Introduction to Git

Alessandro Sciarra

Z02 – Software Development Center

25 February 2022

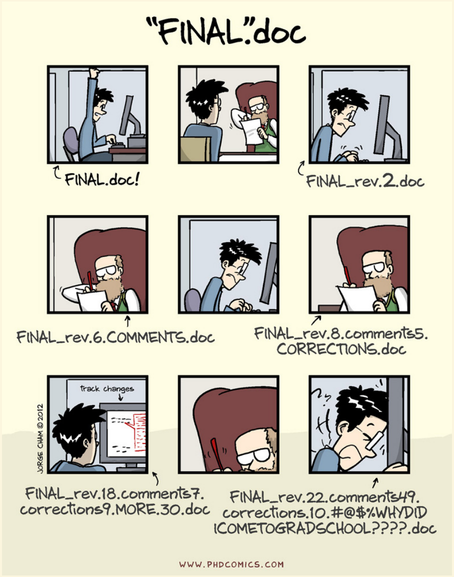
Outline of the talk

Why should you use it?

OK, let’s do it without git

A possible workflow

Writing a review or a Ph.D. thesis


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.CORRECTIONS.doc
track changes
JORGE CHAM © 2012
FINAL_rev.18.comments7.corrections9.MORE.30.doc
FINAL_rev.22.comments49.corrections.10.#@$%WHYDID ICOMETOGRADSCHOOL????.doc
WWW.PHDCOMICS.COM

A possible workflow

Writing a review or a Ph.D. thesis

- How do you make writing experiments?

A possible workflow

Writing a review or a Ph.D. thesis

- How do you make writing experiments?
  - You make a backup of your file
  - You comment out a block of text in your source
  - If the old version was better, you restore it by hand
  - If the new version is better, you clean up by hand

A possible workflow

Writing a review or a Ph.D. thesis

- How do you make writing experiments?
- How do you create/view checkpoints?

A possible workflow

Writing a review or a Ph.D. thesis

- How do you make writing experiments?
- How do you create/view checkpoints?
  - Create a .tar or .zip file
  - Copy it somewhere and uncompress if needed

A possible workflow

Writing a review or a Ph.D. thesis

- How do you make writing experiments?
- How do you create/view checkpoints?
- Which version did you send to your supervisor/colleagues?

A possible workflow

Writing a review or a Ph.D. thesis

- How do you make writing experiments?
- How do you create/view checkpoints?
- Which version did you send to your supervisor/colleagues?
  - Put a copy of the PDF file or of the compressed folder somewhere
  - Keep the sent email for later use

A possible workflow

Writing a review or a Ph.D. thesis

- How do you make writing experiments?
- How do you create/view checkpoints?
- Which version did you send to your supervisor/colleagues?
- How long did it take to write this section?
- When did I start writing this chapter?
- How much did I write on average per day?

A possible workflow

Writing a review or a Ph.D. thesis

- How do you make writing experiments?
- How do you create/view checkpoints?
- Which version did you send to your supervisor/colleagues?
- How long did it take to write this section?
- When did I start writing this chapter?
- How much did I write on average per day?

Everything by hand, error-prone and big overhead!

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
    - You work on separate parts at the same time
    - Only one person works at the same time

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
- How do you merge work from other people in the team?

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
- How do you merge work from other people in the team?
  - You send the changed files per email and put them in the folder by hand
  - Copy/Rsync in some shared place the new status of the project
  - If only one person works at once, a compressed archive can be exchanged

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
- How do you merge work from other people in the team?
- How do you work on different machines?

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
- How do you merge work from other people in the team?
- How do you work on different machines?
    - You don’t, use `SSH`
    - Different machines are as different people, see above

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
- How do you merge work from other people in the team?
- How do you work on different machines?
- How do you know who did what?

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
- How do you merge work from other people in the team?
- How do you work on different machines?
- How do you know who did what?
    - This information is not important
    - Sending work around per email allows to trace this...
    - Put comments into the source!

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
- How do you merge work from other people in the team?
- How do you work on different machines?
- How do you know who did what?
- How do you give credit to authors?

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
- How do you merge work from other people in the team?
- How do you work on different machines?
- How do you know who did what?
- How do you give credit to authors?
    - Detailed information is not important
    - A rough idea about who worked on what is enough
    - See comments into the source!

