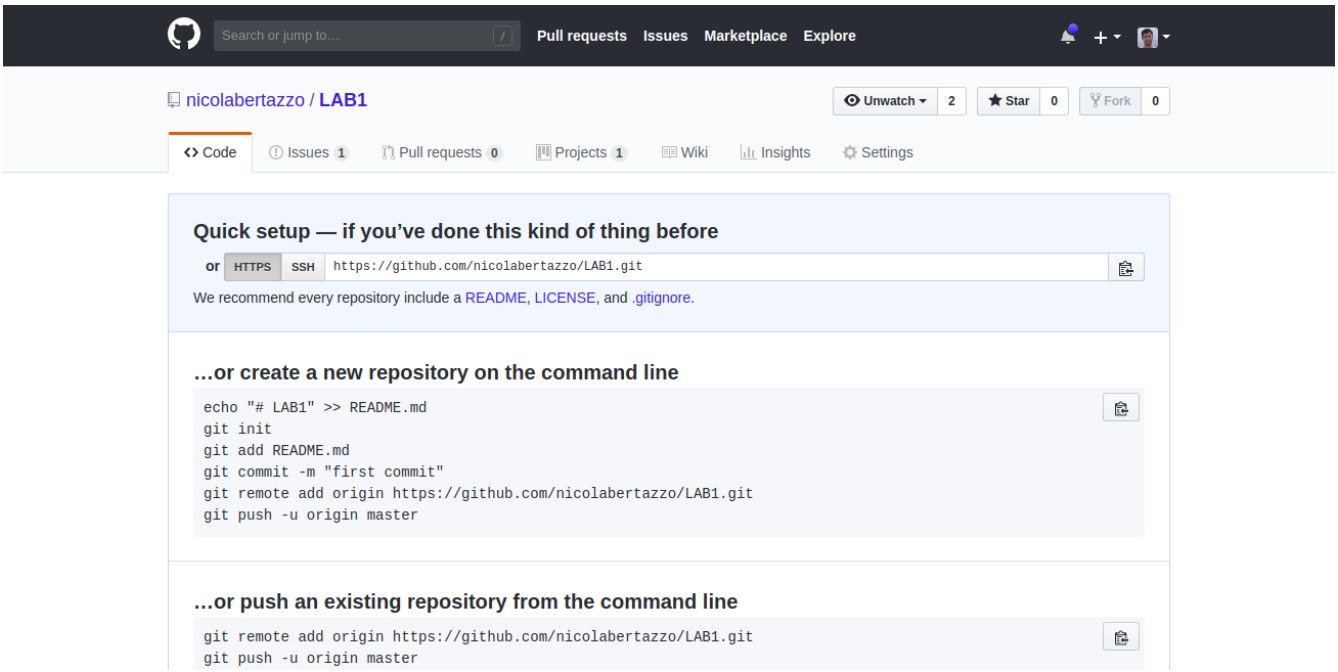
Git Work Flow e GitHub ITS

Table of Contents

Creazione del Repository locale

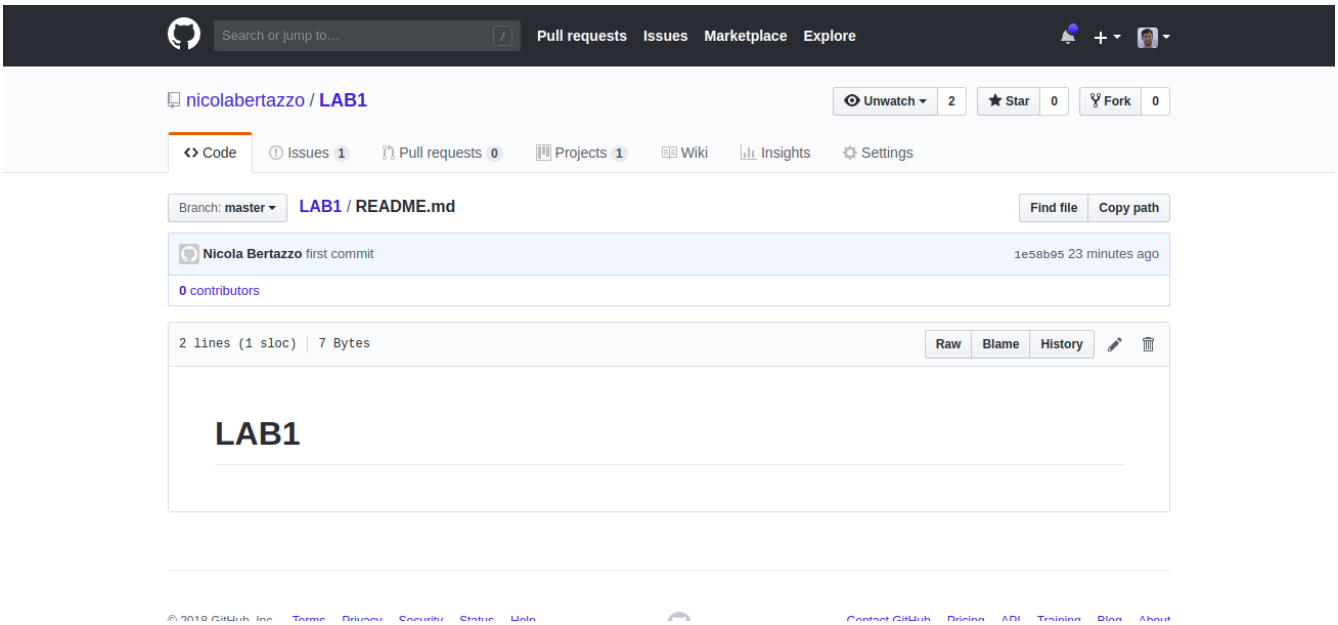
- Accedere a GitHub
- Accedere al repository creato nel laboratorio 1



- Aprire una shell git e seguire le istruzioni riportate in: *...or create a new repository on the command line*

```
echo "# LAB1" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://GitHub.com/[USERNAME]/[REPOSITORY].git
git push -u origin master
```

- verificare che nel file system locale e nel repository remoto è presente il file README.md



Comandi base

- Verifichiamo lo stato del repository locale

```
git status
Sul branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

- verificare i branch locali presenti nel repository

```
git branch
* master
```

- verificare i branch locali e remoti presenti nel repository

```
git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/master
```

- modificare il file *README.md* aggiungendo una nuova riga

```
echo "nuova riga" >> README.md
```

- verificare lo stato del repository locale

```
git status

Sul branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

- visualizzare le differenze apportate al file

```
git diff
diff --git a/README.md b/README.md
index 7936d01..ba0dc76 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,2 @@
 # LAB1
+nuova riga
```

