

GitHub Actions Demystified

Data Club on 29.04.2025
by Henry Webel

Thanks to Pieter Verschaffelt for sharing his slides which these are built upon.

GitHub Actions Demystified

In this DataClub we will look at GitHub Actions. GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline. Therefore, it can help bioinformaticians and data scientists to keep their software working by identifying bugs or dependency conflicts when your code or code you depend on changes. It also avoids that you accidentally forget certain checks you would normally perform. Therefore, if you are tired of repetitive tasks in your daily coding work or need more security, GitHub Actions can be the solution.

We will focus on introducing the core concepts of workflows and jobs in a hands-on manner using a simple hello-world example. Finally, we will show some more elaborated

examples as an outlook for interested users.

Prerequisite is that you have a GitHub account and some familiarity with the command line. We do the tutorial on GitHub Codespaces using VSCode (see [intro](#)).

It will make most sense to attend if you want to use GitHub Actions (or similar on GitLab or BitBucket) on repositories where you save your package or analysis code. Then it will be easy for you after this Data club to add some basic checks, e.g. format checkers or lint checkers for common errors or pitfalls of your code.

Expectations (each bullet point)

- Included
 - How to set up and use Actions
 - How to automate
 - General knowledge
- Maybe:
 - Best practices
 - Collaborative spaces
 - Practical examples on how to apply them on programming projects.
 - Template for easy implementation
 - How to integrate into my data collection workflow
 - Which servers can work with GitHub libraries?
 - How to keep the GitHub repositories synchronized in your server?
- Out-of-scope
 - when my collaborators change the code, what should I do? (Pull-Requests?)
 - Testing and deploying scripts to Azure VM

Program overview

1. Introduction to GitHub Actions
2. First example
3. Yaml format (optional)
4. Workflow file in detail
5. *Vuegen repo: extended example (if time allows)*

What is GitHub Actions?

“GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline.”

Source: docs.github.com

<https://www.youtube.com/watch?v=URmeTqglS58>

What is GitHub Actions?

- Scripts that run based on an event ('trigger') or at specific times
- Many possibilities
 - ▶ Perform unit testing and integration testing
 - ▶ Create new releases
 - ▶ Publishing python packages to PyPI
 - ▶ ... everything you would normally do manually on your computer
- Can be used to protect PRs
 - ▶ PRs need specific checks to succeed before they can be merged

Is this free? Who can use GitHub Actions?

Pricing plans

Plan	Private repositories, per month	
	Storage	Minutes
GitHub Free	500MB	2 000
GitHub Pro* (Free for students)	1GB	3 000
GitHub Team	2GB	3 000
GitHub Enterprise Cloud	50GB	50 000

Minute multipliers

Operating System	Minute multiplier
Linux	1
Windows	2
macOS	10

"GitHub Actions usage is free for standard GitHub-hosted runners in public repositories, and for self-hosted runners."

[Source: About billing for GitHub Actions](#)

Key concepts and terminology

Workflow

A configurable process that is defined by a YAML file inside of your repository and performs one or more specific actions (jobs) after a trigger.



- A single yaml file (simplified)

Key concepts and terminology

Workflow

A configurable process that is defined by a YAML file inside of your repository and performs one or more specific actions (jobs) after a trigger.



Trigger or Event

A specific event that automatically initiates a workflow within a GitHub repository.



Key concepts and terminology

Job

A workflow can have many jobs, which can be connected as a directed acyclic graph (DAG).

Step

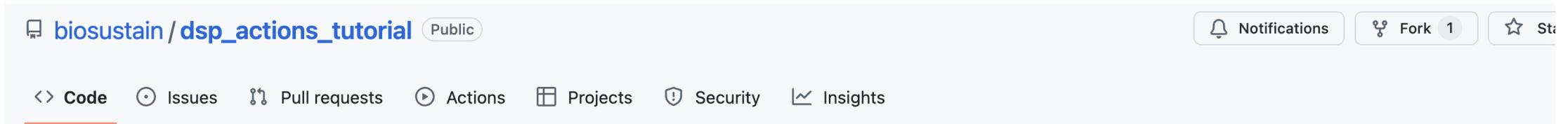
A single command or block of commands (small script) in a **job**.

Program overview

1. Introduction to GitHub Actions
2. First example
3. Yaml format (optional)
4. Workflow file in detail
5. *Vuegen repo: extended example (if time allows)*

First example

github.com/biosustain/dsp_actions_tutorial

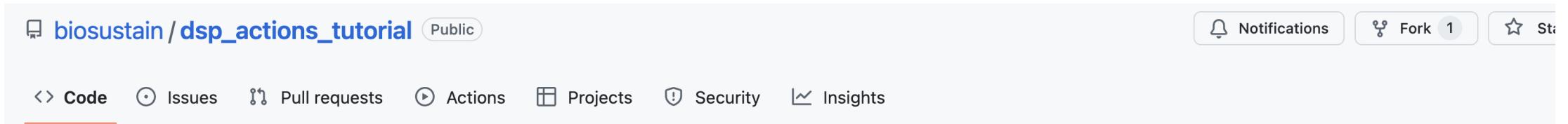
A screenshot of a GitHub repository page for 'biosustain / dsp_actions_tutorial'. The page shows a green 'Code' button highlighted with a red underline. Below it, there are sections for branches ('main'), tags ('0 Tags'), and files ('README', '.github/workflows'). A commit history is shown with two commits from user 'enryH' made 3 days ago. The commit messages mention 'Simple example and some hints (#1)'. On the right side, there's an 'About' section with a description of the repository as a 'Simple GitHub Actions example for the Data Club on the 30th of April at DTU biosustain and DTU bioengineering'. It also lists repository statistics: 0 stars, 2 watching, 1 fork, and a 'Report repository' link.

About

Simple GitHub Actions example for the Data Club on the 30th of April at DTU biosustain and DTU bioengineering

 Readme
 Activity
 Custom properties
 0 stars
 2 watching
 1 fork
Report repository

Releases

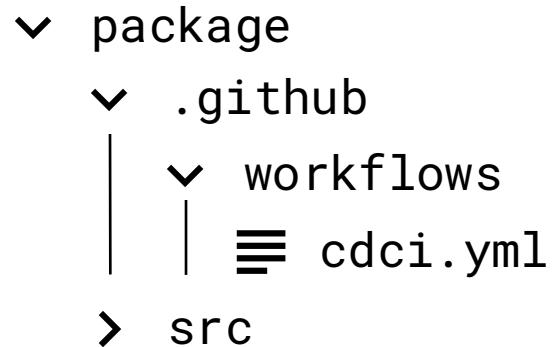
A screenshot of a GitHub repository page for 'biosustain / dsp_actions_tutorial'. The page shows a green 'Code' button highlighted with a red underline. Below it, there are sections for branches ('main'), tags ('0 Tags'), and files ('README', '.github/workflows'). A commit history is shown with two commits from user 'enryH' made 3 days ago. The commit messages mention 'Simple example and some hints (#1)'. On the right side, there's an 'About' section with a description of the repository as a 'Simple GitHub Actions example for the Data Club on the 30th of April at DTU biosustain and DTU bioengineering'. It also lists repository statistics: 0 stars, 2 watching, 1 fork, and a 'Report repository' link.

Data Club on GitHub Actions - Simple example

by Henry Webel from the Data Science Platform at DTU biosustain

Directory structure of a GitHub repository

- All workflows live in the ` `.github/workflows` folder
- Each workflow is defined in a separate ` .yml` file



Program overview

1. Introduction to GitHub Actions
2. First example
3. Yaml format (optional)
4. Workflow file in detail
5. *Vuegen repo: extended example (if time allows)*

YML?

Yet Another Markup Language

- Simple format used to describe a GitHub Action (workflow)
- Human-readable
 - ▶ Designed to be easily read and written by humans
- Use indentation (spaces) to denote structure
- Key-Value pairs
 - ▶ Each 'key: value' pair represents data
 - ▶ Values can be lists of items
- Plus some syntactic sugar for actions, e.g. |

YML?

Lists

```
● ● ●  
  
items:  
  - item1  
  - item2
```

Key-value pairs and nested structures

```
● ● ●  
  
person:  
  name: John Doe  
  age: 30  
  hobbies:  
    - reading  
    - cycling
```

Program overview

1. Introduction to GitHub Actions
2. First example
3. Yaml format (optional)
4. Workflow file in detail
5. *Vuegen repo: extended example (if time allows)*

Hello World example

```
.github > workflows > hello_world.yml
1  # Hello World example for DTU biosustain and DTU bioengineering Data Club
2
3  name: Hello World example
4
5  on:
6    push:
7
8  jobs:
9    say-hello-world:
10      runs-on: ubuntu-latest
11      steps:
12        - name: biosustain
13          run: echo "Hello DTU biosustain!"
14        - name: bioengineering
15          run: echo "Hello DTU bioengineering!"
```

Hello World example

```
.github > workflows > hello_world.yml
1  # Hello World example for DTU biosustain and DTU bioengineering Data Club
2
3  name: Hello World example
4
5  on:
6    push:
7
8  jobs:
9    say-hello-world:
10   runs-on: ubuntu-latest
11   steps:
12     - name: biosustain
13       run: echo "Hello DTU biosustain!"
14     - name: bioengineering
15       run: echo "Hello DTU bioengineering!"
```

Name: (required)

Describe your action in a few words. This will be used by GitHub and help you to easily find a specific action back.

