

JAX, 06.05.2025

# Kubernetes, das unbekannte Wesen Schnelleinstieg für Entwicklerinnen

Sandra Parsick  
[@sparsick@mastodon.social](mailto:@sparsick@mastodon.social)  
[mail@sandra-parsick.de](mailto: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

Ⓜ️ @sparsick@mastodon.social

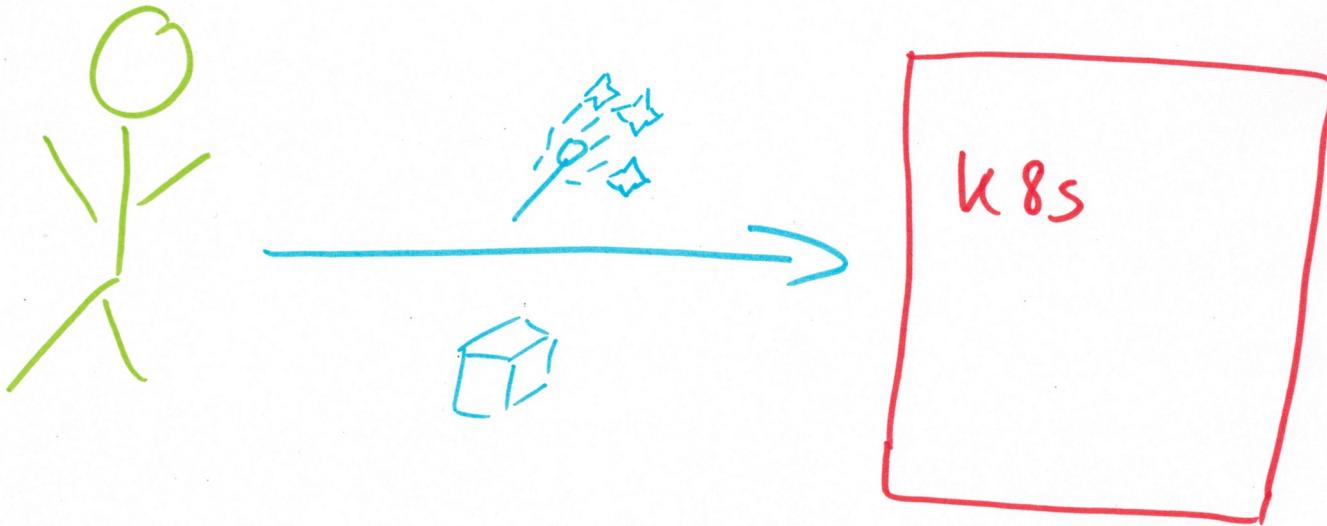
RSS https://www.sandra-parsick.de

🎧 https://ready-for-review.dev



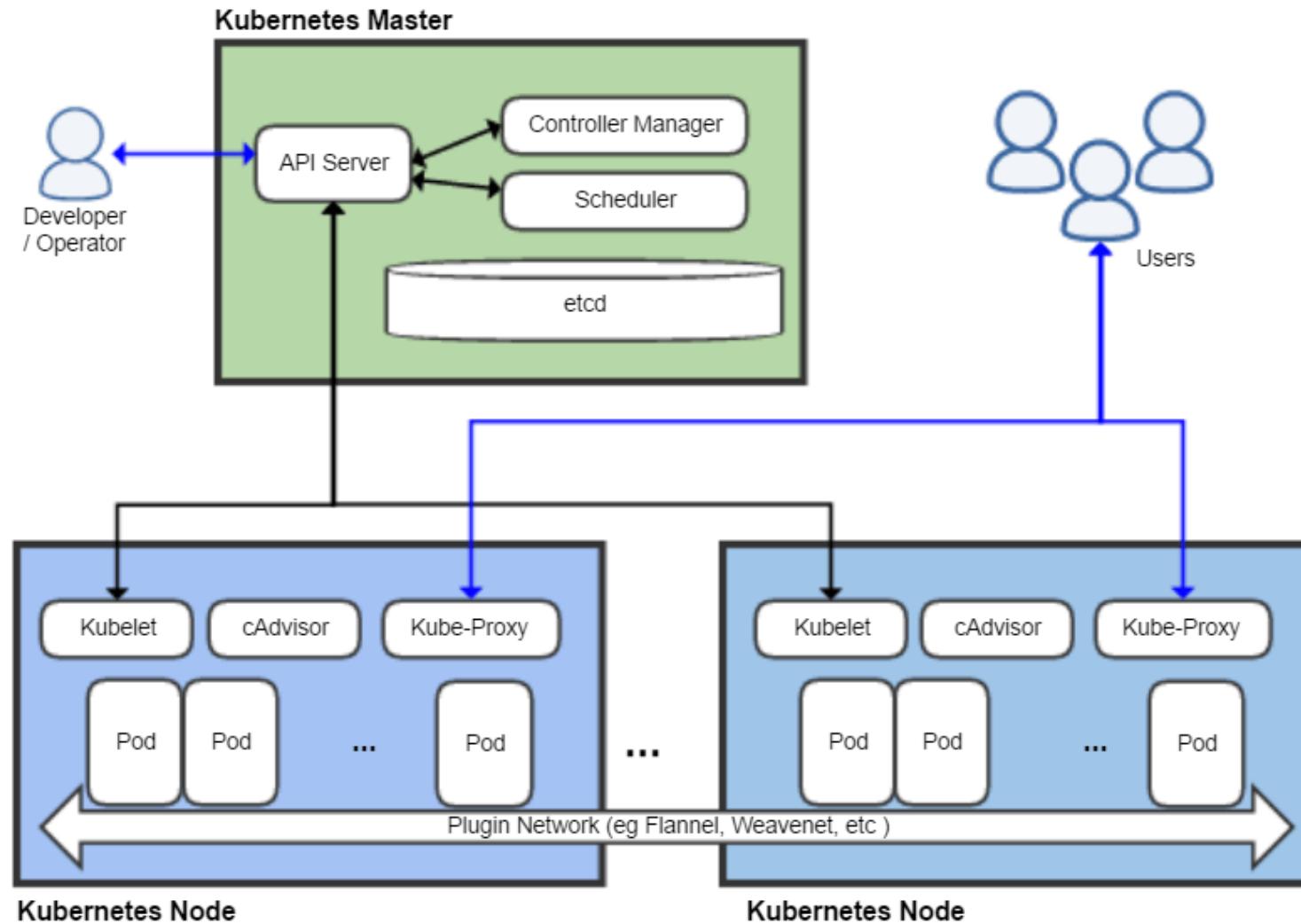
ACE Pro

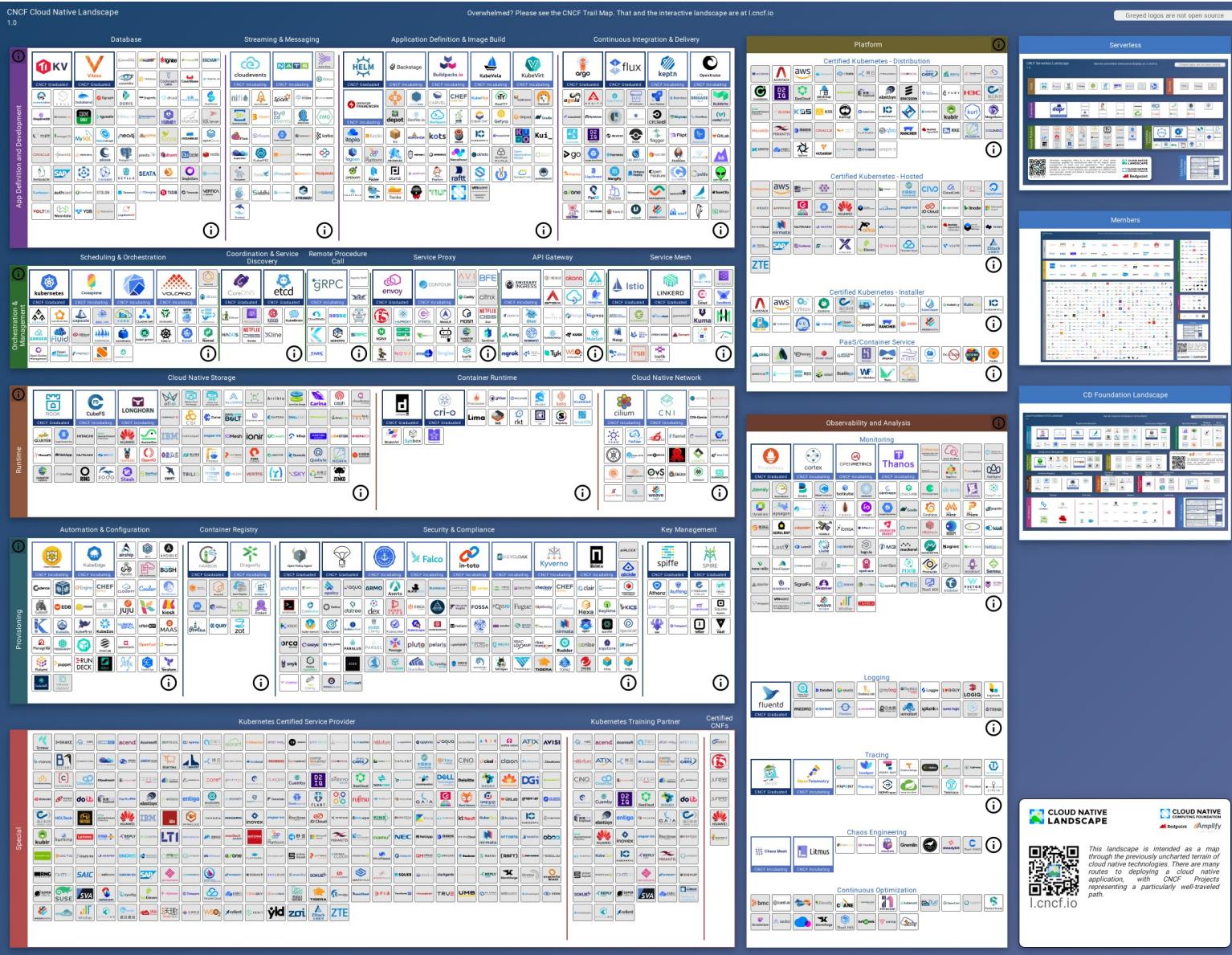


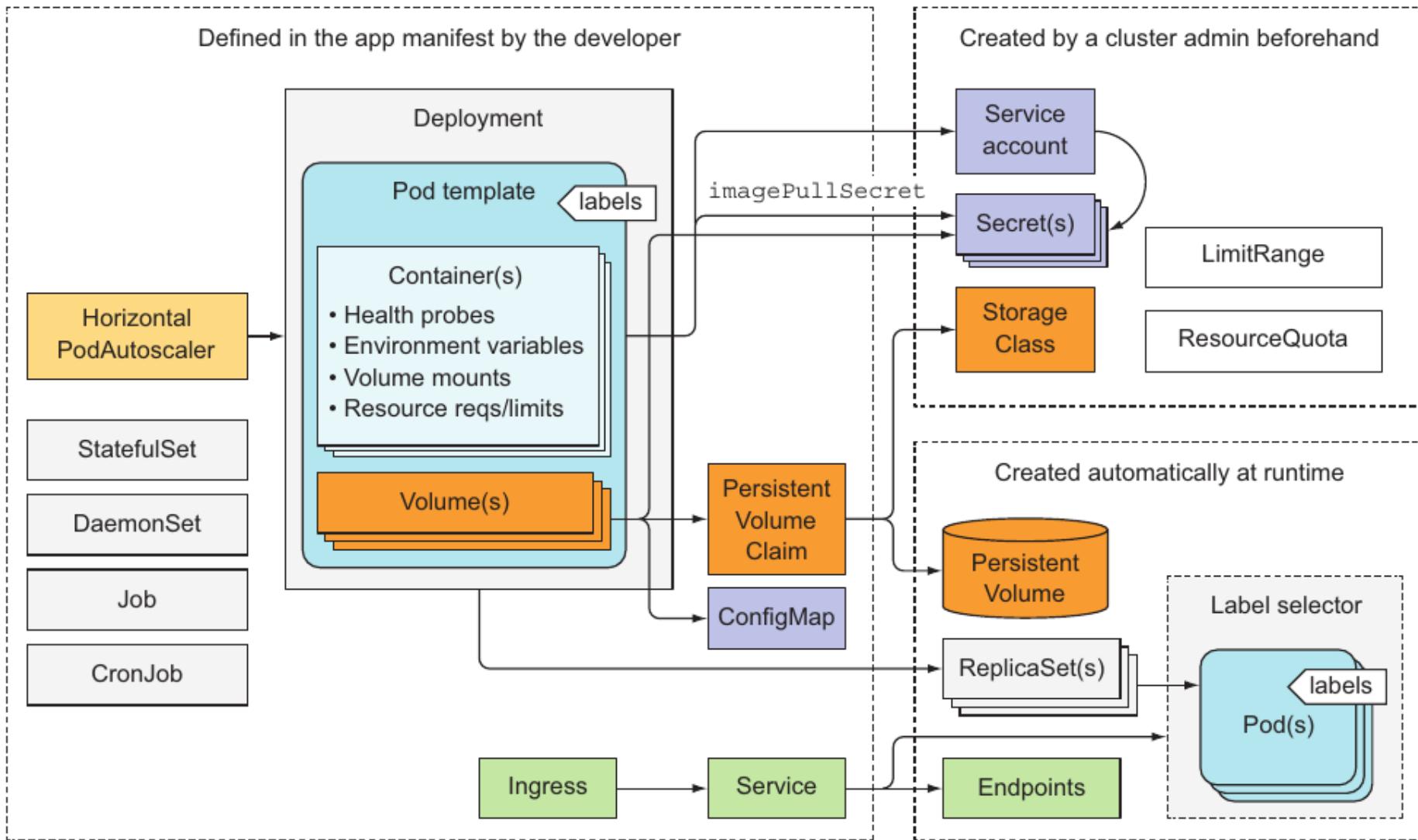


Was ist eigentlich dieses Kubernetes?

