THINK POP WORKSHOP
VERSION CONTROL WITH GIT

Tom Theile - Software Developer
Departement of Digital and
Computational Demography

MPIDR – 6/23/2023

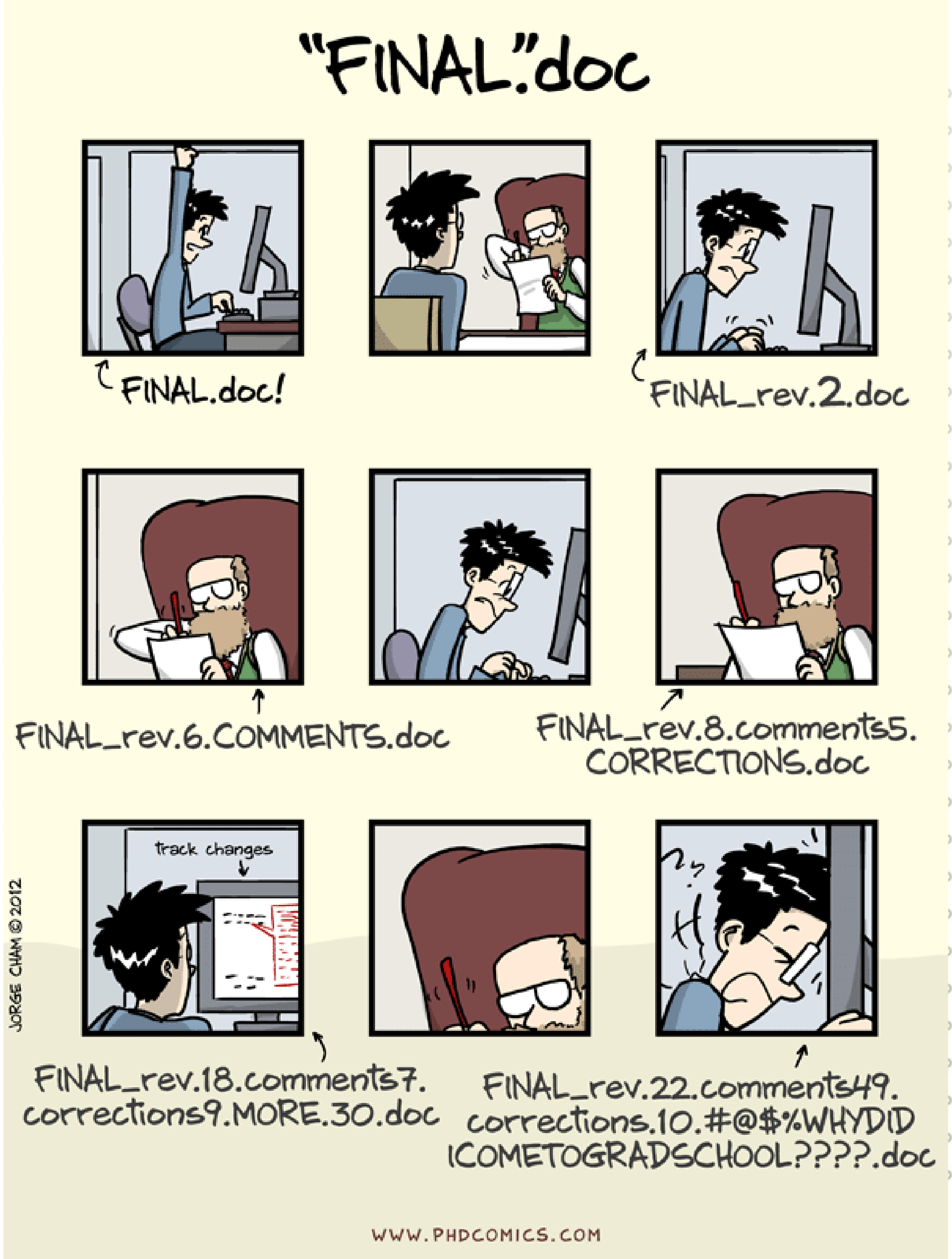
AGENDA

- Why you should use Git
- How does Git work?
- How to use Git
  - *On the command line*
  - *Together with GitHub*
  - *From Rstudio or JupyterLab*
- Best practices

WHY VERSION CONTROL?

- Unlimited undo + redo
  - *Nothing will get lost!*
  - *Code can be cleaner because it is save to delete code*
- Reproducability
  - *You can go „back in time“ to retrieve the exact code that was used in an experiment/analysis/simulation*
- MUCH better collaboration
  - *Everyting that can be handled automatically will be handled automatically*
  - *Many conflicts can be resolved automatically*

WHY VERSION CONTROL?


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
track changes
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDIDICOMETOGRADSCHOOL????.doc
JORGE CHAM © 2012
WWW.PHDCOMICS.COM

WHY GIT?

(AND NOT MERCURIAL, CVS, BITKEEPER, FOSSIL, …)

- It is open and free software
- Everyone else uses Git and GitHub
- *Network effects make it easier to collaborate*

HISTORY OF VERSION CONTROL AND GIT

- First software „like“ VC was released in 1962
  - *According to wikipedia, by IBM*
  - *First „real“ VC in 1972*
  - *Only for single files*

- CVS was released 1990
  - *Can now control directories!*
  - *Needs a central server*
  - *Popular and hated for many years*

- Subversion was released 2000
  - *mostly compatible to CVS (successor)*
  - *Open Software*

HISTORY OF VERSION CONTROL AND GIT

- Bitkeeper (released 2000)
  - *The linux kernel developers used it from 2002 to 2005*
  - *First (?) „distributed“ version control*

- Git (2005)

- Mercurial, Fossil, etc.
  - *Also great software, but with much less adoption*

- In-house solutions at Google (piper), etc.
  - *Google used perforce, but needed better performance, developed their own*
  - *Everything is checked into one giant „Monorepo“*

WHAT DOES „DISTRIBUTED“ VERSION CONTROL MEAN?

- No central server necessary
- *→ Faster and more reliable*

- The whole repository (code + full history) is mirrored on every developers computer [w]

- Everything except synchronisation works offline and fast

LINUX, BITKEEPER AND GIT

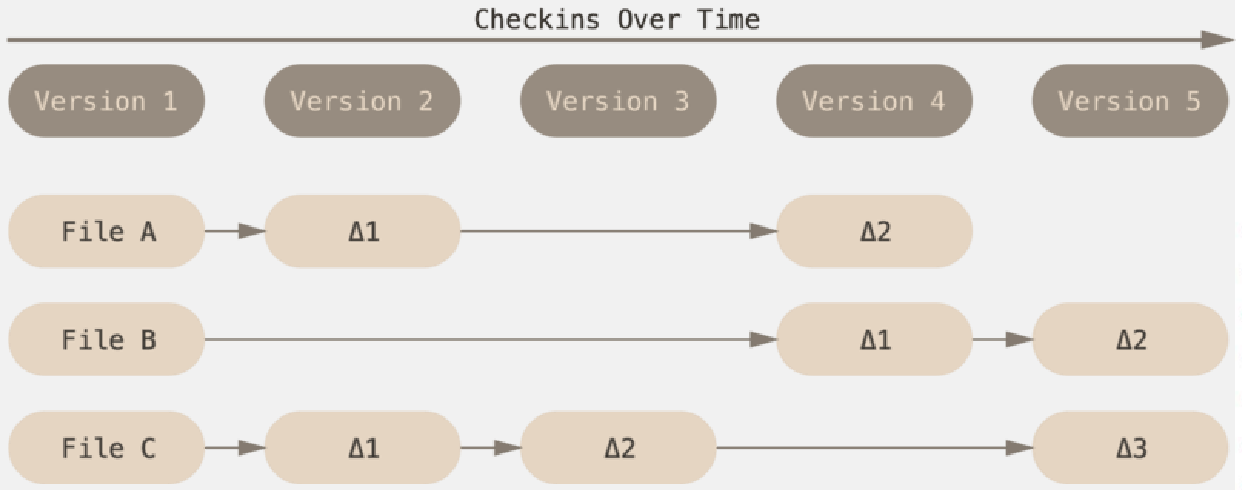
- Linux developers used Bitkeeper for free (from 2002 on)
- In April 2005, Bitkeeper anounced to revoke the free licence from July on, because some Linux developer implemented improvements for free users
- No other VC fullfilled the requirements of the Linux dev community
- Linus Torvalds himself started coding an alternative on April 3
- Linux switched to Git in June 2005
- https://en.wikipedia.org/wiki/BitKeeper
- https://en.wikipedia.org/wiki/Git#History

HOW DOES GIT WORK?

- This part is based on the official guide
- *https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F*

HOW DOES GIT WORK?

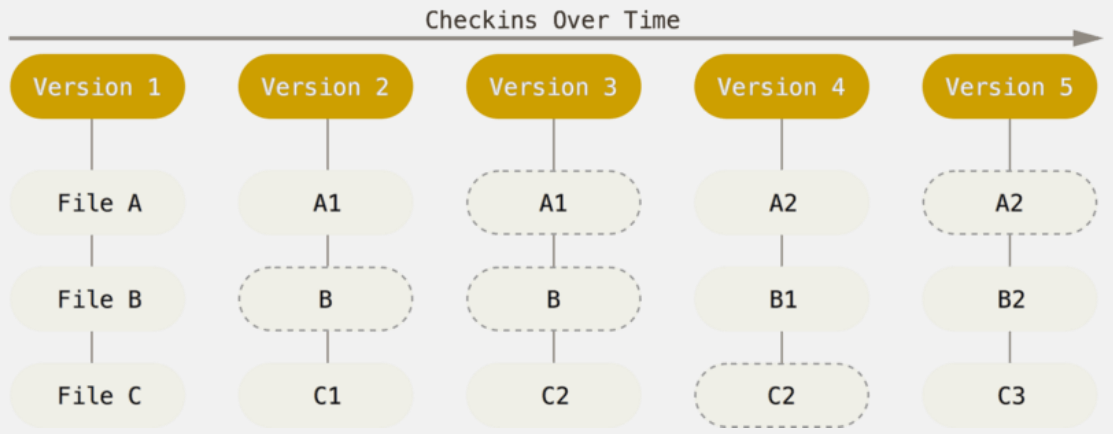
Other VCS: Differences



HOW DOES GIT WORK?

Git: Snapshots, Not Differences

“With Git, every time you commit, or save the state of your project, Git basically takes a picture of what all your files look like at that moment and stores a reference to that snapshot. To be efficient, if files have not changed, Git doesn’t store the file again, just a link to the previous identical file it has already stored.”



HOW DOES GIT WORK?

- Everything is stored in the .git subdirectory
- Most operations are local and do not require a connection to a server

HOW DOES GIT WORK?

- Git checks the integrity of files with checksums

SHA-1 hash: `24b9da6552252987aa493b52f8696cd6d3b00373`

- You will see these hash values all over the place in Git because it uses them so much. In fact, Git stores everything in its database not by file name but by the hash value of its contents.

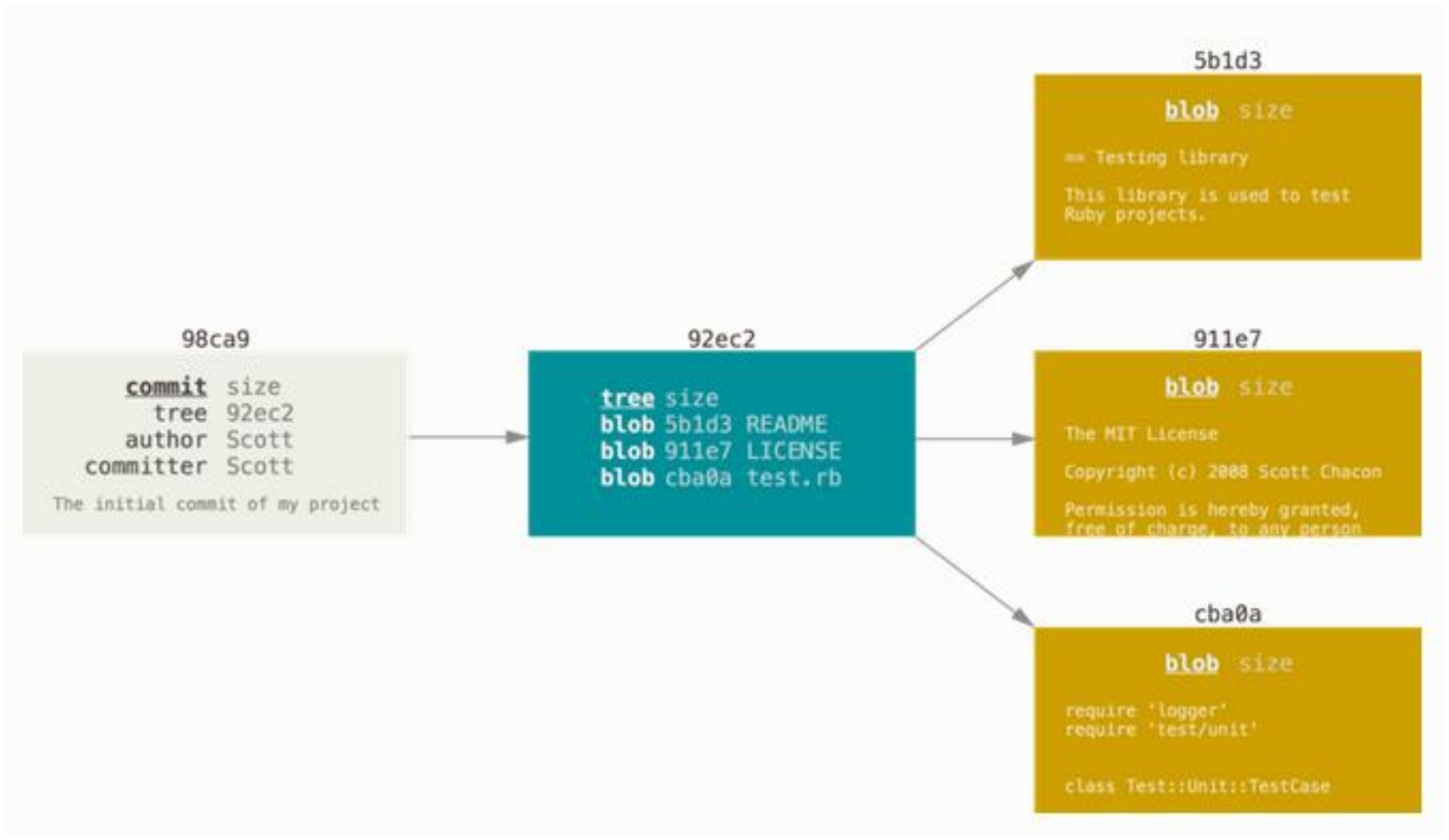
- You can‘t change a file without Git noticing it

HOW DOES GIT WORK?

