GEDOPLAN

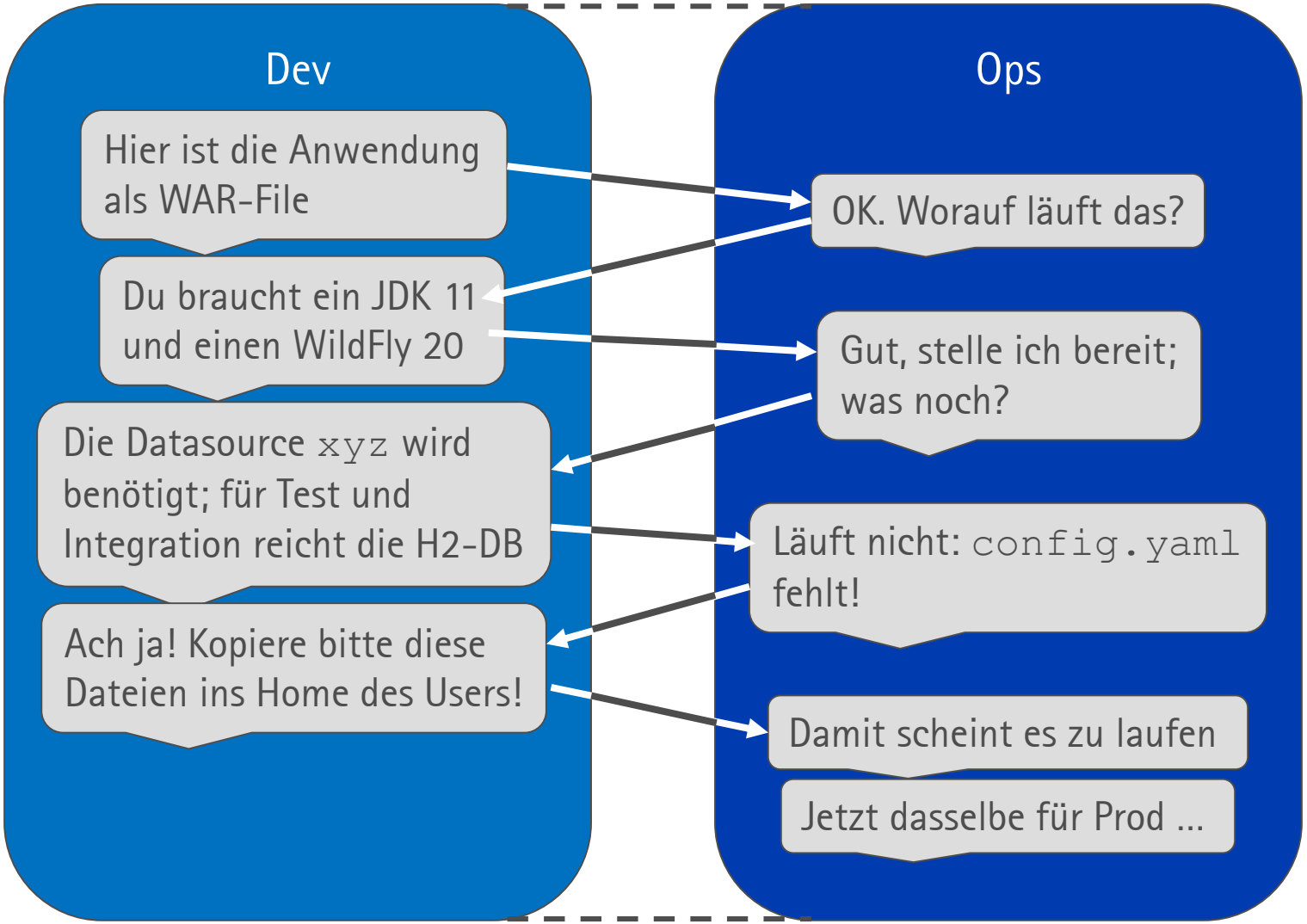
Das Runde muss in das Eckige
Wie aus Java-Anwendungen Container werden

Dirk Weil, GEDOPLAN GmbH

Dirk Weil

- GEDOPLAN GmbH, Bielefeld
  - GEDOPLAN IT Consulting
    Consulting, coaching, concepts, reviews, development
  - GEDOPLAN IT Training & partner
    Java, JEE, tools trainings in Berlin, Bielefeld, Köln, on-site
- JEE since 1998
- Speaker and author

