Continuous Lifecycle, 13.11.2024

Kubernetes Developer Survival Kit

Sandra Parsick
@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

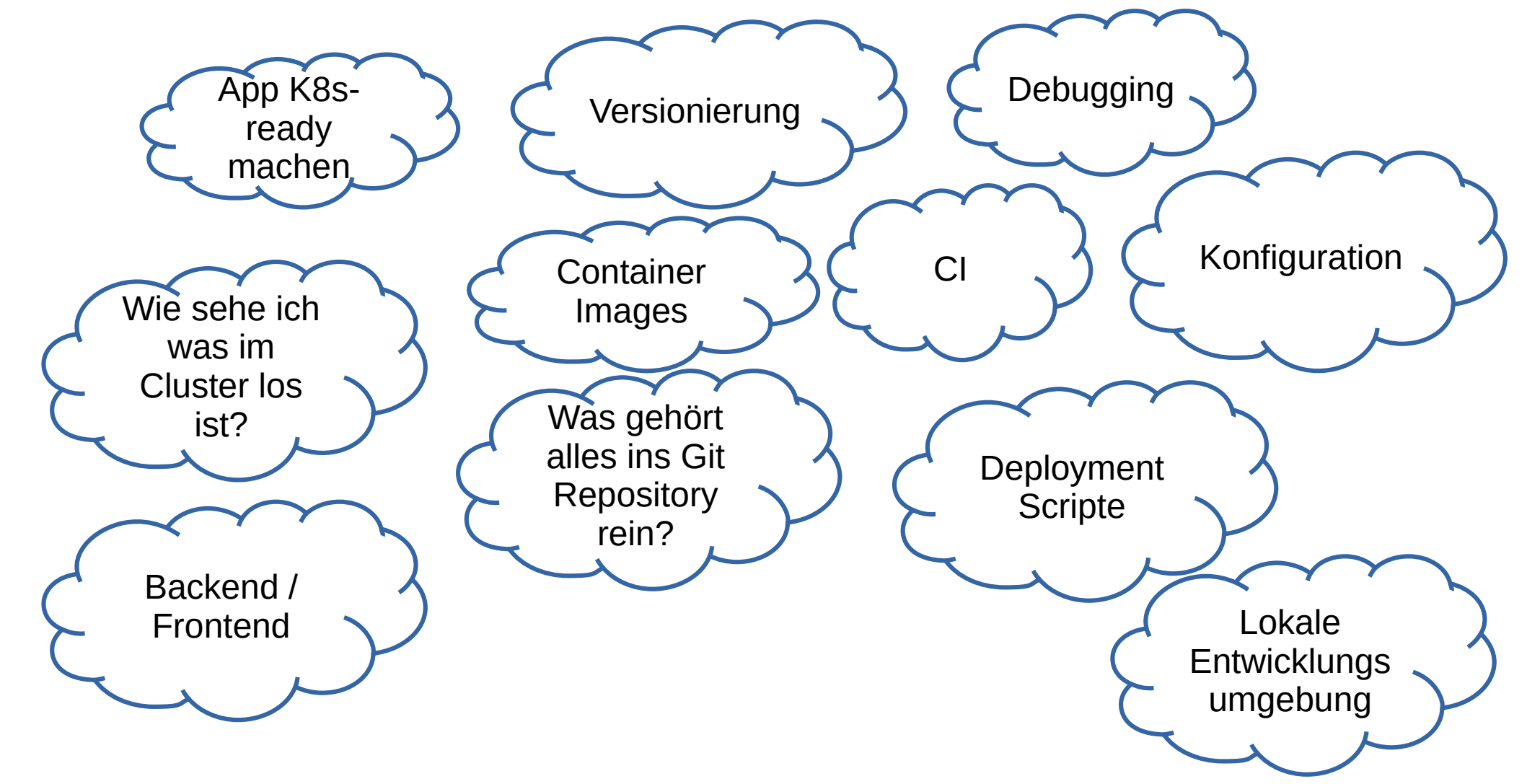
mail@sandra-parsick.de

@sparsick@mastodon.social

https://www.sandra-parsick.de

https://ready-for-review.dev


SSA TERMINALS
SSATerminals
NORTHERN JUSTICE
MADEIRA


App K8s-ready machen
Versionierung
Debugging
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
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

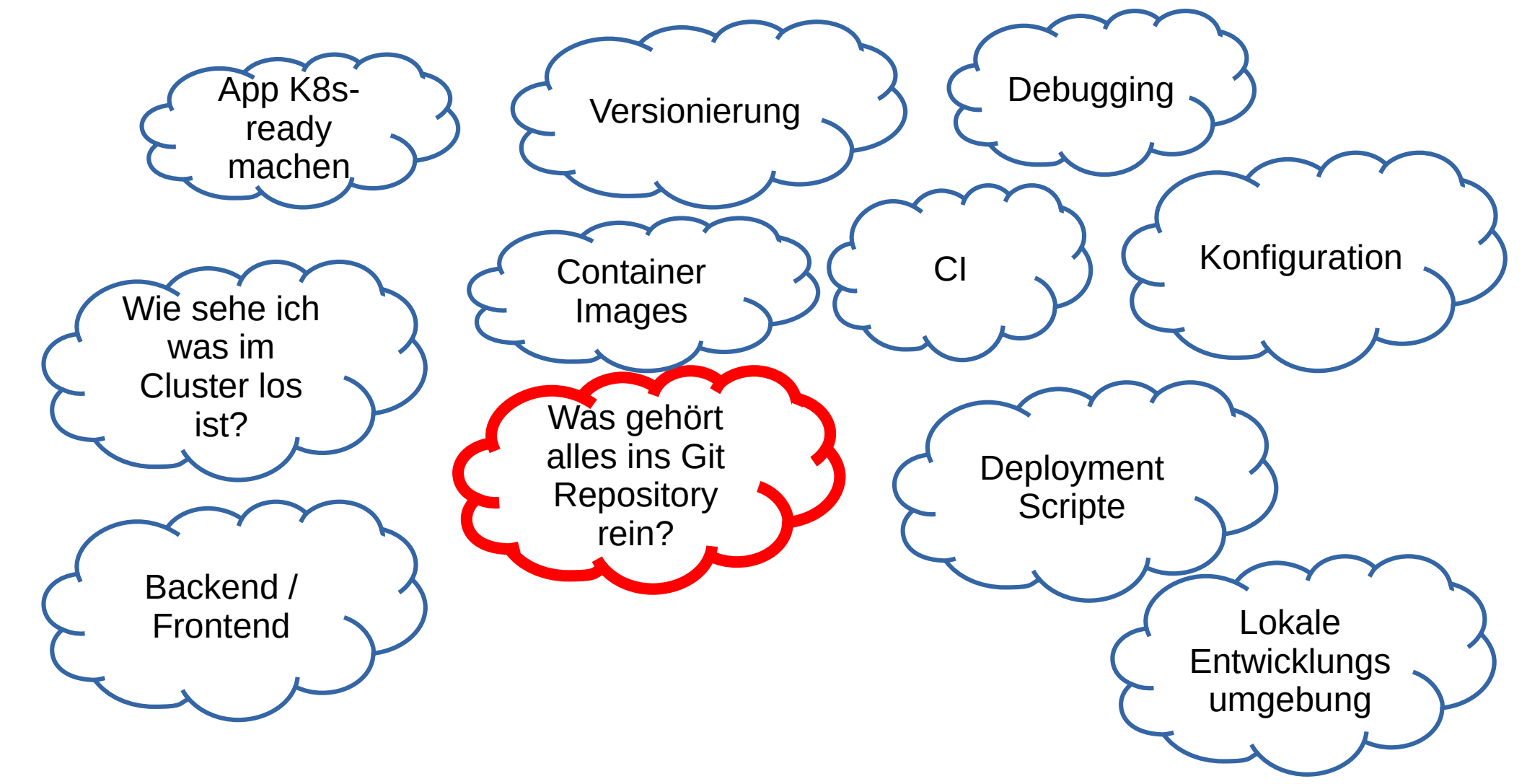
VIII. Concurrency
Scale out via the process model

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

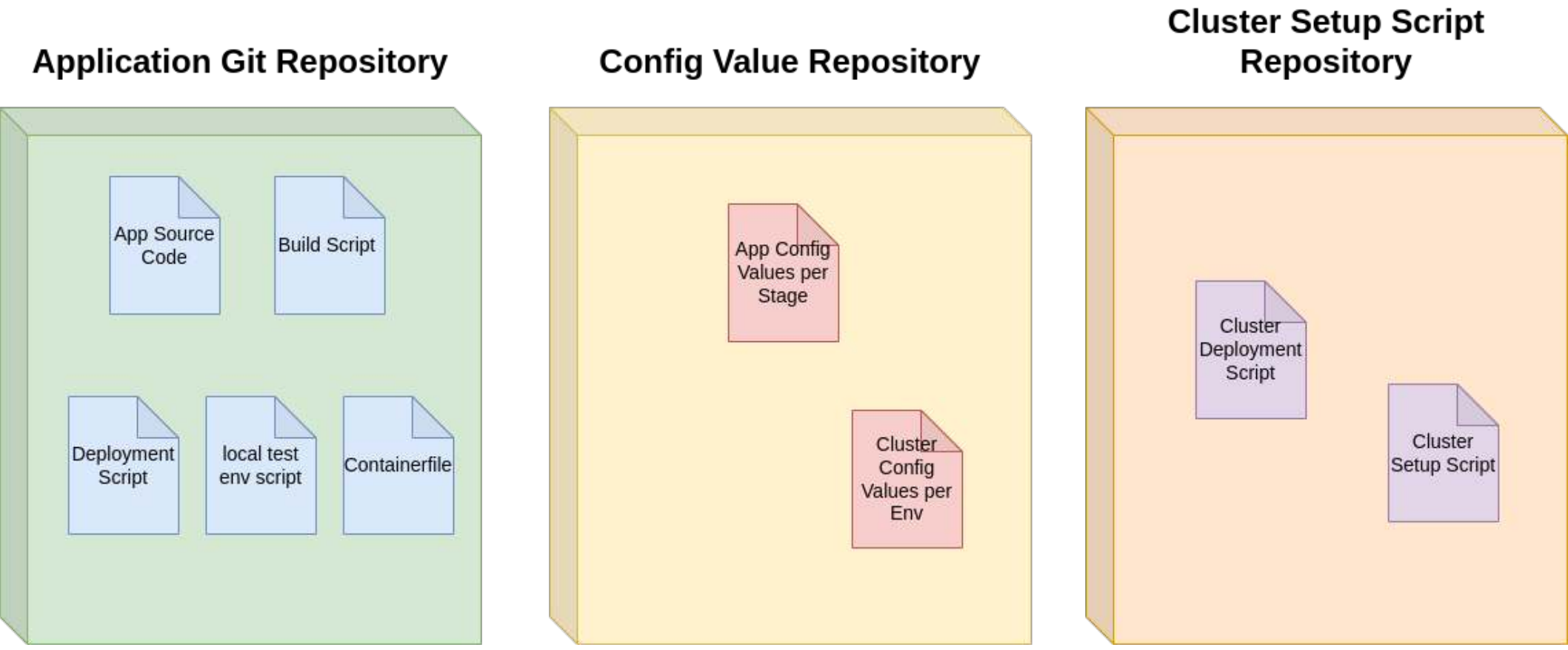
XII. Admin processes
Run admin/management tasks as one-off processes


App K8s-ready machen
Versionierung
Debugging
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
Was gehört alles ins Git Repository rein?
Deployment Scripte
Backend / Frontend
Lokale Entwicklungs umgebung

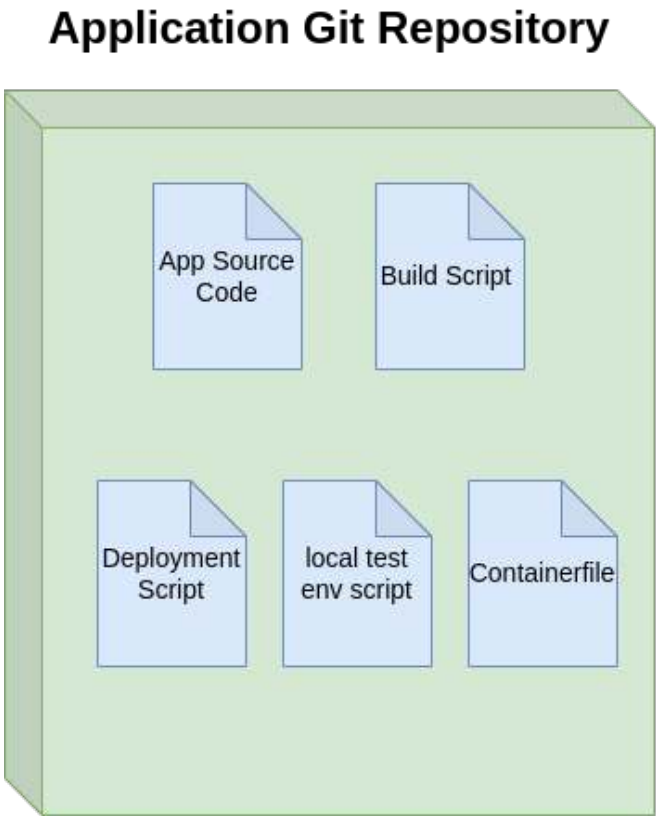
Kurzform: ALLES

Eigentliche Fragestellung:
Wieviele Repositories?

