Anleitung GitHub Actions

1	Login www.GitHub.com
2	Template verwenden → Use this Template (Button) https://github.com/mgrum/flask-example-cicd-template
3	Branch erstellen (**Wichtig:** wir wollen nicht auf **main** entwickeln) z.B. **develop-workflows**

Workflow **DEV - Build**

4	Workflow erstellen mit: A. **Actions** Tab drücken im Repository B. Neuer Workflow → Python Application (nützliche Vorlage) C. Datei umbenennen in **dev_build.yml**
5	Name des Workflows anpassen: `name: DEV - Build and Unittest`
6	Bedingung anpassen: Wir wollen dass dieser Dev Workflow für alle Branches außer **main** läuft `...` `on:` `  push:` `  branches-ignore: [ main ]`
7	Code ergänzen: Dieser Befehl wird benötigt, da die Applikation das Package “Cython” nutzt und sonst nicht funktionieren würde. `...` `  - name: Install dependencies` `  run: |` `  python -m pip install --upgrade pip` `  pip install flake8 pytest` `  if [ -f requirements.txt ]; then pip install -r requirements.txt; fi`

<table>
<tr><td></td><td>

```
    - name: Install project
      run: |
        pip install -e .
...
```

Hilfestellung 1: https://pastebin.com/iiGuhr6i</td></tr>
<tr><td>8</td><td>Jobs anpassen:
Wir wollen Build und Test in einzelne Jobs trennen
Dazu muss pytest aus dem Job build raus und ein Job test angelegt werden


```
...
 test:
    needs: [build]
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Set up Python 3.x
      uses: actions/setup-python@v2
      with:
        python-version: '3.x'
    - name: Display Python version
      run: python -c "import sys; print(sys.version)"
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install pytest
        pip install -r requirements.txt
    - name: Install project
      run: |
        pip install -e .
    - name: Test with pytest
      run: |
        pytest
```

</td></tr>
<tr><td>9</td><td>Workflow Datei speichern / Committen</td></tr>
</table>

10	Auf **Actions** Tab drücken und der Workflow sollte nun starten
11	Gelaufenen Workflow öffnen: A. Build sollte erfolgreich sein B. Test sollte einen Fehler werden → 1/8 Unit-Tests müsste fehlgeschlagen sein In der Ausgabe des Workflows sollte man den Fehler finden können: **flaskr/app.py** → **Zeile 16** → `return {"hello": "world"}`
12	Fehler beheben: `return {"hello": "world"}` → `return {"hello": "IWS"}` Datei Speichern / Committen
13	Auf **Actions** Tab drücken und Workflow sollte nun **erfolgreich funktionieren :-)**

Workflow **STAGE - Test**

14	Einen neuen Workflow anlegen: **.github/workflows/stage_test.yml** Ihr könnt den Code von **dev_build.yml** kopieren
15	Name und Bedingung anpassen: `name: STAGE - Matrix Test` `on:` `  pull_request:` `  branches:` `  - main` `...`
16	Test anpassen: Nun soll nicht nur schnell getestet werden ob es überhaupt funktioniert, sondern ein intensiver Test auf mehreren OS und Python Versionen laufen

	`` ... test: needs: [build] runs-on: ubuntu-latest **strategy:** **matrix:** **os: [ubuntu-latest, macos-latest, windows-latest]** **python-version: [3.6, 3.9]** steps: - uses: actions/checkout@v2 **- name: Set up Python ${{ matrix.python-version }}** **uses: actions/setup-python@v2** **with:** **python-version: ${{ matrix.python-version }}** ... ``` Hilfestellung 2: https://pastebin.com/LXCSueL2 Speichern / Committen nicht vergessen
17	Pull Request stellen Nun einen PR von **develop-workflows** gegen **main** stellen
18	Innerhalb des PR kann man nun die Workflows **DEV** und **STAGE** beobachten Alternativ: Auf **Actions** Tab drücken
19	Wenn alle checks erfolgreich waren, kann der PR gemerged und der branch **develop-workflows** gelöscht werden.

Einzelne Lösungen der Workflows:

dev_build.yml: https://pastebin.com/YZBKLk1i

stage_test.yml: https://pastebin.com/hbchPmbw

Nun sollten wir auf dem Stand von der **Musterlösung** sein:
https://github.com/mgrum/flask-example-cicd-solution

Nützliche Links:

GitHub Actions - Workflow Syntax

Github Actions - Supported Software