



Build software better, together



Maurizio Del Magno
Developer



levante 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>



2019 - XVIII Edizione



Build software better, together



speaker: Maurizio Del Magno



Git

...perchè questo nome, cosa significa?

‘Idiota’

Git Hub?



“Sono maledettamente egoista ed egocentrico, chiamo tutti i miei progetti in modo che rimandino direttamente e me: prima ‘Linux’, ora ‘Git’ ” (cit. Linus Torvalds)

“ ‘Git’ può significare qualsiasi cosa, a seconda del tuo stato d’animo, fai la tua scelta dal dizionario dei gerghi inglese ” (cit. Linus Torvalds)

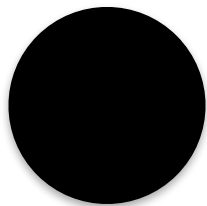
- Idiota / stupido
- Persona polemica, irascibile, che vuole avere sempre ragione
- Testa di porco
- Semplice
- ...

- “Global Information Tracker” (se sei di buon umore e funziona davvero per te, gli angeli cantano e la luce improvvisamente riempie la stanza - cit. Linus Torvalds)
- “Maledetto autocarro carico di m...a” (quando si rompe - cit. Linus Torvalds)

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**

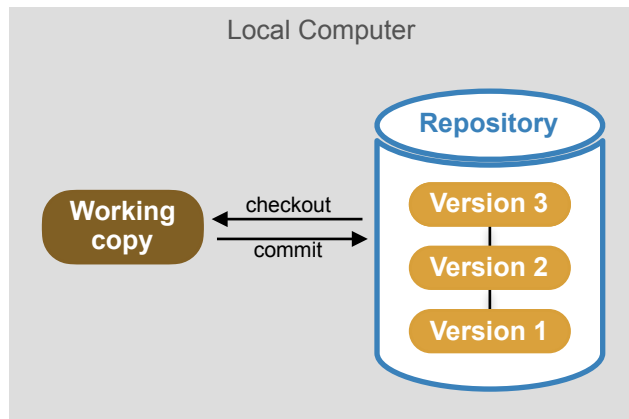


VCS *(Version Control System)*

LVCS

Local Version Control System

- RCS.

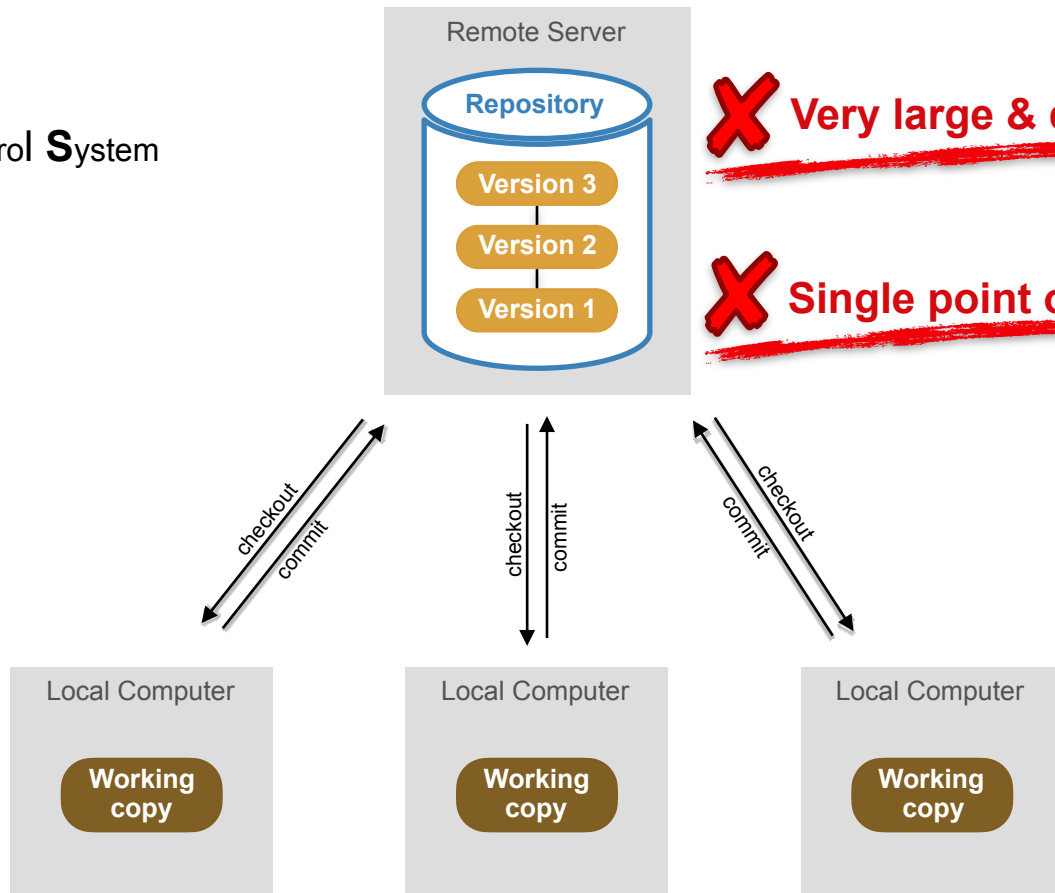


X Team?

CVCS

Centralized Version Control System

- CVS
- Subversion
- Perforce.



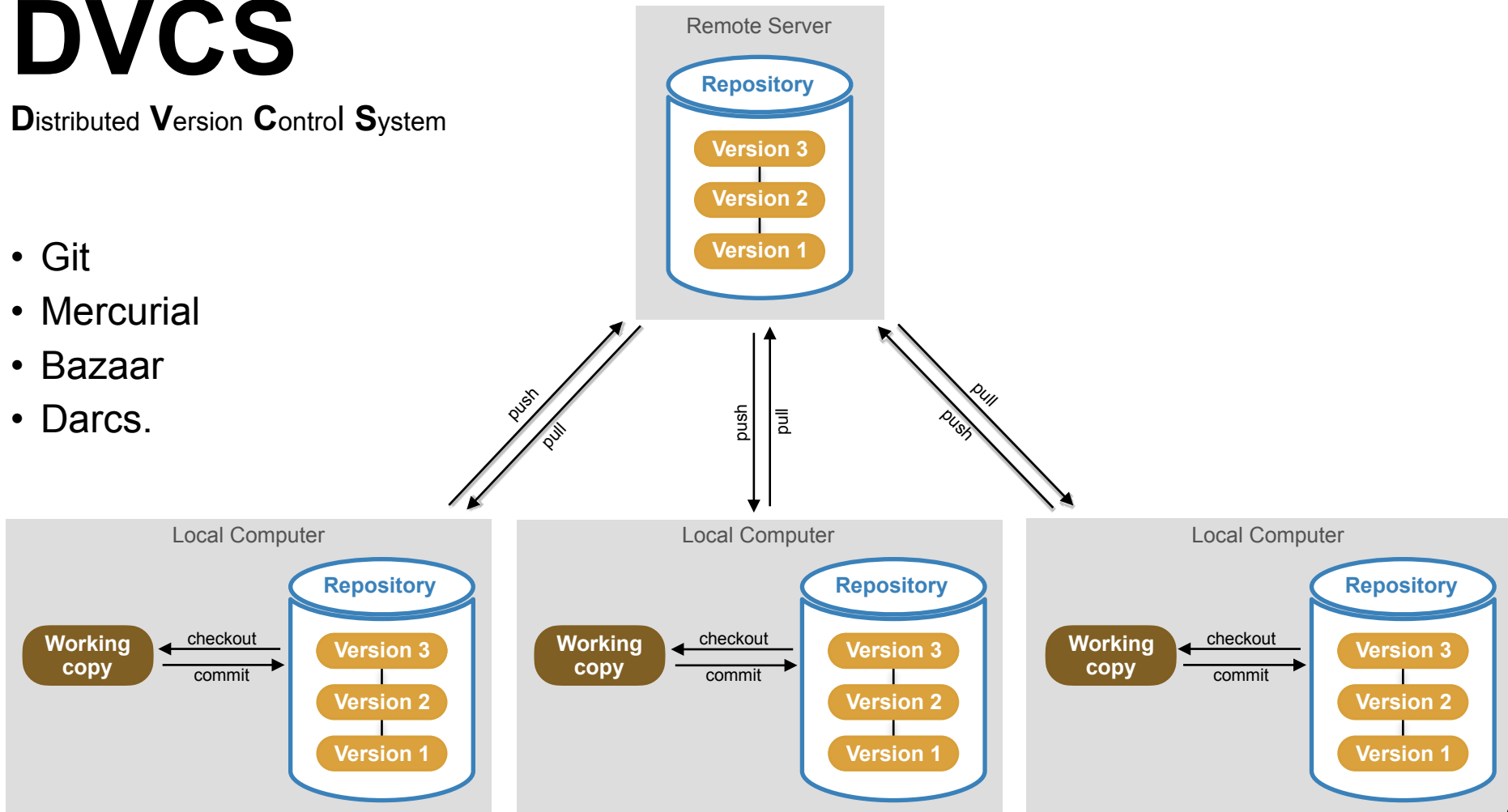
✗ Very large & distributed teams?

