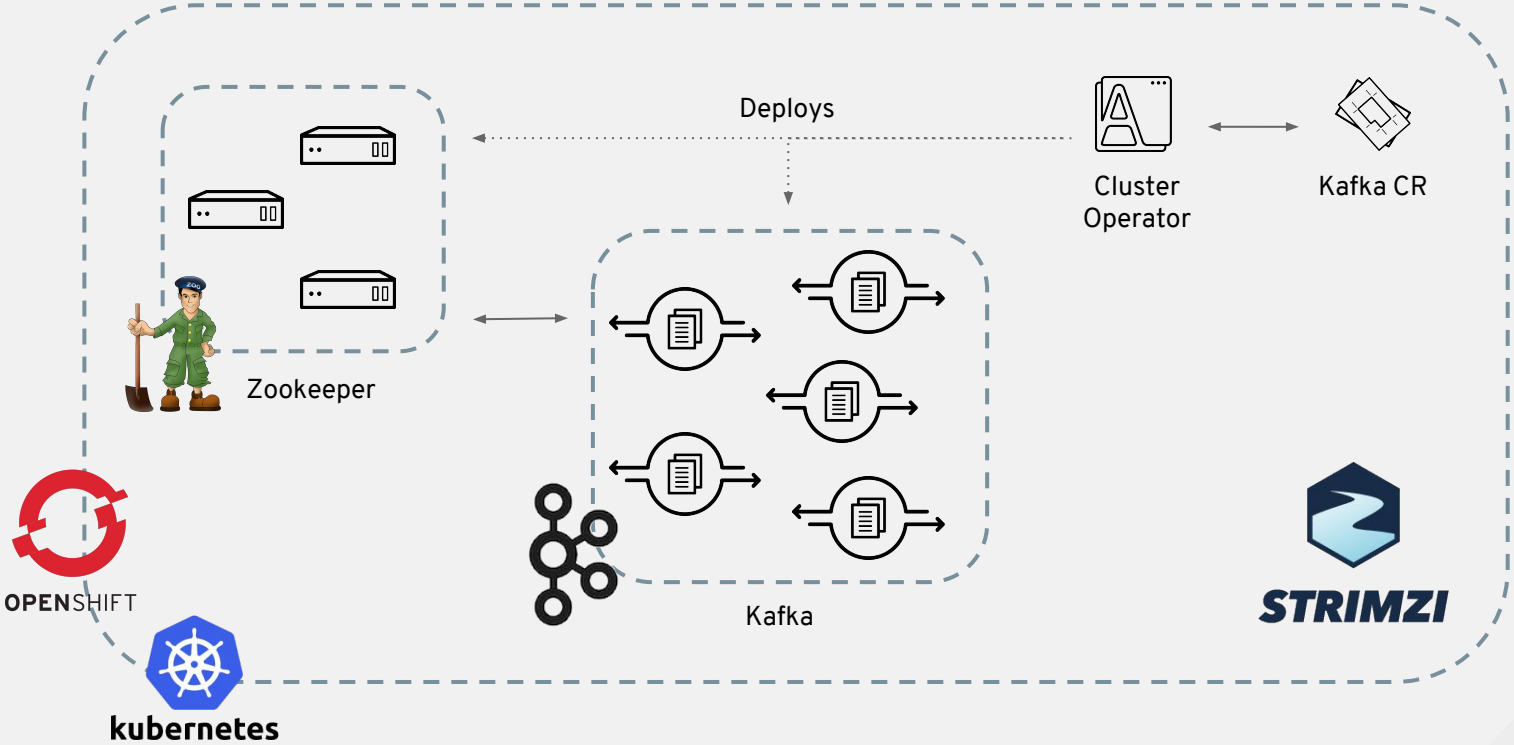
Cluster Operator - Simple



Cluster Operator: Managed Resources

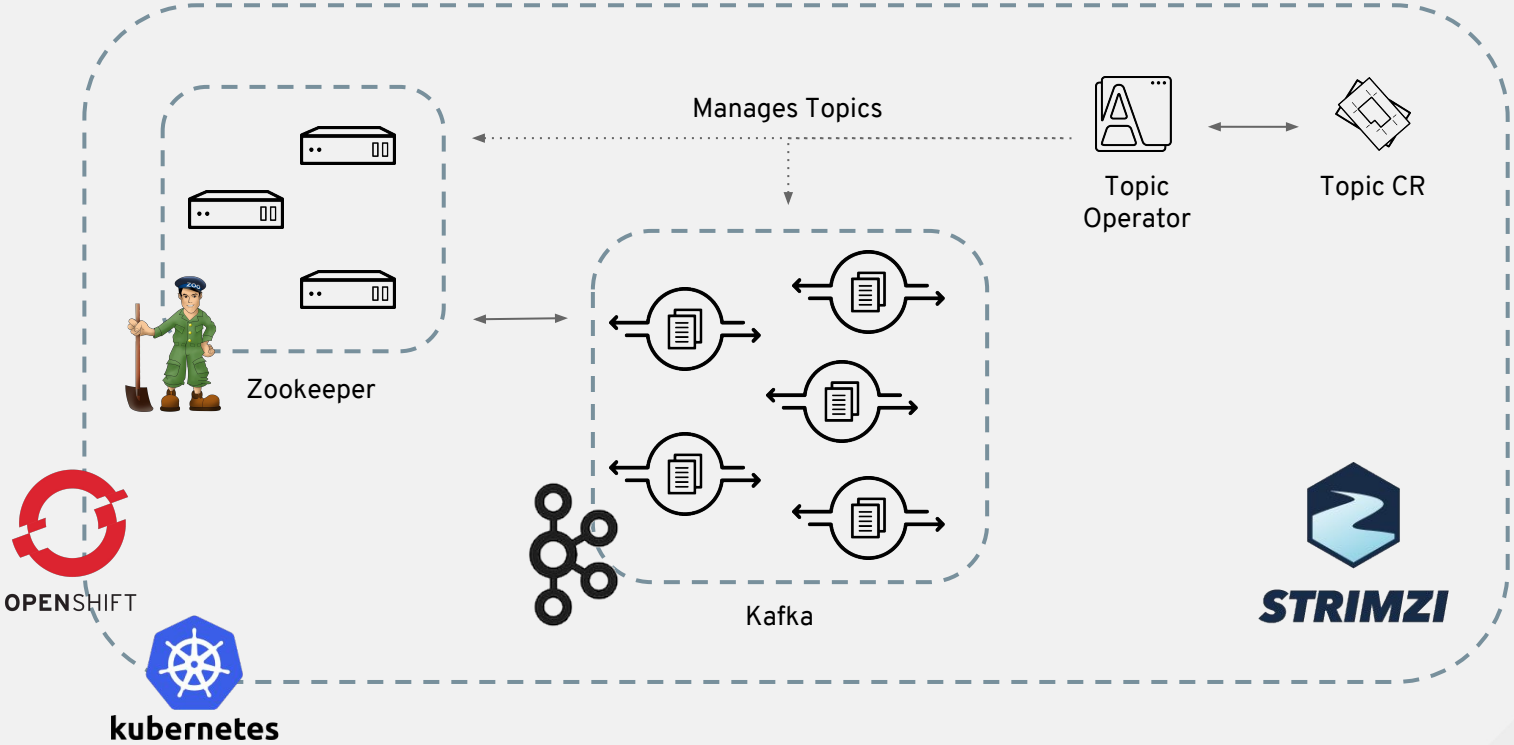
Zookeeper

- StatefulSet
- Service (regular+headless)
- PVC
- Secrets (for TLS)
- CM

Kafka

- StatefulSet
- Service (regular+headless)
- PVC
- Secrets (for TLS)
- CM

Topic Operator - Simple



Anatomy of the `Topic` Custom Resource

```
apiVersion: topic.kafka.strimzi.io/v1alpha1
kind: Topic
metadata:
  name: my-topic
replicas: 3
partitions: 12
config:
  retention.ms: 3600000
  segment.bytes: 1073741824
```

A Map<string, string|number|boolean>

Getting out of Sync

- It's possible for Kafka applications to create topics dynamically.
- Depending on broker configuration this happens automatically simply by producing to/consuming from a topic which doesn't exist
- Which means topics would exist in Kafka-world, but with no `Topic` resource in K8s-world.
- And Kafka exposes APIs for deleting and modifying topics too.
- And these features are used in the Kafka ecosystem (e.g. Kafka Streams)
- We can't tell users not to use these features.
- So Topic Operator would have to synchronize state in both directions

Possibility of Conflict

Which end changed what?

