# Overview

So far, we've learned how to work on Git as a single user. Git offers a suite of collaboration tools to make working with others on a project easier.

Imagine that you're a science teacher, developing some quizzes with Sally, another teacher in the school. You are using Git to manage the project.

In order to collaborate, you and Sally need:

* A complete replica of the project on your own computers
* A way to keep track of and review each other's work
* Access to a definitive project version

You can accomplish all of this by using remotes. 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.

**git clone**

Sally has created the remote repository, **science-quizzes** in the directory **curriculum**, which teachers on the school's shared network have access to. In order to get your own replica of **science-quizzes**, you'll need to *clone* it with:

git clone remote\_location clone\_name

In this command:

* remote\_location tells Git where to go to find the remote. This could be a web address, or a filepath, such as:

/Users/teachers/Documents/some-remote

* clone\_name is the name you give to the directory in which Git will clone the repository.

# git remote -v

Nice work! We have a clone of Sally's remote on our computer. One thing that Git does behind the scenes when you clone **science-quizzes** is give the remote address the name origin, so that you can refer to it more conveniently. In this case, Sally's remote is origin.

You can see a list of a Git project's remotes with the command:

git remote -v

# git fetch

After you cloned **science-quizzes**, you had to run off to teach a class. Now that you're back at your computer, there's a problem: what if, while you were teaching, Sally changed the **science-quizzes** Git project in some way. If so, your clone will no longer be up-to-date.

An easy way to see if changes have been made to the remote and bring the changes down to your local copy is with:

git fetch

This command will not merge changes from the remote into your local repository. It brings those changes onto what's called a remote branch. Learn more about how this works below.

# git merge

Even though Sally's new commits have been fetched to your local copy of the Git project, those commits are on the origin/master branch. Your local master branch has not been updated yet, so you can't view or make changes to any of the work she has added.

In Lesson III, Git Branching we learned how to merge branches. Now we'll use the git mergecommand to integrate origin/master into your local master branch. The command:

git merge origin/master

will accomplish this for us.

**Git workflow**

Now that you've merged origin/master into your local master branch, you're ready to contribute some work of your own. The workflow for Git collaborations typically follows this order:

1. Fetch and merge changes from the remote
2. Create a branch to work on a new project feature
3. Develop the feature on your branch and commit your work
4. Fetch and merge from the remote again (in case new commits were made while you were working)
5. *Push* your 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. Step 5 involves git push, a command you will learn in the next exercise.

# git push

Now it's time to share our work with Sally.

The command:

git push origin your\_branch\_name

will push your branch up to the remote, origin. From there, Sally can review your branch and merge your work into the master branch, making it part of the definitive project version.

**generalizations**

Congratulations, you now know enough to start collaborating on Git projects! Let's review.

* A *remote* is a Git repository that lives *outside* your Git project folder. Remotes can live on the web, on a shared network or even in a separate folder on your local computer.
* The *Git Collaborative Workflow* are steps that enable smooth project development when multiple collaborators are working on the same Git project.

We also learned the following commands

* git clone: Creates a local copy of a remote.
* git remote -v: Lists a Git project's remotes.
* git fetch: Fetches work from the remote into the local copy.
* git merge origin/master: Merges origin/master into your local branch.
* git push origin <branch\_name>: Pushes a local branch to the origin remote.

Git projects are usually managed on Github, a website that hosts Git projects for millions of users. With Github you can access your projects from anywhere in the world by using the basic workflow you learned here.