Version control, Git, GitHub and GitFlow

Criscely Luján[1,2]
criscely.lujan@ird.fr

Nicolas Barrier[2]
nicolas.barrier@ird.fr

[1]Université Paris-Sud, UMR MARBEC

[2]IRD, UMR MARBEC

April 11, 2019

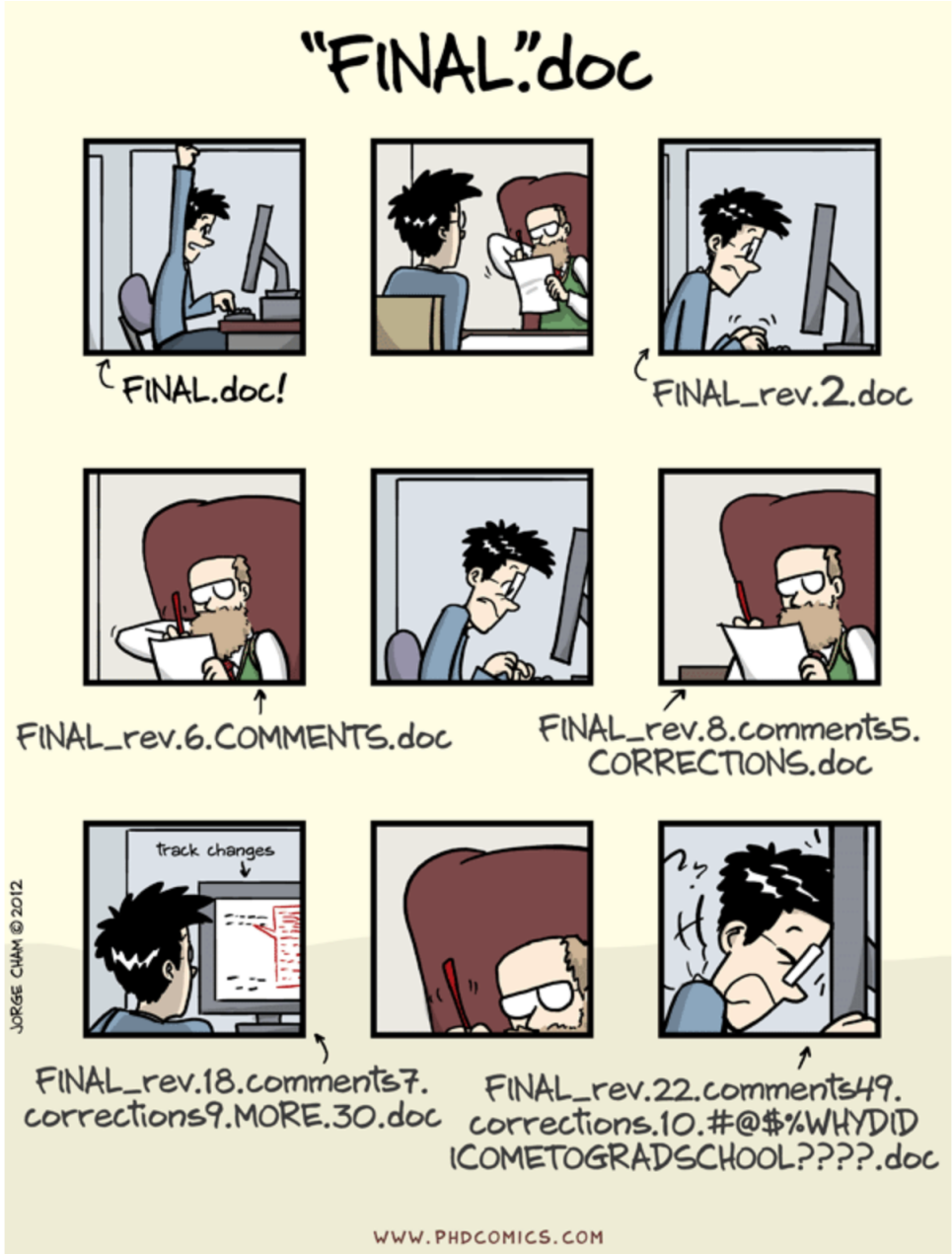
Version control

Also known as **revision control** or **source control**.

... "*is the management of changes:*

- *documents*
- *computer programs*
- *large web sites*
- *other collections of information ...* "

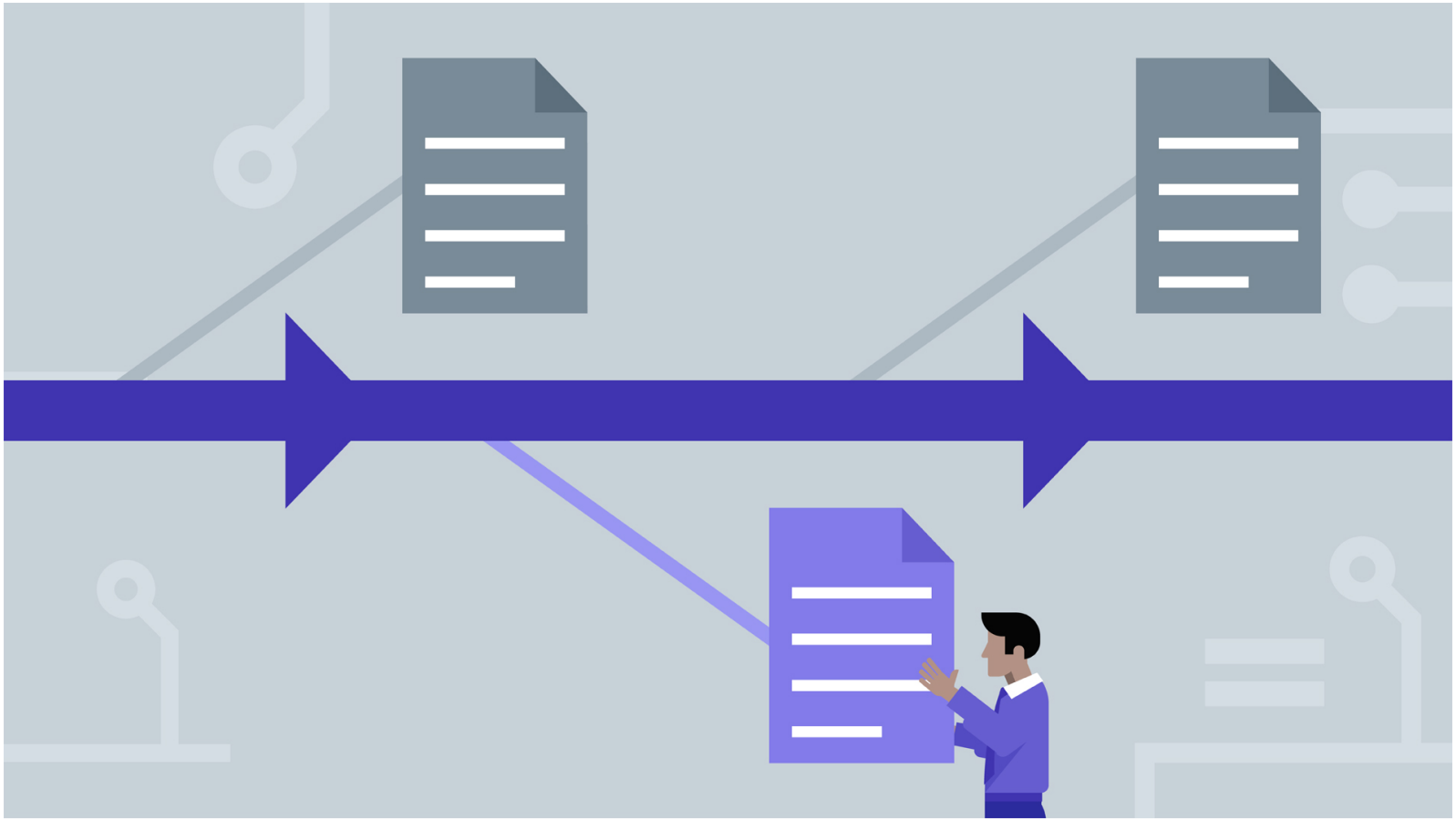
Why version control is important?