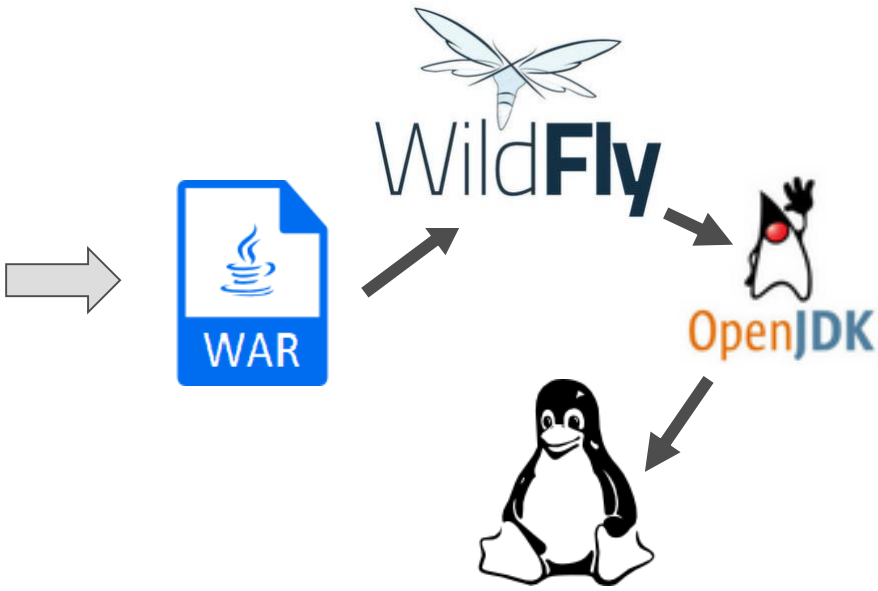
Application Stage Odyssey



Application Stage Odyssey

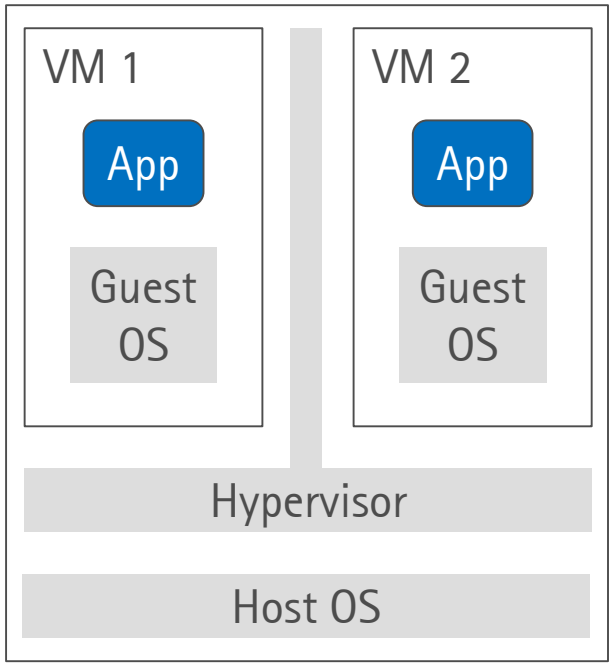
- Anwendung ≠ JAR/WAR/EAR!

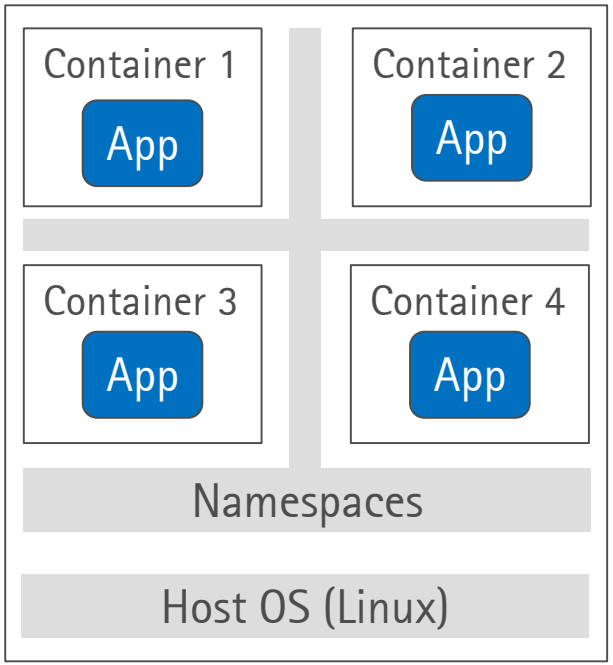
```java
@Path("hello")
@ApplicationScoped
public class HelloResource {
  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String getHello() {
    return "Hello World!";
  }
}
```



- Lösungsmöglichkeit: Virtuelle Maschine
  - Komplettes OS mit Tools, Dateien etc.
  - automatisierbar mit Ansible o. ä.
  - schwergewichtig!

