



# Git

Friday, 06.11.2015

---

## Author

Manish Prakash

## Agenda

This document describes different important git commands and how to use git from command line effectively.

## Basic Concepts

1. What is Git?
    - a. Git is a version control system and code sharing system.
    - b. Git maintains different versions of your code and you can anytime see work done in the past.
    - c. Git makes it easy for multiple people to work on a single project
  2. REPOSITORY
    - a. This is place where the entire git system is setup, where you can view all your files etc. e.g github.com bitbucket.com gitlab.com
  3. CLONE
    - a. Cloning in git means take all files from git repository and setting it up in your system
  4. COMMIT
    - a. This means to put all your local system to local git. Basically there are two parts in git one is server and other is your system. What commit does is move files to your local git and make them ready to transfer it to server.
  5. PUSH
    - a. This uploads all your local commits to git server
  6. PULL
    - a. This download all code from git server to your local system
-

## Setting Up Project First Time

Use this method when a new git repository is created i.e now commit have been done and it's fresh.

1. `mkdir project` *//this will create a new project folder. not required if you already have folder created*
2. `cd project` *//go into the folder*
3. `git init` *// initialized empty git repo*
4. `git remote add origin {{git_url}}` *//add git url*

## Setting Up Project With Existing Git

This method is used for an exist git repository which already has files committed to it.

1. `mkdir project` *//this will create a new project folder. not required if you already have folder created*
2. `cd project` *//go into the folder*
3. `git init` *// initialized empty git repo*
4. `git remote add origin {{git_url}}` *//add git url*

## Commit Your Code

`git add -A && git commit`

This will commit of your local files to local git only.

This will open a text editor as well where you need to write comments . Write proper comment describing what work has been done. Comment is very important.

then write “:wq” to save and close the editor

## Pull Code

`git pull origin master`

This will take files from git server and merge with your local git files. It very important commit your files before doing this.

## Push Code

`git push origin master`

This will push your local committed files to git server.

## How To Sync Your Work

When your done working and need to sync your work to git. These are the steps to do

1. Commit
2. Pull
3. Push

## Branch

This an important concept in git. Branch is like a separate tree in which you can start working and then later merge to the main tree.

<https://git-scm.com/book/en/v1/Git-Branching-What-a-Branch-Is>


When you first time create a git repository a default branch is automatically called "master"

So if you look at previous command discussed "`git pull origin master`" here "master" was a branch.

Next important question is when is a branch used, suppose your current working on master branch. Master is considered the main branch always and it should have stable working code. Now you need to implement a feature which will take few days to complete. If you start this new feature on master branch itself, when you commit your changes to master the master branch becomes unstable. because your feature is not complete, but still you need to commit your code daily. This is where branch comes into play.

When a new feature comes you should ideally create a new branch and start working it. This way when you commit your code it gets committed only to your branch, and the master branch doest get unstable.

## Creating New Branch



```
git checkout -b "branch_name"
```

This command will create a new branch.

## Push, Pull To Your Branch

```
git push origin "branch_name"
```

```
git pull origin "branch_name"
```

## Merge Current Branch To Master

```
git merge
```

## Check Your Current Branch

```
git branch
```

this will show all branches in your project.

\* represent your current working branch

## Pull Request

Pull request is an important concept in git. This is one of the most used features and is very useful, so knowing this properly is important.

Pull Request, is generated after you create a branch. The concept of pull request is simple, let's assume you're working a project with multiple people. You create a new branch to start your work and after completing your entire work on that branch you want to merge it into master. At this point, either you can simple merge to master or create a pull request to merge to master.

The advantage of pull request is that, you get to see all the files which have which you have been change from master in your branch, and you share pull request with your with some other person for approval as well. So the other person is able to look at your code, comment on it if any issues and merge it.

This works very well when you have an manager to approve your code. This way after you done code you send a pull request to your manager, who can review your code, comment on it and merge to master after all is done.

Pull request cannot be created from command line, rather it needs to be created from websites where you git is hosted like github, bitbucket, gitlab etc

## Checking Git Status

Run the command “git status” in your current project to see full git status

## Reverting Git Files

Support you accidentally deleted some file, or your code got lost or you just need to get the file as it on git server.

```
git checkout _path_of_file
```

## Set Your Git User

git username is also very important. it the name which shows up in git logs on website github.com etc

```
git config user.name "your_name"
```

```
git config user.email "your_email"
```

also to your username use command “git config user.name”

## Git Ignore

git ignore is also a very important concept in git. It is required to mark few files and folders as ignored and hence they are not uploaded on git. So any temporary files which you don't want going on git, or files which change very often or are system dependent should be added to git ignore.

To add files to gitignore simple create a new file called .gitignore in the base folder or any folder and put the file path relative to the .gitignore folder

e.g

```
gedit .gitignore
```

and write

temp/ to ignore file

***As best practice in magento var/cache should be in ignore, in angularjs bower\_components/ should be in gitignore, and mobile platforms/ should be in gitignore***

Important Note:

If you have already commit a folder and later you want to put it in gitignore the process is a bit different, follow below link in addition to above

<http://stackoverflow.com/a/1139797>

## Git Shortcuts

1. Git Saving Password
  - a. Whenever you do pull/push git always asks you for password.
  - b. To save password, so that it doesn't ask again again. Go to base folder. open file .git/config using gedit .git/config
  - c. There you will see your git url like  
[https://manishexec@bitbucket.org/manishexec/pricegenie\\_andriod.git](https://manishexec@bitbucket.org/manishexec/pricegenie_andriod.git)  
 change this to  
[https://manishexec:your\\_passwr@bitbucket.org/manishexec/pricegenie\\_andriod.git](https://manishexec:your_passwr@bitbucket.org/manishexec/pricegenie_andriod.git)
  - d. Above we added password to git url itself, so git wont ask for password again.
  - e. But make sure passowrd doesnt have : or @ or else it wont work
2. Git Fast Commit
  - a. use command git add -A && git commit -m "Your Commit Here"
  - b. Basically with -m you can write comment directly in command itself.

## Git Resolving Conflict

When two people are working on a same project through git, it can lead to code conflicts.

Conflict happens when two people are working on same file and git doesnt know how to resolve it. Plus there are more reasons as well, basically when git is not able to merge files it gives conflict.

Usually when a conflict happens git will show in terminal list of files in which conflict happens.

Open that file and search for the word HEAD.

You will see something like

## HEAD

code\_here1

=====>

code\_here2

13dfdfs32432ffsdf =>>>>>>>>>>>>>>>

it will be something like.

HEAD code is basically the code from git server and below it is your local code.

You need to manually remove this lines from file and also put the correct code in file.

commit and sync again.

for more complex conflicts, you can google.