Why version control is important?

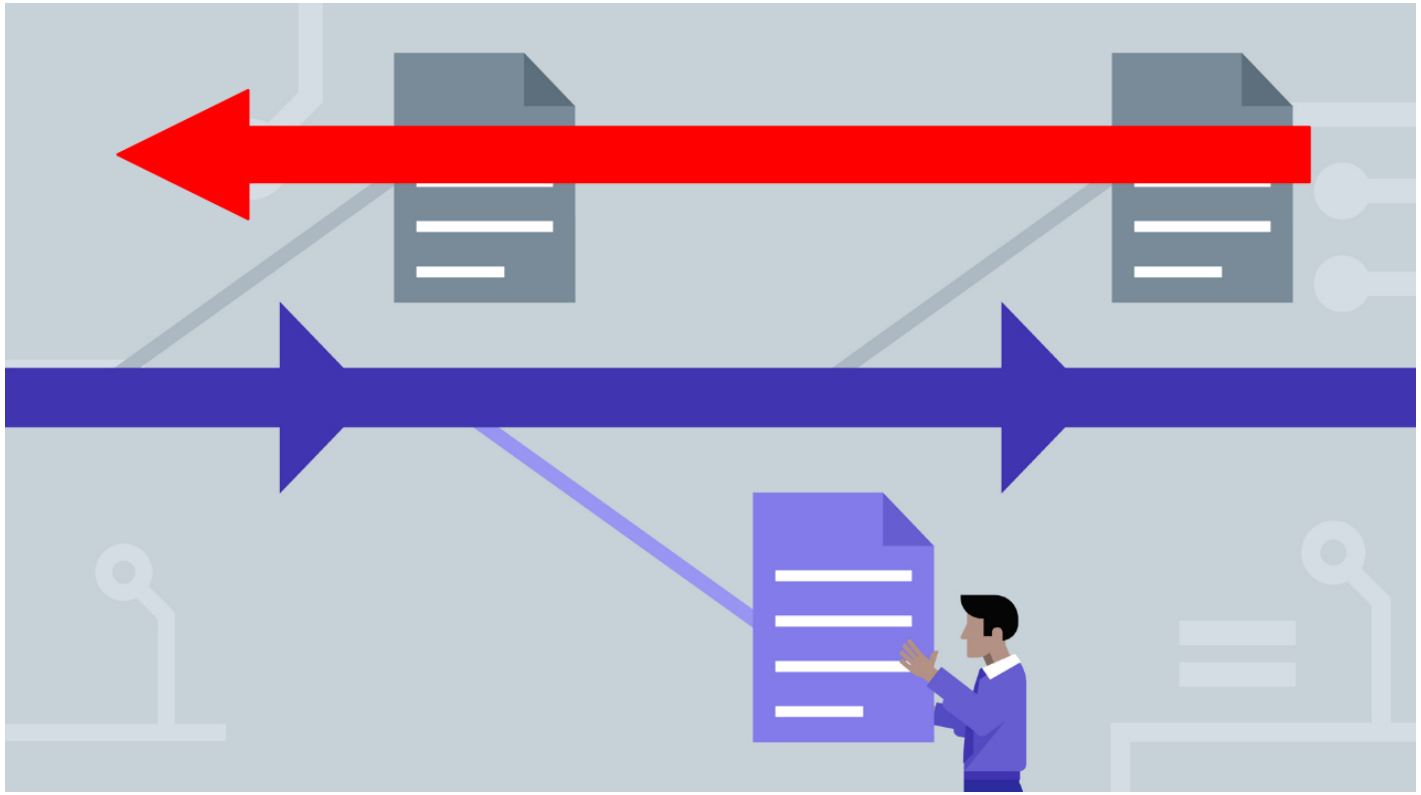
Storing **version** (properly).

- Saving successive changes ("commit")
- Versioning (v0.1)



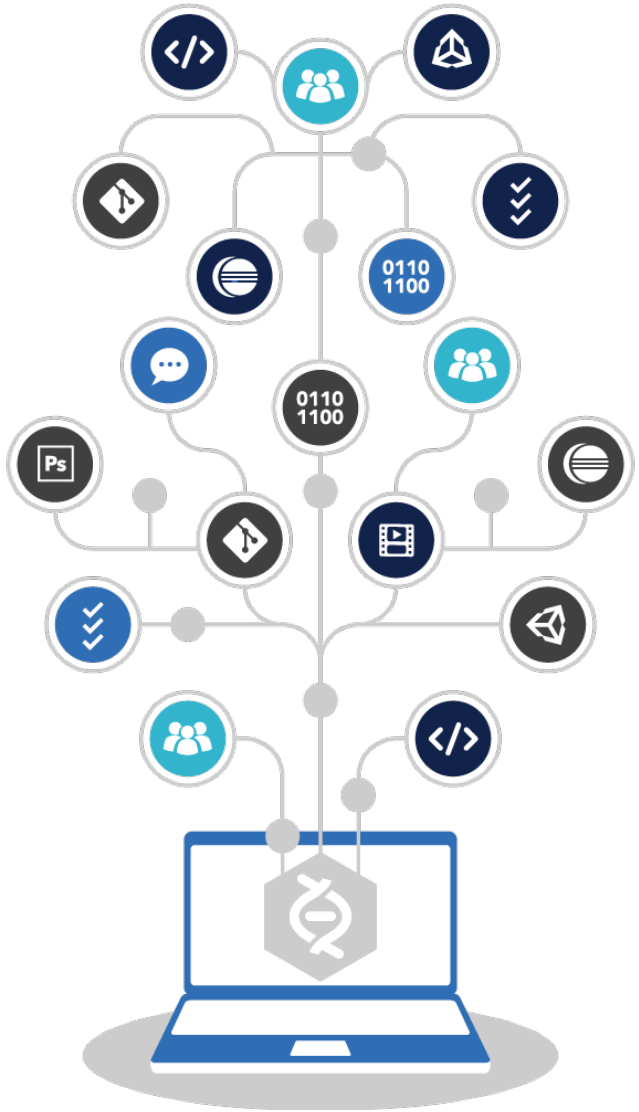
Why version control is important?

Restoring previous versions.



