1. **Spiegare ruolo di “origin”**

“Origin” rappresenta l’alias che Git dà di default al clone di un progetto disponibile

online.

Quindi una volta inizializzato un repository, lo si clona ritrovandolo con il nome

origin.

Ovviamente tale alias può essere modificato.

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

*git init* (per creare una repository git).

*git clone* (crea una copia di un repository locale).

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

Tale comando elimina i commit precedenti e forza quello corrente che voglio pushare.

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

Gli id che Git utilizza per rendere i commit univoci sono calcolati in basi ad un algoritmo *SHA-1*. E’ una stringa di 40 caratteri esadecimale ossia da 0 a 9 e da a-f calcolati in base al contenuto di un file o alla struttura della directory.

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

*git reset* si usa per rimuovere le commit sul branch su cui ci si trova. Tale comando è un modo semplice per annullare le modifiche che non sono state ancora condivise con altri.

Mentre con *git revert* non perdo la storia delle commit precedenti .

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

Sono stati visti due metodi di integrazione: merge e rebase.

Entrambe consentono di includere le modifiche eseguite a carico di un ramo in un altro ramo.

Il primo merge più semplice che può essere effettuato è quello tra develop e feature che permettere di gestire i conflitti prima di essere pushati.

In particolare la differenza nei due metodi è che nel primo si perde la storia dei commit precedenti mentre il secondo prende tutte le modifiche che sono state effetuate e le effettua su un altro ramo.