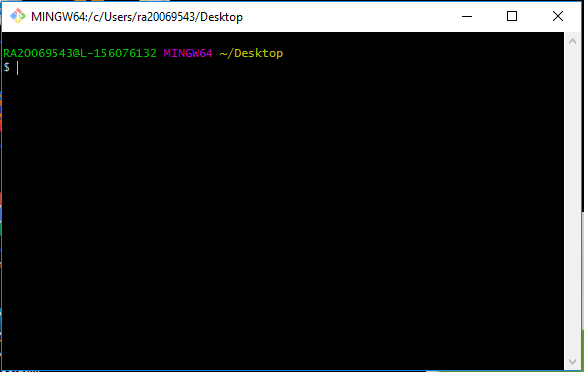
Git installation and configuration:

1. Git software installation document.



1. Configuration:

Project folder -> right click-> git bash here



git config --global user.name "user name"

git config --global user.email "email id"

1. For existing project: local repo to central repo

Project folder -> Right click-> git bash here->

"git init" in Master folder and “git remote add origin <repo url>

"git init" in Dev folder and git remote add origin <repo url>

“git init" in QA folder and git remote add origin <repo url>

"git status" (it will show untracked file in local)

"git add file name"(to track file)

"git add -A"(to track multiple files)

"git commit -m" "message" file name (commit single file with message)

"git commit -a -m" "message"(commit multiple file with message)

git branch QA"(branch name)" (to create a new branch).

git branch DEV"(branch name)" (to create a new branch).

git checkout QA (branch name). It will switch to QA branch.

git checkout DEV (branch name). It will switch to dev branch.

“git push origin master"

“git push origin qa"

“git push origin dev"

1. (Already we have the file in central repo means follow the below steps)

(Central repo to local repo)

git clone <url of central repo> It will download master branch data only.

Particular branch means -> git clone -b "branch name" "project url"

Do some changes in a file...

"git status" (it will show untracked file in local)

"git add"(to track file)

"git add -A"(to track multiple files)

"git commit -m" "message" file name (commit single file with message)

"git commit -a -m" "message"(commit multiple file with message)

“git pull origin master" (one guy working on one project skip this line)

(Multiple guys working on one project follow this line)

“git push origin master"

EX:

Pull -> work on it (project) Status -> add -> commit -> push.

(One guy working on one project follow the above line)

EX:

Pull -> work on it (project) Status -> add -> commit -> pull -> push.

(Multiple guys working on one project follow the above line)

1. Central repo to local repo(for running project)

"git pull origin master"(now it will download all the files from central to local repo)

Do some changes in a file...

"git status" (it will show untracked file in local)

"git add"(to track file)

"git add -A"(to track multiple files)

"git commit -m" "message" file name (commit single file with message)

"git commit -a -m" "message"(commit multiple file with message)

“git pull origin master" (one guy working on one project skip this line)

(Multiple guys working on one project follow this line)

“git push origin master"

EX:

Pull -> work on it (project) Status -> add -> commit -> push.

(One guy working on one project follow the above line)

EX:

Pull -> work on it (project) Status -> add -> commit -> pull -> push.

(Multiple guys working on one project follow the above line)

1. We need to check current branch details means use a command "git branch"(it will show the current branch details and all the branch names also)

We need to check current origin details use a command "git remote -v"(it will show fetch and push project name)

EX:

$ git remote -v

origin https://github.com/amraviraj/final-test.git (fetch)

origin https://github.com/amraviraj/final-test.git (push)

1. Collaborators concept: (Multiple guys working on one project follow this process)

* User1 and user2 need to config git in local system.
* Then user1 upload all the files to the central repo. (Already files are in central repo means use clone command)
* Repo name ->setting ->collaborators->search user2 name->add collaborator.
* User2 will get invitation mail, after that user2 need to approve that invitation.
* Now user1 date will copy to user2 central repo.
* If user2 user doing some changed means automatically it will reflect in a user1 central repo also.
* But this change won’t reflect in user1 local repo.

So……..

* Work on it (project) (user1) Status -> add -> commit -> pull -> push.

1. git branch -d branch name (To delete a branch)
2. Project repo->manage topics->release tab -> here we will take any particular branch final output data (source code) with a zip or tar format.
3. Git merge:

You are in master branch in bash -> git merge dev

Now dev branch data will merge with the master branch.

In gui we need to create a pull request in inside of the repo for git merge.

Inside of repo -> near to branch are you will find “new pull request” option->then select “base” is master and “compare” is dev->create pull request->conversation tab-> select merge pull request before that review your changes.

1. Git stash:

Bash-> git stash

It will remove last saved data in the local repo file.

Bash-> git stash pop

It will recover last deleted data in the local repo file

Bash -> git stash list

It will show a collection of stash. Like stash@(0): wip on master: \*\*\*\*\*\*\*\*\*\*

stash@(1): wip on master: \*\*\*\*\*\*\*\*\*\*

Bash-> git stash drop (it will remove last added stash file)

Bash ->git stash drop stash@(1) (It will remove the particular stash)

12: Git revert:

Bash -> git log (It will show the committed file details and commit id)

git log --oneline (Short form) or git log –oneline --all

git log --before="2/27/2019" (date). It will show particular date log files.

git revert head (Undo the last committed changes)

git revert “git log commit id” (Undo the particular committed changes)

13: Fork and pull request:

* User1 and user2 need to config git in local system.
* Then user1 upload all the files to the central repo. (Already files are in central repo means use clone command)
* Repo name ->setting ->collaborators->search user2 name->add collaborator.
* User2 will get invitation mail, after that user2 need to approve that invitation.
* Now user1 date will copy to user2 central repo.
* Then user2 need to take a fork for user1 repo.
* User1 will make some changes means it won’t reflect to user 2.( user2 will make some changes means it won’t reflect to user1)
* If user1 need to apply the own change to user2 means, user1 need to raise a pull request to user2. (If user2 need to apply the own change to user1 means, user2 need to raise a pull request to user1.)
* Once users accepted the request, as per request changes will reflect.

14: rebase