11.11.2020
INNOQ TECHNOLOGY LUNCH

Infrastructure as Code mit Helm & Helmfile

INNOQ

Tammo van Lessen
Principal Consultant

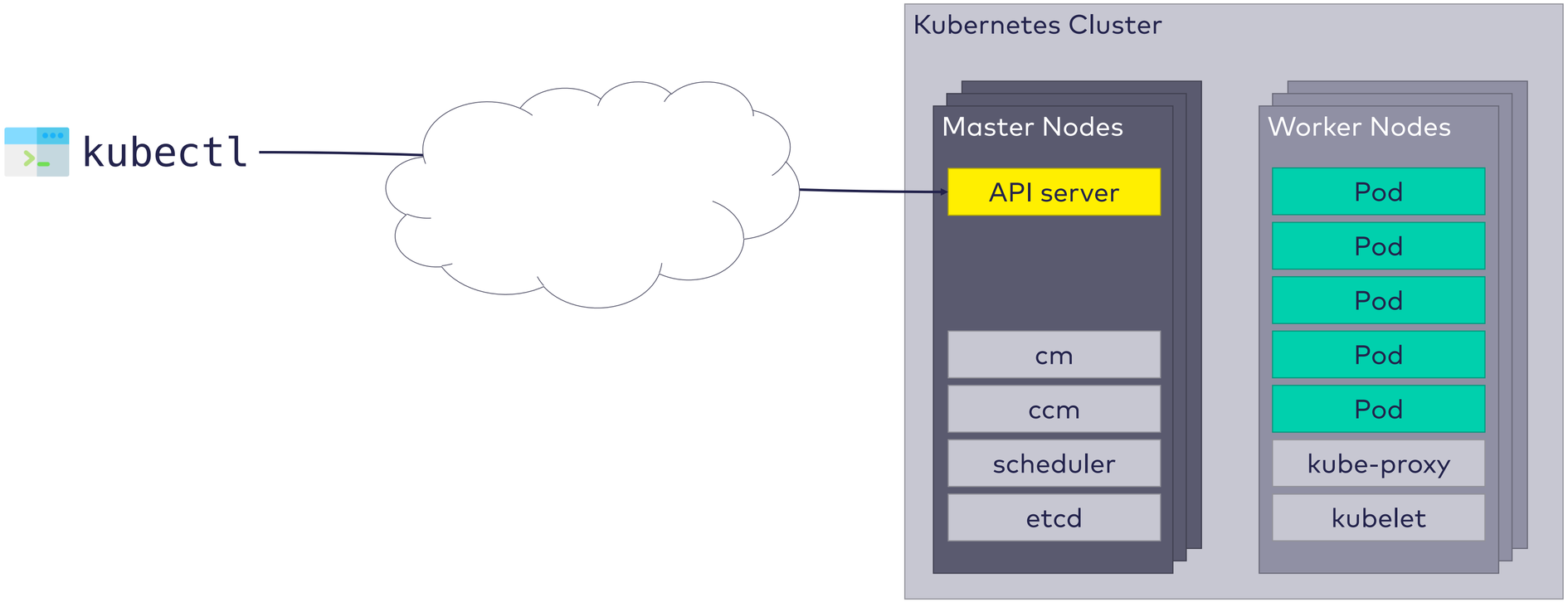
Kontext

Helm

Helmfile

Szenarien

Kubernetes in 11 Sekunden




Viele Wege führen
zum Deployment
https://unsplash.com/photos/FcLq69V7Rsc

Release early, release often

https://unsplash.com/photos/92ef3opsrLU

Helm.

