Technische Universität Braunschweig

Suresoft
SUSTAINABLE RESEARCH SOFTWARE

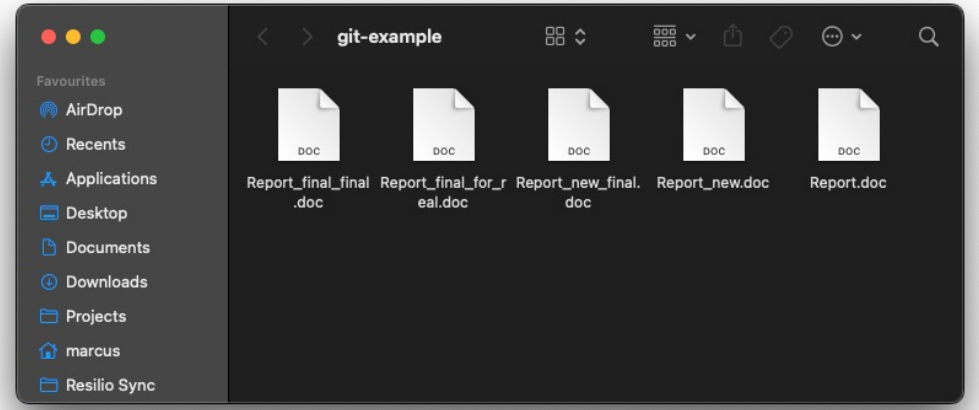
Version Control with Git

Sven Marcus, 16.06.22

Once upon a time before VCS

- *Manual* version control by saving new versions of files with different names or in different folders

