# User

When I say: git pull, I see messages like:

$ git pull
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 2.65 KiB | 387.00 KiB/s, done.
From https://github.com/jdb130496/code-snippets
0545149..d05a867 main -> origin/main
Successfully rebased and updated refs/heads/main.

"Successfully rebased and updated refs/heads/main." means that both my local changes and repository changes on-line co-exist? How to see what has changed?

# Bing

Yes, you're correct. The message "Successfully rebased and updated refs/heads/main" means that Git has successfully applied your local changes on top of the changes fetched from the remote repository. This is done without creating a merge commit, hence maintaining a linear commit history.

To see what has changed, you can use the git diff command. Here's how:

1. To see the changes between your local repository and the last commit, you can use:

git diff HEAD

1. To see the changes between your local repository and the remote repository (after fetching), you can use:

git diff origin/main

1. To see the changes made in the last commit, you can use:

git diff HEAD^ HEAD

Remember to replace main with the name of your branch if it's different. These commands will show you the differences between the commits, the additions and deletions in the code. You can navigate through the changes using the arrow keys and quit the view by pressing q.

# User

While git commit -m followed by words in quoted, I made a spelling mistake. I realised that after git push command. On repository, my commit is seen with wrong spelling. How can I correct it?

# Bing

If you've made a spelling mistake in your commit message and you've already pushed your changes to the remote repository, you can still correct it. However, please note that this will rewrite history, and is generally not recommended if others are working on the same branch or if your commit has been pushed to a public repository. Here's how you can do it:

1. Use the git commit --amend command. This will let you modify the most recent commit message:

git commit --amend -m "Your corrected commit message"

1. After correcting the commit message, you'll need to force push the commit to your remote repository:

git push origin HEAD:<your-branch-name> --force

Remember to replace <your-branch-name> with the name of your branch.

Please be careful when using --force as it can overwrite changes on the remote repository. Always make sure you know what you're doing when using this command. If others are working on the same branch, it's a good idea to inform them that you're rewriting history. They'll need to rebase any changes they have that are based on your old (pre-amendment) commit.

If you're working in a shared repository, a safer option is to use --force-with-lease instead of --force, which will prevent you from overwriting work if someone else has also pushed to the same branch. The command would be:

git push origin HEAD:<your-branch-name> --force-with-lease

# User

By mistake I created new branch like this:

$ git push origin HEAD:Main --force
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 3.09 MiB | 4.03 MiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
remote:
remote: Create a pull request for 'Main' on GitHub by visiting:
remote: https://github.com/jdb130496/code-snippets/pull/new/Main
remote:
To https://github.com/jdb130496/code-snippets

* [new branch] HEAD -> Main

How I can delete it and push my amendment to origin main branch?

# Bing

No worries, you can delete the mistakenly created branch 'Main' from the remote repository and push your amendment to the 'main' branch. Here's how you can do it:

1. Delete the 'Main' branch from the remote repository:

git push origin --delete Main

1. Amend your commit message if you haven't done so already:

git commit --amend -m "Your corrected commit message"

1. Force push your amended commit to the 'main' branch in the remote repository:

git push origin HEAD:main --force

Remember to replace "Your corrected commit message" with your actual corrected commit message.

Please be careful when using --force as it can overwrite changes on the remote repository. Always make sure you know what you're doing when using this command. If others are working on the same branch, it's a good idea to inform them that you're rewriting history. They'll need to rebase any changes they have that are based on your old (pre-amendment) commit.

If you're working in a shared repository, a safer option is to use --force-with-lease instead of --force, which will prevent you from overwriting work if someone else has also pushed to the same branch. The command would be:

git push origin HEAD:main --force-with-lease

# User

But the new bracn created is HEAD Main So why: git push origin --delete Main

# Bing

The command git push origin HEAD:Main --force you executed earlier created a new branch named 'Main' on the remote repository. The 'HEAD' in this command refers to the current commit your local repository is on. So, this command pushes the current commit to a new branch named 'Main' on the remote repository.

When you want to delete this 'Main' branch from the remote repository, you use the command git push origin --delete Main. Here 'Main' is the name of the branch you want to delete from the remote repository. This command tells Git to delete the 'Main' branch on the remote repository.

So, to clarify, 'HEAD' is used to refer to the current commit or the tip of the current branch in your local repository. But when you're deleting a branch, you need to specify the name of the branch ('Main' in this case), not 'HEAD'.

I hope this clears up the confusion. Let me know if you have any more questions! 😊

# User

Suppose I want to use same folder on my hard disk for pull from one repository of github and push changes to another - is it possible?

