Continuous Integration
and
Automated Software Testing

David Luet

Research Software Engineering for Computational & Data Science
Research Computing, OIT

March 10, 2023
Outline

Git and GitHub Workflow for Collaborative Developments

Testing

Automatic Testing

References and Getting Help

Conclusion

Hands-on exercise with GitHub Actions with self-hosted runner
Outline

Git and GitHub Workflow for Collaborative Developments

Testing

Automatic Testing

References and Getting Help

Conclusion

Hands-on exercise with GitHub Actions with self-hosted runner
A Simple Collaborative Workflow
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 generate problems.
- This does not work well with more than two developers.
People can get ...
Introducing Different Roles

To solve this issue, we introduce two roles with different permissions on the shared repository:

▶ code maintainers: push and pull (read/write) permissions.
▶ developers: only pull (read only) permission.
The developers need their own remote repo or branch

Since the developers cannot push to the shared repository they need either:
► their own branch in the shared repo:
  ► small team.
  ► See Managing a branch protection rule.
► their own fork of the repo: larger team.
Forking a Repository on GitHub
Forking a Repository on GitHub

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

Python script to compute the factorial of an integer

- 5 commits
- 1 branch
- 0 releases
- 1 contributor

Branch: master

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Author</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>.gitignore</td>
<td>Add a unit test on known value</td>
<td>luet</td>
<td>3 years ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Add Wikipedia link</td>
<td>luet</td>
<td>3 days ago</td>
</tr>
<tr>
<td>factorial.py</td>
<td>Port to Python 3 and change to PEP8 format</td>
<td>luet</td>
<td>3 days ago</td>
</tr>
<tr>
<td>test.py</td>
<td>Port to Python 3 and change to PEP8 format</td>
<td>luet</td>
<td>3 days ago</td>
</tr>
</tbody>
</table>

Readme.md

This is small Python script that computes the **factorial** of an integer.

- Clone this code with:
  
git clone https://github.com/PrincetonUniversity/jenkins_tutorial.git
Pull-Request: the Different Repositories
Pull-Request Steps

GitHub

Shared

Developer

Developer's Computer
Pull-Request Steps

GitHub

Shared

C0 → C1 → C2 → C3 →
main

dev

Developer

C0 → C1 → C2 → C3 → C4 →
main

dev

Developer's Computer

C0 → C1 → C2 → C3 → C4 →
main

dev

PUSH
Pull-Request Steps

GitHub

Shared

Developer

Developer's Computer

Open Pull-Request
Pull-Request Steps

GitHub

Shared

Developer

Developer's Computer

MERGE
Advantages of Doing a Pull-Request?

- Gives us time to review the changes before committing them.
- So that hopefully no broken code gets committed to the shared repository.
Because honestly . . .
We need tests

- 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 Workflow for Collaborative Developments

Testing

Automatic Testing

References and Getting Help

Conclusion

Hands-on exercise with GitHub Actions with self-hosted runner
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**.
- Good tests provide a **documentation** for the code.
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.
- The entire code
- Test a set of functions:
  - 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

\[
err = \frac{\|\text{ref} - \text{syn}\|}{\sqrt{\|\text{ref}\| \|\text{syn}\|}} < TOL
\]

- reference
- latest

![Graph showing comparison between reference and latest seismograms](image)
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
Example

- GitHub repo: https://github.com/luet/factorial/
- Simple Python code for computation of $n!$ recursively.
import sys

def recur_factorial(n):
    """Function to return the factorial of a number using recursion""
    if n == 1:
        return n
    else:
        return n*recur_factorial(n-1)

if __name__ == '__main__':
    arg = sys.argv
    print recur_factorial(int(arg[1]))
import sys

def recur_factorial(n):
    """Function to return the factorial of a number using recursion""
    if n == 1:
        return n
    else:
        return n*recur_factorial(n-1)

if __name__ == '__main__':
    arg = sys.argv
    print recur_factorial(int(arg[1]))

