



Techtalk

Tobias Wessels

Einführung

Continuous Integration

Continuous Integration

..beschreibt das fortlaufende Zusammenfügen von Komponenten zu einer Anwendung mit dem Ziel die Softwarequalität zu erhöhen.

Continuous Integration

..beschreibt das **fortlaufende Zusammenfügen** von Komponenten zu einer Anwendung mit dem Ziel die Softwarequalität zu erhöhen.

Continuous Integration

..beschreibt das fortlaufende Zusammenfügen von Komponenten zu einer Anwendung mit dem Ziel die Softwarequalität zu erhöhen.

Continuous Deployment

Continuous Deployment

..bezeichnet das automatische Einspielen von Softwareversionen auf Entwicklungs-, Test-, Integrations- und Produktivumgebung

Continuous Deployment

..bezeichnet das automatische Einspielen von Softwareversionen auf Entwicklungs-, Test-, Integrations- und Produktivumgebung

Keine Silver Bullets – außer Continuous Delivery?

- Die Wahrscheinlichkeit, dass Deployments fehlschlagen, ist siebenfach geringer.
- 2604fach schnelleres Wiederanlaufen nach dem Ausfall eines Services.
- 2/3 mehr Arbeitszeit kann für neue Features aufgewendet werden.
- Halb so viel Arbeitszeit ist für Sicherheitsprobleme oder Defekte aufzuwenden, die von Endbenutzern gemeldet werden.
- Man braucht nur ein Drittel der Zeit für Kunden-Support.

2018 State of DevOps Report

Klassische CI Tools

- Hudson/Jenkins
- TeamCity
- CruiseControl
- UVW.



Neue Welle

- TravisCI
- CircleCI
- GitLab CI/CD



Neue Welle

- TravisCI
- CircleCI
- GitLab CI/CD



Speichere im Repository
wie Modul gebaut und deployed werden soll

GitLab CI/CD

"Starting from version 8.0, GitLab Continuous Integration (CI) is fully integrated into GitLab itself and is enabled by default on all projects."

