How do I update a GitHub forked repository?

3204

1718

I recently forked a project and applied several fixes. I then created a pull request which was then accepted.

A few days later another change was made by another contributor. So my fork doesn't contain that change.

How can I get that change into my fork? Do I need to delete and re-create my fork when I have further changes to contribute? Or is there an update button?

git github

edited Apr 27 '18 at 14:54

nbro
6,008 10 53 102

asked Aug 30 '11 at 13:53

Lea Hayes
24.8k 14 47 97

105 This can also be done from the github UI. I'd like to give credit [to this other poster][1]. [1]: stackoverflow.com/a/21131381/728141 – Mike Schroll Feb 20 '14 at 13:00

2 Another good blog post on this - Keeping A GitHub Fork Updated – Arup Rakshit Oct 15 '14 at 17:26

3 Found this in Github help articles: help.github.com/articles/syncing-a-fork – Pranav Apr 2 '15 at 8:57

2 Is this a duplicate of stackoverflow.com/questions/3903817/… ? – David Cary Aug 29 '15 at 12:06

Here's a video demo that does this using two github accounts youtube.com/watch?v=kpE0gTX4ycE – lifebalance Jun 3 '16 at 9:29

18 Answers

3501

In your local clone of your forked repository, you can add the original GitHub repository as a "remote". ("Remotes" are like nicknames for the URLs of repositories - `origin` is one, for example.) Then you can fetch all the branches from that upstream repository, and rebase your work to continue working on the upstream version. In terms of commands that might look like:

```
# Add the remote, call it "upstream":

git remote add upstream https://github.com/whoever/whatever.git

# Fetch all the branches of that remote into remote-tracking branches,
# such as upstream/master:

git fetch upstream

# Make sure that you're on your master branch:
```

```
# other branch:

git rebase upstream/master
```

If you don't want to rewrite the history of your master branch, (for example because other people may have cloned it) then you should replace the last command with `git merge upstream/master`. However, for making further pull requests that are as clean as possible, it's probably better to rebase.

If you've rebased your branch onto `upstream/master` you may need to force the push in order to push it to your own forked repository on GitHub. You'd do that with:

```
git push -f origin master
```

You only need to use the `-f` the first time after you've rebased.

edited Apr 9 '18 at 11:19

NearHuscarl
65 1 2 8

answered Aug 30 '11 at 14:01

Mark Longair
306k 62 359 304

85 As your fork only exists on github, and github does not have tools for doing merges through the web interface, then the right answer is to do the upstream merge locally and push the changes back to your fork. – Tim Keating Jun 19 '12 at 3:50

27 Here is a great tutorial I found on working with github: gun.io/blog/how-to-github-fork-branch-and-pull-request – Tim Keating Jun 19 '12 at 3:55

42 A quick note that rather than having to rebase your own master branch to ensure you are starting with clean state, you should probably work on a separate branch and make a pull request from that. This keeps your master clean for any future merges and it stops you from having to rewrite history with `-f` which messes up everyone that could have cloned your version. – Mateusz Kowalczyk May 29 '13 at 23:09

7 Instead of the rebase command, i used the following: `git merge --no-ff upstream/master` This way your commits aren't on top anymore. – Steckdoserich Oct 17 '16 at 13:20

