A Worked Example

To get a better understanding of "submodules," learn about them with the help of an example.

We'll cover the following

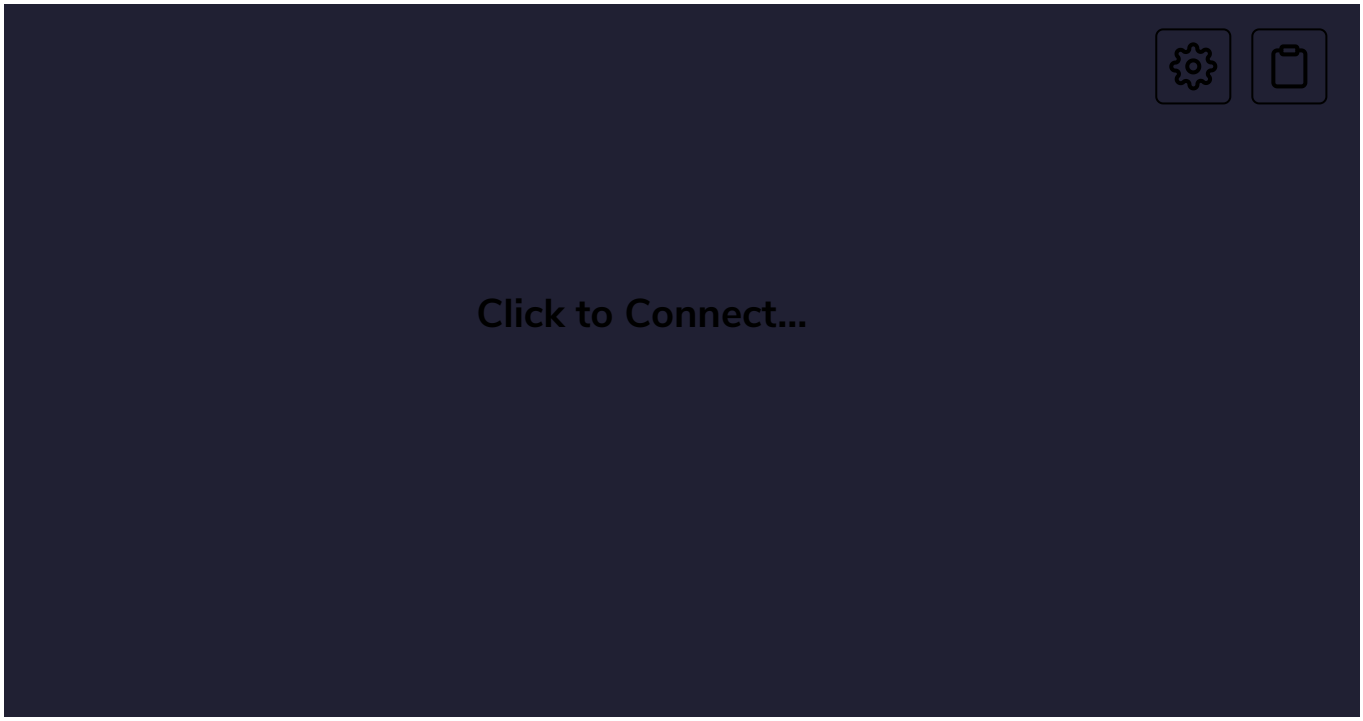
Maintainer

Let’s say Alice maintains a library:

```
mkdir -p lgthw_submodules
cd lgthw_submodules
mkdir alicelib
cd alicelib
git init
echo 'A' > file1
git add file1
git commit -am 'A'
git checkout -b experimental      # Branch to experimental
echo 'C - EXPERIMENTAL' >> file1
git commit -am EXPERIMENTAL
git checkout master
echo 'B' >> file1
git commit -am 'B'
```

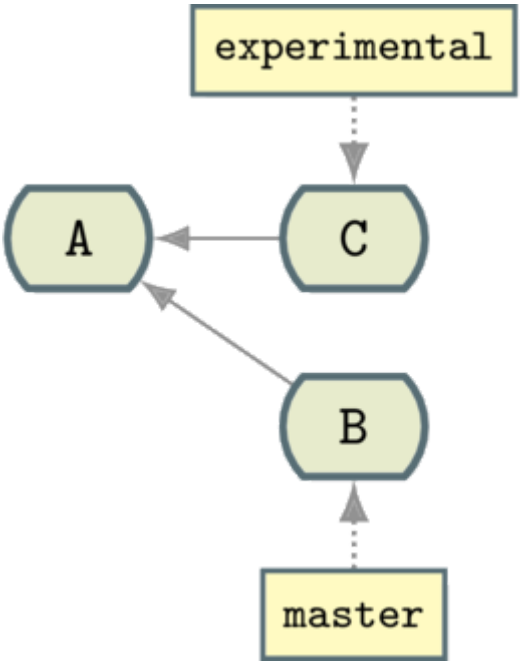
Terminal 1

Terminal



History

Alice’s library’s history looks like this:



Alice's library's history

Using code from a specific branch

Now Bob wants to use Alice's library for his own code but specifically wants to use what's on the `experimental` branch.

One option is to copy the code over directly, but that seems to be against the spirit of Git.

If an improvement is made on the `experimental` branch or Bob wants to move later to follow what's on the `master` branch, then he must copy over the code he wants. For one file, it might be manageable. But managing this will be completely impractical for a more realistic and large project.

Another option is to check out the code in another folder and link to it in some predictable way in the code (e.g., your code might run `source ../alice_experimental`). Again, this causes management problems, as the user checking out the source must remember to keep code outside this Git repository in a certain place for it all to work.

← Back

Introduction: Git Submodules

Next →

The 'git submodule' Command

Mark as Completed

Report an Issue

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