✗ Single point of failure

DVCS

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

- Git
- Mercurial
- Bazaar
- Darcs.



Git

...i perchè di Linus Torvalds

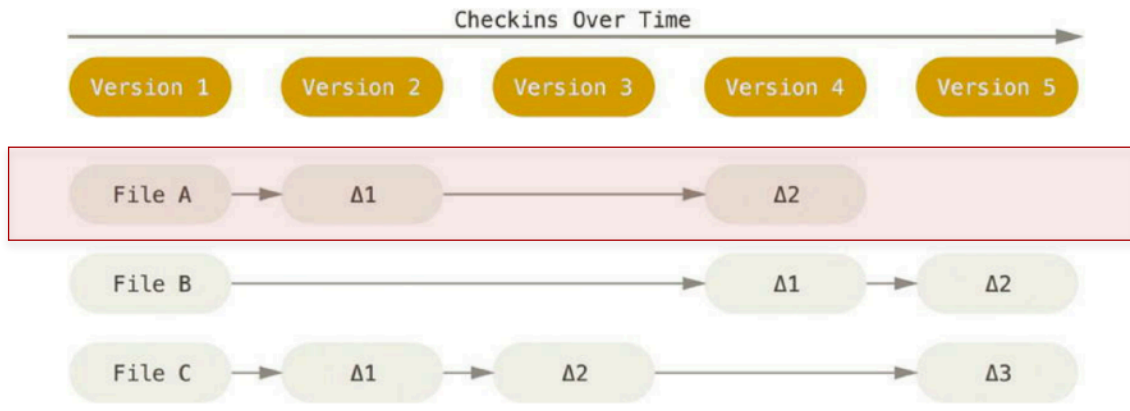
- **Serviva un sostituto di BitKeeper** *(che non era più gratuito)*
- **Nessun altro sistema gratuito soddisfaceva le necessità di Torvalds**
 1. Sistema distribuito *(come BitKeeper)*
 2. Salvaguardia dalla corruzione dei dati *(accidentale o intenzionale)*
 3. Altissime prestazioni
 4. Forte supporto allo sviluppo non lineare *(migliaia di branch paralleli)*
 5. CVS come esempio di cosa “non fare” *(nel dubbio fai il contrario)*

Git Basics:

Dimenticate quello che sapete degli altri VCS!

Snapshots, Not Differences

Gli altri VCS memorizzano liste di differenze per singolo file

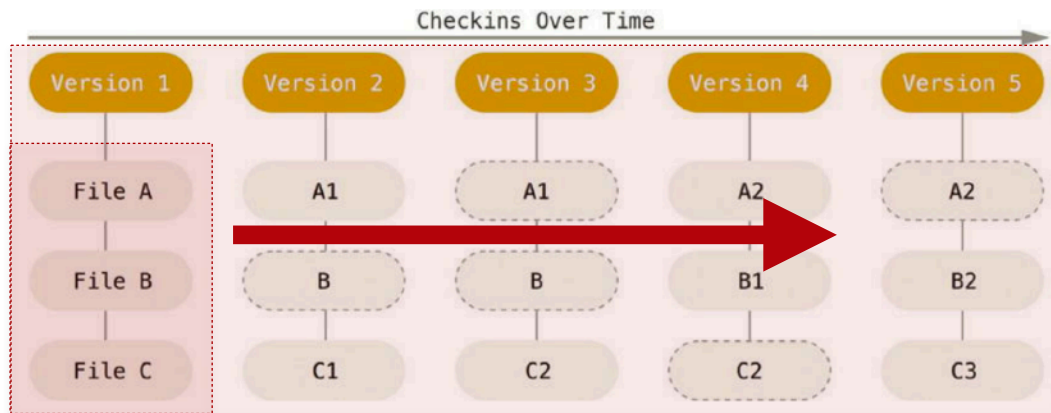


Snapshots, Not Differences

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**



Git

...caratteristiche



Distribuito



Sicuro



Veloce



Forte supporto allo sviluppo non lineare

(migliaia di branches paralleli)

Repository

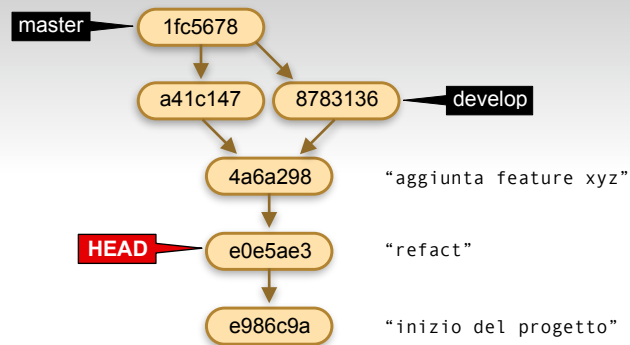
2 tipi di oggetti

● 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

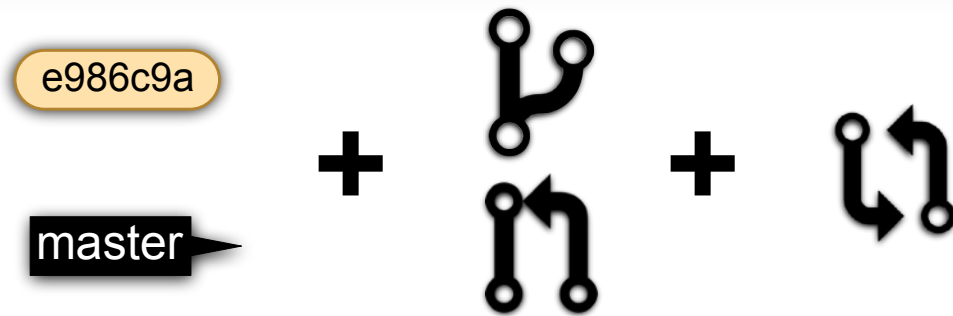
- Fusione tra due rami
- 3 tipi di merge: fast forward, clean, conflict