Virtuelle Maschinen vs. Container



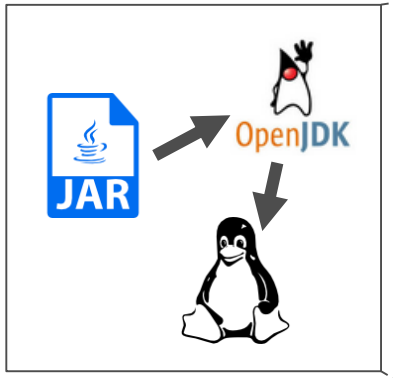


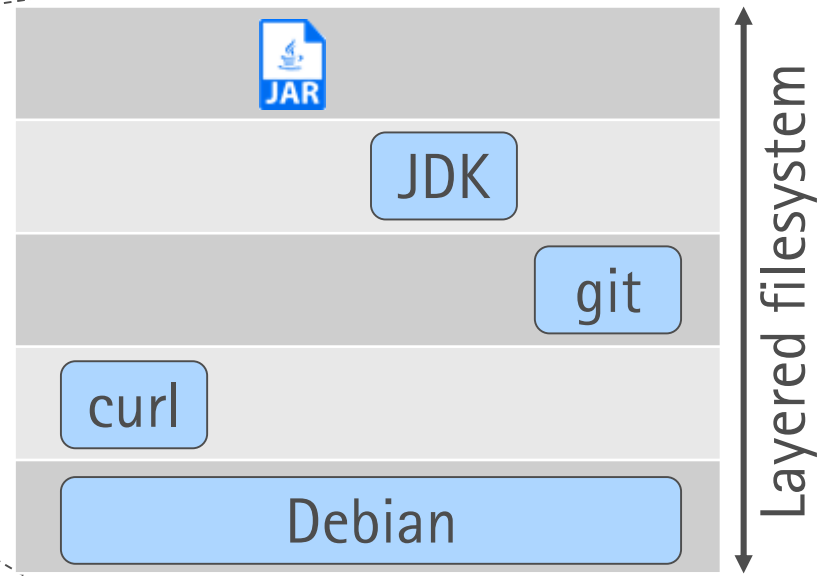
- VM = „kompletter Rechner“
- Hypervisor teilt HW zu
- Isolation durch Hypervisor

- Container = Prozess auf Host OS
- Isolation durch Namespaces

Docker


```
FROM openjdk:8

ADD target/k8s-demo-hello.jar /opt/

ENTRYPOINT [ "java", "-jar", "/opt/k8s-demo-hello.jar" ]
```





Docker

- Demo
  - Docker for Windows

```
# Build image
docker build -t gedoplan/k8s-demo-hello .

# Run container
docker run --rm gedoplan/k8s-demo-hello
```

Docker

- io.fabric8:docker-maven-plugin
  - Integriert Docker Build in Maven Build
  - Interpoliert Maven-Properties im Dockerfile
  - Assembly mit vorbereiteten Descriptoren
  - diverse weitere Operationen

```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>docker-maven-plugin</artifactId>
  <configuration>
    <images>
      <image>
        <name>gedoplan/${project.artifactId}</name>
        <build>
          <dockerFileDir>${project.basedir}</dockerFileDir>
          <assembly>
            <descriptorRef>artifact</descriptorRef>
          </assembly>
          …
```

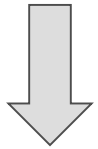
Docker

- Demo

```
# Einfache SE-Anwendung: k8s-demo-hello
mvn -Pdocker
docker run gedoplan/k8s-demo-hello

# REST-Service als WAR auf WildFly: k8s-demo-rest-war
mvn -Pdocker
docker run -p 8888:8080 gedoplan/k8s-demo-rest-war

# REST-Service mit Apache Meecrowave: k8s-demo-rest-mee
mvn -Pdocker
docker run -p 8888:8080 gedoplan/k8s-demo-rest-mee
```

