mobiquity™
make mobile matter

GIT

Agenda

- GIT
  - Tools
  - Learning resources
  - GIT vs. SVN
  - GIT concepts (DAG, branch, tag, etc.)
  - GIT operations (commit, merge, etc.)
- Workflows

git

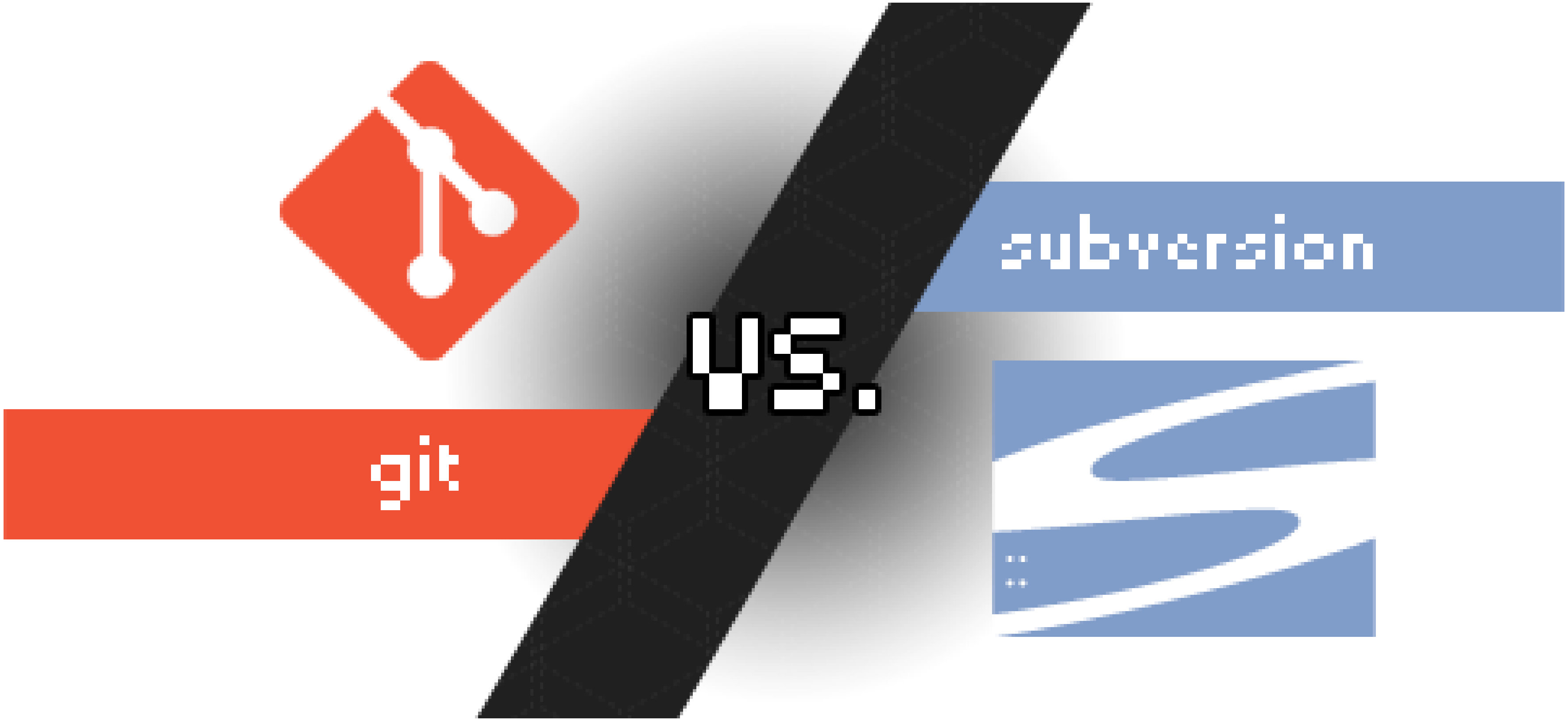
GIT / Tools

- command line – required by other tools (http://git-scm.com/downloads)
- Windows-only: TortoiseGit (https://code.google.com/p/tortoisegit/wiki/Download)
- SourceTree (http://www.sourcetreeapp.com/)
- IntelliJ (http://www.jetbrains.com/idea/download/)
- etc.

