



Build software better, together



speaker: Maurizio Del Magno



Maurizio Del Magno
Developer



Lev@nte **software**



i-ORM
DJSON

github.com/mauriziodm/iORM

github.com/mauriziodm/DJSON



mauriziodm@levantesw.it

mauriziodelmagno@gmail.com



facebook.com/maurizio.delmagno

iORM + DJSON (group)



Membro fondatore

eInvoice4D

<https://github.com/delphiforce/eInvoice4D>



Build software better, together



speaker: Maurizio Del Magno

Strumenti

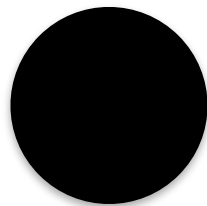
(Download)

- **Git:** <https://git-scm.com/downloads>
(eseguire il setup ed è subito pronto)
- **GitViz:** <https://github.com/Readify/GitViz/releases>
(a questo link trovate la possibilità di scaricare il file “GitViz-1.0.2.0.zip” contenente il programma già compilato, scompattare semplicemente ed è pronto all’uso, non è necessario alcun setup)
- **Sourcetree:** <https://www.sourcetreeapp.com>

Git

...a cosa serve?

- **...a tenere traccia delle modifiche ai sorgenti** (e ai files in generale)
- **...se qualcosa va storto ci permette di tornare senza problemi a versioni precedenti del codice**
- **...a collaborare con altri**

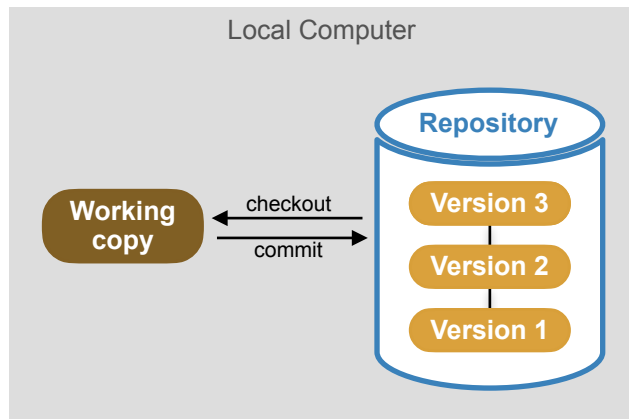


VCS *(Version Control System)*

LVCS

Local Version Control System

- RCS.

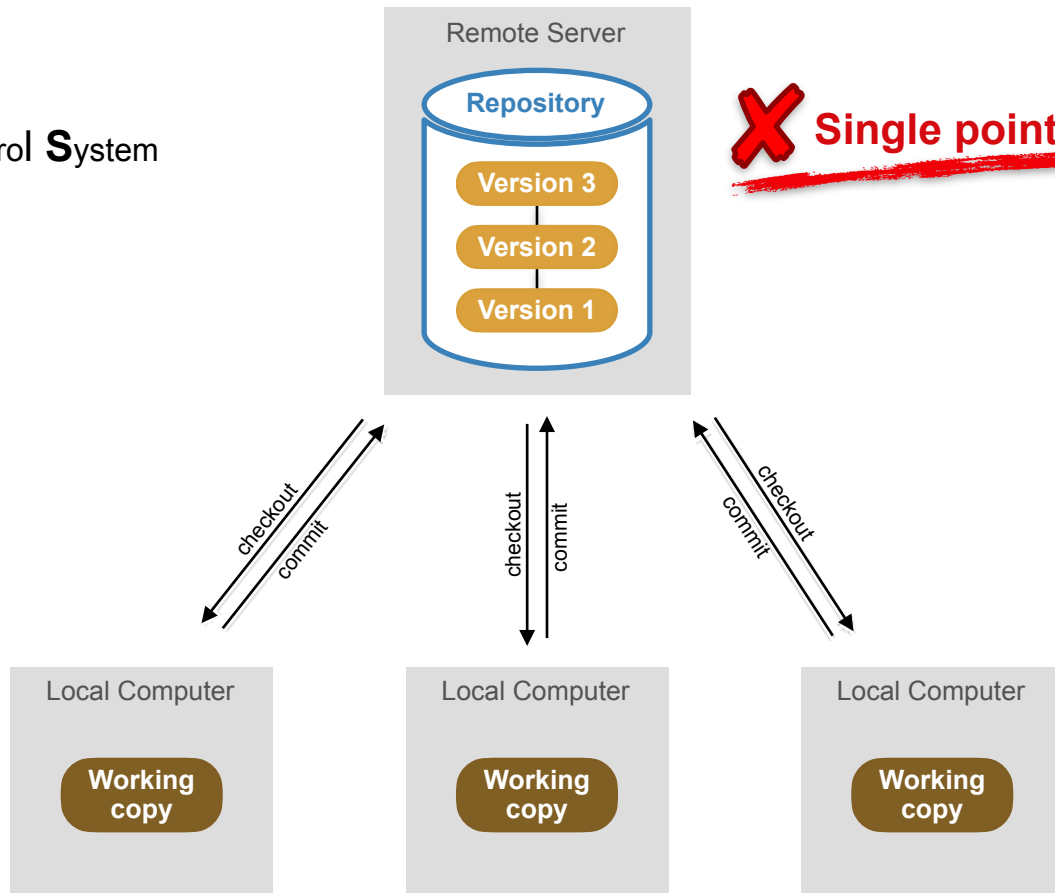


X Team?

CVCS

Centralized Version Control System

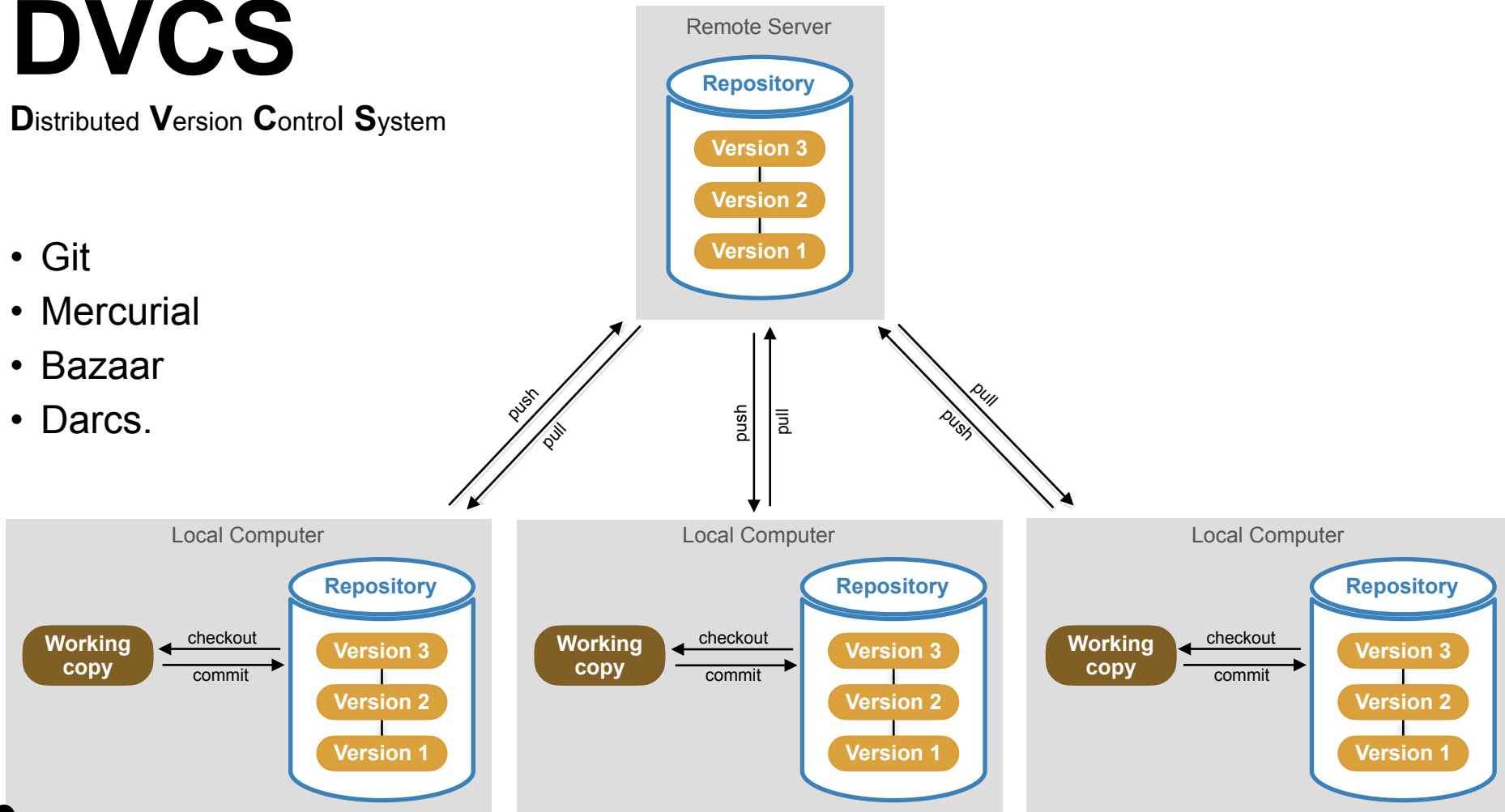
- CVS
- Subversion
- Perforce.



