

Workshop git

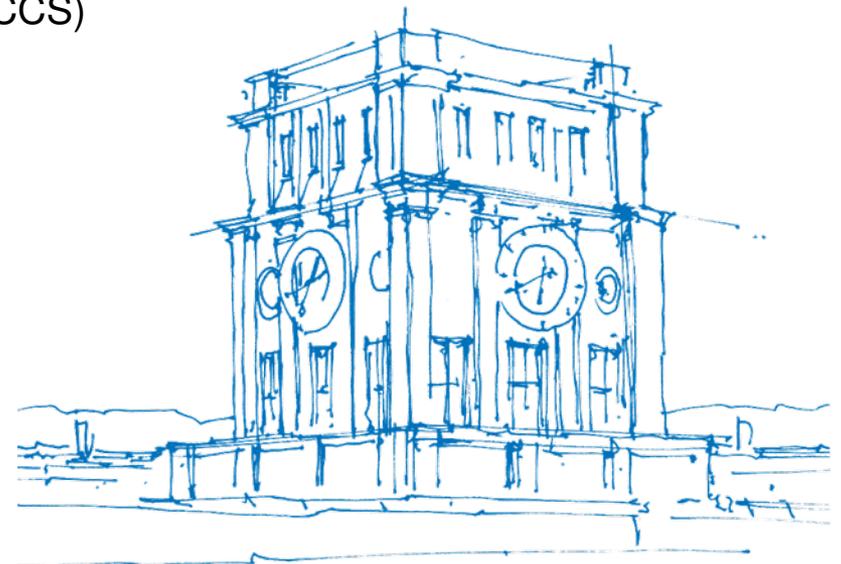
Fabio Grati

Technical University of Munich

Faculty of Informatics

Chair of Scientific Computing in Computer Science (SCCS)

