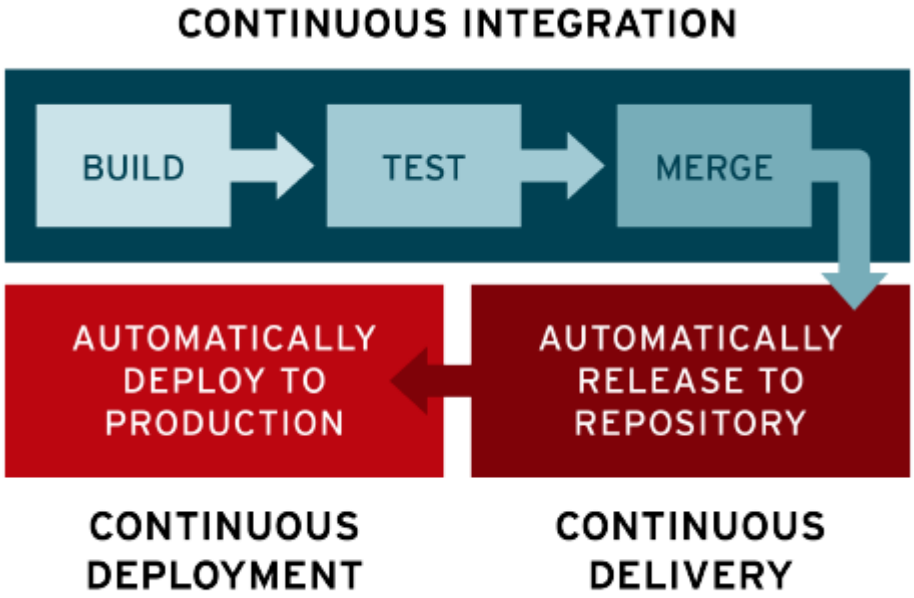
Cheat Sheet CI/CD

Was ist CI/CD?

- Automatisieren von häufigen Tasks, wie Bauen, Testen, Deployen, …



GitHub Actions – Dokumentation

https://docs.github.com/en/actions

GitHub Actions - Begriffe

- **Workflow:**
  - Größte Einheit in Actions
  - Ablauf von mehreren Jobs wie Bauen und Testen einer Anwendung
  - Jede YAML-Datei in ".github/workflows/" wir versucht als Workflow interpretiert zu werden
- **Jobs:**
  - Teile eines Workflows (meist für je ein Thema wie Bauen, Testen, Linter, Deployen, …)
  - Ablauf von mehreren Schritten
  - Laufen standardmäßig parallel
- **Steps/Schritte:**
  - Teile eines Jobs
  - Ablauf von Befehlen wie "echo Hello World!"

GitHub Actions – Syntax (Teil 1)

name: Name eines Workflows/Jobs/Steps

on: Trigger, wann der Workflow ausgeführt werden soll

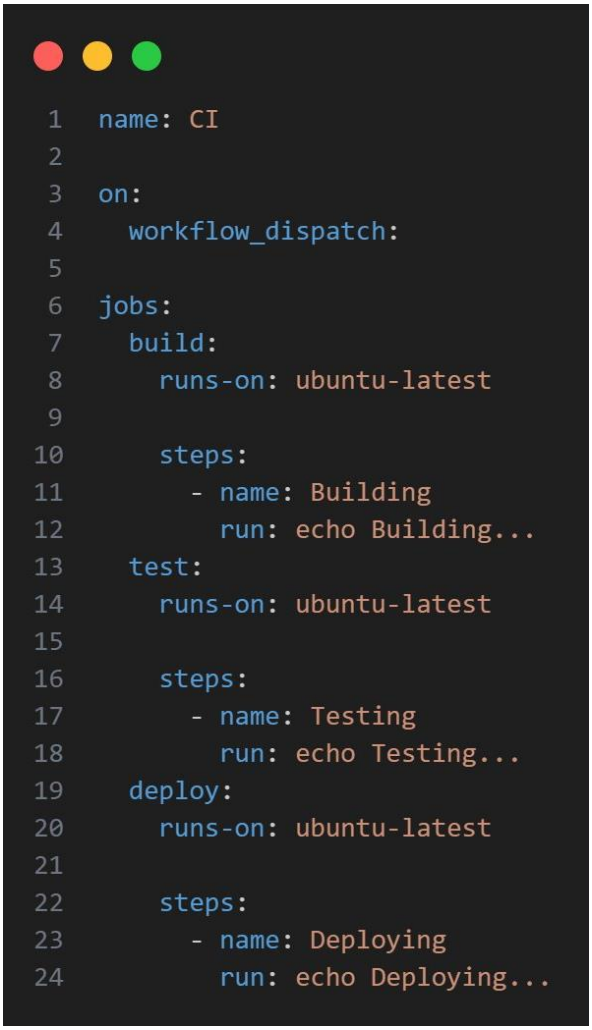
- **workflow_dispatch:** Manueller Start
- **push:** Bei einem Push (Optional: branches: <Name>)
- **pull_request:** Bei Erstellung und Aktualisierung eines Pull-Request
- **schedule:** zeitlich gesteuerter Start

