GIT STRUCTURE

Git structure

files ↑

- Design a git branch structure to use with the coding repository
- Research on what a good branch structure for our project looks like
- create a visualization of the structure

Git branch structures

1. Git flow branching
2. trunk based development
3. Git hub flow
4. Forking strategy
5. Release branching
6. Environment branches
7. three flow branches

Git → organize files

Branching → enables working on different versions of the same codebase simultaneously

Ways of structure Git branches:

- Gitflow branching model :
  defines a set of rules for creating and merging branches
  2 primary branches
  - master : store the official release history
  - develop : integration branch for features

feature branches: develop new features or changes to existing features. Are created of the develop branch and merged back into it when the feature is completed.

Release branches: used to prepare for a new release. Are created of the develop branch and merged into the master and develop branches when the fix is complete

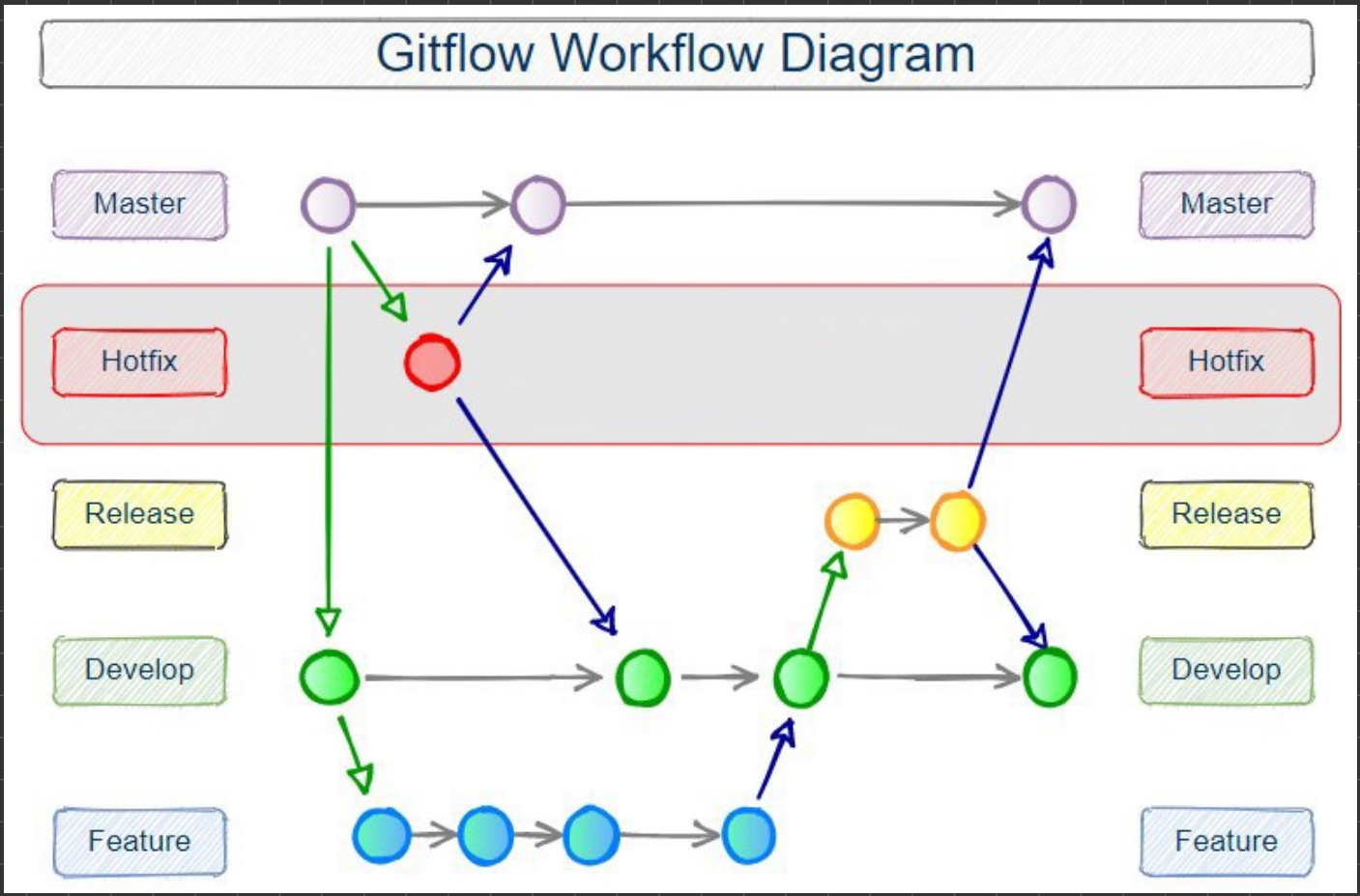
Hotfix branches: used to fix critical bugs in production code. Are created of the master and merged back into the master and develop branches when the fix is complete.

