Cloning a Project With Submodules

Learn to clone a project with submodules with the help of a “recursive” flag.

We'll cover the following

- The git clone --recursive flag

Submodules have a special status within Git repositories. Since they are both included within a repository and at the same time reference as a remote repository, a simple clone will *not* check out the included submodule. Let’s show that:

```
cd ../..
git clone bob_repo bob_repo_cloned
cd bob_repo_cloned
ls -1
cd alicelib
ls
cd ..
```

Terminal 1

Terminal

`alicelib`'s content is not there. Confusingly, `git submodule status` gives you a little clue of what's going on here.

```
git submodule status
```

The dash (or minus sign) at the front indicates the submodule is not checked out. Only by running a `git submodule init` and a `git submodule update` can you retrieve the appropriate submodule repository:

```
git submodule init
git submodule update
git submodule status
```

The submodule status has no dash, and a commit ID has been added to the output (`969b840` ...).

The `git clone --recursive` flag

Fortunately, there is an easier way. You can clone the repository with a `--recursive` flag to automatically init and update any submodules (and submodules of those submodules ad infinitum) within the cloned repository:

```
cd ..
git clone --recursive bob_repo bob_repo_cloned_recursive
```

In case you're wondering if you clone a repository with `--recursive` and there are no submodules, then Git does not complain.

Terminal 1

Terminal

← Back

Git Tracks the Submodule’s State

Next →

What Are Pull Requests?

☑ Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/cloning-a-project-with-submodules__git-submodules__learn-git-the-hard-way)