Git Generally Only Adds Data

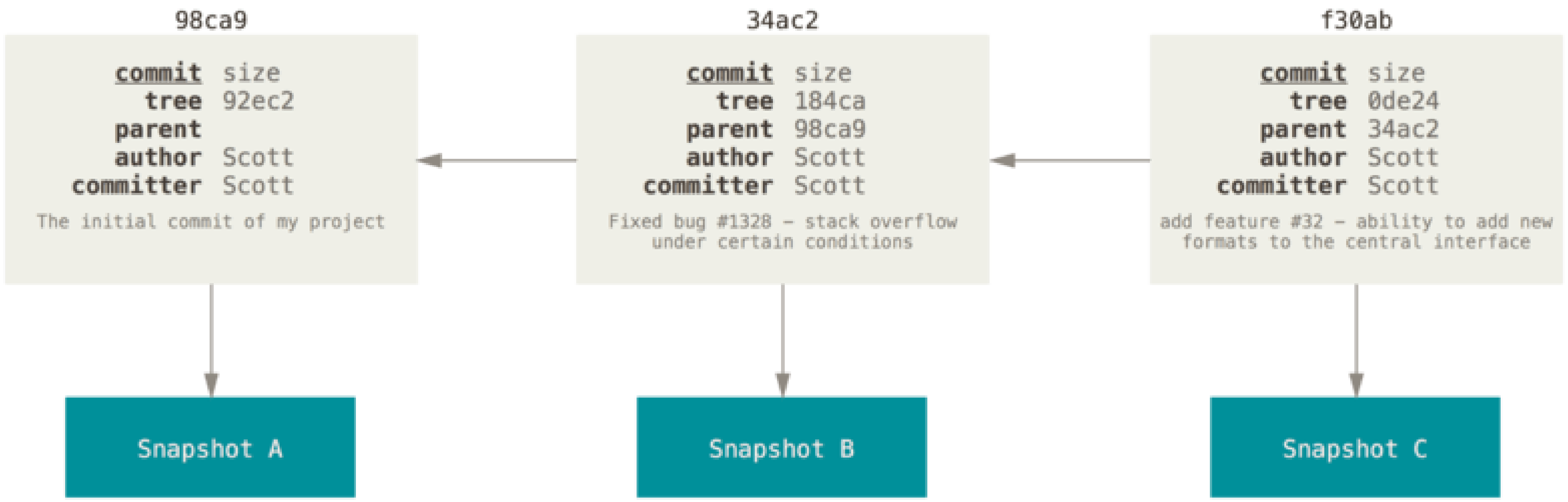
HOW DOES GIT WORK?
HOW DOES GIT STORE THE REPOSITORY?

This is one commit



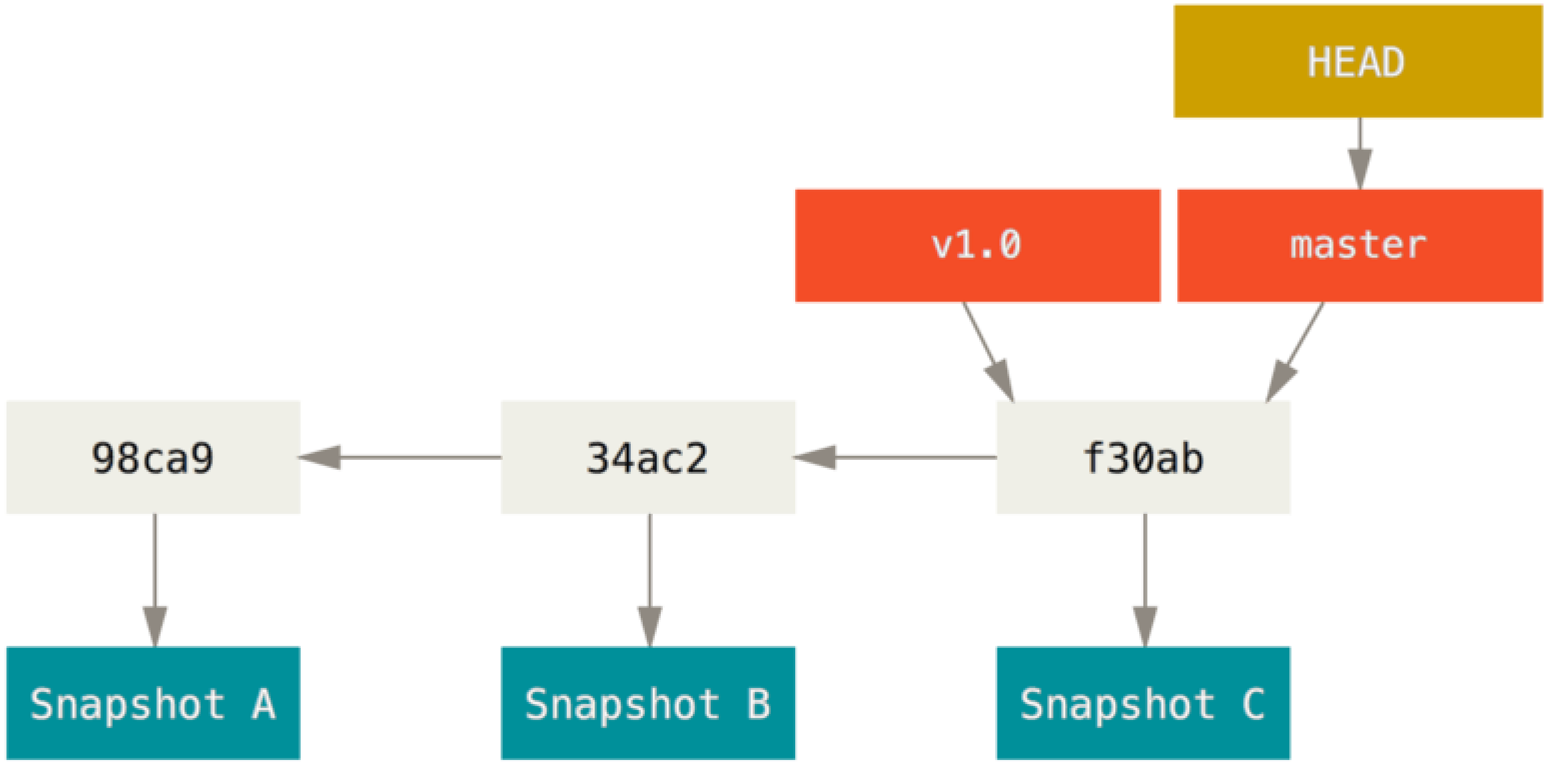
HOW DOES GIT WORK?
HOW DOES GIT STORE THE REPOSITORY?

This is a chain (or tree) of commits:



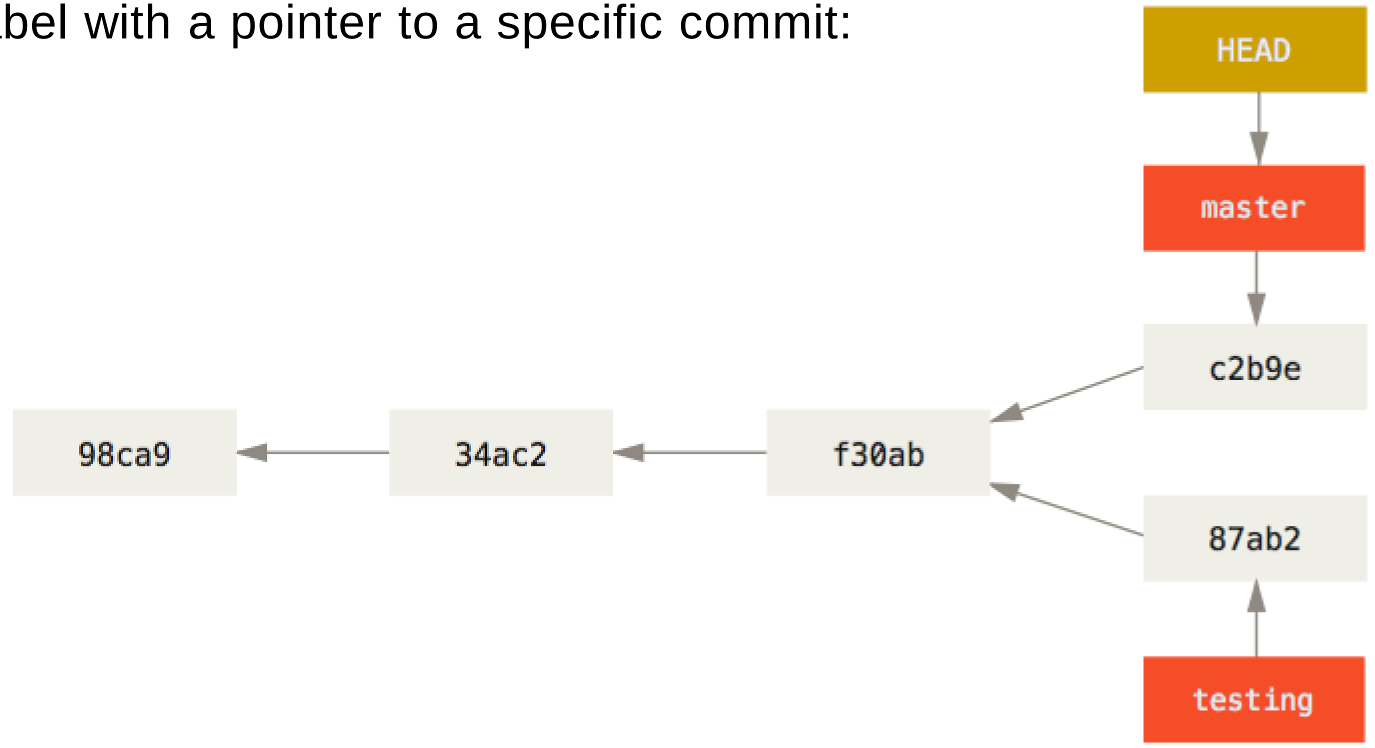
HOW DOES GIT WORK?
HOW DOES GIT STORE THE REPOSITORY?

A branch is just a label with a pointer to a specific commit:



HOW DOES GIT WORK?
HOW DOES GIT STORE THE REPOSITORY?

A branch is just a label with a pointer to a specific commit:



INSTALLATION

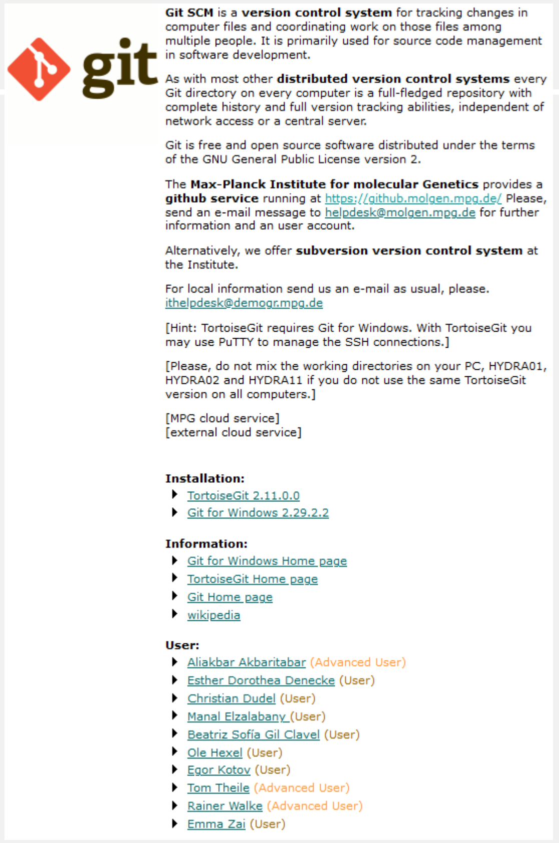
- MPIDR - use the software-shop:
- https://intranet.demogr.mpg.de/cgi-bin/it/software/entry.plx?id=-1&

- Windows:
- https://git-scm.com/download/win

- Linux:
- *You already have git. Or you know how to install it!*

- Mac
- https://git-scm.com/download/mac
- *This will (probably) automatically install git:*

```
$ git --version
```



Git SCM is a **version control system** for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development.

