Git Commands

Introduction

When you install Git-bash, the first thing you should be doing is setting up your user details as follows only one time.

```
#git config --global user.name "rama"
#git config --global user.email "rama@gmail.com"
```

Checking for settings

```
#git config --list
```

You can also check what Git thinks a specific key's value is by typing git config <key>:

```
#git config user.name
```

Task 1: Create the git local repository in local machine (Laptop/Desktop), add one file (DBConnect.java) and update that file, create the github remote repository (https://github.com) and move the local code to github repository.

Go to the directory where you want to create the git repository.

```
# cd ~/Desktop
# mkdir git-practice-commands
#cd git-practice-commands
#git init : Create a local Git empty repository.
Initialized empty Git repository in /Users/Rama/git/git-practice-commands/.git/
#git status : Gives the status of your untracked files.
#touch DBConnect.java
#git status
#vim DBConnect.java
#git add DBConnect.java: Add the files(here DBConnect.java) into your staging area.
#git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:  DBConnect.java

#git status

On branch master
```

nothing to commit, working tree clean

- Open the file (DbConnect.java) and update with some text.

#vim DBConnect.java

#git commit -a -m "Updated DBConnect.java file" : If we use –a along with commit command, no need to execute git add command.

```
[master 7f795a7] Updated DbConnect.java file
 1 file changed, 1 insertion(+)
```

- Create the repository in github as follows.

 Login into github (http://github.com)
On right side top corner click on **"+"** symbol and click on **"New repository"** and give the Repository name and click on Create repository.

#git remote add origin git@github.com:devopsblr/test.git : Adding the URL for the remote repository where your local repository code will be pushed.
# git remote –v :
#git remote show origin : It will give the information on a particular remote (here origin is the remote name)
# git remote remove origin : It will remove the remote origins.

#git push origin master : Push the changes in your local repository to GitHub remote repository. (Here push is the git command , origin is the remote name and master is the branch name)

```
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 479 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To git@github.com:devopstrainingblr/test.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

#git status

```
 On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

#git log : It will give all commit ids.
#git log -2 : It will display only 2 commit ids.

#git show --pretty="" --name-only << Commit ID >> : It will display all the files which are commited in that particular commit.

#git clean -n : It will preview the changes.

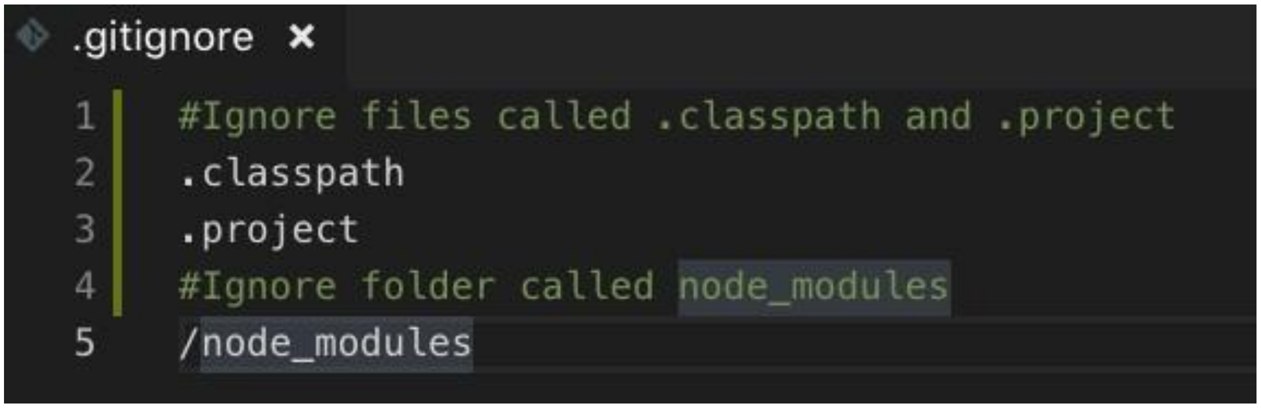
#git clean -f : If we want to remove new files from working area.

#git reset <<File Name>> : To untrack the tracked files (revert back to working area from staging area.).

#git revert <<Commit ID>> : It will revert the changes committed in that particular commit id from local repo.
#git push origin master -f: It will revert the changes from remote repo.

