PIEMONTE
FONDI STRUTTURALI E DI INVESTIMENTO EUROPEI 2014/2020
fondo sociale europeo
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

# Fondamenti di Version Control

## Michele Leone

## Pratica

in collaborazione con:
ISTITUTI TECNICI SUPERIORI
Ministero dell'Istruzione
UNIONE EUROPEA
REGIONE PIEMONTE
per una crescita intelligente, sostenibile ed inclusiva
www.regione.piemonte.it/europa2020
INIZIATIVA CO-FINANZIATA CON FSE

# Fondamenti di Version Control - Pratica

- Vediamo **come** usare Git in pratica

# Fondamenti di Version Control - Pratica

- Vediamo **come** usare Git in pratica
  - Inizieremo con le istruzioni da riga di comando

# Fondamenti di Version Control - Pratica

- Vediamo **come** usare Git in pratica
  - Inizieremo con le istruzioni da riga di comando (aka Git for boomers)

# Fondamenti di Version Control - Pratica

- Innanzitutto scarichiamo git da qui

# Fondamenti di Version Control - Pratica

- Innanzitutto scarichiamo git da qui
- Questo ci dà la possibilità di usare Git Bash, un programmino che fornisce una shell (unix-like) per utilizzare le istruzioni unix (comprese quelle di git) sulla nostra macchina

# Fondamenti di Version Control - Pratica

- Per **CREARE** un repository remoto:

# Fondamenti di Version Control - Pratica

- Per **CREARE** un repository remoto:
  - "git init" nella cartella in cui si desidera crearlo

# Fondamenti di Version Control - Pratica

```
$ mkdir gitTests

$ cd gitTests/

$ mkdir vcTest

$ git init
Initialized empty Git repository in C:/Users/mleone/gitTests/.git/
```

# Fondamenti di Version Control - Pratica

- Per **CLONARE** un repository remoto:

# Fondamenti di Version Control - Pratica

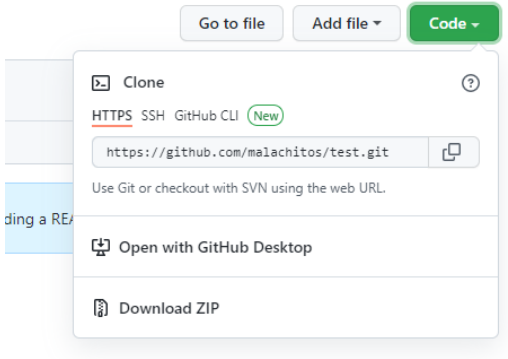
- Per **CLONARE** un repository remoto:
  - "git clone [url repository remoto]" nella cartella in cui si desidera clonare

# Fondamenti di Version Control - Pratica

```
$ mkdir cloneTest


$ cd cloneTest/


$ git clone https://github.com/malachitos/test.git
Cloning into 'test'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

# Fondamenti di Version Control - Pratica

- Ovviamente esistono modi più user-friendly, ad esempio in GitHub:

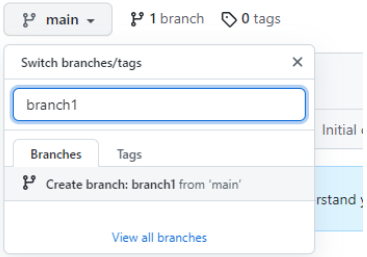


# Fondamenti di Version Control - Pratica

- Ogni volta che si vuole sviluppare un'implementazione, si crea un branch:

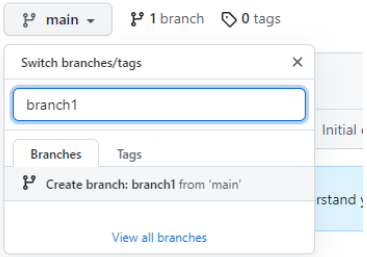
# Fondamenti di Version Control - Pratica

- Ogni volta che si vuole sviluppare un'implementazione, si crea un branch:



# Fondamenti di Version Control - Pratica

- Ogni volta che si vuole sviluppare un'implementazione, si crea un branch:




- Bisogna fare in modo che in locale si lavori sul **branch** appena creato e non su **master**

# Fondamenti di Version Control - Pratica

```
$ git fetch
From https://github.com/malachitos/test
 * [new branch]      branch_test_1 -> origin/branch_test_1


$ git checkout branch_test_1
Switched to a new branch 'branch_test_1'
Branch 'branch_test_1' set up to track remote branch 'branch_test_1' from 'origin'.
```

# Fondamenti di Version Control - Pratica

```
$ git fetch
From https://github.com/malachitos/test
 * [new branch]      branch_test_1 -> origin/branch_test_1


$ git checkout branch_test_1
Switched to a new branch 'branch_test_1'
Branch 'branch_test_1' set up to track remote branch 'branch_test_1' from 'origin'.
```

Faccio un "controllo" sul repository remoto per vedere se ci sono "novità"

# Fondamenti di Version Control - Pratica

