Contribution Workflow

Boyu Wang

2025.12.14

CONTRIBUTING.md

Contributing

For candidates interested in participating in the Google Summer of Code (GSoC), checkout Mesa's GSoC guide.

As an open source project, Mesa welcomes contributions of many forms, and from beginners to experts. If you are curious or just want to see what is happening, we post our development session agendas and development session notes on Mesa discussions. We also have a threaded discussion forum on Matrix for casual conversation.

In no particular order, examples include:

- Code patches
- Bug reports and patch reviews
- New features
- Documentation improvements
- Tutorials

No contribution is too small. Although, contributions can be too big, so let's discuss via Matrix OR via an issue.

To submit a contribution

- Create a ticket for the item that you are working on.
- Fork the Mesa repository.
- Clone your repository from Github to your machine.
- Create a new branch in your fork: `git checkout -b BRANCH_NAME`
- Run `git config pull.rebase true`. This prevents messy merge commits when updating your branch on top of Mesa main branch.
- Install an editable version with developer requirements locally: `pip install -e ".[dev]"`
- Edit the code. Save.
- Git add the new files and files with changes: `git add FILE_NAME`
- Git commit your changes with a meaningful message: `git commit -m "Fix issue X"`
- If implementing a new feature, include some documentation in docs folder.
- Make sure that your submission works with a few of the examples in the examples repository. If adding a new feature to mesa, please illustrate usage by implementing it in an example.
- Make sure that your submission passes the GH Actions build. See "Testing and Standards below" to be able to run these locally.

https://github.com/mesa/mesa/blob/main/CONTRIBUTING.md

Modeller and/or developer

I'm a modeller (but not an experienced developer)

You already know how to build Mesa models (if not skip below), and probably have found things Mesa can't do (elegantly). You want to improve that. Awesome!

First step is to install some proper tools, if you haven't already.

- A good IDE helps for code development, testing and formatting. PyCharm or VSCode for example.
- Dive into Git and GitHub. Watch some videos, this takes some time to click. GitHub Desktop is great.
- `https://github.dev/mesa/mesa` is great for small changes (to docs).

Learn the tools, talk to us about what you want to change, and open a small PR. Or update an example model (check open issues)!

https://github.com/mesa/mesa/blob/main/CONTRIBUTING.md

I'm a developer (but not a modeller)

Awesome! You have the basics of open-source software development (if not check above), but not much modelling experience.

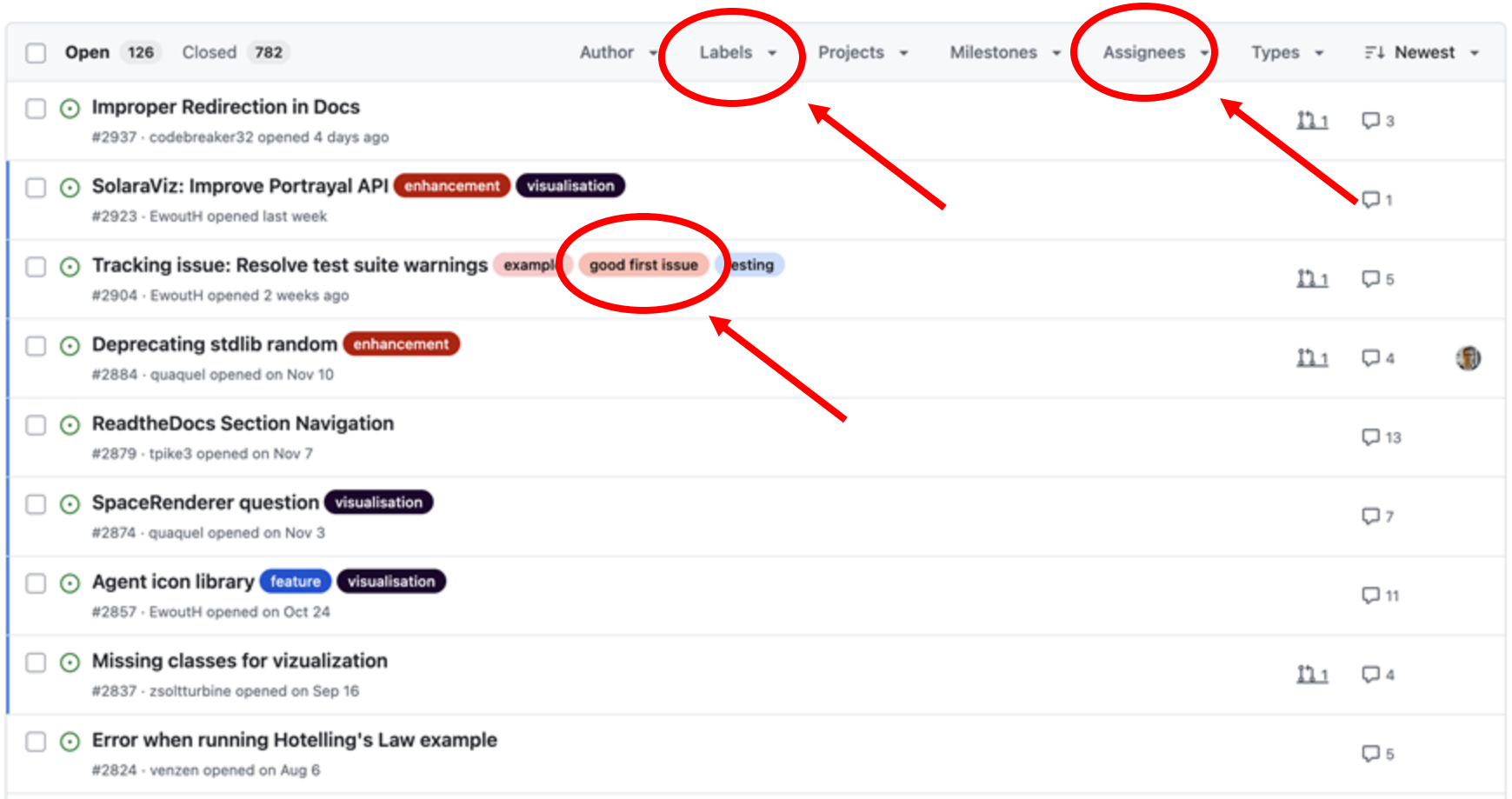
First step is to start thinking like a modeller. To understand the fine details about our library and contribute meaningfully, get some modelling experience:

- Go though our series of introductory tutorials at Getting Started. While going through them, dive into the source code to really see what everything does.
- Follow an ABM course (if possible). They might be a bit outdated programming language wise, but conceptual they're sound.
  - This MOOC on ABM concepts: Agent Based Modeling
  - This MOOC on practical ABM modelling: Agent-Based Models with Python: An Introduction to Mesa
- Go though multiple of our examples. Play with them, modify things and get a feel for Mesa and ABMs.
  - Check our open issues for the examples.
  - If you see anything you want to improve, feel free to open a (small) PR!
- If you have a feel for Mesa, check our discussions and issues.
  - Also go through our release notes to see what we recently have been working on, and see some examples of successful PRs.
- Once you found or thought of a nice idea, comment on the issue/discussion (or open a new one) and get to work!

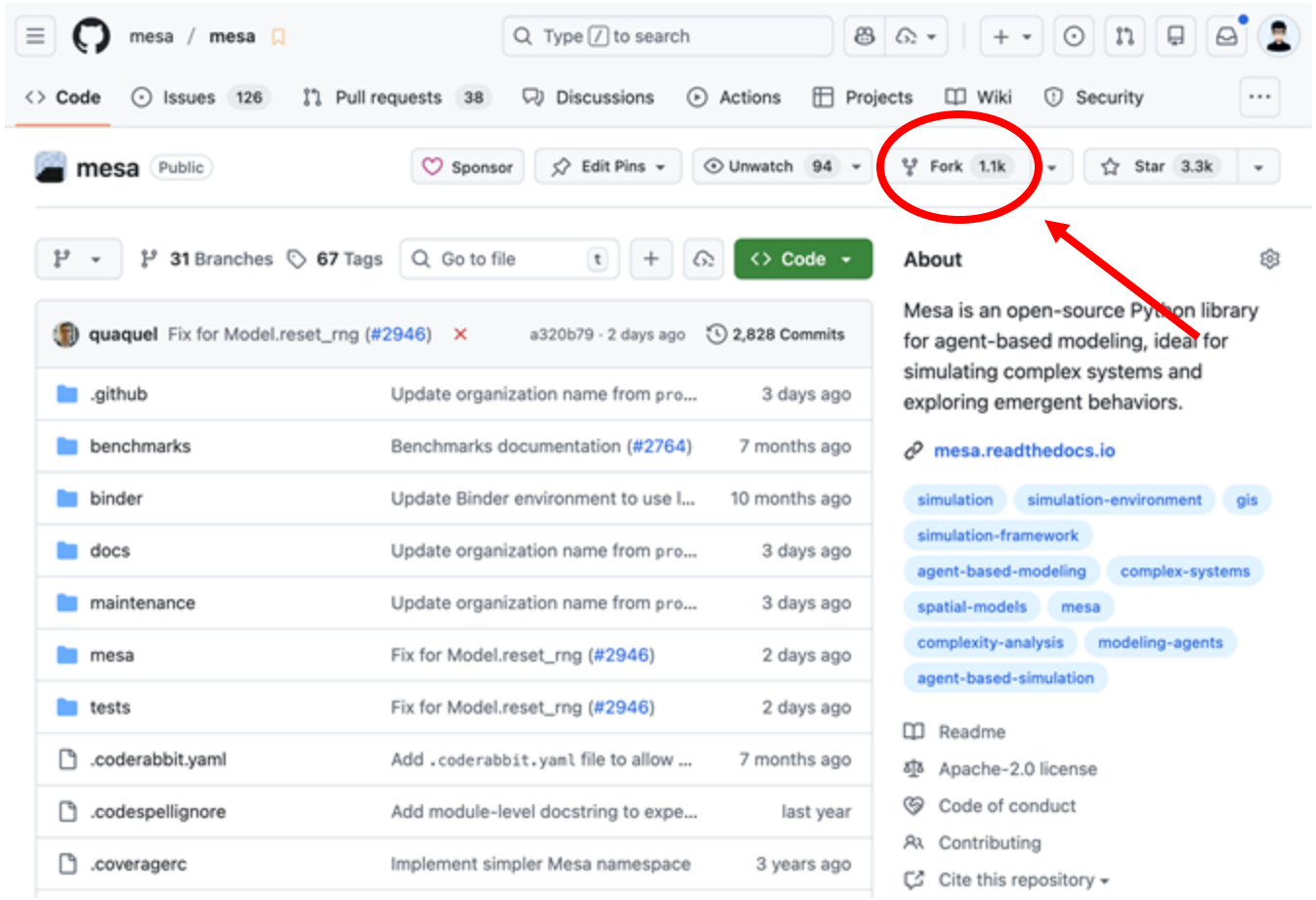
Part I: Setup

- Create a ticket for the item that you are working on.
- Fork the Mesa repository.
- Clone your repository from Github to your machine.
- Create a new branch in your fork: `git checkout -b BRANCH_NAME`
- Run `git config pull.rebase true`. This prevents messy merge commits when updating your branch on top of Mesa main branch.
- Install an editable version with developer requirements locally: `pip install -e ".[dev]"`

1.1 Create a ticket (or find one)

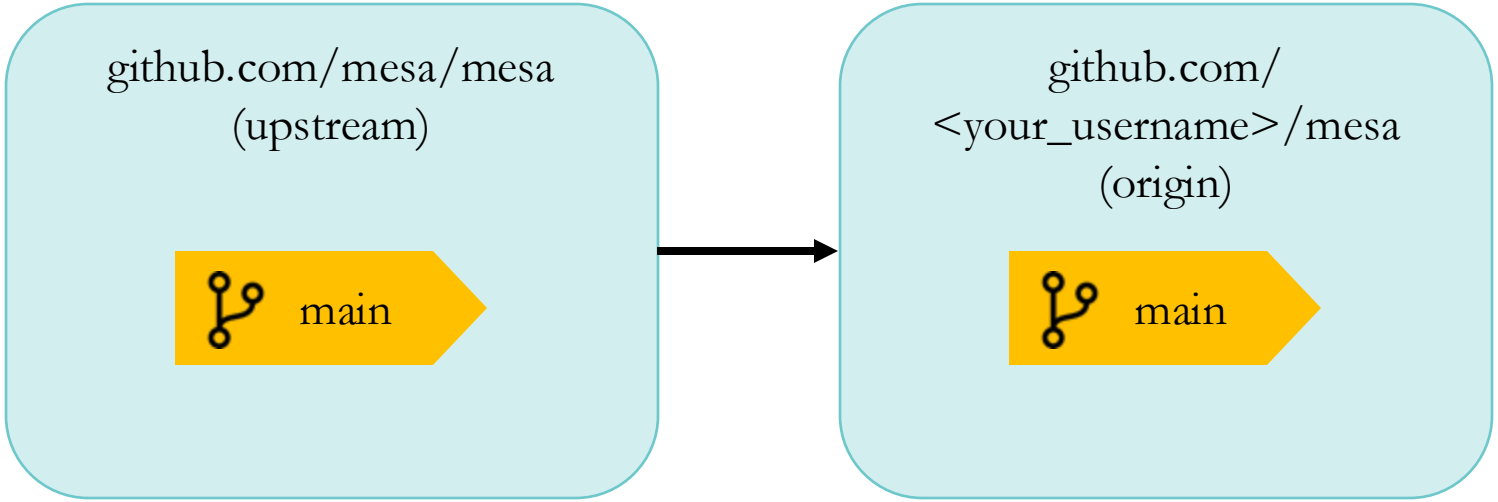


1.2 Fork the repository

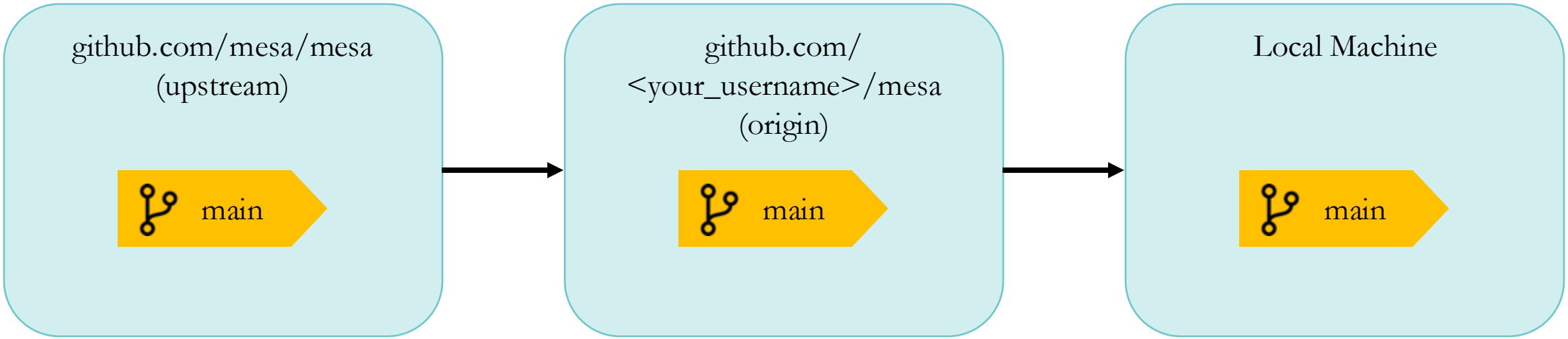


1.2 Fork the repository

After forking, there are now **two** GitHub repos: upstream (Mesa) and your fork.

