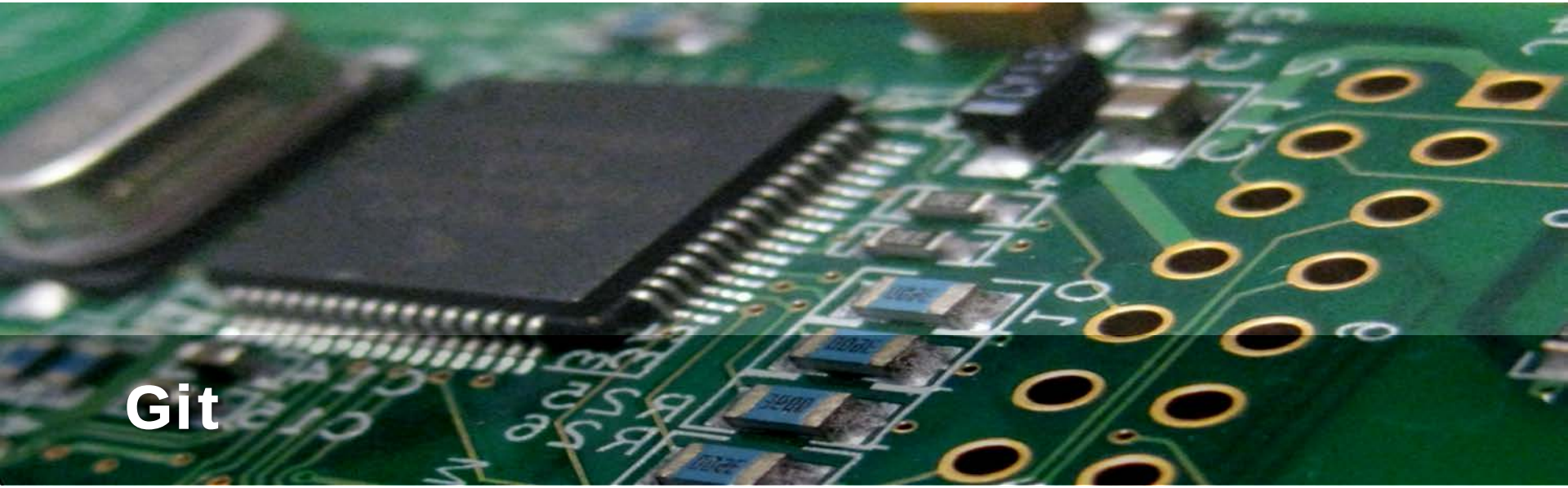
Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Technik & Architektur



Git

"Git is just..... git ☺"

Prof. Erich Styger
erich.styger@hslu.ch
+41 41 349 33 01

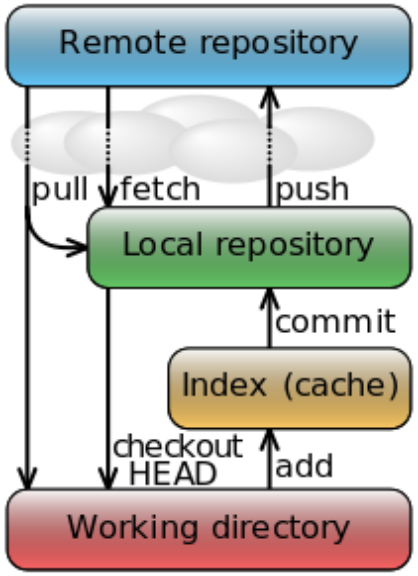
Lucerne University of Applied Sciences and Arts
HOCHSCHULE LUZERN

About Git

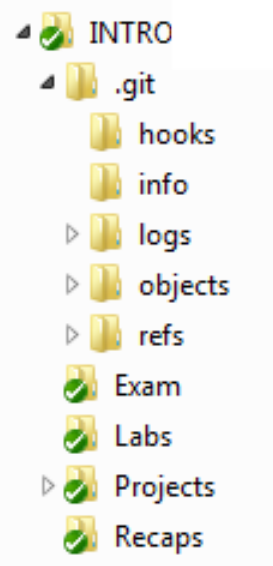
- By Linus Torvalds (GPL2)
- Local repository (.git, Metadata)
- Working tree: Files/Folders on .git level

- **Index**: collection of 'added' and file changes ➔ staged
- **Staged** and unstaged files can co-exist
- 'commit': ➔ .git/objects
- 'push'/'synch'

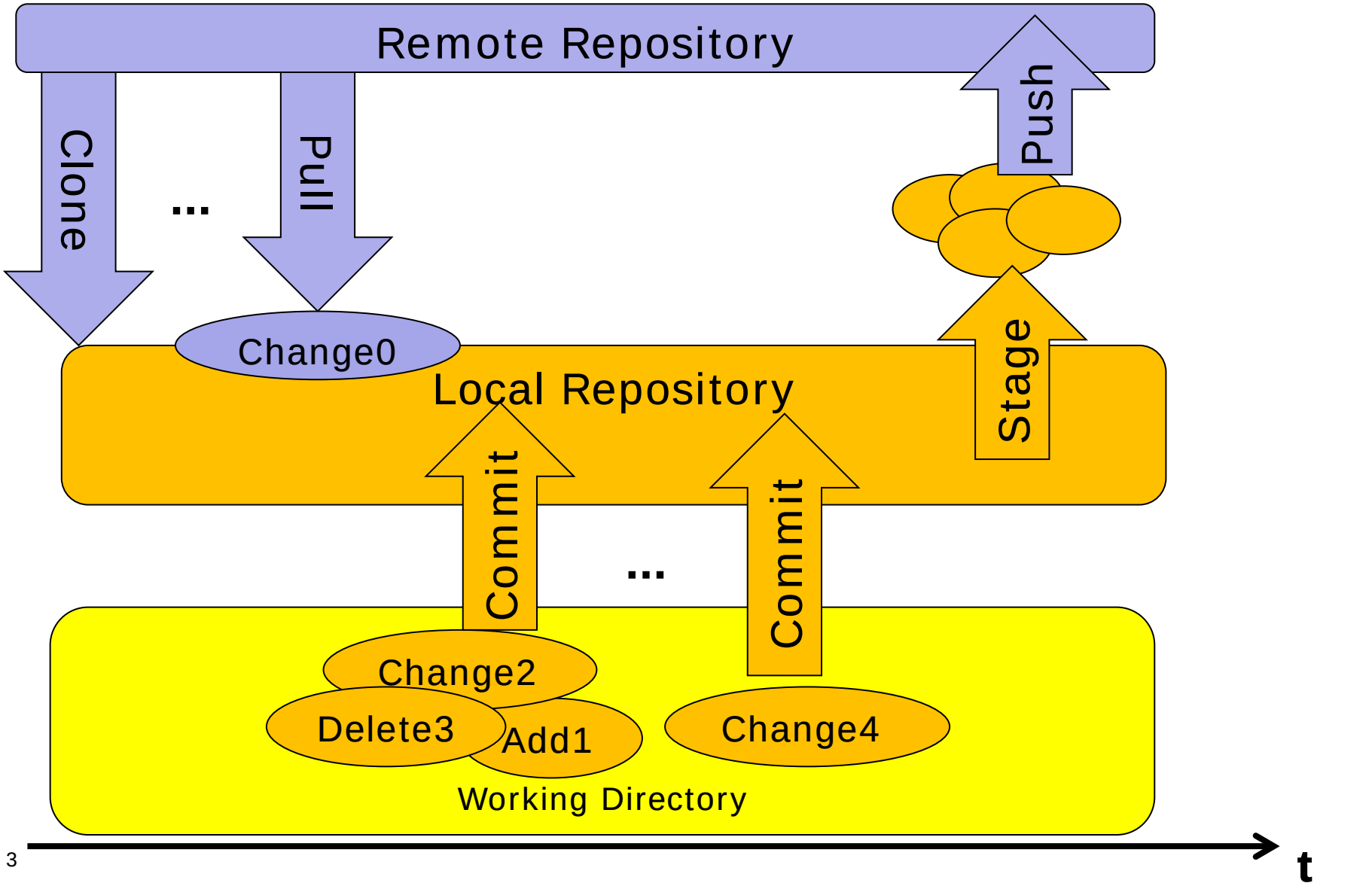
- SVN: revision numbers
- Git: SHA1 (160bit) hex numbers
  - 77f76087ddeb12d9005d6676a1d54085f232a32f