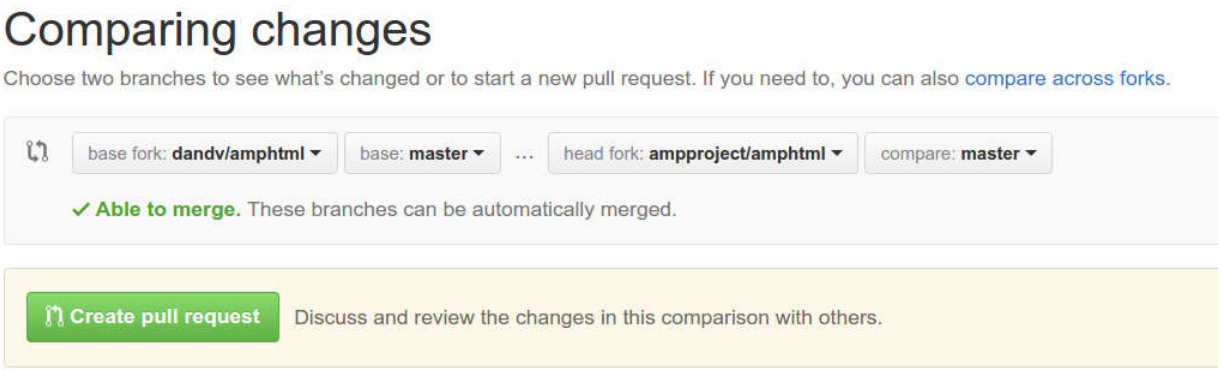
28 Another Git failure. If this tools is supposed to support distributed collaboration, then why is it so difficult to perform a basic workflow? 4 million people and 2200 upvotes mean the tool failed. *"you can add the original GitHub repository as a "remote"* - Why does one even have to do this? Why is it not done during the fork? What is so broken about this tool? – jww Apr 16 '17 at 16:05

689

Starting in May 2014, it is possible to update a fork directly from GitHub. This still works as of September 2017, **BUT** it will lead to a dirty commit history.

1. Open your fork on GitHub.
2. Click on Pull Requests.
3. Click on New Pull Request. By default, GitHub will compare the original with your fork, and there shouldn't be anything to compare if you didn't make any changes.

original, and you should see all the latest changes.



5. Create pull request and assign a predictable name to your pull request (e.g., `Update from original`).
6. Scroll down to Merge pull request, but don't click anything yet.

Now you have three options, but each will lead to a less-than-clean commit history.

1. The default will create an ugly merge commit.
2. If you click the dropdown and choose "Squash and merge", all intervening commits will be squashed into one. This is most often something you don't want.
3. If you click Rebase and merge, all commits will be made "with" you, the original PRs will link to your PR, and GitHub will display `This branch is X commits ahead, Y commits behind <original fork>`.

So yes, you can keep your repo updated with its upstream using the GitHub web UI, but doing so will sully your commit history. Stick to the command line instead - it's easy.

edited Sep 26 '17 at 20:41 — MD XF — 4,332 5 29 55

answered May 25 '14 at 7:31 — lobzik — 8,911 1 22 28

18 This worked great one time. The second time this process did not work the same way: the "Switching the base" link did not show up. And when I hit "Click to create a pull request" it created a PR on the SOURCE repo. NOT what I wanted.. – javadba Aug 21 '14 at 18:14

27 Still works (Marchi 2015), all though the "Switching the base" link is no longer there. You have to change the "Base" drop down's so both point to your fork and then you'll get a prompt to "Compare across repos", which will take you to where you want. – mluisbrown Mar 4 '15 at 14:05

7 April 2015. Works. Thanks. I did get "Switching to base". However, step 6 was "Create pull request" -> enter comment -> "Create pull request". End up with 1 commit ahead of original. – cartland Apr 9 '15 at 0:08

4 @cartland (or others) - yes, it says "This branch is 1 commit ahead of ..." Is this something to worry about? Is it possible to get rid of that message? – RenniePet May 15 '15 at 22:59

6 wouldnt it be better, with a simply update or sync button! – transformer Jan 24 '17 at 3:32

Syncing a fork

The Setup

Before you can sync, you need to add a remote that points to the upstream repository. You may have done this when you originally forked.

Tip: Syncing your fork only updates your local copy of the repository; it does not update your repository on GitHub.

```
$ git remote -v
# List the current remotes
origin  https://github.com/user/repo.git (fetch)
origin  https://github.com/user/repo.git (push)

$ git remote add upstream https://github.com/otheruser/repo.git
# Set a new remote

$ git remote -v
# Verify new remote
origin    https://github.com/user/repo.git (fetch)
origin    https://github.com/user/repo.git (push)
upstream  https://github.com/otheruser/repo.git (fetch)
upstream  https://github.com/otheruser/repo.git (push)
```

Syncing

There are two steps required to sync your repository with the upstream: first you must fetch from the remote, then you must merge the desired branch into your local branch.

