

Versionsmanagement: Git Prinzip & Clients

B. Sc. Simon Müller
Denis Westerheide

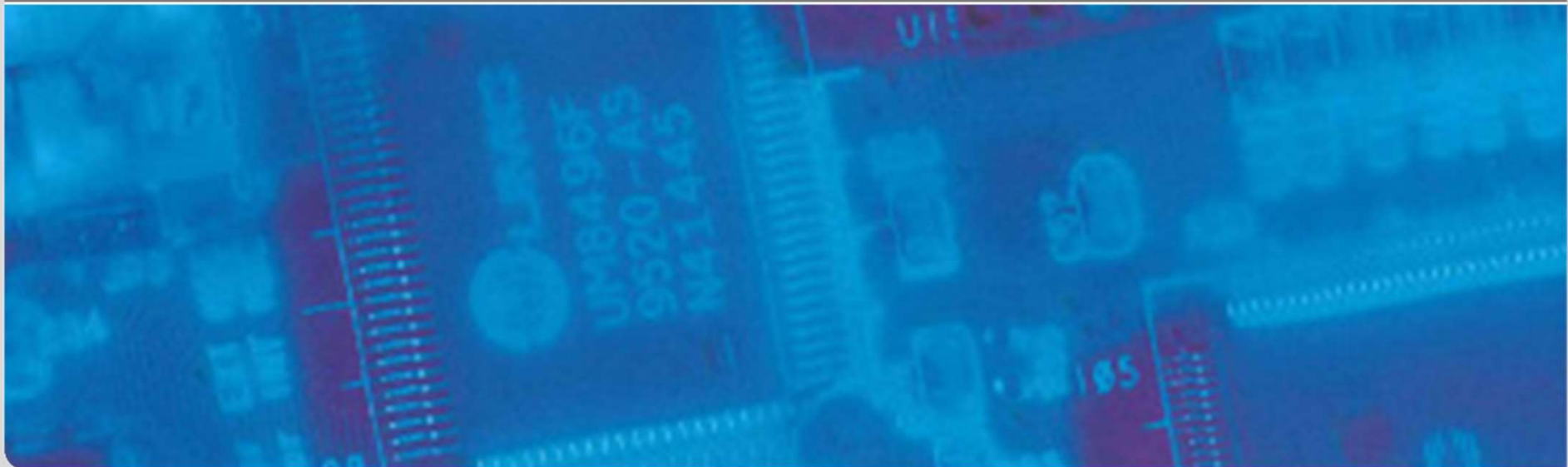
Institutsleitung

Prof. Dr.-Ing. Dr. h. c. J. Becker (Sprecher)

Prof. Dr.-Ing. Eric Sax

Prof. Dr. rer. nat. W. Stork

Institut für Technik der Informationsverarbeitung (ITIV)



Inhalt

■ Versionsmanagement mit Git

- Warum Versionsverwaltung?
- Was ist Git? & Wie funktioniert es?
- Kommandos & Konflikte
- Workflow

■ Tool TortoiseGit

- Vorstellung & Nutzung

■ Tool EclipseGit

- Vorstellung & Nutzung

■ *Optional: Live-Demo TortoiseGit & EclipseGit*

Warum Versionsverwaltung?

Who would win

Most advanced version control system used to keep track of changes in any set of files. Aimed at speed, data integrity, and support for distributed, non-linear workflows.



Me

Development > Creations > Android > My Color Manager All Saves >			
are with ▾	Burn	New folder	
Name	Date modified	Type	Size
ColorManager	13-03-2017 12:47	File folder	
ColorManager 9-5-2016	13-03-2017 12:47	File folder	
ColorManager 11-5-2016	13-03-2017 12:48	File folder	
ColorManager save1	13-03-2017 12:48	File folder	
MyColorManager 03-06-16	13-03-2017 12:48	File folder	
MyColorManager 4-7-2016	13-03-2017 12:49	File folder	
MyColorManager 4-7-2016 2nd	13-03-2017 12:49	File folder	
MyColorManager 5-6-16	13-03-2017 12:50	File folder	
MyColorManager 5-8-16	13-03-2017 12:51	File folder	
MyColorManager 6-7-16	13-03-2017 12:51	File folder	
MyColorManager 10-6-2016	13-03-2017 12:52	File folder	
MyColorManager 10-7-2016	13-03-2017 12:52	File folder	
MyColorManager 10-8-2016	13-03-2017 12:53	File folder	
MyColorManager 13-7-2016	13-03-2017 12:54	File folder	
MyColorManager 15-5-16	13-03-2017 12:54	File folder	
MyColorManager 17-5-16	13-03-2017 12:55	File folder	
MyColorManager 18-7-16	13-03-2017 12:56	File folder	
MyColorManager 19-05-16	13-03-2017 12:56	File folder	
MyColorManager 20-5-16 13-24	13-03-2017 12:57	File folder	
MyColorManager 20-8-2016	13-03-2017 12:58	File folder	
MyColorManager 21-7-2016	13-03-2017 12:58	File folder	
MyColorManager 23-5-16 2	13-03-2017 12:59	File folder	
MyColorManager 23-7-2016	13-03-2017 12:59	File folder	
MyColorManager 24-5-16	13-03-2017 13:00	File folder	
MyColorManager 25-8-2016	13-03-2017 13:01	File folder	
MyColorManager 25-9-2016	13-03-2017 13:02	File folder	
MyColorManager 26-5-16	13-03-2017 13:02	File folder	
MyColorManager 26-7-2016	13-03-2017 13:03	File folder	
MyColorManager 27-7-2016	13-03-2017 13:04	File folder	
MyColorManager 27-09-2016	13-03-2017 13:05	File folder	
MyColorManager 30-7-2016	17-09-2017 01:06	File folder	
MyColorManager 2016-9-21	13-03-2017 13:06	File folder	

