

JCON Europe, 20.06.2023

# Kubernetes Developer Survival Kit

Sandra Parsick

@SandraParsick

@sparsick@mastodon.social

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

 @SandraParsick

 @sparsick@mastodon.social

 <https://www.sandra-parsick.de>

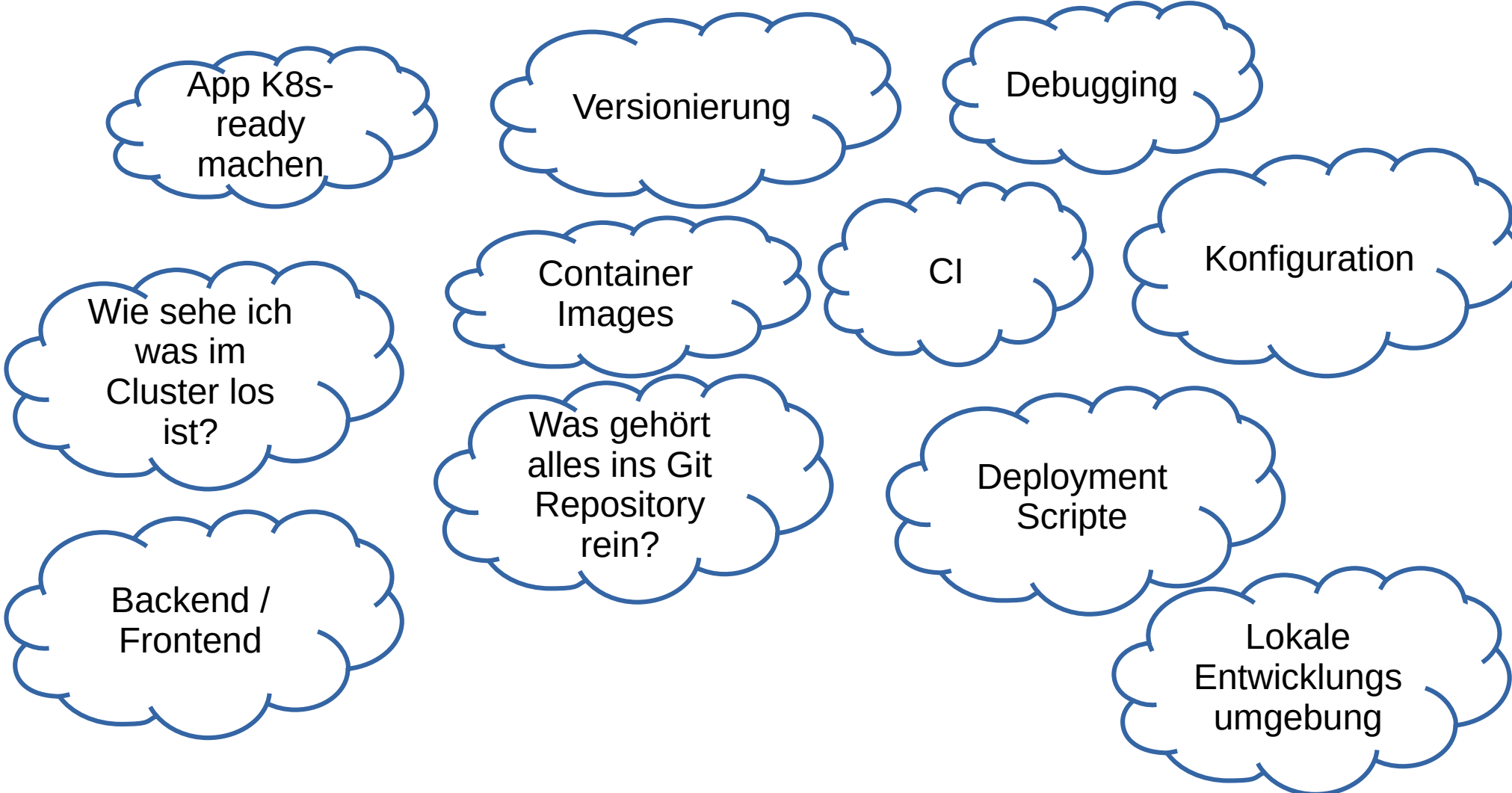
 <https://ready-for-review.dev>











App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

Was gehört  
alles ins Git  
Repository  
rein?

Deployment  
Scripte

Backend /  
Frontend

Lokale  
Entwicklungs-  
umgebung



# Friendly Reminder: 12 Factor App

## **I. Codebase**

One codebase tracked in revision control, many deploys

## **II. Dependencies**

Explicitly declare and isolate dependencies

## **III. Config**

Store config in the environment

## **IV. Backing services**

Treat backing services as attached resources

## **V. Build, release, run**

Strictly separate build and run stages

## **VI. Processes**

Execute the app as one or more stateless processes

# Friendly Reminder: 12 Factor App

## **VII. Port binding**

Export services via port binding

## **VIII. Concurrency**

Scale out via the process model

## **IX. Disposability**

Maximize robustness with fast startup and graceful shutdown

## **X. Dev/prod parity**

Keep development, staging, and production as similar as possible

## **XI. Logs**

Treat logs as event streams

## **XII. Admin processes**

Run admin/management tasks as one-off processes



App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

Was gehört  
alles ins Git  
Repository  
rein?

Deployment  
Scripte

Backend /  
Frontend

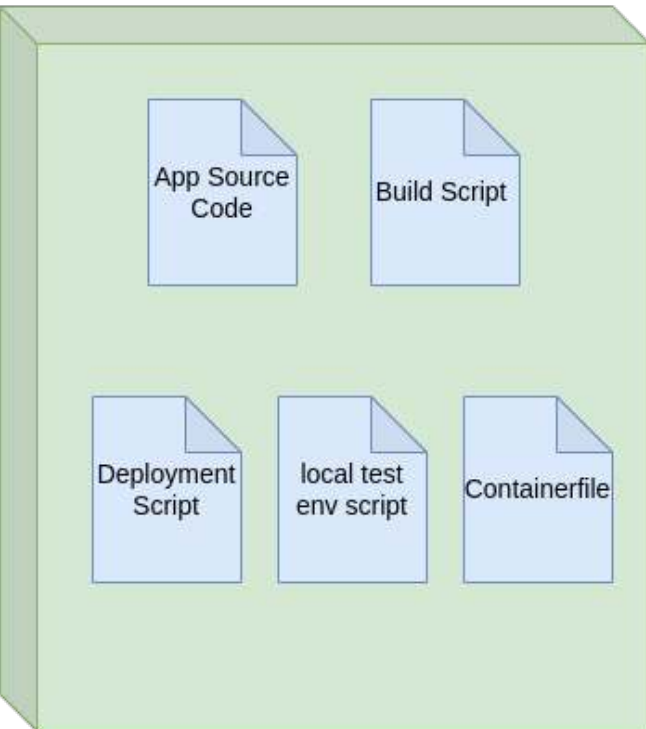
Lokale  
Entwicklungs-  
umgebung

Kurzform: ALLES

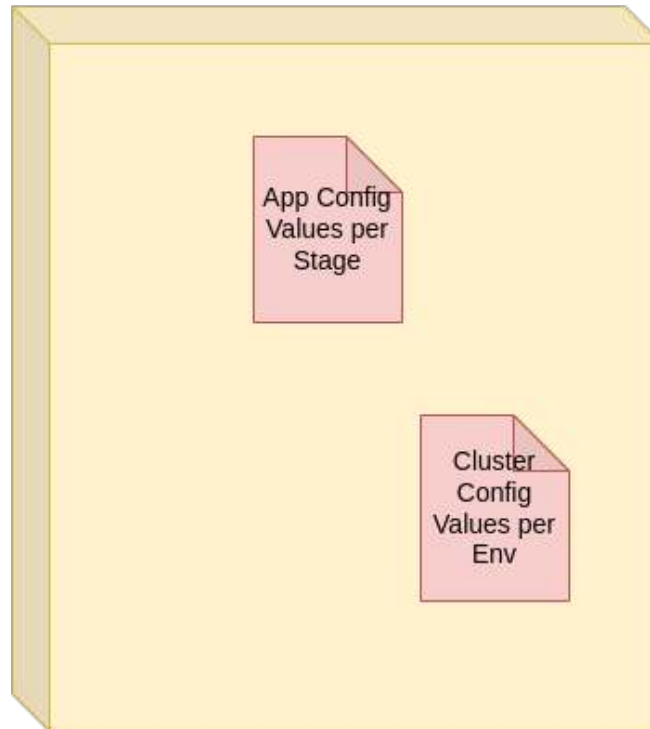
Eigentliche Fragestellung:  
Wieviele Repositories?

# Beispiel für eine Aufteilung

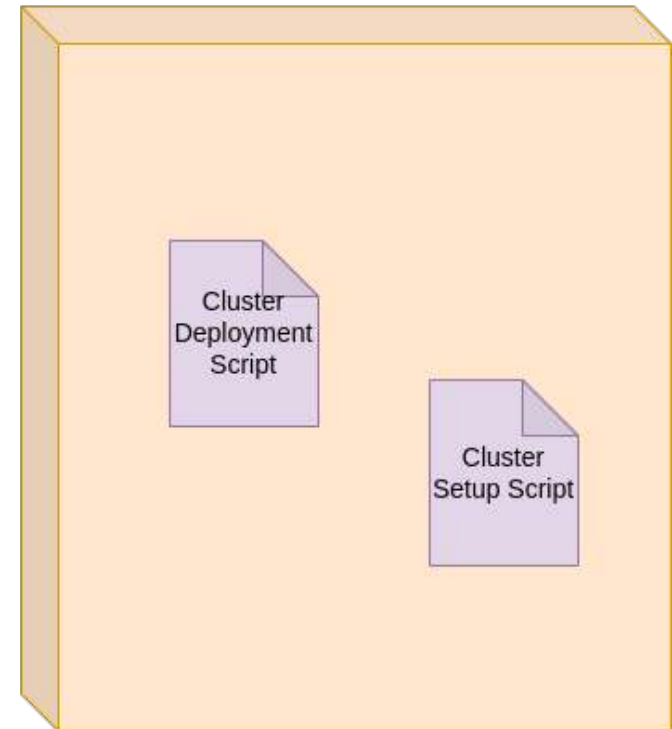
**Application Git Repository**



**Config Value Repository**



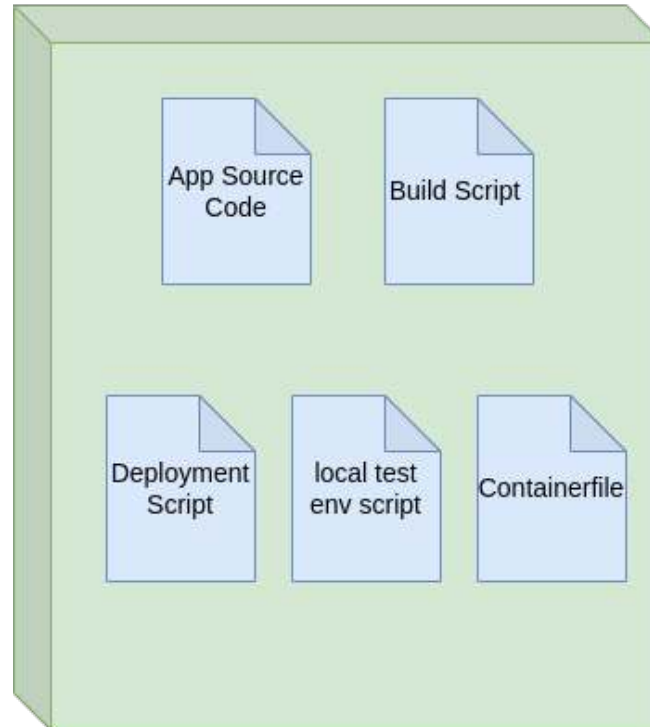
**Cluster Setup Script Repository**



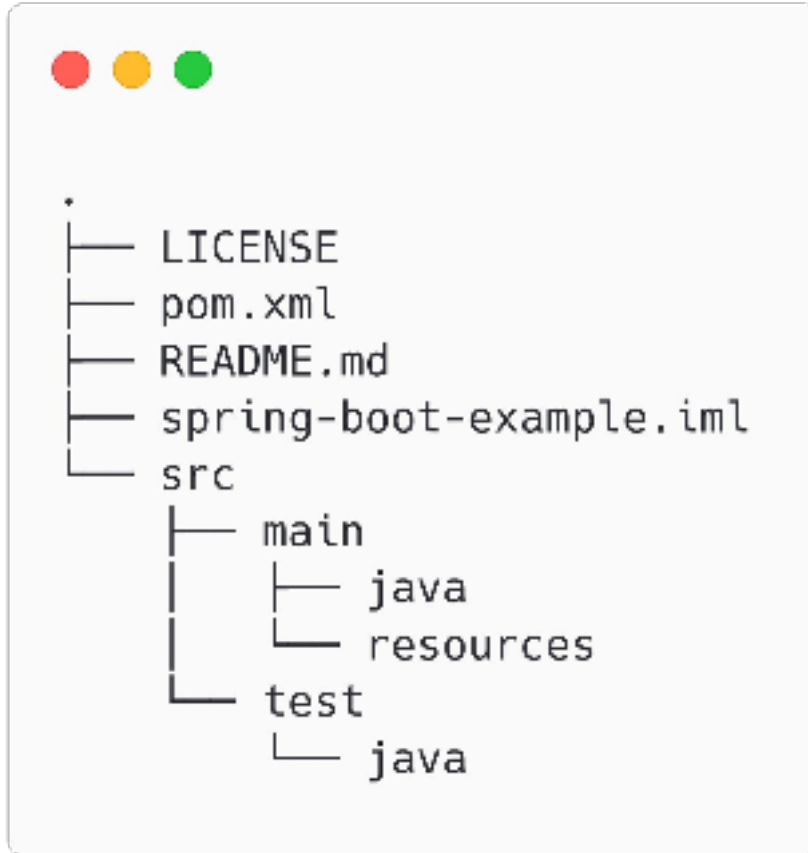


# Für Devs am wichtigsten

## Application Git Repository



# Ausgangspunkt eine Java App



## Technologiestack:

- Java 17
- Spring Boot 3.1.x
- Thymeleaf
- Apache Maven

App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

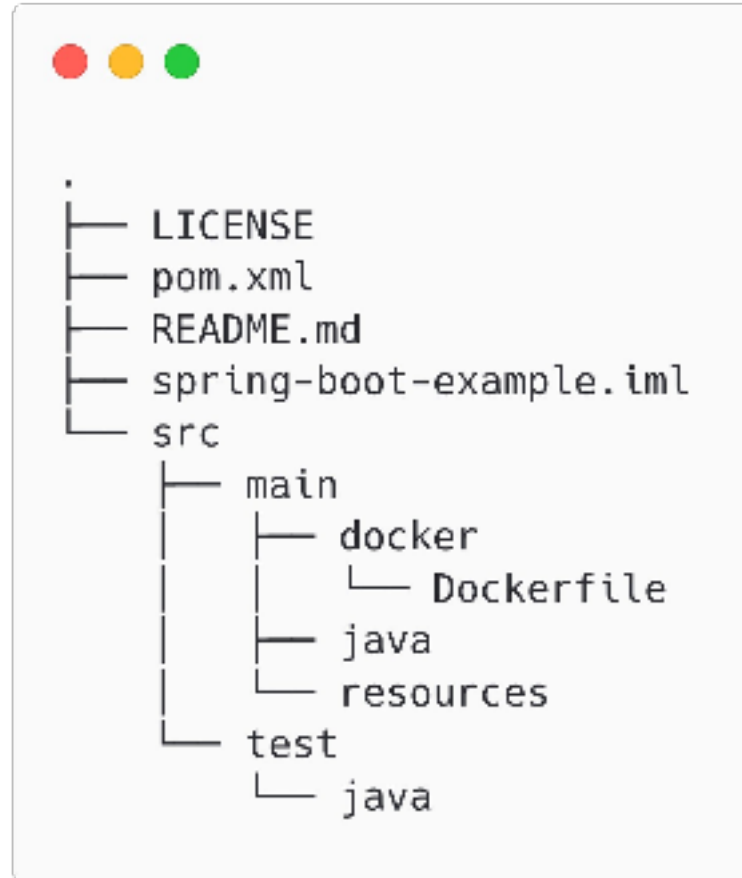
Was gehört  
alles ins Git  
Repository  
rein?

Deployment  
Scripte

Backend /  
Frontend

Lokale  
Entwicklungs-  
umgebung

# Basis: Container





# Basis: Container



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

```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>docker-maven-plugin</artifactId>
  <version>0.40.0</version>
  <executions>
    <execution>
      <id>docker-build</id>
      <goals>
        <goal>build</goal>
        <goal>push</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <images>
      <image>
        <name>spring-boot-demo:latest</name>
        <build>
          <dockerFile>Dockerfile</dockerFile>
          <assembly>
            <descriptorRef>artifact</descriptorRef>
          </assembly>
        </build>
      </image>
    </images>
    <pushRegistry>localhost:6000</pushRegistry>
  </configuration>
</plugin>
```

# Alternativen

- Buildpacks (spring-boot-maven-plugin)
- JIB (jib-maven-plugin)
- Buildah
- Podman
- Weitere Infos im Artikel „Container-Images Deep Dive“ auf Informatik Aktuell

Container-Image-Bau ist Teil des Buildprozess  
und lokal ausführbar



# Good Practises Container Image Build

- unnötige Tools aus dem Image entfernen
  - nur ein Service pro Image verpacken
  - kleine Image bauen
  - Build-Cache optimieren
- Eigene Container Registry benutzen
  - Tags beim Releasen nur einmal verwenden
- Vulnerability-Scans der Container Images

# Optimierter Container Image



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



# Container Registry

- Cloud Provider:
  - Azure Container Registry
  - AWS Elastic Container Registry
  - Google Container Registry
- On Premise:
  - JFrog Container Registry
  - Red Hat Quay
  - Harbor
  - Artifactory
  - Sonatype Nexus

# Vulnerability-Scans (Bsp.: Trivy)

```
→ trivy i --ignore-unfixed -o result spring-boot-demo:latest
2022-06-23T09:55:56.244+0200 INFO Vulnerability scanning is enabled
2022-06-23T09:55:56.245+0200 INFO Secret scanning is enabled
2022-06-23T09:55:56.245+0200 INFO If your scanning is slow, please try '--security-checks vuln'
to disable secret scanning
2022-06-23T09:55:56.245+0200 INFO Please see also https://aquasecurity.github.io/trivy/v0.29.2
/docs/secret/scanning/#recommendation for faster secret detection
2022-06-23T09:55:56.254+0200 INFO Detected OS: debian
2022-06-23T09:55:56.255+0200 INFO Detecting Debian vulnerabilities...
2022-06-23T09:55:56.265+0200 INFO Number of language-specific files: 1
2022-06-23T09:55:56.265+0200 INFO Detecting jar vulnerabilities...

spring-boot-demo:latest (debian 11.2)
=====
Total: 23 (UNKNOWN: 1, LOW: 2, MEDIUM: 6, HIGH: 6, CRITICAL: 8)

Java (jar)
=====
Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)
```

# Vulnerability-Scans (Bsp.: Trivy)



Library	Vulnerability	Severity	Installed Version	Fixed Version	Title
libc6	CVE-2021-33574	CRITICAL	2.31-13+deb11u2	2.31-13+deb11u3	glibc: sq_notify does not handle separately allocated thread attributes <a href="https://avd.aquasec.com/nvd/cve-2021-33574">https://avd.aquasec.com/nvd/cve-2021-33574</a>
	CVE-2022-23218				glibc: Stack-based buffer overflow in secanix_create via long pathnames <a href="https://avd.aquasec.com/nvd/cve-2022-23218">https://avd.aquasec.com/nvd/cve-2022-23218</a>
	CVE-2022-23219				glibc: Stack-based buffer overflow in sunrpc clnt_create via a long pathname <a href="https://avd.aquasec.com/nvd/cve-2022-23219">https://avd.aquasec.com/nvd/cve-2022-23219</a>
	CVE-2021-43396	LOW			glibc: Conversion from ISO-2022-JP-3 with iconv may emit spurious NUL character on... <a href="https://avd.aquasec.com/nvd/cve-2021-43396">https://avd.aquasec.com/nvd/cve-2021-43396</a>



# Vulnerability-Scans

## Weitergedacht

- Was ist mit
  - Container in der Registry
  - Container, die schon im Cluster laufen



App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

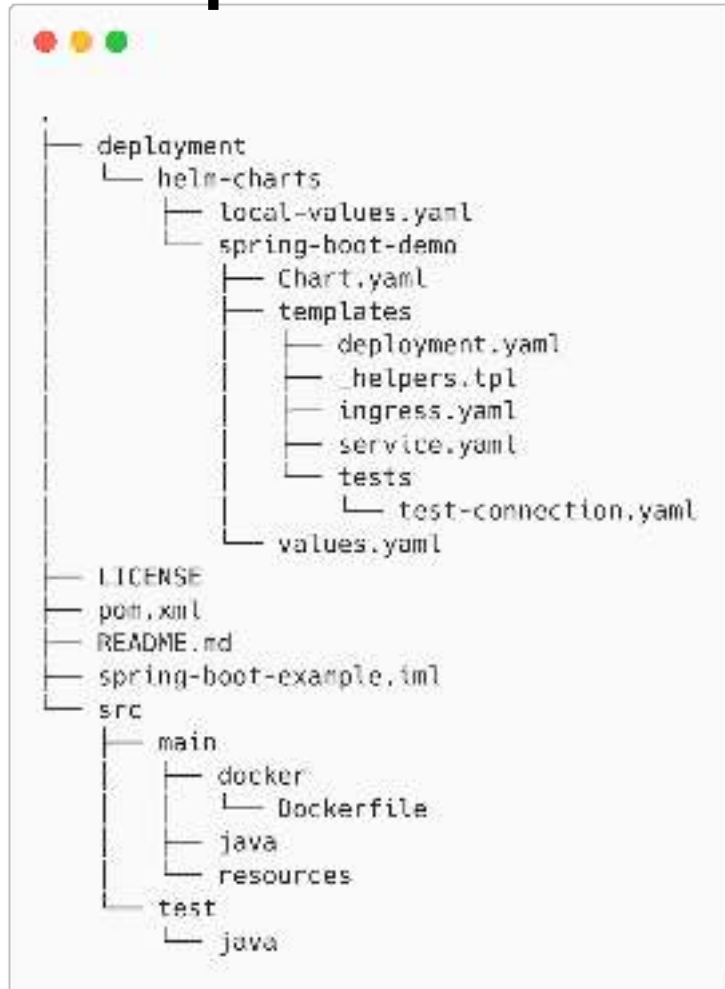
Was gehört  
alles ins Git  
Repository  
rein?

Deployment  
Scripte

Backend /  
Frontend

Lokale  
Entwicklungs-  
umgebung

# Next Step: Helm Charts

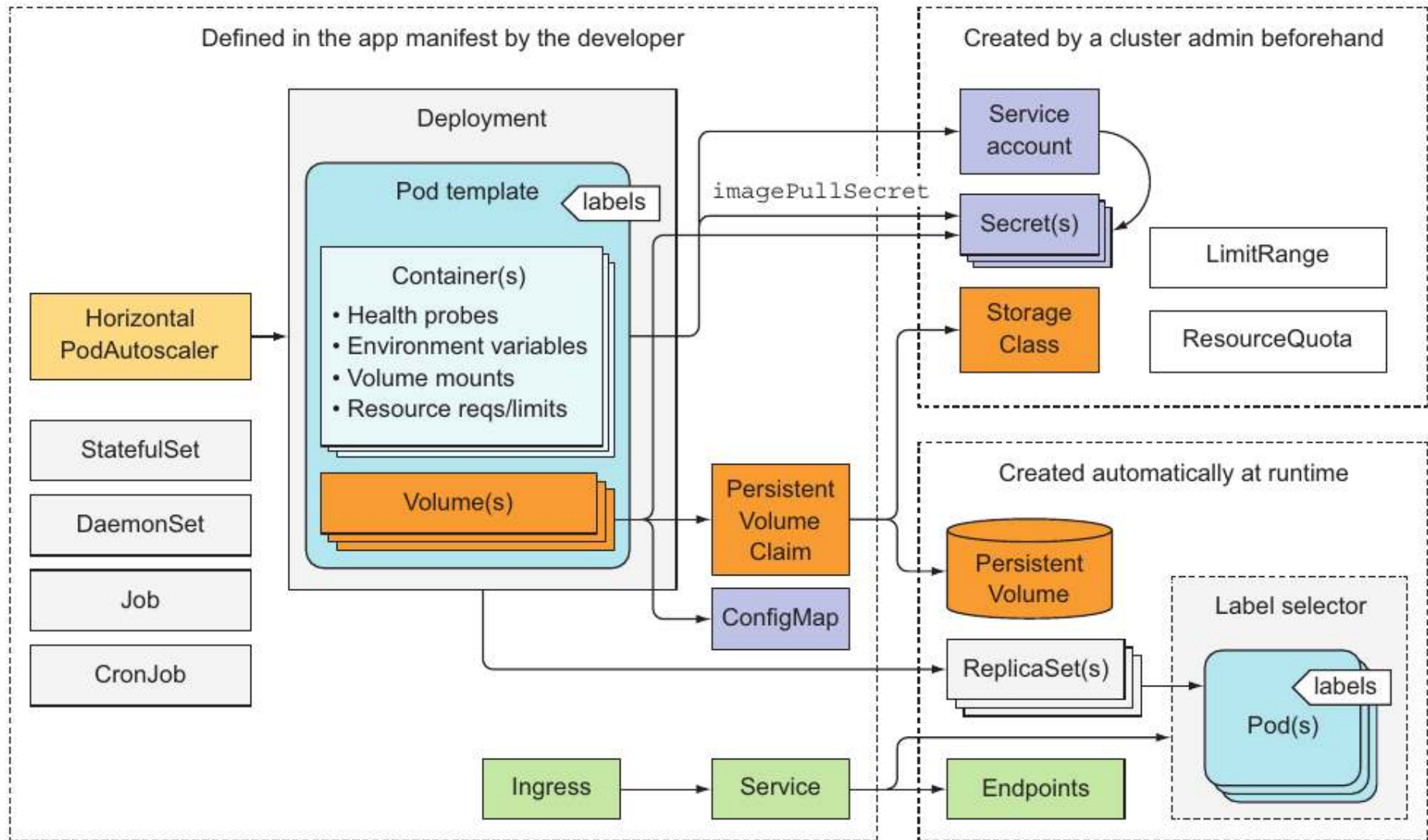


# Auszug: Service Definition



```
apiVersion: v1
kind: Service
metadata:
  name: {{ include "spring-boot-demo.fullname" . }}
  namespace: {{ include "spring-boot-demo.namespaceName" . }}
  labels:
    {{- include "spring-boot-demo.labels" . | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: 8080
      protocol: TCP
      name: http
  selector:
    {{- include "spring-boot-demo.selectorLabels" . | nindent 4 }}
```

Um welche K8s Resource soll ich mich als Dev kümmern?





```
<plugin>
  <groupId>io.kokuwa.maven</groupId>
  <artifactId>helm-maven-plugin</artifactId>
  <version>6.3.0</version>
  <configuration>
    <chartDirectory>${project.basedir}/deployment/helm-charts</chartDirectory>
    <chartVersion>${project.version}</chartVersion>
    <helmVersion>3.8.1</helmVersion>
  </configuration>
  <executions>
    <execution>
      <id>build-chart</id>
      <phase>package</phase>
      <goals>
        <goal>package</goal>
      </goals>
    </execution>
    <execution>
      <id>upload-chart</id>
      <phase>deploy</phase>
      <goals>
        <goal>upload</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

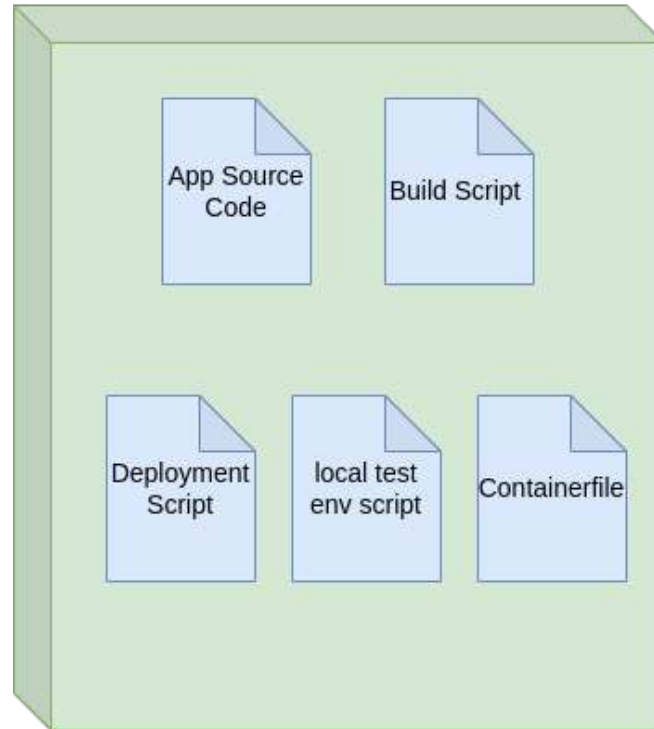
Helm Charts Paketierung Teil des Build Prozesses

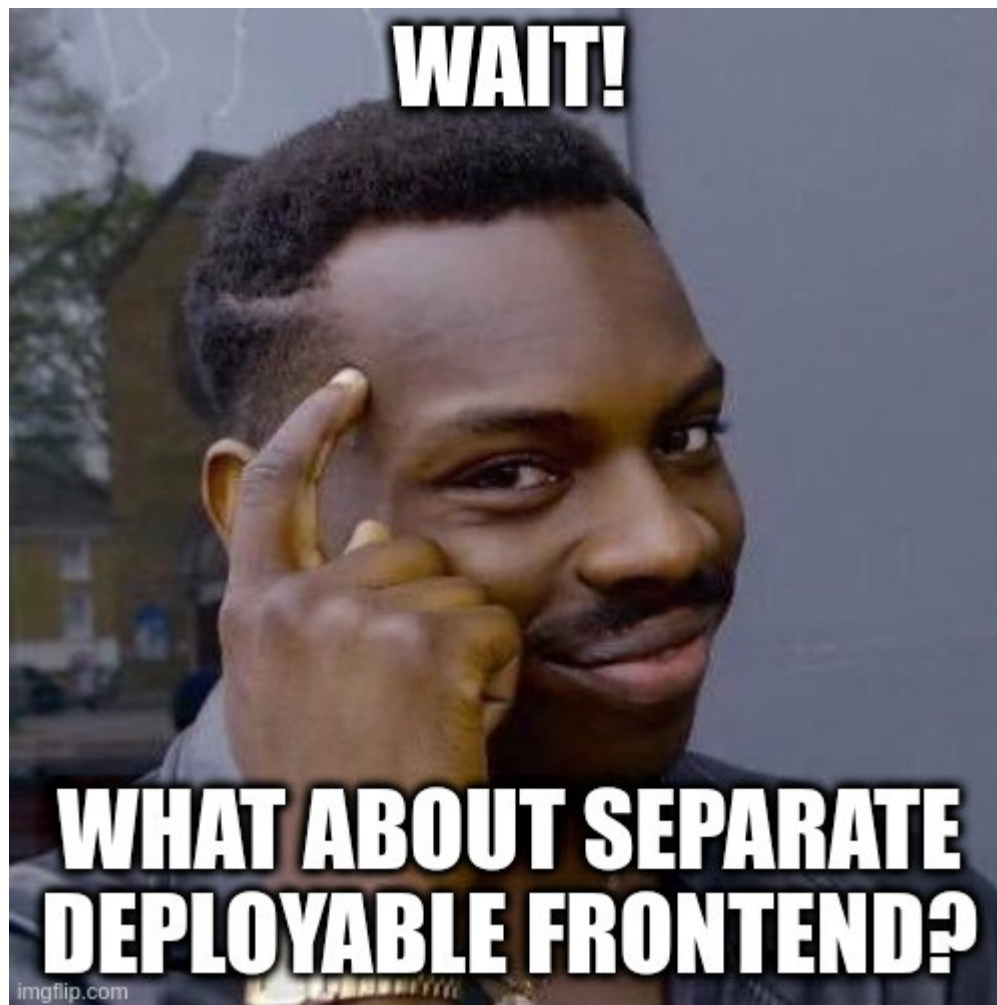


# Helm Chart Repository

- Allgemein:
  - Jede Container Registry kann dafür genutzt werden
- Darauf spezialisiert:
  - Chartmuseum
  - JFrog Container Registry
  - Artifactory
  - Sonatype Nexus

## Application Git Repository





App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

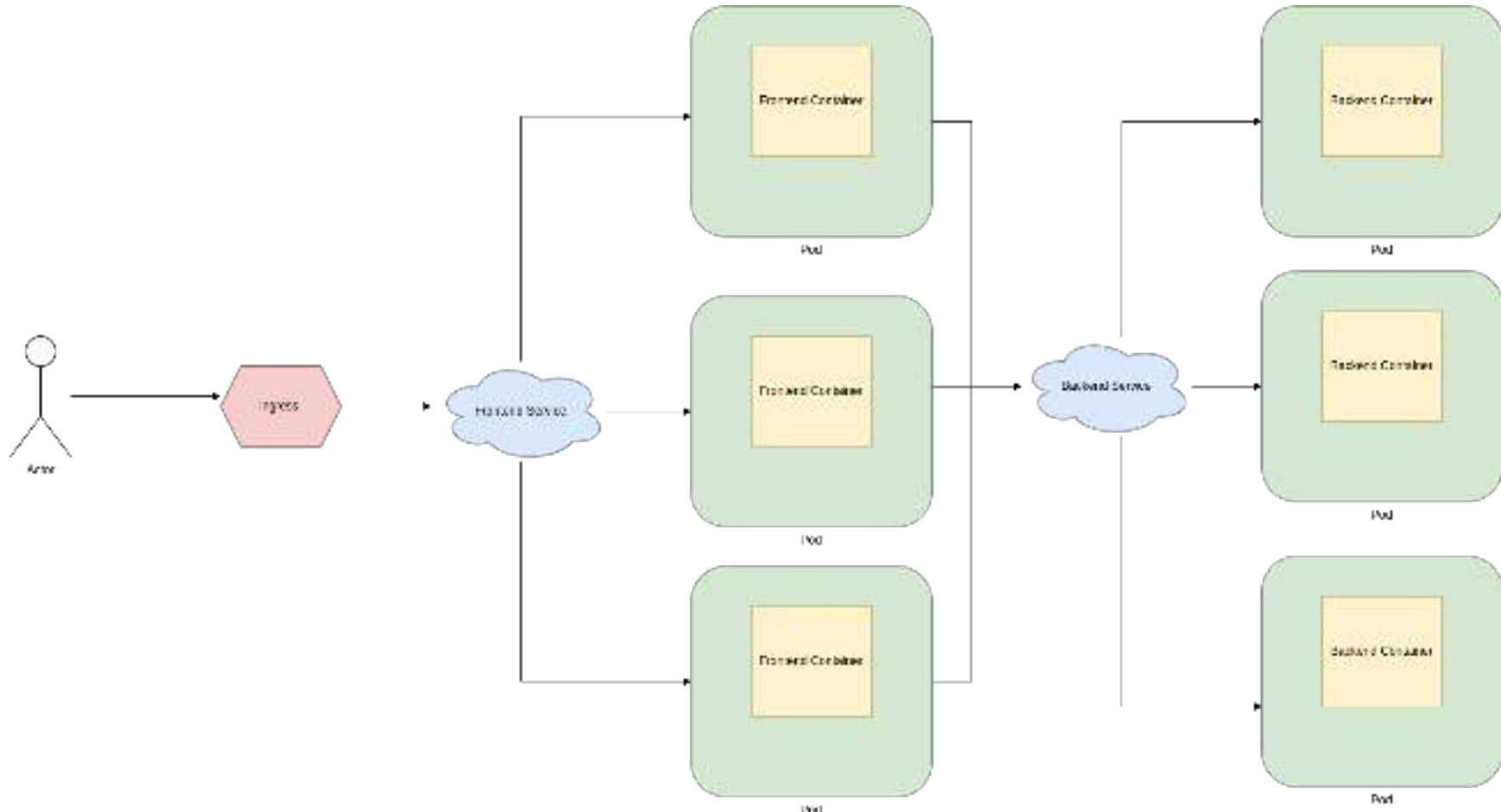
Was gehört  
alles ins Git  
Repository  
rein?

Deployment  
Scripte

Backend /  
Frontend

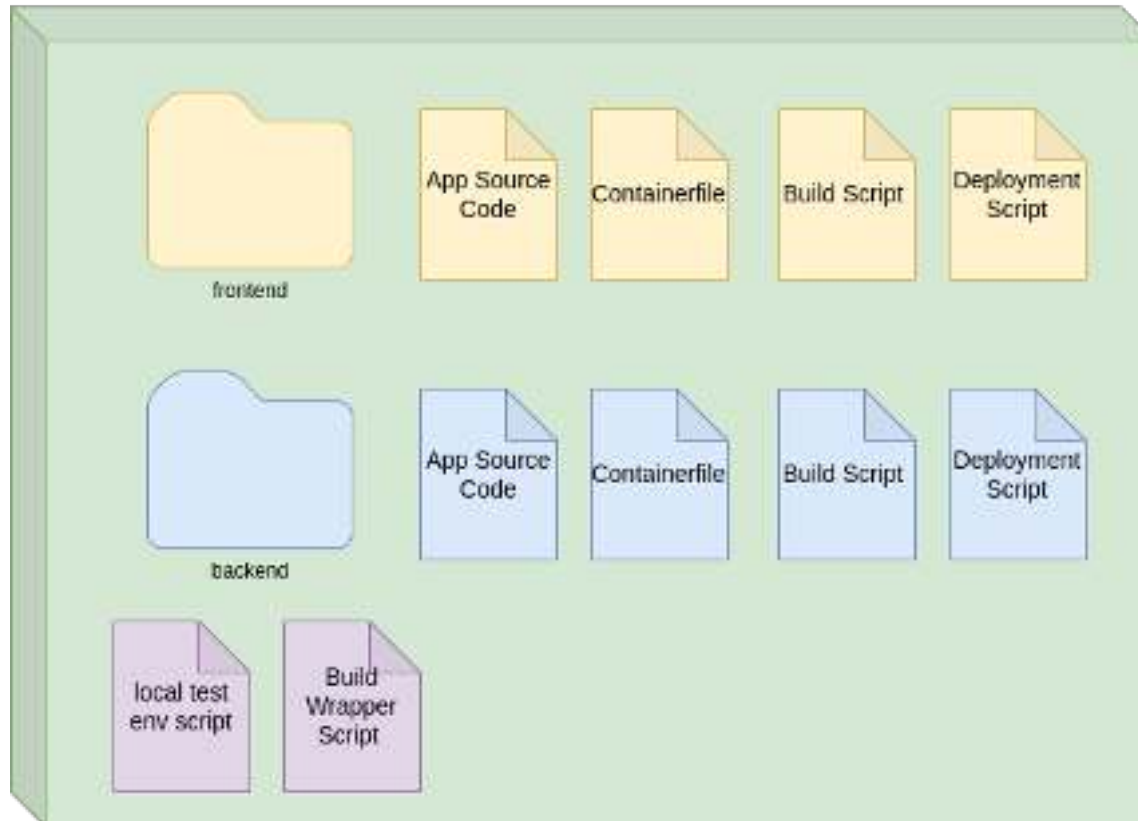
Lokale  
Entwicklungs-  
umgebung

# Frontend und Backend in K8s



# Git Repository Struktur

## Application Git Repository



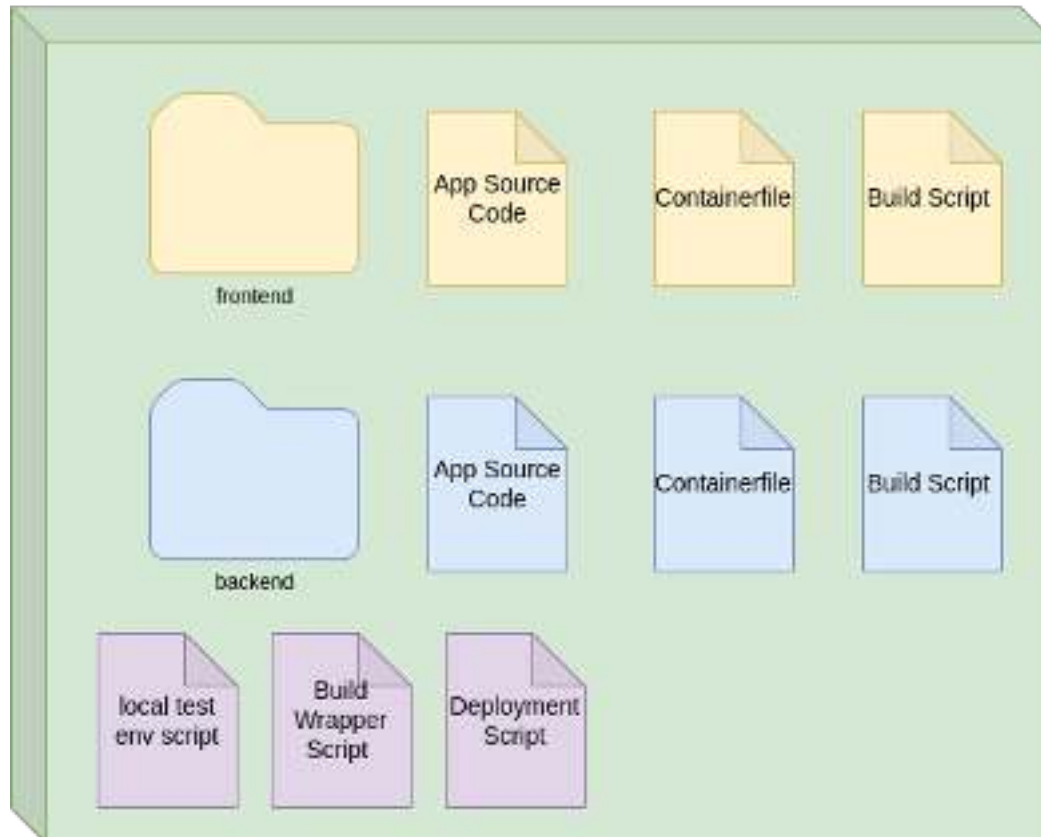
- Ingress
- Frontend Service
- Frontend Deployment



- Backend Service
- Backend Deployment

# Git Repository Struktur

## Application Git Repository



- Ingress
- Frontend Service
- Frontend Deployment
- Backend Service
- Backend Deployment

App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

Was gehört  
alles ins Git  
Repository  
rein?

Deployment  
Scripte

Backend /  
Frontend

Lokale  
Entwicklungs-  
umgebung



Fangt einfach an:  
Eine Versionsnummer über alle Artifakte

## Application Artifacts



App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI


Konfiguration

Was gehört  
alles ins Git  
Repository  
rein?


Deployment  
Scripte

Backend /  
Frontend

Lokale  
Entwicklungs-  
umgebung



Applikation  
lokal testen



Deployment  
Skripte lokal  
entwickeln

# Applikation lokal testen



Spring Boot Maven Plugin

# Applikation lokal testen



