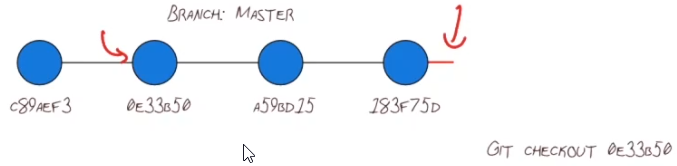
# Cisco DevNet Associate (200-901 DEVASC) – CBT Nuggets Training

Git - Version control

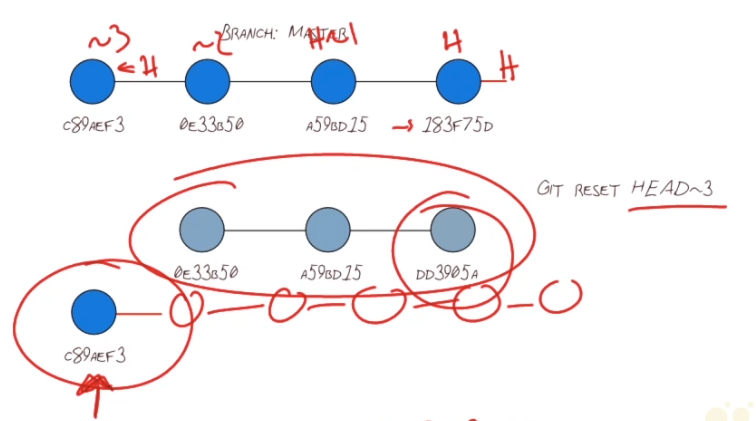
* There are 3-steps in the lifecycle of a file

1. Unmodified
   1. When the file is modified you are making a CHANGE
2. Modified
   1. After the file is modified you should move it to STAGING, with the command *git add .*
3. Staged
   1. Once the file is staged, this version is ready to be committed to the repository with *git commit -m “meaninful message”*

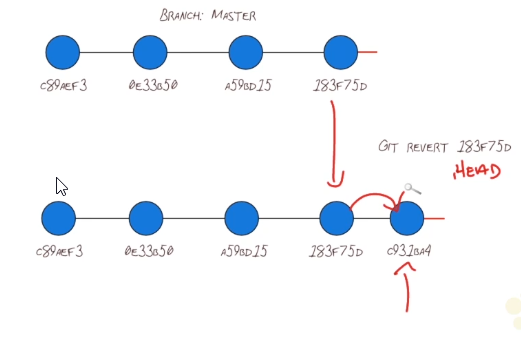
* GIT CHECK OUT
  + We can revert back to any previous commit in a brench.
  + With *git log*, you can see all your commits. And with *git checkout hash*.



* + Git checkout is a tool to observe the code AS IT WAS back in time. And maybe even play with it a little.
  + It should not be used to revert back and undo changes.
* GIT RESET
  + If you reset a branch, use *git reset head <head~3 / head hash>*



* + With git reset, you point the HEAD to a previous commit, deleting further commits from branch.
  + ATTENTION, because git reset moves the header to a previous commit, but it does not change your local code repository. You will still have all recent files (therefore, a bunch of these files will be unstagged relative to the HEADs new position).
* GIT REVERT
  + Git revert is used when you want to change back to previous commits, but without losing history.



* + Instead, the revert created another commit in the branch, with the reverted code.
* GIT RM and .gitignore
  + If you have committed files you don’t want, you can remove then from the repository with:
    - *git rm <file> --cached*
  + you can also create the *.gitignore* file and list, inside it, the files you don’t want to be staged and committed.