We have all been there…



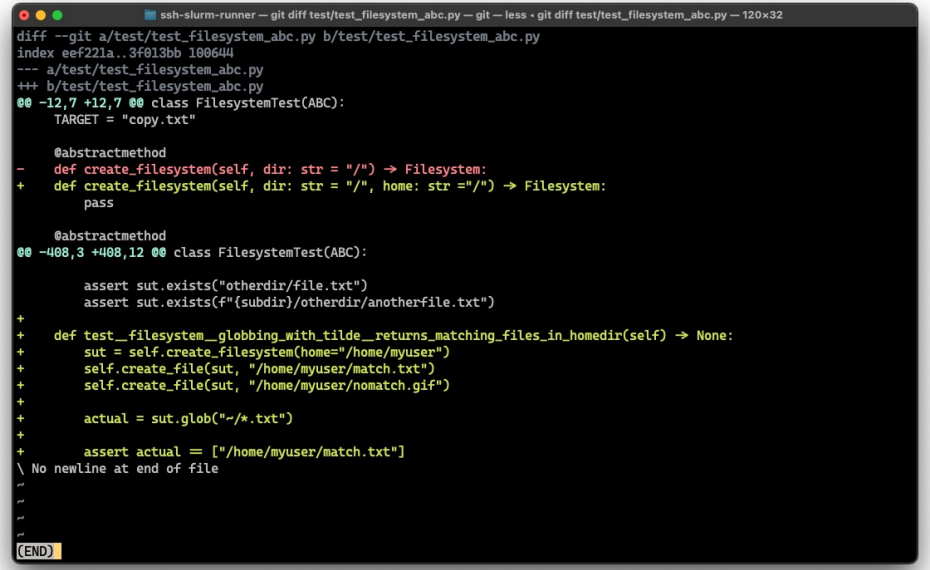
Error prone

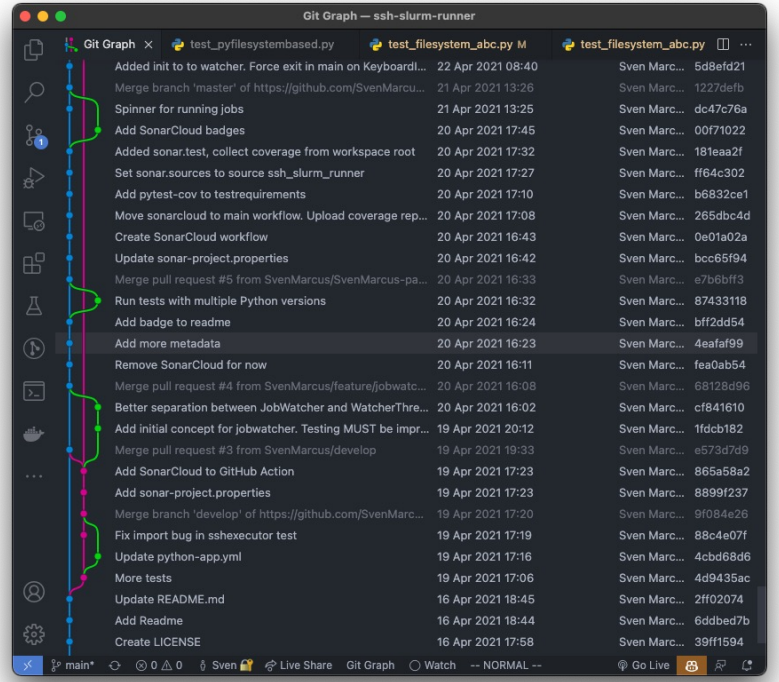
- easy to overwrite the wrong files
- easy to mess up naming

Hard to collaborate with multiple people

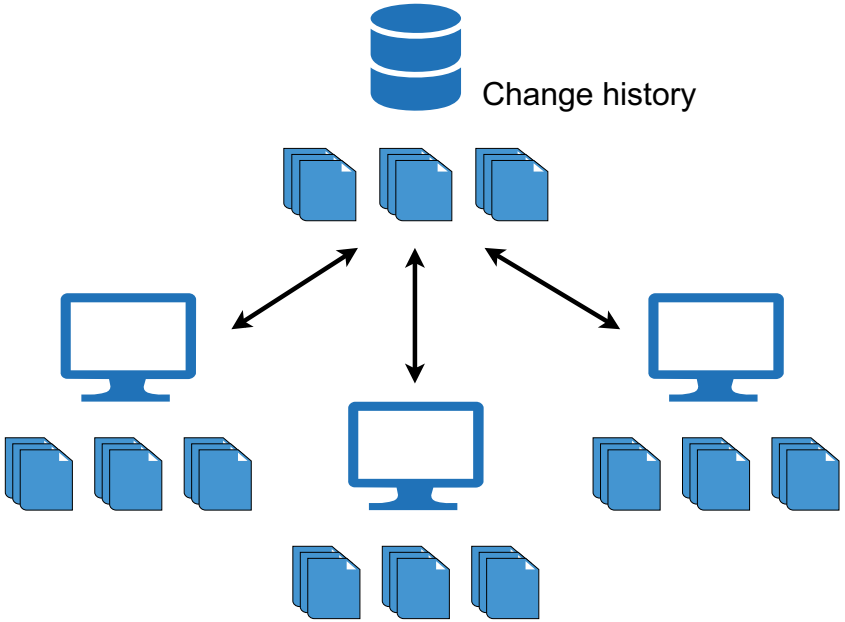
Version Control to the rescue

- Version Control Systems track changes we make to our files
- Allow us to roll back to previous versions





Centralized VCS



- Server manages different file versions in a *shared repository*
- Developers *checkout* the latest version from the server