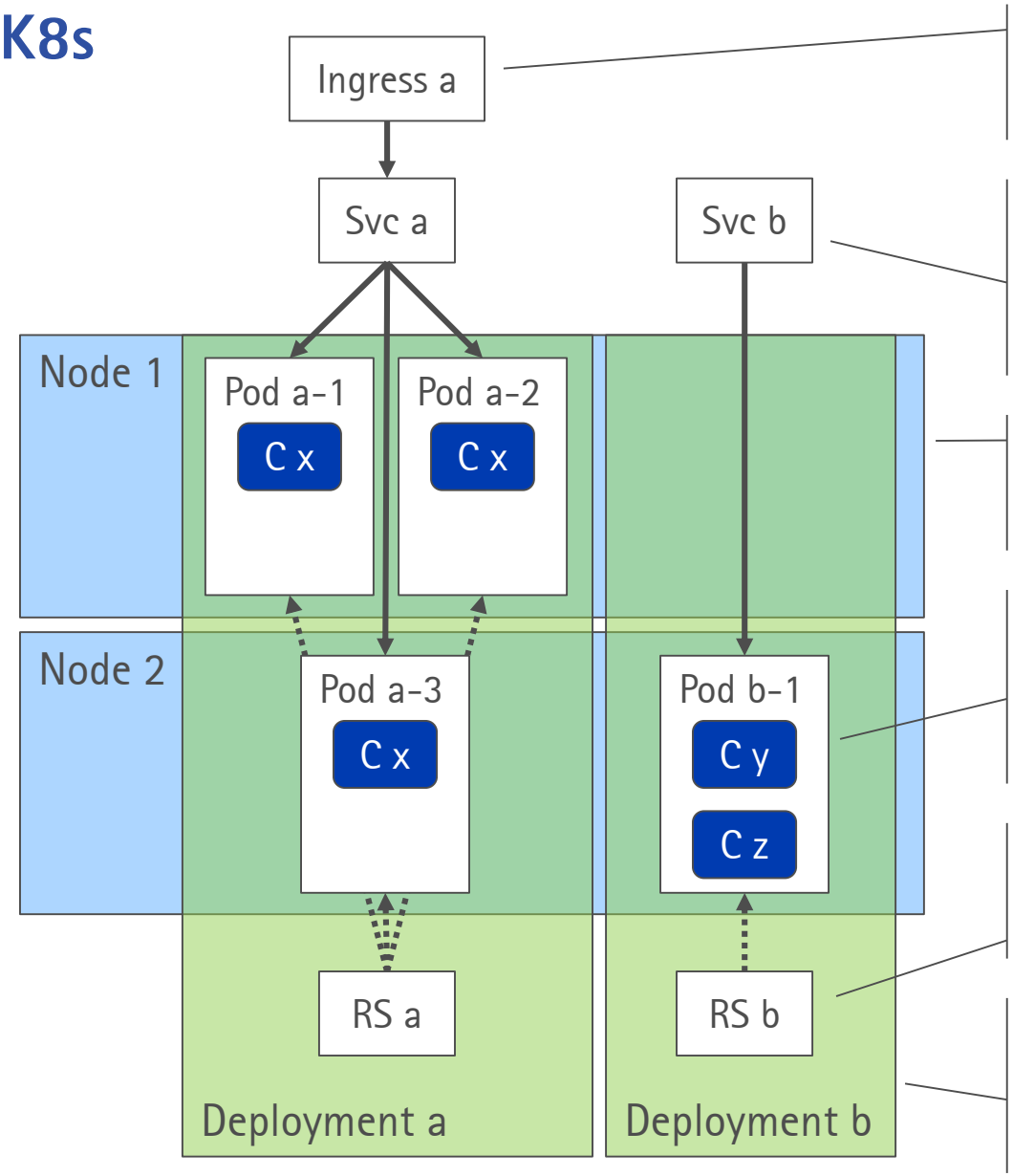
Kubernetes (aka K8s)

- Container (~Prozesse) brauchen Laufzeitplattform (~Betriebssystem)
  - Start, Stopp, Überwachung
  - Verteilung (On-prem / Cloud)
  - Networking / Discovery
  - Skalierung
  - Persistenz



kubernetes

K8s



Ingress
Reverse Proxy

Service
feste IP-Adresse
Load Balancer

Node
„echter Rechner"

Pod
ein Container, ggf. mehrere
Abstraktion eines Rechners

ReplicaSet
sorgt für n laufende Pods

Deployment
ergänzt RS um Rolling Update

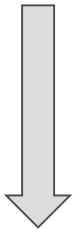
K8s

- Demo

```
# K8s-Manifest für REST-Service als WAR: rest-war.yaml
#  (enthält Deployment, Service + Ingress)
kubectl apply -f rest-war.yaml

# Pod-Neustart nach (simuliertem) Ausfall
kubectl get all -l app=rest-war
kubectl delete pod rest-war-xxx

# Pod-Anzahl manipulieren
kubectl edit deployment rest-war

# Webservice „von Außen“ konsumieren
curl http://rest-war.localtest.me/rs/hello
```

Paketieren und Ausliefern

- K8s-Application =
  - Deployment(s)
  - Service(s)
  - Ingress(es)
  - (ConfigMaps, Secrets, Persistent Volumes, ...)
- Anwendungen in unterschiedlichen Umgebungen ausrollbar
  - Prod, Test, lokal, ...



