

Bioinformatics session

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



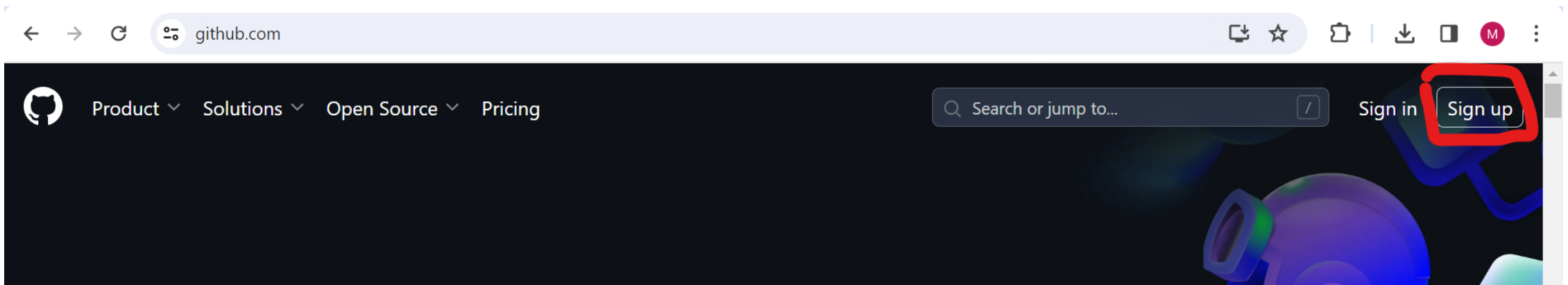
Overview

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

1. Setup

1.1. Create a GitHub account

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



1.1. Create a GitHub account

- enter your email

Welcome to GitHub!

Let's begin the adventure

Enter your email*

→ coder@inpred.no

Continue

1.1. Create a GitHub account

- set a password

Welcome to GitHub!

Let's begin the adventure

Enter your email*

✓ coder@inpred.no

Create a password*

→ ••••••••



Continue

1.1 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

Continue

1.1 Create a GitHub account

- choose email preferences

Email preferences

☐ Receive occasional product updates and announcements.

Continue

1.1 Create a GitHub account

- solve the puzzle



1.1 Create a GitHub account

- create your account



1.1 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

Open GitHub

1.1 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?

<input checked="" type="radio"/> Just me	<input type="radio"/> 2-5	<input type="radio"/> 5-10
<input type="radio"/> 10-20	<input type="radio"/> 20-50	<input type="radio"/> 50+




Are you a student or teacher?

<input checked="" type="radio"/> N/A	<input type="radio"/> Student	<input type="radio"/> Teacher
--------------------------------------	-------------------------------	-------------------------------

Continue

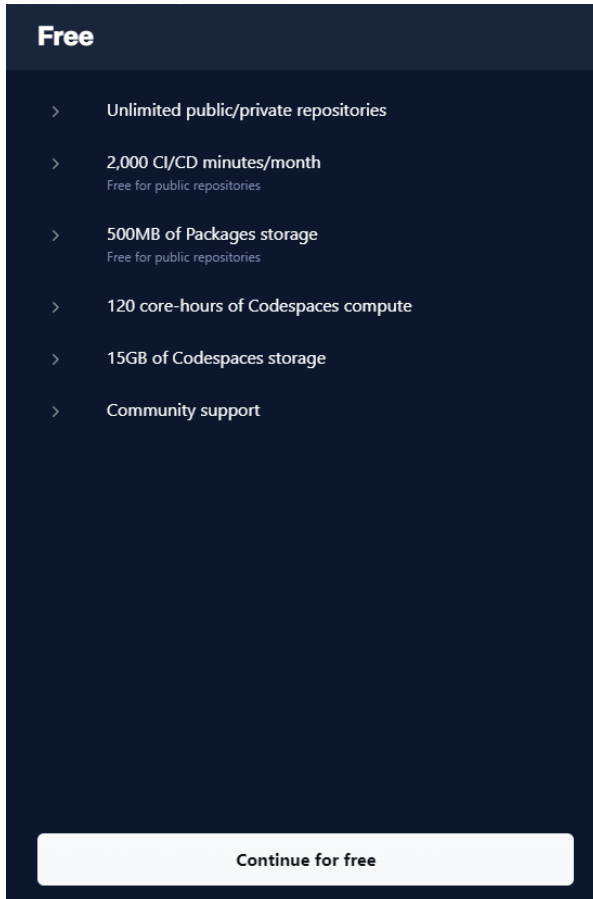
What specific features are you interested in using?

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.

1.1 Create a GitHub account

- choose the free plan



1.2. Be added to InPreD organisation at GitHub

1.3. Resources

- [Getting started with your GitHub account](#)

2. Development & Collaboration

2.1. Short **git** introduction

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

2.1.1 **git config**

```
git config --global user.name <your name>  
git config --global user.email <your email>
```

