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

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
```

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

Create a GitHub account

- choose email preferences

Create a GitHub account

- solve the puzzle

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

Open GitHub

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

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.
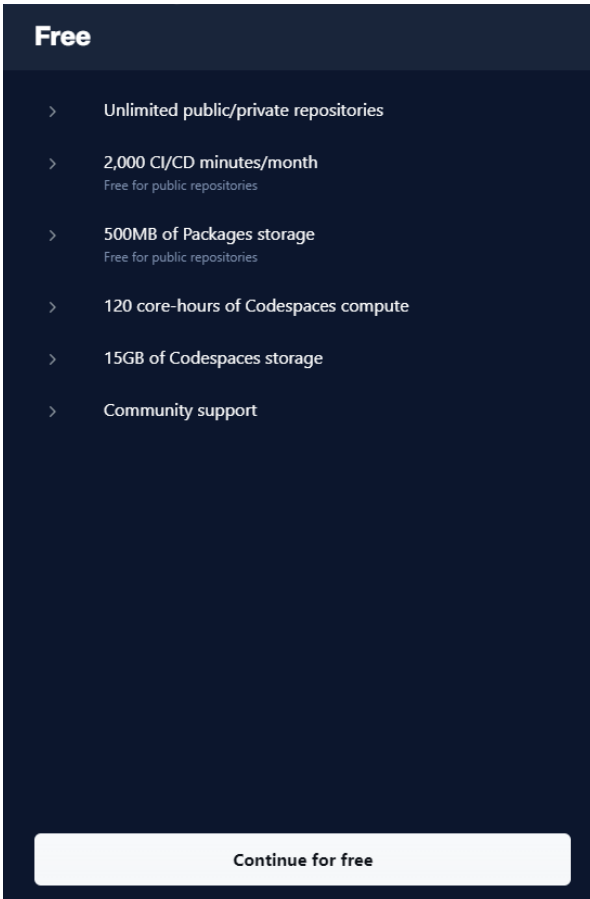
- ☑ Security

