Bioinformatics session

A two-day workshop for bioinformaticians and molecular biologists with focus on the TSO500 pipeline in InPreD

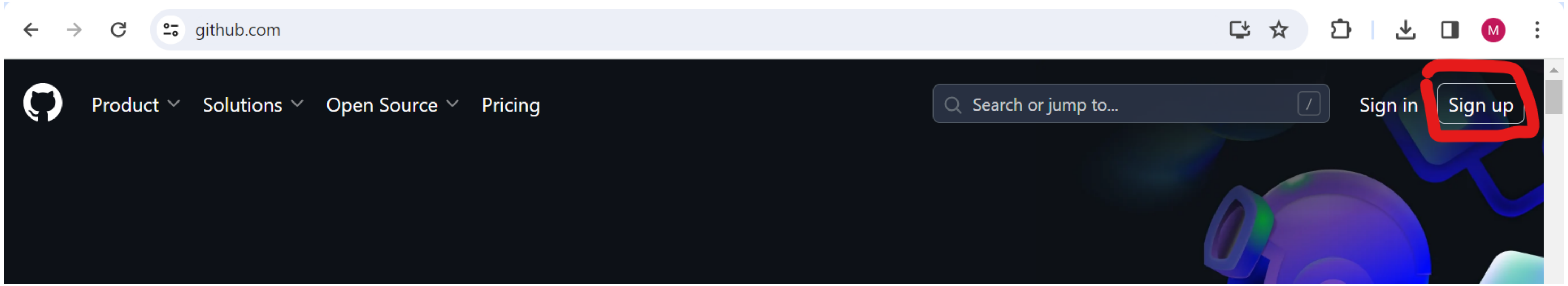
Overview

1. Setup
2. Development & Collaboration
3. Nextflow
4. tso500_nxf_workflow
5. Python

1. Setup

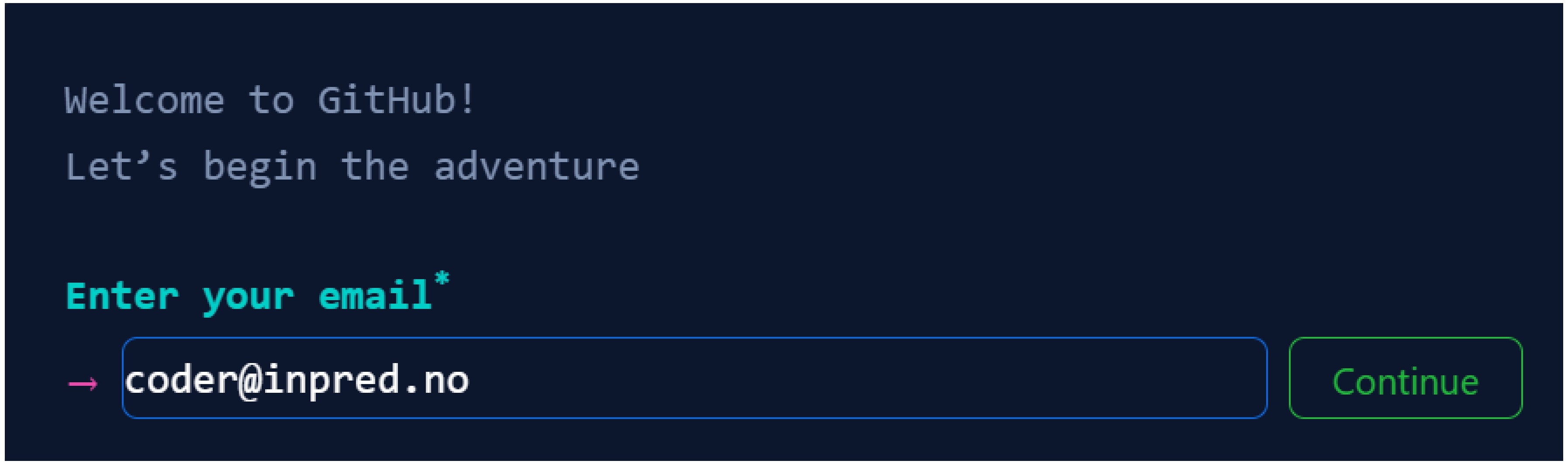
Create a GitHub account

- go to https://github.com/ and click on `Sign up`



Create a GitHub account

- enter your email



Welcome to GitHub!

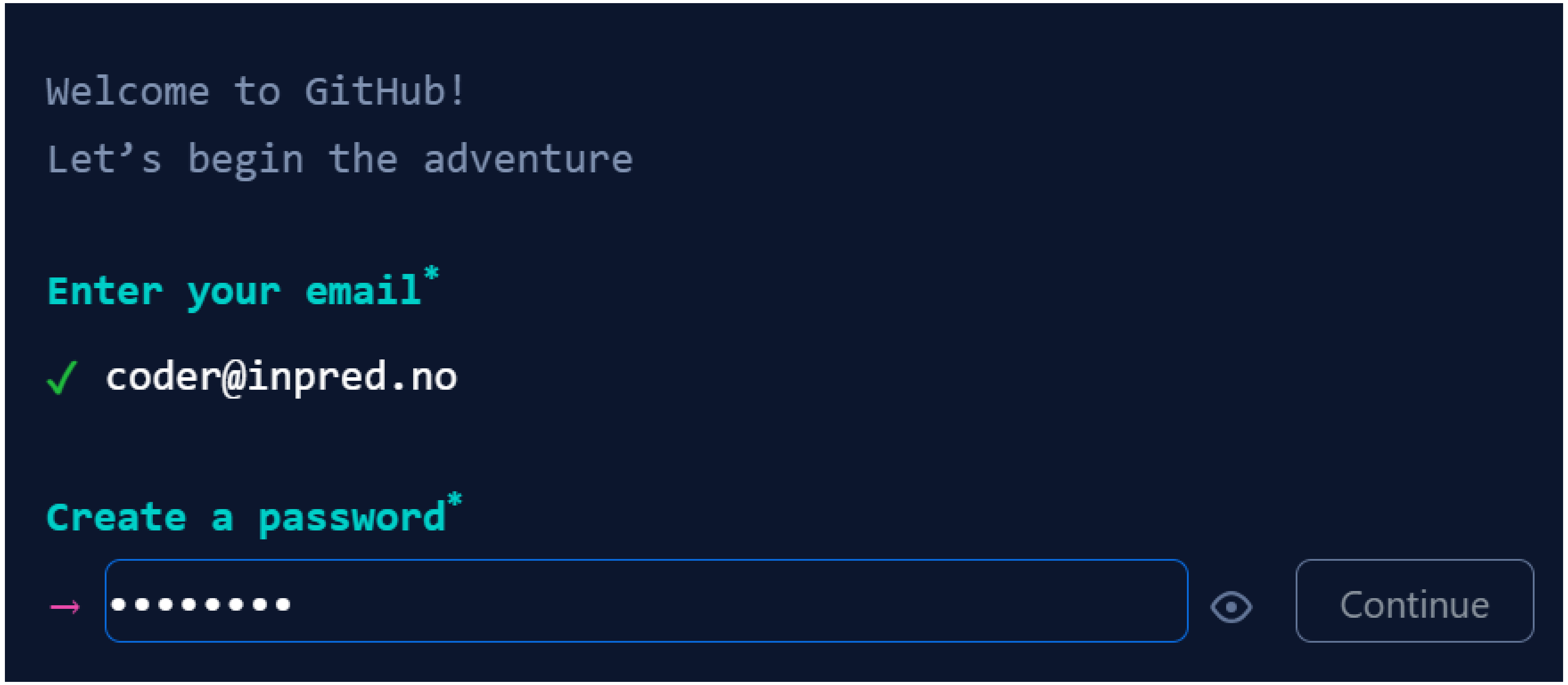
Let’s begin the adventure

Enter your email*

→ coder@inpred.no

Create a GitHub account

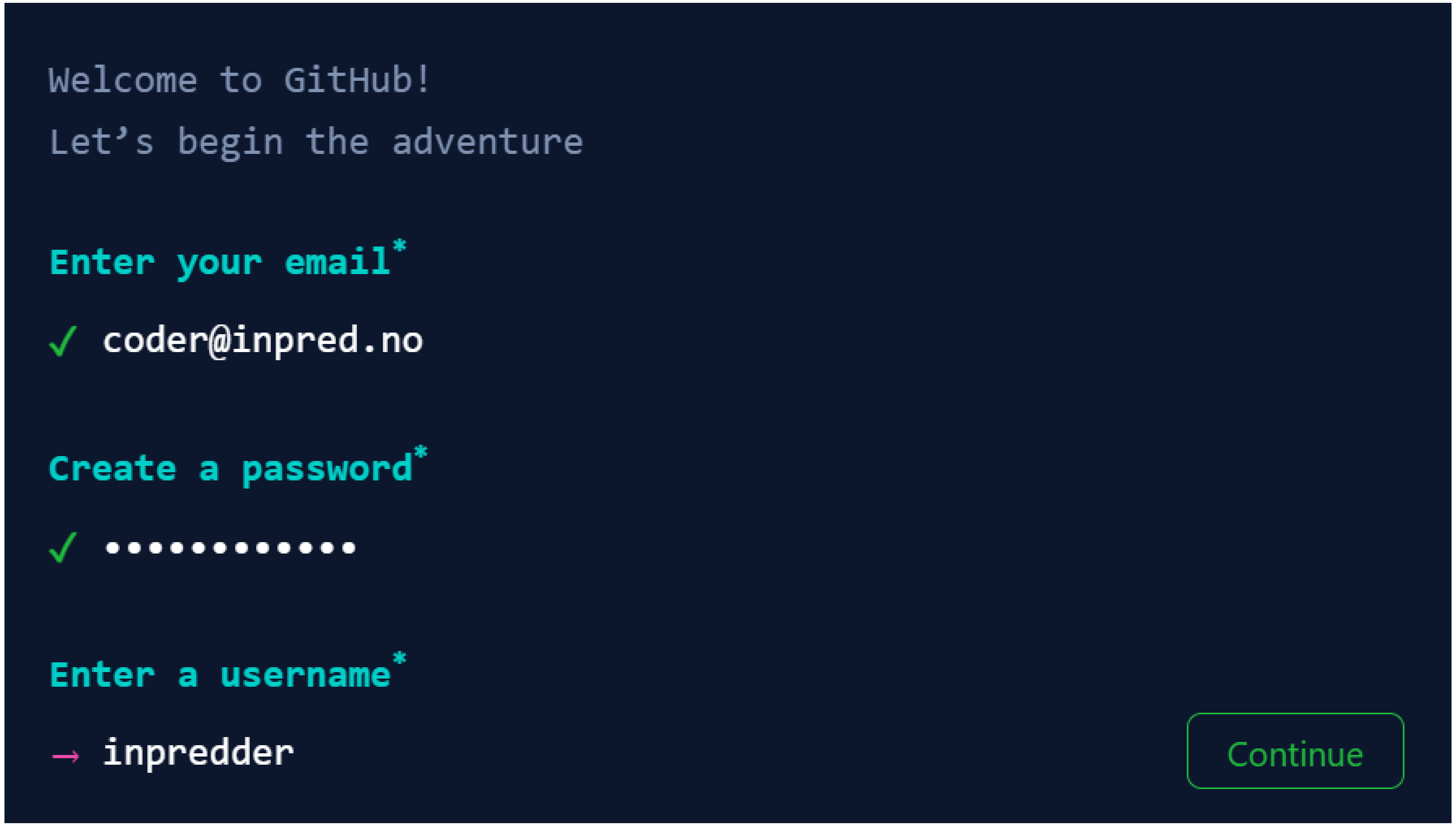
- set a password



```
Welcome to GitHub!
Let's begin the adventure

Enter your email*
✓ coder@inpred.no

Create a password*
→ ••••••••
```

Create a GitHub account

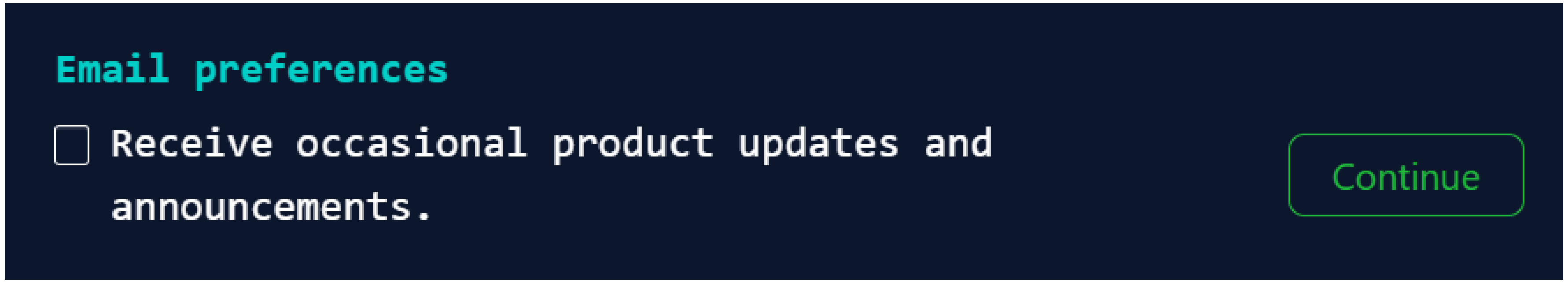
- choose a username



```
Welcome to GitHub!
Let’s begin the adventure

Enter your email*
✓ coder@inpred.no

Create a password*
✓ •••••••••••••

Enter a username*
→ inpredder
```

Create a GitHub account

- choose email preferences

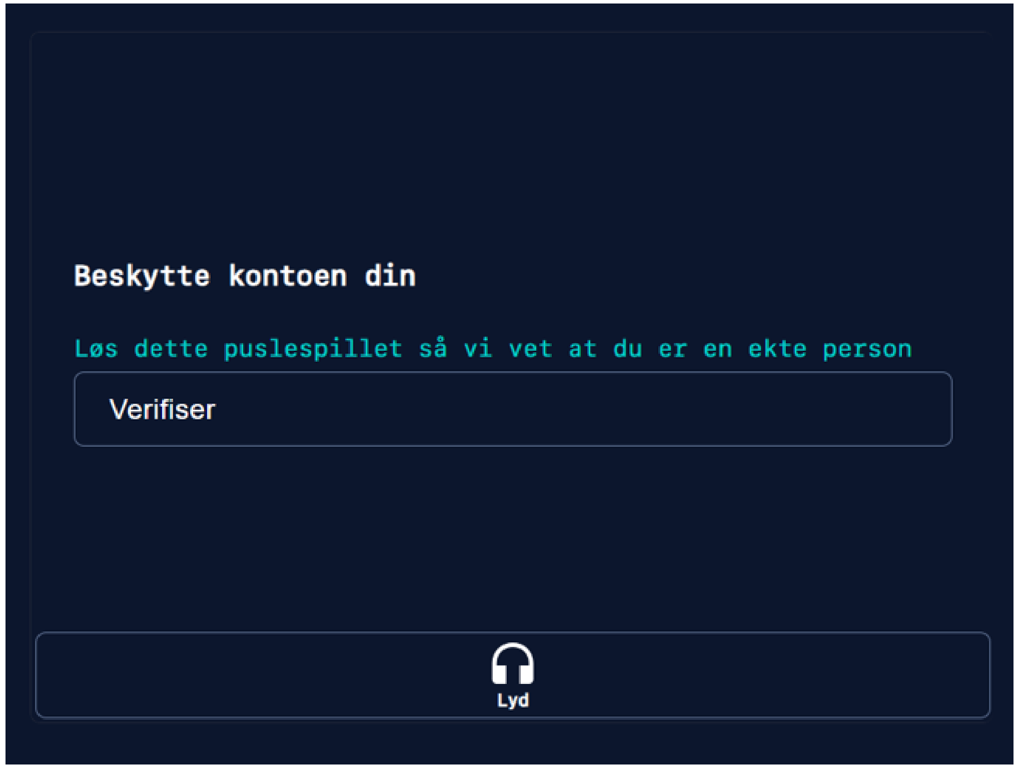


Email preferences

☐ Receive occasional product updates and announcements.

