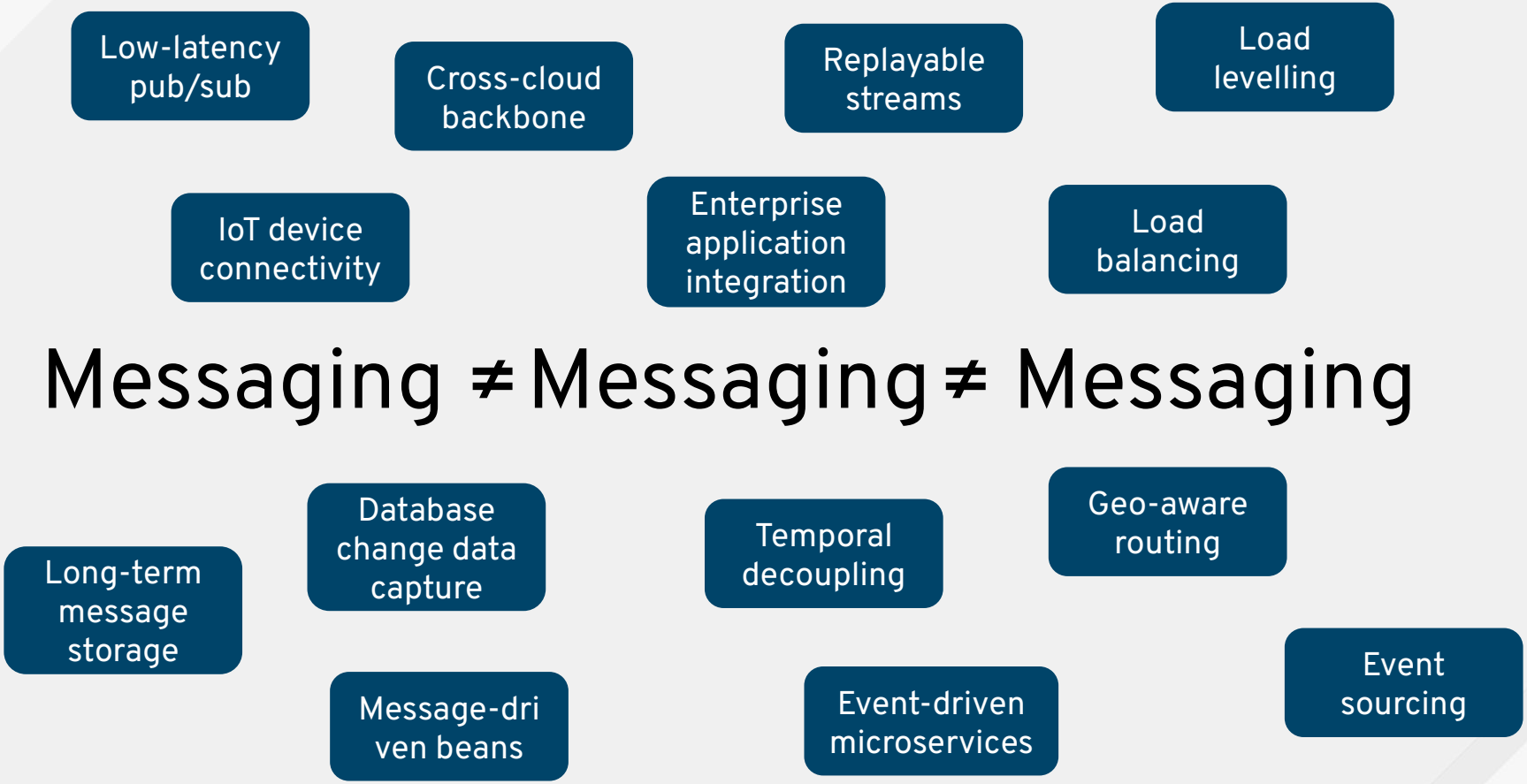
Kafka on OpenShift : make it easy with AMQ Streams

Event Streaming and reactive architectures

Paolo Patierno
Principal Software Engineer @ Red Hat
@ppatierno

#RedHatOSD


Low-latency pub/sub
Cross-cloud backbone
Replayable streams
Load levelling
IoT device connectivity
Enterprise application integration
Load balancing
Messaging ≠ Messaging ≠ Messaging
Long-term message storage
Database change data capture
Temporal decoupling
Geo-aware routing
Message-dri ven beans
Event-driven microservices
Event sourcing

AMQ Streams on OpenShift Container Platform

- Enterprise distribution of Apache Kafka
- Simplified deployment on OpenShift
- Based on OSS project called Strimzi
- Provides:
  - Container images for running Apache Kafka and Zookeeper
  - Operators for managing and configuring Apache Kafka clusters, topics and users

What is Apache Kafka?

A publish/subscribe messaging system?

A streaming data platform?

A distributed, horizontally-scalable, fault-tolerant, commit log?

