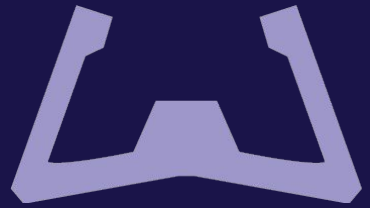


Helm vs. Yoke

aneb jak se nezbláznit z YAML template

Lukáš Stuchlík
lukas.stuchlik@prorocketeers.com

ProRocketeers

O mně

Magisterské studium na Ostravské univerzitě - obor Informační systémy

5 let v ProRocketeers:

- agilní vývoje softwaru pro klienty - Team as a Service
- backend vývoj - Typescript, Golang
- DevOps - CI/CD, správa K8s clusterů, Helm, Docker, Terraform...

Open source contributor:

- WoWAnalyzer
- Traefik
- Yoke

Helm charty - jde to i jinak..

```
{{- define "setup.getDeploymentContainers"  -}}
{{/* parameter - deploymentValues (from "setup.getMergedDeploymentValues") */  }}
{{- $deploymentValues  := . -}}
{{/* From global values, pick properties that configure the main container */  }}
{{- $mainContainer  := pick $deploymentValues  "image" "args" "command" "ports" "envs" "envsRaw" "kubeSecrets"
"vaultSecrets" "resources" "readinessProbe" "readinessProbeRaw" "livenessProbe" "lifecycle" "mainContainerName"  -}}
{{- include "setup.validateMainContainerImage"  $mainContainer.image -}}
{{- $_  := $mainContainer.mainContainerName |  default "main" | set $mainContainer  "name" -}}
{{- $containers  := list $mainContainer  -}}
{{/* add sidecars - containers running parallel to the main container - they have their own config */   }}
{{- range $sidecarName, $sidecar  := $deploymentValues.sidecars -}}
 {{- $_  := set $sidecar  "name" $sidecarName  -}}
 {{- $validImage  := include "setup.validateAndSetSideContainerImage"  (dict "image" $sidecar.image "mainImage"
$deploymentValues.image) | fromYaml -}}
 {{- $_  := set $sidecar  "image" $validImage  -}}
 {{- $containers  = append $containers  $sidecar  -}}
{{- end -}}
{{- $containers  | toJson -}}
{{- end -}} {{/* end template */ }}
```

ProRocketeers

Helm charty - jde to i jinak..

```
func getDeploymentContainers (input schema.InputValues) ([]Container, error) {
 // validate main container image
 if input.Image.Tag == nil {
   return []Container{}, fmt.Errorf("main container must have `image.tag` set" )
 }
 if len(input.Ports) == 0 {
   return []Container{}, fmt.Errorf("main container must have at least one port" )
 }
 containers := []Container{
   convertContainer(input.Container, input.MainContainerName, ptr.To("main")),
 }
 for sidecarName, sidecarInput := range sortedMap(input.Sidecars) {
   if err := validateAndSetSideContainerImage(&sidecarInput.Image, &input.Image); err != nil {
     return []Container{}, fmt.Errorf("error validating sidecar ' %v': %v", sidecarName, err)
   }
   containers = append(containers, convertContainer(sidecarInput, ptr.To(sidecarName)))
 }
 return containers, nil
}
```

ProRocketeers

ProRocketeers Helm chart

Problém - individuální Helm charty pro každou aplikaci

- špatná údržba, globální změny, zdlouhavý setup

Řešení - vlastní obecný Helm chart

- centralizované místo pro změny, rychlé nasazení nové služby
- jednoduché verzování

Počátek komplikací

- jiný tvar vstupních dat než vnitřní datová reprezentace
- transformace, validace…

Ošklivá strana Helm chartů

Mizerná podpora IDE - syntax highlighting, IntelliSense

YAML je syntakticky založený na odsazení ⇒ whitespace chyby

- `{{- hell -}}`

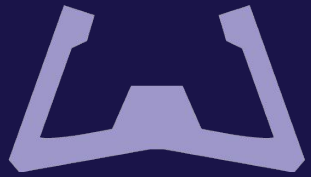
Testing - žádné vestavěné možnosti

- Helm plugin helm-unittest

```
tests:
 - it: can specify image pull policy
   set:
     <<: *required_values
     image.pullPolicy: Always
   asserts:
     - equal:
         path: spec.template.spec.containers[0].
           imagePullPolicy
         value: Always
```

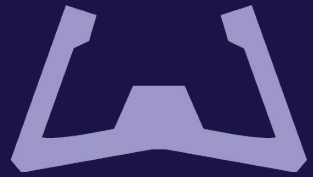
ProRocketeers

Yoke



- YAML template ⟶ vytváření Kube objektů za pomoci Go
- Chart ekvivalent - “Flight”
  - program zkompilovaný do WASM
  - výstup - JSON pole Kubernetes resourců
- proč WASM?
  - společný kompilační target pro více jazyků
  - nezávislý na OS nebo architektuře
  - sandboxed runtime

Yoke



Proč je doporučované Go?