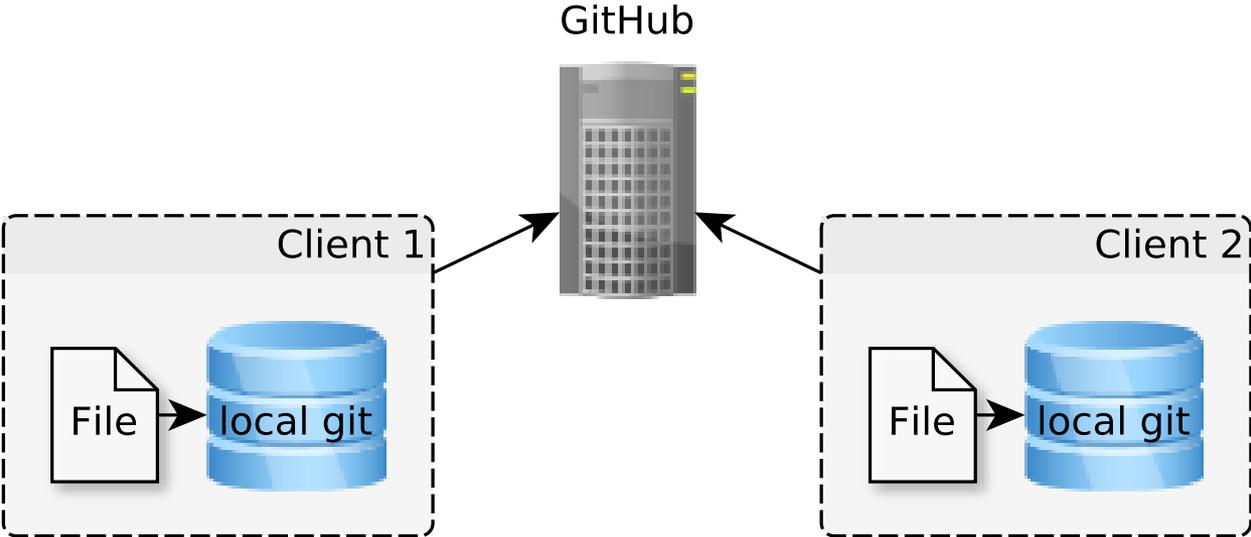
TU Kaiserslautern, 12.10.2018



TUM Uhrenturm

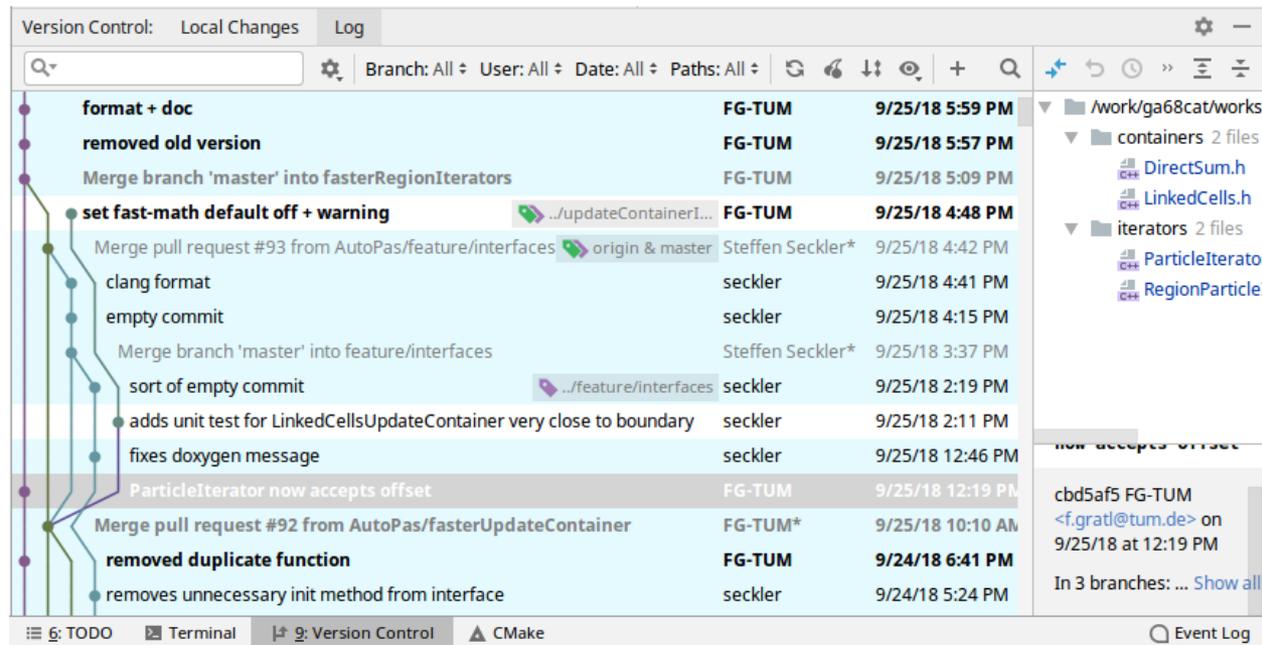
What is git?

- Officially:
`git - the stupid content tracker`
- Distributed Version Control System (DVCS)
 \Rightarrow local + remote repository!
- git-scm.com



