GitOps (Git Operations) Guidelines

Prerequisite: Introduction to Git (5 minutes read): https://rogerdudler.github.io/git-guide/

Vocabulary:

- Issue = Task: a unit amount of work, defined by an exact set of technical requirements and deliverables.
- Merge request = Pull request: make a request to merge your branch to a target (the main) branch. A pull request typically requires approval manually by another team member (and automatically irl with CI/CD pipeline).

First time setup

It is recommended to use VSCode for your development. It is recommended to install `ms-python.black-formatter` extension to enforce a standard Python coding style.

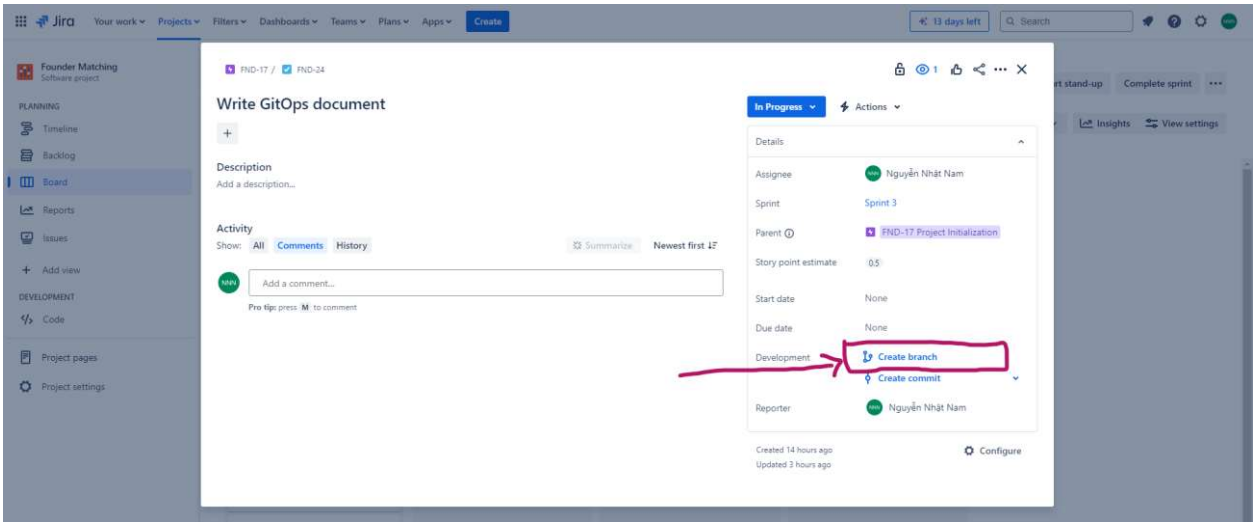
Clone our repository in the current directory with:

```
$ git clone https://github.com/HARDeConstruction/FounderMatching.git
```

in your favorite code terminal.

Start working on your ticket

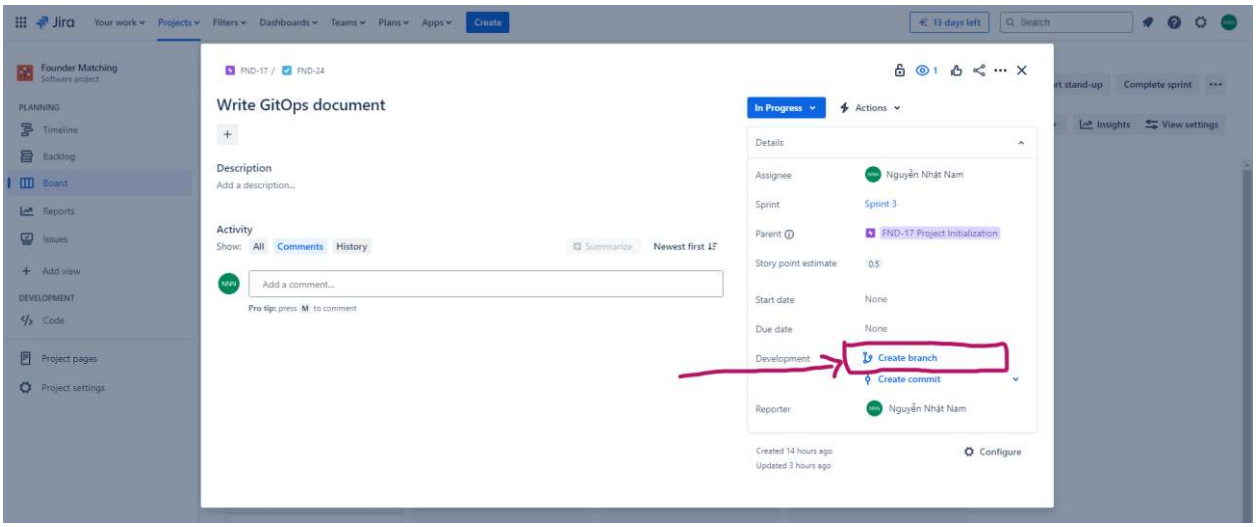
So you have a TO DO ticket assigned to you, every ticket will have an issue key as shown in the image below.



