Operator in a Nutshell

Norris Sam Osarenkhoe, DevOps Architect

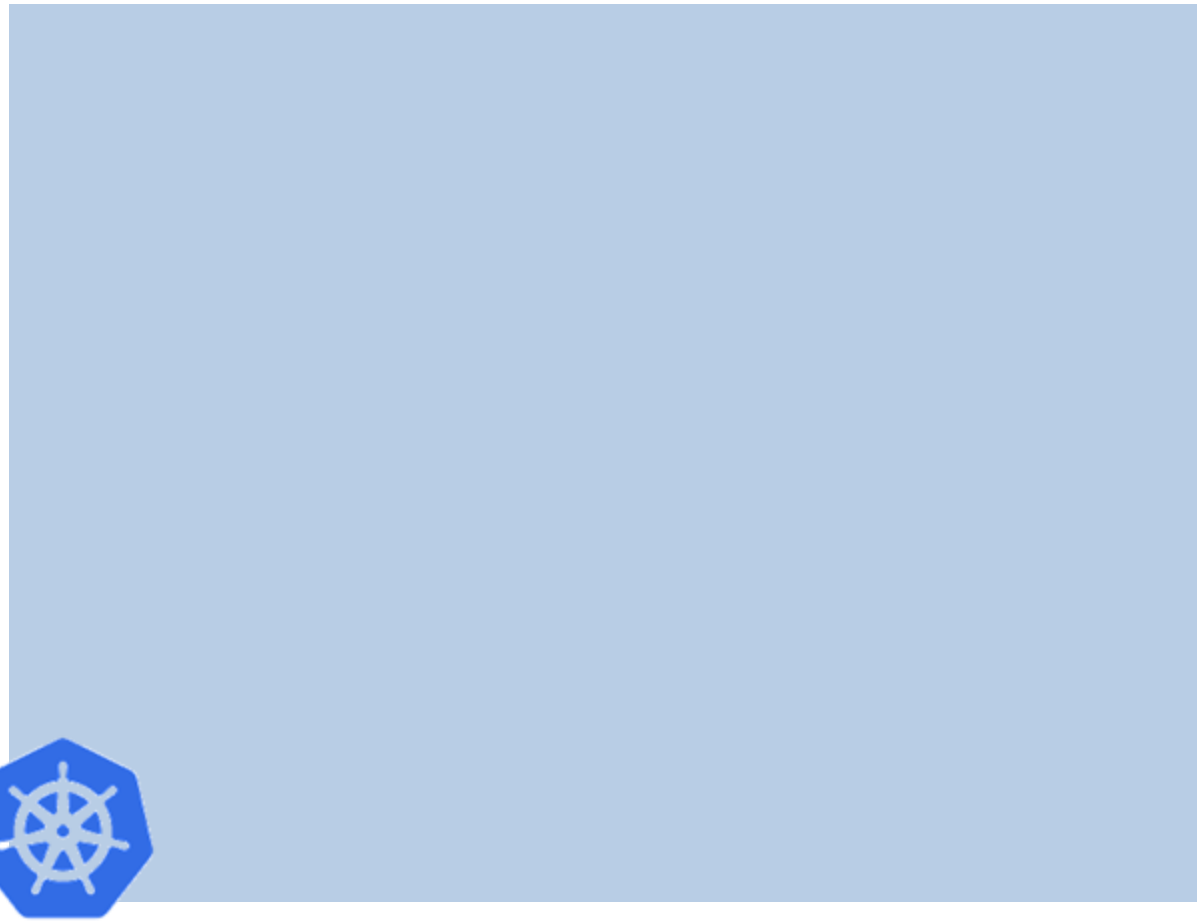
SVA

Die Kubernetes Journey

