git

What is a version control system?

What is a version control system?

Tracking and managing changes in files

- Compare earlier versions
- Recover a previous version
- Protect against catastrophe or human error
- Work with different versions in same time
- Collaborate on the same project at the same time
- Managing conflict between concurrent work

Do you need a version control system for your work?

What is Git?

Distributed Version Control System

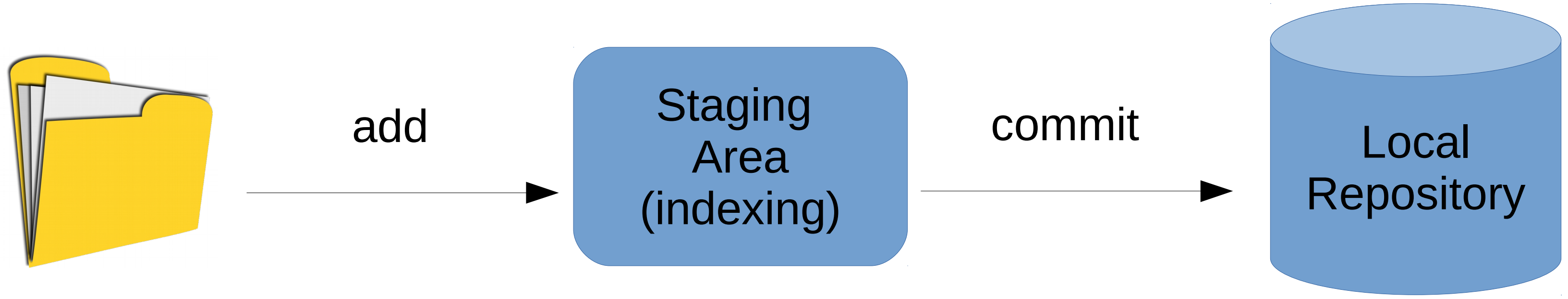
- A complete long-term change history of every file
- Branching and merging
- Traceability

Designed for **text files**, not for **real time** collaboration and not an automatic versioning system

Git for individual work

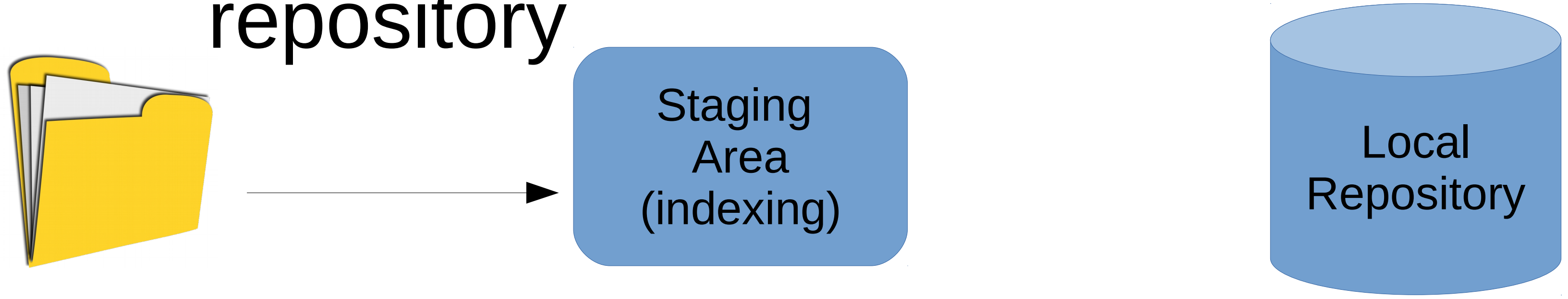
Initialize a Git repository

- git init Create an empty repository
- git add Add the file to track
- git commit Record changes to the repository