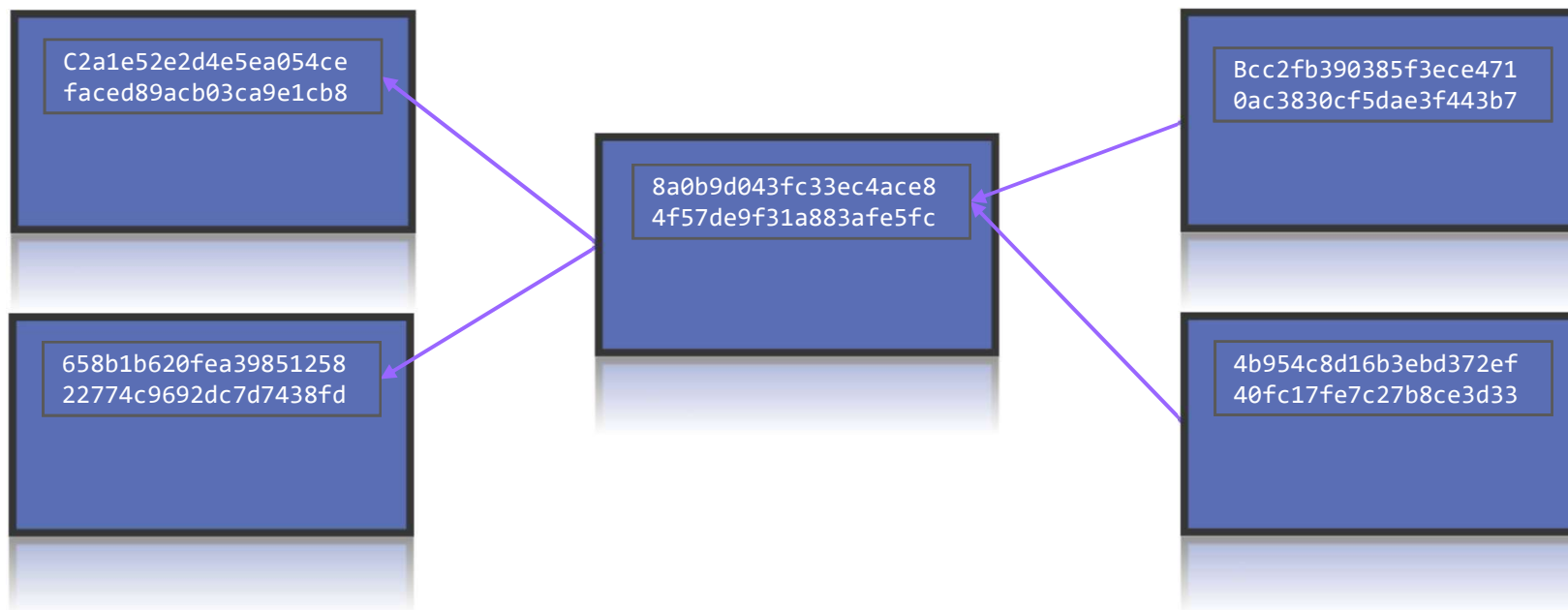
Quelle: reddit.com

Was ist Git?

- Versionskontrollsystem
- Dezentral (theoretisch)
- Praktisch oft:
 - Zentraler Server mit Master-Repository (github.com, Gitlab, ssh-Server, ...)
 - Vollständige lokale Kopien `git clone`
 - Hinzufügen von Änderungssets `git commit`
 - Regelmäßige Synchronisation `git push / git pull`
- Parallele Entwicklung über Branches
 - `git branch`
 - `git checkout`
 - `git merge`
 - `git rebase`

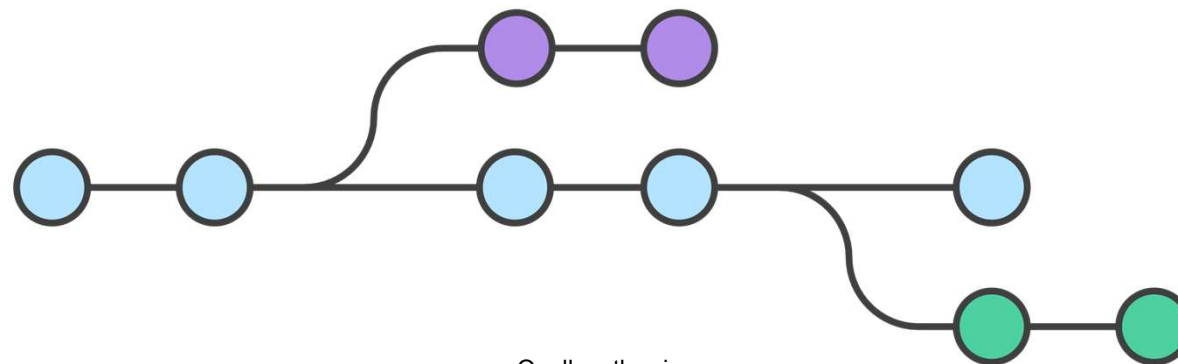
Wie funktioniert Git?

- „Linked List“ von Änderungssets (rückwärts)
- Sha1-Hash als ID
- Mehrere Vorgänger/Nachfolger möglich
- Neuste Version: Alle Änderungen „gestapelt“



Branches

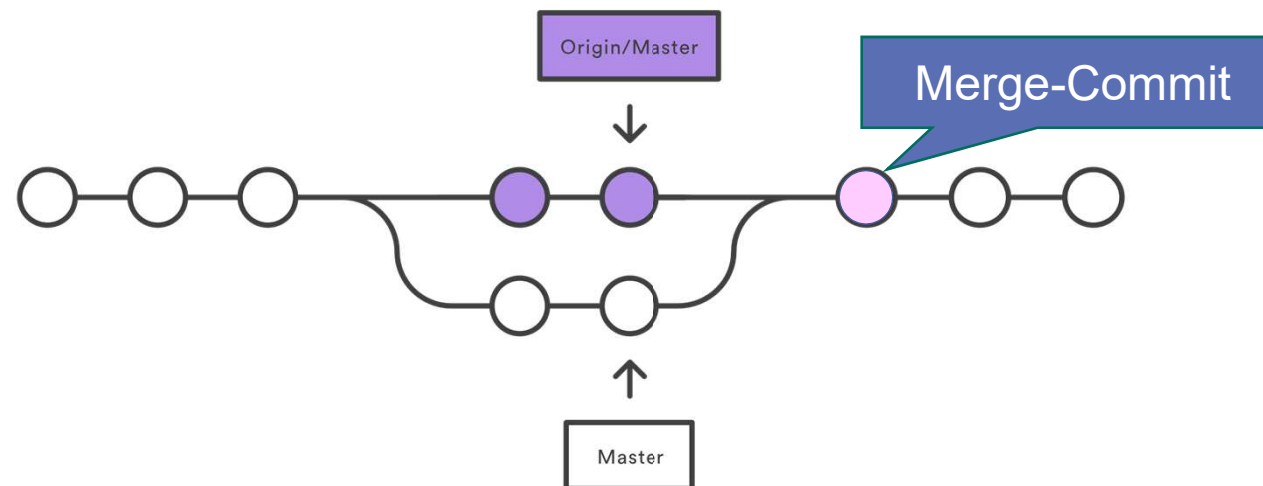
- Ziel: Immer eine funktionierende Code-Version
- Problem: Instabile Zwischenstände sichern?
- Lösung: „Code-Abzweigungen“ → Branches
 - Master Branch mit funktionierendem Code
 - Weitere Branches mit unfertigem Code



Quelle: atlassian.com

Merge/Rebase

```
error: failed to push some refs to '/path/to/repo.git'  
hint: Updates were rejected because the tip of your current branch is behind  
hint: its remote counterpart. Merge the remote changes (e.g. 'git pull')  
hint: before pushing again. hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```



Merge Konflikte

```
git.exe merge dev

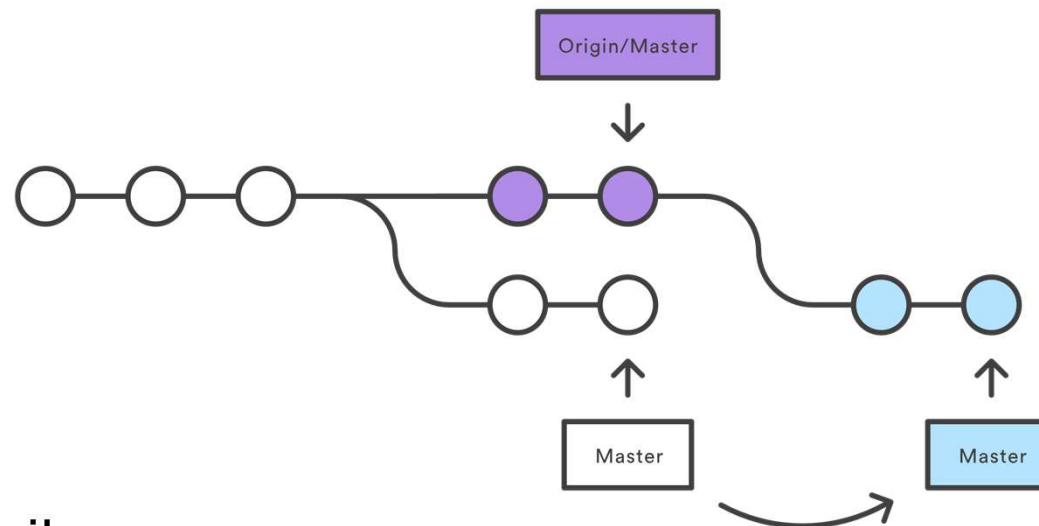
Auto-merging main.c
CONFLICT (content): Merge conflict in main.c
Automatic merge failed; fix conflicts and then commit the result.
```

