Git workflow in client system

Git is a distributed version control system**. Using Git, many developers can make changes to the same code base at the same time without running into accidents like overriding someone else's changes. Git will update only** difference **to a file.**

- Git is distributed version control system.
- We can have code in remote central server and local system also.
- Content stored in git is metadata.it stores the content in .git folder. the folders consists of all the tags, version history, branches etc..
- SVN stores files. They do not have a cloned repository.
- Git was created by **Linus Torvalds** in 2005 for development of the Linux kernel

Configure Git:

you need to configure Git so that it can understand who is pushing the code to remote repository as many people could be working on the project.

Important configurations are user's name and user's email.

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

git config --global user.email subbaraju.challa@violamoney.com

You can view all Git configurations using below commands.

$ git config --list

$ git config --list | grep user

user.email=subbaraju.challa@violamoney.com
user.name=Subbaraju Challa

Command usage:

git init >>> To make folder as git repository

git remote add origin **repo-url** >>> add remote repository

git pull >>> pull remote repository files to local repo

Above three steps can be combined into single step when you already have a remote repository**.** For that, use below command.

git clone **repo-url** >>> To clone remote repo

git add >>> To add files to index

git status >>> To check status of the files

git add -A >>> add all new, modified and deleted files (tracked+untracked) to staging area

git add . >>> Acts like git add -A but it will only pick up files from the current directory

Deference between git pull and git fetch:

When we do git pull it directly pulls the changes and new files form master (remote) branch to local branch and merger with local copy.

but when we do fetch it does same, but it will fetch to new branch not copied to current work flow
after this we need to do git merge otherwise it will not affect in local repo

** git pull = git fetch + git merge

Your local repository has three different virtual zones or areas viz:

- working area,
- staging area and
- commit area.

Working area is where you create new files, delete old files or make changes to already existing files.

Once you are done with these changes, you add these changes to staging area. Staging area sometimes also called as Index. Once you think you have done your job, staging area will contain one or more files which need to be committed.

Whenever you create a commit, Git will take changed code from staging area and make a commit which it then moved it to commit area. Unless you use git push command, these commits won't be sent to remote repository.

What are Commits?

A commit remembers changes in files. A commit has a unique SHA1 hash which is used to keep track of files changed in past. basically, it contains file change metadata, author of commit, timestamp of commit and previous commit hash.

Whenever you use git pull or git push, send these commits to remote repository. Git on remote repository server then merge these commits to it's repository.

Branching:

Creating a different folder like master

Master branch contains total code (main code)

>> if we don’t want to disturb or edit direct master branch (main code)

>> we can create branch. While creating branch data in master branch comes to new branch

>> then we can commit new changes to new branch and we can check it contain master branch data and new commit data.

Merging:

>>Combination of multiple branches

>>what and all changes have done in new branch needs to merge with master branch until merging the new data contains in new branch will not come to master.

Untracked files:

Untracked files are files that are not present in remote repository, hence Git cannot track them with remote repository.

.ignorefile:

This file contains files that should be ignored by Git while making commits. Add file and folders to .ignore file

 # vi .gitignore
/node_modules
/public/hot
/public/storage
/storage/*.key
/vendor

Create a new repository (to clone remote repository)

```
git clone http://gitlab.violamoney.com/root/vwallet-apis-test.git
cd vwallet-apis-test
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Existing folder or code

```
cd existing_folder
git init
git remote add origin  http://gitlab.violamoney.com/root/vwallet-apis-test.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Existing Git repository

```
cd existing_repo
git remote rename origin old-origin
```

git remote add origin http://gitlab.violamoney.com/root/vwallet-apis-test.git

git push -u origin --all

git push -u origin –tags

If you added some files in staging area by mistake, then you can use following command to unstage them.

git reset test

You can use git reset command to unstage all files.

git rest

To see all the commits in the repository (made by all the developers), use following command.

git log

Reset repository to a revision:

It is possible that you made a commit which you didn't intend to make. So, what about that?

HEAD is pointer to the last commit in Git history.

Exmaple:

We added file f1

git add -A

git commit -m "testing commit "

git log
commit ea70c22b58a6cb57fa901090cb8755fb3f0a716d (HEAD -> newcode)
Author: Subbaraju Challa <subbaraju.challa@violamoney.com>

Date: Mon Sep 24 13:23:20 2018 +0530

    testing rest

commit b14afcb325c5ef93c872081b9eb23d3abd3d7740 (newcode/newcode)
Author: Subbaraju Challa <subbaraju.challa@violamoney.com>
Date: Sun Sep 23 16:46:23 2018 +0530

    Signed-off-by: Subbaraju Challa subbaraju.challa@violamoney.com

To change message in previous commit, use below command.

git commit --amend -m "Initial Commit"

If we forget to add come comments or need add some other file. we can reset it back with following command

git reset b14afc
Unstaged changes after reset:
M	.gitignore
M	f1.txt

will remove all commits after commit b14afc and will bring all changed code after that in staging area. You don't need to use full hash of a commit. All commits after this commit is removed from git history.

git status

#git reset --hard b14afc

will remove all commits after commit **b14afc** and destroy all changed code after that. This will also remove changed file in working or staging area. Hence git reset --hard HEAD is also used to get rid of all the changes whether it is inside working area or staging area.

One important thing to remember is that all untracked files (*newly created files*) will not be removed.

Git branching:

Git history is a series of commits linked together forming a chain. A branch is nothing but that chain with a name.

When we add new commit, it gets pushed to the top of that chain. The top commit is now **HEAD**

HEAD is just a pointer to last commit in currently checked out branch (current branch we are in)

Note:

If we switched to new branch the latest commit of current branch will be the head.

Below command will show local branches and the one which contains * is current branch

```
$ git branch
  master
* newbranch
  newcode
```

To create a branch "dev", you need to use following command.

git branch dev

To enter inside dev

git checkout dev

Above two steps can be carried out at once using git checkout -b dev command which will create and checkout branch at the same time.

To see branch history:

git log

To see all local and remote branches:

$ git branch -a
  Master
  dev
* newbranch
  newcode
  remotes/origin/master
  remotes/origin/newcode

To create new branch:

git branch testing

To change branch:

git checkout testing

Push local branch (newly created branch) to remote (git server)

git push -u --set-upstream origin testing

git branch
  master
  newbranch
  newcode
  raja
  revert-c9b5fb1f
* testing

To check if any branches ever merged with current branch which is master, you can use following command.

git branch --merged

To merge dev into master

git merge dev

To delete dev branch

$ git branch -d dev
Deleted branch dev (was c9b5fb1).

To delete remote dev branch

git push --delete origin dev

To bring commit from other branch to current:

Exmaple:

This will be help full when we unfortunately commit to different branch and need to commit to current branch

Unfortunately we are in master branch and add some files in master branch and committed to master branch but you want to commit in testing branch.

Then we take head commit SHA number in master >> then check out to testing >>> give below command “cherry-pick” and commit number.

git cherry-pick 07762b

Then commit will assign to current branch
git log

Git revert:

git revert command which creates a new commit by reverting all the changes in a commit. Since, we want to revert all change, we need to use below command.

git revert 07762b

Diff:

git diff 01f8a6a 4c045d9

Git tags:

git tag v1.0

git tag
v1.0

subbu@DESKTOP-4LUV3QN MINGW64 ~/Music/vwallet (master)
git push origin v1.0