**Spiegare ruolo di “origin”**

“Origin” è il nome che fa riferimento al repository remoto. Esso contiene la storia completa del progetto ed ogni sviluppatore può clonare il repository remoto così da avere una copia in locale(sistema distribuito) su cui lavorare e apportare modifiche che poi verrano pushate nell’ “Origin”.

**Comandi per creazione repository da riga di comando**

Ci sono due modi per creare un repository:

1. Con l’istruzione **$ git init [project-name]** crei un repository locale con un nome specificato dentro una cartella esistente.

Con l’istruzione **$** **git remote add origin <remote\_repo\_url>** una volta che abbiamo il setup del repository remoto aggiungiamo l’url remoto alla nostra configurazione git locale.

Con l’istruzione **$** **git push -u origin master** settiamo un branch master per i nostri branches locali.

1. Usando il comando **$ git clone [url]** cloniamo il progetto con la sua intera storia di versionamento che ci permette di avere una copia in locale.

**Eseguire una git push force e spiegare le conseguenze di tale azione**

Di solito il comando **$** **git push**, il comando rifiuta di aggiornare un riferimento remoto che non è un antenato del riferimento locale utilizzato per sovrascriverlo. Inoltre, quando viene utilizzata l'opzione " --force-with-lease ", il comando rifiuta di aggiornare un riferimento remoto il cui valore corrente non corrisponde a quello previsto. Con **$** **git push –f** vengono disabilitati questi controlli e questo può causare al repository remoto la perdita delle commits effettuate prima del push force. Si consiglia di usurlo con attenzione.

**Ogni commit ha un id univoco. Documentatevi sul come tale id viene assegnato. (non è necessario scendere troppo nel dettaglio).**

L’id di una commit viene assegnato in base a queste informazioni:

* La checksum dell’albero dei contenuti
* Il commit id genitore
* L’autore della commit con timestamp
* Il committer della commit con timestamp
* Il messaggio della commit

Git prende questi dati e genera un codice di 40 cifre in esadecimale tramite l’alogoritmo hash SHA1 effettuato su di essi.

**Spiegare differenza fra una revert e una reset, con relativi esempi**

Una revert crea una nuova commit da quella precedente alle modifiche e conserva la storia delle commit, mentre la reset cancella la commit e ti pulisce la stage area(non reversibile).

Revert : A <- B <- C A <- B <- C <- B'

^ HEAD ^ HEAD

Reset : A <- B <- C A <- B

^ HEAD ^ HEAD

**Spiegare i due metodi di integrazione branch visti finora, con relativi esempi**

* La merge effettua l’integrazione dei due branch e non tiene traccia della storia delle commit ma ne viene creata una nuove della merge, il rebase non fa altro che prendere tutti i cambiamenti che sono stati inviati su un ramo ed applicarli su un altro tenendo traccia della storia delle commit.