Running Apache Kafka on Kubernetes with Strimzi

Marko Štrukelj

OpenCon 2023

OpenCon.si

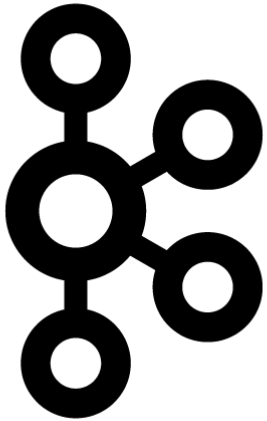
https://strimzi.io

STRIMZI

About me

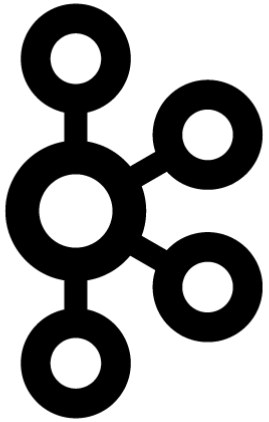
- Java based software developer since Java 1.1
- OpenBlend Slovenian JUG founding member and contributo
- Strimzi Kafka OAuth component lead developer (https://github.com/strimzi/strimzi-kafka-oauth)
- Working for RedHat

https://github.com/mstruk



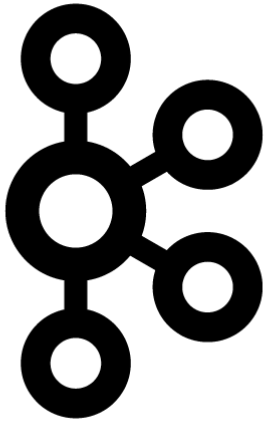
Agenda

- Why Kafka
- Why running Kafka on Kubernetes
- Strimzi



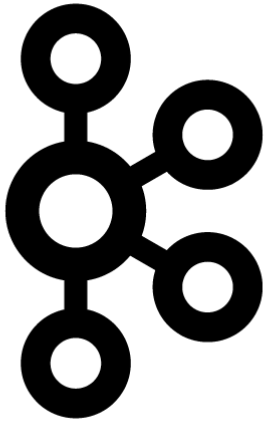
Apache Kafka

- Not just a Pub-Sub messaging system
- Distributed event streaming platform
  - Combines Delivery + Storage + Processing
  - Made for real time message processing
- Huge ecosystem of tools, clients, libraries, connectors
  - All kinds of integrations available with different data stores / eventing systems



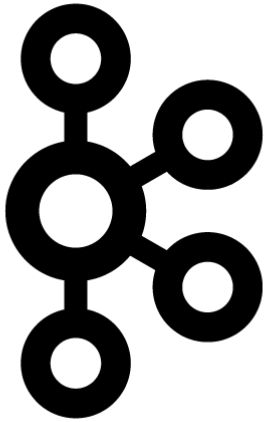
Why Kafka?

- Kafka-centric microservices architecture
  - Can serve as a central message bus between microservices
- High performance, scalability, and availability
- Reliable, durable, fault tolerant
  - Multiple brokers, sharded data
  - Automatic data replication



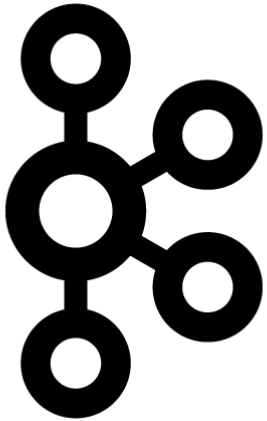
How does it do it?

- Multi-node cluster logic with built-in leader election
- Smart clients connected to multiple brokers as needed
- Distributed transaction log (using zero-copy file append OS kernel facility)
- Efficient TCP based protocol
  - Does not decode the messages, just dump them to disk
  - Allows for high throughput ingest (client-side batching / buffering)
  - Configurable reliability (configurable acks)



Why Kafka on K8s?

- Kubernetes is designed as distributed and scalable
  - Kafka is distributed by nature and is good at scaling out
  - Workloads using Kafka are also distributed and scalable
- Kubernetes is a great abstraction
  - K8S provides primitives for running your software almost everywhere!



How Kafka on K8s?

- Make everything feel like a cloud!
  - Use the operator pattern to deploy cluster “on-demand”
- Why should you learn something new when you already know K8S or OpenShift?

Strimzi

- Running Apache Kafka on Kubernetes
- Started by Red Hat in 2018
- Used as a basis for AMQ Streams on OCP (OpenShift Container Platform)

Web site: http://strimzi.io/

GitHub: https://github.com/strimzi

CLOUD NATIVE
COMPUTING FOUNDATION

Strimzi

- Open source project licensed under Apache License 2.0
- Part of the Cloud Native Computing Foundation (CNCF)
- Provides several components:
  - Container images for Apache Kafka and Apache Zookeeper
  - Operators for managing and configuring Kafka clusters, topics, users or connector
  - HTTP Bridge, OAuth libraries, ...

Strimzi

- Kubernetes native experience
  - Use kubectl to manage Kafka
- Operator based
- Makes Kafka GitOps friendly
- Secure by default
  - Inter-broker and broker to zookeeper connectivity automatically secured with mTLS

Operators

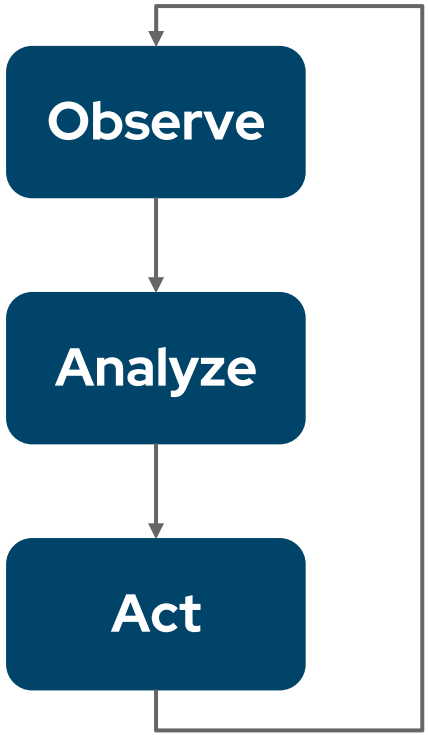
- Pattern for running applications on Kubernetes
- Deploying software is easy
  - YAMLs, Helm Charts can do installation …
  - … but operators do it better!