- Bedarf: Application- / Package-Management und Templating

Werkzeuge zum Paketieren

- ermöglichen:
  - K8s-Manifeste zusammenfassen zu Anwendungen
  - Templating

HELM

Kustomize

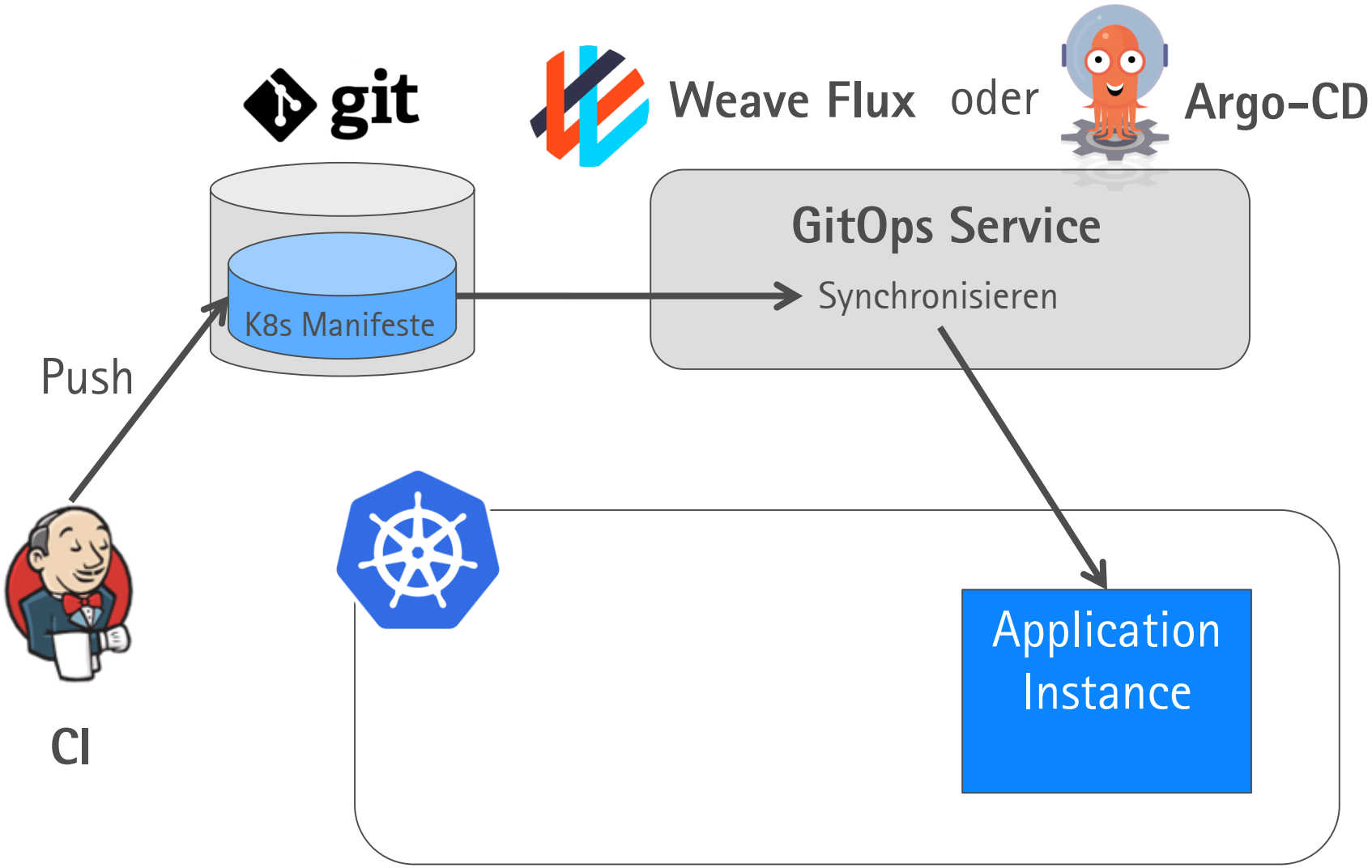
Kapitan

Ausrollen der Anwendung

- Anforderungen
  - Immutable Pakete mit K8s-Manifesten
  - Zentrale Bereitstellung der Pakete -> einfaches Verteilen
  - Historie der Releases
  - Automatisiertes Ausrollen

