

# Continuous Lifecycle, 15.11.2023

## Kubernetes, das unbekannte Wesen Schnelleinstieg für Entwicklerinnen

**Sandra Parsick**

@sparsick@mastodon.social

@SandraParsick

mail@sandra-parsick.de

# Wer bin ich?

- Sandra Parsick
- Freiberuflicher Softwareentwickler und Consultant im Java-Umfeld
- Schwerpunkte:
  - Java Enterprise Anwendungen
  - Agile Methoden
  - Software Craftmanship
  - Automatisierung von Entwicklungsprozessen
- Trainings
- Workshops

✉️ mail@sandra-parsick.de

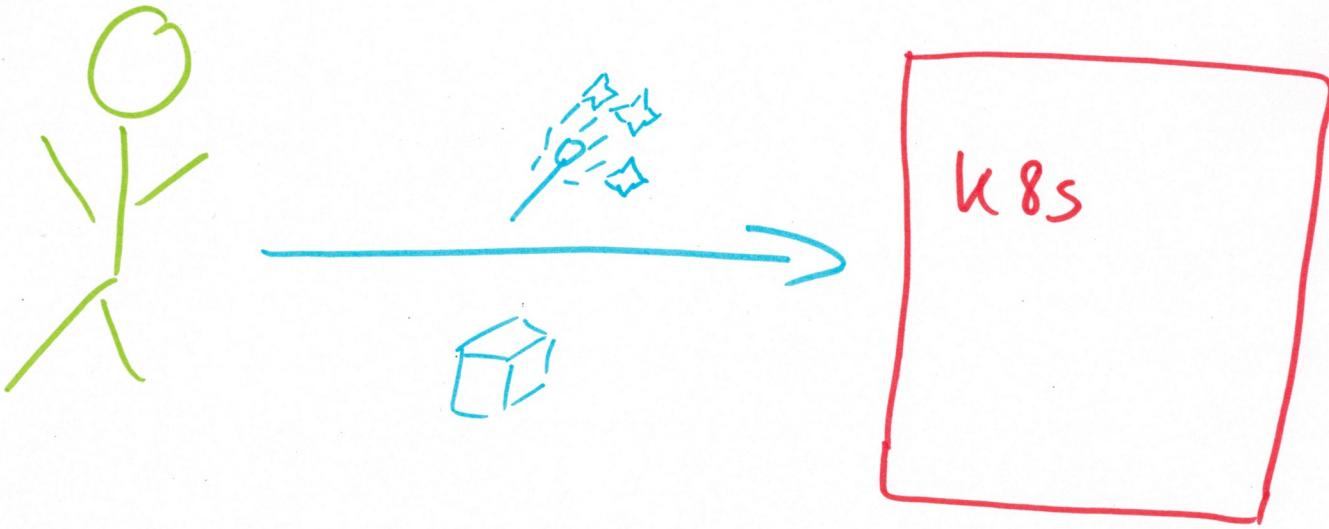
🐦 @SandraParsick

Ⓜ️ @sparsick@mastodon.social

RSS https://www.sandra-parsick.de

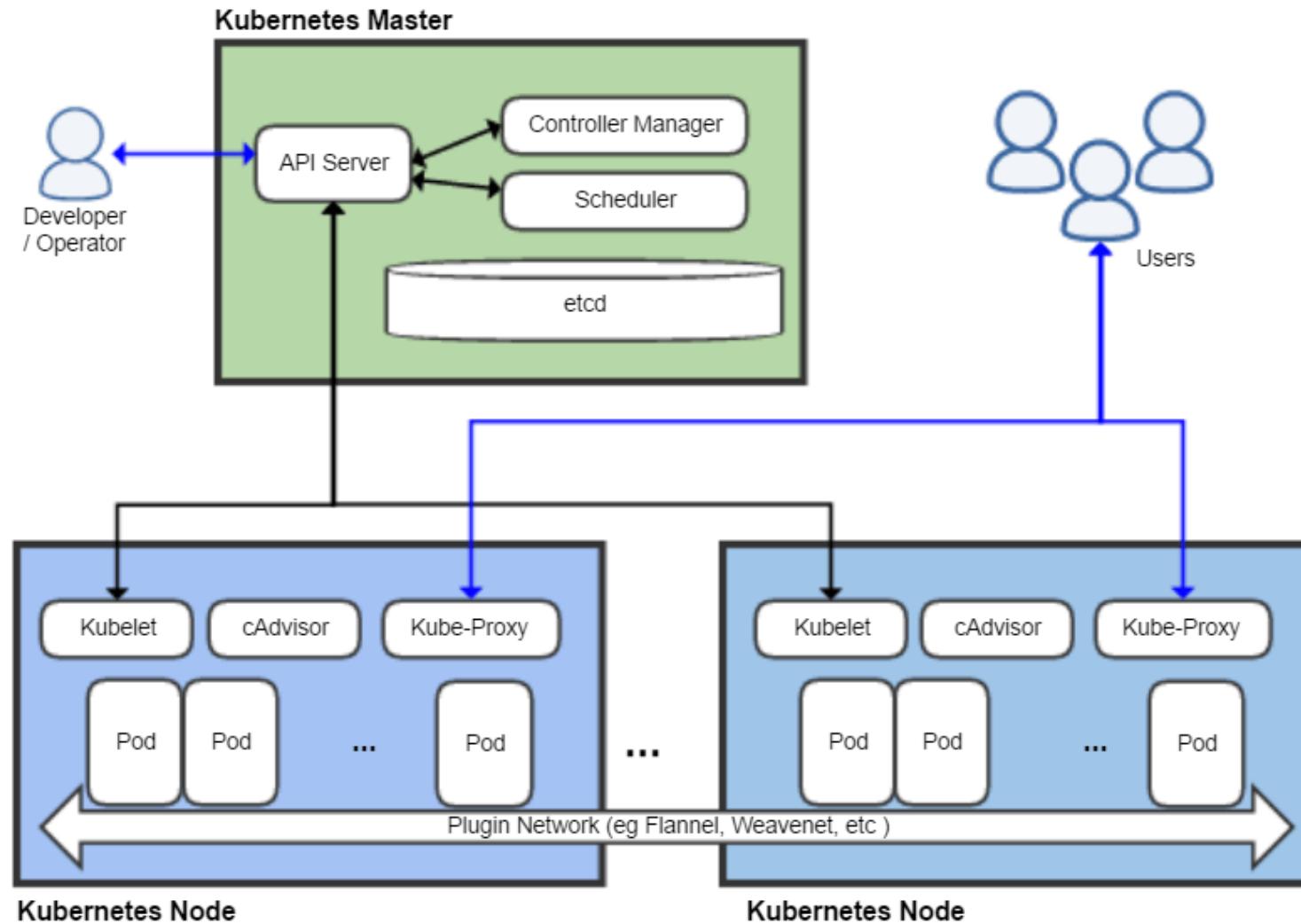
🎧 https://ready-for-review.dev

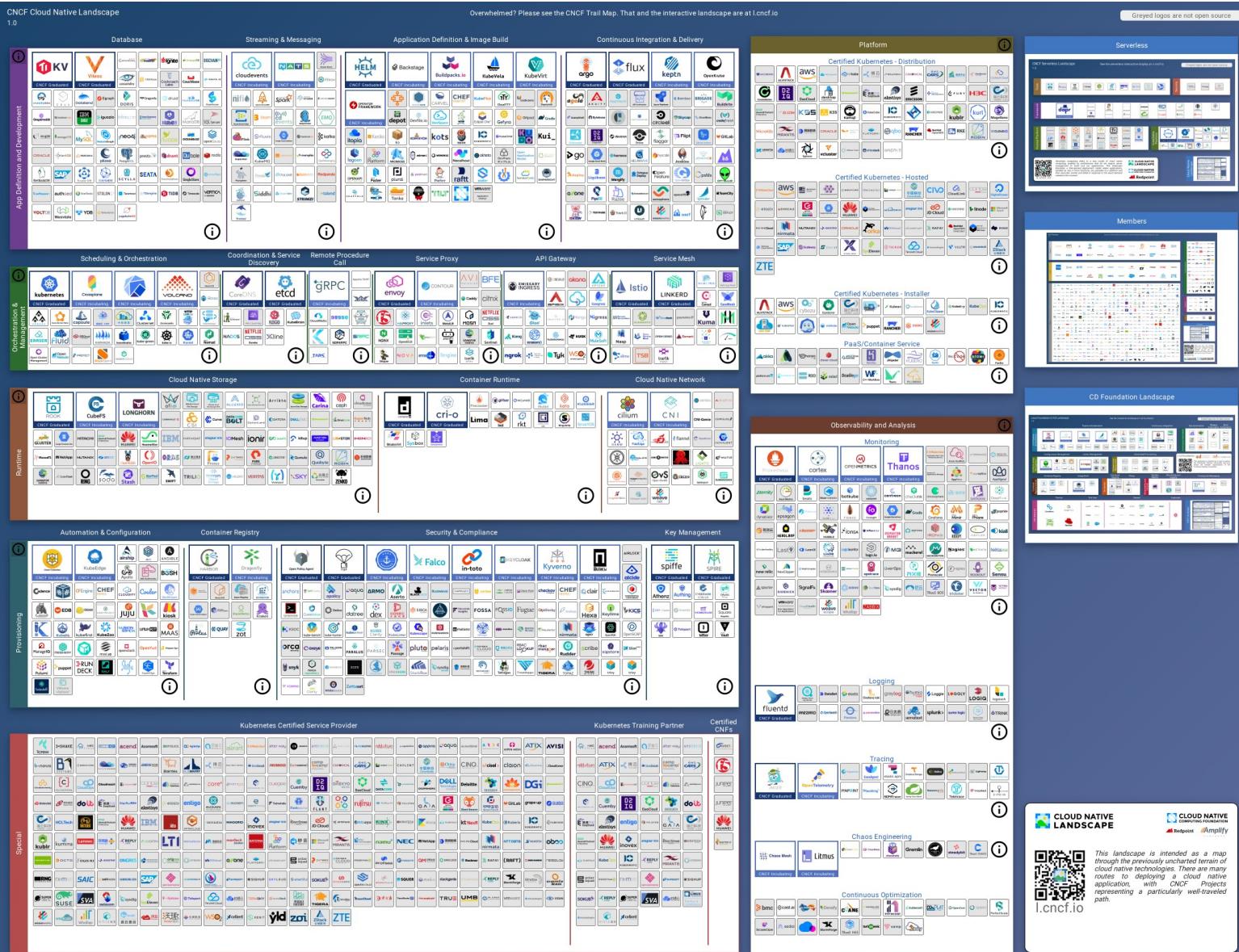


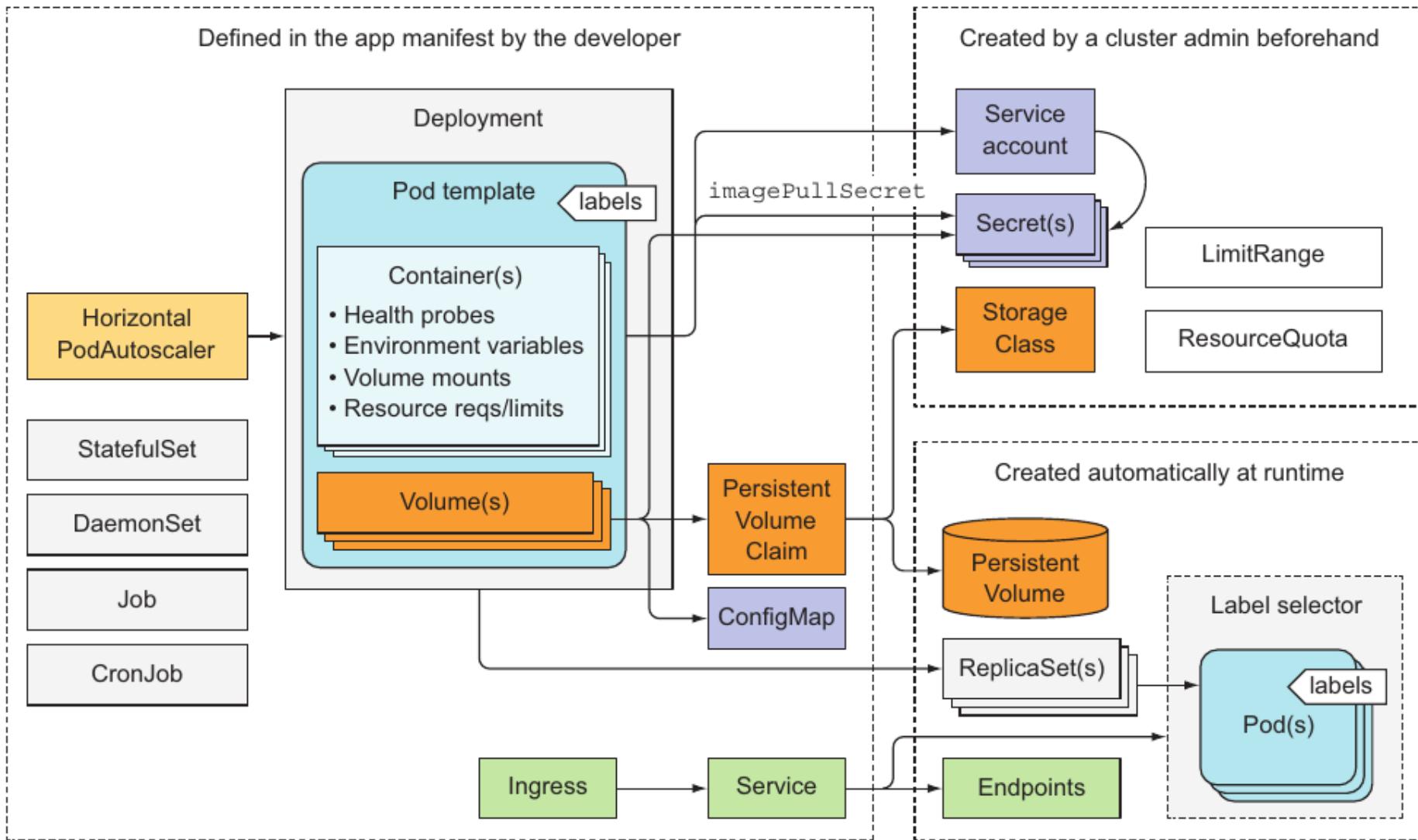


Was ist eigentlich dieses Kubernetes?

