RUD: Rapid Unscheduled Deletion of Critical Infrastructure Workloads

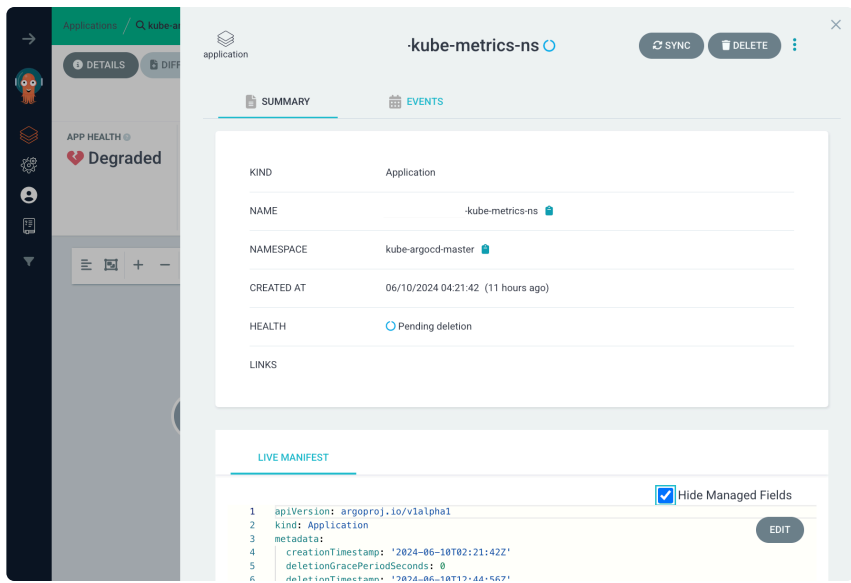
Kubernetes Community Days Zürich - 13th of June 2024

Clément Nussbaumer

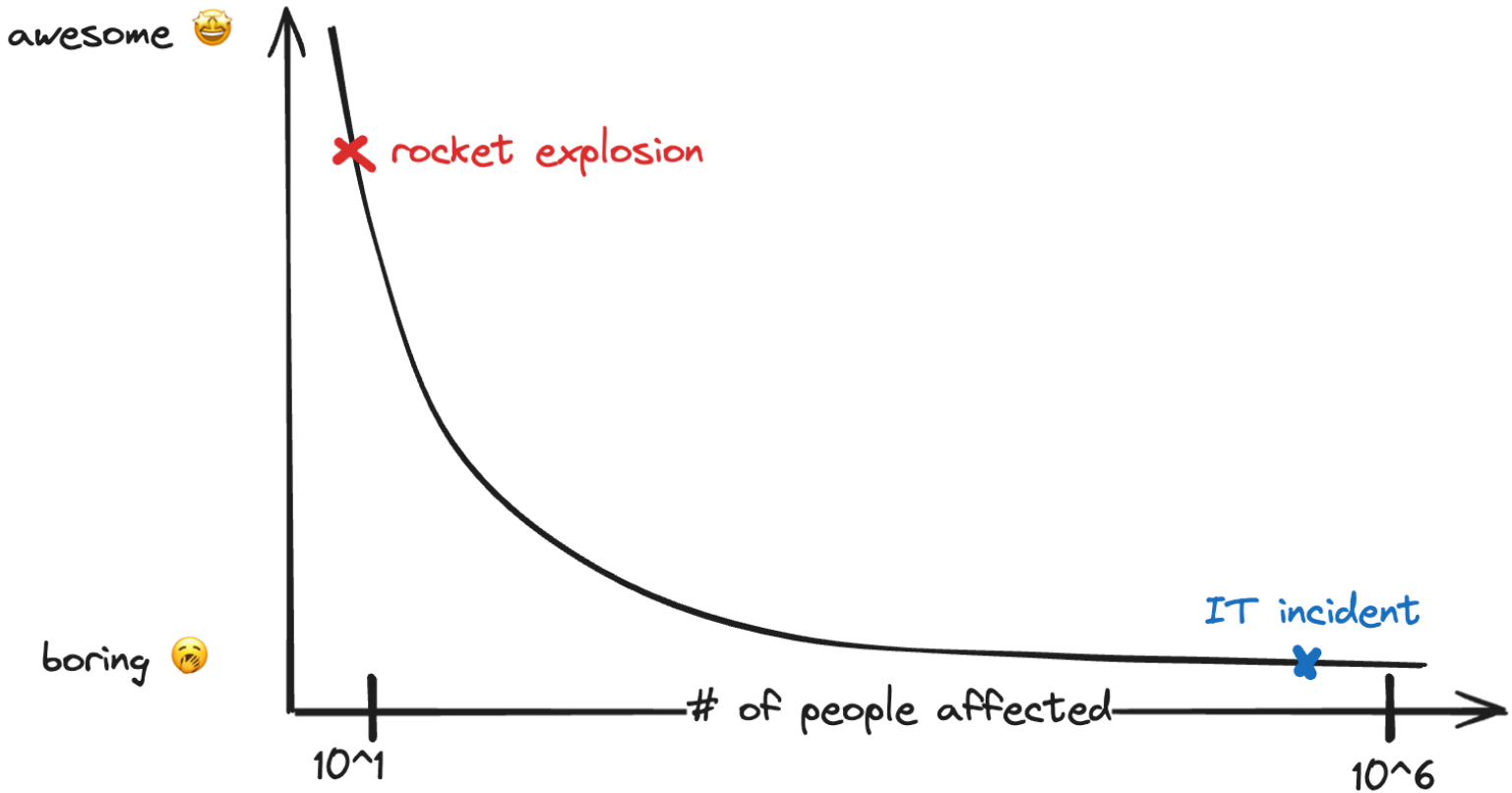
RUD - Rapid Unscheduled ...