Gut Ding will Weile haben

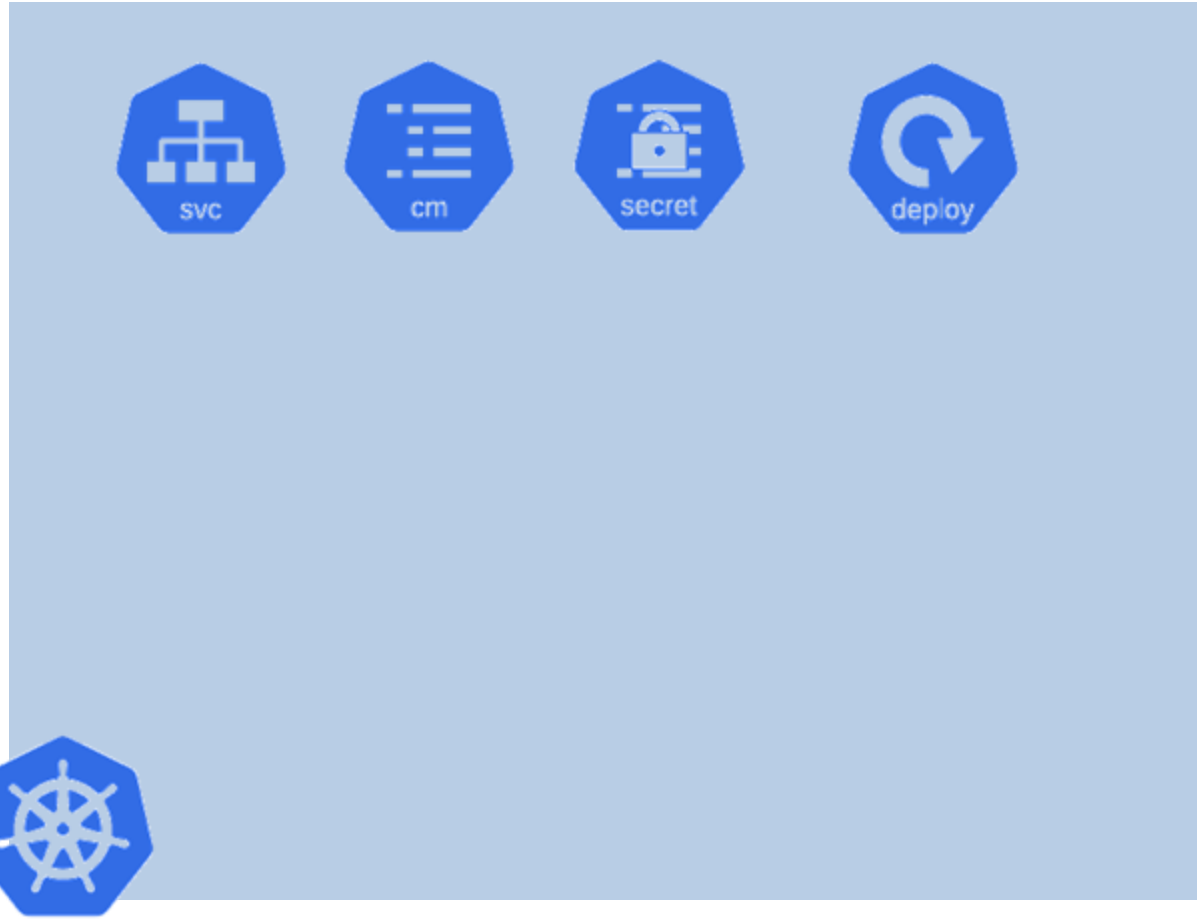
SVA

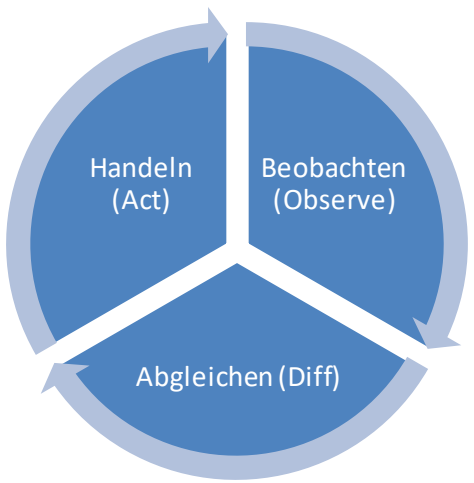
/ Phase 1: Auf geht‘s!