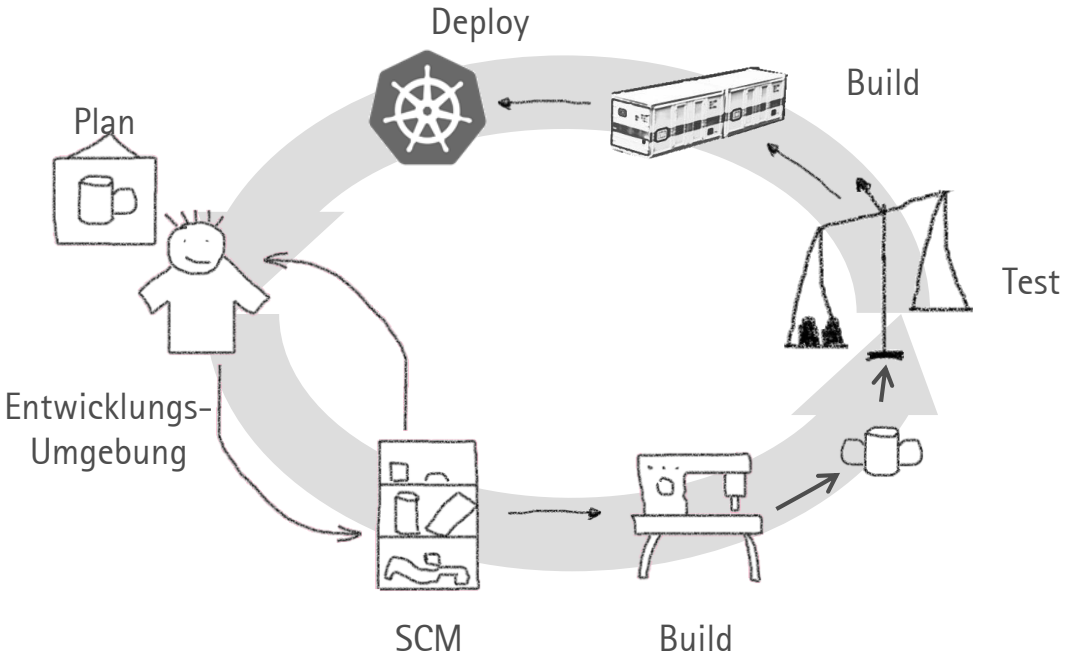
- Probleme beim Ausrollen per CMD-Werkzeug
  - Nicht wiederholbar bei Verwendung von Parametern
  - Erfordert entsprechende Berechtigung

GitOps


git
Weave Flux
oder
Argo-CD
GitOps Service
Synchronisieren
K8s Manifeste
Push
CI
Application Instance

