Git and GitHub

a getting-started guide for new(ish) Macaulay2 developers

OUTLINE:

Git & GitHub

Git Concepts

WORKFLOWS

(bonus) Terminal Stuff

Git

Version Control Software

Developed in 2005 to manage Linux development

GitHub

GitHub is an online UI and wrapper for Git that adds extra development features:

wiki, issues, pull requests, permissions, static hosting, automated testing, etc.

Getting Git

open your terminal to check for git:

```
1    crgibbon@MacBookCaulay2:~$ git --version
2    git version 2.17.1
```

if it's not already installed,

visit https://git-scm.com/downloads

Git Concepts

You will develop code on your own personal branch locally (on your computer) until you're happy with it.

Then you'll **commit** that code to your branch and **push** it to your branch's remote copy.

Eventually, you will **merge** your commits into the team branch.

Branches

remote branch:
the copy on the server

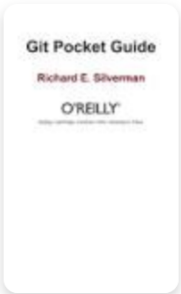
remote tracking branch:
a mystery you need not understand

local branch:
the copy that's on your computer

Git Pocket Guide

A Working Introduction

By Richard E. Silverman · 2013



Preview | Search inside | + Add to my library

Overview | Get the book | More by author | Similar books

About this edition

ISBN:	9781449327521, 1449327524	Page count:	234
Published:	June 25, 2013	Format:	E-book
Publisher:	O'Reilly Media	Language:	English
Author:	Richard E. Silverman		

Create Citation

This pocket guide is the perfect on-the-job companion to Git, the distributed version control system. It provides a compact, readable introduction to Git for new users, as well as a reference to common commands and procedures for those of you with Git experience.

Written for Git version 1.8.2, this handy task-oriented guide is organized around the basic version control functions you need, such as making commits, fixing mistakes, merging, and searching history.

About the work

Originally published: 2013

Subject: Computers / Operating Systems / General, MORE

Author

Richard E. Silverman

Richard E. Silverman has a B.A. in computer science and an M.A. in pure mathematics. Richard has worked in the fields of networking, formal methods in software development, public-key infrastructure, routing security, and Unix systems administration. He co-authored the SSH, The Secure Shell: The Definitive Guide, 2e and the Linux Security Cookbook.

my favorite reference for all things git

Commit: the basic unit of Git

A **commit** is a set of changes to a set of files (i.e., a **patch**) along with a short explanatory message.

```
commit b0f3ac9 (HEAD -> CourtneyG)
Author: Courtney Gibbons <crgibbon@hamilton.edu>
Date:   Sun May 10 12:58:07 2020 -0400

    Add ZZ[x,y] test for CoolFunction

    Ensure that CoolFunction returns correct values on
    tricky inputs like 'ZZ[x,y]'.

diff --git a/Tests/CoolFunctionTests.m2 b/Tests/CoolFunctionTests.m2
index e69de29..8e016b9 100644
--- a/Tests/CoolFunctionTests.m2
+++ b/Tests/CoolFunctionTests.m2
@@ -0,0 +1 @@
+assert( CoolFunction(ZZ[x,y]) == 42 )
```

Each commit is uniquely identified across the project by its **hash** (`b0f3ac9...`)

The Log

The **log** shows the project's history of commits. You can access it with **`git log`**.

```
commit 2d5d8ff (HEAD -> CourtneyG)
Author: Courtney Gibbons <crgibbon@hamilton.edu>
Date:   Sun May 10 12:30:03 2020 -0400

    Add tests for CoolFunction

commit f44ea55 (origin/AwesomePackage, AwesomePackage)
Author: Courtney Gibbons
Date:   Fri May 8 15:42:58 2020 -0400

    Create AwesomePackage.m2

commit ae42c7c (origin/master, origin/HEAD, master)
Author: Federico Galetto
Date:   Sat May 2 18:43:03 2020 -0400

    Add files via upload

commit 78ba33e
Author: Federico Galetto
Date:   Sat May 2 18:42:34 2020 -0400

    Create schedule.md

commit d335d06
Author: Branden Stone
Date:   Wed Apr 22 10:13:07 2020 -0400

    modified README
```

Creating a Commit

When you've done a coherent block of work, it's time to create a commit.

