# Group Repo Tips

You've probably figured out that working on a shared Git repo is better than not having any shared storage, but isn't the same as a shared Google drive or shared Dropbox folder. This sheet is designed to share some recommended tips to help you avoid creating *conflicts* within Git and also facilitate working together.

1. Be sure everyone in your group has successfully cloned their group repo.
2. When working from ANY repo, be sure you PULL before starting coding during any session you work on material.
3. Before ending any session you work on material, be sure you commit and PUSH before stopping work. We want to avoid conflicts where one person didn't update the files with a push and someone else started working on the previous file version.
4. Be sure you are working on files in the right place/repo. If you aren't sure, and want to chat with me about your file organization, just let me know. This is a valuable skill.
5. Be sure your commit messages are informative! This will help group members know what is going on and what you were working on if they look at the commit history. If every commit just says "update", that's not very helpful. Try things like "Added input to tab1" or "fixed issue with pivot in wrangling", etc.

Although Git is not like Google docs, so you can't work simultaneously on the same file, there are still approaches you can take to do some "shared" work, depending on your groups preferences.

1. You can use your readme in the repo to link out to a shared Google doc for notes, brainstorming, or even drafting files. That way, it's easily accessible to the group (and to me) if you need assistance. Just put a link to the Google doc in your readme.
2. You can use "issues" in Git to assign each other various tasks in different files, or to leave notes for each other about what time you'll be working, etc. We haven't explored a lot of the Git functionality - you could even have a wiki about your project. Issues can be useful especially if MemberA finishes work late one night and really needs help with Task1, they can make an issue so the whole group sees Task1 is a priority and MemberA needs help with it. You can then close issues when they are completed.
3. In the Shiny app lab, you saw how a single app had 3 tabs, and those 3 tabs could be split out into separate app files. You can do something similar as you work on the project (though eventually they will need merged). You don't have to do this, but it is a possible strategy. If you split anything up like this, it's very important to pay attention to what you call objects and variables you create. For example, if MemberA calls an input "pt_size", and MemberB does too, that's going to cause problems when you try to merge things together. I'd recommend denoting things based on tabs. For example, if MemberA has Tab2 (arbitrary numbers), call all those inputs something that includes tab2, such as tab2_pt_size. Or maybe you want to call them Vis1, Vis2, etc.

Part of the point of the project is that you help each other. While each person is taking the lead on one app component, you don't (and shouldn't) have to write that code exclusively individually. Help each other out. And in the wrangling steps, everyone should be participating in getting the data into a workable format(s) for the app.