meine-api-v1

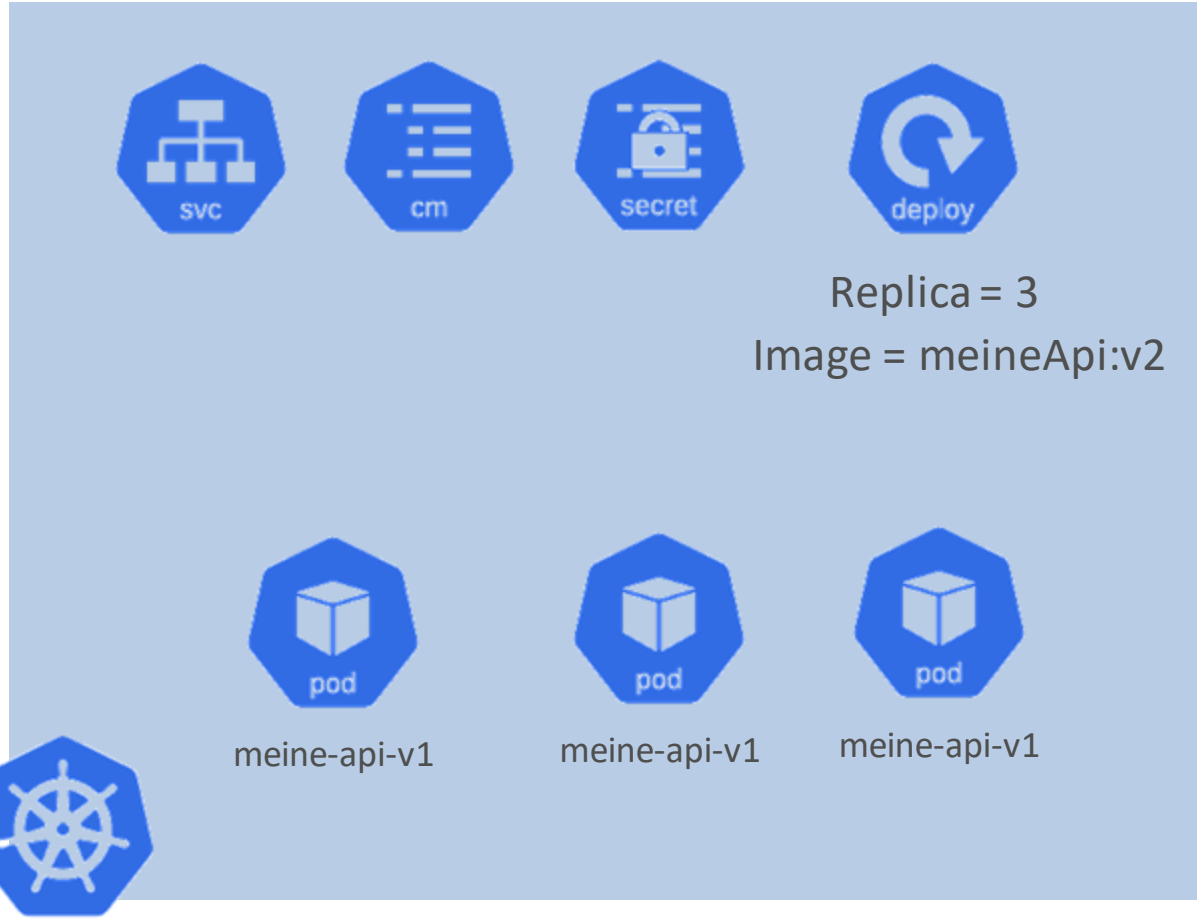
/ Phase 1: Auf geht‘s!