Operators

- Keeping the software running is hard
  - YAMLs or Helm Charts offer only little help here
  - Operators can take the applications through the whole lifecycle
    - Upgrades, cluster balancing, security, configuration changes
- Operator process communicates with Kubernetes API server
  - Creates and manages Pods, Services and other native K8s resources
  - Uses custom Kubernetes resource definitions

Operators

- Observe
  - Get the desired and current state
- Analyze
  - Compare the two states and find the differences
- Act
  - Fix the differences to make sure the desired and current state are identical



Operators

- A whole ecosystem
  - OperatorHub.io
    - Find and use an existing operator
  - OperatorFramework.io
    - SDK with APIs to develop your own operator

Installing Strimzi

- Create a Kubernetes namespace
- Deploy Kubernetes custom resource definitions used by Strimzi operators and the cluster operator:

```
kubectl create -f 'https://strimzi.io/install/latest?namespace=kafka' -n kafka
```

Creating a Kafka cluster

- Create a Kafka custom resource handled by Strimzi operator:

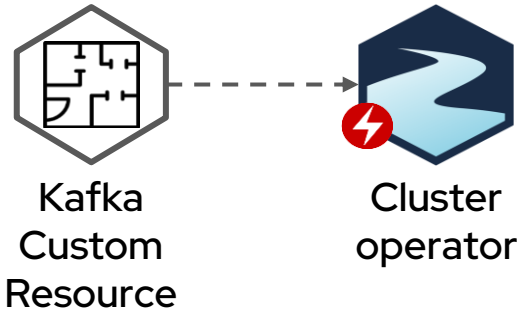
```
kubectl apply -f https://strimzi.io/examples/latest/kafka/kafka-persistent-single.yaml -n kafka
```

