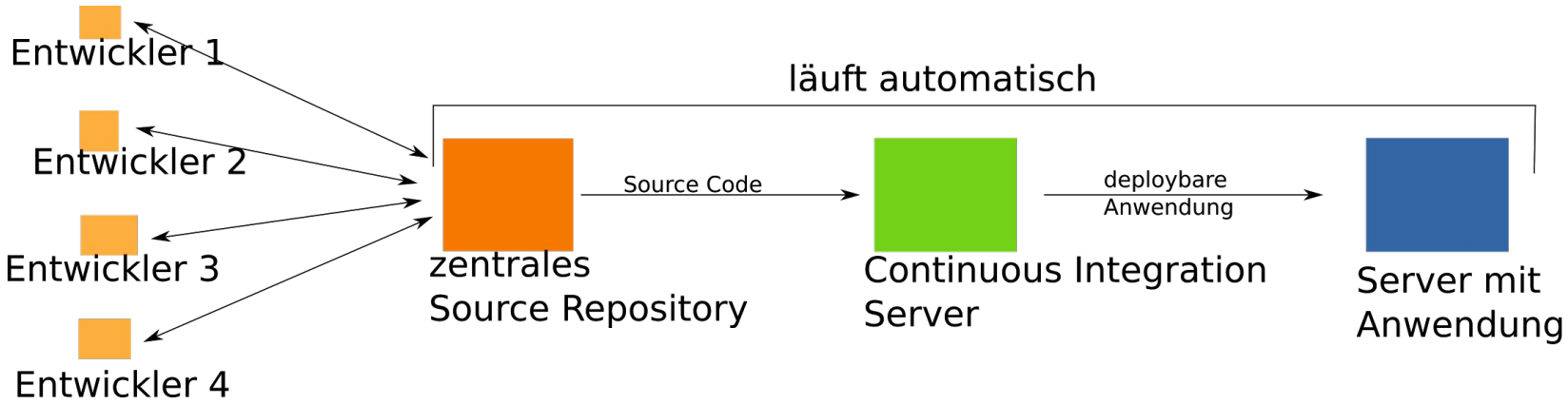
Continuous Integration mit GitLab CI

Christine Koppelt
christine.koppelt@gmail.com
Open-Source-Treffen, 26. Januar 2018

Continuous Integration

Softwareentwicklung & Deployment



Aufgaben eines Continuous Integration Servers

- **Kompilieren und Paketieren**
- **Testen, Testen, Testen**
- **Qualitätssicherung, Reports, Dokumentation**
- **Archivierung/Deployment**
- **Versand von Benachrichtigungen über Fehlschläge**
- **Nachvollziehbarkeit und Reproduzierbarkeit der Builds**

Rahmenbedingungen in 2018

- Mehrere Microservices ggf. mit unterschiedlichen Plattformen realisiert
  - ggf. unterschiedliche Buildumgebungen
  - Tests benötigen ggf. Mehrere Umsysteme wie Datenbanken oder Messaging Systeme
  - … oder weitere Services
- Kurze Time-To-Market gewünscht
- Docker-basiertes Deployment
- Arbeiten mit Pull-/Merge-Requests
- Schnelles Testen von Features (z.B. mittels A/B Testing)
- Effizientes Managen von Abhängigkeiten

Anforderungen an einen CI Server

- **Self-Service: Teams sollen autonom ihre Buildumgebung konfigurieren**
  - **Entkopplung von Betrieb und Entwicklung**
  - **Schnellere Entwicklung**
  - **Kapselung der Buildumgebung**
- **Admin Team stellt Buildserver bereit**
  - **Befasst sich aber nicht mit der Buildkonfiguration einzelner Services**
  - **Keine Modifikation der Konfiguration des CI Servers für einzelne Projekte**
  - **Keine globalen Administrationsrechte für Entwickler**
  - **Keine strikten Prozessvorgaben für Build, Test und Deployment**
  - **Erwartet kontrollierte, gleichartige Builds**

Anforderungen an Services/Anwendungen

- **Separates Deployment möglich**
- **Automate everything**
  - **Skripte mit Parametern oder Umgebungsvariablen für Konfiguration**
- **Version Control everything**
- **Repeatable Reliable Process (z.B. Testausführung)**