Record modifications

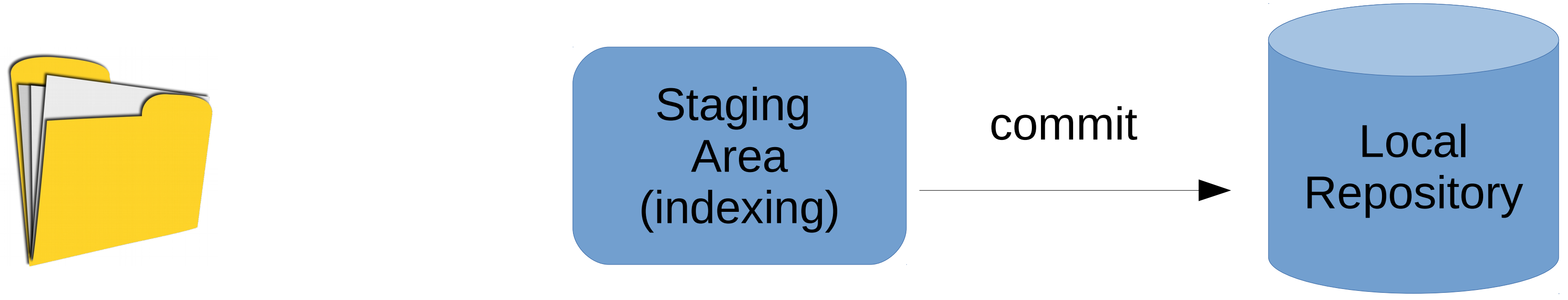
- git add   Index file or modification of file
- git rm   Index the deletion of file
- git mv   Index the displacement of file in a repository



Commit

When is a good time to commit?

What is a good commit message?



When is a good time to commit?

- Commit early and often
- Keep commits focused
- Consistency

What is a good commit message?

A commit message is a letter to yourself in the future. It’s composed of a title (72 characters) and a description.

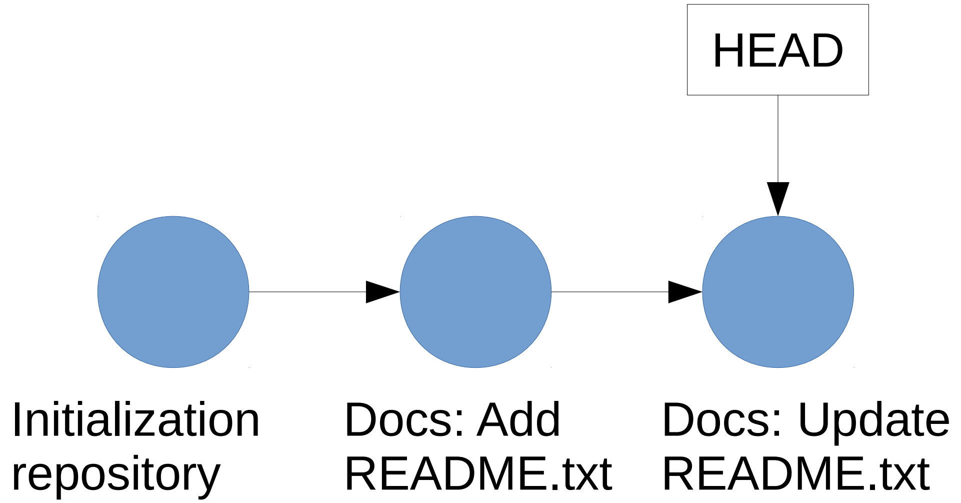
- Short and direct title
- Use imperative mood in the subject line.
- Precise the type of commit

The message should answer the questions:
what and why are you committing?