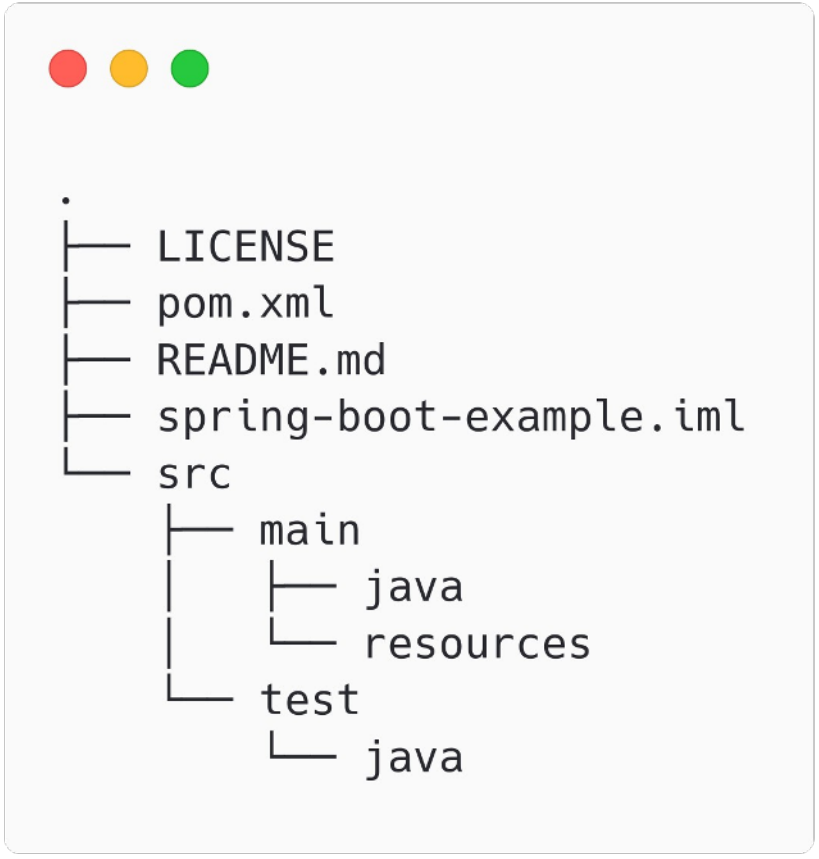
Beispiel für eine Aufteilung



Für Devs am wichtigsten

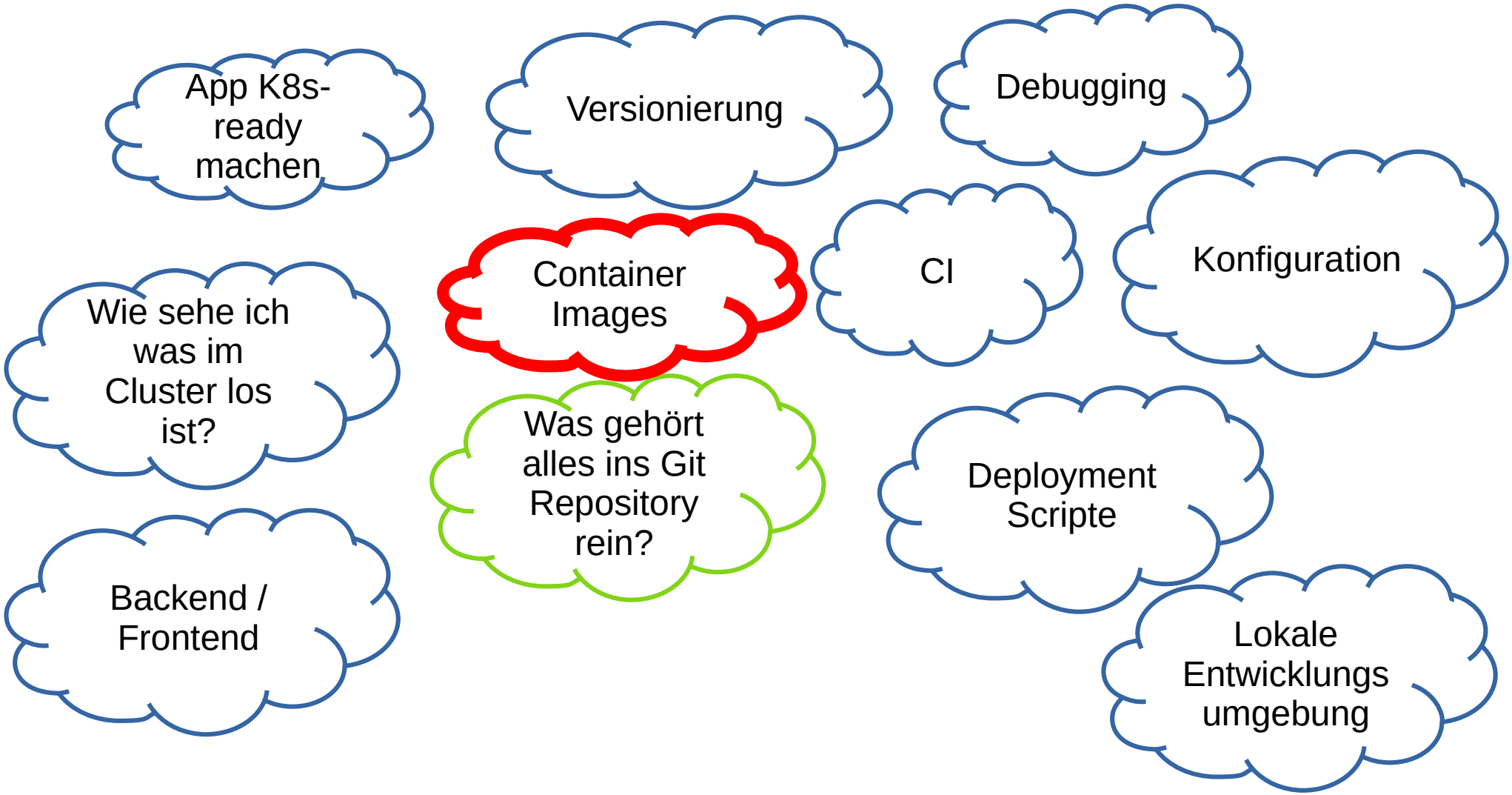


Ausgangspunkt eine Java App

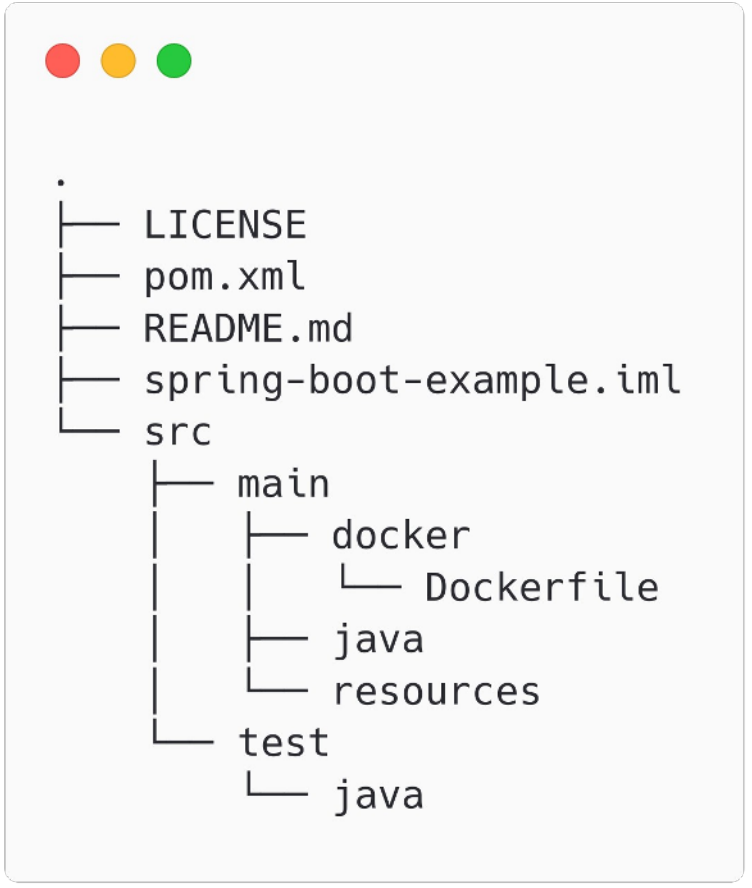


Technologiestack:

- Java 21
- Spring Boot 3.3.x
- Thymeleaf
- Apache Maven


App K8s-ready machen
Versionierung
Debugging
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
Was gehört alles ins Git Repository rein?
Deployment Scripte
Backend / Frontend
Lokale Entwicklungs umgebung

Basis: Container



Basis: Container

```
FROM eclipse-temurin:21.0.5_11-jre as builder
WORKDIR /application
COPY maven/*.jar application.jar
RUN java -Djarmode=layertools -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.launch.JarLauncher"]
```

```xml
<plugin>
    <groupId>io.fabric8</groupId>
    <artifactId>docker-maven-plugin</artifactId>
    <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 eclipse-temurin:21.0.5_11-jre as builder
WORKDIR /application
COPY maven/*.jar application.jar
RUN java -Djarmode=layertools -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.launch.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)

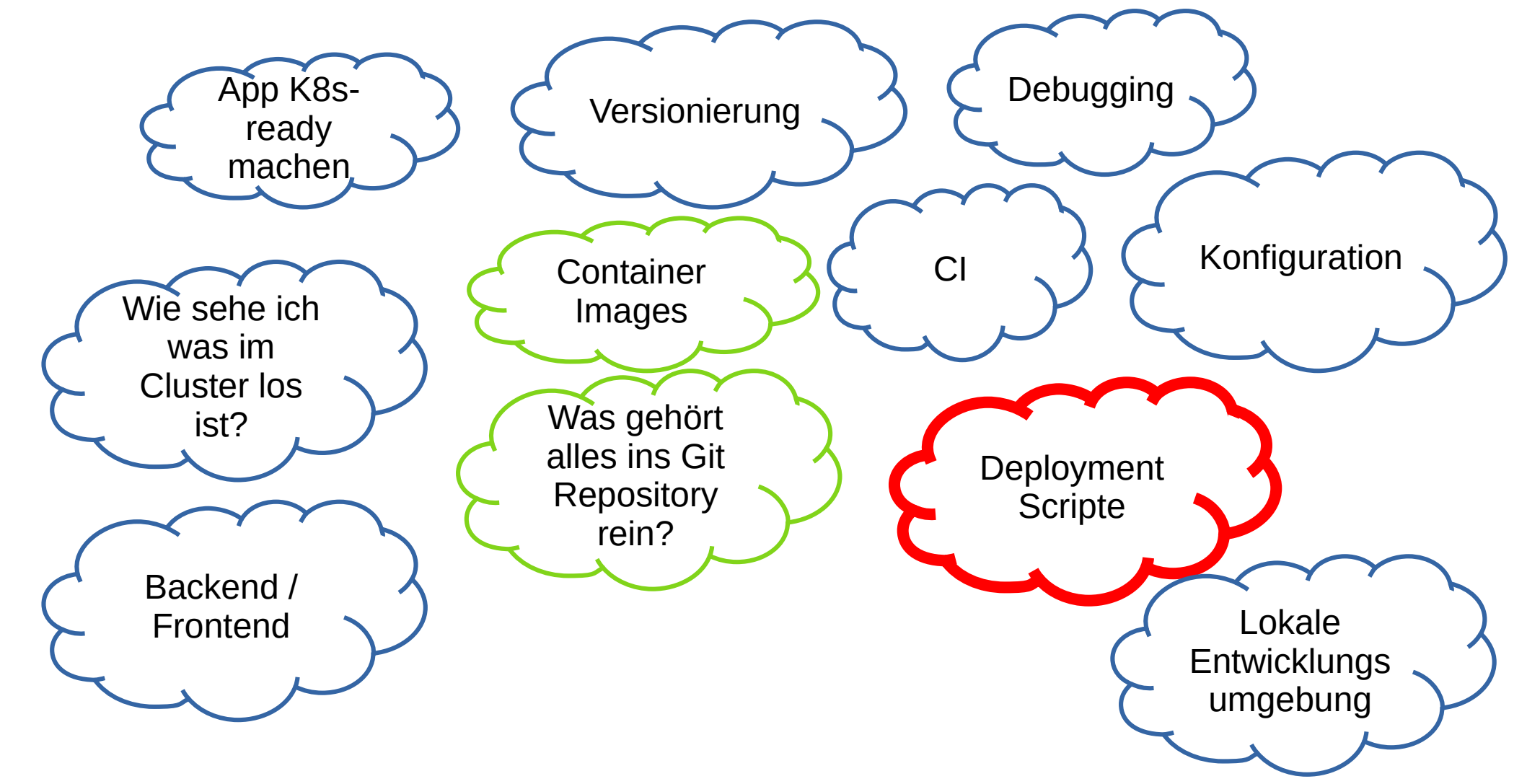
```
→ trivy i --ignore-unfixed -o result  spring-boot-demo:latest
2022-06-23T09:55:56.244+0200    INFO    Vulnerability scanning is enabled
2022-06-23T09:55:56.245+0200    INFO    Secret scanning is enabled
2022-06-23T09:55:56.245+0200    INFO    If your scanning is slow, please try '--security-checks 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)

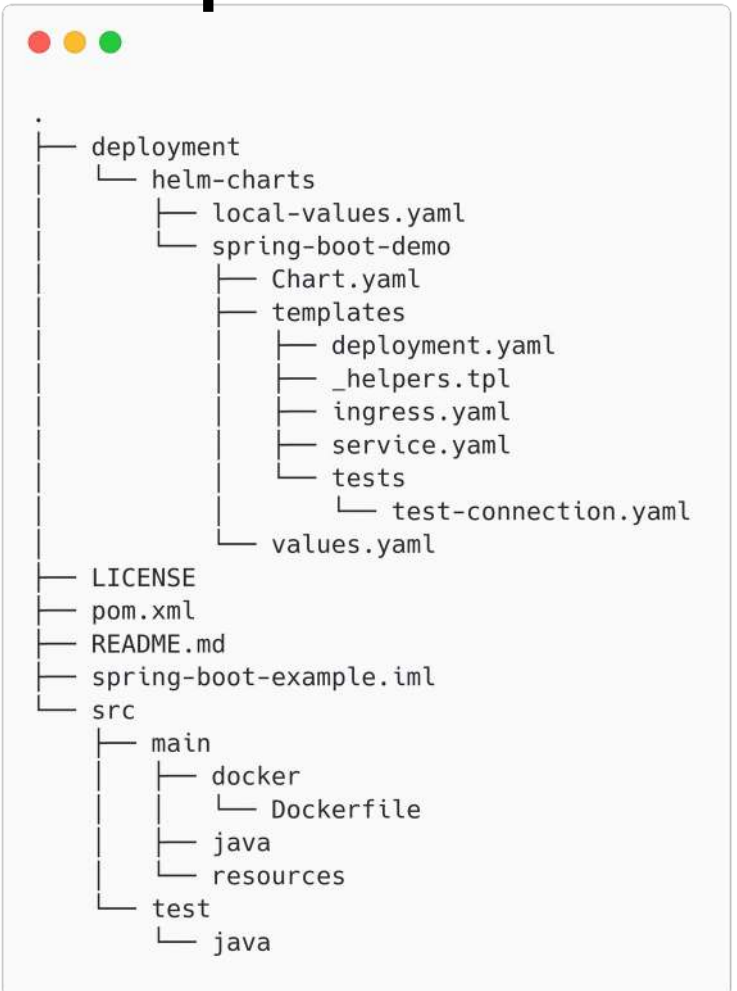
Library	Vulnerability	Severity	Installed Version	Fixed Version	Title
libc6	CVE-2021-33574	CRITICAL	2.31-13+deb11u2	2.31-13+deb11u3	glibc: mq_notify does not handle separately allocated thread attributes https://avd.aquasec.com/nvd/cve-2021-33574
	CVE-2022-23218				glibc: Stack-based buffer overflow in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

Vulnerability-Scans Weitergedacht

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


App K8s-ready machen
Versionierung
Debugging
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
Was gehört alles ins Git Repository rein?
Deployment Scripte
Backend / Frontend
Lokale Entwicklungs umgebung

