

# VCS-GIT-LAB-6

## Branch & more

**MARCELLO MISSIROLI**  
**Tecnologia  
e Progettazione**  
per il mondo **digitale**  
e per il **web II**



 digital docet

# Prerequisiti

## COMPUTER DI LABORATORIO O LAPTOP PERSONALE

Partendo dai laboratori precedenti, dovrete tutti avere:

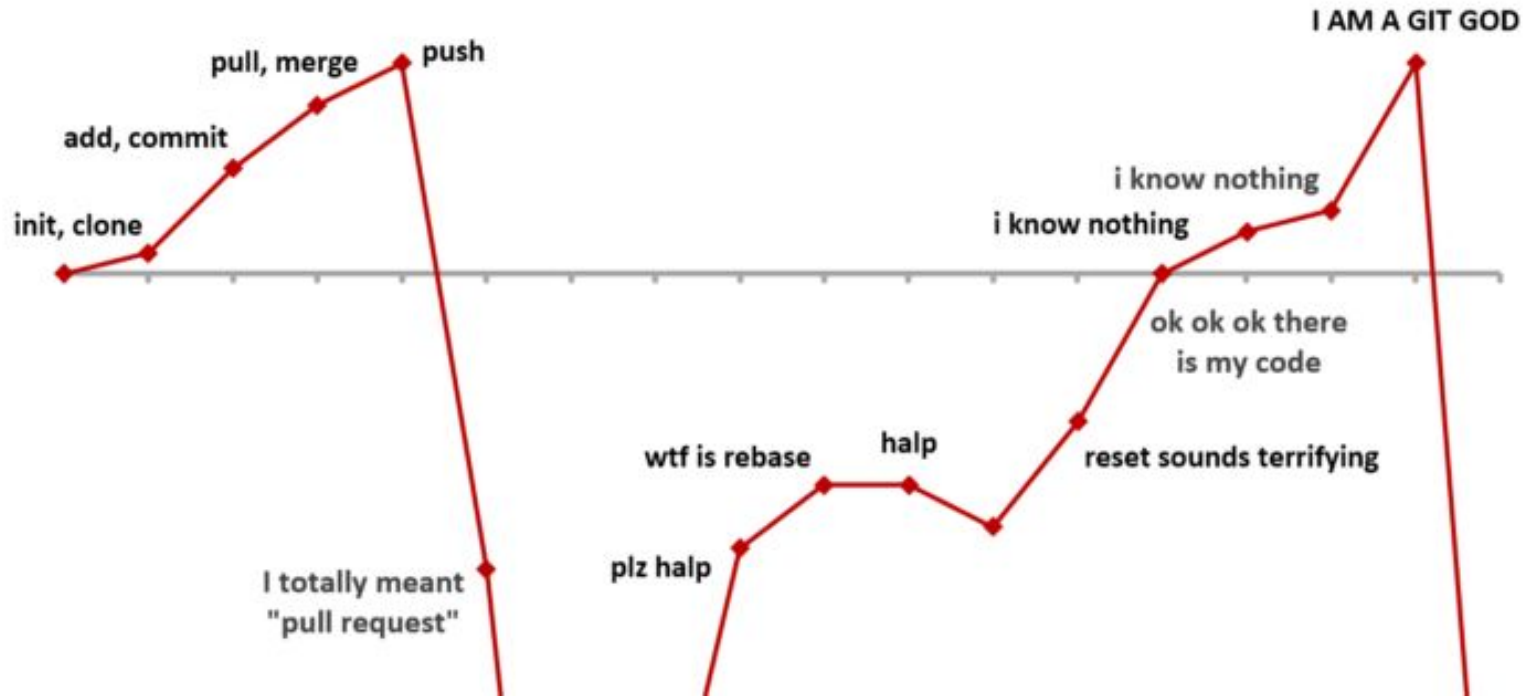
- ▶ Git installato
- ▶ Un repo git locale
- ▶ Un account su Gitlab
- ▶ Un repo remoto collegato al repo locale
- ▶ Costituzione di coppie Capo-Dipendente
- ▶ Eclipse installato localmente (l'esempio prevede Java, ma funziona con qualsiasi altra tipologia di Eclipse)
  - ▷ In alternativa potete provare a usare un altro IDE e “scoprire” i comandi.

**LA DURATA PREVISTA DI QUESTA ATTIVITÀ È DI 1 ORA DI 50 MINUTI.**

## Test (A2 level)

- ▶ Creare un nuovo progetto git “anagrafica”
- ▶ Creare un nuovo file chiamato con con il vostro nome
- ▶ Un maintainer crei un nuovo repo privato, chiamato “anagrafica”
- ▶ Ciascuno aggiunga al proprio repo locale il remoto e sincronizzi
- ▶ (SYNC)
- ▶ Creare un file chiamato anagrafica.md, e dentro mettete il vostro nome seguito dalla data di nascita. Committate
- ▶ L’obiettivo è quello di ottenere una lista dei vostri nomi ordinati anagraficamente (crescente)
- ▶ Pushare, pullare e risolvere i conflitti sino al raggiungimento dell’obiettivo. Idealmente senza parlare.

“



”

1.

# Branching – 101

Partiamo piano

## Create una branch chiamata feature-haiku

Create il branch, scrivete un haiku nel file haiku.txt, committate. Quindi controllate il log (reflog -n 2)

```
$prova@pc:~/demo$ git checkout -b feature-haiku
Si è passati a un nuovo branch 'feature-haiku'
$prova@pc:~/demo$ [nano | gedit | notepad] haiku.txt

$prova@pc:~/demo$ git add .; git commit -m "creazione haiku"
[feature-haiku 4360427] creazione haiku
 1 file changed, 4 insertions(+)
 create mode 100644 haiku.txt
$prova@pc:~/demo$ git reflog -n 2
4360427 (HEAD -> feature-haiku) HEAD@{0}: commit: creazione
haiku
d19a0f3 (origin/main, main) HEAD@{1}: checkout: moving from
main to feature-haiku
```

Esempio di haiku:

*Vecchio stagno  
Una rana si tuffa  
Rumore d'acqua*

## Prima fusione

Tornate al branch principale, verificate che il file haiku.txt non è presente. Quindi fare il merge.

```
$prova@pc:~/demo$ git checkout main
Si è passati a un nuovo branch 'main'.
Il tuo branch è aggiornato rispetto a 'origin/main'.
$prova@pc:~/demo$ git merge feature-haiku
Aggiornamento di d19a0f3..4360427
Fast-forward
 haiku.txt | 4 ++++
 1 file changed, 4 insertions(+)
 create mode 100644 haiku.txt
$prova@pc:~/demo$
```

## Continuare lo sviluppo.

- ▶ Modificate sul branch main il file haiku.txt (la seconda riga, per esempio una rana → un rospo), salvate MA NON FATE IL COMMIT.
- ▶ fare il checkout di feature-haiku
- ▶ modificare il testo (per esempio stagno → lago)
- ▶ Salvare
- ▶ Committare.



## Seconda fusione (con problemi)

Tornate al branch principale e tentate il merge.

```
$prova@pc:~/demo$ git checkout main
Si è passati al branch 'main'.
$prova@pc:~/demo$ git merge feature-haiku
Aggiornamento di 4360427..9a3f59c
error: Le tue modifiche locali ai seguenti file sarebbero
sovrascritte con il merge:
    haiku.txt
Esegui il commit o lo stash delle modifiche prima di eseguire
il merge.
Interrompo l'operazion

$prova@pc:~/demo$
```

## Per risolvere

...git evita di sovrascrivere le modifiche locali.

Possibili soluzioni:

- ▶ Committare le modifiche
- ▶ Eliminare le modifiche (`git reset --HARD`)
- ▶ “Nascondere le modifiche sotto il tappeto” con `git stash`
- ▶ Procedere quindi al merge come al solito.

