GIT

distributed version control system

Corso di

Programmazione di Dispositivi Mobili

prof. Ignazio Gallo

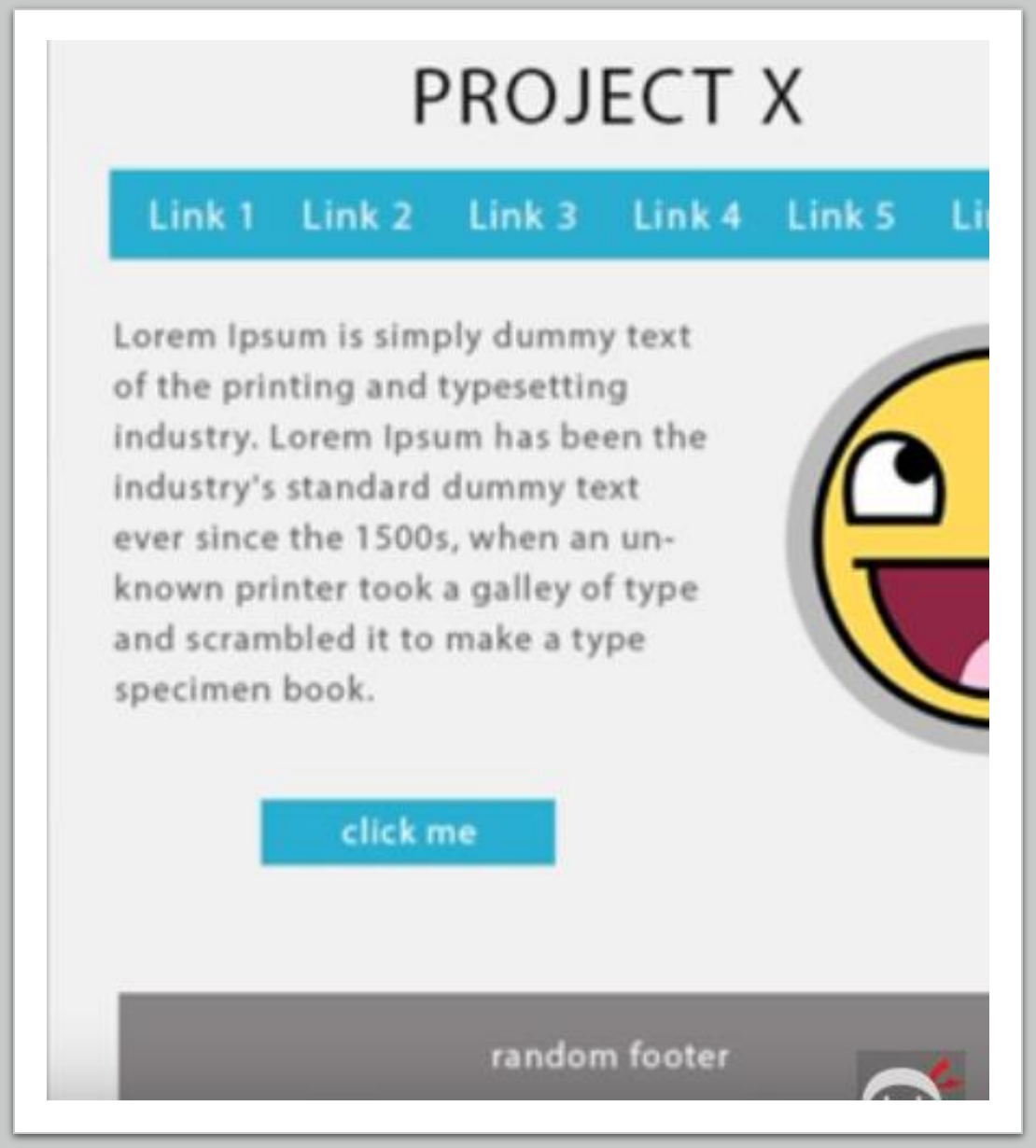
What is GIT?

- GIT is a "**distributed version control system**" !
- Tha basically means:
  - It's a system that records changes to our files over time
  - We can recall specific versions of those files at any time
  - Many people can easily collaborate on a project and have their own version of project files on their computer


if you use dropbox for version control
INSTRUCTOR
you're gonna have a bad time
quickmeme.com