**Kubernetes** is an open-source container  
orchestration system for automating software  
deployment, scaling, and management  
(Wikipedia)

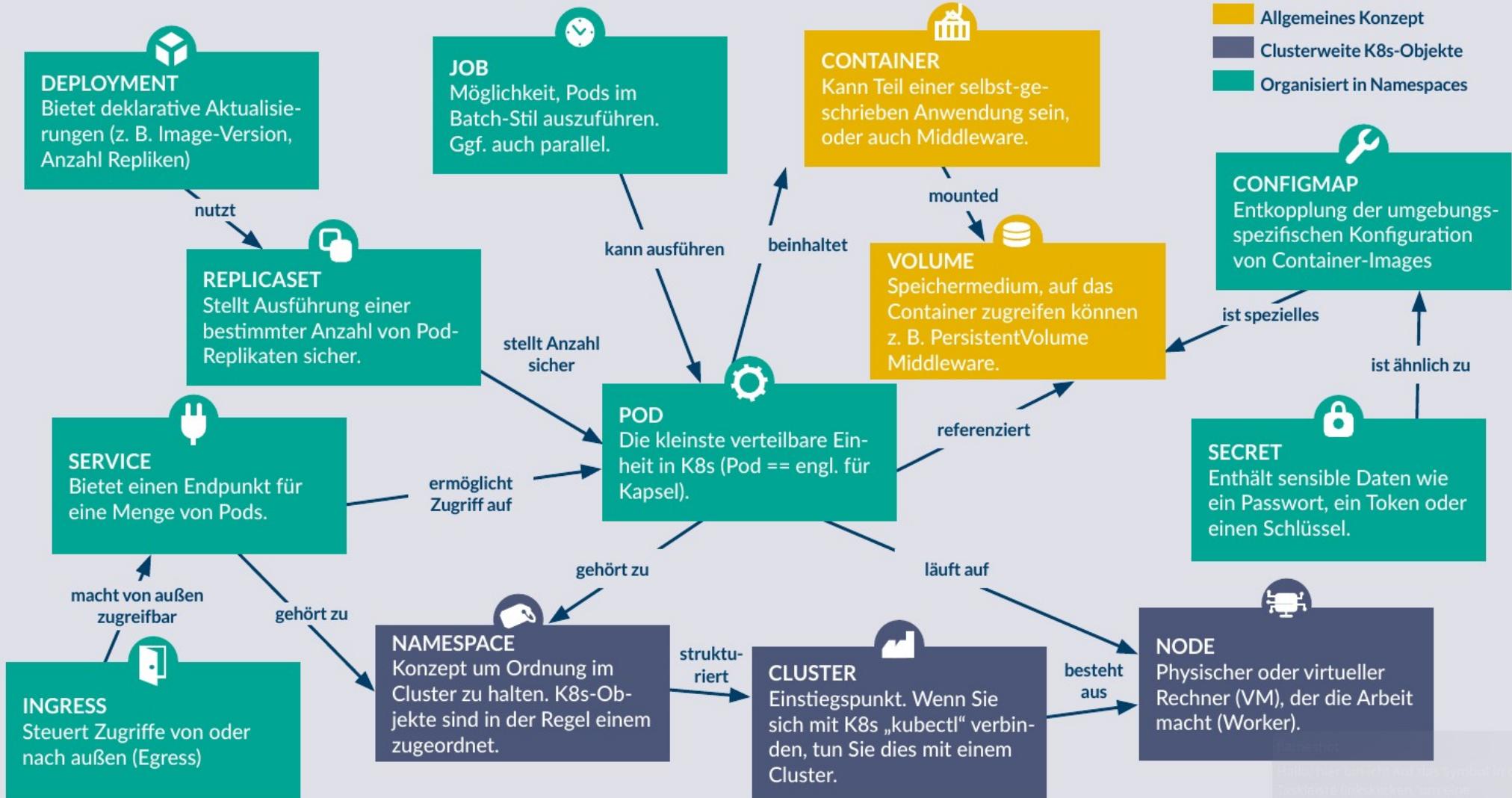






# Ein Begriffsbild für Kubernetes

Das folgende Bild zeigt aus Sicht der Anwendungsentwicklung zentrale Begriffe von k8s im Zusammenspiel.

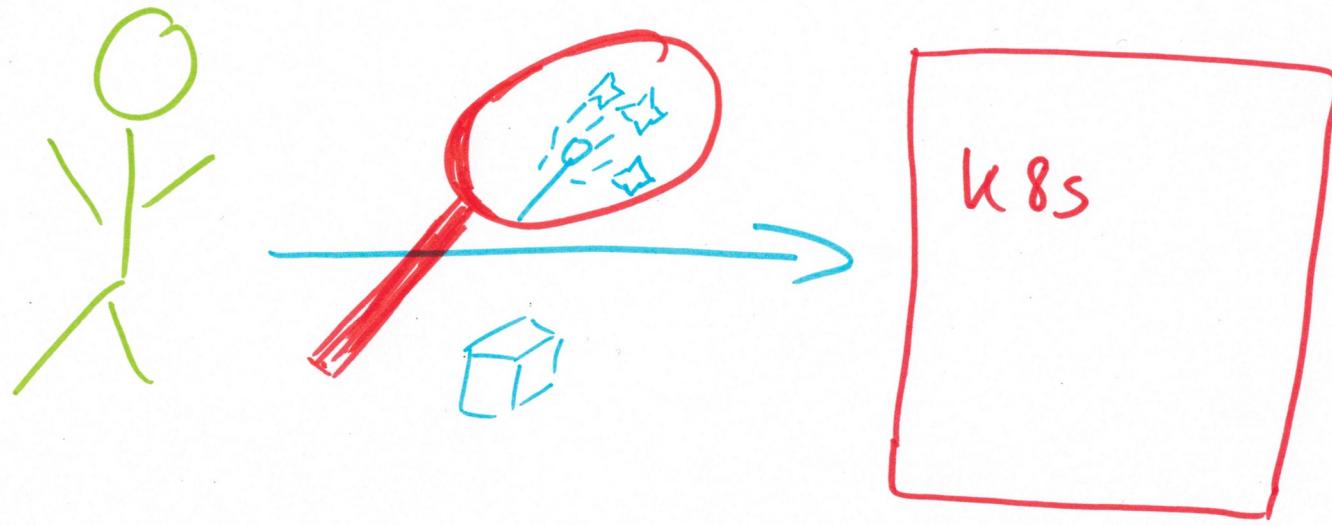


## Legende

- Allgemeines Konzept
- Clusterweite K8s-Objekte
- Organisiert in Namespaces



Hinweis: Klicken Sie auf das Symbol in der rechten oberen Ecke, um die Legende zu vergrößern.

