

Versioning and Stable Releases

Dushyant Mehra

Outline

- Why stable releases?
- Creating tags in Github
- Creating a stable release from a Tag
- Creating a Uploading a Package using PyPI and Twine
- Versioning
- Stable DOIs for publications using Zenodo

Why stable release?

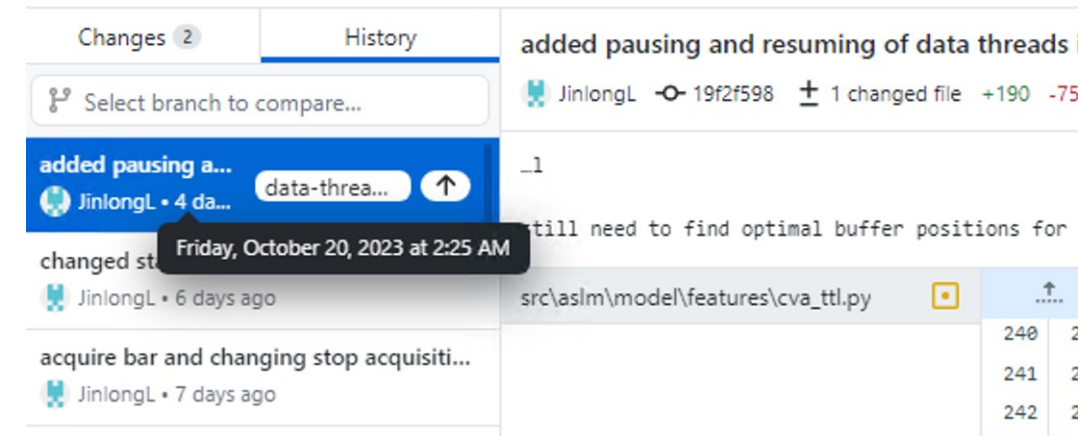
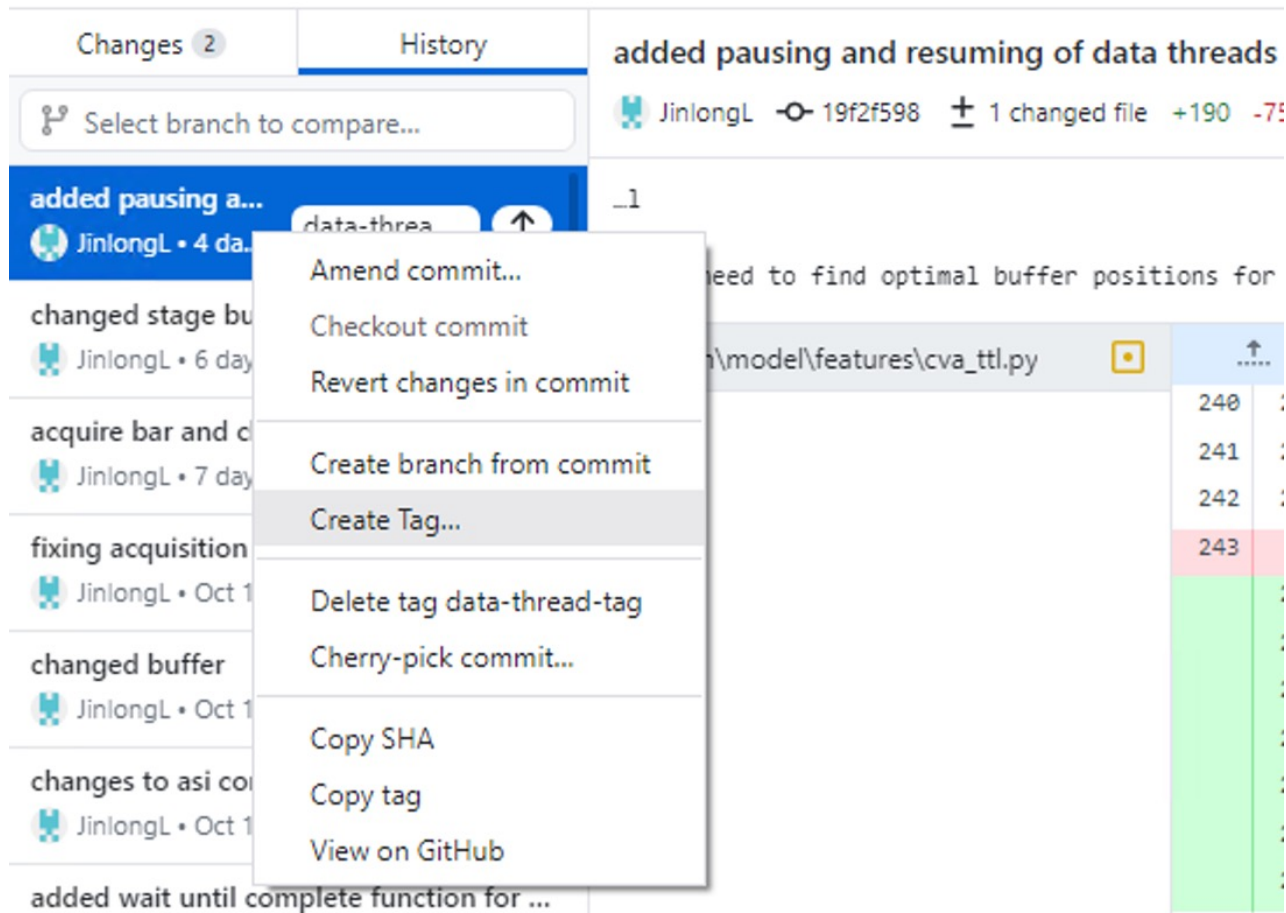
- Collaborative codebases are constantly updated and changed.
 - Having stable versions helps with managing releases
- Important for reproducibility of research for publications.
- It is important to have a specific release that ensures code is still functional
 - Ensure packages that are installed to make the

