***Git***

*Git*

* *everything in git is object*
* *To initialize the git in local*

*→* git init project

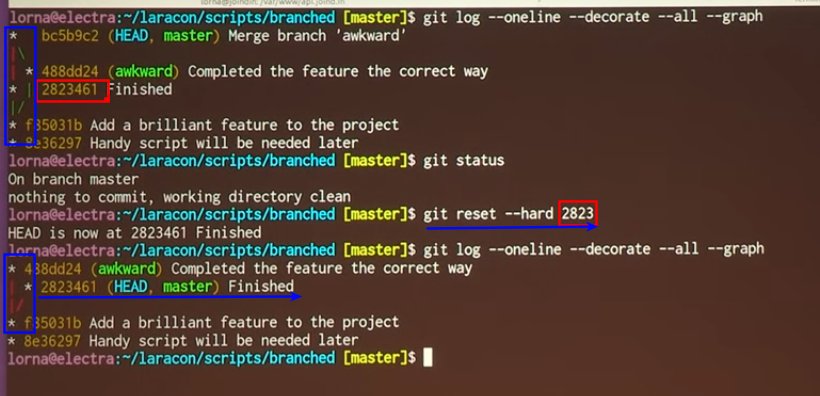
* *.git dir has → branches, config, description, HEAD, hooks, info, objects, refs*

**

* *create new branch*
  + *git checkout -b branchName*
  + *in .get/ref/heads # you see all your local branches*
* *status*
  + *git status*

**

* *compare*
  + *git diff*
* *add code to local/remote*
  + *local*
    - *git add file name*
    - *git add -p*
      * *it allows you to add each change you have made*
  + *Remote*
    - *git add remote “url”*
* *merge code*
  1. *git merge branchName*
  2. *if any conflicts* 
     1. *git merge - -abort*
     2. *resolve conflicts*
     3. *merge again*
  + *undo merge*
    - *git reset - -hard // will loose all committed code in working directory*

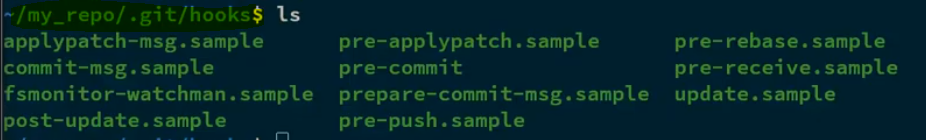
**

***Git Hooks***

*Git hooks allows you to run custom script whenever certain important events occur in the Git life-cycle such as committing, merging and pushing*

***Demo***

*Project folder 🡪 .get 🡪 hooks*

**

* *To execute hook, remove sample extension and make the file executable*
  + *Chmod +x pre-commit – will make file executable in linux*
* *Pre-executes before commit*
* *Post executes after commit*
* *To bypasses this script*
  + *–no-verify*

*Commands*

* *git init // to initialize local repo*
* *git checkout -b branchName // create branch with name branchName or switch to branch*
* *git status // to see the status of the file in local repo*
* *git diff // to check difference current directory*
* *git reset // will unstage files and you don’t loose any files*
* *git log - -online - -decorate - -all - -graph*
  + *- - online will compress the message*
  + *- - decorate will decorate with branchs, head, add lables*
  + *- - all mean show all in my repository*
* *git merge branchName // will merge the branch*
* *git merge -abort // exiting from merging*
* *git config - -local rerere.enabled true // to avoid same conflict in future merge*
  + *reuse recorded resolution*
* *git diff // shows only unstaged edited files & new files*
* *git diff - - staged // will show staged file/ file are going to next commit*
* *git diff HEAD // will the file which you committed in last commit*
* *git bisect*
* *"git config --list" // list of all users*
* ***Add remote repo***
  + *git remote add origin* [*https://github.com/batch.git*](https://github.com/batch.git) *(remote repository)*
  + *Then push the code target branch*
    - *git push origin Develop*
    - *git push origin feature\_branch*
* ***Branch***
* *list all branches*

*"git branch"*

*Note: Current working branch in listed as green color and \* as prefix*

* *Create new branch*

*"git branch master-1"*

* *Create new branch and checkout to that new branch*

*"git checkout -b brachName"*

* *Switch working branch*

*"git checkout branch name" (git checkout master-1)*

* *Delete Branch*

*first exit from branch*

*"git branch -d branch name"*

* ***Merge – Push - Pull***
* *git merge - update the code from one branch to another branch (branch2 -> branch1)*

*first check out to branch 1*

*"git merge branch2"*

* *git push - check in the code to central repo*

*“get push centralRepo localRepo”*

*“get push origin local-changes”*

* *git pull - get the updated files from central repo*

*login into target local repo*

* *“get pull Centeal Repo Path*
* ***Update the code from git (Pull)***
  + git pull origin Develop

***Assessment***

*why would you use a pre-receive hook in your remote repository?*

* *you wouldn’t, you would use it in the local repository*
* *to invoke a hook script when commits are pushed but before any references are*
* *to fire a script after updates are made to the remote repository*

*After checking your git status, you get the following output, which shows the file beta-notes.js in the commit but also unstaged, how can this situation occur?*

*Changes to be commited:*

*(use “git reset HEAD <file>….” to unstage)*

*modified: beta-notes.js*

*Changes not statged for commit:*

*(use “git add <file>...” to update what will be commited)*

*(use “git checkout --<file>...” to discard changes in working directory)*

*modified: beta-notes.js*

* *There were two copies of beta-notes.js, but one was deleted*
* *beta-notes.js was staged, then modified afterwards, creating two different versions of the file*
* *two copies of beta-notes.js were created, but only one is being tracked*
* *there are two tracked copied of beta-notes.js, but one was removed from the commit*

*what files is the following. gitignore programmed to leave out?*

* *All files with a .swift, .txt, or .metadata file extension, as well as the entire build directory*
* *only the build director*
* *all files in the build directory , as well as files ending with .txt or metadata*
* *only files with .swift and .txt extensions*

*how could you squash multiple commits together without using git merge – squash?*

* *Caching*
* *you can’t git merge is the only git command for that operation*
* *rebasing*
* *reflogging*
  + *git reflog doesn't traverse HEAD's ancestry at all. The reflog is an ordered list of the commits that HEAD has pointed to: it's undo history for your repo. The reflog isn't part of the repo itself (it's stored separately to the commits themselves) and isn't included in pushes, fetches or clones; it's purely local.*

*looking at the following commands, describe what is happening.*

*Git checkout feature-user-location*

*git cherry-pick {af5597c29467a96256a70787c9f4db}*

* *the commit is being tagged for release on the feature-user-location branch*
* *a commit is being copied from its original branch over to the feature-user-location branch*
* *the commit is being cherry picked as the new HEAD of the commit history*
* *a Commit is being copied from the feature-user-location branch to the master branch*

*Note: cherry-pick- Apply the changes introduced by some existing commits*