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

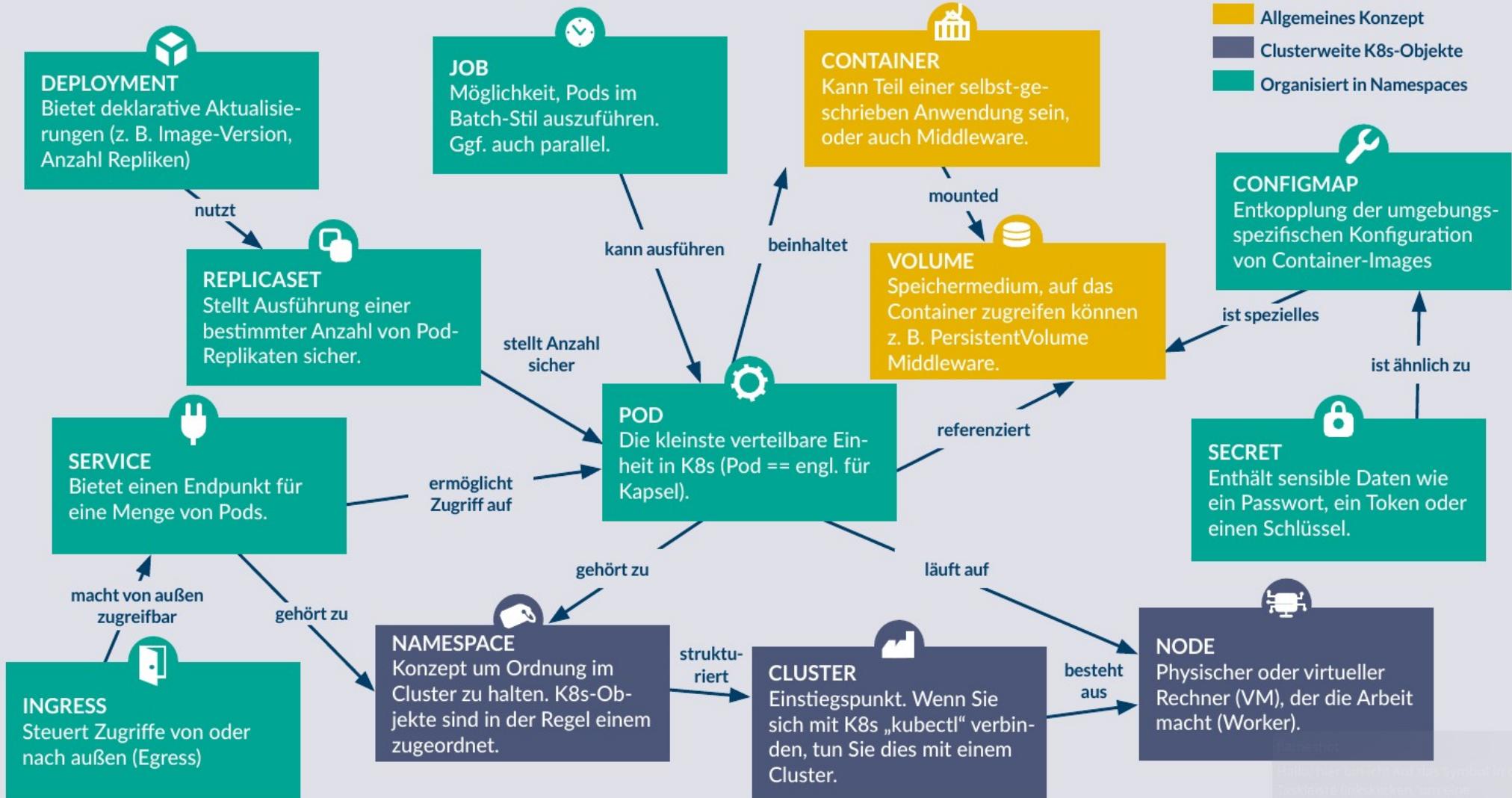






# Ein Begriffsbild für Kubernetes

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

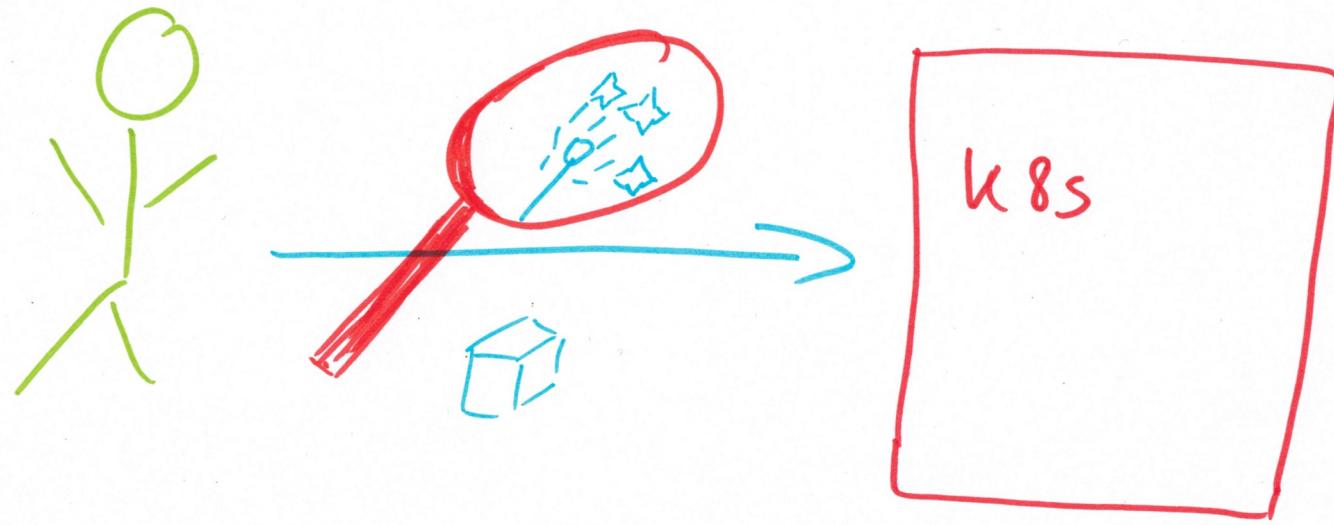


## Legende

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



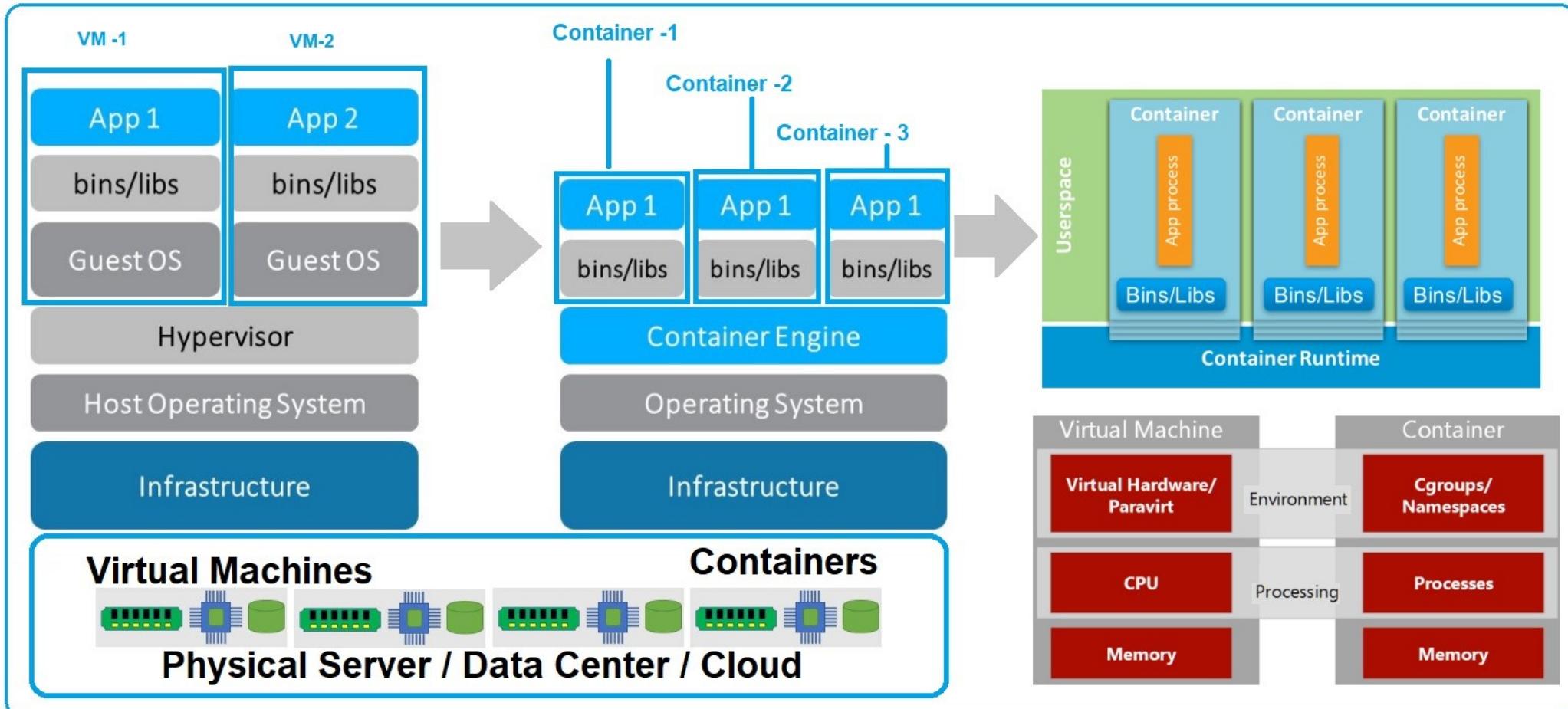
Hinweis: Klicken Sie auf das Symbol in der Legende, um die entsprechenden Begriffe im Diagramm zu markieren.