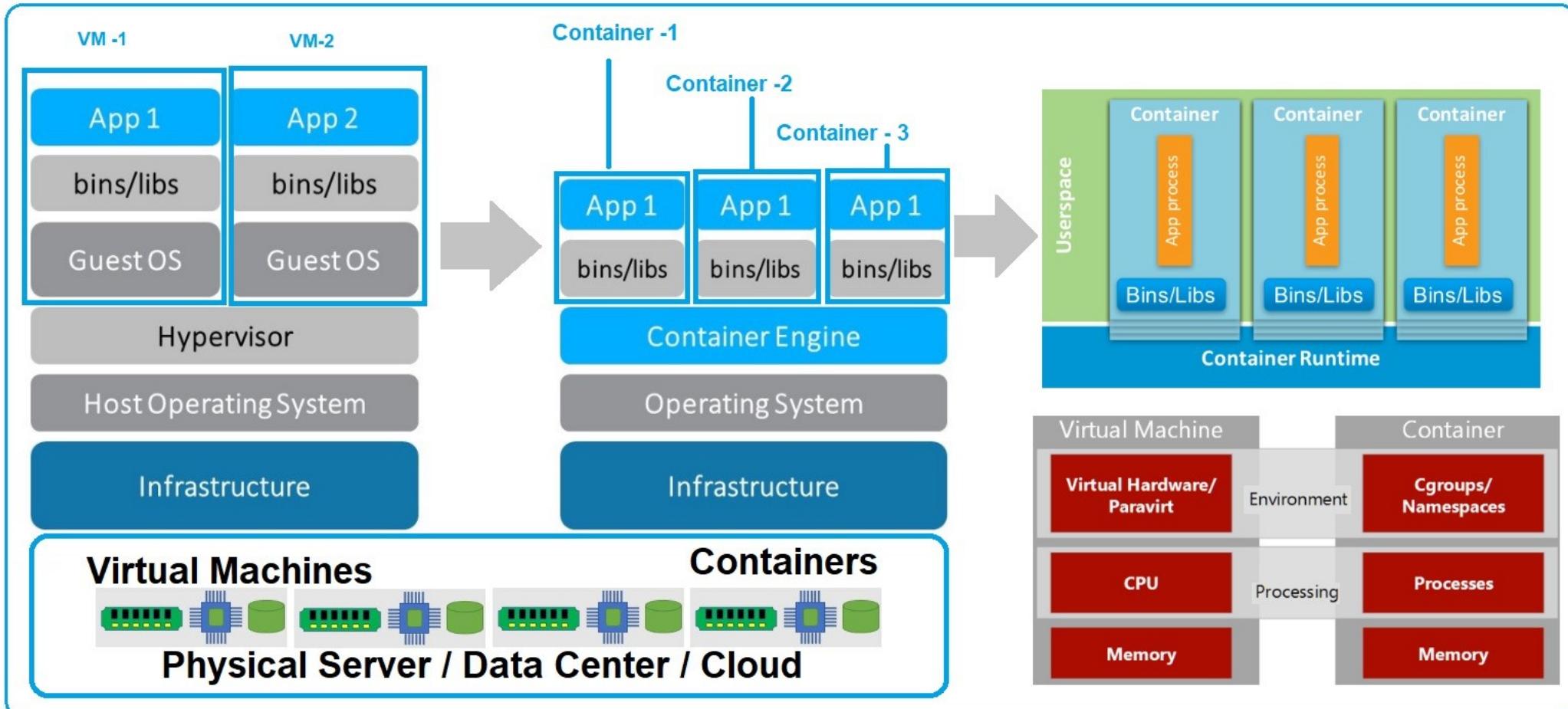


Beispiel: Eine Webapplikation auf Java-Basis  
(Spring Boot)

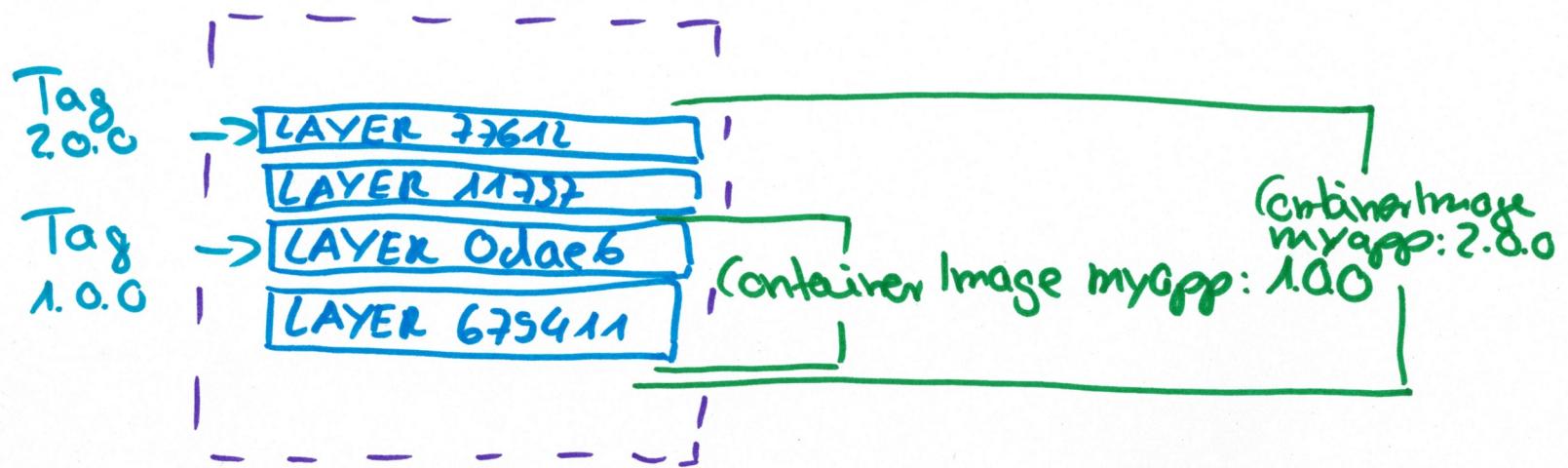
# Container



# Exkurs: Container vs VMs



# Repository myapp



# Basis: Container



```
FROM docker.io/eclipse-temurin:17.0.1_12-jre as builder
WORKDIR /application
COPY maven/*.jar application.jar
RUN java -Djarmode=layer-tools -jar application.jar extract

FROM gcr.io/distroless/java17-debian11
WORKDIR /application
EXPOSE 8080
COPY --from=builder /application/dependencies/ ./
COPY --from=builder /application/spring-boot-loader/ ./
COPY --from=builder /application/snapshot-dependencies/ ./
COPY --from=builder /application/application/ ./
ENTRYPOINT ["java", "org.springframework.boot.loader.JarLauncher"]
```