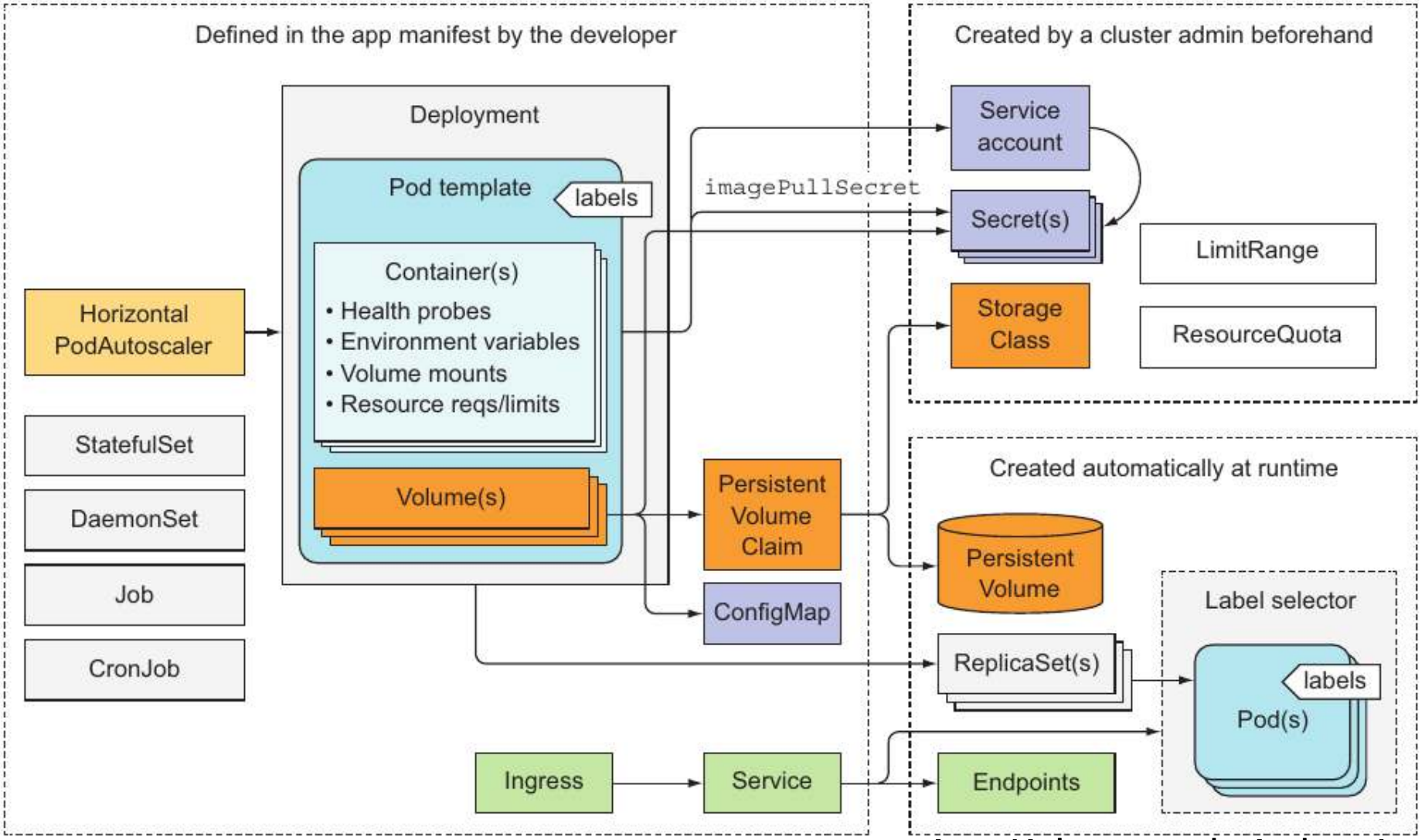
Next Step: Helm Charts



Auszug: Service Definition

```yaml
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?



Aus: Kubernetes in Action, 1st Edition

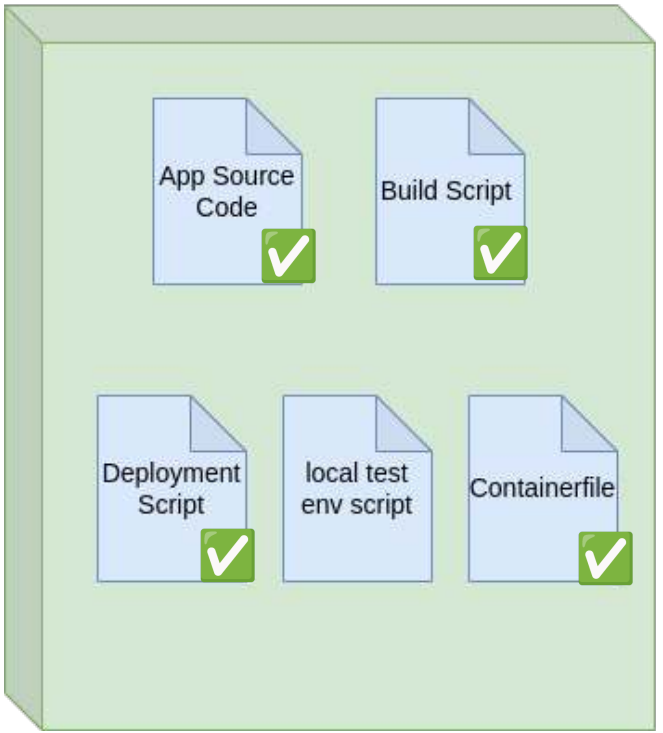
```
            <plugin>
                <groupId>io.kokuwa.maven</groupId>
                <artifactId>helm-maven-plugin</artifactId>
                <version>6.16.0</version>
                <configuration>
                    <chartDirectory>${project.basedir}/deployment/helm-charts</chartDirectory>
                    <chartVersion>${project.version}</chartVersion>
                    <helmVersion>3.26.2</helmVersion>
                    <!-- This is the related section to use local-env binary with auto-detection
enabled. -->
                    <useLocalHelmBinary>true</useLocalHelmBinary>
                    <!-- This is the related section to configure upload repos -->
                    <uploadRepoStable>
                        <name>helm-release</name>
                        <url>http://localhost:8002/repository/helm-release/</url>
                        <type>NEXUS</type>
                    </uploadRepoStable>
                    <uploadRepoSnapshot>
                        <name>helm-snapshot</name>
                        <url>http://localhost:8002/repository/helm-snapshot/</url>
                        <type>NEXUS</type>
                    </uploadRepoSnapshot>
                </configuration>
                <executions>
                    <execution>
                        <id>build-chart</id>
                        <phase>install</phase>
                        <goals>
