[**CodeXam - YouTube**](https://www.youtube.com/channel/UCztgfCRJci6nx0VPVZcFstw/featured)

**Credit : @code.xam (Follow : instagram)**

**@gitxam (Follow : instagram)**

**@javaxam (Follow : instagram)**

**Founder: Subham**

1. **git** (instruction)
2. **pwd** (Print Working Directory: Prints out the current directory )
3. **cd** (Change Directory: To change this current working directory)
4. **git config --global user.name "FIRST\_NAME LAST\_NAME"**(Set your username)
5. **git config --global user.email "MY\_NAME@example.com"** (Set your Email)
6. **git status** (The git status command displays the state of the working directory and the staging area)
7. **git init** (The git init command creates a new Git repository)
8. **git add –a** or **git add -A** (Stages all changes)
9. **git add .** (stages new files and modifications, **without deletions** (on the current directory and its subdirectories)
10. **git add -u**(stages modifications and deletions, **without new files**)
11. **git commit -m “ your comment”** (The -m stands for message. When calling git commit , it is required to include a message.)
12. **git log** (The git log command displays a record of the commits in a Git repository.)
13. **git add File\_Name\_Example.txt** The git add command is used to add file contents to the Index (Staging Area)
14. **ls** (The Bash command ls is used to 'list' contents of the current working directory. ls is equivalent to DIR on a Windows console host terminal)
15. **git clone** [**https://github.com/tensorflow/tensorflow.git**](https://github.com/tensorflow/tensorflow.git) (git clone is primarily used to point to an existing repo and make a clone or copy of that repo at in a new directory, at another location)
16. **rm -rf .git** (Delete all the connection or entire repo)
17. **touch file.log** (Touch command allows to create a blank file example: flie.log)
18. **touch .gitignore** (.gitignore file is a text file that tells Git which files or folders to ignore in a project)

\* is used as a wildcard match and ignores all files. Example: \*.extension , \*.log

/ is used to ignore path names relative to the .gitignore file.Example: /dir

# is used to add comments to a .gitignore file.Example: #something

1. **git diff** (Diff command is used in git to track the difference between the changes made on a file)
2. **git diff –staged** (git diff --staged will only show changes to files in the "staged" area.)
3. **git commit -a -m “ Commit Message “**(skip the staging and directly commit.)
4. **git rm filename.extension** (git rm is used to remove a file from a Git repository.)
5. **git mv filename.extension renamefile.extension** (It will rename the file from filename.extension to renamefile.extension. And it will also get automatically staged by git. All we need to do is commit (mv means move) you can move by mv comment)
6. **git rm –cached file.extension**(It will remove that file from the tracker and it will become an untracked file.)
7. **Tab Key on the keyboard** (Use Tab Completion to Type Commands Faster on Any Operating System.)
8. **git log -p** (It will show what has been changed on a commit.)

**git log -p -2** (for seeing the last 2 changes.)

1. **git log --stat** ( we can get a brief summary of commits by typing.)
2. **git log --pretty=oneline** (If you want to see the commits on one line then type)
3. **git log --pretty=short** (If you just want to see the commits and The author then use .)
4. **git log --pretty=full** ( if you want a little bit more info like who is the committer, commit message.)
5. **git log --since=2.weeks** (If you want to filter commits by time.To see the last 2 months)

**git log --since=2.days** (To see the last 2 days)

**git log --since=2.weeks** (To see the last 2 weeks)

1. **git log --pretty=format:"%h -- %an"** (We can format the output by using the formatting codes provided in [Git’s](https://git-scm.com/docs/pretty-formats)(details) website. For an example we can use this format to print just the author name and the hash.)
2. **git config --global core.editor notepad** (you can easily configure git to use another editor. [All Code Editor](https://git-scm.com/book/en/v2/Appendix-C%3A-Git-Commands-Setup-and-Config))
3. **git commit --amend** (Now to change a commit. So, to change the most recent commit . Then an editor will open where you can change the commit message. All you need to do is change the message and then you need to close it by pressing i then esc key then type “ **:wq** ” to exit the editor. Now you have successfully edited the commit. [Details ( 7. Git Log: Viewing & Changing > **Customized Commit Output )**](https://docs.google.com/document/d/190sAY4DDUiqnZRU8ZprLMTd_6e3r3i9aOdA-26l5IYo/edit#heading=h.fl7q6mu8tu3a) )
4. **git restore --staged file.extension** (It will unstage the file and you can verify it by using “git status).
5. **git checkout -- file.extension** (let’s say we have modified the unstaged file and made some changes which were not necessary. Now the program is not working. So we have to restore that file to its previous state where it was working)
6. **git checkout -f** (To restore your entire working directory to the previous commit . It will restore your entire directory to the last commit. )
7. **git remote add origin git@github.com:yourusername/repositoryname.git** (We have a git repo in our local system we just want to push it to the remote system then copy the code on git’s website which looks like this: “git remote add origin git@github.com:yourusername/repositoryname.git”and paste it to the git bash and press enter. )
8. **Shift+Insert Key** (Paste Text On the terminal)
9. **git push -u origin master** (this command which pushes the files onto the remote server)
10. **ssh-keygen -t rsa -b 4096 -C "your\_email@example.com"** (Now just simply press enter 3 times)

**eval $(ssh-agent -s)** (Now run this command)

**ssh-add ~/.ssh/id\_rsa** (Now run this )

**tail ~/.ssh/id\_rsa.pub** (Now it will show you the **ssh** key ) ([Details:9.Remote Repositories](https://docs.google.com/document/d/190sAY4DDUiqnZRU8ZprLMTd_6e3r3i9aOdA-26l5IYo/edit#heading=h.z8n0twn6f4b8))

1. **git remote -v** (List the current remotes associated with the local repository.)

1. **git remote add [name] [URL]** (Add a remote.)
2. **git remote remove [name]** (Remove a remote.)[Details](https://github.com/git-guides/git-remote)
3. **git config --global alias.st status** (Alias is a term which is associated with shortcuts in Git. It shortens the long commands to increase efficiency Type “git config --global alias.st status” and press enter. Now what it means is if we type “**git st**” it will give us the output of “git status”)

**git config --global alias.ci commit** (Type “git config --global alias.ci commit” and press enter. Now what it means is if we type “**git ci**” it will give us the output of “git commit”)

**git config --global alias.unstage 'restore --staged --'** (Type “git config --global alias.unstage 'restore --staged --'” and press enter. Now what it means is if we type “**git unstage** file.extension” it will give us the output of “git restore --staged file.extension”) [Details(10.Setting Alias)](https://docs.google.com/document/d/190sAY4DDUiqnZRU8ZprLMTd_6e3r3i9aOdA-26l5IYo/edit#)

**git config --global alias.last 'log -p -1''** (Type “git config --global alias.last 'log -p -1''” and press enter. Now what it means is if we type “**git last** ” it will give us the output of “git log -p -1”)

1. **git checkout -b branchname** (creates a new branch and checks out the new branch while git branch BRANCH\_NAME creates a new branch but leaves you on the same branch.)( [Details 11.Creating Switching Branches](https://docs.google.com/document/d/190sAY4DDUiqnZRU8ZprLMTd_6e3r3i9aOdA-26l5IYo/edit#heading=h.mr8ccprk4t3i))
2. **git checkout master** (You can just checkout a remote tracking branch directly, with git checkout origin/master . This is similar to checking out a local branch as you would with git checkout master , but the difference is that any commits you make won't update master.)( [Details 11.Creating Switching Branches](https://docs.google.com/document/d/190sAY4DDUiqnZRU8ZprLMTd_6e3r3i9aOdA-26l5IYo/edit#heading=h.mr8ccprk4t3i))
3. **git checkout branchname** (You can just checkout your create branch.)( [Details 11.Creating Switching Branches](https://docs.google.com/document/d/190sAY4DDUiqnZRU8ZprLMTd_6e3r3i9aOdA-26l5IYo/edit#heading=h.mr8ccprk4t3i))
4. **git merge branchname** (Merging Branches in a Local Repository)
5. **git branch --merged** (If you want to see which branches were merged then use)
6. **git branch --no-merged** (If you want to see which branches were not merged then use)
7. **git branch -d branchname** (To delete a branch we need to use.Note that this command will fail if the branch we are deleting has not merged to the main branch)
8. **git branch -D branchname** (To delete a branch we need to use “git branch -d branchname”. Note that this command will fail if the branch we are deleting has not merged to the main branch. To tackle this error we will use “git branch -D branchname”, it will delete the branch even if it's not even merged yet.
9. **git branch -d origin branchname** ( We will delete the unnecessary branch )
10. **git branch -d origin branchname** ( We will delete the unnecessary branch )
11. **git push origin branchname/master** ( , it will push that branch to github and we can check that on github. )
12. **git push origin branchname/master** ( , it will push that branch to github and we can check that on github. )