# Demo

# Container Builder

- Docker
- Buildpacks
- JIB
- Buildah
- Podman
- Weitere Infos im Artikel „Container-Images Deep Dive“ auf Informatik Aktuell

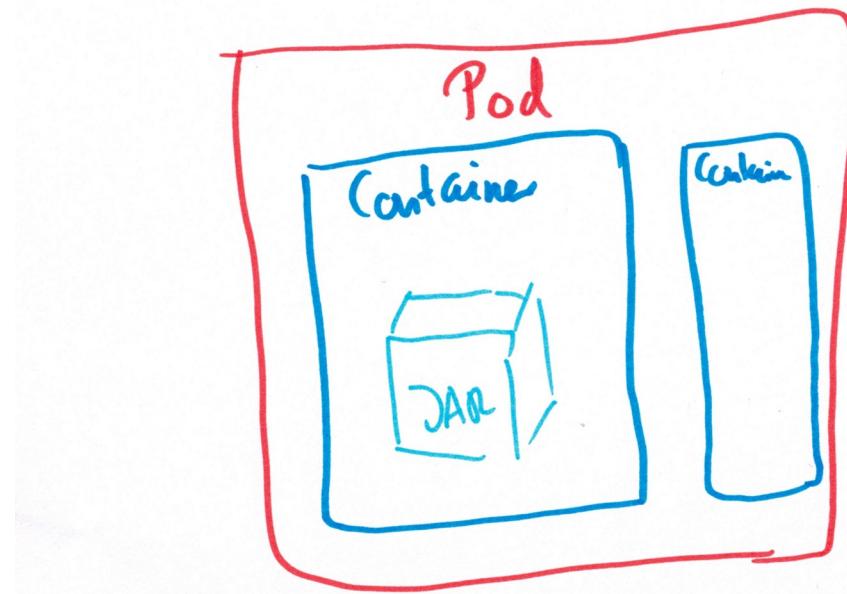
# Exkurs: Lokales Kubernetes

# Demo

# Alternativen zu Minikube

- k3s
- k3d
- kind
- microk8s
- k0s

Pod





```
apiVersion: v1
kind: Pod
metadata:
  name: spring-boot-demo
spec:
  containers:
  - name: hero-app
    image: sparsick/spring-boot-demo:1.5.0
    ports:
    - containerPort: 8080
```



```
kubectl apply -f pod.yaml
```

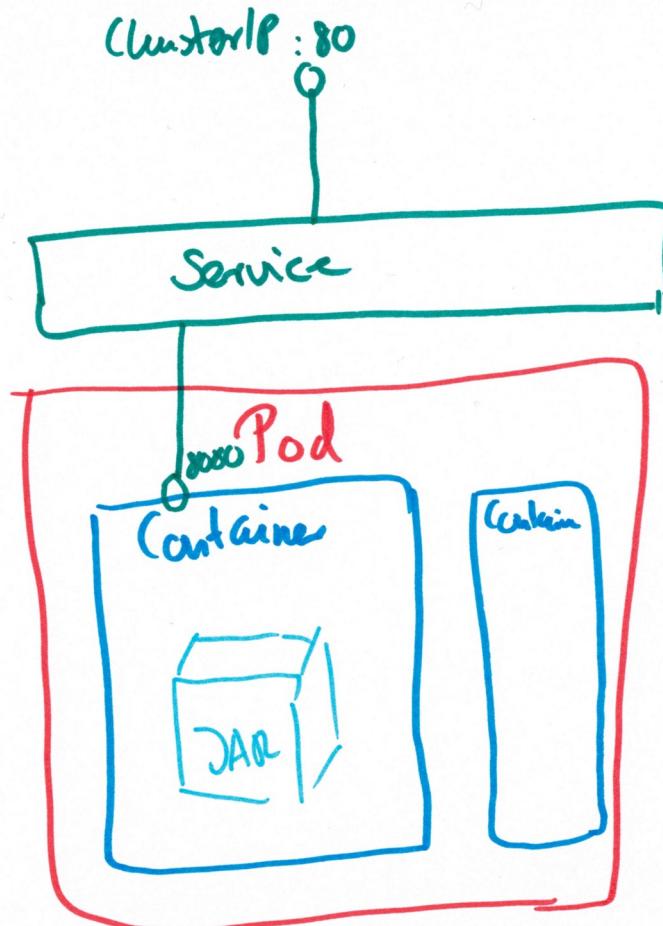
# Demo



```
apiVersion: v1
kind: Pod
metadata:
  name: spring-boot-demo
spec:
  containers:
  - name: hero-app
    image: sparsick/spring-boot-demo:1.5.0
    ports:
    - name: container-http
      containerPort: 8080
      protocol: TCP
    livenessProbe:
      httpGet:
        path: /actuator/health/liveness
        port: container-http
        initialDelaySeconds: 15
        periodSeconds: 10
        timeoutSeconds: 30
    readinessProbe:
      httpGet:
        path: /actuator/health/readiness
        port: container-http
        initialDelaySeconds: 15
        periodSeconds: 10
        timeoutSeconds: 30
```

