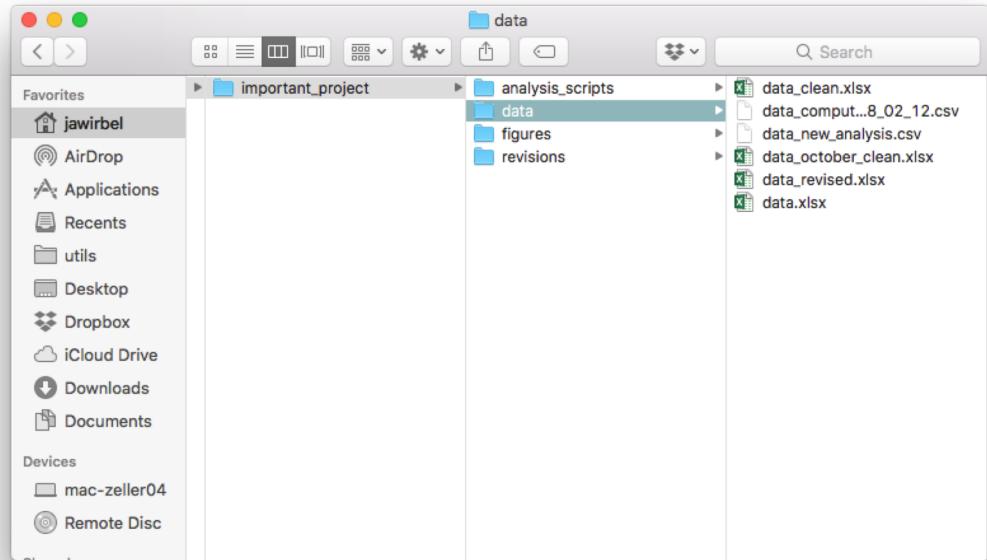
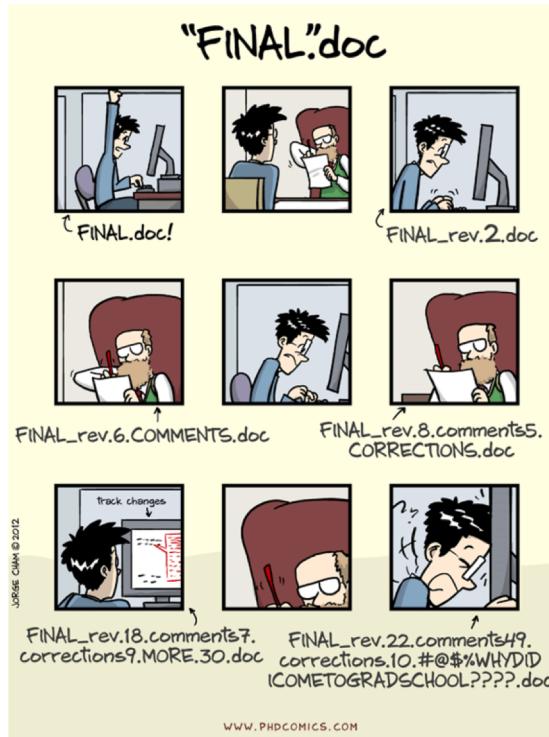


Version control with Git

BTM 2018



What is the problem?



What is the solution?



When to use git?

- Code. Version control was made for code.
- Manuscripts
- Collaborative projects
- Everything

When **not** to use git?

- Short-term stuff

How to start?

- Can also be done in the command line
- You can also start a new git repository with already existing files

The screenshot shows the GitHub interface for creating a new repository. At the top, there are navigation links: Pull requests, Issues, Marketplace, and Explore. Below this, the main title is "Create a new repository" with a subtitle: "A repository contains all the files for your project, including the revision history." A user profile picture and the text "jakob-wirbel" are shown next to a dropdown menu. The repository name is "git_btm_2018", which is highlighted with a green checkmark. A note below says, "Great repository names are short and memorable. Need inspiration? How about [fantastic-guacamole](#)." There is a "Description (optional)" field with a placeholder text area. Below that, there are two radio button options: "Public" (selected) and "Private". The "Public" option is described as "Anyone can see this repository. You choose who can commit." The "Private" option is described as "You choose who can see and commit to this repository." A checked checkbox labeled "Initialize this repository with a README" is present, with a note explaining it allows immediate cloning. Below the checkbox are buttons for "Add .gitignore: None" and "Add a license: None". At the bottom is a large green "Create repository" button. The footer of the page includes copyright information (© 2018 GitHub, Inc.), links to Terms, Privacy, Security, Status, Help, and social media icons, along with links to Contact GitHub, Pricing, API, Training, Blog, and About.

git clone

```
[ -bash-4.2$ git clone https://github.com/jakob-wirbel/git_btm_2018.git
Cloning into 'git_btm_2018'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
-bash-4.2$ █
```

git status

```
[ -bash-4.2$ git status  
# On branch master  
nothing to commit, working directory clean  
-bash-4.2$ ]
```

Let's change something

```
[ -bash-4.2$ nano README.md
[ -bash-4.2$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   README.md
#
no changes added to commit (use "git add" and/or "git commit -a")
```

git diff

```
[~] bash-4.2$ git diff README.md
diff --git a/README.md b/README.md
index 6086c6e..2bce39c 100644
--- a/README.md
+++ b/README.md
@@ -1,3 +1,6 @@
 # git_btm_2018

 This repository contains the presentation introducing `git` in the **B**as
+
+If you know `git` already, please go and check out the [this
+guide](https://swcarpentry.github.io/git-novice/).
[~] bash-4.2$
```

git diff

git add

```
[ -bash-4.2$ git add README.md
[ -bash-4.2$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README.md
#
-bash-4.2$ ]
```

git commit

```
[ -bash-4.2$ git commit -m 'Updated README.md'  
[master 98dc7aa] Updated README.md  
 1 file changed, 3 insertions(+)
```

- `git commit -m 'Message'`
- The message should be informative to others (including you in four months!)

git commit

```
[ -bash-4.2$ git commit -m 'Updated README.md'
[master 98dc7aa] Updated README.md
 1 file changed, 3 insertions(+)
- bash-4.2$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#   (use "git push" to publish your local commits)
#
nothing to commit, working directory clean
- bash-4.2$ ]
```

Pushing & Pulling

```
[ -bash-4.2$ git push
[Username for 'https://github.com': jakob-wirbel
[Password for 'https://jakob-wirbel@github.com':
Counting objects: 5, done.
Delta compression using up to 64 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 452 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/jakob-wirbel/git_btm_2018.git
  e25edd9..98dc7aa  master -> master
-bash-4.2$
```

Command	What does it do?
git clone <i>url</i>	Download a project and its entire history
git status	Lists all new modifications (not yet committed)
git diff <i>filename</i>	Show what has been changed in this file
git add <i>filename</i>	Snapshot this file
git commit –m ‘Message’	Record the snapshot forever
git push	Uploads all commits to the remote repo
git pull	Download all changes since last pull

Where to go from here?

- [Software Carpentry Course on Git](#)
- [Resources to learn GitHub](#)
- [GitHub Documentation](#)

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.



The 99%

- git is very powerful, but hard to learn
- You can do so much more:
 - ignore files in your repository
 - create and merge branches
 - rewrite history, undo mistakes
 - work collaboratively on projects



Hands on

- Organize in pairs
- One of you initiate a repository in Github or git.embl.de
- Add the other person as contributor
- One of you
 - add a file
 - commit it
 - push it
- The other one
 - pull the changes
 - edit the file
 - push it

Command

git clone *url*

git status

git diff *filename*

git add *filename*

git commit –m ‘Message’

git push

git pull