

Managing Thousands of Clusters & Their Workloads with Flux

Max Jonas Werner

Kubernetes Engineer & Flux Engineer

2 Mar 2022



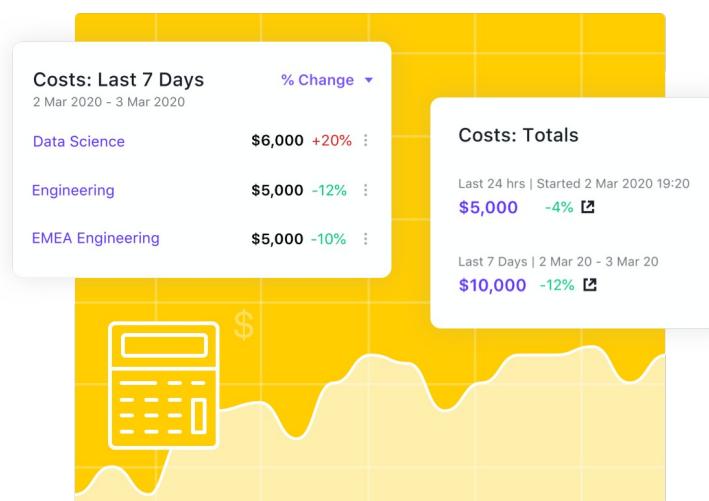
Introduction

What is Kommander?

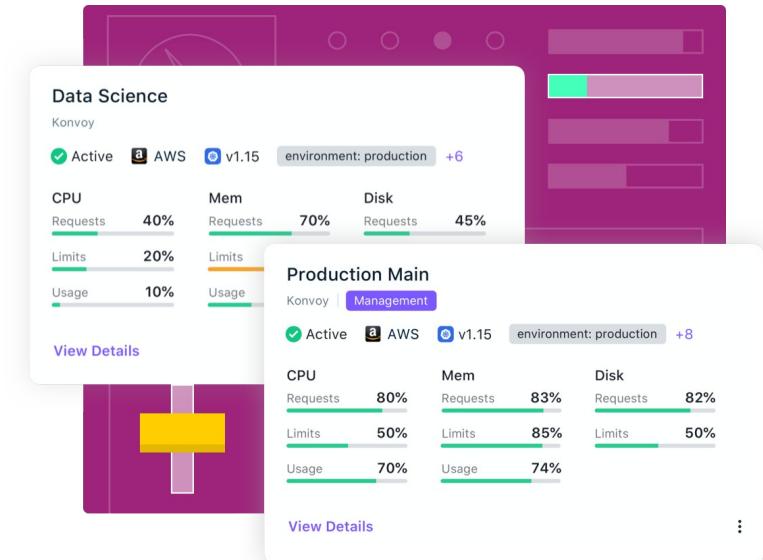
Manage applications and infrastructure



real-time cost management



centralized observability



Introduction

What is Kommander?

The screenshot shows the Kommander dashboard interface. At the top, there's a navigation bar with a menu icon, the text "D2 IQ Kommander", and a "Beta" badge. The left sidebar contains links for Dashboard, Projects, Clusters, Catalog, Istio Mesh, Administration (with sub-links for Identity Providers, Cloud Providers, Access Control, and Licensing), and Support (with sub-links for Documentation and Getting Started). The main content area is titled "Dashboard". It features several key metrics:

- Clusters:** 20 Total Clusters, 0 Clusters with Warnings
- Projects:** 10 Total Projects
- Pods Allocated:** 35% (350 of 1,000)

Below these metrics is a section titled "Centralized Monitoring & Alerts" with the sub-instruction "Monitor metrics and manage alerts across all clusters in Kommander." It includes icons for "Monitoring Grafana v6.3.5" and another unlabelled icon.

