Build & Deployment von Microservices mit GitLab CI

Christine Koppelt
christine.koppelt@innoq.com

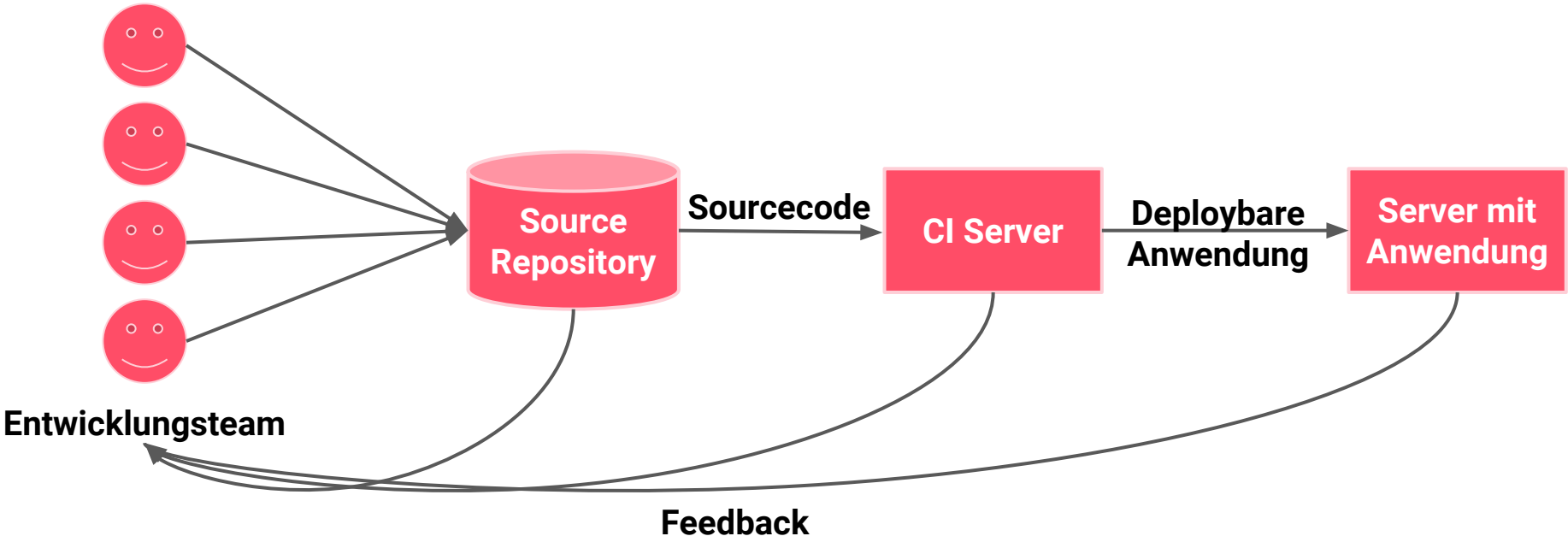
Philipp Haußleiter
philipp.haussleiter@innoq.com

Code Days München, 08.02.2018

INNOQ

Continuous Integration (CI) & Microservices

Continuous Integration (CI)