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
- How do you merge work from other people in the team?
- How do you work on different machines?
- How do you know who did what?
- How do you give credit to authors?
- How do you go back in history e.g. in case of a bug?

Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
- How do you merge work from other people in the team?
- How do you work on different machines?
- How do you know who did what?
- How do you give credit to authors?
- How do you go back in history e.g. in case of a bug?
    - Again, use the archive sent around per email
    - Using a shared place, this is not possible $\longrightarrow$ debug!

Another possible workflow

Collaborating on a project

- How can you ollab ject with colleagues?
- How do you m in the team?
- How do you ork on di chine
- How do you h
- How do you g
- How do you go f a bug?
    - Again, use the ar
    - Using a shared place, this ossible $\longrightarrow$ debug!

OK, and how would it be with Git?

A possible workflow

Writing a review or a Ph.D. thesis

- How do you make writing experiments?
  - Just do them (staging/stash area)
  - `git-branch`
- How do you create/view checkpoints?
  - `git-log` `git-tag` `git-checkout`
- Which version did you send to your supervisor/colleagues?
  - `git-log` `git-tag`
- How long did it take to write this section?
- When did I start writing this chapter?
- How much did I write on average per day?

} `git-shortlog` `git-log` `gitstats`*

* This is just one of the pletora of libraries to make statistics based on a git repository.

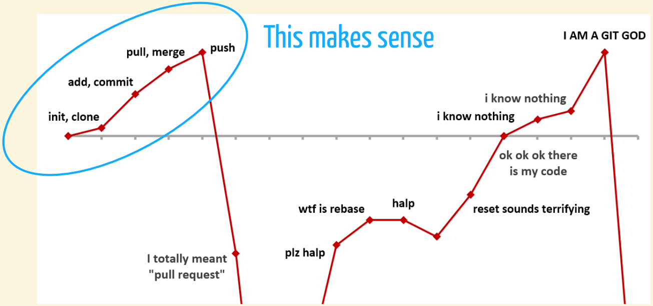
Another possible workflow

Collaborating on a project

- How can you collaborate on the same project with colleagues?
  - `git-pull` `git-push` `git-branch`
- How do you merge work from other people in the team?
  - `git-merge`
- How do you work on different machines?
  - `git-pull` `git-push`
- How do you know who did what?
  - `git-blame`
- How do you give credit to authors?
  - `git-shortlog`
- How do you go back in history e.g. in case of a bug?
  - `git-checkout` `git-bisect`

Yes, but I have to learn all those commands!

There are many jokes on the web...



...but after all it is about having the correct mental set up!

Git was not designed to aim to user-friendliness, but it is by far the most used version control system in the world and hence in continuous evolution.

Yes, but I have to learn all those commands!

- As any new tool, it needs some practice
- The short- to long-term payoff is worth the effort
- It is plenty of 🔗GUI clients
  - 🔗Sourcetree: A Free GIT Client For Windows And Mac
  - 🔗Aurees: Easy-Fast-Free { Windows, Mac & Linux }
  - 🔗Git-Cola: Powerful GUI For GIT { Windows, Mac, Ubuntu & Linux }
  - [...]
- You can work in the terminal

  → after this (and next) talk it will be possible!

Git was not designed to aim to user-friendliness, but it is by far the most used version control system in the world and hence in continuous evolution.

Last but not least

Which large famous products are developed using Git?

Linux, Homebrew, Windows, Tensorflow, Angular, Inkscape, ...

Last but not least

Which large famous products are developed using Git?

Linux, Homebrew, Windows, Tensorflow, Angular, Inkscape, ...

And if I do not have so large projects?

It doesn't matter! There are too many advantages* having a project under a source code management tool. Even alone. **Simply use one (Git). Now.**

For collaborative projects like maintaining code in a group, handing it over from person to person and so on, Git is simply a must. **As project leader, you should think about requiring everybody to work in a Git repository.**

* Among many others, in a Git repository you can undo a `rm` command given by accident on a wrong file.

What is Git?

How does Git define itself?

