



# GitHub-Actions

Maximilian Kampl

## Inhaltsverzeichnis

GitHub Actions .....	2
Theorie.....	2
Praxis.....	3
Workflow erstellen .....	3
Workflow in „action“ .....	5
Quellen .....	8

# GitHub Actions

## Theorie

Ist ein Tool, welches das Testen von Code automatisiert. Actions selbst hat ein Feature namens CI/CD was für Continuous Integration und Continuous Delivery/Deployment steht. Dieses Feature würde, nachdem alle Tests durchgelaufen sind den neu geschriebenen Code automatisch auf das Repository pushen.

Zuerst erstellen wir eine YAML-Datei, in welcher die Prozesse: Events, Jobs, Runners, Steps und Actions spezifiziert.

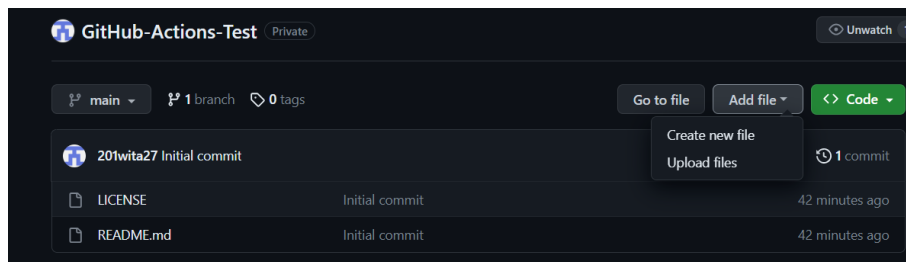
- Event: Trigger für eine Workflow (z.B.: neuer Code wird gepushed)
- Jobs: Aktionen, welche dieser Workflow tätigt, nachdem das Event passiert, ist
- Runner: Umgebung durch welche unser Code ausgeführt wird. (Ubuntu, Windows, MacOS)
- Steps & Actions: In den Steps werden verschiedene Actions ausgeführt, welche unseren Code überprüfen.

Außerdem wird noch ein Linter (meist super-linter) benutzt welcher die Programmiersprache für unseren Workflow lesbar macht.

