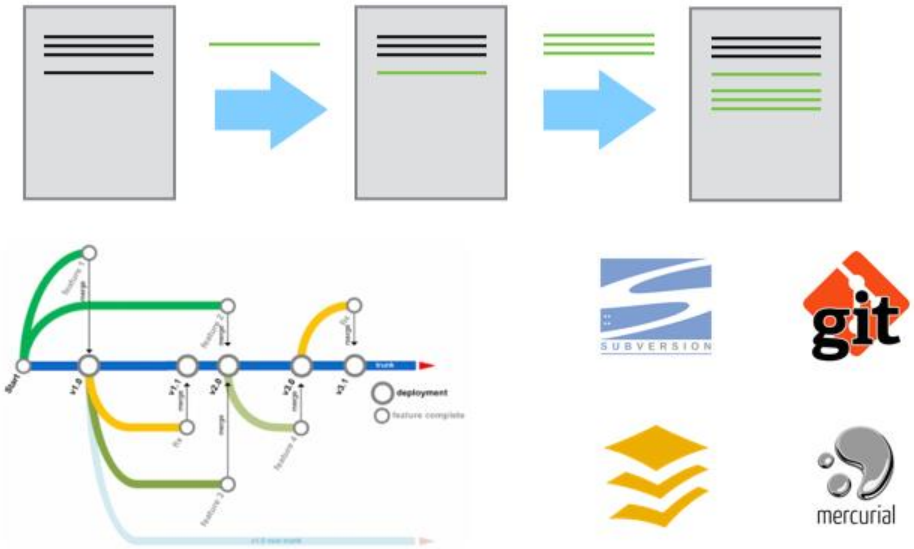


controllo di versione

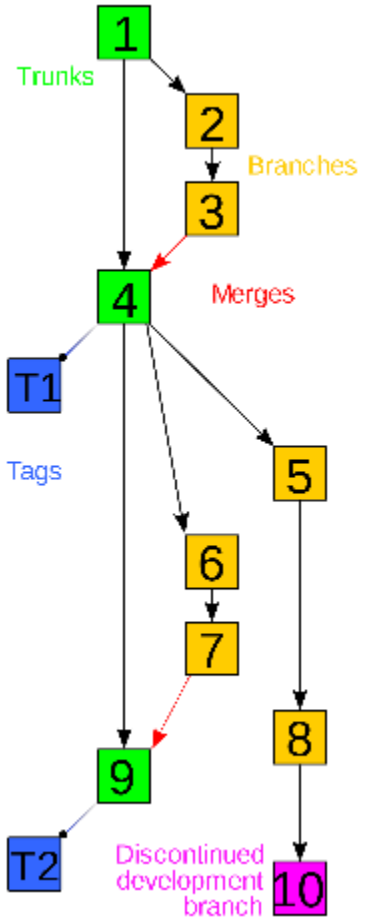
version control systems

version control system

- il controllo di versione è un processo che consente di tenere ***traccia*** di tutte le ***modifiche*** effettuate nel tempo ad un progetto (un insieme di file)
- nel sistema per il controllo di versione viene registrata
  - la ***modifica***
  - ***chi*** ha effettuato la modifica
  - ***quando*** è stata effettuata la modifica
  - il ***motivo*** per cui un file è stato modificato (commento)
- è possibile ***ritornare*** a una versione precedente del progetto e recuperare file che sono stati erroneamente eliminati

software per il versioning

- esistono vari sistemi per il controllo di versione
  - Subversion
  - Perforce
  - TFS
  - ***Git***
  - Mercurial



vantaggi dei sistemi di versioning

A. Ferrari

singolo utente

- backup
- cronologia
- ritorno a versioni precedenti
  - (singolo file o progetto)
- sviluppi sperimentali
  - branch

team di sviluppo

- sincronizzazione
  - aggiornamento copia personale con modifiche di altri
- responsabilità
  - modifica (chi e perché)
- gestione conflitti
  - modifiche incompatibili

concetti e termini

- ***repository***
  - luogo in cui sono archiviati file e cronologia
  - database con tutte le informazioni
- ***working set***
  - progetti all'interno del repository (file locali)
- ***add***
  - aggiornamento del working set
- ***check-in*** (***commit***)
  - aggiornamento del repository dal working set
- c***heck-out*** (***update***)
  - aggiornamento del working set con una versione dei dati dal repository
- ***tag (label)***
  - identificatore dello stato di un intero albero di un progetto o di un repository

