

SU Development Forum

Introduction to Git - Save your projects!



Auteur: Rémi Ducceschi
Supervisor: Francis Klumb

2018-01-26

Git – Distributed Version Control System

What is it?

- Created by Linus Torvalds in 2005
 - For Linux kernel
- Tool – to be integrated in your workflow
- Manage your project files
 - Save them on a server (security)
 - Keep a track of the modifications (history)
- Ease team working



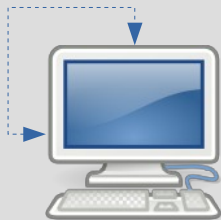
Git – Distributed Version Control System

Distributed?

- Distributed: main difference with SVN
 - No need for server

Remote

Local

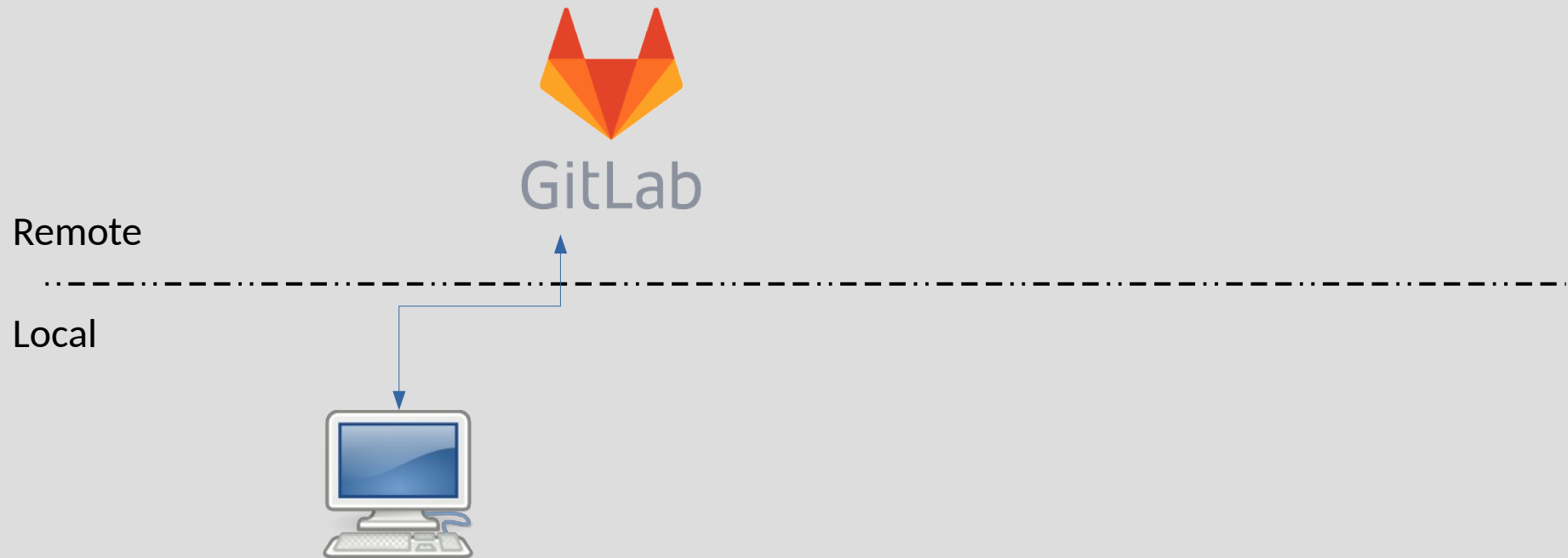


WARNING: No data duplication !

Git – Distributed Version Control System

Distributed?

