Introduction to Git and GitHub

GSND 5345Q, FDS

W. Evan Johnson, Ph.D.
Professor, Division of Infectious Disease
Director, Center for Data Science
Associate Director, Center for Biomedical Informatics and Health AI
Rutgers University – New Jersey Medical School

2026-01-26

Section 1

Introduction to Git and GitHub

Git and GitHub Introduction

Here we will provide some details on Git and GitHub, but we are only scratching the surface. Here are some resources to help you further:

- Codeacademy: https://www.codecademy.com/learn/learn-git
- GitHub Guides: https://guides.github.com/activities/hello-world/
- Try Git tutorial: https://try.github.io/levels/1/challenges/1
- Happy Git and GitHub for the useR: http://happygitwithr.com/

Git and GitHub Introduction

There are three main reasons to use Git and GitHub.

- Sharing: GitHub allows us to easily share code!
- Collaborating: Multiple people make changes to code and keep versions synched. GitHub has a special utility, called a **pull request**, that can be used by anybody to suggest changes to your code.
- Version control: Using `git` permits us to keep track of changes, revert back to previous versions, and create **branches** in to test out ideas, then decide if we **merge** with the original.

Section 2

GitHub Repositories

GitHub accounts

After installing git, go get a GitHub account. Go to https://github.com/. You will see a link to sign up in the top right corner.

Pick a name carefully! Something short, related to your name, and professional. Remember that you will use this to share code with others. You might be sharing this with potential collaborators or future employers!

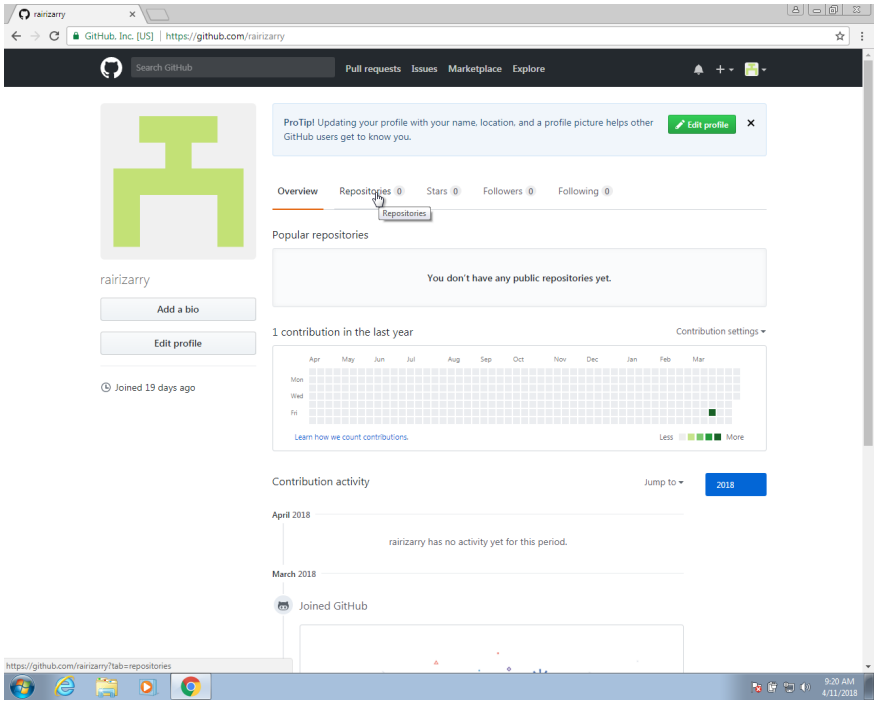
GitHub repositories

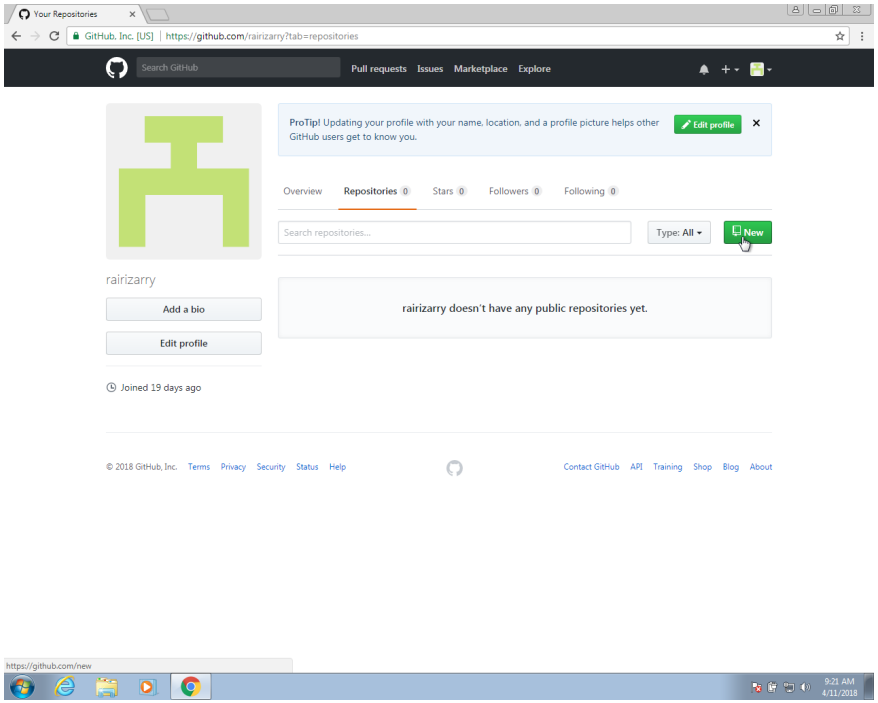
A GitHub repository allows you to have two copies of your code: one on your computer and one on GitHub. If you add collaborators, then each will have a copy on their computer.

The GitHub copy is usually considered the **main** copy (previously called the **master**). Git will help you keep all the different copies synced.

GitHub repositories

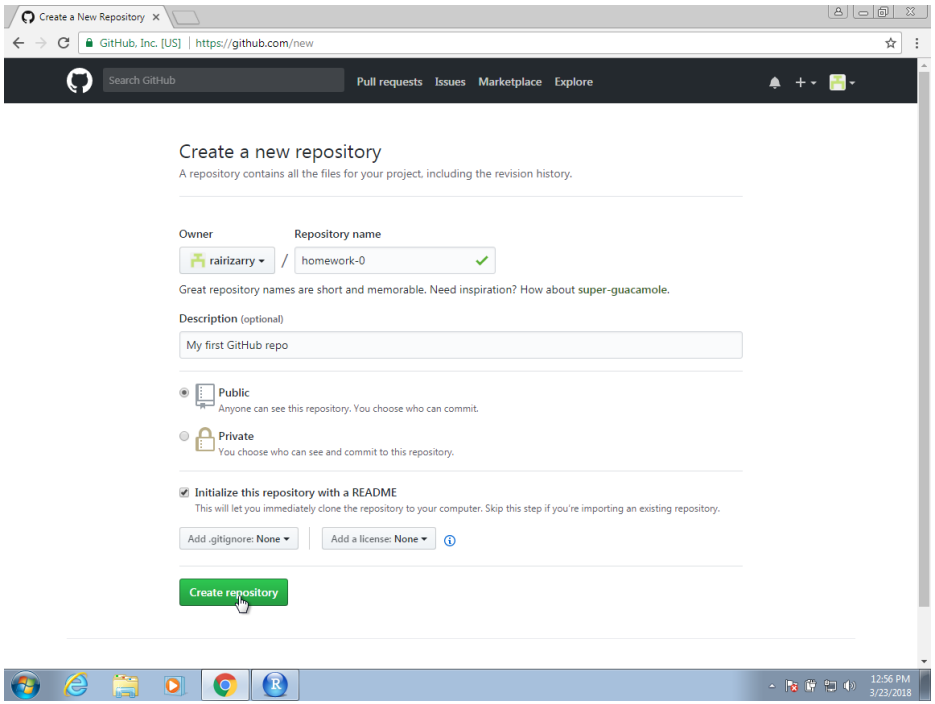
The first step is to initialize the repository on GitHub. You will have a page on GitHub with the URL: http://github.com/username. On your account, you can click on `Repositories` and then click on `New` to create a new repo:

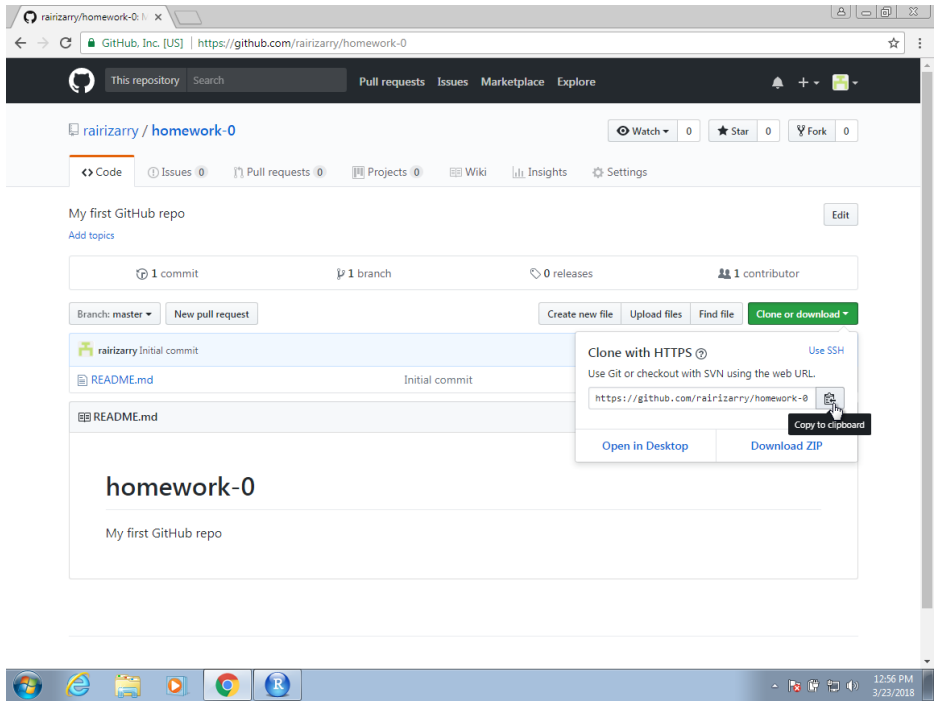




GitHub repositories

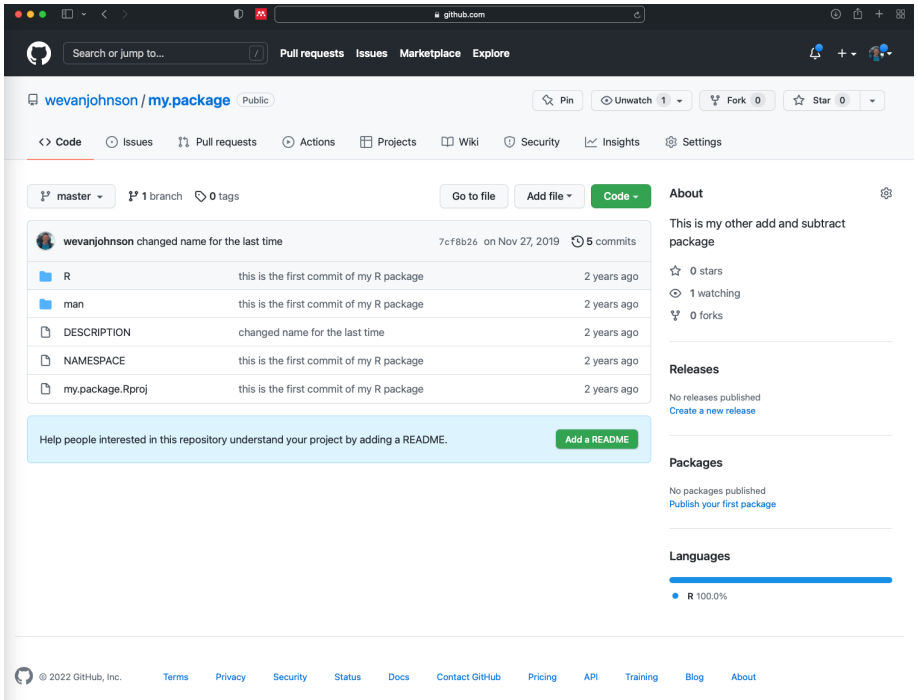
Choose a good descriptive name, for this example make a repo named "FDSLectures". The next step will be to **clone** it on your computer using the terminal. Copy the link to connect to this repo for the next step.