```
$ git fetch
From https://github.com/malachitos/test
 * [new branch]      branch_test_1 -> origin/branch_test_1


$ git checkout branch_test_1
Switched to a new branch 'branch_test_1'
Branch 'branch_test_1' set up to track remote branch 'branch_test_1' from 'origin'.
```

Faccio un "controllo" sul repository remoto per vedere se ci sono "novità"

Scarico dal repository remoto il branch che ho creato (e di cui la fetch mi ha "notificato" l'esistenza)

# Fondamenti di Version Control - Pratica

- Recap veloce, abbiamo:

# Fondamenti di Version Control - Pratica

- Recap veloce, abbiamo:
  - Creato il repository remoto

# Fondamenti di Version Control - Pratica

- Recap veloce, abbiamo:
  - Creato il repository remoto
  - Creato il  repository locale facendo **clone** sul repository remoto

# Fondamenti di Version Control - Pratica

- Recap veloce, abbiamo:
  - Creato il repository remoto
  - Creato il  repository locale facendo **clone** sul repository remoto
  - Creato un branch in  remoto

# Fondamenti di Version Control - Pratica

- Recap veloce, abbiamo:
  - Creato il repository remoto
  - Creato il  repository locale facendo **clone** sul repository remoto
  - Creato un branch in  remoto
  - Trovato il nuovo branch facendo una **fetch**

# Fondamenti di Version Control - Pratica

- Recap veloce, abbiamo:
  - Creato il repository remoto
  - Creato il  repository locale facendo **clone** sul repository remoto
  - Creato un branch in  remoto
  - Trovato il nuovo branch facendo una **fetch**
  - Scaricato il nuovo branch in locale e **switchato** da master a questo branch facendo un **checkout**

# Fondamenti di Version Control - Pratica

- A questo punto, facciamo le modifiche che abbiamo in mente...

# Fondamenti di Version Control - Pratica

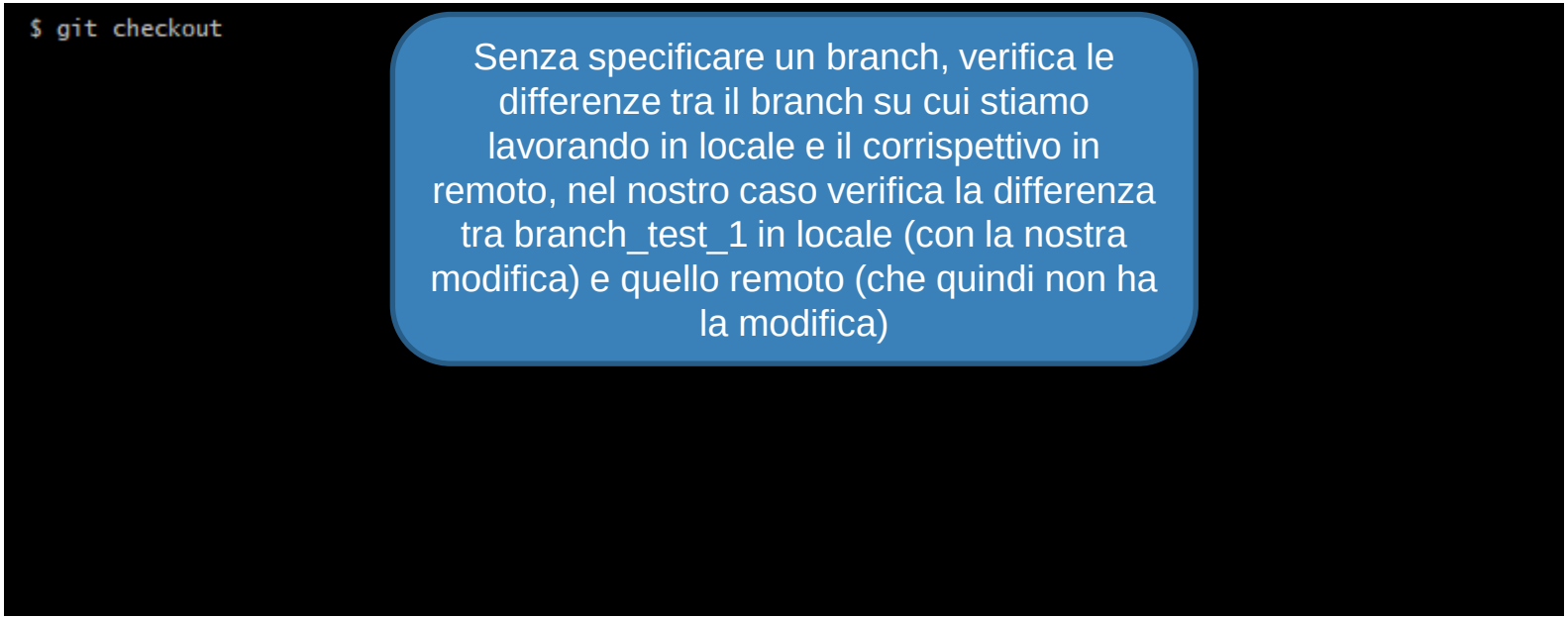
- A questo punto, facciamo le modifiche che abbiamo in mente...
- ...prima o poi dovremo portare su master queste modifiche...

# Fondamenti di Version Control - Pratica

- A questo punto, facciamo le modifiche che abbiamo in mente...
- ...prima o poi dovremo portare su master queste modifiche...
- ...come facciamo?

# Fondamenti di Version Control - Pratica

# Fondamenti di Version Control - Pratica



# Fondamenti di Version Control - Pratica

