Stash - Interrupted workflow

```
# ... hack hack hack ...
git checkout -b my_wip
git commit -a -m "WIP"
git checkout main
# edit emergency fix
git commit -am "Fix"
git checkout my_wip
??? merge o rebase
# ... continue hacking ...
```

```
git checkout -b myWIP
git commit -m "index"
git commit -am "wd"
git checkout main
(git reset --hard)
# edit emergency fix
git commit -am "Fix"
git checkout myWIP
git rebase main
git reset HEAD^
git reset —-soft HEAD^
git checkout main
git branch -D myWIP
```

```
# ... hack hack hack ...
git stash
# edit emergency fix
git commit -am "Fix"
git stash pop --index
# ... continue hacking ...
```

… avrei potuto clonare in altra directory, fare modifiche e poi push

… o non stare sviluppando su ramo principale

Stash - Pulling into a dirty tree

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

Fallisce ma intanto probabilmente ha già fatto almeno il `git fetch` e resta solo da risolvere il problema della directory sporca…

```
$ git stash
$ git pull
$ git stash pop --index
```

Stash - Testing (partial) commits

Risolve il problema di testare una index diverso da working directory che si vuole committare

```
# ... hack hack hack ...
$ git add --patch foo               # add just first part to the index
$ git stash push -u --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-repo

- rimpiazza git filter-branch
  - più veloce
  - più semplice
  - più potente
  - più sicuro
  - più facile da usare

Example: Remove file DA TUTTI I COMMIT

come se un file non fosse stato mai versionato

```
git filter-repo --path file.txt --invert-paths
```

Come funziona

```
git fast-export
          --show-original-ids --reference-excluded-parents
          --fake-missing-tagger --signed-tags=strip
          --tag-of-filtered-object=rewrite --use-done-feature --no-data
          --reencode=yes --mark-tags --all

| filter |

git -c core.ignorecase=false fast-import
         --date-format=raw-permissive
         --force --quiet
```

- cambiare email degli autori

```
git filter-repo --mailmap fname  # con 'new_email old_email' in una linea in fname
```

- ricavare un repository con solo il contenuto di una subdirectory

```
git filter-repo --subdirectory-filter foodir
```

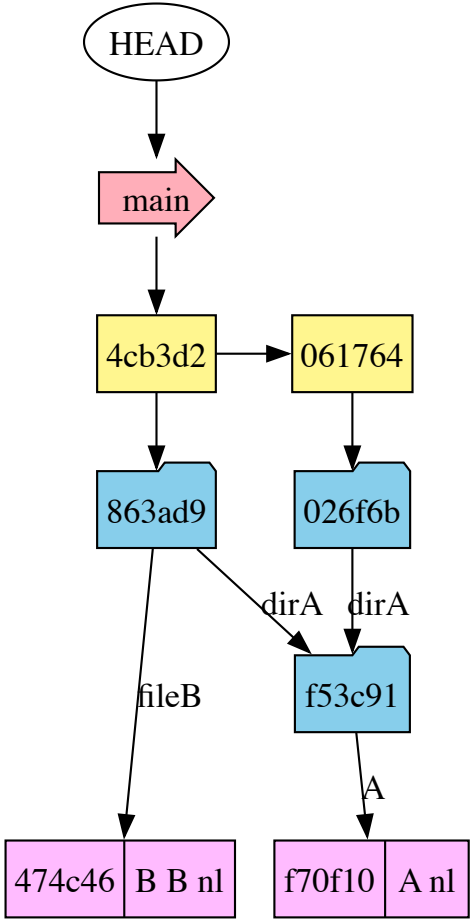
- fare come se alcuni file fossero sempre stati in una sottodirectory

```
git filter-repo --to-subdirectory-filter newsubdir
```

- esempio funzione custom

```
git filter-repo --blob-callback '
  if len(blob.data) > 25:
  # Mark this blob for removal from all commits
    blob.skip()
  else:
    blob.data = blob.data.replace(b"Hello", b"Goodbye")'
```

git fast-export --all