To do so, first **add** the changes you want to the **index** (a staging area) using **`git add`**, then tell git to create a commit using **`git commit`**.

```
$ git status
On branch CourtneyG
Your branch is ahead of 'AwesomePackage' by 2 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   AwesomePackage.m2

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

        OkPackage.m2
        OutstandingPackage.m2

$ git add AwesomePackage.m2
$ git add OutstandingPackage.m2
$ git status
On branch CourtneyG
Your branch is ahead of 'AwesomePackage' by 2 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   AwesomePackage.m2
        new file:   OutstandingPackage.m2

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

        OkPackage.m2

$ git commit -m "Add really primo packages"
[CourtneyG a3986b4] Add really primo packages
 2 files changed, 2 insertions(+), 2 deletions(-)
 create mode 100644 OutstandingPackage.m2
```

Pushing Commits

Creating a commit does not push it to the remote server: your changes are still local.

To share your changes, you must **push** them to your remote branch using **`git push`**. Then anyone who checks out your branch will see these changes.

```
$ git push origin CourtneyG
Counting objects: 11, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (7/7), done.
Writing objects: 100% (11/11), 1.16 KiB | 396.00 KiB/s, done.
Total 11 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
remote:
remote: Create a pull request for 'CourtneyG' on GitHub by visiting:
remote:      https://github.com/Macaulay2/Workshop-2020-Cleveland/pull/new/CourtneyG
remote:
To https://github.com/Macaulay2/Workshop-2020-Cleveland.git
 * [new branch]      CourtneyG -> CourtneyG
```

Merging Code

Finished up your changes to the function your team leader asked you to work on? Tested it? Then it's time to **merge** your work into the team branch.

```
$ git checkout AwesomePackage
Switched to branch 'AwesomePackage'
Your branch is up to date with 'origin/AwesomePackage'.
$ git merge CourtneyG
Updating f44ea55..a3986b4
Fast-forward
 AwesomePackage.m2          | 4 ++--
 OutstandingPackage.m2      | 0
 Tests/CoolFunctionTests.m2 | 1 +
 3 files changed, 3 insertions(+), 2 deletions(-)
 create mode 100644 OutstandingPackage.m2
 create mode 100644 Tests/CoolFunctionTests.m2
```

Merge Conflicts

If all goes well, your merge will take care of itself.

But **conflicts** are inevitable. To deal with them, open the file with conflicts in your favorite editor and manually fix them.

```
$ git merge CourtneyG
Auto-merging Tests/CoolFunctionTests.m2
CONFLICT (content): Merge conflict in Tests/CoolFunctionTests.m2
Automatic merge failed; fix conflicts and then commit the result.

$ git show Tests/CoolFunctionTests.m2
commit 06c6f0f (HEAD -> AwesomePackage)
Author: Courtney Gibbons <crgibbon@hamilton.edu>
Date:   Sun May 10 14:15:17 2020 -0400

    update CoolFunctionsTest

index 8e016b9..bf80f30 100644
--- a/Tests/CoolFunctionTests.m2
+++ b/Tests/CoolFunctionTests.m2
@@ -1 +1 @@
-assert( CoolFunction(ZZ[x,y]) == 42 )
+assert( CoolFunction(ZZ[x,y]) == 43 )

$ cat Tests/CoolFunctionTests.m2   # show the conflicts
<<<<<<< HEAD
assert( CoolFunction(ZZ[x,y]) == 43 )
=======
assert( CoolFunction(ZZ[x,y]) == 44 )
>>>>>>> CourtneyG

$ git checkout CourtneyG Tests/CoolFunctionTests.m2
# just use Courtney's changes

$ cat Tests/CoolFunctionTests.m2
assert( CoolFunction(ZZ[x,y]) == 44 )

$ git add Tests/CoolFunctionTests.m2   # update the index

$ git merge --continue   # finish the merge
[AwesomePackage e3e9bb4] Merge branch 'CourtneyG' into AwesomePackage
```

Other Useful Git Commands

git pull	gets updates from remote branch(es)
git push	sends commits to remote branch
git status	shows the status (staged for commit or not) of files in your local branch
git add	adds files to the list of stuff to commit
git commit -m "Do this thing"	creates a commit with a message
git log -p	show code changes in each commit
git log --name-status	show files changed in each commit
git checkout BranchName	checks out the branch BranchName
git merge --abort	bail out of here
git diff	show unstaged changes
git diff --cached	show staged changes
git branch --all	see all branches (local and remote)
git diff another_branch	show the differences between your branch and another_branch
git reflog	rescue missing code you thought was lost forever

