Spickzettel: GitHub Workflow erstellen

1. Was ist ein GitHub Workflow?

- Ein **automatisierter Prozess** in GitHub Actions.
- Besteht aus **Events**, **Jobs** und **Steps**.
- Wird als **YAML-Datei** in `.github/workflows/` gespeichert.

2. Neuen Workflow anlegen

1. **Verzeichnis für Workflows erstellen (falls nicht vorhanden)**

```
mkdir -p .github/workflows
```

2. **Neue Workflow-Datei anlegen**

```
touch .github/workflows/ci.yml
```

3. Aufbau eines Workflows

```
name: Mein CI-Workflow
on: [push, pull_request]

jobs:
  build:
    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
```

4. Wichtige Bestandteile eines Workflows

- **`name`** – Name des Workflows.
- **`on`** – Events, die den Workflow auslösen (`push`, `pull_request`, `schedule`).
- **`jobs`** – Enthält alle Aufgaben, die parallel oder sequenziell ausgeführt werden.
- **`runs-on`** – Definiert das Betriebssystem für den Job (`ubuntu-latest`, `windows-latest`).
- **`steps`** – Liste von Befehlen oder vorgefertigten Aktionen.

5. Workflow testen & debuggen

- **Manuellen Workflow-Trigger aktivieren** (optional):

```
on:
  workflow_dispatch:
```

- **Fehlgeschlagene Workflows analysieren** → GitHub Actions Logs unter "Actions" in GitHub prüfen.
- **Umgebungsvariablen setzen & verwenden:**

```
env:
  NODE_ENV: production
```

Best Practices

- **Strukturierte Workflows nutzen**, um sie leicht verständlich zu halten.
- **Caching verwenden** (`actions/cache`), um Builds zu beschleunigen.
- **Jobs parallelisieren**, falls möglich (`strategy.matrix`).
- **Fehlermeldungen direkt in den GitHub Actions Logs prüfen**.