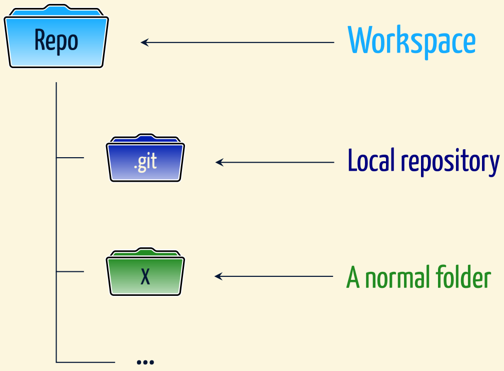
«Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning fast performance.»

Git homepage

1. Free and open
2. **Distributed version control system**
3. From small to very large projects
4. With speed and efficiency
5. Easy to learn

How does it work?

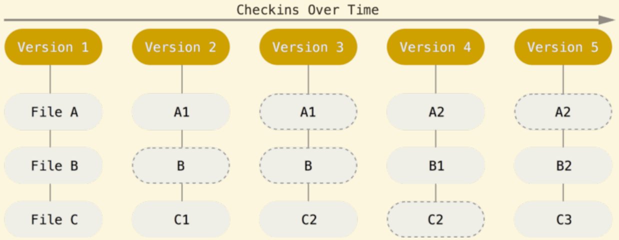
- **Repository:** a database containing all versions of the files



How does it work?

- **Repository:** a database containing all versions of the files
- Snapshot-based system
  - Snapshots are called commits
  - Commits are named by checksums (also used to ensure data integrity)

{ It's impossible to change the contents of any file or directory without Git knowing about it }



🔗 From the Git-Book

How does it work?

- **Repository:** a database containing all versions of the files
- Snapshot-based system
  - Snapshots are called commits
  - Commits are named by checksums (also used to ensure data integrity)

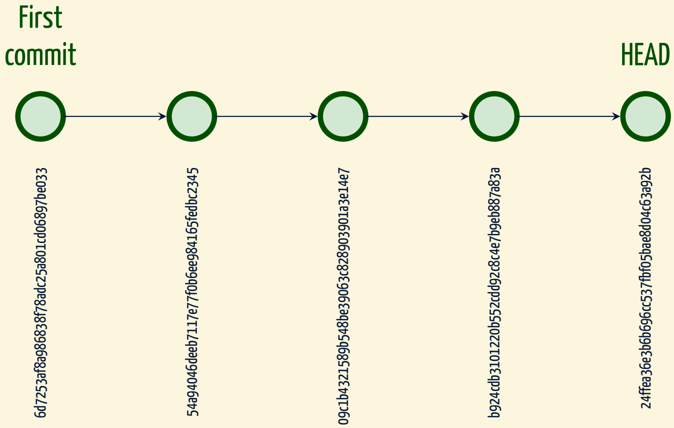
    { It's impossible to change the contents of any file or directory without Git knowing about it }
- Almost every operation is local
  - Working without network connecting
  - Distributed system $\longrightarrow$ everyone carries a backup!

Are you curious to know how Git works bottom-up?

Refer to 🔗this 31-pages document , well written, but not needed at start.

