# Операторы, custom resource definitions

# Не забудь включить запись!

# План

- Operators
- Controllers
- Custom resource definition
- Инструменты для написания операторов:
  - Operator SDK (ansible)
  - Kopf
- Популярные операторы
  - Prometheus
  - Configconnector

# Операторы

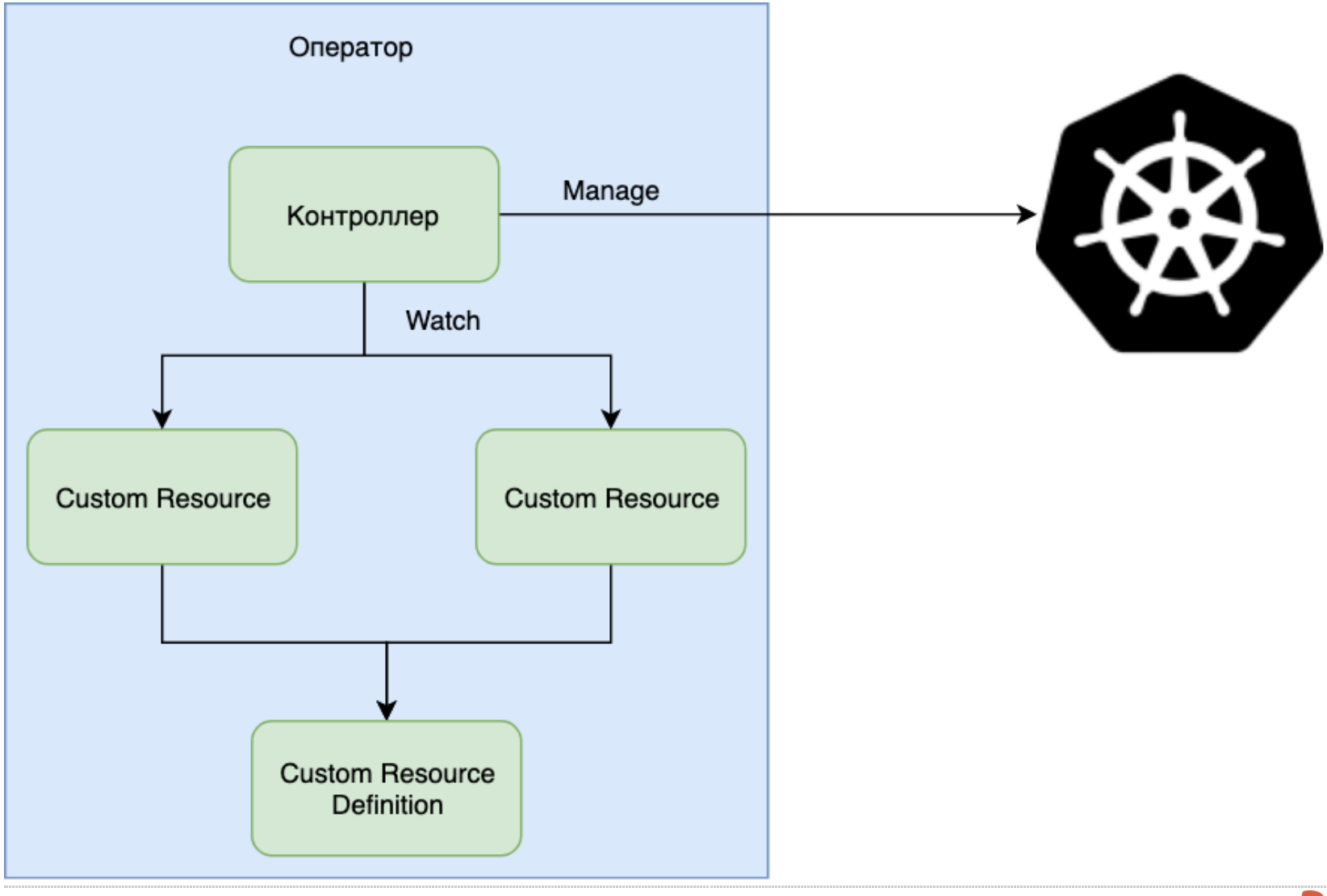
# Зачем использовать операторы?

- Встроенные абстракции kubernetes позволяют хорошо работать с:
  - stateless приложениями
  - приложениями, состоящими из набора одинаковых компонент
  - с некоторыми statefull приложениями, например, consul (с использованием persistent volumes)
- Их функций недостаточно:
  - Когда поды имеют различные роли
  - Когда нужны дополнительные шаги, для того, чтобы масштабировать контейнеры
  - Нужно добавить спецефичную логику работы существующим объектам

# Примеры задач операторов

- Операторы позволяют превратить операционное знание в процесс
- Деплой приложения по требованию
- Обработка изменений объектов рядом с приложением (изменения конфигурации, схемы базы данных)
- Резервное копирование и восстановление состояния приложения
- Деплой, конфигурирование репликации и масштабирование statefull сервисов