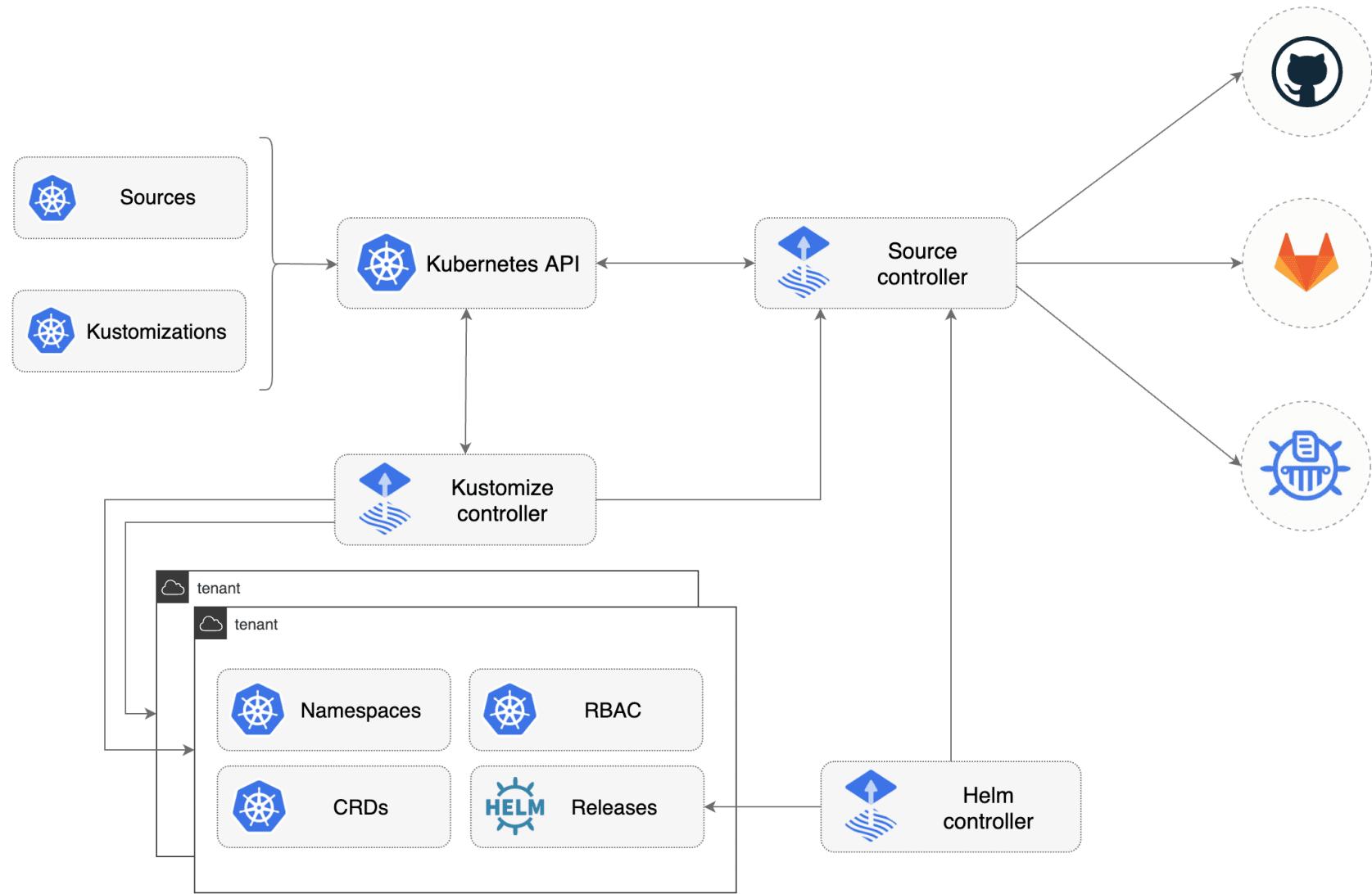
Introduction

Before Flux came along

kubeaddons + kubefed

Introduction

Along came Flux



Introduction

Along came Flux

```
$ flux install
$ flux create source git podinfo \
  --url=https://github.com/stefanprodan/podinfo \
  --branch=master --interval=10m
$ flux create kustomization common \
  --source=GitRepository/podinfo \
  --path=deploy/webapp/common
$ flux create kustomization backend \
  --source=GitRepository/podinfo \
  --path=deploy/webapp/backend --depends-on common
$ kubectl -n webapp get deploy -w
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
backend   0/1     1           0           11s
backend   1/1     1           0           21s
```