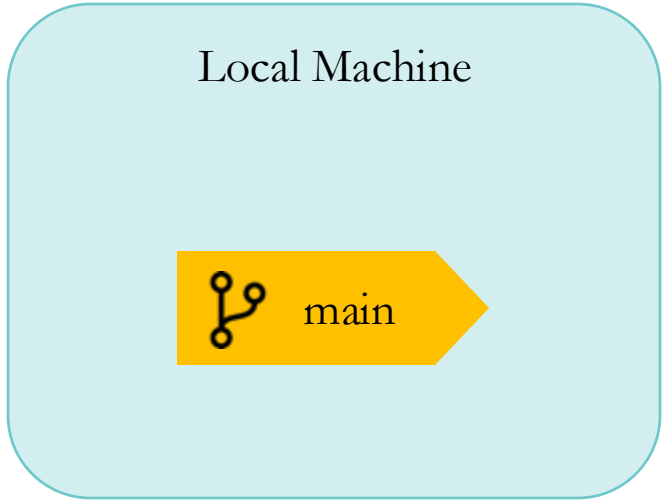


1.3 Clone to local machine



1.3 Clone to local machine

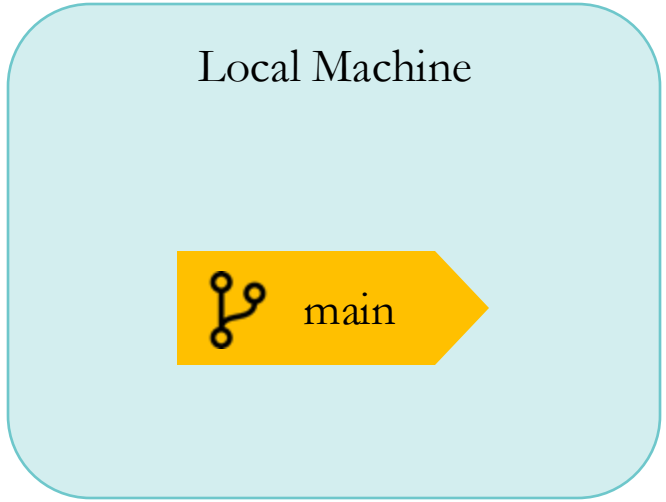
```
$ git remote -v
origin    https://github.com/<your_username>/mesa.git (fetch)
origin    https://github.com/<your_username>/mesa.git (push)
```



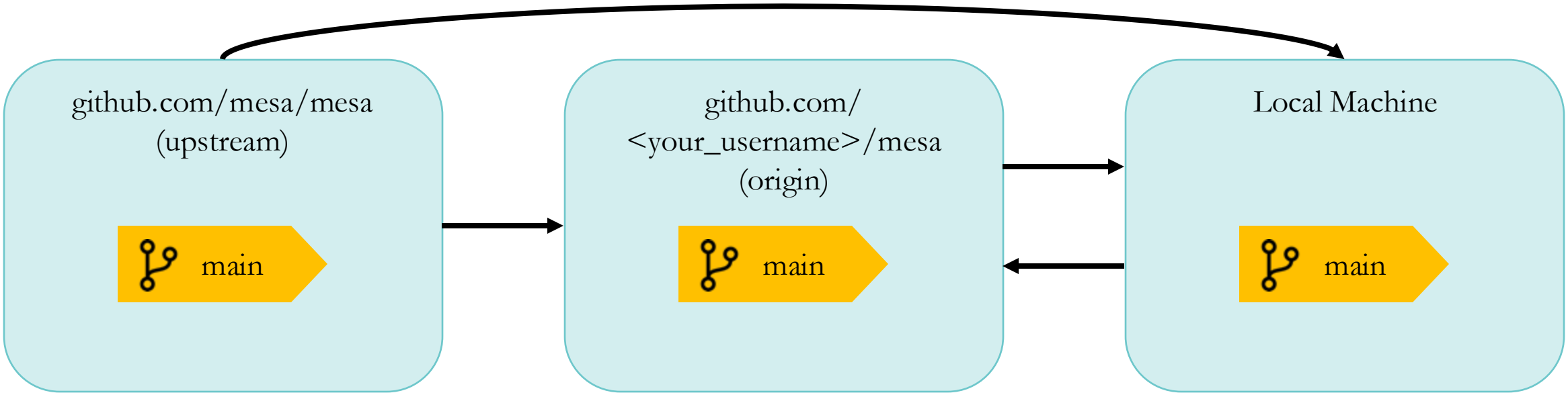
1.3 Clone to local machine

Optionally add `upstream` remote to pull latest from Mesa:

```
$ git remote add upstream https://github.com/mesa/mesa
$ git remote -v
origin   https://github.com/<your_username>/mesa.git (fetch)
origin   https://github.com/<your_username>/mesa.git (push)
upstream        https://github.com/mesa/mesa (fetch)
upstream        https://github.com/mesa/mesa (push)
```



