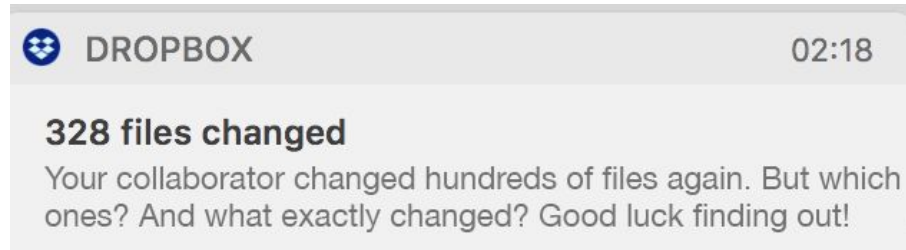# Versioning Using Git
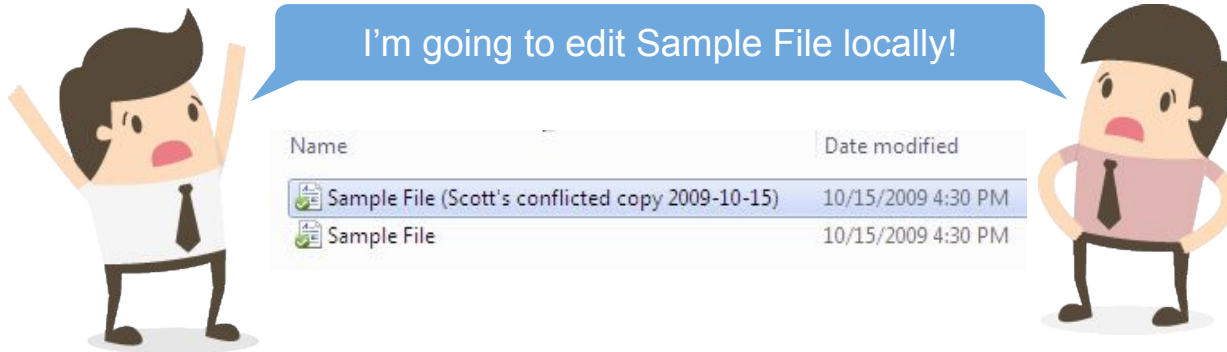
By **Pieter Delobelle** & Thomas Winters

