Software Engineering Techniques for Collaborative Software Development

Princeton University Bootcamp 2018

David Luet

PICSciE, Research Computing/OIT, Department of Geosciences

October 30, 2018

Outline

Git and GitHub for Collaborative Developments

Testing

Automatic Testing

Other Useful Tools For Collaborative Software Development

References and Getting Help

Conclusion

Outline

Git and GitHub for Collaborative Developments

Testing

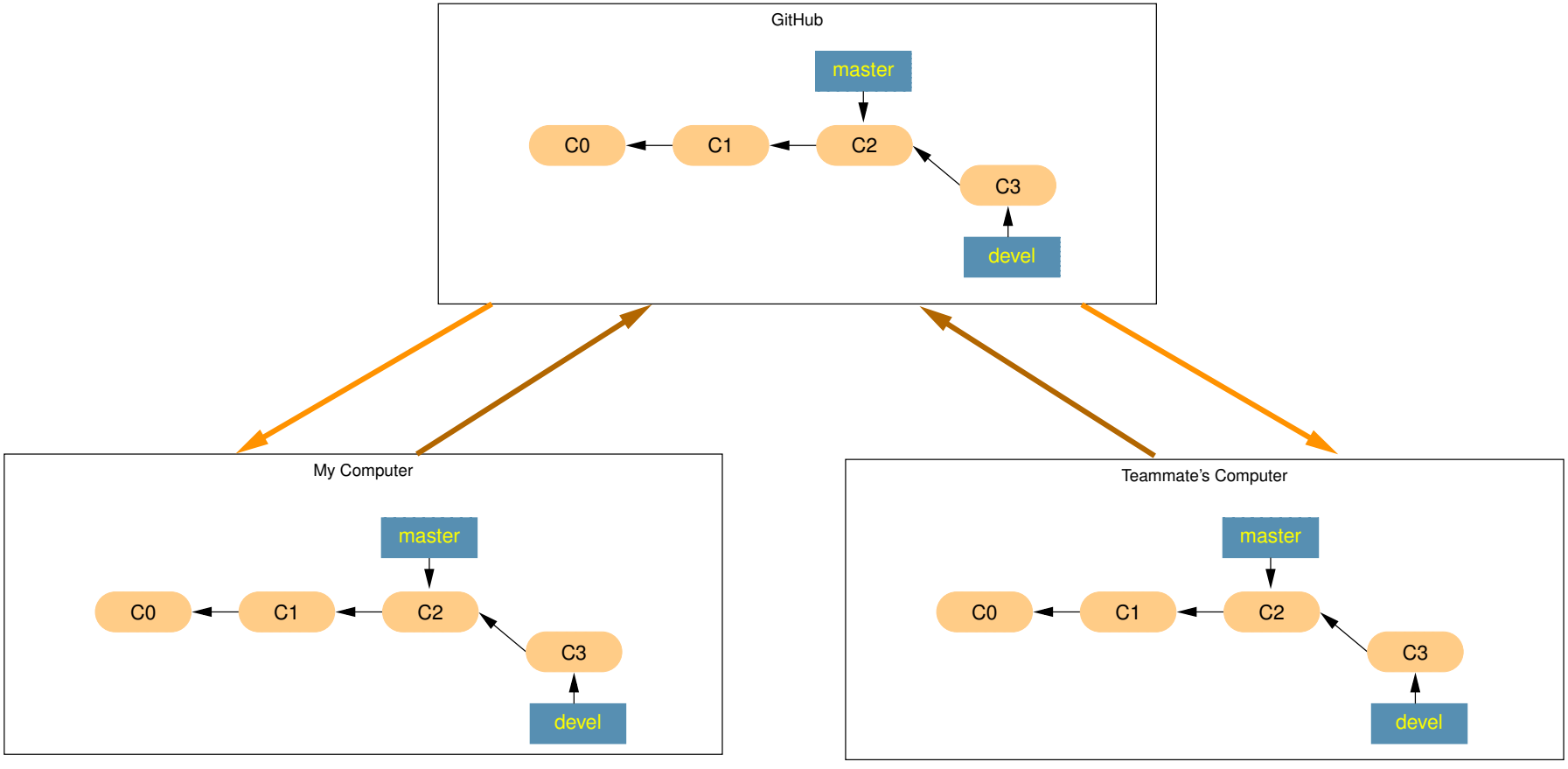
Automatic Testing

Other Useful Tools For Collaborative Software Development

References and Getting Help

Conclusion

A Simple Collaborative Workflow

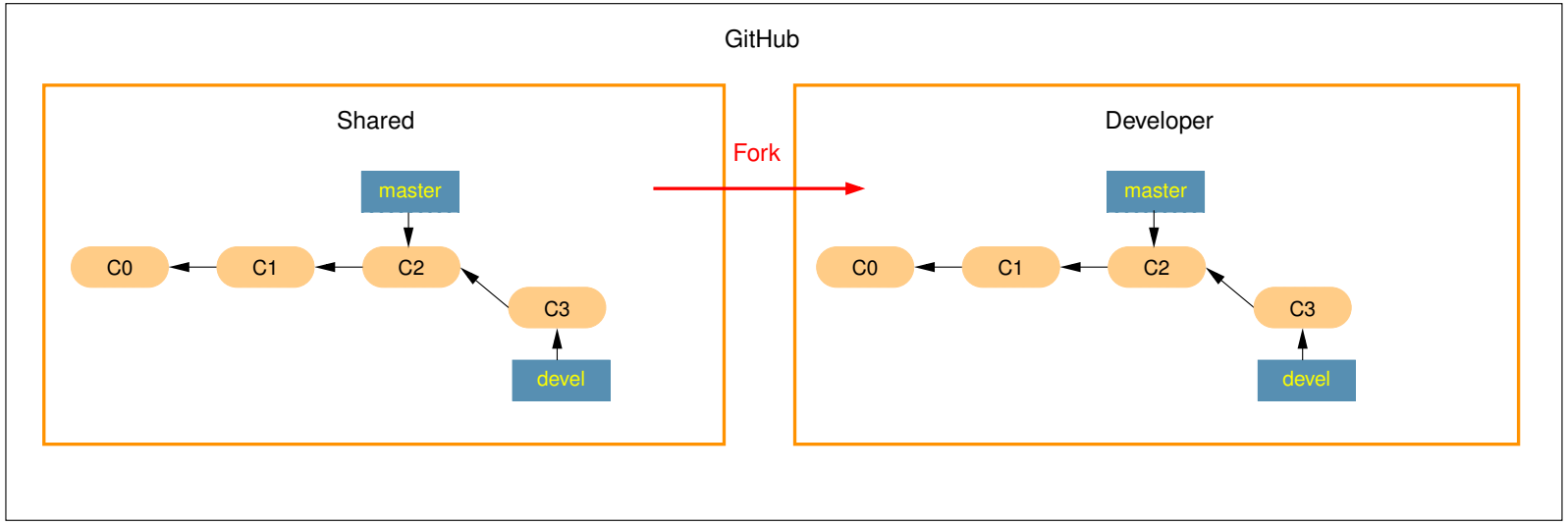


Remote Repository Permissions

- Problem with that simple workflow:
  - It's easy to push broken code to the shared repository.
  - when others pull the changes and start adding their development, it can create problem.
  - this does not work well with more than two developers.
- To solve this issue, we introduce two roles with different permissions on the shared repository:
  - code maintainers: push and pull permissions.
  - developers: only pull permission.

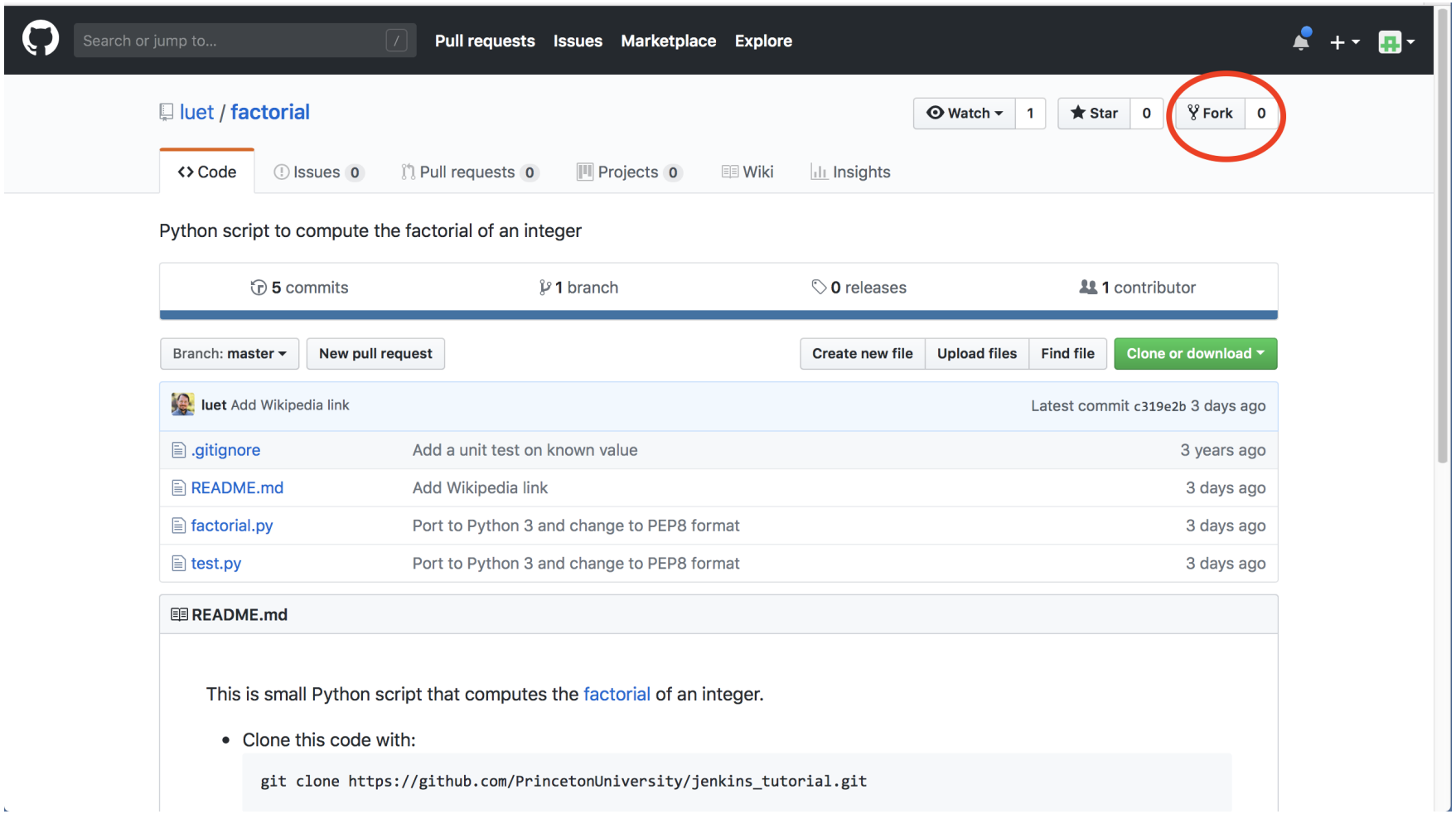
Forking a Repository on GitHub

Forks are basically a copy of a repo on GitHub.

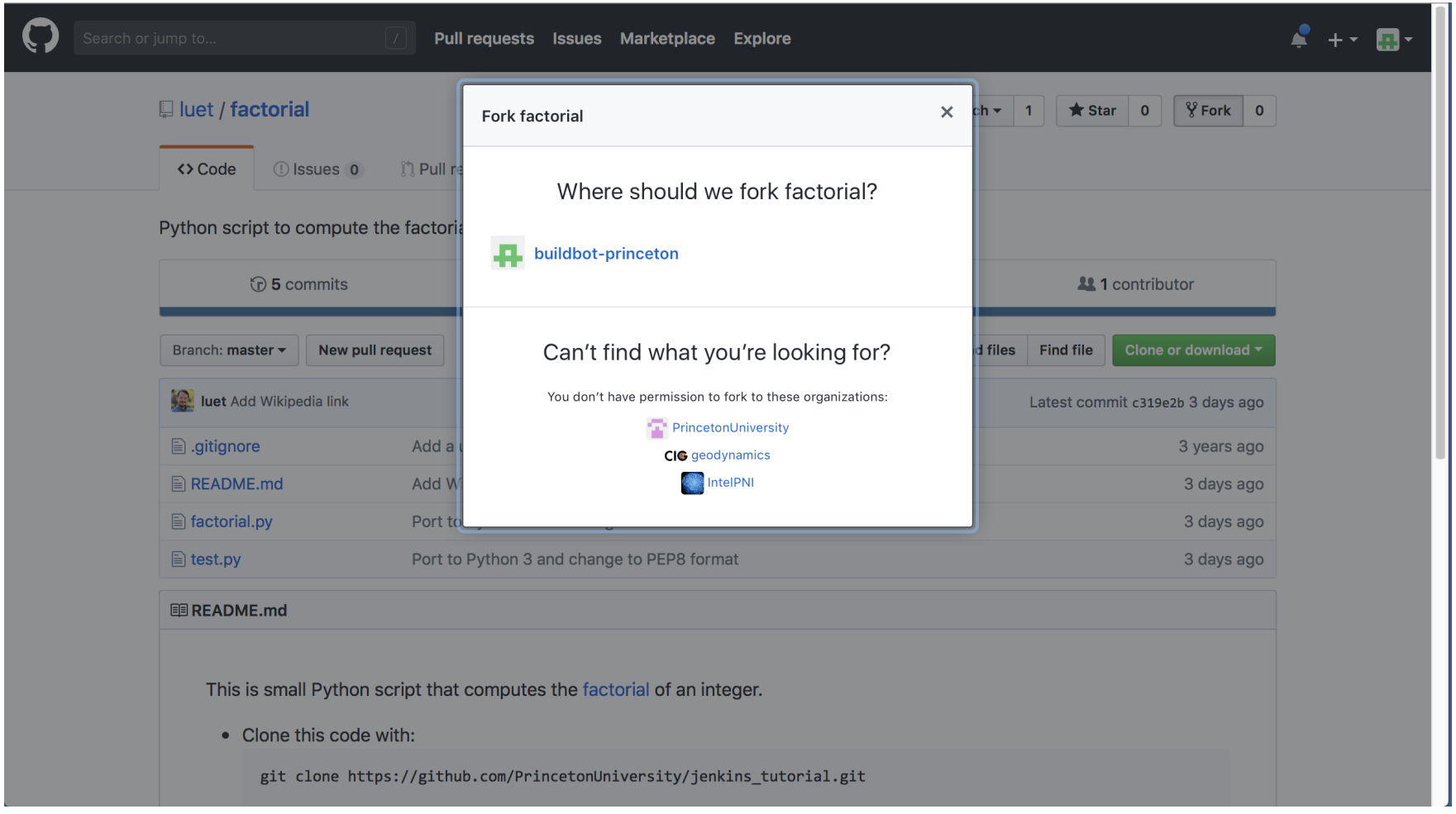


Forking a Repository on GitHub

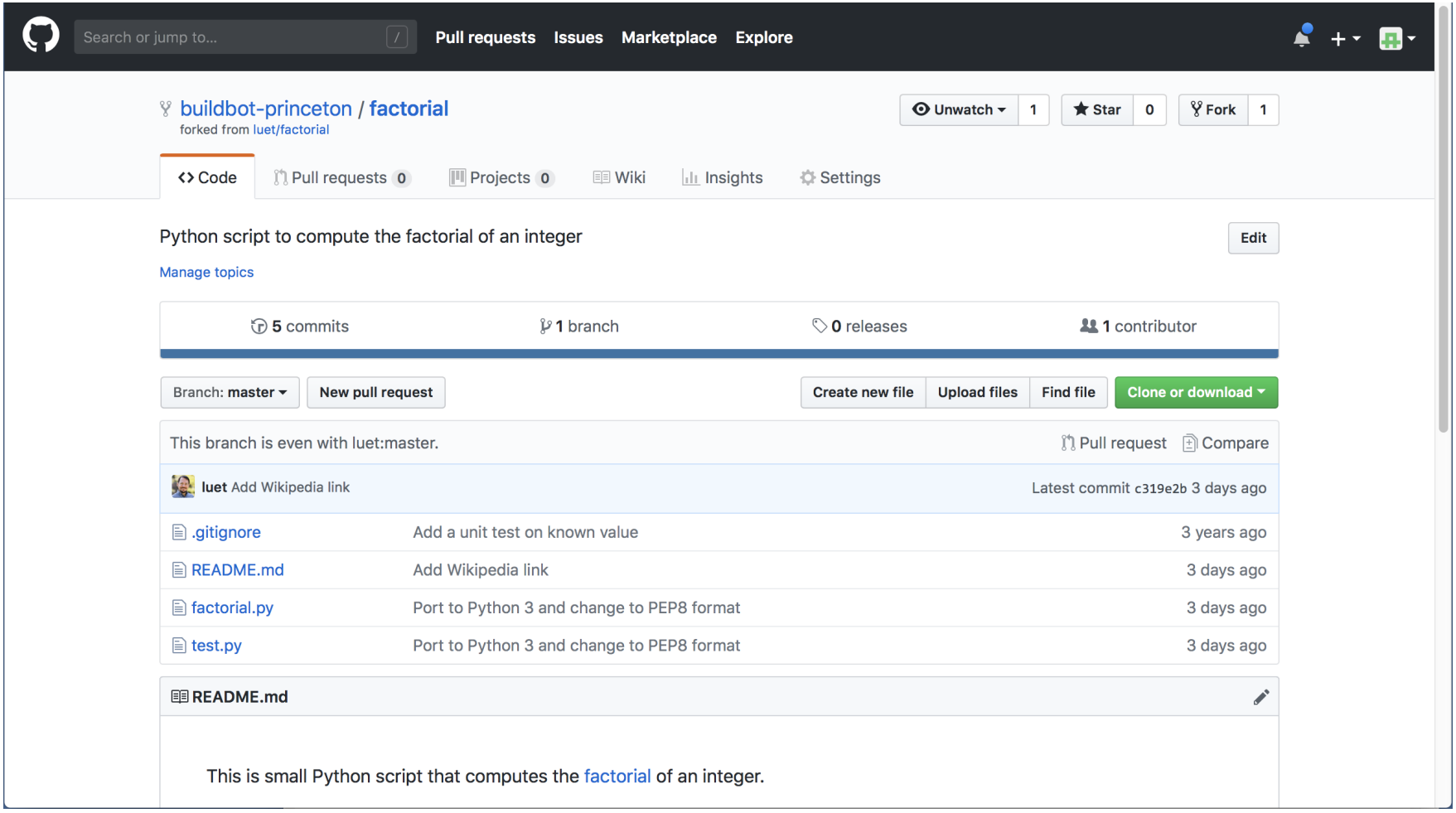
As the GitHub user `buildbot-princeton` I want to fork:
`https://github.com/luet/factorial`



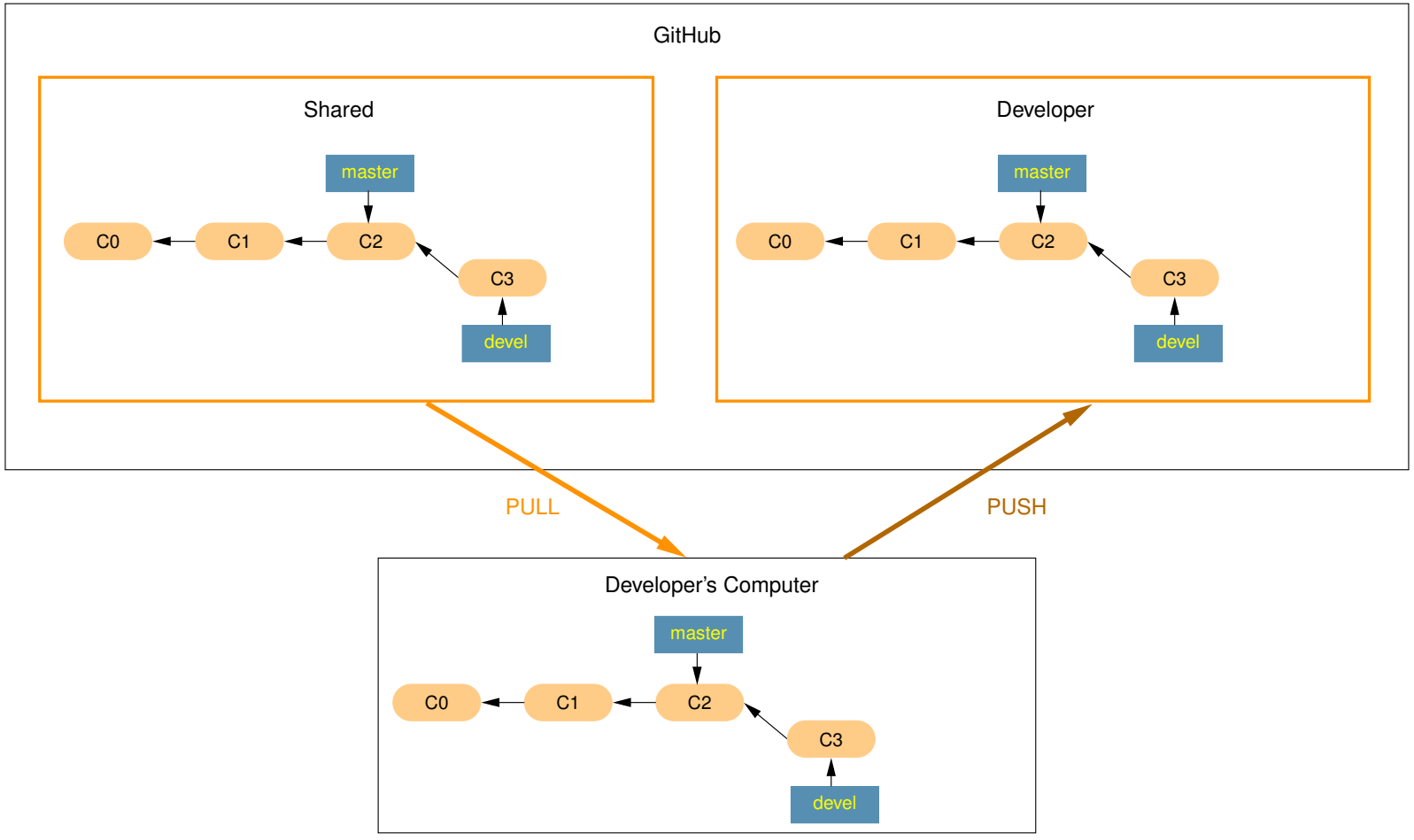
Forking a Repository on GitHub



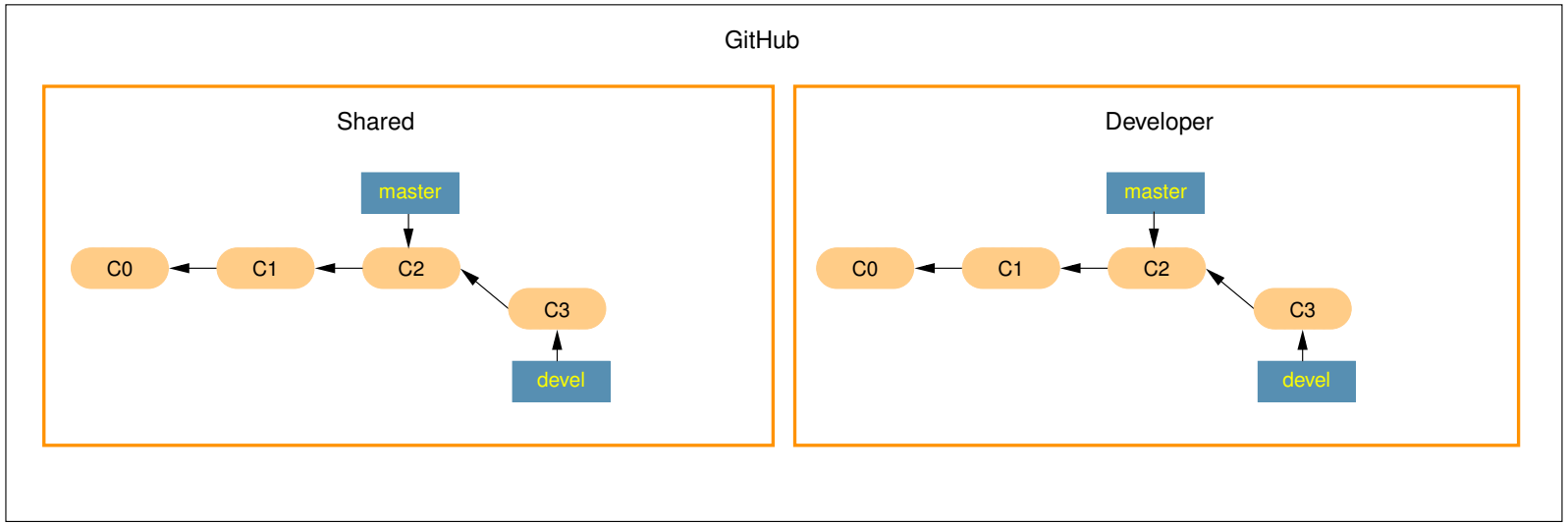
Forking a Repository on GitHub

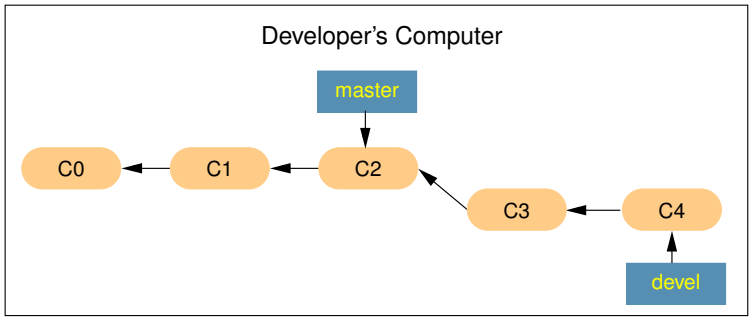


Pull-Request: the Different Repositories

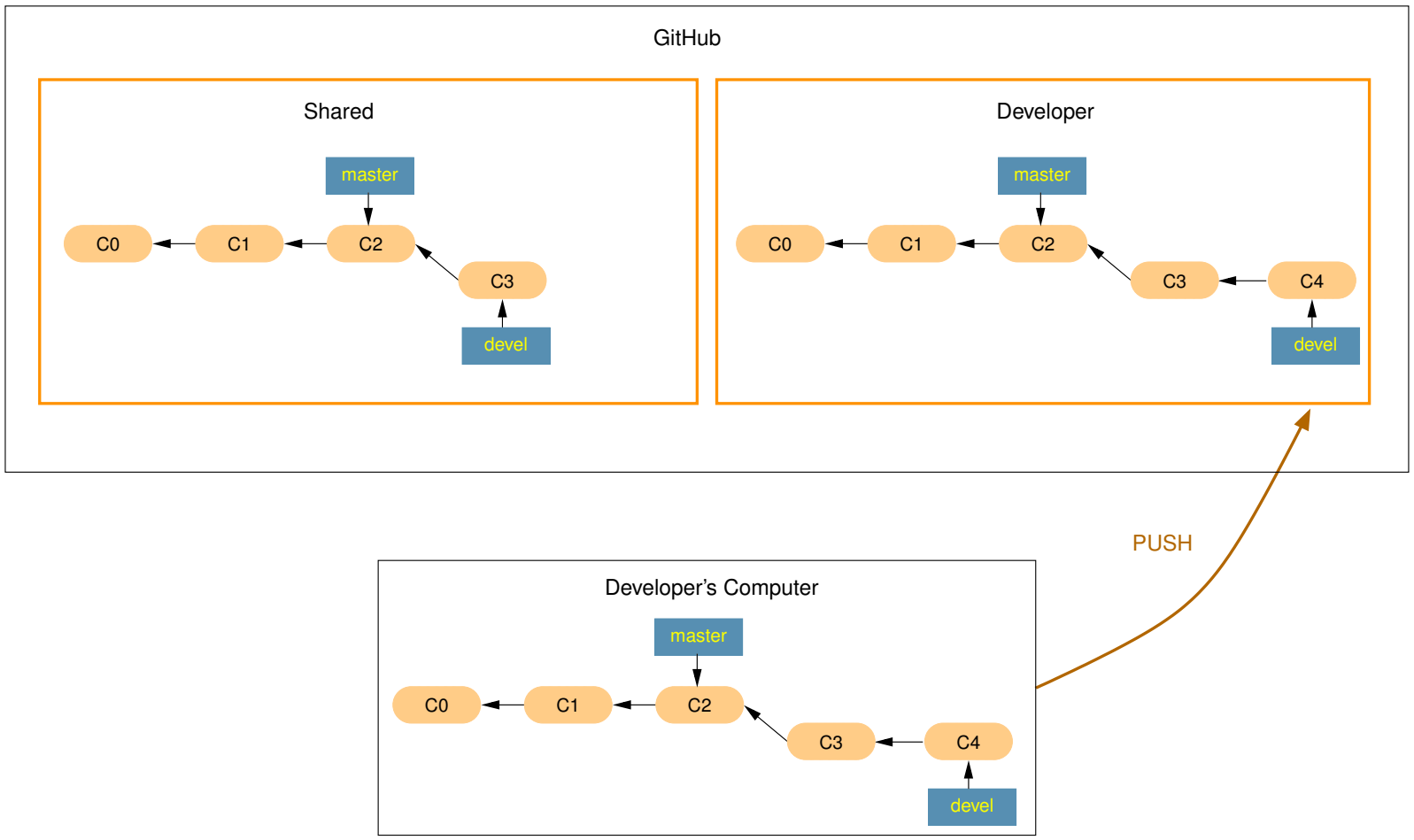


Pull-Request Steps

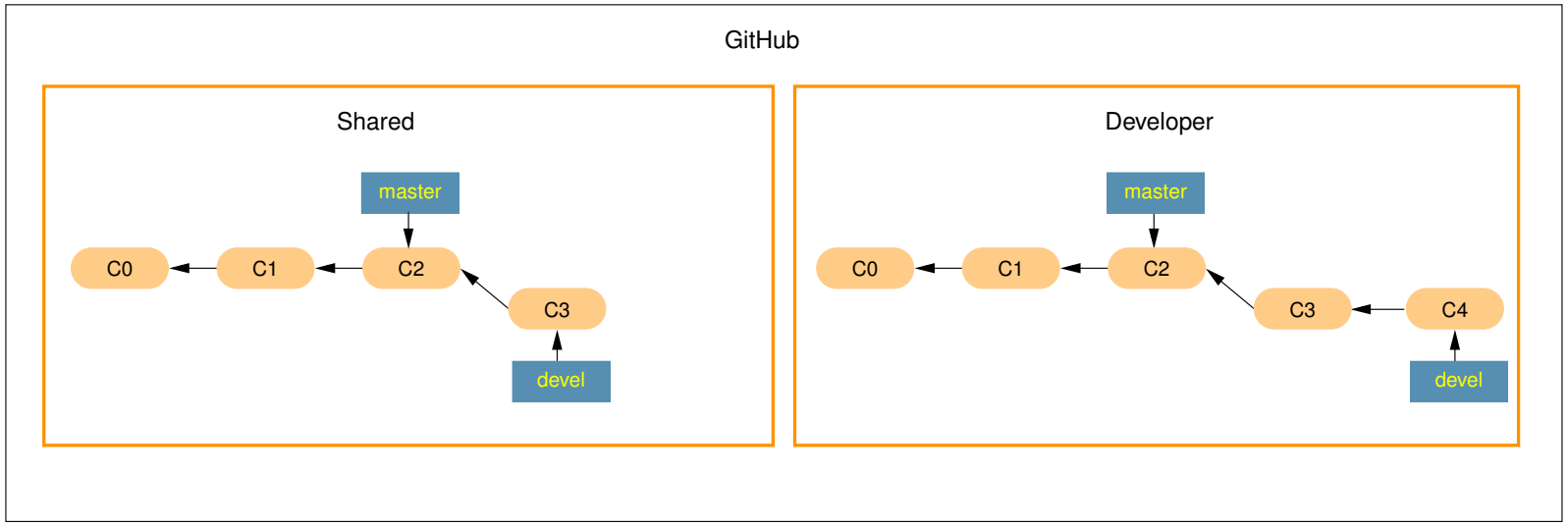


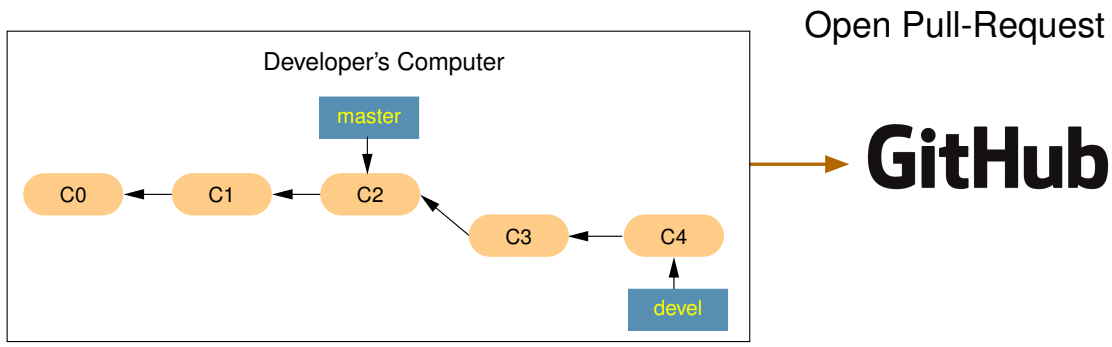


Pull-Request Steps

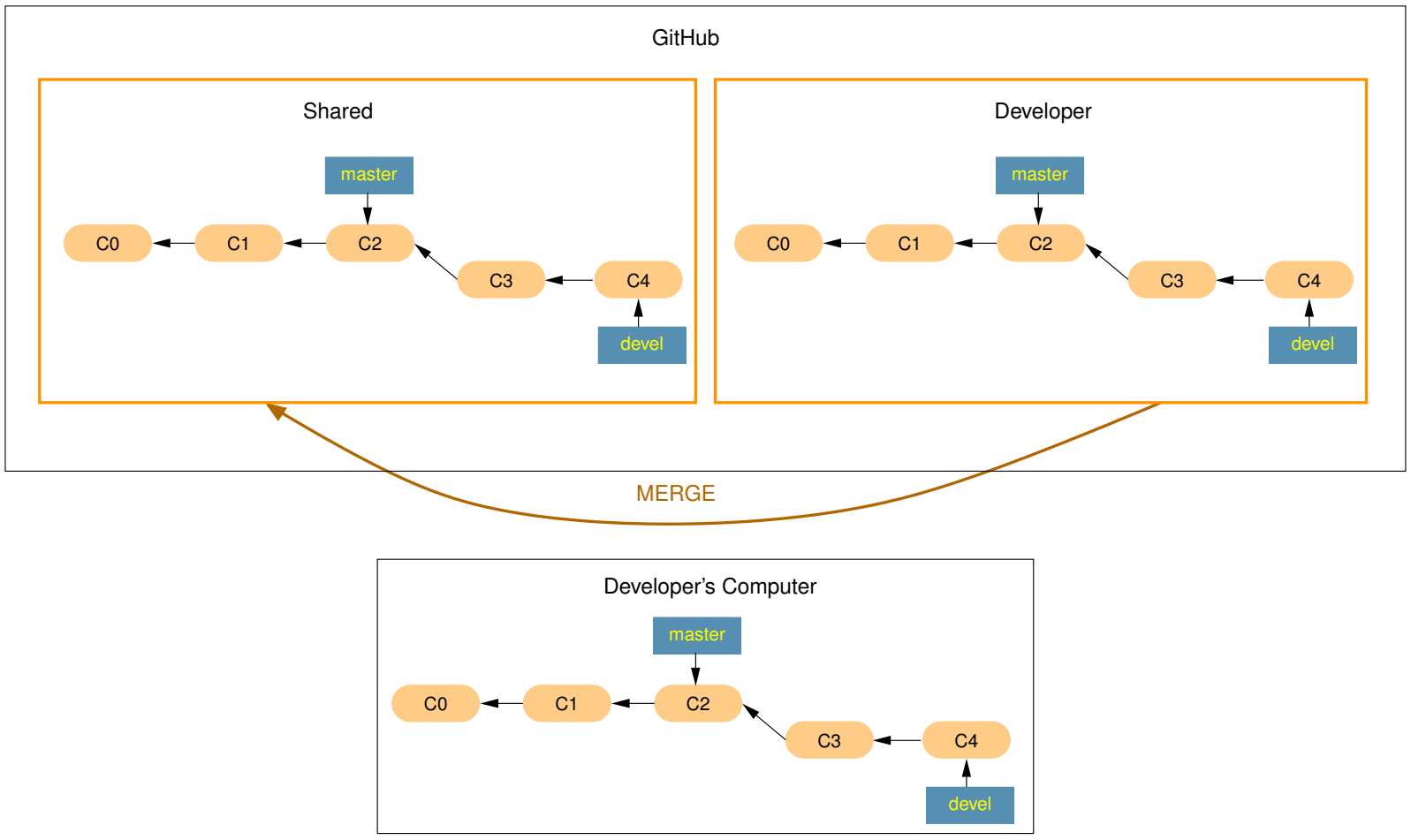


Pull-Request Steps

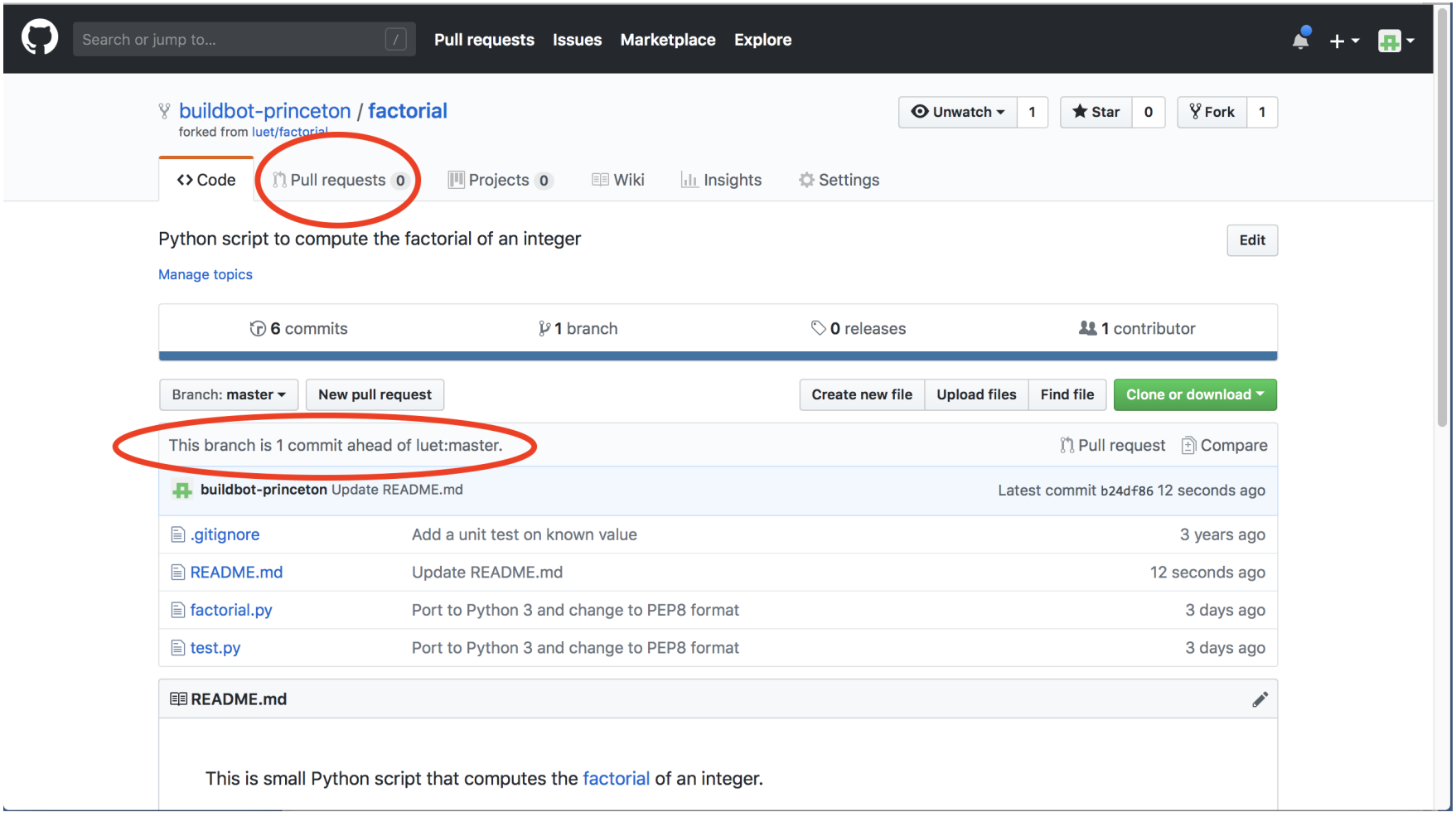




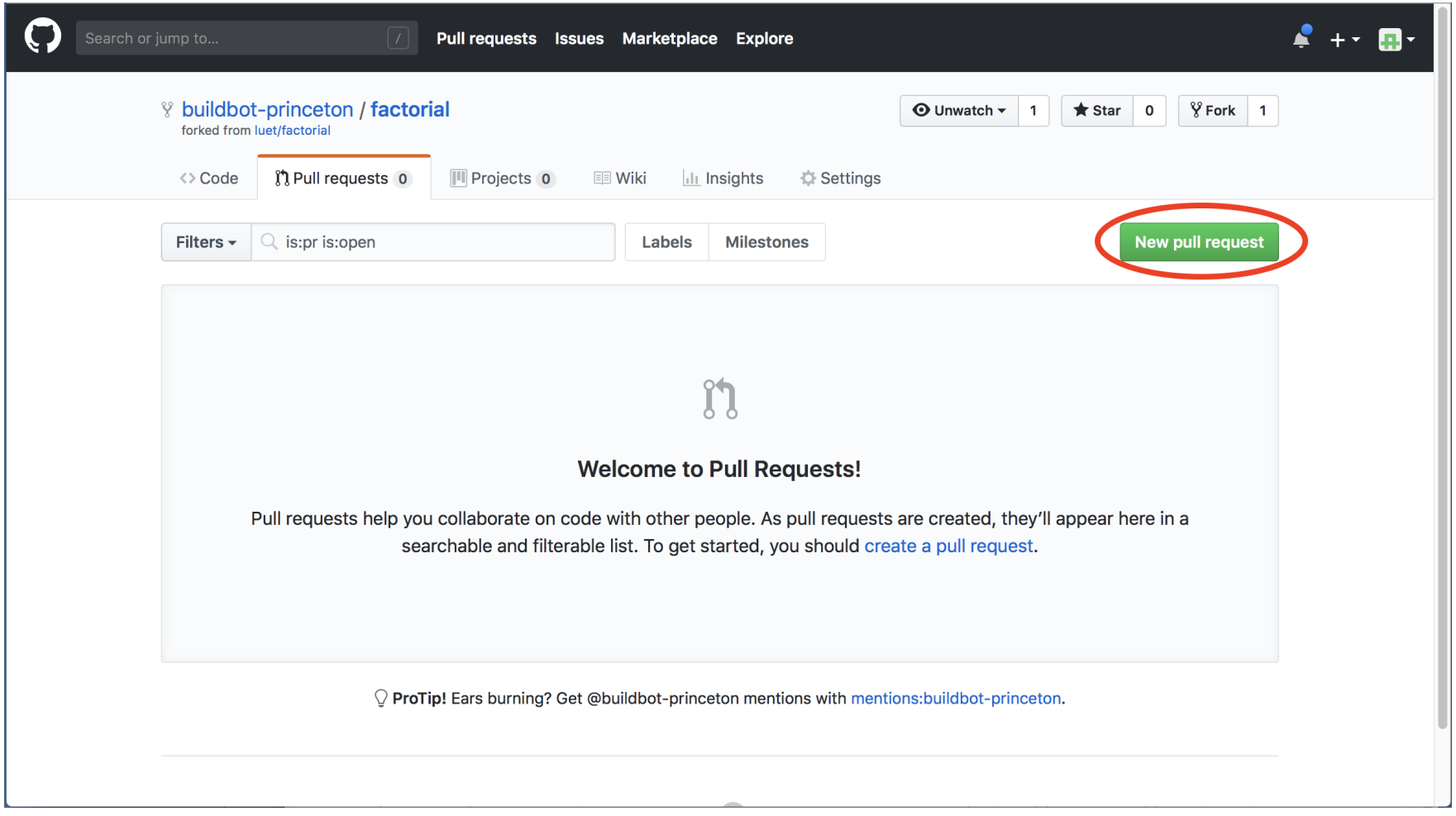
Pull-Request Steps



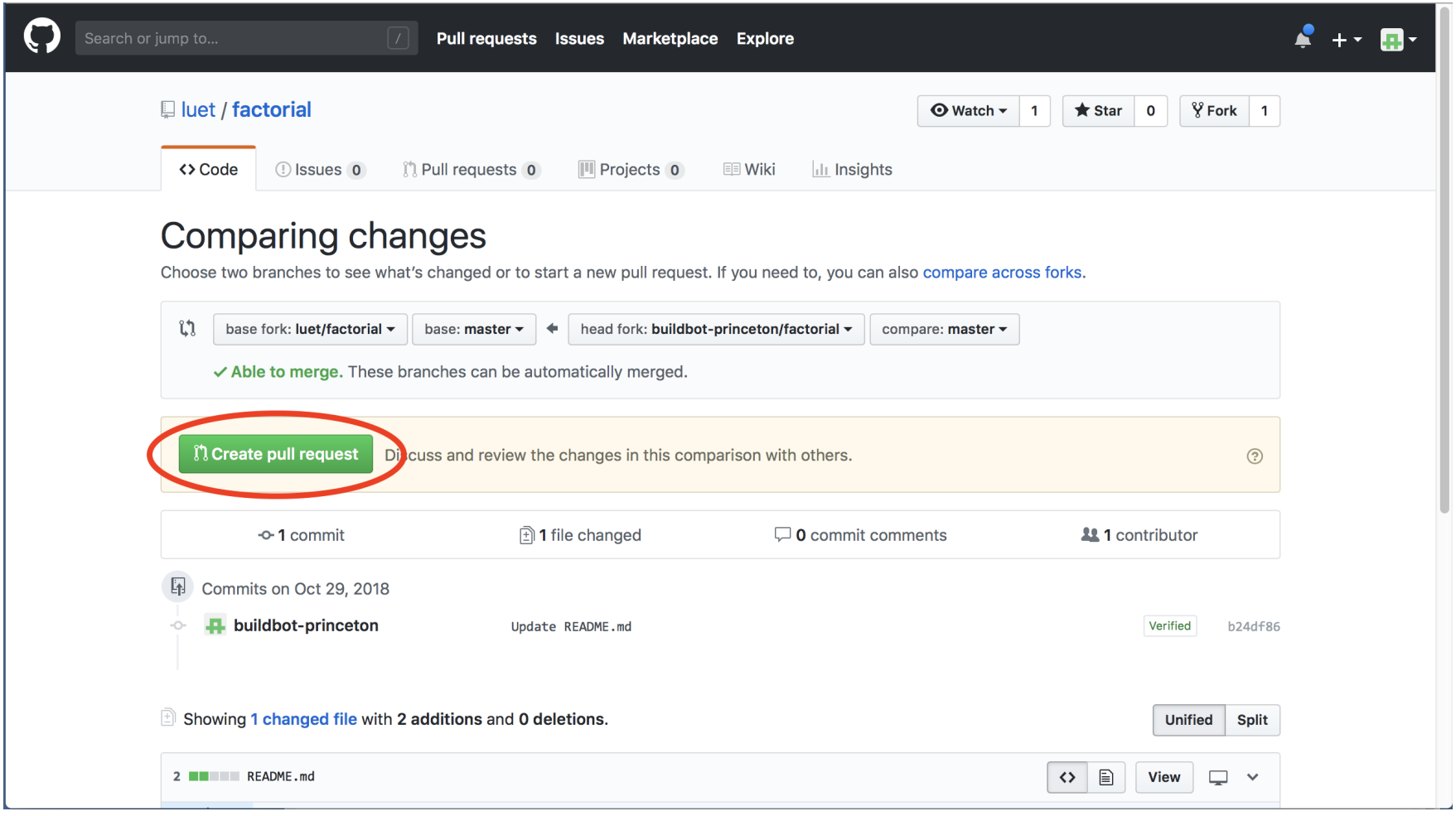
Pull-Request Steps on GitHub



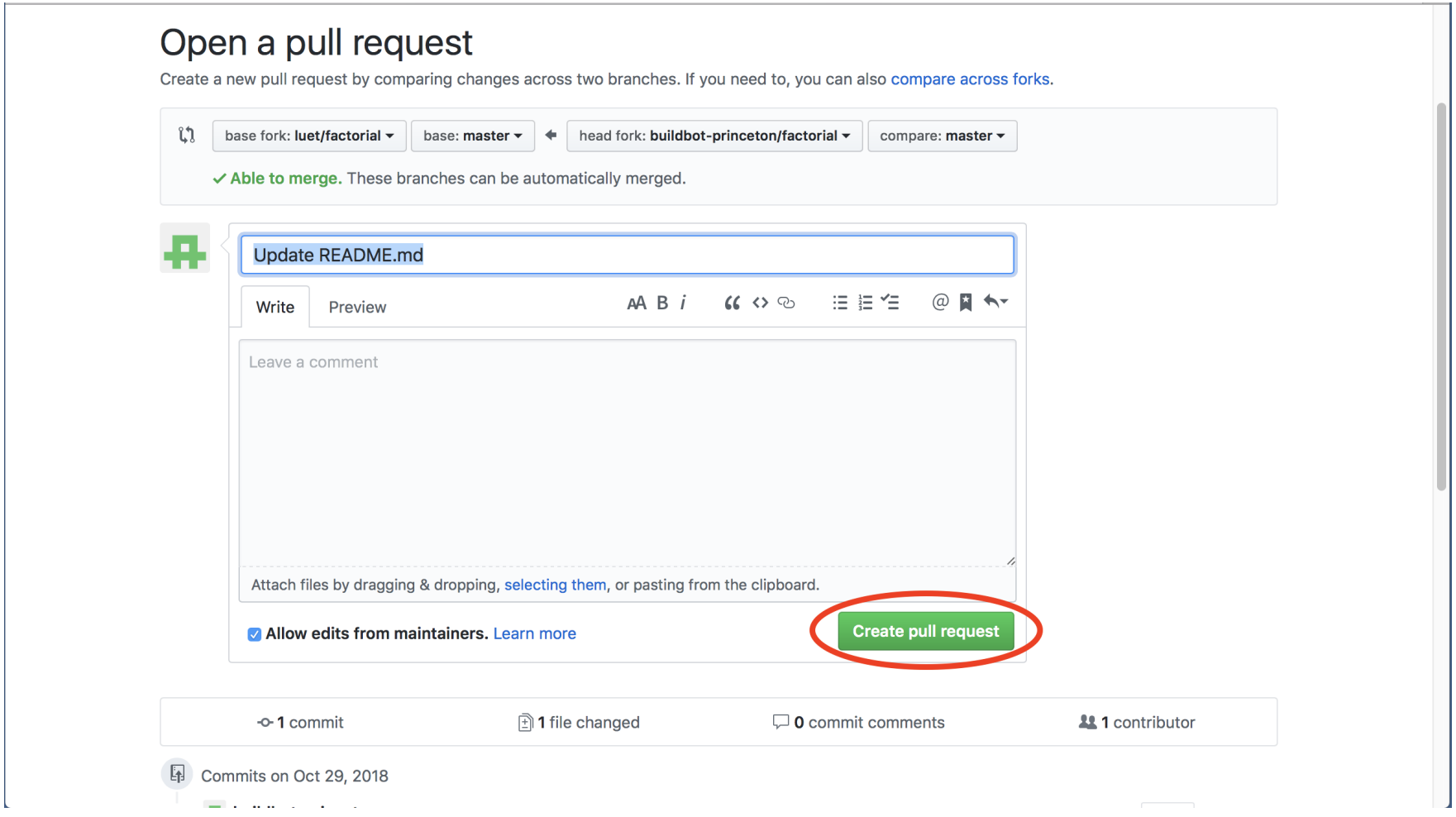
Pull-Request Steps on GitHub



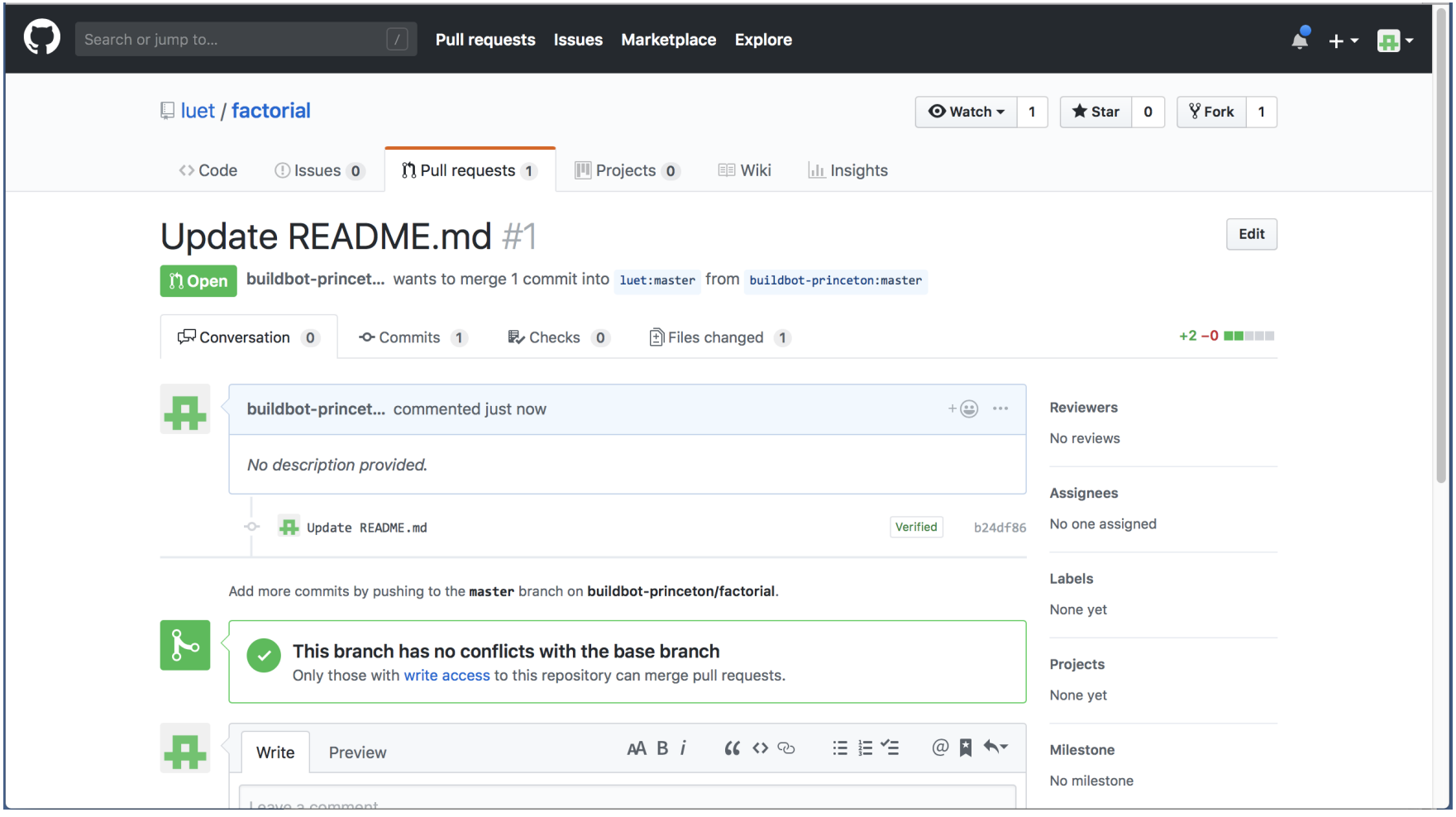
Pull-Request Steps on GitHub



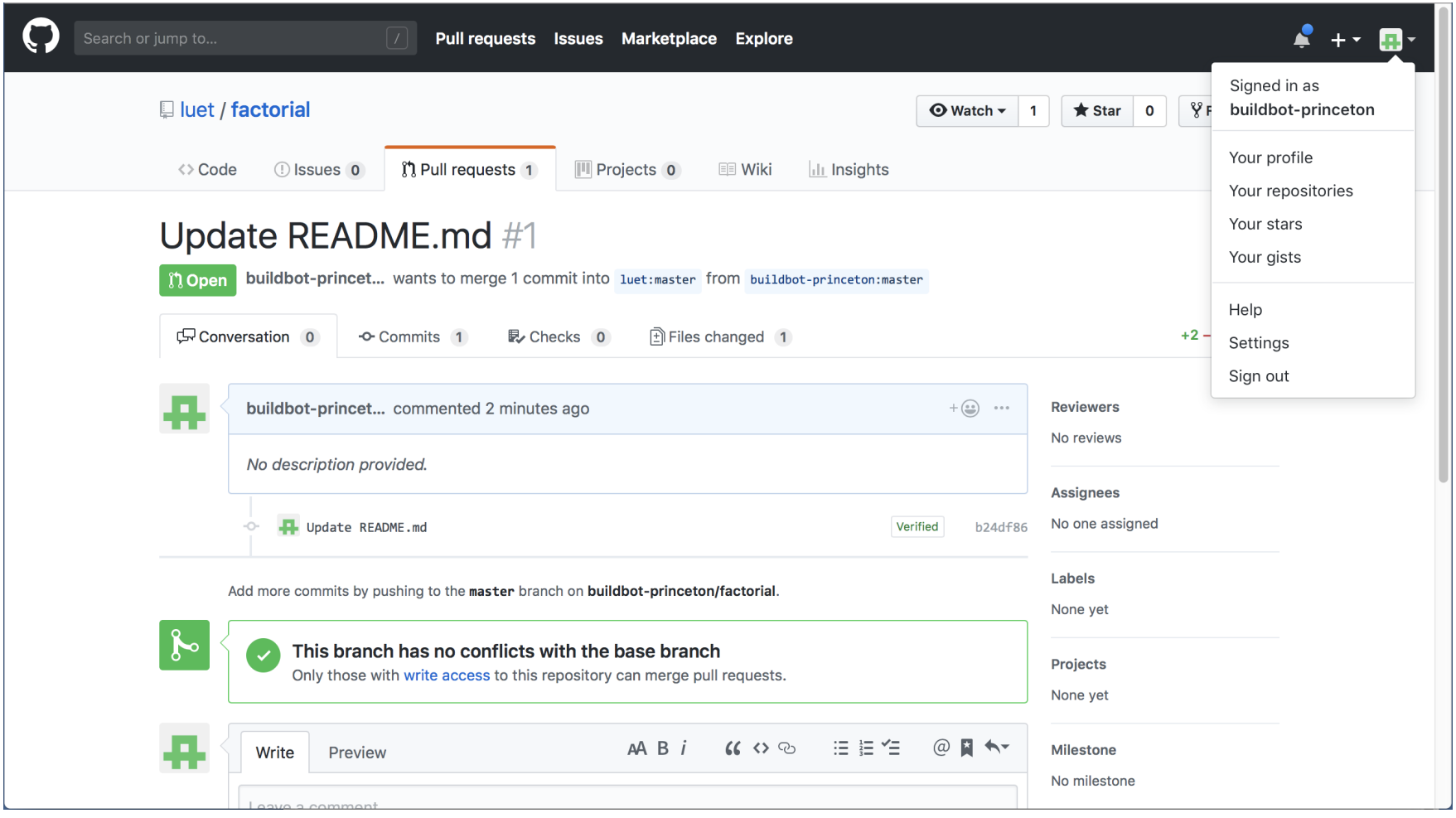
Pull-Request Steps on GitHub



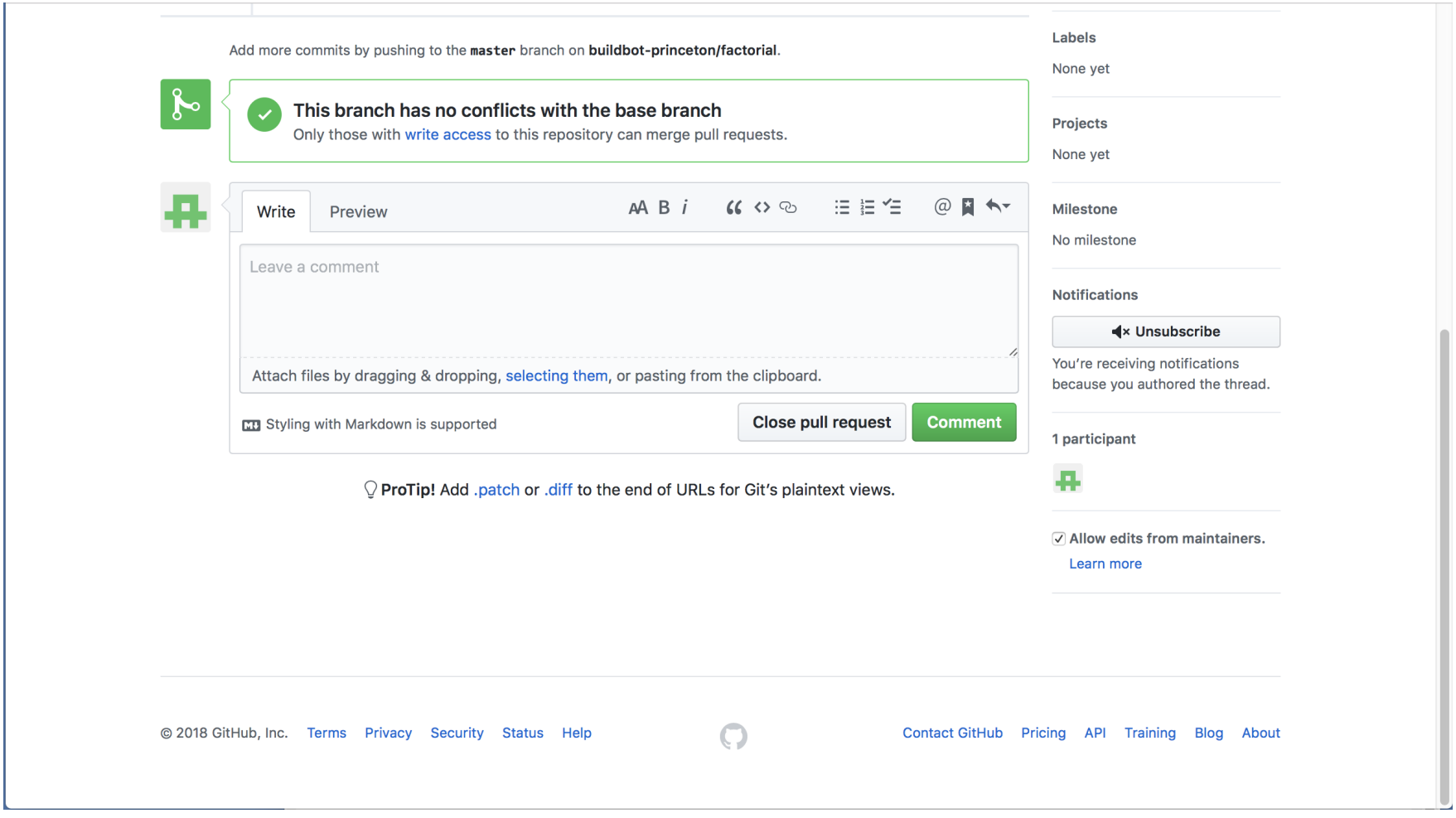
Pull-Request Steps on GitHub



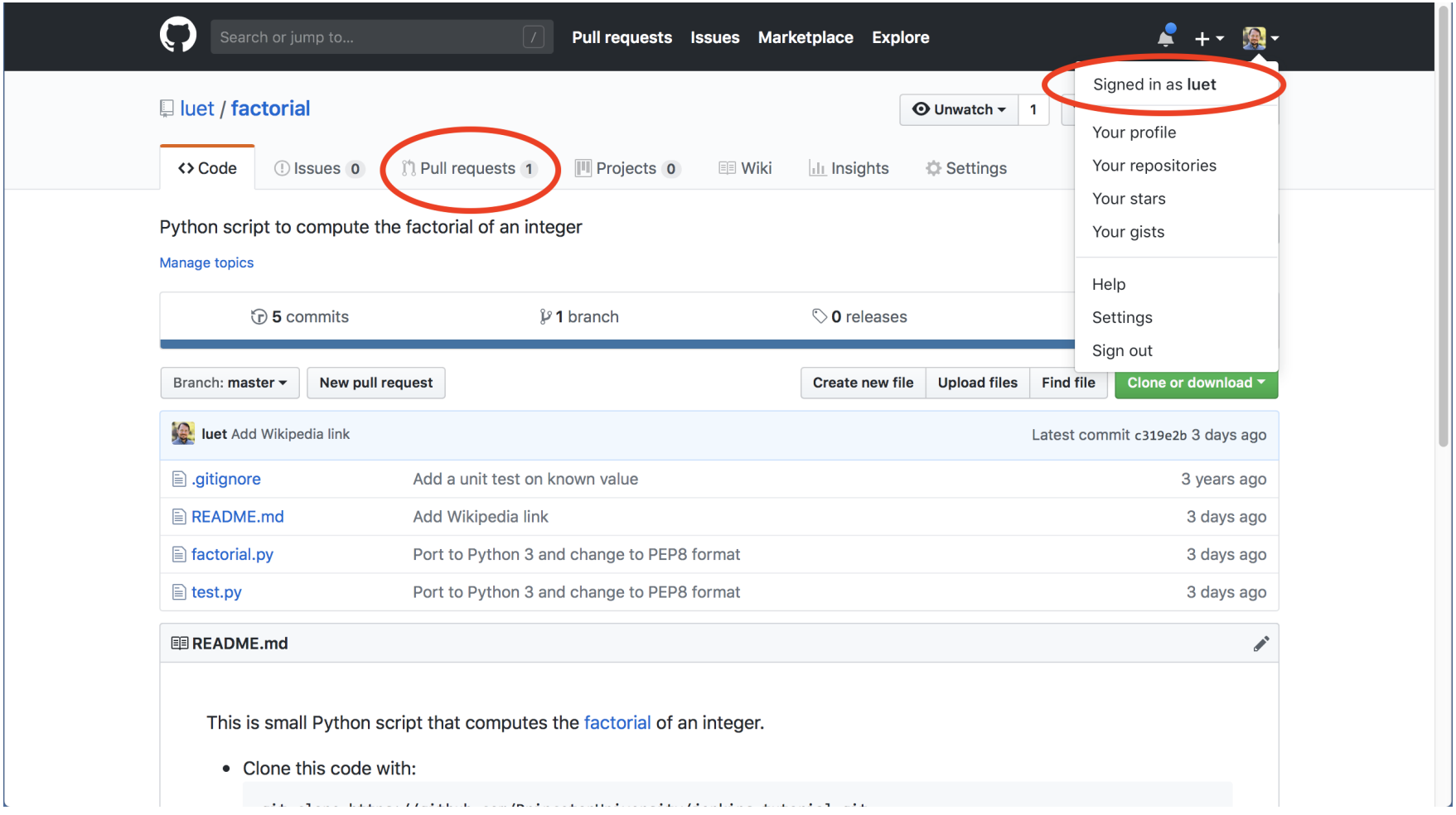
Pull-Request Steps on GitHub



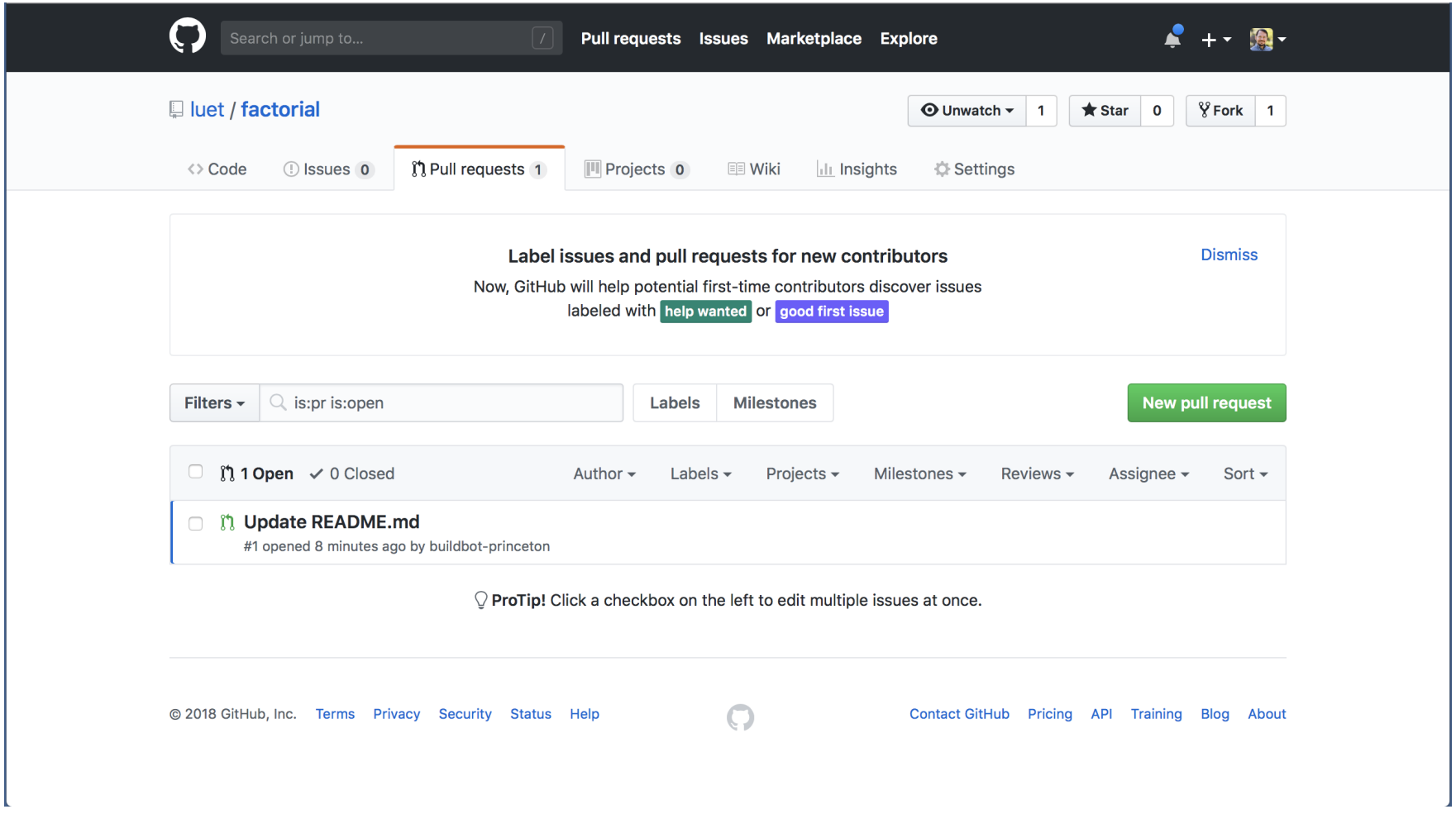
Pull-Request Steps on GitHub



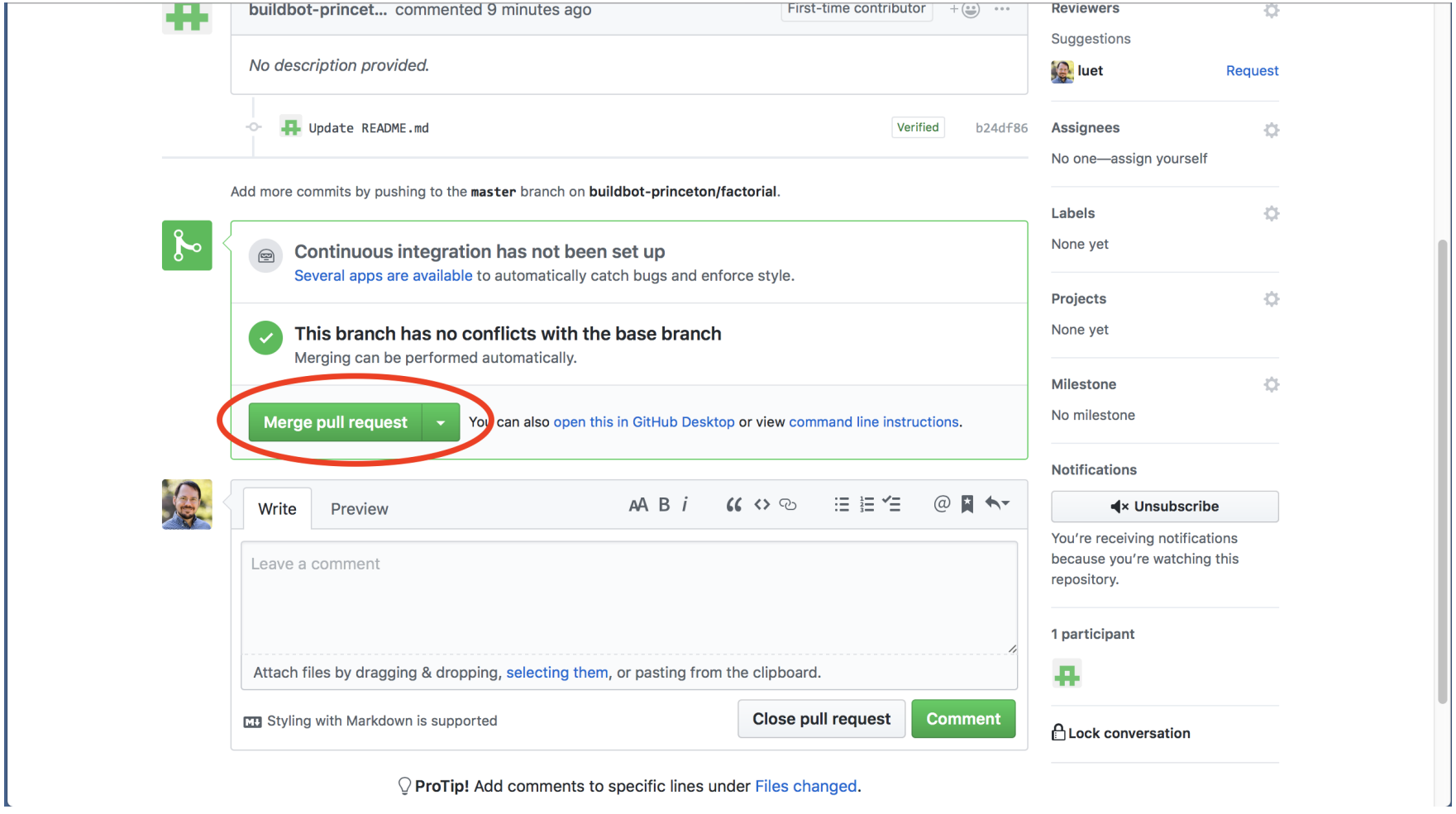
Pull-Request Steps on GitHub



Pull-Request Steps on GitHub



Pull-Request Steps on GitHub



hands-on #1

http://luet.princeton.edu/cicd/

Advantages of Doing a Pull-Request?

- Gives us time to review and test the changes before committing them.
- So that no broken code gets committed to the shared repository.
- The problem with this simple workflow is that it can be hard for the code maintainers to know whether or not changes break the code.
- That's why we need to build some tests.

Outline

Git and GitHub for Collaborative Developments

Testing

Automatic Testing

Other Useful Tools For Collaborative Software Development

References and Getting Help

Conclusion

Definition and Motivation

- Debugging is what you do when you know that a program is broken.
- Testing is a determined, systematic attempt to break a program that you think is working.
- Testing for Quality Assurance: make sure some changes didn't change the results compared to the last version.
- When you write code with testing in mind, you write better code because you write better interfaces.

When to Write the Tests

- Test while you are writing the code.
- Test incrementally:
  - write part of a program,
  - test it,
  - add some more code,
  - test that,
  - and so on.
- Some programming techniques (e.g. Extreme Programming) even instruct you to write the tests first.

