Готовим свой Kubernetes operator на примере Spring Cloud Gateway

Киричук Игорь

Senior Tech Lead, Райффайзенбанк

Обо мне

- Senior Tech Lead в Райффайзенбанке
- Tech lead в нескольких командах разработки
- Занимаюсь разработкой и внедрением общих решений для команд
- Тесно работаю с kubernetes 4 года - и как потребитель и как поставщик

Контакты: @ing8ar

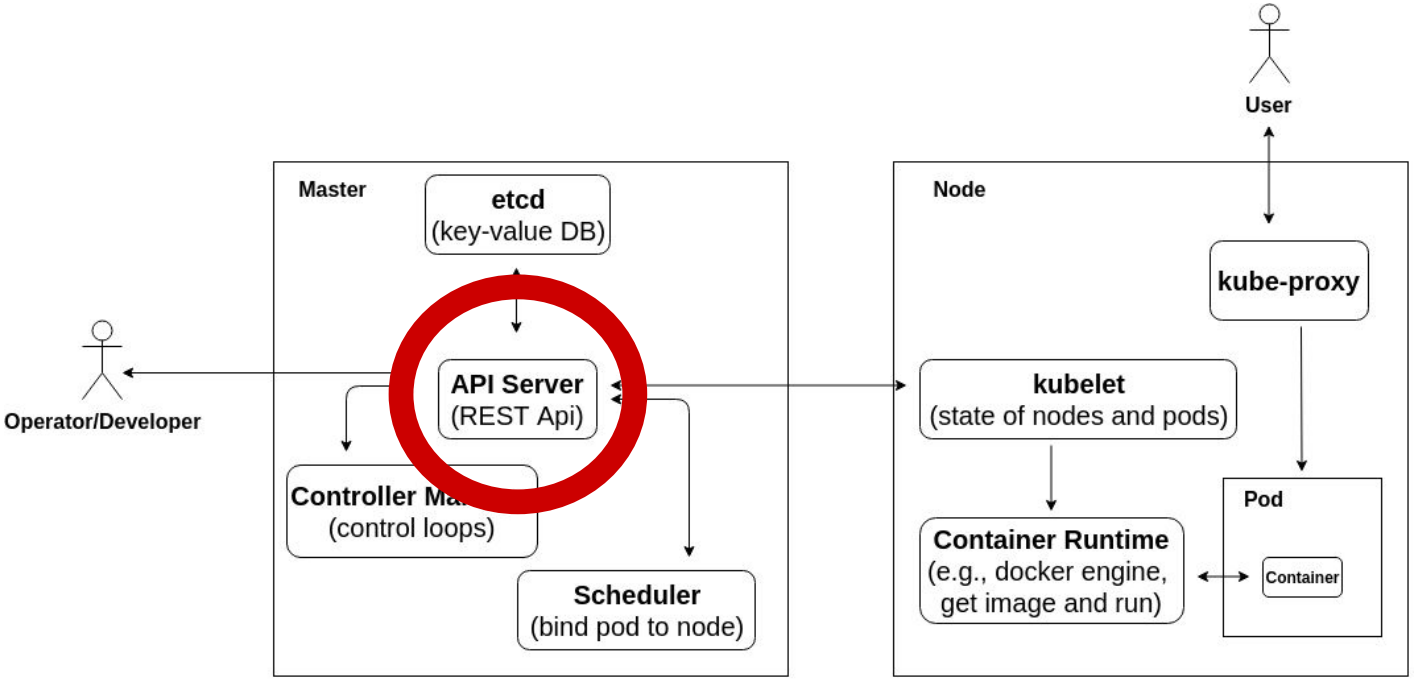
Kubernetes как конструктор

- Из простых компонентов выстраивается мощная экосистема
- Можно создавать свои компоненты
- Расширять экосистему можно самому

Kubernetes API

- Полнофункциональное REST API
- Позволяет создавать/менять/удалять любые сущности k8s
- Можно расширять посредством Custom Resource Defenition (CRD) и/или API Aggregation Layer
- С api можно работать как внутри кластера так и снаружи
- kubectl, kubeadm, operators - используют k8s api

Kubernetes API Server



https://michalswi.medium.com/introduction-to-kubernetes-api-the-way-to-understand-the-concept-of-kubernetes-operators-ed667385caf4

А что по Java?