## Verifica

```
$prova@pc:git stash
Directory di lavoro e stato indice salvati: WIP on main:
4360427 creazione haiku
prova@SILENCIO2:~/demo$ git merge feature-haiku
Aggiornamento di 4360427..9a3f59c
Fast-forward
  haiku.txt | 2 +-
  1 file changed, 1 insertion(+), 1 deletion(-)
prova@SILENCIO2:~/demo$ git stash pop
Merge automatico di haiku.txt in corso
CONFLITTO (contenuto): conflitto di merge in haiku.txt
La voce di stash è mantenuta nel caso in cui tu ne abbia
nuovamente bisogno.
```

## Soluzione

Risolvete il conflitto e eliminate il vostro stash

```
$prova@pc:~/demo$ git add haiku.txt ; git commit -m "risolto"
[main ec9c03c] risolto
 1 file changed, 1 insertion(+), 1 deletion(-)
prova@SILENCIO2:~/demo$ git stash drop
Ho scartato refs/stash@{0}
(bc1d7c67544f690766395ce999694633f3b61421)
$prova@pc:~/demo$
```

## Branch remoto

Tornate al branch feature-haiku e pushate il branch su origin.

```
$prova@pc:~/demo$ git push -u origin feature-haiku
Enumerazione degli oggetti in corso: 34, fatto.
Conteggio degli oggetti in corso: 100% (34/34), fatto.
Compressione delta in corso, uso fino a 8 thread
Compressione oggetti in corso: 100% (22/22), fatto.
Scrittura degli oggetti in corso: 100% (34/34), 5.40 KiB |
1.80 MiB/s, fatto.
34 oggetti totali (5 delta), 0 riutilizzati (0 delta)
remote:
To https://gitlab.com/alex/vostronome.git
 * [new branch]      feature-haiku -> feature-haiku
Branch 'feature-haiku' impostato per tracciare il branch
remoto 'feature-haiku' da 'origin'.
$prova@pc:~/demo$
```

## Tagging

I tag sono un modo semplice per “marcare” un particolare commit. Ad esempio milestones, fine sprint, o particolari eventi.

Gitlab permette di accedere istantaneamente alle versioni taggate del repo.

Sono molto semplici da usare, ma possono avere qualche trucco nascosto.

## Tagging: due tipi

Git supporta due tipi di tag: leggeri (lightweight) e annotato (annotated)

Il primo è semplicemente un puntatore a un particolare commit. Analogo allo snapshot di un filesystem.

Il secondo è più complesso, ha un messaggio associato, può essere firmato tramite PC e altro.

## Tagging: lightweight

```
git tag pippo
```

Aggiunge il tag. Con il comando `git tag` si elencano tutti i tag. E' possibile fare il checkout diretto usando il tag.



## Tagging: annotated

`git tag -a pluto -`  
“messaggio”

Aggiunge il tag annotato.  
Oltre alle funzionalità già  
viste, il comando `git`  
`show` mostra parecchie  
informazioni

```
$ git show pluto
tag pluto
Tagger: Walt Disney <walt@disney.com>
Date:   Sat May 3 20:19:12 2014 -0700
```

messaggio

```
commit
ca82a6dff817ec66f44342007202690a93763949
Author: Matt Dillon <matt@disney.com>
Date:   Mon Mar 17 21:52:11 2008 -0700
```

Commento al commit

Contrariamente al solito, i tag non sono pushati automaticamente, perché pensati a livello personale. Per pusharli occorre il comando preciso:

```
git push origin <tagname>.
```

**oppure**

```
git push origin --tags
```

## Capo tagger

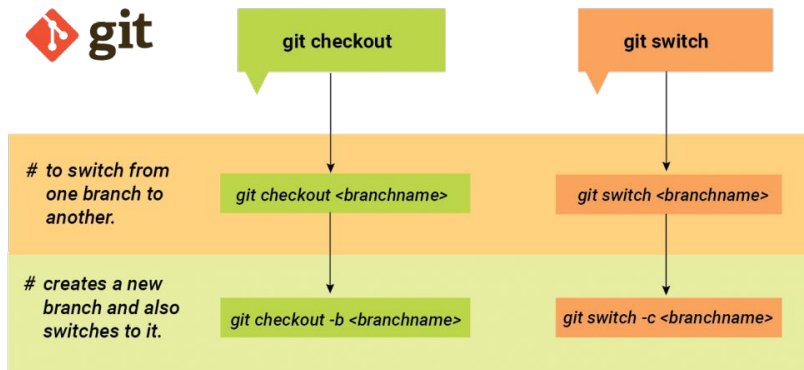
Il responsabile del progetto ora taggi il repository come “betarelease”, e pushi sul remoto il tag.

Che accade su gitlab?

# Git switch

Switch è un nuovo comando che si aggiunge a checkout. Questione di gusti

Utile soprattutto nella sua forma `git switch` - che permette di tornare al branch precedente.



“

## *Checkpoint #1:*

- ▶ *Sapete usare i branch e merge*
- ▶ *Sapete creare un branch remoto*
- ▶ *Sapete usare stash*
- ▶ *Sapete usare i tag*
- ▶ *Avete utilizzato il feature branching flow*

”

## 2. Eclipse

## Integrazione IDE

Come sappiamo, tutti gli IDE maggiori supportano Git in un modo o nell'altro: Netbeans, Visual Studio, la serie IntelliJ. Non possiamo vederli tutti, per cui ci concentreremo su Eclipse, IDE molto diffuso che presenta qualche particolarità.

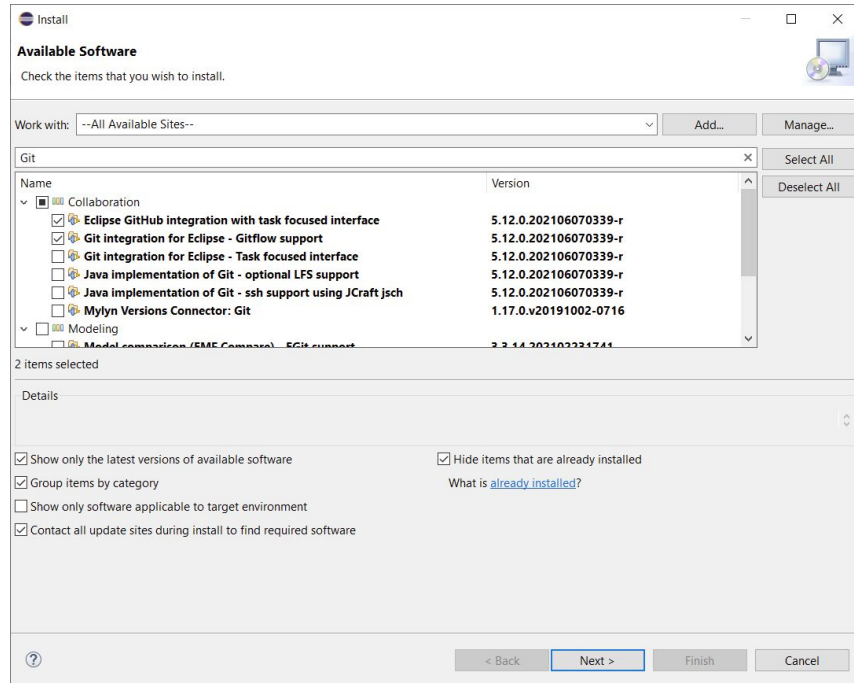
# Tutti: installazione

Git richiede l'installazione di un Plugin. Per farlo andate su

Help >

Install new software >

Quindi scegliete "All available sites" dal menu a tendina e scrivete "Git"

