operator is deployed.
  - Add extra stand-by replicas in case of disruption.
  - One replica is elected leader

- The Cluster Operator watches for updates in the namespaces where the Kafka resources are deployed.
  - We can specify which namespaces to watch in the Kubernetes cluster.
  - Watching multiple selected namespaces has the most impact on performance due to increased processing overhead.

- To optimize performance for namespace monitoring, it is generally recommended to either watch:
  - a single namespace
  - monitor the entire cluster.

# Thank you!

dojobits

https://dojobits.io/