

**RED HAT
SUMMIT**

Data streaming with Apache Kafka using AMQ streams

Tom Bentley
Principal Software Engineer

Jakub Scholz
Principal Software Engineer

May 9th, 2019

Introductions

Red Hat AMQ

AMQ Online

- Scalable, easy-to-manage messaging based on OpenShift container platform
- Developer self-service model; Metering, etc.

AMQ Broker

- Store and forward
- Volatile and durable
- JMS 2.0
- Standardized AMQP 1.0 and MQTT

AMQ Interconnect

- High performance direct messaging
- Distributed messaging backbone

AMQ Streams

- Streaming platform
- Durable pub/sub
- Replayable streams
- Based on Apache Kafka

Common Management



Apache Kafka

- Streaming data platform
- Pub/sub messaging
- Main features
 - Horizontally scalable
 - Fault tolerant
 - Immutable commit log
- With an ecosystem of software around it, including:
 - numerous language bindings for producers and consumers
 - Connectors for getting information to/from other systems
 - An API (Kafka Streams) for writing real-time event-based applications



AMQ Streams

- Apache Kafka packaged and supported by Red Hat
 - Broker, Java clients, Kafka Connect, Kafka Streams
- Available to run on two platforms:
 - On RHEL, for bare metal or virtualized deployment
 - On OCP, for on-premise or public cloud deployment
- AMQ Streams on OCP is based on Strimzi project
- All components are open source



STRIMZI

AMQ Streams on OpenShift

An Operator for Kafka Clusters

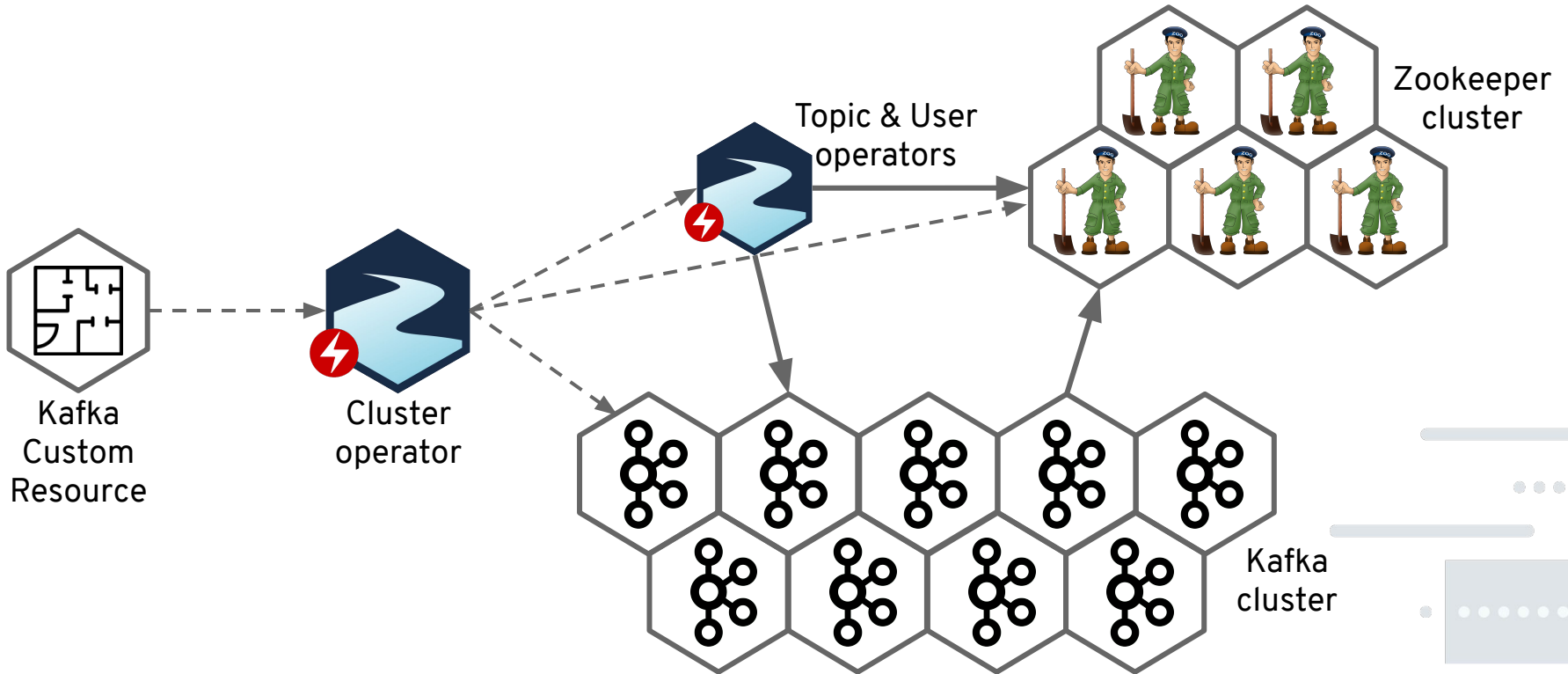
The **Kafka** custom resource describes the desired Kafka (and Zookeeper) cluster(s)