- Официальный Java Client (https://github.com/kubernetes-client/java)
- Fabric8 Java Client (https://github.com/fabric8io/kubernetes-client)

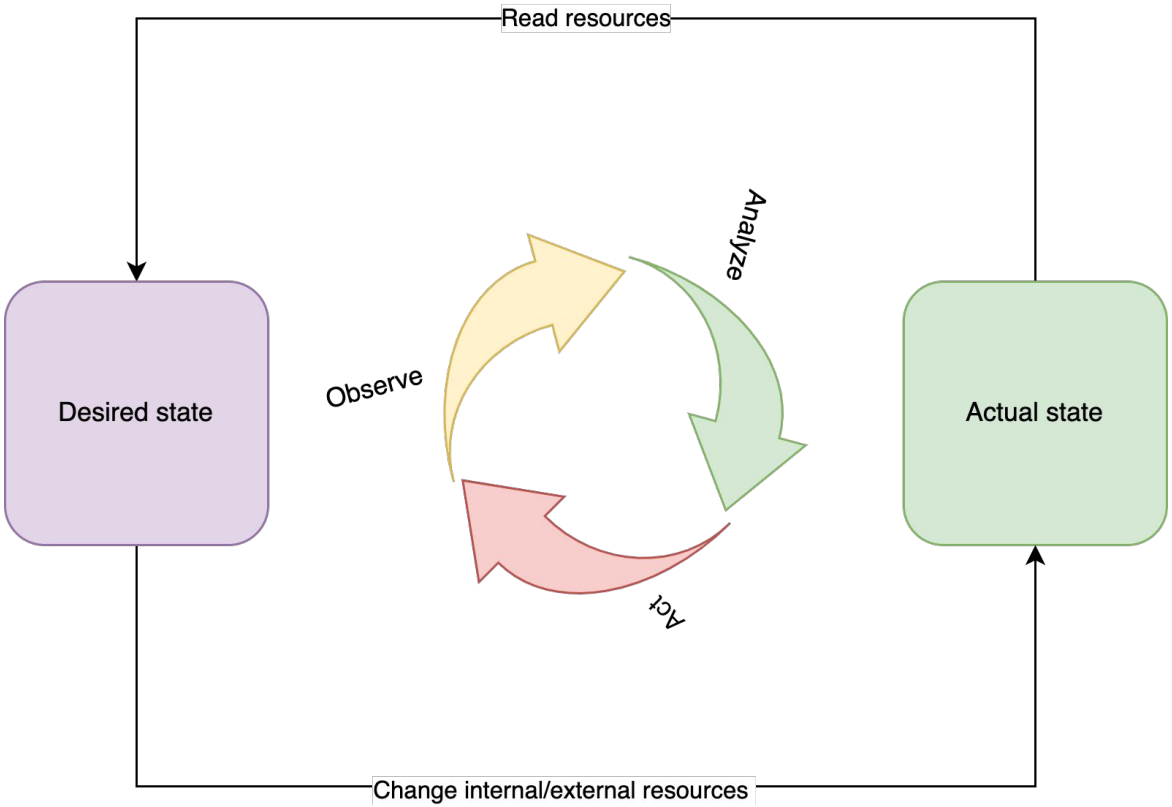
А что по Spring?

- Появилась поддержка части Spring Cloud функций используя нативное окружение k8s (spring-cloud-starter-kubernetes-all) (Discovery, Configuration via ConfigMap, etc)
- Автоконфигурации для Kubernetes Client Java и Fabric8 Java Client