Introduction

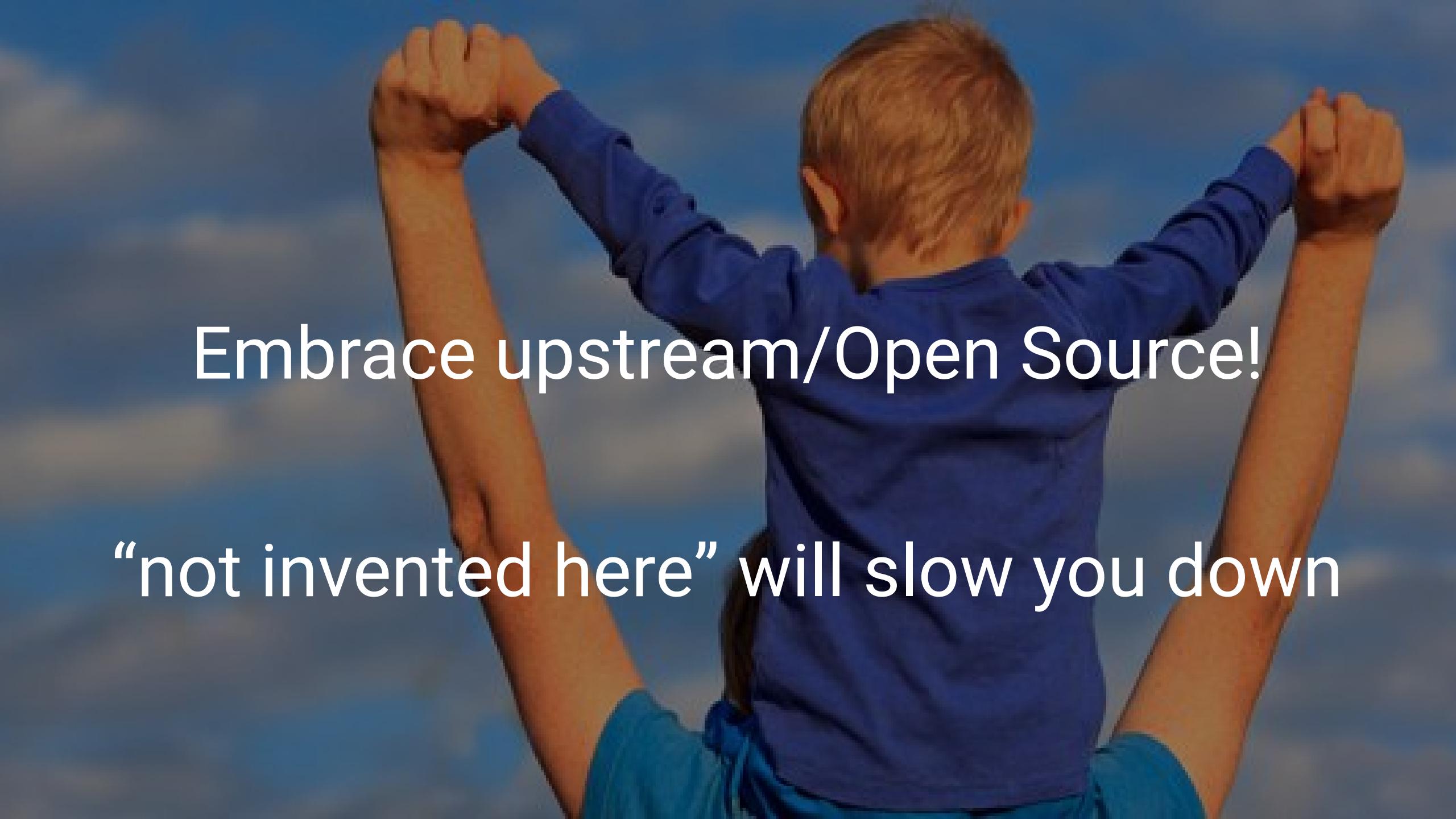
Why Flux now?

Provide users with benefits of GitOps workflow that they already successfully apply:

- central log of which applications have been enabled/disabled on which cluster
- central storage of application configuration
- employing well-known practices like code review
- simple way to roll back changes

Demo time

Takeaways?!

A photograph of a person from behind, wearing a blue long-sleeved shirt. They are flexing their right arm, showing a large bicep. Their left arm is also raised, with the hand near their head. The background is a cloudy, blue-grey sky.

Embrace upstream/Open Source!

