warn-scraper

unknown

Sep 21, 2023

CONTENTS

Command-line interface for downloading WARN Act notices of qualified plant closings and mass layoffs from state government websites

CHAPTER

ONE

INSTALLATION

Use *pip* or *pipenv* to install the Python library and command-line interface.

```
pipenv install warn-scraper
```

CHAPTER TWO

DOCUMENTATION

2.1 Usage

You can use the warn-scraper command-line tool to scrape available states by supplying one or more two-letter state postal codes. It will write files, by default, to a hidden directory in the user's home directory. On Apple and Linux systems, this will be ~/.warn-scraper.

```
# Scrape a single state
warn-scraper AK

# Scrape multiple states
warn-scraper AK CT
```

To use the warn library in Python, import a state's scraper and run it directly.

```
from warn.scrapers import ak

ak.scrape()
```

2.1.1 Configuration

You can set the WARN_OUTPUT_DIR environment variable to specify a different download location.

Use the --help flag to view additional configuration and usage options:

```
warn-scraper --help

Usage: python -m warn [OPTIONS] [STATES]...

  Command-line interface for downloading WARN Act notices.

  STATES -- a list of one or more state postal codes to scrape. Pass `all` to
  scrape all supported states.

Options:
  --data-dir PATH                  The Path were the results will be saved
  --cache-dir PATH                 The Path where results can be cached
  --delete / --no-delete           Delete generated files from the cache
  -l, --log-level [DEBUG|INFO|WARNING|ERROR|CRITICAL]
```

(continues on next page)

(continued from previous page)

```
                                 Set the logging level
  --help                         Show this message and exit.
```

2.2 Contributing

Our project welcomes new contributors who want to help us fix bugs and improve our scrapers. The current status of our effort is documented in our sources list and issue tracker. We want your help. We need your help. Here's how to get started.

Adding features and fixing bugs is managed using the GitHub's pull request system.

The tutorial that follows assumes you have the Python programming language, the pipenv package manager and the git version control system already installed. If you don't, you'll want to address that first.

The typical workflow

2.2.1 Create a fork

Start by visiting our project's repository at github.com/biglocalnews/warn-scraper and creating a fork. You can learn how from GitHub's documentation.

2.2.2 Clone the fork

Now you need to make a copy of your fork on your computer using GitHub's cloning system. There are several methods to do this, which are coverd in the GitHub documentation.

A typical terminal command will look something like the following, with your username inserted in the URL.

```
git clone git@github.com:yourusername/warn-scraper.git
```

2.2.3 Install dependencies

You should change directory into folder where you code was downloaded.

```
cd warn-scraper
```

The *pipenv* package manager can install all of the Python tools necessary to run and test our code.

```
pipenv install --dev
```

Now install *pre-commit* to run a battery of automatic quick fixes against your work.

```
pipenv run pre-commit install
```

2.2.4 Create an issue

Before you being coding, you should visit our issue tracker and create a new ticket. You should try to clearly and succinctly describe the problem you are trying to solve. If you haven't done it before, GitHub has a guide on how to create an issue.

2.2.5 Create a branch

Next will we create a branch in your local repository where you an work without disturbing the mainline of code.

You can do this by running a line of code like the one below. You should substitute a branch that summarizes the work you're trying to do.

```
git checkout -b your-branch-name
```

For instance, if you were trying to add a scraper for the state of Iowa, you might name your branch *ia-scraper*.

```
git checkout-b ia-scraper
```

We ask that you follow a pattern where the branch name includes the postal code of the state you're working on, combined with the issue number generated by GitHub. In this example, work on the New Jersey scraper collected in GitHub issue #100.

```
git checkout -b nj-100
```

2.2.6 Perform your work

Now you can begin your work. You can start editing the code on your computer, making changes and running scripts to iterate toward your goal.

Creating a new scraper

When adding a new state, you should create a new Python file in the *warn/scrapers* named with the state's postal code. Here is an example of a starting point you can paste in to get going.

```
from pathlib import Path

from .. import utils
from ..cache import Cache


def scrape(
    data_dir: Path = utils.WARN_DATA_DIR,
    cache_dir: Path = utils.WARN_CACHE_DIR,
) -> Path:
    """
    Scrape data from Iowa.

    Keyword arguments:
    data_dir -- the Path were the result will be saved (default WARN_DATA_DIR)
    cache_dir -- the Path where results can be cached (default WARN_CACHE_DIR)

    Returns: the Path where the file is written
    """
    # Grab the page
    page = utils.get_url("https://xx.gov/yy.html")
    html = page.text

    # Write the raw file to the cache
    cache = Cache(cache_dir)
    cache.write("xx/yy.html", html)

    # Parse the source file and convert to a list of rows, with a header in the first␣
↪row.
    ## It's up to you to fill in the blank here based on the structure of the source file.
    ## You could do that here with BeautifulSoup or whatever other technique.
    pass

    # Set the path to the final CSV
    # We should always use the lower-case state postal code, like nj.csv
    output_csv = data_dir / "xx.csv"

    # Write out the rows to the export directory
    utils.write_rows_to_csv(output_csv, cleaned_data)

    # Return the path to the final CSV
    return output_csv


if __name__ == "__main__":
    scrape()
```

When creating a scraper, there are a few rules of thumb.

1. The raw data being scraped — whether it be HTML, CSV or PDF — should be saved to the cache unedited. We

aim to store pristine versions of our source data.

2. The data extracted from source files should be exported as a single file. Any intermediate files generated during data processing should not be written to the data folder. Such files should be written to the cache directory.
3. The final export should be the state's postal code, in lower case. For example, Iowa's final file should be saved as *ia.csv*.
4. For simple cases, use a cache name identical to the final export name.
5. If many files need to be cached, create a subdirectory using the lower-case state postal code and apply a sensible naming scheme to the cached files (e.g. *mo/page_1.html*).

Here's an example directory demonstrating the above conventions:

```
├── cache
│   ├── mo.csv
│   ├── nj
│   │   ├── Jan2010Warn.html
│   │   └── Jan2011Warn.html
│   ├── ny_raw_1.csv
│   └── ny_raw_2.csv
└── exports
    ├── mo.csv
    ├── nj.csv
    └── ny.csv
```

A fully featured example to learn from is Chris Zubak-Skees' scraper for Georgia.

Running the CLI

After a scraper has been created, the command-line tool provides a method to test code changes as you go. Run the following, and you'll see the standard help message.

```
pipenv run python -m warn.cli --help

Usage: python -m warn.cli [OPTIONS] [SCRAPERS]...

  Command-line interface for downloading WARN Act notices.

  SCRAPERS -- a list of one or more postal codes to scrape. Pass `all` to
  scrape all supported states and territories.

Options:
  --data-dir PATH                 The Path were the results will be saved
  --cache-dir PATH                The Path where results can be cached
  --delete / --no-delete          Delete generated files from the cache
  -l, --log-level [DEBUG|INFO|WARNING|ERROR|CRITICAL]
                                  Set the logging level
  --help                          Show this message and exit.
```

Running a state is as simple as passing arguments to that same command. If you were trying to develop an Iowa scraper found in the *warn/scrapers/ia.py* file, you could run something like this.

```
pipenv run python -m warn.cli IA
```

For more verbose logging, you can ask the system to showing debugging information.

```
pipenv run python -m warn.cli IA -l DEBUG
```

You could continue to iterate with code edits and CLI runs until you've completed your goal.

2.2.7 Run tests

Before you submit your work for inclusion in the project, you should run our tests to identify bugs. Testing is implemented via pytest. Run the tests with the following.

```
make test
```

If any errors, arise, carefully read the traceback message to determine what needs to be repaired.

2.2.8 Push to your fork

Once you're happy with your work and the tests are passing, you should commit your work and push it to your fork.

```
git commit -am "Describe your work here"
git push -u origin your-branch-name
```

If there have been significant changes to the main branch since you started work, you should consider integrating those edits to your branch since any differences will need to be reconciled before your code can be merged.

```
# Checkout and pull updates on main
git checkout main
git pull

# Checkout your branch again
git checkout your-branch-name

# Rebase your changes on top of main
git rebase main
```

If any code conflicts arise, you can open the listed files and seek to reconcile them yourself. If you need help, reach out to the maintainers.

Once that's complete, commit any changes and push again to your fork's branch.

```
git commit -am "Merged in main"
git push origin your-branch-name
```

2.2.9 Send a pull request

The final step is to submit a pull request to the main respository, asking the maintainers to consider integrating your patch. GitHub has a short guide that can walk you through the process. You should tag your issue number in the request so that they linked in GitHub's system.

2.3 Releasing

How to release a new version of the warn-scraper module to the Python Package Index.

