BanzaiCloud

Kafka Operator

and

cluster Autoscaling

kafka operator

BY BANZAICLOUD

Who Am I ?

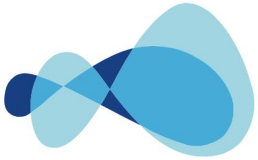
Sebastien Thomas / Prune

“Customer Reliability Engineer”

https://github.com/prune998

https://www.linkedin.com/in/prune/

Tetrate.io



Enterprise ready service mesh

for any workload on any environment

Powered by **Envoy and Istio**

we’re Hiring ! https://www.tetrate.io/about-us/careers/

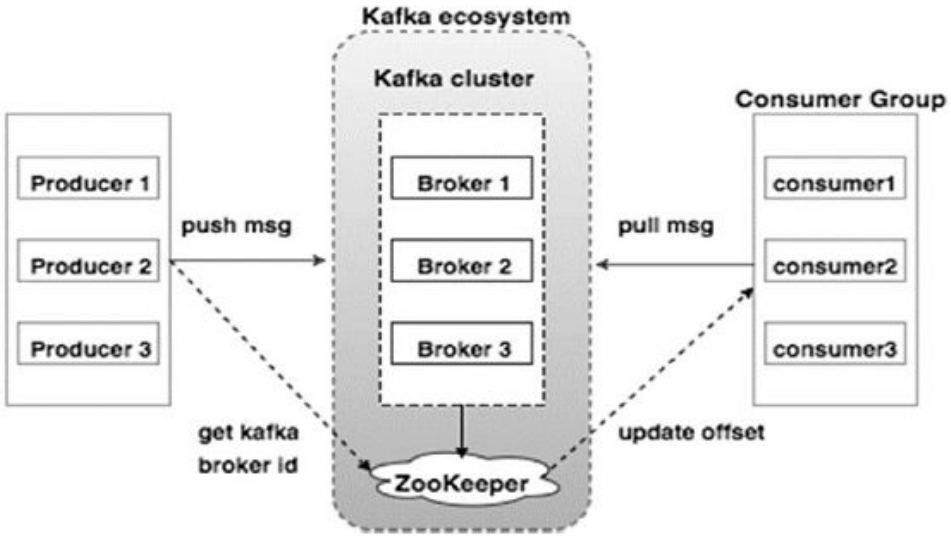
Schedule

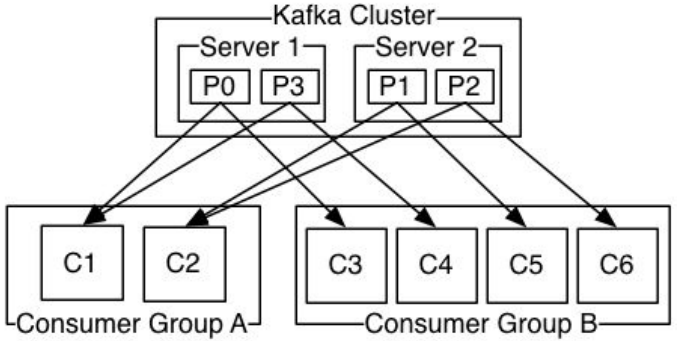
- Kafka recap
- Another Operator ?
- Overview of the Operator
- Install
- CRD
- Topics
- Alerting / Scaling
- Recap
- Takeout

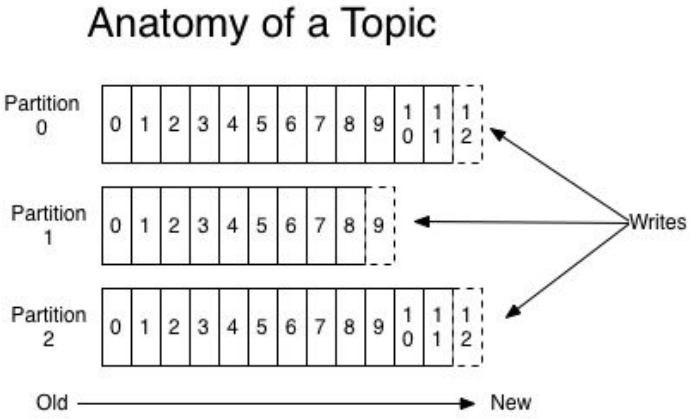
Schedule

- Kafka recap
- Another Operator ?
- Overview of the Operator
- Install
- CRD
- Topics
- Alerting / Scaling
- Recap
- Takeout