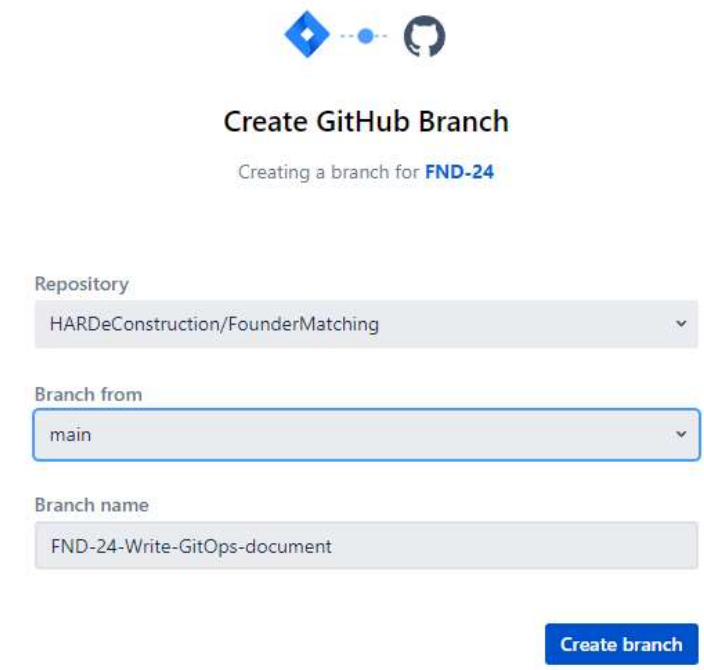
Each task should be done within its own branch. Branches are used to develop features isolated from each other. The "main" branch is the "default" branch when you create a repository. Use other branches for development and merge them back to the master branch upon completion.

Creating your development branch

The first thing you want to do is to create a new branch from `main`, naming the new branch with the issue key of your ticket. This can be done in the issue page by clicking "Create branch" from the issue menu:

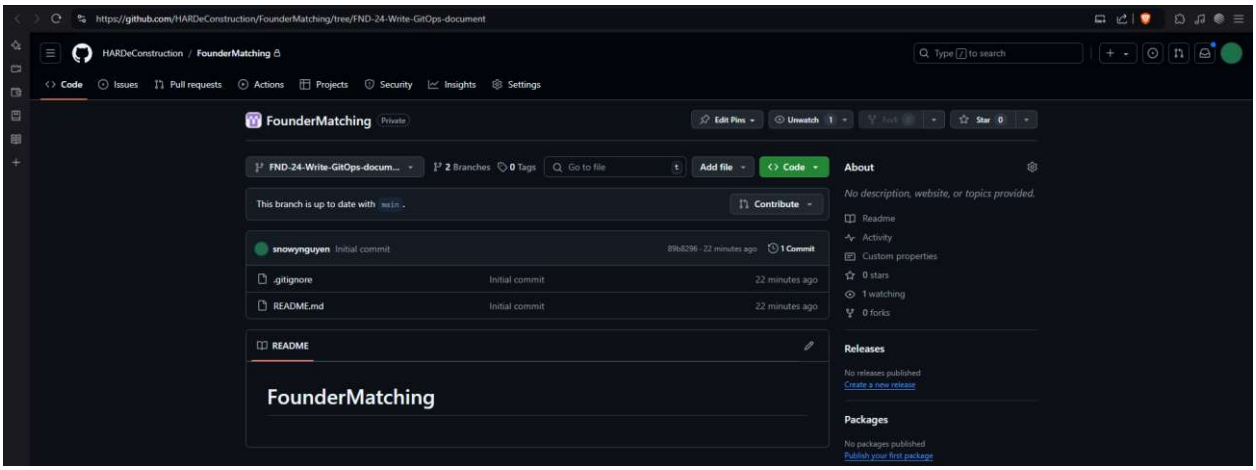


Then, a window appears. Make sure that the branch name begins with "FND-?" - your issue key.



