Introduction to GIT

Dott. Ing. **Marco Rabozzi**
Politecnico di Milano
Email: marco.rabozzi@polimi.it

Credits: Jeffrey Leek - Johns Bloomberg School of Public Health

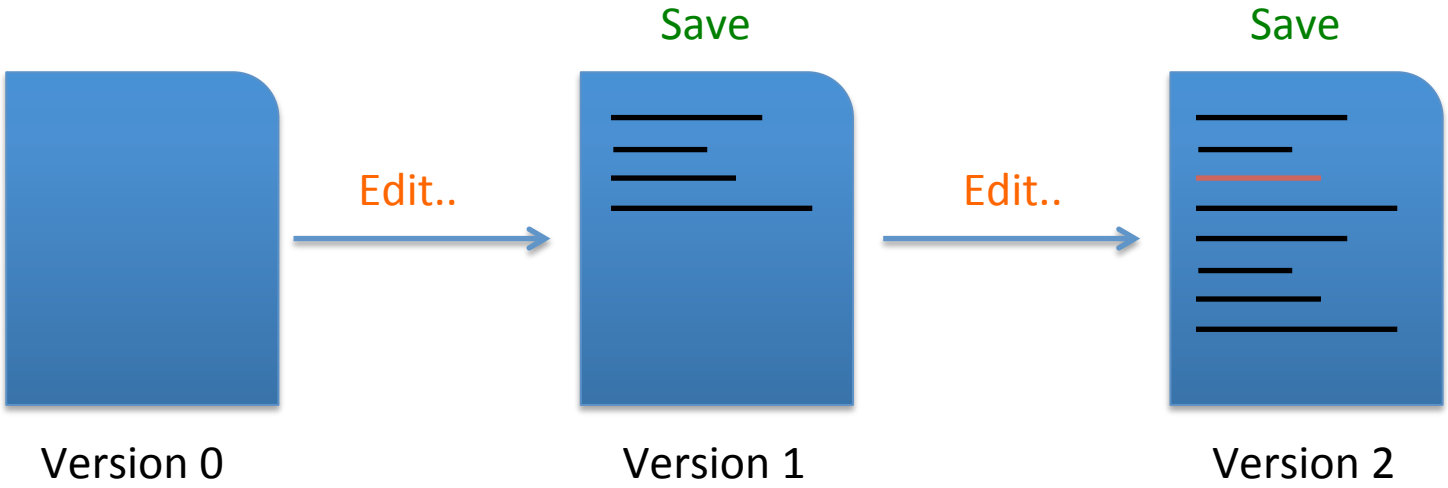
First of all…

- Download git from: http://git-scm.com/downloads

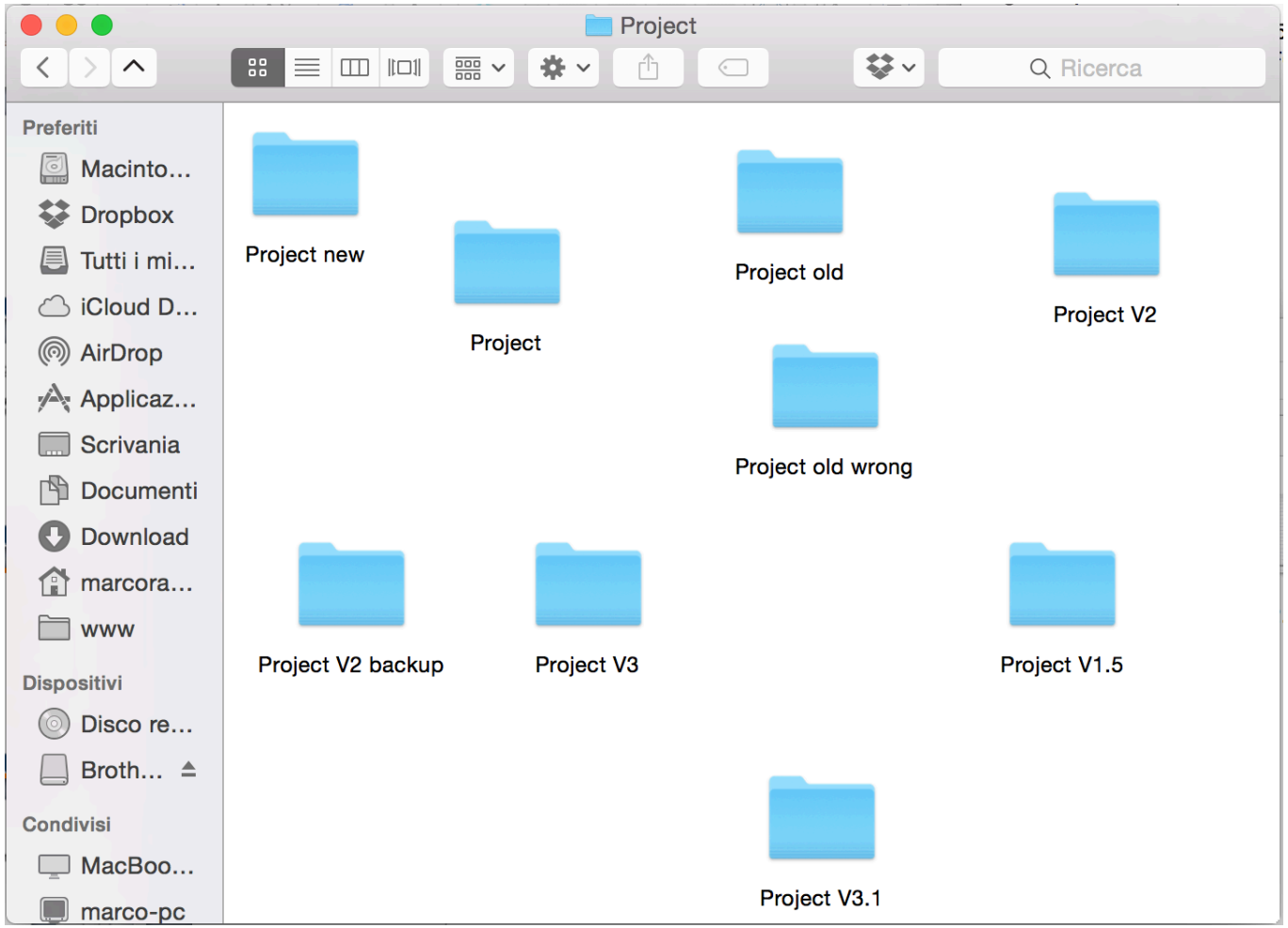
- These slides are available at: https://github.com/marcorabo/brief-git-course.git

What is a Version Control System?

