Configure name:

git config –global user.name “Duy Le”

Configure email:

git config –global user.email [hle@nysus.net](mailto:hle@nysus.net)

Staged change:

git add <filename1> <filename2>

Commit change:

git commit

Commit change will message:

git commit -m “Message”

Stage and commit all change:

git commit -am

Fix previous commits mistake:

git commit -amend

Ignore files inside .gitignore file

View all branch:

git branch

Create new branch:

git branch <branchname>

Switch branch:

git switch <branchname> or git checkout <branchname>

Create and switch to new branch:

git switch -c <branchname> or git checkout -b <branchname>

Delete a branch: (to delete a branch, we cannot be in that branch, have to switch to the other one)

git branch -d <branchname> or git branch -D <branchname> (force)

Rename a branch: (to rename a branch, we must be in that branch)

git branch -m <newbranchname>

Fast forward merge (merge master to catch up with some branch, we also need to move to the receiving branch, in this case it is master)

git switch master

git merge bugfix

List all the changes in the working directory that are not staged for the next commit

git diff

List all the changes in the working directory that are staged for the next commit

git diff - - staged or git diff - - cached

List all the changes in the working directory that are staged or unstaged or untracked file

git diff HEAD

Diffing Specific files

git diff HEAD filename git diff - - staged filename

Comparing Branches

git diff branch1..branch2

Comparing Commits

git diff commit1..commit2

HEAD~1: parent commit of HEAD

Push current changes not ready to commit on 1 branch:

git stash git stash save

Retrieve changes after solving on other branch:

git stash pop

Copy the changes in stash without removing change from stash:

git stash apply

View list of stashses:

git stash list

Apply specific stash:

git stash apply stash@{1} git stash apply stash@{2}

Dropping/deleting stash:

git stash drop stash@{1}

Clear all stash:

git stash clear

Go to DETACHED HEAD state:

git checkout [commit-hash]

Return to branch before detaching HEAD:

git switch –

Discard change to content previously committed:

git checkout HEAD [filename] git checkout - - [filename]

git restore [filename]

Discard changes to content many previous commits ago:

git restore - - source HEAD~2 [filename]

Unstage file:

git restore - - staged [filename]

Reset a repo to a specific commit but does not remove the changes from working directory:

git reset [commit-hash]

Reset a repo to a specific commit but also remove the changes:

git reset - - hard [commit-hash]

Git Reset vs Git Revert:

Reset is going backwards and delete commits

Revert is also going backwards but create new commit while undoing changes

Git Revert:

git revert [commit-hash] git revert HEAD~2

Clone a repo:

git clone [github-url]

Check to see if there is any remote yet:

git remote -v

Add remote to existing project:

git remote add origin [github-url]

Push a local copy to github: (push to master branch of Github)

git push origin master

Connect upstream:

git push -u origin

View all remote branches:

git branch -r

Checkout remote branch pointers: (go to DETACHED HEAD)

git checkout origin/master

Checkout remote branch: (change local reference to new remote branch => DETACHED HEAD)

git checkout origin/[remote-branch]

Switch remote branch: (create a local copy of remote branch)

git switch [remote-branch] git checkout - - track origin/[remote-branch]

Git Fetch: get changes from GitHub and “create new branch origin/master” to store the change, not affect local working directory on master branch, use checkout origin/master to see the changes

git fetch [remote] git fetch origin

git fetch [remote] [branch]

Git Pull: get changes from GitHub and update working directory, pull = fetch + merge

git pull [remote] git pull origin

git pull [remote] [branch]

Git Rebase: (never rebase commits that already shared with other people)

git switch feature

git rebase master

Interactive Rebase:

git rebase -I HEAD~4

Viewing all tags:

git tag

View tags by LIKE ‘%%’:

git tag -l “\*beta\*” (view all tags that include “beta” in their name)

Check out tag:

git checkout <tag>

Diff tags:

git diff <tag1> <tag2>

Create lightweight tag:

git tag <tagname> git tag 1.0.0

Create annotated tag:

git tag -a <tagname>

Move tag:

git tag <tagname> <commit-hash> -f

Delete tag:

git tag -d <tagname>

Push tags:

git push - - tags (push all tags)

git push origin <tagname> (push specific tags)

Show reflog:

git reflog show HEAD@{2}

Reflog reference:

git reflog master{yesterday}

git reflog master{one.week.ago}

Create custom alias:

git config - -global alias.[custom-name] [git-command]