Benefits of operator approach for Kafka:

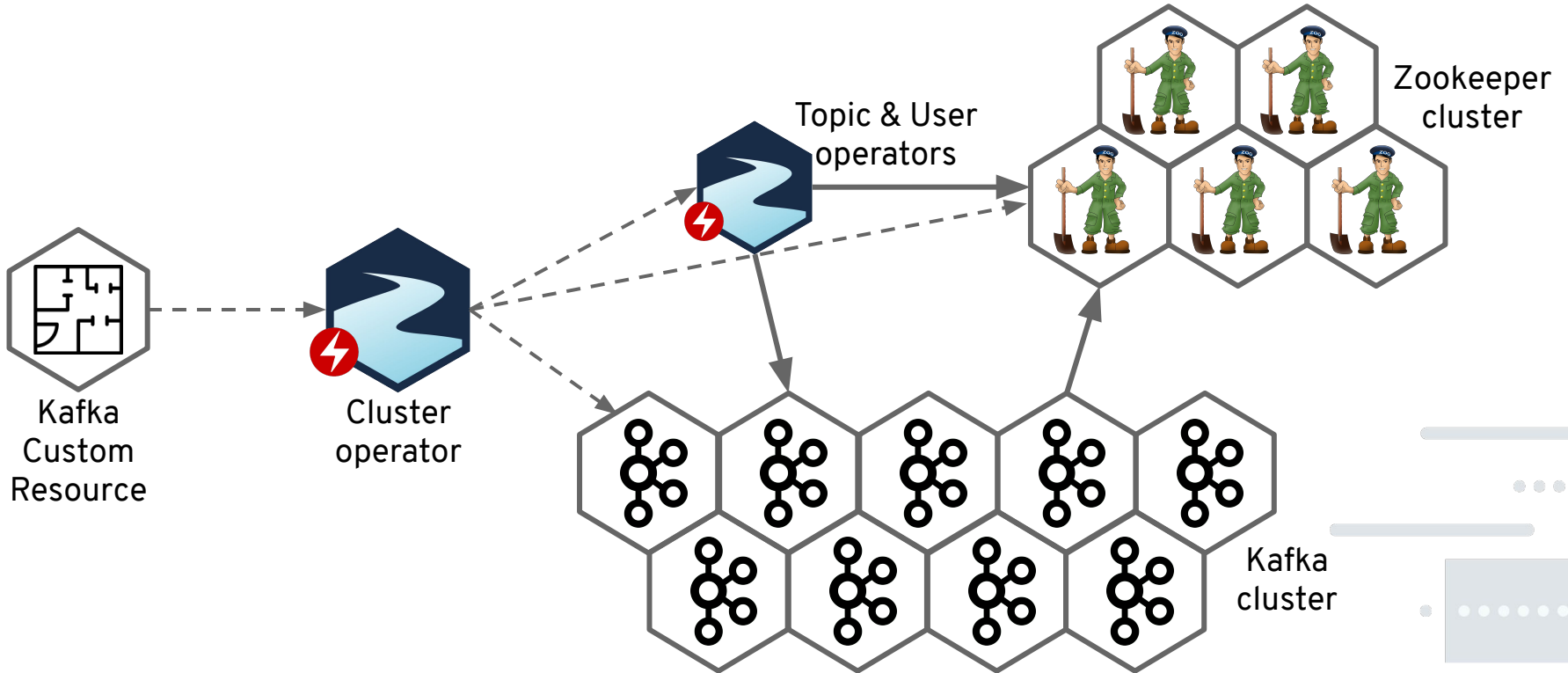
- Elastic Kafka clusters (leverage OpenShift's elasticity)
- Lowers barriers for using complex technologies such as Kafka
- No need for complicated, error-prone manual configuration of TLS, authentication, authorization, etc.
- Benefits of declarative configuration for full-lifecycle
- OpenShift-centric: DevOps can deploy whole application as native OpenShift resources

```
apiVersion: kafka.strimzi.io/v1beta1
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    version: 2.2.0
    replicas: 3
    listeners:
      plain: {}
      tls: {}
    storage:
      type: jbod
      volumes:
        - id: 0
          type: persistent-claim
          size: 100Gi
          deleteClaim: false
        - id: 1
          type: persistent-claim
          size: 100Gi
          deleteClaim: false
    config:
      offsets.topic.replication.factor: 3
      transaction.state.log.replication_factor: 3
```

