Spickzettel: Beispiel-Workflow für GitHub Actions

1. Einfache CI/CD-Pipeline (Build & Test)

```
name: CI/CD Pipeline

on:
  push:
    branches:
      - development
  pull_request:
    branches:
      - development

jobs:
  build-and-test:
    runs-on: ubuntu-latest

    steps:
      - name: Repository auschecken
        uses: actions/checkout@v3

      - name: Node.js einrichten
        uses: actions/setup-node@v3
        with:
          node-version: '16'

      - name: Abhängigkeiten installieren
        run: npm install

      - name: Tests ausführen
        run: npm test

  deploy:
    needs: build-and-test
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/development'
    steps:
      - name: Deploy starten
        run: echo "Deployment wird hier ausgeführt..."
```

2. Erklärung der Workflow-Struktur

- **`name`** → Name des Workflows.
- **`on`** → Wann der Workflow ausgeführt wird (bei `push` oder `pull_request` auf `main`).
- **`jobs`** → Enthält einzelne Aufgaben (`build-and-test` & `deploy`).
- **`runs-on`** → Betriebssystem für den Job (`ubuntu-latest`).
- **`steps`** → Liste von Befehlen oder vorgefertigten GitHub Actions.
- **`needs: build-and-test`** → Stellt sicher, dass `deploy` erst nach `build-and-test` läuft.
- **`if: github.ref == 'refs/heads/development'`** → Deployment wird nur auf `main` ausgeführt.

Best Practices

- **Nur auf `main` oder bestimmten Branches deployen** (`if`-Bedingungen verwenden).
- **Workflows in `.github/workflows/` gut strukturieren**.
- **Tests immer vor dem Deployment ausführen**, um Fehler frühzeitig zu erkennen.
- **Parallelisierung & Caching nutzen**, um Workflows zu optimieren.