  Private repos, 2FA, Required reviews, Required status checks, Code scanning, Secret scanning, Dependency graph, Dependabot alerts, and more.

Create a GitHub account

- choose the free plan

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
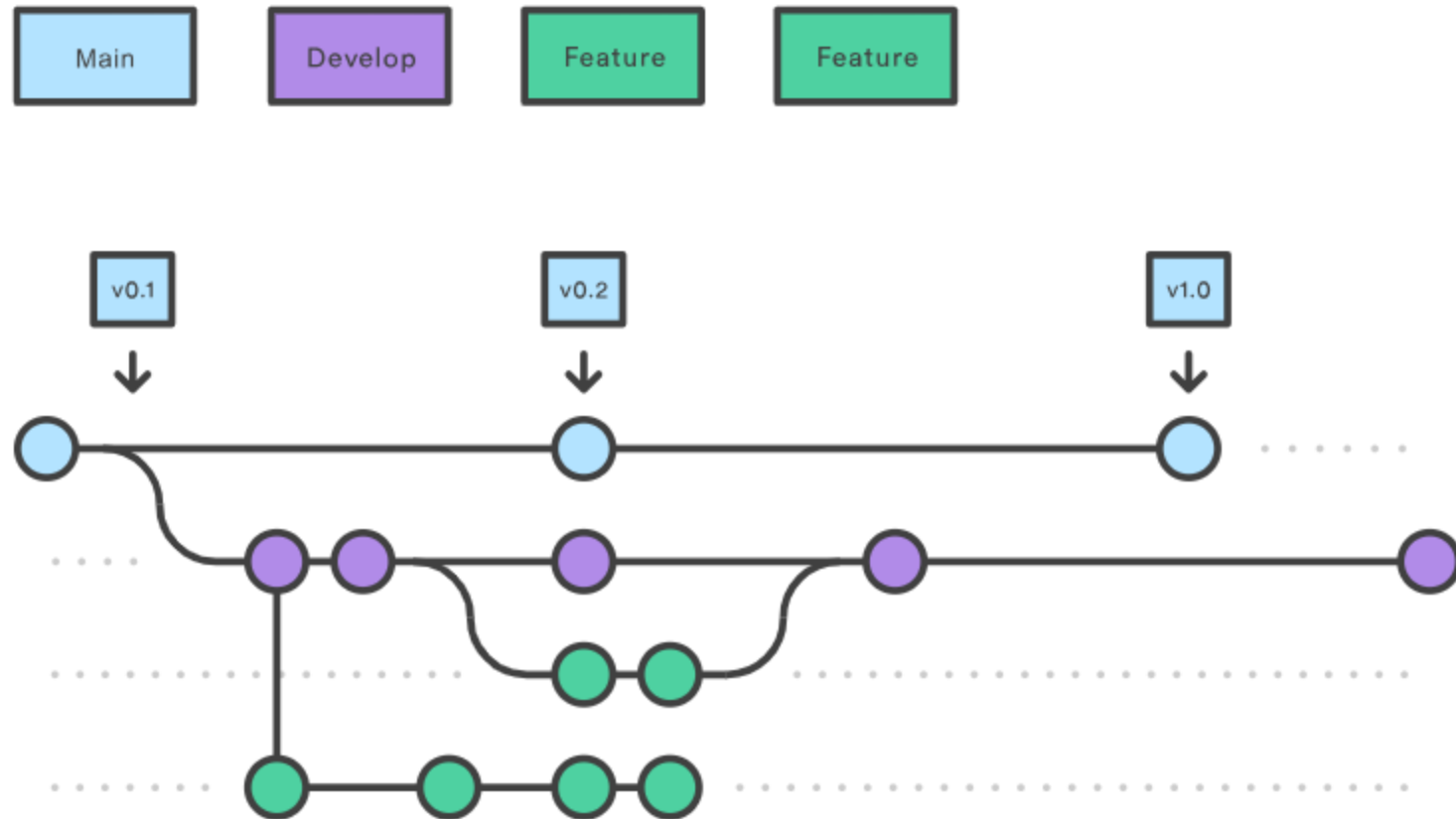
2. Development & Collaboration

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

GitHub Actions

GitHub Actions

2. Development & Collaboration

GitHub workflow

- go to issues and create a `New issue`