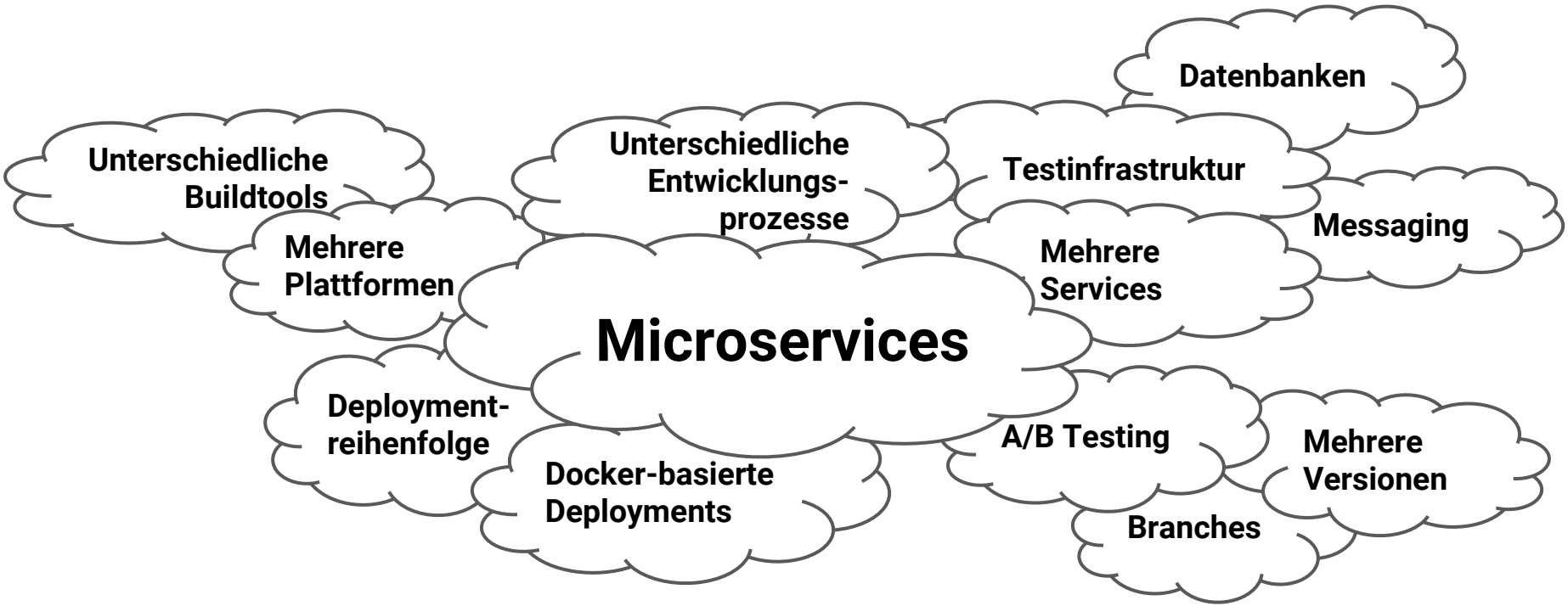


Microservices

- **Mehrere eigenständige Services bilden ein Gesamtsystem**
  - **Unterteilt nach Bounded Contexts (Domain Driven Design)**
  - **Eigenständige Anwendungen (Frontend, Backend, Datenhaltung)**
  - **Eigenständig deploybare Einheiten**

- **Eigenständige Entwicklungsteams**
  - **Komplette Verantwortung für einen Service**
  - **Extremfall: “You build it you run it”**
  - **Freiheiten bei technischen und organisatorischen Entscheidungen**

Microservices & CI



Microservices & Infrastruktur

DO

Entkopplung Infra-Team und Entwicklung

Eigenständige Konfiguration von Build, Test und Deployment

Kapselung der Buildumgebung

Eigenständige Konfiguration der Projektmanagement-Tools

Eigenständige Konfiguration von Staging Umgebungen

Microservices & Infrastruktur

DON’T

Modifikation der Konfiguration des CI Servers für einzelne Projekte

Adminrechte für Entwickler

Langwierige Abstimmungen zwischen Infra-Team und Entwicklung

Globale Workflow-Definitionen im Issue Tracker


Self
Service

Jenkins

Jenkins

- **Verschiedene Wege Jobs zu erstellen:**
  GUI, Job DSL, Pipeline
- **Viele Plugins**
- **Groovy**