Disassembly 🚀 💥

Deletion of Critical Infrastructure Workload 🤭 ⏳



Awesomeness scale



Agenda:

1. Kubernetes @ PostFinance
2. Infrastructure Workloads
3. ArgoCD ApplicationsSet
4. The RUD Incident

Kubernetes @ PostFinance

Some figures:

- **1406** namespaces across **30** Kubernetes clusters
- **18'000** pods, **4200** ingresses
- **450 nodes** (24 CPU / 64 GiB RAM)
  - 14 GPU nodes for machine learning workloads
- **4000 RPS** peak load across all ingresses
- **400** card transaction per second

Some facts:

- **Vanilla v1.28 Kubernetes**
  - Oldest cluster will soon turn **5 years old**
- **Self-service repository + controller** to manage all namespaces
- **ArgoCD** and GitOps for almost all deployments
- **Chaos Monkey** on all clusters 🐒

Most critical workload: **card payment services** 💳 💰

Infastructure Workloads

A little overview

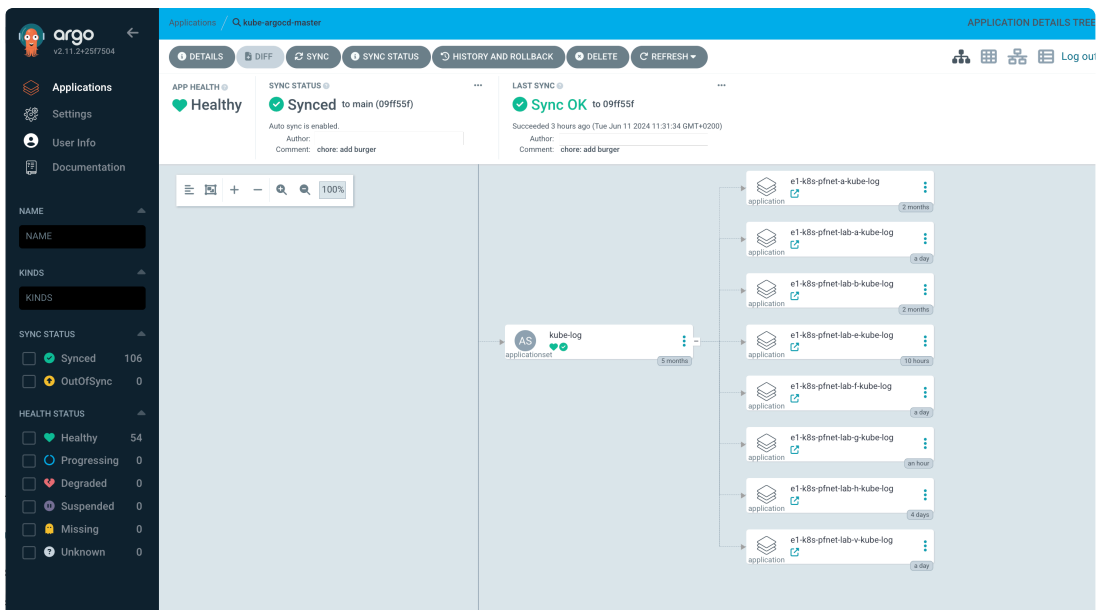
```
> kubectl get namespace kube-metrics
NAME           STATUS   AGE
kube-metrics   Active   215d

> kubectl --namespace=kube-metrics get deployments.apps
NAME                 READY   UP-TO-DATE   AVAILABLE   AGE
kube-state-metrics   1/1     1            1           215d
metrics-server       1/1     1            1           215d

> kubectl --namespace=kube-metrics get daemonsets.apps
NAME            DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
node-exporter   103       103       103     103          103         kubernetes.io/os=linux   215d

> kubectl get apiservice v1beta1.metrics.k8s.io
NAME                     SERVICE                        AVAILABLE   AGE
v1beta1.metrics.k8s.io   kube-metrics/metrics-server   True        216d
```

