

GitOps e
Kustomize


pipefy

› pipefy.com


Whoami

Cearense, cabra da peste, reside em Floripa, trabalho com tecnologia há 16 anos, formado em Sistemas de Informação, sysadmin root, casado há 12 anos, Pai do Thomas, Skatista fulero, Unix-like addict e Cloud Computing e Cloud Native Evangelist, SRE na Pipefy.



A large, stylized blue arrow graphic pointing to the right, composed of multiple parallel lines of varying shades of blue, creating a sense of depth and movement.

**Quais são os problemas que
precisamos resolver?**



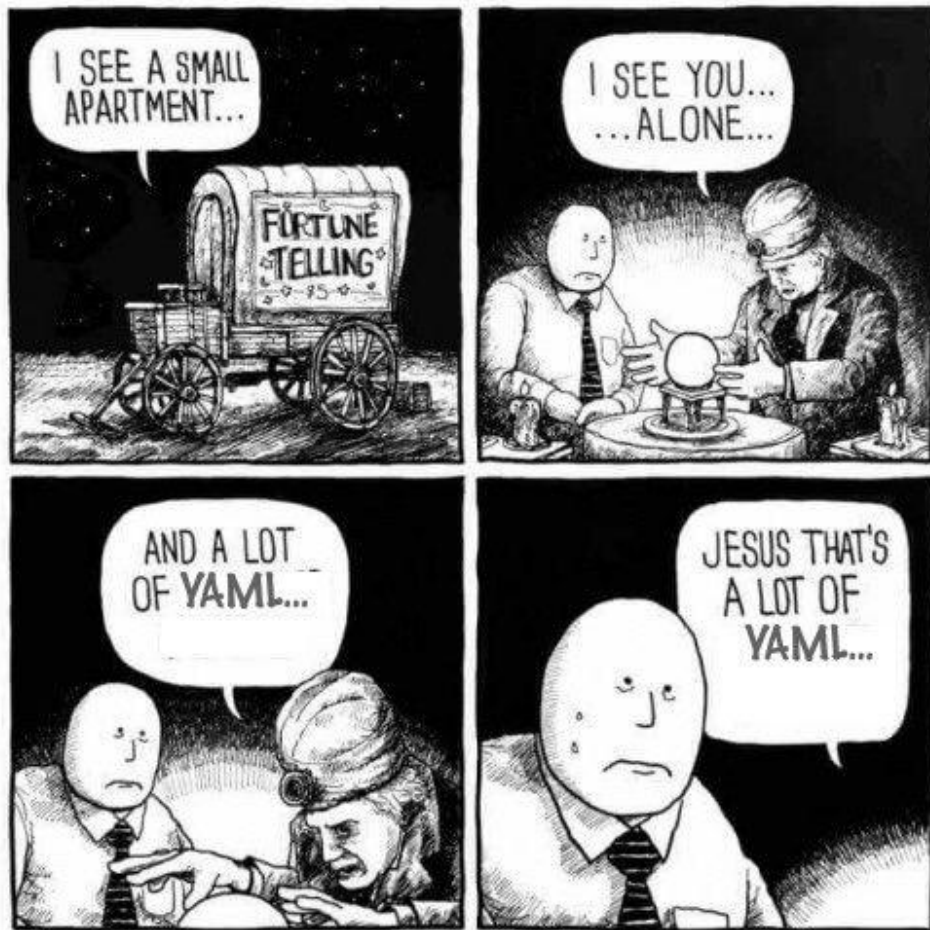
Evoluir a infraestrutura na mesma velocidade da aplicação

Os times estarem na mesma página quando se trata de modificações referente ao ambiente ou que possam causar impacto em determinada aplicação

Quebrar as barreiras entre time de Ops e Devs

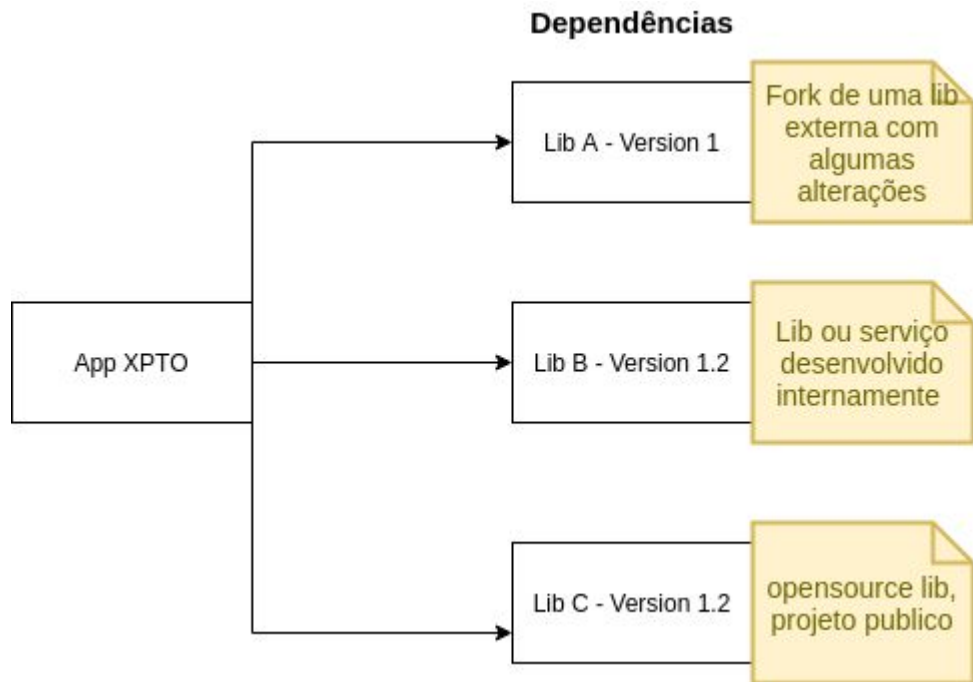
Formas que não evolui na mesma velocidade da aplicação