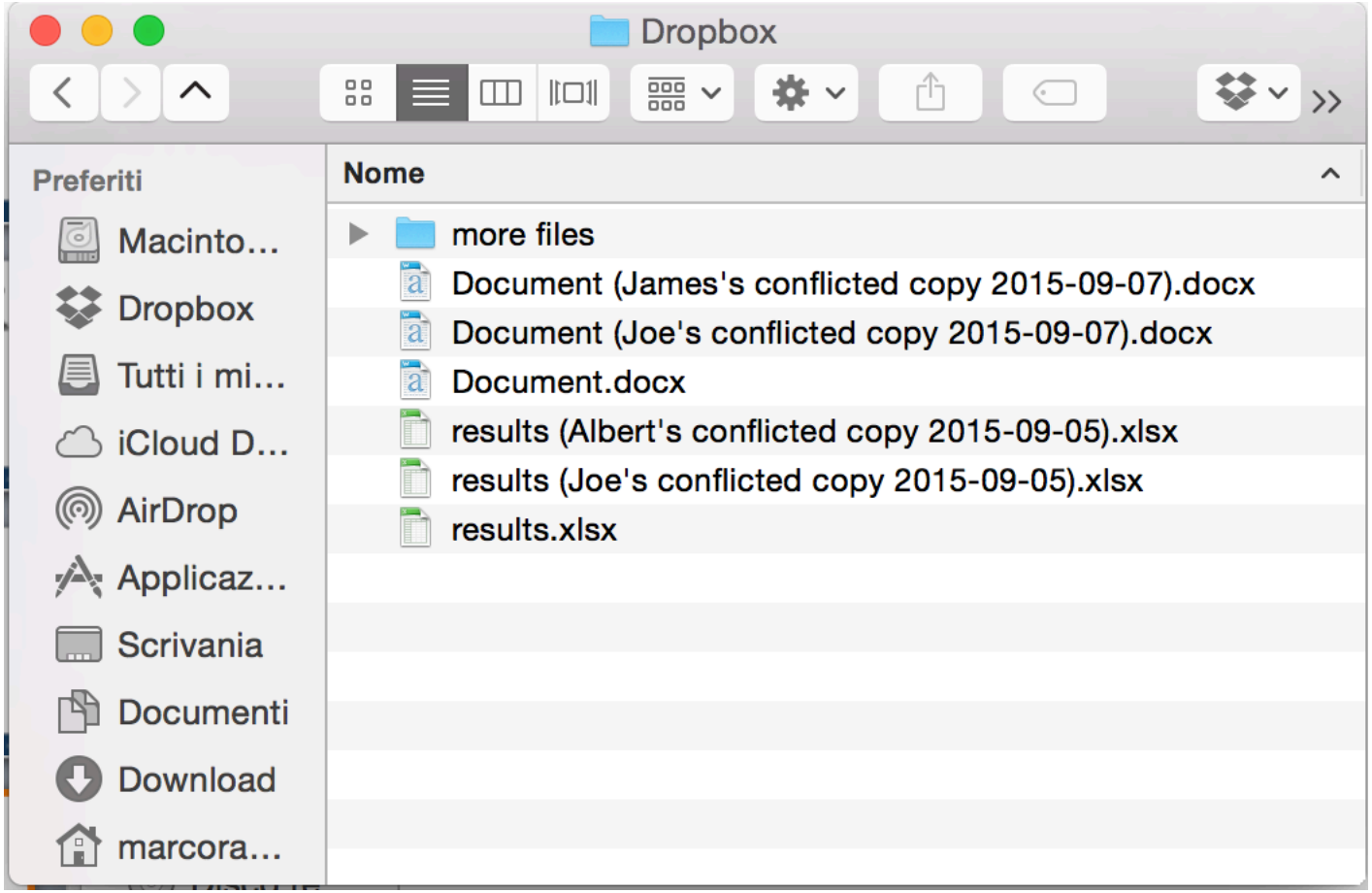
- A Version Control System (VCS) is a software that **records changes** to a set of **files** over time so that you can recall specific versions later



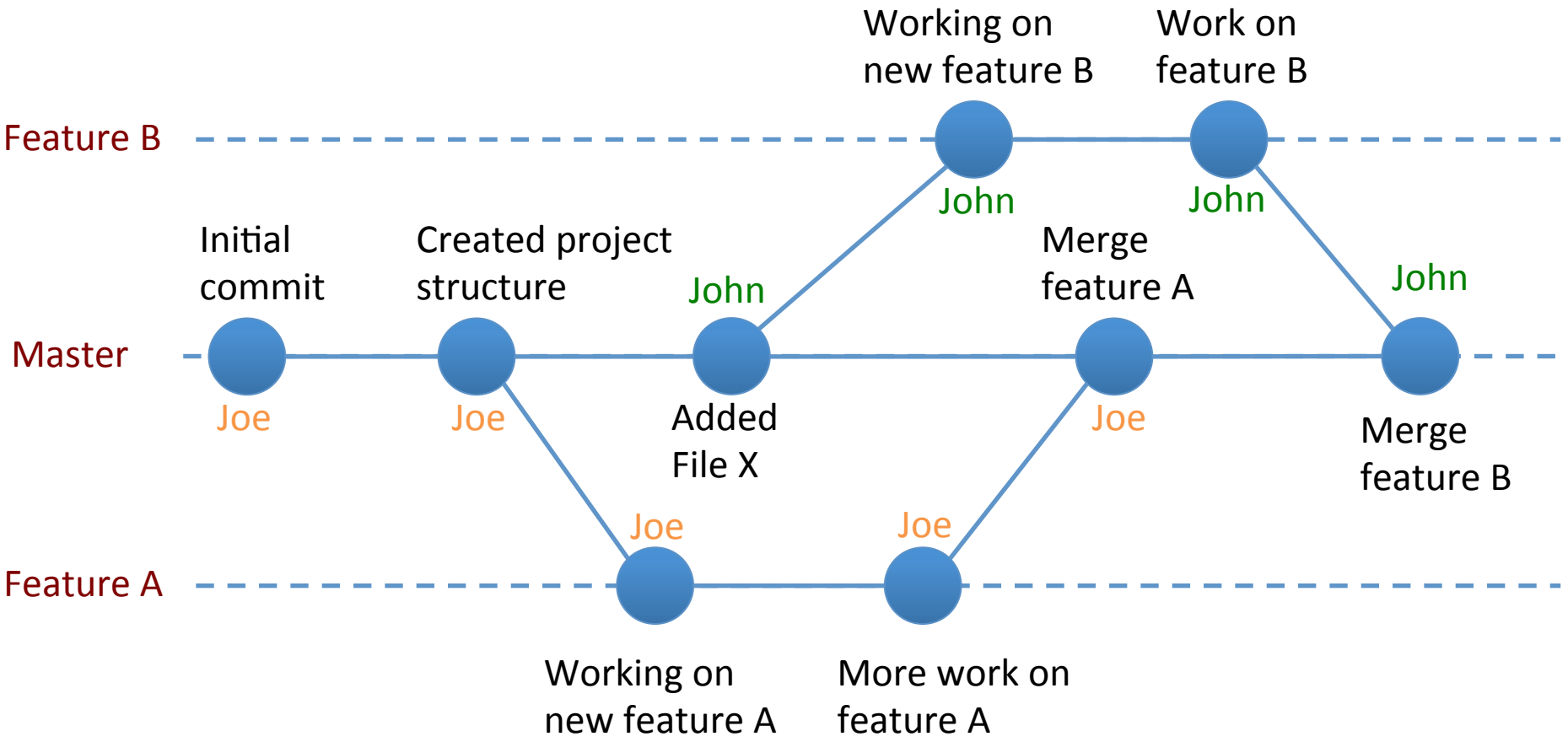
Why you need a VCS?



Why you need a VCS?



