

---

# Git/GitLab for R Users

DIW R User Group, June 7th, 2018

*Leibniz*

### Git: Distributed Version Control System

version control system:

- tracks changes in computer files (text files) and coordinates work on those files among multiple people

distributed:

- no central server needed
- but we use a GitLab server, which offers some nice features

### Git: (Not) Use Cases

- plain text files, like your R code (or tex files)
- explicitly declare a new version (“commit”) and include a “commit message”
- restore previous versions
- work simultaneously even on the same file
- no collaboration on binary files (e.g. Excel, Stata data, PDF)
  - this would result in “conflicts” which cannot easily be solved
- no need to be always online

## Git: Simple Workflow

		<code>git clone</code> ②		<code>git add</code> ⑤ <code>git commit</code> <code>git push</code>			<code>git pull</code> ⑫		<code>git add</code> ⑮ <code>git commit</code> <code>git push</code>
Jane		●	● ● ④	● ●	● ●		● ● ● ⑬ ● ● ● ⑪	● ● ● ⑭ ● ● ●	● ● ● ⑯ ● ● ●
GitLab	● ①	●	●	● ● ⑥	● ●	● ●	● ● ● ⑪ ● ● ● ⑬	● ● ● ● ● ●	● ● ● ⑯ ● ● ●
Bob		●	●	● ● ⑦ ● ●	● ●	● ● ●	● ● ● ● ● ●	● ● ● ● ● ●	● ● ● ● ● ●
		<code>git clone</code> ③			<code>git add</code> ⑧ <code>git commit</code> <code>git push</code>	<code>git pull</code> ⑨	<code>git push</code> ⑩		

### Installation

- Git and DiffMerge via green puzzle
- some initial [configuration](#), also for [DiffMerge](#)
- no additional account needed for [git.soep.de](https://git.soep.de)
  - use windows credentials
  - if external collaborators need account: email