Example: Docs:Update README

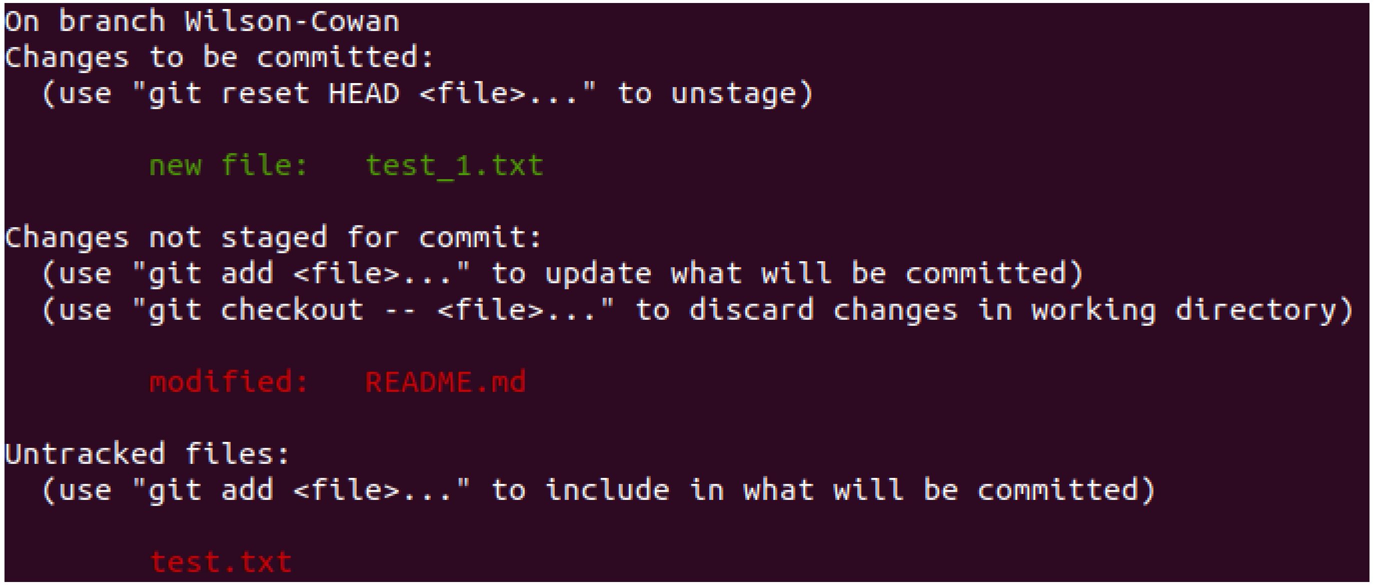
History

- git log	show the history of commit
- git show show a commit



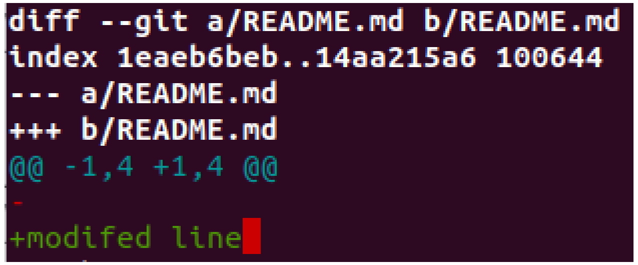
Look at your repository

- git status



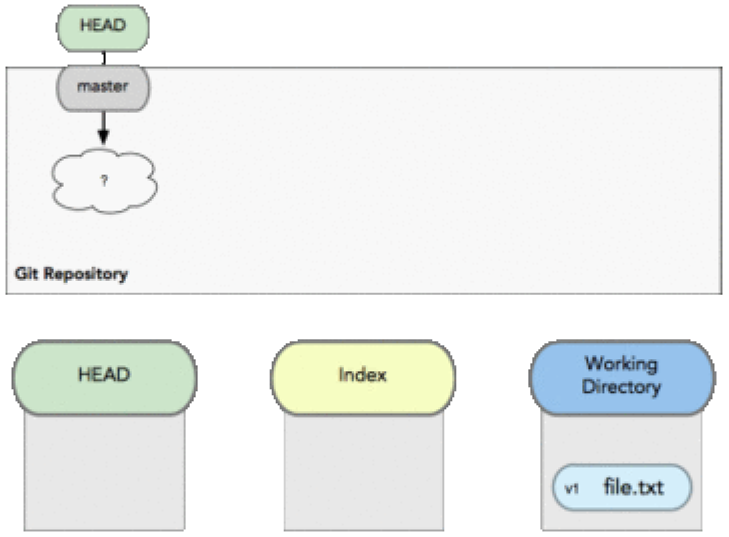
```
On branch Wilson-Cowan
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   test_1.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

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

        test.txt
```

- git diff



```
diff --git a/README.md b/README.md
index 1eaeb6beb..14aa215a6 100644
--- a/README.md
+++ b/README.md
@@ -1,4 +1,4 @@
-
+modifed line
```