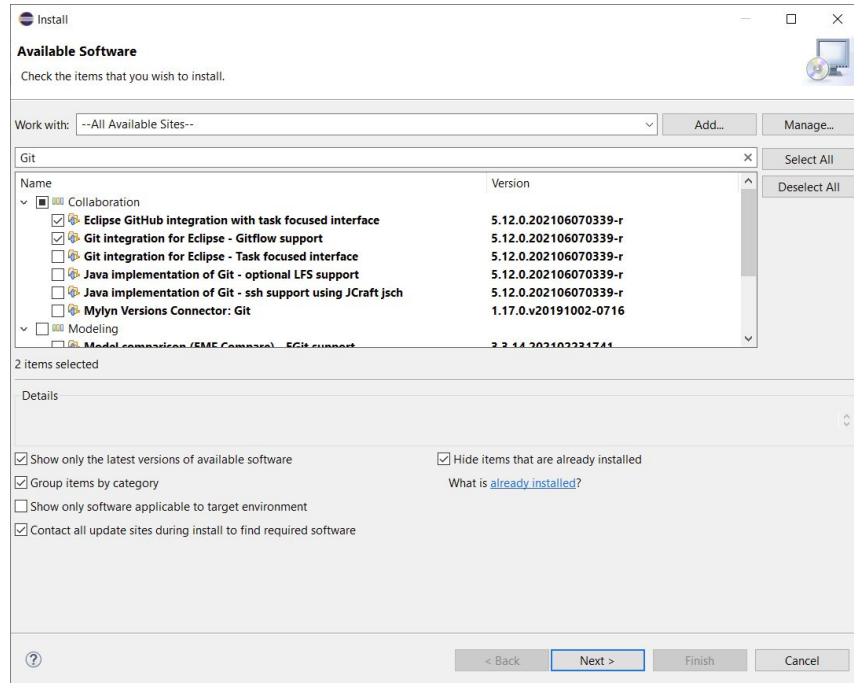




## Tutti: installazione

Selezionate il primo (e opzionalmente il secondo) quindi date una serie di “Next” e accettate tutto.

Il plugin si scarica, e dovrete riavviare Eclipse.



## Piano del workshop

Parte 1: uso di Git in locale (Capo)

Parte 2: uso di Git con Gitlab (Capo)

Parte 3: Forking e Cloning (Dipendente)

Parte 4: Branching e Merge request (Dipendente)

Parte 5: Accettazione Merge request (Capo).

## Parte 1 (Capo): git in locale

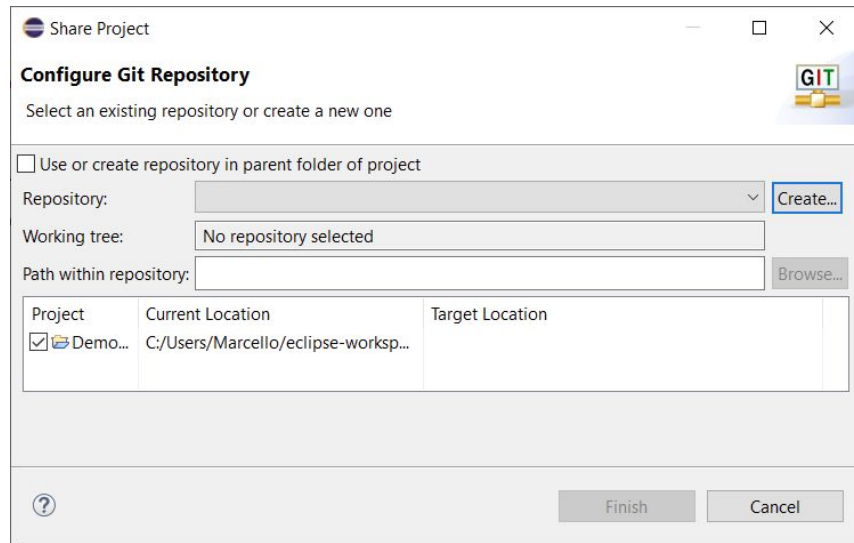
Creare un semplice progetto Java. Creare un oggetto Contatore e iniziarlo con questo codice.

<https://pastebin.com/Nks5eFWB>

Compilare per controllare sia tutto OK

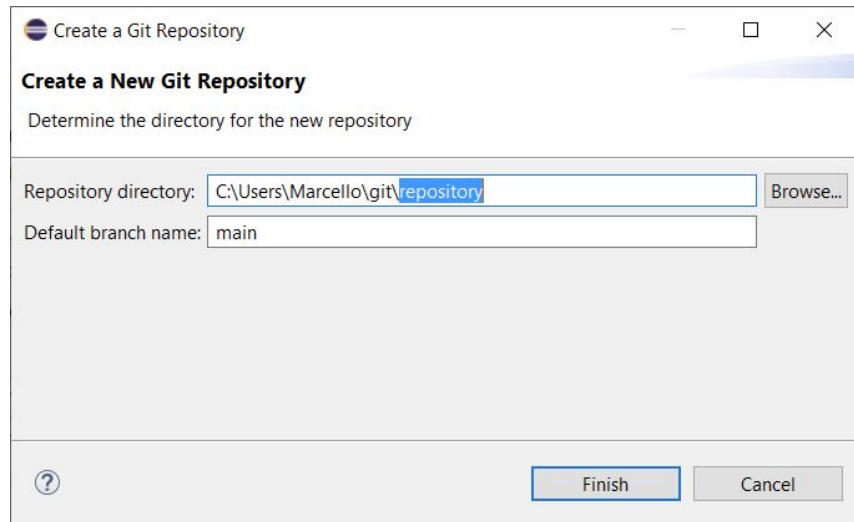
## Parte 1 (Capo): git in locale - git init

Per inizializzare il repo,  
fare click destro sul  
progetto quindi Team >  
Share Project



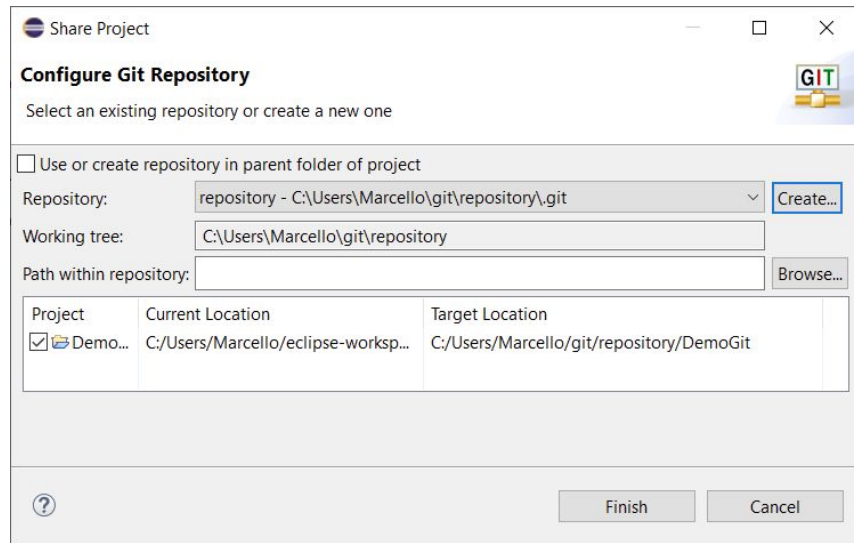
## Parte 1 (Capo): git in locale – git init (2)

Premete “Create” e scegliete un posto dove memorizzerete tutti i vostri repo Git (A Eclipse piace così).



