**Forking git workflow**

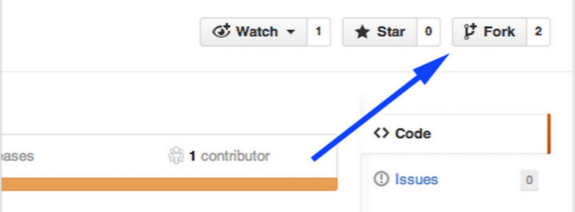
os Ubuntu 14.04 via Terminal

contributed repository account on github [http://github.com/contrib](http://github.com/contrib/repo)

my repository account on github <http://github.com/user>

repository name repo

**Fork repository you contribute to from github and clone forked**



$ git clone “<http://github.com/user/repo>

**Add upstream to update fork when necessary**

$ git remote add upstream “[http://github.com/contrib](http://github.com/contrib/repo)/repo.git”

Show remote

$ git remote -v

origin [http://github.com/](http://github.com/contrib/repo)user/repo.git (fetch)

origin [http://github.com/](http://github.com/contrib/repo)user/repo.git (push)

upstream [http://github.com/](http://github.com/contrib/repo)contrib/repo.git (fetch)

upstream [http://github.com/contrib](http://github.com/contrib/repo)/repo.git (fetch)

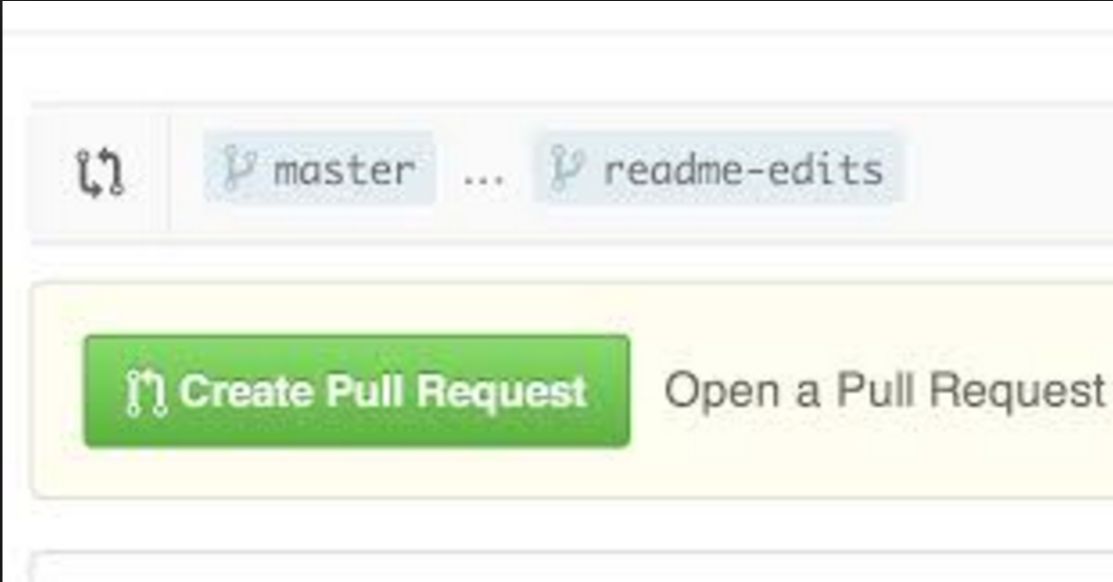
**Work on particular feature and add commits using add and commit**

$ git add file

$ git commit -m “Adding file”

**Push to your fork and let it be review by others on github by creating pull request**

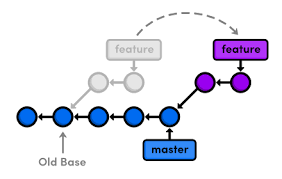
$ git push origin master



**Update repository to synchronize with upstream**

$ git fetch upstream

$ git rebase upstream/master (Put your commits on top of upstream master )



**Various scenarios of update**

There are three possibilities what could happen,

1. no new commits on upstream and origin

**Update repository to synchronize with upstream**

$ git status

On branch master

Your branch is up-to-date with 'origin/master'.

2. new commits on upstream, origin without changes, push changes from upstream to your origin

**Update repository to synchronize with upstream**

$ git status

Your branch is ahead of 'origin/master' by 1 commit.

(use "git push" to publish your local commits)

$ git push origin master

3. new commits on upstream, new commits on origin. For a simplification imagine, you commited new file1 to origin, and created pull request. In between somebody commited file2 to upstream and your collegue also commented on you Pull Request Changes.You would like to correct your problems according to the comment with another commited file3 to origin. Now you would like to update your Pull Request

**Update repository to synchronize with upstream**

$ git status

Your branch and 'origin/master' have diverged,

and have 3 and 1 different commit each, respectively.

(use "git pull" to merge the remote branch into yours)

(this means upstream history -> file1 (one commit)

origin history -> file2 -> file1 ->fil3 (three commits)

**Resolving diverged branch merge**

1. Using patches

Export your commits as series of patches

$ git format-patch -M @{upstream}

will produce 3 patch files -- one for each of your commited files file1, file2 and file3

0001\_file2.patch

0002\_file1.patch

0003\_file3.patch

Reset your origin to match upstream

git reset --hard @{upstream}

(this means origin history -> file1

Now you can apply patches to update your origin to use your last commits (file3) or use changes from others that you are dependent on (file2)

$ git am 0003\_file3.patch

(this means origin history -> file1 -> file3

You can update your files with adding patches and removing latest n commits like

$ git am 0001\_file2.patch

(this means origin history -> file1 -> file3 -> file2

After doing your work, when you are prepared to commit again and make your pull request do not forget to remove the commits that are not your own (from your collegues), in our case simply apply reset to last commit and push your changes to origin

$ git reset --soft HEAD~1 (to remove 0001\_file2.patch)

(this means origin history -> file1 -> file3 before merge)

$ git push origin master

Create Pull Request

2. Using cherry-picking

Create a separate branch but stay in master

$ git branch file

Reset your origin master to the upstream master

$ git reset --hard @{upstream}

Cherry pick the commits you need from file branch to your master branch, or reset them

$ git cherry-pick file~1 git cherry-pick file~3 git cherry-pick file~2 …

When you are ready to update your Pull Request

$ git reset --soft HEAD~1

(this means history -> file1 -> file3)

$ git push origin master

**Reset last commit on your repo, clearing index files and working tree, no unsaved changes restored**

$ git reset --hard HEAD~1

$git push origin master -f (force)