Helm Package Management

Agenda

1. Helm Installation

   - Installation of kubectl under Linux
   - Installation of helm under Linux
   - Installation bash completion (helm)

2. Background

   - Where to find helm charts

3. Install helm-chart

   - Install/Upgrade mariadb / bitnami

4. Helm Repository

   - The most important helm commands

5. Basics

   - Feature / No-Features von Helm
   - TopLevel objects / Built-In variables

6. Helm Charts best practices

   - Development
   - Naming conventions a.s.o.

7. Helm-Commands

   - Setup repo
   - Download specific version of chart and unpack
   - Search in Repo und Artifacts Hub
   - Show informations of charts online
   - Upgrades and occuring problems

8. Structure of a Helm - Charts

   - Overview

9. Basics of Helm-Charts

   - Spaces in templates and how to test (2 topics)

Development

1. Creation of Helm-Charts

   - Creation of a Guestbooks
   - Create Hook for guestbook
   - Downloads dependencies herunterladen
   - Simple Testing
   - Input validation within templates
   - Advanced Testing with chart-testing

- Publish chart to github

2. FlowControl Helm-Charts (if,with,range)

- if
- with
- range

3. Security of helm-charts

- Security Encrypted Passwords in helm

4. Testing in Helm-Charts

- Testing in/of helm - charts

5. Tipps & Tricks

- Set namespace in config of kubectl
- Create Ingress Redirect

6. Integration with other tools

- yamllint for syntaxcheck of yaml - files

7. Troubleshooting und Debugging

- helm template --validate - testing against api-server

8. Security of helm-Chart

- Basics / Best Practices

Helm Installation

Installation of kubectl under 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 of helm under 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 (helm)

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

Background

Where to find helm charts

- https://artifacthub.io

Install helm-chart

Install/Upgrade mariadb / bitnami

Install

```
cd
chmod g-r,o-r .kube/config
helm repo add bitnami https://charts.bitnami.com/bitnami
helm -n jochen2 install my-mariadb bitnami/mariadb --version 19.0.5 --create-namespace
```

```
## OR: upgrade and if not install -> install
helm -n jochen2 upgrade my-mariadb bitnami/mariadb --install --version 19.0.5 --
create-namespace
```

Upgrade to primary / secondary

```
cd
mkdir manifests
cd manifests/
mkdir mariadb-release
cd mariadb-release
echo "archtitecture: replication" > values.yaml
```

```
architecture: replication
auth:
  rootPassword: zfGb7nFsMZ
  replicationPassword: myreplication
  forcePassword: true
```

```

```

helm -n jochen2 upgrade my-mariadb bitnami/mariadb --install --version 19.0.5 --create-namespace

```


## Helm Repository

### The most important helm commands


### Installation
```

helm repo add bitnami https://charts.bitnami.com/bitnami helm -n jochen2 install my-mariadb bitnami/mariadb --version 19.0.5 --create-namespace helm -n jochen2 upgrade my-mariadb bitnami/mariadb --install --version 19.0.5 --create-namespace

```

### After installation
```

show all realeases

helm -n jochen list

all namespaces

helm list -A

```
```

get specific information from release

use value file (if there was one)

helm -n jochen get values my-mariadb helm -n jochen get notes my-mariadb helm -n jochen get manifest my-mariadb

```

### Communicating with chart
```

helm show values bitnami/mariadb

```


## helm repo commands
```

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

```

### See all versions of a chart
```

helm search repo mariadb -l

```

## Basics

### Feature / No-Features von Helm


  * Sorts the manifests / Objects automatically for the usage with apply against the
kube-api-server


### Which order is it ?

  * see also Internals [Helm Sorting Objects](/helm/internals.md)


### TopLevel objects / Built-In variables


### .Chart

 * Shows all informaton of the Chart.yaml
 * Alle properties start with a capital (although lower in Chart.yaml), e.g.
.Chart.Name

### .Values
```

```
 * Reading of data from  Values-File or  Default Values

### .Release

 * Get specific properties from the Relese itself, e.g. Release.Name

## Helm Charts best practices

### Development

  * https://helm.sh/docs/howto/charts_tips_and_tricks/

### Naming conventions a.s.o.

  * https://helm.sh/docs/chart_best_practices/

## Helm-Commands

### Setup repo


### Installation
```

helm repo add bitnami https://charts.bitnami.com/bitnami helm -n jochen2 install my-mariadb bitnami/mariadb --version 19.0.5 --create-namespace helm -n jochen2 upgrade my-mariadb bitnami/mariadb --install --version 19.0.5 --create-namespace

```

### After installation
```

show all realeases

helm -n jochen list

all namespaces

helm list -A

```

```

get specific information from release

use value file (if there was one)

helm -n jochen get values my-mariadb helm -n jochen get notes my-mariadb helm -n jochen get manifest my-mariadb

```

### Communicating with chart
```

helm show values bitnami/mariadb

```
## helm repo commands
```

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

```
### See all versions of a chart
```

helm search repo mariadb -l

```
### Download specific version of chart and unpack
```