```
<<<<<< HEAD
int main() {
=====
int main (int argc, char *arg[]) {
>>>>>> dev
```

- Ursache: Parallele Änderung einer Datei
 - Automatisches Mergen teilweise unmöglich
 - Manuelles Mergen in Texteditor/Tortoise Merge
- Wie vermeiden?
 - Bearbeitung von gleichen Dateien meiden
 - Kleine Commits, häufig mergen

Git Workflow – SVN-Style

- Ähnlich wie SVN-Workflow → Alle arbeiten auf Master
- Vor push: `git pull --rebase origin master`

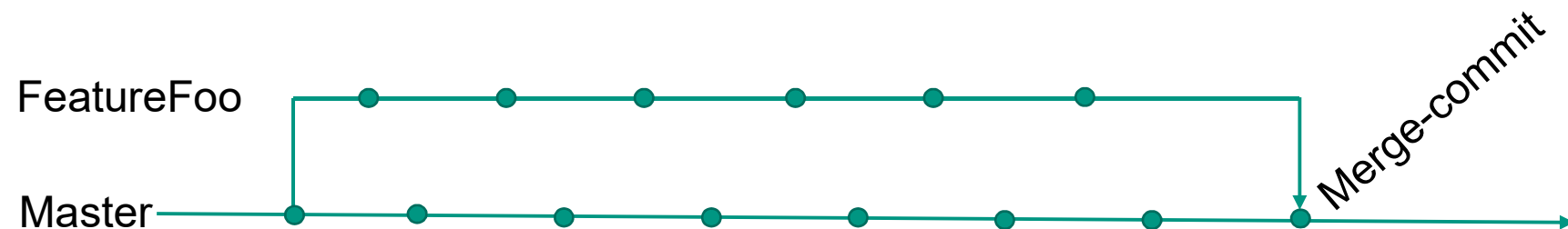


Quelle: atlassian.com

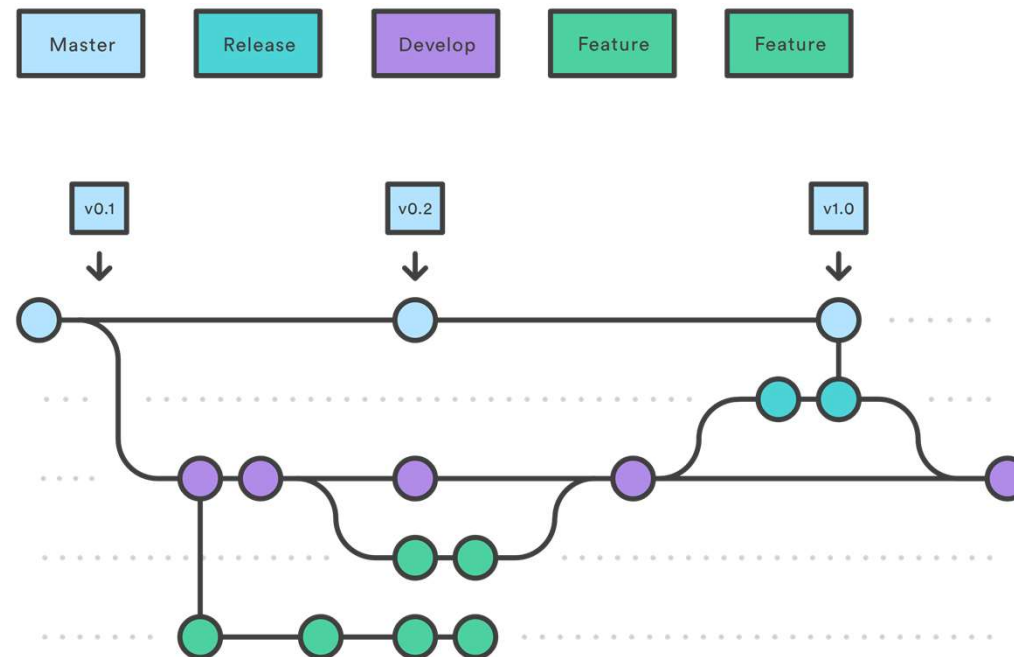
- Nachteile:
 - Änderungen evtl. lange Zeit nur lokal
 - Mergekonflikte sehr häufig

Git Workflow – Feature Branches

- Ziel: Neues Feature Foo
- Erzeuge Branch: FeatureFoo
 - Featurecommits NUR in FeatureFoo
 - Gleichzeitig: Bugfixes etc. auf Master
- Feature fertig → Pull Request/Merge FeatureFoo in Master



Git Workflow – Gitflow



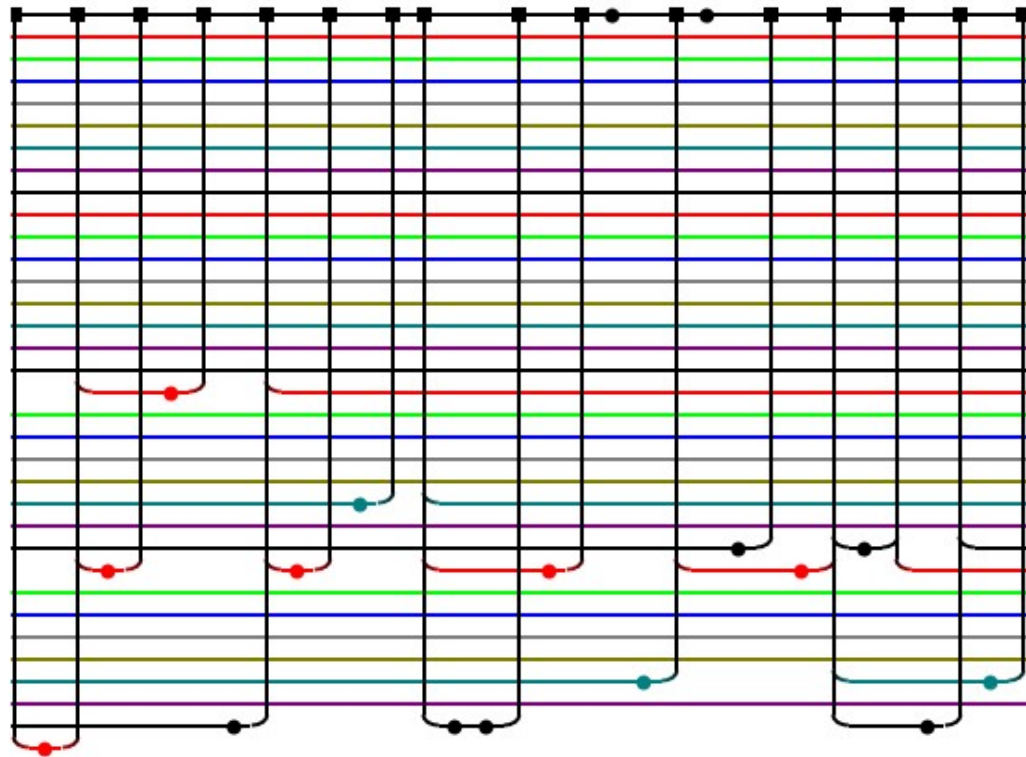