# Demo

# Service

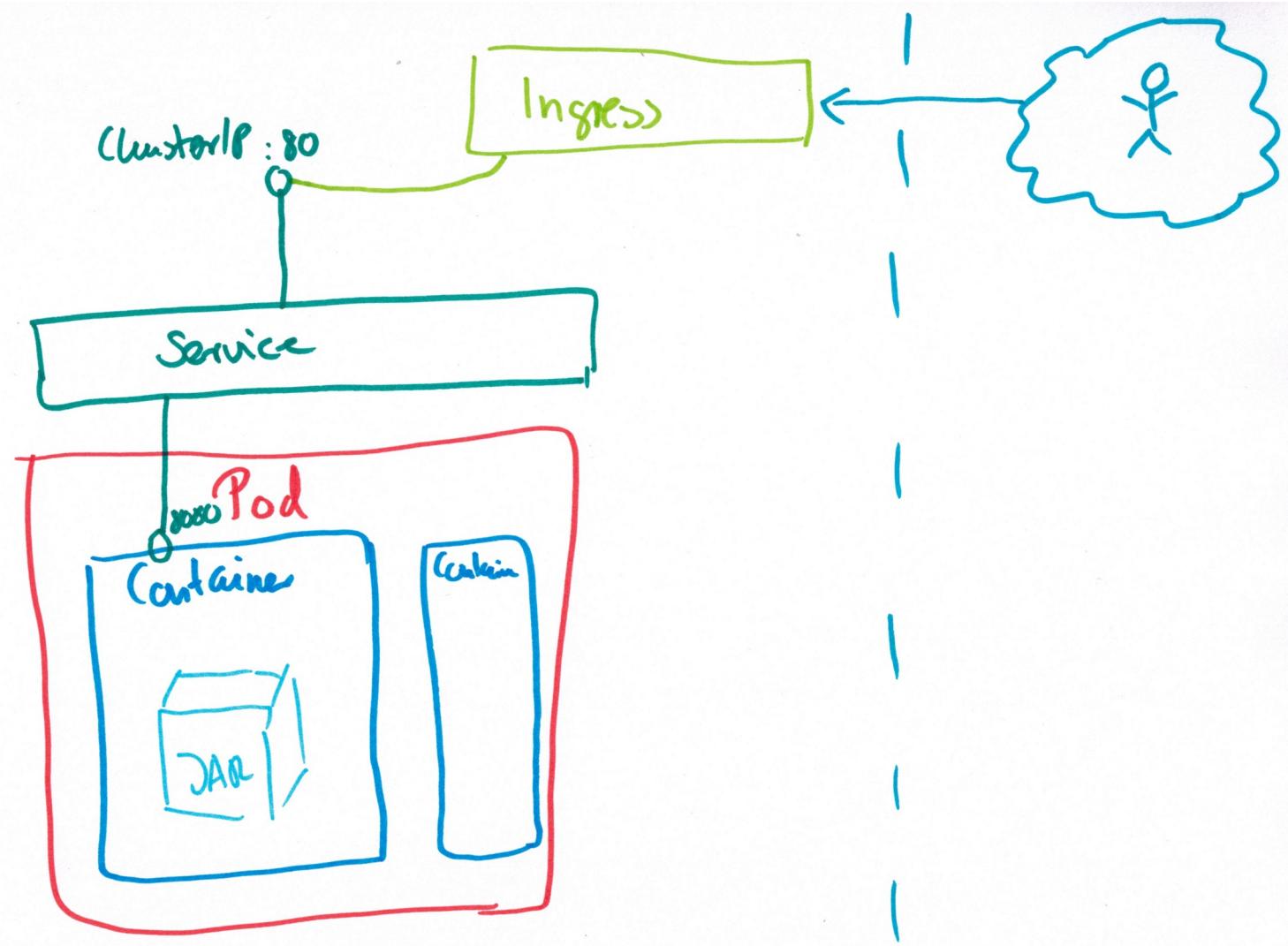


```
apiVersion: v1
kind: Pod
metadata:
  name: spring-boot-demo
  labels:
    app.kubernetes.io/name: hero
spec:
  containers:
  - name: hero-app
    image: sparsick/spring-boot-demo:1.5.0
    ports:
    - name: container-http
      containerPort: 8080
      protocol: TCP

---
apiVersion: v1
kind: Service
metadata:
  name: spring-boot-demo-srv
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: hero
  ports:
  - name: hero-http
    protocol: TCP
    port: 80
    targetPort: 8080
```

# Demo

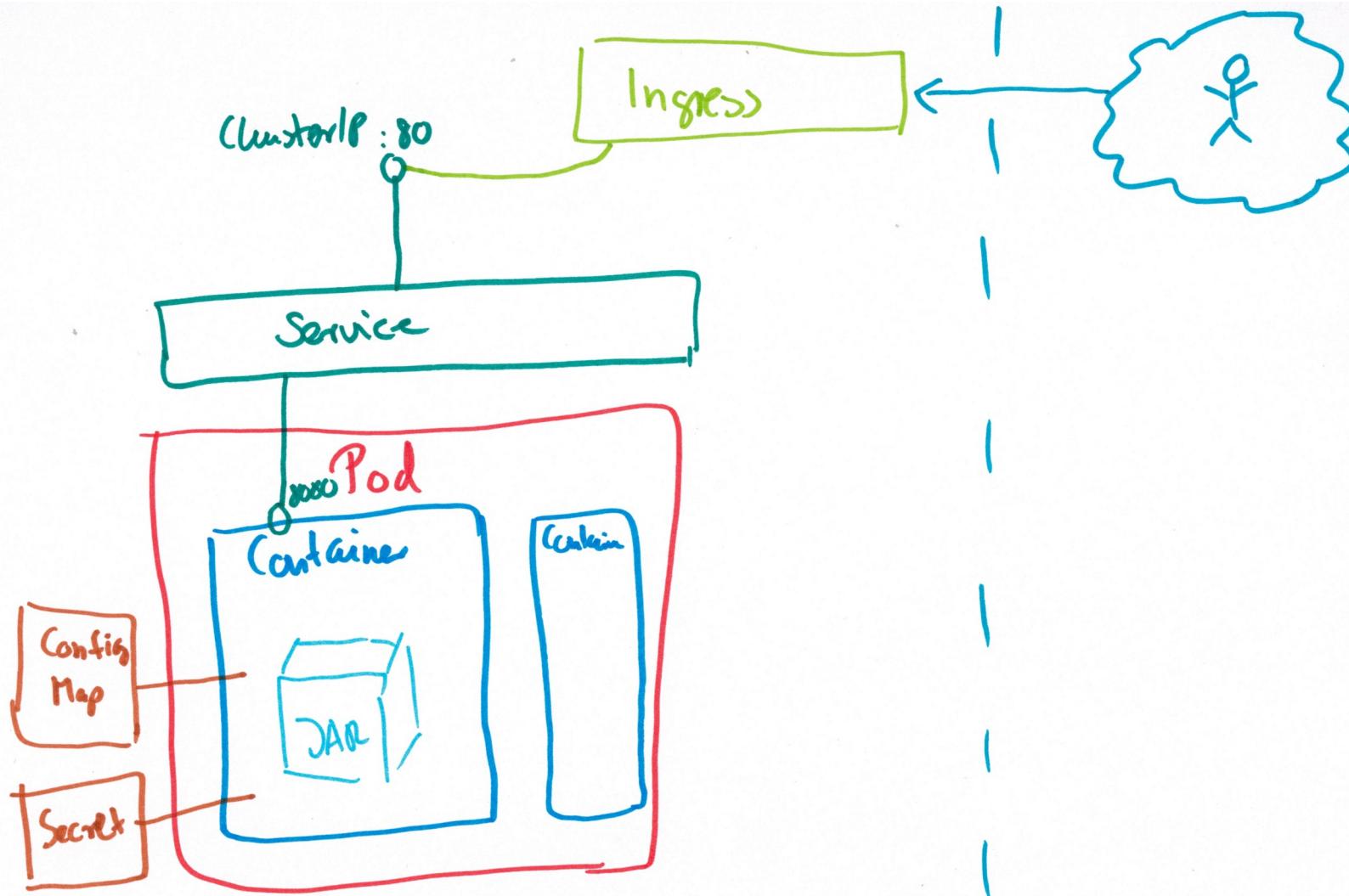
# Ingress



```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: spring-demo-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/x-forwarded-prefix: "/"
spec:
  rules:
    - host: spring-boot-demo.local
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: spring-boot-demo-srv
                port:
                  number: 80
```

# Demo

# ConfigMap / Secret





```
apiVersion: v1
kind: ConfigMap
metadata:
  name: spring-boot-demo-config
data:
  MONGODB_ENABLED: "false"

---
apiVersion: v1
kind: Secret
metadata:
  name: secret-sample
stringData:
  password: geheim
```