GIT / Learning resources

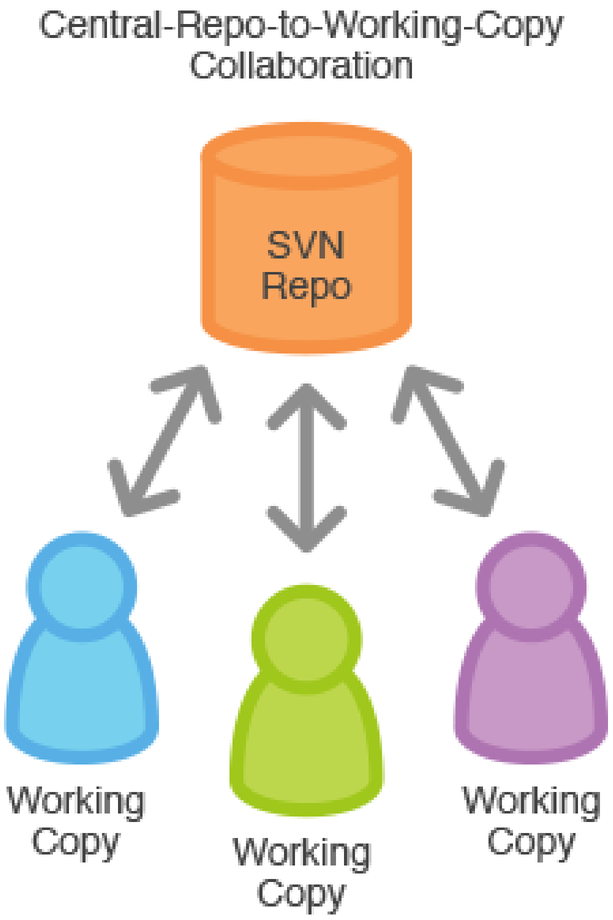
- GitHub training videos: http://youtu.be/FyfwLX4HAxM
- Youtube tutorials: https://www.youtube.com/results?search_query=git+tutorial
- git help [command]
- ProGit Book (free): http://git-scm.com/book
- A Visual Git Reference: http://marklodato.github.io/visual-git-guide/index-en.html
- git ready (tips): http://gitready.com/
- Git Internals (really useful to remove the "magic"): https://github.com/pluralsight/git-internals-pdf

GIT vs. SVN



GIT vs. SVN / Distribution / Repositories

- GIT is distributed, SVN is not → more flexibility



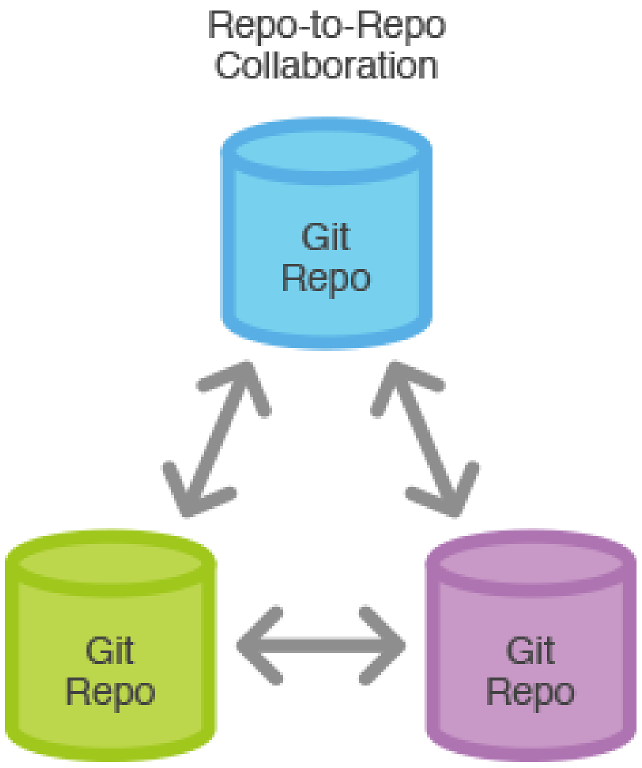
Central-Repo-to-Working-Copy Collaboration