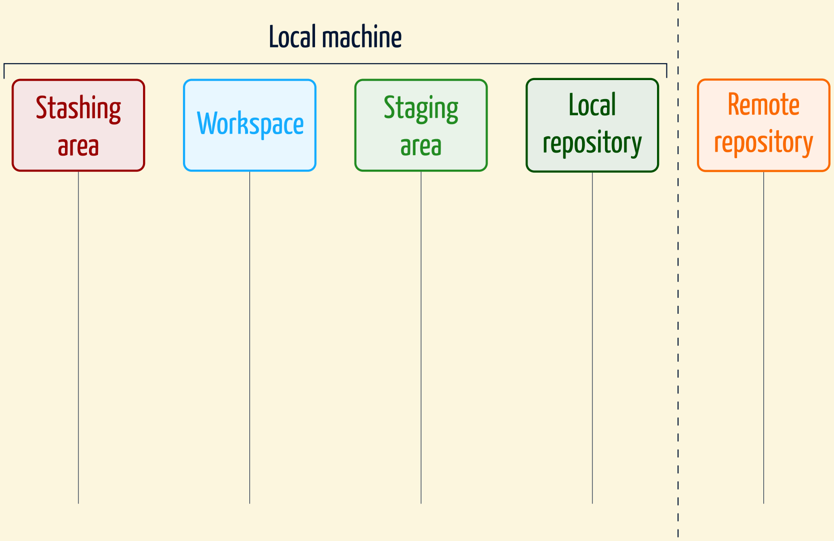
An example of Git history

Every commit is a snapshot of the state of the repository at that point

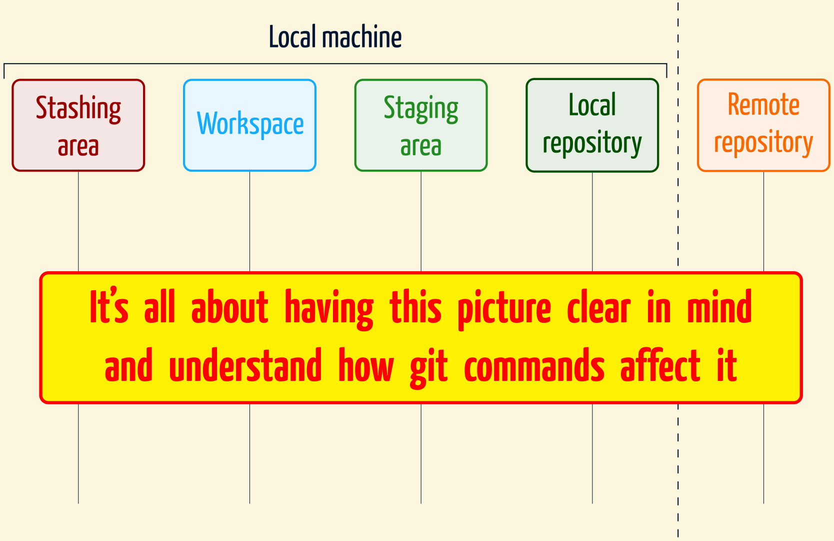

First commit
HEAD
6d7253af8a986838f78adc25a801cd06897be033
54a94046deeb7117e77f0b6ee984165fedbc2345
09c1b4321589b548be39063c82890390 1a3e14e7
b924cdb3101220b552cdd92c8c4e7b9eb887a83a
24ffea36e3b6b696cc537fbf05bae8d04c63a92b

The correct abstract mental setup

The correct abstract mental setup

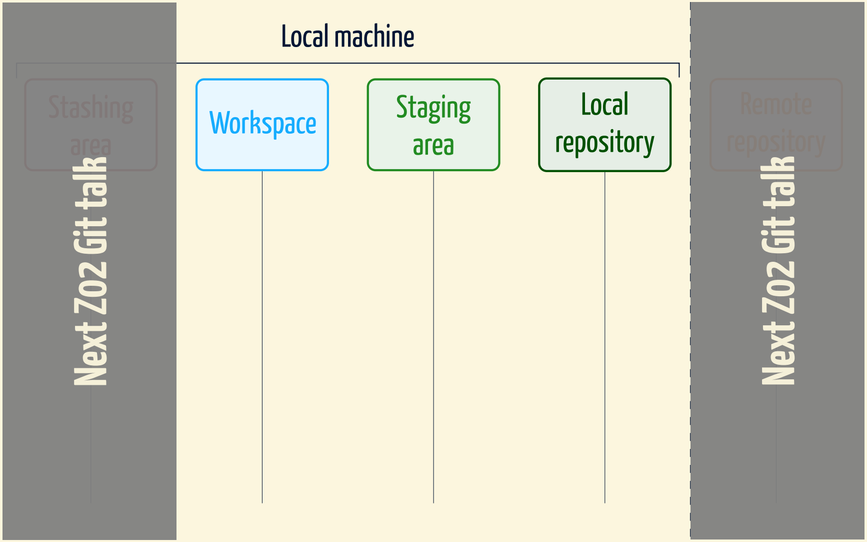


The correct abstract mental setup



It’s all about having this picture clear in mind and understand how git commands affect it

The correct abstract mental setup



How to use Git locally?

Preliminary steps

Be sure to introduce yourself to Git on **each** machine from which you work

1. It is likely that Git is installed on your machine.
   - Check it in a terminal e.g. via `git version`
   - If needed, install it
2. Optionally, get/enable autocompletion in the terminal
3. Tell Git who you are and your email address

   → this information will be used to sign your work in history

```
$ git config --global user.name 'Alessandro Sciarra'
$ git config --global user.email 'sciarra@itp.uni-frankfurt.de'
```

4. Set your favourite editor e.g. to write commit messages

```
$ git config --global core.editor 'emacs -nw'
```

The dollar symbol $ denotes the end of the prompt in the terminal. It will be from now on to denote commands to be used in the terminal.