Implementing stable release

- Github Tag
- Github Stable Release
- Packaging Projects using PyPI and twine
- Github Release and stable DOI via Zenodo
- If we have time: Conda Constructor

Creating a Github Tag to a commit

- In Github it is possible tag a to a commit



Creating a stable release in Github

- Can create a stable release from a tag on recent commit using Github

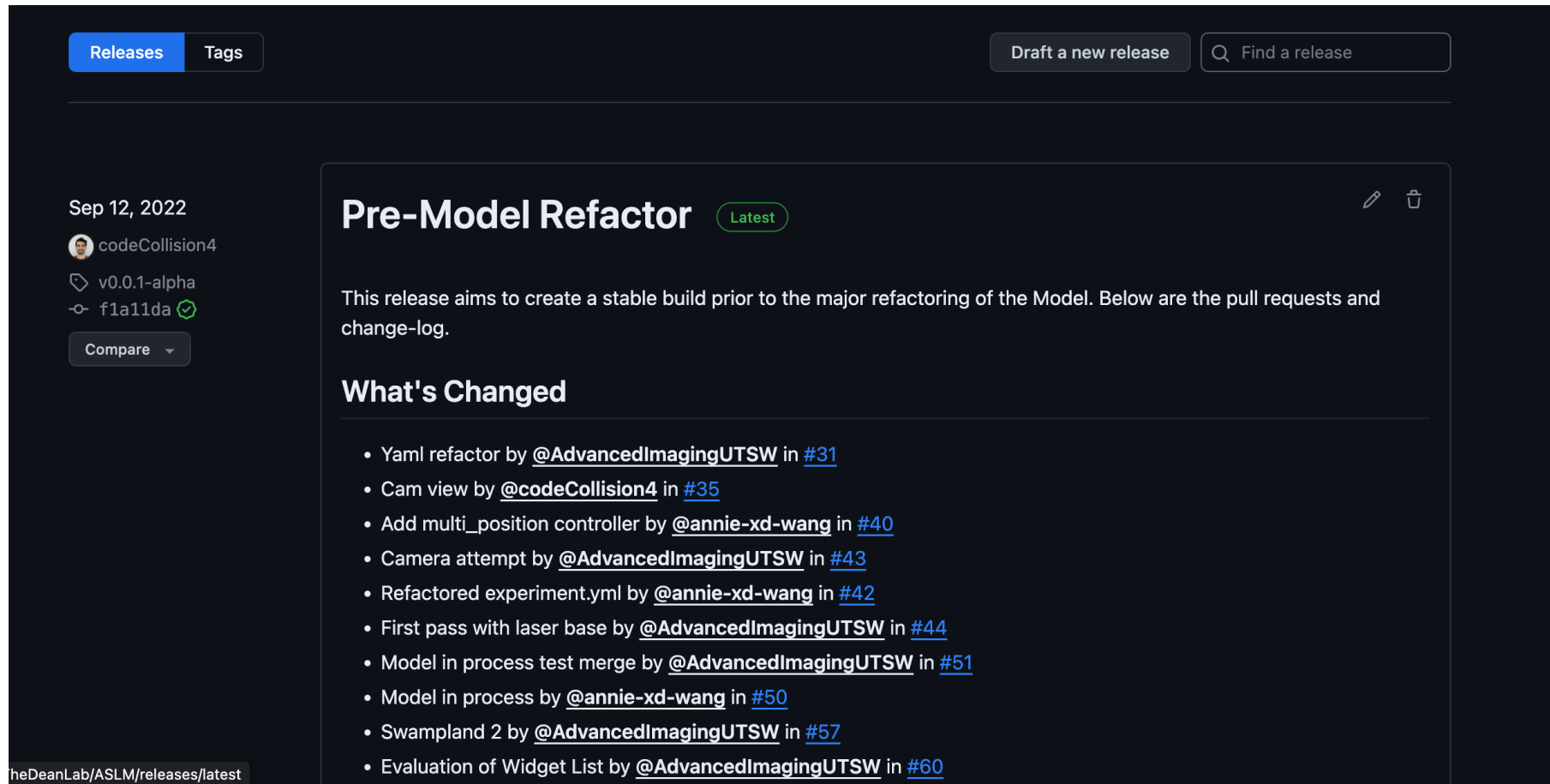
The screenshot shows the GitHub repository page for 'AdvancedImagingUTSW'. The repository is in the 'develop' branch, has 219 branches, and 1 tag. A merge pull request #644 from 'zacsimile/fix641' is visible, with a green checkmark indicating it is ready to merge. The repository has 2,465 commits and was last updated 15 hours ago.

The repository contains the following files and folders:

File/Folder	Description	Last Updated
.github/workflows	Attempt to fix GitHub issue	last month
design	update camera setting controller	last year
docs	Sphinx documentation for the feature container.	3 weeks ago
src	Force ASI axes to uppercase	18 hours ago
test	Fix tests	17 hours ago
.gitignore	Delete test.xml	last month
.pre-commit-config.yaml	Fix pre-commit	9 months ago
LICENSE	Update LICENSE	6 months ago
MANIFEST.in	Update and rename src/MANIFEST.in to MANIFEST.in	6 months ago
README.md	Update README.md	6 months ago
aslm_architecture.jpg	Create aslm_architecture.jpg	2 years ago
codecov.yml	Update codecov.yml	4 months ago
pytest.ini	Add flag for physical hardware tests on instrument machines	last year
requirements.txt	Requirements update, test TODO	6 months ago
setup.py	a start...	9 months ago

The right sidebar shows the repository's 'About' section, which describes it as 'Autonomous Software for Light Microscopy - Open source light-sheet microscope controls.' It also lists the repository's statistics: 3 stars, 6 watching, and 1 fork. The 'Releases' section shows a single release, 'Pre-Model Refactor', which is the latest release, dated Sep 12, 2022. The 'Packages' section indicates that no packages have been published. The 'Contributors' section shows 15 contributors.

