Kubernetes e Controllers

Quem sou eu?

Nathan Martins

Atualmente sou Tech Lead / Sênior SRE na Palenca (YC 21) e responsável por todo o time do BR. Neuro-divergente 🫶🏼

Quase uma década na área

- Ex-Argyle
- Ex-Stone
- Ex- Congressy
- Contribuidor no #SIG-CLI no Kubernetes

(ESTAMOS CONTRATANDO)

Público-alvo

Experiência variada no Docker/Kubernetes

Objetivos

Objetivos

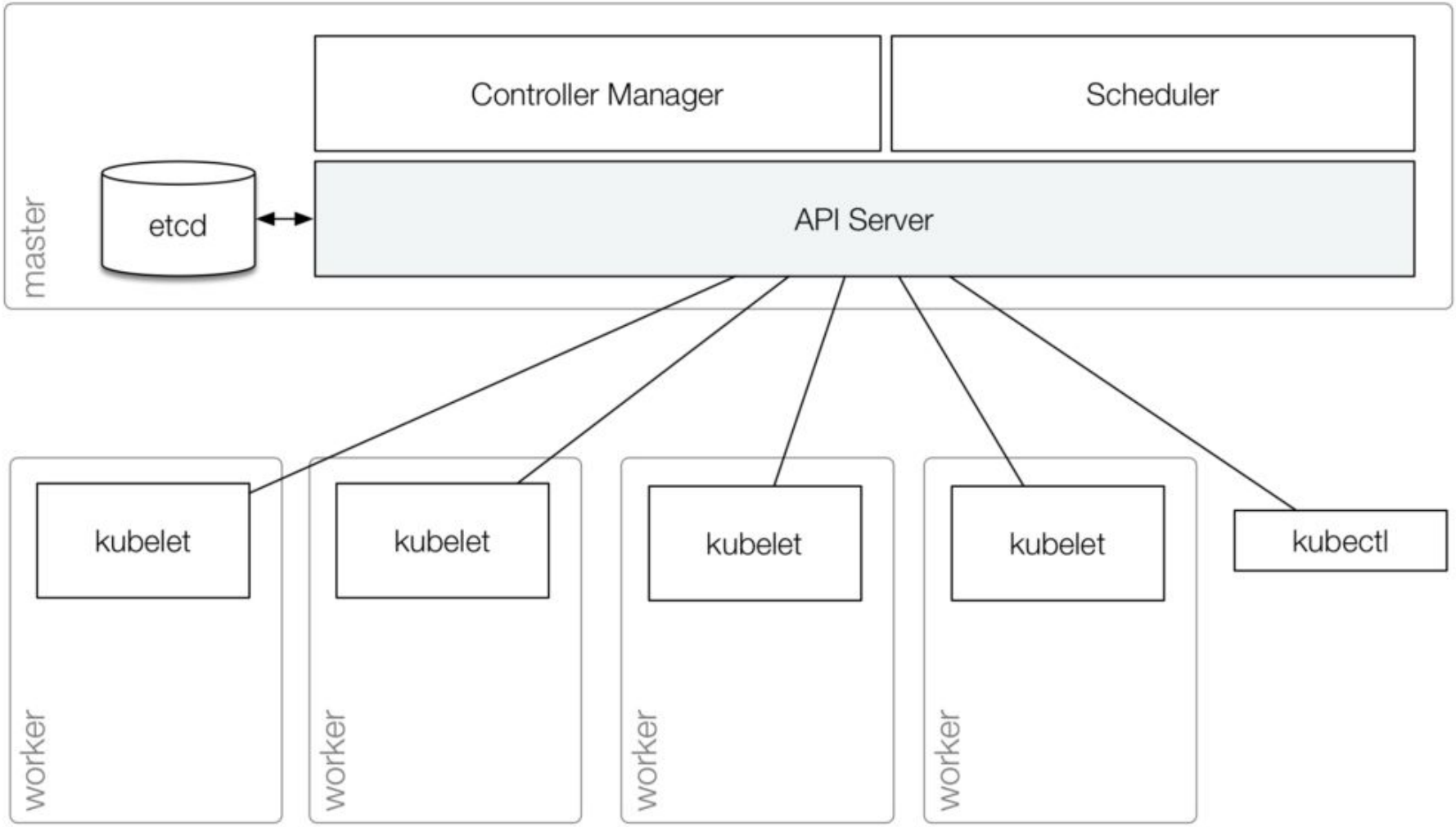
- Entenda o que são controladores e operadores Kubernetes.