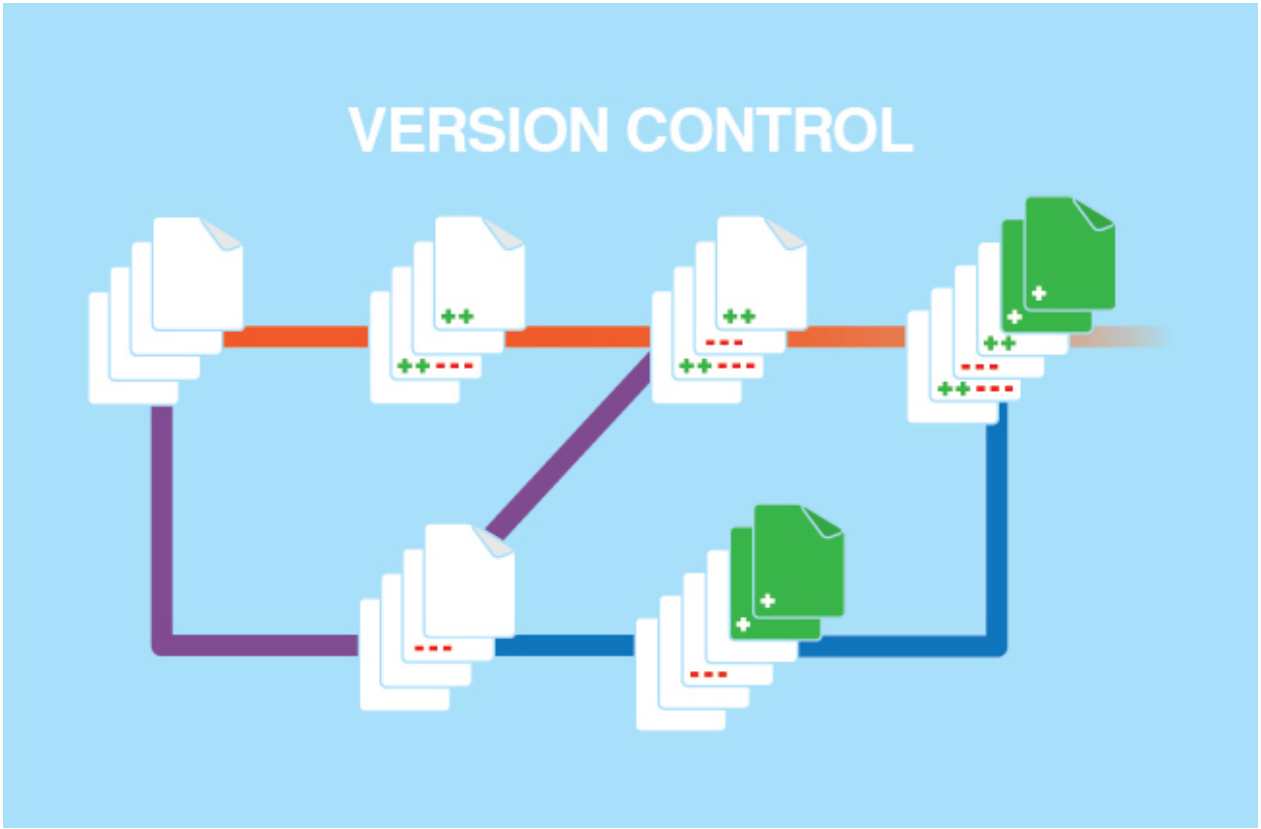
Why version control is important?

Collaborations (networking).



Why version control is important?

Save **time**.



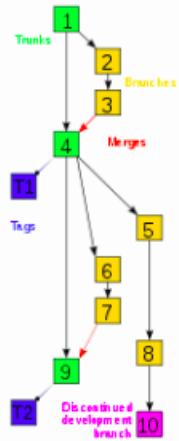
Version control software

V • T • E **Version control software** [hide]

Years, where available, indicate the date of first stable release. Systems with names *in italics* are no longer maintained or have planned end-of-life dates.

Local only	**Free/open-source**	RCS (1982) · SCCS (1972)
	Proprietary	PVCS (1985) · *QVCS* (1991)
Client–server	**Free/open-source**	CVS (1986, 1990 in C) · CVSNT (1998) · QVCS Enterprise (1998) · Subversion (2000)
	Proprietary	AccuRev SCM (2002) · ClearCase (1992) · *CMVC* (1994) · Dimensions CM (1980s) · *DSEE* (1984) · Endevor (1980s) · Integrity (2001) · Panvalet (1970s) · Perforce Helix (1995) · SCLM (1980s?) · Software Change Manager (1970s) · StarTeam (1995) · Surround SCM (2002) · Synergy (1990) · Team Concert (2008) · Team Foundation Server (2005) · Visual Studio Team Services (2014) · Vault (2003) · *Visual SourceSafe* (1994)
Distributed	**Free/open-source**	*ArX* (2003) · BitKeeper (2000) · *Codeville* (2005) · Darcs (2002) · *DCVS* (2002) · Fossil (2007) · Git (2005) · *GNU arch* (2001) · GNU Bazaar (2005) · Mercurial (2005) · Monotone (2003) · Pijul (2015) · *SVK* (2003) · Veracity (2010)
	Proprietary	*TeamWare* (1990s?) · Code Co-op (1997) · Plastic SCM (2006) · Team Foundation Server (2013) · Visual Studio Team Services (2014)
Concepts	Baseline · Branch · Changeset · Commit · Data comparison · Delta compression · Fork (Gated commit) · Interleaved deltas · Merge · Repository · Tag · Trunk	

Category · Comparison · List



What is Git?

Git is a distributed version control system for tracking changes in source code during the development of software.

Why use Git?

- **Popular and successful**
  - Active development
  - Fast
- **Distributed**
  - Work online and offline
  - Collaborate with large groups
- **Tracks any type of file**
  - Works best with text
- **Branching**
  - Smarter merges

What is GitHub Inc.?

GitHub is a web-based hosting service for version control using **Git**.

GitHub

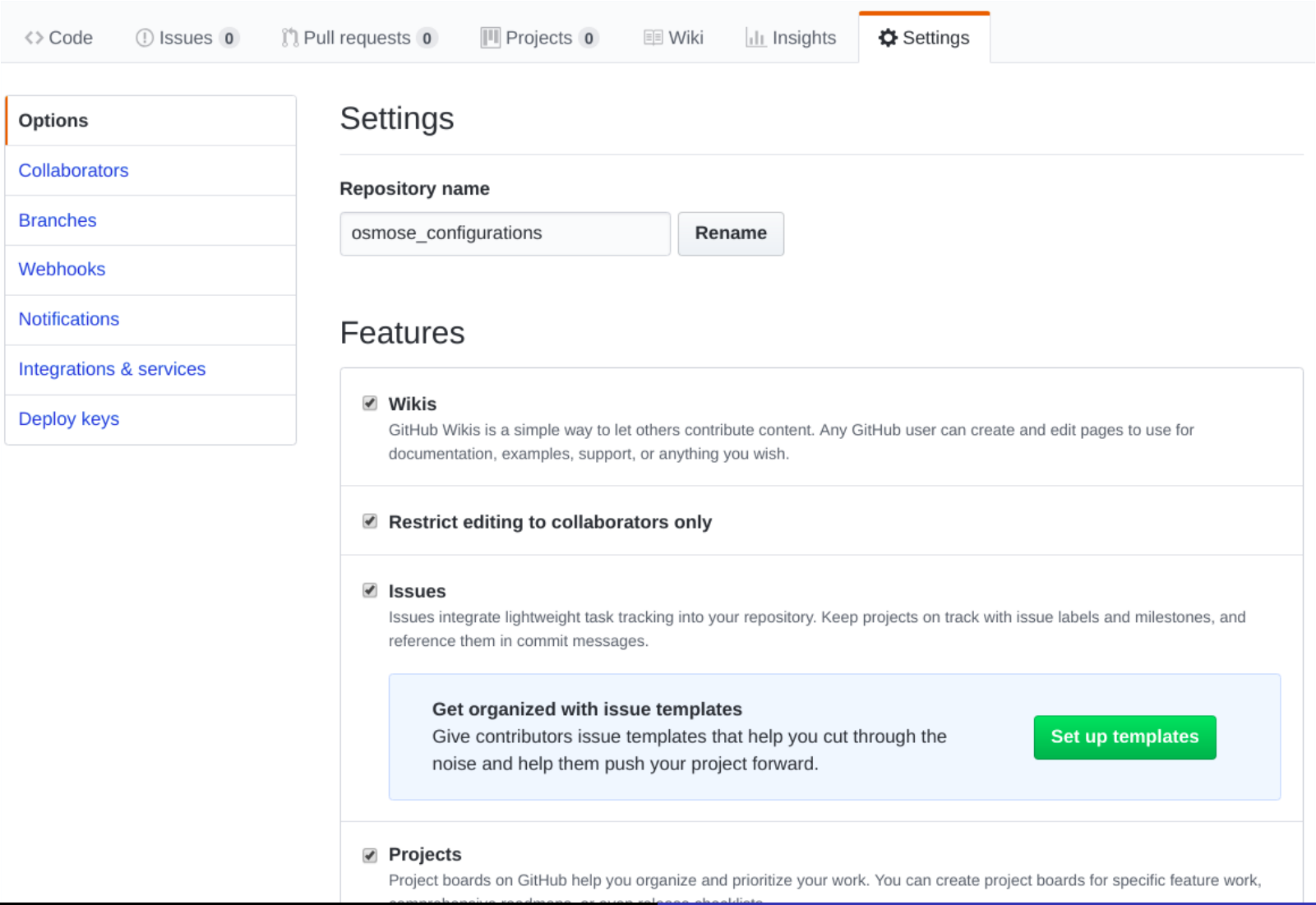
Download logo

Download mark

Download Octocat

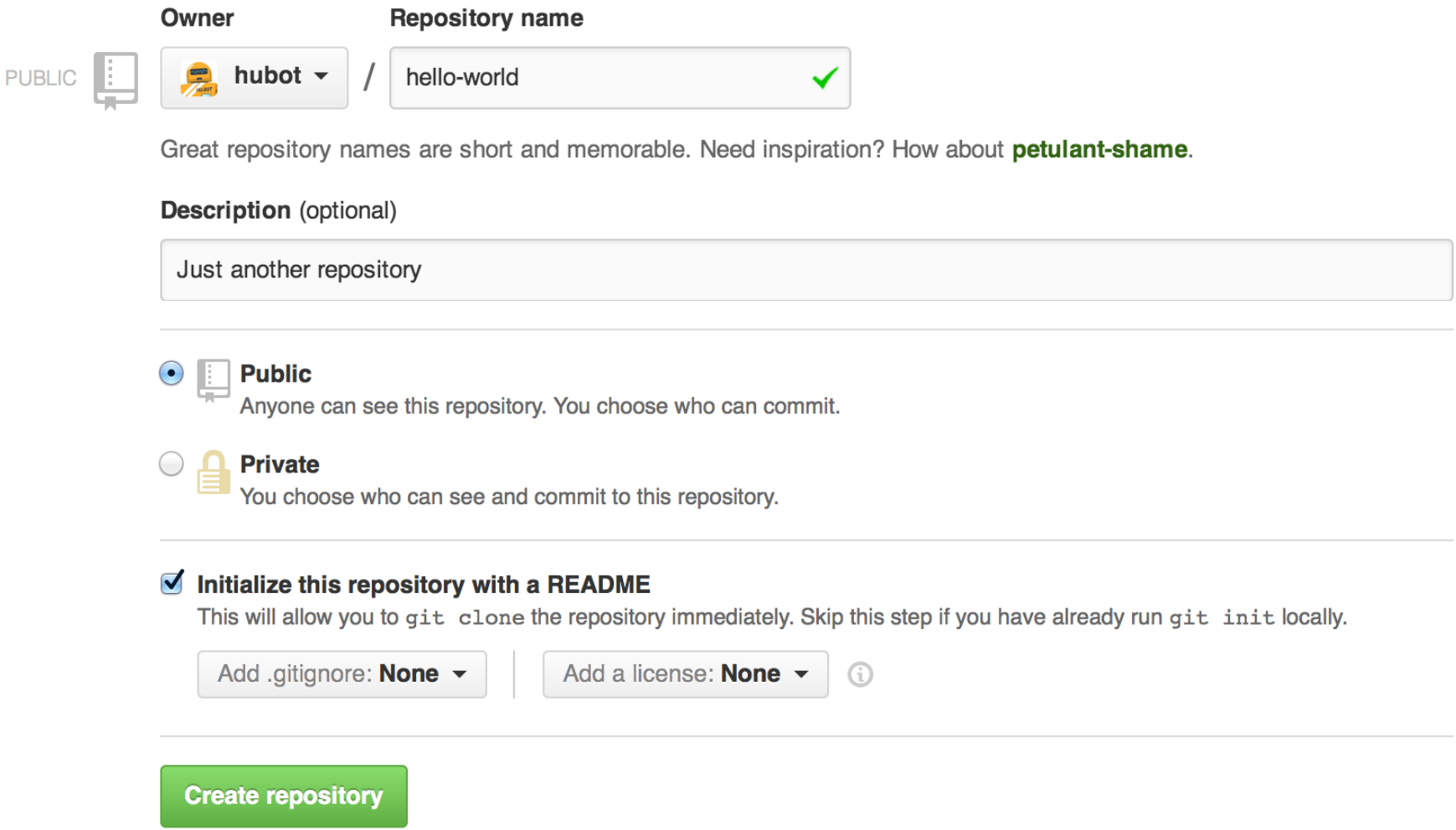
GitHub Inc.

- Access to the control and collaboration features for every **project**.