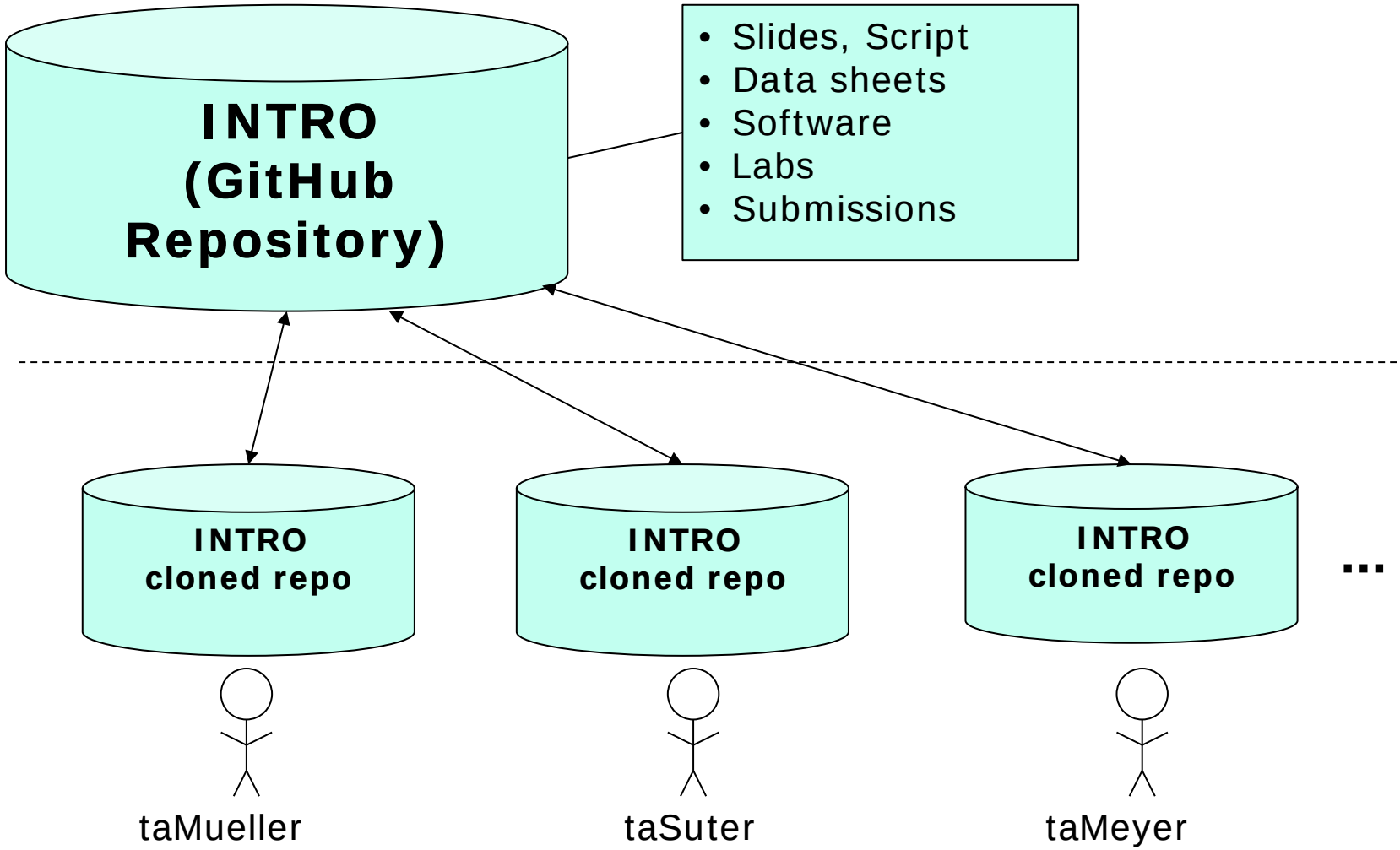
Source: Wikipedia



Typical Git Workflow



INTRO Git Setup

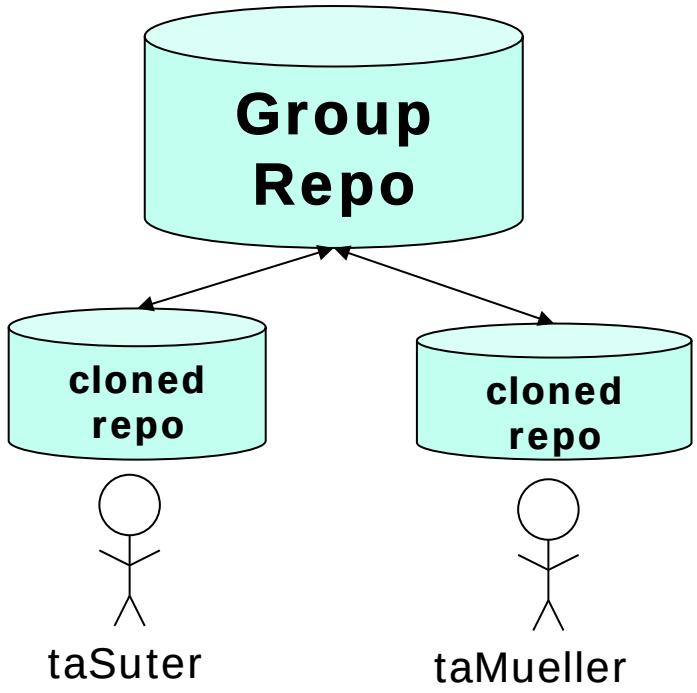


Group Repository Setup

- Public (free) or Private (paid) repo

- Public
  - Everyone on the Internet can see it
- Private
  - Owner controls access

- Your group account
  - Group repo on GitHub
  - Or: add Files to INTRO repo
    - Projects\Groups



Git Clients

- TortoiseGit
  - Explorer extension
  - Pros: easy for SVN users, good user interface, powerful
  - Cons: difficult password caching
- **eGit**
  - Eclipse plugin
  - Pros: development flow integration
  - Cons: only for 'simple' tasks
- **SourceTree or GitHub Desktop**
  - Standalone application
  - Multiple Repositories in one view

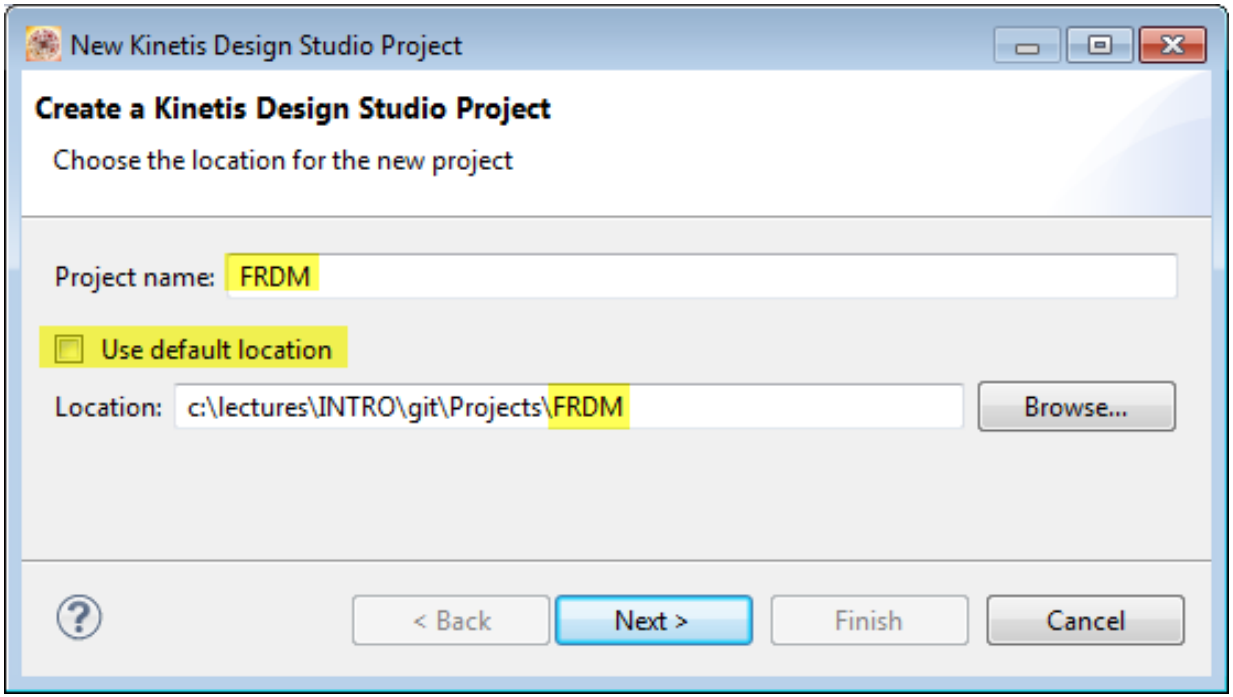
- Many different clients available ☺

Eclipse Workspace, Projects & Repos

- **Workspace**
  - Contains Eclipse settings (.metadata folder)
    - Which projects, tabs, colors, windows, ...
    - List of projects
  - Recommendation: name it e.g. 'wsp_Intro'
- **Project folders** can be 'anywhere'
  - By default, projects are created in workspace
  - Recommendation: separate workspace and project folders, especially if working with version control systems
- Example:
  - Projects in
    - C:\Lectures\INTRO**git\Projects**
  - Wsp in
    - C:\Lectures\INTRO**wsp_Intro**

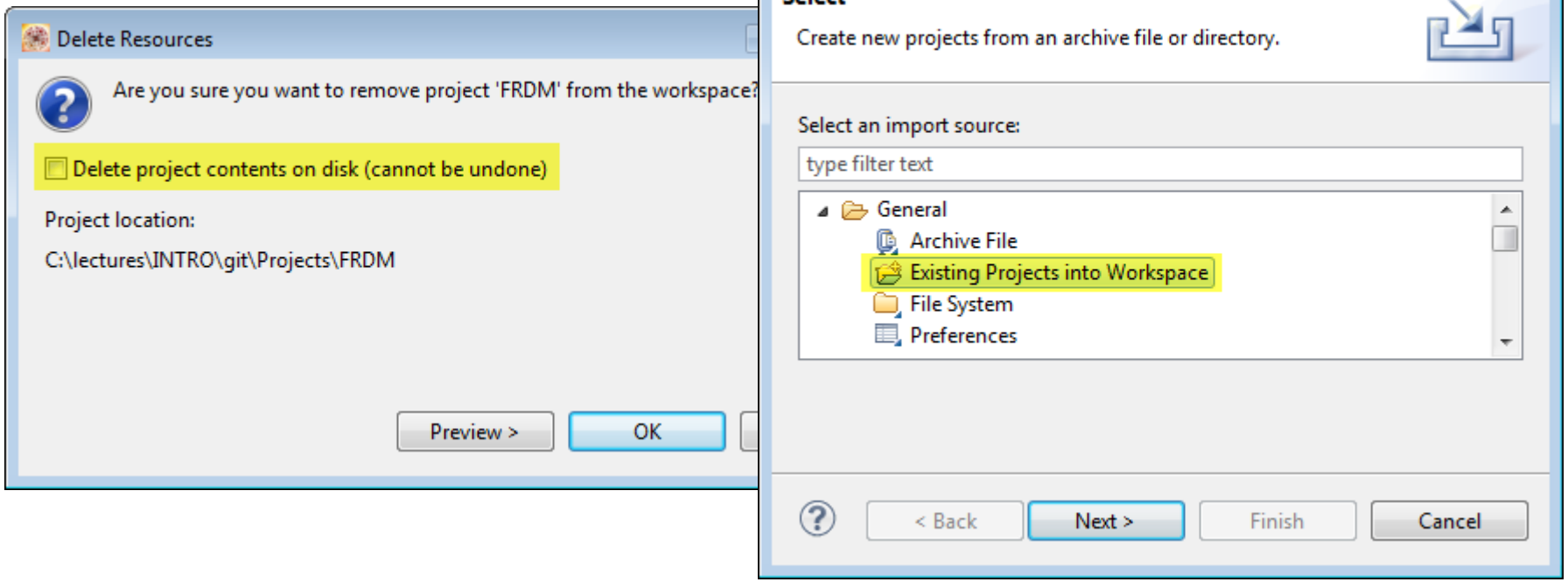
Eclipse Project Creation Outside Workspace

- Cannot create project in non-empty folder!
- Project Creation outside workspace
  - Give Project Name
  - Uncheck 'use default location'
  - Enter path with project name at the end
  - Will create project in that folder



Eclipse Project Move (to Git Location)

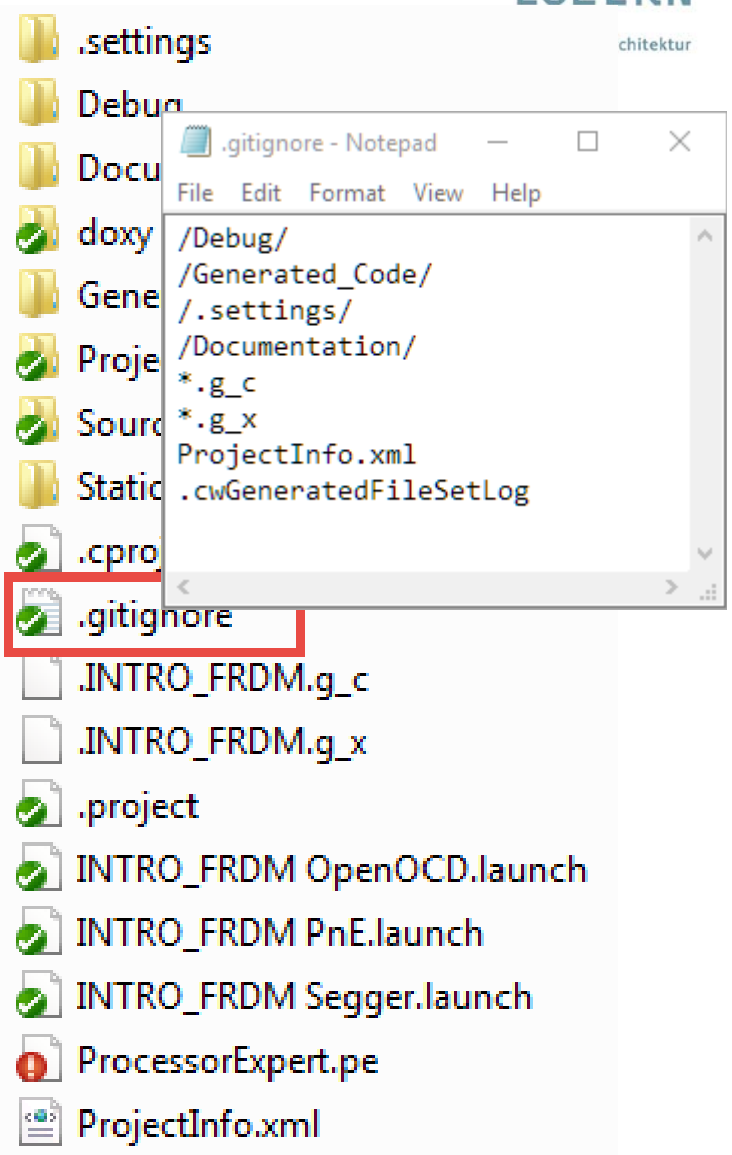
- Delete project (but NOT from disk)
- Move project to git project folder
- Add Project again with *File > Import > General Existing Projects*



