Creating a new file in a new Git branch

Asked 8 months ago Active 8 months ago Viewed 616 times

0

I'm still experimenting with Git.

Right now I have only a local `master` branch and a remote `origin/master` branch on Gitlab.

What I've been doing so far:

- Code something new on `master`
- Run `git commit -m "added something new"`
- Run `git push origin master`

I wanted to create a new branch, so I did:

- `git checkout -b newBranch`

After this command I'm already "checked out" on my new branch, right? And I could confirm that with `git branch -a`

So I created a new file (using VSCode file explorer) `example.txt` (on my new branch).

And then I did:

- `git checkout master`

So I could go back to my `master` branch.

And at that point I was expecting that I would no longer be able to see the file, since it was created using a different branch. But the file is still there.

What am I missing?

git git-branch

asked Oct 30 '19 at 16:28

cbdeveloper
7,558 4 24 67

2 The file is not tracked by git, since you never ran `git add` – Paolo Oct 30 '19 at 16:31

@UnbearableLightness Do I need to add and commit the file to my `newBranch` ? After that will it be "invisible" to the `master` ? – cbdeveloper Oct 30 '19 at 16:32

That's exactly what happened! Thanks! – cbdeveloper Oct 30 '19 at 16:35

Possible duplicate of Modified files in a git branch are spilling over into another branch – phd Oct 30 '19 at 17:01

2 Answers

Active | Oldest | Votes

1

The file you created has not been staged or committed, so it is an untracked file. Untracked files do not yet belong to any branch(es); they exist on your working tree and remain there when you switch from branch to branch.

(To fully understand this, it may help to familiarize yourself with the three general storage areas in git - the worktree, the index (sometimes called staging area), and the database. Going into detail about that would kind of be out of scope for answering this question, but I do recommend reading up on this topic.)

There is a word of warning that goes with that: a file that is committed on one branch might not exist on another branch. For example, if you go back to `newBranch` and commit your file

```
git checkout newBranch
git add example.txt
git commit
```

now `example.txt` exists on `newBranch` but not on `master`. If you check out `master`, the file will go away as you expected.

That means that if, on `master`, you create another `example.txt` (in the same directory so that the path is exactly the same), this new file is untracked even though the same path refers to an existing, committed file on another branch.

If you then tried to checkout to `newBranch` git should refuse with a warning that you have data which would be lost by checking out.

Really this is just one special case of "trying to change branches with uncommitted changes", but it is worth being aware of given your current task.

answered Oct 30 '19 at 16:43

Mark Adelsberger
28.2k 2 17 32

0

Another Possible Solution:

You can use stash command of git.

What git stash do?

git stash is used to clipboard for your changes.

If you have to switch context - e.g. because you need to work on an urgent bug - you need to get these changes out of the way. You shouldn't just commit them, of course, because it's unfinished work.

Example:

```
git stash
```

Your working copy is now clean: all uncommitted local changes have been saved on this kind of "clipboard" that Git's Stash represents. You're ready to start your new task (for example by pulling changes from remote or simply switching branches).

1. You created one file example.txt on branch newBranch.
2. Now urgently you need to switch another branch master then before switch just apply following command.

   git add example.txt

   git stash
3. Now when you switch master. Then example.txt file will disappear.
4. When you go back your branch newBranch then apply following command.

   git stash apply

then you can see you file example.txt.

answered Oct 30 '19 at 18:27

Jeet Kumar
515 3 11

Yes. You should not commit that files which are not completed yet. In that case stash command will be helpfull more than commit that files. – Jeet Kumar Oct 30 '19 at 18:31

Two things: First - While stash can be useful, it doesn't really have anything to do with the question (which is about how files relate to branches). Although technically they are based on specific commits, stashes also do not belong to any branch and you can apply them (or at least attempt to, though there may be conflicts) anywhere. – Mark Adelsberger Nov 4 '19 at 14:47

Second - stashes are decent temporary holding areas for changes, but they have important limitations. Just making a commit is often a better solution, and ultimately statements like "you just shouldn't commit ... unfinished work" are opinion-based and up to specific teams t decide. To *very* temporarily get work out of the way, stash is fine. But stashes are strictly local, so they don't benefit from the "free backup" that git remotes give you; and they're poorly organized and easily forgotten if they stick around for long or if you ever have more than one in a repo – Mark Adelsberger Nov 4 '19 at 14:51