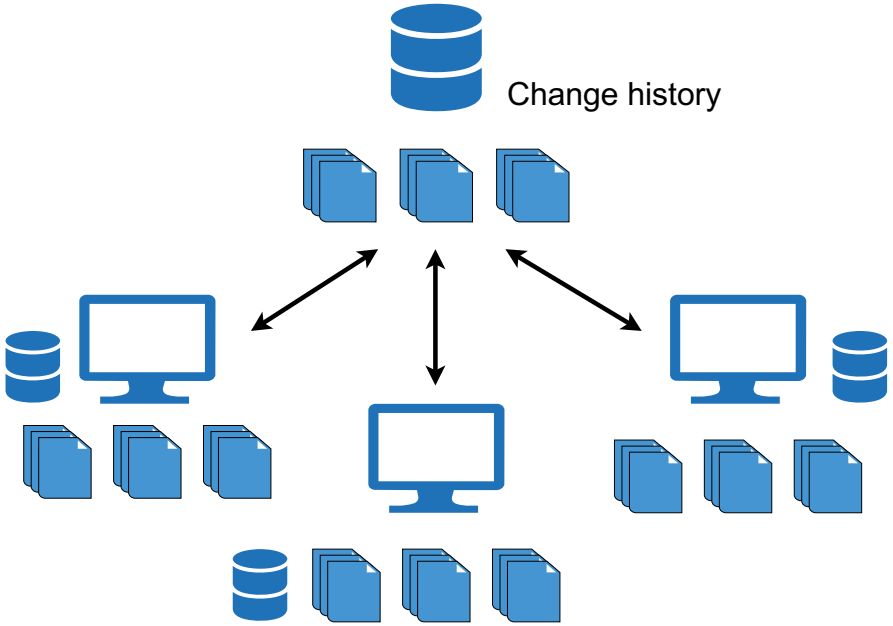
Pros:

- Enables collaboration
- Admins can control access

Cons:

- Single point of failure.
  When the server goes down nobody can collaborate

Distributed VCS



- Server and clients have a copy of the full history of the project

Pros:

- No single point of failure.
  If server dies, clients can restore history from their local repository

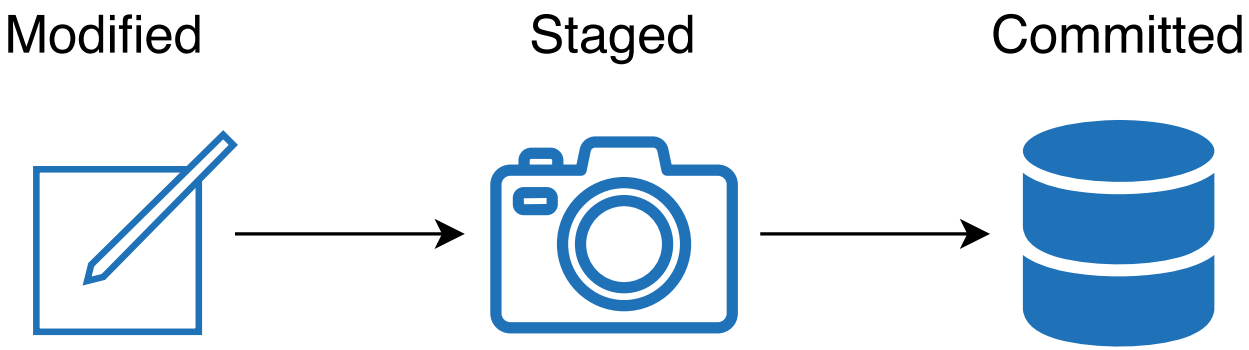
Cons:

- Possibly large repository size on every client machine

Git

- Distributed
- Can be used almost entirely on a local machine
- Stores snapshots of entire files instead of only tracking differences
- Uses checksums to track and ensure the integrity of files

The three file states



Modified

```
example-repo — marcus@hensolo — ../example-repo — -zsh — 120×32
marcus:example-repo/ (master*) $ git status                                  [15:58:52]
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file.txt

no changes added to commit (use "git add" and/or "git commit -a")
marcus:example-repo/ (master*) $                                             [15:58:55]
```

Modified

git status

Shows information about modified and staged files

Staged

```
example-repo — marcus@hensolo — ../example-repo — -zsh — 120×32
marcus:example-repo/ (master*) $ git add file.txt                    [16:01:32]
marcus:example-repo/ (master*) $ git status                          [16:01:49]
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file.txt

marcus:example-repo/ (master*) $                                     [16:01:56]
```

