

Software project

December 11, 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

- ***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.