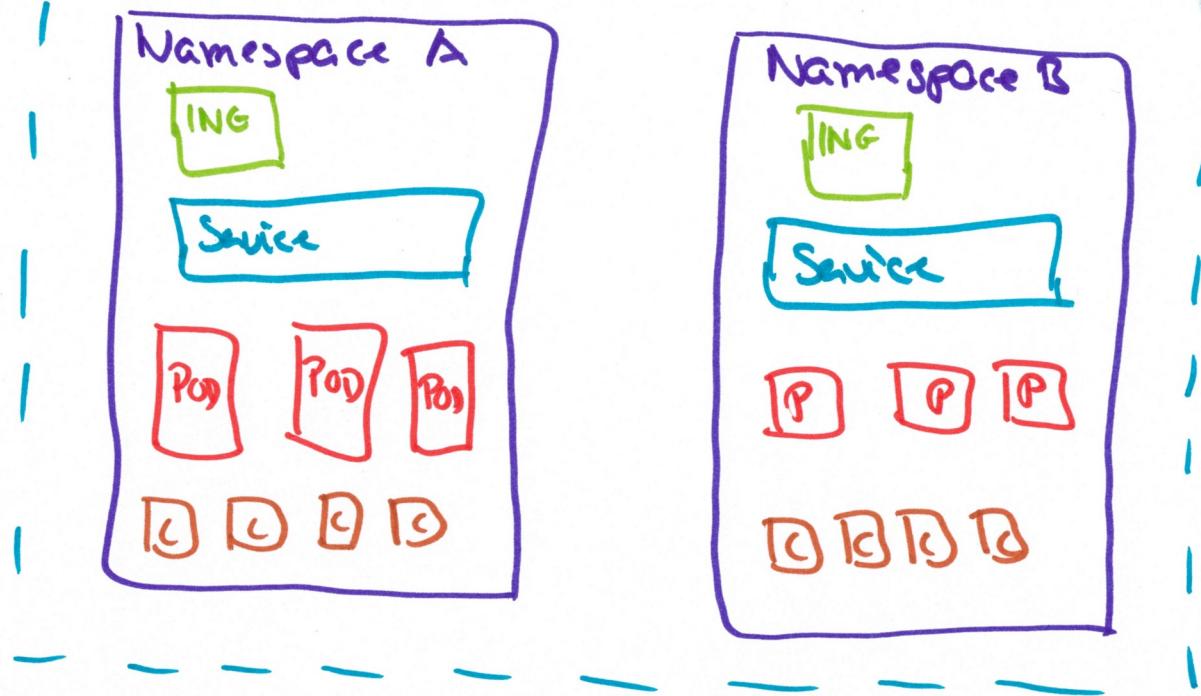


```
apiVersion: v1
kind: Pod
metadata:
  name: spring-boot-demo
spec:
  containers:
    - name: hero-app
      image: sparsick/spring-boot-demo:1.5.0
      ports:
        - name: container-http
          containerPort: 8080
          protocol: TCP
      envFrom:
        - configMapRef:
            name: spring-boot-demo-config
```

# Demo

# Namespace

# CLUSTER



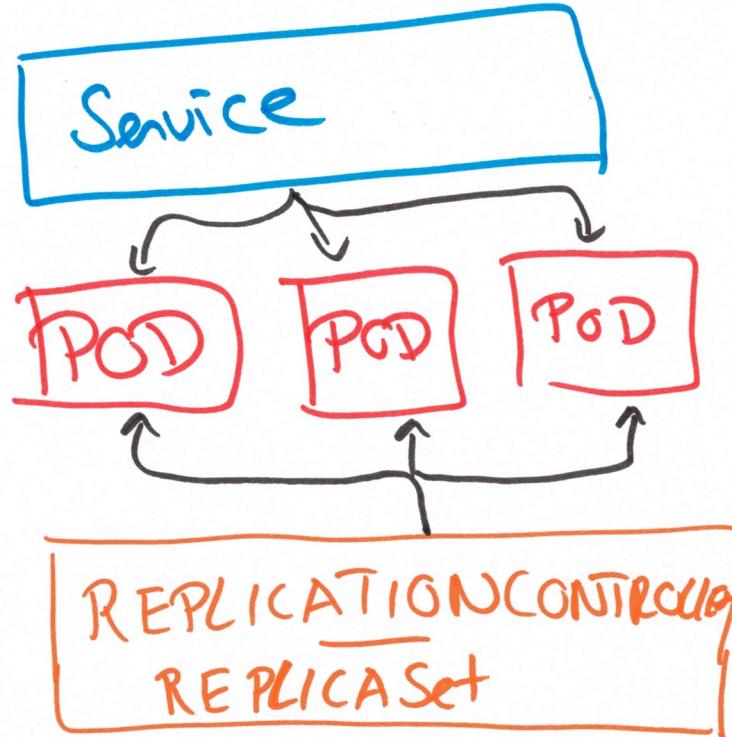


```
apiVersion: v1
kind: Namespace
metadata:
  name: spring-boot-demo-namespace

---
apiVersion: v1
kind: Pod
metadata:
  name: spring-boot-demo
  namespace: spring-boot-demo-namespace
spec:
  containers:
    - name: hero-app
      image: sparsick/spring-boot-demo:1.5.0
      ports:
        - containerPort: 8080
```

