**Git Workflow Digital Solutions**

**According to this post .. https://gist.github.com/blackfalcon/8428401**

**Clone repo**

$ git clone <https://github.com/User/repo.git>

**Sync update changes on develop**

$ git pull origin develop

**Create new feature branch**

$ git checkout -b my\_feature

**Change to feature branch and make changes**

$ git checkout my\_feature

Stage changes one by one $ git add .. $ git rm ..

or all of them at once $ git add --all

Remove all $ git rm -u

**Commit changes**

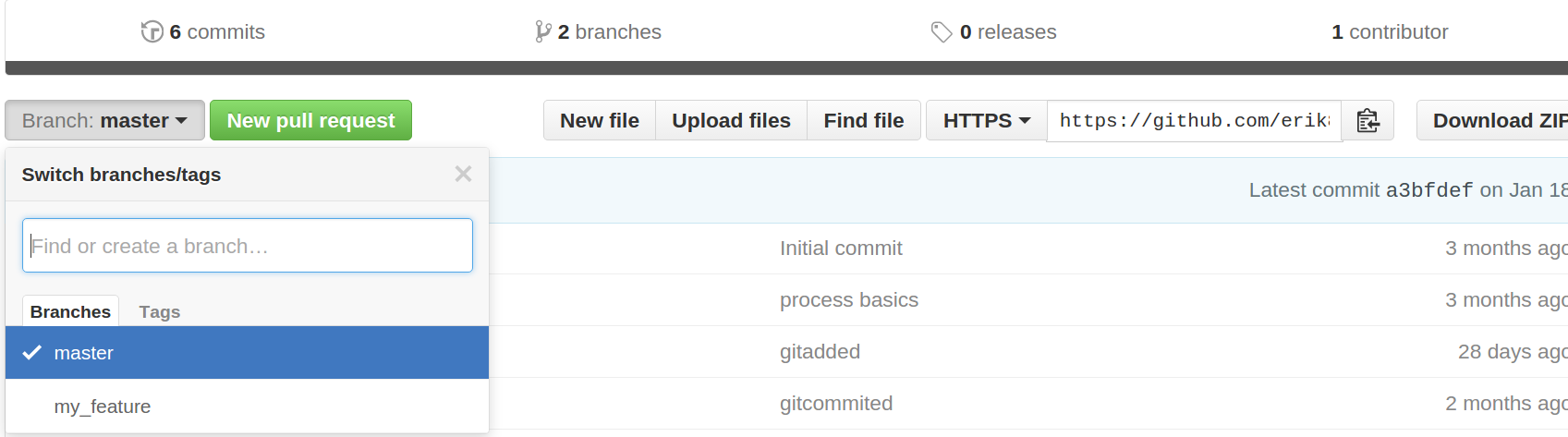
$ git commit -m “Changes in my\_feature”