- aggiungere il file nell'area di staging e visualizzare lo stato

```
git add README.md
git status
Sul branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   README.md
```

- rimuovere il file dall'area di stanging senza perdere le modifiche effettuate

```
git reset HEAD README.md
```

- verificare che il file contenga ancora le modifiche

```
cat README.md
```

- verificare che il file non è più nell'area di staging

```
git status
Sul branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

- rimuovere le modifiche al file (e riportarlo alla precedente versione)

```
git checkout -- README.md
```

- verificare lo stato del repository

```
git status
```

- modificare nuovamente il file e inviare le modifiche al repository locale

```
echo "nuova riga" >> README.md
git commit -a -m "aggiunta riga"
```

- verificare che il commit è stato salvato nel repository locale

```
git log
Sul branch master
Il tuo branch è avanti rispetto a 'origin/master' di 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
bertazzo@bertazzo3:/tmp/git_repo/LAB1$ git log
commit 4f29bda8c42480a4160516c442f2f00629cffff4 (HEAD -> master)
Author: Nicola Bertazzo <nicola.bertazzo@eng.it>
Date:   Mon Oct 15 22:29:28 2018 +0200

    aggiunta riga

commit 1e58b95ccc413ddbe177decc2eaee1d8a93226dd (origin/master, origin/HEAD)
Author: Nicola Bertazzo <nicola.bertazzo@eng.it>
Date:   Mon Oct 15 21:59:37 2018 +0200