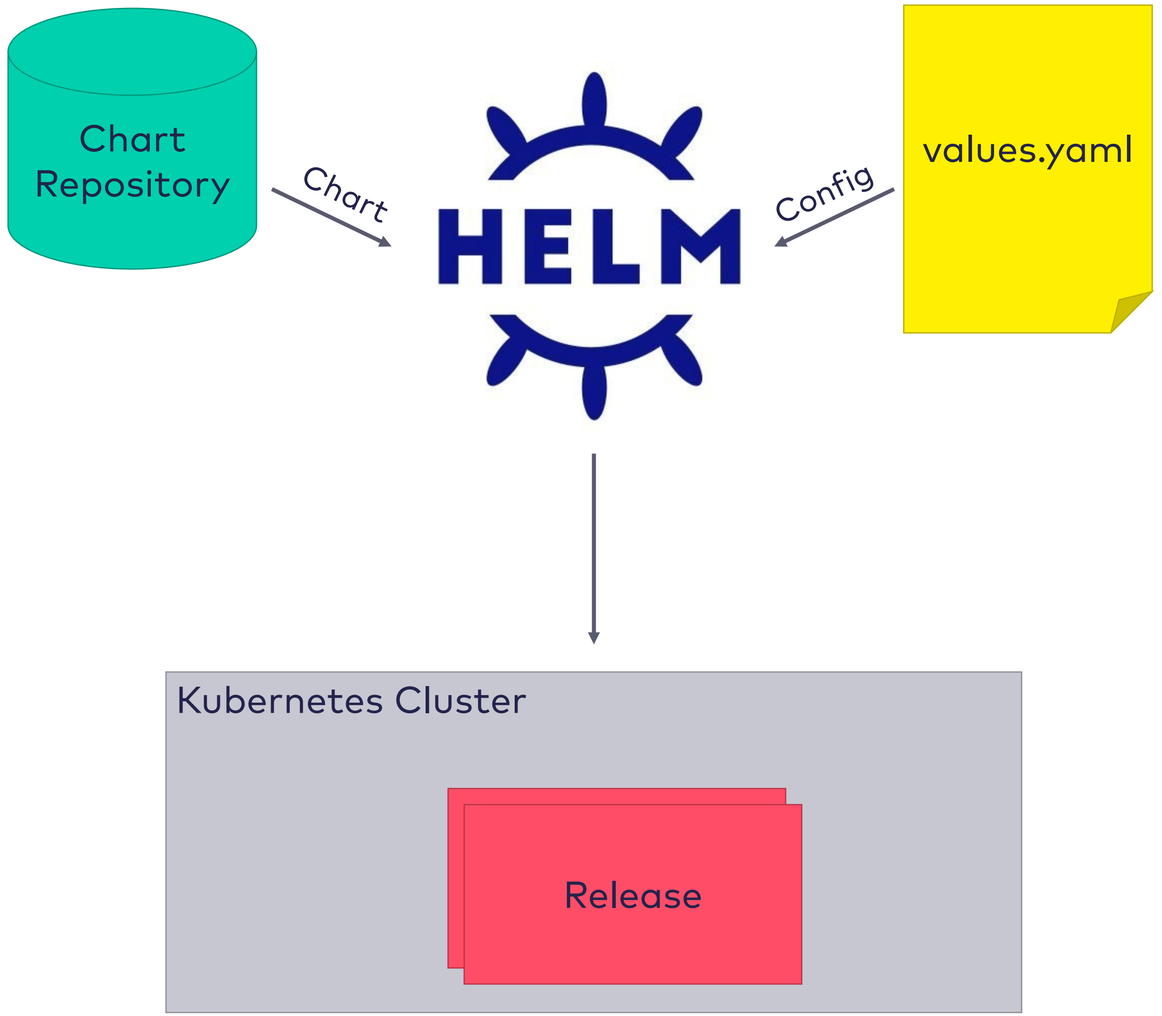
Was ist Helm?

- Paketmanager für Kubernetes
- De-facto Standard (je nach dem wen man fragt)
- Vitales Ökosystem im Artifact Hub (artifacthub.io)