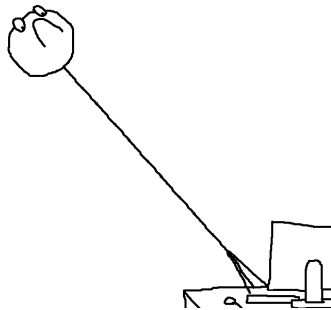
Quelle: atlassian.com

- Master nur für Releases
- Neuer Code nur in Development-Branch
- Abschließende Arbeiten in Release-Branch
- Merge in Master-Branch für neuen Release

Git Workflow – Best Practices

- Committe oft
 - Erleichtert Reverten bei Problemen
- Committe nur zusammengehörige Änderungen
 - Vermeidet Nebeneffekte beim Reverten
- Fülle die Commitmessage sinnvoll
 - Sinn eines Commits sollte sofort erkennbar sein
- **NIEMALS** push --force benutzen
 - Zerstört sämtliche lokale Kopien des Repositorys

Wie man Git (in kleinen Teams) nicht benutzt



Quelle: <https://github.com/rails/rails>

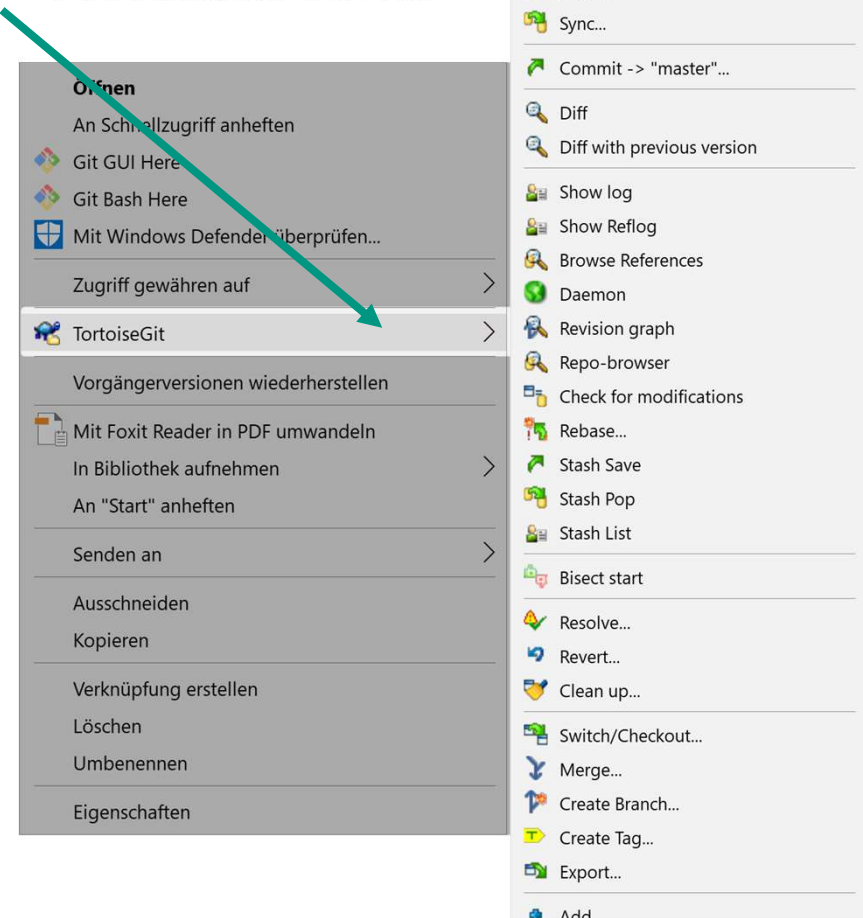


Vorstellung und Nutzung

TortoiseGit Vorstellung



- TortoiseGit ist ein Client für Windows
- Macht Kommandos im Explorer-Kontextmenü verfügbar
- Nützliche Zusatzfunktionen
 - Vergleichswerkzeug,
 - Icon Overlays,
 - uvm.

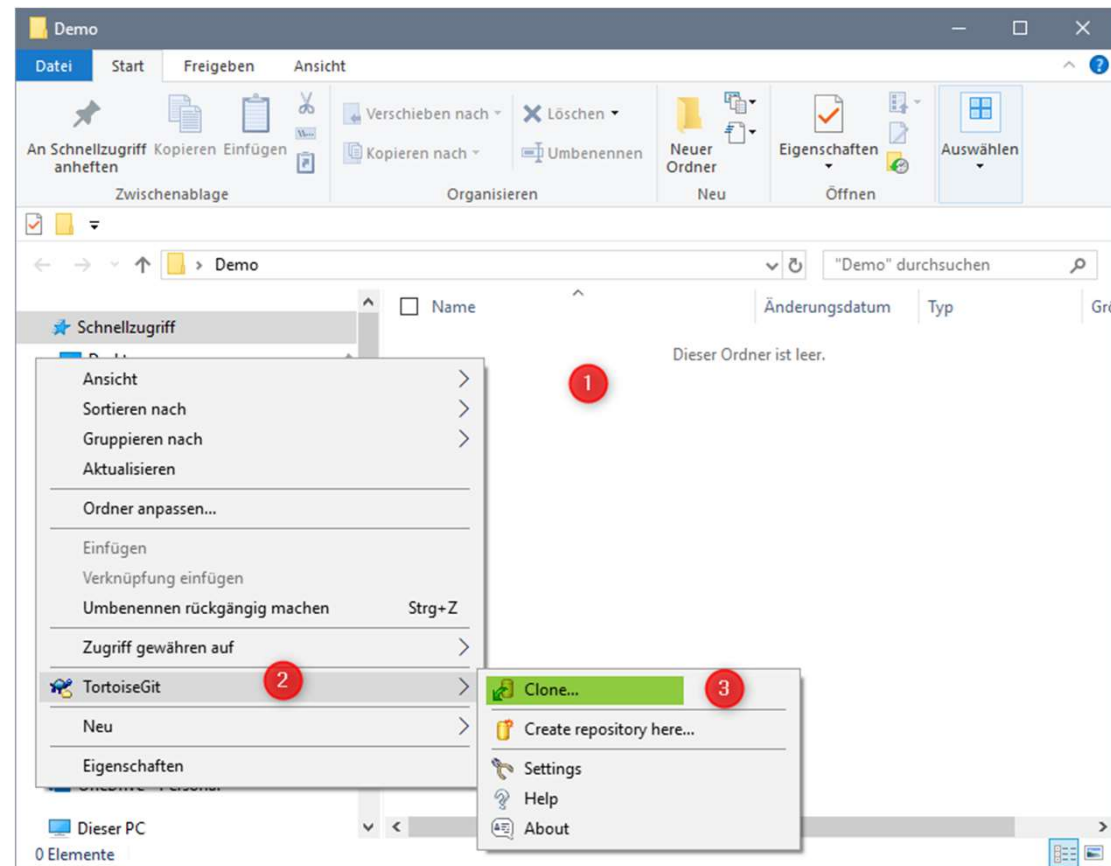


