In the Previous Parts of the series I have covered the most Up and running topics like Git Tutorial For Beginners, Manage Branches & Conflicts Resolution In Git and Best Git GUI Clients For Different Platforms. In this article I will cover the most used Git commands. I will list down commands by the category and provides the description of each. Git has an enormous set of commands which will be hard to remember.

You can save the cheat sheet for not having any confusions in commands while maintaining the versions. In addition I will also define some Terminologies for your ease so that you know what the specific meaning of the term in Git.

Let's start with the Git Terminologies first.

Git Terminologies

S.no	Terminology	Description
1	Bare Repository	Repository without working directory
2	Branch	A branch is an active area of development in Git. The most recent commit shows the tip of branch
3	Blame	Describes the last modifications of each line in file. Show Revision Author & Time
4	checkout	This means to take any given commit from the repository and Recreate the state of the associated file and directory tree in the working directory.
5	commit	This is a single point in git history which holds the information about the changeset
6	diff	A diff is the difference in changes between two commits, or saved changes
7	Detached Head	This exact state - when a specific commit is checked out instead of a branch - is what's called a "detached HEAD"
8	Fetch	Fetching means to get latest changes in branch and local/remote repos
9	Fork	By forking the repository you will be able to add commits and create Pull Requests
10	Hash	A unique SHA1 code for every commit
11	Head	A named reference to the commit at the tip of a branch
12	Index	It is a collection of files with state informations.

13	Merge	To bring out the content of another branch in the current branch
14	Master	It is the default development branch in git
15	Origin	It is the default Upstream Repository
16	Pull Request	Suggest some changes into the master branch
17	Push	Push new changes after committing them
18	Repository	A collection of commits, and branches and tags to identify commits.
19	Working Tree	The tree of actual checked out files

Git Configuration

S.no	Git Commands	Description
1	git config --global user.name <name>	Set the user name to be used for all actions.
2	git config --global user.email <email>	Set the email to be used for all the actions.
3	git config --global alias.<alias-name><git-command>	Create a shortcut for the git command
4	git config --system core.editor <editor>	Set text editor for all the command actions
5	git config --global --edit	Open global configuration file in text editor for manual editing
6	git config --global color.ui auto	Enable helpful colourization of command line outputs

Setting Up a Git Repo

S.no	Git Command	Description
1	git init	Initialize an empty git repo in current project
2	git clone (Repo URL)	Clone the repository from github to the project folder
3	git clone (Repo URL) (Folder)	Clone the repository into the specific folder

4	git remote add origin https://github.com/username/(repo_name).git	Create a remote repo pointing on your GitHub repository which you already created there.
5	git remote	Show the name of remote repositories
6	git remote -v	Show the name and URL's of the remote repositories
7	git remote rm (remote repo name)	Remove the remote repository
8	git remote set-url origin (git URL)	Change the URL of the Repository
9	git fetch	Get the latest changes from the origin but not merge
10	git pull	Get the latest changes from the origin and merged them

Local File Changes

S.no	Git Command	Description
1	git add (file name)	Add the current changes of file to staging
2	git add .	Add the whole directory changes to staging(no delete files)
3	git add -A	Add all new, modified and deleted files to staging
4	git rm (file_name)	Remove the file and untrack it.
5	git rm --cached (file_name)	Untrack the current file
6	git mv (file_name) (new_file_name)	Changes the filename and prepare to commit
7	git checkout <deleted file name>	Recover the deleted file and prepare for commit
8	git status	Shows the status of files modified
9	git ls-files --other --ignored --exclude-standard	Shows the list of all ignored files
10	git diff	Show unstaged changes in index and working directory
11	git diff --staged	Shows file differences between staging and the last file version
12	git diff (file_name)	Show changes in single file compared to last commit

Declare Commits

S.no	Git Commands	Description
1	git commit -m “(message)”	Commit the changes with a message
2	git commit -am “(message)”	Add all changes to staging and commit them with message
3	git checkout <commit hash>	Switch to the provided commit
4	git show <commit hash>	Outputs metadata and content changes of the specified commit
5	git reset <commit hash>	Undo all commits after this commit. Preserving changes locally
6	git reset --hard <commit hash>	Discard all history and changes back to the given commit
7	git reset --hard Head	Discard all local changes in working directory
8	git log	Show history of changes
9	git log -p	Shows the full display of each commit
10	git log -oneline	Shows the list of commits with only message
11	git log --follow (file_name)	List all the history for the current file
12	git blame (file_name)	Shows all changes with name of user
13	git stash	Temporarily saves all modified tracked files
14	git stash pop	Restores the most recently stashed files
15	git stash list	List all stash changedsets
16	git stash apply	Apply the latest stashed contents
17	git stash drop	Discard the most recently stashed files
18	git stash apply (stash id)	Re-Apply a specific stash content by ID
19	git stash drop (stash_id)	Drop a specific stash content by ID
20	git push	Push changes to the Origin
21	git push origin (branch_name)	Push branch to the Origin

22	Git push -f origin (branch_name)	Force push the changes to the origin
23	git tag (tag_name)	Define tag for a version
24	git diff <commit>^<commit>	

Branching

S.no	Git Commands	Description
1	git branch	Shows the list of all branches
2	git branch <branch_name>	Create a new branch
3	git branch -a	List all branches local and remote
4	git branch -m <old_name> <new_name>	Rename the branch
5	git checkout -b <branch_name>	Create a branch and switch to it
6	git checkout <branch_name>	
7	git checkout -b <new_branch_name> origin/<branch_name>	Get a remote branch from origin to the local directory
8	git branch -d <branch_name>	Delete a specified branch
9	git merge <branch_name>	Merge the current branch into master(first checkout to master)
10	git rebase <branch_name>	Take all the changes of the branch and restate on other.
11	git rebase <base>	Rebase the current branch onto base. Base can be a commit ID or branch name.
12	git fetch remote <branch_name>	Fetches the specific branch from the repository
13	git diff <branch_name>..<branch_name>	Shows the differences of two branches
14	git pull --rebase	Fetch the remote’s copy of current branch and rebases it into the local copy
15	git push --all	Push all of your local branches to the specified remote

Resorces:
https://scotch.io/bar-talk/git-cheat-sheet
https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf
https://atlassian.com/git
https://git-scm.com/docs/gitglossary

Conclusion

Commands are the essential part of the git. Without commands you cannot perform actions so it’s better to memorize them or save.I have list down almost all the necessary commands in Git. But you can also contribute by commenting more commands if i missed any. I will add them in above tables.
If you have any query or suggestion please comment below