bln

unknown

Sep 20, 2023

CONTENTS

Python client for the biglocalnews.org API

CHAPTER

ONE

INSTALLATION

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

```
pipenv install bln
```

CHAPTER

TWO

DOCUMENTATION

2.1 Getting started

Sections

2.1.1 Setup

Installation

The Python SDK can be installed using `pip`.

```
pip install bln
```

Getting an API Key

API keys can be generated at https://biglocalnews.org/.

1. Log into Big Local News using your Google account.
2. Expand the Developer menu item in the left sidebar.
3. Click Manage Keys.
4. Generate a key and copy it to the clipboard.
5. Export the token to your environment

```
export BLN_API_TOKEN = "<Paste your API Key here>"
```

Initializing the SDK client

The client is what will handle calls to the Big Local News API. To set up it you must authorize the client using your API key.

```
from bln import Client

client = Client()
```

2.1.2 Working with projects

Creating a project

To create a project, use the client's `createProject` method. The only argument required to create a project is a name. However, it is helpful to include a description. The `createProject` method will return a dictionary of project metadata if it is successful and `None` if it fails.

```
project_name = "Big Local News SDK Demo Project"
project_description = "This is a sample project to show users how the SDK works"

project = client.createProject(project_name, description=project_description)
```

Get metadata for an existing project

Helper methods can assist you with selecting projects by name or id. It will return one and only one project.

```
client.get_project_by_name("Big Local News SDK Demo Project")
client.get_project_by_id(project_id)
```

The client's `search_projects` method can be used to for more complex queries. It takes a lambda function that returns `True` or `False` based on whether or not the project metadata meets the search criteria. The `search_projects` method will return a list of project metadata for those projects matching the search query.

```
client.search_projects(lambda project: "SDK" in project["description"])
```

Updating project metadata

The client's `updateProject` method is used to edit project metadata. It takes the project's ID as a required argument and optional keyword arguments to update the metadata. The method will return the updated project metadata.

```
new_name = "Big Local News SDK Demo Project - edited"
new_description = "This is a sample project to show users how the SDK works. The␣
↪description and name have been edited."

client.updateProject(project_id, name=new_name, description=new_description)
```

Deleting a project

Use the client's `deleteProject` method to delete a project. The method requires a project ID as it's sole argument.

WARNING: This process is irreversible. Use with caution.

```
client.deleteProject(project_id)
```

2.1.3 Working with files

The SDK client has a host of methods available for working with files. All of the methods will require a project ID as an argument.

Uploading

The client has two different methods for file uploads - one for a single file and a second for batch uploads.

A single file

```
client.upload_file(project_id, "./data/demo_a.csv")
```

Multiple files

The SDK client also has an `upload_files` method which takes a list of files as an argument.

```
client.upload_files(project_id, files_to_upload)
```

Viewing files in a project

```
# Get the first project returned from a search
project = client.search_projects(lambda project: project["id"] == project_id)[0]

project["files"]

[
    {
        "createdAt": "2022-01-12T23:40:44.443000+00:00",
        "name": "demo_a.csv",
        "tags": [],
        "updatedAt": "2022-01-12T23:40:44.443000+00:00",
    },
    {
        "createdAt": "2022-01-12T23:40:46.372000+00:00",
        "name": "demo_b.csv",
        "tags": [],
        "updatedAt": "2022-01-12T23:40:46.372000+00:00",
    },
]
```

Downloading a file

The client's `download_file` takes three arguments: project ID, filename and an optional output directory. If an output directory is not specified, the client will download to the current working directory.

```
client.download_file(project_id, "demo_a.csv", output_dir="./data")
```

2.2 pandas extensions

The `bln` package includes extensions for reading and writing to biglocalnews.org with the pandas data analysis library.

They can be easily imported into your environment. First import pandas as usual.

```
from pandas import pd
```

Then import the Big Local News package and connect it with your `pd` module.

```
import bln

bln.pandas.register(pd)
```

Your standard `pd` object now contains a set of custom methods for interacting with biglocalnews.org. This is accomplished by "monkey patching" the pandas library.

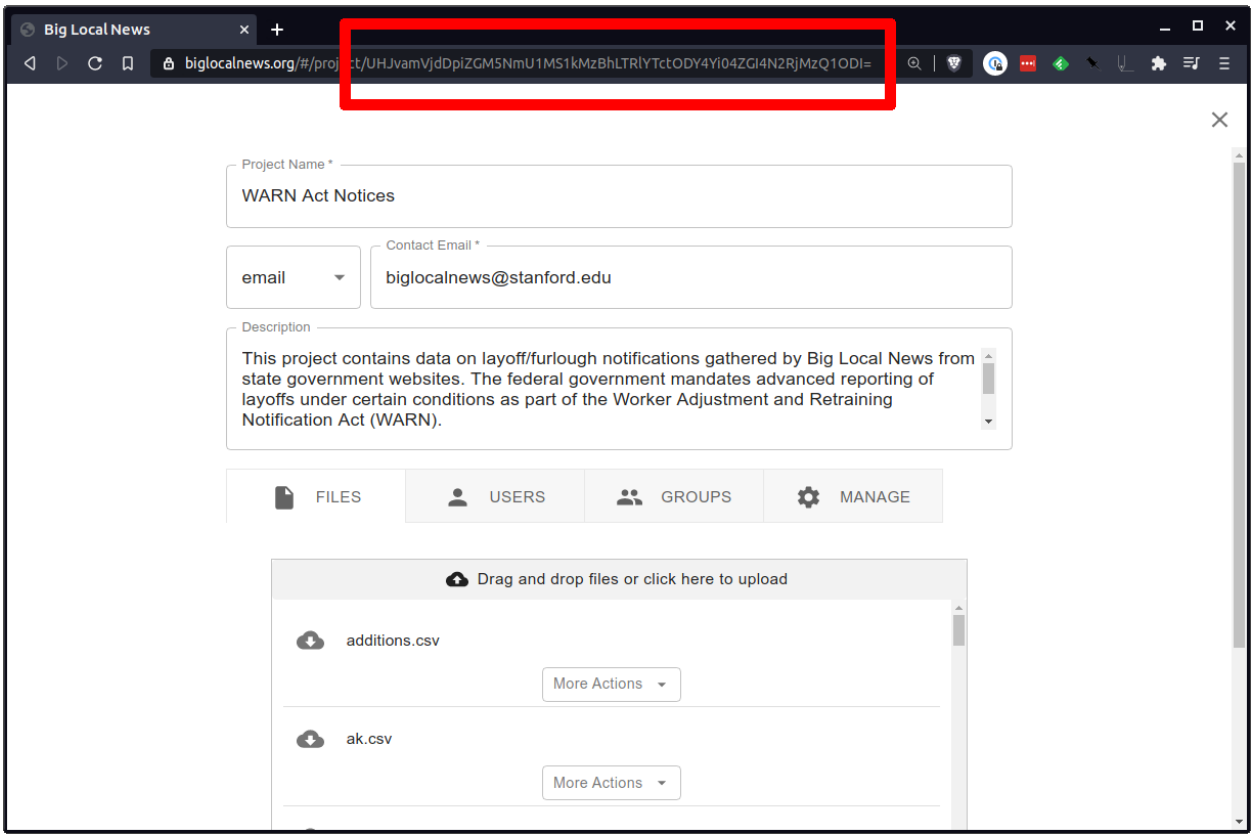
2.2.1 Reading data

You can read in files from biglocalnews.org as a pandas dataframe using the `read_bln` function. It requires three inputs:

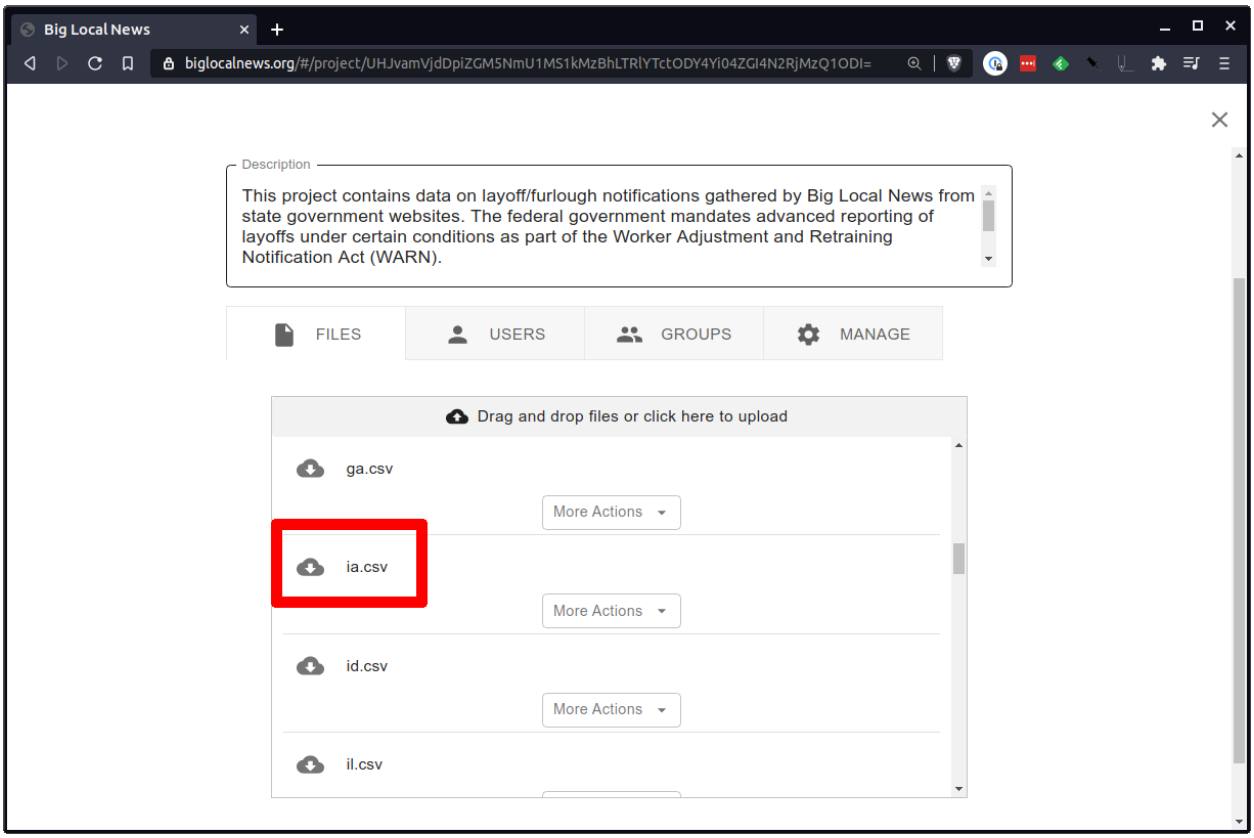
1. The unique identifier of the biglocalnews.org project where the file is stored
2. The name of the file within the biglocalnews.org project
3. An API key from biglocalnews.org with permission to read from the project.

```
df = pd.read_bln(your_project_id, your_file_name, your_api_token)
```

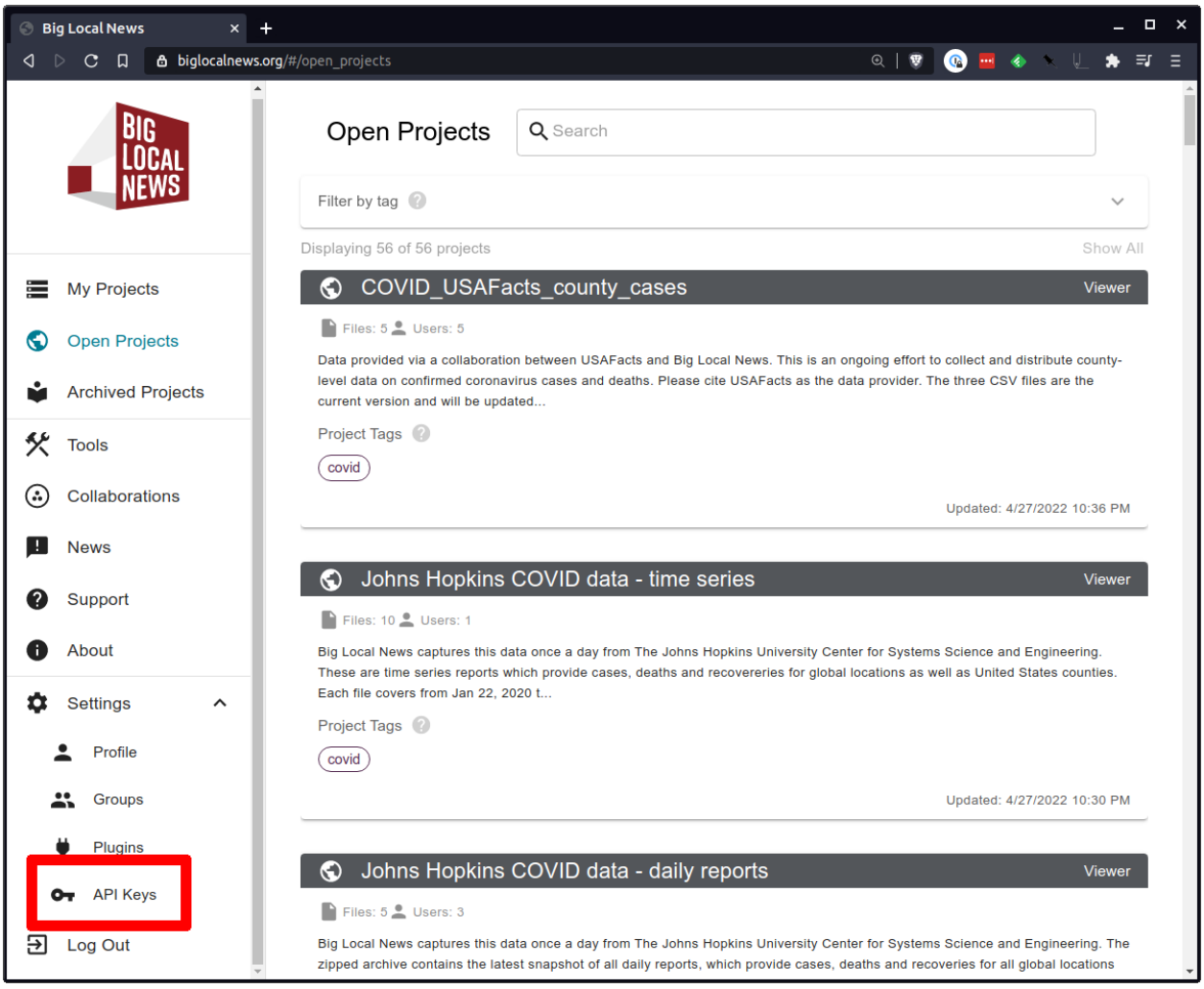
You can find the project id by visiting the project on biglocalnews.org and pulling the long hash from the URL. Here's the WARN Act Notices open project.



The name of the file can be found in the Files panel. Here's the Iowa data stored in `ia.csv`.



You can get an API key by visiting the link in the Settings menu.



If the token is set to the `BLN_API_TOKEN` environment variable, it doesn't have to be passed into the `read_bln` function.

Combine the entire example together and you can access the Iowa data file from the WARN Act Notices project with the following:

```
import pandas as pd
import bln

bln.pandas.register(pd)

project_id = "UHJvamVjdDpiZGM5NmU1MS1kMzBhLTRlYTctODY4Yi04ZGI4N2RjMzQ1ODI="
file_name = "ia.csv"
df = pd.read_bln(project_id, file_name)
```

Now you've got a dataframe to work with.

```
df.head()
```

```
                                Company          Address Line 1          City  ...
0                  SSP America, Inc.        5800 Fleur Drive      Des Moines  ...
1            Premier Linen & Drycleaning     461 West 9th Street        Dubuque  ...
```

(continues on next page)

(continued from previous page)

```
2                          Caterpillar, Inc.      1003 Miller Street          Elkader  ...
3  General Dynamics Information Technology  2400 Oakdale Boulevard       Coralville  ...
4                              Alorica, Inc.    2829 Westown Parkway  West Des Moines  ...
```

The `read_bln` function will also accept any of the standard configuration options offered by pandas reader functions, like `read_csv`. Here's an example using the `parse_dates` input.

```
df = pd.read_bln(project_id, file_name, parse_dates=["Notice Date"])
```