$ python factorial.py 4
24
import unittest
from factorial import recur_factorial

class TestReturnValue(unittest.TestCase):
    def test_value(self):
        
        """
        Test for a known return value
        """

        self.assertEqual(recur_factorial(3), 6)

if __name__ == '__main__':
    unittest.main()
import unittest
from factorial import recur_factorial

class TestReturnValue(unittest.TestCase):
    def test_value(self):
        """
        Test for a known return value
        """
        self.assertEqual(recur_factorial(3), 6)

if __name__ == '__main__':
    unittest.main()

$ python test.py
.
---------------------------------------
Ran 1 test in 0.000s
OK
Outline

Git and GitHub Workflow for Collaborative Developments

Testing

Automatic Testing

References and Getting Help

Conclusion

Hands-on exercise with GitHub Actions with self-hosted runner
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.
Automation Servers

▶ Travis:
  ▶ good for small scripts, not parallel code.
  ▶ We are having issues with the PrincetonUniversity GitHub organization since they changed their pricing model.

▶ Buildbot.

▶ GitLab CI/CD.

▶ Bamboo (Atlassian).

▶ GitHub Actions:
  ▶ University has 50,000 min/month to run on GitHub’s servers.
  ▶ You have unlimited time when running on self-hosted runner (adroit).

▶ With Jenkins you can run on:
  ▶ the Research Computing clusters.
  ▶ any machine that you have ssh access to.
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.
You provide the machine where your code is going to be run.

This can be any machine you have remote access to:

- clusters (SLURM).
- desktop machine (mac mini).
- VM.

Advantages:

- Large number of cores.
- Compilers.
- Licensed software e.g. Matlab.
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).
Some Currently Used Features

- There are many events that can trigger a test e.g.:
  - pull-request.
  - push.
  - scheduled.
- Jenkins can send emails outside of Princeton.
  - Only the code maintainers need an account on Jenkins.
- Slack notifications.
- Jenkins provides status badges.
- It is not just for testing:
  - Automation server.
  - Jenkins Pipeline for Continuous delivery.
Getting Started

- Step by step tutorial (30 min):
  https://github.com/PrincetonUniversity/jenkins-getting-started
Outline

Git and GitHub Workflow for Collaborative Developments

Testing

Automatic Testing

References and Getting Help

Conclusion

Hands-on exercise with GitHub Actions with self-hosted runner
References

- The Practice of Programming, by Brian W. Kernighan and Rob Pike.
- Testing with Python:
We are here to help

- You can e-mail cses@princeton.edu.
- Come to the help sessions Tuesdays (10:30-11:30 am) and Thursdays (2-3 pm).
  https://researchcomputing.princeton.edu/support
- Job opportunity for Graduate students.
Outline

Git and GitHub Workflow for Collaborative Developments

Testing

Automatic Testing

References and Getting Help

Conclusion

Hands-on exercise with GitHub Actions with self-hosted runner
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.
  - generate a better organized and better written code.
Testing new idea with a sense of freedom

Let you modify your code with confidence that you are not breaking it.

Thank you Robin Higgins:  
https://pixabay.com/users/robinhiggins-1321953/
Survey

- Please fill out this survey before leaving:
- I am especially interested in your feedback on the hands-on exercise.
Outline

Git and GitHub Workflow for Collaborative Developments

Testing

Automatic Testing

References and Getting Help

Conclusion

Hands-on exercise with GitHub Actions with self-hosted runner
GitHub Actions: running

1. push

2. trigger
   - run the workflow
   - report to github

3. report

laptop
modify the code

github.com
GitHub Actions: configuration

github.com
- setup actions:
- enable Actions on the repo.
- set up the self-hosted runner.
- configure your Actions: .github/workflows

connect

adroit
- install runner software.
- run within tmux session

laptop
set up a private fork of my repo
https://github.com/PrincetonUniversity/github_actions_tutorial