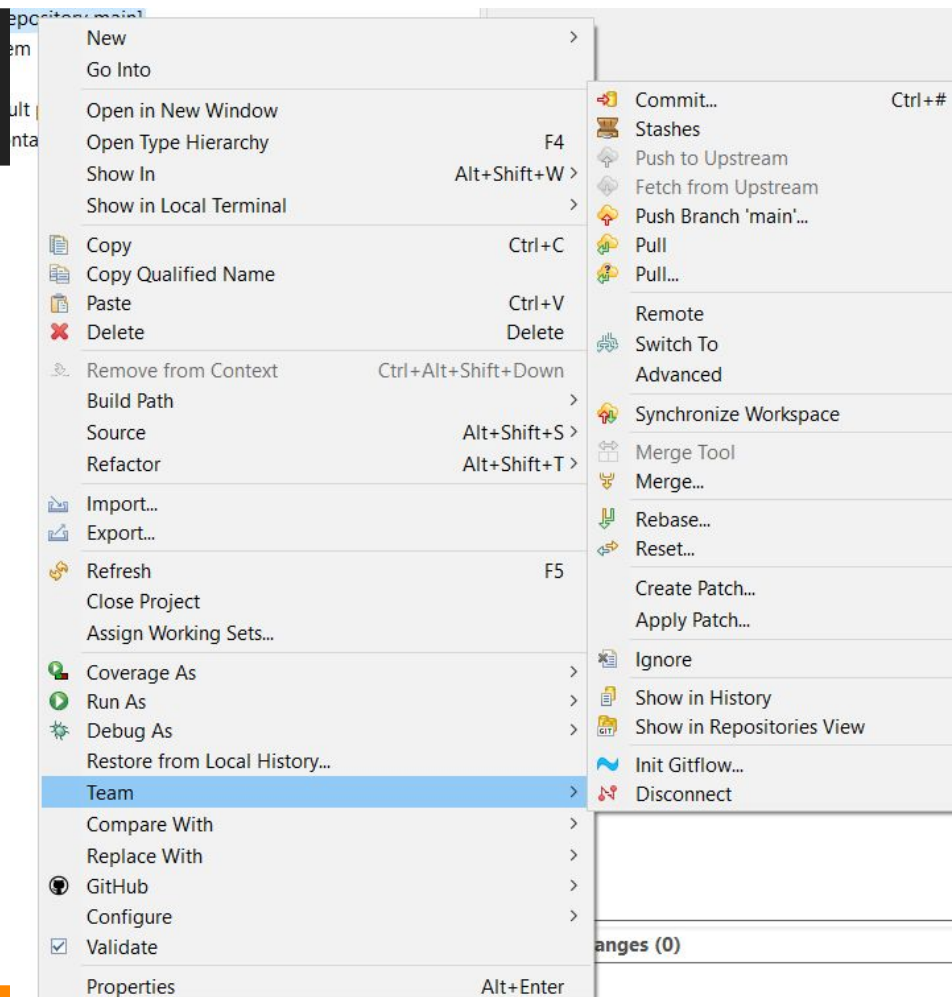
## Parte 1 (Capo): git in locale - git init (3)

Ora premete [Finish] e siete a posto. Il repo è stato inizializzato il progetto SI SPOSTA nella locazione indicata. Aguzzando la vista vedrete delle piccole “?” sotto ogni file, che indicano che sono file non tracciati

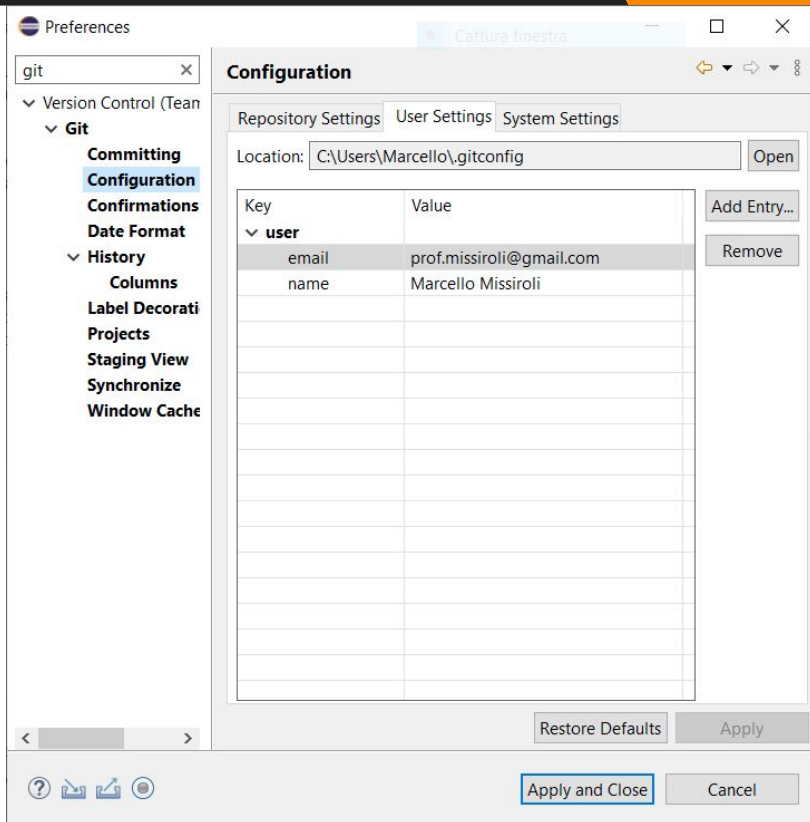


## Menu git

Facendo ora click destro sul progetto, appare il menu che permette di accedere a tutti i comandi git in modo grafico.



Nelle preferenze, selezionando Git, si possono modificare in modo semplice le configurazioni.





## Parte 1 (Capo): git in locale – Primo commit

Fate click destro sul progetto > Team

Add to index

(questo è di fatto equivalente a git add .)

Poi selezionate commit.

# Parte 1 (Capo): git in locale – Primo commit

Appare la “Staging view” che mostra cosa farà il commit. Dopo il controllo, premere [Commit]. Il repo è pronto all’uso.

The screenshot shows the Eclipse IDE's Git Staging view. The top bar includes tabs for Problems, Javadoc, Declaration, Console, and Git Staging. The main area is divided into three sections:

- Unstaged Changes (0):** An empty box indicating no unstaged changes.
- Staged Changes (5):** A list of files staged for commit:
  - .classpath - DemoGit
  - .gitignore - DemoGit
  - .project - DemoGit
  - Contatore.java - DemoGit/src
  - org.eclipse.jdt.core.prefs - DemoGit/.settings
- Commit Message:** A section for writing the commit message. It includes an information icon and the text: "Unborn branch: this commit will create the branch 'main'." Below this is a text area containing the message "Commit iniziale".

At the bottom, there are input fields for the commit metadata:

- Author:** Marcello Missiroli <prof.missiroli@gmail.com>
- Committer:** Marcello Missiroli <prof.missiroli@gmail.com>

At the bottom right, there are two buttons: "Commit and Push..." and "Commit".

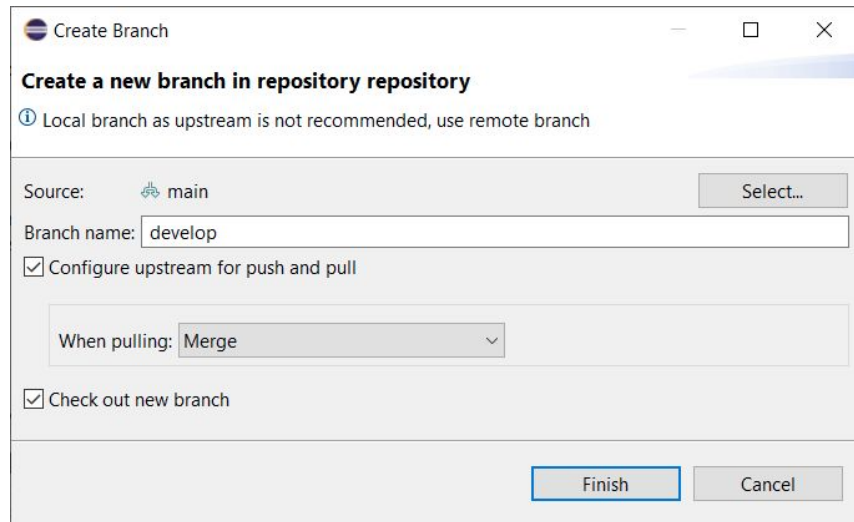
## Parte 2 (Capo): git & Gitlab - Branch di sviluppo

Aggiungiamo un branch di sviluppo.