Ignoring files

- ***.gitIgnore*** File ignores files and folders
- Multiple levels
  - Whole repository (root)
  - On Folder level (inside repository)
- Recommendation: .gitignore inside project
- Copy existing .gitignore

- NOTE: path in .gitignore file is *relative* to ignore file location

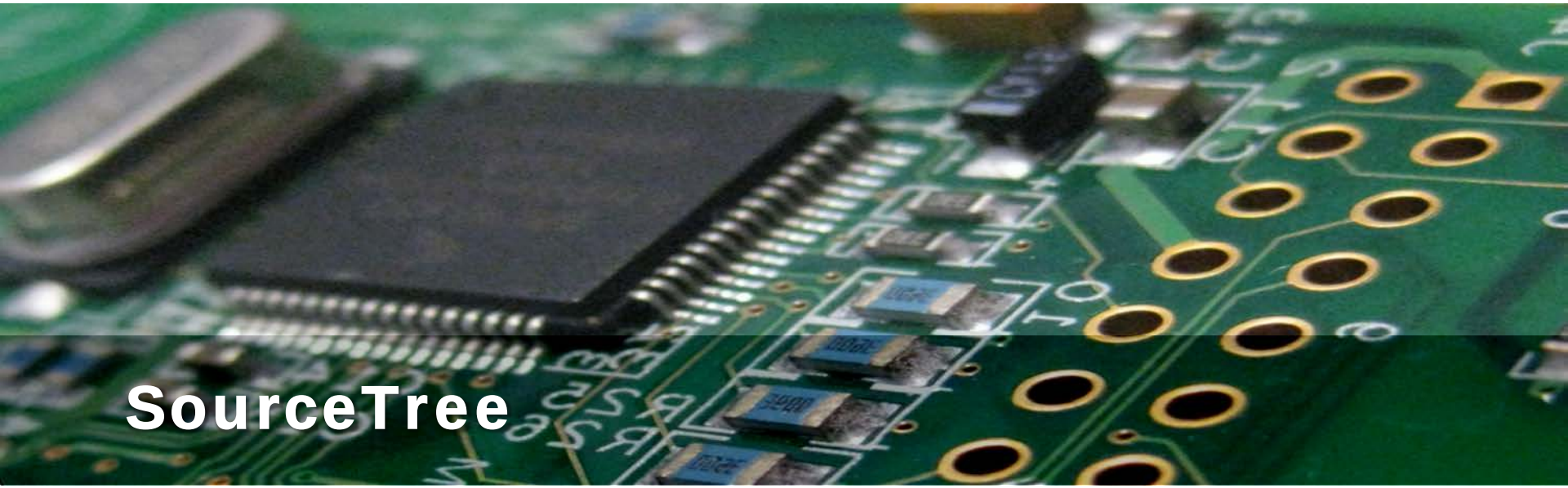


.gitignore Format

- Processed top to bottom
- Blank lines: ignored
- #: starts comment
  ```
  # this is a comment
  ```
- Linie item: ignore file/folder/pattern
- !: negates the pattern
  ```
  /!myFile.txt
  ```

```
# ignore generated code and all text files …
/Generated_Code
/*.txt
# … except 'readme.txt'
!/readme.txt
```

Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Technik & Architektur



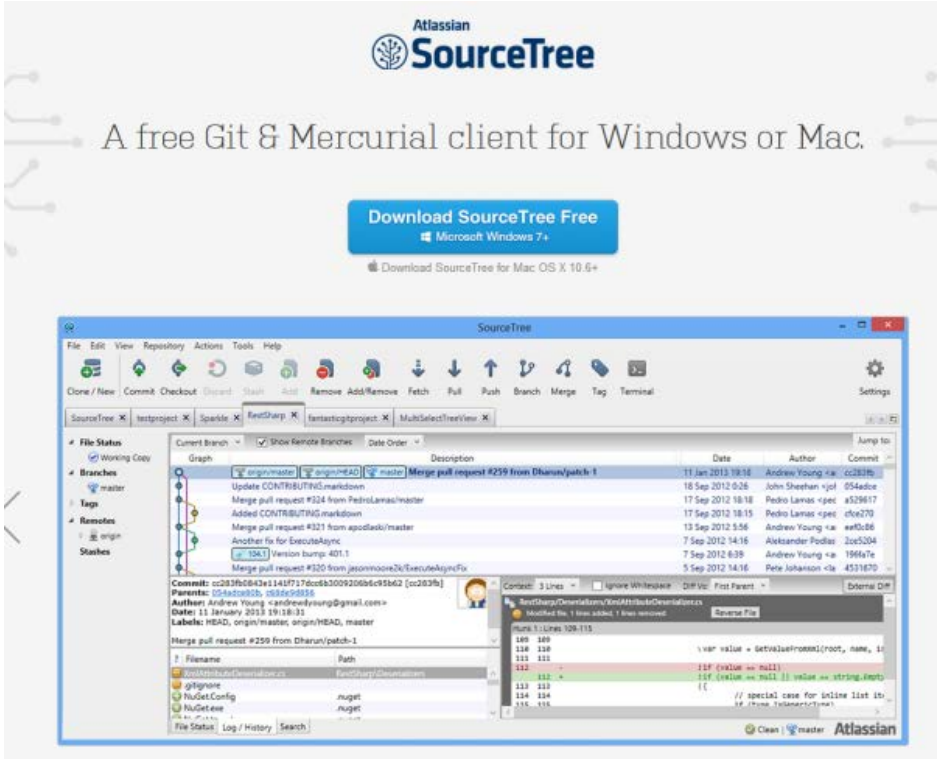
SourceTree

"A nice client would be nice."

Prof. Erich Styger
erich.styger@hslu.ch
+41 41 349 33 01

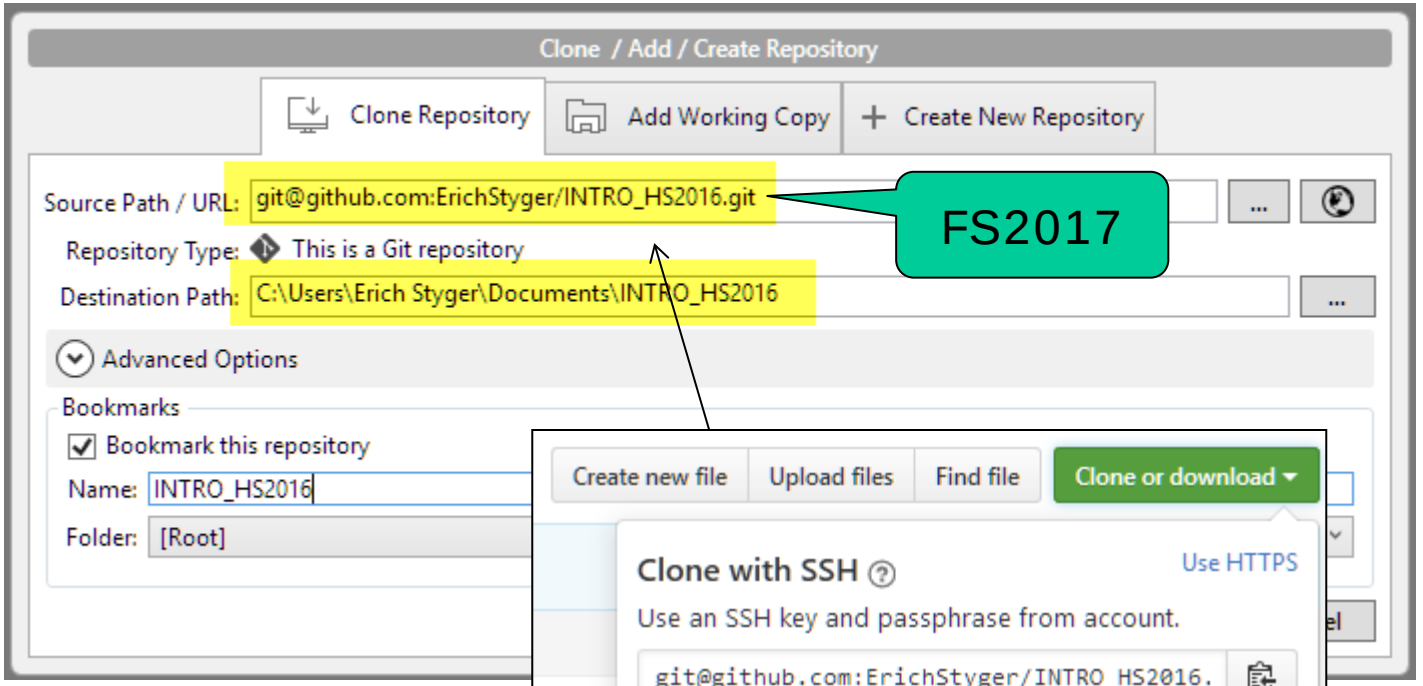
Installation

- Download latest version from web
- https://www.atlassian.com/software/sourcetree/overview
- http://sourcetreeapp.com/
- (free) registration required (reminder after 30 days)



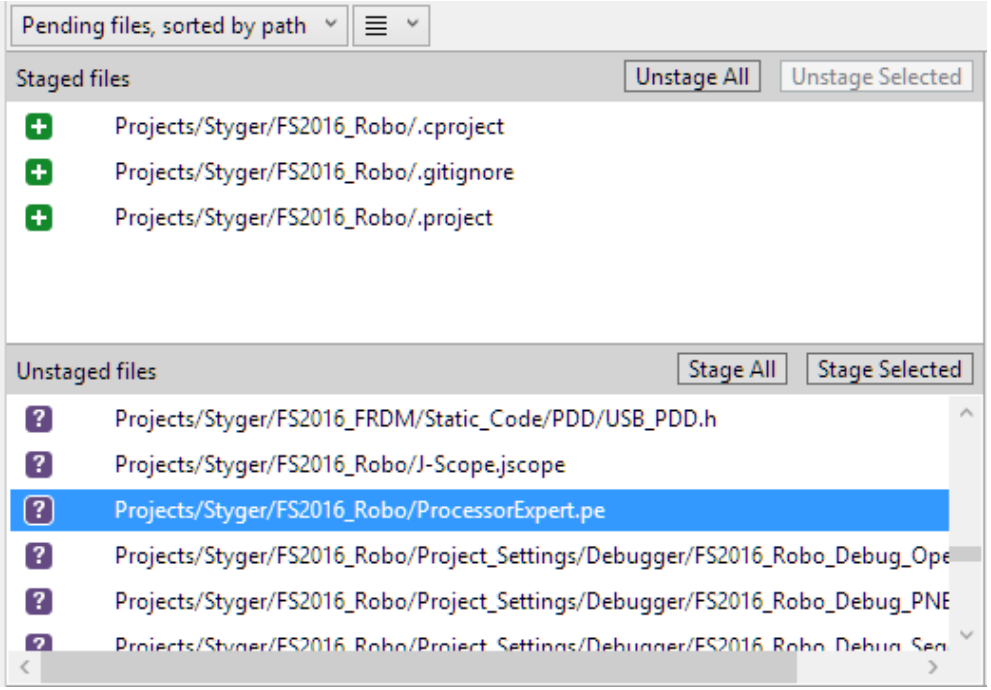
Adding Repo in SourceTree

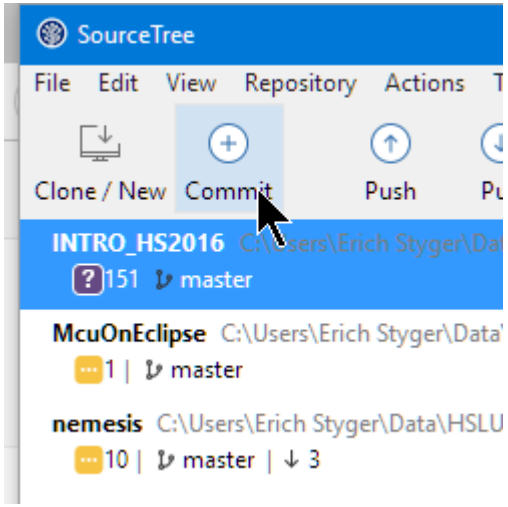
- Create folder where you want to have the local repo(s)
  - E.g. C:\Lectures\git\INTRO
- Clone Repo (File > Clone)
  - Copy URL from GitHub to Clipboard