```
$ git checkout
Your branch is up to date with 'origin/branch_test_1'.
```

# Fondamenti di Version Control - Pratica

```
$ git checkout
Your branch is up to date with 'origin/branch_test_1'.
```

???

# Fondamenti di Version Control - Pratica

```
$ git checkout
Your branch is up to date with 'origin/branch_test_1'.
```

Non vede niente da committare…

# Fondamenti di Version Control - Pratica

```
$ git checkout
Your branch is up to date with 'origin/branch_test_1'.

$ git add modifiche.txt
```

Non vede niente da committare…

Aggiungo allo staging il file che ho creato

# Fondamenti di Version Control - Pratica

```
$ git checkout
Your branch is up to date with 'origin/branch_test_1'.

$ git add modifiche.txt

$ git checkout
A       modifiche.txt
Your branch is up to date with 'origin/branch_test_1'.
```

Non vede niente da committare…

Aggiungo allo staging il file che ho creato

Ora vedo che è stato aggiunto il file "modifiche.txt"

# Fondamenti di Version Control - Pratica

```
$ git checkout
Your branch is up to date with 'origin/branch_test_1'.

$ git add modifiche.txt

$ git checkout
A       modifiche.txt
Your branch is up to date with 'origin/branch_test_1'.
```

Non vede niente da committare…

Aggiungo allo staging il file che ho creato

Ora vedo che è stato aggiunto il file "modifiche.txt"

Inoltre, vedo che il branch è allineato a quello remoto

# Fondamenti di Version Control - Pratica

```
$ git checkout
Your branch is up to date with 'origin/branch_test_1'.

$ git add modifiche.txt

$ git checkout
A       modifiche.txt
Your branch is up to date with 'origin/branch_test_1'.
```

Non vede niente da committare…

Aggiungo allo staging il file che ho creato

Ora vedo che è stato aggiunto il file "modifiche.txt"

Inoltre, vedo che il branch è allineato a quello remoto

Che vuol dire?

# Fondamenti di Version Control - Pratica

```
$ git checkout
Your branch is up to date with 'origin/branch_test_1'.

$ git add modifiche.txt

$ git checkout
A       modifiche.txt
Your branch is up to date with 'origin/branch_test_1'.
```

Non vede niente da committare…

Aggiungo allo staging il file che ho creato

Ora vedo che è stato aggiunto il file "modifiche.txt"

Inoltre, vedo che il branch è allineato a quello remoto

Che vuol dire?

Che nessun altro ha aggiornato il branch!

# Fondamenti di Version Control - Pratica

- Potrei non voler committare **tutte** le modifiche che ho fatto...
- Per questo esiste il concetto dell'**area di staging**
  - Ho creato/modificato 3 files ma ne voglio committare solo 1
  - "git add [nomefile]" per mettere un **file specifico** in **staging**
  - "git add" per mettere **tutti i file** in **staging**
    - Questa pratica è **sconsigliatissima** perché si rischia di committare file di cui si può ignorare l'esistenza e che non devono essere committati

# Fondamenti di Version Control - Pratica

- Adesso siamo pronti per il **commit** (che salva sul repository locale) e il **push** (che "invia" il repository locale sul server remoto sovrascrivendolo)

# Fondamenti di Version Control - Pratica

```
$ git commit -m "aggiunto modifiche.txt"
[branch_test_1 9094fd2] aggiunto modifiche.txt
 1 file changed, 1 insertion(+)
 create mode 100644 modifiche.txt
```

Committo con un messaggio che faccia capire cosa ho fatto. Git mi notifica che ha effettuato il commit, 1 file aggiunto

# Fondamenti di Version Control - Pratica

```
$ git commit -m "aggiunto modifiche.txt"
[branch_test_1 9094fd2] aggiunto modifiche.txt
 1 file changed, 1 insertion(+)
 create mode 100644 modifiche.txt


