[Cannot push to GitHub - keeps saying need merge](https://stackoverflow.com/questions/10298291/cannot-push-to-github-keeps-saying-need-merge)

1.

This normally happens when you git commit and try to git push changes before git pulling on that branch x where someone else has already made changes.

**The normal flow would be as below,**

**STEP 1**: git stash your local uncommitted changes on that branch.

**STEP 2**: git pull origin branch\_name -v to pull and merge to locally committed changes on that branch (give this merge some message, and fix conflicts if any.)

**STEP 3**: git stash pop the stashed changes (Then you can make commits on popped files if you want or push already committed changes (STEP4) first and make new commit to files later.)

**STEP 4**: git push origin branch\_name -v the merged changes.

Replace *branch\_name* with *master* (for *master* branch).

2.

Some of you may be getting this error because Git doesn't know which branch you're trying to push.

If your error message also includes

error: failed to push some refs to 'git@github.com:jkubicek/my\_proj.git'

hint: Updates were rejected because a pushed branch tip is behind its remote

hint: counterpart. If you did not intend to push that branch, you may want to

hint: specify branches to push or set the 'push.default' configuration

hint: variable to 'current' or 'upstream' to push only the current branch.

then you may want to follow the handy tips from Jim Kubicek, [*Configure Git to Only Push Current Branch*](http://jimkubicek.com/blog/2012/08/16/git-push-current-branch/), to set the default branch to current.

git config --global push.default current

3.

In addition to the answers above, the following worked for me : -

**Scenario -**

1. I pushed **my\_branch** to origin successfully.
2. I made few more changes.
3. When I tried to push again, (after doing add, commit of course), I got the above mentioned error.

