**All about GitHub Commands**

**Setup**

Configuring user information used across all local repositories

* **git config --global user.name “[firstname lastname]”**

- Set a name that is identifiable for credit when review version history

* **git config --global user.email “[valid-email]”**

- set an email address that will be associated with each history marker

* **git config --global color.ui auto**

- Set automatic command line coloring for Git for easy reviewing

**INIT**

Configuring user information, initializing and cloning repositories

* **git init**

- Initialize an existing directory as a Git repository

* **git clone [url]**

- Retrieve an entire repository from a hosted location via URL

**Stage and Snapshot**

Working with snapshots and the Git staging area

* **git status**

- show modified files in working directory, staged for your next commit

* **git add [file]**

- Add a file as it looks now to your next commit (stage)

* **git reset [file]**

- Unstage a file while retaining the changes in working directory

* **git add . or git add -A**

- Add all new and changed files to the staging area

* **git diff**

- Diff of what is changed but not staged

* **git diff --staged**

- Diff of what is staged but not yet committed

* **git commit -m “[descriptive message]”**

- Commit your staged content as a new commit snapshot

**Branch and Merge**

Isolating work in branches, changing context, and integrating changes

* **git branch**

- List your branches. A \* will appear next to the currently active branch

* **git branch [branch-name]**

- Create a new branch at the current commit

* **git checkout -b [branch-name]**

- Create a new branch and switch to it

* **git checkout -b [branch-name] origin/[branch-name]**

- Clone a remote branch and switch to it

* **git merge [branch]**

- Merge the specified branch’s history into the current one

* **git checkout  [branch-name]**

- Switch to a branch

* **git checkout -**

- Switch to the branch last checked out

* **git checkout -- [file-name]**

- Discard changes to a file

* **git branch -d [branch-name]**

- Delete a branch

* **git push origin -delete [branch-name]**

- Delete a remote branch

* **git merge [source branch] [target branch]**

- Merge a branch into a target branch

* **git merge [branch-name]**

- Merge a branch into the active branch

* **git log**

- Show all commits in the current branch’s history

**Inspect and Compare**

Examining logs, diffs and object information

* **git log**

- Show the commit history for the currently active branch

* **git log branchB..branchA**

- Show the commits on branchA that are not on branchB

* **git log --follow [file]**

- Show the commits that changed file, even across renames

* **git diff branchB...branchA**

- Show the diff of what is in branchA that is not in branchB

* **git show**

- Show any object in Git in human-readable format

**Tracking Path Changes**

Versioning file removes and path changes

* **git rm [file]**

- Delete the file from project and stage the removal for commit

* **git mv [existing-path] [new-path]**

- Change an existing file path and stage the move

* **git log --stat -M**

- Show all commit logs with indication of any paths that moved

**Ignoring Patterns**

Preventing unintentional staging or commiting of files

* **git config --global core.excludesfile [file]**

- System wide ignore pattern for all local repositories

**Share and Update**

Retrieving updates from another repository and updating local repos

* **git remote add [alias] [url]**

- Add a git URL as an alias

* **git fetch [alias]**

- Fetch down all the branches from that Git remote

* **git merge [alias]/[branch]**

- Merge a remote branch into your current branch to bring it up to date

* **git push origin [branch]**

- Push a branch to your remote repository

* **git push -u origin [branch]**

- Push changes to remote branch and remember the branch

* **git push**

- Push changes to remote repository to remembered branch

* **git push origin --delete [branch]**

- Delete a remote branch

* **git pull**

- Fetch and merge any commits from the tracking remote branch

* **git pull origin [branch]**

- Pull changes from remote repository

**Rewrite History**

Rewriting branches, updating commits and clearing history

* [**git rebase**](https://git-scm.com/docs/git-rebase) **[branch]**

- Apply any commits of current branch ahead of specified one

* **git reset --hard [commit]**

- Clear staging area, rewrite working tree from specific commit

**Temporary Commits**

Temporarily store modified, tracked files in order to change branches

* **git stash**

- Save modified and staged changes

* **git stash list**

- List stack-order of stashed file changes

* **git stash pop**

- Write working from top of stash stack

* **git stash drop**

- Discard the changes from top of stash stack

**Undo to previous file**

* **git log --oneline**

- List of all commit with commit id and commit message

* **git checkout[commit-id]**

- Return to previous commit

* **git revert [commit-id]**

- Revert to previous commit

* **git reset [commit-id]**

- Reset to previous commit(remove history of all commit after)

* **git rm --cached [file/folder]**

- Stop a file being cached

* **git checkout [file/to/restore]**

- Return a file to a previous commit

**Others:**

**Debugging**

* [**git bisect <subcommand> <options>**](https://git-scm.com/docs/git-bisect)

- The git bisect tool is an incredibly helpful debugging tool used to find which specific commit was the first one to introduce a bug or problem by doing an automatic binary search

* [**git blame**](https://git-scm.com/docs/git-blame)

- The git blame command annotates the lines of any file with which commit was the last one to introduce a change to each line of the file and what person authored that commit. This is helpful in order to find the person to ask for more information about a specific section of your code

* [**git grep**](https://git-scm.com/docs/git-grep) **- Print lines matching a pattern**

- The git grep command can help you find any string or regular expression in any of the files in your source code, even older versions of your project

**Patching**

* [**git cherry-pick**](https://git-scm.com/docs/git-cherry-pick)

- Apply the changes introduced by some existing commits

**Note: For commands which i found new i have attached link for each command(at its name)**