Spickzettel: CI/CD Deployment mit GitHub Actions – auf Server oder Cloud

Ziel

Code nach erfolgreichem Build automatisch auf einen Zielserver oder Cloud-Dienst bereitstellen – per GitHub Actions.

Voraussetzungen

- GitHub Actions aktiviert
- Zielumgebung: Linux-Server (z. B. via SSH) oder Cloud (z. B. Heroku, AWS, Docker Hub)
- Geheimnisse (`secrets`) hinterlegt: SSH-Key, API-Token etc.

Beispiel: Deployment via SSH auf Linux-Server

`.github/workflows/deploy.yml`

```
name: Deployment

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest

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

      - name: Auf Server kopieren (via rsync)
        run: |
          mkdir -p ~/.ssh
          echo "$SSH_KEY" > ~/.ssh/id_rsa
          chmod 600 ~/.ssh/id_rsa
          rsync -avz ./ myuser@myhost:/var/www/project/
        env:
          SSH_KEY: ${{ secrets.SSH_KEY }}
```

Beispiel: Deployment auf Docker Hub

```
- name: Docker Login
  run: echo ${{ secrets.DOCKER_PASSWORD }} | docker login -u $
{{ secrets.DOCKER_USERNAME }} --password-stdin
```

```
- name: Image bauen und pushen
  run: |
    docker build -t myuser/myimage:latest .
    docker push myuser/myimage:latest
```

Typische `secrets`

- `SSH_KEY` → für Serverzugang
- `DOCKER_USERNAME` / `DOCKER_PASSWORD`
- `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`
- `HEROKU_API_KEY`, `FLY_API_TOKEN`, etc.

Best Practices

- Deployment nur bei `push` auf `main`, `release/*` oder nach Tests
- Secrets niemals direkt ins YAML schreiben → immer über `secrets.*`
- Vor dem Deploy: Tests, Linting, Build prüfen
- Rollback-Möglichkeit vorsehen (z. B. per Tagging)
- Logs überwachen (`Actions → Run → Logs`)

GitHub Actions ermöglichen vollständige CI/CD-Pipelines direkt im Repository – für klassische Server, Container oder Cloud-Plattformen.