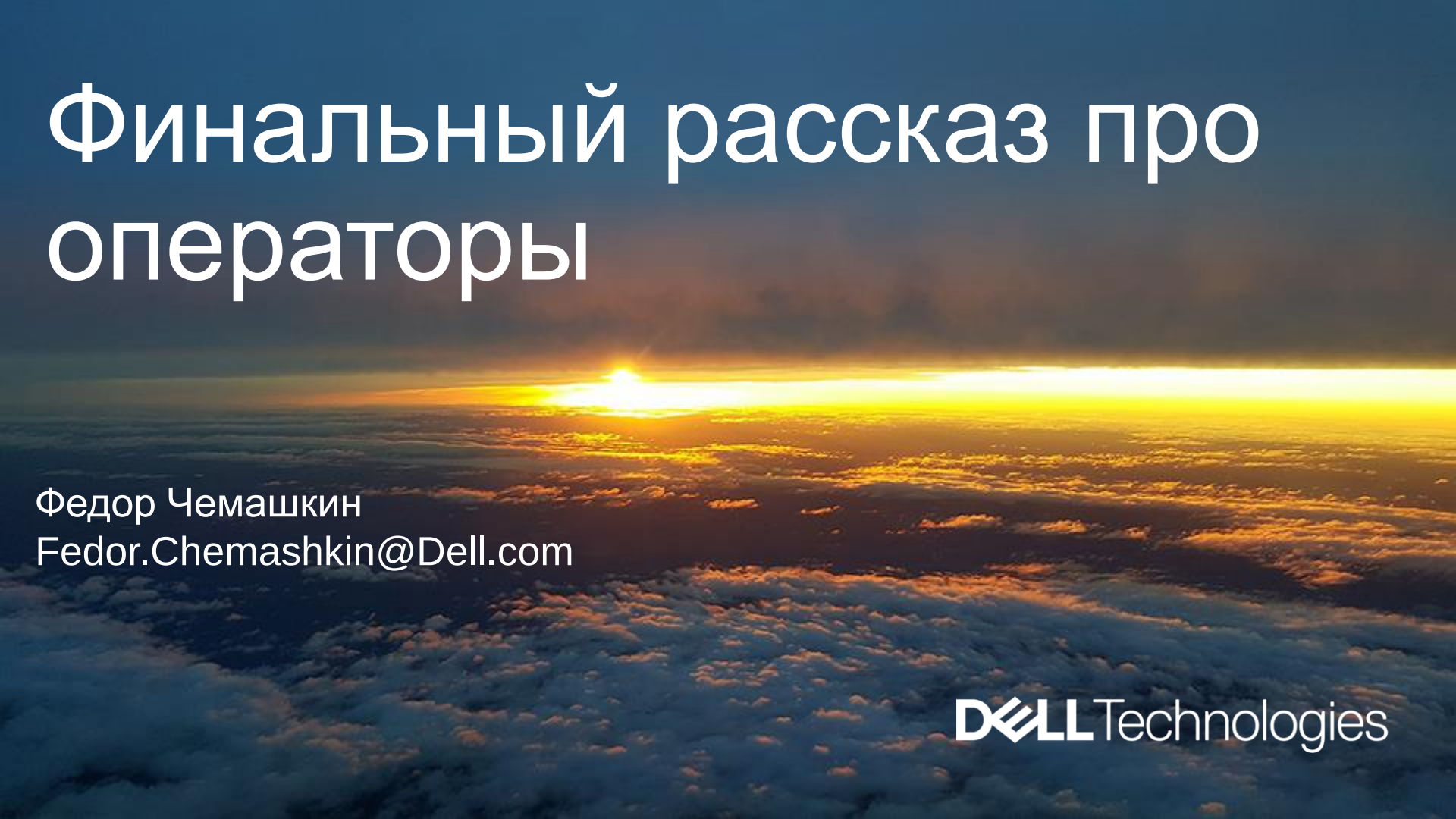
Финальный рассказ про операторы

Федор Чемашкин
Fedor.Chemashkin@Dell.com



О чем поговорим

- Системах управления
- Операторах
- Эксплуатации в Cloud Native мире

Disclaimer – Not Rocket Science

Обо мне

- Senior Engineer
- Но еще студент
- Dell EMC
- Elastic Cloud Storage

Система управления

Система управления

Эксплуатация меняется

?

А как быть с Kubernetes?

- Как нам устанавливать приложение?
- Что делать с эксплуатацией?
- CI/CD?
- Управление инфраструктурой?
- …

Поставлять. ПО. В. Чужой. Kubernetes.

DELL Technologies

Поставлять. ПО.
В. Чужой.
Kubernetes.

А как быть с ***чужим*** Kubernetes?

- Как нам устанавливать приложение?
- Что делать с эксплуатацией?
- CI/CD?
- Управление инфраструктурой?
- …

Helm

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ template
"app.name" . }}-operator
```

Python

```
k8s_beta =
client.ExtensionsV1beta1Api()
resp =
k8s_beta.create_namespaced_de
ployment(body=dep,
namespace="default")
```

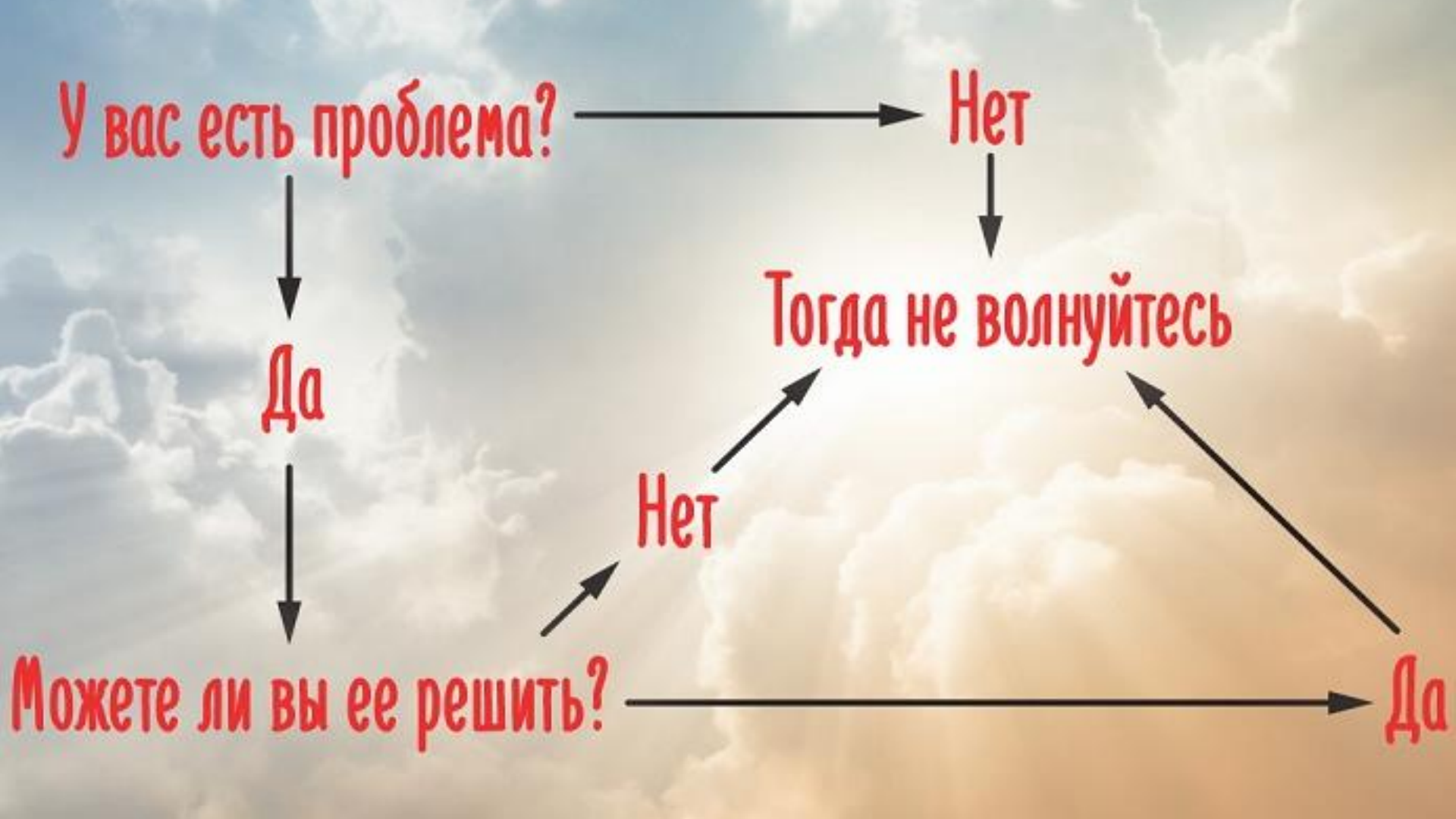
Ansible

```
- name: Create namespace
  kubernetes:
    api_endpoint: …
    url_username: …
    url_password: …
    file_reference: …
```

Envbust