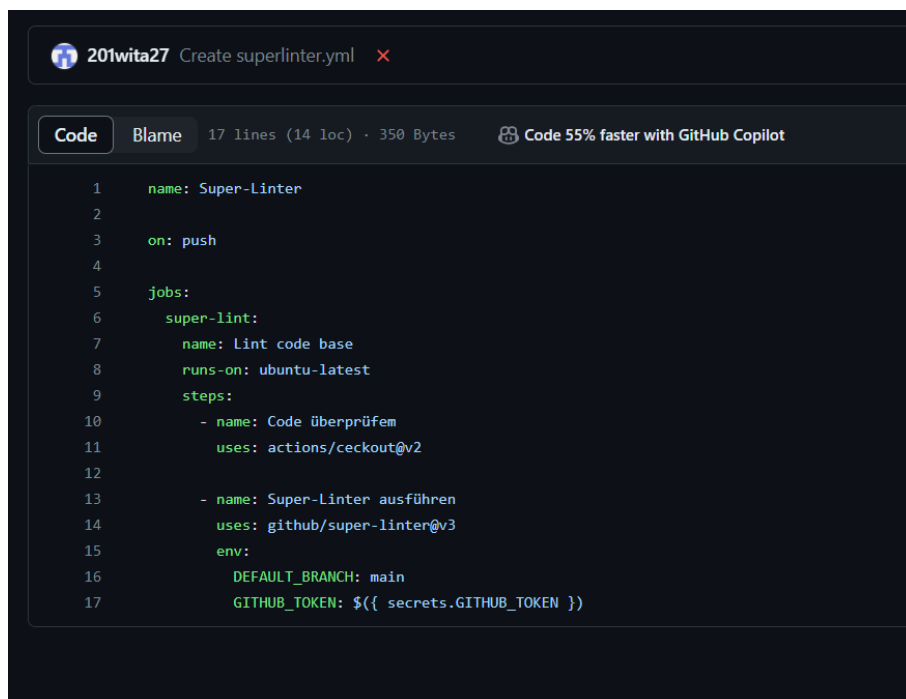
# Praxis

## Workflow erstellen

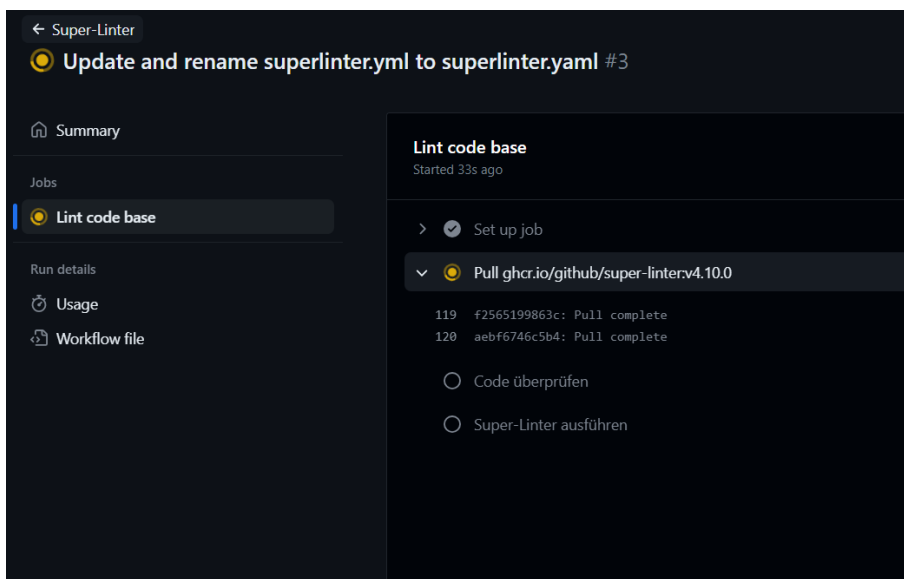
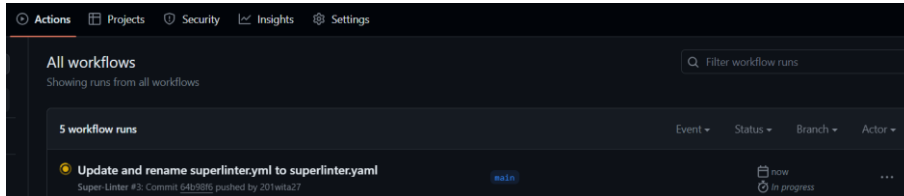
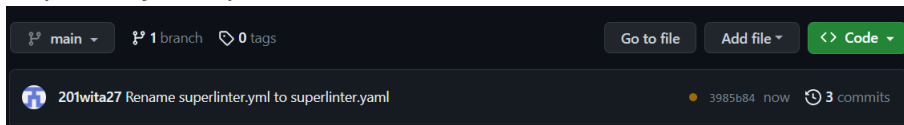
Zuerst erstellen wir eine neue YAML-Datei welche aber im Verzeichnis `.github/workflows` gespeichert werden muss.