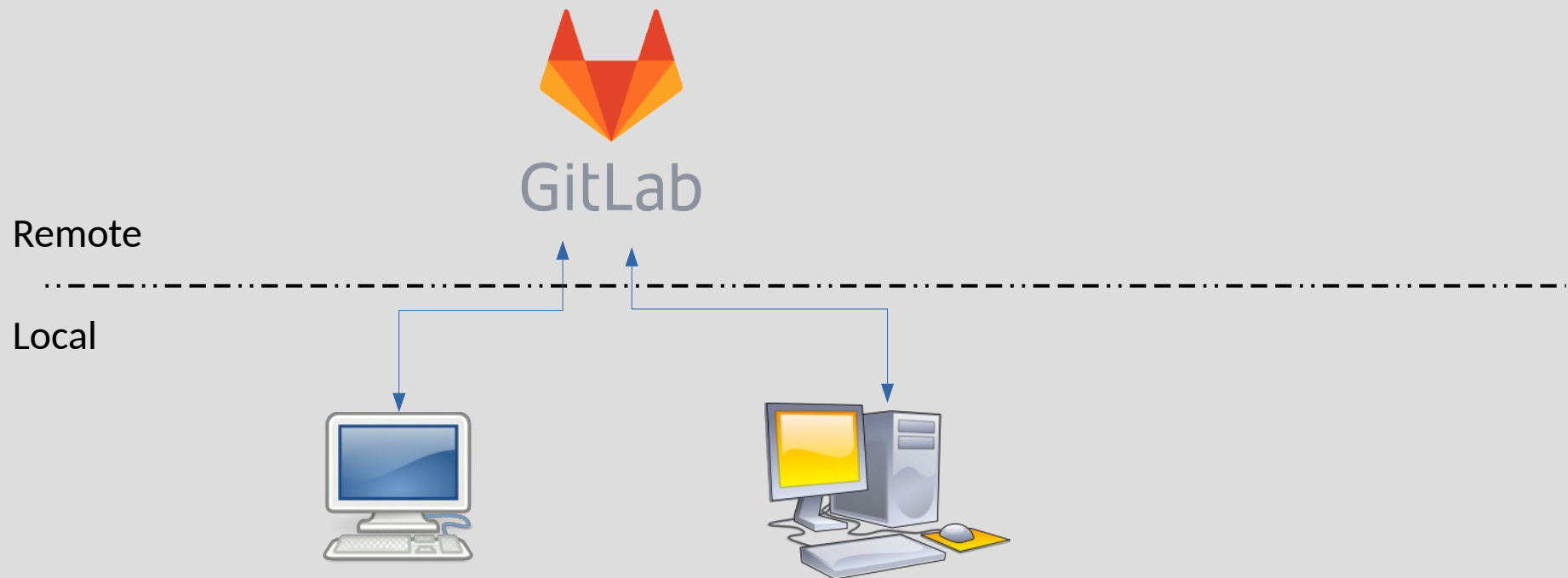
- Distributed: main difference with SVN
 - SVN style – mono user



Git – Distributed Version Control System

Distributed?

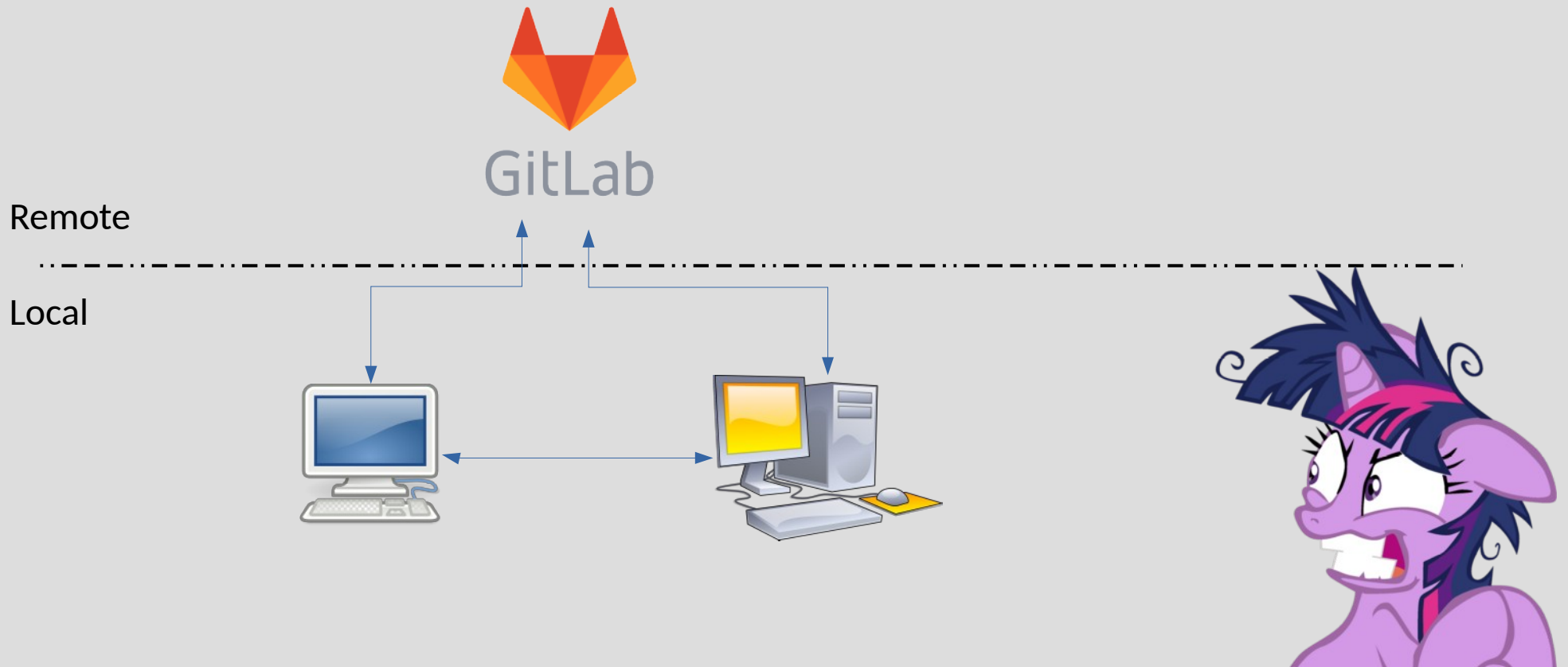
- Distributed: main difference with SVN
 - SVN style – multi user



