1. “Origin” è un alias del sistema destinata alla repository remota, ossia un nome di default impostato come chiave in sostituzione dell’URL della repository remota.
2. Da riga di comando digitiamo: **$ git init**, per la creazione di una nuova repository mentre per creare una copia sulla repository locale digitiamo: **$** **git clone /directory della repository**. Inoltre, per aggiungere file alla nostra repository basta digitare: **$** **git add “nome file”** con un commit successivo.

Successivamente, tramite l’URL della repository remota inseriamo nel prompt dei comandi: **$** **git remote add origin URL**, quest’ultimo aggiunge l’URL della repository remota a quella locale.

1. Con **$** **git push<nome> –force** o **git push -f c**ancella la commit più recente e viene eseguito un push sulla commit attuale, quindi se si effettua una merge la force sovrascriverà la commit precedente.
2. Un’ID contenente le informazioni di un commit:

* La data del commit
* L’autore del commit
* Il messaggio di log digitando: **$** **git commit -m “messaggio”**
* L’email che ha eseguito il commit.
* L’ID del commit precedente.

1. **$ git reset --*file*** rimuove i file dallo stage; ovvero copia i *file* dell'ultimo commit nello stage. Utilizza questo comando per annullare un git add *file*. Puoi anche utilizzare git reset per rimuovere tutto dallo stage. Invece, git revert mantiene la storia dei commit precedenti e allo stesso tempo ne crea uno nuovo. Per esempio se una commit è finita nella repository tramite il push è necessario eseguire un’altra procedura **$ git revert HEAD,** questo comando crea una nuova commit annullando quella indesiderata
2. Metodi d’integrazione branch: registrando la storia delle commit precedenti creiamo una merge, o meglio, unisce la cronologia di due branch in quella attuale e successivamente viene creata una nuova cronologia di branch. Nel caso della rebase che si occupa di **riscrivere la cronologia di un progetto**, essa genera dei nuovi commit su un ramo e ne applica in un altro. In cui vi sia la necessità di incorporare frequentemente i commit di un ramo nell’altro il rebasing non crea delle sovrapposizioni tra rami, questo significa che la cronologia stessa risulterà più comprensibile.

Comando per il merge**: $** **git merge –no -ff** <branch name>

Comando per il rebase: **$ git rebase <base>,** dove il base può essere l’ID del nome del branch e successivamente **$ git rebase –i origin/develop**