# Demo

# Deployment



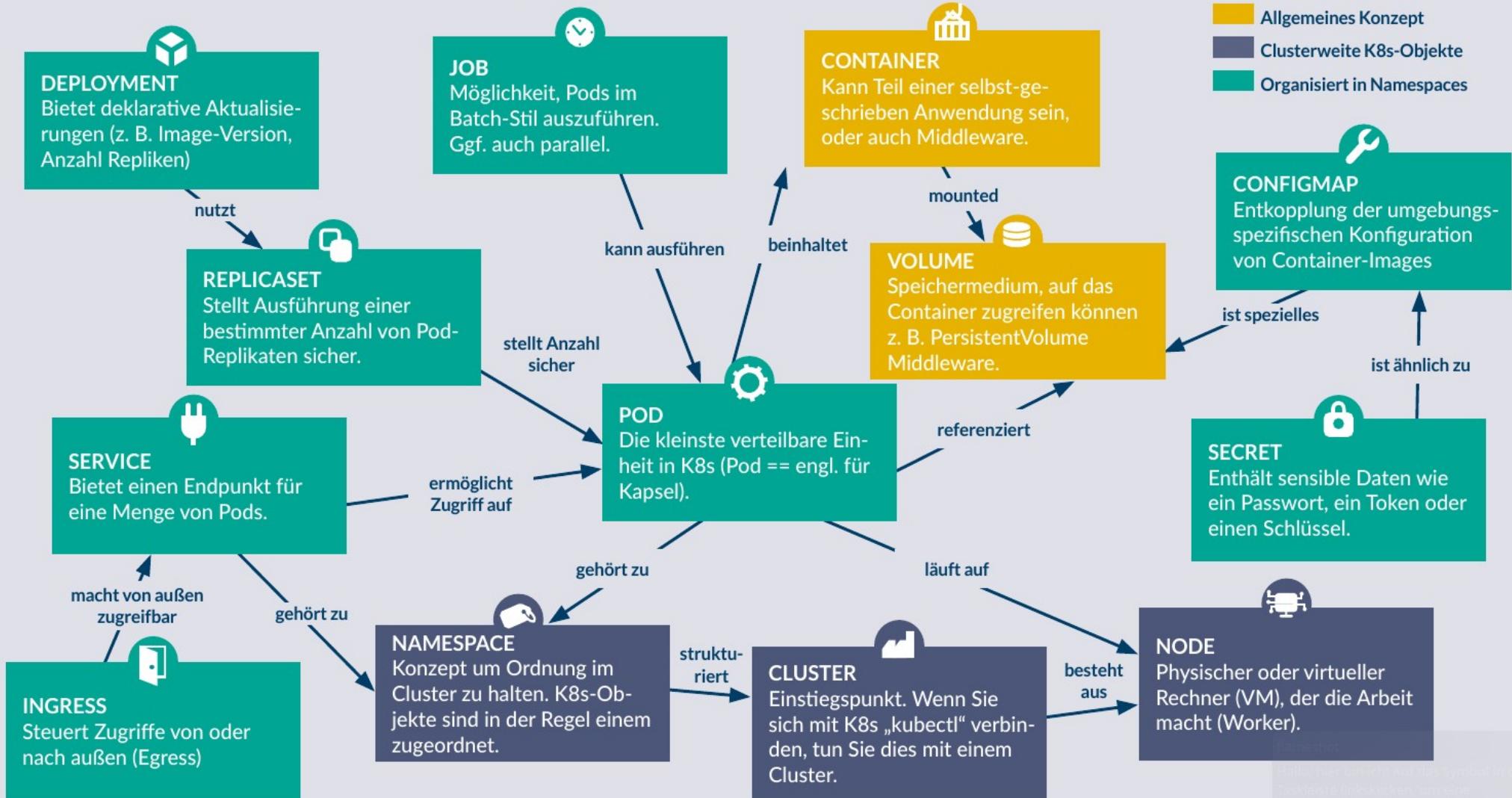


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-boot-demo-deploy
  namespace: spring-boot-demo-namespace
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hero-app
  template:
    metadata:
      labels:
        app: hero-app
  spec:
    containers:
    - name: hero-app
      image: sparsick/spring-boot-demo:1.5.0
      ports:
      - name: container-http
        containerPort: 8080
        protocol: TCP
      envFrom:
      - configMapRef:
          name: spring-boot-demo-config
```

# Demo

# Ein Begriffsbild für Kubernetes

Das folgende Bild zeigt aus Sicht der Anwendungsentwicklung zentrale Begriffe von k8s im Zusammenspiel.



## Legende

- Allgemeines Konzept
- Clusterweite K8s-Objekte
- Organisiert in Namespaces



Hinweis: Klicken Sie auf das Symbol in der rechten oberen Ecke, um die Legende zu vergrößern.

App K8s-ready machen

Wie sehe ich was im Cluster los ist?

Backend / Frontend

Versionierung

Container Images

Was gehört alles ins Git Repository rein?

Debugging

CI

Deployment Scripte

Konfiguration

Lokale Entwicklungsumgebung



**architektur SPICKER**  
Übersichten für die konzeptionelle Seite der Softwareentwicklung

**embarc** Software Consulting GmbH  
**SIGS DATACOM** PRAXISINFORMATIONEN FÜR IT-PROFESSIONALS

# Container-Anwendungen entwickeln

Der Einsatz von Containern verspricht die immer größer werdende Komplexität der Anwendungslandschaft besser zu beherrschen. Dieser Spicker erklärt, wie Sie und Ihr Team Applikationen in Containern bauen und was Sie beachten müssen, um diese zu betreiben.

## Worum geht's?

- Sie wollen Anwendungen in Containern betreiben. Welche Vorteile bringen diese bei welchen Architekturen mit?
- Alte Anwendungen verharren noch in monolithischen Deployments oder schwergewichtigen Applikationsservern. Wie kriegen Sie diese schmerzarm in die „schöne neue Welt“?
- Um den richtigen Containermix wird schon lange viel diskutiert. Wie setzen Sie Container heute richtig ein?
- Der Markt stellt soviele Orchestrationslösungen bereit. Welche passt auf Ihre Situation am besten?

### Anwendungsentwurf, Container und Orchestrierung

**Die Zusammenhänge, Fragen und Antworten**

1. Worum geht es bei Anwendungen?  
Anwendungen in Containern sind ohne Hilfsmittel schwierig zu betreiben. Wie hängen die zentralen Begriffe zusammen? Was motiviert den Einsatz von Docker, Kubernetes und Co?

2. Was haben Microservices damit zu tun?  
Die Zerlegung einer komplexen Anwendung macht sie wertbar und bewirtschaftbar.

3. Warum sollen wir Anwendungen in verschiedenen Prozessen laufen lassen?  
Microservices sind ein Architekturstil, also eine grundlegende Art Anwendungen zu bauen. So lassen sich Anwendungsteile gezielter skalieren.

4. Und was sind Container?  
Die Zerlegung kann zu mehreren Prozessen und einer verteilten Anwendung führen (Alternative: Modularität).

5. Wie helfen Container bei verteilten Anwendungen?  
Jeder Microservice ist ein Prozess und lose gekoppelte Services.

6. Wie kommen Orchestrierungsstrategien ins Spiel?  
Zu Architekturen siehe Seite 2  
Zu Microservices siehe Spicker #3  
Mit Orchestrierungslösungen lassen sich Anwendungen einfacher definieren, strukturiert und konfigurieren.

Mehr zu Containern ab Seite 3  
Docker ist die am Markt verbreitetste Container-Technologie.  
Sie bieten ein einheitliches Deployment-Format (Images) für unterschiedliche Technologien.

Mehr zu Kubernetes ab Seite 4  
Kubernetes ist die am Markt verbreitetste Orchestrierungslösung.  
Container teilen sich die Ressourcen und sind gleichzeitig voneinander isoliert.  
Auf einem Betriebssystem können viele Container parallel laufen.

Fragen?

[mail@sandra-parsick.de](mailto:mail@sandra-parsick.de)

@SandraParsick

@sparsick@mastodon.social

<https://github.com/sparsick/k8s-intro-talk>

# Weitere Vorträge zu diesem Thema

- From 0 to Kubernetes

# Weitere Informationen

- <https://www.informatik-aktuell.de/entwicklung/methode-n/container-images-deep-dive-101-wege-zum-bauen-und-bereitstellen.html>
- „Kubernetes in Action“ von Marko Lukša
- „Docker in Action“ von Jeff Nickoloff, Stephen Kuenzli
- „Container-Anwendungen entwickeln“  
<https://www.architektur-spicker.de/>
- „Continuous Delivery“  
<https://www.architektur-spicker.de/>

# Bildnachweisweise

- <https://dzone.com/articles/docker-containers-and-kubernetes-an-architectural>