The 'git fetch' Command

Learn how to use the "git fetch" command.

We'll cover the following

Fetching from remote

First, you will look at fetching from the remote.

The command `git fetch` gets the latest changes from the remote repository and copies them into the local repository, updating the local copies of any branches that have been changed.

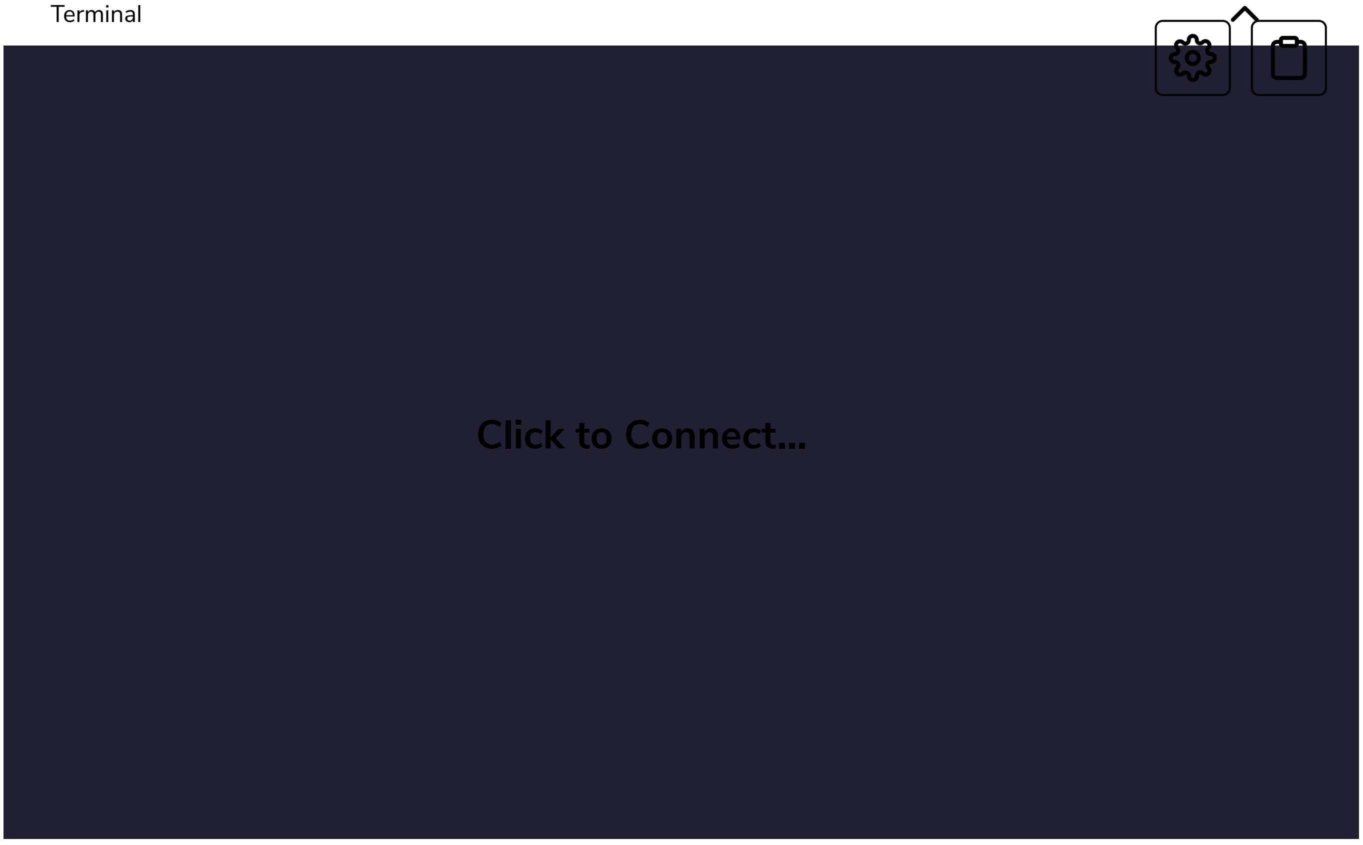
Crucially, these changes are not *mixed* with your local branches but are kept in a separate place. So if you have a `master` branch on the remote Git repository and a `master` branch on your local Git repository, then your local one will not be affected.

First, make a change to the origin's repository:

```
cd ..
cd git_origin
echo 'fetchable change' >> file1
git commit -am fetchable
```

Terminal 1

Terminal



Then go to the cloned repository and fetch the changes on the `master` branch, which is on the `origin` remote (`git_origin`). Type the following two commands in the terminal provided above.

```
cd ../git_cloned
git fetch origin master
```

You did not fetch all the changes from the remote but specified that you wanted the changes on the remote `master` branch brought down by adding the branch to the command.

The above output means that the `origin`'s `master` has been brought into this repository's references. This branch is now referred to locally as: `origin/master`.

This has not affected the local `master` branch at all. Check out this:

```
git log
```

You can see this repository's view of all the branches it is aware of by running this:

```
git branch --all
```

Terminal 1

Terminal

Here, you see the local `master` branch that is followed by the `remotes/origin/HEAD` pointer (remember: `HEAD` is a pointer to a location in the repository), which is linked to `remotes/origin/master`.

Digging deeper

If you want to dig into the internals at this point, you can peek at the `.git` folder again:

```
ls .git/refs/
```

It has `heads`, which contains references to the local branch:

```
cat .git/refs/heads/master
```

And similarly for remote branches:

```
cat .git/refs/remotes/origin/master
```

Terminal 1

Terminal

So you've `git fetch`ed the remote branch and now you have a copy of it locally.

Applying remote changes to local

To apply the remote `master`'s changes to the local one, merge it just as you would for any other reference:

```
git merge origin/master
git log
```

The key point to take from all this is that pulling involves two actions: fetching (which just brings over the changes from a remote repository and caches it locally for reference) and merging (which actually messes with your local repository).

Personal preference

In general, I prefer fetching and merging rather than using `git pull`. If you do `git fetch` and `git merge` separately, keep reminding yourself of what's going on with respect to *remotes* in your Git repository. Once you've internalized that workflow, start using `git pull` as a convenience. If you use `git pull` too early, there is a danger of seeing it as magical or at least, not feeling entirely sure about what's going on.

Back
Two Git Repos: 'git_origin' and 'git_clo...

Next
Introduction: Working With Multiple R...

✓ Mark as Completed