Version Control System



Version Control System

- Allows multiple users to collaborate on a project
- Keeps track of changes made by different users over time
- Example of VCS:
  - GIT
  - SVN
  - Mercurial

What is GIT?

http://git-scm.com

"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"

About GIT

- Created by the same people who developed Linux
- The most popular implementation of version control today
- Everything is stored in local repositories on your computer
- Operated from the command line

http://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git

When to use GIT

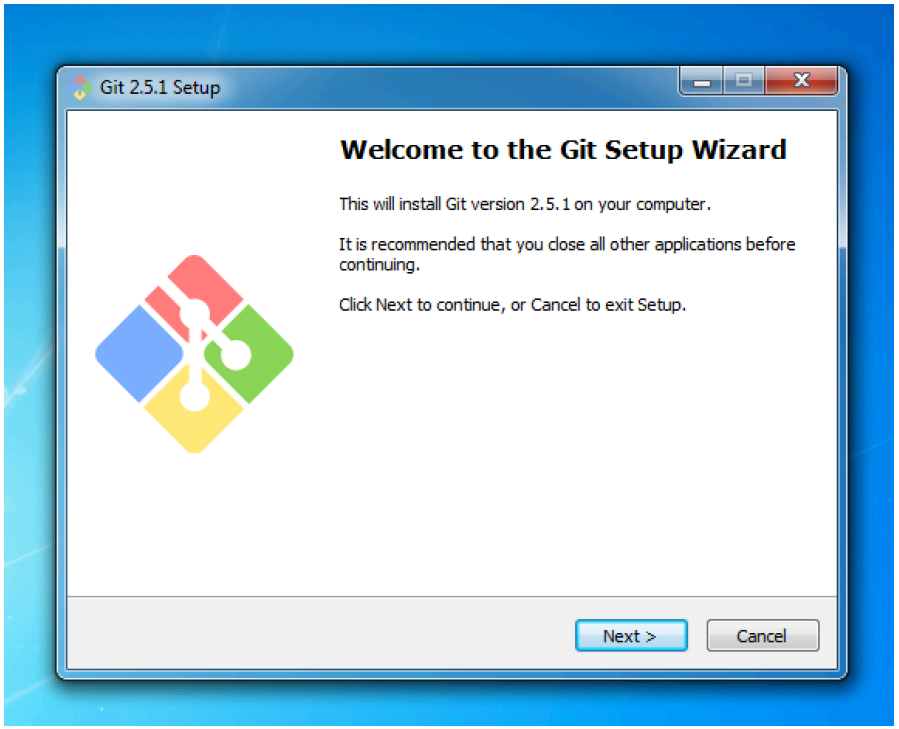
- GIT is powerful at versioning text files:
  - C
  - C++
  - MATLAB
  - Latex
  - HTML
  - …
- Usage scenarios:
  - **Single user**: keep track of the file changes for your own project
  - **Multiple users**: allow a team to share its work for a project
  - **Open source**: allow the world to propose fixes and new features to a project

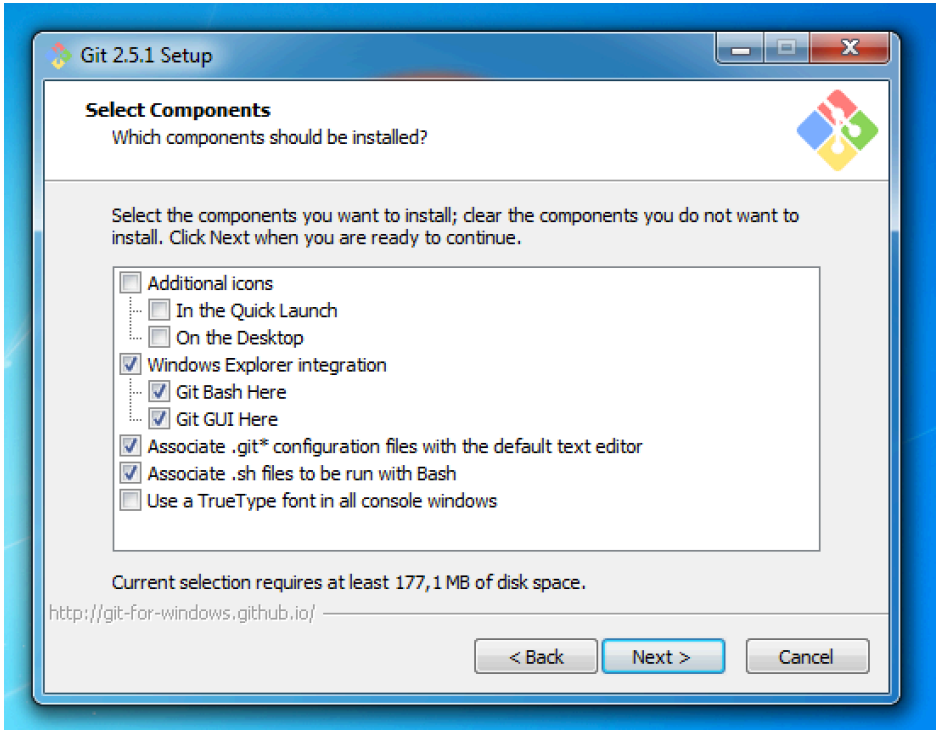
Getting Started!

- Download git from: http://git-scm.com/downloads

Setup GIT

Leave all the parameters to their default values



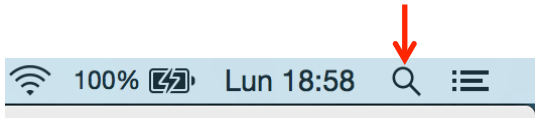


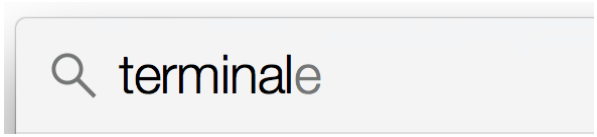
(Note: Screenshots for Windows users)

Verify your setup

- Open the Command Line Interface (CLI)
  - Windows: search “Git Bash” from the Start Menu

