# Git Pull - How to Override Local files with Git Pull

camperbot Aug 4

Jun 2016
1 / 2
Jun 2016
Aug 2017

## When do you need to overwrite local files?

If you feel the need to discard all your local changes and just reset/overwrite everything with a copy from the remote branch then you should follow this guide.

Important: If you have any local changes, they will be lost. With or without `--hard` option, any local commits that haven't been pushed will be lost.
If you have any files that are not tracked by Git (e.g. uploaded user content), these files will not be affected.

### The Overwrite workflow:

To overwrite your local files do:

```
git fetch --all
git reset --hard <remote>/<branch_name>
```

For example:

```
git fetch --all
git reset --hard origin/master
```

### How it works:

`git fetch` downloads the latest from remote without trying to merge or rebase anything.

Then the git reset resets the master branch to what you just fetched. The `--hard` option changes all the files in your working tree to match the files in `origin/master`.

### Additional Information:

It's worth noting that it is possible to maintain current local commits by creating a branch from `master` or whichever branch you want to work on before resetting:

For Example:

```
git checkout master
git branch new-branch-to-save-current-commits
git fetch --all
git reset --hard origin/master
```

After this, all of the old commits will be kept in `new-branch-to-save-current-commits`. Uncommitted changes however (even staged), will be lost. Make sure to stash and commit anything you need.

### Git Pull

`git pull` is a Git command used to update the local version of a repository from a remote.

It is one of the four commands that prompts network interaction by Git. By default, `git pull` does two things.

1. Updates the current local working branch (currently checked out branch)
2. Updates the remote tracking branches for all other branches.

`git pull` fetches ( `git fetch` ) the new commits and merges **( `git merge` )** these into your local branch.

This command's syntax is as follows:

```
# General format
git pull OPTIONS REPOSITORY REFSPEC

# Pull from specific branch
git pull REMOTE-NAME BRANCH-NAME
```

in which:

- **OPTIONS** are the command options, such as `--quiet` or `--verbose` . You can read more about the different options in the **Git documentation**
- **REPOSITORY** is the URL to your repo. Example: **https://github.com/freeCodeCamp/freeCodeCamp.git**
- **REFSPEC** specifies which refs to fetch and which local refs to update
- **REMOTE-NAME** is the name of your remote repository. For example: *origin* .
- **BRANCH-NAME** is the name of your branch. For example: *develop* .

**Note**

If you have uncommitted changes, the merge part of the `git pull` command will fail and your local branch will be untouched.

Thus, you should *always commit your changes in a branch before pulling* new commits from a remote repository.

## Using git pull

Use `git pull` to update a local repository from the corresponding remote repository. Ex: While working locally on `master` , execute `git pull` to update the local copy of `master` and update the other remote tracking branches. (More information on remote tracking branches in the next section.)

But, there are a few things to keep in mind for that example to be true:

- The local repository has a linked remote repository
    - Check this by executing `git remote -v`
    - If there are multiple remotes, `git pull` might not be enough information. You might need to enter `git pull origin` or `git pull upstream` .
- The branch you are currently checked out to has a corresponding remote tracking branch
    - Check this by executing `git status` . If there is no remote tracking branch, Git doesn't know where to pull information *from* .

## Distributed Version Control

Git is a **Distributed Version Control System** (DVCS). With DVCS, developers can be working on the same file at the same time in separate environments. After *pushing* code up to the shared remote repository, other developers can *pull* changed code.

#### Network interactions in Git

There are only four commands that prompt network interactions in Git. A local repository has no awareness of changes made on the remote repository until there is a request for information. And, a remote repository has no awareness of local changes until commits are pushed.

The four network commands are:

- `git clone`
- `git fetch`
- `git pull`
- `git push`

#### Branches in DVCS

When working with Git, it can feel like there are lots of copies of the same code floating all over the place. There are different versions of the same file on each branch. And, different copies of the same branches on every developer's computer and on the remote. To keep track of this, Git uses something called **remote tracking branches** .

If you execute `git branch --all` within a Git repository, remote tracking branches appear in red. These are read-only copies of the code as it appears on the remote. ( When was the last network interaction that would have brought information locally? Remember when this information was last updated. The information in the remote tracking branches reflects the information from that interaction.)

With **remote tracking branches** , you can work in Git on several branches without network interaction. Every time you execute `git pull` or `git fetch` commands, you update **remote tracking branches** .

### git fetch plus git merge

`git pull` is a combination command, equal to `git fetch` + `git merge` .

#### git fetch

On its own, `git fetch` updates all the remote tracking branches in local repository. No changes are actually reflected on any of the local working branches.

#### git merge

Without any arguments, `git merge` will merge the corresponding remote tracking branch to the local working branch.

#### git pull

`git fetch` updates remote tracking branches. `git merge` updates the current branch with the corresponding remote tracking branch. Using `git pull` , you get both parts of these updates. But, this means that if you are checked out to `feature` branch and you execute `git pull` , when you checkout to `master` , any new updates will not be included. Whenever you checkout to another branch that may have new changes, it's always a good idea to execute `git pull` .

### git pull in IDEs

Common language in other IDES may not include the word `pull` . If you look out for the words `git pull` but don't see them, look for the word `sync` instead.

### fethcing a remote PR (Pull Request) in to local repo

For purposes of reviewing and such, PRs in remote should be fetched to the local repo. You can use `git fetch` command as follows to achieve this.

```
git fetch origin pull/ID/head:BRANCHNAME
```

ID is the pull request id and BRANCHNAME is the name of the branch that you want to create. Once the branch has been created you can use `git checkout` to switch to that branch.

### Points to Note

`git pull` is not to be confused with **PR(Pull Request)** . `git pull` is a command offered by git. While Pull Request is a feature provided by github.

Pull Request is raised by a developer to ensure the code from the developer's branch is merged back to the main branch. Before the code is merged back to the main branch, a reviewer will review the code in the pull request to ensure the code meets all the necessary standards.

**CLOSED AUG 6, '17**