= organize system

pushing in git flow: updating the codebase that other developers can access and work on

= folders

repository: directory that stores digital objects for an archive



https://youtu.be/WQuxeEvaCxs?si=vVENkIsmfkoNSZby

↑
important for the code

https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Gitflow-release-branch-process-start-finish

Why use it?

- Expect to do a single release at the end of the sprint
- manual testing of the code before it goes to production

Pros

- allows to work on multiple releases in parallel
- each release is tagged and individually tested
- allows multiple developers to work on the same feature

- allows for jumping between work for current and future releases

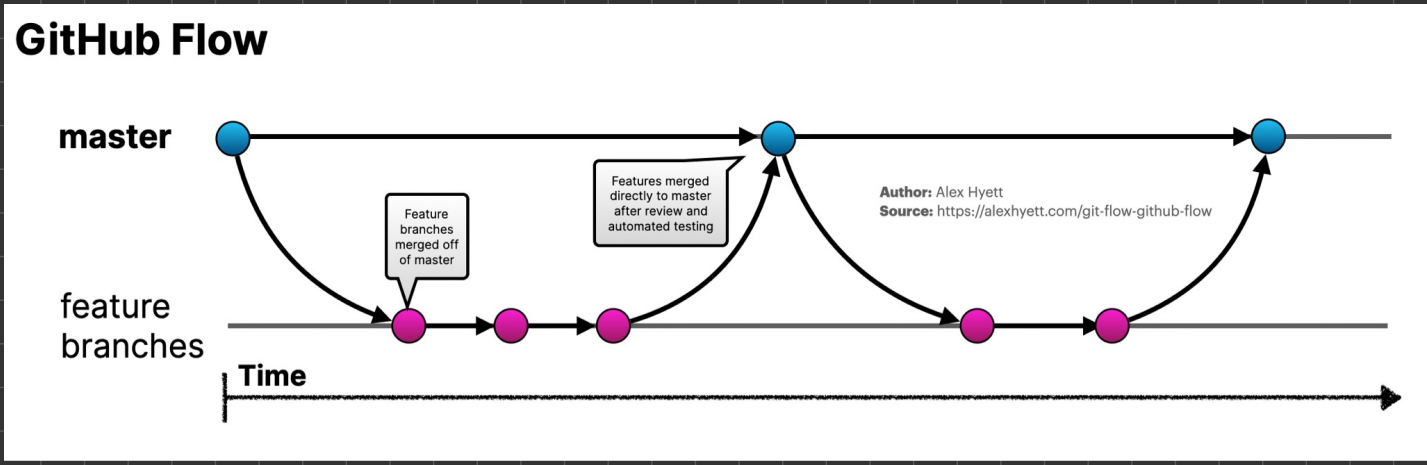
cons

- not suitable for continuous delivery or continuous deployment
- many branches to maintain
- can lead to a technical debt build-up.

https://www.alexhyett.com/git-flow-github-flow/

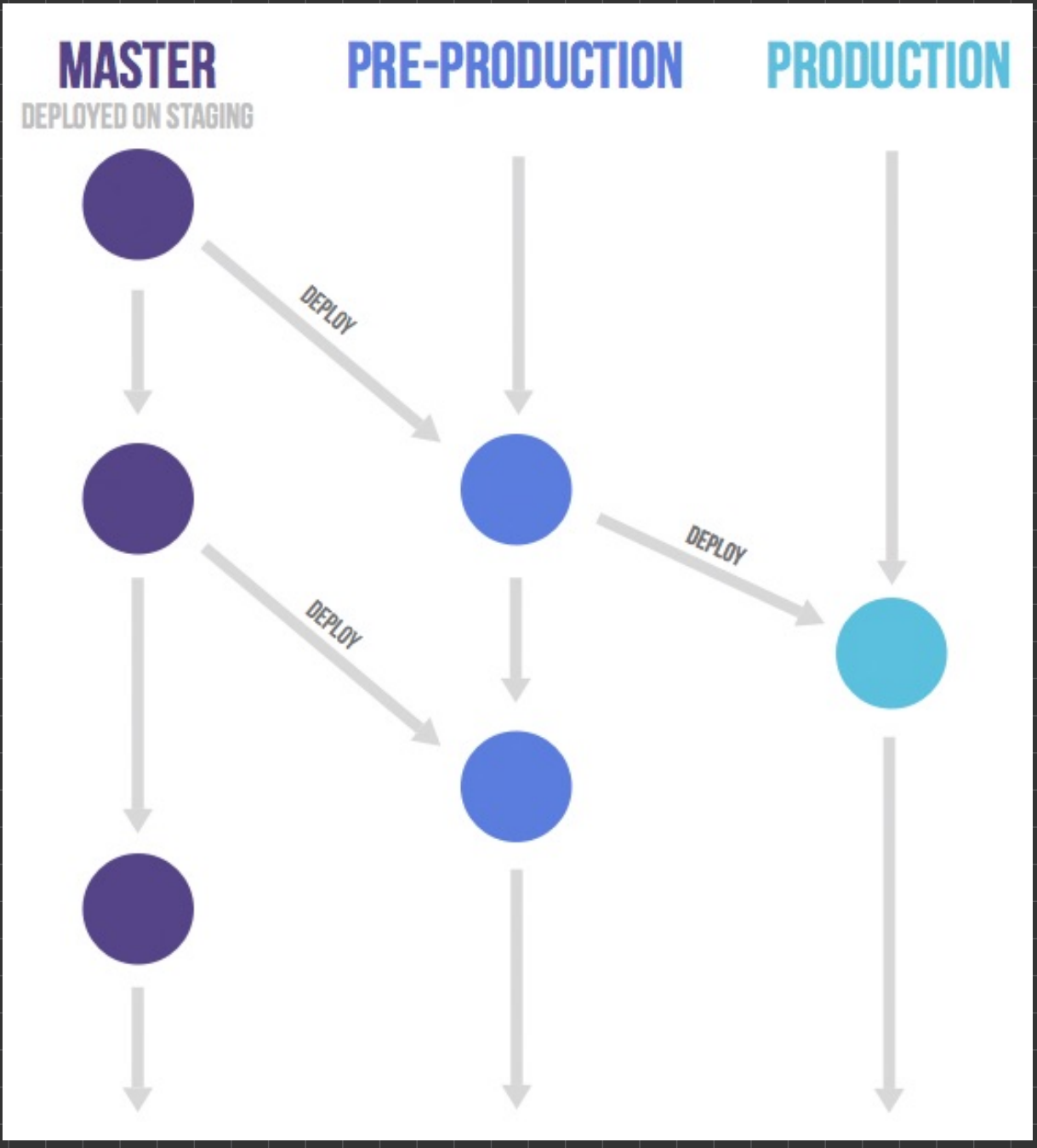
↑ important!

Github flow



developers work on a single branch (main/master). Developers create feature branches to work on specific changes. Once the feature is complete it is merged back into the main branch through a pull request

Git lab flow

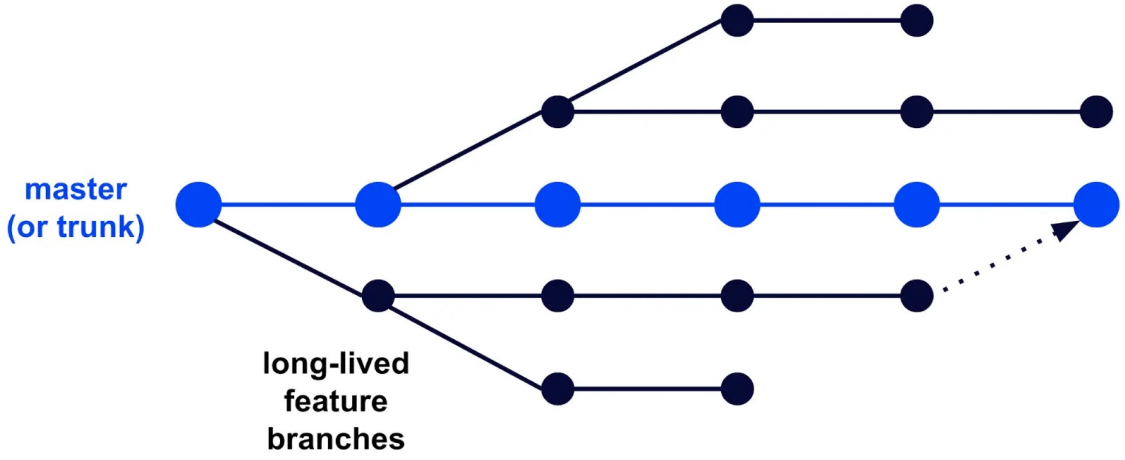

MASTER
DEPLOYED ON STAGING
PRE-PRODUCTION
PRODUCTION
DEPLOY
DEPLOY
DEPLOY

- Trunk-based Development model: emphasizes continuous integration and delivery by keeping the main branch stable at all times. Developers use feature flags or toggles to control the visibility of new features. Developers work directly on the main branch,

https://www.optimizely.com/contentassets/569ac3ee0b124da19a5ac9ea2e8b2b4d/trunk-based-development.png

making smaller and frequent commits

Feature-branched development



merging is difficult on a longer-lived feature branch

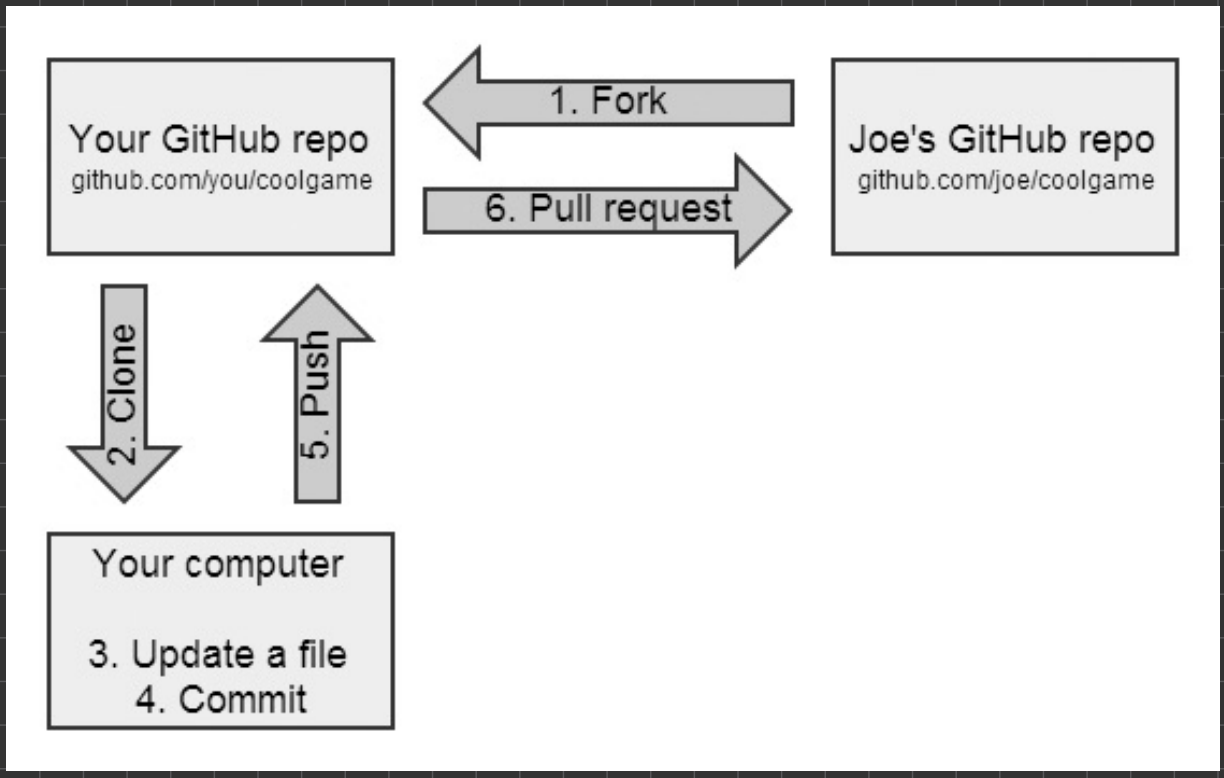
Forking strategy

Each developer has 2 Git repositories: a private local one and a public server-side one