  - Mac OS X: search “terminal” from spotlight search

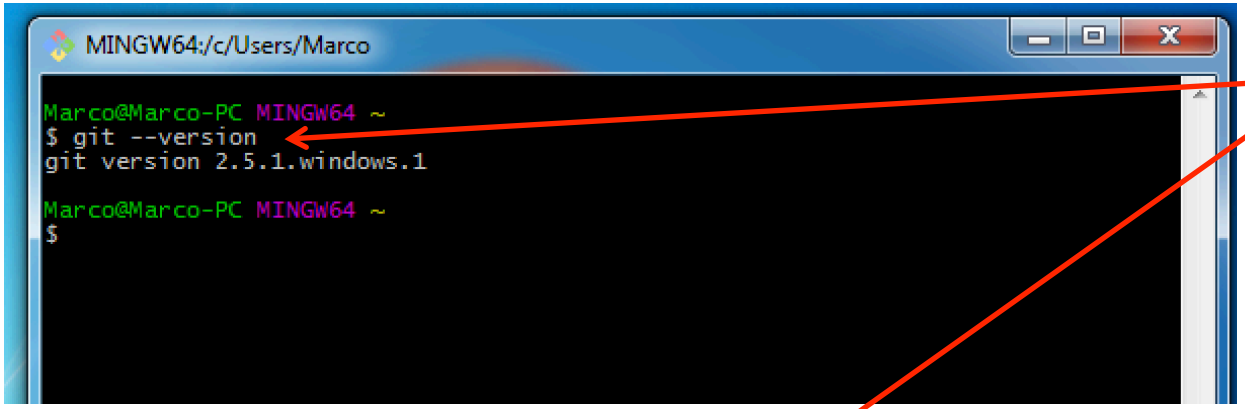


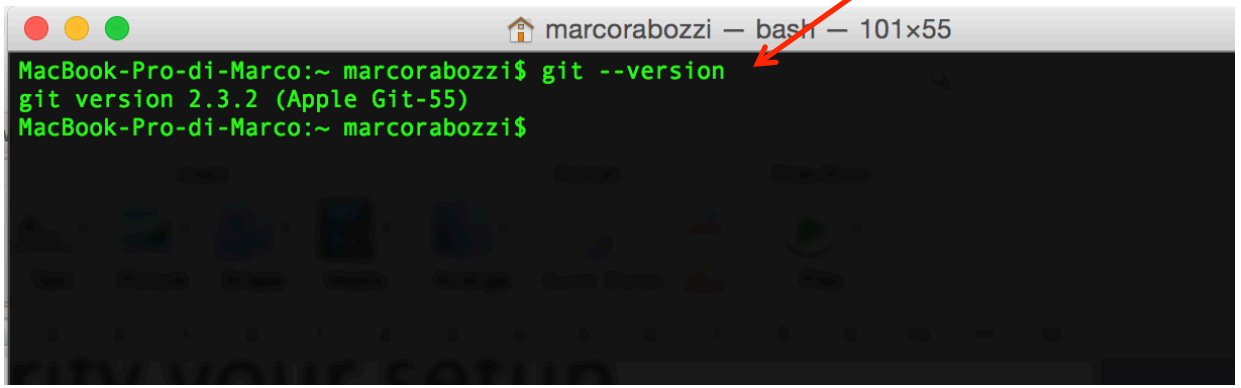


  - Linux (Ubuntu): search “terminal” from dash

Verify your setup

- Check from the CLI that GIT is installed





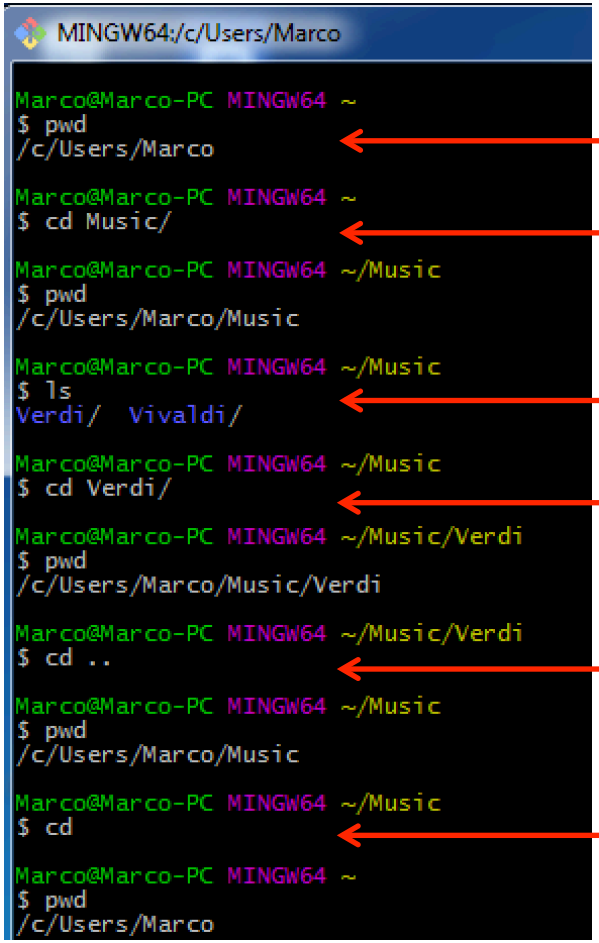
Run the command:
`git --version`

The installed version of GIT should be printed

Command line interface (CLI)

- Useful commands
  - `pwd`
  - `cd`
  - `ls`
  - `touch`
  - `mkdir`
  - `cp`
  - `rm`
  - `mv`
- Useful tools:
  - `vim`

CLI – navigate folders



Print current directory

Move to folder “Music”

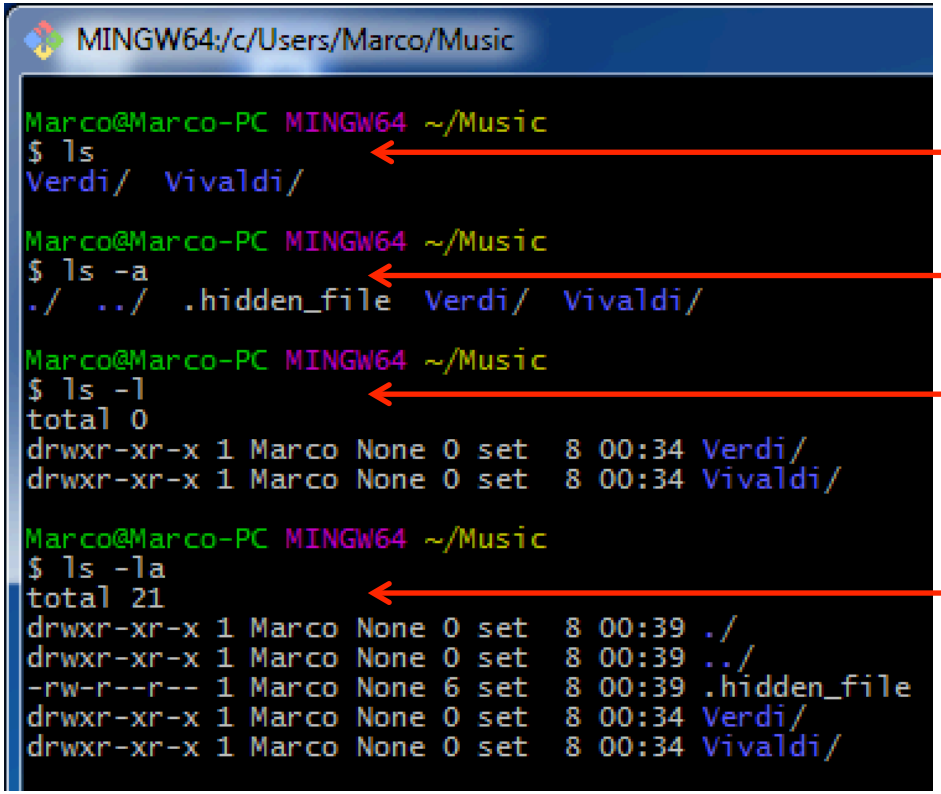
List files/folder in current directory

Move to folder “Verdi”

Go UP one level (Move to parent folder)

Go to home directory

CLI – list files



List files/folders in current directory

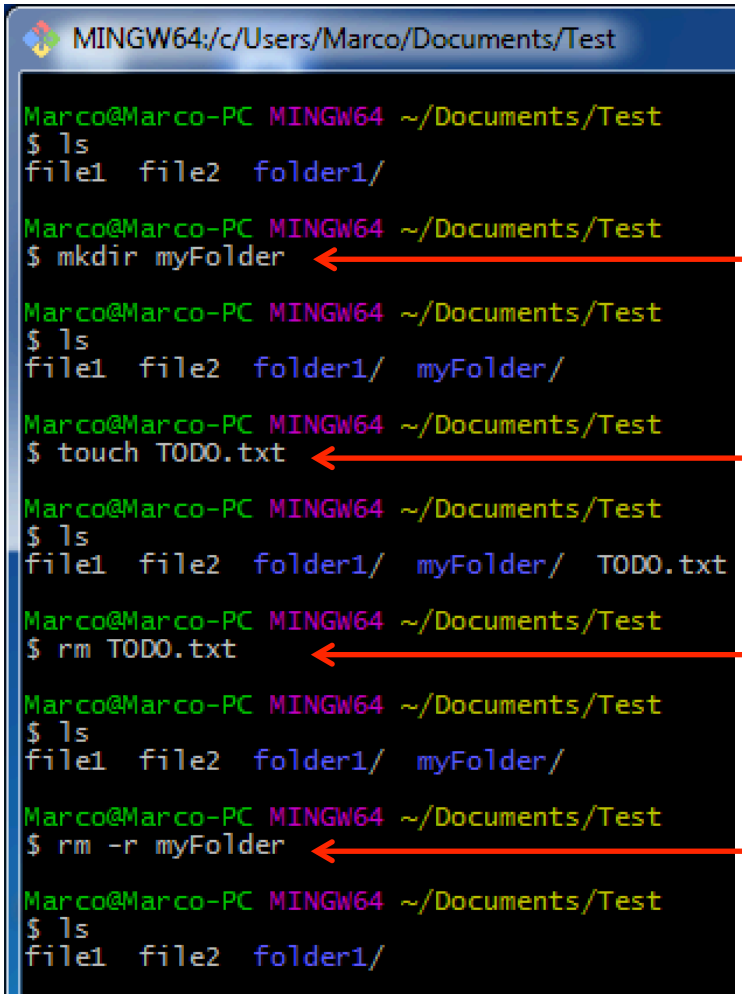
List all files/folders including hidden files