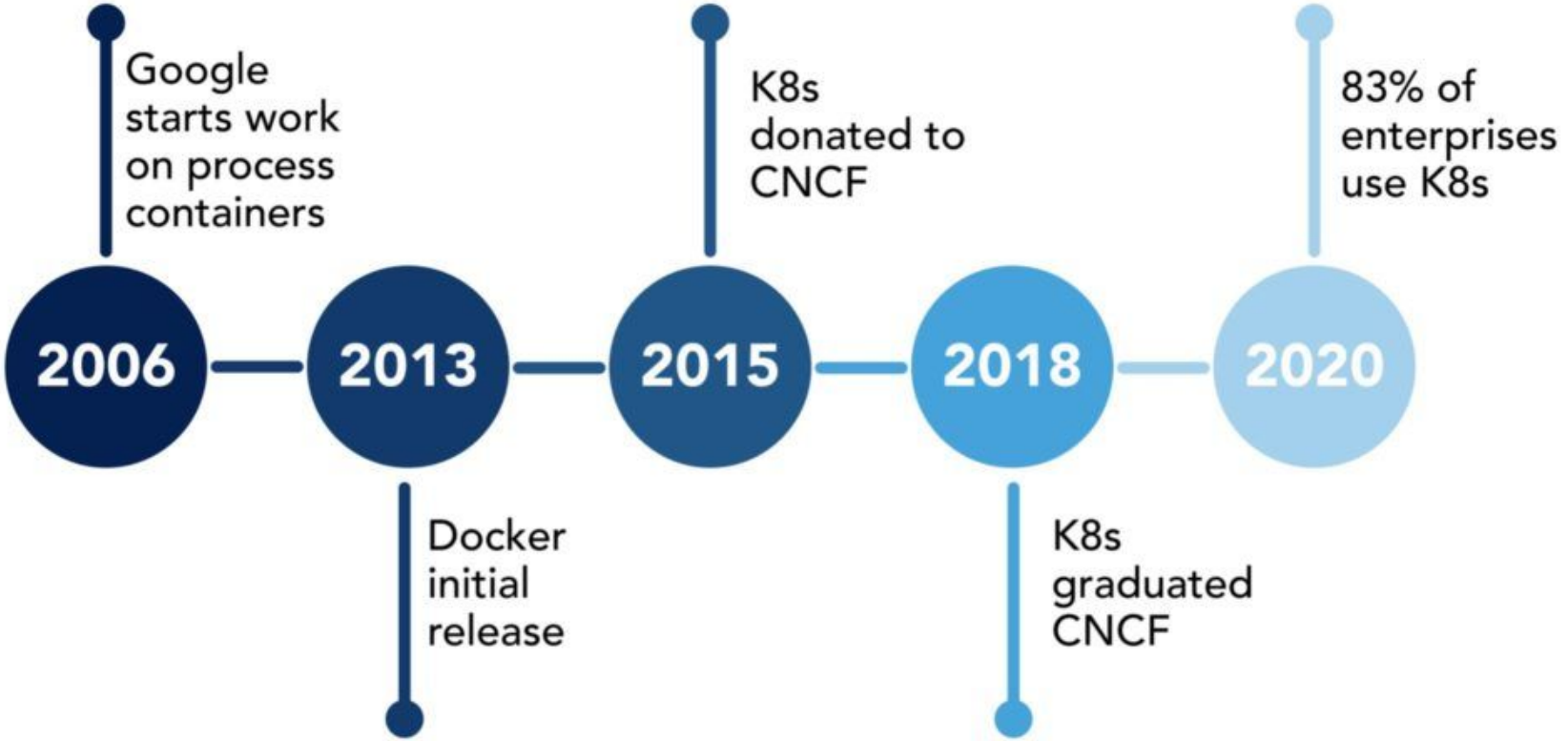
- Reconhecer os benefícios da utilização de operadores.

- Obtenha uma compreensão básica de por que você pode escrever um.