Infastructure Workloads

How to deploy them on all clusters ?

Up until September 2022: combination of `ansible`, `helm`, and GitLab pipelines → 🐢

After September 2022: migration to ArgoCD `ApplicationSets` → 🐇



ArgoCD Application

What is it ?

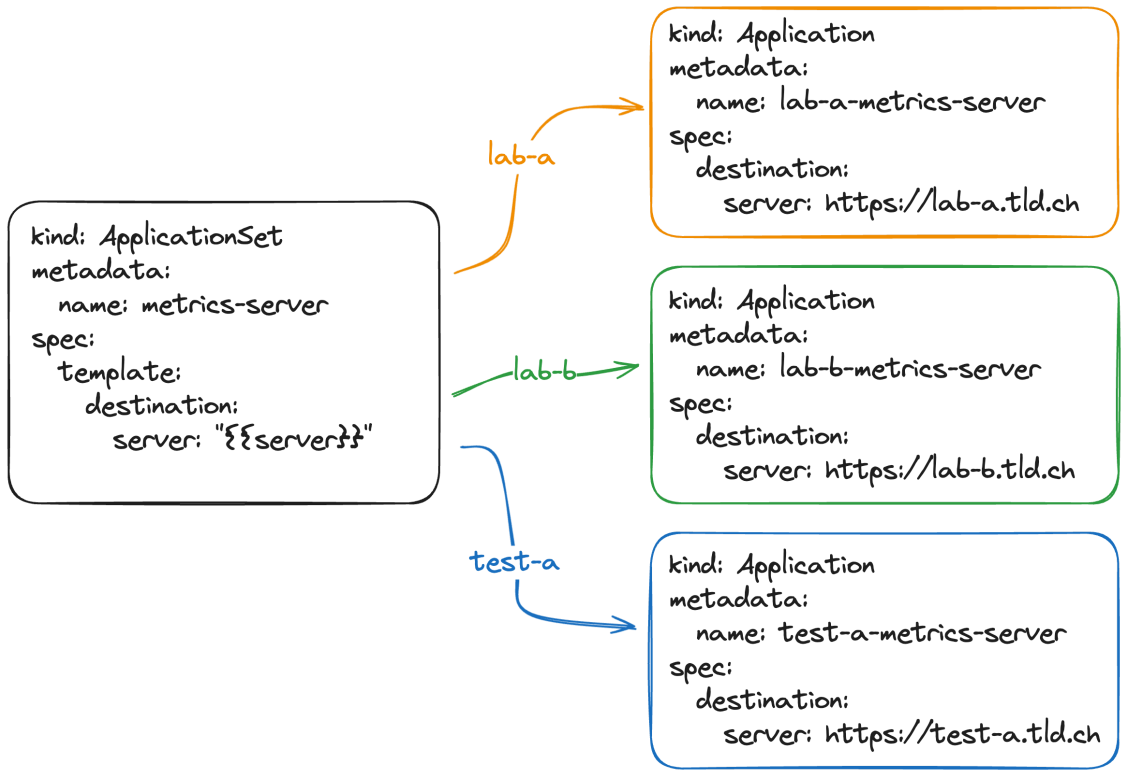
```
> kubectl explain application.spec
GROUP:      argoproj.io
KIND:       Application
VERSION:    v1alpha1

FIELDS:
  destination   <Object> -required-
    Destination is a reference to the target Kubernetes server and namespace

  source        <Object>
    Source is a reference to the location of the application's manifests or
    chart

  syncPolicy    <Object>
    SyncPolicy controls when and how a sync will be performed
```