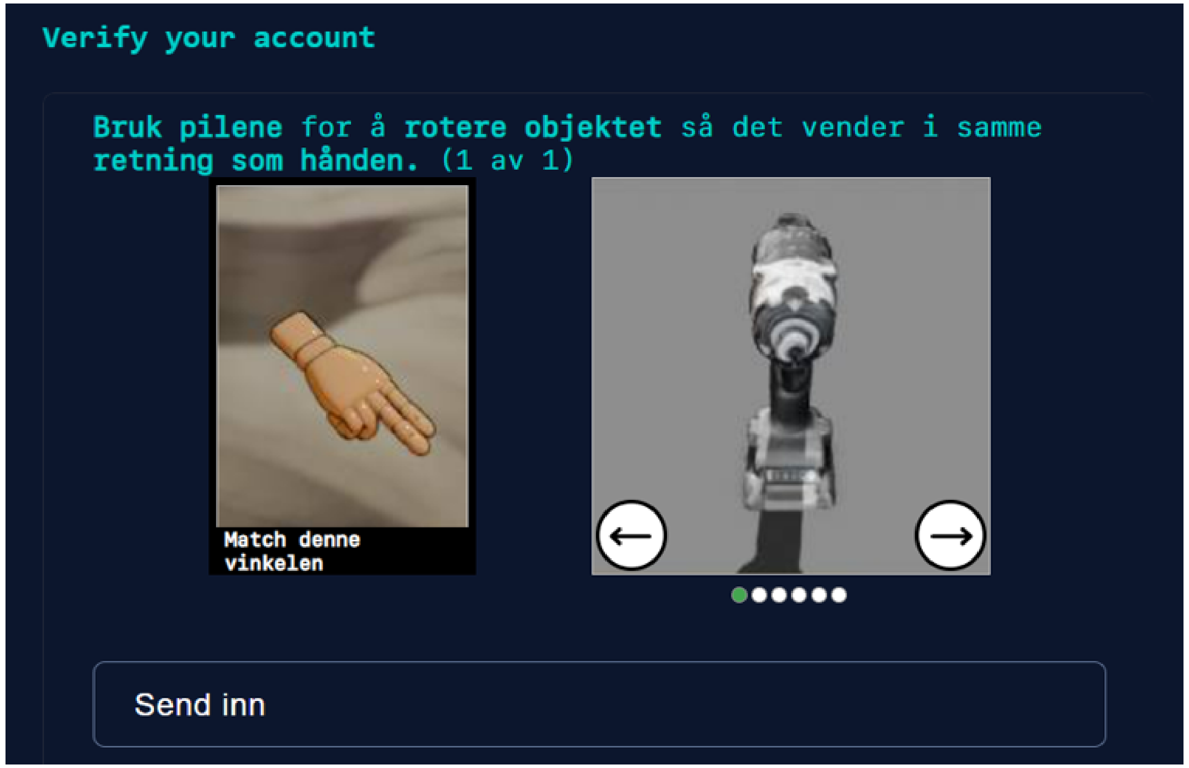
Create a GitHub account

- solve the puzzle



Beskytte kontoen din

Løs dette puslespillet så vi vet at du er en ekte person

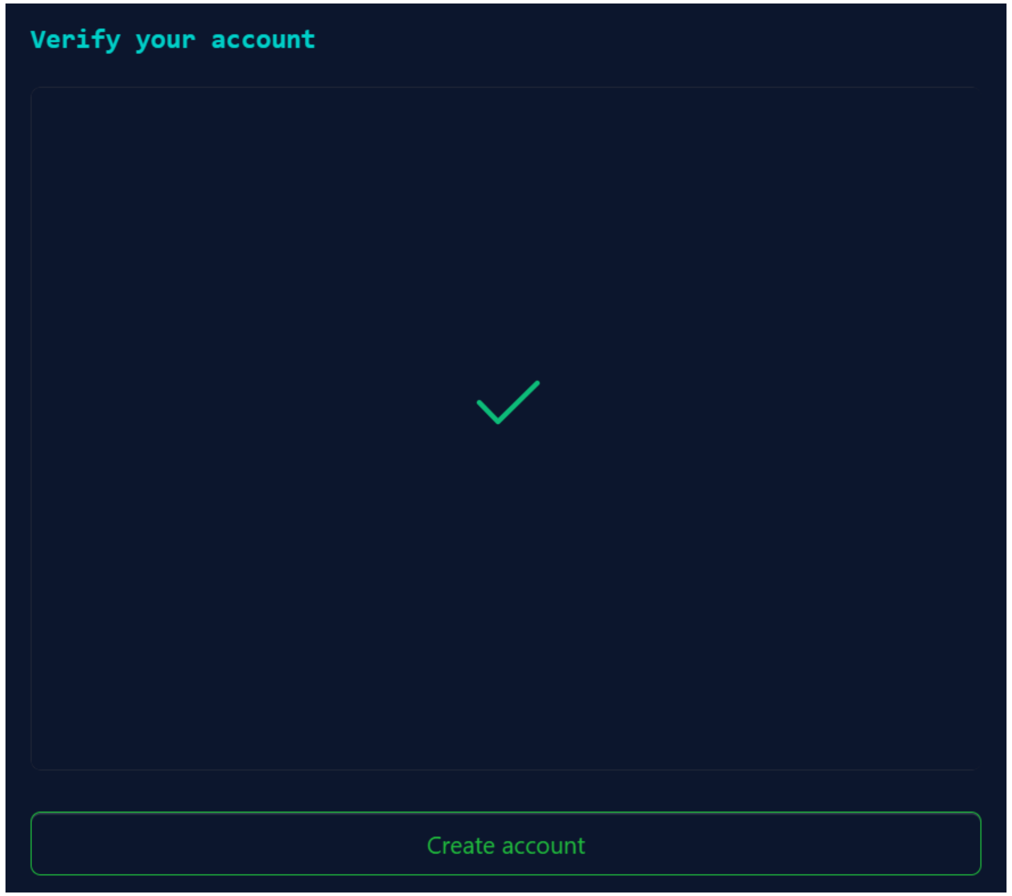


Verify your account

Bruk pilene for å rotere objektet så det vender i samme retning som hånden. (1 av 1)

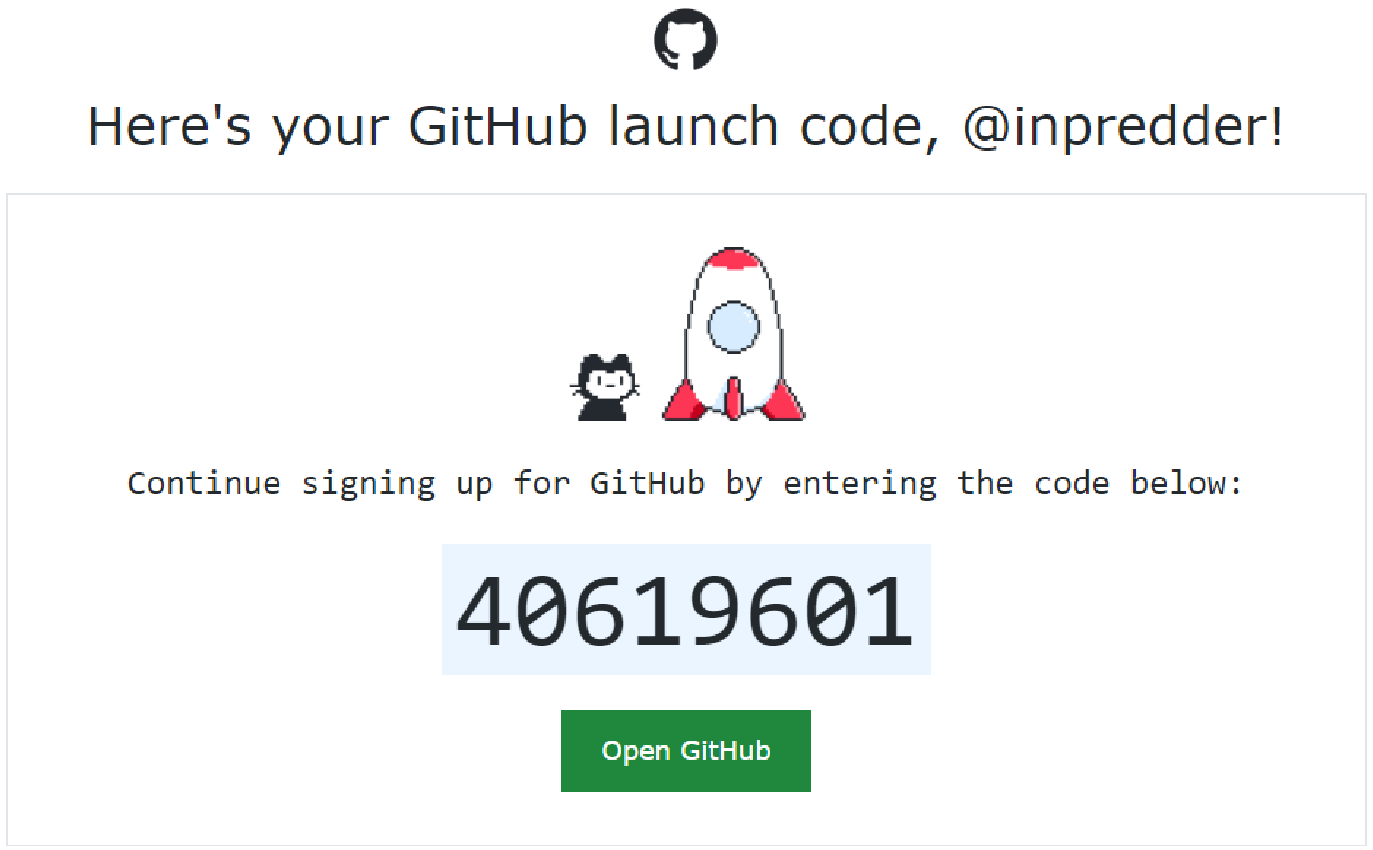
Create a GitHub account

- create your account



Create a GitHub account

- find the activation code in the email you received



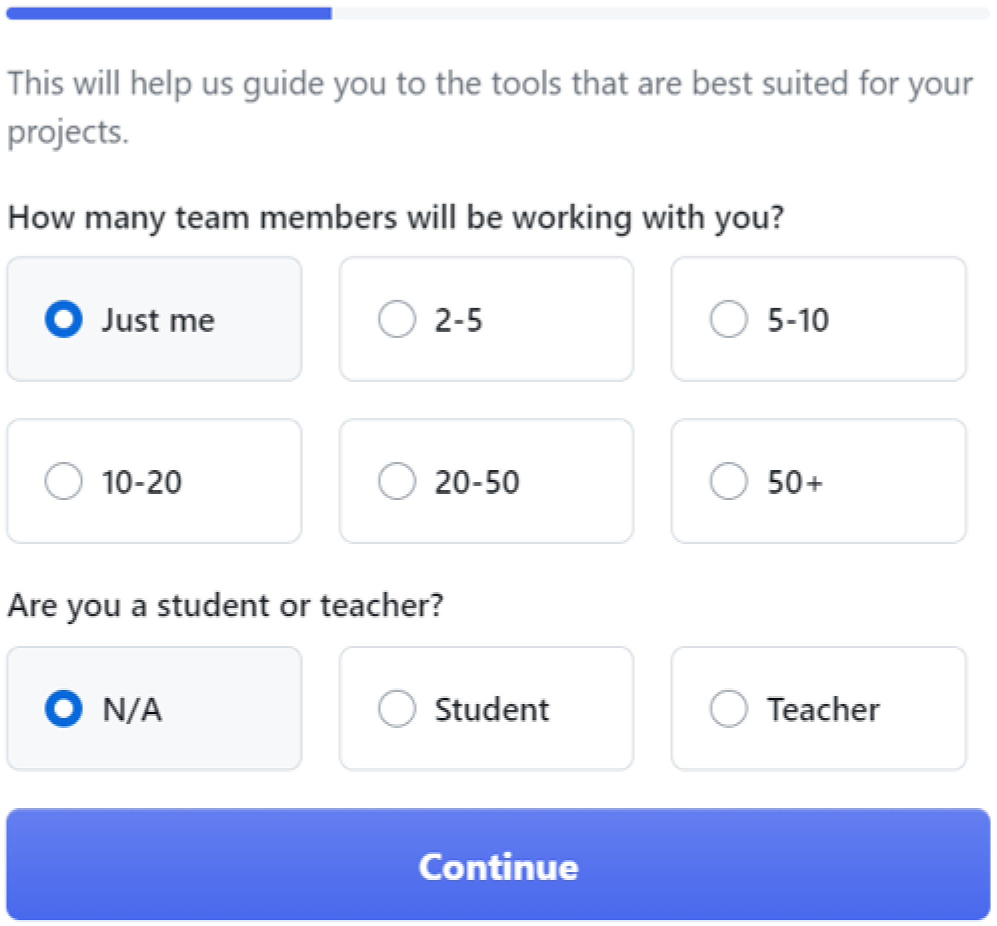
Here's your GitHub launch code, @inpredder!

Continue signing up for GitHub by entering the code below:

40619601

Create a GitHub account

- select the desired options



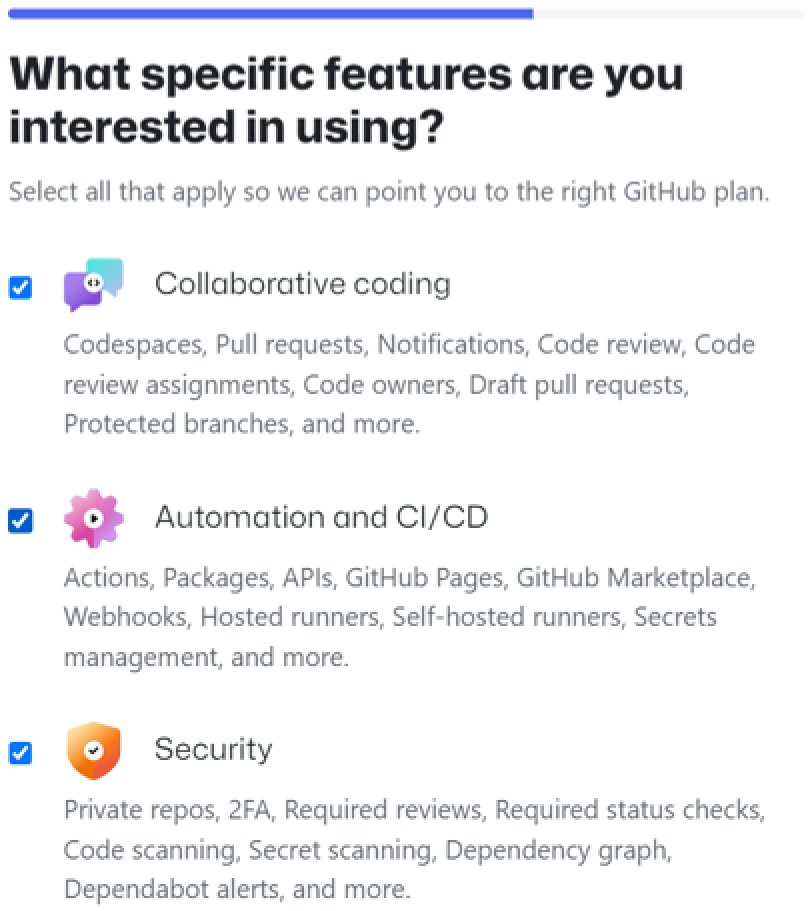
This will help us guide you to the tools that are best suited for your projects.

How many team members will be working with you?

[✓ Just me] 2-5 5-10
10-20 20-50 50+

Are you a student or teacher?

[✓ N/A] Student Teacher



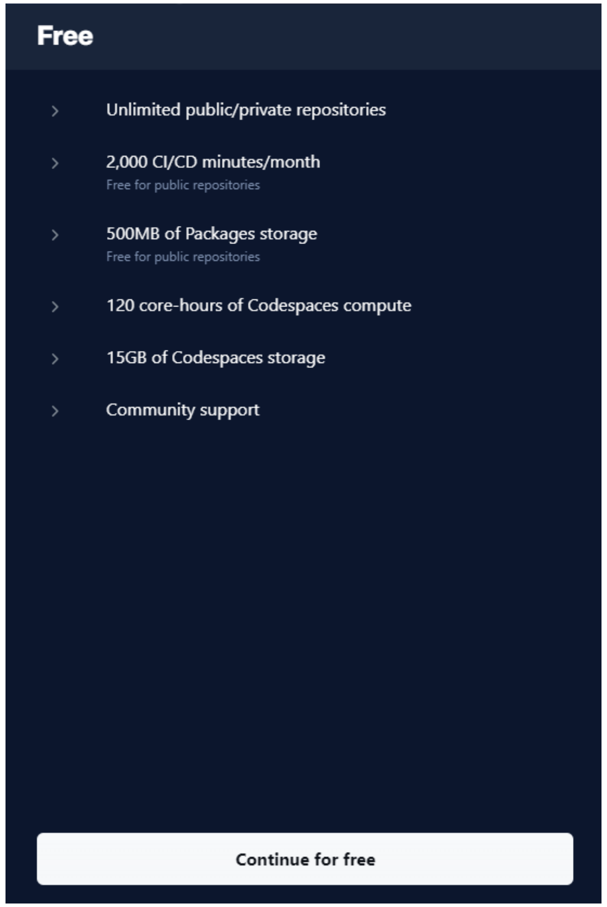
What specific features are you interested in using?

Select all that apply so we can point you to the right GitHub plan.

- [✓] Collaborative coding
  Codespaces, Pull requests, Notifications, Code review, Code review assignments, Code owners, Draft pull requests, Protected branches, and more.
- [✓] Automation and CI/CD
  Actions, Packages, APIs, GitHub Pages, GitHub Marketplace, Webhooks, Hosted runners, Self-hosted runners, Secrets management, and more.
- [✓] Security
  Private repos, 2FA, Required reviews, Required status checks, Code scanning, Secret scanning, Dependency graph, Dependabot alerts, and more.