2.3.1 How it works

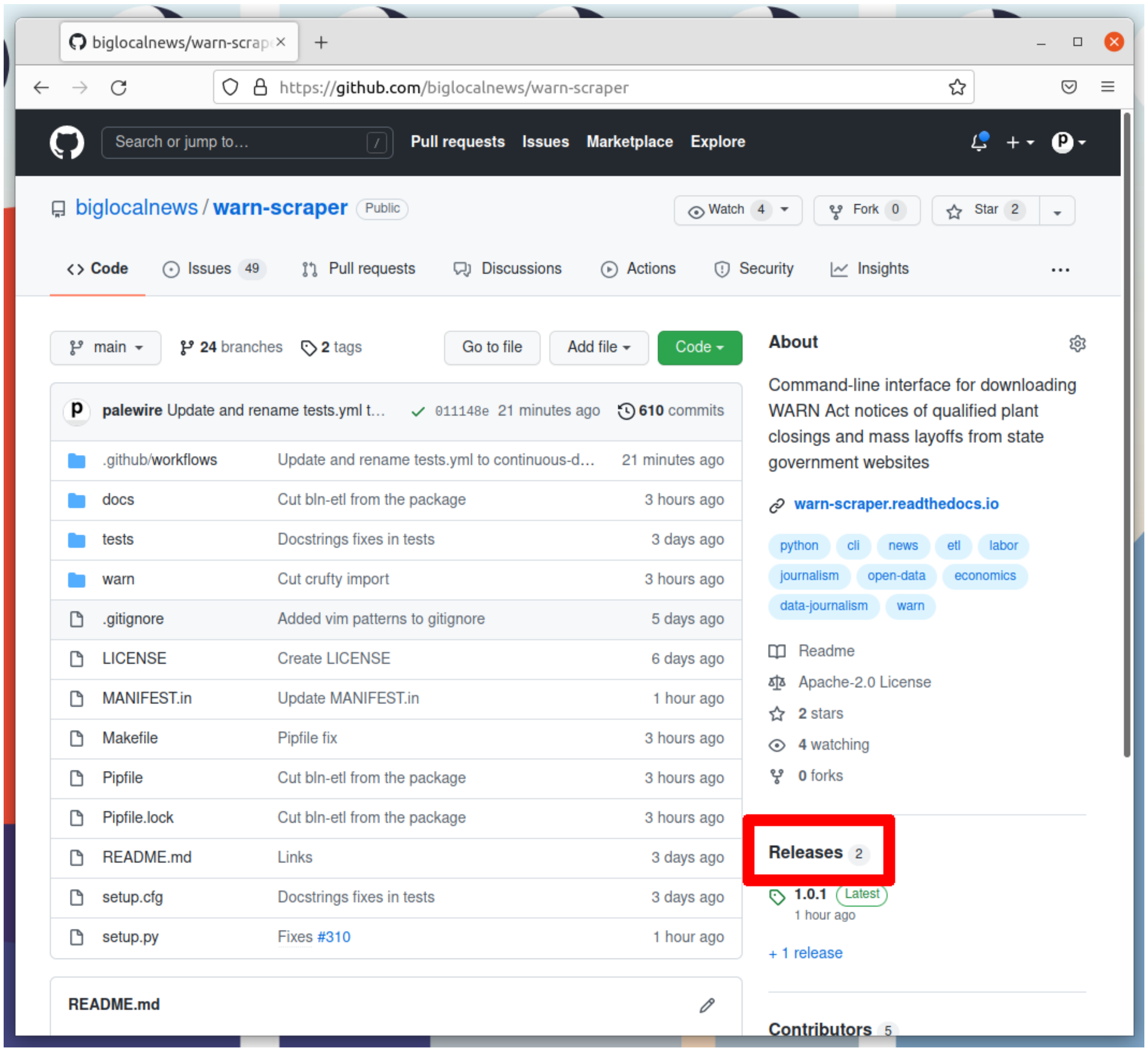
Our release process is automated as a continuous deployment via the GitHub Actions framework. The logic that governs the process is stored in the `workflows` directory.

That means that everything necessary to make a release can be done with a few clicks on the GitHub website. All you need to do is make a tagged release at biglocalnews/warn-scraper/releases, then wait for the computers to handle the job.

Here's how it's done, step by step.

2.3.2 1. Go to the releases page

The first step is to visit our repository's homepage and click on the "releases" headline in the right rail.



palewire Update and rename tests.yml t… ✓ 011148e 21 minutes ago 610 commits

.github/workflows	Update and rename tests.yml to continuous-d…	21 minutes ago
docs	Cut bln-etl from the package	3 hours ago
tests	Docstrings fixes in tests	3 days ago
warn	Cut crufty import	3 hours ago
.gitignore	Added vim patterns to gitignore	5 days ago
LICENSE	Create LICENSE	6 days ago
MANIFEST.in	Update MANIFEST.in	1 hour ago
Makefile	Pipfile fix	3 hours ago
Pipfile	Cut bln-etl from the package	3 hours ago
Pipfile.lock	Cut bln-etl from the package	3 hours ago
README.md	Links	3 days ago
setup.cfg	Docstrings fixes in tests	3 days ago
setup.py	Fixes #310	1 hour ago

README.md

About

Command-line interface for downloading WARN Act notices of qualified plant closings and mass layoffs from state government websites

warn-scraper.readthedocs.io

python cli news etl labor journalism open-data economics data-journalism warn

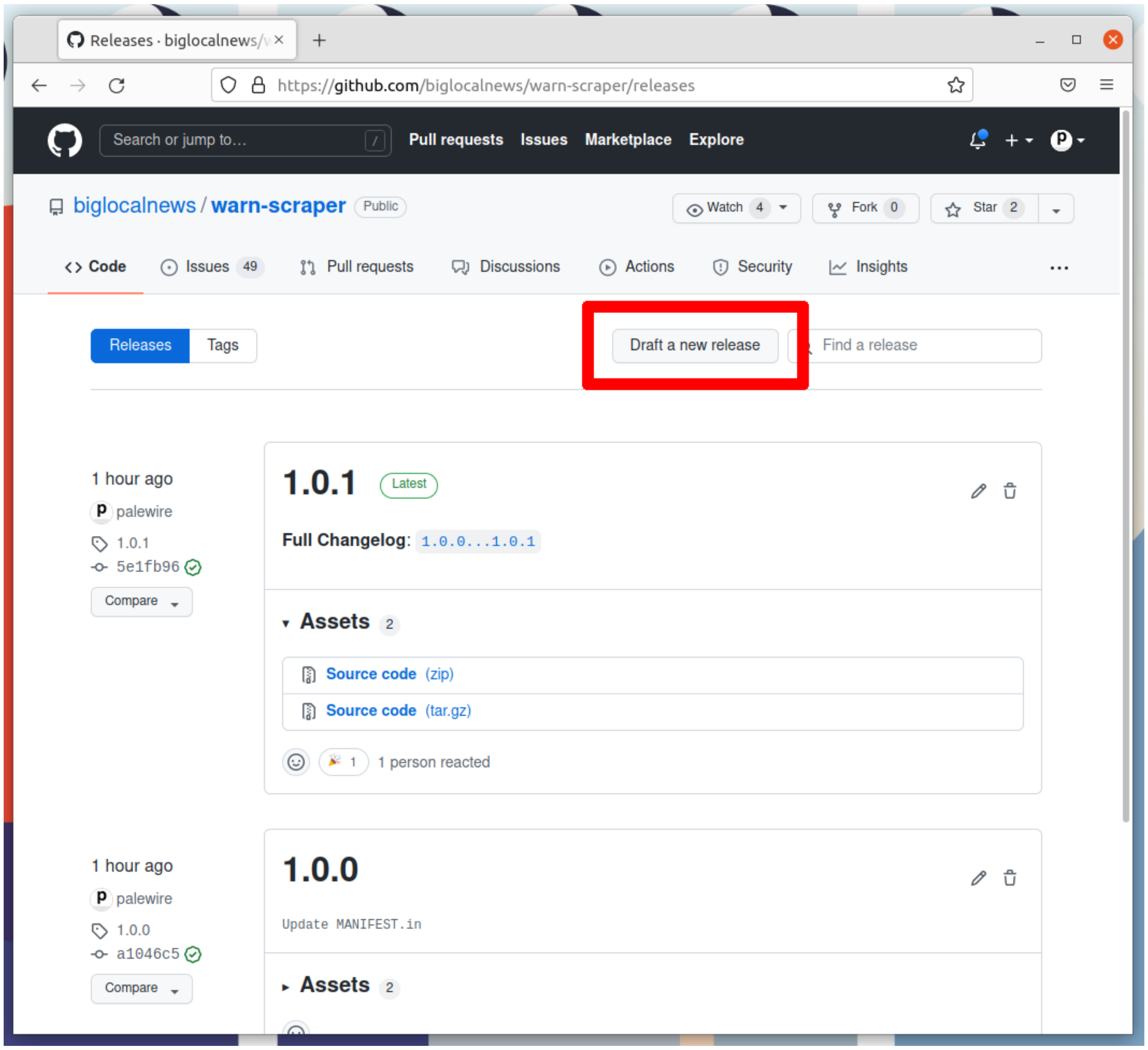
Readme
Apache-2.0 License
2 stars
4 watching
0 forks

Releases 2
1.0.1 Latest
1 hour ago
+ 1 release

Contributors 5

2.3.3 2. Click ‘Draft a new release’

Note the number of the latest release. Click the “Draft a new release” button in the upper-right corner. If you don’t see this button, you do not have permission to make a release. Only the maintainers of the repository are able to release new code.