1.3 Clone to local machine



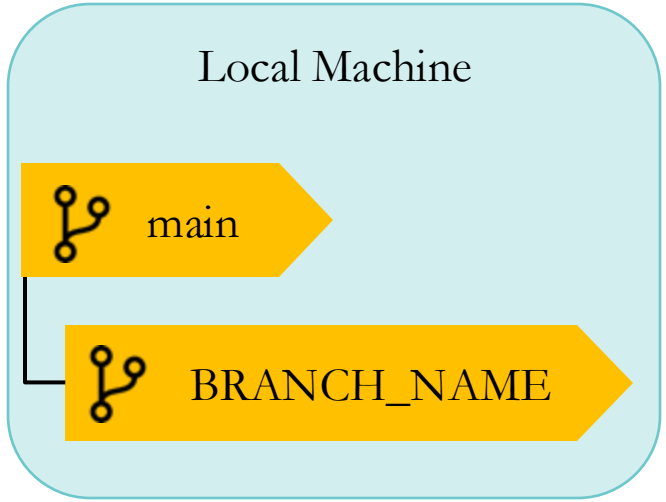
1.4 Create a branch

For clear history:

```
$ git config pull.rebase true
```

To isolate your work:

```
$ git checkout -b BRANCH_NAME
```



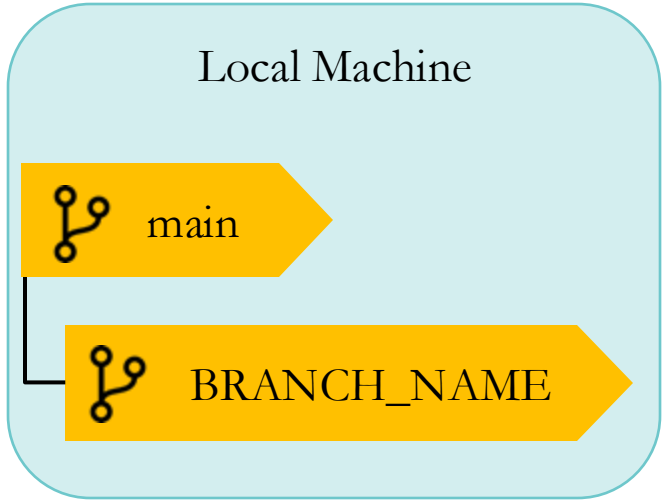
1.5 Install dev dependencies

Create a virtual environment (using venv as example):

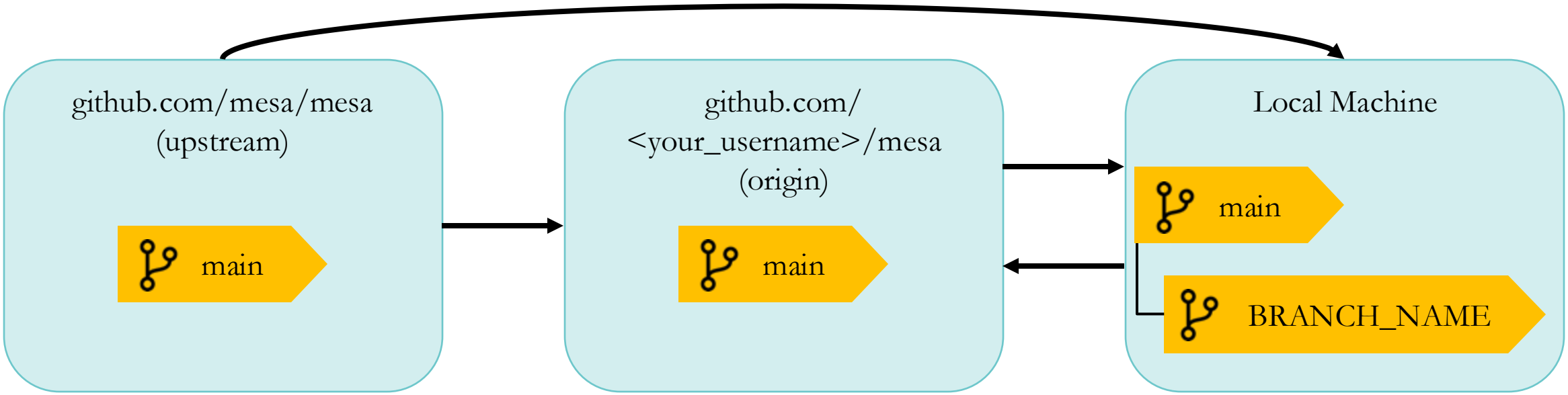
```
$ python3 -m venv .venv
$ source .venv/bin/activate
```

Install development dependencies:

```
$ pip install -e ".[dev]"
```



Part I: Setup 🎉🎉🎉


github.com/mesa/mesa
(upstream)
main
github.com/
<your_username>/mesa
(origin)
main
Local Machine
main
BRANCH_NAME

Part II: Development

- Edit the code. Save.
- Git add the new files and files with changes: `git add FILE_NAME`
- Git commit your changes with a meaningful message: `git commit -m "Fix issue X"`
- If implementing a new feature, include some documentation in docs folder.
- Make sure that your submission works with a few of the examples in the examples repository. If adding a new feature to mesa, please illustrate usage by implementing it in an example.
- Make sure that your submission passes the GH Actions build. See "Testing and Standards below" to be able to run these locally.
- Make sure that your code is formatted according to [the black] standard (you can do it via pre-commit).

2.1 Make changes

- Edit the code. Save.
- If implementing a new feature, include some documentation in docs folder.
- Make sure that your submission works with a few of the examples in the examples repository. If adding a new feature to mesa, please illustrate usage by implementing it in an example.