Create a GitHub account

- choose the free plan



Free

- Unlimited public/private repositories
- 2,000 CI/CD minutes/month
  Free for public repositories
- 500MB of Packages storage
  Free for public repositories
- 120 core-hours of Codespaces compute
- 15GB of Codespaces storage
- Community support

Continue for free

1. Setup

Be added to InPreD organisation at GitHub

1. Resources

- Getting started with your GitHub account

2. Development & Collaboration

Short `git` introduction

- distributed version control system
- tracks history of changes commited by different contributors
- every developer has full copy of project and its history

Short `git` introduction

Basic `git` commands

`git init`: initialises new git repository

`git clone <repository url>`: creates local copy of remote repository

`git add <file/s>`: stage new or changed files (anything that should be committed to the repository)

`git commit -m "feat: my new feature"`: commit changes to the repository

Basic `git` commands

commit message conventions

```
<type>[optional scope]: <description>
```

- `feat`: new feature
- `fix`: patching bug
- `refactor`: code change that neither is neither feat nor fix
- `build`: build system related changes
- `perf`: improving performance

commit message conventions

`<type>[optional scope]: <description>`

- `chore`: code unrelated changes, e.g. dependencies
- `style`: code change that does not change meaning
- `test`: changes to tests
- `docs`: adding/updating documentation
- `ci`: continuous integration, e.g. github actions

Basic `git` commands

`git status`: overview over untracked, modified and staged changes

`git branch`: show local branches

`git merge`: merge branches

`git pull`: load changes from remote counterpart

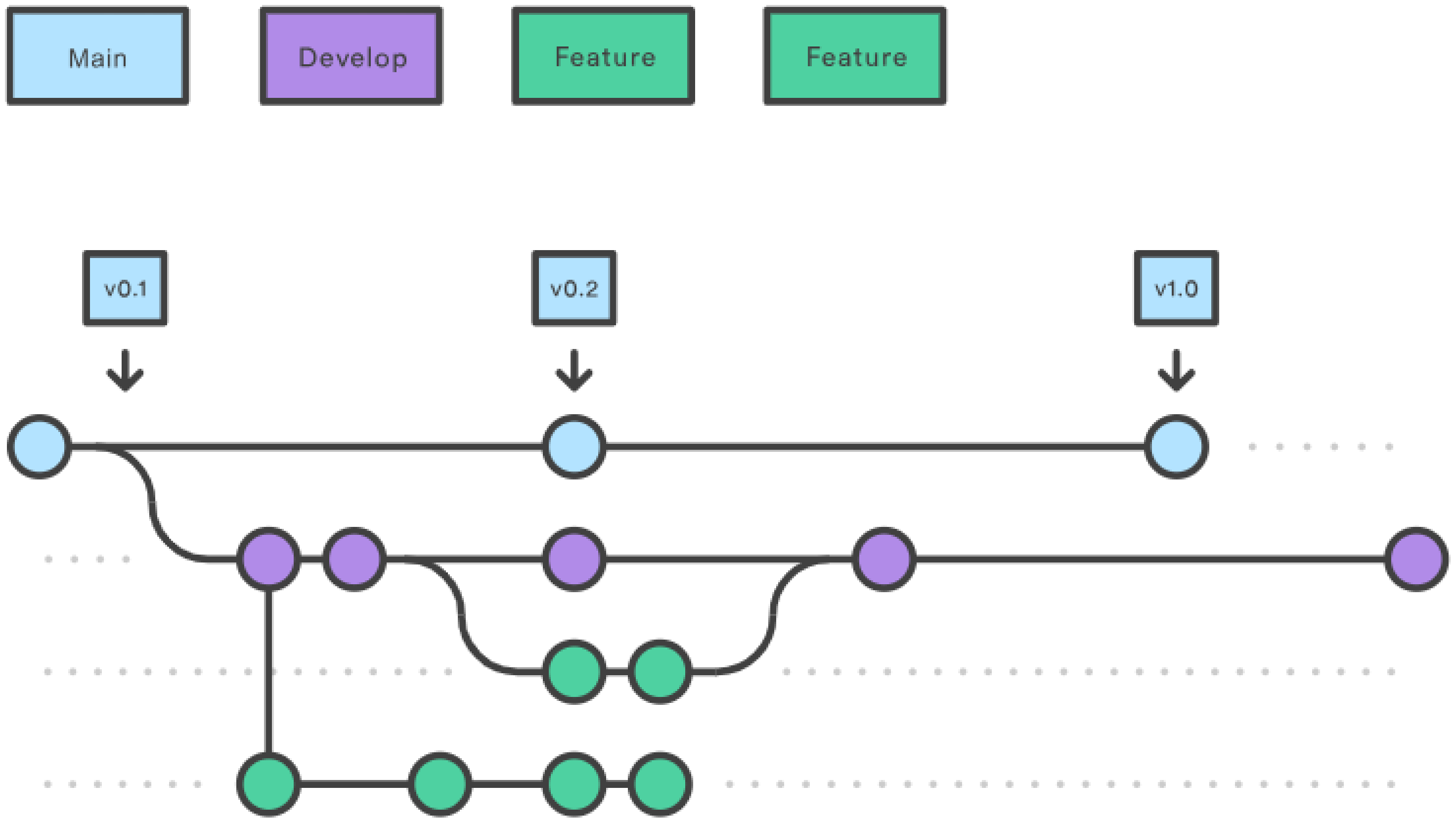
`git push`: upload changes to remote counterpart

2. Development & Collaboration

Branching model: simplied Gitflow workflow

- start with two branches to record project history: `main` and `develop`
- each new feature resides in its own branch (feature branch)
- feature branch is generally created off latest `develop` commit
- upon feature completion, feature branch is merged into `develop`

Branching model: simplied Gitflow workflow



2. Development & Collaboration

GitHub Actions

- continuous integration (CI) and continuous deployment (CD)
- building, testing and deploying directly from GitHub
- set up by adding yaml instructions to `.github/workflows`

```yaml
name: GitHub Actions Demo
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "Hello world!"
```

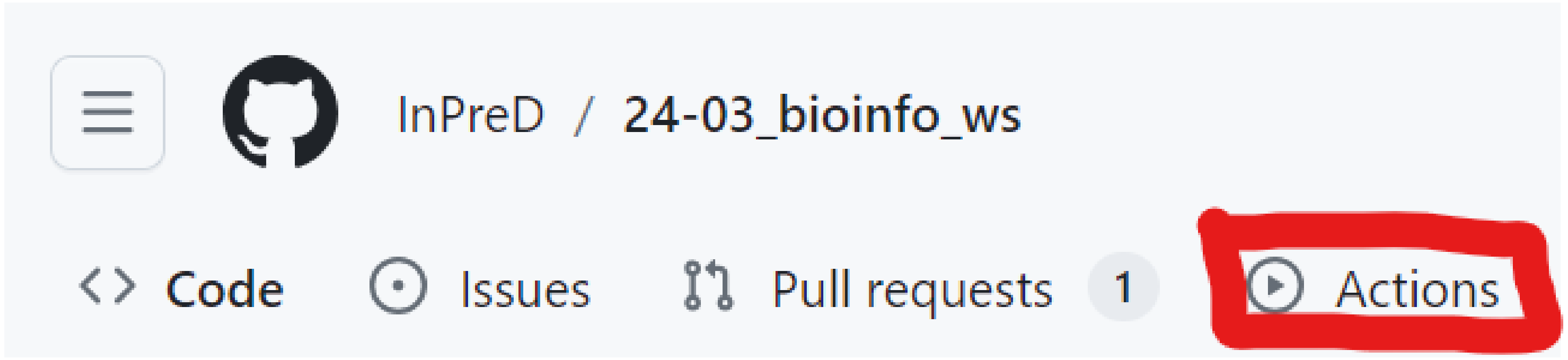
GitHub Actions

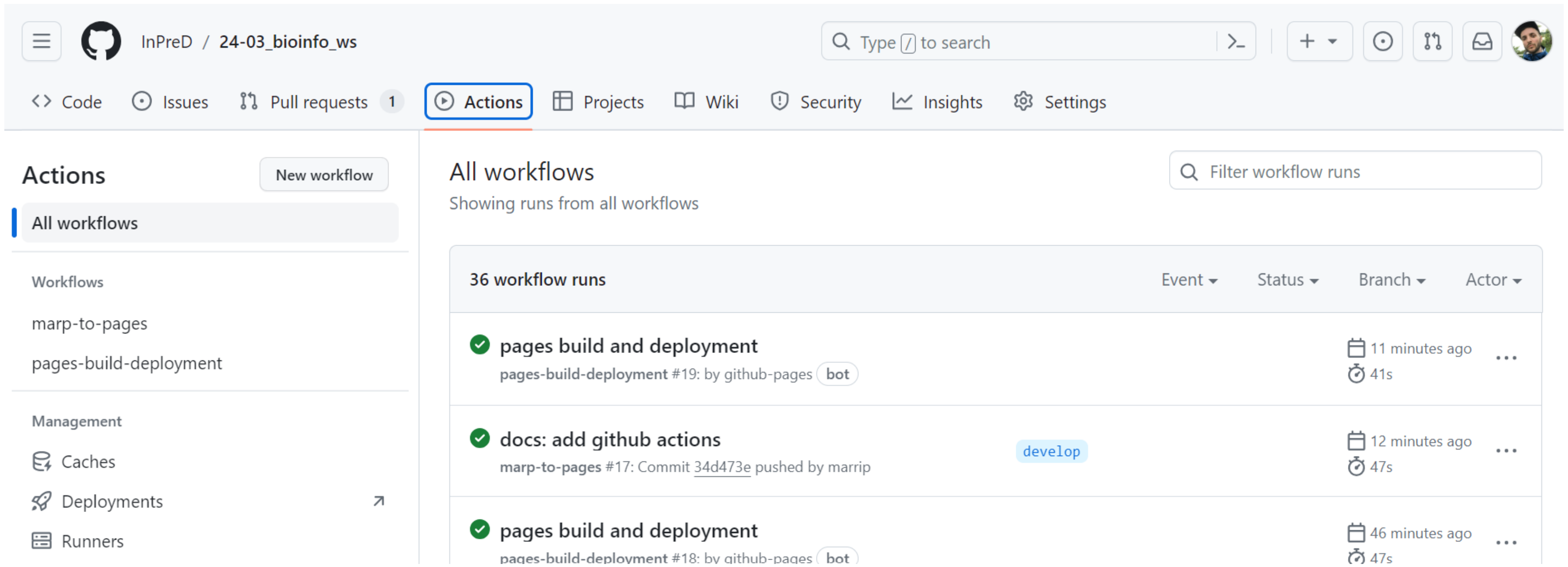
```yaml
name: Docker Build
on:
  push:
    branches:
      - main
      - develop
    tags:
      - '*.*.*'

jobs:
  test:
    name: Run unit tests
    runs-on: ubuntu-latest
    steps:
      -
        name: Check out the repo
        uses: actions/checkout@v4
      -
        name: Unit testing
        uses: fylein/python-pytest-github-action@v2
        with:
          args: pip3 install -r requirements.txt && pytest
  ...
```

GitHub Actions

```yaml
...
  build:
    name: Build Image
    runs-on: ubuntu-latest
    needs: test
    steps:
      -
        name: Check out the repo
        uses: actions/checkout@v4
      -
        name: Lint Dockerfile
        uses: hadolint/hadolint-action@v3.1.0
      -
        name: Docker Meta
        id: meta
        uses: docker/metadata-action@v5
        with:
          images: |
            inpred/local_app_prepper
          tags: |
            latest
            type=semver,pattern={{version}}
            type=semver,pattern={{major}}.{{minor}}
            type=semver,pattern={{major}}
      -
        name: Login to Dockerhub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}
      -
        name: Build and push image to Docker Hub
        uses: docker/build-push-action@v5
        with:
          push: true
          tags: ${{ steps.meta.outputs.tags }}
          labels: ${{ steps.meta.outputs.labels }}
```

GitHub Actions





All workflows
Showing runs from all workflows

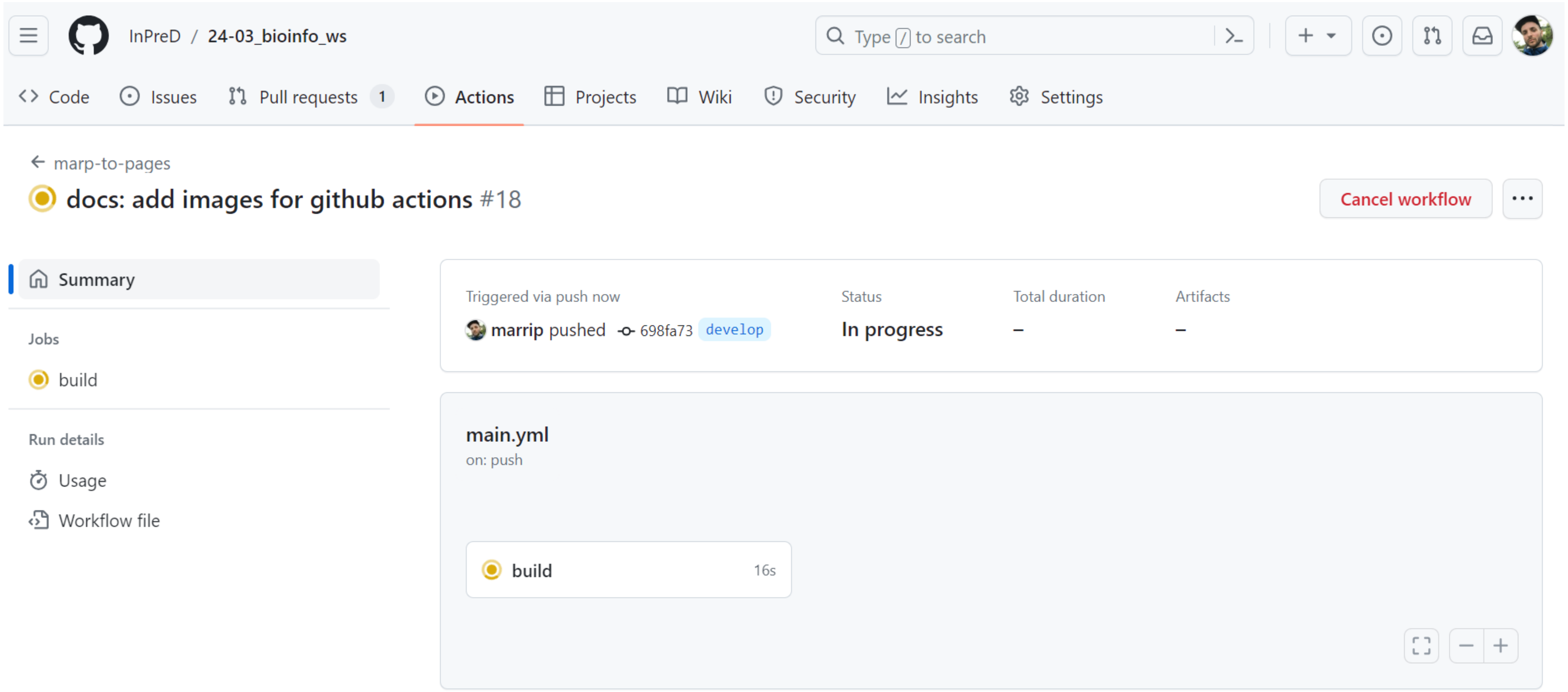
36 workflow runs

pages build and deployment
pages-build-deployment #19: by github-pages bot
11 minutes ago
41s

docs: add github actions
marp-to-pages #17: Commit 34d473e pushed by marrip
develop
12 minutes ago
47s

pages build and deployment
pages-build-deployment #18: by github-pages bot
46 minutes ago
47s

GitHub Actions



← marp-to-pages

docs: add images for github actions #18

Triggered via push now: marrip pushed 698fa73 develop

Status: In progress

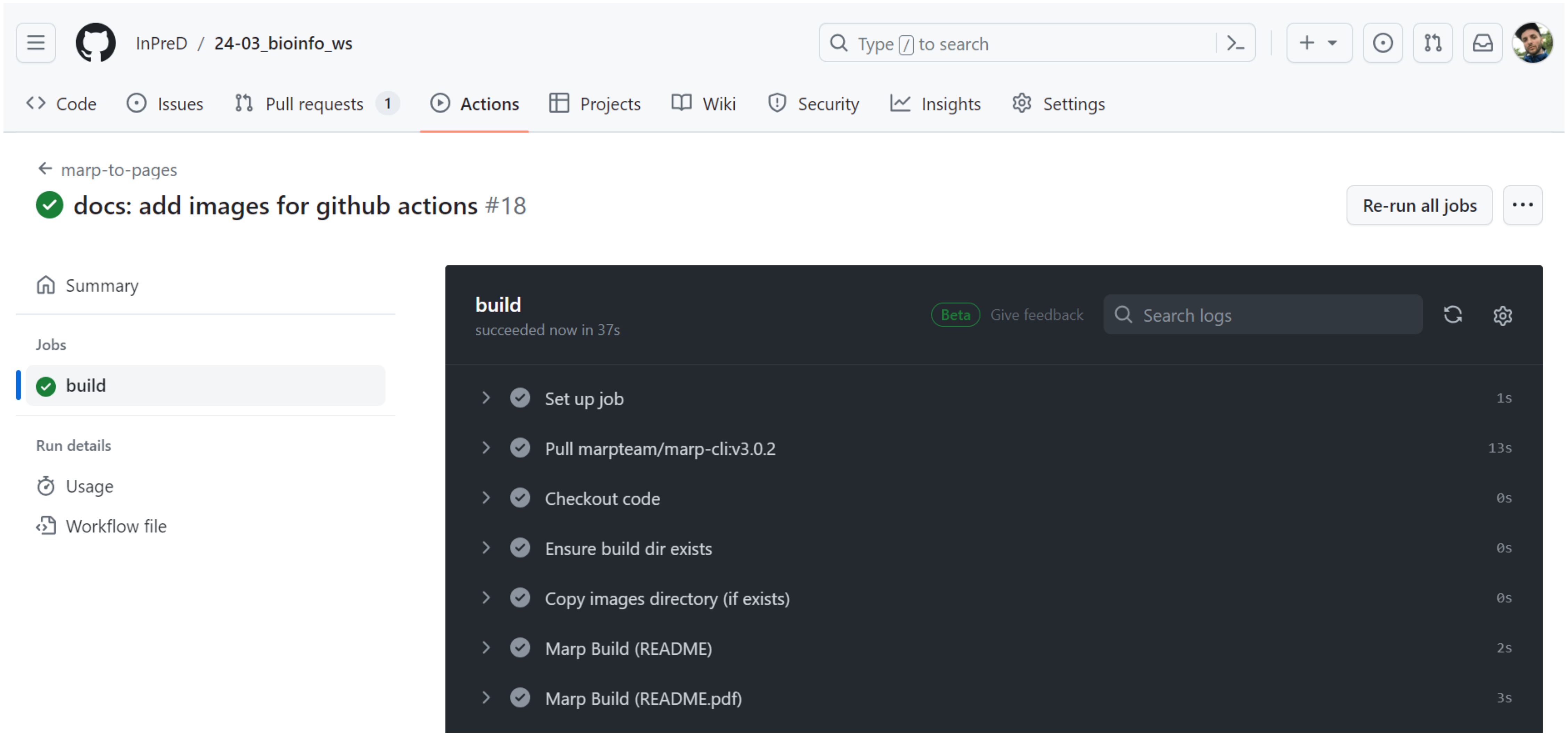
Total duration: –

Artifacts: –

main.yml
on: push

build 16s

GitHub Actions



docs: add images for github actions #18

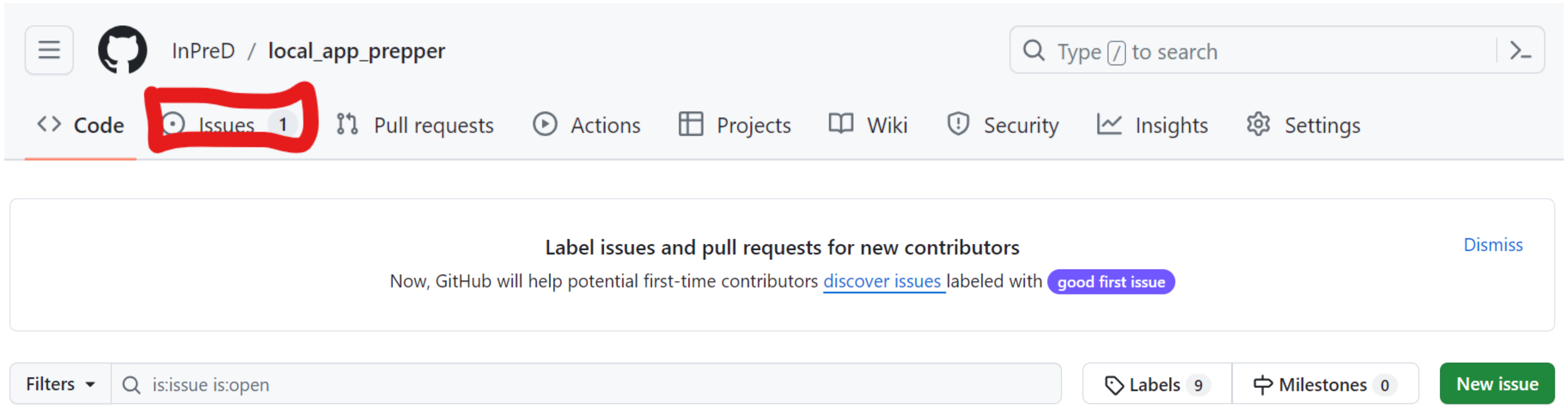
build
succeeded now in 37s

```
Set up job                          1s
Pull marpteam/marp-cli:v3.0.2       13s
Checkout code                       0s
Ensure build dir exists             0s
Copy images directory (if exists)   0s
Marp Build (README)                 2s
Marp Build (README.pdf)             3s
```

2. Development & Collaboration

GitHub workflow

- go to issues and create a `New issue`



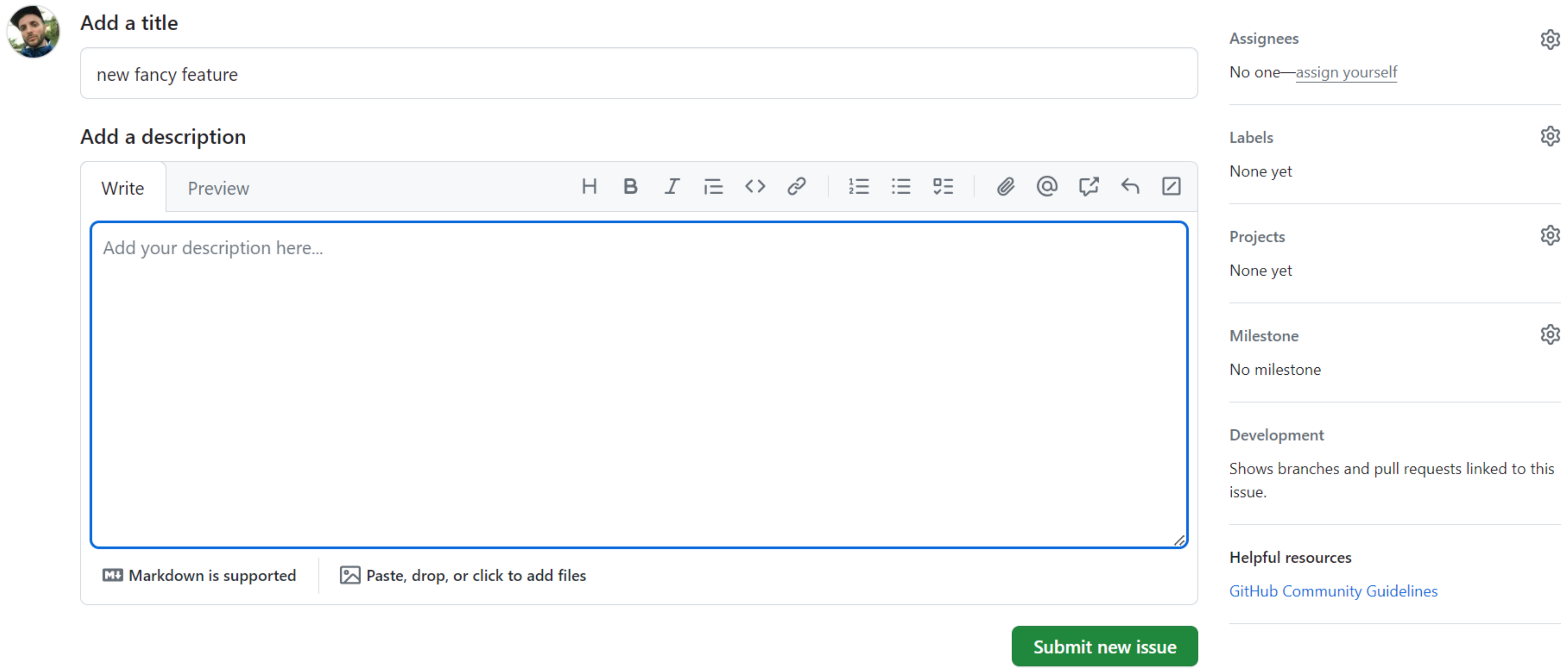
Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

is:issue is:open

GitHub workflow

- give the issue a descriptive title and a description and `Submit new issue`


Add a title

new fancy feature

Add a description

Add your description here...

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

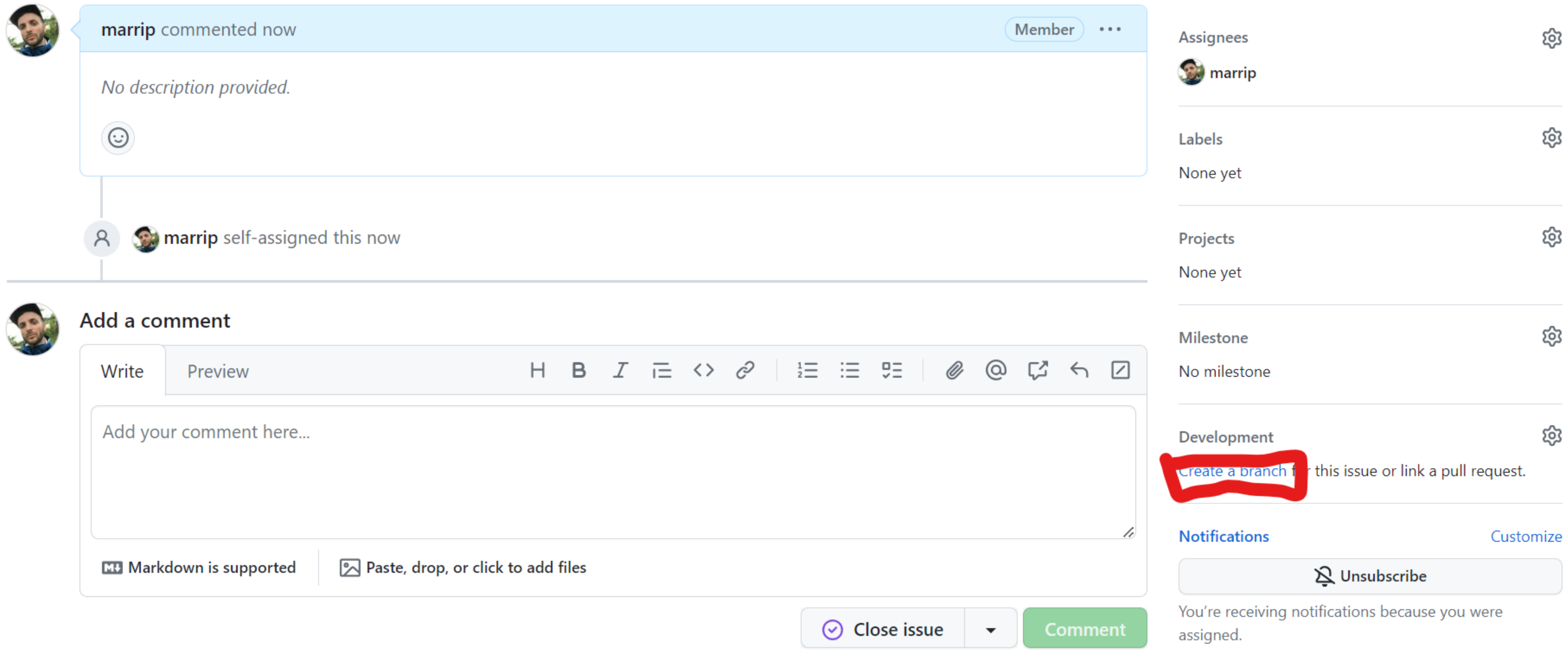
Shows branches and pull requests linked to this issue.

Helpful resources

GitHub Community Guidelines

GitHub workflow

- if you decide to work on the issue (own repository), `Create a branch` via the issue



marrip commented now · Member

No description provided.

marrip self-assigned this now

Add a comment

Write | Preview

Add your comment here...

Markdown is supported · Paste, drop, or click to add files

Close issue · Comment

Assignees: marrip

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Create a branch for this issue or link a pull request.

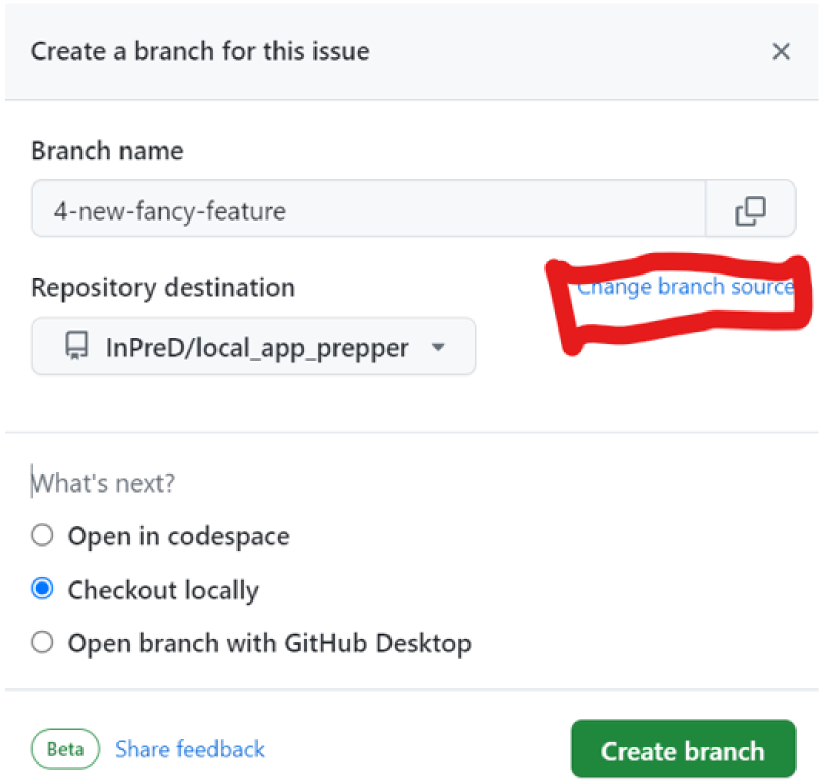
Notifications · Customize

Unsubscribe

You're receiving notifications because you were assigned.

GitHub workflow

- `Change branch source` to `develop` and `Create branch`



Create a branch for this issue

Branch name

4-new-fancy-feature

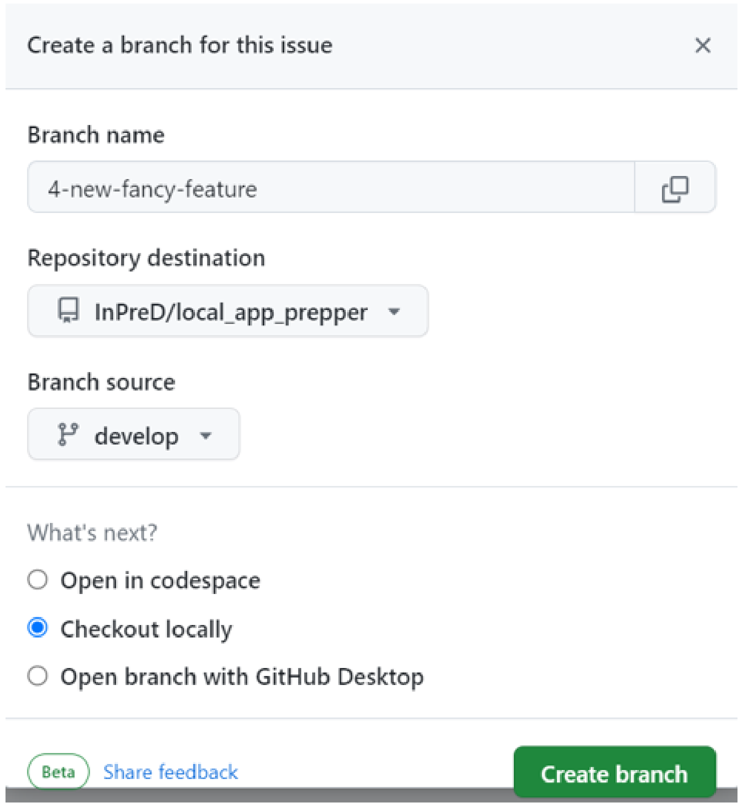
Repository destination

InPreD/local_app_prepper

Change branch source

What's next?

Open in codespace [✓ Checkout locally] Open branch with GitHub Desktop



Create a branch for this issue

Branch name

4-new-fancy-feature

Repository destination

InPreD/local_app_prepper

Branch source

develop

What's next?

Open in codespace [✓ Checkout locally] Open branch with GitHub Desktop

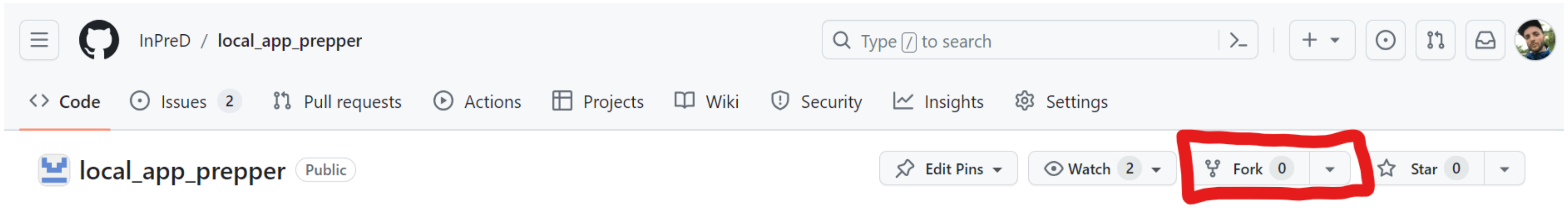
GitHub workflow

- load the new branch to your local repository, check it out and start working
- push your changes back to the remote

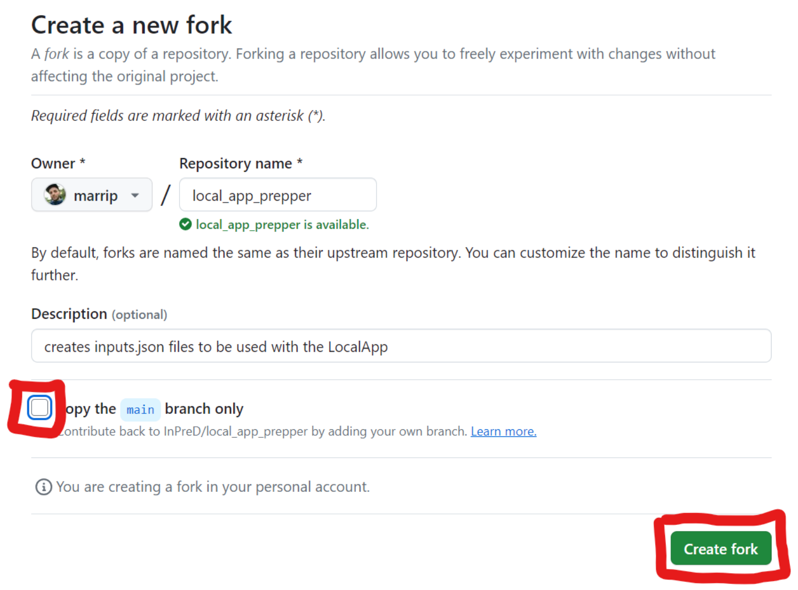
```
$ git pull
$ git checkout 4-new-fancy-feature
$ git add README.md
$ git commit -m "docs: updating docs"
$ git push
```

GitHub workflow

- for repositories you don't have access to, create a fork



GitHub workflow



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.

Required fields are marked with an asterisk (*).

Owner * marrip / Repository name * local_app_prepper

local_app_prepper is available.

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

Description (optional)

creates inputs.json files to be used with the LocalApp

☐ opy the `main` branch only

ontribute back to InPreD/local_app_prepper by adding your own branch. Learn more.

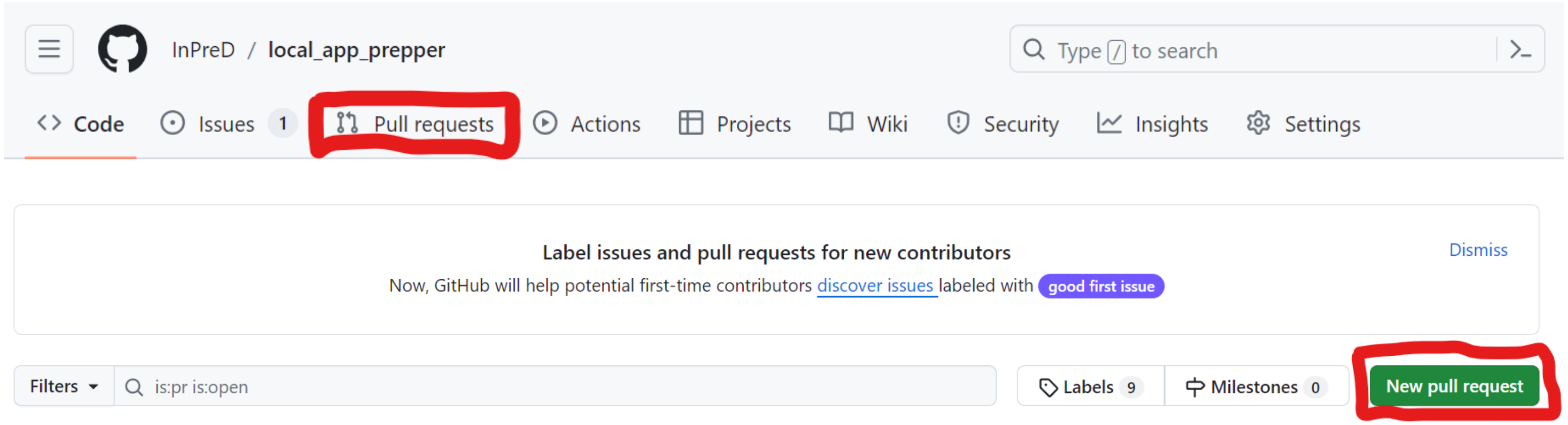
You are creating a fork in your personal account.

GitHub workflow

- once you have a fork, `git clone` your forked repository
- create a new branch and work on that
- `git push` your changes back to the forked remote

GitHub workflow

- when you are done, go to pull requests and create a `New pull request`

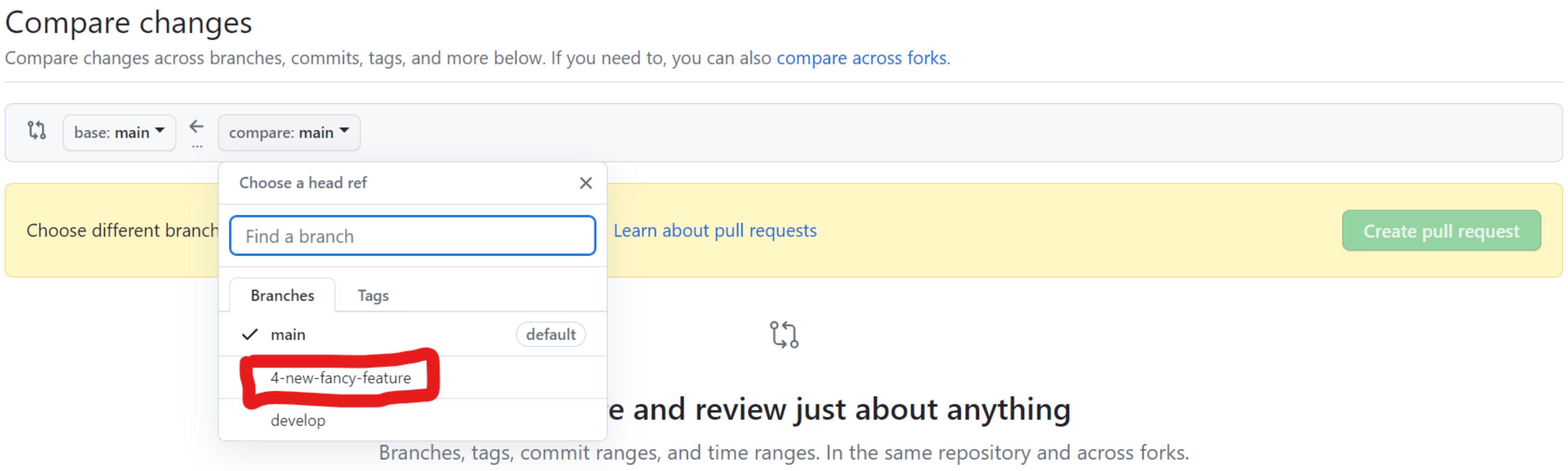


Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

GitHub workflow

- choose `develop` as `base` and your new feature branch for `compare`



Compare changes

Compare changes across branches, commits, tags, and more below. If you need to, you can also compare across forks.

base: main ← compare: main

Choose a head ref

Find a branch

Branches Tags

main default

4-new-fancy-feature

develop

Choose different branch... Learn about pull requests

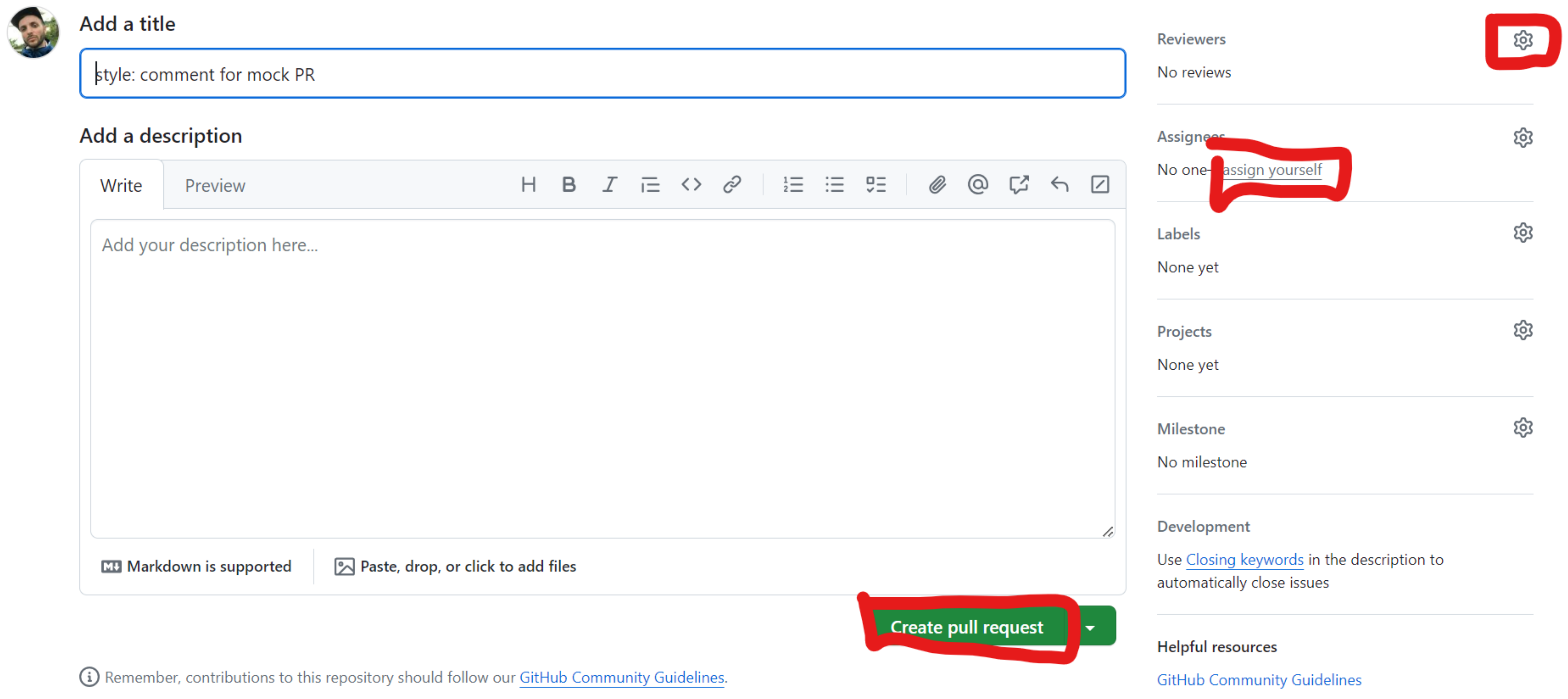
Create pull request

...e and review just about anything

Branches, tags, commit ranges, and time ranges. In the same repository and across forks.

GitHub workflow

- `assign yourself`, add at least one reviewer (cog icon), provide some context and `Create pull request`


Add a title

style: comment for mock PR

Add a description

Add your description here...

Remember, contributions to this repository should follow our GitHub Community Guidelines.

Reviewers

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

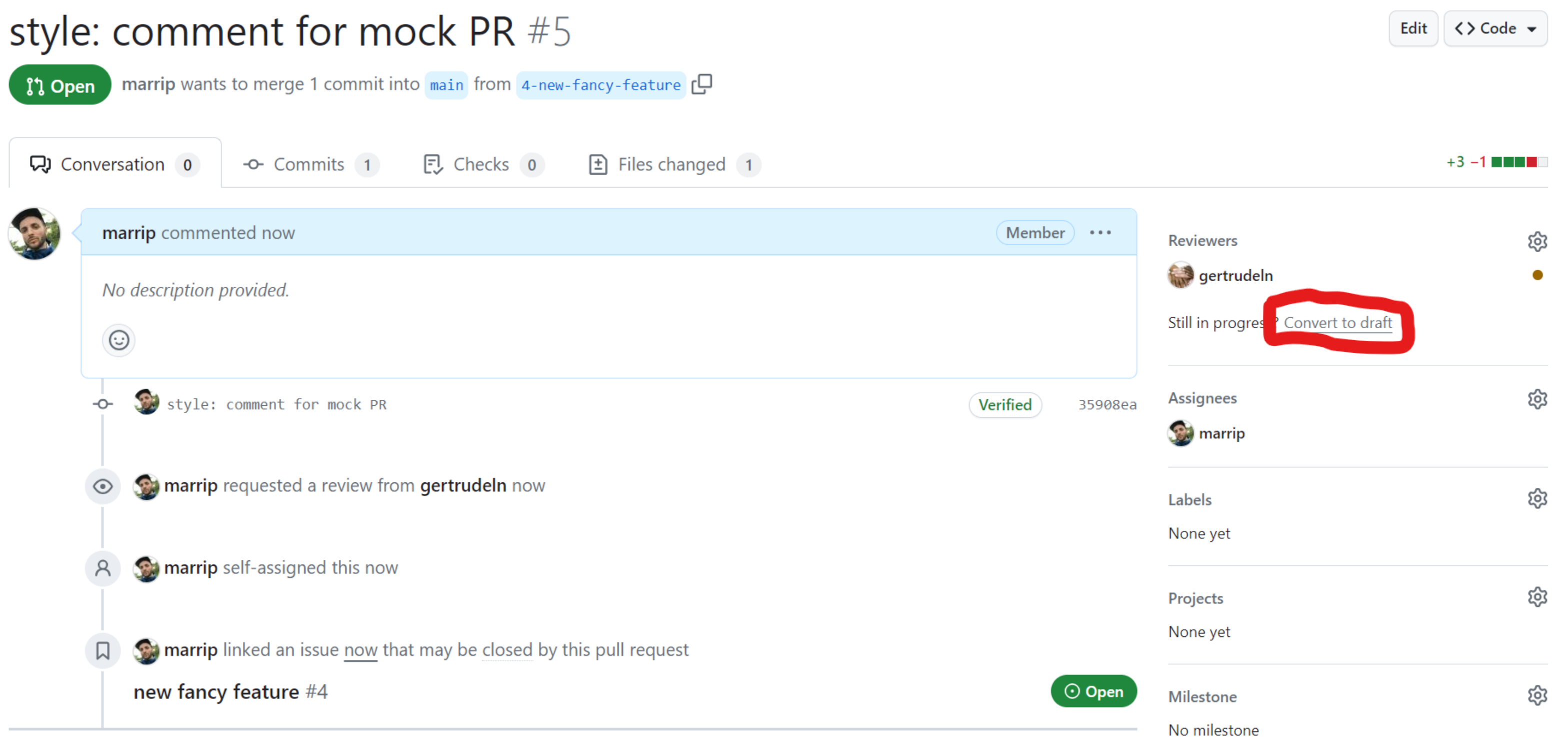
Use Closing keywords in the description to automatically close issues

Helpful resources

GitHub Community Guidelines

GitHub workflow

- if you still want to work on the pull request, you can `Convert to draft` to let the reviewers know that it is not done yet
- otherwise you can just wait for them to review your changes



style: comment for mock PR #5

Open marrip wants to merge 1 commit into `main` from `4-new-fancy-feature`

marrip commented now — Member

No description provided.

`style: comment for mock PR` Verified 35908ea

marrip requested a review from gertrudeln now

marrip self-assigned this now

marrip linked an issue now that may be closed by this pull request

new fancy feature #4 Open

Reviewers: gertrudeln

Still in progress? Convert to draft

Assignees: marrip

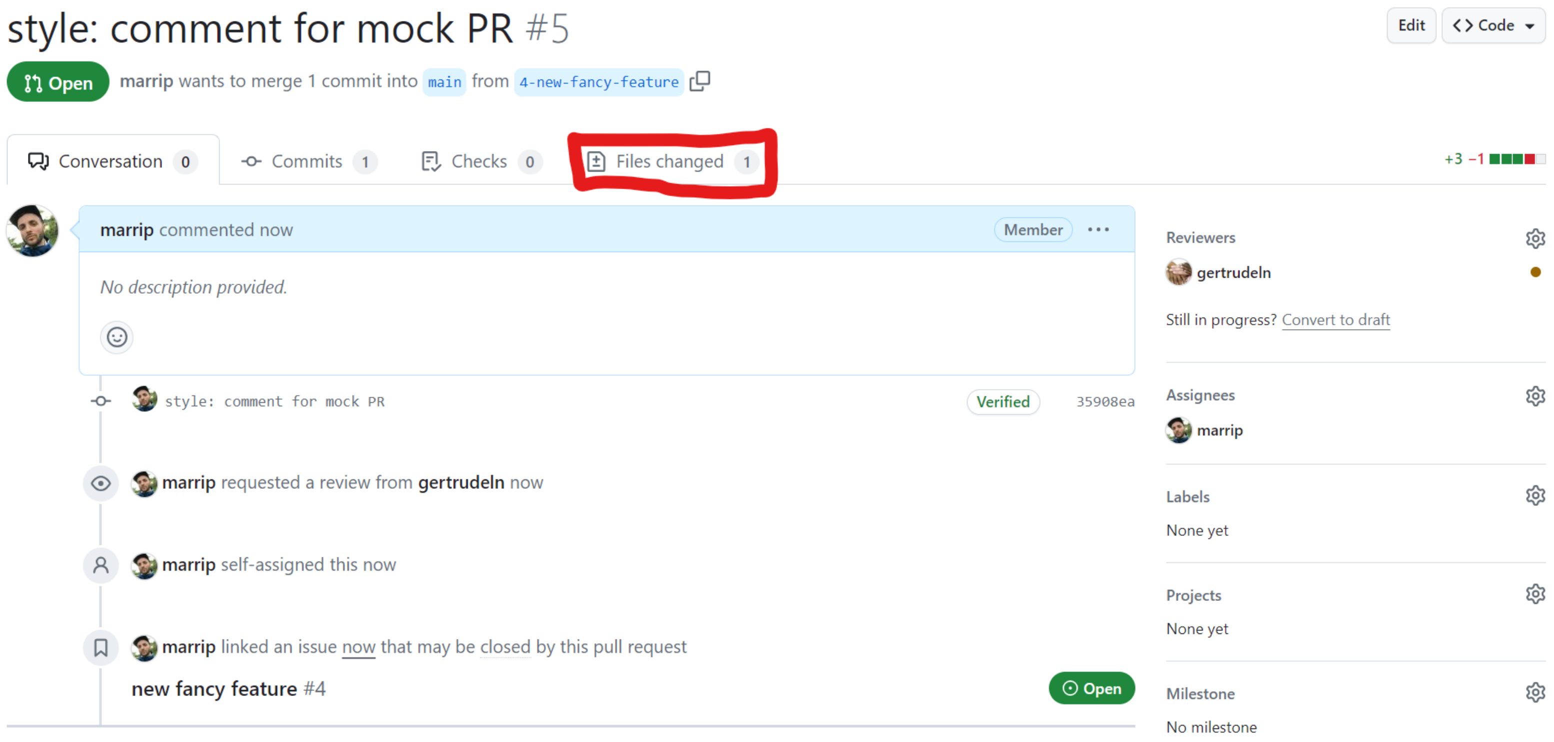
Labels: None yet

Projects: None yet

Milestone: No milestone

GitHub workflow

- as a reviewer, make your you check your email notifications to see if there is pull requests waiting for you
- open the pull request and start the review in the `Files changed` tab



style: comment for mock PR #5

Open marrip wants to merge 1 commit into `main` from `4-new-fancy-feature`

marrip commented now — Member

No description provided.

style: comment for mock PR — Verified — 35908ea

marrip requested a review from gertrudeln now

marrip self-assigned this now

marrip linked an issue now that may be closed by this pull request

new fancy feature #4 — Open

Reviewers: gertrudeln

Still in progress? Convert to draft

Assignees: marrip

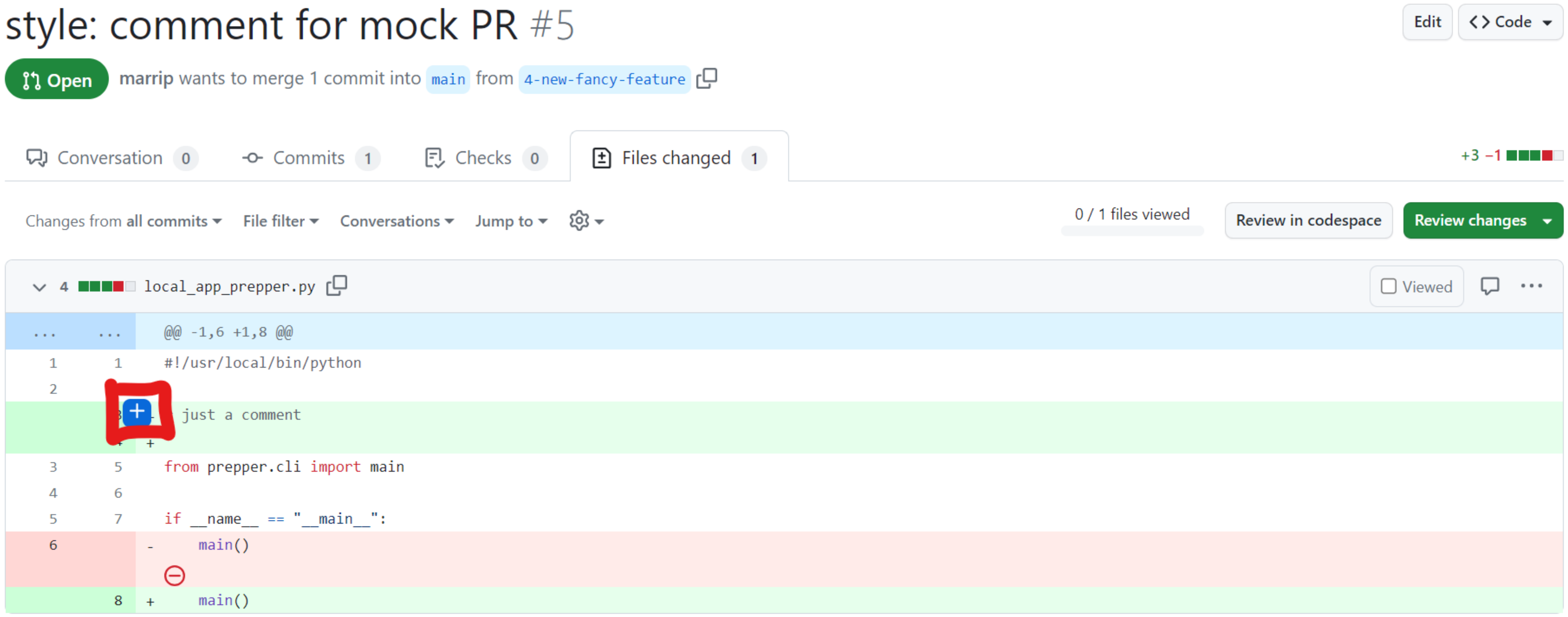
Labels: None yet

Projects: None yet

Milestone: No milestone

GitHub workflow

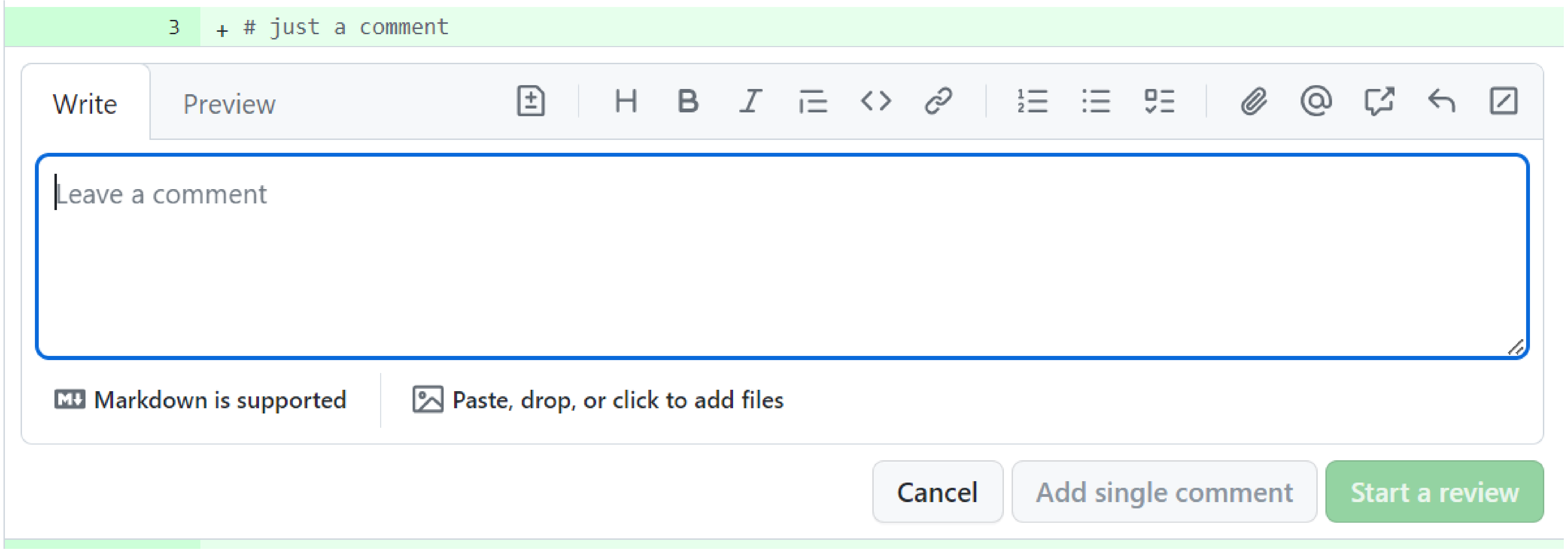
- you can leave comments and suggestions in the code by hovering over the line with the changes and clicking on `+`



```diff
@@ -1,6 +1,8 @@
 #!/usr/local/bin/python

+# just a comment
+
 from prepper.cli import main

 if __name__ == "__main__":
-    main()
+    main()
```

GitHub workflow

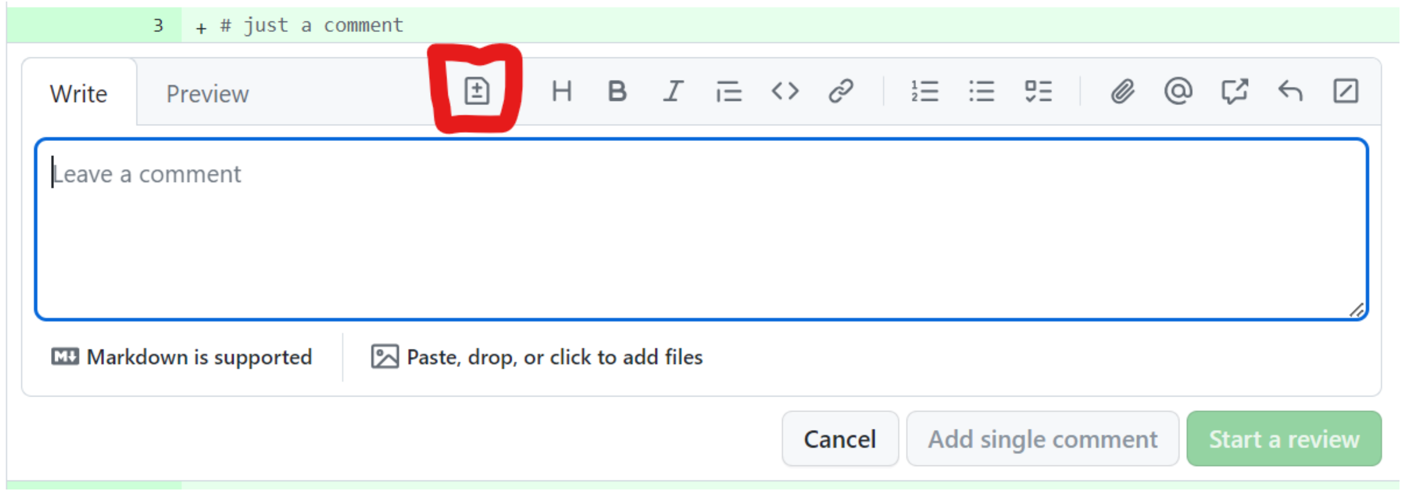
- you can type your comment



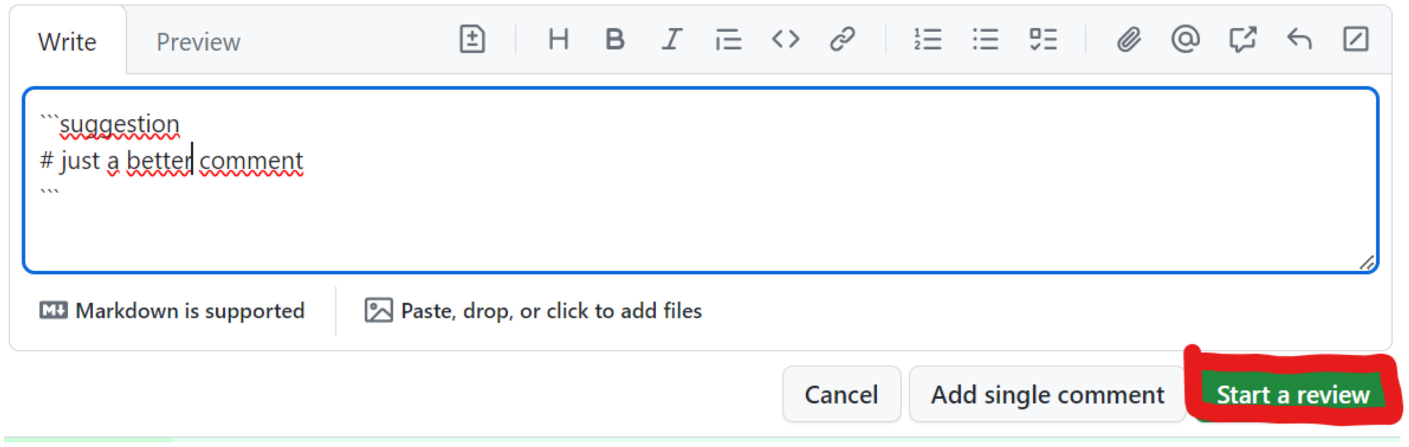
```
3  + # just a comment
```

GitHub workflow

- or you leave a suggestion, ideally you click `Start a review` to initialise the reviewing process



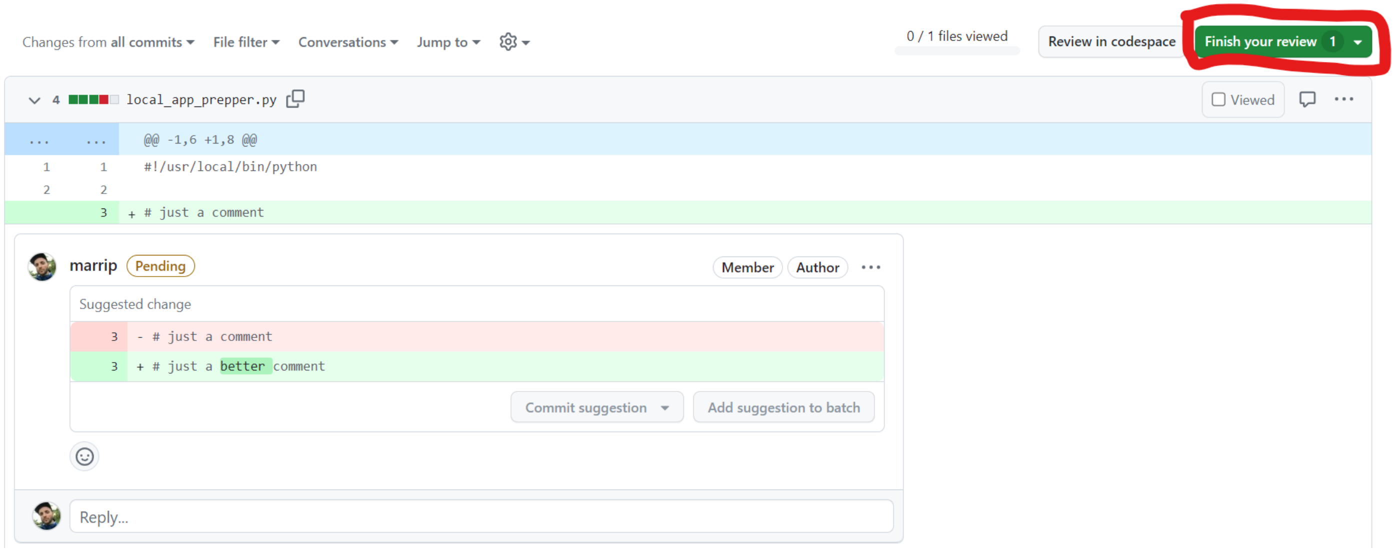
```
3  + # just a comment
```



~~~
```suggestion
# just a better comment
```
~~~

GitHub workflow

- when you are done with reviewing, `Finish your review`



local_app_prepper.py

```
@@ -1,6 +1,8 @@
1 1   #!/usr/local/bin/python
2 2
  3 + # just a comment
```

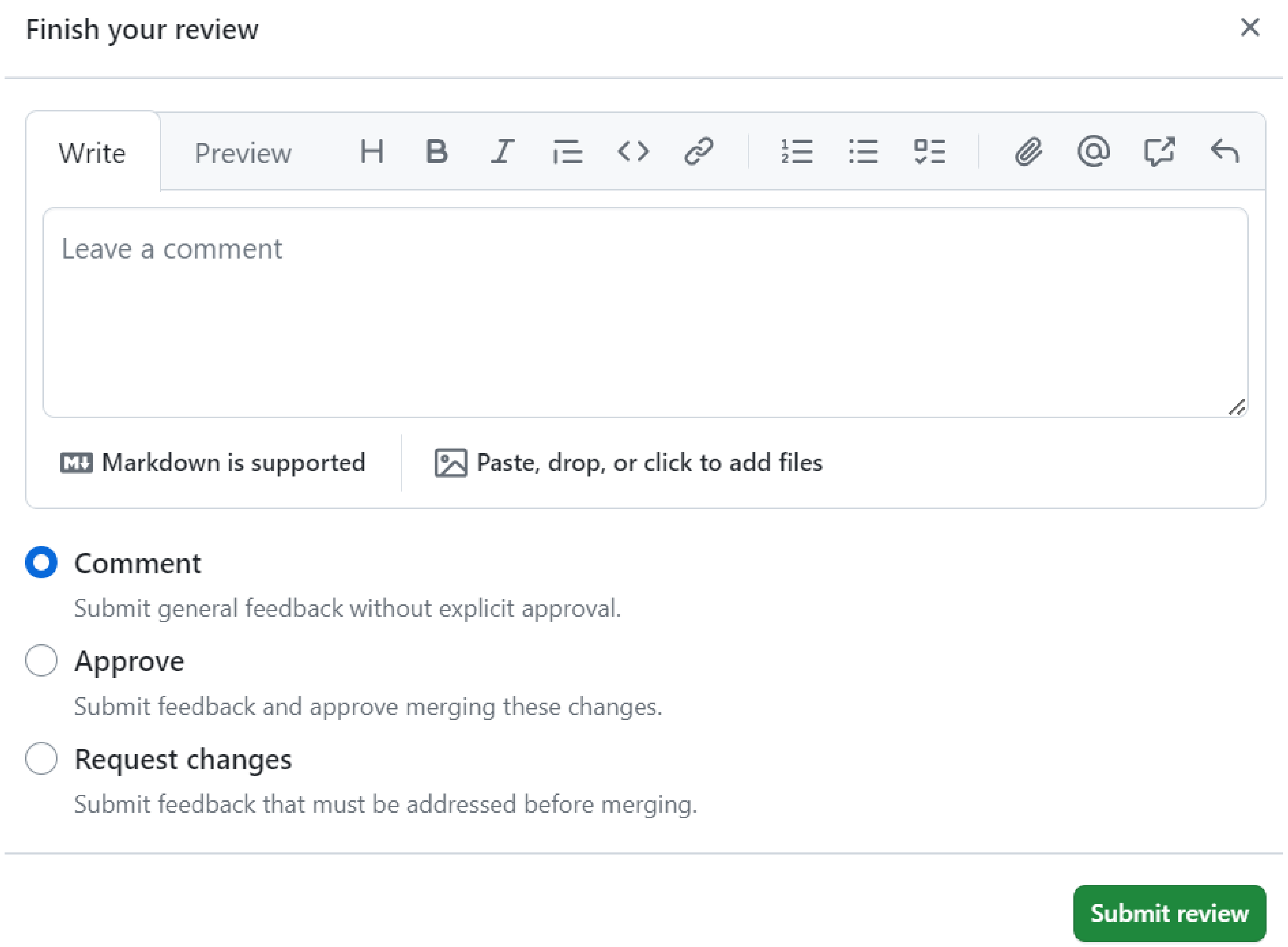
marrip Pending Member Author

Suggested change

```
3 - # just a comment
3 + # just a better comment
```

GitHub workflow

- again, leave a comment if you like, and choose if you just want to `Comment`, `Approve` or `Request changes`


Finish your review

Leave a comment

Markdown is supported

Paste, drop, or click to add files

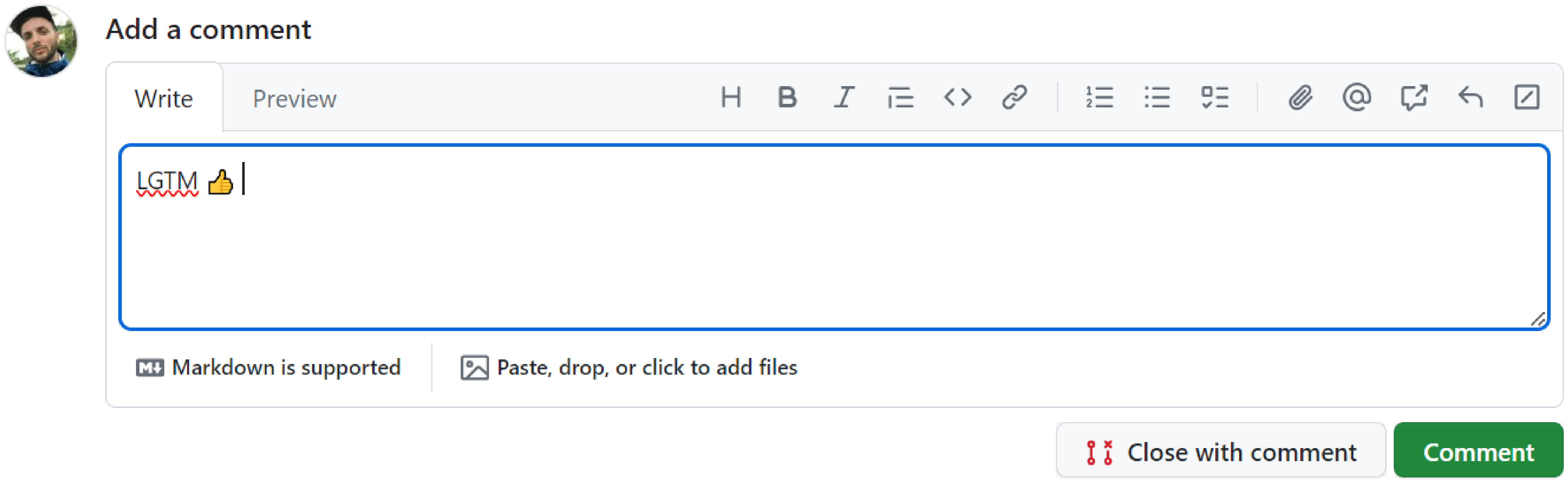
[✓ Comment]
Submit general feedback without explicit approval.

Approve
Submit feedback and approve merging these changes.

Request changes
Submit feedback that must be addressed before merging.

GitHub workflow

- you can add a general comment to the pull request under `Conversation`

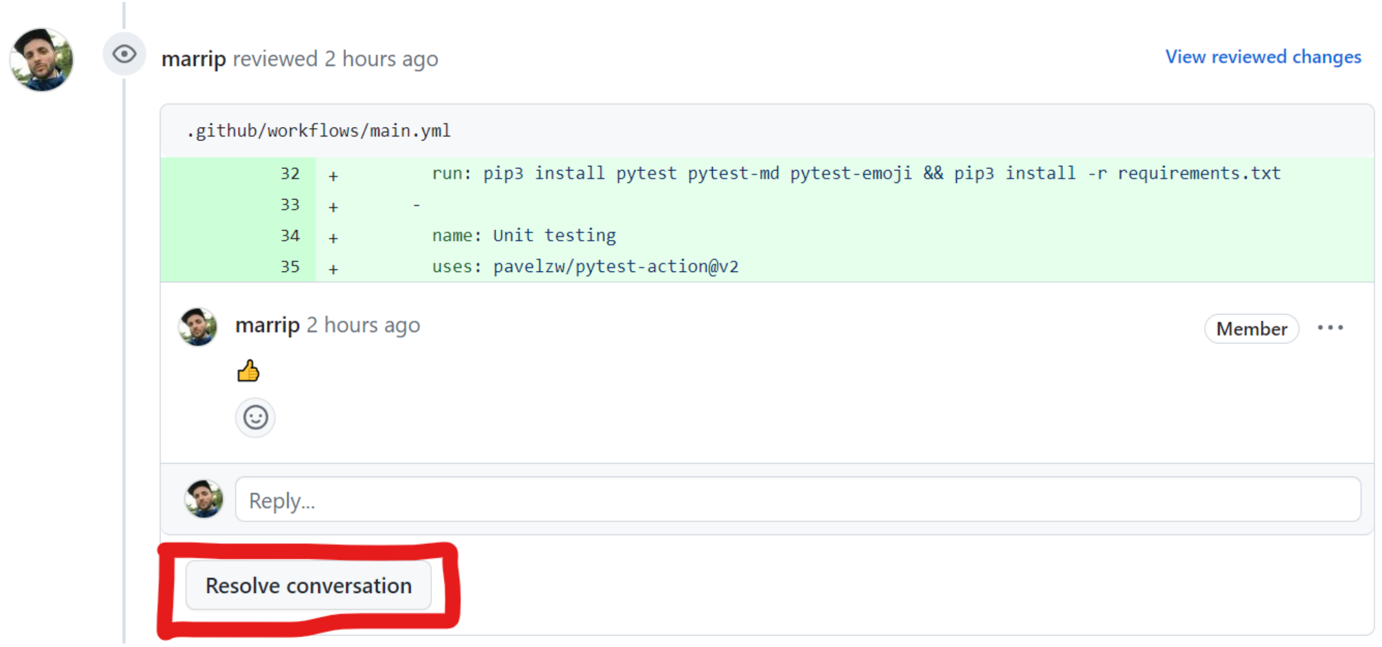


Add a comment

LGTM 👍

GitHub workflow

- after the reviewer left their comments and suggestions, you can address them one by one by replying or applying the suggested changes
- whenever a certain comment/suggestion is handled (discussion comes to conclusion, suggestion was applied), you can resolve it



marrip reviewed 2 hours ago

.github/workflows/main.yml

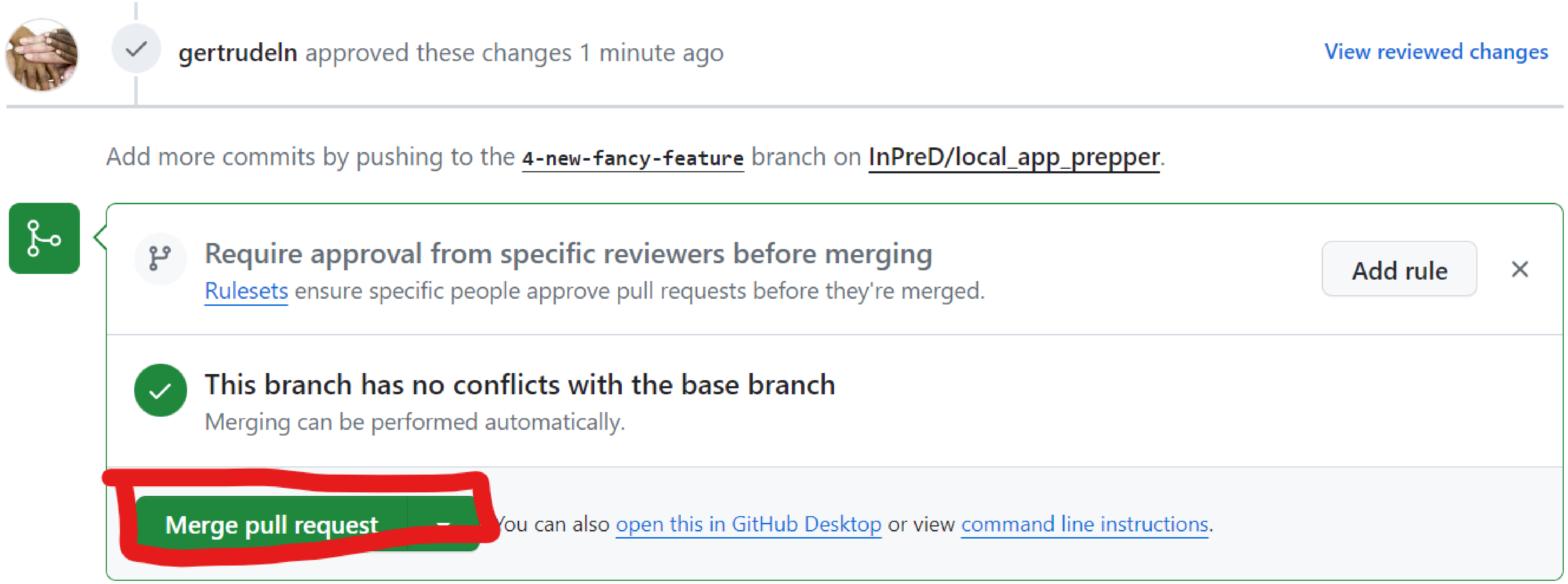
```
+         run: pip3 install pytest pytest-md pytest-emoji && pip3 install -r requirements.txt
+       -
+         name: Unit testing
+         uses: pavelzw/pytest-action@v2
```

marrip 2 hours ago

👍

GitHub workflow

- as soon as the reviewers gave you an approval, you can finally `Merge pull request`



gertrudeln approved these changes 1 minute ago

Add more commits by pushing to the `4-new-fancy-feature` branch on InPreD/local_app_prepper.

Require approval from specific reviewers before merging
Rulesets ensure specific people approve pull requests before they're merged.

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

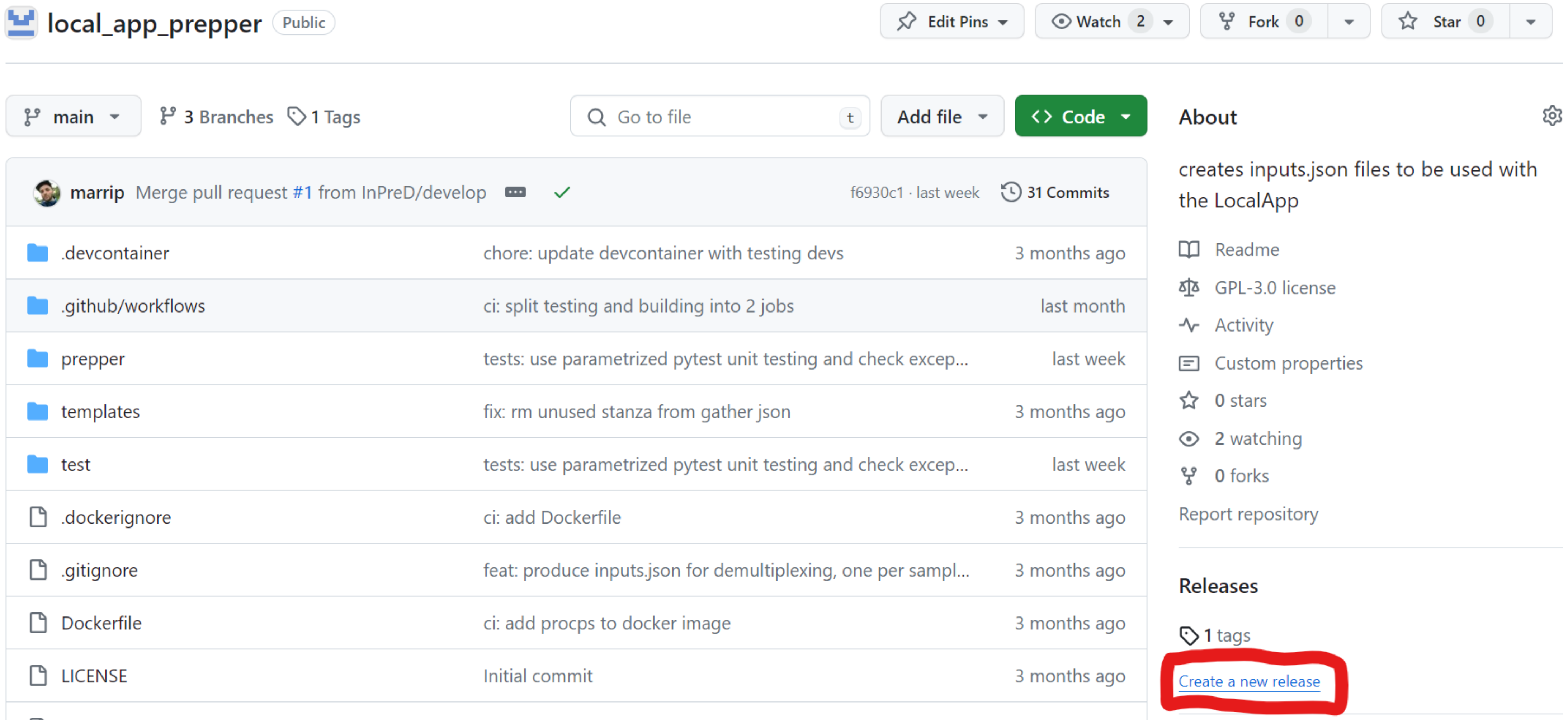
2. Development & Collaboration

Release

- releases should be from `main` branch
- good practice is to open a pull request for `develop` into `main` when you are done with the desired features

Release

- whenever you are ready for a new release, `create a new release`



local_app_prepper Public

Edit Pins | Watch 2 | Fork 0 | Star 0

main | 3 Branches | 1 Tags | Go to file | Add file | Code

marrip Merge pull request #1 from InPreD/develop ✓ f6930c1 · last week 31 Commits

.devcontainer	chore: update devcontainer with testing devs	3 months ago
.github/workflows	ci: split testing and building into 2 jobs	last month
prepper	tests: use parametrized pytest unit testing and check excep...	last week
templates	fix: rm unused stanza from gather json	3 months ago
test	tests: use parametrized pytest unit testing and check excep...	last week
.dockerignore	ci: add Dockerfile	3 months ago
.gitignore	feat: produce inputs.json for demultiplexing, one per sampl...	3 months ago
Dockerfile	ci: add procps to docker image	3 months ago
LICENSE	Initial commit	3 months ago

About

creates inputs.json files to be used with the LocalApp

Readme
GPL-3.0 license
Activity
Custom properties
0 stars
2 watching
0 forks
Report repository