`Topic` changed in Kubernetes

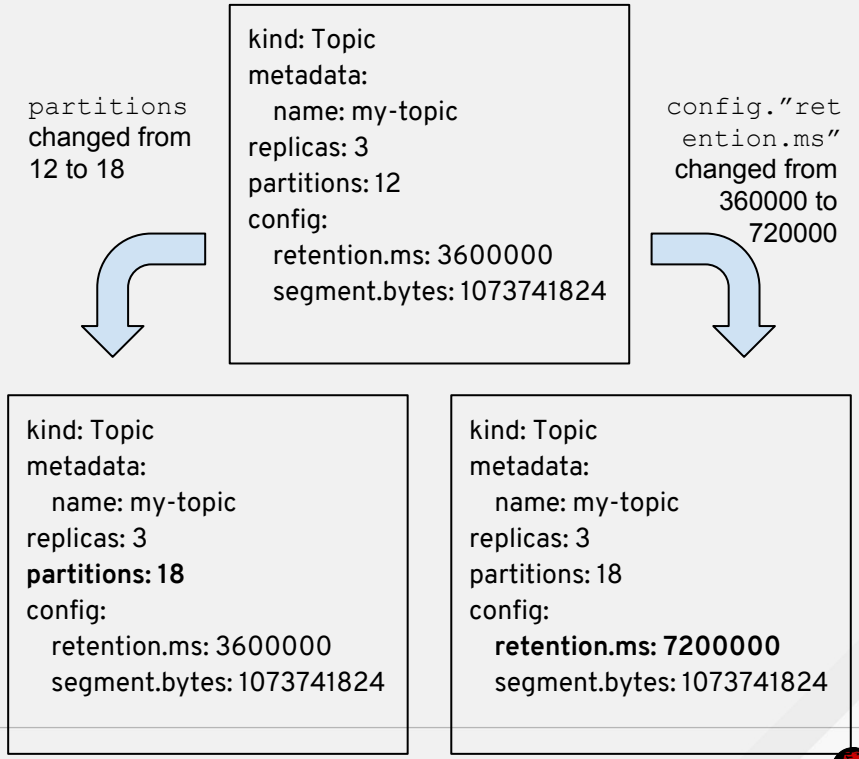
```
apiVersion: topic.kafka.strimzi.io/v1alpha1
kind: Topic
metadata:
  name: my-topic
replicas: 3
partitions: 18
config:
  retention.ms: 3600000
  segment.bytes: 1073741824
```

Topic changed in Kafka

```
apiVersion: topic.kafka.strimzi.io/v1alpha1
kind: Topic
metadata:
  name: my-topic
replicas: 3
partitions: 12
config:
  retention.ms: 7200000
  segment.bytes: 1073741824
```

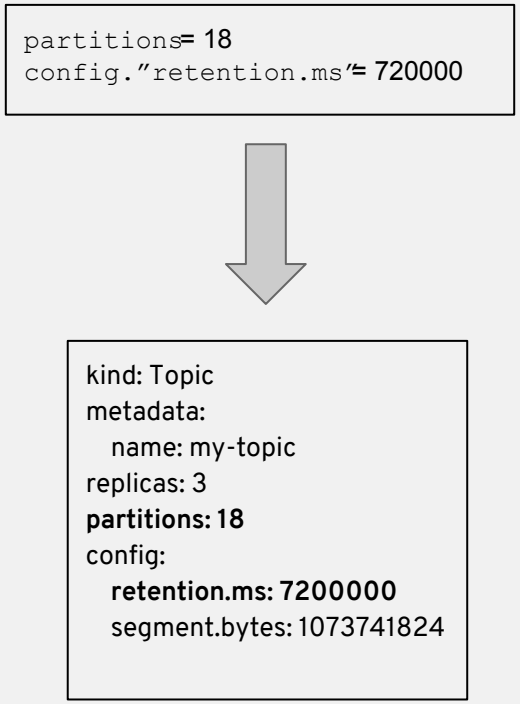
3-way diff

- The problem is neither end is the “source of truth” ⇒ neither can be “trusted” when figuring out what changed
- Keep our own “source of truth” which users/applications cannot modify.
- Figure out what changed at on each side by comparing with this.
- Construct the “union” of these two diffs (detect when their intersection is non-empty ⇒ true conflict)
- Apply union diff to both sides
- Use a winner policy in case of true conflict.



3-way diff

- The problem is neither end is the “source of truth” ⇒ neither can be “trusted” when figuring out what changed
- Keep our own “source of truth” which users/applications cannot modify.
- Figure out what changed at on each side by comparing with this.
- Construct the “union” of these two diffs (detect when their intersection is non-empty ⇒ true conflict)
- Apply union diff to both sides
- Use a winner policy in case of true conflict.



Consequences

- The “private state” makes the operator stateful.
- This state needs to be highly available.
- Ordering of updates matters for fault tolerance correctness
- For us it made sense to use Zookeeper.
- Other options? Etcd