A simple example

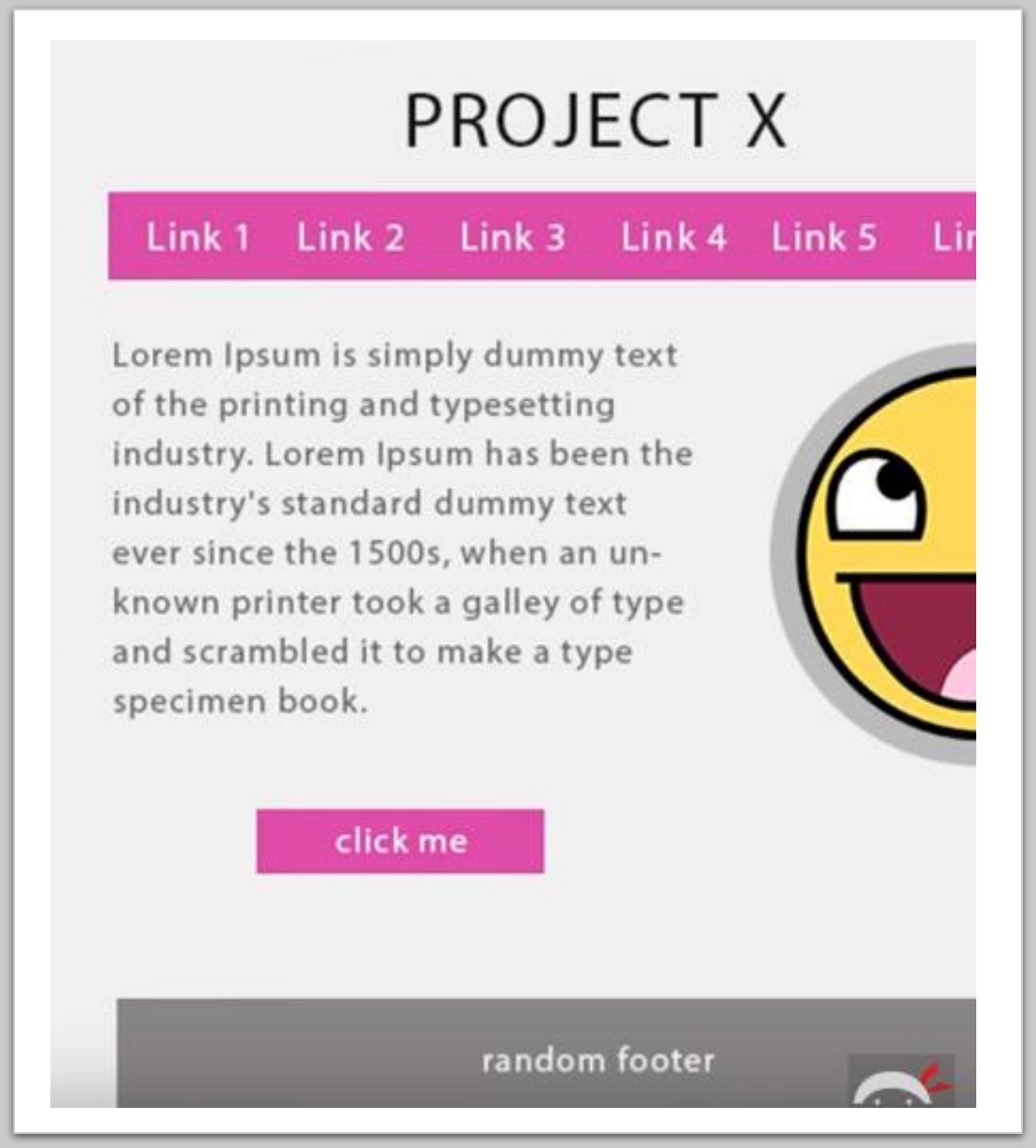
Project X for Client Y



A simple example

Project X for Client Y

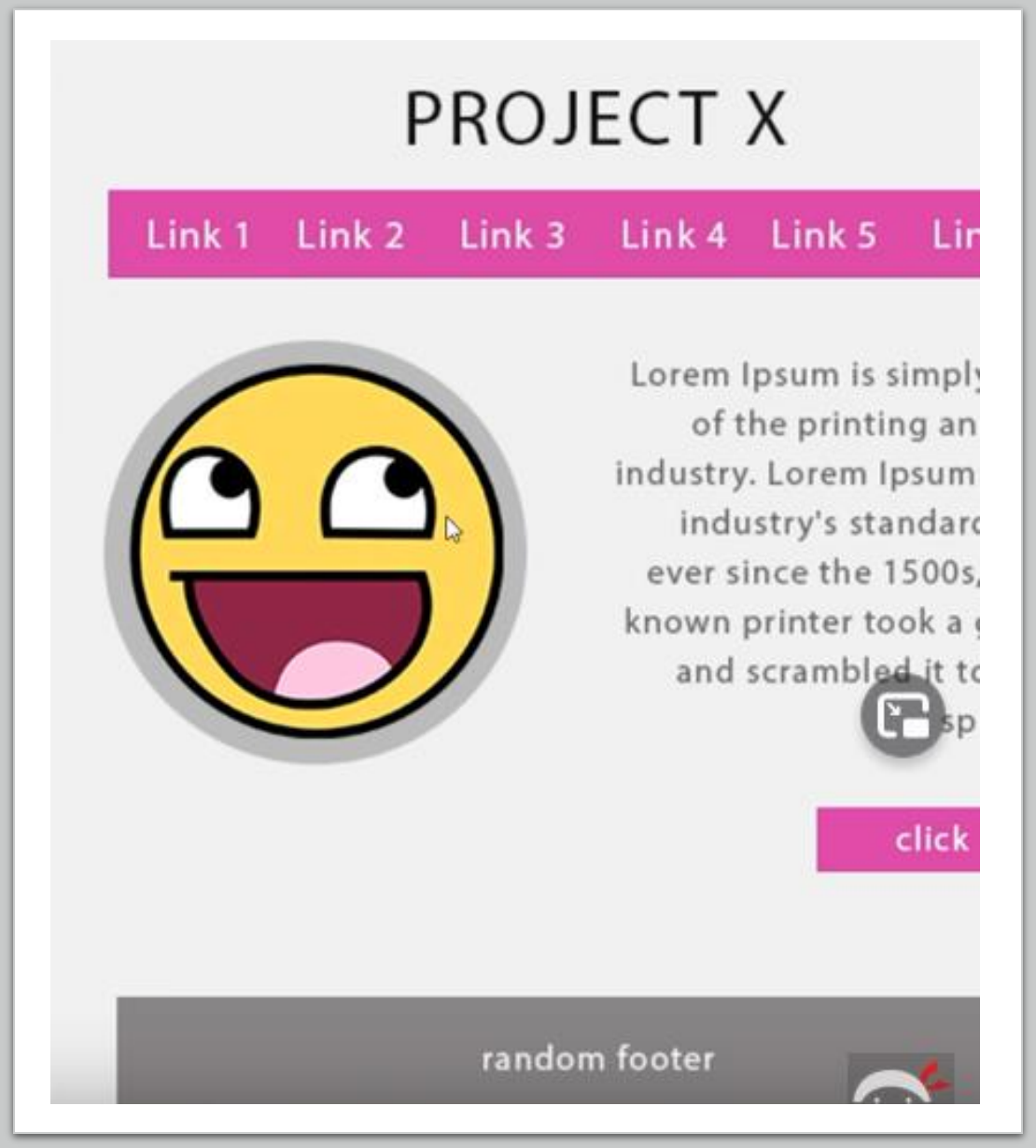
… The Client Y wants a different color



A simple example

Project X for Client Y

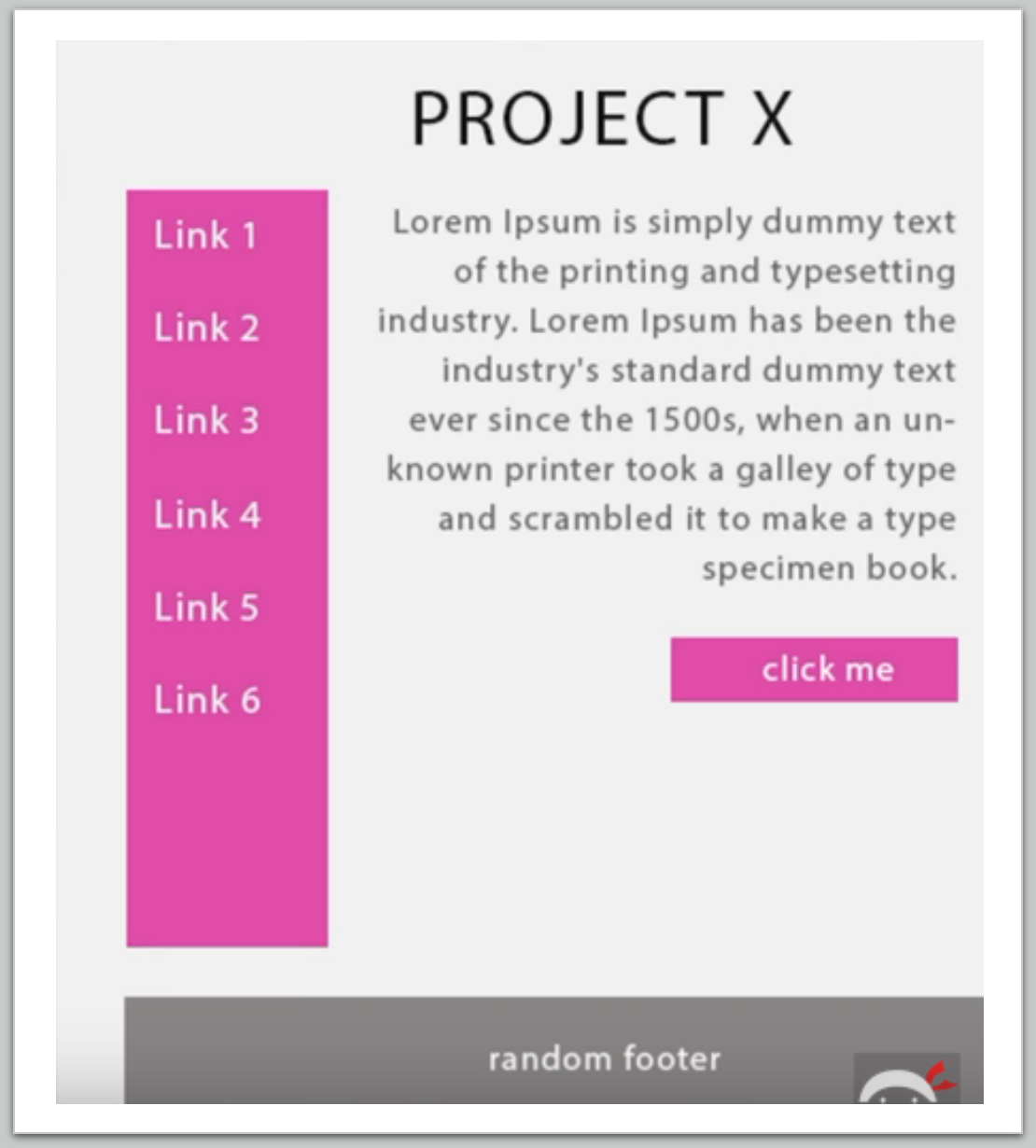
- … The Client Y wants the image on the left



A simple example

Project X for Client Y

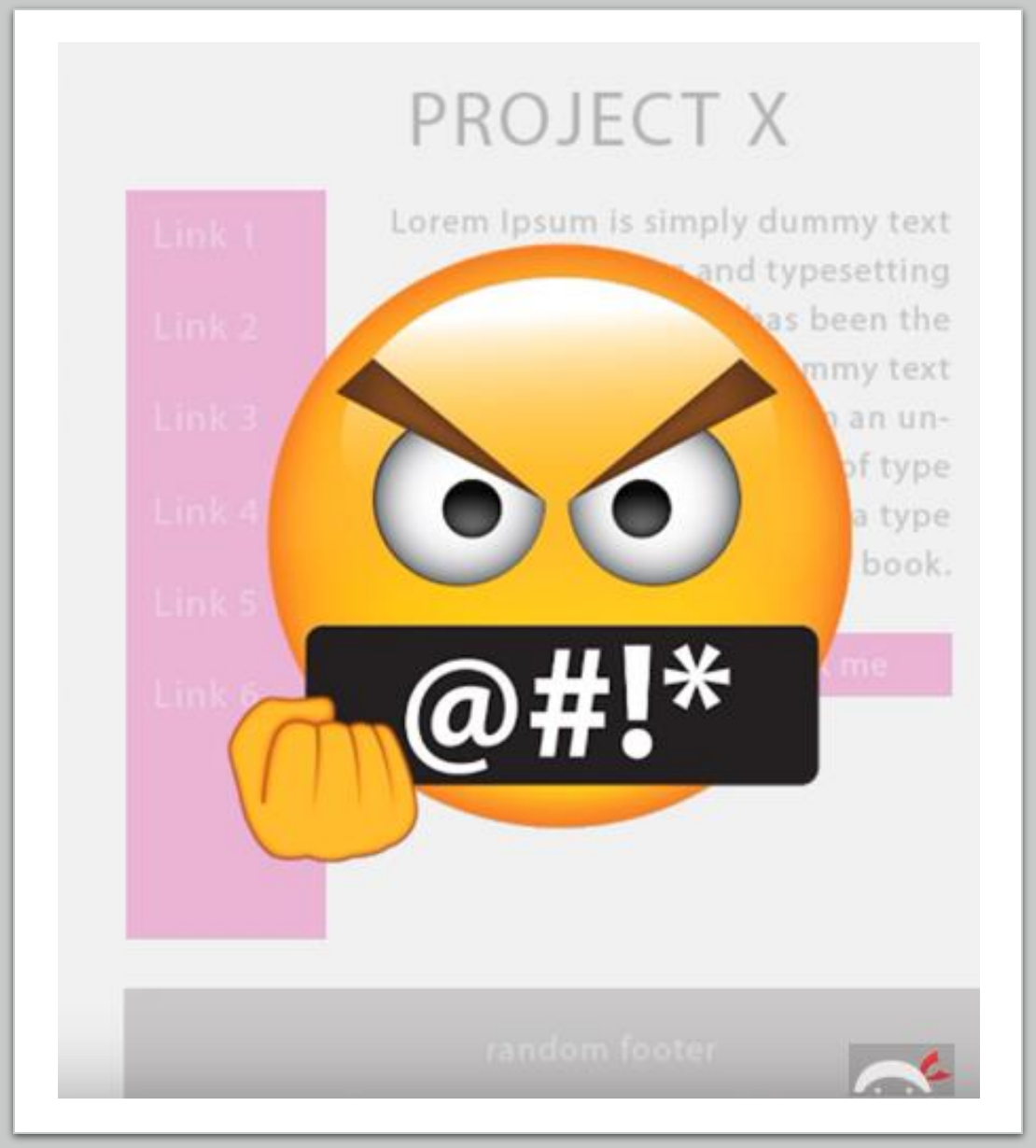
… The Client Y wants menu on the left



A simple example

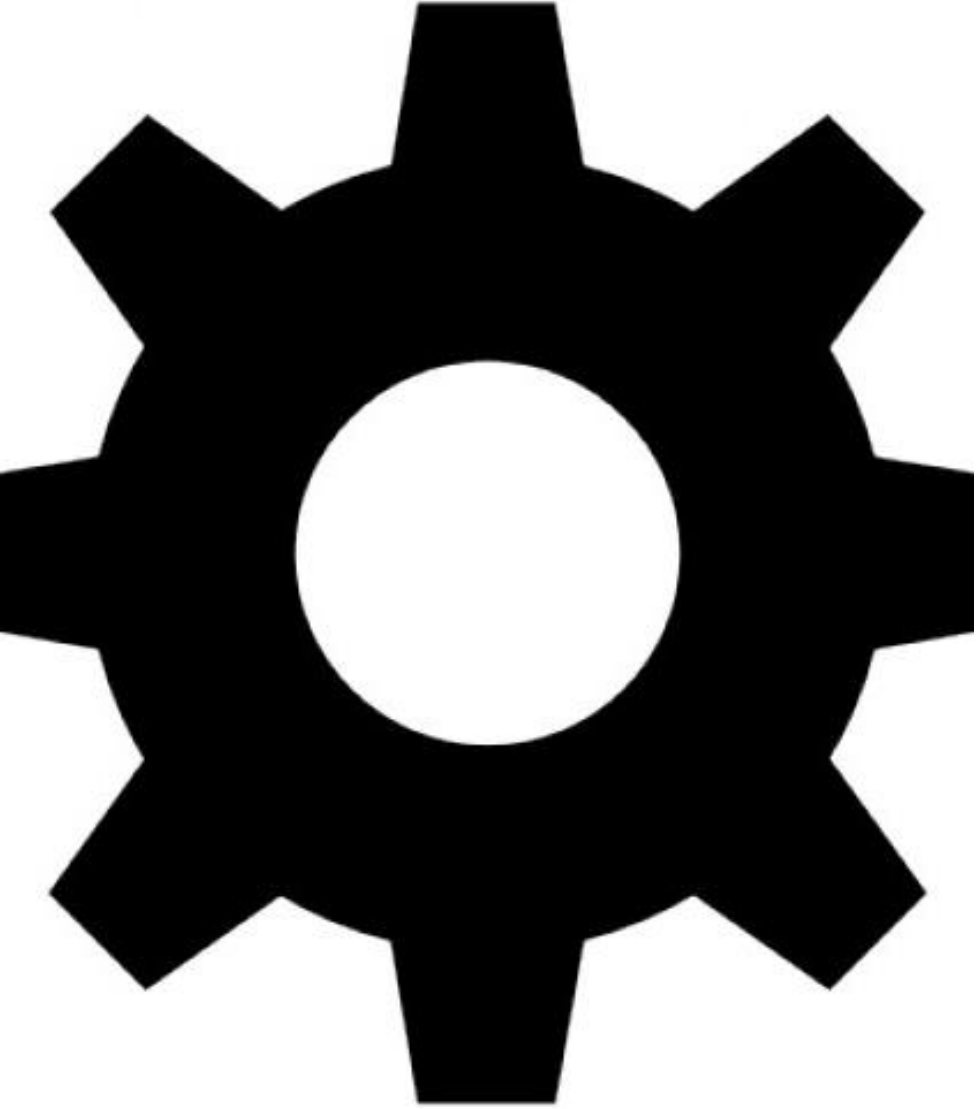
Project X for Client Y

… The Client Y prefers the first version!



A simple example

- Project X for Client Y - v1
- Project X for Client Y - v2
- Project X for Client Y - v3
- Project X for Client Y - v4
- … several days later…
- Project X for Client Y - v25



Installare GIT

- Si può scaricare https://git-scm.com/downloads
- oppure utilizzando i package manager del vostro sistema operativo.

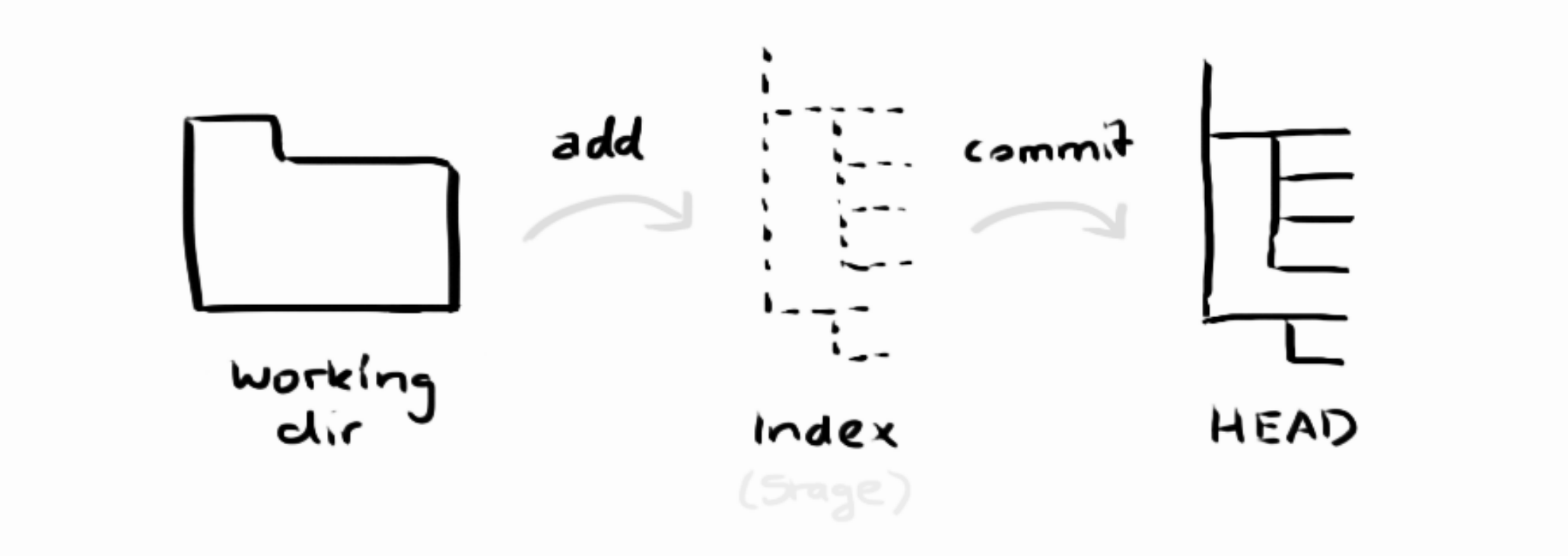
Create Git projects

- **Import**: Creating a new Git project means transforming the directory that hosts our code into a repository for versioning.
  - We just need to move inside the directory that interests us and run the init command
  - `$ git init`
- **Clone**: of a Git repository, when it already exists on another server (e.g., on GitLab or Github).
  - `$ git clone` https://gitlab.com/ignazio.gallo/todoapp-kotlin-2020.git