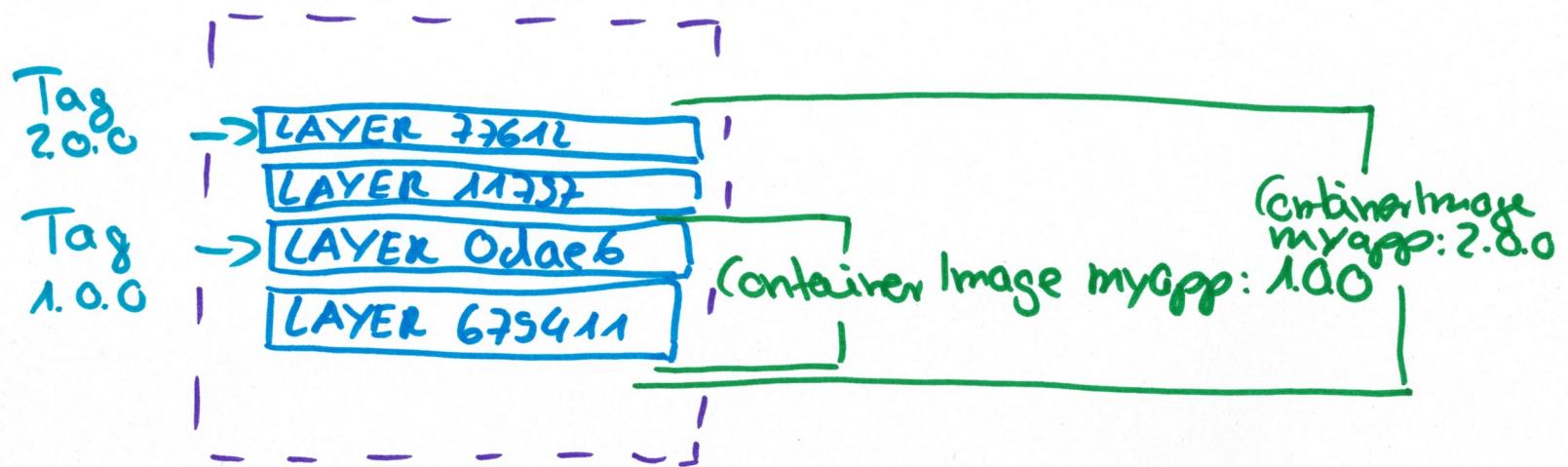
# Container



# Exkurs: Container vs VMs



# Repository myapp



# Basis: Container



```
FROM docker.io/eclipse-temurin:21.0.7_6-jre-noble AS builder
WORKDIR /application
COPY maven/*.jar application.jar
RUN java -Djarmode=layer-tools -jar application.jar extract

FROM gcr.io/distroless/java21-debian12
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"]
```