                            <goal>package</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>upload-chart</id>
                        <phase>deploy</phase>
                        <goals>
                            <goal>upload</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
```

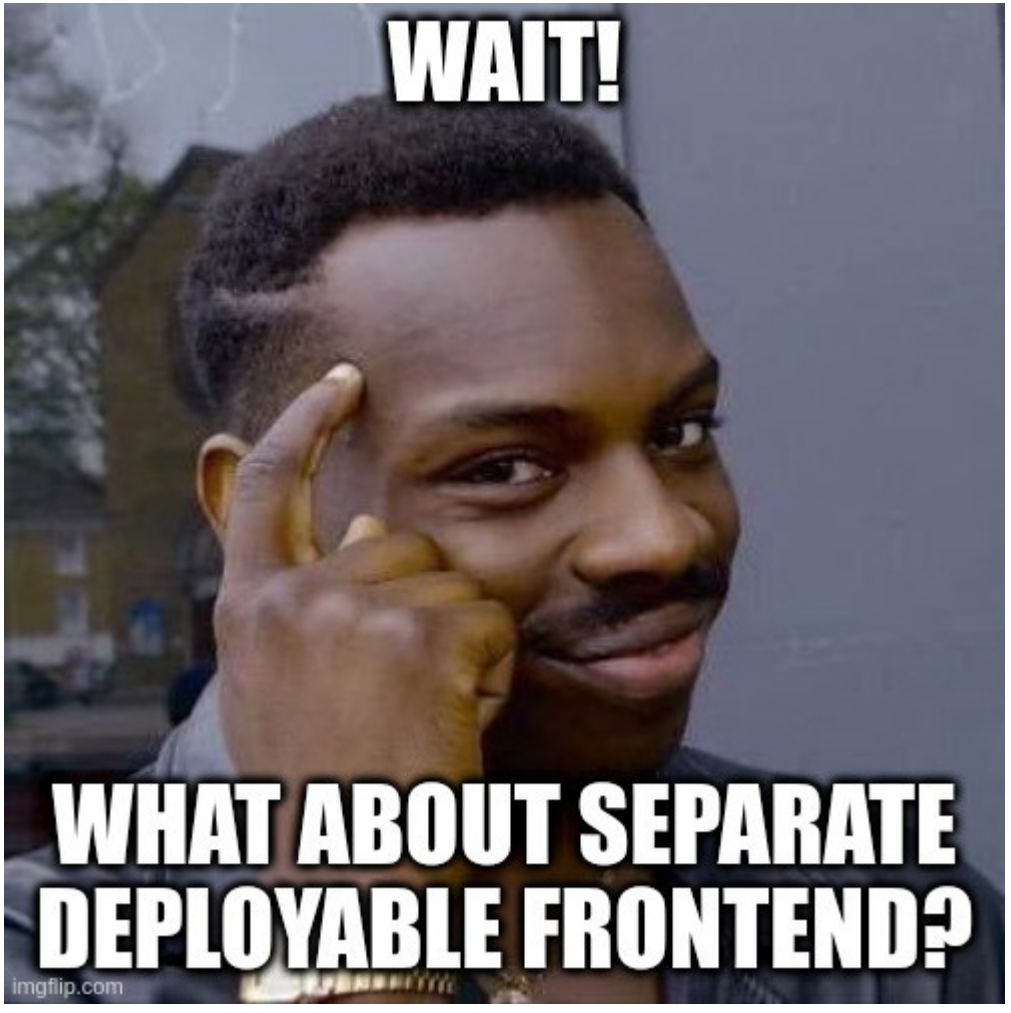
Helm Charts Paketierung Teil des Build Prozesses

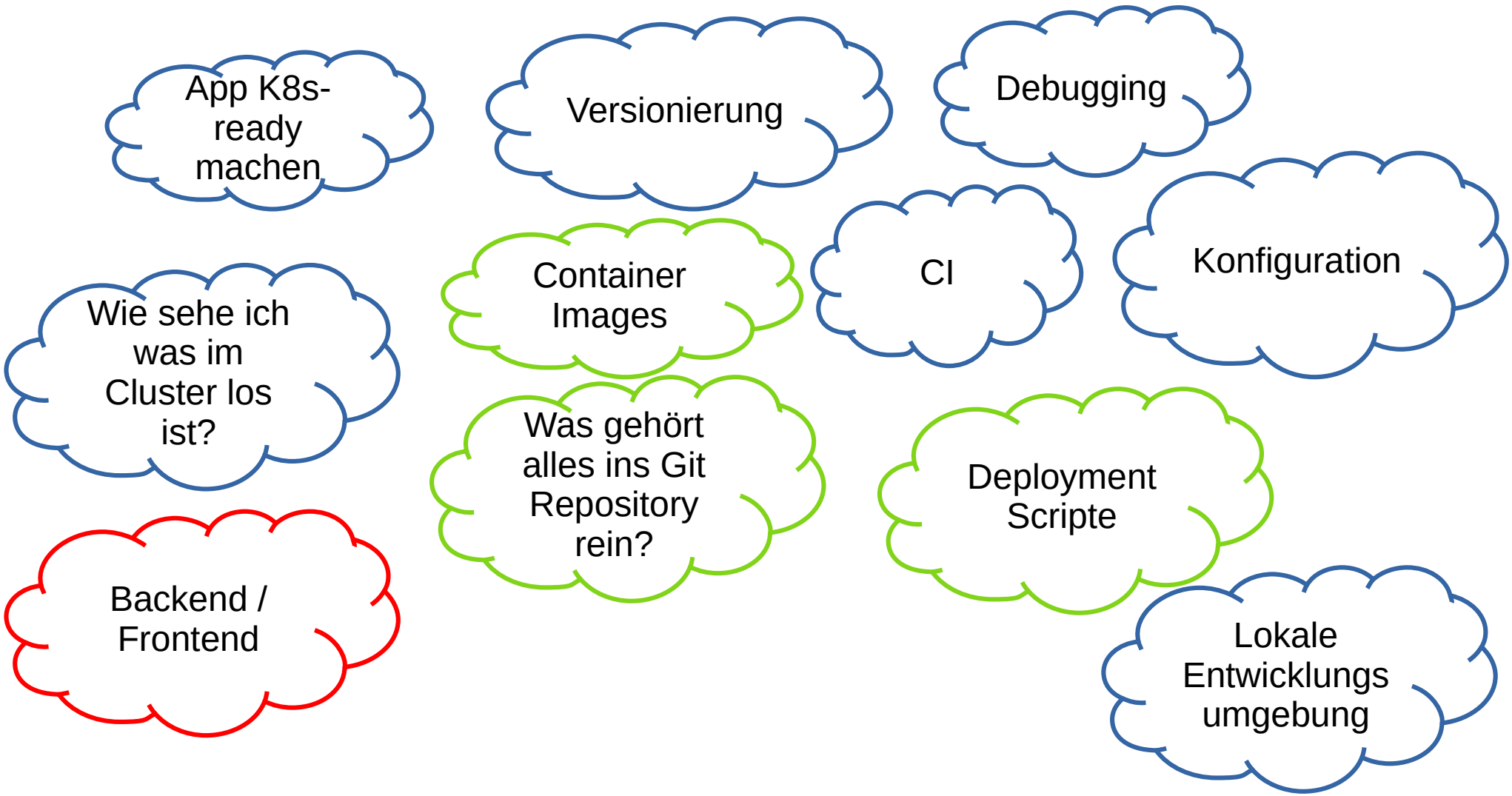
Helm Chart Repository

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

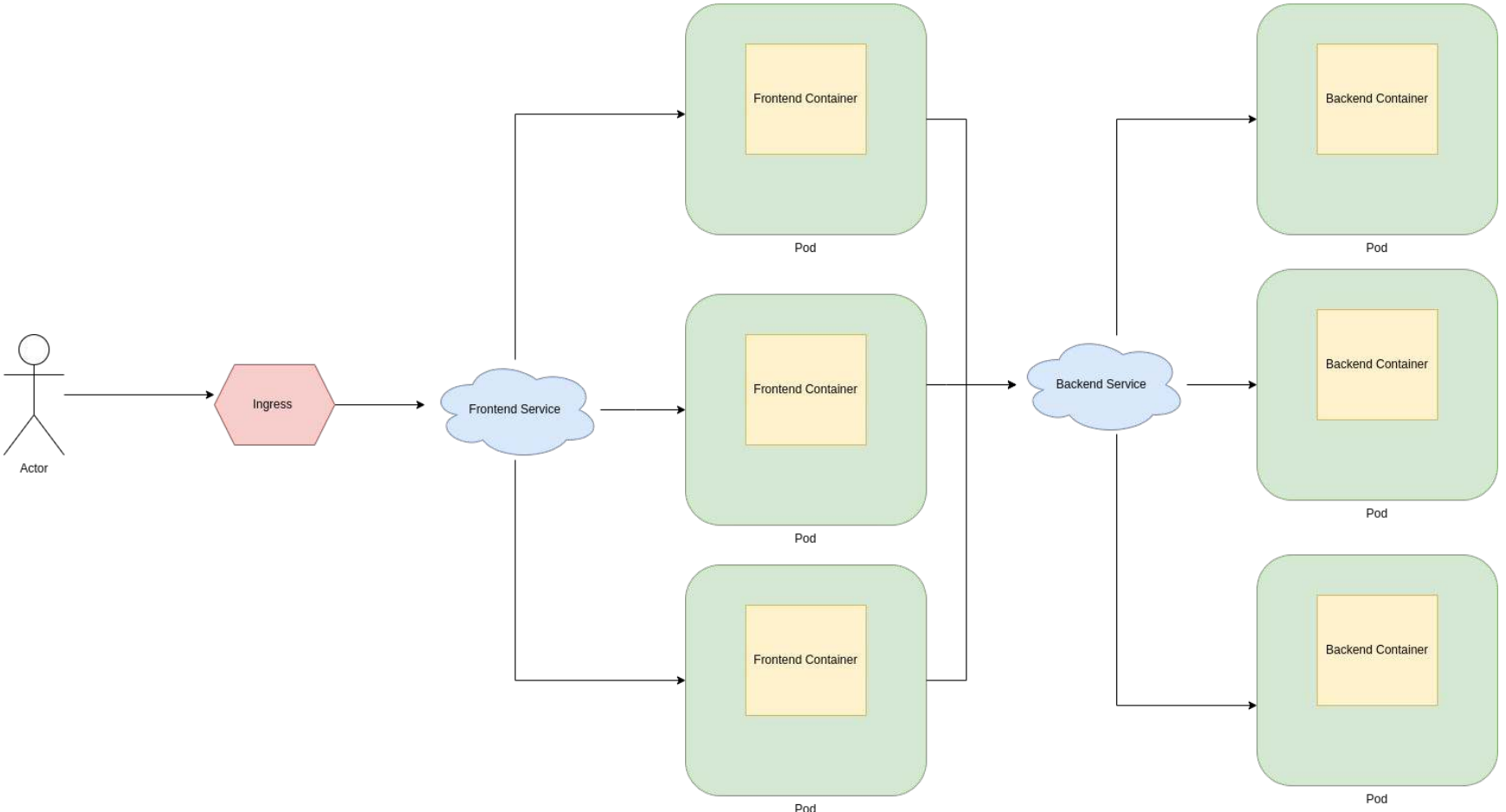
Application Git Repository



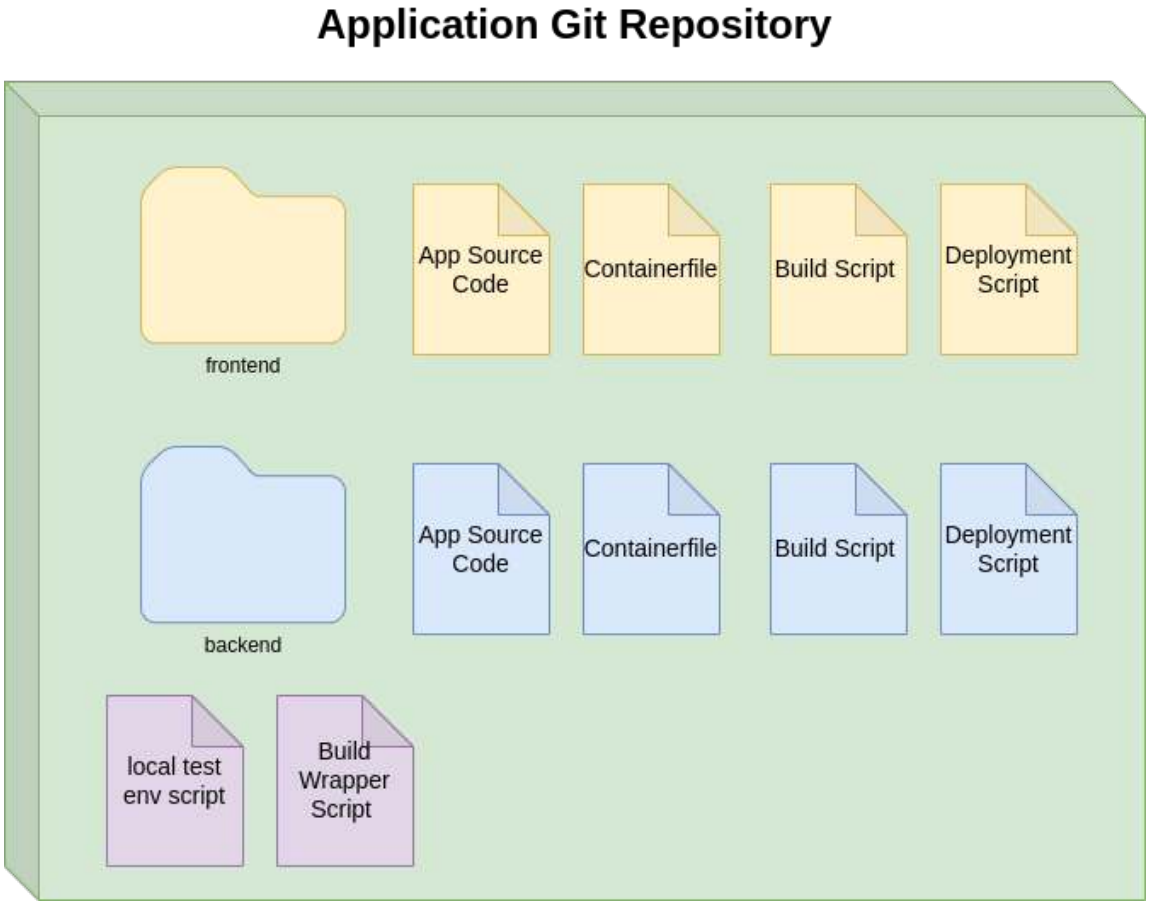

WAIT!
WHAT ABOUT SEPARATE DEPLOYABLE FRONTEND?
imgflip.com


App K8s-ready machen
Versionierung
Debugging
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
Was gehört alles ins Git Repository rein?
Deployment Scripte
Backend / Frontend
Lokale Entwicklungs umgebung

Frontend und Backend in K8s



Git Repository Struktur



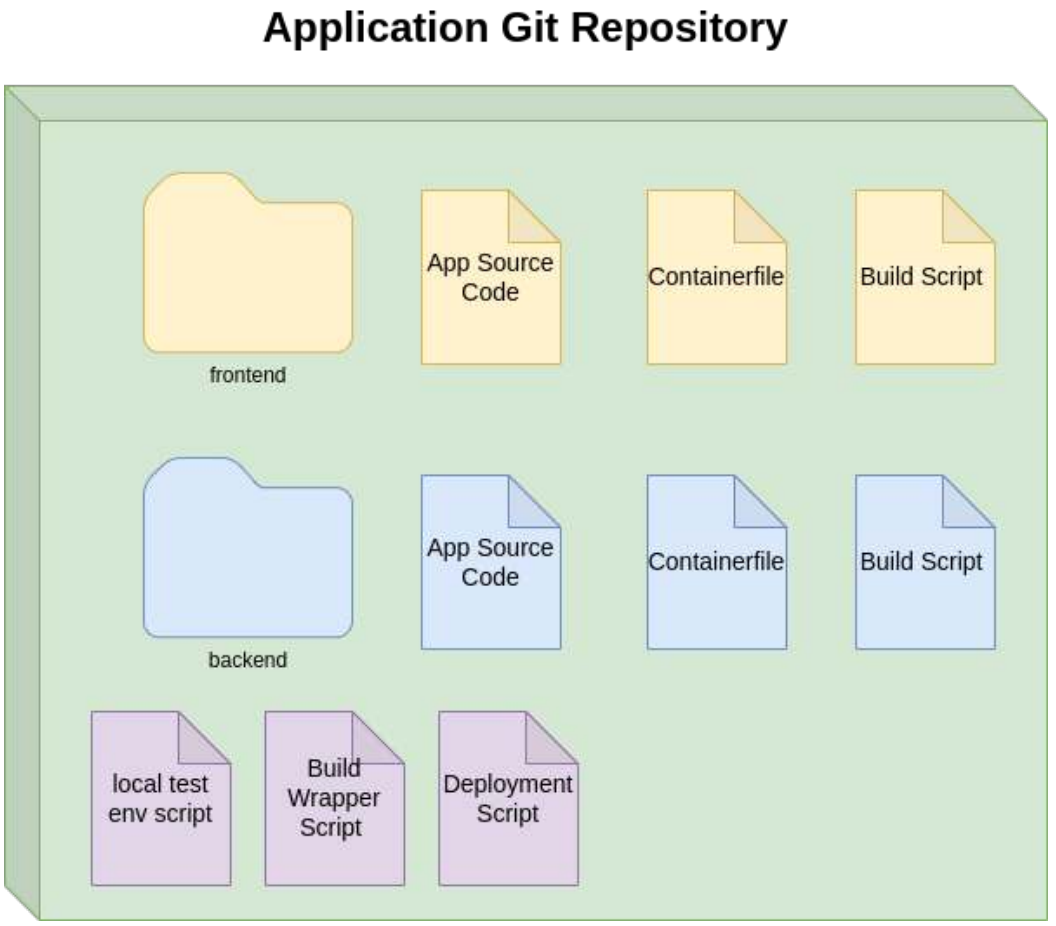
Deployment Script

- Ingress
- Frontend Service
- Frontend Deployment

Deployment Script

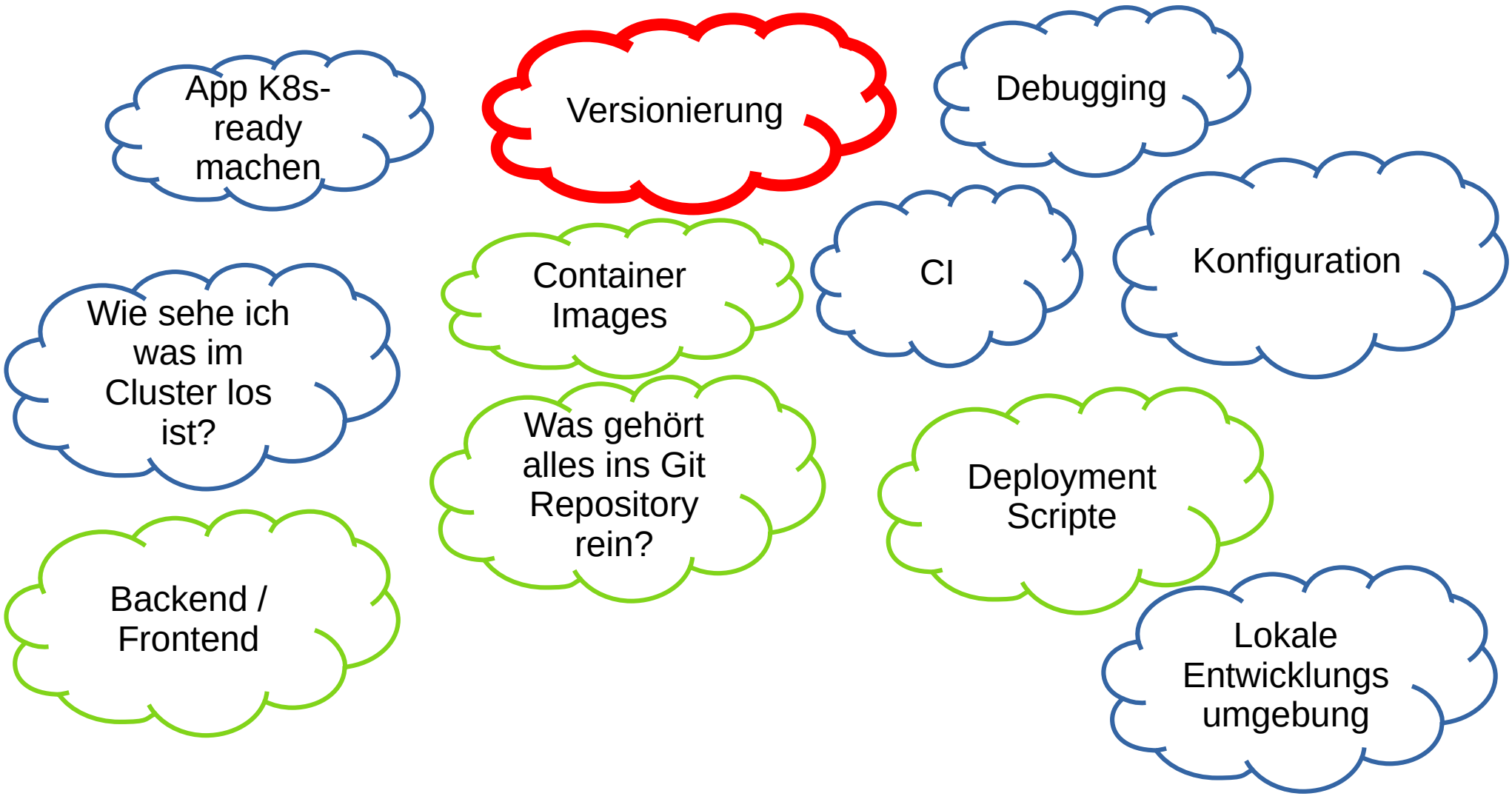
- Backend Service
- Backend Deployment

Git Repository Struktur



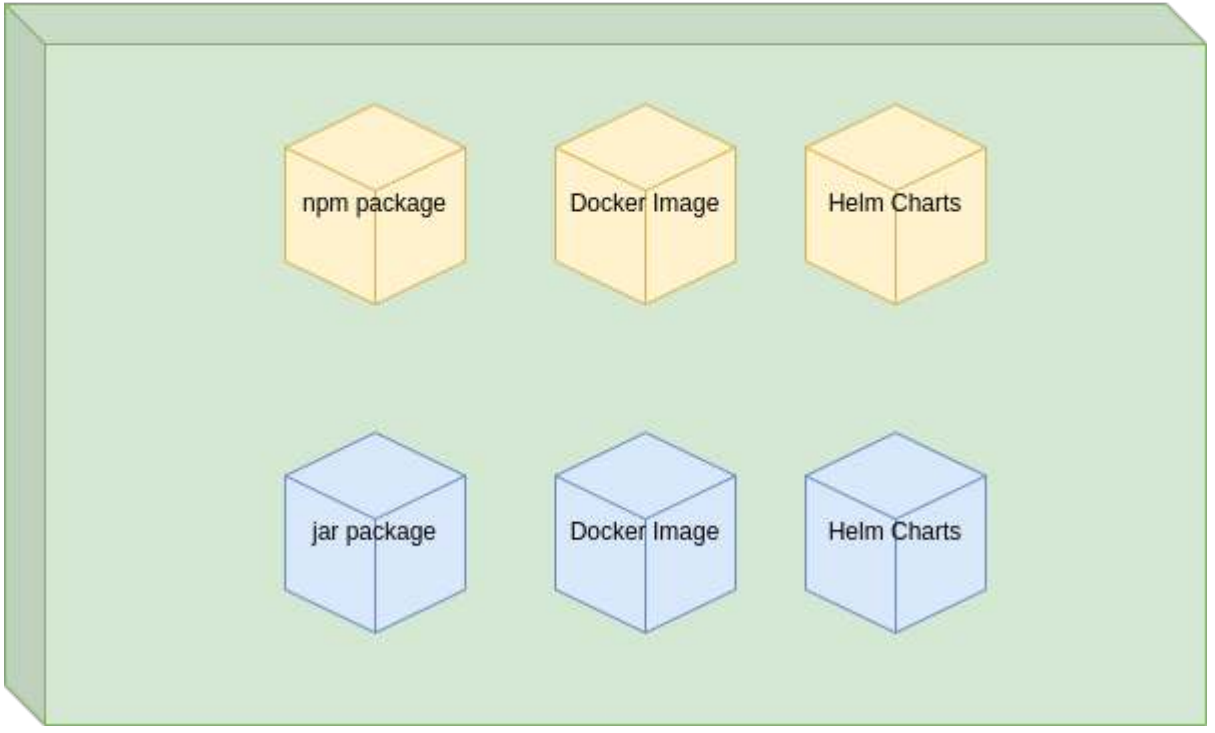
Deployment Script

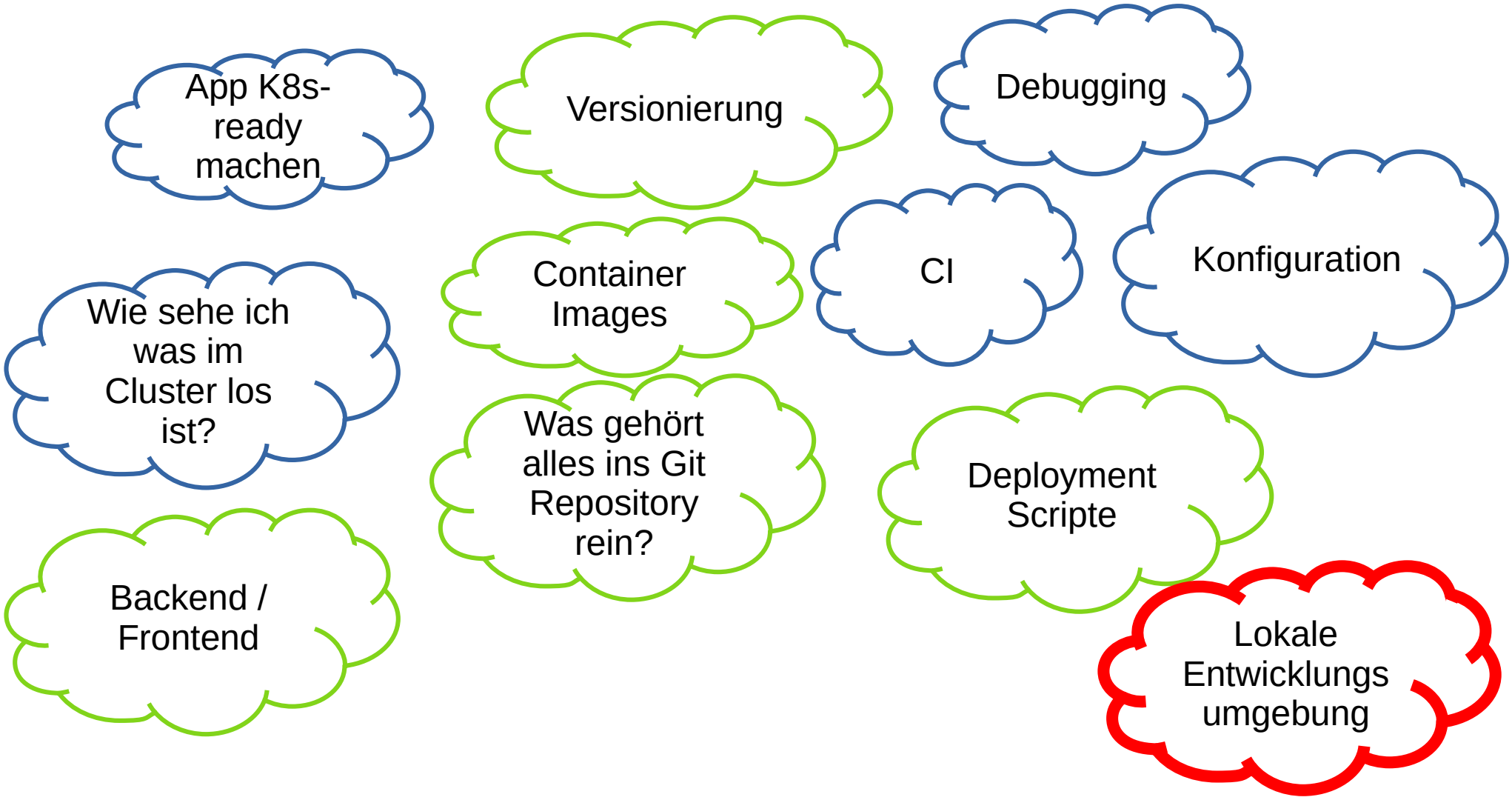
- Ingress
- Frontend Service
- Frontend Deployment
- Backend Service
- Backend Deployment


App K8s-ready machen
Versionierung
Debugging
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
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
CI
Konfiguration
Container Images
Wie sehe ich was im Cluster los ist?
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