    first commit
```

- inviamo le modifiche al repository remoto

```
git push origin master
```

- verifichiamo lo stao del repository locale

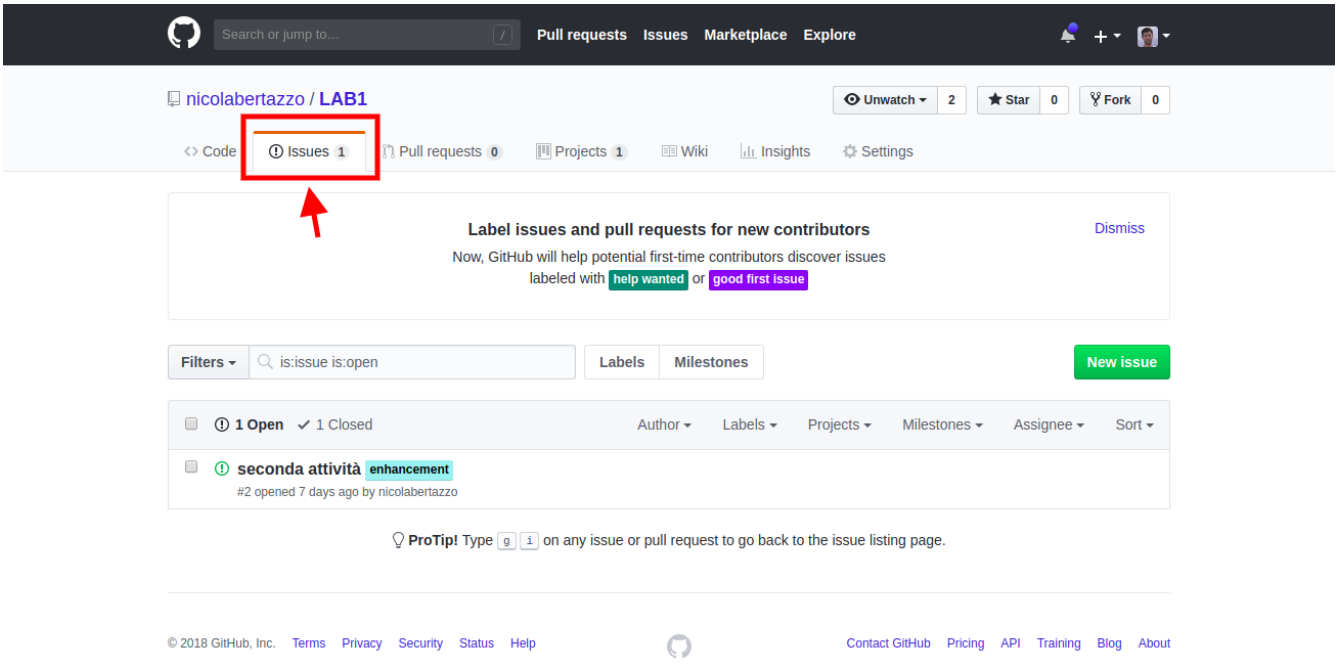
```
git status
Sul branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

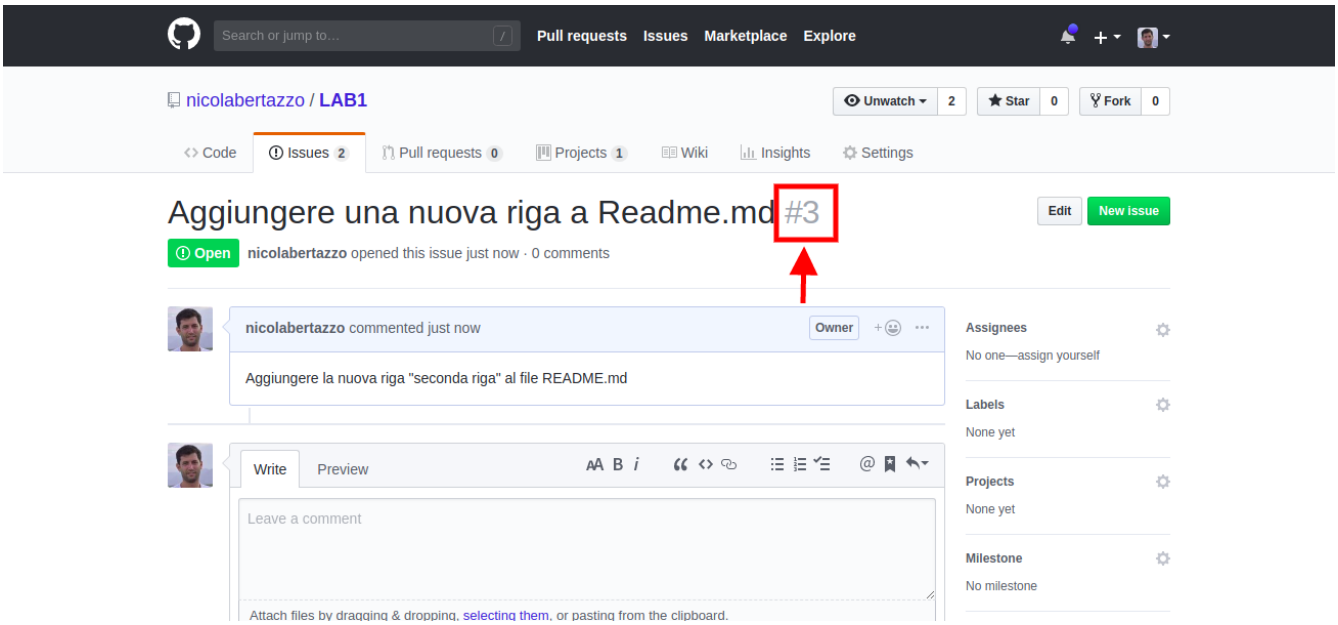
Relazioni tra commit e *Issue* in GitHub