List files/folders providing more details

List all files/folder with details including hidden files

NOTE: `ls -la` == `ls -al`

CLI – create/delete files and folders



Create folder “myFolder” in current directory

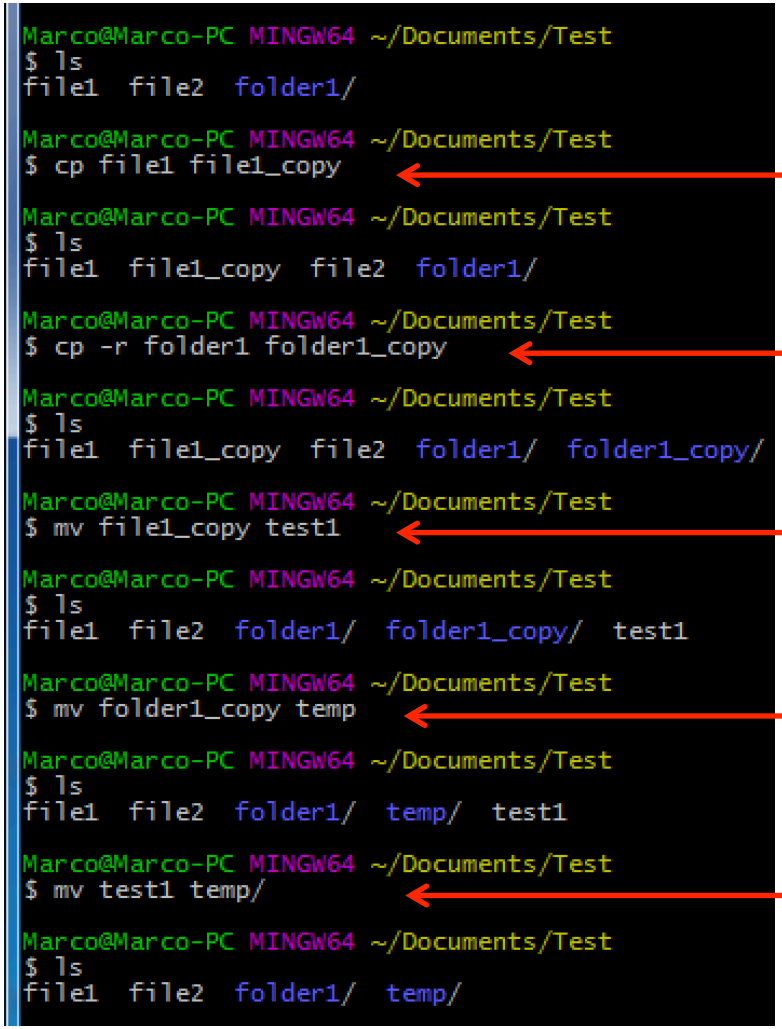
Create file “TODO.txt” in current directory

Remove file “TODO.txt”

Remove folder “myFolder”

BE CAREFUL There is no undo for `rm`!

CLI – copy/move files and folders



Make a copy of “file1” called “file1_copy”

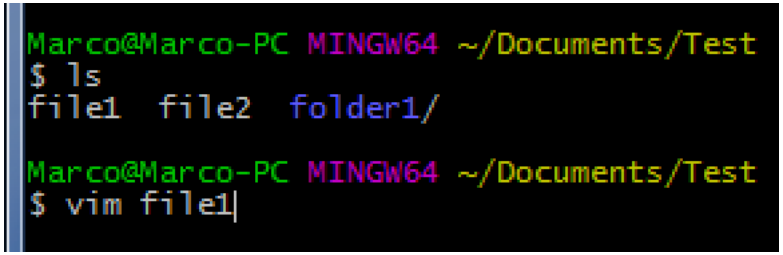
Make a copy of “folder1” called “folder1_copy”

Rename “file1_copy” to “test1”

Rename “folder1_copy” to “temp”

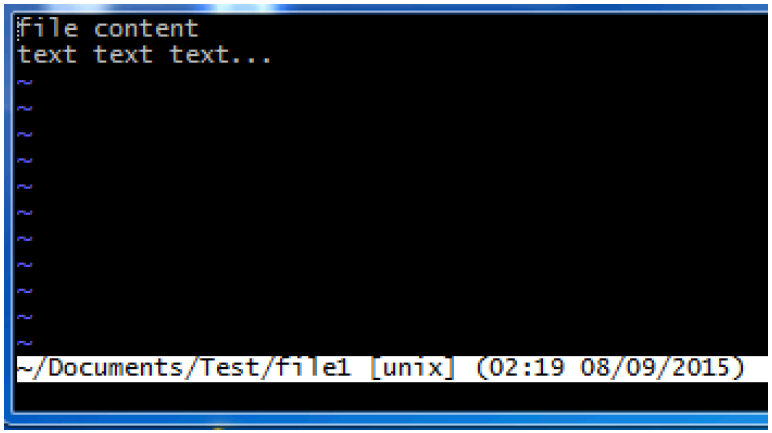
Move file “test1” in folder “temp”

VIM text editor



`vim <file>`
(enter vim)

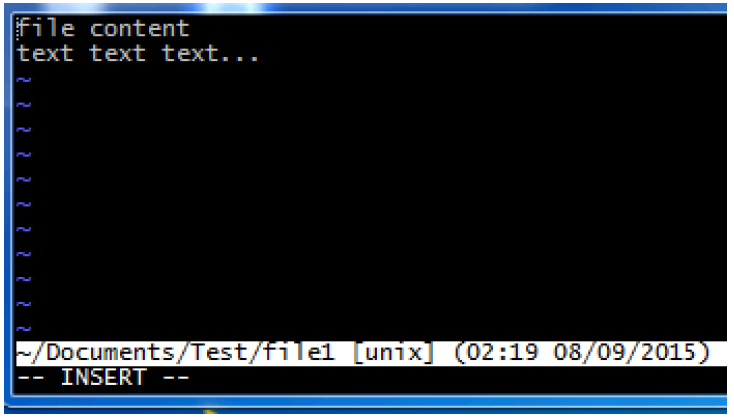
"`:wq`" (quit and save changes)
"`:q!`" (quit without saving)



Normal mode

"`i`"

ESC



Insert mode

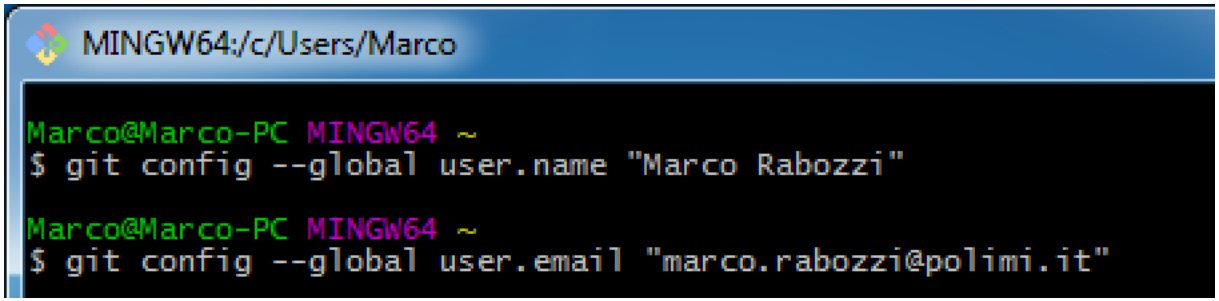
CLI - recap

- Useful commands
  - `pwd` Print current working directory
  - `cd` Change directory
  - `ls` List files and folders
  - `touch` Create a file
  - `mkdir` Create a folder
  - `cp` Copy file/folder
  - `rm` Remove file/folder
  - `mv` Move file/folder (e.g.: cat and paste)
- Useful tools:
  - `vim` Open a text editor

