Helm Paketmanagement

Agenda

1. Helm Grundlagen
    - Installation von kubectl unter Linux
    - Installation von helm unter Linux
    - Installation bash completion
2. Grundlagen
    - Feature / No-Features von Helm
    - TopLevel Objekte
3. Helm-Befehle und -Funktionen
    - Repo einrichten
    - Chart runterladen und evtl. entpacken und bestimmte Version
    - Suche in Repo und Artifacts Hub
    - Anzeigen von Informationen aus dem Chart von Online
    - Upgrade und auftretende Probleme
4. Helm Repository
    - Die wichtigsten Repo-Befehle
5. Struktur von Helm - Charts
    - Überblick
6. Grundlagen Helm-Charts
    - Testumgebung und Spaces (2 Themen)
7. Erstellen von Helm-Charts
    - Erstellen eines Guestbooks
    - Hooks für Guestbook erstellen
    - Dependencies/Abhängigkeiten herunterladen
    - Einfaches Testen
    - Input Validierung innerhalb von templates
    - Advanced Testing mit chart-testing
    - Chart auf github veröffentlichen
8. FlowControl Helm-Charts (if,with,range)
    - if
    - with
    - range
9. Sicherheit von helm-Chart
    - Grundlagen / Best Practices
    - Security Encrypted Passwords in helm
10. Testing in Helm-Charts

- Testing in/von helm - charts

11. Durchführung von Upgrades und Rollbacks von Anwendungen

12. Helm in Continuous Integration / Continuous Deployment (CI/CD) Pipelines

13. Tipps & Tricks

    - Set namespace in config of kubectl
    - Create Ingress Redirect

14. Integration mit anderen Tools

    - yamllint für Syntaxcheck von yaml - Dateien

15. Troubleshooting und Debugging

    - helm template --validate - gegen api-server testen

Helm Grundlagen

Installation von kubectl unter Linux

Walkthrough (Start with unprivileged user like training or kurs)

```
sudo su -
```

```
## Get current version
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
## install the kubectl to the right directory
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

Installation von helm unter Linux

Walkthrough (Start as unprivileged user, e.g. training or kurs)

```
sudo su -
```

```
curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
chmod 700 get_helm.sh
./get_helm.sh
```

```
exit
```

Reference:

- https://helm.sh/docs/intro/install/

Installation bash completion

```
sudo su -
helm completion bash > /etc/bash_completion.d/helm
exit
## z.B.
su - tln11
```

Grundlagen

Feature / No-Features von Helm

- Sortiert, die Manifeste bzw. Objekte bereits automatisch in der richtigen Reihenfolge für das Anwenden (apply) gegen den Server (Kube-Api-Server)

TopLevel Objekte

.Chart

- Zieht alle Infomationen aus der Chart.yaml

- Alle Eigenschaften fangen mit einem grossen Buchstaben, statt klein wie im Chart, z.B. .Chart.Name

.Values

- Ansprechen der Values bzw. Default Values

.Release

- Ansprechen aller Eigenschaften aus der Release z.B. Release.Name

Helm-Befehle und -Funktionen

Repo einrichten

```
helm repo list
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo remove bitnami
helm repo update
```

Chart runterladen und evtl. entpacken und bestimmte Version

```
## Vorher müssen wir den Repo-Eintrag anlegen
helm repo add bitnami https://charts.bitnami.com/bitnami

## Lädt die letzte herunter
helm pull bitnami/mariadb

## Lädt bestimmte chart-version runter
helm pull bitnami/mariadb --version 12.1.6
## evtl. entpacken wenn gewünscht
## tar xvf mariadb-12.1.6.tgz

## Schnelle Variante
helm pull bitnami/mariadb --version 12.1.6 --untar
```

Suche in Repo und Artifacts Hub

Suche im hub

```
helm search hub mariadb
## Zeige kompletten Zeilen an ohne abszuschneiden
helm search hub mariadb --max-col-width=0
```

Suche im Repo

