JOHANNES KEPLER
UNIVERSITY LINZ

Praktikum Software Engineering

Johannes Bräuer

Requirements und Tools Workshop

JOHANNES KEPLER
UNIVERSITY LINZ

Agenda

- Organisatorisches
- Anforderungen an Lunchify
- Präsentationen & Demo

© 2025 Business Informatics - Software Engineering

Nächste Aufgaben bis 21.3.

- **Fokus: UI Prototyp**
- **Deliverables**
  - ➜ UI Prototype
  - ➜ UML-Diagramme Grobkonzept zum Aufbau der Applikation
  - ➜ *Lauffähige Builds in GitHub Actions*
  - ➜ Planung Release 1
  - ➜ Live Demo des UI Prototyps & Grobkonzept

© 2025 Business Informatics - Software Engineering

Anforderungen an Lunchify

Projektbeschreibung auf GitHub:

https://github.com/jku-win-se/teaching.ss25.prse.prwiki.braeuer/blob/main/documents/2025-SS-Anforderungen%20Lunchify.pdf

© 2025 Business Informatics - Software Engineering

GitHub Login

Außerhalb der GitHub Webseite (z.B. über Eclipse) ist ein Login mit Username und Passwort nicht mehr möglich.

Sie müssen in Ihrem GitHub Profil einen **Personal Access Token** generieren und dieser dient dann als Passwortersatz für den Login in der IDE.

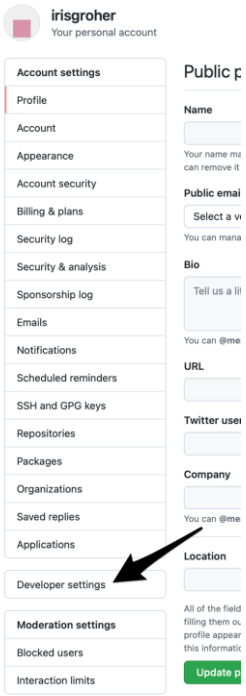
Speichern Sie den Token, falls Sie ihn später nochmals eingeben müssen.

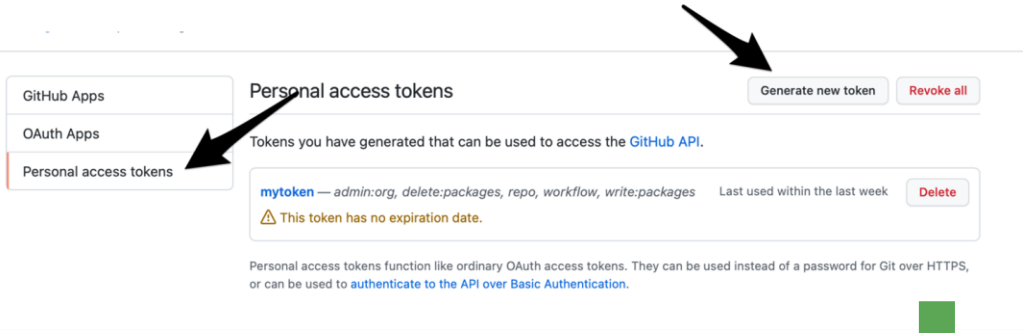
GitHub → Profile → Settings → Developer settings → Personal access token

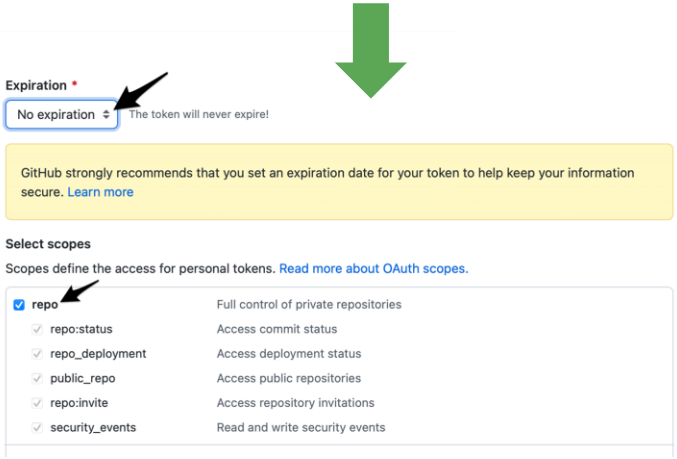
(https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token)