First we need to set the repo - entry

helm repo add bitnami https://charts.bitnami.com/bitnami

download the latest availabe chart

helm pull bitnami/mariadb

Downloads a specific version

helm pull bitnami/mariadb --version 12.1.6

untar it if wanted

tar xvf mariadb-12.1.6.tgz

Quick version

helm pull bitnami/mariadb --version 12.1.6 --untar

```
### Search in Repo und Artifacts Hub


### Search in hub
```

helm search hub mariadb

Show complete lines without cutting it of

helm search hub mariadb --max-col-width=0

```
### Search in Repo
```

Search for all charts, that have mariadb in name or description

helm search repo mariadb

Show all versions of charts, that start with bitnami/mariadb

helm search repo bitnami/mariadb --versions

```
### Show informations of charts online
```

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

```
```

Show Chart-Definitions, Readme a.s.o. (=everything) - templates are missing / but saved in data in etcd

helm show all bitnami/mariadb

```
```

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

```
### Upgrades and occuring problems


### Walkthrough

#### Step 1: Upgrade
```

helm install my-mariadb bitnami/mariadb -f db/prod-values.yaml --set auth.database=db1 helm get values my-mariadb

prod-values will be overwritten, because of using --set -> defaults to using switch --reset-values (in the background)

for upgrade of mariadb, you will need passwort

export MARIADB_ROOT_PASSWORD=$(kubectl get secret --namespace "jochen" my-mariadb -o jsonpath="{.data.mariadb-root-password}" | base64 -d) helm upgrade my-mariadb bitnami/mariadb --set auth.database=db2 --set auth.rootPassword=$MARIADB_ROOT_PASSWORD helm get values my-mariadb

if you want to reuse the values from last release -> set --reuse-values

helm upgrade my-mariadb bitnami/mariadb --reuse-values --set auth.rootPassword=$MARIADB_ROOT_PASSWORD helm get values my-mariadb

```
#### Step 2: Rollback
```

helm history my-mariadb helm rollback my-mariadb 1

```
### Problems with upgrade

  * in some circumstances --reset-values are set
  * in come circumstances --reuse-values are set

#### Default strategy:

  * if you NOT set any values during upgrade, helm implicitly uses --reuse-values
strategy
  * if you ARE setting values during upgrade, helm implicitly uses --reset-values
strategy

### Strategy can get enforce

  * --reuse-strategy or --reuse-values

### Best choice (SOLUTION) , if you want to have values from the new chart version
```

helm get values example-loki > prev-values.yaml

Values from old chart are merge with new chart, with merge of set on top

helm upgrade example-loki -f prev-values.yaml --set grafana.enabled=true

```
### Reference:

  * https://shipmight.com/blog/understanding-helm-upgrade-reset-reuse-values

## Structure of a Helm - Charts

### Overview


### Components of helm charts

#### Chart.yml

#### Chart.lock (generated automatically)

##### _helper.tpl

  * Not considered, parsed a manifests
  * Hold snippets (named templates) can be included with "include" (Preferred) or
```

```
"template"
  * Best practice: name of named template with  define ChartName.Property z.B.
botti.fullname

##### NOTES.txt

  * is shown, after installation of chart with helm install
    * or: with helm get notes

```

after installation

helm install my-botti -n my-application --create-namespace botti

helm get -n my-application notes my-botti

```

#### charts/

  * Hier dependencies are downloaded which are given in Charts.yml




## Basics of Helm-Charts

### Spaces in templates and how to test (2 topics)


### 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

## Creation of Helm-Charts

### Creation of a 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

```
```

Which version do it need ?

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/

### Create Hook for guestbook


### 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 setup job
```

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/

### Downloads dependencies herunterladen
### Voraussetzung:

  * Dependencies are in Chart.yml
  * Achtung: Version ist the version of the chart not the App !!!

### The first time
```

1. All dependencies are downloaded as .tgz - archives

```
 -> into the chart folder
```

2. Eine Chart.lock - datei wird erstellt. (hält den aktuellen Stand fest)

helm dependancy update $CHART_PATH

Explained beneath in the Walkthrough

helm dependancy update botti

```

### The 2. time (if Chart.lock is there, but charts/ does not need to be there
```

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

```

### Simple Testing


### Walkthrough
```

helm create demo helm install demo demo helm test demo

```

### Reference

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

### Input validation within templates


### Walkthrough
```

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

```

```

nano service.yaml with the following content

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 with chart-testing


### Reference

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

### Publish chart to github


### 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 }}

```
## Security of helm-charts

### 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/of helm - charts


### Walkthrough
```

helm create demo helm install demo demo helm test demo

```

### Reference

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

## 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 with other tools

### yamllint for syntaxcheck of yaml - files

```

apt install -y yamllint

```



## Troubleshooting und Debugging

### helm template --validate - testing against api-server


### How ?
```

helm template guestbook --validate

```

## Security of helm-Chart

### Basics / Best Practices



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