Staging & Commit (SourceTree)

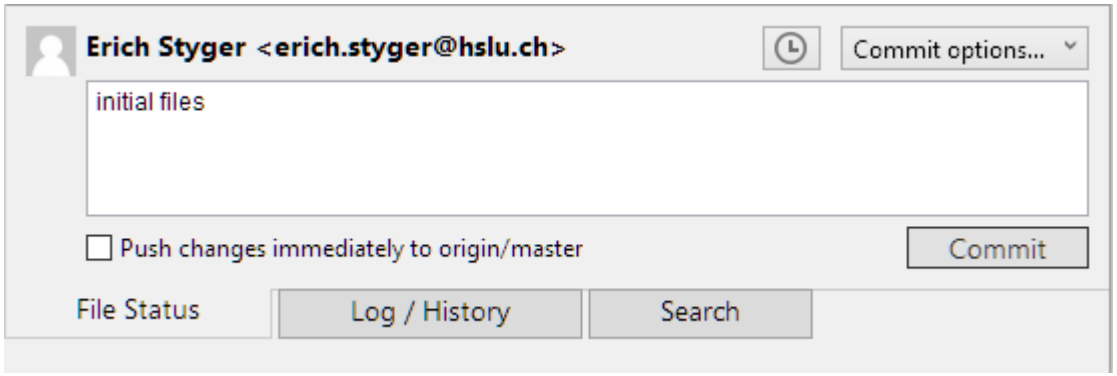
- Select 'Working Copy'
- Stage Files: 'put them on the stage for commit'
- Check your files!!!!!!
- Commit with Commit message

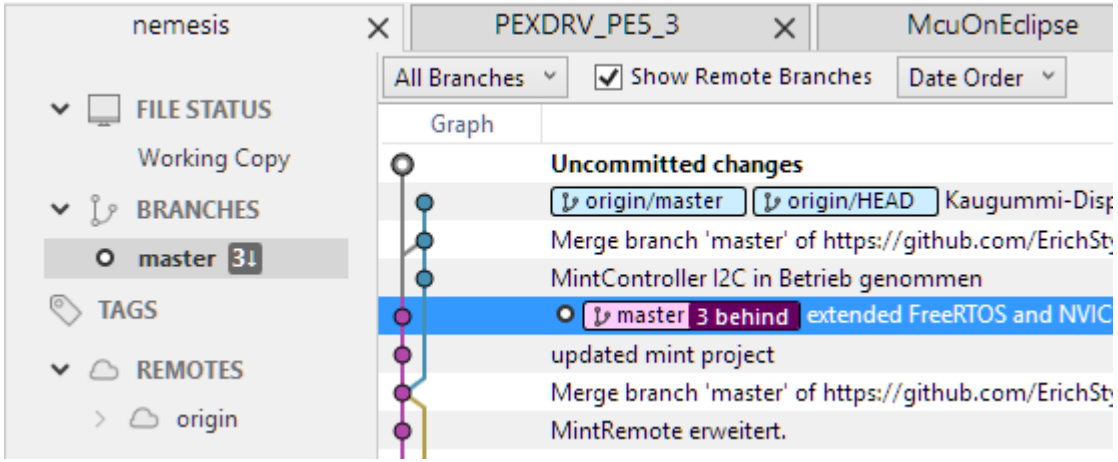




Commit and Push

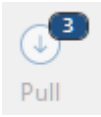
- Commit with useful message
- Then push local commits to remote master branch

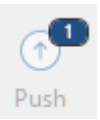




Other Actions

- **Fetch**
  - Get remote status information
- **Pull**
  - Get remote updates
- **Discard / Revert**
  - Undo a local change
- **Delete**
  - Deletes a file from the index/disk
- **Stash**
  - Save local changes away for later usage
  - Restore state later
- **Tag**
  - Mark with a label





Lucerne University of
Applied Sciences and Arts
HOCHSCHULE
LUZERN
Technik & Architektur

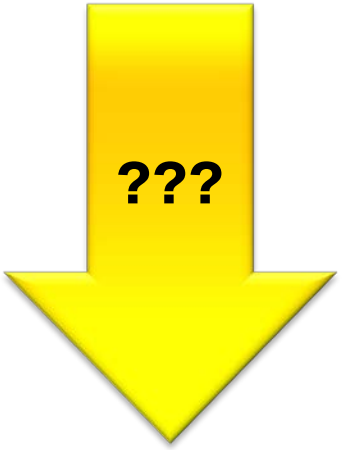
How to Git: GitHub Desktop

"RTF, well, I'll give you Step by Step instructions…"

Prof. Erich Styger
erich.styger@hslu.ch
+41 41 349 33 01

Learning Goals

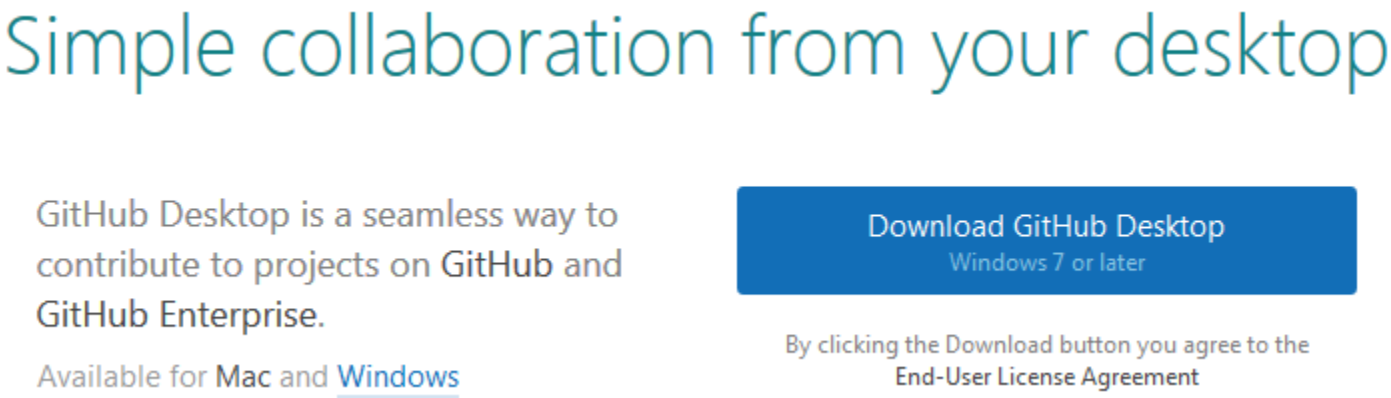
- Quick step and introduction using GitHub Desktop
- Git Client, optimized for GitHub
- Learn basic actions
    - Add
    - Clone
    - Compare
    - Commit
    - Push/Sync
- Advanced
    - Delete



GitHub Desktop

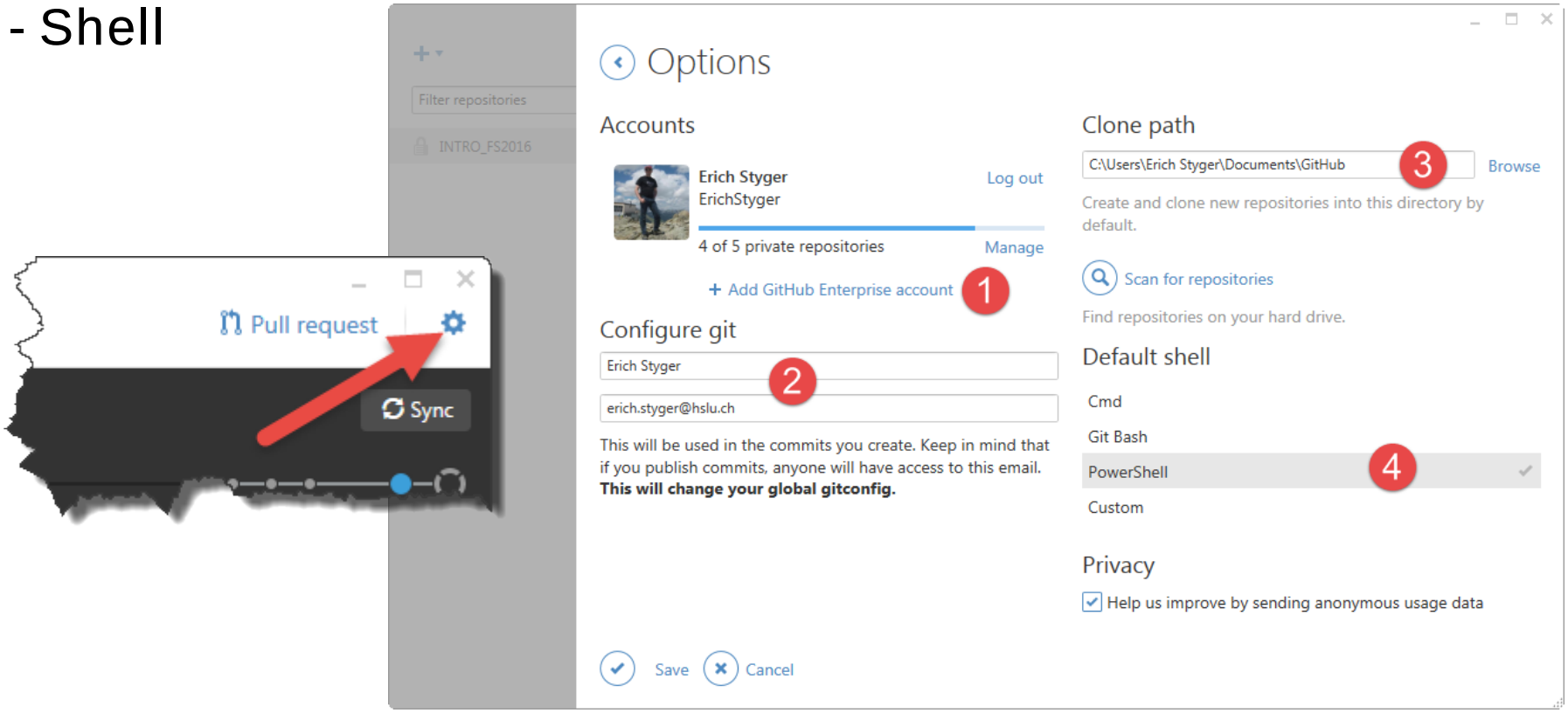
- Simple, easy GUI Tool
  - Clone
  - Commit, Push/Sync
- Good tool for using with GitHub

- Download https://desktop.github.com/ (Mac, Windows)



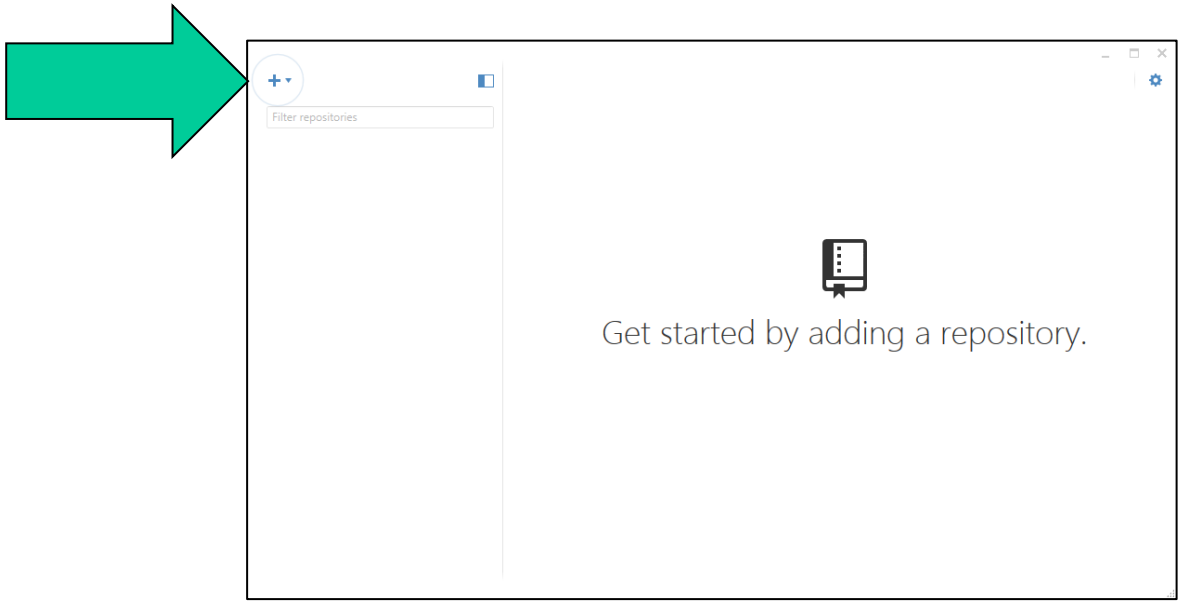
Setup

- Add account
- Name/email
- Default clone path
- Shell



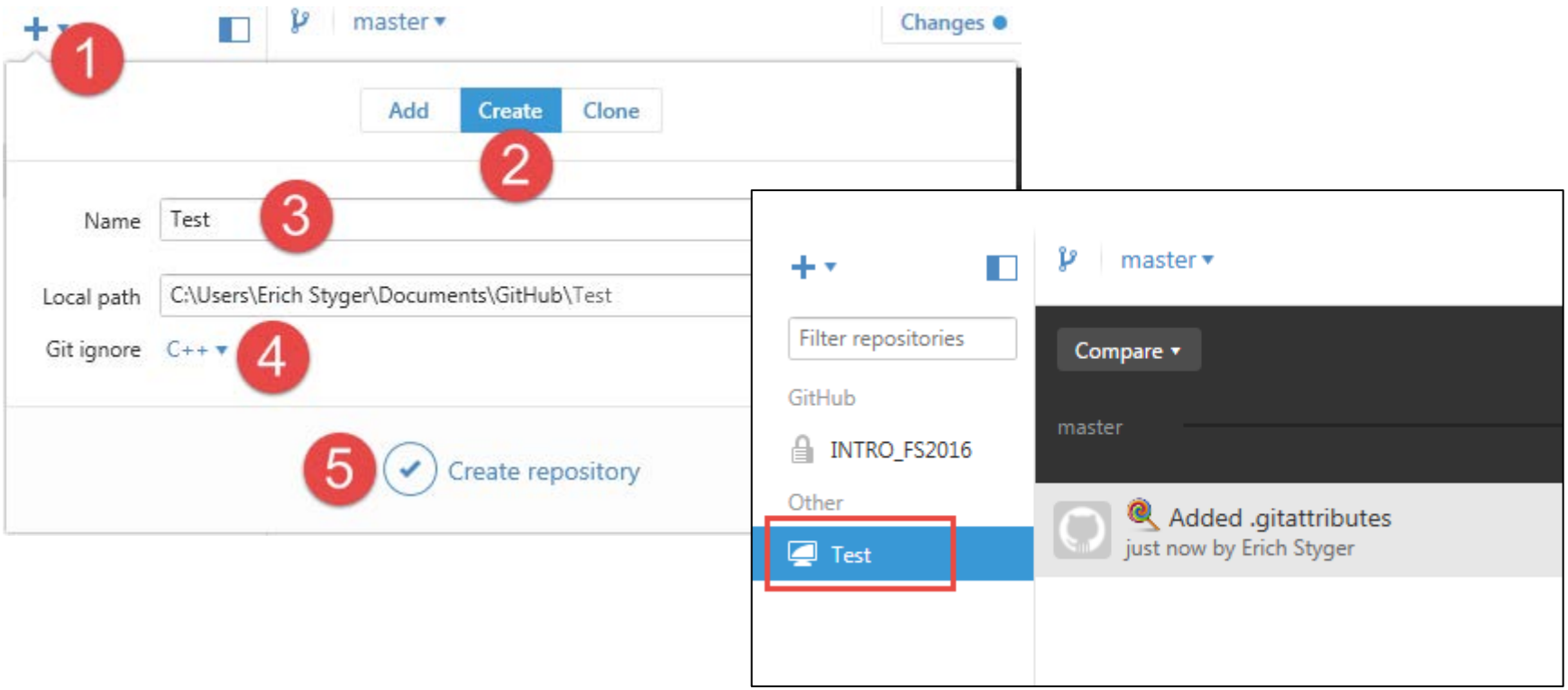
Cloning

- Launch Application
- Click on '+'
  - **Add**: add an local repo (or drag&drop)
  - **Create**: new repo, can push on server later
  - **Clone**: copy repository to local machine



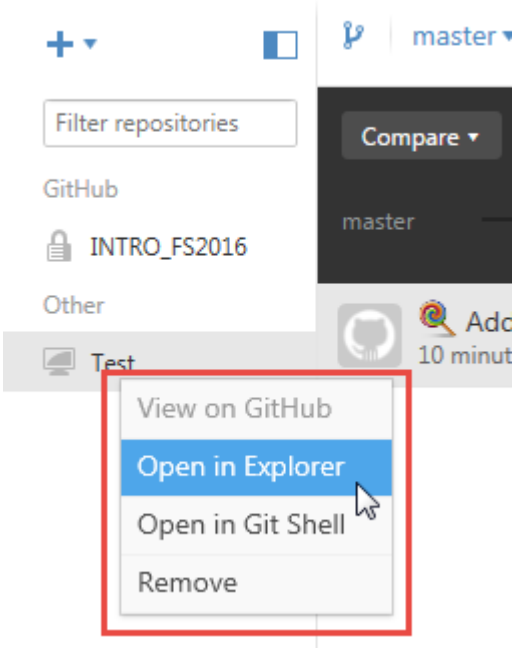
Create New Local Repo

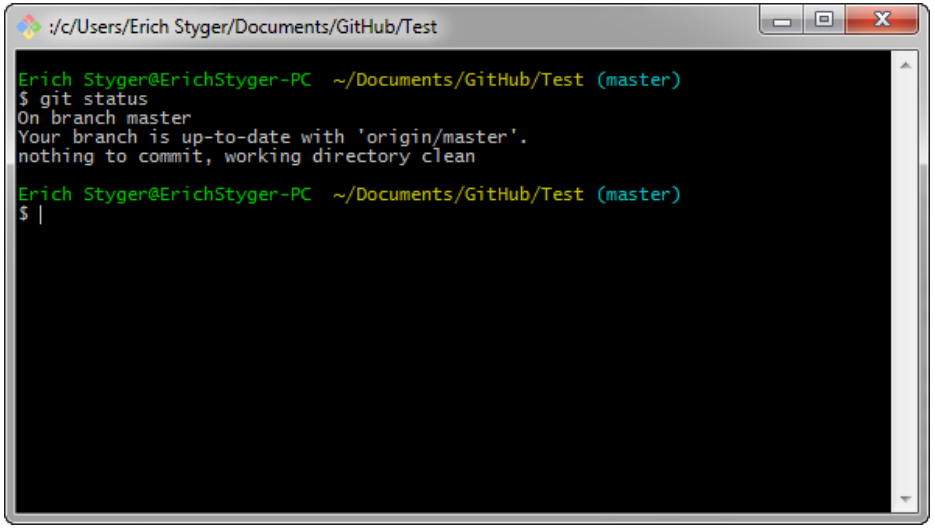
- Creates local new repo
- Can keep it locally, no need to publish it



Repository Actions

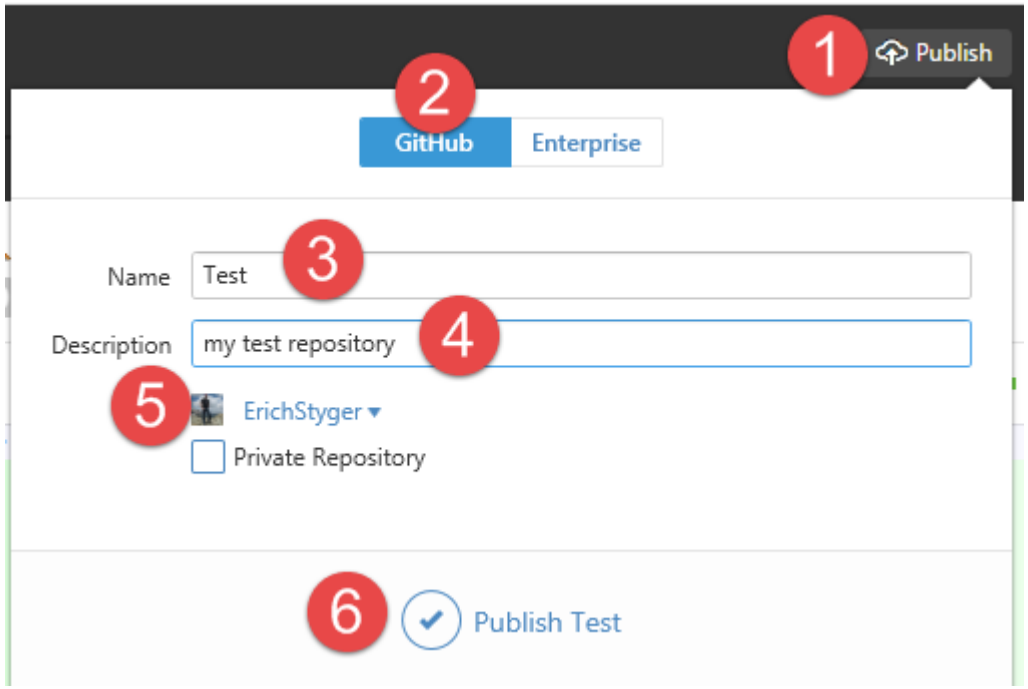
- View on GitHub (if published)
- Open in Explorer
- Open Shell (git help)
- Remove it from repository list (not on disk!)
- Re-add repo tip: drag & drop