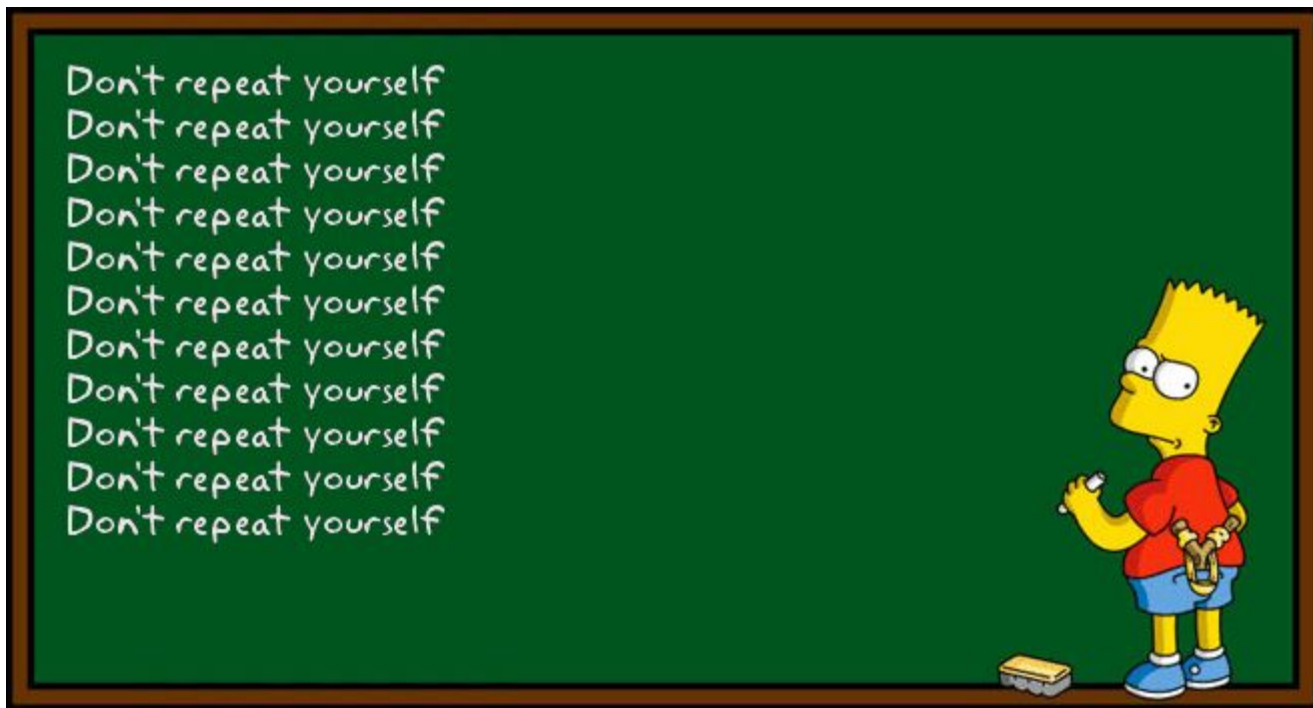


* Parece piada mas é a vida de Ops hoje em dia

Entendendo a evolução de aplicações



DRY Principle

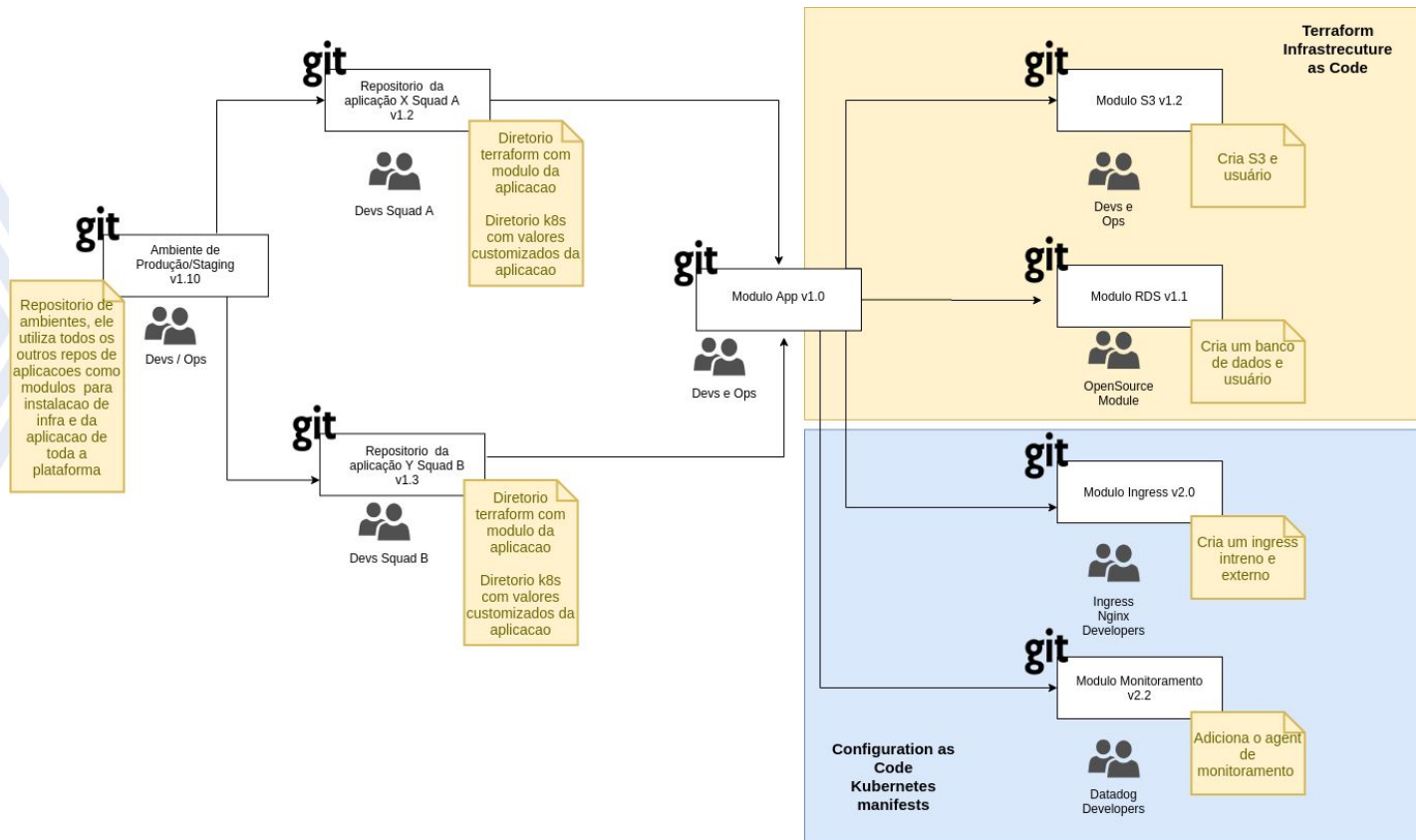


LINES OF CODE:

SO MANY!



Como evoluir a infraestrutura da mesma forma que as aplicações ou melhor em conjunto com a aplicação



A large, stylized blue arrow pointing to the right, composed of multiple parallel lines of varying shades of blue, creating a sense of depth and movement.

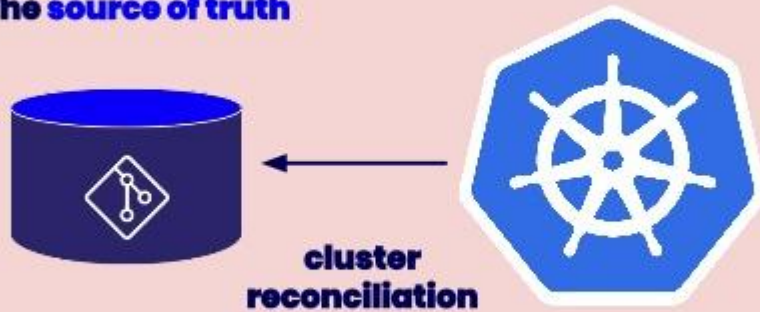