# Working with code
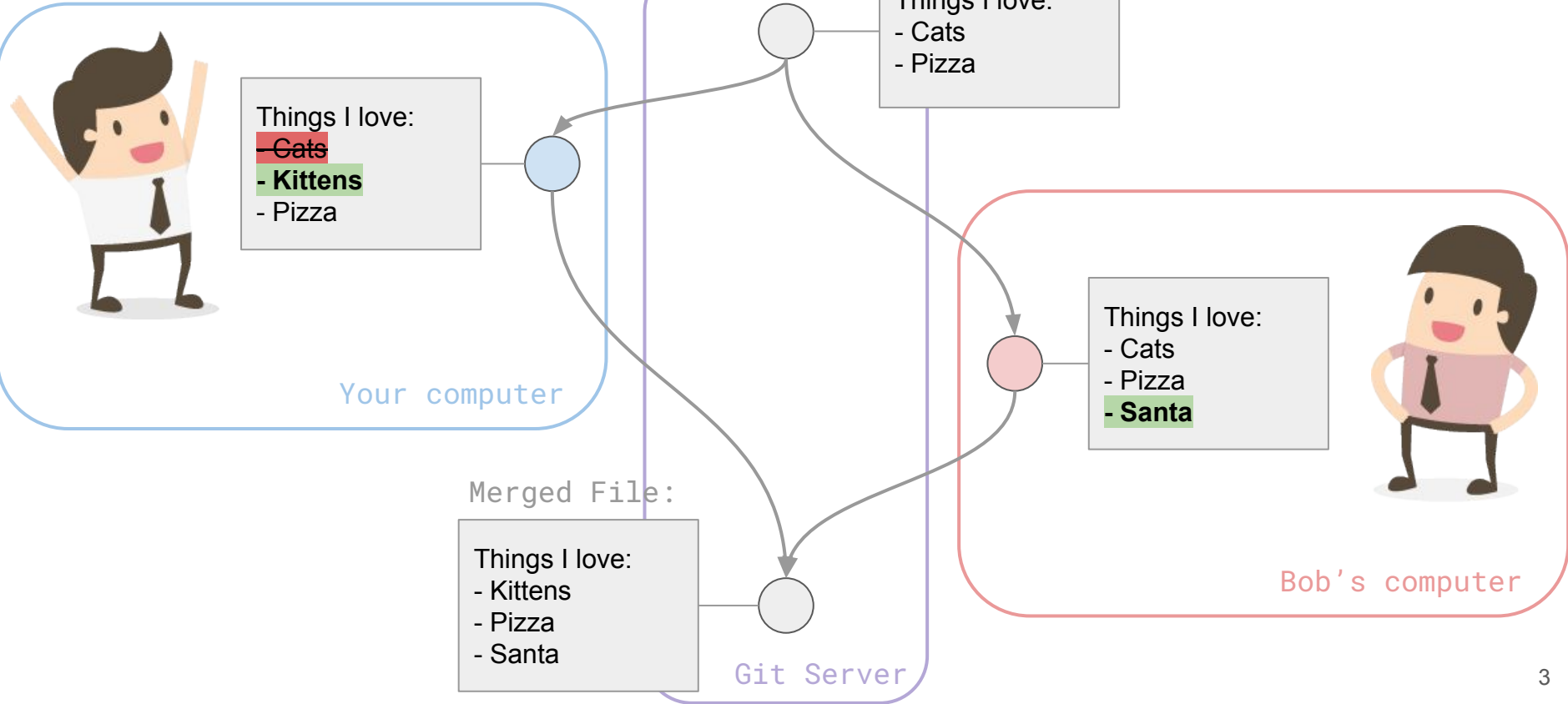
One might use classic cloud services for code sharing & backups (e.g. Dropbox), **BUT:**
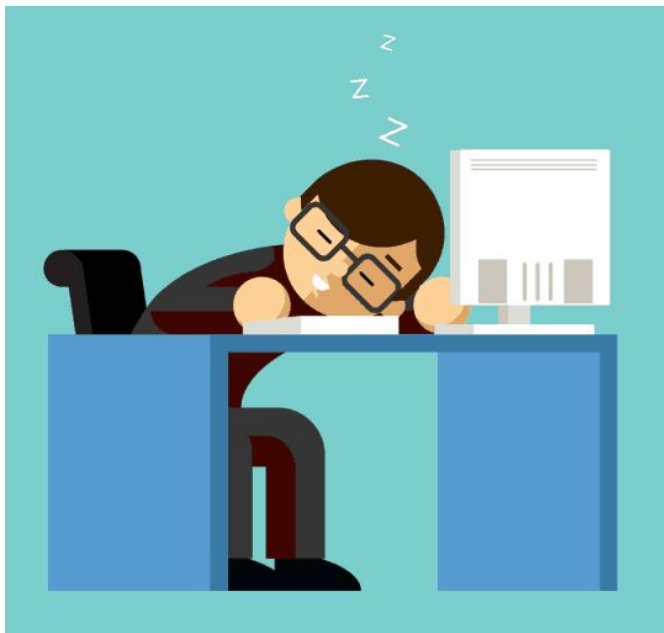


**DROPBOX** 02:18

**328 files changed**
Your collaborator changed hundreds of files again. But which ones? And what exactly changed? Good luck finding out!