ArgoCD ApplicationSet

What is it ?

```
> kubectl explain applicationset.spec
GROUP:      argoproj.io
KIND:       ApplicationSet
VERSION:    v1alpha1

FIELDS:
  generators    <[]Object> -required-
    <no description>

  syncPolicy    <Object>
    <no description>

  template      <Object> -required-
    <no description>
```

ArgoCD ApplicationSet



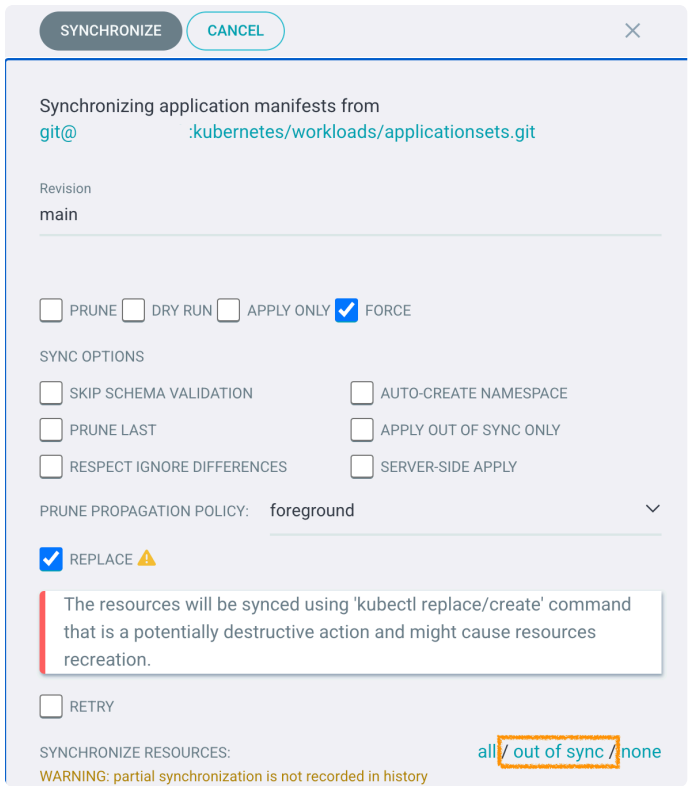
ArgoCD ApplicationSet

`metrics-server` example

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: kube-metrics-metrics-server
spec:
  generators:
  - clusters:
      selector:
        matchLabels:
          stage: lab
      values:
        revision: HEAD
  - clusters:
      selector:
        matchLabels:
          stage: test
      values:
        revision: v0.1.4
```

Rapid Unscheduled Deletion Incident

November 8th 2023 - 16:35

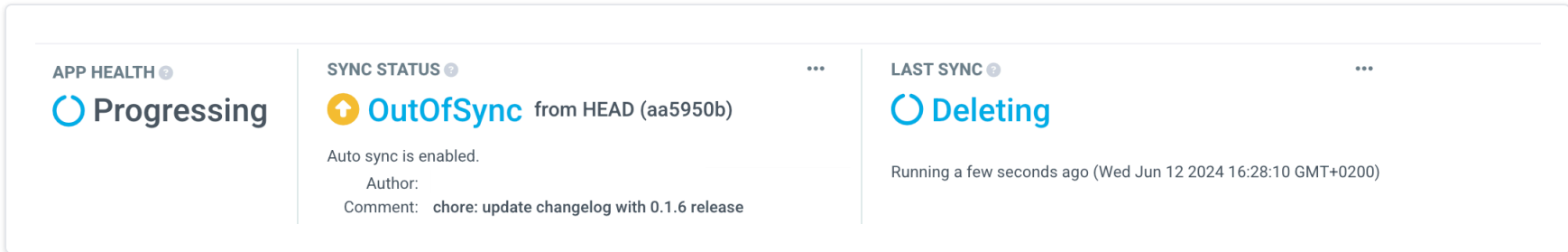