Kafka : producer/consumer service







Schedule

- Kafka recap
- Another Operator ?
- Overview of the Operator
- Install
- CRD
- Topics
- Alerting / Scaling
- Recap
- Takeout

	Banzai Cloud	Krallistic	Strimzi	Confluent
Open source	Apache 2	Apache 2	Apache 2	No
Fine grained broker config support	Yes	Limited via StatefulSet	Limited via StatefulSet	Limited via StatefulSet
Fine grained broker volume support	Yes	Limited via StatefulSet	Limited via StatefulSet	Limited via StatefulSet
Monitoring	Yes	Yes	Yes	Yes
Encryption using SSL	Yes	Yes	Yes	Yes
Rolling Update	Yes	No	No	Yes
Cluster external accesses	Envoy (single LB)	Nodeport	Nodeport or LB/broker	Yes (N/A)
User Management via CRD	Yes	No	Yes	No
Topic management via CRD	Yes	No	Yes	No
Reacting to Alerts	Yes (Prometheus + Cruise Control	No	No	No
Graceful Cluster Scaling (up and down)	Yes (using Cruise Control)	No	No	Yes

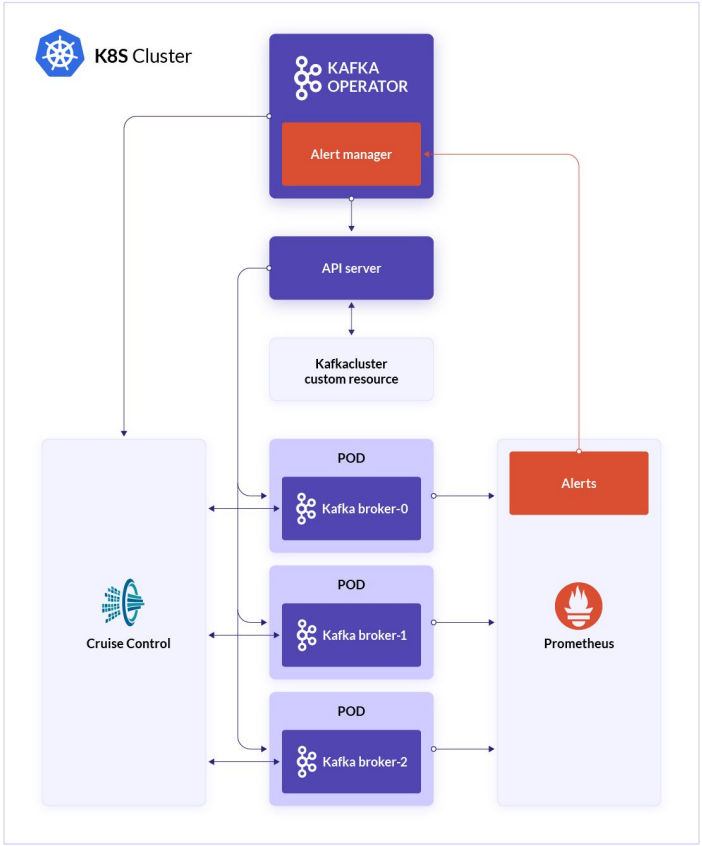
Schedule

- Kafka recap
- Another Operator ?
- Overview of the Operator
- Install
- CRD
- Topics
- Alerting / Scaling
- Recap
- Takeout

Overview

Key concepts :

- create and manage brokers as Pods
- talk to Linkedin CruiseControl for health
- use CRD for clusters, users, topics
- embed a Prometheus AlertManager
- dev in Go (CC in Java)



Cruise Control

https://github.com/linkedin/cruise-control

Key concepts :

- resource utilization tracking for brokers, topics, and partitions
- multi-goal rebalance proposal generation
- anomaly detection, alerting, and self-healing for the Kafka cluster
- dev in Java
- slides at https://www.slideshare.net/JiangjieQin/introduction-to-kafka-cruise-control-68180931