- **Bauen von Merge Requests und Branches?**
- **Bauen mit Docker?**

=> möglich, aber manuelle Anpassung notwendig

- **Issue Tracker?**

GitLab

GitLab Features

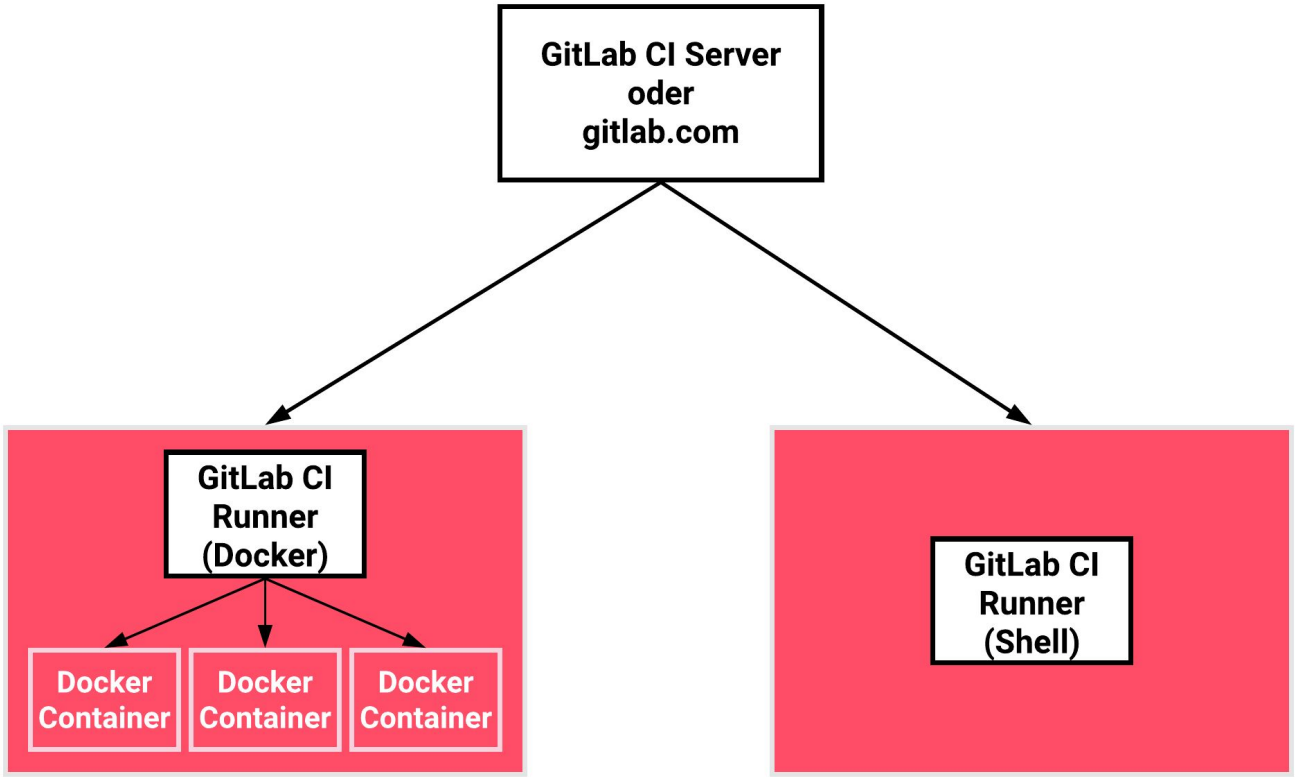
- Issue Tracker integriert
- Bauen von Merge Requests und Branches integriert
- Anlegen von Jobs über eine deklarative Konfiguration mittels Yaml
- Builds innerhalb Docker

