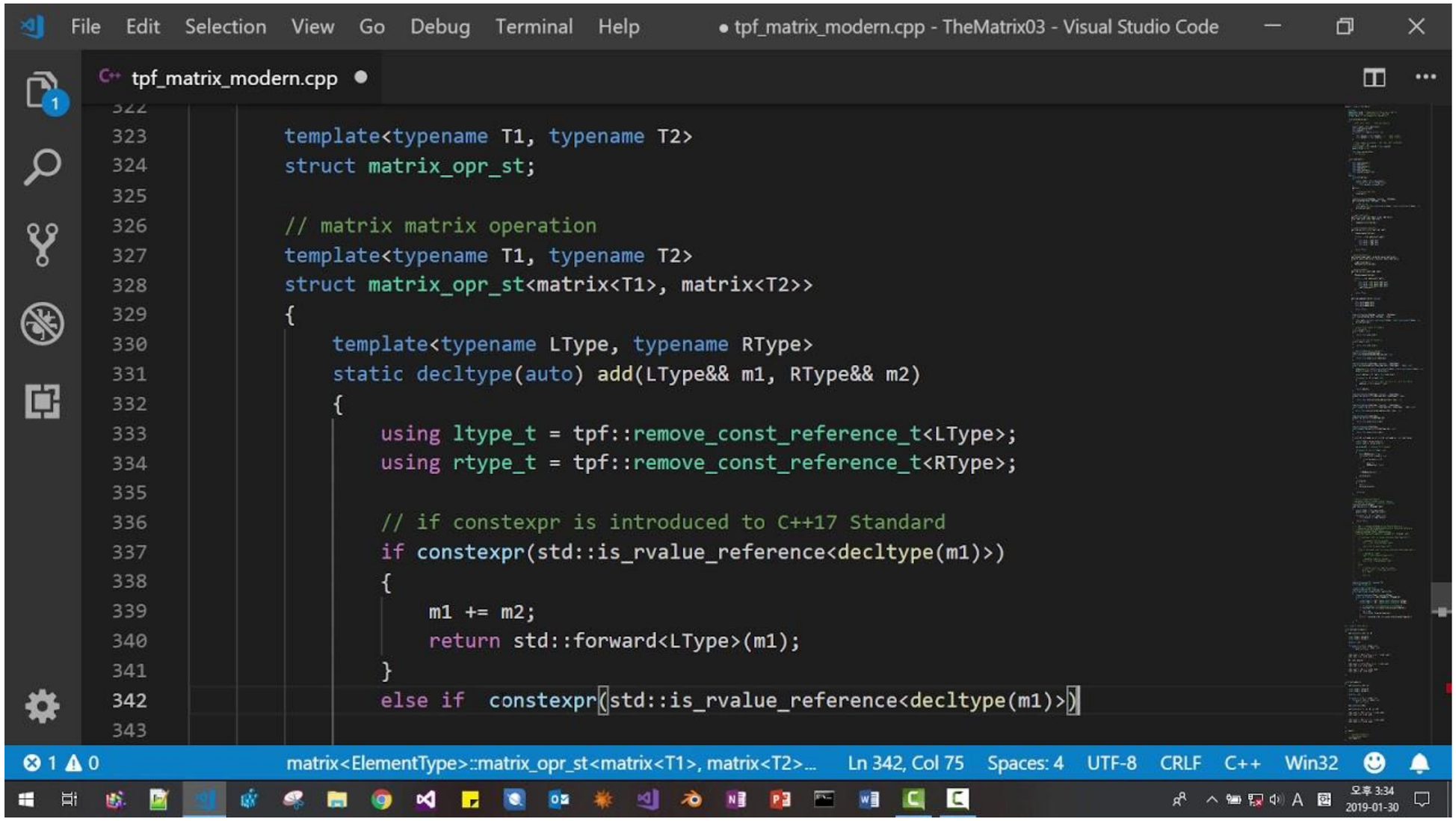
The Missing Semester:
Metaprogamming



Sri Venkateswara
College of Engineering

What is Metaprogramming?

- Imagine a guy who builds cars. Say it's the same thing as using a computer.
- At some point he realizes he's always doing the same thing, more or less. So he builds factories to build cars, and it's much better.
- He's now programming!
- Nevertheless, once again, at some point, he realizes he's always doing the same thing, to some extent. Now he decides to build factories that build factories that build cars.
- That's metaprogramming.

Simply put , Metaprogramming is the practice of writing code that writes code for us .

The Missing Semester:
An Introduction to CI/CD with Travis CI and Python

Sri Venkateswara
College of Engineering

Part I - Choosing a continuous integration service

What to consider when creating a Python3 CI/CD pipeline :

- **Python Community on GitHub** - support for all python3 versions.
- **Choose your testing framework** - Use it with any popular testing framework: pytest, unittest, etc.
- **Open vs. closed source** - whenever possible choose an open source option. The below providers generally offer zero cost options for new open source projects with reasonable build constraints. If you go closed source, you're going to needs to fork over some .

Part I - Choosing a continuous integration service

Common Providers

- **Travis CI**: The one used in today's session
- **Circle CI**: Seems to have a little saner syntax and finer configuration relative to Travis CI
- **GitLab**: Leans open source where the others are closed source
- **GitHub Actions**: Newer, native to GitHub, and offers Linux, Windows, and MacOS builds.

Part II - Prerequisites

You should have basic knowledge about

- GitHub
- Python
- Testing frameworks
- How to use the Terminal.

Travis CI

Knowledge of Pipenv (Pipfiles) and pytest is a plus, but not required. CI/CD tools are a great way to learn both.

YAML Basics

Most CI services interact with your repositories using a YAML configuration file.

YAML - YAML Syntax, from Red Hat Ansible

Sample Travis CI YAML Program

```yaml
# .travis.yml

dist: xenial

language: python

cache: pip

python:
    - "3.6"
    - "3.7"
    - "3.8"
    - "nightly"

matrix:
    allow_failures:
        - python: "nightly"

install:
    - pip install pipenv --upgrade-strategy=only-if-needed
    - pipenv install --dev

script:
    - bash scripts/test.sh

after_script:
    - bash <(curl -s https://codecov.io/bash)
```

Travis CI

CI/CD SETUP
STEP I - Get setup to install my python package project template

Ensure you have a compatible python 🐍 environment on your computer.

```
python3 --version
python3 -m pip --version
python3 -m pytest --version
```

If you are missing any of the above, you can add the packages to your main python install.

Travis CI

Step I - Get setup to install my python package project template

```
sudo apt-get update
sudo apt-get install python3-pip
sudo apt-get install python3-pytest
```

Now that we have confirmed your base python setup, let's go ahead and ensure the pipenv and cookiecutter python packages are installed.

```
python3 -m pip install --user pipenv
python3 -m pip install --user cookiecutter
```

Travis CI

Step II – Create a Python project and setup the Pipenv virtual environment

```
# make sure your path finds --user installs

## add `export PATH="$HOME/.local/bin:$PATH"`
## to your ~/.bashrc, ~/.zshrc file on linux

cookiecutter https://github.com/iancleary/pypackage
# For the sake of brevity, repos on GitHub can just use the 'gh' pre
cookiecutter gh:iancleary/pypackage
```

- Use cd new-directory to change into the new directory you just created (replace new-directory)
- Run pipenv to display it's options. Take a look at what’s offered.
- Next, run pipenv install --dev to install the production and development dependencies specified in the Pipfile.
- Run pipenv shell to load the virtual environment.

Travis CI

Step III- Run the tests locally and make sure they pass

1. Execute ./scripts/tests.sh from within the directory's Pipenv.

This command executes a bash script that does several things:

- runs pytest to check test cases and check test coverage
- checks formatting with the black package
- runs static type checking against your code base with mypy
- checks import sorting from standard lib, your application, and custom packages .

2. Push the directory to a remote git repo

- Git website: [Adding an existing project to GitHub using the command line]

Travis CI

Step IV – Connect Travis-CI to the repo

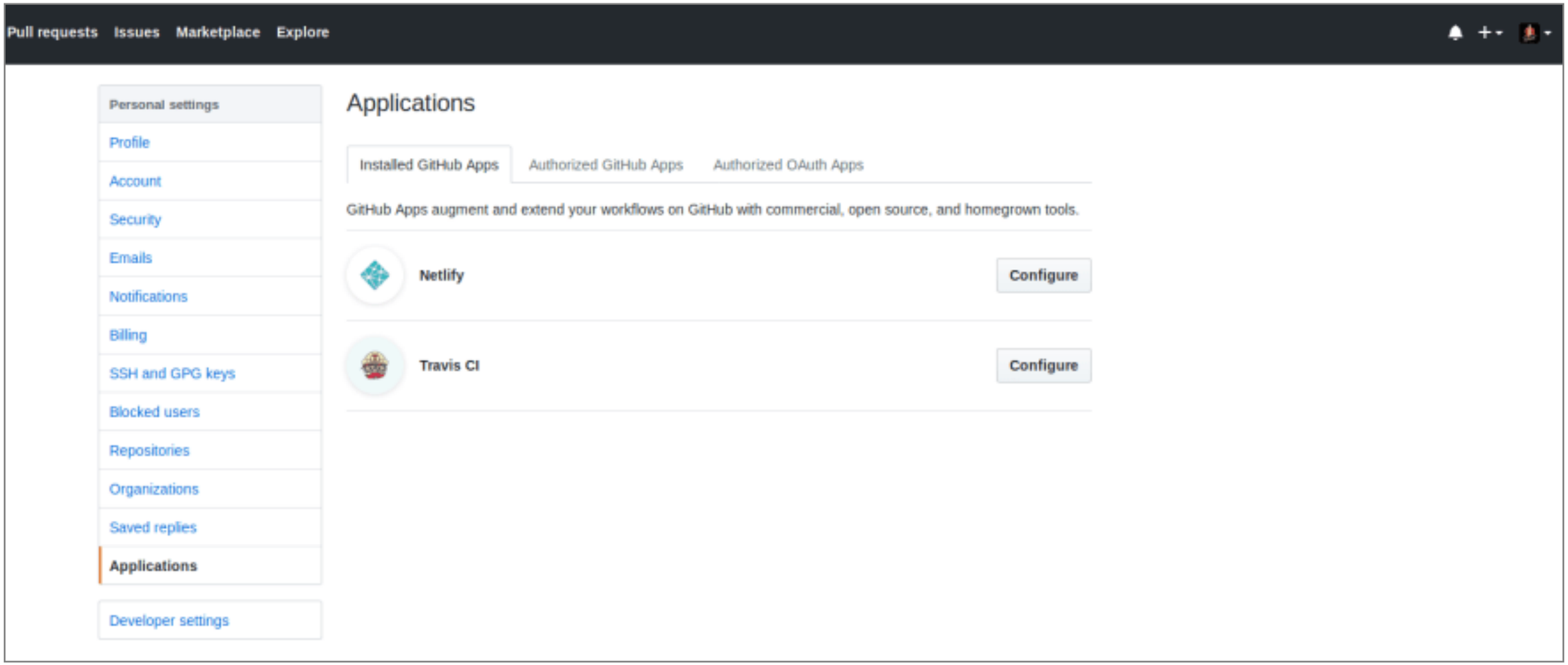
Head over to Travis CI and sign in with your GitHub account:

- https://help.github.com/en/enterprise/2.16/admin/developer-workflow/continuous-integration-using-travis-ci

Once you login to Travis CI and Enable GitHub Access, you'll want to configure Travis from your GitHub Settings page.

Travis CI

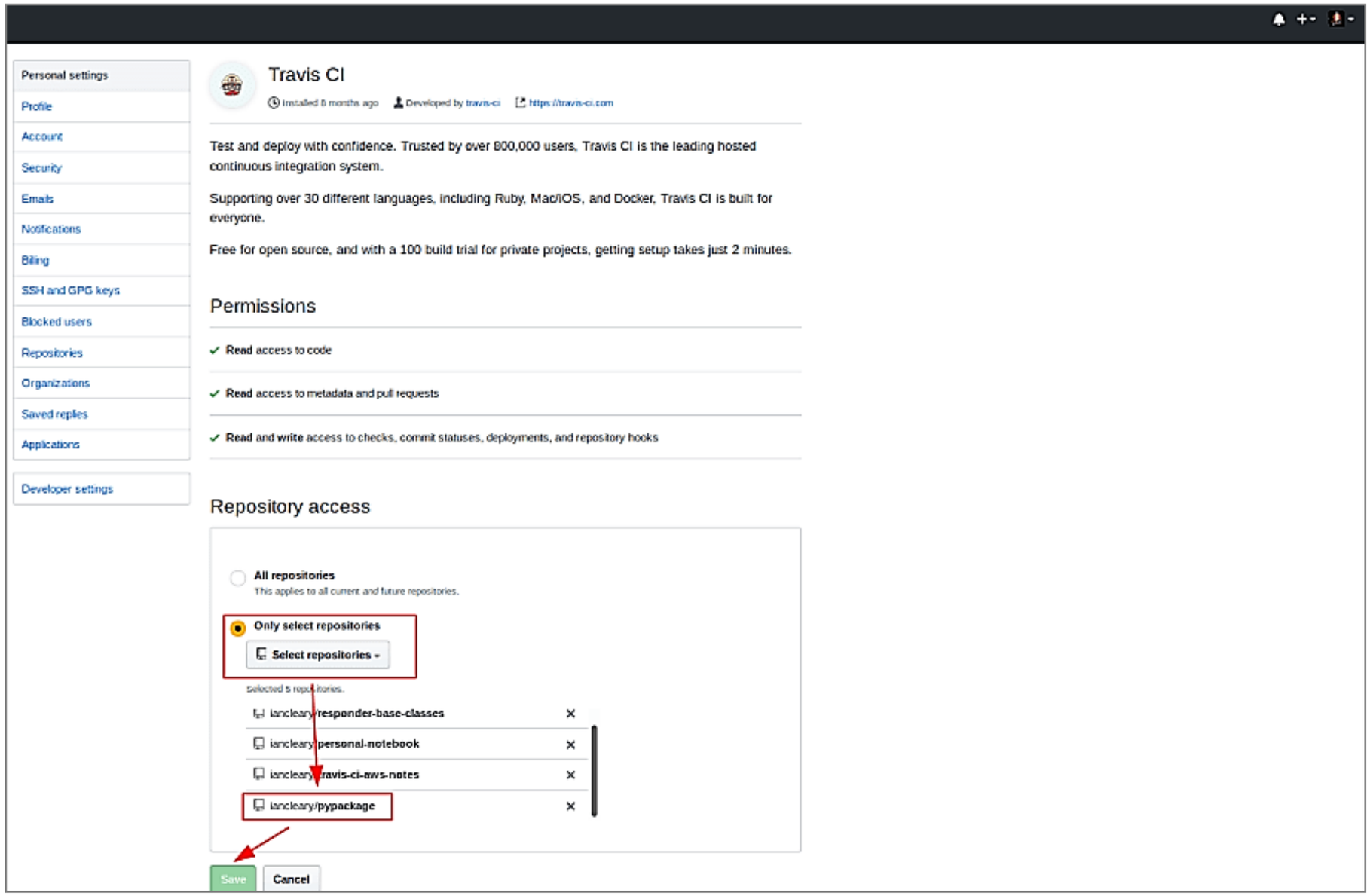
Step IV – Connect Travis-CI to the repo



Travis CI

Sri Venkateswara College of Engineering

Step IV – Connect Travis-CI to the repo



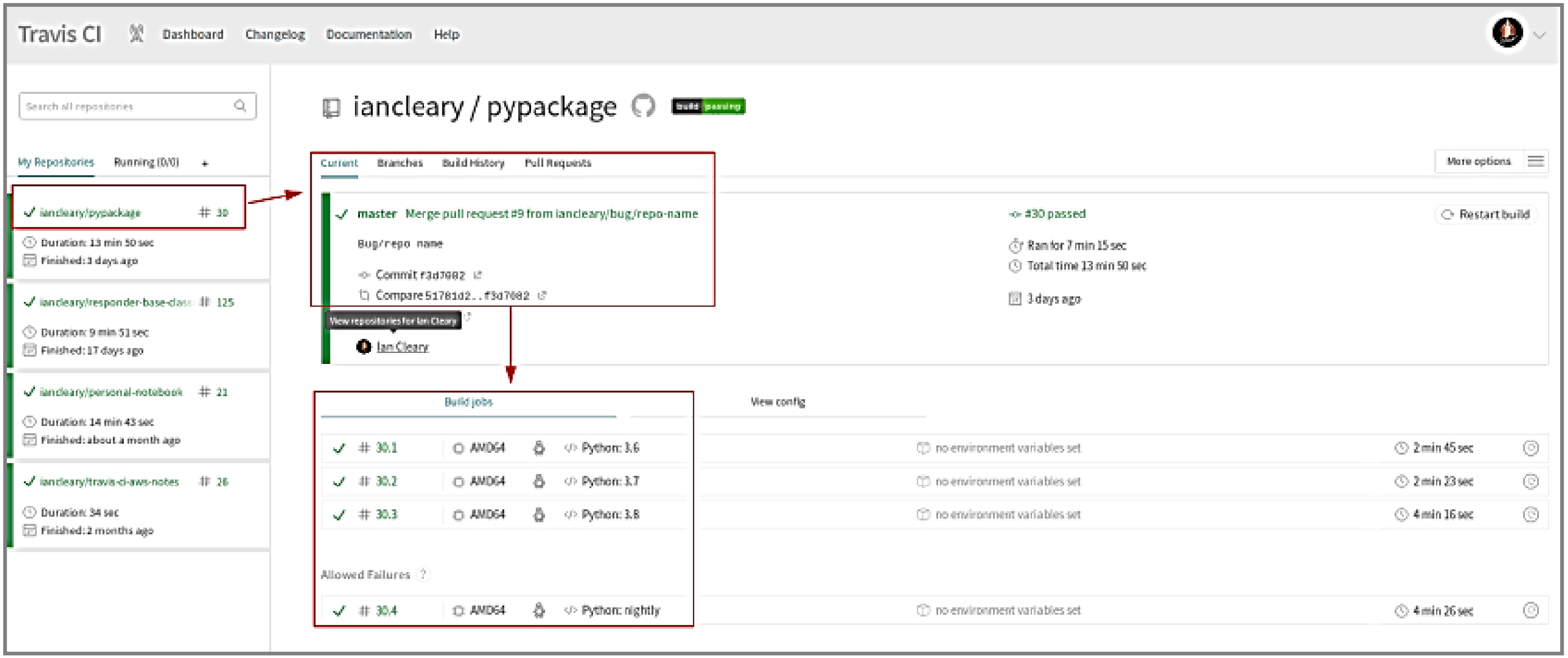
Travis CI

Sri Venkateswara College of Engineering

Step V –Now we are configured, let's start a build

```
$ 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 cookiecutter
$ pip install pipenv --upgrade-strategy=only-if-needed
$ bash ./test.sh
Courtesy Notice: Pipenv found itself running within a virtual environment, so it will automatically use that environment, instead of creating its own for any project. You can set PIPENV_IGNORE_VIRTUALENVS=1 to force pipenv to ignore that environment and
create its own instead. You can set PIPENV_VERBOSITY=-1 to suppress this warning.
Pipfile.lock not found, creating…
Locking [dev-packages] dependencies…
Locking [packages] dependencies…
Updated Pipfile.lock (ea2c70)!
Installing dependencies from Pipfile.lock (ea2c70)…
     ============================== 0/0 - 00:00:12
+pytest --cov=samplemod --cov=tests --cov-report=term-missing
============================= test session starts ==============================
platform linux -- Python 3.6.7, pytest-5.2.2, py-1.8.0, pluggy-0.13.0
rootdir: /home/travis/build/iancleary/pypackage/pypackage, inifile: pytest.ini
plugins: cov-2.8.1
collected 1 item

tests/test_module.py .                                                   [100%]

============================= warnings summary =================================
/home/travis/virtualenv/python3.6.7/lib/python3.6/distutils/__init__.py:4
  /home/travis/virtualenv/python3.6.7/lib/python3.6/distutils/__init__.py:4: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
    import imp

-- Docs: https://docs.pytest.org/en/latest/warnings.html

---------- coverage: platform linux, python 3.6.7-final-0 ----------
Name                    Stmts   Miss  Cover   Missing
-------------------------------------------------------
samplemod/__init__.py       1      0   100%
samplemod/module.py         3      0   100%
tests/__init__.py           0      0   100%
tests/conftest.py           0      0   100%
tests/test_module.py        4      0   100%
-------------------------------------------------------
TOTAL                       8      0   100%

======================== 1 passed, 1 warnings in 0.13s =========================
+bash ./scripts/lint.sh
+mypy samplemod --disallow-untyped-defs
Success: no issues found in 2 source files
+black samplemod tests --check
All done! ✨ 🍰 ✨
5 files would be left unchanged.
+isort --multi-line=3 --trailing-comma --force-grid-wrap=0 --combine-as --line-width 88 --recursive --check-only --thirdparty samplemod samplemod tests
Courtesy Notice: Pipenv found itself running within a virtual environment, so it will automatically use that environment, instead of creating its own for any project. You can set PIPENV_IGNORE_VIRTUALENVS=1 to force pipenv to ignore that environment and
create its own instead. You can set PIPENV_VERBOSITY=-1 to suppress this warning.
INFO    -  Cleaning site directory
INFO    -  Building documentation to directory: /home/travis/build/iancleary/pypackage/pypackage/site
The command "bash ./test.sh" exited with 0.
store build cache


Done. Your build exited with 0.
```

1.1:Make a simple change to the repo, either a commit, or open a new branch to start a build! Grab your 🍿 and let's watch the CI process start and complete!



1.2: New project added to Travis CI

The three red boxes above are:

1. The connected repo that is currently selected.
2. The action that started a build (commit or pull request)
3. The build matrix visualized, with links to view more details.

Travis CI

Where to go from here?

- Start offloading and automating your testing, deployments, and other activities. There is so many opportunities to tie in different hooks or features, this is just the beginning.

Some examples of what else can you automate:

- **PyPi Publishing**
- **Jupyter Notebook image deployment with CI/CD to Docker Hub**
- **Amazon EC2 instance deployments**

Travis CI