```yaml
services:
  database:
    image: mongo:8.0.3
    restart: always
    ports:
      - 27017:27017
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: root123
    volumes:
      - ./local-env/:/docker-entrypoint-initdb.d/
```

```xml
        <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>mongodb://test:test123@localhost/test</MONGODB_URI>
                </environmentVariables>
            </configuration>
        </plugin>
```

Demo

```yaml
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

```xml
<!-- 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>
```

Demo

Andere
Abhängigkeiten?

Mocking

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

```yaml
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
```

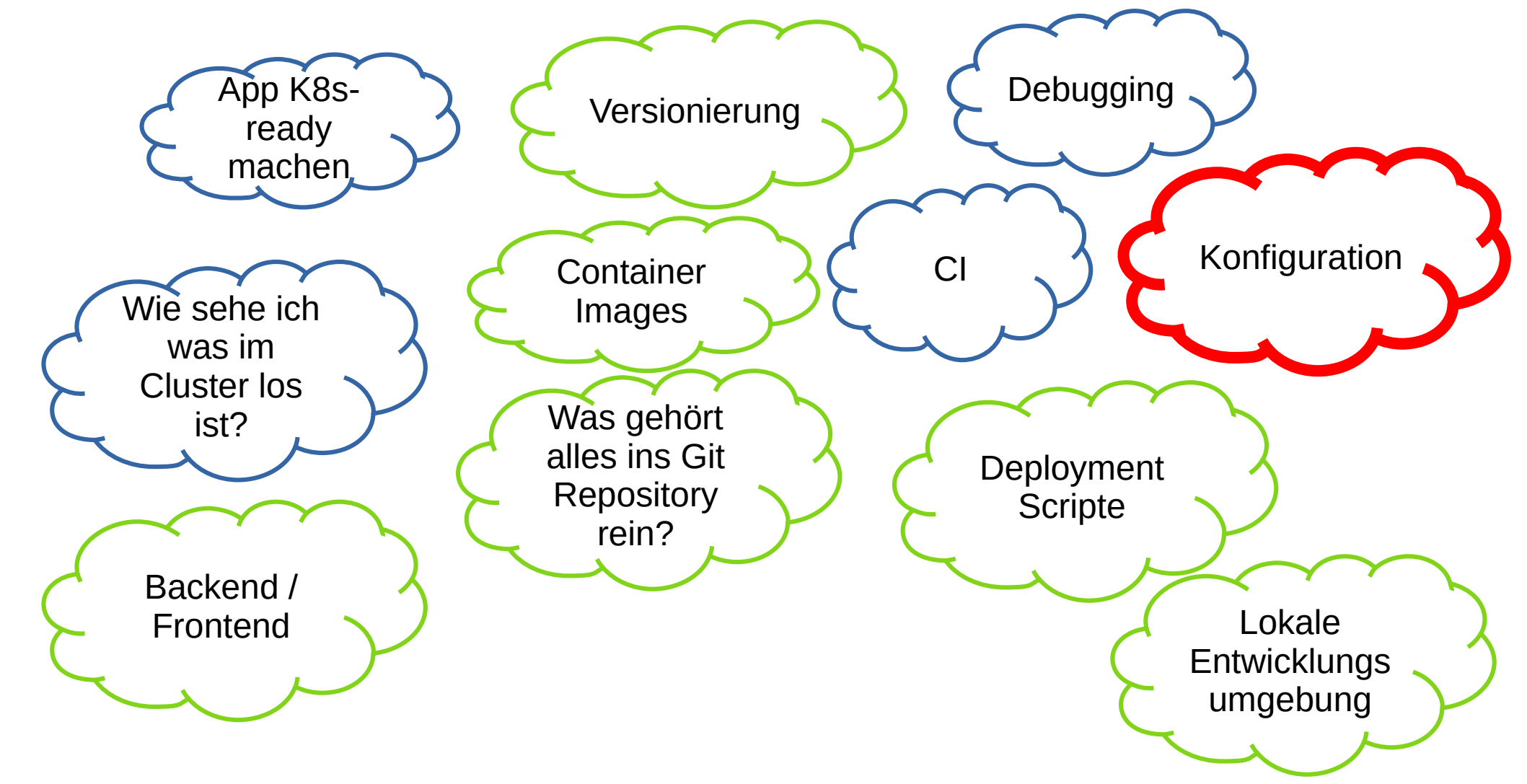
```
[
  {
    "httpRequest": {
      "path": "/success"
    },
    "httpResponse": {
      "body": "Successful!"
    }
  },
  {
    "httpRequest": {
      "path": "/fail"
    },
    "httpResponse": {
      "statusCode": 400
    }
  }
]
```

Deployment Skripte lokal entwickeln

Demo

Alternativen zu Minikube

- k3s
- k3d
- kind
- microk8s
- k0s


App K8s-ready machen
Versionierung
Debugging
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
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

```yaml
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 }}"
```

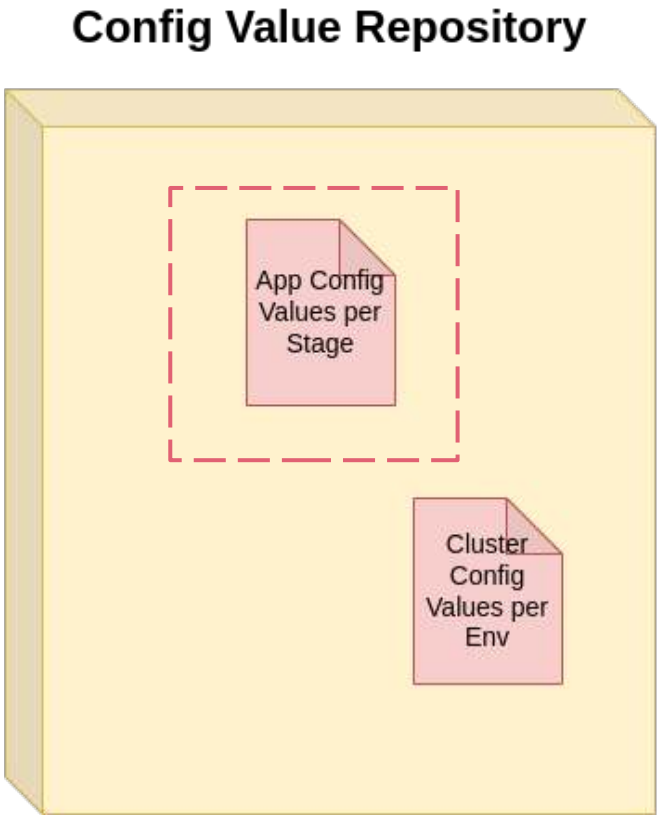
Helm Charts anpassen

```
# 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
```

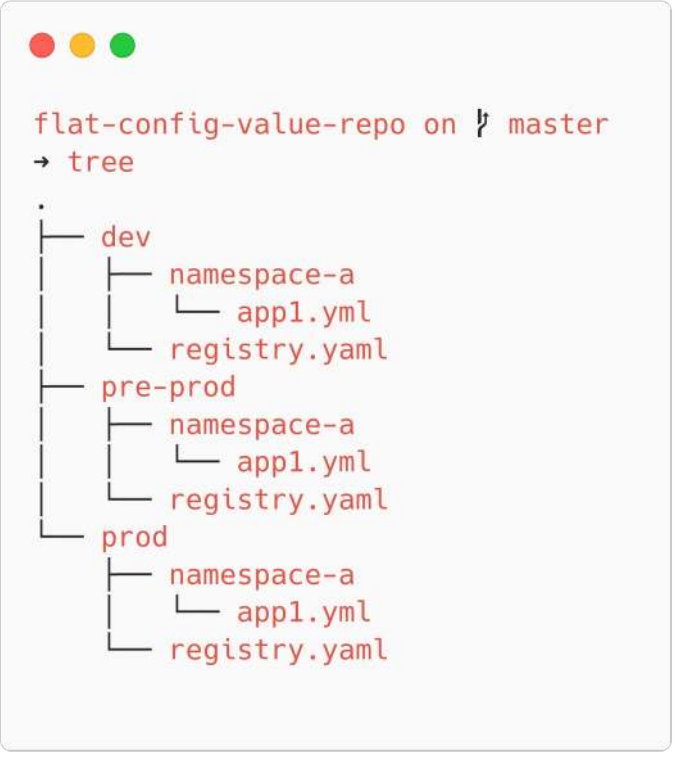
Helm Charts anpassen

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

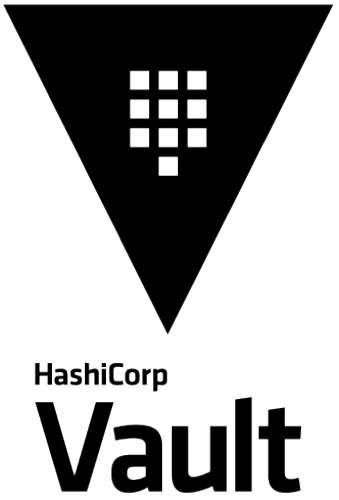
Konfiguration verwalten



Konfiguration verwalten



Secrets



Cloud Lösungen (Bsp):

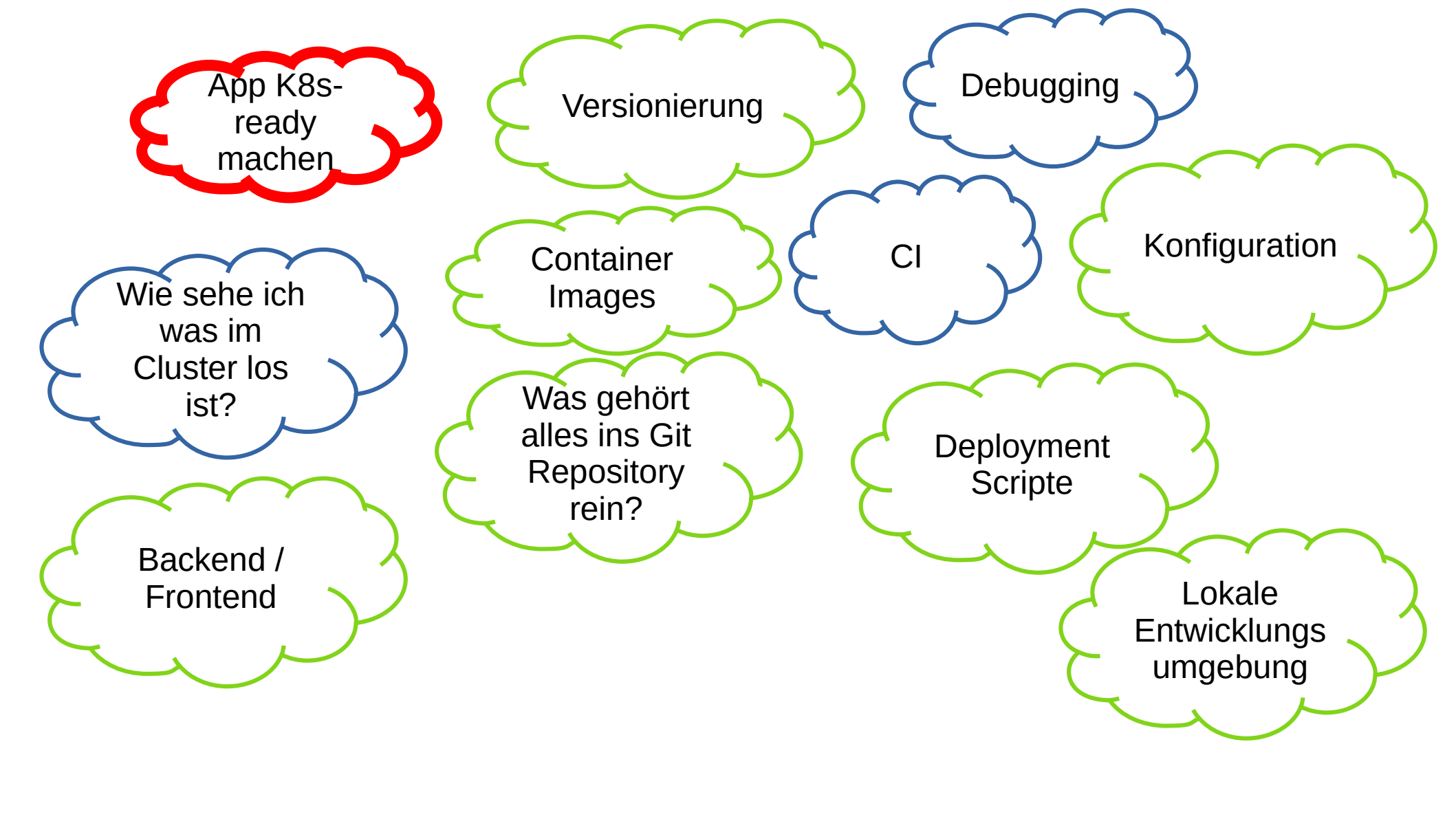
- Google Secret Manager
- AWS Secrets & Configuration Provider
- Azure Key Vault Provider

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

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
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
Was gehört alles ins Git Repository rein?
Deployment Scripte
Backend / Frontend
Lokale Entwicklungs umgebung