DVCS

Distributed **V**ersion **C**ontrol **S**ystem

- Git
- Mercurial
- Bazaar
- Darcs.

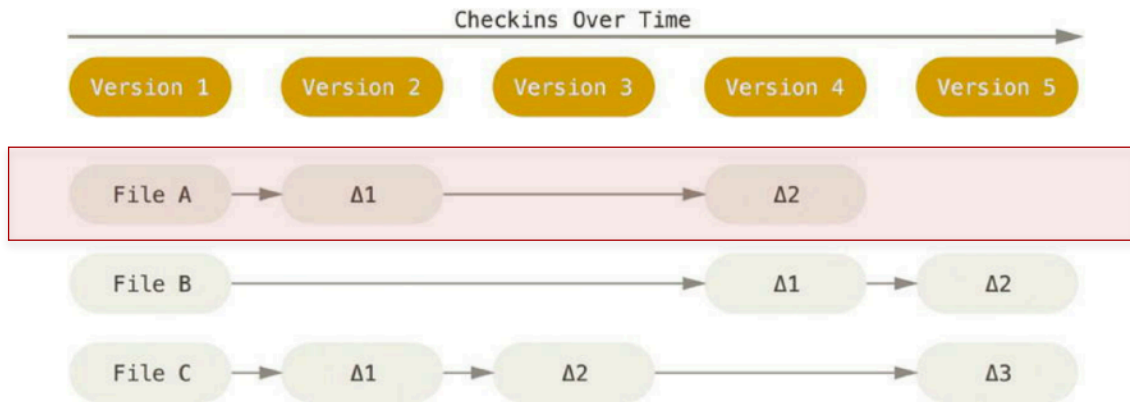


Git Basics:

Dimenticate quello che sapete degli altri VCS!

Snapshots, Not Differences

Gli altri VCS memorizzano liste di differenze per singolo file

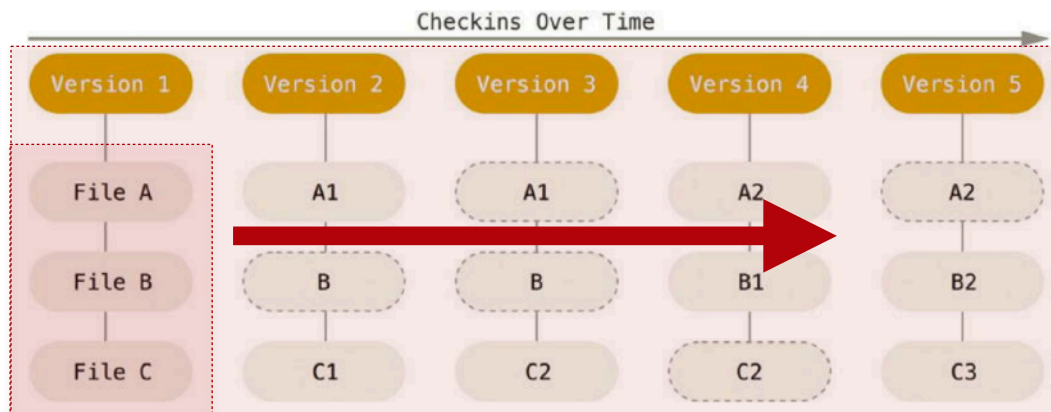


Snapshots, Not Differences

Git memorizza snapshots del filesystem

Commit = foto/snapshot dello stato del filesystem in un dato istante

Repository = Stream di snapshots



Snapshots, Not Differences

Più simile a un mini-filesystem...

... con tools aggiuntivi particolarmente potenti.

(Linus Torvalds)



Nearly Every Operation Is Local

- **Quasi ogni operazione è locale** *(non ha bisogno di accedere a info remote)*
 - Scorrere la storia del progetto
 - Ripristinare una versione precedente (checkout)
 - Commit
 - Branch, Merge
- **Lavorare offline** *(es: develop, commit, branch, merge, stash, patch... upload differito)*
 - Aereo
 - Treno
 - Off-VPN
 - In caso di guasti o comunque senza connessione per qualunque motivo



Git Has Integrity

- **Tutto è check-summed prima di essere archiviato**
 - Strettamente integrato in Git al più basso livello
 - Impossibili variazioni, corruzioni, perdite di informazioni che Git non sia in grado di rilevare
 - Il tipo di check-sum usato è SHA-1 (40 car. es: "24b9da6552252987aa493b52f8696cd6d3b00373")
 - E' usato anche come nome per i commit (No filenames, bastano i primi 7 caratteri es: "24b9da6")
- **Non cancella nulla, aggiunge solo**
- **E' veramente difficile convincere Git a fare qualcosa che causi perdita di informazioni o che non sia annullabile**



Caratteristiche



Distribuito



Sicuro



Veloce



Forte supporto allo sviluppo non lineare

(migliaia di branches paralleli)

Repository

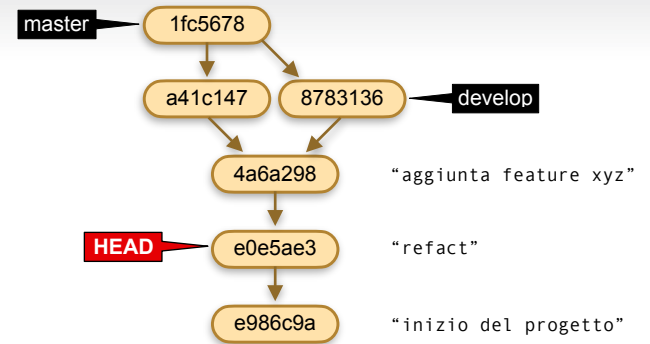
2 tipi di oggetti

● Commit

- Rappresenta lo stato di un progetto in un dato istante (*foto/snapshot*)
- Nome (*SHA-1*)
- Descrizione
- Riferimento a genitore
- Un oggetto commit può avere più di un genitore.

● Intestazioni (Heads)

- Puntatore a un oggetto commit
- Nome / Titolo (*alla creazione del repository viene creata una intestazione 'master' per default*)
- Possono esserci molteplici intestazioni
- **HEAD**: speciale intestazione che punta all'oggetto commit su cui stiamo lavorando.
(*working directory*) (*checkout*) (*attenzione al maiuscolo*)



Repository

2 “azioni” principali + 1



Branch *(o ramo, braccio)*

- Una linea indipendente di sviluppo
- Ha un nome
- C'è sempre un ramo principale “master”
- Sperimentare/testare nuove feature o bug-fix senza introdurre anomalie nel ramo principale
- Lavorare in team