Git – Distributed Version Control System

Distributed?

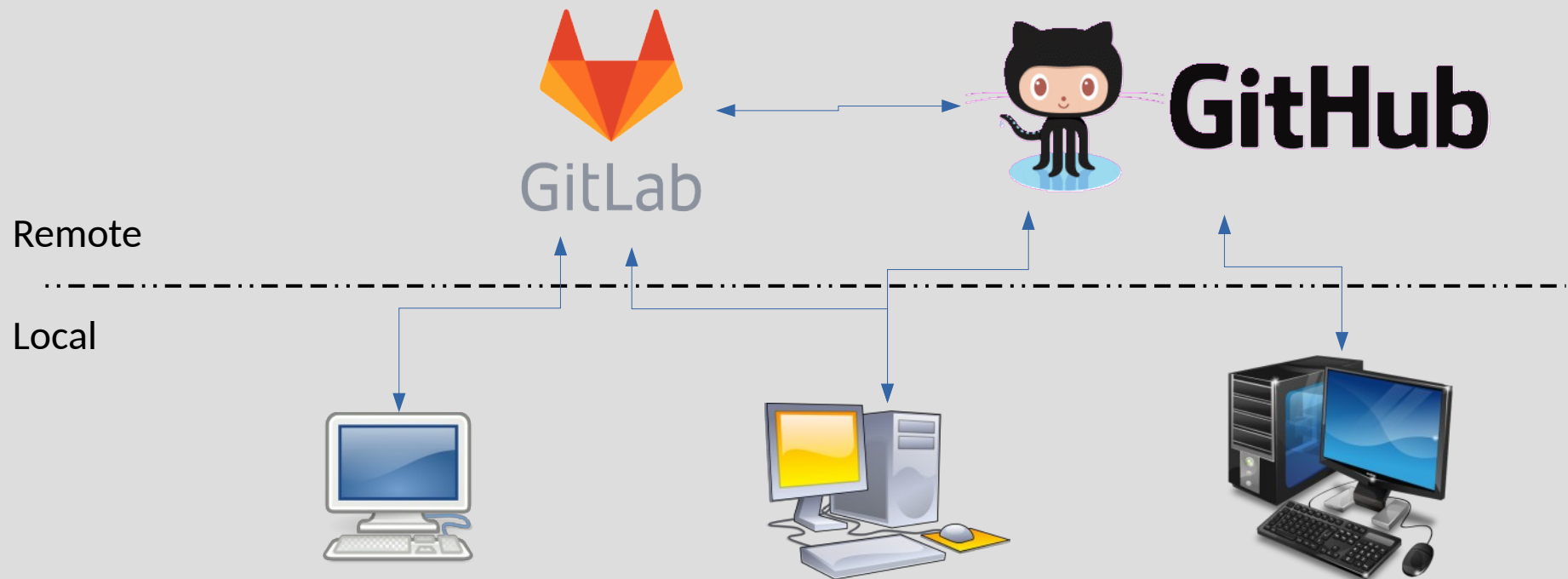
- Distributed: main difference with SVN
 - Distributed: peer to peer



Git – Distributed Version Control System

Distributed?