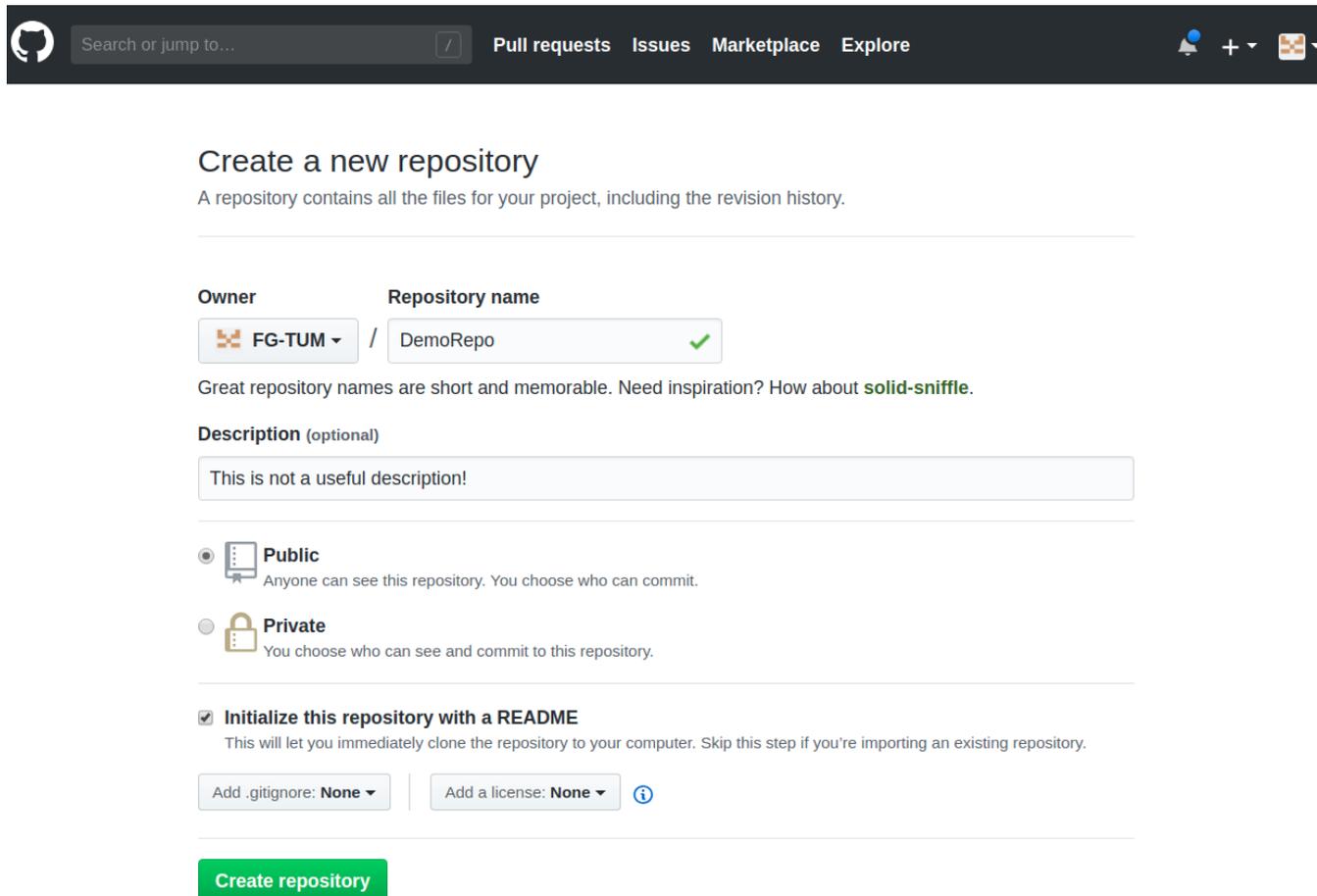
Getting help

- Manpages: `man git`, `man gittutorial`, `man giteveryday`
- Built-in help: `git help $COMMAND` (same as `man git-$COMMAND`)
- GUI tools for all platforms exist (but not covered here).
- IDEs can handle git (eclipse, intelliJ, CLion...)



A new repository (GitHub)

- Create a new repository on GitHub or look for one you like.



Search or jump to... Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner **Repository name**

FG-TUM / DemoRepo ✓

Great repository names are short and memorable. Need inspiration? How about [solid-sniffle](#).

Description (optional)

This is not a useful description!

Public
Anyone can see this repository. You choose who can commit.

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

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None** ⓘ

Create repository

A new repository (GitLab)

- Create a new repository on GitLab or look for one you like.

The screenshot shows the 'New project' form in GitLab. The form is divided into three tabs: 'Blank project', 'Create from template', and 'Import project'. The 'Blank project' tab is selected. The form contains the following fields and options:

- Project path:** A dropdown menu showing 'https://gitlab.lrz.de/' and a text input field containing 'ga68cat'.
- Project name:** A text input field containing 'DemoRepo'.
- Project description (optional):** A text area containing the placeholder text 'This is not a useful description!'.
- Visibility Level:** Three radio button options: 'Private' (with a lock icon), 'Internal' (with a shield icon), and 'Public' (with a globe icon). The 'Public' option is selected.
- Initialize repository with a README:** A checked checkbox with the label 'Initialize repository with a README'. Below it, a note reads: 'Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.'

At the bottom of the form, there are two buttons: 'Create project' (a green button) and 'Cancel' (a white button with a grey border).

Bring it on your machine (GitHub)

- Clone via **ssh** (requires public-key upload¹) or https.

The screenshot shows the GitHub interface for a repository named 'DemoRepo' by 'FG-TUM'. The repository has 1 commit, 1 branch, 0 releases, and 1 contributor. The 'Clone or download' button is highlighted with a red box. A dropdown menu is open, showing the following options:

- Clone with SSH (with a help icon) and Use HTTPS
- Use an SSH key and passphrase from account.
- git@github.com:FG-TUM/DemoRepo.git (with a copy icon)
- Download ZIP

¹<https://help.github.com/articles/connecting-to-github-with-ssh/>

Bring it on your machine (GitHub)

- Clone via **ssh** (requires public-key upload¹) or https.

Search or jump to... Pull requests Issues Marketplace Explore

FG-TUM / DemoRepo Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

This is not a useful description! Edit

Manage topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

FG-TUM Initial commit

README.md Initial commit

README.md

DemoRepo

This is not a useful description!

Clone with SSH Use HTTPS

Use an SSH key and passphrase from account.

git@github.com:FG-TUM/DemoRepo.git

Download ZIP

¹<https://help.github.com/articles/connecting-to-github-with-ssh/>

Bring it on your machine (GitLab)

- Clone via **ssh** (requires public-key upload²) or https.

