Lesson 3. First steps with git: clone, add, commit, push

Intro version control git
Max Joseph, Leah Wasser

🎓 Learning objectives

At the end of this activity, you will be able to:

- Create a new repository on GitHub

- `Clone` your repository to your local computer

- Modify files in your repository and track changes using commits with git

- Push your changes back to GitHub

☑ What you need

- A GitHub user account

- A terminal running bash, and

- git installed and configured on your computer.

Follow the setup instructions here:

- Setup instructions

Create a new repository on GitHub

1. To begin, sign in to your user account on GitHub.

2. In the upper right corner, click the `+` sign icon, then choose **New repository**. This will take you to a page where you can enter a repository name (this tutorial uses `test-repo` as the repository name), description, and choose to initialize with a README (a good idea!).

3. It is a good idea to add a `.gitignore` file by selecting one of the languages from the drop down menu, though for this tutorial it will not be necessary.

4. Similarly, in practice you should choose a license to that people know whether and how they can use your code.

5. Once you have entered a repository name and made your selection, select **Create repository**, and you will be taken to your new repository web page.

Git at the command line

Below you will learn a series of commands that you can run at the command line in git bash, terminal of whatever bash tool you are using. There are 2 types of commands that you will use

1. **Bash commands:** These are commands that are native to bash / shell. They allow you to navigate around your computer, explore directory structures, create and manipulate files and directories, and more. (e.g. `ls`, `cd`, `mkdir`, etc)

2. **Git commands:** These are commands that are specific to git and will only be available if you have git installed on your computer. Git specific commands will always started with a call to git (e.g. `git status`, `git clone`, etc)

Clone your repository to your local machine

Next, clone your newly created repository from GitHub to your local computer. From your repository page on GitHub, click the green button labeled **Clone or download**, and in the "Clone with HTTPs" section, copy the URL for your repository.

Next, on your local machine, open your bash shell and change your current working directory to the location where you would like to clone your repository. Note that here we are using a bash command - `cd` (change directory).

For example, on a Unix based system, if you wanted to have your repository in your `Documents` folder, you change directories as follows:

```
</>

cd Documents
```

Once you have navigated to the directory where you want to put your repository, you can use:

`git clone https://github.com/URL-TO-REPO-HERE`

The `git clone` command copies your repository from GitHub to your local computer. Note that this is a git specific command.

```
</>

git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
```

When you run `git clone repo-path-here`, You should see output like:

```
</>

Cloning into 'test-repo'...
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (5/5), done.
Checking connectivity... done.
```

Note: The repository name and output numbers that you see on your computer, representing the total file size, etc, may differ from the example provided above.

To verify that your repository now exists locally, type `ls` in your terminal. The ls command lists the files & folders available in your current directory. You should see a directory with the same name as the repository that you created previously on GitHub.

Tracking changes with `git add` and `git commit`

Next use `cd` to **c**hange **d**irectories using the syntax:

`cd my-repo-name`

Replace my-repo-name with the folder name of your repo (this should be your repo name - e.g. 14ers-git)

```
</>

    cd test-repo
```

If you list all the files in this directory (using `ls -a`), you should see all of the files that exist in your GitHub repository:

```
</>

    ls -a
```

```
</>

    .git  .gitignore  LICENSE  README.md
```

Alternatively, we can view the local repository in the finder (Mac), a Windows Explorer (Windows) window, or GUI file browser (Linux).

Simply open your file browser and navigate to the new local repo.

> ★ **Important Tip** The `.git` element is listed when you use the `ls -a` command shows up is actually a directory which will keep track of your changes (the commits that you make) in git. **Warning:** Do not edit the files in this directory manually!

Using either method, we can see that the file structure of our cloned repo mirrors the file structure of our forked GitHub repo.

Edit a file in your repo

Next, open up your favorite text editor and make a few edits to the `README.md` file. Save your changes.

Once you are happy with your changes and have saved them, go back to your terminal window and type `git status` and hit return to execute the command.

```
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")
```

The output from `git status` indicates that you have modified the file `README.md`. To keep track of this change to this file, you need to
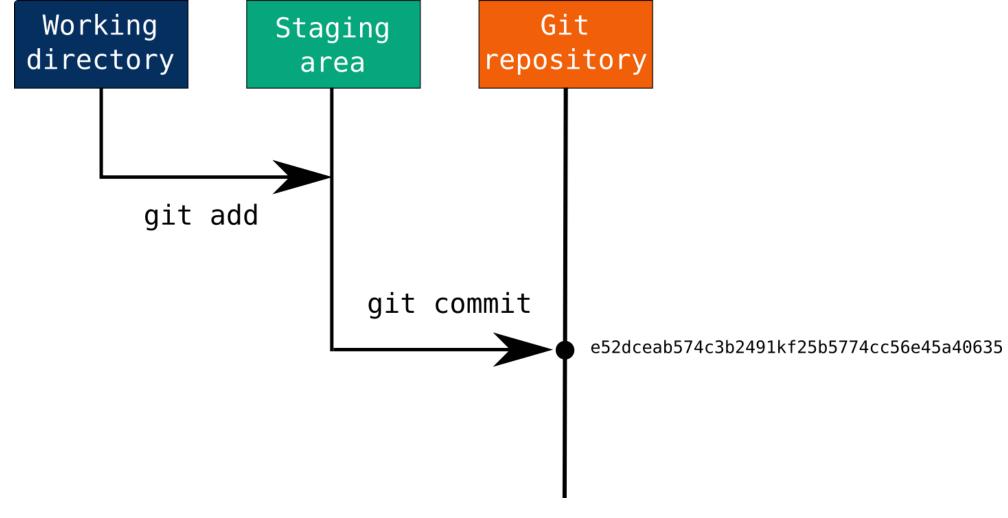
1. `add` the changes, then
2. `commit` the changes.