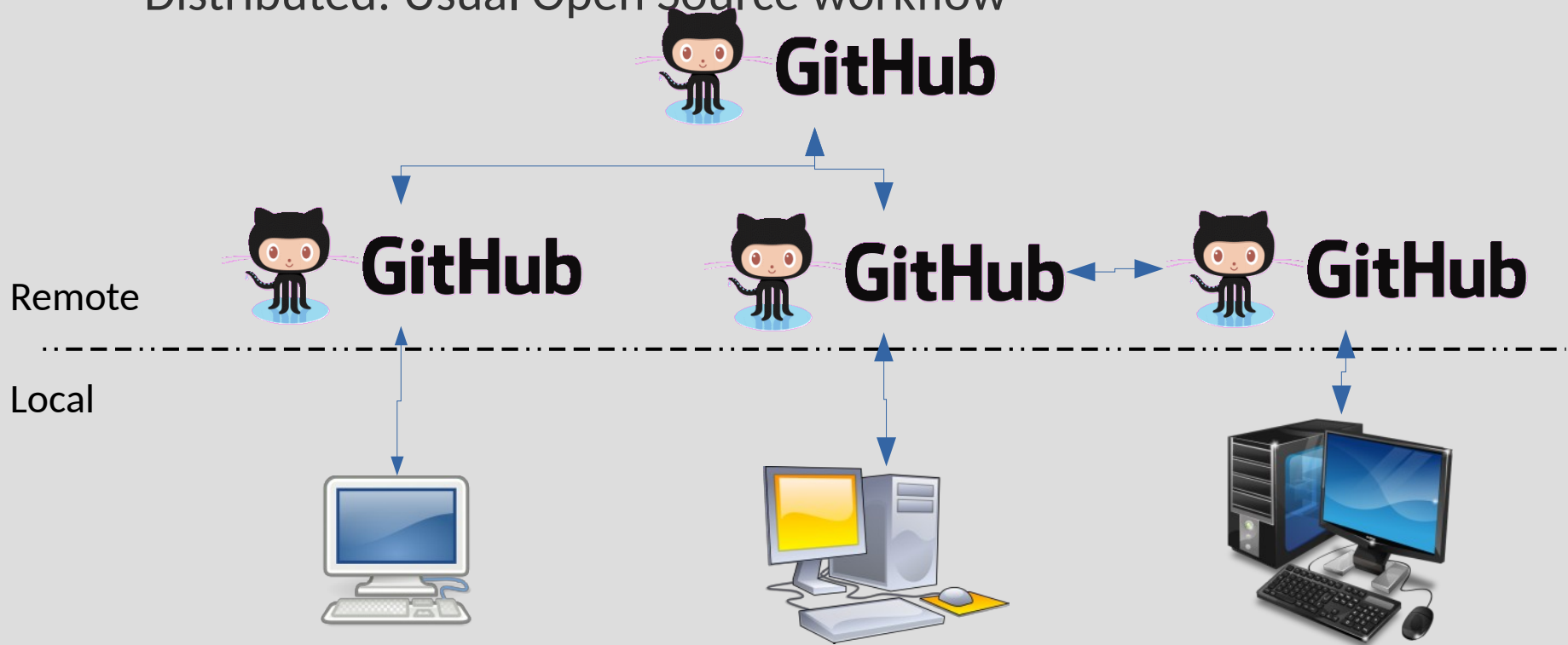
- Distributed: main difference with SVN
 - Distributed: multi remote



Git – Distributed Version Control System

Distributed?

- Distributed: main difference with SVN
 - Distributed: Usual Open Source workflow