Asking for help about Git

1. There are 3 ways in terminal
   - `git help <command>` e.g. `git help config`
   - `git <command> --help` e.g. `git config --help`
   - `man git-<command>` e.g. `man git-config`
2. List of commands on the official reference
3. Ask Google

There is plenty of cheat-sheets online:

GitHub education GitLab Bitbucket

Creating a repository

It is as simple as running one command

```
$ git config --get user.name
Alessandro Sciarra
$ git config --get user.email
sciarra@itp.uni-frankfurt.de
# Suppose to be in a folder you want to turn into a repository
$ pwd
/home/asciarra/Documents/first-repo
$ ls -a
.  ..  Paper.aux  Paper.log  Paper.out  Paper.pdf  Paper.tex
```

Creating a repository

It is as simple as running one command

```
$ git config --get user.name
Alessandro Sciarra
$ git config --get user.email
sciarra@itp.uni-frankfurt.de
# Suppose to be in a folder you want to turn into a repository
$ pwd
/home/asciarra/Documents/first-repo
$ ls -a
.  ..  Paper.aux  Paper.log  Paper.out  Paper.pdf  Paper.tex
```

```
$ git init  # <--- Here you go!
Initialised empty Git repository in ~/Documents/first-repo/.git/
$ ls -a
.  ..  .git  Paper.aux  Paper.log  Paper.out  Paper.pdf  Paper.tex
```

Creating a repository

It is as simple as running one command

```
$ git config --get user.name
Alessandro Sciarra
$ git config --get user.email
sciarra@itp.uni-frankfurt.de
# Suppose to be in a folder you want to turn into a repository
$ pwd
/home/asciarra/Documents/first-repo
$ ls -a
.  ..  Paper.aux  Paper.log  Paper.out  Paper.pdf  Paper.tex
```

```
$ git init  # <--- Here you go!
Initialised empty Git repository in ~/Documents/first-repo/.git/
$ ls -a
.  ..  .git  Paper.aux  Paper.log  Paper.out  Paper.pdf  Paper.tex
```

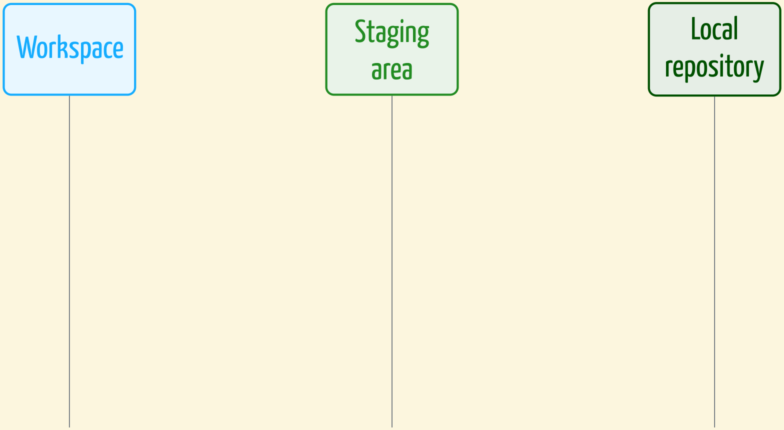
Do not shoot yourself!

Never ever touch by hand the content of the `.git` folder.

What comes next?

What comes next?