Creating a stable release using Github



The screenshot shows the GitHub interface for a repository. At the top, there are tabs for 'Releases' and 'Tags', with 'Releases' selected. To the right, there are buttons for 'Draft a new release' and a search bar labeled 'Find a release'. On the left sidebar, the date 'Sep 12, 2022' is shown, followed by the user 'codeCollision4', the version 'v0.0.1-alpha', and the commit 'f1a11da' with a green checkmark. Below this is a 'Compare' button. The main content area displays the release title 'Pre-Model Refactor' with a 'Latest' badge. Below the title, a paragraph states: 'This release aims to create a stable build prior to the major refactoring of the Model. Below are the pull requests and change-log.' Underneath this is a section titled 'What's Changed' which contains a bulleted list of pull requests. At the bottom left, the repository path 'heDeanLab/ASLM/releases/latest' is visible.

Releases Tags

Draft a new release Find a release

Sep 12, 2022

codeCollision4

v0.0.1-alpha

f1a11da

Compare

Pre-Model Refactor Latest

This release aims to create a stable build prior to the major refactoring of the Model. Below are the pull requests and change-log.

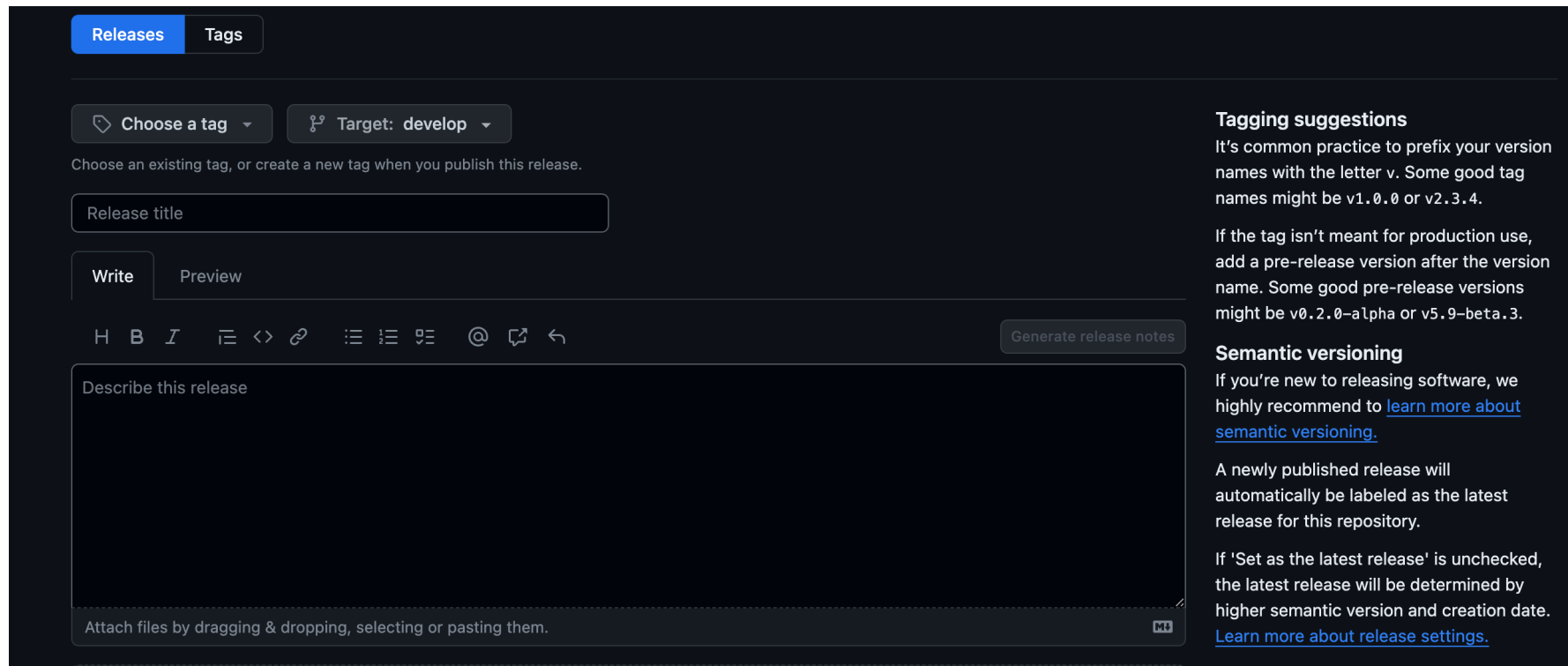
What's Changed

- Yaml refactor by [@AdvancedImagingUTSW](#) in [#31](#)
- Cam view by [@codeCollision4](#) in [#35](#)
- Add multi_position controller by [@annie-xd-wang](#) in [#40](#)
- Camera attempt by [@AdvancedImagingUTSW](#) in [#43](#)
- Refactored experiment.yml by [@annie-xd-wang](#) in [#42](#)
- First pass with laser base by [@AdvancedImagingUTSW](#) in [#44](#)
- Model in process test merge by [@AdvancedImagingUTSW](#) in [#51](#)
- Model in process by [@annie-xd-wang](#) in [#50](#)
- SwampLand 2 by [@AdvancedImagingUTSW](#) in [#57](#)
- Evaluation of Widget List by [@AdvancedImagingUTSW](#) in [#60](#)

