KubeCon

North America 2017

Kafka-Operator: Managing and Operating Kafka Clusters in Kubernetes

Nenad Bogojević, Solution Architect, *Amadeus*

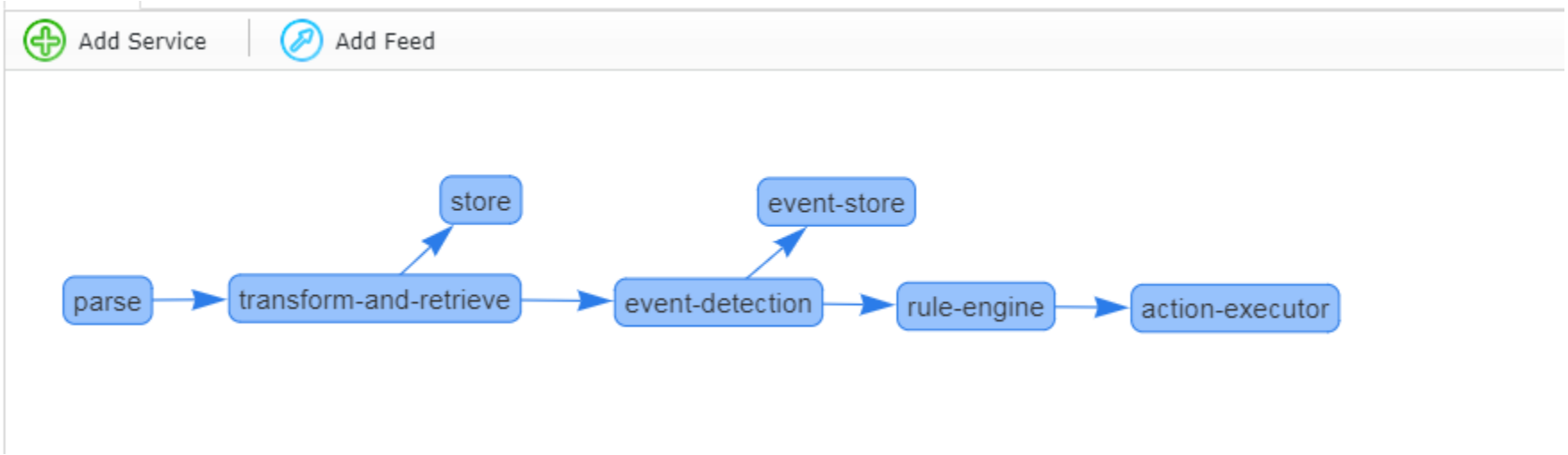
About Me & Amadeus

- amadeus provides IT services for travel industry
  - 30 years
  - Runs kubernetes deployments almost since inception
    - In own premises, in public clouds

- Me: Nenad Bogojević
  - Solution architect involved in design of new platform and migrating existing applications