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



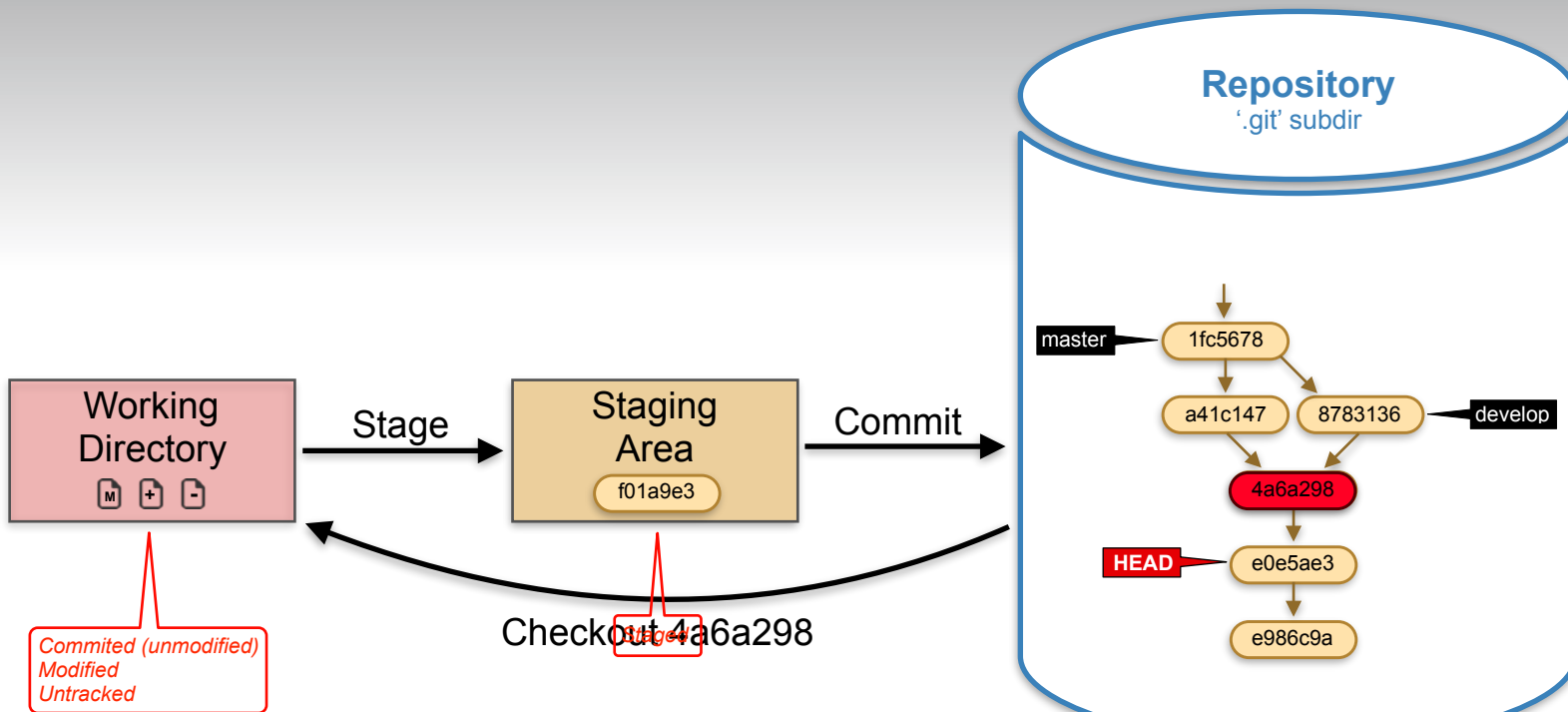
Files: 3 Stati Possibili

1. **Committed:** già salvato in modo sicuro nel repository locale
2. **Modified:** modificato e non ancora committato (modifiche pendenti)
3. **Staged:** file modificato marcato per essere incluso nel prossimo commit/snapshot

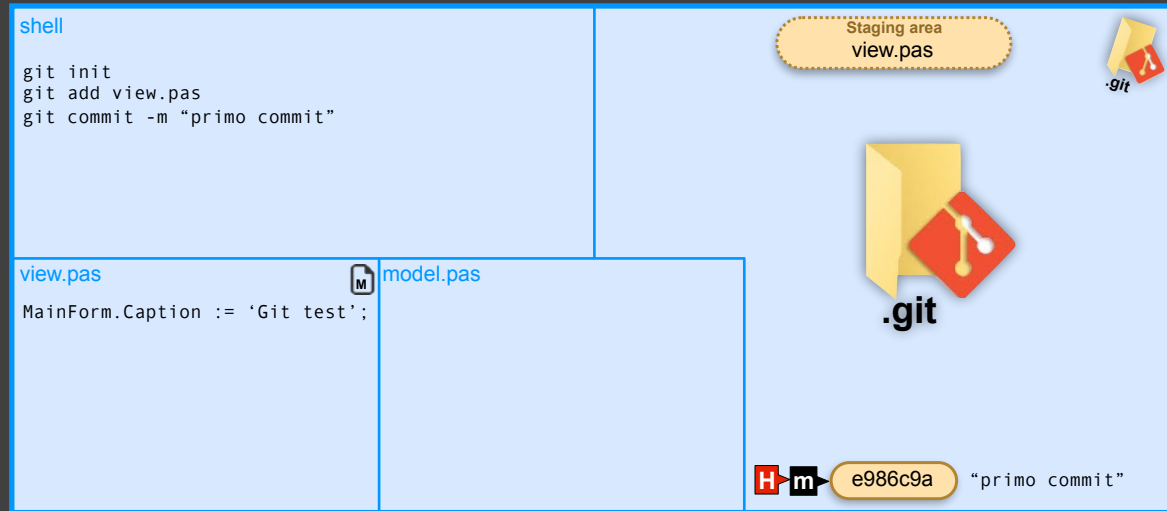
Untracked: file non tracciato da Git

Repository: 3 Sezioni

Local



1) Inizializzazione e 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..."

→

 e0e5ae3 "Risolto bug..."

→

e986c9a "primo commit"

4) Come era prima?

shell

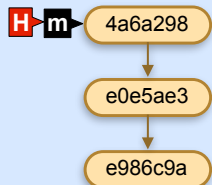
git checkout e0e5ae3

view.pas

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

model.pas

TCliente = class...



```
graph TD; H[H] -- m --> 4a6a298; 4a6a298 --> e0e5ae3; e0e5ae3 --> e986c9a;
```


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

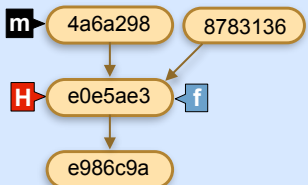


view.pas

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

model.pas

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



```
graph TD
    m[m] --> 4a6a298
    4a6a298 --> e0e5ae3
    e0e5ae3 --> e986c9a
    e0e5ae3 --> 8783136
    style m fill:#000,color:#fff
    style 4a6a298 fill:#f9d71c
    style e0e5ae3 fill:#f9d71c
    style e986c9a fill:#f9d71c
    style 8783136 fill:#f9d71c
    linkStyle 0 stroke:#f00,stroke-width:2px
    linkStyle 1 stroke:#f00,stroke-width:2px
    linkStyle 2 stroke:#f00,stroke-width:2px
    linkStyle 3 stroke:#f00,stroke-width:2px
```

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

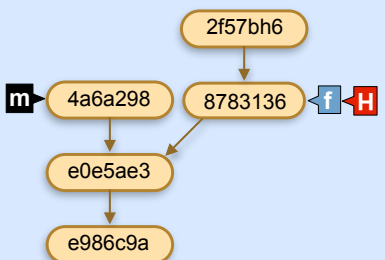


view.pas

```
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 e0e5ae3, and 8783136 points to e0e5ae3. The commit 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

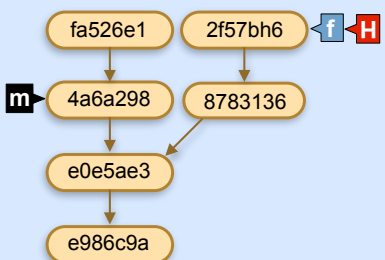


view.pas

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

model.pas