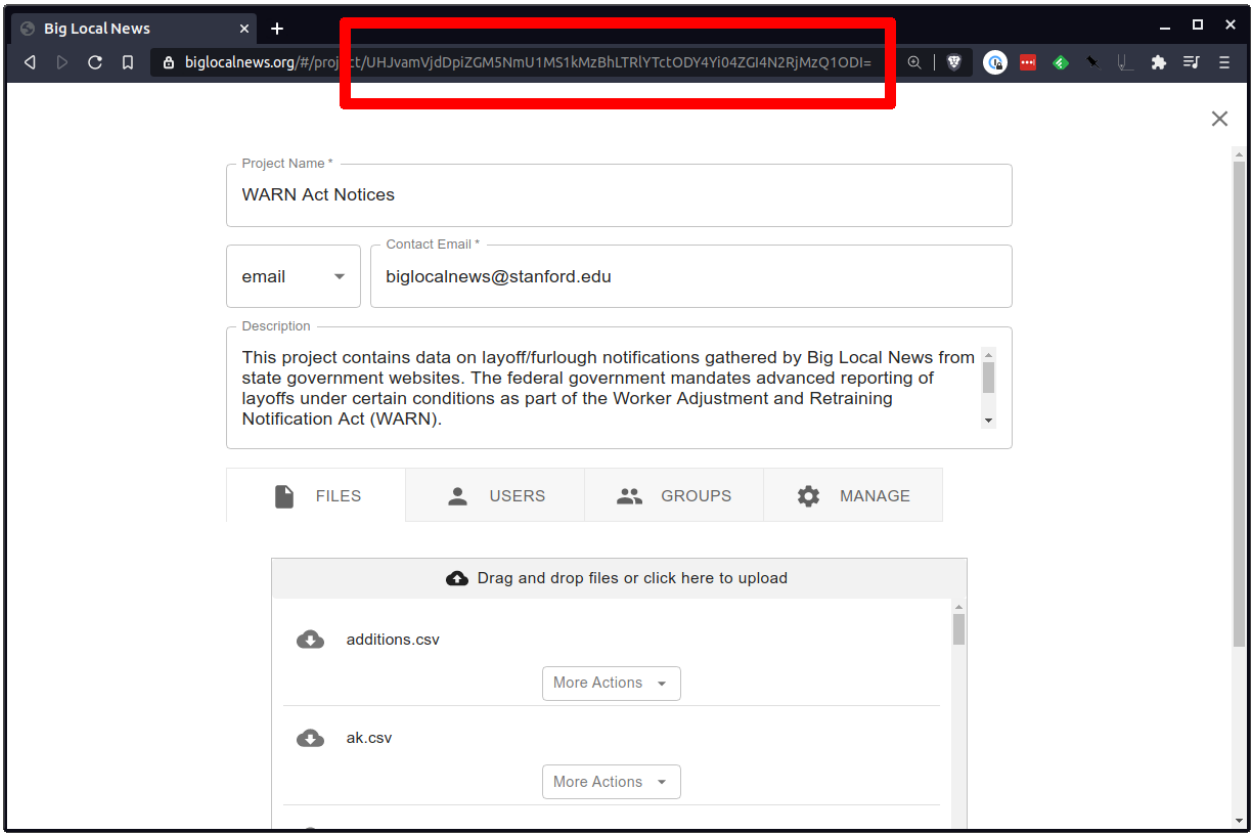
2.2.2 Writing data

You can write a file to biglocalnews.org using our custom `to_bln` dataframe accessor. Like the `read_bln` method, it requires three input:

1. The unique identifier of the biglocalnews.org project where the file will be stored
2. The name of the file to create within the biglocalnews.org project
3. An API key from biglocalnews.org with permission to read from the project.

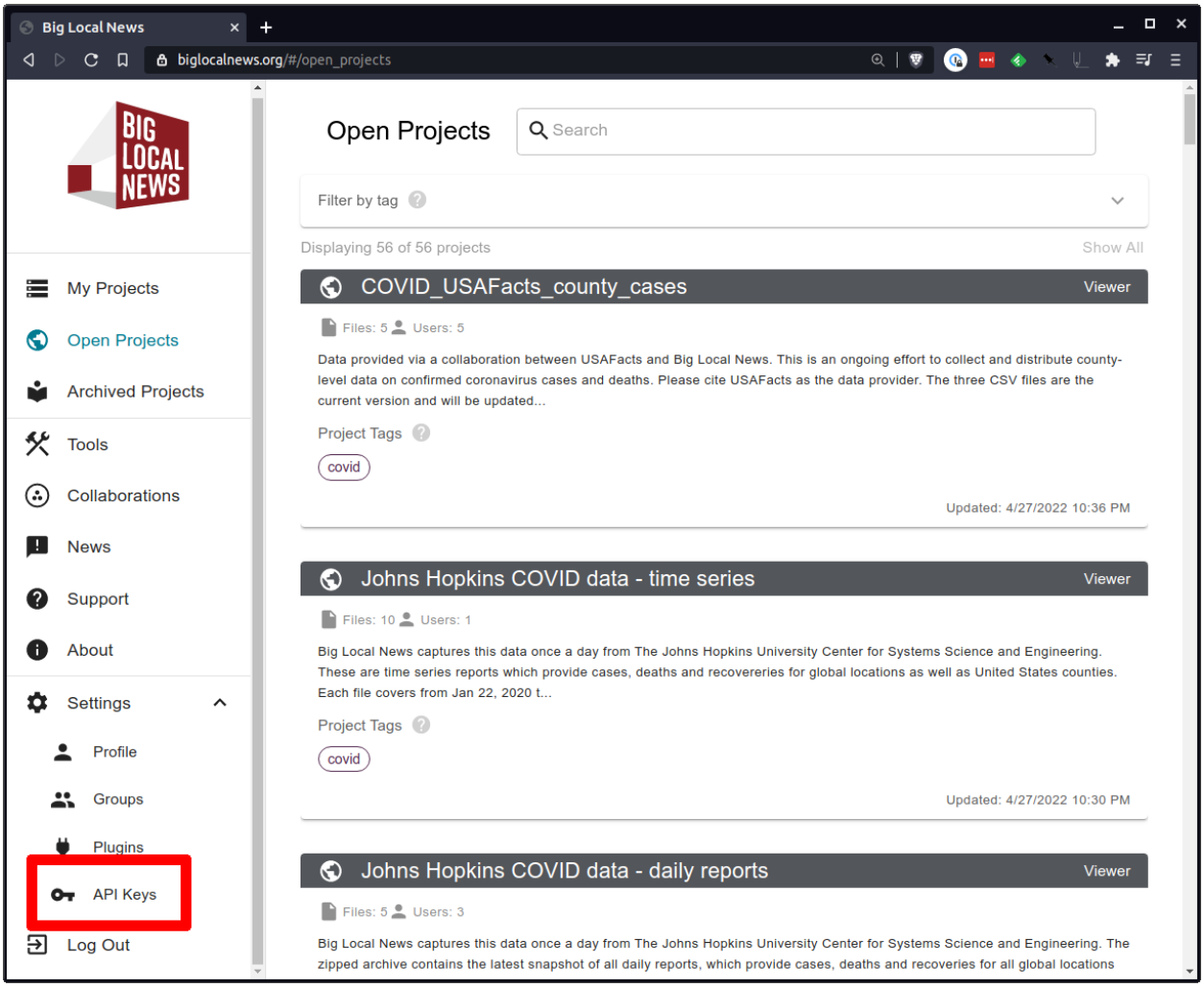
```
df.to_bln(your_project_id, your_file_name, your_api_token)
```

You can find the project id by visiting the project on biglocalnews.org and pulling the long hash from the URL. Here's the WARN Act Notices open project.



You can name the file whatever you like. The filenames must end with .csv, .json, .xls, or .xlsx. The extensions are mapped to the appropriate pandas writer function.

You can get an API key by visiting the link in the Settings menu.



If the token is set to the `BLN_API_TOKEN` environment variable, it doesn't have to be passed into the `read_bln` function.

Combine the entire example together and you can upload a data file from your computer to WARN Act Notices project with the following:

```
import pandas as pd
import bln

bln.pandas.register(pd)

df = pd.read_csv("my-local-file.csv")

project_id = "UHJvamVjdDpiZGM5NmU1MS1kMzBhLTRlYTctODY4Yi04ZGI4N2RjMzQ1ODI="
file_name = "my-uploaded-file.csv"
df.to_bln(project_id, file_name)
```

The `to_bln` method will also accept any of the standard configuration options offered by pandas writer functions, like `to_csv`. Here's an example using the `index` input.

```
df.to_bln(project_id, file_name, index=False)
```

2.3 Contributing

This documents guides you through how to make improvements to the `bln` library by installing its source code and suggesting improvements.

Sections

2.3.1 Create a fork

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

2.3.2 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 covered 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:your-username/bln-python-client.git
```

or

```
gh repo clone your-username/bln-python-client
```

2.3.3 Install dependencies

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

```
cd bln-python-client
```

The `pipenv` package manager can install all of the Python tools necessary to run and test our code. Since you'll be using the special tools we need as part of code development, add the `--dev` flag.

```
pipenv install --dev
```

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

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

2.3.4 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 fix an upload bug, you might name it something like this:

```
git checkout-b upload-bug-fix
```

2.3.5 Perform you work

Now you do your thing. Make the necessary changes to the code until you get the job done.

2.3.6 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.

2.3.7 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 origin your-branch-name
```

2.3.8 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 are linked in GitHub's system.

2.4 Releasing

How to release a new version of the bln module to the Python Package Index.