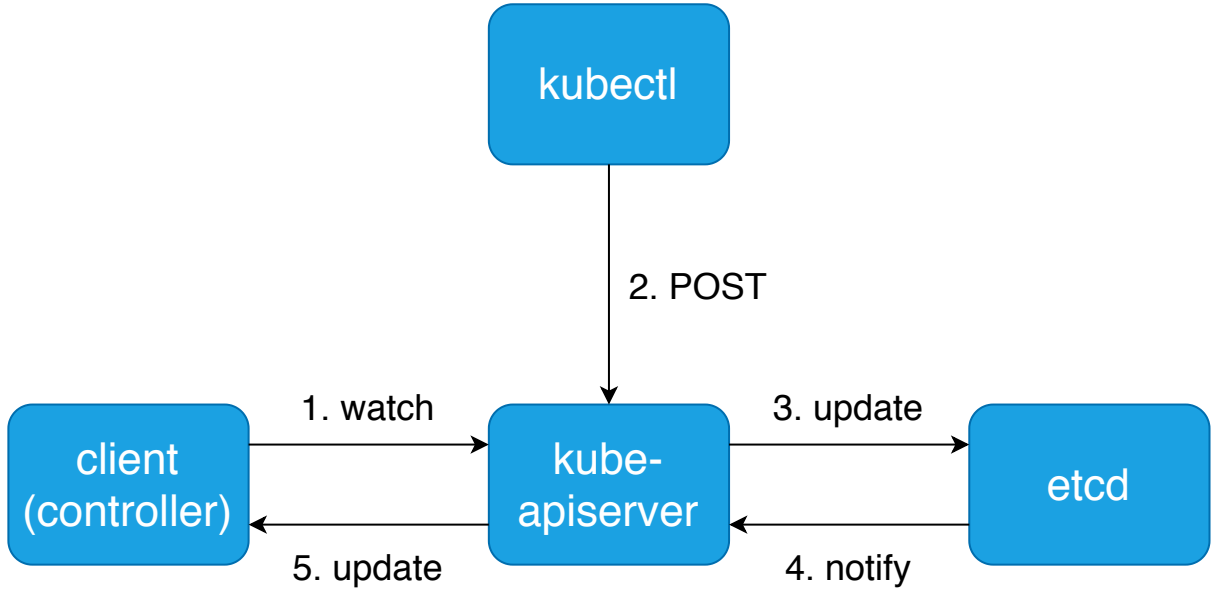
# Компоненты оператора



# Контроллеры

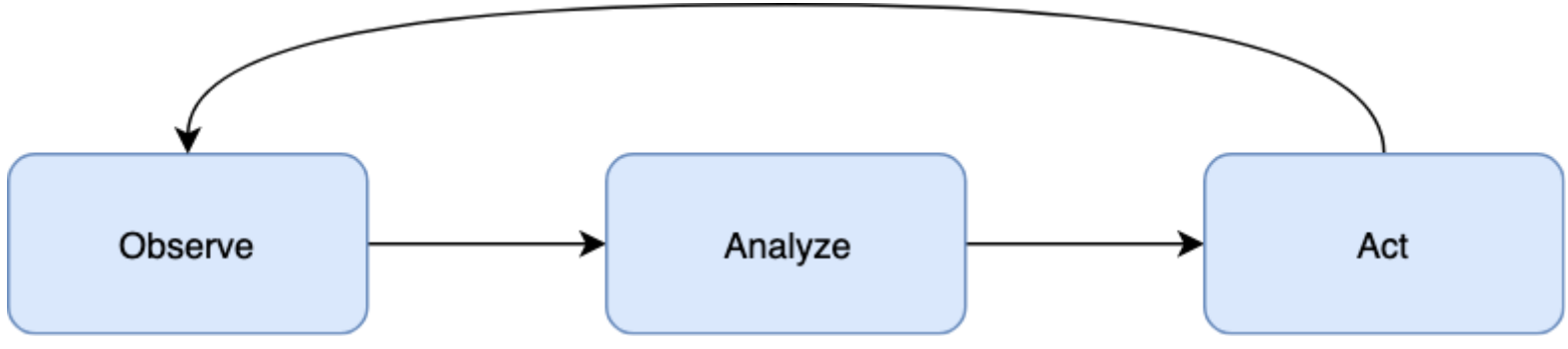
# Контроллер

Контроллеры постоянно следят за объектами kubernetes и поддерживают их в желаемом состоянии.



# Observe Analyze Act

- `Observe` - наблюдение за актуальным состоянием кластера
- `Analyze` - определение различий с желаемым состоянием
- `Act` - выполнение набора действий для достижения желаемого состояния



# Примеры контроллеров

- fabric8/configmapcontroller

```
metadata:
  annotations:
    configmap.fabric8.io/update-on-change: "foo"
```