Team > Switch to > New Branch

Segnando la voce “Configure upstream” si inizializza anche il branch su Gitlab

Segnando la voce “Check out new branch” si passa direttamente al nuovo branch

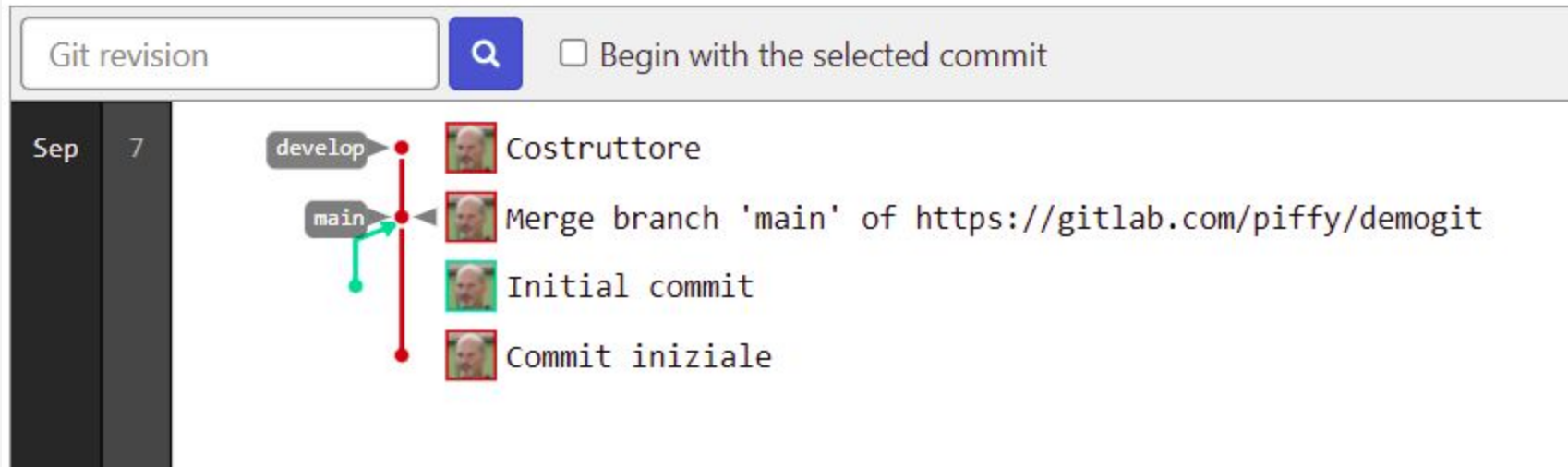


## Parte 2 (Capo): git & Gitlab - Commit su develop

Nel branch develop, aggiungete un costruttore all'oggetto, salvate e committate.



## Parte 2 (Capo): git & Gitlab - Branch di sviluppo



Segnando la voce "Check out new branch" si passa direttamente al nuovo branch

## Collegiamo al remoto 3:

Risultato: qualcosa di simile a questo (dopo avervi chiesto la password)

```
Enumerazione degli oggetti in corso: 17, fatto.  
Conteggio degli oggetti in corso: 100% (17/17), fatto.  
Compressione delta in corso, uso fino a 8 thread  
Compressione oggetti in corso: 100% (9/9), fatto.  
Scrittura degli oggetti in corso: 100% (17/17), 1.37 KiB |  
1.37 MiB/s, fatto.  
17 oggetti totali (0 delta), 0 riutilizzati (0 delta)  
[...]  
To https://gitlab.com/piffy/vostronome.git  
* [new branch]      main -> main  
Branch 'main' impostato per tracciare il branch remoto 'main'  
da 'origin'
```

Pulliamo:

Dare i seguenti comandi e controllate su Github:

```
$prova@pc:~/demo$  
  
git remote add origin [URL DI  
GITLAB]  
  
git push -u origin --all --force  
  
$prova@pc:~/demo$
```

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

Name	Last commit
 temp	gitignore
 .gitignore	gitignore
 README.md	Initial commit
 altrodemo.txt	secondo file
 demo.txt	terzo commit

# Troubleshooting

- ▶ Sul sito sono presenti due branch (main e master)?
  - ▷ `git checkout main`
  - ▷ `git merge master`  
`--allow-unrelated-histories`
  - ▷ `git branch -d master`

Quindi cancellare il branch obsoleto su gitlab con `git push origin --delete master` oppure

Marcello Missiroli > Vostronome > Repository > **Branches**

**Overview** Active Stale All

Filter by branch name



Delete merged branches

New branch

Protected branches can be managed in [project settings](#).

Active branches



## Evitiamo problemi

- ▶ In questa configurazione avete incluso nel repo tutte le informazioni del progetto, incluso versioni della JDK, variabili di sistema.
- ▶ Nel caso cloniate su macchine non perfettamente identiche si possono verificare problemi
- ▶ Meglio rimuovere dai progetti le informazioni specifiche del sistema, dell'IDE e altro.
- ▶

## Evitiamo problemi

- ▶ Modificare .gitignore
  - ▷ # Eclipse Core
  - ▷ .project
  - ▷ # Eclipse Java Development Tools
  - ▷ .classpath
  - ▷ .class
- ▶ Quindi
  - ▷ `git rm --cached .project`
  - ▷ `git rm --cached .classpath`
  - ▷ `git add .gitignore; git commit -m "fix"`

“

## *Checkpoint #2:*

- ▶ *Sapete inizializzare un repo remoto*
- ▶ *Sapete collegarlo a un repo esistente*
- ▶ *Sapete tenere sincronizzati i due repo*

”

## 2. Forking & Cloning

## Forking


Potreste non aver voglia di cedere il controllo totale a un'altra persona.

Oppure, vorreste voler lavorare su codice di cui non avete diritto in scrittura.

In questi casi, Fork & clone è la soluzione


# Dipendente: trovare il sito del capo e forkare



 **Demogit**   
Project ID: 29429675 [Request Access](#)



  Star 0  Fork 0



The Alternate Prof > Demogit

 **Demogit**   
Project ID: 29470670

  Star 0  Fork 0

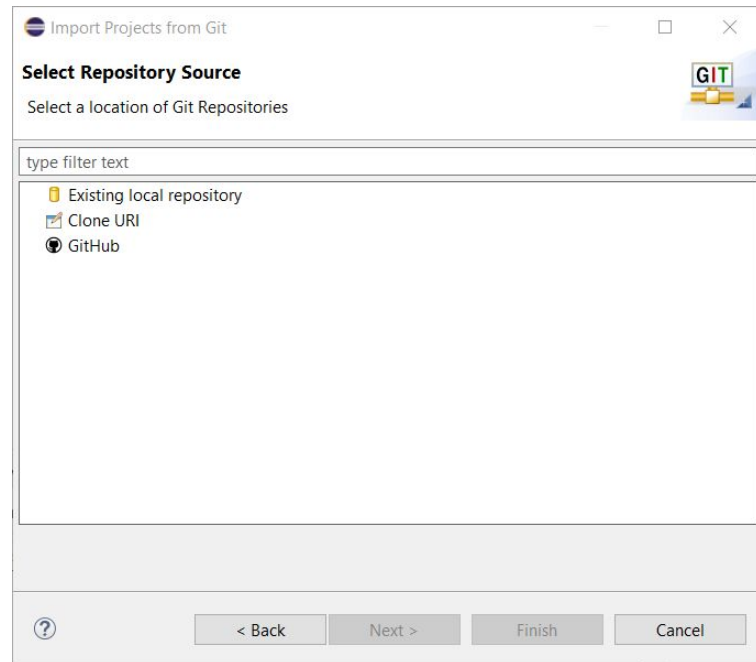
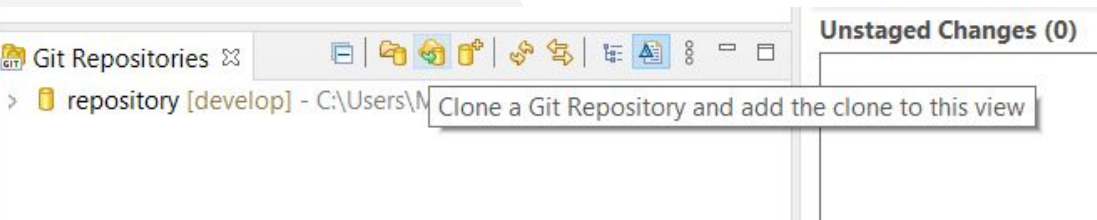
 3 Commits  2 Branches  0 Tags  143 KB Files  256 KB Storage