2.1.2. 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

2.1.2.1. 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

2.1.2.1. 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

2.1.2. Basic `git` commands

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

`git branch` : show local branches

`git merge` : merge branches

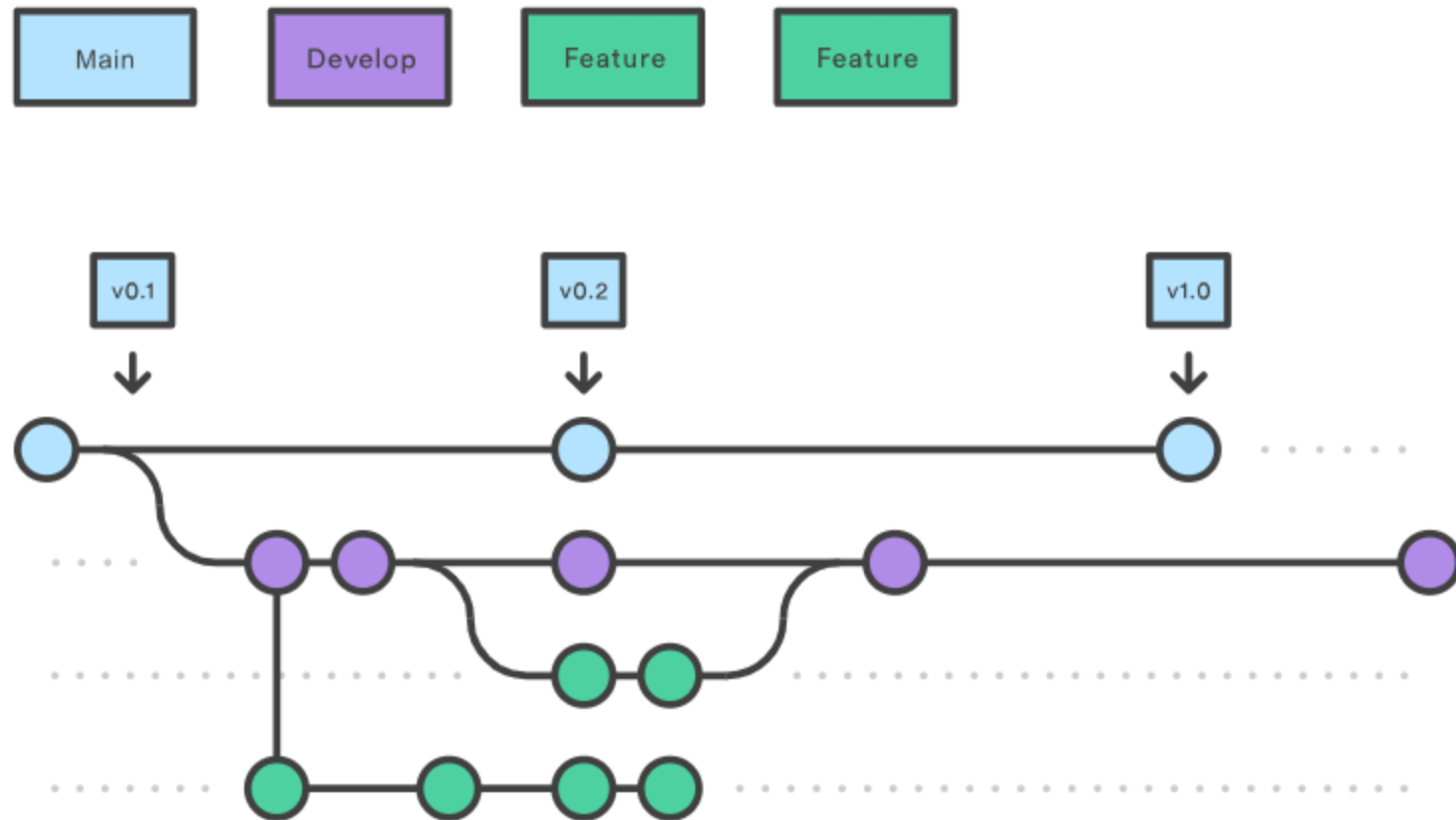
`git pull` : load changes from remote counterpart

`git push` : upload changes to remote counterpart

2.2. Branching model: simplified 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`

2.2. Branching model: simplified Gitflow workflow



2.3. 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`

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


2.3. GitHub Actions

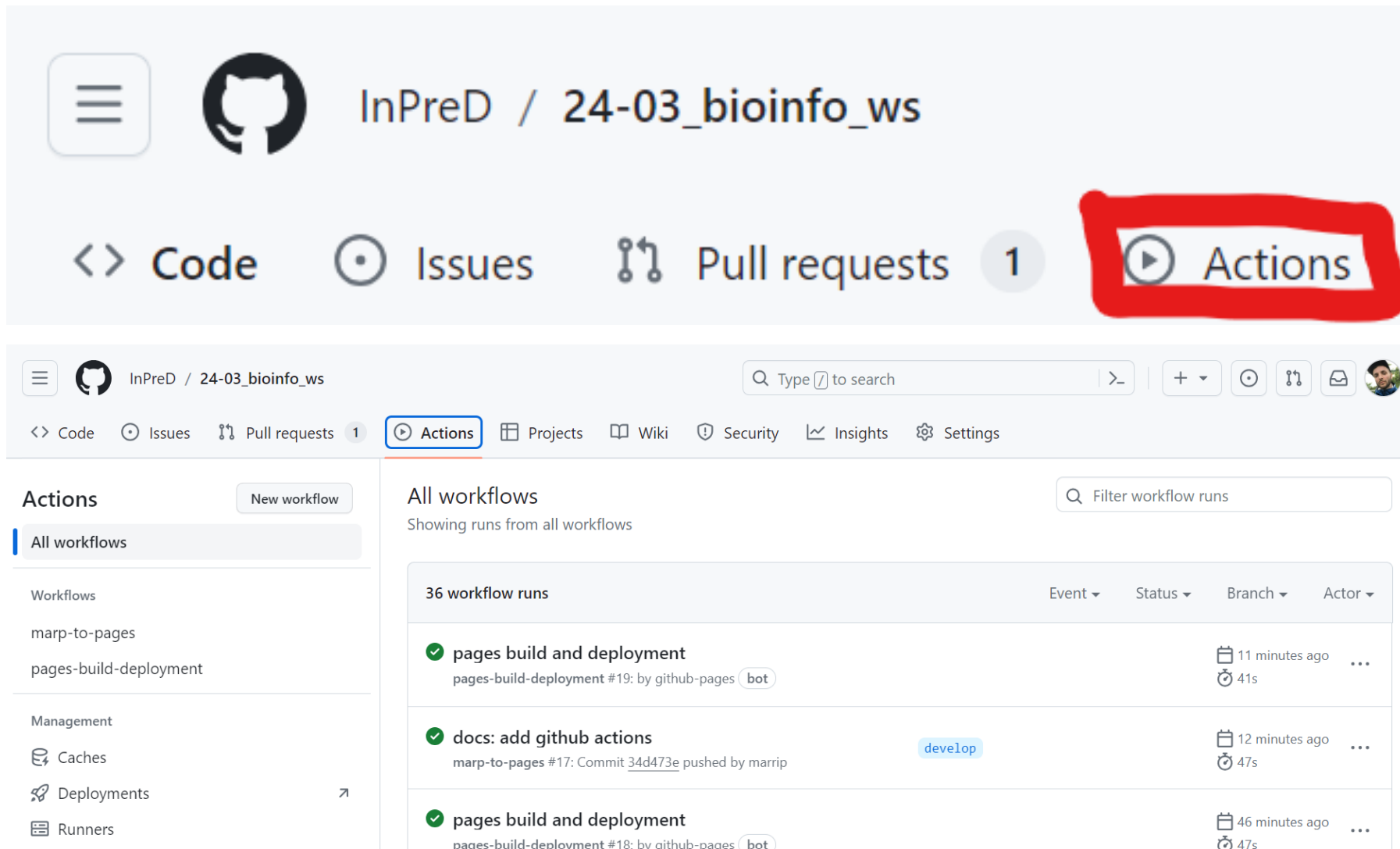
```
name: Docker Build
on:
  push:
    branches:
      - main
      - develop
    tags:
      - '!*.*.*'

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
    ...
```

2.3. GitHub Actions

```
...
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 }
```

2.3. GitHub Actions



The screenshot shows the GitHub repository interface for 'InPreD / 24-03_bioinfo_ws'. The 'Actions' tab is highlighted with a red box. Below the repository name, the navigation bar includes 'Code', 'Issues', 'Pull requests' (with a count of 1), and 'Actions' (highlighted). The main content area shows the 'All workflows' section with a search bar and a table of workflow runs.

Actions New workflow

All workflows

Workflows

- marp-to-pages
- pages-build-deployment

Management

- Caches
- Deployments
- Runners

All workflows Filter workflow runs

Showing runs from all workflows

36 workflow runs Event Status Branch Actor

✓ 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	12 minutes ago 47s	...
✓ pages build and deployment pages-build-deployment #18: by github-pages (bot)	46 minutes ago 47s	...

2.3. GitHub Actions

InPreD / 24-03_bioinfo_ws

Q Type / to search

+ ▾

<> Code

Issues

Pull requests 1

Actions

Projects

Wiki

Security

Insights

Settings

← marp-to-pages

docs: add images for github actions #18

Cancel workflow

...

Summary

Jobs

build

Run details

Usage

Workflow file

Triggered via push now

Status

Total duration

Artifacts

marrip pushed 698fa73 develop In progress - -

main.yml

on: push

build 16s

-

+

2.3. GitHub Actions

InPreD / 24-03_bioinfo_ws

Q

Type / to search

>_

+

▼

<>

Code

⌚

Issues

🔗

Pull requests

1

▶

Actions

📁

Projects

📖

Wiki

🛡️

Security

📈

Insights

⚙️

Settings

← marp-to-pages

✓ docs: add images for github actions #18

Re-run all jobs

⋮

🏠 Summary

Jobs

✓ build

Run details

🕒 Usage

📄 Workflow file

build

succeeded now in 37s

Beta

Give feedback

Q Search logs

↺

⚙️

> ✓ Set up job1s

> ✓ Pull marpteam/marp-cli:v3.0.213s

> ✓ Checkout code0s

> ✓ Ensure build dir exists0s

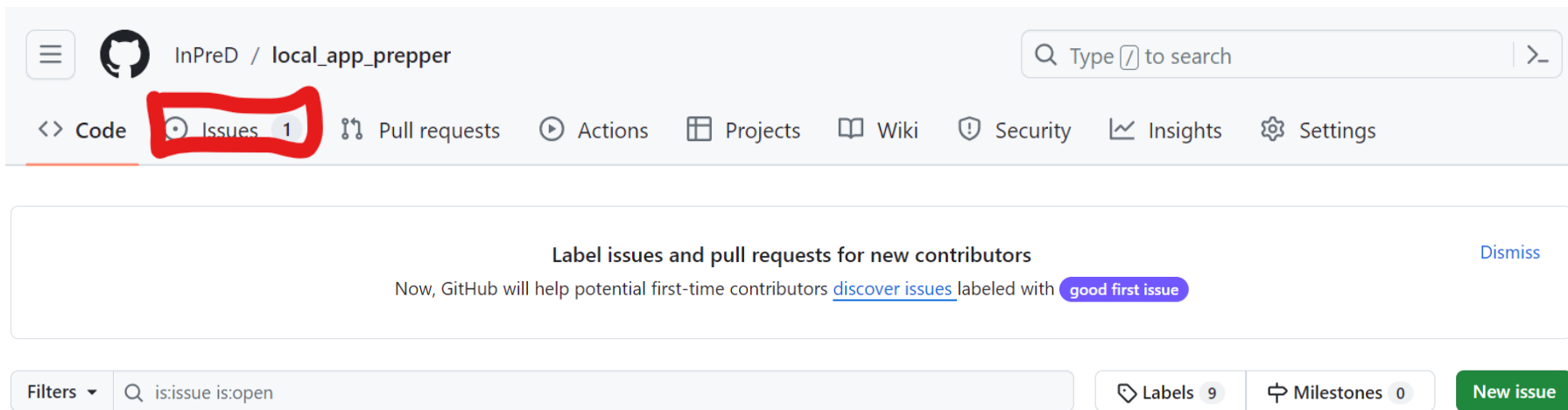
> ✓ Copy images directory (if exists)0s

> ✓ Marp Build (README)2s

> ✓ Marp Build (README.pdf)3s

2.4. GitHub workflow

- go to issues and create a **New issue**



2.4. GitHub workflow

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



Add a title

new fancy feature

Add a description

Write

Preview

H B I ≡ <> 🔗 | ☰ ☷ ☶ | 📎 @ ↻ ↩

Add your description here...

📄 Markdown is supported

🖼️ Paste, drop, or click to add files

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](#)

Submit new issue

2.4. GitHub workflow

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

The screenshot displays a GitHub issue interface. At the top, a comment by user 'marrip' is shown with the text 'No description provided.' Below this, a notification indicates 'marrip self-assigned this now'. The main section is titled 'Add a comment' and features a text input area with a rich text editor toolbar. The toolbar includes buttons for bold (B), italic (I), code (<>), link, and other formatting options. A red box highlights the 'Create a branch' link in the 'Development' section on the right. The right sidebar contains sections for 'Assignees' (listing 'marrip'), 'Labels' (None yet), 'Projects' (None yet), 'Milestone' (No milestone), and 'Development' (with the highlighted 'Create a branch' link). At the bottom, there are buttons for 'Close issue' and 'Comment', along with a notification that says 'You're receiving notifications because you were assigned.'

marrip commented now

Member

No description provided.

marrip self-assigned this now

Add a comment

Write Preview

H B I

Add your comment here...

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

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.

Close issue

Comment

2.4. 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 ▾

What's next?
☐ Open in codespace
☒ Checkout locally
☐ Open branch with GitHub Desktop