Note that, sometimes, it can be convenient to branch from a branch other than main, especially when issue B develops on issue A that has not been merged to main. In that case, it is advised to branch from A instead.

After which, your branch will be created on Github, you can check it on our Github repo located at https://github.com/HARDeConstruction/FounderMatching/branches.

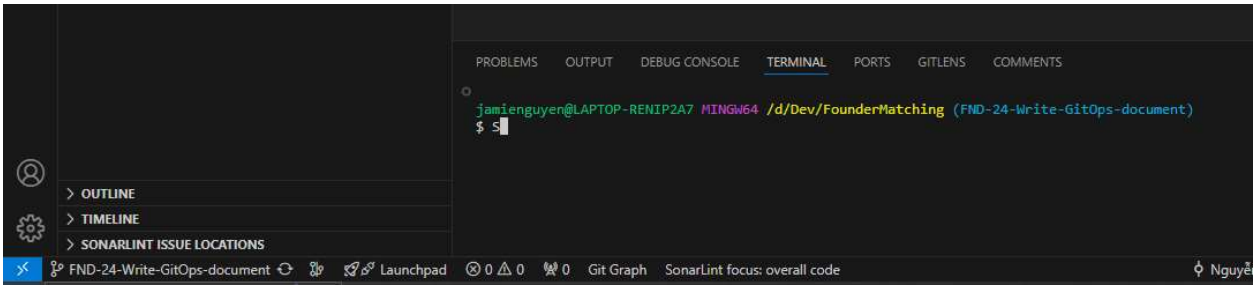


Start working by checking out the newly created branch. On your favorite IDE terminal, for example, bash on VSCode. Don't forget to pull before checking out to "update" your local git repository with the remote git repository.

```
git pull
```

```
git checkout <branch-name>
```

Make sure you are on your branch before you start to work! Your terminal should show your current branch. Your bottom left corner also shows your current branch name.



While working on your ticket

You may think committing code is as straightforward as just committing and pushing your changes to the code. But there's a little more to it than that.

Make small, specific commits

Smaller commits make it easier to revert code to a previous state if there's a problem. If your commit affects too many areas, reverting back could mean losing a lot of code.

Check your warning

Sometimes you get a compiler warning. Black extensions also set warnings. Don't ignore warnings. If you don't know how to fix a warning, let the team know in your commit message. If you are sure that a warning is a false warning, note that in your commit message, or better try to suppress the false warning programmatically or with a setting configuration.

Write meaningful commit message

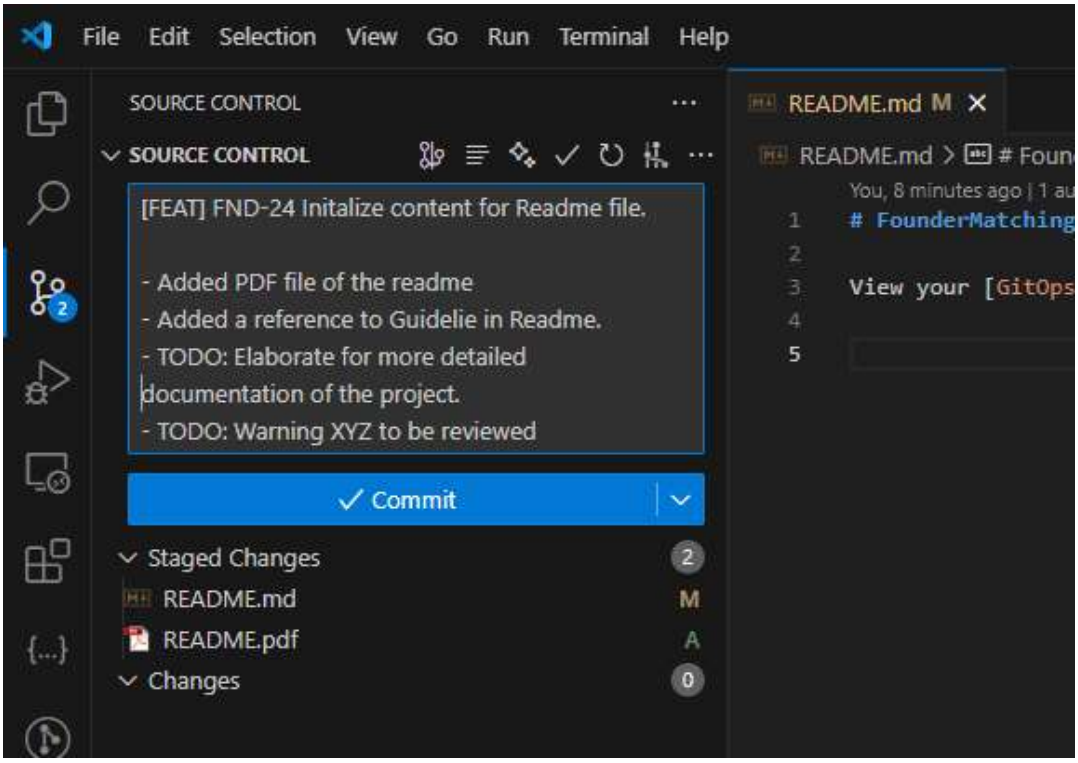
Prefix your commit messages with imperative commands such as: feature [FEAT], bug fix [FIX], refactor [REFACT], remove [REM].

Every single commit message must also contain the issue key. A basic example:

```
[FIX] FND-24: Fix UI login button not showing
```

A commit message can be multi-line. In this case, the first line contains the key summarizing the commit, subsequent lines elaborating on the content of the commit. For example, below is a commit message for FND-24:

```
[FEAT] FND-24 Initalize content for Readme file.
- Added PDF file of the readme
- Added an introduction for Readme.
- TODO: Elaborate for more detailed documentation of the project.
- TODO: Warning XYZ to be reviewed
```



Further Reading: https://www.freecodecamp.org/news/git-best-practices-commits-and-code-reviews/ (10 minutes read)

Push your commits

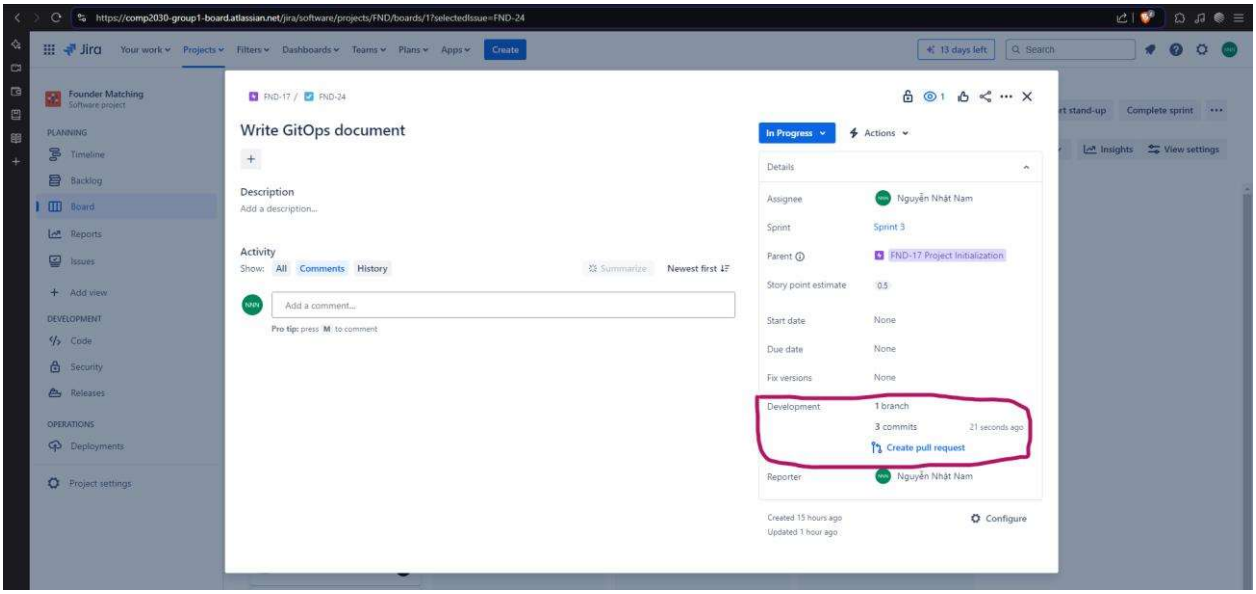
You have to push your commits to make your changes visible to other developers.

```
git push
```

It is recommended to push your commit as soon as you commit it locally. Reverting a pushed commit is a bit hazardous though, so use your own judgement. At least push your commits before you end your coding session.