GIT - Workflow

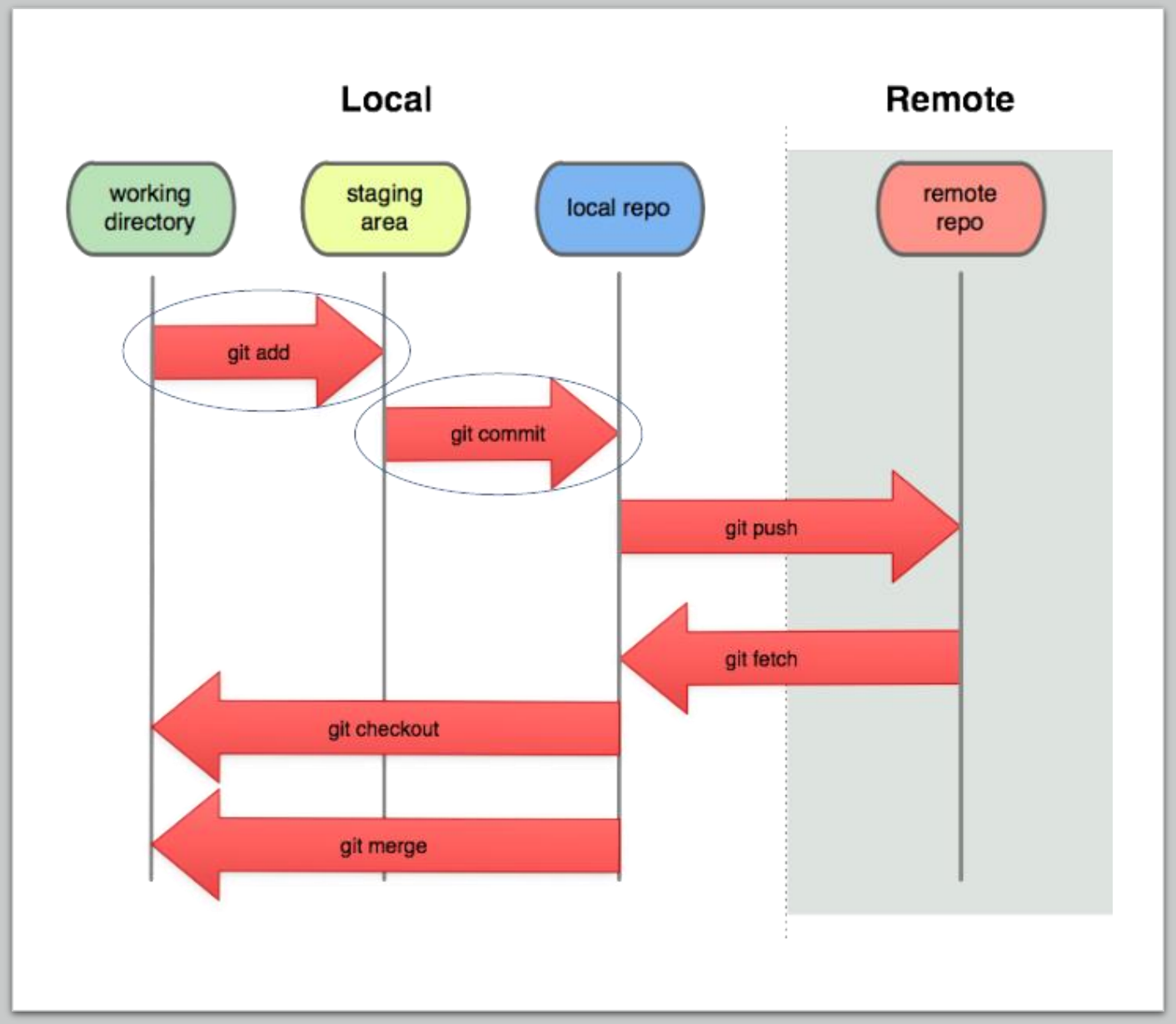
Your local repository consists of three "trees" maintained by git.

- the first one is your **Working Directory** which holds the actual files.
- the second one is the Index which acts as a **staging area** and
- finally, the **HEAD** which points to the last commit you've made.



GIT – Add and Commit

- You can propose changes (add it to the Index) using
  - git add <filename>
  - git add *
- To actually commit these changes use
  - git commit -m "Commit message"
- Now the file is committed to the HEAD, but **not** in your remote repository yet.

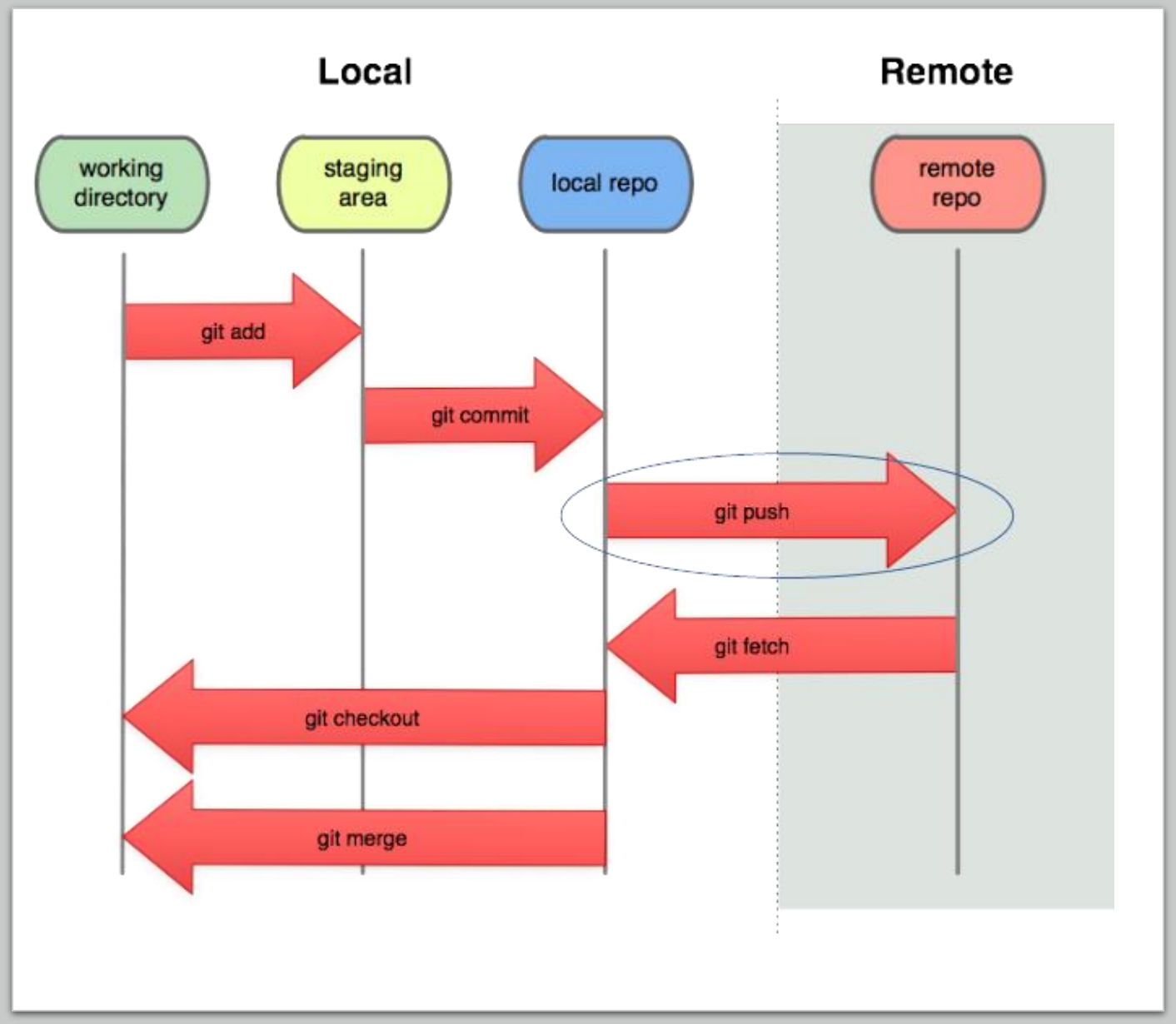


GIT – pushing changes

- Your changes are now in the HEAD of your local working copy.
- To send those changes to your remote repository, execute
  - git push origin master

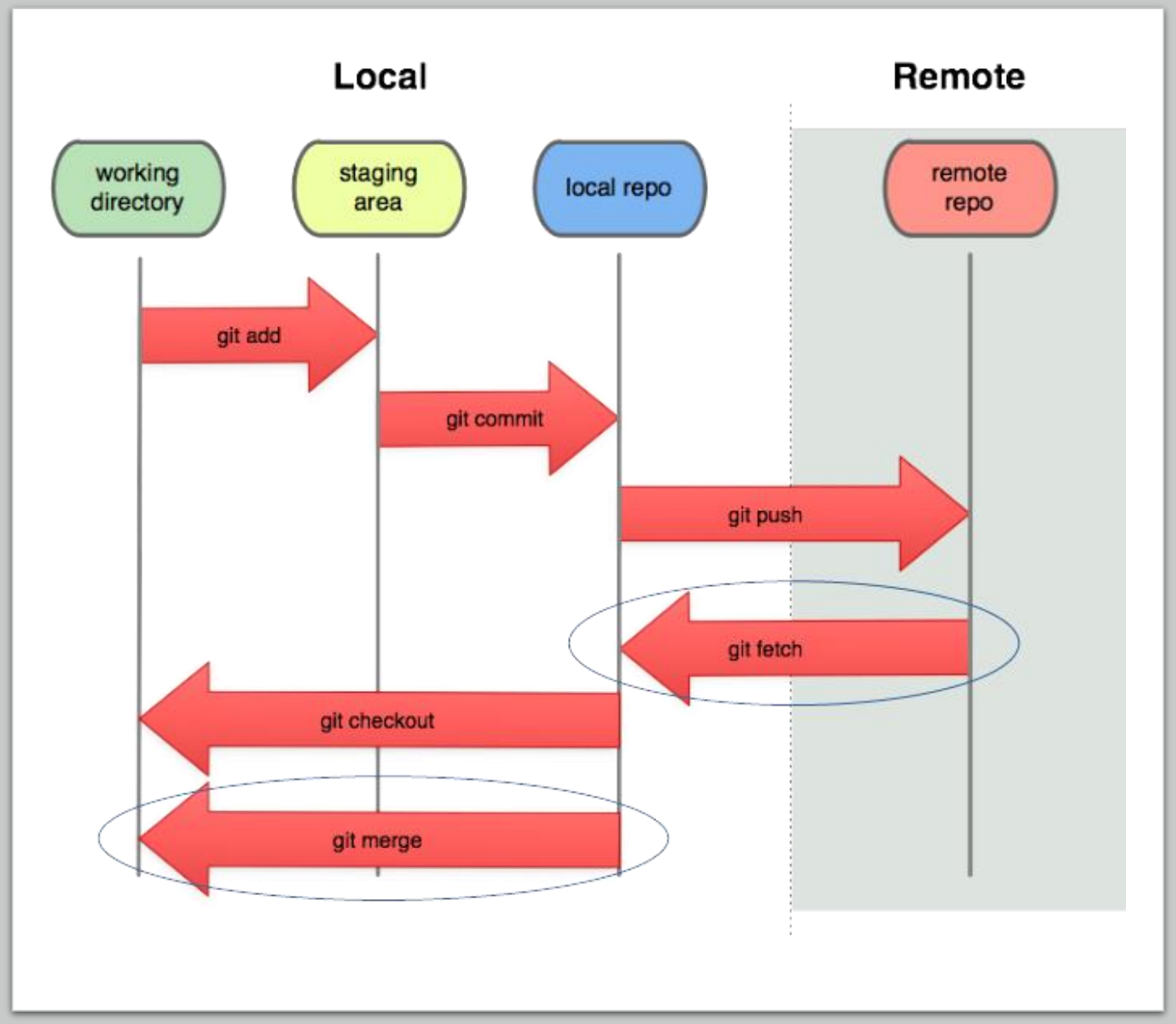
origin is the default name given to a remote repository

