# Git

Git Tutorial
http://www.youtube.com/watch?v=ZDR433b0HJY
43 mins
http://www.vogella.com/articles/Git/article.html
Common git commands

https://confluence.atlassian.com/display/STASH/Basic+Git+commands

# git config --global user.name "Your Name"

# git config --global user.email you@example.com

# git init

# git status

# git log

# git add *

# git commit -m ‘comments’

# git commit -a -m ‘comments’

# git commit -am ‘comments’

Check out particular branch
# git checkout branchName

Get current branch name
# git branch

Download a branch
# git clone https://github.com/stariqch/sheraz.git

To push changes to server
# git push

To update
# git pull

---
**Git Version**

$ git --version

**Install Latest version of git on Linux**
# sudo add-apt-repository ppa:git-core/ppa
# sudo apt-get update
# sudo apt-get install git

**Editing git config files - system, global, local**
$ git config --system --edit
$ git config --global --edit
$ git config --local --edit

**Set Global System variables**
$ git config --global user.name "Sheraz"
$ git config --global user.email sheraz@office.com
$ git config --global core.editor "vim"

**To view all global settings**
$ git config --list

**Global config file**
*You can edit this file in any text editor*
~/.gitconfig
*Or have git open it by giving it this command*
$ git config --global --edit

**Initialize local directory/Initialize git repository**
$ cd my_project_dir
$ git init

**To ignore files from git**
In the root of the project create a file .gitignore
In the file list all the *.file_types or directories/ that you need to ignore.
To get commonly ignored file types for different types of projects you can get it from:
https://github.com/github/gitignore

**To add new and existing modified files to git repo**
$ git add .
*Dot means all files. You can also specify individual filenames by using file name instead of a dot.*
*add means that new or existing modified files are ready to be committed*

**To stage files**
$ git stage .

find out what is the difference between add and stage

**Current status of all the files that are in git repo**

$ git status

**Commit file**
$ git commit
*This will open editor(vim) to give comments*
*Before committing we have to add or stage files*

**Single command to commit and give comments**
$ git commit -m "My Comments"

**Single command to add and commit files, and give comments**
$ git commit -a -m "My Comments"

**Unstage/untrack current file changes**
$ git reset

**Delete File**
$ git rm file_name.txt

**Delete File force**
$ git rm -f file_name.txt
*If a file is modified and we try to delete it the git gives and error. So we can use rm -f to force delete*

**Delete File cached**
$ git rm --cached file_name.txt
*This will delete the file from git repo but leave it in the file system. It would be like the file is a new file.*

**Rename File**
$ git mv file_name1.txt file_name2.txt

**View Committed logs**
$ git log
$ git log --stat
$ git log --pretty=oneline
$ git log --pretty=format:"%h : %an : %ar : %s"
$ get log -2
*last 2 commits*

$ git log -p -2
*last 2 commits along with diffs*

$ git log --since=1.week
*changes in last week*

$ git log --since='2015-12-31'

*Changes since a date*

$ git log --before='2015-12-31'
*Changes before a date*

$ git log --author='Sheraz'
*Changes by an author*

$ git log --author='Sheraz' --since='2015-12-31' --pretty=format:"%h : %an : %ar : %s"
Combining different log criterias

**Fix last commit comment/message**
$ git commit --amend

**Add remote to existing project**
$ git remote add origin https://bitbucket.org/sherazc/gitprac

**Push/upload local changes to remote repo**
$ git push origin master
*"origin" is URL alias and "master" is the branch name*

**Start Fresh with remote repository and local project**
*If you just created remote repository e.g. "gitprac" and you want to link your new empty directory*

$ mkdir /path/to/your/project
$ cd /path/to/your/project
$ git init
$ git remote add origin https://bitbucket.org/sherazc/gitprac.git
*"origin" is alias to the URL. In push commands you can use URL or alias*

$ echo "Sheraz Chaudhry" >> contributors.txt
$ git add contributors.txt
$ git commit -m 'Initial commit with contributors'
$ git push -u origin master
*"master" is the default first branch you get with git*

**Use Existing code to upload to a new branch**
$ cd /path/to/my/repo
$ git remote add origin https://bitbucket.org/sherazc/gitprac.git
$ git push -u origin --all # pushes up the repo and it's refs for the first time
$ git push -u origin --tags # pushes up any tags

**Download existing remote project_branch**
*This method only fetches and checkout one remote branch. Once fetch is called then you can go offline until you need to push changes*

$ mkdir project_branch
$ cd project_branch
$ git init
$ git remote add bb https://bitbucket.org/sherazc/gitprac.git
$ git fetch bb project_branch
$ git checkout -b project_branch bb/project_branch
*This last command creates new local branch and links it to remote branch*

**Download existing remote project_branch (clone method)**
*Clone fetches(download all branches/files into git repo) all remote branches*
$ git clone https://bitbucket.org/sherazc/gitprac.git
$ cd gitprac
$ git checkout -b project_branch origin/project_branch

