UNIVERSITÀ DEGLI STUDI
DI MILANO

Sviluppo Software in Gruppi di Lavoro Complesso

prof. Carlo Bellettini

prof. Mattia Monga

a.a. 2020-21

Spunto

GITHUB TORWALDS

SPIEGAZIONE

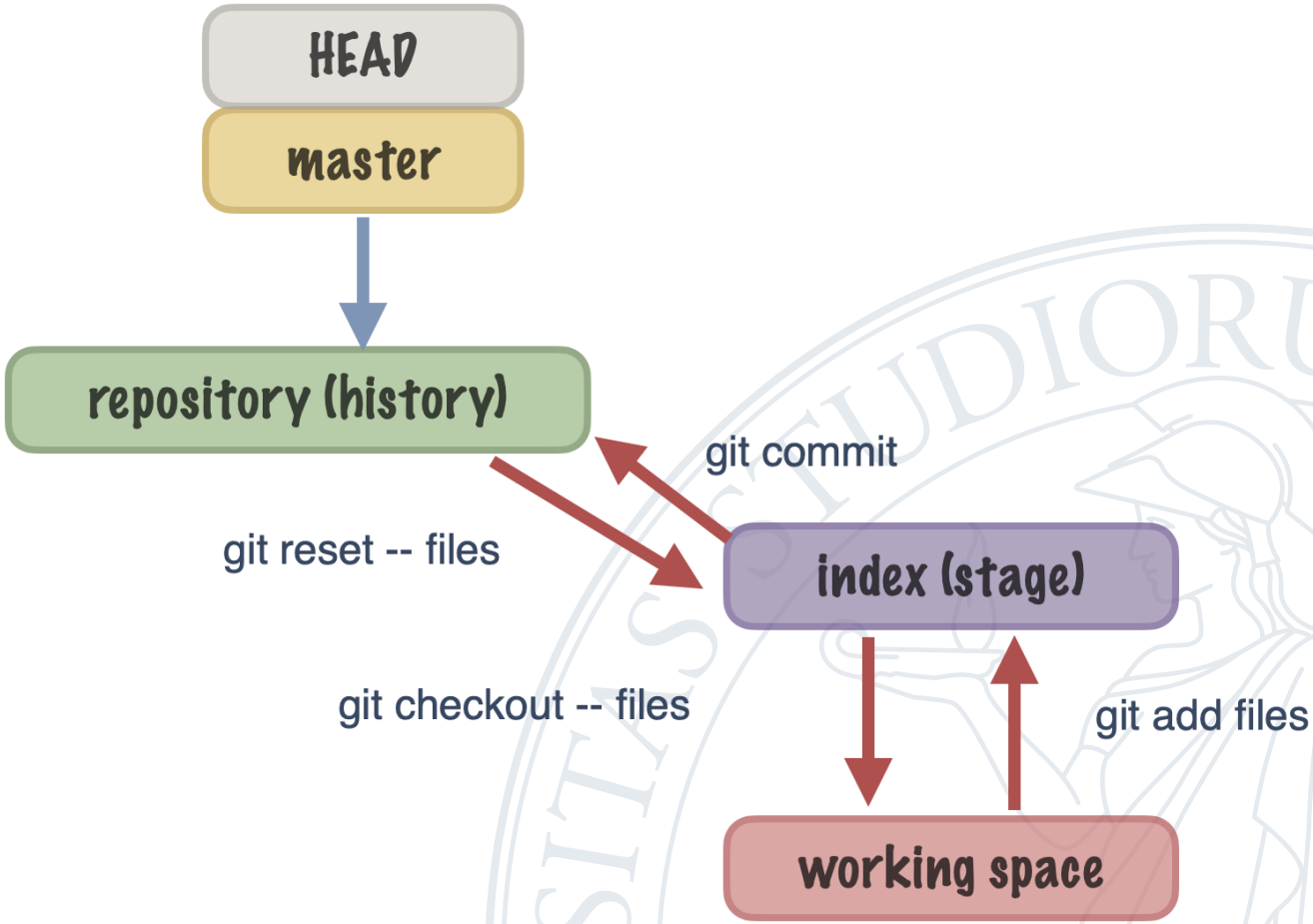
Humor

If that doesn't fix it, git.txt contains the phone number of a friend of mine who understands git. Just wait through a few minutes of 'It's really pretty simple, just think of branches as...' and eventually you'll learn the commands that will fix everything.

Formato degli objects

```ruby
def put_raw_object(content, type)
  size = content.length.to_s

  header = "#{type} #{size}\0" # type(space)size(null byte)
  store = header + content

  sha1 = Digest::SHA1.hexdigest(store)
  path = @git_dir + '/' + sha1[0...2] + '/' + sha1[2..40]

  if !File.exists?(path)
    content = Zlib::Deflate.deflate(store)

    FileUtils.mkdir_p(@directory+'/'+sha1[0...2])
    File.open(path, 'w') do |f|
      f.write content
    end
  end
  return sha1
end
```