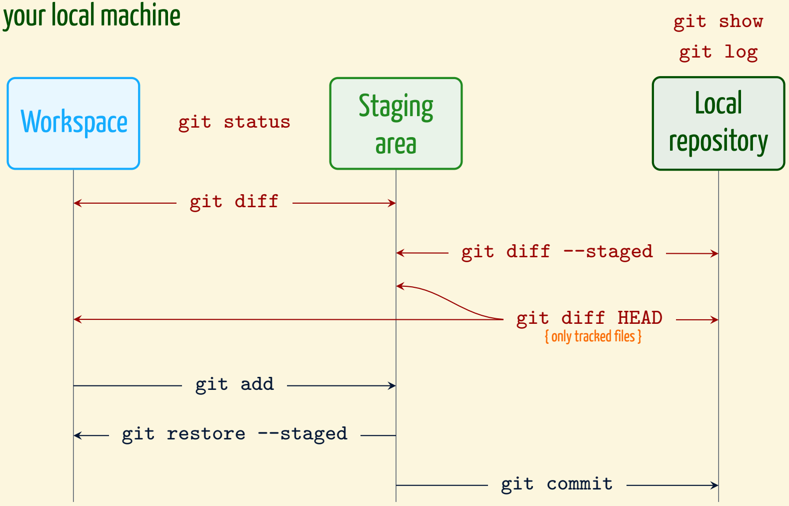
On your local machine



Commands marked in dark red do not change anything in the repository!

What comes next?

On your local machine



Commands marked in dark red do not change anything in the repository!

Git introduced `git-restore` in v2.23 but this stayed buggy for a while. Use it from v2.27 on, otherwhise use `git-reset`.

Git status

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
     Paper.aux
     Paper.log
     Paper.out
     Paper.pdf
     Paper.tex

nothing added to commit but untracked files present
(use "git add" to track)
```

You do not want to put everything in a repository!

It is possible to tell git to ignore some files, like temporary ones

Letting Git ignore some files

```
$ printf '*.%s\n' {aux,log,out,pdf} > .gitignore
$ cat .gitignore
*.aux
*.log
*.out
*.pdf
$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present
(use "git add" to track)
```

github/gitignore   for LaTeX projects

Our first commit

In your terminal

```
$ git log
fatal: your current branch 'master' does not have any commits yet
$ git add .gitignore
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .gitignore

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

Our first commit

In your favourite editor

```
█

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   .gitignore
#
# Untracked files:
#       Paper.tex
#
```

Our first commit

In your favourite editor

```
Add .gitignore file for TeX project

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   .gitignore
#
# Untracked files:
#       Paper.tex
#
```

Our first commit

In your terminal

```
$ git log
fatal: your current branch 'master' does not have any commits yet
$ git add .gitignore
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   .gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Paper.tex
$ git commit
# Your editor opens -> type commit message, save and exit
[master (root-commit) bb8c78b] Add .gitignore file for TeX project
 1 file changed, 4 insertions(+)
 create mode 100644 .gitignore
```

Inspecting history

```
$ git log
commit bb8c78b68075dacf8467420bc00867c73ef5ba8c (HEAD -> master)
Author: Alessandro Sciarra <asciarra@fias.uni-frankfurt.de>
Date:   Thu Dec 23 10:13:05 2021 +0100

    Add .gitignore file for TeX project
$ git log --oneline
bb8c78b (HEAD -> master) Add .gitignore file for LaTeX project
```

Use a pager to avoid polluting terminal

```
$ git config --global core.pager 'less -+$LESS -R'
```

Use `git show` or `git show <SHA1>` to inspect what has been done in last or given commit

Our second commit

```
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Paper.tex

nothing added to commit but untracked files present
(use "git add" to track)
$ git add Paper.tex # Always add to the staging
                    # area before committing!
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
       new file: Paper.tex
$ git commit -m 'Add paper main document'
[master 9c6154d] Add paper main document
 1 file changed, 147 insertions(+)
 create mode 100644 Paper.tex
```

Use good commit messages

```
$ git log --oneline
9c6154d (HEAD -> master) Add paper main document
bb8c78b Add .gitignore file for LaTeX project
```

- Write them like a email to yourself / the other developers
  $\longrightarrow$ Subject line + body, follow the 50/72 rule
- Subject: Summarize what has been done
  $\longrightarrow$ **Use present tense and no period at the end!**
- Body: After empty line, document why you made the changes
  ↳ { add one only if needed }