```
blob
mark :1
data 2
A

reset refs/heads/main
commit refs/heads/main
mark :2
author Carlo Bellettini <carlo.bellettini@unimi.it> 1744208702 +0200
committer Carlo Bellettini <carlo.bellettini@unimi.it> 1744208958 +0200
data 3
cA
M 100644 :1 dirA/A

blob
mark :3
data 3
BB

commit refs/heads/main
mark :4
author Carlo Bellettini <carlo.bellettini@unimi.it> 1744209318 +0200
committer Carlo Bellettini <carlo.bellettini@unimi.it> 1744209318 +0200
data 16
commited file B
from :2
M 100644 :3 fileB
```

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`
- può determinare il fallimento di un commit

```
git stash -u -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
```

post-commit hook

- Il commit è stato ormai accettato quindi non può più cambiare esito del commit
- Si può usare per mandare delle notifiche
  - magari multimediali: lolcommits

server-side hooks

`pre-receive` e `update`

- possono essere usati per rifiutare un push (CI con capacità di bloccare commit)
- differiscono nel come vengono chiamati
  - (pre-receive) lista di riferimenti da stdin
  - (update) terna "nome branch, old HASH, new HASH" come argomento

`post-receive`

- di nuovo può essere usato per notifiche

esempio "banale":

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

Come si usano i branch? (workflow)

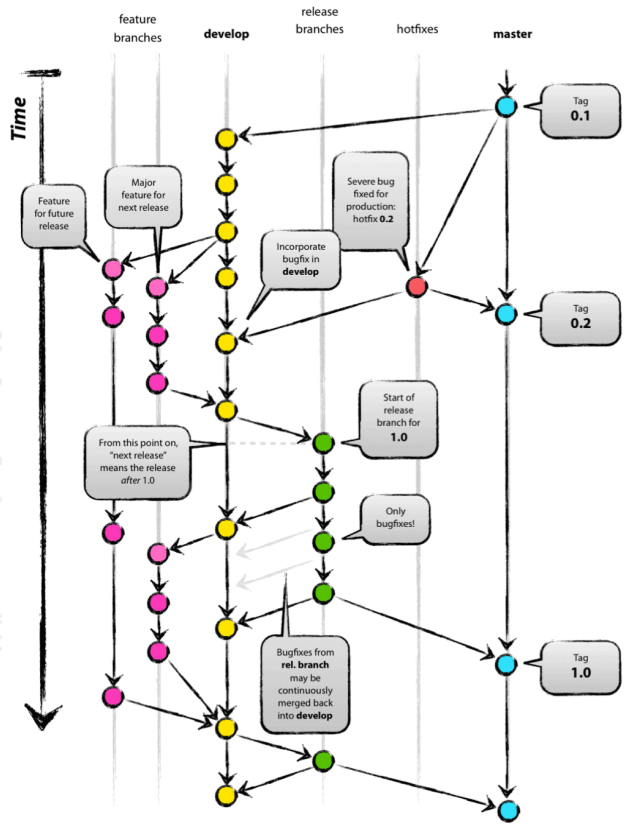