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    version: 3.6.0
    replicas: 1
    listeners:
      - name: plain
        port: 9092
        type: internal
        tls: false
      - name: tls
        port: 9093
        type: internal
        tls: true
    config:
      offsets.topic.replication.factor: 1
      transaction.state.log.replication.factor: 1
      transaction.state.log.min.isr: 1
      default.replication.factor: 1
      min.insync.replicas: 1
      inter.broker.protocol.version: "3.6"
    storage:
      type: jbod
      volumes:
        - id: 0
          type: persistent-claim
          size: 100Gi
          deleteClaim: false
  zookeeper:
    replicas: 1
    storage:
      type: persistent-claim
      size: 100Gi
      deleteClaim: false
  entityOperator:
    topicOperator: {}
    userOperator: {}
```

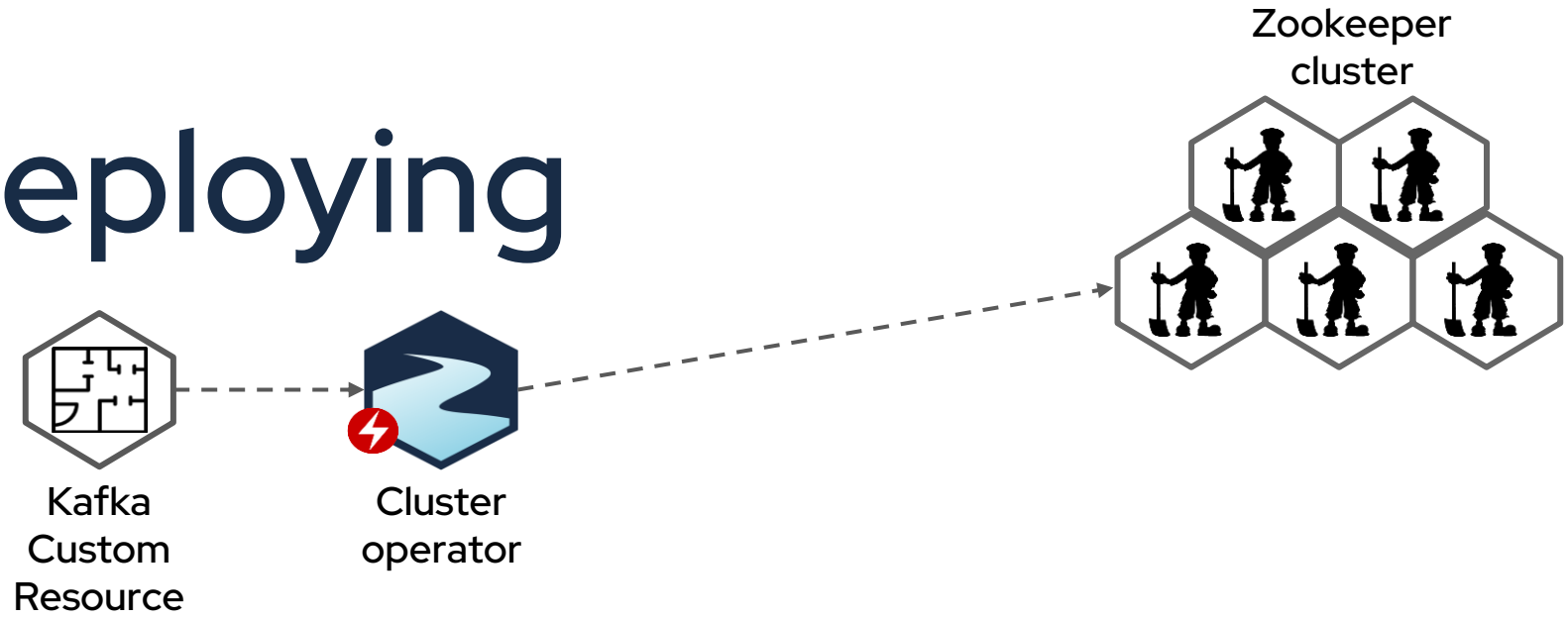
Deploying

Cluster operator

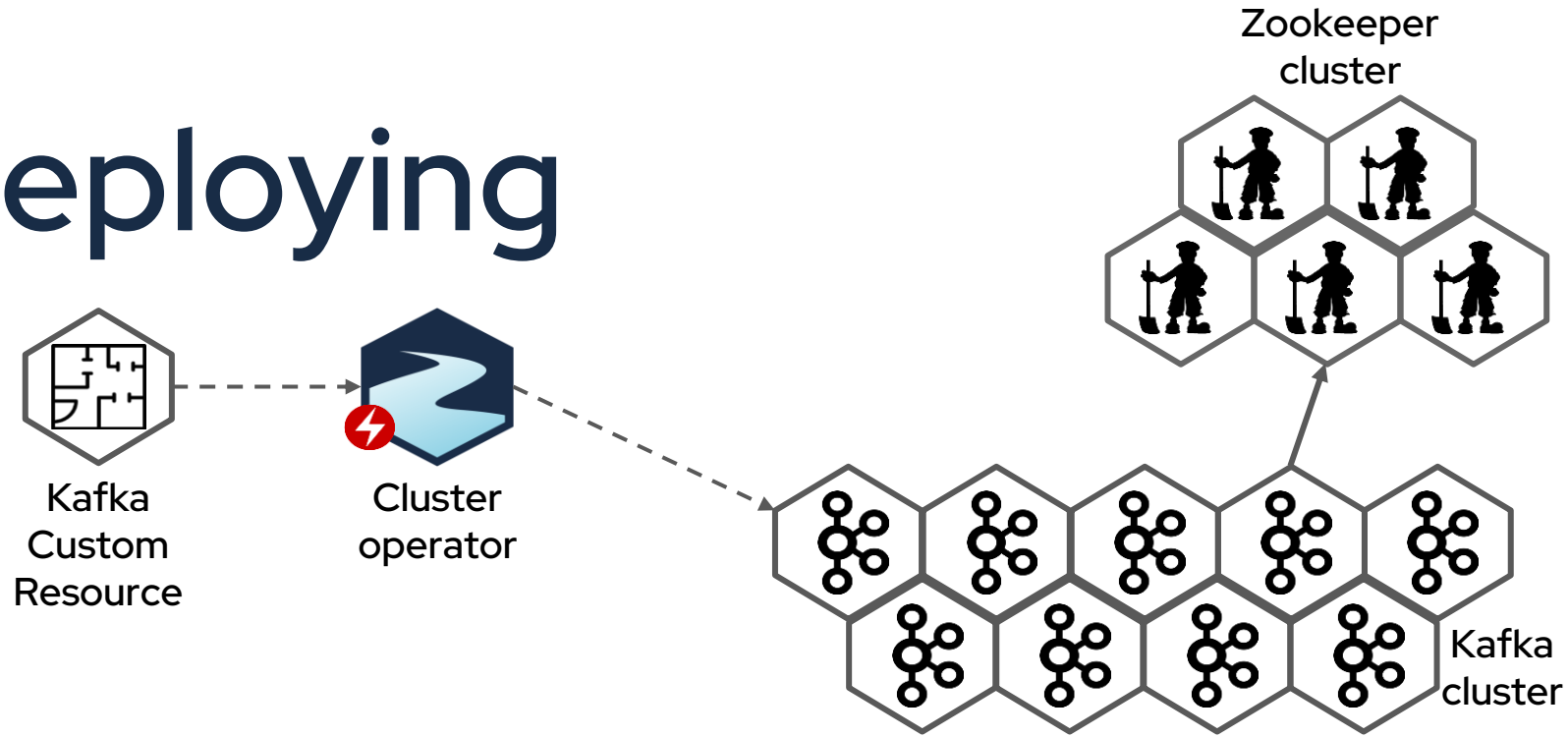
Deploying



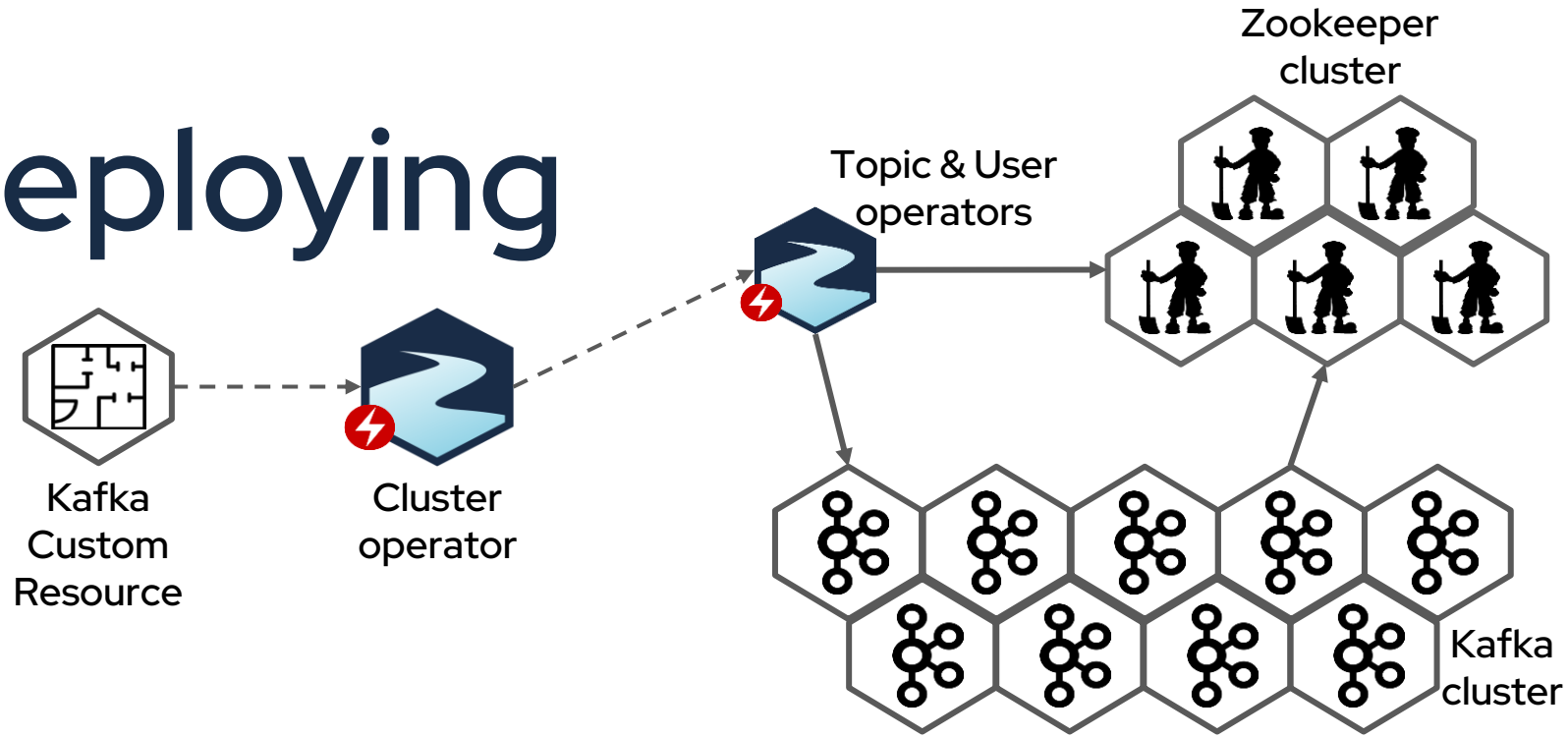
Deploying


Zookeeper cluster
Kafka Custom Resource
Cluster operator

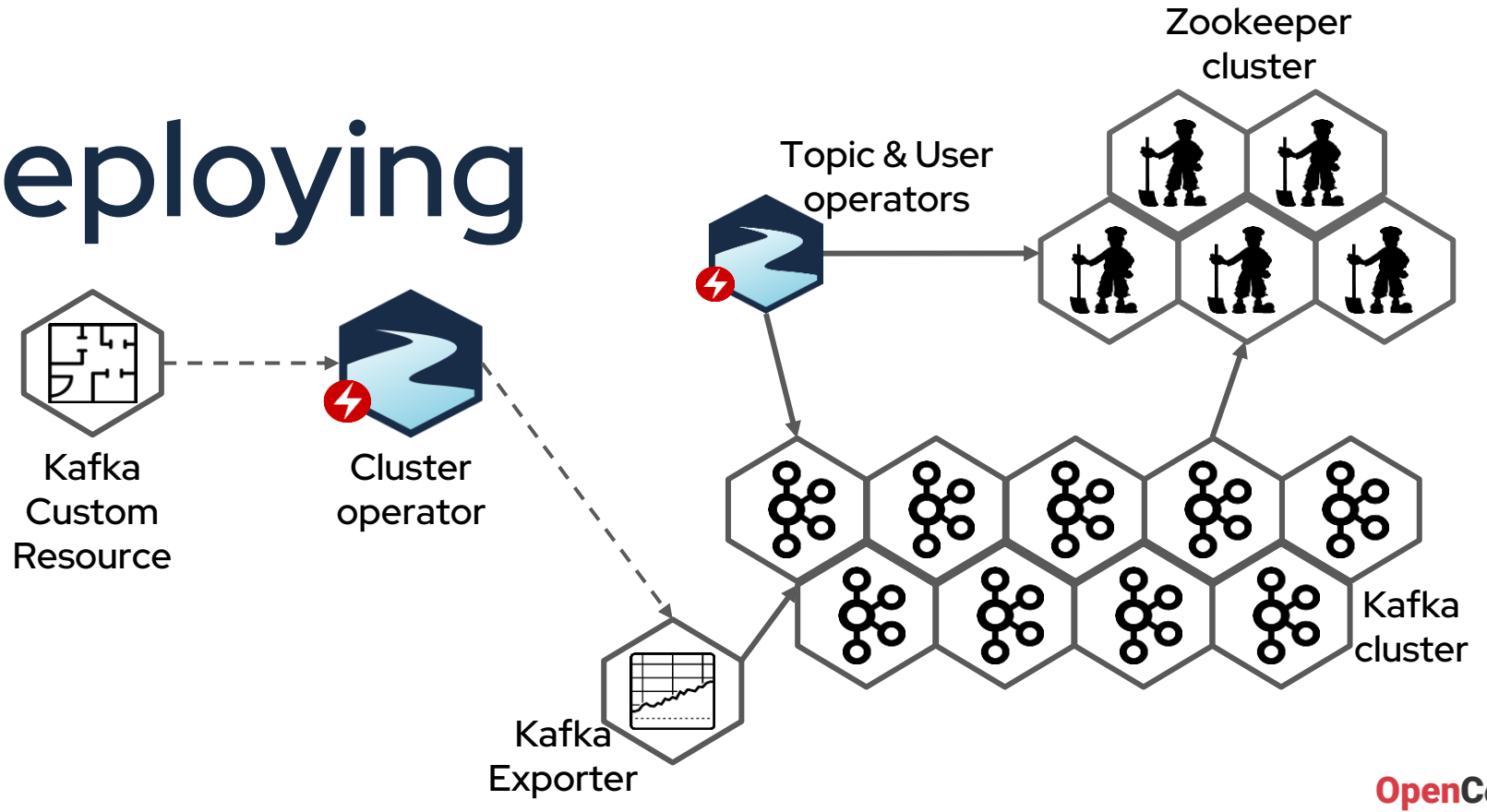
Deploying


Zookeeper cluster
Kafka Custom Resource
Cluster operator
Kafka cluster

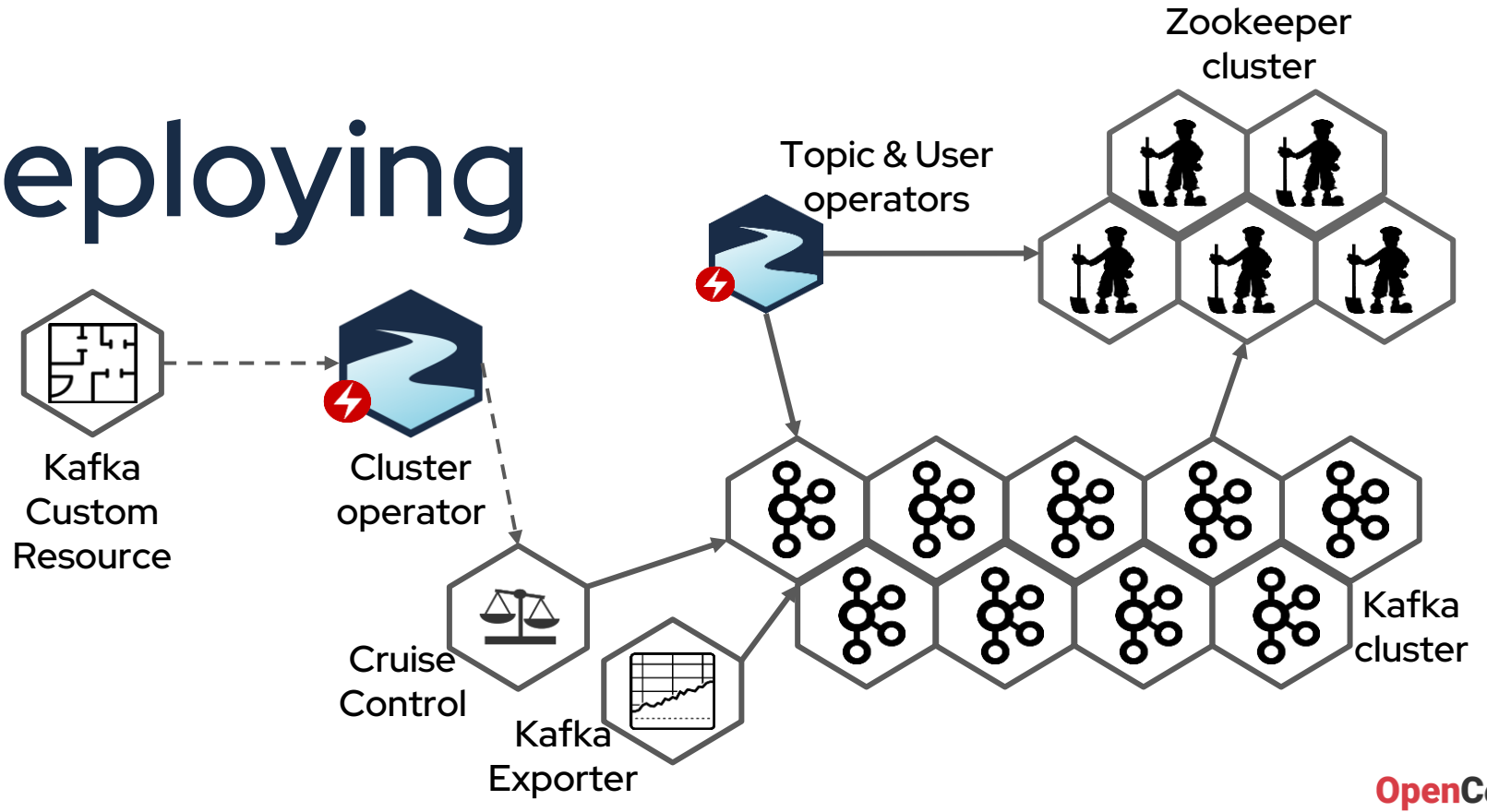
Deploying


Zookeeper cluster
Topic & User operators
Kafka Custom Resource
Cluster operator
Kafka cluster

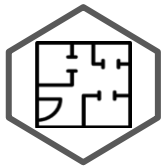