- git è un software di controllo versione creato da Linus Torvalds nel ***2005***
- git (nello slang americano significa ***idiota***) nasce per essere un semplice strumento per facilitare lo sviluppo del ***kernel Linux***
- oggi è diventato uno degli strumenti di controllo versione ***più diffusi***

git - utilizzo

- *# creazione cartella locale per repository*

```
$ mkdir repo/demo
```

- *# posizionamento in cartella repository*

```
$ cd repo/demo
```

- *# inizializzazione (creazione) repository*

```
$ git init
Initialized empty Git repository in xxx/repo/demo/.git/
```

- *Git memorizza tutte le informazioni nella directory nascosta .git.*
- *# visualizzazione dello stato del repository*

```
$ git status
```

git – utilizzo

- nella cartella del working set è possibile creare e/o modificare file di qualsiasi tipo (es MyClass.java)
- è necessario comunicare a git che i file devono essere ***inseriti nel working*** set con il comando add specificando il singolo file o tutti i file (*)

```
$ git add MyClass.java
$ git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   MyClass.java
```

inserimento nel repository

- commit aggiorna il repository

```
$ git commit -a -m "Aggiunta classe MyClass"
[master (root-commit) d9cffef] Aggiunta classe MyClass
 1 file changed, 3 insertions(+)
 create mode 100644 MyClass.java
```

- stato del repository

```
$ git log
commit d9cffefa554f4ce5552debd9fc460f6aa76bbebe (HEAD -> master)
Author: Alberto Ferrari <alberto.ferrari@unipr.it>
Date:   Mon May 13 16:43:48 2019 +0200
Aggiunta classe MyClass
```

modifiche

- dopo aver effettuato modifiche ai file nella cartella è possibile effettuare un confronto con quanto memorizzato nel repository

$ git log -p

```
…
diff --git a/MyClass.java b/MyClass.java
new file mode 100644
index 0000000..ef8a32f
--- /dev/null
+++ b/MyClass.java
@@ -0,0 +1,3 @@
+public class MyClass {
+
+}
```

ritorno a versioni precedenti

A. Ferrari

- in alcuni casi si rende necessario tornare a versioni precedenti del progetto
- numSha è il codice associato a ogni commit

```
$ git reset --hard numSha
```

- i file vengono riportati alla versione specificata

github

- GitHub è un ***servizio di hosting*** per progetti software
- è una ***implementazione*** dello strumento di controllo versione distribuito ***Git***
- utilizzato da: *Google, Apple, Microsoft, NASA, Facebook, Twitter …*
- fondata nel 2008
- nel 2009 135.000 repository pubblici
- 2013 3 milioni di utenti e più di 5 milioni di repository
- 2018 Microsoft acquisto (7,5 miliardi di dollari)

operare con github

A. Ferrari

- ***creazione*** di un repository
  - importante includere sempre un README o un file con informazioni sul progetto
- creazione di un ***branch***
  - un branch permette di lavorare su più versioni di un progetto
  - default branch master
- ***commit***
  - le modifiche salvate sono chiamate commit
  - ogni commit ha un messaggio associato (una descrizione che spiega perché una particolare modifica è stata fatta)
- ***pull request***
  - una pull request propone le modifiche
  - le pull requests mostrano le differenze dei contenuti dei vari i branch
- ***fondere*** una pull request