GIT configuration

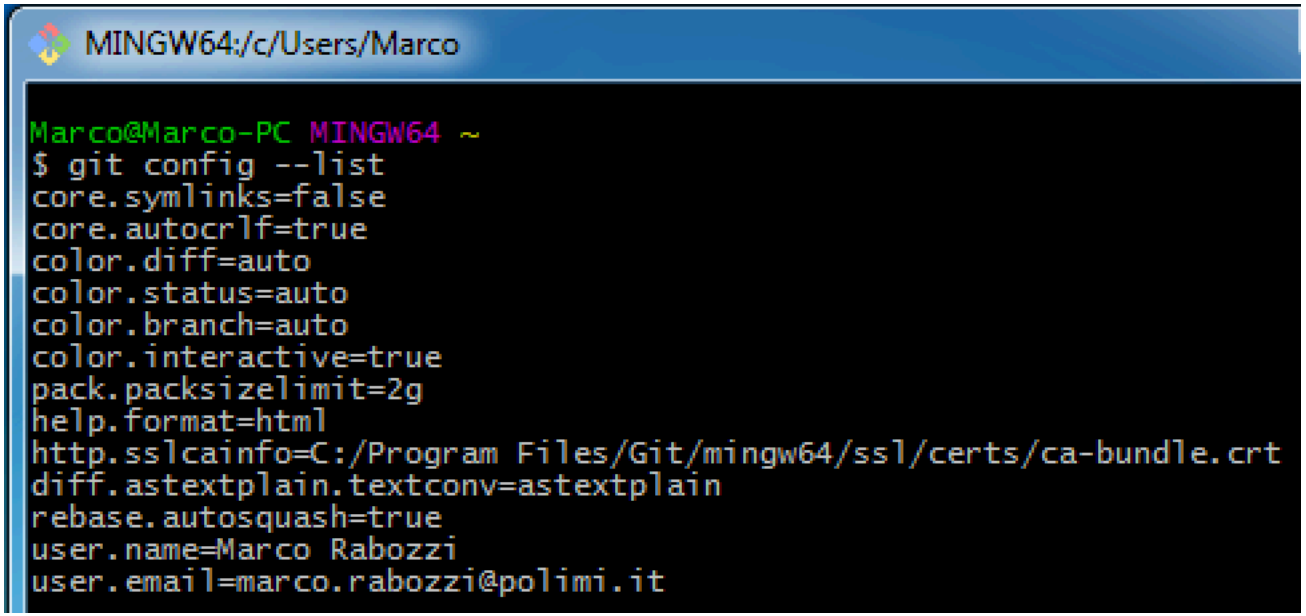
- Each commit to a Git repository will be “tagged” with the name of the person who made the commit

- The following commands set your name and email:



GIT configuration

- Run the command `git config --list` to see your current configuration:



Create a project

- In order to start using GIT, we first need a folder to store our project:

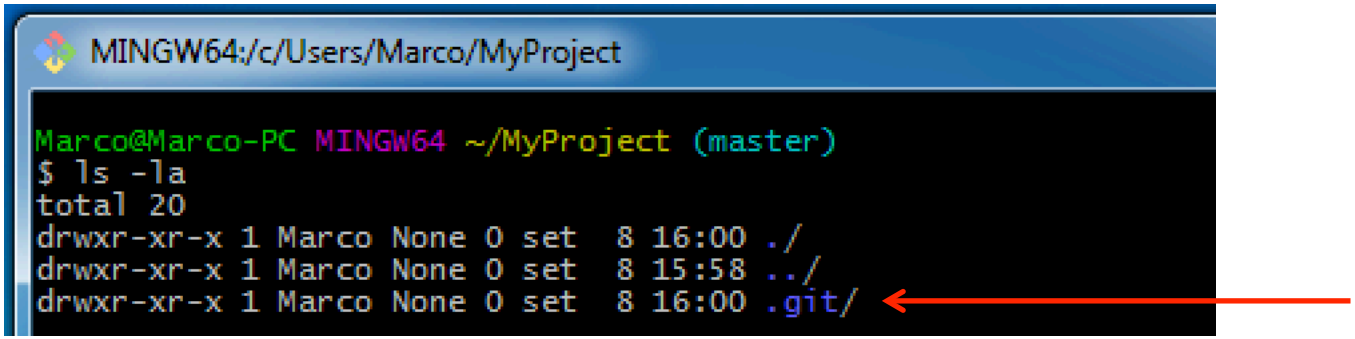
```
Marco@Marco-PC MINGW64 ~
$ mkdir MyProject
```

- To initialize a GIT ***repository*** for your project, enter the project folder and run `git init`

```
Marco@Marco-PC MINGW64 ~
$ cd MyProject/

Marco@Marco-PC MINGW64 ~/MyProject
$ git init
Initialized empty Git repository in C:/Users/Marco/MyProject/.git/

Marco@Marco-PC MINGW64 ~/MyProject (master)
$
```