Deploying


Zookeeper cluster
Topic & User operators
Kafka Custom Resource
Cluster operator
Kafka cluster
Kafka Exporter

Deploying


Zookeeper cluster
Topic & User operators
Kafka Custom Resource
Cluster operator
Cruise Control
Kafka Exporter
Kafka cluster

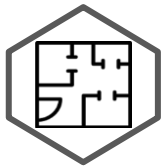
Deploying



Kafka
Custom
Resource

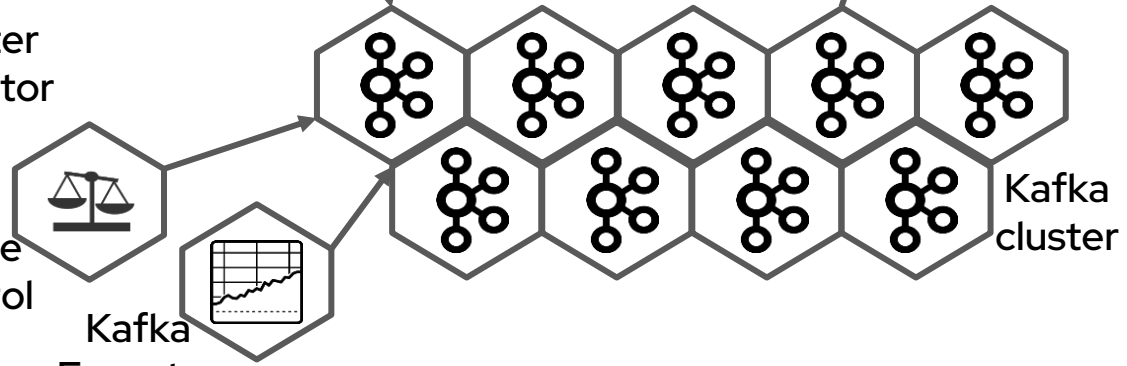
Cluster
operator

Managing

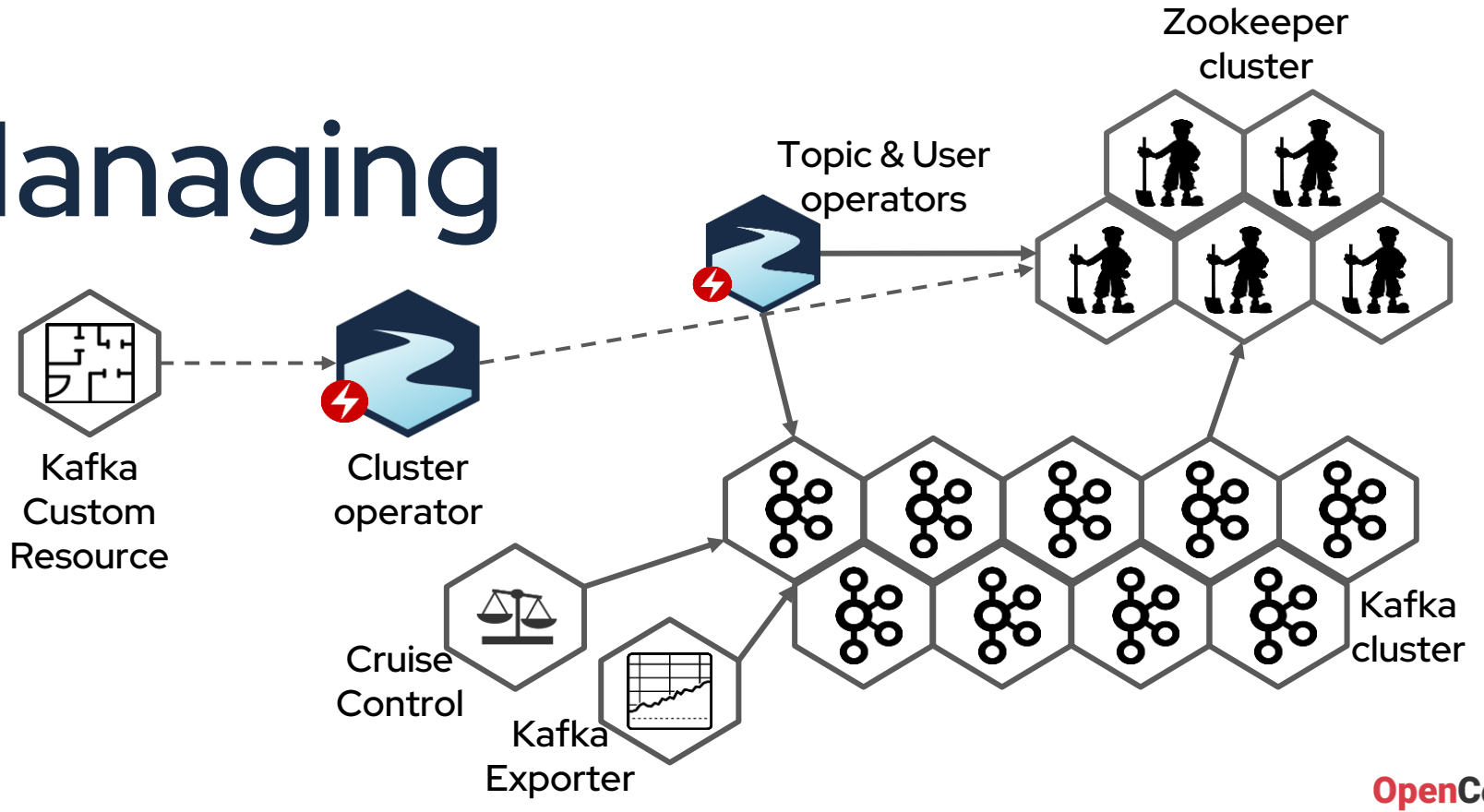


Kafka Custom Resource

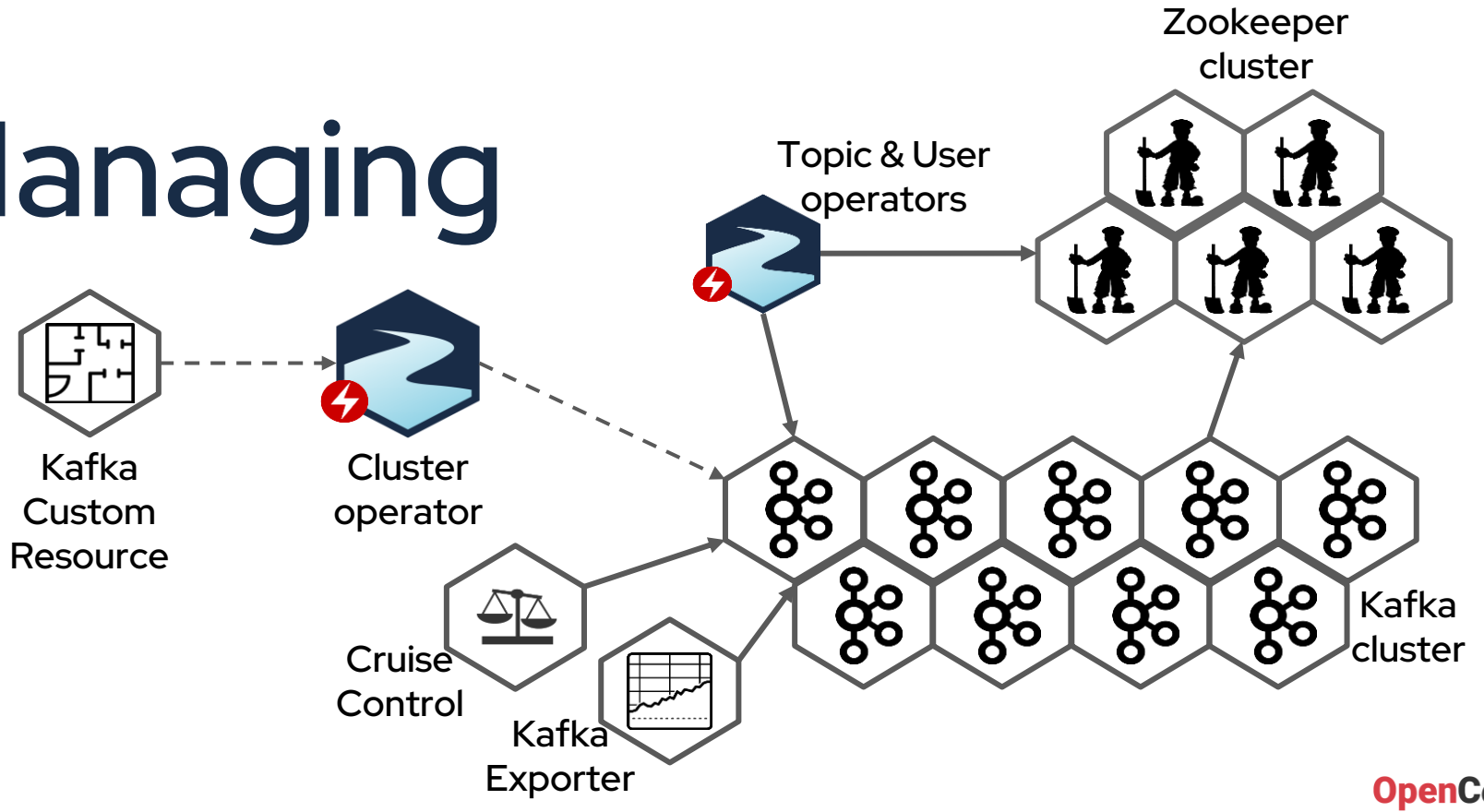
Cluster operator



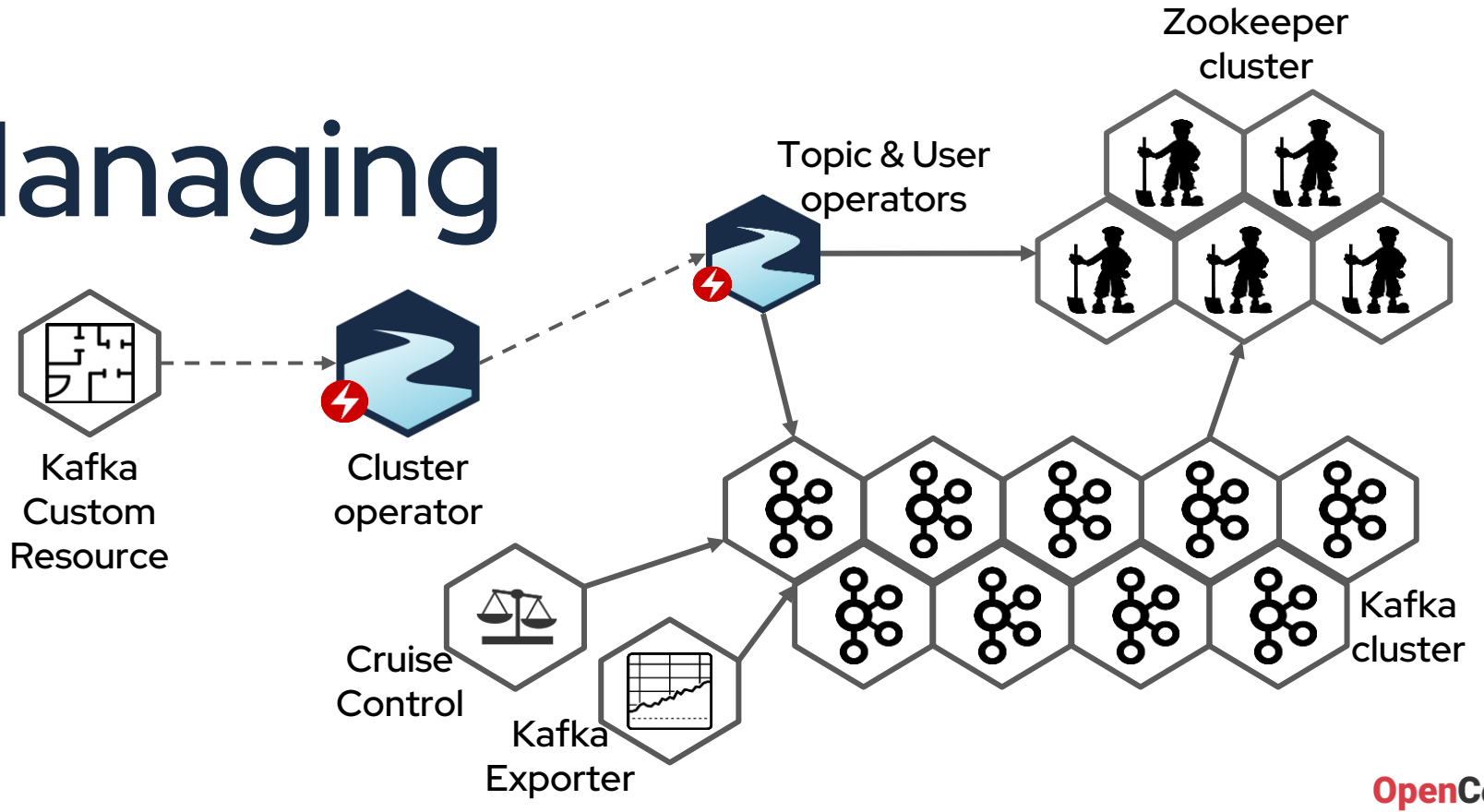
Managing


Zookeeper cluster
Topic & User operators
Kafka Custom Resource
Cluster operator
Cruise Control