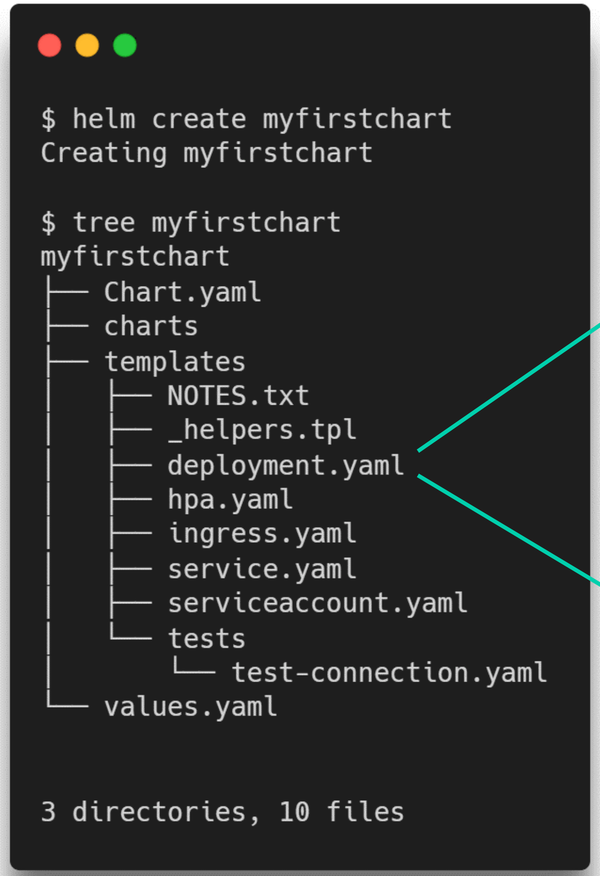
- Template-basiertes Rendering von K8s-Manifests
- Versteckt K8s-Details, values.yaml ist die Schnittstelle
- Bildet logische Klammer um zusammengehörige Manifeste
- Versionierte Pakete (semver)
- Wohl gealtert

Helm-Terminologie

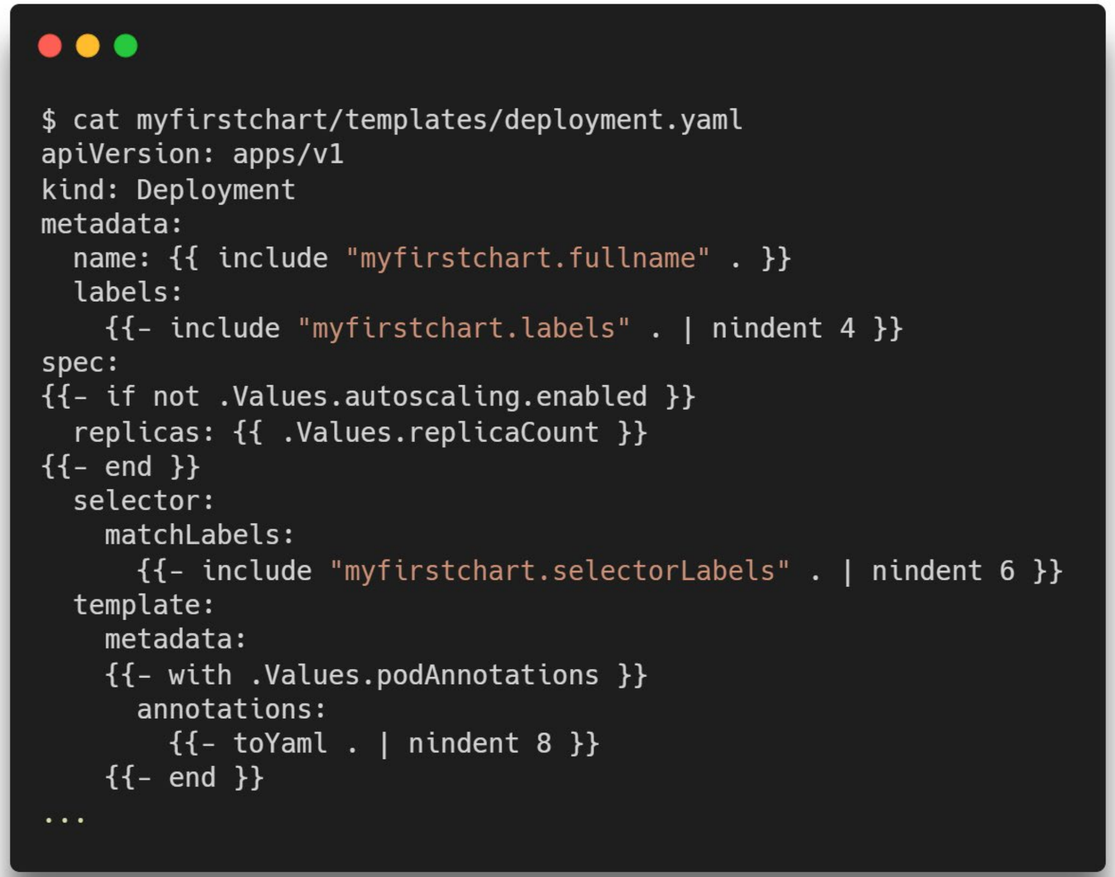
- Chart
- Repository
- Template
- Config
- Release
- Dependencies / Umbrella Chart
- (Tiller)



