Continuous Integration

COMP3613: Software Engineering II

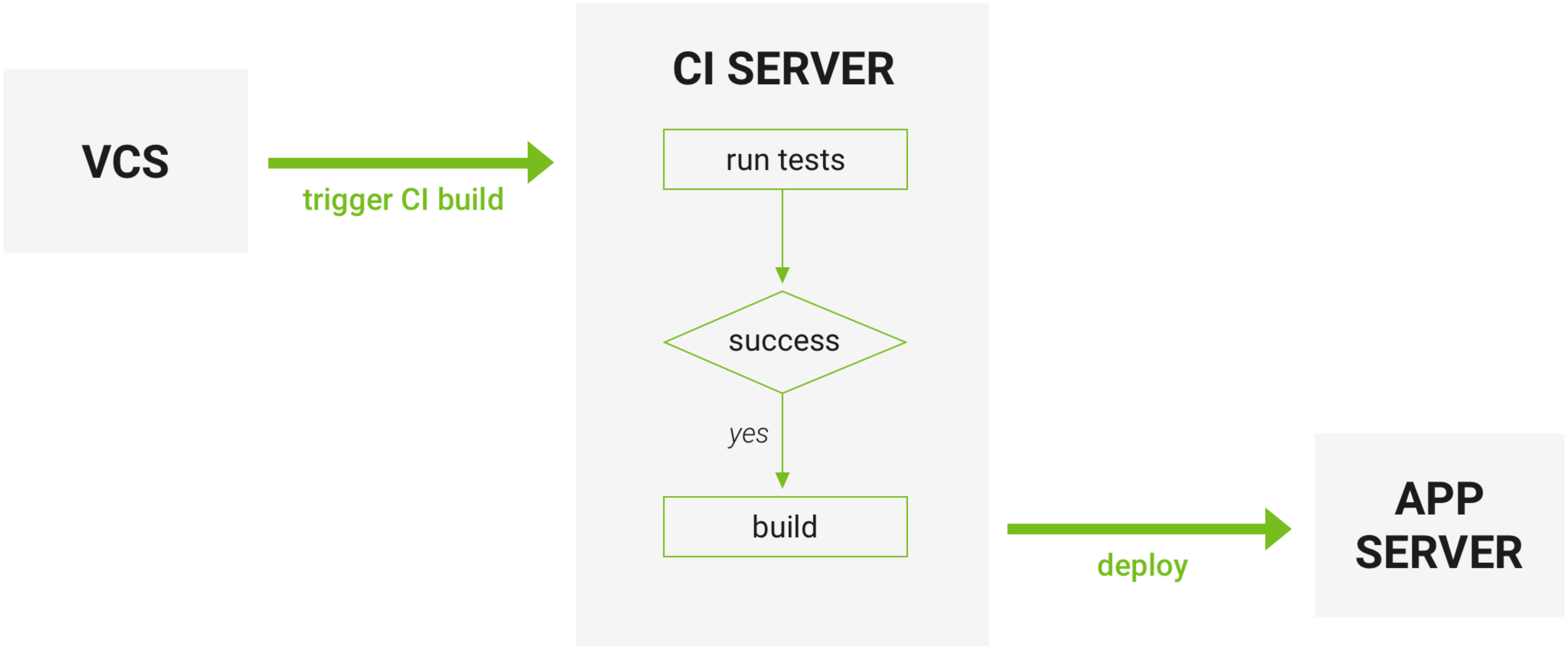
What is CI

- Want to develop software quickly
  - Integrate new code into codebase or build rapidly
  - But only want to integrate working code…
- How do we check if code is working?

Answer: Test!

What is CI

- CI (Continuous Integration) is about automating the process of testing and integrating code into operating build
  - CI tools “hook” into VCS (Version Control System). Generates test and build process on events - usually on push
- Pulls pressure off of dev team
- Allows for faster integration of code into build
- Can also perform checks for linting, coding style, etc…



From https://hackernoon.com/continuous-integration-circleci-vs-travis-ci-vs-jenkins-41a1c2bd95f5

How to do CI

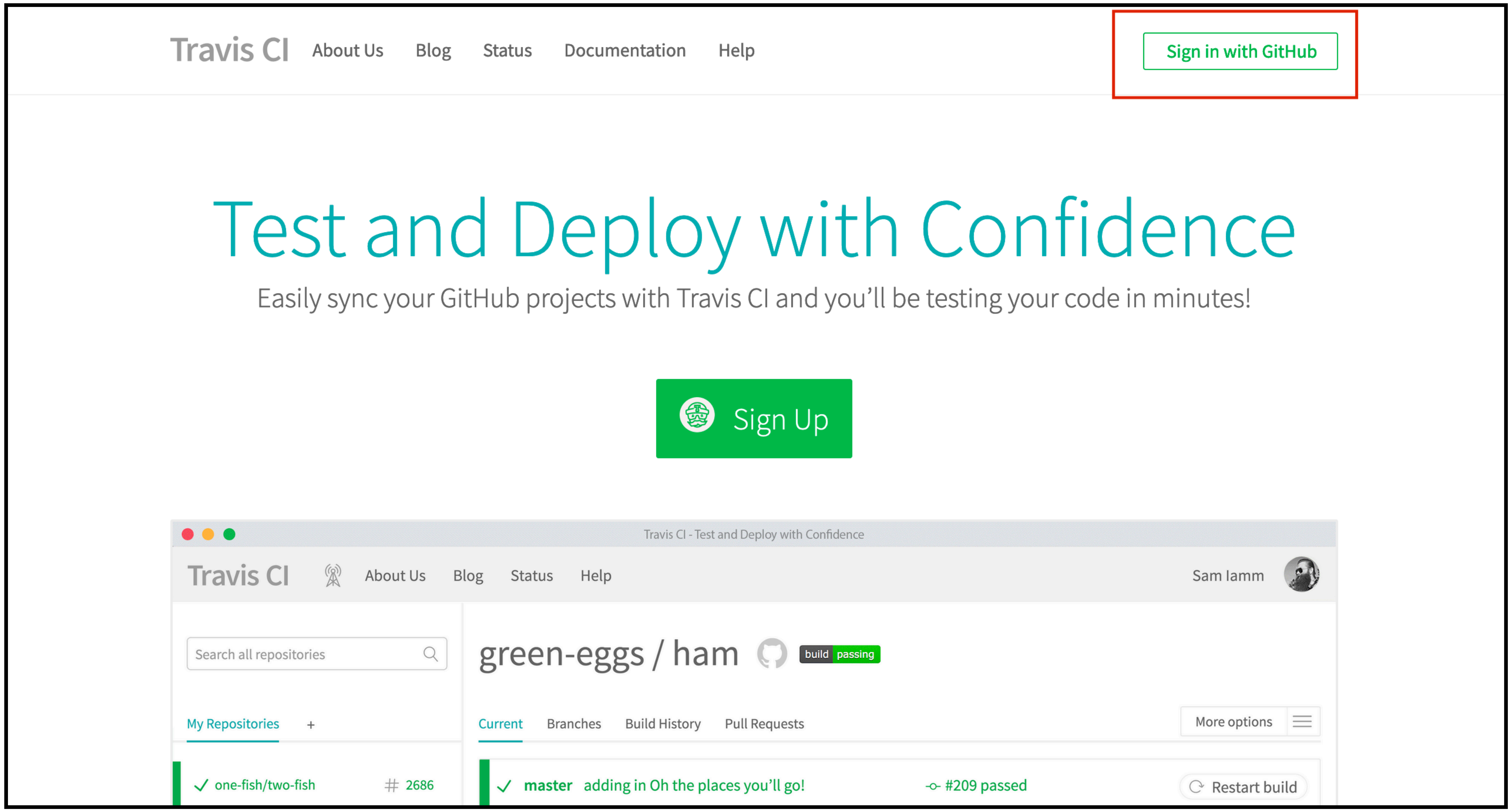
- Several providers with different advantages and disadvantages
  - Examples: Travis CI, Jenkins, Circle CI, etc…
  - All play well with Git
- Travis CI is the easiest start get up and running with
- Configuration differs, but underlying concepts remain the same between providers

Using Travis CI

- Travis CI is free for open source projects
- Firstly, you need to create a Github account, and then

Using Travis CI

- Sign up/sign in into Travis CI

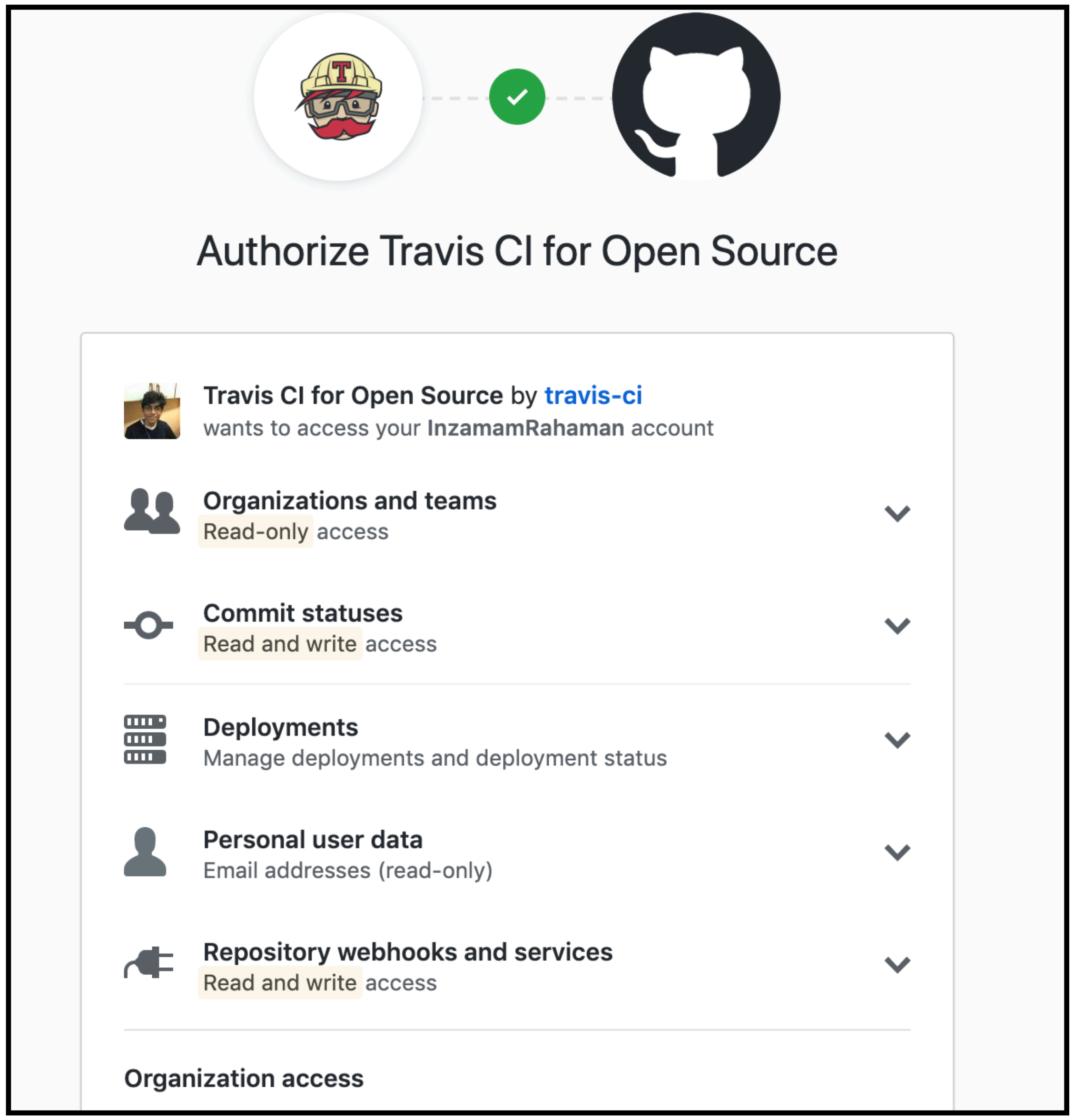


Test and Deploy with Confidence

Easily sync your GitHub projects with Travis CI and you'll be testing your code in minutes!

Using Travis CI

- Travis CI needs permission to read/write to repos



Authorize Travis CI for Open Source

Travis CI for Open Source by **travis-ci**
wants to access your **InzamamRahaman** account

Organizations and teams
Read-only access

Commit statuses
Read and write access

Deployments
Manage deployments and deployment status

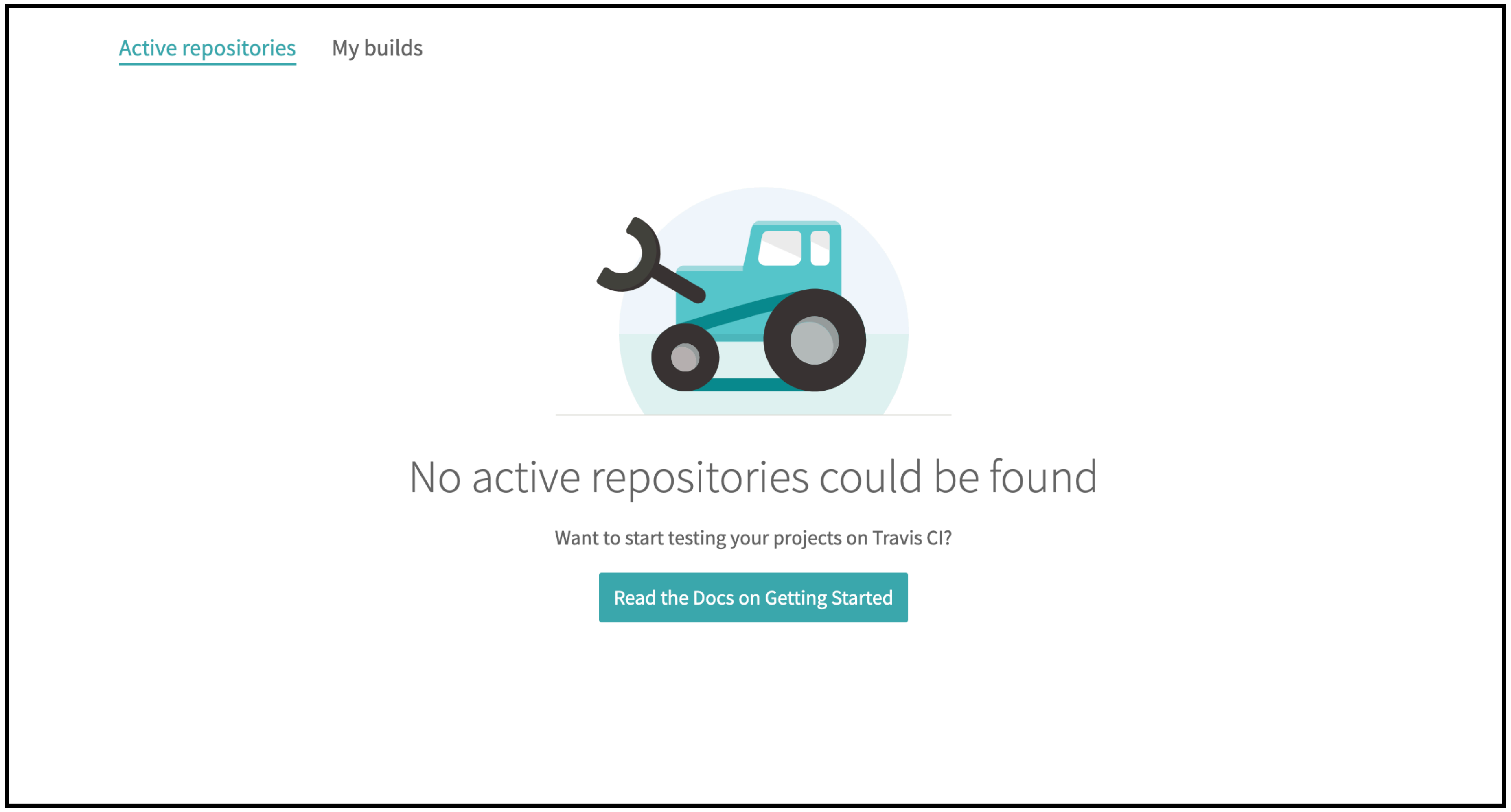
Personal user data
Email addresses (read-only)

Repository webhooks and services
Read and write access

Organization access

Using Travis CI

- No active repos as first
- Need to setup repos toe allow Travis CI to monitor them



No active repositories could be found

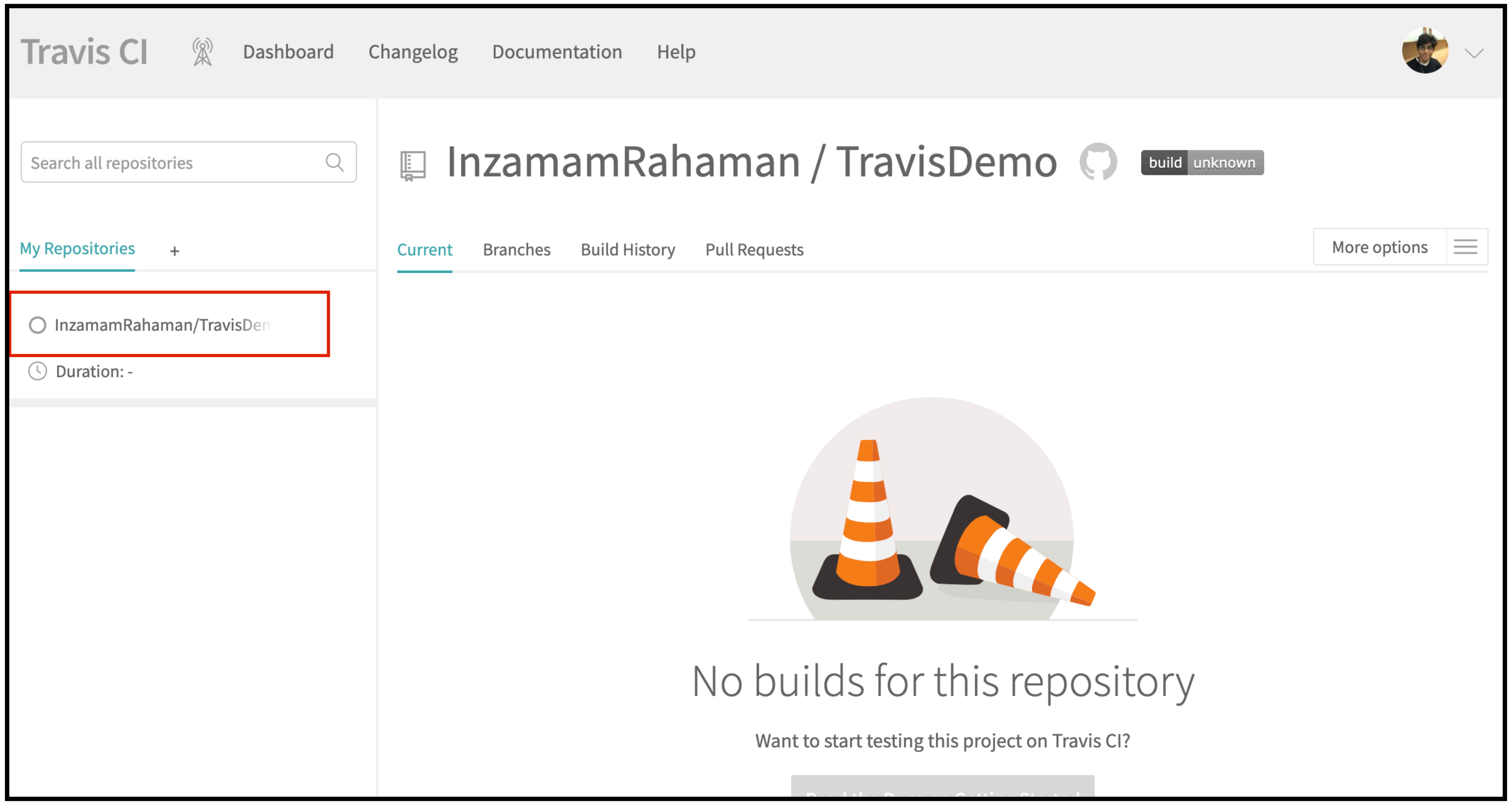
Want to start testing your projects on Travis CI?

Using Travis CI

- Repo needs .travis.yml file
  - Hidden (hence the “.” as the prefix)
  - YAML file
    - Way to easily encode configuration
  - Can also use Tox files
- Let’s create such a repo. Use the .zip folder provided
- After adding your repo, you might need to log out and then log back in

Using Travis CI

- No active repos as first
- Need to setup repos toe allow Travis CI to monitor them



No builds for this repository

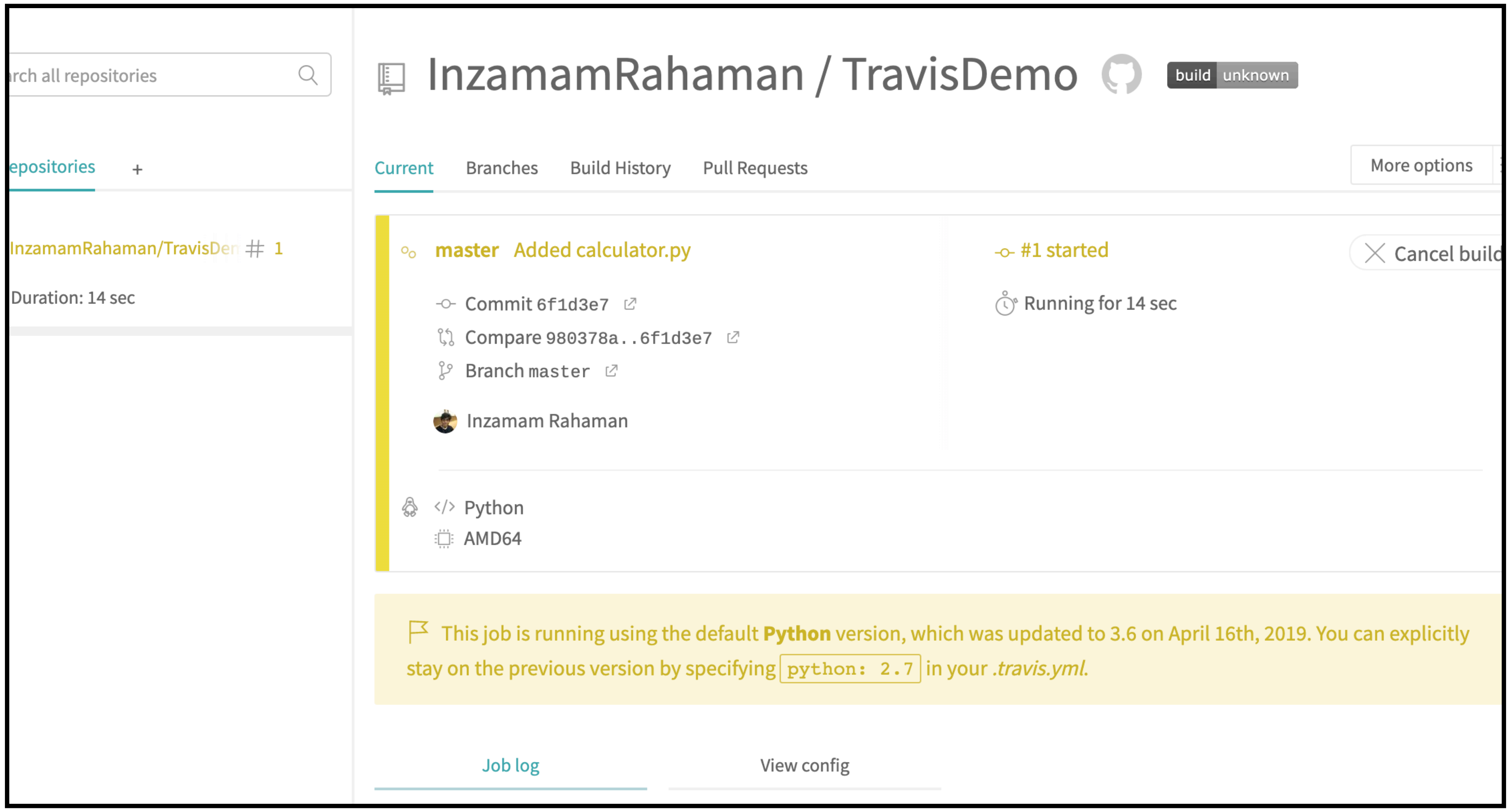
Want to start testing this project on Travis CI?

Using Travis CI

- Try to push the calculator.py file into the repo
- You should see that Travis has started a build process

Using Travis CI

- Building :-)



rch all repositories

epositories +

InzamamRahaman/TravisDe # 1

Duration: 14 sec

InzamamRahaman / TravisDemo build unknown

Current Branches Build History Pull Requests More options

master Added calculator.py

Commit 6f1d3e7

Compare 980378a..6f1d3e7

Branch master

Inzamam Rahaman

#1 started

Running for 14 sec

Cancel build

Python

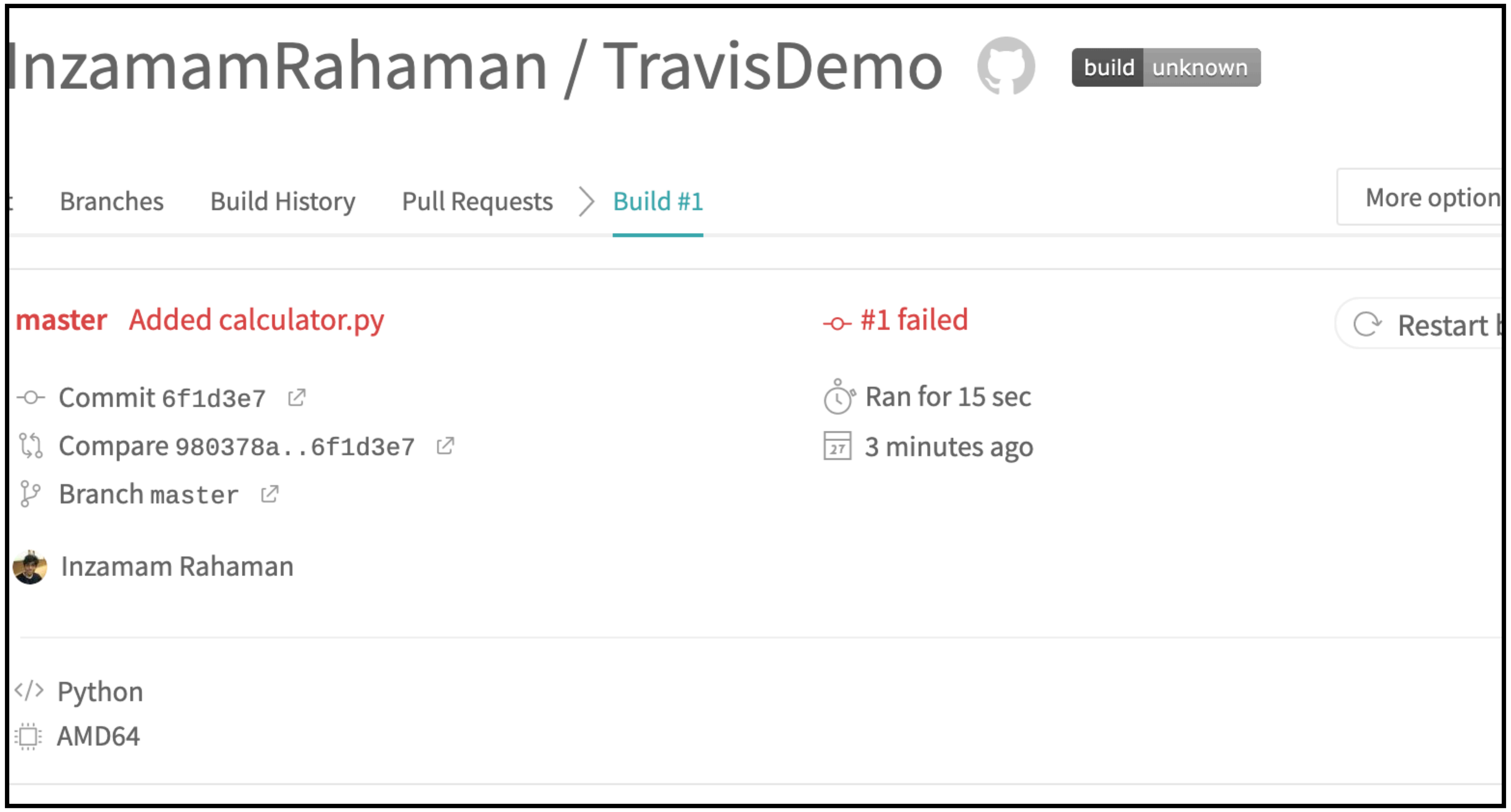
AMD64

This job is running using the default **Python** version, which was updated to 3.6 on April 16th, 2019. You can explicitly stay on the previous version by specifying `python: 2.7` in your *.travis.yml*.

Job log View config

Using Travis CI

- Build failed :-(
- Let's scroll down and see why



InzamamRahaman / TravisDemo

master Added calculator.py

#1 failed

Commit 6f1d3e7

Compare 980378a..6f1d3e7

Branch master

Ran for 15 sec

3 minutes ago

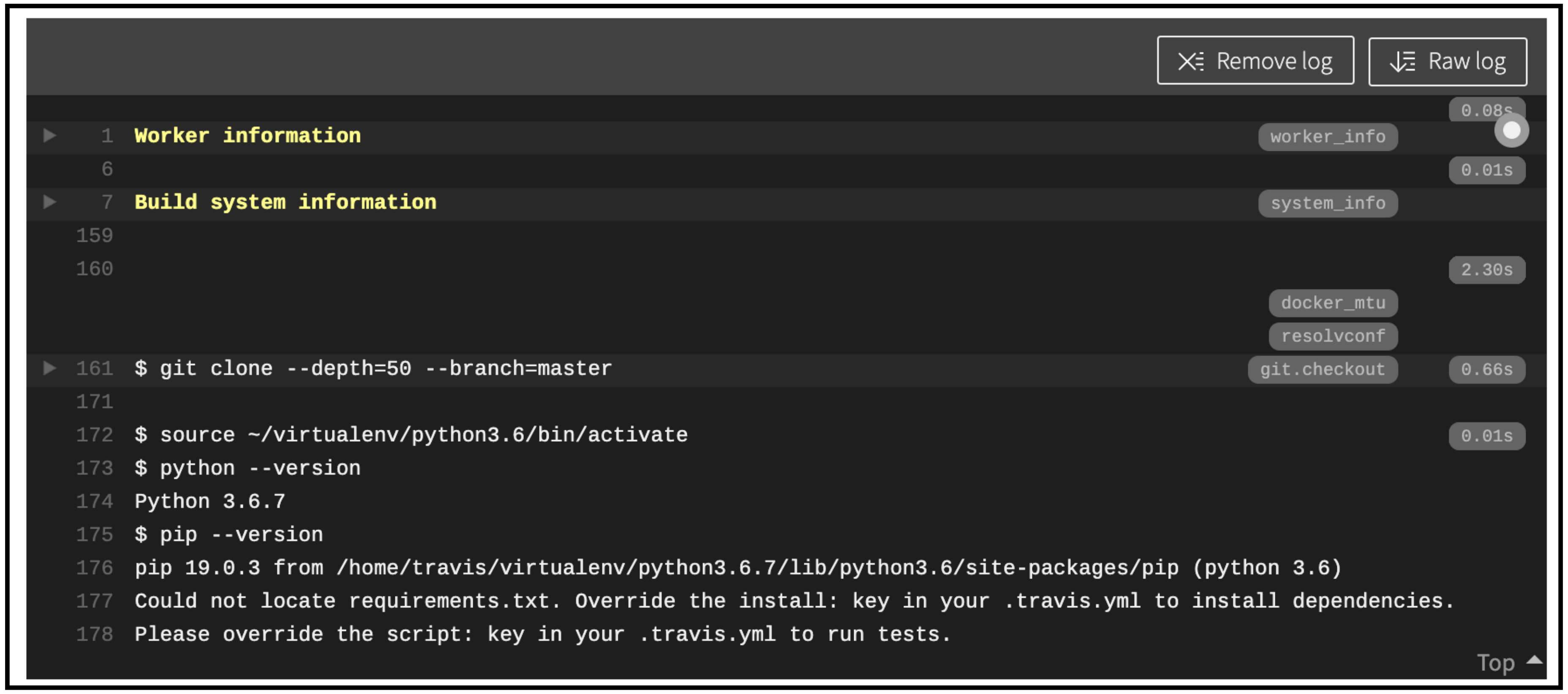
Inzamam Rahaman

Python

AMD64

Using Travis CI

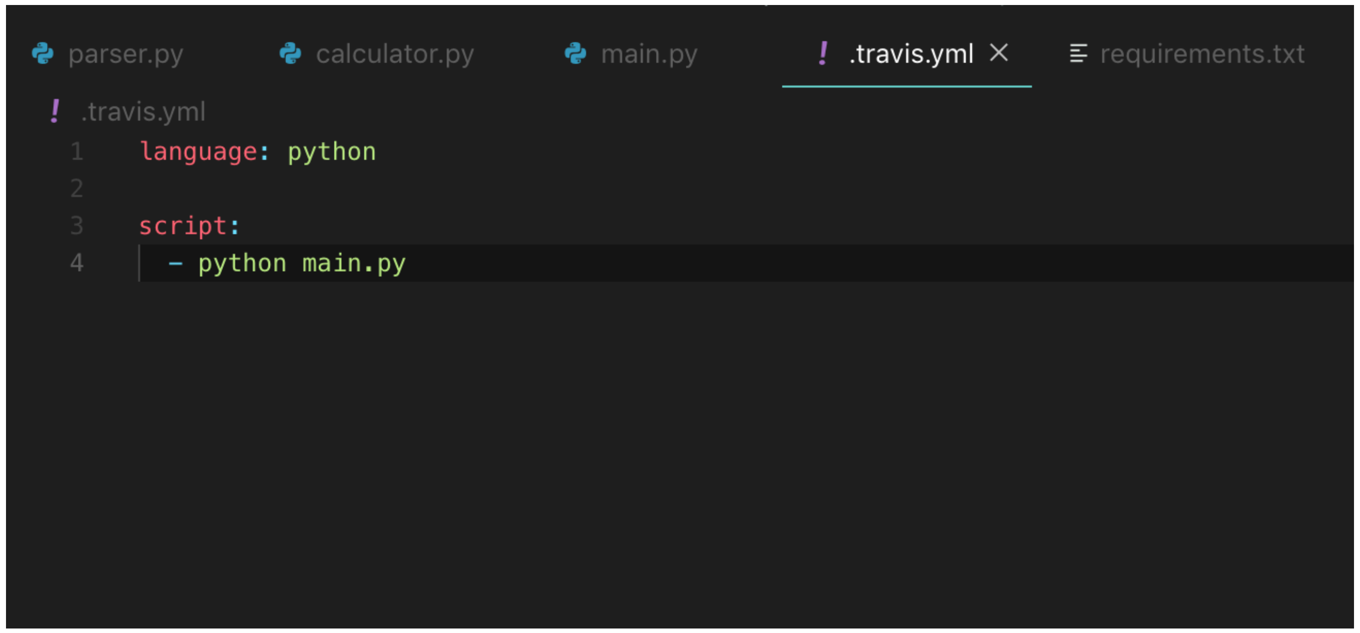
- Need to specify specifics of environment
- In the case of Python, that means a requirements.txt file! We also need to specify the script configuration of the .travis.yml
- Fix and run again



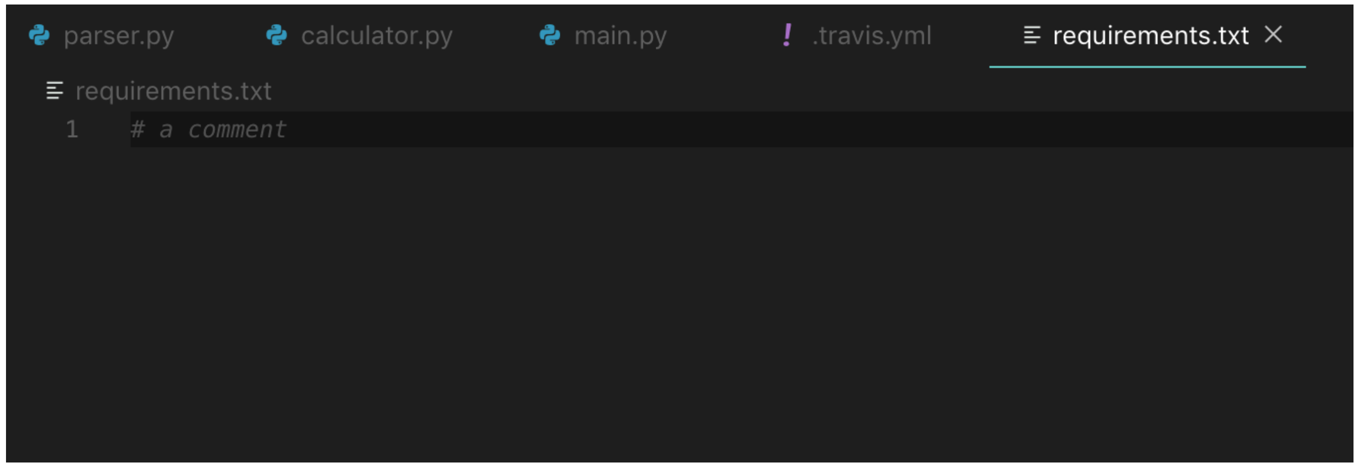
```
Worker information                                  worker_info   0.08s
                                                                  0.01s
Build system information                            system_info
                                                                  2.30s
                                                    docker_mtu
                                                    resolvconf
$ git clone --depth=50 --branch=master              git.checkout  0.66s

$ source ~/virtualenv/python3.6/bin/activate                      0.01s
$ python --version
Python 3.6.7
$ pip --version
pip 19.0.3 from /home/travis/virtualenv/python3.6.7/lib/python3.6/site-packages/pip (python 3.6)
Could not locate requirements.txt. Override the install: key in your .travis.yml to install dependencies.
Please override the script: key in your .travis.yml to run tests.
```

Fix YAML file

- Fix the .yml file
- Create requirements.txt file with a single line comment



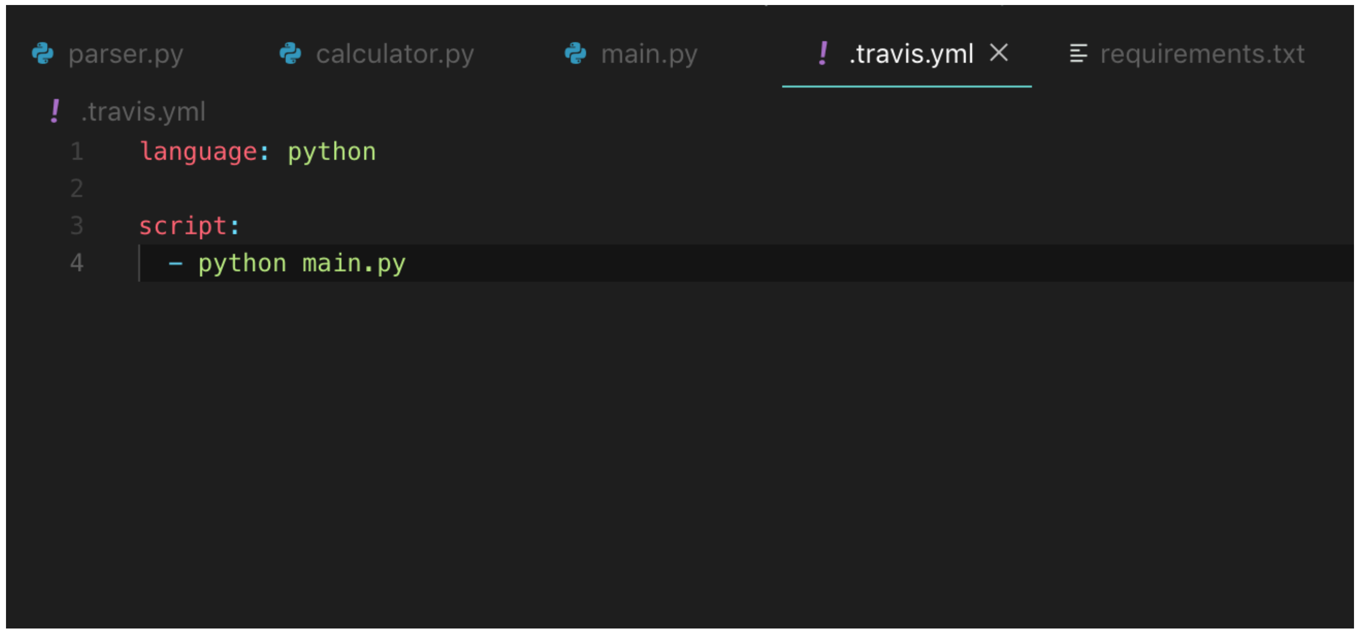
```
language: python

script:
  - python main.py
```



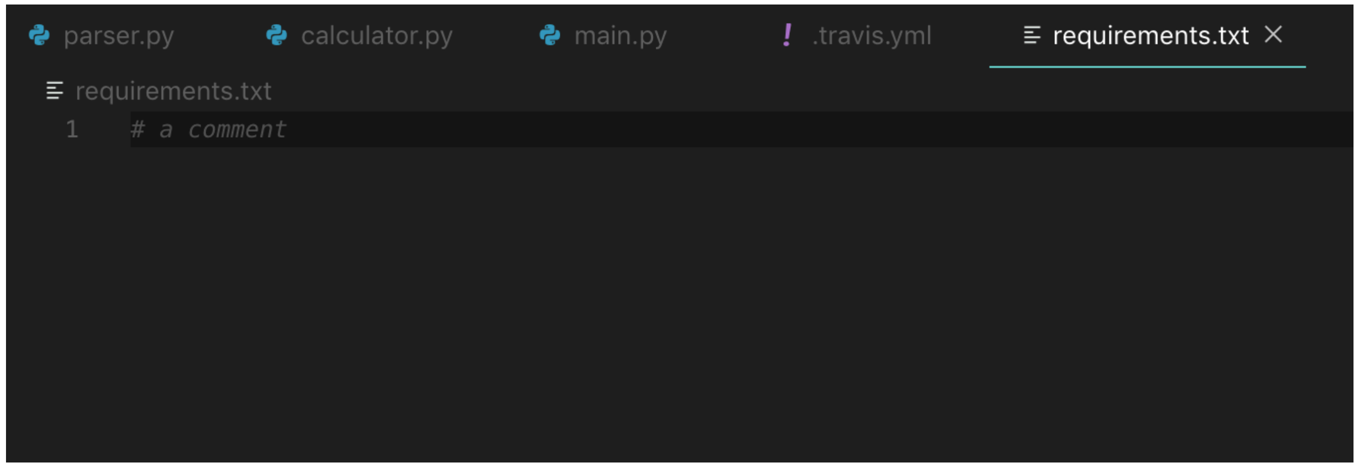
```
# a comment
```

Fix YAML file

- Fix the .yml file
- Create requirements.txt file with a single line comment



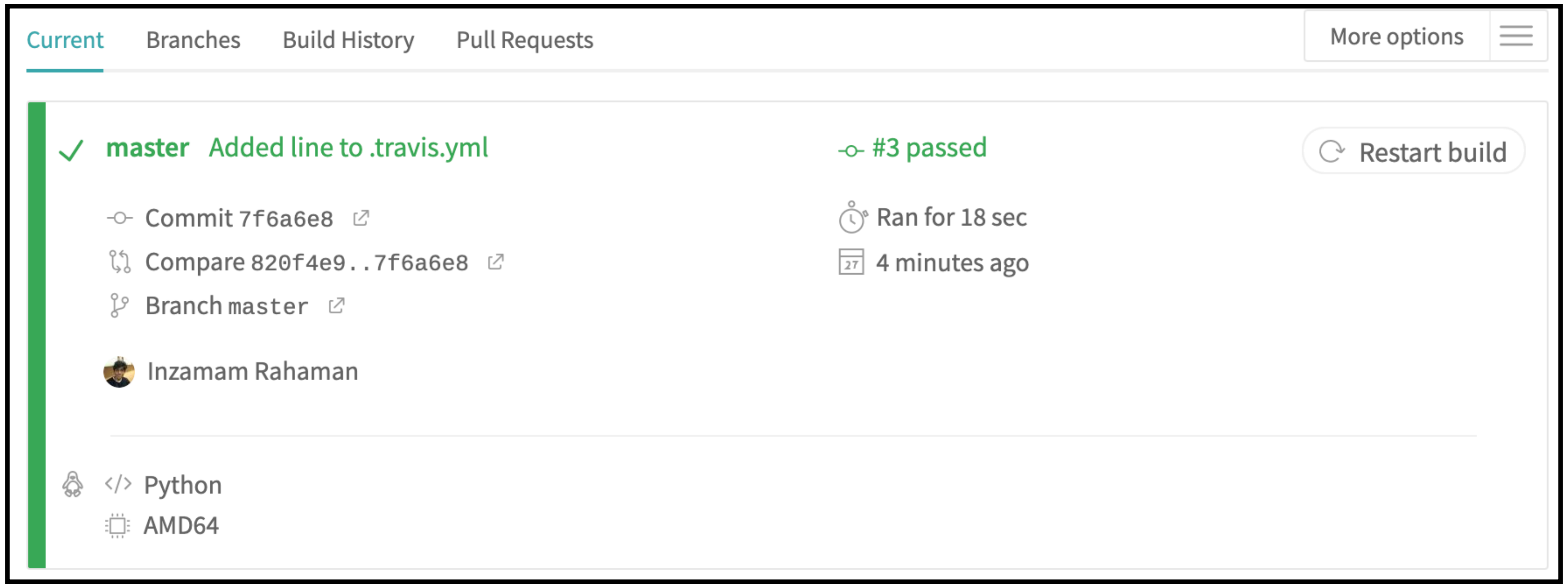
```yaml
language: python

script:
  - python main.py
```



```
# a comment
```

Using Travis CI

- Build passed
- Let's setup some tests!



✓ master Added line to .travis.yml

#3 passed

Commit 7f6a6e8

Compare 820f4e9..7f6a6e8

Branch master

Ran for 18 sec

4 minutes ago

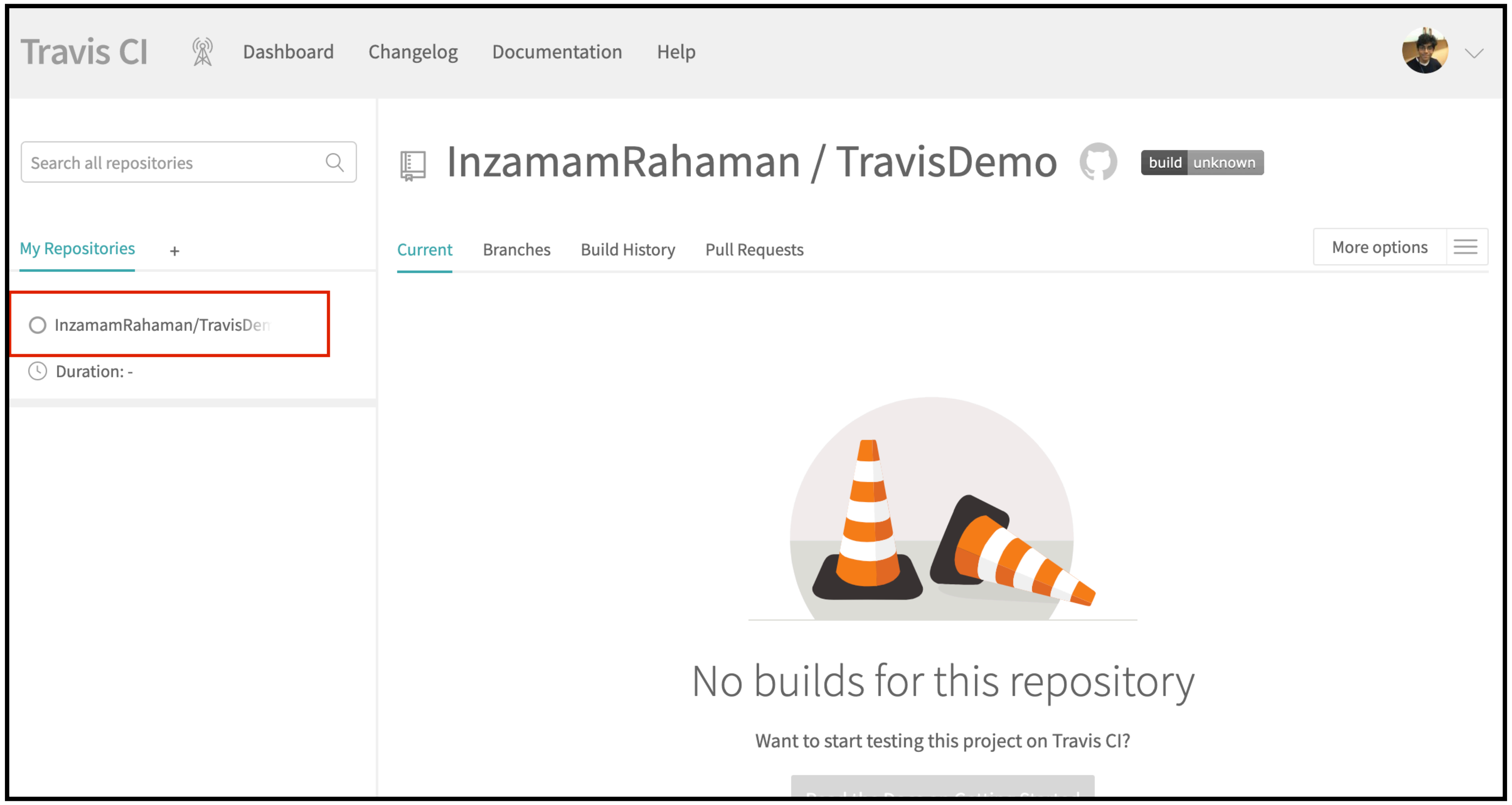
Inzamam Rahaman

Python

AMD64

Using Travis CI

- No active repos as first
- Need to setup repos toe allow Travis CI to monitor them

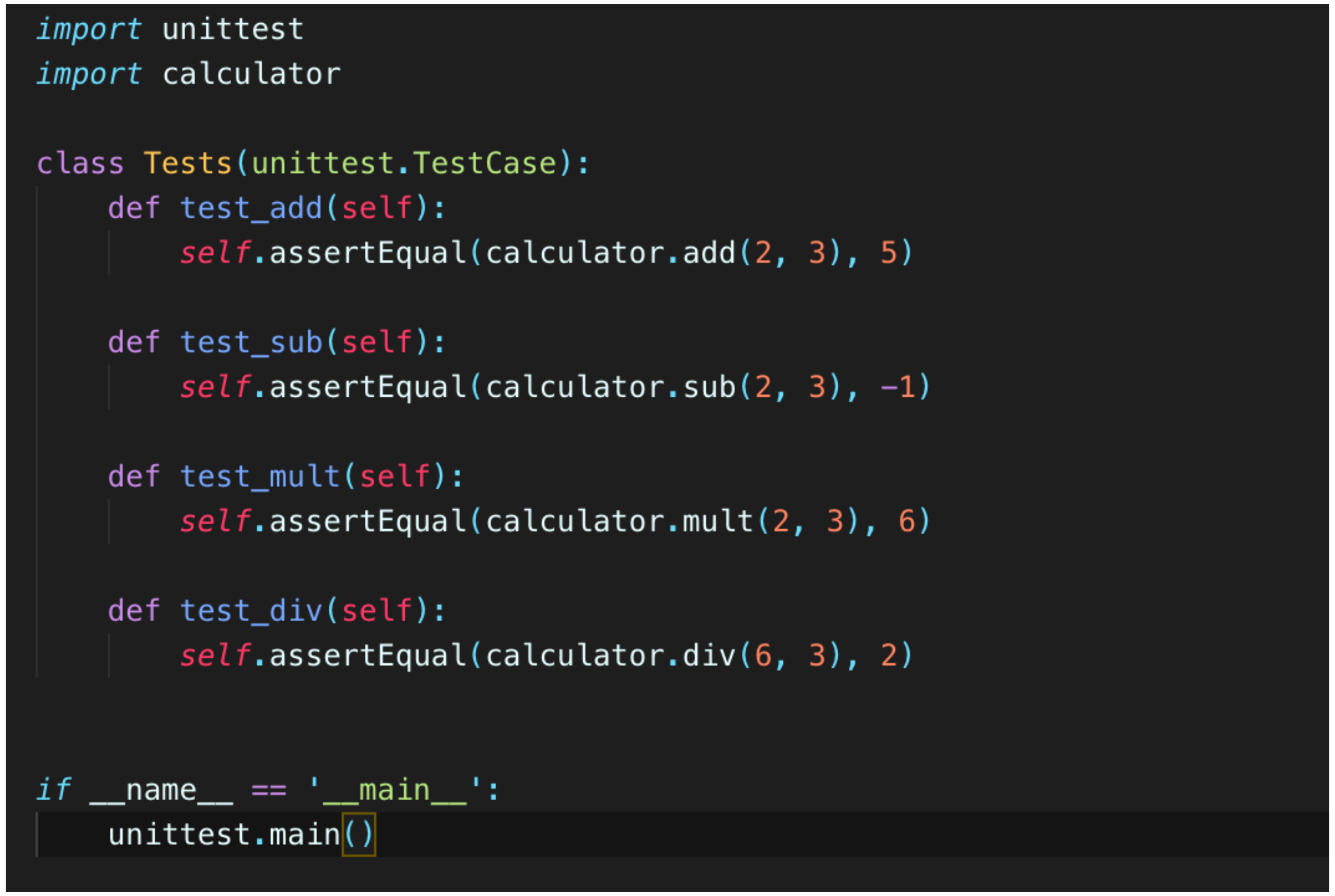


No builds for this repository

Want to start testing this project on Travis CI?

Create Unittest

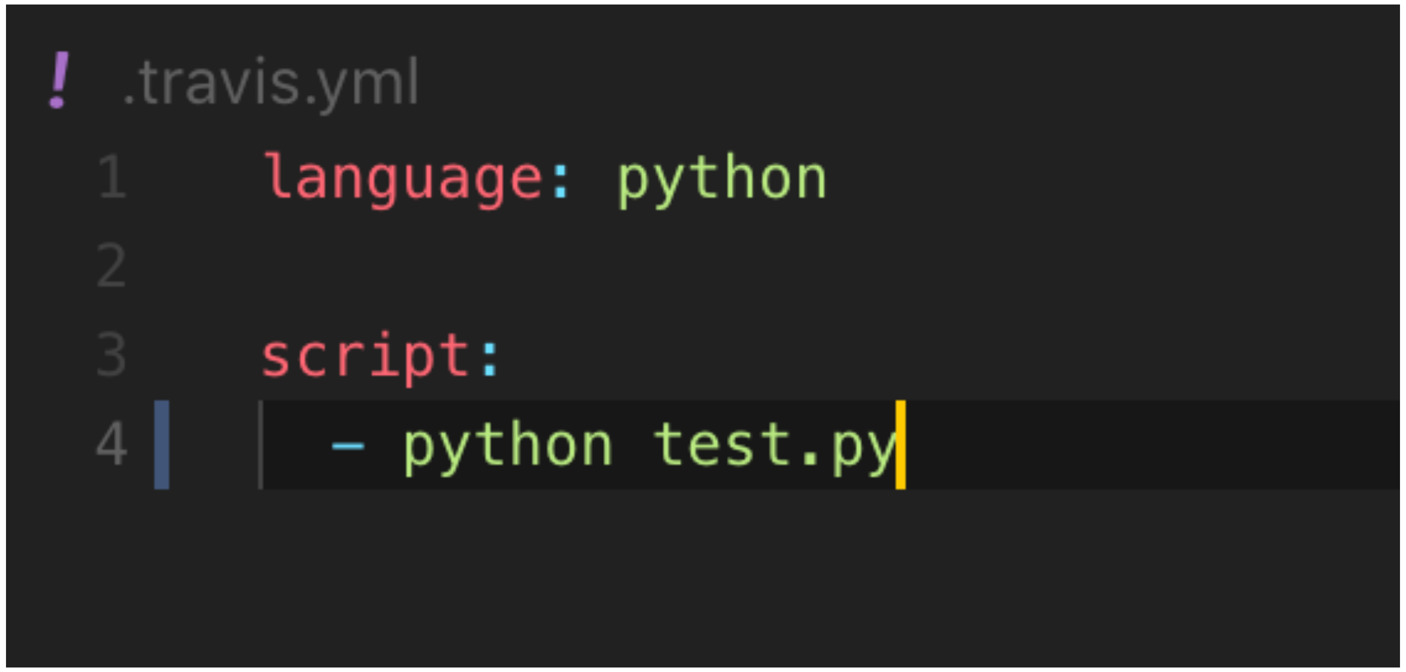
- Create a unittest for the functions in the calculator.py file
- Name the file "test.py"
- Change script action from "python main.py" to "python test.py"



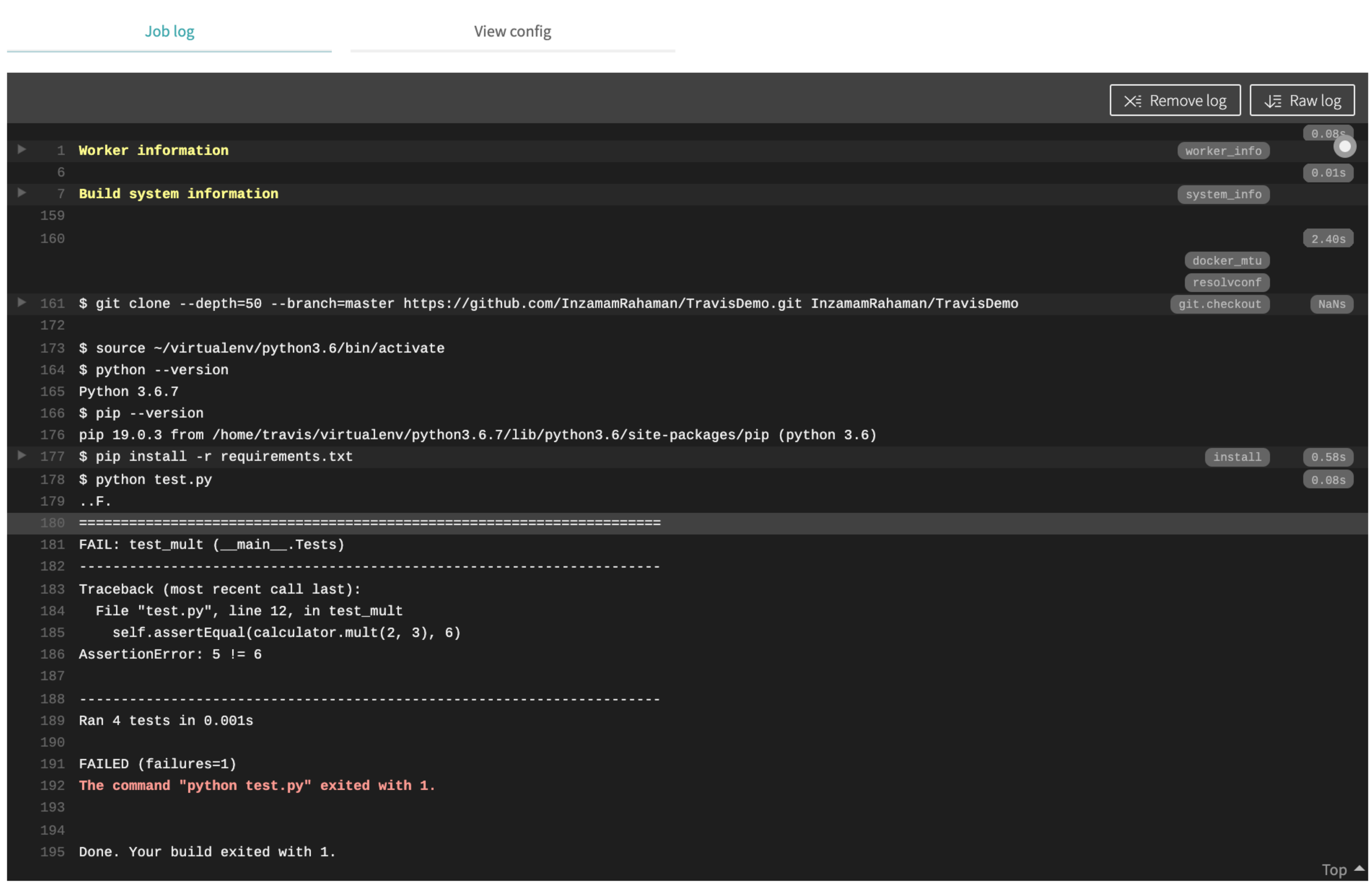
```python
import unittest
import calculator

class Tests(unittest.TestCase):
    def test_add(self):
        self.assertEqual(calculator.add(2, 3), 5)

    def test_sub(self):
        self.assertEqual(calculator.sub(2, 3), -1)

    def test_mult(self):
        self.assertEqual(calculator.mult(2, 3), 6)

    def test_div(self):
        self.assertEqual(calculator.div(6, 3), 2)


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

Edit .travis.yml

- Edit the .yml file
- Add
- Commit
- Push to git



```yaml
language: python

script:
  - python test.py
```



```
Worker information

Build system information


$ git clone --depth=50 --branch=master https://github.com/InzamamRahaman/TravisDemo.git InzamamRahaman/TravisDemo

$ source ~/virtualenv/python3.6/bin/activate
$ python --version
Python 3.6.7
$ pip --version
pip 19.0.3 from /home/travis/virtualenv/python3.6.7/lib/python3.6/site-packages/pip (python 3.6)
$ pip install -r requirements.txt
$ python test.py
..F.
======================================================================
FAIL: test_mult (__main__.Tests)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test.py", line 12, in test_mult
    self.assertEqual(calculator.mult(2, 3), 6)
AssertionError: 5 != 6

----------------------------------------------------------------------
Ran 4 tests in 0.001s

FAILED (failures=1)
The command "python test.py" exited with 1.


Done. Your build exited with 1.
```

Using Travis CI

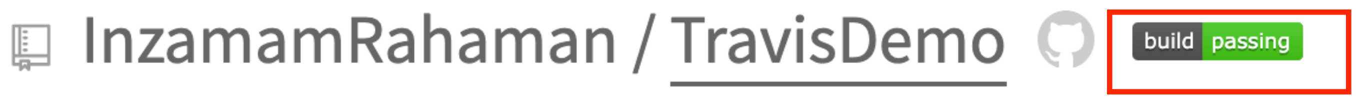
- Test failed
- Once we fix, we commit and push again
- We should have a passing build :-)

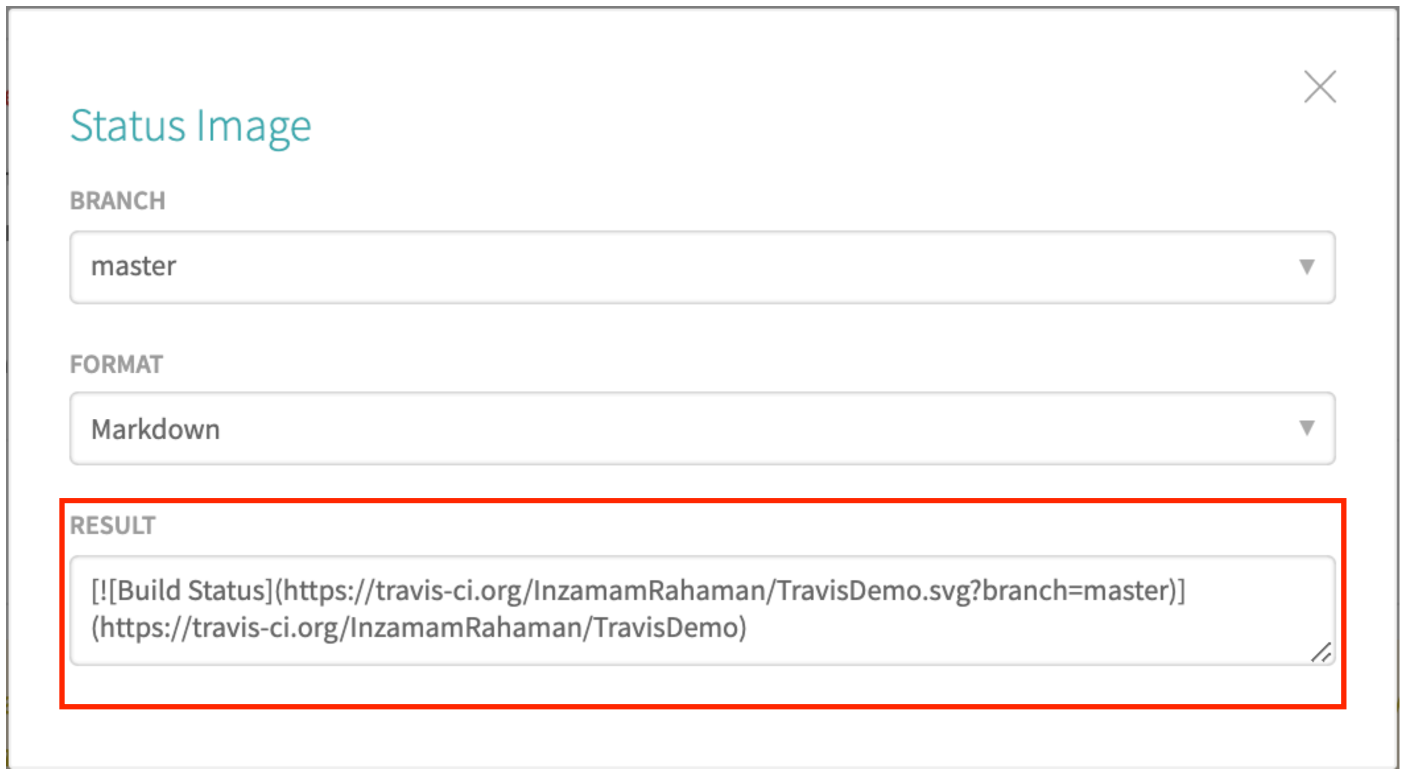
Using Travis CI

- deployment is another config option
- Stores steps to deploy application once build passes
- Won’t cover today, but useful concept to research

Badges

- Can show badge on README.md if build is passing!
- Click badge on Travis site
- Copy Markdown into README.md




Status Image

BRANCH
master

FORMAT
Markdown

RESULT
[![Build Status](https://travis-ci.org/InzamamRahaman/TravisDemo.svg?branch=master)](https://travis-ci.org/InzamamRahaman/TravisDemo)