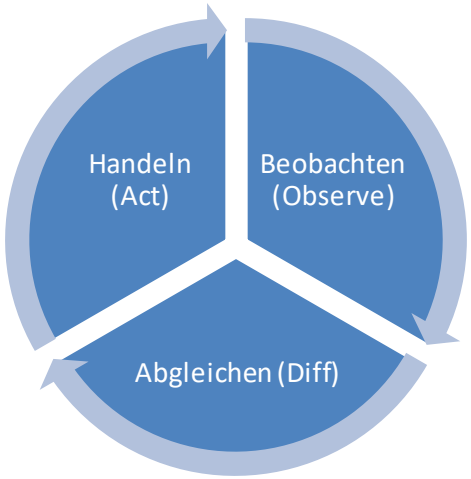




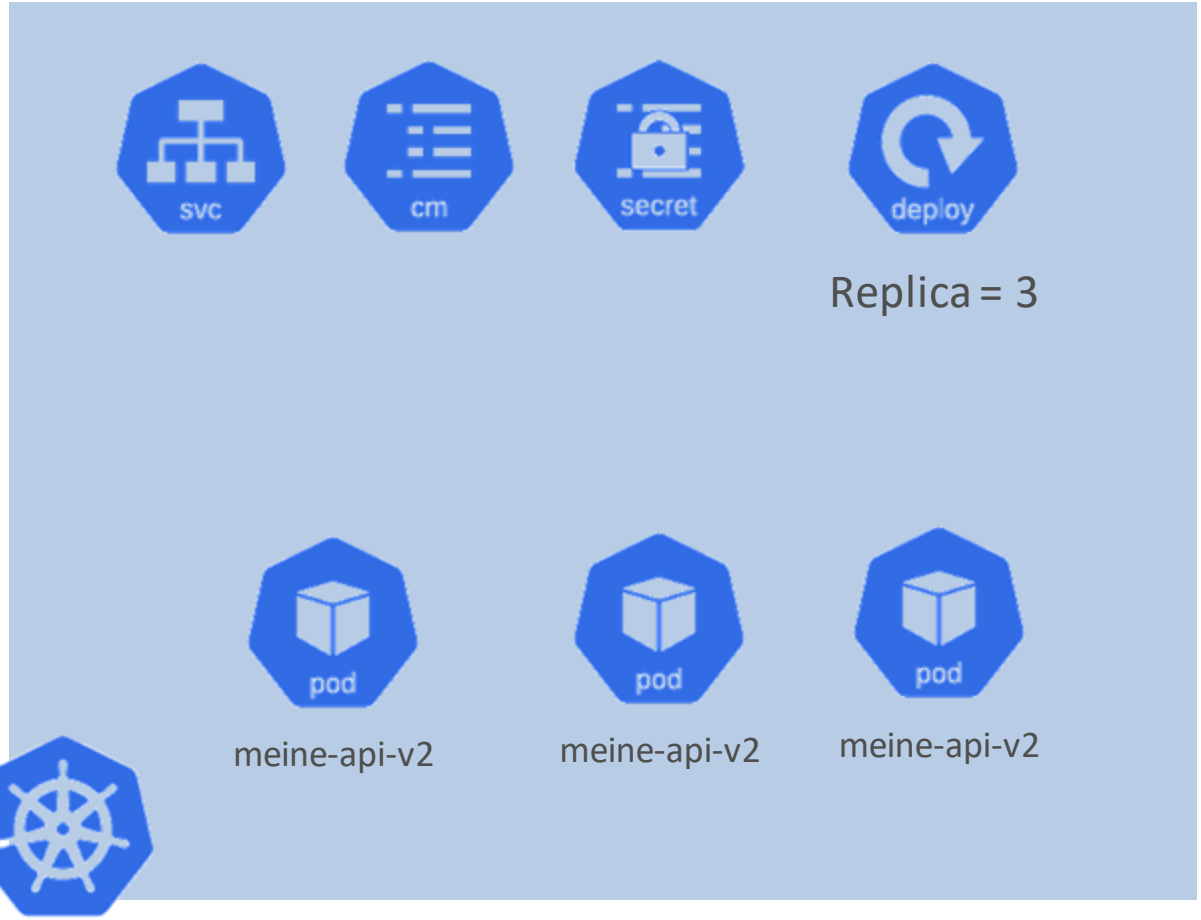
meine-api-v1

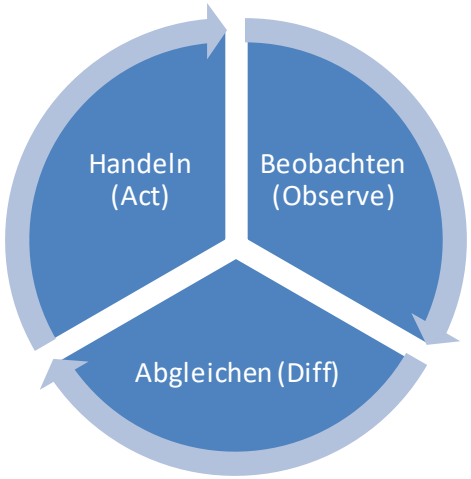
/ Phase 1: Auf geht‘s!





