In addition, GitHub makes it easy to share your files with others. GitHub users can fork or clone your repository. People who don’t have GitHub accounts can still browse your shared files online, and even download the entire repository as a zip file.

And finally, once you learn Markdown (which we will be doing here, very soon) you can easily create a webpage for your project, hosted on GitHub, at no cost. This is most commonly used for documentation, but it’s a simple and easy way to get on the web. Just last week, I met a young programmer who showed me his portfolio, hosted on GitHub.

OK, let’s get started!

**Register a GitHub Account**

First, register a free GitHub account: [https://github.com](https://github.com/). For now, just use the free service. You can upgrade to a paid account, create private repositories, join organizations, and other things, later. But one thing you should think about at the very beginning is your username. I would suggest using some variant of your real name. You’ll want something that you feel comfortable revealing to a future potential employer. Also consider that things change; don’t include your current employer, school, or organization as part of your user name.

If you’ve been following along in this series, you’ve already installed R and R Studio. Otherwise, you should do that now. Instructions are in [Part 1](http://geraldbelton.com/getting-started-with-r-a-beginners-guide-part-1/) of this series.

**Installing and Configuring Git**

Next, you’ll need to install git. If you are a Windows user, install [Git for Windows](https://git-for-windows.github.io/" \t "_blank). Just click on the link and follow the instructions. Accept any default settings that are offered during installation. This will install git in a standard location, which makes it easy for RStudio to find it. And it installs a BASH shell, which is a way to use git from a command line. This may come in handy if you want to use git outside of R/RStudio.

LINUX users can install git through their distro’s package manager. Mac users can install git from <https://git-scm.com/downloads>.

Now let’s tell git who you are. Go to a command prompt (or, in R Studio, go to *Tools > Shell*) and type:

git config --global [user.name](http://user.name/) 'Your Name'

For Your Name, substitute your own name, of course. You could use your GitHub user name, or your actual first and last name. It should be something recognizable to your collaborators, as your commits will be tagged with this name.

git config --global user.email '[you@whatever.com](mailto:you@whatever.com)'

The email address you put here **must be** the same one you used when you signed up for GitHub.

To make sure this worked, type:

git config --global --list

and you should see your name and email address in the output.

**Connect Git, GitHub, and RStudio**

Let’s run through an exercise to make sure you can pull from, and push to, GitHub from your computer.

Go to [https://github.com](https://github.com/) and make sure you are logged in. Then click the green “New Repository” button. Give your repository a name. You can call it whatever you want, we are going to delete this shortly. For demonstration purposes, I’m calling mine “demo.” You have the option of adding a description. You should click the checkbox that says “Initialize this repository with a README.” Then click the green “Create Repository” button. You’ve created your first repository!

Click the green “Clone or download” button, and copy the URL to your clipboard. Go to the shell again, and take note of what directory you are in. I’m going to create my repository in a directory called “tmp,” so at the command prompt I typed “mkdir ~/tmp” followed by “cd ~/tmp”.

To clone the repository on your local computer, type “git clone” followed by the url you copied from GitHub. The results should look something like this:

geral@DESKTOP-0HM18A3 MINGW64 ~/tmp

$ git clone <https://github.com/gbelton/>demo.git

Cloning into 'demo'...

remote: Counting objects: 3, done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), done.

Make this your working directory,  list its files, look at the README file, and check how it is connected to GitHub. It should look something like this:

geral@DESKTOP-0HM18A3 MINGW64 ~/tmp

$ cd demo

geral@DESKTOP-0HM18A3 MINGW64 ~/tmp/demo (master)

$ ls

README.md

geral@DESKTOP-0HM18A3 MINGW64 ~/tmp/demo (master)

$ head README.md

# demo

geral@DESKTOP-0HM18A3 MINGW64 ~/tmp/demo (master)

$ git remote show origin

\* remote origin

Fetch URL: <https://github.com/gbelton/>demo.git

Push URL: <https://github.com/gbelton/>demo.git

HEAD branch: master

Remote branch:

master tracked

Local branch configured for 'git pull':

master merges with remote master

Local ref configured for 'git push':

master pushes to master (up to date)

Let’s make a change to a file on your local computer, and push that change to GitHub.

echo "This is a new line I wrote on my computer" >> README.md

git status

And you should see something like this:

$ git status

On branch master

Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: README.md

no changes added to commit (use "git add" and/or "git commit -a")

Now commit the changes, and push them to GitHub:

git add -A

git commit -m "A commit from my local computer"

git push

Git will ask you for your GitHub username and password if you are a new user. Provide them when asked.

The -m flag on the commit is important. If you don’t include it, git will prompt you for it. You should include a message that will tell others (or yourself, months from now) what you are changing with this commit.

Now go back to your browser, and refresh. You should see the line you added to your README file. If you click on commits, you should see the one with the message “My first commit from my local computer.”

Now let’s clean up. You can delete the repository on your local computer just by deleting the directory, as you would any other directory on your computer. On GitHub, (assuming you are still on your repository page) click on “settings.” Scroll down until you see the red “Danger Zone” flag, and click on “Delete This Repository.” Then follow the prompts.

**Connecting GitHub to RStudio**

We are going to repeat what we did above, but this time we are going to do it using RStudio.

Once again, go to GitHub, click “New Repository,” give it a name, check the box to create a README, and create the repository. Click the “clone or download” button and copy the URL to your clipboard.

In RStudio, start a new project: *File > New Project > Version Control > Git*

In the “Repository URL” box, paste in the URL that you copied from GitHub. Put something (maybe “demo”) in the box for the Directory Name. Check the box marked “Open in New Session.” Then click the “Create Project” button.

And, just that easy, you’ve cloned your repository!

In the file pane of RStudio, click README.md, and it should open in the editor pane. Add a line, perhaps one that says “This line was added in R Studio.” Click the disk icon to save the file.

Now we will commit the changes and push them to GitHub. In the upper right pane, click the “Git” tab. Click the “staged” box next to README.md. Click “Commit” and a new box will pop up. It shows you the staged file, and at the bottom of the box you can see exactly what changes you have made. Type a commit message in the box at the top right, something like “Changes from R Studio.” Click the commit button. ANOTHER box pops up, showing the progress of the commit. Close it after it finishes. Then click “Push.” ANOTHER box pops up, showing you the progress of your push. It may ask you for a user name and password. When it’s finished, close it. Now go back to GitHub in your web browser, refresh, and you should see your changed README file.