Unser erstes Helm Chart



```
$ helm create myfirstchart
Creating myfirstchart

$ tree myfirstchart
myfirstchart
├── Chart.yaml
├── charts
├── templates
│   ├── NOTES.txt
│   ├── _helpers.tpl
│   ├── deployment.yaml
│   ├── hpa.yaml
│   ├── ingress.yaml
│   ├── service.yaml
│   ├── serviceaccount.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml


3 directories, 10 files
```



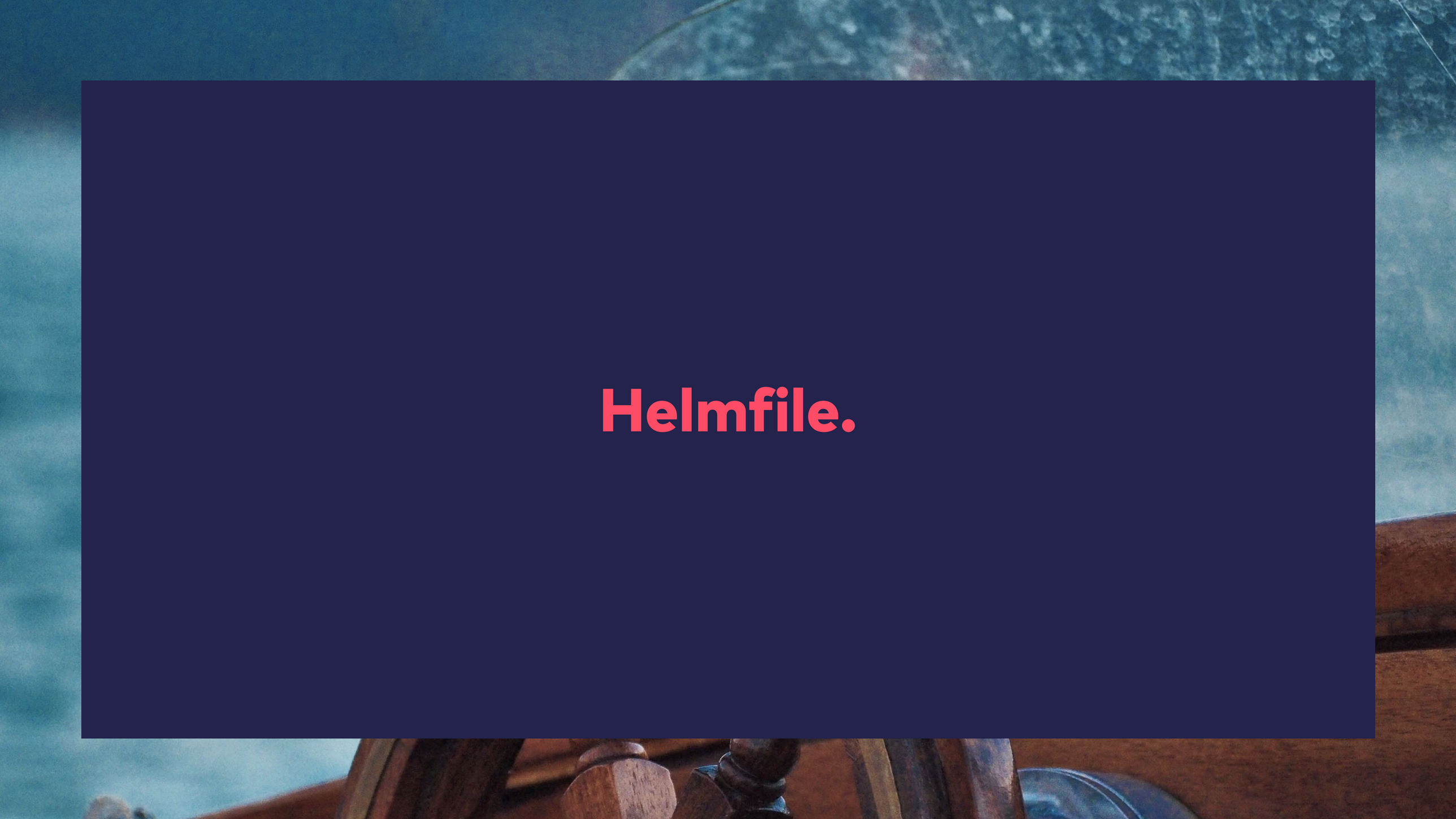
```
$ cat myfirstchart/templates/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "myfirstchart.fullname" . }}
  labels:
    {{- include "myfirstchart.labels" . | nindent 4 }}
spec:
{{- if not .Values.autoscaling.enabled }}
  replicas: {{ .Values.replicaCount }}
{{- end }}
  selector:
    matchLabels:
      {{- include "myfirstchart.selectorLabels" . | nindent 6 }}
  template:
    metadata:
    {{- with .Values.podAnnotations }}
      annotations:
        {{- toYaml . | nindent 8 }}
    {{- end }}
...
```