## Use

- start program “Git Bash”
- cd: change directory (use tab for paths and file names)
- URL for clone displayed by GitLab
- ls -l: **l**is folder content

```
MINGW64:/d/lokal/diw-r-meetings

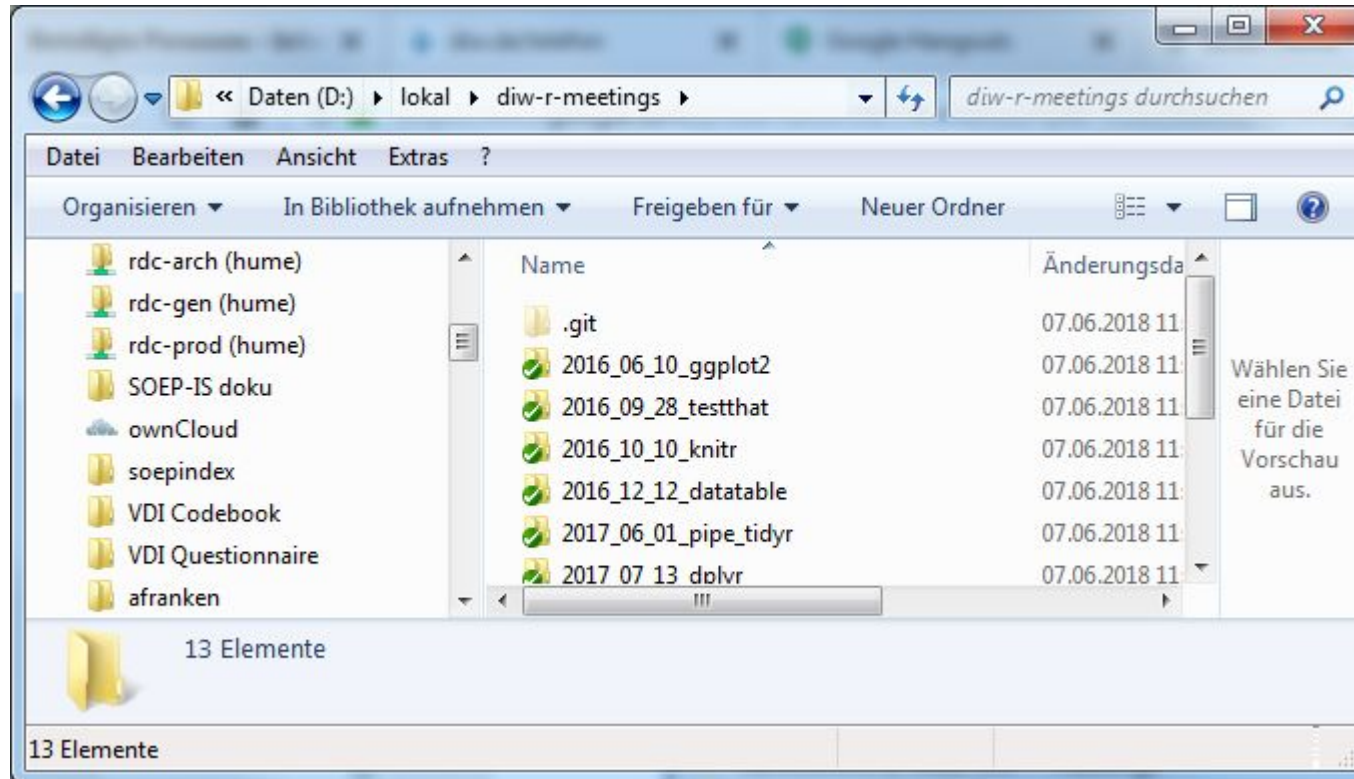
kwenzig@DIW-15-032 MINGW64 /
$ cd /d/lokal/

kwenzig@DIW-15-032 MINGW64 /d/lokal
$ git clone https://git.soep.de/DIW-R-Users/diw-r-meetings.git
Cloning into 'diw-r-meetings'...
remote: Counting objects: 112, done.
remote: Compressing objects: 100% (103/103), done.
remote: Total 112 (delta 19), reused 0 (delta 0)
Receiving objects: 100% (112/112), 8.07 MiB | 11.78 MiB/s, done.
Resolving deltas: 100% (19/19), done.

kwenzig@DIW-15-032 MINGW64 /d/lokal
$ cd diw-r-meetings/

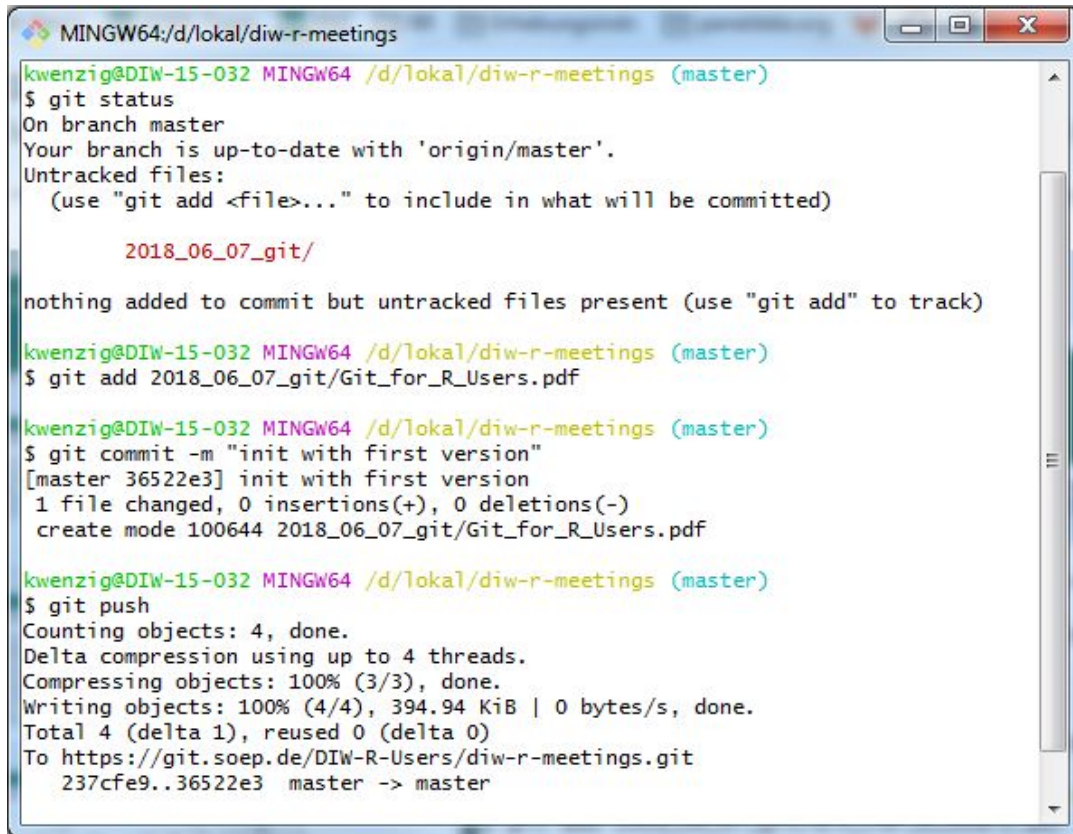
kwenzig@DIW-15-032 MINGW64 /d/lokal/diw-r-meetings (master)
$ ls -l
total 16
drwxr-xr-x 1 kwenzig 1049089 0 Jun  7 11:49 2016_06_10_ggplot2/
drwxr-xr-x 1 kwenzig 1049089 0 Jun  7 11:49 2016_09_28_testthat/
drwxr-xr-x 1 kwenzig 1049089 0 Jun  7 11:49 2016_10_10_knitr/
drwxr-xr-x 1 kwenzig 1049089 0 Jun  7 11:49 2016_12_12_datatable/
drwxr-xr-x 1 kwenzig 1049089 0 Jun  7 11:49 2017_06_01_pipe_tidy/
```

