**GIT Commands**

**git config --global user.name “Pradeep”** : to create user for git in your system

**git config --global user.email** [**pradeep@gmail.com**](mailto:pradeep@gmail.com) : to create user email git git in your system

**git --version : to check the version og git**

**git config --details: to check the git configuration details.**

**git init <repository>** : to create repository

**git init <repository>.git --bare** : to create bare repository

**git clone https://github.com/Pradipta-P/Novel.git <repository>** : to clone repository form remote repository

**touch 10.java** : to create a file named as 10.java

**git status** : to check the status of current repository

**git add <file>** : to staging the specified file

**git add .** : to staging the all files in repo

**git commit –m “message”** : to commit the changes

**git commit --amend** : to change the last commit message.

Note: if you change the message using --amend the commit id will also changed. So you have push the commit id to central repo using **git push --force** or **git push origin master --force**. Because already you have pushed the same changes so you need to forcefully push the new commit.

**git log** : to see the all commitID and its details

**git log --oneline** : to see the all commit in shot(oneline)

**git log --oneline –n** : to see the latest n number of commit in oneline

**git branch <branch\_name>** : to create a branch

**git branch** : to lsit all available branch

**git checkout <branch>** : to switch to the specified branch

**git merge <source\_branch> <destination\_branch>** : to merge data from source to destination branch

**git merge <source\_branch>** : to merge data from source branch to current branch

**git cherry-pick <commitId>** : to merge a specific commit

**git push --set-upstream origin <branch\_name>** : to make specified branch to a upstream branch in central repository

**git push –u –all** : to push everything from local repo to central repo including branches

**git pull <central repo> <branch\_name>** : to pull updates for central repo and particular branch.

Exa: **git pull** [**https://github.com/Pradipta-P/DevOps.git**](https://github.com/Pradipta-P/DevOps.git) **Development**

**git branch –d <branch\_name>** : to delete the branch

**git branch –D <branch\_name>** : to delete the unmerged branch

**git push origin --delete <branch\_name>** : to delete the branch from central repo

**git push --delete origin <branch\_name>** : to delete the branch from central repo

**git stash** : to create a backup of current state(uncommitted changes) and back to previous stage(commited)

**git stash list** : to list the all available stash

**git stash apply stash@{n}** : to go back to the previous stage(before the stash). n= Arry number 0=latest arry, highest number= oldest stash

**git stash clear** : to clear the stash

**git reset --soft** : to reset the change in repo(temp commit)(before the commit)

**git reset --mixed/ git reset** : to reset the changes in staging area & repository (before commit)

**git reset --hard** : to reset the change in all three satge(working dir, staging area & repository) before the commit.

**Git revert <previous commitid>** : to revert the change to previous commit(latestcommit) after the commit.

**Git checkout <commitid>** : to move to particular commit(moving the head pointer to specified commit)

**Git checkout <branch\_name>** : to move to the latest commit (moving the head pointer to latest commit)

**Git tag –a <parameter> -m <message> <commitid>** : to adda tag to particular commitid

Exa: git tag –a Relaese3.2.1 –m “1st release” 7a60003b

**Git tag** : to list the all tags

**Git show <tag>** : to see the details of tag(commitid)

**Git push --tags** : to push the tags to central repository.

**Git pull --tags** : to pull the tags from central repository

**Git tag –d <tag>** : to delete the specified tag

**Git push --delete origin <tag>** : to delete tag from central repository

**git push origin --delete <tag>:** to delete tag from central repository

Exa: git push --delete origin Release3.2.1

**Git fetch --all** : to fetcing all from central to local repo.(including branch)

**Git clean –n** : to see the all untracked file which are going to delete by **git clean** cmd

**Git clean –f** : forcefully delete all untracked files.

**.gitignore** : It’s a file basically used to ignore to push some specific files and folders to remote repo.

We have to mention the specific files and folders name inside the **.gitignore** (before staging)file so it will ignore to push the specific files and folders into central repo, even those files and folders are available inside the local repo.

**Git Error**

**remote: error: GH001: Large files detected. You may want to try Git Large File Storage**

**Cause(Why)**

In GitHub, By default you can upload the files which less than 100MB. We are getting above error because one or more files in our local repo are more than 100MB in size.

**Solution**

We needs to use **git-lfs** (large file storage) to upload the large files(size more than **100**MB).

steps:

1st install the git-lfs(**apt-get install git-lfs**) – go to the your local repo and track the large files(**git lfs track ‘<file\_name>**) - then use **git add .** - **git commit** – **git push.**

**Exa:** git lfs track ‘\*.java’

**Note: Your central repo(GitHub account) should be enable for uploading the large files.**