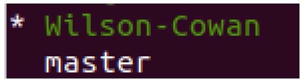
Summary


HEAD
master
?
Git Repository
HEAD
Index
Working Directory
v1
file.txt

Git for team work

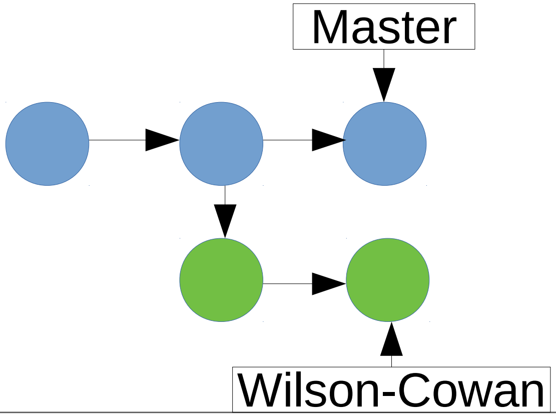
Usage of Branch

- git branch: list and manage branches



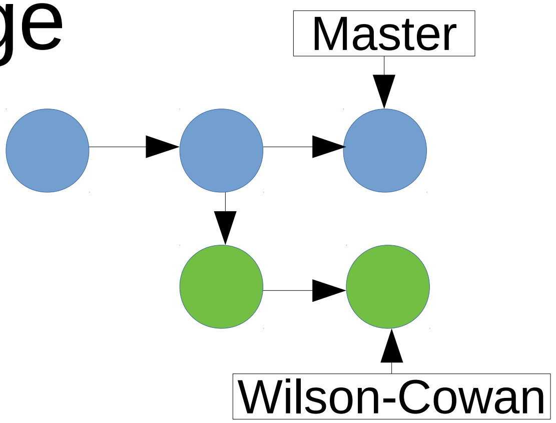
```
* Wilson-Cowan
  master
```

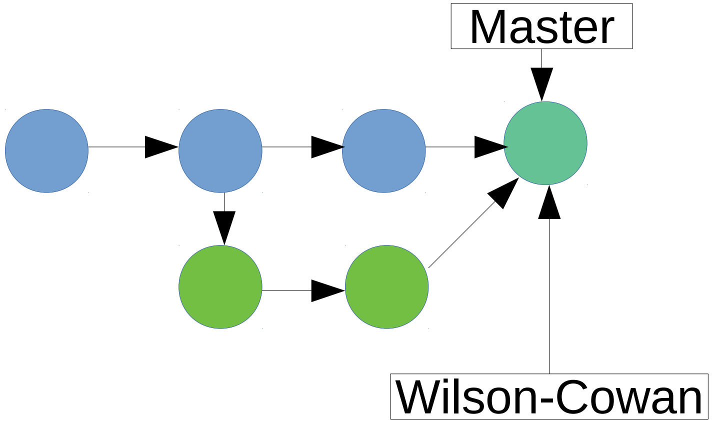
- git checkout: moving between commits
  -b : move to a commit and create a branch