# Using Git/GitLab



## Use

- git status: always a good idea
- git add <file>: add to staging area
- git commit: make commit, add an commit message
- git push: transfer to server



```
MINGW64/d:/lokal/diw-r-meetings
kwenzig@DIW-15-032 MINGW64 /d:/lokal/diw-r-meetings (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

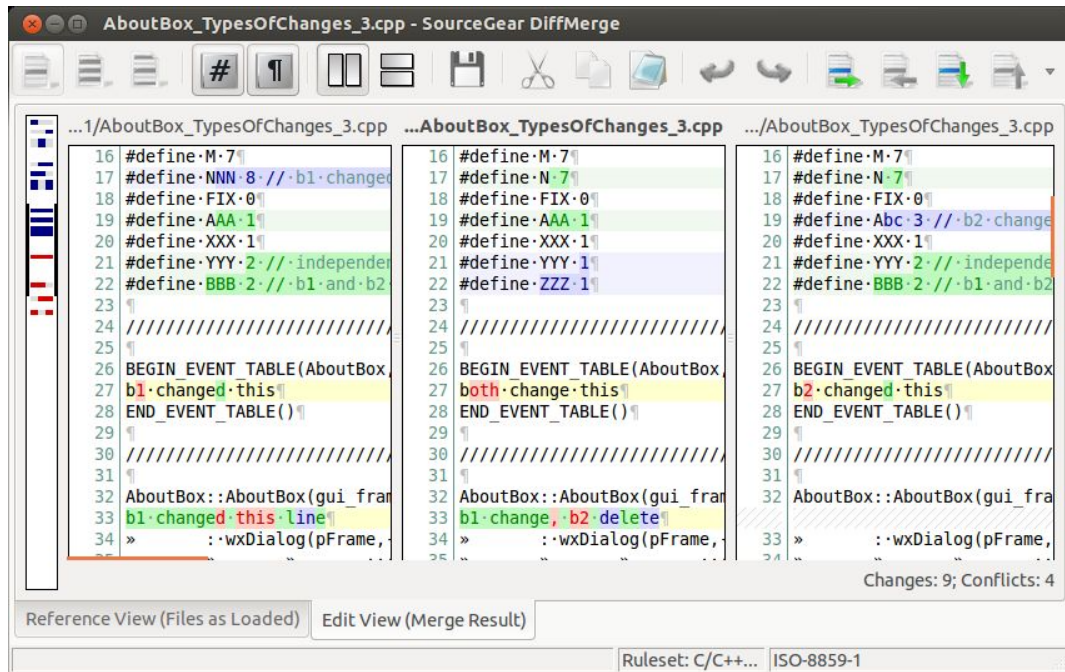
        2018_06_07_git/

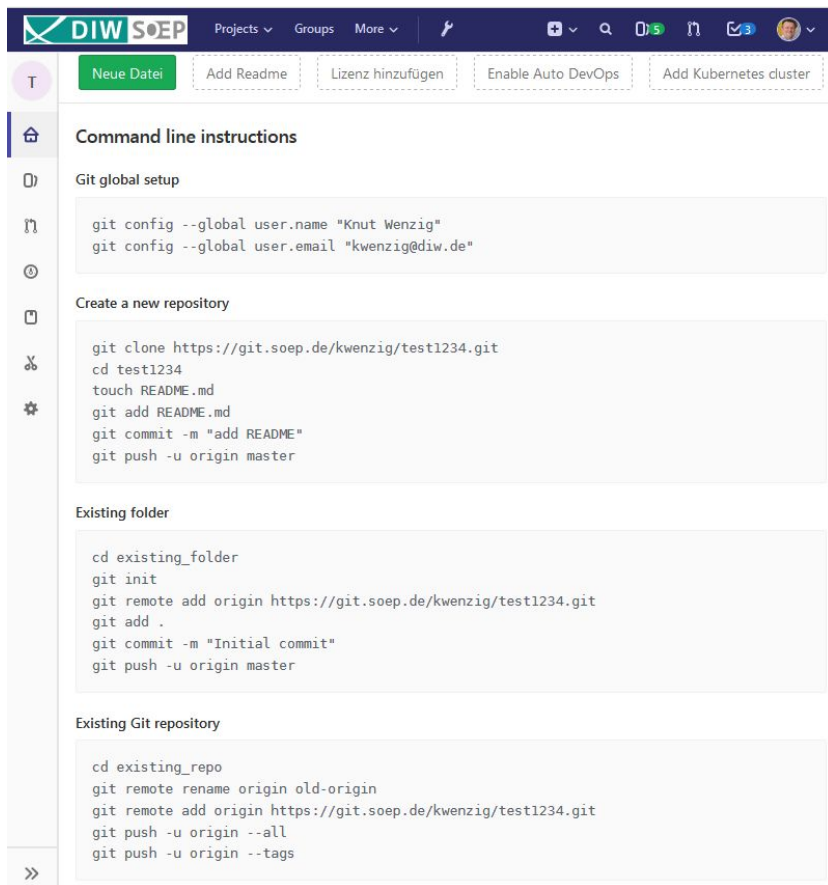
nothing added to commit but untracked files present (use "git add" to track)
kwenzig@DIW-15-032 MINGW64 /d:/lokal/diw-r-meetings (master)
$ git add 2018_06_07_git/Git_for_R_Users.pdf
kwenzig@DIW-15-032 MINGW64 /d:/lokal/diw-r-meetings (master)
$ git commit -m "init with first version"
[master 36522e3] init with first version
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 2018_06_07_git/Git_for_R_Users.pdf
kwenzig@DIW-15-032 MINGW64 /d:/lokal/diw-r-meetings (master)
$ git push
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 394.94 KiB | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
To https://git.soep.de/DIW-R-Users/diw-r-meetings.git
   237cfe9..36522e3  master -> master
```