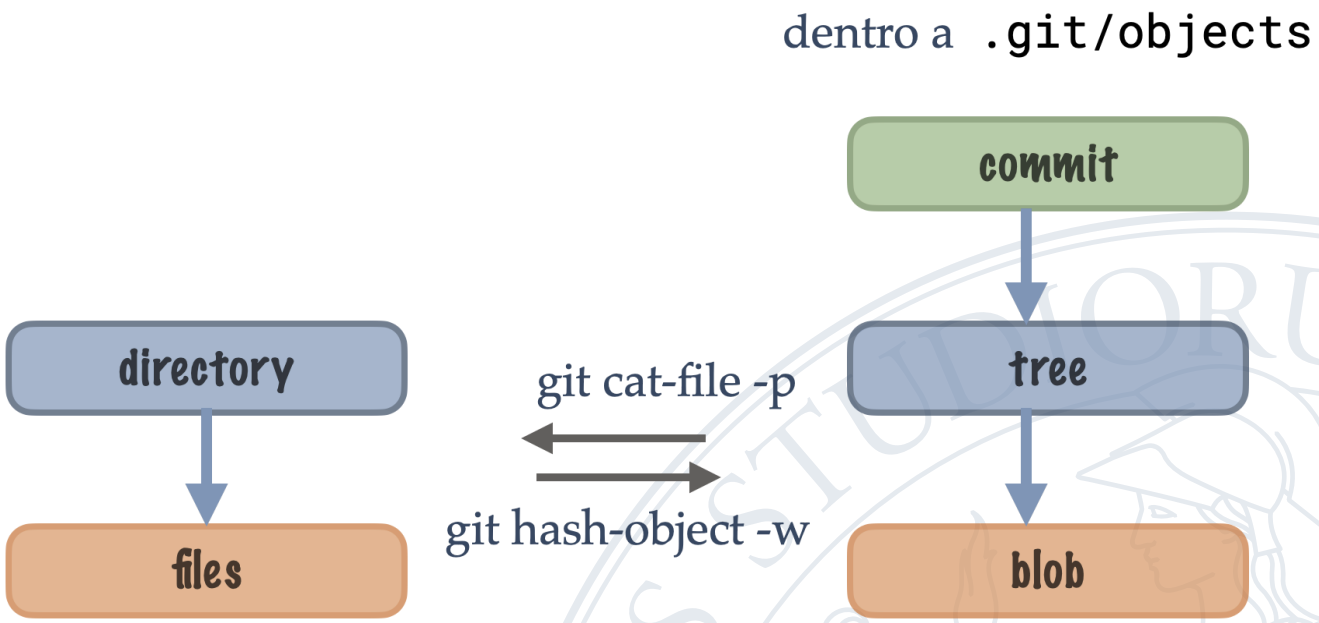
Index


HEAD
master
repository (history)
git commit
git reset -- files
index (stage)
git checkout -- files
git add files
working space

Git data: index

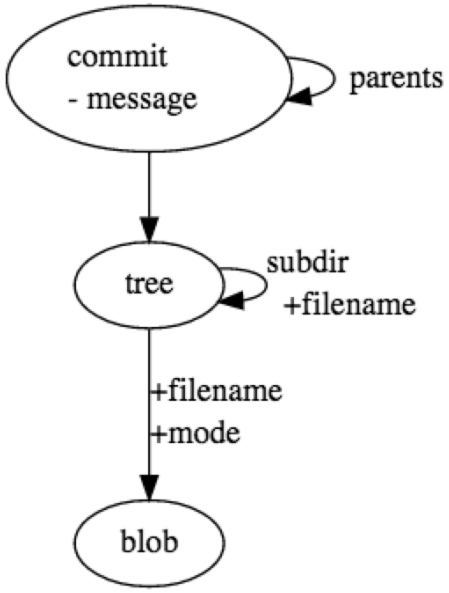
- contiene tutte le informazioni per potere fare il commit
- ogni volta che viene fatto un git -add vengono creati gli object relativi a tali snapshot

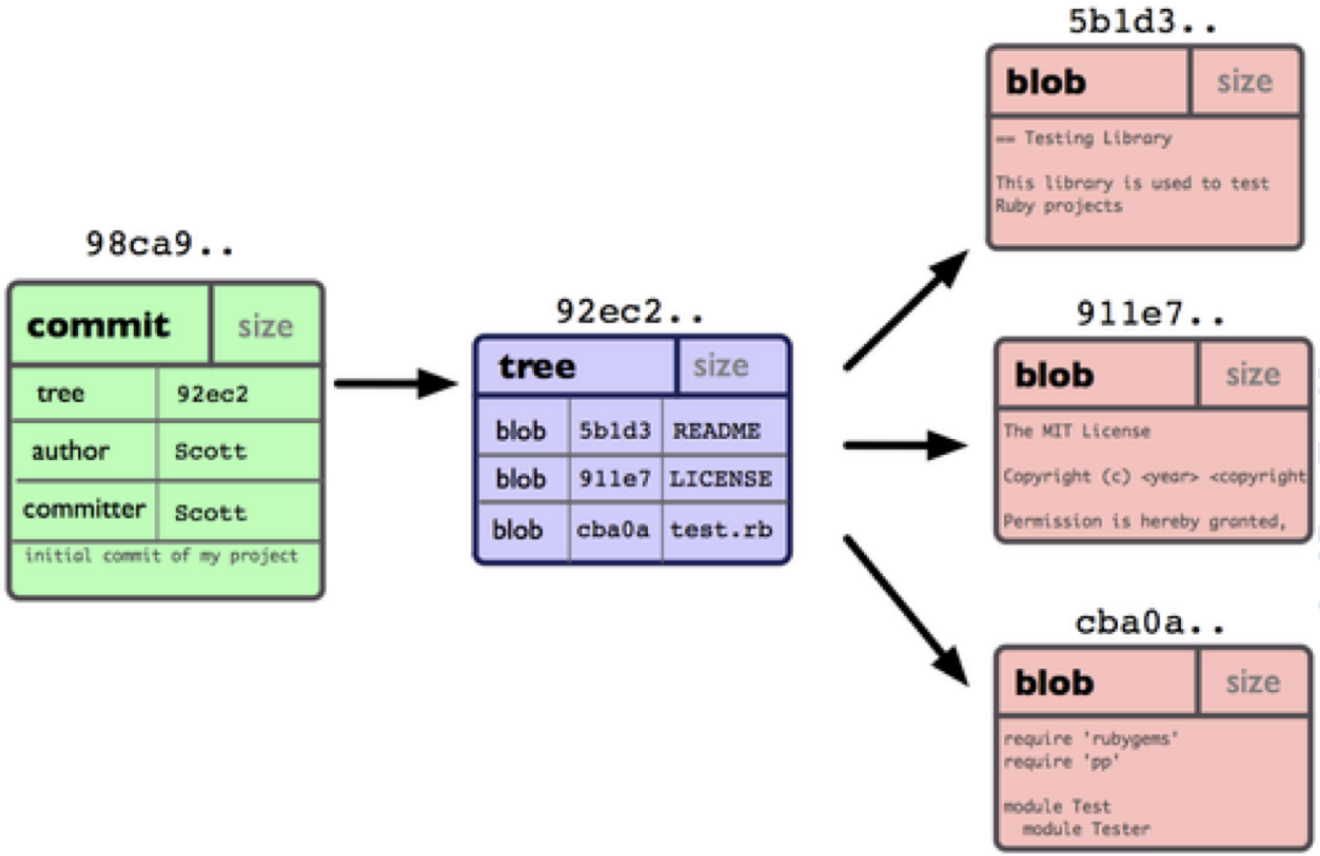
Working space vs .git/...