Introdução

- O Kubernetes e seu papel na orquestração de contêineres.
- Conceito de estado desejado versus estado real.
- Qual a função do plano de controle do Kubernetes.


Controller Manager
Scheduler
master
etcd
API Server
kubelet
kubelet
kubelet
kubelet
kubectl
worker
worker
worker
worker


Google starts work on process containers
2006
Docker initial release
2013
K8s donated to CNCF
2015
K8s graduated CNCF
2018
83% of enterprises use K8s
2020

Porque o K8s tem um ecosistema tão vasto?

Extensibilidade!

Controladores no Kubernetes

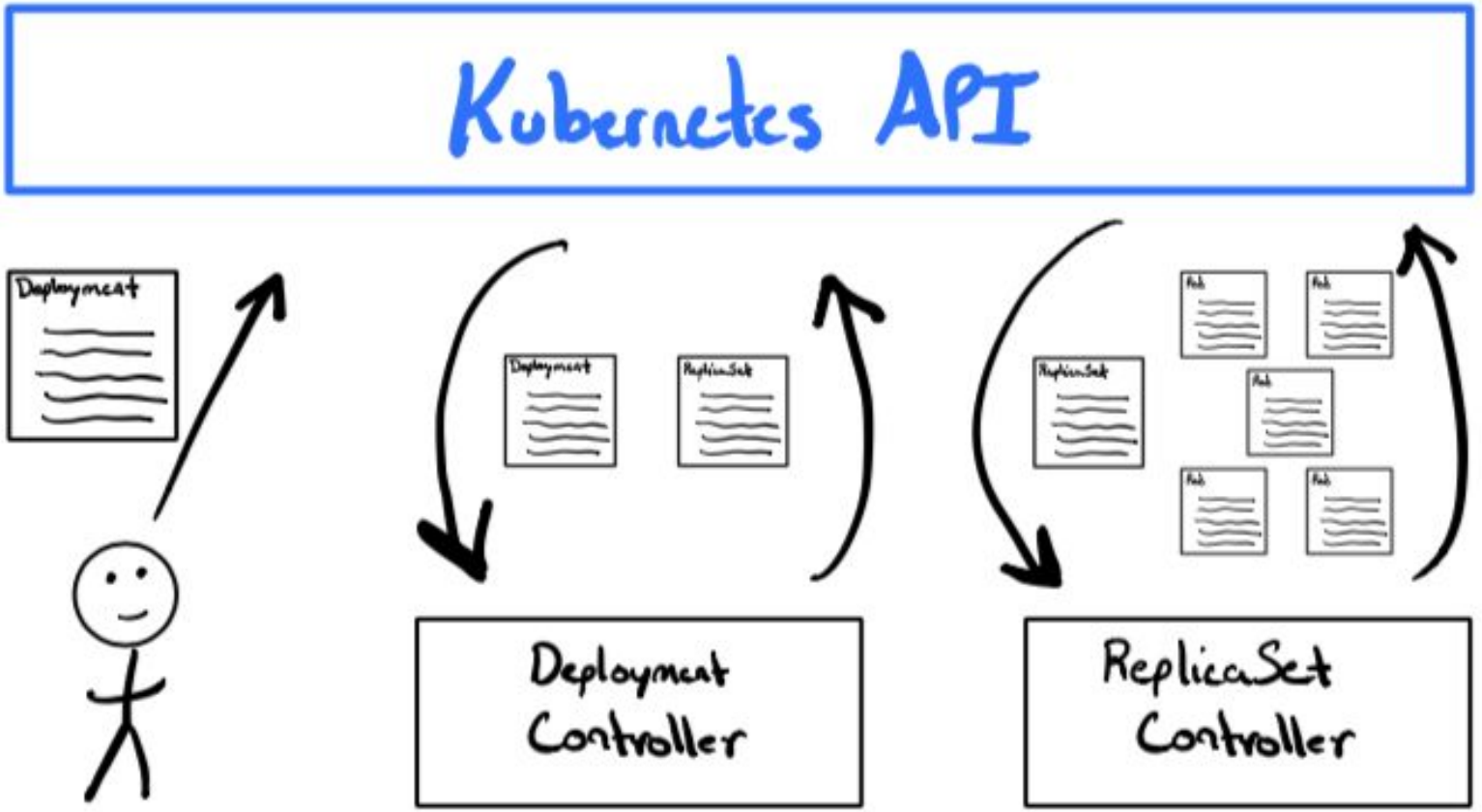
Funcionalidade principal: monitorar continuamente o estado do cluster e tomar ações para reconciliar quaisquer discrepâncias entre o estado desejado e o real.

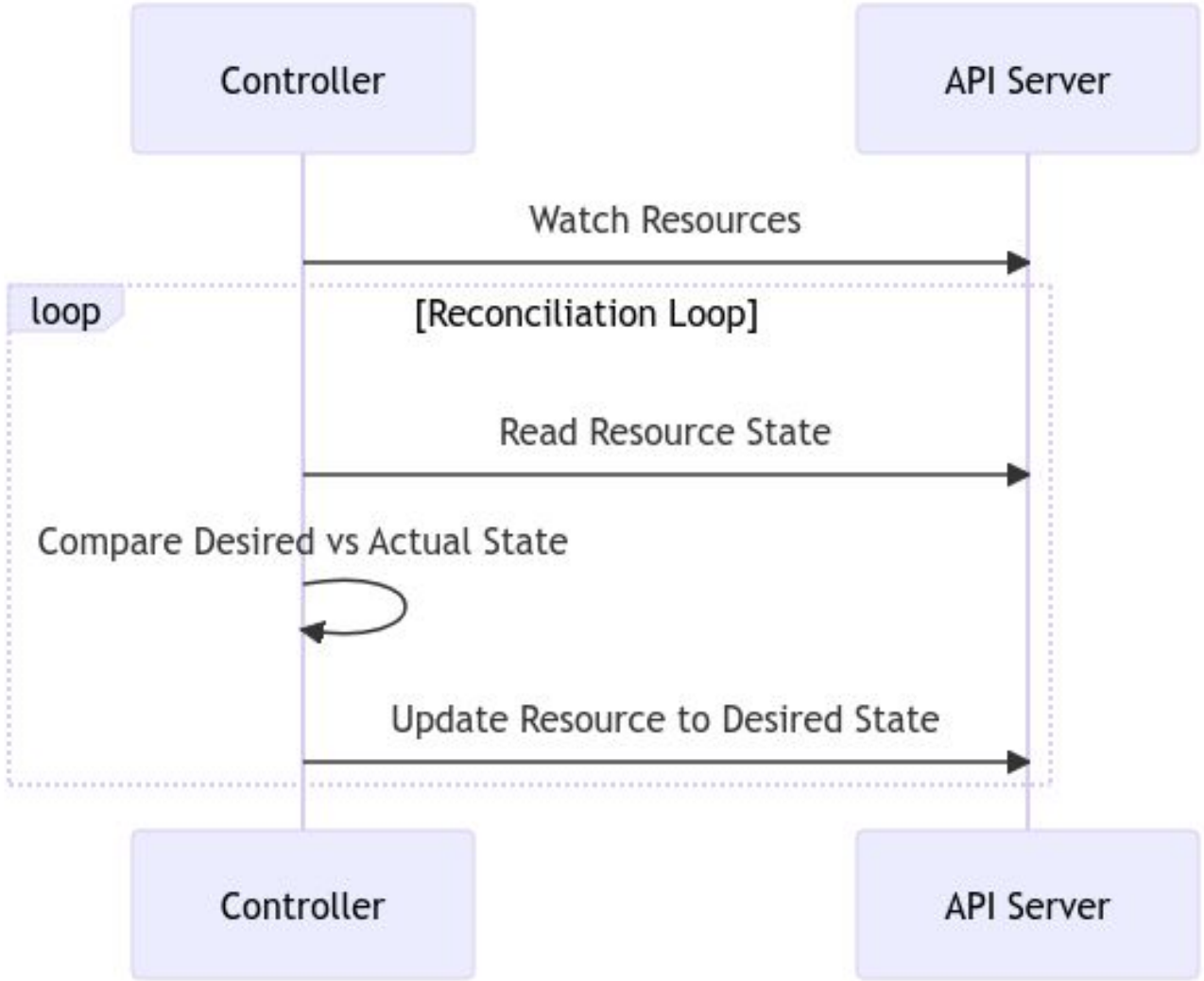
Controladores no Kubernetes

Controladores exemplos:

- Controlador ReplicaSet (mantém o número desejado de réplicas de Pod).
- Controlador de Nodes (gerencia nodes de trabalho).


