Git flow

Version control

- Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
- It allows you to
  - revert files back to a previous state,
  - revert the entire project back to a previous state,
  - review changes made over time,
  - see who last modified something that might be causing a problem

What is Git ?

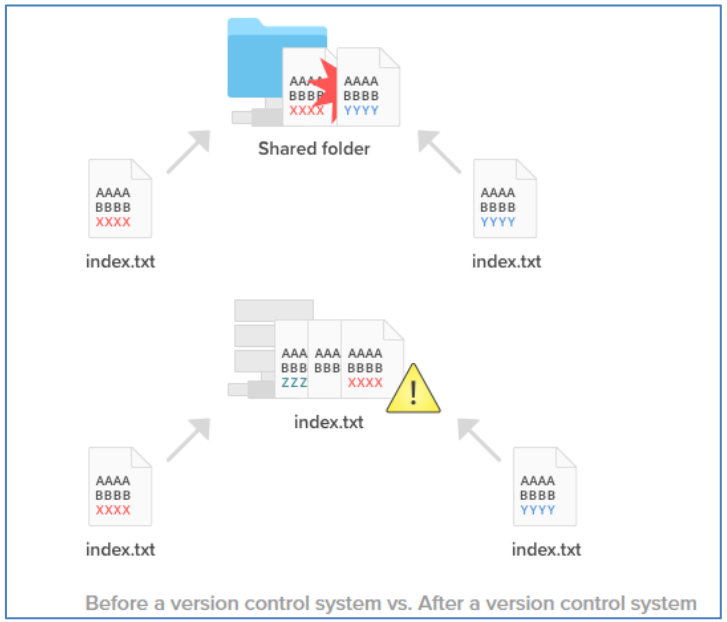
- Git is a version-control system for tracking changes in computer files and coordinating work on those files among multiple people.
- Git is a ***Distributed Version Control System***
  - So Git does not necessarily rely on a central server to store all the versions of a project's files.
  - Instead, every user "clones" a copy of a repository (a collection of files) and has the ***full*** history of the project on their own hard drive.
- It is basically the history tab for your code editor
- It helps you
  - ***keep track of the changes*** you make to your code
  - ***synchronize code*** between multiple people

Git installation

- Download and install git for windows
- Download and install SourceTree (or any other similar tool)