Sections

2.4.1 How it works

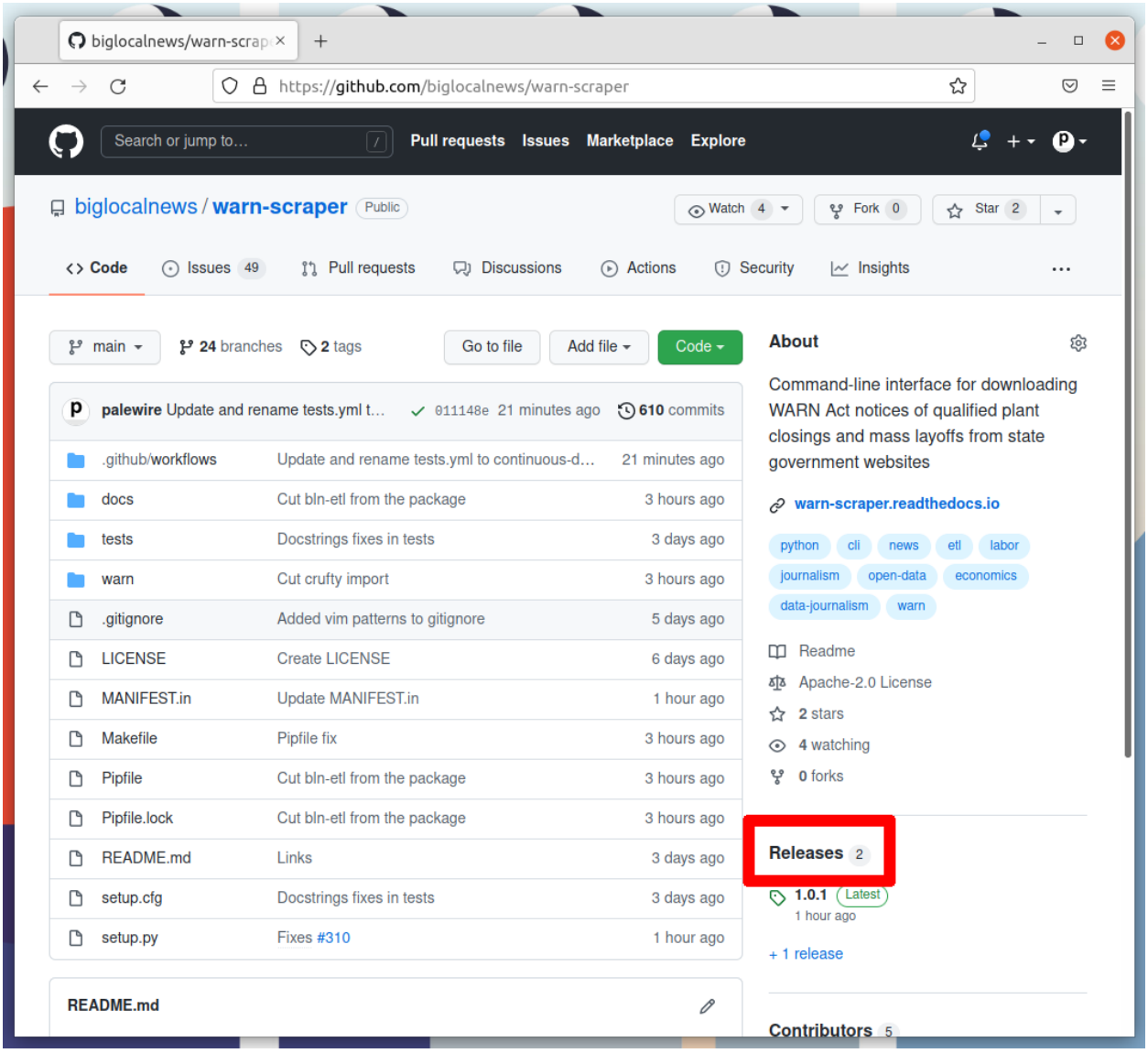
Our release process is automated as a continuous deployment via the GitHub Actions framework. The logic that governs the process is stored in the `workflows` directory.

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/bln-python-client/releases, then wait for the computers to handle the job.

Here's how it's done, step by step. The screenshots are from a different repository, but the process is the same.

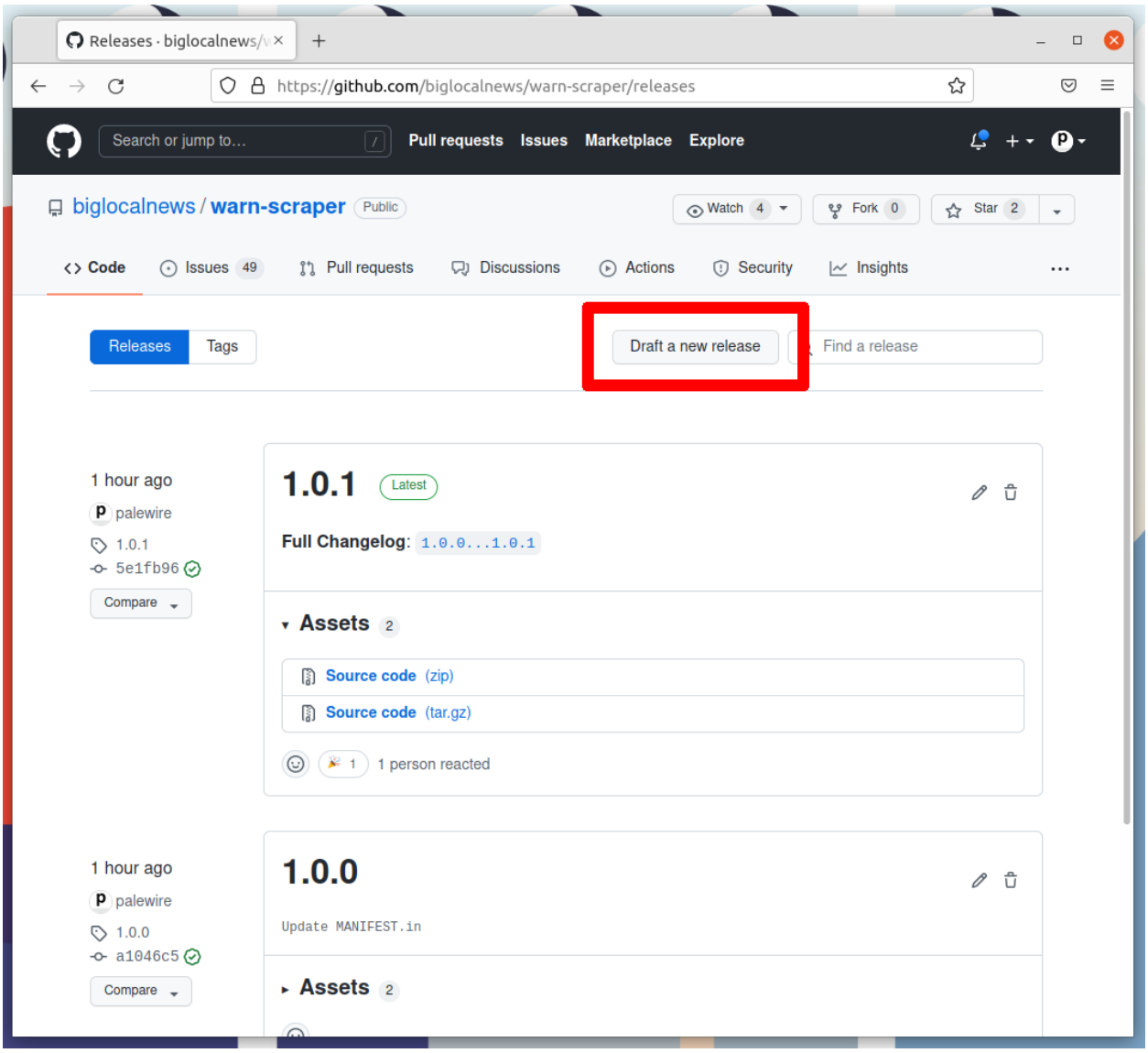
2.4.2 1. Go to the releases page

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



2.4.3 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.