Rollout mit Helm

```
$ helm repo add stable https://charts.helm.sh/stable
$ helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
$ helm repo update

$ helm upgrade --install nginx-ingress ingress-nginx/ingress-nginx --set
controller.metrics.enabled=true --set controller.replicaCount=2 -n ingress --create-namespace

oder:

$ helm upgrade --install nginx-ingress ingress-nginx/ingress-nginx -f values.yaml -n ingress --create-
namespace

irgendwann:

$ helm uninstall nginx-ingress
```

Wo Helm nervt…

- Helm Repositories müssen in separatem Schritt bekannt gemacht werden
- Kein Templating bzw. Wiederverwendung innerhalb der Values möglich
- Umgang mit Umgebungsvariablen mühsam
- Unklar, was eigentlich auf dem Cluster verändert wird
- Semver Version Ranges nur in Umbrella-Charts
- Es gibt nicht für alles Helm Charts – oder die Qualität der Charts ist fragwürdig
- Ausweg: Shell-Skripte mit helm, kubectl, jq, yq, gettext envsubst


Helmfile.

Was ist Helmfile?

- Wrapper für Helm – führt Helm für euch aus
- https://github.com/roboll/helmfile – 2,5k ☆ – 🚌-factor?

- Alles in einer Datei – aber aufteilbar
- Mehr Kontrolle über die Abhängigkeiten
- Mehrstufiges Templating
- DRY
- Konzept von Umgebungen
- Integration von helm-diff und helm-secrets
- Werte aus externen Quellen oder Skripten beziehen

Minimale Helmfile

Beschreibt den desired state des Rollouts

```
$ cat helmfile.yaml
repositories:
  - name: stable
    url: https://charts.helm.sh/stable
  - name: ingress-nginx
    url: https://kubernetes.github.io/ingress-nginx

releases:
  - name: nginx-ingress
    namespace: ingress
    createNamespace: true
    labels:
      component: ingress
    chart: ingress-nginx/ingress-nginx
    version: ~3.9.0
    set:
      - name: controller.metrics.enabled
        value: true
      - name: controller.replicaCount
        value: 2
```

```
$ cat helmfile.yaml
repositories:
  - name: stable
    url: https://charts.helm.sh/stable
  - name: ingress-nginx
    url: https://kubernetes.github.io/ingress-nginx

releases:
  - name: nginx-ingress
    namespace: ingress
    createNamespace: true
    labels:
      component: ingress
    chart: ingress-nginx/ingress-nginx
    version: ~3.9.0
    values:
      - controller:
          metrics:
            enabled: true
          replicaCount: 2
```