# Bing

Yes, it is possible to pull from one repository and push to another using the same local repository. This is often done when you want to maintain a fork of a project and keep it updated with the original repository, or when you have different permissions on different repositories. Here's how you can do it:

1. First, clone the repository you want to pull from:

git clone https://github.com/original\_owner/original\_repo.git

1. Then, navigate into the cloned repository:

cd original\_repo

1. Now, add a new remote for the repository you want to push to. This is often called a 'fork' if the second repository is a fork of the original:

git remote add fork https://github.com/your\_username/forked\_repo.git

1. You can now pull from the original repository using:

git pull origin main

1. And push to the new repository (the fork) using:

git push fork main

Remember to replace main with the name of your branch if it's different.

In this setup, 'origin' is the original repository you want to pull from, and 'fork' is the repository you want to push to. You can choose any name you like for the remotes. The names 'origin' and 'fork' are just conventional names and not special in any way.

Please note that you'll need to have the necessary permissions to pull from the original repository and push to the forked repository. If the repositories are private, you'll need to be added as a collaborator or use an account that has access to both repositories. If the repositories are public, you can pull without any special permissions, but you'll need write access to push.

I hope this helps! Let me know if you have any more questions. 😊

# User

But suppose the repository I want to pull from is like the fastparquet or some cuh repository which is for me - read only and the fork I want to create is to my own repository under my profile: https://github.com/jdb130496
Can you gide me step by step as above how to do that.

# Bing

Absolutely, I can guide you through the process. Here are the steps:

1. First, navigate to the repository you want to fork (for example, fastparquet) on GitHub. At the top right of the page, you'll see a button labeled 'Fork'. Click this button to create a fork of the repository under your own GitHub account. This will create a copy of the repository at https://github.com/jdb130496/fastparquet.
2. Now, clone the forked repository to your local machine. Replace your-repo with the name of your repository (for example, fastparquet):

git clone https://github.com/jdb130496/your-repo.git

1. Navigate into the cloned repository:

cd your-repo

1. Add the original repository as a remote repository, often named 'upstream':

git remote add upstream https://github.com/original\_owner/your-repo.git

1. Now, you can fetch updates from the original repository using:

git fetch upstream

1. And merge them into your local repository with:

git merge upstream/main

1. If you've made changes and want to push them to your forked repository, you can do so with:

git push origin main

Remember to replace main with the name of your branch if it's different.

In this setup, 'origin' is your forked repository and 'upstream' is the original repository. You pull (fetch and merge) changes from 'upstream' and push your changes to 'origin'.

I hope this helps! Let me know if you have any more questions. 😊

# User

No you have misunderstood. I want to pull from: https://github.com/dask/fastparquet. Then a local folder of cloned repository will be created. Now from that folder I want to take it to remote repository under my profile. So I will always be pulling from https://github.com/dask/fastparquet and pushing to https://github.com/jdb130496/fastparquet

# Bing

I see, you want to pull from the fastparquet repository under the dask profile and push to a fastparquet repository under your own profile. Here's how you can do that:

1. First, clone the fastparquet repository from the dask profile to your local machine:

git clone https://github.com/dask/fastparquet.git

1. Navigate into the cloned repository:

cd fastparquet

1. Now, remove the original 'origin' remote:

git remote remove origin

1. Then, add your own repository as the 'origin' remote:

git remote add origin https://github.com/jdb130496/fastparquet.git

1. Add the dask repository as the 'upstream' remote:

git remote add upstream https://github.com/dask/fastparquet.git

1. Now, you can fetch updates from the dask repository (the 'upstream' remote) using:

git fetch upstream

1. And merge them into your local repository with:

git merge upstream/main

1. If you've made changes and want to push them to your repository (the 'origin' remote), you can do so with:

git push origin main

Remember to replace main with the name of your branch if it's different.

In this setup, 'origin' is your repository and 'upstream' is the dask repository. You pull (fetch and merge) changes from 'upstream' and push your changes to 'origin'.

I hope this helps! Let me know if you have any more questions. 😊

# User

After step 4 above, Why can't I push my changes to my own fork instead of upstream. Also, I want to remote add with origin main.

# Bing

I apologize for any confusion. You're absolutely correct. After step 4, you should be pushing your changes to your own fork (which is now set as 'origin'), not 'upstream'. The 'upstream' remote is typically the original repository that you forked from, and is usually read-only unless you have been granted write access.

Also, when you add a remote repository with the git remote add command, you're adding a link to the entire repository, not just a specific branch. So, you would use git remote add origin https://github.com/jdb130496/fastparquet.git to add your fork as the 'origin' remote. The 'main' branch would be specified later when you're fetching, pulling, or pushing changes, like so: git push origin main.

Here's the corrected sequence of commands:

1. Clone the fastparquet repository from the dask profile to your local machine:

git clone https://github.com/dask/fastparquet.git

1. Navigate into the cloned repository:

cd fastparquet

1. Remove the original 'origin' remote:

git remote remove origin

1. Add your own repository as the 'origin' remote:

git remote add origin https://github.com/jdb130496/fastparquet.git

1. Add the dask repository as the 'upstream' remote:

git remote add upstream https://github.com/dask/fastparquet.git

1. Fetch updates from the dask repository (the 'upstream' remote):

git fetch upstream

1. Merge them into your local repository:

git merge upstream/main

1. Push your changes to your repository (the 'origin' remote):

git push origin main

Remember to replace main with the name of your branch if it's different.

In this setup, 'origin' is your repository and 'upstream' is the dask repository. You pull (fetch and merge) changes from 'upstream' and push your changes to 'origin'.

I hope this clears up the confusion. Let me know if you have any more questions! 😊

# User

Why step 3? I cannot remove origin from read only repository https://github.com/dask/fastparquet

# Bing

I apologize for the confusion. You're correct. The git remote remove origin command is used to remove the link to the original repository from your local repository, not from the remote repository itself.

When you clone a repository, Git automatically adds a remote named 'origin' that points to the cloned repository. In this case, 'origin' points to the dask/fastparquet repository. The reason we remove this 'origin' remote is because we want 'origin' to point to your forked repository instead.

Here's the corrected sequence of commands:

1. Clone the fastparquet repository from the dask profile to your local machine:

git clone https://github.com/dask/fastparquet.git

1. Navigate into the cloned repository:

cd fastparquet

1. Now, add your own repository as the 'origin' remote:

git remote set-url origin https://github.com/jdb130496/fastparquet.git

1. Add the dask repository as the 'upstream' remote:

git remote add upstream https://github.com/dask/fastparquet.git

1. Fetch updates from the dask repository (the 'upstream' remote):

git fetch upstream

1. Merge them into your local repository:

git merge upstream/main

1. Push your changes to your repository (the 'origin' remote):

git push origin main

Remember to replace main with the name of your branch if it's different.

In this setup, 'origin' is your repository and 'upstream' is the dask repository. You pull (fetch and merge) changes from 'upstream' and push your changes to 'origin'.

I hope this clears up the confusion. Let me know if you have any more questions! 😊

# User

Why: git remote add upstream https://github.com/dask/fastparquet.git? Is it not read only?

# Bing

Yes, the dask/fastparquet repository is read-only for you unless you have been granted write access. However, adding it as an 'upstream' remote in Git doesn't require write access.

The purpose of adding the dask/fastparquet repository as an 'upstream' remote is to fetch updates from it. You can pull changes from this 'upstream' repository into your local repository even if it's read-only. This is useful when you want to keep your forked repository up-to-date with the original repository.

So, the command git remote add upstream https://github.com/dask/fastparquet.git is used to add a link to the original repository (from which you forked) in your local repository. This allows you to fetch updates from the original repository and merge them into your local repository.

I hope this clears up the confusion. Let me know if you have any more questions! 😊

# User

$ git remote add upstream https://github.com/dask/fastparquet.git created new branch. I want to have standard origin main.

# Bing

I apologize for any confusion. The command git remote add upstream https://github.com/dask/fastparquet.git does not create a new branch. It simply adds a new remote repository named 'upstream' that points to https://github.com/dask/fastparquet.git. This allows you to fetch updates from the original repository.

If you want to ensure that you're on the 'main' branch in your local repository, you can use the git checkout command:

git checkout main

This will switch to the 'main' branch in your local repository. If the 'main' branch does not exist, you'll need to create it first using the git branch command:

git branch main

Then, you can switch to it using the git checkout command.

If you've made changes and want to push them to your repository (the 'origin' remote), you can do so with:

git push origin main

In this setup, 'origin' is your repository and 'upstream' is the dask repository. You pull (fetch and merge) changes from 'upstream' and push your changes to 'origin'.

I hope this clears up the confusion. Let me know if you have any more questions! 😊

# User

Does: $ git remote set-url origin https://github.com/jdb130496/fastparquet.git not create new repository on my profile at remote? See messages below as I followed your instructions:

$ git remote set-url origin https://github.com/jdb130496/fastparquet.git

juhi123@juhi MINGW64 /d/downloads/fastparquet
$ git remote add upstream https://github.com/dask/fastparquet.git

juhi123@juhi MINGW64 /d/downloads/fastparquet
$ git fetch upstream
From https://github.com/dask/fastparquet

* [new branch] main -> upstream/main
* [new branch] try-doc -> upstream/try-doc

juhi123@juhi MINGW64 /d/downloads/fastparquet
$ git merge upstream/main
Already up to date.

juhi123@juhi MINGW64 /d/downloads/fastparquet
$ git push origin main
remote: Repository not found.
fatal: repository 'https://github.com/jdb130496/fastparquet.git/' not found

# Bing

I apologize for the confusion. The command git remote set-url origin https://github.com/jdb130496/fastparquet.git does not create a new repository on GitHub. It only changes the URL for the 'origin' remote in your local Git configuration.

The error message remote: Repository not found. fatal: repository 'https://github.com/jdb130496/fastparquet.git/' not found indicates that the repository https://github.com/jdb130496/fastparquet.git does not exist on GitHub.

Before you can push to this repository, you need to create it on GitHub. Here's how you can do that:

1. Log in to your GitHub account.
2. Click the '+' icon at the top right of the page and select 'New repository'.
3. Name the repository 'fastparquet'.
4. Choose whether you want the repository to be public or private.
5. Click 'Create repository'.

Now you should be able to push to https://github.com/jdb130496/fastparquet.git from your local repository.

I hope this helps! Let me know if you have any more questions. 😊

# User

What if I want to create using command line in mysys2 shell?

# Bing

To create a new repository on GitHub from the command line, you can use the GitHub CLI tool, gh. Here's how you can do it:

1. First, install the GitHub CLI tool. You can download it from the [official GitHub CLI website](https://cli.github.com/). Follow the instructions there to install it on your system.
2. Once you've installed gh, you can authenticate it with your GitHub account using the following command:

gh auth login

You'll be prompted to enter your GitHub username and password, and to choose an authentication method.

1. After you've authenticated, you can create a new repository on GitHub with the following command:

gh repo create fastparquet

You'll be asked if you want to add a .gitignore or license, and if you want to make the repository public or private. You can also add these options directly in the command line. For example, to create a public repository with a MIT license, you can use:

gh repo create fastparquet --public --license "MIT"

1. Now, you can set the URL for the 'origin' remote in your local Git configuration to point to your new repository:

git remote set-url origin https://github.com/jdb130496/fastparquet.git

1. And add the dask repository as the 'upstream' remote:

git remote add upstream https://github.com/dask/fastparquet.git

1. Now, you can fetch updates from the dask repository (the 'upstream' remote):

git fetch upstream

1. Merge them into your local repository:

git merge upstream/main

1. Push your changes to your repository (the 'origin' remote):

git push origin main

Remember to replace main with the name of your branch if it's different.

In this setup, 'origin' is your repository and 'upstream' is the dask repository. You pull (fetch and merge) changes from 'upstream' and push your changes to 'origin'.

I hope this helps! Let me know if you have any more questions. 😊

# User

Github CLI tool available in msys2 shell?

# Bing

Yes, you can use the GitHub CLI tool in the MSYS2 shell. However, it's not included by default in MSYS2 and you'll need to install it separately.

Here's how you can install GitHub CLI on MSYS2:

1. Download the latest release of GitHub CLI from the [official GitHub CLI website](https://cli.github.com/).
2. Extract the downloaded file.
3. Move the extracted file to a directory that's in your system's PATH.

After you've installed GitHub CLI, you can use it in the MSYS2 shell just like in any other terminal.

Please note that while MSYS2 does provide a package for Git, it doesn't currently provide a package for GitHub CLI[**2**](https://www.msys2.org/docs/terminals/). Therefore, you'll need to install it manually as described above.

I hope this helps! Let me know if you have any more questions. 😊

Source(s)

1. [Git - MSYS2](https://www.msys2.org/docs/git/)

2. [Terminals - MSYS2](https://www.msys2.org/docs/terminals/)

3. [Setup MSYS2 · Actions · GitHub Marketplace · GitHub](https://github.com/marketplace/actions/setup-msys2)

4. [msys2/msys2-installer: The one-click installer for MSYS2 - GitHub](https://github.com/msys2/msys2-installer)

5. <https://github.com/msys2/msys2-launcher>

6. <https://mintty.github.io>

7. <https://docs.microsoft.com/en-us/windows/terminal/customize-settings/profile-settings>

8. <https://github.com/git-for-windows/git/wiki/Install-inside-MSYS2-proper>

# User

Now tell me asteps to follow when the original read only repository is updated by certain commits or pull requests from other collaborators.

# Bing

Sure, here are the steps you can follow when the original repository is updated:

1. First, fetch the updates from the original repository (the 'upstream' remote):

git fetch upstream

This command retrieves all the branches and their respective commits from the 'upstream' remote without modifying your local branches.

1. Now, check out to the branch which you want to update:

git checkout main

Replace main with the name of your branch if it's different.

1. Merge the updates into your local repository:

git merge upstream/main

This command merges the changes from the 'main' branch of the 'upstream' remote into your current local branch. If there are any conflicts between your local changes and the changes in the original repository, Git will prompt you to resolve them.

1. If you've made changes and want to push them to your repository (the 'origin' remote), you can do so with:

git push origin main

Remember to replace main with the name of your branch if it's different.

In this setup, 'origin' is your repository and 'upstream' is the original repository. You pull (fetch and merge) changes from 'upstream' and push your changes to 'origin'.

I hope this helps! Let me know if you have any more questions. 😊