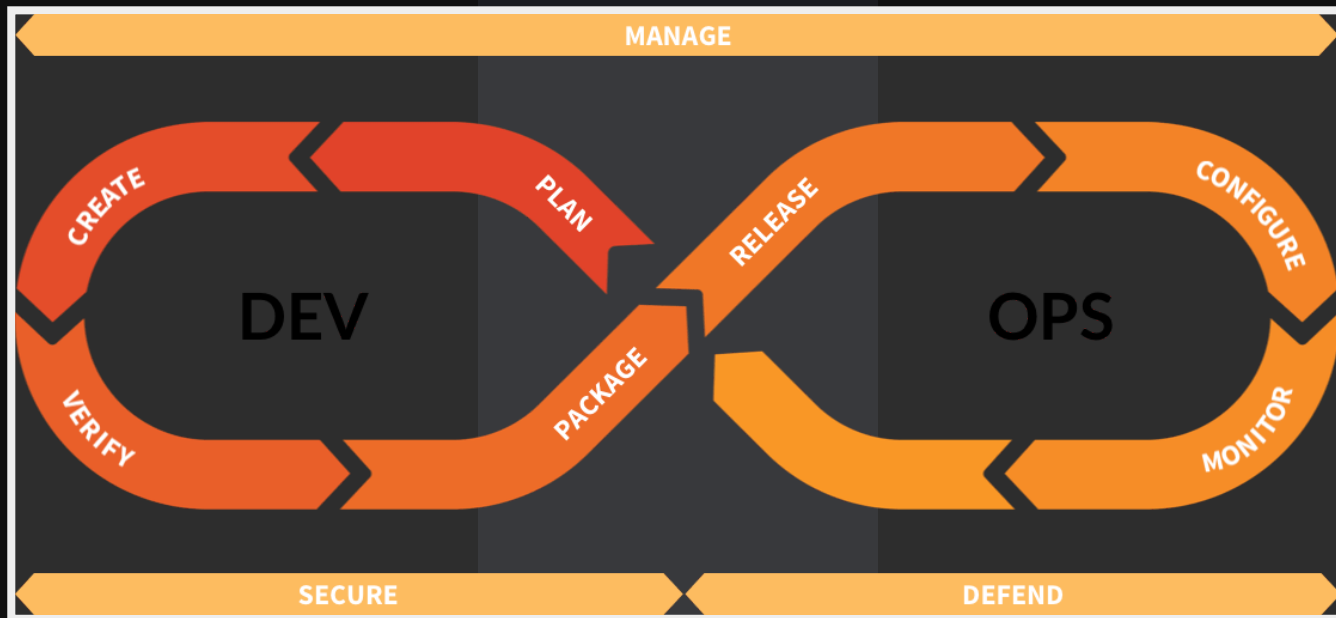
GitLab CE

- CI/CD ist ein Kernstück von GitLab
- Kann auch auf GitHub verwendet werden
- Open Source

Features

- Application Performance Monitoring
- Alerting
- Built-in Container Registry
- Built for using containers and Docker
- Cloud Native
- Code Quality Reports











Auto DevOps



Auto DevOps

1. Auto Build
2. Auto Test
3. Auto Code Quality ⓘ
4. Auto SAST (Static Application Security Testing) ⓘ
5. Auto Dependency Scanning ⓘ
6. Auto License Compliance ⓘ
7. Auto Container Scanning ⓘ
8. Auto Review Apps
9. Auto DAST (Dynamic Application Security Testing) ⓘ
10. Auto Deploy
11. Auto Browser Performance Testing ⓘ
12. Auto Monitoring

gitlab-ci.yml

Name	Last commit	Last update
 .settings	#1 Removed complex order logic (order of shipment and delivery status) ...	2 years ago
 src	Added cross-origin support	2 years ago
 .classpath	Initial commit, including order and line item domain objects	2 years ago
 .gitignore	Initial commit	2 years ago
 .gitlab-ci.yml	Without nohup	1 hour ago
 .project	#1 Removed complex order logic (order of shipment and delivery status) ...	2 years ago
 .travis.yml	Renamed YML file to have the correct extension	2 years ago
 Dockerfile	Added Dockerfile	4 hours ago
 README.md	Update README.md fixing typo	1 year ago
 pom.xml	Added basic structure for automated acceptance tests	2 years ago



*“Ein edles Beispiel macht die
schweren Taten leicht.”*

Simple Beispiel

```
image: maven:3-jdk-8
```

```
build:
```

```
  stage: build
```

```
  script: mvn compile
```

```
test:
```

```
  stage: test
```

```
  script: mvn test
```

```
deploy:
```

```
  stage: deploy
```

```
  script: mvn deploy
```

```
only:
```

```
  - master
```

*Auch branches sollten gebaut werden,
wenn etwas eingecheckt wird!*

Artefakte persitieren

```
maven-build:
  stage: build_jar
  script:
    - echo "Building the app"
    - mvn package -B
  artifacts:
    paths:
      - target/*
    expire_in: 5min
```


Docker Images bauen

```
docker-build:  
  stage: build_container  
  image: docker:stable  
  script:  
    - mv target/*.jar my-webapp.jar  
    - docker build -t my-docker-image .
```

Deploy

```
deploy:
  stage: deploy
  image: docker:stable
  script:
    - echo "Deploying endpoint"
    - docker run -d -p 8080:8080 my-docker-image
only:
- master
```

Rules

```
docker build:
```

```
  script: docker build -t my-image:$CI_COMMIT_REF_SLUG .
```

```
  rules:
```