=> Sinnvolle Konventionen, modularer Ansatz

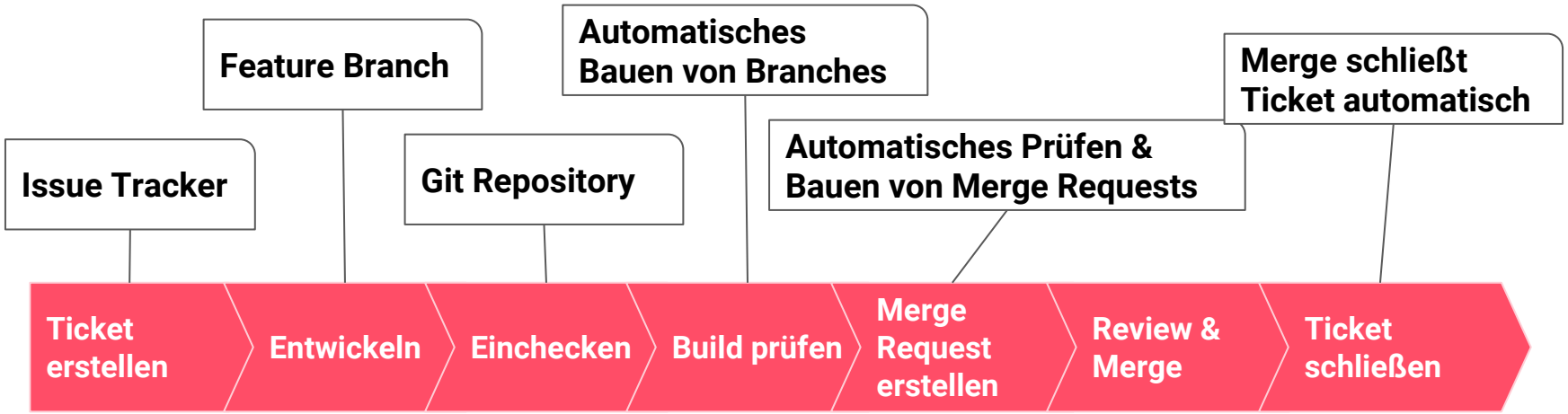
GitLab Übersicht

- Gestartet als webbasierter Git Repository Manager
- CI Pipelines seit 2016
- Mittlerweile: Umfangreiche Softwareentwicklungssuite
  - Epics bis Deployment
- On Premise Edition
  - Libre (ehemals Community Edition CE)
  - Starter
  - Premium (ehemals Enterprise Edition EE)
  - Ultimate
- Cloud Version: gitlab.com

Setup


GitLab CI Server
oder
gitlab.com
GitLab CI
Runner
(Docker)
Docker
Container
Docker
Container
Docker
Container
GitLab CI
Runner
(Shell)

Komponenten Entwicklung



Feature

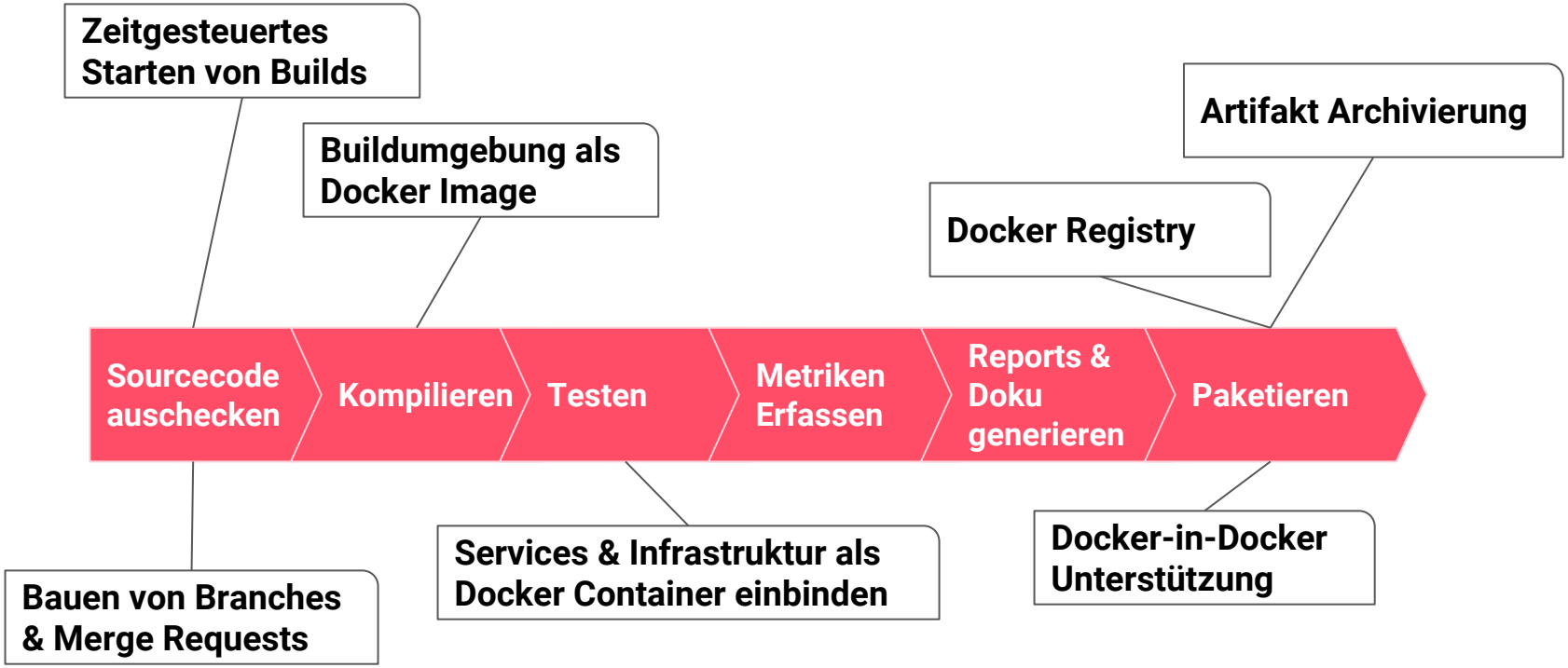
Chat (Mattermost)