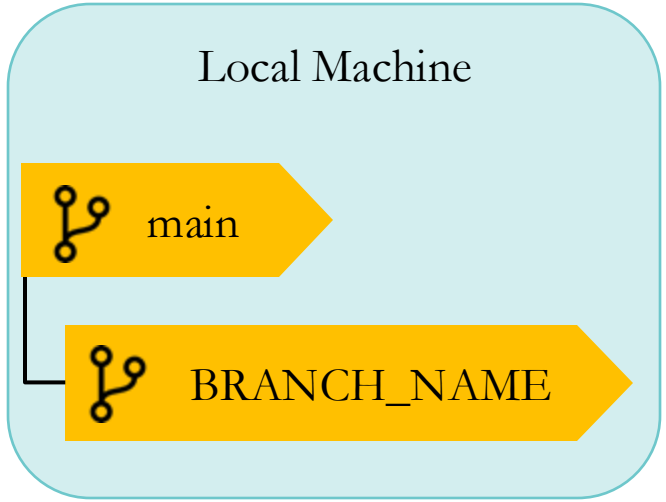
2.2 Add & commit changes

- Git add the new files and files with changes:

```
$ git add FILE_NAME
```

- Git commit your changes with a meaningful message (optionally with the #):

```
$ git commit -m "fix issue #123"
```



2.3 Testing & formatting

Set up pre-commit if it's not available:

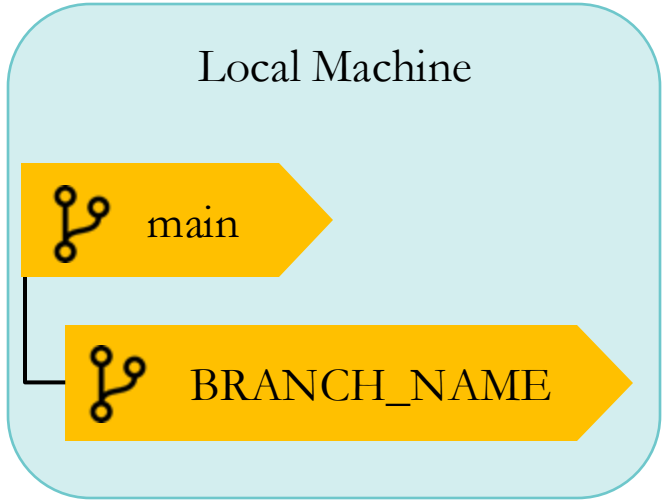
```
$ pip install pre-commit
$ pre-commit install
```

Run pre-commit (or just ruff) locally:

```
$ pre-commit run --all-files
$ ruff check . --fix
```

Run tests locally:

```
$ py.test --cov=mesa tests/
```



Note: GitHub will run these checks again via GitHub Actions when you open your PR.

Part II: Development 🎉 🎉 🎉

- Edit the code. Save.
- Git add the new files and files with changes: `git add FILE_NAME`
- Git commit your changes with a meaningful message: `git commit -m "Fix issue X"`
- If implementing a new feature, include some documentation in docs folder.
- Make sure that your submission works with a few of the examples in the examples repository. If adding a new feature to mesa, please illustrate usage by implementing it in an example.
- Make sure that your submission passes the GH Actions build. See "Testing and Standards below" to be able to run these locally.
- Make sure that your code is formatted according to [the black] standard (you can do it via pre-commit).

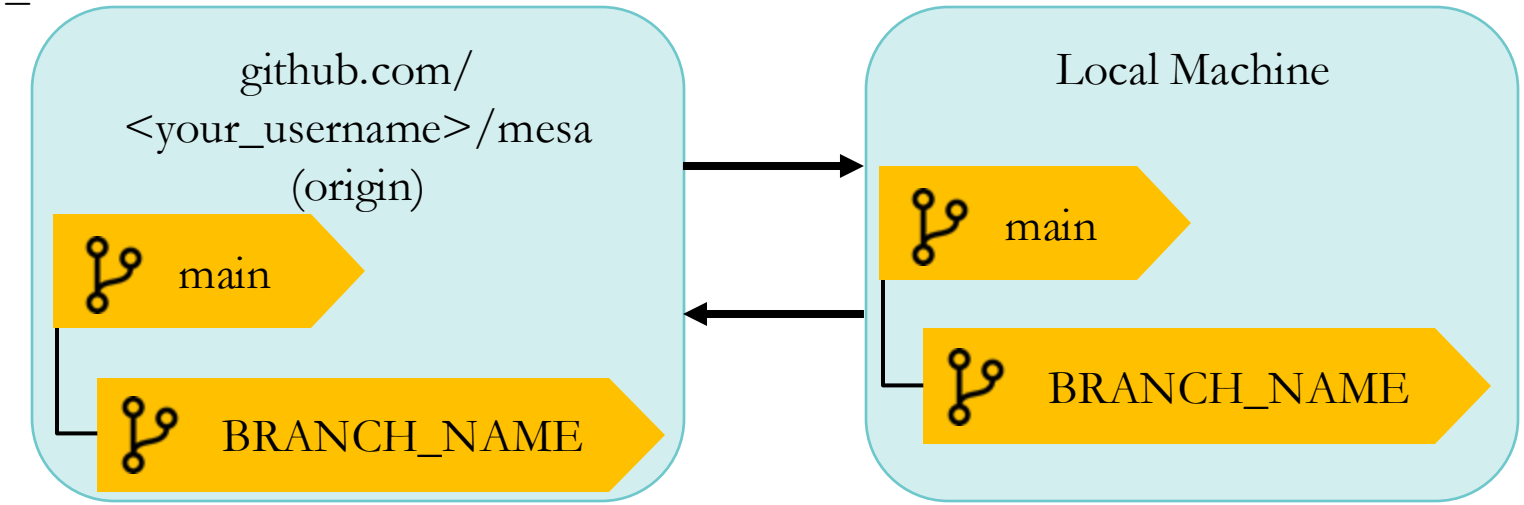
Part III: Submission

- Push your changes to your fork on Github: `git push origin NAME_OF_BRANCH`
- Create a pull request.
- Describe the change w/ ticket number(s) that the code fixes.
- Format your commit message as per Tim Pope's guideline.

3.1 Push to your fork

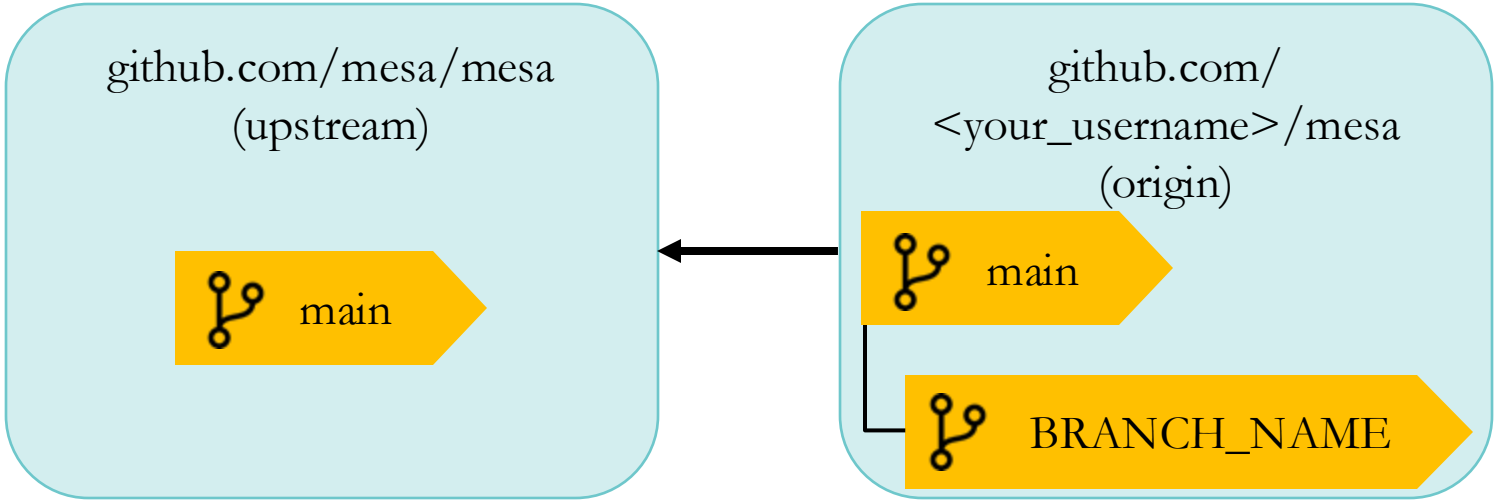
Push your changes to your fork on GitHub:

```
$ git push -u origin BRANCH_NAME
```

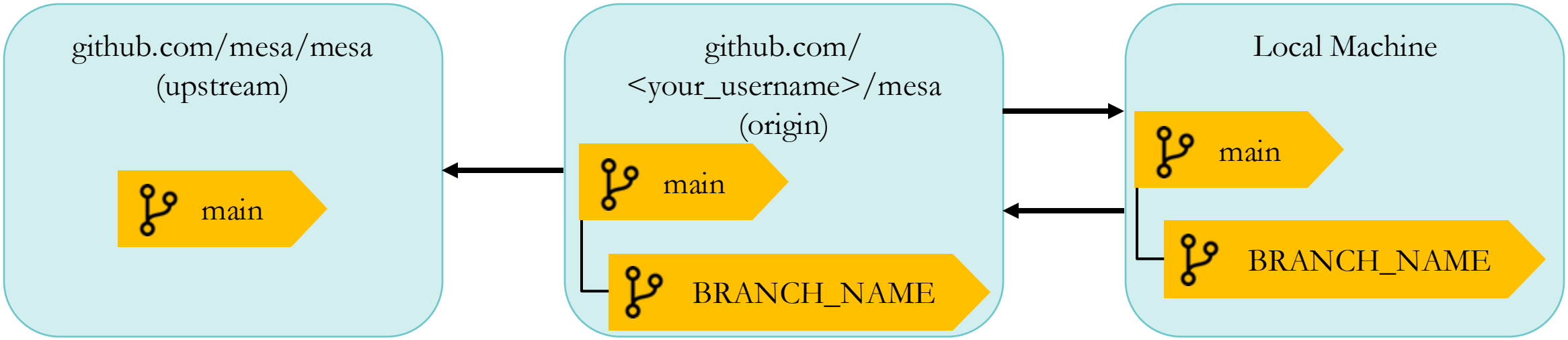


3.2 Create a pull request

On GitHub, propose merging BRANCH_NAME from your fork into Mesa's main branch. Link PR to ticket.