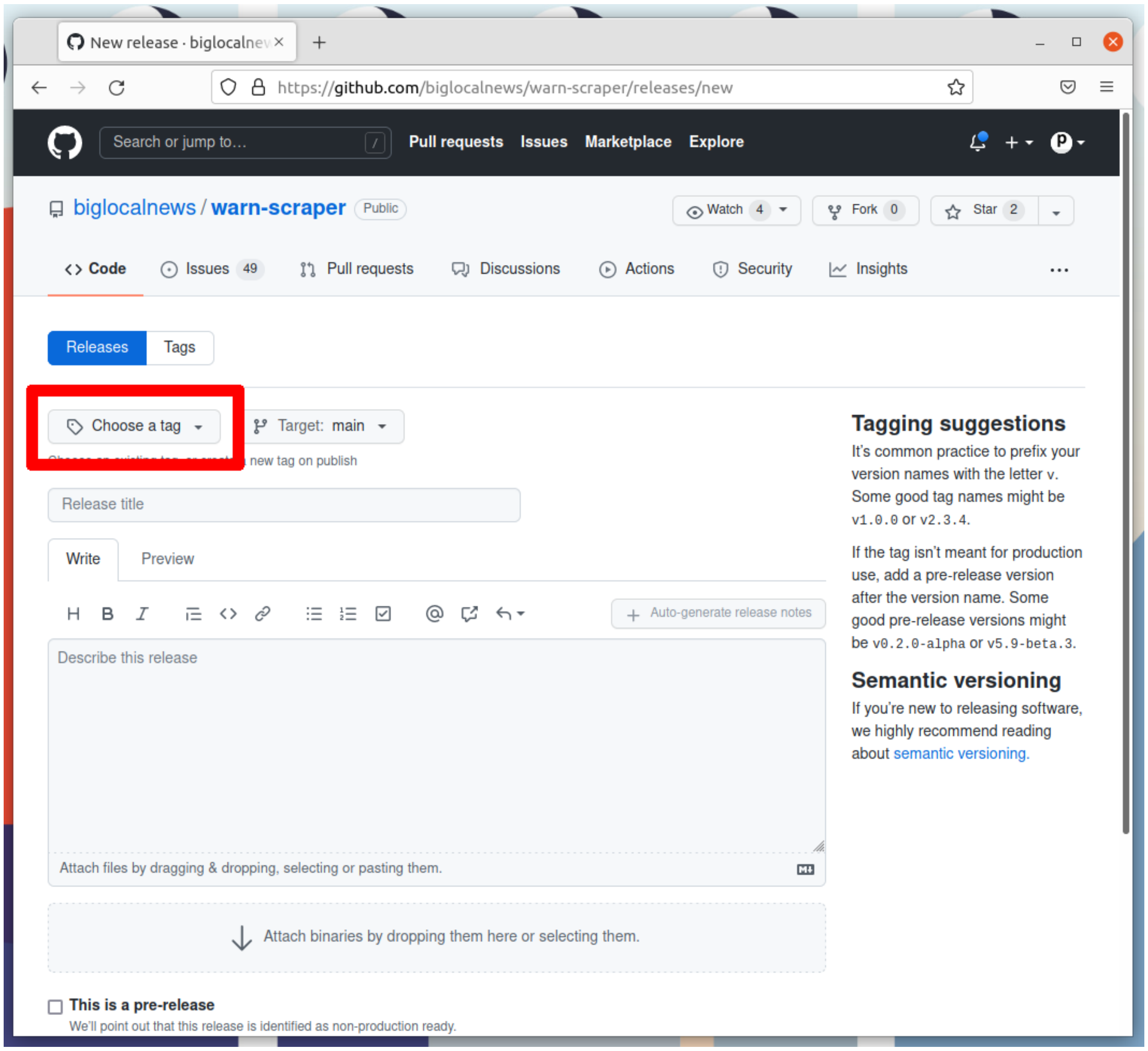
2.3.4 3. Create a new tag

Think about how big your changes are and decide if you're a major, minor or patch release.

All version numbers should feature three numbers separated by the periods, like `1.0.1`. If you're making a major release that isn't backwards compatible, the latest release's first number should go up by one. If you're making a minor release by adding a feature or major a large change, the second number should go up. If you're only fixing bugs or making small changes, the third number should go up.

If you're unsure, review the standards defined at semver.org to help make a decision. In the end don't worry about it too much. Our version numbers don't need to be perfect. They just need to be three numbers separated by periods.

Once you've settled on the number for your new release, click on the "Choose a tag" pull down.


New release · biglocalnews
https://github.com/biglocalnews/warn-scraper/releases/new
Search or jump to…
Pull requests
Issues
Marketplace
Explore
biglocalnews / warn-scraper
Public
Watch 4
Fork 0
Star 2
Code
Issues 49
Pull requests
Discussions
Actions
Security
Insights
Releases
Tags
Choose a tag
Target: main
new tag on publish
Release title
Write
Preview
Auto-generate release notes
Describe this release
Attach files by dragging & dropping, selecting or pasting them.
Attach binaries by dropping them here or selecting them.
This is a pre-release
We'll point out that this release is identified as non-production ready.

Tagging suggestions

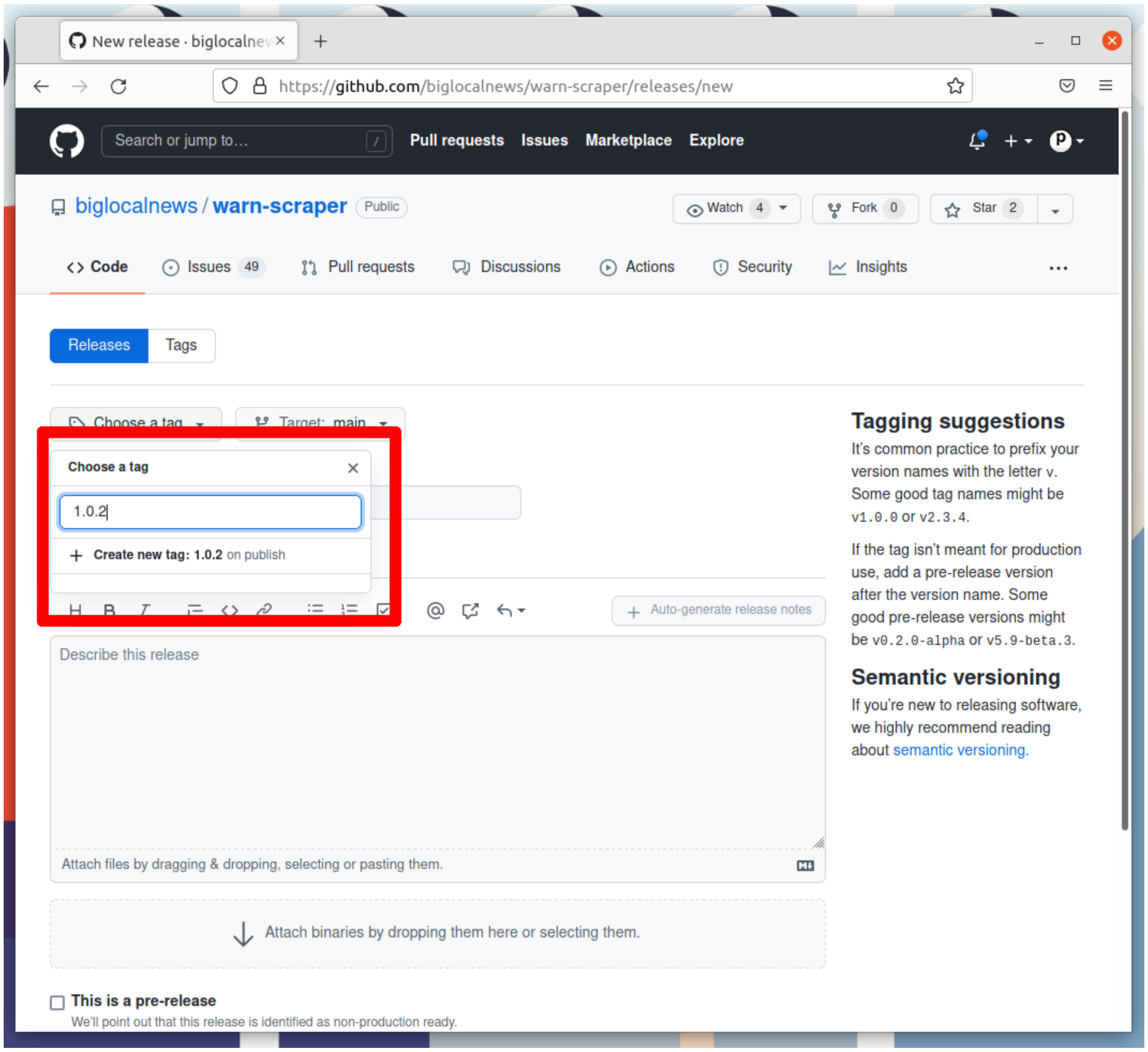
It's common practice to prefix your version names with the letter v. Some good tag names might be v1.0.0 or v2.3.4.