Aggiungere una nova attività in GitHub

- accedere al progetto e cliccare nella tab *issues*



- creare una nuova *Issue* con i seugenti campi:
  - **Title:** Aggiungere una nuova riga a README.md
  - **Leave a Comment:** Aggiungere la nuova riga "seconda riga" al file README.md
- cliccare il bottone *"Submit new issue"*
- recuperare l'*ID* della nuova issue



Svolgimento dell'attività

Chiudere un *Issue* con un commit

- modificare il file README.md aggiungendo la "seconda riga"

```
echo "seconda riga" >> README.md
```

- aggiungere il file nell'area di staging

```
git add README.md
```

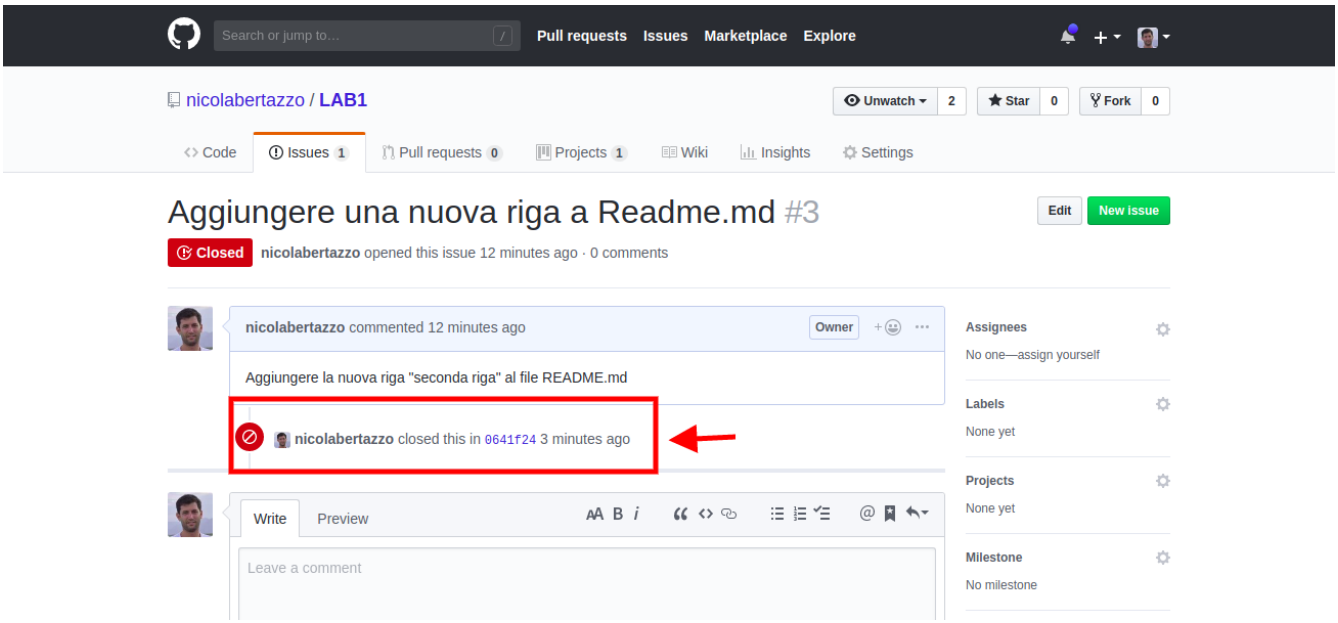
- effettuare il commit aggiungendo nel messaggio la seguente frase *"close #3"*

```
git commit -m "close #3"
```

