https://gist.github.com/Chaser324/ce0505fbed06b947d962

GitHub Standard Fork & Pull Request Workflow

[**Raw**](https://gist.github.com/Chaser324/ce0505fbed06b947d962/raw/8dec7cbd3e1b984a98f71b3b8ca158e627201ebb/GitHub-Forking.md)

[**GitHub-Forking.md**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#file-github-forking-md)

Whether you're trying to give back to the open source community or collaborating on your own projects, knowing how to properly fork and generate pull requests is essential. Unfortunately, it's quite easy to make mistakes or not know what you should do when you're initially learning the process. I know that I certainly had considerable initial trouble with it, and I found a lot of the information on GitHub and around the internet to be rather piecemeal and incomplete - part of the process described here, another there, common hangups in a different place, and so on.

In an attempt to coallate this information for myself and others, this short tutorial is what I've found to be fairly standard procedure for creating a fork, doing your work, issuing a pull request, and merging that pull request back into the original project.

**Creating a Fork**

Just head over to the GitHub page and click the "Fork" button. It's just that simple. Once you've done that, you can use your favorite git client to clone your repo or just head straight to the command line:

# Clone your fork to your local machine

git clone git@github.com:USERNAME/FORKED-PROJECT.git

**Keeping Your Fork Up to Date**

While this isn't an absolutely necessary step, if you plan on doing anything more than just a tiny quick fix, you'll want to make sure you keep your fork up to date by tracking the original "upstream" repo that you forked. To do this, you'll need to add a remote:

# Add 'upstream' repo to list of remotes

git remote add upstream https://github.com/UPSTREAM-USER/ORIGINAL-PROJECT.git

# Verify the new remote named 'upstream'

git remote -v

Whenever you want to update your fork with the latest upstream changes, you'll need to first fetch the upstream repo's branches and latest commits to bring them into your repository:

# Fetch from upstream remote

git fetch upstream

# View all branches, including those from upstream

git branch -va

Now, checkout your own master branch and merge the upstream repo's master branch:

# Checkout your master branch and merge upstream

git checkout master

git merge upstream/master

If there are no unique commits on the local master branch, git will simply perform a fast-forward. However, if you have been making changes on master (in the vast majority of cases you probably shouldn't be - [see the next section](https://gist.github.com/Chaser324/ce0505fbed06b947d962#doing-your-work), you may have to deal with conflicts. When doing so, be careful to respect the changes made upstream.

Now, your local master branch is up-to-date with everything modified upstream.

**Doing Your Work**

**Create a Branch**

Whenever you begin work on a new feature or bugfix, it's important that you create a new branch. Not only is it proper git workflow, but it also keeps your changes organized and separated from the master branch so that you can easily submit and manage multiple pull requests for every task you complete.

To create a new branch and start working on it:

# Checkout the master branch - you want your new branch to come from master

git checkout master

# Create a new branch named newfeature (give your branch its own simple informative name)

git branch newfeature

# Switch to your new branch

git checkout newfeature

Now, go to town hacking away and making whatever changes you want to.

**Submitting a Pull Request**

**Cleaning Up Your Work**

Prior to submitting your pull request, you might want to do a few things to clean up your branch and make it as simple as possible for the original repo's maintainer to test, accept, and merge your work.

If any commits have been made to the upstream master branch, you should rebase your development branch so that merging it will be a simple fast-forward that won't require any conflict resolution work.

# Fetch upstream master and merge with your repo's master branch

git fetch upstream

git checkout master

git merge upstream/master

# If there were any new commits, rebase your development branch

git checkout newfeature

git rebase master

Now, it may be desirable to squash some of your smaller commits down into a small number of larger more cohesive commits. You can do this with an interactive rebase:

# Rebase all commits on your development branch

git checkout

git rebase -i master

This will open up a text editor where you can specify which commits to squash.

**Submitting**

Once you've committed and pushed all of your changes to GitHub, go to the page for your fork on GitHub, select your development branch, and click the pull request button. If you need to make any adjustments to your pull request, just push the updates to GitHub. Your pull request will automatically track the changes on your development branch and update.

**Accepting and Merging a Pull Request**

Take note that unlike the previous sections which were written from the perspective of someone that created a fork and generated a pull request, this section is written from the perspective of the original repository owner who is handling an incoming pull request. Thus, where the "forker" was referring to the original repository as upstream, we're now looking at it as the owner of that original repository and the standard origin remote.

**Checking Out and Testing Pull Requests**

Open up the .git/config file and add a new line under [remote "orgin"]:

fetch = +refs/pull/\*/head:refs/pull/origin/\*

Now you can fetch and checkout any pull request so that you can test them:

# Fetch all pull request branches

git fetch origin

# Checkout out a given pull request branch based on its number

git checkout -b 999 pull/origin/999

Keep in mind that these branches will be read only and you won't be able to push any changes.

**Automatically Merging a Pull Request**

In cases where the merge would be a simple fast-forward, you can automatically do the merge by just clicking the button on the pull request page on GitHub.

**Manually Merging a Pull Request**

To do the merge manually, you'll need to checkout the target branch in the source repo, pull directly from the fork, and then merge and push.

# Checkout the branch you're merging to in the target repo

git checkout master

# Pull the development branch from the fork repo where the pull request development was done.

git pull https://github.com/forkuser/forkedrepo.git newfeature

# Merge the development branch

git merge newfeature

# Push master with the new feature merged into it

git push origin master

Now that you're done with the development branch, you're free to delete it.

git branch -d newfeature

**Sources**

* [GitHub - Fork a Repo](https://help.github.com/articles/fork-a-repo)
* [GitHub - Syncing a Fork](https://help.github.com/articles/syncing-a-fork)
* [GitHub - Checking Out a Pull Request](https://help.github.com/articles/checking-out-pull-requests-locally)

[](https://gist.github.com/dhimmel)

[**dhimmel**](https://gist.github.com/dhimmel)**commented**[**on Nov 4, 2016**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-1914159)**• edited**

|  |
| --- |
| The first part of this guide assumes you're forking a GitHub repository which is referred to as the upstream remote. Therefore, the [Accepting and Merging a Pull Request](https://gist.github.com/Chaser324/ce0505fbed06b947d962#accepting-and-merging-a-pull-request) may be more useful from the perspective of checking out pull requests from upstream rather than origin? |

[](https://gist.github.com/Chaser324)

Owner

[**Chaser324**](https://gist.github.com/Chaser324)**commented**[**on Dec 15, 2016**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-1948207)**• edited**

|  |
| --- |
| [**@dhimmel**](https://github.com/dhimmel) You may be right, but I guess my thought process when I was writing this was that I wanted to address the entire life cycle and workflow of a pull request. I added a little bit to the gist to clarify that the later section is written from the perspective of the original repo owner.  Apologies for the late reply...never noticed any sort of notification that someone had made a comment.  Also, wow, how does this have 100+ stars?! I mostly wrote this for myself as a reference when I forget something. |

[](https://gist.github.com/danhett)

[**danhett**](https://gist.github.com/danhett)**commented**[**on Dec 15, 2016**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-1948212)

|  |
| --- |
| This is a really handy thing! I get asked about this stuff a lot by people in my workshops who are learning, and concise references like this are gold. Gold star get! ^\_\_^ |

[](https://gist.github.com/Chaser324)

Owner

[**Chaser324**](https://gist.github.com/Chaser324)**commented**[**on Dec 15, 2016**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-1948217)

|  |
| --- |
| [**@danhett**](https://github.com/danhett). Nice! Glad you've found it helpful and that it's maybe helped out some budding new programmers. |

[](https://gist.github.com/strobelight)

[**strobelight**](https://gist.github.com/strobelight)**commented**[**on Feb 15**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2001686)

|  |
| --- |
| So, does *Now, go to town hacking away and making whatever changes you want to.* mean that commits and pushes to the cloned fork are OK to do? |

[](https://gist.github.com/arunksaha)

[**arunksaha**](https://gist.github.com/arunksaha)**commented**[**on Feb 18**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2004700)

|  |
| --- |
| For submitting or pushing, the following command worked for me:  git push --set-upstream origin newfeature |

[](https://gist.github.com/Chaser324)

Owner

[**Chaser324**](https://gist.github.com/Chaser324)**commented**[**on Feb 20**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2006218)

|  |
| --- |
| [**@strobelight**](https://github.com/strobelight): Yeah. That's what I meant by that. After you've created a branch for your work, you can start committing to it and pushing to your fork. If you want to make a pull request, you would then later generate a pull request for the commits you've made on that branch. |

[](https://gist.github.com/fireballwayne)

[**fireballwayne**](https://gist.github.com/fireballwayne)**commented**[**on Mar 2**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2015921)

|  |
| --- |
| This is a good article. I use this to rebase my newfeature on the latest changes in the upstream/master and upload to github: git fetch upstream git rebase upstream/master git push -f origin newfeature  After that I recompile to make sure my new feature still works, then create a pull request. It works great, however it doesn't update your copy of the master branch. |

[](https://gist.github.com/bendichter)

[**bendichter**](https://gist.github.com/bendichter)**commented**[**on Mar 10**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2024048)

|  |
| --- |
| Thanks for the article. Do you generally use a virtual environment for development? Could be really useful if you are using this library for other projects as well. |

[](https://gist.github.com/Chaser324)

Owner

[**Chaser324**](https://gist.github.com/Chaser324)**commented**[**on Apr 10**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2055141)

|  |
| --- |
| [**@fireballwayne**](https://github.com/fireballwayne) - See the section above in my post titled "Cleaning Up Your Work" for a better approach to making sure you're up to date with the upstream master branch.  The set of commands you have listed there would in fact have the problem of not updating your copy of the master branch. Those commands are essentially pulling/merging directly from the upstream master to your feature branch skipping over your copy of the master branch. Beware of any time you're throwing that "-f" force argument into a push, it's a sign that what you're doing is probably not good.  The better workflow, as described above is to merge upstream/master into your copy of the master branch (this makes it now up to date) and then rebasing your feature branch against your master branch. The end result for your feature branch is the same, but with this process, your master branch is also consistent with the upstream repo.  [**@bendichter**](https://github.com/bendichter) - I don't personally use a virtual environment all that often. |

[](https://gist.github.com/NaridaL)

[**NaridaL**](https://gist.github.com/NaridaL)**commented**[**on Apr 13**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2058579)**• edited**

|  |
| --- |
| [**@Chaser324**](https://github.com/chaser324) If I have a PR which is rejected/pending, but I still want to have a complete version locally with all my changes, should I merge the branch into my local master? If I then create additional PRs, should I base them on upstream/master or my local master? Or would you recommend an alternative workflow? |

[](https://gist.github.com/Chaser324)

Owner

[**Chaser324**](https://gist.github.com/Chaser324)**commented**[**on Apr 13**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2059020)**• edited**

|  |
| --- |
| [**@NaridaL**](https://github.com/naridal) I would say that at least partly depends on whether or not you think there will be any significant overlap between your PRs. Always basing your PRs on the current upstream master is probably the safest bet, but if it's unlikely they'll touch the same files or code, you could possibly merge to your local master and base your PRs on that.  Another solid option would be to keep your master branch consistent with the upstream, but then create a branch off of that, let's call it pending-master, and do your PR work branched off of that. You can then merge to this pending-master all you want and still have the convenience of a clean master branch that's consistent with the upstream one. When you issue your PRs, you could then just rebase your PR branches onto the master. For example, you could have a setup like this:  ---o---o---o---o master (consistent w/ upstream)  \  o---o---o---o---o pending-master (merge/branch from this as much as you want)  \  o---o---o newfeature  When you're ready to issue the pull request, you could then rebase your newfeature branch onto master.  # Move newfeature branch onto master.  git rebase --onto master pending-master newfeature  This would give you something like this:  ---o---o---o---o master  | \  | o´--o´--o´ newfeature  \  o---o---o---o---o pending-master |

[](https://gist.github.com/NaridaL)

[**NaridaL**](https://gist.github.com/NaridaL)**commented**[**on Apr 15**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2061237)**• edited**

|  |
| --- |
| [**@Chaser324**](https://github.com/chaser324) Thanks for the writeup! |

[](https://gist.github.com/marnen)

[**marnen**](https://gist.github.com/marnen)**commented**[**on May 16**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2097066)

|  |
| --- |
| Thanks for the writeup, although I hate to see yet another guide recommending rebasing your feature branch (in general, merging works better). |

[](https://gist.github.com/Chaser324)

Owner

[**Chaser324**](https://gist.github.com/Chaser324)**commented**[**on May 25**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2105830)

|  |
| --- |
| Merging does have some benefits over rebasing in terms of safety and some other factors, but I generally think rebasing makes for a smoother and cleaner pull request workflow.  Here's a pretty good writeup from Atlassian on the subject: <https://www.atlassian.com/git/tutorials/merging-vs-rebasing> |

[](https://gist.github.com/wangf1)

[**wangf1**](https://gist.github.com/wangf1)**commented**[**on May 31**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2111442)**• edited**

|  |
| --- |
| Thanks! Very good article. I know how to create pull request by reading this article. I have one finding: In section "Checking Out and Testing Pull Requests", if I change the "origin" to "upstream", I can use same local repository checkout the pull request instead of clone the original github repository.  //In .git/config file  [remote "upstream"]  url = https://\*\*\*.git  fetch = +refs/heads/\*:refs/remotes/upstream/\*  fetch = +refs/pull/\*/head:refs/pull/upstream/\*  # Fetch all pull request branches  git fetch upstream  # Checkout out a given pull request branch based on its number  git checkout -b 999 pull/upstream/999 |

[](https://gist.github.com/marnen)

[**marnen**](https://gist.github.com/marnen)**commented**[**on Jun 1**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2112305)

|  |
| --- |
| [**@Chaser324**](https://github.com/chaser324):  I generally think rebasing makes for a smoother and cleaner pull request workflow.  I can't think of any respect—not one—in which this statement is accurate. It is just as easy to accept a pull request that contains merge commits, and as you acknowledge, rebasing is less safe than merging. In fact, I'd go further, and say that many uses of rebase make Git into something that is no longer a VCS (because it is not maintaining a persistent historical record).  So...I'm curious how you think rebasing makes pull requests "smoother and cleaner", because I don't even know where to begin with that claim. |

[](https://gist.github.com/sledge909)

[**sledge909**](https://gist.github.com/sledge909)**commented**[**on Jun 3**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2113635)

|  |
| --- |
| Really useful thanks! Bookmarked. |

[](https://gist.github.com/Chaser324)

Owner

[**Chaser324**](https://gist.github.com/Chaser324)**commented**[**on Jun 5**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2115051)

|  |
| --- |
| [**@marnen**](https://github.com/marnen) Let's just chill out and keep things civil. I get that this is something that's debated, and I understand your point of view. This just maybe isn't the best place for having a back and forth on this.  This post is just a suggestion based on how I do things and how teams I've worked with have done things. I'd never claim that this is some sort of definitive post about the one and only way to do things. There are a lot of factors to consider, and it's up to individual teams to decide what works best for them.  Feel free to fork and create a different version of this post. |

[](https://gist.github.com/thedude42)

[**thedude42**](https://gist.github.com/thedude42)**commented**[**17 days ago**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2131887)

|  |
| --- |
| The Atlassian article you mention talks about some pitfalls and good practices around the use of rebasing. Seems like it may be a good idea to reference those in this piece for folks casually finding your work, and who may not be inclined to read through all the comments. I found the Atlassian article a great folow-up after reading yours. |

[](https://gist.github.com/manpen)

[**manpen**](https://gist.github.com/manpen)**commented**[**16 days ago**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2132106)

|  |
| --- |
| Thank you for the great summary! There is a small typo in "Checking Out and Testing Pull Requests" which is hard to spot for copy/paste users as me:  Open up the .git/config file and add a new line under [remote "or**i**gin"]: |

[](https://gist.github.com/StoneyJackson)

[**StoneyJackson**](https://gist.github.com/StoneyJackson)**commented**[**9 days ago**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2137904)

|  |
| --- |
| [**@Chaser324**](https://github.com/chaser324) Very nice! I've written something similar, but I prefer yours. Plus, with so many stars, yours has clear validation. Would you be willing to add a license, preferably an open license (e.g., creative commons, etc.)? |

[](https://gist.github.com/BobCochran)

[**BobCochran**](https://gist.github.com/BobCochran)**commented**[**8 days ago**](https://gist.github.com/Chaser324/ce0505fbed06b947d962#gistcomment-2138965)

|  |
| --- |
| Hi! I am really rusty with my git knowledge because I don't use it very often. Today, I forked another repository, set up my local repository of the fork, created a branch called page44, and went ahead to make some changes on that branch. Then I did a pull request for those changes. The PR is now pending. I kept thinking my PR must have a lot of clutter inside it because I renamed a source code directory and that was a bigger change than it seemed. I think I could have done a much cleaner pull request if I had followed your suggestions to [**@NaridaL**](https://github.com/naridal) and also tried rebasing like the Atlassian article suggests. |