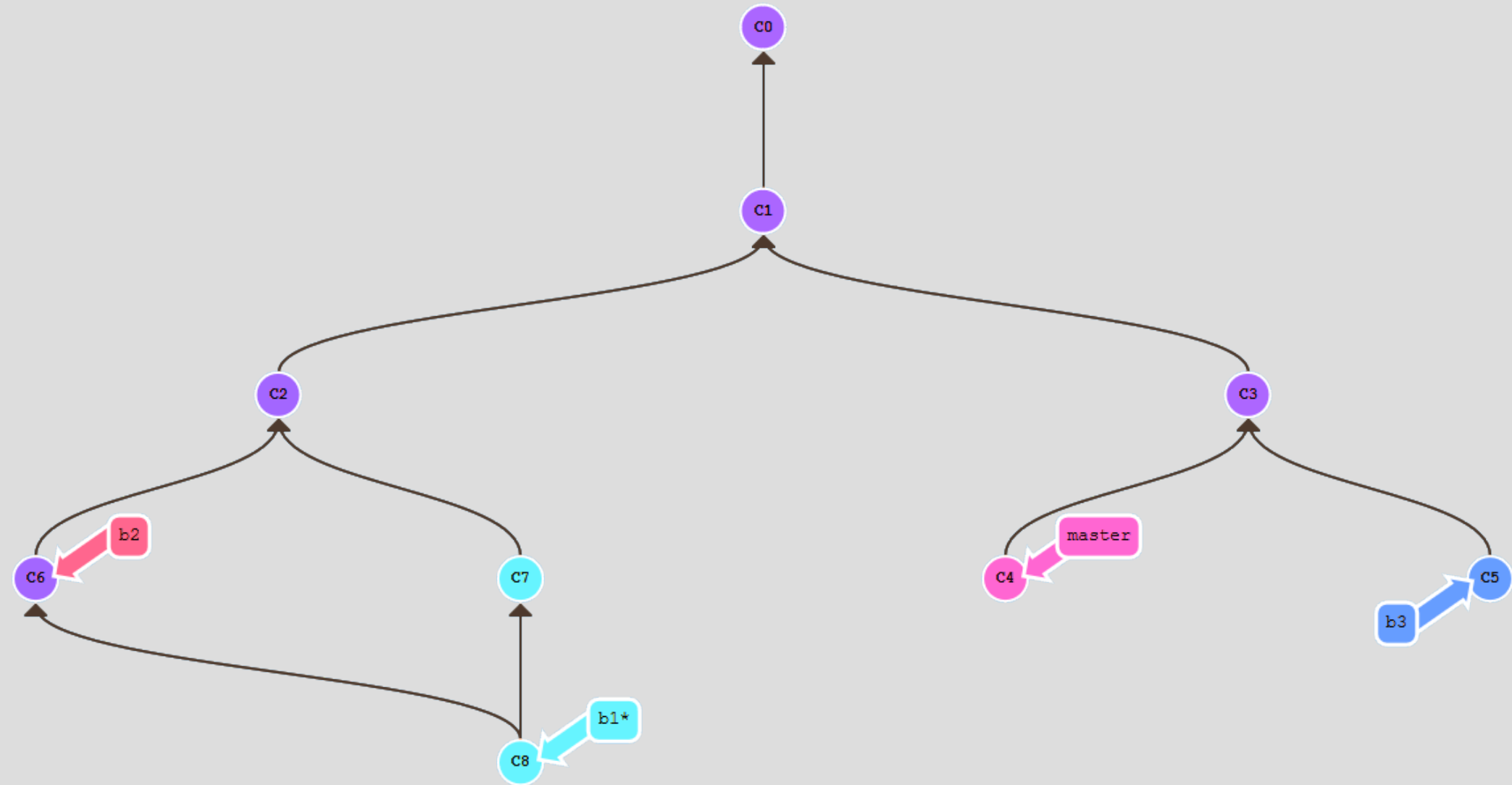
Git – Concepts

Commits

- What is a *commit*?
 - Contains the state of all the files at one moment
 - Contains a message explaining the reasons of the changes
 - Contains the the whole history that leads to it
- Commits are organized in a graph
 - 1 or more parents (history)
 - 1 or more children (future)
- Commits have name
 - A hash: 76e22ab887fe24de6a7b3d19809af392de3036e6
 - Will be simplified in this presentation

Git – Concepts

Graph



Git – Concepts

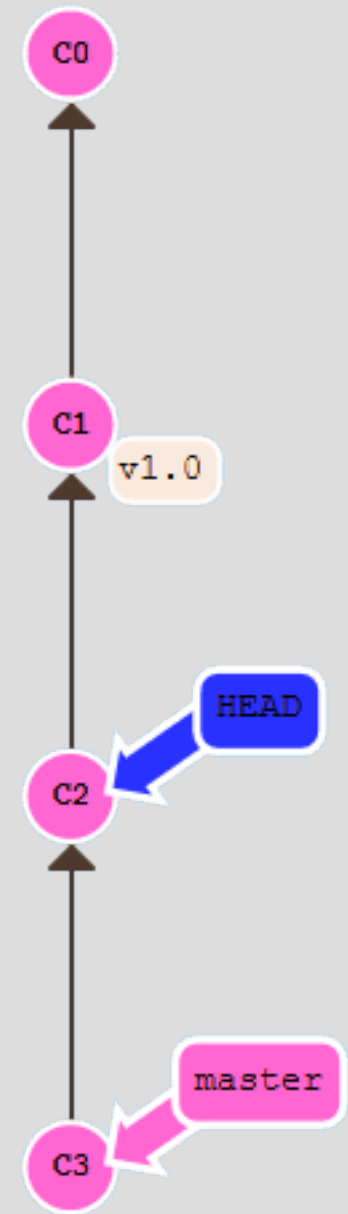
References

- How to organize the commits?
 - Use references:
 - Tag → direct pointer to a commit (label)
 - Branch → dynamic pointer
 - Points to one commit
 - Moves to the most recent commit dynamically
 - Default branch = `master`
 - `HEAD` → current position in the tree

Git – Concepts

References

- How to move in the graph?
 - Git checkout:
 - to commit name (ad65ert5gd12)
 - to relative position (HEAD~2)
 - to tag
 - to branch



