# Kubernetes Operatori

Danijel Radaković

# 1 Uvod

# Proširivanje Kubernetes-a



- Kubernetes omogućava upravljanje nativnih resurasa (`Pod`, `Deployment`, `CoinfigMap` itd.).
- Međutim, Kubernetes se može proširiti da upravlja novim tipovima resurasa.
- Postoje 2 načina kako se može proširiti Kubernetes da upravlja novim tipovima resurasa:
  - Korišćenjem **Custom Resource Definition** (CRD) i implementacijom operatora za definisane CRD-jeve.
  - Konfiguracijom Aggregation Layer-a.
- Mogućnost proširivanje je glavna prednost Kubernetes-a u odnosu na druge orkestratore. Na ovaj način možemo registrovati svoje komponente sistema kojima će upravljati Kubernetes.
- Takođe, svoje komponente sistema možete integrasiti sa alatatima i rešenjima koji su deo Kuberenetes-ovog ekosistema.

# Proširivanje Kubernetes-a



- Postoje popriličan broj alata u Kubernetes-ovom ekosistemu koji se zasnivaju na ovoj proširivosti.
- Među najpoznatijima je Crossplain, koji omogućava da pomoću Kubernetes-a upravljate infrastrukturoma na različitim Cloud provajderima.
- Na primer, pomoću Crossplain alata možete upravljati EC2 instancom na AWS nalogu.
- Crossplain je zapravo IaC alat i predstavlja alterantivu za Terraform.

# 2 Operatori

# Opšti pojmovi



- Svi resursi u Kubernetes klasteru se upravljaju od strane nekog operatora.

**Resource**

A resource is an endpoint in the Kubernetes API that stores a collection of **API objects** of a certain kind; for example, the built-in pods resource contains a collection of Pod objects.

**API object**

An entity in the Kubernetes system, represanting the part of the state of your cluster.

- Nativni resursi se upravljanju od strane operatora (`controller-manager`) koji se nalaze u `kube-system` namespace-u.

# Opšti pojmovi



- Operator je u suštini `Pod/Deployment` koji sluša na izmene stanja određenih objekata, procesira ih tako da objektne dovede u željeno stanje.
- Operator se sastoji od skupa kontrolera.

**Controller**

In Kubernetes, controllers are control loops that watch the state of your cluster, then make or request changes where needed. Each controller tries to move the current cluster state closer to the desired state (doc).

- Samim tim postoje `Deployment` kontroler, `Deamonset` kontroler, `Ingress` kontroler itd. koji su deo `controller-manager`-a.

# Opšti pojmovi



- Pored nativnih, postoje custom resursi i kontroleri.

**Custom Resource**

A custom resource is an extension of the Kubernetes API that is not necessarily available in a default Kubernetes installation. It represents a customization of a particular Kubernetes installation. Custom resources let you store and retrieve structured data (doc).

**Custom Controller**

Custom controllers can work with any kind of resource, but they are especially effective when combined with custom resources (doc).

# Opšti pojmovi



**Operator Pattern**

The Operator pattern combines custom resources and custom controllers.

- **Primer operatora**: Hoćemo da koristimo Discord za notifikacije ali da se kreiranje servera, grupa i kanala radi preko Kubernetes-a.
- Posotjala bi 3 resursa: `DiscordServer`, `DiscordGroup`, `DiscordChannel`.
- Postojla bi 3 kontrolera: `DiscordServerController`, `DiscordGroupController`, `DiscordChannelController`, koji bi bili nadležni za kreiranje, brisanje, i izmene odgovarajućih objekata.

# Primeri i implementacije operatora



- Konkretne implementacije operatora:
  - Prometheus Operator,
  - MongoDB Operator,
  - CloudNativePG,
  - ostali operatori koji su deo ekosistema.
- Postoje gotovi alati koji omogućavaju implentaciju operatora (koristeći operator pattern):
  - Kubebuilder (Golang),
  - Operator SDK (Java),
  - kube-rs (Rust).

# 3 Kubebuilder

# Uvod



- Kubebuilder je alat koji nam omogućava da implementiramo operator i kontrolere za naše custom resurse.
- U konrektnim primerimara radimo implementaciju operatora za Dojo aplikaciju.
- Operator ima sledeće custom resurse i njihove kontrolere:
  - `Dojo`: kreiara Deployment za Dojo aplikaciju.
  - `DiscordServer`: kreira Discord server.
  - `DiscordGroup`: kreira Discord grupu.
  - `DiscordChannel`: kreira Discord text kanal.
- Implementacija operatora je dostupna ovde.