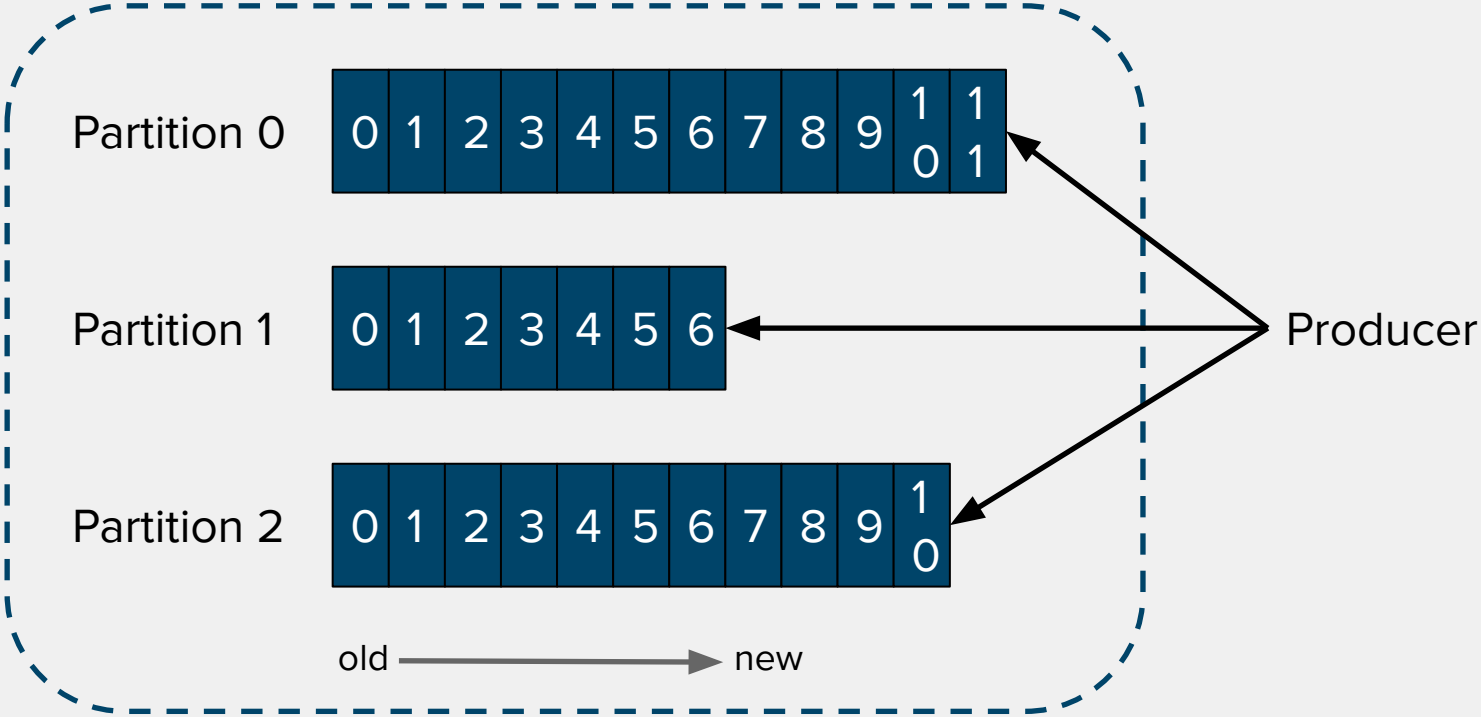
Apache Kafka

Concepts

- Messages are sent to and received from a topic
  - Topics are split into one or more partitions (aka shards)
  - All actual work is done on partition level, topic is just a virtual object
- Each message is written only into a one selected partition
  - Partitioning is usually done based on the message key
  - Message ordering within the partition is fixed
- Retention
  - Based on size / message age
  - Compacted based on message key