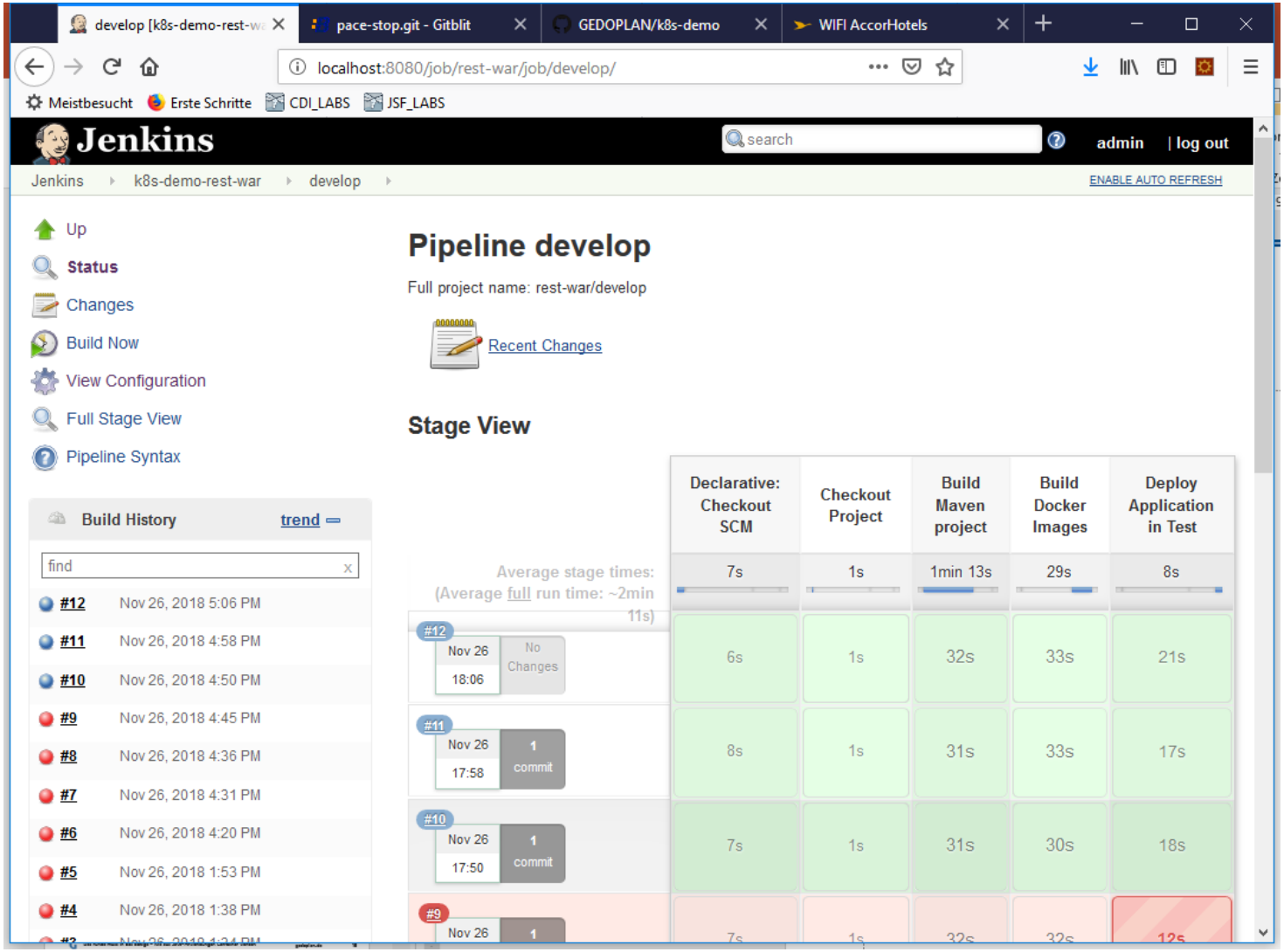
Jenkins

- CI/CD-Server
- Ausführung in K8s
  - Jenkins läuft im Cluster
  - Konfigurierbare Build-Container
  - Deployment in K8s per GitOps

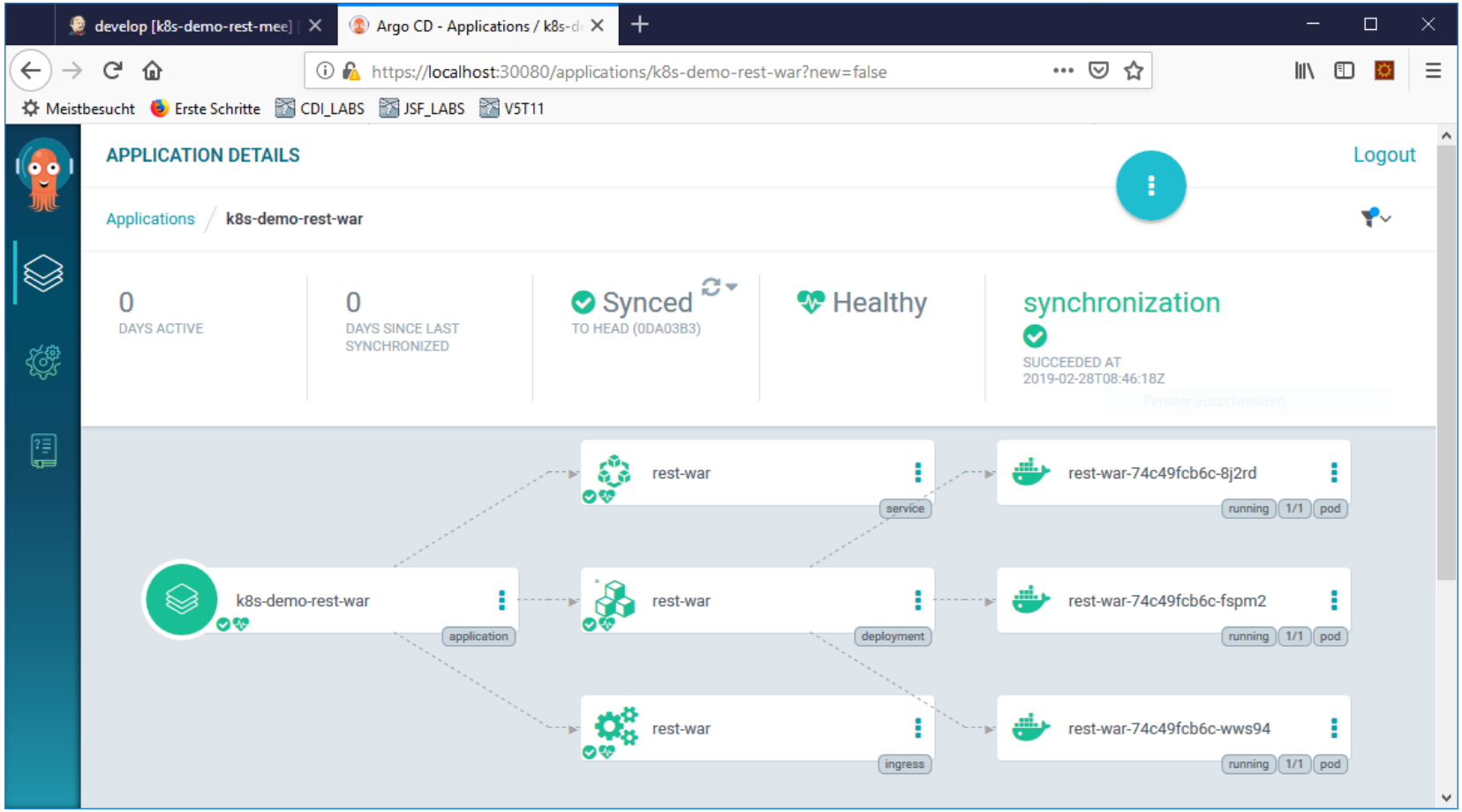



```
pipeline {
  agent {
    kubernetes {
      label 'rest-war-build‘
    …
  }

stages {
  stage('Checkout Project‘) { … }
  stage('Build Maven project‘) { … }
  stage('Build Docker Images‘) { … }
  stage('Deploy Application in Test’) { … }
```