Git repo

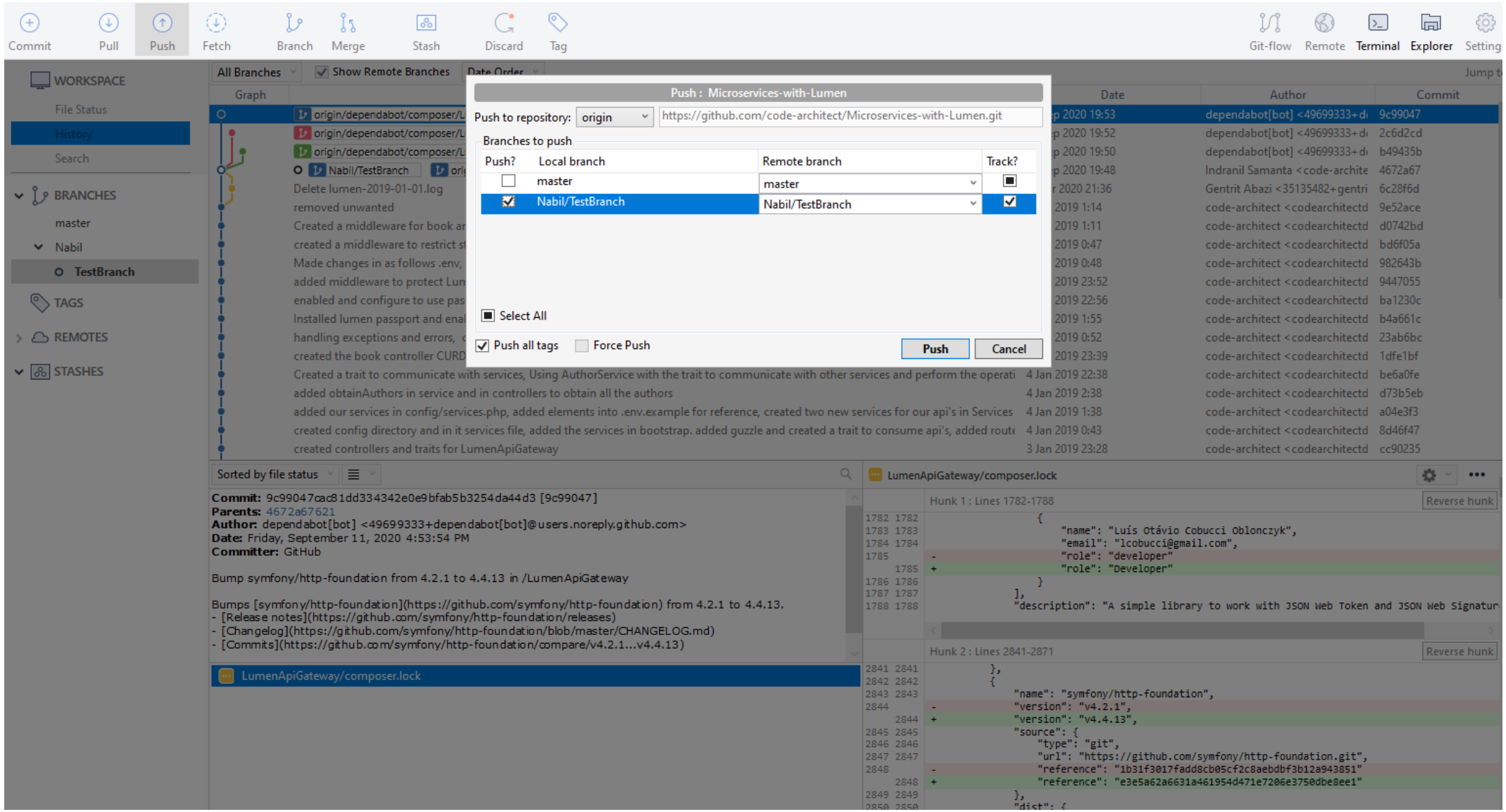
- A **repository** a.k.a. **repo**
  - is nothing but a collection of source code
  - It holds all the commits—a snapshot of all your files at a point in time—that have been made



Git branches

- Git branches
  - are a part of your everyday development process
  - Git branches are effectively a pointer to a snapshot of your changes
  - When you want to add a new feature or fix a bug—no matter how big or how small—you create a new branch to encapsulate your changes
- There are two types of branches
  - remote : hosted on a remote, or off-site, server that is shared among multiple team members
  - local : hosted on a local machine for an individual user

