



ITCS MÜNCHEN

K8S IN A NUTSHELL

28.10.2022

WLAN

- WLAN SSID: ITCS2022
- WLAN Passwort: ITCS2022

VORSTELLUNG

- Manfred Schreiber
 - 2005: BSc Informatik
 - Seit 2013 bei Jambit
 - Senior Systems Architect



- Michael Beham
 - 2017: MSc Informatik
 - Seit 2017 bei Jambit
 - Senior Systems Architect



JAMBIT



Wer sind wir?

- Software & IT-Dienstleister
- 450 Mitarbeitern an 5 Standorten
- Vielfältige, anspruchsvolle Projekte aus unterschiedlichsten Branchen

Warum Projekte?

- Unterschiedliche Tech-Stacks
- Lerne unterschiedliche Firmen und deren Kultur kennen
- Es wird nicht langweilig

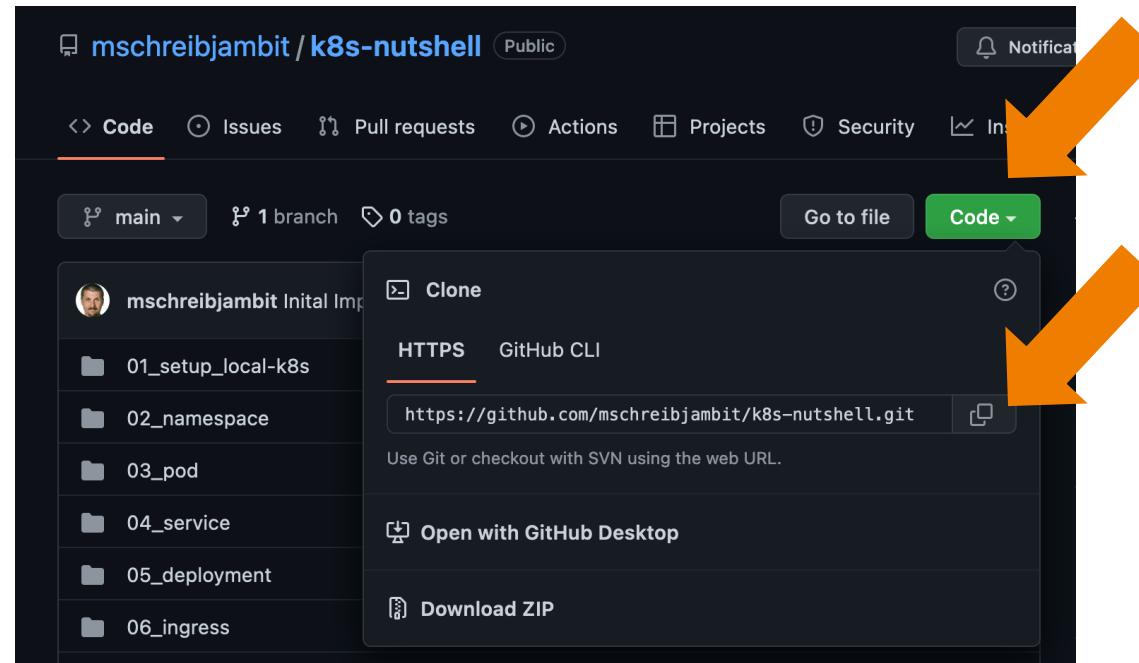
WAS WIR BRAUCHEN

- Das habt ihr
 - Docker bzw. Docker Desktop
 - Linux, MacOS oder Windows WSL2
- Podman etc. ist ein Problem und funktioniert mit k3d nicht



GIT CLONE UND TOOLS

- Gehe mit deinem Browser auf die URL:
<https://jamb.it/itcs>
- Bitte clone das Git-Repo oder lade dir die Zip-Datei herunter.





SETUP

- Im Repo findet sich eine `setup.sh` Datei.
Führe folgendes aus:

`source setup.sh`

- Bei Bedarf nochmal in weiteren Shells ausführen!