The screenshot shows the GitLab web interface for a project named 'DemoRepo'. The left sidebar contains navigation options: Project, Details, Activity, Cycle Analytics, Repository, Issues (0), Merge Requests (0), Wiki, and Settings. The main content area shows the project details, including the repository name 'DemoRepo', a description 'This is not a useful description!', and the project ID '25304'. A red box highlights the SSH clone URL 'git@gitlab.lrz.de:ga68cat/Demo' and the 'Copy URL to clipboard' button. Below the URL, there are buttons for 'Add Changelog', 'Add License', 'Add Contribution guide', and 'Set up CI/CD'. The commit history shows an 'Initial commit' by 'FG-TUM' 41 seconds ago with the commit hash 'b9f2758a'. A table below lists the files in the repository:

Name	Last commit	Last update
README.md	Initial commit	42 seconds ago

²<https://docs.gitlab.com/ee/ssh/>

Bring it on your machine

git clone <repository> [<directory >]

- Copies remote repository to local machine.
- Fetches all branches.
- Requires target folder to be empty!

Example:

```
1 ~$ git clone git@gitlab.lrz.de:ga68cat/DemoRepo.git
2 Cloning into 'DemoRepo'...
3 ~$ cd DemoRepo
4 ~/DemoRepo$ ls -a
5 . .. .git README.md
```

Gaining Overview

git status

- Show status of current working copy.
- List modifications, new files, deletions, merge conflicts...
- Always provides hints what to do! (not shown here)

git log [-<number>]

- Show information about the last n commits.
- Range based query also possible.

Example:

```
1 ~/DemoRepo$ git status
2 On branch master
3 Your branch is up–to–date with 'origin/master'.
4 nothing to commit, working directory clean
5 ~/DemoRepo$ git log –1
6 commit 139e2f8be08bb6fba96b27fd30f31008880584d4
7 Author: FG–TUM <FG–TUM@users.noreply.github.com>
8 Date: Fri Oct 5 11:52:47 2018 +0200
9     Initial commit
```

Ignoring Stuff

.gitignore

- List of files and paths to be ignored by git.
- Accepts "*" as wildcard.
- This file should be uploaded as any other.
- Useful for output or IDE files.

Example:

```
1 ~/DemoRepo$ touch a.foo b.foo
2 ~/DemoRepo$ echo "*.foo" >> .gitignore
3 ~/DemoRepo$ mkdir dir && touch dir/one dir/two dir/c.foo
4 ~/DemoRepo$ echo "dir" >> .gitignore
```

Small Changes

- Modify README.md
- Create a new file FileA.txt

```
1 ~/DemoRepo$ echo "some useless text..." >> README.md
2 ~/DemoRepo$ echo "Let there be text\nCommon line." >> FileA.txt
3 ~/DemoRepo$ ls
4 FileA.txt README.md
5 ~/DemoRepo$ git status
6 On branch master
7 Your branch is up–to–date with 'origin/master'.
8 Changes not staged for commit:
9     modified: README.md
10 Untracked files:
11     FileA.txt
12 no changes added to commit
```

Committing Changes

git add

- Track new files.
- Stage existing files.

git commit [--amend | --message | --signoff]

- Commit changes to local repository.
- Checkpoint to revert or compare to.

Example:

```

1 ~/DemoRepo$ git add FileA.txt README.md
2 ~/DemoRepo$ git status
3 On branch master
4 Your branch is up-to-date with 'origin/master'.
5 Changes to be committed:
6   new file:   FileA.txt
7   modified:  README.md
8 ~/DemoRepo$ git commit --message "meaningful message"
9 ~/DemoRepo$ git status
10 On branch master
11 Your branch is ahead of 'origin/master' by 1 commit.

```

Pushing Changes

git push

- Sends all committed changes to remote.

Example:

```
1 ~/DemoRepo$ git push
2 To git@gitlab.lrz.de:ga68cat/DemoRepo.git
3 139e2f8..014affb master -> master
```

Pulling Changes

`git pull [--rebase]`

- Updates all files in the local branch.
- Updates information about other remote branches.
- rebase: Put local changes on top of remote instead of merging.

Example: (Suppose someone added a line break in README.md.)

```
1 ~/DemoRepo$ git pull
2 From git@gitlab.lrz.de:ga68cat/DemoRepo.git
3 014affb..d20ae23 master -> origin/master
4 Updating 014affb..d20ae23
5 Fast-forward
6 README.md | 1 +
7 1 file changed, 1 insertion(+)
```

Starting a new Feature

git checkout <branch> | <File>

- Switch to existing branch.
- OR reset <File> to last commit.

git branch [<new_branch>]

- Create a new branch.
- OR list all local branches.