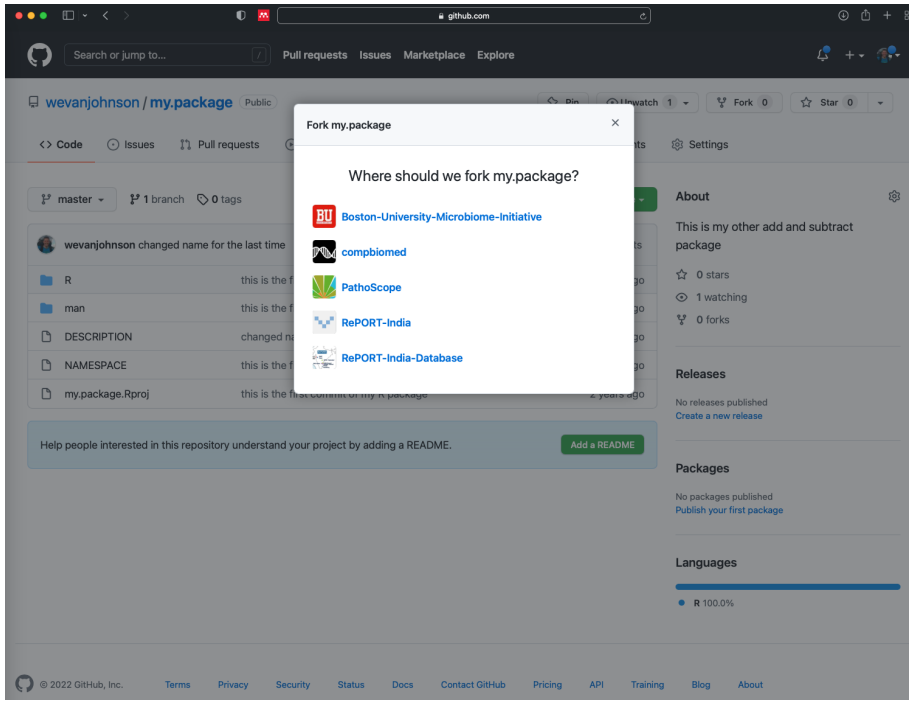




GitHub repositories

GitHub also allows you to **fork** others' repos. Go to https://github.com/wevanjohnson/my.package





Click `fork` in the top right and this will create a fork of the repository in your GitHub account.

Section 3

Git Basics

Git Setup

First let `git` know who you are–will make it easier to connect with GitHub. In a terminal window use the `git config` command:

```
git config --global user.name "My Name"
git config --global user.mail "my@email.com"
```

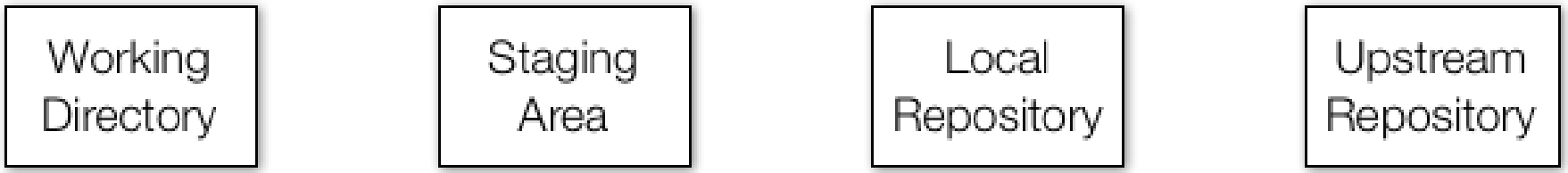
Git Setup

The main actions in git are to:

1. **pull** changes from the remote GitHub repo
2. **add** files, or as we say in the `git` lingo: **stage** files
3. **commit** changes to the local repo
4. **push** changes to the **remote** GitHub repo

Git Setup

To effectively permit version control and collaboration in `git`, files move across four different areas:

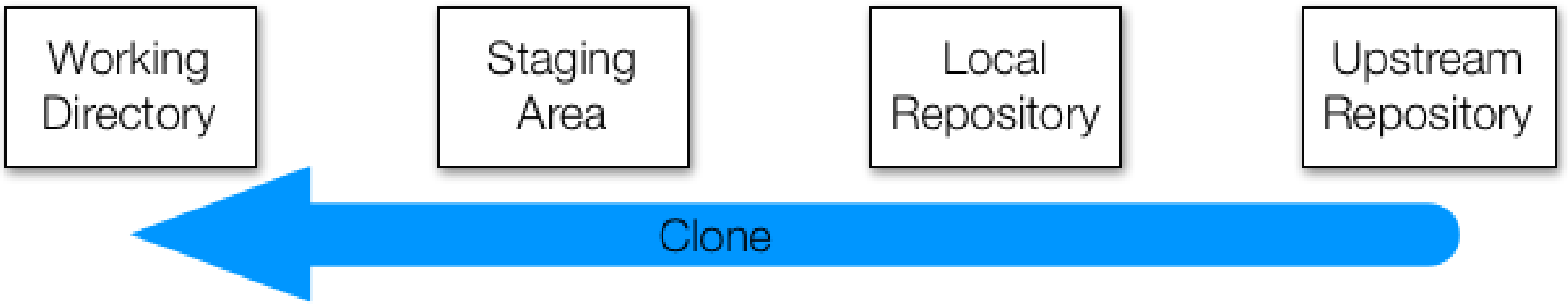


But how does it all get started? We can clone an existing repo or initialize one. We will explore cloning first.

Cloning Git repositories

We will **clone** your existing my.package **upstream repository** to your local computer.

What does clone mean? We are going to actually copy the entire `git` structure, files and directories to all stages: working directory, staging area, and local repository.



Cloning Git repositories

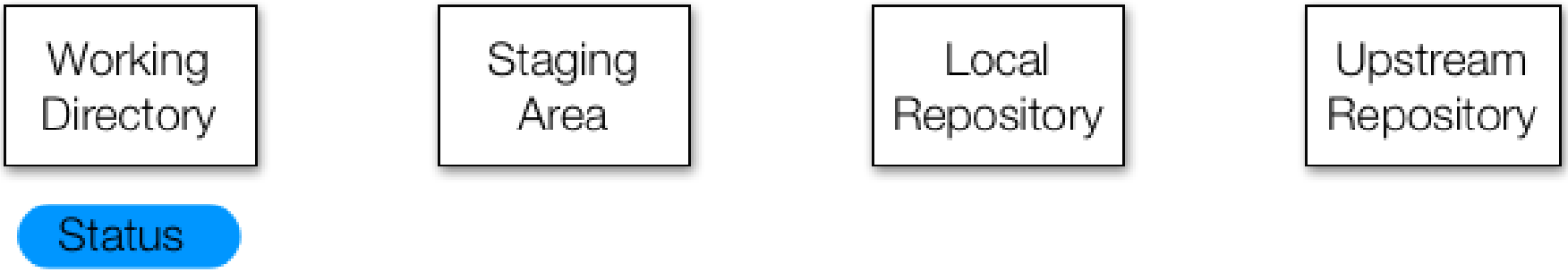
Open a terminal and type:

```
pwd
mkdir git-example
cd git-example
git clone https://github.com/yourusername/my.package.git
cd my.package
ls
```

Cloning Git repositories

Note: the **working directory** is the same as your Unix working directory. When you edit files (e.g., RStudio), you change the files in this directory. `git` can tell you how these files relate to the upstream directory:

```
git status
```



Working with Git repositories

Now lets add some changes to the local repository, open the `DESCRIPTION` file in the my.package directory, and add your name as an author of the package.

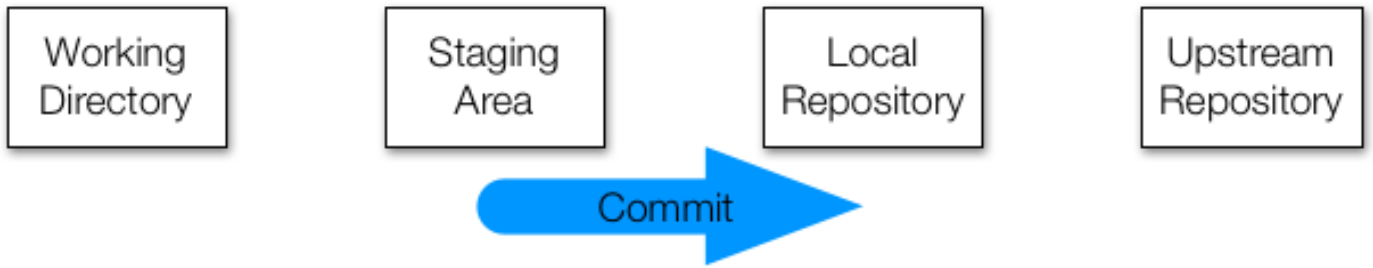
And, as we will do this soon, change the description in the file to include `multiplication`.

Working with Git repositories

Now lets **add** the changes to the staging area and **commit** the changes to the local Git directory:

```
git add DESCRIPTION
```

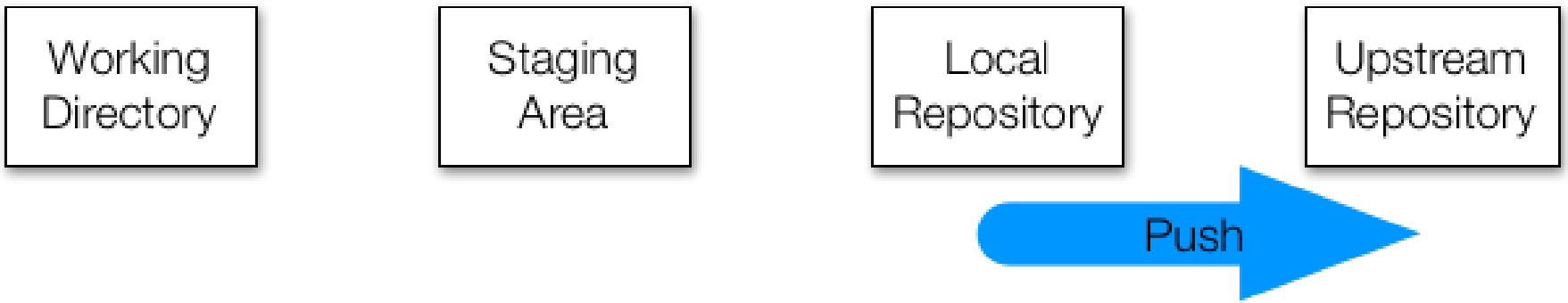

```
git commit -m "adding my name as an author"
```



Working with Git repositories

Now we can **push** the changes to the remote repo:

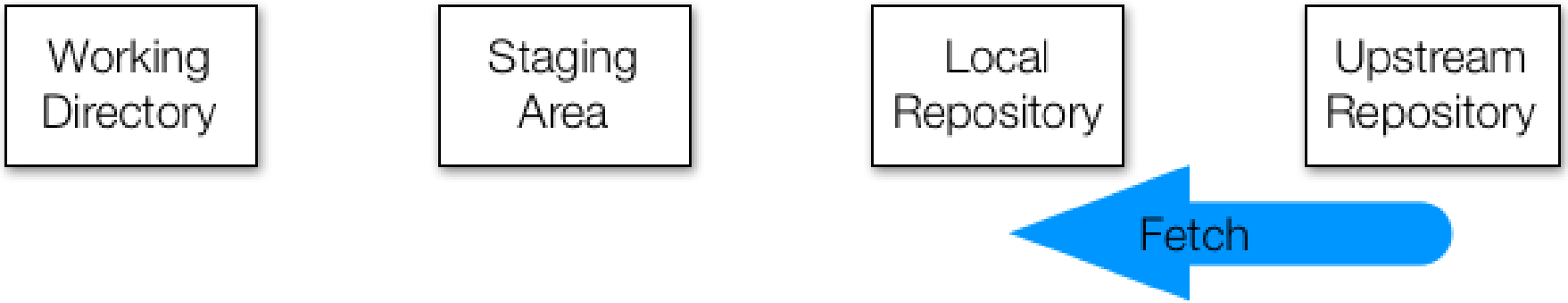
```
git push
```



Working with Git repositories

We can also **fetch** any changes on the remote repo (how do you think this is different from `clone`?):

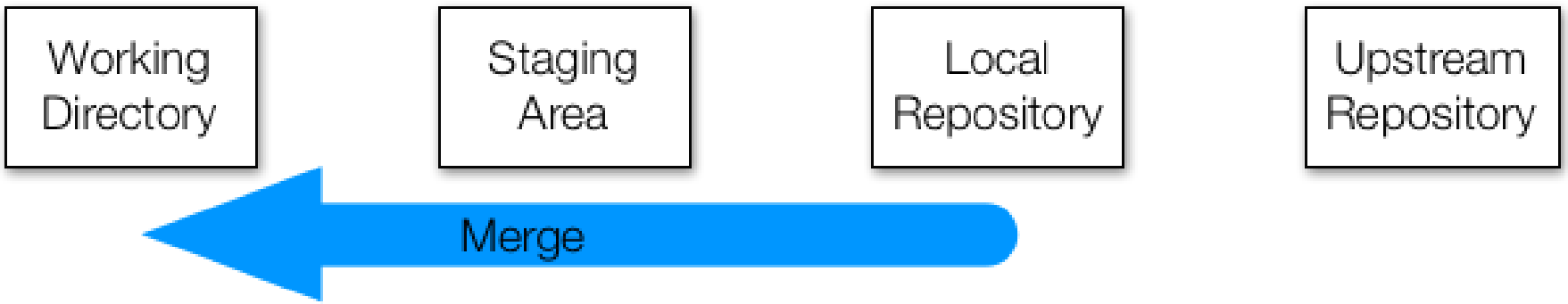
```
git fetch
```



Working with Git repositories

And then we need to **merge** these changes to our staging and working areas:

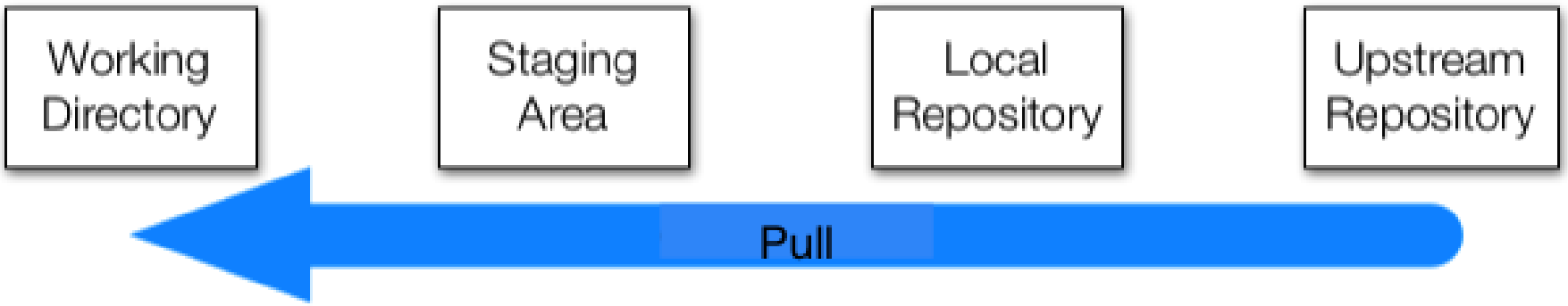
```
git merge
```



Working with Git repositories

Often we want to change both with one command. For this, we use:

```
git pull
```



