GIT COMMANDS

Git task	Notes	Git commands
Tell Git who you are	Configure the author name and email address to be used with your commits. Note that Git strips some characters (for example trailing periods) from user.name.	git config --global user.name "Sam Smith" git config --global user.email sam@example.com
Create a new local repository		git init
Check out a repository	Create a working copy of a local repository:	git clone /path/to/repository
	For a remote server, use:	git clone username@host:/path/to/repository
Add files	Add one or more files to staging (index):	git add <filename> git add *
Commit	Commit changes to head (but not yet to the remote repository):	git commit -m "Commit message"
	Commit any files you've added with git add, and also commit any files	git commit -a

	you've changed since then:	
Push	Send changes to the master branch of your remote repository:	git push origin master
Status	List the files you've changed and those you still need to add or commit:	git status
Connect to a remote repository	If you haven't connected your local repository to a remote server, add the server to be able to push to it:	git remote add origin <server>
	List all currently configured remote repositories:	git remote -v
Branches	Create a new branch and switch to it:	git checkout -b <branchname>
	Switch from one branch to another:	git checkout <branchname>
	List all the branches in your repo, and also tell you what branch you're currently in:	git branch
	Delete the feature branch:	git branch -d <branchname>

	Push the branch to your remote repository, so others can use it:	git push origin <branchname>
	Push all branches to your remote repository:	git push --all origin
	Delete a branch on your remote repository:	git push origin :<branchname>
Update from the remote repository	Fetch and merge changes on the remote server to your working directory:	git pull
	To merge a different branch into your active branch:	git merge <branchname>
	View all the merge conflicts: View the conflicts against the base file: Preview changes, before merging:	git diff git diff --base <filename> git diff <sourcebranch> <targetbranch>
	After you have manually resolved any conflicts, you mark the changed file:	git add <filename>
Tags	You can use tagging to mark a significant changeset, such as a release:	git tag 1.0.0 <commitID>

	CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using:	git log
	Push all tags to remote repository:	git push --tags origin
Undo local changes	If you mess up, you can replace the changes in your working tree with the last content in head: Changes already added to the index, as well as new files, will be kept.	git checkout -- <filename>
	Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this:	git fetch origin git reset --hard origin/master
Search	Search the working directory for foo():	git grep "foo()"

Rebasing:

- ✓ 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.

Example: git stash
Git stash pop

Managing Multiple Stashes:

Git stash list

For particular stash:
git stash pop <stashid>

For Viewing differernces between multiple stashes:
Git stash show
Or git stash show -p

To Delete a file from git:
Git rm <filename>

To delete a directory:
Git rm -r <directoryname>

Revert an entire commit in history by commit ID:

Let's assume that you discovered a bug somewhere in the code and you found that the bug exists in one of the commits in the history and the only way to resolve this is to remove the entire commit but this time, the commit is not the last one (head) in your local repository. In this section, you will remove a whole commit by commit ID.

The first step is to run the git log command (mentioned before) to check the commit IDs in the history then copy the target commit ID you want to delete and run the following command:
git revert 089148c

Adding Remote Repositories

To add a new remote Git repository as a shortname you can reference easily, run git remote add <shortname> <url>:

$ git remote origin
$ git remote add pb https://github.com/srinivaskolaparthi/ticgit
git remote -v

Fetch using shortname:
Git fetch <shortname>

Pushing to Your Remotes

- When you have your project at a point that you want to share, you have to push it upstream.
- The command for this is simple: git push <remote> <branch>.
- If you want to push your master branch to your origin server (again, cloning generally sets up both of those names for you automatically), then you can run this to push any commits you've done back up to the server:

$ git push origin master

Inspecting a Remote

If you want to see more information about a particular remote, you can use the git remote show <remote> command. If you run this command with a particular shortname, such as origin, you get something like this:

$ git remote show origin

Renaming and Removing Remotes

You can run git remote rename to change a remote's shortname. For instance, if you want to rename pb to kola, you can do so with git remote rename:

$ git remote rename pb kola
$ git remote origin

For removing remote:

git remote remove kola

Note:
1)git fetch is similar to pull but doesn't merge. i.e. it fetches remote updates (refs and objects) but your local stays the same (i.e. origin/master gets updated but master stays the same) .
2)git pull pulls down from a remote and instantly merges.
3)Checkout:
1st case : switch between branch in local repository For instance :
git checkout exists_branch_to_switch

You can also create new branch and switch out in throught this case with -b

git checkout -b new_branch_to_switch

2nd case : restore file from x rev
git checkout rev file_to_restore
4)Install mergetool:

Step 1: Run following commands in your terminal
git config merge.tool vimdiff
git config merge.conflictstyle diff3
git config mergetool.prompt false
This will set vimdiff as the default merge tool.

Step 2: Run following command in terminal
git mergetool

LOCAL – this is file from the current branch
BASE – common ancestor, how file looked before both changes

REMOTE – file you are merging into your branch
MERGED – merge result, this is what gets saved in the repo