```
metadata:
 name:          app
 namespace:   ${NAMESPACE}

spec:
  replicas: ${REPLICAS}
```


YAML’и кастомеру - ужасно и стыдно!


У вас есть проблема?
Нет
Да
Тогда не волнуйтесь
Нет
Можете ли вы ее решить?
Да

Все в код!

Все в код!

OPERATOR

Операторы. Начало

- *«Introducing Operators: Putting Operational Knowledge into Software»*

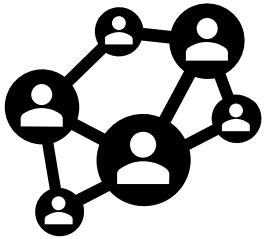
Pravega

- *«Storage Reimagined for a Streaming World»*
- Open Source
- Stream – примитив для хранения/обслуживания непрерывных данных

Pravega

Use cases

IoT

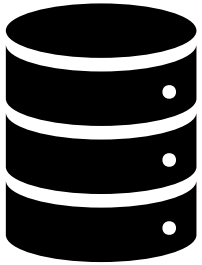


Messaging

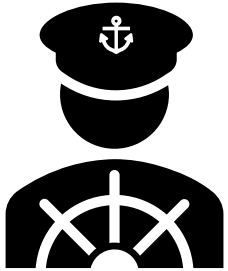
API

Processing

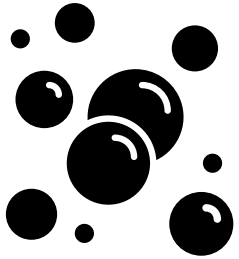
Pravega в k8s – в чем сложность?



Хранение данных



Специфичная эксплуатация

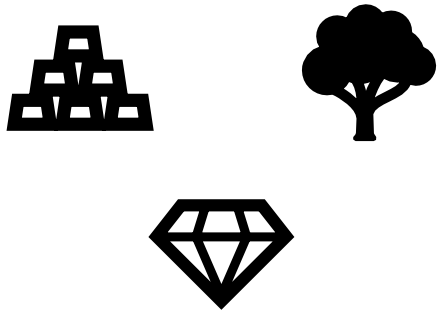


Распределенная система

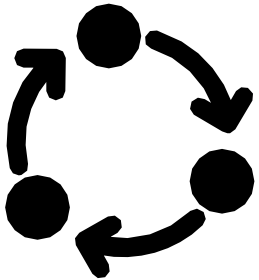
Pravega Operator

- Кастомизированная работа с PV и PVC
- Установка и удаление Pravega
- «Правильный» Rolling upgrade
- Проверка состояния кластера
- …

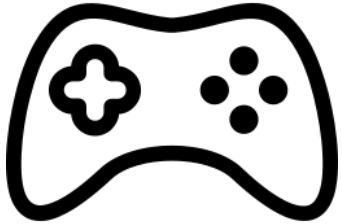
Kubernetes в трех картинках



Resources

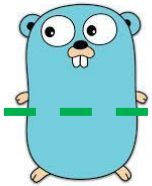


Event Stream



Controllers

controller-manager



ReplicaSet

Deployment

CronJob

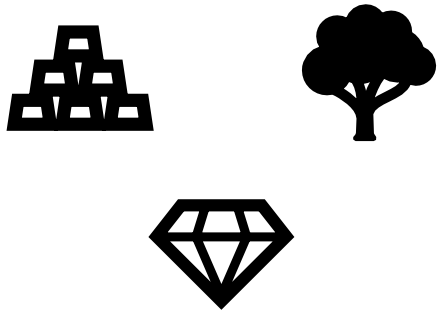
DaemonSet

StatefulSet

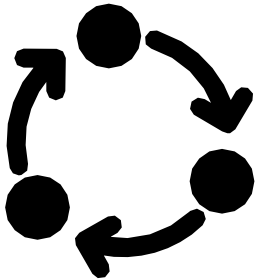
Job

...

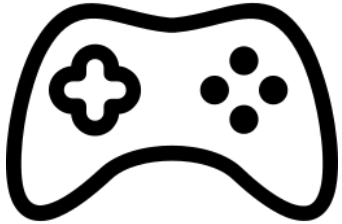
Kubernetes в трех картинках



Resources

