Including mutation testing as part of a continuous integration workflow

Tim Waterson

What is mutation testing?

If our application code breaks, will our tests detect that?

```
def flip(char):
  codepoint = ord(char)

  if 64 < codepoint < 91:
    return chr(codepoint + 32)

  if 96 < codepoint < 123:
    return chr(codepoint - 32)

  return char
```

- 'a' $\rightarrow$ 'A'
- 'z' $\rightarrow$ 'Z'
- '1' $\rightarrow$ '1'
- 'Z' $\rightarrow$ 'z'
- 'A' $\rightarrow$ 'a'
- '+' $\rightarrow$ '+'

ROR Relational Operator Replacement
This mutation operator swaps relational operators. Substitution rules implemented by this operator (**Table 4**) refer to similar relational operators (eg. `<' vs `<=`) or opposite ones (eg. `<` vs `>`).

Table 4. Substitution rules of relational operators

Operator before mutation	<	<	>	>	<=	<=	>=	>=	==	!=
Operator after mutation	>	<=	<	>=	>=	<	<=	>	!=	==

Examples before mutation:

```
if x < y:
while i >= 10:
```

after mutation:

```
if x > y:
while i > 10:
```

Figure 1: The ROR mutation operator. Image is an extract from *Operators for Mutation Testing of Python Programs*, a 2014 paper by Derezinska and Halas.

```
def flip(char):
  codepoint = ord(char)

  if 64 < codepoint <= 91:
    return chr(codepoint + 32)

  if 96 < codepoint < 123:
    return chr(codepoint - 32)

  return char
```

All our tests still pass. . .

- `'a'` $\rightarrow$ `'A'`
- `'z'` $\rightarrow$ `'Z'`
- `'1'` $\rightarrow$ `'1'`
- `'Z'` $\rightarrow$ `'z'`
- `'A'` $\rightarrow$ `'a'`
- `'+'` $\rightarrow$ `'+'`

. . . however we now have an erroneous result.

- `'['` $\rightarrow$ `'{'`

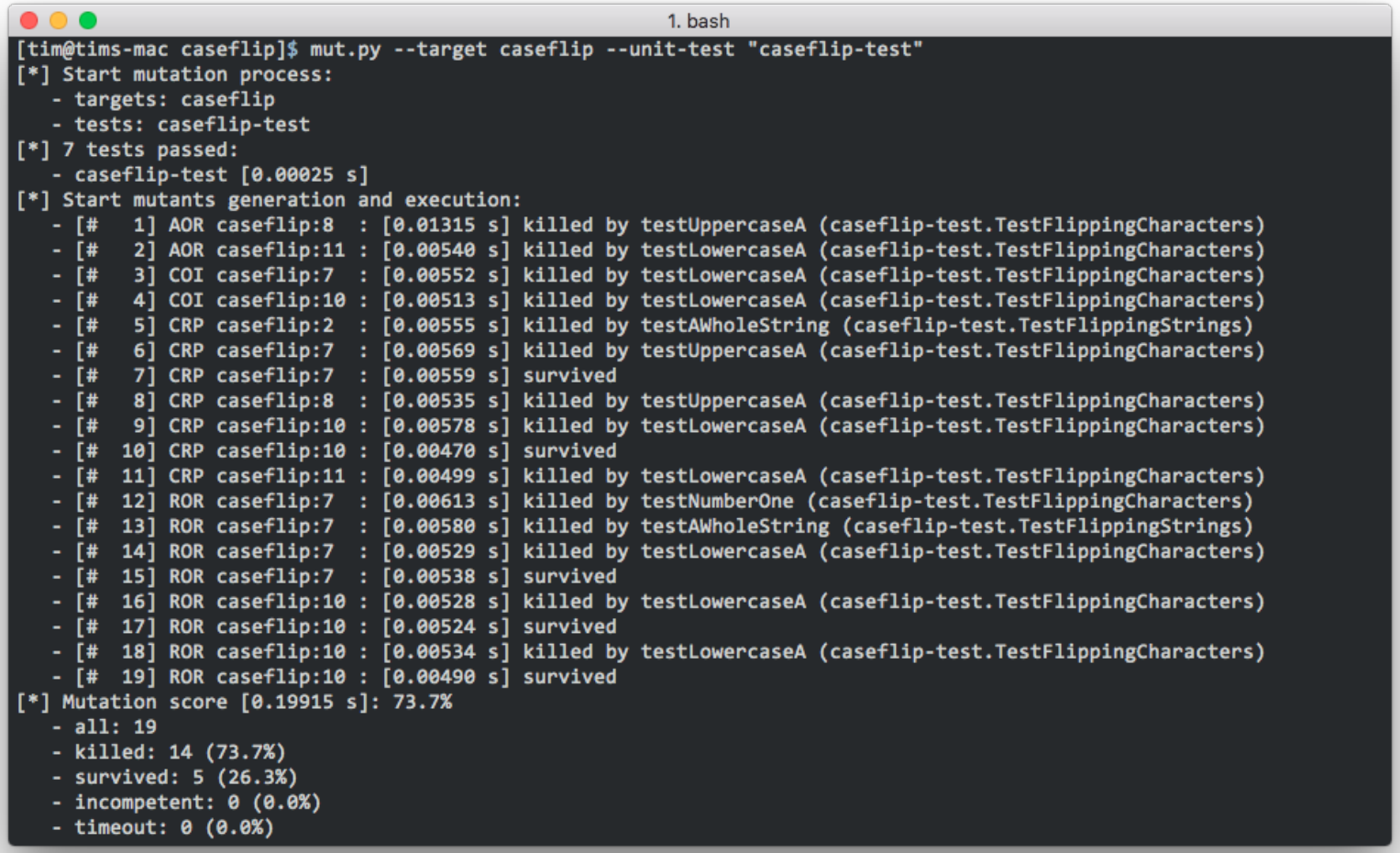

Figure 2: Mutation testing output from MutPy

What is continuous integration?

Every time the code changes, build the system and check its quality.

Figure 3: One way to show the CI build status

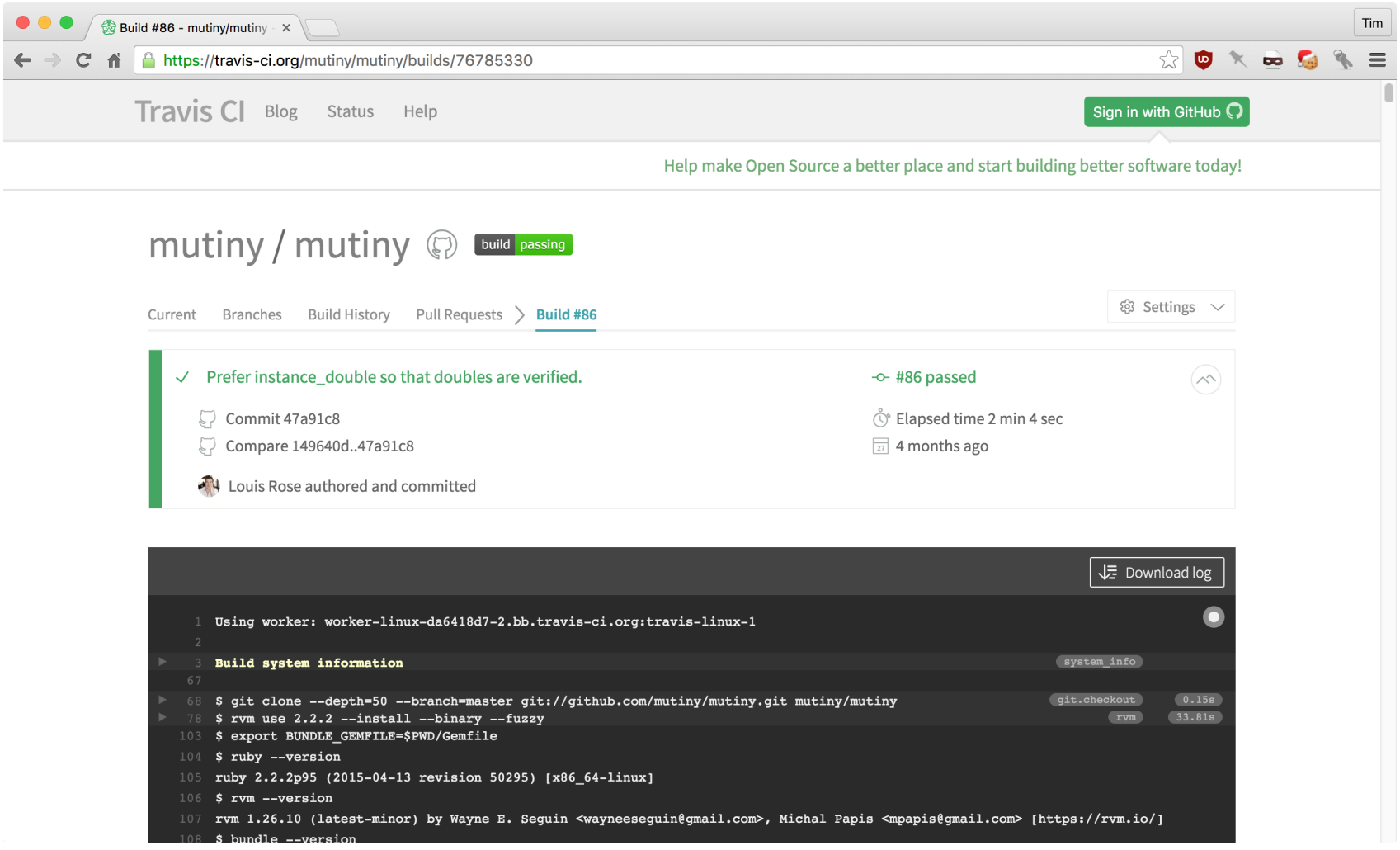

Figure 4: Travis CI in action on a GitHub project

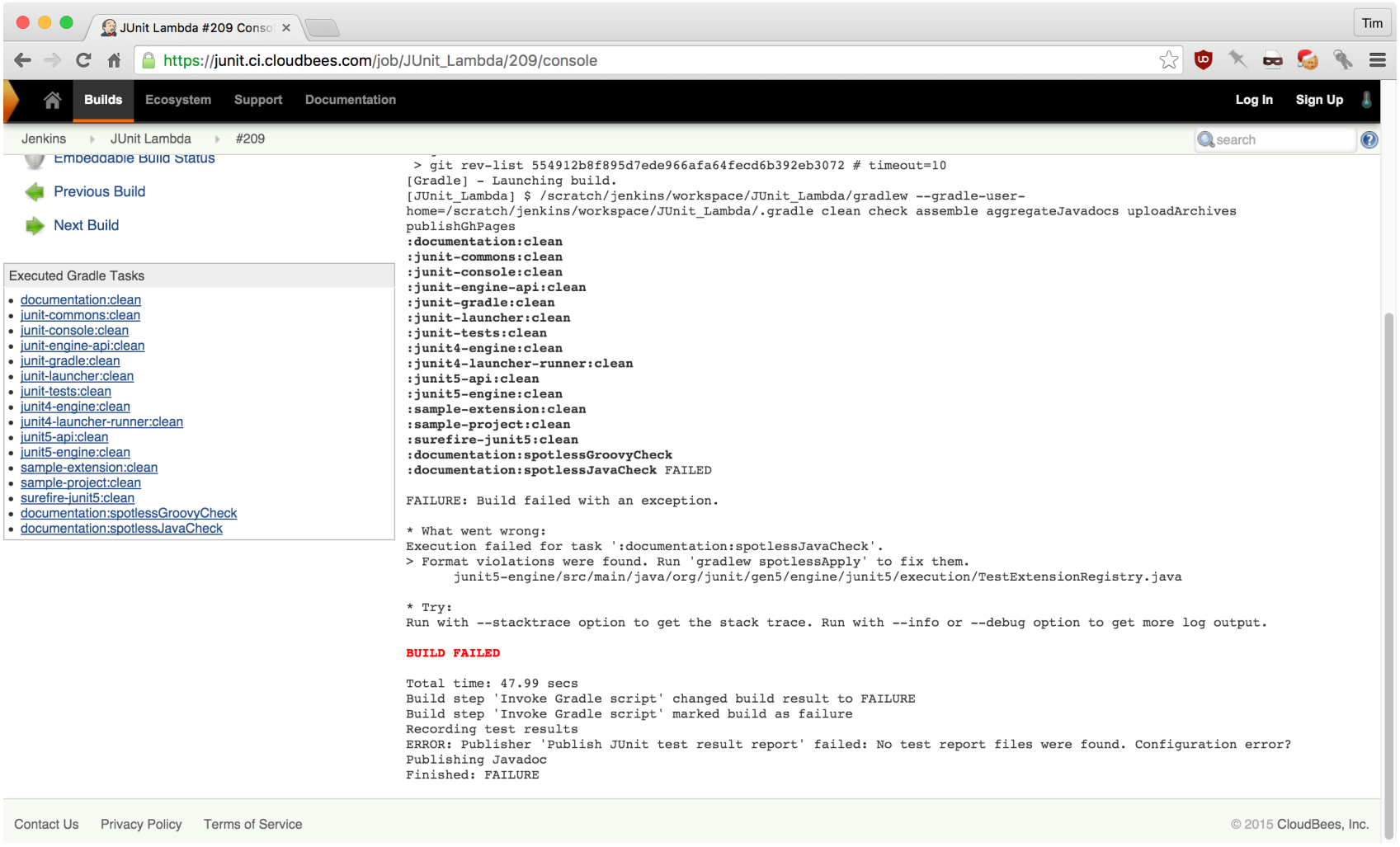

Figure 5: A failed build in the Jenkins console

What are the benefits arising from bringing mutation testing and continuous integration together?

What are the challenges in running mutation testing in a continuous integration environment?

Speed

Reporting

What's my project doing to address those challenges?

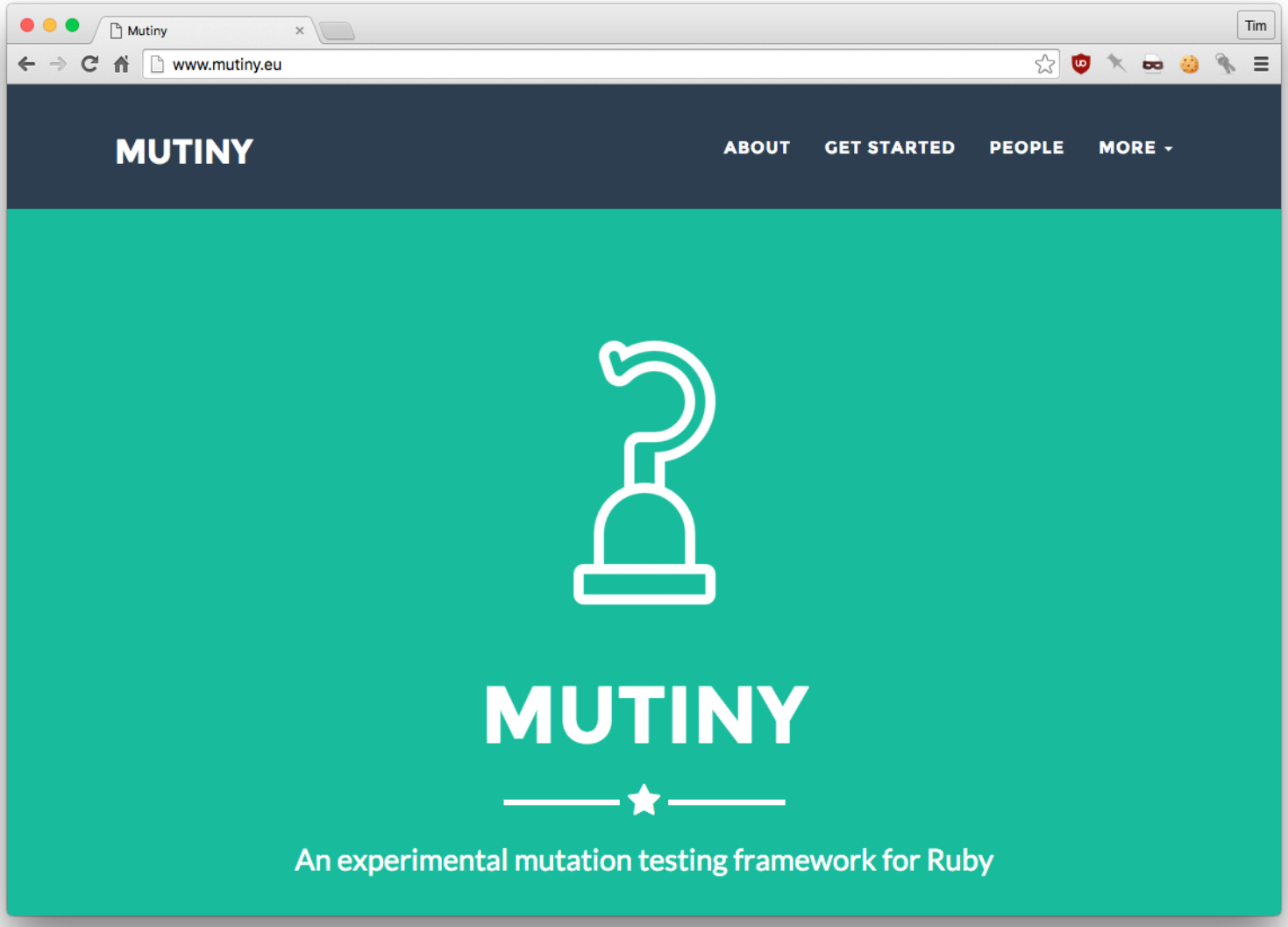

Figure 6: The mutiny website

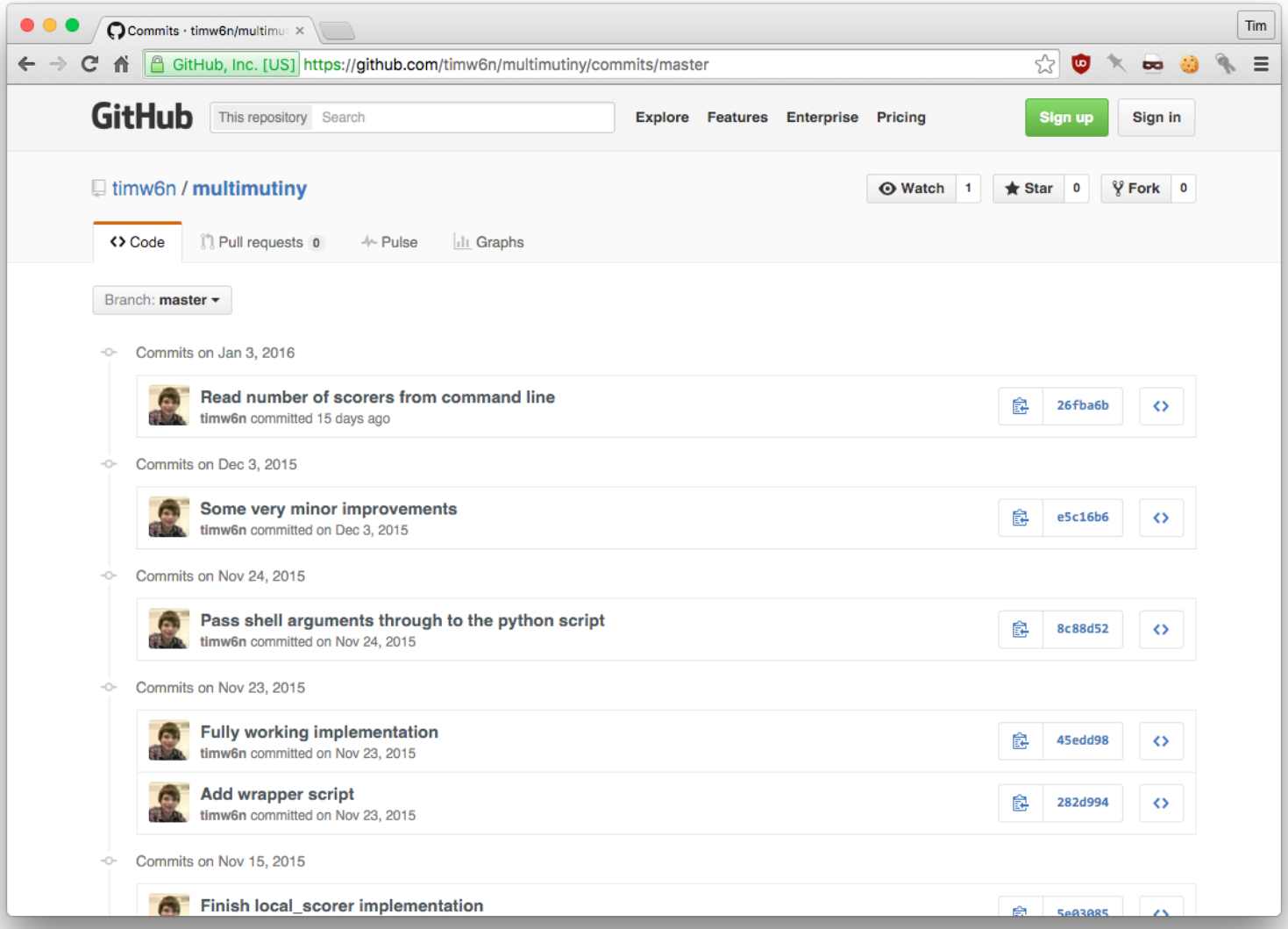

Figure 7: multimutiny, the main codebase for my project

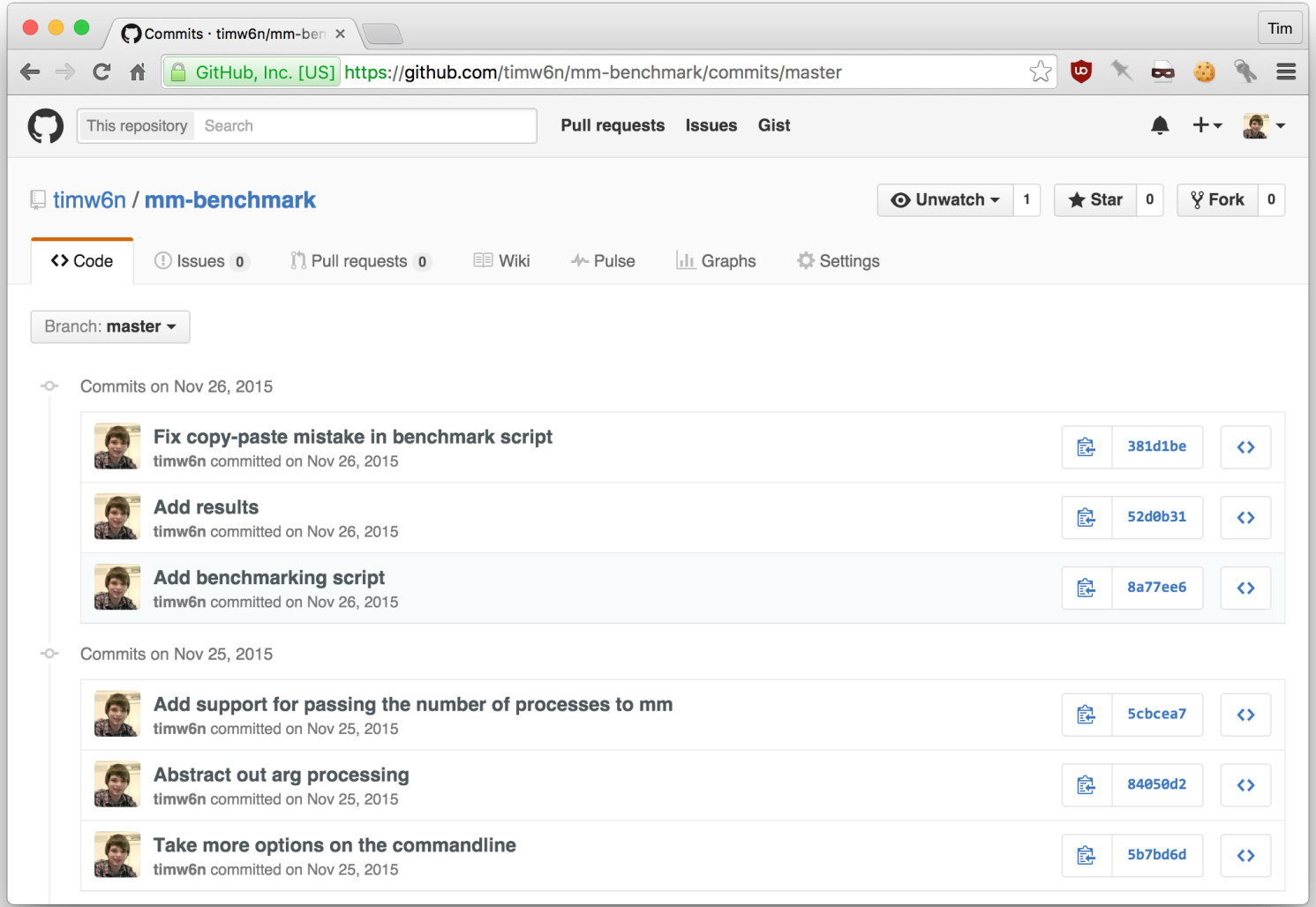

Figure 8: mm-benchmark, a tool for comparing multimutiny and mutiny performance across different-sized codebases and different numbers of parallel processes

Any questions?