Beta [Share feedback](#) Create branch

change branch source

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

Beta [Share feedback](#) Create branch

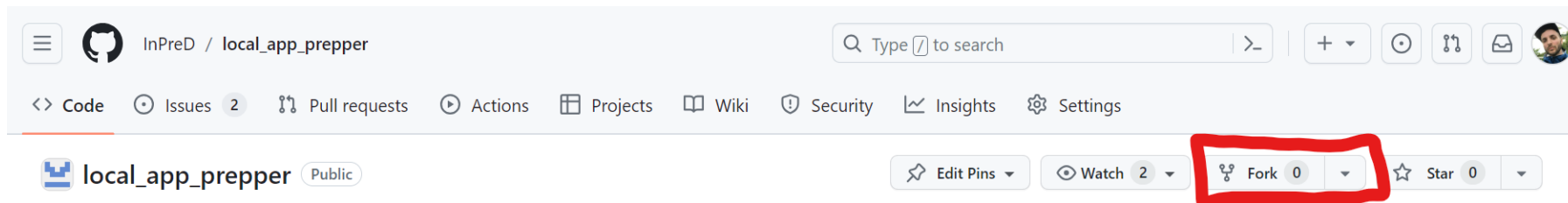
2.4. GitHub workflow

- 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
```

2.4. GitHub workflow

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




2.4. 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 * Repository name *


 marrip / 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

 Copy the **main** branch only

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

ⓘ You are creating a fork in your personal account.

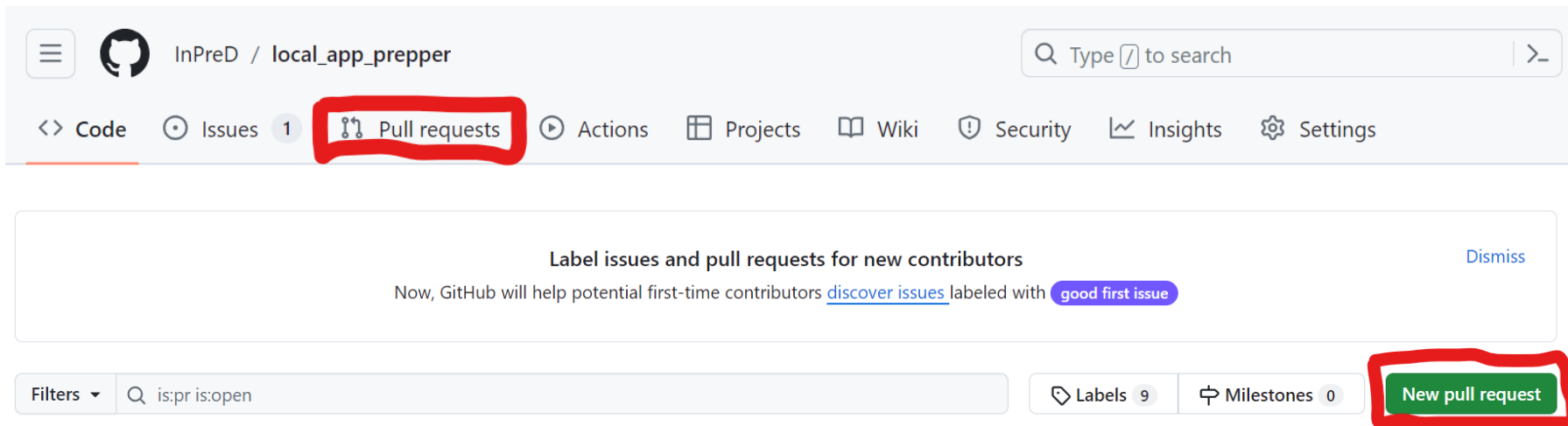
Create fork

2.4. 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

2.4. GitHub workflow

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

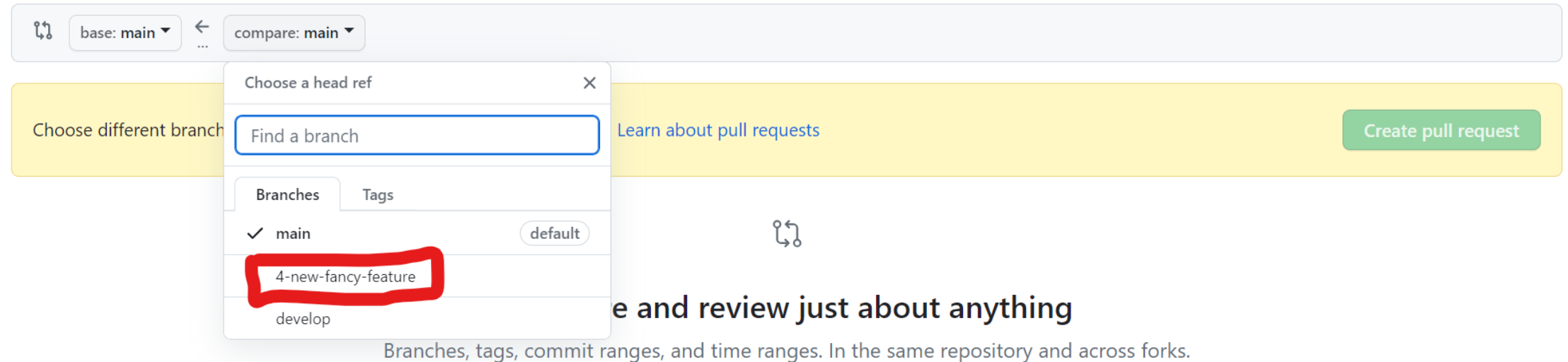


2.4. GitHub workflow

- choose `develop` as `base` and your new feature branch (same repo or forked) 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 different branch

Find a branch

Learn about pull requests

Create pull request

Choose a head ref

Find a branch

Branches Tags

✓ main default

4-new-fancy-feature

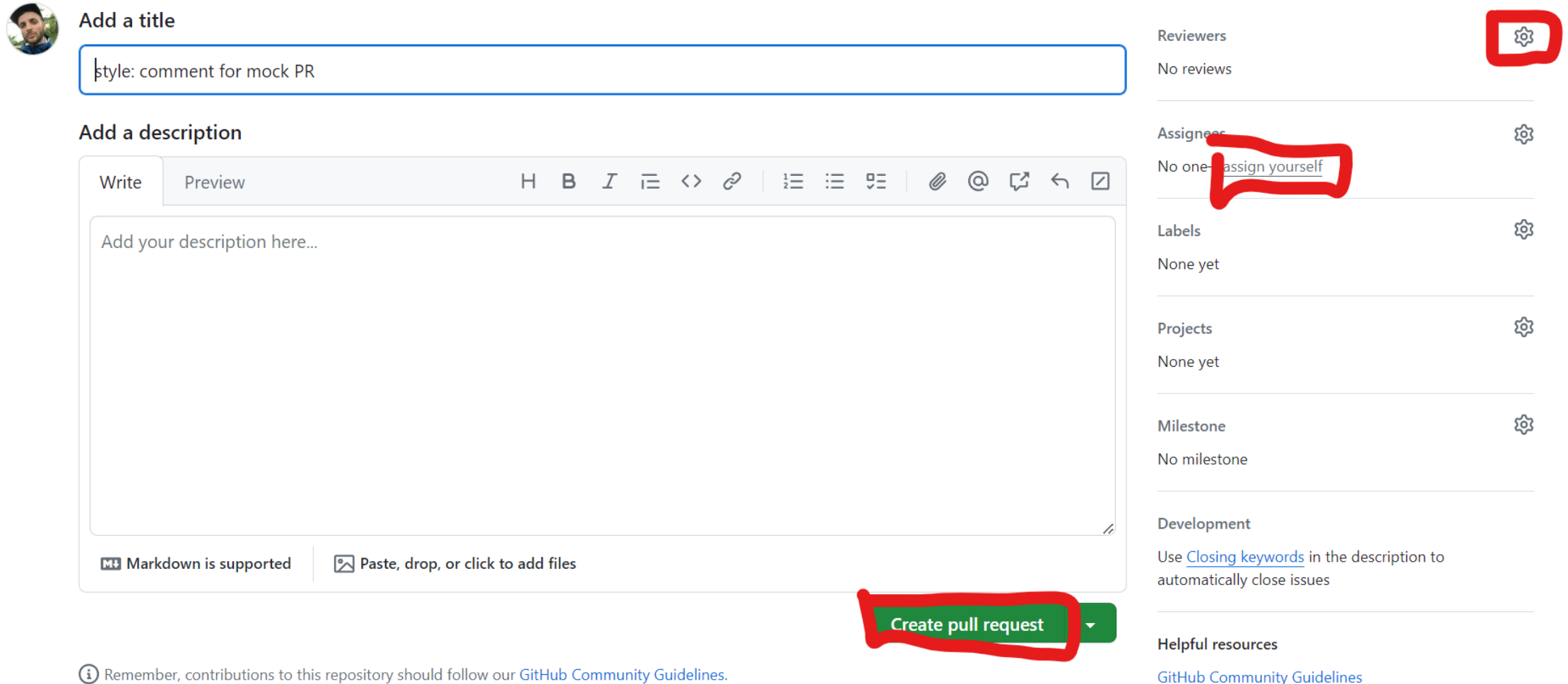
develop

e and review just about anything

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

2.4. GitHub workflow

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



The screenshot shows the GitHub 'Create pull request' form. A red box highlights the 'Add a title' section, which contains the text 'style: comment for mock PR'. Another red box highlights the 'Add a description' section, which is empty. A third red box highlights the 'Assignees' section, where 'No one' is selected and 'assign yourself' is highlighted. A fourth red box highlights the 'Reviewers' section, where a cog icon is highlighted. A fifth red box highlights the 'Create pull request' button at the bottom right. The right sidebar shows sections for 'Reviewers', 'Assignees', 'Labels', 'Projects', 'Milestone', 'Development', and 'Helpful resources'.

Add a title

style: comment for mock PR


Add a description

Write Preview


H B I

Add your description here...


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

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](#)

Create pull request

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

2.4. 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

Edit <> Code



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

Conversation 0

Commits 1

Checks 0

Files changed 1

+3 -1



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

2.4. 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


Edit <> Code

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

 Conversation 0

 Commits 1

 Checks 0

 **Files changed 1**

+3 -1



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

2.4. GitHub workflow


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