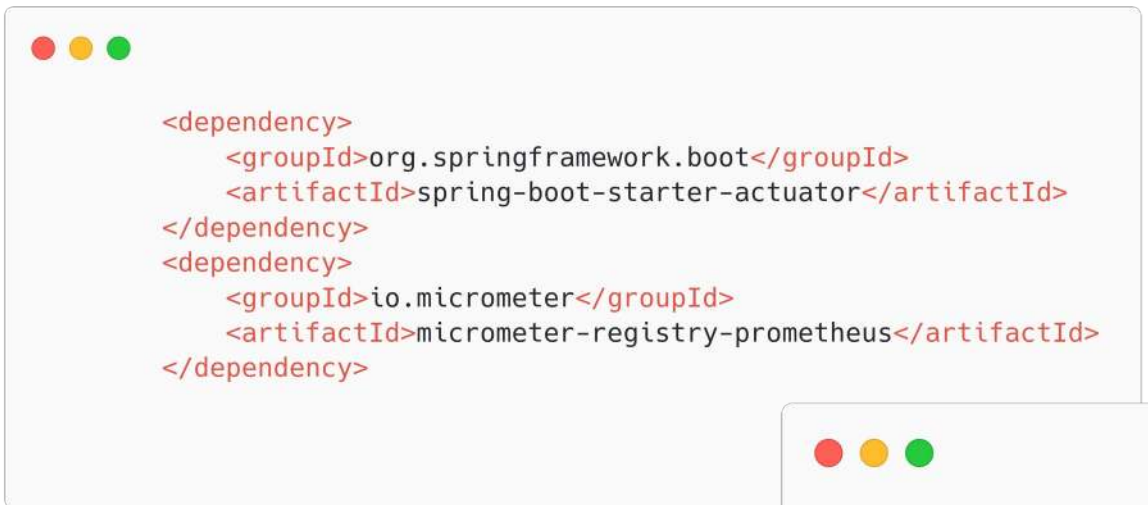
Good Practices für Anwendungen in Container

- 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

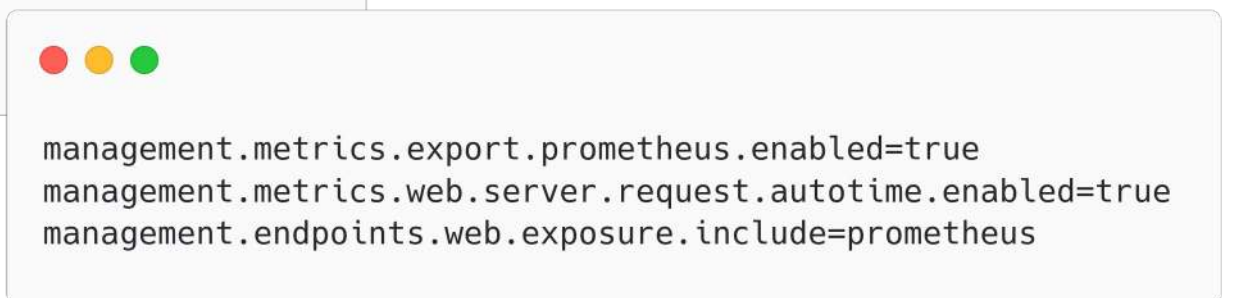
Log-Nachrichten auf stdout

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

```yaml
# 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

```yaml
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

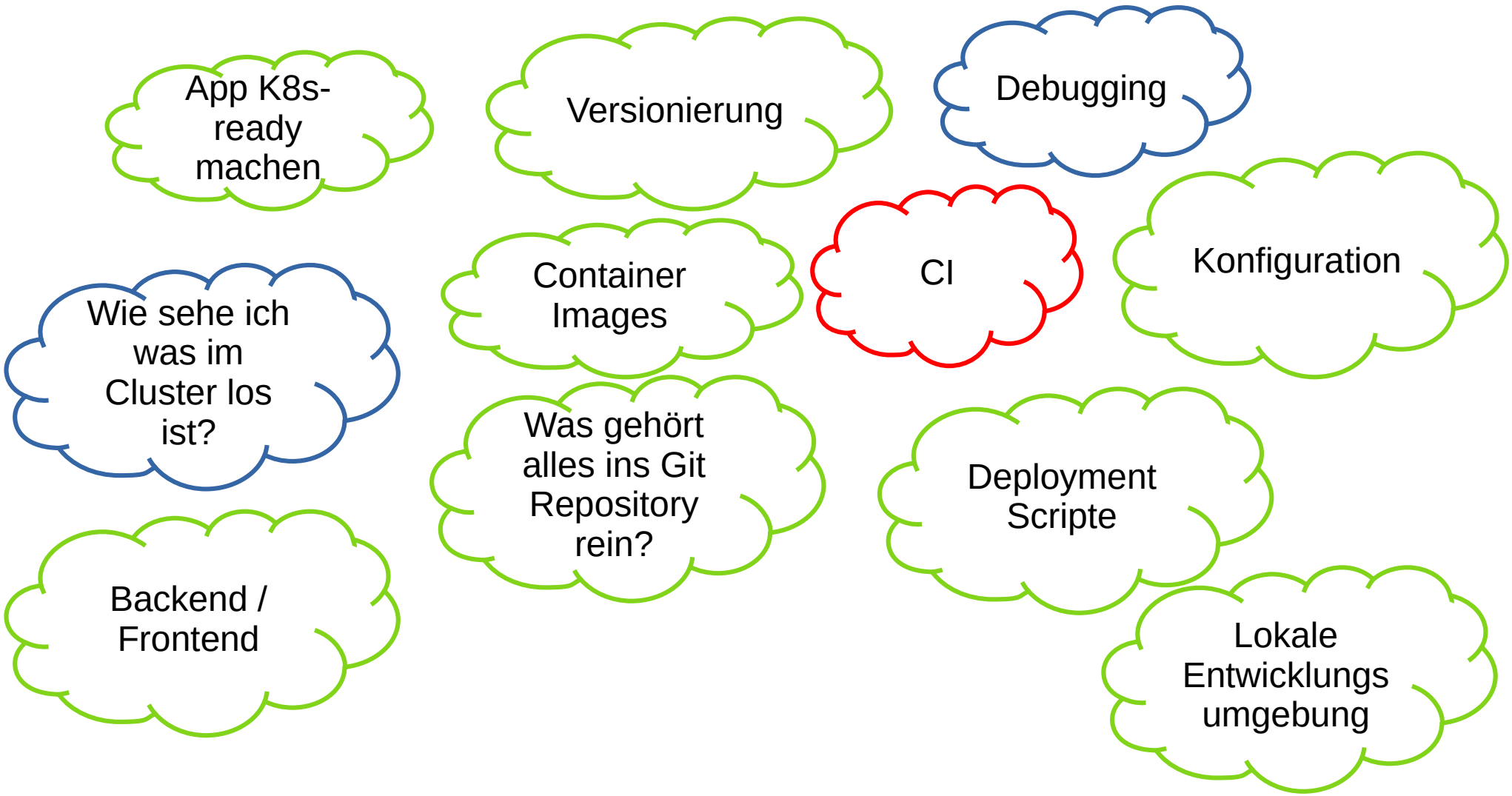
- JVM Default Einstellung (welcher GC oder default max Heap Size) abh. von Umgebung und Java Version
- Lesson Learnt: Immer die JVM konfigurieren
- Don’t worry: Es gibt Empfehlungen

JVM und Kubernetes

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

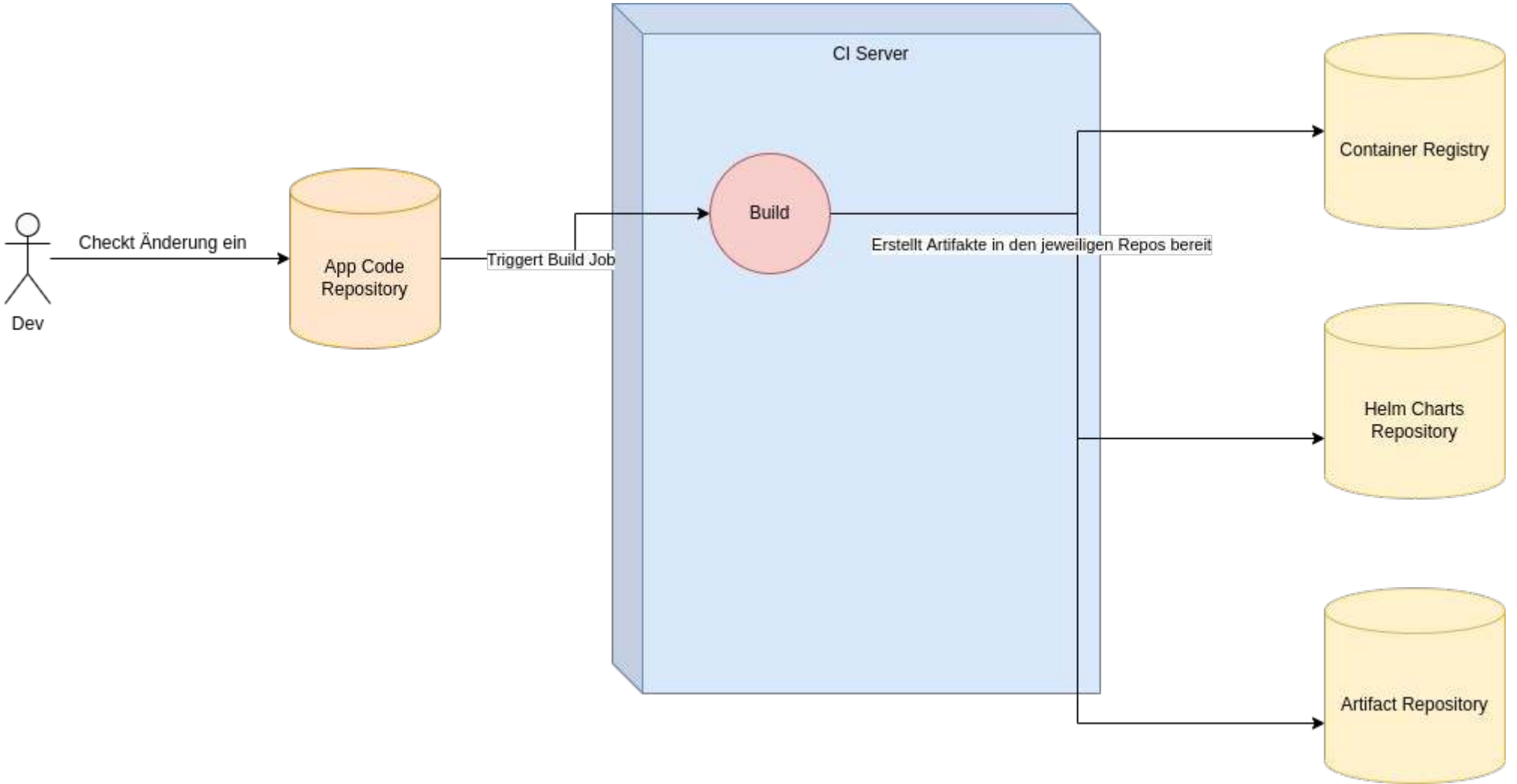
Weitere Infos zu JVM und Kubernetes

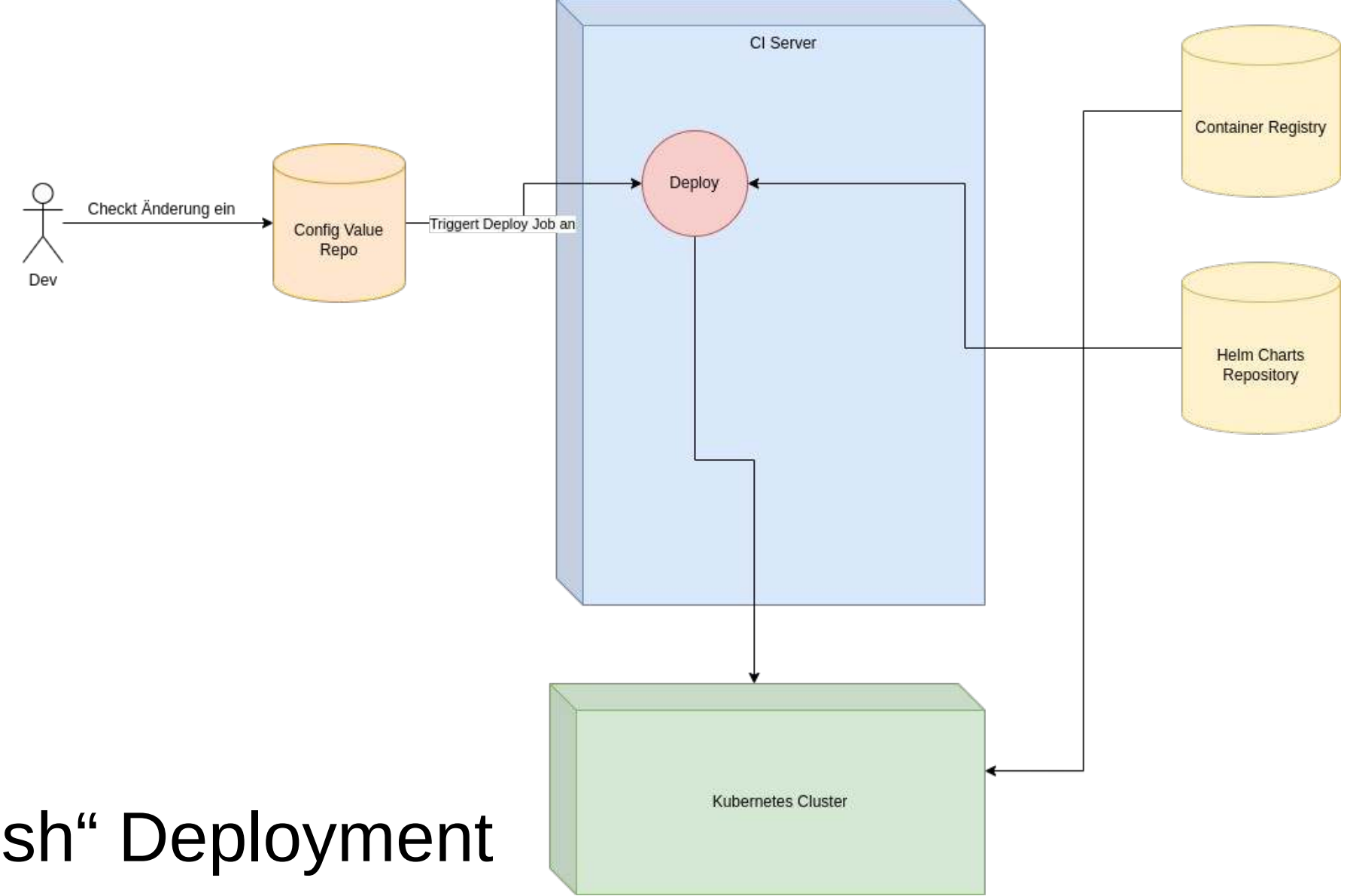
- Secrets of Performance Tuning Java on Kubernetes von Bruno Borges
- We Moved one Java Product to Kubernetes and This Is What We Learned von Carlos Sanchez
- Tuning and Optimizing Java Garbage Collection von Monica Beckwith


App K8s-ready machen
Versionierung
Debugging
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
Was gehört alles ins Git Repository rein?
Deployment Scripte
Backend / Frontend
Lokale Entwicklungs umgebung