***Or worse:*** *simultaneous edits* leading to conflicted copies



I'm going to edit Sample File locally!

| Name | | Date modified |
|------|--|---------------|
| Sample File (Scott's conflicted copy 2009-10-15) | | 10/15/2009 4:30 PM |
| Sample File | | 10/15/2009 4:30 PM |

# Git to the rescue

File: favorites.txt

Things I love:
- Cats
- Pizza

Your computer

Things I love:
- ~~Cats~~
- **Kittens**
- Pizza

Bob's computer

Things I love:
- Cats
- Pizza
- **Santa**

Merged File:

Things I love:
- Kittens
- Pizza
- Santa

Git Server

3

# But still useful when working on your own

**Not-so-hypothetical situation**

- Updating your scripts one late evening
- Next day, something broke

**Questions:**

- What broke where? Reset all progress from last night to get working version again?
- Which lines changed exactly?

→ Git to the rescue!

# How to share code & models afterwards?

Dropbox-like cloud storage?

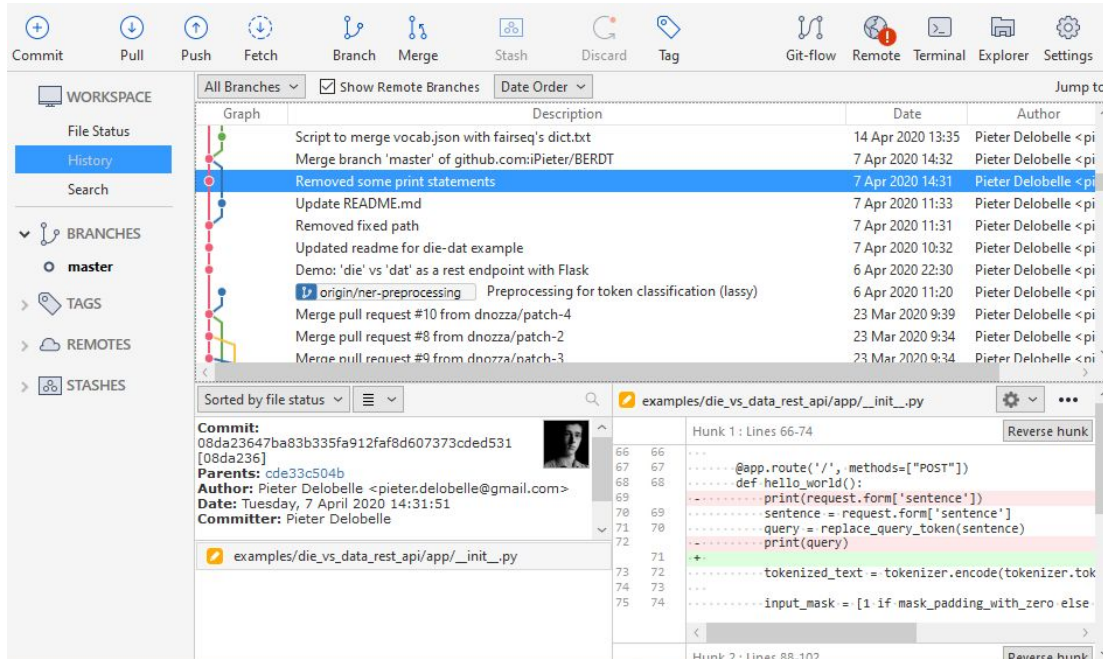**BUT:** What if full, or after quitting academia?

Upload to own website?

**BUT:** Lot of work, and less discoverability

What if want to discuss with downloaders & easily release new versions?