Git is on your side

When you input a command, git often gives you helpful suggestions.

```
$ git status
On branch CourtneyG
Your branch is ahead of 'AwesomePackage' by 2 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   AwesomePackage.m2

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

        OkPackage.m2
        OutstandingPackage.m2

$ git add AwesomePackage.m2
$ git add OutstandingPackage.m2
$ git status
On branch CourtneyG
Your branch is ahead of 'AwesomePackage' by 2 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   AwesomePackage.m2
        new file:   OutstandingPackage.m2

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

        OkPackage.m2

$ git commit -m "Add really primo packages"
[CourtneyG a3986b4] Add really primo packages
 2 files changed, 2 insertions(+), 2 deletions(-)
 create mode 100644 OutstandingPackage.m2
```

WORKFLOWS

Today:

Team Leaders - Create Team Branches

Every day:

All developers -

Checkout the team branch

Create your own branch from it

Develop your feature

Merge your code back into the team branch

Delete your branch, start next feature

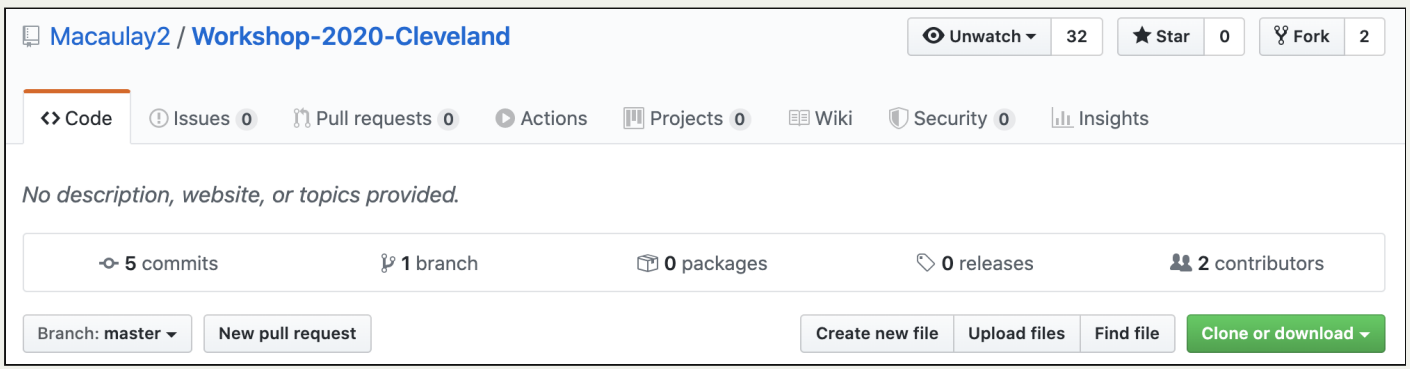
Team leader - submit Pull Request

Working on a Package

Clone our repository (in a pinch, via the website)

https://github.com/Macaulay2/Workshop-2020-Cleveland

click "clone or download" (and use https)



1. In terminal, navigate to where you want to work and type:

```
$ git clone https://github.com/Macaulay2/Workshop-2020-Cleveland.git
```

Team Leaders: Working on a Package

2. **Team leader only:** Create package, add, commit, and push it:

Working on an existing package? Get a copy of the package from the Macaulay2 repository:

https://github.com/Macaulay2/M2/tree/master/M2/Macaulay2/packages

and save it to your working directory. Otherwise, create a template package by copying

```
$ cp PackageTemplate.m2 AwesomePackage.m2
$ git add AwesomePackage.m2
$ git commit -m "Create AwesomePackage.m2"
$ git push origin master
```

3. **Team leader only:** create **team** working branch

```
$ git checkout -b AwesomePackage
$ git push --set-upstream origin AwesomePackage
```

Everyone: Working on a Package

4. Everyone (includes team leaders): create **your own** branch

```
$ git pull   # gets the latest stuff in the repo
$ git checkout AwesomePackage   # checks out a branch
$ git checkout -b CourtneyG   # copies the branch to your branch
$ git push --set-upstream origin CourtneyG
```

5. Work on your code, add changes, commit them, and push

```
$ git status   # checks to see what you've been up to
$ git add <files or directories to commit>   # stages for commit
$ git commit -m "Imperative tense description of changes"
$ git push   # pushes commits to remote branch
```

Repeat until you finish a feature...

Everyone: Working on a Package

6. End of feature: merge your branch into the team branch

```
$ git checkout AwesomePackage
$ git pull
$ git merge CourtneyG   # prompts you to resolve conflicts if any
```

7. Delete your branch (it's okay, your work is still saved!)

```
$ git branch -d CourtneyG   # deletes local copy
$ git push origin --delete CourtneyG   # deletes remote copy
```

8. Start again from Step 4 for your next feature

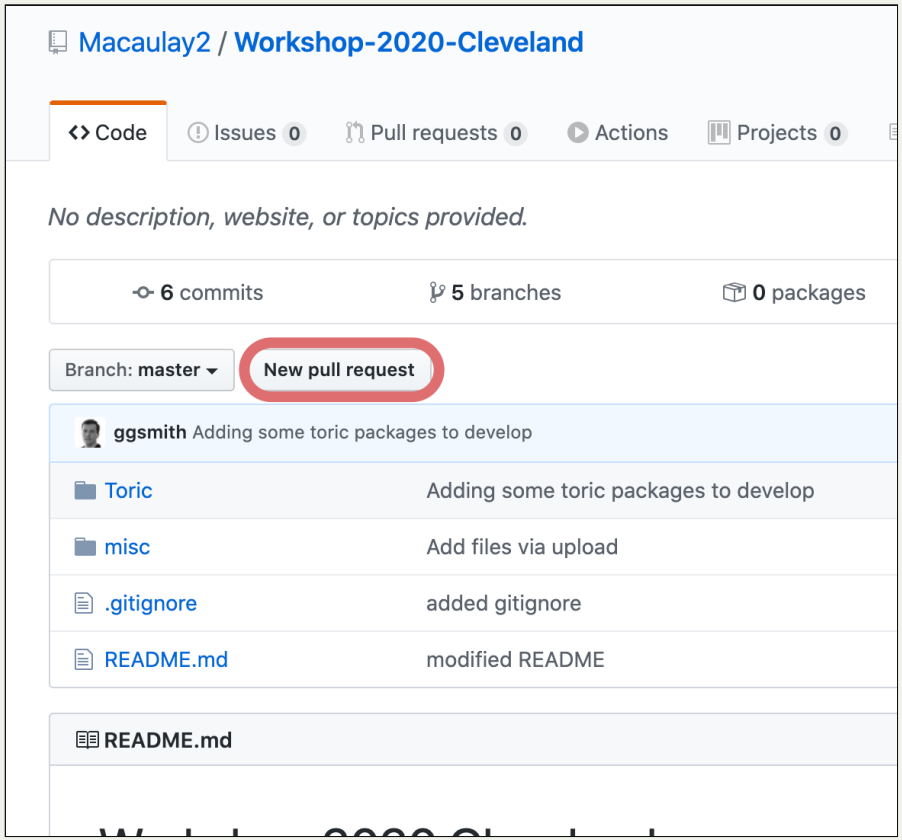
Team Leaders:
Working on a Package

End of day:

create a pull request

for your team branch

in 3 easy steps!