jobs: Startet die Aufzählung an Job-Definitionen

steps: Startet die Aufzählung an Step-Definitionen

runs-on: Gibt auf Job-Ebene an, auf welcher Umgebung ein Job laufen soll (bei uns einfachheitshalber immer ubuntu-latest)

run: Führt einzelne Befehle aus



```yaml
name: CI

on:
  workflow_dispatch:

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Building
        run: echo Building...
  test:
    runs-on: ubuntu-latest

    steps:
      - name: Testing
        run: echo Testing...
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Deploying
        run: echo Deploying...
```

GitHub Actions – Implementation anderer Workflows

- Man kann andere Workflows als Step in seinem Eigenen ausführen lassen
- Es gibt einen GitHub Actions Marketplace (https://github.com/marketplace?type=actions) mit vorgefertigten Workflows
- Wichtige Workflows:
    - **Checkout** - Lädt den Code des Repositories in die Workflowumgebung, dass das Projekt z.B. gebaut und getestet werden kann

```yaml
steps:
      - name: Checkout repository
        uses: actions/checkout@v4
```

    - **Setup-Java** – Richtet die angegebene JDK ein (Parameter mit “with:”)

```yaml
steps:
      - name: Set up JDK 21
        uses: actions/setup-java@v4
        with:
          distribution: 'temurin'
          java-version: '21'
```

    - **Upload-Artifacts** – Hochladen von Dateien, sodass sie außerhalb eines Jobs oder sogar der Workflows verfügbar sind

```yaml
steps:
      - name: Archive test results
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: junit-results
          path: target/surefire-reports/
```

GitHub Actions – Syntax (Teil 2)

with: Bei einem integrierten Workflow: Angabe der Parameter

needs: Einstellen, dass für Job-Start erst ein anderer abgeschlossen sein muss

if: always() Einstellen, dass Job, auch bei fehlerhaftem Workflow, ausgeführt wird

env: Anlegen von Variablen auf Workflow-, Job- und Step-Ebene

$VARIABLEN_NAME Variablen Wert erfragen

strategy: Sorgt dafür, dass ein Job mit mehreren Konfigurationen ausgeführt wird

matrix:

KONFIGURATIONS_VARIABLEN_NAME

${{ matrix.NAME}} (Konfigurations-)Variablen Wert erfragen

```yaml
example:
    runs-on: ubuntu-latest
    needs: other-job
    if: always()
    env:
      Test: test
    strategy:
      matrix:
        version: [1, 2]

    steps:
      - name: Example Step
        run: echo §Test ${{ matrix.version }}
```

GitHub Actions – Weitere evtl. benötigte Features

- **Inputs:** Zum Auswählen von Werten vor Start eines Workflows, um z.B. zu steuern welche Umgebung genommen werden soll
- **Environments:** Umgebungen, um z.B. auf unterschiedliche Server zu deployen => eine gleichnamige Variable in jeder Umgebung mit verschiedenen Werten
- **Ausdrücke in ${{ ... }}:** Ausdrücke wie ${{ matrix.version == 21 }} sind möglich

Maven Befehle

- Bauen und testen: **mvn –B clean verify**
- Lint (Failure bei Funden) **mvn checkstyle:check**
- Lint (Funde “ignoriert”) **mvn checkstyle:checkstyle**