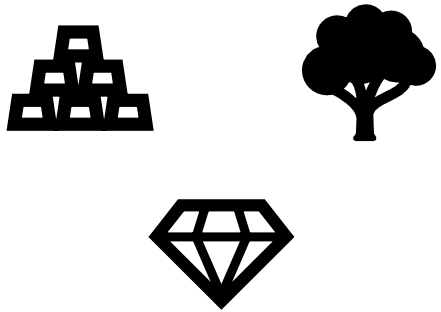


Event Stream

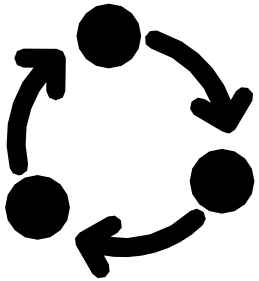


Controllers

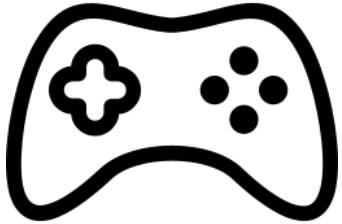
Operator в трех картинках



Custom
Resource



Event Stream



Custom
Controller

Operator vs. Custom Controller

“We call this new class of software Operators. ***An Operator is an application-specific controller that extends the Kubernetes API*** *to create, configure, and manage instances of complex stateful applications on behalf of a Kubernetes user…”*

Operator vs. Custom Controller

Operator

- Controller Pattern
- Содержат специфичные знания
- Управляют lifecycle

Custom Controller

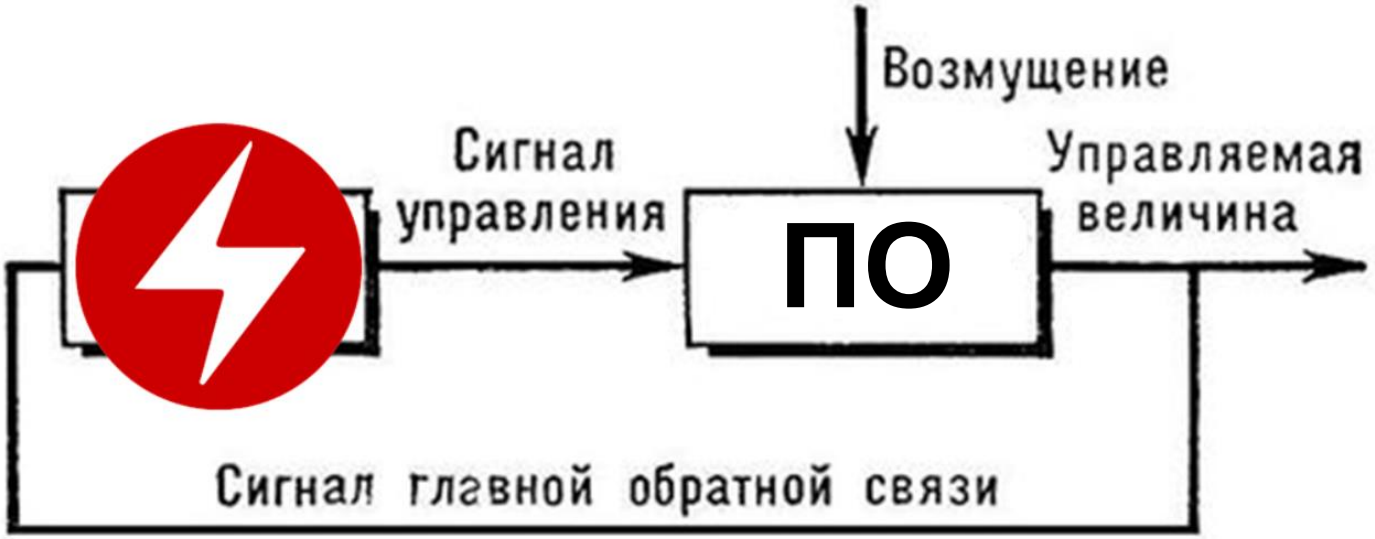
- Controller Pattern
- Создание нового «простого» ресурса

Kube Builder

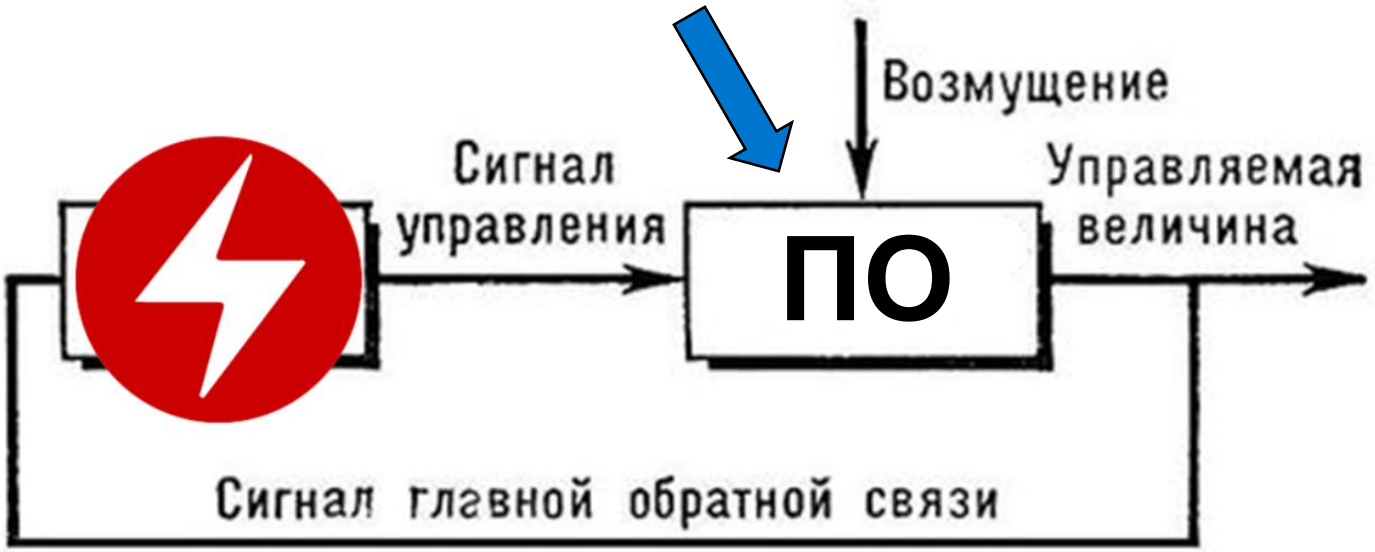
Оператор и ПО

ПО

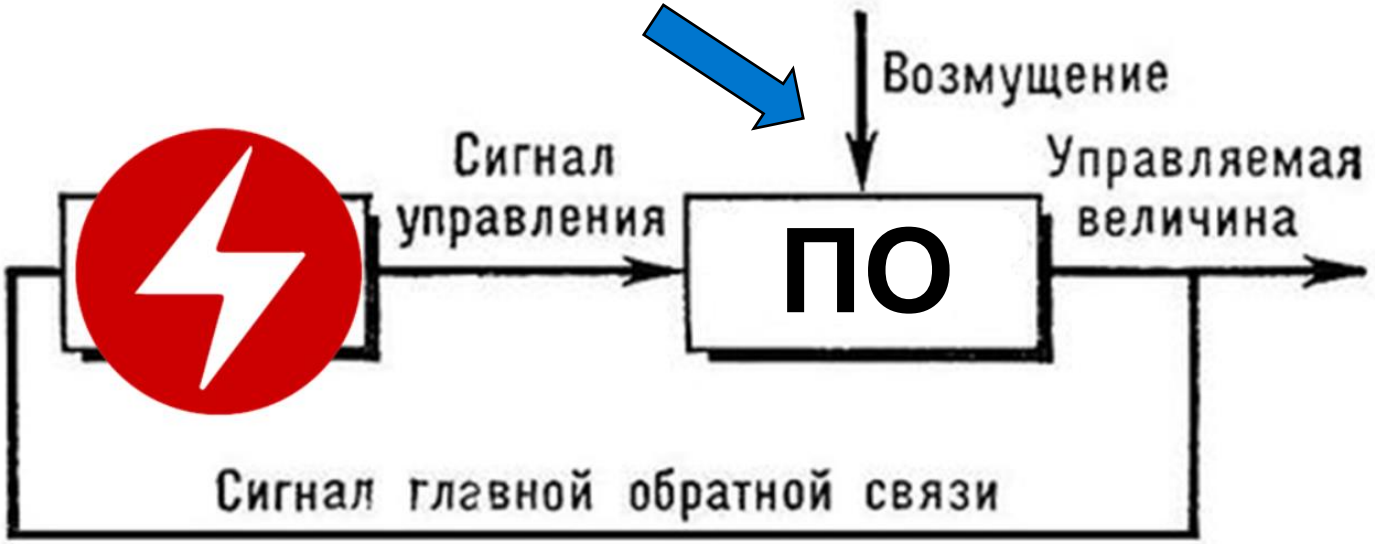
Оператор – система управления



Оператор – система управления

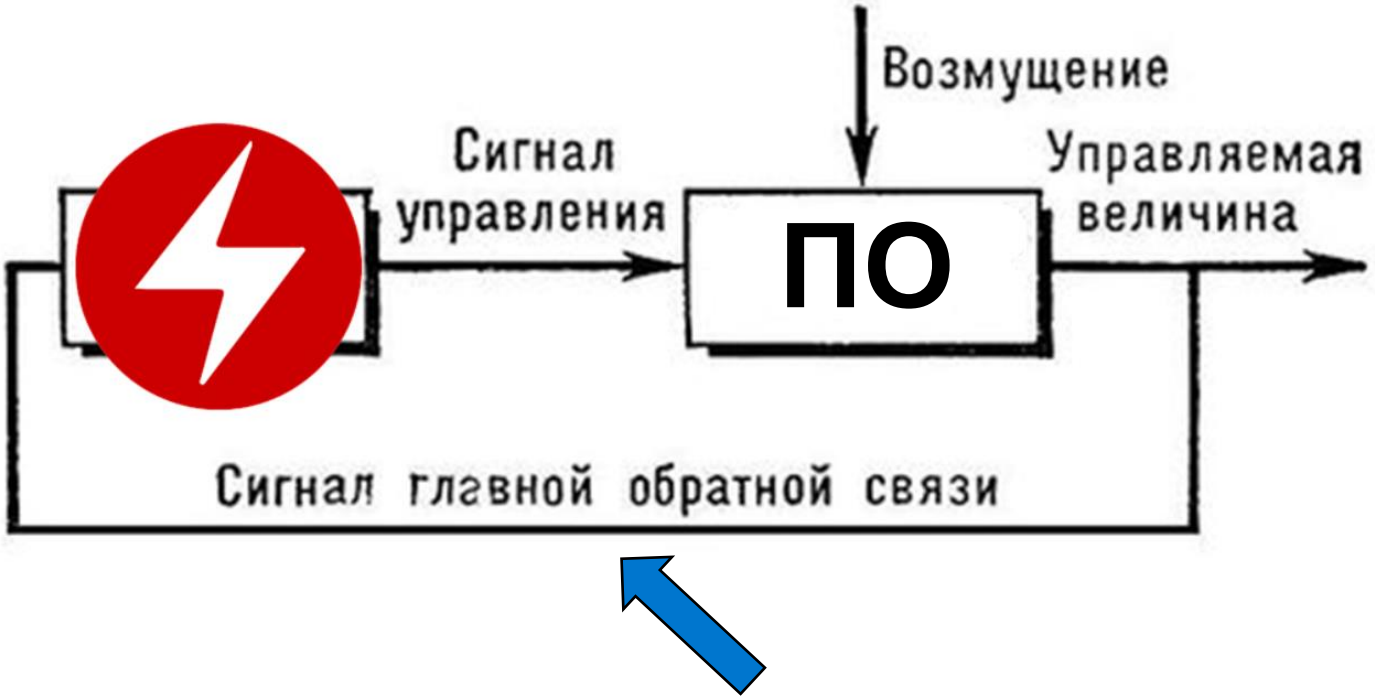


Оператор – система управления

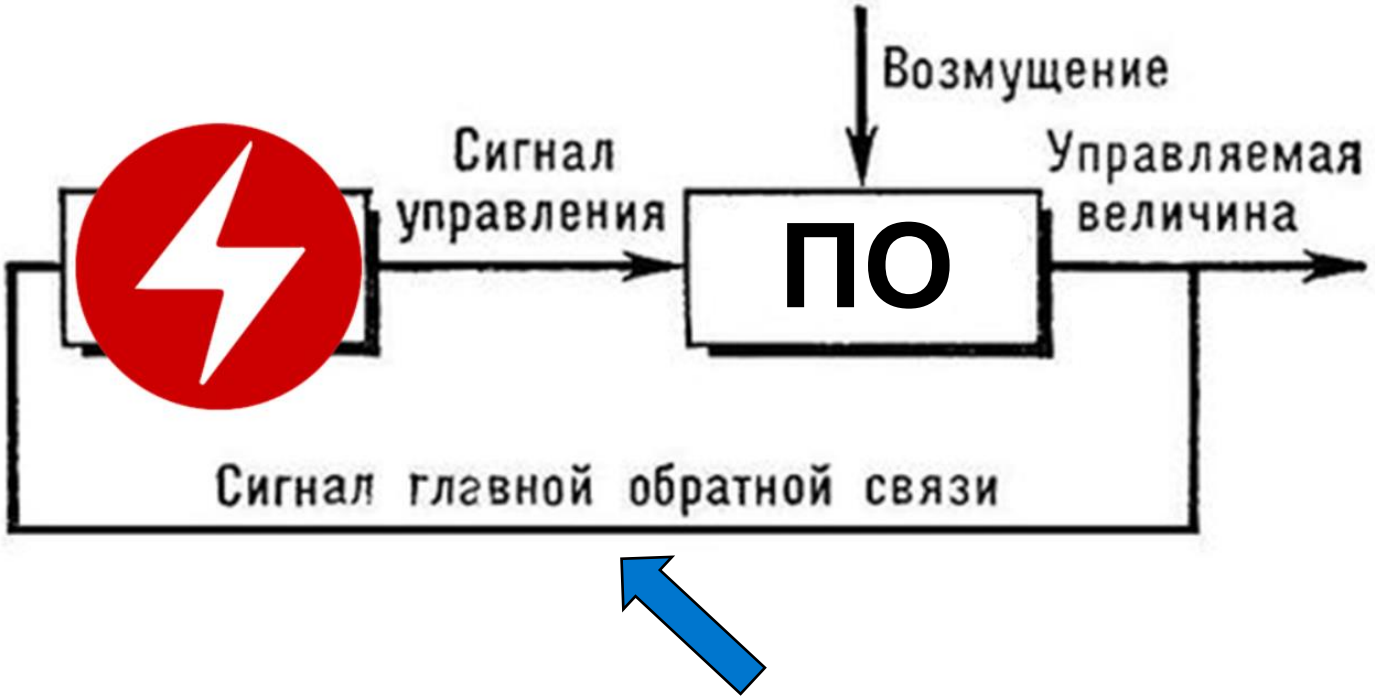


Оператор – система управления

Оператор – система управления

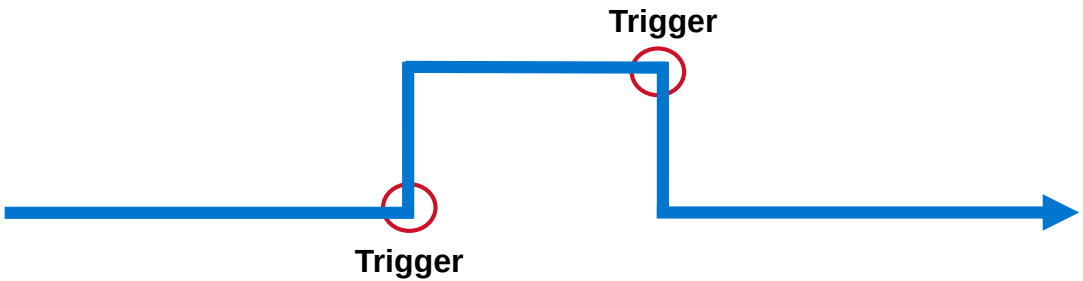


Оператор – система управления

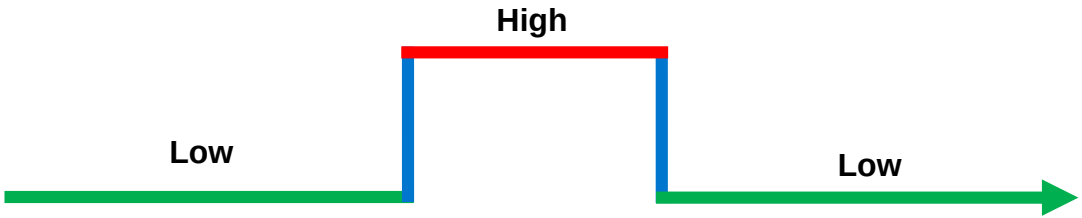


Edge triggering vs. Level triggering

Edge triggering

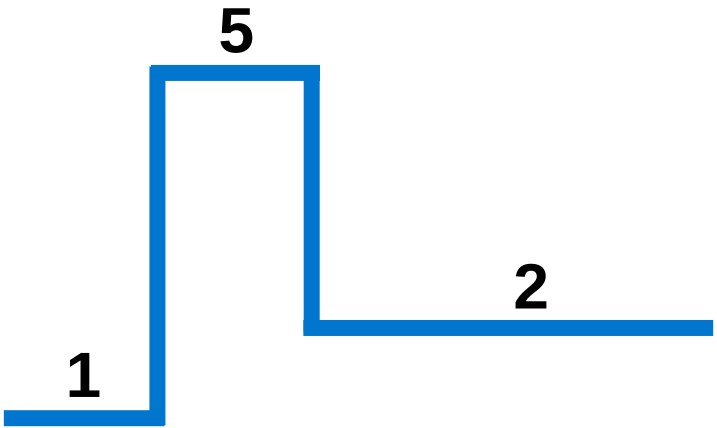


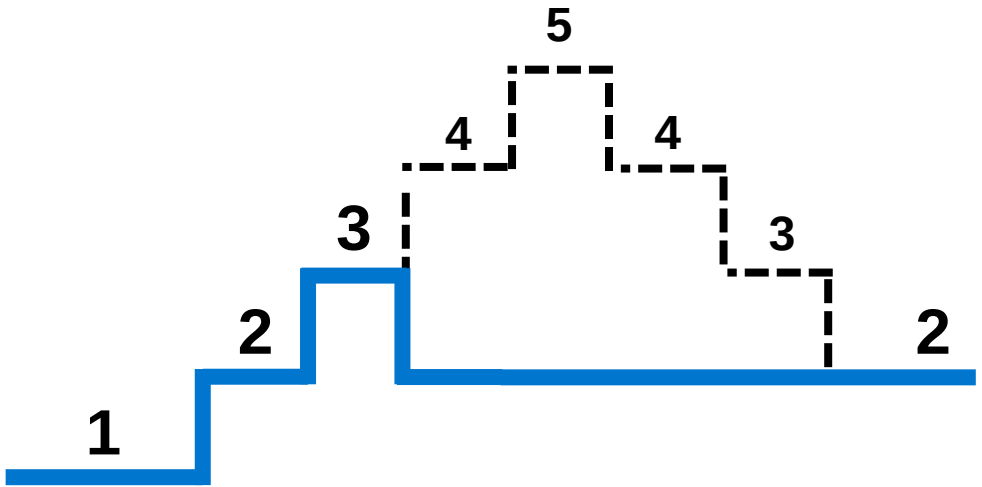
Level triggering



Источник: [Level Triggering and Reconciliation in Kubernetes]