## Use

- git pull: get recent changes from server
- solve possible - but rare - conflicts with DiffMerge (git mergetool)





The screenshot shows the GitLab web interface for a user named Knut Wenzig. The top navigation bar includes the DIW SOEP logo, a sidebar menu, and a top bar with navigation links (Projects, Groups, More), search, and user profile. Below the top bar, there are buttons for 'Neue Datei', 'Add Readme', 'Lizenz hinzufügen', 'Enable Auto DevOps', and 'Add Kubernetes cluster'. The main content area is titled 'Command line instructions' and contains three sections of code:

```
Git global setup

git config --global user.name "Knut Wenzig"
git config --global user.email "kwenzig@diw.de"

Create a new repository

git clone https://git.soep.de/kwenzig/test1234.git
cd test1234
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master

Existing folder

cd existing_folder
git init
git remote add origin https://git.soep.de/kwenzig/test1234.git
git add .
git commit -m "Initial commit"
git push -u origin master

Existing Git repository

cd existing_repo
git remote rename origin old-origin
git remote add origin https://git.soep.de/kwenzig/test1234.git
git push -u origin --all
git push -u origin --tags
```

## Start project

- press button “New project”
- choose project name
- look at “Command line instructions”

### Best Practises

- start always with git pull (avoids conflicts)
- git status often gives a good advice
- read the feedback from git for your commands
  - if you do not use branches there should always be (master)
- atomic commits are better to understand and to undo than one commit with the work of a whole day
- only use git add --all or git commit -a if you know what you're doing (e.g. after git status)

### Best Practises 2

- use meaningful commit messages
  - refer to issues, e.g. #1, or even close issues by message
- think about your workflow
- only push working code

### Help

- some basic remarks at the [Wiki](#) of project GitAtSOEP
- Book “Pro Git” ([en](#), [de](#))
- Atlassian [Git Tutorial](#), e.g. for [undoing changes](#)
- git - the simple guide ([en](#), [de](#))
- [Think Like \(a\) Git](#)
- interactive: <https://learngitbranching.js.org/>

### Not covered

- branches: you can play around in a branch and merge this back to the master branch ([feature branch](#))
- [forks and merge requests](#) to control what gets into your repository
- tags, which are an additional info for a commit, like “final”
- GitLab features (compare versions, wiki, issue boards)
- Integration in RStudio and other GUIs