The 'git submodule' Command

Learn about the "git submodule" command with the help of an example.

We'll cover the following

Git submodules solve these external repository dependency issues with a little overhead. Now that you understand local and remote repositories, it will be *much* easier to grasp how submodules work.

Tracking copies

Git submodule commands are Git commands used to track copies of *other* repositories within *your* repository. The tracking is under your control (so you decide when it gets updated, regardless of how the other repository moves on), and the tracking is done within a file that is stored with your Git repository.

> **Pay Attention Here!**
>
> Git submodules can be very confusing if you don't follow a few basic patterns or understand how they work. So it's worth paying attention to this.

Let's make this more clear with a walkthrough.

You are going to assume you have the `alicelib` repository created in the previous lesson (https://www.educative.io/collection/page/10370001/6075350136651776/6572936616476672).

Create Bob's repository (`bob_repo`):

```
cd ..
mkdir bob_repo
cd bob_repo
git init
echo 'source alicelib' > file1
git add file1
git commit -am 'sourcing alicelib'
echo 'do something with alicelib 9  experimental' >> file1
git commit -am 'using alicelib experimental'
cat file1
```

Terminal 1

Terminal

Now you have Alice's repository referenced in `bob_repo` 's code, but `bob_repo` does not have a link to `alice_repo` 's code.

The first step to including `alicelib` in `bob_repo` is to initialize submodules:

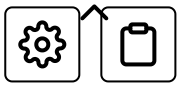
```
git submodule init
```

Once a `git submodule init` has been performed, you can `git submodule add` the submodule you want:

```
git submodule add ../alicelib
```

Terminal 1

Terminal



A new file has been created (`.gitmodules`), and the folder `alicelib` has been created:

```
14  ls -a
```

`alicelib` has been cloned just as any other Git repository would be anywhere else:

```
15  ls -a alicelib/
```

But the `.gitmodules` file tracks where the submodule comes from:

```
16  cat .gitmodules
```

If you get confused, Git provides a useful `git submodule status` command that works in a similar way to the standard `git status` command:

```
17  git submodule status
```

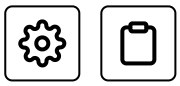
Possible questions

You may have some questions at this point such as:

- How do you get to the experimental branch?
- What happens if Alice's branch changes? Does my code automatically update?
- What if I make a change to `alicelib` within *my* repository submodule checkout? Can I push those to Alice's? Can I keep those private to my repository?
- What if there are conflicts between these repositories?

And so on. I certainly had these questions when it came to Git submodules. With some trial and error, it took me some time to understand what was going on. I really recommend playing with these

simple examples to get the relationships clear in your mind.



Back

A Worked Example

Next

Git Tracks the Submodule's State

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/the-git-submodule-command__git-submodules__learn-git-the-hard-way)