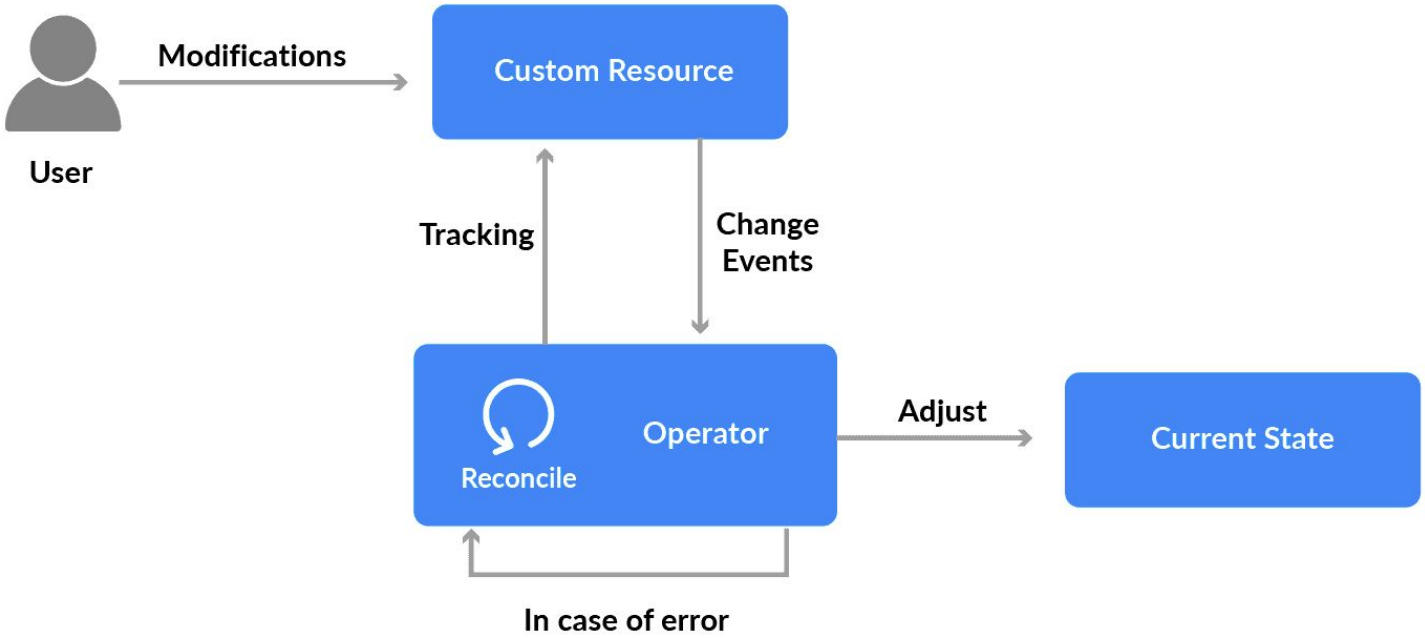
Operator pattern

- Один из способов расширить функциональность кластера без изменения кода самого kubernetes
- Для написания своего оператора достаточно того функционала который есть в k8s api clients
- В основе лежит принцип control loop