Example:

```
1 ~/DemoRepo$ git branch branchForAwesomeFeature
2 ~/DemoRepo$ git branch
3 branchForAwesomeFeature
4 * master
5 ~/DemoRepo$ git checkout branchForAwesomeFeature
6 Switched to branch 'branchForAwesomeFeature'
```

Merging branches

git merge <branch>

- Merges given branch in current branch.

Example: (Suppose we changed FileA.txt on the new branch and someone else on master.)

```
1 ~/DemoRepo$ git merge master
2 Auto-merging FileA.txt
3 CONFLICT (content): Merge conflict in FileA.txt
4 Automatic merge failed; fix conflicts and then commit the result.
5 ~/DemoRepo$ git status
6 On branch branchForAwesomeFeature
7 You have unmerged paths.
8 Unmerged paths:
9   both modified: FileA.txt
```

Resolving conflicts

git mergetool (needs to be configured)

- vimdiff, Meld, SmartGit...
- IDEs

Example: Merge view in CLion:

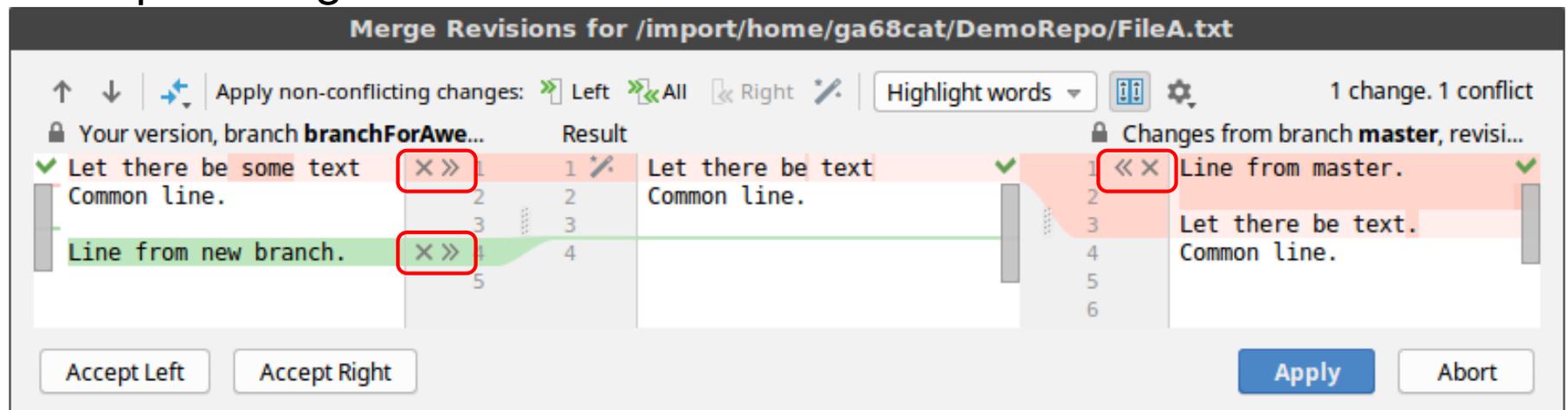
The screenshot shows the 'Merge Revisions for /import/home/ga68cat/DemoRepo/FileA.txt' dialog in CLion. It displays a conflict between two versions of the file. The left pane shows 'Your version, branch branchForAwe...' with a green checkmark on line 1 ('Let there be some text') and a red 'X' on line 4 ('Line from new branch.'). The right pane shows 'Changes from branch master, revisi...' with a red 'X' on line 1 ('Line from master.') and a green checkmark on line 4 ('Common line.'). The 'Result' pane shows the merged state with line 1 from the left and line 4 from the right. The dialog includes navigation arrows, 'Apply non-conflicting changes' options (Left, All, Right), a 'Highlight words' dropdown, and buttons for 'Accept Left', 'Accept Right', 'Apply', and 'Abort'. The status bar indicates '1 change. 1 conflict'.

Resolving conflicts

git mergetool (needs to be configured)

- vimdiff, Meld, SmartGit...
- IDEs

Example: Merge view in CLion:



Resolving conflicts

git mergetool (needs to be configured)

- vimdiff, Meld, SmartGit...
- IDEs

Example: Merge view in CLion:

The screenshot shows the 'Merge Revisions for /import/home/ga68cat/DemoRepo/FileA.txt' dialog in CLion. The dialog is split into three panes: 'Your version, branch branchForAwe...', 'Result', and 'Changes from branch master, revisi...'. The 'Your version' pane shows lines 1-5 with 'Let there be some text' on line 1 and 'Line from new branch.' on line 4. The 'Result' pane shows lines 1-5 with 'Let there be text' on line 1 and 'Line from new branch.' on line 4. The 'Changes from branch master' pane shows lines 1-6 with 'Line from master.' on line 1 and 'Let there be text.' on line 3. The 'Result' pane shows a conflict between the 'Your version' and 'Changes from branch master' panes. The 'Apply non-conflicting changes' section is set to 'Left'. The 'Apply' button is highlighted in blue.