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 reading about semantic versioning.

Enter your version number into the box. Then click the "Create new tag" option that appears.


New release · biglocalnews
https://github.com/biglocalnews/warn-scraper/releases/new
Search or jump to...
Pull requests
Issues
Marketplace
Explore
biglocalnews / warn-scraper
Public
Watch 4
Fork 0
Star 2
Code
Issues 49
Pull requests
Discussions
Actions
Security
Insights
Releases
Tags
Choose a tag
Target: main
Choose a tag
1.0.2
Create new tag: 1.0.2 on publish
Auto-generate release notes
Describe this release
Attach files by dragging & dropping, selecting or pasting them.
Attach binaries by dropping them here or selecting them.
This is a pre-release
We'll point out that this release is identified as non-production ready.

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`.

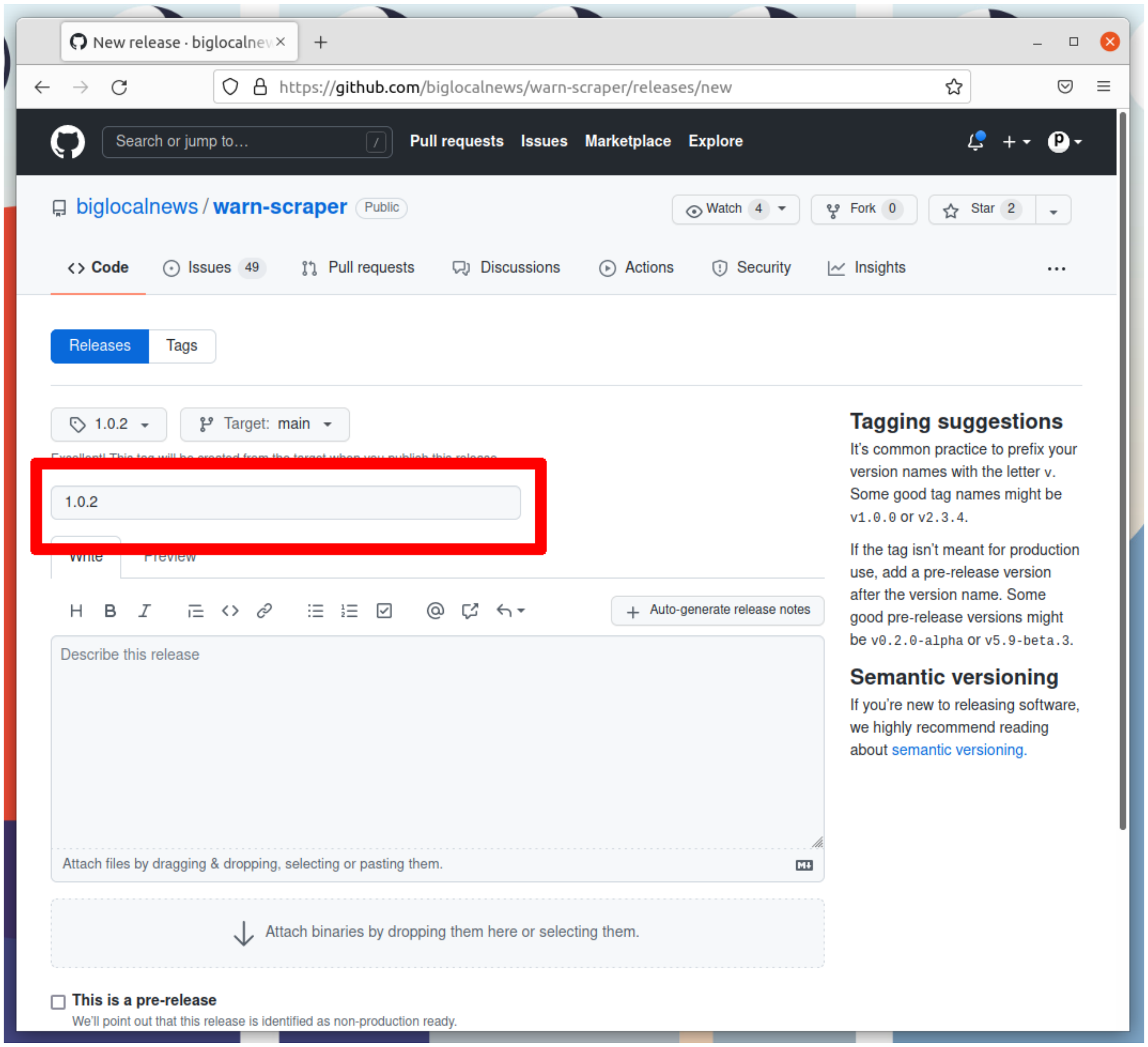
If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be `v0.2.0-alpha` or `v5.9-beta.3`.

Semantic versioning

If you're new to releasing software, we highly recommend reading about semantic versioning.

2.3.5 4. Name the release

Enter the same number into the "Release title" box.


New release · biglocalnew
https://github.com/biglocalnews/warn-scraper/releases/new
Search or jump to…
Pull requests
Issues
Marketplace
Explore
biglocalnews / warn-scraper
Public
Watch 4
Fork 0
Star 2
Code
Issues 49
Pull requests
Discussions
Actions
Security
Insights
Releases
Tags
1.0.2
Target: main
1.0.2
Write
Preview
Auto-generate release notes
Describe this release
Attach files by dragging & dropping, selecting or pasting them.
Attach binaries by dropping them here or selecting them.
This is a pre-release
We'll point out that this release is identified as non-production ready.

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`.

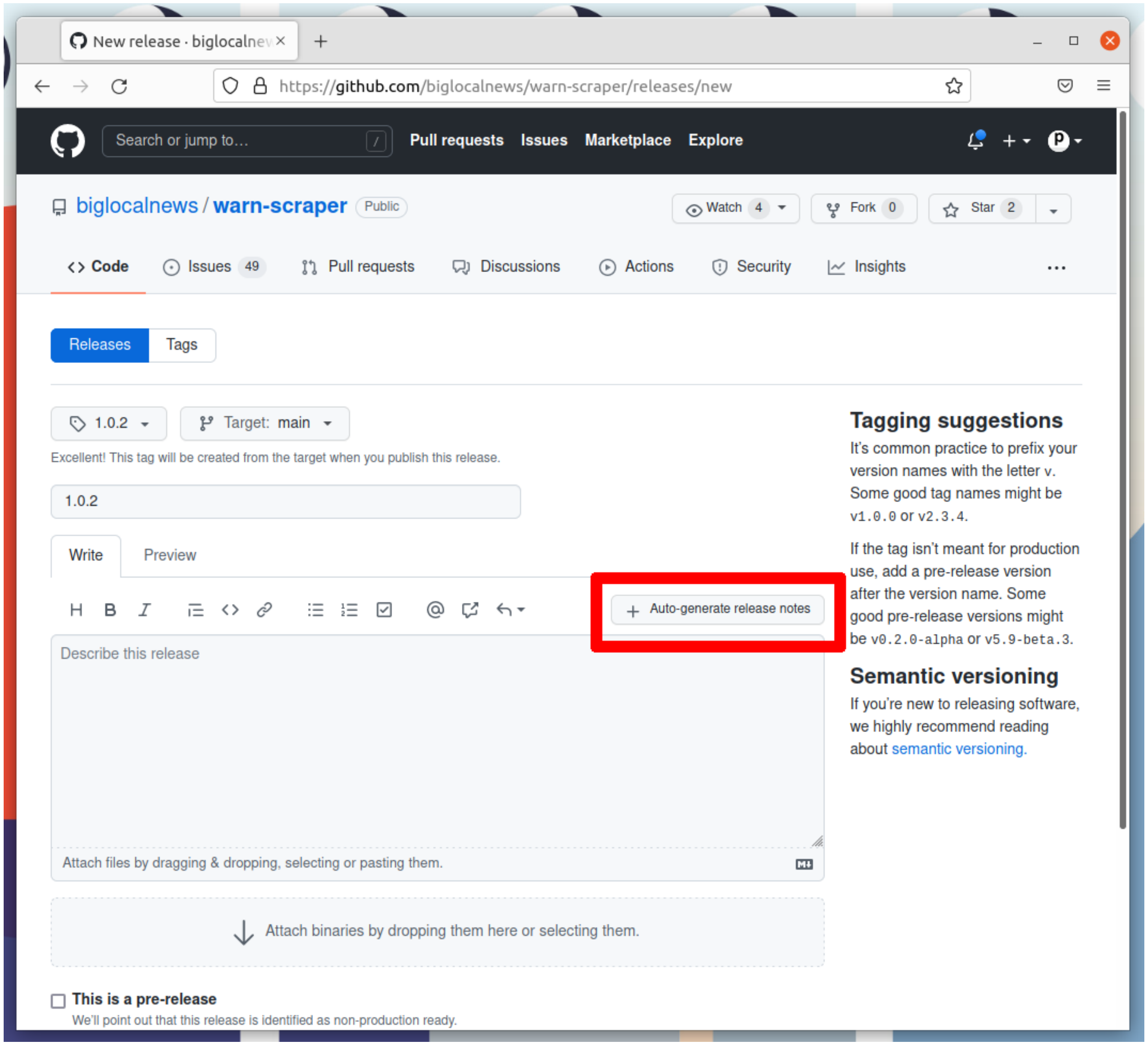
If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be `v0.2.0-alpha` or `v5.9-beta.3`.

