

Example Git/GitHub Workflow

Jeffrey Leek, Assistant Professor of Biostatistics



May 17, 2016

An example workflow

- ▶ Feel free to create a test repo of your own (on your GitHub account) and follow along
- ▶ This is not meant to be an exhaustive introduction to all Git and GitHub features
- ▶ Best way to learn Git and GitHub are by using them
- ▶ If you get stuck, use the resources at the end of the last lecture

Create a new repo on GitHub


Owner **Repository name**


PUBLIC   ncarchedi / test_repo ✓

Great repository names are short and memorable. Need inspiration? How about [laughing-octo-dubstep](#).

Description (optional)

This is a test repo.

☒  **Public**
Anyone can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** | Add a license: **None** ⓘ

Create repository

- ▶ Leave the box next to “Initialize this repository with a README” **unchecked**, since we will add one later

Copy repo URL to clipboard

- ▶ After clicking “create repository”, you will get a screen with some basic instructions for getting started
- ▶ They are a good starting point, but we won't follow them exactly here
- ▶ Copy the URL displayed below to your clipboard (yours will be slightly different since it depends on your username and repo name)

Quick setup — if you've done this kind of thing before



Set up In Desktop

or

HTTP

SSH

`https://github.com/ncarchedi/test_repo.git`



copy to clipboard

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

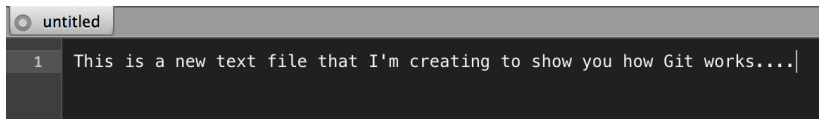
Create a new local directory

- ▶ Open Git Bash or Terminal (Windows or Mac, respectively) and create a new directory on your computer
- ▶ “Step inside” of this new directory with the `cd` command
- ▶ `mkdir` stands for “make directory”, `cd` stands for “change directory”, and `ls` stands for “list”
- ▶ Since we get no output after typing `ls`, we can see that the directory we created is still empty

```
~ $ mkdir ~/test_repo  
~ $ cd ~/test_repo/  
~/test_repo $ ls  
~/test_repo $
```

Create a new file in directory

- ▶ Open your favorite text editor and create a new text file
- ▶ Make sure to save it in the directory you just created



The screenshot shows a text editor window with a tab labeled 'untitled'. The editor has a dark background and a light-colored line number column on the left. Line 1 contains the text: 'This is a new text file that I'm creating to show you how Git works....|'. The cursor is at the end of the line. To the right of the editor, a portion of another window is visible, showing a 'Save' button and a file icon.

```
1 This is a new text file that I'm creating to show you how Git works....|
```

Create a new repo locally

- ▶ We already created a **GitHub** repository, but we still need to create a **Git** repository locally on your computer
- ▶ We can see that our new file is now in our chosen directory with `ls`
- ▶ `git init` initializes a Git repo in our current directory
- ▶ `git status` is a helpful command that we'll make frequent use of
- ▶ Does exactly what it sounds like – gives us a “status report” for our local repo

```
~/test_repo $ ls
test_file.txt
~/test_repo $ git init
Initialized empty Git repository in /Users/nicholasacarchedi/test_repo/.git/
~/test_repo $ git status
On branch master

Initial commit

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

Stage file for commit

- ▶ Notice that our newly created file falls under “untracked files” when we look at `git status`
- ▶ Use `git add` to tell Git that we want it to start “paying attention” to this file
- ▶ Could have used `git add test_file.txt` for the same result, but `git add .` is often easier if you are okay tracking all currently untracked files

```
~/test_repo $ git add .  
~/test_repo $ git status  
On branch master
```

Initial commit

Changes to be committed:
(use “`git rm --cached <file>...`” to unstage)

Commit changes

- ▶ Using `git commit` with a `-m` after it tells Git that whatever follows in double quotes is the message that we want to attach to this round of changes
- ▶ Another call to `git status` confirms that there is nothing new to commit (since our first commit)

```
~/test_repo $ git commit -m "first commit, which includes a test text file"
[master (root-commit) d7126e2] first commit, which includes a test text file
1 file changed, 1 insertion(+)
create mode 100644 test_file.txt
~/test_repo $ git status
On branch master
nothing to commit, working directory clean
~/test_repo $
```

Check log

- ▶ `git log` shows us a history of all commits
- ▶ So far, there's only one

```
~/test_repo $ git log
commit d7126e2ab3ec1d6df4cac1c40d0f9cc239888017
Author: Nick Carchedi <nick.carchedi@gmail.com>
Date:   Mon Jan 20 16:58:21 2014 -0500

    first commit, which includes a test text file
~/test_repo $
```

Add link to remote repo

- ▶ We now have a remote repo on GitHub's servers and a local repo on our computer, but they still don't know about each other
- ▶ To establish a link between the two, we paste the URL copied earlier from GitHub as follows