BEGRIFFSKLÄRUNG

- k8s: Abkürzung für Kubernetes
- k3d / k3s: Tool für einen kleinen Kubernetes-Cluster
- Node: Jeder K8s besteht aus ein oder mehreren Nodes mit unterschiedlichen Rollen
- Worker: Node, auf der die Nutzlast-Container laufen
- kubectl: CLI-Tool
- k9s: TUI-Tool

```
1 ---
2 apiVersion: v1
3 kind: Namespace
4 metadata:
5   name: demo
6
7 ---
8 apiVersion: v1
9 kind: Namespace
10 metadata:
11   name: demo2
12
```

NAMESPACES

- Strukturierungsebene innerhalb eines Clusters
- Fast alle Ressourcen (z.B. Container) sind genau einem Namespace zugeordnet



NAMESPACE

- Im Ordner 02_namespace liegt die passende YAML-Datei
- Installieren mit
kubectl apply -f namespace.yaml
- Überprüfen mit
kubectl get namespaces

```
1 ---
2 apiVersion: v1
3 kind: Pod
4 metadata:
5   name: helloworld
6   namespace: demo
7   labels:
8     app: helloworld
9 spec:
10  containers:
11    - name: helloworld
12      image: docker.io/helloworld:v0.1.0
13    ports:
14      - containerPort: 8080
15
```

PODS

- kleinste ausführbare Einheit
- bestehen aus einem oder mehreren Containern
- gebunden an einen Namespace

POD

- Im Ordner 03_pod liegt die passende YAML-Datei
- Installieren mit
kubectl apply -f pod.yaml
- Überprüfen mit
kubectl get pods -n demo
- Pod läuft, aber wir können den HTTP-Port nicht erreichen

```
1 ---
2 apiVersion: v1
3 kind: Service
4 metadata:
5   name: helloworld-lb
6   namespace: demo
7 spec:
8   type: LoadBalancer
9   ports:
10  - port: 8080
11    protocol: TCP
12    targetPort: 8080
13 selector:
14   app: helloworld
```

SERVICE

- fasst mehrere Pods zu einem logischen Dienst zusammen
- Load-Balancing über alle zugehörigen Pods
- Ist per DNS im Cluster erreichbar:
`<service-name>`
(im gleichen Namespace)
(hier: "helloworld-lb")



SERVICE

- Im Ordner `04_service` liegt die passende YAML-Datei
- Installieren mit
`kubectl apply -f service.yaml`
- Überprüfen im Browser mit
`http://localhost:8080`
- Pod läuft, Netzwerk funktioniert.
- Aber wie sieht es mit Ausfallsicherheit aus, wenn z.B. ein Node ausfällt?

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   labels:
5     app: helloworld
6   name: helloworld
7   namespace: demo
8 spec:
9   replicas: 1
10  selector:
11    matchLabels:
12      app: helloworld
13  template:
14    metadata:
15      labels:
16        app: helloworld
17    spec:
18      containers:
19        - name: helloworld
20          image: helloworld:v0.1.0
```

DEPLOYMENT

- erzeugt und verwaltet mehrere Pods
- Wrapper um einzelne Pods
- Ermöglicht Skalierung
- Kümmert sich um Ausfälle von Nodes
- Zero-Downtime-Updates

POD VS DEPLOYMENT

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: helloworld
5   namespace: demo
6   labels:
7     app: helloworld
8 spec:
9   containers:
10  - name: helloworld
11    image: docker.io/helloworld:v0.1.0
12    ports:
13      - containerPort: 8080
14    readinessProbe:
15      failureThreshold: 3
16      httpGet:
17        path: /
18        port: 8080
19      periodSeconds: 10
20      successThreshold: 1
21    resources:
22      limits:
23        memory: 100Mi
24      requests:
25        memory: 100Mi
```

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   labels:
5     app: helloworld
6   name: helloworld
7   namespace: demo
8 spec:
9   replicas: 1
10  selector:
11    matchLabels:
12      app: helloworld
13  template:
14    metadata:
15      labels:
16        app: helloworld
17 spec:
18   containers:
19     - name: helloworld
20       image: docker.io/helloworld:v0.1.0
21       ports:
22         - containerPort: 8080
23       readinessProbe:
24         failureThreshold: 3
25         httpGet:
26           path: /
27           port: 8080
28         periodSeconds: 10
29         successThreshold: 1
30       resources:
31         limits:
32           memory: 100Mi
33         requests:
34           memory: 100Mi
```



DEPLOYMENT

- Im Ordner `05_deployment` liegt die passende YAML-Datei
- Installieren mit
`kubectl apply -f deployment.yaml`
- Auch hier kannst du die Verfügbarkeit prüfen
`http://localhost:8080`
- Wie sieht es hier mit der Ausfallsicherheit aus?
- Was passiert, wenn im YAML den Wert für `replicas` verändert?
- Wie kann man sich die Deployments anzeigen lassen? Wie kann man es löschen?