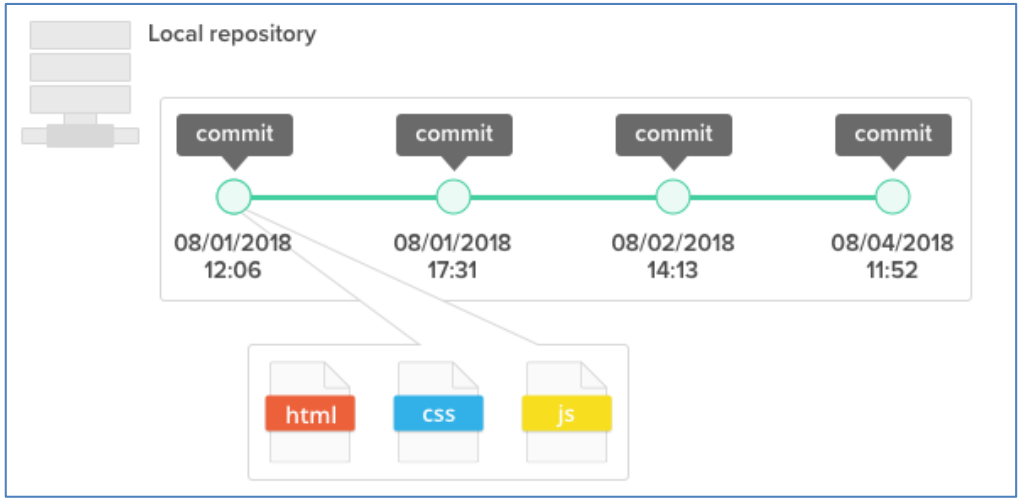
GIT BRANCHES



GIT BRANCHES

Commit

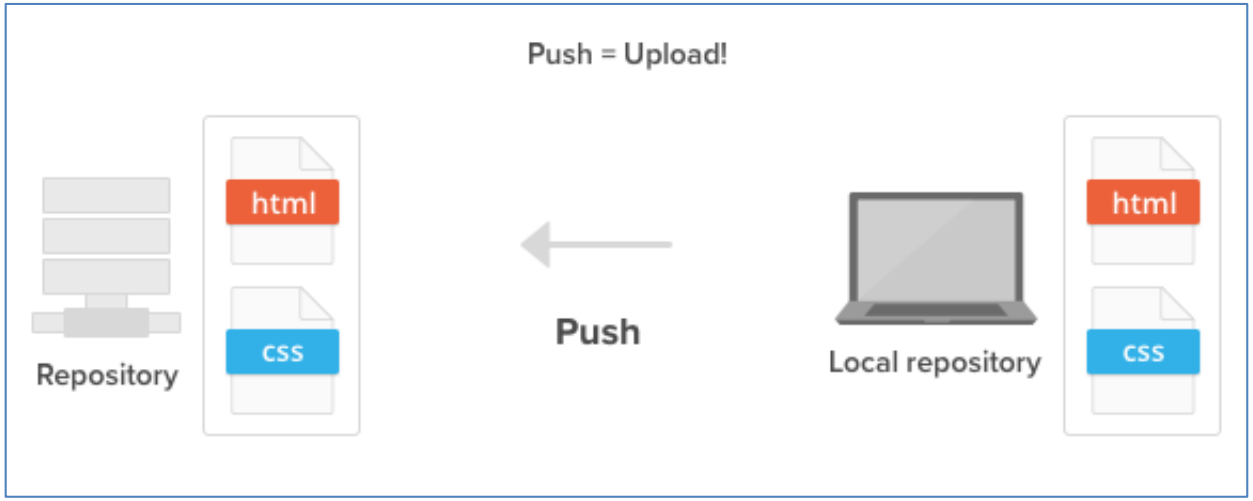
- Is used to save changes to local repository
- It enables you to record file changes in the repository's Git history. So you can revert to an old commit if needed.



Git branches

Push

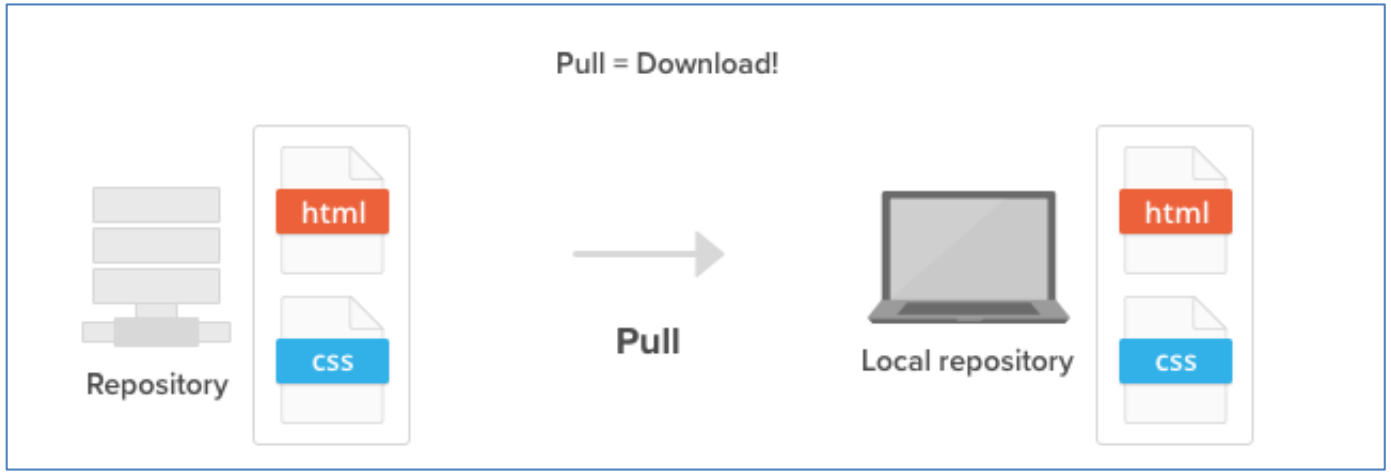
- In order to start sharing changes with others, you have to push them to a remote branch using the "push" command.
- This will cause the remote repository to update and synchronize with your local repository



Git branches

Pull

- Whenever somebody pushes their changes to the shared remote repository, your local repository becomes out of date.
- To re-synchronize your local repository with the newly updated remote repository, simply run the git pull operation.



GIT BRANCHES

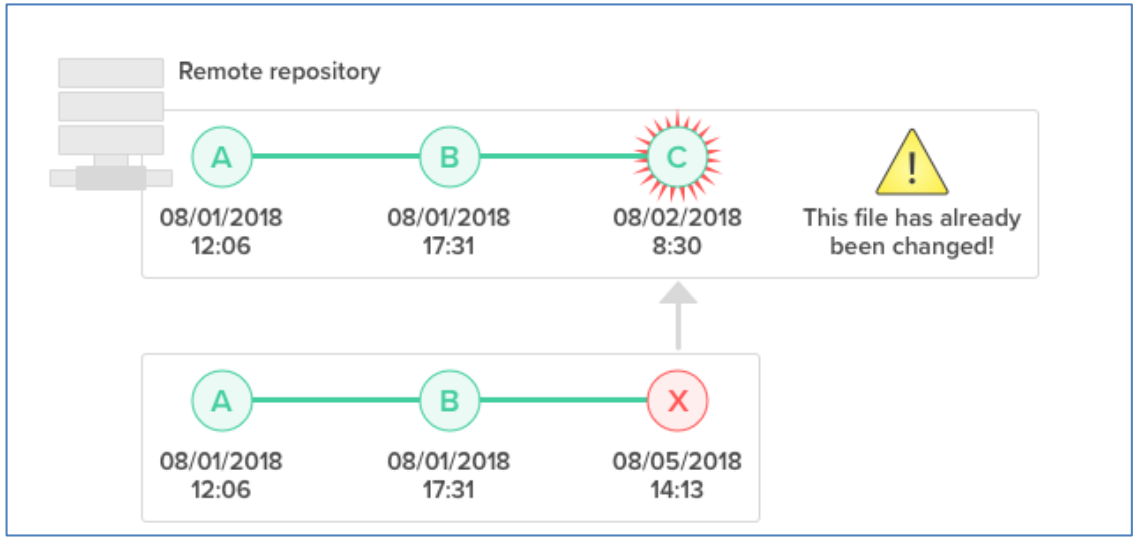
Merge

- Your push to the remote repository will be rejected if your local repository is out of date, possibly because there are some updates on the remote repository that you do not have locally yet.
- If that is the case, you'll have to use the git merge command to grab the latest change from the remote repository before you are allowed to push