Git – Setup your environment

Install Git

- Windows
 - Install Git for Windows
 - <https://git-scm.com/downloads>
 - Do not use the CMF version: not up to date
 - Min version to install: 2.16
 - Some UI exists (Tortoise Git, Github Desktop)
 - CLI is easy to learn
 - See the CheatSheet → 5 commands for basics
 - CLI is way more powerful
 - You will need CLI at one point anyway
- Linux: `yum install git`

Git – Setup your environment

Basics

- Tell git who you are
 - `git config --global user.name "Rémi Ducceschi"`
 - `git config --global user.email "remi.ducceschi@cern.ch"`
- Change default editor (nano)
 - `git config --global core.editor "C:/Program\ Files/Notepad+/notepad++.exe -multiInst -notabbar -nosession -noPlugin"`
- Colors (on by default)
 - `git config --global color.ui true`
- Stored in `~/.gitconfig`
 - Can be overwritten in project by adding a `.gitconfig`

Git[lab] – Workflow

From editing, publishing to integrating the code

- Create a branch
 - One branch per feature
- Edit files
- Commit your changes
 - Do it several times
 - To save your work (end of the day, lunch time...)
 - To separate minor states of development
- Push to remote
- Create a Pull Request (PR) / Merge Request

Git[lab] – Workflow

From editing, publishing to integrating the code

- Create a branch
 - One branch per feature
- Edit files
- Commit your changes
 - Do it several times
 - To save your work (end of the day, lunch time...)
 - To separate minor states of development
- Push to remote
- Create a Pull Request (PR) / Merge Request

Git[lab] – Workflow

Create a branch

- Checkout on master and update it
 - `git checkout master`
 - `git pull`
- Create your branch and checkout on it
 - `git checkout -b my_branch`
- You can push your branch now
 - Even without any modifications
 - `git push --set-upstream origin my_branch`

Git[lab] – Workflow

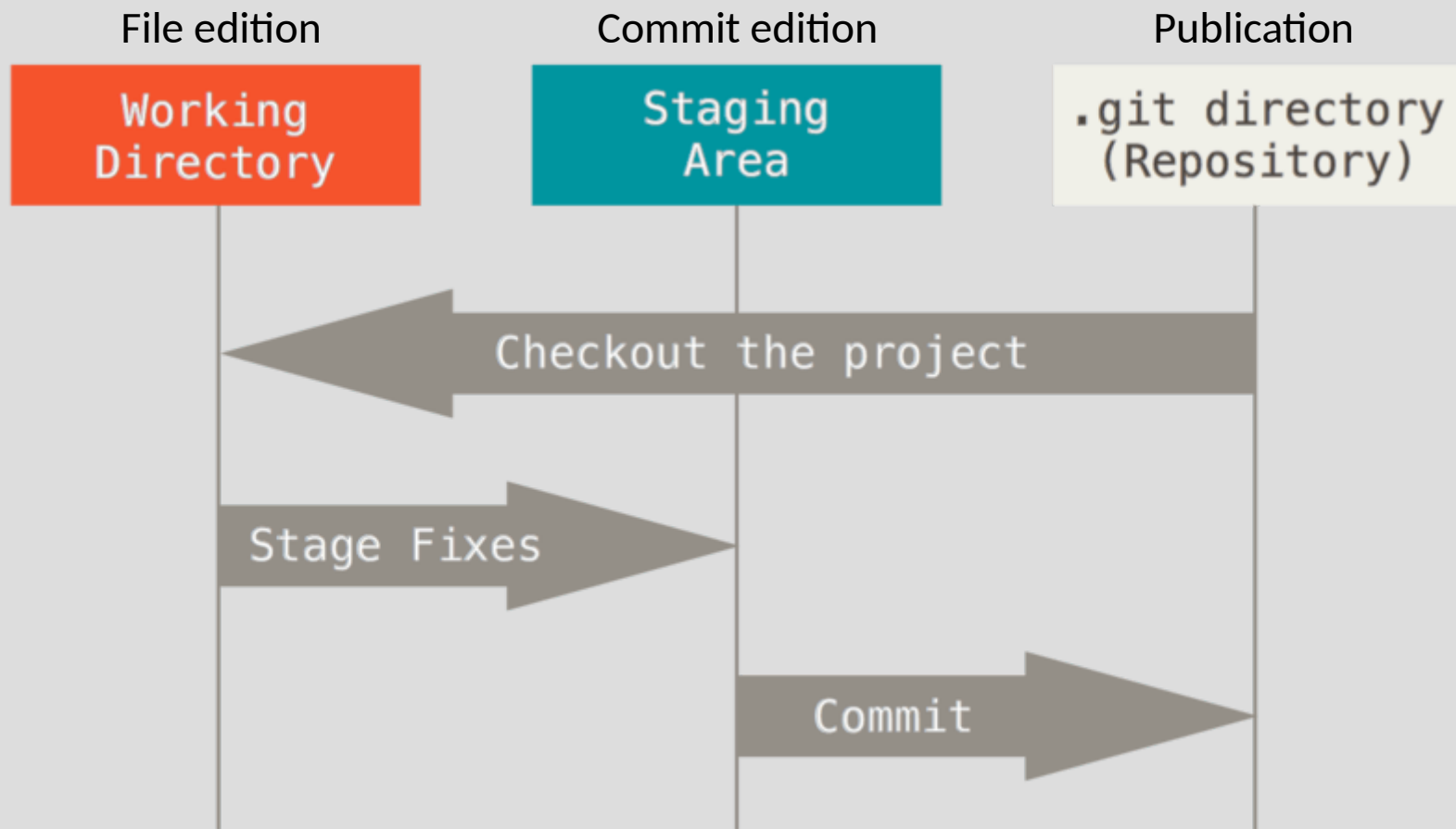
From editing, publishing to integrating the code

