Development workflows in GIT

Ing. Pablo A. Deymonnaz - pdeymon@fi.uba.ar

Uriel Kelman - urielkelman@gmail.com

This document was translated from contents of the Programming Workshop I from the Engineering Faculty of the Buenos Aires University (Taller de Programación I, Facultad de Ingeniería, Universidad de Buenos Aires)

Index

1. GIT review
2. Branches
3. Workflows and PRs

1. Git review

What is GIT?

- Git is a free, open-source (meaning its source code is freely available) distributed version control system designed for efficiently managing projects of various sizes (large or small).
- It was developed in 2005 by Linus Torvalds (who is also the creator of the Linux operating system kernel) and is currently under continuous maintenance (the latest version is 2.26.2).
- The main use of Git as a tool is to deal with the need to version control the code of a project.

What does version control mean?

- When we work on a project, there is a flow that constantly repeats: we create new files, edit them, save them, reopen them, edit them again, save them again, and so on.
- Version control allows us to keep a record of the moment when we save the code of a project, recording in that record when we made the change, why we made it, and what were the things that changed compared to the previous version.
- Version control allows us to have a history of the code's project.
- For a single individual working on the project, we might think that keeping a record of its history is relatively simple: a couple of annotations would suffice.
- This is where Git shines: when we work collaboratively on a project. It is not the same to work alone on a project as it is to work with a group of people who want to modify the project code simultaneously (even the code of the same files).
- Git, as a version control system, provides us with all the tools to work collaboratively on a project with other programmers in a controlled manner.

Github

- GitHub is a platform that allows for the storage of Git projects.
- Is it the only one that exists? No, there are some other popular ones such as BitBucket or GitLab.
- GitHub (like any other storage platform) adds the abstraction of the repository: a place where all files belonging to a project can be saved. Each project has a unique repository, identified by a unique URL.
- On June 4, 2018, Microsoft acquired GitHub for a total of $7.5 billion.

Everyday operations in Git

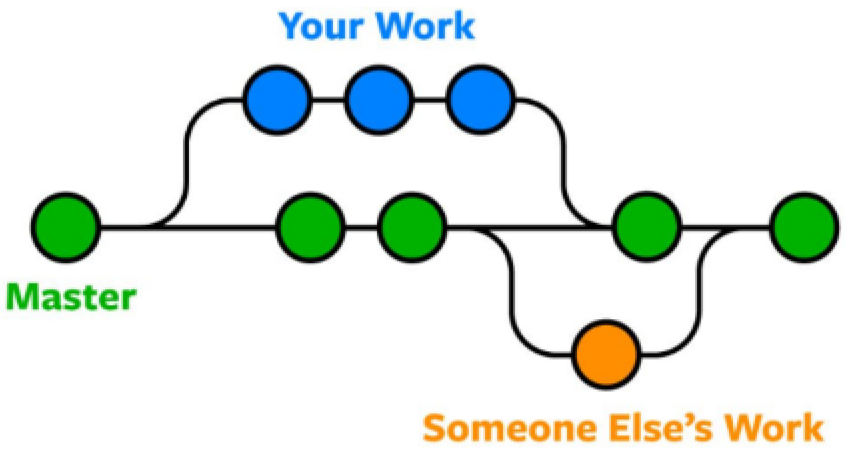
- Creating/cloning a repository.
- Creating a branch.
- Adding changes made to a branch.
- Committing the changes made.
- Pushing the changes to the remote repository.