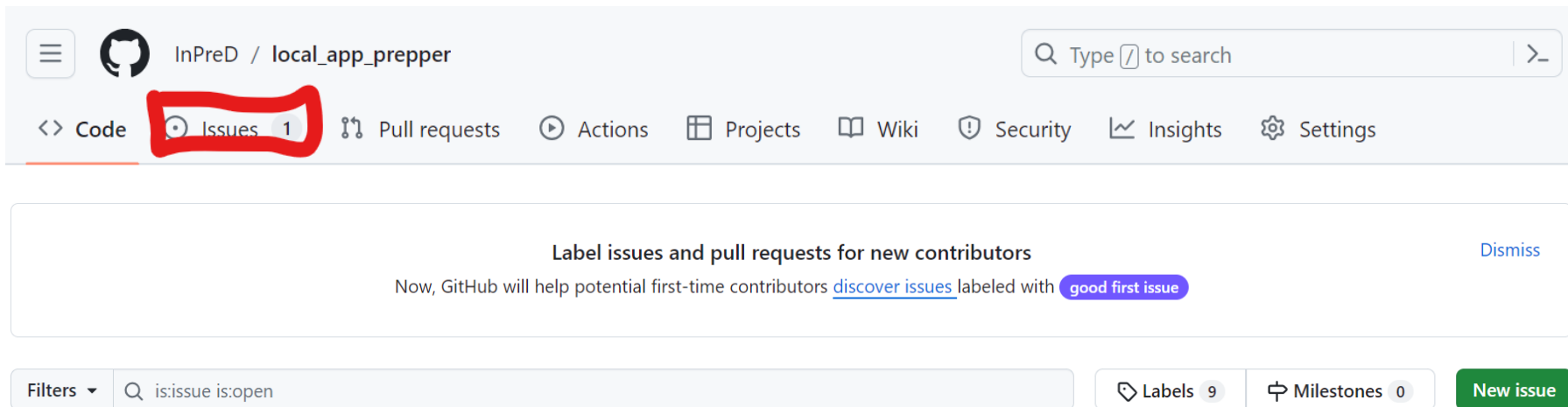
GitHub workflow

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

GitHub workflow

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

GitHub workflow

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

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
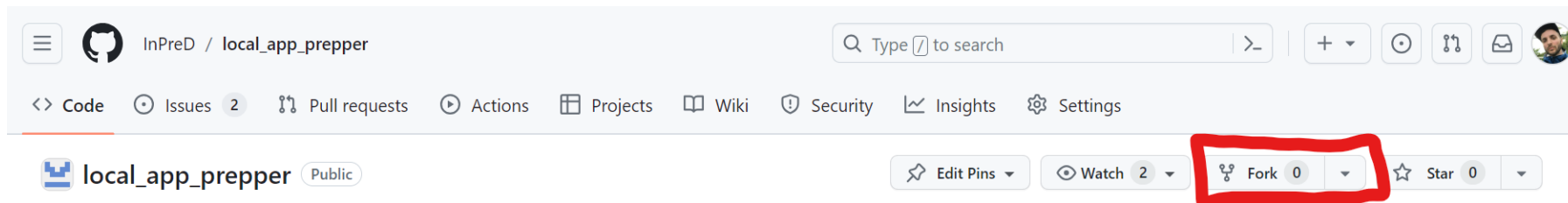
GitHub workflow

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 *   **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