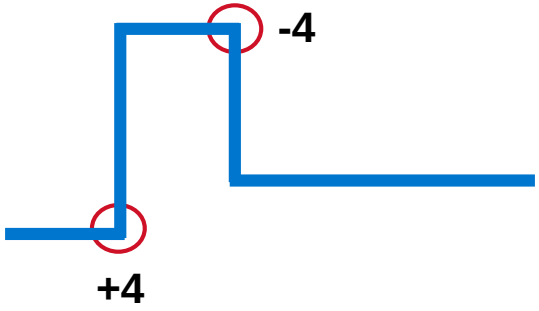
Пример

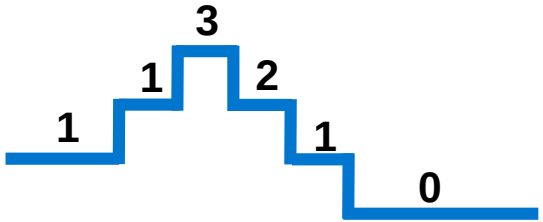




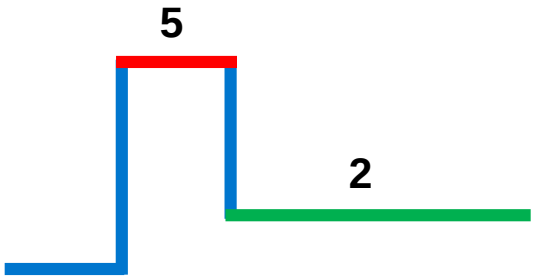
Реализация в Kubernetes

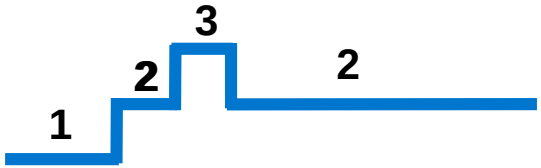
Edge triggering





Level triggering

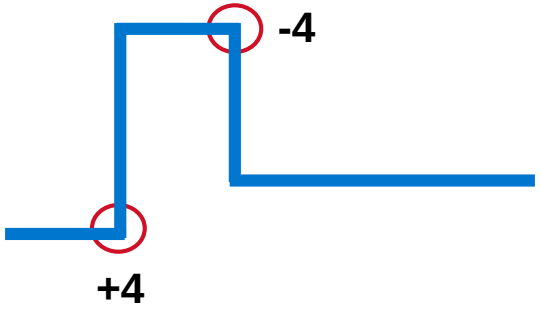


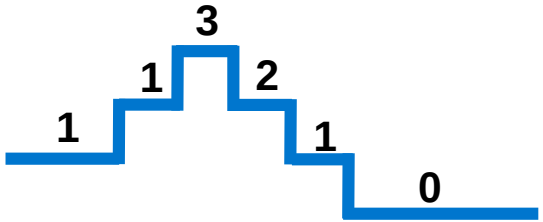


Источник: [Level Triggering and Reconciliation in Kubernetes]

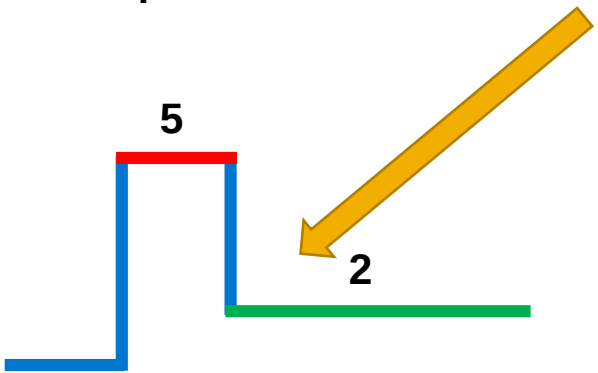
Реализация в Kubernetes

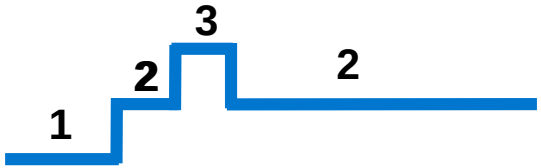
Edge triggering





Level triggering



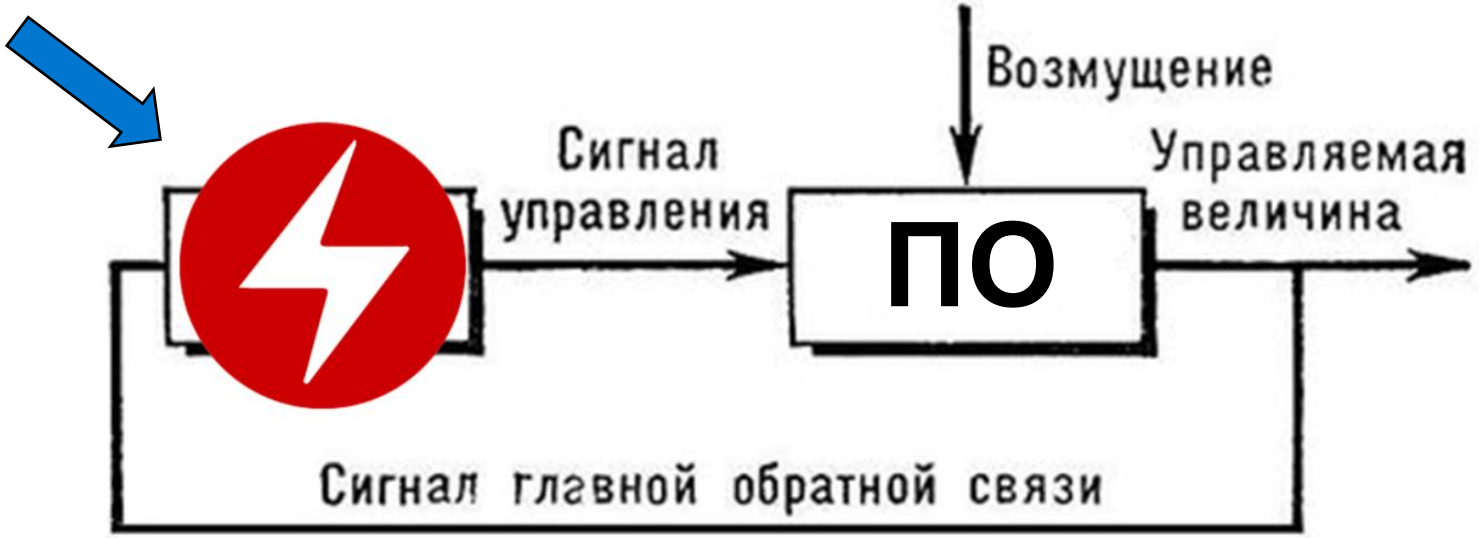


Источник: [Level Triggering and Reconciliation in Kubernetes]

Reconciliation

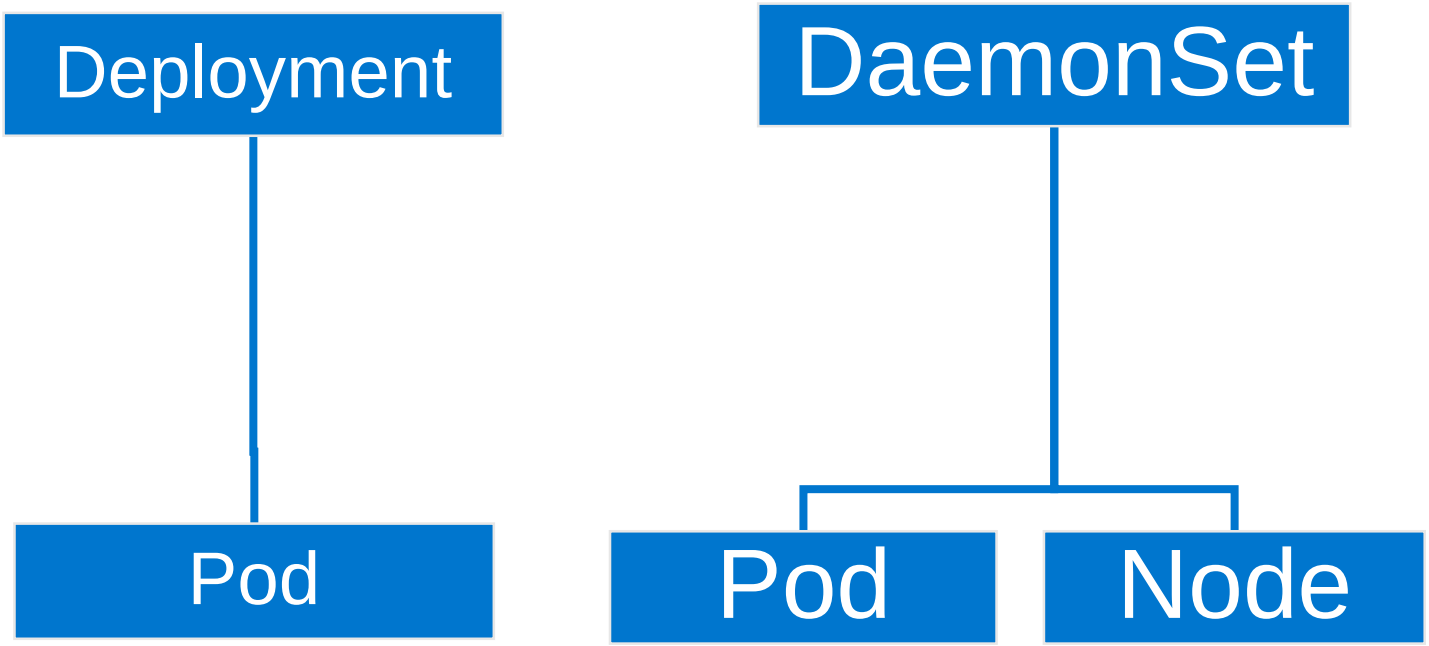
- Каждый ресурс имеет:
  - желаемое состояние
  - фактическое состояние
- Reconciliation Loop:
  - Обнаруживает изменения ресурсов
  - Изменяет фактическое состояние
  - Попытка соответствовать желаемому состоянию
- Разработчик может и ***должен*** управлять этим

Оператор – система управления



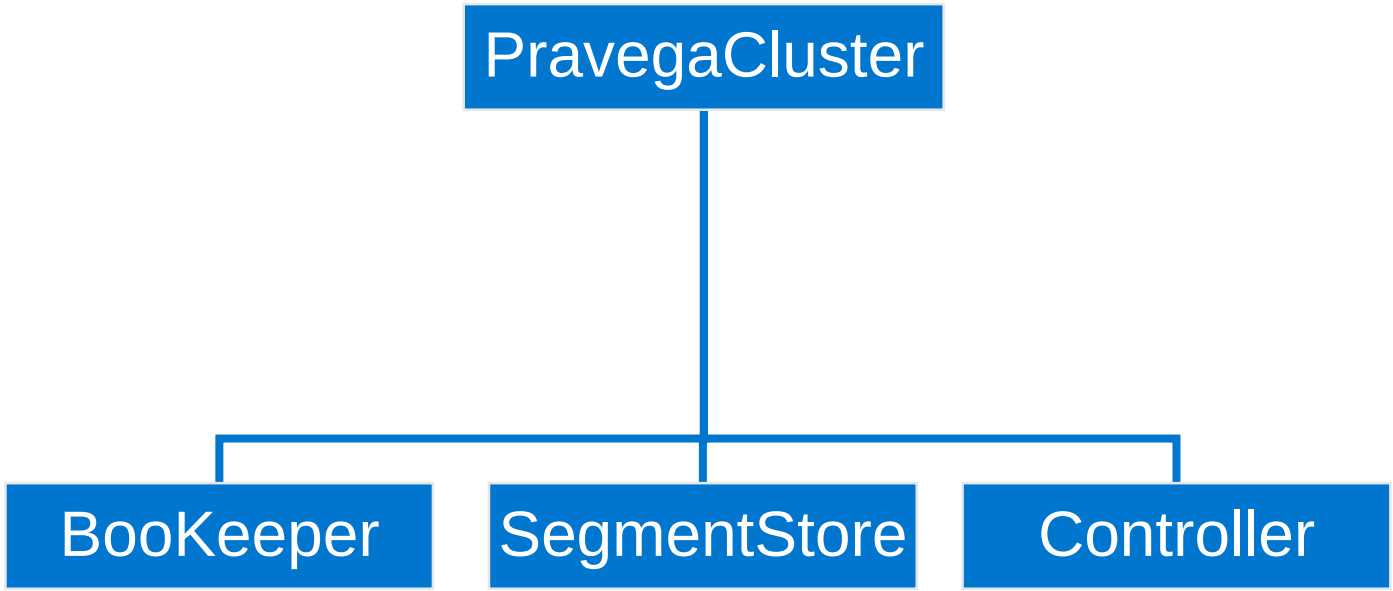
Primary и Secondary ресурсы

Primary:



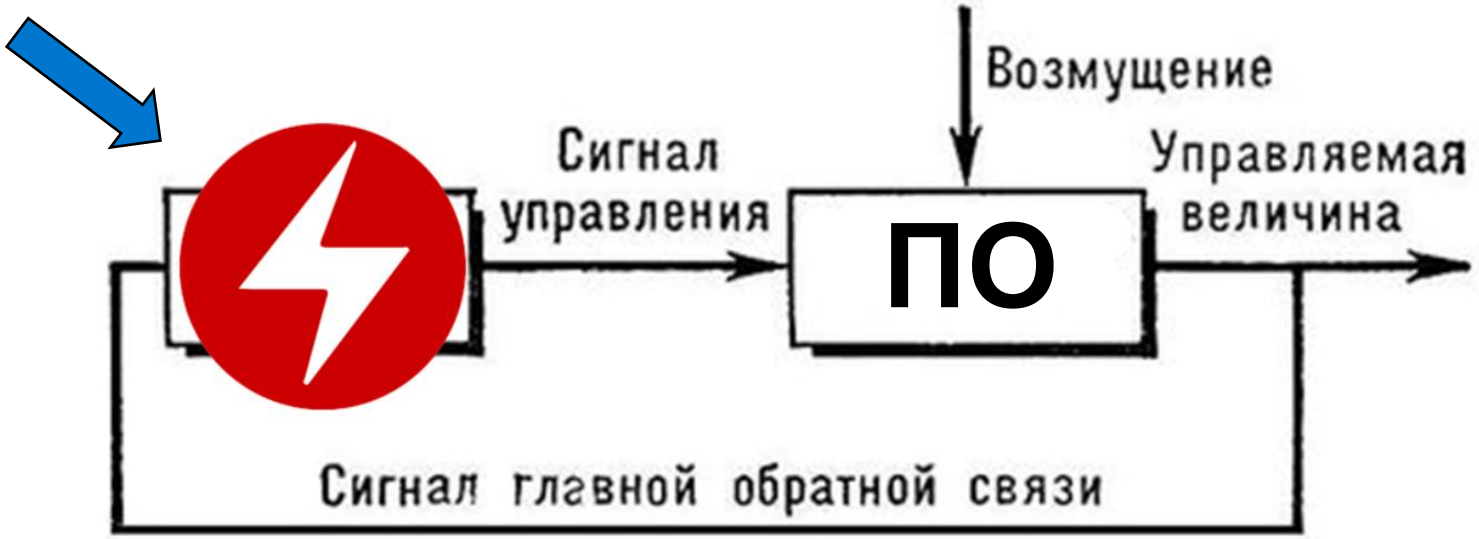
Primary и Secondary ресурсы

Primary:

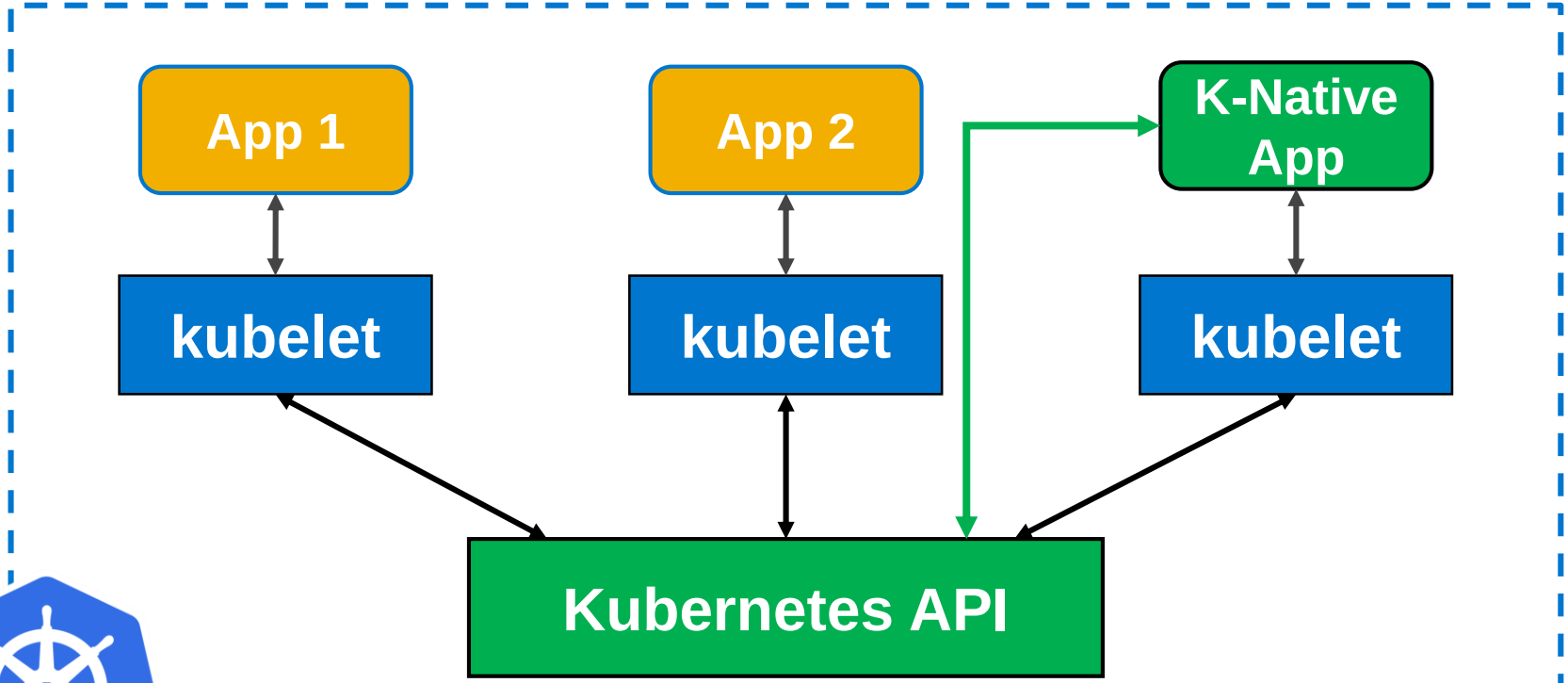


Secondary:

Оператор – система управления



Оператор – пример k8s-native ПО



Создаем оператор

Operator SDK

- CoreOS (Red Hat)
- Очень быстрый старт
- Кодогенерация «половины» оператора
- Часть Operator Framework

Инициализация проекта