Always has been
Wait, it's all CONTROLLERS?
K8S USER
KUBE API
imgflip.com


Kubernetes API
Deployment
Deployment
ReplicaSet
Deployment Controller
ReplicaSet Controller


Controller
API Server
Watch Resources
loop
[Reconciliation Loop]
Read Resource State
Compare Desired vs Actual State
Update Resource to Desired State
Controller
API Server

Controladores no Kubernetes

Os controladores são genéricos e não específicos da aplicação.

Operadores: Assumindo o Controle

O que somos???


YAML
LIGHTYEAR
YAML EVERYWHERE
imgflip.com

Operadores: Assumindo o Controle

Um tipo especializado de controlador que gerencia aplicações complexas.

Operadores aproveitam as Definições de Recursos Personalizados (CRDs)

KUBE API

Operadores: Assumindo o Controle

Os CRDs definem novos tipos de recursos específicos para a aplicação que o operador gerencia (por exemplo, um MySQLCRD para um banco de dados MySQL).

Operadores: Assumindo o Controle

- Gerenciamento do ciclo de vida (implantação, escalonamento, atualizações)

- Conhecimento específico de domínio para aplicações complexas

- Gerenciamento simplificado de aplicativos para usuários

Operadores: Assumindo o Controle

Os SDKs do operador disponíveis para facilitar o desenvolvimento (por exemplo, SDK do operador para Go).

Por que escrever um operador?

Por que escrever um operador?

- Foco em cenários onde os operadores se destacam:
    - Gerenciar aplicações stateful (bancos de dados, sistemas de mensagens)

Por que escrever um operador?

- Foco em cenários onde os operadores se destacam:
    - Aplicativos que exigem configurações ou implantações específicas

Por que escrever um operador?

- Foco em cenários onde os operadores se destacam:
    - Padronização do gerenciamento do ciclo de vida de aplicativos em implantações

Tutorial do Controller Runtime

Controller Runtime GitHub

File: CRD.yaml

```yaml
apiVersion: myapp.example.com/v1alpha1
kind: MyApp
metadata:
  name: my-nginx-app
spec:
  size: 2
```

File: spec.go

```go
package v1alpha1

import (
    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
)

// MyAppSpec defines the desired state of the MyApp
type MyAppSpec struct {
    // Size is the number of Nginx pod replicas desired
    Size int32 `json:"size"`
}

// MyApp is the Schema for the myapps API
type MyApp struct {
    metav1.TypeMeta   `json:",inline"`
    metav1.ObjectMeta `json:"metadata,omitempty"`

    Spec MyAppSpec `json:"spec,omitempty"`
}
```

File: reconcile.go

```
// MyAppReconciler reconciles a MyApp object
type MyAppReconciler struct {
    client.Client
    Log log.Logger
}
```

File: reconcile.go

```
func (r *MyAppReconciler) Reconcile(req ctrl.Request) (ctrl.Result,
error) {
    ctx := context.Background()
    log := r.Log.WithValues("myapp", req.NamespacedName)

    // Get the MyApp instance
    // Get the current number of replicas (deployment might not
exist yet)

    // Ensure the deployment exists and has the desired replicas
    // Create a new deployment if it doesn't exist
    // Exit

}
```

Explicação

Planejamento: O controlador identifica o MyApp CRD e percebe a necessidade de reconciliação.

Explicação

Execução: O controlador recupera a especificação do CRD (duas réplicas).

Explicação

Verificação: O controlador verifica o cluster e vê que não há pods do Nginx em execução (estado real diferente do desejado).

Explicação

Ação: O controlador cria dois pods do Nginx para alcançar o estado desejado.

Explicação

Repetição: O ciclo continua. O controlador monitora o cluster e garante que haja sempre duas réplicas do Nginx em execução, mesmo que um pod falhe ou seja reiniciado.

Conclusão

Conclusão

- Os controladores mantêm o estado desejado, os operadores gerenciam aplicações complexas.

- O crescente ecossistema de operadores Kubernetes disponíveis para diversas aplicações.

- Espaço para perguntas.