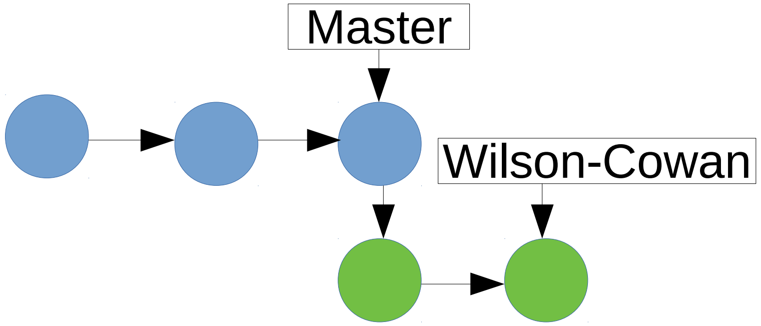
Update Master

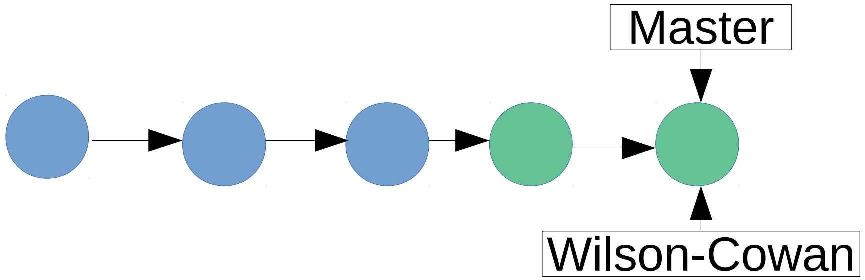
- git merge



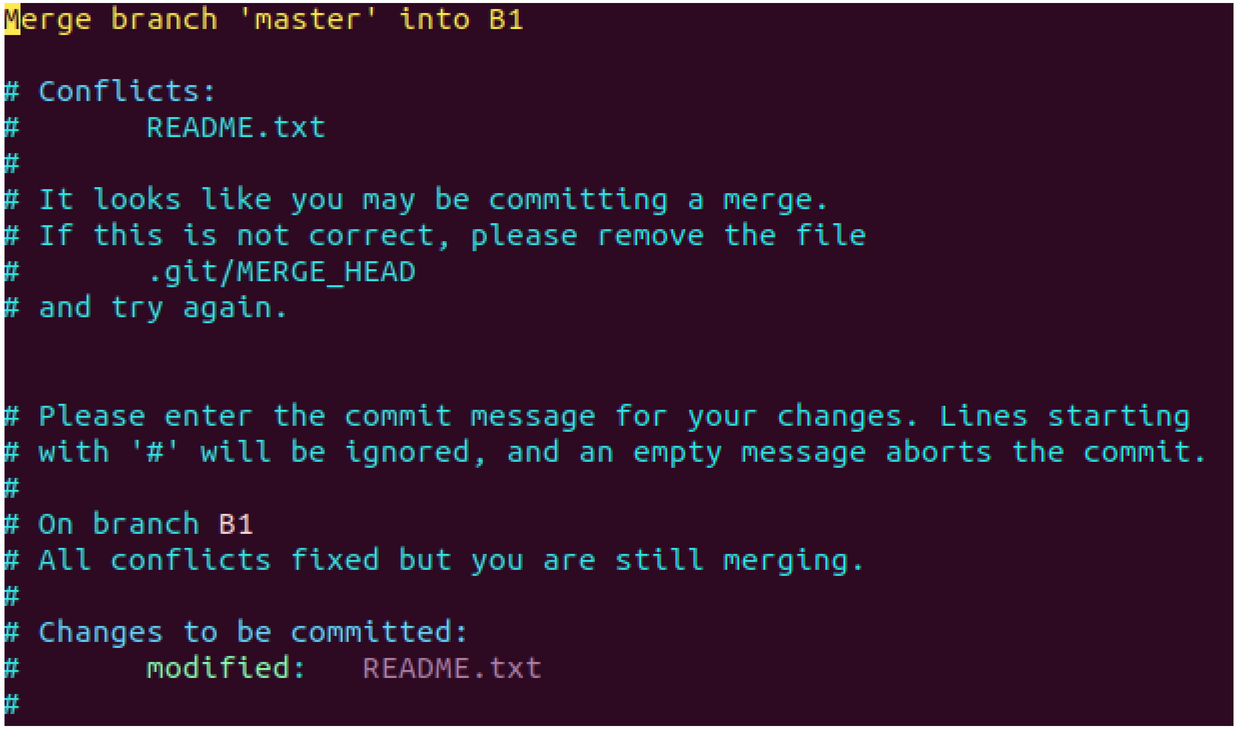


Fast forward





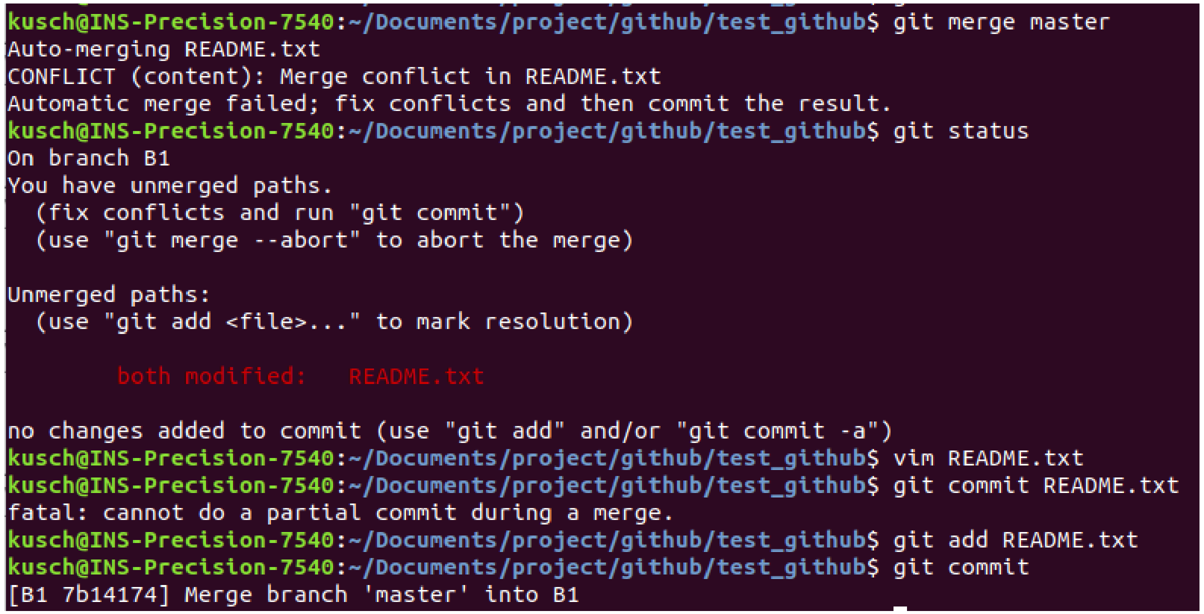
Automatic commit



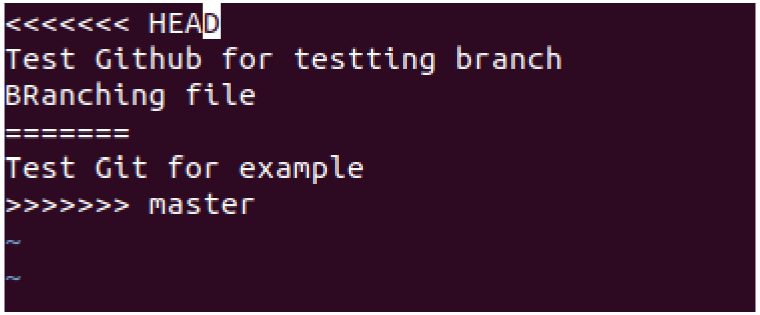
```
Merge branch 'master' into B1

# Conflicts:
#       README.txt
#
# It looks like you may be committing a merge.
# If this is not correct, please remove the file
#       .git/MERGE_HEAD
# and try again.


# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch B1
# All conflicts fixed but you are still merging.
#
# Changes to be committed:
#       modified:   README.txt
#
```

Institut de Neurosciences des Systèmes

Merge with a conflict



```
kusch@INS-Precision-7540:~/Documents/project/github/test_github$ git merge master
Auto-merging README.txt
CONFLICT (content): Merge conflict in README.txt
Automatic merge failed; fix conflicts and then commit the result.
kusch@INS-Precision-7540:~/Documents/project/github/test_github$ git status
On branch B1
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   README.txt

no changes added to commit (use "git add" and/or "git commit -a")
kusch@INS-Precision-7540:~/Documents/project/github/test_github$ vim README.txt
kusch@INS-Precision-7540:~/Documents/project/github/test_github$ git commit README.txt
fatal: cannot do a partial commit during a merge.
kusch@INS-Precision-7540:~/Documents/project/github/test_github$ git add README.txt
kusch@INS-Precision-7540:~/Documents/project/github/test_github$ git commit
[B1 7b14174] Merge branch 'master' into B1
```