Schedule

- Kafka recap
- Another Operator ?
- Overview of the Operator
- Install
- CRD
- Topics
- Alerting / Scaling
- Recap
- Takeout

Install

Use the Helm chart

```
git clone https://github.com/banzaicloud/kafka-operator.git
cd kafka-operator
helm template charts/kafka-operator \
--set fullnameOverride=kafka \
--set prometheus.enabled=false \
--set prometheusMetrics.authProxy.enabled=false \
--set operator.image.repository="< private repo >/kafka-operator" \
--set operator.image.tag="0.6.1" \
--set prometheus.server.configMapOverrideName="" \
--set imagePullSecrets={docker-images-registry-secret} \
--namespace tools > charts/kafka-operator/generated.yaml

kubectl apply -n tools  charts/kafka-operator/generated.yaml
```

Schedule

- Kafka recap
- Another Operator ?
- Overview of the Operator
- Install
- CRD
- Topics
- Alerting / Scaling
- Recap
- Takeout

CRDs (1)

create a KafkaCluster to… create a Kafka Cluster

```
apiVersion: kafka.banzaicloud.io/v1beta1
kind: KafkaCluster
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
    kafka_cr: kf-kafka
  name: kf-kafka
  namespace: alerting
spec:
  headlessServiceEnabled: false
  zkAddresses:
    - "zk-zookeeper:2181"
  rackAwareness:
    labels:
      - "failure-domain.beta.kubernetes.io/region"
      - "failure-domain.beta.kubernetes.io/zone"
  oneBrokerPerNode: false
  clusterImage: "your-own-repo/kafka:2.3.0.7"
  rollingUpgradeConfig:
    failureThreshold: 1
```

CRDs (2)

Create a **brokerConfigGroup** so all your brokers inherits the same defaults

```
brokerConfigGroups:
    # Specify desired group name (eg., 'default_group')
    default_group:
      # all the brokerConfig settings are available here
      serviceAccountName: "kf-kafka"
      imagePullSecrets:
        - name: docker-images-registry
      kafkaJvmPerfOpts: "-server -XX:+UseG1GC
-XX:MaxGCPauseMillis=20
-XX:InitiatingHeapOccupancyPercent=35
-XX:+ExplicitGCInvokesConcurrent -Djava.awt.headless=true
-Dsun.net.inetaddr.ttl=60 -Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false
-Djava.rmi.server.hostname=${HOSTNAME}
-Dcom.sun.management.jmxremote.rmi.port=9099"
      storageConfigs:
        - mountPath: "/kafka-logs"
          pvcSpec:
            accessModes:
              - ReadWriteOnce
            storageClassName: ssd
            resources:
              requests:
                storage: 30Gi
```

CRDs (3)

Create your individual brokers

This is the key : you can manage each broker individually.

```
brokers:
    - id:  0
      brokerConfigGroup: "default_group"
      brokerConfig:
        resourceRequirements:
          limits:
            memory: "3Gi"
          requests:
            cpu: "0.3"
            memory: "512Mi"
    - id:  1
      brokerConfigGroup: "default_group"
      brokerConfig:
        resourceRequirements:
          limits:
            memory: "3Gi"
          requests:
            cpu: "0.3"
            memory: "512Mi"
    - id:  2
      brokerConfigGroup: "default_group"
      brokerConfig:
        resourceRequirements:
          limits:
            memory: "3Gi"
          requests:
            cpu: "0.3"
            memory: "512Mi"
```

CRDs (4)

More config…

and Listeners. This is where you can add SSL

```
#clusterWideConfig: |
  #  background.threads=2
readOnlyConfig: |
  offsets.topic.replication.factor=2
  default.replication.factor=2
  transaction.state.log.min.isr=1
  log.dirs=/kafka-logs/data
  delete.topic.enable=true
  num.partitions=32
  auto.create.topics.enable=false
  transaction.state.log.replication.factor=2
```

```
listenersConfig:
    internalListeners:
      - type: "plaintext"
        name: "plaintext"
        containerPort: 9092
        usedForInnerBrokerCommunication: true
```

CRDs (5)

Define CruiseControl objectives.