**Rename remote URL**
$ git remote rename origin myname

**Remove remote URL**
$ git remote remove origin

**View remote URL, and other details**
$ git remote -v
$ git remote show
$ git remote show origin
*shows remote branches*

**Check for remote changes and merge them**
*Update local git's remote information*
$ git fetch origin
*or update only one branch information*
$ git fetch origin master

*View/diff all the changes*
$ git diff origin/master

*Accept all changes*
$ git merge origin/master

**Check local and remote differences**
*In the commands below you can use "log" or "diff" and their switches. ".." is used as range. e.g. from_branch..to_branch.*
*In commands below you can switch from and to branches. e.g. from remote to local "origin/master..master" e.g. from local to remote "master..origin/master"*
*Fetch remote branch before giving commands below*

$ git log origin/master..master
*View commited changes need to be pushed*

$ git log origin/master..master
*View remote changes need to be pulled*

**Branches**
*NOTE: You can not create a branch until you commit file*

**View Current Branch**
$ git branch
*Or*
$ git branch -v

**View All Branches (even hidden)**
$ git branch -a
*You will see remote branches as well*

**Create new branch**
$ git branch new_branch_name
Or create new branch and switch to it in the same command
$ git checkout -b new_branch_name

**Switch to existing branch**
$ git checkout existing_branch

**Delete a branch**
$ git branch -d my_branch_name
*To delete unmerged branch. Git don't want us to delete branch which is not merged into another branch because recovery a deleted branch is possible but not easy. That's why we have to use -D flag*
$ git branch -D my_branch_name

**To delete remote branch**
$ git push origin :my_branch
OR
$ git push origin --delete remote_branch_name

**Rename branch**
$ git branch -m new_branch_name

**View Files that have not been pushed**
$ git diff --numstat bb/project05
*Number of changes*

$ git diff --stat --cached origin/remote_branch_name

*Detail Number of changes*

$ git diff bb/project05
*All Code changes*

**Merge 2 branches**
*First checkout 2 branches. e.g.*
$ git branch
* master
  project07
*Now do this:*
$ git merge project07

**Revert committed changes**
*To revert back last commit*
$ git revert HEAD
*HEAD means last commit revision number*

$ git revert 2fbc28
2fbc28 is first 6 digits in revision number

**Revert unstaged/uncommitted changes**
$ git checkout -- .

**Revert staged and unstaged/uncommitted changes**
*Give following commands:*
*Revert changes to modified files. Unstage files. But do not remove new file created since HEAD.*
$ git reset --hard
*View all new/unstaged file and directories that will be removed.*
$ git clean -nd
*Remove all new untracked files and directories. (`-f` is `force`, `-d` is `remove directories`)*
$ git clean -fd

**Tag**
*Tags are checkout just like branches. Tags are used to mark some important event like project release in the history. But we normally don't commit anything when a tag is checked out. If we do then we will have to create a new branch otherwise changes will be lost.*

*Add tag*
$ git tag -a v01.0
*Or*
$ git tag -a v01.1 -m "comments"

*View all tag*

$ git tag

*View all remote tags*
$ git ls-remote --tags

*View tag details*
$ git show v1.0

*Push all tags*
$ git push -u origin --tags
*Or*
$ git push --tags

*Push a single tag*
$ git push origin v1.0

*Add tag to previous commit*
$ git tag -a v01.11 fc5ce2 -m 'comment'

*Checkout a tag*
$ git checkout v01.3
*Or*
$ git fetch && git checkout v01.3

**Alias**
*Create alias*
$ git config --global alias.co commit
*Then from now onwards you can commit by just using "co"*
$ git co -m 'ok'

**Helpfull global configs**
*Git will fix line feeds according to environment*
$ git config --global core.autocrlf false

$ git config --global push.default matching
*git by default will push to the same branch on remote. For both push and pull*

$ git config --global credential.helper wincred
*On Windows git will store the password in "wincred"*

$ git config --global credential.helper osxkeychain
*On MAC git will store the password in "wincred"*

$ git config credential.helper store
*On Ubuntu/linux git will store the password in "store"*

$ git config credential.helper 'cache --timeout=3600'
*stores password for 60 minutes*

$ git config --global merge.tool opendiff
*Use this command on OSX if xcode is installed. xcode has tool called opendiff that git can use to merge.*

**Ignore files in git**
**.gitignore**

```
*.ipr
*.iws
*.iml
*.class
*.orig
*.Backup
*.Base
*.Local
*.Remote
classes/
build/
gen/
.idea/
.git/
target/
.svn/
```

http://stackoverflow.com/questions/67699/clone-all-remote-branches-with-git

```
#!/bin/bash
for branch in `git branch -a | grep remotes | grep -v HEAD | grep -v master`; do
    git branch --track ${branch##*/} $branch
done

git fetch --all
git pull --all
```