2. Branches

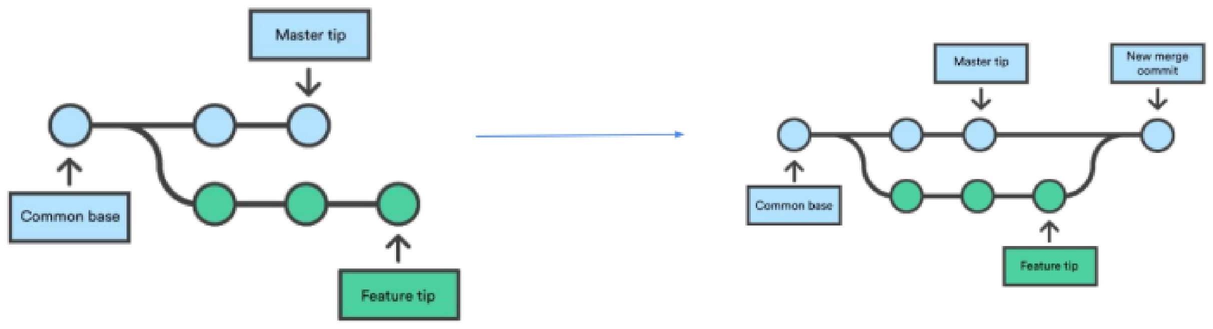
We can think of a branch as an isolated and independent unit for writing code. This has several advantages:

- Simple context changes.
- Branches with different roles (production, testing, etc.).
- Workflow based on features.
- Disposable experimentation.

It is of vital interest to be able to work on different branches at the same time. In this way, different developers can work on different functionalities in an isolated manner.



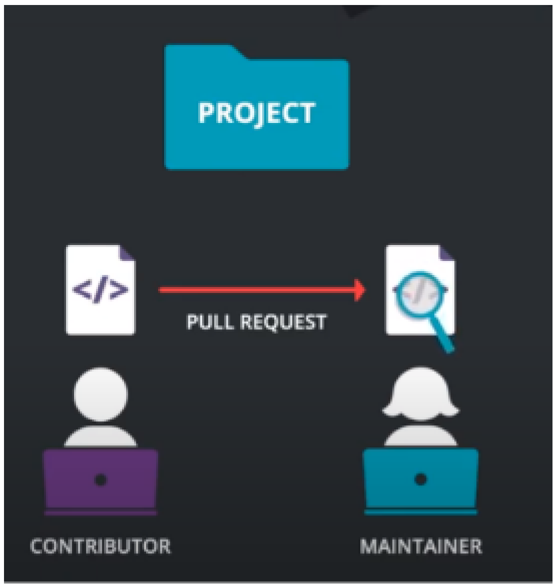
Once we finish working on a branch, the usual thing is to want that functionality to be mixed with all the previous ones. For this, we must perform a merge between the branches we want to mix.



3. Workflows and PRs

Defining a PR

A pull request (also known as merge request) is an event where a developer requests a review of code that implements a new functionality and that they wish to merge into some branch of the project (usually the main branch).