style: comment for mock PR #5

Edit


<> Code ▾

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

 Conversation 0

 Commits 1

 Checks 0

 Files changed 1



+3 -1 

Changes from all commits ▾ File filter ▾ Conversations ▾ Jump to ▾ 

0 / 1 files viewed

Review in codespace

Review changes ▾

4  local_app_prepper.py 

☐ Viewed

 ...

... @@ -1,6 +1,8 @@

1 1 #!/usr/local/bin/python

2



3 + just a comment

4 +

3 5 from prepper.cli import main

4 6

5 7 if __name__ == "__main__":

6 - main()



8 + main()
















2.4. GitHub workflow

- you can type your comment


3 + # just a comment


Write

Preview

 |       |    |     

Leave a comment

 Markdown is supported

 Paste, drop, or click to add files

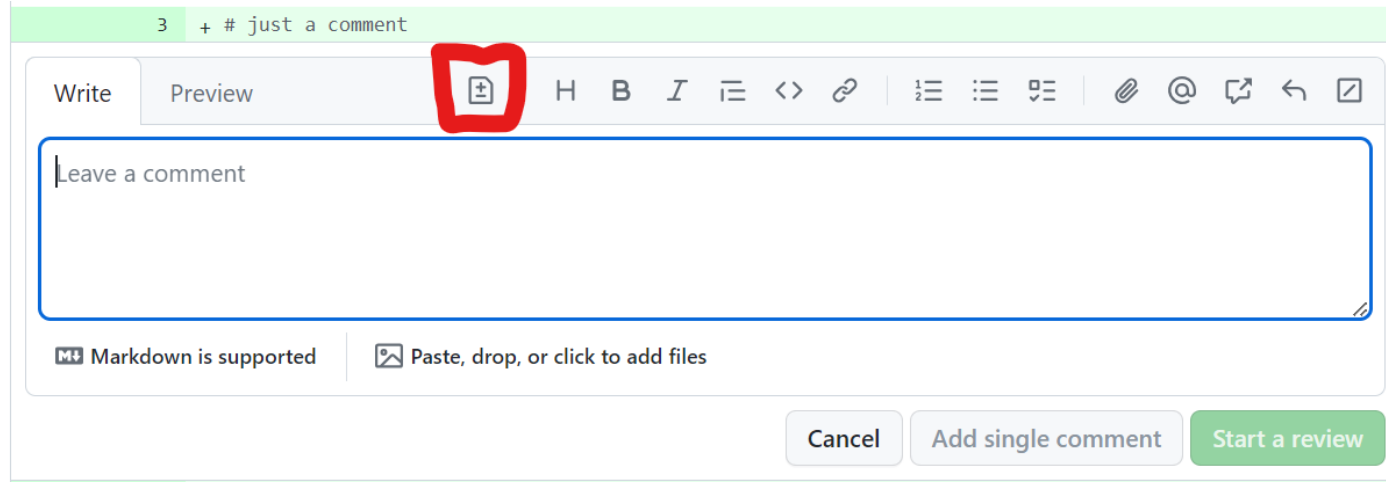
Cancel

Add single comment












Start a review

2.4. GitHub workflow



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



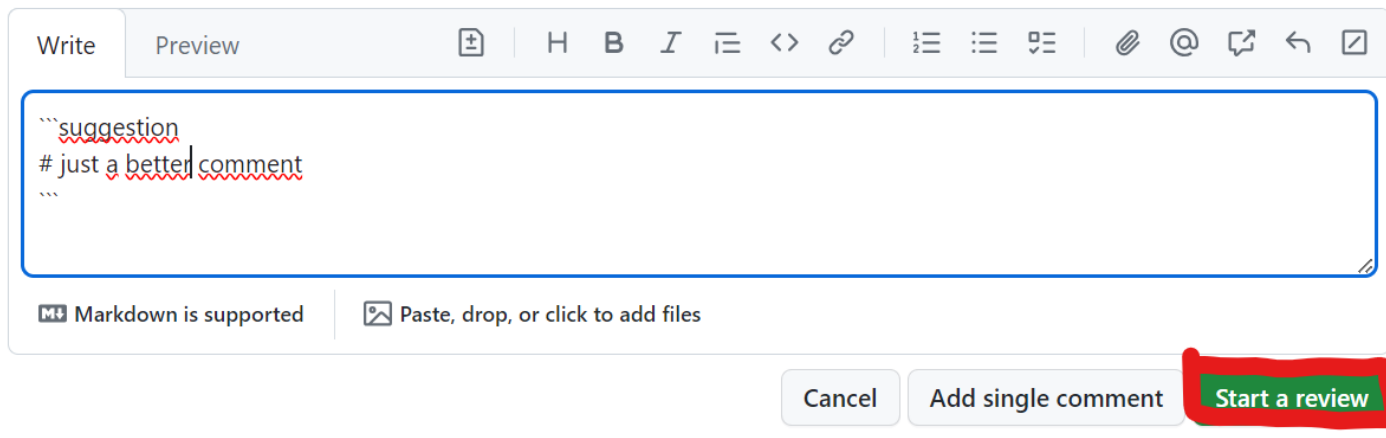
3 + # just a comment












Write Preview  H B I        @   

Leave a comment



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

Cancel Add single comment **Start a review**



Write Preview  H B I        @   

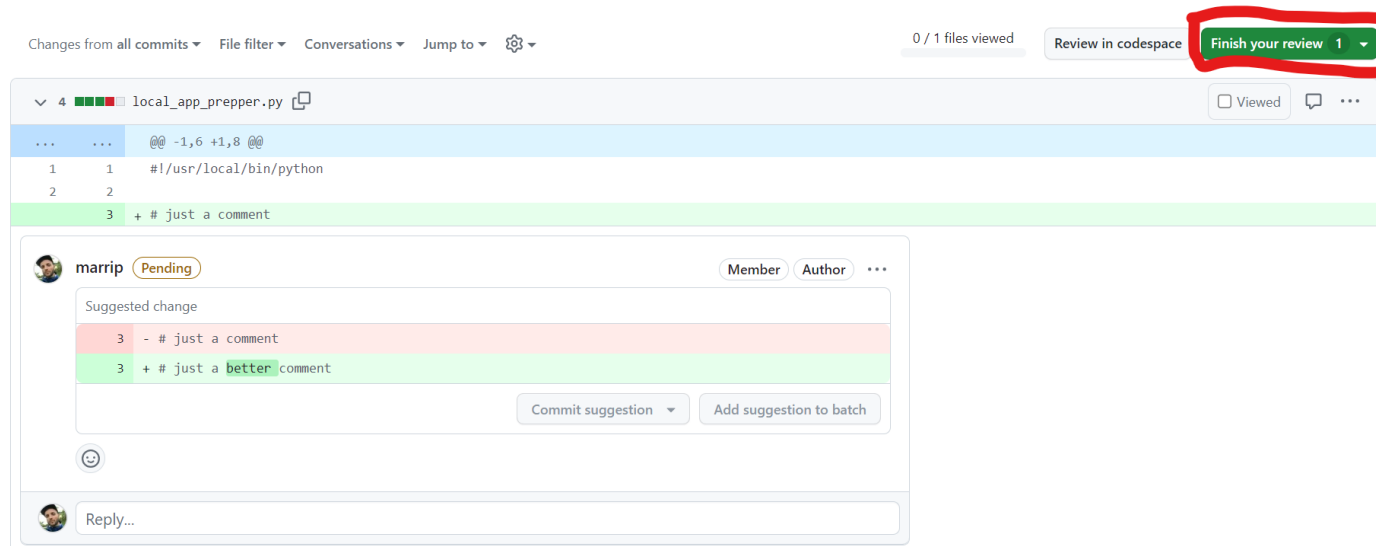
```  
suggestion  
# just a better comment  
```

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

Cancel Add single comment **Start a review**

2.4. GitHub workflow

- when you are done with reviewing, Finish your review



2.4. GitHub workflow

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

Finish your review



Write

Preview

H

B

I

≡

<>

🔗

≡

≡

≡

📎

@

↗

↶

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.

Submit review

2.4. GitHub workflow

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



Add a comment

Write

Preview

H B I ≡ <> 🔗 | ≡ ≡ ≡ | 📎 @ ↗ ↶ 🗑

LGTM 👍 |

Markdown is supported

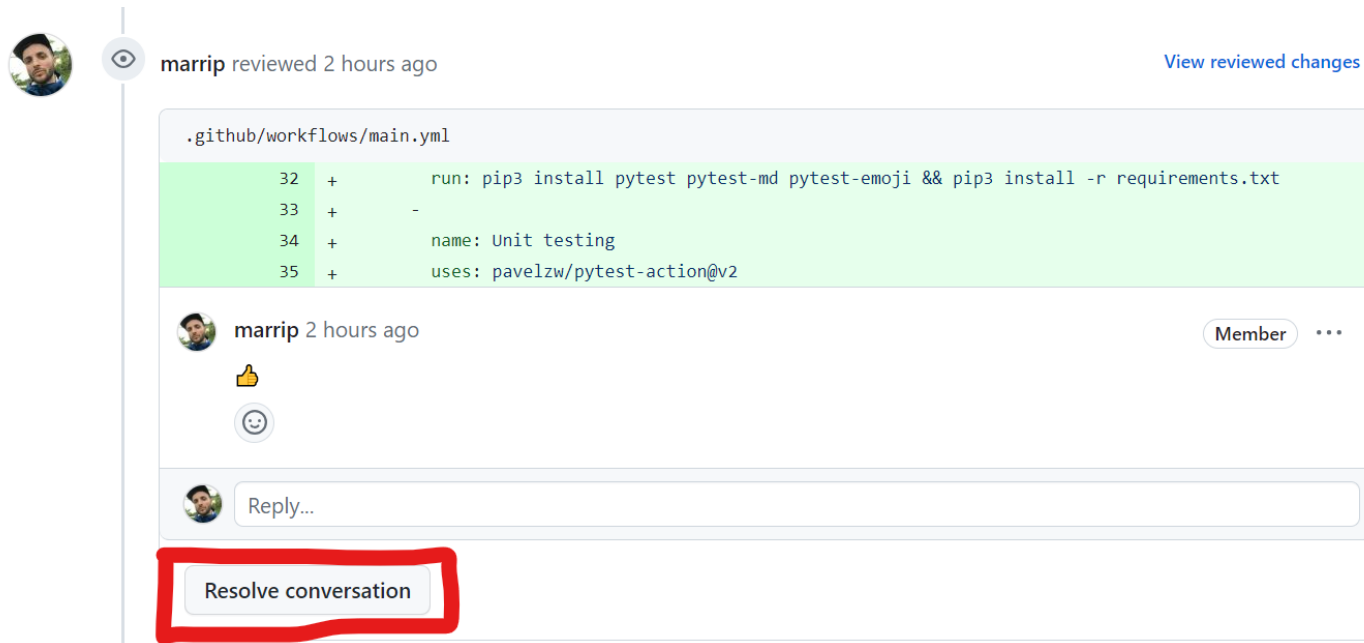
Paste, drop, or click to add files

Close with comment

Comment

2.4. 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



The screenshot shows a GitHub pull request review interface. At the top, a user profile icon for 'marrip' is shown next to the text 'marrip reviewed 2 hours ago'. To the right of this is a link 'View reviewed changes'. Below this is a code diff for the file '.github/workflows/main.yml'. The diff shows lines 32 to 35, all with '+' signs, indicating additions. The code is: line 32: 'run: pip3 install pytest pytest-md pytest-emoji && pip3 install -r requirements.txt', line 33: a hyphen '-', line 34: 'name: Unit testing', and line 35: 'uses: pavelzw/pytest-action@v2'. Below the code diff is a comment by 'marrip' from '2 hours ago'. The comment contains a thumbs-up emoji and a smiley face emoji. To the right of the comment is a 'Member' badge and a three-dot menu. Below the comment is a 'Reply...' input field. At the bottom of the comment box, a 'Resolve conversation' button is highlighted with a red rectangular border.

marrip reviewed 2 hours ago [View reviewed changes](#)