```
$ operator-sdk new pravega-operator
```

```
INFO[0000] Creating new Go operator 'pravega-operator'.
INFO[0000] Created go.mod
INFO[0000] Created tools.go
INFO[0000] Created cmd/manager/main.go
INFO[0000] Created build/Dockerfile
…
```

Сгенерированная структура

```
|-- build
|-- cmd
|   `-- manager
|       `-- main.go
|-- deploy
    `-- *.yaml
|-- pkg
|   |-- apis
|   `-- controller
```


Operator
Manager

Manager

pravega-operator/cmd/manager/main.go

- Точка входа
- Инициализация Manager
- Автоматическая регистрация CR - `pkg/apis/*`
- Запуск контроллеров - `pkg/controller/*`

cmd/manager/main.go

```
mgr, err := manager.New(cfg, manager.Options{...})

if err := apis.AddToScheme(mgr.GetScheme()); err != nil

if err := controller.AddToManager(mgr); err != nil
```

Kubernetes API Server

Kubernetes API Server

- Kubernetes API Server – это Rest API Server

Kubernetes API Server

- Kubernetes API Server – это Rest API Server
- CRUD операции над ресурсами
- Custom Resource
- Custom Controller

Терминология API

- Kind – тип сущности
- API Group – коллекции логически связанных Kinds
- Version – версии API Group (`v1alpha1, v1beta1`)

Создание Custom Resource Definition

pravega-operator/pkg/apis/

```
$ operator-sdk add api --api-version=pravega/v1alpha1 \     --kind=Pravega

INFO[0000] Generating api version pravega/v1alpha1 for kind Pravega.
INFO[0000] Created pkg/apis/pravega/v1alpha1/pravega_types.go
INFO[0000] Created pkg/apis/addtoscheme_pravega_v1alpha1.go
INFO[0000] Created pkg/apis/pravega/v1alpha1/register.go
INFO[0000] Created pkg/apis/pravega/v1alpha1/doc.go
…
```

Структура

pravega-operator/pkg/apis/

```
|-- addtoscheme_pravega_v1alpha1.go
|-- apis.go
`-- pravega
    `-- v1alpha1
        |-- doc.go
        |-- pravega_types.go
        `-- register.go
```

Структура

pravega-operator/pkg/apis/

```
|-- addtoscheme_pravega_v1alpha1.go
|-- apis.go
`-- pravega
    `-- v1alpha1
        |-- doc.go
        |-- pravega_types.go
        `-- register.go
```

Структура

pravega-operator/pkg/apis/

```
|-- addtoscheme_pravega_v1alpha1.go
|-- apis.go
`-- pravega
    `-- v1alpha1
        |-- doc.go
        |-- pravega_types.go
        `-- register.go
```

Структура

pravega-operator/pkg/apis/

```
|-- addtoscheme_pravega_v1alpha1.go
|-- apis.go
`-- pravega
    `-- v1alpha1
        |-- doc.go
        |-- pravega_types.go
        `-- register.go
```

Структура

pravega-operator/pkg/apis/

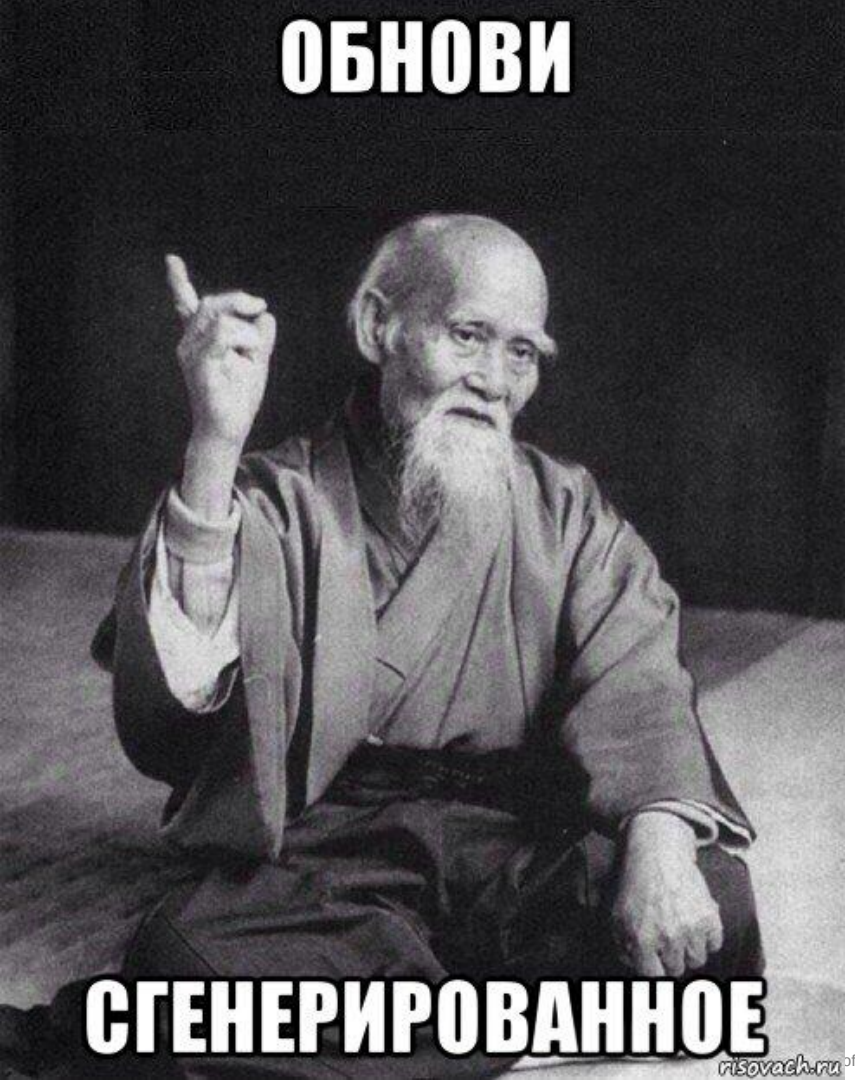
```
|-- addtoscheme_pravega_v1alpha1.go
|-- apis.go
`-- pravega
    `-- v1alpha1
        |-- doc.go
        |-- pravega_types.go
        `-- register.go
```

pravega_types.go

```
type PravegaCluster struct {
  metav1.TypeMeta         `json:",inline"`
  metav1.ObjectMeta       `json:"metadata,omitempty"`

  Spec    ClusterSpec   `json:"spec,omitempty"`
  Status  ClusterStatus `json:"status,omitempty"`
}
```

pravega_types.go

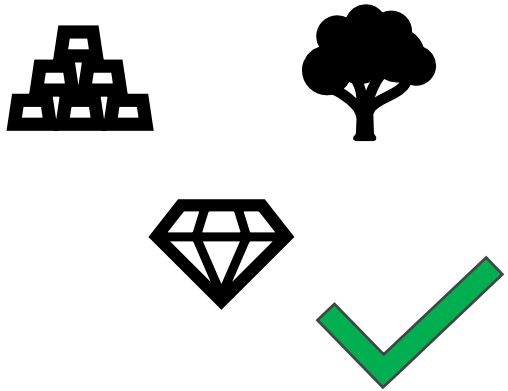
```
type PravegaCluster struct {
  metav1.TypeMeta          `json:",inline"`
  metav1.ObjectMeta        `json:"metadata,omitempty"`

  Spec    ClusterSpec   `json:"spec,omitempty"`
  Status  ClusterStatus `json:"status,omitempty"`
}
```



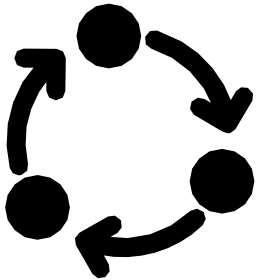

```
$ operator-sdk generate k8s
```


DELL Technologies

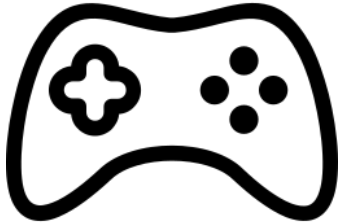
Что получили?



Custom Resource



Event Stream



Custom Controller

Создание контроллера

```
$ operator-sdk add controller --api-version=pravega/v1alpha1 \
--kind=Pravega

INFO[0000] Generating controller version pravega/v1alpha1 for kind
Pravega.
INFO[0000] Created pkg/controller/pravega/pravega_controller.go
INFO[0000] Created pkg/controller/add_pravega.go
...
```

Структура

pravega-operator/pkg/controller/

```
|-- add_pravega.go
|-- controller.go
`-- pravega
    `-- pravega_controller.go
```

Структура

pravega-operator/pkg/controller/

```
|-- add_pravega.go
|-- controller.go
`-- pravega
    `-- pravega_controller.go
```

pravega/pravega_controller.go