/ Phase 1: Auf geht‘s!





/ Phase 1: Auf geht‘s!

- Kennenlernen der Basis Ressourcen Typen
  - Pod, Deployment, Service, ReplicaSet, Secret, ConfigMap

- Erste Versuche Kubernetes zu bändigen erfolgen
  - kubectl steht uns als treuer Verbündeter zur Seite

- Es werden immer mehr YAML Dateien
  - Helm wird entdeckt

Fazit Phase 1:
Kubernetes hat uns überzeugt. Super Sache!

/ Phase 2: Don‘t Panic!

- Der Erfolg hat sich rumgesprochen:
  - Es sollen nun auch komplexe Anwendungen wie Datenbanken in das Cluster.

- Weitere Ressourcen Typen und Konzepte werden entdeckt:
  - StatefulSets, PersistentVolumes, Jobs

- Es werden immer mehr YAML Dateien
  - Helm Charts werden nun selber riesig. Es werden nun aufwendige Runbooks geschrieben, um Fehler bei der Konfiguration zu vermeiden und gängige Szenarien abzudecken (z.B. Backups).

Fazit Phase 2:
Ganz schön aufwendig. Wie soll das mit 3 Leuten skalieren?

/ Kubernetes Resource Basics

k8s Resource

- Resource Definition
- Resource Controller

k8s API

Phase 3: Neue Wege.