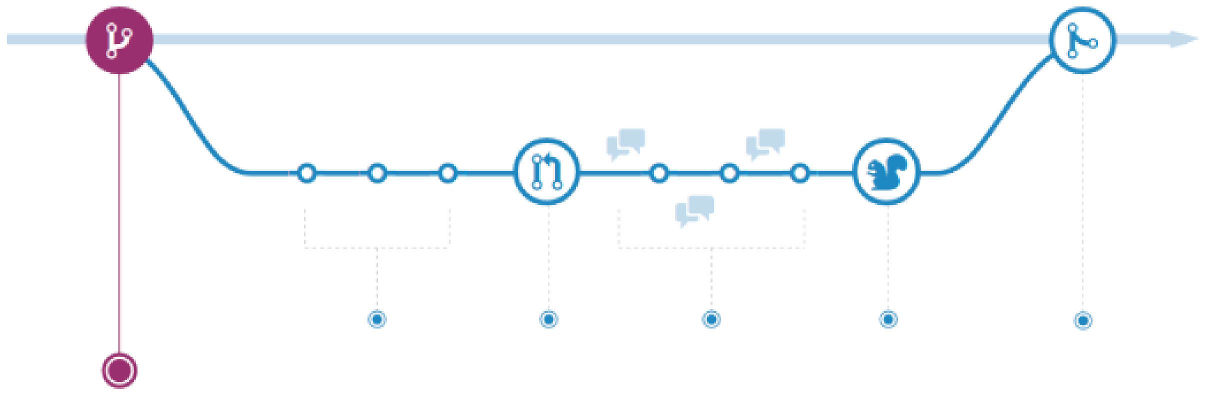


Advantages of working with PRs

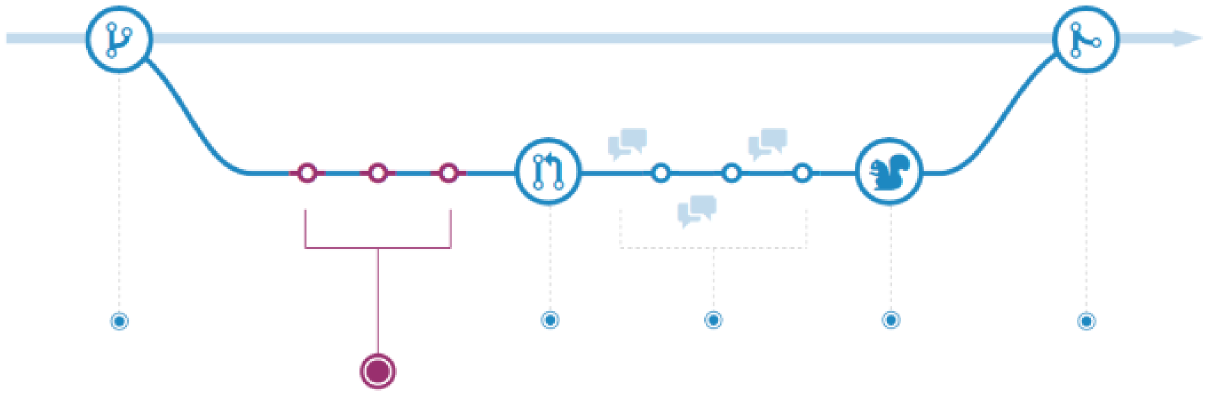
- It forces code reviews before merging new code into a branch.
- It allows for discussing changes to a feature before bringing it to a production environment.
- They provide a natural way to give feedback on code written by a collaborator.

Github Flow

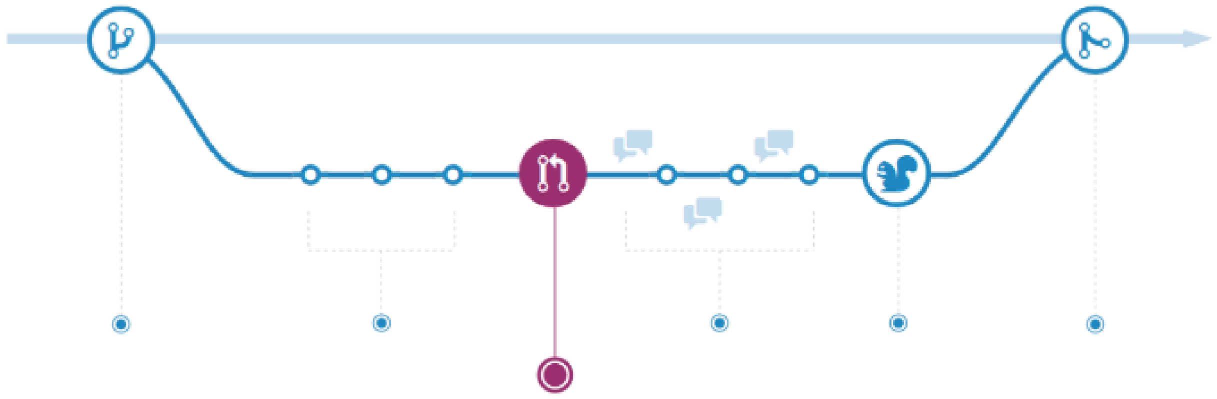
Create a branch based on a feature that we want to implement.