**To configure Winmerge as mergetool of git, add below configuration in ~/.gitconfig file**
https://gist.github.com/shawndumas/6158524

```
[mergetool]
```

```
  prompt = false
  keepBackup = false
  keepTemporaries = false

[merge]
  tool = winmerge

[mergetool "winmerge"]
  name = WinMerge
  trustExitCode = true
  cmd = "/c/Program\\ Files\\ \\(x86\\)/WinMerge/WinMergeU.exe" -u -e -dl \"Local\" -dr
\"Remote\" $LOCAL $REMOTE $MERGED

[diff]
  tool = winmerge

[difftool "winmerge"]
  name = WinMerge
  trustExitCode = true
  cmd = "/c/Program\\ Files\\ \\(x86\\)/WinMerge/WinMergeU.exe" -u -e $LOCAL $REMOTE
```

**DiffMerge Configuration:**
Download DiffMerge

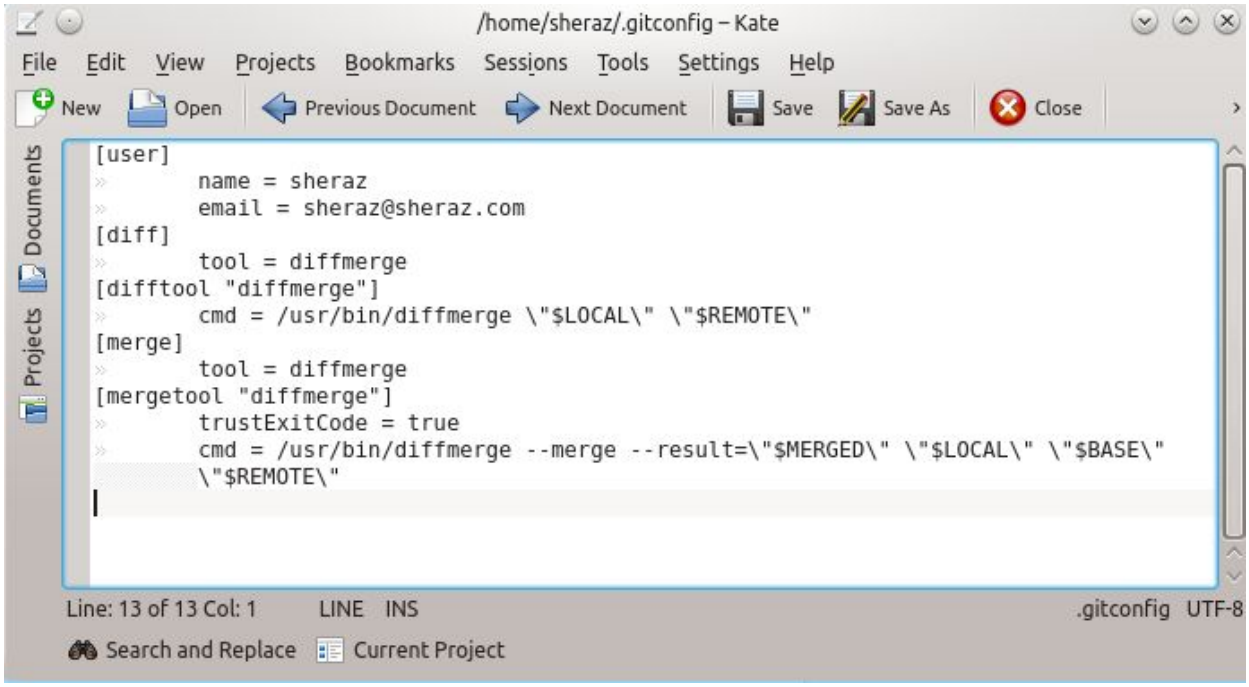
Run the following commands

```
$ git config --global diff.tool diffmerge
$ git config --global difftool.diffmerge.cmd "/usr/bin/diffmerge \"\$LOCAL\" \"\$REMOTE\""
$ git config --global merge.tool diffmerge
$ git config --global mergetool.diffmerge.trustExitCode true
$ git config --global mergetool.diffmerge.cmd "/usr/bin/diffmerge --merge --result=\"\$MERGED\"
\"\$LOCAL\" \"\$BASE\" \"\$REMOTE\""
```

**OR**
Add following lines in ~/.gitconfig file

```
[diff]
        tool = diffmerge
[difftool "diffmerge"]
        cmd = /usr/bin/diffmerge \"$LOCAL\" \"$REMOTE\"
[merge]
        tool = diffmerge
[mergetool "diffmerge"]
```

trustExitCode = true
cmd = /usr/bin/diffmerge --merge --result=\"$MERGED\" \"$LOCAL\" \"$BASE\" \"$REMOTE\"




**Stash**

$ git stash list
$ git stash save "Comments"
$ git stash apply "stash@{#}"
$ git stash drop "stash@{#}"
$ git stash pop
$ git stash clear

**Set upstream branch**
By doing this we have to specify branch name when pushing or pulling
$ git branch --set-upstream-to=origin/mybranch