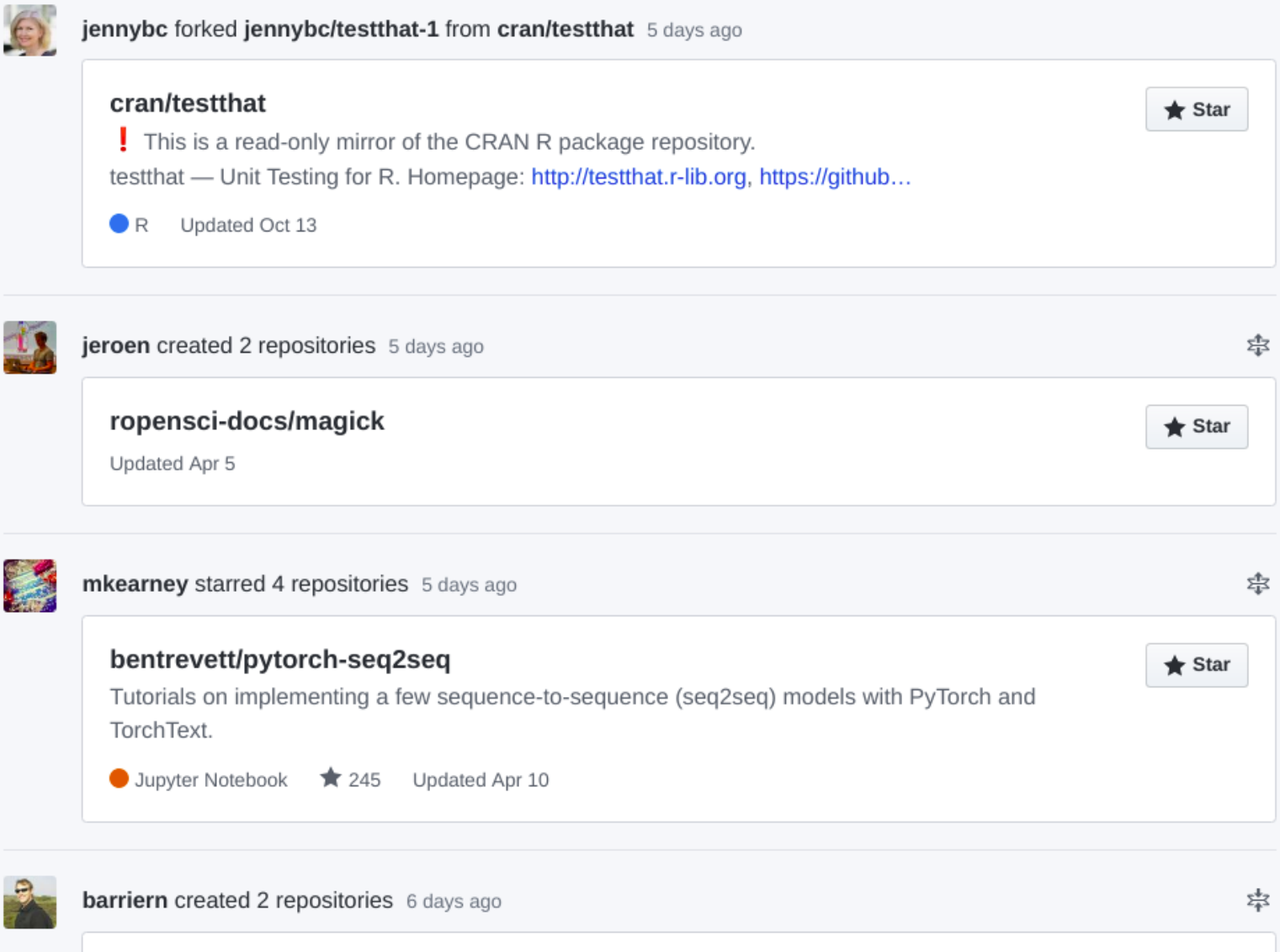
GitHub Inc.

- Work with public and private **repositories**.



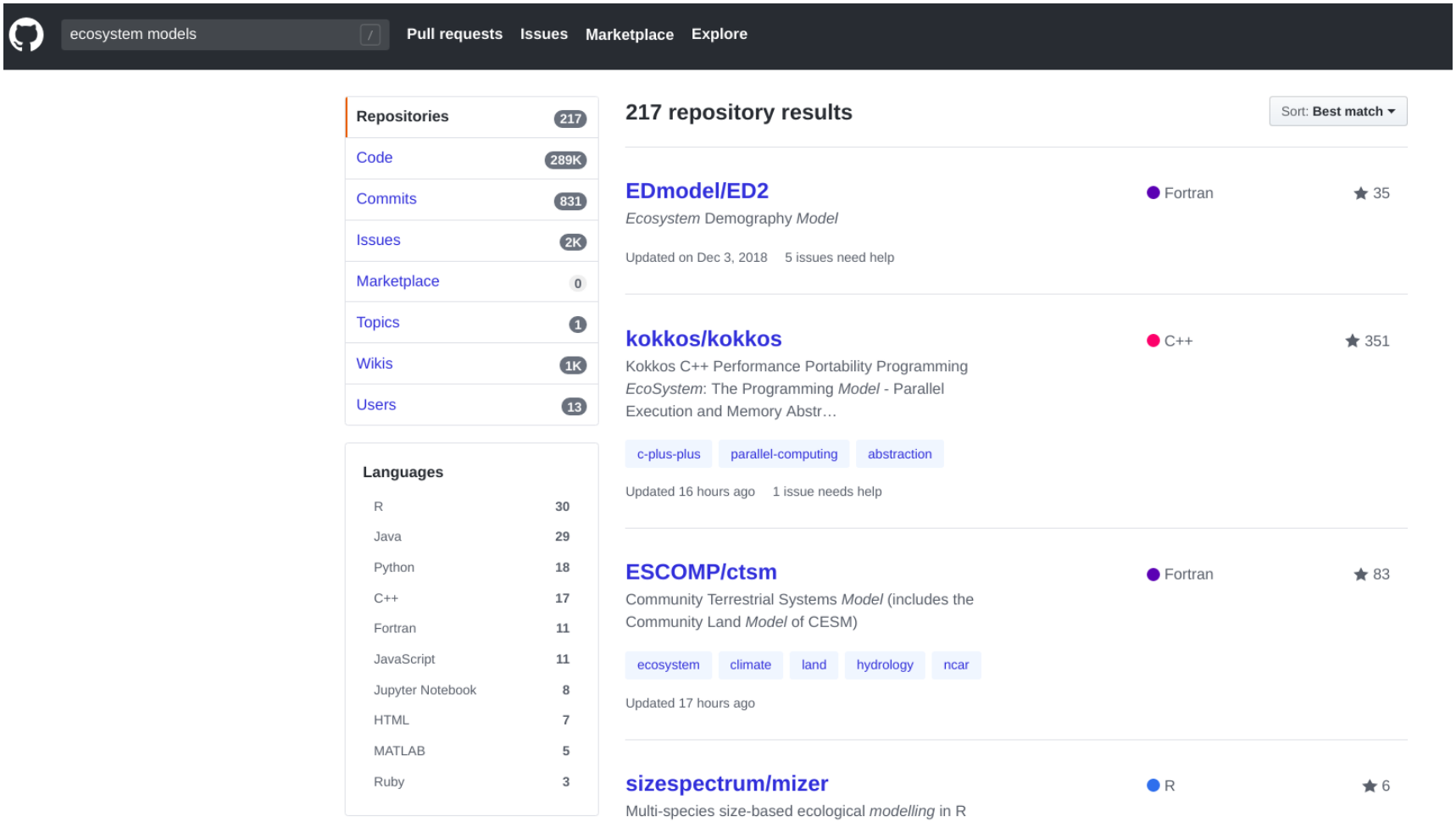
GitHub Inc.

- Develop a **networking**.

