

Software project

December 19, 2023

Project management with Gitlab

0.1 Project structure

See `README.md` of the repo. Main tools include:

- **poetry**: Build tool and packet manager for Python projects
- **docker**: Package app and its dependencies into a container and deploy by pushing the Docker Image to Docker Hub

Naming convention: - for directory (ex: `a-directory`), _ for file (ex: `a_file`)

0.2 Branching strategy

- **Feature branching**: `main` for stable releases and other branches with the following branch naming conventions: `your-initials/<feature-name>` (ex: `jn/symplectic-integrator`)

0.3 Documentation

- **In-Code Documentation**: Encourage clear in-code documentation (like docstrings).
- **README and Wikis**: Maintain a project README and use GitLab's Wiki feature for more extensive documentation.

0.4 Commit Guidelines

- Small, frequent commits

0.5 Testing

- **Unit and Integration Tests**: Write comprehensive unit and integration tests.
- **Test Coverage**: Aim for high test coverage and consider incorporating a tool that measures it.
- Make sure the unit and integration tests pass before asking for revision of your MR!

0.6 Code Reviews and Merge Requests

- **Merge Requests (MRs)**: Use GitLab's merge request feature for merging any branch into `main`.
- **Code Review Process**: A code review process where at least one other team member reviews the code before it's merged.
- **Discussion and Approval**: Use the discussion feature in MRs for feedback and approvals.

0.7 Deployment *Optional*

- **TODO**: set up CI/CD pipelines for deploying to staging and production environments.

0.8 Versioning and Releases

- **Semantic Versioning**: Use semantic versioning for your project.
- **Tagging Releases**: Tag releases in GitLab with the version number.

Things that could be helpful

- Can I numlify / scipify this line / code / function? (The core of Numpy is implemented in C so it allows very efficient numerical computations. So instead of using basic data structures such as list, use Numpy array and the various Numpy functions)
- Strive for modularity (→ break your code into modules of specific purposes, also easier for debugging, testing), explicitness and readability (names of objects should reflect their nature and functionalities). Keep things short, brevity is your friend!
- Please be mindful! Coding/programming can be frustrating so take short breaks, keep yourself hydrated, energized, peaceful, etc. But please don't push broken or incomprehensible code to the repo and hope things work. Somebody will have to clean your mess... If you are not sure about something, it's always good to ask.
- ChatGPT could be a great assistant for debugging / suggestions. But again please don't mindlessly paste suggested code snippets. Please try understand the response and see if it's applicable. Of course, some numbers will come out when your code runs but do they actually make sense? Do a quick sanity check!
- Yes, test and test and test! Always test your implementation to see if it works as expected! Point is easy does it. Tests should be simple, lightweighted, etc. If you don't know what the output of the input you are putting into the test, what's the point of testing?
- All in all, let's practice The Zen of Python