Wiki

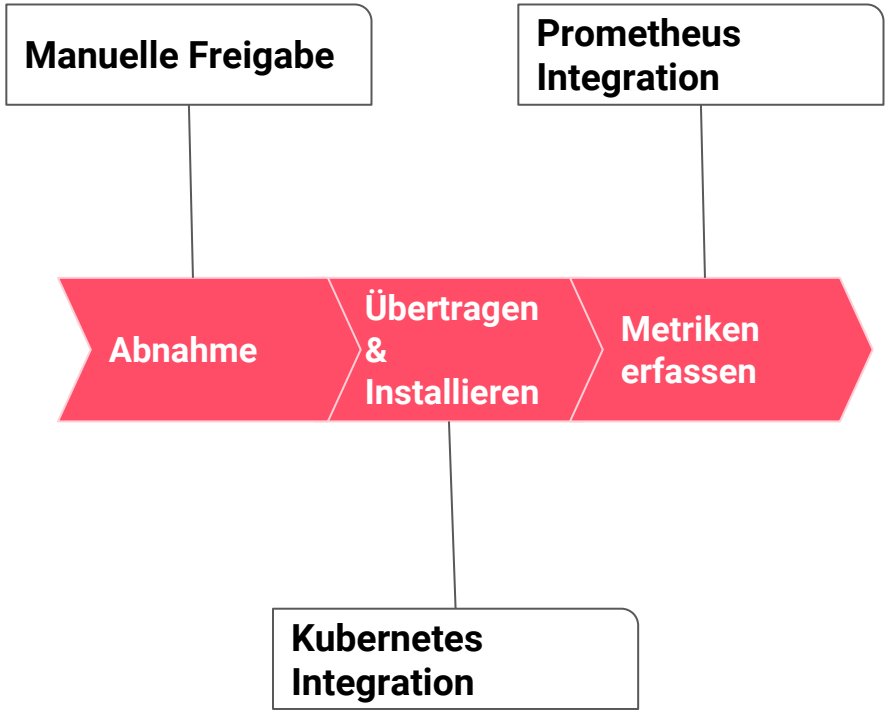
Static Page Hosting

Boards

Komponenten CI

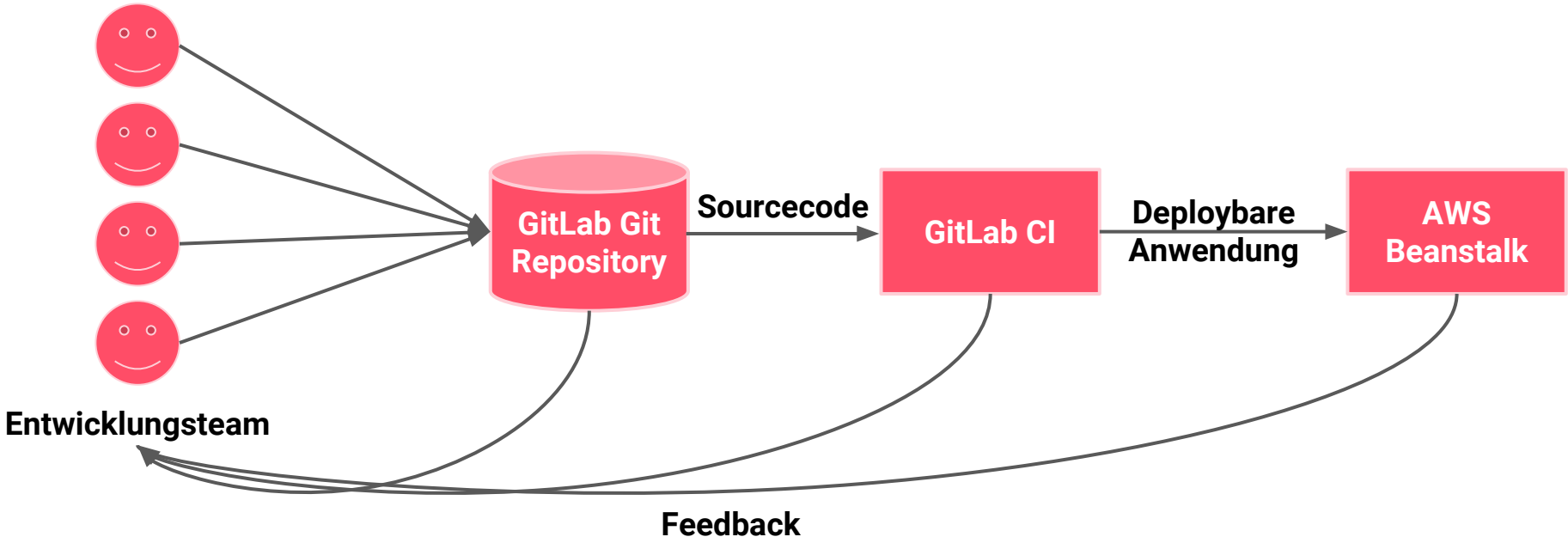


Komponenten Deployment



Beispiel

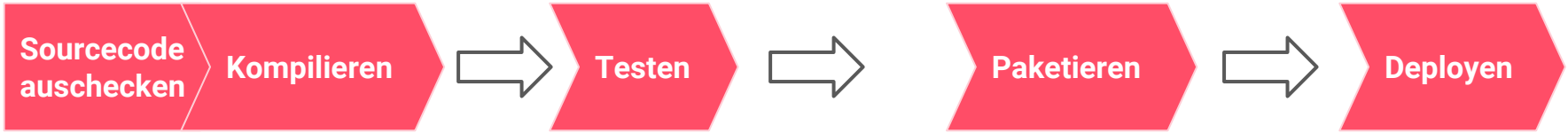
Ziel

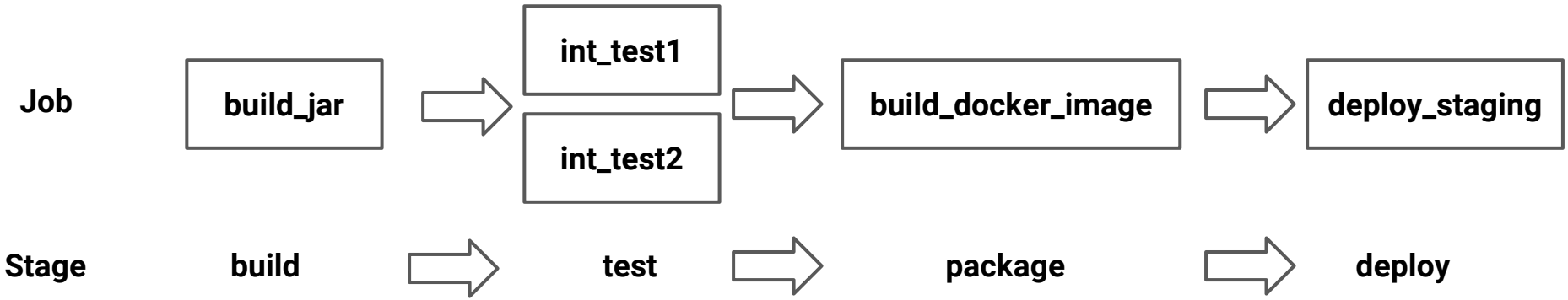

GitLab Git Repository
Sourcecode
GitLab CI
Deploybare Anwendung
AWS Beanstalk
Entwicklungsteam
Feedback

Struktur .gitlab-ci.yml

```