Non è possibile non usarli!

- libertà completa
- git-flow
- pull request
- gerrit

GitFlow AVH o CJS

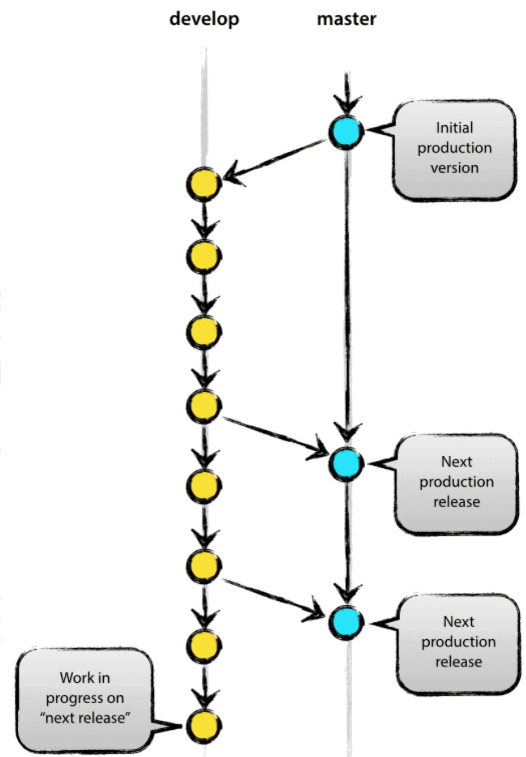
http://nvie.com/posts/a-successful-git-branching-model
https://github.com/petervanderdoes/gitflow-avh
https://github.com/CJ-Systems/gitflow-cjs

- Differenziazione tra tipi di branch
- nuove operazioni “guidate”
- non visibile nella figura, ma tratta anche i remote



master e develop

- Sono due rami con “vita infinita”
- **master** (e in particolare origin/master) contiene le versioni “stabili e pronte alla consegna”
- **develop** è il ramo di integrazione



https://github.com/petervanderdoes/gitflow-avh/blob/develop/git-flow-feature

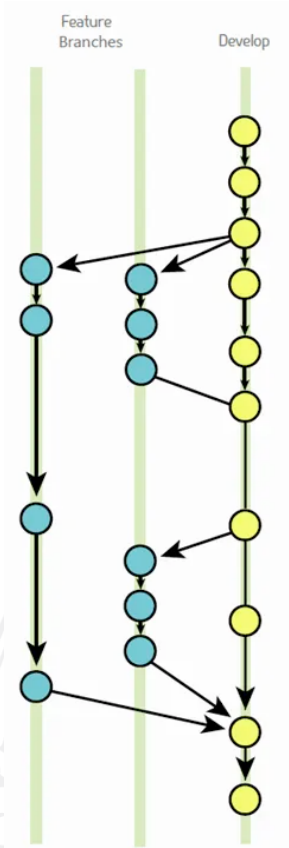
feature

- git flow feature start *feat1*

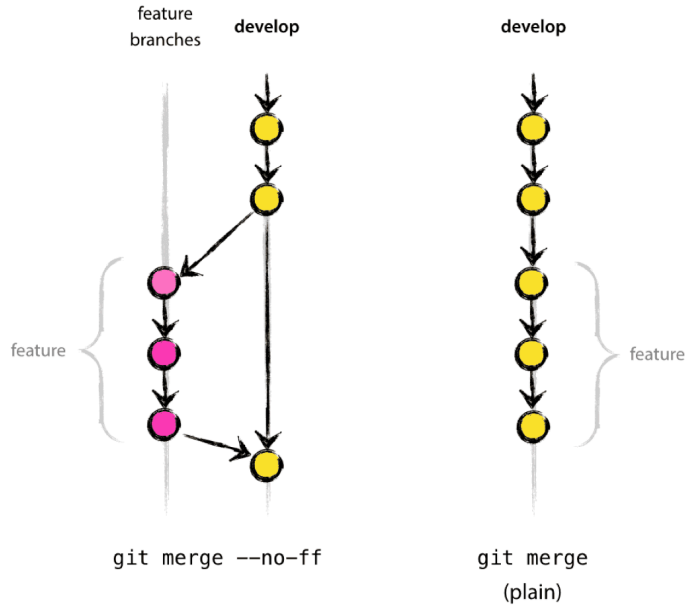
```
git checkout develop
git branch feat1
git checkout feat1
```

- git flow feature finish *feat1*

```
git checkout develop
git merge --no-ff feat1
git branch -d feat1
```



?



https://github.com/petervanderdoes/gitflow-avh/blob/develop/git-flow-release

release

- git flow release start *ver*

```
git checkout -b ver develop
```

- git flow release finish *ver*

```
git checkout master
git merge --no-ff release-ver
git tag -a ver
git checkout develop
git merge --no-ff release-ver
git branch -d release-ver
```