```
func Add(mgr manager.Manager) error {
    return add(mgr, newReconciler(mgr))
}

func newReconciler(mgr manager.Manager) reconcile.Reconciler {
    return &ReconcilePravegaCluster{ client: mgr.GetClient(),
scheme: mgr.GetScheme()
     }
}
```

pravega/pravega_controller.go

```
func Add(mgr manager.Manager) error {
    return add(mgr, newReconciler(mgr))
}

func newReconciler(mgr manager.Manager) reconcile.Reconciler {
    return &ReconcilePravegaCluster{ client: mgr.GetClient(),
scheme: mgr.GetScheme()
     }
}
```

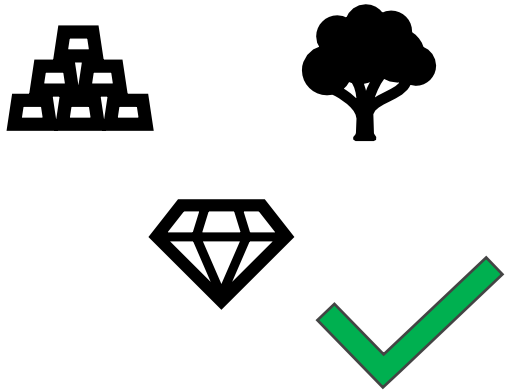
pravega/pravega_controller.go

```
func Add(mgr manager.Manager) error {
    return add(mgr, newReconciler(mgr))
}

func newReconciler(mgr manager.Manager) reconcile.Reconciler {
    return &ReconcilePravegaCluster{ client: mgr.GetClient(),
scheme: mgr.GetScheme()
     }
}
```

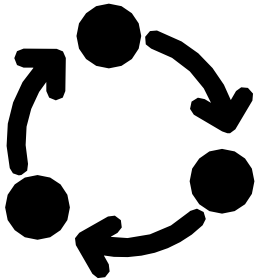
pravega/pravega_controller.go

```
func add(mgr manager.Manager, r reconcile.Reconciler) error {
 c, err := controller.New("pravegacluster-controller", mgr,
controller.Options{Reconciler: r})
 if err != nil { return err }


 err = c.Watch(&source.Kind{Type&pravegav1alpha1.PravegaCluster{}},
&handler.EnqueueRequestForObject{})
 if err != nil { return err }
 return nil
}
```

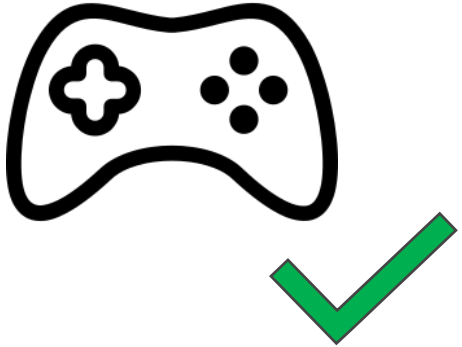
Что получили?



Custom Resource



Event Stream

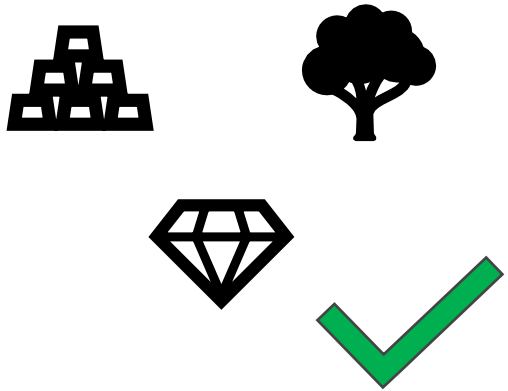


Custom Controller

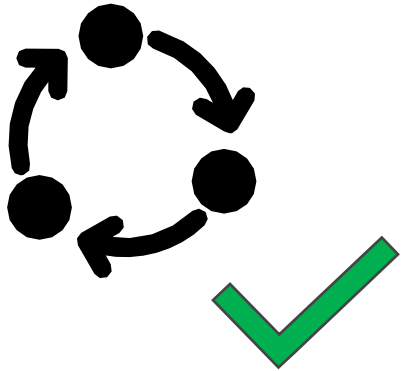
pravega/pravega_controller.go

```
func add(mgr manager.Manager, r reconcile.Reconciler) error {
 c, err := controller.New("pravegacluster-controller", mgr,
controller.Options{Reconciler: r})
 if err != nil { return err }


 err = c.Watch(&source.Kind{Type&pravegav1alpha1.PravegaCluster{}},
&handler.EnqueueRequestForObject{})
 if err != nil { return err }
 return nil
}
```

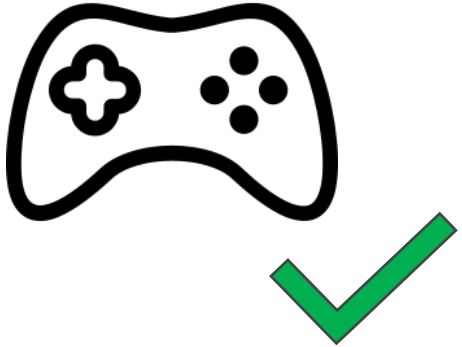
Что получили?



Custom Resource



Event Stream



Custom Controller

Reconcile()

```
func (…) Reconcile(request reconcile.Request) (…) {
  pravegaCluster := &pravegav1alpha1.PravegaCluster{}

  err := r.client.Get(context.TODO(), request.NamespacedName,
pravegaCluster)
  if err != nil {}

  changed := pravegaCluster.WithDefaults()

  if changed { }
  ...

  return reconcile.Result{RequeueAfter: ReconcileTime}, nil
}
```

Reconcile()

```
func (…) Reconcile(request reconcile.Request) (…) {
  pravegaCluster := &pravegav1alpha1.PravegaCluster{}

  err := r.client.Get(context.TODO(), request.NamespacedName,
pravegaCluster)
  if err != nil {}

  changed := pravegaCluster.WithDefaults()

  if changed { }
  ...

  return reconcile.Result{RequeueAfter: ReconcileTime}, nil
}
```

Reconcile()

```
func (…) Reconcile(request reconcile.Request) (…) {
  pravegaCluster := &pravegav1alpha1.PravegaCluster{}

  err := r.client.Get(context.TODO(), request.NamespacedName,
pravegaCluster)
  if err != nil {}

  changed := pravegaCluster.WithDefaults()

  if changed { }
  ...

  return reconcile.Result{RequeueAfter: ReconcileTime}, nil
}
```

Reconcile()

```
func (…) Reconcile(request reconcile.Request) (…) {
  pravegaCluster := &pravegav1alpha1.PravegaCluster{}

  err := r.client.Get(context.TODO(), request.NamespacedName,
pravegaCluster)
  if err != nil {}

  changed := pravegaCluster.WithDefaults()

  if changed { }
  ...

  return reconcile.Result{RequeueAfter: ReconcileTime}, nil
}
```

WithDefaults()

```
func (s *ClusterSpec) withDefaults() (changed bool) {
    ...
    if s.ZookeeperUri == "" {
        changed = true
        s.ZookeeperUri = DefaultZookeeperUri
    }
    ...
    if s.Bookkeeper == nil {... }
    ...
    return changed
}
```


Оператор
Controller-runtime
Reconciler
Go-client

Тестирование операторов

У всех

- Fake go-client
- Operator SDK
- e2e – Go + Ginkgo

У нас

- Robot Framework
- Longevity

Где эксплуатация?


ТУТ

В коде!

- Реализованы в Pravega Operator:
  - Установка Pravega
  - Rolling Upgrade
  - Проверка компонент кластера
- Реализованы «где-нибудь» еще:
  - Масштабирование
  - Использование других операторов
  - Multitenancy
  - Сервисные процедуры

Установка Pravega

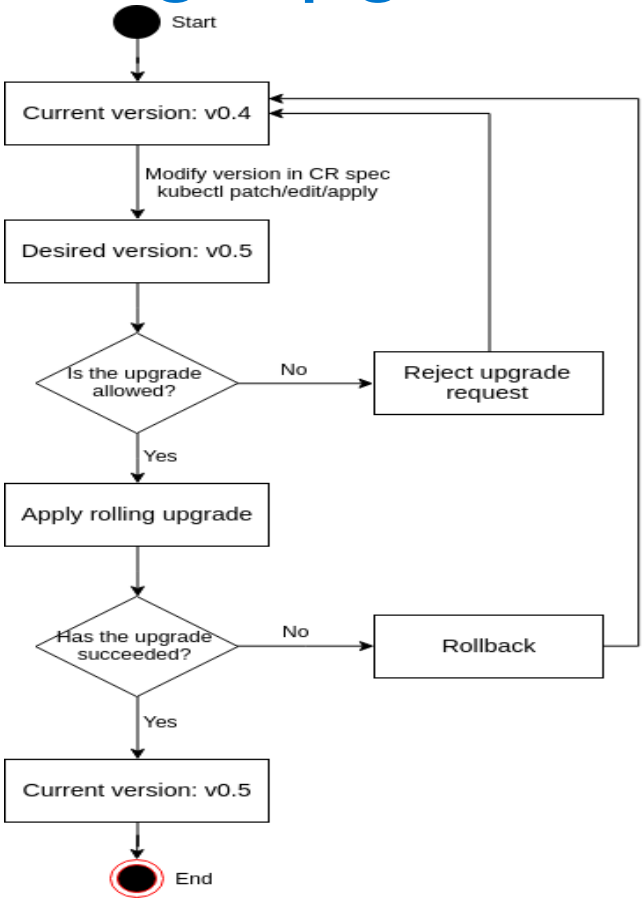
1. Установка внешних компонент
2. Установка внутренних:
    1. BookKeeper
    2. Pravega Controller
    3. Pravega Segment Storage

Установка Pravega

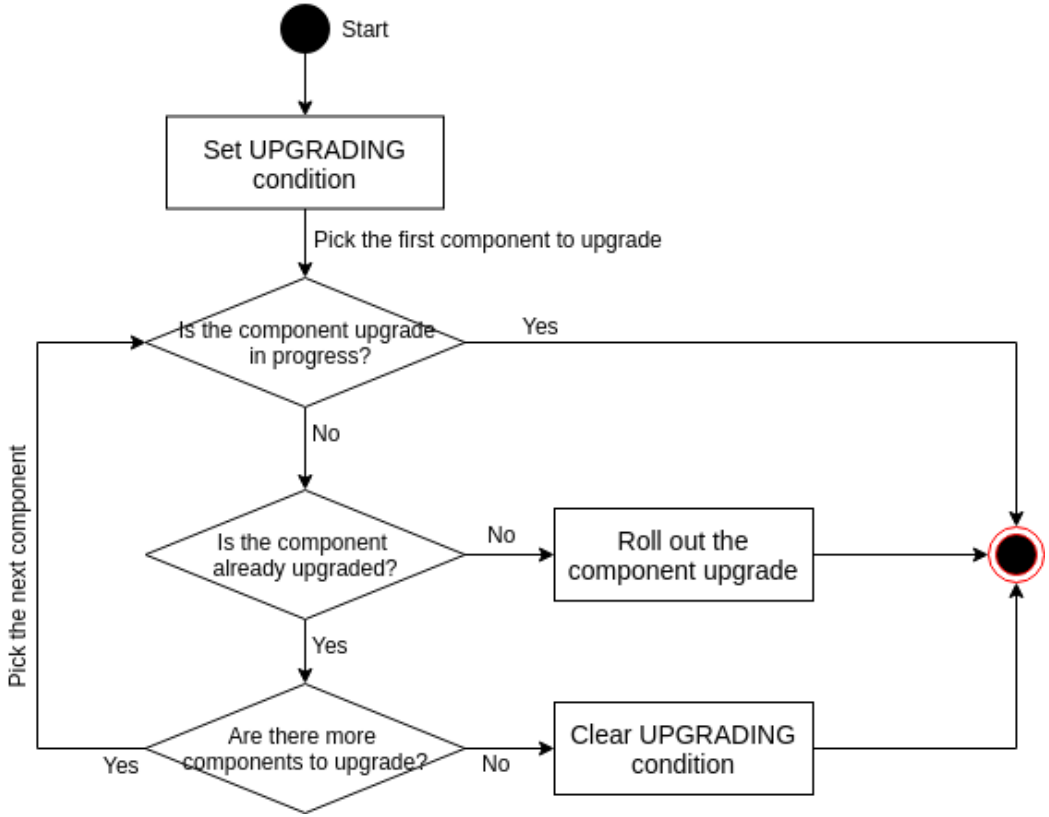
```
func deployCluster(p *pravegav1alpha1.PravegaCluster)() {

