Git Remote

$ git remote add feature1 git@github.com:phpcodemaker/feature1.git

$ git remote add feature2 git@github.com:phpcodemaker/feature2.git

$ git remote –v

$ git push feature2

Initializing repository

$ git init

Creating new branch

$ git branch <new-branch-name>

$ git checkout -b <new-branch-name>

$ git switch -c <new-branch-name>

Rename a branch-name

$ git branch -M <new-branch-name>

Listing Branches

List local branches

$ git branch -a

Listing remote branches which took from the last updates

$ git branch -r

Switching branch

$ git switch <branch-name>

$ git checkout <branch-name>

Fetch/Pull updates from Repo

$ git fetch # simply pull updates from remote repo and do not merge

$ git pull # combine of fetch & merge updates from remote repo

$ git pull –no-ff # fatal: Not possible to fast-forward, aborting.

$ git pull –rebase # on top of remote changes, local changes will be applied

$ git pull --no-ff --allow-unrelated-histories # fatal: refusing to merge unrelated histories

Fetch → useful for fetching remote updates without affecting(merging) local changes. In case a new branch is available in remote, that'll we be pulled in local.

Rebase commits

$ git rebase -i < HEAD~n >

Rebase c18ab44..68391ee onto c18ab44 (2 commands)

Commands:

p, pick <commit> = use commit

r, reword <commit> = use commit, but edit the commit message

e, edit <commit> = use commit, but stop for amending

s, squash <commit> = use commit, but meld into previous commit

f, fixup [-C | -c] <commit> = like "squash" but keep only the previous

commit's log message, unless -C is used, in which case

Keep only this commit's message; -c is same as -C but

opens the editor

x, exec <command> = run command (the rest of the line) using shell

b, break = stop here (continue rebase later with 'git rebase --continue')

d, drop <commit> = remove commit

l, label <label> = label current HEAD with a name

t, reset <label> = reset HEAD to a label

m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]

. create a merge commit using the original merge commit's

. message (or the oneline, if no original merge commit was

. specified); use -c <commit> to reword the commit message

These lines can be re-ordered; they are executed from top to bottom.

If you remove a line here THAT COMMIT WILL BE LOST.

However, if you remove everything, the rebase will be aborted.

Staging changed files

$ git add . # Stage all the files

$ git add <filename1> <filename2> …

$ git add (–p | --patch) # Add changes to staging what you require but not everything

Committing changes

$ git commit –m “Commit message here”

staging and committing files with message, won’t work for new files

$ git commit –am “commit message here”

rewriting the previous commit with new msg and files

$ git commit –amend –m “Commit message here”

Merge a branch <& commits> in the current branch

$ git merge <branch-name>

$ git pull <branch-name>

Revert changes/commits in git

$ git revert <commit-sha> # Reverting merged changes from a branch

$ git reset <commit-sha>

$ git reset HEAD

$ git reset –soft < commit-sha | HEAD^n | HEAD~n >

$ git reset –HARD < commit-sha | HEAD^n | HEAD~n >

Undo `$ git reset` revert

$ git reset ORIG_HEAD # Reset back the HEAD to where it was

Git Cherry-pick

$ git cherry-pick <commit-sha>

Alternate to achieve Cherry-pick

(main|master)$ git checkout –b <new-branch> **# checkout current branch commits to new branch**

(new-branch)$ git push origin <new-branch> **# push new branch commits to remote**

(new-branch)$ git checkout (main|master) **# checkout back the accidental commit branch**

(main|master)$ git reset –hard <commit-sha> **# remove the commits using commit-sha**

Reflog

$ git reflog # View reflog

Restore hard deleted commits from Reflog

$ git reset --hard <commit-sha> # Restore in same branch

$ git branch <new-branch-name> <commit-sha> # Restore in new branch

Restore deleted branch from Reflog

$ git branch <new-branch-name> <commit-sha> # Restore the branch

Resolving conflicts

$ git mergetool

This message is displayed because 'merge.tool' is not configured.

See 'git mergetool --tool-help' or 'git help config' for more details.

'git mergetool' will now attempt to use one of the following tools:

meld opendiff kdiff3 tkdiff xxdiff tortoisemerge gvimdiff diffuse diffmerge ecmerge p4merge araxis bc codecompare smerge emerge vimdiff nvimdiff