# Podešavanje okruženja



- Instalirati Go, verzije `>=1.25.6`.
- Instalirati Kubebuilder, verzije `>=4.11.0`.
- Instalirati k3d, verzije `>=5.8.3`.
- Namesiti bash completion:

```
sudo sh -c 'k3d completion bash > /etc/bash_completion.d/k3d'
sudo sh -c 'kubebuilder completion bash > /etc/bash_completion.d/
kubebuilder'
```

# Kreiranje klastera



```yaml
# cluster.yaml
apiVersion: k3d.io/v1alpha5
kind: Simple
metadata:
  name: local
servers: 1
```

```
k3d cluster create --config cluster.yaml
```

# Kreiranje projekta



asldkjf adsf adsf a adsf

# Kreiranje custom resursa (proširavanje API-a)



- Želimo da definišemo `Dojo` custom resurs.
-

# Custom Resource Definition (CRD)



- Za definisanje custom resursa koristi se Custom Resource Definition (CRD) resurs.
- To je tip resursa koji kreira RESTful resource path i OpenAPI v3.0 šemu na API Serveru koju koristi za validaciju REST zahteva.
- Samim tim mnoge funkcionalnosti koje podržava OpenAPI V3 podržava i CRD uz neke razlike i ograničenja doc.
- Svaki CRD mora da ima sledeće:
  - ApiVersion:
  - Kind:
  - Scope:
  - Spec: želeno stanje resursa
  - Status: trenutno stanje resursa
  - Singular:

# Custom Resource Definition (CRD) (ii)



- Plural:

# Kreiranje custom resursa (proširavanje API-a)



- Kubebuilder nam omogućava da na jednostavan način kreiramo CRD i kontroler.

```
kubebuilder create api
```

- Za implentaciju CRD treba da:
  - Dodamo polja koja su nam potrebna u Spec i Status (nalaze se na putanji )
  - Anotiramo te strkture sa kubebuilder markups-a.

**Kubebuilder markups**

Predstavljaju direktive za `controller-gen` alat koji generise OpenAPI v3.0 šemu.

# Kreiranje custom resursa (proširavanje API-a)



```
type struct Spec{}
```


```
type struct Spec{}
```


make manifest generate

# Upravljanje `reconcile` petljom



napravi prvo loop koji samo ispisjuje, i ne to bude veryija v0 na githubu. zatim uradi instalaciju CRD i reci koji RESTful path-ovi su kreirani. Probaj te resurse da preko kubectl i curl da pozoves

Da bi smo na pravilan način implenetirali `reconcile` petlju, moramo prvo bolje da znamo kako API Server i ostali mehanizmi funkcionišu u pozadini.

# Pisanje testova



Pisanje testova

# Kreiranje Postgres baze



- Aplikacija zahteva konekciju ka bazi. Očekuje da se krenedicijali nalaze u `env var` koji će se popuniti iz `Secret` objekta.
- **Problem**: kako znati koji `Secret` korisiti i šta treba da bude njegov sadržaj?
- Jedno rešenje bi bilo da kreiramo custom resurs `Postgres` i kontroler koji bi kreirao bazu kao `Statefulset` i `Secret` sa odgovarjućim kredencijalima.
- Međutim, ova implementacija bi bila minimalna i pitanje koliko mi moglo da posluži u produkcionim okruženjima.
- Bolja opcija je da koristimo već gotove operatore za upravljanje Postgres bazom, kao što je CloudNativePG.

# Kreiranje Postgres baze



- Custom resurse koje `CloudNativePG` operator nudi se nalaze ovde.
- Od posebnog značaja je `Database` resurs koji omogućava kreiranje baze unutar klustera i `Secret` objekta sa odgovarajućim kredencijalima.
- Sve što naj je ostalo jeste da proširimo `Spec` sekciju u kojoj je moguće definisati tip baze koji se koristi i naziv `Secret` objekta.

# Nadogradnja `Spec` sekcije



dodaj kako izgleda golang kod

dodaj kako izgleda yaml primer