Merge

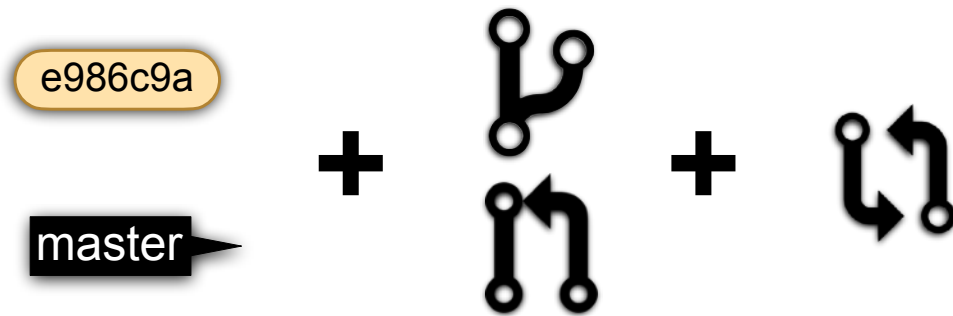
- Fusione tra due rami
- 3 tipi di merge: fast forward, clean, conflict



Push/Pull

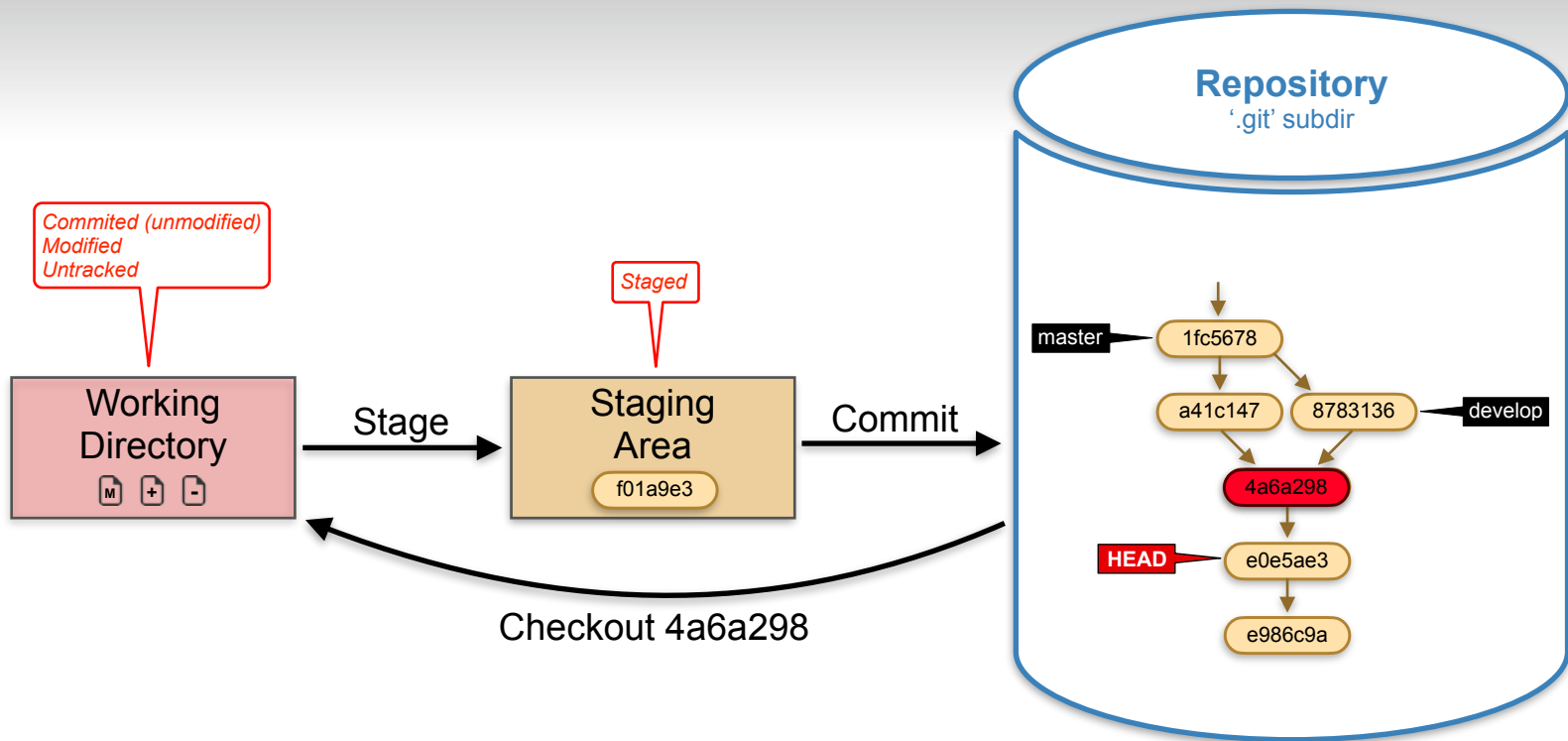
- **Push:** scrivere sul repository remoto (origin) le modifiche fatte nel nostro repository locale (*upload*)
- **Pull:** leggere dal repository remoto (origin) le modifiche fatte da altri (*download*)

2 + 2 + 1 =  git



Repository: 3 Sezioni

Local



1) Inizializzazione e primo commit

The diagram illustrates the initial setup and first commit in a Git repository. It is divided into three main sections:

- Shell:** Contains the commands to initialize the repository, add the file, and commit it.

```
git init
git add view.pas
git commit -m "primo commit"
```
- Files:** Shows the files in the repository. The `view.pas` file is shown with its content: `MainForm.Caption := 'Git test';`. The `model.pas` file is also shown but is empty.
- Git Status:** Shows the state of the repository. The `Staging area` contains `view.pas`. The `.git` directory is shown with a commit hash `e986c9a` and the message `"primo commit"`.


Visual elements include a yellow folder icon for the `Staging area`, a red Git logo for the `.git` directory, and a commit hash `e986c9a` with the message `"primo commit"`.

2) Modifichiamo il file

shell

```
git init
git add view.pas
git commit -m "primo commit"
git add view.pas
git commit -m "Risolto bug MainForm non visibile"
```

Staging area
view.pas



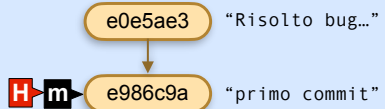
view.pas

model.pas

```
MainForm.Caption := 'Git test';
MainForm.Show;
```

e0e5ae3 "Risolto bug..."

e986c9a "primo commit"




3) Un altro commit?

shell

```
git add view.pas  
git add model.pas  
git commit -m "Aggiunto model e commento"
```

Staging area
view.pas; model.pas



view.pas

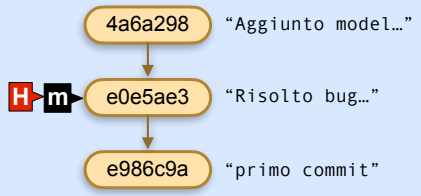
```
MainForm.Caption := 'Git test';  
MainForm.Show; // Necessario;
```

model.pas

4a6a298 "Aggiunto model..."

H m e0e5ae3 "Risolto bug..."

e986c9a "primo commit"



4) Come era prima?

shell

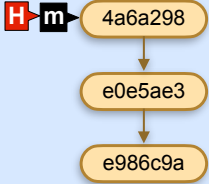
git checkout e0e5ae3

view.pas


MainForm.Caption := 'Git test';
MainForm.Show; // Necessario;

model.pas

TCliente = class...



A diagram showing a vertical sequence of three commit hashes in yellow rounded rectangles: 4a6a298, e0e5ae3, and e986c9a. Arrows point downwards from 4a6a298 to e0e5ae3, and from e0e5ae3 to e986c9a. To the left of the top hash is a red square icon with a white 'H' and a black square icon with a white 'm'.




The Git logo, consisting of a yellow folder icon and a red square icon with a white 'A'.

5) Ci chiedono una nuova feature (ma senza toccare il master)


shell

```
git checkout e0e5ae3
git branch feature1
git checkout feature1
git add .
git commit "Nuova classe cliente..."
```

Staging area
view.pas; model.pas




view.pas





```
MainForm.Caption := 'Git test';
MainForm.Show;
LCliente := TNewCliente.Create;
```



model.pas



```
TNewCliente = class...
```

 4a6a298

 8783136

 e0e5ae3 

e986c9a


```
graph TD
    4a6a298 --> e0e5ae3
    8783136 --> e0e5ae3
    e0e5ae3 --> e986c9a
```

6) Continuiamo a lavorare sulla nuova feature

shell

```
git checkout e0e5ae3
git branch feature1
git checkout feature1
git add .
git commit "Nuova classe cliente..."
git commit -a -m "aggiunto proprietà Age"
```

Staging area
model.pas

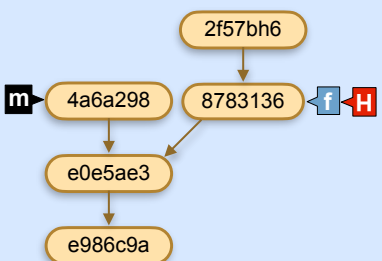


view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show;
LCliente := TNewCliente.Create;
```

model.pas

```
TNewCliente = class...
property Age: integer read FAge;
```




The diagram illustrates the Git commit history. It shows a sequence of commits represented by yellow rounded rectangles with their hash values: 2f57bh6, 4a6a298, 8783136, e0e5ae3, and e986c9a. Arrows indicate the parent-child relationships: 2f57bh6 points to 8783136, 4a6a298 points to 8783136, 8783136 points to e0e5ae3, and e0e5ae3 points to e986c9a. Social media icons (m, f, H) are placed next to the 4a6a298, 8783136, and e0e5ae3 commits respectively.

7) Ora però dobbiamo tornare alla versione di produzione

shell

```
git checkout e0e5ae3
git branch feature1
git checkout feature1
git add .
git commit "Nuova classe cliente..."
git commit -a -m "aggiunto proprietà Age"
git checkout master
git commit -a -m "visualizzazione età cliente"
```

Staging area
view.pas

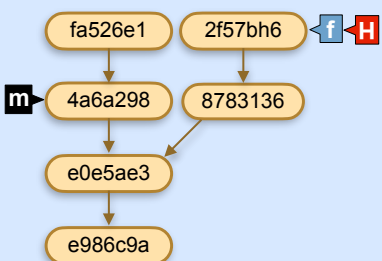


view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show; // Necessario;
LbEtà.Caption := 'Nuova classe cliente';
```

model.pas

```
TNewClientClass = class(TClientClass)
property Age: integer read FAge;
```



The diagram illustrates the Git commit history. It shows a sequence of commits represented by yellow rounded rectangles with commit hashes. The main branch (master) is shown as a vertical line of commits: fa526e1, 4a6a298, e0e5ae3, and e986c9a. A feature branch (feature1) is shown as a branch that diverges from 4a6a298 and contains commits 2f57bh6 and 8783136. The feature1 branch is currently checked out, indicated by a blue 'f' icon. The master branch is indicated by a black 'm' icon. The commit e0e5ae3 is the common ancestor between the two branches.

8) Merge (no conflict)

shell

```
git merge feature1
```

view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show; // Necessario  
Label1.Caption := LCliente.Age;
```

model.pas

```
TCliente = class...  
property Age: integer read FAge;
```

```
graph TD; 156he1b --> fa526e1; 156he1b --> 2f57bh6; fa526e1 --> 4a6a298; 4a6a298 --> e0e5ae3; e0e5ae3 --> e986c9a; 2f57bh6 --> 8783136; 8783136 --> e0e5ae3; fa526e1 --> |H-m|; 2f57bh6 --> |f|
```

The diagram illustrates a Git commit history. It starts with a root commit 156he1b (yellow oval) which branches into two paths. The left path consists of commits fa526e1 (red oval, marked with a red 'H' and black 'm' icon), 4a6a298 (yellow oval), e0e5ae3 (yellow oval), and e986c9a (yellow oval). The right path consists of commits 2f57bh6 (red oval, marked with a blue 'f' icon) and 8783136 (yellow oval). Both paths converge at commit e0e5ae3, which is the result of a merge. A small Git logo is visible in the top right corner of the diagram area.

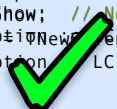
9) Merge (with conflict)

shell

```
git merge feature1
Auto-merging model.pas
CONFLICT (content): Merge conflict in model.pas
Automatic merge failed; fix conflicts and then
commit the result.
git add model.pas
git commit "feature1 merged on master"
```

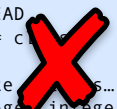
view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show; // Necessario
Label1.Caption := NewClient.FAge;
Label1.Caption := LCliente.Age;
```




model.pas


```
ICliente = class...
<<<<<< HEAD
TCliente = class...
=====
TNewCliente = class...
property Age: integer read FAge;
>>>>>> feature1
...
```



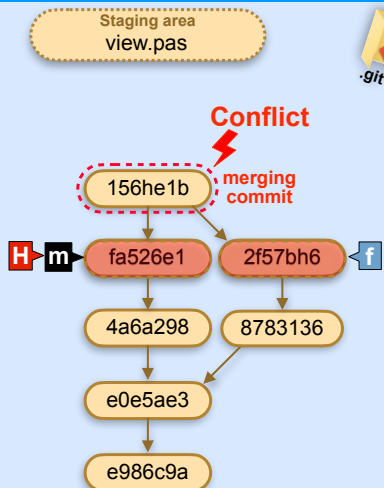
Staging area
view.pas



Conflict



merging
commit



```
graph TD
    fa526e1[fa526e1] --> 4a6a298[4a6a298]
    fa526e1 --> 2f57bh6[2f57bh6]
    2f57bh6 --> 8783136[8783136]
    4a6a298 --> e0e5ae3[e0e5ae3]
    e0e5ae3 --> e986c9a[e986c9a]
    156he1b[156he1b] -.-> fa526e1
    156he1b -.-> 2f57bh6
```

10) Eliminiamo il branch (oppure no?)

shell

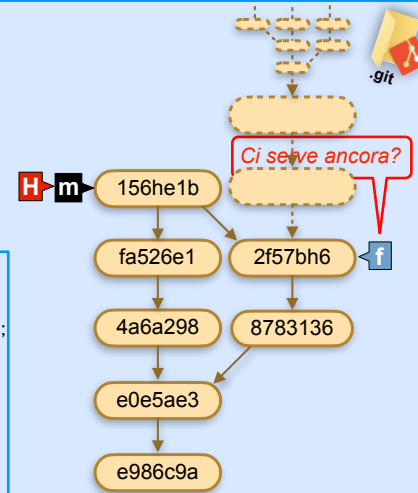
```
git merge feature1
Auto-merging model.pas
CONFLICT (content): Merge conflict in model.pas
Automatic merge failed; fix conflicts and then
commit the result.
git add model.pas
git commit "feature1 merged on master"
git branch -D feature1
```

view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show; // Necessario
LCliente := TNewCliente.Create;
Label1.Caption := LCliente.Age;
```

model.pas

```
TNewCliente = class...
property Age: integer read FAge;
```



11) Se branch il viene eliminato prima del merge

reflog command
git branch NewBranchName SHA1
git checkout -b NewBranchName SHA1

shell

```
git branch -D feature1
git reflog

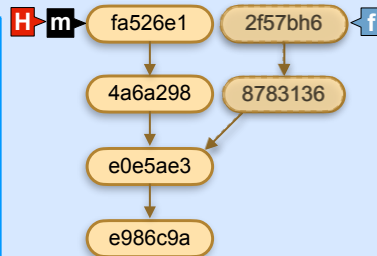
fa526e1 HEAD@{0}: checkout: moving from branch feature1 to master
2f57bh6 HEAD@{1}: commit: visualizzazione età cli...
8783136 HEAD@{2}: commit: aggiunto proprietà Age
E0e5ae3 HEAD@{4}: checkout: moving from branch master to feature1
...
git branch oldfeature1 HEAD@{1}
```

view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show; // Necessario
LCliente := TNewCliente.Create;
Label1.Caption := LCliente.Age;
```

model.pas

```
TNewCliente = class...
property Age: integer read FAge;
```



12) Stash

shell

```
git checkout feature1
git stash save
git checkout feature1
...
git checkout master
git stash apply
git stash drop
```

Pending

view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show; // Necessario
LCliente := TNewCliente.Create;
Label1.Caption := LCliente.Age;
MainForm.Close;
```

Pending

model.pas

```
TNewCliente = class...
property Age: integer read FAge;
procedure Reset;
```

Stash

The diagram illustrates the Git commit history. It shows a sequence of commits: fa526e1, 4a6a298, e0e5ae3, and e986c9a. A branch is shown with a 'Pending' state, indicated by a red lightning bolt and the word 'Pending'. The 'Stash' is represented by a folder icon and the word 'Stash'. The commit 2f57bh6 is shown as a 'Stash' entry, with a red lightning bolt and the word 'Pending' next to it. The commit 2f57bh6 is also marked with a red 'H' and a black 'm' icon, indicating it is a merge commit. The commit 2f57bh6 is also marked with a blue 'f' icon, indicating it is a file commit. The commit 2f57bh6 is also marked with a red 'H' and a black 'm' icon, indicating it is a merge commit. The commit 2f57bh6 is also marked with a blue 'f' icon, indicating it is a file commit.

13) Rebase (no conflict)

Fast forward merge alla fine

shell

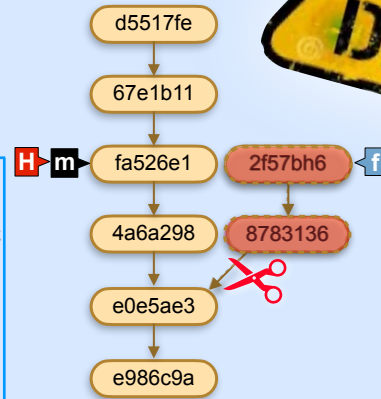
```
git checkout feature1
git rebase master
First, rewinding head to replay your work on top of it...
Applying: commit
Applying: commit
git checkout master
git merge feature1
git branch -D feature1
```

view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show; // Necessario
Label1.Caption := 'New Client';
Label1.Caption := LCliente.Age;
```

model.pas

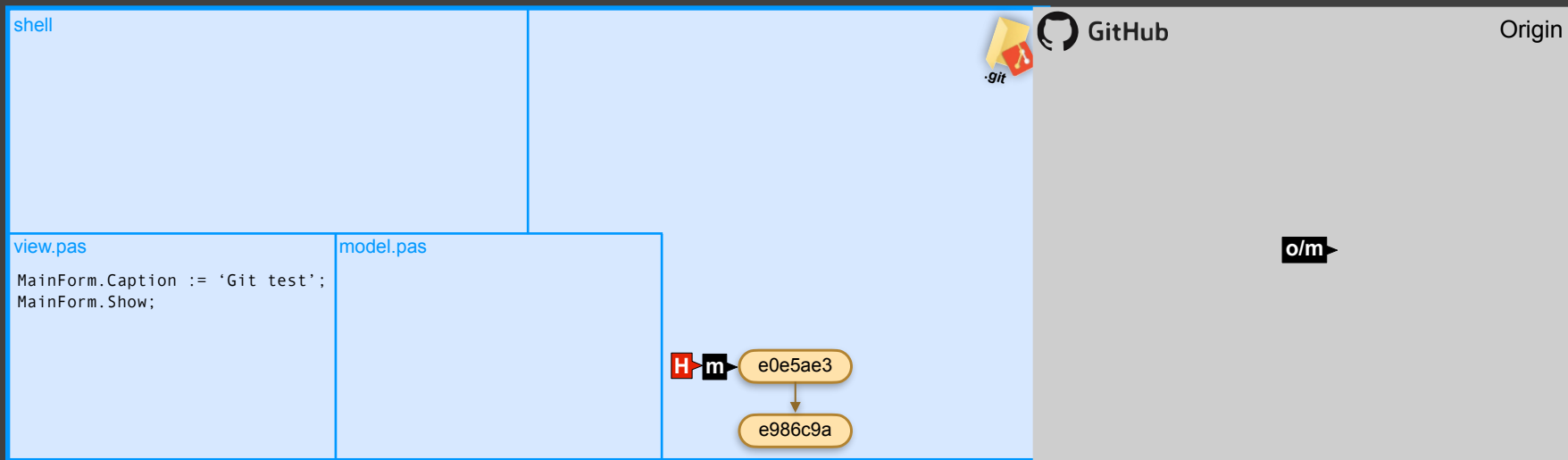
```
TNewClient = class...
property Age: integer read FAge;
```



Non fare rebase su rami già inviati a un repository remoto

14) Cominciamo a collaborare

Creazione repository remoto su GitHub



15) creare un repository remoto (Origin)

git remote -v

collegiamo il repository remoto (origin) al nostro repository locale

collegiamo il nostro branch master locale con il master remoto

cacciamo il nostro primo Push

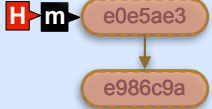
shell

```
git remote add origin https://github.com/mauriziodm/delphiedintorni.git  
git push -set-upstream origin master
```

view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show;
```

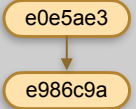
model.pas



The diagram shows the local repository state. It features a commit icon labeled 'H-m' pointing to a commit hash 'e0e5ae3'. Below this hash is another hash 'e986c9a', connected by a downward arrow, indicating a parent-child relationship.

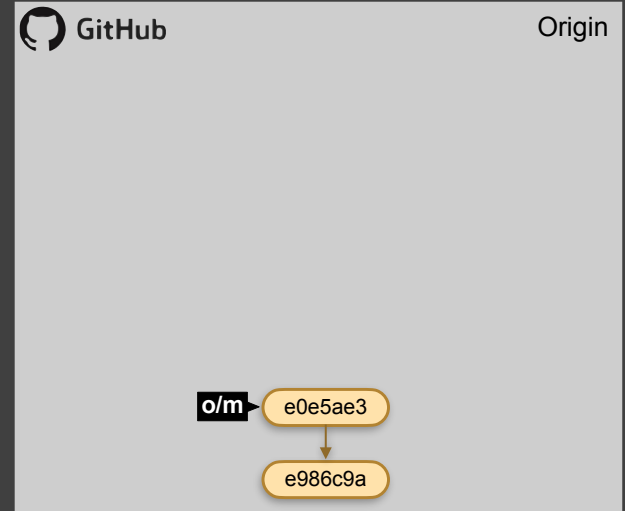
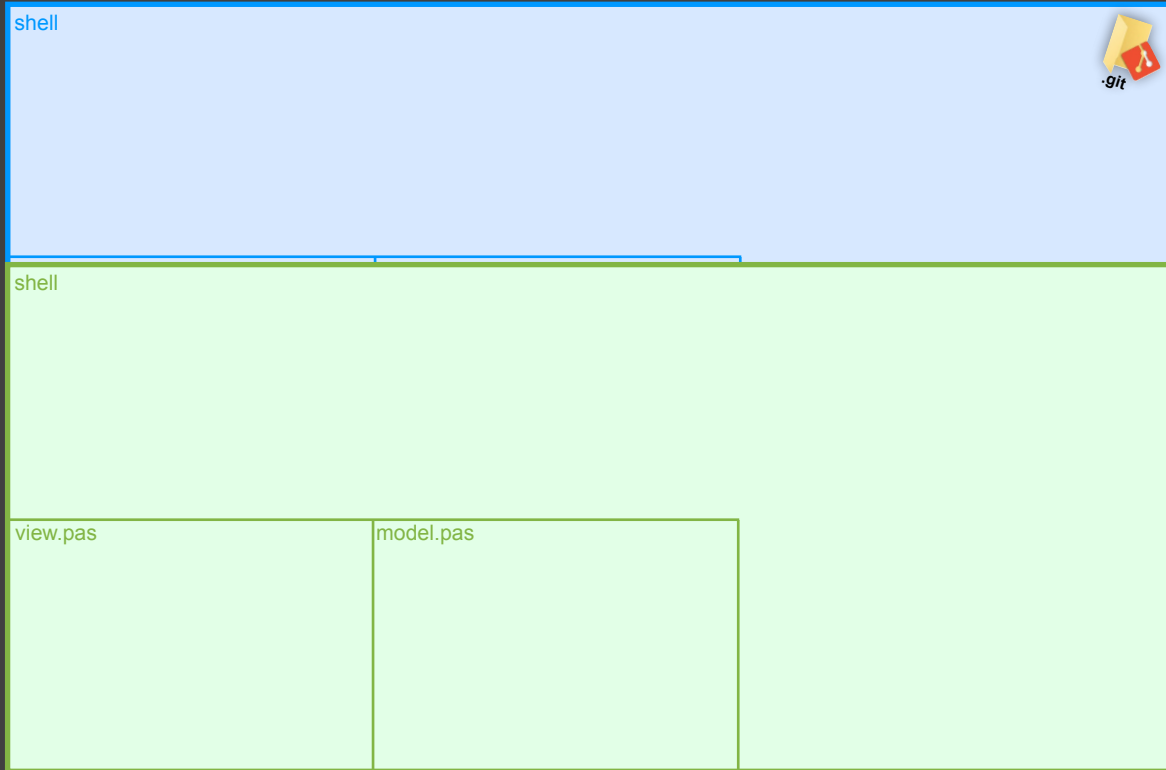
GitHub

Origin



The diagram shows the remote repository state on GitHub. It features a commit icon labeled 'o/m' pointing to a commit hash 'e0e5ae3'. Below this hash is another hash 'e986c9a', connected by a downward arrow, indicating a parent-child relationship.

16) Un nuovo contributor

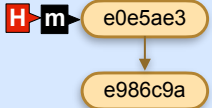


shell

view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show
```

model.pas



A diagram showing a commit labeled 'H-m' with hash 'e0e5ae3' and a child commit with hash 'e986c9a'.

17) Un nuovo contributor git clone

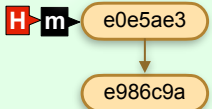
shell

```
git clone https://mauriziodm:password@github.com/mauriziodm/delphiedintorni.git
```

view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show;
```

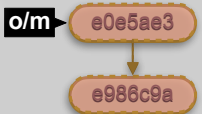
model.pas



A diagram showing a commit labeled 'H-m' with hash 'e0e5ae3' and a child commit with hash 'e986c9a'.

GitHub

Origin



A diagram showing a commit labeled 'o/m' with hash 'e0e5ae3' and a child commit with hash 'e986c9a'.

shell

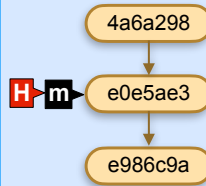
```
git fetch
git status
On branch master
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)
```

```
nothing to commit, working tree clean
git pull
```

view.pas

```
MainForm.Caption := 'Git test'; TNewCliente = class...
MainForm.Show
```

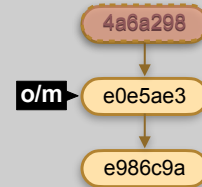
model.pas



18) Un nuovo contributor push/pull (same branch, no conflict)



Origin



shell

```
git clone https://mauriziodm:password@github.com/mauriziodm/itdevcon.git
git commit -a -m "aggiunta classe TNewCliente"
git push
```

Staging area
model.pas

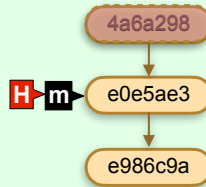


view.pas

```
MainForm.Caption := 'Git test'; TNewCliente = class...
MainForm.Show;
```



model.pas



19) Un nuovo contributor push/pull (same branch, divergent)

shell

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

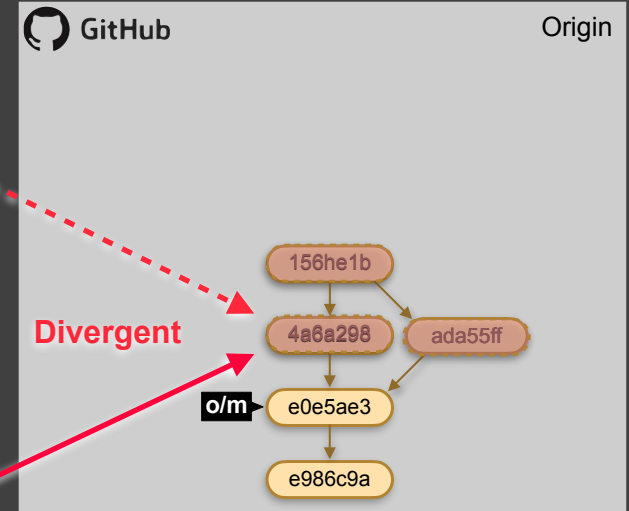
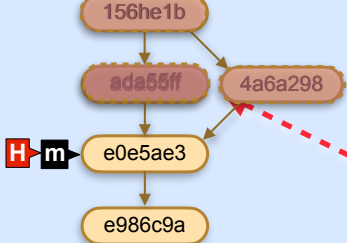
view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show;
```

model.pas

```
TOtherCliente = class...
```

Staging area
model.pas



shell

```
git clone https://mauriziodm:password@github.com/mauriziodm/delphiedintorni.git
git commit -a -m "aggiunta classe TNewCliente"
git push
git pull
```

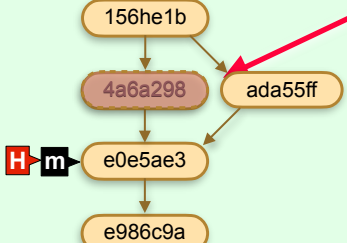
view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show;
```

model.pas

```
TNewCliente = class...
```

Staging area
model.pas



shell

```
git commit -a -m "aggiunta classe TOtherCliente"  
git push  
git pull  
git add model.pas  
git commit -m "fatto merge TNewCliente"  
git push
```

Staging area
model.pas

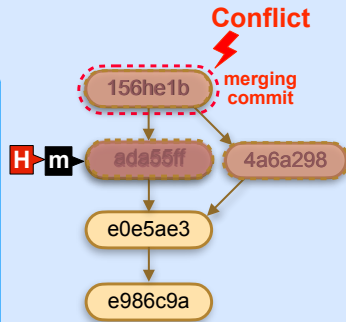


view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show
```

model.pas

```
TOtherCliente = class...  
<<<<<< HEAD  
TOtherCliente = class...  
=====  
TNewCliente = class...  
>>>>>> origin/master  
...
```



shell

```
git clone https://mauriziodm:password@github.com/mauriziodm/delphiedintorni.git  
git commit -a -m "aggiunta classe TNewCliente"  
git push  
git pull
```

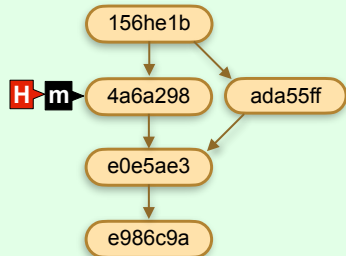


view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show;
```

model.pas

```
TNewCliente = class...
```

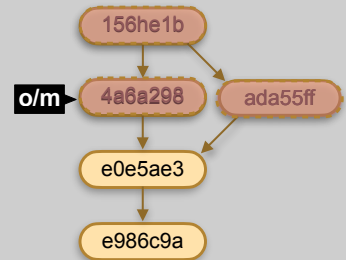


20) Un nuovo contributor

push/pull (same branch, divergent+conflict)



Origin



shell

```
git commit -a -m "aggiunto classe TOtherCliente"  
git push  
git fetch  
git pull
```

Staging area
model.pas

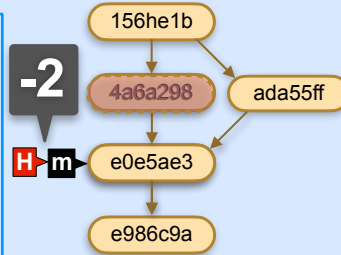


view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show;
```

model.pas

```
TOtherCliente = class...
```



shell

```
git branch newfeature  
git checkout newfeature  
git commit -a -m "nuova classe TNewCliente"  
git push --set-upstream origin feature2  
git checkout master  
git fetch  
git pull  
git merge newfeature  
git push
```