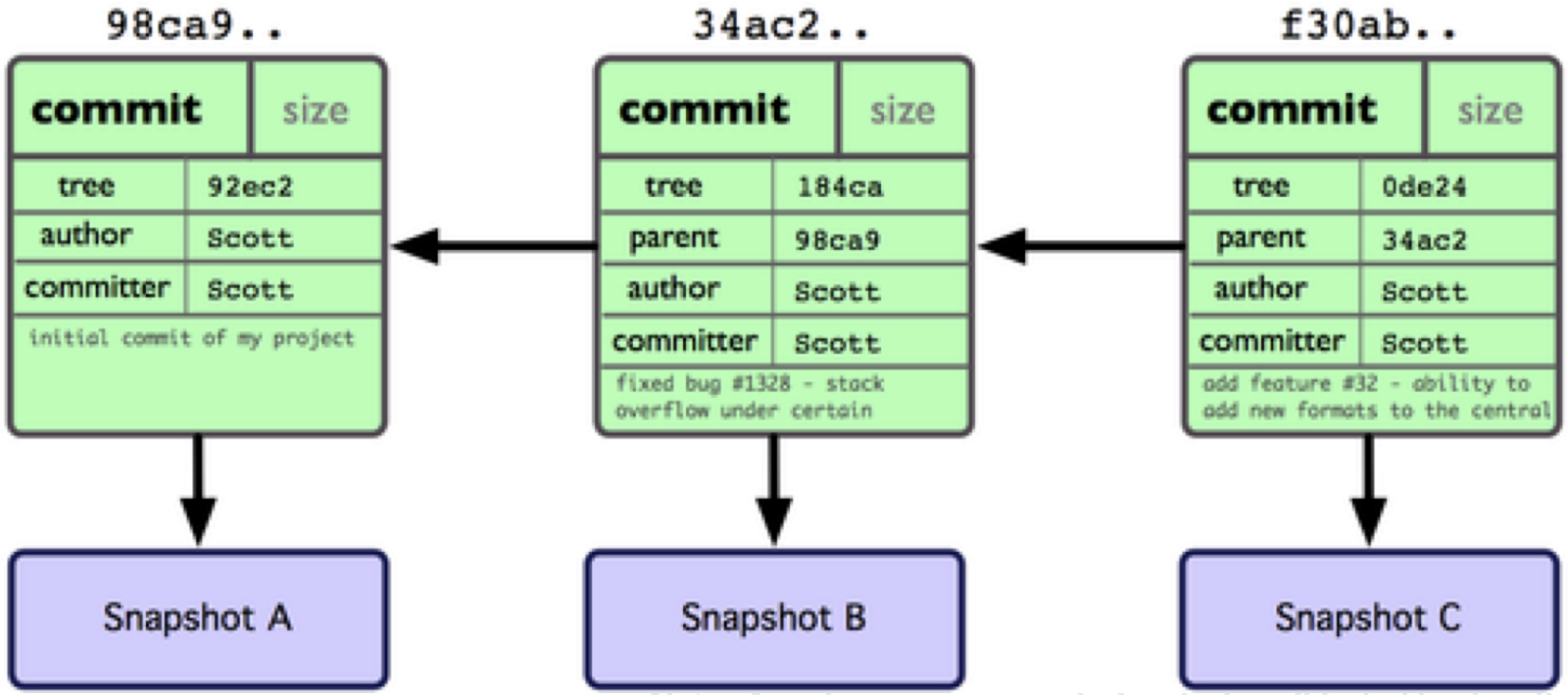
http://eagain.net/articles/git-for-computer-scientists
http://arokem.github.io/2013-09-16-ISI/lessons/git-notebook/git-for-scientists.slides.html