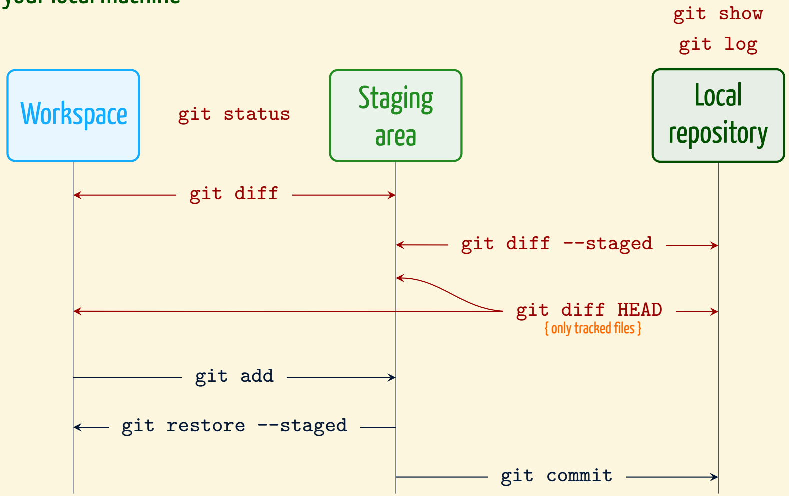
Good commits

Commit small and conceptually separated changes, commit often and do not add binary files to your repository.

In coding projects, although not always possible/trivial, try to commit working/compiling code.

Back to our mental picture

On your local machine



Commands marked in dark red do not change anything in the repository!

Git introduced `git-restore` in v2.23 but this stayed buggy for a while. Use it from v2.27 on, otherwise use `git-reset`.

Working and displaying changes

In your terminal

```
# Make some changes
$ git status
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: Paper.tex

no changes added to commit (use "git add" and/or "git commit -a")
$ git diff
```

Working and displaying changes

In your pager, e.g. `less`

```
diff --git a/Paper.tex b/Paper.tex
index 3c408e5..3669114 100644
--- a/Paper.tex
+++ b/Paper.tex
@@ -42,7 +42,7 @@ pdftitle={LaTeX Seminar for PhD students}
     {Sprecher \& Seminarleiter}%
 }

-\date{\February 25, 2022}
+\date{23. Dezember 2021}

 \newcommand{\etc}{etc.}
 \newcommand{\zB}{z.B.}
```

Working and displaying changes

In your terminal

```
# Make some changes
$ git status
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: Paper.tex

no changes added to commit (use "git add" and/or "git commit -a")
$ git diff
$ git diff --staged  # Nothing in the staging area!
$ git add Paper.tex
$ git diff           # No changes anymore in the workspace!
$ git diff --staged  # Our changes are now staged
$ git commit -m 'Fix date for main document'
# ...
```

What else can I easily explore?

- Stage all tracked modified files at once

```
git add -u
```

- Stage partial modification in a file

```
git add -p
```

- Define your aliases

```
git config --global alias.unstage 'reset HEAD --'
# From now on, you can use 'git unstage'
```

- Let git correct you when you mistype*

```
git config --global help.autocorrect 1
```

- Change/correct your last commit message

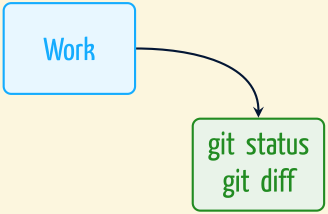
```
git commit --amend
```

* If you want a friend to correct your mistyped last command **in general** in the terminal, check-out this hilarious project .

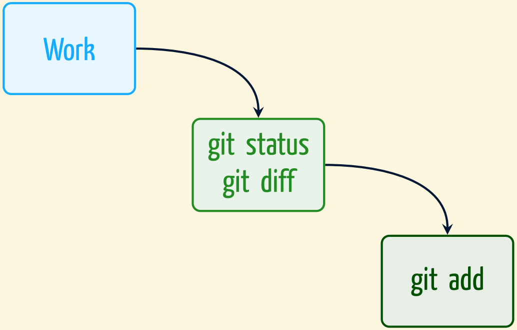
Summary and conclusions

Your workflow from now on, right?

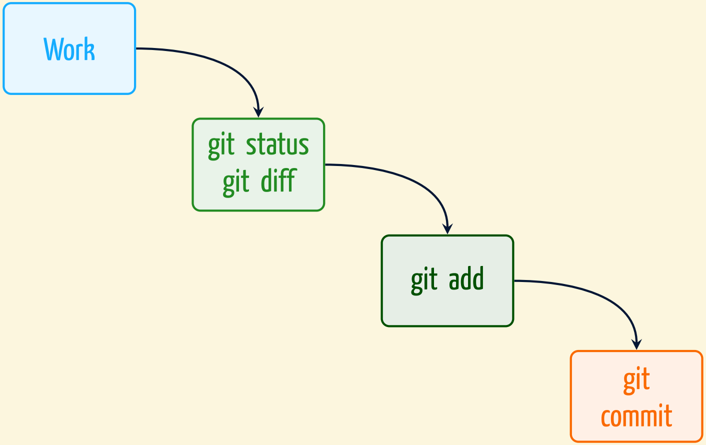
Your workflow from now on, right?