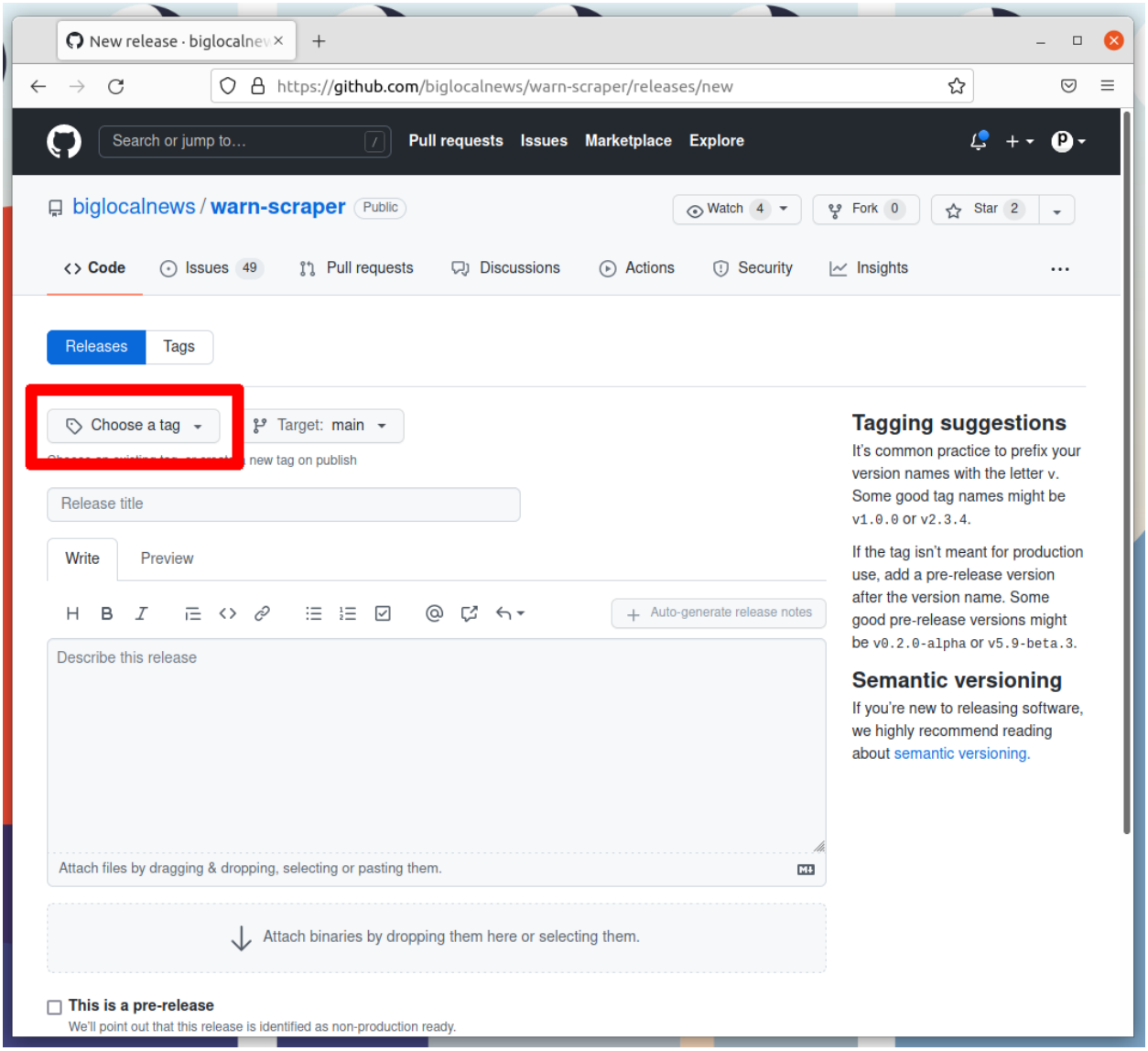
2.4.4 3. Create a new tag

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

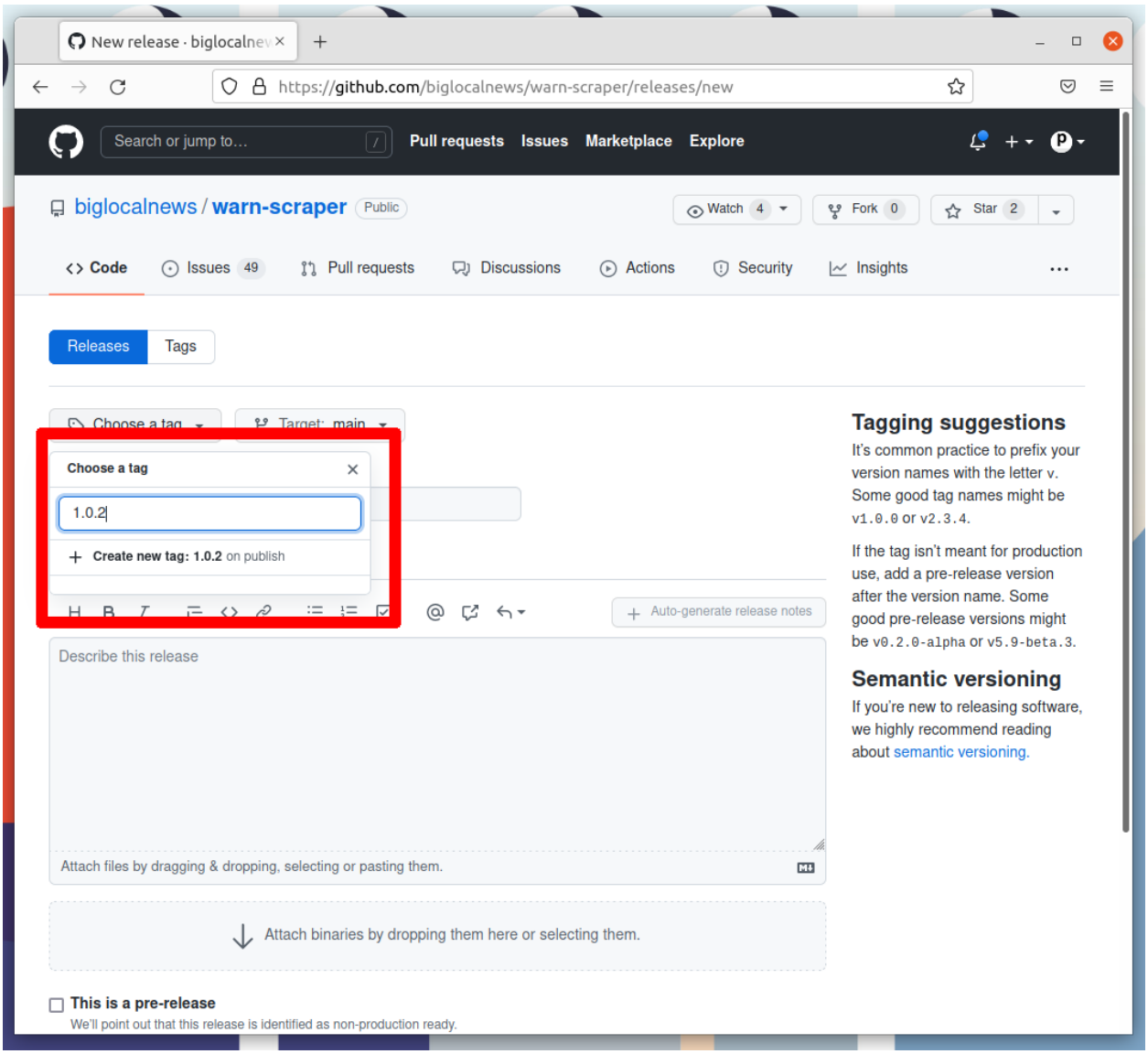
All version numbers should feature three numbers separated by the periods, like `1.0.1`. If you're making a major release that isn't backwards compatible, the latest release's first number should go up by one. If you're making a minor release by adding a feature or major a large change, the second number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

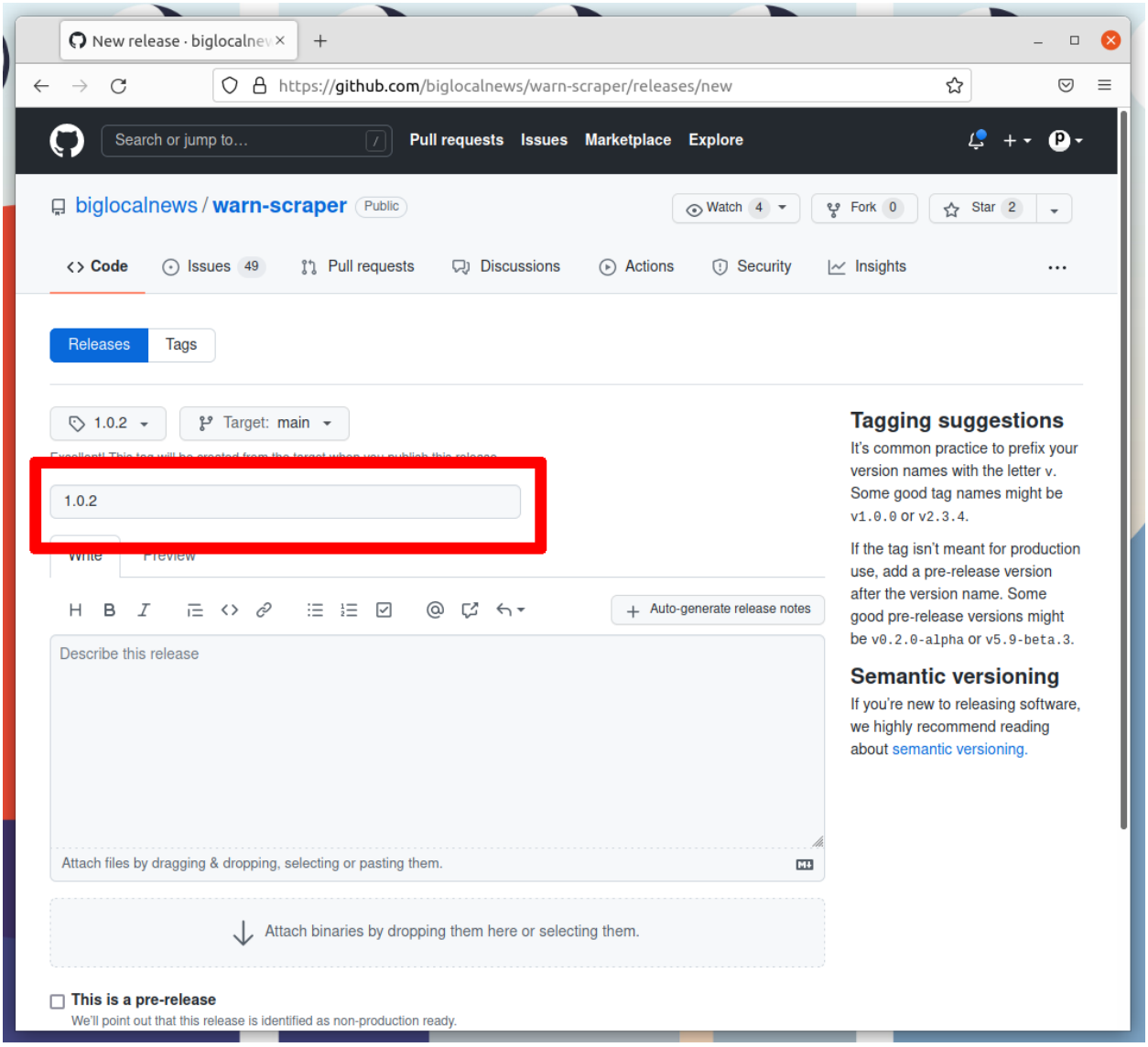


Enter your version number into the box. Then click the “Create new tag” option that appears.



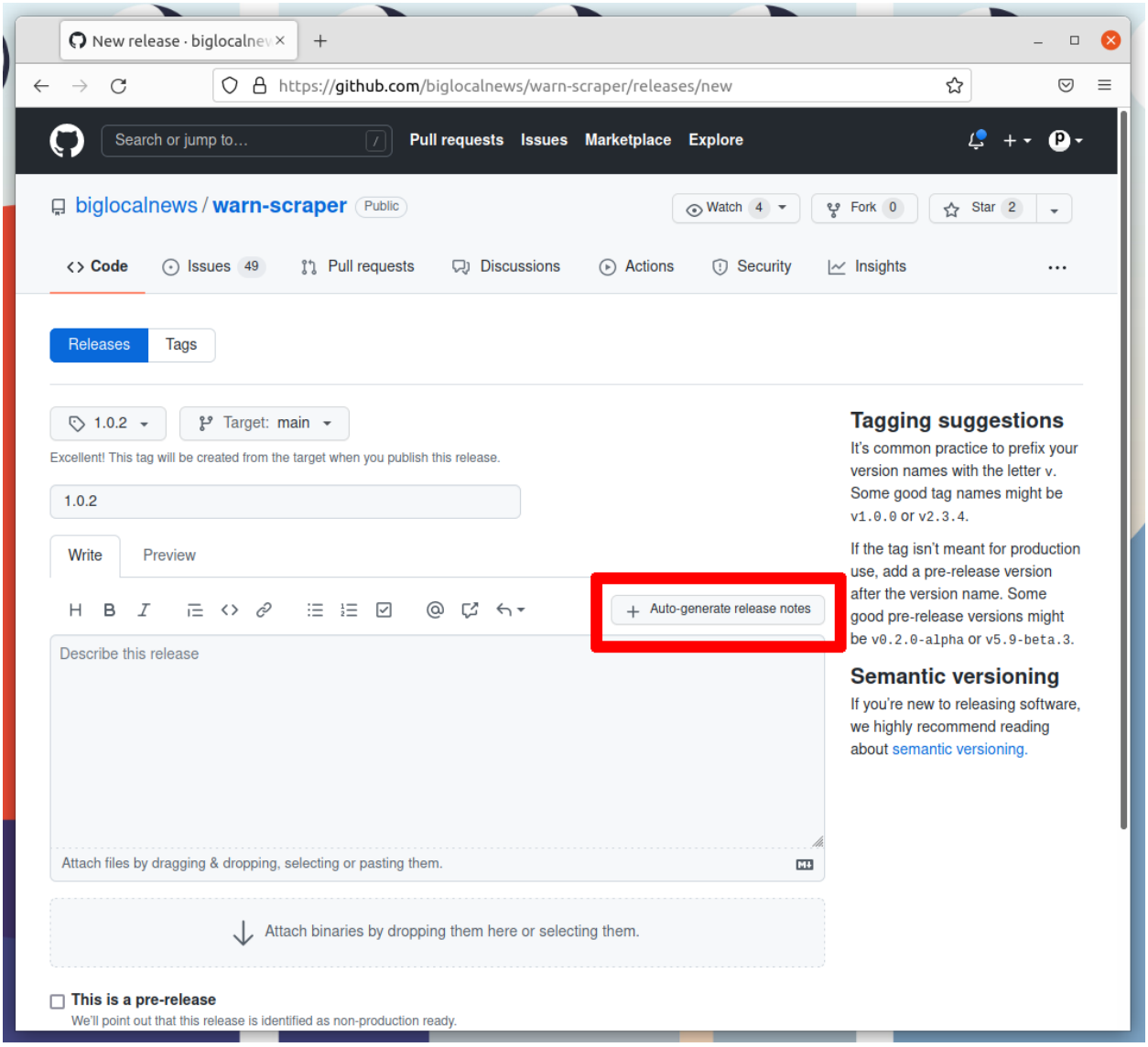
2.4.5 4. Name the release

Enter the same number into the “Release title” box.

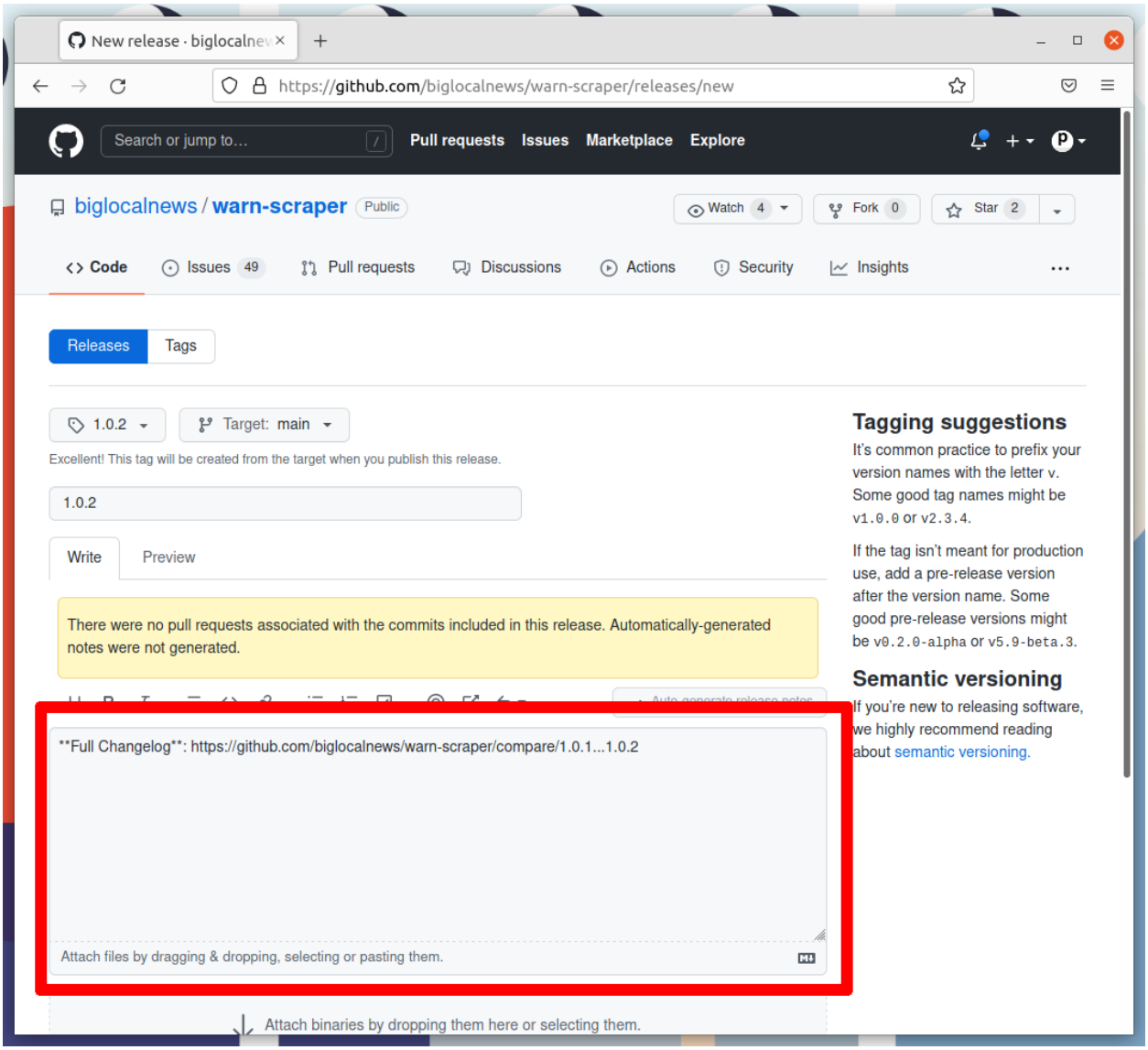


2.4.6 5. Auto-generate release notes

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

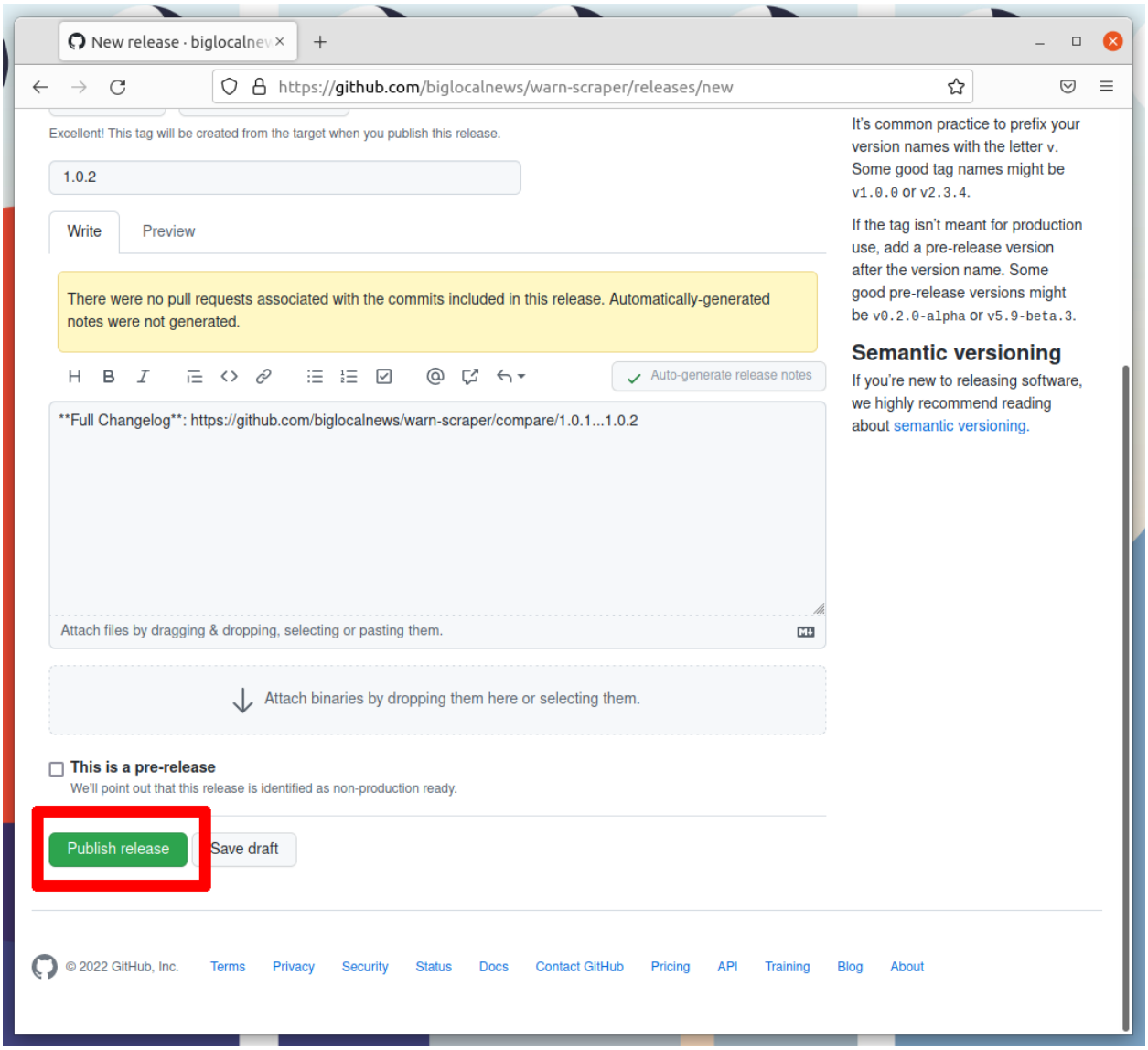


That should fill in the box below. What appears will depend on how many pull requests you've merged since the last release.



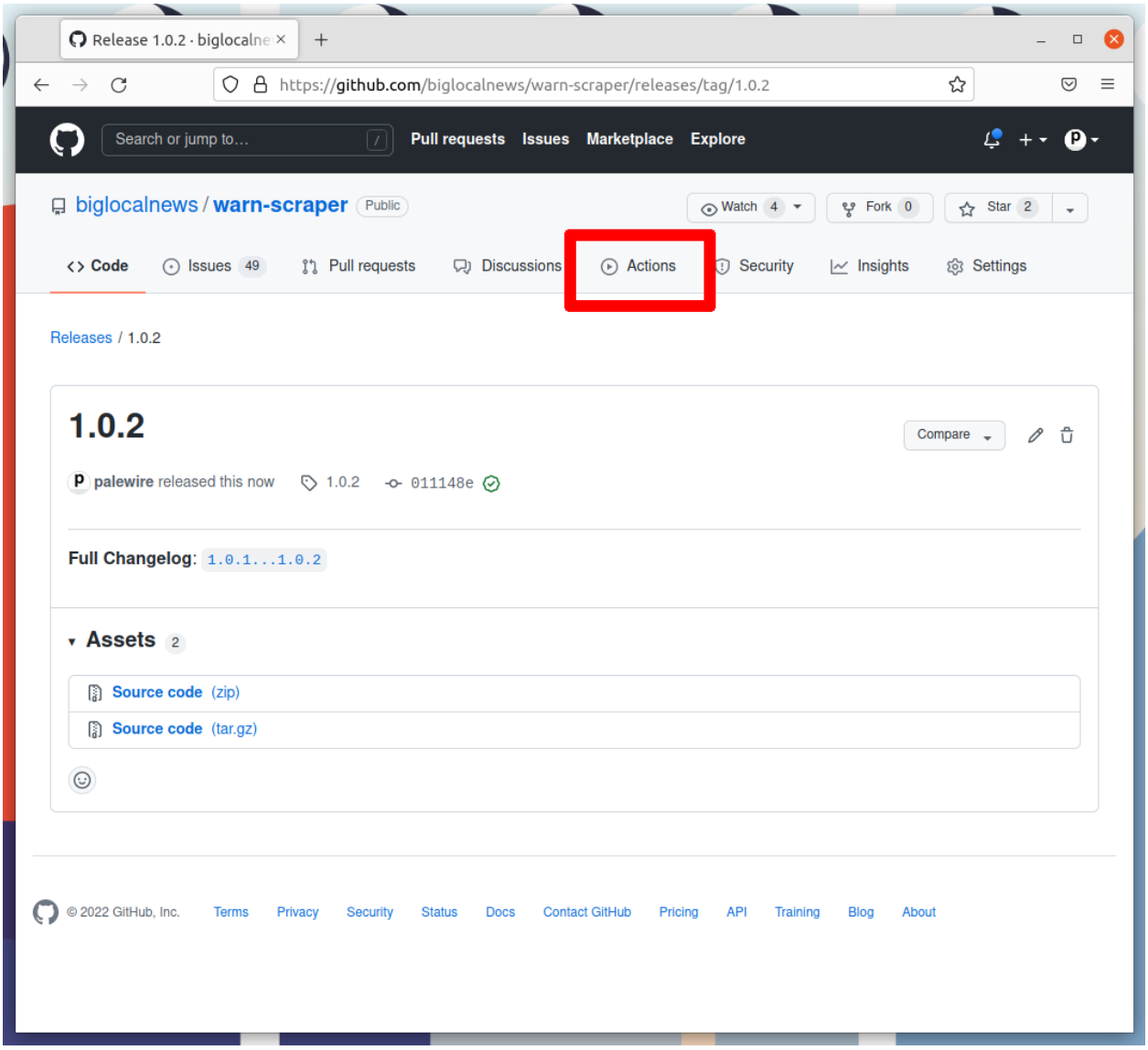
2.4.7 6. Publish the release

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

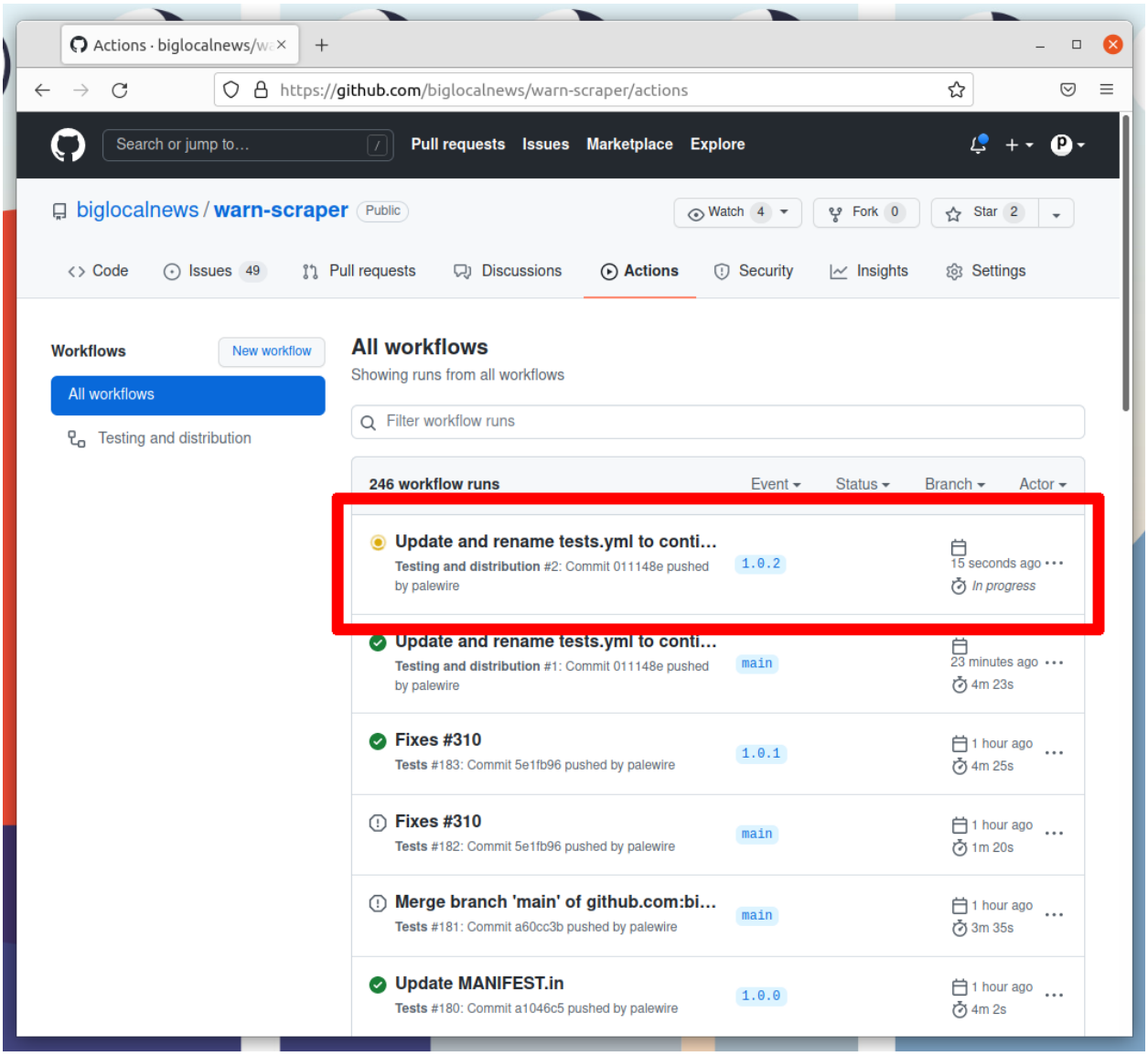


2.4.8 7. Wait for the Action to finish

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.