Semantic versioning

If you're new to releasing software, we highly recommend reading about semantic versioning.

2.3.6 5. Auto-generate release notes

Click the "Auto-generate release notes" button in the upper right corner of the large description box.


New release · biglocalnews
https://github.com/biglocalnews/warn-scraper/releases/new
Search or jump to...
Pull requests
Issues
Marketplace
Explore
biglocalnews / warn-scraper
Public
Watch 4
Fork 0
Star 2
Code
Issues 49
Pull requests
Discussions
Actions
Security
Insights
Releases
Tags
1.0.2
Target: main
Write
Preview
Auto-generate release notes
Describe this release
Attach files by dragging & dropping, selecting or pasting them.
Attach binaries by dropping them here or selecting them.

Excellent! This tag will be created from the target when you publish this release.

1.0.2

This is a pre-release

We'll point out that this release is identified as non-production ready.

Tagging suggestions

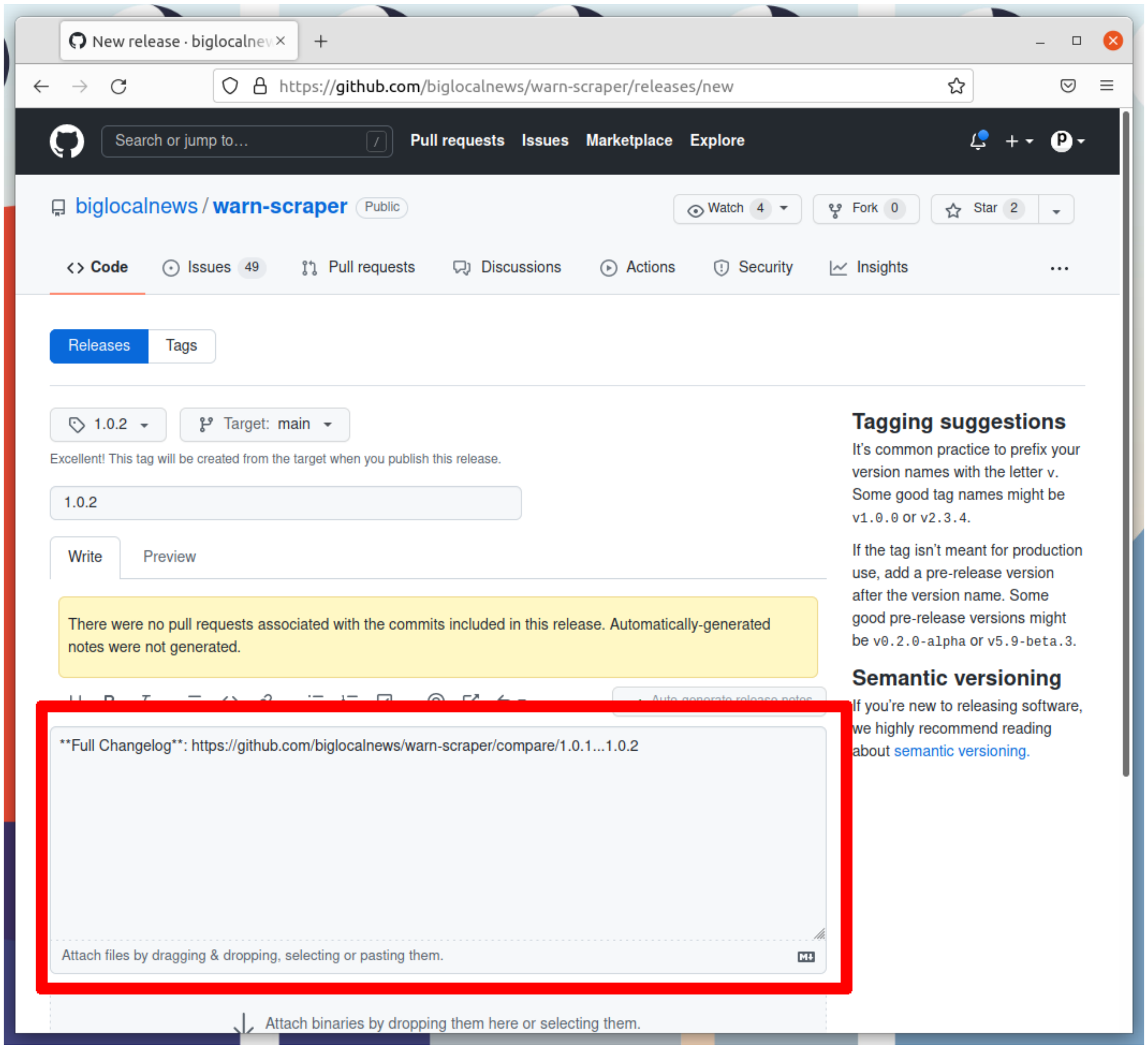
It's common practice to prefix your version names with the letter v. Some good tag names might be `v1.0.0` or `v2.3.4`.

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 reading about semantic versioning.

That should fill in the box below. What appears will depend on how many pull requests you've merged since the last release.


New release · biglocalnews
https://github.com/biglocalnews/warn-scraper/releases/new
Search or jump to…
Pull requests
Issues
Marketplace
Explore
biglocalnews / warn-scraper
Public
Watch 4
Fork 0
Star 2
Code
Issues 49
Pull requests
Discussions
Actions
Security
Insights
Releases
Tags
1.0.2
Target: main
Write
Preview
Auto-generate release notes
Attach binaries by dropping them here or selecting them.

Excellent! This tag will be created from the target when you publish this release.

1.0.2

There were no pull requests associated with the commits included in this release. Automatically-generated notes were not generated.

Full Changelog: https://github.com/biglocalnews/warn-scraper/compare/1.0.1...1.0.2

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.