heDeanLab/ASLM/releases/latest

Drafting a stable release using Github

- Click draft stable release using Github



The screenshot shows the GitHub interface for drafting a new release. At the top, there are two tabs: 'Releases' (active) and 'Tags'. Below the tabs, there are two dropdown menus: 'Choose a tag' and 'Target: develop'. A note below these says 'Choose an existing tag, or create a new tag when you publish this release.' Below that is a text input field for 'Release title'. There are two buttons: 'Write' (active) and 'Preview'. Below these is a rich text editor with various formatting icons (bold, italic, link, etc.) and a 'Generate release notes' button. The main text area is labeled 'Describe this release'. At the bottom, there is a note about attaching files by dragging and dropping. On the right side, there is a 'Tagging suggestions' section with advice on version naming (e.g., v1.0.0, v2.3.4, v0.2.0-alpha, v5.9-beta.3) and a 'Semantic versioning' section with a link to 'learn more about semantic versioning'.










Releases Tags

Choose a tag Target: develop

Choose an existing tag, or create a new tag when you publish this release.

Release title

Write Preview

H B I         

Generate release notes

Describe this release

Attach files by dragging & dropping, selecting or pasting them.

Tagging suggestions
It's common practice to prefix your version names with the letter v. Some good tag names might be v1.0.0 or v2.3.4.

If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be v0.2.0-alpha or v5.9-beta.3.

Semantic versioning
If you're new to releasing software, we highly recommend to [learn more about semantic versioning](#).

A newly published release will automatically be labeled as the latest release for this repository.

If 'Set as the latest release' is unchecked, the latest release will be determined by higher semantic version and creation date. [Learn more about release settings](#).

Creating a stable release from a tag using Github

ReleasesTags

Choose a tag









Target: develop

Choose a tag

Find or create a new tag

v0.0.1-alpha

publish this release.

H B I       @  

Generate release notes

Describe this release

Attach files by dragging & dropping, selecting or pasting them.

↓ Attach binaries by dropping them here or selecting them.

Tagging suggestions

It's common practice to prefix your version names with the letter v. Some good tag names might be v1.0.0 or v2.3.4.

If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be v0.2.0-alpha or v5.9-beta.3.

Semantic versioning

If you're new to releasing software, we highly recommend to [learn more about semantic versioning](#).

A newly published release will automatically be labeled as the latest release for this repository.

If 'Set as the latest release' is unchecked, the latest release will be determined by higher semantic version and creation date. [Learn more about release settings](#).

Creating a stable release from a branch

The screenshot shows the GitHub 'Releases' page for a repository. The 'Releases' tab is active, and the 'Tags' tab is also visible. A dropdown menu is open for 'Choose a tag', showing a list of branches and recent commits. The 'Target: develop' dropdown is also visible. The 'Release title' field is empty. The 'Write' button is highlighted, and the 'Preview' button is also visible. The 'Describe this release' text area is empty. The 'Attach files by dragging & dropping or selecting them' area is also visible. The 'Generate release notes' button is present. The right sidebar contains 'Tagging suggestions' and 'Semantic versioning' sections.

Releases **Tags**

Choose a tag ▾ Target: develop ▾

Choose an existing tag, or create a new one

Release title

Write Preview

H B I ≡ <>

Describe this release

Attach files by dragging & dropping or selecting them

Generate release notes

Pick a branch or recent commit

Filter branches...

Branches Recent Commits

- 21-volume-search
- 191-sphinx-autodoc-creator
- 199-program-asi-stage
- 203-offset-spinbox-in-remote-focus-settings-doesnt-allow-negative-values
- 223
- 227-automatic-handling-of-synthetic-stage-axes-gui-should-only-display-available-axes
- 240-features-unit-tests
- 252-sub-controller-unit-tests
- 262-stage-unit-tests

Tagging suggestions

It's common practice to prefix your version names with the letter v. Some good tag names might be v1.0.0 or v2.3.4.

If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be v0.2.0-alpha or v5.9-beta.3.

Semantic versioning

If you're new to releasing software, we highly recommend to [learn more about semantic versioning](#).

A newly published release will automatically be labeled as the latest release for this repository.

If 'Set as the latest release' is unchecked, the latest release will be determined by higher semantic version and creation date. [Learn more about release settings](#).

Creating a stable release from a commit

The screenshot shows the GitHub Releases interface. At the top, there are tabs for 'Releases' and 'Tags'. Below these, there's a 'Choose a tag' dropdown and a 'Target: develop' dropdown. A modal window titled 'Pick a branch or recent commit' is open, showing a list of recent commits. The modal has a search bar 'Filter recent commits...' and two tabs: 'Branches' and 'Recent Commits'. The 'Recent Commits' tab is active, showing a list of commits with their hashes and descriptions. The main form on the left has a 'Release title' field, a 'Write' button, and a 'Describe this release' text area. On the right, there's a 'Generate release notes' button and a section titled 'Tagging suggestions' with text about version naming. Below that is a 'Semantic versioning' section with text about release labeling and a link to 'Learn more about release settings'.

Releases **Tags**

Choose a tag Target: develop

Choose an existing tag, or create a new one

Release title

Write Preview

H B I ≡ <>

Describe this release

Attach files by dragging & dropping them here or selecting them.

Pick a branch or recent commit

Filter recent commits...

Branches Recent Commits

- 9c7d1d8
Merge pull request #644 from zacsimile/fix641 @ 15 hours ago
- 2d615be
Fix tests @ 17 hours ago
- cb4f696
Force ASI axes to uppercase @ 18 hours ago
- 1185adb
Merge pull request #643 from zacsimile/xml-advances @ 2 days ago
- d2d041e
Remove extraneous common_features adjustment @ 2 days ago
- 30d9d7c
Merge remote tracking branch

Generate release notes

Tagging suggestions

It's common practice to prefix your version names with the letter v. Some good tag names might be v1.0.0 or v2.3.4.

If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be v0.2.0-alpha or v5.9-beta.3.

Semantic versioning

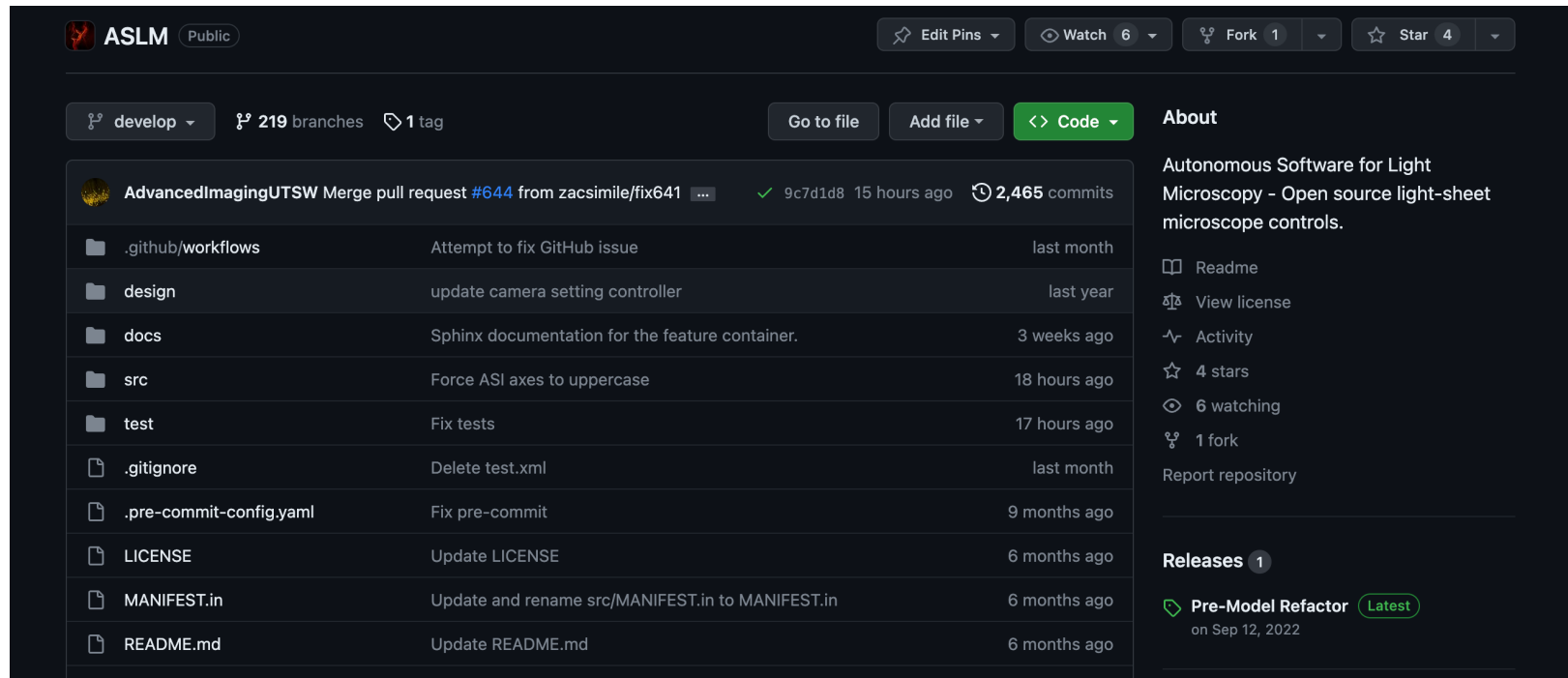
If you're new to releasing software, we highly recommend to [learn more about semantic versioning](#).

A newly published release will automatically be labeled as the latest release for this repository.

If 'Set as the latest release' is unchecked, the latest release will be determined by higher semantic version and creation date. [Learn more about release settings](#).

Stable Release

- Stable releases are shown on the main github page and will link to that branch
- Make sure versions of dependences are also included in the pyproject.toml file and operating systems are defined.
- Release will come with a link that can be used for citation