```
TNewClientClass...
property Age: integer read FAge;
```



```
graph TD
    fa526e1 --> 4a6a298
    2f57bh6 --> 8783136
    4a6a298 --> e0e5ae3
    8783136 --> e0e5ae3
    e0e5ae3 --> e986c9a
```

The diagram illustrates the Git commit history. It shows a sequence of commits represented by yellow rounded rectangles with their hash values. The commits are: fa526e1, 2f57bh6, 4a6a298, 8783136, e0e5ae3, and e986c9a. Arrows indicate the parent-child relationships: fa526e1 is the parent of 4a6a298; 2f57bh6 is the parent of 8783136; both 4a6a298 and 8783136 are parents of e0e5ae3; and e0e5ae3 is the parent of e986c9a. A black square with a white 'm' is next to 4a6a298. A blue square with a white 'f' and a red square with a white 'H' are next to 2f57bh6.

8) Merge (no conflict)

shell

```
git merge feature1
```

view.pas

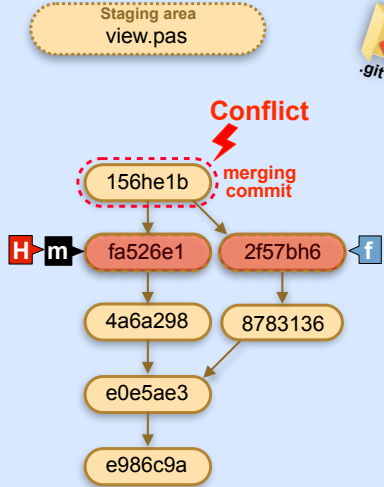
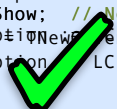
```
MainForm.Caption := 'Git test';  
MainForm.Show; // Necessario  
Label1.Caption := LCliente.Age;
```

model.pas

```
TClienteTaccass...  
property Age: integer read FAge;
```

The diagram illustrates a Git commit history. It starts with a root commit '156he1b' (yellow). This commit has two children: 'fa526e1' (red, labeled 'H-m') and '2f57bh6' (red, labeled 'f'). 'fa526e1' has a child '4a6a298' (yellow), which in turn has a child 'e0e5ae3' (yellow). '2f57bh6' has a child '8783136' (yellow), which also has a child 'e0e5ae3'. Finally, 'e0e5ae3' has a child 'e986c9a' (yellow). A small Git logo is visible in the top right corner of the diagram area.

9) Merge (with conflict)

shell <pre>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"</pre>	Staging area view.pas  Conflict merging commit 156he1b fa526e1 2f57bh6 4a6a298 8783136 e0e5ae3 e986c9a 	
view.pas <pre>MainForm.Caption := 'Git test'; MainForm.Show; //Necessario Label1.Caption := NewClient.FAge; Label1.Caption := LCliente.Age;</pre> 	model.pas <pre>ICliente = class... <<<<<< HEAD TCliente = c ===== TNewClient... property Age: integer read FAge; >>>>>> feature1 ...</pre> 	

10) Eliminiamo il branch (oppure no?)

shell

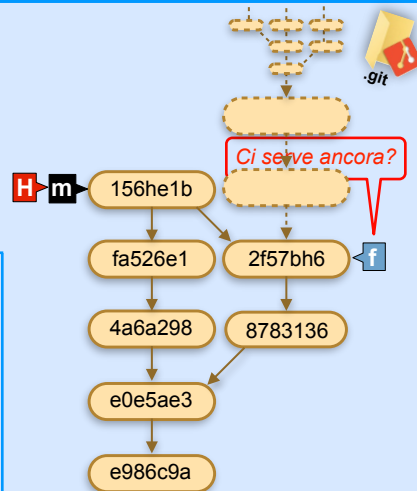
```
git merge feature1
Auto-merging model.pas
CONFLICT (content): Merge conflict in model.pas
Automatic merge failed; fix conflicts and then
commit the result.
git add model.pas
git commit "feature1 merged on master"
git branch -D feature1
```

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

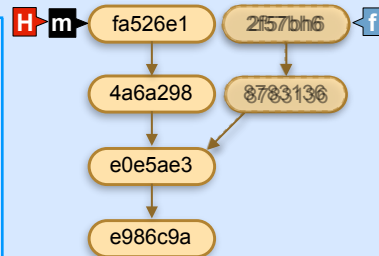
fa526e1 HEAD@{0}: checkout: moving from branch feature1 to master
2f57bh6 HEAD@{1}: commit: visualizzazione età cli...
8783136 HEAD@{2}: commit: aggiunto proprietà Age
E0e5ae3 HEAD@{4}: checkout: moving from branch master to feature1
...
git branch oldfeature1 HEAD@{1}
```

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
```

view.pas

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

model.pas

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

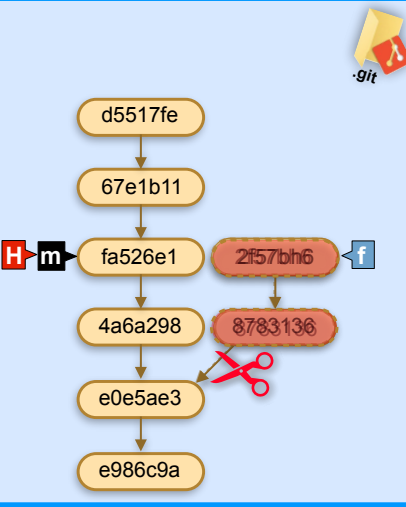
Git Stash Diagram

The diagram illustrates the Git Stash workflow. It shows a sequence of commits: fa526e1, 4a6a298, e0e5ae3, and e986c9a. A stash is created from fa526e1, resulting in a new commit 2f57bh6. The stash is then applied to the next commit, 8783136. The diagram also shows a 'Pending' state with a red lightning bolt icon and a 'Stash' folder icon.