# Basis: Container



```
docker build . -t sparsick/spring-boot-demo:latest
```

# Demo

# Container Builder

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

# Exkurs: Lokales Kubernetes



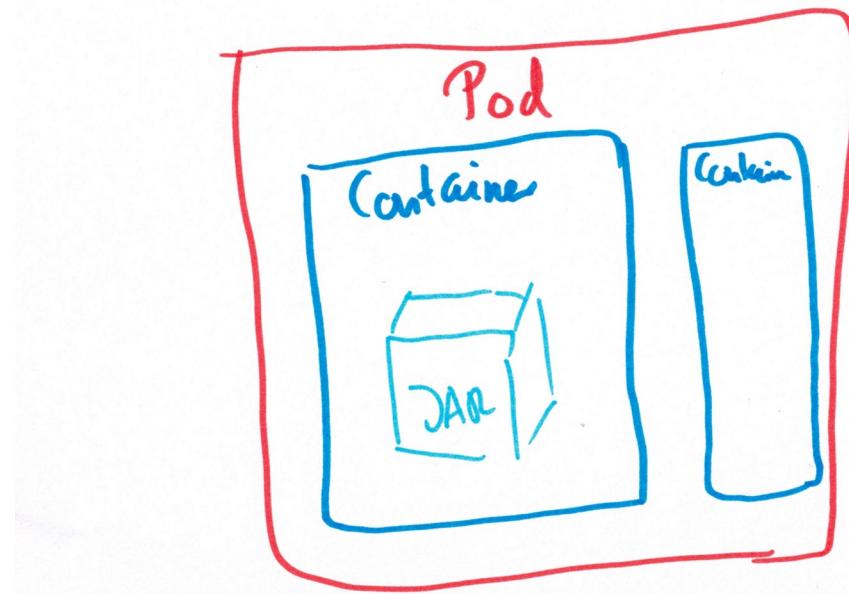
minikube

# 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
```



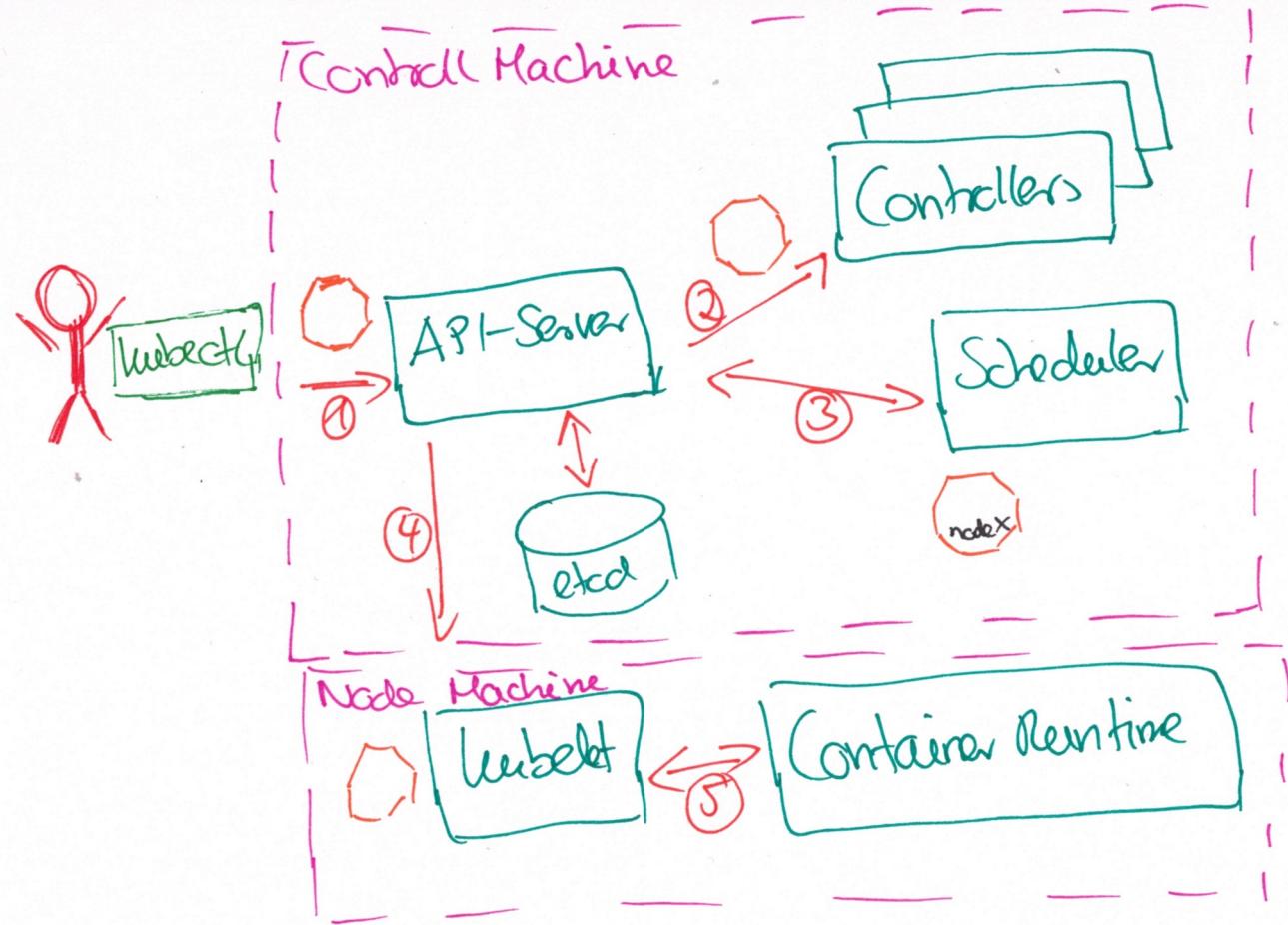
```
kubectl apply -f pod.yaml
```

# Demo

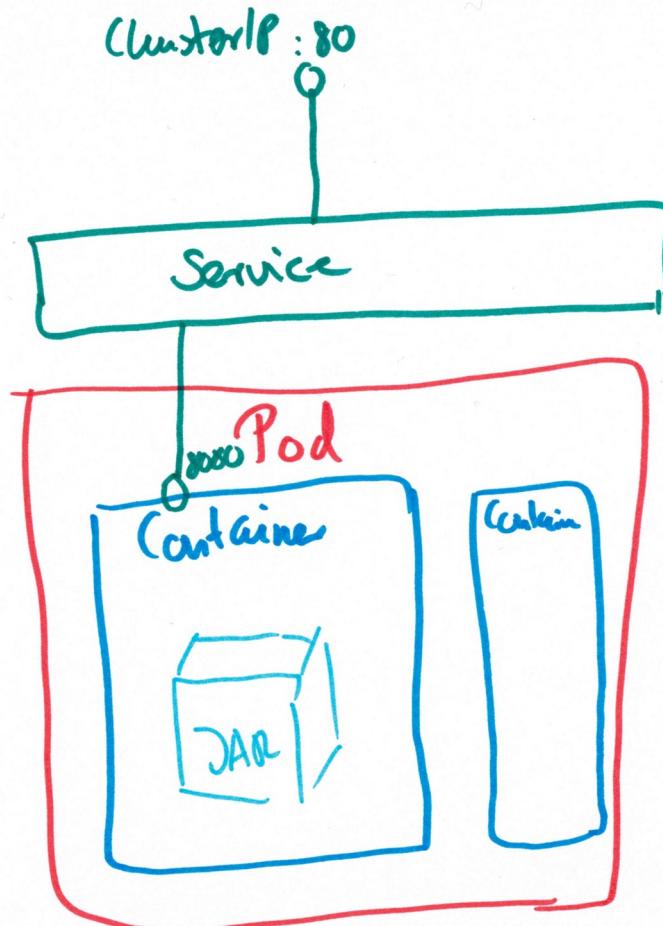


```
apiVersion: v1
kind: Pod
metadata:
  name: spring-boot-demo
spec:
  containers:
  - name: hero-app
    image: sparsick/spring-boot-demo:1.5.0
    ports:
    - name: container-http
      containerPort: 8080
      protocol: TCP
    livenessProbe:
      httpGet:
        path: /actuator/health/liveness
        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

