Git Questions & Answers

Frequently asked questions around Git & Version Control

What's a "detached HEAD" in Git?

It might very well be that you'll never come across this "mysterious" state in your Git career. However, if you do one day, you'd probably like to know what a "detached HEAD" is - and how you might have arrived at that state.

Understanding how "checkout" works

With the "**git checkout**" command, you determine which revision of your project you want to work on. Git then places all of that revision's files in your working copy folder.

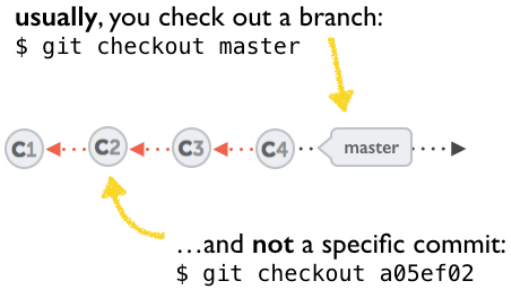
Normally, you use a *branch name* to communicate with "git checkout":

```
$ git checkout development
```

However, you *can* also provide the *SHA1 hash* of a specific commit instead:

```
$ git checkout 56a4e5c08
Note: checking out '56a4e5c08'.

You are in 'detached HEAD' state...
```

This exact state - when a specific *commit* is checked out instead of a *branch* - is what's called a "detached HEAD".



The problem with a detached HEAD

The HEAD pointer in Git determines your current working revision (and thereby the files that are placed in your project's working directory). Normally, when checking out a proper branch name, Git automatically moves the HEAD pointer along when you create a new commit. You are automatically on the newest commit of the chosen branch.

When you instead choose to check out a *commit hash*, Git won't do this for you. The consequence is that when you make changes and commit them, these **changes do NOT belong to any branch**.
This means they can easily get lost once you check out a different revision or branch: not being recorded in the context of a branch, you lack the possibility to access that state easily (unless you have a brilliant memory and can remember the commit hash of that new commit...).

When a detached HEAD shows up

There are a handful of situations where detached HEAD states are common:

› Submodules are indeed checked out at specific commits instead of branches.

› Rebase works by creating a *temporary* detached HEAD state while it runs.

Where a detached HEAD should **not** show up

Additionally, another situation might spring to mind: what about going back in time to try out an older version of your project? For example in the context of a bug, you want to see how things worked in an older revision.

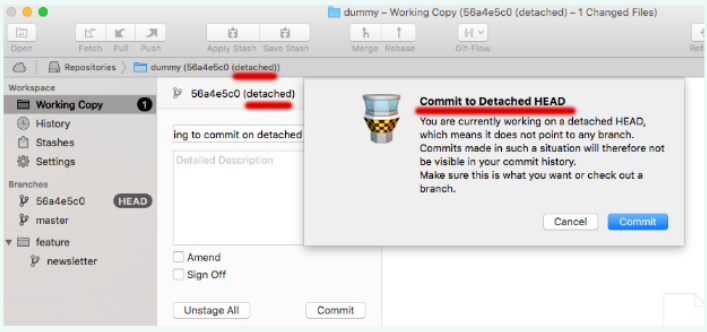
This is a perfectly valid and common use case. However, you don't have to maneuver yourself into a detached HEAD state to deal with it. Instead, remember how simple and cheap the whole concept of branching is in Git: you can simply create a (temporary) branch and delete it once you're done.

```
$ git checkout -b test-branch 56a4e5c08

...do your thing...

$ git checkout master
$ git branch -d test-branch
```

TIP

Detached HEAD handling in Tower

In case you are using the Tower Git client, the app will prominently inform you when you're in a detached HEAD state. More importantly, Tower will also explicitly warn you in case you're trying to commit in such a state.



Learn More

› Check out the chapters Checking Out a Local Branch and Submodules in our free online book

› More frequently asked questions about Git & version control

Get our popular **Git Cheat Sheet** for free!

You'll find the most important commands on the front and helpful best practice tips on the back. Over 100,000 developers have downloaded it to make Git a little bit easier.

☐ Yes, send me the cheat sheet and sign me up for the Tower newsletter. It's free, it's sent infrequently, you can unsubscribe any time.

☐ I have read and accept the Privacy Policy. I understand that I can unsubscribe at any time.

Your email address

About Us

As the makers of Tower, the best Git client for Mac and Windows, we help over 100,000 users in companies like Apple, Google, Amazon, Twitter, and Ebay get the most out of Git.

Just like with Tower, our mission with this platform is to help people become better professionals.

That's why we provide our guides, videos, and cheat sheets (about version control with Git and lots of other topics) for free.