Git data: commit, tree, blob

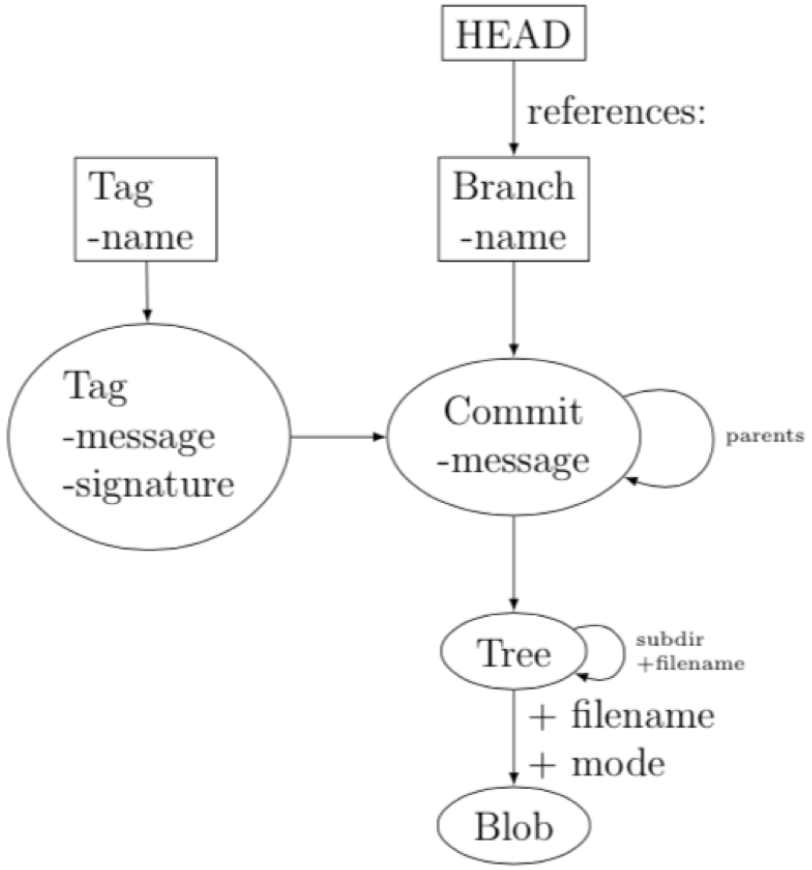


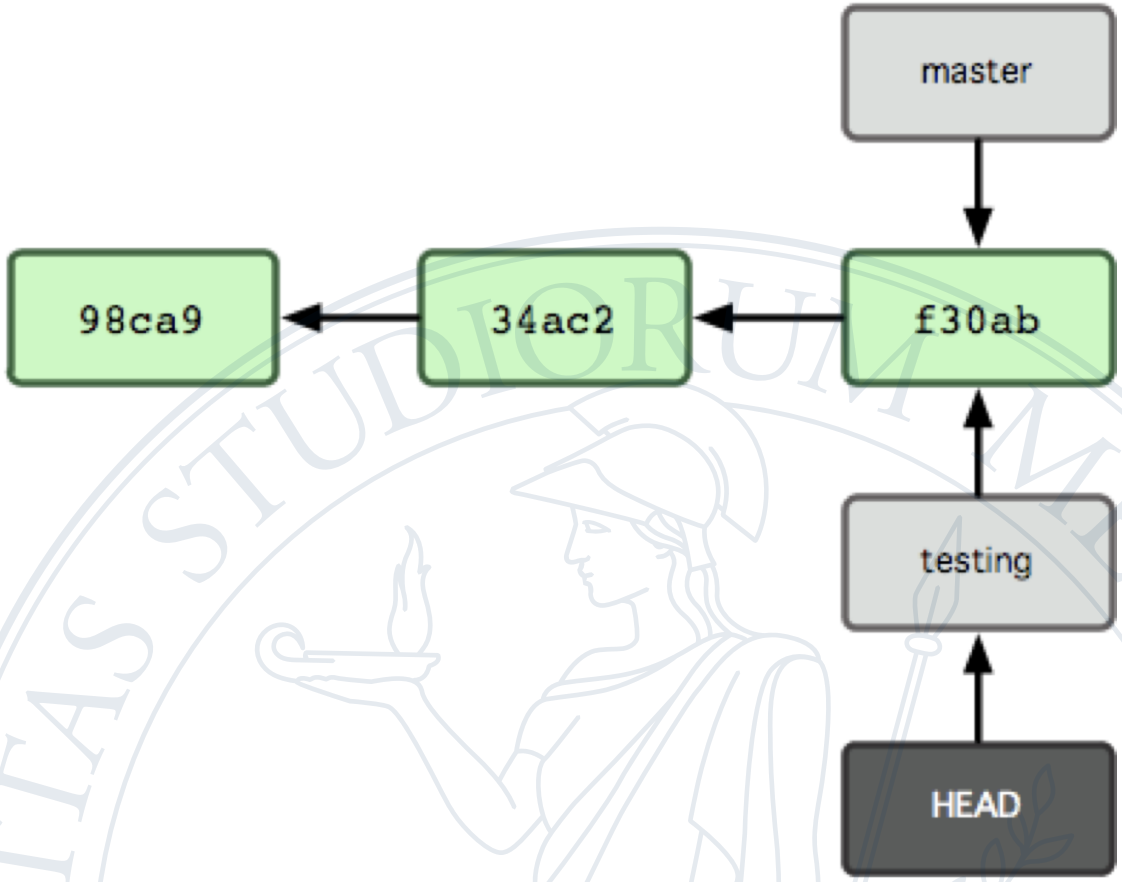


Git data: repository



Git data: HEAD, tag





Directories

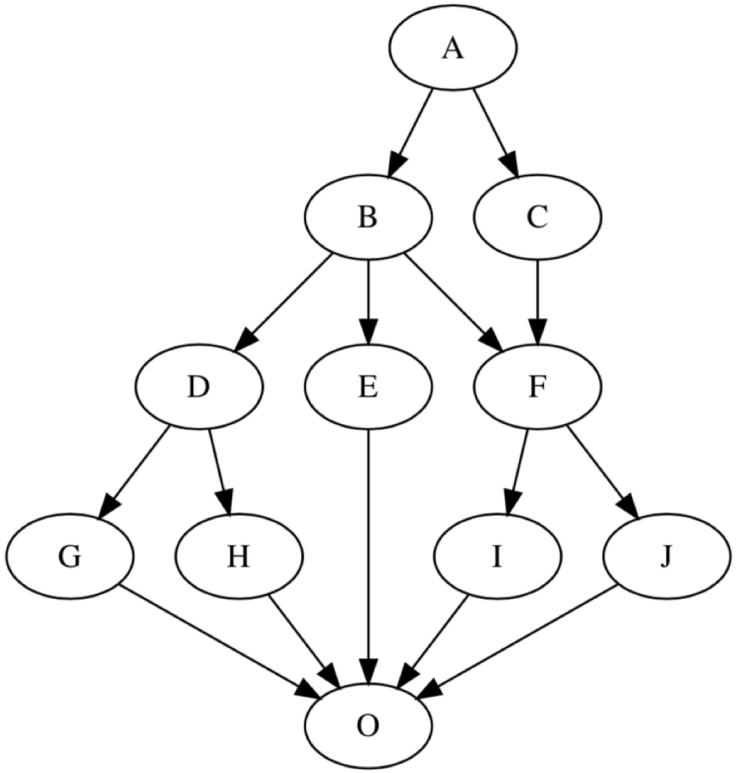
```
.git
├── HEAD
├── config
├── description
├── hooks
├── info
│    └── exclude
├── objects
│    ├── info
│    └── pack
└── refs
    ├── heads
    └── tags
```

http://git-scm.com/docs/gitrevisions