$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 331 bytes | 331.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/malachitos/test.git
   61454f3..9094fd2  branch_test_1 -> branch_test_1
```

Committo con un messaggio che faccia capire cosa ho fatto. Git mi notifica che ha effettuato il commit, 1 file aggiunto

# Fondamenti di Version Control - Pratica

ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

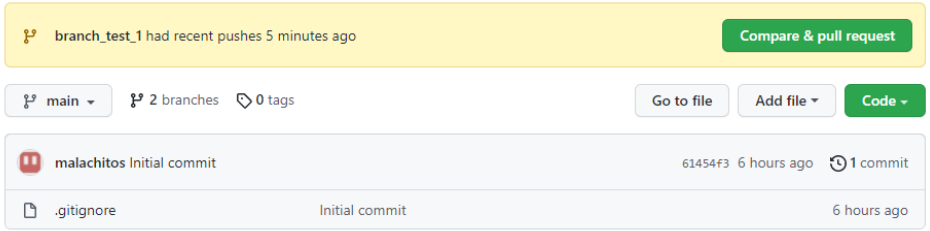
POR Piemonte
FSE 2014-2020

# Fondamenti di Version Control - Pratica

- Andiamo a controllare sul repository remoto...

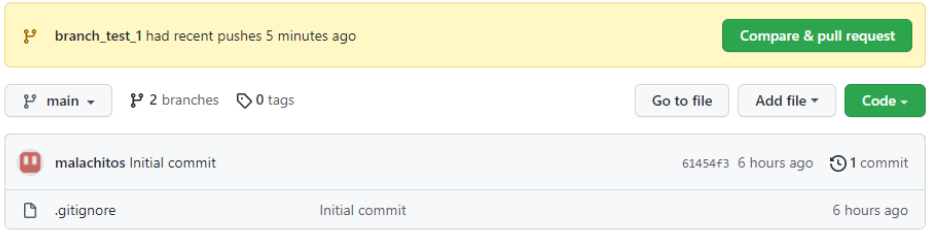
# Fondamenti di Version Control - Pratica

- Andiamo a controllare sul repository remoto...



# Fondamenti di Version Control - Pratica

- Andiamo a controllare sul repository remoto...




- ...well done!

# Fondamenti di Version Control - Pratica

- La dicitura "Compare and pull request" si riferisce ad una pratica diffusa che consiste nel verificare le modifiche presenti sul branch prima di **integrarle** (cioè riportarle) sul master. Questo permette di:

- La dicitura "Compare and pull request" si riferisce ad una pratica diffusa che consiste nel verificare le modifiche presenti sul branch prima di **integrarle** (cioè riportarle) sul master. Questo permette di:
  - Decidere quando integrare le modifiche, potrei avere un passaggio in produzione programmato a breve col quale non voglio portare le modifiche presenti su un determinato branch

# Fondamenti di Version Control - Pratica

- La dicitura "Compare and pull request" si riferisce ad una pratica diffusa che consiste nel verificare le modifiche presenti sul branch prima di **integrarle** (cioè riportarle) sul master. Questo permette di:
  - Decidere quando integrare le modifiche, potrei avere un passaggio in produzione programmato a breve col quale non voglio portare le modifiche presenti su un determinato branch
  - Verificare il funzionamento, l'efficacia e la pulizia del codice prima di integrarle sul master

- Ipotizziamo che qualcuno stia lavorando su un altro branch, **branch_test_2**, e debba prendere le modifiche appena committate su **branch_test_1** e integrate su master; dovrà:

# Fondamenti di Version Control - Pratica

- Ipotizziamo che qualcuno stia lavorando su un altro branch, **branch_test_2**, e debba prendere le modifiche appena committate su **branch_test_1** e integrate su master; dovrà:
  - Aggiornare il master locale
  - Fare il merge da **master** a **branch_test_2**

# Fondamenti di Version Control - Pratica

```
$ git switch master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

Switch a master

# Fondamenti di Version Control - Pratica

```
$ git switch master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

Perchè non mi dice che ci sono modifiche da prendere???

# Fondamenti di Version Control - Pratica

```
$ git switch master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

Perchè non mi dice che ci sono modifiche da prendere???

Perchè non abbiamo fatto la fetch!

# Fondamenti di Version Control - Pratica

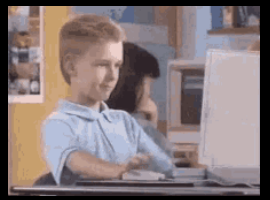
```
$ git switch master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.


$ git fetch
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 628 bytes | 11.00 KiB/s, done.
From https://github.com/malachitos/test
   64a71c9..0f55b53  master     -> origin/master
```

# Fondamenti di Version Control - Pratica

```
$ git switch master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.


$ git fetch
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 628 bytes | 11.00 KiB/s, done.
From https://github.com/malachitos/test
   64a71c9..0f55b53  master     -> origin/master


$ git checkout
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
```

# Fondamenti di Version Control - Pratica

```
$ git switch master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.


$ git fetch
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), 628 bytes | 11.00 KiB/s, done.
From https://github.com/malachitos/test
   64a71c9..0f55b53  master     -> origin/master


$ git checkout
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
```