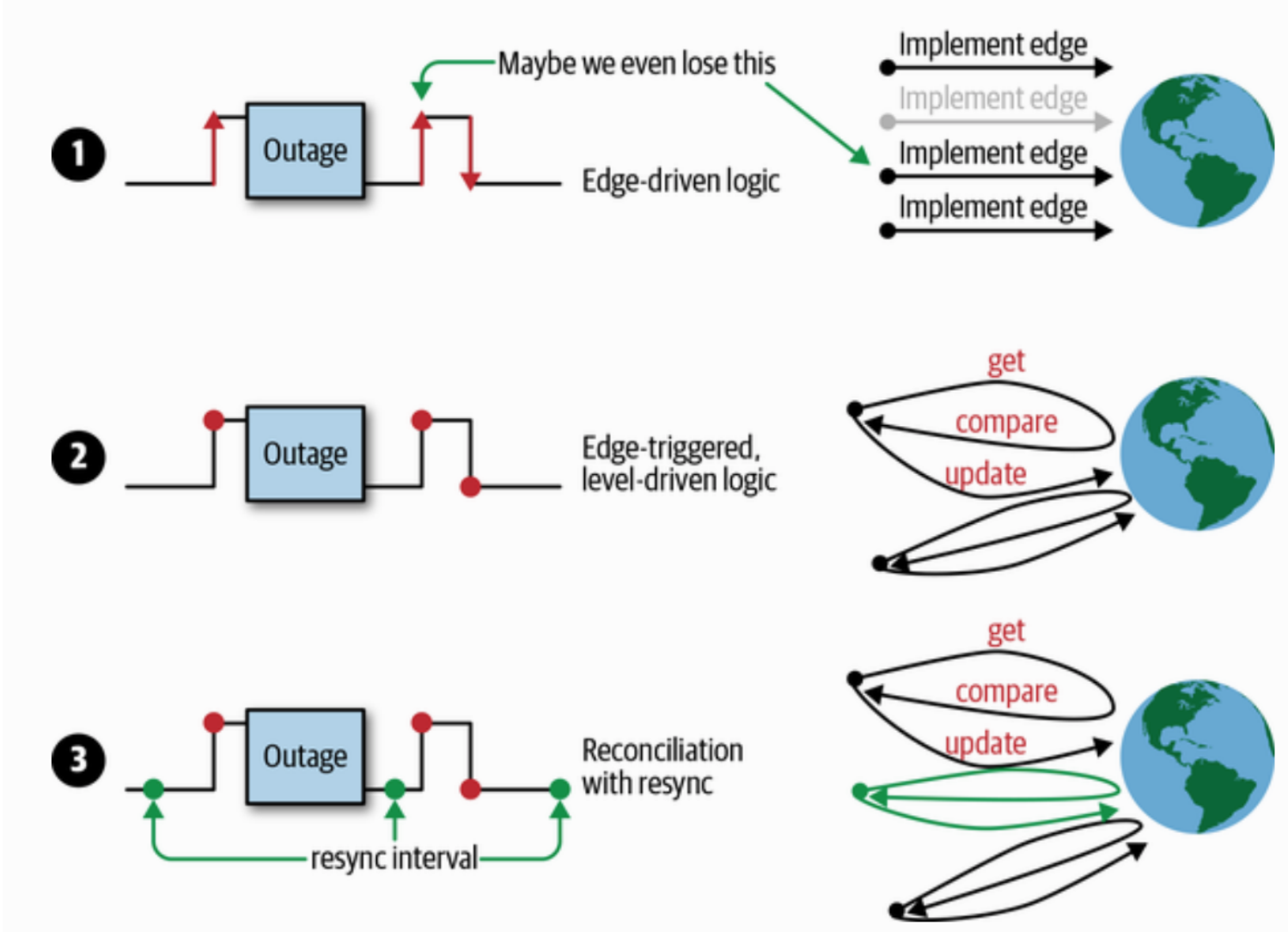
- jenkins-x/exposecontroller

> Подробнее про jenkins-x/exposecontroller

# Где хранить информацию, используемую контроллером

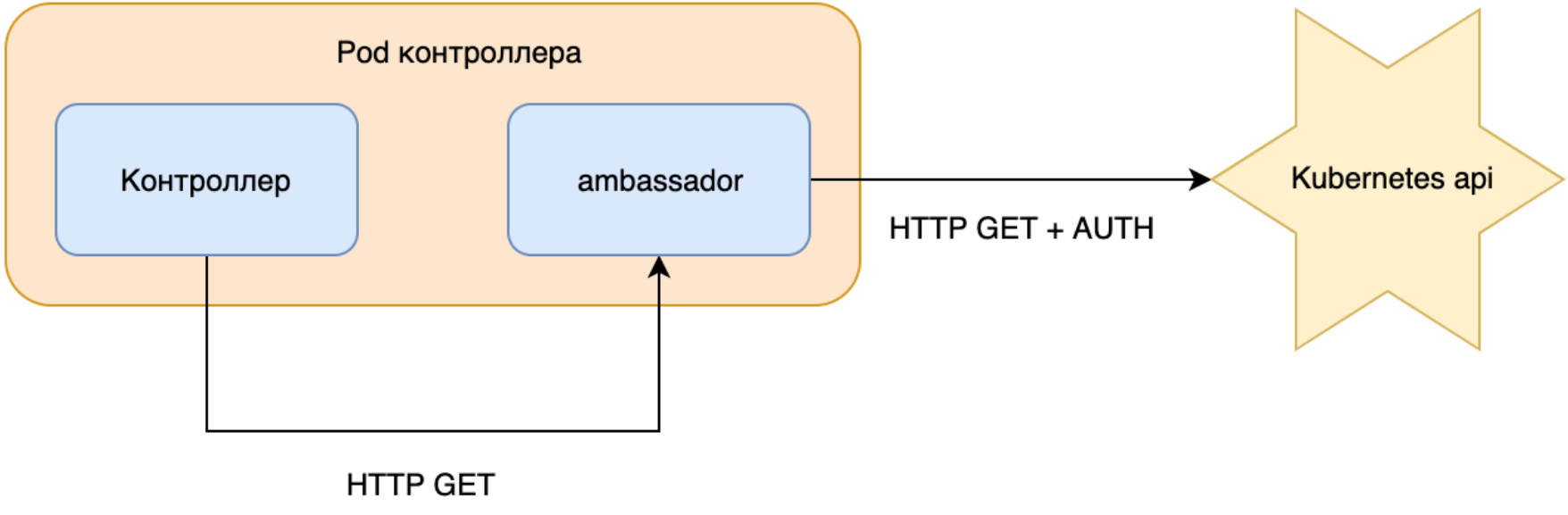
- Labels
- Annotation
- ConfigMaps
- X

# Триггеры в контроллерах



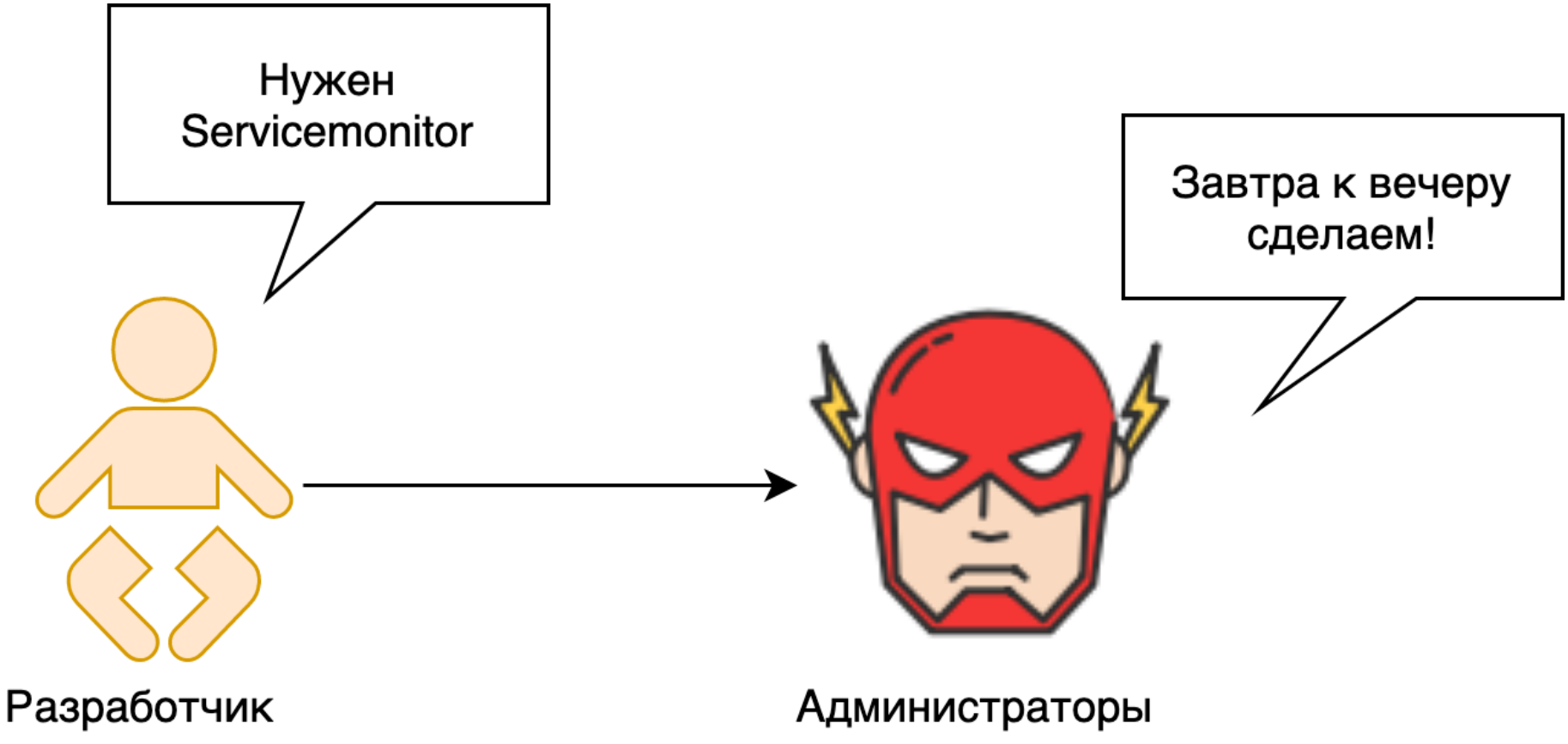
# Как запускать контроллер

- Обычно контроллер запускается внутри кластера