**Create a Git repository!**

# Getting visual overview on your repository: SourceTree



Use a **nice GUI** (e.g. *SourceTree*) for getting **an easy overview** to your repository

https://www.sourcetreeapp.com/

*(Command line is also often used, but less accessible for beginners)*

# Git Terminology

The layout of a Git Project

# Repository

Your code for one project lives in a **repository**

**Repository:** my-super-awesome-project

# Committing your changes

Every time you "save" your code, you create a new **commit**
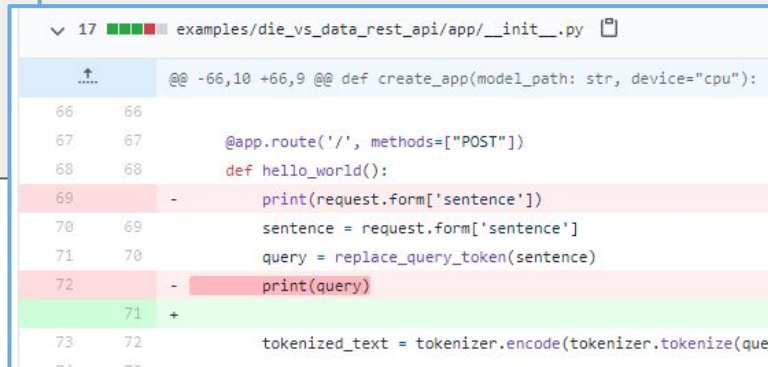
**Repository:** my-super-awesome-project

○

**Commit 1**

Initial commit

# Committing your changes

Every time you "save" your code, you create a new **commit**



**Repository:** my-super-awesome-project

**Commit 1**
Initial commit

```
  17 ▪▪▪▪□   examples/die_vs_data_rest_api/app/__init__.py  📋

  ⬆...     @@ -66,10 +66,9 @@ def create_app(model_path: str, device="cpu"):

  66   66
  67   67            @app.route('/', methods=["POST"])
  68   68            def hello_world():
  69        -           print(request.form['sentence'])
  70   69            sentence = request.form['sentence']
  71   70            query = replace_query_token(sentence)
  72        -           print(query)
       71   +
  73   72            tokenized_text = tokenizer.encode(tokenizer.tokenize(que
```

# Code branches

A lineage of commits is called a **branch**, which refers to the latest commit
For example the **main branch**

**Repository:** my-super-awesome-project

○——————○- - - - - - - - - - - - - - - - - - - - - - - - - - - - **main**

**Commit 1**
Initial commit

# Code branches

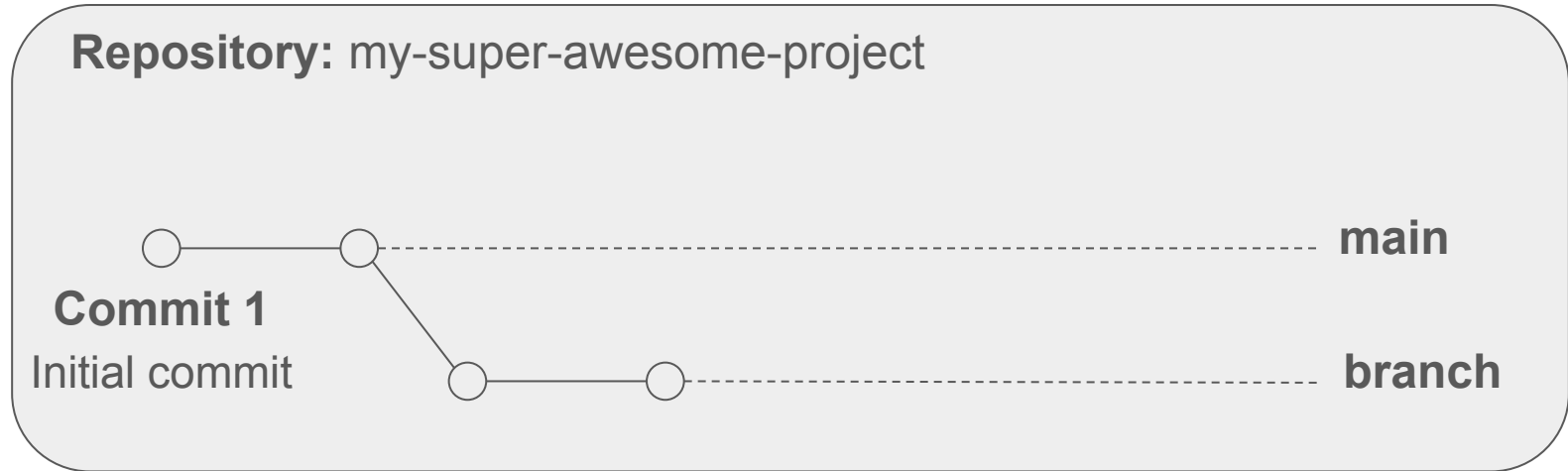You have a **main branch** (or **master**) and you can add other named branches for creating variations of your code

