Accademy 2022 - Sesa Group

Versionamento codice con Git

Docente: **Samuele Di Rito**

T&O Consulting

Cos'è Git?

- permette di far collaborare tra loro più sviluppatori contemporaneamente sul medesimo progetto
- rende possibile la gestione di multiple versioni di un insieme di files
- è gratuito


alt text

Git - Walkthrough #1

- Cos'è un repository e come si crea
- Cos'è una branch
- Cos'è un commit e come si modifica una branch
- Operazioni locali e operazioni remote
- Comando: git add
- Comando: git commit
- Comando: git push
- Comando: git pull

Cos'è un repository e come si crea

Il repo è il magazzino delle informazioni relative al nostro progetto.

Un repository git è una cartella `.git` all'interno di un progetto che crea un'intera cronologia del progetto stesso, dalla sua inizializzazione fino all'ultima sincronizzazione

> La cartella `.git` è intoccabile. L'unica maniera di interagire con il suo contenuto è usare i comandi git.

Cos'è un repository e come si crea (continua...)

1. **Creare una nuova cartella**

Windows	Unix
`md training-git-sesa`	`mkdir training-git-sesa`
`cd training-git-sesa`	`cd training-git-sesa`

2. Creare un nuovo repo in questa cartella con il comando:

   `git init`


alt text

Cambiare l'author

3. Impostazione user name e email

```
git config user.name "Samuele Di Rito"
git config user.email "samuele.dirito@toconsulting.it"
```

I comandi non danno output. Verificheremo più tardi se funzionano o no.

Cos'è una branch

Una branch è un puntatore ad un commit. E' necessario assegnare un nome in fase di creazione.

La branch `master` ha la sola particolarità di essere la prima generata a valle dell'inizializzazione del repo git.

Cos'è un commit

Ogni commit è l'istantanea della working directory ad un certo istante di tempo.

Ogni nuovo commit creato nel repository, mantiene al suo interno un riferimento a un commit precedente.

Comandi: `git add` / `git commit`

```
git add <percorso-completo-file>
git commit -m "<Messaggio del commit>"
```

Untracked e tracked files

Untracked: file ancora da aggiungere ad un commit

Tracked: file aggiunti ad un commit

Comandi: `git log`

Serve per visualizzare la storia del branch corrente.

Questo è il comando per avere un log sintetico:

```
git log
```

Questo è il comando per avere il log sintetico e l'elenco di tutti i files modificati

```
git log --stat
```

Proviamo

Fisionomia di un commit

- Hash (autogenerato)
- Author (letto da configurazione)
- Date (data del **commit**)
- Message (messaggio del commit)

Puntatore ad HEAD

Il commit più recente si chiama HEAD.

Ogni volta che committiamo, HEAD si sposta in avanti.

Questo concetto sarà utile per verificare il funzionamento della branches in git

Comandi locali (recap)

```
git init
git add
git commit
git log
```

Comandi: `git pull` / `git push`

```
git push
```

Serve ad inviare i commit della branch corrente, presenti soltanto localmente, al server git remoto.
Se la push va a buon fine, gli altri team member potranno pullare le modifiche e aggiornare la propria HEAD in locale.

```
git pull
```

Serve a recuperare i commit della branch corrente da remoto ed aggiornare la propria HEAD in locale. Si parla di allineamento della branch a remoto.

Risoluzione dei conflitti

Quando cominciamo a lavorare con i comandi remoti (pull, push), ci troviamo in situazioni in cui gli stessi file vengono modificati da commit provenienti da altri sviluppatori del team.

Git ha un proprio meccanismo di risoluzione dei conflitti, che entra in gioco nel momento in cui ci allineiamo a remoto.

In un caso simile, le modifiche che saranno in conflitto verranno segnalate dal `git status` che è il modo per verificare la situazione attuale della branch rispetto a remoto e per analizzare l'attuale pacchetto di commit.

Proviamo

Checkout

Permette di spostarsi tra una branch e un'altra.

Quando facciamo il checkout, tutte le modifiche pendenti vengono trasferite sulla branch di destinazione dalla branch originale.

Dobbiamo quindi accertarci che non ci siano modifiche prima di lanciarlo.

Per non avere modifiche pendenti, abbiamo due strade possibili:

- commit
- stash

Stash

Permette di salvare totalmente o parzialmente le modifiche pending della nostra branch.

Lo stash NON è un commit.

Ma allo stessa maniera in cui un commit traccia le modifiche sui files, anche lo stash lo fa.

Nel momento in cui viene creato, tutte le modifiche pending spariscono dalla branch e vengono messe da parte.

Dopo aver creato lo stash, è possibile estrarlo in un qualsiasi momento e riapplicare le modifiche "stashate" alla branch nello stato attuale.

Rebase

Il rebase è l'operazione che aggiorna la branch su cui siamo posizionati facendola ripuntare all'HEAD del branch di origine.

E' un'operazione locale.

Necessita quindi di aver pullato in locale tutte le modifiche del branch di origine.

Comandi: `git rebase <branch>`

```
git rebase <nome-branch>
```

Fa partire il processo di rebase. Per ogni commit "mancante" nella branch corrente, Git chiama il risolutore dei conflitti. Se non ne trova, procede al successivo commit senza warnings. Altrimenti dovranno prima essere risolti per poter procedere con il successivo.

Al termine della risoluzione, si può procedere con la seguente sequenza di comandi:

```
git add .
git rebase --continue
```


alt text

Dev Experience - Lavoro quotidiano

- Si committa ogni volta che funziona qualcosa
- Si pusha quando si arriva a fine serata
- Si pusha quando si completa il task assegnato
- Si pulla master la mattina
- Si rebasa ogni volta che si pulla master
- Quando si pusha, si fa una MR

Branching strategies #1

Feature Branch Workflow

Fondamentale nei progetti con piccoli team o personali.

https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow

Branching strategies #2

Gitflow Workflow

Fondamentale nei progetti con team medio/grandi.

https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow