Git and GitHub Workflows for Solo Development

Getting Started with Git and GitHub Part 2
Coffee, Cookies, and Coding (C-cubed) Workshops

January 26, 2026

This workshop provides three examples demonstrating common uses of a configured Git and GitHub setup. The first two, given here, show how to establish a connection starting with either a local project directory or by cloning an existing remote repository. In a separate example, students can find a practice collaboration exercise to do with peers, where they will propose code adjustments for collaborators to review via pull requests.

To effectively follow these guidelines, we suggest that you use the following file structure. By the end of this module, your workshop directory should contain:

```
dsde-workshops/
  |- start-local/
  |- JHU-CRC-Vaccinations/
  |- JHU-CRC-Cases-and-Deaths/
```

! Important

Specific directions are provided for establishing these repositories. To interact with the pre-prepared JHU-CRC examples, you must first create a "clean-break" copy in your own GitHub account. When practicing collaboration, only one team member needs to complete this step; all others can then clone that member's copy to their local device.

Find the directions for creating a "clean-break" on the Book of Workshops webpage for this workshop: Getting Started with Git and GitHub - Accessing the Codespaces

Start Locally

In this example, we will create a mock project folder from scratch, initialize Git, and link it to a remote repository on GitHub. We will then test the connection by adding "Hello World" to the `README.md` file on GitHub and pulling the changes to the local repository.

Create an Empty Remote Repository

1. Log in to your personal GitHub account.
2. In the top-right of the page navigation bar, select the **+ ▾** dropdown menu and click **New repository**[1].
3. Fill out the following sections:
   a. Adjust the GitHub account owner as needed and create the name for the new repository.
   b. It is good practice to initially set the repository to "Private".
   c. Do **NOT** use a template or include a description, `README.md`, `.gitignore`, or license.

Create a New Local Project

1. Open the command-line application (i.e. Terminal for Macs and Git Bash for Windows) and navigate to the file location you want to temporarily store the repository copy.

Command-Line Interface

```
cd "~/dsde-workshops/"       # Go to workshop directory
mkdir "start-local"          # Create project folder
cd "start-local"             # Enter project folder
```

2. Confirm that Git has **NOT** yet been initialized by checking the project status. If the project is initialized, you should see that Git is present and tracking files.

   Command-Line Interface

   ```
   git status
   ```

   Output

   ```
   fatal: not a git repository (or any of the parent directories): .git
   ```

3. Initialize Git in the project directory for the first time and create the primary branch `main`.

   Command-Line Interface

   ```
   git init -b main
   ```

   Output

   ```
   Initialized empty Git repository in ~/dsde-workshops/start-local/.git/
   ```

4. Running `git status` again should no longer produce an error. Instead, it should indicate that we are on a branch called `main` with no commits yet. Before linking our project with the remote repository, we need to add a file that Git can track. Let's create an empty `README.md` file.

   Command-Line Interface

```
touch "README.md"
```

5. Running `git status` should now show an untracked file in our project directory.

Command-Line Interface

```
git status
```

Output

```
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    README.md

nothing added to commit but untracked files present (use "git add" to track)
```

6. Use `git add` to promote changes from the **Working Tree** to **Staged Edits**. You can specify individual files or use the wildcard `git add .` to add all untracked and changed files at once.

Command-Line Interface

```
git add "README.md"          # Option #1
git add .                    # Option #2
```

7. Running `git status` should show the untracked file is ready to be promoted to the **Committed Edits** domain. Keep in mind that only versions of the project that have been "committed" will be shared with the remote repository.

Command-Line Interface

```
git status
```

Output

```
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   README.md
```

8. Use `git commit` to promote changes from **Staged Edits** to **Committed Edits** and include a message describing your changes. If prompted, enter your password (for example, if you have commit signing with an SSH key configured).

   ⚠ Commit Message Required

   Every commit requires a message. If you forget to include one as shown below, you will be prompted to write your message automatically. For most people, this will open an in-application text editor, such as vim, as was shown in the walk-through.

   If this sounds unfamiliar, or you do not remember how to navigate the in-application text editor, refer to the Detailed Walk-Through webpage on the Book of Workshops.

Command-Line Interface

```
git commit -m "Init repo"
```

Output

```
[main (root-commit) b669479] Init repo
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
```

Establish the Connection

1. To link our local, version-controlled project directory with our empty GitHub repository, we need to define the repository location using either its HTTPS or SSH URL. Which one you choose depends on your preference and configurations (see Configurations and Credentials - Transfer Protocols for additional details).