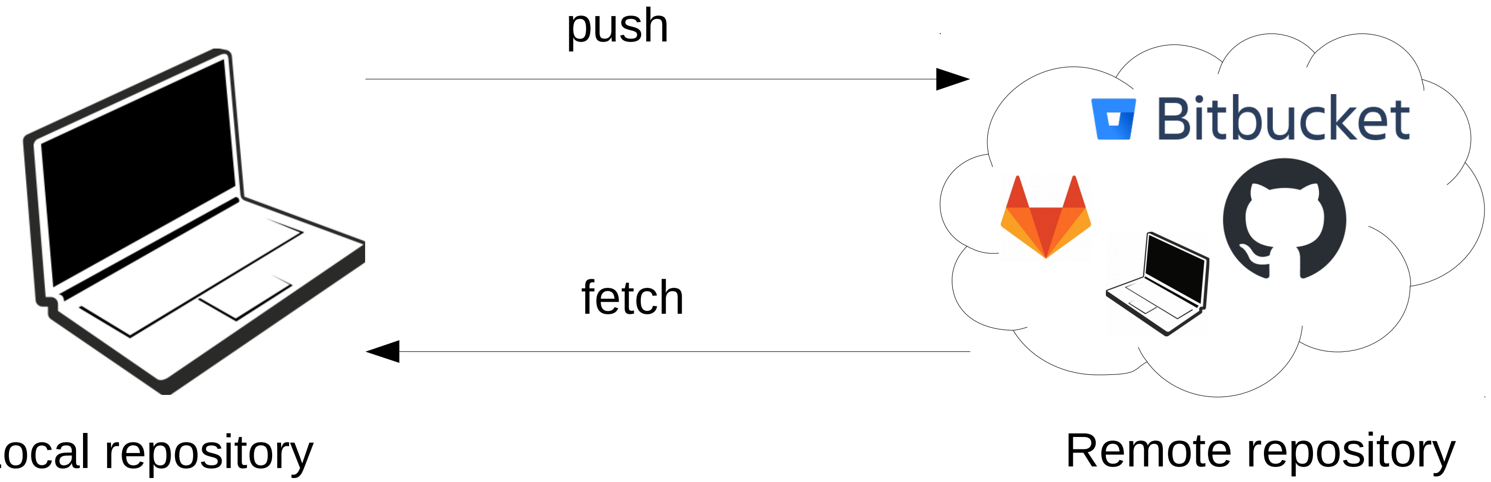
```
<<<<<<< HEAD
Test Github for testting branch
BRanching file
=======
Test Git for example
>>>>>>> master
~
~
```

Managing conflicts also requires communication between people.

Institut de Neurosciences des Systèmes

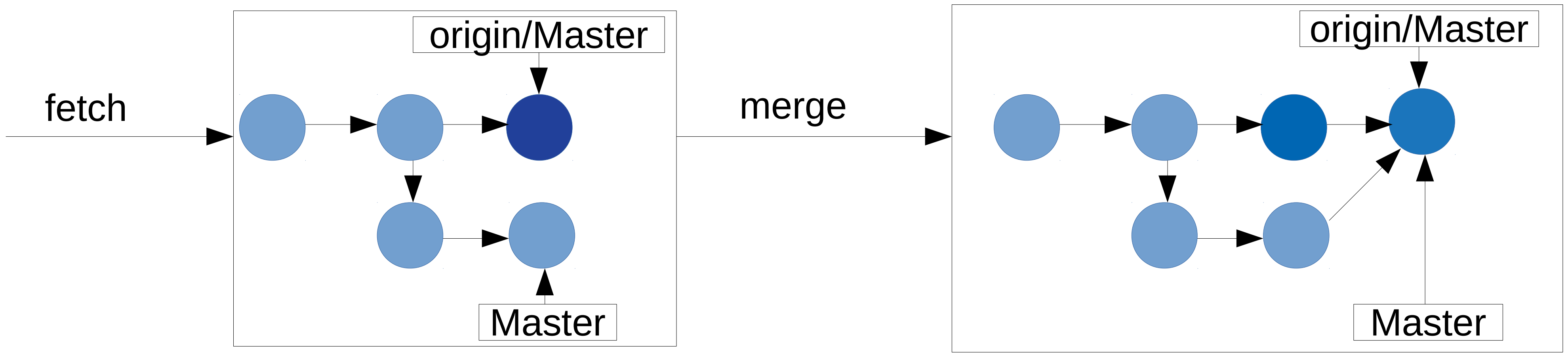
Sharing work or Saving it

- git fetch: get commits from a server
- git push: push commits to a server



Sharing work or Saving it

- git pull: it’s a combination of two commands



Collaboration to open source project

- git clone: copy a project
- **Issue**: track ideas, feedback, tasks, or bugs for a project
- **Fork**: a copy of a repository
- **Pull request**: propose and review changes
- **Continuous integration**: test, code styling ...