# Fondamenti di Version Control - Pratica

```
$ git pull
Updating 64a71c9..0f55b53
Fast-forward
 modifiche.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 modifiche.txt
```

Pull per recuperare la nuova versione da remoto

# Fondamenti di Version Control - Pratica

```
$ git pull
Updating 64a71c9..0f55b53
Fast-forward
 modifiche.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 modifiche.txt

$ git switch branch_test_2
Switched to branch 'branch_test_2'
Your branch is up to date with 'origin/branch_test_2'.
```

Pull per recuperare la nuova versione da remoto

Torno sul branch che ho bisogno di allineare a master

# Fondamenti di Version Control - Pratica

```
$ git pull
Updating 64a71c9..0f55b53
Fast-forward
 modifiche.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 modifiche.txt

$ git switch branch_test_2
Switched to branch 'branch_test_2'
Your branch is up to date with 'origin/branch_test_2'.


$ git merge master
Updating 61454f3..0f55b53
Fast-forward
 modifiche.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 modifiche.txt
```

Pull per recuperare la nuova versione da remoto

Torno sul branch che ho bisogno di allineare a master

Merge di master nel mio branch

# Fondamenti di Version Control - Pratica

- Recap veloce, dopo aver fatto le modifiche sul nostro branch abbiamo:

# Fondamenti di Version Control - Pratica

- Recap veloce, dopo aver fatto le modifiche sul nostro branch abbiamo:
  - Aggiunto le nostre modifiche allo **staging**

# Fondamenti di Version Control - Pratica

- Recap veloce, dopo aver fatto le modifiche sul nostro branch abbiamo:
  - Aggiunto le nostre modifiche allo **staging**
  - Fatto il **commit** delle nostre modifiche

# Fondamenti di Version Control - Pratica

- Recap veloce, dopo aver fatto le modifiche sul nostro branch abbiamo:
  - Aggiunto le nostre modifiche allo **staging**
  - Fatto il **commit** delle nostre modifiche
  - Fatto il **push** delle nostre modifiche

# Fondamenti di Version Control - Pratica

- Recap veloce, dopo aver fatto le modifiche sul nostro branch abbiamo:
  - Aggiunto le nostre modifiche allo **staging**
  - Fatto il **commit** delle nostre modifiche
  - Fatto il **push** delle nostre modifiche
- Qualcuno integrerà le modifiche sul master (verificando la **pull request**)

# Fondamenti di Version Control - Pratica

- Recap veloce, dopo aver fatto le modifiche sul nostro branch abbiamo:
  - Aggiunto le nostre modifiche allo **staging**
  - Fatto il **commit** delle nostre modifiche
  - Fatto il **push** delle nostre modifiche
- Qualcuno integrerà le modifiche sul master (verificando la **pull request**)
- Se qualcuno sta lavorando su altri branches o deve modificare dei file che sono stati aggiornati con la modifica appena integrata, dovrà riallineare il branch su cui sta lavorando con la nuova versione in remoto

# Fondamenti di Version Control - Pratica

- Per riallineare il branch su cui sta lavorando con la nuova versione in remoto, bisogna:

# Fondamenti di Version Control - Pratica

- Per riallineare il branch su cui sta lavorando con la nuova versione in remoto, bisogna:
  - Fare lo **switch** a master

# Fondamenti di Version Control - Pratica

- Per riallineare il branch su cui sta lavorando con la nuova versione in remoto, bisogna:
  - Fare lo **switch** a master
  - Fare una **pull** per aggiornare il master in locale

# Fondamenti di Version Control - Pratica

- Per riallineare il branch su cui sta lavorando con la nuova versione in remoto, bisogna:
  - Fare lo **switch** a master
  - Fare una **pull** per aggiornare il master in locale
  - Fare lo **switch** al branch su cui si vuole lavorare

# Fondamenti di Version Control - Pratica

- Per riallineare il branch su cui sta lavorando con la nuova versione in remoto, bisogna:
  - Fare lo **switch** a master
  - Fare una **pull** per aggiornare il master in locale
  - Fare lo **switch** al branch su cui si vuole lavorare
  - Fare il merge di **master** nel branch

# Fondamenti di Version Control - Pratica

- Per riallineare il branch su cui sta lavorando con la nuova versione in remoto, bisogna:
  - Fare lo **switch** a master
  - Fare una **pull** per aggiornare il master in locale
  - Fare lo **switch** al branch su cui si vuole lavorare
  - Fare il **merge** di master nel branch
- Prima di eseguire queste operazioni, è sempre consigliabile fare una fetch in modo da recuperare le informazioni in remoto. In generale, è bene fare una fetch ed eventualmente aggiornare master quotidianamente