Staging area
model.pas

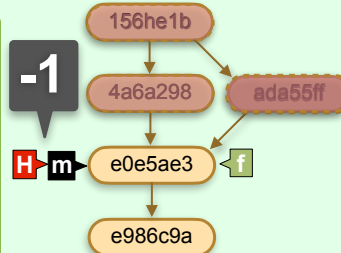


view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show;
```

model.pas

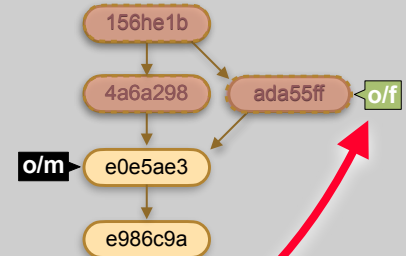
```
TNewCliente = class...
```

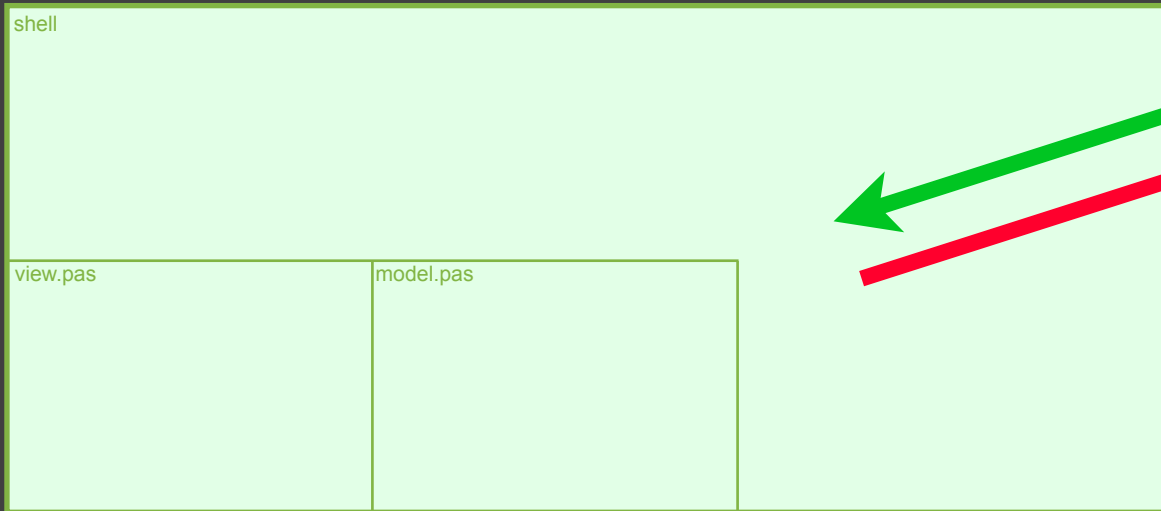
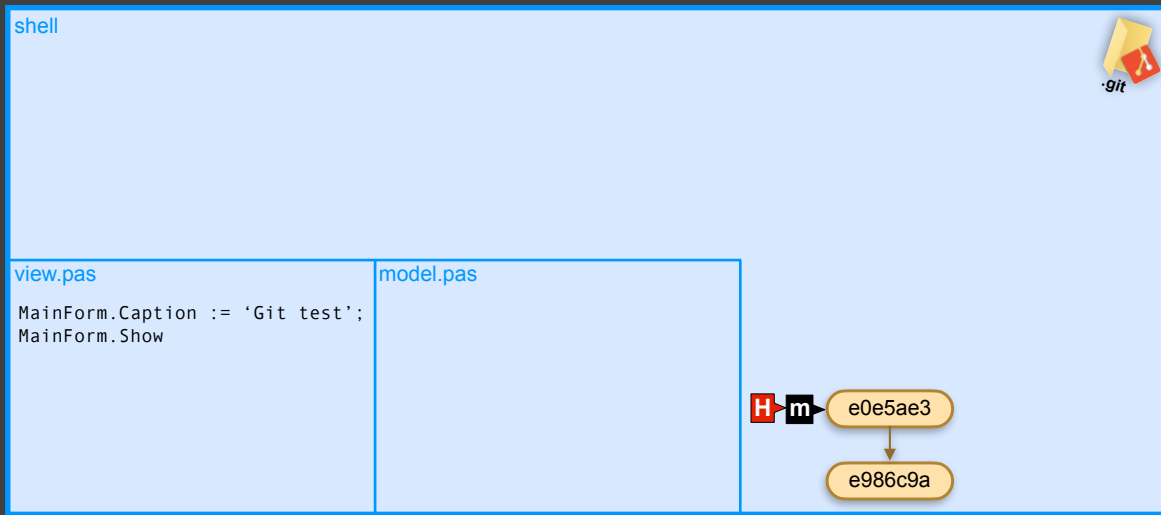


21) Un nuovo contributor branch separati

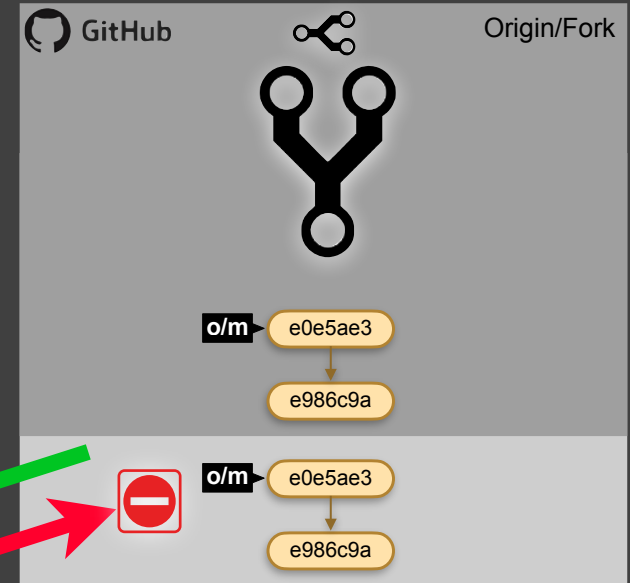


Origin





22) Se il contributor non ha i diritti?
fork

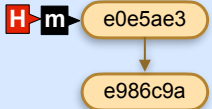


shell

view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show
```

model.pas



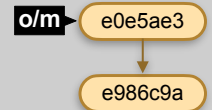
Git icon

23) Se il contributor non ha i diritti?

Fork: clone locale

GitHub

Origin



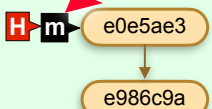
shell

```
git clone https://rinobosco70:BoscoRino70!!@github.com/rinobosco70/delphiedintorni.git
```

view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show;
```


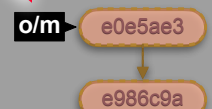
model.pas



Git icon

GitHub

Origin/Fork



shell

```
git commit -a -m "aggiunto classe TOtherCliente"
git push
```

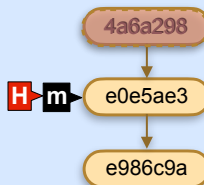
Staging area
model.pas



view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show;
```

model.pas



shell

```
git branch newfeature
git checkout newfeature
git commit -a -m "nuova classe TNewCliente"
git push --set-upstream origin newfeature
```

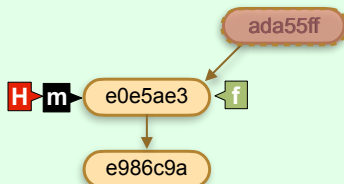
Staging area
model.pas



view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show;
```

model.pas



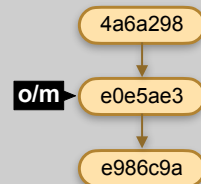
24) Se il contributor non ha i diritti?

New Branch (ora possiamo)
Push



GitHub

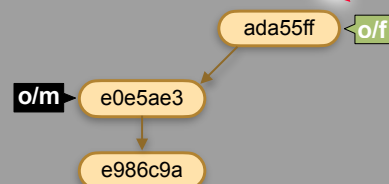
Origin



GitHub



Origin/Fork



shell

```
git commit -a -m "aggiunto classe TOtherCliente"
git push
```

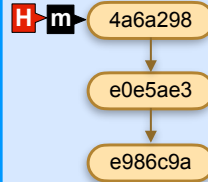


view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show
```

model.pas

```
TOtherCliente = class...
```



shell

```
git branch newfeature
git checkout newfeature
git commit -a -m "nuova classe TNewCliente"
git push
git commit -a -m "aggiunto proprietà age"
git push
```