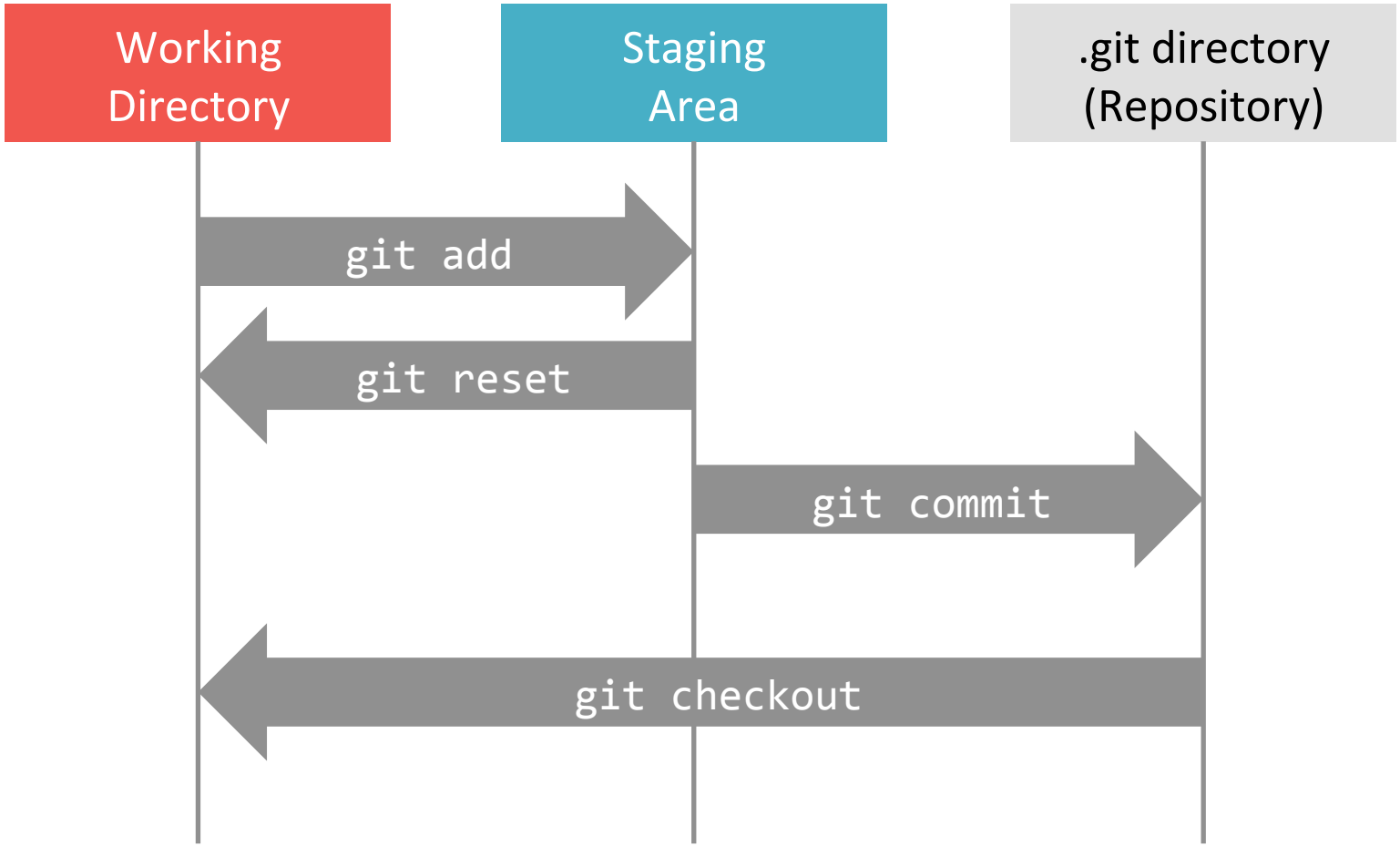
Create a project

- The repository is contained within the hidden folder ".git" in your project directory:



- This folder is used by GIT to manage your repo, it should not be modified manually

How GIT works (locally)



Towards the first commit

- Run: `git status`
  to show the current status of your repo
  (i.e.: what changed in your project since the last commit)

```
Marco@Marco-PC MINGW64 ~/MyProject (master)
$ git status
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
```

Nothing has changed

Towards the first commit

- Create a file in your project, you can create/edit the file with the editor of your choice (VIM, notepad, sublime, …)

```
Marco@Marco-PC MINGW64 ~/MyProject (master)
$ touch contributors.txt

Marco@Marco-PC MINGW64 ~/MyProject (master)
$ ls
contributors.txt
```

- Check the status of your repo:

```
Marco@Marco-PC MINGW64 ~/MyProject (master)
$ git status
On branch master

Initial commit

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

        contributors.txt

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

Adding files

- Use the command `git add <file>` to add the desired files to the staging area

```
Marco@Marco-PC MINGW64 ~/MyProject (master)
$ git add contributors.txt
```

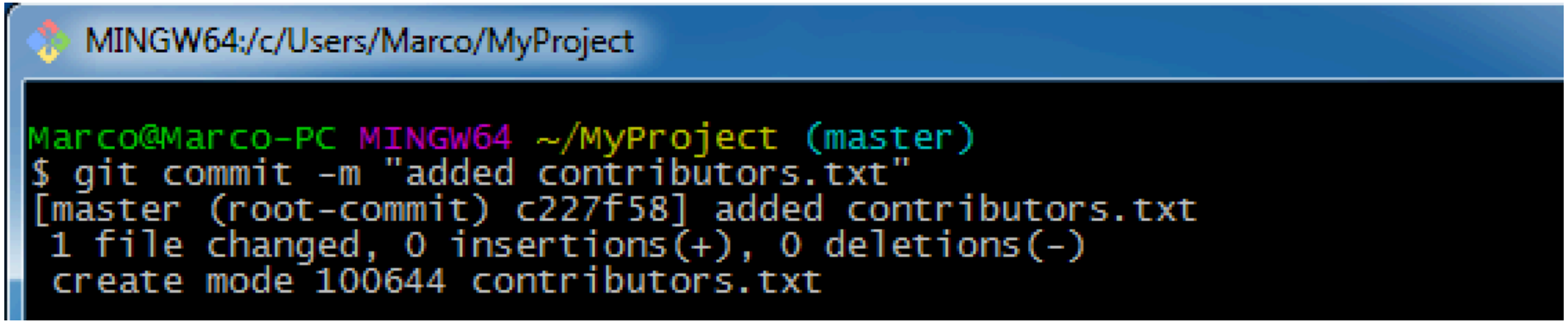
- Check the status of your repo:

```
Marco@Marco-PC MINGW64 ~/MyProject (master)
$ git status
On branch master

Initial commit

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

        new file:   contributors.txt
```

- NOTE: `git add` can be undone using `git reset`

The first commit

- The file “contributors.txt” is now in the staging area but has not been committed yet to the repo
- To commit the files in the staging area use:
  - `git commit –m “your commit message”`



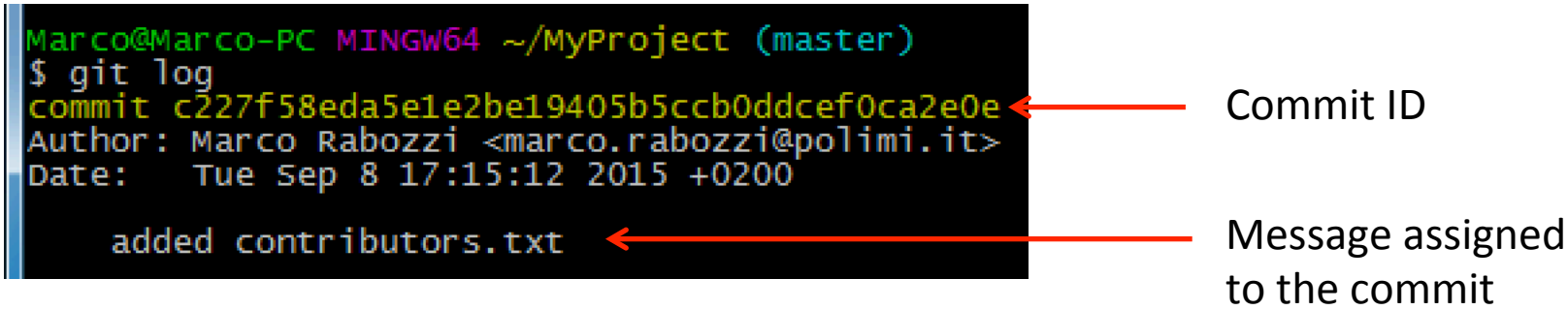
Verify the status of your repo

- `git status`

```
Marco@Marco-PC MINGW64 ~/MyProject (master)
$ git status
On branch master
nothing to commit, working directory clean
```

- `git log` (shows the commit history of your repo)

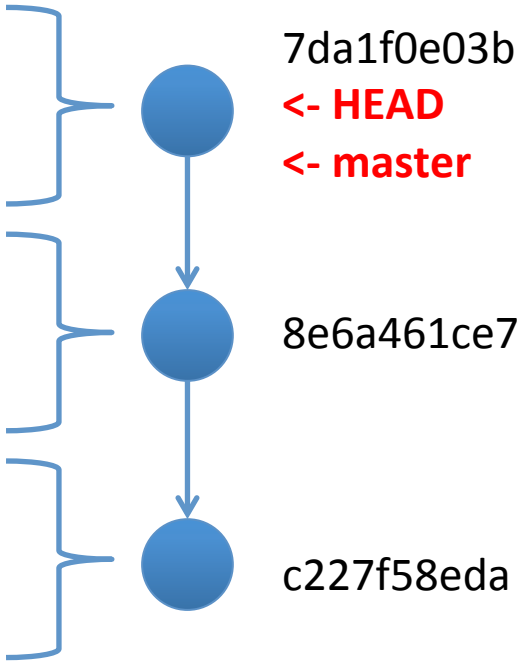
```
Marco@Marco-PC MINGW64 ~/MyProject (master)
$ git log
commit c227f58eda5e1e2be19405b5ccb0ddcef0ca2e0e
Author: Marco Rabozzi <marco.rabozzi@polimi.it>
Date:   Tue Sep 8 17:15:12 2015 +0200

    added contributors.txt
```



GIT history

– Assume that after a few commits `git log --graph` returns:

```
Marco@Marco-PC MINGW64 ~/MyProject (master)
$ git log --graph
* commit 7da1f0e03bf68794db6f4716d7acc6c2c843fe8d
| Author: Marco Rabozzi <marco.rabozzi@polimi.it>
| Date:   Tue Sep 8 19:14:34 2015 +0200
|
|     Added more contributors
|
* commit 8e6a461ce749bcfad38e2c95c43000ad5dcd4c56
| Author: Marco Rabozzi <marco.rabozzi@polimi.it>
| Date:   Tue Sep 8 19:13:21 2015 +0200
|
|     added Marco Rabozzi to contributors
|
* commit c227f58eda5e1e2be19405b5ccb0ddcef0ca2e0e
  Author: Marco Rabozzi <marco.rabozzi@polimi.it>
  Date:   Tue Sep 8 17:15:12 2015 +0200

      added contributors.txt
```



– Commits can be identified by the first 10 characters of their ID

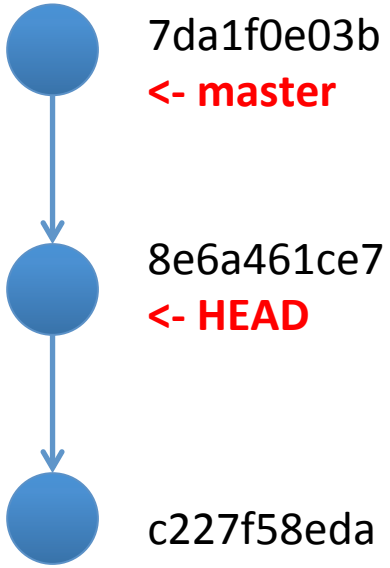
– **HEAD** is a pointer to the current commit

– **master** is a pointer to the last commit of the branch “master”

Checkout a previous commit

- If your working directory is clean you can revert using `git checkout <commit_id>`:

```
Marco@Marco-PC MINGW64 ~/MyProject (master)
$ git checkout 8e6a461ce7
Note: checking out '8e6a461ce7'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at 8e6a461... added Marco Rabozzi to contributors
```

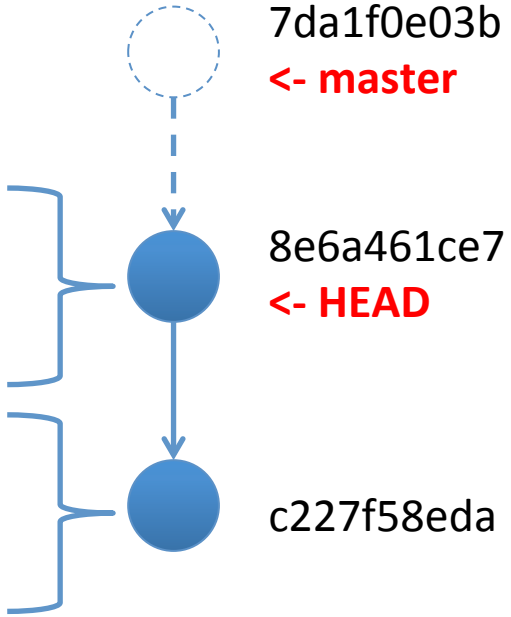


- The working directory will show the content of the files as they were at the specified commit

Checkout a previous commit

- Running `git log --graph` now shows:

```
Marco@Marco-PC MINGW64 ~/MyProject ((8e6a461...))
$ git log --graph
* commit 8e6a461ce749bcfad38e2c95c43000ad5dcd4c56
| Author: Marco Rabozzi <marco.rabozzi@polimi.it>
| Date:   Tue Sep 8 19:13:21 2015 +0200
|
|     added Marco Rabozzi to contributors
|
* commit c227f58eda5e1e2be19405b5ccb0ddcef0ca2e0e
  Author: Marco Rabozzi <marco.rabozzi@polimi.it>
  Date:   Tue Sep 8 17:15:12 2015 +0200

      added contributors.txt
```

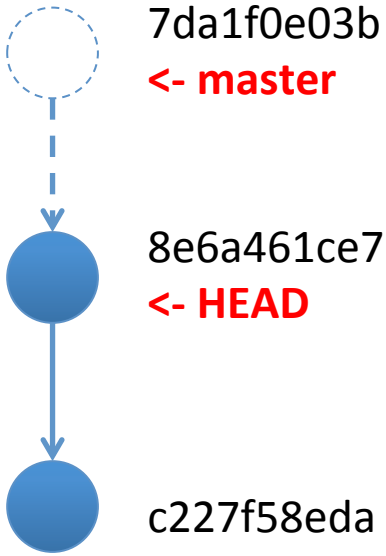


Checkout a previous commit

- Running `git status` shows:

```
MINGW64:/c/Users/Marco/MyProject

Marco@Marco-PC MINGW64 ~/MyProject ((8e6a461...))
$ git status
HEAD detached at 8e6a461
nothing to commit, working directory clean
```

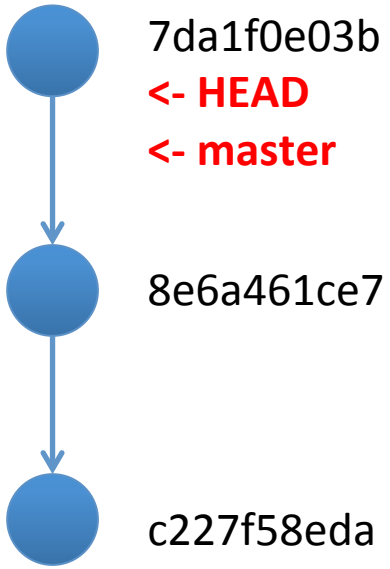
HEAD is not pointing to any referenced commit (e.g. “master”)



Return to the last commit

- To go to the last commit of a branch run:
  `git checkout <branch_name>`

```
Marco@Marco-PC MINGW64 ~/MyProject ((8e6a461...))
$ git checkout master
Previous HEAD position was 8e6a461... added Marco Rabozzi to
contributors
Switched to branch 'master'
```



How to check last changes

- To show what are the changes not committed yet run:
  - `git diff HEAD`
- To show what are the changes not committed yet for a specific file run:
  - `git diff HEAD -- <file>`

How to undo things

- To remove a file form the staging (i.e. undo a git add <file>) run:
  - `git reset <file>`

- If you want to discard the changes made to a file since the last commit (or since the last add) run:
  - `git checkout -- <file>`

WARNING! If undo changes that are not committed yet, these are lost and cannot be recovered

GIT introduction recap

- `git --version`
- `git config –-global user.name`
- `git config –-global user.email`
- `git config –-list`
- `git init`
- `git status`
- `git add <file>`
- `git reset <file>`
- `git commit -m “<message>”`
- `git log`
- `git log --graph`
- `git checkout <commit_id>`
- `git checkout <branch_name>`
- `git checkout -- <file>`
- `git diff HEAD`
- `git diff HEAD -- <file>`

Useful links

- Git Book:
  - http://git-scm.com/doc
- Git Tutorial:
  - https://www.atlassian.com/git/tutorials/
- Video lectures (check course Week 2):
  - https://class.coursera.org/datascitoolbox-032/lecture