As with most other **distributed version control systems** every Git directory on every computer is a full-fledged repository with complete history and full version tracking abilities, independent of network access or a central server.

Git is free and open source software distributed under the terms of the GNU General Public License version 2.

The **Max-Planck Institute for molecular Genetics** provides a **github service** running at https://github.molgen.mpg.de/ Please, send an e-mail message to helpdesk@molgen.mpg.de for further information and an user account.

Alternatively, we offer **subversion version control system** at the Institute.

For local information send us an e-mail as usual, please.
ithelpdesk@demogr.mpg.de

[Hint: TortoiseGit requires Git for Windows. With TortoiseGit you may use PuTTY to manage the SSH connections.]

[Please, do not mix the working directories on your PC, HYDRA01, HYDRA02 and HYDRA11 if you do not use the same TortoiseGit version on all computers.]

[MPG cloud service]
[external cloud service]

Installation:
- TortoiseGit 2.11.0.0
- Git for Windows 2.29.2.2

Information:
- Git for Windows Home page
- TortoiseGit Home page
- Git Home page
- wikipedia

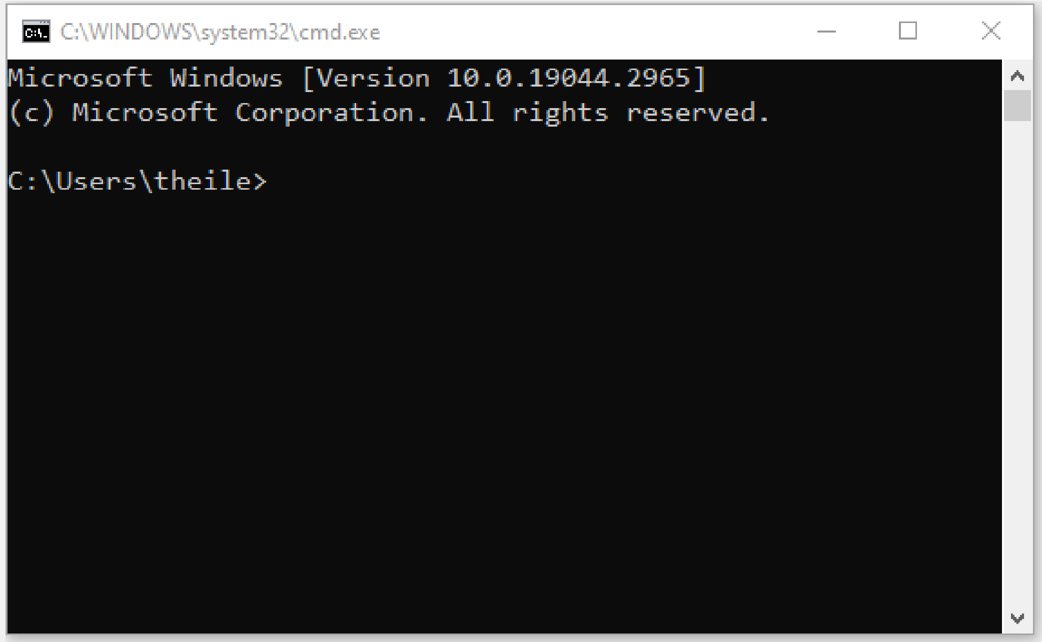
User:
- Aliakbar Akbaritabar (Advanced User)
- Esther Dorothea Denecke (User)
- Christian Dudel (User)
- Manal Elzalabany (User)
- Beatriz Sofía Gil Clavel (User)
- Ole Hexel (User)
- Egor Kotov (User)
- Tom Theile (Advanced User)
- Rainer Walke (Advanced User)
- Emma Zai (User)

GETTING STARTED

- Open a Terminal

- Windows:
- *Type „cmd“ into the Task Bar and click on „Command Prompt“*

- Mac:
- ***Click the Launchpad icon in the Dock, type Terminal in the search field, then click Terminal***

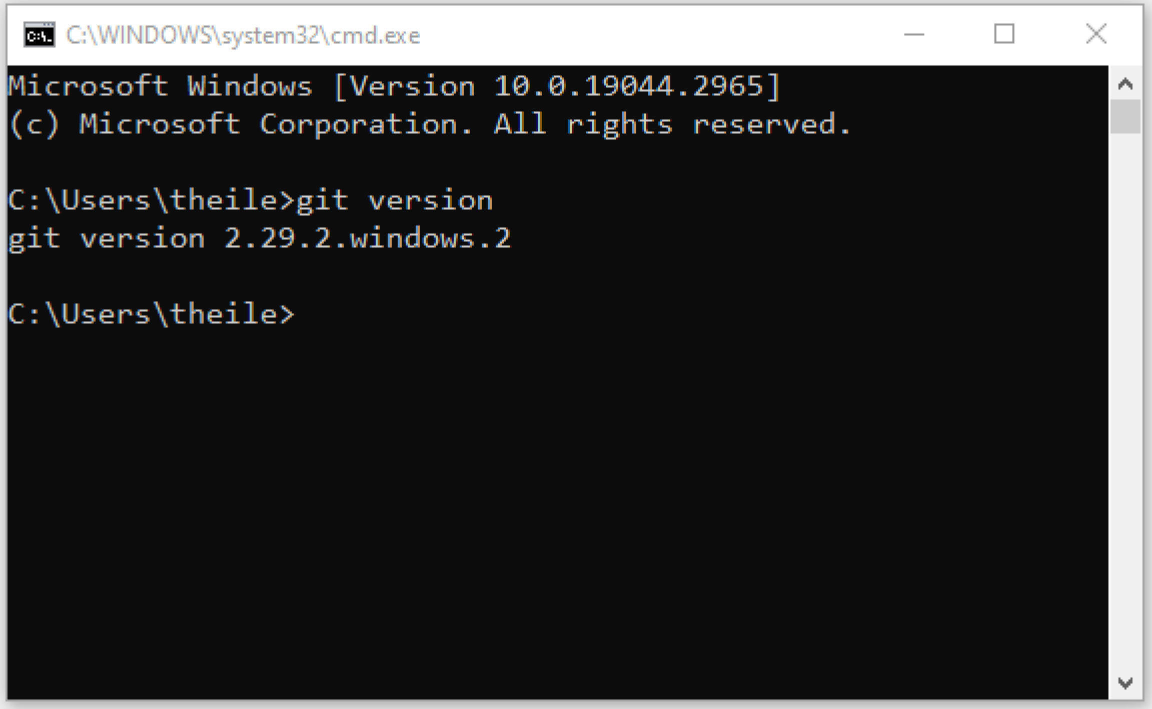


```
Microsoft Windows [Version 10.0.19044.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\theile>
```

GETTING STARTED

- Git version

- Does it work?



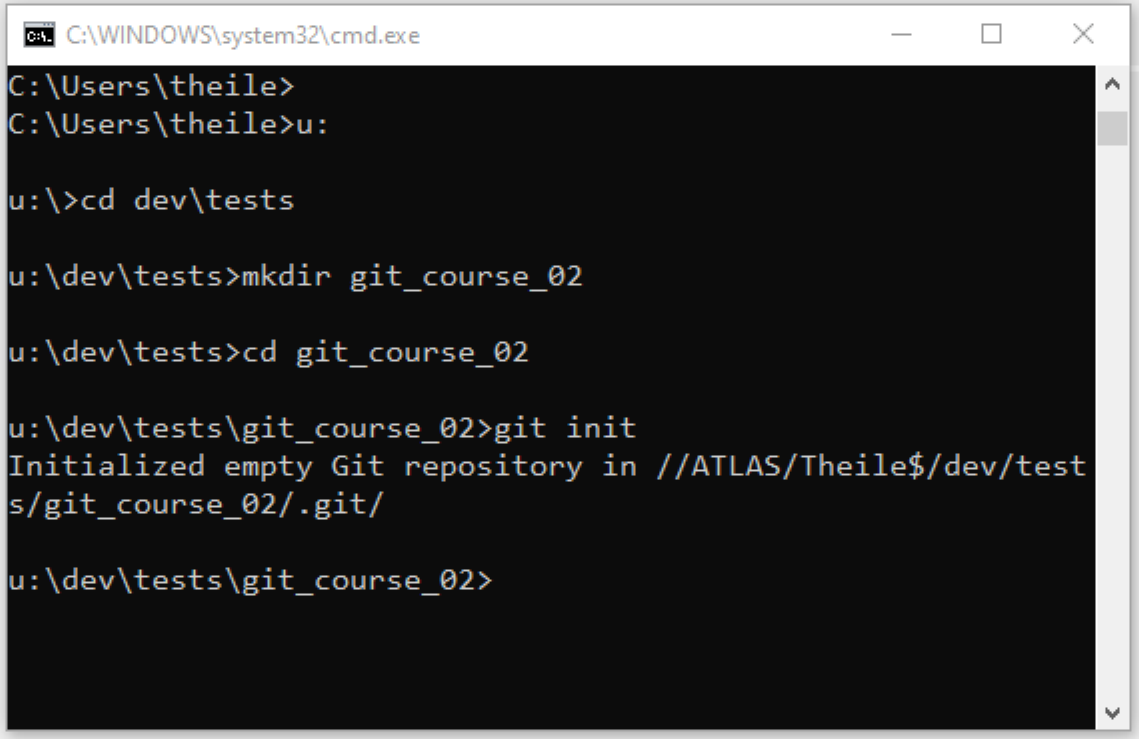
```
Microsoft Windows [Version 10.0.19044.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\theile>git version
git version 2.29.2.windows.2

C:\Users\theile>
```

GETTING STARTED

- Git config
- https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup
- `$ git config --global user.name “Tom Theile"`
- `$ git config --global user.email tom-github@theilemail.de`
- Use your github email address

INITIALISING A REPOSITORY

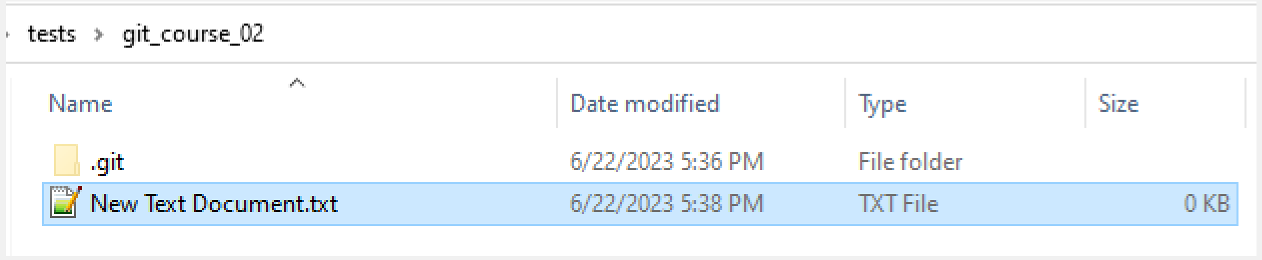
- A „repository“ is a folder with a project
- Init creates a hidden subfolder „.git“
- *Git stores everything in this subfolder. The whole history and configuration of this project is there*
- But you have to „commit“ files so that they are stored in Git



```
C:\Users\theile>
C:\Users\theile>u:

u:\>cd dev\tests

u:\dev\tests>mkdir git_course_02

u:\dev\tests>cd git_course_02

u:\dev\tests\git_course_02>git init
Initialized empty Git repository in //ATLAS/Theile$/dev/test
s/git_course_02/.git/

u:\dev\tests\git_course_02>
```

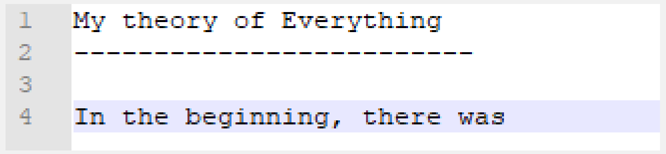
ADDING FILES

- Let‘s create a new text file



```
tests > git_course_02

Name                      Date modified        Type          Size
.git                      6/22/2023 5:36 PM    File folder
New Text Document.txt     6/22/2023 5:38 PM    TXT File      0 KB
```