Come riferirsi a un commit

- *Nome*: il suo hash (o le sue prime lettere)
- *Nickname*: un tag, un branch, HEAD
- *Parentela*: aggiungere a uno dei precedenti i simboli
  - ^n ennesimo padre (se ce ne è più di uno)
  - ~n ennesimo antenato
- Data: “master@{1 hour ago}“

Come riferirsi a un commit



Come posso a partire da A nominare :

G = ?

E = ?

H = ?

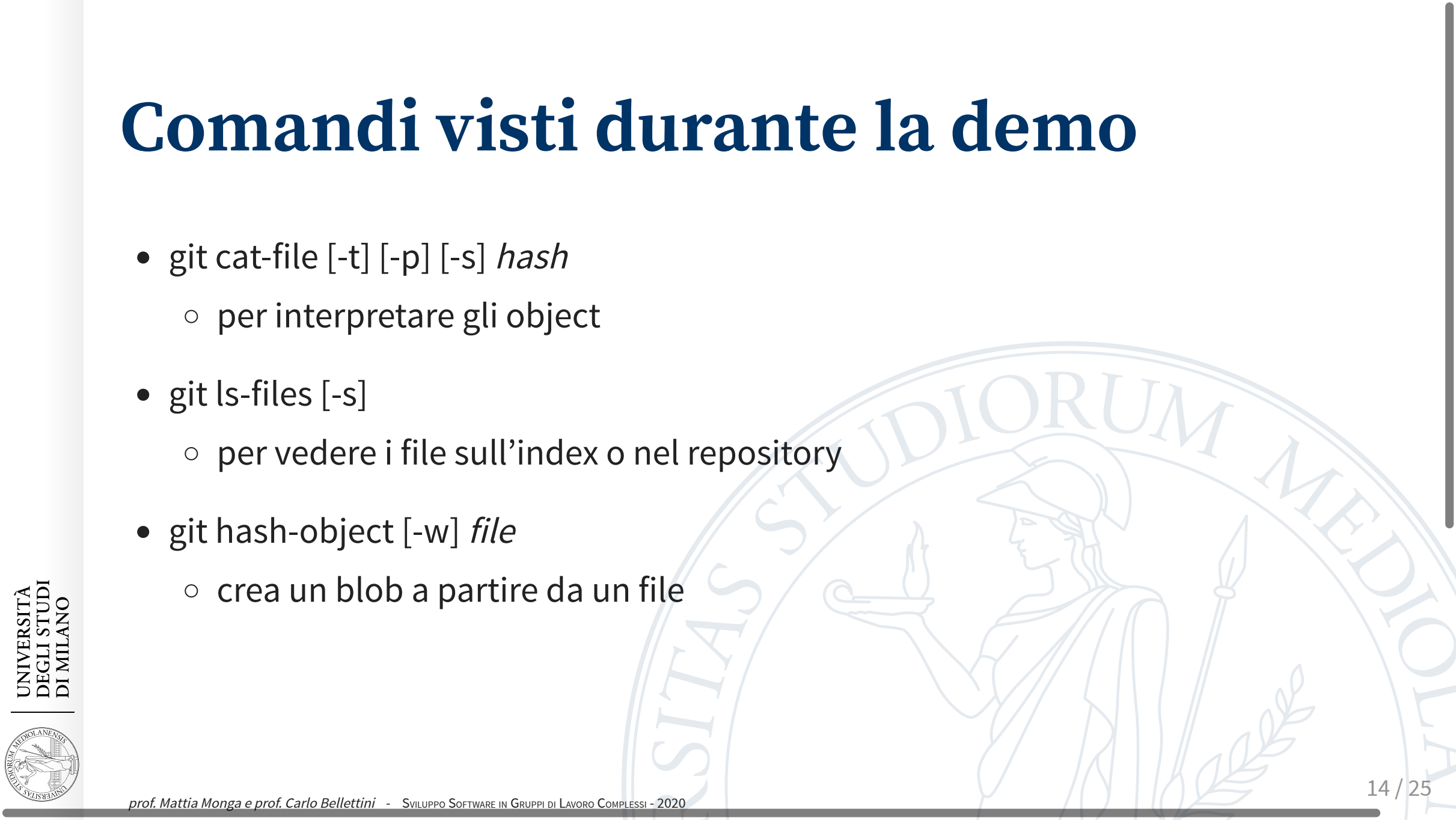
I = ?