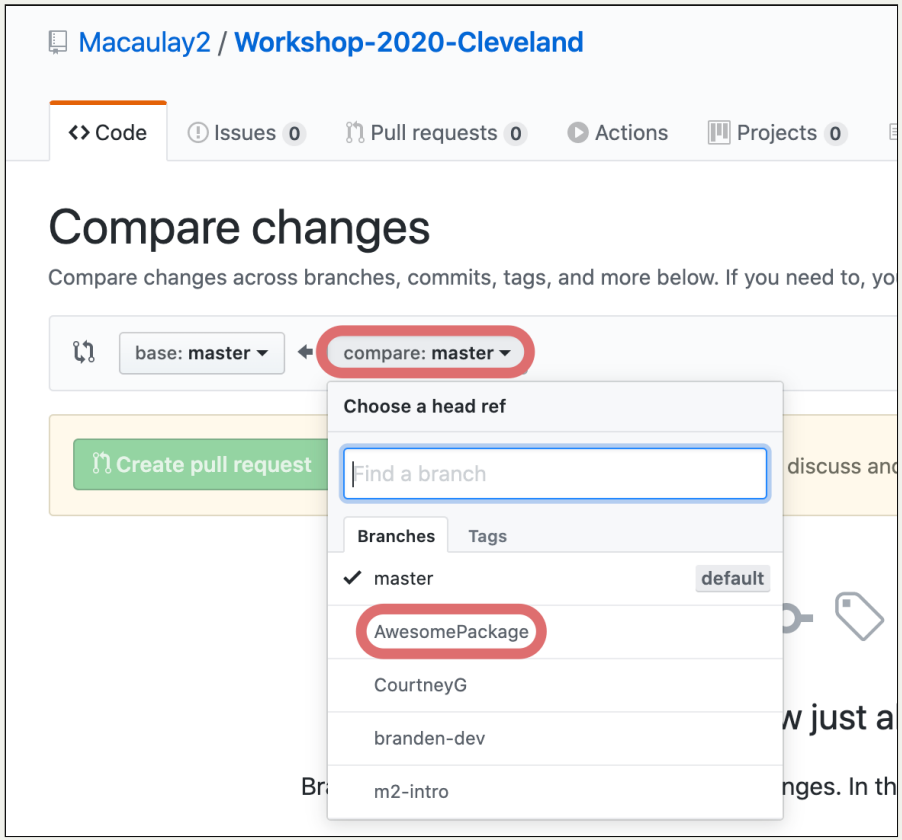
Team Leaders:
Working on a Package

End of day:

create a pull request

for your team branch

in 3 easy steps!



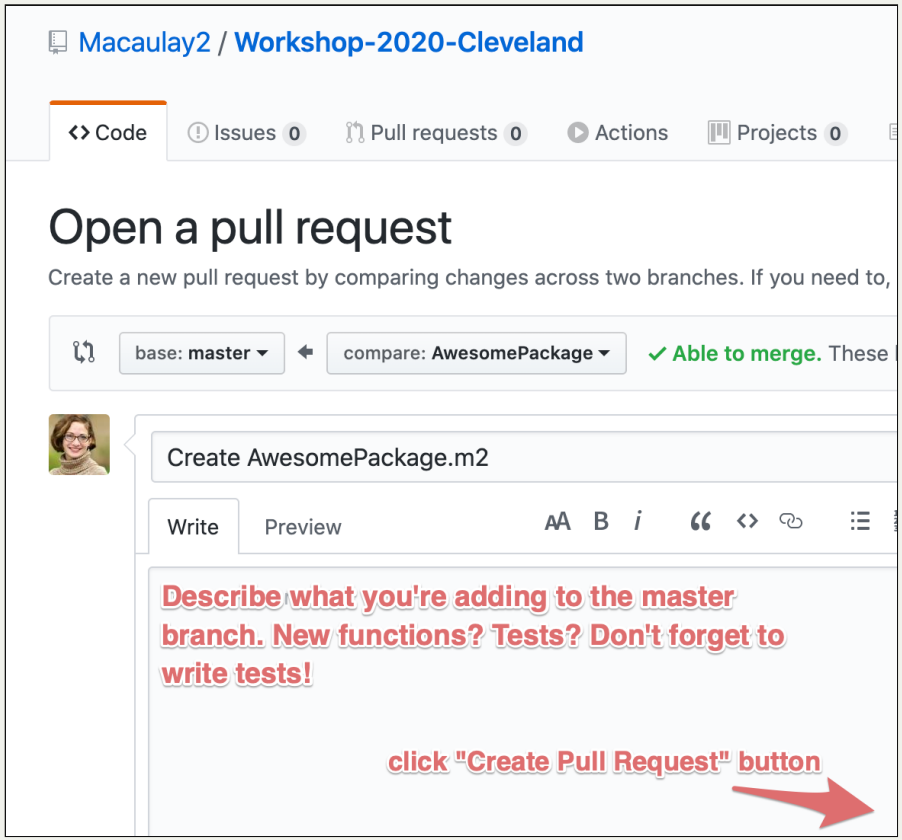
Team Leaders:
Working on a Package

End of day:

create a pull request

for your team branch

in 3 easy steps!



HELP!

Git help in the #help-git channel on slack

Terminal Stuff

- Changing Directories:
  - **`ls -a`** # tells you what's in the current directory
  - **`cd PATH`** # changes directory to a path
    - eg: **`cd ~/2020M2/Workshop-Cleveland-2020/`**
  - **`mkdir DIR`** # makes a new folder
    - eg: **`mkdir MyCode/`**
- Exiting VIM:
  - **`esc`**
  - **`:wq`** # write out and quit

Terminal Stuff

- Changing Default Git Editor to Emacs:
  - **`git config --global core.editor emacs`**
- Exiting Emacs:
  - **`C-x C-c`**
- Install Command Line Tools (mac):
  - **`xcode-select --install`**