**Repository:** my-super-awesome-project

main

**Commit 1**
Initial commit

branch

# Code branches

You can commit on this new branch without affecting the main branch other collaborators might simultaneously be working on

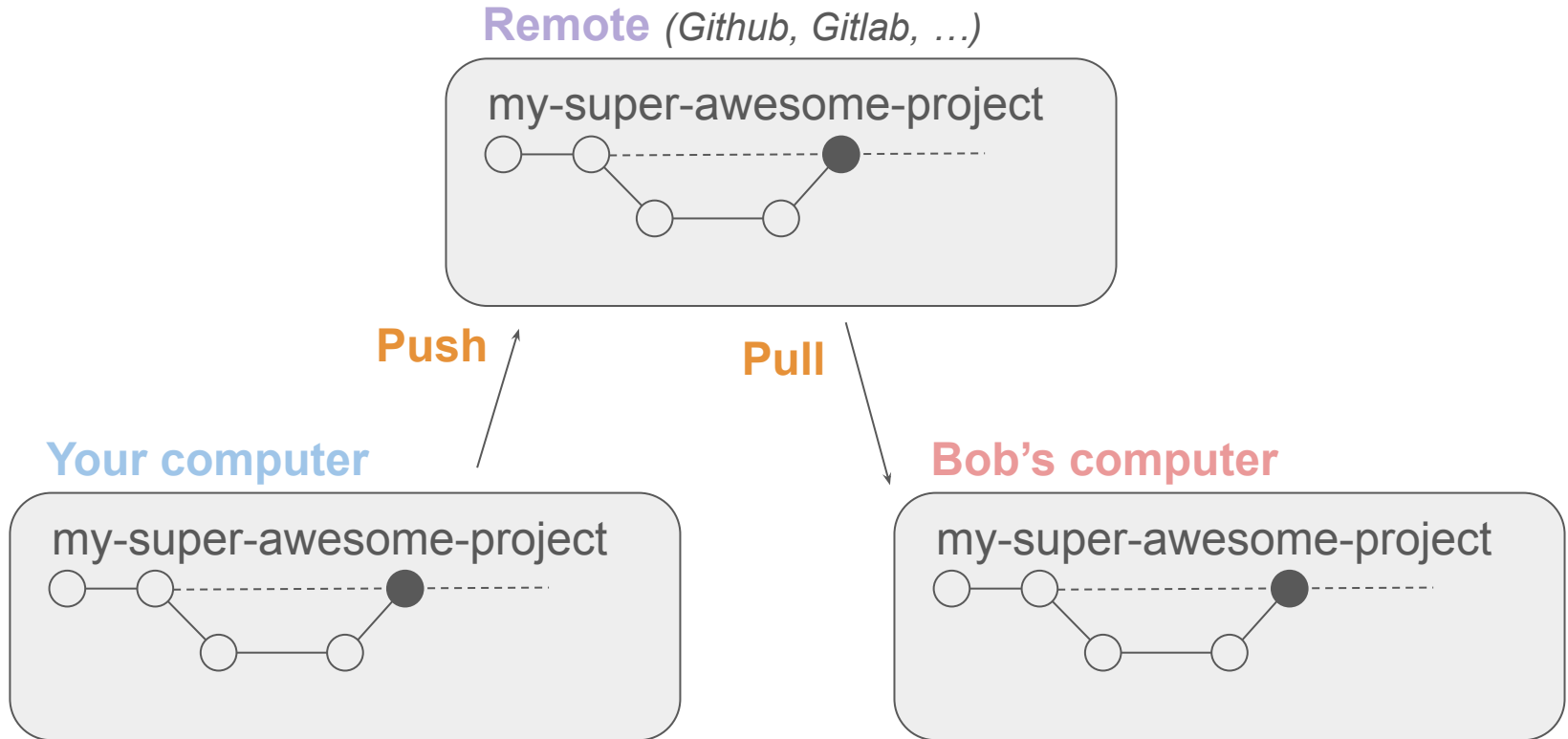**Repository:** my-super-awesome-project

**Commit 1**
Initial commit

**main**

**branch**

# Code branches

Repository: my-super-awesome-project

**main**

**Commit 1**
Initial commit

**branch**

# Merging branches

And when the code is ready, you can put it back on master with a **merge commit**