```
1 apiVersion: networking.k8s.io/v1
2 kind: Ingress
3 metadata:
4   name: helloworld-ingress
5   namespace: demo
6 spec:
7   rules:
8     - host: helloworld.127.0.0.1.nip.io
9       http:
10         paths:
11           - backend:
12             service:
13               name: helloworld
14               port:
15                 number: 80
16             path: /
17             pathType: Prefix
```

INGRESS

- Funktion eines HTTP-Proxy
- Mapping auf Apps kann per
 - Domain erfolgen
 - http Pfad erfolgen
- Erweiterung Service:
 - Mehrere Apps hinter IP/Port
 - Kann TLS Terminierung

INGRESS

- Im Ordner 06_ingress liegen die passende YAML-Dateien
- Installieren mit
kubectl apply -f <datei>
- Ist im Browser unter
helloworld.127.0.0.1.nip.io verfügbar
- Warum führt ein direkter Aufruf von 127.0.0.1 zu einem Fehler?
- nip.io kann Zuhause Probleme machen:
Stichwort: DNS-Rebind-Schutz

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   labels:
5     app: helloworld
6   name: helloworld
7   namespace: demo
8 spec:
9   replicas: 1
10  selector:
11    matchLabels:
12      app: helloworld
13 template:
14   metadata:
15     labels:
16       app: helloworld
17 spec:
18   containers:
19     - name: helloworld
20       image: docker.io/helloworld:v0.1.0
21   pods:
22     - containerPort: 8080
23   readinessProbe:
24 [...]
```

DEPLOYMENT BIS INGRESS

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: helloworld
5   namespace: demo
6 spec:
7   ports:
8     - port: 80
9       protocol: TCP
10      targetPort: 8080
11   selector:
12     app: helloworld
```

```
1 apiVersion: networking.k8s.io/v1
2 kind: Ingress
3 metadata:
4   name: helloworld-ingress
5   namespace: demo
6 spec:
7   rules:
8     - host: helloworld.127.0.0.1.nip.io
9       http:
10         paths:
11           - backend:
12             service:
13               name: helloworld
14               port:
15                 number: 80
16             path: /
17             pathType: Prefix
```



AUSSICHT

- Für lokale Entwicklung interessant:
 - Lokale CI/CD Pipeline:
skaffold <https://skaffold.dev/>
 - Deployment auf Kubernetes Cluster
 - Generatoren für die K8s-YAMLS
 - Kustomize <https://kustomize.io/>
 - Helm <https://helm.sh/>
 - Tool für GitOps
FluxCD <https://fluxcd.io/>



AUFRÄUMEN

- Mit `01_setup_local-k8s/local-k8s.sh stop` kann der Cluster wieder heruntergefahren werden.
- Wenn ihr das später nochmal wiederholen wollt, müsst ihr die Skripte etwas verändern. Anleitung ist unter `01_setup_local-k8s/readme.md`

FRAGEN? FRAGEN!

JETZT ODER SPÄTER AM STAND



K8S IN A NUTSHELL

SOFTWARE & SOLUTION DEVELOPER
INNOVATION PARTNER
COFFEE LOVER

Geschäftsführer:
Peter F. Fellinger, Markus Hartinger

Friedenheimer Brücke 20 | 80639 München | +49.89.45 23 47 - 0
Friedrichstraße 45 | 70174 Stuttgart | +49.711.21 95 28 - 0
Klostergasse 3 | 04109 Leipzig | +49.341.22 178 - 0
Zeil 109 | 60313 Frankfurt am Main | +49.89.45 23 47 - 0
6/4 Abelyan Street | Yerevan 0038 | Armenia

office@jambit.com
www.jambit.com

Dieses Dokument ist vertraulich. Eine Weitergabe an Dritte ist nur mit Zustimmung von jambit möglich.