GIT BRANCHES

Merge

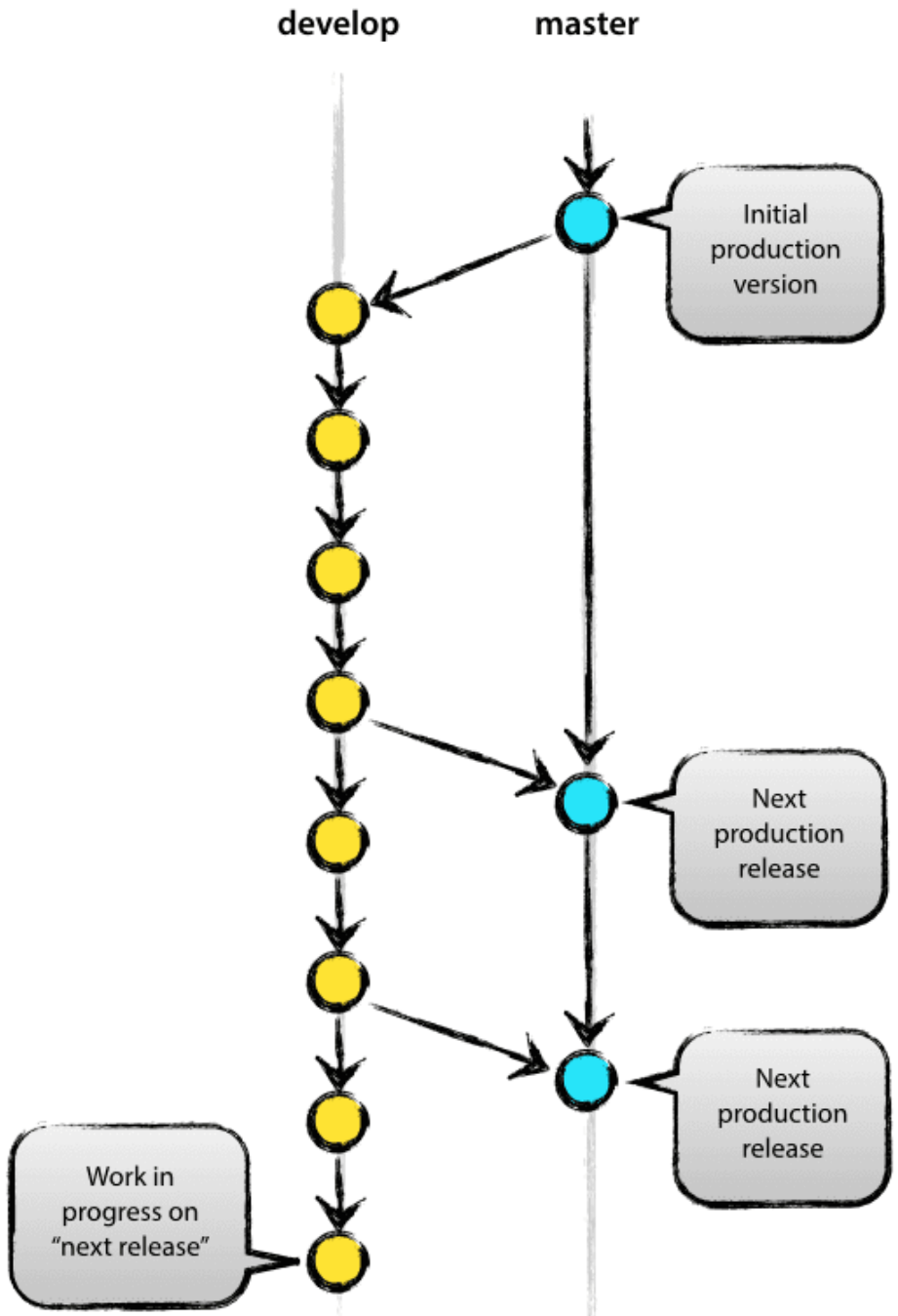
- You can also pull the latest change, fix any conflict, then push your code to remote branch



GIT BRANCHES - TYPES

The different types of branches :