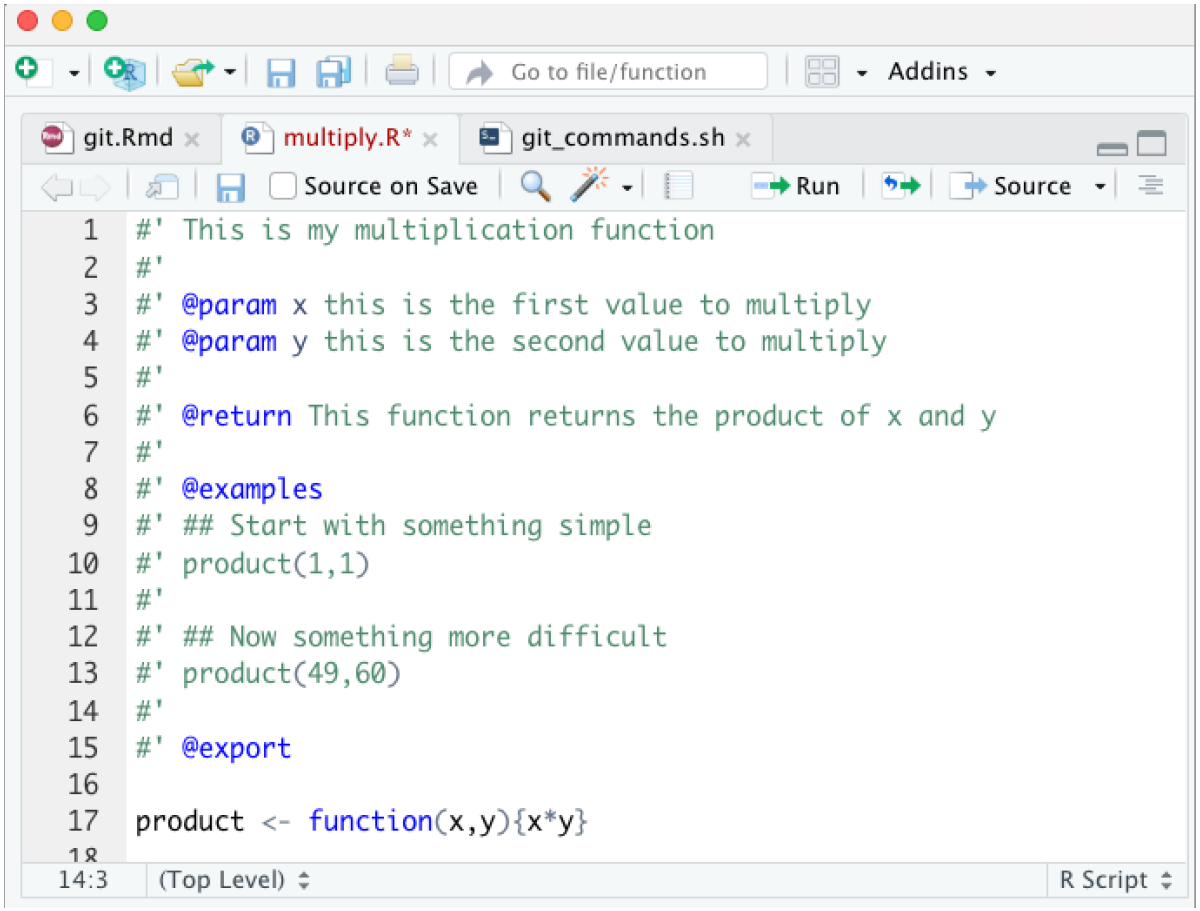
Important note: it is often a good idea to pull any changes when you start each day, so as to avoid **conflicts**.

Working with Git repositories

Now lets do something more substantial.

Create a new file in the `R` directory named `multiply.R` to contain the code displayed at the right.

(Hint: you can copy the `add.R` file and change it)



Now save the file and use the **add**, **commit**, **push** commands to move it to the local and remote repos.