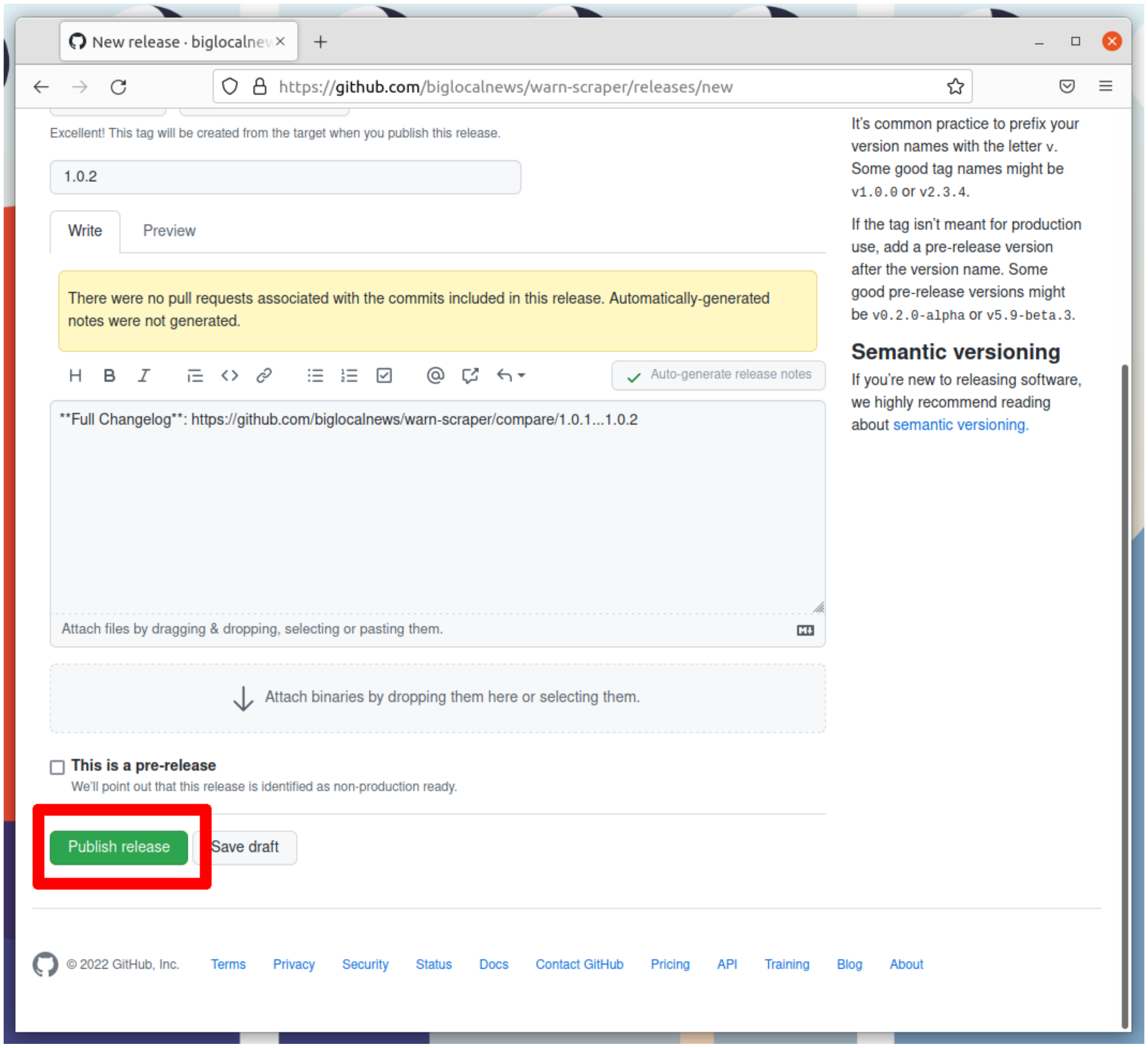
If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be v0.2.0-alpha or v5.9-beta.3.

Semantic versioning

If you're new to releasing software, we highly recommend reading about semantic versioning.

2.3.7 6. Publish the release

Click the green button that says "Publish release" at the bottom of the page.



Excellent! This tag will be created from the target when you publish this release.

1.0.2

There were no pull requests associated with the commits included in this release. Automatically-generated notes were not generated.

Full Changelog: https://github.com/biglocalnews/warn-scraper/compare/1.0.1...1.0.2

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

Attach binaries by dropping them here or selecting them.

This is a pre-release
We'll point out that this release is identified as non-production ready.

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`.

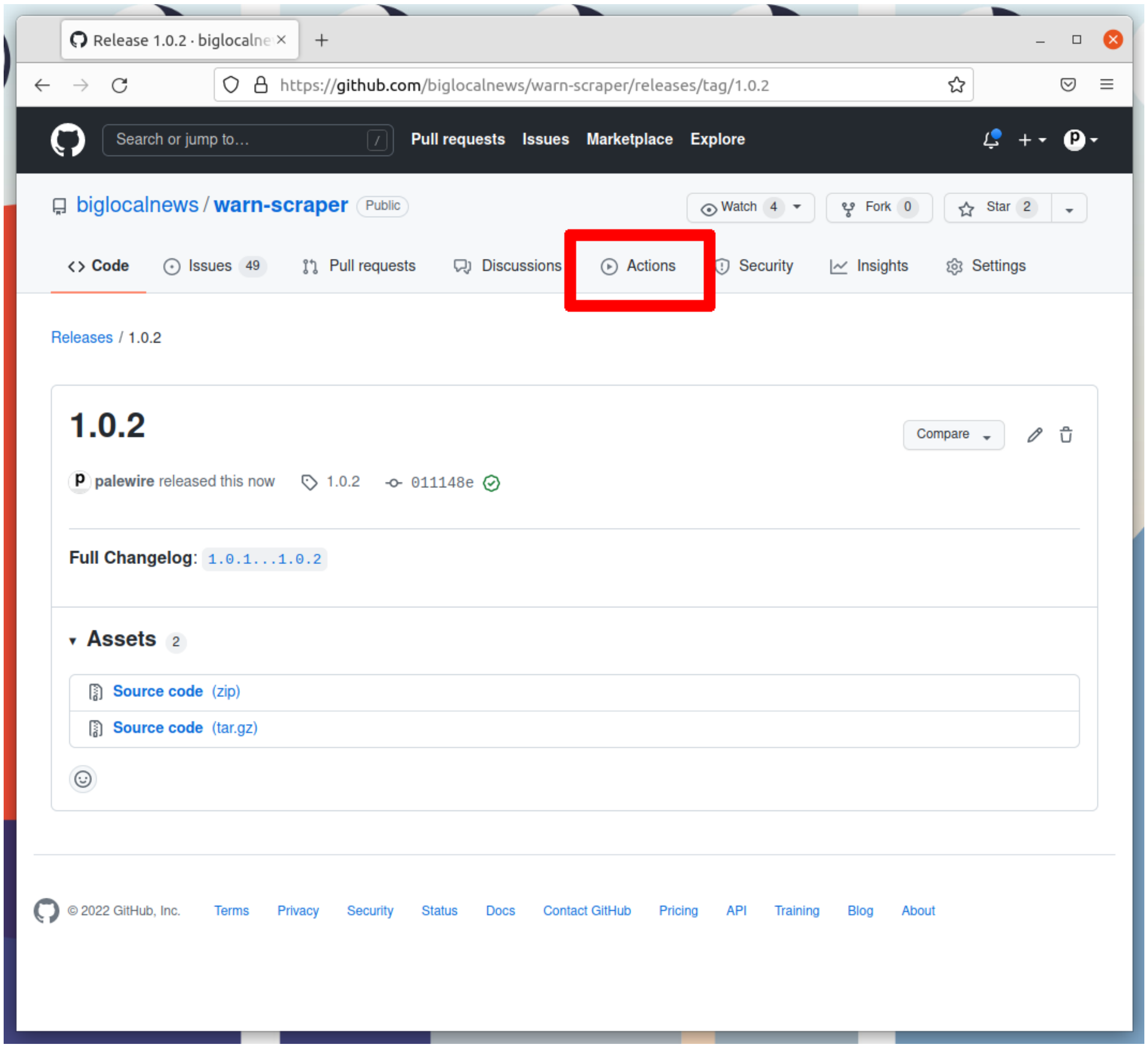
If the tag isn't meant for production use, add a pre-release version after the version name. Some good pre-release versions might be `v0.2.0-alpha` or `v5.9-beta.3`.

Semantic versioning

If you're new to releasing software, we highly recommend reading about semantic versioning.

2.3.8 7. Wait for the Action to finish

GitHub will take you to a page dedicated to your new release and start an automated process that release our new version to the world. Follow its progress by clicking on the Actions tab near the top of the page.



Releases / 1.0.2

1.0.2

palewire released this now 1.0.2 011148e

Full Changelog: `1.0.1...1.0.2`

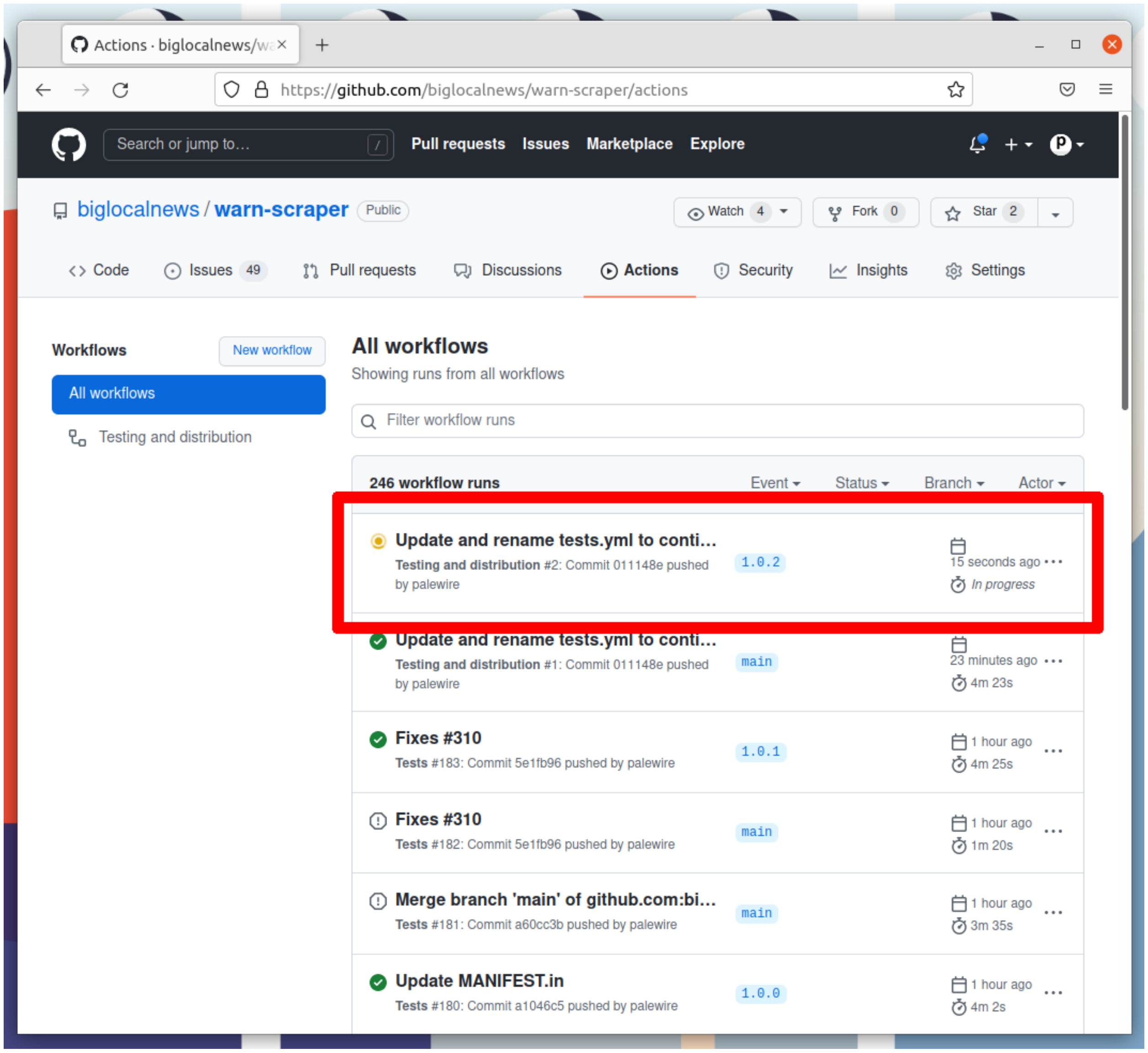
Assets 2

Source code (zip)

Source code (tar.gz)

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

That will take you to the Actions monitoring page. The task charged with publishing your release should be at the top.



All workflows

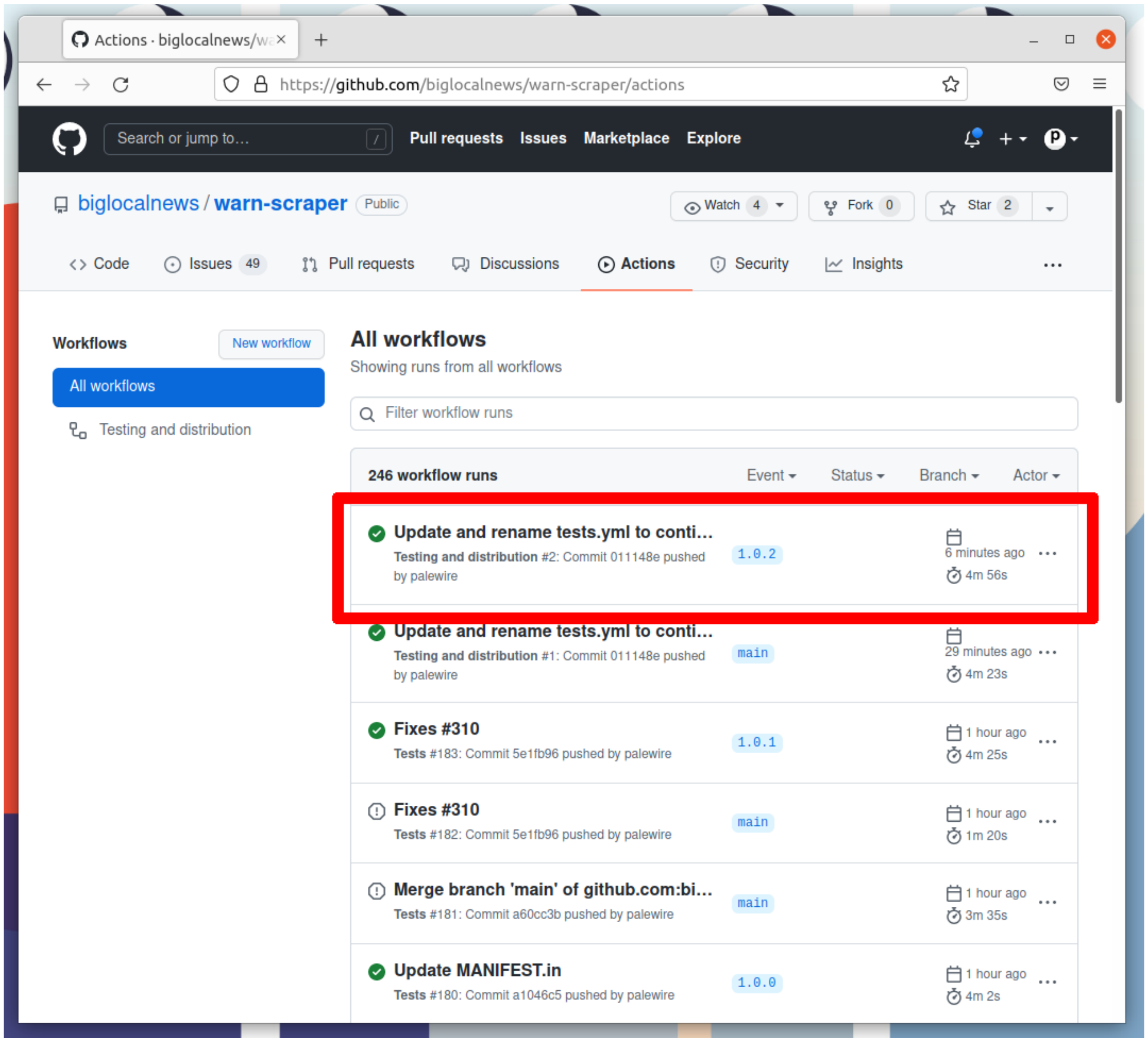
Showing runs from all workflows

246 workflow runs

Run	Branch	Time	Duration
Update and rename tests.yml to conti… — Testing and distribution #2: Commit 011148e pushed by palewire	1.0.2	15 seconds ago	In progress
Update and rename tests.yml to conti… — Testing and distribution #1: Commit 011148e pushed by palewire	main	23 minutes ago	4m 23s
Fixes #310 — Tests #183: Commit 5e1fb96 pushed by palewire	1.0.1	1 hour ago	4m 25s
Fixes #310 — Tests #182: Commit 5e1fb96 pushed by palewire	main	1 hour ago	1m 20s
Merge branch 'main' of github.com:bi… — Tests #181: Commit a60cc3b pushed by palewire	main	1 hour ago	3m 35s
Update MANIFEST.in — Tests #180: Commit a1046c5 pushed by palewire	1.0.0	1 hour ago	4m 2s

2.3.9 8. Check the results on PyPI

After a few minutes, the process there should finish and show a green check mark.