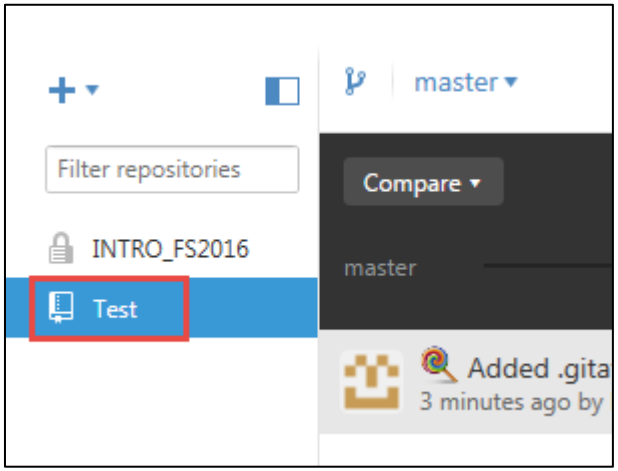




Publish

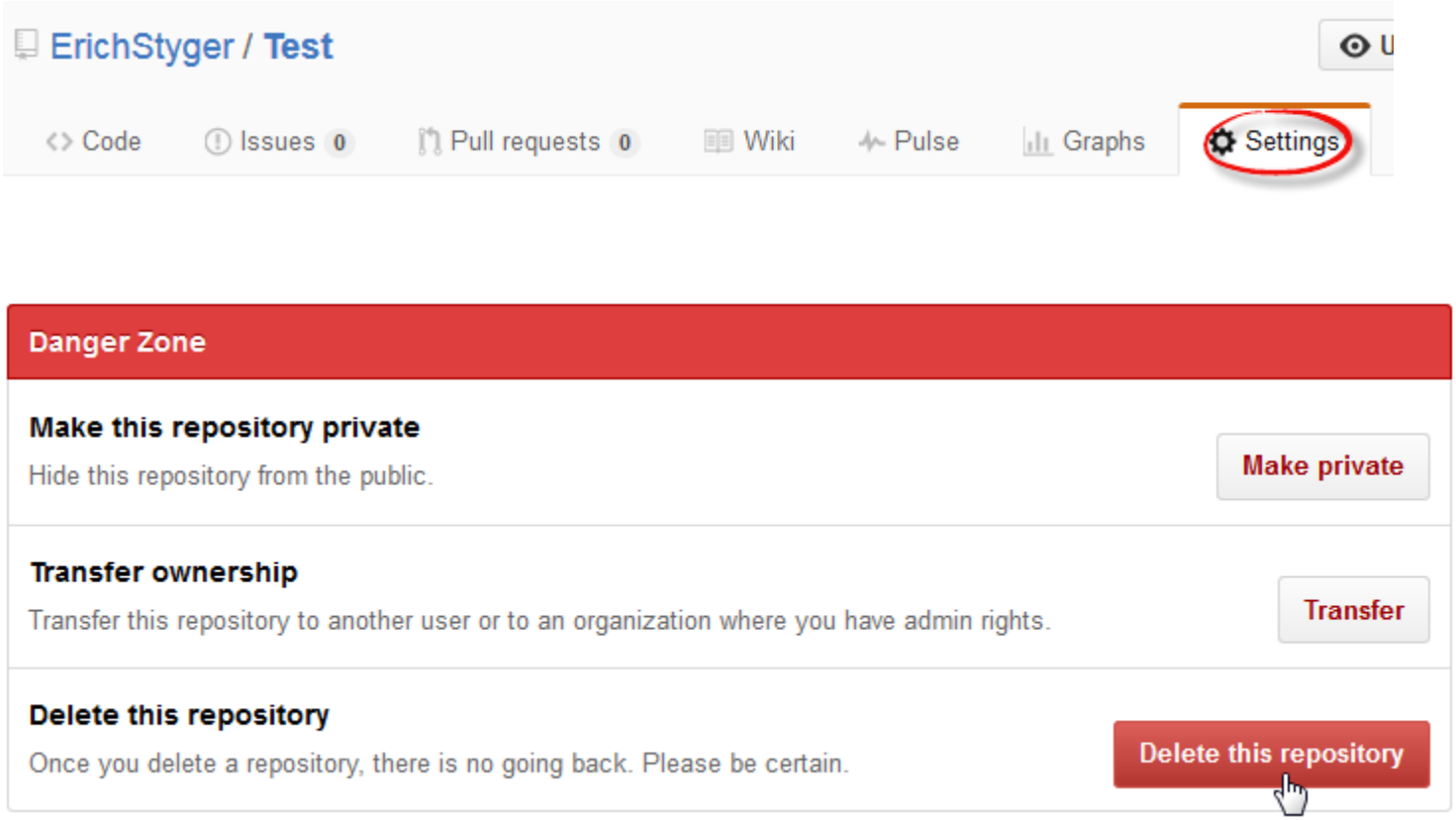
- Publish local repository to server/GitHub Account
- Delete on server if needed





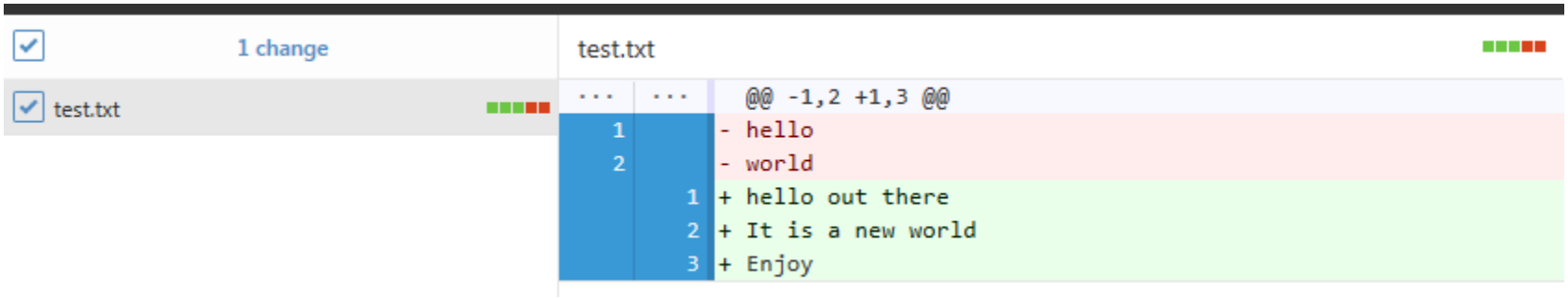
Deleting Repository (Server)

- Danger Zone!



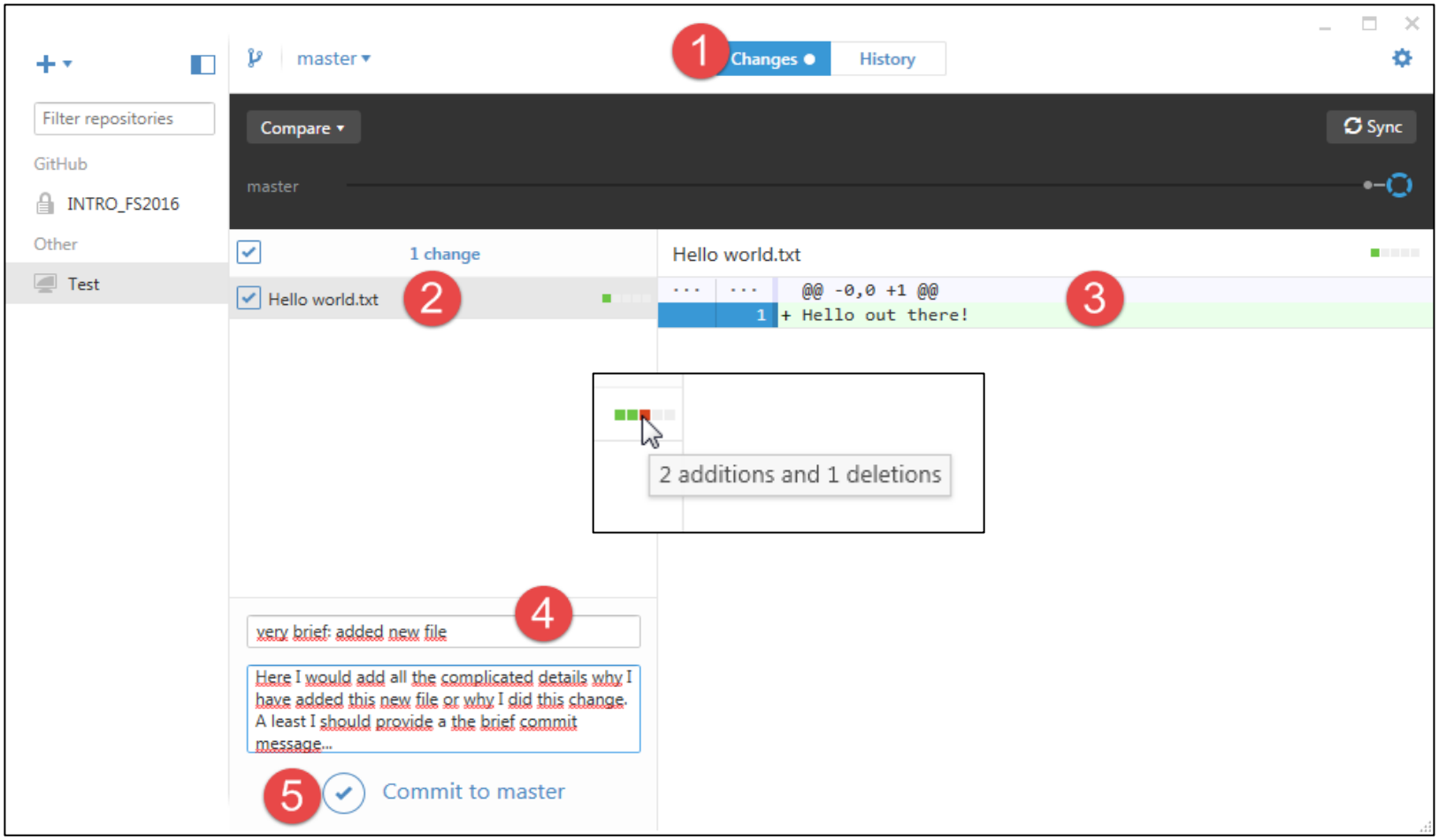
Compare

- Always inspect changes/files before commit!!!!!
- (-) Line deletions
- (+) Line additions