```
version: "3.9"
services:
  database:
    image: mongo:4.2.21
    restart: always
    ports:
      - 27017:27017
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: root123
    volumes:
      - ./local-env/./dockerentrypoint-initdb.d/
```



```
<plugin>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-maven-plugin</artifactId>
  <configuration>
    <layers>
      <enabled>true</enabled>
    </layers>
    <environmentVariables>
      <MONGODB_ENABLED>true</MONGODB_ENABLED>
      <MONGODB_URI>nongodb://test:test123@localhost/test</MONGODB_URI>
    </environmentVariables>
  </configuration>
</plugin>
```



```
version: "3.9"
services:
  demo-app:
    image: spring-boot-demo:latest
    restart: always
    ports:
      - 80:8080
    environment:
      MONGODB_ENABLED: "true"
      MONGODB_URI: mongodb://test:test123@database:27017/test
    depends_on:
      - database

  database:
    image: mongo:4.2.21
    restart: always
    ports:
      - 27017:27017
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: root123
    volumes:
      - ./local-env:/docker-entrypoint-initdb.d/
```

# Spring Boot Docker Compose Support

```
<!-- Since Spring Boot 3.1 -->
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-docker-compose</artifactId>
    <optional>true</optional>
  </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <configuration>
        <layers>
          <enabled>true</enabled>
        </layers>
      </configuration>
    </plugin>
  </plugins>
</build>
```



```
# application.properties
# default: compose.yml
spring.docker.compose.file=docker-compose.yml
```





Andere  
Abhängigkeiten?

# Mocking

- <https://www.mock-server.com>
- <https://github.com/navikt/mock-oauth2-server>



version: "3.9"

services:

mockserver:

image: mockserver/mockserver:latest

restart: always

ports:

- 1080:1080

environment:

MOCKSERVER\_INITIALIZATION\_JSON\_PATH: /config/expectation.json

volumes:

- ./local-env/mockserver:/config



```
[
  {
    "httpRequest": {
      "path": "/success"
    },
    "httpResponse": {
      "body": "Successful!"
    }
  },
  {
    "httpRequest": {
      "path": "/fail"
    },
    "httpResponse": {
      "statusCode": 400
    }
  }
]
```

# Deployment Skripte lokal entwickeln



**minikube**

# Alternativen zu Minikube

- k3s
- k3d
- kind
- microk8s
- k0s

App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

Was gehört  
alles ins Git  
Repository  
rein?

Deployment  
Scripte

Backend /  
Frontend

Lokale  
Entwicklungs-  
umgebung

# 12 Factor App: Die Konfiguration in Umgebungsvariablen ablegen



# Applikation vorbereiten



```
# snippet application.properties
```

```
spring.data.mongodb.uri=${MONGODB_URI:mongodb://localhost/test}
```

```
mongodb.enabled=${MONGODB_ENABLED:false}
```

# Helm Charts anpassen



```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ include "spring-boot-demo.fullname" . }}-config
  namespace: {{ include "spring-boot-demo.namespaceName" . }}
  labels:
    {{- include "spring-boot-demo.labels" . | nindent 4 }}
data:
  MONGODB_URI: "{{ .Values.mongodb.uri }}"
  MONGODB_ENABLED: "{{ .Values.mongodb.enabled }}"
```

# Helm Charts anpassen



```
# code snippet with the important part
apiVersion: apps/v1
kind: Deployment
# ...
spec:
  template:
    metadata:
      annotations:
        checksum/config: {{ include (print $.Template.BasePath "/config.yaml") . | sha256sum }}
    spec:
      containers:
      - name: {{ .Chart.Name }}
        image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
        args: [{{ .Values.spring_boot_demo_chart.container_args }}]
        imagePullPolicy: {{ .Values.image.pullPolicy }}
        envFrom:
        - configMapRef:
            name: {{ include "spring-boot-demo.fullname" . }}-config
```

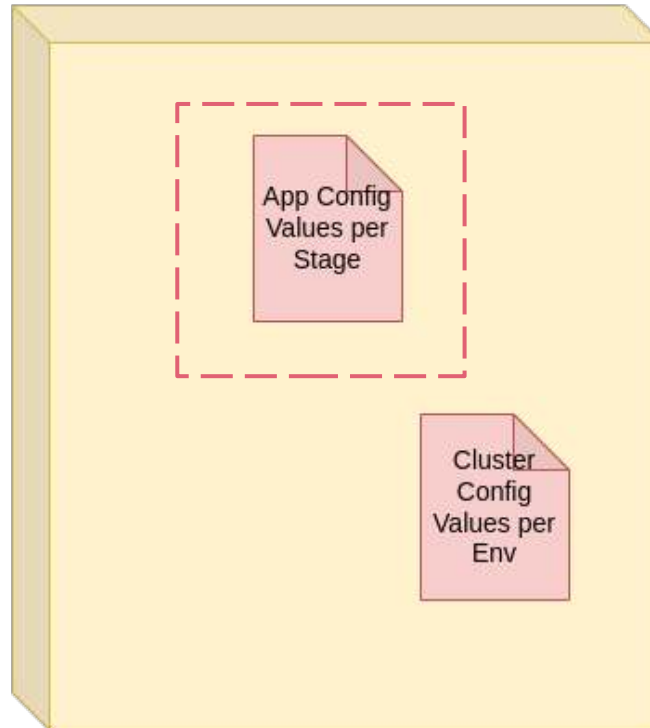
# Helm Charts anpassen