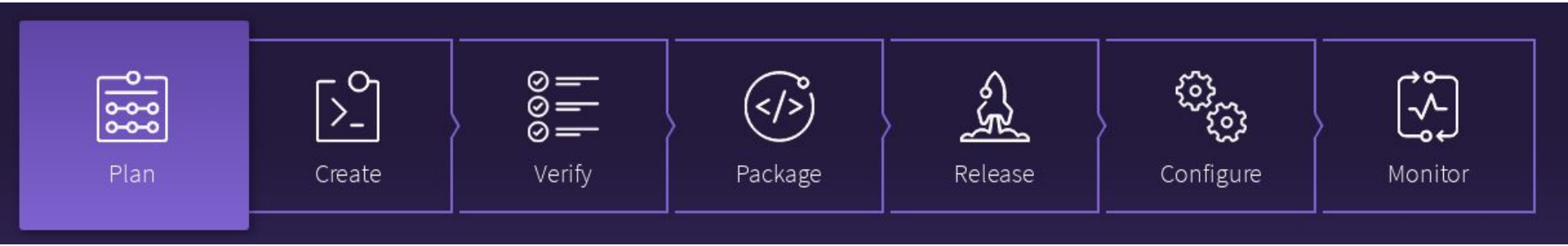
GitLab CI

GitLab

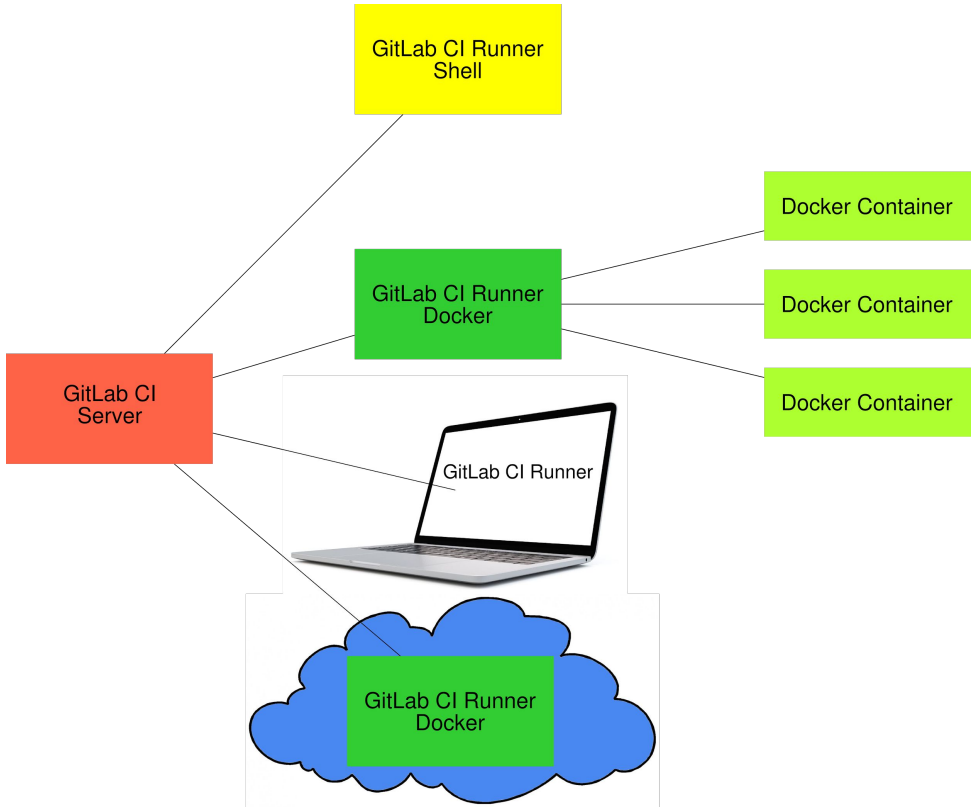
- Gestartet als Web-Basierter Git Repository Manager
- CI Pipelines seit 2016
- Mittlerweile: umfangreiche Softwareentwicklungssuite
- On Premise Edition
  - Community Edition (Open Source/free)
  - Enterprise
  - Ultimate
- Cloud Version: gitlab.com
- Geschrieben in Ruby

Komponenten

- Web-basierter Git-Repository Manager
  - Merge Requests (Code Quality Checks, Comments)
- Wiki, Chat (Mattermost), Issue Tracker, Boards
- Static Page Hosting
- Docker Registry
- API
- Kubernetes Integration