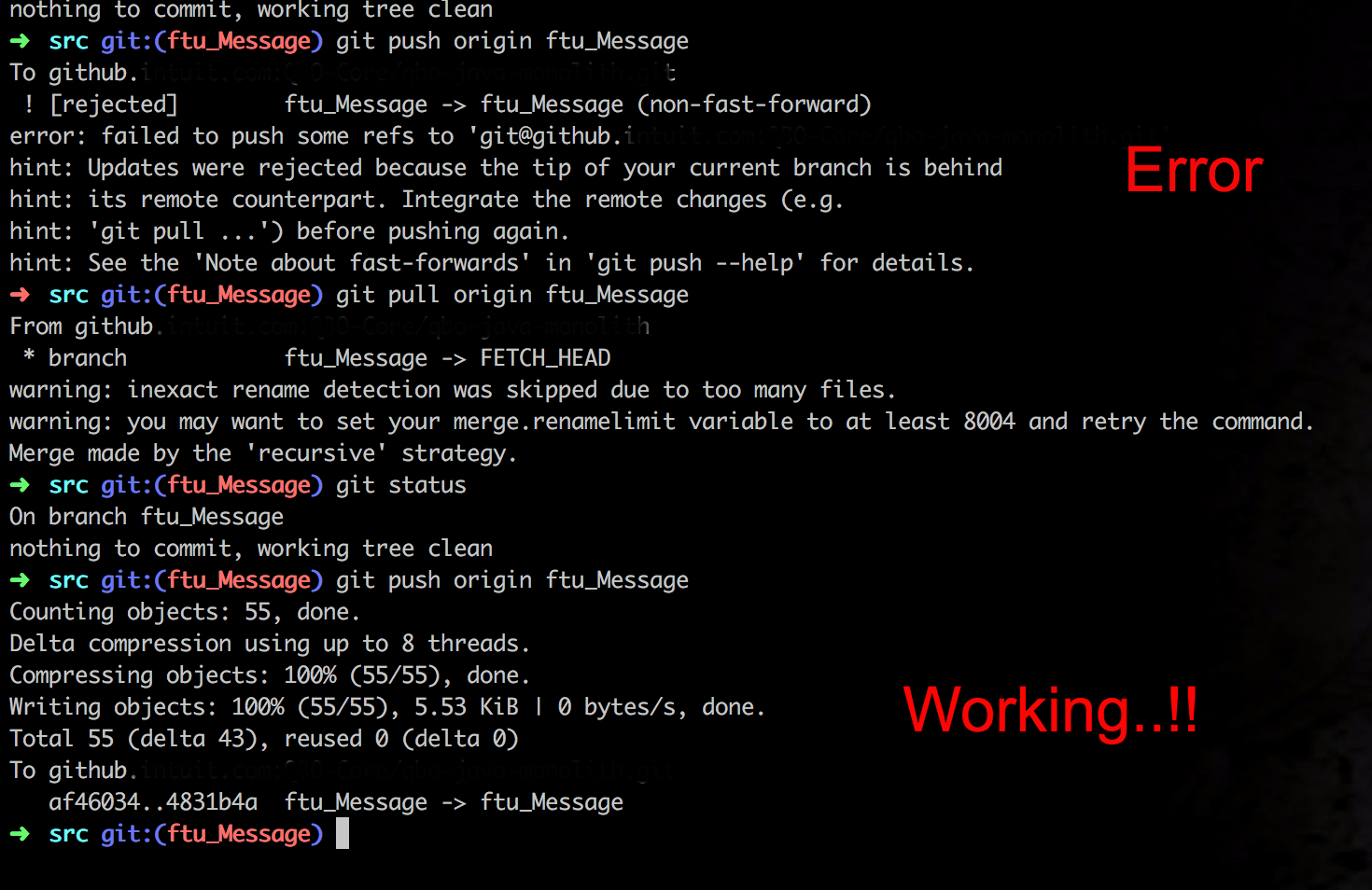
**Solution -**

1. git checkout \*\*my\_branch\*\*

2. git add, commit your changes.

3. git pull origin \*\*my\_branch\*\* (not origin, master, or develop)

4. git push origin \*\*my\_branch\*\*

[](https://i.stack.imgur.com/H8vNX.png)

4.

I had the same problem , what I did was I first pushed it by force by using this

git push --force

I did this after I commited the files and was getting an error as you got.It did commit all the files and it pushed them. Then the next time I was pushing to the github .I did what it asked me to and it was alright then. Hope this works for you too :)

5.

git pull origin branch\_name --rebase

This worked for me -- the command git pull origin branch\_name --rebase will pull changes from remote branch\_name at first, then rebase current branch on the top of it.

6.

When you create a new repository on GitHub, GitHub may ask you to create a readme file. If you create a readme file directly on GitHub, then you will need to first make a ‘pull’ request before the ‘push’ request will be successful. These commands will ‘pull’ the remote repository, merge it with your current files, and then ‘push’ all the files back to GitHub:

git pull https://github.com/thomas07vt/MyFirstRepo.git master

git push https://github.com/thomas07vt/MyFirstRepo.git master

7.

git push -f origin **branchname**

Use the above command only if you are sure that you don't need remote branch code otherwise do merge first and then push the code

8.

Is your branch name the same as the remote branch name?

If no, you should checkout a new branch with the same name as the remote branch and try push it again.

Assume the remote branch you want to push is [*testing*] and your local branch is named as [*test*].

If you`re not in *test* branch, first switch to it.

git checkout test

Then open a new branch and name it *testing*.

git checkout -b testing

Now, it`s time to push it:

git push [remote repo] testing

9.

I had a similar issue and it turned out that my workflow for keeping my branch up to date was at fault. I was doing the following:

**In my local 'master'**

git fetch upstream

git merge upstream/master --ff-only

**then back in my local branch**

git rebase master

This worked well for a previous git flow but not with github. The git rebase was the problem here causing issues with syncing (and I'll admit that's something I've had to accept without fully understanding) and unfortunately put me in a position where git push -f became probably the easiest option. Not good.

My new flow is to update the branch directly using git merge as follows:

**In my local branch**

git fetch upstream

git merge upstream/master

No fast forward, as I will have made changes of course in the local branch.

As you can probably tell, I'm no git expert but I'm reliably informed that this workflow will probably avoid the specific problems that I had.

10.

I have resolve this issue at my GIT repository. No need to rebase or force commit in this case. Use below steps to resolve this -

local\_barnch> git branch --set-upstream to=origin/<local\_branch\_name>

local\_barnch>git pull origin <local\_branch\_name>

local\_barnch> git branch --set-upstream to=origin/master

local\_barnch>git push origin <local\_branch\_name>

hope it will help.

11.

use git pull https://github.com/username/repository It's because the Github and remote repositories aren't in sync. If you pull the repo and then Push everything will be in sync and error will go away.