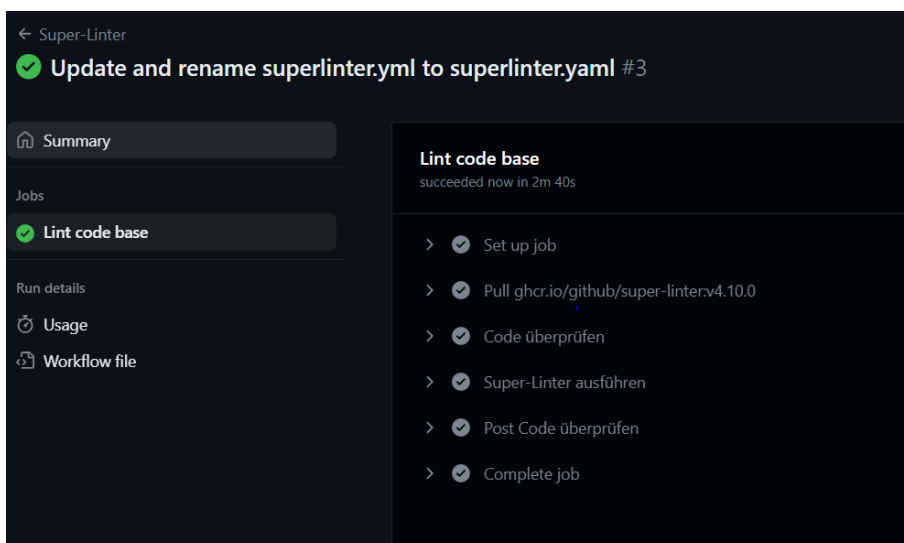
In dieser **YAML-Datei** definieren wir unsere **Workflow**, welchem mithilfe vom **Event on: push** gesagt wird wann dieser abgerufen wird und getestet danach mit den **Steps Code Überprüfen & Super-Linter ausführen**, ob unser Code auch so funktioniert wie er soll. In den beiden **Steps** werden auch noch jeweils **Actions** (`uses: actions/checkout@v2` | `uses: github/super-linter@v4`) definiert welche der **Step** ausführt.



Hier sehen wir unseren erstellten Workflow am Arbeiten wie er gerade unser erstelltes Repository überprüft.

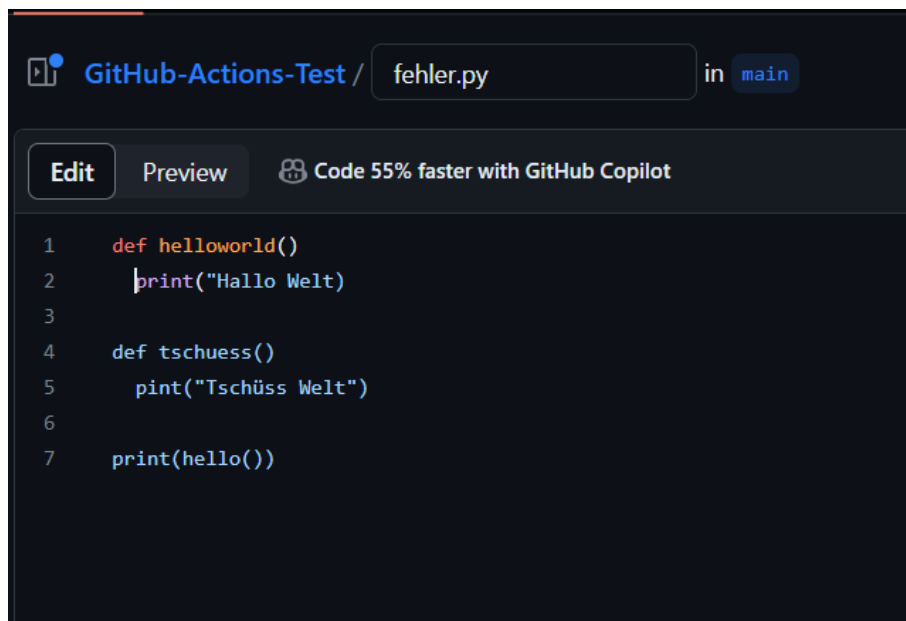


Hier sehen wir das unser Workflow durchgelaufen ist und uns keinen Fehler wirft. Wir haben aber auch noch nicht wirklich Dateien hinzugefügt.