GitOps

GITOPS

GitOps = IaC + Pull Requests + CI/CD

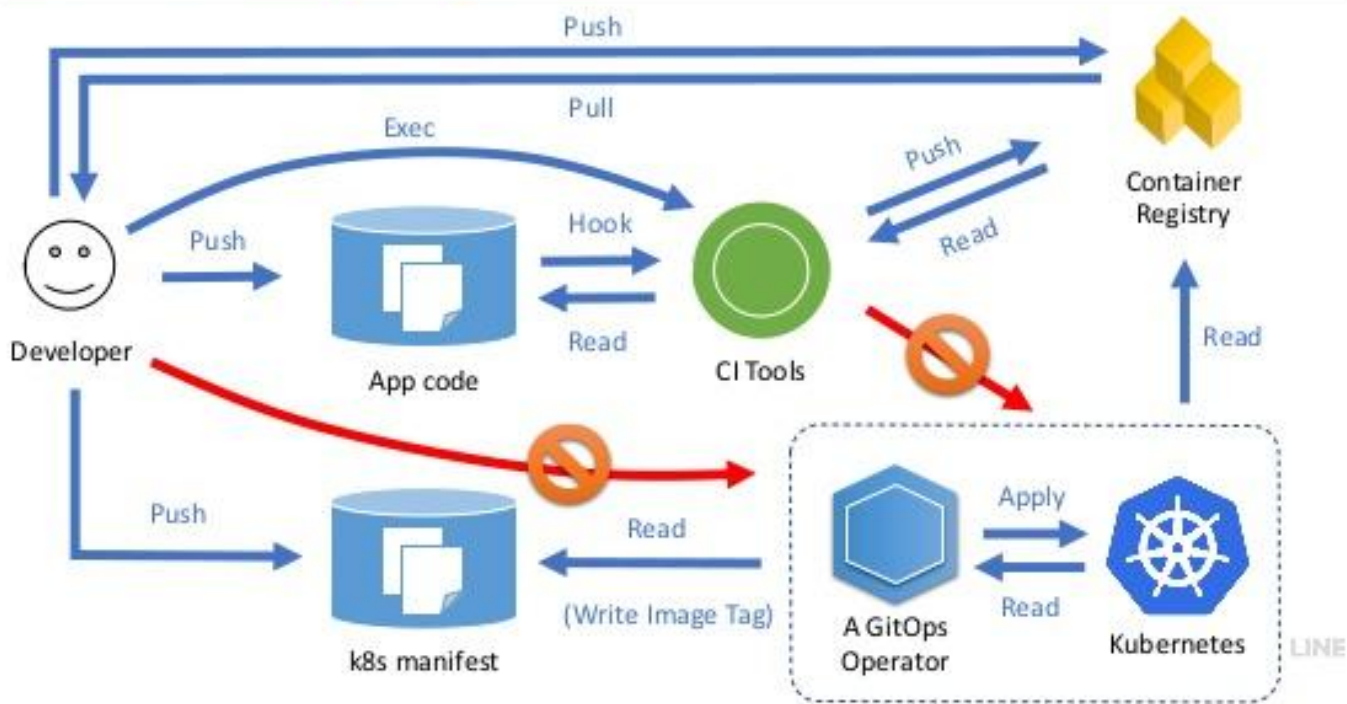
GitOps é uma maneira de fazer o gerenciamento de cluster do Kubernetes e entrega de aplicativos e infraestrutura. Ele funciona usando Git como uma única fonte de verdade para infraestrutura declarativa e aplicativos. Com o Git no centro de seus pipelines de entrega, os desenvolvedores podem fazer solicitações pull para acelerar e simplificar as implantações de aplicativos e tarefas operacionais.