master is simply a branch name



GIT – update & merge

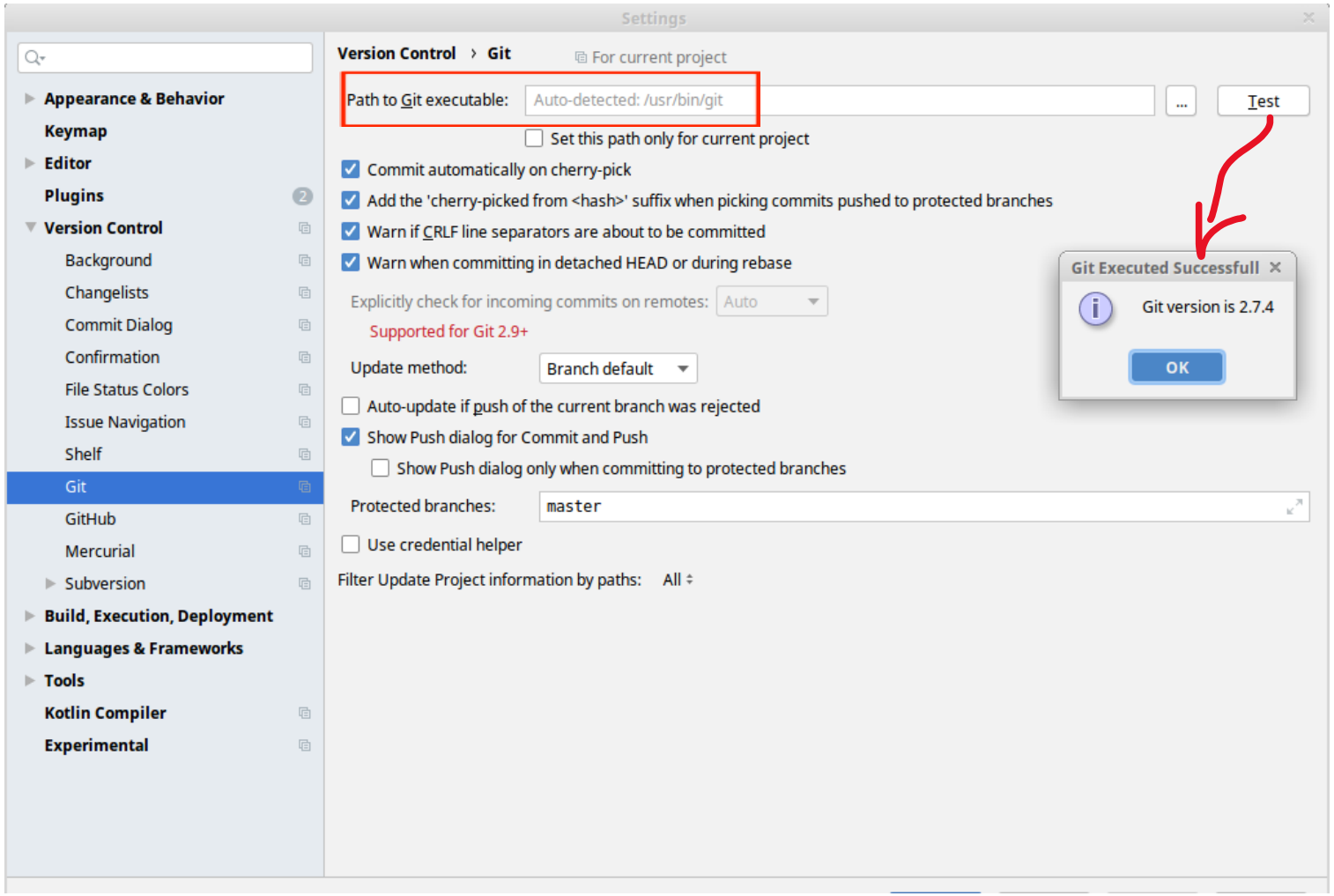
- to update your local repository to the newest commit, execute
  - git pull
- in your working directory to fetch and merge remote changes.



Git with Android Studio

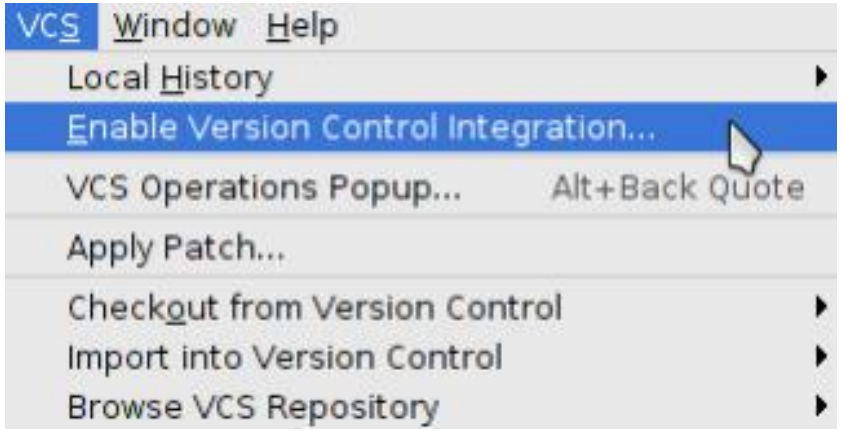
Connect GIT with AndroidStudio

Select: File > Settings...



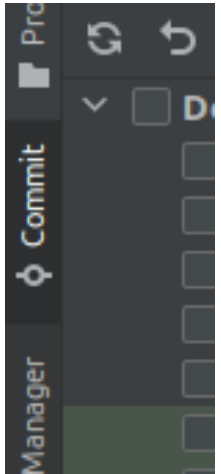
AndroidStudio – **Create** a Local GIT Repository

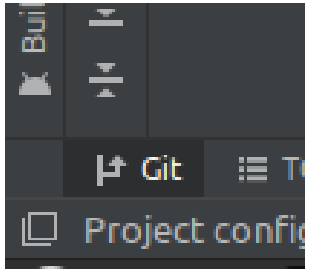
- You already have a working project
- Now you need to create a local repository for it
- Android Studio helps to automate this process.
- Under the VCS menu on the menu bar of Android Studio, select “Enable Version Control...”
- A dialog box will pop up asking you
- which version control system to use.
- Select Git and continue.



AndroidStudio
Add project files to local repository

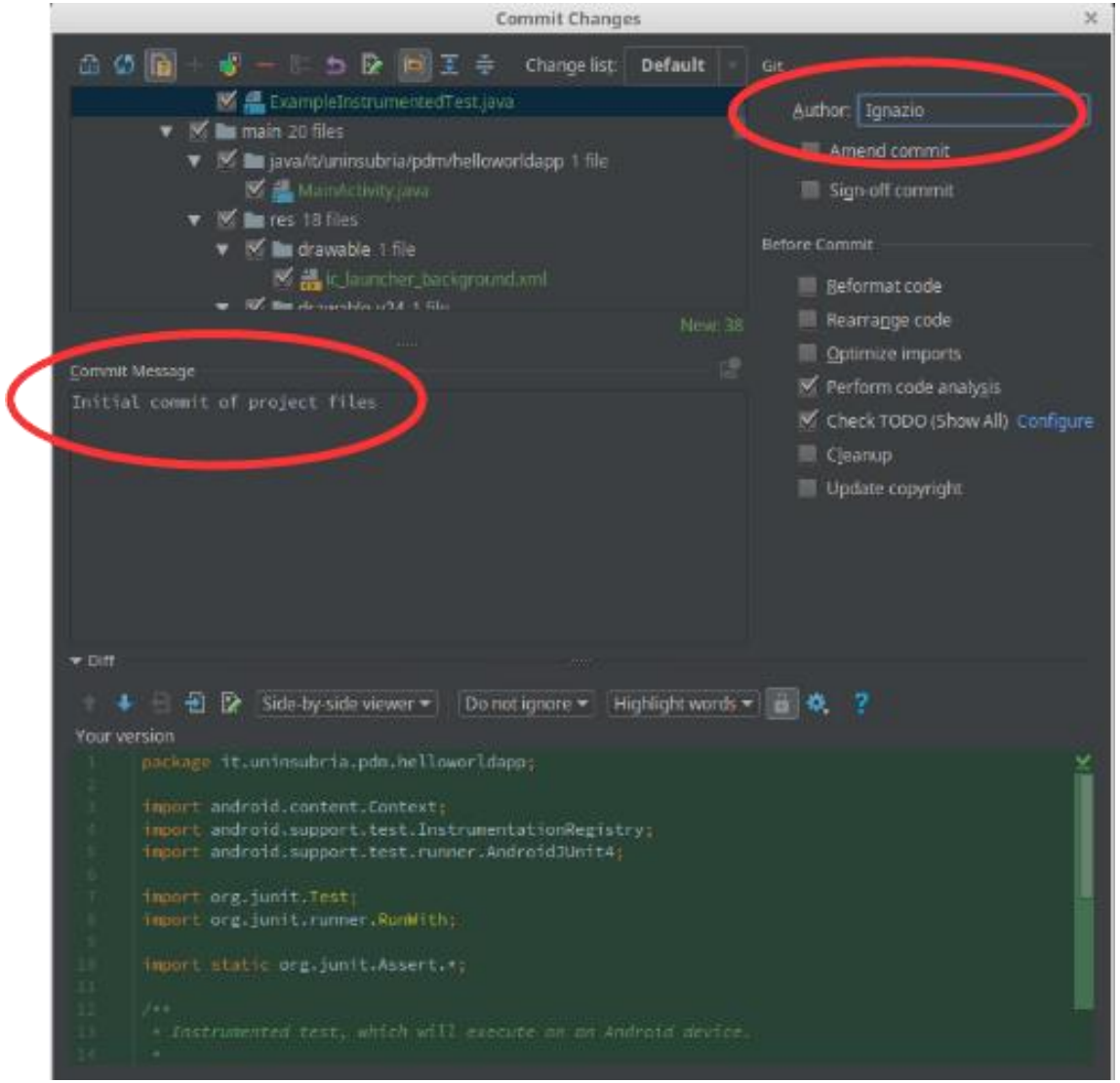
- Notice that most of the files in your projects are now colored red in the Project. View (the left frame).
- This means that they are not yet tracked by Git.
- In order for Git to track their histories, you must identify exactly which files to track.
- Open the "Commit" frame of the IDE. It shows the present status that Git reports for files. Explore also the "Git" frame.
- Clicking the triangle next to “Unversioned Files”
- Select all the files at once (Ctrl+A), right click, and
- Select “Add to VCS” from the menu.





AndroidStudio
Commit changes to the Local GIT Repository

- Select the files again, right click, and select "Commit Changes" to finish adding these files to the tracked history in Git.
- Notice the "Commit Message" portion of the dialog that pops up.
- This is where you should type a meaningful description of the changes that you are committing to the project.
- This time, just type "Initial commit of project files".
- Enter your name into the Author field as well and then "Commit".

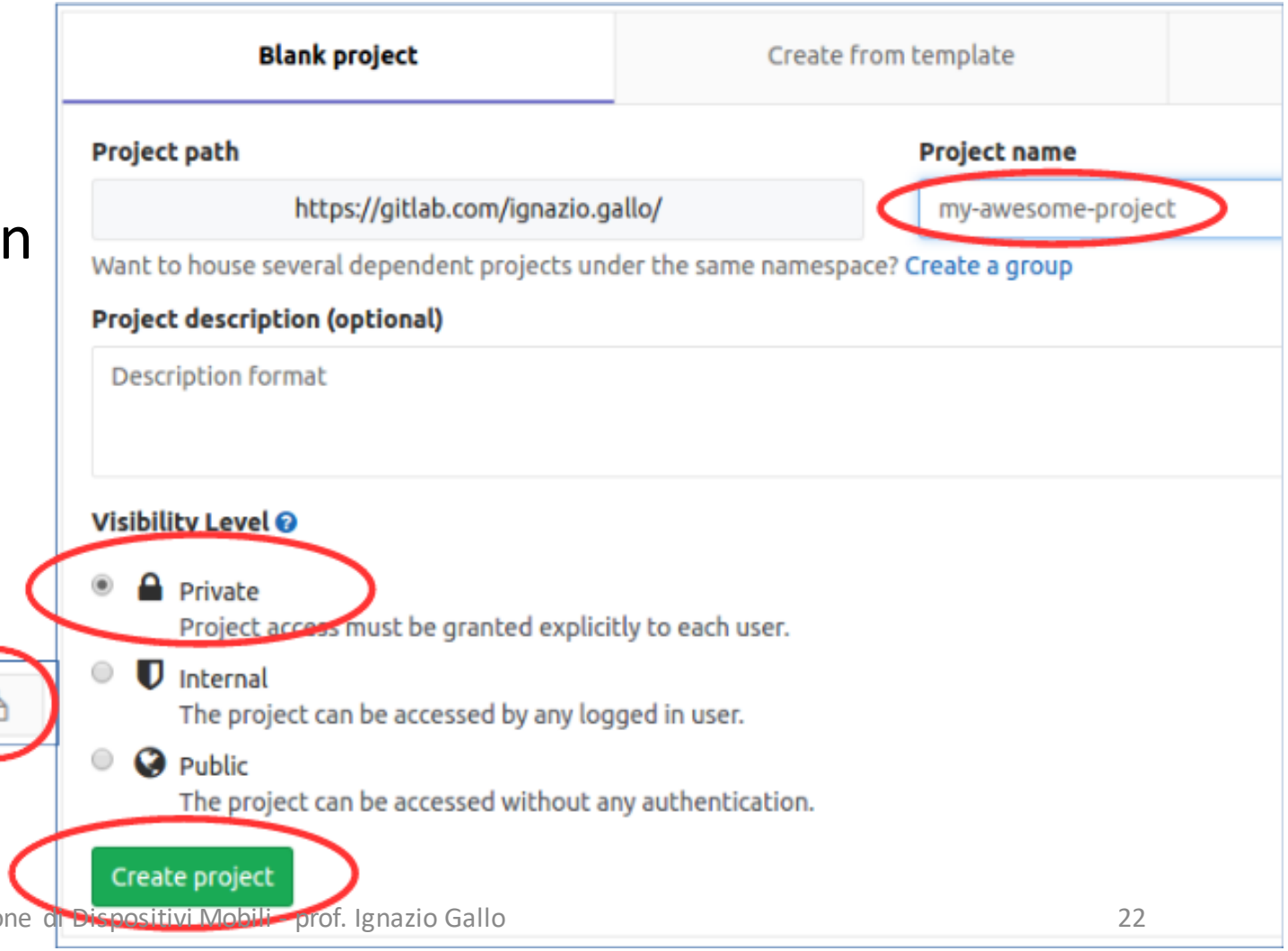


AndroidStudio – **Change** and **Commit**

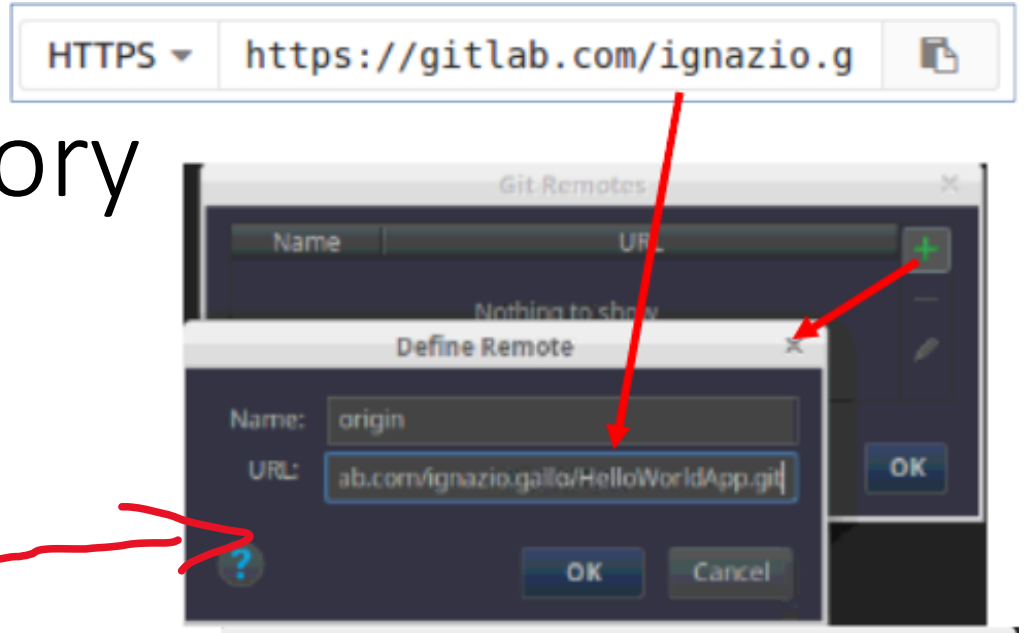
- Add a comment into the MainActivity.java file.
- Notice that the modified files turn blue within the Project View.
- This indicates that the files have changed, and those changes have not yet been committed to the local repository.
- Now commit your changes to Git using
  VCS→Commit Changes
  to commit all of the changes to the project
- Enter a meaningful commit message and continue.