12.

If by any chance git pull prints Already up-to-date then you might want to check the global git push.default param (In ~/.gitconfig). Set it to simple if it was in matching. The below answer explains why:

[Git - What is the difference between push.default "matching" and "simple"](https://stackoverflow.com/questions/21839651/git-what-is-the-difference-between-push-default-matching-and-simple)

Also, it is worth checking if your local branch is out of date using git remote show origin and do a pull if needed

**Git - What is the difference between push.default “matching” and “simple”**

git push can push all branches or a single one dependent on this configuration:

**Push all branches**

git config --global push.default matching

It will push all the branches to the remote branch and would merge them. If you don't want to push all branches, you can push the current branch only.

**Push only the current branch**

git config --global push.default simple

So, it's better, in my opinion, to use this option and push your code branch by branch. It's better to push branches manually and individually.

Below gives the full information. In short, simple will only push the current working branch and even then only if it also has the same name on the remote. This is a very good setting for beginners and will become the default in GIT 2.0

Whereas matching will push all branches locally that have the same name on the remote. (Without regard to your current working branch ). This means potentially many different branches will be pushed, including those that you might not even want to share.

In my personal usage, I generally use a different option: current which pushes the current working branch, (because I always branch for any changes). But for a beginner I'd suggest simple

push.default  
Defines the action git push should take if no refspec is explicitly given. Different values are well-suited for specific workflows; for instance, in a purely central workflow (i.e. the fetch source is equal to the push destination), upstream is probably what you want. Possible values are:

nothing - do not push anything (error out) unless a refspec is explicitly given. This is primarily meant for people who want to avoid mistakes by always being explicit.

current - push the current branch to update a branch with the same name on the receiving end. Works in both central and non-central workflows.

upstream - push the current branch back to the branch whose changes are usually integrated into the current branch (which is called @{upstream}). This mode only makes sense if you are pushing to the same repository you would normally pull from (i.e. central workflow).

simple - in centralized workflow, work like upstream with an added safety to refuse to push if the upstream branch's name is different from the local one.

When pushing to a remote that is different from the remote you normally pull from, work as current. This is the safest option and is suited for beginners.

This mode will become the default in Git 2.0.

matching - push all branches having the same name on both ends. This makes the repository you are pushing to remember the set of branches that will be pushed out (e.g. if you always push maint and master there and no other branches, the repository you push to will have these two branches, and your local maint and master will be pushed there).

To use this mode effectively, you have to make sure all the branches you would push out are ready to be pushed out before running git push, as the whole point of this mode is to allow you to push all of the branches in one go. If you usually finish work on only one branch and push out the result, while other branches are unfinished, this mode is not for you. Also this mode is not suitable for pushing into a shared central repository, as other people may add new branches there, or update the tip of existing branches outside your control.

This is currently the default, but Git 2.0 will change the default to simple.

**Git – What is the difference between push.default “matching” and “simple”**

Struggling with git push.default, not sure what is the difference between push.defaut “matching” and “simple”. Hope this blog will helps you out!

Lets start with what is git push?

git push - can push all branches or a single one dependent on this configuration:

**1.Configuration 1 – Push all branches**

git config --global push.default matching

It will push all the branches to the remote branch and would merge them. If you don’t want to push all branches, you can push the current branch only.

**2. Configuration 2: Push only the current branch**

git config --global push.default simple

This will push branches one by one. It Mostly connected with current branch.

It will push all the branches to the remote branch and would merge them. If you don't want to push all branches, you can push the current branch only.

**push.default**

Defines the action git push should take if no refspec is explicitly given. Possible values are:

1. nothing – do not push anything (error out) unless a refspec is explicitly given. This is primarily meant for people who want to avoid mistakes by always being explicit.
2. current – push the current branch to update a branch with the same name on the receiving end. Works in both central and non-central workflows.
3. upstream – push the current branch back to the branch whose changes are usually integrated into the current branch (which is called @{upstream}). This mode only makes sense if you are pushing to the same repository you would normally pull from (i.e. central workflow).
4. simple – in centralized workflow, work like upstream with an added safety to refuse to push if the upstream branch’s name is different from the local one.
5. matching – push all branches having the same name on both ends. This makes the repository you are pushing to remember the set of branches that will be pushed out.