Scientific Software Development

Inga Ulusoy, Scientific Software Center, Interdisciplinary Center for Scientific Computing, Heidelberg University

March 2022

Unit 6: Continuous integration - GitHub actions

- GitHub actions basics
- Run your tests and linter automatically through GitHub actions
- Build and publish your documentation on readthedocs/GitHub pages
- How to publish your research and your software
- Optionally: Publish your python module as a package

The python module will be completed.
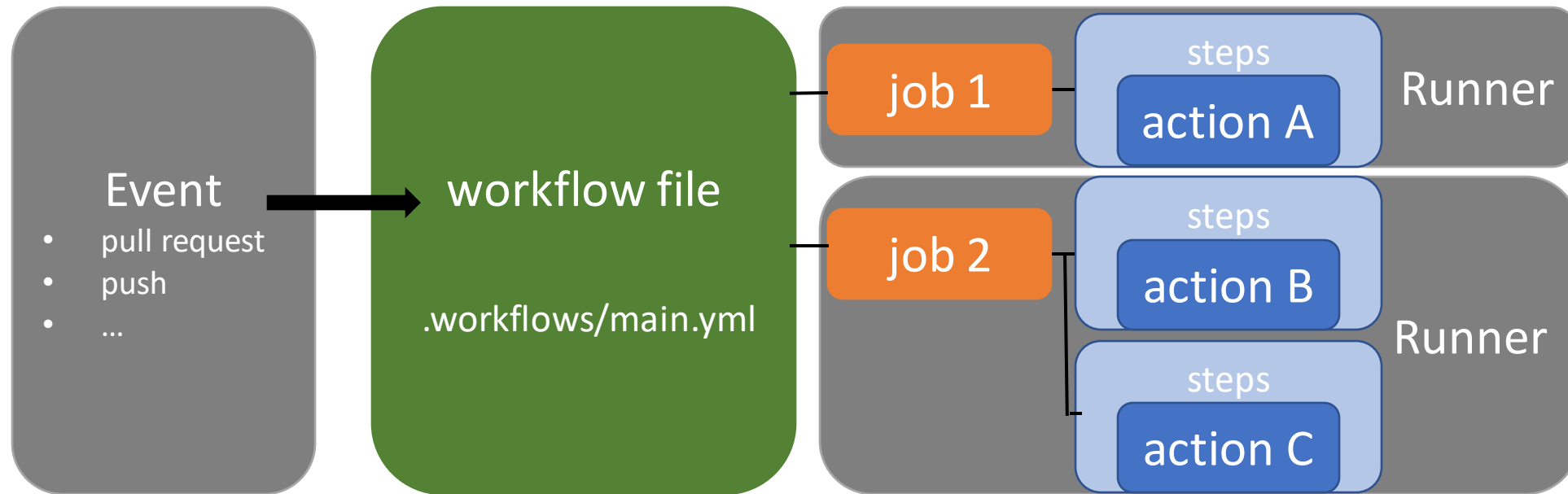Optional: Publish your modules as a package on PyPi.

What are GitHub actions?

GitHub actions are a way to automatize syntax checking and testing upon certain events ("whenever something changes"), i.e. pull requests, merging of branches, etc.

This provides a convenient tool to check your code before it is "unleashed" for a more general use.

You only need to set this up once and it will save you time in the long run.

GitHub actions

Event
- pull request
- push
- …

workflow file

.workflows/main.yml

job 1

steps

action A

Runner

job 2

steps

action B

steps

action C

Runner

GitHub actions

GitHub action pricing:
- Free for public repositories
- For private repositories: ~2000 min/month (execution minutes for hosted runners)
- 1 min actually is 60s on Ubuntu, but: 60s ≙ 2 min on Windows; 60s ≙ 10 min on MacOS

OS	Resources	Price per extra minute
Linux	2 cores, 7 GB	$ 0,008
Windows	2 cores, 7 GB	$0,016
MacOS	2 cores, 7 GB	$ 0,08

The workflow file

- The workflow file is written in YAML (which stands for "YAML Ain't Markup Language") and is a data serialization language; indentation similar to python

```
# This is a basic workflow to help you get started with Actions

name: CI

# Controls when the action will run.
on:
  # Triggers the workflow on push or pull request events but only for the main branch
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  test_and_doc:
    # The type of runner that the job will run on
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ubuntu-18.04, macos-10.15, windows-2019]
        python-version: [3.7, 3.8]
    steps:
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v2
      with:
        python-version: ${{ matrix.python-version }}
    - name: Getting repository
      uses: actions/checkout@v2
```

name of your workflow

triggering event

trigger manually

runners

check out repository action

6

Workflow syntax

```yaml
# This is a basic workflow to help you get started with Actions

name: CI

# Controls when the action will run.
on:
  # Triggers the workflow on push or pull request events but only for the main branch
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  test_and_doc:
    # The type of runner that the job will run on
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ubuntu-18.04, macos-10.15, windows-2019]
        python-version: [3.7, 3.8]
    steps:
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v2
      with:
        python-version: ${{ matrix.python-version }}
    - name: Getting repository
      uses: actions/checkout@v2
```

name of your workflow

triggering event

list

dictionary

trigger manually

runners

check out repository action

list

key: value

list or collection:
- list item 1
- list item 2

my multi-line value: |
    instruction 1
    instruction 2

- understands JSON syntax

7

Workflow syntax

- file needs .yml or .yaml extension
- has to be stored in .github/workflows

`name:` the name of your workflow	`on:` the action that triggers the workflow	`jobs:` the jobs that constitute the workflow

Workflow syntax

```
on: [push, pull_request]
```

```
job:
  job_id:
    name: my job name
    needs: job1 # this ensures job1 is run first
    runs-on: myOS # the architecture that should be used
    steps:
      ...
```

Virtual environment	YAML workflow label
Windows Server 2019	windows-latest or windows-2019
Ubuntu 20.04	ubuntu-latest or ubuntu-20.04
Ubuntu 18.04	ubuntu-latest or ubuntu-18.04
Ubuntu 16.04	ubuntu-16.04
macOS Big Sur 11.0	macos-11.0
macOS Catalina 10.15	macos-latest or macos-10.15

```
# This is a basic workflow to help you get started with Actions

name: CI

# Controls when the action will run.
on:
  # Triggers the workflow on push or pull request events but only for the main branch
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  test_and_doc:
    # The type of runner that the job will run on
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ubuntu-18.04, macos-10.15, windows-2019]
        python-version: [3.7, 3.8]
    steps:
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v2
      with:
        python-version: ${{ matrix.python-version }}
    - name: Getting repository
      uses: actions/checkout@v2
```

Workflow syntax

> **strategy:**
> creates a build matrix for the job to run in

```
# This is a basic workflow to help you get started with Actions

name: CI

# Controls when the action will run.
on:
  # Triggers the workflow on push or pull request events but only for the main branch
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  test_and_doc:
    # The type of runner that the job will run on
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        os: [ubuntu-18.04, macos-10.15, windows-2019]
        python-version: [3.7, 3.8]
    steps:
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v2
      with:
        python-version: ${{ matrix.python-version }}
    - name: Getting repository
      uses: actions/checkout@v2
```

10

Workflow syntax: The actions

- The actions are individual tasks that can be written in different languages
- Write your own or use available ones

```
job:
  job_id:
    name: my job name
    needs: job1 # this ensures job1 is run first
    runs-on: myOS # the architecture that should be used
    steps:
      - name: checkout the repo
        uses: specify an action
```

specify version number of the referenced action
otherwise updates to the action may break your workflow

actions are either JavaScript files or Docker containers
for Docker containers, job must be run in linux environment

https://github.com/actions
https://github.com/marketplace?type=actions
https://hub.docker.com/

relevant actions: `{owner}/{repo}/{path}@{ref}` or `docker://{image}:{tag}`
`actions/checkout@v2` *# checks out your repository on the runner – you will always need this if you run tests/linter*
`actions/setup-python@v2 # sets up python environment`

`sonarsource/sonarcloud-github-action@master # code quality analysis through sonarcloud`

Workflow syntax: run

```
job:
  job_id:
    name: my job name
    needs: job1 # this ensures job1 is run first
    runs-on: myOS # the architecture that should be used
    steps:
      - name: build the documentation
        run: | # run a script, execute a command-line command
          cd doc
          build html
      - name: run the linter
        run: flake8
```

Example running a script using bash
```
steps:
  - name: Display the path
    run: echo $PATH
    shell: bash
```

Example running a script using Windows cmd
```
steps:
  - name: Display the path
    run: echo %PATH%
    shell: cmd
```

Example running a script using PowerShell Core
```
steps:
  - name: Display the path
    run: echo ${env:PATH}
    shell: pwsh
```

Example: Using PowerShell Desktop to run a script
```
steps:
  - name: Display the path
    run: echo ${env:PATH}
    shell: powershell
```

Example running a python script
```
steps:
  - name: Display the path
    run: |
      import os
      print(os.environ['PATH'])
    shell: python
```

GitHub actions

- Fork the sample repo https://github.com/iulusoy/actions-example-iulusoy and adapt it to your unittest example from unit 5.

Unit 6: Continuous integration - GitHub actions

- *GitHub actions basics*
- Run your tests and linter automatically through GitHub actions
- Build and publish your documentation on readthedocs/GitHub pages
- How to publish your research and your software
- Optionally: Publish your python module as a package

The python module will be completed.
Optional: Publish your modules as a package on PyPi.

Linter in GitHub actions

- Now using the sample repository that you just forked, add the following lines to the workflow file in `.github/workflows/main.yml`:

```
- name: Run linter
  run: flake8
```

Take care of proper indentation! Yaml syntax is very strict.

Linter in GitHub actions

- Trigger the workflow and see what happens.

Unit tests in GitHub actions

- Add the following lines to your GitHub actions file:

```
- name: Run unittest
    run: |
      cd src
      python -m unittest
```

- Take care of proper indentation!

Unit tests in GitHub actions

- Trigger the workflow and see what happens.

Linter and tests in GitHub actions

- Meddle with your code so that the linter/unit tests will fail. Commit to a branch and open a Pull Request. What happens?

Unit 6: Continuous integration - GitHub actions

- *GitHub actions basics*

- *Run your tests and linter automatically through GitHub actions*

- Build and publish your documentation on readthedocs/GitHub pages

- How to publish your research and your software

- Optionally: Publish your python module as a package

The python module will be completed.
Optional: Publish your modules as a package on PyPi.

Documentation in GitHub actions

- Add the following lines to your workflow file:

```
- name: Build sphinx documentation
  run: |
      cd doc
      make html
```

Take care of proper indentation!

Documentation in GitHub actions

- Open a PR and see what happens. Check out the documentation page on readthedocs/GitHub actions.

Unit 6: Continuous integration - GitHub actions

- *GitHub actions basics*
- *Run your tests and linter automatically through GitHub actions*
- *Build and publish your documentation on readthedocs/GitHub pages*
- How to publish your research and your software
- Optionally: Publish your python module as a package

The python module will be completed.
Optional: Publish your modules as a package on PyPi.

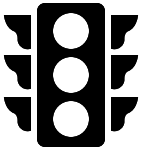# How to publish research and software

Scenario 1:
You publish your research but not the data nor the software.
Data availability statement:
".. data is available from the authors upon reasonable request..."

Please consider to publish a preprint on a preprint server like arxiv (after submission of your paper to a journal but before its publication)

How to publish research and software

Scenario 1:
You publish your research but not the data nor the software.
Data availability statement:
".. data is available from the authors upon reasonable request..."

Scenario 2:
You publish your research and data but not the software.
Data availability statement:
".. data is available at DOI..."

Please consider to publish a preprint on a preprint server like arxiv (after submission of your paper to a journal but before its publication)

- publish preprint
- publish data on a platform like zenodo, and obtain a DOI

How to publish research and software

Scenario 3:
You publish your research, data and software.
Data availability statement:
".. data and software is available at DOI ..."

- publish preprint
- publish data with DOI
- publish software with DOI (ie zenodo)

How to publish research and software

Scenario 3:
You publish your research, data and software.
Data availability statement:
".. data and software is available at DOI ..."

- publish preprint
- publish data with DOI
- publish software with DOI (ie zenodo)

Scenario 4:
You publish your research and data separate from the software, both in a journal/data in a database.
Data availability statement:
".. data is available at DOI..."
Software is referenced via its publication.

- publish preprint
- publish data with DOI
- publish software in dedicated, peer-review journal*

*List of possible journals: https://www.software.ac.uk/which-journals-should-i-publish-my-software

Unit 6: Continuous integration - GitHub actions

- *GitHub actions basics*

- *Run your tests and linter automatically through GitHub actions*

- *Build and publish your documentation on readthedocs/GitHub pages*

- *How to publish your research and your software*

- Optionally: Publish your python module as a package

The python module will be completed.
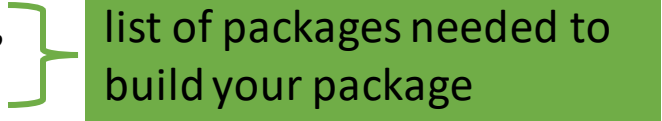Optional: Publish your modules as a package on PyPi.

Publish a python package on PyPi

- Work through:
- https://packaging.python.org/tutorials/packaging-projects/

Publish a python package

- You need a file __init__.py in your package source directory so that the directory can be imported as a package

- Unit tests are in tests/

- Create the file pyproject.toml - this file communicates with build tools like pip and build

```
[build-system]
requires = [
    "setuptools>=42",
    "wheel"
]
build-backend =
"setuptools.build_meta"
```

list of packages needed to build your package

Configure the metadata

- Static metadata setup.cfg: Always the same. Try to keep it static rather than dynamic.

- Dynamic metadata setup.py: Determined at install-time. Only use when absolutely necessary.

Live lesson

- In the live lesson, we will set up GitHub actions for your python package.

Live lesson - Demonstrations

- The following demonstrations will take place in the beginning of the live session:
  - This will be decided by the participants.