Pushing to Repositories With Different Content

Learn to push to repositories with different content.

We'll cover the following

Same branch name but different content?

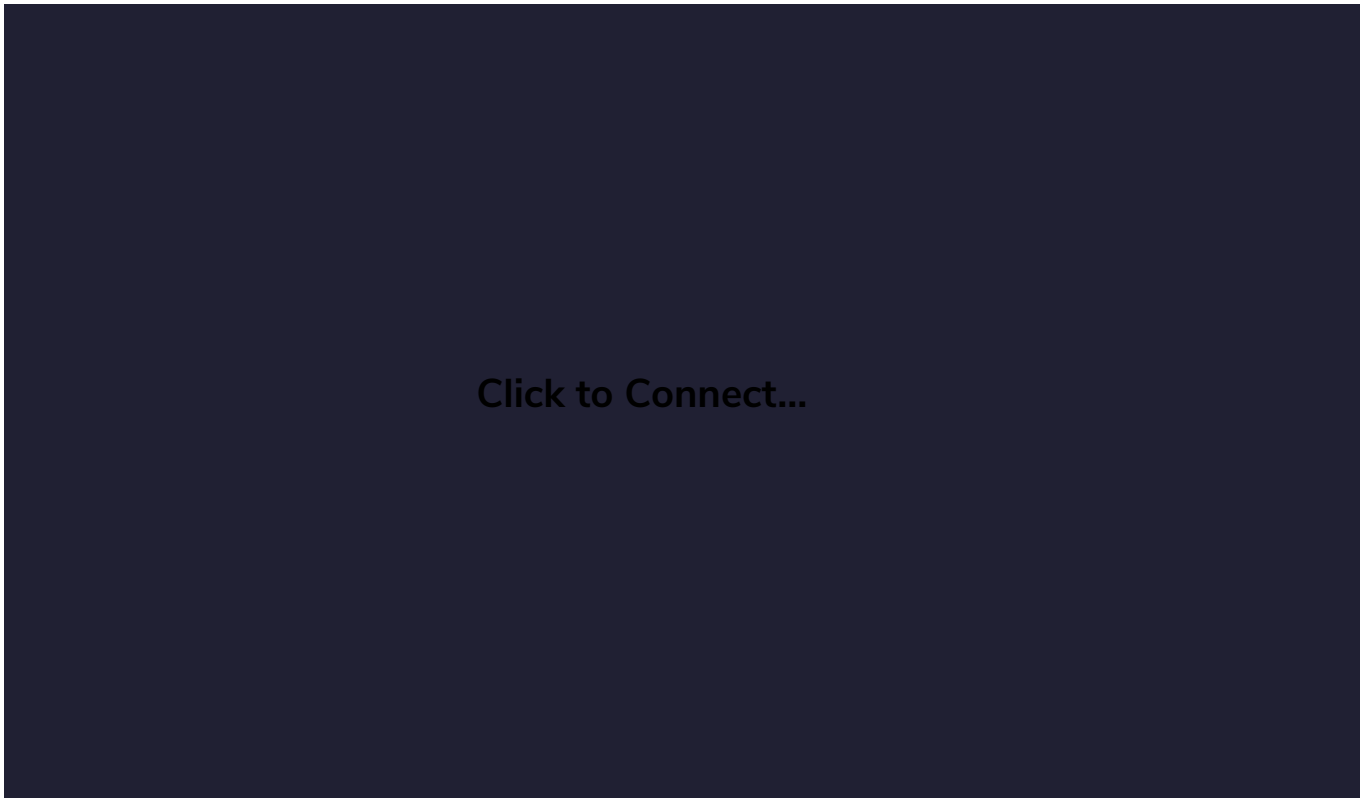
You might be asking yourself at this point: what happens if both repositories have a branch with the same name but *different* content?

Let’s see! Type this out.

```
mkdir git_origin
cd git_origin
git init
echo 'first commit' > file1
git add file1
git commit -am file1
cd ..
git clone git_origin git_clone
cd git_clone
git checkout -b abranch
echo 'cloned abranch commit' >> file1
git commit -am 'cloned abranch commit'
cd ../git_origin
git checkout -b abranch
echo 'origin abranch commit' >> file1
git commit -am 'origin abranch commit'
cd ../git_clone
git push origin abranch:abranch
```

Terminal 1

Terminal



Expected output

The output of the last command will look something like this:

```
To /lgthw_pushing/git_origin
 ! [rejected]        abranch -> abranch (fetch first)
error: failed to push some refs to '/lgthw_pushing/git_origin'
hint: Updates were rejected because the remote contains work that you
do
hint: not have locally. This is usually caused by another repository p
ushing
hint: to the same ref. You may want to first integrate the remote chan
ges
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for deta
ils.
```

Read the output carefully. It tells you exactly what's going on.

Breaking it down

```
Updates were rejected because the remote contains work that you do no
t have locally.
```

The remote (the `origin`) has a commit (with the content: `origin abranch commit`) that you have *no record of locally* in your branch with the same name.

It goes on:

```
This is usually caused by another repository pushing to the same ref.
```

It has correctly diagnosed the problem as another repository (`git_remote`), pushing to the same branch name (ref) on the receiving remote. Finally, it offers some advice.

```
You may want to first integrate the remote changes (e.g., 'git pul
l ...') before pushing again.
```

Performing a fetch and merge

But by now, you should know better than to `git pull` without thinking! Do a fetch and merge to really understand what is going on:

```
19  git fetch origin
```

Terminal 1

Terminal

Check the branches you have locally:

```
20  git branch -v -a
```

Observe that the `remotes/origin/abranch` branch you now have locally is *different* from the *local* `abranch` branch.

To complete your manual pull, merge the remote branch into the local:

```
21  git merge remotes/origin/abranch
```

Follow the instructions to resolve the conflict and commit the result.

Later, we will cover doing a rebase here rather than a merge for a cleaner history!

> **Note:** You may be wondering why you don't push at this point. A push, in this specific case, will not work because you are pushing a conflicting change to a repository that is not a bare Git repository. This is explained further in a later section on bare Git repositories. If you clone from a Git server, then you will be able to push back in this case as normal, as those servers have bare repositories.

Back

Next

Creating and Pushing Branches

The Branch Exists Only On The Remote

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/pushing-to-repositories-with-different-content__pushing-code__learn-git-the-hard-way)