   In the newly created GitHub repository, under "Quick setup," you will find the repository's SSH or HTTPS URL. Copy one of these URLs to define the remote location and transfer protocol you want to use between your local device and this GitHub repository.

   For example, if the repository name is "NEW-REPOSITORY," the URLs will look like this:

   ```
   # SSH
   git@github.com:EXAMPLE-USER/NEW-REPOSITORY.git

   # HTTPS
   https://github.com/EXAMPLE-USER/NEW-REPOSITORY.git
   ```

2. Designate the remote location and transfer protocol (SSH/HTTPS), and associate the URL with the `origin` repository alias.

   Command-Line Interface

   ```
   # SSH
   git remote add origin git@github.com:EXAMPLE-USER/NEW-REPOSITORY.git

   # HTTPS
   git remote add origin https://github.com/EXAMPLE-USER/NEW-REPOSITORY.git
   ```

3. Push the project contents to the remote repository. Since this is the first push from the `main` branch, we need to create a corresponding remote branch to complete the transfer. This step is only necessary when the branch doesn't already exist remotely.

Command-Line Interface

```
git push -u origin main
```

Output

```
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 438 bytes | 438.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:sgolde13/start-local.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

4. Refresh the GitHub page to see the changes reflected.

5. You should now see a new block for the `README.md` contents on the repository GitHub page. Click to edit (top-right pencil icon), and type in "Hello World".

6. Commit the changes (you can modify the default commit message if you like), then return to the repository landing page to confirm the changes were saved.

7. Now let's update our local copy. In the command-line application, enter `git pull` to download the changes.

Command-Line Interface

```
git pull
```

Output

```
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 904 bytes | 226.00 KiB/s, done.
From github.com:sgolde13/start-local
   b669479..2ccf99e  main       -> origin/main
Updating b669479..2ccf99e
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

Start Remotely

In this example, we are going to copy one of the pre-prepared codebases from the remote repository "ysph-dsde/JHU-CRC-Vaccinations" on GitHub to your local device. We will then test this connection by creating a small change to the plot the code generates and publishing those changes to the repository.

Making a Clean-Break Copy

1. Open the Getting Started with Git and GitHub - Accessing the Codespaces webpage. From there, open the "ysph-dsde/JHU-CRC-Vaccinations" GitHub repository in a new tab and scroll down to the *Making a Clean-Break Copy* section.
2. Follow **Method #1** or **Method #2** to create a clean-break copy of the remote repository in your personal GitHub account. We recommend starting with **Method #1**, but if that takes too long or does not work, **Method #2** is available as a backup option.

Clone the Project Codebase

1. Open the command-line application (i.e. Terminal for Macs and Git Bash for Windows) and navigate to the file location you want to temporarily store the repository copy.

Command-Line Interface

```
cd "~/dsde-workshops/"      # Go to workshop directory
```

2. In the copied GitHub repository, "YOUR-USER-ID/JHU-CRC-Vaccinations", find the repository's SSH or HTTPS URL. Copy the SSH or HTTPS URL from your GitHub repository by clicking the <> Code ▾ button. Which one you choose depends on your preference and configurations (see Configurations and Credentials - Transfer Protocols for additional details).

For example the URLs will look like this:

```
# SSH
git@github.com:YOUR-USER-ID/JHU-CRC-Vaccinations.git

# HTTPS
https://github.com/YOUR-USER-ID/JHU-CRC-Vaccinations.git
```

3. Clone the repository to your local device.

Command-Line Interface

```
# SSH
git clone git@github.com:YOUR-USER-ID/JHU-CRC-Vaccinations.git

# HTTPS
git clone https://github.com/YOUR-USER-ID/JHU-CRC-Vaccinations.git
```

Test the Connection

1. Open the cloned project directory.

Command-Line Interface

```
cd "JHU-CRC-Vaccinations"
```

2. Open the R project environment.

Command-Line Interface

```
open JHU-CRC-Vaccinations.Rproj
```

3. In RStudio, open `Plot Vaccinations.R` and initialize the environment.

RStudio

```
renv::init()      # Set up renv environment
renv::restore()   # Install locked package versions
```

4. Run the code to generate a JPEG of the bar graph showing the percentage of the U.S. population with at least one COVID-19 vaccination dose by March 2023. Modify the plot code (e.g., change colors, labels, or title) as you would like before exporting the image.

5. Commit and share these changes by pushing the new file and any code modifications to the remote repository on GitHub.

Command-Line Interface

```
git add .                          # Stage all changes
git commit -m "New plot"           # Commit staged changes
git push                           # Push to remote repository
```

6. Refresh the GitHub page to see the changes reflected.

References

1. GitHub. Creating a new repository. *GitHub Docs*.