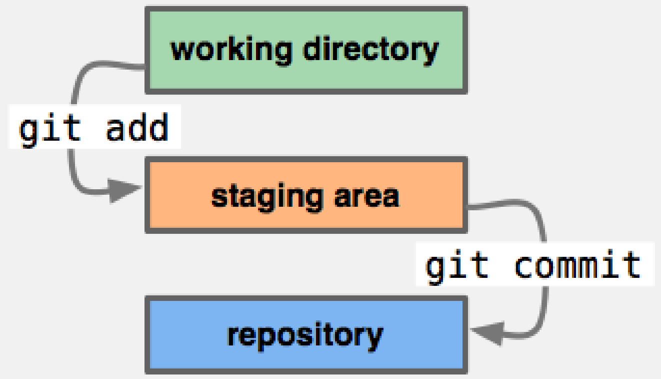
- And enter some text

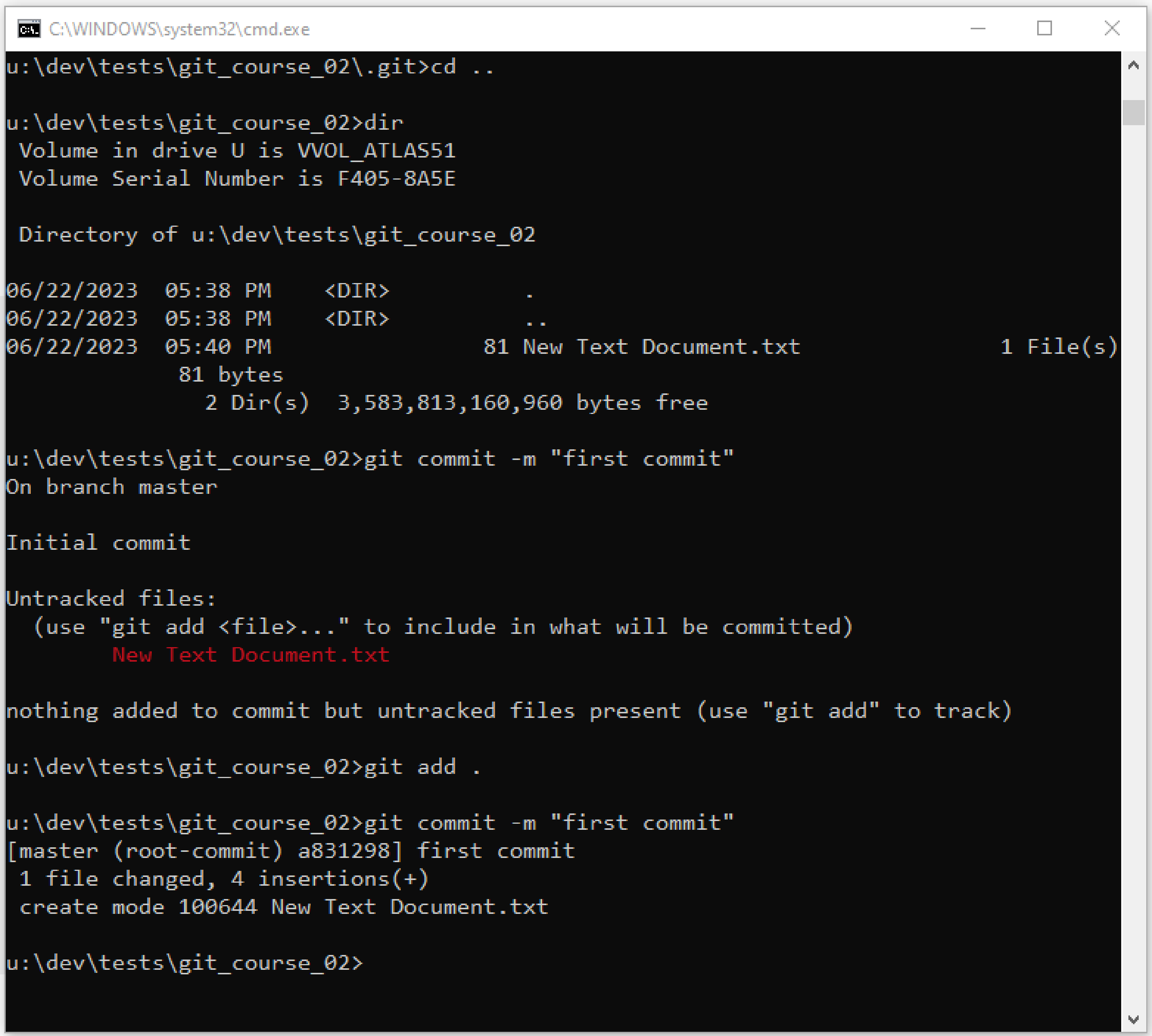


```
My theory of Everything
-------------------------

In the beginning, there was
```

ADDING AND COMMI

- On the command line, files have to be „added“ to the „stage“ first

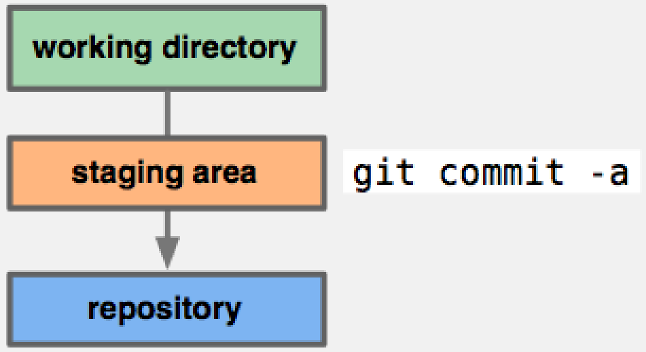


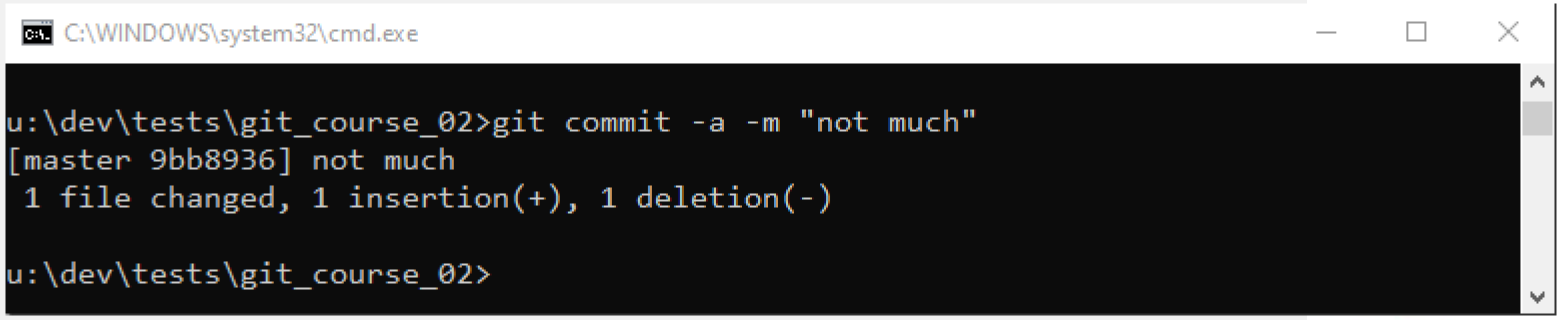


```
u:\dev\tests\git_course_02\.git>cd ..

u:\dev\tests\git_course_02>dir
 Volume in drive U is VVOL_ATLAS51
 Volume Serial Number is F405-8A5E

 Directory of u:\dev\tests\git_course_02

06/22/2023  05:38 PM    <DIR>          .
06/22/2023  05:38 PM    <DIR>          ..
06/22/2023  05:40 PM                81 New Text Document.txt               1 File(s)
             81 bytes
               2 Dir(s)  3,583,813,160,960 bytes free

u:\dev\tests\git_course_02>git commit -m "first commit"
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        New Text Document.txt

nothing added to commit but untracked files present (use "git add" to track)

u:\dev\tests\git_course_02>git add .

u:\dev\tests\git_course_02>git commit -m "first commit"
[master (root-commit) a831298] first commit
 1 file changed, 4 insertions(+)
 create mode 100644 New Text Document.txt

u:\dev\tests\git_course_02>
```

COMMITING DIRECTLY

- But you can skip this with -a





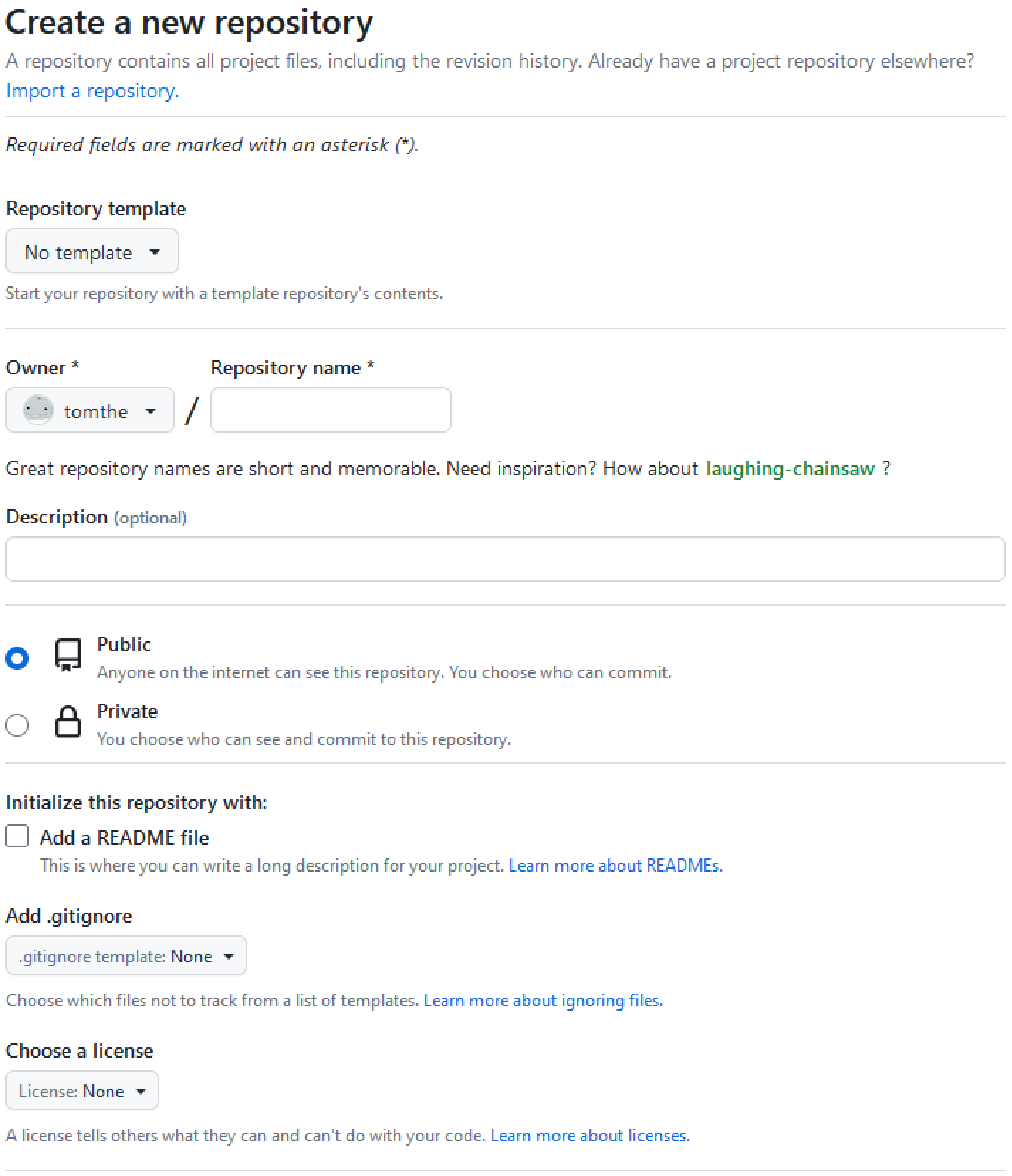
```
u:\dev\tests\git_course_02>git commit -a -m "not much"
[master 9bb8936] not much
 1 file changed, 1 insertion(+), 1 deletion(-)

u:\dev\tests\git_course_02>
```

CLONE A REPOSITORY FROM GITHUB

```
git clone https://github.com/tomthe/git_course_04.git
```

CREATE A REMOTE REPOSITORY ON GITHUB

- Everyone: create your own repository!
- *https://github.com/new*



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

Required fields are marked with an asterisk (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner * tomthe ▾ / **Repository name ***

Great repository names are short and memorable. Need inspiration? How about laughing-chainsaw ?

Description (optional)

[✓ **Public**]
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

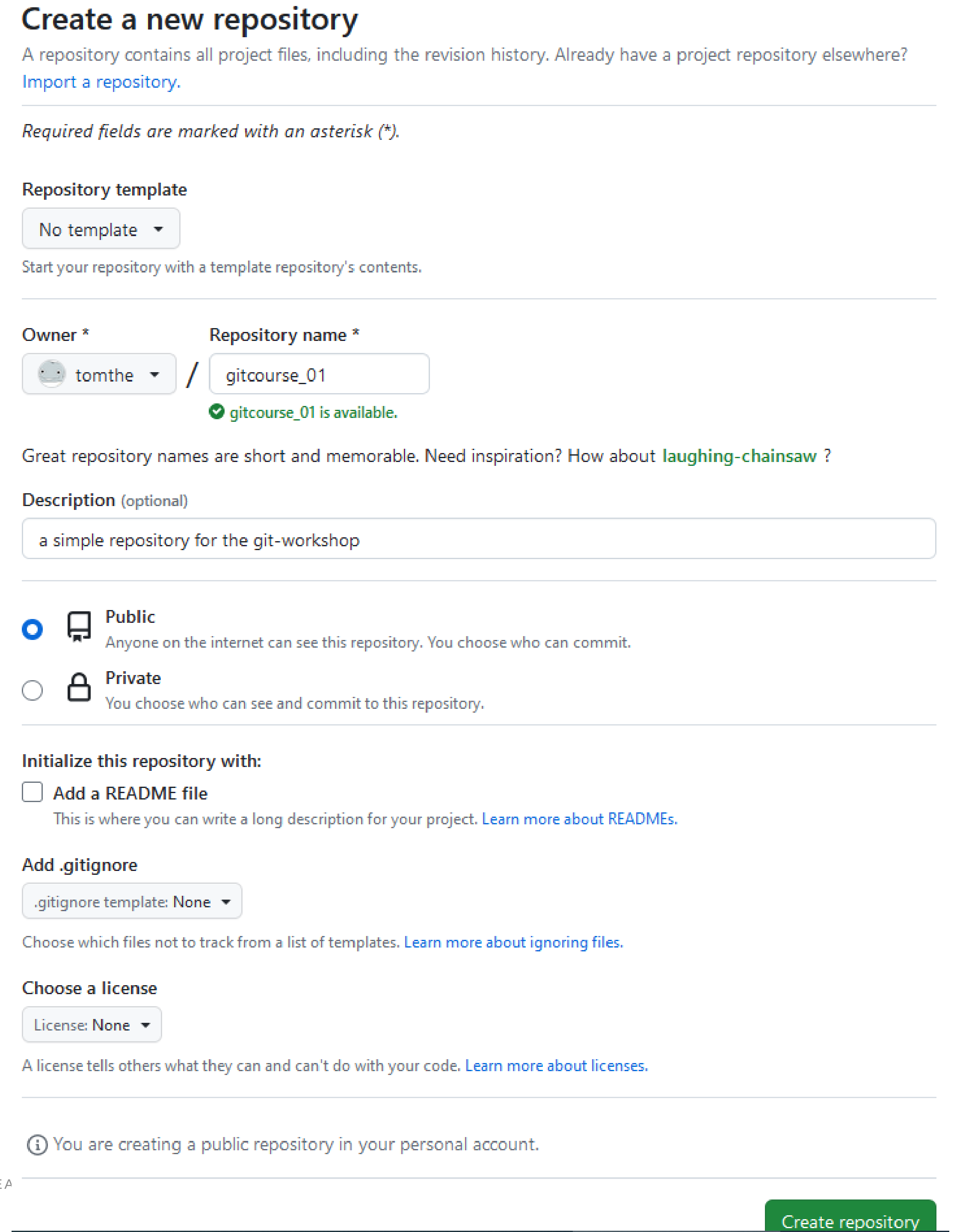
.gitignore template: None ▾

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. Learn more about licenses.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Repository template

No template

Start your repository with a template repository's contents.