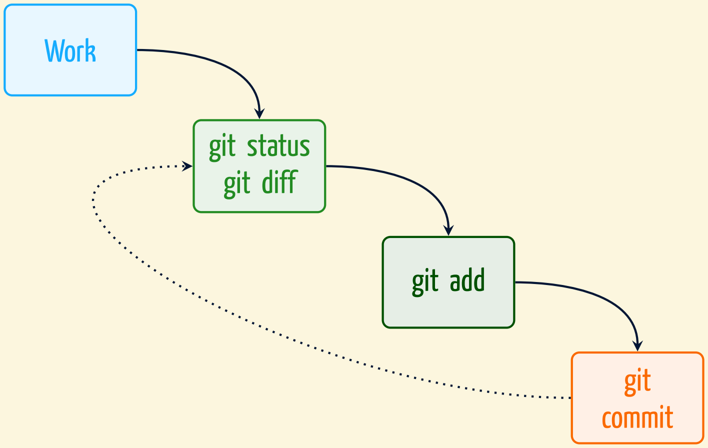
Your workflow from now on, right?



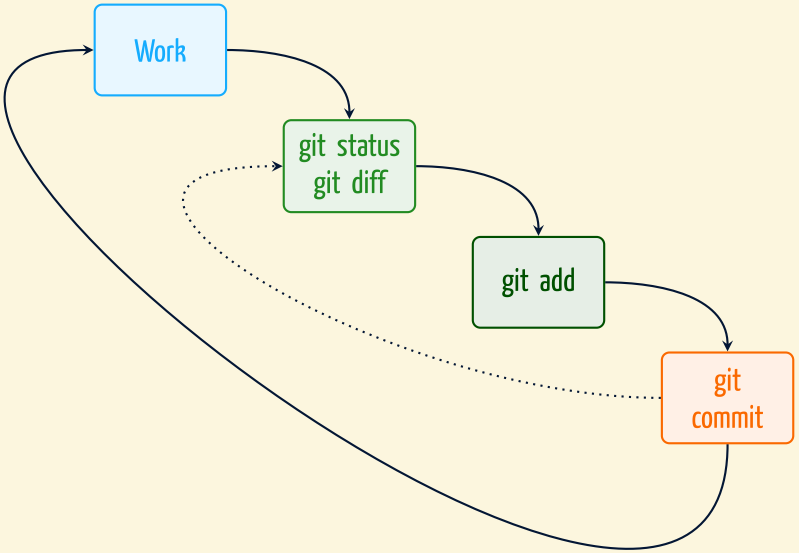
Your workflow from now on, right?



Your workflow from now on, right?



Your workflow from now on, right?



That's **NOT** all folks

1. Start using Git. **Now.** Not tomorrow or next week, today!
   $\longrightarrow$ Repeat what done on these slides

That's **NOT** all folks

1. Start using Git. **Now.** Not tomorrow or next week, today!
   → Repeat what done on these slides
2. Was anything unclear? Do you get stuck at some point?
   → Drop me ✉ an email

That's **NOT** all folks

1. Start using Git. **Now.** Not tomorrow or next week, today!
   → Repeat what done on these slides
2. Was anything unclear? Do you get stuck at some point?
   → Drop me ✉ an email
3. Git is much more than this!
   → Come to next Z02 talk: «Let's git **together**»

```
git clone
git branch
git switch
git checkout
git merge
git pull
git push
```

That's **NOT** all folks

Thank you!

1. Start using Git. **Now.** Not tomorrow or next week, today!
   → Repeat what done on these slides
2. Was anything unclear? Do you get stuck at some point?
   → Drop me ✉ an email
3. Git is much more than this!
   → Come to next Z02 talk: «Let's git **together**»

```
git clone
git branch
git switch
git checkout
git merge
git pull
git push
```

Believe me, it's worth it!