Forked from [Marcello Missiroli / Demogit](#)

## Dipendente: clonare con Eclipse

Strada 1: File → Import  
→ Git → Projects from  
Git

Strada due: dalla git  
perspective:



## Dipendente: clonare con Eclipse (2)

Inserite l'URI: il dialogo si popolerà

Import Projects from Git

**Source Git Repository**

Enter the location of the source repository.

Location

URI:  Local Folder... Local Bundle File...

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

☐ Store in Secure Store

< Back Next > Finish Cancel

Import Projects from Git

**Source Git Repository**

Enter the location of the source repository.

Location

URI:  Local Folder... Local Bundle File...

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

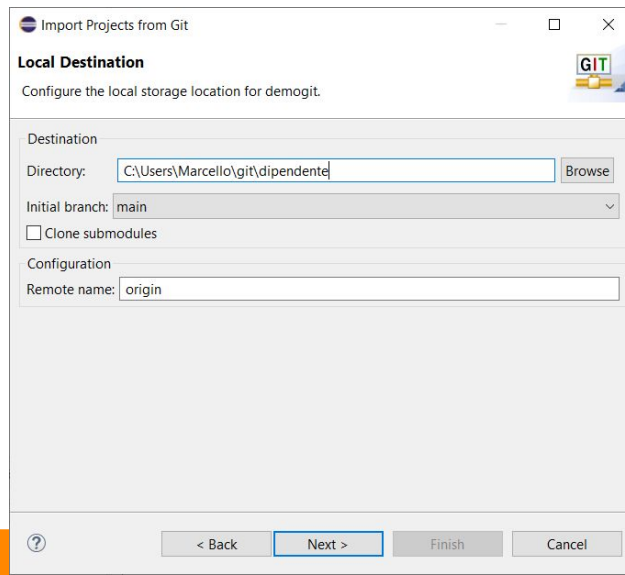
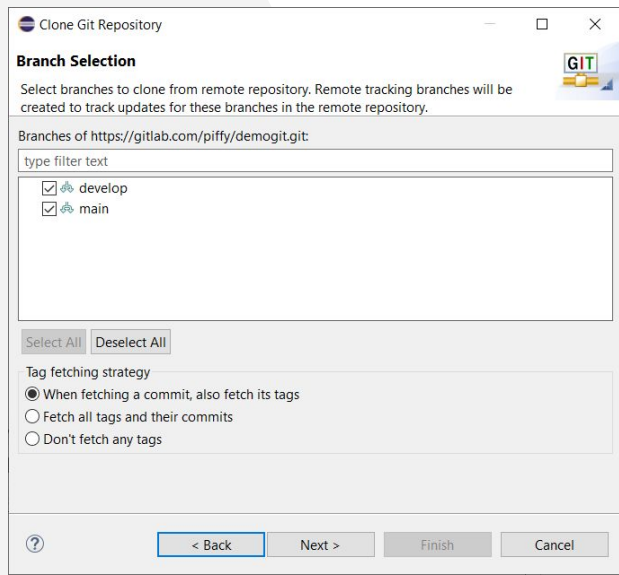
☒ Store in Secure Store

< Back Next > Finish Cancel

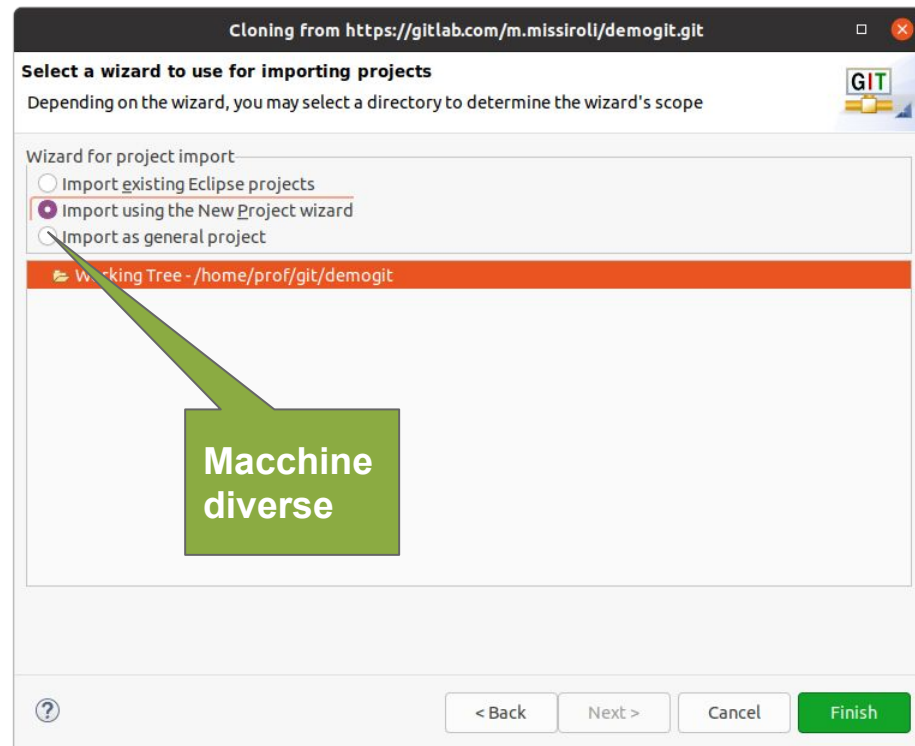
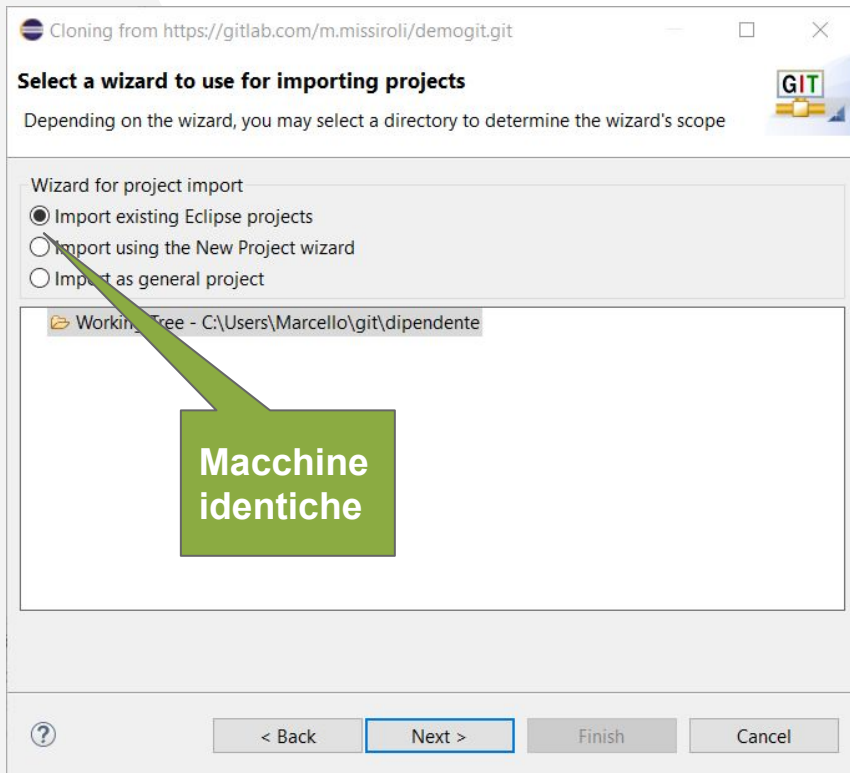


## Dipendente: clonare con Eclipse (3)

Scegliete le branch (generalmente tutte) e il nome di destinazione



# Dipendente: clonare con Eclipse (4)



“

### *Checkpoint #3:*

- ▶ *Sapete forkare un repo su Gitlab*
- ▶ *Sapete clonare un repo da Gitlab*

”

4.

