codecademy

Git Teamwork

Git Remote

A *remote* is a shared Git repository that allows multiple collaborators to work on the same Git project from different locations. Collaborators work on the project independently and merge changes together when they are ready to do so.

Cloning a Remote Repository

In Git, the `git clone remote_location clone_name` command creates a local copy of a remote repository.

- `remote_location` tells Git where to find the remote repository and can be a filepath or web address.
- `clone_name` is the name of the directory where the remote repository's contents will be copied.

In the example, `my-quizzes` is a new directory created as a local copy of the `science-quizzes` Git project. Committing changes to `my-quizzes` will not impact `science-quizzes`.

```
$ ls
science-quizzes

$ git clone science-quizzes/ my-quizzes
Cloning into 'my-quizzes'...
done.

$ ls
my-quizzes  science-quizzes
```

List the Git Remotes

In Git, the `git-remote -v` command returns a list of remote repositories that the current project is tied to.

- Git lists the name of the remote repository as well as its locations.
- Git automatically names this remote `origin`, because it refers to the remote repository of origin. However, it is possible to safely change its name.
- The remote is listed twice: once for `(fetch)` and once for `(push)`.

```
$ git remote -v
origin  /home/ccuser/workspace/curriculum/
science-quizzes/ (fetch)
origin  /home/ccuser/workspace/curriculum/
science-quizzes/ (push)
```

Fetching Remote Origin Changes

In Git, the `git fetch` command downloads objects from the `origin` remote repository. The changes, however, are not merged into the current `branch-name` branch. Instead, they are stored in the `origin/branch-name` branch, waiting to be merged.

In the provided example, using the `git branch -a` command to see the existing branches, we can see that fetched data has been stored in a new `origin/master` branch.

```
$ git branch -a
* master

$ git fetch
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (5/5),
done.
remote: Total 5 (delta 1), reused 0 (delta
0)
Unpacking objects: 100% (5/5), done.
From /home/ccuser/workspace/curriculum-
a/science-quizzes
 * [new branch]      master     ->
origin/master

$ git branch -a
* master
  remotes/origin/master
```

Merging Fetched Changes

In Git, the `git merge origin/branch-name` command will merge fetched changes, stored in `origin/branch-name` to the current `branch-name` branch.

In the example, `master` is the name of the branch being merged.

```
$ git merge origin/master
Updating 2fd7d9b..3a29454
Fast-forward
 biology.txt | 4 ++++
 1 file changed, 4 insertions(+)
 create mode 100644 biology.txt
```

Git Collaboration Workflow

A common Git collaboration workflow is:

1. Fetch and merge changes from the remote
2. Create a branch to work on a new project feature
3. Develop the feature on a branch and commit the work
4. Fetch and merge from the remote again (in case new commits were made)
5. Push branch up to the remote for review

Steps 1 and 4 are a safeguard against merge conflicts, which occur when two branches contain file changes that cannot be merged with the git merge command.

Pushing Branch Changes

In Git, the `git push origin branch-name` command pushes the branch, and all committed changes, to the remote. This branch can now be reviewed by collaborators.
In the example, the current branch containing the committed changes is called `bio-questions` .

```
$ git push origin bio-questions
Counting objects: 3, done.
Delta compression using up to 16 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 392 bytes | 0
bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To /home/ccuser/workspace/curriculum-
a/science-quizzes
 * [new branch]      bio-questions -> bio-
questions
```