- Create a branch
 - One branch per feature
- Edit files
- Commit your changes
 - Do it several times
 - To save your work (end of the day, lunch time...)
 - To separate minor states of development
- Push to remote
- Create a Pull Request (PR) / Merge Request

Git[lab] – Workflow

How to commit

- A bit of theory:
 - 3 states



Git[lab] – Workflow

How to commit

- Edit files
 - `git status` to see changes
- Add the changes
 - `git add file1 file2...`
- Commit
 - `git commit`

Git[lab] – Workflow

From editing, publishing to integrating the code

- Create a branch
 - One branch per feature
- Edit files
- Commit your changes
 - Do it several times
 - To save your work (end of the day, lunch time...)
 - To separate minor states of development
- Push to remote
- Create a Pull Request (PR) / Merge Request

Git[lab] – Workflow

How to publish

- First we update the branch
 - `git pull`
- Then we push
 - `git push`
 - `git push --set-upstream origin my_branch`
 - Needed if the branch has never been pushed

Git[lab] – Workflow

From editing, publishing to integrating the code

- Create a branch
 - One branch per feature
- Edit files
- Commit your changes
 - Do it several times
 - To save your work (end of the day, lunch time...)
 - To separate minor states of development
- Push to remote
- Create a Pull Request (PR) / Merge Request



Git[lab] – Workflow

How to Pull request (PR)

Remi Ducceschi > tmpshowcase > Repository > **Branches**

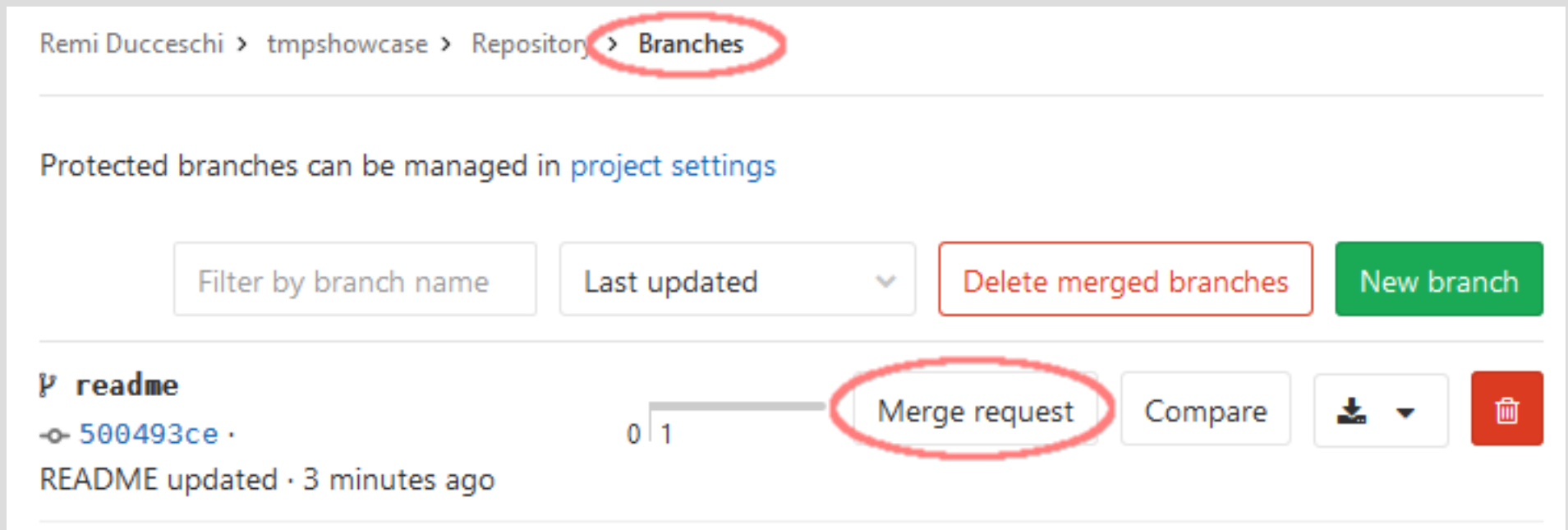
Protected branches can be managed in [project settings](#)