при помощи deployment с размером 1, но можно вынести наружу
- Для него также скорее всего понадобится ServiceAccount, Role, RoleBinding

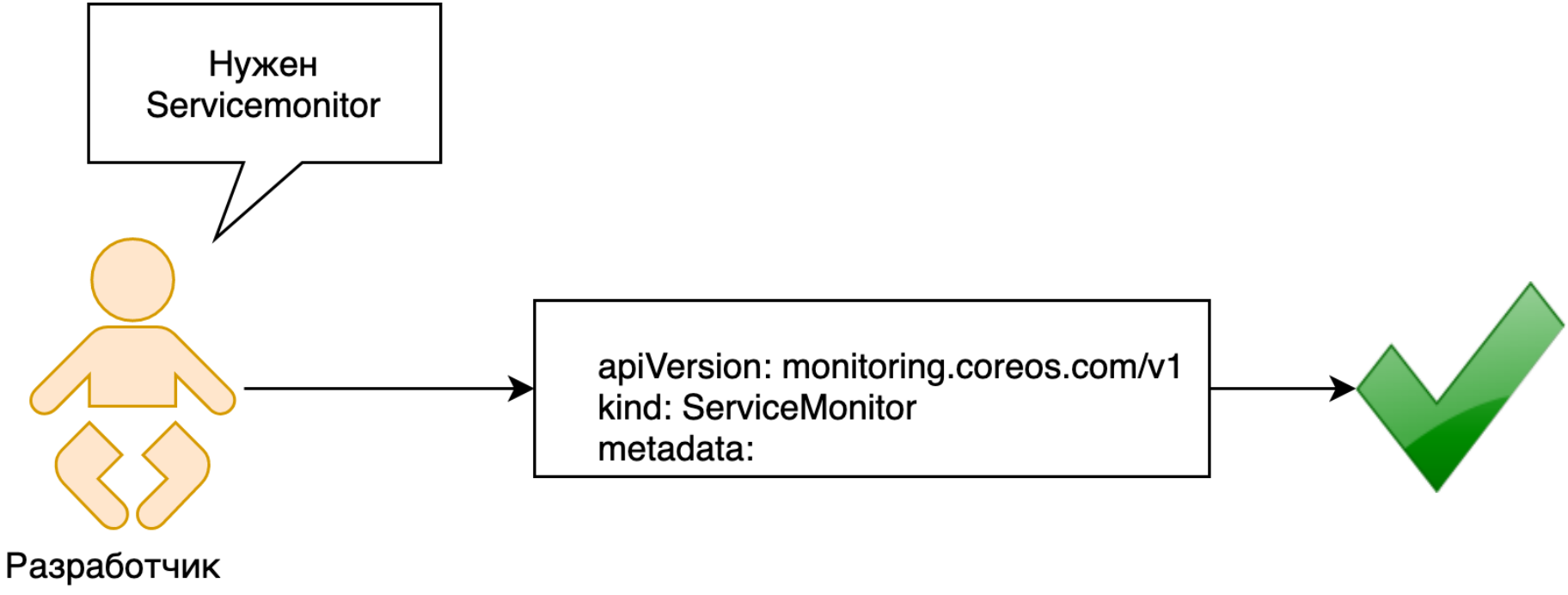


# Чего-то не хватает...

# Как бывает


Нужен Servicemonitor
Завтра к вечеру сделаем!
Разработчик
Администраторы

# Как нужно


Нужен
Servicemonitor
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
Разработчик

