Einführung in GitHub Actions

Mit Schwerpunkt auf CI/CD

1. Einleitung

- **Definition von GitHub Actions:**
  - GitHub Actions ist eine integrierte Funktion von GitHub, die Automatisierung in den Entwicklungsworkflow ermöglicht.
  - Mit GitHub Actions können Entwickler Workflows erstellen, um **verschiedene Aktionen automatisch auszuführen**.
- **Bedeutung von Automatisierung im Entwicklungsprozess:**
  - Automatisierung spielt eine entscheidende Rolle in der Effizienz des Entwicklungsprozesses.
  - GitHub Actions ermöglicht die **Automatisierung von wiederholbaren Aufgaben**, was Zeit spart und die Zuverlässigkeit des Codes verbessert.
- **Integration von GitHub Actions in den Entwicklungsworkflow:**
  - GitHub Actions ist nahtlos in GitHub integriert, wodurch Entwickler ihre Workflows direkt im Repository konfigurieren können.

2. Warum GitHub Actions verwenden?

- **Automatisierung von Workflows:**
  - Minimierung menschlicher Fehler und Erhöhung der Geschwindigkeit der Softwareentwicklung wird.
- **Kontinuierliche Integration (CI) und Kontinuierliche Bereitstellung (CD):**
  - GitHub Actions erleichtert die Implementierung von CI/CD, indem es automatisch:
    - Code überprüft
    - Tests durchführt
    - Anwendungen nach erfolgreichen Tests bereitstellt

3. Grundlegende Konzepte

- **Workflow:**
  - Ein Workflow ist eine automatisierte Abfolge von Aktionen, die in einem Repository ausgeführt wird.
  - Workflows werden in einer **YAML-Datei** definiert und können auf verschiedene Ereignisse reagieren.
- **Aktionen (Actions):**
  - Aktionen sind einzelne Aufgaben, die im Workflow ausgeführt werden.
  - GitHub Actions bietet **vordefinierte Aktionen**, und Entwickler können auch ihre **eigenen Aktionen** erstellen und verwenden.
- **Ereignisse (Events):**
  - Workflows werden durch **bestimmte Ereignisse ausgelöst**, wie z.B. das Pushen von Code, das Erstellen eines Pull Requests oder das Auslösen eines Zeitplans.

4. CI/CD mit GitHub Actions

- **CI: Kontinuierliche Integration**
  - Automatisches Testen des Codes bei jeder Änderung, um frühzeitig Fehler zu erkennen und die Codequalität sicherzustellen.
- **CD: Kontinuierliche Bereitstellung**
  - Automatisches Bereitstellen von Anwendungen nach erfolgreichem CI, um einen schnellen und zuverlässigen Veröffentlichungsprozess zu gewährleisten.

5. Beispiel eines einfachen CI/CD-Workflows

```yaml
name: GitHub Actions Demo
run-name: ${{ github.actor }} is testing out GitHub Actions 🚀
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "🎉 The job was automatically triggered by a ${{ github.event_name }} event."
      - name: Check out repository code
        uses: actions/checkout@v4
      - name: List files in the repository
        run: |
          ls ${{ github.workspace }}
```

6. Praktische Anwendungsszenarien

- **Tests automatisieren:**
  - Nutzung von GitHub Actions, um automatisch Tests durchzuführen und die Codequalität zu gewährleisten.
- **Automatische Bereitstellung von Anwendungen:**
  - Implementierung von Workflows für die automatische Bereitstellung von Anwendungen nach erfolgreichen Tests.
- **Benachrichtigungen und Kommunikation automatisieren:**
  - Verwendung von GitHub Actions, um automatische Benachrichtigungen oder Kommunikationsprozesse in Entwicklungsworkflows zu integrieren.

7. Ressourcen für weitere Informationen

- **GitHub Actions Dokumentation:**
  - https://docs.github.com/en/actions
- **GitHub Actions Marketplace:**
  - https://github.com/marketplace?type=actions
- **Beispiele für GitHub Actions:**
  - https://github.com/actions/starter-workflows