Summary

- git clone
- git pull
- git push

Saving regularly to keep a backup in the case of a problem on your computer

Example and Advice of Git usage

Advices

- Don’t share new versions of files outside of git. Use git in order to keep track of the author and the modifications
- Don’t track big file (some servers limit the size of files)

List of main git command

git add
git am
git archive
git bisect
git branch
git bundle
git checkout
git cherry-pick
git citool
git clean

git clone
git commit
git describe
git diff
git fetch
git format-patch
git gc
git grep
git gui
git init
git log
git maintenance

git merge
git mv
git notes
git pull
git push
git range-diff
git rebase
git reset
git restore
git revert
git rm
git shortlog

git show
git sparse-checkout
git stash
git status
git submodule
git switch
git tag
git worktree

Example

- https://github.com/git/git
- https://github.com/the-virtual-brain/tvb-root/actions

- Find Ebrains git server : https://gitlab.ebrains.eu/

Git Cheat Sheet

REBELLABS by ZEROTURNAROUND

Create a Repository

From scratch -- Create a new local repository
```
$ git init [project name]
```

Download from an existing repository
```
$ git clone my_url
```

Observe your Repository

List new or modified files not yet committed
```
$ git status
```

Show the changes to files not yet staged
```
$ git diff
```

Show the changes to staged files
```
$ git diff --cached
```

Show all staged and unstaged file changes
```
$ git diff HEAD
```

Show the changes between two commit ids
```
$ git diff commit1 commit2
```

List the change dates and authors for a file
```
$ git blame [file]
```

Show the file changes for a commit id and/or file
```
$ git show [commit]:[file]
```

Show full change history
```
$ git log
```

Show change history for file/directory including diffs
```
$ git log -p [file/directory]
```

Working with Branches

List all local branches
```
$ git branch
```

List all branches, local and remote
```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory
```
$ git checkout my_branch
```

Create a new branch called new_branch
```
$ git branch new_branch
```

Delete the branch called my_branch
```
$ git branch -d my_branch
```

Merge branch_a into branch_b
```
$ git checkout branch_b
$ git merge branch_a
```

Tag the current commit
```
$ git tag my_tag
```

Make a change

Stages the file, ready for commit
```
$ git add [file]
```

Stage all changed files, ready for commit
```
$ git add .
```

Commit all staged files to versioned history
```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history
```
$git commit -am "commit message"
```

Unstages file, keeping the file changes
```
$ git reset [file]
```

Revert everything to the last commit
```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)
```
$ git fetch
```

Fetch the latest changes from origin and merge
```
$ git pull
```

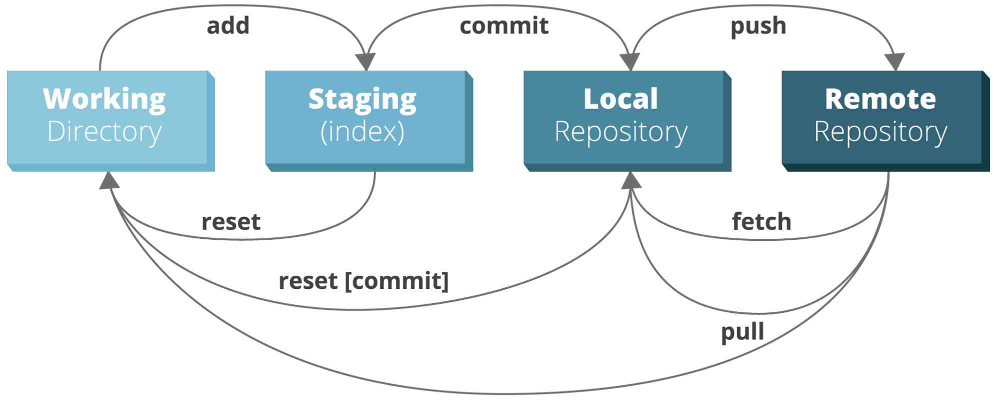
Fetch the latest changes from origin and rebase
```
$ git pull --rebase
```

Push local changes to the origin
```
$ git push
```

Finally!

When in doubt, use git help
```
$ git command --help
```

Or visit https://training.github.com/ for official GitHub training.



source

- https://www.freecodecamp.org/news/how-to-write-better-git-commit-messages/
- https://git-scm.com/
- https://www.atlassian.com/git/tutorials/what-is-version-control
- https://xosh.org/explain-git-in-simple-words/
- https://medium.com/upperlinecode/how-to-teach-git-commits-github-to-teenagers-a3f740b2f500
- https://rachelcarmena.github.io/2018/12/12/how-to-teach-git.html

Exercise

- https://github.com/lionelkusch/INS_presentation
- https://gitlab.ebrains.eu/kuschlionel/ins_presentation