Spickzettel: Docker Compose + GitHub Actions für CI/CD

Ziel

- **Automatisiertes Bauen, Testen und Starten** von Docker Compose-Projekten via GitHub Actions.
- Unterstützt **Multi-Container-Projekte** in lokalem CI/CD-Setup.

Beispiel: `docker-compose.yml`

```
version: '3.8'
services:
  web:
    build: .
    ports:
      - "8000:8000"
    depends_on:
      - db
  db:
    image: postgres:15
    environment:
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
      POSTGRES_DB: appdb
```

Beispielworkflow `.github/workflows/ci.yml`

```
name: CI mit Docker Compose

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest
    services:
      docker:
        image: docker:20.10.16-dind
        privileged: true

    steps:
      - name: Checkout Code
        uses: actions/checkout@v3

      - name: Docker Compose installieren
        run: |
          sudo apt-get update
          sudo apt-get install -y docker-compose

      - name: Container starten (Build + Up)
        run: docker-compose up -d --build
```

```
    - name: Teste Web-Service
      run: |
        sleep 5  # Warten auf Services
        curl http://localhost:8000 || exit 1

    - name: Container herunterfahren
      run: docker-compose down
```

Best Practices

- `docker-compose up -d --build` für frischen Containerstart mit Build.
- Immer `docker-compose down` am Ende ausführen, um saubere Runs zu garantieren.
- Optional: Tests via `pytest`, `unittest` oder `curl`-Befehle einbinden.
- Variablen und Secrets via `.env` oder `secrets.GITHUB_TOKEN` verwalten.

Hinweise

- GitHub Actions Runner kann Docker Compose ausführen, benötigt aber ggf. manuelle Installation.
- Für produktive Umgebungen: Deployment per `docker-compose -f docker-compose.prod.yml`.