Filter by branch name Last updated **Delete merged branches** **New branch**

readme **Merge request** Compare  

500493ce ·
README updated · 3 minutes ago

0 | 1



Git[lab] – Workflow

How to Pull request (PR)

Title README updated

Start the title with **WIP:** to prevent a **Work In Progress** merge request from being merged before it's ready.

Add [description templates](#) to help your contributors communicate effectively!

Description

Write Preview **B I ” </> ☰ ☷ ☑ ✕**

Write a comment or drag your files here...

Markdown and [quick actions](#) are supported [Attach a file](#)

Source branch readme

Target branch master [Change branches](#)

Remove source branch when merge request is accepted.

Squash commits when merge request is accepted. [About this feature](#)

Git[lab] – Workflow

How to Pull request (PR)

Discussion 0 Commits 1 **Changes 1**

Showing 1 changed file ▾ with 5 additions and 0 deletions

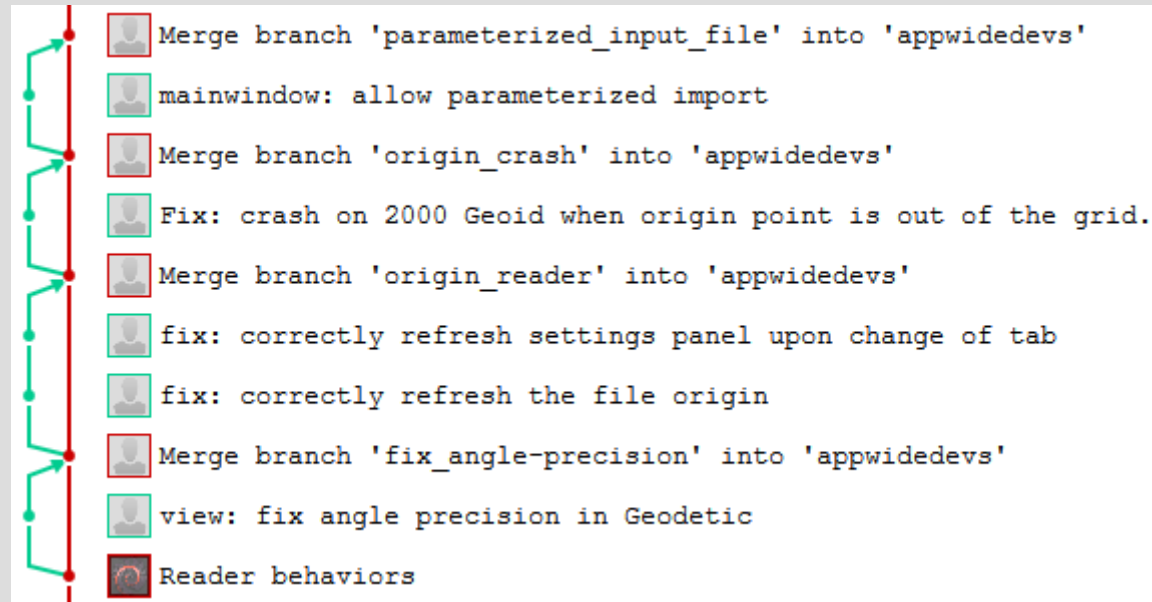
▾ README.md   Edit View file @ 500493ce

1	This is an example and temporary repository for https://indico.cern.ch/event/697540/	1 + TMPShowCase
		2 + =====
		3 +
		4 This is an example and temporary repository for https://indico.cern.ch/event/697540/
		5 +
		6 + The goal is to show how to do basic git commands, and create a PR.

Git[lab] – Workflow

How to Pull request (PR)

- With good repository setup
 - Click merge when ready!



Git - Conclusion

- I didn't talk about merge
 - You shouldn't merge by hand!
 - Only merge via Pull Requests
 - Except rare occasion
 - Rebase only!
- Advanced git possibilities
 - Rewrite history
 - Reflogs
 - ...

Git – External links

Very useful!

- Learn git from basics to expert
 - <https://learngitbranching.js.org/>
- Customize your environment with aliases
 - <https://git-scm.com/book/tr/v2/Git-Basics-Git-Aliases>

Questions