```
graph TD
    fa526e1 --> 4a6a298
    4a6a298 --> e0e5ae3
    e0e5ae3 --> e986c9a
    fa526e1 --> 2f57bh6
    2f57bh6 --> 8783136
    8783136 --> e0e5ae3
```


13) Rebase (no conflict)

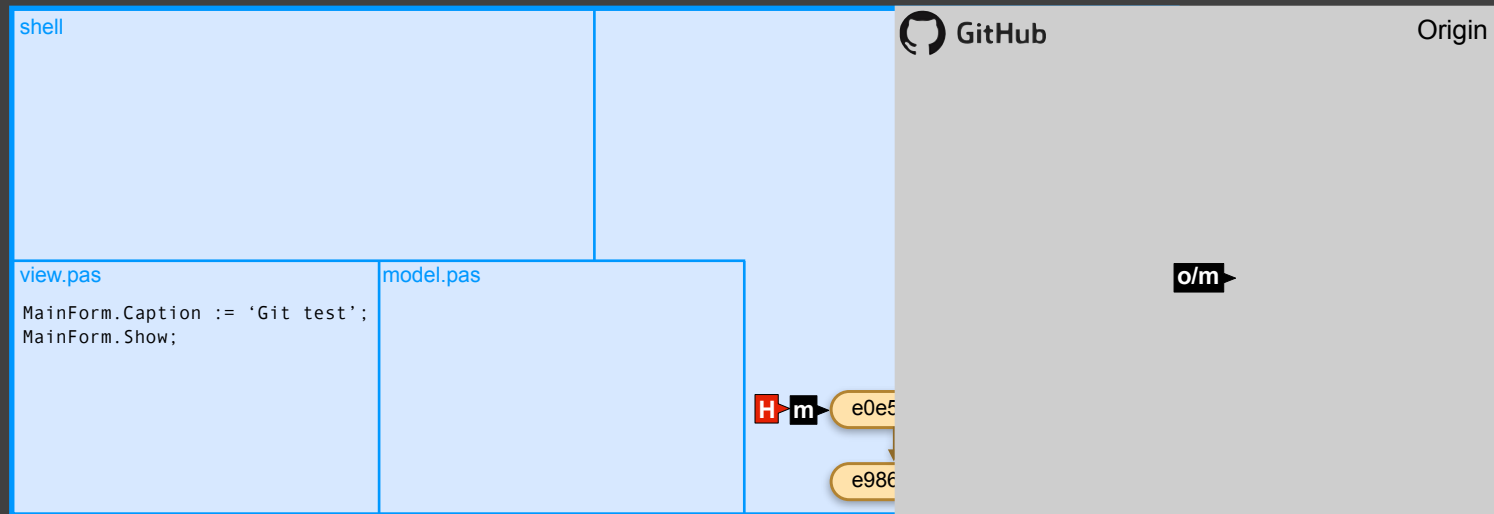
Fast forward merge alla fine

shell <pre>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</pre>		
<table border="1"><tr><td data-bbox="382 547 749 806"> view.pas <pre>MainForm.Caption := 'Git test'; MainForm.Show; // Necessario LCliente := TNewCliente.Create; Label1.Caption := LCliente.Age;</pre></td><td data-bbox="749 547 1116 806"> model.pas <pre>TNewCliente = class... property Age: integer read FAge;</pre></td></tr></table>		view.pas <pre>MainForm.Caption := 'Git test'; MainForm.Show; // Necessario LCliente := TNewCliente.Create; Label1.Caption := LCliente.Age;</pre>
view.pas <pre>MainForm.Caption := 'Git test'; MainForm.Show; // Necessario LCliente := TNewCliente.Create; Label1.Caption := LCliente.Age;</pre>	model.pas <pre>TNewCliente = class... property Age: integer read FAge;</pre>	

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

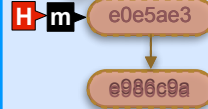
```
git remote add origin https://github.com/mauriziodm/delphiday2019.git
git push --set-upstream origin master
git push
```



view.pas

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

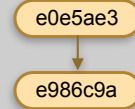
model.pas



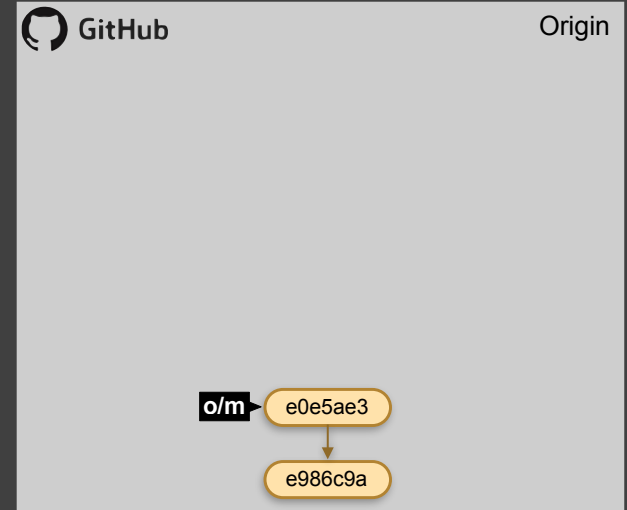
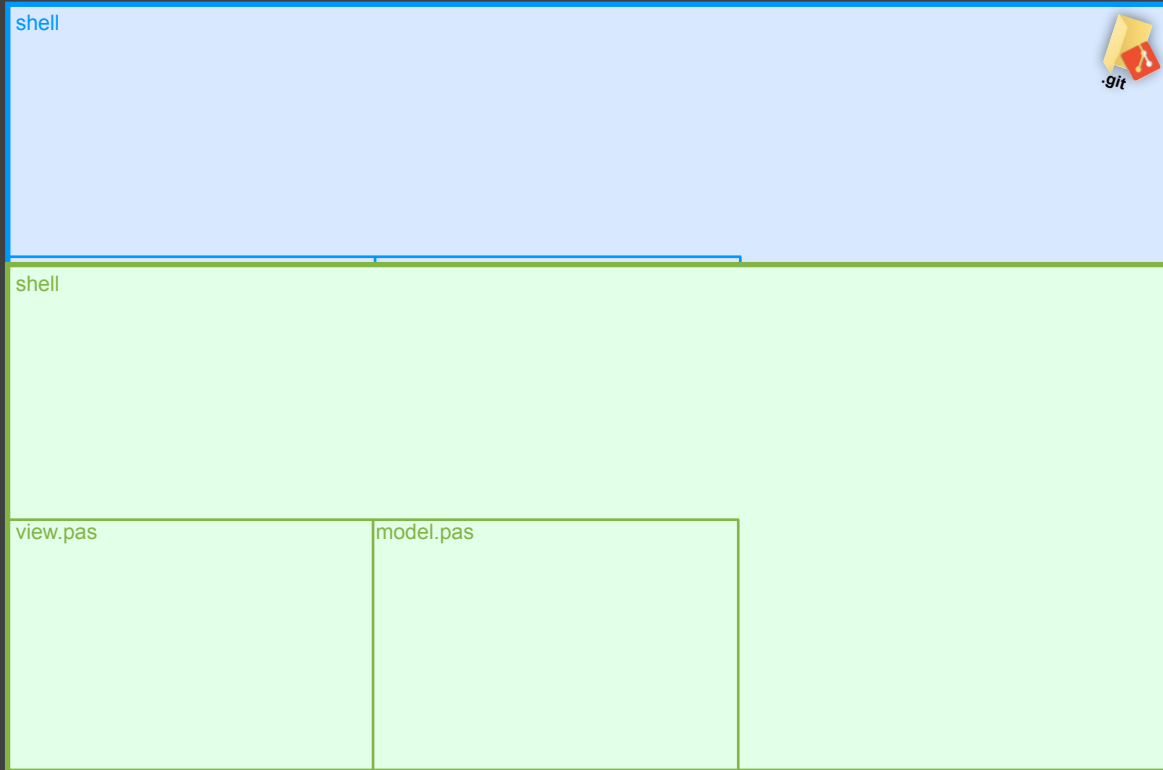
GitHub

Origin

o/m



16) Un nuovo contributor

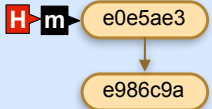


shell

view.pas

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

model.pas



A diagram showing a commit 'H-m' at hash e0e5ae3. An arrow points down to a new commit at hash e986c9a.

17) Un nuovo contributor git clone

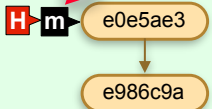
shell

```
git clone https://mauriziodm:password@github.com/mauriziodm/delphiday2019.git
```

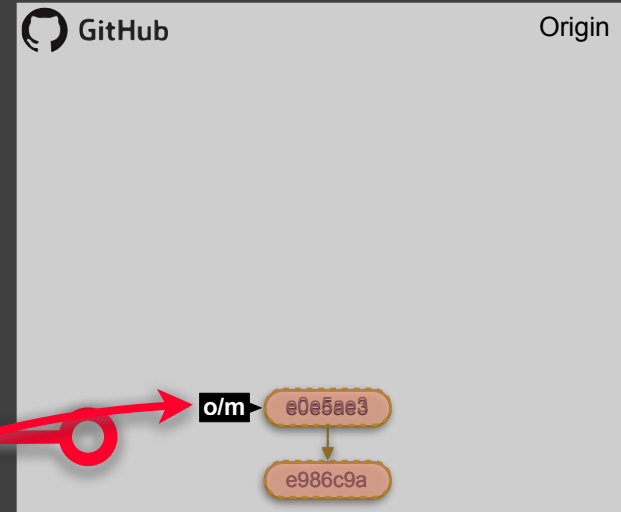
view.pas

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

model.pas



A diagram showing a commit 'H-m' at hash e0e5ae3. An arrow points down to a new commit at hash e986c9a.



shell

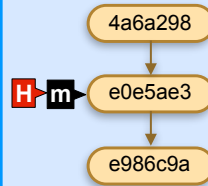
```
git fetch
git status
On branch master
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)
```

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

view.pas

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

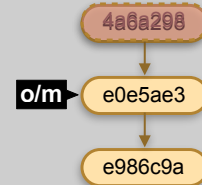
model.pas



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



Origin



shell