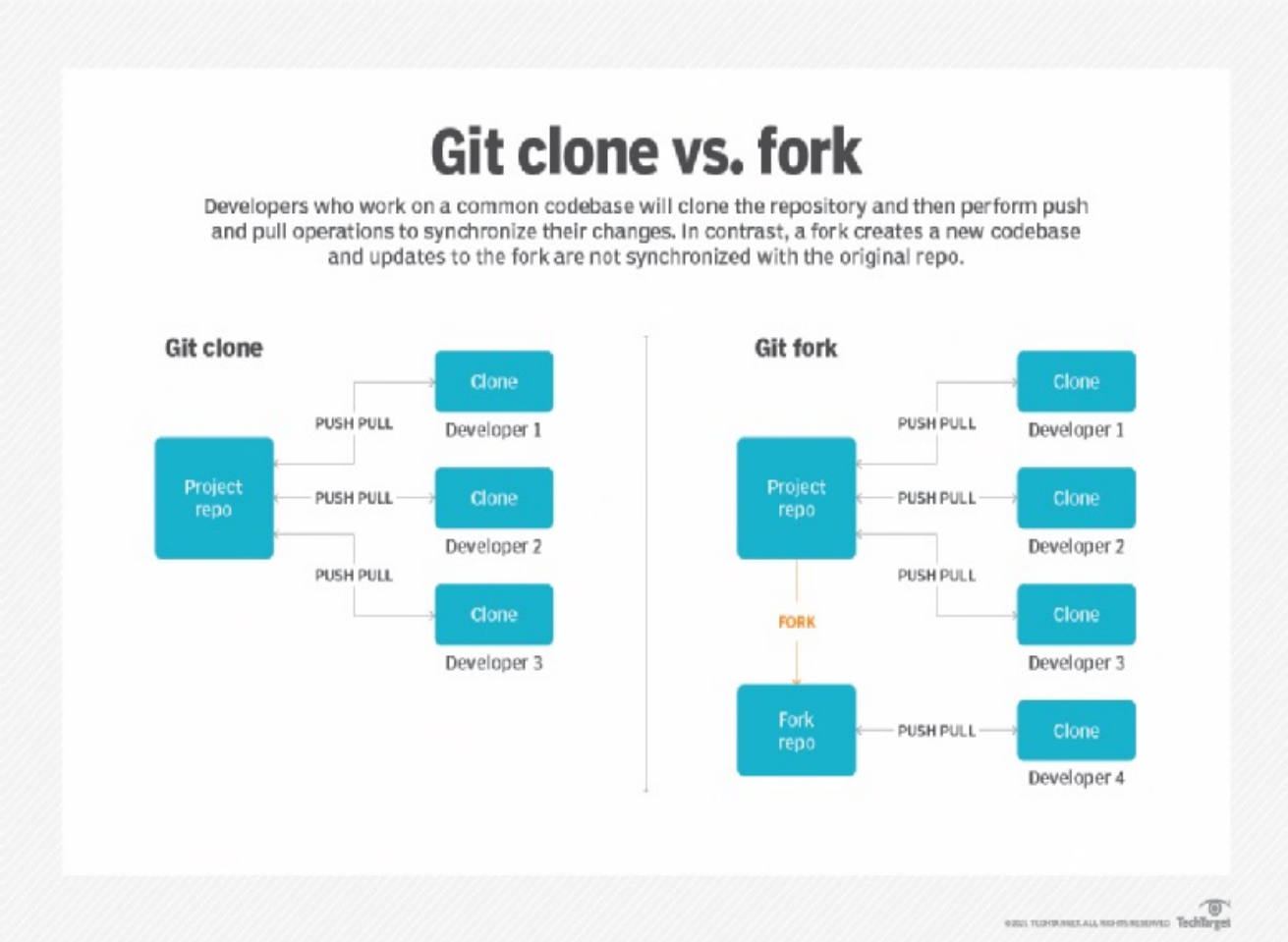
Advantage: contributions can be integrated without the need for everybody to push

https://github.blog/2022-05-02-friend-zone-strategies-friendly-fork-management/

https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/command-line-GitHub-fork-CLI-terminal-shell



https://docs.gitlab.com/ee/user/project/repository/forking_workflow.html



forking is done when you want to make your own version of a project and possibly contribute back to it

cloning is done when you simply want to have a local copy of the code to work on or run