Create GitLab Account and a Remote Repository

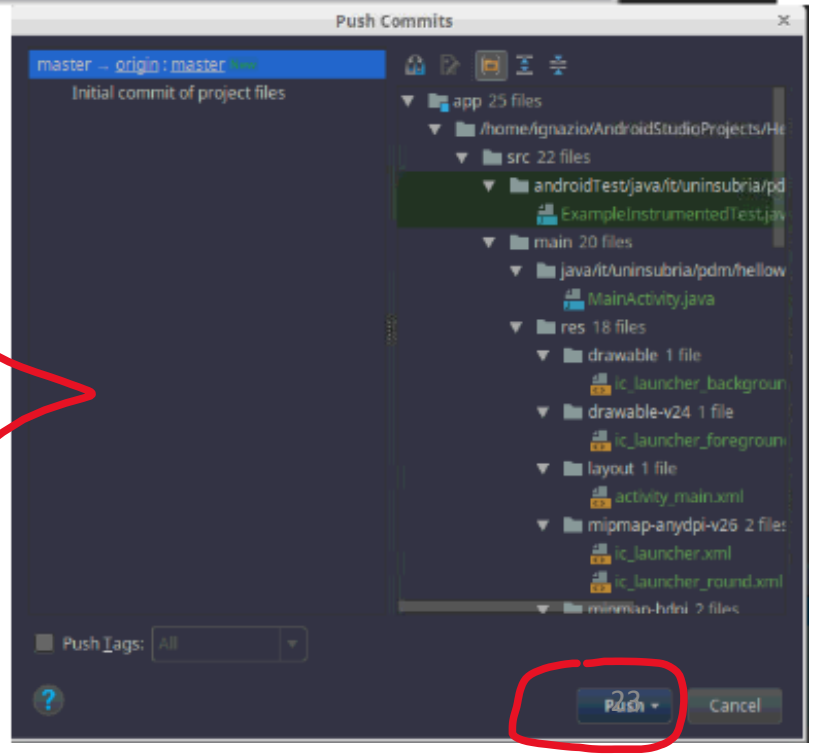
- https://gitlab.com and Sign In/Register
- Click on the New Project button
- Edit the Project Name
- Click Create project
- Copy the URL



Push to the Remote Repository

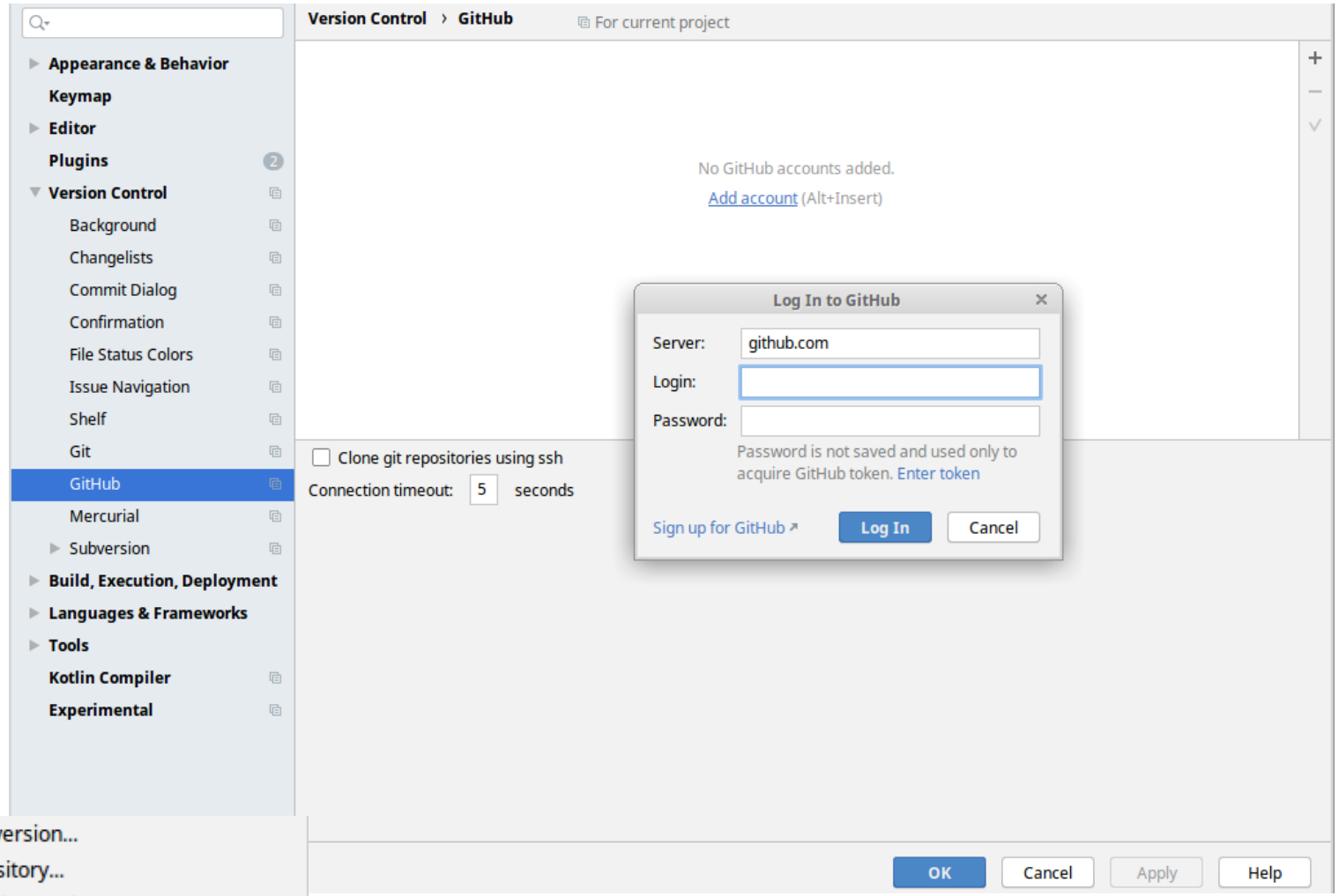


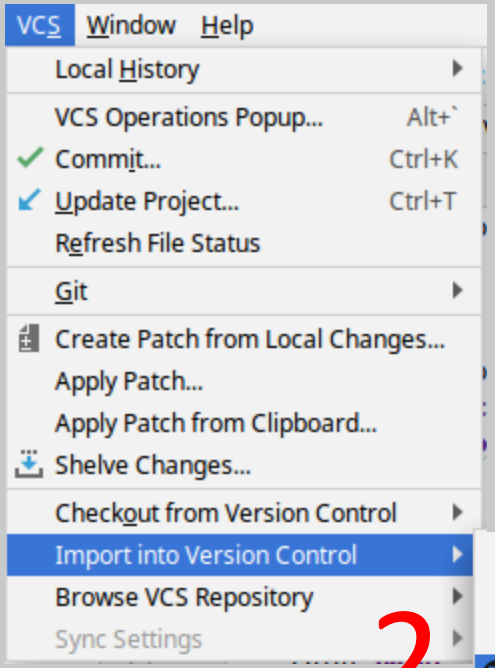
- Copy the GitLab URL
- From AndroidStudio Select the VCS → Git → Remotes
- Add a new URL
- Paste the URL of your GitLab repository
- Now you can Push all the project in the remote repository VCS → Push
- Now check the web page of your project in GitLab...