Owner * tomthe / **Repository name *** gitcourse_01

gitcourse_01 is available.

Great repository names are short and memorable. Need inspiration? How about laughing-chainsaw ?

Description (optional)

a simple repository for the git-workshop

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private — You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

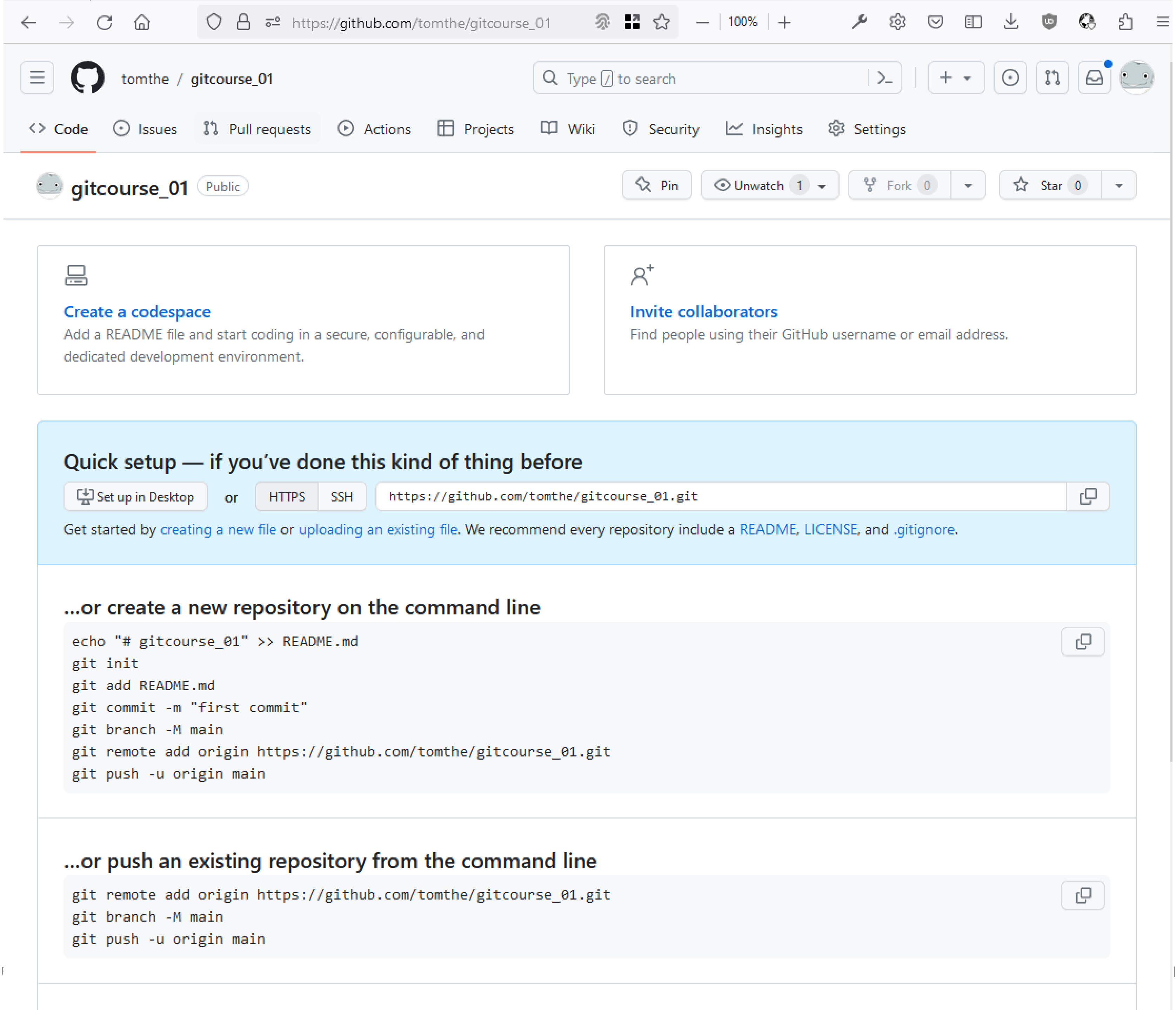
Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a public repository in your personal account.



Create a codespace
Add a README file and start coding in a secure, configurable, and dedicated development environment.

Invite collaborators
Find people using their GitHub username or email address.

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/tomthe/gitcourse_01.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# gitcourse_01" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/tomthe/gitcourse_01.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/tomthe/gitcourse_01.git
git branch -M main
git push -u origin main
```

CREATE A REMOTE REPOSITORY ON GITHUB AND ADD IT AS A REMOTE ON YOUR LOCAL MACHINE

- Everyone: create your own repository!
- git remote add origin https://github.com/tomthe/git_course_01.git

CREATE A REMOTE REPOSITORY ON GITHUB AND ADD IT AS A REMOTE ON YOUR LOCAL MACHINE THEN PUSH YOUR CHANGES

- Everyone: create your own repository!

- git remote add origin https://github.com/tomthe/git_course_01.git
- Push your local repository (and your changes) to GitHub:

```
git push -u origin main
```

- Open your repository on GitHub

PULL CHANGES

- Open your repository on GitHub
- Change the text file
- Pull the changes to your local machine:

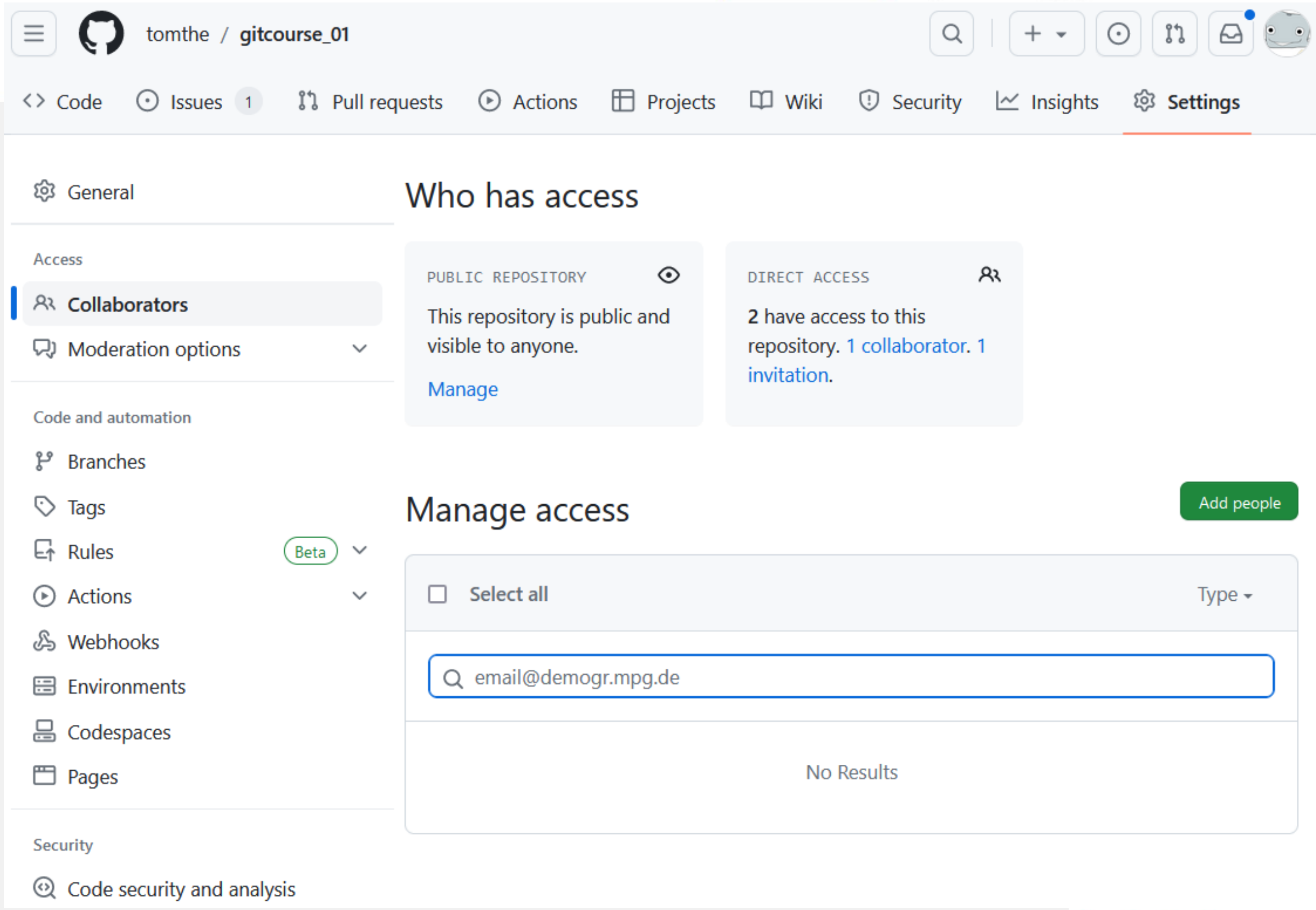
```
git pull
```

- Look into your local file

COLLABORATION!

- Exchange your repository with your neighbour(s)
- Play Rock, Paper, Scissors. The winner has to provide a repository link
- Add your neighbour as a collaborator
- Collaborator has to accept and clone the repo
- Then both can add a new text file with some text and commit, push
- Pull, merge, push for the one who was slower
- Look at the repository now!

- Use issues to talk about your files



Who has access

PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage

DIRECT ACCESS
2 have access to this repository. 1 collaborator. 1 invitation.

Manage access

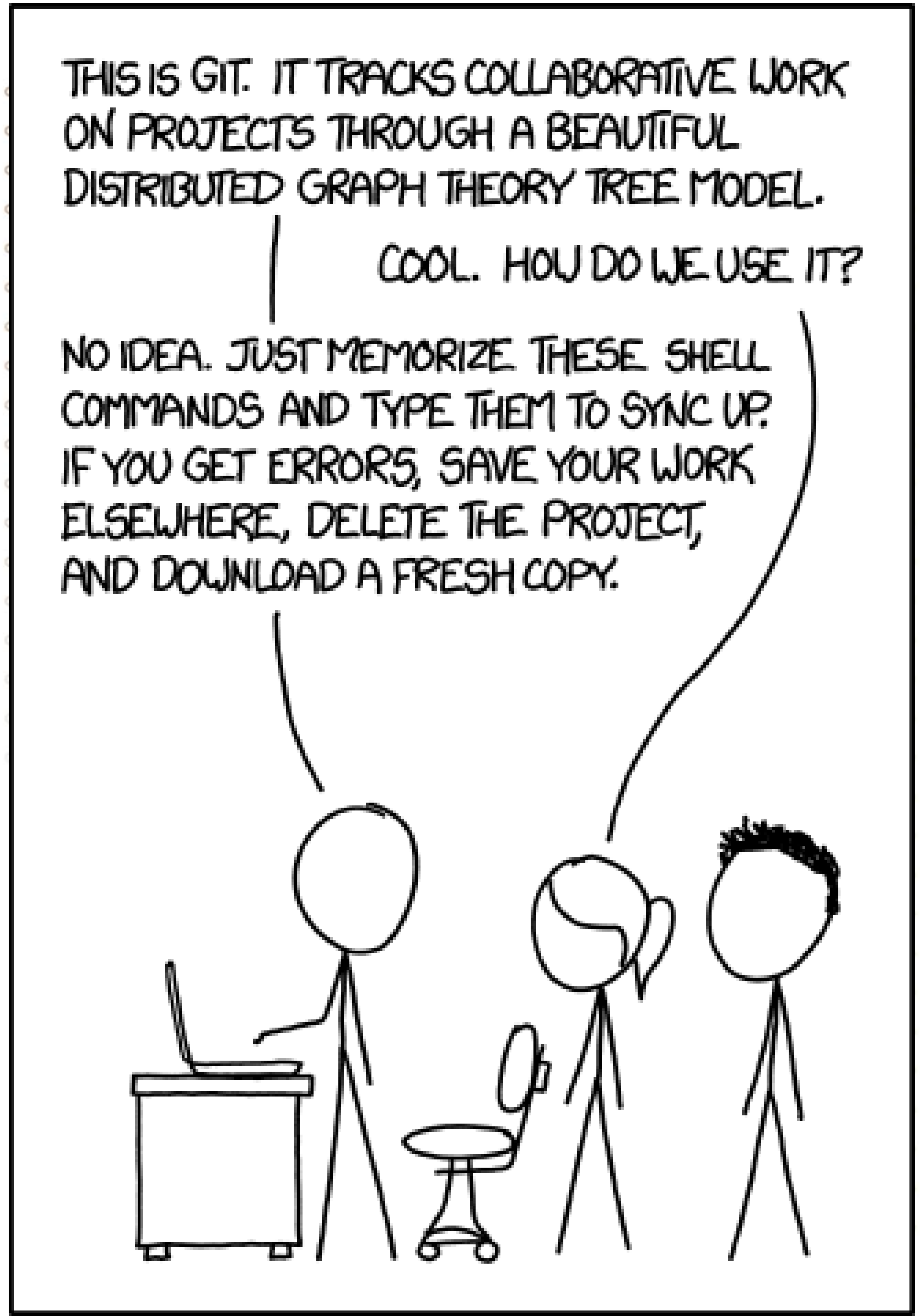
Add people

Select all   Type

email@demogr.mpg.de

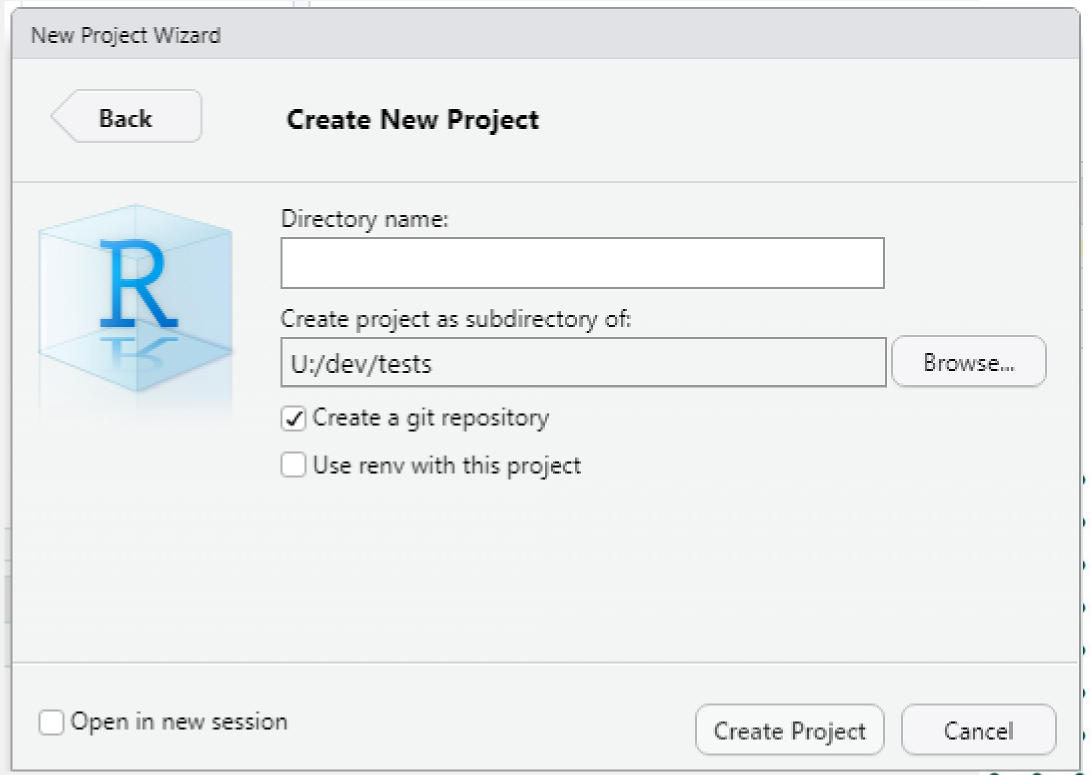
No Results

GIT IN REAL LIFE


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 MEMORIZE 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.

RSTUDIO AND GIT

- Rstudio creates a .git repository for you, if you want


New Project Wizard

Create New Project

Directory name:

Create project as subdirectory of: U:/dev/tests

[✓] Create a git repository

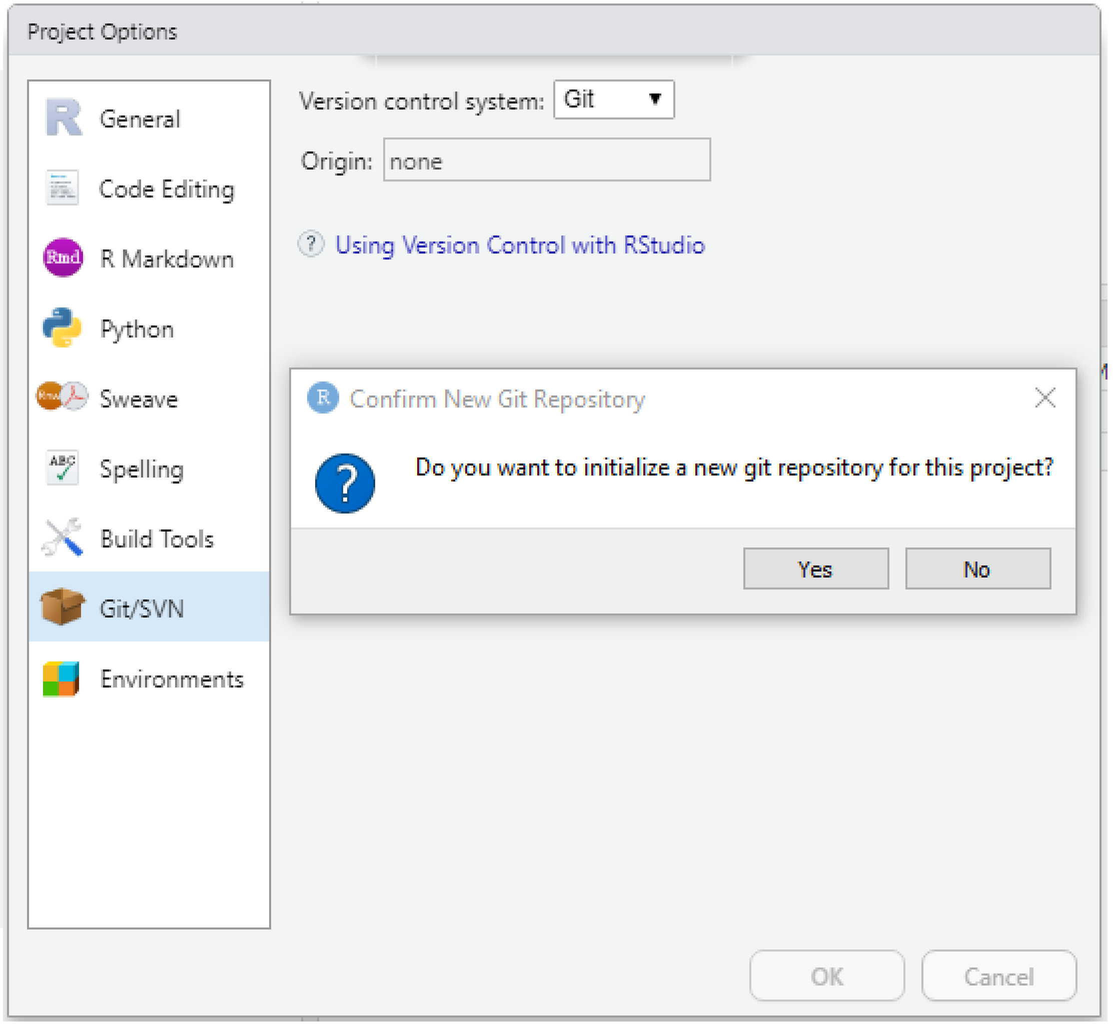
[ ] Use renv with this project

[ ] Open in new session

RSTUDIO AND GIT

- Rstudio creates a .git repository for you, if you want
- *But you can also initialize a new git repository on an existing project:*
- *Tools→ Version Control → …*

- *→A new register will appear*



Version control system: Git

Origin: none

Confirm New Git Repository

Do you want to initialize a new git repository for this project?

RSTUDIO AND GIT

- Rstudio creates a .git repository for you, if you want
- *But you can also initialize a new git repository on an existing project:*
- *Tools→ Version Control → …*

- *→A new register will appear*
- *…. Demonstration….*

RSTUDIO AND GIT

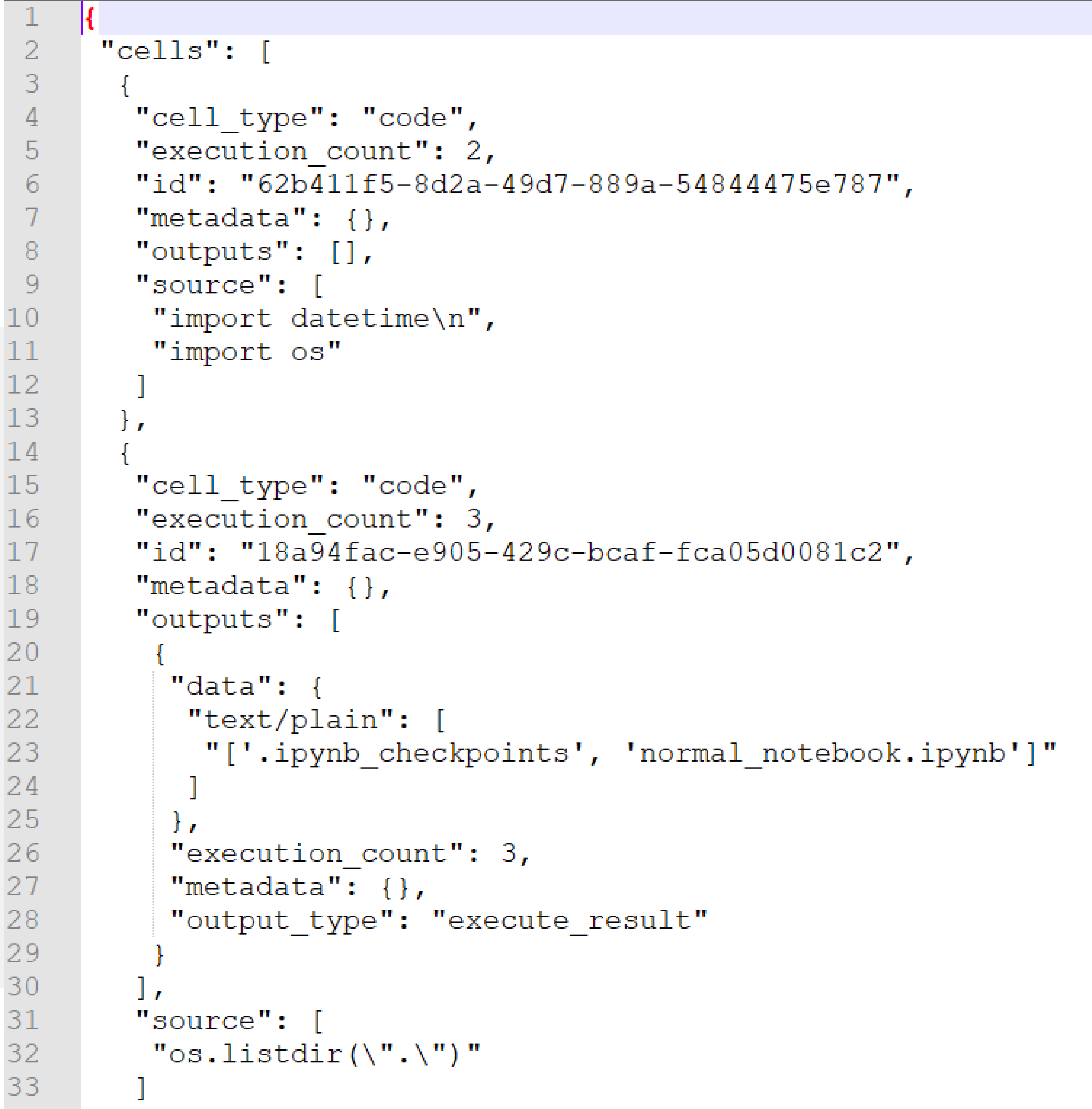
This is a very good introduction and reference for git in RStudio:

https://intro2r.com/use_git.html

JUPYTER AND GIT

Jupyter notebooks are JSON

JSON conflicts can often not be resolved automatically