CLI

```
$ helmfile list      # alle definierten Releases anzeigen

$ helmfile sync      # alle Releases ausrollen

$ helmfile diff      # Manifeste der Releases generieren und Diff anzeigen

$ helmfile apply     # nur Releases ausrollen deren Manifeste sich geändert haben

$ helmfile destroy   # alle definierten Releases deinstallieren


$ helmfile apply --selector component=ingress   # apply, für bestimmte Labels

$ helmfile apply --environment production       # apply, mit umgebungsspezifischen Values
```

Environments

- Konstrukt, um umgebungsspezifische Werte zu definieren
- Inline oder aus Dateien
- Name steht in **`.Environment.Name`**, Werte unter **`.Values`** für Templating zur Verfügung

```
...
environments:
  development:
    values:
      - dev-values.yaml
      - nginx-version: 3.9
  production:
    values:
      - prod-values.yaml
      - nginx-version: 3.8

...
```

Mehrstufiges Templating

- Go templates mit Sprig und Erweiterungen
- Helmfile Environments sind die Wertquellen
- Platzhalter in Helmfile selbst und Dateireferenzen die auf .gotmpl enden werden evaluiert

```
{{ readFile "values.yaml" | fromYaml | setValueAtPath "foo.bar" "FOO_BAR" | toYaml }}
```

```
{{ exec "htpasswd" (list "-nb" "prometheus" (requiredEnv "PROMETHEUS_PASSWORD")) }}
```

Don’t repeat yourself

Eine Helmfile ist YAML, also kann man YAML auch nutzen.

```yaml
repositories:
  - name: myorg
    url: https://charts.myorg.io/

templates:
  default: &default
    chart: myorg/{{`{{ .Release.Name }}`}}
    namespace: {{`{{ .Environment.Name }}`}}
    missingFileHandler: Warn
    values:
    - config/{{`{{ .Release.Name }}`}}/values.yaml
    - config/{{`{{ .Release.Name }}`}}/{{`{{ .Environment.Name }}`}}.yaml
    secrets:
    - config/{{`{{ .Release.Name }}`}}/secrets.yaml
    - config/{{`{{ .Release.Name }}`}}/{{`{{ .Environment.Name }}`}}-secrets.yaml

releases:
- name: cartservice
  <<: *default
- name: productcatalogservice
  <<: *default
- name: paymentservice
  <<: *default
- name: shippingservice
  <<: *default
...
```

https://github.com/roboll/helmfile/blob/master/docs/writing-helmfile.md#release-template--conventional-directory-structure