Control loop



Operator

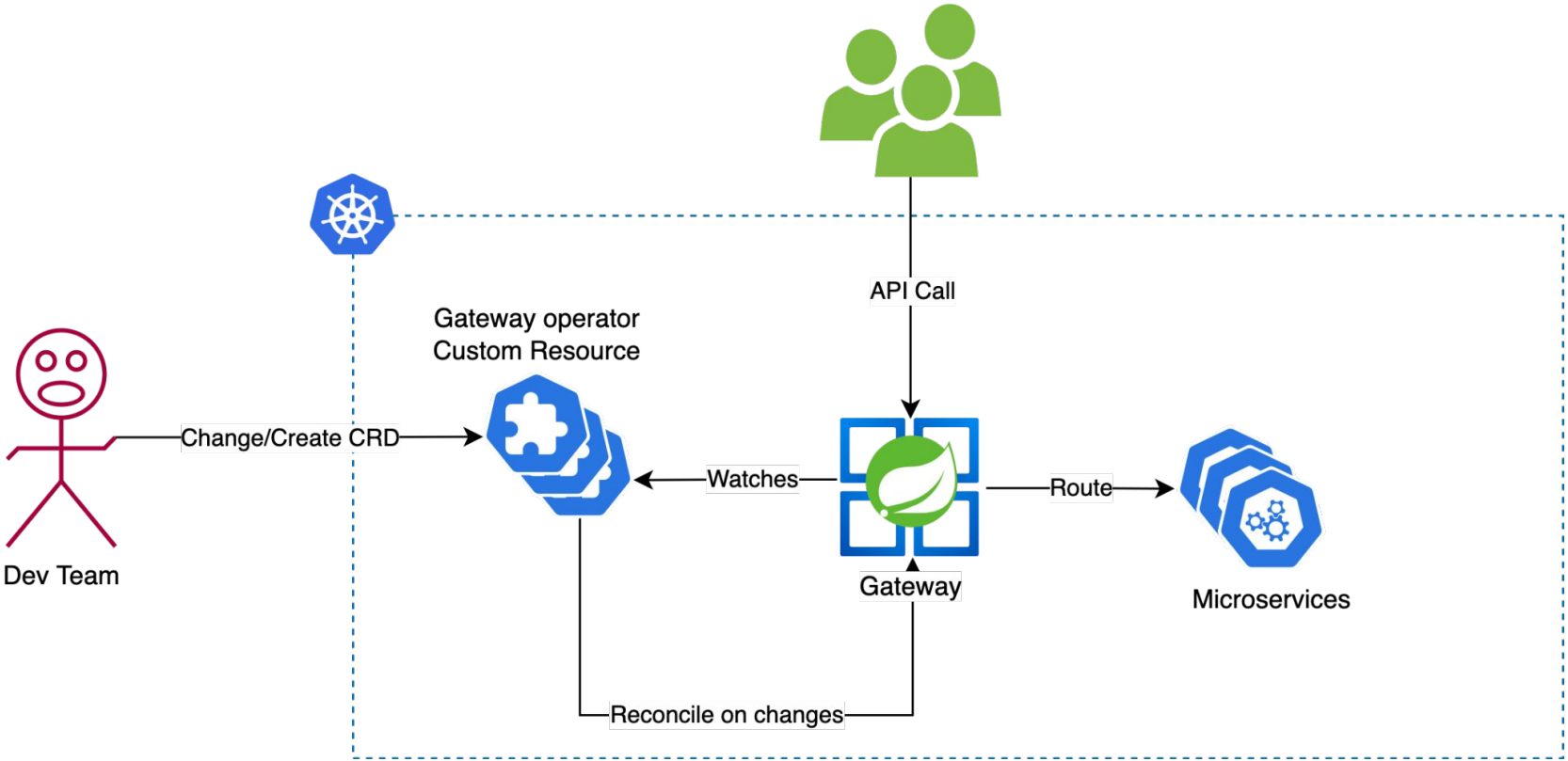


https://k21academy.com/docker-kubernetes/kubernetes-operator

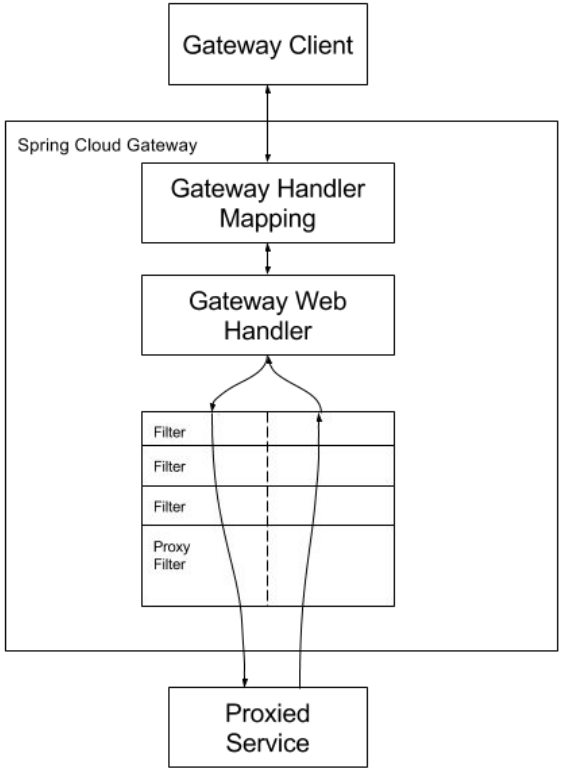
Что такое эти CRD?

- Custom Resource Definition
- Позволяет стандартизировать и валидировать создаваемые ресурсы (openAPIV3Schema)
- Через CRD мы можем управлять нашим оператором

Что мы будем готовить?



Spring Cloud Gateway



https://docs.spring.io/spring-cloud-gateway/docs/current/reference/html/#gateway-how-it-works

Spring Cloud Gateway Config >> CRD

```
spring:
 cloud:
   gateway:
     routes:
       - id: api-route
         uri: http://gateway-operator.default.svc:8080
         predicates:
           - Path=/api/features/**
         filters:
           - StripPrefix=2
```

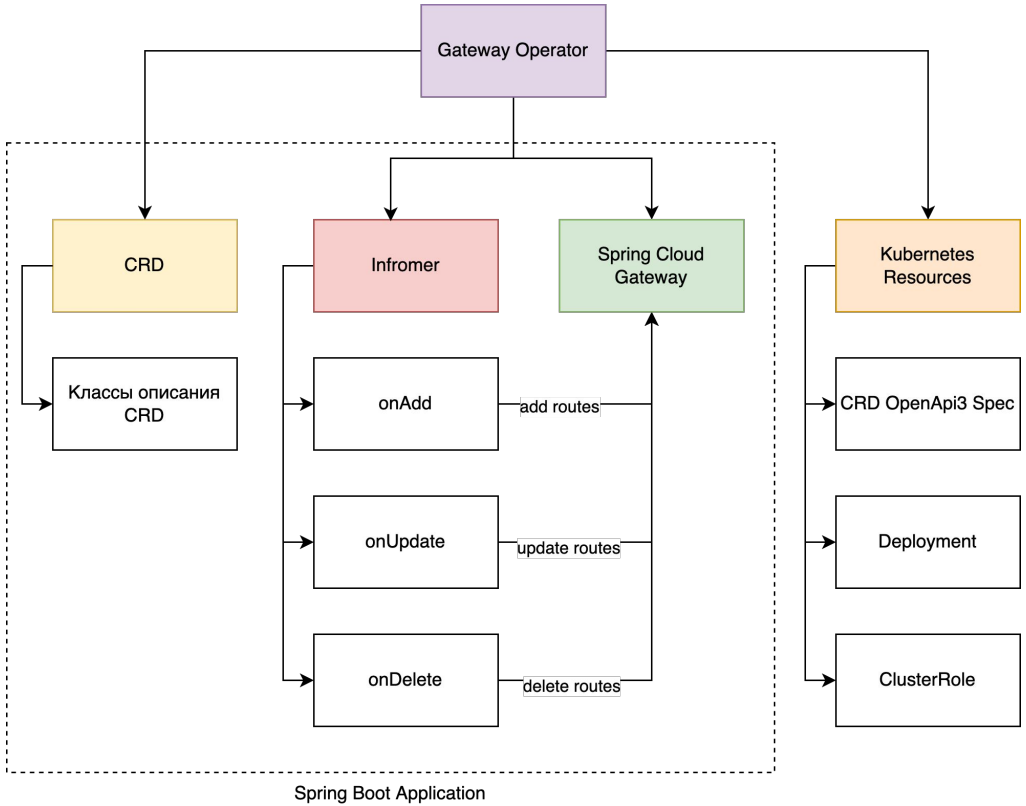


```
apiVersion: k8s.jpoint.ru/v1
kind: SpringCloudGatewayRouteConfig
metadata:
 name: to-gateway
spec:
 routes:
   - id: api-route
     uri: http://gateway-operator.default.svc:8080
     predicates:
       - Path=/api/features/**
     filters:
       - StripPrefix=2
```

Watch & Informer

- Watch
  - Слушает изменения (добавление, изменение, удаление) ресурса k8s через long polling
- Informer
  - Абстракции над Watch
  - Переподключается при разрыве соединения
  - Периодическое перечитывание ресурсов (reconcile)
  - Имеет локальный кэш

Архитектура нашего оператора



Локальная разработка

KinD

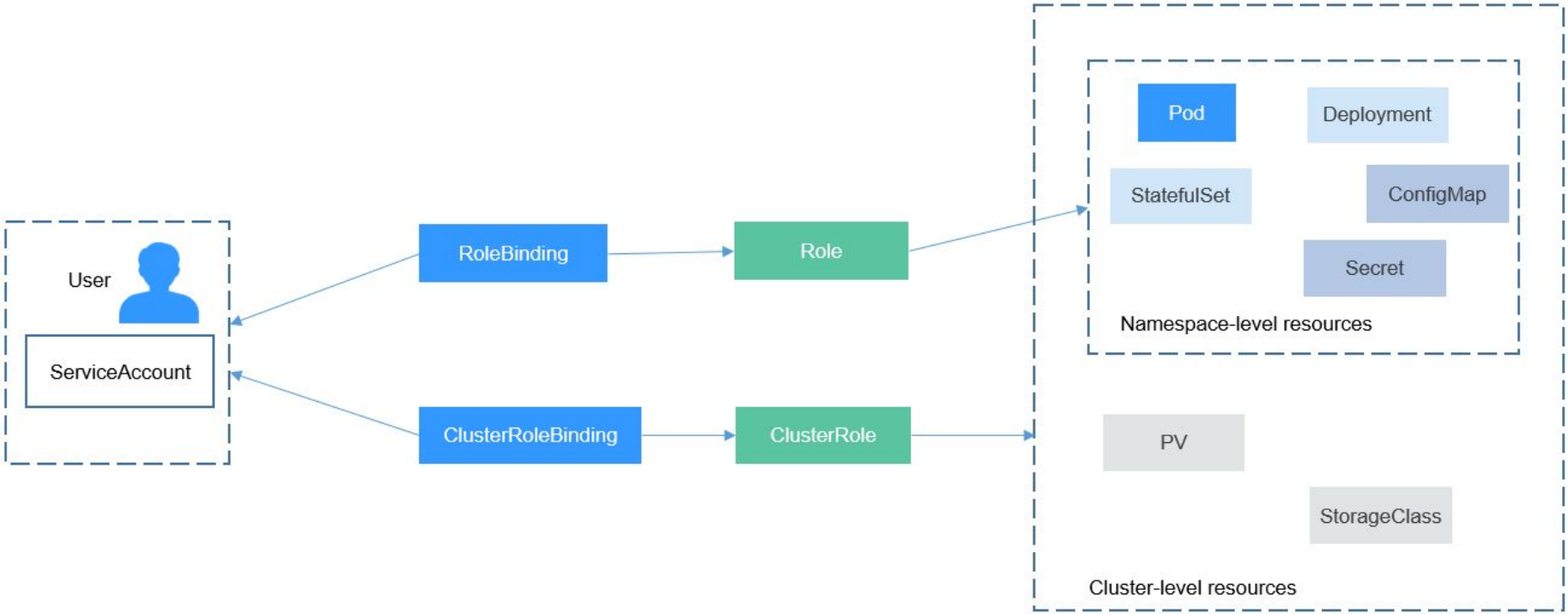
- Kubernetes in Docker
- Изначально создавался для тестирования kubernetes
- Отлично подходит для локальной разработки

Skaffold

- Хорошо подходит для локальной разработки для kubernetes - сам отслеживает изменения, запускает сборку и деплой
- Поддерживает много вариантов сборок образов и разные варианты деплоя
- Можно использовать в CI/CD

Coding time …

RBAC



https://support.huaweicloud.com/intl/en-us/usermanual-cce/cce_01_0189.html

Нюансы

- Необходимо учитывать reconcile ресурсов, они приходят в событие onUpdate и необходимо уметь сравнивать с текущими состоянием
- Горизонтальное масштабирование - необходимо учитывать при контроле применения изменений

Выводы

- Расширять функциональность kubernetes под ваши нужды очень просто
- Писать операторы на jvm + Spring Boot легко и быстро
- Не нужно быть kubernetes оператором чтобы написать свой оператор

Ссылки

https://github.com/ing8ar/gateway-operator

Вопросы?!