# Fondamenti di Version Control - Pratica

- Abbiamo visto il mondo ideale, ora facciamo un esempio più realistico

# Fondamenti di Version Control - Pratica

- Abbiamo visto il mondo ideale, ora facciamo un esempio più realistico
  - Per questioni di tempo, 2 programmatori lavorano sugli stessi sorgenti

# Fondamenti di Version Control - Pratica

- Abbiamo visto il mondo ideale, ora facciamo un esempio più realistico
  - Per questioni di tempo, 2 programmatori lavorano sugli stessi sorgenti
  - Uno dei due finisce le sue implementazioni, fa **commit** e **push**

# Fondamenti di Version Control - Pratica

- Abbiamo visto il mondo ideale, ora facciamo un esempio più realistico
  - Per questioni di tempo, 2 programmatori lavorano sugli stessi sorgenti
  - Uno dei due finisce le sue implementazioni, fa **commit** e **push**
  - Il suo codice viene **integrato** sul master

# Fondamenti di Version Control - Pratica

- Abbiamo visto il mondo ideale, ora facciamo un esempio più realistico
  - Per questioni di tempo, 2 programmatori lavorano sugli stessi sorgenti
  - Uno dei due finisce le sue implementazioni, fa **commit** e **push**
  - Il suo codice viene **integrato** sul master
  - L'altro, che stava già modificando lo stesso sorgente, che deve fare?

# Fondamenti di Version Control - Pratica

- Abbiamo visto il mondo ideale, ora facciamo un esempio più realistico
  - Per questioni di tempo, 2 programmatori lavorano sugli stessi sorgenti
  - Uno dei due finisce le sue implementazioni, fa **commit** e **push**
  - Il suo codice viene **integrato** sul master
  - L'altro, che stava già modificando lo stesso sorgente, che deve fare?
  - IL MERGE

# Fondamenti di Version Control - Pratica

```
$ git switch master
Switched to branch 'master'
M       modifiche.txt
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
```

Switch a master, questa volta mi dice che nello switch dal mio branch a master mi sto portando dietro un file modificato

# Fondamenti di Version Control - Pratica

```
$ git pull
error: Your local changes to the following files would be overwritten by merge:
        modifiche.txt
Please commit your changes or stash them before you merge.
Aborting
Updating 0f55b53..1d6a466
```

# Fondamenti di Version Control - Pratica

```
$ git pull
error: Your local changes to the following files would be overwritten by merge:
        modifiche.txt
Please commit your changes or stash them before you merge.
Aborting
Updating 0f55b53..1d6a466
```

ATTENZIONE!

# Fondamenti di Version Control - Pratica

- Abbiamo 2 strade:

# Fondamenti di Version Control - Pratica

- Abbiamo 2 strade:
  - Il commit (NO!)

# Fondamenti di Version Control - Pratica

- Abbiamo 2 strade:
  - Il **commit** (NO!)
  - Lo **stash** (e che è?)

# Fondamenti di Version Control - Pratica

- Abbiamo 2 strade:
  - Il **commit** (NO!)
  - Lo **stash** (e che è?)
- Lo **stash** (letteramente "messo da parte") è l'operazione con cui si accantonano temporaneamente le modifiche fatte, resettando di fatto il branch all'ultima versione committata.
- Le modifiche vengono "pacchettizzate" e conservate in una specie di array che conserva tutti gli stash creati

# Fondamenti di Version Control - Pratica

```
$ git switch branch_test_3
Switched to branch 'branch_test_3'
M       modifiche.txt
Your branch is up to date with 'origin/branch_test_3'.
```

Sempre meglio fare quest'operazione dal branch su cui abbiamo lavorato

# Fondamenti di Version Control - Pratica

```
$ git switch branch_test_3
Switched to branch 'branch_test_3'
M       modifiche.txt
Your branch is up to date with 'origin/branch_test_3'.


$ git stash push -m "20220107"
Saved working directory and index state On branch_test_3: 20220107
```

Sempre meglio fare quest'operazione dal branch su cui abbiamo lavorato

Creo uno stash e gli do un nome

# Fondamenti di Version Control - Pratica

```
$ git switch branch_test_3
Switched to branch 'branch_test_3'
M       modifiche.txt
Your branch is up to date with 'origin/branch_test_3'.


$ git stash push -m "20220107"
Saved working directory and index state On branch_test_3: 20220107


$ git switch master
Switched to branch 'master'
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
```

Sempre meglio fare quest'operazione dal branch su cui abbiamo lavorato

Creo uno stash e gli do un nome

Torno su master…

# Fondamenti di Version Control - Pratica

```
$ git switch branch_test_3
Switched to branch 'branch_test_3'
M       modifiche.txt
Your branch is up to date with 'origin/branch_test_3'.


$ git stash push -m "20220107"
Saved working directory and index state On branch_test_3: 20220107


$ git switch master
Switched to branch 'master'
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)

$ git pull
Updating 0f55b53..1d6a466
Fast-forward
 modifiche.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Sempre meglio fare quest'operazione dal branch su cui abbiamo lavorato

Creo uno stash e gli do un nome

Torno su master…

…stavolta la pull va a buon fine!

# Fondamenti di Version Control - Pratica

```
$ git switch branch_test_3
Switched to branch 'branch_test_3'
Your branch is up to date with 'origin/branch_test_3'.
```

Torno al mio branch, questa volta non segnala differenze perchè …

# Fondamenti di Version Control - Pratica

```
$ git switch branch_test_3
Switched to branch 'branch_test_3'
Your branch is up to date with 'origin/branch_test_3'.