```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 2,
   "id": "62b411f5-8d2a-49d7-889a-54844475e787",
   "metadata": {},
   "outputs": [],
   "source": [
    "import datetime\n",
    "import os"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "id": "18a94fac-e905-429c-bcaf-fca05d0081c2",
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "['.ipynb_checkpoints', 'normal_notebook.ipynb']"
      ]
     },
     "execution_count": 3,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "os.listdir(\".\")"
   ]
```

JUPYTER AND GIT

3 possible solutions:

- Use Git, but never work on the same file at once from 2 places
- Use JupyText, it is a JupyterLab extension that keeps a Notebook in sync with a .py-file that has the same code (but not the output)
- Use Visual Studio Code

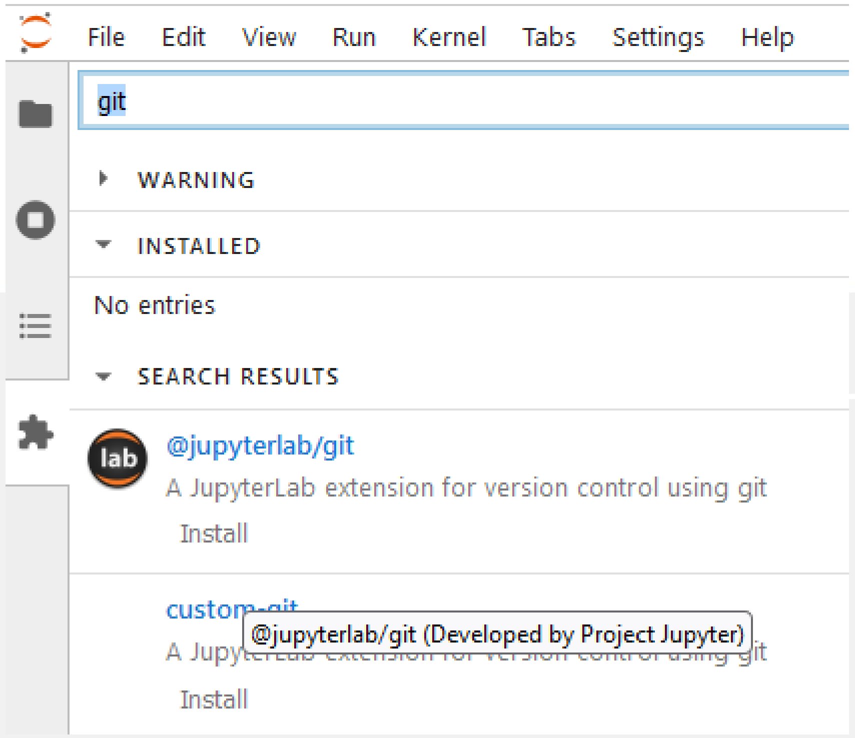
JUPYTERLAB AND GIT

Search for extension, install,

Stop JupyterLab

Install with pip

Start JupyterLab



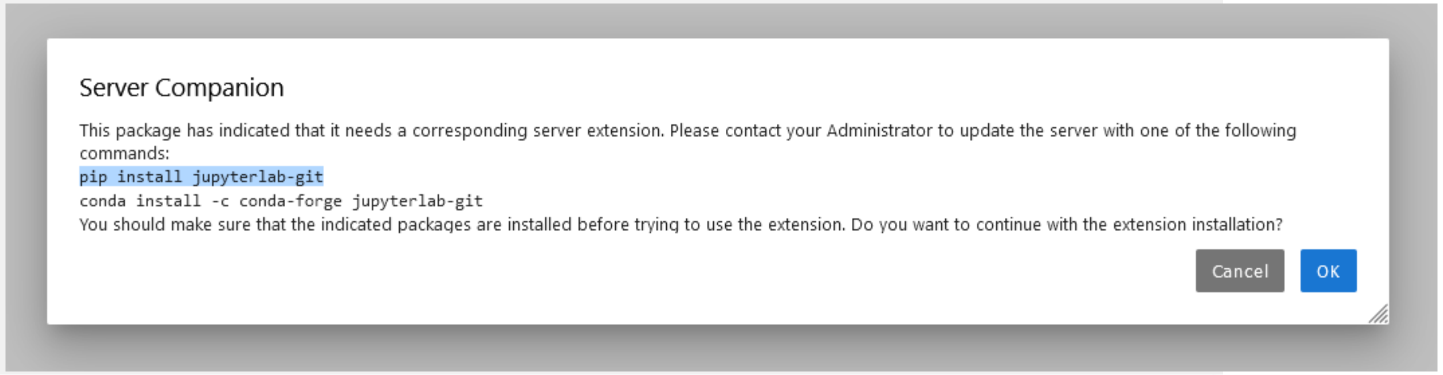
@jupyterlab/git

A JupyterLab extension for version control using git

custom-git

@jupyterlab/git (Developed by Project Jupyter)

A JupyterLab extension for version control using git

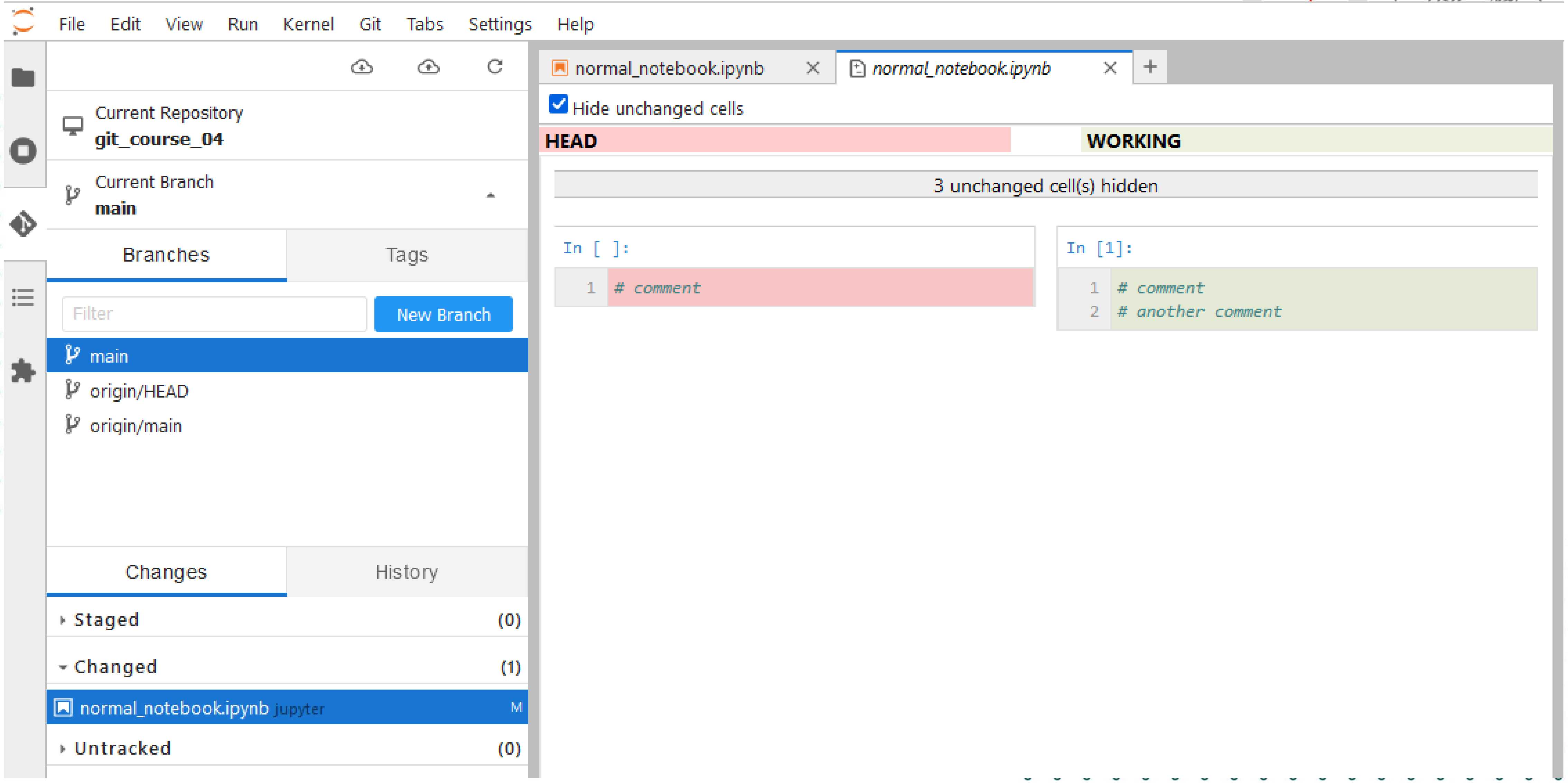


Server Companion

This package has indicated that it needs a corresponding server extension. Please contact your Administrator to update the server with one of the following commands:

```
pip install jupyterlab-git
conda install -c conda-forge jupyterlab-git
```

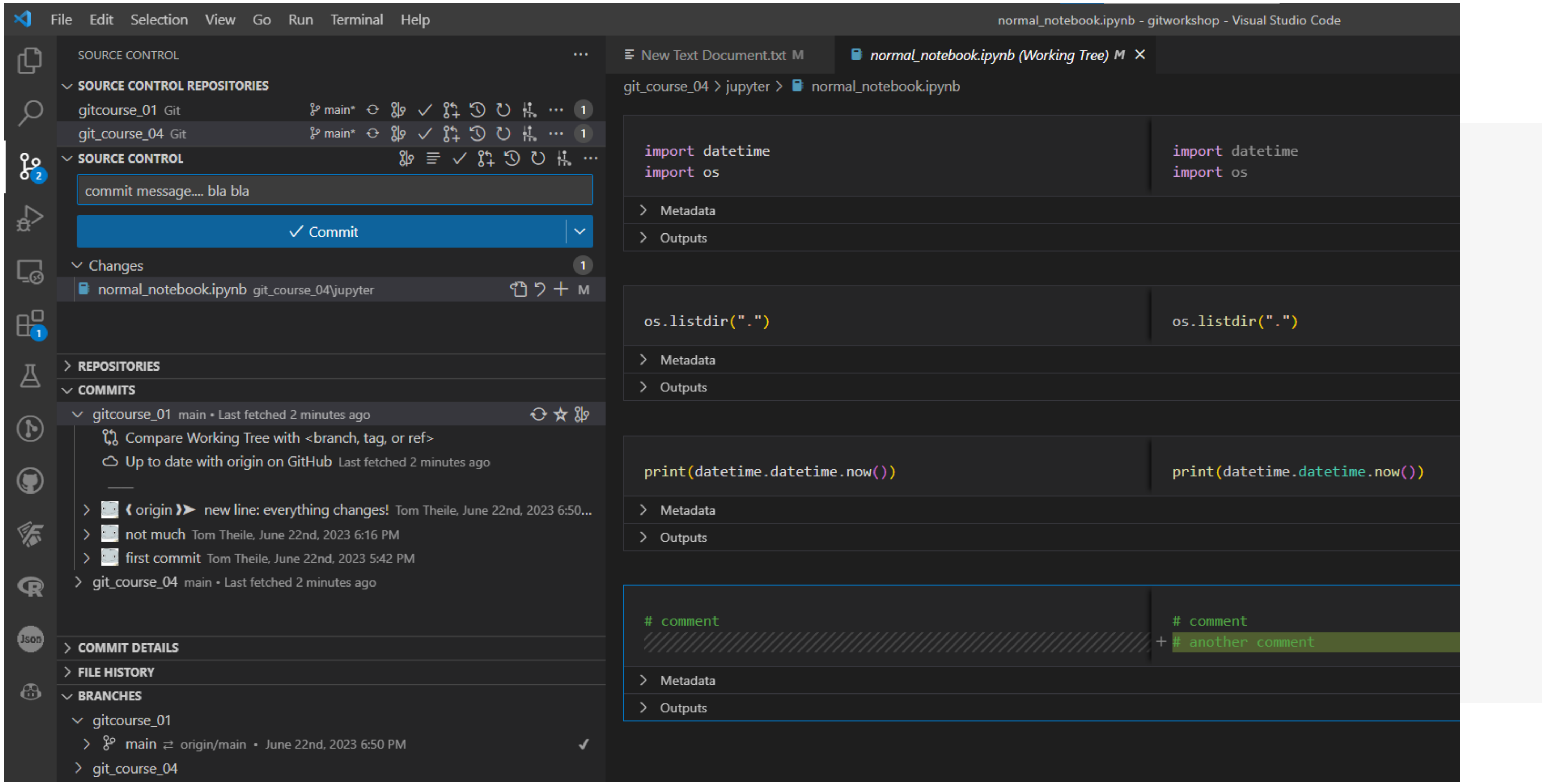
You should make sure that the indicated packages are installed before trying to use the extension. Do you want to continue with the extension installation?