# Custom resources

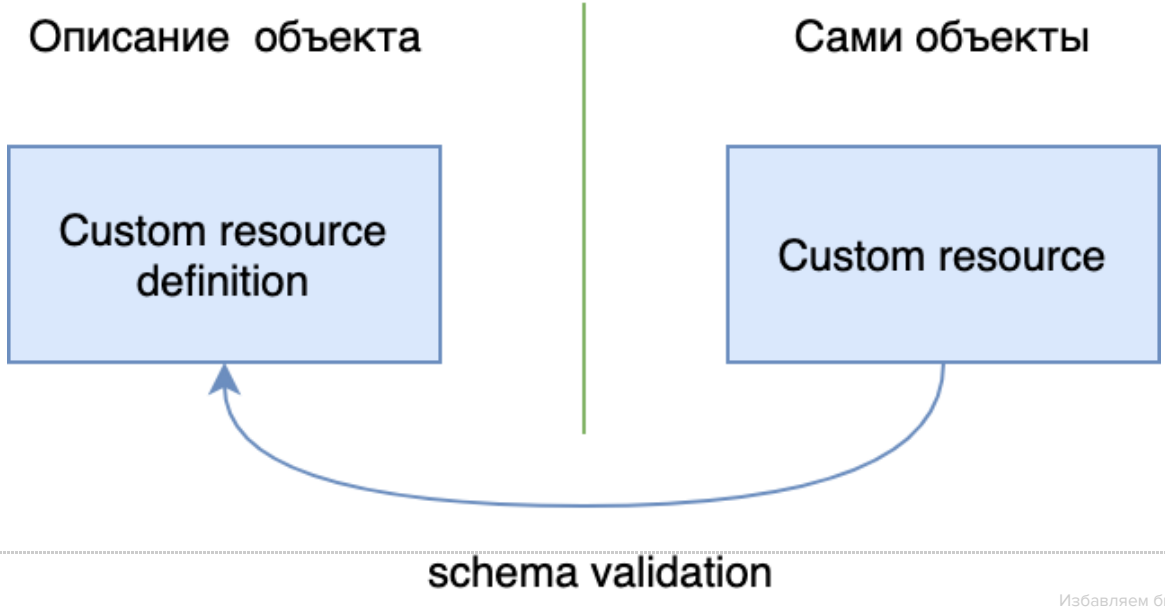
- ***Resource*** - `endpoint` в kubernetes API, который хранит набор объектов определенного типа (kind)
- ***Custom resource*** - `endpoint` Kubernetes API, которого нет в установке kubernetes по умолчанию
- Использование Custom resource:
  - Предоставляет для пользователя абстракцию, для решения каких-либо задач
  - Позволяет разработчику платформы стандартизировать запросы на добавление компонентов
  - Делает работу с контроллерами значительно удобнее

# CustomResource пример:

```yaml
apiVersion: enterprises.upmc.com/v1
kind: ElasticsearchCluster
metadata:
  name: es
spec:
  kibana:
    image: example/kibana-oss:6.1.3
    image-pull-policy: Always
  cerebro:
    image: example/cerebro:0.7.2
    image-pull-policy: Always
  elastic-search-image: example/elasticsearch-kubernetes:6.1.3_0
  image-pull-policy: Always
  client-node-replicas: 1
  master-node-replicas: 1
  data-node-replicas: 2
  network-host: 0.0.0.0
  use-ssl: false
  data-volume-size: 10Gi
  java-options: "-Xms512m -Xmx512m"
```

# Custom resource definition

- ***CustomResourceDefinition*** - ресурс для определения CustomResource
- Создание CustomResourceDefinition создает RESTful путь для каждой описанной версии CustomResource
- Определяет структуру и доступные версии для конкретного CustomResource



# CRD | API, kind , metadata

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: reddits.zhenkins.express42
spec:
  group: zhenkins.express42
  version: v1
```

Имя CRD должно иметь форат plural.group.

Параметры plural и group определяются в spec CRD

# Custom resource definition | names

```
kubectl get rds          # short name
kubectl get reddit       # singular name
kubectl get reddits      # plural name
```

```
  names:
     plural: reddits
     singular: reddit
     kind: Reddit
     shortNames:
      - rds
```

Как это выглядит в CustomResource:

```
apiVersion: "zhenkins.express42/v1"
kind: Reddit       # spec.names.kind из CRD
```

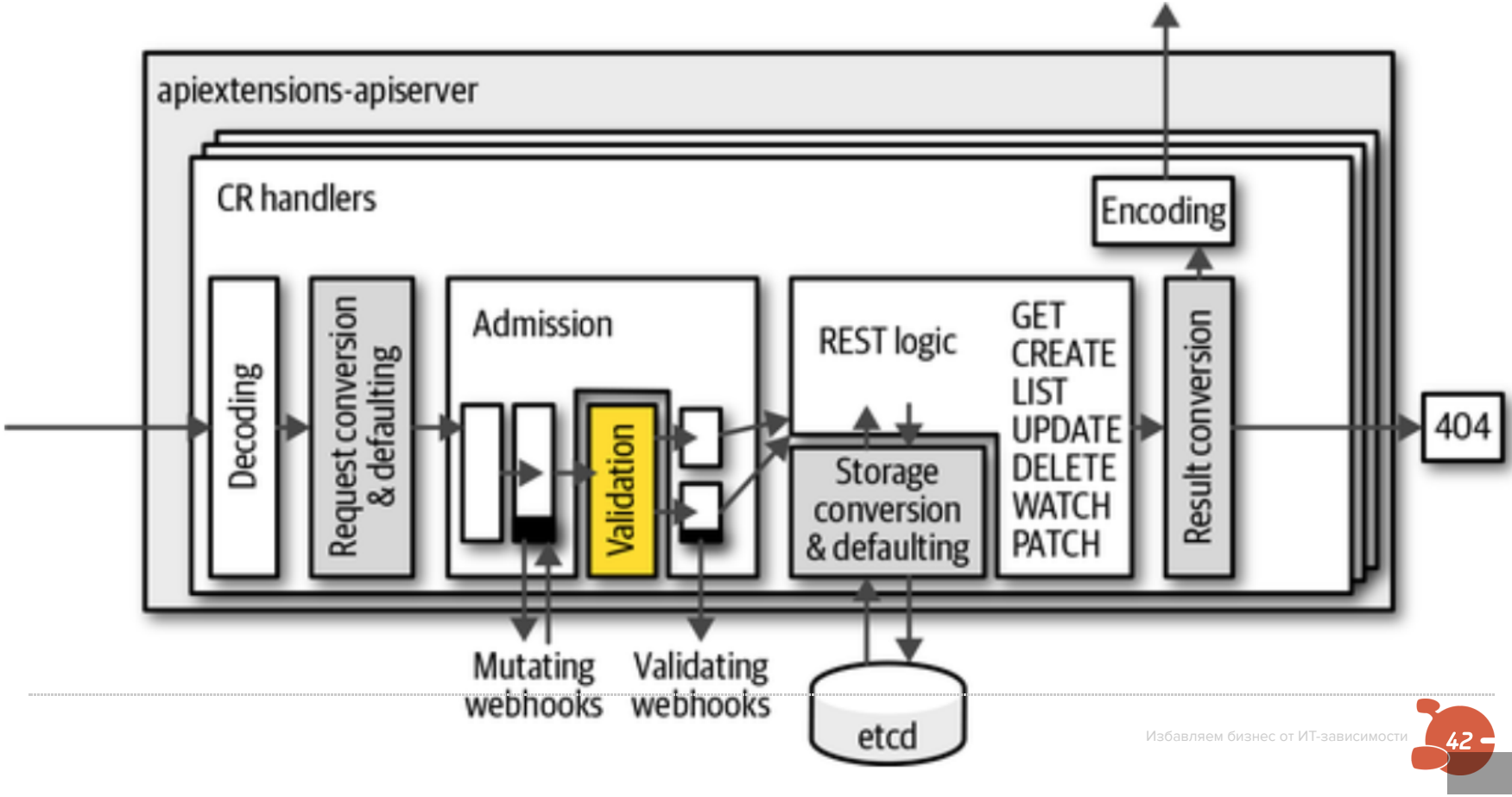
# Custom resource definition | Сейчас это уже будет работать:

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: reddits.zhenkins.express42
spec:
  group: zhenkins.express42
  names:
     plural: reddits
     singular: reddit
     kind: Reddit
     shortNames:
      - rds
  versions: # Список версий
    - name: v1
      served: true
      storage: true
  scope: Cluster
```

```yaml
apiVersion: "zhenkins.express42/v1"
kind: Reddit
metadata:
  name: test1-reddit2
  namespace: default
spec:
  postReplicas: 3
  uiReplicas: vosem
  commentReplicas: 1
```

# Custom resource definition | openAPIV3Schema, validation

Можно описывать структуру Custom Resources, использовать webhook



# CRD | validation

```yaml
validation:
  openAPIV3Schema:
    type: object
    properties:
      apiVersion:
        type: string
      kind:
        type: string
      metadata:
        type: object
      spec:
        properties:
          postReplicas:
            type: integer
        type: object
        required:
          - postReplicas
      required:
        - metadata
        - apiVersion
        - spec
```

# Custom resource definition | webhook

- `Mutating webhook` - Позволяют модифицировать объекты (Custom Resources)
- `Validating webhook` - Позволяют подвергунть объекты дополнительной проверке

> В Kubernetes `1.15` в alpha появилась возможность использовать default. До этой версии, mutating webhook.

> Рекомендация: не заменять поля объектов используя mutating webhooks

# Custom resource definition | Subresources

Pod:

```
* /api/v1/namespace/namespace/pods/name/logs
* /api/v1/namespace/namespace/pods/name/portforward
* /api/v1/namespace/namespace/pods/name/exec
* /api/v1/namespace/namespace/pods/name/status
```

CR:

```
* scale
* status
```

Пользователь не должен писать в `status` объекта

# Примеры scale, status в CRD

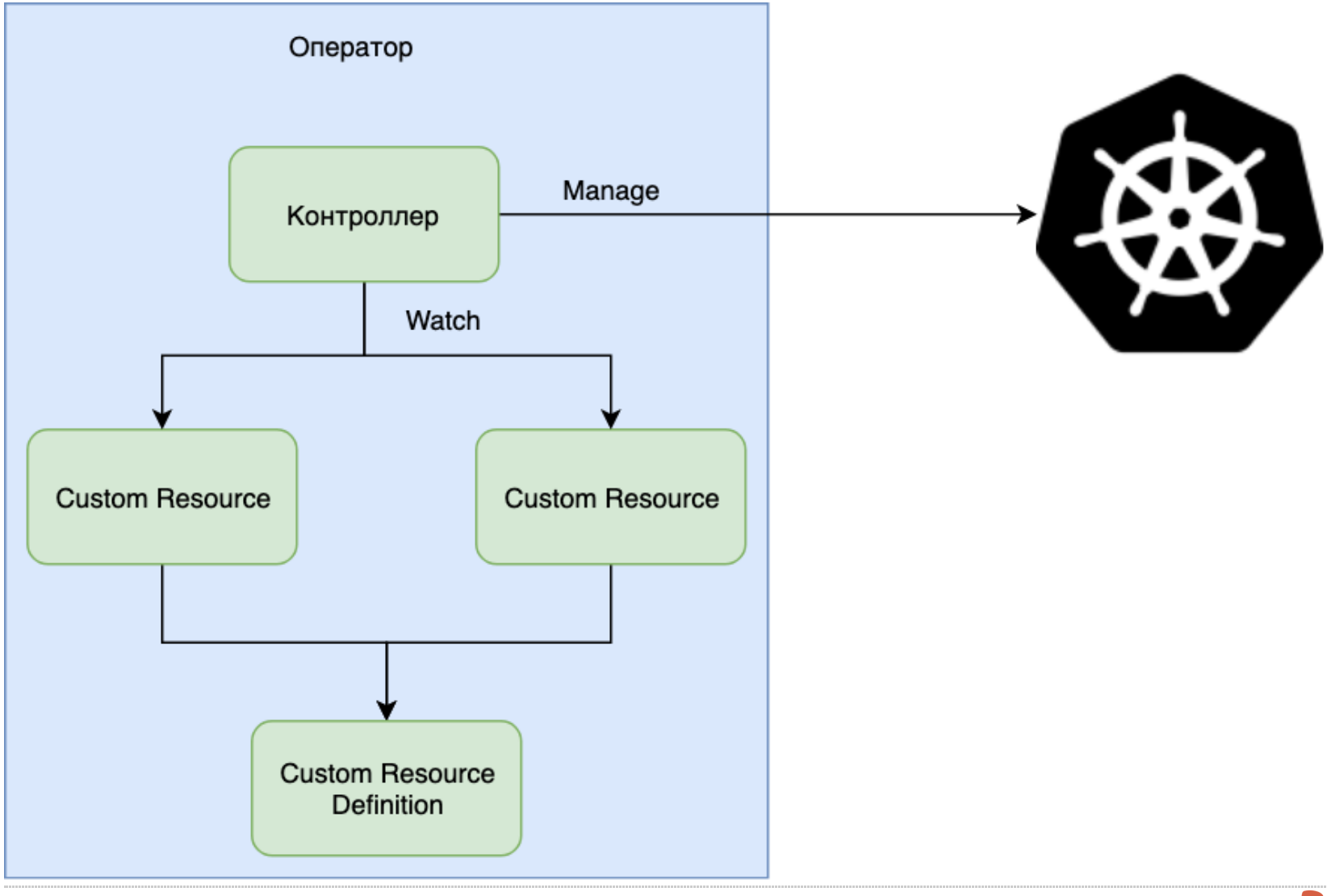
```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
spec:
  subresources:
    scale:
      specReplicasPath: .spec.replicas
      statusReplicasPath: .status.replicas
      labelSelectorPath: .status.labelSelecto
```

