UNIVERSITÀ DEGLI STUDI
DI MILANO

Sviluppo Software in Gruppi di Lavoro Complesso

prof. Carlo Bellettini

prof. Mattia Monga

a.a. 2020-21

UNIVERSITÀ DEGLI STUDI
DI MILANO

12. Git (cont)

`git stash`

- mette da parte workingspace e index e poi fa reset

Alcuni scenari

- Interrupted workflow
- Pulling into a dirty tree
- Testing (partial) commits

Stash - Interrupted workflow

Il mio capo mi chiede qualcosa di urgente (EMERGENCY FIX)

```
# ... hack hack hack ...
$ git checkout -b my_wip
$ git commit -a -m "WIP"
$ git checkout master
$ edit emergency fix
$ git commit -a -m "Fix in a hurry"
$ git checkout my_wip
$ git reset --soft HEAD^
# ... continue hacking ...
```

```
# ... hack hack hack ...
$ git stash
$ edit emergency fix
$ git commit -a -m "Fix in a hurry"
$ git stash pop
# ... continue hacking ...
```

Stash - Pulling into a dirty tree

```
$ git pull
...
file foobar not up to date, cannot merge.
```

```
$ git stash
$ git pull
$ git stash pop
```

Stash - Testing (partial) commits

risolve il problema di testare una index diverso da wordking directory che si vuole committare

```
# ... hack hack hack ...
$ git add --patch foo            # add just first part to the index
$ git stash push --keep-index    # save all other changes to the stash
$ edit/build/test first part
$ git commit -m 'First part'     # commit fully tested change
$ git stash pop                  # prepare to work on all other changes
# ... repeat above five steps until one commit remains ...
$ edit/build/test remaining parts
$ git commit foo -m 'Remaining parts'
```

git filter-branch

permette di riscrivere la storia in maniera ancora più completa...

- fare come se un file non fosse stato mai versionato

```
git filter-branch --index-filter 'git rm --cached --ignore-unmatch filename'
```

- ricavare un repository con solo il contenuto di una subdirectory

```
git filter-branch --subdirectory-filter foodir -- --all
```

- fare come se alcuni file fossero sempre stati in una sottodirectory

```
git filter-branch -f --tree-filter 'mkdir ._p ; mv * ._p; mv ._p core;' -- --all
```

http://onlywei.github.io/explain-git-with-d3/#freeplay

Lavorare con repository remoti

- git remote *[subcmd]*
- git clone
- git push
- git fetch
- git pull

```
http://git-scm.com/docs/githooks
http://githooks.com/
```

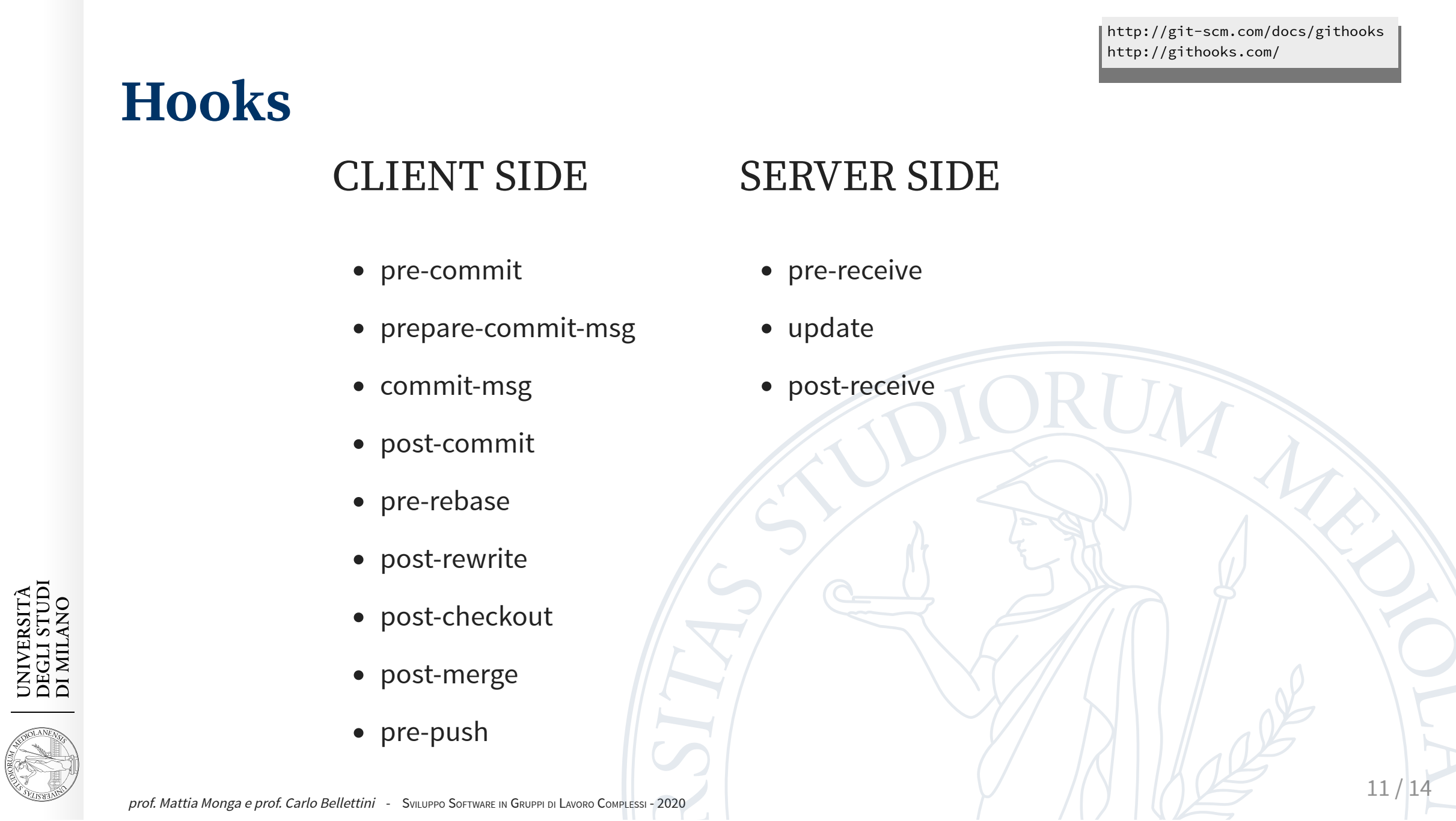
Hooks

CLIENT SIDE

- pre-commit
- prepare-commit-msg
- commit-msg
- post-commit
- pre-rebase
- post-rewrite
- post-checkout
- post-merge
- pre-push

SERVER SIDE

- pre-receive
- update
- post-receive



pre-commit hook

viene lanciato subito dopo un comando commit (prima di editare il messaggio) a meno che non si sia usata opzione `--no-verify`

```
git stash -a -q --keep-index
./run_tests.sh # oppure  gradle test
RESULT=$?
git reset --hard
git stash pop --index -q
[ $RESULT -ne 0 ] && exit 1
exit 0
```

può determinare il fallimento di un commit

post-commit hook

Il commit è stato ormai accettato quindi non può più cambiare esito del commit

Si può usare per mandare delle notifiche

oppure per

lolcommits (https://lolcommits.github.io/)

server-side hooks

- `pre-receive`
- `update`
  - possono essere usati per rifiutare un push (CI con capacità di bloccare commit))
  - differiscono nel come vengono chiamati
- `post-receive`
  - di nuovo può essere usato per notifiche (ma non foto visto che è in remoto)

esempio "banale":

```
GIT_WORK_TREE=/users/belletc/public_html/sito git checkout -f
```