Add and commit changes

You will use the add and commit functions to add and commit changes that you make to git.

- `git add` : takes a modified file in your working directory and places the modified version in a staging area.
- `git commit` takes everything from the staging area and makes a permanent snapshot of the current state of your repository that is associated with a unique identifier.

These two commands make up the bulk of many workflows that use git for version control.

Modified files are staged using git add, and following a commit, all files in the staging area are snapshotted and become part of the repository's history, receiving a unique SHA-1 hash identifier. Source: Maxwell Joseph, adapted from Pro Git by Chacon and Straub (2014).

Add files

You can add an individual file or groups of files to git tracking. To add a single file, use

```
git add file-name-here-with-extension
```

To add the README.md file that you just modified, you'd use:

```
</>

    git add README.md
```

To add ALL of the files that you have edited at the same time, you can use:

```
</>

    git add --all
```

Use `git add --all` with caution. You do not want to accidentally add things like credential files, `.DS_Store` files, or history files.

Commit files

Once you are ready to make a snapshot of the current state of your repository, you can use `git commit`. The git commit command requires a **commit message** that describes the snapshot / changes that you made in that commit.

A commit message should outline what changed and why. These messages

1. help collaborators and your future self understand what was changed and why
2. allow you and your collaborators to find (and undo if necessary) changes that were previously made.

If you are not committing a lot of changes, you can create a short one line commit message using the `-m` flag:

```
</>

    git commit -m "Editing the README to try out git add/commit"
```

Alternatively, if you are committing many changes, or a small number of changes that require explanation, you'll want to write a detailed multi-line commit message using a text editor.

If you have configured git to use your favorite text editor (via `git config --global core.editor your-fav-editor-here`), then you can open that editor to write your commit message using the `git commit` command:

```
git commit
```

Once you save your commit message and exit the text editor, the file that you created will contain your commit message.

✍ Challenge

Make changes to files in your git repo. These changes may includes

1. adding new files to the repo
2. adding content to the files (Text)
3. adding new directories to your repo

For each change, practice using

- `git add` to stage the change, and
- `git commit` to take a snapshot of your repository

After you've made a few commits, check out the output of the `git log` command. You should see the history of your repository, including all of the commit messages!

```
git log
```

```
commit 778a307bcc8350bddba47e96a940acafed55f5d8
Author: Fred Flinstone <flinstone@bedrock.com>
Date:   Tue Sep 19 18:38:28 2017 -0600

    adding a file in a subdirectory

commit f2b0ff9af905fa2792bf012982e10f0214148c70
Author: Fred Flinstone <flinstone@bedrock.com>
```

```
Date:    Tue Sep 19 16:50:59 2017 -0600

    fixing typo


commit e52dceab576c3b2491af25b5774cc56e65a40635
Author: Fred Flinstone <flinstone@bedrock.com>
Date:    Tue Sep 19 10:27:05 2017 -0600

    Initial commit
```

Push changes to GitHub

So far we have only modified our local copy of the repository. To add the changes to your git repo files on your computer to the version of your repository on GitHub, you need to **push** them GitHub.

You can push your changes to GitHub with:

```
                                                                    </>

    git push

```

You will then be prompted for your GitHub user name and password. After you've pushed your commits, visit your repository on GitHub and notice that your changes are reflected there, and also that you have access to the full commit history for your repository!

Additional resources

- Creating a new repository on GitHub
- Git and GitHub learning resources

🏷 Tags 🏷 **Reproducible science and programming:** [git](#) , [version control](#)

📅 **Updated:** September 03, 2019

The workshop is subject to the [CC BY-NC-ND 4.0 License](#) . Citation DOI: https://zenodo.org/badge/latestdoi/186890040

LEAVE A COMMENT

1 Comment   **earthlabcu**   ① **Login** ▾

♡ **Recommend** 1   🐦 **Tweet**   f **Share**   Sort by Best ▾

👤	Join the discussion…

LOG IN WITH   **OR SIGN UP WITH DISQUS** ⑦

Name

👤 **moon** • 6 months ago

Thank you. Learned a few things, despite looking at numerous other tutorials.

∧ | ∨ • Reply • Share ›

✉ **Subscribe**   ⅅ **Add Disqus to your site**Add DisqusAdd   🔒 **Disqus' Privacy Policy**Privacy PolicyPrivacy