$ git merge master
Updating 0f55b53..1d6a466
Fast-forward
 modifiche.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)


 $ git stash apply stash@{0}
 Auto-merging modifiche.txt
 CONFLICT (content): Merge conflict in modifiche.txt
```

Torno al mio branch, questa volta non segnala differenze perchè le abbiamo stashate!

Merge del master appena aggiornato sul mio branch

CONFLICT

# Fondamenti di Version Control - Pratica

- Quando si fa un merge che comprende sorgenti modificati da noi che sono anche stati modificati su master, si verifica un conflitto; il conflitto può essere risolto

# Fondamenti di Version Control - Pratica

- Quando si fa un merge che comprende sorgenti modificati da noi che sono anche stati modificati su master, si verifica un conflitto; il conflitto può essere risolto
  - Automaticamente: perfetto, il merge viene concluso in automatico

# Fondamenti di Version Control - Pratica

- Quando si fa un merge che comprende sorgenti modificati da noi che sono anche stati modificati su master, si verifica un conflitto; il conflitto può essere risolto
  - Automaticamente: perfetto, il merge viene concluso in automatico
  - Manuale: come nell'esempio, il conflitto non è risolvibile in automatico, bisogna risolverlo manualmente e segnare il conflitto come "risolto"

# Fondamenti di Version Control - Pratica

- Quando si fa un merge che comprende sorgenti modificati da noi che sono anche stati modificati su master, si verifica un conflitto; il conflitto può essere risolto
  - Automaticamente: perfetto, il merge viene concluso in automatico
  - Manuale: come nell'esempio, il conflitto non è risolvibile in automatico, bisogna risolverlo manualmente e segnare il conflitto come "risolto"
    - ATTENZIONE: una volta che il conflitto è segnato come risolto, una volta committato, quella è la versione ufficiale! Se la portiamo su master, **sovrascriverà sempre e comunque la versione precedente**.

# Fondamenti di Version Control - Pratica

- Un file in conflitto verrà "marchiato" così:

# Fondamenti di Version Control - Pratica

- Un file in conflitto verrà "marchiato" così:

```
<<<<<<< Updated upstream
queste sono le mie modifiche aggiornate da branch 2
=======
queste sono le mie modifiche aggiornate da branch 3
>>>>>>> Stashed changes
```

# Fondamenti di Version Control - Pratica

- Un file in conflitto verrà "marchiato" così:

```
<<<<<<< Updated upstream
queste sono le mie modifiche aggiornate da branch 2
=======
queste sono le mie modifiche aggiornate da branch 3
>>>>>>> Stashed changes
```

- Tutto ciò tra "Updated upstream" e la sequenza ====== è il codice attualmente è presente sul branch, tutto ciò tra la sequenza ====== e "Stashed upstream" è il codice presente sullo stash

# Fondamenti di Version Control - Pratica

- Un file in conflitto verrà "marchiato" così:

```
<<<<<<< Updated upstream
queste sono le mie modifiche aggiornate da branch 2
=======
queste sono le mie modifiche aggiornate da branch 3
>>>>>>> Stashed changes
```

- Tutto ciò tra "Updated upstream" e la sequenza ====== è il codice attualmente è presente sul branch, tutto ciò tra la sequenza ====== e "Stashed upstream" è il codice presente sullo stash
- Faccio il merge manuale, il risultato finale è questo:

# Fondamenti di Version Control - Pratica

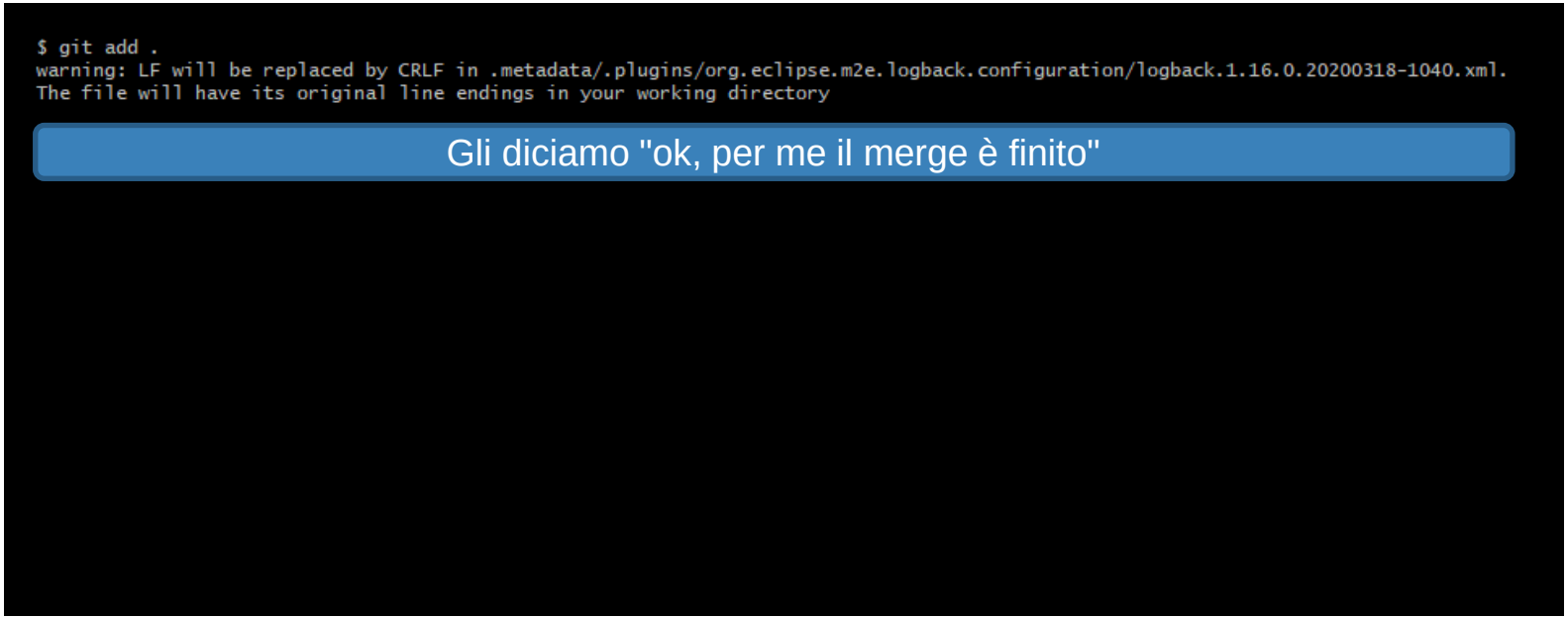
- Un file in conflitto verrà "marchiato" così:

```
<<<<<<< Updated upstream
queste sono le mie modifiche aggiornate da branch 2
=======
queste sono le mie modifiche aggiornate da branch 3
>>>>>>> Stashed changes
```

- Tutto ciò tra "Updated upstream" e la sequenza ====== è il codice attualmente è presente sul branch, tutto ciò tra la sequenza ====== e "Stashed upstream" è il codice presente sullo stash
- Faccio il merge manuale, il risultato finale è questo:

```
queste sono le mie modifiche aggiornate da branch 2 e mergeate con branch 3
```

# Fondamenti di Version Control - Pratica



# Fondamenti di Version Control - Pratica

```
$ git add .
warning: LF will be replaced by CRLF in .metadata/.plugins/org.eclipse.m2e.logback.configuration/logback.1.16.0.20200318-1040.xml.
The file will have its original line endings in your working directory
```

Gli diciamo "ok, per me il merge è finito"

```
$ git commit -m "aggiornato il modifiche.txt"
[branch_test_3 f11f8f9] aggiornato il modifiche.txt
 38 files changed, 3115 insertions(+), 1 deletion(-)
 create mode 100644 .metadata/.lock
```

Commit OK

# Fondamenti di Version Control - Pratica

```
$ git add .
warning: LF will be replaced by CRLF in .metadata/.plugins/org.eclipse.m2e.logback.configuration/logback.1.16.0.20200318-1040.xml.
The file will have its original line endings in your working directory
```

Gli diciamo "ok, per me il merge è finito"

```
$ git commit -m "aggiornato il modifiche.txt"
[branch_test_3 f11f8f9] aggiornato il modifiche.txt
 38 files changed, 3115 insertions(+), 1 deletion(-)
 create mode 100644 .metadata/.lock
```

Commit OK

```
$ git push
Enumerating objects: 58, done.
Counting objects: 100% (58/58), done.
Delta compression using up to 4 threads
Compressing objects: 100% (42/42), done.
Writing objects: 100% (56/56), 63.38 KiB | 1.47 MiB/s, done.
Total 56 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/malachitos/test.git
   0f55b53..f11f8f9  branch_test_3 -> branch_test_3
```

Push OK

ITS
TECNOLOGIE
DELL'INFORMAZIONE E
DELLA COMUNICAZIONE

POR Piemonte
FSE 2014-2020

ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

# GRAZIE!

POR Piemonte FSE 2014-2020