“not invented here” will slow you down

Keep an ear to the ground!



Custom CLIs for tailored user experience!



Experiment!



Reach out!

Email:

max@e13.dev

Twitter:

@makkes

CNCF Slack:

#flux

Kubernetes Slack:

#sig-multicloud

Get hired!

<https://d2iq.com/careers>



Fin!

Multi-cluster Application Management

Challenges

- Deploy same app multiple times
- Deploy different versions of same app to different clusters
- Dynamically inject values to Flux-managed resources

Multi-cluster Application Management

Challenge: Deploy same app multiple times

AppDeployment "my-metallb":

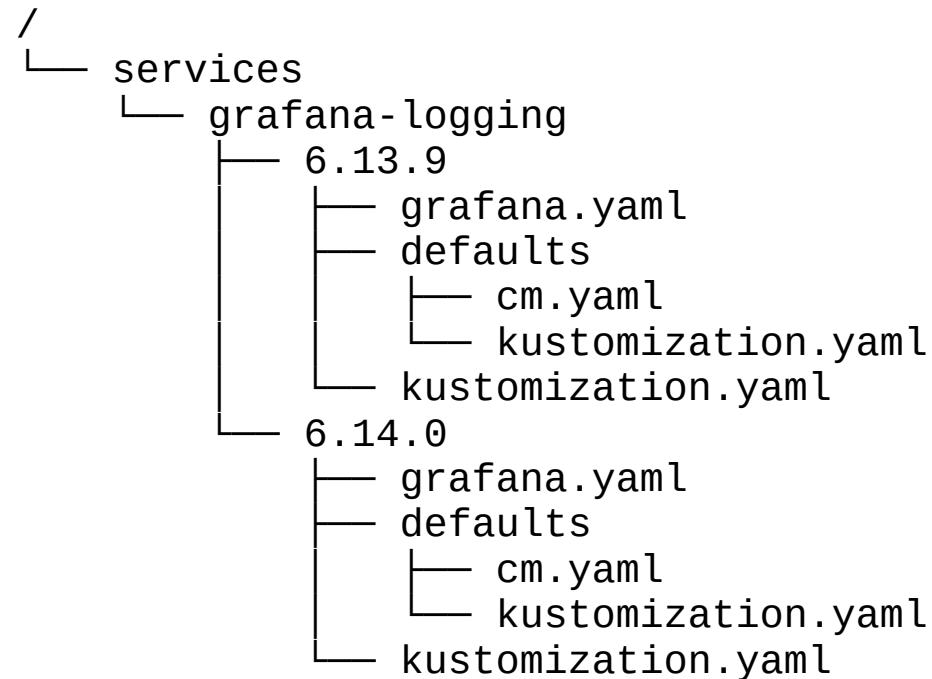
```
patchesJson6902:  
- patch:  
  - op: replace  
    path: /metadata/name  
    value: my-metallb  
target:  
  kind: HelmRelease  
  name: metallb
```

AppDeployment "my-metallb-2":

```
patchesJson6902:  
- patch:  
  - op: replace  
    path: /metadata/name  
    value: my-metallb-2  
target:  
  kind: HelmRelease  
  name: metallb
```

Multi-cluster Application Management

Challenge: Deploy different versions of same app to different clusters



Multi-cluster Application Management

Challenge: Dynamically inject values to Flux-managed resources

```
apiVersion:  
helm.toolkit.fluxcd.io/v2beta1  
kind: HelmRelease  
metadata:  
  name: kubefed  
  namespace: ${releaseNamespace}
```

```
apiVersion:  
kustomize.toolkit.fluxcd.io/v1beta1  
kind: Kustomization  
[...]  
  postBuild:  
    substituteFrom:  
      - kind: ConfigMap  
        name: substitution-vars
```

```
  apiVersion: v1  
  kind: ConfigMap  
  data:  
    releaseNamespace: foo  
[...]
```

Automate all the things!

Don't be afraid of writing controllers
that contribute to a Git repository!

The Future

- Use Flux-native resources to surface Application status
 - Kustomization health checks
 - declarative health checking
 - no additional code needed
- Our mentality is upstream-first
- Multiple Git repositories for better tenant/cluster isolation
- Remote Kustomization application (with `.spec.kubeConfig`)