```
## Suche nach allen Charts, die mariadb im Namen oder der Beschreibung tragen
helm search repo mariadb

## Zeige alle Version von charts an, die mit bitnami/mariadb beginnen
helm search repo bitnami/mariadb --versions
```

Anzeigen von Informationen aus dem Chart von Online

```
helm show values bitnami/mariadb
helm show values bitnami/mariadb | grep -B 20 -i "image:"
```

```
## Zeigt Chart-Definition, Readme usw. (=alles) an
helm show all bitnami/mariadb
```

```
helm show readme
helm show readme bitnami/mariadb
helm show chart bitnami/mariadb
```

Upgrade und auftretende Probleme

Die wichtigsten Repo-Befehle

```
helm repo list
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo remove bitnami
helm repo update
```

Struktur von Helm - Charts

Überblick

Komponenten von Helm-Charts

Chart.yml

Chart.lock (wird automatisch generiert)

templates/

_helper.tpl

- Enthält snippet die mit include oder templates inkludiert werden können
- Konvention der Snippets mit define ChartName.Eigenschaft z.B. botti.fullname

NOTES.txt

- Wird ausgegeben, nachdem das Chart installiert wurde
    - oder:

```
## after installation
## helm install my-botti -n my-application --create-namespace botti
helm get -n my-application notes my-botti
```

charts/

- Hier werden die abhängigen charts runtergeladen und als .tgz

Grundlagen Helm-Charts

Testumgebung und Spaces (2 Themen)

Explanation

- {{- -> trim on left side
- -}} -> trim on right side
- trim tabs, whitespaces a.s.o. (see ref)

Walkthrough

```
## When ever we encounter error while parsing yaml, we can use comment !!!
helm create testenv
cd testenv/templates
rm -f *.yaml
```

```
nano test.yaml
```

```
## "{{23 -}} < {{- 45}}"
```

```
helm template ..
helm template --debug ..
```

Reference:

- https://pkg.go.dev/text/template#hdr-Text_and_spaces

Erstellen von Helm-Charts

Erstellen eines Guestbooks

Step 1: Create namespace and structure of helm chart

```
cd
```

```
helm create guestbook
## now we have in folder "guestbook"
## charts/
## Chart.yaml
## templates
## values.yaml
```

Step 2: Explore templates folder and cleanup

```
cd templates
ls -la
rm -fR tests
```

Step 3: Explore the Chart.yaml

```
cd ..
cat Chart.yaml
```

```
## type: Application or Library # please explain !
## dependencies - what other charts are needed - we will download them by helm command
and they will be put in the charts - folder
```

Step 4: Add redis as dependency

```
## find the redis chart
helm search hub --max-col-width=0  redis | grep bitnami
## adding the repo for bitnami
helm repo add bitnami https://charts.bitnami.com/bitnami
```

```
## now find the availabe versions (these are the chart versions
helm search repo redis --versions
```

```
nano Chart.yaml
```

```
## now add the dependency-block at the end of the file
dependencies:
  - name: redis
    version: "17.14.x"  # quotes are important here
    repository: https://charts.bitnami.com/bitnami
```

```
## Save the file and leave nano:
STRG + o + RETURN -> then -> STRG + x
```

```
cd ..
helm dependency update guestbook
```

```
## explore the newly populated folder
cd guestbook/charts
ls -la
cd ../..
```

Step 5: Modifying the values.yaml file

```
## the version might have changed since i wrote this / adjust
helm show values charts/redis-17.14.5.tgz
## what are the service name of the redis leader and the redis follower
helm show values charts/redis-17.14.5.tgz | grep -B 4 -i fullnameoverride
```

```
## the service names need to be adjusted, add the following to the values.yaml
## The guestbook - application needs the redis - services called. redis-leader and
redis-follower
```