```
git clone https://mauriziodm:password@github.com/mauriziodm/delphiday2019.git
git commit -a -m "aggiunta classe TNewCliente"
git push
```

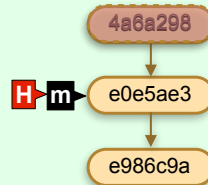
Staging area
model.pas



view.pas

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

model.pas



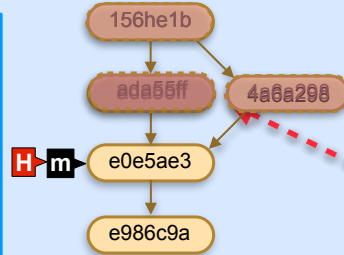
```
shell
git commit -a -m "aggiunta classe TOtherCliente"
git push
git pull
git push
```

Staging area
model.pas



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

```
model.pas
TOtherCliente = class...
```



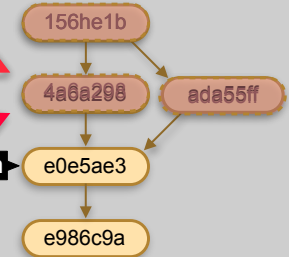
19) Un nuovo contributore push/pull (same branch, divergent)



Origin

Divergent

o/m



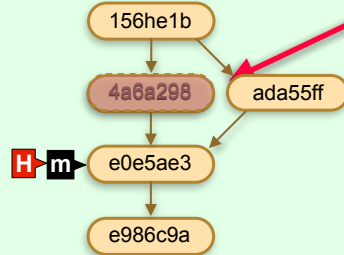
```
shell
git clone https://mauriziodm:password@github.com/mauriziodm/delphiday2019.git
git commit -a -m "aggiunta classe TNewCliente"
git push
git pull
```

Staging area
model.pas



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

```
model.pas
TNewCliente = class...
```



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

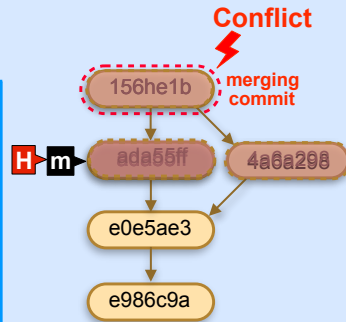


view.pas

```
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/delphiday2019.git
git commit -a -m "aggiunta classe TNewCliente"
git push
git pull
```

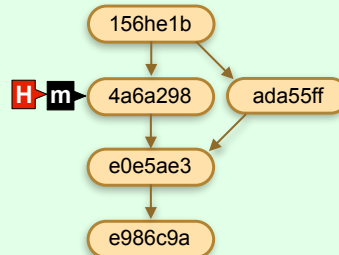


view.pas

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

model.pas

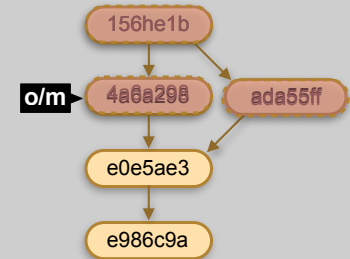
```
TNewCliente = class...
```



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
```

view.pas

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

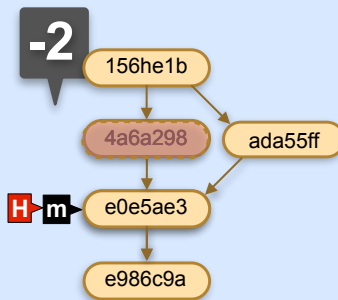
model.pas

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

Staging area
model.pas



-2



shell

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

view.pas

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

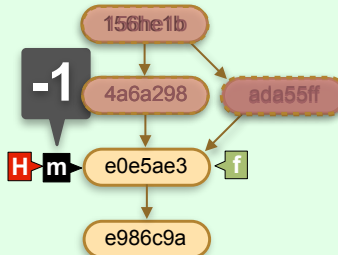
model.pas

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

Staging area
model.pas



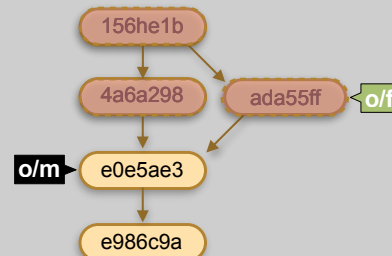
-1

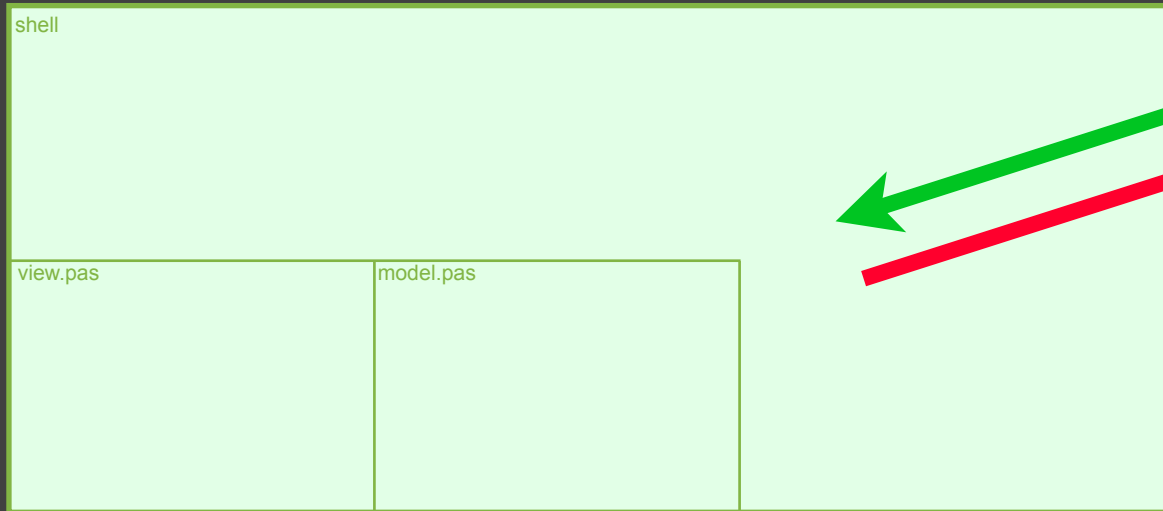
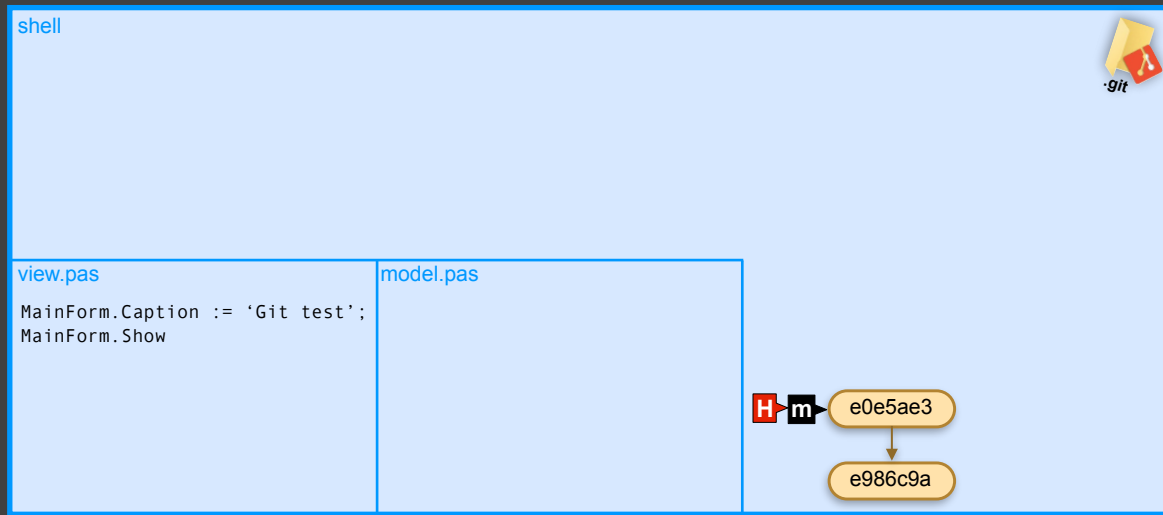