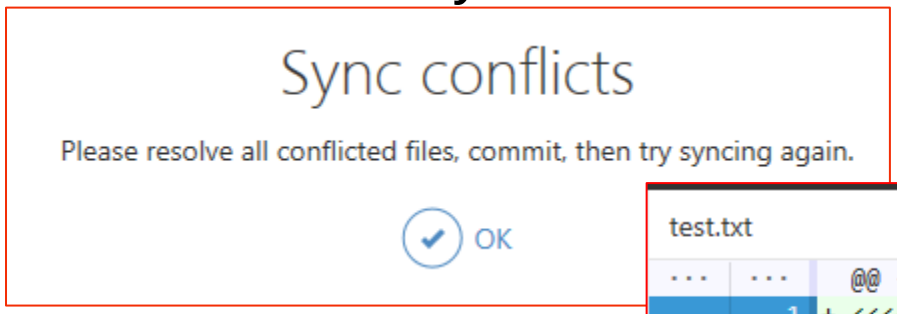
Changes: Commit

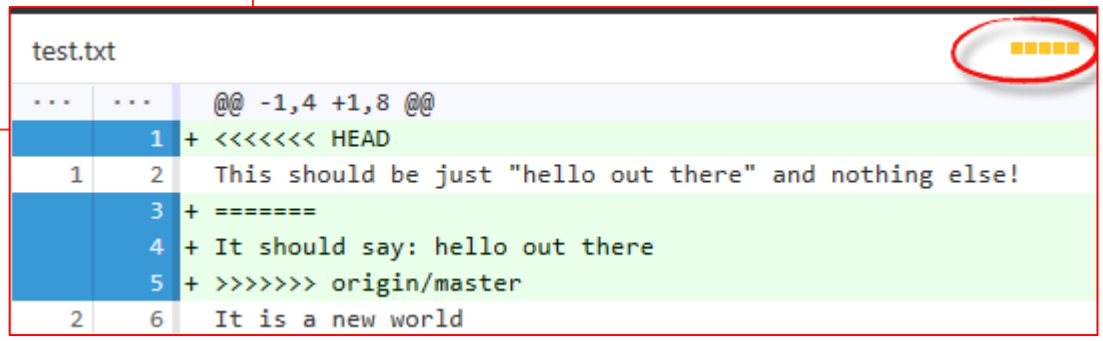
- Provide a good (!!!) commit message

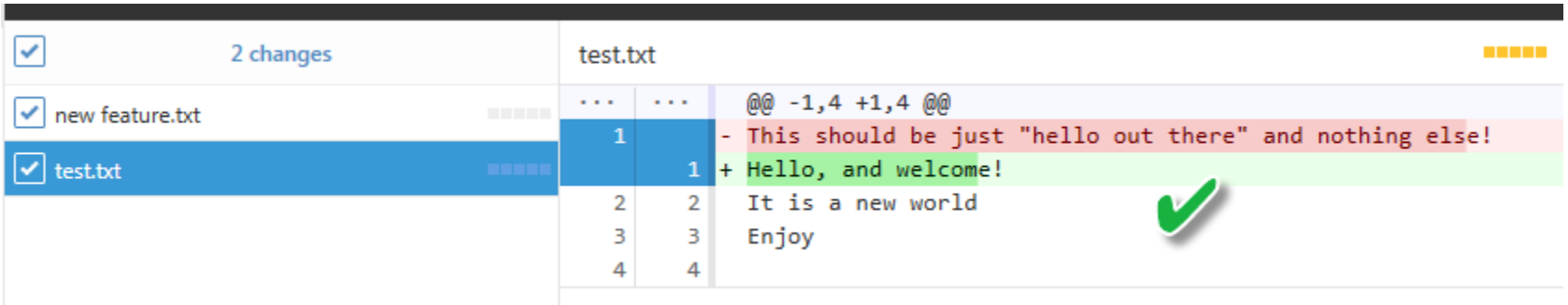


Conflicts

- Multiple changes on a line cannot be resolved ➔ Conflict!
- Resolve the conflict in the sources!!!!
- Commit and sync

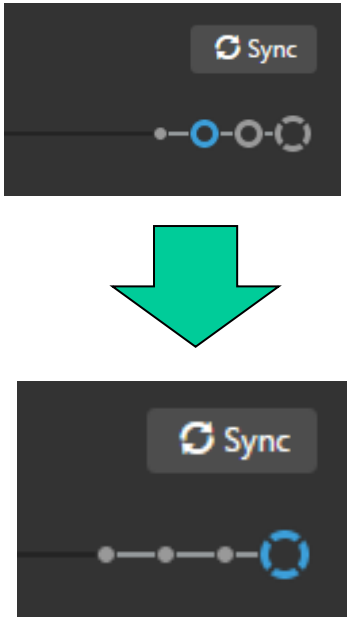






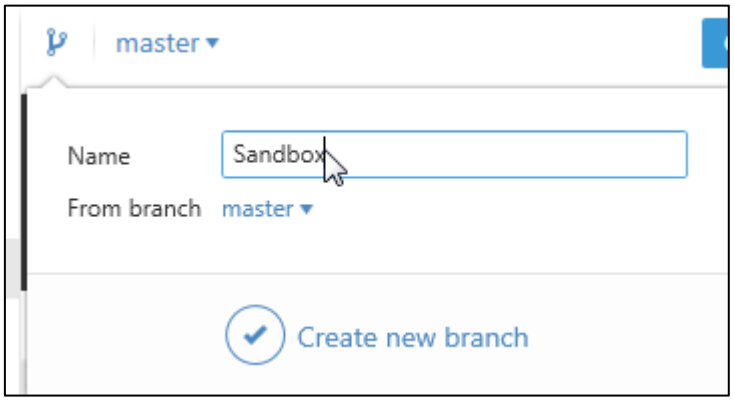
History Line

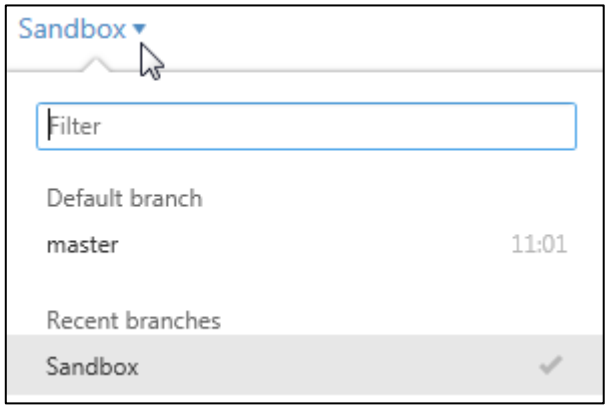
- Use history line to inspect changes
- Blue: current view/focus
- Dot: pushed history point on server
- Ring: committed, not on server yet
- Sync
  - Pull changes from server
  - Push commits to server

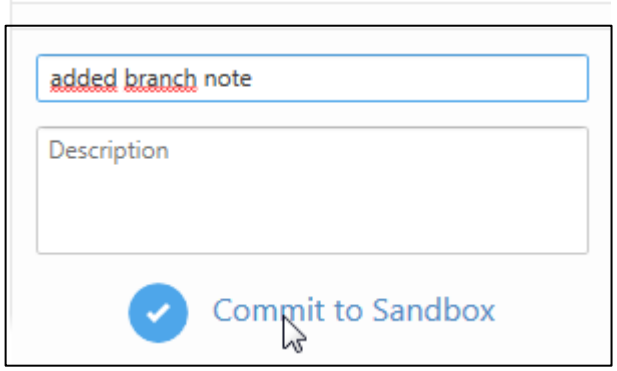


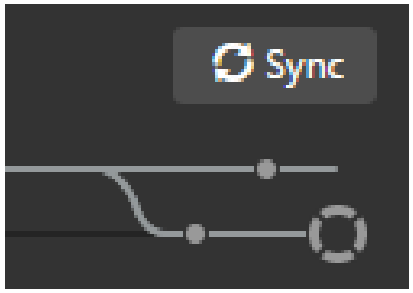
Branching

- Changes on branch do not affect master/mainline
- Create Branch, Switch between branches
- Careful if you have files open in applications!
- Commit to specific branch



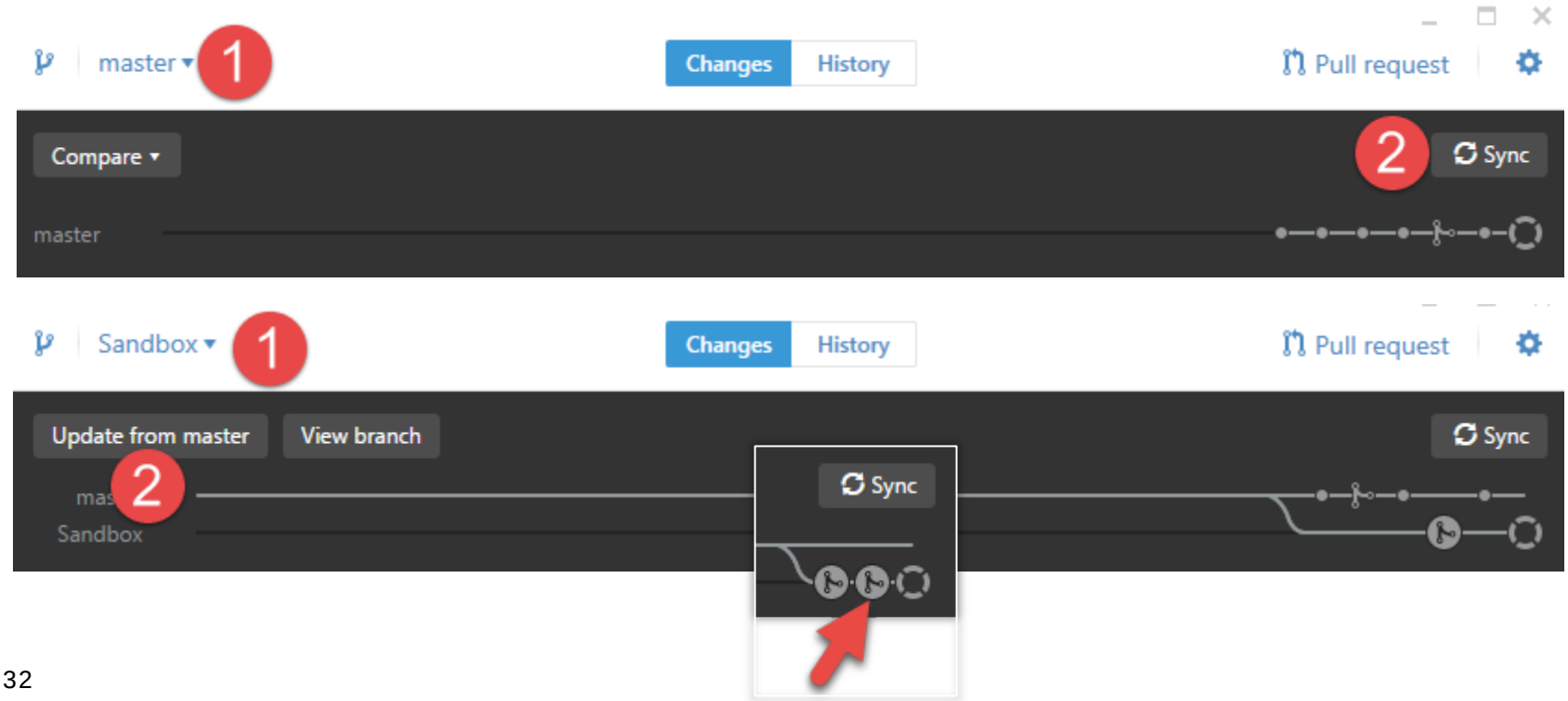






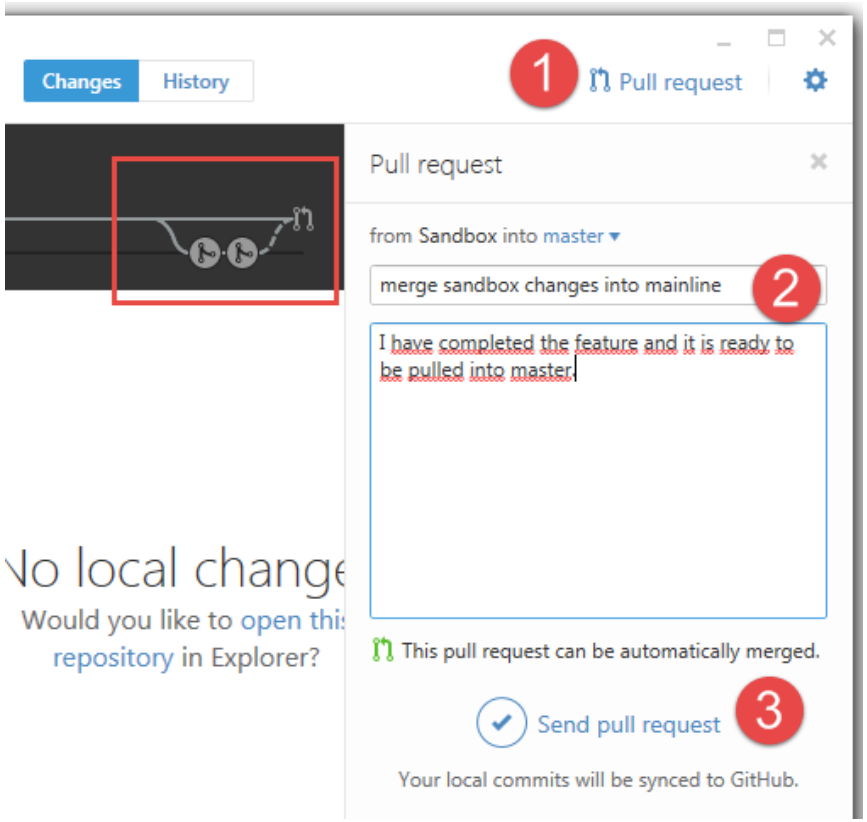
Synching Branch (master → branch)