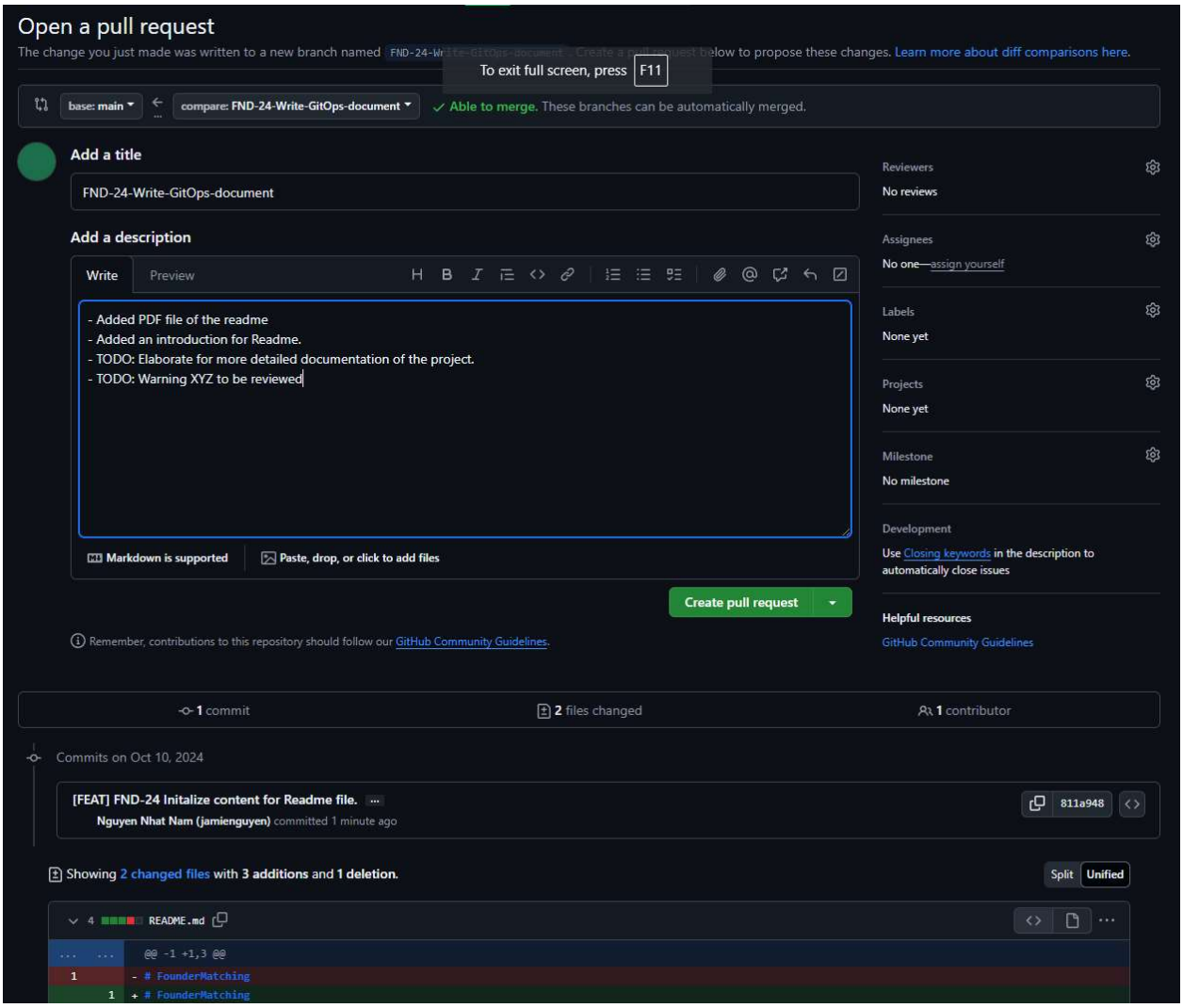
Creating a pull request (PR)

After finishing the task, you then create a PR to `main` (or to the parent branch you were cloning from), by going to your issue on Jira and click on "Create pull request":



Your pull request will be automatically populated with the title `(issue-key): (Ticket summary)` and its description will be filled with the commit messages. If necessary, make some changes to the pull request title and description.

Remember to keep the issue key in the title. If applicable, include testing details, notes, warning. If you find something confusing, or unsure if your branch is working as expected, note it down.



After creating a PR, you then invite the team members to review your PR. Drag your issue to “In review” column in Jira as well!

Under any circumstances, **DO NOT MERGE** pull requests by yourself.