Kafka Exporter
Kafka cluster

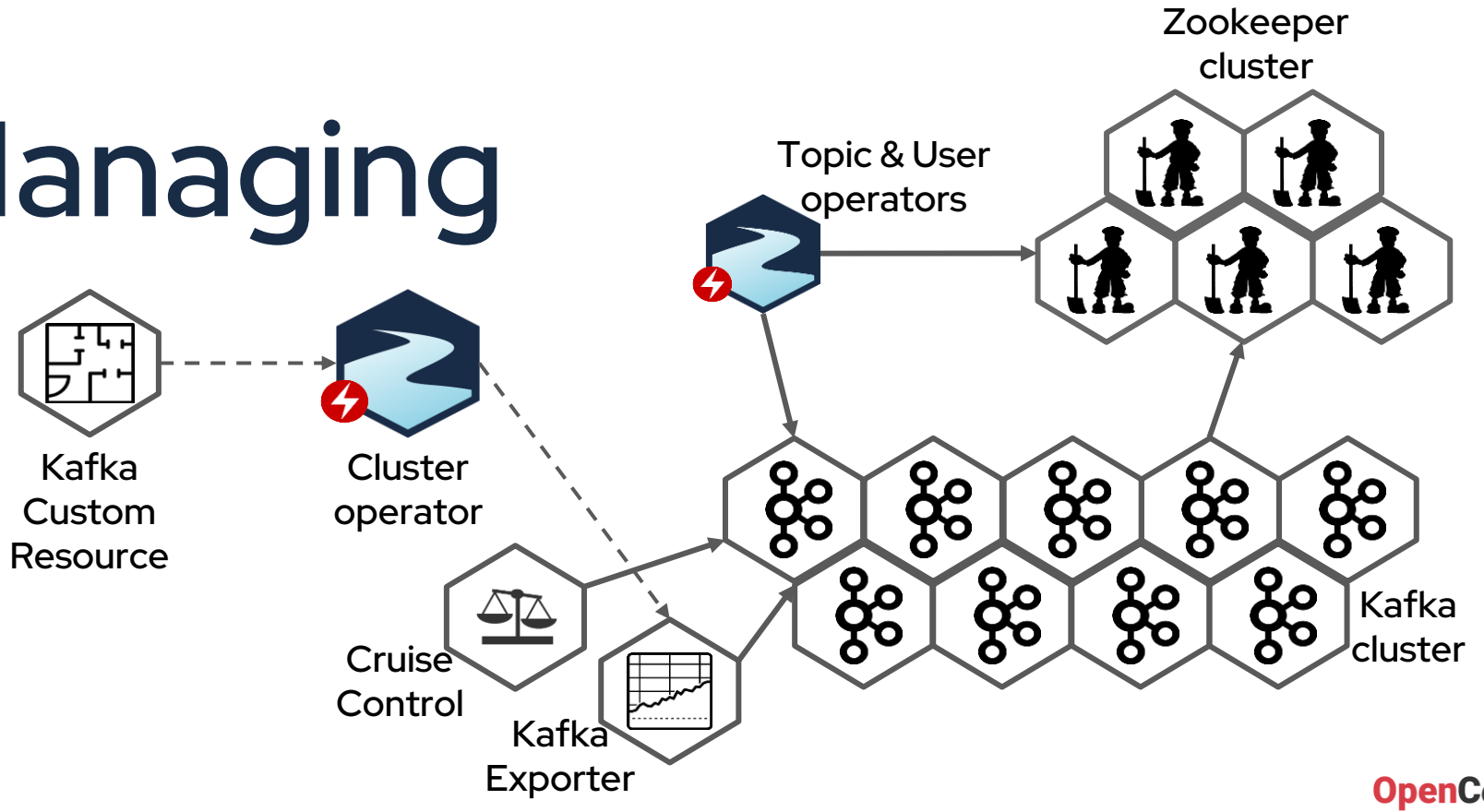
Managing


Zookeeper cluster
Topic & User operators
Kafka Custom Resource
Cluster operator
Cruise Control
Kafka Exporter
Kafka cluster

Managing


Zookeeper cluster
Topic & User operators
Kafka Custom Resource
Cluster operator
Cruise Control
Kafka Exporter
Kafka cluster

Managing


Zookeeper cluster
Topic & User operators
Kafka Custom Resource
Cluster operator
Cruise Control
Kafka Exporter
Kafka cluster

Operators deploy Operators

- In Strimzi, we want more than just manage Kafka cluster!
  - Why should the users know the Kafka specific tools for managing topics or users?
  - Let's give users ability to deploy Kafka clients "the Kubernetes native way"