## Workflow in „action“

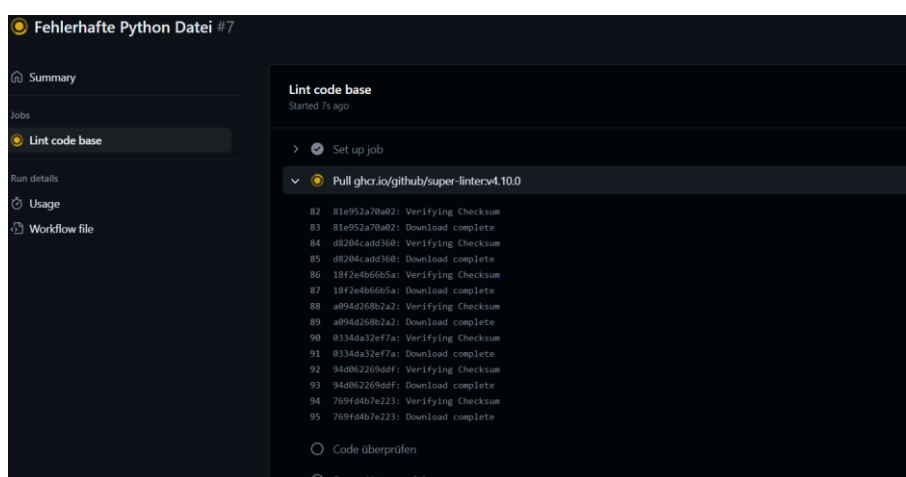
Hier habe ich ein kleines Python Programm, welches aber extra falsch geschrieben wurde. In diesem Programm will ich zwei Funktionen erstellen, welche jeweils eine Text ausgeben.



The screenshot shows the GitHub interface for a file named `fehler.py` in the `main` branch of the repository `GitHub-Actions-Test`. The file content is as follows:

```
1 def helloworld()  
2     print("Hallo Welt")  
3  
4 def tschuess()  
5     pint("Tschüss Welt")  
6  
7 print(hello())
```

Bei der Überprüfung stellt unser Workflow aber nun fest das unser Programm Fehler enthält und sogar welche Fehler. Es gibt uns auch Vorschläge zur Verbesserung unseres Codes.



The screenshot displays the GitHub Actions workflow run for the job `Lint code base`. The workflow is titled `Fehlerhafte Python Datei #7`. The job status is `Completed`. The workflow file is `.github/workflows/lint.yml`. The job details show the following steps:

- `Set up job`
- `Pull ghcr.io/github/super-linter:4.10.0`

The job output shows the following logs:

```
82 81e952a7b0d2: Verifying Checksum  
83 81e952a7b0d2: Download complete  
84 a8204eadd368: Verifying Checksum  
85 a8204eadd368: Download complete  
86 18f2e4b6d55a: Verifying Checksum  
87 18f2e4b6d55a: Download complete  
88 a894258b2a2: Verifying Checksum  
89 a894258b2a2: Download complete  
90 0334a32e7a: Verifying Checksum  
91 0334a32e7a: Download complete  
92 9489c226b04f: Verifying Checksum  
93 9489c226b04f: Download complete  
94 769f6b7e223: Verifying Checksum  
95 769f6b7e223: Download complete
```

The job output also shows the following options:

- ☐ Code überprüfen
- ☐ Super-Linter ausführen

```
200 2023-12-06 14:34:38 [ERROR] Found errors in [black] linter!
201 2023-12-06 14:34:38 [ERROR] Error code: 1. Command output:
202 -----
203 --- /github/workspace/fehler.py 2023-12-06 14:34:26.209552 +0000
204 +++ /github/workspace/fehler.py 2023-12-06 14:34:38.643182 +0000
205 @@ -1,7 +1,9 @@
206  def helloworld():
207      print("Hallo Welt")
208  -
209  +
210  +
211  def tschuess():
212  - print("Tschüss Welt")
213  + print("Tschüss Welt")
214  +
215
216  print(helloworld())
217 would reformat /github/workspace/fehler.py
218
219
220 Oh no! 🌟💔🌟
221 1 file would be reformatted.
222 -----
```

```
240 2023-12-06 14:34:39 [ERROR] Found errors in [flake8] linter!
241 2023-12-06 14:34:39 [ERROR] Error code: 1. Command output:
242 -----
243 /github/workspace/fehler.py:3:1: W293 blank line contains whitespace
244 /github/workspace/fehler.py:4:1: E302 expected 2 blank lines, found 1
245 /github/workspace/fehler.py:5:3: E111 indentation is not a multiple of 4
246 /github/workspace/fehler.py:7:1: E305 expected 2 blank lines after class or function definition, found 1
247 -----
```

Hier ist dann die ausgebesserte Datei. Es wurden einige Leerzeichen und Absätze hinzugefügt.