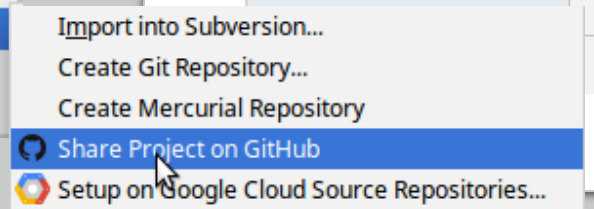
An alternative Configuration Process

1 Select: File > Settings...





2



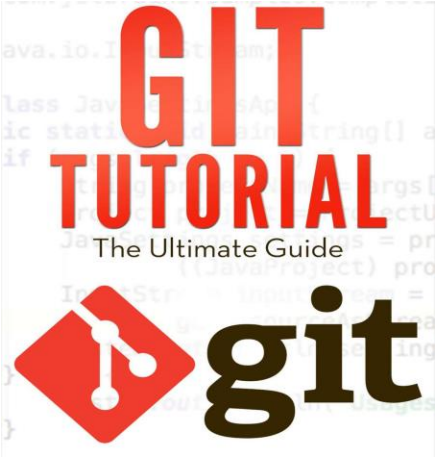
GIT – tutorial

- Some exercises about using GIT
- By Nicola Landro
  PhD student

La potenza di git

Nicola Landro Follow

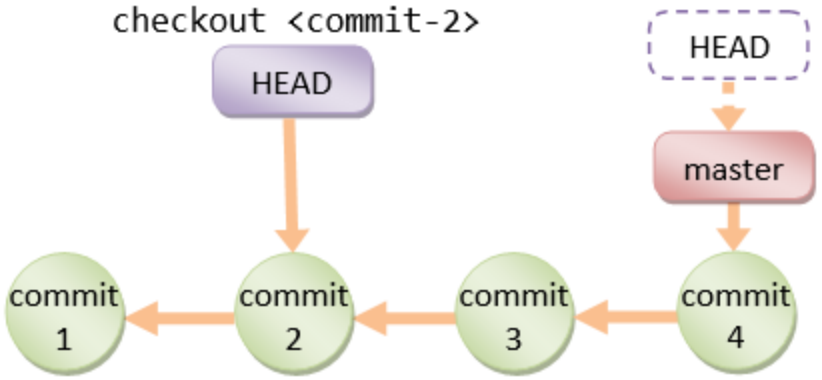
Apr 6, 2019 · 6 min read



https://medium.com/@coladibriatico94/la-potenza-di-git-7494a137b925

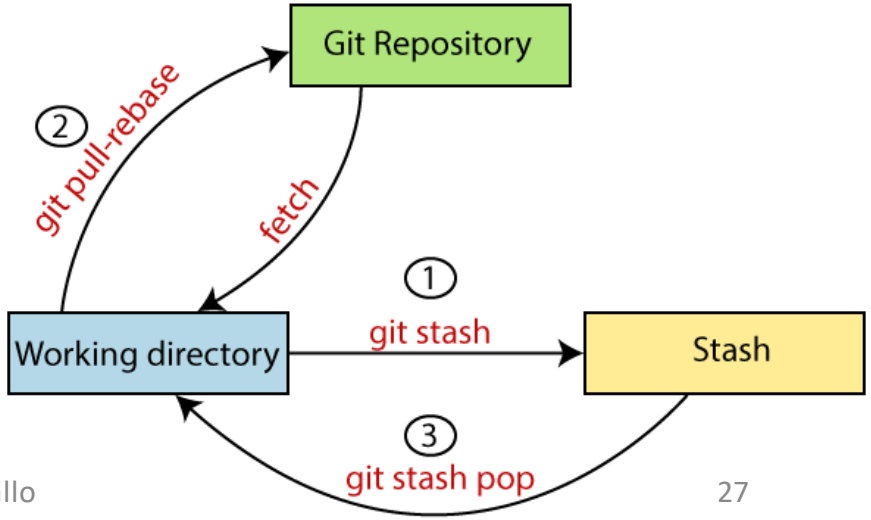
Git <HEAD>

- Whenever a [commit] is performed on a project, the HEAD is the reference to the current commit of your branch.
- Esercizio: replichiamo questa situazione in figura
  1. Aggiungiamo una riga in MainActivity.kt e facciamo
     `git add .; git commit -m "msg2"`
  2. Aggiungiamo una nuova riga e facciamo
     `git add .; git commit -m "msg3"`
  3. Aggiungiamo una nuova riga e facciamo
     `git add .; git commit -m "msg4"`
  4. `git log` # Ogni commit ha un id
  5. `git checkout <id commit "msg2">`
  6. Per tornare all'ultima versione del codice
     `git checkout master`



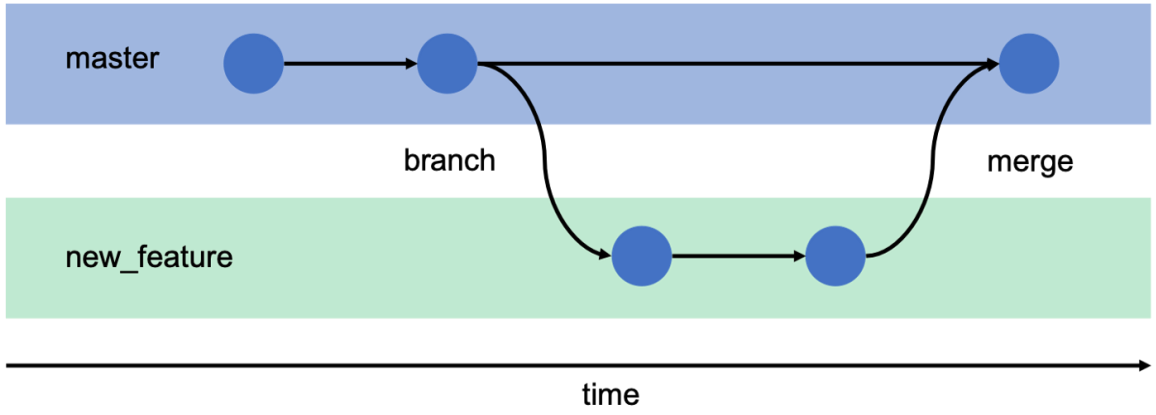
Git stash

- Git temporarily saves your data safely <u>without committing</u>.
- Esercizio:
  1. Cancelliamo le ultime modifiche in MainActivity.kt e facciamo
     `git status`
  2. Annulliamo le modifiche con
     `git stash`
  3. La modifica eliminata con lo stash può essere recuperata.
     `git stash pop`
     `git status`



Git branch

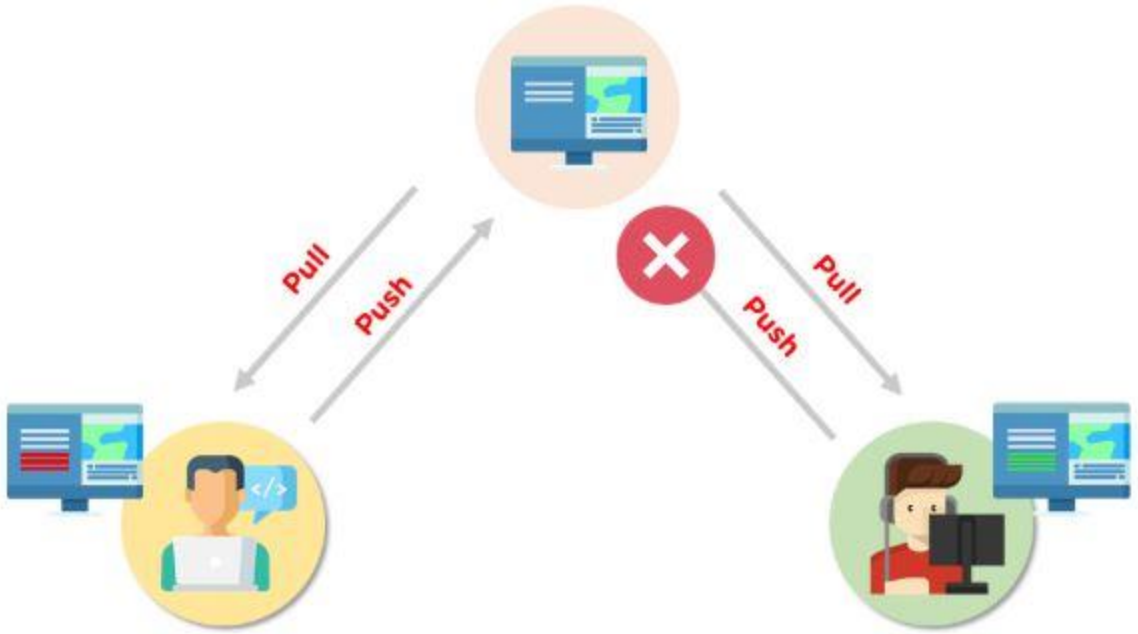
- A branch represents an independent line of development.
- Esercizio:
  1. Creiamo un nuovo branch
     ```
     git checkout -b experiment1
     ```
  2. Facciamo un paio di modifiche seguite dal commit
     ```
     git add .; git commit -m "msg4 exp1"
     ```
  3. Facciamo il merge del branch con il master.
     ```
     git checkout master
     git merge experiment1
     ```




MERGE CONFLICT?
WHY SHOULD THAT MATTER?

What is a Git Merge Conflict?

- A merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits.
- Git can merge the changes automatically only if the commits are on different lines or branches.



Git Merge Conflict example

```
println("Ciao, mondo!");
```

```
println("Ciao, mondo! Tutto bene");
```

```
println("Ciao, tutto bene");
```

```
<<<<<<< Updated upstream
println("Ciao, tutto bene?");
=======
println("Ciao, mondo! Tutto bene?");
>>>>>>> Stashed changes
```

- git add -A
- git commit -m "first commit"
- git stash
- git add -A
- git commit -m "modified line"
- git stash pop
- git diff
- git add text.java
- git commit -m "merged conflicts"

How to Resolve Merge Conflicts in Git?

- The easiest way to resolve a conflicted file is to open it and make any necessary changes
- After editing the file, we can use the git add a command to stage the new merged content
- The final step is to create a new commit with the help of the git commit command
- Git will create a new merge commit to finalize the merge

Resolve Merge Conflicts: exercise

- Create un file di testo e aggiungetelo sia al repo locale che remoto
- Modificate una riga esistente del file in remoto da github
- Fate una pull
- Risolete il conflitto