■ Übersicht der wichtigsten Funktionen

- Clone / Create
- Commit
- Sync
 - Pull, Push usw.
- Branch
- Switch/Checkout
- Merge

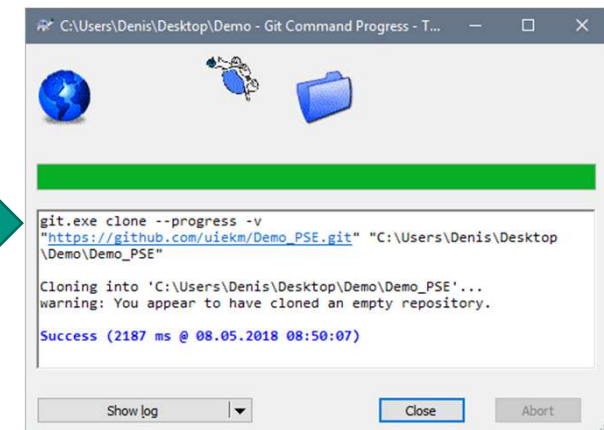
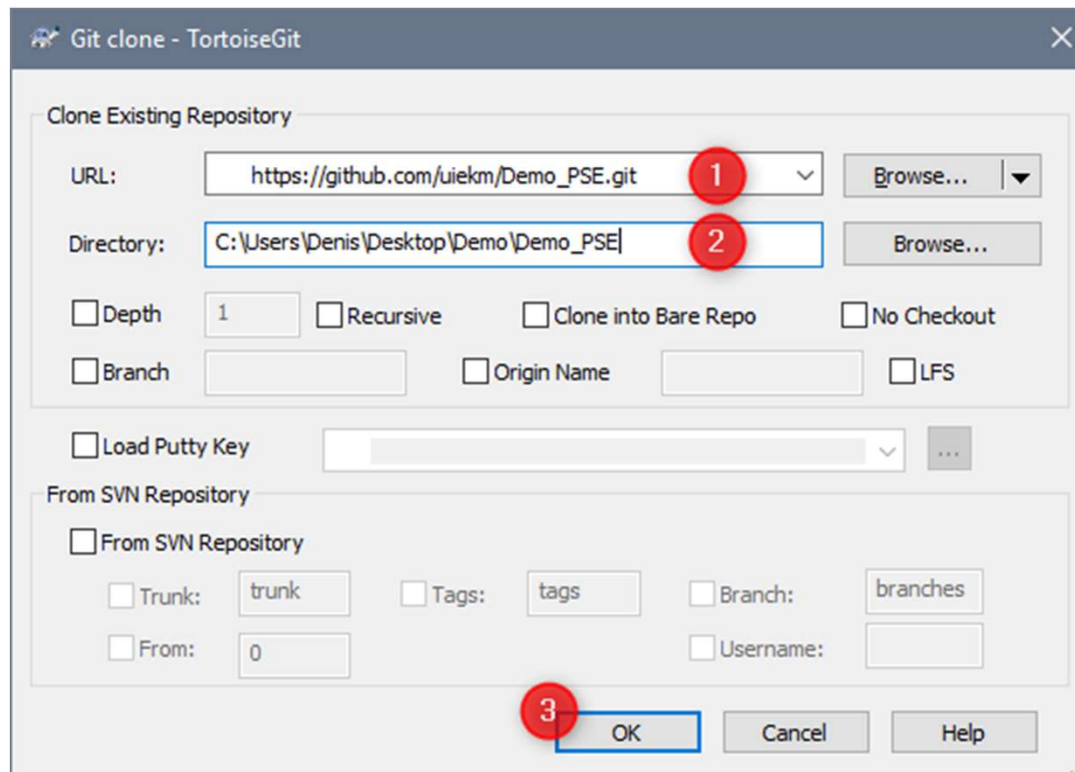
„Clone“

- erstmalig für lokale Kopie eines Repos
 - rechtecklick an gewünschter Stelle
 - wähle TortoiseGit
 - „clone...“



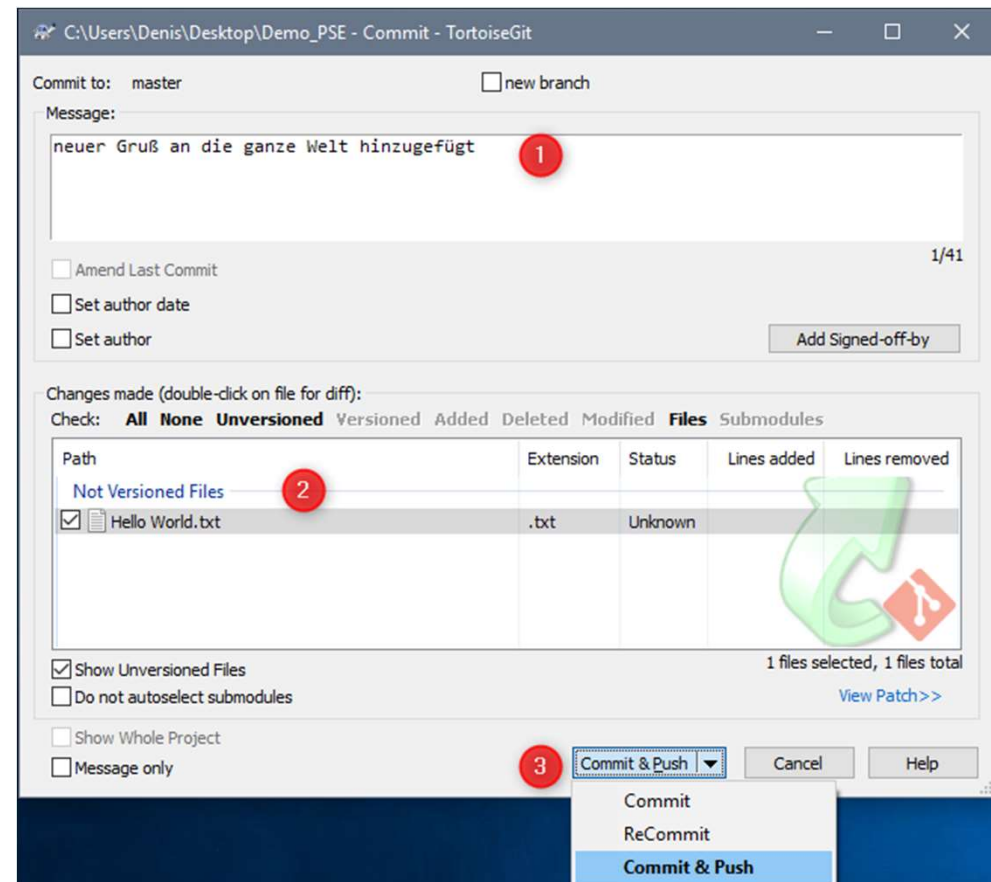
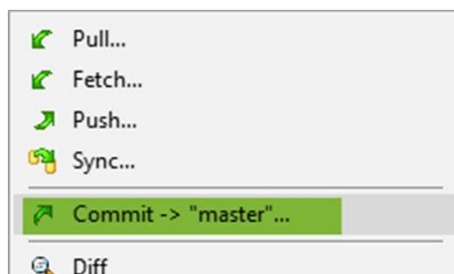
„Clone“

- jetzt remote repo-Pfad angeben
- ggf. lokalen repo-Pfad nochmal anpassen



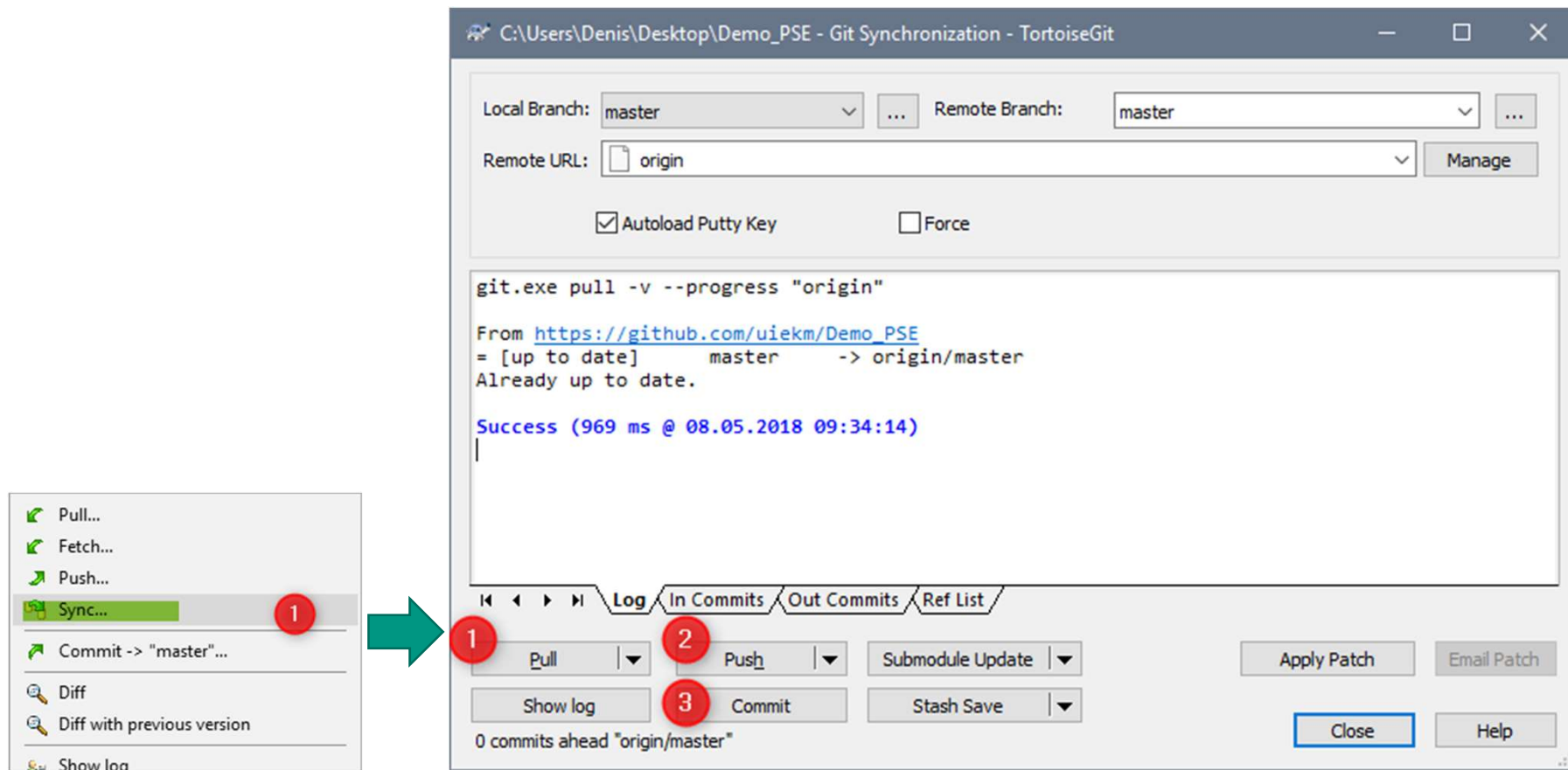
„Commit“

- Nach Änderungen commit durchführen!
 - Sinnvoller Kommentar passend zur Änderung!
 - Dateien auswählen
 - Commit & Push
 - standardmäßig angewählt
 - Versioniert lokal + remote



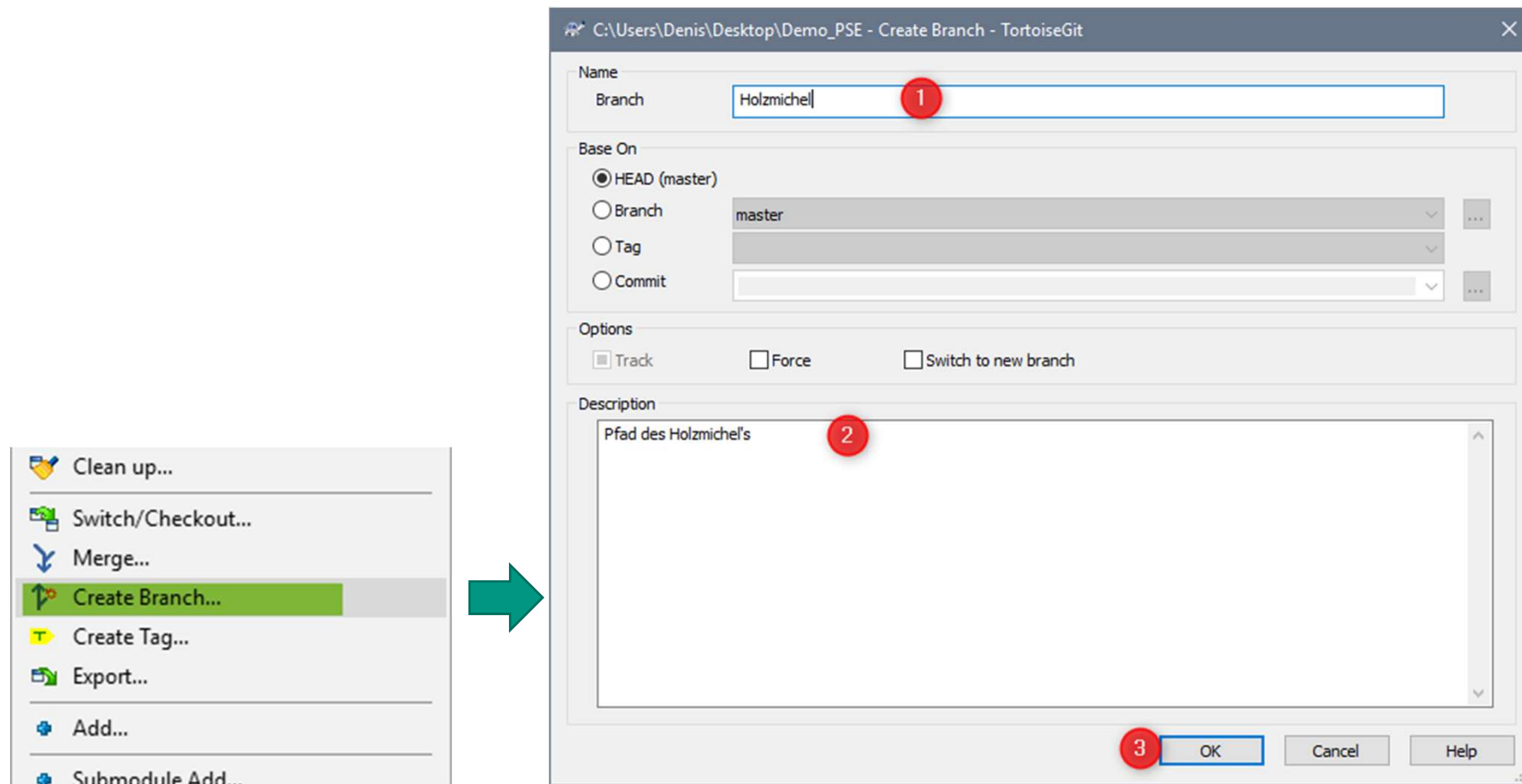
„Sync“

- Ein Fenster für viele Funktionen, z.B. „Pull“ „Push“



„Create Branch“

- Neuen Zweig anlegen (pro Entwickler ein Pfad)

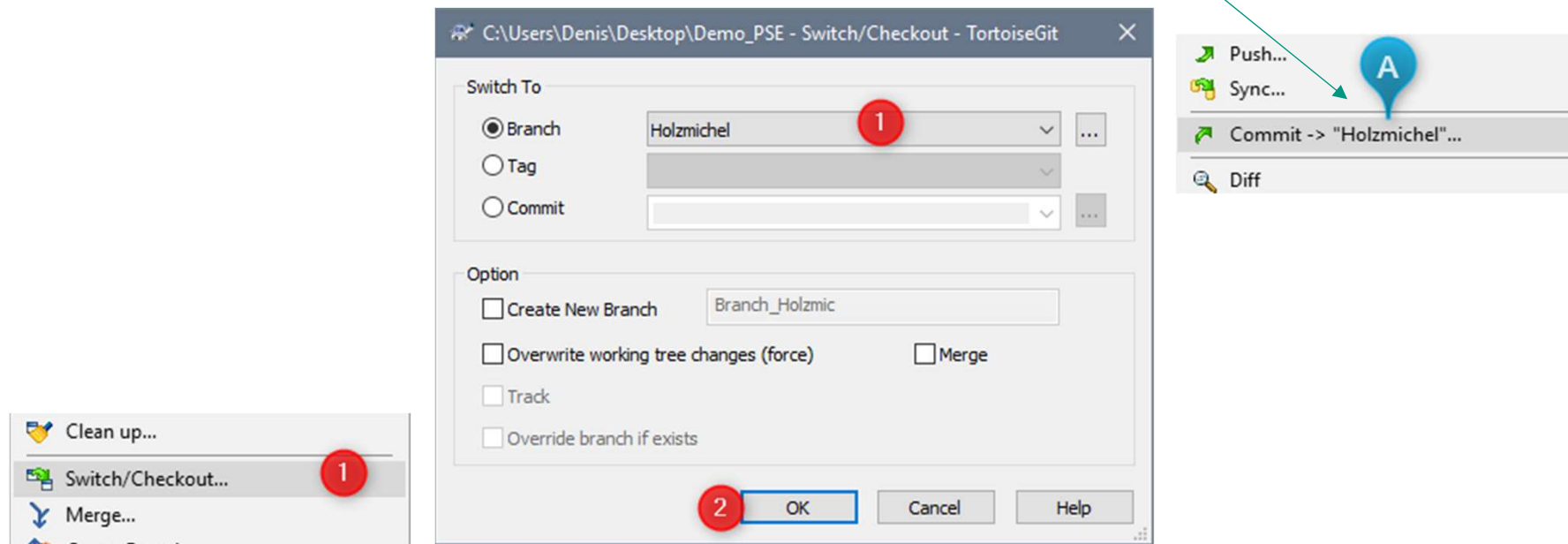