Comandi base

- git init
- git clone
- git diff
- git status
- git log
- git branch
- git add
- git commit
- git checkout
- git merge
- git cherry-pick
- git remote
- git push
- git pull

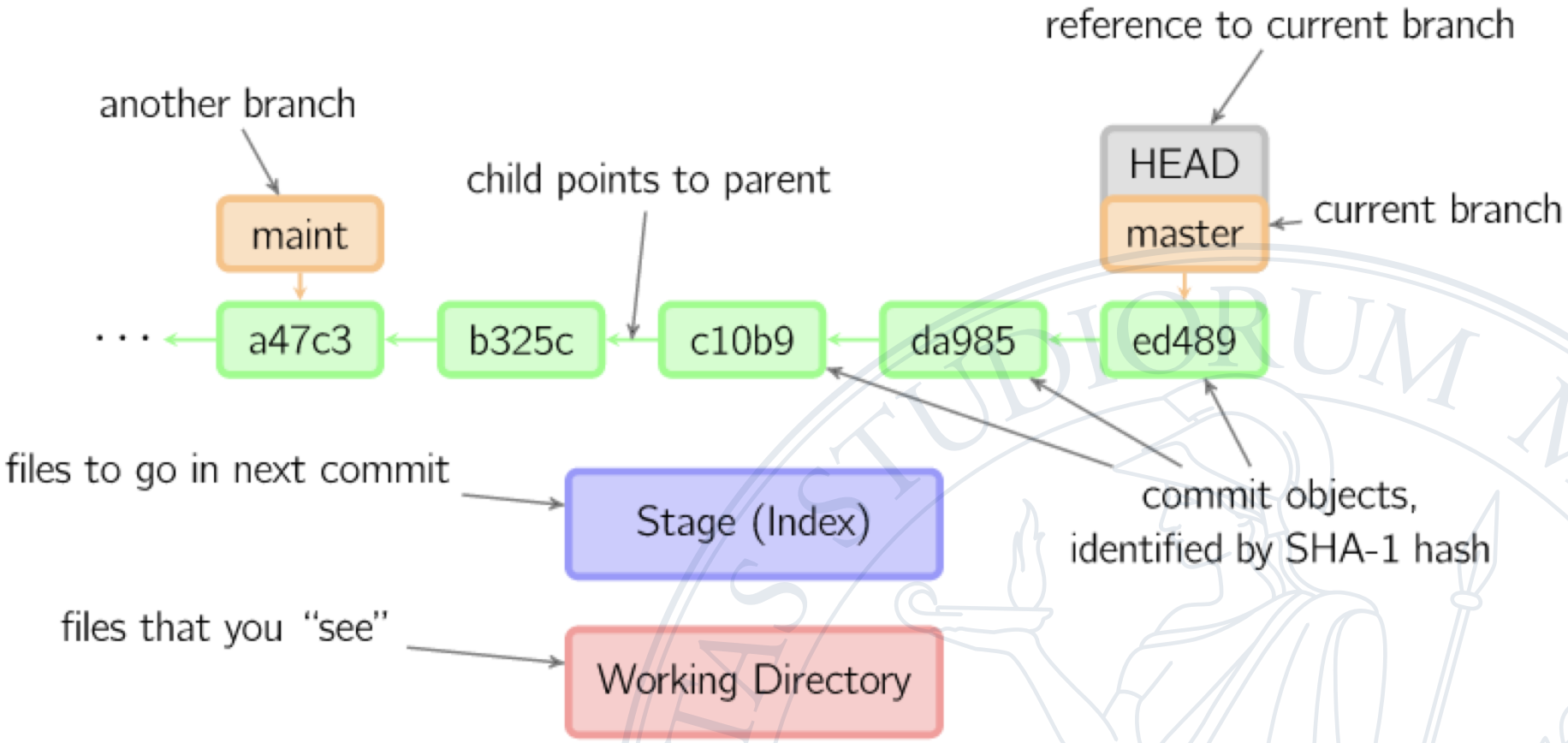
Comandi visti durante la demo

- git cat-file [-t] [-p] [-s] *hash*
  - per interpretare gli object
- git ls-files [-s]
  - per vedere i file sull'index o nel repository
- git hash-object [-w] *file*
  - crea un blob a partire da un file