Git Ignore file (.gitignore)

Some times we don't want to commit the files, which are generated by IDE like .project and .classpath files or some node module folders like node_module folder into a git repository.
To ignore these files and folders to commit we will create one file called .gitignore and we will keep the file names or directory names which we don't want to commit as follows.



```
#Ignore files called .classpath and .project
.classpath
.project
#Ignore folder called node_modules
/node_modules
```

https://www.atlassian.com/git/tutorials/saving-changes/gitignore

Branches

#git branch : It gives the branch names in current repository.
#git branch bugfix : It will create the bugfix branch in local git repository.

#git branch -v: It will display all the branches in your repo, and also tell you what branch you're currently in.
bugfix 87226db initial commit
* master 87226db initial commit
Note: Here * indicate currently in use branch.
# git checkout bugfix : Switch to bugfix branch.
Switched to branch 'bugfix'

Update the Bhaskar.txt like change 2 – bugfix branch
# git add . : Add one or more files to staging
# git commit -m "bugfix commit"

# git checkout master : Switch to master branch.
Switched to branch 'master'

Updat the Bhaskar.txt like change 3 – master branch
# git add .
# git commit -m "master commit"

# git checkout bugfix : Switch to bugfix branch.
Switched to branch 'bugfix'

Check the file and see the contents in file.

#git checkout master

#git diff master bugfix

#git merge bugfix
Fix the conflicts
#git add .
#git commit –m "merging"
#git push origin --all : Push all branches code to your remote repository.

#git branch -d bugfix: Deletes the bugfix branch in local repo.
#git push origin : bugfix (OR) git push origin --delete bugfix: It will delete a remote branch in the repository.

Tags

git tag : It will displays the tags.
git tag <<Tag Name>> : It will create the tag.
git push origin tag <<Tag Name>> : It will push the tag to remote repo.
git push origin --tags: It will push all the tags to remote repo.

Note: Tags are not automatically pushed when you push a branch or use the --all option. The --tags flag sends all of your local tags to the remote repository.

git tag -d <<Tag Name>> : It will delete the tag.

git stash: git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on. Stashing is handy if you need to quickly switch context and work on something else, but you're mid-way through a code change and aren't quite ready to commit.

git stash (OR) git stash save "Updated some code" :
git stash list :

git stash show : This command shows the summary of the stash diffs.
The above command considers only the latest stash.

git stash show -p : It will give you the detailed list of differences.

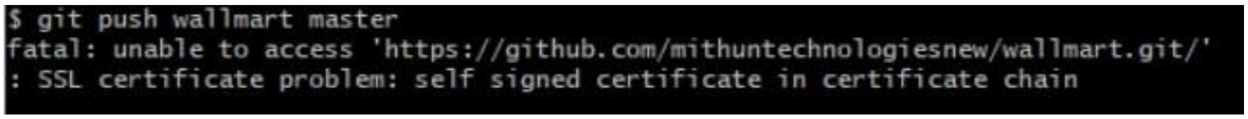
git stash show stash@{1}:
git stash apply stash@{0}:

git stash drop stash@{0} :

git stash list
vim mithun.txt
git stash
git stash list
git stash pop: It apply the latest stash and then immediately
drop it from your stack.
git stash list

git stash pop stash@{1} : It apply the particular stash and then immediately
drop it from your stack.
----
git cherry-pick: Cherry picking in git means to choose a commit from one branch and apply it onto another.

git log
git branch
git checkout master
cat mithun.txt
git cherry-pick <<CID>
cat mithun.txt



```
$ git push wallmart master
fatal: unable to access 'https://github.com/mithuntechnologiesnew/wallmart.git/'
: SSL certificate problem: self signed certificate in certificate chain
```

When you get the above error while committing the code from local repository to remote repository execute the following command in git bash.

git config --global http.sslVerify false

Steps for Code Checkout into local from Remote Repository
==

Go to the directory where we need to commit the code/checkout the code
cd C:\MithunTecnologies\JavaWorkspace\MTWorkSpace

Get the code from Git Repository as follows.