```
.github/workflows/main.yml
32 + run: pip3 install pytest pytest-md pytest-emoji && pip3 install -r requirements.txt
33 + -
34 + name: Unit testing
35 + uses: pavelzw/pytest-action@v2
```

marrip 2 hours ago Member ...

👍 😊

Reply...


Resolve conversation


2.4. GitHub workflow

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



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.

Merge pull request — You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Add rule ×

2.4.1 Hands-on pull request

- go to https://github.com/InPreD/24-03_bioinfo_ws/
- create fork to your own account
- open an issue "test pull request" or similar and create a branch
- go to the branch and add a markdown file with your first name and favorite emoji to the `participants` folder, ideally the file is named `<your first name>.md`
- open a pull request in the original repository and add someone else in the group to review your pull request
- review someone else's pull request, give feedback and approve if correct

2.5. 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

2.5. Release

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

The screenshot shows the GitHub repository page for `local_app_prepper`. The repository is public and has 0 stars, 0 forks, and 2 watchers. The main branch is selected, and there are 3 branches and 1 tag. The file list shows the following files and their commit history:

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


On the right side of the repository page, under the 'About' section, there is a link to 'Create a new release' which is highlighted with a red box.


2.5. 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

H B I ≡ <>      @  ↩

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

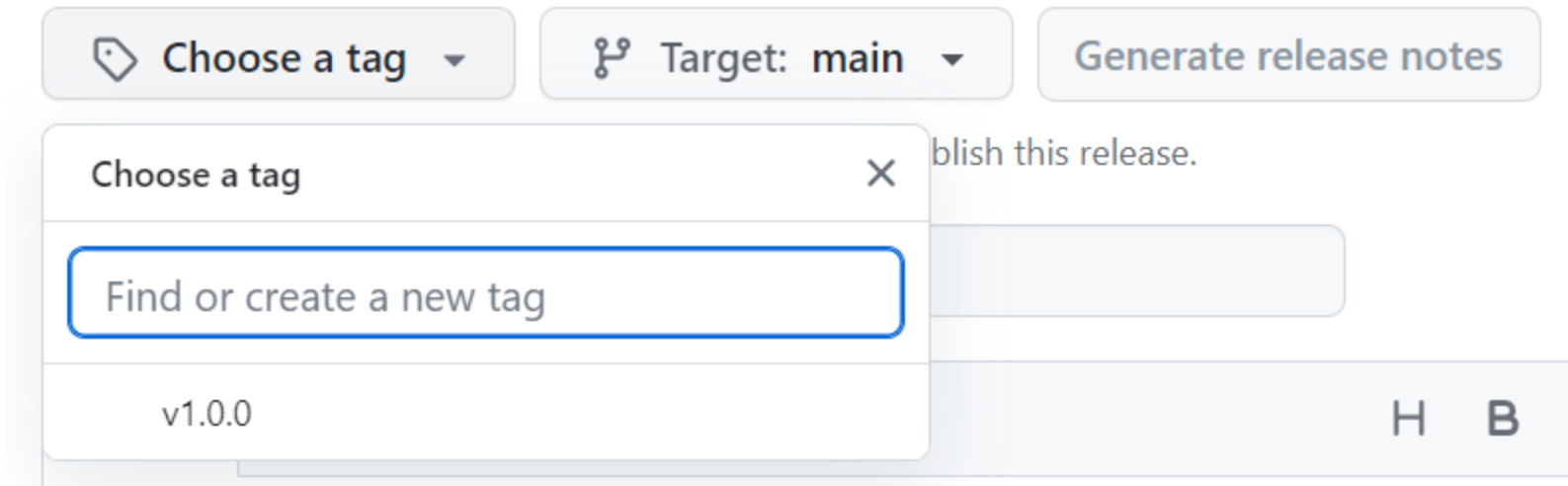
If you're new to releasing software, we highly recommend to [learn more about semantic versioning](#).

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](#).

2.5. Release

- ideally, you choose a tag according to semantic versioning



The image shows a user interface for creating a release. At the top, there are three buttons: 'Choose a tag' (with a tag icon), 'Target: main' (with a branch icon), and 'Generate release notes'. Below the 'Choose a tag' button, a dropdown menu is open. The menu has a title 'Choose a tag' and a close button 'X'. Inside the menu, there is a text input field with the placeholder 'Find or create a new tag'. Below the input field, the tag 'v1.0.0' is listed. To the right of the dropdown, the text 'Publish this release.' is visible. At the bottom right, there are two buttons labeled 'H' and 'B'.

2.5.1. 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

2.5. Release

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

Write

Preview

H B I ≡ < > 🔗 ≡ ≡ ≡ ⌨ @ 🗨 ↩

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

2.6. 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.

2.7. Resources

- [About Git](#)
- [Gitflow workflow](#)
- [GitHub Actions](#)
- [Semantic versioning](#)
- [Licensing](#)

3. Nextflow

3.1. 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
- can be used with an array of executors, such as SLURM, k8s, AWS, Azure, Google Cloud and many more
- **nf-core**: project/community that develops framework for nextflow including guidelines, tools, modules, subworkflows, pipelines and test data

3.2. Requirements

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

3.3. 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>

3.4. Best practice: nf-core template

```
workflow_repo
├── LICENSE
├── README.md
├── assets
│   ├── mock.genome.fasta
│   ├── samplesheet.csv
│   └── schema_input.json
├── bin
│   └── script.py
├── conf
│   ├── base.config
│   ├── modules.config
│   └── test_stub.config
├── lib
│   ├── NfcoreSchema.groovy
│   ├── NfcoreTemplate.groovy
│   ├── WorkflowMain.groovy
│   └── nfcore_external_java_deps.jar
├── main.nf
├── modules
│   ├── local
│   │   ├── module_1.nf
│   │   └── module_2.nf
│   └── nf-core
│       ├── module_1
│       │   └── arg_1
│       │       ├── main.nf
│       │       └── meta.yml
│       └── custom
│           └── dumpsoftwareversions
│               ├── main.nf
│               ├── meta.yml
│               └── templates
│                   └── dumpsoftwareversions.py
├── modules.json
├── nextflow.config
├── nextflow_schema.json
├── workflows
│   └── main.nf
```


3.6 Resources

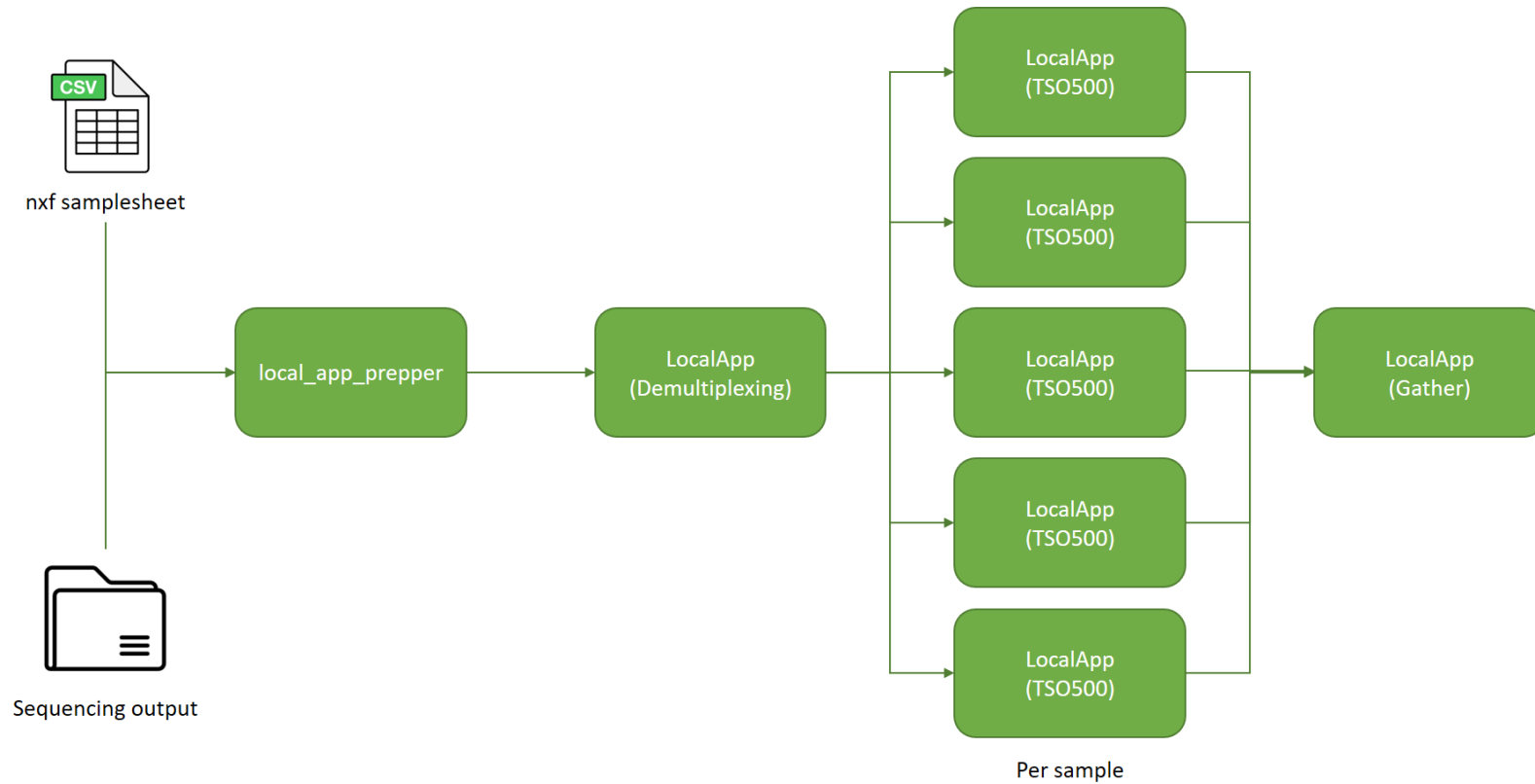
- nextflow.io
- nf-co.re
- [nf-core github](https://github.com/nf-core)

4. `tso500_nxf_workflow`

4.1. 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

4.2. Overview



4.3. Demonstration

4.4. Outlook

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

4.5. Resources

- [repository](#)
- [local_app_prepper](#)
- [samplesheet_generator](#)

5. Python project

5.1. Repository structure

- consistency/standard
- keep main script short and sweet - functionality in modules

```
#!/usr/local/bin/python

from my_module import main

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

5.1. Repository structure

- module folder should contain `__init__.py`
- keep functions short and try to refactor big functions
- leave descriptive comments in code
- use libraries to make your life easier
 - `pandas` : csv/tsv files
 - `click` or `argparse` : define cli input flags
- introduce proper exception handling
- logging with log levels

5.1.1. Unit testing

- `pytest` for testing
- include unit tests for functions, preferable table-driven

```
def addition(x, y):  
    return x+y
```

```
import pytest  
  
@pytest.mark.parametrize("x, y, z", [(1, 1, 2), (1, -1, 0)])  
def test_eval(x, y, z):  
    assert addition(x, y) == z
```

```
$ pytest
```

5.1. Repository structure

- include test data for unit testing if necessary
- create container image from project, preferably docker
- include all necessary dependencies in `requirements.txt` (locked versions)
- add GitHub actions for testing, linting, building, etc.
- preferable include a devcontainer definition
- `README.md` and other `docs`

5.1. Repository structure

```
/repo
|-- .devcontainer
|   |-- devcontainer.json
|-- .github
|   |-- workflows
|       |-- main.yml
|-- .gitignore
|-- Dockerfile
|-- docs
|-- LICENSE
|-- README.md
|-- my_tool.py
|-- my_module
|   |-- __init__.py
|   |-- my_module.py
|   |-- tests
|       |-- __init__.py
|       |-- my_module_test.py
|-- requirements.txt
|-- test
```

5.2. Resources

- [pandas](#)
- [click](#)
- [argparse](#)
- [exception handling](#)
- [logging](#)
- [pytest unittesting](#)
- [inprod dockerhub](#)
- [devcontainers](#)