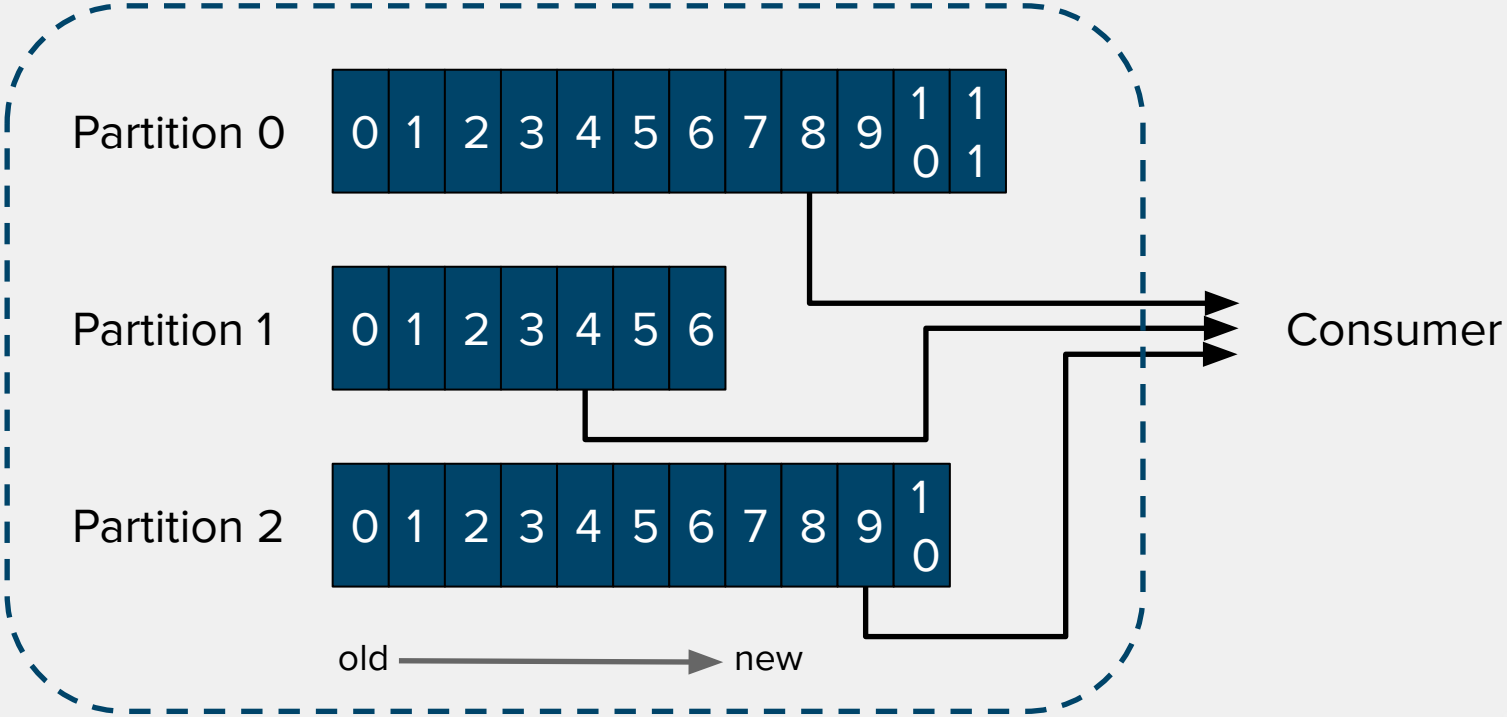
Apache Kafka concepts

Topics & partitions



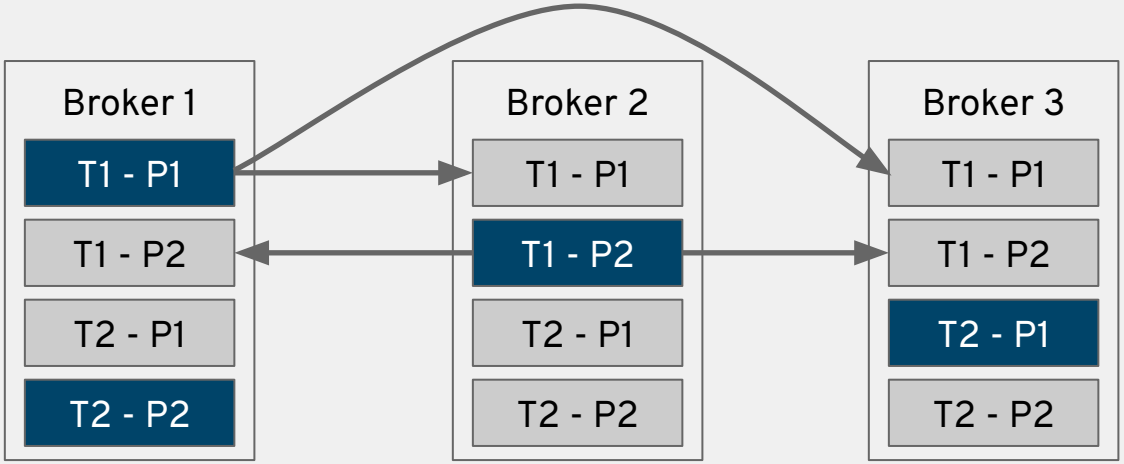
Apache Kafka concepts

Topics & partitions



Apache Kafka concepts

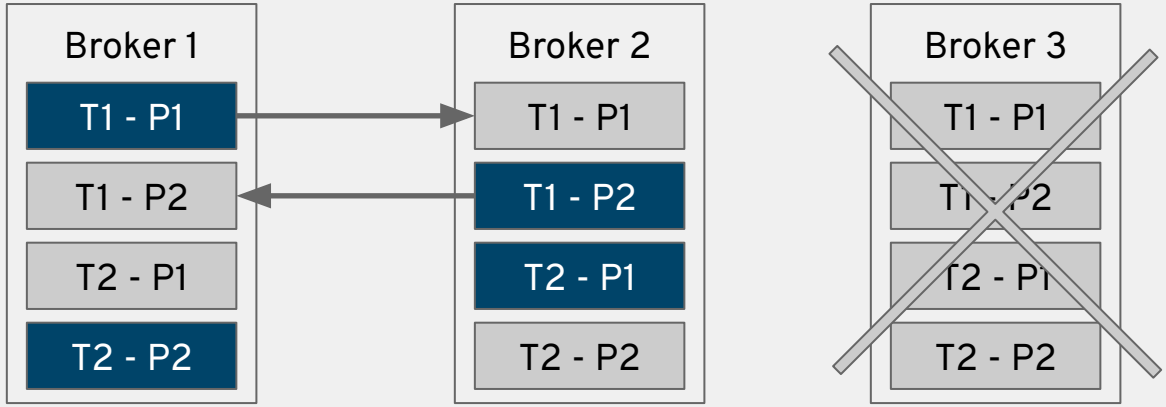
