Git: Essential Commands

Setup

Configure user email:
git config --global user.email 'email_id'
Configure user name:
git config -- user.name 'name'

Start Project

Initialize current directory (unversioned project) to Git repository:
git init
Initialize a new, empty repository:
git init <directory>
Create a local copy of a remote repository
git clone <url>

Branches

1. List all local branches (asterisk denotes the current branch):
git branch
2. List all branches (local & remote):
git branch -a
3. Create a new local branch:
git branch <branch_name>
4. Create a new branch & switch to it:
git checkout -b <branch_name>
5. Clone a remote branch & switch to it:
git checkout -b <branch_name> origin/<remote_branch_name>
6. Rename local branch:
git branch -m <old_name> <new_name>
7. Switch to a branch:
git checkout <branch_name>
8. Switch to last checked out branch:
git checkout -
9. Delete a local branch:
git branch -d <branch_name>
10. Delete a remote branch:
git push origin --delete <branch_name>

Making Changes

1. Lists all files that yet to be committed:
git status
2. Add file or files to the staging area:
git add <filename, filename2>
3. Add all files to the staging area:
git add .
4. Commit changes (snapshots commits permanently in the version history)
git commit -m 'commit_msg'
5. Commit all changes (added changes using ***add*** & any changes since then):
git commit -a -m 'commit_msg'
6. Add all changes & commit:
git commit -am 'commit_msg'
7. Reset changes:
git reset <filename>

Merge

1. Merge given branch into the current branch:
git merge <source_branch>
2. Merge a branch into a target branch:
git merge <source_branch> <target_branch>
3. Merge source branch into the current branch, but always generates a merge commit (even if it's a fast-forward merge):
git merge --no-ff <source_branch>
4. Merge & squash all commits to one new commit all changes to one commit:
git merge –squash <source_branch>

Stash

1. Stash takes uncommitted changes (staged and unstaged), saves them away for later use, and then reverts them from your working copy. Add ***-u*** to include untracked files, Add ***-a*** to include untracked, ignored files:
git stash
2. Stash with a comment:
git stash save 'comment'

Stash cont'd

3. Partial stashing one file or set of files:
git stash -p
5. List all stashes:
git stash list
4. Reapply stash without deleting it:
git stash apply
5. Reapply stash at index 2, then delete it from stash list. Omit stash@{n} to pop the most recent stash:
git stash pop stash@{2}
6. Show diff summary of stash , Pass -p flag to see the full diff:
git stash show stash@{1}
7. Delete stash at index, Omit stash@{n} to delete last stash made:
git stash drop stash@{1}
8. Delete all stashes:
git stash clear

Log & Comparisons

1. View changes:
git log
2. View summary of changes:
git log --summary
3. View changes briefly (one-line):
git log --oneline
4. Preview changes before a merge:
git diff <source_branch> <target_branch>
5. Show changes between two commits:
git diff <commit_id1> <commit_id2>

Rebase

1. Rebase(standard mode) will take the commits in current working branch & apply them to the head of passed branch:
git rebase <base_branch_name>
2. Rebase(interactive mode) launches a interactive rebasing session used to clean up history by removing, splitting, and altering an existing series of commits:
git rebase -i <base_branch_name>

Synchronization

1. Fetch the complete repository:
git fetch <repo_url>
2. Fetch a specific branch:
git fetch <repo_url> <branch_name>
3. Fetch all the branches simultaneously:
git fetch -all
4. Default git pull is equivalent to git fetch origin HEAD & git merge HEAD where HEAD is ref pointing current branch:
git pull
5. Fetch specified remote's copy of current branch & merge it into the local copy:
git pull <remote>
6. Pull changes from a specific branch:
git pull origin <branch_name>
7. Fetches the remote content but does not create a new merge commit:
git pull --no-commit <remote>
8. Same as the previous pull but instead of using git merge to integrate the remote branch with the local one, use git rebase:
git pull --rebase <remote>
9. Gives verbose output during a pull which displays the content being downloaded and the merge details:
git pull --verbose
10. Push the specified branch to , along with all of the necessary commits & internal objects. This creates a local branch in the destination repository:
git push <remote> <branch>
11. Same as the above command, but force the push even if it results in a non-fast-forward merge:
git push <remote> --force
11. Push all local branches to remote repo:
git push <remote> --all
12. Push all of local tags to remote repo:
git push <remote> --tags

Teja Maridu
@tjmr
Follow for more