helm-secrets

- Helmfile integriert helm-secrets (zendesk), welches SOPS (Mozilla) verwendet.
- Kann YAML-Dateien mittels AWS KMS, GCP KMS, Azure Key Vault and PGP ver- und entschlüsseln.
- So können Secrets auch im Git-Repository hinterlegt werden.

```
controller:
    metrics:
        enabled:
ENC[AES256_GCM,data:8M5Jzw==,iv:CcQUrfgidO1aYDY0Xww4uETtL/EzYq+HXfsf08vXLMI=,tag:K/kAfkoO1f8uWkkJ9GOanA
==,type:bool]
    replicaCount:
ENC[AES256_GCM,data:ag==,iv:jMJawyP0Fz4puJmMD35a8FcMBTZnh2v1tr3cI8OaBcA=,tag:5ZuxjP/6MFS7jiBcNxHBUA==,t
ype:int]
sops:
    kms: []
    gcp_kms: []
    lastmodified: '2020-11-10T23:34:52Z'
    mac:
ENC[AES256_GCM,data:chiFh7aA6RACoFgteNRtxp42zMJAJ4UZmXzweCRDKgIBrMYzDRoeTj1gpPxinakWwd4V5JH2GYVlJnBfvcp
tiBflMy1LGmavdqgwiEBOG0pEHECMz24K5XK81ZPriPwrG1wORN7Baazo8pxrWTBH4jQRyQ1i7EEINrTT33/n2NI=,iv:mUKac1ZZoo
tSUe4lRqbjKvAqt2CzmjAYZdAMY75bMoY=,tag:PcSdXoX5rthUAcHn/RnSYQ==,type:str]
    pgp:
    -   created_at: '2020-11-10T23:34:52Z'
        enc: |
            -----BEGIN PGP MESSAGE-----

            hQGMA5AWsR/DbtDfAQwAsKEb+V0aQUMv1TMypgYLIOGoTwKaQgq+dIHjVHToBt1n
            NPOy5TVA23HpfZyhorzKydps8fPXwsuuhyaceUsLGDHwCj7C8cVDPBOb7qI7pxyj
            aasyucmZyZqh+KYxPqojpHrYGPEafdfzpCH5k4me8Y0b1TLGAttSj6nxlSfdkZk4
            RE+kmhy6r8LZsIQ/B0xkiy7hJFSIrxsvyKcFoJJ42BFxjYyf4nz71CbcH4JuFxmb
            Dfi4VEI/QkHR1EbpR9Av6uIUNBzu+NgT5BAJpVBRKRcODnGhuRf9bbNuf1rFRL/+
            a+Dzbip1IdSXz6nww6DShQM8PvUMm1ETB+lQglJJC2zoyg480HfPmenA0PujDtLx
            V45mOw7aiGhENbZhBY2RRZtY+hnQXfbx+aD4r4iBG7mV+OvR0n1RdF89dpV2qam3
            FWuC7XRdfCPdAVdB/EaAhmx/mEP/Zx95ljyQ6mcRTC/L06PkO4OTaHunJoiMt2We
            p4U/tTQWYOWid732MqZm0lwBaHF4fQyW1Ffqap3V4b/Eo8wSd8x9ts/LJAn+0nCu
            oO31KciLiklRTo7K+x4WUn1mIhskJH9hWE065YOeP39IeVvHdiYrc/p9/I9miZJD
            EJ4rIqwC3K4tGQn/+Q==
            =OBU+
            -----END PGP MESSAGE-----
        fp: 27EBD9E5F820F9E3980E65C0A3317B6BB0A5CAA0
    unencrypted_suffix: _unencrypted
    version: 3.0.3
```

Externe Values

- Weitere Möglichkeiten externe Werte und Secrets einzubinden:
  - Aufruf von Skripten via exec
  - Remote value.yaml files via Git-URL
  - variantdev/vals – Einbindung über URI-Referenzen
    - Vault, AWS (S3, SSM, Secrets Manager), GCP Secrets Manager, SOPS, Terraform State, File
    - **`mysql-password: refs+vault://secret/data/mycomponent#/mysqlpw`**

Ich hab doch nur ein wenig YAML...

- incubator/raw ist ein Helm Chart, das Manifests als Values entgegen nimmt.

```
$ cat helmfile.yaml
releases:
- name: 'influxdb-alerts'
  chart: "incubator/raw"
  namespace: "{{ .Environment.Name }}-influxdb"
  version: "0.2.3"
  labels:
    component: "influxdb"
  atomic: true
  force: true
  recreatePods: true
  needs: ["{{ .Environment.Name }}-influxdb/influxdb"]
  values:
  - "../values/influxdb/{{ .Environment.Name }}/influxdb-prometheus-rules.yaml.gotmpl"
```

```
$ cat values/influxdb/prod/influxdb-prometheus-rules.yaml.gotmpl
resources:
  - apiVersion: monitoring.coreos.com/v1
    kind: PrometheusRule
    metadata:
      labels:
        role: alert-rules
        app: influxdb
      name: prometheus-influxdb-rules
    spec:
...
```

Szenario 1.
Managed Staged Environments.

Anforderungen

- Ziel: 3 „fast gleiche Umgebungen“: dev, preprod, prod
- 2 AKS-Cluster (1x für dev+preprod, 1x für prod)
- 3 Gruppen von Rollouts:
  - Cluster-Infrastruktur (je Cluster; nginx-ingress, cert-manager, external-dns, …)
  - Umgebungsinfrastuktur (je Umgebung; Kafka, InfluxDB, …)
  - Anwendungen/SCS (je Umgebung)
- Es sollen definierte Versionsstände durch die Umgebungen propagiert werden.
- Bugfixes sollen automatisch auf prod ausgerollt werden.