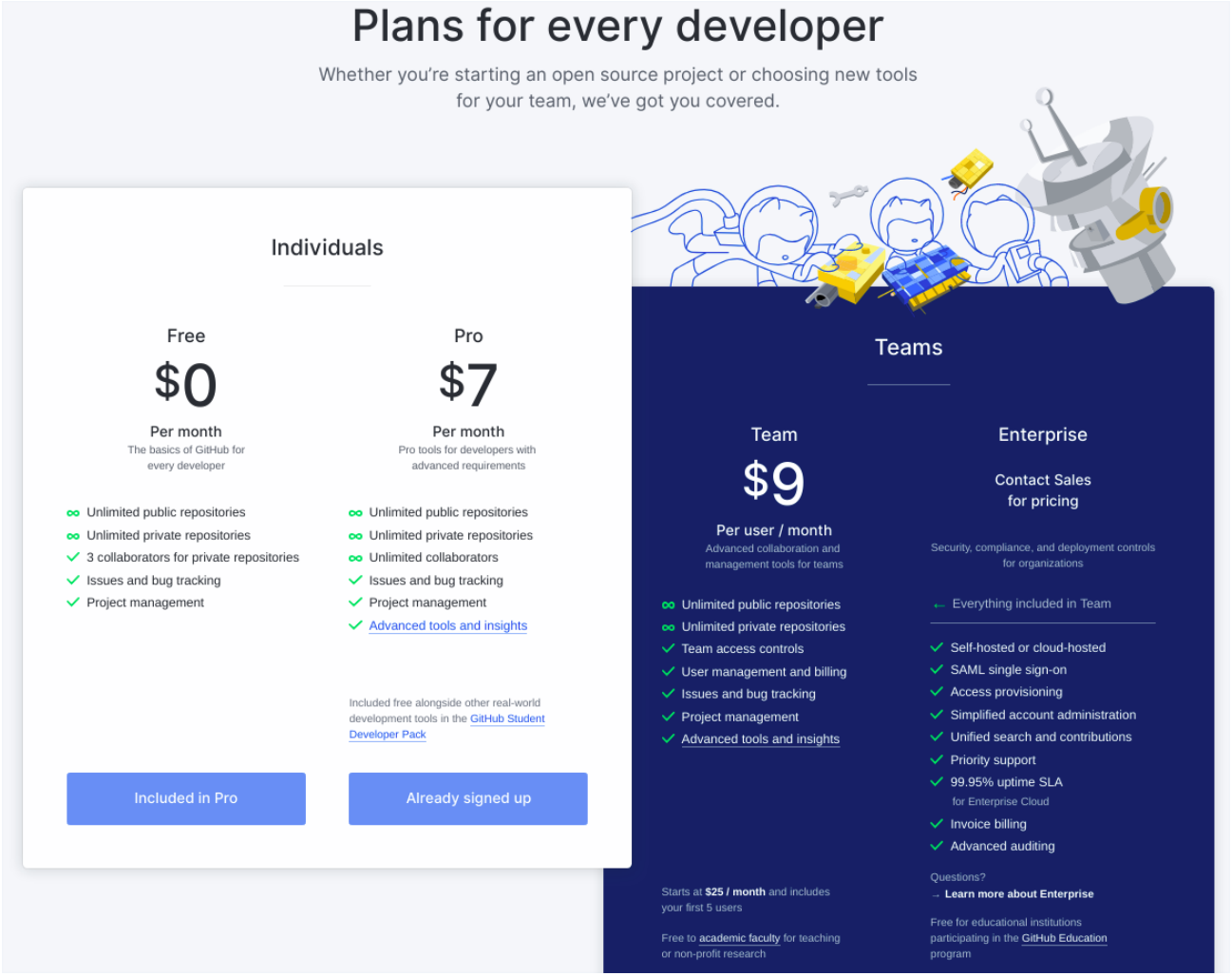


GitHub Inc.

- Source of information.



- **Plans** for enterprise, teams, pro and free accounts.



GitHub Inc.

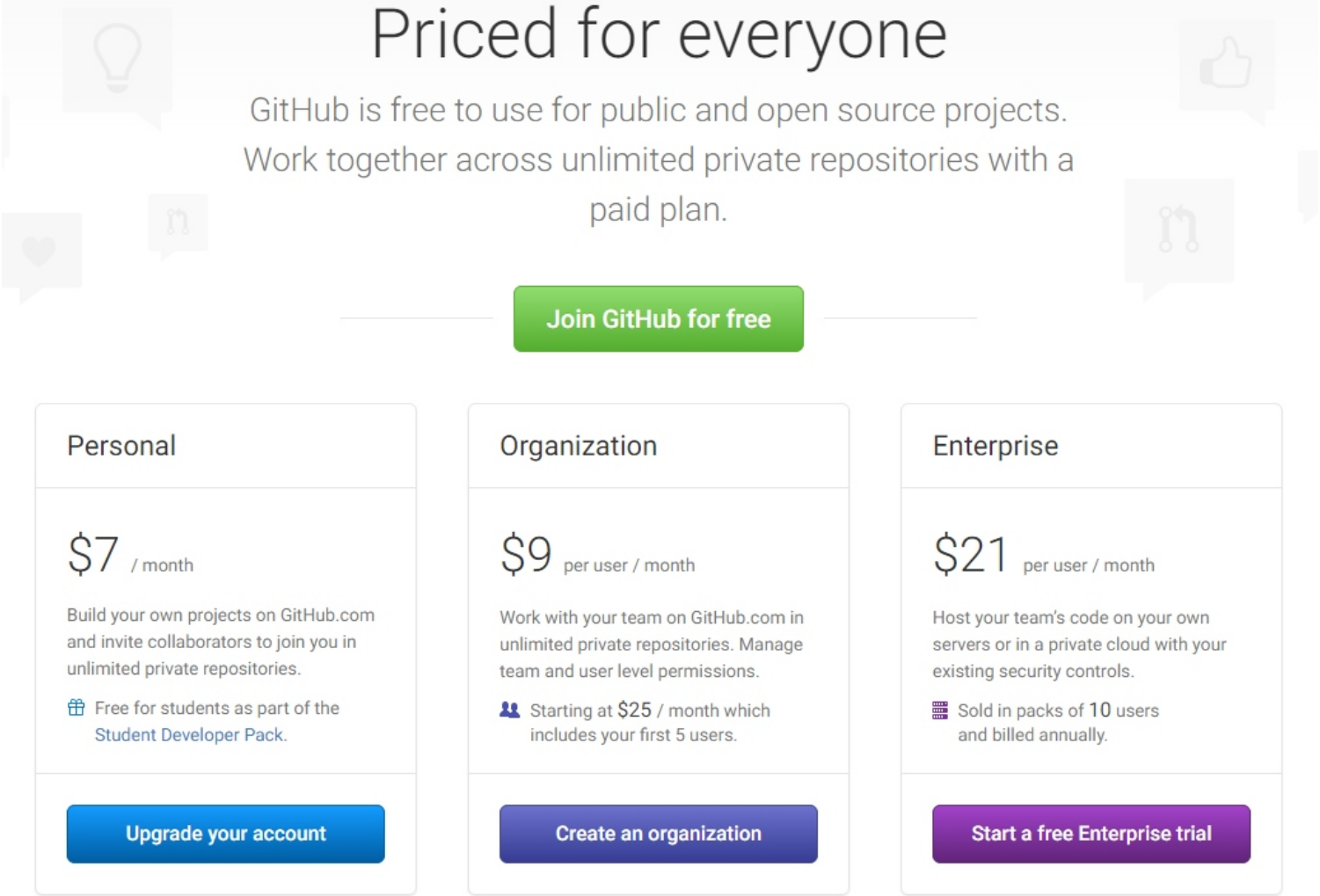
- Is the **largest** host of source code in the world! *(28 million users, 57 million repositories (28 million public) - June 2018)*.