„Switch/Checkout“

■ Wechselt den Zweig

HINWEIS: Der aktuell angewählte Zweig, steht immer nach dem Pfeil bei Commit Kommando!

■ Jetzt Wechseln auf Pfad Holzmichel:



The image shows the TortoiseGit interface with two main components highlighted by red circles and arrows:

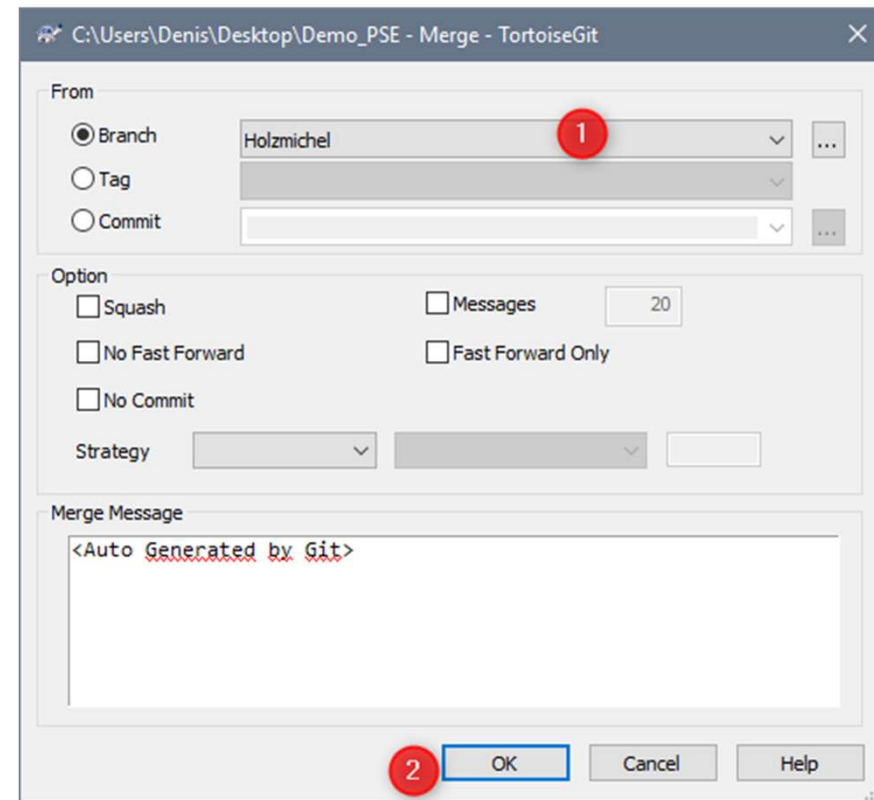
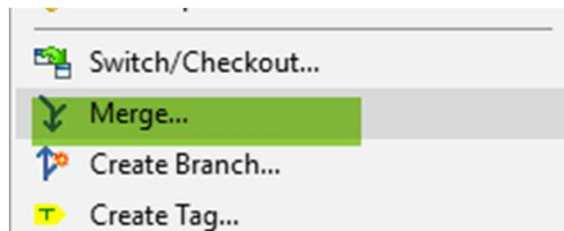
- Top Left (Menu):** A context menu with options: Clean up..., Switch/Checkout... (marked with a red circle '1'), Merge..., and Create Branch...
- Top Right (Menu):** A context menu with options: Push..., Sync..., Commit -> "master"... (marked with a red circle 'A'), and Diff.
- Center (Dialog Box):** A dialog box titled "C:\Users\Denis\Desktop\Demo_PSE - Switch/Checkout - TortoiseGit". It has two sections:
 - Switch To:** Radio buttons for Branch, Tag, and Commit. The Branch option is selected, and the dropdown menu shows "Holzmichel" (marked with a red circle '1').
