Github Actions - Cheat Sheet

Grundlegende Begriffe

Was ist GitHub Actions?

GitHub Actions ist eine Plattform zur Automatisierung von Softwareentwicklungs-Workflows direkt in einem GitHub-Repository. Es wird oft für Continuous Integration (CI) und Continuous Deployment (CD) genutzt.

Schlüsselbegriffe

- **Workflow:** Eine YAML-Datei, die eine Serie von Automatisierungsschritten beschreibt.
- **Job:** Eine Reihe von Schritten, die auf derselben Maschine ausgeführt werden.
- **Step:** Ein einzelner Task innerhalb eines Jobs (z. B. ein Skript oder eine Aktion).
- **Action:** Wiederverwendbarer Code, der in einem Workflow verwendet werden kann (z. B. Plugins oder vorgefertigte Aufgaben).
- **Runner:** Eine Maschine, die einen Job ausführt (GitHub-hosted oder self-hosted).
- **Events:** Ereignisse wie `push`, `pull_request` oder zeitgesteuerte Trigger, die Workflows starten.

Aufbau von Workflows

Grundstruktur eines Workflows

Workflows werden in YAML-Dateien im Verzeichnis `.github/workflows/` gespeichert.

```yaml
name: CI Workflow

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v3
```

```
    - name: Set up Java
      uses: actions/setup-java@v3
      with:
        java-version: '11'

    - name: Build with Maven
      run: mvn install
```

Trigger-Optionen (Events)

- **push:** Startet den Workflow bei jedem Push.
- **pull_request:** Startet den Workflow bei Pull-Requests.
- **schedule:** Zeitgesteuerte Ausführung (Cron-Syntax).
- **workflow_dispatch:** Manuelles Starten durch Benutzer.

Beispiel für `schedule`:

```
on:
  schedule:
    - cron: "0 0 * * *" # Täglich um Mitternacht
```

Jobs und Steps

- **Job:** Eine Sammlung von Steps. Alle Steps in einem Job werden auf derselben Runner-Instanz ausgeführt.
- **Step:** Ein Einzelschritt, entweder ein Kommando oder eine Aktion.

```
jobs:
  example-job:
    runs-on: ubuntu-latest
    steps:
      - name: Ein Kommando ausführen
        run: echo "Hallo, Welt!"
      - name: Eine Aktion verwenden
        uses: actions/checkout@v3
```

Mehrere Jobs

Jobs können parallel ausgeführt werden oder voneinander abhängen.

```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
```

```
      - run: echo "Building..."

  test:
    runs-on: ubuntu-latest
    needs: build
    steps:
      - run: echo "Testing..."
```

Nutzung von Workflows im Java-Umfeld

Java-Setup mit GitHub Actions

1. **Java-Umgebung einrichten**:
   Verwende die Aktion `actions/setup-java`:

   ```
   - name: Set up Java
     uses: actions/setup-java@v3
     with:
       java-version: '11' # Unterstützte Versionen: 8, 11, 17, etc.
   ```

2. **Maven- oder Gradle-Builds ausführen**:

   ```
   - name: Build with Maven
     run: mvn install
   ```

   ```
   - name: Build with Gradle
     run: ./gradlew build
   ```

Beispiel: CI Workflow für eine Java-Anwendung

```
name: Java CI

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
```

```yaml
    - name: Checkout code
      uses: actions/checkout@v3

    - name: Set up Java
      uses: actions/setup-java@v3
      with:
        java-version: '11'

    - name: Build with Maven
      run: mvn install

    - name: Run tests
      run: mvn test
```

Verwendung von Cache für Abhängigkeiten

Caching kann den Build-Prozess beschleunigen:

```yaml
- name: Cache Maven dependencies
  uses: actions/cache@v3
  with:
    path: ~/.m2/repository
    key: ${{ runner.os }}-maven-${{ hashFiles('**/pom.xml') }}
    restore-keys: |
      ${{ runner.os }}-maven-

- name: Build with Maven
  run: mvn install
```

Nützliche Links

- GitHub Actions Dokumentation
- Marketplace für Aktionen
- Beispiele und Best Practices