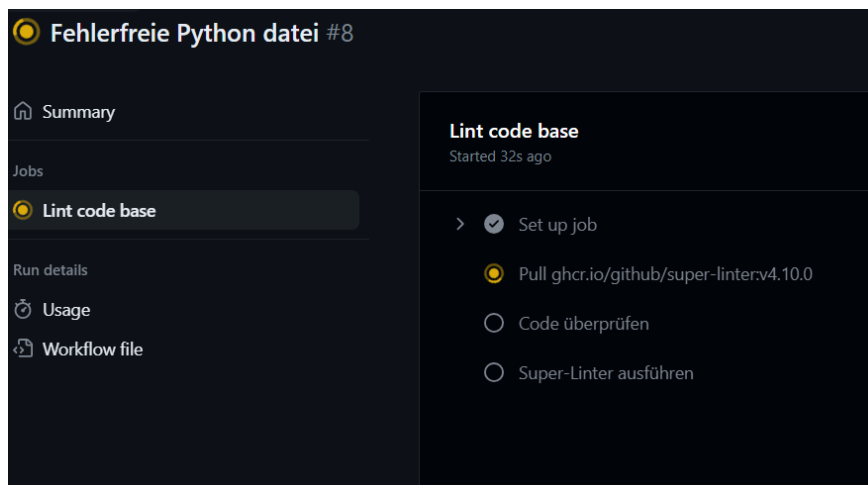


```
def helloworld():
    print("Hallo Welt")

def tschuess():
    print("Tschüss Welt")

print(helloworld())
```

Nun zeigt uns unser Workflow das unser gepushder Code einwandfrei funktioniert.



**Fehlerfreie Python datei #8**

Summary

Jobs

- Lint code base

Run details

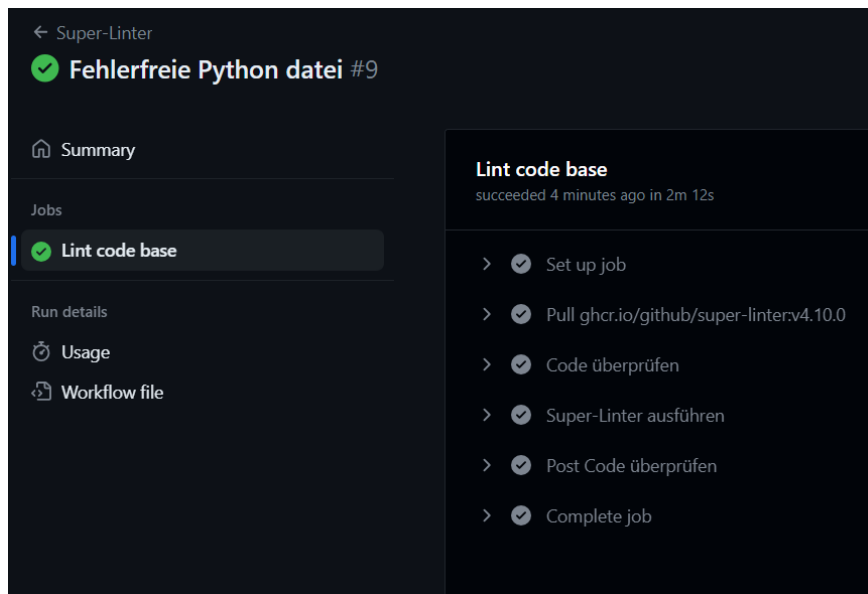
- Usage
- Workflow file

**Lint code base**

Started 32s ago

- > ☒ Set up job
- ☒ Pull ghcr.io/github/super-linter:v4.10.0
- ☐ Code überprüfen
- ☐ Super-Linter ausführen





## Quellen

<https://www.youtube.com/watch?v=mFFXuXjVgkU>

<https://github.com/features/actions>

<https://chat.openai.com/>