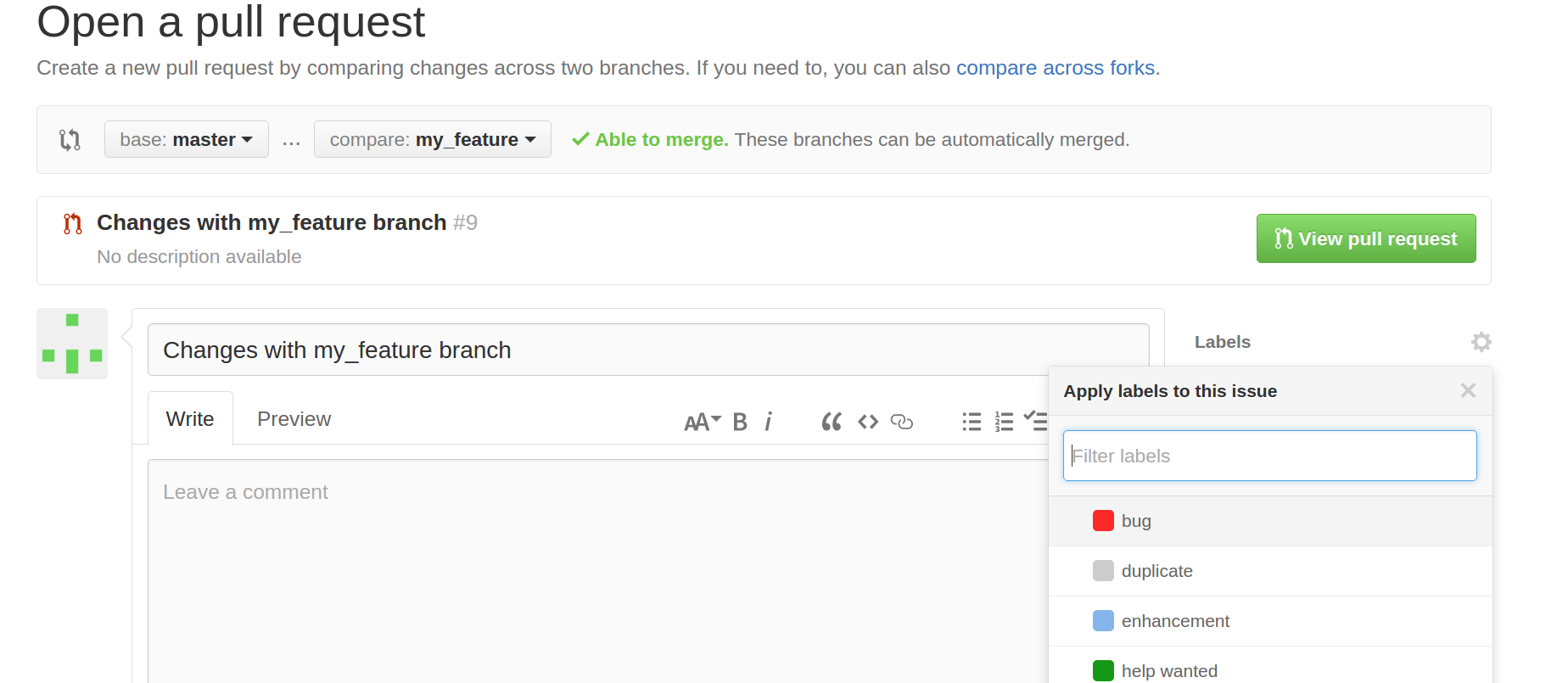
**Create pull requests for peer reviews from your collegues**

**Push your branch to github**

$ git push origin my-new-feature-branch

Change to your branch, Create direct Pull Request, and choose Label, and it is ready to go!





**Rebase 3 last commits on my\_feature branch when pushed**

$ git rebase -i origin/my\_feature~3 my\_feature

Use squash or fixup for commits you want to merge, fixup is the same as squash melding into previous commit but fixup also discards commit’s log messages

$ git push origin +my\_feature

**Test collegues PR in your repo, PR in branch my\_feature**

$ git clone <https://github.com/user/Repo.git>

$ git branch

\*master

$git checkout my\_feature

// Test the changes

**To show new branches and change to them**

$ git branch -a // shows all branches

$ git remote update // updates remote with all branches that were not added before

**To delete your local git branch**

$ git branch -d the\_local\_branch

**To delete branch remotedly**

$ git push origin --delete the\_local\_branch

Your changes and commits automatically propagate among your branches in local git

**The solution is to stash the changes**

$git branch

master

\*new\_branch

$ git stash

$ git stash list

stash@{0}: WIP on new\_branch: 121899a filemaster

$ git stash apply stash@{0}

To delete drop stash

$ git stash drop stash@{0}

Show stash diff

git stash show -p stash@{0}

**You have two branches, master and my\_feature, you updated my feature with your colleague and master has been updated during the process, 3 new commits on my\_feature and 14 new on master**

C - - -

C - - - - - - - - - - - - -

$ git branch

\*master

\*my\_feature

First update changes on both to see all 3 and 14 commits

$ git checkout my\_feature

$ git pull origin my\_feature

$ git checkout master

$ git pull origin master

Go to my\_feature and rebase on top of master

$ git checkout my\_feature

Interactivile squash or fixup commit comments to one

C-------------- ---

$ git rebase -i master

Go to master and merge your branch to it

$ git checkout master

$ git merge my\_feature

Update master

$ git push origin master

When going to my\_feature ??

$ git checkout my\_feature

$ git status

Switched to branch 'notify-feature'

Your branch and 'origin/notify-feature' have diverged,

and have 15 and 1 different commit each, respectively.

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

**Git differences**

Before add

git diff

After commit

git show

After add

git diff --staged

**Ignore tests in git**

git commit -n -m "Message"

-n option ignores the tests that will not run

**Merging branch to develop**

git pull

git checkout CIR-65-Adding-interstitial-confirmation-page

git rebase -i origin/develop

Squashed all commits but the first. (Since the full diff is fairly small it seemed like a single commit was appropriate)

git checkout develop

git merge CIR-65-Adding-interstitial-confirmation-page

git push

**Updates everything and all branches in git**

Git pull

**Reflog reset back all rebase**

Reflog contains all references to git commands

git reflog

git reset --hard HEAD@{1}

git push origin my\_feature