Komponenten/Architektur

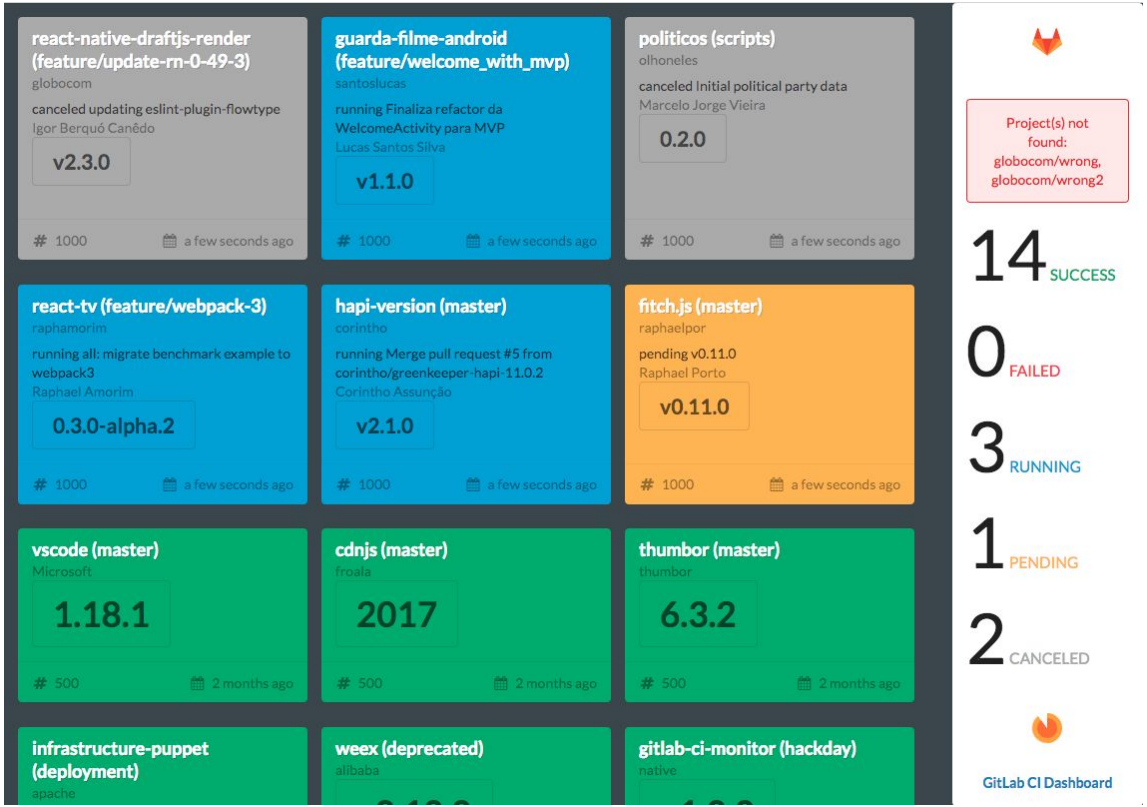


GitLab CI - Funktionen

- Funktioniert nur mit GitLab
- Deklarative Konfiguration von Pipelines & Jobs
- Builds: Docker oder Shell-basiert
- Zeitgesteuertes Starten von Builds
- Branch-spezifische Jobs
- Deployment zu Kubernetes

Demo

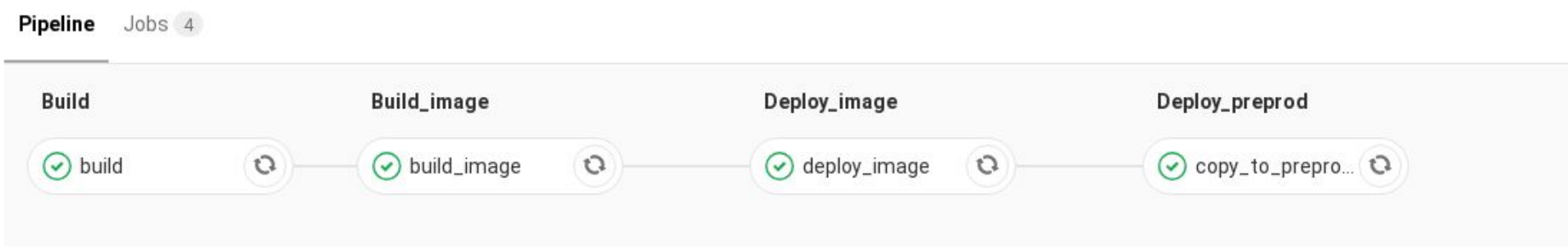
Erweiterungstools



Erstellen von Pipelines

Erstellen einer Pipeline

- Aktivieren in Konfiguration
- Yml Datei erstellen
  - Definiert Stages der Pipeline
  - Definiert Jobs
  - Jobs werden Stages zugeordnet - Mehrfachzuordnung möglich



Struktur .gitlab-ci.yml

Deklaration der Stages

Deklaration von Umgebungsvariablen

Job-Deklarationen

- Stage
- Image
- Services
- Script
- Artifacts
- Tags

Bau eines Java Projektes

```
stages:
  - build

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

PostgreSQL Datenbank mit einbinden

```
stages:
  - build

variables:
  POSTGRES_DB: enco
  POSTGRES_USER: postgres
  POSTGRES_PASSWORD: postgres

build_jar:
  stage: build
  image: maven:3-jdk-8
  services:
    - postgres:9.6
  script:
    - mvn install
  artifacts:
    paths:
      - server/target/*.jar
  tags:
    - docker
```

Docker Container bauen & pushen

```
stages:
  - build
  - build_image

build_jar:
...

build_image:
  stage: build_image
  image: docker:latest
  services:
   - docker:dind
  script:
   - docker login -u gitlab-ci-token -p $CI_BUILD_TOKEN $CI_REGISTRY
   - docker build -t registry.gitlab.com/cko/ci-demo .
   - docker push registry.gitlab.com/cko/ci-demo
  dependencies:
    - build_jar
  when: on_success
  tags:
    - docker
```

Caching

```
stages:
  - build

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

cache:
  paths:
    - .m2/repository

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

Branchspezifische Builds

```
build_image:
  stage: build_image
  image: docker:latest
  services:
   - docker:dind
  script:
   - docker login -u gitlab-ci-token -p $CI_BUILD_TOKEN $CI_REGISTRY
   - docker build -t registry.gitlab.com/cko/ci-demo .
   - docker push registry.gitlab.com/cko/ci-demo
 only:
    - master
    - preprod
  dependencies:
    - build_jar
  when: on_success
  tags:
    - docker
```

Fragen?