 - Option:** Checkboxes for "Create New Branch" (with text "Branch_Holzmich"), "Overwrite working tree changes (force)", "Merge", "Track", and "Override branch if exists".
- Bottom (Buttons):** Buttons for OK (marked with a red circle '2'), Cancel, and Help.

Arrows indicate the workflow: from the "Switch/Checkout..." menu item to the dialog box, and from the "Commit -> 'Holzmichel'..." menu item to the "Commit" radio button in the dialog box.

„Merge“

■ Führt Zweige zusammen

- Wechsle dazu in den Zweig in den zusammengeführt werden soll
- Hier der Zweig „Holzmichel“ in „master“



FRAGEN

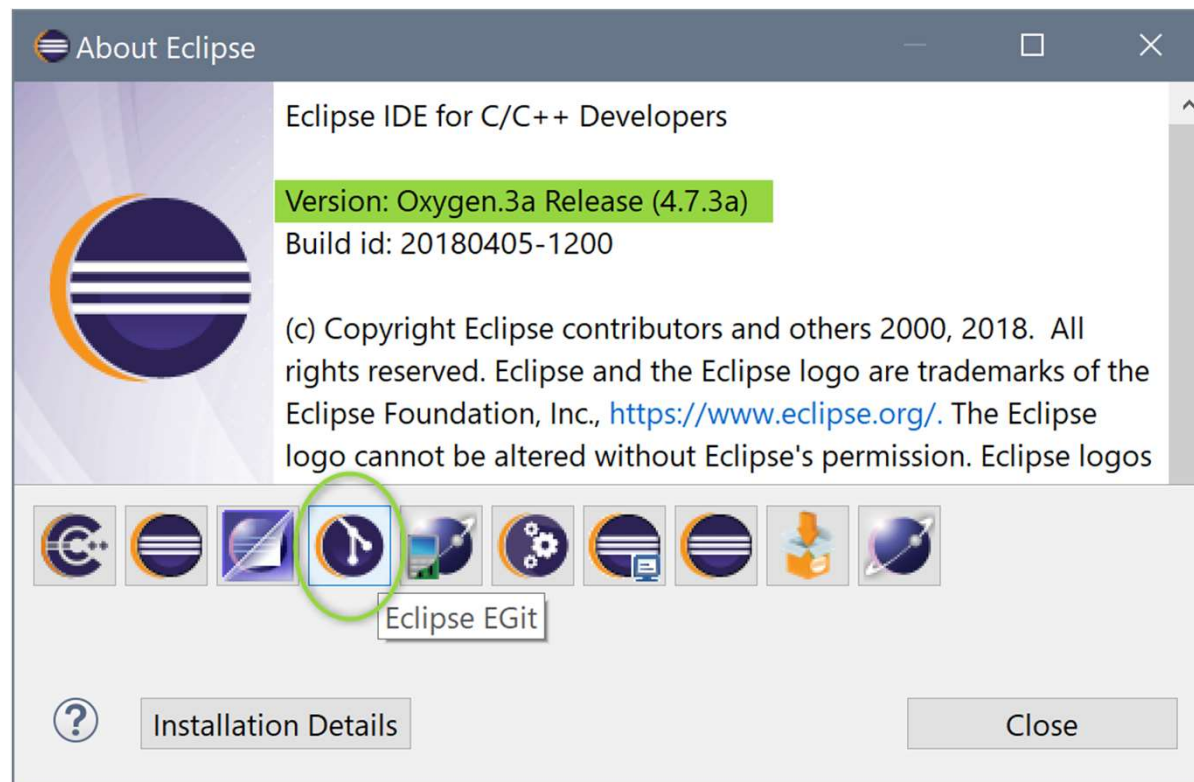


Integration von Git in Eclipse

Vorstellung und Nutzung

EclipseGit Vorstellung

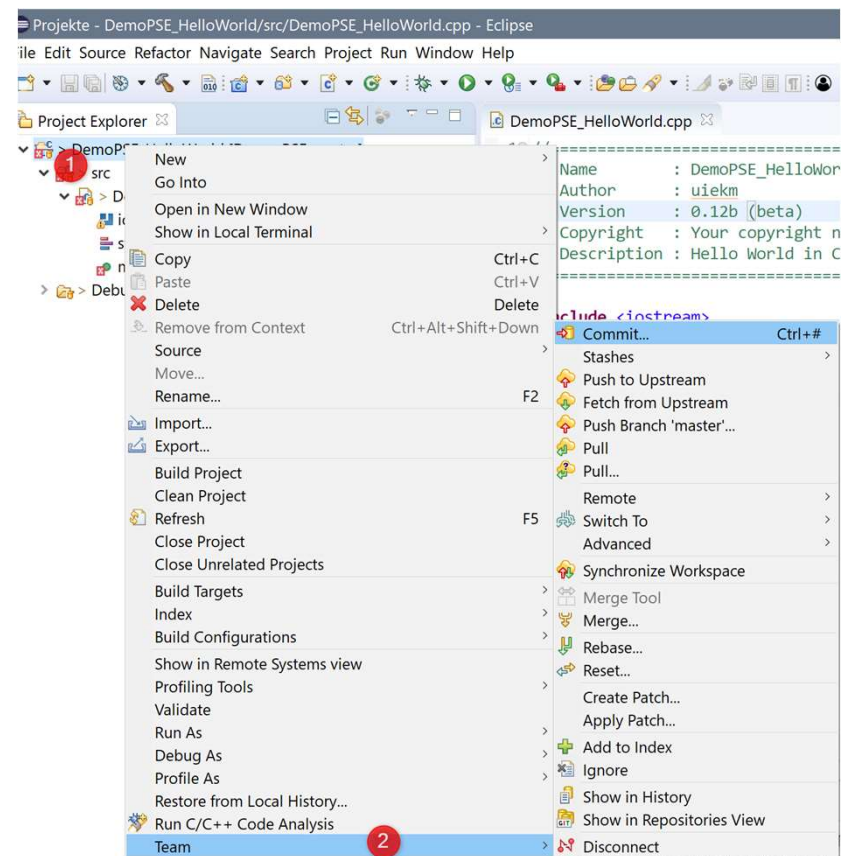
- Funktionen/Kommandos direkt in IDE¹
- Wird mit Eclipse Oxygen (hier 4.7.3) automatisch installiert



¹ *Integrated Development Environment*

EclipseGit Einrichtung

- Repository über „File“ ► „Import“ hinzufügen
- Anschließend sind Git Funktionen unter:
 - Rechtsklick im Project Explorer
 - „Team“ zu finden.



Live-Demo



FRAGEN

Vielen Dank für Eure Aufmerksamkeit



Quellen

■ Git

- <https://www.atlassian.com/git/tutorials/using-branches>
- <https://git-scm.com/book/de/v1/Git-Branching-Branching-Workflows>
- <https://infos.seibert-media.net/display/Productivity/Git-Workflows+-+Der+Gitflow-Workflow>
- <https://www.ralfebert.de/git/workflows/>

■ TortoiseGit

- <https://tortoisegit.org/docs/tortoisegit/>

■ Egit

- <http://www.vogella.com/tutorials/EclipseGit/article.html>

Bildquellen

- <http://i0.kym-cdn.com/photos/images/newsfeed/000/041/343/index.php20110724-22047-58b7hk.png>
- <https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow>
- <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>
- https://www.reddit.com/r/ProgrammerHumor/comments/7l3jtr/someday_all_hype_about_git_will_be_over/