© 2025 Business Informatics - Software Engineering

GitHub Login







© 2025 Business Informatics - Software Engineering

Themen (1)

- **Topic-1: Git (mit GitHub)**
  - ➜ Git Funktionalitäten
  - ➜ Git: Desktop-Anwendungen und Integration in Programmierumgebung
- **Topic-2: GitHub & Clockify**
  - ➜ Issues, Projects, Milestones, Releases
  - ➜ Verantwortlichkeiten, Zeiterfassung mittels Clockify
- **Topic-3: Maven & GitHub Actions**
  - ➜ Maven Funktionalität
  - ➜ Projekt mit GitHub Actions bauen

© 2025 Business Informatics - Software Engineering

Themen (2)

- **Topic-4: UML Tools / Editoren**
  - ➜ Open Source oder Demo Version bis Juli
  - ➜ Mindestens 3 Tools (bevorzugt SaaS Lösungen)

- **Topic-5: UI Prototyping + Tools**
  - ➜ Open Source oder Demo Version bis Juli
  - ➜ Mindestens 3 Tools (bevorzugt SaaS Lösungen)

- **Topic-6: GitFlow**
  - ➜ Wie entwickelt man ein neues Feature?
  - ➜ How-to Guide und Live-Übung mit allen Studierenden (bring your Notebook)

© 2025 Business Informatics - Software Engineering

GitFlow

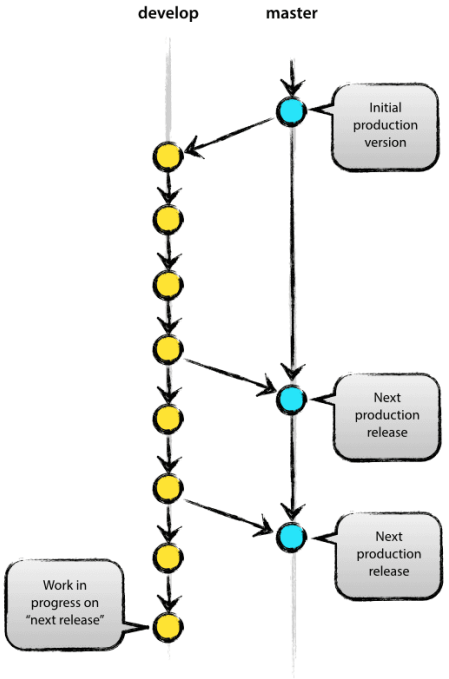
GitFlow is a branching model for Git, created by Vincent Driessen. It supports the tasks of:

- **Parallel development**: Isolates new development from finished work
- **Collaboration:** Makes it easier for two or more developers to collaborate on the same feature
- **Release and staging process:** As new development is completed, it gets merged back into the main branch, which is a staging area for all completed features that have not yet been released
- **Emergency fixes:** Supports hotfix - branches made from a tagged release that contains only the emergency fix

© 2025 Business Informatics - Software Engineering

GitFlow - The main branches: ~~master~~ main & develop

- Both have an infinite lifetime (compared to supporting branches shown next)
- `develop:` source code reflects a state with the latest delivered development changes for the next release
- When the source code in the `develop` branch reaches a stable point and is ready to be released → all changes are merged back into `main`
- `main:` is tagged with a release number

