Introduction to Git

What is version control and why is it useful?

Managing large software projects with many developers as contributors can be a daunting task. Without the proper tools and processes to track and document changes, there is no way to guarantee that the different contributions are appropriately integrated into the project without breaking features from older versions. Communication between contributors is also tricky since there is no way to track who is responsible for a particular component or module. In addition, when problems occur, rolling back to a previous version is not easy.

This is where **version control** comes in handy. According to the Atlassian Bitbucket website, version control can be defined as follows:

> *Version control, also known as source control, is the practice of tracking and managing changes to software code.*

Version control offers so many advantages and there are many free options to choose from that there is no reason to go without them these days. Among those advantages:

- **Easy collaboration:** contributors can work on their features separately and then merge their work easily. There is no need for a developer to wait for someone else's work to be complete to start his/her own. The merge process insures that teammates do not step on each other's toes.
- **Transparency:** every developer has access to the whole development history log. This way, they all know who contributed what and when. This makes communication easier among teammates and creates a culture of shared responsibility.
- **Troubleshooting:** every small change to your code is tracked and documented. This way, when problems occur, you can easily compare versions to identify the

problem instead of going through a costly debugging process.

- **Rich tooling:** version control solutions integrate seamlessly in development environments, with CI/CD frameworks that we will discuss later. Also, online platforms that make it possible to store code in the cloud like GitHub and GitLab offer interesting features to communicate between collaborators.

Git: the most popular software for version control

Git is software for version control that was first introduced by Linus Torvalds to suit the needs of the largest open-source project on the planet: the Linux operating system!

Git predecessors like CVS (Concurrent Versions System) and SVN (Subversion) are based on a centralized model, meaning that the project code base is stored on a central server. That way, contributors only keep the latest version of the files they are working on locally. All the version history is kept on the server.

Conversely, Git is a **distributed version control system (DVCS)**. It means that every contributor stores a copy with **the full version history on his/her local computer**. However, a remote server is generally used to store the “authoritative” code repository. This offers several advantages:

- **Speed/efficiency:** saving one’s work is faster since it is done locally, without having to communicate with a remote server. Access to the shared remote repository is only necessary when they need to push a complete bulk of work.
- **Offline work:** developers do not need an internet connexion to add their work to the version history so they can work offline. They can push it later in the shared remote repository.
- **Easy branching:** having the whole version history available locally makes it easier for developers to create new branches and experiment. They only share their work with others once they have something ready for production.
- **Backup by design:** DVCS makes it very unlikely to lose the code base. Unlike centralized approaches, even if something catastrophic happens to the central server and its backups, every developer has a full copy of the repository locally, and could act as a backup source!

Today, Git is undeniably the most popular software for version control.

Online version control and collaboration platforms

You might have heard of GitHub, GitLab, or Bitbucket. These are not to be confused with Git. They are commercial platforms that use Git (and other version control software for some) to offer fully managed services for developers to share their code and collaborate online.

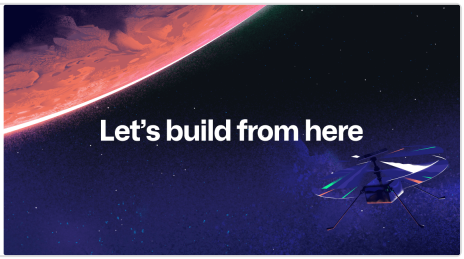
In this course, we will be using GitLab.

References

GitHub: Let's build from here

GitHub is where over 83 million developers shape the future of software, together. Contribute to the open source community, manage your Git repositories, review code like a pro, track bugs

https://github.com/



The One DevOps Platform

From planning to production, bring teams together in one application. Ship secure code more efficiently to deliver value faster.

https://about.gitlab.com/

Bitbucket | Git solution for teams using Jira

Bitbucket Cloud is a Git-based code and CI/CD tool optimized for teams using Jira.

https://bitbucket.org/product