Fetching

Fetching from the remote repository will bring in its branches and their respective commits. These are stored in your local repository under special branches.

```
$ git fetch upstream
# Grab the upstream remote's branches
remote: Counting objects: 75, done.
remote: Compressing objects: 100% (53/53), done.
remote: Total 62 (delta 27), reused 44 (delta 9)
Unpacking objects: 100% (62/62), done.
From https://github.com/otheruser/repo
 * [new branch]      master     -> upstream/master
```

We now have the upstream's master branch stored in a local branch, upstream/master

```
$ git branch -va
# List all local and remote-tracking branches
* master                  a422352 My local commit
  remotes/origin/HEAD     -> origin/master
```

Now that we have fetched the upstream repository, we want to merge its changes into our local branch. This will bring that branch into sync with the upstream, without losing our local changes.

```
$ git checkout master
# Check out our local master branch
Switched to branch 'master'

$ git merge upstream/master
# Merge upstream's master into our own
Updating a422352..5fdff0f
Fast-forward
 README                    |    9 -------
 README.md                 |    7 ++++++
 2 files changed, 7 insertions(+), 9 deletions(-)
 delete mode 100644 README
 create mode 100644 README.md
```

If your local branch didn't have any unique commits, git will instead perform a "fast-forward":

```
$ git merge upstream/master
Updating 34e91da..16c56ad
Fast-forward
 README.md                 |    5 +++--
 1 file changed, 3 insertions(+), 2 deletions(-)
```

Tip: If you want to update your repository on GitHub, follow the instructions here

edited Sep 21 '15 at 19:38 — Peter Mortensen — 14.2k — 19 — 88 — 114

answered Oct 21 '13 at 23:04 — jumpnett — 4,867 — 1 — 14 — 22

1 This updates my local fork, but my fork on Github.com still says "43 commits behind". I had to use lobzik's technique to create a pull request for myself to merge the master changes into my Github.com fork. – Michael McGinnis Jan 23 '15 at 17:38

9 @MichaelMcGinnis After merging locally, you would have to push your changes to github. `git push origin master` – jumpnett Feb 11 '15 at 22:50

1 Might be smart to push with `--follow-tags` : stackoverflow.com/a/26438076/667847 – kenny Nov 6 '15 at 15:19

I have to do it for all branches separately `git merge upstream/master` , then check out to develop branch and do `git merge upstream/develop` – Shobi May 28 '17 at 13:02

stackoverflow.com/a/14074925/470749 was helpful to me because I was getting `Permission denied (publickey). fatal: Could not read from remote repository.` when trying to fetch from Facebook's Github account upstream. – Ryan Jan 26 '18 at 4:28

A lot of answers end up moving your fork **one commit ahead** of the parent repository. This answer summarizes the steps found here which will **move your fork to the same commit as the parent.**

2. Add the parent as a remote repository, `git remote add upstream <repo-location>`
3. Issue `git fetch upstream`
4. Issue `git rebase upstream/master`
    - At this stage you check that commits what will be merged by typing `git status`
5. Issue `git push origin master`

For more information about these commands, refer to step 3.

edited Apr 19 '16 at 4:49 — Peter Mortensen — 14.2k 19 88 114

answered Aug 5 '15 at 14:59 — Sahar Rabinoviz — 1,333 12 20

12 @MT: Where do you enter these commands, though? The gist of the question, as I understand it, is how to resynchronize your personal *GitHub* fork with the main project, and **do this all from GitHub**. In other words, how can you update your remote fork *without* a local repository? – John Y May 16 '16 at 15:33

3 @JohnY Using GitHub will always create an extra commit. You need to do all this in a shell on a local repo to avoid that extra commit. – Jonathan Cross Oct 14 '16 at 21:51

43

Since November 2013 there has been an unofficial feature request open with GitHub to ask them to add a very simple and intuitive method to keep a local fork in sync with upstream:

https://github.com/isaacs/github/issues/121

Note: Since the feature request is unofficial it is also advisable to contact `support@github.com` to add your support for a feature like this to be implemented. The unofficial feature request above could be used as evidence of the amount of interest in this being implemented.

edited Apr 19 '16 at 4:50 — Peter Mortensen — 14.2k 19 88 114

answered Feb 21 '16 at 10:42 — isedwards — 1,335 12 22

40

Foreword: Your fork is the "origin" and the repository you forked from is the "upstream".

Let's assume that you cloned already your fork to your computer with a command like this:

```
git clone git@github.com:your_name/project_name.git
cd project_name
```

If that is given then you need to continue in this order:

1. Add the "upstream" to your cloned repository ("origin"):

```
git remote add upstream git@github.com:original_author/project_name.git
```

3. Switch to the "master" branch of your fork ("origin"):

```
git checkout master
```

4. Stash the changes of your "master" branch:

```
git stash
```

5. Merge the changes from the "master" branch of the "upstream" into your the "master" branch of your "origin":

```
git merge upstream/master
```

6. Resolve merge conflicts if any and commit your merge

```
git commit -am "Merged from upstream"
```

7. Push the changes to your fork

```
git push
```

8. Get back your stashed changes (if any)

```
git stash pop
```

9. You're done! Congratulations!

GitHub also provides instructions for this topic: Syncing a fork

edited Dec 9 '16 at 13:53

answered Mar 16 '16 at 12:24

Benny Neugebauer
29.4k 17 155 157

Helped partly: Is `git remote add upstream git@github.com:original_author/project_name.git` just an alias for `git remote add upstream https://github.com/original_author/project_name.git` ? – Wolf Jun 26 '17 at 14:44

1 Wolf, guessing you know this by now, but for posterity... It is the format for ssh. help.github.com/articles/configuring-a-remote-for-a-fork – Brad Ellis Jan 26 '18 at 21:02

1 Thank you very much. `git stash` and `git stash pop` part very helpful – krishna Mar 4 at 5:41

37

If, like me, you **never commit anything directly to master**, which you should really, you can do the following.

From the local clone of your fork, create your upstream remote. You only need to do that once:

Then whenever you want to catch up with the upstream repository master branch you need to:

```
git checkout master
git pull upstream master
```

Assuming you never committed anything on master yourself you should be done already. Now you can push your local master to your origin remote GitHub fork. You could also rebase your development branch on your now up-to-date local master.

Past the initial upstream setup and master checkout, all you need to do is run the following command to sync your master with upstream: **git pull upstream master**.

edited May 18 at 16:55

answered Jan 3 '17 at 16:59

Slion
725 8 15

22

As of the date of this answer, GitHub has not (or shall I say no longer?) this feature in the web interface. You can, however, ask `support@github.com` to add your vote for that.

In the meantime, GitHub user bardiharborow has created a tool to do just this:
https://upriver.github.io/

Source is here: https://github.com/upriver/upriver.github.io

answered Sep 14 '16 at 14:22

Lucero
52.7k 6 95 142

1 While I do find the tool a good idea the reality is that's BROKEN. It did load only 20 repos from my account and even the footer redirects to a website that does not exists. If that's fixed I will be a big advocate. – sorin Oct 28 '16 at 16:07

1 As of today, I have successfully used upriver to sync a fork with the upstream repo, so it's working for my purposes and I will continue to use it. – NauticalMile Jul 17 '17 at 2:55

@sorin These 20 repo/branch limitation (rather, it is 30 now) comes from the GitHub default paging settings. There needs to be some adaptions to the code in order to handle this. – Andreas Jan 18 '18 at 8:19

15

If you are using GitHub for Windows then now they have a one-click feature to update forks:

1. Select the repository in the UI.
2. Click "Update from user/branch" button the top.

edited Jan 25 '18 at 7:06

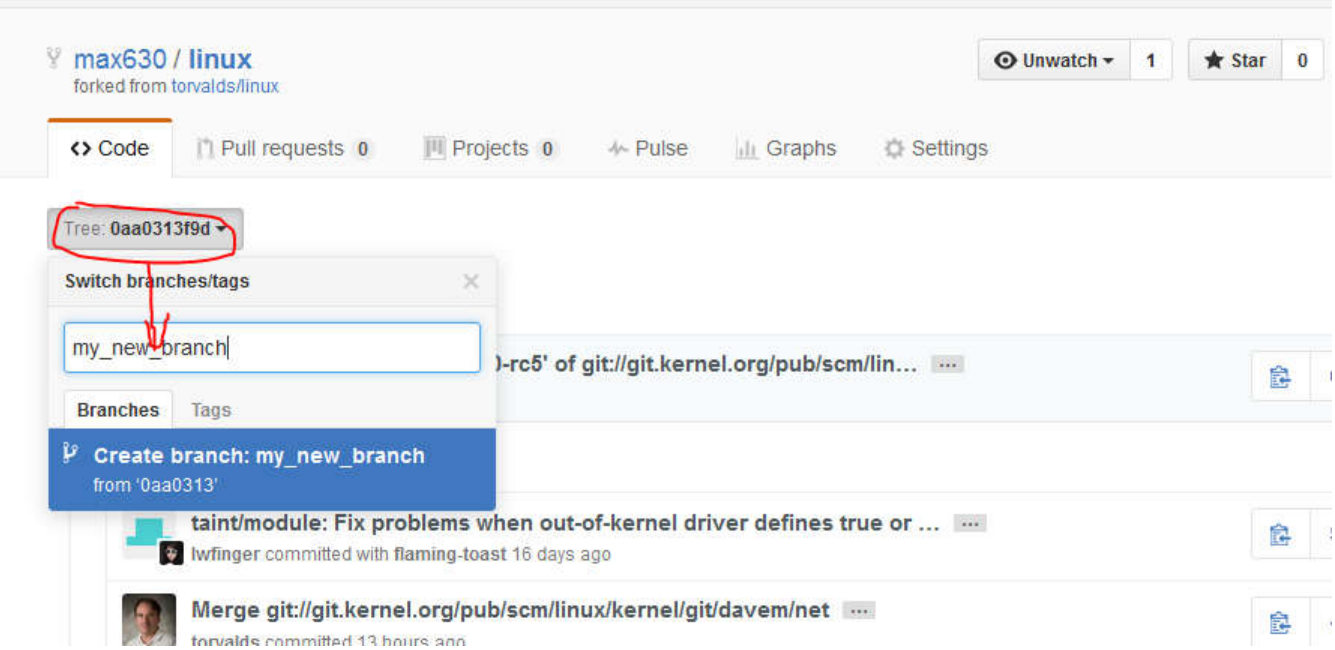
answered Mar 31 '16 at 21:45

Shital Shah
27.8k 6 117 101

9

Actually, it is possible to create a branch in your fork from any commit of the upstream in the browser:

- Open `https://github.com/<repo>/commits/<hash>` , where *repo* is your fork, and *hash* is full hash of commit which you can find in the upstream web interface. For example, I can open https://github.com/max630/linux/commits/0aa0313f9d576affd7747cc3f179feb097d28990, which points to `linux master` as time of writing.
- Click on the "Tree: ...." button.
- Type name of the new branch and press Enter



You can then fetch that branch to your local clone, and you won't have to push all that data back to GitHub when you push edits on top of that commit. Or use the web interface to change something in that branch.

How it works (it is a guess, I don't know how exactly GitHub does it): forks share object storage and use namespaces to separate users' references. So you can access all commits through your fork, even if they did not exist by the time of forking.

edited Jan 19 '18 at 3:22
Peter Mortensen
14.2k 19 88 114

answered Jan 18 '17 at 6:41
max630
5,587 1 15 42

1 This is great! This avoids the totally pointless upload of those commits to github. – Rotsor Mar 13 '17 at 3:34

Follow the below steps. I tried them and it helped me.

> **Syntax:** git branch yourDevelopmentBranch
> **Example:** git checkout master

Pull source repository branch for getting the latest code

> **Syntax:** git pull https://github.com/tastejs/awesome-app-ideas master
> **Example:** git pull https://github.com/ORIGINAL_OWNER/ORIGINAL_REPO.git BRANCH_NAME

edited Apr 19 '16 at 4:49

Peter Mortensen
14.2k 19 88 114

answered Jan 15 '16 at 12:31

Venkat.R
4,903 3 30 47

If you're using GitHub, you might also want to push your changes to your GitHub branch. `git push HttpsForYourForkOfTheRepo BRANCH_NAME` – user3731622 Jan 22 '16 at 19:34

5

I update my forked repos with this one line:

```
git pull https://github.com/forkuser/forkedrepo.git branch
```

Use this if you dont want to add another remote endpoint to your project, as other solutions posted here.

answered Sep 8 '17 at 2:00

R.Bravo
543 5 14

1 Are there limitations on this? i.e. does it apply only to cases where you have not added commits, merges, pull requests, or had pull requests merged into upstream since the last update? – LightCC Sep 11 '17 at 7:30

it does work like a normal pull from a remote branch. If you did X commits on your local repo and now you are Y commits behind the original repo, it will bring the Y commits to your local branch and, probably, get you some conflicts to resolve. – R.Bravo Sep 11 '17 at 20:23

@LightCC This is not different than pulling from a previously added remote at all, except for the fact that you haven't added a remote. So the disadvantage is that you'll have to enter the full repository URL everytime you want to `pull` . – Marc.2377 Apr 17 at 2:34

5

As a complement to this answer, I was looking for a way to update all remote branches of my cloned repo (*origin*) from *upstream* branches in one go. This is how I did it.

This assumes you have already configured an *upstream* remote pointing at the source repository (where *origin* was forked from) and have synced it with `git fetch upstream` .

```
for branch in $(git ls-remote --heads upstream|sed 's#^.*refs/heads/##'); do git push
origin refs/remotes/upstream/$branch:refs/heads/$branch; done
```

The first part of this command lists all heads in the *upstream* remote repo and removes the SHA-1 followed by `refs/heads/` branch name prefix.

Then for each of these branches, it pushes the local copy of the *upstream* remote tracking branch (`refs/remotes/upstream/<branch>` on local side) directly to the remote branch on *origin* (`refs/heads/<branch>` on remote side).

Any of these branch sync commands may fail for one of two reasons: either the *upstream* branch have been rewritten, or you have pushed commits on that branch to your fork. In the first case where you haven't committed anything to the branch on your fork it is safe to push forcefully (Add the **-f** switch; i.e. `git push -f` in the command above). In the other case this is normal as your fork branch have diverged and you can't expect the sync command to work until your commits have been merged back into *upstream*.

edited Jun 8 '18 at 4:31

answered Jun 12 '17 at 20:17

Thomas Guyot-Sionnest
996 10 9

3

Android Studio now has learned to work with GitHub fork repositories (you don't even have to add "upstream" remote repository by console command).

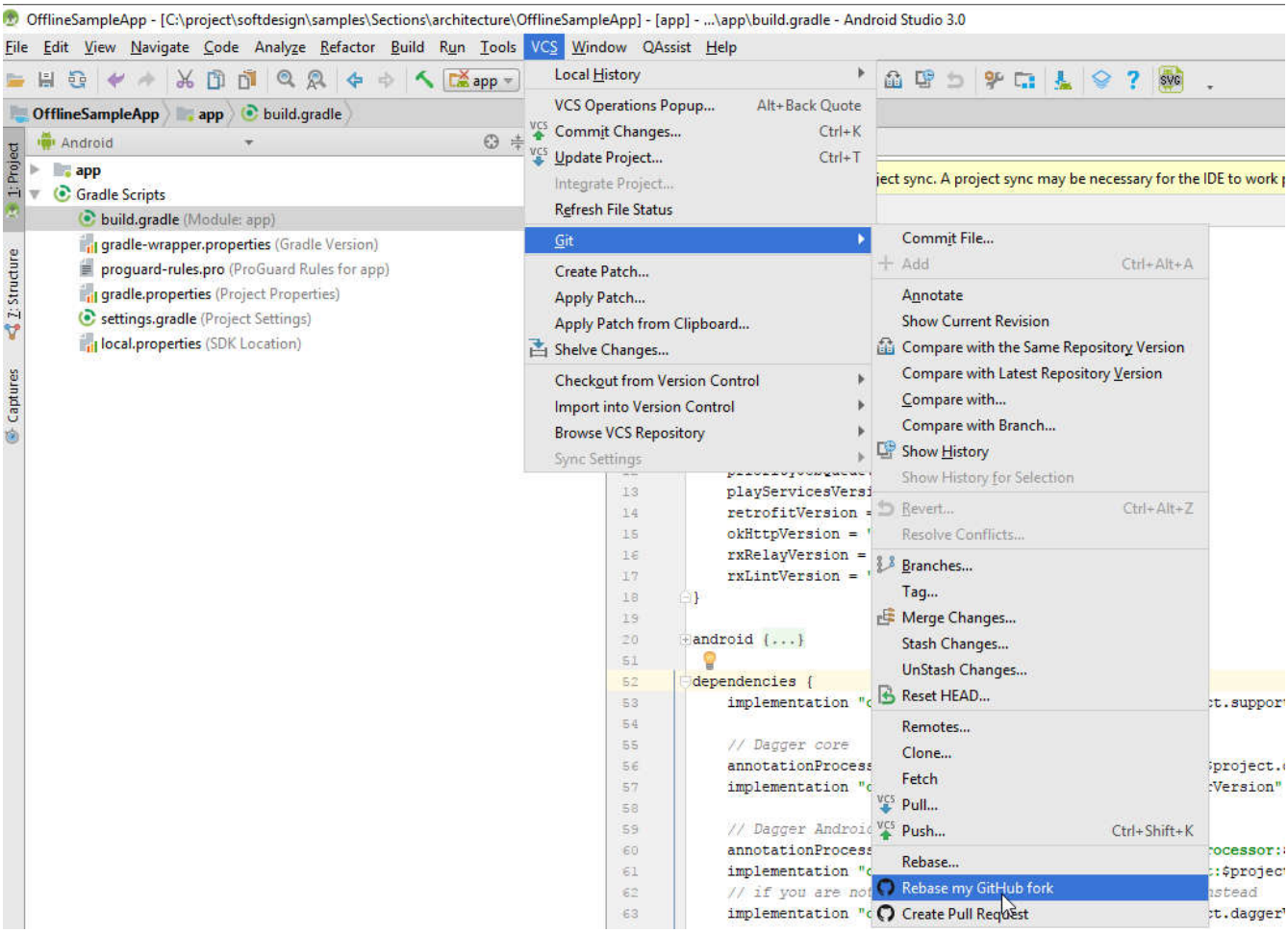
Open menu *VCS → Git*

And pay attention to the two last popup menu items:

- *Rebase my GitHub fork*
- *Create Pull Request*

Try them. I use the first one to synchronize my local repository. Anyway the branches from the parent remote repository ("upstream") will be accessible in Android Studio after you click "Rebase my GitHub fork", and you will be able to operate with them easily.

(I use Android Studio 3.0 with "Git integration" and "GitHub" plugins.)



edited Jan 19 '18 at 3:25

Peter Mortensen
14.2k 19 88 114

answered Nov 13 '17 at 20:03

alexshr
561 5 10

3

When you have cloned your forked repository, go to the directory path where your clone resides and the few lines in your Git Bash Terminal.

```
$ cd project-name

$ git remote add upstream https://github.com/user-name/project-name.git
 # Adding the upstream -> the main repo with which you wanna sync

$ git remote -v # you will see the upstream here

$ git checkout master # see if you are already on master branch

$ git fetch upstream
```

And there you are good to go. All updated changes in the main repository will be pushed into your fork repository.

answered Mar 10 '18 at 2:21

Prateek Chanda
111 2 2

1

That depends on the size of your repository and how you forked it.

If it's quite a big repository you may have wanted to manage it in a special way (e.g. drop history). Basically, you can get differences between current and upstream versions, commit them and then cherry pick back to master.

Try reading this one. It describes how to handle big Git repositories and how to upstream them with latest changes.

edited Nov 3 '18 at 19:35

itsmysterybox
1,244 3 10 21

answered Apr 23 '17 at 12:47

s0nicYouth
168 1 11

0

There are two main things on keeping a forked repository always update for good.

> **1. Create the branches** from the fork master and **do changes there**.

So when your *Pull Request* is accepted then you can safely delete the branch as your contributed code will be then live in your master of your forked repository when you update it with the upstream. By this your master will always be in clean condition to create a new branch to do another change.

> **2. Create a scheduled job** for the fork master to **do update automatically**.

This can be done with *cron*. Here is for an example code if you do it in linux.

```
$ crontab -e
```

put this code on the `crontab file` to execute the job in hourly basis.

```
0 * * * * sh ~/cron.sh
```

then create the `cron.sh` script file and a *git interaction* with *ssh-agent* and/or *expect* as below

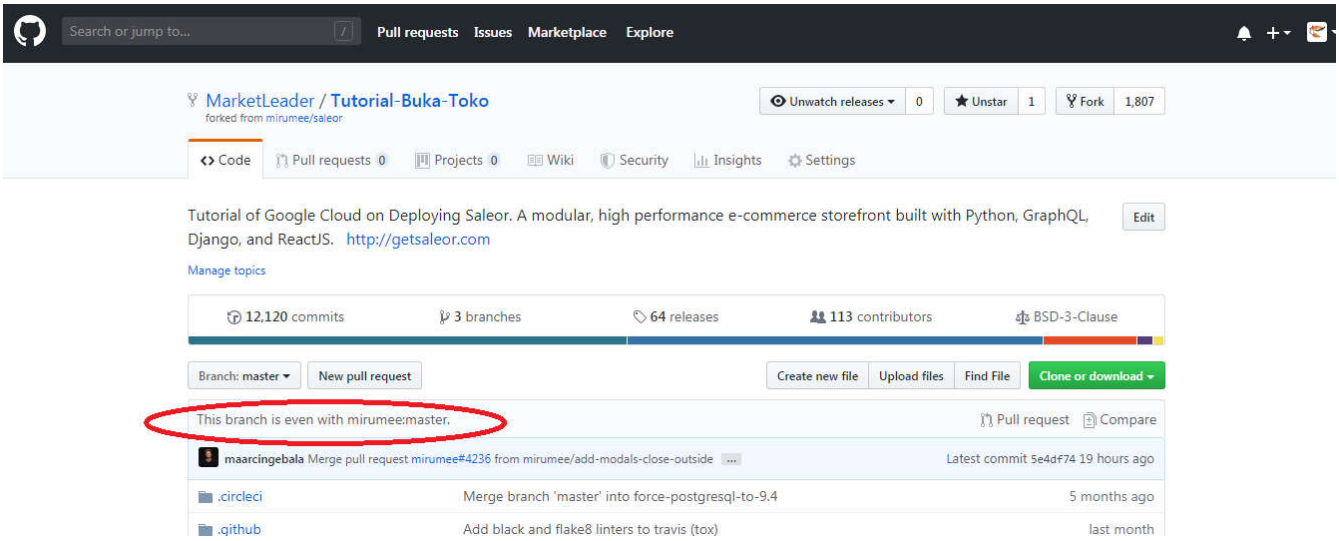
```
#!/bin/sh
WORKDIR=/path/to/your/dir
REPOSITORY=<name of your repo>
MASTER="git@github.com:<username>/$REPOSITORY.git"
UPSTREAM=git@github.com:<upstream>/<name of the repo>.git

cd $WORKDIR && rm -rf $REPOSITORY
```

```
then
    echo "all the same, do nothing"
else
    echo "update exist, do rebase!"
    git reset --hard upstream/master
    git push origin master --force
fi
cd $WORKDIR && rm -rf $REPOSITORY
eval `ssh-agent -k`
```

Check your forked repository. From time to time it will always show this notification:

> **This branch is even with `<upstream>` :master.**



edited Jun 8 at 13:34

answered Jun 8 at 7:33

Chetabahana
4,185 1 26 47

0

If you set your upstream. Check with `git remote -v` , then this will suffice.

```
git fetch upstream
git checkout master
git merge --no-edit upstream/master
git push
```

answered Jun 11 at 5:32

prosti
8,271 1 32 44

---

protected by Samuel Liew ♦ Oct 5 '15 at 9:18

Would you like to answer one of these unanswered questions instead?