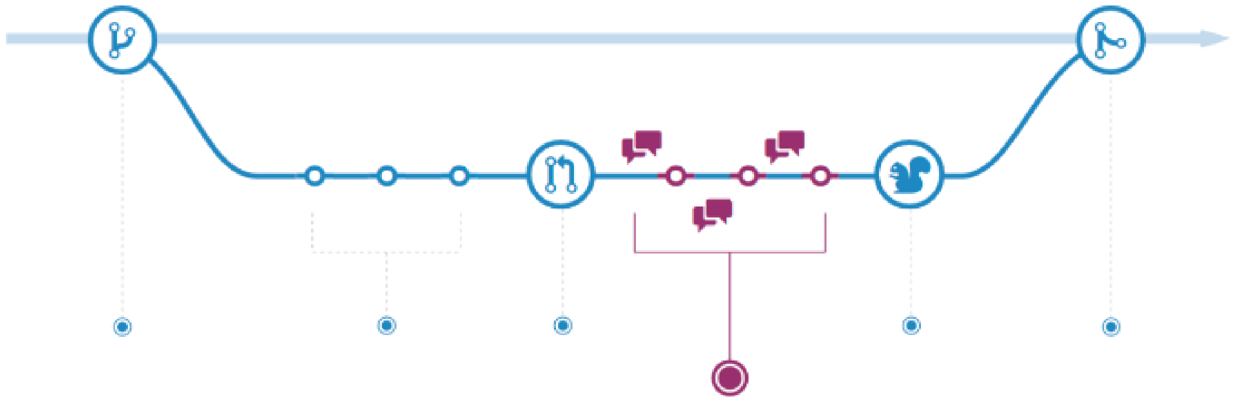
Add changes and commits to the branch being worked on.



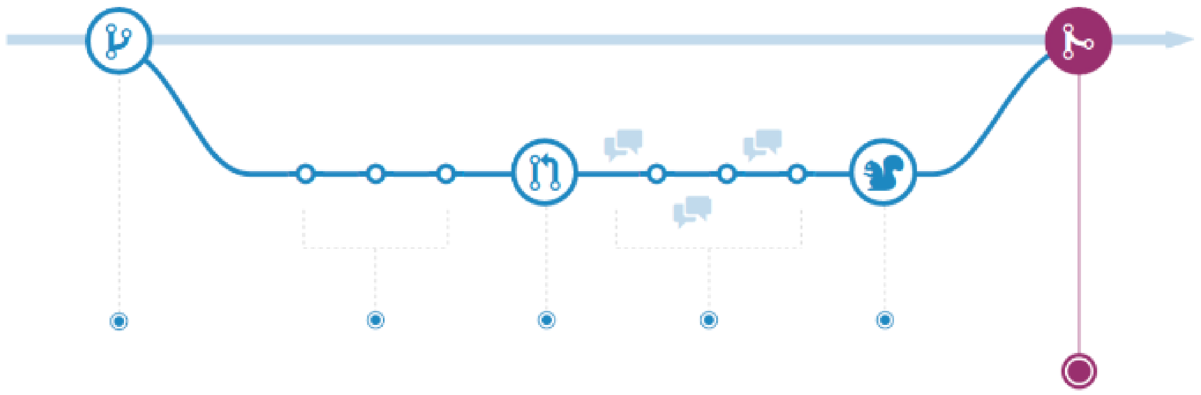
Once we finish the feature, we open a pull request that points to the branch we want to merge into.



Review of the code and possible discussion about it.



The approval of the pull request and the merge.



Synthesis

1. The developer finishes the feature and opens a pull request that points to the corresponding branch.
2. The PR is open for another developer to review and start a discussion.
3. The reviewer provides feedback in the form of comments.
4. The developer responds to the comments, sometimes making modifications to the code, other times answering questions or explaining a development decision.
5. Once all comments are resolved, the PR can be approved and merged.