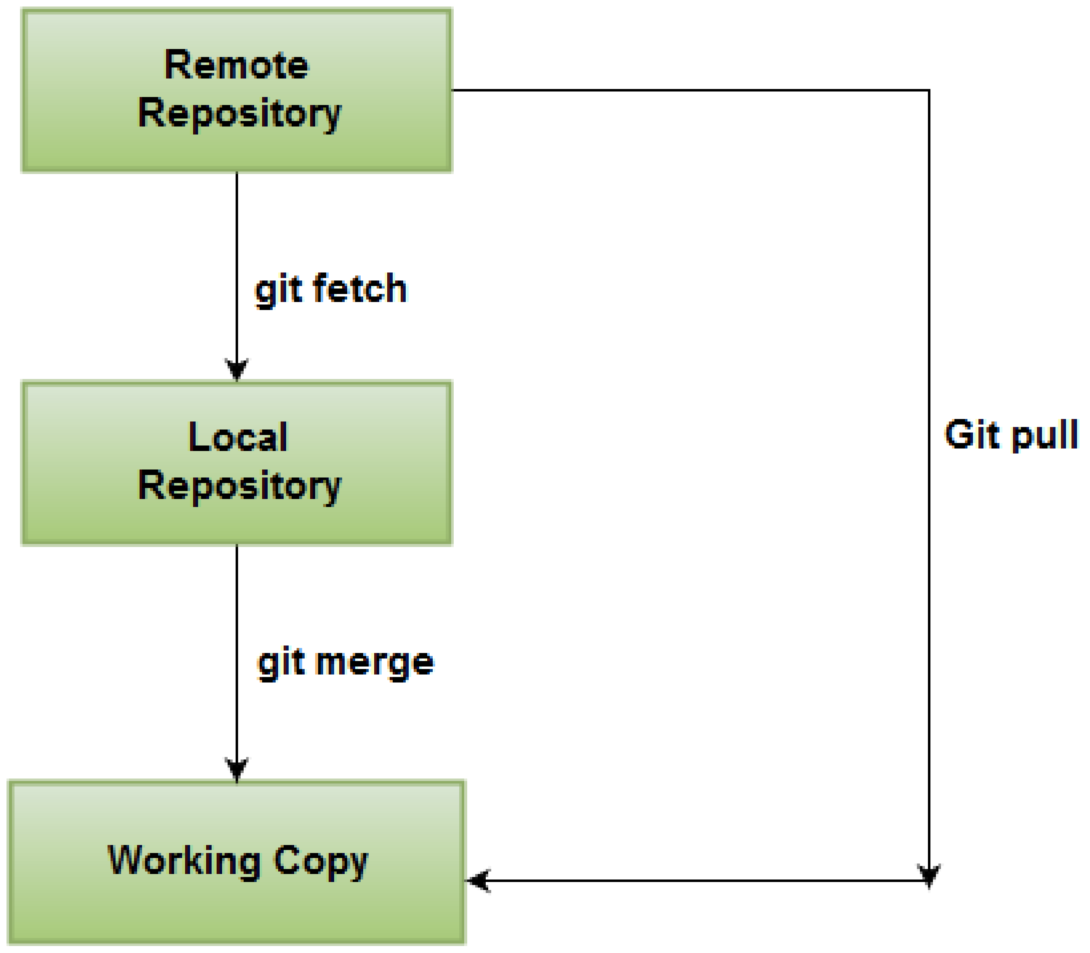
git clone <<GitHub URL>>

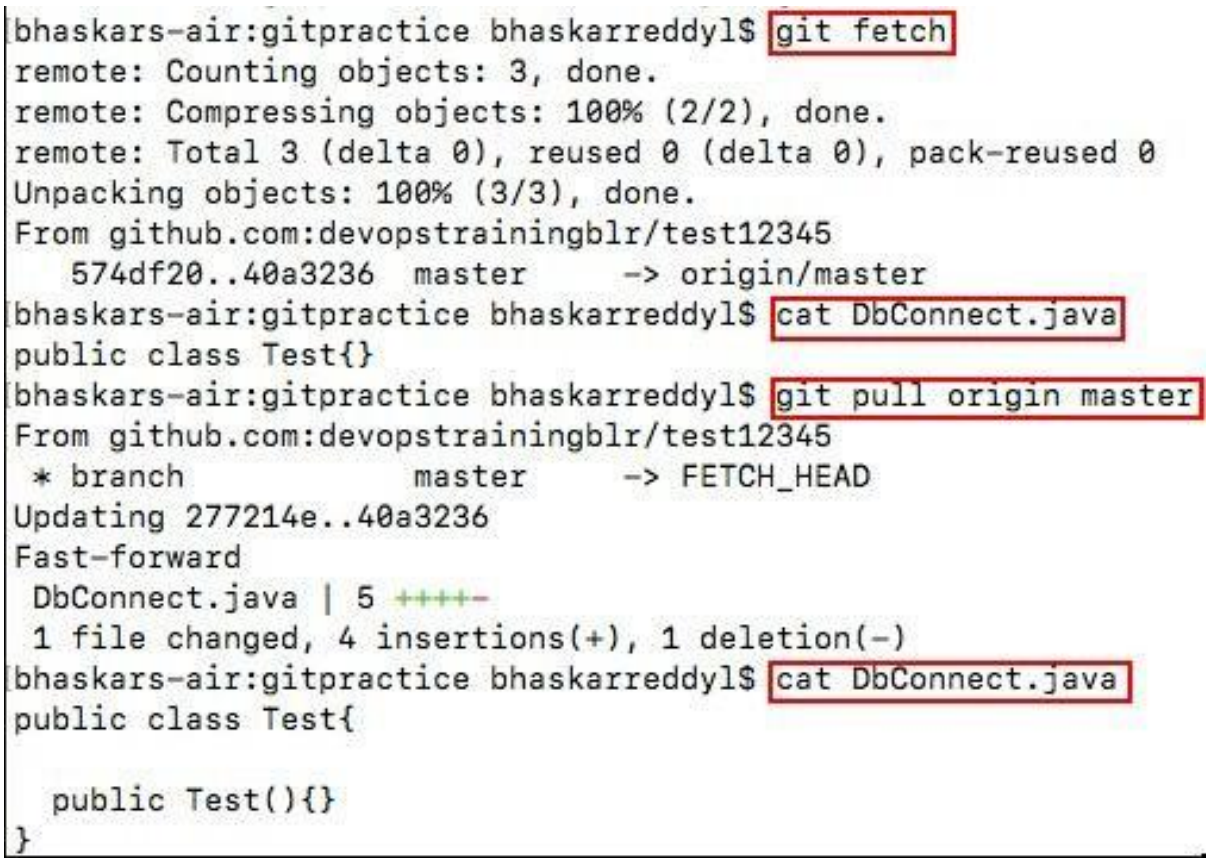
What is the difference between git fetch and get pull?

Ans) git fetch : It will get the update from git remote repo and will update your local repo. But it will not merge with Local working copy.

git pull : It will get the update from git remote repo and will update your local repo as well it will merge with Local working copy also.

So **git pull = git fetch + git merge origin/master**



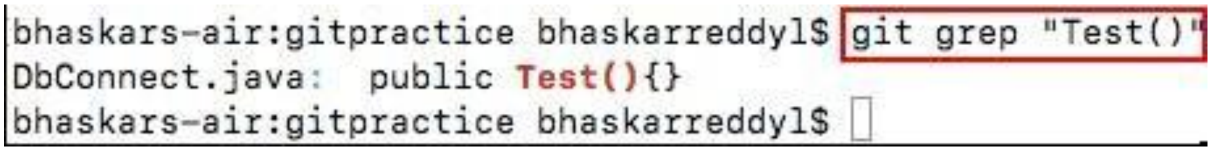


```
bhaskars-air:gitpractice bhaskarreddyl$ git fetch
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:devopstrainingblr/test12345
   574df20..40a3236  master     -> origin/master
bhaskars-air:gitpractice bhaskarreddyl$ cat DbConnect.java
public class Test{}
bhaskars-air:gitpractice bhaskarreddyl$ git pull origin master
From github.com:devopstrainingblr/test12345
 * branch            master     -> FETCH_HEAD
Updating 277214e..40a3236
Fast-forward
 DbConnect.java | 5 ++++-
 1 file changed, 4 insertions(+), 1 deletion(-)
bhaskars-air:gitpractice bhaskarreddyl$ cat DbConnect.java
public class Test{

  public Test(){}
}
```

#git commit --amend -m "an updated commit message" : Change most recent Git commit message.

# git grep "Test()" : Search the working directory for Test()



```
bhaskars-air:gitpractice bhaskarreddyl$ git grep "Test()"
DbConnect.java:   public Test(){}
bhaskars-air:gitpractice bhaskarreddyl$ 
```

#git checkout <<Branch name>> : This will switch the branch.