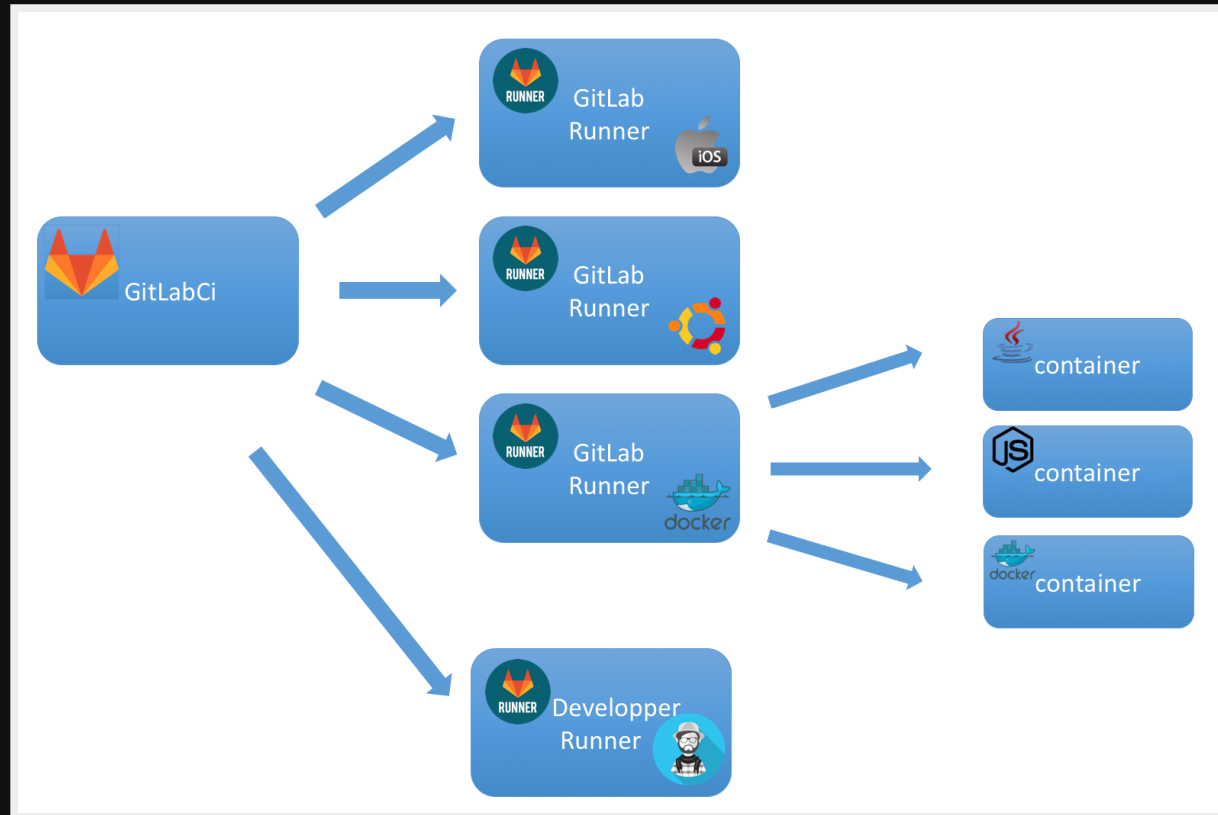
```
    - changes:
```

```
      - Dockerfile
```

Pipeline Architektur

[pipeline_architectures.html](https://www.coursera.org/lecture/parallel-processing/pipeline-architectures-4)

Runners



Shell Runner

```
1 sudo gitlab-runner register -n \  
2   --url https://gitlab.com/ \  
3   --registration-token REGISTRATION_TOKEN \  
4   --executor shell \  
5   --description "My Runner"
```

Docker Runner

```
1 sudo gitlab-runner register -n \  
2   --url https://gitlab.com/ \  
3   --registration-token REGISTRATION_TOKEN \  
4   --executor docker \  
5   --description "My Docker Runner" \  
6   --docker-image "docker:19.03.8" \  
7   --docker-privileged \  
8   --docker-volumes "/certs/client"
```

Docker Runner

- Polyglot Pipelines
- Test- und Livesystem müssen gar nicht mehr so verschieden sein

Demonstration

GitLab CI/CD Installation

<http://gitlabtechtalk.ddns.net/>

Fazit

Überflüssig?

- Jenkins
- Nexus Repository Manager
- SonarQube
- Alerting / Monitoring VM

Vorteile gegenüber Jenkins

- Templating
- Build- und Deployanweisungen im Repository
- Autoscaling for Runners
- Docker Support
- DAG Pipeline verkürzt Bauzeit
- Rules Keyword: Baue nur wenn wirklich nötig

A close-up, slightly out-of-focus photograph of a very fluffy white dog, possibly a Samoyed, looking directly at the camera. The dog's fur is thick and white, with its dark eyes and black nose visible. Several blue, stylized question marks are floating around the dog's head, suggesting confusion or a lack of understanding. In the background, a glass of red wine and some indistinct objects are visible on a table.

... Fragen?