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
spec:
  versions:
    - name: v1
      served: true
      storage: true
    - name: v2
      served: true
      subresources:
        status: {}
```

# Operator summary

- Операторы позволяют превратить операционное знание в процесс
- Оператор использует CustomResourceDefinition и custom controller
- Для определения CustomResource в API используется CustomResourceDefinition
    - CRD могут создаваться контроллером
- Контроллеры подписаны на обновления custom resources и реагируют на них

# Компоненты оператора



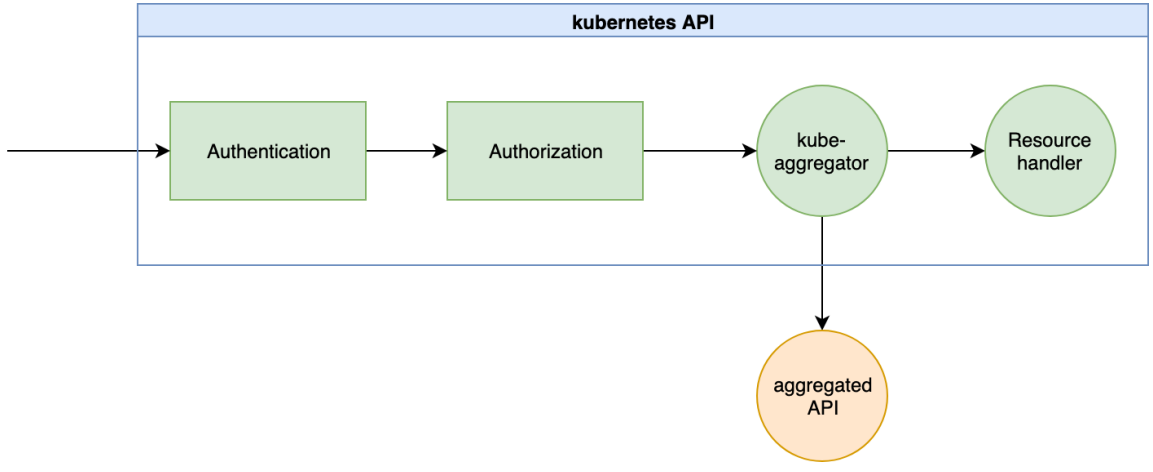
# Operator и практики

- Автоматическая сборка контроллера
- Тестирование контроллера:
  - Локальное в minikube/kind
  - В рамках процесса CI/CD
- Использовать Helm для деплоя и распространения контроллеров
  - Следует приложить Role, RoleBinding...
- Логирование и мониторинг

# Какие моменты стоит учесть при написании своего контроллера

- Периодические запросы к API с целью сверить состояние объекта
- Использовать отдельные операторы для разных приложений
- Использование декларативного API
- Используйте SDK
- Реализовывать различную функциональность отдельными котроллерами
- ~~Списывать у встроенных контроллеров k8s~~

# Custom Api & Api aggregation




- Больше `subresource`
- Не только JSON
- Валидация без использования webhook
- graceful deletion
- Более гибкое управление объектами
- Есть возможность хранить объекты вне etcd kubernetes

# API aggregation

```yaml
apiVersion: apiregistration.k8s.io/v1
kind: APIService
...
spec:
  ...
  group: my.extension.example.com
  versions: v1
  service:
    namespace: my-service-namespace
    name: my-service-name
    port: 9292
  caBundle: "Ci0tLS0tQk...<base64-encoded PEM bundle>...tLS0K"
...
```

Подробнее про API-aggregation

# Operator SDK (ansible)

- Управление k8s осуществляется при помощи ansible
- Создает заготовки:
  - Dockerfile для сборки контроллера и его deployment
  - ServiceAcount, Role, Rolebinding
  - CR, CRD
- Не требует знания языков программирования
- Использует molecule и kind для тестирования

> Документация по оператору

# Operator SDK (ansible)

Watchers.yaml:

```
- version: v1
  group: reddits.zhenkins.express42
  kind: Reddit
  playbook: /opt/ansible/playbooks/reddit-deploy.yml
```

reddit-deploy.yaml:

```
- hosts: localhost
  roles:
    - reddit
```

reddit role tasks:

```
- name: Create reddit namespace
  k8s:
    api_version: v1
    kind: Namespace
    name: "{{ meta.name }}"
```

# Kubernetes Operator Pythonic Framework (Kopf)

- Использует python
- Хорошая документация
- Удобная отладка
- Прост в освоении

Kopf на GitHub

Kopf документация

# Kopf | Как работать?