☐ 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.

Create fork

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`
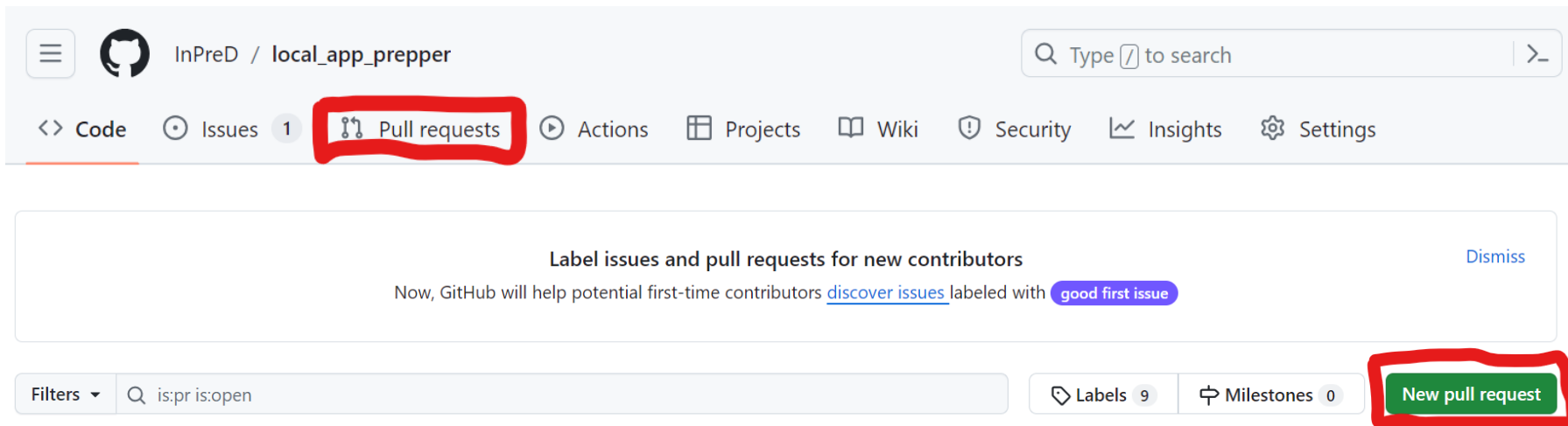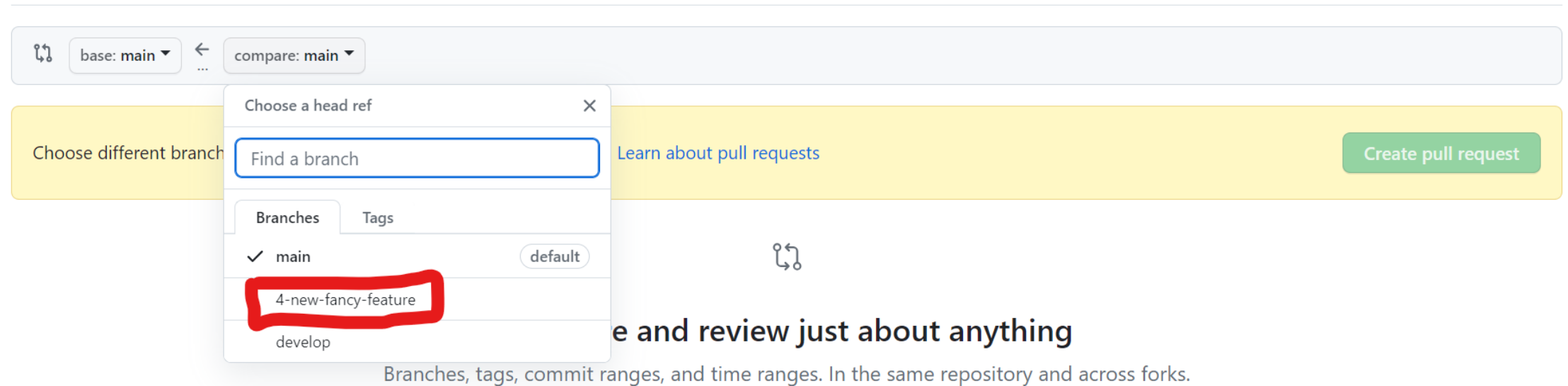
GitHub workflow

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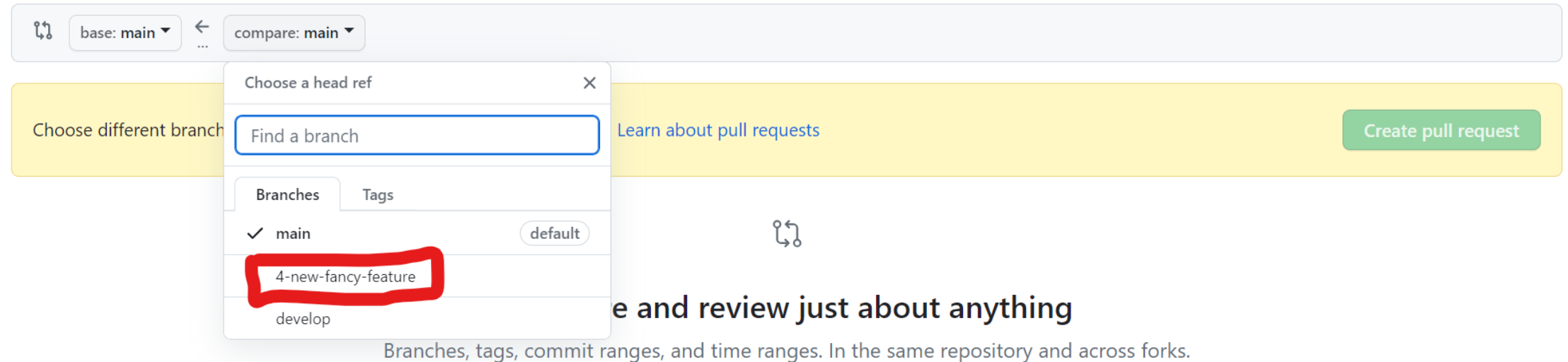
GitHub workflow

- `assign yourself`, add at least one reviewer, provide some context and `Create pull request`
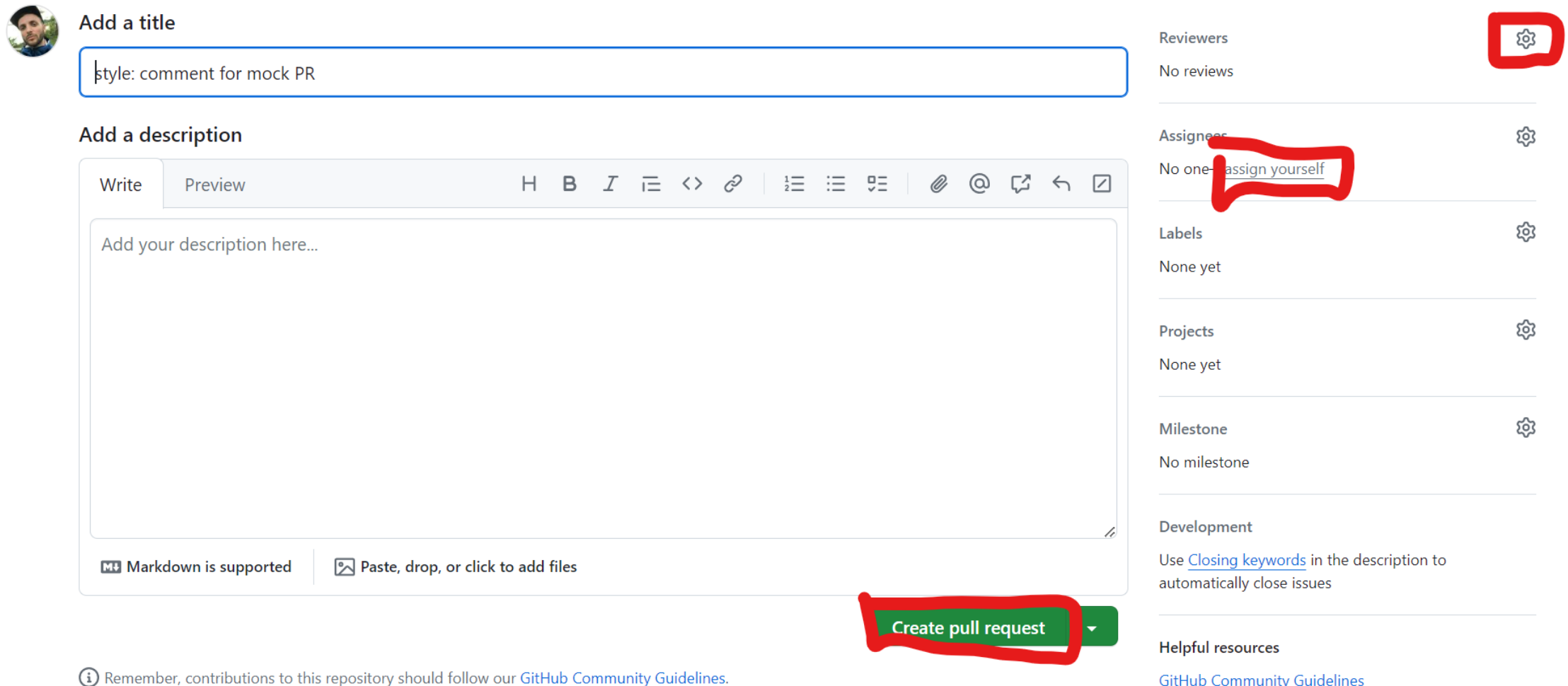
GitHub workflow

- 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

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

- release and semantic versioning

- licensing

2. Resources

- About Git
- Gitflow workflow
- GitHub Actions

3. Nextflow

- general (install, best practice)

- nf-core template

- stubbing

4. `tso500_nxf_workflow`

- status
- demonstration

5. Python

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

Resources

- github actions
- nf-core
- pytest unittesting
- semantic versioning