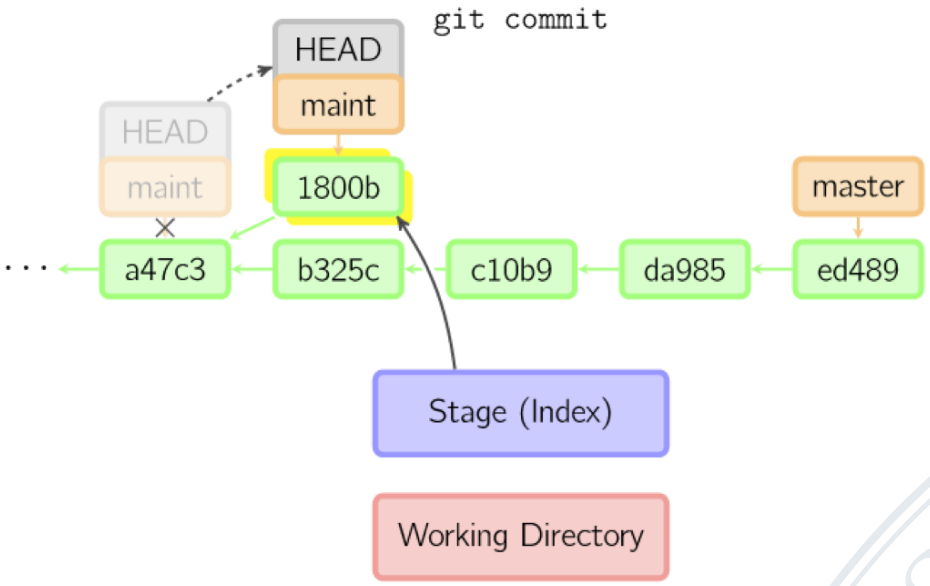


http://marklodato.github.io/visual-git-guide/index-en.html

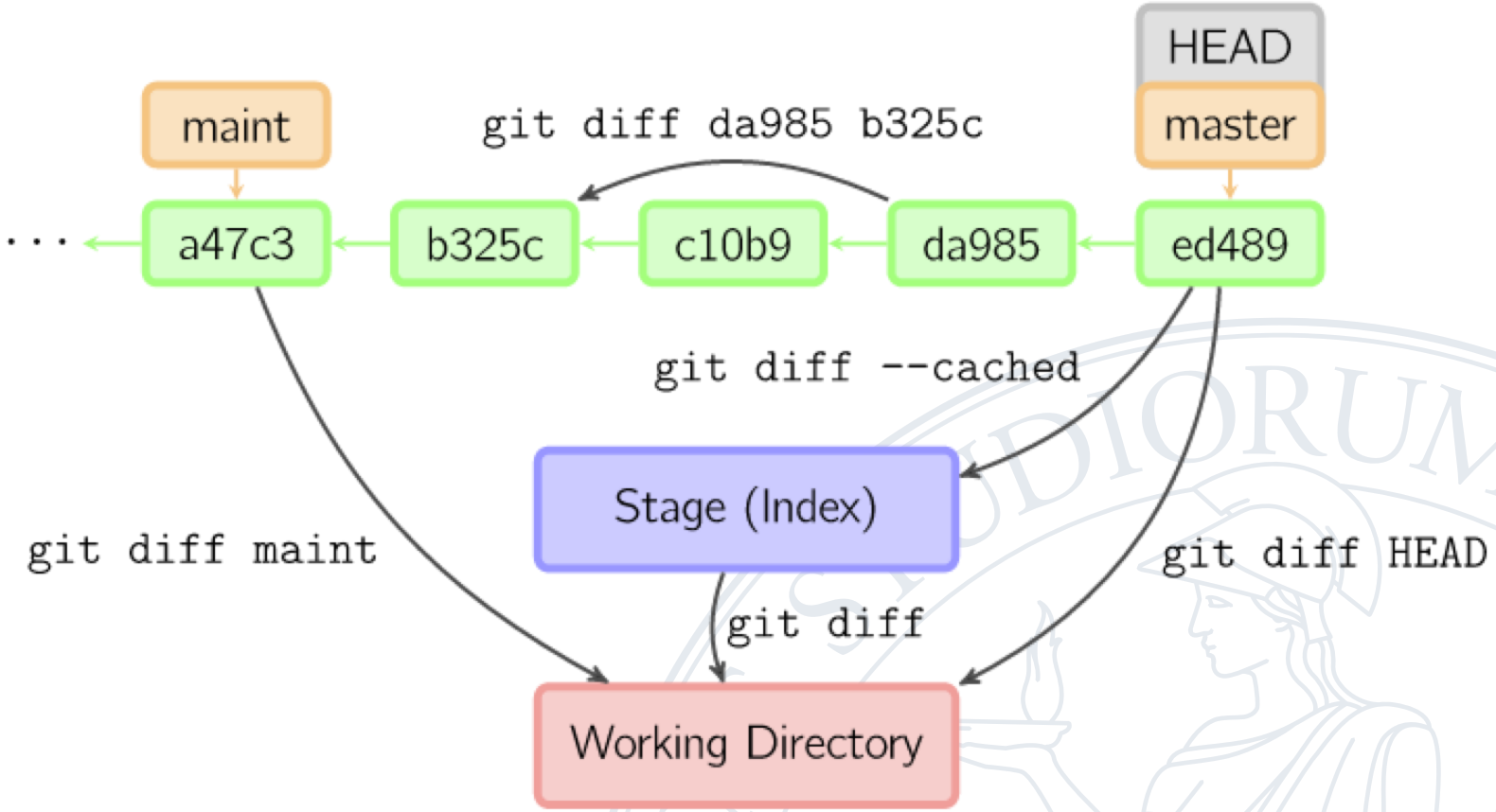
Convenzioni grafiche



git commit

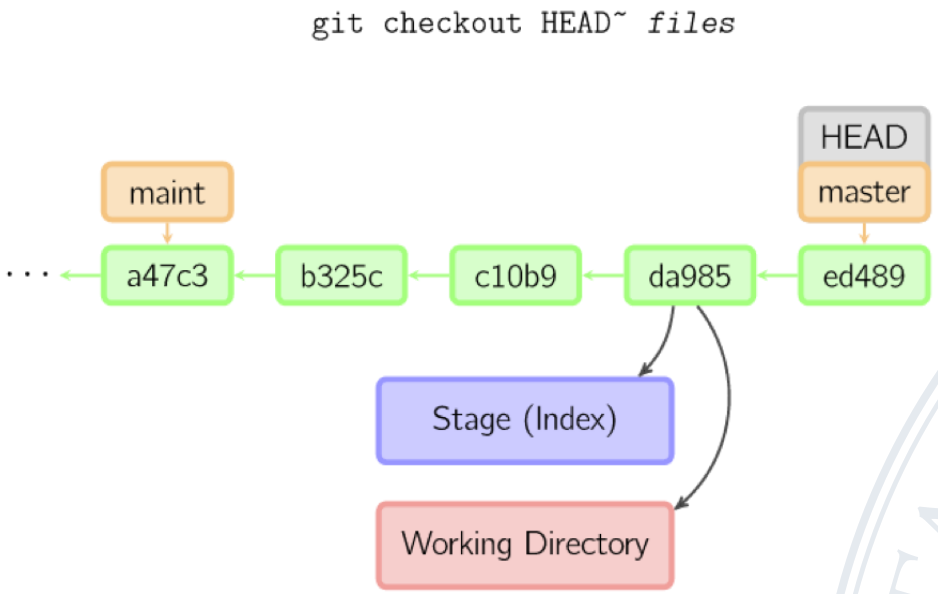

git commit
HEAD
maint
HEAD
maint
1800b
master
a47c3
b325c
c10b9
da985
ed489
Stage (Index)
Working Directory

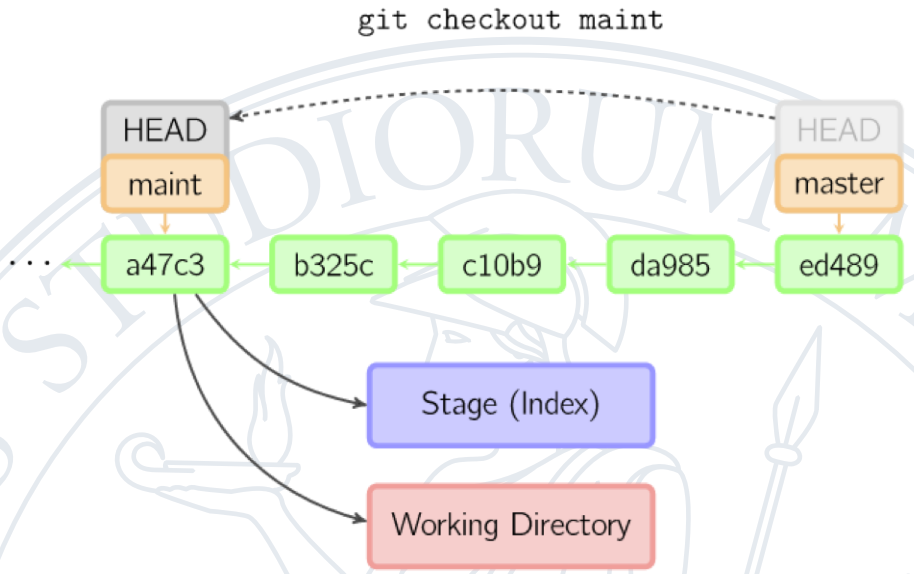
git diff [--cached] [ref] [ref1] [--] [files]