- verificare lo stato del repository locale
- inviare le modifiche al repository remoto

```
git push origin master
```

- verificare lo stato della *Issue* #3 in GitHub



TIP GitHub permette di chiudere una *Issue* tramite l'invio di un commit al VCS con come messaggio *close #[ID]*

TIP Se la *Issue* chiusa dal commit è presente in una *project board* configurata con il template *Automated kamban* questo verrà spostata nella colonna *Done*

Per maggiori informazioni vedi: https://help.GitHub.com/articles/closing-issues-using-keywords/

Collegare le attività di sviluppo con le Issue

- modificare nuovamente il file

```
echo "terza riga" >> README.md
```

- aggiungere il file all'area di staging

```
git add README.md
```

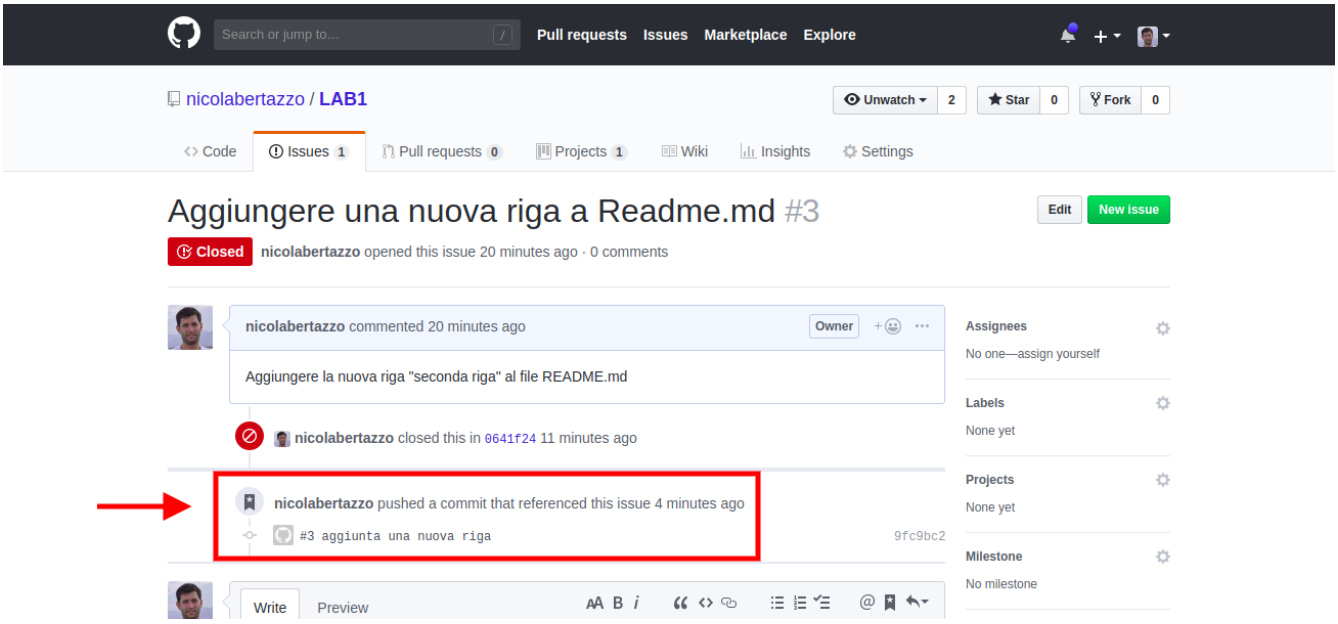
- effettuare il commit con messaggio *"#3 aggiunta una nuova riga"*

```
git commit -m "#3 aggiunta una nuova riga"
```

- effettuare il push

```
git push origin master
```

- verificare i commenti nella *Issue* #3



TIP GitHub permette di associare le modifiche effettuate nel VCS alle *Issue* censiti nel ITS riportando l'ID del workitem nei commit

Work Flow

Centralized Work Flow

Come descritto in https://www.atlassian.com/git/tutorials/comparing-workflows:

The Centralized Workflow is a great Git workflow for teams transitioning from SVN. Like Subversion, the Centralized Workflow uses a central repository to serve as the single point-of-entry for all changes to the project. Instead of *trunk*, the default development branch is called *master* and all changes are committed into this branch. This workflow doesn't require any other branches besides master.