```python
import kopf

@kopf.on.create('zalando.org', 'v1', 'kopfexamples')
def my_handler(spec, **_):
    pass

@kopf.on.update('zalando.org', 'v1', 'kopfexamples')
def my_handler(spec, old, new, diff, **_):
    pass

@kopf.on.delete('zalando.org', 'v1', 'kopfexamples')
def my_handler(spec, **_):
    pass

@kopf.on.field('zalando.org', 'v1', 'kopfexamples', field='spec.somefield')
def somefield_changed(old, new, **_):
    pass
```

# Kopf

Для отладки можно запускать код, не выполняя сборку:

```
kopf run  /kopf-kubernetes.py
```

Dockerfile:

```
FROM python:3.7
COPY kopf-kubernetes.py /kopf-kubernetes.py
RUN pip install kopf kubernetes pyyaml
CMD kopf run  /kopf-kubernetes.py
```

# А что еще?

- KUDO
- Go, Helm operator SDK
- Metacontroller
- Kubebuilder (книжка)

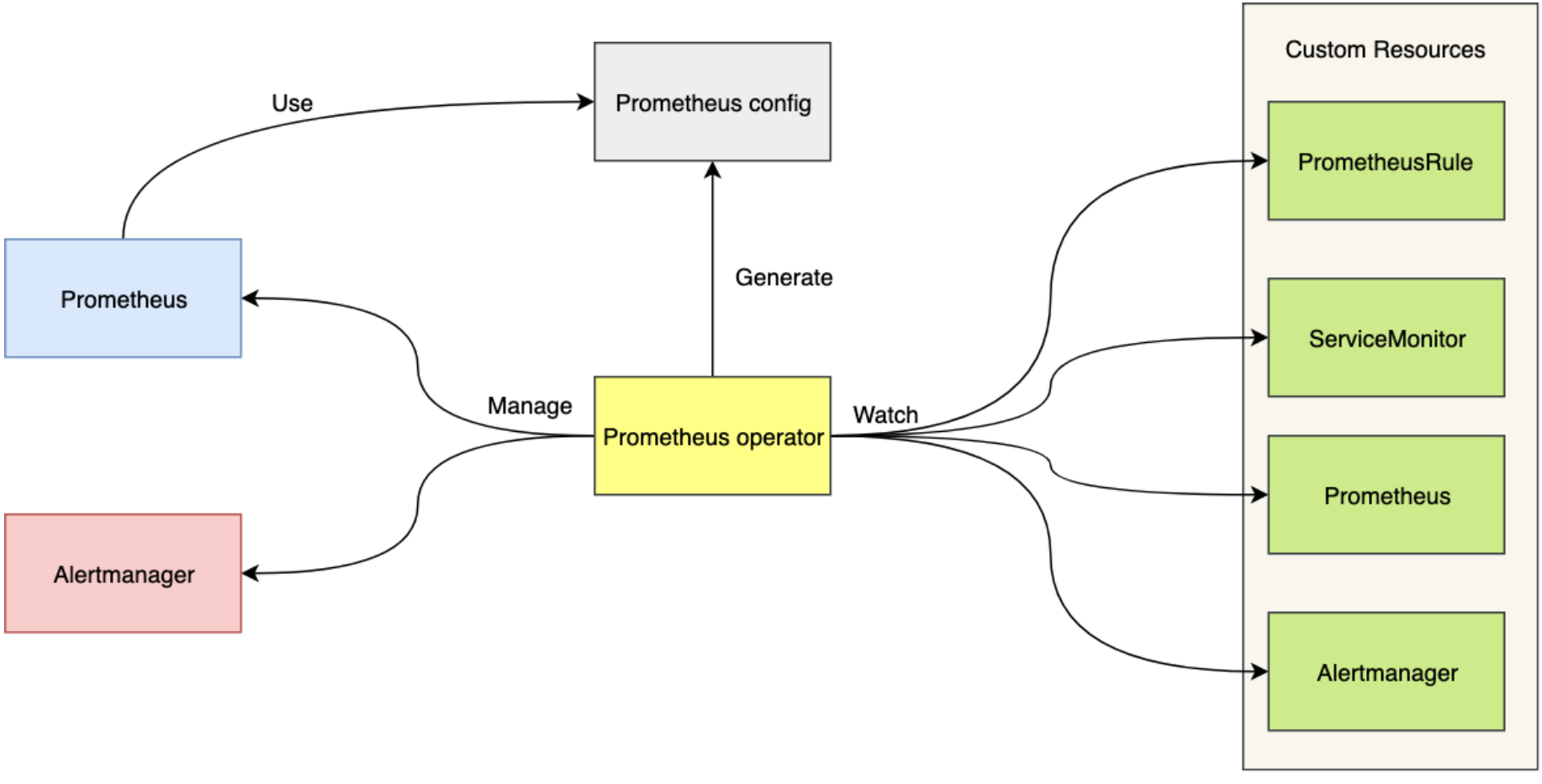
# Описание популярных операторов

# Prometheus-operator

- Пожалуй, самый популярный оператор
- Позволяет легко разворачивать и конфигурировать prometheus
- Использование CRD для управления конфигурацией
- Можно развернуть сразу весь необходимый стек
- Обнаружение хостов по меткам (`labels`).
- Kube-prometheus и community helm чарт


GitHub

# Prometheus-operator что внутри?



# Config-connector

- Аддон для kubernetes, содержащий набор контроллеров и CRD для управления ресурсами в GCP
- Можно определять и настраивать зависимости между ресурсами
- Удобно использовать для взаимодействия с облаком в процессе деплоя

Документация

Список доступных ресурсов

# Спасибо за внимание!