

Criando Kubernetes Operators com o operator-sdk

Kubernetes

O Kubernetes foi projetado para automação. Existe muita automação integrada do núcleo do Kubernetes. Você pode usar o Kubernetes para automatizar o deploy e execução de workloads e você pode automatizar como o Kubernetes faz isso.

Operators

O conceito de **operator pattern** do Kubernetes permite estender o comportamento do cluster sem modificar o código do próprio Kubernetes, vinculando **controllers** a um ou mais **custom resources**.

Controller

Um **controller** rastreia pelo menos um **tipo de recurso** do Kubernetes. Esses objetos têm um campo **spec** que representa o **estado desejado**.

O(s) controlador(es) desse recurso são responsáveis por fazer com que o estado atual se aproxime daquele estado desejado.

Resources

Um **recurso** é um endpoint na API do Kubernetes que armazena uma coleção de objetos de API de um determinado tipo; por exemplo, o recurso built-in **pods** contém uma coleção de objetos **Pod**.

Custom resources

Um **recurso personalizado** é uma extensão da API do Kubernetes que não está necessariamente disponível em uma instalação padrão do Kubernetes. Ele representa uma personalização de uma instalação específica do Kubernetes.

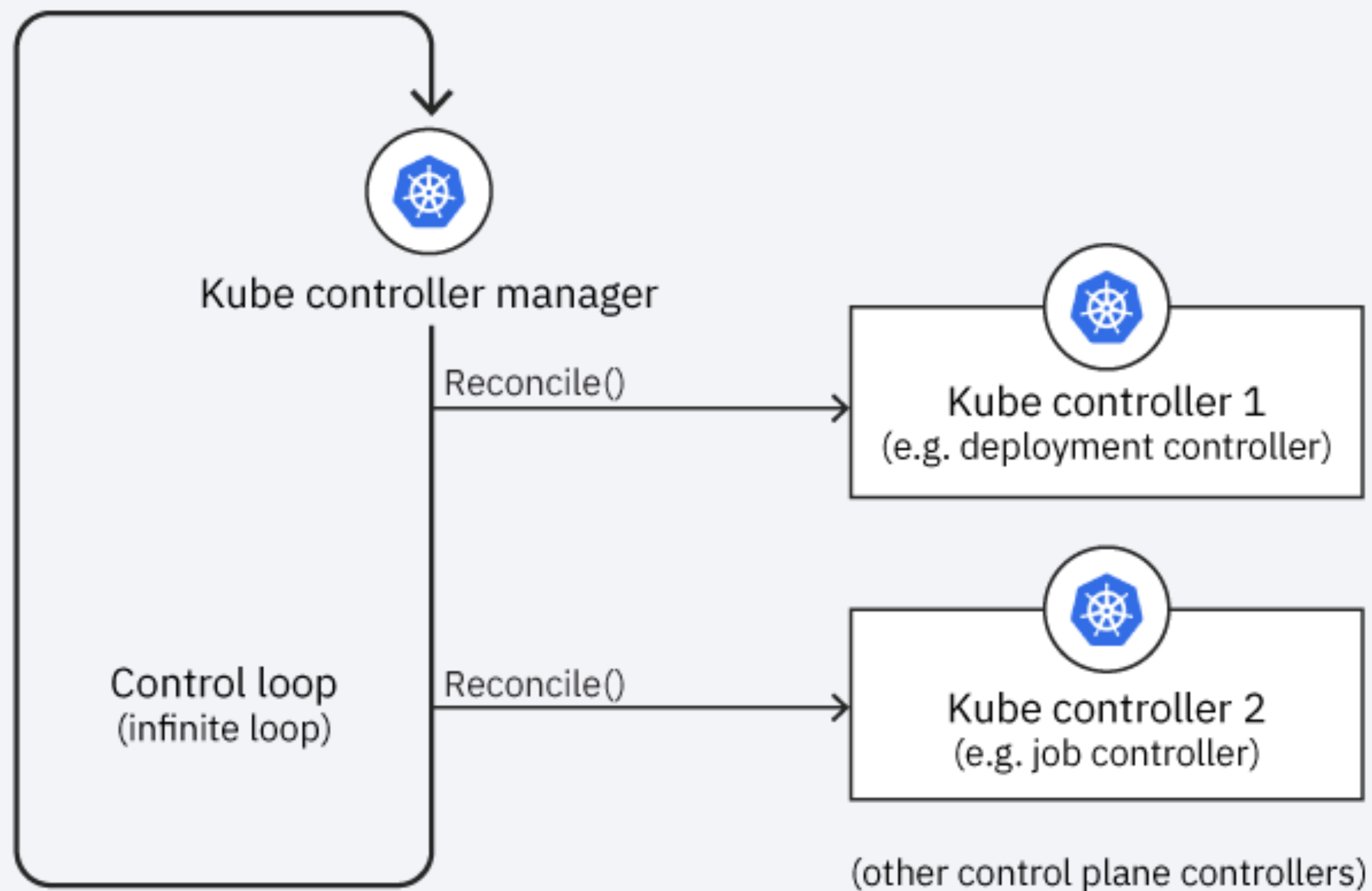
Algumas das coisas que você pode usar um operador para automatizar:

- deploy de uma aplicação sob demanda
- fazer e restaurar backups do estado desse aplicativo
- lidar com atualizações do código do aplicativo juntamente com alterações relacionadas, como esquemas de banco de dados ou configurações extras

Reconciliation Loop

O Kubernetes é baseado no conceito de uma especificação declarativa do estado desejado do cluster e no uso de loops de reconciliação para direcionar o estado real para o estado desejado.

Control plane



Operator SDK

<https://sdk.operatorframework.io/>

Show me the
code

Objetivo

```
apiVersion: v1
kind: Namespace
metadata:
  name: application-sample
---
apiVersion: minnetto.dev/v1alpha1
kind: Application
metadata:
  name: application-sample
  namespace: application-sample
spec:
  image: nginx:latest
  replicas: 2
  port: 80
```

Scaffolding

```
operator-sdk init --domain minetto.dev --repo github.com/eminetto/k8s-operator-talk  
operator-sdk create api --version v1alpha1 --kind Application --resource --controller
```

Adicionar informações ao Custom Resource Application

```
// api/v1alpha1/application_types.go
type ApplicationSpec struct {
    Image      string `json:"image,omitempty"`
    Replicas   int32  `json:"replicas,omitempty"`
    Port       int32  `json:"port,omitempty"`
}
```

Gerar os manifests

make manifests

Código do Controller

controllers/application_controller.go

main.go

```
if err = (&controllers.ApplicationReconciler{
    Client: mgr.GetClient(),
    Scheme: mgr.GetScheme(),
}).SetupWithManager(mgr); err != nil {
    setupLog.Error(err, "unable to create controller", "controller", "Application")
    os.Exit(1)
}
```

Deploy

```
make docker-build docker-push IMG=registry.hub.docker.com/eminetto/k8s-operator-talk:latest  
make deploy IMG=registry.hub.docker.com/eminetto/k8s-operator-talk:latest
```

Demo

OperatorHub

<https://operatorhub.io/>

Código

<https://github.com/eminetto/k8s-operator-talk>

Links

- ◎ Criando Kubernetes Operators com o operator-sdk
- ◎ Escrevendo testes para um Kubernetes Operator