`out of sync` ApplicationSets:

- `kube-contour`
- `kube-log`
- `kube-metrics-ns`
- `kube-metrics-server`
- `kube-otel`

→ all got replaced 💣

Unable to delete namespaces

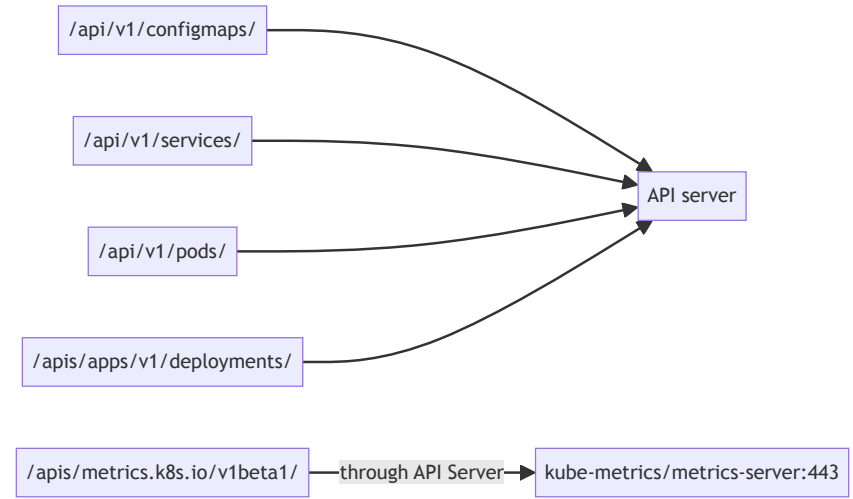



```
> kubectl delete namespace kube-metrics --timeout=10s
namespace "kube-metrics" deleted
error: timed out waiting for the condition on namespaces/kube-metrics

> kubectl get namespace kube-metrics -o=yaml | grep -e "^status" -A8
status:
  conditions:
  - lastTransitionTime: "2024-06-11T17:20:17Z"
    message: 'Discovery failed for some groups, 1 failing: unable to retrieve the
      complete list of server APIs: metrics.k8s.io/v1beta1: stale GroupVersion discovery:
      metrics.k8s.io/v1beta1'
    reason: DiscoveryFailed
    status: "True"
    type: NamespaceDeletionDiscoveryFailure
```

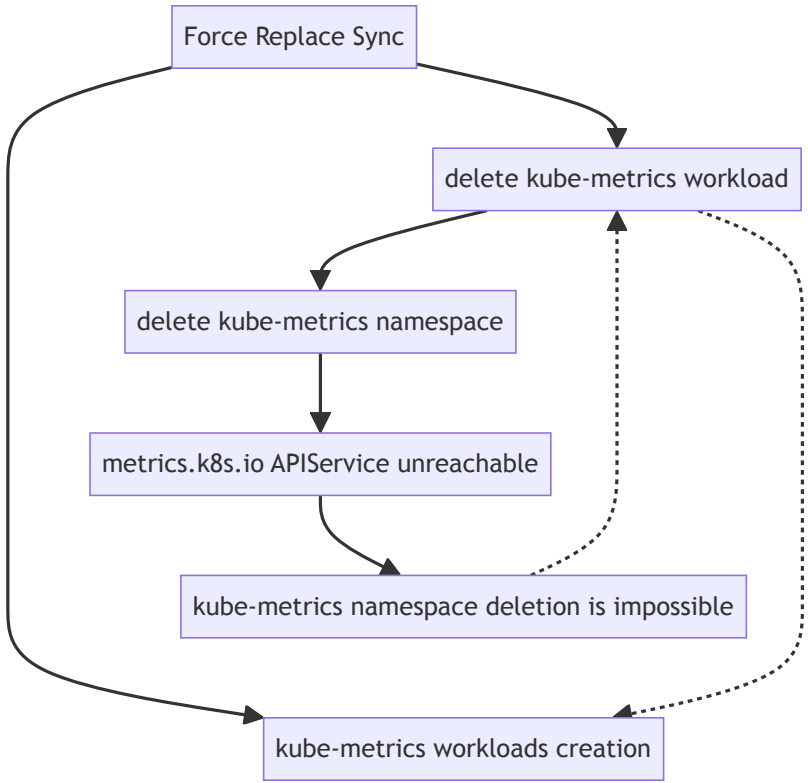
`metrics.k8s.io` API Service