```
cd
cd guestbook
nano values.yaml
```

```
## add at the end of the file
redis:
  fullnameOverride: redis

## enable unauthorized access to redis
  usePassword: false
## Disable AOF persistence
  configmap: |-
    appendonly no
```

```
## save file and exit
STRG + o + ENTER -> then -> STRG + x
```

```
## now check, if this really worked
cd
cd guestbook
helm template . | grep -A 20 master/service
```

Setting the right repo and the right version

```
cd
cd guestbook
cat templates/deployment.yaml
```

```
Welche Version brauche ich ?
https://kubernetes.io/docs/tutorials/stateless-application/guestbook/#creating-the-
guestbook-frontend-deployment
## Stand 2023-08-08
gcr.io/google_samples/gb-frontend:v5
```

```
## nano Chart.yaml
## korrigieren
appVersion: "v5"
```

```
## nano values.yaml
image:
  repository: gcr.io/google_samples/gb-frontend
```

Step 6: Changing LoadBalancer to NodePort

```
## nano values.yaml
service:
  type: NodePort
  port: 80
```

Step 7: Installing helm chart

```
helm install my-guestbook guestbook -n jochen --create-namespace
kubectl -n jochen get all
```

Reference:

- https://kubernetes.io/docs/tutorials/stateless-application/guestbook/

Hooks für Guestbook erstellen

Step 1:

```
cd
mkdir guestbook/templates/backup
touch guestbook/templates/backup/persistentVolume-claim.yaml
touch guestbook/templates/backup/job.yaml
```

Step 2: persistentvolumeclaim.yaml und job bevölkern

```
## nano guestbook/templates/backup/persistentVolume-claim.yaml

{{- if .Values.redis.master.persistence.enabled }}
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: redis-data-{{ .Values.redis.fullnameOverride }}-master-0-backup-{{ sub
.Release.Revision 1 }}
  labels:
    {{- include "guestbook.labels" . | nindent 4 }}
  annotations:
    "helm.sh/hook": pre-upgrade
    "helm.sh/hook-weight": "0"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: {{ .Values.redis.master.persistence.size }}
{{- end }}
```

```
## nano guestbook/templates/backup/job.yaml
{{- if .Values.redis.master.persistence.enabled }}
apiVersion: batch/v1
kind: Job
metadata:
  name: {{ include "guestbook.fullname" . }}-backup
  labels:
    {{- include "guestbook.labels" . | nindent 4 }}
  annotations:
    "helm.sh/hook": pre-upgrade
    "helm.sh/hook-delete-policy": before-hook-creation,hook-succeeded
    "helm.sh/hook-weight": "1"
```

```
spec:
  template:
    spec:
      containers:
        - name: backup
          image: redis:alpine3.11
          command: ["/bin/sh", "-c"]
          args: ["redis-cli -h {{ .Values.redis.fullnameOverride }}-master save && cp
/data/dump.rdb /backup/dump.rdb"]
          volumeMounts:
            - name: redis-data
              mountPath: /data
            - name: backup
              mountPath: /backup
      restartPolicy: Never
      volumes:
        - name: redis-data
          persistentVolumeClaim:
            claimName: redis-data-{{ .Values.redis.fullnameOverride }}-master-0
        - name: backup
          persistentVolumeClaim:
            claimName: redis-data-{{ .Values.redis.fullnameOverride }}-master-0-
backup-{{ sub .Release.Revision 1 }}
{{- end }}
```

Step 3: pre-rollback hook erstellen

```
mkdir guestbook/templates/restore
touch guestbook/templates/restore/job.yaml
```

```
## nano guestbook/templates/restore/job.yaml
{{- if .Values.redis.master.persistence.enabled }}
apiVersion: batch/v1
kind: Job
metadata:
  name: {{ include "guestbook.fullname" . }}-restore
  labels:
    {{- include "guestbook.labels" . | nindent 4 }}
  annotations:
    "helm.sh/hook": pre-rollback
    "helm.sh/hook-delete-policy": before-hook-creation,hook-succeeded
spec:
  template:
    spec:
      containers:
        - name: restore
          image: redis:alpine3.11
          command: ["/bin/sh", "-c"]
          args: ["cp /backup/dump.rdb /data/dump.rdb &&
            redis-cli -h {{ .Values.redis.fullnameOverride }}-master debug restart ||
true"]
```