Esempio

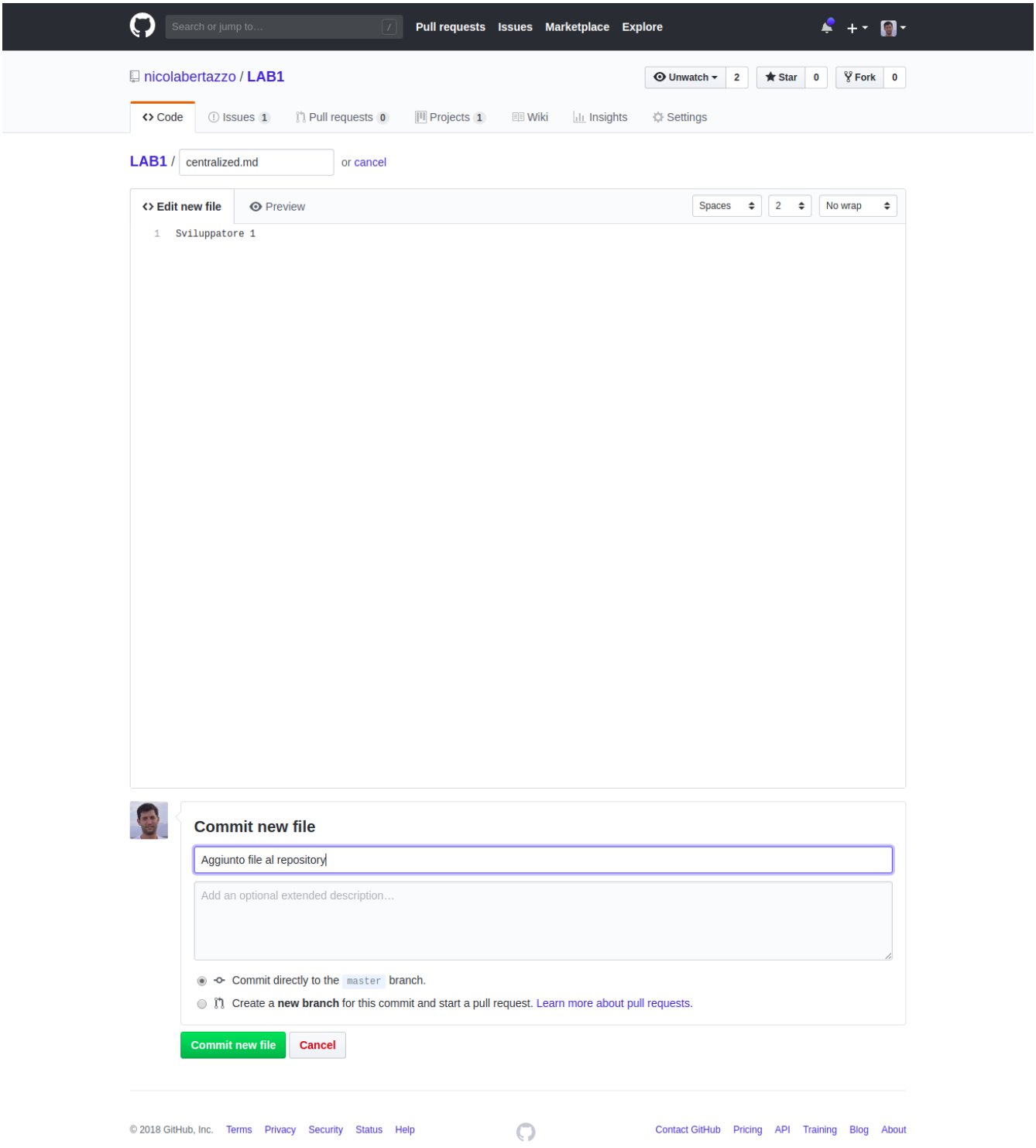
Simuliamo la modifica di due sviluppatori al file *centralized.md*

- Lo sviluppatore 1 effettuerà l'editing da GitHub tramite l'interfaccia web.
- Lo sviluppatore 2 effettuerà l'editing da client GIT

Sviluppatore 1

- crea il file in GitHub e inserisce il seguente contenuto:

```
Sviluppatore 1
```



Sviluppatore 2

- aggiorna il repository locale e aggiunge una riga al file

```
git pull
echo "sviluppatore 2" >> centralized.md
```