Deklaration der Stages (optional)

Deklaration von Umgebungsvariablen (optional)

Job-Deklaration1 (Stage, Image, Services, Script, Artifacts)

Job-Deklaration2

Job-Deklaration3

. . .
```

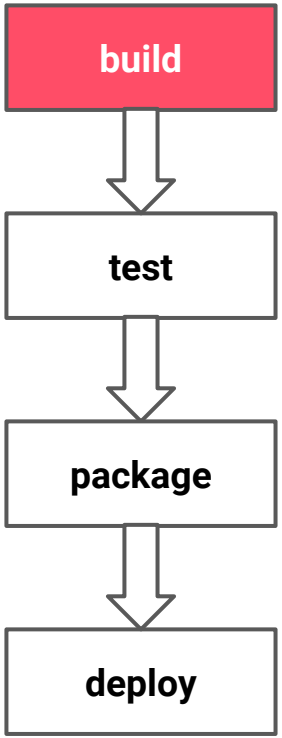
CI Pipeline


Sourcecode auschecken
Kompilieren
Testen
Paketieren
Deployen


Job
build_jar
int_test1
int_test2
build_docker_image
deploy_staging
Stage
build
test
package
deploy

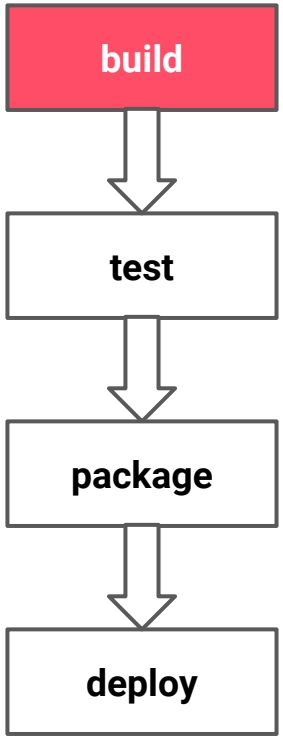
Bau eines Java Projektes

```
stages:
  - build

build_jar:
  stage: build
  image: maven:3-jdk-8
  script:
    - mvn install
  artifacts:
    paths:
      - target/*.jar
```



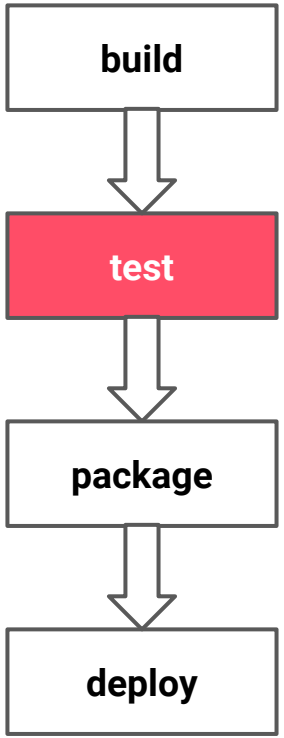
Caching

```
stages:
  - build

variables:
 MAVEN_OPTS: "-Dmaven.repo.local=.m2/repository"

cache:
  paths:
    - .m2/repository

build_jar:
  ...
```