High availability



Leaders and followers spread across the cluster

Apache Kafka concepts

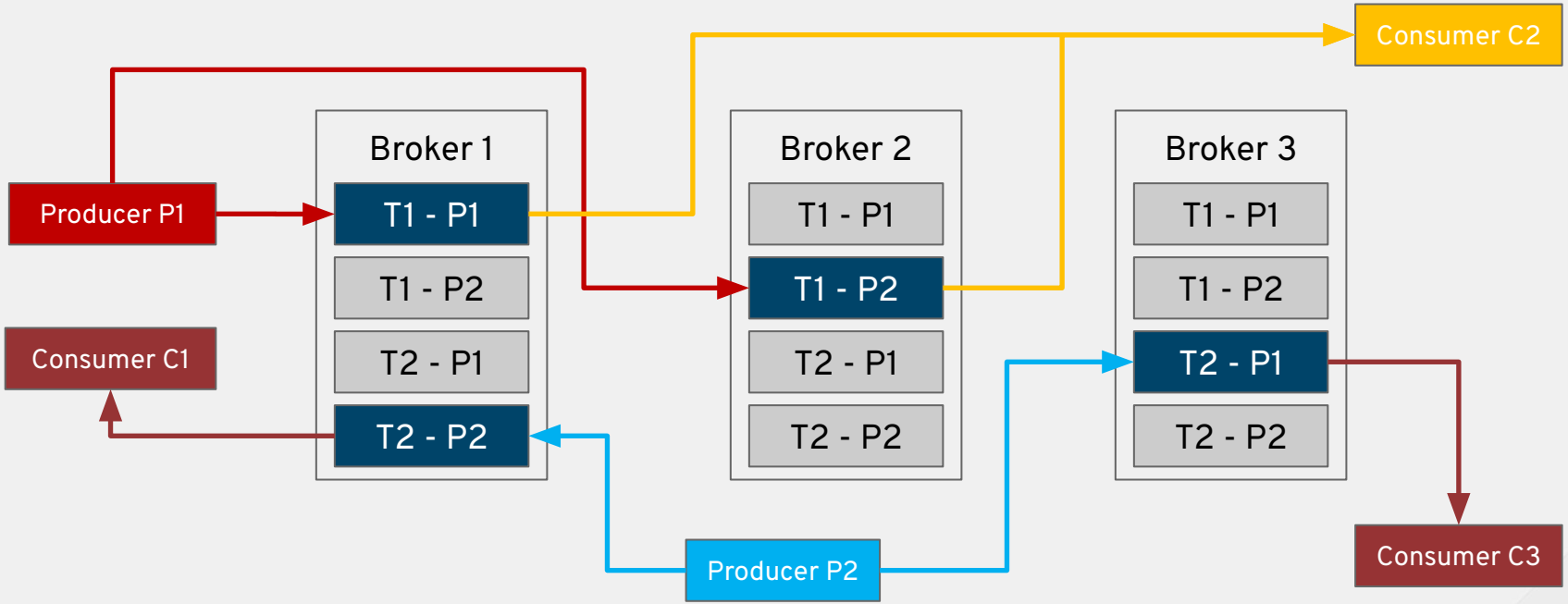
High availability



If a broker with leader partition goes down, a new leader partition is elected on different node