If the cluster use more resources, CC will take action to remediate, like rebalance the topics/partitions/consumergroups or add a new PVC...

```
cruiseControlConfig:
    image: "solsson/kafka-cruise-control:latest"
    serviceAccountName: "kf-kafka"
    config: |
...
    capacityConfig: |
      {
        "brokerCapacities":[
          {
            "brokerId": "-1",
            "capacity": {
              "DISK": "200000",
              "CPU": "100",
              "NW_IN": "50000",
              "NW_OUT": "50000"
            },
            "doc": "This is the default capacity. Capacity
unit used for disk is in MB, cpu is in percentage, network
throughput is in KB."
          }
        ]
      }
    clusterConfigs: |
      {
        "min.insync.replicas": 2
      }
```

Schedule

- Kafka recap
- Another Operator ?
- Overview of the Operator
- Install
- CRD
- Topics
- Alerting / Scaling
- Recap
- Takeout

Topics

The operator will create the topics if they do not exist.

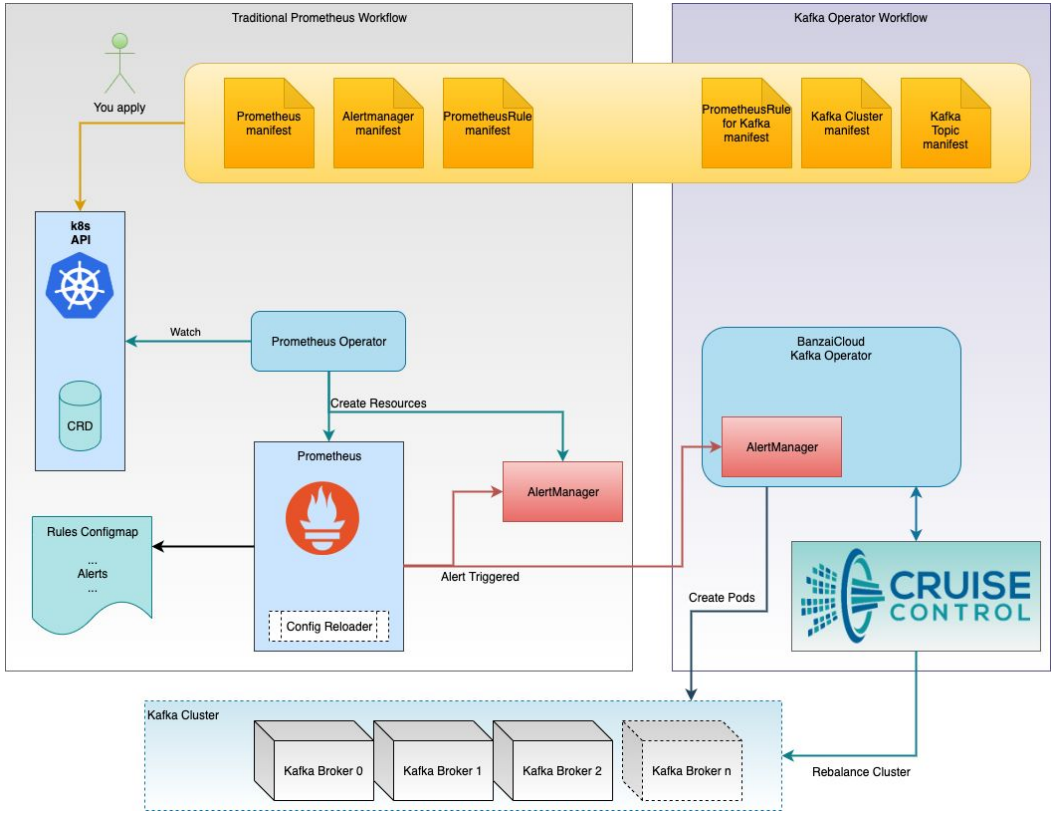
- will not modify a topic that already exists
- will not delete a topic if you remove the CRD
- will not create the topic if the number of broker < replicationFactor (will keep retrying)

```
apiVersion: kafka.banzaicloud.io/v1alpha1
kind: KafkaTopic
metadata:
  name: compactedtopic
  namespace: alerting
spec:
  clusterRef:
    name: kf-kafka
  name: compactedtopic
  partitions: 8
  replicationFactor: 2
  config:
    segment.bytes: "104857600"
    delete.retention.ms: "8640000"
    retention.ms: "259200000"
    cleanup.policy: "compact"
---
apiVersion: kafka.banzaicloud.io/v1alpha1
kind: KafkaTopic
metadata:
  name: regulartopic
  namespace: alerting
spec:
  clusterRef:
    name: kf-kafka
  name: regulartopic
  partitions: 128
  replicationFactor: 2
```