- aggiunge il la modifica all'area di staging e committa la modifica

```
git add centralized.md
git commit -m "aggiunta riga al file centralized.md"
```

- il repository locale dello sviluppatore 2 conterrà un commit non ancora presente nel repository remoto

Sviluppatore 1

- modifica nuovamente il file centralized.md aggiungendo una nuova riga

```
Sviluppatore 1
Nuova riga
```

- effettua il commit e il slvataggio direttamente nel repository remoto

Sviluppatore 2

- Prova ad inviare le modifiche al repository remoto

```
git push
...
! [rejected]        master -> master (fetch first)
error: failed to push some refs to 'https://github.com/nicolabertazzo/LAB1.git'
suggerimento: Updates were rejected because the remote contains work that you do
suggerimento: not have locally. This is usually caused by another repository pushing
suggerimento: to the same ref. You may want to first integrate the remote changes
suggerimento: (e.g., 'git pull ...') before pushing again.
suggerimento: See the 'Note about fast-forwards' in 'git push --help' for details.
```

- lo sviluppatore 2 aggiorna il suo branch locale

```
git pull origin master

Auto-merging centralized.md
CONFLICT (content): Merge conflict in centralized.md
Merge automatico fallito; risolvi i conflitti ed eseguire il commit
del risultato.
```

- lo sviluppatore 2 prima di inviare le modifiche al repository remoto deve risolvere i conflitti
  - modificare il file centralized.md e risolvere i conflitti
  - aggiungere il file all'area di stage, effettuare il commit e il push

```
git add centralized.md
git commit -m "risolti i conflitti"
git push origin master
```

TIP Opzionale: vedere l'opzione git pull --rebase

Feature Branch Work Flow

Come descritto in https://www.atlassian.com/git/tutorials/comparing-workflows:

Feature Branching is a logical extension of Centralized Workflow. The core idea behind the Feature Branch Workflow is that all feature development should take place in a dedicated branch instead of the master branch. This encapsulation makes it easy for multiple developers to work on a particular feature without disturbing the main codebase. It also means the master branch should never contain broken code, which is a huge advantage for continuous integration environments.

Esempio

Supponiamo che lo sviluppatore 1 deve creare un'attività che prevede la creazione e modifica di un file *feature.md*

- lo sviluppatore 1 si posiziona sul master, recupere le modifiche dal repository remoto e aggiorna la sua copia locale con l'ultima versione scaricata

```
git checkout master
git fetch origin
git reset --hard origin/master      Questa terza riga cancella le modifiche sul branch master
```

- lo sviluppatore 1 crea un nuovo branch e si posiziona sul nuovo branch creato

```
git checkout -b new-feature
```

- lo sviluppatore 1 crea il nuovo file ed effettua le modifiche

```
echo "nuove modifiche" >> feature.md
```

- verifica lo stato dei file presenti nella sua copia locale, aggiunge il nuovo file all'area di staging e lo committa

```
git status
git add feature.md
git commit -m "aggiunto file feature.md"
```

- condivide il nuovo branch nel repository remoto

```
git push -u origin new-feature
```

git push --set-upstream origin new-feature
Per pushare su un nuovo branch

- Per integrare lo sviluppo nel ramo principale lo sviluppato 1 può:
  - effettuare una pull request
  - o effettuare il merge direttamente nel ramo master

Vediamo la seconda opzione

- lo sviluppatore 1 si posiziona sul ramo master e lo aggiorna

```
git checkout master
git fetch origin
git reset --hard origin/master
```

- effettua il merge con il nuovo ramo

```
git merge new-feature
```

- se sono presenti conflitti, devono essere risolti altrimenti si possono invare le modifiche al repository remoto

```
git push origin master
```

GitFlow

Come descritto in https://www.atlassian.com/git/tutorials/comparing-workflows:

The Gitflow Workflow was first published in a highly regarded 2010 blog post from Vincent Driessen at nvie. The Gitflow Workflow defines a strict branching model designed around the project release. This workflow doesn't add any new concepts or commands beyond what's required for the Feature Branch Workflow. Instead, it assigns very specific roles to different branches and defines how and when they should interact.

Come esempio utilizzare i tutorial:

- https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
- https://danielkummer.github.io/git-flow-cheatsheet/

seguendo i passi:

- Inizializzare il repository con git flow
- Creating a feature branch

- Finishing a feature branch
- Creating a release branch
- Finishing a release branch