Section 4

More on Git and GitHub

Initializing a Git directory (needed for your homework)

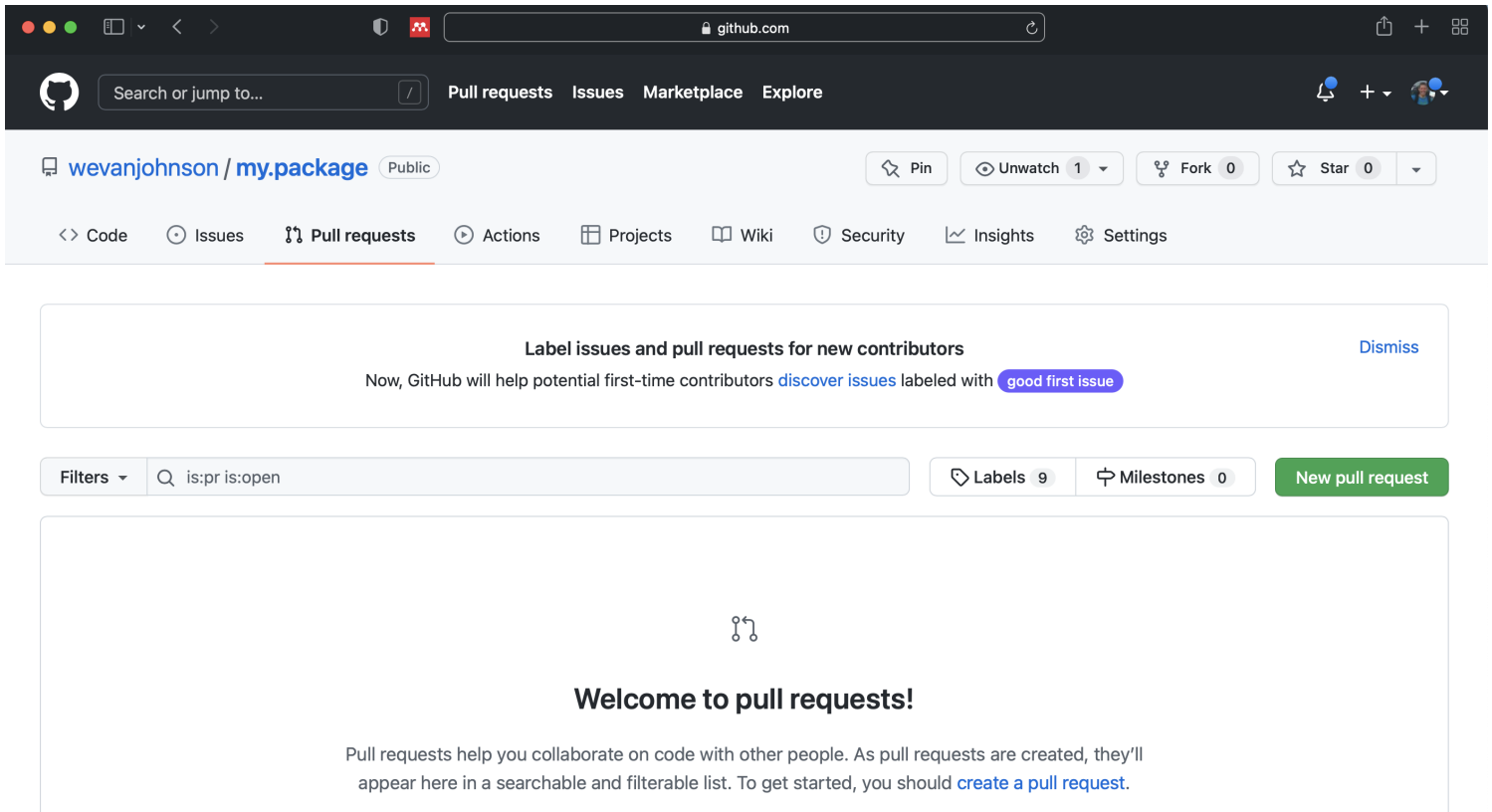
What if we already have a local directory and want to move it to a GitHub repository? See the following:

1. Create a new GitHub repository (e.g., my_fds_homework)
2. **Initialize** the local repository
3. Use the **add** and **commit** commands to add to the local repository
4. Connect the local and remote repos and push:

```
git remote add origin \
'https://github.com/username/my_fds_homework.git'
git push -u origin main
```

Pull requests (needed for your Homework)

Pull requests enable sharing of changes from other branches/forks of a repo. Potential changes can be reviewed before merged into the main branch.



Next Lecture: Advanced R Programming

The next lecture will be on advanced R programming. You will need to have R and RStudio installed on your computer!

Session info

```
sessionInfo()
```

```
## R version 4.5.1 (2025-06-13)
## Platform: aarch64-apple-darwin20
## Running under: macOS Tahoe 26.2
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;  LAPACK version 3
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Denver
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.5.1    fastmap_1.2.0     cli_3.6.5         tools_4.5.1
##  [5] htmltools_0.5.9   otel_0.2.0        rstudioapi_0.17.1 yaml_2.3.12
##  [9] rmarkdown_2.30    knitr_1.51        xfun_0.55         digest_0.6.39
## [13] rlang_1.1.6       evaluate_1.0.5
```