Register a GitHub account

- Create an account in GitHub is free!
- Free private repositories
  - Students, faculty, and educational / research staff: GitHub Education.
  - Official nonprofit organizations and charities: GitHub for Good.

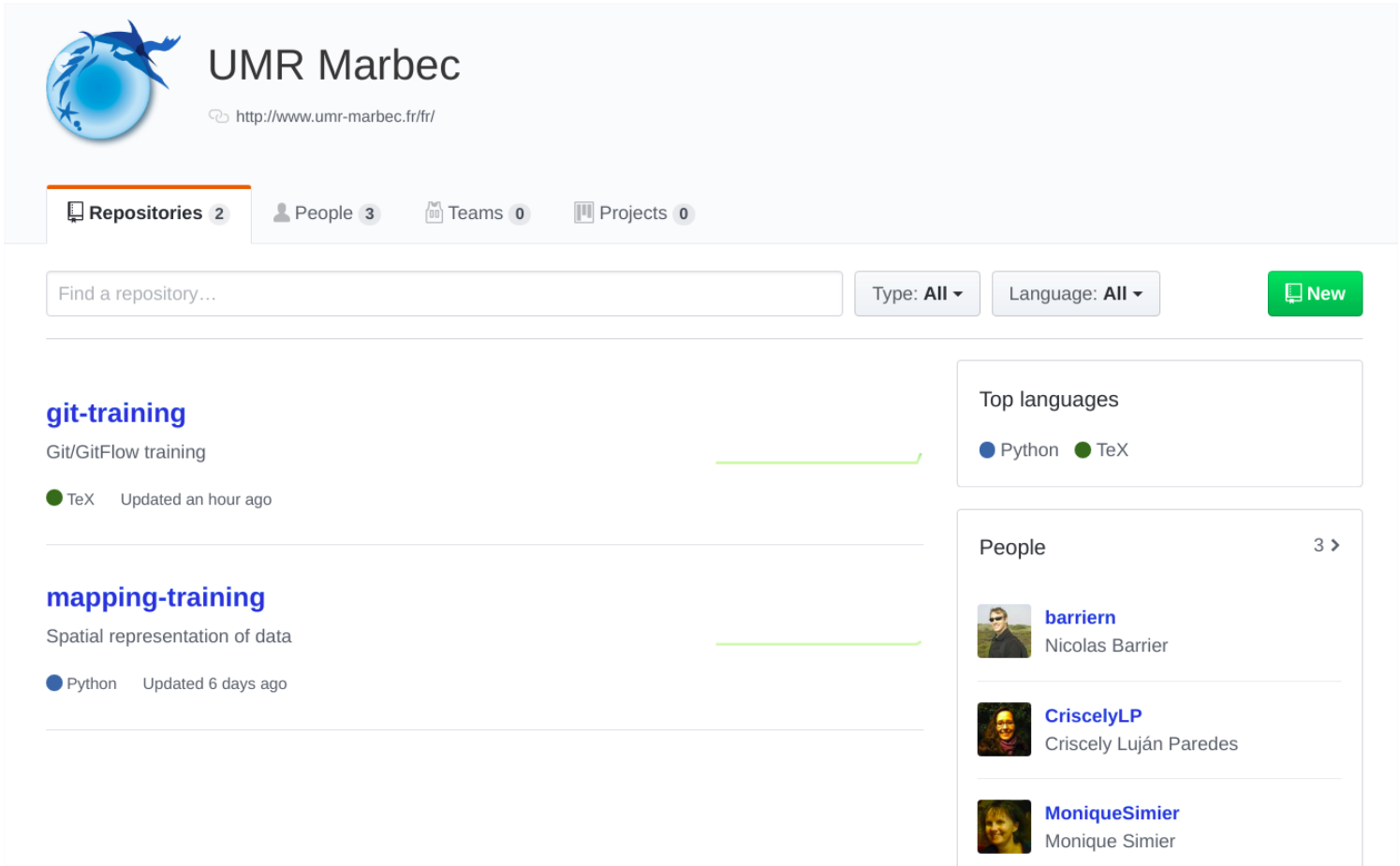
Register a GitHub account

- Pay for private repositories
  - Individual cost is 7 dollars per month: GitHub Pricing.



Marbec in GitHub

All the materials of Pole Modelisation's technical "workshop" are now stored in an institutionnal GitHub account: https://github.com/umr-marbec.



Institutionnal repositories

GitHub is a private US company. There are also *institutional* repositories on which Git can be used:

- Sourcesup: this is a Renater platform (login possible from any French research institute or through CRU accounts)
- Forge Ifremer: very close to SourceSup (Ifremer extranet account required)
- IRD GitLab: GitLab IRD platform (IRD account required).

However, the projects hosted on these repositories may have less visibility...

Git clients

Git and Git client **are not** the same! Like R and RStudio is not the same thing!

Git client:

- IDE (Integrated development environment)!
- Make the experience more pleasant providing a richer visual representation.

Some example of Git clients:

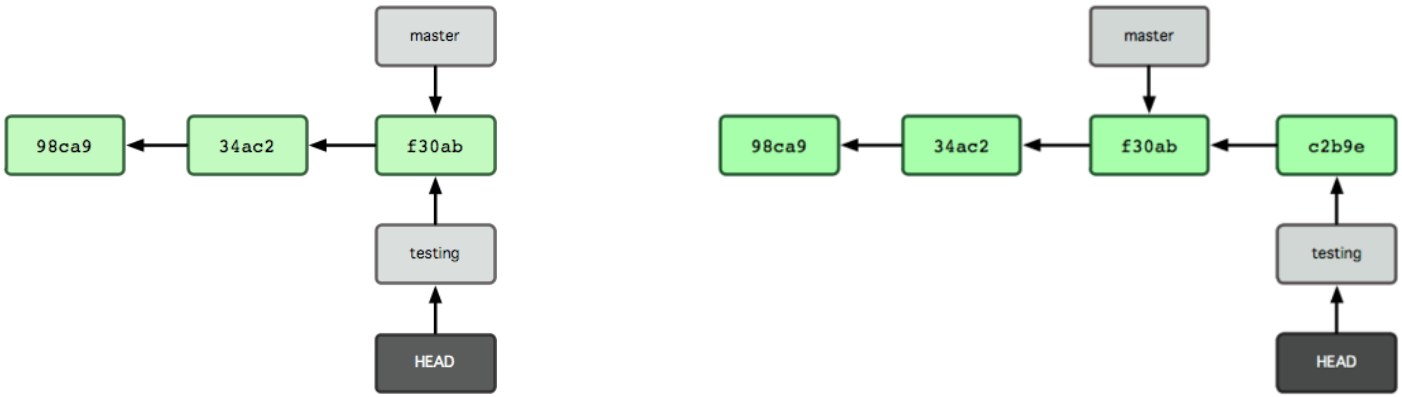
- SourceTreen
- GitKraken
- GitUp
- SmartGit
- git-cola
- RStudio

Git branches

One main advantage of Git is the use of *branches*, which allow multiple developments of the same code at the same time.

Definition

A branch in Git is simply a lightweight movable pointer to one of thes commits.