The source of truth



LUNAR

Common GitOps Pipeline



Antes do GitOps

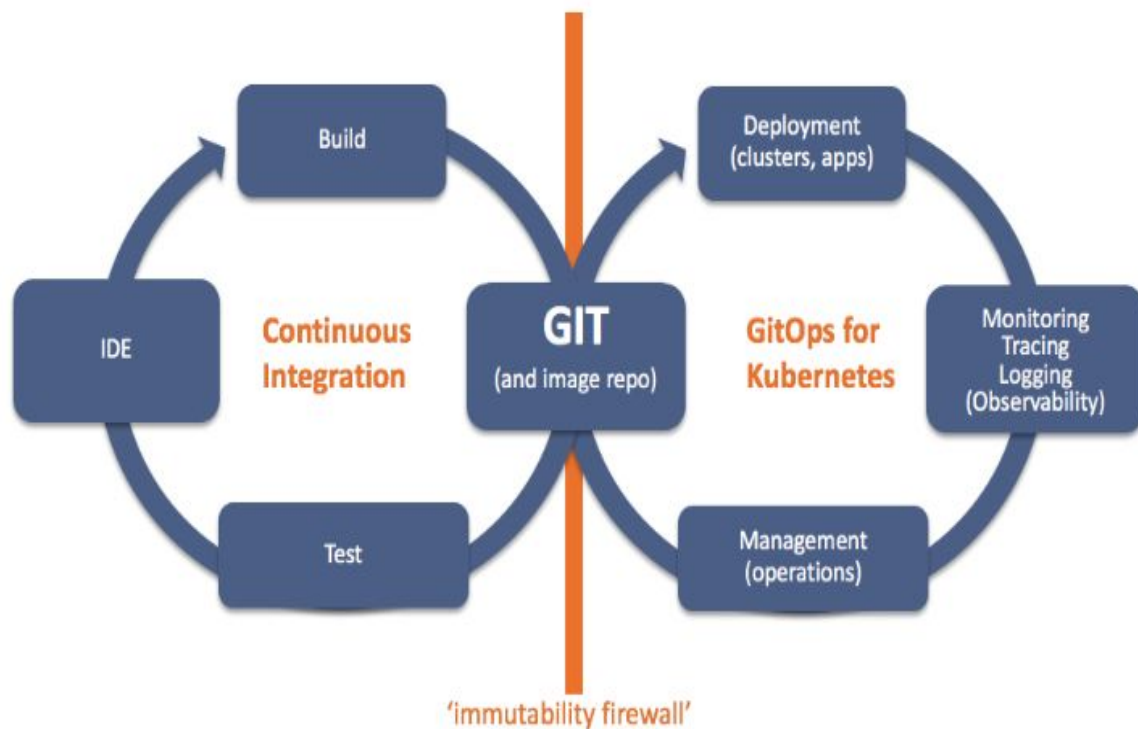
- Alterações eram manuais
 - Secrets e criação de infra estrutura era sempre feita antes ou depois da aplicação mas não ao mesmo tempo
- Alto risco de erros e problemas
- Rollback e visibilidade era mais complicada
- Uma simples alteração poderia causar um incidente e era mais difícil de identificar o que foi alterado e quando foi alterado

Exemplo de pequena alteração incorreta na infraestrutura



Resolvendo problemas de infraestrutura





Git as the single source of truth of a system's desired state

GitOps Diffs compare desired state with observed state (eg Kubediff, Terradiff, Canary..)