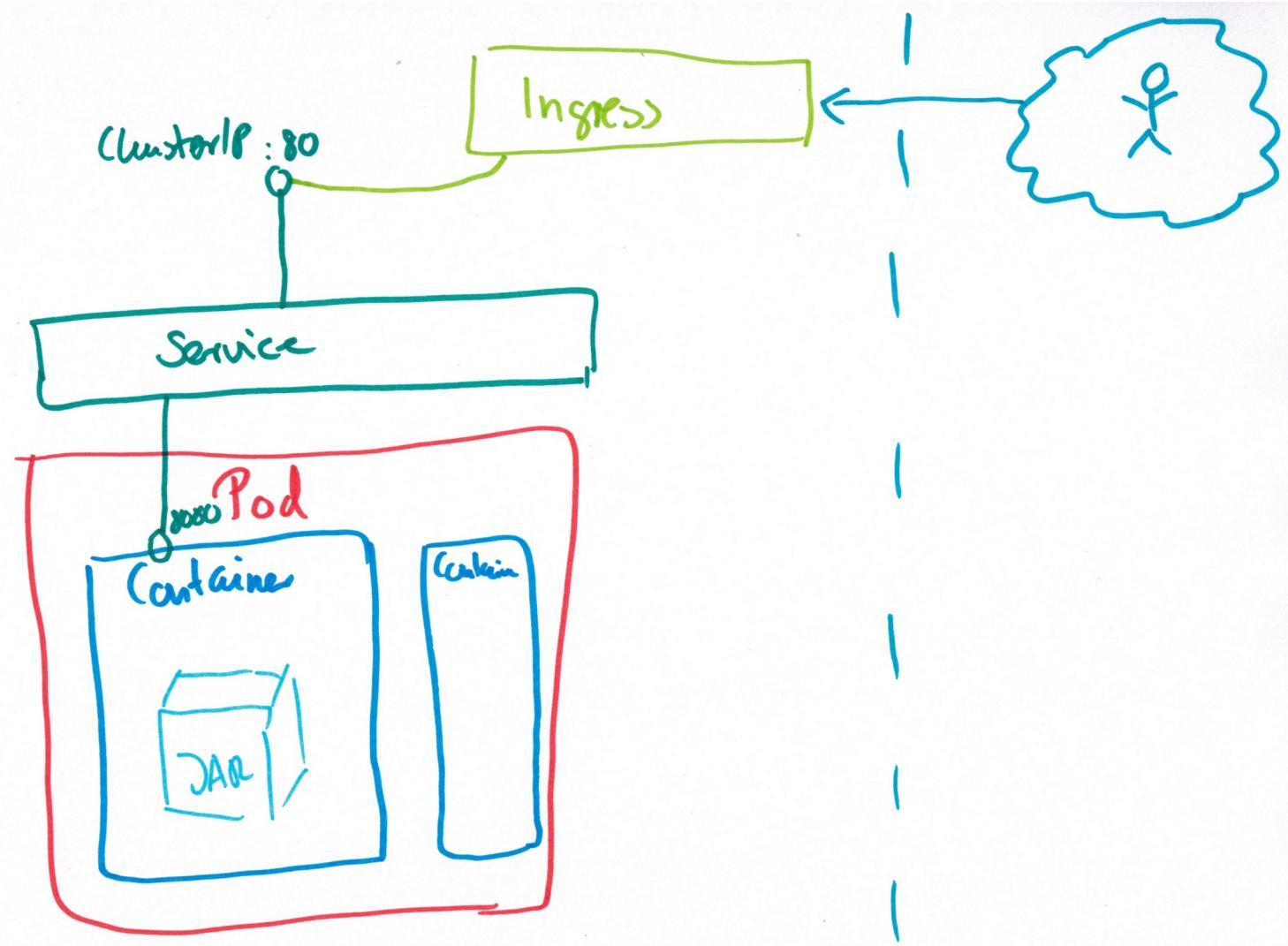


```
apiVersion: v1
kind: Pod
metadata:
  name: spring-boot-demo
  labels:
    app.kubernetes.io/name: hero
spec:
  containers:
  - name: hero-app
    image: sparsick/spring-boot-demo:1.5.0
    ports:
    - name: container-http
      containerPort: 8080
      protocol: TCP

---
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

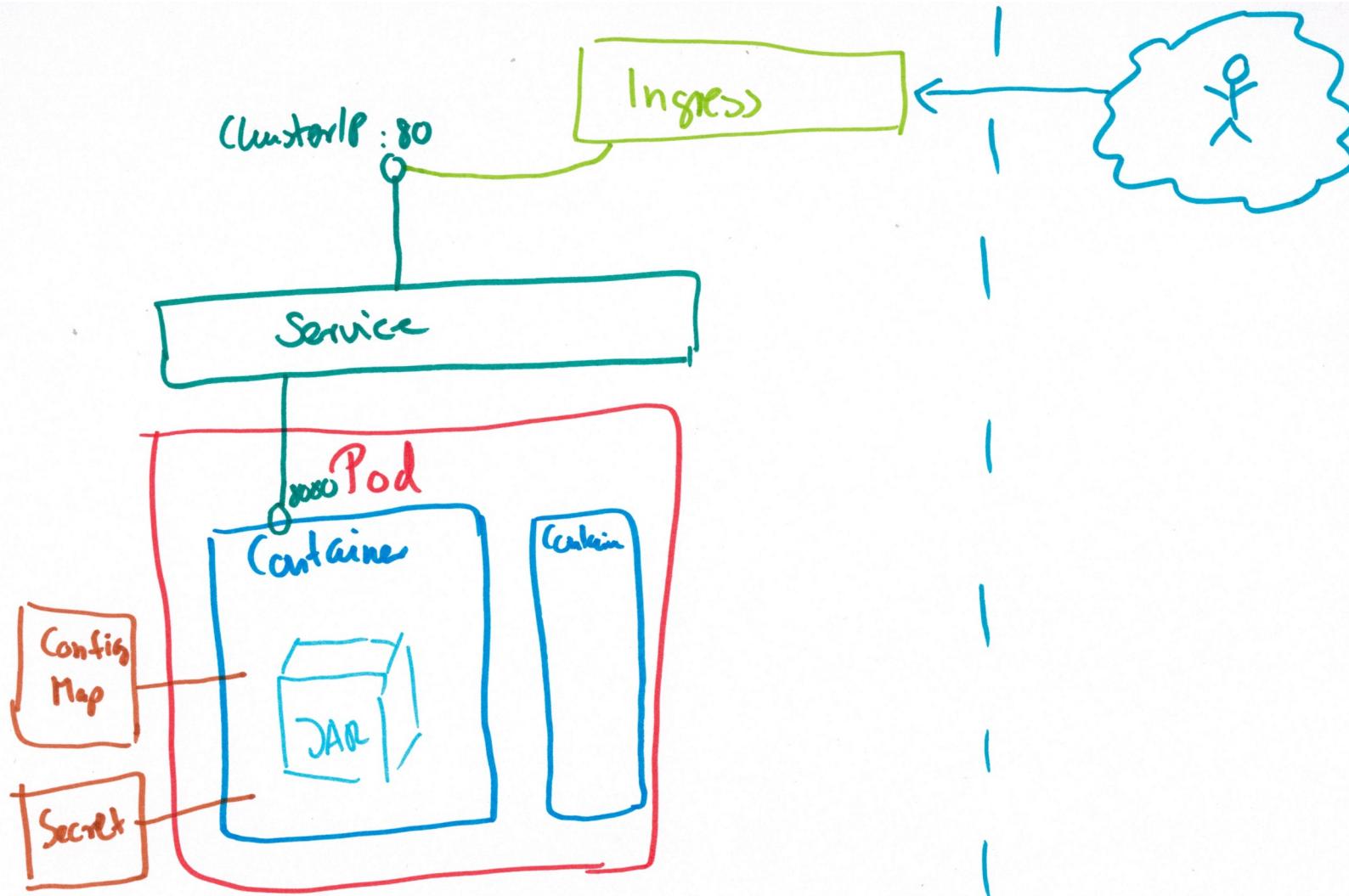
# Ingress