```
# code snippet with the important part from value.yaml  
mongodb:  
  enabled: false  
  uri: mongodb://test:test@localhost/test
```

# Konfiguration verwalten

## Config Value Repository



# Konfiguration verwalten



```
config-value-repo on ? dev  
→ tree
```

```
.  
├── namespace-a  
│   └── appl.yml  
└── registry.yml
```

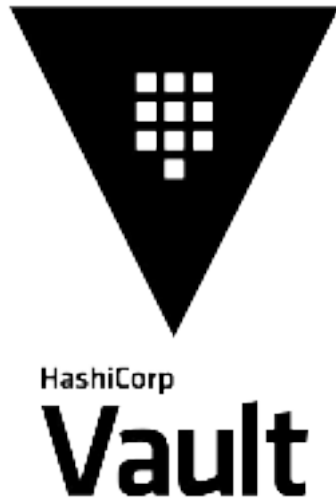
```
→ git branch  
* dev  
  pre-prod  
  prod
```



```
flat-config-value-repo on ? master  
→ tree
```

```
.  
├── dev  
│   ├── namespace-a  
│   │   └── appl.yml  
│   └── registry.yml  
├── pre-prod  
│   ├── namespace-a  
│   │   └── appl.yml  
│   └── registry.yml  
└── prod  
    ├── namespace-a  
    │   └── appl.yml  
    └── registry.yml
```

# Secrets



Cloud Lösungen (Bsp):

- Google Secret Manager
- AWS Secrets & Configuration Provider
- Azure Key Vault Provider

# Helm Secret Plugin



```
→ helm plugin install https://github.com/jkroepke/helm-secrets --version v3.12.0  
→ helm secrets help
```

Secrets encryption **in** Helm Charts

This plugin provides ability to encrypt/decrypt secrets files to store **in** less secure places, before they are installed using Helm.

For more information, see the README at [github.com/jkroepke/helm-secrets](https://github.com/jkroepke/helm-secrets)

To decrypt/encrypt/edit you need to initialize/first encrypt secrets with sops - <https://github.com/mozilla/sops>



# Helm Secret Plugin



```
// sops must be configured  
→ helm secrets enc examples/sops/secrets.yaml  
Encrypting examples/sops/secrets.yaml  
Encrypted examples/sops/secrets.yaml  
→ helm upgrade name . -f secrets://examples/sops/secrets.yaml value.yaml
```

App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

Was gehört  
alles ins Git  
Repository  
rein?

Deployment  
Scripte

Backend /  
Frontend

Lokale  
Entwicklungs-  
umgebung

# Good Practices für Anwendungen in Container

- Nur ein Anwendungsprozess pro Container
  - Ausführung als root vermeiden
  - Privilegierte Container vermeiden
  - Zustandslose Anwendungen bevorzugen
- |  |
|--|
| • Log-Nachrichten auf stdout           |
| • Anwendungsüberwachung bedenken       |
| • Robust hoch- und runterfahren können |



# Log-Nachrichten auf stdout



Tipp: Nutzt Spring Default Logging Settings

# Anwendungsüberwachung



```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
  <groupId>io.micrometer</groupId>
  <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

```
management.metrics.export.prometheus.enabled=true
management.metrics.web.server.request.autotime.enabled=true
management.endpoints.web.exposure.include=prometheus
```

# Robust hoch- und runterfahren



```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>
```



```
management.endpoints.web.exposure.include=info,health
```

# Robust hoch- und runterfahren

```
# code snippet with the important part
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      containers:
      - name: {{ .Chart.Name }}
        image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
        ports:
        - name: container-http
          containerPort: 8080
          protocol: TCP
        livenessProbe:
          httpGet:
            path: /actuator/health/liveness
            port: container-http
            initialDelaySeconds: {{ .Values.livenessProbe.initialDelaySeconds }}
            periodSeconds: {{ .Values.livenessProbe.periodSeconds }}
            timeoutSeconds: {{ .Values.livenessProbe.timeoutSeconds }}
        readinessProbe:
          httpGet:
            path: /actuator/health/readiness
            port: container-http
            initialDelaySeconds: {{ .Values.readinessProbe.initialDelaySeconds }}
            periodSeconds: {{ .Values.readinessProbe.periodSeconds }}
            timeoutSeconds: {{ .Values.readinessProbe.timeoutSeconds }}
```



# Robust hoch- und runterfahren

Wichtig:  
Sichert diese Endpunkte nach außen ab!



