GIT

Table of Contents

From https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners

New to git? Follow the steps below to get comfortable making changes to the code base, opening up a pull request (PR), and merging code into the primary branch. Any important git and GitHub terms are in bold with links to the official git reference materials.

Step 0: Install git and create a GitHub account

The first two things you'll want to do are install git and create a free GitHub account.

Follow the instructions here to install git (if it's not already installed). Note that for this tutorial we will be using **git on the command line only**. While there are some great git GUIs (graphical user interfaces), I think it's easier to learn git using git-specific commands first and then to try out a git GUI once you're more comfortable with the command. A note: 95% of other online git resources and discussions will also be for the command-line interface.

Once you've done that, create a GitHub account here.

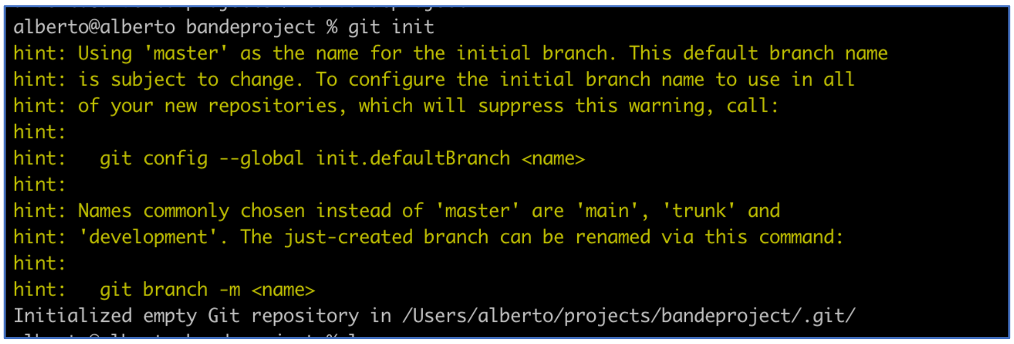
Step 1: Create a local git repository

When creating a new project on your local machine using git, you'll first create a new repository (or often, 'repo', for short).

To use git we'll be using the terminal. If you don't have much experience with the terminal and basic commands, check out this tutorial (If you don't want/ need a short history lesson, skip to step three.)

To begin, **open up a terminal and move to where you want to place the project on your local machine** using the cd (change directory) command.

To **initialize a git repository** in the root of the folder, run the ***git init*** command:



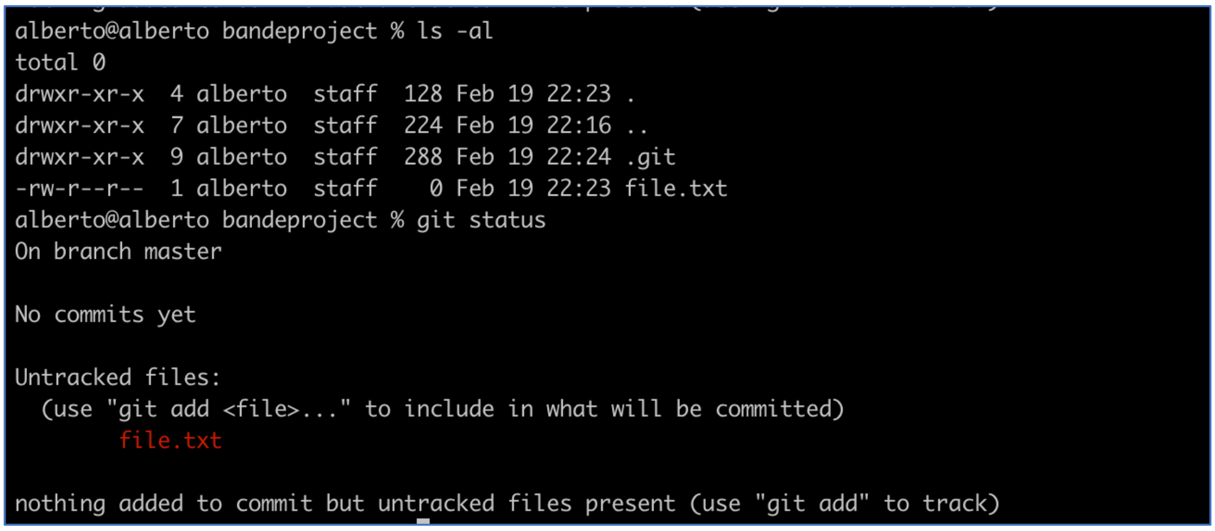
```
alberto@alberto bandeproject % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/alberto/projects/bandeproject/.git/
```

Step 2: Add a new file to the repo

Go ahead and add a new file to the project, using any text editor you like or running a touch command. `touch newfile.txt` just creates and saves a blank file named newfile.txt.

Once you've added or modified files in **a folder containing a git repo**, git will notice that the file exists inside the repo. But, git won't track the file unless you explicitly tell it to. **Git only saves/manages changes to files that it tracks**, so we'll need to send a command to confirm that yes, we want git to track our new file.

After creating the new file, you can use the ***git status*** command to see which files git knows exist.



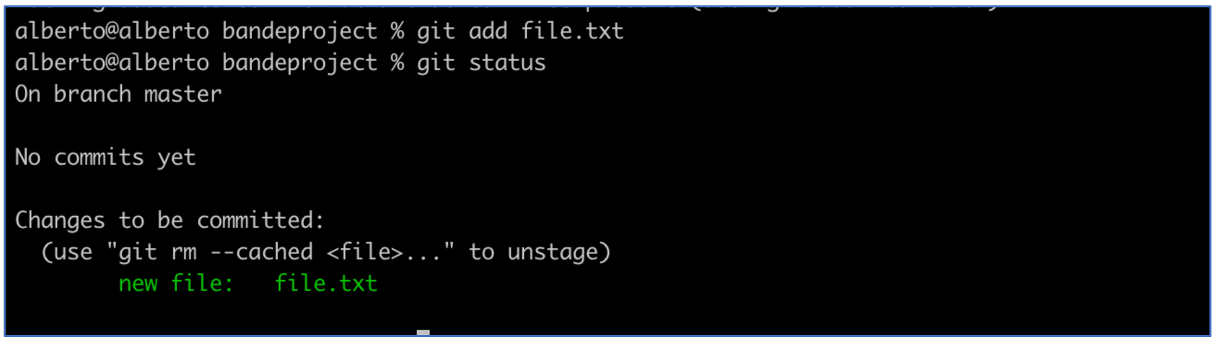
```
alberto@alberto bandeproject % ls -al
total 0
drwxr-xr-x  4 alberto  staff  128 Feb 19 22:23 .
drwxr-xr-x  7 alberto  staff  224 Feb 19 22:16 ..
drwxr-xr-x  9 alberto  staff  288 Feb 19 22:24 .git
-rw-r--r--  1 alberto  staff    0 Feb 19 22:23 file.txt
alberto@alberto bandeproject % git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
	file.txt

nothing added to commit but untracked files present (use "git add" to track)
```

What this basically says is, "Hey, we noticed you created a new file called mnelson.txt, but unless you use the 'git add' command we aren't going to do anything with it."

Step 3: Add a file to the staging environment

Add a file to the staging environment using the ***git add <filename>*** command.

If you rerun the ***git status*** command, you'll see that git has added the file to the staging environment (notice the "Changes to be committed" line).



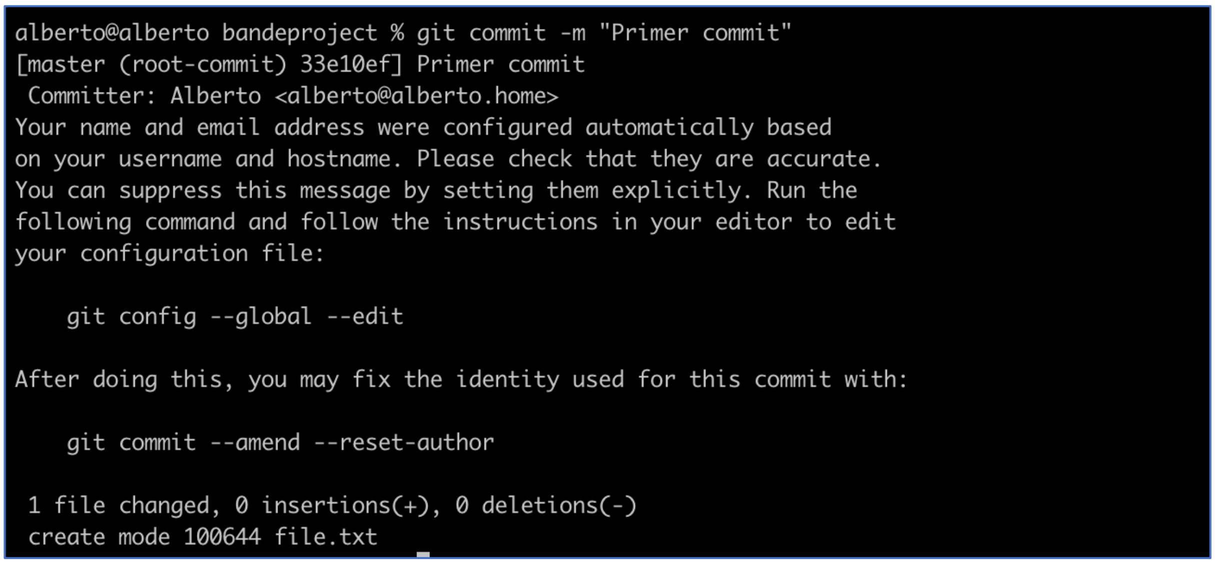
```
alberto@alberto bandeproject % git add file.txt
alberto@alberto bandeproject % git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
	new file:   file.txt
```

To reiterate, the file **has not yet been added** to a commit, but it's about to be.

Step 4: Create a commit

It's time to create your first commit!

Run the command ***git commit -m <"Your message about the commit">***



```
alberto@alberto bandeproject % git commit -m "Primer commit"
[master (root-commit) 33e10ef] Primer commit
 Committer: Alberto <alberto@alberto.home>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file.txt
```

The message at the end of the commit should be **something related to what the commit contains** - maybe it's a new feature, maybe it's a bug fix, maybe it's just fixing a typo. Don't put a message like "asdfadsf" or "foobar". That makes the other people who see your commit sad. Very, very, sad. **Commits live forever in a repository** (technically you can delete them if you really, really need to but it’s messy), so if you leave a clear explanation of your changes it can be extremely helpful for future programmers (perhaps future you!) who are trying to figure out why some change was made years later.

Step 5: Create a new branch

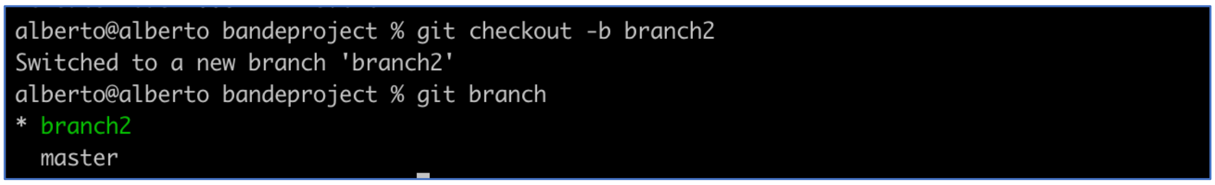
Now that you've made a new commit, let's try something a little more advanced.

Say you want to make a new feature but are worried about making changes to the main project while developing the feature. This is where git branches come in.

Branches allow you to move back and forth between 'states' of a project. Official git docs describe branches this way: ‘A branch in Git is simply a lightweight movable pointer to one of these commits.’ For instance, if you want to add a new page to your website you can create a new branch just for that page without affecting the main part of the project. Once you're done with the page, you can merge your changes from your branch into the primary branch. When you create a new branch, Git keeps track of which commit your branch 'branched' off of, so it knows the history behind all the files.

Let's say you are on the primary branch and want to **create a new branch** to develop your web page. Here's what you'll do: Run ***git checkout -b <my branch name>***. This command will automatically create a new branch and then 'check you out' on it, meaning **git will move you to that branch, off of the primary branch**.

After running the above command, you can use the ***git branch*** command to confirm that your branch was created:



```
alberto@alberto bandeproject % git checkout -b branch2
Switched to a new branch 'branch2'
alberto@alberto bandeproject % git branch
* branch2
  master
```

The branch name with the asterisk next to it indicates which branch you're on at that given time.

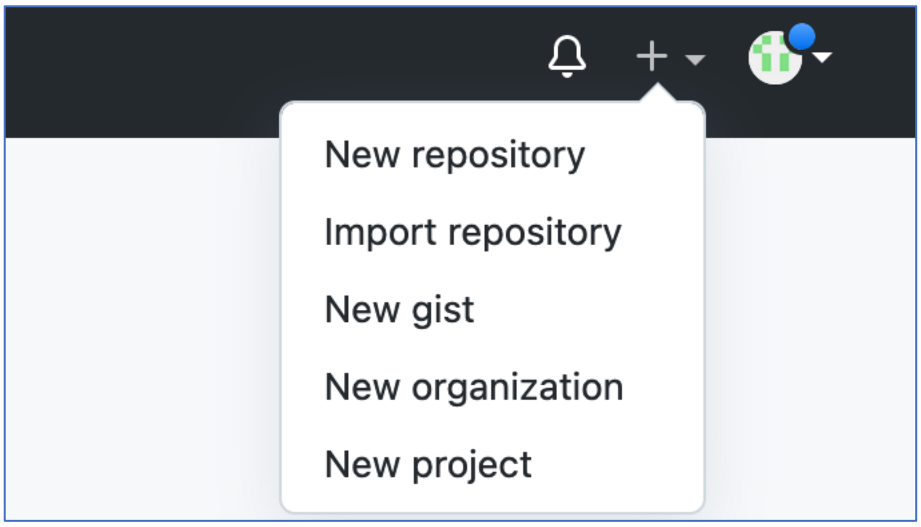
Nearly every repository has a **primary branch** that can be thought of as the *official version of the repository*. If it's a website, then the primary branch is the version that users see. If it's an application, then the primary branch is the version that users download. This isn't technically necessary (git doesn't treat any branches differently from other branches), but it's how git is traditionally used in a project.

Now, if you switch back to the primary branch and make some more commits, your new branch won't see any of those changes until you merge those changes onto your new branch.

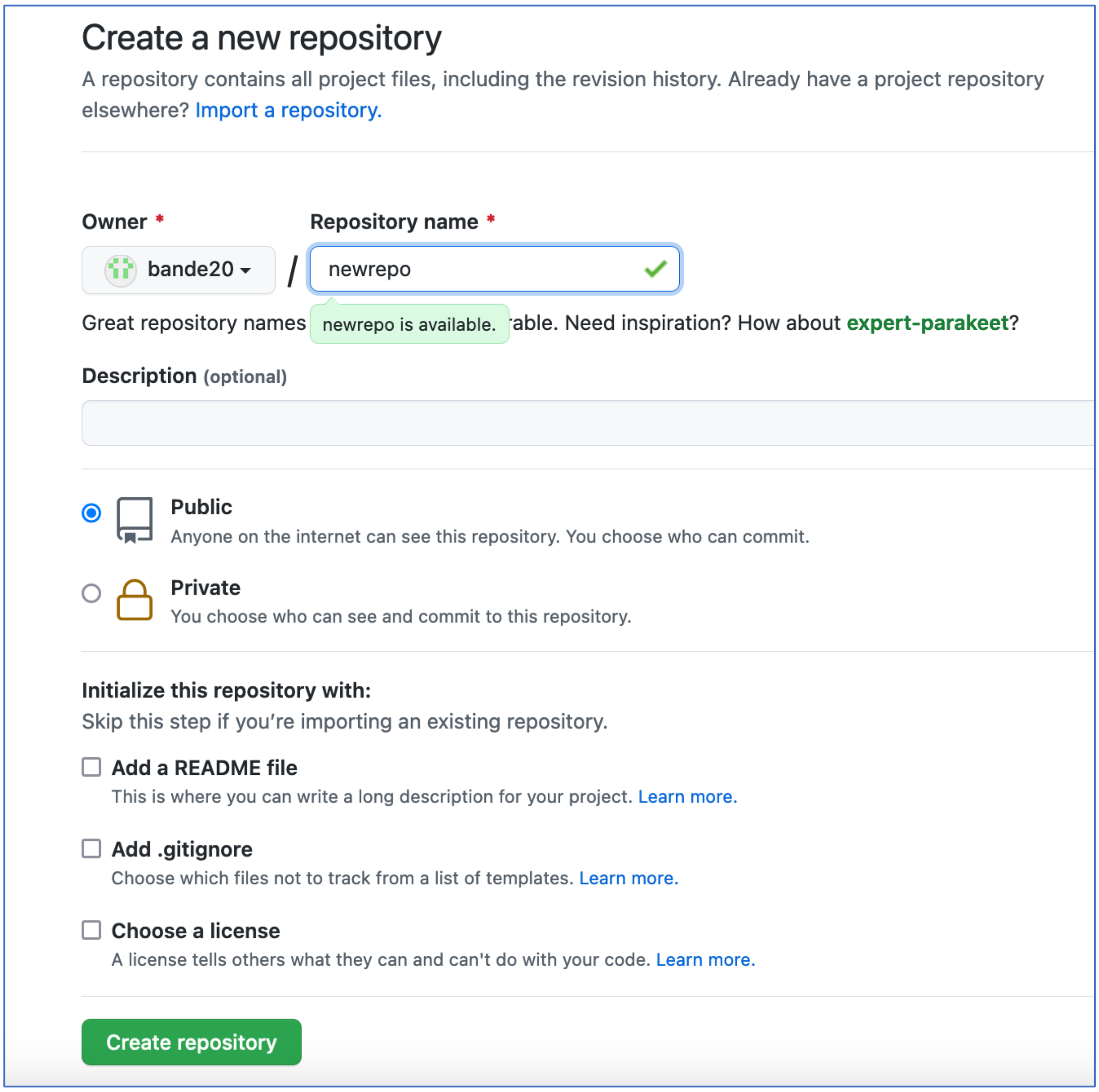
Step 6: Create a new repository on GitHub

If you only want to keep track of your code locally, you don't need to use GitHub. But if you want to work with a team, you can use GitHub to collaboratively modify the project's code.

To create a new repo on GitHub, log in and go to the GitHub home page. You can find the "New repository" option under the "+" sign next to your profile picture, in the top right corner of the navbar:



After clicking the button, GitHub will ask you to name your repo and provide a brief description:



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * bande20 / Repository name * newrepo

Great repository names newrepo is available. able. Need inspiration? How about expert-parakeet?

Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. Learn more.

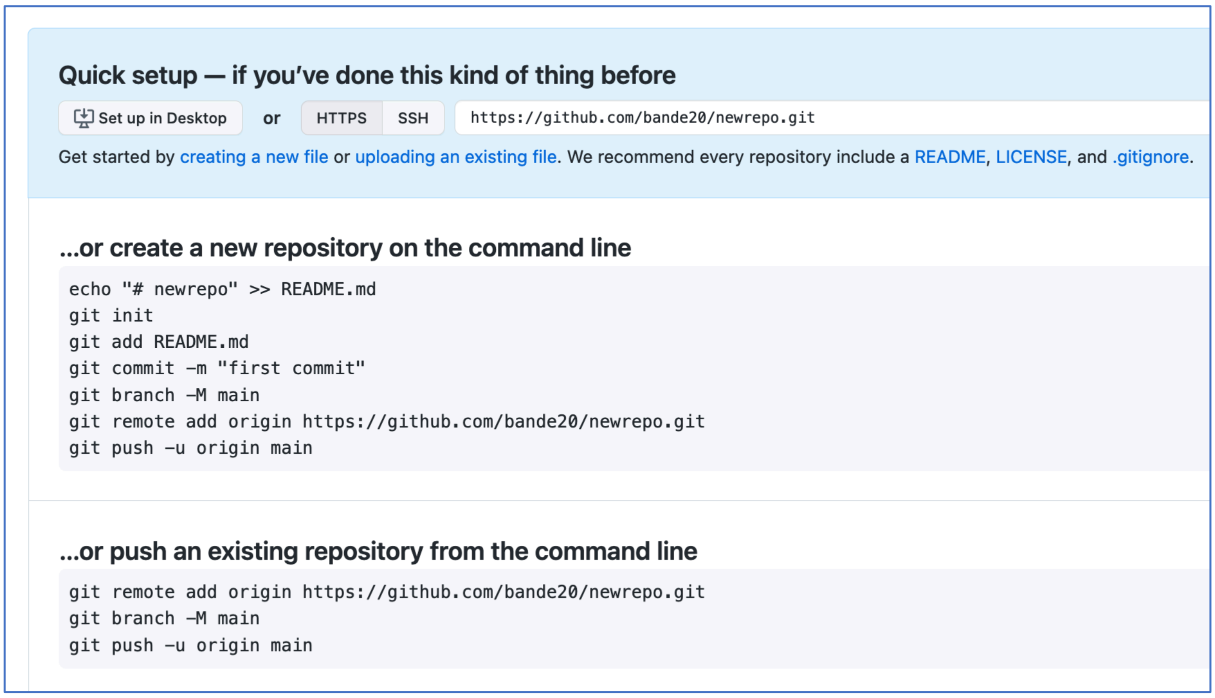
☐ Add .gitignore
Choose which files not to track from a list of templates. Learn more.

☐ Choose a license
A license tells others what they can and can't do with your code. Learn more.

Create repository

When you're done filling out the information, press the **'Create repository'** button to make your new repo.

GitHub will ask if you want to create a new repo from scratch or if you want to add a repo you have created locally. In this case, since we've already created a new repo locally, we want to push that onto GitHub so follow the **'....or push an existing repository from the command line'** section:



```
https://github.com/bande20/newrepo.git
```

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# newrepo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/bande20/newrepo.git
git push -u origin main
```

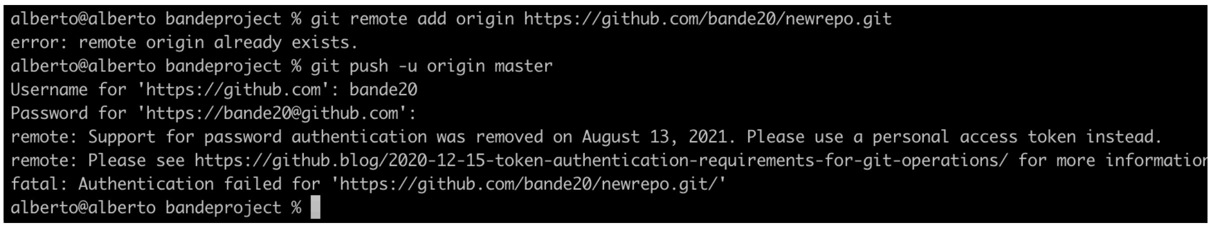
...or push an existing repository from the command line

```
git remote add origin https://github.com/bande20/newrepo.git
git branch -M main
git push -u origin main
```

git remote add origin https://github.com/bande20/newrepo.git
git push -u origin master

bande20 → username of GitHub (may differ)
master → name of the main branch for specific project (may differ)

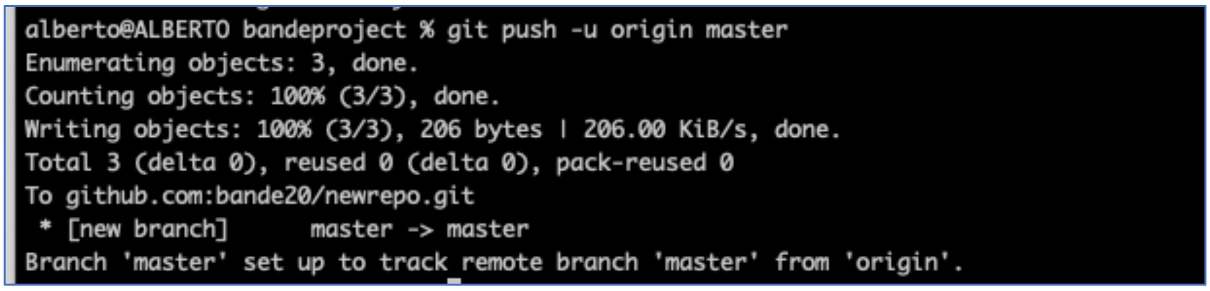
WATCH OUT!!!:



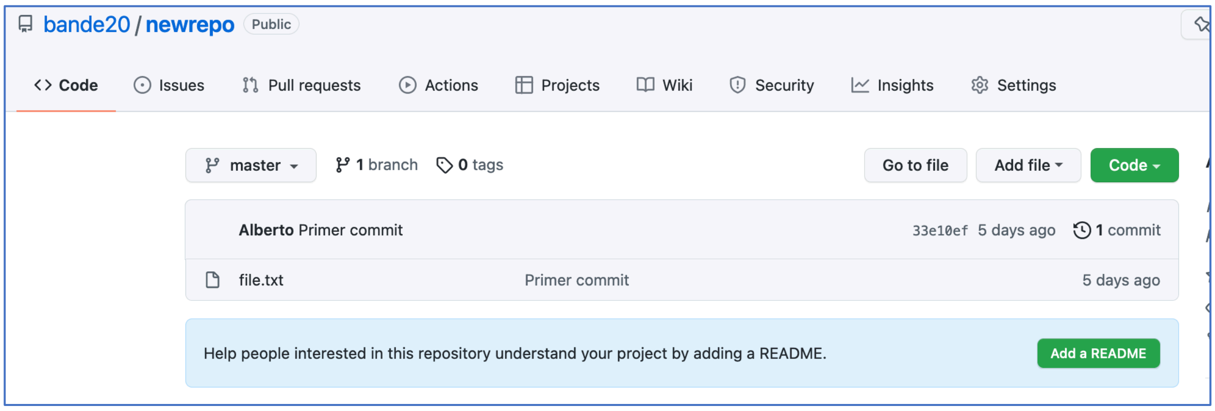
```
alberto@alberto bandeproject % git remote add origin https://github.com/bande20/newrepo.git
error: remote origin already exists.
alberto@alberto bandeproject % git push -u origin master
Username for 'https://github.com': bande20
Password for 'https://bande20@github.com':
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more informatio
fatal: Authentication failed for 'https://github.com/bande20/newrepo.git/'
alberto@alberto bandeproject %
```

We get the above error because user/passw authentication was deprecated. We must set up a private/public key authentication instead (See Appendix1).

Once we have set the correct key access, we try again:



```
alberto@ALBERTO bandeproject % git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 206 bytes | 206.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:bande20/newrepo.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```



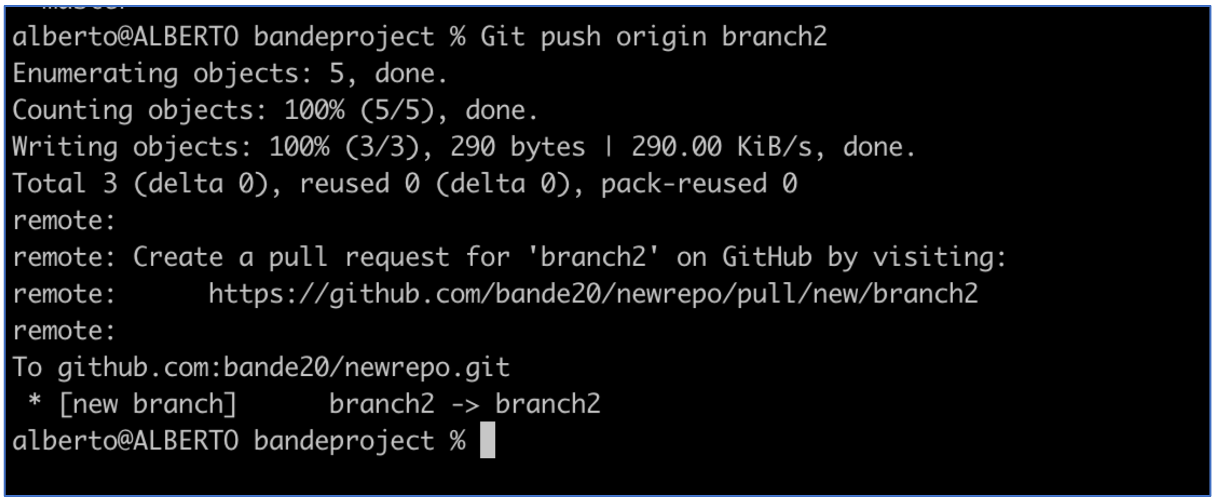
Note that the name of the local folder may be different from the repo name.

Step 7: Push a branch to GitHub

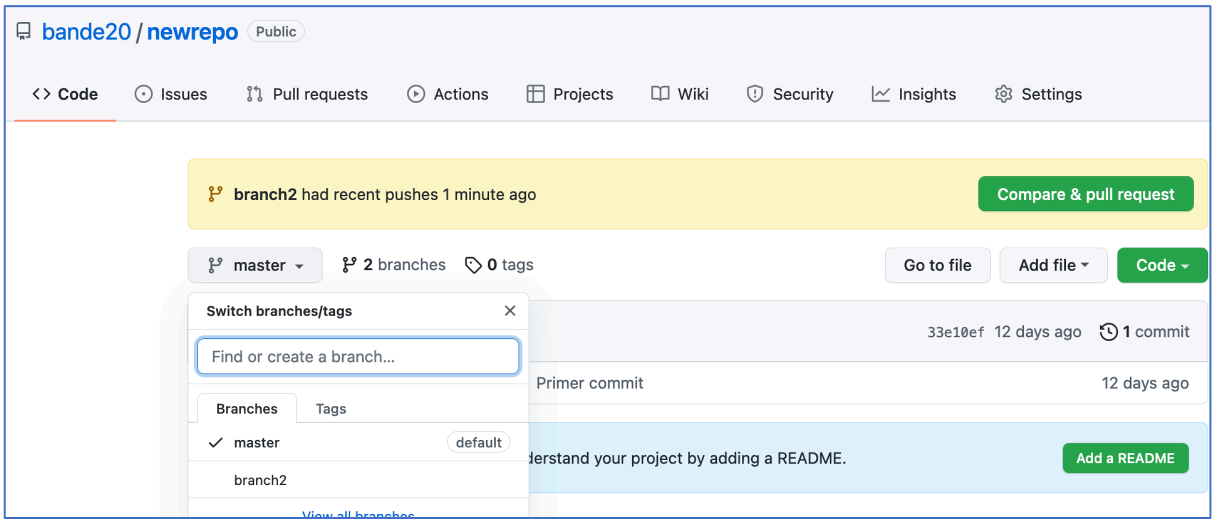
Now we'll push the commit in your branch to your new GitHub repo. This allows other people to see the changes you've made. If they're approved by the repository's owner, the changes can then be merged into the primary branch.

To push changes onto a new branch on GitHub, you'll want to run git push origin yourbranchname. GitHub will automatically create the branch for you on the remote repository

Git push origin branch2



```
alberto@ALBERTO bandeproject % Git push origin branch2
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 290 bytes | 290.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch2' on GitHub by visiting:
remote:      https://github.com/bande20/newrepo/pull/new/branch2
remote:
To github.com:bande20/newrepo.git
 * [new branch]      branch2 -> branch2
alberto@ALBERTO bandeproject %
```



branch2 had recent pushes 1 minute ago

master · 2 branches · 0 tags

Switch branches/tags

Find or create a branch...

Branches · Tags

master default

branch2

View all branches

33e10ef 12 days ago · 1 commit

Primer commit 12 days ago

derstand your project by adding a README.

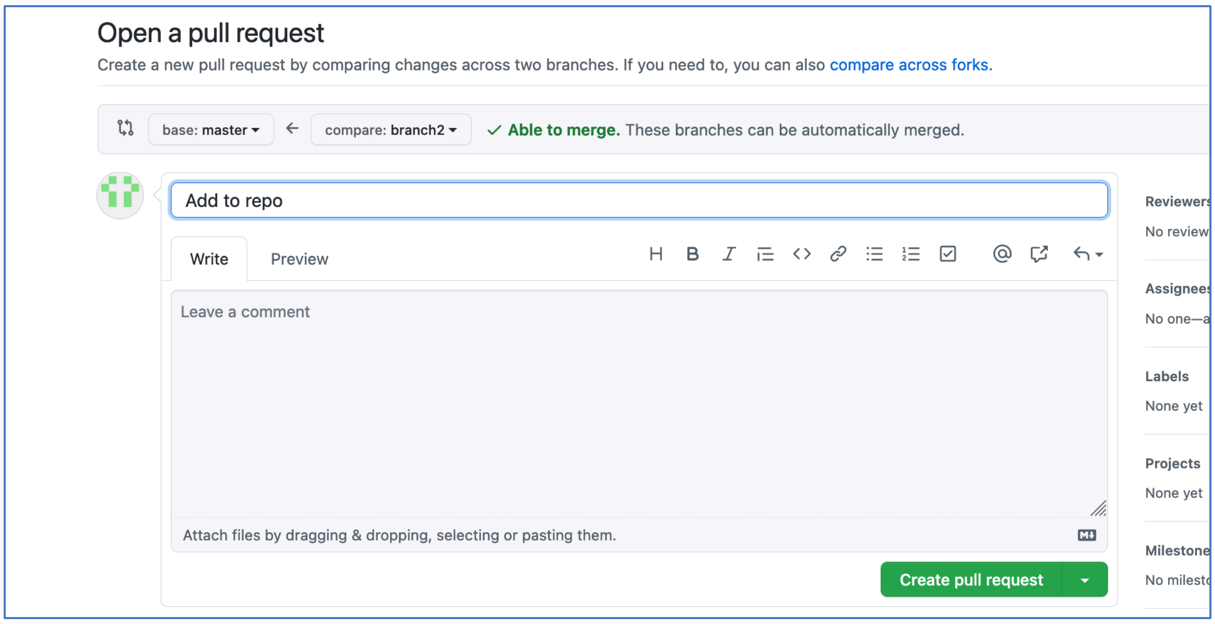
You might be wondering what that "origin" word means in the command above. What happens is that when you clone a remote repository to your local machine, git creates an alias for you. In nearly all cases this alias is called "origin." It's essentially shorthand for the remote repository's URL. So, to push your changes to the remote repository, you could've used either the command: git push git@github.com:git/git.git yourbranchname or git push origin yourbranchname.

Now click the green button in the screenshot above. We're going to make a pull request!

Step 8: Create a pull request (PR)

A pull request (or PR) is a way to alert a repo's owners that you want to make some changes to their code. It allows them to review the code and make sure it looks good before putting your changes on the primary branch.

This is what the PR page looks like before you've submitted it:



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

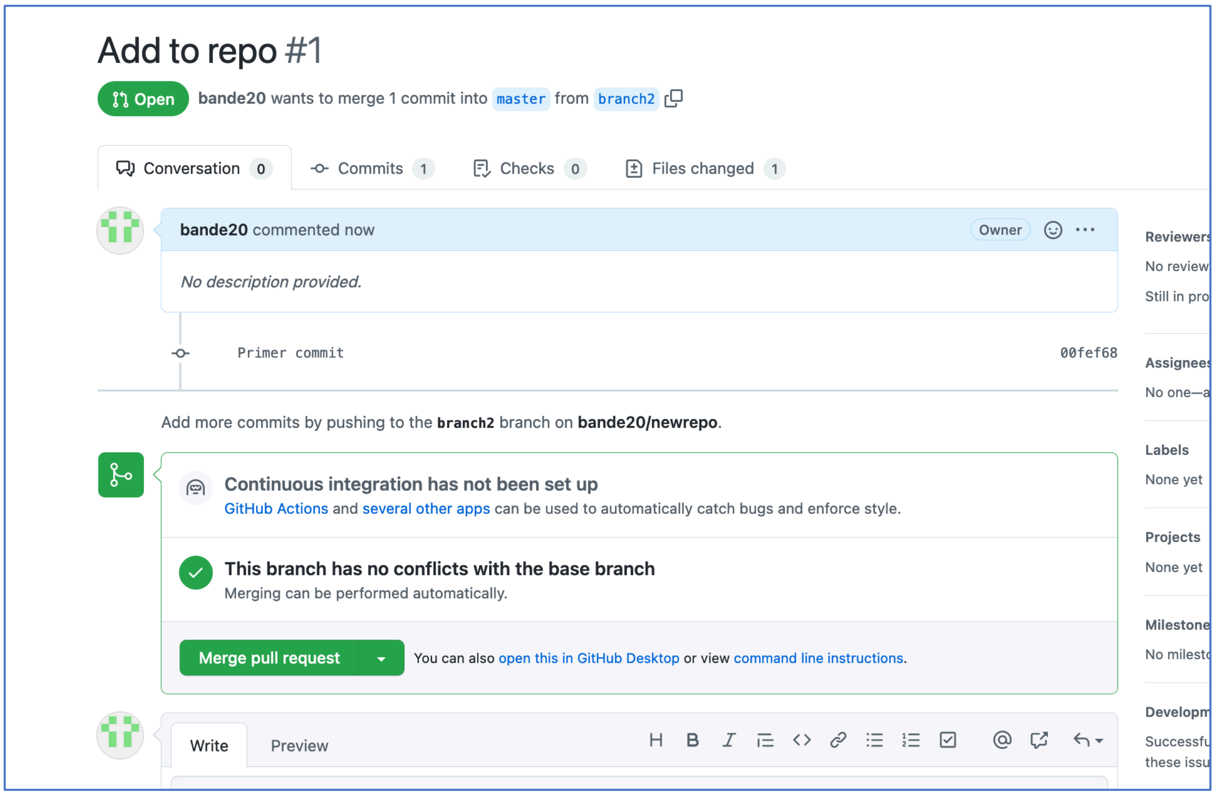
base: master ← compare: branch2 ✓ Able to merge. These branches can be automatically merged.

Add to repo

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

And this is what it looks like once you've submitted the PR request:



Add to repo #1

Open bande20 wants to merge 1 commit into master from branch2

bande20 commented now

No description provided.

```
Primer commit                                                    00fef68
```

Add more commits by pushing to the **branch2** branch on **bande20/newrepo**.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

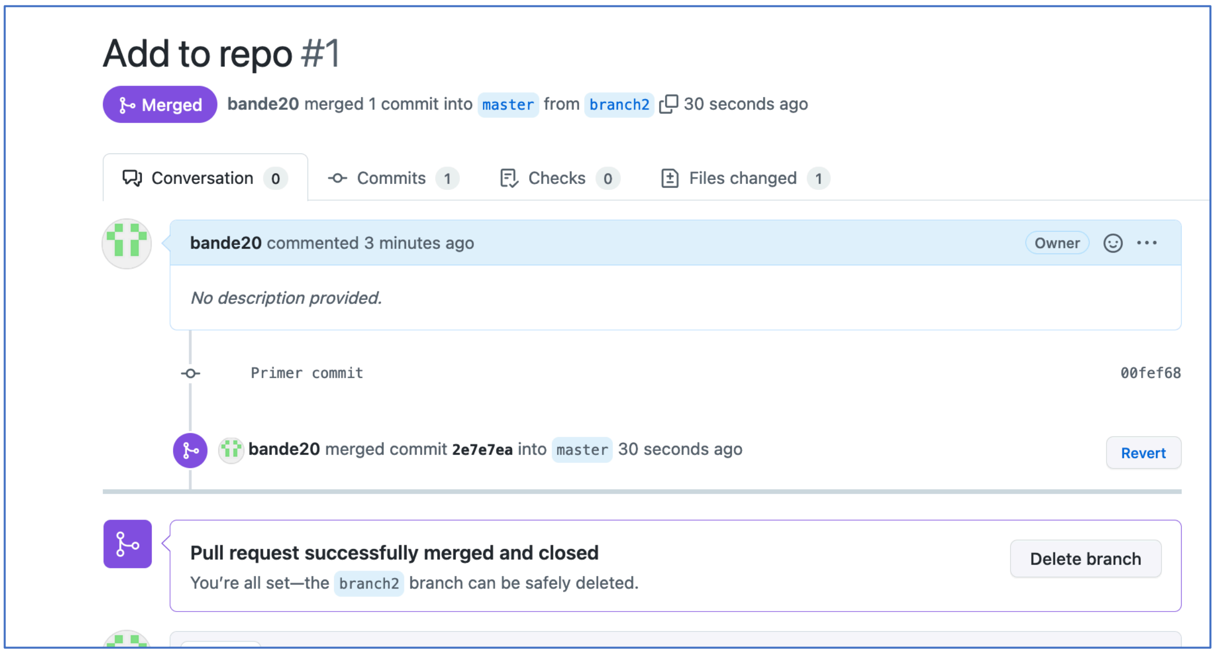
You can also open this in GitHub Desktop or view command line instructions.

You might see a big green button at the bottom that says 'Merge pull request'. Clicking this means you'll merge your changes into the primary branch..

Sometimes you'll be a co-owner or the sole owner of a repo, in which case you may not need to create a PR to merge your changes. However, it's still a good idea to make one so you can keep a more complete history of your updates and to make sure you always create a new branch when making changes.

Step 9: Merge a PR

Go ahead and click the green 'Merge pull request' button. This will merge your changes into the primary branch.


Add to repo #1

Merged bande20 merged 1 commit into master from branch2 30 seconds ago

bande20 commented 3 minutes ago Owner

No description provided.

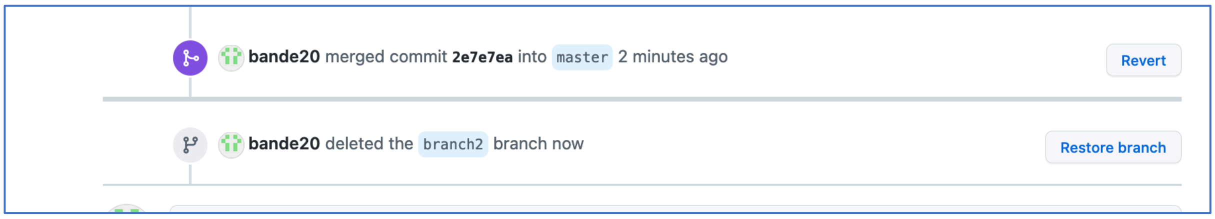
Primer commit 00fef68

bande20 merged commit 2e7e7ea into master 30 seconds ago

Pull request successfully merged and closed

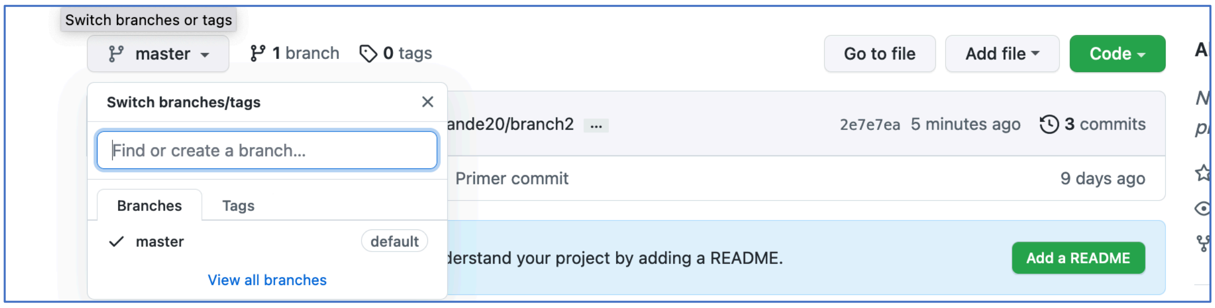
You're all set—the branch2 branch can be safely deleted.

When you're done, I recommend deleting your branch (too many branches can become messy), so hit that grey 'Delete branch' button as well.


bande20 merged commit 2e7e7ea into master 2 minutes ago

bande20 deleted the branch2 branch now

You can double check that your commits were merged by clicking on the 'Commits' link on the first page of your new repo.

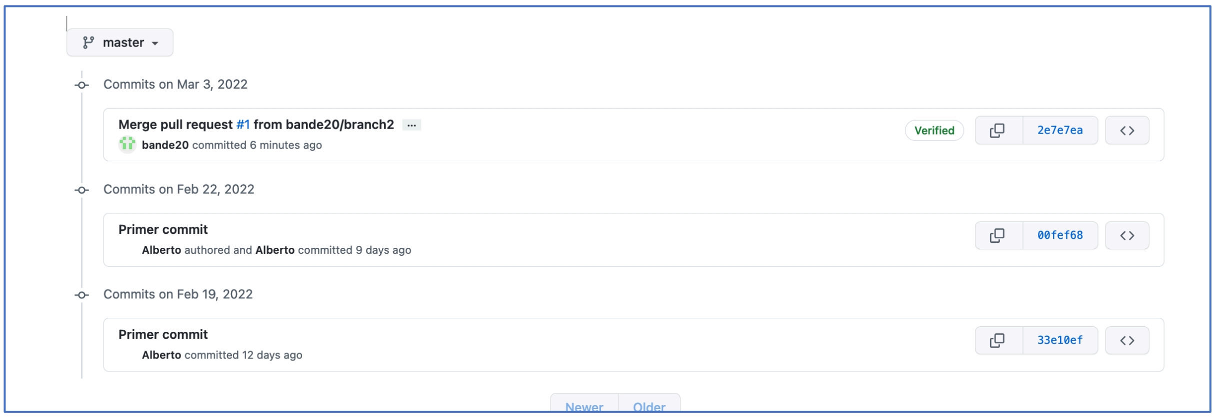


ande20/branch2 ... 2e7e7ea 5 minutes ago 3 commits

Primer commit 9 days ago

derstand your project by adding a README.

This will show you a list of all the commits in that branch



Commits on Mar 3, 2022

Merge pull request #1 from bande20/branch2 …
bande20 committed 6 minutes ago — 2e7e7ea

Commits on Feb 22, 2022

Primer commit
Alberto authored and Alberto committed 9 days ago — 00fef68

Commits on Feb 19, 2022

Primer commit
Alberto committed 12 days ago — 33e10ef

Step 10: Get changes on GitHub back to your computer

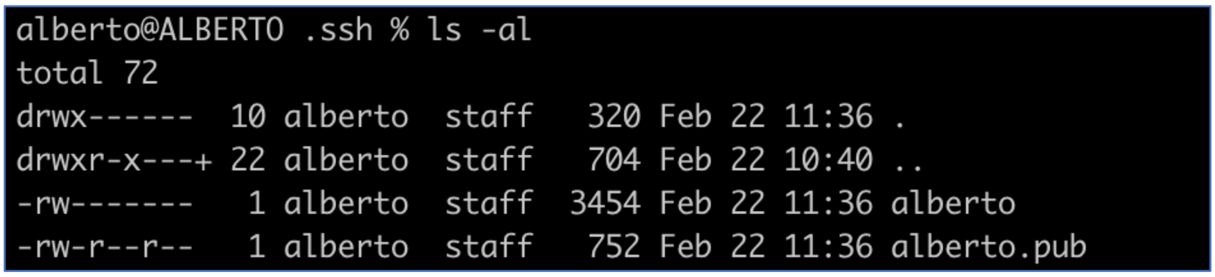
Right now, the repo on GitHub looks a little different than what you have on your local machine. For example, the commit you made in your branch and merged into the primary branch doesn't exist in the primary branch on your local machine.

In order to get the most recent changes that you or others have merged on GitHub, use the ***git pull origin master*** command (when working on the primary branch). In most cases, this can be shortened to "*git pull*".

Appendix 1: Setting Up key access

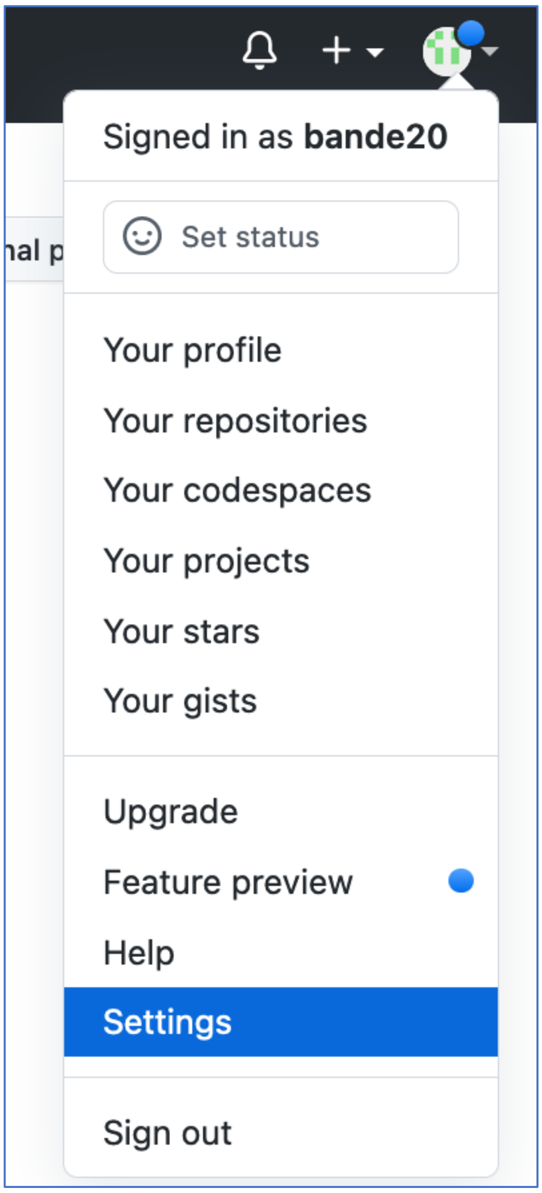
Create public-private key pair

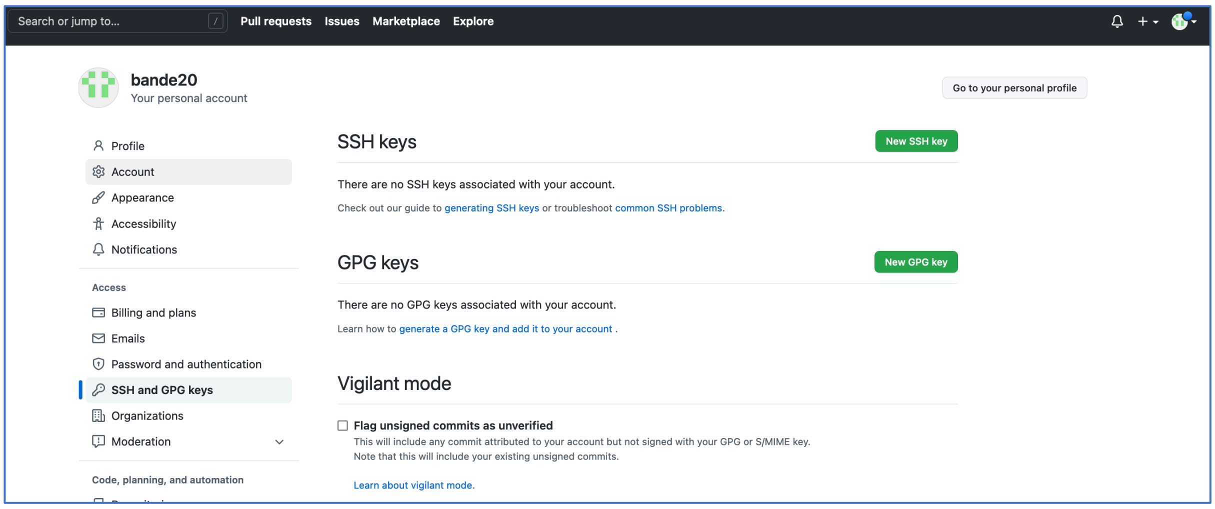
ssh-keygen -t rsa -b 4096 -f alberto -C "alberto.bandeira@gmail.com"



```
alberto@ALBERTO .ssh % ls -al
total 72
drwx------   10 alberto  staff   320 Feb 22 11:36 .
drwxr-x---+ 22 alberto  staff   704 Feb 22 10:40 ..
-rw-------    1 alberto  staff  3454 Feb 22 11:36 alberto
-rw-r--r--    1 alberto  staff   752 Feb 22 11:36 alberto.pub
```

Add the public key to GitHub





Search or jump to... / Pull requests Issues Marketplace Explore

bande20
Your personal account

Go to your personal profile

Profile
Account
Appearance
Accessibility
Notifications

Access
Billing and plans
Emails
Password and authentication
SSH and GPG keys
Organizations
Moderation

Code, planning, and automation

SSH keys — New SSH key

There are no SSH keys associated with your account.

Check out our guide to generating SSH keys or troubleshoot common SSH problems.

GPG keys — New GPG key

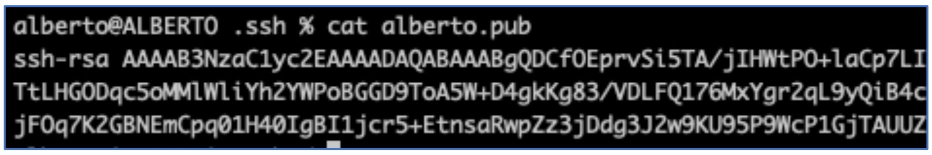
There are no GPG keys associated with your account.

Learn how to generate a GPG key and add it to your account .

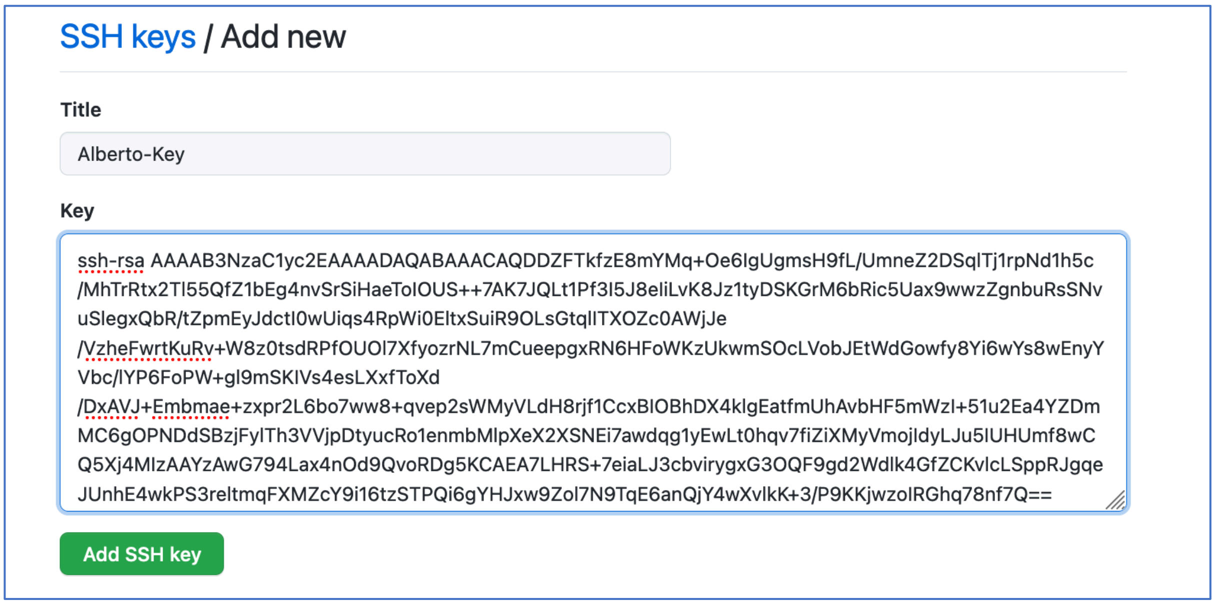
Vigilant mode

☐ **Flag unsigned commits as unverified**
This will include any commit attributed to your account but not signed with your GPG or S/MIME key.
Note that this will include your existing unsigned commits.

Learn about vigilant mode.



```
alberto@ALBERTO .ssh % cat alberto.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDCfOEprvSi5TA/jIHWtPO+laCp7LI
TtLHGODqc5oMMlWliYh2YWPoBGGD9ToA5W+D4gkKg83/VDLFQ176MxYgr2qL9yQiB4c
jFOq7K2GBNEmCpq01H40IgBI1jcr5+EtnsaRwpZz3jDdg3J2w9KU95P9WcP1GjTAUUZ
```



SSH keys / Add new

Title

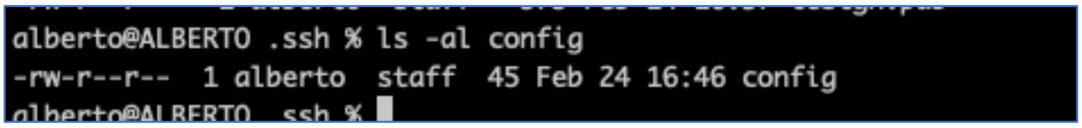
Alberto-Key

Key

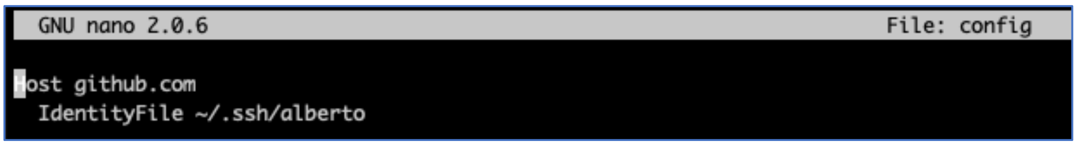
```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDDZFTkfzE8mYMq+Oe6IgUgmsH9fL/UmneZ2DSqlTj1rpNd1h5c
/MhTrRtx2Tl55QfZ1bEg4nvSrSiHaeToIOUS++7AK7JQLt1Pf3I5J8eliLvK8Jz1tyDSKGrM6bRic5Uax9wwzZgnbuRsSNv
uSlegxQbR/tZpmEyJdctI0wUiqs4RpWi0EltxSuiR9OLsGtqlITXOZc0AWjJe
/VzheFwrtKuRv+W8z0tsdRPfOUOl7XfyozrNL7mCueepgxRN6HFoWKzUkwmSOcLVobJEtWdGowfy8Yi6wYs8wEnyY
Vbc/lYP6FoPW+gl9mSKIVs4esLXxfToXd
/DxAVJ+Embmae+zxpr2L6bo7ww8+qvep2sWMyVLdH8rjf1CcxBIOBhDX4klgEatfmUhAvbHF5mWzI+51u2Ea4YZDm
MC6gOPNDdSBzjFylTh3VVjpDtyucRo1enmbMlpXeX2XSNEi7awdqg1yEwLt0hqv7fiZiXMyVmojldyLJu5lUHUmf8wC
Q5Xj4MIzAAYzAwG794Lax4nOd9QvoRDg5KCAEA7LHRS+7eiaLJ3cbvirygxG3OQF9gd2Wdlk4GfZCKvlcLSppRJgqe
JUnhE4wkPS3reltmqFXMZcY9i16tzSTPQi6gYHJxw9Zol7N9TqE6anQjY4wXvlkK+3/P9KKjwzolRGhq78nf7Q==
```

Add SSH key

Define key to use for GitHub



```
alberto@ALBERTO .ssh % ls -al config
-rw-r--r--  1 alberto  staff  45 Feb 24 16:46 config
alberto@ALBERTO .ssh %
```



```
  GNU nano 2.0.6                                      File: config

Host github.com
  IdentityFile ~/.ssh/alberto
```

```
Host github.com
  IdentityFile ~/.ssh/alberto
```

Test the connection

ssh -vT git@github.com → Always use the user ‘git’.

```
debug1: rekey in after 134217728 blocks
debug1: Will attempt key: /Users/alberto/.ssh/alberto RSA
SHA256:SGCQLx4whLDJ7SIZPdXfG6Dhgio7cHgtCOYcMyTgijY explicit
debug1: SSH2_MSG_EXT_INFO received
….
debug1: Offering public key: /Users/alberto/.ssh/alberto RSA
SHA256:SGCQLx4whLDJ7SIZPdXfG6Dhgio7cHgtCOYcMyTgijY explicit
debug1: Server accepts key: /Users/alberto/.ssh/alberto RSA
SHA256:SGCQLx4whLDJ7SIZPdXfG6Dhgio7cHgtCOYcMyTgijY explicit
debug1: Authentication succeeded (publickey).
Authenticated to github.com ([140.82.121.3]:22).
…
debug1: client_input_channel_req: channel 0 rtype exit-status reply
0
Hi bande20! You've successfully authenticated, but GitHub does not
provide shell access.
debug1: channel 0: free: client-session, nchannels 1
```

Appendix 2: Glossary of commands