Lösung

- Automatisierung mit Gitlab CI, Provisionierung von AKS per Terraform
- Cloud-Infrastruktur + Umgebungsinfrastruktur
  - Helmfile mit 2 Environments (devcluster + prodcluster)
  - Helmfile mit 3 Environments (dev, preprod, prod)
  - Umgebungen definieren Chart-Versionen für unabhängige Updates
  - raw-Charts in for-loops um Config und Secrets für die Anwendungen zu verteilen
- Anwendungen
  - Automatische Bugfix-Rollouts durch Semver Range (~1.2.0)
  - DRY durch Release Template (für Kafka- und DB-Verbindungen)

Szenario 2.
Multi-Tenancy Rollouts

Anforderungen

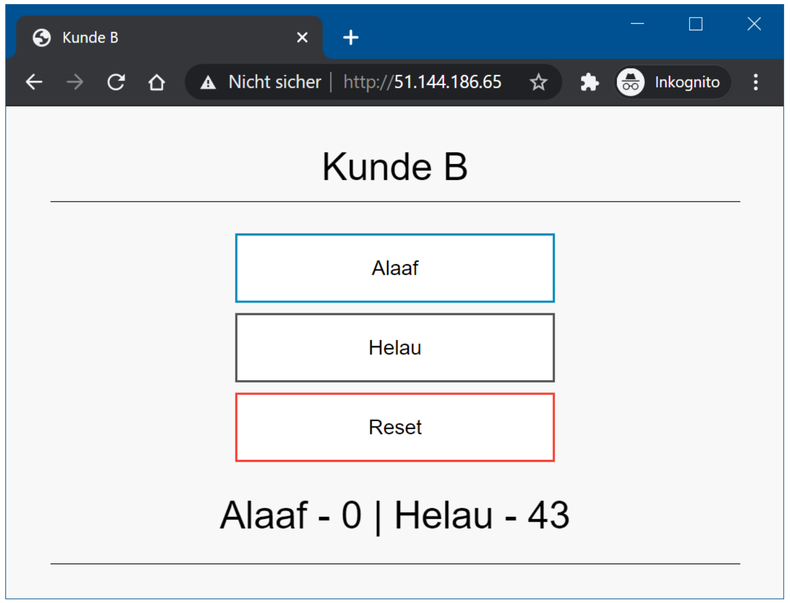
- Grundlage ähnlich zu Szenario 1.

- Plus: Die Anwendung soll isoliert verschiedenen Kunden zur Verfügung gestellt werden.
- Provisionierung soll möglichst automatisch über eine zentrale Konfigurationsdatei erfolgen

Beispiellösung

```
$ cat helmfile.yaml
repositories:
  - name: stable
    url: https://charts.helm.sh/stable
  - name: azure-samples
    url: https://azure-samples.github.io/helm-charts/

environments:
  default:
    values:
      - customers.yaml

releases:
{{- range $key := index .Values "customers" }}
  - name: azure-vote-{{ index $key "id" }}
    namespace: {{ index $key "id" }}
    createNamespace: true
    chart: azure-samples/azure-vote
    version: 0.1.1
    labels:
      customer: {{ index $key "id"}}
      component: app
    values:
      - title: {{ index $key "title" }}
        value1: {{ index $key "option1" }}
        value2: {{ index $key "option2" }}
{{ end }}
```

```
$ cat customers.yaml
customers:
  - id: custa
    title: Kunde A
    option1: Kölsch
    option2: Alt
  - id: custb
    title: Kunde B
    option1: Alaaf
    option2: Helau
```



Fazit

Extrem flexibel dank Templating

Werte aus verschiedenen Quellen beziehen

Ein Tool für alles

Einfaches Secrets Management

Danke! Fragen?

Tammo van Lessen
tammo.van-lessen@innoq.com
+49 171 2774 106
@taval

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin
Germany
+49 2173 3366-0

Ludwigstr. 180E
63067 Offenbach
Germany
+49 2173 3366-0

Kreuzstr. 16
80331 München
Germany
+49 2173 3366-0

Hermannstrasse 13
20095 Hamburg
Germany
+49 2173 3366-0

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
+41 41 743 0116