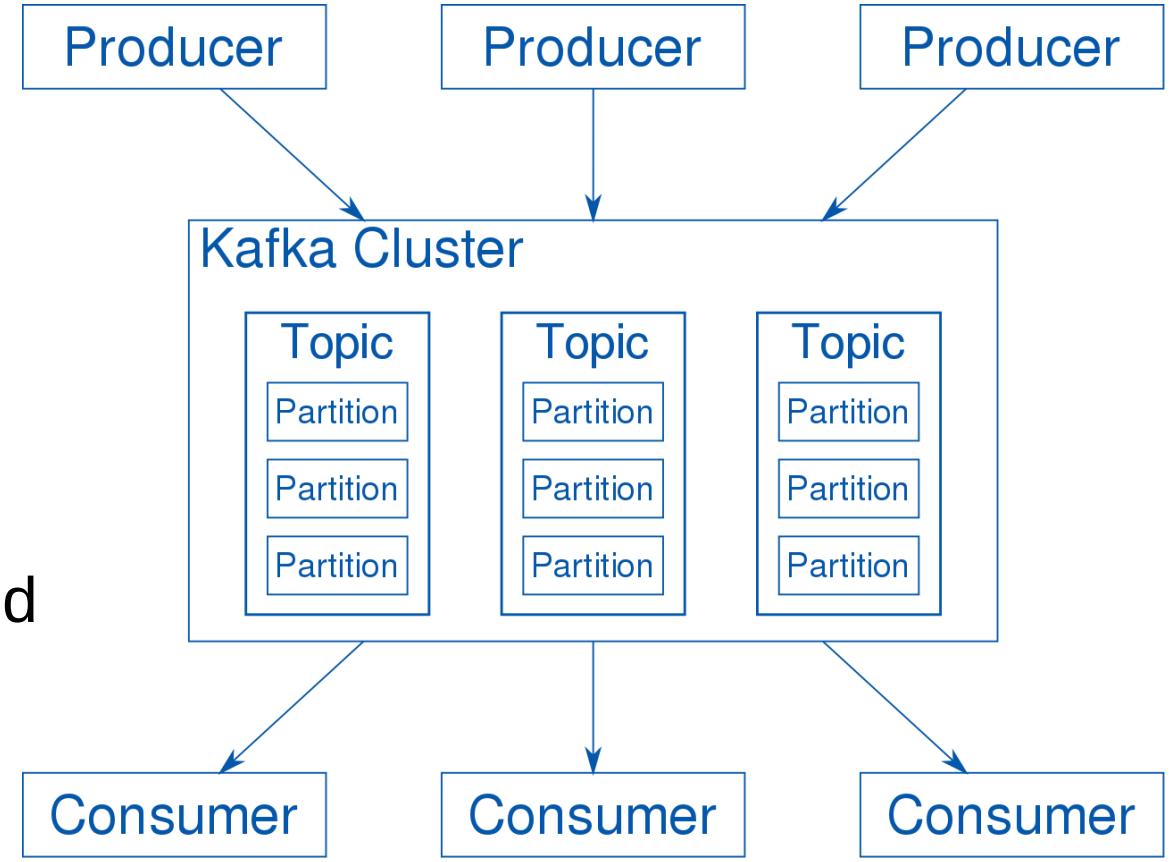
Amadeus uses Kafka

- For log & events collection
  - Installed with puppet
- For *streaming platform*



What is Kafka?

- Streaming platform
  - cluster of servers called *brokers*.
  - stores streams of *records* in *topics*.
  - records stored in *partitioned* log across brokers
  - partitions *replicated* across brokers
  - records generated by *producers* and read by *consumers* (*clients*)



Kafka in Kubernetes?

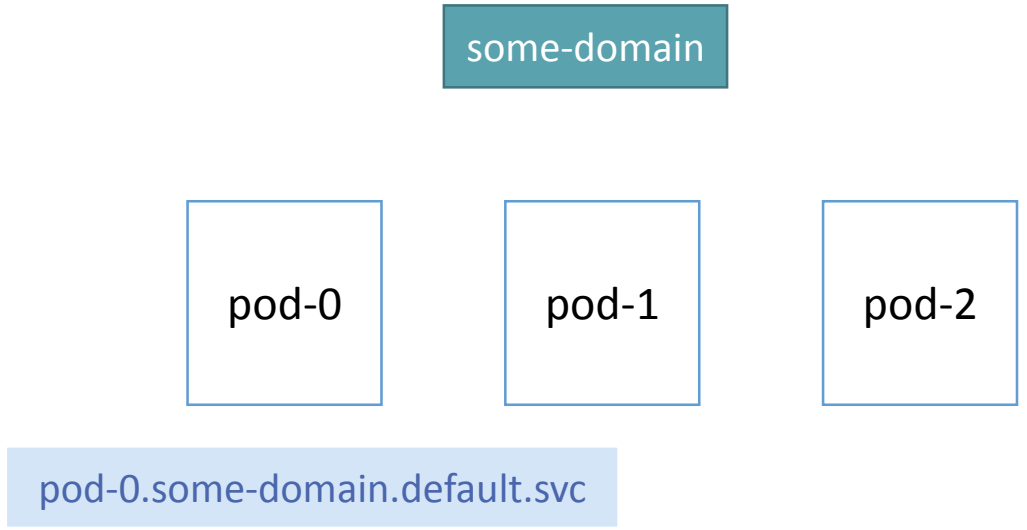
~~broker-178523330-61d6n~~

- Kafka cluster
  - Each broker has identity – need to find each other
  - Brokers need persistence to store partition logs
- Zookeeper cluster
  - Another cluster with persistence

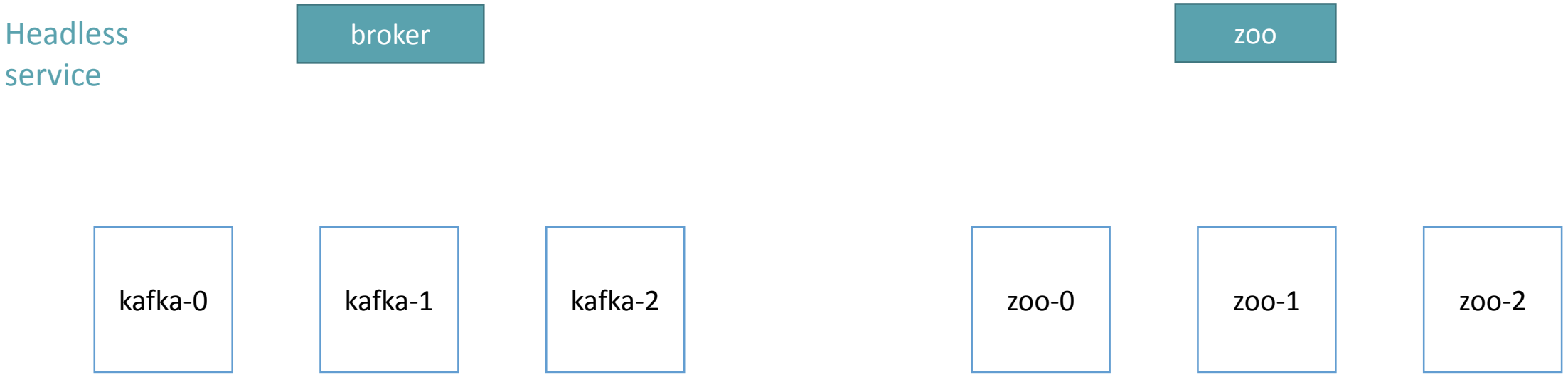
- StatefulSet

StatefulSet refresher

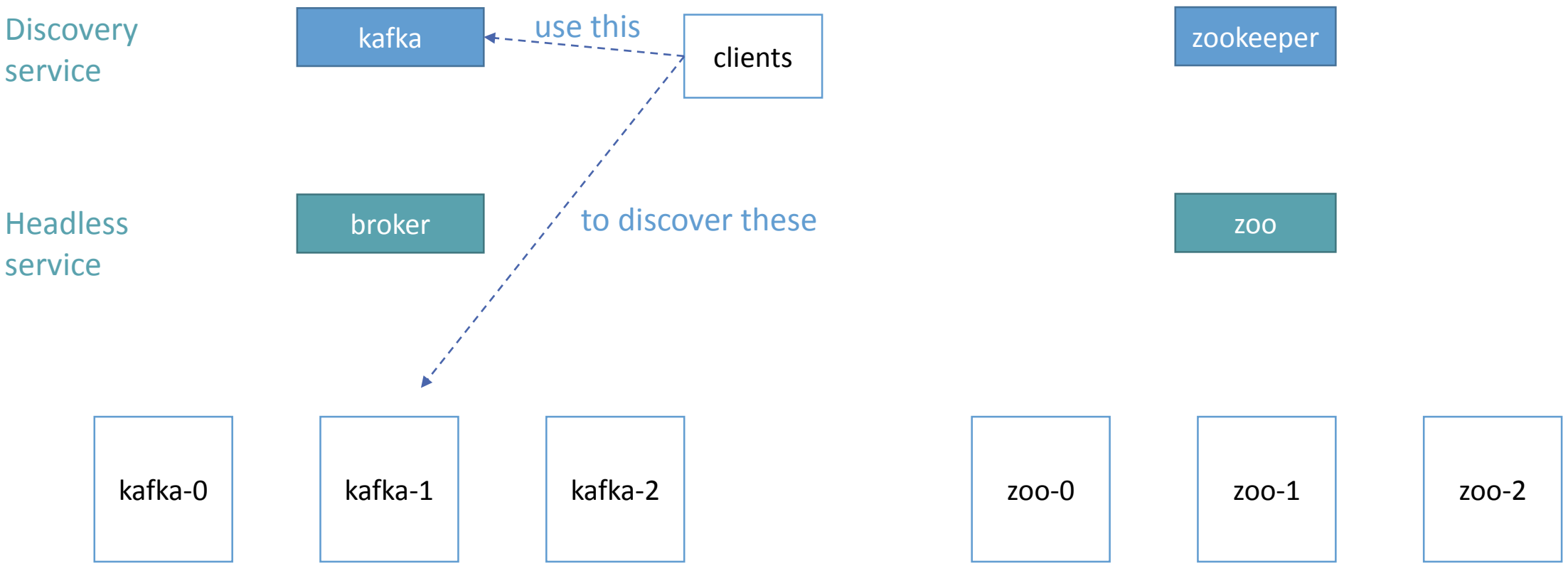
- Provide stable pod identity
- Provide stable storage
- Ordered startup, shutdown
- Rolling updates



Kafka and Zookeeper StatefulSet



Kafka and Zookeeper StatefulSet



Containers, Charts & Co

- Containers
  - https://github.com/Yolean/kubernetes-kafka
- Charts & co
  - https://github.com/kubernetes/contrib/tree/master/statefulsets/kafka
  - https://github.com/EnMasseProject/barnabas
  - **https://github.com/nbogojevic/kubernetes-kafka**
- Operators:
  - https://github.com/krallistic/kafka-operator
  - **https://github.com/nbogojevic/kafka-operator**

KubeCon

North America 2017

Demo time

Affinity

- Node selectors
  - Using node selector to land instances on machines with good hardware (e.g. SSD).
- Anti-affinity
  - Using anti affinity to spread instances across different physical machines

```yaml
nodeSelector:
  # ...
  disk: fast
```

```yaml
affinity:
  podAntiAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
    - weight: 1
      podAffinityTerm:
        labelSelector:
          matchExpressions:
          - key: app
            operator: In
            values:
            - kafka-cluster
        topologyKey: "kubernetes.io/hostname"
```

Storage

StatefulSet

```
containers:
  # ...
  volumeMounts:
  - name: datadir
    mountPath: /opt/kafka/data

# Is empty directory good idea?
volumes:
- name: datadir
  emptyDir: {}
```

- If your pod restarts, it will get same emptyDir, and recover data
- If pod moves to other node, it will lose data!

- Kafka performance is mostly disk bound (network also)

- Kafka has replication!

Persistence

- Common wisdom
  - Use persistent volume, otherwise you'll lose Kafka logs
- Relying on Kafka replication
  - Use empty volume
    - if container crashes, you keep the logs
    - if node crashes, rely on replication
    - have enough replicas and brokers
      - 5 broker cluster, 2 replicas – you can lose 1 broker
- Soon: local persistent volume

Monitoring (Kubernetes)

StatefulSet

```
# Checks that server is ready
readinessProbe:
  exec:
    command:
    - bin/kafka-broker-api-versions.sh
    - --bootstrap-server=localhost:9092
# Checks that server accepts connection
livenessProbe:
  tcpSocket:
    port: 9092
```

Monitoring (JMX and Prometheus)

StatefulSet

```
template:
  metadata:
    annotations:
      prometheus.io/scrape: "true"
      prometheus.io/path: "/metrics"
      prometheus.io/port: "9779"
```

Container

kafka-container:

```
FROM fabric8/s2i-java
ADD kafka
```

KubeCon

North America 2017

Operators

What are Operators?

- Transposing domain knowledge of SRE/operations into executable code

- @Amadeus we use operators for
  - Prometheus
  - Redis cluster
  - **Workflow**
  - Kafka

Provisioning Clusters

- Helm charts/Openshift templates
- Once platform is set up, cluster stays in place
- We can scale up
- But scale down, evacuation and upgrades are tricky

How About Topics?

- Operating topics means
  - Make sure that topic exists in target environments
  - Make sure that topic is deleted once it is no longer used
  - Propagate same configuration across environments
  - Configure retention based on available disk space
  - Configure clients with credentials
  - Deliver configuration and requirements as code

Topics as a Code

- ConfigMap or CustomResource describing
  - Name
  - Partition count
  - Replication factor
  - Topic properties
- Equivalent to provision/unprovision of ServiceCatalog

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: sample-topic
  labels:
    config: kafka-topic
data:
  partitions: "20"
  replication-factor: "2"
  properties: |
    retention.ms=1000000
```

Topics – Access Control

- Deployments describe which topic they use via annotations
- Operator assigns user, generates java JAAS configuration, creates secret
- Equivalent to bind/unbind of ServiceCatalog

```
labels:
  app: kafka-client
  kafka-operator: inject-credentials
annotations:
  "topic.kafka.nb/consumes": "input-topic"
  "topic.kafka.nb/produces": "output-topic"
```

Kafka Upgrades

- Inter broker protocol
  - Set protocol version to current
  - Upgrade brokers one by one
  - Set protocol version to new
- Storage format
  - Have consumers on up-to date version
  - Update format version to new
- Easy way out: don’t upgrade, but re-create?

Performance

- Dominated by disk I/O – using SSD
  - Then by network
  - Almost never CPU or memory (2-4 VCPU, 2-4GB), half for JVM heap
- 100K messages/sec
- Getting zookeeper and brokers on same nodes reduces network – same goes with clients ;)

KubeCon

North America 2017

Thanks!

K8S & Kafka, Pulse! tribe, and in particular Serge Beuzit, David Benque, Mathieu Bruyen, Laurent Cognard, Florent Coquelet, Pierre Dor, Yassine Ferkouche, Vincent Magry, Clement Seveillac, Stefano Troiani