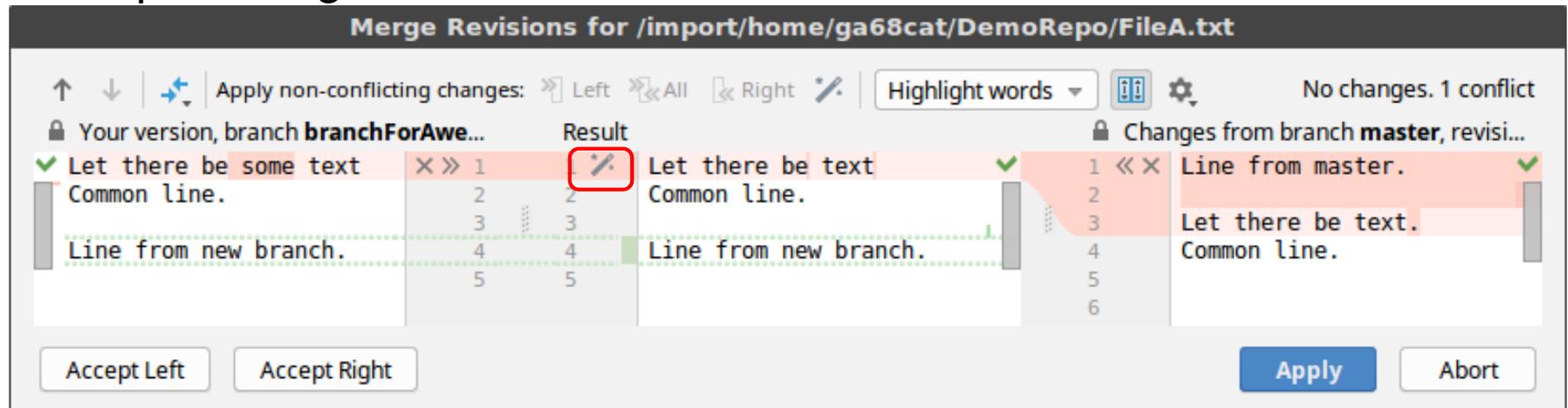
Version	Line	Content
Your version, branch branchForAwe...	1	Let there be some text
Your version, branch branchForAwe...	2	Common line.
Your version, branch branchForAwe...	3	...
Your version, branch branchForAwe...	4	Line from new branch.
Your version, branch branchForAwe...	5	
Result	1	Let there be text
Result	2	Common line.
Result	3	...
Result	4	Line from new branch.
Result	5	
Changes from branch master , revisi...	1	Line from master.
Changes from branch master , revisi...	2	
Changes from branch master , revisi...	3	Let there be text.
Changes from branch master , revisi...	4	Common line.
Changes from branch master , revisi...	5	
Changes from branch master , revisi...	6	

Resolving conflicts

git mergetool (needs to be configured)

- vimdiff, Meld, SmartGit...
- IDEs

Example: Merge view in CLion:



Resolving conflicts

git mergetool (needs to be configured)

- vimdiff, Meld, SmartGit...
- IDEs

Example: Merge view in CLion:

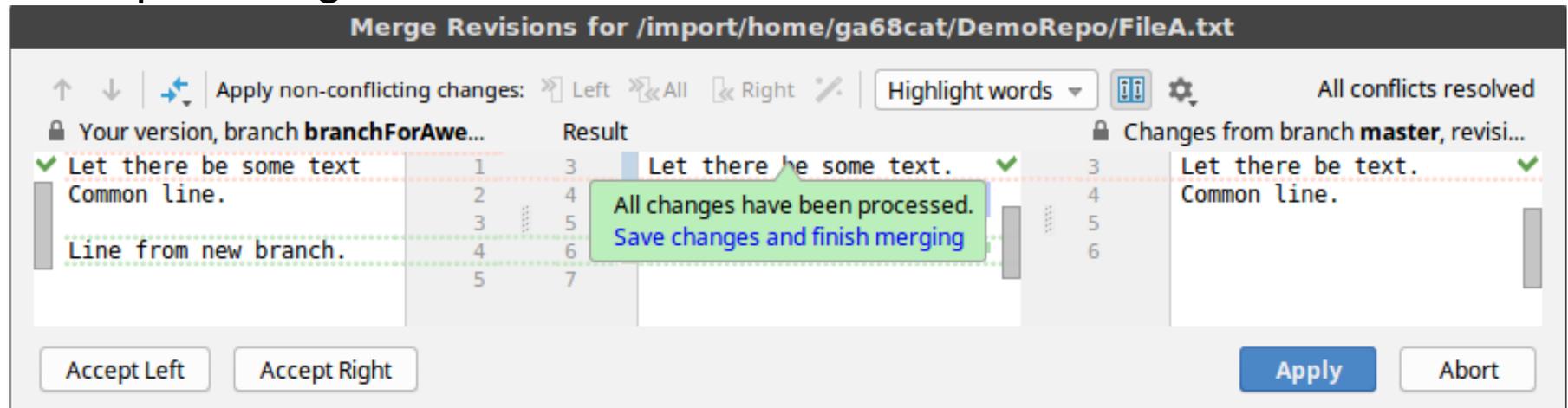
The screenshot shows the 'Merge Revisions for /import/home/ga68cat/DemoRepo/FileA.txt' dialog in CLion. The dialog is divided into three main sections: 'Your version, branch branchForAwe...', 'Result', and 'Changes from branch master, revisi...'. The 'Your version' section shows lines 1-5, with line 4 being 'Line from new branch.'. The 'Result' section shows lines 1-6, with line 4 being 'Common line.'. The 'Changes from branch master' section shows lines 1-6, with line 3 being 'Let there be text.'. The dialog includes a toolbar with 'Apply non-conflicting changes', 'Left', 'All', 'Right', 'Highlight words', and 'Settings' buttons. At the bottom, there are 'Accept Left', 'Accept Right', 'Apply', and 'Abort' buttons. The status 'All conflicts resolved' is displayed in the top right corner.

Resolving conflicts

git mergetool (needs to be configured)

- vimdiff, Meld, SmartGit...
- IDEs
- After resolving all conflicts: **git add** and **git commit**.

Example: Merge view in CLion:



Merging back to master (GitHub)

The screenshot shows a GitHub pull request interface. At the top, the repository is identified as 'FG-TUM / DemoRepo'. The 'Pull requests' tab is selected and highlighted with a red box. The pull request title is 'Branch for awesome feature #2'. A green 'Open' button indicates the pull request is ready for review. The pull request is from the 'branchForAwesomeFeature' branch to the 'master' branch. The interface shows 2 commits, 0 checks, and 1 file changed. A comment from 'FG-TUM' is visible, along with a commit history showing 'Who knows...' and 'merged from master'. On the right side, there are settings for reviewers, assignees, labels, projects, milestones, and notifications. At the bottom, a green box highlights the status: 'Continuous integration has not been set up' and 'This branch has no conflicts with the base branch'. A 'Merge pull request' button is visible at the bottom of this box.

Merging back to master (GitHub)

The screenshot shows a GitHub pull request interface. At the top, the repository is identified as 'FG-TUM / DemoRepo'. The pull request title is 'Branch for awesome feature #2'. The status indicates that 'FG-TUM wants to merge 2 commits into master from branchForAwesomeFeature'. The interface includes navigation tabs for Code, Issues, Pull requests, Projects, Wiki, Insights, and Settings. A comment from 'FG-TUM' is visible, with a 'Reviewers' section highlighted in red, showing 'No reviews'. Below the comment, a commit history shows 'FG-TUM added some commits 3 hours ago' with a 'merged from master' entry. A green box highlights a notification area with two messages: 'Continuous integration has not been set up' and 'This branch has no conflicts with the base branch'. At the bottom, there is a 'Merge pull request' button.

Merging back to master (GitLab)

The screenshot shows the GitLab Merge Request interface for a repository named 'DemoRepo'. The left sidebar contains navigation options: Project, Repository, Issues (1), Merge Requests (1, highlighted with a red box), Wiki, and Settings. The main content area displays a merge request titled 'Branch for awesome feature' with a status of 'Open' and '1 of 2 tasks completed'. It includes a 'Close merge request' button and a 'Request to merge' section with options to 'Open in Web IDE' and 'Check out branch'. Below this, there are 'Merge' and 'Squash commits' options, along with a 'Modify commit message' button. The interface also shows a 'Todo' list on the right side with various settings like 'Assignee', 'Milestone', 'Time tracking', 'Labels', 'Lock merge request', and 'Notifications'. At the bottom, there are statistics for 'Discussion', 'Commits', and 'Changes', and a timestamp '10 Oct, 2018 2 commits'.