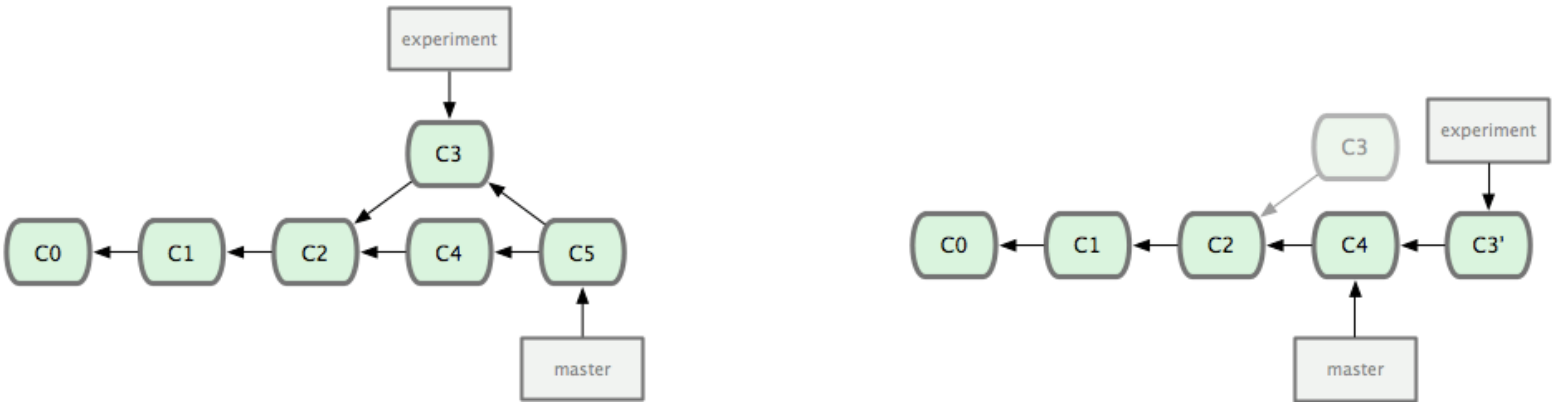
In this example, the `master` branch points to the `f30ab` commit, while the `testing` branch points to the `c2b9e` one. `HEAD` points to the active branch (here, `testing`).

Source: https://git-scm.com/book/en/v1/Git-Branching-What-a-Branch-Is

Merging branches

To merge a branch (for instance a feature branch) to another branch (for instance the main one), several options are offered.

- `merge`: Three-points branch (common ancestor + tips of the two branches)
- `rebase`: Compresses all the changes into a single "patch."



(a) Merge (b) Rebase

Figure: Merging versus rebasing

Source: https://git-scm.com/book/fr/v1/Les-branches-avec-Git-Rebaser

Git workflows

There are several ways to use Git branches (we talk about **workflows**).

- *Centralized workflow*: one main branch, everyone commit in the same place.
- *Feature Branch Workflow*: developments are made in dedicated branches (feature branches), which are regularly merged into the master one.
- **Gitflow Workflow**: Strict branching model designed around the project release.

Source: https://www.atlassian.com/git/tutorials/comparing-workflows

GitFlow branches

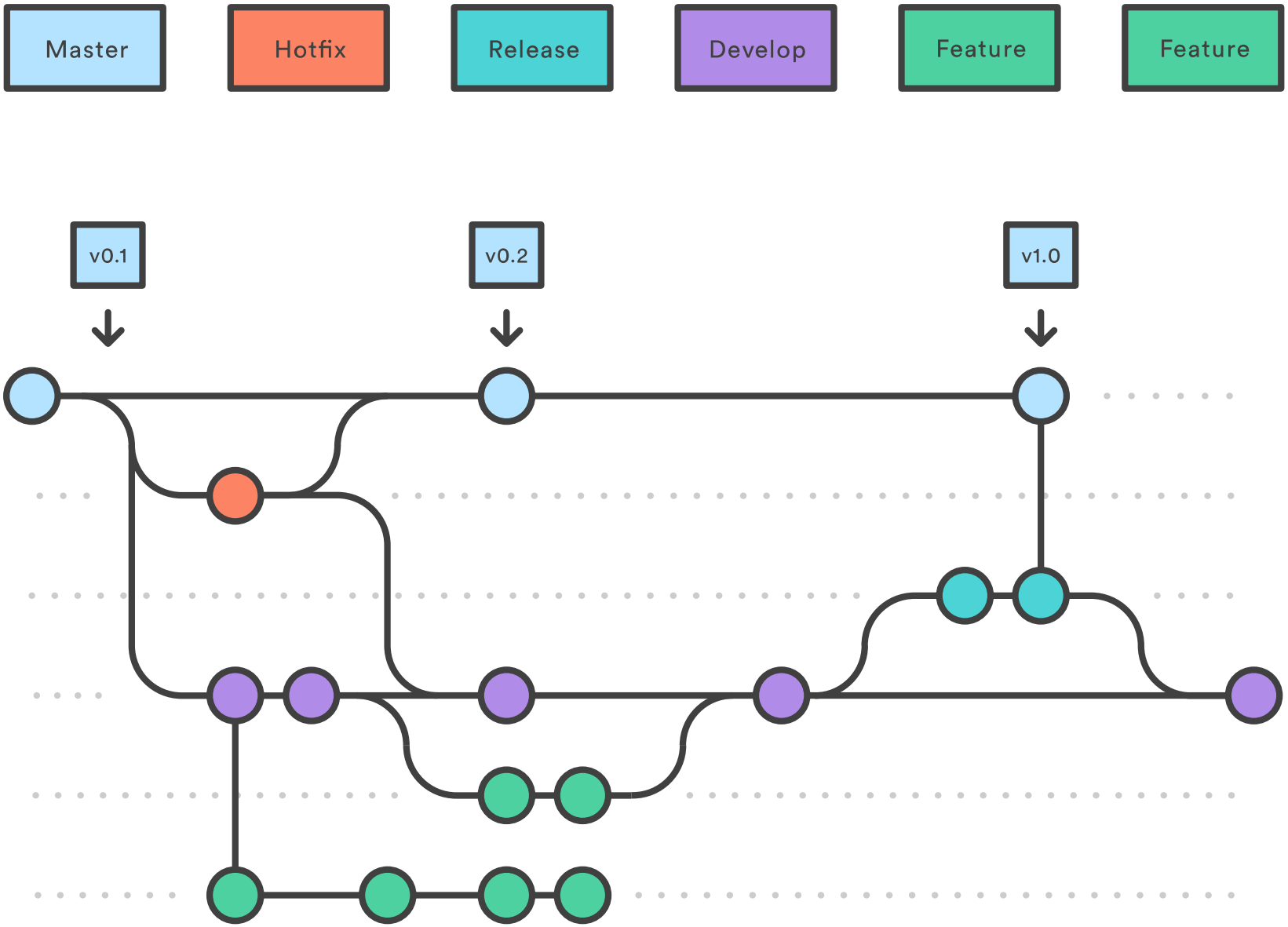
GitFlow workflow contains two main branches:

- `master`: official release history. Branch which is shared to the world!
- `develop`: integration branch for features

It also contains additional temporal branches:

- `feature`: feature branches (one for each new feature to add to the code)
- `release`: branch created when enough features have been added (new version of the code) to develop
- `hotfix`: branch for maintenance and bug correction of the production release

In summary...



Source: https://www.atlassian.com/git/tutorials/comparing-workflows

Thanks for your attention

Now, let's crack on it!

Source: https://www.pinterest.fr/pin/447263806724736402/