**Geospatial Centroid GitHub Collaboration Workflow**

**Starting a new project:**

1. From the Geospatial Centroid GitHub account: create a new repository with appropriate naming convention (short/concise, no spaces).
2. Admin (the person who created the repo): Clone the repo to local machine
   1. Create a basic folder structure (i.e., data, scripts, outputs, etc.) with a basic README file describing the project and push to GitHub. This sets the folder/file structure for the rest of the project.
   2. For R projects, also create an R project with git version control (guide [here](https://happygitwithr.com/existing-github-first.html#make-a-repo-on-github-3)) and push to GitHub
3. Add collaborators with write/push access by going to repo Settings -> Manage Access -> Invite a Collaborator
4. All team members (Admin must also do this from their personal account):
   1. Accept the collaboration invitation
   2. Go to the repo and click “Fork”. This will create a copy of the repo on your personal GitHub account
   3. Clone your forked repo to local machine. You will now have a folder with repo name and basic folder/file structure set up by admin.

*Repo is created and all set up. Now time to start coding!*

**Project workflow:**

1. Work on local repo, edit, commit, push.
2. When you have some finalized edits you want to add to the Centroid repo (referred to as the **upstream branch**), submit a pull request. BUT FIRST:
3. If any changes have been made to the upstream branch, fetch those before submitting your pull request. You can do this two ways:
   1. On Github: navigate to the main repo on the Centroid page and click ‘Fetch Upstream’. Review the details about the commits made to the main repo, then click ‘fetch and merge’. If there are conflicts, GitHub will tell you to create a pull request to resolve conflicts
   2. From the command line: In git, make sure you are working in the directory of your local project and run the following:

Add 'upstream' repo to list of remotes with the appropriate user and project name (you should only need to do this once):

git remote add upstream https://github.com/UPSTREAMUSER/ORIGINAL-PROJECT.git

Verify the new remote named 'upstream' now exists

git remote -v

Then whenever you want to update your fork with the latest changes made by others to the central repo:

git fetch upstream

Then checkout your local main branch and merge it with the upstream changes

git checkout main

git merge upstream/main

Now your fork is in sync with the upstream Centroid repo (while keeping your new local changes)

Finally push all changes to your local repo and submit a pull request to have your changes incorporated into the Centroid (upstream) repo

**Creating a pull request from fork**[: https://docs.github.com/en/github/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request-from-a-fork](file:///C:\Users\ccmothes\Desktop\%20https\docs.github.com\en\github\collaborating-with-pull-requests\proposing-changes-to-your-work-with-pull-requests\creating-a-pull-request-from-a-fork)

* + - 1. Navigate to the upstream Centroid repo and click “Pull Request” in the right-hand corner
      2. On the compare changes click “compare across forks”
      3. In the ‘base branch’ drop down menu, select the Centroid repo as the base repository and “main” as the main branch
      4. In the “head fork” drop down menu select your fork (local repo and main branch)
      5. Add a title and description of your pull request
      6. Check “Allow edits from maintainers” to allow other collaborators to make changes to your pull request
      7. Click “Create Pull Request”

**Requesting a pull request review:** <https://docs.github.com/en/github/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/requesting-a-pull-request-review>

You can assign a specific person to review your pull request and they will get notified, but any other collaborator can also review your pull request

**Reviewing a pull request:** <https://docs.github.com/en/github/collaborating-with-pull-requests/reviewing-changes-in-pull-requests/reviewing-proposed-changes-in-a-pull-request>

**Linking Data Folder**

Due to size limitations, the keep the data folder on local machines/in the cloud only and do not push that folder to github (by adding data/ to the .gitignore file). For most projects, the data folder will be hosted on someone’s cloud (OneDrive) account. You can link that folder on OneDrive to your local repo programmatically, and therefore every time you add or change data locally it will also update the drive folder and anyone else’s folders that are also linked.

To link the OneDrive data folder to a local computer:

1. If the folder is shared with you, on your OneDrive account go to “Shared” on the left hand side, click on the folder, and click “Add shortcut to My files” in the upper banner
2. Open a command prompt on your computer and run the following (adjusting to your local/personal file paths):

mklink /j "[full path on your local comp]\data" "[full path to your One Drive folder]\data"

Note that you should do this **before** adding a ‘data/’ folder in your local directory, it may throw an error if you do. If you do have a folder called ‘data’ in the local repo, you’ll need to move/rename the current local ‘data/’ folder (can remove after synced data folder has everything), and then run the above code. That folder is now synced with the OneDrive data folder.

*If working in feature branches:*

Make sure forked main is updated with upstream changes (as above) then start a new branch:

git checkout -b <new branch name>

Branch starts with a copy of the branch you were on (normally the main branch)

Make edits in this branch and add/commit as usual

When complete, push feature branch to remote (your forked repo):

git push -u origin <branch name>

From here, you can keep using git push to push changes to remote, but these stay in your branch and are not merged with the main branch yet.

If you want to merge an existing branch with upstream changes made since you started it, update your main branch as described above, then:

git checkout <branch you want to update>

git merge <branch you’re merging from>

To merge branch with central repo: Issue a pull request (as above)

**Best Practices:**

* **Keep your fork up to date regularly**
* COMMUNICATE. Each team member works on separate files/features (no overlap, feature collaboration can take place in the pull requests).
* Pull request comments are written in Markdown, so take advantage of embedding images, emojis, text blocks, etc. You can also associate issues with code. For example, entering the phrase “Closes #32” in a pull request would close issue number 32 in the repo.
* Standardize file/variable/function style and naming conventions at the beginning of the project. Suggest following tidyverse style guide for all R stuff: <https://style.tidyverse.org/index.html> (some of this can apply to other languages too)

**Reference Guides:**

GitHub Docs: <https://docs.github.com/en>

Understanding the GitHub flow: <https://guides.github.com/introduction/flow/>

Comparing Workflows: <https://www.atlassian.com/git/tutorials/comparing-workflows>

Using Git and GitHub for Team Collaboration: <https://medium.com/anne-kerrs-blog/using-git-and-github-for-team-collaboration-e761e7c00281>

Starting a new group project on GitHub: <https://www.digitalcrafts.com/blog/learn-how-start-new-group-project-github>