Merging back to master (GitLab)

The screenshot shows the GitLab Merge Request interface for a repository named 'DemoRepo'. The main content area displays the merge request details for the 'Branch for awesome feature' branch. It includes a status bar indicating it is 'Open' and was opened 3 minutes ago by 'FG-TUM'. Below this, there are options to 'Edit' or 'Close merge request'. The title of the merge request is 'Branch for awesome feature', and it includes a checklist with two items: 'comment with tickboxes' and 'comments like this directly close #1 (<-Issue)'. The merge request was edited 2 minutes ago by 'FG-TUM'. A 'Request to merge' section shows the source branch 'branchForAwesomeFeature' and the target 'into master', with buttons for 'Open in Web IDE' and 'Check out branch'. A 'Merge' button is highlighted in green, with options to 'Remove source branch' and 'Squash commits'. Below the merge button, it says 'Closes #1' and 'Assign yourself to this issue'. A note states 'You can merge this merge request manually using the command line'. At the bottom, there are icons for thumbs up, thumbs down, and a smiley face, all with a count of 0. The bottom navigation bar shows 'Discussion 0', 'Commits 2', and 'Changes 0'. The left sidebar contains navigation links for 'Project', 'Repository', 'Issues', 'Merge Requests', 'Wiki', and 'Settings'. The right sidebar contains a 'Todo' section with an 'Add todo' button, an 'Assignee' section with 'No assignee - assign yourself' (highlighted with a red box), a 'Milestone' section with 'None', a 'Time tracking' section with 'No estimate or time spent', a 'Labels' section with 'None', a 'Lock merge request' section with 'Unlocked', a '1 participant' section with a user icon, a 'Notifications' section with a toggle switch, and a 'Reference' section with 'ga68cat/DemoRepo!1'.

Summary

- **git pull**
- **git branch** branchForAwesomeFeature
- **git checkout** branchForAwesomeFeature
- Do what you must...
- **git commit**
- **git merge** master
- **git push**
- Create pull request

And anytime you feel lost:

- **git status**

Backup Content

Initialize an existing Repository

git init

- Create a new local repository (.git folder)

git remote add <name> <url>

- Manage remote tracking repositories.

Example: First create repository on GitLab³

```
1 ~/DemoRepo$ git init
2 Initialized empty Git repository in ~/DemoRepo/.git/
3 ~/DemoRepo$ git add . && git commit -m"First commit"
4 ~/DemoRepo$ git remote add origin git@gitlab.lrz.de:ga68cat/DemoRepo.git
5 ~/DemoRepo$ git remote -v
6 origin git@gitlab.lrz.de:ga68cat/DemoRepo.git (fetch)
7 origin git@gitlab.lrz.de:ga68cat/DemoRepo.git (push)
```

³<https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/>

Undoing a Commit

git revert

`[--soft | --hard] [<commit>]`

- Undo commit (not pushed yet!).
- soft: preserve changes.
- hard: revert changes.

`[<tree-ish>] <paths>...`

- tree-ish: can be a commit or relative to HEAD
- Unstaging single files (not committed yet!).

Example:

```

1 ~/DemoRepo$ git reset --soft HEAD~1
2 ~/DemoRepo$ git status
3 On branch master
4 Your branch is up-to-date with 'origin/master'.
5 Changes to be committed:
6   new file:   FileA.txt
7   modified:  README.md
8 ~/DemoRepo$ git reset HEAD README.md
9 Unstaged changes after reset:
10 M README.md

```

Who did that!?

git blame [-w] <File>

- Show author of last line change.
- -w Exclude whitespace-only changes

Example: (Suppose there is a second author)

```
1 ~/AutoPas$ git blame README.md
2 f5344b80 README.md (FG-TUM 2018-07-31 15:53:56 +0200 1) # AutoPas
3 e46ec677 README.md (Seckler 2018-04-13 17:03:54 +0900 2) AutoPas is a ...
```

CLion: Annotations

```
AutoPas > README.md >
README.md x
1 4/13/18 Seckler # AutoPas
2 4/13/18 Seckler AutoPas is a node-level auto-tuned particle simulation library developed
3 4/24/18 Sprinz in the context of the **TaLPas** project. [Build Status](https://www5.in.tum.de/jenk:
```

Put stuff aside

git stash [save <message> | list | pop [<stash>]]

- Save all current changes with a message.
- List all stashes.
- Apply a stash and remove it from the list.

Example:

```
1 ~/DemoRepo$ git checkout branchForAwesomeFeature
2 error: Your local changes to the following files would be overwritten by checkout:
3     README.md
4 Please commit your changes or stash them before you switch branches.
5 Aborting
6 ~/DemoRepo$ git stash save "my message"
7 Saved working directory and index state On master: my message
8 ~/DemoRepo$ git stash list
9 stash@{0}: On master: my message
10 ~/DemoRepo$ git stash pop
11 Changes not staged for commit:
12     README.md
```