# Robust hoch- und runterfahren

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: {{ include "spring-boot-demo.fullname" . }}
  namespace: {{ include "spring-boot-demo.namespaceName" . }}
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
    nginx.ingress.kubernetes.io/x-forwarded-prefix: "/"
    nginx.ingress.kubernetes.io/server-snippet: |
      location ~* "/actuator/" {
        deny all;
        return 403;
      }
spec:
  rules:
    - host: {{ .Values.ingress.host }}
      http:
        paths:
          - path: /(.*)
            pathType: Prefix
            backend:
              service:
                name: {{ include "spring-boot-demo.fullname" . }}
                port:
                  number: 8080
```

# JVM und Kubernetes

- JVM Default Einstellung (welcher GC oder default max Heap Size) abh. von Umgebung und Java Version
- Lesson Learnt: Immer die JVM konfigurieren
- Don't worry: Es gibt Empfehlungen

# JVM und Kubernetes

- Topologie des Kubernetes Cluster ist abhängig von deiner Anwendung
  - Kleine JVMs + viele Replicas vs Große JVMs + wenige Replicas
- Messen, Messen, Messen

# Weitere Infos zu JVM und Kubernetes

- Secrets of Performance Tuning Java on Kubernetes von Bruno Borges
- We Moved one Java Product to Kubernetes and This Is What We Learned von Carlos Sanchez
- Tuning and Optimizing Java Garbage Collection von Monica Beckwith

App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

Was gehört  
alles ins Git  
Repository  
rein?

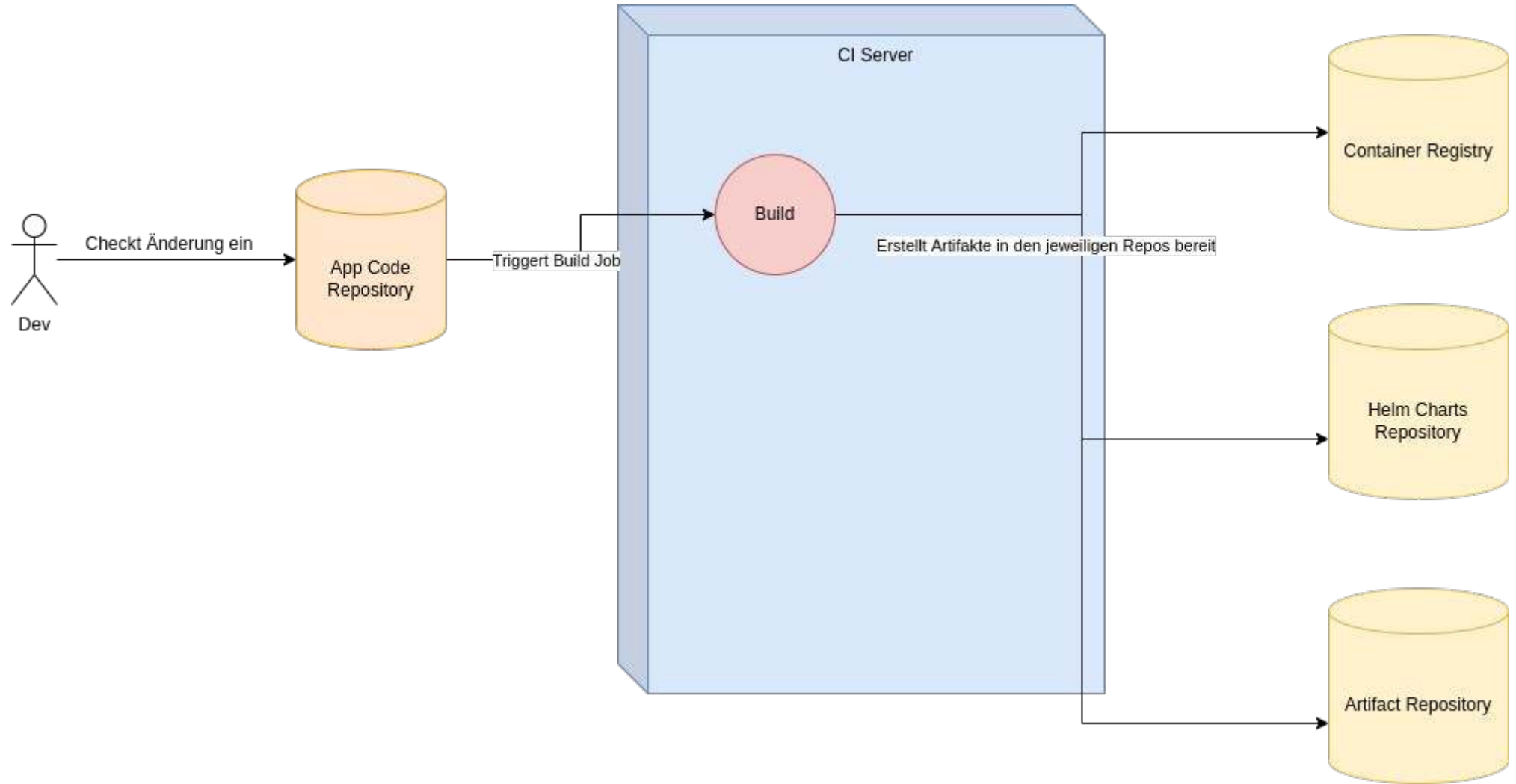
Deployment  
Scripte

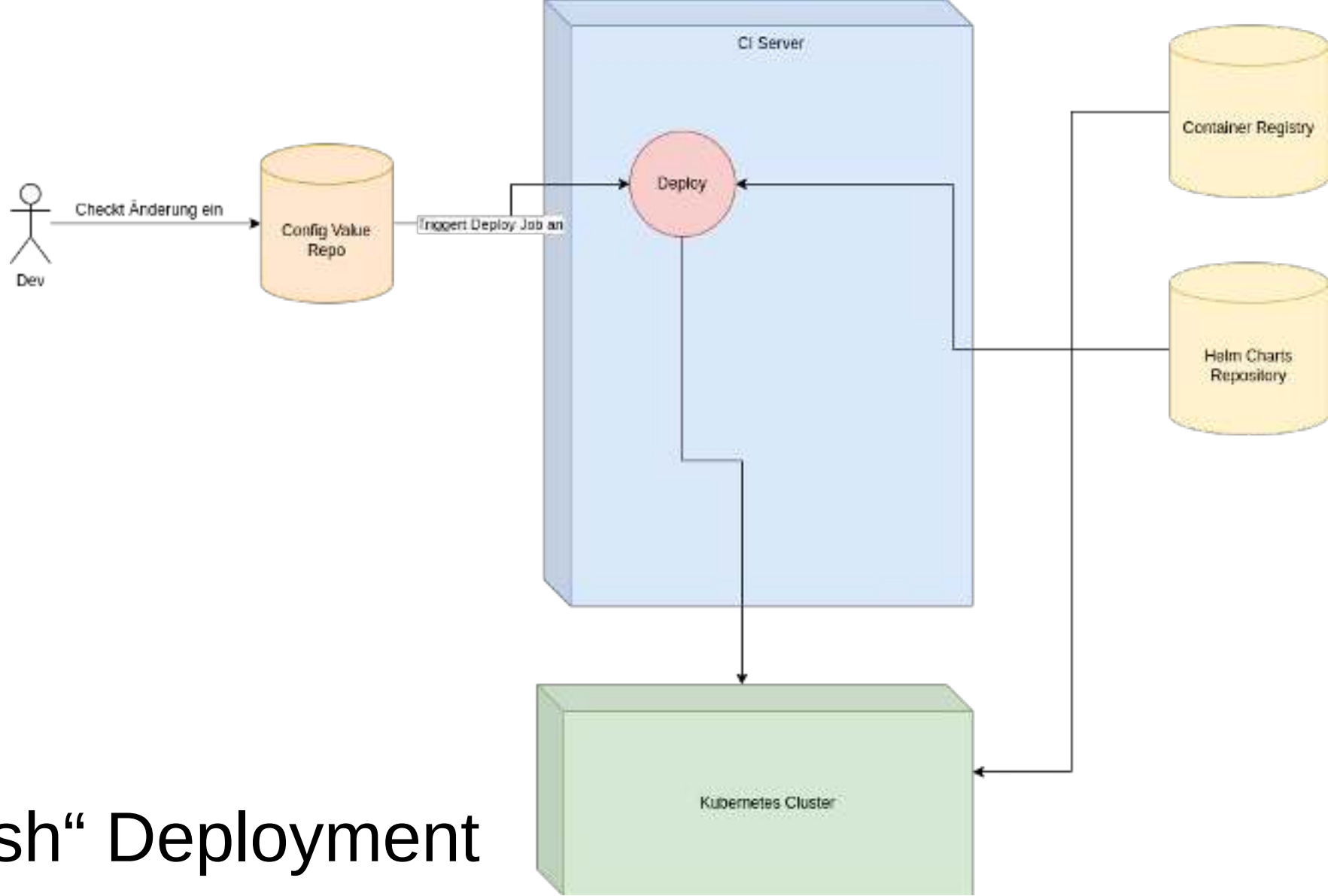
Backend /  
Frontend

Lokale  
Entwicklungs-  
umgebung

12 Factor App:  
Build- und Run-Phase strikt trennen

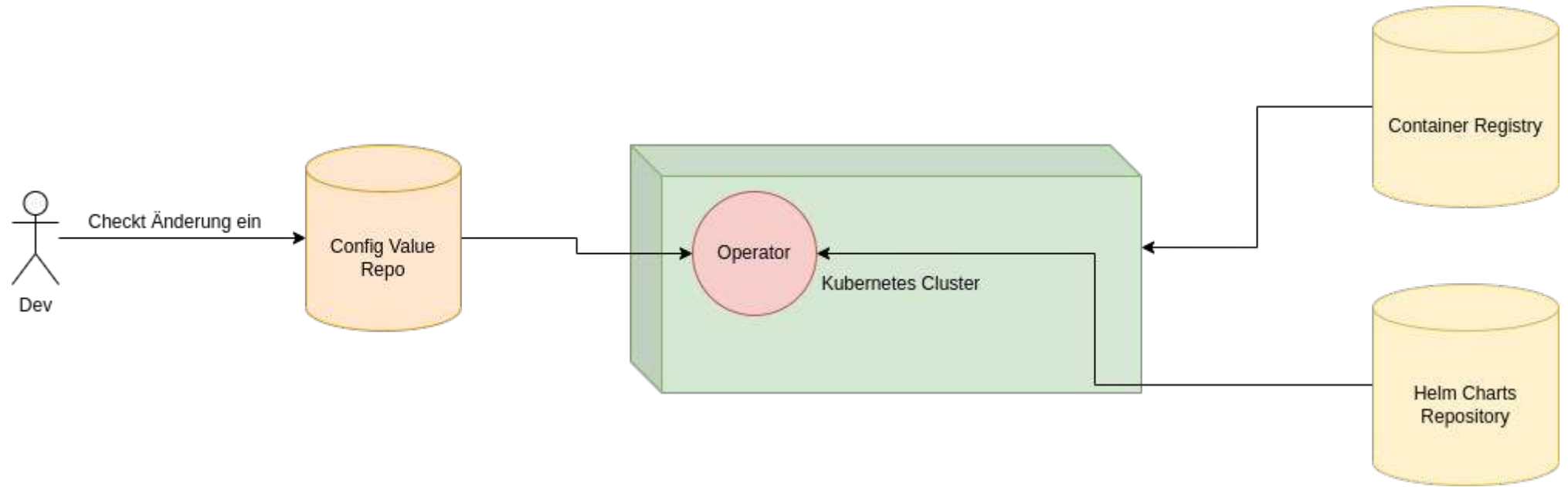
# Build





„Push“ Deployment





# „Pull“ Deployment

App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

Was gehört  
alles ins Git  
Repository  
rein?

Deployment  
Scripte

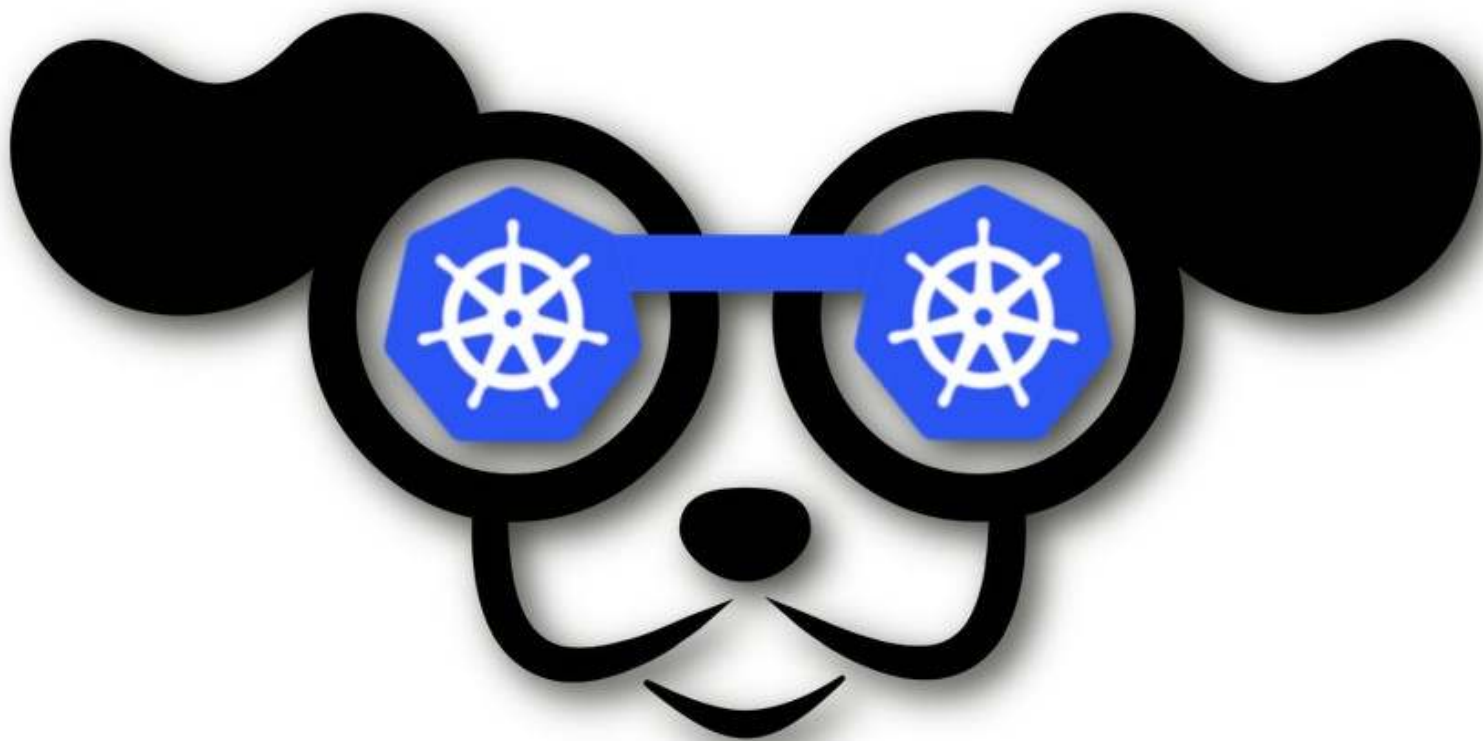
Backend /  
Frontend

Lokale  
Entwicklungs-  
umgebung

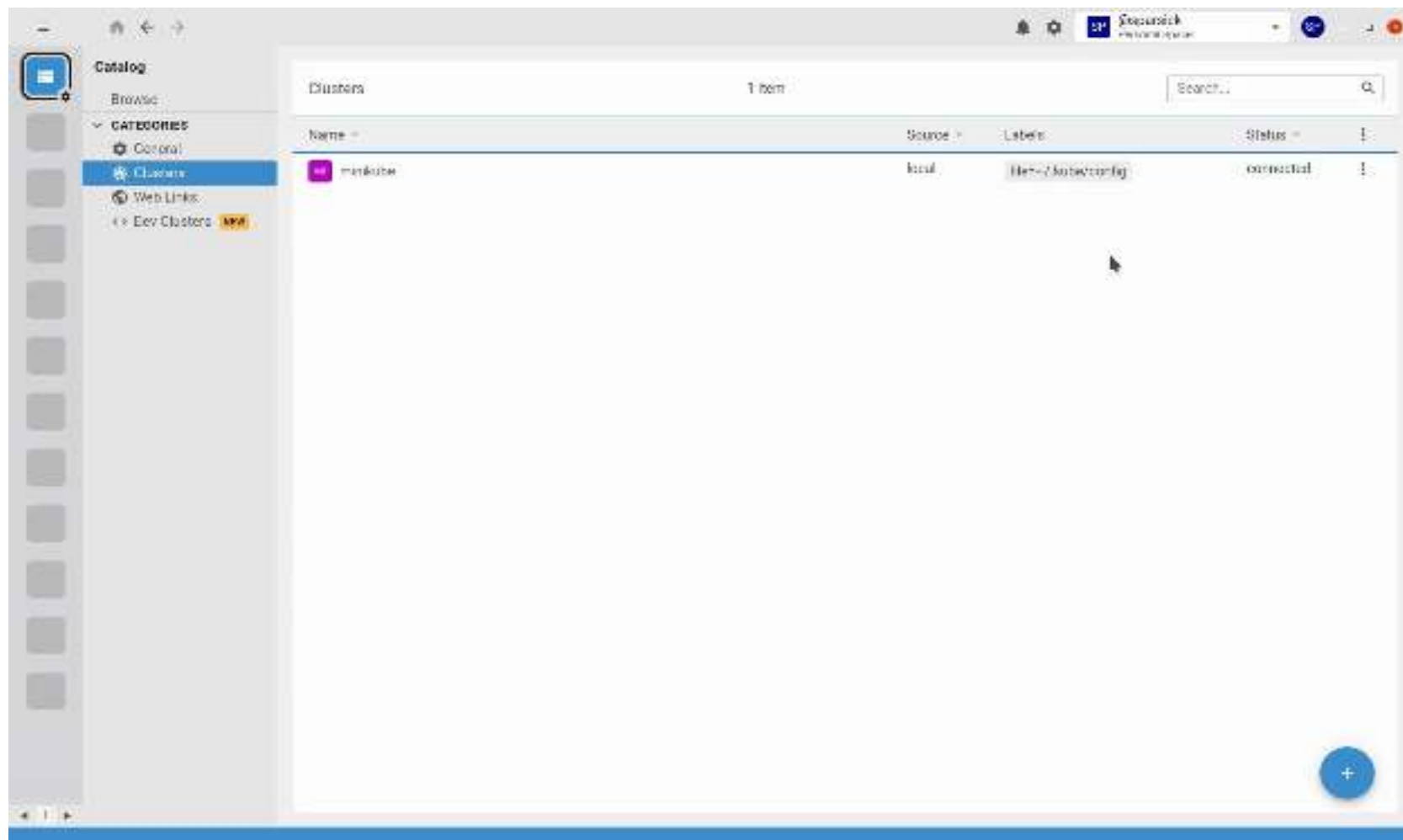
kubectl

# k9s

*Kubernetes CLI To Manage Your Clusters In Style!*



K8s Lens / Open Lens



# Monokle Desktop

App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

Was gehört  
alles ins Git  
Repository  
rein?

Deployment  
Scripte

Backend /  
Frontend

Lokale  
Entwicklungs-  
umgebung





# Troubleshooting Applications

This doc contains a set of resources for fixing issues with containerized applications. It covers things like common issues with Kubernetes resources (like Pods, Services, or StatefulSets), advice on making sense of container termination messages, and ways to debug running containers.

---

[Debug Pods](#)

[Debug Services](#)

[Debug a StatefulSet](#)

[Debug Init Containers](#)

[Debug Running Pods](#)

[Determine the Reason for Pod Failure](#)

[Get a Shell to a Running Container](#)

<https://kubernetes.io/docs/tasks/debug/debug-application/>

# debug container (K8s v1.23)