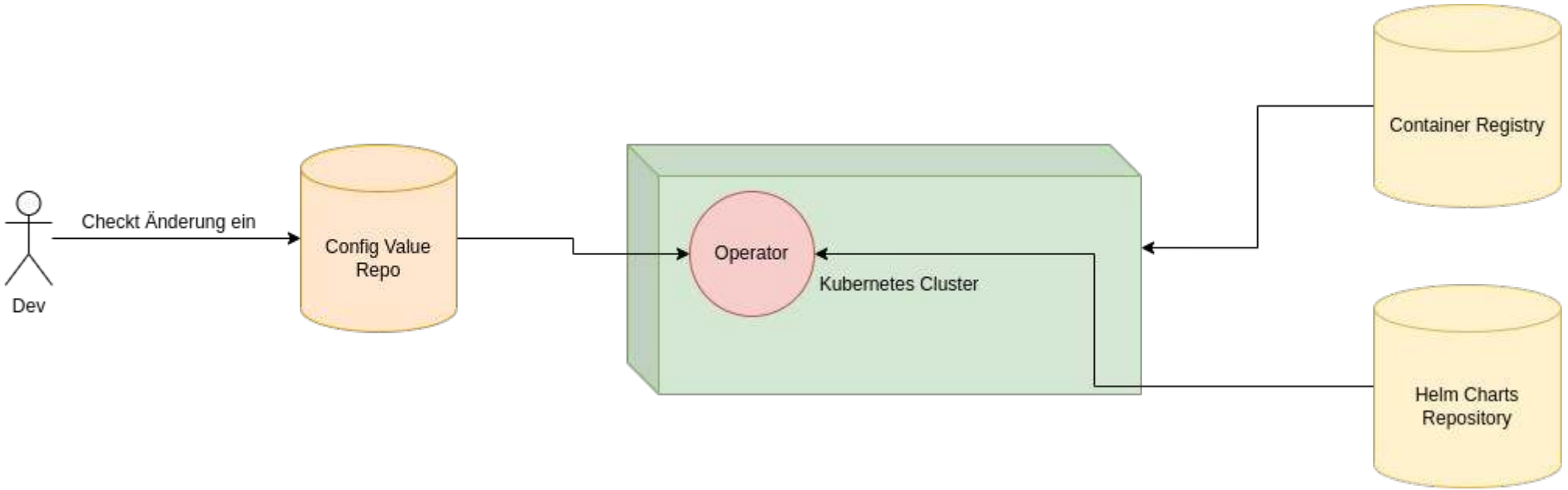
12 Factor App:
Build- und Run-Phase strikt trennen

Build

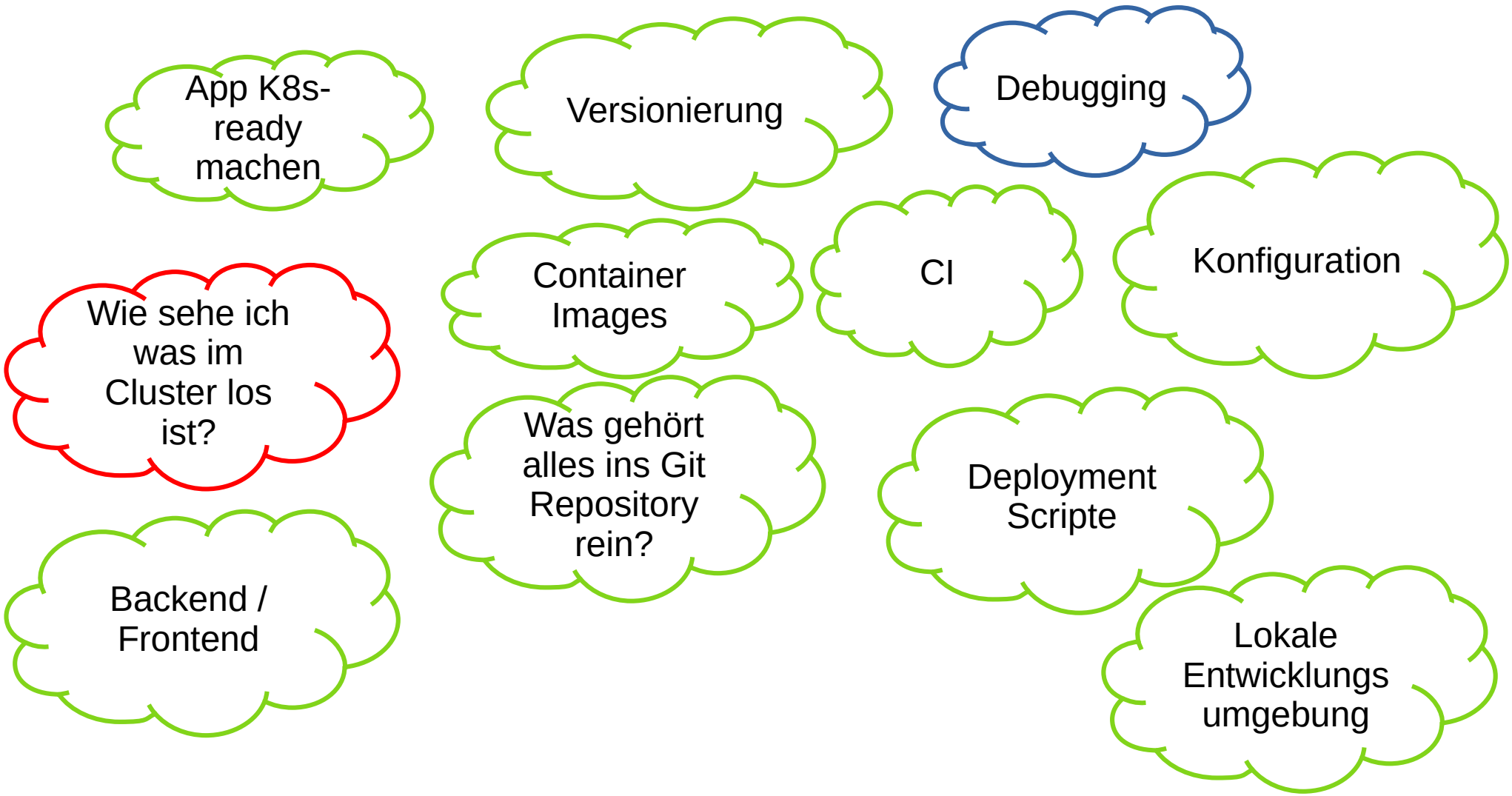

CI Server
Container Registry
Build
Checkt Änderung ein
App Code Repository
Triggert Build Job
Erstellt Artifakte in den jeweiligen Repos bereit
Helm Charts Repository
Dev
Artifact Repository



„Push“ Deployment



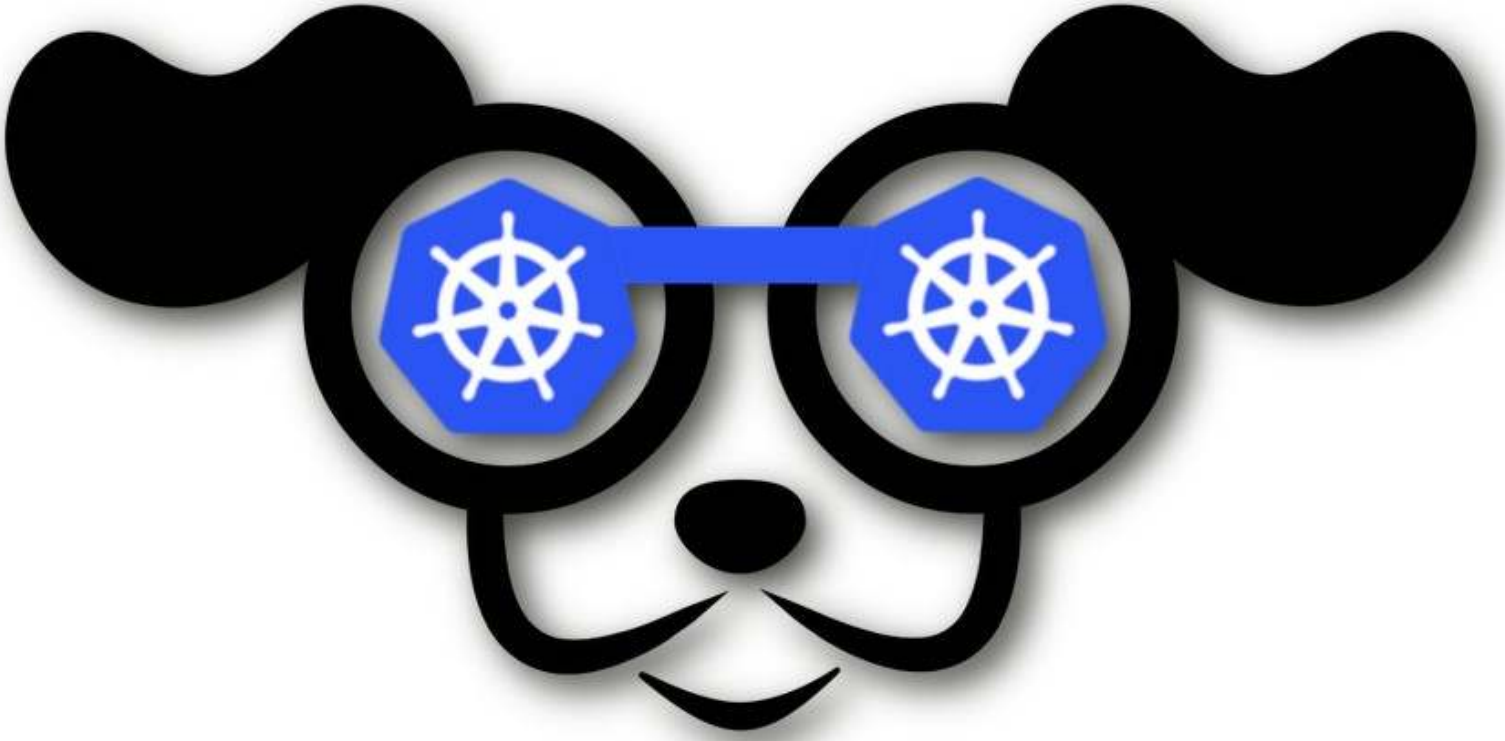
„Pull“ Deployment


App K8s-ready machen
Versionierung
Debugging
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
Was gehört alles ins Git Repository rein?
Deployment Scripte
Backend / Frontend
Lokale Entwicklungs umgebung

