Keeping your forked repo synced with the upstream source

Nitin Venkatesh's Gravatar

[Nitin Venkatesh](https://2buntu.com/accounts/8/nitin-venkatesh/)   
published 2 years ago

We code, we fork, we code again, submit pull requests and the cycle goes on and on. Welcome to the world of Open Source. You fork a project to add some upgrade or feature and submit a pull request. However, when you're new like me, one big question that arises is - how do we keep the forked repo up-to-date with the changes that happens in the original repo (upstream)?

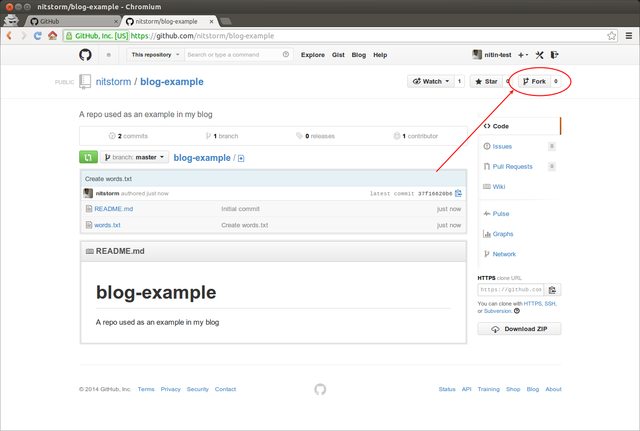
Let's go through this step-by-step:

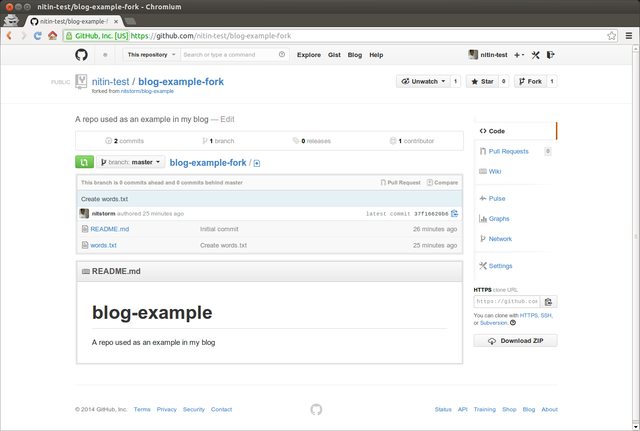
In our example, the main repo is called blog-example and is maintained by the user nitstorm. The forked repo is called blog-example-fork and belongs to the user nitin-test. We'll be doing the following in this post:

1. Fork a repo
2. Upgrade code and submit a pull request
3. Keep the forked repo synced with the main repo

Step 1: Forking a repo

Navigate to the project that you want to work on and click on the Fork button that you see.

[](http://i.imgur.com/7wapvt2.png)

[](http://i.imgur.com/aa5AOiy.png)

Now we need to clone the forked repo on our local machine to make the required changes. We can do that with git clone <repo-location>

nitin@jane-saucy:~$ git clone https://github.com/nitin-test/blog-example-fork.git

Cloning into 'blog-example-fork'...

remote: Counting objects: 6, done.

remote: Compressing objects: 100% (4/4), done.

remote: Total 6 (delta 0), reused 6 (delta 0)

Unpacking objects: 100% (6/6), done.

Checking connectivity... done

nitin@jane-saucy:~$ cd blog-example-fork/

nitin@jane-saucy:~/blog-example-fork$ ls

README.md words.txt

The origin remote is pre-set for you by GitHub to point to your forked repo. Let's add another remote called upstream - this will be your main repo (the repo that you forked from). The command to do that is - git remote add upstream <repo-location>

nitin@jane-saucy:~/blog-example-fork$ git remote add upstream https://github.com/nitstorm/blog-example.git

nitin@jane-saucy:~/blog-example-fork$ git remote

origin

upstream

nitin@jane-saucy:~/blog-example-fork$ git remote show origin

\* remote origin

Fetch URL: https://github.com/nitin-test/blog-example-fork.git

Push URL: https://github.com/nitin-test/blog-example-fork.git

HEAD branch: master

Remote branch:

master tracked

Local branch configured for 'git pull':

master merges with remote master

Local ref configured for 'git push':

master pushes to master (up to date)

nitin@jane-saucy:~/blog-example-fork$ git remote show upstream

\* remote upstream

Fetch URL: https://github.com/nitstorm/blog-example.git

Push URL: https://github.com/nitstorm/blog-example.git

HEAD branch: master

Remote branch:

master new (next fetch will store in remotes/upstream)

Local ref configured for 'git push':

master pushes to master (up to date)

Step 2: Making changes and submitting Pull Requests

Our plan is to have our master branch of the forked repo **mirror** the master branch of the main repo while each feature we work on gets it's own branch.

Let's say we wanted to add words to the words.txt file. In order to do that, let's create a branch called word-addition. This is done with the command git checkout -b <branch-name>. This creates a new branch and automatically switches you into it.

nitin@jane-saucy:~/blog-example-fork$ git checkout -b word-addition

Switched to a new branch 'word-addition'

Now we add some words to it, commit the changes and push the updates to our forked repo. Remember to push the commits to the newly-created word-addition branch.

#### Adding new word

nitin@jane-saucy:~/blog-example-fork$ cat words.txt

Random

Access

nitin@jane-saucy:~/blog-example-fork$ echo "Memory" >> words.txt

nitin@jane-saucy:~/blog-example-fork$ cat words.txt

Random

Access

Memory

#### Committing the change

nitin@jane-saucy:~/blog-example-fork$ git commit -am "Adds the word memory"

[word-addition 45ef9f3] Adds the word memory

1 file changed, 1 insertion(+)

#### Pushing to our forked repo

nitin@jane-saucy:~/blog-example-fork$ git push origin word-addition

Username for 'https://github.com': nitin-test

Password for 'https://nitin-test@github.com':

Counting objects: 5, done.

Delta compression using up to 6 threads.

Compressing objects: 100% (2/2), done.

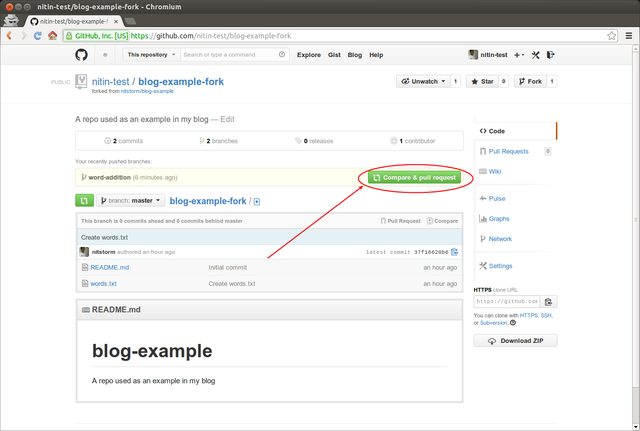
Writing objects: 100% (3/3), 310 bytes | 0 bytes/s, done.

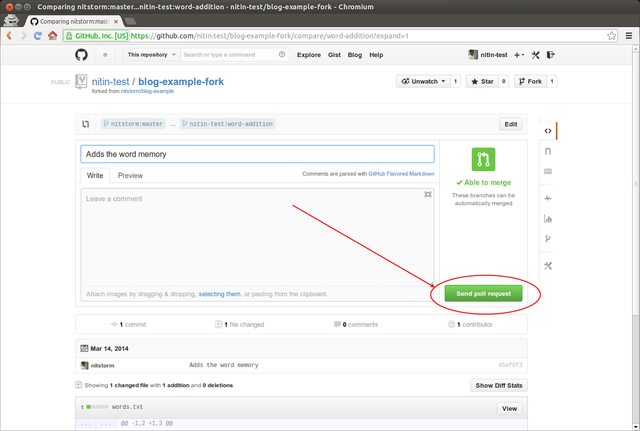
Total 3 (delta 0), reused 0 (delta 0)

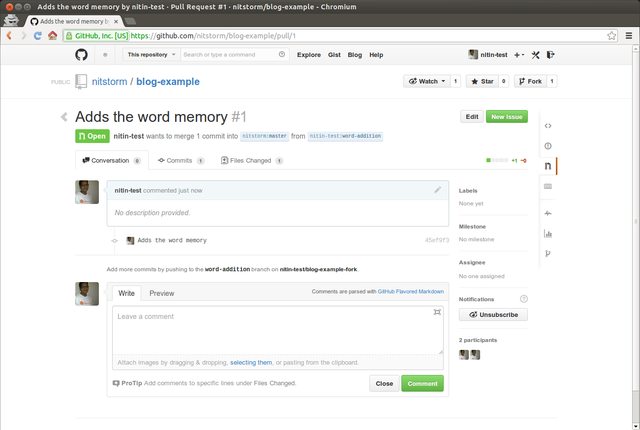
To https://github.com/nitin-test/blog-example-fork.git

\* [new branch] word-addition -> word-addition

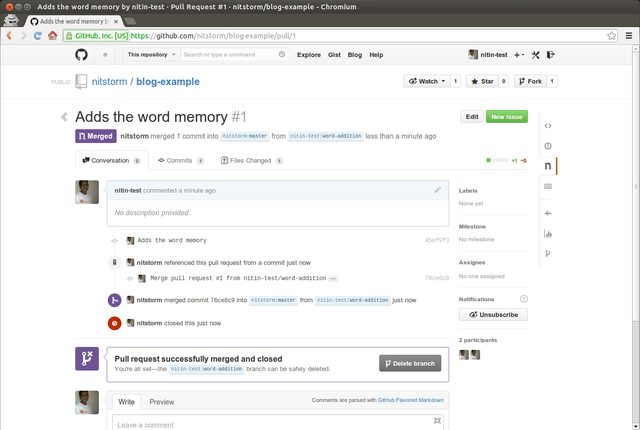
We can see the newly added branch on our forked repo's page on GitHub. Click on the **Compare & Pull request** button and submit a pull request.

[](http://i.imgur.com/CXlT0f4.png)

[](http://i.imgur.com/8l2heSy.png)

[](http://i.imgur.com/yCNd9bo.png)

Once our pull request has been accepted and merged with the main repo, you should see your pull request accepted and closed.

[](http://i.imgur.com/4JiBYkj.png)

Step 3: Keeping the forked repo synced with the main repo

And here comes the juiciest part, the one that this post was written for in the first place - making your repo include all changes made in the main repo.

Let's say a new file called sentences.txt has now been added to the main repo. This needs to show up even in your own forked repo so that you can create a new branch to provide upgrades to this new file. What do we do?

Remember that upstream remote that we created way back in Step 1? Well, it's time to bring it out to play. We first fetch the changes from upstream and then rebase our repo with upstream's master.

git fetch upstream

git rebase upstream/master

Please make sure you are in the master branch of your repo before you rebase. You can switch between branches using git checkout <branch-name>

nitin@jane-saucy:~/blog-example-fork$ git checkout master

Switched to branch 'master'

nitin@jane-saucy:~/blog-example-fork$ ls

README.md words.txt

nitin@jane-saucy:~/blog-example-fork$ git fetch upstream

remote: Counting objects: 6, done.

remote: Compressing objects: 100% (3/3), done.

remote: Total 4 (delta 0), reused 4 (delta 0)

Unpacking objects: 100% (4/4), done.

From https://github.com/nitstorm/blog-example

\* [new branch] master -> upstream/master

nitin@jane-saucy:~/blog-example-fork$ git rebase upstream/master

First, rewinding head to replay your work on top of it...

Fast-forwarded master to upstream/master.

Ofcourse, you'll also need to push the commits to your forked repo. So,

nitin@jane-saucy:~/blog-example-fork$ git status

# On branch master

# Your branch is ahead of 'origin/master' by 3 commits.

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

#

nothing to commit, working directory clean

nitin@jane-saucy:~/blog-example-fork$ git push origin master

Username for 'https://github.com': nitin-test

Password for 'https://nitin-test@github.com':

Counting objects: 6, done.

Delta compression using up to 6 threads.

Compressing objects: 100% (3/3), done.

Writing objects: 100% (4/4), 598 bytes | 0 bytes/s, done.

Total 4 (delta 0), reused 0 (delta 0)

To https://github.com/nitin-test/blog-example-fork.git

37f1662..e9f63f5 master -> master

And that's it!