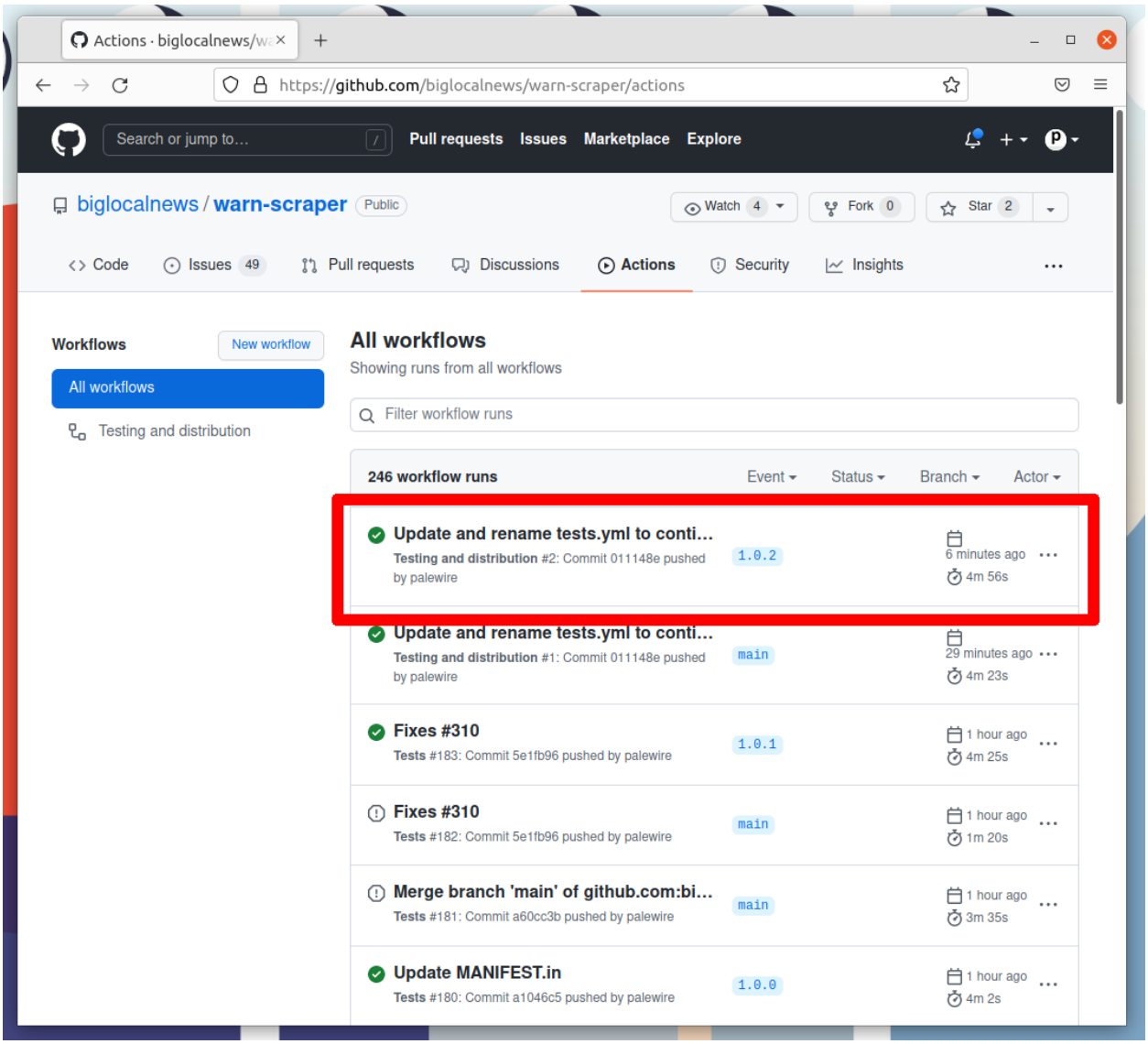


That will take you to the Actions monitoring page. The task charged with publishing your release should be at the top.

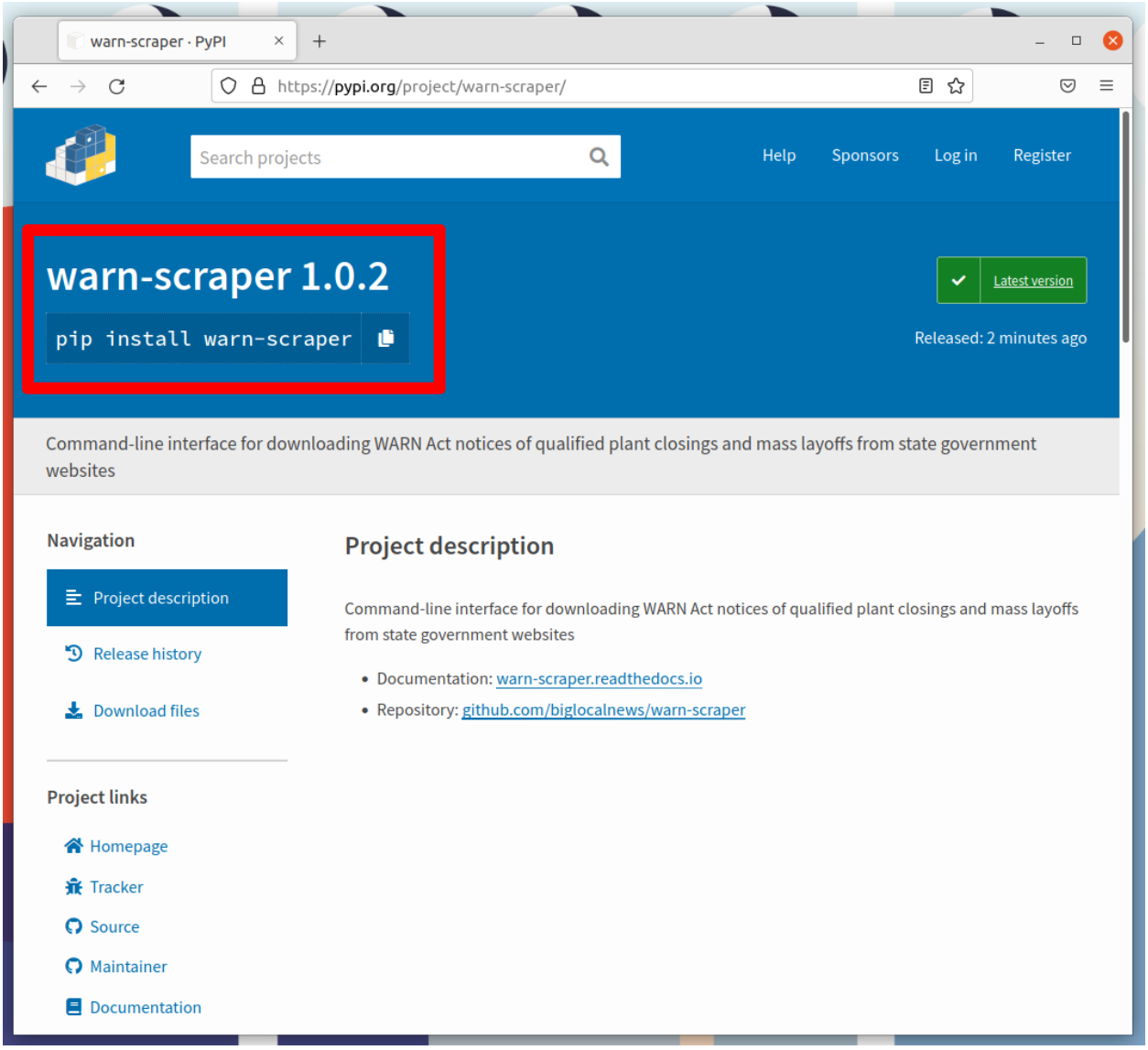


2.4.9 8. Check the results on PyPI

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



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



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.

2.5 Reference

Sections

- *Python client*
- *pandas extensions*

2.5.1 Python client

2.5.2 pandas extensions

read_bln

to_bln

CHAPTER

THREE

LINKS

- Documentation: bln-python-client.readthedocs.io
- Repository: github.com/biglocalnews/bln-python-client
- Packaging: pypi.org/project/bln

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.