- master
  - production releases
- develop
  - used mainly for integration and regression testing
  - developers push their changes to and tester can test



https://nvie.com/posts/a-successful-git-branching-model/

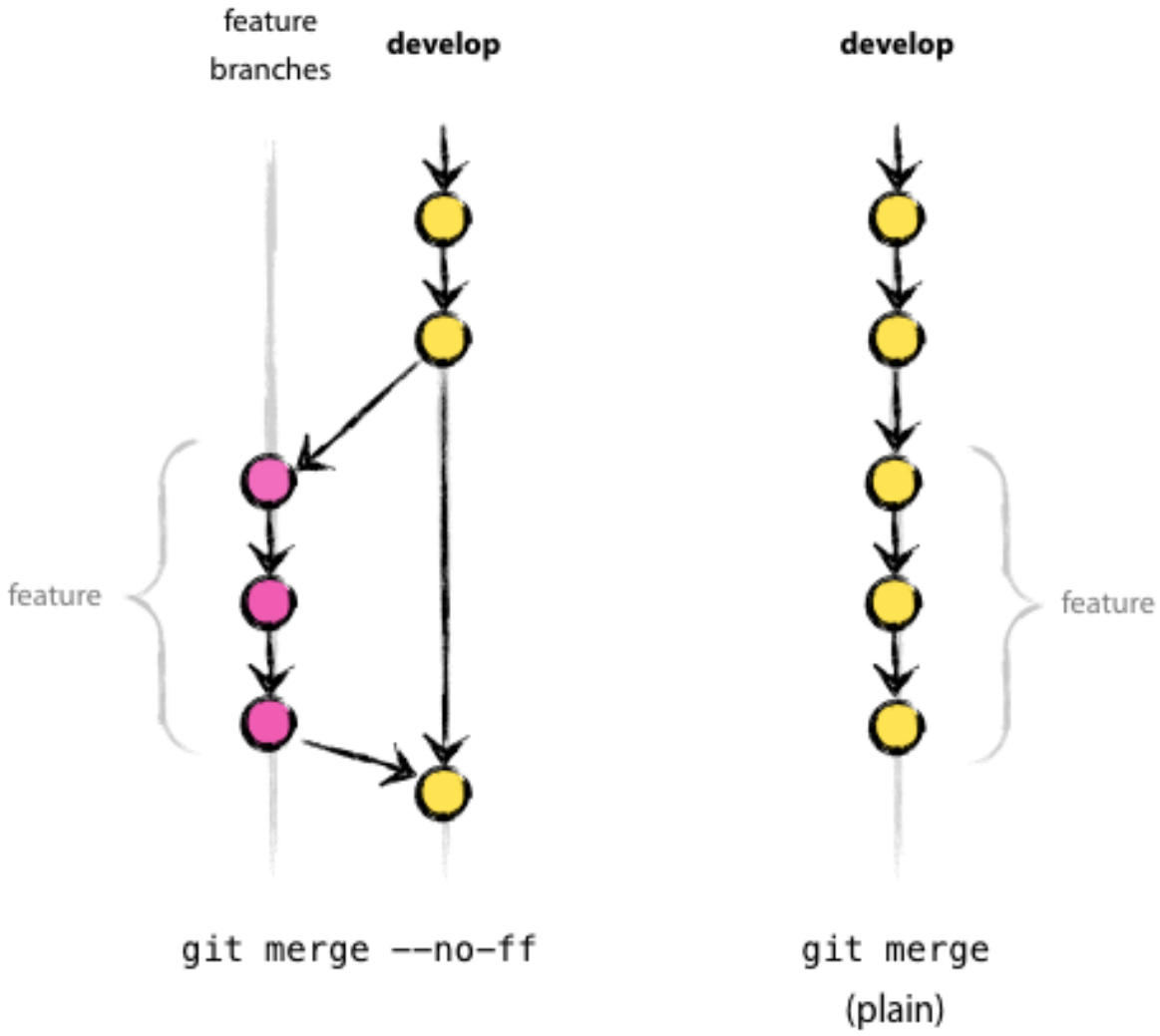
GIT BRANCHES - TYPES

Feature branches (feature-*)