/ Custom Resource Definition

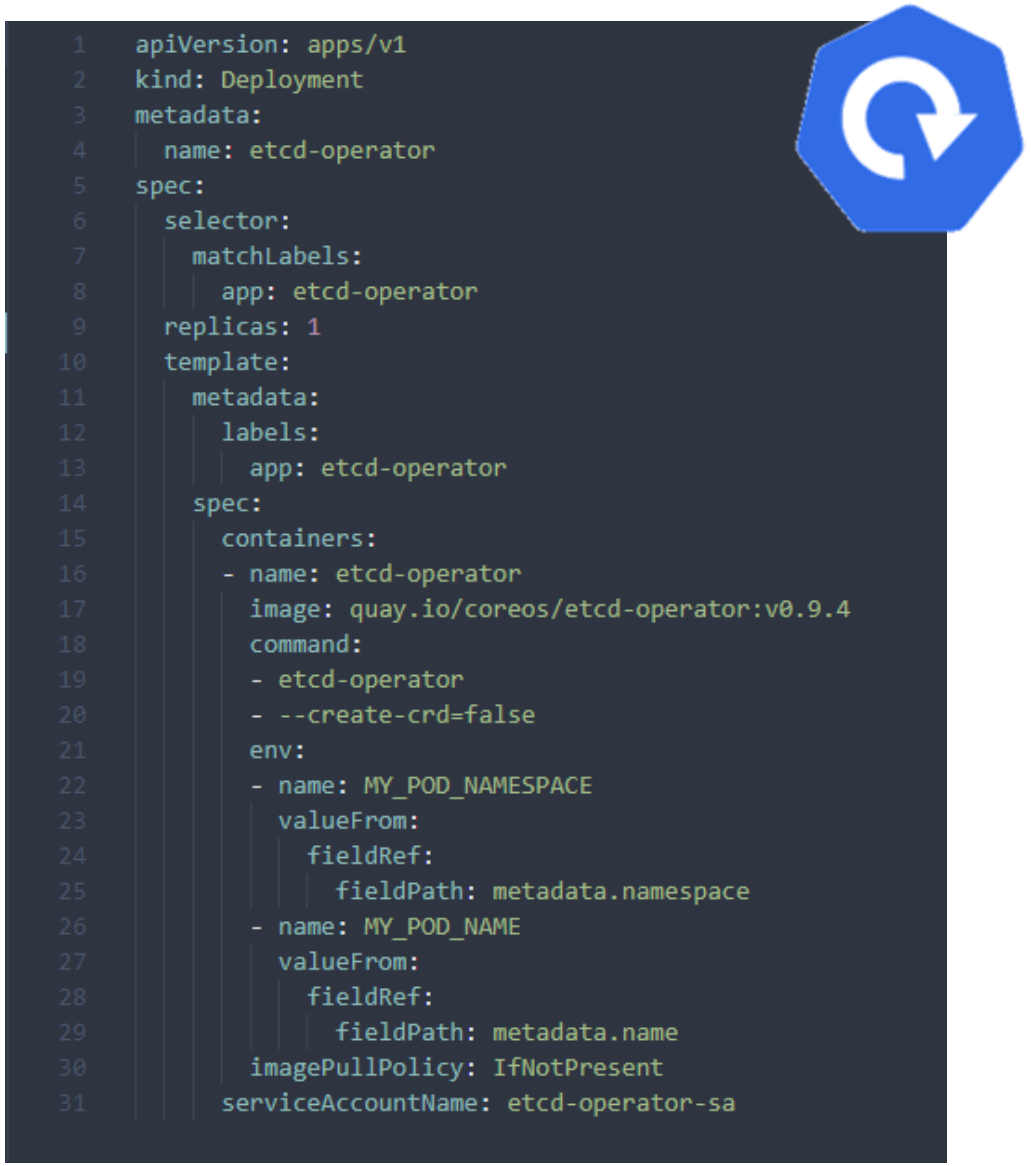
- Beschreibt und Registriert die CRD gegenüber der Kubernetes API
- CRD wird somit wie andere Ressourcen Teil der Control Loop
  - D.h. Events zur CR werden nun durch Kubernetes veröffentlicht und man kann diesen Events lauschen.
  - Ermöglicht uns die standardmäßige Control Loop, um unsere benutzerdefinierte Logik zu ergänzen.