Jenkins


develop [k8s-demo-rest-wa
pace-stop.git - Gitblit
GEDOPLAN/k8s-demo
WIFI AccorHotels
localhost:8080/job/rest-war/job/develop/
Meistbesucht
Erste Schritte
CDI_LABS
JSF_LABS
Jenkins
search
admin
log out
Jenkins
k8s-demo-rest-war
develop
ENABLE AUTO REFRESH
Up
Status
Changes
Build Now
View Configuration
Full Stage View
Pipeline Syntax
Pipeline develop
Full project name: rest-war/develop
Recent Changes
Stage View
Build History
trend
find
#12 Nov 26, 2018 5:06 PM
#11 Nov 26, 2018 4:58 PM
#10 Nov 26, 2018 4:50 PM
#9 Nov 26, 2018 4:45 PM
#8 Nov 26, 2018 4:36 PM
#7 Nov 26, 2018 4:31 PM
#6 Nov 26, 2018 4:20 PM
#5 Nov 26, 2018 1:53 PM
#4 Nov 26, 2018 1:38 PM
Declarative: Checkout SCM
Checkout Project
Build Maven project
Build Docker Images
Deploy Application in Test
Average stage times:
(Average full run time: ~2min 11s)
7s
1s
1min 13s
29s
8s
#12 Nov 26 18:06 No Changes
6s 1s 32s 33s 21s
#11 Nov 26 17:58 1 commit
8s 1s 31s 33s 17s
#10 Nov 26 17:50 1 commit
7s 1s 31s 30s 18s
#9 Nov 26 1
7s 1s 32s 32s 12s

Argo-CD


develop [k8s-demo-rest-mee]
Argo CD - Applications / k8s-d
https://localhost:30080/applications/k8s-demo-rest-war?new=false
Meistbesucht
Erste Schritte
CDI_LABS
JSF_LABS
V5T11
APPLICATION DETAILS
Logout
Applications
k8s-demo-rest-war
0
DAYS ACTIVE
0
DAYS SINCE LAST SYNCHRONIZED
Synced
TO HEAD (0DA03B3)
Healthy
synchronization
SUCCEEDED AT
2019-02-28T08:46:18Z
k8s-demo-rest-war
application
rest-war
service
rest-war
deployment
rest-war
ingress
rest-war-74c49fcb6c-8j2rd
running
1/1
pod
rest-war-74c49fcb6c-fspm2
running
1/1
pod
rest-war-74c49fcb6c-wws94
running
1/1
pod

More

- **`github.com/GEDOPLAN/k8s-demo`**
  Demo project, slides

- **`www.gedoplan-it-training.de`**
  Trainings in Berlin, Bielefeld, inhouse
  - Kurse zum Thema:
    Docker und Kubernetes für Java-Entwickler
    Java DevOps: Development und Delivery mit Docker und Kubernetes

- **`www.gedoplan-it-consulting.de`**
  Reviews, Coaching, ...
  Blog

- **`dirk.weil@gedoplan.de`**
- **`@dirkweil`**