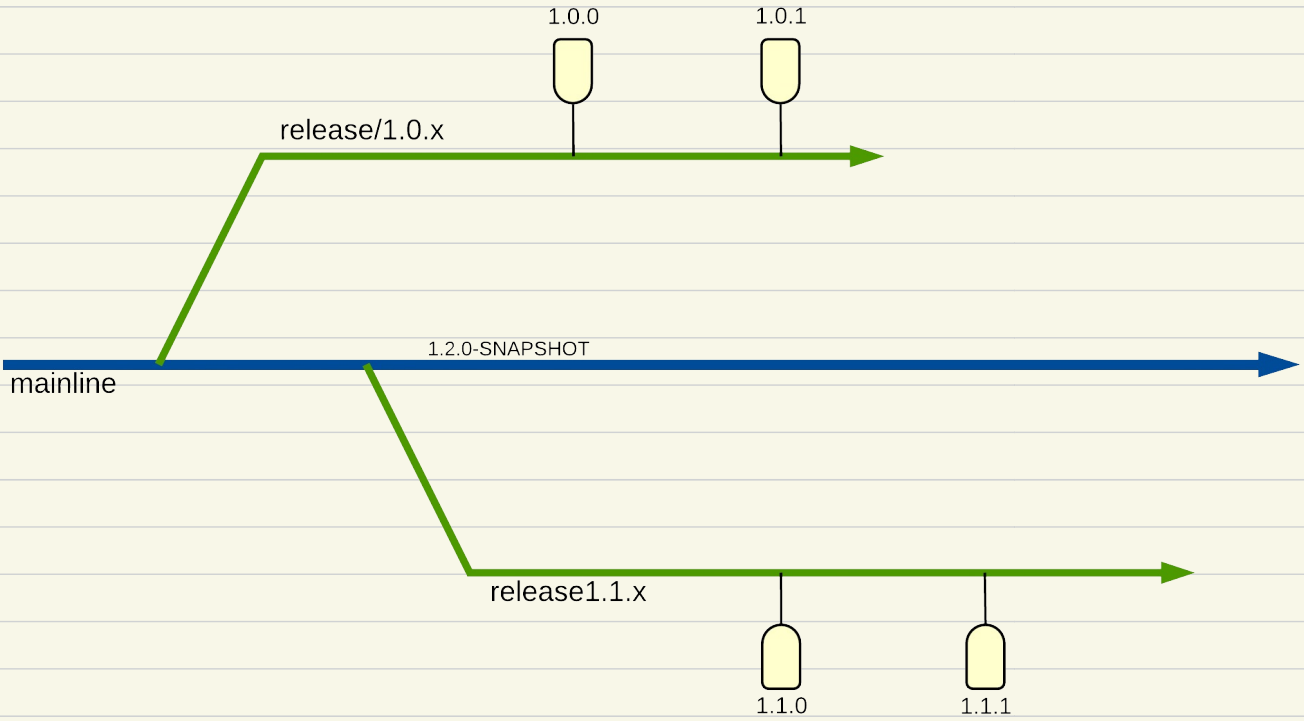
Release branching

Use to manage the release of new features and bug fixes

Creation of a separate branch from the main development branch to isolate the changes intended for a specific release

Release manager makes a branch from the main development branch

release branch → contains all the changes specific to that release such as bug fixes and configuration updates



http://releaseflow.org/#:~:text=Definition,-Making%20it%20clear&text=The%20%22Release%20Branching%20Strategy%22%20is,Continuous%20Integration%20book%20%5BCD%5D.