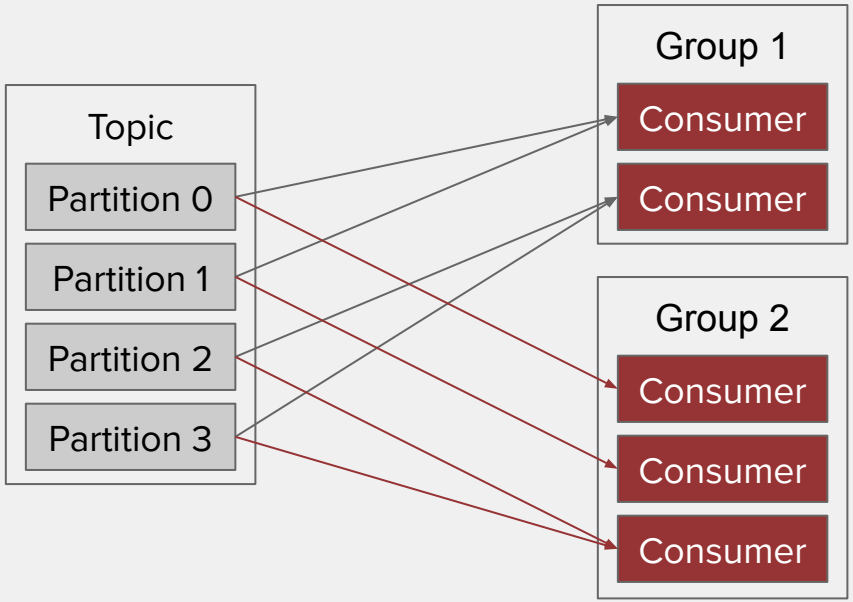
Apache Kafka concepts

Reading and writing to leaders



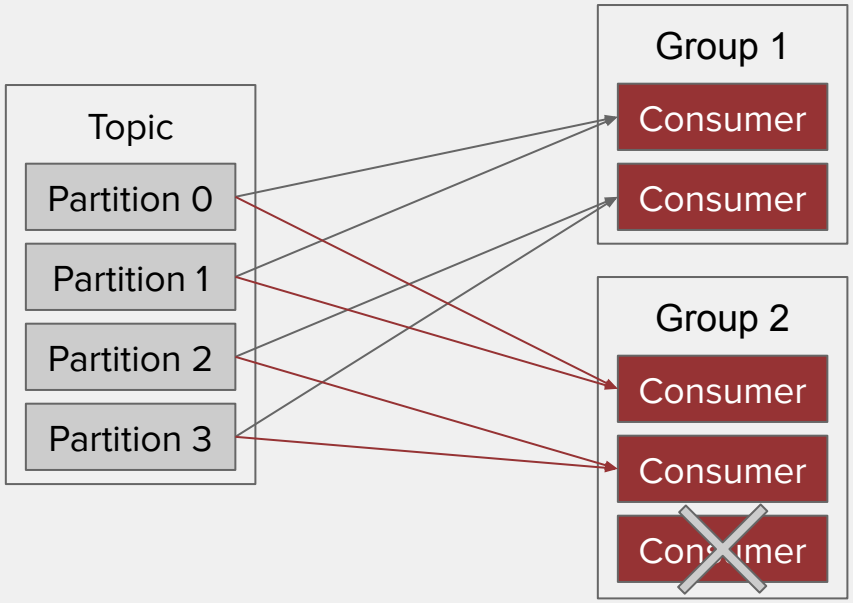
Apache Kafka concepts

Consumer Groups: partitions assignment



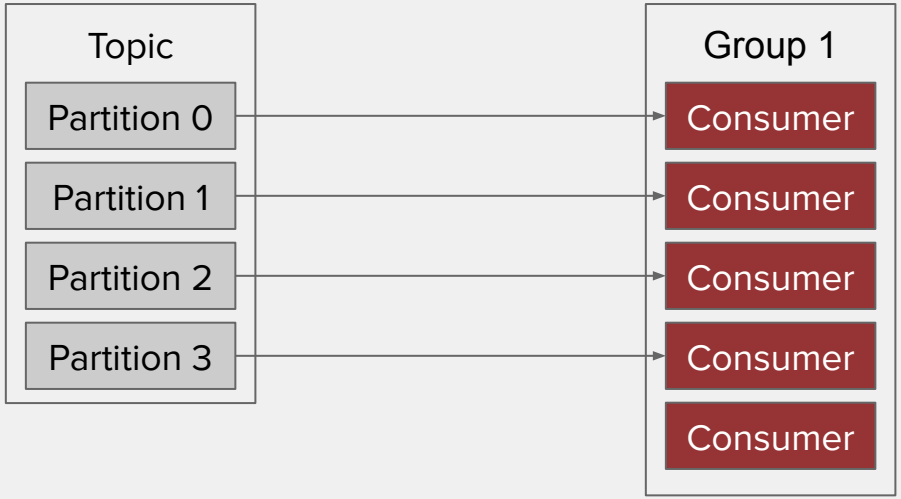
Apache Kafka concepts

Consumer Groups: rebalancing



Apache Kafka concepts

Consumer Groups: max parallelism & idle consumer



AMQ Broker & AMQ Streams

Key differences

	AMQ Broker (ActiveMQ Artemis)	AMQ Streams (Kafka)
Model	“Smart broker, dumb clients”	“Dumb broker, smart clients”
Durability	Volatile or durable storage	Durable storage
Storage duration	Temporary storage of messages	Potential long-term storage of messages
Message retention	Retained until consumed	Retained until expired or compacted
Consumer state	Broker managed	Client managed (can be stored in broker)
Selectors	Yes, per consumer	No
Stream replay	No	Yes
High-availability	Replication	Replication
Protocols	AMQP, MQTT, OpenWire, Core, STOMP	Kafka protocol
Delivery guarantees	Best-effort or guaranteed	Best-effort or guaranteed

AMQ Streams on OCP

The challenges

- Apache Kafka is *stateful* which means we require …
  - … a stable broker identity
  - … a way for the brokers to discover each other on the network
  - … durable broker state (i.e., the messages)
  - … the ability to recover broker state after a failure
- All the above are true for Apache Zookeeper as well
- StatefulSets, PersistentVolumeClaims, Services can help but …

It’s not easy!

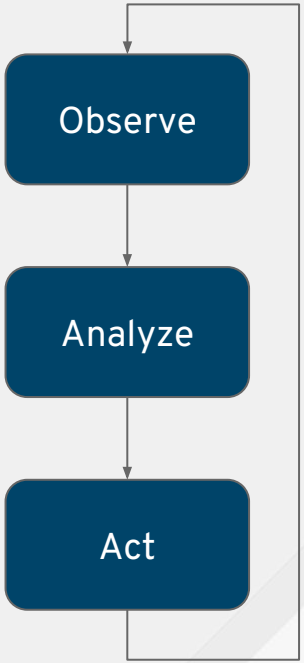
Goals

- Simplifying the Apache Kafka deployment on OpenShift
- Using the OpenShift native mechanisms for...
  - Provisioning the cluster
  - Managing the topics and users
- ... thereby removing the need to use Kafka command-line tools
- Providing a better integration with applications running on OpenShift
  - microservices, data streaming, event-sourcing, etc.