Example: Creating a cluster



Example: Updating a cluster



Demonstration: Spinning up a Kafka Cluster

Enterprise Kafka applications in OpenShift

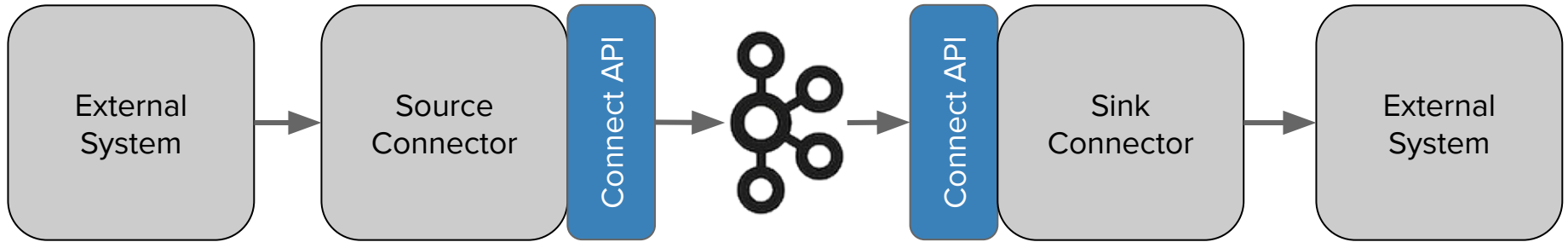
- Having a Kafka cluster is great but... pointless without applications
- DevOps need to create topics, authenticate and authorize access to Kafka resources, etc.
- DevOps should be OpenShift-native
- AMQ Streams uses custom resources & operators for Topics, Authnz, Kafka Connect, Mirror Maker
- These can then be deployed at the same time as the rest of the application

```
apiVersion: kafka.strimzi.io/v1beta1
kind: KafkaUser
metadata:
  name: my-user
  labels:
    strimzi.io/cluster: my-cluster
spec:
  authentication:
    type: tls
  authorization:
    type: simple
  acls:
    - resource:
        type: topic
        name: my-topic
        patternType: literal
      operation: Read
      host: "*"

```

Demonstration: Creating topics and users

Kafka Connect

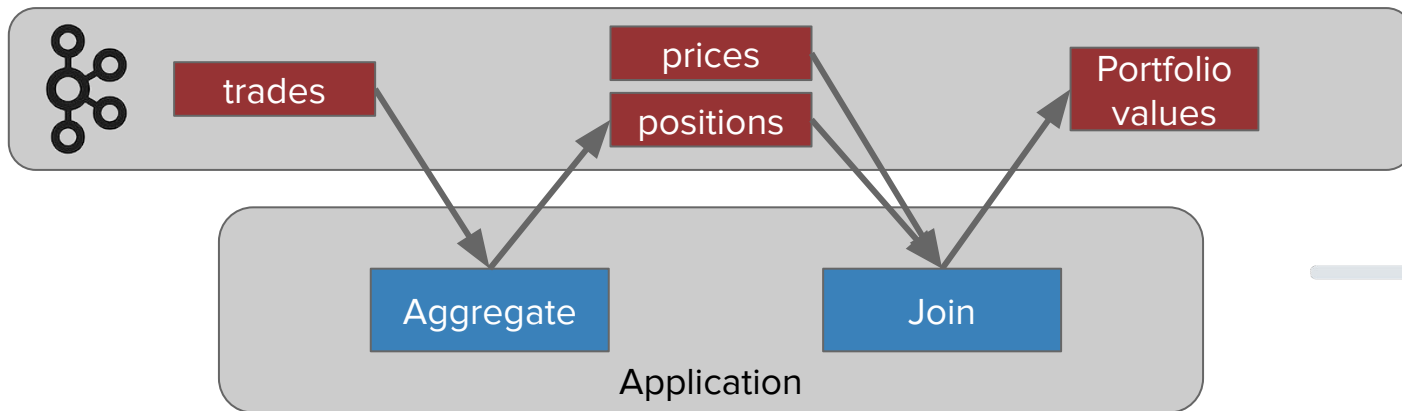


- Kafka Connect is a framework for connecting Kafka to external systems
- Connectors run inside Kafka Connect (i.e. are plugins)
- Common Kafka producer/consumer machinery
- Easy to write
- Connector developers focus on getting data into, or out of, their particular system
- Large ecosystem of connectors
- Example: Debezium

