KUBERNETES MIT IAC
GITOPS MIT FLUX CI/CD

makandra

AGENDA

1. Vorstellung
2. Die Herausforderung: Kubernetes verwalten
3. Lösungsidee: GitOps und Infrastructure as Code
4. Lösungsmöglichkeit: Flux CI/CD
5. Zusammenfassung und Ausblick
6. Fragen und Antworten

WHOAMI

Emma Heinle

- Senior Project Lead Operations bei makandra
- Wir entwickeln und betreiben Webanwendungen On Premise und in der Cloud.
- Wir betreiben auch gern Eure Anwendungen oder unterstützen Euer (Dev)Ops-Team.

makandra

KUBERNETES

Ganz kurz:

- Kubernetes ist ein Tool zum Verwalten und Deployen von Containern.
- Kubernetes wird für gewöhnlich in Clustern betrieben.
- Cluster bestehen aus beliebig vielen Nodes.

DIE HERAUSFORDERUNG

Kubernetes ist unglaublich mächtig.

Alles ist YAML. Berufsbezeichnung könnte auch YAML-Engineer sein.

Man fängt an mit `kubectl apply -f ./meine-coole-anwendung.yml`.

Irgendwann hat man dann `./meine-coole-anwendung_kopie.bak.final_2_1_staging.yml` und viele weitere daneben.

- Welche davon ist jetzt gerade live? Und in welcher Umgebung? Dev? Prod?
- Macht große Probleme wenn man Multiplayer Kubernetes spielt.
- Wenn plötzlich mal alles kaputt ist, wird wiederherstellen anspruchsvoll.
- Problem: Dev läuft gut, aber Prod scheitert an Copy+Paste-Fehlern.

LÖSUNGSIDEE: GIT

💡 Wir nehmen einfach Git!

GITOPS UND INFRASTRUCTURE AS CODE

Wir tragen unsere Einstellungen in ein Git-Repository ein. Das bringt:

- Versionierung
- Konsistenz
- Wiederholbarkeit
- Kollaboration

Dann haben wir unsere *Infrastructure as Code*!

Wir machen jetzt nichts mehr auf der Kommandozeile mit `kubectl`.

Alles kommt ins Git.

Flux setzt das dann für uns um.

Aber wie?

KURZER EXKURS: GRUNDLAGEN

DESIRED STATE

Kubernetes-Manifeste sind *deklarativ* und beschreiben einen *desired state*,
sie beschreiben keine Aktionen sondern einen gewünschten Zustand!

HELM

Eine Anwendung bei Kubernetes zu betreiben hat viele Komponenten. Load Balancer, Container, Konfigurationseinstellungen, etc. Den genauen Bedarf weiß die Entwicklung.

Dev stellt eine Menge YAML-Manifeste bereit, als Templates.

Helm akzeptiert Parameter für die Anwendungsinstallation und rendert die Templates.

CONTROLLER

Ein Controller ist ein Pod im Cluster, der Änderungen am Cluster für uns umsetzt.

Dazu spricht er mit der Kubernetes API.

RECONCILIATION-LOOP

In einem Reconciliation-Loop wird regelmäßig der gewünschte Zustand mit der aktuellen Realität im Cluster verglichen. Falls notwendig, werden Änderungen am Cluster gemacht, bis die Realität dem entspricht, was im gewünschten Zustand steht.

LÖSUNGSMÖGLICHKEIT: FLUX CI/CD

Flux CI/CD kann sowas.

- Synchronisiert Git-Repo(s) mit Kubernetes-Stand.
- Freie Software (Apache License 2)
- Mature
- Cloud Native Foundation Graduated Project

FEATURES

Verschiedene Controller setzen Desired State um:

- Source Controller checkt Git aus
- Kustomize Controller setzt Manifeste um
- Helm Controller verwendet helm-Repos und Releases
- Notification Controller schickt Nachrichten

CUSTOM RESOURCE DEFINITIONS

Wir bekommen ein paar neue Ressourcentypen bei Kubernetes. Damit können wir den Controllern deklarativ sagen, wie unser Cluster aussehen soll.