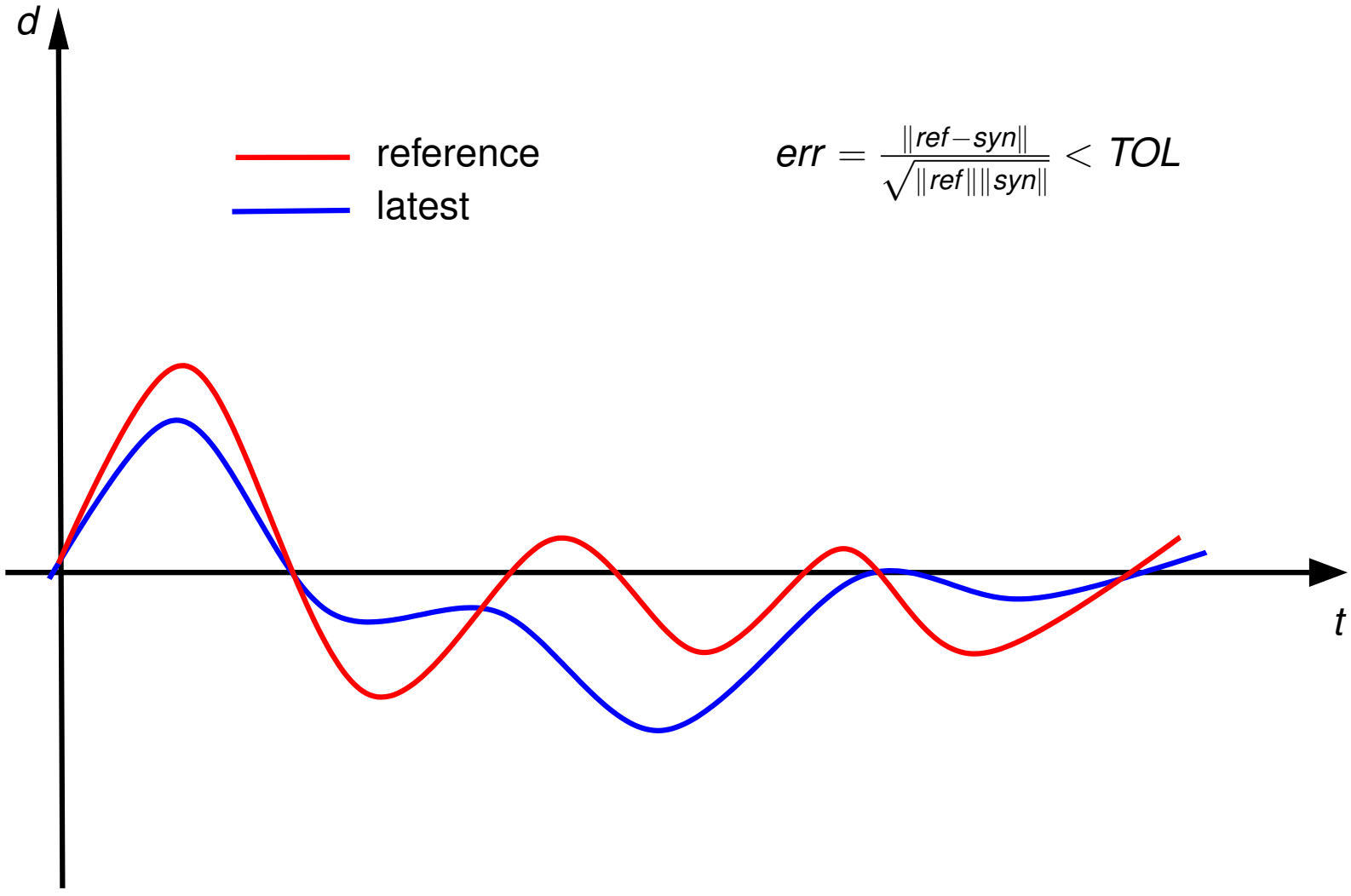
Testing for Functionality

- Unit testing: test one function.
- Test a set of functions or the entire code:
  - It can be hard to design a test that will exercise a certain portion of your code by running the entire code.
  - Use libraries and drivers to isolate functions or a group of functions.

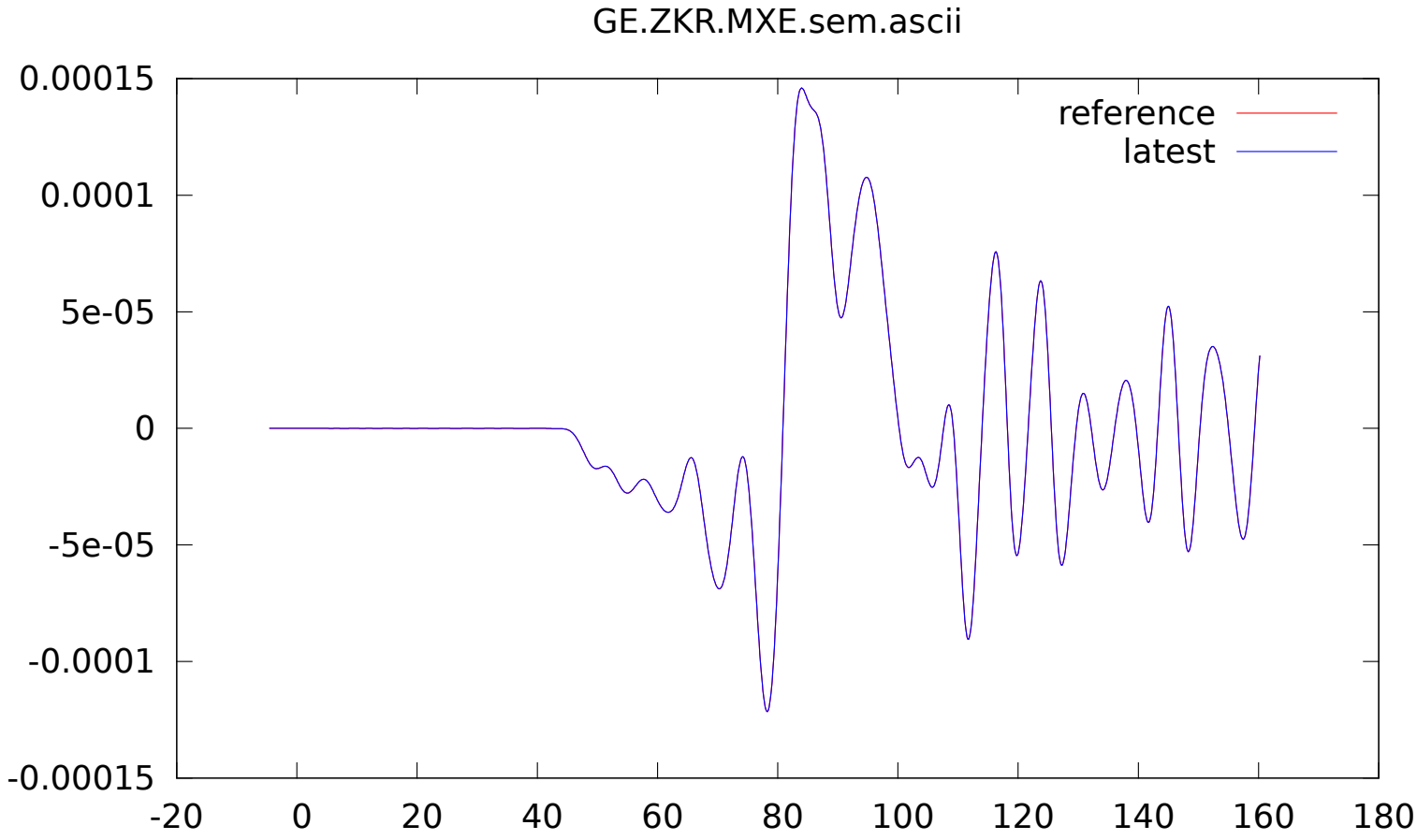
Regression Testing: An Example

- SPECFEM3D_GLOBE (Tromp et. al): simulates global and regional seismic wave propagation.
- This code produces seismograms, which are records of the ground motion in one direction at a measuring station as a function of time.

Regression Testing: Comparing Seismograms



An Actual Seismograms



Other Tests

- Use different compilers to:
  - check that it will compile.
  - find programming mistakes.
  - compare the results.
- Use different versions of scripting languages e.g. Python, Matlab.
- Run on different OS, hardware to make sure:
  - the code runs.
  - the code gives the same results.

Testing frameworks

- Google framework for C++: Google Test.
- Python: unittest
- Matlab

Testing Frameworks Example

- Look at my factorial calculation repository:
  https://github.com/luet/factorial/

Outline

Git and GitHub for Collaborative Developments

Testing

Automatic Testing

Other Useful Tools For Collaborative Software Development

References and Getting Help

Conclusion

Why Run Test Automatically?

- The temptation when you develop code is to test only that part that you just wrote.
  - But there might be side effects to your changes.
  - So you want to run a your entire suite of tests every time you make a change.
  - you are less likely to do that if the tests have to be run manually.
- Not all the developers have access to all tools.
- Once it's set up you don't have to spend any time running your tests.

Test Automation with Travis

- An example: https://github.com/uvaaland/travis_tutorial
- You can get a free account at https://travis-ci.com/.
- When you can login with your GitHub credential.
- It's only really free for open source (public repositories).

Test Automation with Jenkins

- Service offered by Research Computing.
- Jenkins is a web-based application for automatic testing.
- Simple user interface: easy to configure.
- The advantage other Travis is that with Jenkins you have access to the Research Computing resources:
  - Large number of cores.
  - Compilers.
  - Licensed software e.g. Matlab.
- Email `cses@princeton.edu` to request an account.
- There is a tutorial at:
  `http://jenkins-doc.princeton.edu/tutorial.html`

A Workflow with Jenkins and GitHub

Typical workflow:

1. A Pull-Request is open on GitHub.
2. GitHub sends a signal to our Jenkins server (webhook).
3. Jenkins runs the tests suite.
4. Jenkins reports the results of the tests on the GitHub web site.
   - If the changes passed the test, the code maintainer can merge the changes.
   - If the changes failed the test, the developer needs to solve the problem and push the changes to Github.

Scheduled Tests

- A Pull-Request only triggers short (< 15 min) tests.
- We use Jenkins to schedule longer tests:
  - daily (< 1 hour).
  - weekly (> 1 hour).

Jenkins vs. Travis

- With Jenkins you can run on:
  - the Research Computing clusters.
  - any machine that you have `ssh` access to.
- Travis is good for small scripts, not parallel code.

Outline

Git and GitHub for Collaborative Developments

Testing

Automatic Testing

Other Useful Tools For Collaborative Software Development

References and Getting Help

Conclusion

Code Documentation

- It's important to document your code when someone else will have to read it.
  - especially when this someone else can be you in a couple years.
- Doxygen:
  - Documentation is in the code.
  - Supports adding Latex to the documentation.
  - Build calling graph.
  - *show example*
- You can use Sphinx with Python.

Documenting outside the code: GitHub Wiki

- GitHub Wiki
  - It's easy to write in Markdown.
  - https://help.github.com/articles/about-github-wikis/
  - *show examples*
- Issues:
  - https://guides.github.com/features/issues/
  - Like a shared TODO list.
- Gists:
  - https://gist.github.com/
  - For sharing small codes.

Outline

Git and GitHub for Collaborative Developments

Testing

Automatic Testing

Other Useful Tools For Collaborative Software Development

References and Getting Help

Conclusion

References

- The Practice of Programming, by Brian W. Kernighan and Rob Pike.
- Testing with Python:
  - The Hitchhiker's Guide to Python!: `http://docs.python-guide.org/en/latest/`
  - Testing your code: `http://docs.python-guide.org/en/latest/writing/tests/#testing-your-code`
- Agile development: Manifesto for Agile Software Development

Outline

Git and GitHub for Collaborative Developments

Testing

Automatic Testing

Other Useful Tools For Collaborative Software Development

References and Getting Help

Conclusion

Conclusion

- I encourage you to, in order of urgency:
  - use a Version Control System.
  - design some tests.
  - run those tests automatically.
- In the long run, it will:
  - save you some time in debugging and troubleshooting.
  - let you modify your code with confidence that you are not breaking it.
  - generate a better organized and better written code.
- We are here to help.
  - You can e-mail us at: cses@princeton.edu.
  - Come to the help sessions Tuesdays (10-11 am) and Thursdays (2-3 pm), room 347 Lewis Library.
- Job opportunity for Graduate students.