  err = r.deployBookie(p)
  ...
  err = r.deployController(p)
  ...
  err = r.deploySegmentStore(p)
  ...
}
```

Непростой Rolling Upgrade

Непростой Rolling Upgrade



Непростой Rolling Upgrade



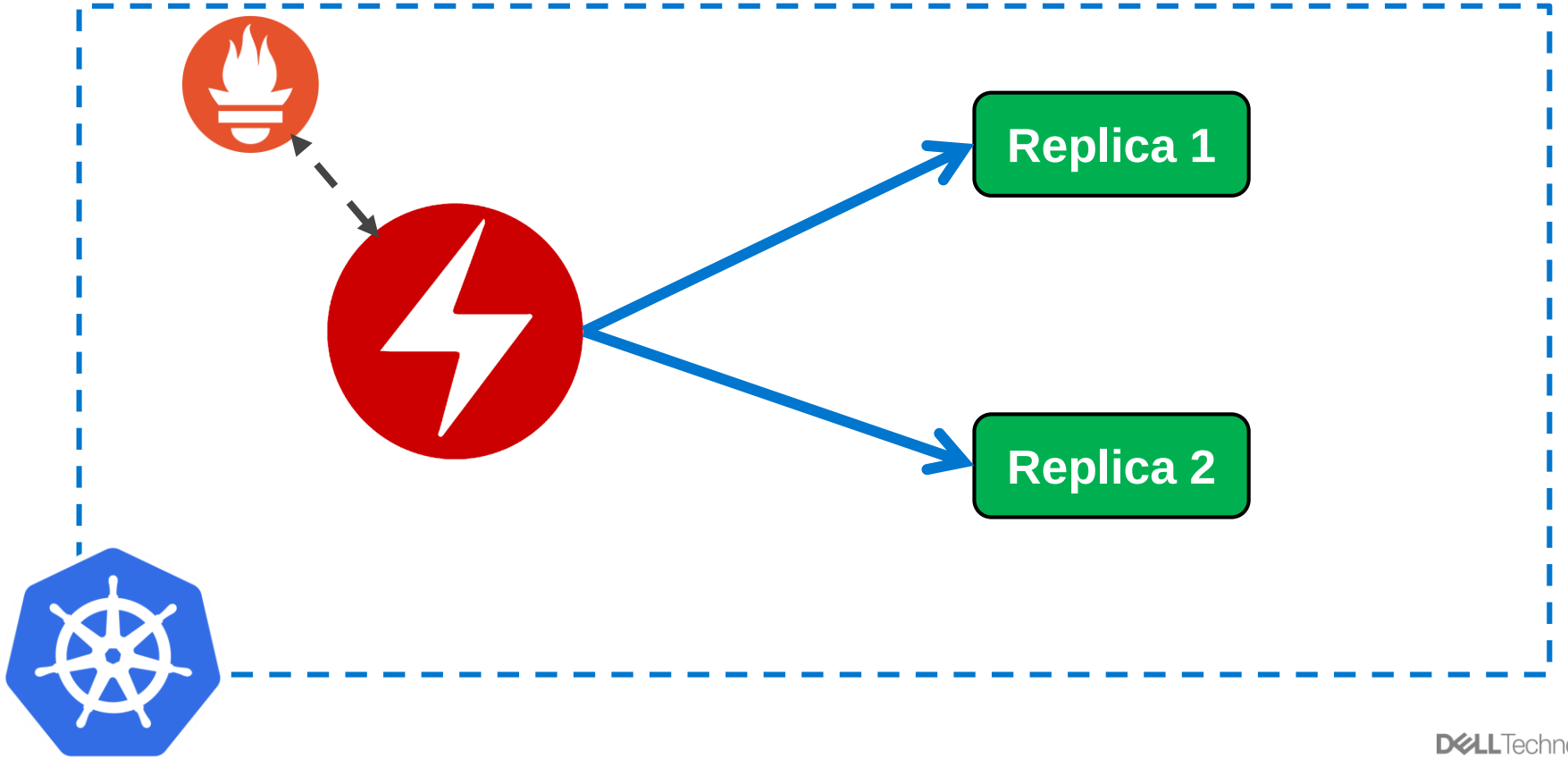
Непростой Rolling Upgrade

- Обновляем ***только*** внутренние компоненты
- Обновление ***только*** по этапам:
  1. BookKeeper
  2. Pravega Controller
  3. Pravega Segment Store

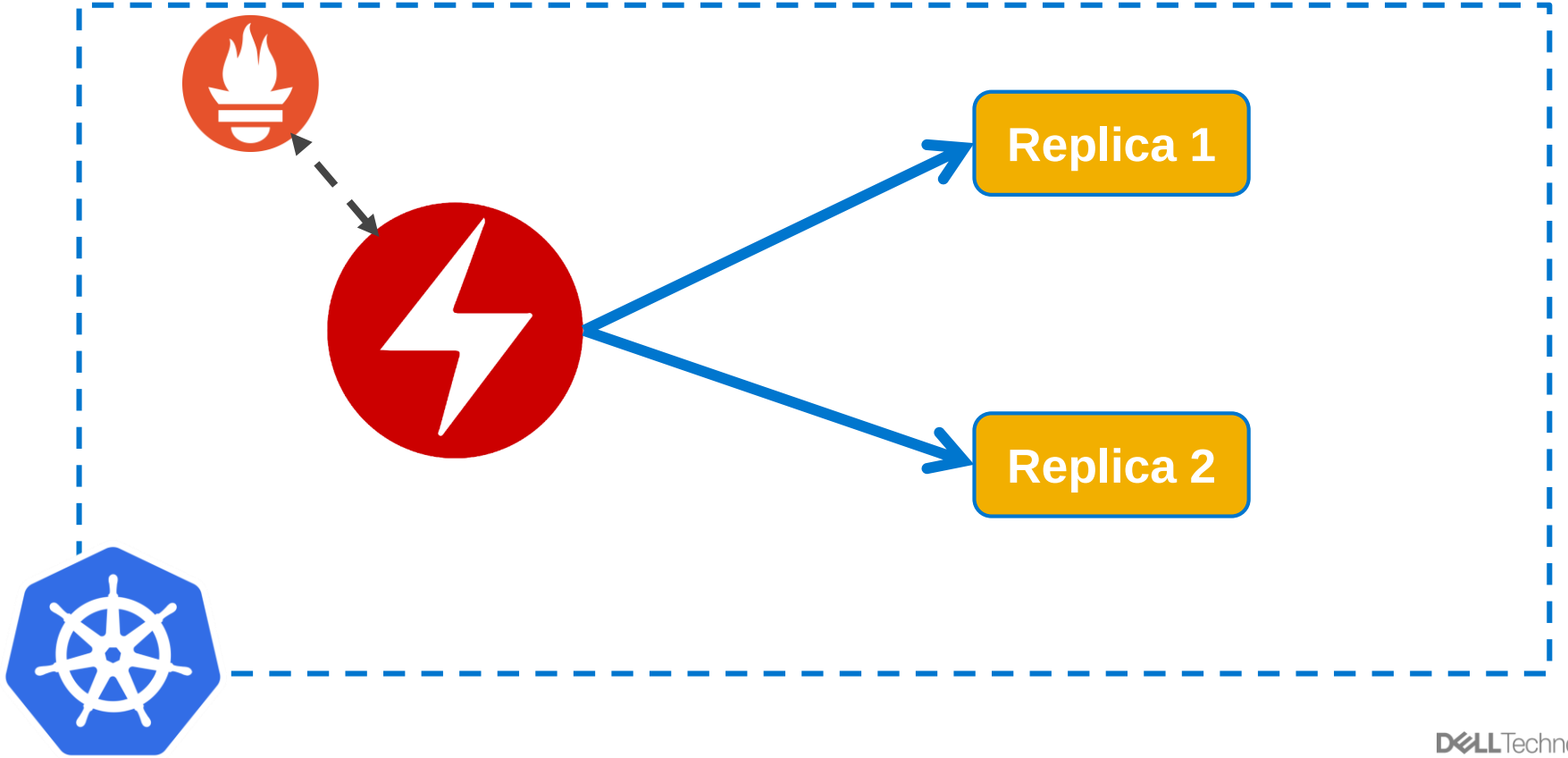
Проверка компонент кластера