CRD: GitRepository

Wir machen erst ein Flux Bootstrap in den Cluster, das ist noch imperativ.

```
$ export GITHUB_TOKEN=geheimertoken
$ flux bootstrap github \
  --owner eheinle-mak
  --repository flux
  --branch=main \
  --path=./clusters/cluster1 \
  --personal
```

```
► connecting to github.com
► cloning branch "main" from Git repository "https://github.com/eheinle-mak/flux.git"
✓ cloned repository
✓ committed component manifests to "main" ("433fed4ba6cd4d5552d28f7b87734f151a23a6a7")
► pushing component manifests to "https://github.com/eheinle-mak/flux.git"
► installing components in "flux-system" namespace
✓ reconciled components
✓ public key: ecdsa-sha2-nistp384 AAAAE…
✓ configured deploy key "flux-system-main-flux-system-./clusters/cluster1" for "https://github.com/eheinle-mak/flux"
✓ generated sync manifests
✓ committed sync manifests to "main" ("c04d6ffd5b203e8609bdc265e8b890c42788a955")
► pushing sync manifests to "https://github.com/eheinle-mak/flux.git"
◎ waiting for GitRepository "flux-system/flux-system" to be reconciled
✓ GitRepository reconciled successfully
◎ waiting for Kustomization "flux-system/flux-system" to be reconciled
✓ Kustomization reconciled successfully
► confirming components are healthy
✓ helm-controller: deployment ready
✓ kustomize-controller: deployment ready
✓ notification-controller: deployment ready
✓ source-controller: deployment ready
✓ all components are healthy
```

CRD: GITREPOSITORY

WAS IST JETZT IM GIT?

```yaml
apiVersion: source.toolkit.fluxcd.io/v1
kind: GitRepository
metadata:
  name: flux-system
  namespace: flux-system
spec:
  interval: 1m0s
  ref:
    branch: main
  secretRef:
    name: flux-system
  url: ssh://git@github.com/eheinle-mak/flux
```

CRD: KUSTOMIZATION

Wir legen im `clusters/cluster1`-Verzeichnis eine Kustomization an:

```yaml
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: secret-operator
spec:
  interval: 15m
  path: "app/secret-operator"
  targetNamespace: default
  sourceRef:
    kind: GitRepository
    name: flux-system
    namespace: flux-system
```

CRD: HELMRELEASE

Wir könnten schon unsere gewöhnlichen Manifeste hier rein legen. Oder Helm nutzen für aufwändigere Deployments von Setups:

```yaml
---
apiVersion: helm.toolkit.fluxcd.io/v2beta2
kind: HelmRelease
metadata:
  name: secret-operator
spec:
  chart:
    spec:
      chart: kubernetes-secret-generator
      version: v3.4.0
      sourceRef:
        kind: HelmRepository
        name: secret-operator
  interval: 15m
  timeout: 5m
  values:
    secretLength: 12
```

WAS PASSIERT?

```
$ flux logs
2025-04-26T06:22:03.127Z info GitRepository/flux-system.flux-system - stored artifact for commit 'feat(apps): install secret-operator'
$ kubectl get Kustomization
NAME              AGE     READY   STATUS
secret-operator   2m11s   True    Applied revision: main@sha1:e08db22cb0dbcff088f0655e783049a00a6ccc17
$ kubectl get HelmRelease/secret-operator deployment/secret-operator-kubernetes-secret-generator
NAME                                                    AGE    READY   STATUS
helmrelease.helm.toolkit.fluxcd.io/secret-operator   2m12s     True    Helm upgrade succeeded for release[…]
```

KLAPPT DAS DENN?

```
$ kubectl get deployment/secret-operator-kubernetes-secret-generator
NAME                                                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/secret-operator-kubernetes-secret-generator   1/1     1            1           3m0s
$ kubectl get pods
NAME                                                         READY   STATUS    RESTARTS   AGE
secret-operator-kubernetes-secret-generator-5d4b599b96-fx92s   1/1     Running   0          3m42s
```

makandra

UND WENN ICH WAS ÄNDERE?

Dann passen wir die `kustomization` an:

```
diff --git a/app/secret-operator/helm.yaml b/app/secret-operator/helm.yaml
index 9265d81..172034f 100644
--- a/app/secret-operator/helm.yaml
+++ b/app/secret-operator/helm.yaml
@@ -25,4 +25,4 @@ spec:
   interval: 15m
   timeout: 5m
   values:
-    secretLength: 12
+    secretLength: ${SECRET_LENGTH}
diff --git a/clusters/cluster1/kustomization-secretoperator.yaml b/clusters/cluster1/kustomization-secretoperator.yaml
index 0a7b6c8..cd17f9e 100644
--- a/clusters/cluster1/kustomization-secretoperator.yaml
+++ b/clusters/cluster1/kustomization-secretoperator.yaml
@@ -13,3 +13,6 @@ spec:
     kind: GitRepository
     name: flux-system
     namespace: flux-system
+  postBuild:
+    substitute:
+      SECRET_LENGTH: "18"
```

JA ABER WAS IST MIT GEHEIMNISSEN?

- Secrets niemals unverschlüsselt im Git einchecken, wenn sie geheim bleiben sollen.
- Nach Möglichkeit kann man die Secrets auch mit einem eigenen Dienst wie Vault bereitstellen.
- Weniger aufwändig und auch im Git: Sops
- Secrets mit z.B. GPG oder Age verschlüsselt speichern.

BEISPIEL

```
apiVersion: v1
data:
    foo: ENC[AES256_GCM,data:N+N3PA==,iv:0OCHCKxmOf6XHSQFb0TtsGpMCvvOp2cj1hCqJaSptw8=,tag:XppQFTV68jEek3SdZ7dHtw==,type:str]
kind: Secret
metadata:
    name: secret-example
sops:
    age:
        - recipient: age10w6kksnwrkpra8nf3mwkcwg4ltmc5qksf9ra7j858g3f5luncstqk3cska
          enc: |
            -----BEGIN AGE ENCRYPTED FILE-----
            YWdlLWVuY3J5cHRpb24ub3JnL3YxCi0+IFgyNTUxOSBSWTFsWEZrM1Q4b3dMVG9Q
            ZklOTlNTUHZDQnpoTThFaFg0dit4SzhzNlEwCng1QWdkNHI5N202UFN2cVFiY01v
            ZWlON0JWakxXaHVCSUh0RFlmV04zZGsKLS0tIDVLdEJ4YlQ1a3l4U3hFQk0rTFF3
            SmE4Vk44YkFHQW5WTEpkclpTa0dndGsKLbSCdquRhEvL6+y7Uu0Cg3g0LcdubY6f
            OQWUSvWnAiqRWoop0uzlxJW/44ciZfNGDToYZXfyA7Kl0aGF7h1A5Q==
            -----END AGE ENCRYPTED FILE-----
```

ZUSAMMENFASSUNG

- Mit Git macht's mehr Spaß, weil alles Struktur und Nachvollziehbarkeit bekommt.
- Flux ist eine Anwendung im Cluster, die den Cluster verwaltet.
- Flux arbeitet mit vielen Controllern, welche die notwendigen Aufgaben umsetzen.

AUSBLICK: MULTIPLAYER KUBERNETES

- Arbeit im Team an Kubernetes-Konfiguration sollte konfliktarm laufen.
- Verschiedene Abhängigkeiten lassen sich einbinden. Helm-Repos oder Weitere Git-Repos können referenziert werden, die z.B. vom Dev-Team der Anwendung selbst verwaltet werden.
- Workflows für Merges wie "Tests müssen grün sein", "Linter ist zufrieden" etc lassen sich auch abbilden, ehe man was an den Cluster schickt.
- Berechtigungen können in den bestehenden Git-Tools verwaltet werden.
- Mehrere Cluster verwalten in einem Repo wird möglich - Code wiederverwenden statt Copy and Paste.

VIELEN DANK FÜR DIE AUFMERKSAMKEIT

FRAGEN UND ANTWORTEN

makandra

NOCH MEHR FRAGEN?

- makandra bietet DevOps as a Service.
- makandra hat ein Traineeprogramm DevOps.

makandra