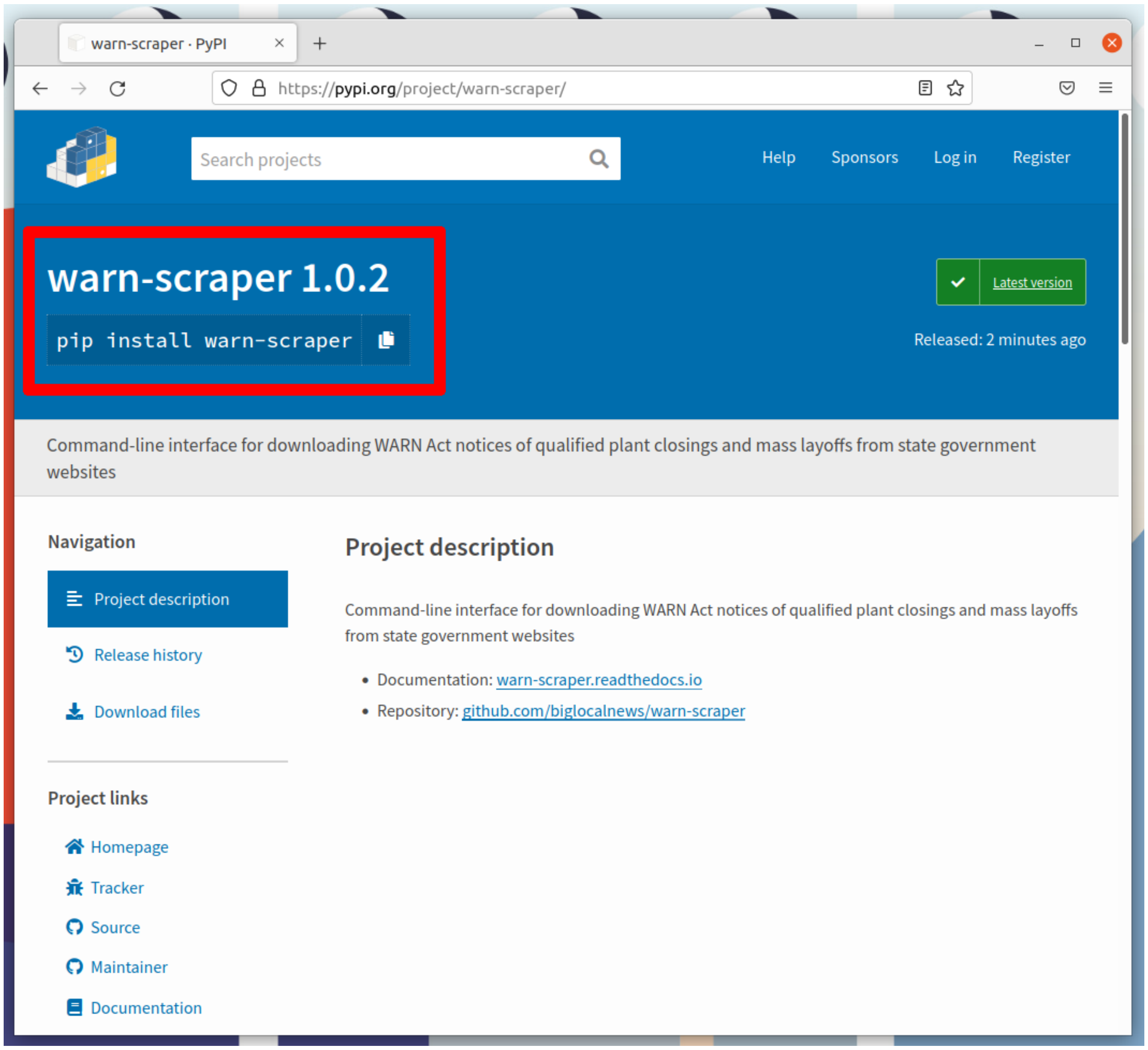
All workflows

Showing runs from all workflows

246 workflow runs

Run	Branch	Time
Update and rename tests.yml to conti… Testing and distribution #2: Commit 011148e pushed by palewire	1.0.2	6 minutes ago, 4m 56s
Update and rename tests.yml to conti… Testing and distribution #1: Commit 011148e pushed by palewire	main	29 minutes ago, 4m 23s
Fixes #310 Tests #183: Commit 5e1fb96 pushed by palewire	1.0.1	1 hour ago, 4m 25s
Fixes #310 Tests #182: Commit 5e1fb96 pushed by palewire	main	1 hour ago, 1m 20s
Merge branch 'main' of github.com:bi… Tests #181: Commit a60cc3b pushed by palewire	main	1 hour ago, 3m 35s
Update MANIFEST.in Tests #180: Commit a1046c5 pushed by palewire	1.0.0	1 hour ago, 4m 2s

When it does, visit warn-scraper's page on PyPI, where you should see the latest version displayed at the top of the page.


warn-scraper · PyPI

https://pypi.org/project/warn-scraper/

warn-scraper 1.0.2

```
pip install warn-scraper
```

Latest version

Released: 2 minutes ago

Command-line interface for downloading WARN Act notices of qualified plant closings and mass layoffs from state government websites

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage
- Tracker
- Source
- Maintainer
- Documentation

Project description

Command-line interface for downloading WARN Act notices of qualified plant closings and mass layoffs from state government websites

- Documentation: warn-scraper.readthedocs.io
- Repository: github.com/biglocalnews/warn-scraper

If the action fails, something has gone wrong with the deployment process. You can click into its debugging panel to search for the cause or ask the project maintainers for help.

2.4 Sources

There are currently scrapers for 39 of America's 56 states and territories.

State	Source	Docs	Authors	Tags
Alabama			Dilcia19, zstumgoren	html
Alaska			Dilcia19, zstumgoren	html
Arizona			Dilcia19, zstumgoren	jobcenter
California			Dilcia19, ydoc5212, zstumgoren	excel, html, pdf
Colorado			anikasikka	html
Connecticut			Dilcia19, stucka, zstumgoren	html

continues on next page

Table 1 – continued from previous page

State	Source	Docs	Authors	Tags
Delaware			Dilcia19, zstumgoren	jobcenter
Florida			Dilcia19, shallotly, zstumgoren	html, pdf
Georgia			chriszs	html
Idaho			chriszs	pdf
Illinois			chriszs	excel, html
Indiana			Dilcia19, zstumgoren	html
Iowa			Dilcia19, palewire, shallotly, zstumgoren	excel, html
Kansas			Dilcia19, zstumgoren	jobcenter
Kentucky			palewire	excel
Louisiana			chriszs	html, pdf
Maine			zstumgoren	jobcenter
Maryland			Dilcia19, shallotly, zstumgoren	html
Michigan			anikasikka	html, pdf
Missouri			Dilcia19, shallotly, zstumgoren	html
Montana			ydoc5212, zstumgoren	excel, html
Nebraska			Dilcia19, zstumgoren	html
New Jersey			Dilcia19, palewire, zstumgoren	html
New Mexico			chriszs	pdf
New York			Dilcia19, palewire, ydoc5212, zstumgoren	excel, historical
Ohio			Dilcia19, chriszs, zstumgoren	html, pdf
Oklahoma			Dilcia19, zstumgoren	jobcenter
Oregon			Dilcia19, ydoc5212, zstumgoren	excel, historical
Rhode Island			Dilcia19, chriszs, ydoc5212, zstumgoren	excel
South Carolina			palewire	html, pdf
South Dakota			Dilcia19, ydoc5212, zstumgoren	html
Tennessee			anikasikka	html, pdf
Texas			Dilcia19, ydoc5212	excel, historical, html
Utah			Dilcia19, zstumgoren	html
Vermont			zstumgoren	jobcenter
Virginia			Dilcia19, shallotly, zstumgoren	csv, html
Washington			Dilcia19, zstumgoren	html
Wisconsin			Dilcia19, ydoc5212, zstumgoren	html
District of Columbia			Dilcia19, shallotly, zstumgoren	html

2.4.1 To do

These 17 areas need a scraper:

- Arkansas
- Hawaii
- Massachusetts
- Minnesota
- Mississippi
- Nevada
- New Hampshire
- North Carolina

- North Dakota
- Pennsylvania
- West Virginia
- Wyoming
- American Samoa
- Guam
- Northern Mariana Islands
- Puerto Rico
- Virgin Islands

2.5 Reference

Documentation for a selection of our system's common internal tools

Table of contents

2.5.1 Makefile

A number of custom commands are available to developers via make. They offer options for quickly running tests, working with the docs and executing scrapers.

```
build-release        builds source and wheel package
check-release        check release for potential errors
coverage             check code coverage
format               automatically format Python code with black
help                 Show this help. Example: make help
lint                 run the linter
mypy                 run mypy type checks
run                  run a scraper. example: `make run scraper=IA`
serve-docs           start the documentation test server
tally-sources        update sources dashboard in the docs
test-docs            build the docs as html
test                 run all tests
```

For example, you can run this from the root of the project.

```
make run scraper=IA
```

2.5.2 Caching

The *Cache* class is used to save the raw HTML, PDFs and CSVs files our scrapers collect.

2.5.3 Utilities

The utils module contains a variety of variables and functions used by our scrapers.

2.5.4 Research

CRS

A 2013 summary by the Congressional Research Service

GAO

A 2003 audit by the Government Accountability Office

CHAPTER THREE

LINKS

- Documentation: warn-scraper.readthedocs.io
- Repository: github.com/biglocalnews/warn-scraper
- Packaging: pypi.org/project/warn-scraper

CHAPTER FOUR

ABOUT

The project is sponsored by Big Local News, a program at Stanford University that collects data for impactful journalism. The code is maintained by Stanford Lecturer Serdar Tumgoren and Ben Welsh, a visiting data journalist from the Los Angeles Times.