```
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
```

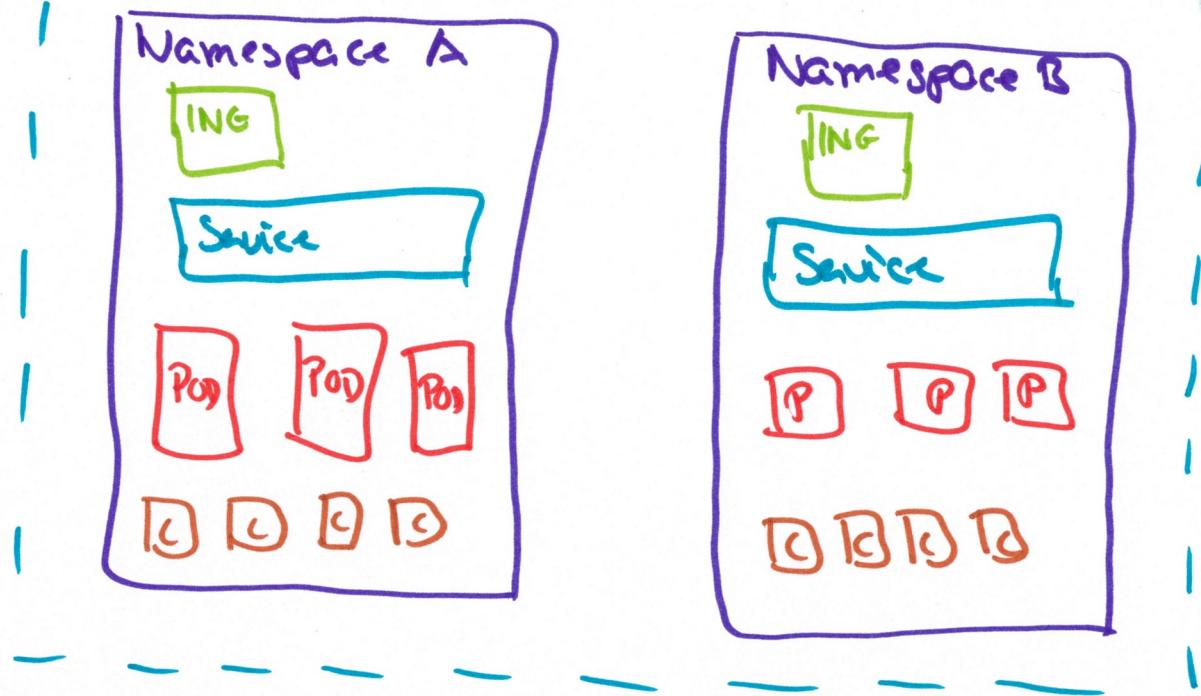


```
apiVersion: v1
kind: Pod
metadata:
  name: spring-boot-demo
spec:
  containers:
    - name: hero-app
      image: sparsick/spring-boot-demo:1.5.0
      ports:
        - name: container-http
          containerPort: 8080
          protocol: TCP
  envFrom:
    - configMapRef:
        name: spring-boot-demo-config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

# Demo

# Namespace

# CLUSTER



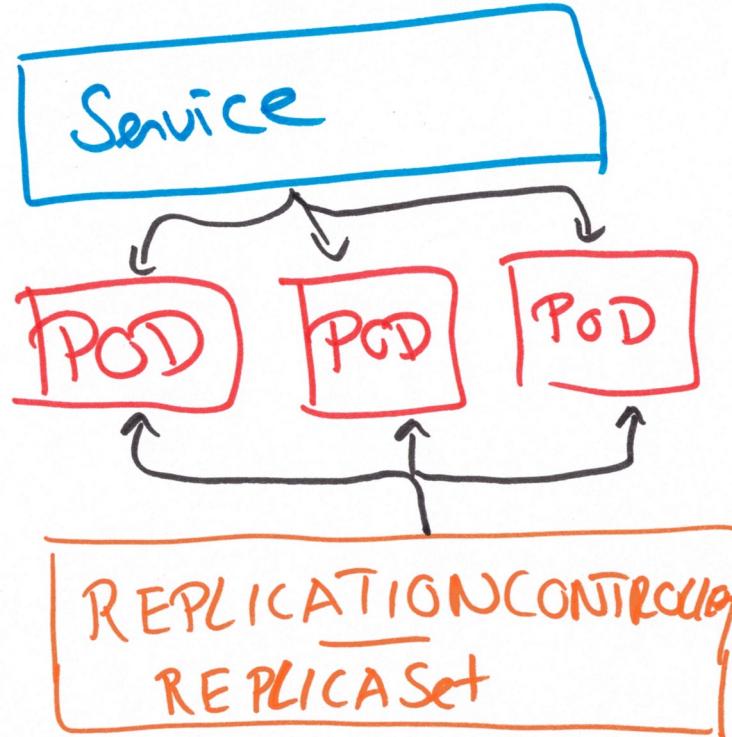


```
apiVersion: v1
kind: Namespace
metadata:
  name: spring-boot-demo-namespace

---
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

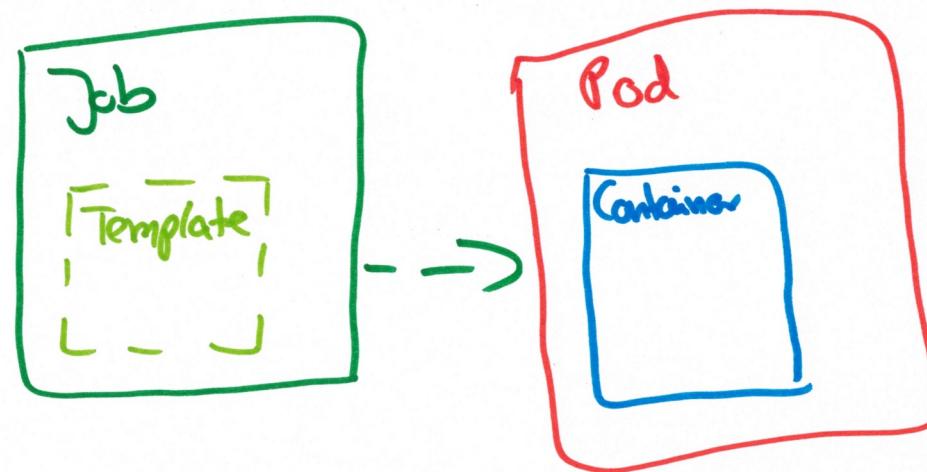