ALL intended operations are committed by pull request, for all environments

ALL diffs between GIT and observed state lead to (auto) convergence using tools like K8s

ALL changes are observable, verifiable and audited indisputably, with rollback & D/R

Não é uma boa prática para GitOps na pipeline

```
k8s-deploy() {  
  kubectl set image deployment/frontend www=image:v2  
}
```

Boa prática usando GitOps na pipeline



```
k8s-deploy() {  
    export RELEASE_TAG=$CI_COMMIT_SHORT_SHA  
    git config --global user.email "sre@pipefy.com"  
    git config --global user.name "sre"  
    git checkout -f $CI_COMMIT_REF_NAME  
    if [ -f k8s/$ENVIRONMENT/kustomization.* ]; then  
        yq w -i k8s/$ENVIRONMENT/kustomization.* images[0].newTag $RELEASE_TAG  
        sed -i -e "s/"$RELEASE_TAG"/\"'$RELEASE_TAG'"/g" k8s/$ENVIRONMENT/kustomization.*  
    else  
        for yaml in `grep -r -l "$CI_REGISTRY_IMAGE" k8s/$ENVIRONMENT`; do  
            if grep -q "kind: Deployment" $yaml; then  
                yq w -i $yaml spec.template.spec.containers[0].image $CI_REGISTRY_IMAGE:$RELEASE_TAG  
            fi  
            if grep -q "kind: CronJob" $yaml; then  
                yq w -i $yaml spec.jobTemplate.spec.template.spec.containers[0].image $CI_REGISTRY_IMAGE:$RELEASE_TAG  
            fi  
        done  
    fi  
    git add k8s/$ENVIRONMENT  
    git commit -m "Release update version: $RELEASE_TAG"  
    git push -o ci_skip origin/master  
}
```


reconciliation



ArgoCD

sync-waves



Project: default
Labels: app.kubernetes.io/instance=apps
Status:  Missing  OutOfSync
Repositor... <https://github.com/argoproj/argocd-exam...>
Target Re... HEAD
Path: sync-waves
Destinati... <https://kubernetes.default.svc>
Namespa... sync-waves

 SYNC

 REFRESH

 DELETE

kustomize-guestbook



Project: default
Labels: app.kubernetes.io/instance=apps
Status:  Missing  OutOfSync
Repositor... <https://github.com/argoproj/argocd-exam...>
Target Re... HEAD
Path: kustomize-guestbook
Destinati... <https://kubernetes.default.svc>
Namespa... kustomize-guestbook

 SYNC

 REFRESH

 DELETE

helm-guestbook



Project: default
Labels: app.kubernetes.io/instance=apps
Status:  Missing  OutOfSync
Repositor... <https://github.com/argoproj/argocd-exam...>
Target Re... HEAD
Path: helm-guestbook
Destinati... <https://kubernetes.default.svc>
Namespa... helm-guestbook

 SYNC

 REFRESH

 DELETE

helm-hooks



Project: default
Labels: app.kubernetes.io/instance=apps
Status:  Missing  OutOfSync
Repositor... <https://github.com/argoproj/argocd-exam...>
Target Re... HEAD
Path: helm-hooks
Destinati... <https://kubernetes.default.svc>
Namespa... helm-hooks

 SYNC

 REFRESH

 DELETE

apps

Project: default
Labels:
Status:  Healthy  Synced
Repositor... <https://github.com/alexec/argocd-exampl...>
Target Re... latest
Path: apps
Destinati... <https://kubernetes.default.svc>
Namespa... argocd

 SYNC

 REFRESH

 DELETE

Applications

+ NEW APP

SYNC APPS

🔍

🏠

🔧

👤

📄

📁

🔍

📄

📄

📁

🔍

📄

📄

FILTER BY:

SYNC

☐ Synced

1

☐ Unknown

0

☐ OutOfSync

0

HEALTH

☐ Healthy

1

☐ Unknown

0

☐ Progressing

0

☐ Suspended

0

☐ Degraded

0

☐ Missing

0

LABELS

PROJECTS

CLUSTERS

guestbook

Project:

default

Labels:

Status:

🟢 Healthy 🟢 Synced

Repository:

https://github.com/argoproj/argocd-exa...

Target R...

latest

Path:

guestbook

Destination:

https://kubernetes.default.svc

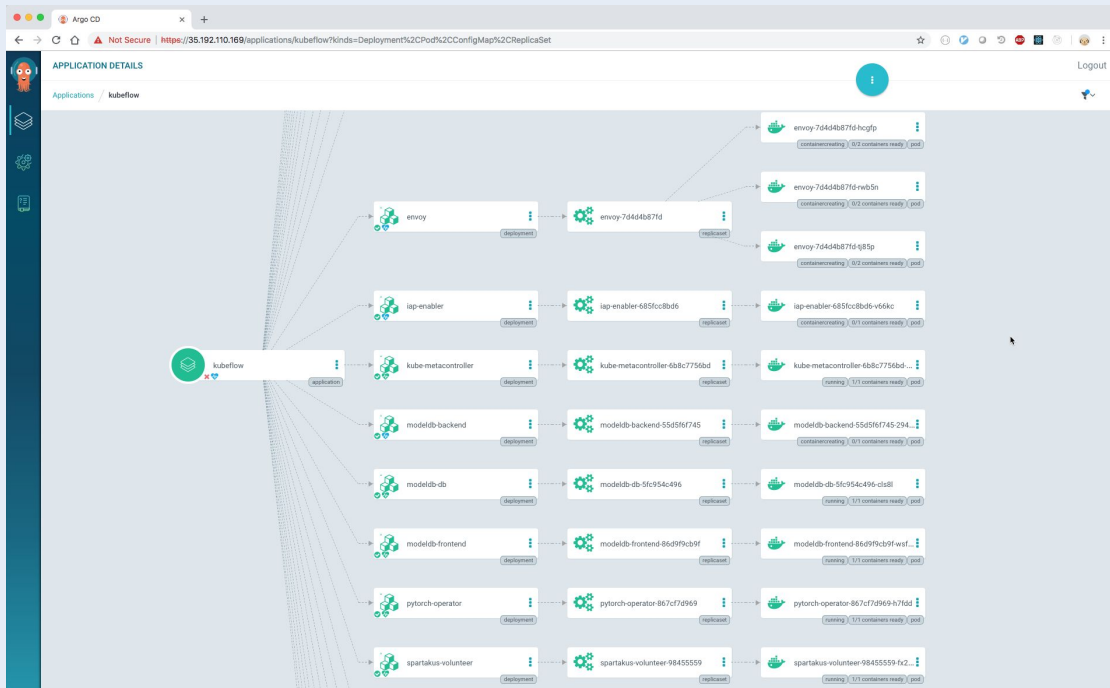
Namespace:

default

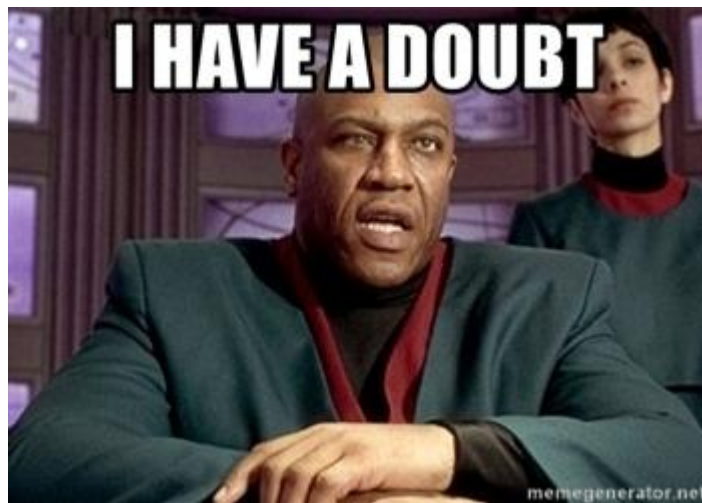
🔄 SYNC

🔄 REFRESH

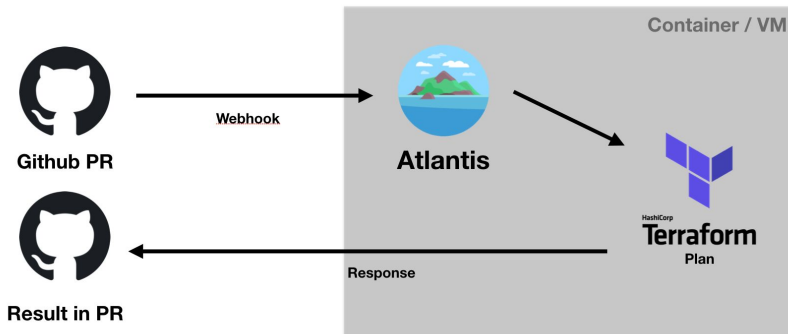
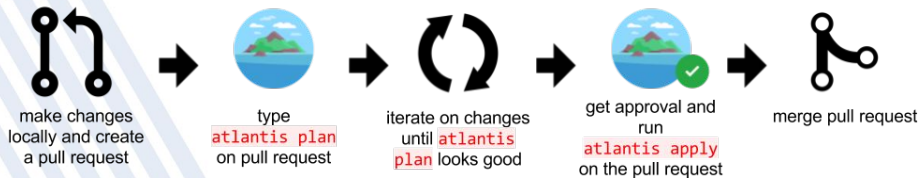
🗑️ DELETE



OK Ceará, está tudo lindo e maravilhoso mas o mundo não é só Kubernetes, como criar infraestrutura utilizando métodos de GitOps no mundo que não é Cloud Native?



Atlantis e Terraform



expel-devops commented 20 minutes ago

Ran Plan for dir: tf workspace: default

▼ Show Output

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

```
# module.super-slick-service-memorystore-staging.google_dns_record_set.dns will be created
+ resource "google_dns_record_set" "dns" {
  + id           = (known after apply)
  + managed_zone = "super-slick-service-dns"
  + name         = "super-slick-service-redis."
  + project      = (known after apply)
  + rrdatas     = (known after apply)
  + ttl         = 300
  + type        = "A"
}

# module.super-slick-service-memorystore-staging.google_redis_instance.redis_instance will be created
+ resource "google_redis_instance" "redis_instance" {
  + alternative_location_id = "us-east1-c"
  + authorized_network      = "https://www.googleapis.com/compute/v1/projects/.../globalNetworks/..."
  + create_time            = (known after apply)
  + current_location_id    = (known after apply)
  + display_name           = "Terraform Instance - super-slick-service"
  + host                   = (known after apply)
  + id                     = (known after apply)
  + location_id            = "us-east1-b"
  + memory_size_gb         = 1
  + name                   = "staging-super-slick-service"
  + port                   = (known after apply)
  + project                = (known after apply)
  + redis_version          = "REDIS_3_2"
  + region                 = "us-east1"
  + reserved_ip_range      = "..."
  + tier                    = "STANDARD_HA"
}
```