Demonstration: An example connector Fetching stock prices

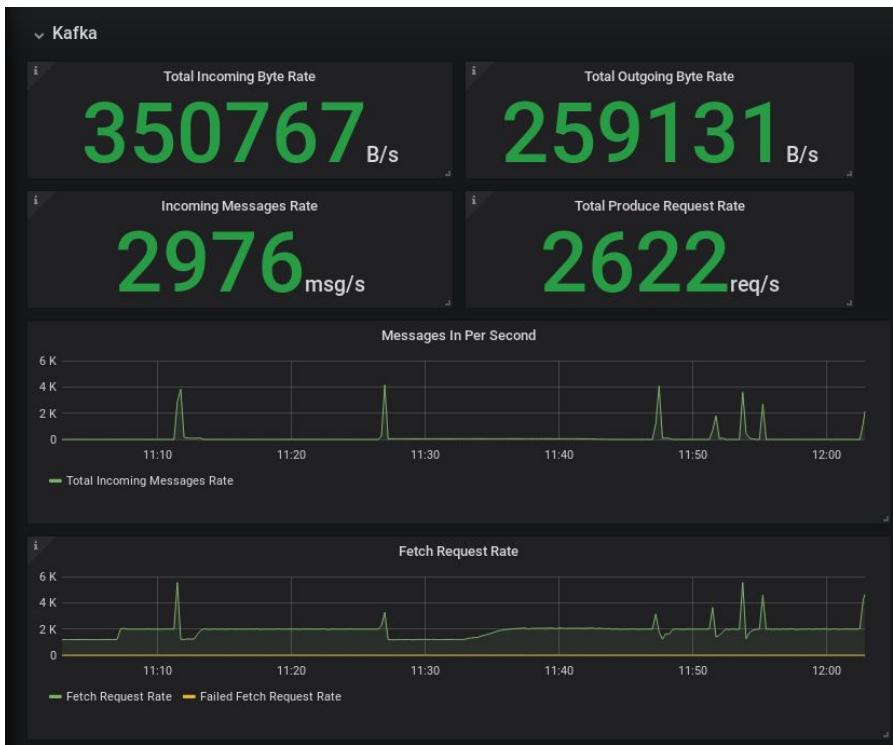
Kafka Streams

- Message often encapsulating events
- The events have business value
- Real-time, stream
- “Streaming Applications”
- In terms of functional operations (map, filter, join, etc.)
- E.g. Sum a stream of stock trades to a stream of aggregated positions
- E.g. Join positions with a stream of stock prices to get portfolio value



Demonstration:
Example Kafka Streams application
Real time Stock portfolio valuation

Monitoring Kafka



- AMQ Streams on OCP integrates with Prometheus for monitoring
- Alerting is also supported via Prometheus
- Grafana dashboards are provided OOTB
- Separate Prometheus and Grafana instances are used (not the ones used for OCP itself)

Demonstration: Monitoring

What we've seen

- Apache Kafka is great
- AMQ Streams offers a Red Hat supported distribution of Apache Kafka:
 - On RHEL
 - On OCP
- Operators are a great way to give users an OpenShift-native experience
- The AMQ Streams operators make it super-easy to deploy Kafka on OCP
- Operators for topics, users etc extend the OpenShift-native experience even further
- Example Connector and Kafka Streams application
- Monitoring

Features you've not seen

Prometheus

Encryption

Storage

TLS by default

Mirroring

Labels

Affinity

Tolerations

Metrics

Annotations

Users

Scale Up

CPU and RAM

Off-cluster access

Upgrades

Secrets

SASL SCRAM
SHA

Healthchecks

Logging

Configuration

Authentication

Authorization

Kafka
Connect

Pod Disruption
Budgets

Scale
Down

Topic

Grafana

Source2Image

ACLs

HA

Network
Policies

ImagePullSecrets

Zookeeper

JVM
Configuration

Roadmap

Roadmap

Currently planned for AMQ Stream 1.2:

- Support for Kafka 2.2.x
- HTTP Proxy
- Improvements in Storage reconfiguration:
 - Adding disks
 - Changing disk sizes

In future versions of AMQ Streams:

- External authentication (RH SSO)
- Schema Registry
- Cluster balancing
- Kafka Connect connectors (Debezium, AMQP)
- Console (GUI)
- SQL Stream processing
- Kafka \rightleftharpoons AMQP

**RED HAT
SUMMIT**

THANK YOU



plus.google.com/+RedHat



linkedin.com/company/red-hat



youtube.com/user/RedHatVideos



facebook.com/redhatinc



twitter.com/redhat