# Conflitti in Eclipse

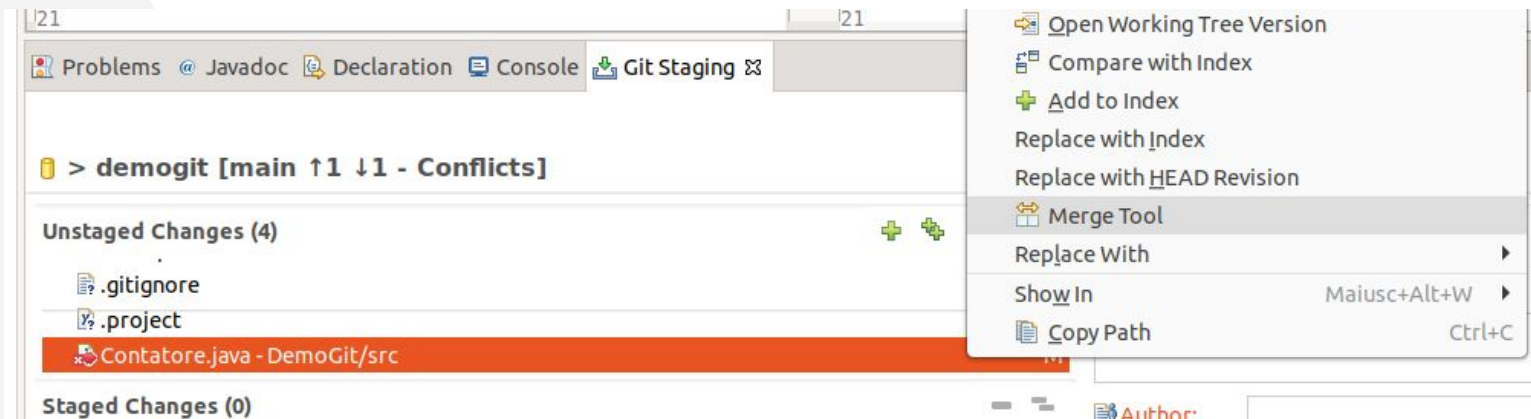
Più facile di prima

## Dipendente: Simuliamo un conflitto

- ▶ Su Gitlab, editate il file Contatore.java con il web IDE.
- ▶ Sotto la riga 15, scrivete:  
`c.setValore(6);`
- ▶ Committate con il pulsante Commit
- ▶ Localmente invece scrivete invece allo stesso posto  
`c.setValore(4);`
- ▶ Committate localmente poi fate Pull

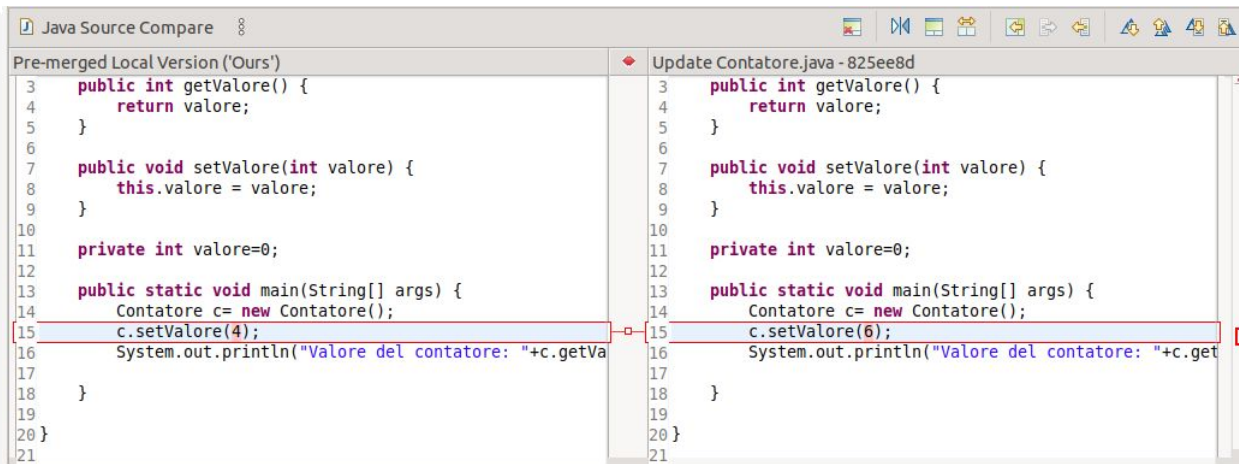
## Dipendente: risolviamo un conflitto

- ▶ La freccia rossa indica conflitto.
- ▶ Fate click destro → Merge tool



## Dipendente: risolviamo un conflitto

- ▶ Appare il confronto diretto tra le due versioni.
- ▶ Potete modificare la vostra versione (o usare i pulsanti per accettare le modifiche). Poi add /add to Index, commitare e pushare le modifiche



The screenshot shows the 'Java Source Compare' window with two panels. The left panel is titled 'Pre-merged Local Version ('Ours')' and the right panel is titled 'Update Contatore.java - 82See8d'. Both panels show the same Java code for a class named 'Contatore'. The code includes methods 'getValore()', 'setValore()', a private field 'valore', and a static 'main' method. A conflict is highlighted on line 15, where the left version has 'c.setValore(4);' and the right version has 'c.setValore(6);'. A red box and a conflict icon (two overlapping squares) are shown on the right side of the code line in the right panel.

```
Java Source Compare

Pre-merged Local Version ('Ours')
3 public int getValore() {
4     return valore;
5 }
6
7 public void setValore(int valore) {
8     this.valore = valore;
9 }
10
11 private int valore=0;
12
13 public static void main(String[] args) {
14     Contatore c= new Contatore();
15     c.setValore(4);
16     System.out.println("Valore del contatore: "+c.getVa
17 }
18
19
20 }
21

Update Contatore.java - 82See8d
3 public int getValore() {
4     return valore;
5 }
6
7 public void setValore(int valore) {
8     this.valore = valore;
9 }
10
11 private int valore=0;
12
13 public static void main(String[] args) {
14     Contatore c= new Contatore();
15     c.setValore(6);
16     System.out.println("Valore del contatore: "+c.get
17 }
18
19
20 }
21
```

“

*Checkpoint #4:*

- ▶ *Sapete usare il merge tool*

”



4.

Merge requests

## Dipendente: proposta di merge

Il dipendente vuole che la sua bellissima modifica sia incorporata nel repo del capo.

Per questo, usa le merge request (pull request in GitHub).

Fare clic su merge request (a sinistra)

## Dipendente: proposta di merge

Il dipendente vuole che la sua bellissima modifica sia incorporata nel repo del capo.

Per questo, usa le merge request (pull request in GitHub).

Fare clic su merge request (a sinistra)

# Dipendente: proposta di merge

› Merge requests

Click su [New Merge Request]



**Merge requests are a place to propose changes you've made to a project and discuss those changes with others**

Interested parties can even contribute by pushing commits if they want to.