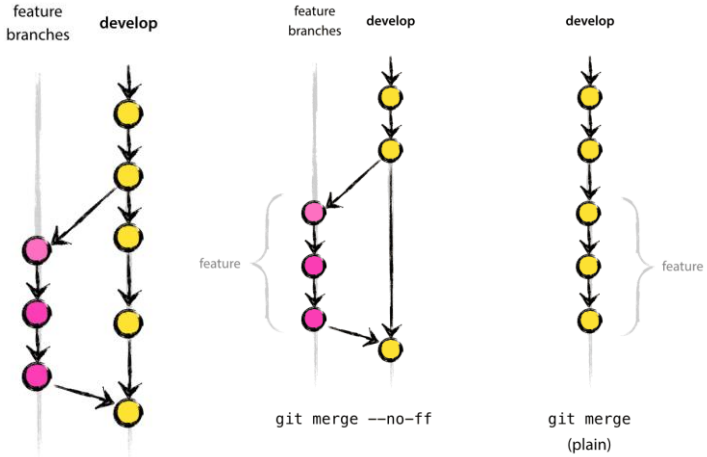


© 2025 Business Informatics - Software Engineering

GitFlow - The supporting branches: feature, release & hotfix

- Used to develop new features for the upcoming release
- Sandbox where the only changes are the changes necessary to get the new feature working

- May branch off from:
  - ➜ `develop`
- Must merge back into:
  - ➜ `develop`
- Branch naming convention:
  - ➜ anything except `main`, `develop`, `release-*`, or `hotfix-*`

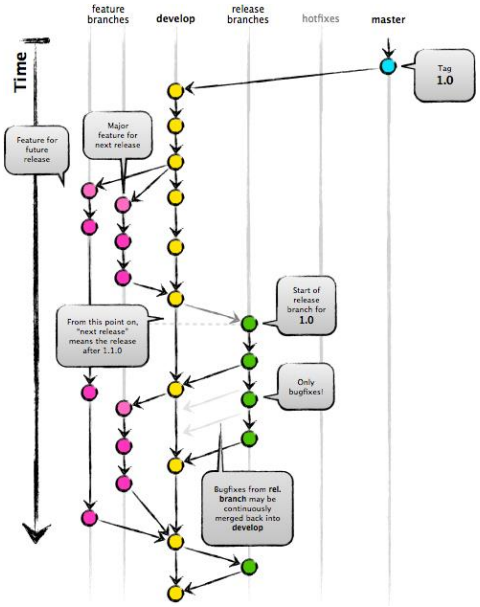


© 2025 Business Informatics - Software Engineering

GitFlow - The supporting branches: feature, release & hotfix

- Support preparation of a new production release
- Allow minor bug fixes and preparing meta-data for a release

- May branch off from:
  - → `develop`
- Must merge back into:
  - → `develop` and `main`
- Branch naming convention:
  - → `release-*`

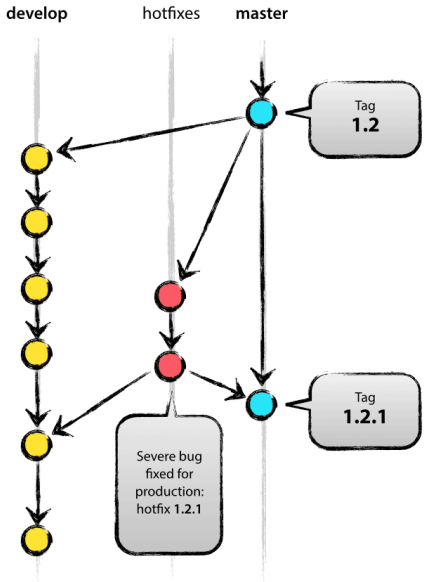


© 2025 Business Informatics - Software Engineering

GitFlow - The supporting branches: feature, release & hotfix

- Meant to prepare for a new (unplanned) production release
- Arise from the necessity to act immediately upon an undesired state (e.g., critical bug) of a live production version

- May branch off from:
  - → `main`
- Must merge back into:
  - → `develop` and `main`
- Branch naming convention:
  - → `hotfix-*`



© 2025 Business Informatics - Software Engineering

JOHANNES KEPLER
UNIVERSITY LINZ