- for developing features
- branched from develop

Release branches (release-*)

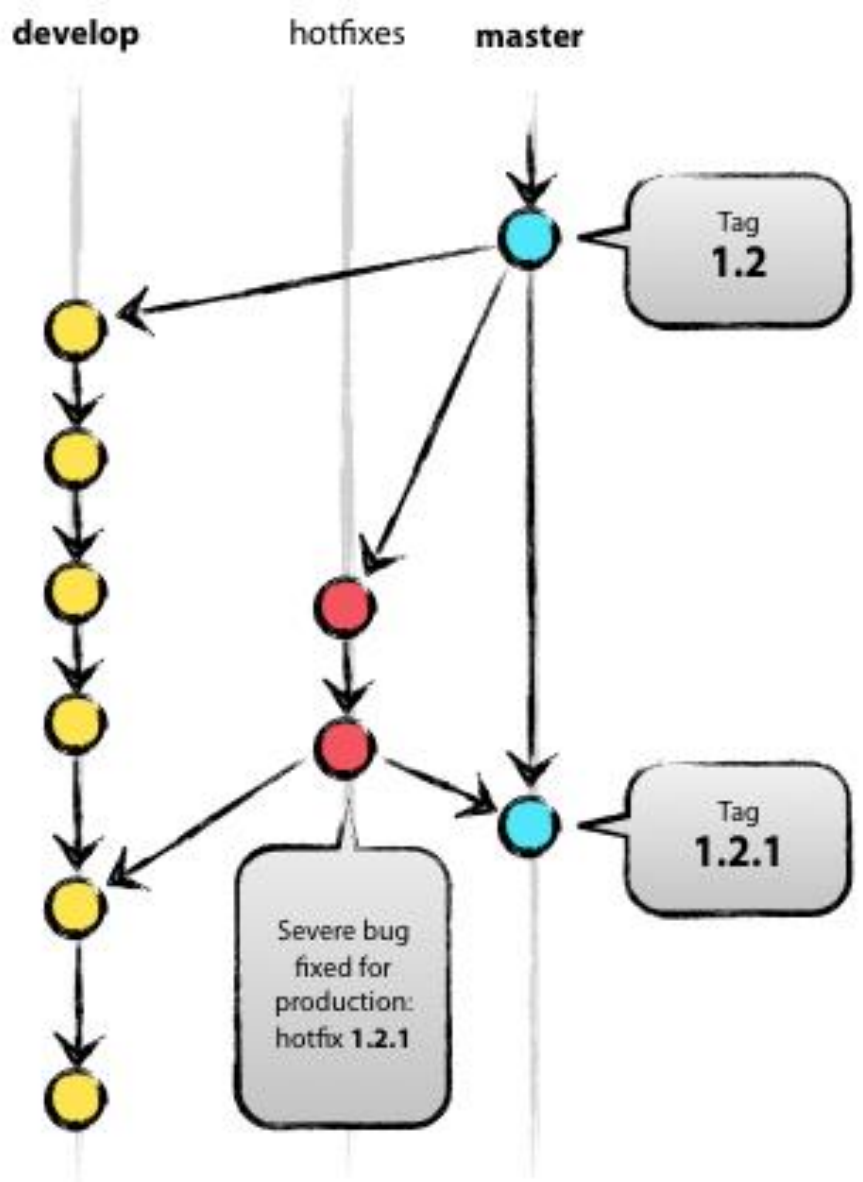
- for finalizing a major/minor release
- branched from develop



https://nvie.com/posts/a-successful-git-branching-model/

GIT BRANCHES - TYPES

- Hotfix branches (hotfix-*)
  - are like release branches but for production bugs
- Support branches (support-*)
  - for applying patches to old release versions
  - branched from master



https://nvie.com/posts/a-successful-git-branching-model/