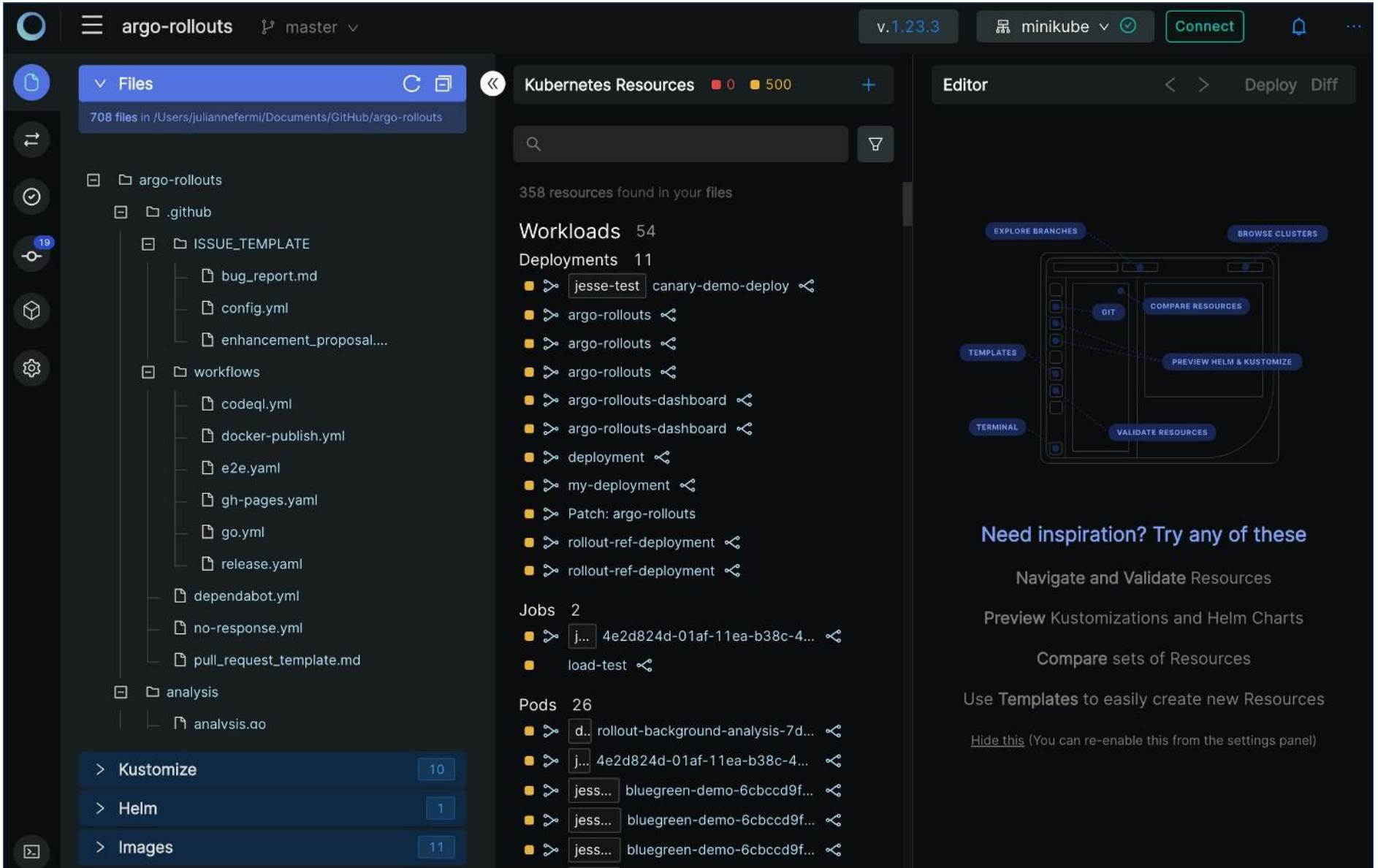
kubectl

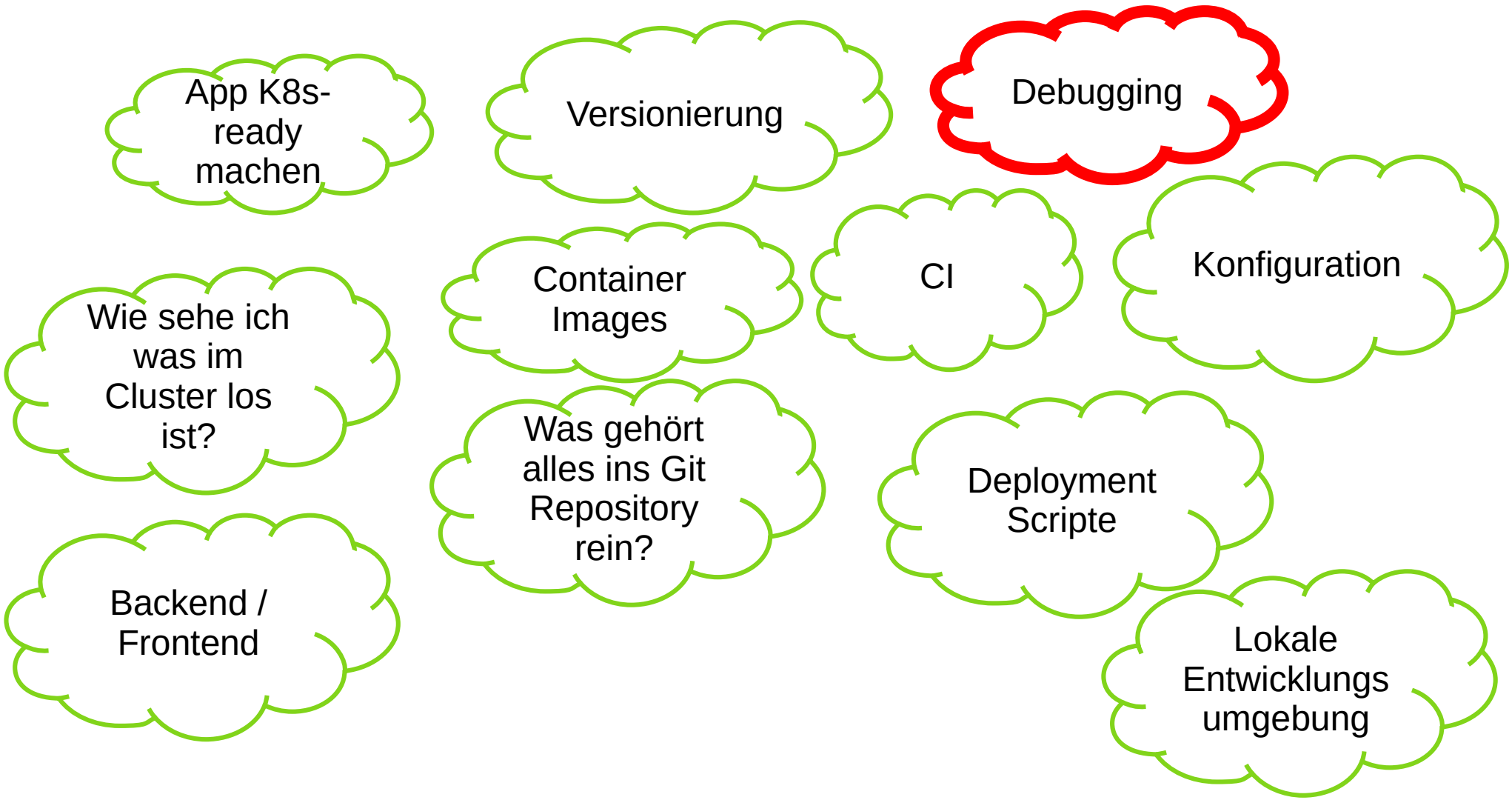
k9s

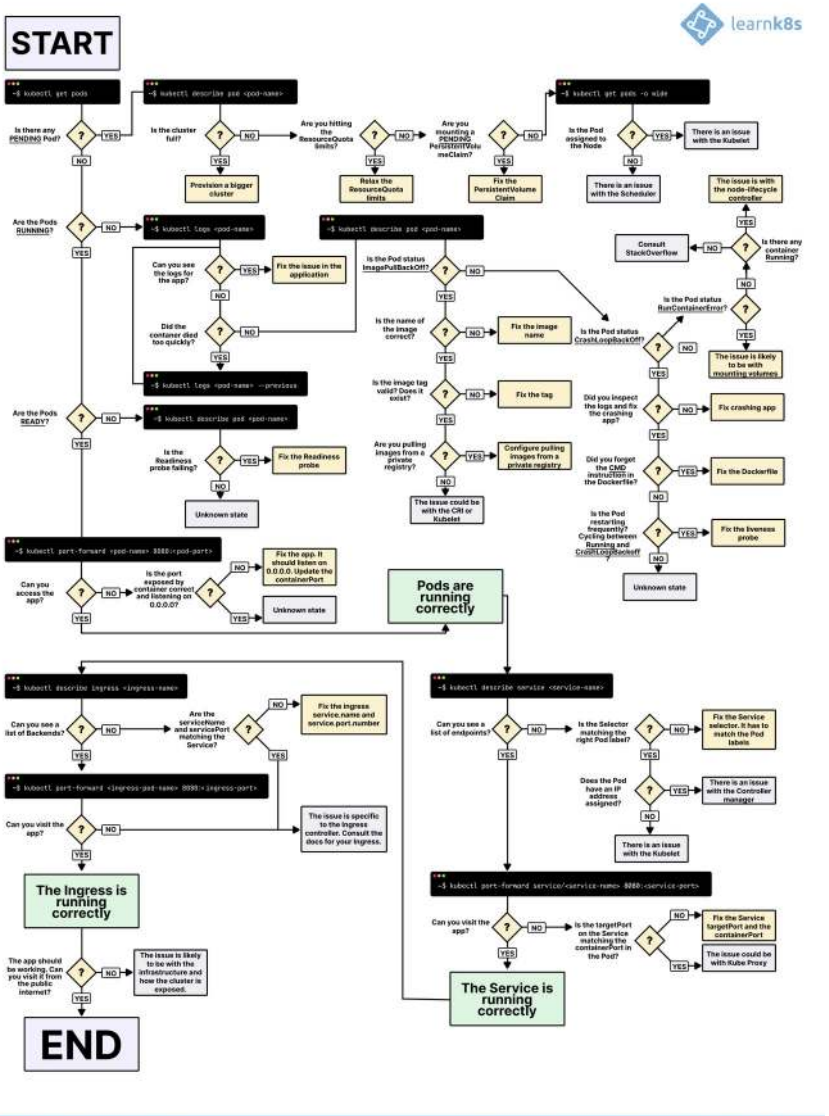
Kubernetes CLI To Manage Your Clusters In Style!



Monokle Desktop


argo-rollouts
master
v.1.23.3
minikube
Connect
Files
708 files in /Users/juliannefermi/Documents/GitHub/argo-rollouts
argo-rollouts
.github
ISSUE_TEMPLATE
bug_report.md
config.yml
enhancement_proposal....
workflows
codeql.yml
docker-publish.yml
e2e.yaml
gh-pages.yaml
go.yml
release.yaml
dependabot.yml
no-response.yml
pull_request_template.md
analysis
analysis.go
Kustomize 10
Helm 1
Images 11
Kubernetes Resources 0 500
358 resources found in your files
Workloads 54
Deployments 11
jesse-test canary-demo-deploy
argo-rollouts
argo-rollouts
argo-rollouts
argo-rollouts-dashboard
argo-rollouts-dashboard
deployment
my-deployment
Patch: argo-rollouts
rollout-ref-deployment
rollout-ref-deployment
Jobs 2
j... 4e2d824d-01af-11ea-b38c-4...
load-test
Pods 26
d.. rollout-background-analysis-7d...
j... 4e2d824d-01af-11ea-b38c-4...
jess... bluegreen-demo-6cbccd9f...
jess... bluegreen-demo-6cbccd9f...
jess... bluegreen-demo-6cbccd9f...
Editor
Deploy
Diff
EXPLORE BRANCHES
BROWSE CLUSTERS
GIT
COMPARE RESOURCES
TEMPLATES
PREVIEW HELM & KUSTOMIZE
TERMINAL
VALIDATE RESOURCES
Need inspiration? Try any of these
Navigate and Validate Resources
Preview Kustomizations and Helm Charts
Compare sets of Resources
Use Templates to easily create new Resources
Hide this (You can re-enable this from the settings panel)


App K8s-ready machen
Versionierung
Debugging
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
Was gehört alles ins Git Repository rein?
Deployment Scripte
Backend / Frontend
Lokale Entwicklungs umgebung



https://learnk8s.io/troubleshooting-deployments

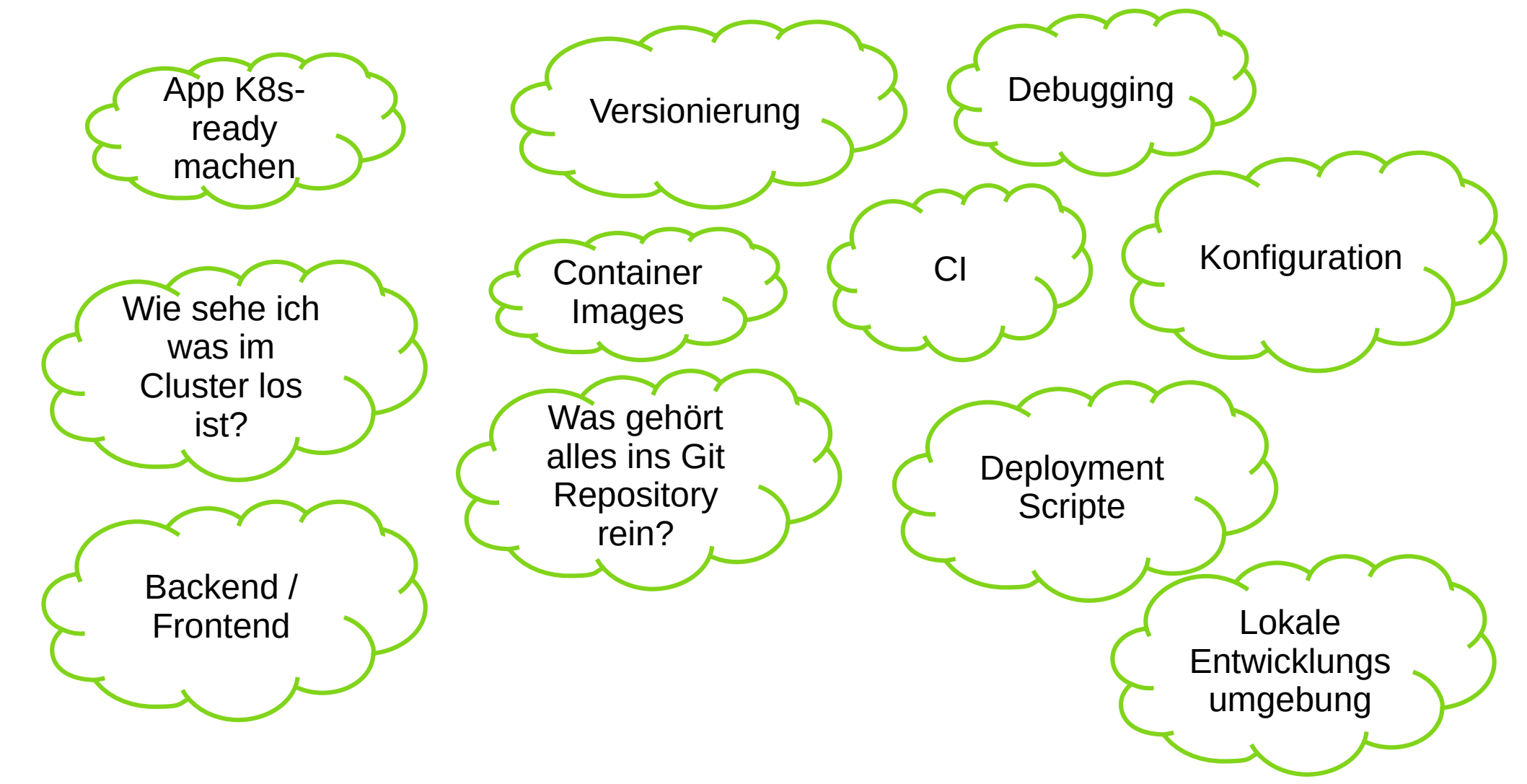
Troubleshooting Applications

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.

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:
\"sh\": executable file not found in $PATH": unknown

$ 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
Container Images
CI
Konfiguration
Wie sehe ich was im Cluster los ist?
Was gehört alles ins Git Repository rein?
Deployment Scripte
Backend / Frontend
Lokale Entwicklungs umgebung

Fragen?

mail@sandra-parsick.de
@sparsick@mastodon.social

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

architektur SPICKER

Übersichten für die konzeptionelle Seite der Softwareentwicklung

MEHR WISSEN IN KOMPAKTER FORM:
Weitere Architektur-Spicker gibt es als kostenfreies PDF unter www.architektur-spicker.de

NR. 11

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.

IN DIESER AUSGABE

Worum geht's?

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

Anwendungsentwurf, Container und Orchestrierung

Die Zusammenhänge. Fragen und Antworten

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

1. Warum zerlegen wir Anwendungen?

Die Zerlegung einer komplexen Anwendung macht sie wartbar und beherrschbar.

Ohne Zerlegung ließe sich z.B. nicht mit mehreren Teams gleichzeitig daran arbeiten.

Die Zerlegung kann zu mehreren Prozessen und einer verteilten Anwendung führen (Alternative: Modulith).

2. Was haben Microservices damit zu tun?

Microservices sind ein Architekturstil, also eine grundlegende Art Anwendungen zu bauen.

Die einzelne Anwendung besteht dabei aus relativ kleinen, lose gekoppelten Services.

Jeder Microservice ist ein Prozess und unabhängig installierbar.

→ Zu Architekturstilen siehe Seite 2
→ Zu Microservices siehe Spicker #3

3. Warum sollten wir Anwendungen in verschiedenen Prozessen laufen lassen?

So lassen sich Anwendungsteile gezielter skalieren.

Es erhöht die technologische Freiheit; Teile können unterschiedlich gebaut sein (z.B. Programmiersprachen).

4. Und was sind Container?

Container sind vergleichbar mit VMs, aber kleiner.

Auf einem Betriebssystem können viele Container parallel laufen.

Container beinhalten jeweils nur eine Anwendung und alles, was es dazu braucht.

Container teilen sich die Ressourcen und sind gleichzeitig voneinander isoliert.

Docker ist die am Markt verbreitetste Container-Technologie.

5. Wie helfen Container bei verteilten Anwendungen?

Container können Anwendungsteile effizient isolieren.

Sie bieten ein einheitliches Deployment-Format (Images) für unterschiedliche Technologien.

6. Wie kommen Orchestrierungslösungen ins Spiel?

Mit Orchestrierungslösungen lassen sich Anwendungen einfacher definieren, strukturieren und konfigurieren.

Ein Betrieb auf mehreren Rechnerknoten erhöht die Verfügbarkeit und eröffnet eine bessere Lastverteilung.

Sie überwachen den Zustand der Anwendung und heilen bei Bedarf.

Kubernetes ist die am Markt verbreitetste Orchestrierungslösung.

→ Mehr zu Containern ab Seite 3
→ Mehr zu Kubernetes ab Seite 4

architektur SPICKER

Übersichten für die konzeptionelle Seite der Softwareentwicklung

MEHR WISSEN IN KOMPAKTER FORM:
Weitere Architektur-Spicker gibt es als kostenfreies PDF unter www.architektur-spicker.de

NR. 7

Continuous Delivery

Zeitgemäße Techniken aus Continuous Integration (CI) und Continuous Delivery (CD) unterstützen wichtige Architekturziele wie Stabilität und Reaktionsfähigkeit. Dieser Spicker zeigt den Aufbau einer passenden CI/CD-Kette.

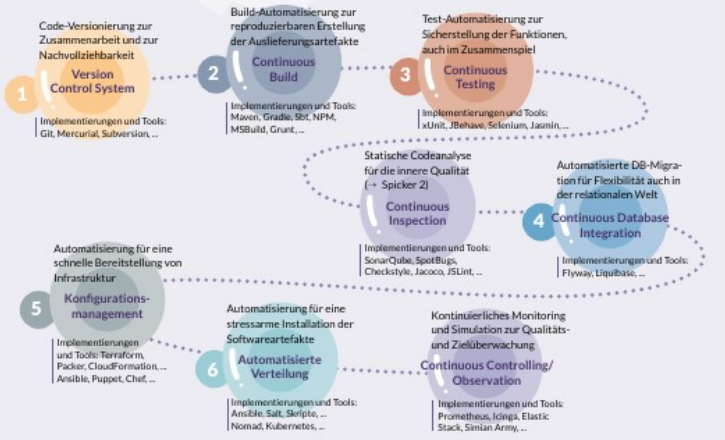
IN DIESER AUSGABE

Worum geht's? (Herausforderungen/Ziele)

- Neue Features in eure Lösung zu integrieren ist aufwändig und fehleranfällig. Wie minimiert ihr dieses Risiko?
- Moderne Architekturansätze wie Microservices haben hohe Anforderungen bzgl. Integration und Verteilung. Welche Wechselwirkungen bestehen zwischen Architekturstil und CI/CD?
- Manuelle, wiederkehrende Tätigkeiten binden Kräfte und lassen sich nicht in gleichbleibender Qualität wiederholen. Wie eliminiert das Team diese monotonen Aufgaben?
- Auswirkungen von Änderungen in Quelltext, Technologie und Konfiguration werden erst spät im Entwicklungsprozess erkannt. Wie erhaltet ihr rasch Feedback?

Die Continuous Delivery Perlenkette

CD automatisiert die Integrations- und Verteilungsprozesse von der Codierung bis zur lauffähigen Software, um schnell und verlässlich zu liefern. Die folgende Abbildung fädelt die wesentlichen Elemente zu einer CD-Kette zusammen:



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&utm_content=creditShareLink
- https://unsplash.com/photos/CpsTAUPoScw?utm_source=unsplash&utm_medium=referral&utm_content=creditShareLink