```
~/test_repo $ git remote add origin https://github.com/ncarchedi/test_repo.git
~/test_repo $ git remote -v
origin https://github.com/ncarchedi/test_repo.git (fetch)
origin https://github.com/ncarchedi/test_repo.git (push)
~/test_repo $
```

- ▶ `git remote -v` shows us that our GitHub repo is now set up as a “remote” repository for our local repo, which allows the two repos to communicate

Push changes to GitHub

- ▶ We want our GitHub repo to reflect the changes we've made locally (i.e. to contain our new text file)
- ▶ `git push -u origin master` tells Git to push our changes to the “master” (or main) branch of the “origin” (or primary) remote
- ▶ You only need to include the `-u origin master` once, as Git will remember this configuration for future pushes
- ▶ `git push` then becomes sufficient, assuming you don't want to do anything fancy

```
~/test_repo $ git push -u origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 306 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/ncarchedi/test_repo.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
~/test_repo $
```

Check status

- ▶ Check status again for piece of mind
- ▶ Notice that it confirms *Your branch is up-to-date with 'origin/master'*

```
~/test_repo $ git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean
~/test_repo $
```

Check GitHub

- ▶ We want to make sure that our changes made it to GitHub safely and indeed they did
- ▶ Our text file shows up in the file list

This is a test repo. — Edit


 1 commit

 1 branch

 0 releases

 1 contributor




 branch: master ▾

test_repo / 

first commit, which includes a test text file



ncarchedi authored 4 minutes ago

latest commit **d7126e2ab3** 



test_file.txt

first commit, which includes a test text file

4 minutes ago

We recommend [adding a README](#) to this repository to help give people an overview of your project.



Add a README

Add README file from GitHub

- ▶ How can we can “pull” changes from a remote repository to our local repository?
- ▶ To illustrate, we'll add and edit a README file directly from the GitHub website
- ▶ A more common scenario would be that a collaborator makes changes to a shared repository and you want to incorporate those changes into your local repo
- ▶ Click on the “Add a README” button on your GitHub repo page, under the list of files

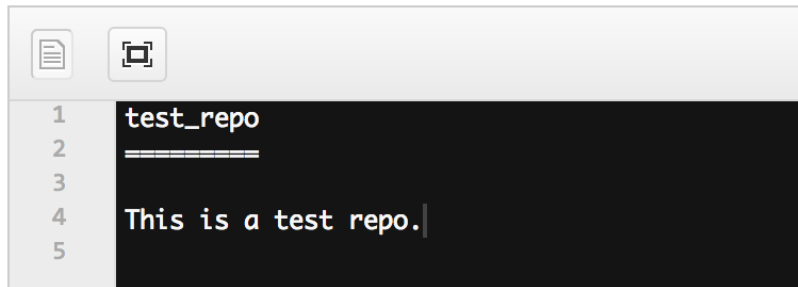
We recommend [adding a README](#) to this repository to help give people an overview of your project.

 **Add a README**

Edit README file from GitHub

- ▶ Put anything you want in the README, then press “Commit New File” to commit the file to your GitHub repo
- ▶ **Note:** README files written in Markdown will render in HTML on your repository’s homepage.

test_repo / README.md or **cancel**



The screenshot shows the GitHub README editor interface. At the top, there's a header bar with a document icon and a preview icon. Below this, the editor area has a dark background with white text. The text content is as follows:

```
1 test_repo
2 =====
3
4 This is a test repo.
5
```

On the right side of the editor, there's a partial view of a button with a red 'C' icon, likely for 'Commit'.

Fetch changes from GitHub

- ▶ Now that README.md exists on GitHub, we want to “pull” it down to our local repo
- ▶ There is a `git pull` command that allows you to do this, but it's recommended that you instead use a combination of `git fetch` and `git merge`
- ▶ `git fetch origin` tells Git to fetch all changes from the “origin” (primary) remote repo, which we set up earlier with `git remote add origin ...`

```
~/test_repo $ git fetch origin
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From https://github.com/ncarchedi/test_repo
   d7126e2..d0e46d4  master    -> origin/master
~/test_repo $
```

Merge changes into local repo

- ▶ Git is now aware of all changes that have been made to the remote (i.e. GitHub) repo
- ▶ Still need to incorporate these changes into our local repo
- ▶ `git branch -a` shows us that we now have two “branches” stored on our computer: `master` and `remotes/origin/master`
- ▶ `master` represents the files on our local repo and `remotes/origin/master` represents the files we pulled from our remote repo
- ▶ Use `git merge origin/master` to incorporate the changes from our remote repo

```
~/test_repo $ git branch -a
```

```
* master
```

```
remotes/origin/master
```

```
~/test_repo $ git merge origin/master
```

```
Updating d7126e2..d0e46d4
```

Check status

- ▶ A quick call to `ls` confirms that the `README.md` file is now in our local directory
- ▶ `git status` tells us that we have no new changes and are up-to-date with our remote repo

```
~/test_repo $ ls
README.md      test_file.txt
~/test_repo $ git status
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working directory clean
~/test_repo $
```