Staged

git add <files…>

Marks the specified files as staged and creates snapshots ready to be committed

Committed

```
example-repo — marcus@hensolo — ../example-repo — -zsh — 120×32
marcus:example-repo/ (master*) $ git commit -m "Added some text to file.txt"        [16:08:29]
[master 7616d3f] Added some text to file.txt
 1 file changed, 1 insertion(+)
marcus:example-repo/ (master) $                                                      [16:08:42]
```

Committed

git commit –m “<message>”

Commits the changes to the local repository

Adding a remote

- Log into the TU BS GitLab: git.rz.tu-bs.de
- Create a new project (without readme)

GitLab – creating a project

New project › **Create blank project**

Project name

example-repo

Project URL

https://git.rz.tu-bs.de/ Pick a group or namespace /

Project slug

example-repo

Project description (optional)

Description format

Visibility Level

- Private
  Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.
- Internal
  The project can be accessed by any logged in user except external users.
- Public
  The project can be accessed without any authentication.

Project Configuration

- [ ] Initialize repository with a README
  Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.
- [ ] Enable Static Application Security Testing (SAST)
  Analyze your source code for known security vulnerabilities. Learn more.

Create project | Cancel

GitLab – push an existing Git repository

We don't need this, because we don't have an existing server (remote) yet

Push an existing Git repository

```
cd existing_repo
~~git remote rename origin old_origin~~
git remote add origin ~~git@git.rz.tu-bs.de:s.marcus/example-repo.~~git
git push -u origin --all
git push -u origin --tags
```

Use: https://git.rz.tu-bs.de/<your-account>/example-repo.git

Adding a remote

```
example-repo — marcus@hensolo — ../example-repo — -zsh — 120×32
marcus:example-repo/ (master) $ git remote add origin https://git.rz.tu-bs.de/s.marcus/example-repo.git      [16:15:13]
marcus:example-repo/ (master) $                                                                              [16:42:55]
```

Adding a remote

git remote add <name> <url>

Adds a remote repository with the given name and URL.
By convention, the default remote is called *origin*.

Pushing to the remote

```
example-repo — marcus@hensolo — ../example-repo — -zsh — 120×32
marcus:example-repo/ (master) $ git push -u origin --all                    [16:42:55]
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 450 bytes | 450.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://git.rz.tu-bs.de/s.marcus/example-repo.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
marcus:example-repo/ (master) $                                             [16:44:47]
```

Second commit & push

```
example-repo — marcus@hensolo — ../example-repo — -zsh — 120×32
marcus:example-repo/ (master*) $ git add file.txt                          [17:17:59]
marcus:example-repo/ (master*) $ git commit -m "More important text in file.txt"  [17:18:13]
[master 37232ee] More important text in file.txt
 1 file changed, 1 insertion(+)
marcus:example-repo/ (master) $ git push                                   [17:18:28]
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 288 bytes | 288.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://git.rz.tu-bs.de/s.marcus/example-repo.git
   7616d3f..37232ee  master -> master
marcus:example-repo/ (master) $                                            [17:18:32]
```

For the second push, we can simply use *git push*, because the *master* branch already exists on our remote