Conflicts

- When merging two branches or trying to perform a pull request, it's possible that git will notify us of a conflict between the branches being merged.
- This happens when there are modifications in both branches for the same portion of code. In this case, git doesn't know which version should prevail, so a conflict is generated.
- To properly perform the merge, the conflict must be resolved locally, thus eliminating ambiguities and indicating which portion of code will appear in the remote repository.

More Complex Needs

When we are working in a company, the needs around a project differ substantially from those we have in an academic project. Some of them could be:

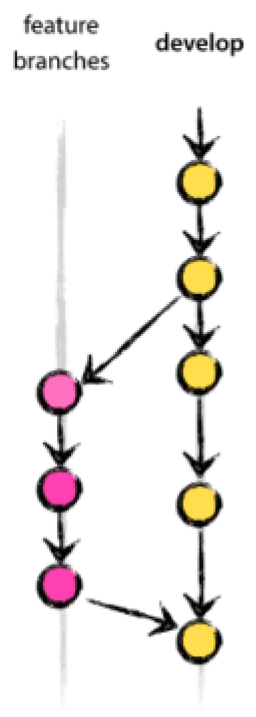
- Possibility of having more than one environment to test the application.
- Use CI/CD methods on different branches.
- Have an organized method of making a hotfix in case there is an error in a version uploaded to production.

Branches and workflow

A slightly more sophisticated approach is to have two main branches instead of two:

- Main: Contains the latest version uploaded to production.
- Develop: This is where developers start when they want to develop a feature. They also merge feature branches into this branch.

When developers want to code a feature, they start from the develop branch. When they finish, they create a pull request against develop and wait for its approval.



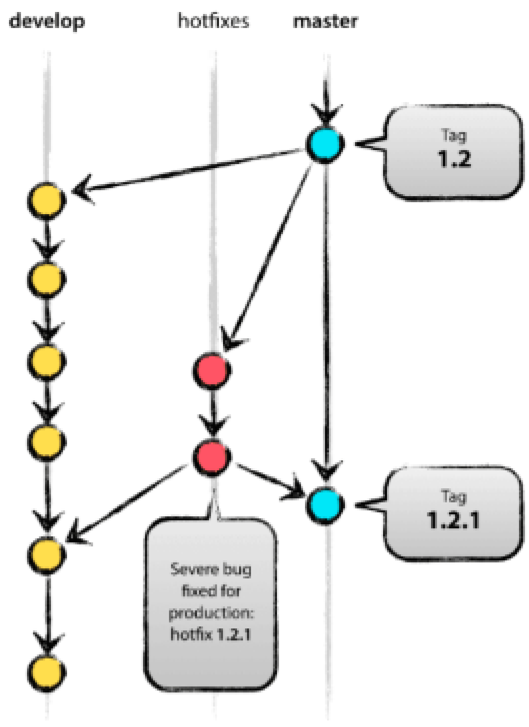
Once it is desired to deploy a version into production because it is considered that enough features were coded, a release branch is created. On this branch:

- Errors are searched for.
- Fixes are made on errors that can be found.
- A set of regression tests that comprehensively test the application can be run.

Once everything is ready:

- The release branch is merged both into master and develop.
- A tag is created in the master branch (it should reflect the name of the version).
- The application is deployed in the production environment.

Additionally, branches can be created for hotfixes: urgent errors that cannot wait for the deployment of the next version.



Bibliography

- Pro Git Book: https://git-scm.com/book/en/v2
- Introductory videos: https://git-scm.com/videos
- https://www.atlassian.com/es/git/tutorials
- https://guides.github.com/
- https://medium.com/@patrickporto/4-branching-workflows-for-git-30d0aaee7bf
- https://blog.axosoft.com/pull-requests-gitflow/
- https://nvie.com/posts/a-successful-git-branching-model/
- https://open.spotify.com/episode/0QHoMlwANPAp5XYFbiOxCK?si=49a9562c09a444cc&nd=1