A próxima de geração de infraestrutura como código

Crossplane

```
apiVersion:  
database.aws.crossplane.io/v1beta1  
kind: RDSInstance  
metadata:  
  name: rdspostgresql  
spec:  
  forProvider:  
    dbInstanceClass: db.t2.small  
    masterUsername: masteruser  
    allocatedStorage: 20  
    engine: postgres  
    engineVersion: "9.6"  
skipFinalSnapshotBeforeDeletion:  
true  
writeConnectionSecretToRef:  
  namespace: crossplane-system  
  name: aws-rdspostgresql-conn  
providerRef:  
  name: aws-provider  
reclaimPolicy: Delete
```

<https://github.com/crossplane/crossplane>

ACK (AWS Controllers for Kubernetes)

```
$ kubectl get buckets  
ack-test-smoke-s3 -o yaml
```

```
apiVersion:  
s3.services.k8s.aws/v1alpha1  
kind: Bucket  
metadata:  
  name: ack-test-smoke-s3  
  namespace: default  
spec:  
  name: ack-test-smoke-s3
```

<https://aws.github.io/aws-controllers-k8s/>

A large blue arrow pointing to the right, composed of multiple parallel lines of varying shades of blue, creating a sense of depth and movement.

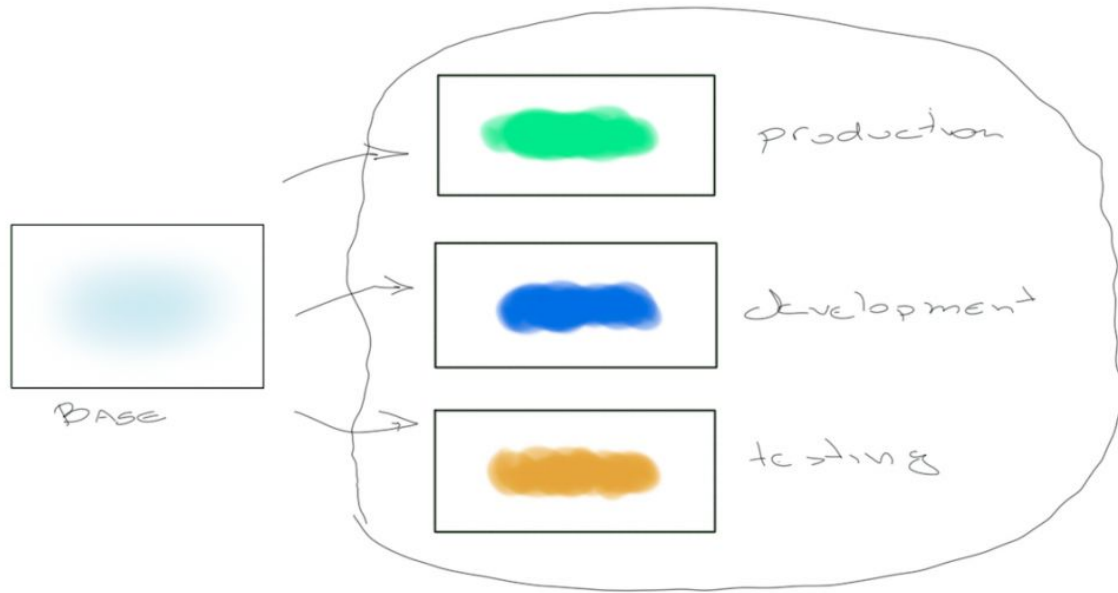
O que é Kustomize



TL;DR

Kustomize utiliza manifestos de Kubernetes para adicionar, remover ou atualizar opções de configuração sem fork. Ele está disponível como um binário autônomo e como um recurso nativo do kubectl.

OVERLAYS



A large, stylized blue arrow graphic pointing to the right, composed of multiple parallel lines of varying shades of blue, creating a sense of depth and movement.

Por que usar Kustomize?

1. Evitar repetição de código;
2. Nativo - desde kubectl 1.14;
3. É muito fácil de usar e compreender;
4. Altere arquivos de manifesto sem bifurcação;
5. Use várias referências de códigos-fonte em um único lugar;
6. Implante novos clusters em tempo recorde;
7. Manter vários ambientes sincronizados;





Kustomize Features

The Kustomization file

File structure:

```
~/someApp
├── deployment.yaml
├── kustomization.yaml
└── service.yaml
```

base: kustomization + resources

kustomization.yaml

```
commonLabels:
  app: myWord
resources:
- deployment.yaml
- service.yaml
configMapGenerator:
- name: wordpress-map
  files:
  - env.startup.txt
```

deployment.yaml

```
apiVersion: v1
kind: Deployment
metadata:
  name: wordpress
labels:
  app: wordpress
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wordpress
  template: ...
```

service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress
spec:
  ports:
  - port: 389
  selector:
    app: wordpress
```

The Kustomization file

File structure:

```
~/someApp
├── base
│   ├── deployment.yaml
│   ├── kustomization.yaml
│   └── service.yaml
└── overlays
    ├── development
    │   ├── cpu_count.yaml
    │   ├── kustomization.yaml
    │   └── replica_count.yaml
    └── production
        ├── cpu_count.yaml
        ├── kustomization.yaml
        └── replica_count.yaml
```

overlay: **kustomization** + **patches** + more resources
(referencing a base)

kustomization.yaml