New merge request

## Dipendente: proposta di merge

Selezionare a sinistra la branch locale, a destra il repository upstream e la branch. Click su [Compare branch and continue]

The Alternate Prof > Demogit > Merge requests > New

### New merge request

#### Source branch

m.missiroli/demogit

main



#### Update Contatore.java

The Alternate Prof authored 35 minutes ago

825ee8dd



#### Target branch

piffy/demogit

main


Compare branches and continue

# Dipendente: proposta di merge

## Compilare i vari campi e premere [Create merge request]

### New merge request

From `m.missiroli/demogit:main` into `piffy/demogit:main` [Change branches](#)

 This merge request is from a private project to a public project.  
Review the target project before submitting to avoid exposing private changes.

Title

[Start the title with Draft:](#) to prevent a merge request that is a work in progress from being merged before it's ready.

Description

**Write** Preview

**B** *I* **»** **</>**      

Una prima prova di Merge Request

[Markdown and quick actions](#) are supported

 [Attach a file](#)

Merge options

☐ Squash commits when merge request is accepted. 

# Dipendente: proposta di merge

A posto!

Open

Created just now by  The Alternate Prof 

Edit

Mark as draft



## Modifica valore al contatore


Overview 0

Commits 4

Changes 4

Una prima prova di Merge Request



Request to merge `m.missiroli:main`  into `main`

Open in Web IDE

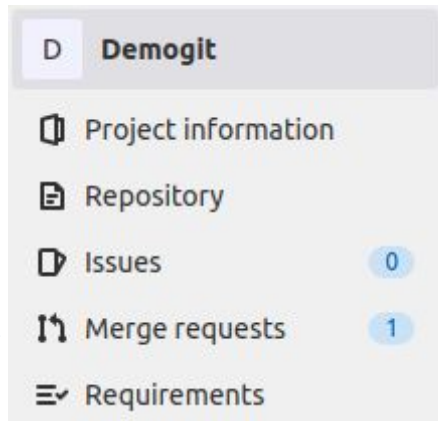


Check out branch



## Capo: controllare le richieste

Chi riceve a merge request viene avvertito in questo modo (si possono ricevere anche email specifiche)



Marcello Missiroli > Demogit > Merge requests

Open 1 Merged 0 Closed 0 All 1



Edit merge requests

New merge request

Recent searches ▾

Search or filter results...

Created date ▾



Modifica valore al contatore

11 · created 12 minutes ago by The Alternate Prof



updated 12 minutes ago



## Capo: controllare le richieste

Facendo click sul nome appare un riassunto della modifica.

Si può esaminare in dettaglio premendo “Changes”

### Modifica valore al contatore

Overview 0 Commits 4 Changes 4

Una prima prova di Merge Request

Request to merge `m.missiroli:main` into `main`

Open in Web IDE

Check out branch



Approve

Approval is optional



[View eligible approvers](#)



Merge

☐ Squash commits



4 commits and 1 merge commit will be added to main. [Modify merge commit](#)



0



0



Oldest first

Show all activity

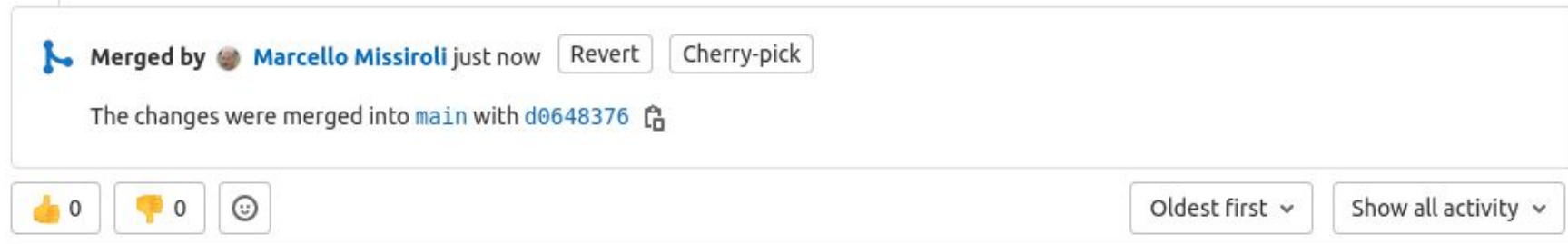
# Capo: controllare le richieste

Si possono esaminare le modifiche....



## Capo: controllare le richieste

Si può rifiutare la richiesta (con messaggio) o approvarla (come in questo caso) con un merge automatico.



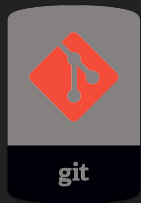
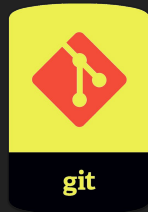
“

## *Checkpoint #5:*

- ▶ *Sapete fare una merge request*
- ▶ *Sapete approvare una merge request*
- ▶ *Avete visto un (parziale) esempio di forking Gitflow*

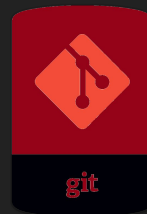
”

# Your level?



**Checkout, clone  
reset, branch  
Repo di Gitlab**

**Remote branch,  
social coding,  
IDE integration,  
Merge request...  
e altro.**



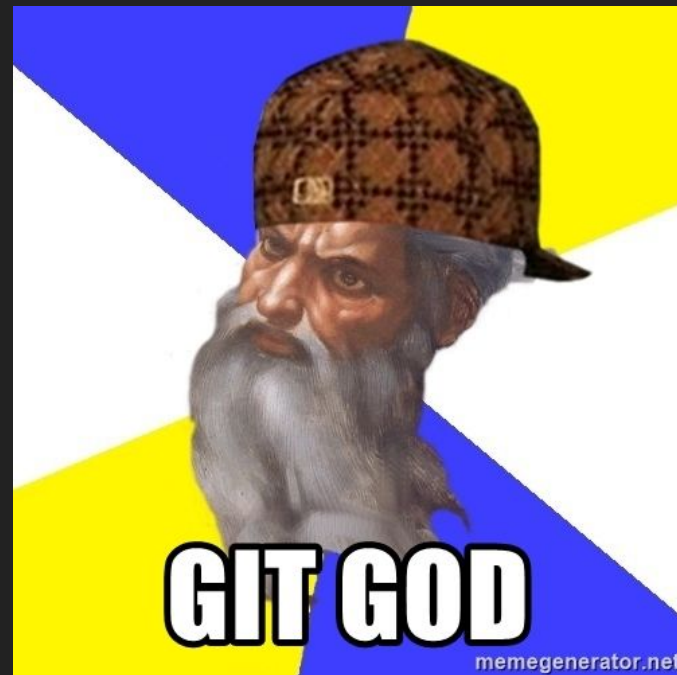
**Basi di Git (CLI)  
init-add-commit**

2

1

3

# And now... you're all alone!



GRAZIE!

## Credits

Special thanks to all the people who made and released these awesome resources for free:

- ▶ Presentation template by [SlidesCarnival](#)
- ▶ Photographs by [Startupstockphotos](#)
- ▶ Anil Gupta ([www.guptaanil.com](http://www.guptaanil.com))
- ▶ Pete Nicholls ([github.com/Aupajo](https://github.com/Aupajo))
- ▶ Armando Fox

***Questo documento è distribuito con licenza CreativeCommon BY-SA 3.0***



# Presentation design

This presentation uses the following typographies and colors:

- ▶ Titles: **Dosis**
- ▶ Body copy: **Roboto**

You can download the fonts on these pages:

<https://www.fontsquirrel.com/fonts/dosis>

<https://material.google.com/resources/roboto-noto-fonts.html>

- ▶ Orange **#ff8700**

You don't need to keep this slide in your presentation. It's only here to serve you as a design guide if you need to create new slides or download the fonts to edit the presentation in PowerPoint®

## SlidesCarnival icons are editable shapes.

This means that you can:

- Resize them without losing quality.
- Change line color, width and style.

Isn't that nice? :)

Examples:

