

LMU

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Praktikum Mobile und Verteilte Systeme

Arbeiten mit Git

Prof. Dr. Claudia Linnhoff-Popien
André Ebert, Sebastian Feld
<http://www.mobile.ifi.lmu.de>

WS 2017/18



Was ist Git

- Verteiltes **Versionierungssystem** (siehe auch SVN)
- Open Source
- **Kollaboratives Arbeiten** und Versionskontrolle



Wo gibt's Git?

- Zur Nutzung werden 2 Komponenten benötigt
 - **Lokale Installation** eines Git-Clients
 - **Git-Repository** auf einem Server
 - viele kostenlose und kostenpflichtige Online-Angebote: *GitHub*, *Gitlab*, *Bitbucket*, etc.
 - Kostenloses Repository (Gitlab-based) an der LMU ohne Berechtigungs- und Privacy-Einschränkungen.



Versionierungssysteme: Git vs. SVN

Git	SVN
<ul style="list-style-type: none">• Verteilte Versionsverwaltung (mit zentralem Repository)	<ul style="list-style-type: none">• Zentrale Versionsverwaltung
<ul style="list-style-type: none">• lokal vorliegende Repository-Kopien, in denen gearbeitet wird (Branching)	<ul style="list-style-type: none">• ein zentrales Repository, in dem Arbeitskopien erzeugt werden
<ul style="list-style-type: none">• Zugang zum Gesamtverzeichnis	<ul style="list-style-type: none">• Pfadbasierte Zugangsberechtigungen
<ul style="list-style-type: none">• Inhaltsbasierte Änderungsverfolgung	<ul style="list-style-type: none">• Dateibasierte Änderungsverfolgung
<ul style="list-style-type: none">• Repository und Arbeitskopien enthalten die komplette Änderungshistorie	<ul style="list-style-type: none">• Änderungshistorie nur im Repository komplett, Arbeitskopien enthalten nur neueste Version
<ul style="list-style-type: none">• Netzwerkanbindung ist nur zur Synchronisation notwendig	<ul style="list-style-type: none">• Netzwerkanbindung ist bei jedem Zugriff notwendig.

Git-Installation

Unter Linux/Debian

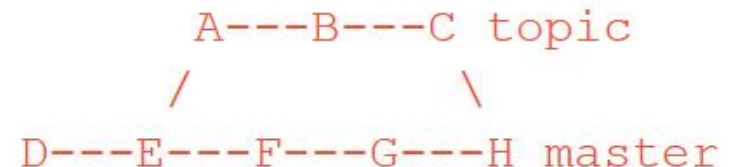
- `sudo apt-get install git-all`
 - Installiert Git Client-Software, Nutzung via Kommandozeile möglich
 - GUIs: Tortoise SVN, GitKraken, etc.

Unter Windows

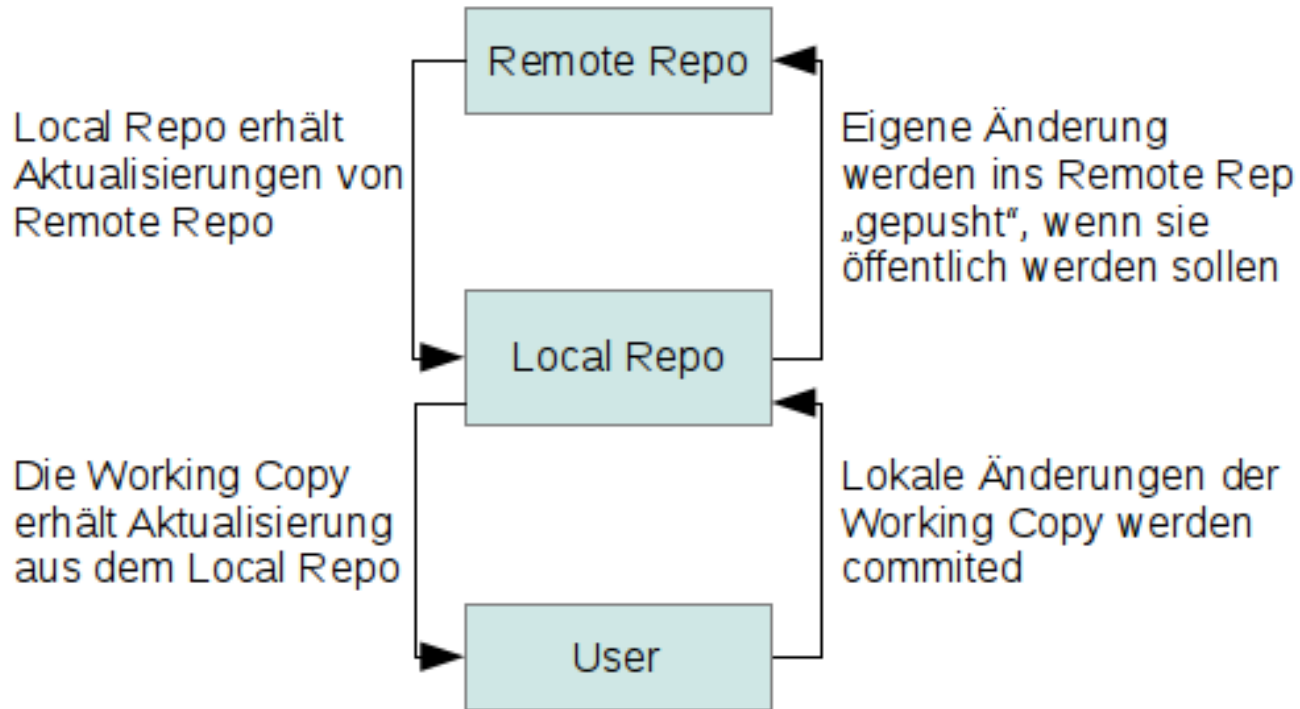
- **Git for Windows:** <https://git-for-windows.github.io/>
 - Grafische GUI und Kommandozeilen Tool
 - Visualisierung von Commit- und Merge-Histories
- GitKraken: <https://www.gitkraken.com/>
- etc.

Remote Repository

- **LMU Uni Gitlab**
 - https://gitlab.cip.ifi.lmu.de/users/sign_in
 - Benutzung bei der RGB beantragen
 - Login via CIP-Account



Genereller Workflow



Quelle: https://www.thomas-krenn.com/de/wiki/Git_Grundbegriffe

Working with Git

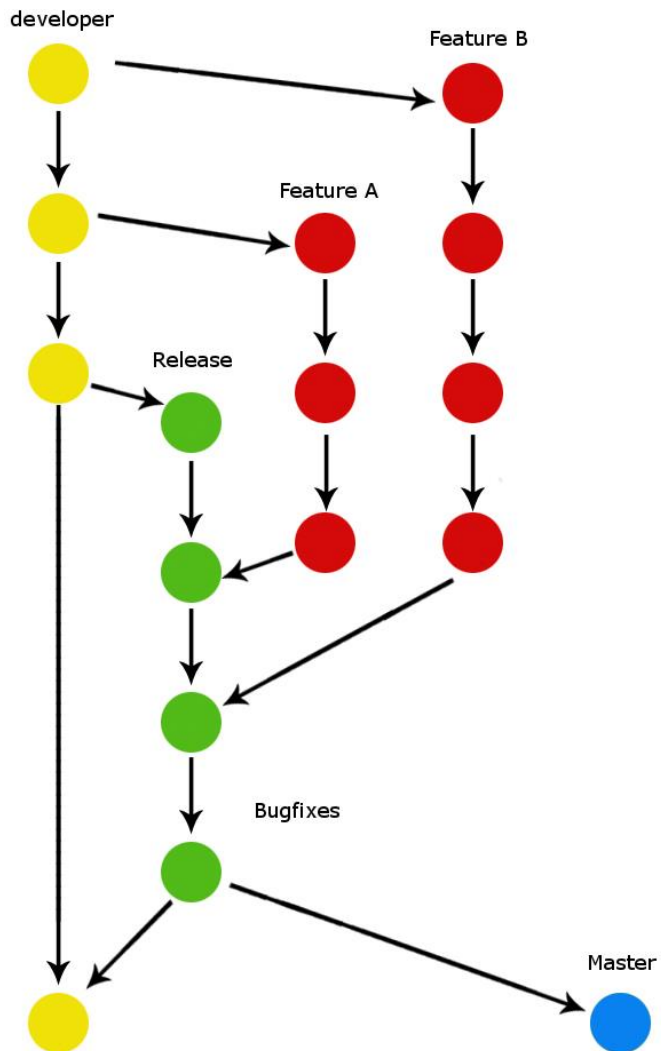
1. `git clone <remote-url>` (einmalig) oder `git init`
 - Klonen eines existierenden Repositories oder erstellen eines neuen
2. `git pull` oder `git fetch`
 - Vor jeder Arbeits-Session, um sicherzugehen dass lokal die neuste Version als Grundlage genommen wird
3. `git add <file oder folder>`
 - Merkt Änderungen an der Datei für den nächsten Commit vor (Staging Area)
 - Mit `git status` lassen sich der Zustand von Arbeitsverzeichnis und Staging Area sowie aller vorgemerkten Files überprüfen
4. `git commit -m „<commit message>“`
 - Übernimmt die vorgemerkten Änderungen in das lokale Repository
5. `git push <ggf. local-branch><ggf. remote-branch>`
 - Menge aller lokalen Commits wird in das Remote Repository gepushed. Somit sind nun alle Änderungen für alle anderen Team-Mitglieder sichtbar.

Git Branches

Beispiel zum Arbeiten mit Branches

- **1 Master-Branch** im Repository:
 - *Remote: origin/master* und *Lokal: /master*
 - Remote (und Lokal) sollte **nur von einer Person verändert** werden
 - Enthält die lauffähige oder aktuellste Version der Software
- **1 allgemeiner Development-Branch** bspw. (/dev)
 - Zum Entwickeln und Testen
- **X individuelle Feature-Banches**, bspw. /feature-x
 - Entwicklung unabhängiger Features mit definierten Schnittstellen
- **1 Release-Branch** (bspw. /release)
 - Sammeln und Mergen von Dev- und Feature-Banches
 - Nur eine Person ist für das Merging und den Push zum Master-Branch verantwortlich
 - » `git push <branch-to-push> origin/master`

Beispiel: Release-Planung mit Branches



Quelle: <https://i.stack.imgur.com/J4QQd.png>

1. Im Development-Branch und in den Feature-Branches wird parallel entwickelt
2. Bei geplantem Release wird der aktuellste Development-Branch als Grundlage für den Release-Branch genommen
3. Feature-Branches werden sequentiell mit Development gemerged und getestet
4. Abschließendes Bugfixing und schließlich Push des Release in den Master-Branch
5. Nutzen des Release als neue Development Grundlage

Links

Ausführliches Tutorial:

<https://git-scm.com/docs/gittutorial>

Git Dokumentation:

<https://git-scm.com/doc>



Und keine Panik, Git vergisst nie!