AMQ Streams on OCP

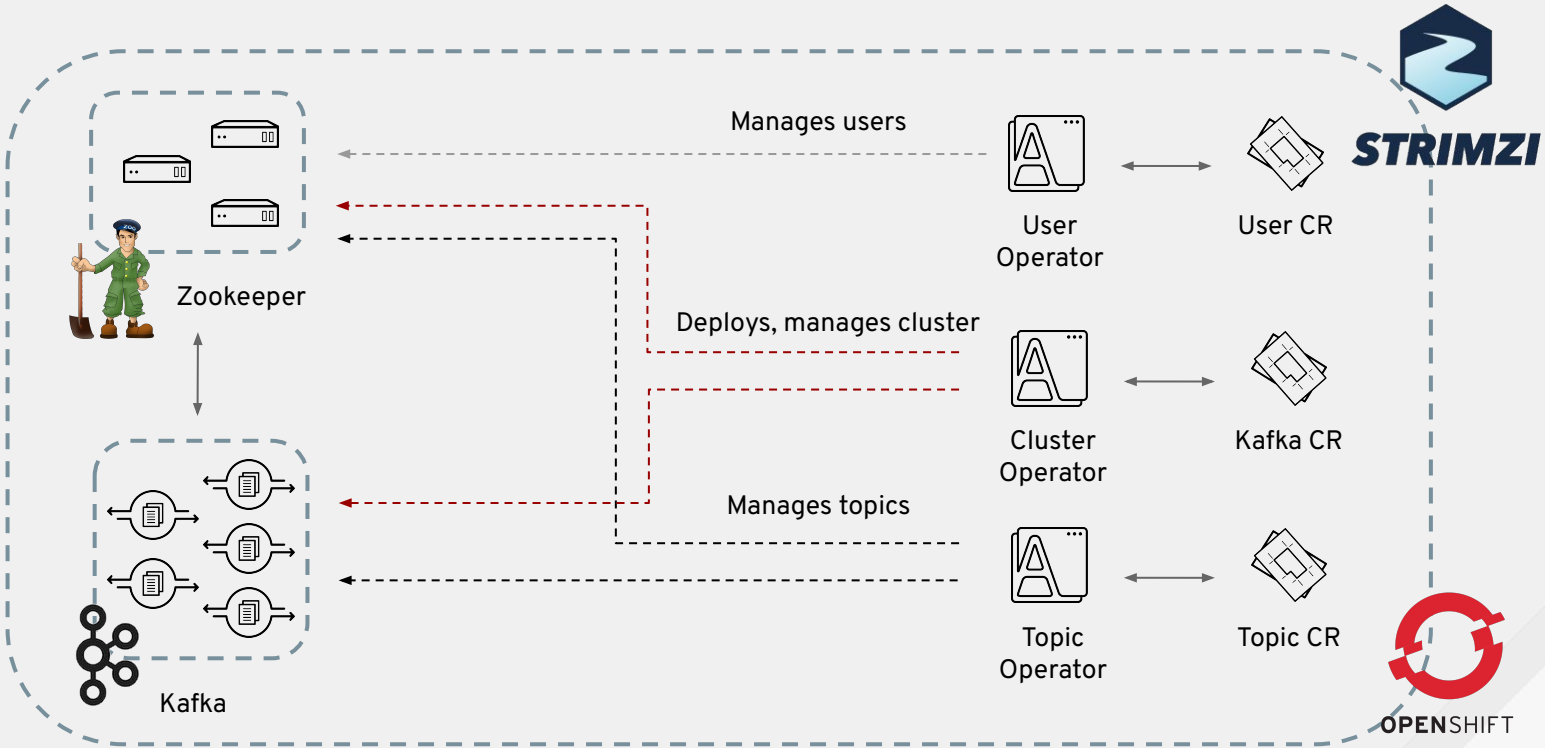
The “Operator” model

- An application used to create, configure and manage other complex applications
  - Contains specific domain / application knowledge
- Operator works based on input from Config Maps or Custom Resource Definitions
  - User describes the desired state
  - Controller applies this state to the application
- It watches the *desired* state and the *actual* state …
  - … taking appropriate actions