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: etcdclusters.etcd.database.coreos.com
spec:
  group: etcd.database.coreos.com
  names:
    kind: EtcdCluster
    listKind: EtcdClusterList
    plural: etcdclusters
    shortNames:
    - etcdclus
    - etcd
    singular: etcdcluster
  scope: Namespaced
  versions:
  - name: v1beta2
    served: true
    storage: true

```

/ Custom Resource Controller

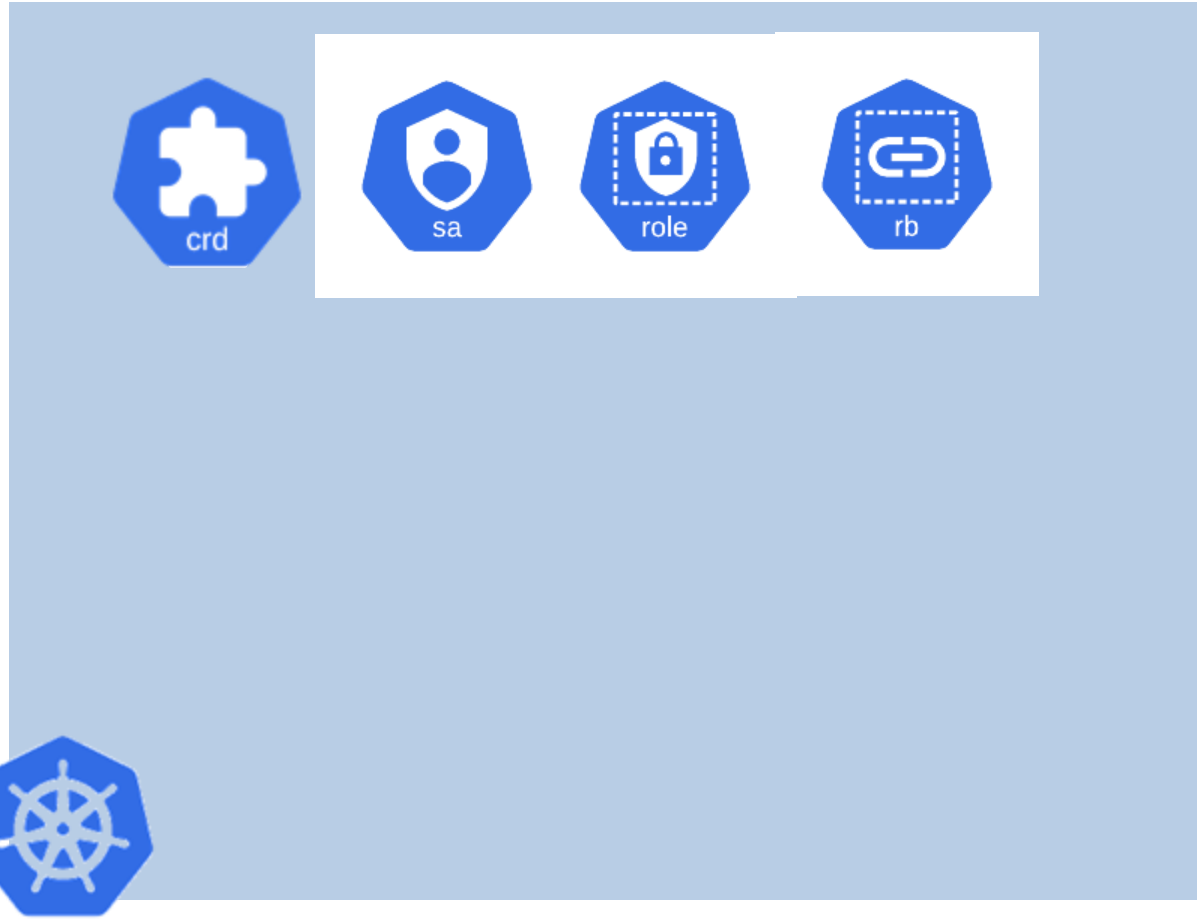
- Unser Programm, welches die Kubernetes Control Loop um unsere benutzerdefinierte Control Loop ergänzt.



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: etcd-operator
spec:
  selector:
    matchLabels:
      app: etcd-operator
  replicas: 1
  template:
    metadata:
      labels:
        app: etcd-operator
    spec:
      containers:
      - name: etcd-operator
        image: quay.io/coreos/etcd-operator:v0.9.4
        command:
        - etcd-operator
        - --create-crd=false
        env:
        - name: MY_POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: MY_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        imagePullPolicy: IfNotPresent
      serviceAccountName: etcd-operator-sa
```

/ Phase 3: Neue Wege.