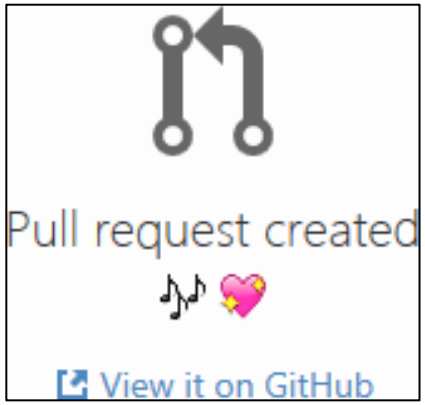
- Make sure your upstream/master is up-to-date!
  - Select master, do a sync
- Go back to branch
- Press ‘Update from master’
- →merges changes from master into branch



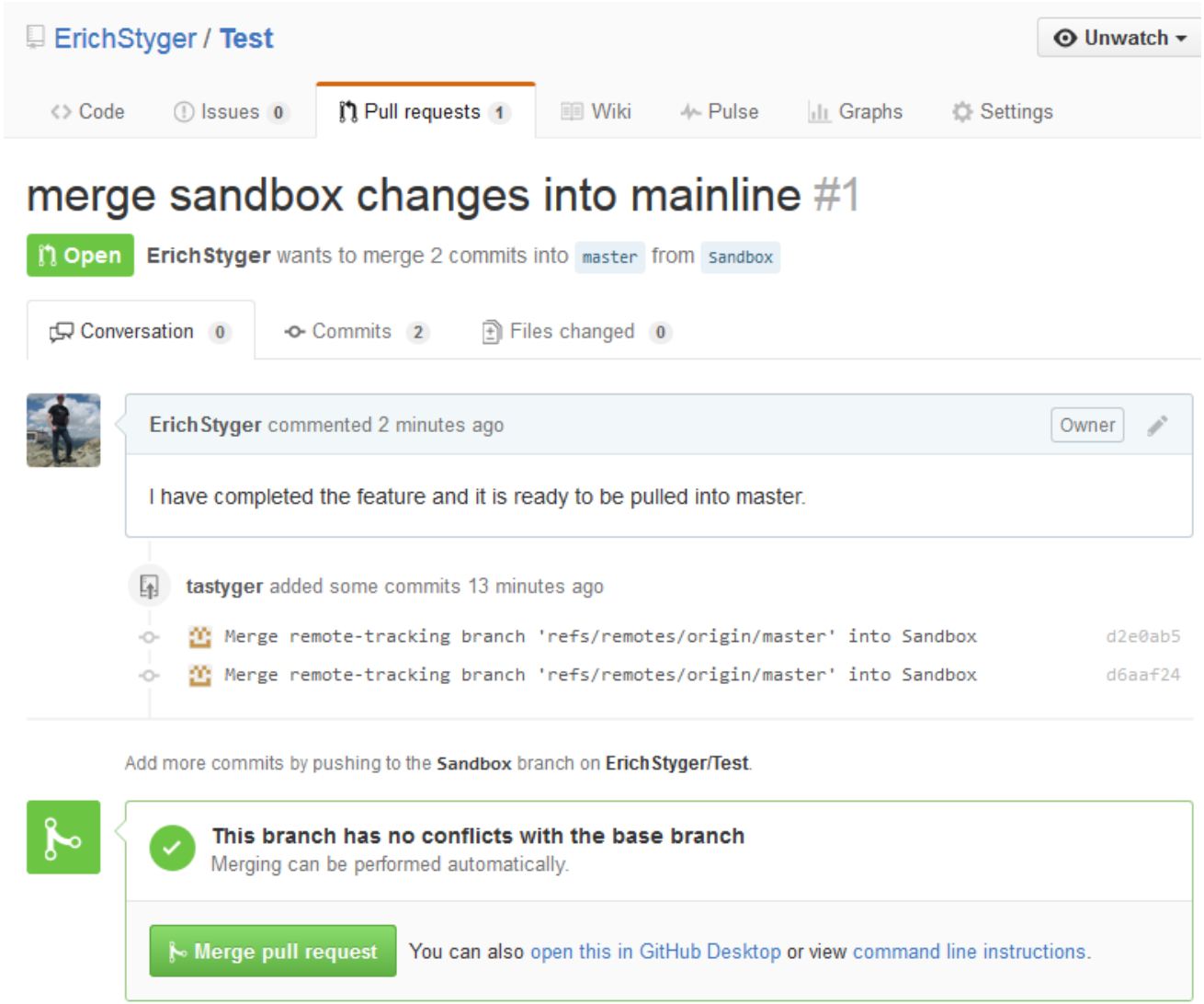
Merging Branch into Mainline / Master

- 'Pull Request': starts 'discussion' to merge things into mainline
- Create pull request from branch
- View pull request on GitHub



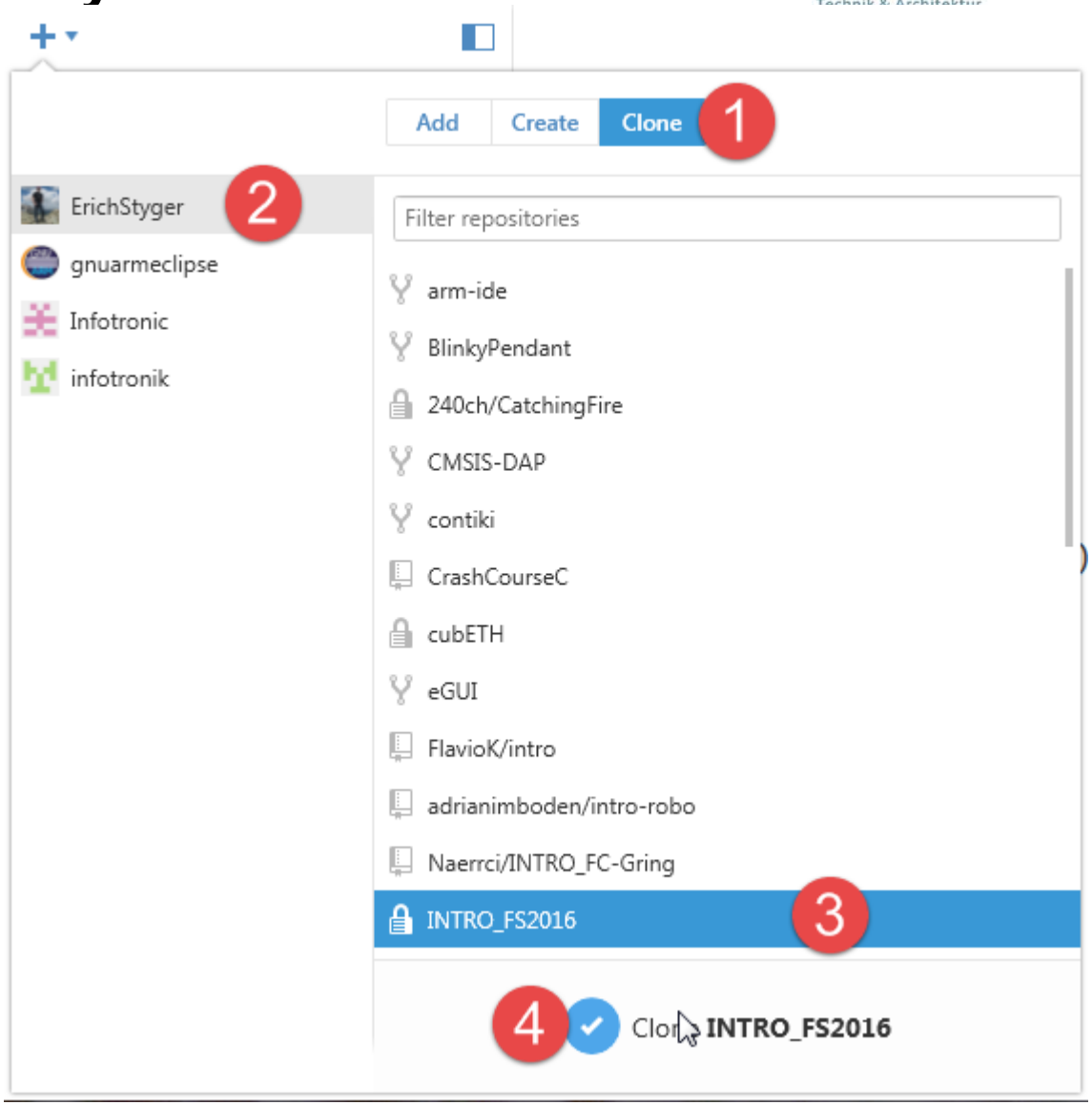


Pull Request



Clone INTRO Repository

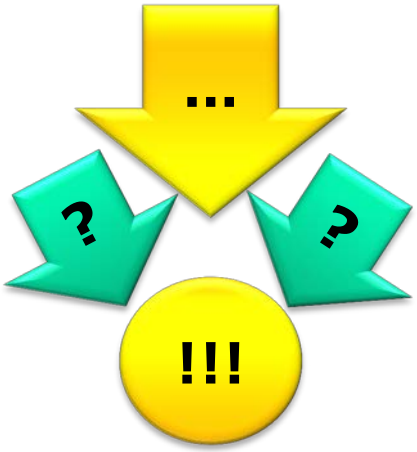
- Clone
- Select user account
- Select Repository
- Clone it
- (must have access to repo!)
- Specify folder where to clone



Summary

- GitHub Desktop
  - simple Git tool
  - optimized for GitHub usage
- Create, Add, Clone
- Commit and Push/Sync
- Conflicts
- Compare and History line
- Branch
  - Merge from master
  - Pull request to master

- Lab
  - practice, practice, practice!
  - Clone INTRO Repo



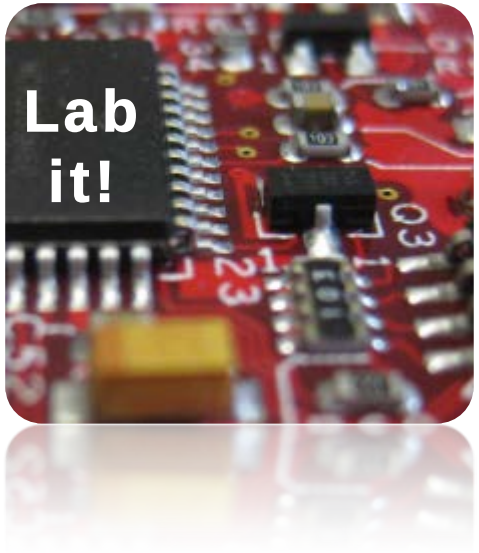
Check: Can you do this?

- Do you need a server or not?
- Cloning repository into multiple/different folders?
- File committed, but should have been ignored?
- What if I have file deleted on disk?
  - Delete it in repository?
  - Undo deletion?
- File changed, but want to revert change?

- How to verify you have everything you need in repository?

- Find out what is the difference/change?
- How to resolve conflict?
- How to 'simulate' a conflict?

Lab: Git Clients



- Install Client
  - SourceTree or GitHub Desktop or ???
- Install eGit (Eclipse Client)
- Clone INTRO repository

- Create your own git repository and clone it
  - Or use INTRO repository

- Move your wsp projects to git location
- Commit/Push to repository
- Get familiar with resolving conflicts/push/pull