```
In [ ]:
# comment
```

```
In [1]:
# comment
# another comment
```

GIT WITH VISUAL STUDIO CODE


File Edit Selection View Go Run Terminal Help
normal_notebook.ipynb - gitworkshop - Visual Studio Code
SOURCE CONTROL
SOURCE CONTROL REPOSITORIES
gitcourse_01 Git main*
git_course_04 Git main*
SOURCE CONTROL
Commit
Changes
REPOSITORIES
COMMITS
COMMIT DETAILS
FILE HISTORY
BRANCHES
New Text Document.txt M
normal_notebook.ipynb (Working Tree) M
git_course_04 > jupyter > normal_notebook.ipynb
Metadata
Outputs

commit message.... bla bla

normal_notebook.ipynb git_course_04\jupyter M

gitcourse_01 main • Last fetched 2 minutes ago
Compare Working Tree with <branch, tag, or ref>
Up to date with origin on GitHub Last fetched 2 minutes ago
❬origin❭ new line: everything changes! Tom Theile, June 22nd, 2023 6:50...
not much Tom Theile, June 22nd, 2023 6:16 PM
first commit Tom Theile, June 22nd, 2023 5:42 PM
git_course_04 main • Last fetched 2 minutes ago

gitcourse_01
main ⇄ origin/main • June 22nd, 2023 6:50 PM ✓
git_course_04

```
import datetime
import os
```

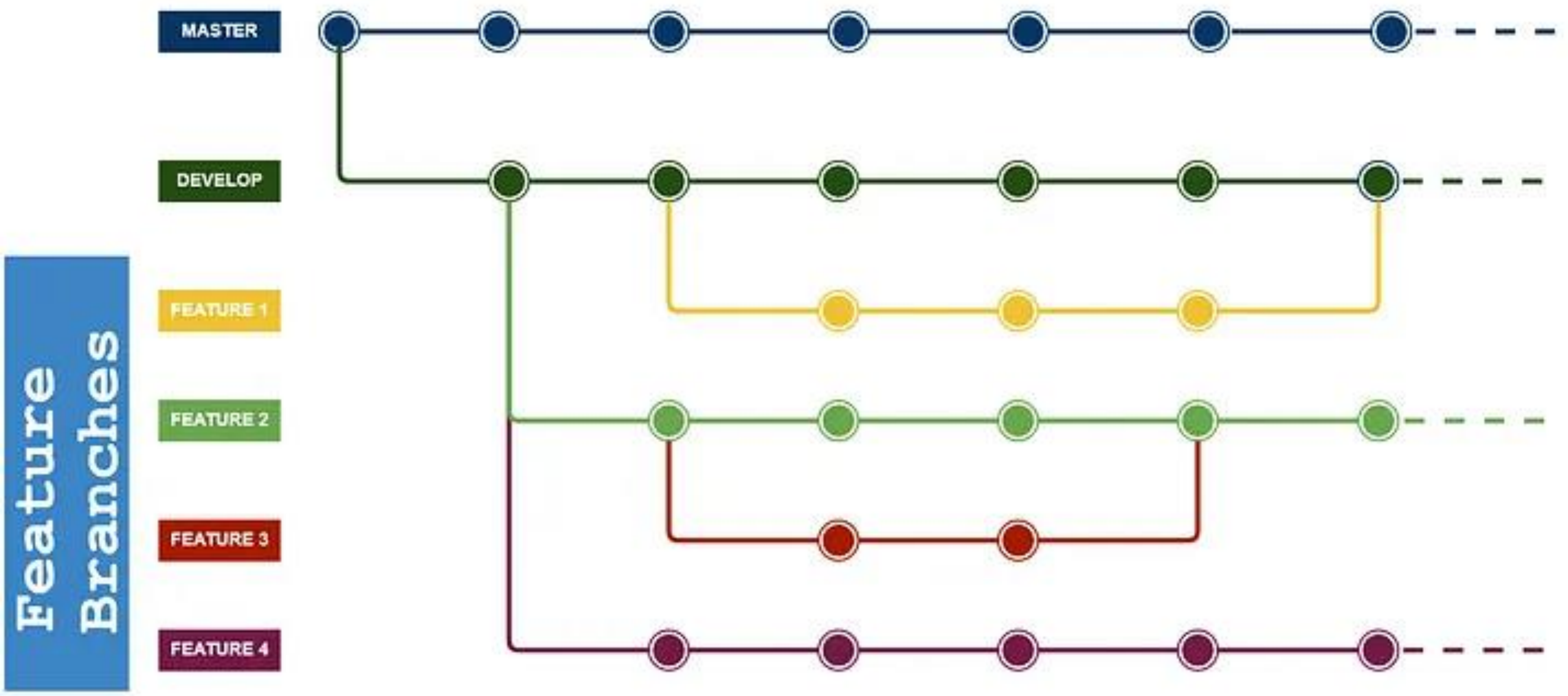
```
os.listdir(".")
```

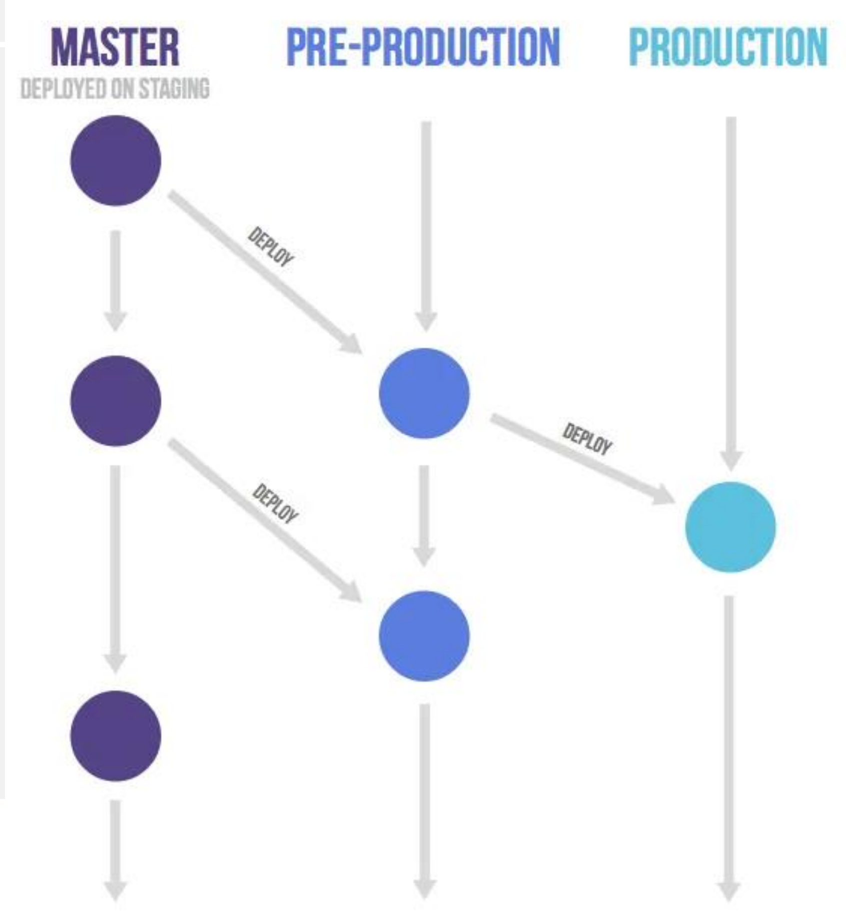
```
print(datetime.datetime.now())
```

```
# comment
+ # another comment
```

GIT WORKFLOWS

Source: https://medium.com/javarevisited/5-different-git-workflows-50f75d8783a7



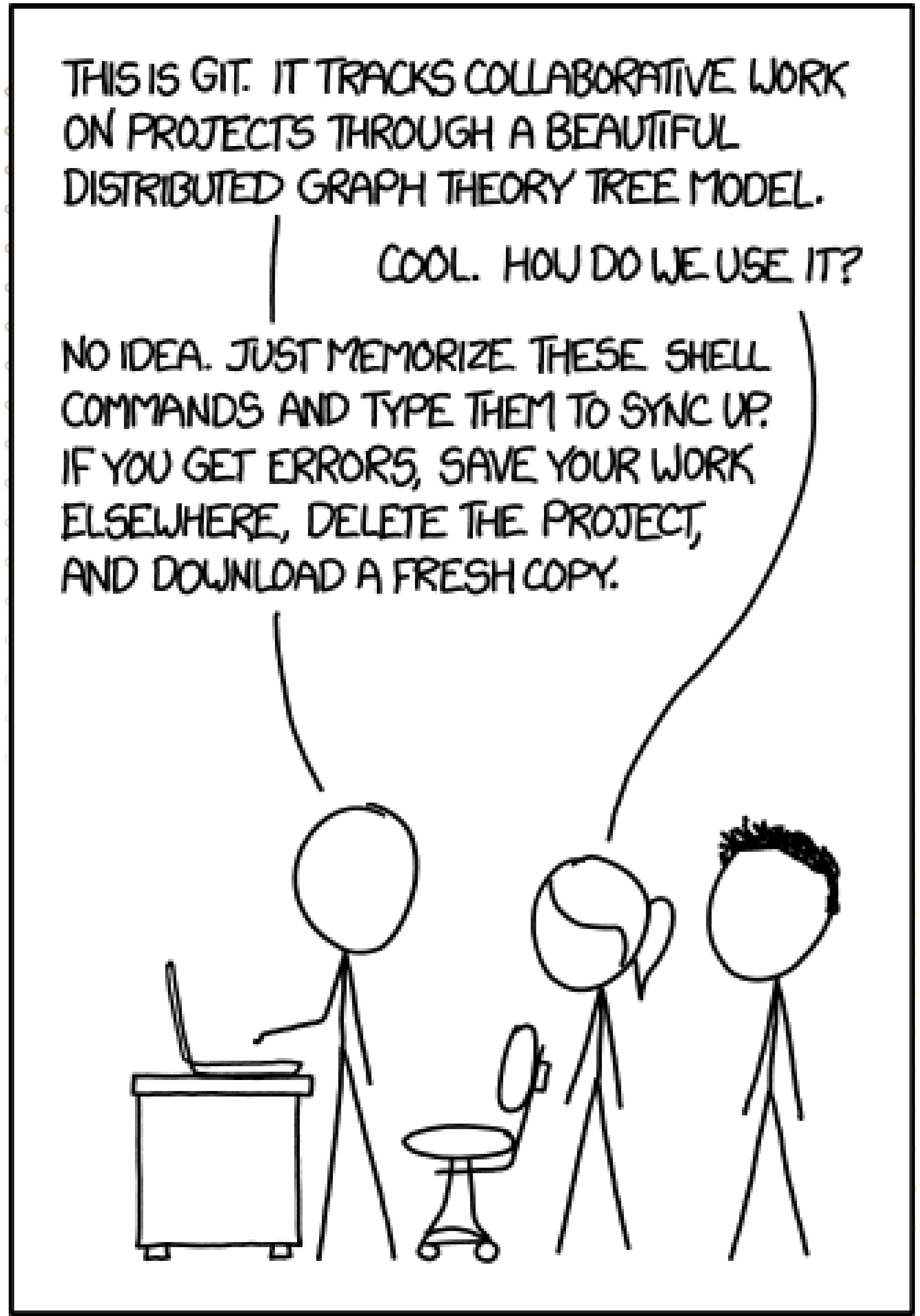


TROUBLESHOOTING

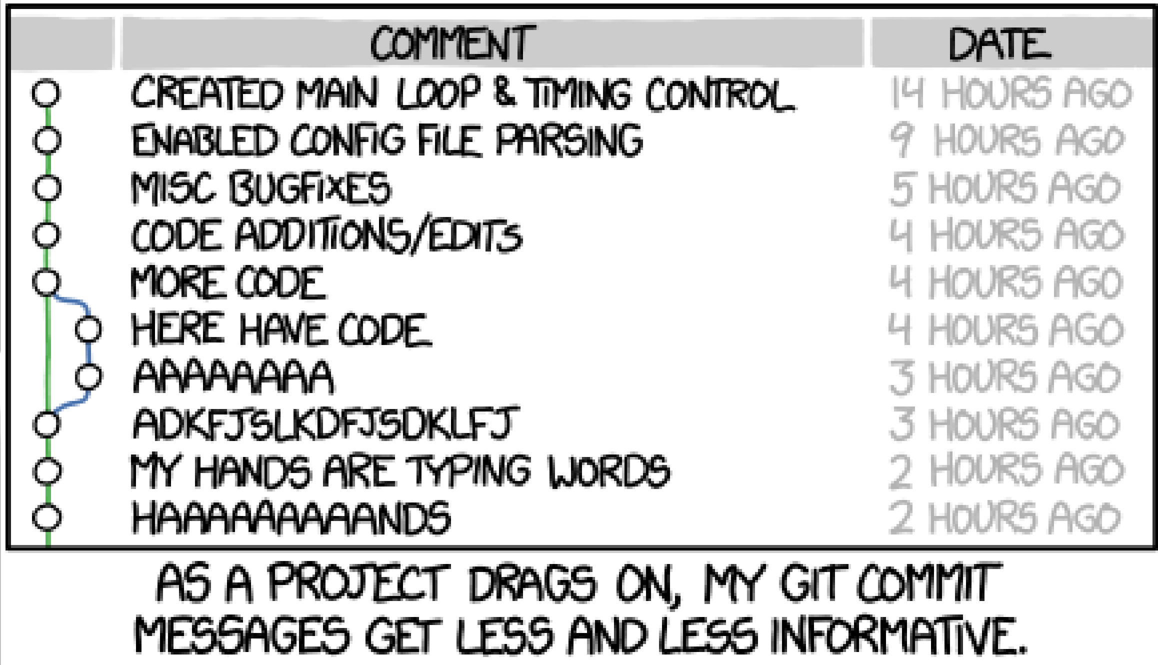
Remember: Git will only add things to its storage. It is really hard to delete committed files

Git gives helpful error messages on the command line. If they are not helpful enough, search engines will help you. If Google won't help you, ask someone from your institute ☺

GIT IN REAL LIFE


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 MEMORIZE 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.

BOB, ALL YOUR COMMIT MESSAGES SAY "SMALL CHANGES"!
NEXT TIME, PLEASE WRITE A MEANINGFUL MESSAGE
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
MEANINGFUL MESSAGE...
GIT COMMIT -M "WHAT IS LIFE IF FULL OF CARE. IF WE HAVE NO TIME TO STAND AND STARE..."
@_workchronicles
workchronicles.com
follow on Twitter & Instagram for more (and don't forget the underscore)



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Author et al. (YYYY

THANK YOU FOR YOUR ATTENTION!

Tom Theile

Software Developer
DCD
theile@demogr.mpg.de