Spickzettel: GitHub Releases automatisieren

Ziel

Automatisch GitHub-Releases erstellen und Anhänge (z. B. Binaries, PDFs, Artefakte) hochladen – per GitHub Actions.

Voraussetzungen

- GitHub Actions aktiviert
- Repository mit Tags oder Versionierung
- Build-Artefakte werden generiert (optional)

Workflow: Release bei neuem Tag

`.github/workflows/release.yml`

```
name: Release

on:
  push:
    tags:
      - 'v*'

jobs:
  release:
    runs-on: ubuntu-latest

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

      - name: Version aus Tag extrahieren
        run: echo "RELEASE_VERSION=${GITHUB_REF#refs/tags/}" >> $GITHUB_ENV

      - name: Artefakte bauen (optional)
        run: |
          mkdir build
          echo "Demo-Datei" > build/info.txt

      - name: Release erstellen
        uses: softprops/action-gh-release@v1
        with:
          tag_name: ${{ env.RELEASE_VERSION }}
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}

      - name: Dateien anhängen
        uses: softprops/action-gh-release@v1
```

```
      with:
        files: build/*
      env:
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

Alternative Trigger

- Release bei Push auf `main`, kombiniert mit `git tag`
- Manuelles Triggern via `workflow_dispatch`

Sicherheit & Token

- `GITHUB_TOKEN` wird automatisch bereitgestellt
- Falls externe Tools nötig: `repo`-Token in Secrets speichern

Best Practices

- Versionsnummer aus `pyproject.toml`/`package.json` lesen (für Konsistenz)
- Pre-Releases mit `prerelease: true` markieren
- Änderungslog automatisch generieren (z. B. `github-release-notes`)
- Signierte Tags für Release-Sicherheit verwenden

Automatisierte Releases sparen Zeit, verhindern manuelle Fehler und machen Builds nachvollziehbar – ideal für CI/CD-Workflows.