```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: spring-boot-demo-deploy
  namespace: spring-boot-demo-namespace
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hero-app
  template:
    metadata:
      labels:
        app: hero-app
  spec:
    containers:
    - name: hero-app
 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

# Demo

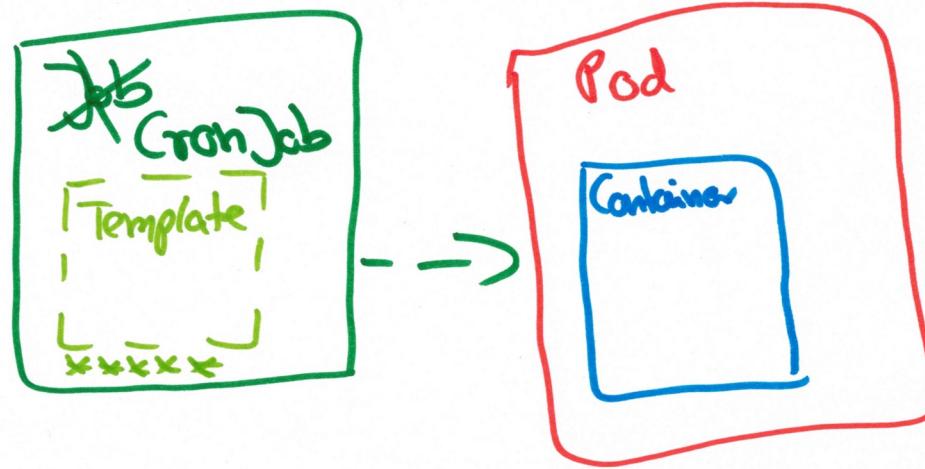
# Job





```
apiVersion: batch/v1
kind: Job
metadata:
  name: demo-job
spec:
  template:
    metadata:
      name: hello-world-job
    spec:
      containers:
        - name: job
          image: busybox
          command: ["echo", "Hello World"]
      restartPolicy: OnFailure
```

# Demo



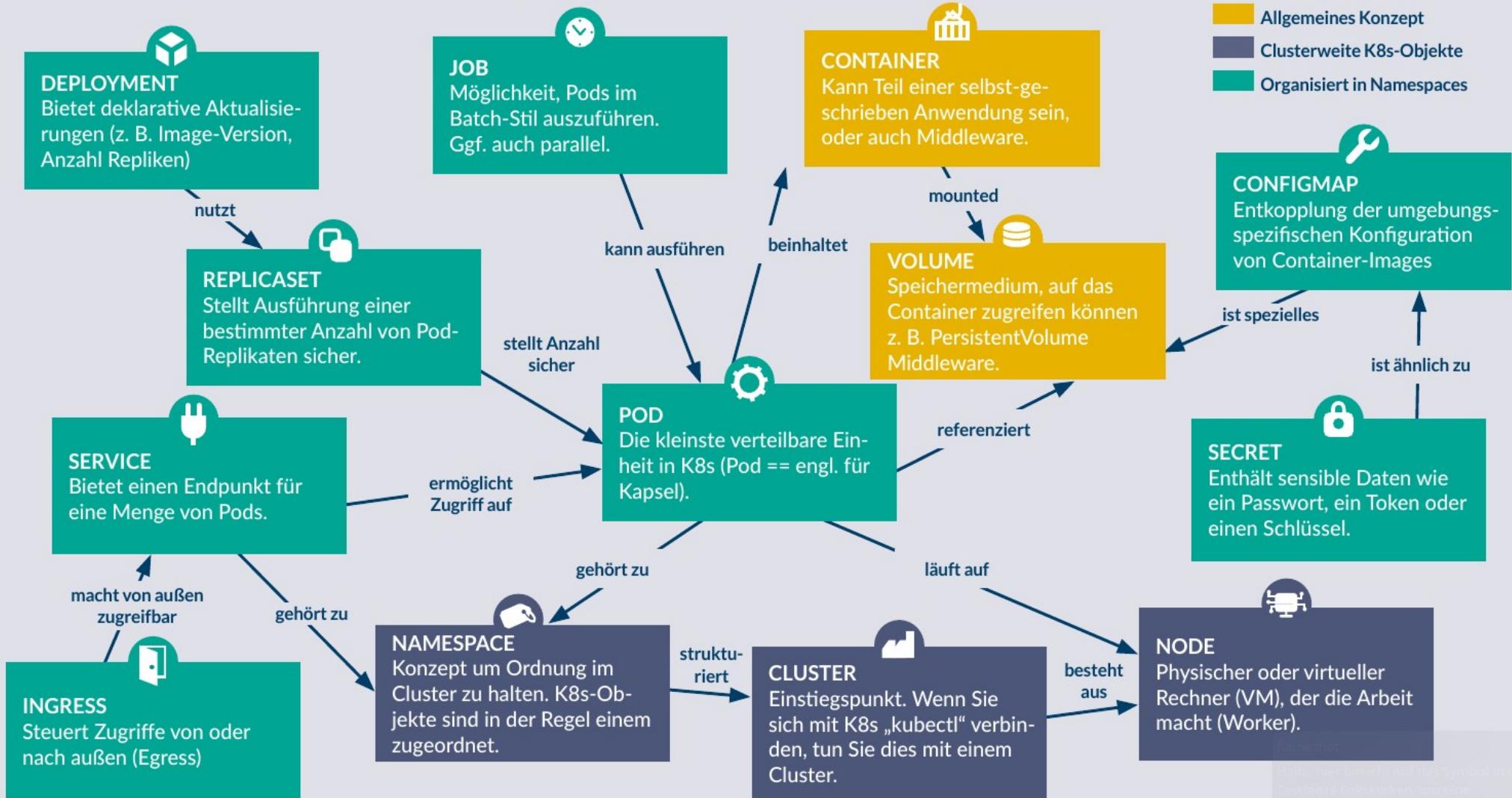


```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: cronjob-demo
spec:
  schedule: "* * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: hello-world-cronjob
              image: busybox
              imagePullPolicy: IfNotPresent
              command:
                - /bin/sh
                - -c
                - date; echo "Hello World, again"
          restartPolicy: OnFailure
```

# 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



### CONFIGMAP

Entkopplung der umgebungs-spezifischen Konfiguration von Container-Images



### SECRET

Enthält sensible Daten wie ein Passwort, ein Token oder einen Schlüssel.



### NODE

Physischer oder virtueller Rechner (VM), der die Arbeit macht (Worker).



Haben Sie kein Bild auf das Symbol in der Legende geklickt, um eine Verbindung herzustellen, so kann es vorkommen, dass keine Verbindungen zwischen den Begriffen gezeigt werden.

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

Fragen?

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

@sparsick@mastodon.social

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

# Weitere Informationen

- <https://www.informatik-aktuell.de/entwicklung/methoden/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>