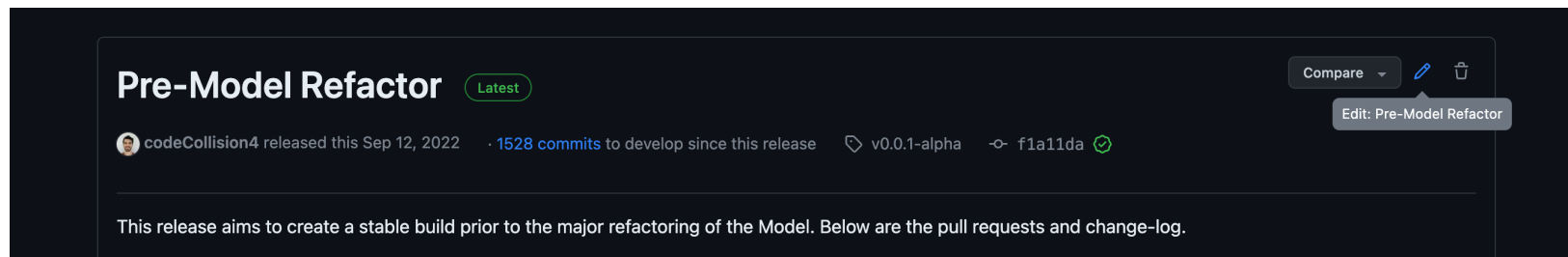
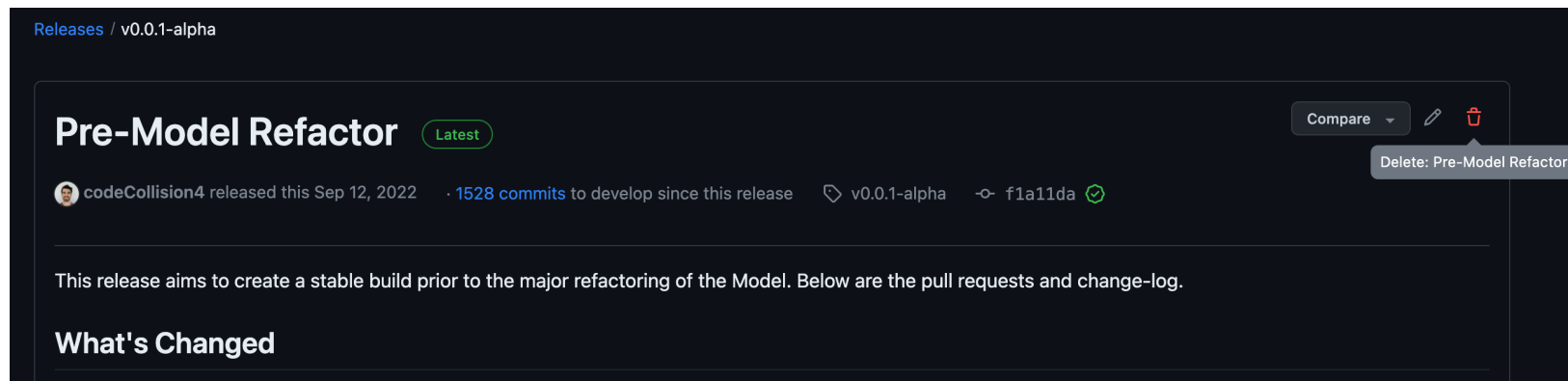
The screenshot displays the GitHub interface for the ASLM repository. At the top, the repository name 'ASLM' is shown with a 'Public' badge. Interaction buttons include 'Edit Pins', 'Watch' (6), 'Fork' (1), and 'Star' (4). Below this, a navigation bar shows the current branch 'develop', '219 branches', and '1 tag'. Action buttons for 'Go to file', 'Add file', and 'Code' are present. The main content area is divided into two columns. The left column lists repository files and their commit history:

File	Commit Message	Time
.github/workflows	Attempt to fix GitHub issue	last month
design	update camera setting controller	last year
docs	Sphinx documentation for the feature container.	3 weeks ago
src	Force ASI axes to uppercase	18 hours ago
test	Fix tests	17 hours ago
.gitignore	Delete test.xml	last month
.pre-commit-config.yaml	Fix pre-commit	9 months ago
LICENSE	Update LICENSE	6 months ago
MANIFEST.in	Update and rename src/MANIFEST.in to MANIFEST.in	6 months ago
README.md	Update README.md	6 months ago

The right column contains the 'About' section, describing the project as 'Autonomous Software for Light Microscopy - Open source light-sheet microscope controls.' It includes links for 'Readme', 'View license', and 'Activity', along with statistics: '4 stars', '6 watching', and '1 fork'. Below this is the 'Releases' section, showing one release: 'Pre-Model Refactor' (Latest) from Sep 12, 2022.

Editing or Deleting Stable Release

- It is possible to edit or delete a stable release by editing the tag or changing the fix.