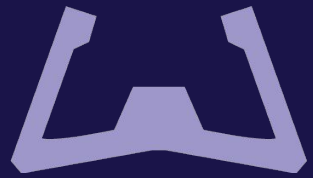
- Yoke SDK
- oficiální podpora Kubernetes API typů

Hlavní výhody Yoke

- plná podpora IDE - IntelliSense, unit testing…
- jednoduchý refactoring
- klasický debugging s breakpointy
- zpětná kompatibilita s Helmem

ProRocketeers

Yoke



```go
type values struct {
 Name      string `json:"name"`
 Namespace string `json:"namespace"`
 Replicas  int32  `json:"replicas"`
 Image     string `json:"image"`
 Port      int32  `json:"port"`
}
```

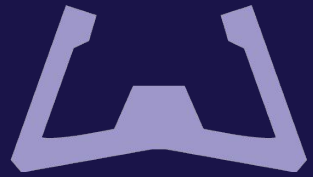
```go
func run() error {
 var values values
 err := yaml.NewYAMLToJSONDecoder(os.Stdin)
             .Decode(&values)

 if err != nil && err != io.EOF {
   return err
 }

 return json.NewEncoder(os.Stdout).Encode([]any{
   createDeployment(values),
   createService(values),
 })
}
```

ProRocketeers

Yoke



```
func createDeployment(values values) appsv1.Deployment {
 return appsv1.Deployment{
   TypeMeta: metav1.TypeMeta{ /**/ },
   ObjectMeta: metav1.ObjectMeta{ /**/ },
   Spec: appsv1.DeploymentSpec{
     Replicas: &values.Replicas,
     Selector: &metav1.LabelSelector{ MatchLabels: commonLabels(values) },
     Template: corev1.PodTemplateSpec{
       Spec: corev1.PodSpec{
         Containers: []corev1.Container{
           {
             Name:  values.Name,
             Image: values.Image,
             Ports: []corev1.ContainerPort{
               {Name: "http", Protocol: corev1.ProtocolTCP, ContainerPort: values.Port},
             },
           },
```

ProRocketeers

Yoke - testing, publishing

- klasicky to spustit
  - `cat input.yaml | go run . | jq`
  - `go run . --file input.yaml`
    - vhodnější pro debugging
- zkompilovat do WASM a spustit
  - `GOOS=wasip1 GOARCH=wasm go build -o flight.wasm`
  - `yoke takeoff -dry -stdout test flight.wasm < input.yaml | jq`
- uploadnout/publishnout někam
  - `yoke stow -tag latest flight.wasm oci://ghcr.io/…/flight`

ArgoCD

Helm - nativní integrace

- Argo využívá Helm pouze pro templating

```
sources:
  - repoURL: <Helm repository URL>
    chart: <chart name>
    targetRevision: <chart version>
    helm:
      valueFiles:
        - $values/path/to/values.yaml
  - repoURL: <values git URL>
    targetRevision: HEAD
    ref: values
```

ProRocketeers

ArgoCD

Yoke - jako Content Management Plugin

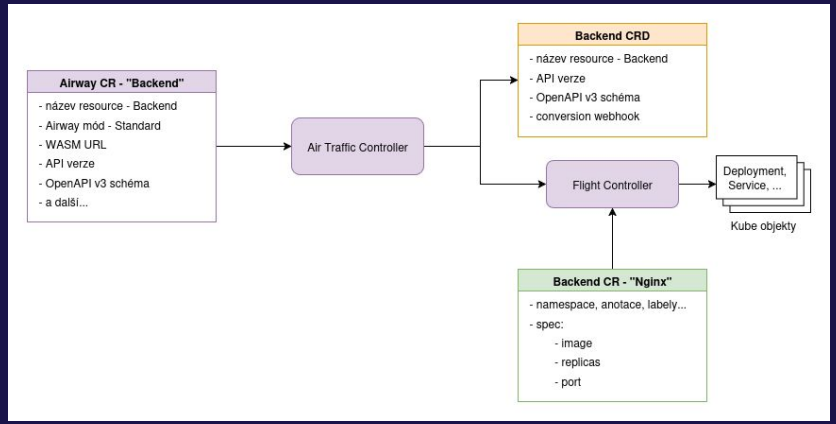
- upravení ArgoCD Repo Server podu, přidání dalšího kontejneru

```
source:
  repoURL: <values git URL>
  targetRevision: HEAD
  path: .
  plugin:
    name: yokecd
    parameters:
      - name: wasm
        string: <WASM URL>
      - name: inputFiles
        array:
          - path/to/values.yaml
```

ProRocketeers

Yoke - ATC

- Helm - nedostatečná validace vstupů
  - nutnost se spoléhat na JSON schéma v Chartu anebo “aplikační” logiku
- Air Traffic Controller (ATC)
  - možnost jednoduše vytvářet vlastní Custom Resources (CR), které jsou implementované Yoke Flighty
  - daleko pohodlnější způsob implementace než klasicky s kubebuilderem / controller-managerem…



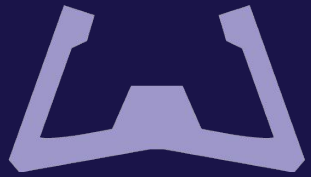
ProRocketeers

Srovnání - Helm

+ battle tested - v provozu 9-10 let, od 2018 pod CNCF, Graduated v 2020
+ jednoduchý na použití - rychlá instalace, vytváření i vlastních Chartů
+ nativní support v známých systémech - ArgoCD, FluxCD...
+ Helm hooky

- ne úplně bezpečný - supply chain útoky, horší kontrola implementace Chartu
- udržovatelnost složitějších chartů - IDE support, testování, debugging...

Srovnání - Yoke



+ využití programovacího jazyka ke tvorbě Kube resourců
  + plná podpora IDE, type safety, testování, debugging…
+ slušná kompatibilita s Helmem - plynulejší přechod
+ ATC - reprezentace Flightů ve formě custom resources

- opět supply chain útoky (spíš teoreticky)
- projekt je *velmi* nový - leden 2024, žádní sponzoři / velké firmy, 1 maintainer

Díky za pozornost!

Lukáš Stuchlík
lukas.stuchlik@prorocketeers.com

ProRocketeers