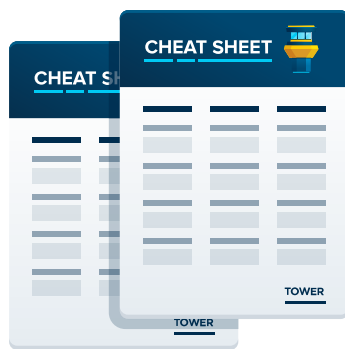


[Home](#) [Blog](#)[More Learning Content](#)[Try Tower - For Free](#)

in mind when inspecting remote branches and commits!

Let's now look at the fine but important differences between "fetch" and "pull".



The Git Cheat Sheet

No need to remember all those commands and parameters: get our popular

"Git Cheat Sheet" - for free!

[Download Now for Free](#)

Fetch

```
$ git fetch origin
```

git fetch really only downloads new data from a remote repository - but it doesn't integrate any of this new data into your working files. Fetch is great for getting a fresh view on all the things that happened in a remote repository.

[Home](#) [Blog](#)[More Learning Content](#)[Try Tower - For Free](#)

Pull

```
$ git pull origin master
```

git pull, in contrast, is used with a different goal in mind: to update your current HEAD branch with the latest changes from the remote server. This means that pull not only downloads new data; it also directly **integrates** it into your current working copy files. This has a couple of consequences:

- › Since "git pull" tries to merge remote changes with your local ones, a so-called "merge conflict" can occur. Check out our in-depth tutorial on [How to deal with merge conflicts](#) for more information.
- › Like for many other actions, it's highly recommended to start a "git pull" only with a clean working copy. This means that you should *not* have any uncommitted local changes before you pull. Use Git's Stash feature to [save your local changes temporarily](#).

TIP

Auto-Fetching + Auto-Stashing in Tower

In case you are using the [Tower Git client](#), you don't