21) Un nuovo contributore branch separati

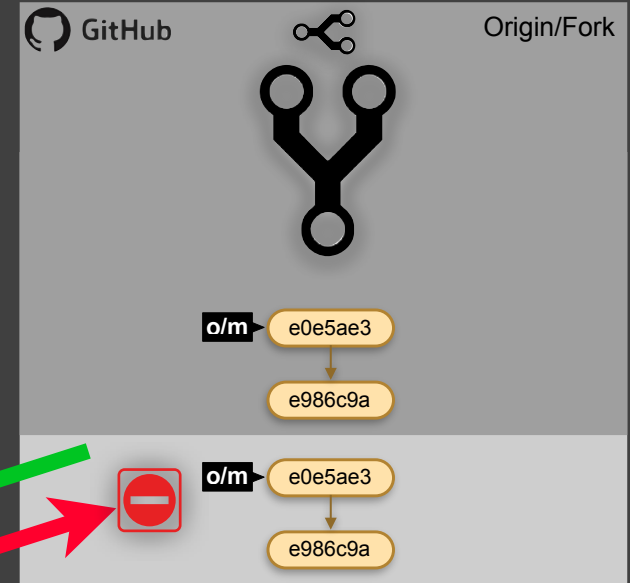


Origin





22) Se il contributor non ha i diritti?
fork

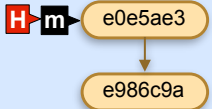


shell

view.pas

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

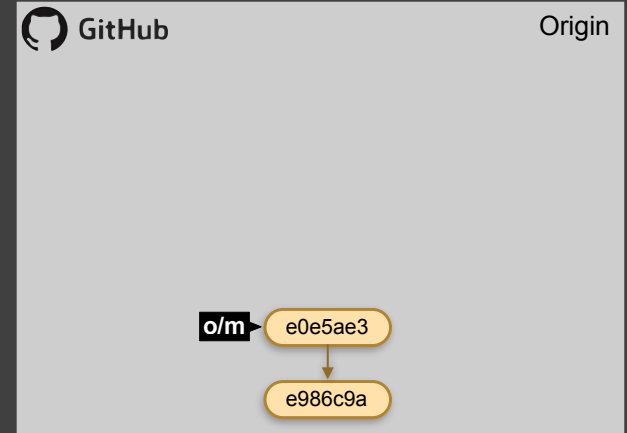
model.pas



Git icon

23) Se il contributor non ha i diritti?

Fork: clone locale



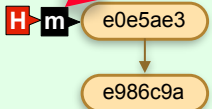
shell

```
git clone https://rinobosco70:BoscoRino70!!@github.com/mauriziodm/delphiday2019.git
```

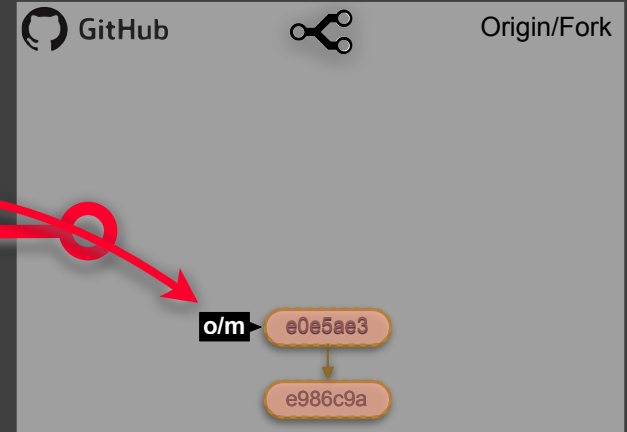
view.pas

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

model.pas



Git icon



shell

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

Staging area
model.pas



view.pas

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

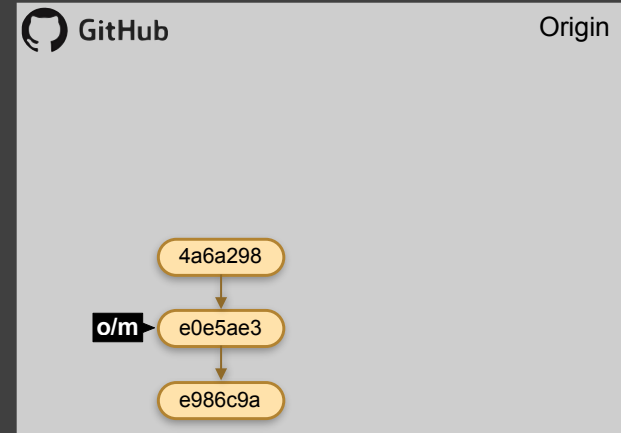
model.pas

TOtherCliente = class...

```

graph TD
    A[4a6a298] --> B[e0e5ae3]
    B --> C[e986c9a]
    style B fill:#f9f,stroke:#333,stroke-width:1px
    
```

24) Se il contributor non ha i diritti?
New Branch (ora possiamo)
Push



shell

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

Staging area
model.pas



view.pas

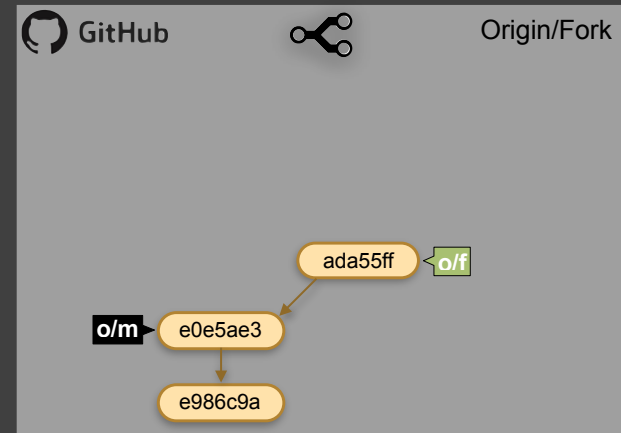
```
MainForm.Caption := 'Git test';
MainForm.Show;
```

model.pas

TNewCliente = class...

```

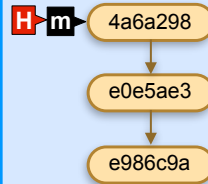
graph TD
    A[4a6a298] --> B[e0e5ae3]
    B --> C[e986c9a]
    B --> D[ada55ff]
    style B fill:#f9f,stroke:#333,stroke-width:1px
    style D fill:#f9f,stroke:#333,stroke-width:1px
    
```



```
git commit -a -m "aggiunto classe T0therCliente"
git push
```