```
          volumeMounts:
            - name: redis-data
              mountPath: /data
            - name: backup
              mountPath: /backup
      restartPolicy: Never
      volumes:
        - name: redis-data
          persistentVolumeClaim:
            claimName: redis-data-{{ .Values.redis.fullnameOverride }}-master-0
        - name: backup
          persistentVolumeClaim:
            claimName: redis-data-{{ .Values.redis.fullnameOverride }}-master-0-
backup-{{ .Release.Revision }}
{{- end }}
```

Reference

- https://helm.sh/docs/topics/charts_hooks/

Dependencies/Abhängigkeiten herunterladen

Voraussetzung:

- Dependencies sind in Chart.yml eingetragen
- Achtung: Version ist die Version des Charts nicht der App !!!

Das 1. Mal

```
## 1. Alle Abhängigkeiten werden in Form von .tgz - Archiven heruntergeladen
     -> in das charts - Verzeichnis
## 2. Eine Chart.lock - datei wird erstellt. (hält den aktuellen Stand fest)
## helm dependancy update $CHART_PATH
## botti erklärt sich gleich unten im Walkthrough
helm dependancy update botti
```

Das 2. Mal (wenn Chart.lock vorhanden, aber charts/ muss nicht da sein

```
helm dependancy build botti
```

List all dependencies

```
helm dependancy list botti
```

Walkthrough

```
cd
helm create botti
```

```
cd botti
## add dependency
```

```
nano Chart.yml
```

```
## at the end of the file add

## After that save and exit STRG + O + ENTER , STRG  + X
```

```
## Update to download depdendancies
cd ..
helm dependency update botti
cd botti/charts
ls -la
cd ../../
```

```
## Add repo to be able to do helm dependency build
rm -fR botti/charts
## Chart.lock needs to be there
ls -la botti/Chart.lock

## Add repo / needs to be there, otherwice
helm repo add bitnami https://charts.bitnami.com/bitnami
helm dependency build botti
```

Einfaches Testen

Input Validierung innerhalb von templates

Walkthrough

```
cd
helm create inputtest
cd inputtest
cd templates/
rm d* h* i* servicea*
rm -fR tests
```

```
## nano service.yaml mit folgendem Inhalt
apiVersion: v1
kind: Service
metadata:
  name: {{ include "inputtest.fullname" . }}
  labels:
    {{- include "inputtest.labels" . | nindent 4 }}
spec:
{{- $serviceType := list "ClusterIP" "NodePort" }}
{{- if has .Values.service.type $serviceType }}
  type: {{ .Values.service.type }}
{{- else }}
  {{- fail "value 'service.type' must be either 'ClusterIP' or 'NodePort'" }}
{{- end }}
  ports:
    - port: {{ .Values.service.port }}
```

```
      targetPort: http
      protocol: TCP
      name: http
  selector:
    {{- include "inputtest.selectorLabels" . | nindent 4 }}
```

```
cd
cd inputtest
nano values.yaml
```

```
service:
  type: nodePorty # written wrong
  port: 80
```

```
cd
helm template --debug inputtest
## and eventually also test against server
helm template inputtest --validate
```

Advanced Testing mit chart-testing

Reference

- https://github.com/helm/chart-testing/
- https://github.com/helm/chart-testing/blob/main/doc/ct_install.md

Chart auf github veröffentlichen

Prep

```
Create new public repo with README.md
Go to Settings -> Pages -> an enable for branch "main"
git clone the repo locally
```

Locally pack, index and upload it.