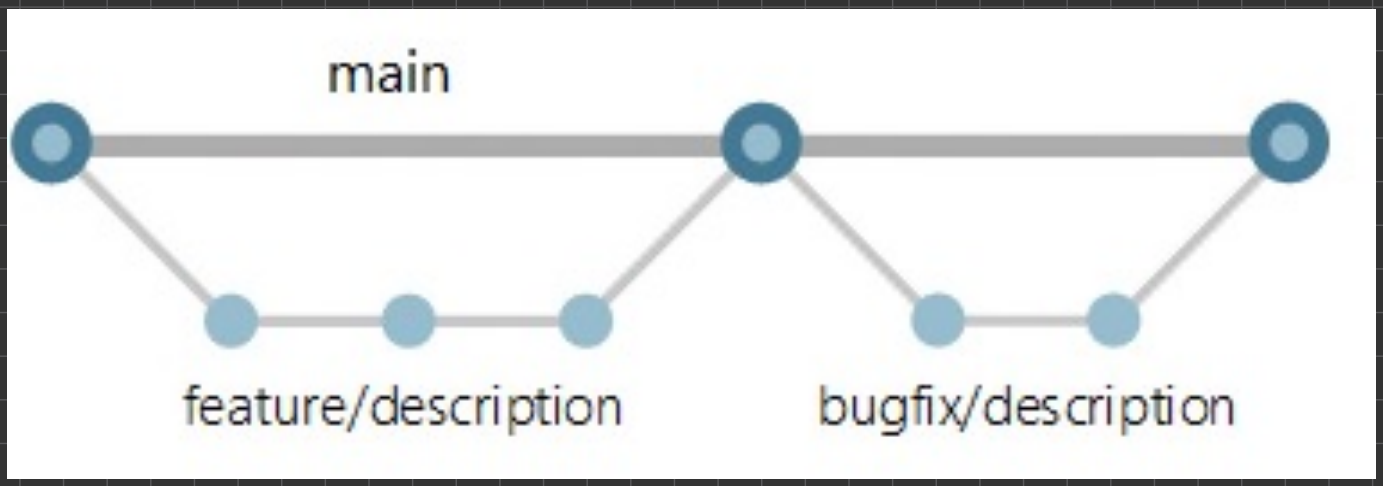
https://www.abtasty.com/blog/git-branching-strategies/

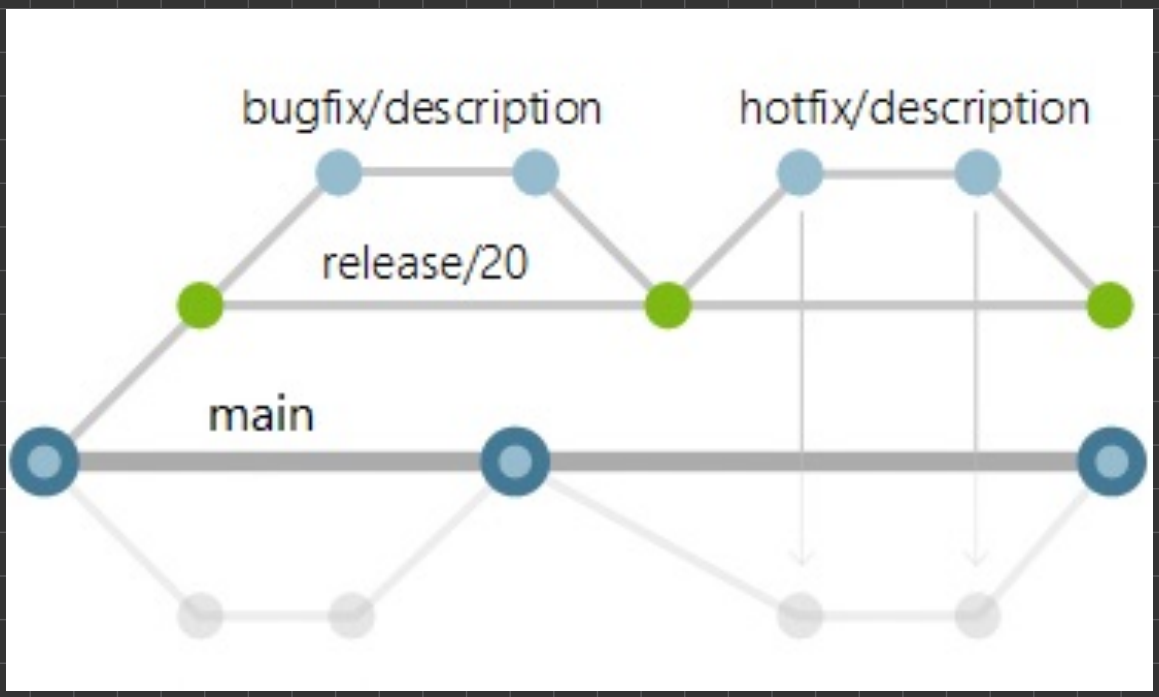
LEER !!