Hello World example

```
.github > workflows > hello_world.yml
1  # Hello World example for DTU biosustain and DTU bioengineering Data Club
2
3  name: Hello World example
4
5  on:
6    | push:
7
8  jobs:
9    say-hello-world:
10   | runs-on: ubuntu-latest
11   | steps:
12   |   - name: biosustain
13   |     run: echo "Hello DTU biosustain!"
14   |   - name: bioengineering
15   |     run: echo "Hello DTU bioengineering!"
```

Triggers: (required)

After which event should this action be executed? Multiple triggers can be provided simultaneously.

Hello World example

```
.github > workflows > hello_world.yml
1  # Hello World example for DTU biosustain and DTU bioengineering Data Club
2
3  name: Hello World example
4
5  on:
6    push:
7
8  jobs: jobs:
9    say-hello-world:
10   runs-on: ubuntu-latest
11   steps:
12     - name: biosustain
13       run: echo "Hello DTU biosustain!"
14     - name: bioengineering
15       run: echo "Hello DTU bioengineering!"
```

Jobs: (required)

A list of actions that should be executed by this workflow. At least one action is required!

Hello World example

```
.github > workflows > hello_world.yml
 1 # Hello World example for DTU biosustain and DTU bioengineering Data Club
 2
 3 name: Hello World example
 4
 5 on:
 6   push:
 7
 8 jobs:
 9   say-hello-world:
10     runs-on: ubuntu-latest
11     steps:
12       - name: biosustain
13         run: echo "Hello DTU biosustain!"
14       - name: bioengineering
15         run: echo "Hello DTU bioengineering!"
```

Job name

Hello World example

```
.github > workflows > hello_world.yml
1  # Hello World example for DTU biosustain and DTU bioengineering Data Club
2
3  name: Hello World example
4
5  on:
6    push:
7
8  jobs:
9    say-hello-world:
10      runs-on: ubuntu-latest
11      steps:
12        - name: biosustain
13          run: echo "Hello DTU biosustain!"
14        - name: bioengineering
15          run: echo "Hello DTU bioengineering!"
```

Runner: (required)

Which operating system
should this action be executed
on? (e.g. ubuntu, windows,
macos, ...)

Standard GitHub-hosted runners

[about GitHub hosted runners](#)

Virtual Machine	Processor (CPU)	Memory (RAM)	Storage (SSD)	Architecture	Workflow label
Linux	4	16 GB	14 GB	x64	ubuntu-latest , ubuntu-24.04 , ubuntu-22.04 , ubuntu-20.04
Windows	4	16 GB	14 GB	x64	windows-latest , windows-2025 [Public preview], windows-2022 , windows-2019
Linux [Public preview]	4	16 GB	14 GB	arm64	ubuntu-24.04-arm , ubuntu-22.04-arm
macOS	4	14 GB	14 GB	Intel	macos-13
macOS	3 (M1)	7 GB	14 GB	arm64	macos-latest , macos-14 , macos-15 [Public preview]

Hello World example

```
.github > workflows > hello_world.yml
1  # Hello World example for DTU biosustain and DTU bioengineering Data Club
2
3  name: Hello World example
4
5  on:
6    push:
7
8  jobs:
9    say-hello-world:
10   runs-on: ubuntu-latest
11   steps:
12     - name: biosustain
13       run: echo "Hello DTU biosustain!"
14     - name: bioengineering
15       run: echo "Hello DTU bioengineering!"
```

Steps: (required)
What does the action need to do? Steps are executed one-by-one. If a step fails, subsequent steps will not be executed.

← Hello World example

  add exercises #13

 Summary

Jobs

 say-hello-world

Run details

 Usage

 Workflow file

say-hello-world

succeeded yesterday in 3s

>  Set up job

>  biosustain

✓  bioengineering

1 ► Run echo "Hello DTU bioengineering!"

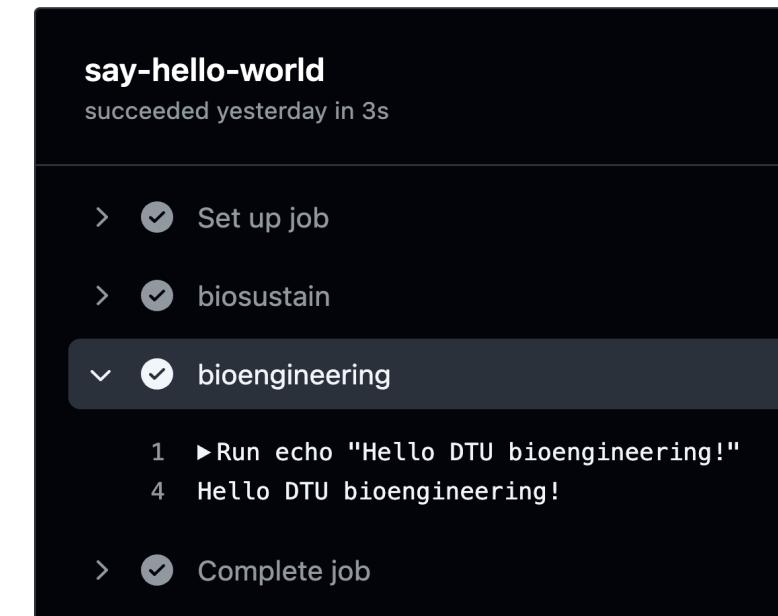
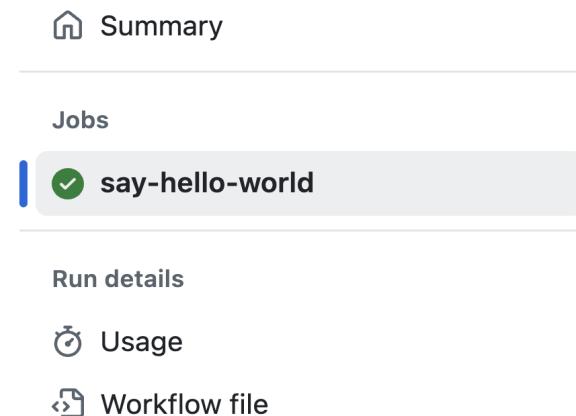
4 Hello DTU bioengineering!

>  Complete job

What can or cannot see in run summary?

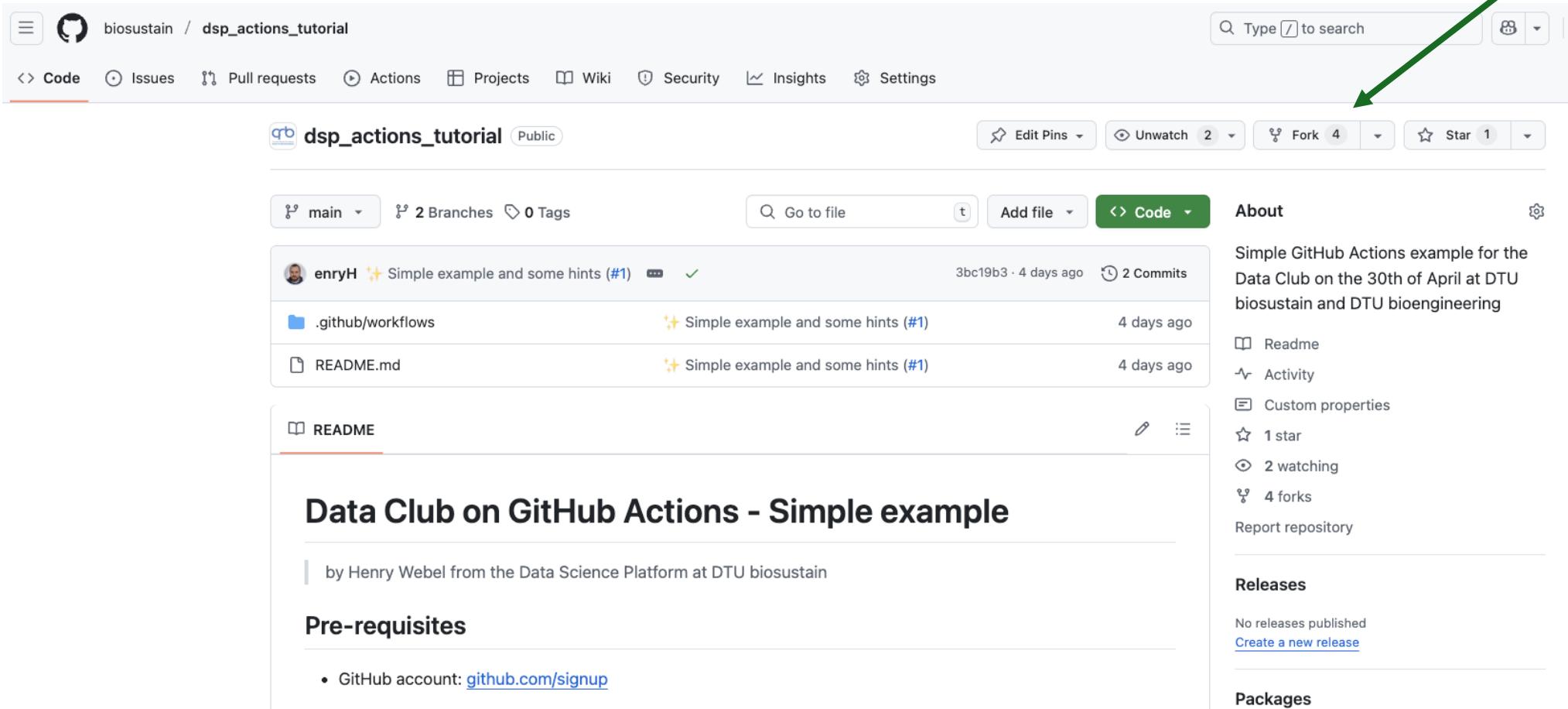
```
.github > workflows > hello_world.yml
1  # Hello World example for DTU biosustain and DTU bioengineering Data Club
2
3  name: Hello World example
4
5  on:
6    push:
7
8  jobs:
9    say-hello-world:
10      runs-on: ubuntu-latest
11      steps:
12        - name: biosustain
13          run: echo "Hello DTU biosustain!"
14        - name: bioengineering
15          run: echo "Hello DTU bioengineering!"
```

← Hello World example
✓ add exercises #13



Fork the repository

github.com/biosustain/dsp_actions_tutorial



The screenshot shows the GitHub repository page for 'dsp_actions_tutorial'. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, the repository name 'dsp_actions_tutorial' is displayed, along with its status as 'Public'. To the right of the repository name are buttons for Edit Pins, Unwatch, Fork (with a count of 4), and Star (with a count of 1). A green arrow points to the 'Fork' button. The main content area shows the repository's structure: a 'main' branch, 2 branches, and 0 tags. It lists three commits from user 'enryH': one commit to '.github/workflows' and two commits to 'README.md', all labeled 'Simple example and some hints (#1)'. Below this, there's a 'README' section with the heading 'Data Club on GitHub Actions - Simple example' and a note that it's by Henry Webel from the Data Science Platform at DTU biosustain. The 'Pre-requisites' section lists 'GitHub account: github.com/signup'. On the right side of the page, there's an 'About' section with a detailed description of the repository, including its purpose as a simple GitHub Actions example for the Data Club, and sections for Readme, Activity, Custom properties, and repository statistics (1 star, 2 watching, 4 forks). There are also sections for Releases and Packages.

Create fork

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk ().*

Owner *



enryH

Repository name *

dsp_actions_tutorial

dsp_actions_tutorial is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Simple GitHub Actions example for the Data Club on the 30th of April at DTU biosustain and DTU bioengir

Copy the `main` branch only

Contribute back to biosustain/dsp_actions_tutorial by adding your own branch. [Learn more.](#)

You are creating a fork in your personal account.

Create fork

Hands-on: Setup Code-Space on your fork

https://github.com/biosustain/dsp_actions_tutorial

The screenshot shows a GitHub repository page for 'dsp_actions_tutorial'. The top navigation bar includes 'Edit Pins', 'Unwatch', 'Fork' (with 1 fork), and 'Star' (0 stars). A green arrow points from the text above to the 'Fork' button. The main content area shows a file tree on the left with 'main' checked out, containing files like '.github/workflows', 'README.md', and 'README'. A large modal window is open in the center, titled 'Data Club on GitHub example'. It has tabs for 'Local', 'Codespaces', and 'Copilot', with 'Codespaces' selected. The 'Codespaces' section displays the message 'Your workspaces in the cloud' and 'No codespaces'. It features a prominent green 'Create codespace on main' button and a link to 'Learn more about codespaces...'. At the bottom of the modal, it says 'Codespace usage for this repository is paid for by enryH.' and has a 'Calendar' link. The right sidebar contains sections for 'About', 'Readme', 'Activity', 'Custom properties', '0 stars', '2 watching', '1 fork', 'Report repository', and 'Releases'.

Exercises

```
# first job
## install step:
pip install cowsay
## use it
cowsay -c cheese -t 'hey there'

# second job
## https://github.com/jeffbuttar/cowpy
pip install cowpy

cowpy -c tux hi there
```

Say yes if you see the pop-Up: Install GitHub Actions Extension?
- also to git fetch etc.

- *First:* Add **two jobs** which use either cowsay or cowpy
 - Separate installation and usage into separate **steps**
- *Medium:* Call it from a script (to understand file structure)
 - You will need the files!
 - Copy examples from [cowpy README.md](#)
- *Advanced:* Build a directed acyclic graph combining jobs
 - Look at the extended examples in the [README](#) to figure this out

One Solution

- Add bin/cowpy_examples.py to repository
 - Need to check repository out
- Does it also run on windows or Mac?
- Caveat: Shell behaviour of (Python) libraries: see cowpy step for command line execution. Check default shell configuration on runners [here](#) and warnings [here](#)

```
3   name: Hello World example
4
5   ▼ on:
6     | push:
7     | pull_request:
8
9   ▼ jobs:
10  > say-hello-world:...
11
12  ▼ cowsay-test:
13    runs-on: ubuntu-latest
14    steps:
15      - name: install cowsay
16        run: |
17          pip install cowsay
18
19      - name: run cowsay
20        run: cowsay -c cheese -t 'hey there'
21
22  ▼ cowpy-test:
23    runs-on: ubuntu-latest
24    steps:
25      - uses: actions/checkout@v4
26      - name: install cowpy
27        run: |
28          pip install cowpy
29
30      - name: run cowpy
31        run: |
32          cowpy -c tux "hola mundo"
33
34          echo "hola mundo" | cowpy -c tux
35
36      - name: run script with examples
37        run: |
38          python bin/cowpy_examples.py
```

Predefined actions

- Predefined action contain a collection of steps and have arguments ('with')
- Most common two:
 - Checkout to get the repository files
 - Setup-python to install a specific Python version for testing

```
cowpy-test:  
  runs-on: ubuntu-latest  
  defaults:  
    run:  
      shell: bash -el {0}  
  steps:  
    - uses: actions/checkout@v4  
    - uses: actions/setup-python@v5  
      with:  
        python-version: '3.12'  
    - name: install cowpy  
      run: |  
        pip install cowpy  
    - name: run cowpy  
      run: |  
        cowpy -c tux hola mundo  
        echo "hola mundo" | cowpy -c tux  
    - name: run script with examples  
      run: |  
        python bin/cowpy_examples.py
```

Program overview

1. Introduction to GitHub Actions
2. First example
3. Yaml format (optional)
4. Workflow file in detail
5. *Vuegen repo: extended example (if time allows)*

VueGen

github.com/Multiomics-Analytics-Group/vuegen

The screenshot shows the GitHub repository page for 'vuegen'. The repository is owned by 'Multiomics-Analytics-Group' and has a public status. It features 21 issues, 6 pull requests, and 1 project. The 'Code' tab is selected. The repository has 9 branches and 10 tags. A recent commit by 'sayalaruano' titled 'Api and chatbot components update (#107)' is highlighted, showing changes in '.github/workflows', 'docs', 'gui', 'src/vuegen', 'tests', '.gitignore', '.readthedocs.yaml', 'CITATION.cff', 'CONTRIBUTING.md', and 'LICENSE'. The commit was made 2 weeks ago and includes 183 commits. The 'About' section describes VueGen as automating report creation from bioinformatics outputs in various formats like PDF, HTML, DOCX, ODT, PPTX, Reveal.js, Jupyter notebooks, and Streamlit web applications. It also lists tags: python, data-visualization, reports, developer-tools, quarto, and streamlit. Other repository details include a Readme, MIT license, citation information, activity, custom properties, and 15 stars.

Multiomics-Analytics-Group / vuegen

Type to search

Code Issues 21 Pull requests 6 Discussions Actions Projects 1 Security Insights Settings

vuegen Public

Edit Pins Unwatch 2 Fork 0 Starred 15

main 9 Branches 10 Tags Go to file Add file Code

sayalaruano Api and chatbot components update (#107) ec43047 · 2 weeks ago 183 Commits

File	Description	Time
.github/workflows	Add Quarto checks and output dire parameters	last month
docs	Api and chatbot components update (#107)	2 weeks ago
gui	Add Quarto checks and output dire parameters	last month
src/vuegen	Api and chatbot components update (#107)	2 weeks ago
tests	ensure a valid python identifier for report_manger.py st...	2 weeks ago
.gitignore	Docs: update docker image name on README	last month
.readthedocs.yaml	readthedocs configuration (#48)	3 months ago
CITATION.cff	Add info into the CITATION.cff file	last month
CONTRIBUTING.md	Docs: fix broken link contributing file	last month
LICENSE	Initial commit	2 years ago

<https://github.com/Multiomics-Analytics-Group/vuegen>

About

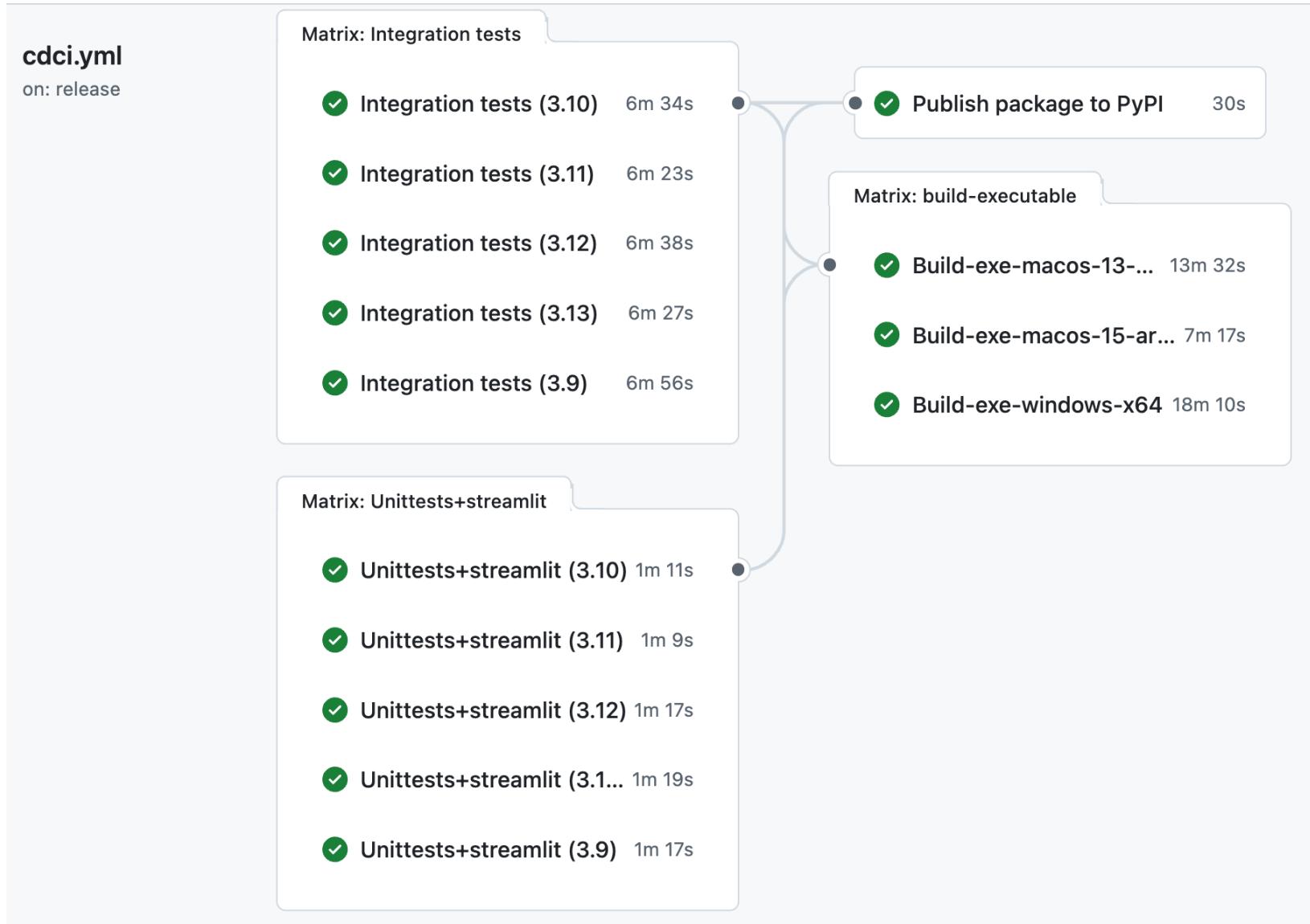
VueGen automates the creation of reports from bioinformatics outputs, supporting formats like PDF, HTML, DOCX, ODT, PPTX, Reveal.js, Jupyter notebooks, and Streamlit web applications. Users simply provide a directory with output files and VueGen compiles them into a structured report.

[vuegen.readthedocs.io](#)

python data-visualization reports developer-tools quarto streamlit

Readme MIT license Cite this repository Activity Custom properties 15 stars

Vuegen release 0.3.1



Python package template repository

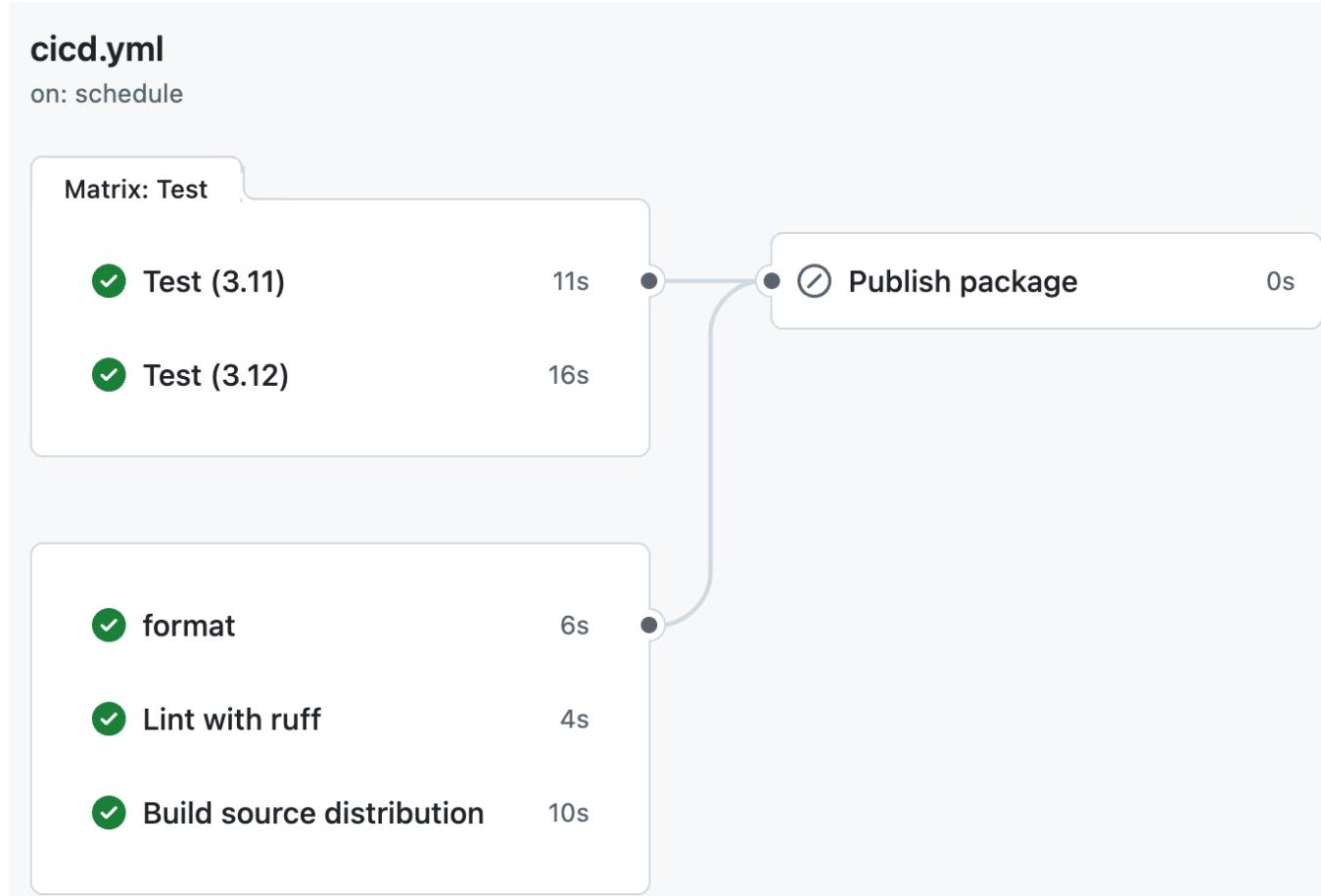
- Python project example
- Lint and format

github.com/RasmussenLab/python_package

The screenshot shows a GitHub repository page for 'python_package'. The top navigation bar includes 'Edit Pins', 'Unwatch', 'Fork', 'Star', and a green 'Use this template' button. Below the header, there's a search bar, a 'Code' button, and a dropdown for 'main'. It displays 3 branches and 1 tag. The main content area shows a list of commits from user 'enryH' over the past 5 months. The commits include changes to workflows, documentation, tests, and file configurations like .gitignore and .readthedocs.yaml. On the right side, there's an 'About' section with a 'Readme' link, an 'MIT license' link, activity stats (1 star, 1 watching, 0 forks), and a 'Report repository' link.

Commit	Message	Date
enryH	Make notebook colab startable again...	5 months ago
.github/workflows	Change to py:percent notebooks for do...	6 months ago
docs	Make notebook colab startable again...	5 months ago
src/mockup	Cicd (#4)	last year
tests	Add tests	2 years ago
.gitignore	Change to py:percent notebooks for do...	6 months ago
.readthedocs.yaml	Readthedocs configuration	last year
LICENSE	Initialize repository	2 years ago

First example



Acknowledgments and Hints

- Pieter Verschaffelt, Workshop on EuBIC Winterschool 2024
 - ▶ New Winterschool in 2026 in Czech!
- Check out VueGen!
- Remember to delete your codespaces: github.com/codespaces

More material:

- <https://book.the-turing-way.org/reproducible-research/ci>

The End – Happy coding!

Python package template repository

- Python project example
- Lint and format

[RasmussenLab/python_package](#)

The screenshot shows a GitHub repository page for 'python_package'. The top navigation bar includes 'Edit Pins', 'Unwatch', 'Fork', 'Star', and a green 'Use this template' button. Below the header, there are buttons for 'main', '3 Branches', '1 Tag', a search bar 'Go to file', and a 'Code' dropdown. The main content area displays a list of commits from user 'enryH':

- Make notebook colab startable again... - 5 months ago
- .github/workflows - Change to py:percent notebooks for do... - 6 months ago
- docs - Make notebook colab startable again... - 5 months ago
- src/mockup - Cicd (#4) - last year
- tests - Add tests - 2 years ago
- .gitignore - Change to py:percent notebooks for do... - 6 months ago
- .readthedocs.yaml - Readthedocs configuration - last year
- LICENSE - Initialize repository - 2 years ago

On the right side, the 'About' section provides details about the repository:

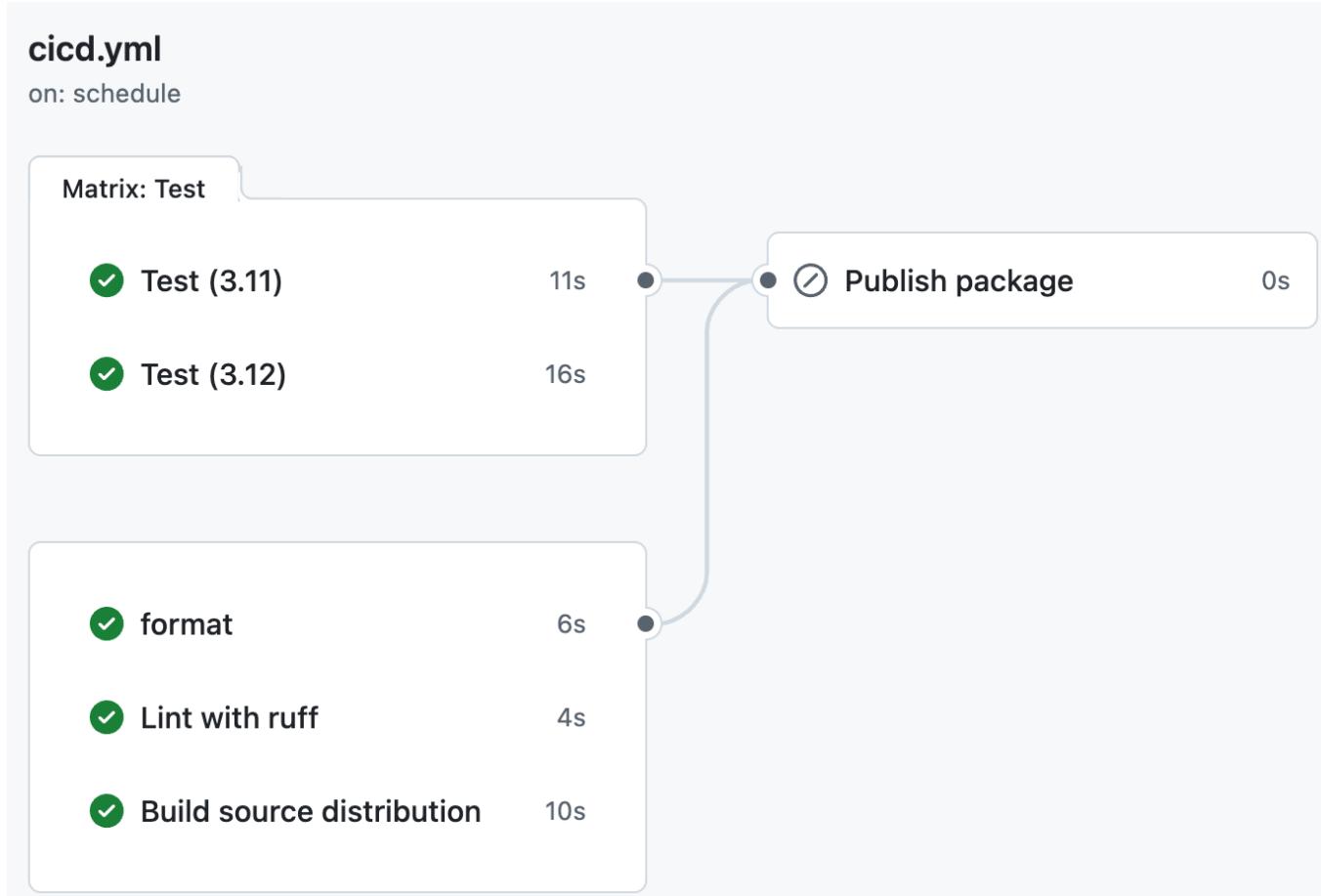
Example Python package
rasmussenlab-python-package.readth...

python template-project

- Readme
- MIT license
- Activity
- Custom properties
- 1 star
- 1 watching
- 0 forks

Report repository

CICD action



Workflow in Detail

```
python_package > .github > workflows > %a cicd.yml
1 # This workflow will install Python dependencies, run tests and lint with a single version of Python
2 # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
3
4 name: Python application
5
6 on:
7   push:
8   pull_request:
9     branches: [ "main" ]
10  schedule:
11    - cron: '0 2 * * 3'
12
13 permissions:
14   contents: read
15
16
17 jobs:
18   format:
19     runs-on: ubuntu-latest
20     steps:
21       - uses: actions/checkout@v4
22       - uses: psf/black@stable
23   lint:
24     name: Lint with ruff
25     runs-on: ubuntu-latest
26     steps:
27       - uses: actions/checkout@v4
28
29       - uses: actions/setup-python@v5
30         with:
31           python-version: "3.11"
32       - name: Install ruff
33         run: |
34           pip install ruff
35       - name: Lint with ruff
36         run: |
37           # stop the build if there are Python syntax errors or undefined names
38           ruff check .
```

Focus on jobs

```
python_package > .github > workflows > cicd.yml
1  # This workflow will install Python dependencies, run tests and lint with a single version of Python
2  # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
3
4  name: Python application
5
6  > on: ...
12
13 > permissions: ...
15
16 > jobs:
17   format:
18     runs-on: ubuntu-latest
19     steps:
20       - uses: actions/checkout@v4
21       - uses: psf/black@stable
22     lint:
23       name: Lint with ruff
24       runs-on: ubuntu-latest
25       steps:
26         - uses: actions/checkout@v4
27
28         - uses: actions/setup-python@v5
29         with:
30           python-version: "3.11"
31         - name: Install ruff
32         run:
33           pip install ruff
34         - name: Lint with ruff
35         run:
36           # stop the build if there are Python syntax errors or undefined names
37             ruff check .
```

Workflow in Detail

```
python_package > .github > workflows > cicd.yml
 1 # This workflow will install Python dependencies, run tests and lint with a single version of Python
 2 # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
 3
 4 name: Python application
 5
 6 > on: ...
12
13 > permissions: ...
15
16 jobs:
17   format:
18     runs-on: ubuntu-latest
19     steps:
20       - uses: actions/checkout@v4
21       - uses: psf/black@stable
22
23   lint:
24     name: Lint with ruff
25     runs-on: ubuntu-latest
26     steps:
27       - uses: actions/checkout@v4
28
29       - uses: actions/setup-python@v5
30         with:
31           python-version: "3.11"
32       - name: Install ruff
33         run: |
34             pip install ruff
35       - name: Lint with ruff
36         run: |
37             # stop the build if there are Python syntax errors or undefined names
             ruff check .
```

Name: (required)

Describe your action in a few words. This will be used by GitHub and help you to easily find a specific action back.

Workflow in Detail

```
python_package > .github > workflows > 8a_cicd.yml
 1 # This workflow will install Python dependencies, run tests and lint with a single version of Python
 2 # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
 3
 4 name: Python application
 5
 6 on:
 7   push:
 8   pull_request:
 9     branches: [ "main" ]
10   schedule:
11     - cron: '0 2 * * 3'
12
13 permissions:
14   contents: read
15
16
17 jobs:
18   format:
19     runs-on: ubuntu-latest
20     steps:
21       - uses: actions/checkout@v4
22       - uses: psf/black@stable
23   lint:
24     name: Lint with ruff
25     runs-on: ubuntu-latest
26     steps:
27       - uses: actions/checkout@v4
28
29       - uses: actions/setup-python@v5
30         with:
31           python-version: "3.11"
32       - name: Install ruff
33         run:
34           pip install ruff
35       - name: Lint with ruff
36         run:
37           # stop the build if there are Python syntax errors or undefined names
38           ruff check .
```

Triggers: (required)

After which event should this action be executed? Multiple triggers can be provided simultaneously.

Workflow in Detail

```
python_package > .github > workflows > cicd.yml
 1  # This workflow will install Python dependencies, run tests and lint with a single version of Python
 2  # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
 3
 4  name: Python application
 5
 6  > on: ...
12
13  > permissions: ...
15
16  > jobs: jobs:
17    format:
18      runs-on: ubuntu-latest
19      steps:
20        - uses: actions/checkout@v4
21        - uses: psf/black@stable
22    lint:
23      name: Lint with ruff
24      runs-on: ubuntu-latest
25      steps:
26        - uses: actions/checkout@v4
27
28        - uses: actions/setup-python@v5
29          with:
30            python-version: "3.11"
31        - name: Install ruff
32          run: |
33            pip install ruff
34        - name: Lint with ruff
35          run: |
36            # stop the build if there are Python syntax errors or undefined names
37            ruff check .
```

Jobs: (required)

A list of actions that should be executed by this workflow. At least one action is required!

Workflow in Detail

```
python_package > .github > workflows > cicd.yml
1  # This workflow will install Python dependencies, run tests and lint with a single version of Python
2  # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
3
4  name: Python application
5
6  > on: ...
12
13 > permissions: ...
15
16 > jobs:
17   format:
18     runs-on: ubuntu-latest
19     steps:
20       - uses: actions/checkout@v4
21       - uses: psf/black@stable
22   lint:
23     name: Lint with ruff
24     runs-on: ubuntu-latest
25     steps:
26       - uses: actions/checkout@v4
27
28       - uses: actions/setup-python@v5
29         with:
30           python-version: "3.11"
31         name: Install ruff
32         run:
33           pip install ruff
34         name: Lint with ruff
35         run:
36           # stop the build if there are Python syntax errors or undefined names
37             ruff check .
```

Job name

Workflow in Detail

```
python_package > .github > workflows > cicd.yml
1  # This workflow will install Python dependencies, run tests and lint with a single version of Python
2  # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
3
4  name: Python application
5
6  > on: ...
12
13 > permissions: ...
15
16 > jobs:
17   format:
18     runs-on: ubuntu-latest
19     steps:
20       - uses: actions/checkout@v4
21       - uses: psf/black@stable
22   lint:
23     name: Lint with ruff
24     runs-on: ubuntu-latest
25     steps:
26       - uses: actions/checkout@v4
27
28       - uses: actions/setup-python@v5
29         with:
30           python-version: "3.11"
31       - name: Install ruff
32         run: |
33           pip install ruff
34       - name: Lint with ruff
35         run: |
36           # stop the build if there are Python syntax errors or undefined names
37           ruff check .
```

Runner: (required)

Which operating system
should this action be executed
on? (e.g. ubuntu, windows,
macos, ...)

Standard GitHub-hosted runners

Virtual Machine	Processor (CPU)	Memory (RAM)	Storage (SSD)	Architecture	Workflow label
Linux	4	16 GB	14 GB	x64	ubuntu-latest , ubuntu-24.04 , ubuntu-22.04 , ubuntu-20.04
Windows	4	16 GB	14 GB	x64	windows-latest , windows-2025 [Public preview], windows-2022 , windows-2019
Linux [Public preview]	4	16 GB	14 GB	arm64	ubuntu-24.04-arm , ubuntu-22.04-arm
macOS	4	14 GB	14 GB	Intel	macos-13
macOS	3 (M1)	7 GB	14 GB	arm64	macos-latest , macos-14 , macos-15 [Public preview]

[about GitHub hosted runners](#)

Workflow in Detail

```
python_package > .github > workflows > cicd.yml
1  # This workflow will install Python dependencies, run tests and lint with a single version of Python
2  # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-with-the-python-action
3
4  name: Python application
5
6  > on: ...
12
13 > permissions: ...
15
16 > jobs:
17   format:
18     runs-on: ubuntu-latest
19     steps:
20       - uses: actions/checkout@v4
21       - uses: psf/black@stable
22
23   lint:
24     name: Lint with ruff
25     runs-on: ubuntu-latest
26     steps:
27       - uses: actions/checkout@v4
28
29       - uses: actions/setup-python@v5
30         with:
31           python-version: "3.11"
32       - name: Install ruff
33         run: |
34             pip install ruff
35       - name: Lint with ruff
36         run: |
37             # stop the build if there are Python syntax errors or undefined names
38             ruff check .
```

Steps: (required)

What does the action need to do? Steps are executed one-by-one. If a step fails, subsequent steps will not be executed.

Run overview

The screenshot shows the GitHub Actions Run overview for the repository 'RasmussenLab / python_package'. The left sidebar contains actions like 'All workflows' (selected), 'Python application' (disabled), and management options like 'Caches', 'Attestations', 'Runners', 'Usage metrics', and 'Performance metrics'. The main area displays '49 workflow runs' for the 'Python application' workflow. One run, 'Python application #41: Scheduled', is highlighted with a green rounded rectangle. The table lists three runs, each with a green checkmark icon, the workflow name, the run ID, the branch ('main'), the time it was run ('2 months ago', '3 months ago', '3 months ago'), and the duration ('28s', '28s', '24s'). A search bar at the top right allows filtering by workflow runs.

Workflow	Run ID	Branch	Event	Status	Duration
Python application	Python application #41: Scheduled	main	2 months ago	28s	...
Python application	Python application #40: Scheduled	main	3 months ago	28s	...
Python application	Python application #39: Scheduled	main	3 months ago	24s	...

GitHub Actions view

← Python application

Python application #41

Summary

Jobs

- ✓ format
- ✓ Lint with ruff
- ✓ Test (3.11)
- ✓ Test (3.12)
- ✓ Build source distribution
- ✗ Publish package

Run details

Usage

Workflow file

Triggered via schedule 2 months ago

Status: Success

Total duration: 28s

Artifacts: 1

cicd.yml

on: schedule

Matrix: Test

✓ 2 jobs completed

Show all jobs

✓ Publish package 0s

✓ format 6s

✓ Lint with ruff 7s

✓ Build source distribution 7s

```
graph LR; A["Matrix: Test  
✓ 2 jobs completed  
Show all jobs"] --> B["Publish package 0s"]; B --> C["format 6s"]; B --> D["Lint with ruff 7s"]; B --> E["Build source distribution 7s"];
```

Format Action in detail

format
succeeded on Jan 1 in 6s

Search logs ⚙️

Step	Description	Duration
> ✓ Set up job		1s
> ✓ Run actions/checkout@v4		1s
> ✓ Run psf/black@stable	Part we defined!	4s
> ✓ Post Run actions/checkout@v4		0s
> ✓ Complete job		0s

Workflow in Detail

```
python_package > .github > workflows > cicd.yml
1  # This workflow will install Python dependencies, run tests and lint with a single version of Python
2  # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-python
3
4  name: Python application
5
6  > on: ...
12
13 > permissions: ...
15
16 > jobs:
17   format:
18     runs-on: ubuntu-latest
19     steps:
20       - uses: actions/checkout@v4
21       - uses: psf/black@stable
22   lint:
23     name: Lint with ruff
24     runs-on: ubuntu-latest
25     steps:
26       - uses: actions/checkout@v4
27
28       - uses: actions/setup-python@v5
29         with:
30           python-version: "3.11"
31         name: Install ruff
32         run:
33           pip install ruff
34         name: Lint with ruff
35         run:
36           # stop the build if there are Python syntax errors or undefined names
37             ruff check .
```

Job name

Program overview

1. Introduction to GitHub Actions
2. First example
3. Yaml format (optional)
4. Workflow file in detail
5. PIMMS repo: add first example (if time allows)

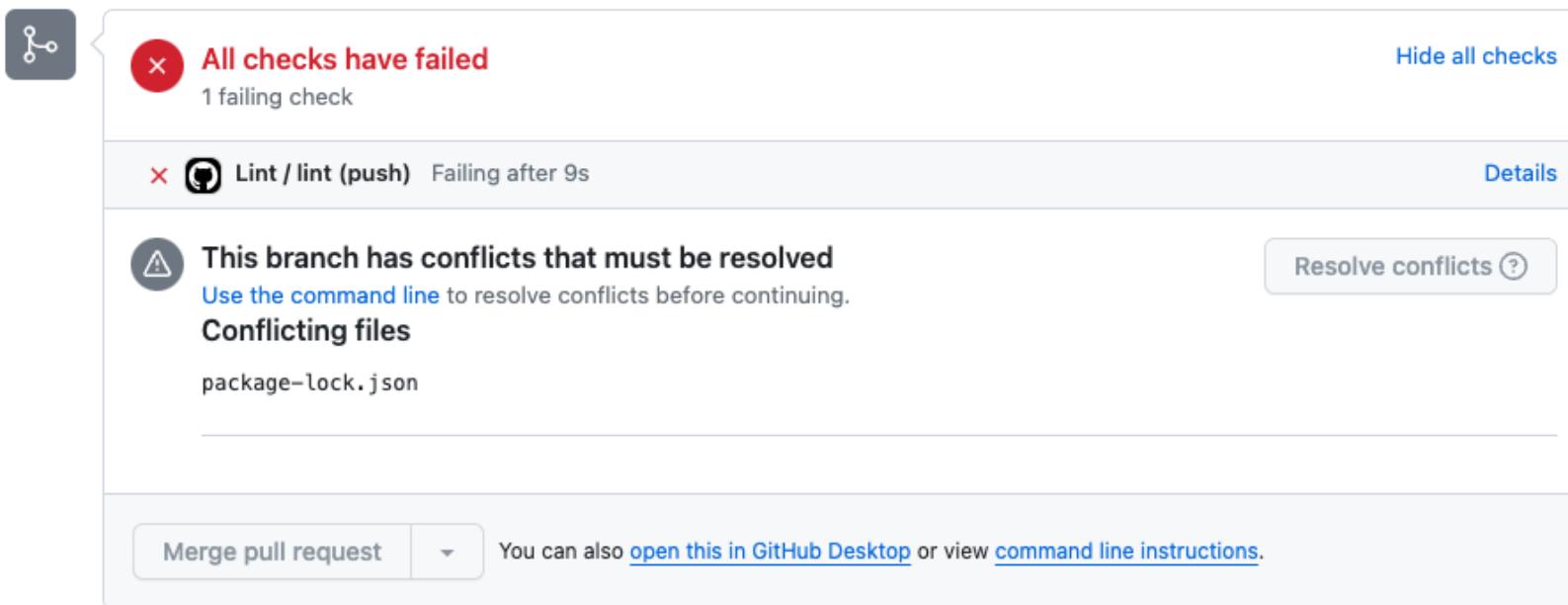
- RasmussenLab/pimms/blob/main/.github/workflows/ci.yaml
 - ▶ Run integration test
 - ▶ Test pip package works for core models in PIMMS
 - ▶ Can publish package to PyPI
- No code formatting checks or linter checks

Workflow triggers

- **Specific event that initiates a workflow**
- Can be categorised in different types
 - ▶ **Event-based:** Triggers workflows based on GitHub events
 - e.g. push, pull_request, issue, ...
 - ▶ **Time-based:** Triggers workflows at a specific moment
 - e.g. each day / week / month / ...
 - ▶ **Manual:** Triggers workflows after you click a button
 - ▶ ...
- Can be combined!

Event-based triggers

- push, pull_request, issue, ...
 - ▶ Workflows are automatically executed after push to repository, creation of new pull request, creation of new issue, ...
 - ▶ Typically used to perform unit tests or code linting
 - ▶ Pull requests **can be blocked** if not all checks pass

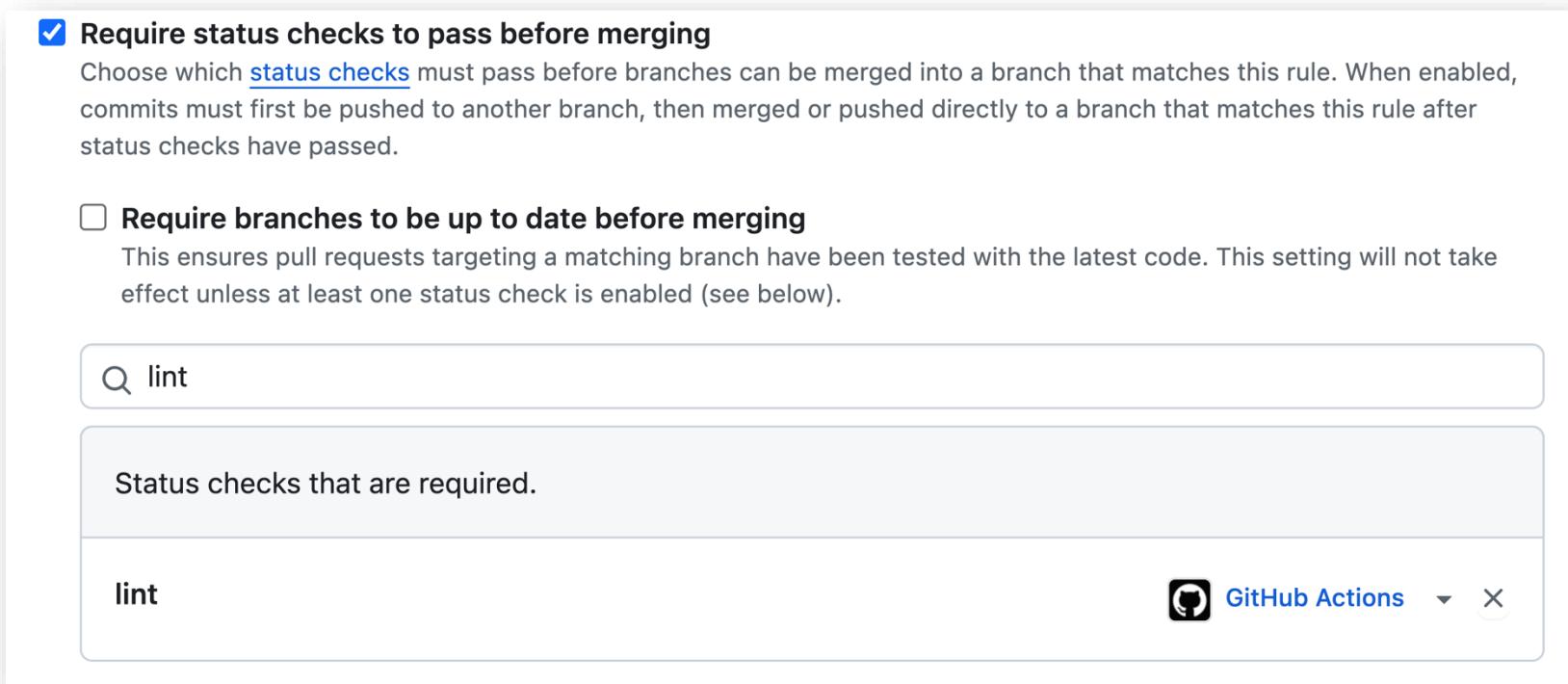


Block merging of pull requests

- Actions can be used to block pull requests if some checks fail
 - ▶ e.g. Failing unit tests, linting, automated code quality assessments
- Action is marked as “**failed**” when one of the steps return non-zero exit code
- Merging can be blocked if one or more actions failed

Block merging of pull requests

- Go to “Settings” > “Branches” > “Add protection rule”
 - ▶ Check “Require status checks to pass before merging”
 - ▶ Select one or more actions that need to successfully pass



Time-based triggers

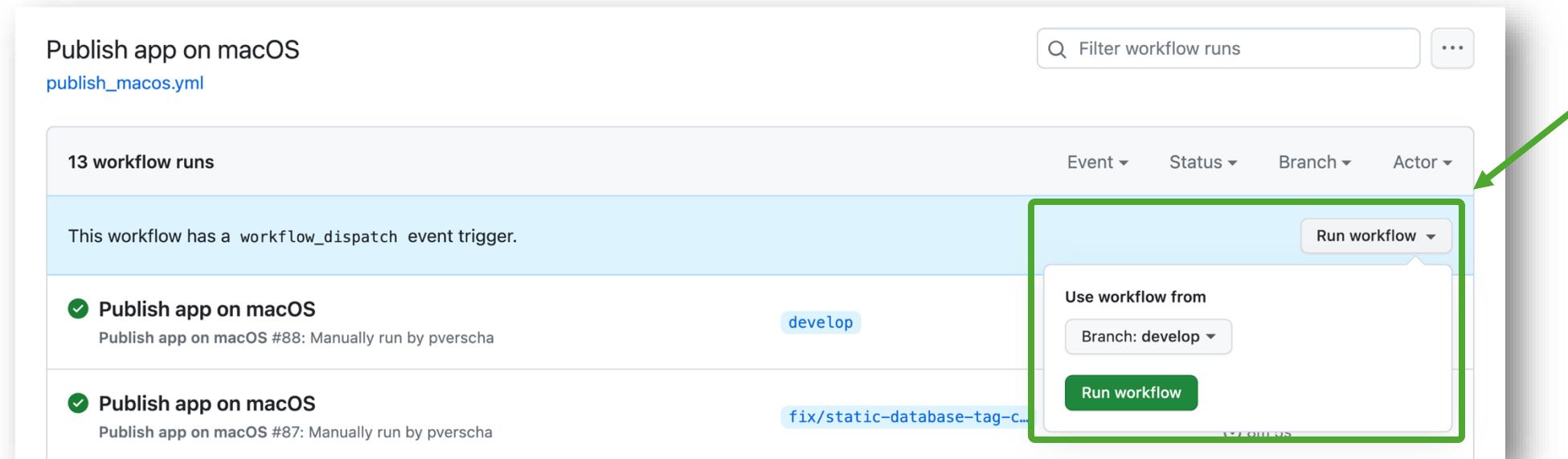
- Automatically triggers the workflow at a specific point in time
- Requires the `schedule` keyword and a cron-compatible expression

```
● ● ●  
name: Monthly status report  
  
on:  
  schedule:  
    # * is a special character in YAML so you have to quote this string  
    - cron: '0 0 1 * *'
```

Tip: use <https://crontab.guru/> to easily construct cron-compatible expressions

Manual triggers

- Add a button to GitHub's interface
- Run the action for a specific branch of the repository
- Simply add `workflow_dispatch` as extra trigger to the workflow
- E.g. manually test code from a specific branch



Manual triggers

- Can even be configured to ask user for input

```
name: Deploy Specific Version

on:
  workflow_dispatch:
    inputs:
      version:
        description: 'Version tag to deploy'
        required: true

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repository
        uses: actions/checkout@v2
        with:
          ref: ${{ github.event.inputs.version }}

      - name: Deploy
        run: |
          echo "Deploying version ${{ github.event.inputs.version }}"
          # Add deployment scripts or commands here
```

Premade actions can be reused

- Similar actions typically require the same initial steps
- Big actions can be split up into smaller actions
 - ▶ Identical steps can be reused by multiple actions
- GitHub Marketplace provides extensive library of reusable actions
 - ▶ <https://github.com/marketplace?type=actions>

Revisiting our code linting example

- Instead of manually installing Python, we can look for premade actions that already do this
- Employ ‘uses’ key when defining step with reusable action

```
● ● ●  
- name: Install Python 3.8  
  run: |  
    sudo apt update  
    sudo apt install python3-pip
```



```
● ● ●  
- name: Install Python  
  uses: actions/setup-python@v2
```

Secrets

- API keys or other “secrets” cannot be added to a repository
 - ▶ Effectively exposes the secret to everyone with access to the repository
- Some actions require knowledge about these keys to function
- GitHub allows you to securely store secrets and reuse these in workflows

1) Go to repository settings

2) Click on “Actions” under “Secrets and variables”

The screenshot shows the GitHub repository settings page for a repository named "Actions secrets and variables". The left sidebar contains a navigation menu with sections like General, Access, Collaborators, Moderation options, Code and automation, Security, and Integrations. Under the "Actions" section in the "Code and automation" category, there is a sub-section titled "Secrets and variables" which is currently selected, indicated by a green border and a green arrow pointing to it from the second step instruction. This section includes links for Actions, Codespaces, and Dependabot.

Actions secrets and variables

Secrets and variables allow you to manage reusable configuration data. Secrets are **encrypted** and are used for sensitive data. [Learn more about encrypted secrets](#). Variables are shown as plain text and are used for **non-sensitive** data. [Learn more about variables](#).

Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.

Environment secrets

This repository has no environment secrets.

[Manage environment secrets](#)

Repository secrets

This repository has no secrets.

[New repository secret](#)

3) Create a new “repository secret”

Secrets

- Can be referenced in workflows using `\${{ secrets.MY_SECRET }}`

```
name: My Workflow

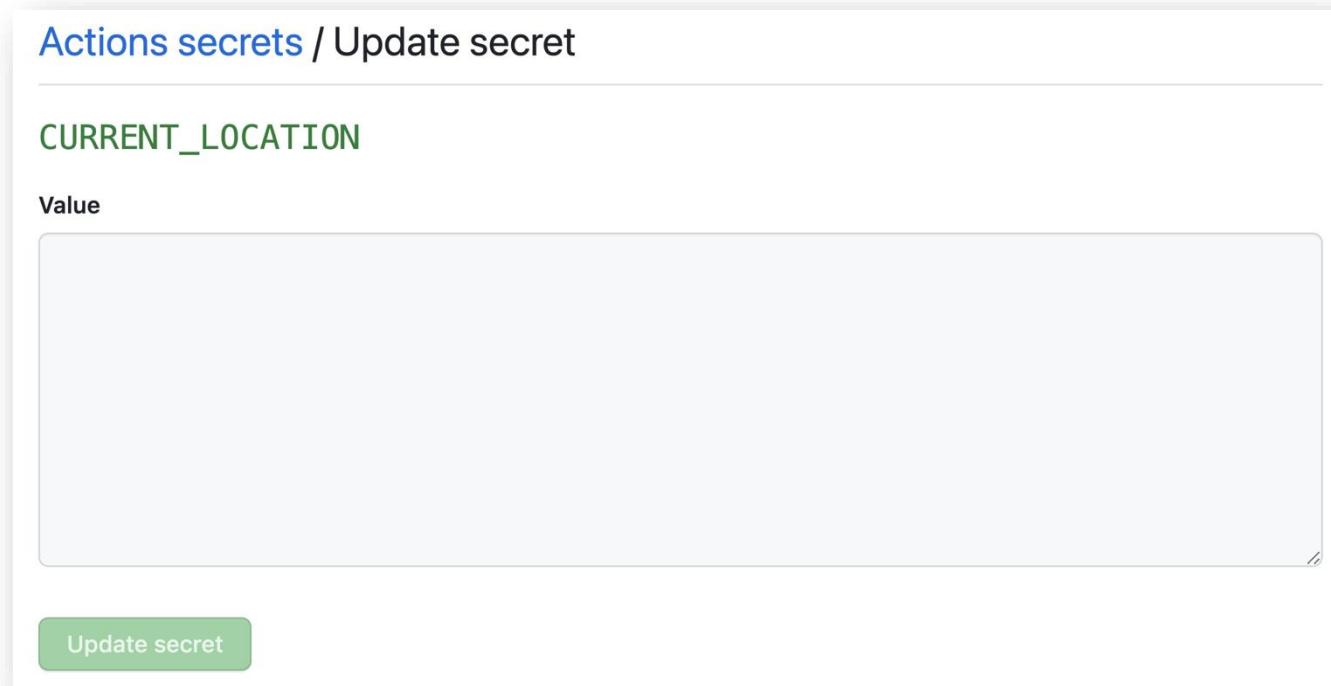
on:
  push:
    branches:
      - main

jobs:
  my-job:
    runs-on: ubuntu-latest
    env:
      MY_ENV_VAR: ${{ secrets.MY_SECRET }}
    steps:
      - name: My Step
        run: echo $MY_ENV_VAR
```



Secrets

- Can only be read by workflows in your repository
 - ▶ Is not even visible to you
- Value can be updated, but previous values are invisible!



Environment variables

- Small pieces of data that define how a program or script behaves
- Are typically used for paths or settings related to the application
 - ▶ E.g. where are configuration files stored?
 - ▶ E.g. which URL should be used to request specific pieces of information from?



```
name: Example Workflow

on: [push]

jobs:
  run-script:
    runs-on: ubuntu-latest

  env:
    CUSTOM_VAR: 'Hello World'

    # Setting an environment variable

steps:
  - name: Checkout Repository
    uses: actions/checkout@v2

  - name: Run Script
    run: |
      echo "The value of CUSTOM_VAR is $CUSTOM_VAR"
```

Continuous Integration (CI)

- The practice of frequently integrating code changes into a shared repository. Ideally, developers will integrate their changes daily, if not multiple times a day.
- Each integration is automatically tested to detect errors as quickly as possible
- GitHub Actions are an ideal candidate for this!

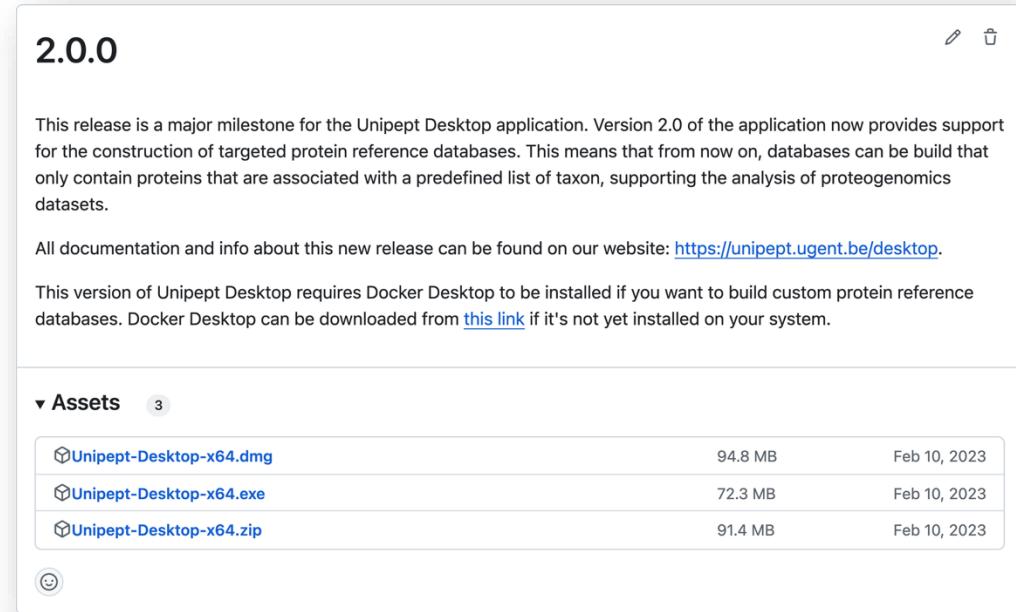
Continuous Development (CD)

- The practice of continuously developing, testing, integrating, and deploying software
- Release early, release often
 - ▶ New features are released as soon as they are ready
 - ▶ User feedback can be gathered soon
 - ▶ Adjustments can be applied early in the development process

Other world examples (1)

Automatically build new version of Unipept Desktop application

- A different installer is required for each operating system
- Accompanying GitHub release needs to be created with all installers
 - ▶ <https://github.com/unipept/unipept-desktop/tree/develop/.github/workflows>



Other world examples (2)

Reconstruct new taxon database monthly

- New NCBI taxa are added every month
- Our application requires all metadata for these taxa
- A GitHub action automatically parses the new taxa and generates a database file
 - ▶ https://github.com/unipept/unipept-database/blob/master/.github/workflows/static_database.yml

Optimizing workflow performance

- Complicated GitHub actions can take a long time to execute
- Here are some tips and tricks to speed up workflow performance
 - ▶ Less GitHub credits wasted
 - ▶ Less time waiting for checks to pass
 - ▶ More convenient in some situations

Cache installed dependencies

- Installing dependencies takes a long time
- Reusing dependencies skips a big step in the workflow process
- Use GitHub Cache Action
 - ▶ <https://github.com/actions/cache>
 - ▶ Checks if a cached environment is available
 - ▶ If present, check if cache is still valid and reuses previously installed deps
 - ▶ If not, installs dependencies and creates new cache entry

```
steps:  
- uses: actions/checkout@v4  
- uses: actions/setup-python@v5  
  with:  
    python-version: '3.9'  
    cache: 'pip' # caching pip dependencies  
- run: pip install -r requirements.txt
```

Test actions locally with `act`

- Updating and testing actions:
 - ▶ Make local change, commit change, wait and check for action status
 - ▶ Can take a long time
 - ▶ Tracking down persistent bugs can be hard
 - ▶ Tedious process
- Running GitHub actions locally is possible with `act`
 - ▶ Creates a new Docker container reflecting the specified workflow runner
 - ▶ <https://github.com/nektos/act>
 - ▶ Has some limitations, but can be very handy