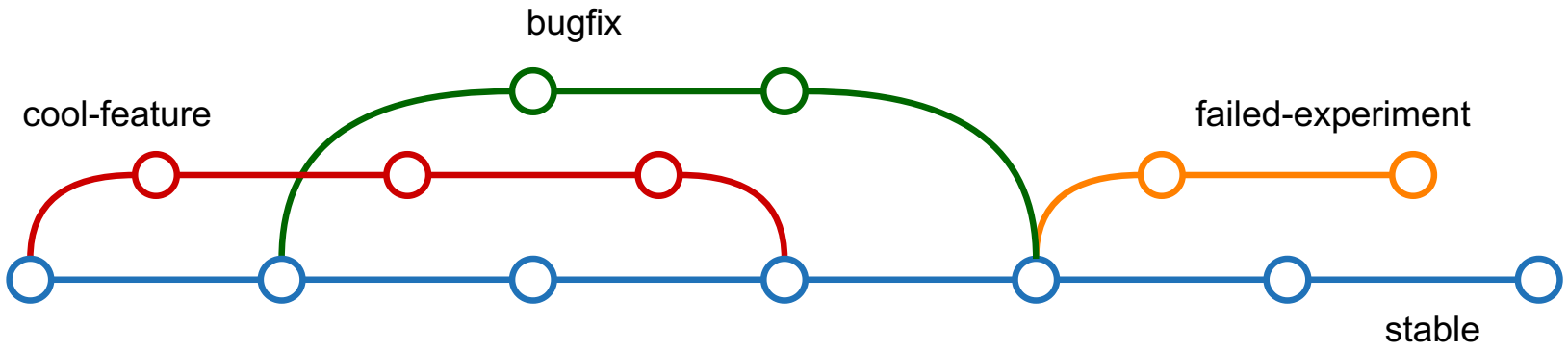
Pushing to the remote

git push

Pushes the committed changes to a remote repository.
Optionally specify a remote and a branch name to push to

Branches

- Enable parallel development
- Can be used to develop features, fix bugs or experiments without fear of breaking the main code line
- Danger: long living branches can diverge from the main code line a lot and cause hard to resolve merge conflicts



Create and switch to a new branch

```
example-repo — marcus@hensolo — ../example-repo — -zsh — 120×32
marcus:example-repo/ (master) $ git checkout -b my-cool-new-feature        [17:18:32]
Switched to a new branch 'my-cool-new-feature'
marcus:example-repo/ (my-cool-new-feature) $                                [17:56:01]
```

Create and switch to a new branch

git checkout <branchname>

Switch to a different branch of the repository.
Use the –*b* flag to create a new branch.

Make some changes on the new branch

```
example-repo — marcus@hensolo — ../example-repo — -zsh — 120×32
marcus:example-repo/ (my-cool-new-feature*) $ git status                    [9:29:49]
On branch my-cool-new-feature
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        newfile.txt

nothing added to commit but untracked files present (use "git add" to track)
marcus:example-repo/ (my-cool-new-feature*) $ git add .                     [9:29:52]
marcus:example-repo/ (my-cool-new-feature*) $ git commit -m "Added a new file"  [9:29:56]
[my-cool-new-feature afe3745] Added a new file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 newfile.txt
marcus:example-repo/ (my-cool-new-feature) $                                [9:30:03]
```

Push the branch to the remote

example-repo — marcus@hensolo — ../example-repo — -zsh — 120×32

```
marcus:example-repo/ (my-cool-new-feature) $ git push -u origin my-cool-new-feature          [10:15:21]
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 277 bytes | 277.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: To create a merge request for my-cool-new-feature, visit:
remote:   https://git.rz.tu-bs.de/s.marcus/example-repo/-/merge_requests/new?merge_request%5Bsource_branch%5D=my-cool-ne
w-feature
remote:
To https://git.rz.tu-bs.de/s.marcus/example-repo.git
 * [new branch]      my-cool-new-feature → my-cool-new-feature
Branch 'my-cool-new-feature' set up to track remote branch 'my-cool-new-feature' from 'origin'.
marcus:example-repo/ (my-cool-new-feature) $                                                  [10:16:43]
```

Merge a branch

```
example-repo — marcus@hensolo — ../example-repo — -zsh — 120×32
marcus:example-repo/ (my-cool-new-feature) $  git checkout master                [9:30:03]
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
marcus:example-repo/ (master) $ git merge my-cool-new-feature                    [9:32:16]
Updating 37232ee..afe3745
Fast-forward
 newfile.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 newfile.txt
marcus:example-repo/ (master) $                                                  [9:32:26]
```

Merge a branch

git merge <branch>

Merges the specified branch into the currently selected one.

Learn more about (research) software engineering

Suresoft is a project aiming to increase the sustainability of research software and its development process by teaching scientists about principles and practices of the Software Engineering discipline.

Full list of planned workshops at: tu-braunschweig.de/suresoft

Next Workshop: 11.07.22

Topic: Clean Code and Refactoring

- We will learn how to
  - write readable and maintainable code
  - improve the structure of existing code

Register here