  - Use Custom Resources and YAML to manage Kafka topics and users
  - Topic and User Operators are deployed per Kafka cluster
  - Topics and Users can be stored in YAML together with the applications using them

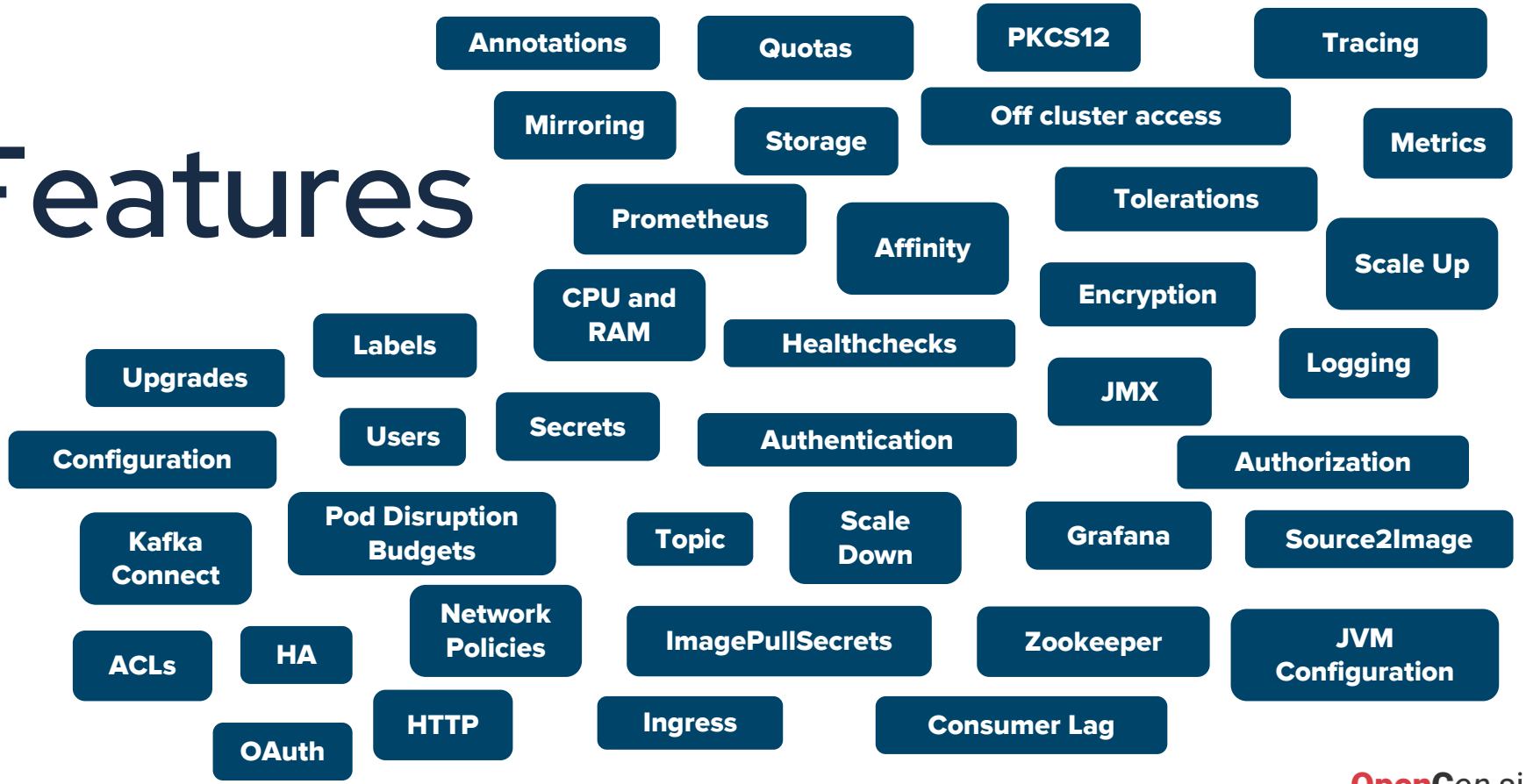
Topic Operator

- Lots of applications create topics directly in Kafka
  - Kafka Connect, Streams API, etc.
- Custom Resources cannot serve as a single source of truth
- Bi-directional synchronization between Kafka topics and Custom Resources
- Naming schema for Kafka topics is different from naming schema for Kubernetes resources
  - Some names need to contain hashes for uniqueness
- Can be used on its own

User Operator

- Manages:
  - Users
  - ACLs
  - User quotas
- Creates Kubernetes secret with credentials
- Can be used on its own

Features


Annotations
Quotas
PKCS12
Tracing
Mirroring
Storage
Off cluster access
Metrics
Prometheus
Affinity
Tolerations
Scale Up
CPU and RAM
Encryption
Labels
Healthchecks
Logging
Upgrades
JMX
Users
Secrets
Authentication
Configuration
Authorization
Kafka Connect
Pod Disruption Budgets
Topic
Scale Down
Grafana
Source2Image
Network Policies
ACLs
HA
ImagePullSecrets
Zookeeper
JVM Configuration
HTTP
Ingress
Consumer Lag
OAuth

Thank you

Web site: http://strimzi.io/

GitHub: https://github.com/strimzi

YouTube: https://youtube.com/@Strimzi