Continuous Integration

CI for research & data science

Leon Reteig (@LeonReteig), UPOD Data Science Meeting, May 22 2020

What is Continuous Integration (CI)?

*A software development practice to **frequently** integrate changes to a project into a remote version*

- Help find problems and conflicts early
- Keep all team members up to date

*Each integration is put through an **automated** build process (e.g. tests)*

- Ensure integration takes place, and does so frequently

Continuous…

- … integration: combine work from individuals into main, shared version
- … delivery: run steps required to build and test the project
- … deployment: run the actual code and build its outputs

The CI process

Each time a commit is pushed to a repository…

1. Clone a copy of the project
2. Recreate its computational environment on a new (virtual) machine
3. Build the project within the new environment
4. Run tests, deploy project, etc.
5. Report the results

CI providers

Hosted (free for open source)

- Travis CI
- GitLab CI/CD
- GitHub Actions

+ easy to use

- less flexible

DIY

- Jenkins

- configure and host yourself

+ more powerful

A toy example

With Github Actions: https://github.com/lcreteig/ci-python-example/blob/master/.github/workflows/test.yml

25 lines (21 sloc) | 531 Bytes — Raw | Blame | History

```yaml
name: Joke

on: [push]

jobs:
  build:

    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version: [2.7, 3.5, 3.6, 3.7, 3.8]

    steps:
    - uses: actions/checkout@v2
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v2
      with:
        python-version: ${{ matrix.python-version }}
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt
    - name: Test with pytest
      run: |
        pytest
```

Specify python versions to test agains

Default action that sets up python environments (actions made by GitHub)

Install dependencies with pip

Test the code with pytest

A (more) real-world example

With Travis: https://github.com/lcreteig/amsterdown/blob/master/.travis.yml

19 lines (16 sloc) | 548 Bytes — Raw | Blame | History

```yaml
# R for travis: see documentation at https://docs.travis-ci.com/user/languages/r

language: R
cache: packages

before_cache:
  - Rscript -e 'remotes::install_cran("pkgdown")' # package for building documentation
after_success:
  - R CMD INSTALL . # install the package so we can use it to...
  - Rscript -e 'pkgdown::build_site()' # build documentation
  - Rscript R/_render.R # render the output samples

deploy:
  provider: pages
  local_dir: docs
  skip_cleanup: true
  keep-history: true
  github_token: $GITHUB_TOKEN
  target_branch: gh-pages
```

before installing

Build and check the package (default, so no need to specify)

Install the package
Build the documentation and example output

Deploy the built docs and examples to website

Applications?

- Running unit tests for your code
- Checking whether your code runs on a machine other than yours
- Building the output of your research: an analysis report, a paper, an ML model, a (Shiny/Dash) app, etc.

Limitations

- CI service needs access to your code/data
- Deployment can take a long time for complex products (huge dataset)

References

- Chapter 14 of “The Turing Way: A Handbook for Reproducible Data Science”: https://the-turing-way.netlify.app

- https://help.github.com/en/actions/language-and-framework-guides/github-actions-for-python

- https://docs.travis-ci.com/user/languages/r/