```
git clone https://github.com/jmetzger/chart-test.git
## guestbook must be present as folder with charts
helm package guestbook
cp guestbook-0.1.0.tgz chart-test/
helm repo index chart-test/
git add .
git commit -m "initial release"
git push -u origin main
```

Work with it

```
helm repo add githubrepo https://jmetzger.github.io/chart-test/
helm search repo guestbook
helm repo list
helm pull githubrepo/guestbook
```

FlowControl Helm-Charts (if,with,range)

if

Prepare (if not done yet)

```
helm create testenv
cd testenv/templates
rm -f *.yaml
```

Step 1: Simple inline

```
## Adjust values.yaml file accordingly
favorite:
  food: PIZZA
  drink: coffee
```

```
nano iftest.yaml
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
  drink: {{ .Values.favorite.drink | default "tea" | quote }}
  food: {{ .Values.favorite.food | upper | quote }}
  {{ if eq .Values.favorite.drink "coffee" }}mug: "true"{{ end }}
```

```
helm template ..
```

Step 2: (Problem) That will produce food: "PIZZA"mug: "true" because it consumed newlines on both sides.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
  drink: {{ .Values.favorite.drink | default "tea" | quote }}
  food: {{ .Values.favorite.food | upper | quote }}
  {{- if eq .Values.favorite.drink "coffee" -}}
  mug: "true"
  {{- end -}}
```

Step 3: Other solution

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
  drink: {{ .Values.favorite.drink | default "tea" | quote }}
  food: {{ .Values.favorite.food | upper | quote }}
  {{- if eq .Values.favorite.drink "coffee"}}{{ nindent 2 "mug: true" }}
  {{- end }}
```

Step 4: Probably the best solution

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
  drink: {{ .Values.favorite.drink | default "tea" | quote }}
  food: {{ .Values.favorite.food | upper | quote }}
  {{- if eq .Values.favorite.drink "coffee"}}
  {{ "mug: true" }}
  {{- end }}
```

Reference

- https://helm.sh/docs/chart_template_guide/control_structures/

with

Walkthrough

Preparation

```
helm create testenv
cd testenv/templates
rm -fR *.yaml
```

```
## vi values.yml
## Adjust values.yaml file accordingly
favorite:
  food: PIZZA
  drink: coffee
```

Step 1:

```
## nano cm.yaml
```

```
apiVersion: v1
kind: ConfigMap
```

```
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
  {{- with .Values.favorite }}
  drink: {{ .drink | default "tea" | quote }}
  food: {{ .food | upper | quote }}
  {{- end }}
```

Step 2a: Does not work because scope does not fit

```
  {{- with .Values.favorite }}
  drink: {{ .drink | default "tea" | quote }}
  food: {{ .food | upper | quote }}
  release: {{ .Release.Name }}
  {{- end }}
```

Step 2b: Solution 1: (Outside with)

```
  {{- with .Values.favorite }}
  drink: {{ .drink | default "tea" | quote }}
  food: {{ .food | upper | quote }}
  {{- end }}
  release: {{ .Release.Name }}
```

Step 2c: Changing the scope

```
  {{- with .Values.favorite }}
  drink: {{ .drink | default "tea" | quote }}
  food: {{ .food | upper | quote }}
  release: {{ $.Release.Name }}
  {{- end }}
```

range

Preparation

```
helm create testenv
cd testenv/templates
rm -f *.yaml
```

Step 1: Values.yaml

```
favorite:
  drink: coffee
  food: pizza
pizzaToppings:
  - mushrooms
  - cheese
```

```
  - peppers
  - onions
```

Step 2 (Version 1):

```
## nano cm.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
  {{- with .Values.favorite }}
  drink: {{ .drink | default "tea" | quote }}
  food: {{ .food | upper | quote }}
  {{- end }}
  toppings: |-
    {{- range .Values.pizzaToppings }}
    - {{ . | title | quote }}
    {{- end }}
```

Step 3 (Version 2 - works as well)