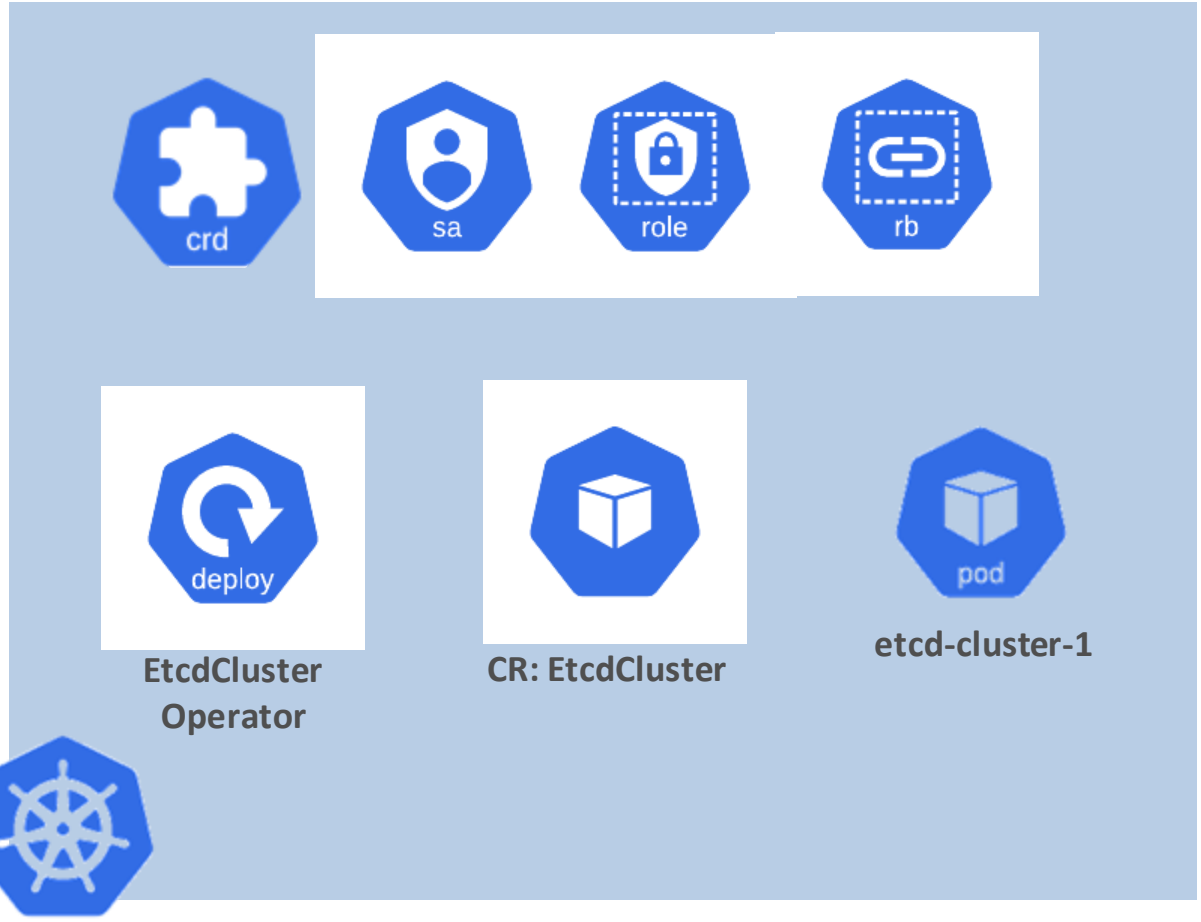
/ Phase 3: Neue Wege.



CR: EtcdCluster

EtcdCluster Operator

/ Phase 3: Neue Wege.



Demo

Quellcode mit Anleitung:
https://github.com/svalabs/workshop-operator-in-a-nutshell

SVA

Take-Away

- Operator erweitern die standardmäßigen Kubernetes Ressourcen.
- Mit Ihnen können zustandsbehaftete, individuelle und komplexe Softwarelösungen in einem Kubernetes Cluster bereitgestellt werden.
- Operator erlauben das spezifische und über Jahre gebildete Wissen des Betriebs solcher komplexen Lösungen als ausführbaren Code zu definieren.
- Dadurch werden wiederkehrende, monotone Konfigurations- und Wartungstätigkeiten automatisiert.
- Treten wiederkehrende Probleme auf, wird nicht an der Software selber gearbeitet, sondern ein Patch für den Operator veröffentlicht.
- Dies erlaubt eine Entkopplung zwischen Fehlerbehebung und Bereitstellung eines Patches. Die Bereitstellung kann so simpel sein wie ein „kubectl apply“. Dies erlaubt Skalierung.


At last

/ Weiterführende Links

- MySQL Operator: https://github.com/oracle/mysql-operator
- Hands-On Kurs 1: https://learn.openshift.com/ansibleop/ansible-operator-overview
- Weiterführende Kurse: https://developers.redhat.com/courses/openshift-operators
- E-Book O‘reilly: https://developers.redhat.com/books/kubernetes-operators
- Marketplace mit Openshift zertifizierten Operator:
  - https://marketplace.redhat.com/en-us
- Operator-SDK:
  - https://sdk.operatorframework.io/
- Ansible-Kubernetes Plugin:
  - https://docs.ansible.com/ansible/latest/collections/community/kubernetes/k8s_module.html
- Marketplace mit diversen „Out-Of-The-Box“ Operator:
  - https://operatorhub.io/