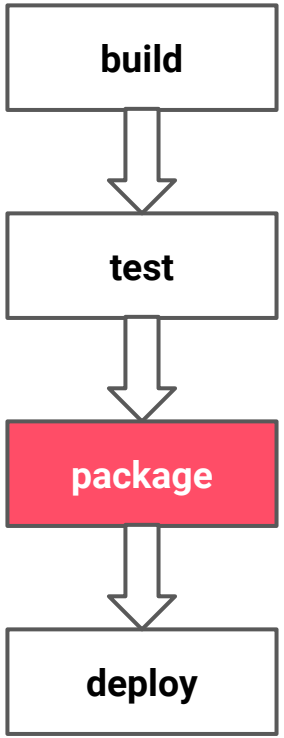
Services einbinden am Beispiel PostgreSQL

```
...
variables:
  POSTGRES_DB: enco
  POSTGRES_USER: postgres
  POSTGRES_PASSWORD: postgres

int_test1:
  stage: test
  image: maven:3-jdk-8
  services:
    - postgres:9.6
  script:
    - mvn install -P integration-test1
  artifacts:
    paths:
      - target/reports/*
```



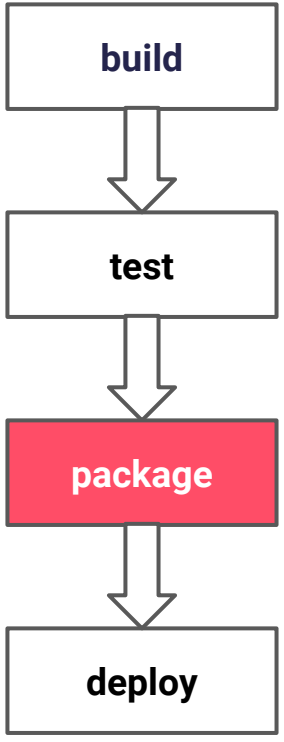
Docker Image bauen & pushen

```
stages:
  - build
  - test
  - package
...

build_docker_image:
  stage: package
  image: docker:latest
  services:
   - docker:dind
  script:
   - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN $CI_REGISTRY
   - docker build -t $CI_REGISTRY_IMAGE:latest .
   - docker push $CI_REGISTRY_IMAGE:latest
```



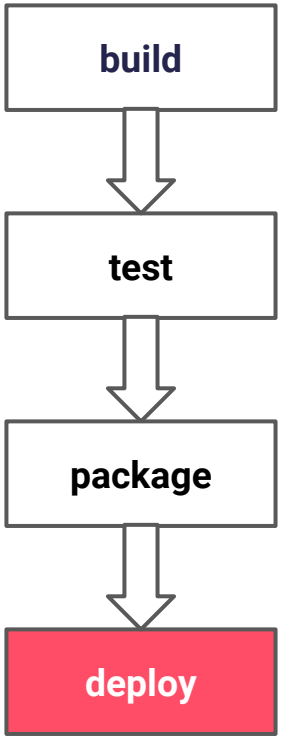
Branch-Spezifische Jobs

```
stages:
  - build
  - test
  - package
...

build_docker_image:
  stage: package
  image: docker:latest
  services:
   - docker:dind
  script:
   - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN $CI_REGISTRY
   - docker build -t $CI_REGISTRY_IMAGE:latest .
   - docker push $CI_REGISTRY_IMAGE:latest
 only:
    - master
```



Deployment bei AWS Beanstalk

```
deploy_staging:
  ...
  script:
    - aws elasticbeanstalk create-application-version \
        --application-name "$EB_APP_NAME" \
        --version-label "$CI_COMMIT_SHA" \
        --description "$CI_COMMIT_SHA" \
        --source-bundle S3Bucket="$S3_BUCKET", \
                        S3Key="$CI_COMMIT_SHA.zip"
    - aws elasticbeanstalk update-environment \
        --application-name "$EB_APP_NAME" \
        --environment-name $EB_ENVIRONMENT \
        --version-label "$CI_COMMIT_SHA"
  environment:
    name: staging
    url: $DEPLOYMENT_URL
```



Demo

Fragen?