```
$ kubectl run ephemeral-demo --image=k8s.gcr.io/pause:3.1 --restart=Never
```

```
$ kubectl exec -it ephemeral-demo -- sh
```

```
OCI runtime exec failed: exec failed: container_linux.go:346: starting container process caused "exec: \
```

```
$ kubectl debug -it ephemeral-demo --image=busybox:1.28 --target=ephemeral-demo
```

```
Defaulting debug container name to debugger-8xzrl.
```

```
If you don't see a command prompt, try pressing enter.
```

```
/ #
```

App K8s-  
ready  
machen

Versionierung

Debugging

Wie sehe ich  
was im  
Cluster los  
ist?

Container  
Images

CI

Konfiguration

Was gehört  
alles ins Git  
Repository  
rein?

Deployment  
Scripte

Backend /  
Frontend

Lokale  
Entwicklungs-  
umgebung

Fragen?

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

@SandraParsick

@sparsick@mastodon.social

<https://github.com/sparsick/k8s-dev-survival-kit-talk>

# architektur SPICKER

Handwörterbuch für Architekten und IT-Experten

## Container-Anwendungen entwickeln

Wie man Container-Anwendungen entwickelt und in der Cloud betreibt

**11**

**IN DIESE RUBRIK**

- Container-Anwendungen entwickeln
- Container-Anwendungen betreiben
- Container-Anwendungen integrieren
- Container-Anwendungen sichern
- Container-Anwendungen überwachen

**Was ist neu?**

- Container-Anwendungen entwickeln
- Container-Anwendungen betreiben
- Container-Anwendungen integrieren
- Container-Anwendungen sichern
- Container-Anwendungen überwachen

## Anwendungen über Container und Container-Orchestrierung

Wie man Container-Anwendungen entwickelt und in der Cloud betreibt

**1** Container-Anwendungen entwickeln

**2** Container-Anwendungen betreiben

**3** Container-Anwendungen integrieren

**4** Container-Anwendungen sichern

**5** Container-Anwendungen überwachen

**6** Container-Anwendungen integrieren

**7** Container-Anwendungen sichern

**8** Container-Anwendungen überwachen

**9** Container-Anwendungen integrieren

**10** Container-Anwendungen sichern

**11** Container-Anwendungen überwachen

# architektur SPICKER

Handwörterbuch für Architekten und IT-Experten

## Continuous Delivery

Wie man Continuous Delivery implementiert und in der Cloud betreibt

**7**

**IN DIESE RUBRIK**

- Continuous Delivery implementieren
- Continuous Delivery betreiben
- Continuous Delivery integrieren
- Continuous Delivery sichern
- Continuous Delivery überwachen

**Was ist neu?**

- Continuous Delivery implementieren
- Continuous Delivery betreiben
- Continuous Delivery integrieren
- Continuous Delivery sichern
- Continuous Delivery überwachen

**Wie geht es?** (Kontext: Continuous Delivery)

- Continuous Delivery implementieren
- Continuous Delivery betreiben
- Continuous Delivery integrieren
- Continuous Delivery sichern
- Continuous Delivery überwachen

**Die Continuous Delivery Pipeline**

CD ist ein Prozess zur Integration und Verifizierung von Code, der von der Entwicklung bis zur Produktion führt. Er ist ein Prozess, der die Integration und Verifizierung von Code, der von der Entwicklung bis zur Produktion führt. Er ist ein Prozess, der die Integration und Verifizierung von Code, der von der Entwicklung bis zur Produktion führt.

© 2018 embarc

1



# Weitere gute Vorträge zum Thema

- Vortrag „Wenn ich das nur vorher gewusst hätte: Kubernetes für Entwickler“ von Stefan Schlott
- Vortrag „Kubernetes-Lektionen aus der Wolke“ von Jochen Mader
- Vortrag „What's going on in my cluster?“ von Matthias Häussler

# 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/>



# Bildnachweise

- [https://unsplash.com/photos/RfwGg5ZZh4Q?  
utm\\_source=unsplash&utm\\_medium=referral&u  
tm\\_content=creditShareLink](https://unsplash.com/photos/RfwGg5ZZh4Q?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink)
- [https://unsplash.com/photos/CpsTAUPoScw?  
utm\\_source=unsplash&utm\\_medium=referral&u  
tm\\_content=creditShareLink](https://unsplash.com/photos/CpsTAUPoScw?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink)