

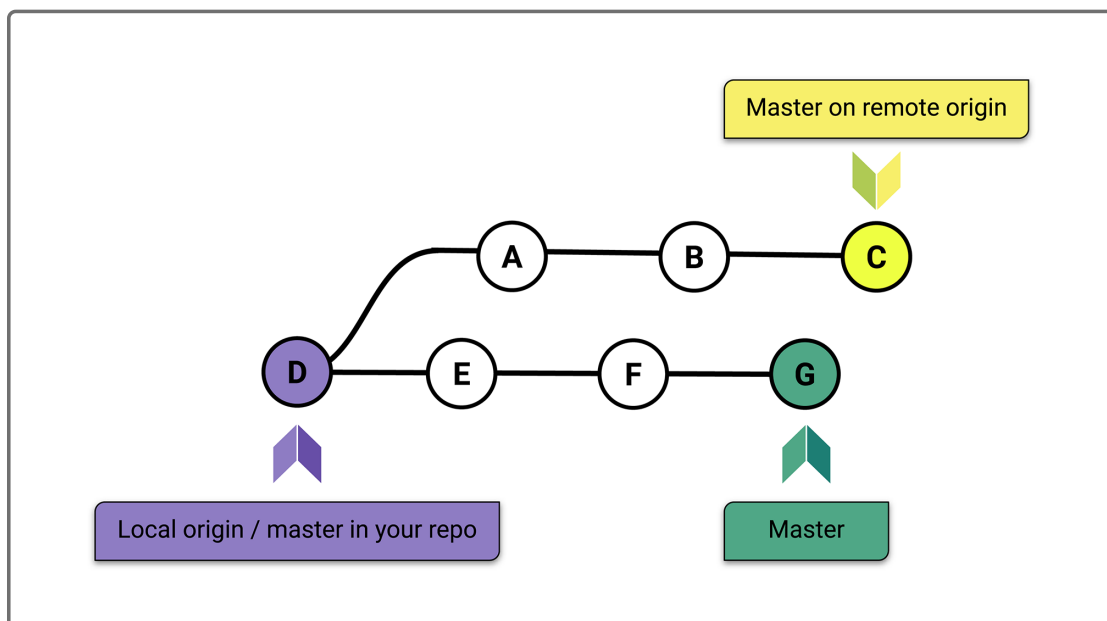
## 2.1.6 Interact with GitHub

It's important to keep in mind that the local repo (on your computer) is different than the remote repo (on GitHub). Let's see this in action.

Because we made a change to the `README.md` on the `master` branch in GitHub, let's take a look in the local `master` branch and see if it reflects that. To do that, type `git checkout master` at the command prompt.

As you can see, the `README.md` file you just created in GitHub isn't in your local folder. This is because the remote repo and local repo are out of sync. Let's figure out how this happened and learn how to fix it.

The following diagram shows the relationship between local and remote branches and commits:



In the image, the circles represent commits and the letters indicate unique commits or save points. As you can see, the remote `master` branch is not in sync with the `master` branch on the local repo. The commits are different on the two repos, which makes the versions different and out of sync.

So how do we sync our local repo with the remote repo? Let's find out!

## Update the Local Repo Using Git Pull

In order to fix this, we need to use a new Git command: `git pull`. This is the opposite operation of `git push`, which we used previously to upload our codebase to the remote server on GitHub. The `git pull` command "pulls down" (or downloads) the code from the remote server to the repo on our computer, whereas `git push` pushes up (or uploads) the local codebase to the remote repo.

To help visualize the difference between the `git pull` and `git push` commands, check out this video before moving on:



Before we update the local repo, it's always a good practice to see if any local changes need to be staged and committed. Let's first make sure we're in our `master` branch by using the following command:

```
git checkout master
```

Then we'll run the status check command:

```
git status
```

Remember that this command verifies if our active branch is "clean," which means that any changed files in the branch have been staged and committed. If the branch is not clean, then you must `add` and `commit` the file changes first.

## NERD NOTE

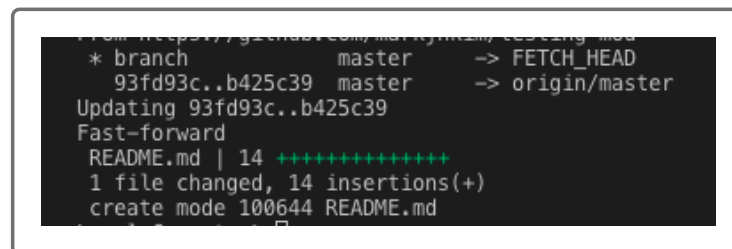
So what is all this talk about branches and trees? These are just terms that visually express the folder hierarchy of directories and the version control process of Git.

Now that we have a clean working tree, we're ready to **pull** or update the local branch by downloading any new changes or commits that are in the remote branch.

Let's **pull** the remote **master** branch that contains the **README.md** file into our local **master** branch. To do that, follow these steps:

1. Checkout into the **master** branch.
2. Check the branch status. If clean, proceed. Otherwise, stage and commit file changes.
3. Pull remote into active branch using **git pull origin master**.

You should see the following screen after a successful **pull**:

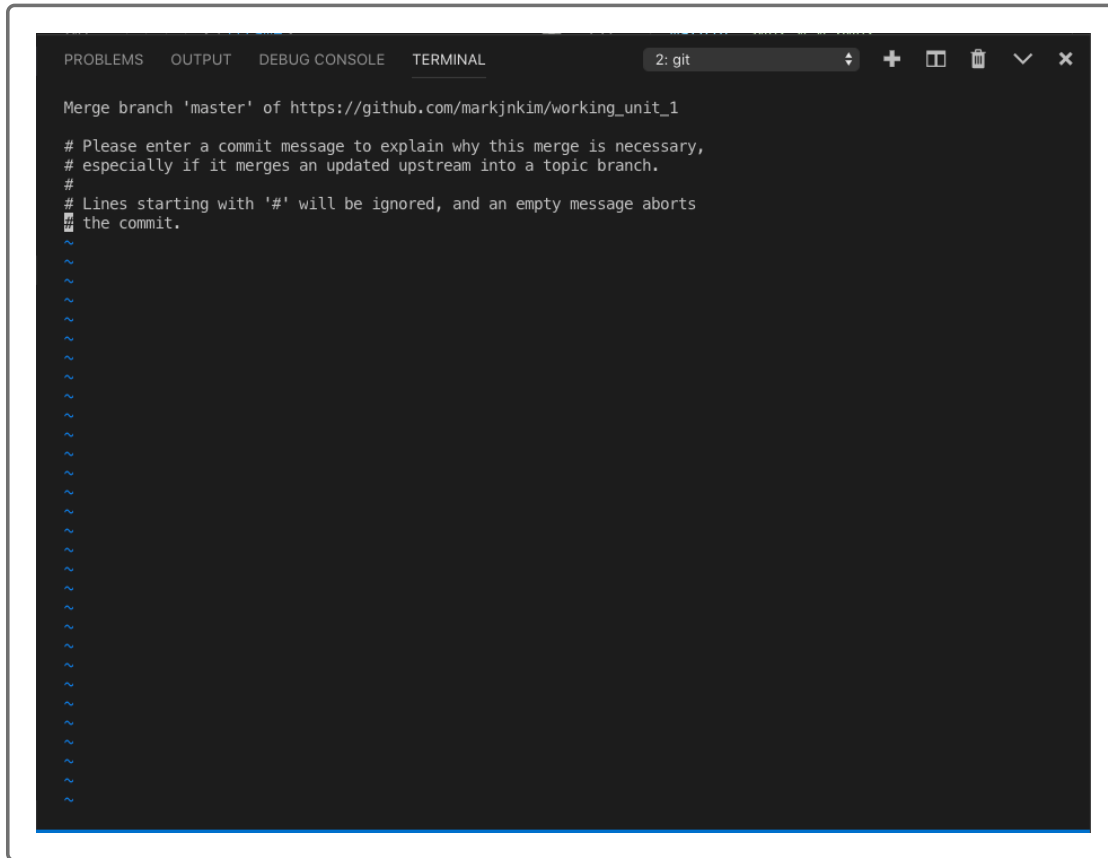


```
from https://github.com/markjmain/testing_repo
* branch      master      -> FETCH_HEAD
  93fd93c..b425c39  master    -> origin/master
Updating 93fd93c..b425c39
Fast-forward
 README.md | 14 ++++++++
 1 file changed, 14 insertions(+)
 create mode 100644 README.md
```

This image shows that there are 14 new lines of code in the **README.md** file, as evidenced by the file name, the number 14, and the green plus signs.

Sometimes when you pull from a remote branch, instead of the success response, you'll see something in the terminal that looks like the following

image:

A screenshot of a terminal window with a dark background. The terminal title bar shows '2: git'. The prompt 'Merge branch 'master' of https://github.com/markjnkim/working\_unit\_1' is displayed. Below it, instructions for entering a commit message are shown: '# Please enter a commit message to explain why this merge is necessary, # especially if it merges an updated upstream into a topic branch. # Lines starting with '#' will be ignored, and an empty message aborts the commit.' The prompt character is a blue tilde '~'. There are several more tilde characters on subsequent lines, indicating the cursor is at the start of each line in the text editor.

Git has opened the text editor Vim inside the terminal and is asking you to enter a commit message. This happens when there's a local commit that differs from the commit history in the remote branch. Usually when this happens, you can just exit out of Vim without entering the commit message.

To exit out of Vim, follow these instructions:

1. Type `:` to enter command mode in Vim. (Vim has several modes; the command mode lets you type commands.)
2. Type `q` to quit.
3. Press Enter.

Or you can press the following keys: Shift + Z + Z

This will exit Vim and return you to the command prompt in terminal, which should display the success response shown earlier.

## DEEP DIVE ▼

---

### Use Terminal Inside VS Code

By now, you're familiar with using terminal on your computer, but did you know that VS Code has a built-in terminal that offers all the same capabilities?

One of the advantages of using the terminal window in VS Code is that when we `checkout` to a different branch or perform a `merge` operation, VS Code has an Explorer panel that displays a folder tree showing all the files and folders in the repo:

## ▼ OPEN EDITORS

# style.css assets/... U

<> privacy-policy.h... U

# secondary-style... U

x ⓘ README.md

## ▼ FSFO-MODULE-2

▼ assets ●

▼ CSS ●

# secondary-styles... U

# styles.css U

> images

<> index.html

<> privacy-policy.html U

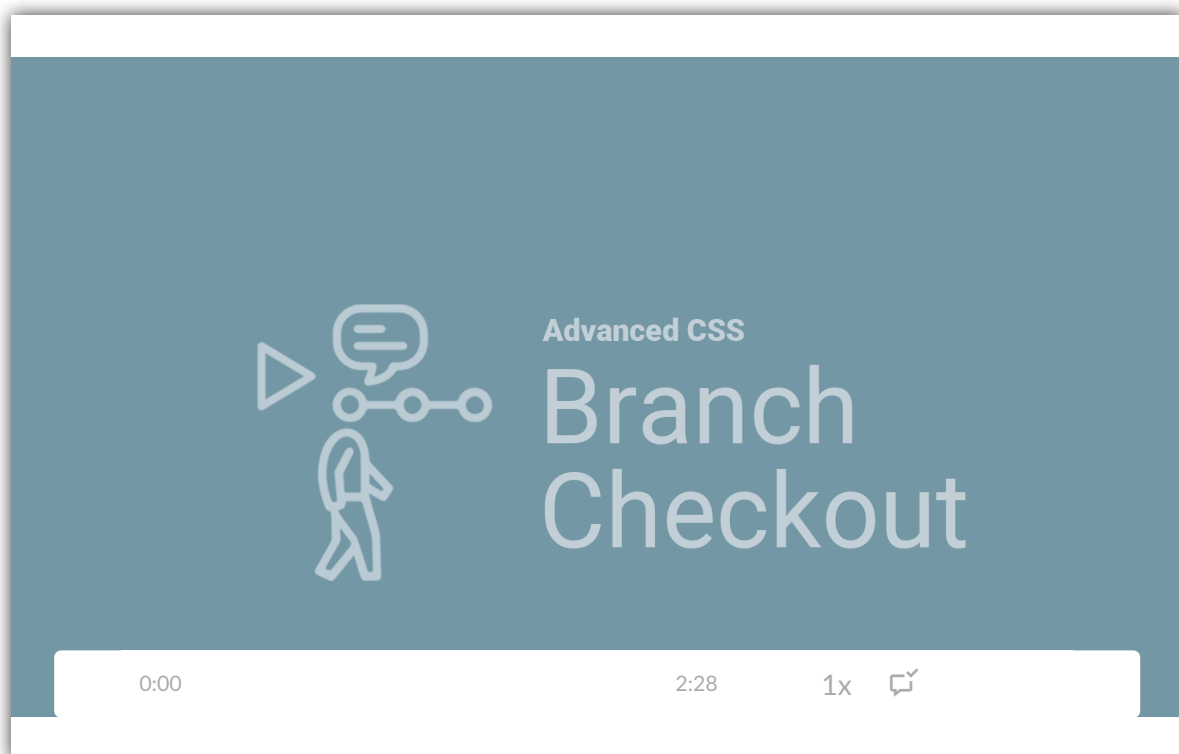
ⓘ README.md

## SHOW HINT

The folder tree in the Explorer panel is interactive. You can expand or close folders, and open or add files. A file can also be dragged and dropped into the split screen to view multiple files simultaneously.

Also note that the active branch is displayed on the bottom left status bar in VS Code. This helps you keep track of the version being edited or copied. It's important to always make sure you're not working in the `master` branch. Run Buddy's management team would be very unhappy if the production environment got corrupted!

To see how VS Code changes when we switch Git branches, check out this video:





## ON THE JOB

It's good practice to get into the habit of doing a `git pull` to update your local development environment whenever you start editing code. Team members could have pushed enhancements or bug fixes up to the remote development environment since the last time you pulled down the codebase. If your branch is up to date, Git will let you know that a `pull` is not necessary.

## Using Git Pull

Let's dive a bit deeper into the following `pull` command:

```
git pull origin master
```

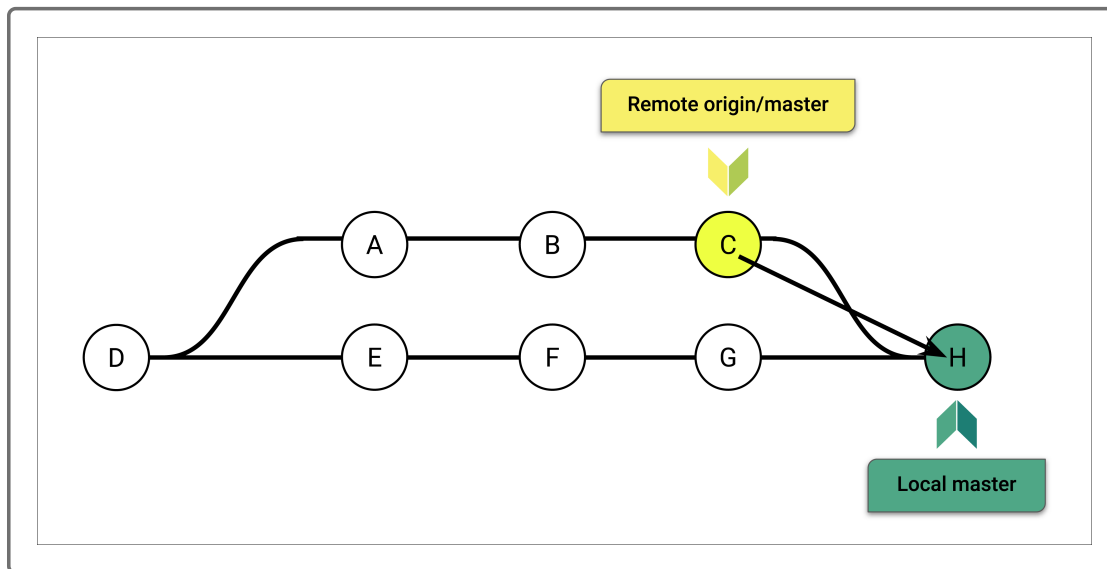
The `pull` command directs the download process to occur. The words `origin` and `master` designate the source of the download or `pull`. `origin` describes the remote (aka GitHub) repo, and `master` designates the branch or version in the remote repo we'd like to pull. Please note that if the `<branch-name>` that's being pulled does not exist on the remote repo, the `pull` process will give an error and not occur.

## PAUSE

How does the `pull` command know where the download will go to?

[Show Answer](#)

As you can see in the following diagram, the remote repo has converged with the local repo in the "H" merge commit:



Let's dive deeper into the `pull` process. The `pull` command is actually a combination of two different operations: `git fetch` and `git merge`.

## Git Fetch

The `git fetch` command updates the local repo with any changes in the remote repo since the last `fetch` command. A use case for `git fetch` would be to look at a team member's feature branch for a review process or to help find a bug.

The remote branch is downloaded by using the `fetch` command:

```
git fetch origin <remote-branch-name>
```

Next, use the following command to switch to that branch:

```
git checkout <remote-branch-name>
```

These operations create a separate, isolated version of the remote branch on the local machine. No updates or merges occurred to any other local branch during this process. This code can be rendered in the browser, tested, code reviewed, or changed as needed. These changes can then be staged, committed, and pushed back to the remote repo.

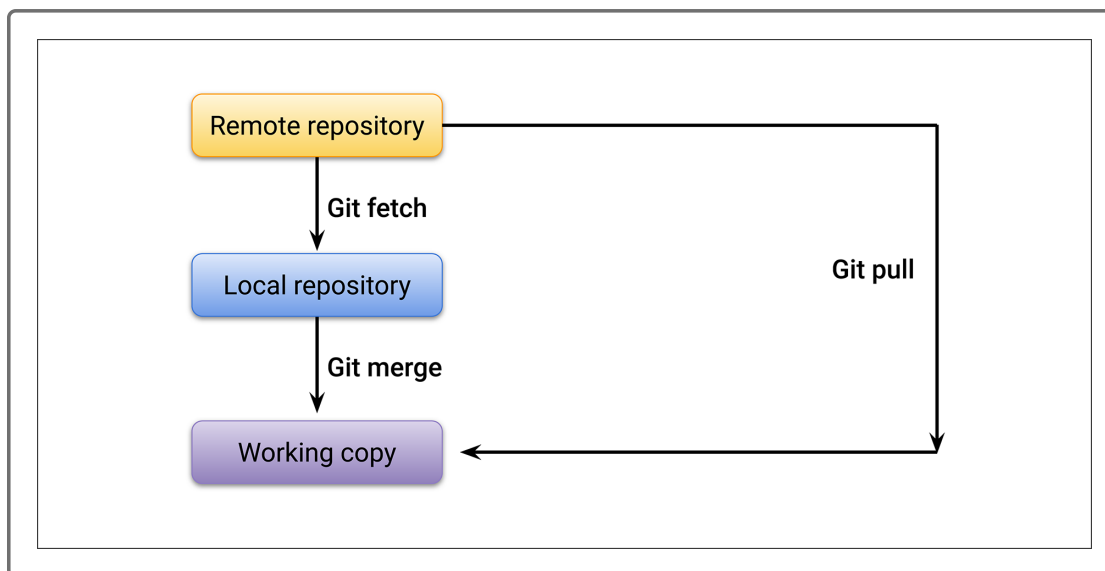
## Git Merge

The `git merge` operation, on the other hand, updates the active branch with changes from the target branch. The target branch refers to the branch that contains the changes we need to add. Once they are merged, the active branch has been updated with the changes in the target branch. The target branch however is unchanged.

```
git merge <target-branch>
```

In this operation, Git will determine any differences in the code between the active branch and the target branch, and update the active branch with those changes. Conversely, the target branch will remain the same.

The following diagram illustrates how these commands work:



After the merge, the active branch (in our case, the `master` branch) will receive the new file updates from the remote `master` branch and all of the commits associated with that branch.

## DEEP DIVE ▼

With the update of the `master` branch, the `develop` branch is no longer current. Because we'll be working consistently with the `develop` branch as we add new features, it is important that we make sure it's updated.

So how do we update our local `develop` branch? There are a number of ways to do this:

- Use `git pull origin master`. Similar to how we updated our local `master` branch, we can also update the local `develop` branch by checking out into the `develop` branch first.
- Use `git merge master`. We can checkout into the `develop` branch, then merge the changes from the `master` to copy over the `README.md` file into the `develop` branch.
- Make the changes in GitHub. We can make the same change to the `README.md` file in the remote `develop` branch, then perform another `git pull origin develop` in our local `develop` branch to update it from the remote branch. However, it is strongly recommended **NOT** to do it this way as it could cause several merge conflicts, which will be explained shortly.

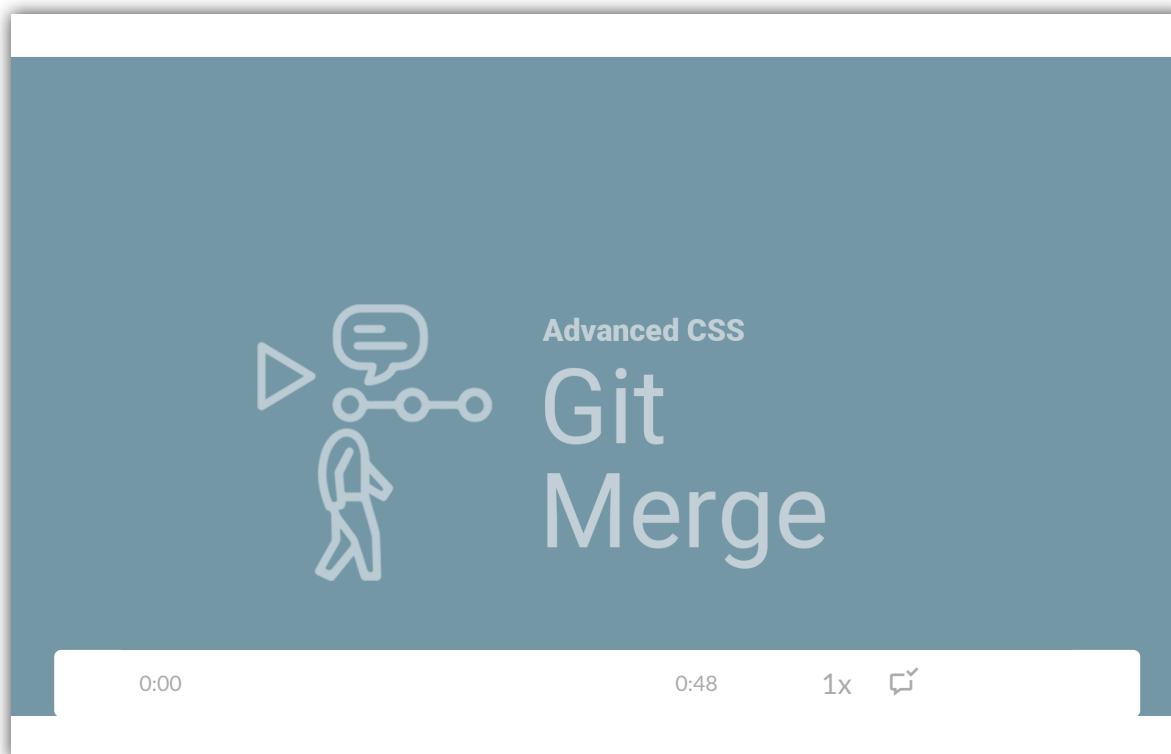
Let's go with the second option, using `git merge master`, so we can practice merging branches locally and not just from GitHub. The first option is completely acceptable as well, however.

Go ahead and type the following commands to update the `develop` branch, pressing Enter after each one:

```
git checkout develop  
git merge master
```

To verify that the merge was successful, look for the `README.md` file in the `develop` branch. If you were able to successfully add it, congrats!

Merging code in Git can be a scary process, as it leads you to believe you'll ruin your codebase. More often than not, this looks more intimidating than it actually is, and this video walkthrough can help you visualize the process for handling situations like this:



© 2020 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.