Schedule

- Kafka recap
- Another Operator ?
- Overview of the Operator
- Install
- CRD
- Topics
- Alerting / Scaling
- Recap
- Takeout

Alerting


Traditional Prometheus Workflow
Kafka Operator Workflow
You apply
Prometheus manifest
Alertmanager manifest
PrometheusRule manifest
PrometheusRule for Kafka manifest
Kafka Cluster manifest
Kafka Topic manifest
k8s API
CRD
Watch
Prometheus Operator
Create Resources
Prometheus
AlertManager
Rules Configmap
...
Alerts
...
Config Reloader
Alert Triggered
BanzaiCloud Kafka Operator
AlertManager
Create Pods
CRUISE CONTROL
Kafka Cluster
Kafka Broker 0
Kafka Broker 1
Kafka Broker 2
Kafka Broker n
Rebalance Cluster

Alerting

From the Prometheus point of view :

- Prometheus monitors the kafka cluster
- Prometheus have the Kafka Operator as an AlertManager endpoint
- Alert Rules will trigger and Prometheus will send a message with specific values to the Kafka Operator AlertManager

Alerting

Example rule that will trigger and notify the Kafka Operator.

In this case, the Operator will apply the **‘upscale’** command.

spec.rules.annotations is used to send specific config to the Operator

Commands : upScale, downScale, AddPVC, more to come.

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  creationTimestamp: null
  labels:
    prometheus: k8s
    role: alert-rules
  name: kafka-alerts
spec:
  groups:
  - name: KafkaAlerts
    rules:
    - alert: BrokerOverLoaded
      expr: avg(sum by(brokerId, kafka_cr, namespace)
(rate(kafka_network_requestmetrics_requests_total[15m]))) > 500
      for: 5m
      labels:
        severity: alert
      annotations:
        description: 'broker {{ $labels.brokerId }} overloaded
(current value is: {{ $value }})'
        summary: 'broker overloaded'
        brokerConfigGroup: 'default_group'
        command: 'upScale'
```

Schedule

- Kafka recap
- Another Operator ?
- Overview of the Operator
- Install
- CRD
- Topics
- Alerting / Scaling
- Recap
- Takeout

Recap

- better control of the cluster
- support Kafka 2.3.x
- auto-rebalance if out of normal operations
- cluster health not based on stupid health checks
- easier upgrade or config change
- cluster auto-managed from Prometheus metrics
- external access to Kafka through Envoy + SSL
- a lot of features and more to come (Hashicorp Vault...)
- great team/support from Banzaicloud (check them on Slack)
- you can contribute!

Cons :

- rely on cert-manager and ca-injector for Topic management (but is it bad? you're using it already anyways)
- too much control? Maybe too complex for your setup
- you have to install/manage Zookeeper yourself (like all other Kafka operators though)

Schedule

- Kafka recap
- Another Operator ?
- Overview of the Operator
- Install
- CRD
- Topics
- Alerting / Scaling
- Recap
- Takeout

Takeout

Banzai Cloud Kafka Operator :

- https://banzaicloud.com/products/kafka-operator/
- https://banzaicloud.com/blog/kafka-operator/
- https://github.com/banzaicloud/kafka-operator

Cruise Control

- https://github.com/linkedin/cruise-control
- https://www.slideshare.net/JiangjieQin/introduction-to-kafka-cruise-control-68180931

My blog posts :

- Install : https://medium.com/@prune998/banzaicloud-kafka-operator-tour-56fca7d6261e
- Scaling : https://medium.com/@prune998/banzaicloud-kafka-operator-and-broker-autoscaling-1c7324260de1

Questions ?

Thanks !

(we’re Hiring !)

https://www.tetrate.io/about-us/careers/

Keep in touch :

prune@lecentre.net

https://github.com/prune998

https://www.linkedin.com/in/prune/