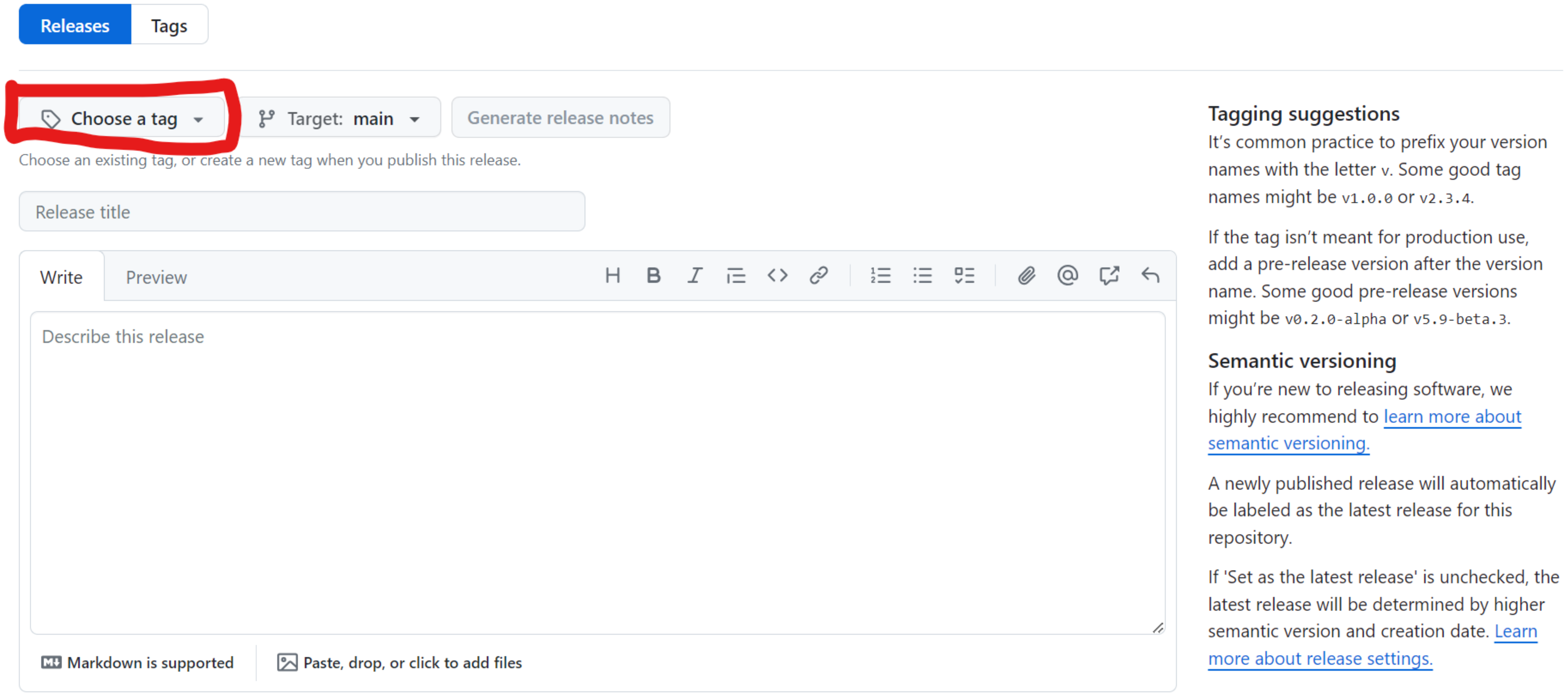
Releases

1 tags

Create a new release

Release

- add a title and a description for your release and `Choose a tag`



Releases | Tags

Choose a tag | Target: main | Generate release notes

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

Release title

Write | Preview

Describe this release

Markdown is supported | Paste, drop, or click to add files

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

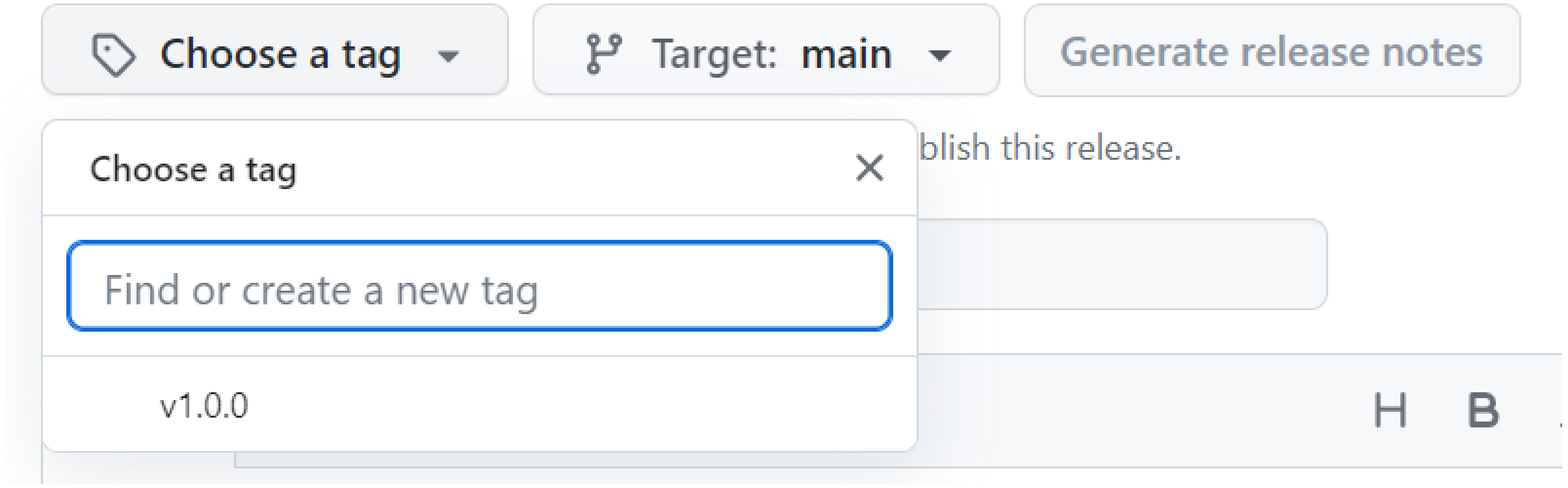
If you're new to releasing software, we highly recommend to learn more about semantic versioning.

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.

Release

- ideally, you choose a tag according to semantic versioning



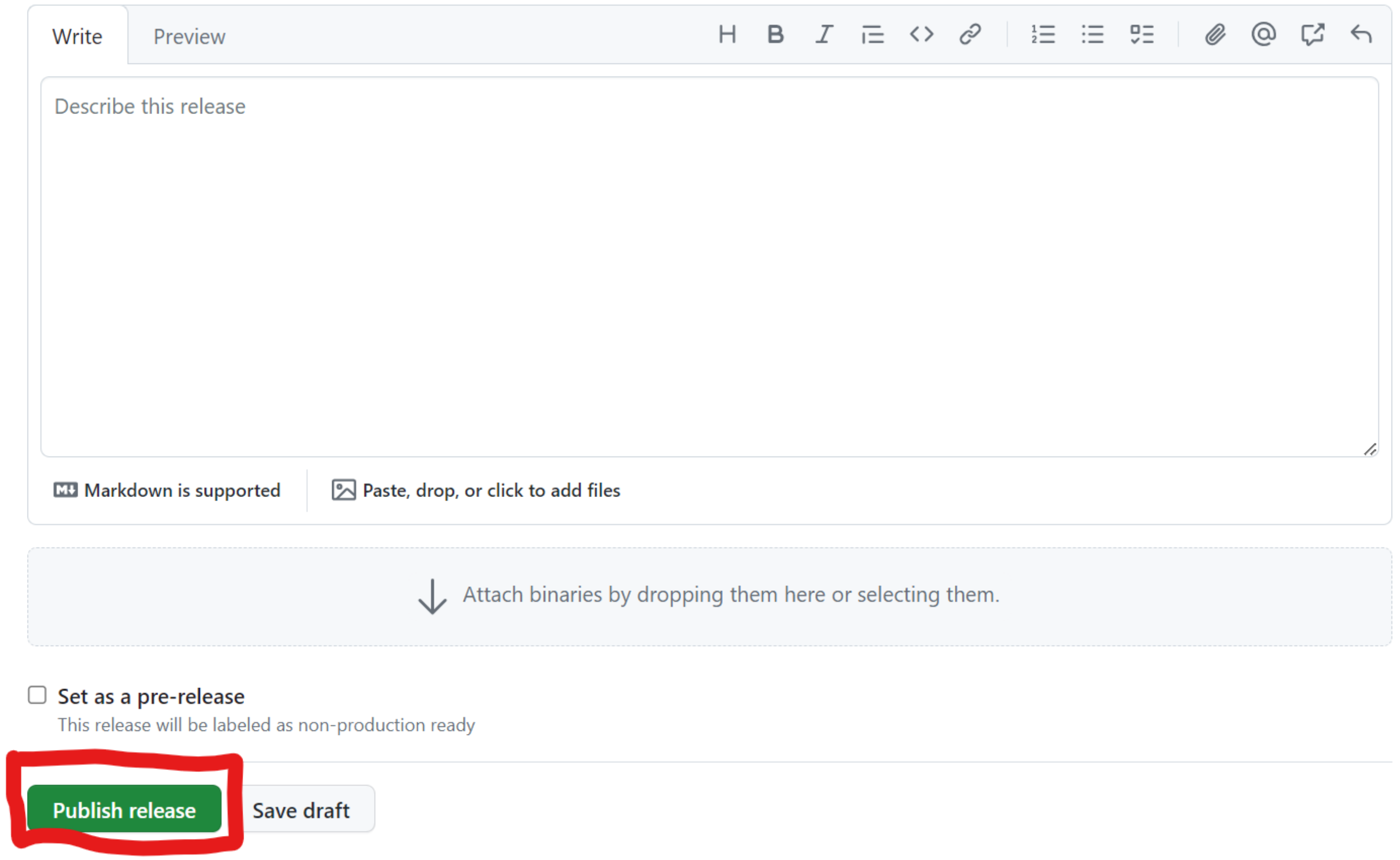
Release

Semantic versioning

- version tag should be **MAJOR.MINOR.PATCH**
- you increment one of the three depending on the change
  - **MAJOR**: version when you make incompatible API changes
  - **MINOR**: version when you add functionality in a backward compatible manner
  - **PATCH**: version when you make backward compatible bug fixes

Release

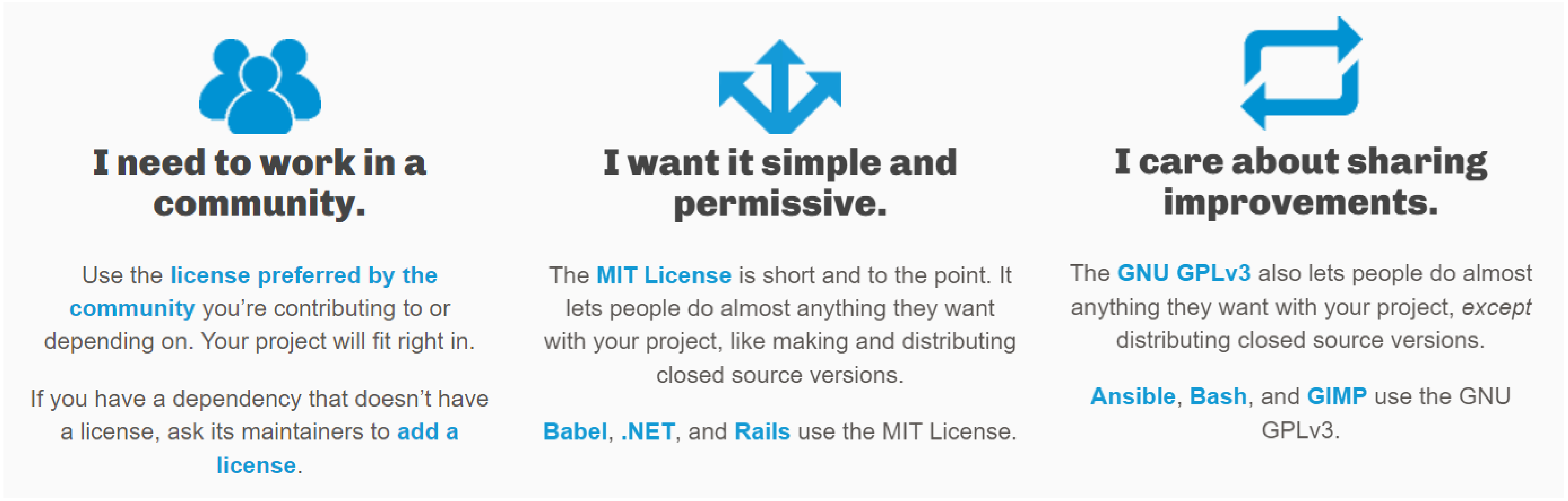
- when you are satisfied with your release, `Publish release`


Write
Preview
Describe this release
Markdown is supported
Paste, drop, or click to add files
Attach binaries by dropping them here or selecting them.
Set as a pre-release
This release will be labeled as non-production ready
Publish release
Save draft

2. Development & Collaboration

Licensing

- let's discuss



I need to work in a community.

Use the **license preferred by the community** you're contributing to or depending on. Your project will fit right in.

If you have a dependency that doesn't have a license, ask its maintainers to **add a license**.

I want it simple and permissive.

The **MIT License** is short and to the point. It lets people do almost anything they want with your project, like making and distributing closed source versions.

Babel, **.NET**, and **Rails** use the MIT License.

I care about sharing improvements.

The **GNU GPLv3** also lets people do almost anything they want with your project, *except* distributing closed source versions.

Ansible, **Bash**, and **GIMP** use the GNU GPLv3.

2. Resources

- About Git
- Gitflow workflow
- GitHub Actions
- Semantic versioning
- Licensing

3. Nextflow

Short introduction

- workflow manager that enables scalable and reproducible scientific workflows using software containers
- an extension of groovy which is object-oriented programming language for the Java platform
- **nf-core**: project/community that develops framework for nextflow including guidelines, tools, modules, subworkflows, pipelines and test data

3. Nextflow

Requirements

- POSIX compatible system (e.g. Linux, Os X)
- Bash
- Java ≥ 11 / ≤ 21
- Docker/Singularity

3. Nextflow

Installation

```
$ curl -s https://get.nextflow.io | bash
$ chmod +x nextflow
```

or

```
$ wget -O nextflow https://github.com/nextflow-io/nextflow/releases/download/v23.10.1/nextflow-23.10.1-all
```

or via browser at https://github.com/nextflow-io/nextflow/releases

3. Nextflow

Something

- stubbing

3. Nextflow

nf-core template

3. Resources

- nextflow.io
- nf-co.re
- nf-core github

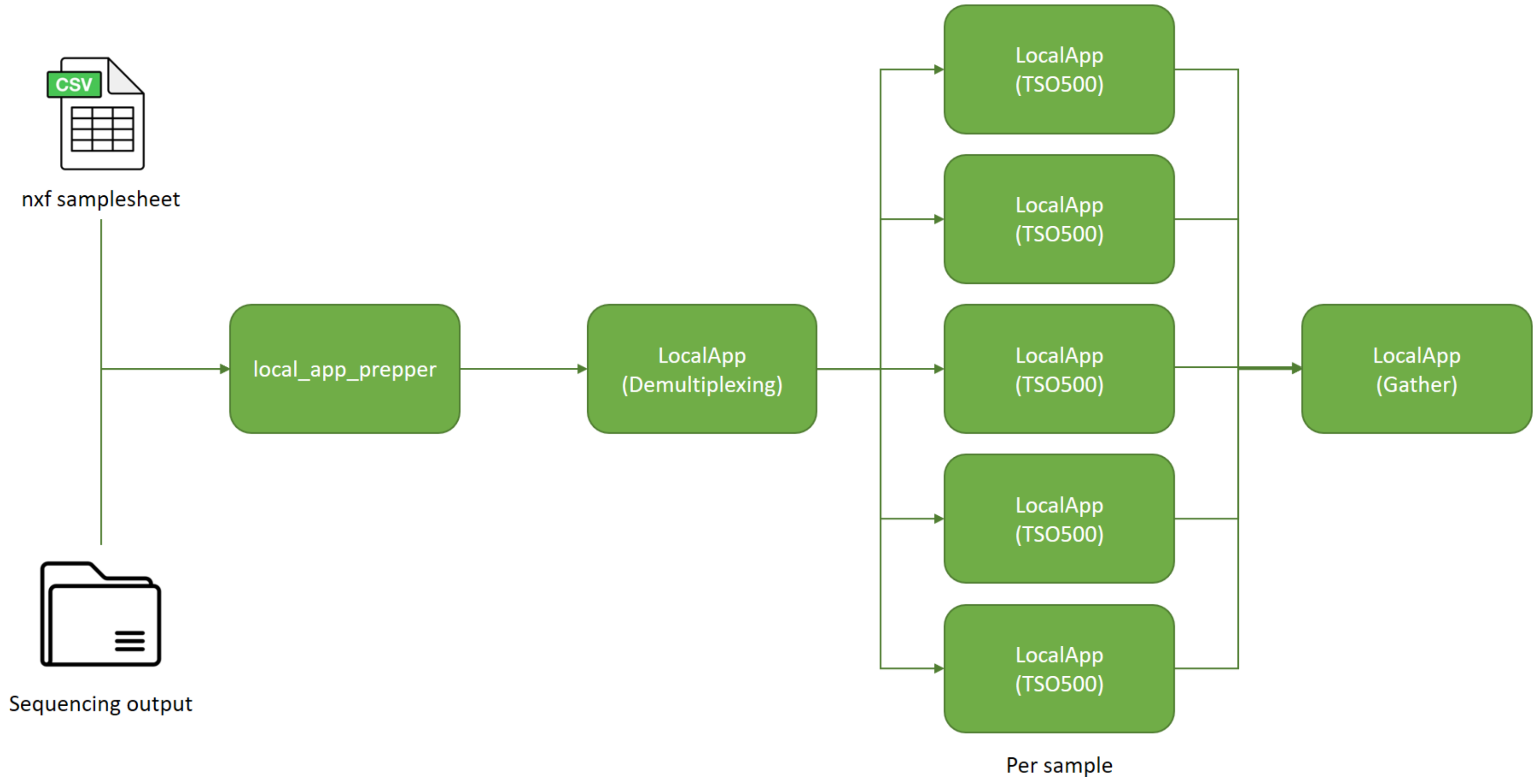
4. `tso500_nxf_workflow`

Status update

- modified nf-core template (removed unnecessary functionality, config and metadata files)
- added devcontainer to have controlled environment (dind and sind available)
- stubbing data available
- containing three modules so far (`localapp_prepper`, `LocalApp`, `dumpsoftwareversions`)
- using nf-validation plugin

4. `tso500_nxf_workflow`

Overview



4. `tso500_nxf_workflow`

Demonstration

4. `tso500_nxf_workflow`

Outlook

- `samplesheet_generator`
- `tsoppi` (requires some restructuring)
- `PRONTO`
- include configuration files for each node
- Documentation

4. Resources

- repository
- local_app_prepper
- samplesheet_generator

5. Python

- general (best practice, cli)
- unit testing (pytest)

Resources

- nf-core
- pytest unittesting