BUENA EXPLICACIÓN DE TODOS

https://learn.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops

good explanation of feature branches





THREE FLOW Branching

Uses 3 branches : master, candidate and release

- master
- candidate
- release

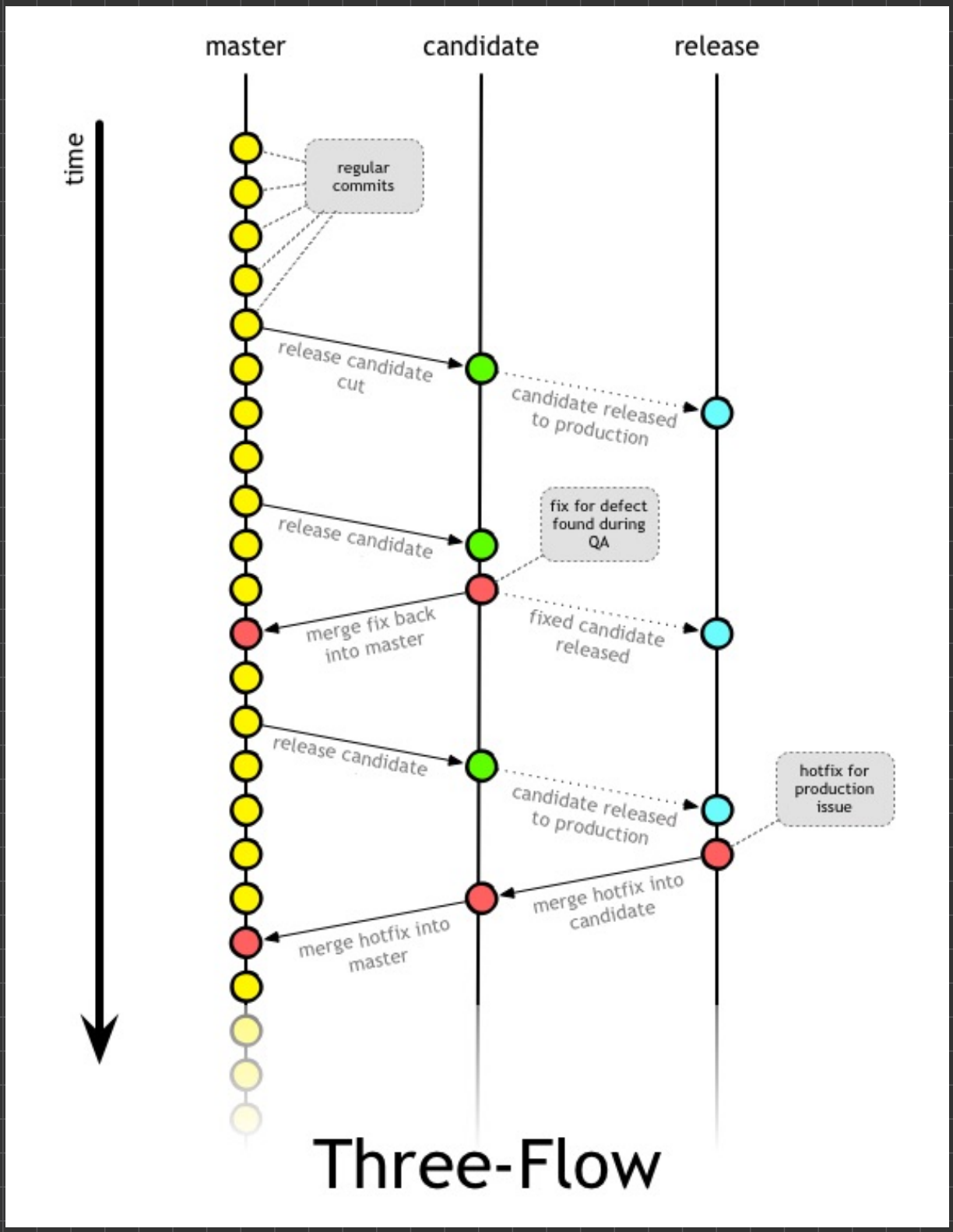
avoids feature branches

↓

more difficult to track changes

Reduced complexity

https://www.rodhilton.com/2017/04/09/a-different-branching-strategy/


master
candidate
release
time
regular commits
release candidate cut
candidate released to production
release candidate
fix for defect found during QA
merge fix back into master
fixed candidate released
release candidate
candidate released to production
hotfix for production issue
merge hotfix into candidate
merge hotfix into master
Three-Flow

GIT HUB FLOW

https://docs.github.com/en/get-started/quickstart/github-flow

How it works:

- create a new branch
- make changes and add commits
- open a pull request
- review
- deploy
- merge