GIT Cheat Sheet

By Ganesh Kumar

What is Git ?

Git is the most commonly used Version Control System. Git tracks the changes you make to files, so you have a record of what has been done, and you can revert to specific versions should you ever need to. Git also makes the collaborations easier, allowing changes by multiple people to all be merge into one source.

SETUP:

Configuring user information used across all local repositories:

1	**Git config --global user.name "Firstname Lastname"**	Configuring the name
2	**Git config --global user.name "Valid-email"**	Configuring the email

GIT BASICS:

Configuring user information, initializing and cloning repositories

1	**git init**	Initialize an existing directory as a Git repository
2	**git clone [url]**	Retrieve an entire repository from a hosted location via URL

STAGE & SNAPSHOT:

Working with snapshots and the Git staging area

1	**git status**	Show the modified files in the working directory, staged for your next commit
2	**git add [file]**	Add a file as it looks now to your next commit (stage)
3	**git reset [file]**	Unstage a file while retaining the changes in working directory
4	**git diff**	Diff of what is changed but not staged
5	**git diff --staged**	Diff of what is staged but not yet committed
6	**git commit -m "[descriptive message]"**	Commit your staged content as a new commit snapshot

BRANCH & MERGE :

Isolating work in branches, changing context, and integrating changes

1	**git branch**	List your branches. a * will appear next to the currently active branch
2	**git branch [branch-name]**	Create a new branch at the current commit
3	**git checkout**	Switch to another branch and check it out into your working directory
4	**git merge [branch]**	Merge the specified branch's history into the current one
5	**git log**	Show all commits in the current branch's history

INSPECT & COMPARE

Examining logs, diffs and object information

1	**git log**	Show the commit history for the currently active branch
2	**git log branchB..branchA**	Show the commits on branchA that are not on branchB
3	**git log --follow [file]**	Show the commits that changed file, even across renames
4	**git show [SHA]**	Show any object in Git in human-readable format
5	**git diff branchB...branchA**	Show the diff of what is in branchA that is not in branchB

SHARE & UPDATE

Retrieving updates from another repository and updating local repos

1	**git remote add [alias] [url]**	Add a git URL as an alias
2	**git fetch [alias]**	Fetch down all the branches from that Git remote
3	**git merge [alias]/[branch]**	Merge a remote branch into your current branch to bring it up to date
4	**git push [alias] [branch]**	Transmit local branch commits to the remote repository branch
5	**git pull**	Fetch and merge any commits from the tracking remote branch

TRACKING PATH CHANGES

Versioning file removes and path changes

1	**git rm [file]**	Delete the file from project and stage the removal for commit
2	**git mv [existing-path] [new-path]**	Change an existing file path and stage the move
3	**git log --stat -M**	Show all commit logs with indication of any paths that moved

REWRITE HISTORY

Rewriting branches, updating commits and clearing history

1	**git rebase [branch]**	Apply any commits of current branch ahead of specified one
2	**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

1	**git stash**	Save modified and staged changes
2	**git stash list**	List stack-order of stashed file changes
3	**git stash pop**	Write working from top of stash stack
4	**git stash drop**	Discard the changes from top of stash stack