```
namePrefix: prod-
commonLabels:
  variant: prod
commonAnnotations:
  note: Hello, I am production!
bases:
- ../../base
patches:
- replica_count.yaml
- cpu_count.yaml
```

replica_count.yaml

```
apiVersion: v1
kind: Deployment
metadata:
  name: wordpress
spec:
  replicas: 80
```

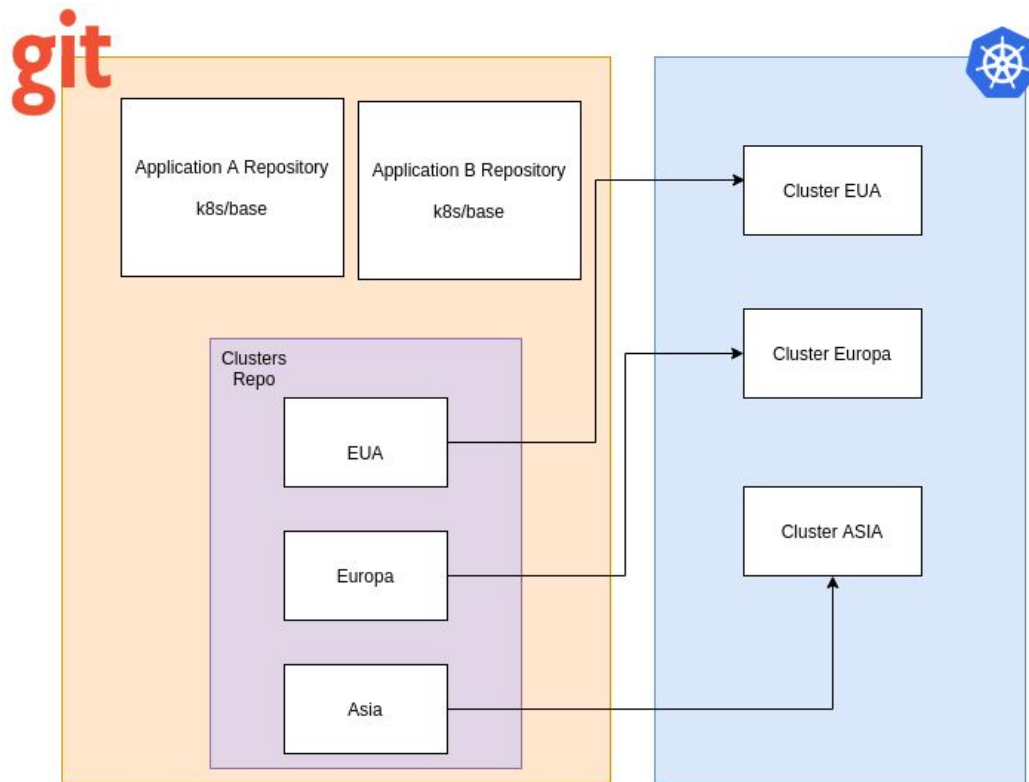
cpu_count.yaml

```
apiVersion: v1
kind: Deployment
metadata:
  name: wordpress
spec:
  template:
    spec:
      containers:
        - name: my-container
          resources:
            limits:
              cpu: 7000m
```

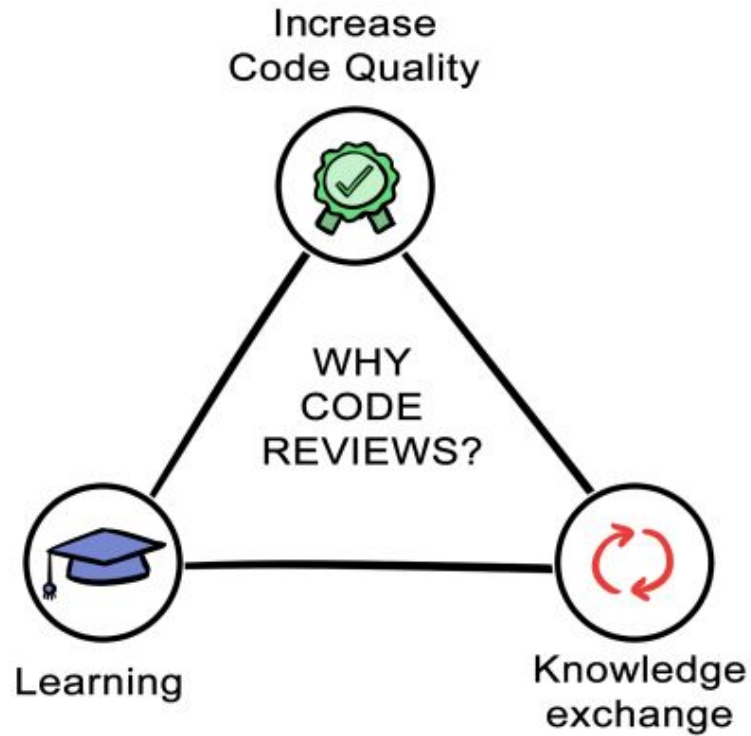
Kustomize Features

- Resources
- Bases
- Remote builds
- Generator
- Patch
- Prefix/Suffix
- Common Labels
- Transformers

Multiplos Clusters



Melhor prevenir/revisar que remediar



Questions?



Thank you

Yros Aguiar
SRE
yros.aguiar@pipefy.com

pipefy

› pipefy.com