`git checkout [ ref ] [--] [ files ]`

- vengono copiati i files nella versione presente in *ref* dentro a *index* e *working directory*
- se non sono indicati *files* allora viene spostata la HEAD



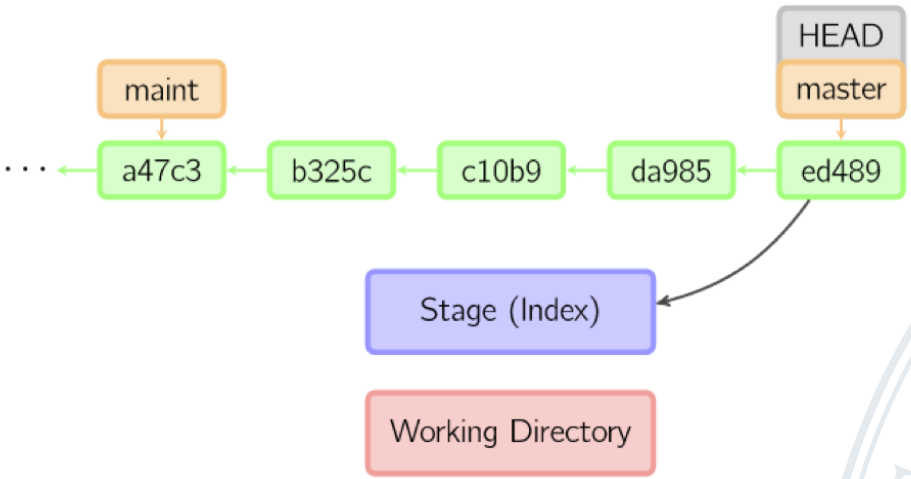


`git reset [ref] [--] [files]`

simile a checkout ma:

- con *files*, non modifica working directory
- senza *files*, sposta non solo HEAD, ma anche il branch puntato

`git reset -- files`



`git reset HEAD~3`