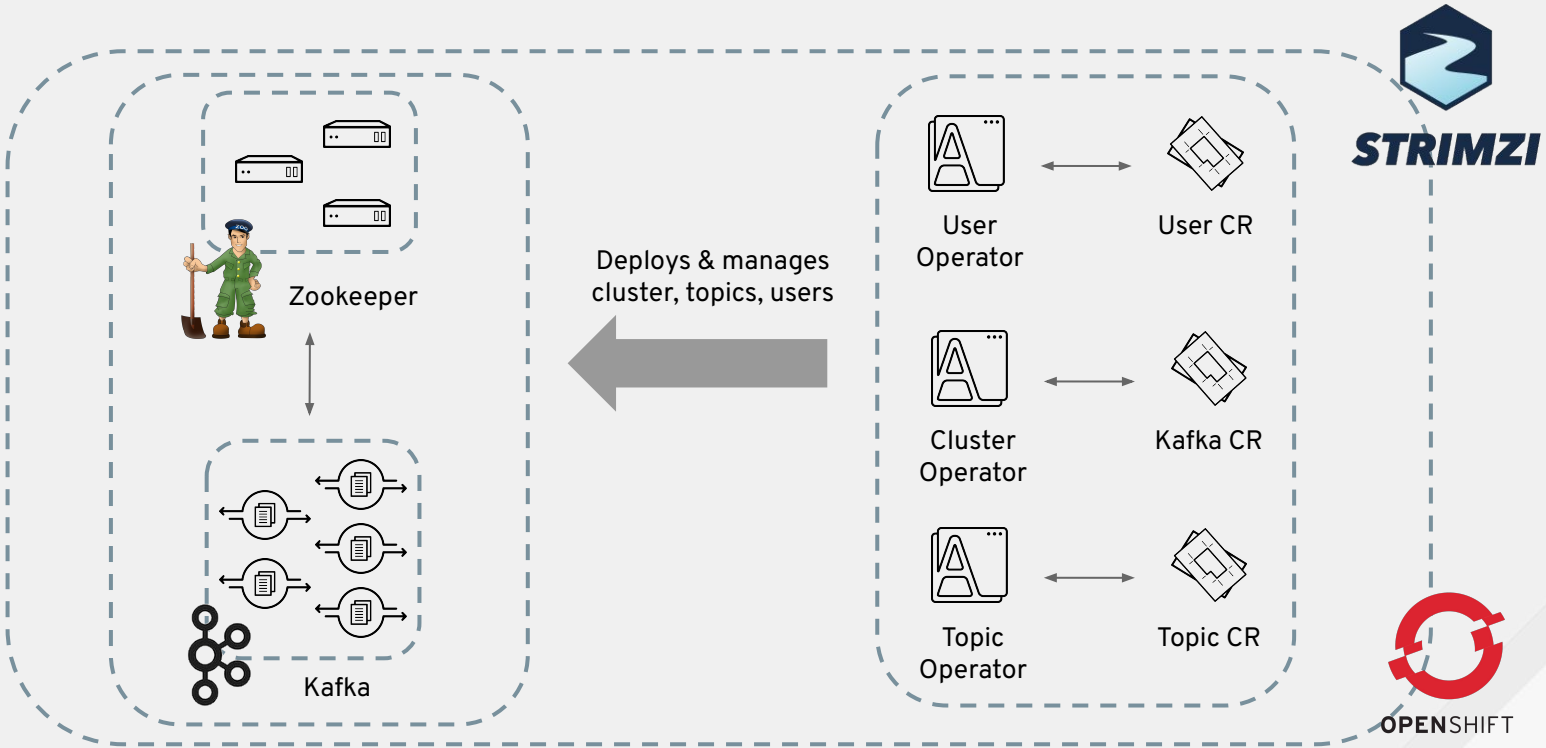
AMQ Streams on OCP

The Operators



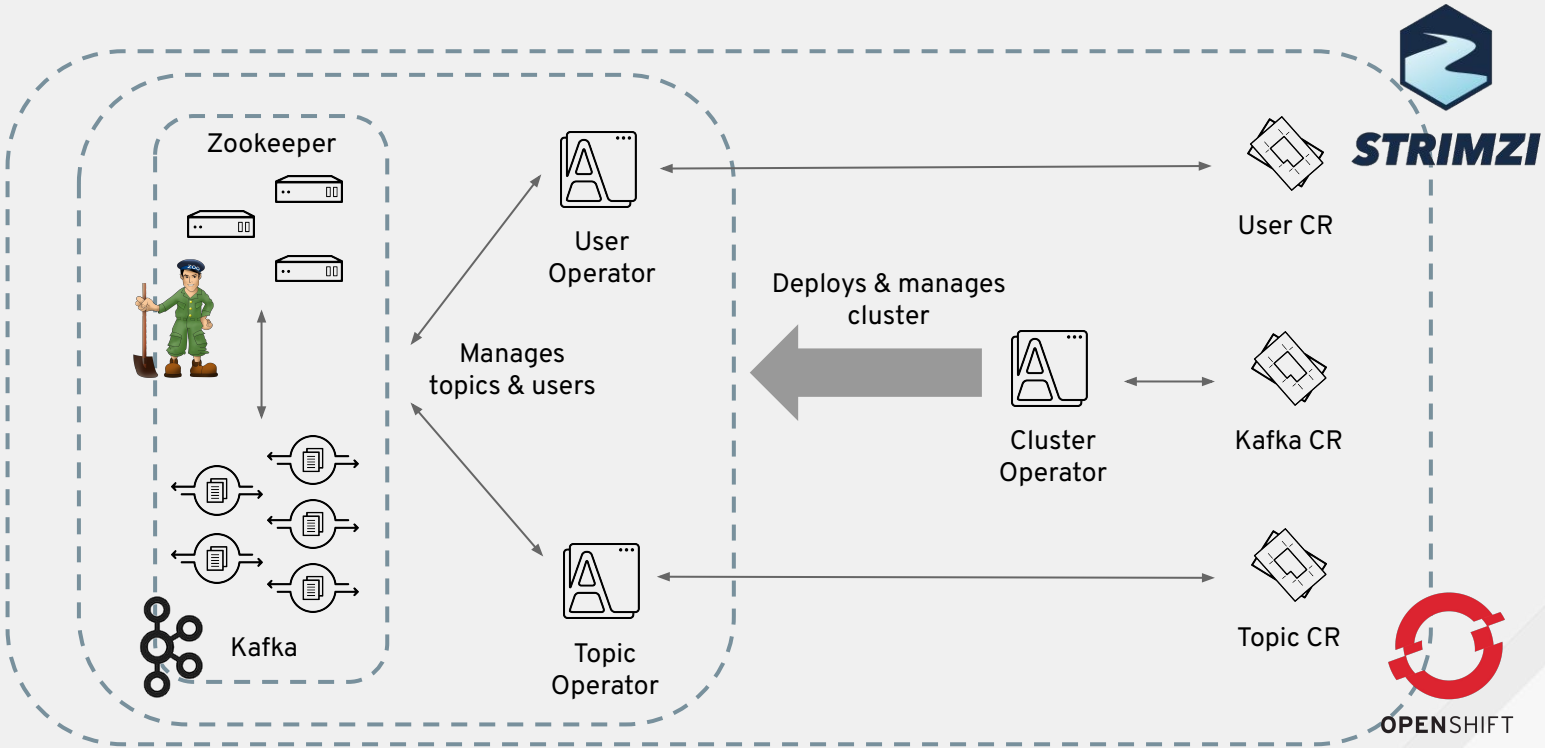
AMQ Streams on OCP

The Operators



AMQ Streams on OCP

The Operators



AMQ Streams on OCP

Cluster Operator

- Responsible for managing clusters
  - Kafka brokers (including Zookeeper)
  - Kafka Connect clusters
  - Kafka Mirror Maker
- Responsible for
  - Deployment
  - Scale-up / Scale-down
  - Re-configuration

AMQ Streams on OCP

Topic Operator

- Responsible for managing Kafka topics
  - You can create, update and delete topics “the Kubernetes way”
  - No need to know Kafka commands
  - Applications can still create topics directly in Kafka
    - Topic operator synchronizes the topics bi-directionally
    - For topics created in Kafka, it will create Custom Resources
    - In case of conflicts, it will use 3-way-diff to resolve them

AMQ Streams on OCP

User Operator

- Responsible for managing users
  - Allows to create, update and delete users
  - Currently two supported authentication mechanisms
    - TLS client certificates
    - SASL SCRAM-SHA-512 (username and password based authentication)
  - Authorization manages using Kafka ACL plugin
    - Allowed / Denied operations can be defined together with the user