Ex: git checkout development

#git checkout -b <<Branch name>> : It will create the branch from currently using branch and will switch to new branch.

#git checkout -b feature master: It will create a branch called feature from master branch and will switch to new branch.

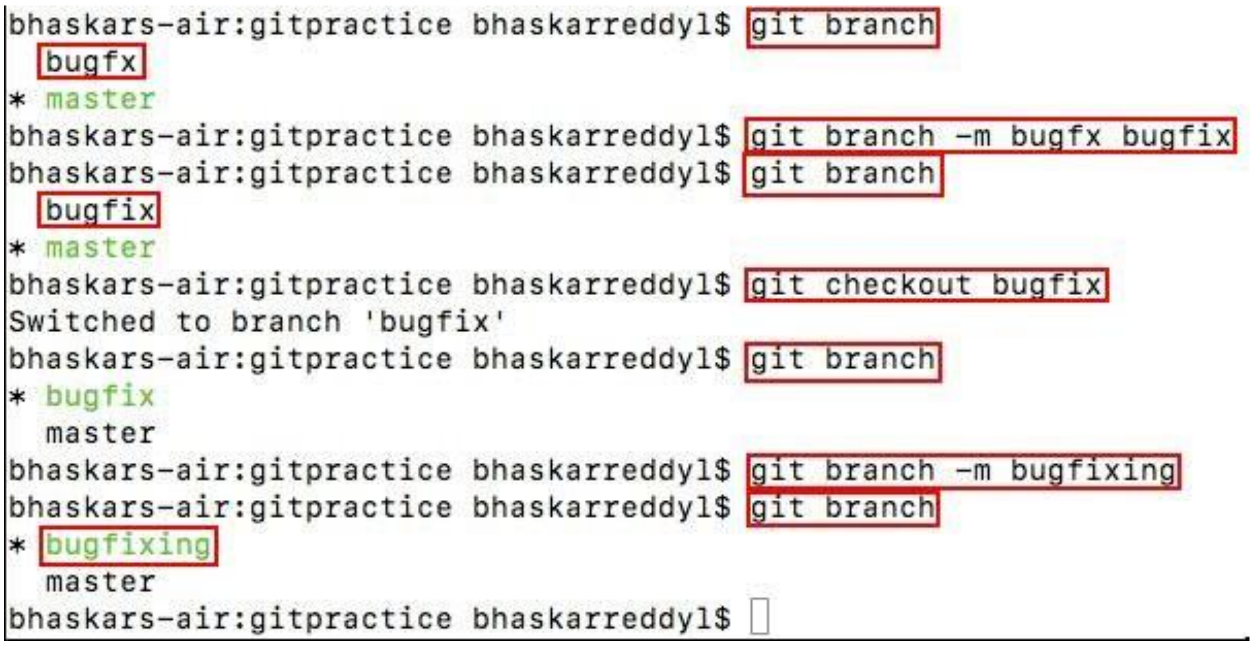
#git checkout -b release-aadhar master: It will create a branch called relese-aadhar from master branch and will switch to new branch.

How to Rename a git branch name?

Ans) git branch -m <oldname> <newname>

Or, if you are already in the branch:

git branch -m <newname>



```
bhaskars-air:gitpractice bhaskarreddyl$ git branch
  bugfx
* master
bhaskars-air:gitpractice bhaskarreddyl$ git branch -m bugfx bugfix
bhaskars-air:gitpractice bhaskarreddyl$ git branch
  bugfix
* master
bhaskars-air:gitpractice bhaskarreddyl$ git checkout bugfix
Switched to branch 'bugfix'
bhaskars-air:gitpractice bhaskarreddyl$ git branch
* bugfix
  master
bhaskars-air:gitpractice bhaskarreddyl$ git branch -m bugfixing
bhaskars-air:gitpractice bhaskarreddyl$ git branch
* bugfixing
  master
bhaskars-air:gitpractice bhaskarreddyl$ 
```

git branch -a : It will display all the remote and local branches.

git branch -r : It will display all the remote branches.

git clone <<Git URL>> : To get the code from repository into your local machine.

git log : It will display the commit history.

git log <FileName>: It will display the commits related to the specified file.