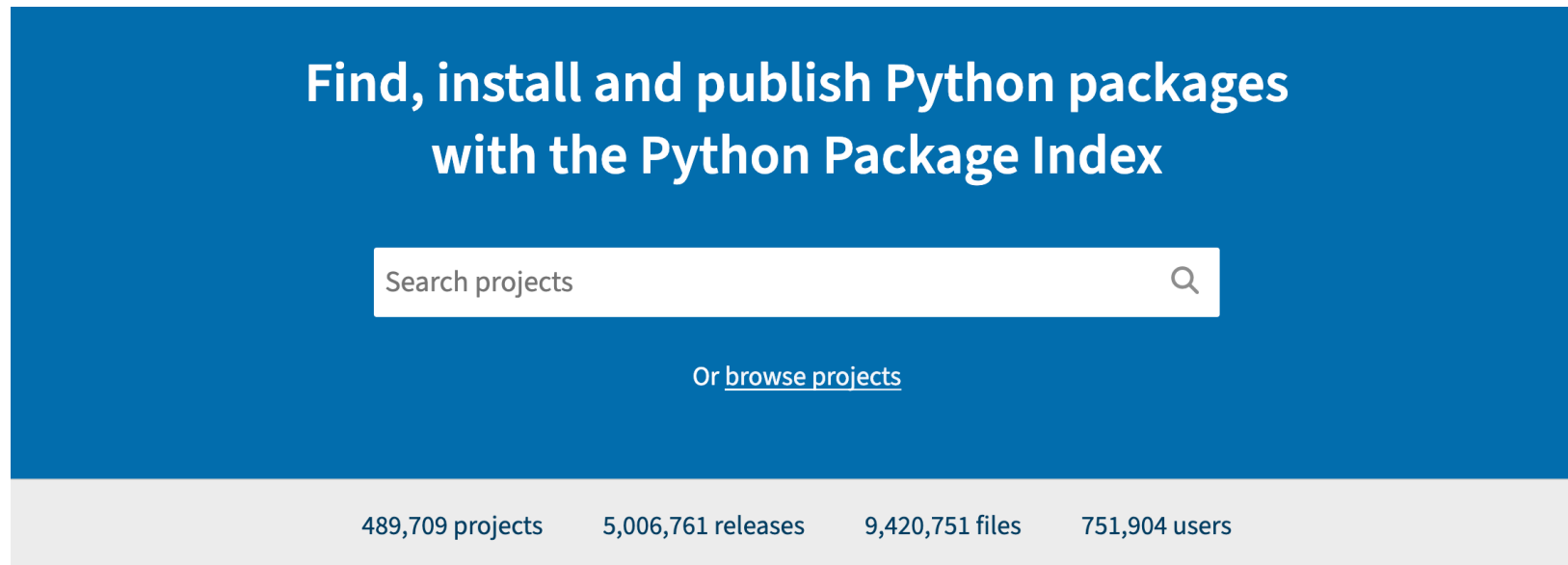


Downloading Release from Github

- To download latest version of software package, we can use
 - `pip install "git+https://github.com/user/project"`
- To download a stable release:
 - `Pip install "git+https://github.com/user/project1.2.3"`
- Include stable release as dependency in `pyproject.toml`:
 - `Dev = ["project @ git+https://github.com/user/project1.2.3"]`
- Zip files of releases can be created and can be used for faster install. For stable release:
 - `Pip install "https://github.com/user/project1.2.3.zip"`
- For dependency in `pyproject.toml`:
 - `Dev = ["project @ https://github.com/user/project1.2.3.zip"]`

Python Packaging Index (PyPI)

- We have used PyPI to install python packages before
- PyPI is a package management server that can be used
- PyPI is a package management server to find install and publish packages



Publishing Packages using PyPI and Twine

- Step 1: Create a Readme.md file
- Step 2: Create a license file
- Step 3: Organize files in src and tests folder
- Step 4: buiding and versioning in pyproject.toml file
- Step 5: building wheel
- Step 6: pushing to PyPI
- Step 7: automatically publish releases using Github Actions

Step 1: Create a README.md file

- Create a README.md file that includes
 - Description of Package
 - How to install package
 - How to use package
 - Authors
- Include README.md in pyproject.toml
- Include citation in README.md for academic research

```
# pyqt5-calc

## Simple calculator based on MVC pattern with Python and PyQt5 library.

Calculator based on tutorial from [Real Python](https://realpython.com/): [Python and PyQt: Building a GUI Desktop Calculator](https://realpython.com/python-pyqt-gui-calculator/). I updated the implementation to split the application into separate [modules](https://docs.python.org/3/tutorial/modules.html) to follow the [MVC (model-view-controller)](https://realpython.com/the-model-view-controller-mvc-paradigm-summarized-with-legos) pattern used in the tutorial, with those modules further placed into a package.

## Installation

For a development install, enter `pip install -e .[dev]`.
```

Step 2: Create a license file

- Create a license file that includes rights for usage or redistribution
- Important for code distribution
- Use UTSW license file.
- Include license in `pyproject.toml`

MIT License

Copyright (c) 2021 TimothyDJones

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

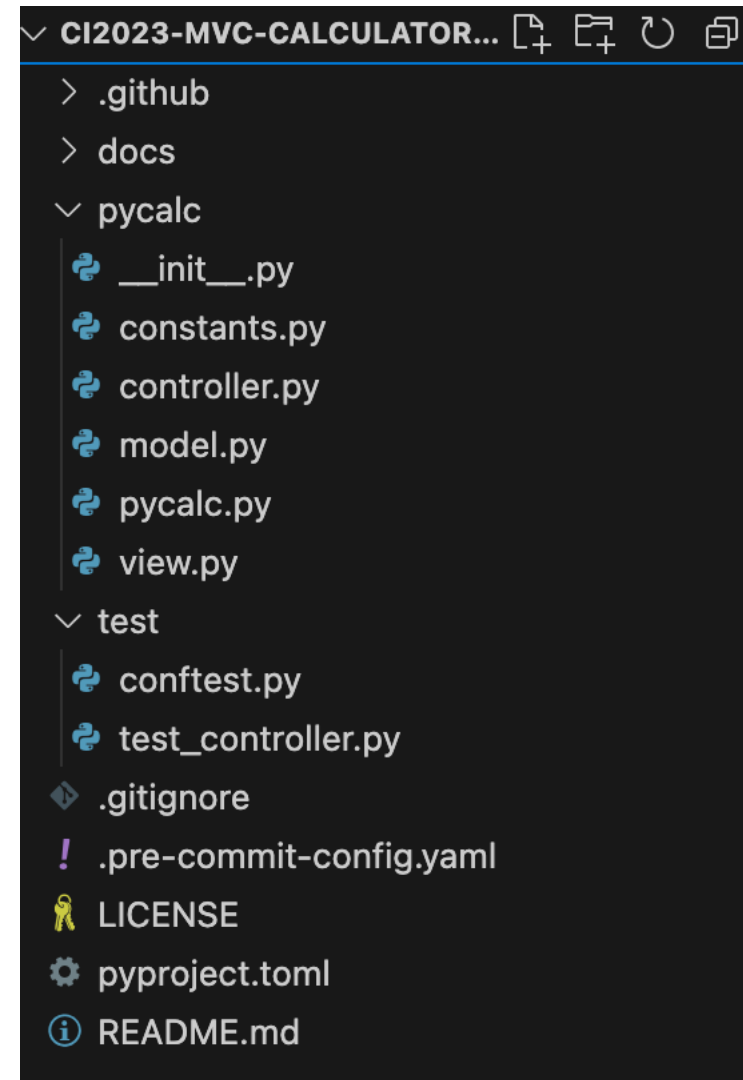
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

```
[project]
name = "pycalc"
description = "An MVC calculator made in Qt."
authors = [{name = "Tim Jones"}]
readme = "README.md"
license = {file = "LICENSE.md"}
dynamic = ["version"]
```