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

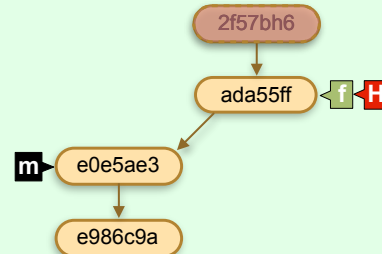
```
T0therCliente = class...
```



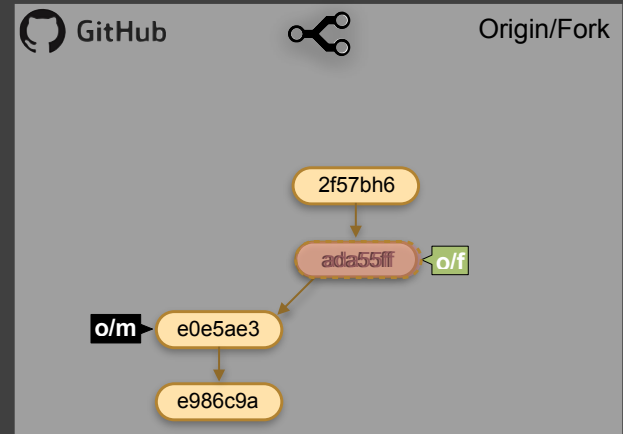
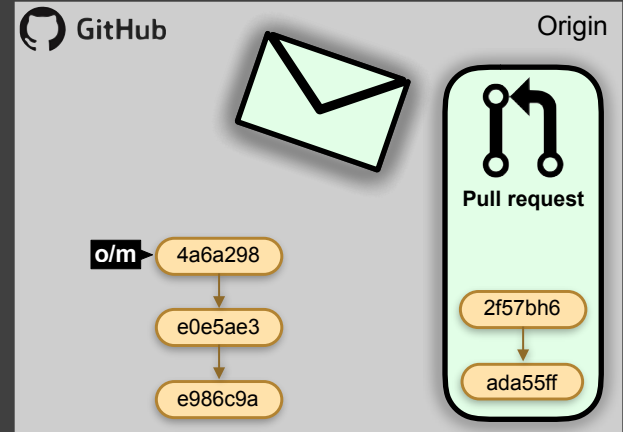
```
git branch newfeature
git checkout newfeature
git commit -a -m "nuova classe TNewCliente"
git push
git commit -a -m "aggiunto proprietà age"
git push
```

```
MainForm.Caption := 'Git test'; TNewCliente = class...
MainForm.Show;      property Age: integer read FAge;
```

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



Pull request (su GitHub)



shell

view.pas

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

model.pas

TOtherCliente = class...
MainForm.Show

H

m

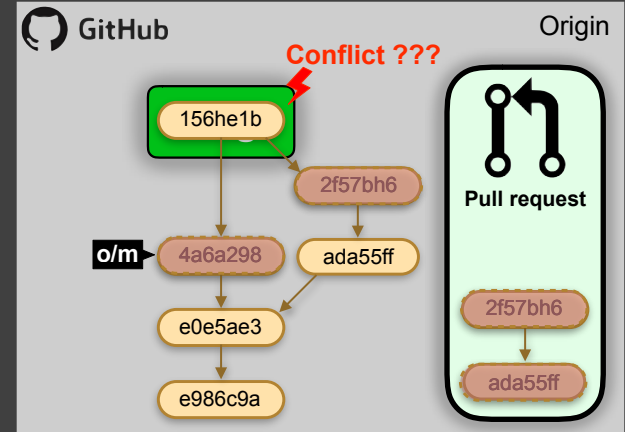
4a6a298

e0e5ae3

e986c9a

26) Se il contributor non ha i diritti?

Pull request: merge a cura del manteiner



shell

view.pas

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

model.pas

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

m

f

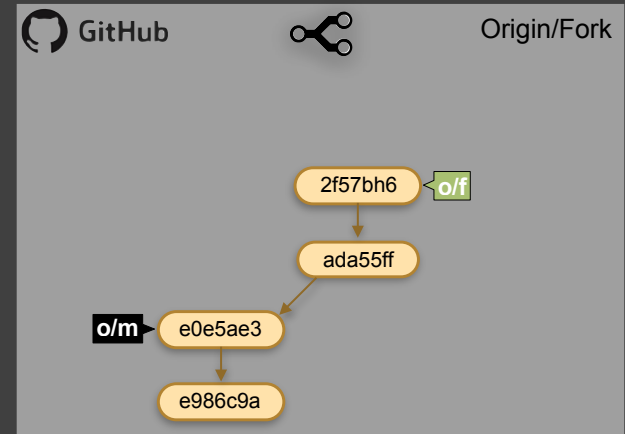
H

2f57bh6

ada55ff

e0e5ae3

e986c9a



shell

git fetch
git pull

view.pas

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

model.pas

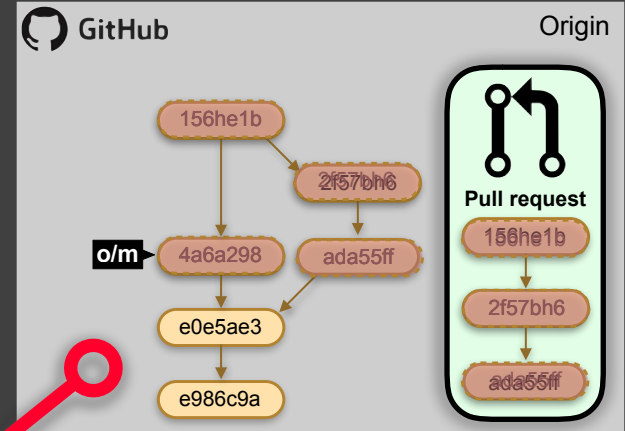
TOtherCliente = class...

```

graph TD
    156he1b --> 4a6a298
    156he1b --> 2f57bh6
    4a6a298 --> e0e5ae3
    2f57bh6 --> ada55ff
    e0e5ae3 --> e986c9a
    
```

27) Se il contributor non ha i diritti?

Pull request: merge a cura del contributor



shell

git remote add upstream https://github.com/mauriziodm/delphiday2019.git
git fetch upstream
git merge upstream/master
git push

view.pas

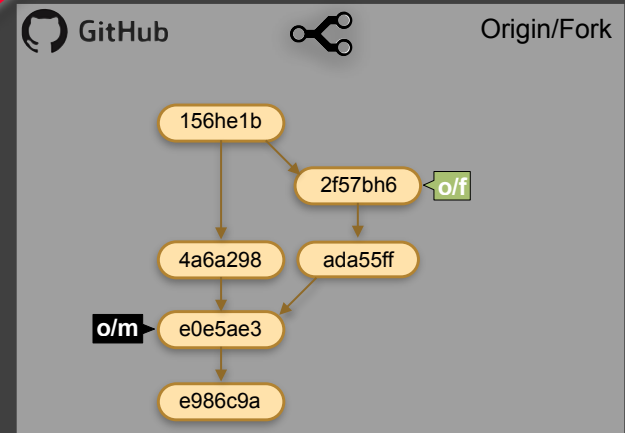
MainForm.Caption := 'Git test';
MainForm.Show;

model.pas

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

```

graph TD
    156he1b --> 4a6a298
    156he1b --> 2f57bh6
    4a6a298 --> e0e5ae3
    2f57bh6 --> ada55ff
    e0e5ae3 --> e986c9a
    
```



DOMANDE?





Maurizio Del Magno
Developer



levante 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>



2019 - XVIII Edizione

Grazie!!!