- Accessing the parent scope

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
  {{- with .Values.favorite }}
  drink: {{ .drink | default "tea" | quote }}
  food: {{ .food | upper | quote }}
  toppings: |-
    {{- range $.Values.pizzaToppings }}
    - {{ . | title | quote }}
    {{- end }}
  {{- end }}
```

Sicherheit von helm-Chart

Grundlagen / Best Practices

- https://sysdig.com/blog/how-to-secure-helm/

Security Encrypted Passwords in helm

Reference:

- https://www.thorsten-hans.com/encrypted-secrets-in-helm-charts/
- https://github.com/jkroepke/helm-secrets

Alternative: SealedSecrets

- https://dev.to/timtsoitt/argo-cd-and-sealed-secrets-is-a-perfect-match-1dbf

Testing in Helm-Charts

Testing in/von helm - charts

Walkthrough

```
helm create demo
helm install demo demo
helm test demo
```

Reference

- https://helm.sh/docs/topics/chart_tests/

Durchführung von Upgrades und Rollbacks von Anwendungen

Helm in Continuous Integration / Continuous Deployment (CI/CD) Pipelines

Tipps & Tricks

Set namespace in config of kubectl

```
kubectl create ns mynamespace
kubectl config set-context --current --namespace=mynamespace
```

Create Ingress Redirect

```
cd
helm create testprojekt
cd testprojekt
cd templates
```

```
mkdir routes/
cd routes
nano 01-redirect.yaml
```

Schritt 1: Mit der Basis anfangen

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/permanent-redirect: https://www.google.de
    nginx.ingress.kubernetes.io/permanent-redirect-code: "308"
  creationTimestamp: null
```

```
  name: destination-home
  namespace: my-namespace
spec:
  rules:
  - host: web.training.local
    http:
      paths:
      - backend:
          service:
            name: http-svc
            port:
              number: 80
        path: /source
        pathType: ImplementationSpecific
```

Schritt 2: values - file mit eigenen Werten ergänzen (Default - Werte)

```
## cd ../..
## nano values.yaml
## Zeilen ergänzt.
## Achtung: Eigenschaft UNBEDINGT ! ohne "-"
myRedirect:
  url: "http://www.google.de"
  code: 302
```

Schritt 3: Variablen aus values in template einbauen

```
cd templates/routes
```

```
## nano 01-redirect.yaml
## Neue Fassung: Alle Änderungen beginnen mit Platzhalter - Zeichen {{

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/permanent-redirect: {{ .Values.myRedirect.url }}
    nginx.ingress.kubernetes.io/permanent-redirect-code: {{ .Values.myRedirect.code |
quote }}
  creationTimestamp: null
  name: destination-home
  namespace: my-namespace
spec:
  rules:
  - host: web.training.local
    http:
      paths:
      - backend:
          service:
            name: http-svc
```

```
            port:
              number: 80
        path: /source
        pathType: ImplementationSpecific
```

Schritt 4: Test mit Default - Werten aus values.yaml

```
helm template ../..
## achten auf ausgaben von Ingress
helm template ../.. | grep -A 40 "kind: Ingress"
```

Schritt 5: Default - Werte überschreibung für Produktion mit speziellen prod-values.yaml (Name beliebig)

```
## Empfehlung: ausserhalb des Charts anlegen
cd
nano prod-values.yaml
```

```
myRedirect:
  url: "http://www.stiftung-warentest.de"
```

```
## Testen wie folgt
helm template -f prod-values.yaml testprojekt
## oder aber auch testen mit validate
helm template --validate -f prod-values.yaml testprojekt
## oder aber direkt release installation
helm install --dry-run -f prod-values.yaml testprojekt
```

Integration mit anderen Tools

yamllint für Syntaxcheck von yaml - Dateien

```
apt install -y yamllint
```

Troubleshooting und Debugging

helm template --validate - gegen api-server testen

How ?

```
helm template guestbook --validate
```