git Most Used Commands

GIT Commands

Commands	Description
git --version	To check version of Git installed.
git config--system user.name <username>	System level (All users on the system)
git config --global user.name <username>	Global level (for all the projects of a user)
git config --local user.name <username>	Local (default and specific for a repository)
git config --global core.excludesfile %USERPROFILE%\.gitignore	. gitignore files should have the list of files to be ignored.
git add .	To add new and modified but not deleted.
git add -u	To add modified and deleted but not new files/directories

GIT Commands

Commands	Description
git Add -u	To add modified and deleted and new files/directories.
git commit --am "<message to commit>	Creates a commit and adds files to staging.
git commit --m "<message to commit>	Creates a commit from staging. Assumes files already added to staging
git init	Creates .git subfolder in the project, which contains all necessary metadata
git remote add origin https://github.com/Viyaan/<repository>.git	To add a new repo
git push origin <source_branch>:<destination branch>	Actual syntax but the below syntax is enough
git push -u origin <source branch name>	push your changes to selected branch, since source and destination will always be same. It's enough to mention only source branch.

GIT Commands

Commands	Description
git status	gives status of tracked and untracked changes in your local repository.
git branch	list all available branches in local system.
git branch -r	list all branches in repository
git branch <new branch name>	create a new branch
git checkout <branch name>	switch to another branch
git checkout -b <branch name>	creates and switches to another branch
git merge <branch name>	Merges specified branch into current.

GIT Commands

Commands	Description
git rm <file name>	To delete a file, you must commit and push after deleting.
git diff <file-name>	To see difference between uncommitted version to remote version of a file.
git diff	shows the changes you have done in your existing file (compare different versions of your files)
git log	Prints log of past commits
git log --oneline	Prints log of past commits in one line.
git tag	list all tags.
git tag <tag name>	create a tag. You need to push after new tag is created.

GIT Commands

Commands	Description
git branch -D <branch name>	To delete a branch. You must push after delete command.
git branch -d <branch name>	Same as above but this time it will ask for confirmation If its not merged.
git push origin :<destination branch>	After deleting you must push and while pushing you should mention the destination branch. Destination branch is the branch u deleted.
git tag -d <tag name>	To delete a tag, use small d and not capital.
git clone <clone url>	To clone a repository
git fetch origin	To fetch new data from remote repository. But it doesn't integrate any of those new data into your working files.

GIT Commands

Commands	Description
git pull	pull not only downloads new data; it also directly integrates it into your current working copy files.
git remote add <remote-name> <remote-url>	To add a new remote, use the git remote add command on the terminal, in the directory your repository is stored at.
git remote show	To see number of remote repositories available.
git stash	Git stash temporarily shelves or stashes changes made to your working copy so you can work on something else, and come back and re-apply them later.
git stash save <stash name>	To save list of uncommitted changes in stash.
git stash list	List of stashes.
git stash pop	Apply changes which were stashed.

GIT Commands

Commands	Description
git commit --amend	Modify the most recent commit
git commit --amend -m "an updated commit message"	Modify the most recent commit message
git rev -parse HEAD	Specify where head pointing too.

Reset

- git reset --soft <commit id>
- git reset --mixed <commit id>
  - The difference between --mixed and --soft is whether or not your index is also modified. So, if we're on branch master with this series of commits.
  - - A - B - C (master)
  - HEADpoints to C and the index matches C.
  - When we run git reset --soft B, master (and thus HEAD) now points to B, but the index still has the changes from C; git status will show them as staged. So if we run git commit at this point, we'll get a new commit with the same changes as C.
  - _____
  - Okay, so starting from here again.
  - - A - B - C (master)
  - Now let's do git reset --mixed B. (Note. --mixed is the default option). Once again, master and HEAD point to B, but this time the index is also modified to match B. If we run git commit at this point, nothing will happen since the index matches HEAD. We still have the changes in the working directory, but since they're not in the index, git status shows them as unstaged. To commit them, you would git add and then commit as usual.
  - _____
  - And finally, --hard is the same as --mixed (it changes your HEAD and index), except that --hard also modifies your working directory. If we're at C and run git reset --hard B, then the changes added in C, as well as any uncommitted changes you have, will be removed, and the files in your working copy will match commit B. Since you can permanently lose changes this way, you should always run git status before doing a hard reset to make sure your working directory is clean or that you're okay with losing your uncommitted changes.

Reset

- git reset --hard <commit id>

  resets the HEAD to any commit mentioned in command. And throws away all your changes which is made after the commit corresponding to the commit id.