```
err = r.syncBookieSize(p)
...
err = r.syncSegmentStoreSize(p)
...
err = r.syncControllerSize(p)
```

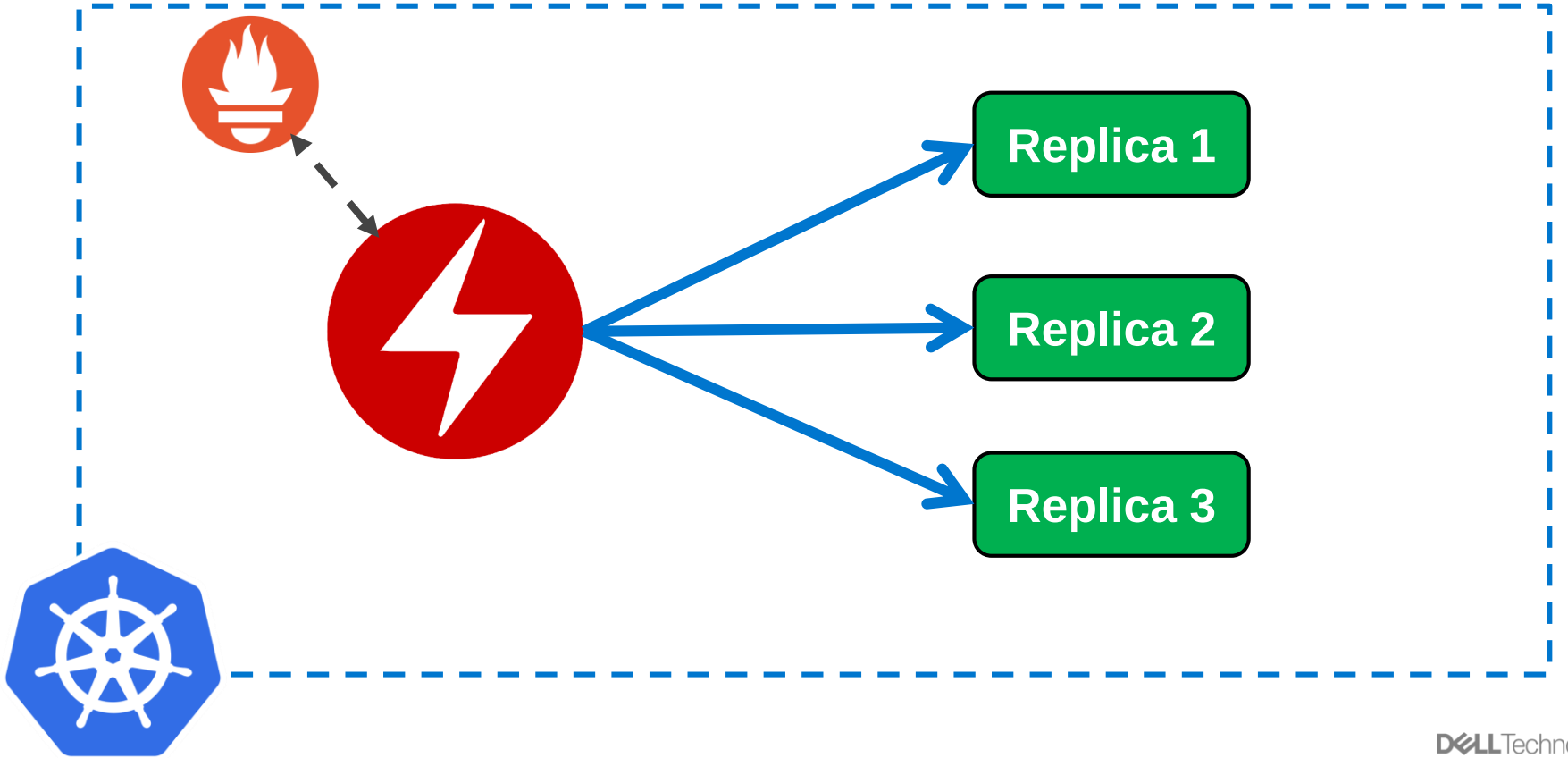
Масштабирование



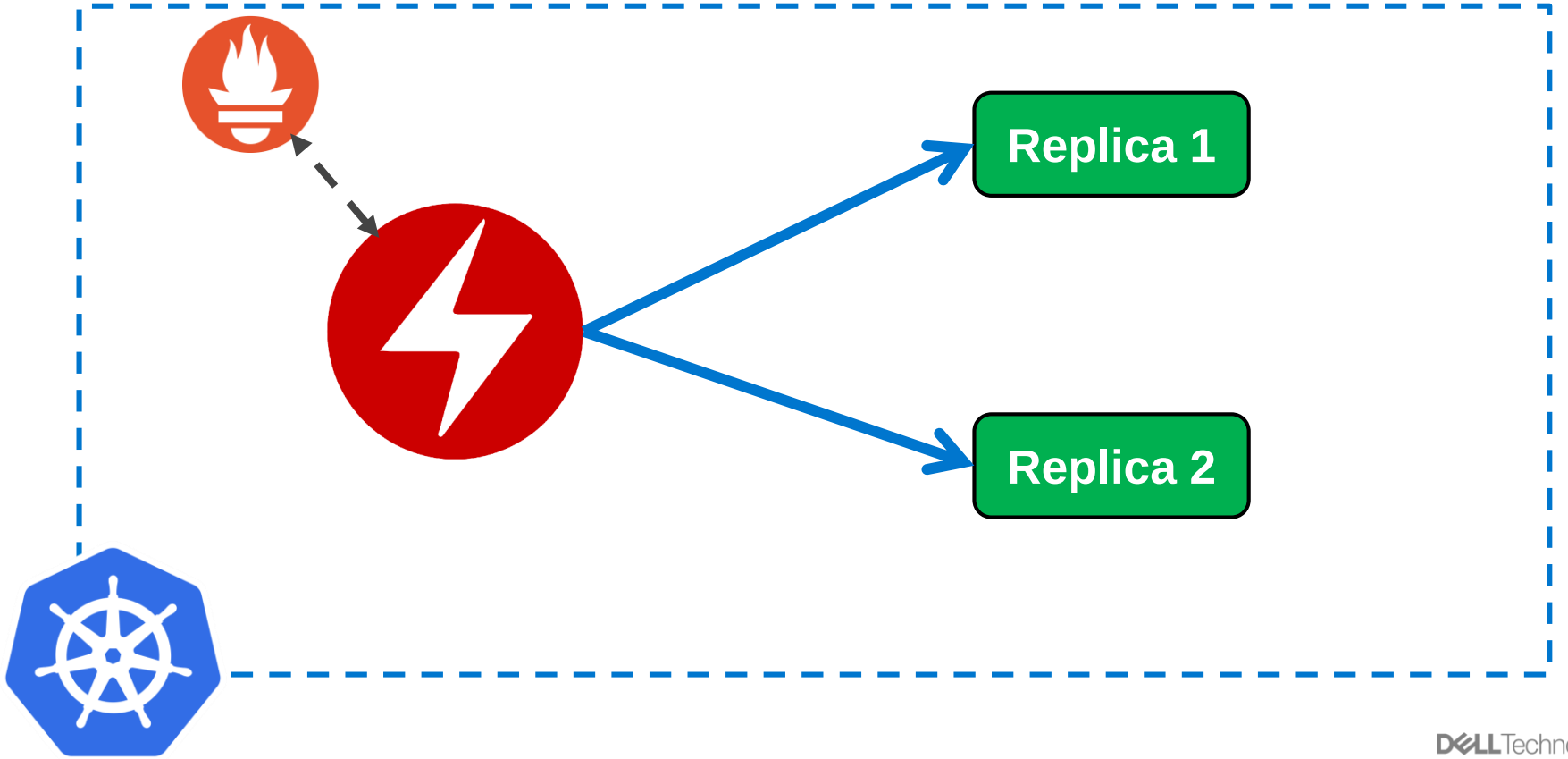
Масштабирование



Масштабирование



Масштабирование



Использование других операторов

- GoLang magic:

```
import "…/zookeeper-operator/pkg/apis/zookeeper/v1beta1"
```

- Helm:

```
dependencies:
- name: zookeeper-operator
```

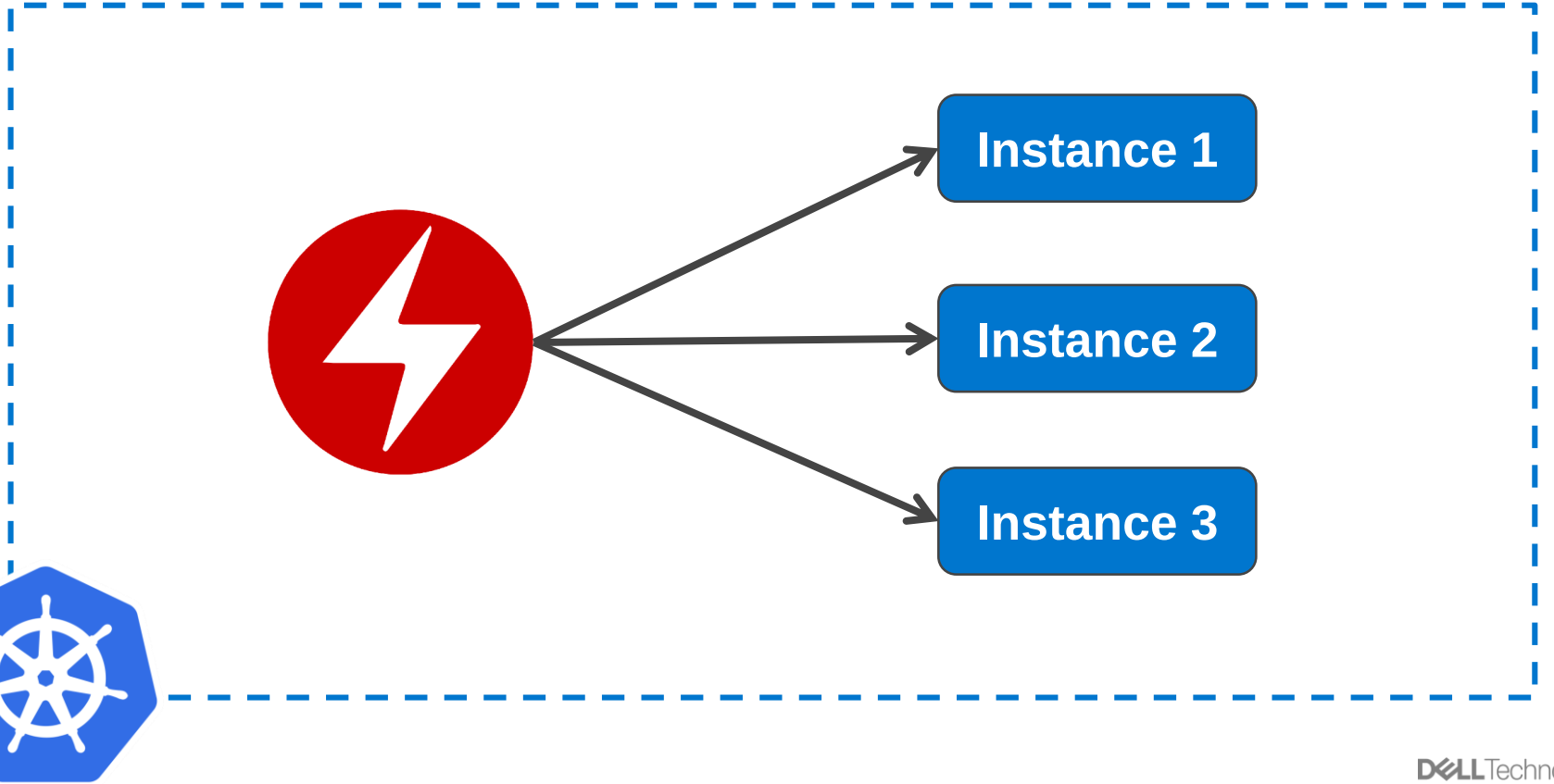
- kubectl:

```
$ kubectl apply -f zk-operator.yaml
```

Multitenancy

- Несколько экземпляров работают в одной среде
- Логическая изоляция

Multitenancy



Multitenancy

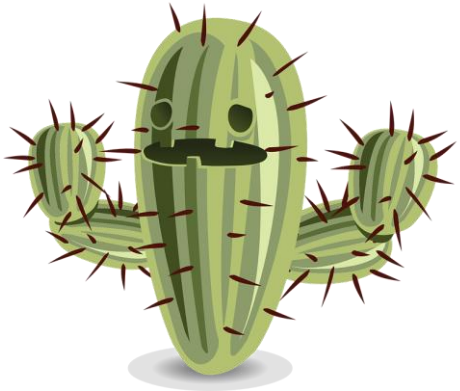
```
type InstanceContext struct {
    mu sync.Mutex parent
    context.Context list
    map[string]ctxWithCancel
}
```

Сервисные процедуры над ПО

- Миграции:
  - сервисов
  - ноды
  - PV
- Maintenance mode
- Health Checks
- …

«Мыши плакали, кололись, но…

- Не k8s-way для развертывания
- **НО** «удобен» для конфигурирования
- «Максимально минимальную» конфигурацию



Инкапсуляция создания CRD, RBAC,…

```
charts/
|-- pravega
|   |-- Chart.yaml
|   |-- templates
|   |   |-- clusterrolebinding.yaml
|   |   |-- clusterrole.yaml
|   |   |-- crd.yaml
|   |   |-- rolebinding.yaml
|   |   |-- role.yaml
|   |   |-- ….yaml
|   `-- values.yaml
```

Понятная конфигурация

```
pravega:
  image:
    repository: pravega/pravega
  controllerReplicas: 1
  segmentStoreReplicas: 1
  debugLogging: false
  cacheVolumeRequest: 20Gi
```

Понятная конфигурация

```
pravega:
  image:
    repository: pravega/pravega
  controllerReplicas: 1
  segmentStoreReplicas: 1
  debugLogging: false
  cacheVolumeRequest: 20Gi
```

Индустрия идет этим путем

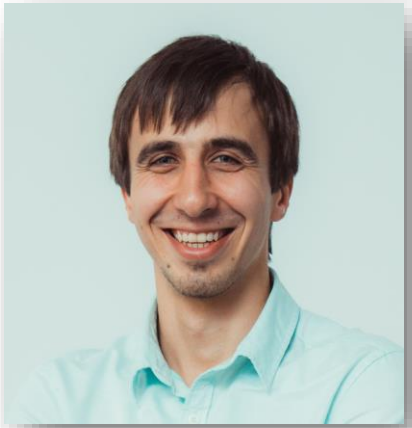
Elastic Cloud on Kubernetes

ClickHouse Operator

Prometheus Operator

. . .

#WinningTogether, расширяя k8s



Михаил Борисов

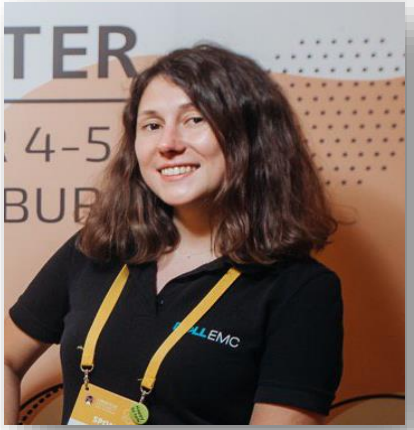
Руслан Шмелев

Александр Рассанов

#WinningTogether

Алексей Акопян

Михаил Саламатов



Анастасия Соловьева

Выводы

- Все в код!
- Kubernetes – это фреймворк
- Операторы – это система управления
- Низкий порог входа – многое делается за нас


«Чем больше сила, тем больше ответственность» (с)


DELL Technologies

#askmeplease

Fedor.Chemashkin@dell.com