Step 3: Organize files in src and tests folder

- Organize files into test and src folders similar to first calculator exercise
- Make sure that README.md, pyproject.toml, and license file are at the top level



Step 4: building and versioning in pyproject.toml file

- Versions of code are defined in 3 locations
 - Git Tag
 - Version field in pyproject.toml
 - `__version__` in `__init__.py`
- By using `setuptools.scm` it will be possible to sync all three
- Add `setuptools` to build section of `pyproject.toml`
- Add `set` dynamic to version in project section of `pyproject.toml`

```
[build-system]
requires = ["setuptools"]
build-backend = "setuptools.build_meta"
```

```
[project]
name = "pycalc"
description = "An MVC calculator made in Qt."
authors = [{name = "Tim Jones"}]
readme = "README.md"
license = {file = "LICENSE.md"}
dynamic = ["version"]
```

Step 5: Building wheel

- With using pyproject.toml, build is included so we can use the function to build a wheel
 - Python -m build
- Without using pyproject.toml, we would need to install build using
 - Pip install build
 - Python -m build
- Wheel file will be created in dist folder within repo
- Check wheel file by going to directory and unzipping file
 - Cd dist
 - Unzip project.whl

```
dist
|
|___ epi_models-0.1.0-py3-none-any.whl
|___ epi_models-0.1.0.tar.gz
```

Step 6: publishing wheel to PyPI using twine

- Install twine to convert packages to wheels to push to PyPI
 - Pip install twine
- Check package with twine
 - `twine check dist/*`
- Register for PyPI and possibly TestPyPI
 - <https://pypi.org/>
 - <https://test.pypi.org/> (check if package works)
- Install into testPyPI
 - `twine upload --repository testpypi dist/*`
 - `pip install -i https://test.pypi.org/pypi/`
 - Wheel file will be created in dist folder within repo
- Install into PyPI
 - `twine upload dist/*`
 - Check if package works by using
 - `pip install project`

Step 7: automatically publish releases using Github Actions

- Use Github actions to automatically publish a package when a new release is setup in github
- Specify Yaml file to build package into wheel file
- Use github actions to publish package of a specific release instead of using twine.

```
# Provide a name for the workflow
name: Upload Python Package

# Tell GitHub when to run the action
# This will run every time a new release is published
on:
  release:
    types: [published]


jobs:
  deploy:
    # Run the workflow on the latest Ubuntu version
    runs-on: ubuntu-latest
    steps:
      # This will checkout our GitHub project and enter
      # the directory
      - uses: actions/checkout@v3
      # This will set up a Python environment
      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.x'
      # Here we update pip to the latest version and
      # install 'build'. We won't need 'twine' here.
      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install build
      # Here we run build to create a wheel and a
      # .tar.gz source distribution.
      - name: Build package
        run: python -m build --sdist --wheel
      # Finally, we use a pre-defined action to publish
      # our package in place of twine.
      - name: Publish package
        uses: pypa/gh-action-pypi-publish@release/v1
        with:
          user: __token__
          password: ${ secrets.PYPI_API_TOKEN }
```


Notes on Versioning

- Versioning is an important part of stable releases
 - Different stable release versions can alter functionality of code.
- Versions of stable releases need to be defined just like packages when listing dependencies
- Versions below 1 often signify that a repo is going through constant updates between versions and is considered unstable
 - Very little to no notification of changes before release is published
- Versions 1 or above signify a stable release constant updates between versions and is considered unstable
 - Functions which are being phased out in newer versions might be tagged with a depreciation warning so users know that this function will be removed or updated in a new released

Archiving using Github and Zenodo

- For publications, we would need a stable DOI that shows what code was used to facilitate the study
- Often, we adapt a repo to generate specific plots for a paper
- You don't want to place all of the code you used to create the plots into the repo
 - You might also have datasets you want to include that are larger than the limit allowed by github (~50 gb) or not necessary for the repo.
- Go to Zenodo Github page and login with github account
- Authorize zenodo access to github
- Follow Zenodo instructions to create a stable doi for a release in a repository

 GitHub Repositories (updated just now) Sync now

 Get started

1 Flip the switch

Select the repository you want to preserve, and toggle the switch below to turn on automatic preservation of your software.

ON (example)

2 Create a release

Go to GitHub and [create a release](#). Zenodo will automatically download a .zip-ball of each new release and register a DOI.

3 Get the badge

After your first release, a DOI badge that you can include in GitHub README will appear next to your repository below.

DOI [10.5281/zenodo.8475](#) (example)

Conclusions

- Creating stable releases of code is a useful method to share codebases with others
 - Helps with reproducibility
- It is possible to create releases with github tags, releases, and creating packages on PyPI
- Create stable DOIs for publication using zenodo.

Exercise

- Build a wheel and create a stable release of codebase using PyPI and twine

Further Reading

- Creating tags and releases in github: <https://docs.github.com/en/repositories/releasing-projects-on-github/managing-releases-in-a-repository>
- Creating packages with PyPI: https://carpentries-incubator.github.io/python_packaging/instructor/05-publishing.html#versions-and-releases
- Create stable releases with zenodo: <https://docs.github.com/en/repositories/archiving-a-github-repository/referencing-and-citing-content>
- Conda Constructor: <https://anaconda.org/anaconda/constructor>,
<https://github.com/conda/constructor>
- Creating packages from releases with github actions:
<https://docs.github.com/en/packages/managing-github-packages-using-github-actions-workflows/publishing-and-installing-a-package-with-github-actions>
- Versioning packages:
https://setuptools.pypa.io/en/latest/userguide/pyproject_config.html