**Repository:** my-super-awesome-project

**merge commit**

**main**

**Commit 1**
Initial commit
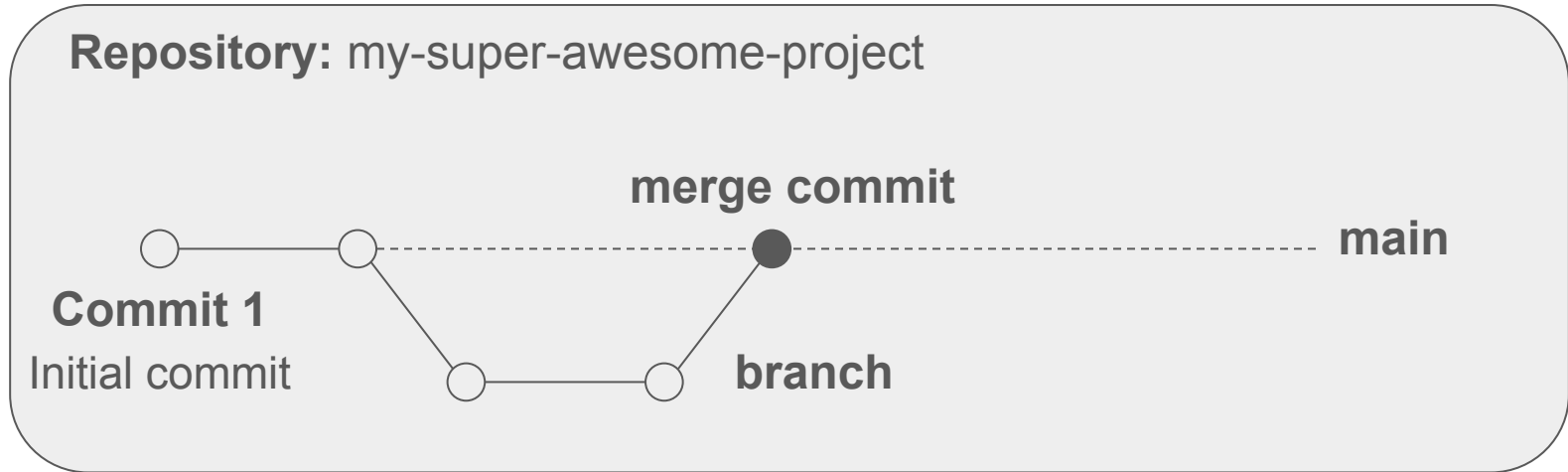
**branch**

# Push your changes to remote server, and pull for latest

**Remote** *(Github, Gitlab, …)*

my-super-awesome-project

**Push**

**Pull**

**Your computer**

my-super-awesome-project

**Bob's computer**

my-super-awesome-project

# GUI provides easy access to all these concepts

# Who can host your repositories?

**KU Leuven servers:** gitlab.kuleuven.be

**Third party service:** github.com, gitlab.com, … *(if project NDA allows)*

# Interface

sourcetreeapp.com

# Further tutorials

try.github.io/