Part III: Submission



Practice tasks

Check the tasks at https://github.com/abmind-community/contrib-practice

Task 01: Add yourself

- Practise fork/branching/PR workflow.

Task 02: Fix style

- Use ruff & pre-commit to fix style issues.

Task 03: Add tests

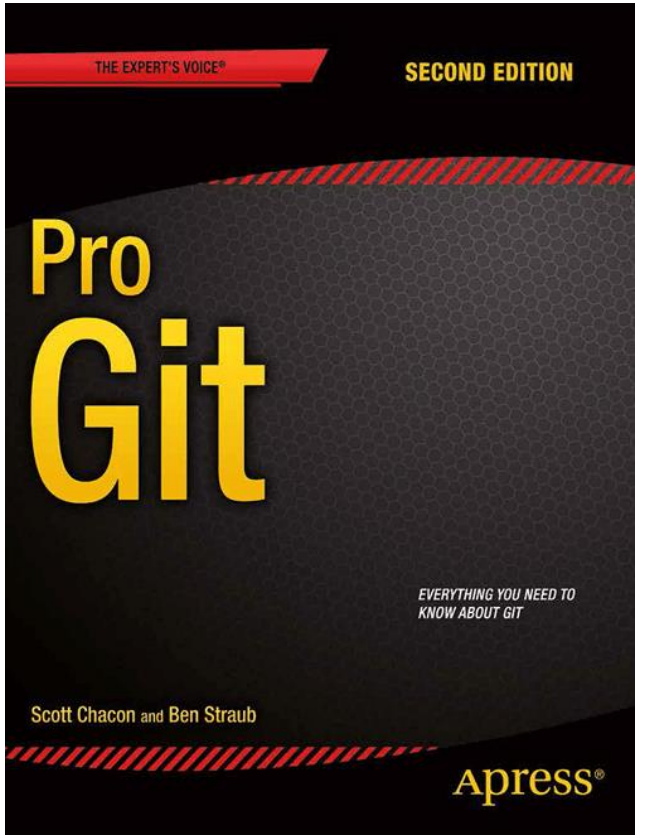
- Write unit tests and update existing PR.

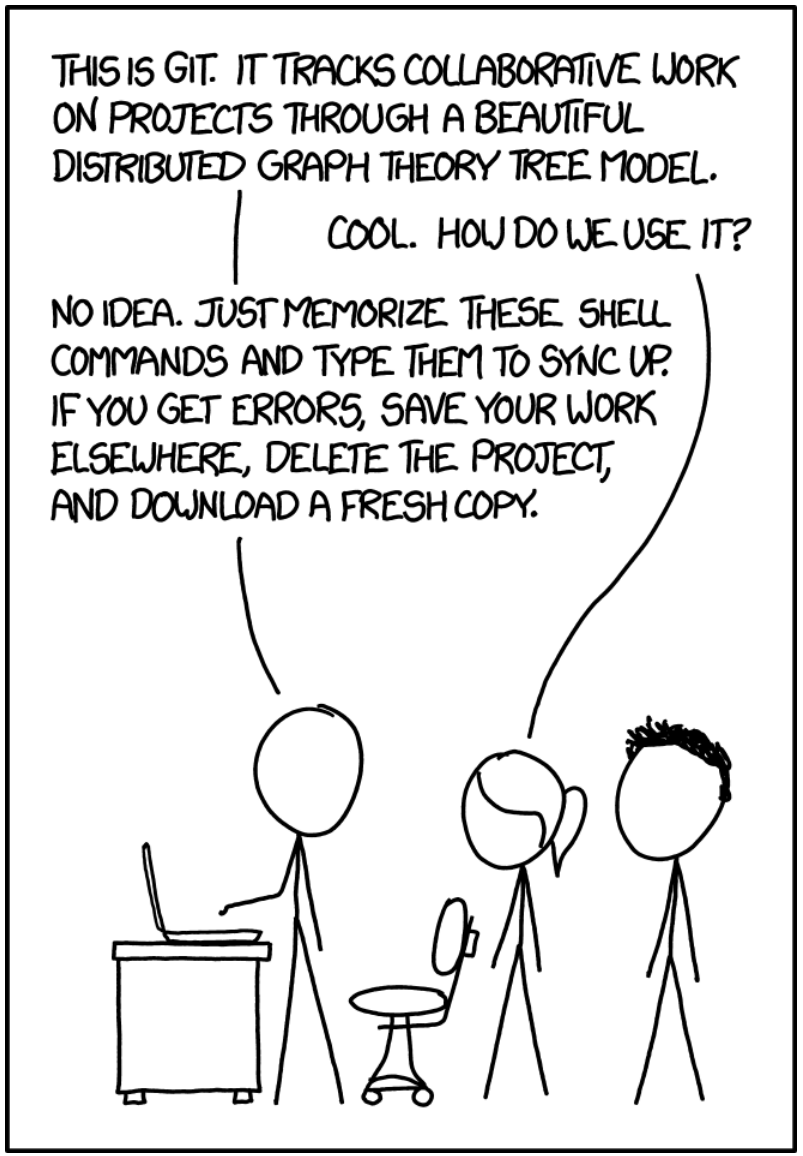
Task 04 & 05: Merge conflict

- Simulate and resolve a basic merge/rebase conflict.

Book

- Available at: https://git-scm.com/book
- Chapters 1, 2, 3, 5 & 6.





Source: https://xkcd.com/1597