```yaml
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  name: v1beta1.metrics.k8s.io
spec:
  group: metrics.k8s.io
  service:
    name: metrics-server
    namespace: kube-metrics
    port: 443
  version: v1beta1
```

```
> kubectl get --raw /apis/metrics.k8s.io/v1beta1/nodes/
{
  "kind": "NodeMetricsList",
  "apiVersion": "metrics.k8s.io/v1beta1",
  "metadata": {},
  "items": [
    ...
```



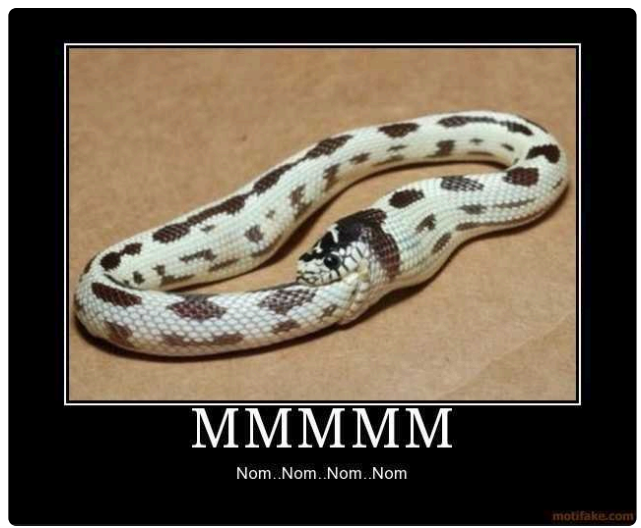
`metrics.k8s.io` circular dependency



Consequences

all deleted namespaces stay in `Terminating` state indefinitely

`kube-contour`, `kube-log`, etc. are stuck



snake biting its tail

`metrics.k8s.io` circular dependency

How to break it ?

```
> kubectl foreach '/^p1-k8s-.*/' -- delete apiservice v1beta1.metrics.k8s.io
Will run command in context(s):
  - p1-k8s-cluster-a
  - p1-k8s-cluster-b
  - p1-k8s-cluster-c
  - p1-k8s-cluster-d
  - p1-k8s-cluster-e
Continue? [Y/n]: Y
p1-k8s-cluster-a | apiservice.apiregistration.k8s.io "v1beta1.metrics.k8s.io" deleted
p1-k8s-cluster-b | apiservice.apiregistration.k8s.io "v1beta1.metrics.k8s.io" deleted
p1-k8s-cluster-c | apiservice.apiregistration.k8s.io "v1beta1.metrics.k8s.io" deleted
p1-k8s-cluster-d | apiservice.apiregistration.k8s.io "v1beta1.metrics.k8s.io" deleted
p1-k8s-cluster-e | apiservice.apiregistration.k8s.io "v1beta1.metrics.k8s.io" deleted
```

Safeguards

How do we prevent this from happening again ?

1. Split ArgoCD instances (nonprod / prod)
2. `syncPolicy` change on critical `ApplicationSet`

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: kube-metrics-metrics-server
spec:
  syncPolicy:
    preserveResourcesOnDeletion: true
  template:
    metadata:
      name: "{{server}}-metrics-server"
      namespace: kube-metrics
    spec:
      ...
```

Considerations ⚠

Which other circular dependencies ?

- Any other API Service extension
- Kubernetes resources with a finalizer
  - `PersistentVolumes`
  - `CustomResourceDefinition`
  - ...
- `ValidatingWebhook`
- `MutatingWebhook`

GitOps saved the day

Once the circular dependencies were broken

all workloads were back in one minute

Demo


Questions ?