Staging area
model.pas

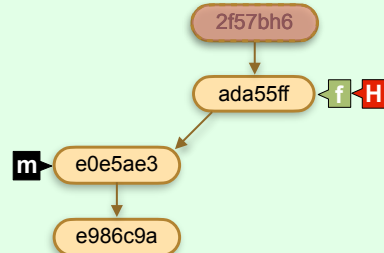


view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show;
```

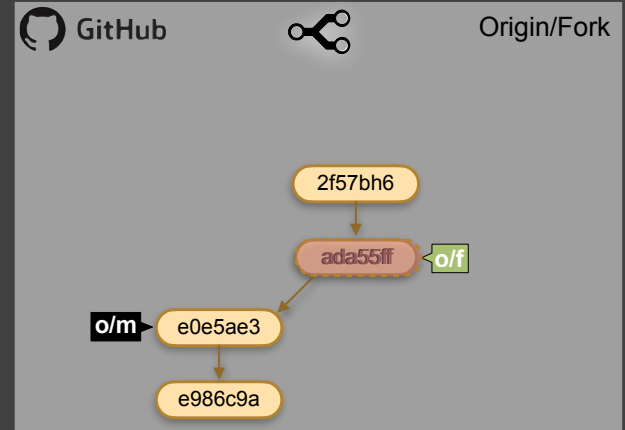
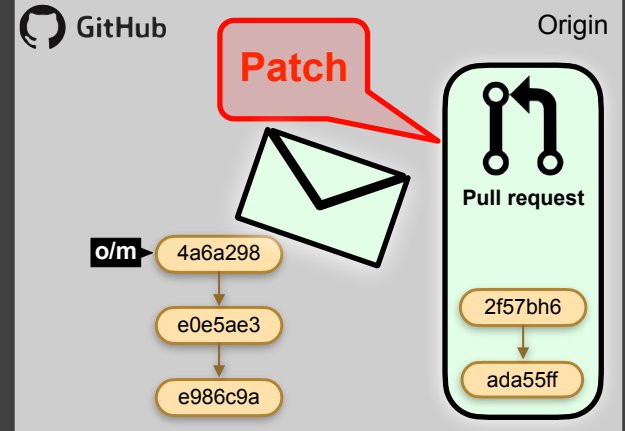
model.pas

```
TNewCliente = class...
property Age: integer read FAge;
```



25) Se il contributor non ha i diritti?

Pull request (su GitHub)



shell

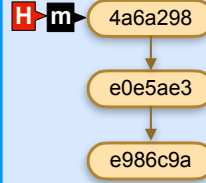


view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show
```

model.pas

```
TOtherCliente = class...
```



shell

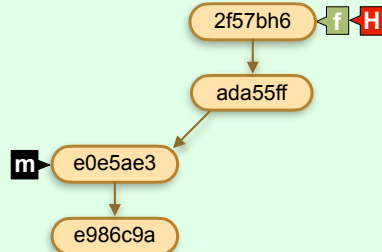


view.pas

```
MainForm.Caption := 'Git test';  
MainForm.Show;
```

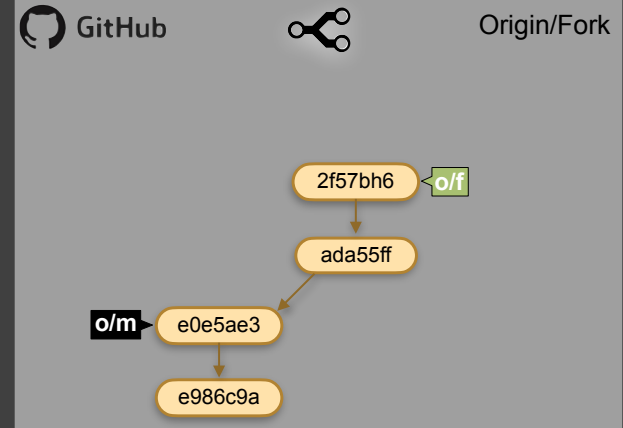
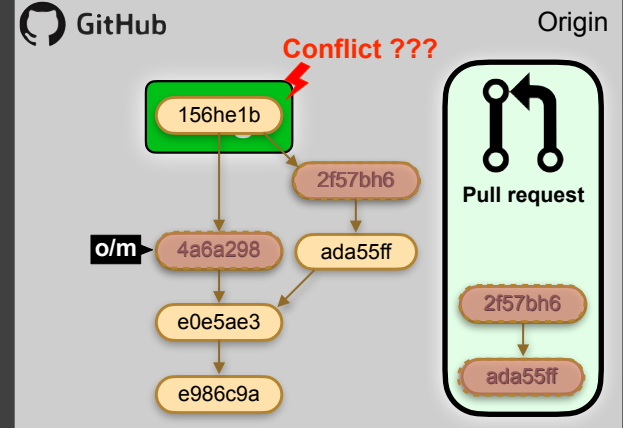
model.pas

```
TNewCliente = class...  
property Age: integer read FAge;
```



26) Se il contributor non ha i diritti?

Pull request: merge a cura del mantainer



shell

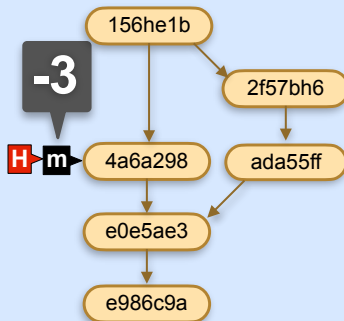
```
git fetch
git pull
```

view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show
```

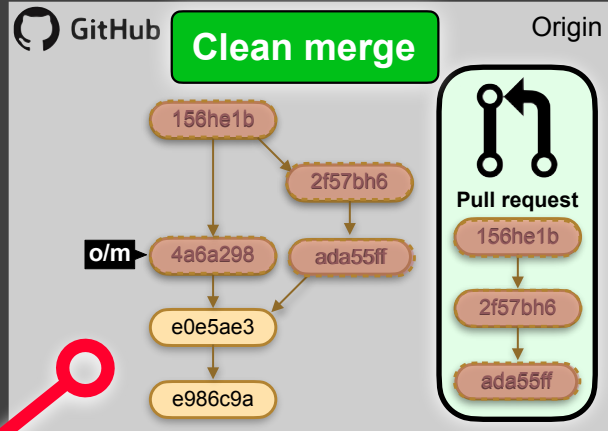
model.pas

```
TOtherCliente = class...
```



27) Se il contributor non ha i diritti?

Pull request: merge a cura del contributor



shell

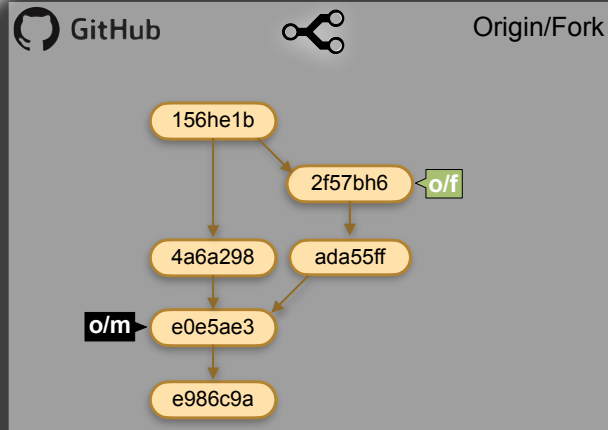
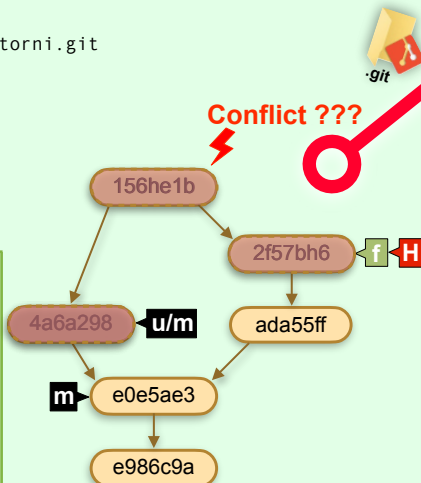
```
git remote add upstream https://github.com/mauriziodm/delphiedintorni.git
git fetch upstream
git merge upstream/master
git push
```

view.pas

```
MainForm.Caption := 'Git test';
MainForm.Show;
```

model.pas

```
TNewCliente = class...
property Age: integer read FAge;
```





Domande?





Maurizio Del Magno
Developer



Lev@nte **software**



i-ORM

github.com/mauriziodm/iORM

DJSON

github.com/mauriziodm/DJSON



mauriziodm@levantesw.it

mauriziodelmagno@gmail.com



facebook.com/maurizio.delmagno

iORM + DJSON (group)



Membro fondatore

eInvoice4D

<https://github.com/delphiforce/eInvoice4D>



delphiedintorni.it
delphi italian community

Grazie!!!



speaker: Maurizio Del Magno