AMQ Streams on OCP

Main features

Mirroring

Tolerations

High Availability

Memory and CPU resources

Storage

Encryption

Authentication

Affinity

Metrics

Access from outside

Logging

JVM Configuration

Scale Down

Scale Up

Healthchecks

Authorization

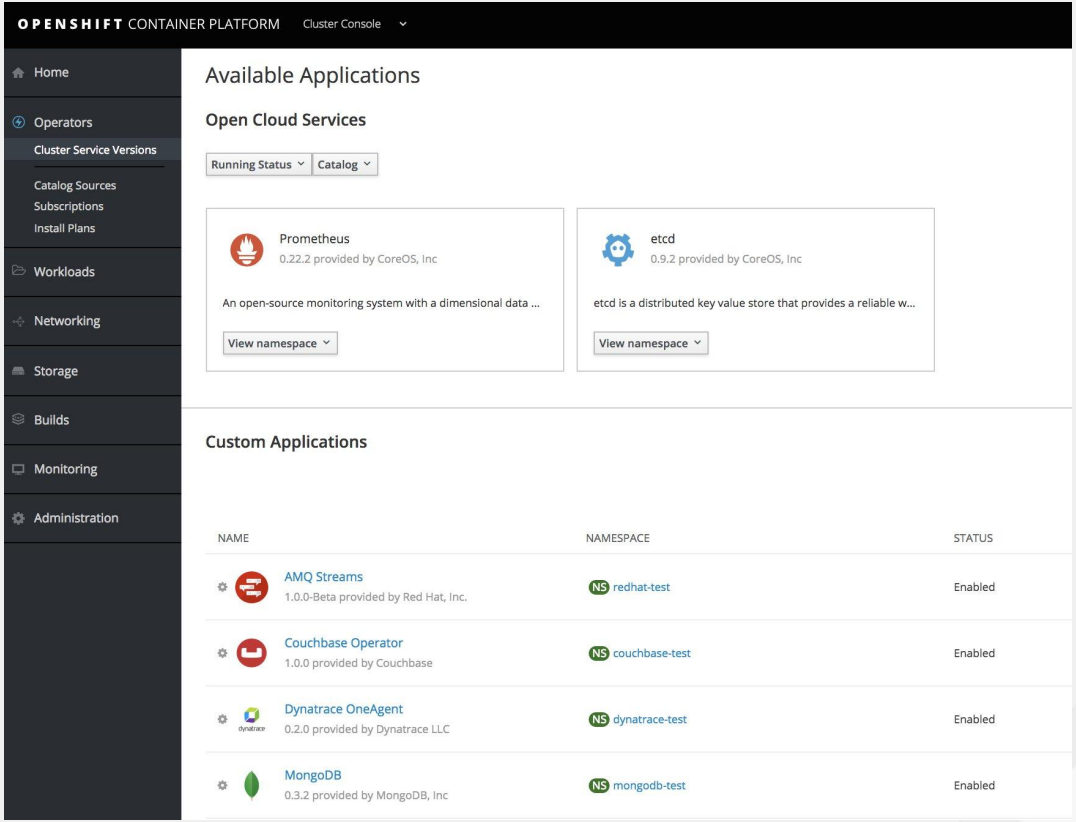
Source2Image

Configuration

AMQ Streams on OCP

Operator

- OCP 3.11 provides a few operators
  - Prometheus, etcd, ...
- AMQ Streams 1.0 available

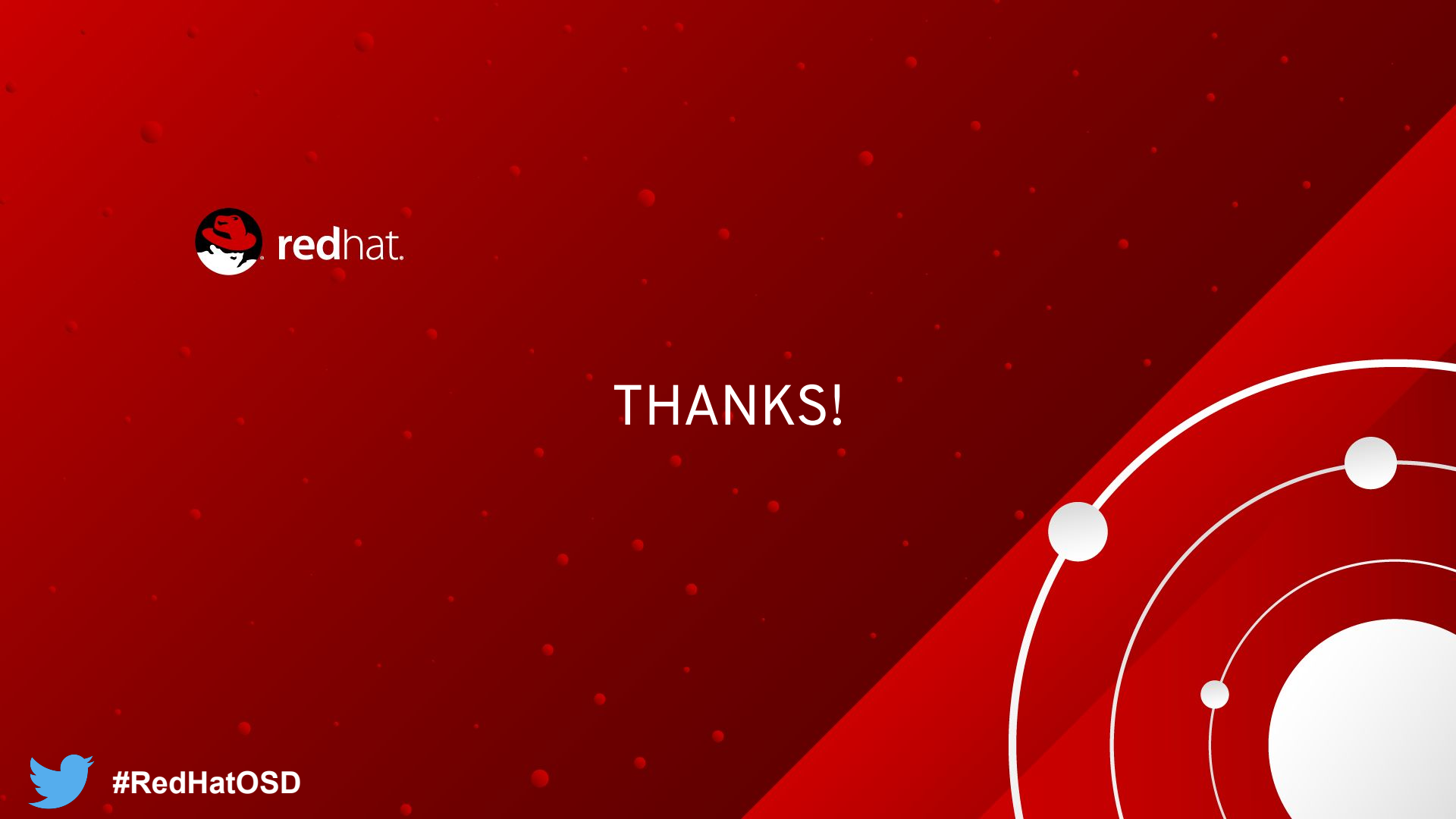


AMQ Streams on OpenShift Container Platform is GA!

DEMO TIME

Resources

- AMQ Streams : https://access.redhat.com/products/red-hat-amq-streams
- Strimzi : http://strimzi.io/ - @strimziio
- Apache Kafka : https://kafka.apache.org/
- Demo : https://github.com/ppatierno/rh-osd-2018


redhat.
THANKS!
#RedHatOSD