SVN Repo

Working Copy

Working Copy

Working Copy



Repo-to-Repo Collaboration

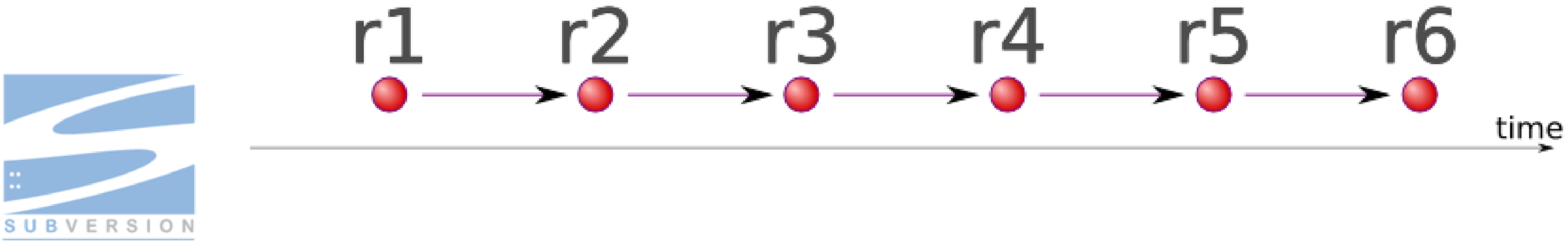
Git Repo

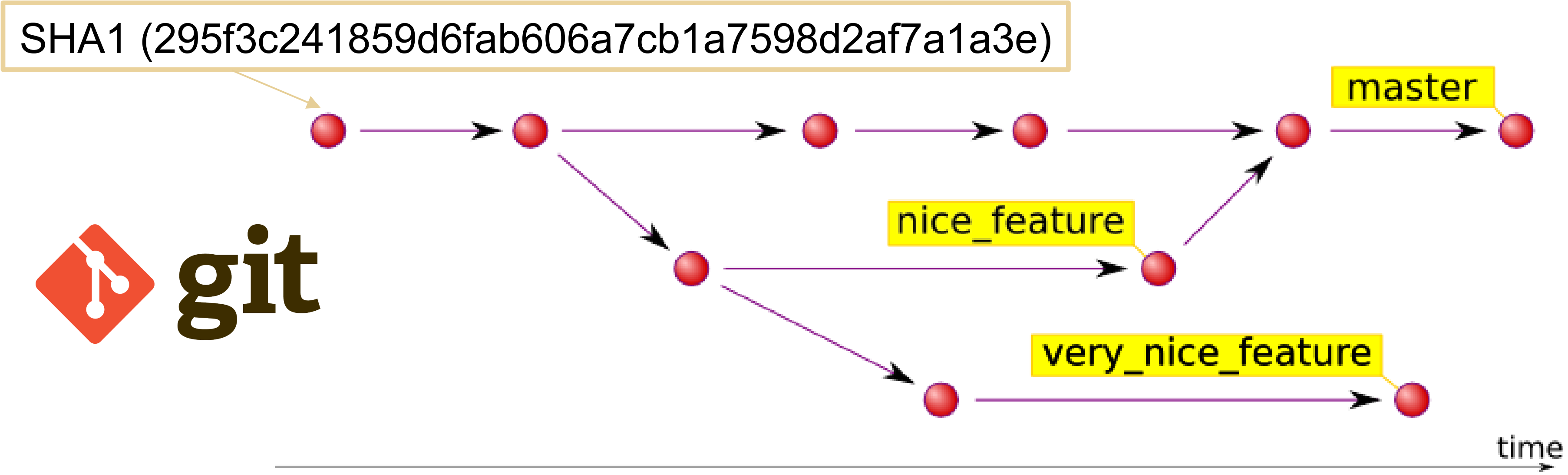
Git Repo

Git Repo

GIT vs. SVN / History

- GIT has real branches (DAG), SVN has linear history with simulated branches (just directories)

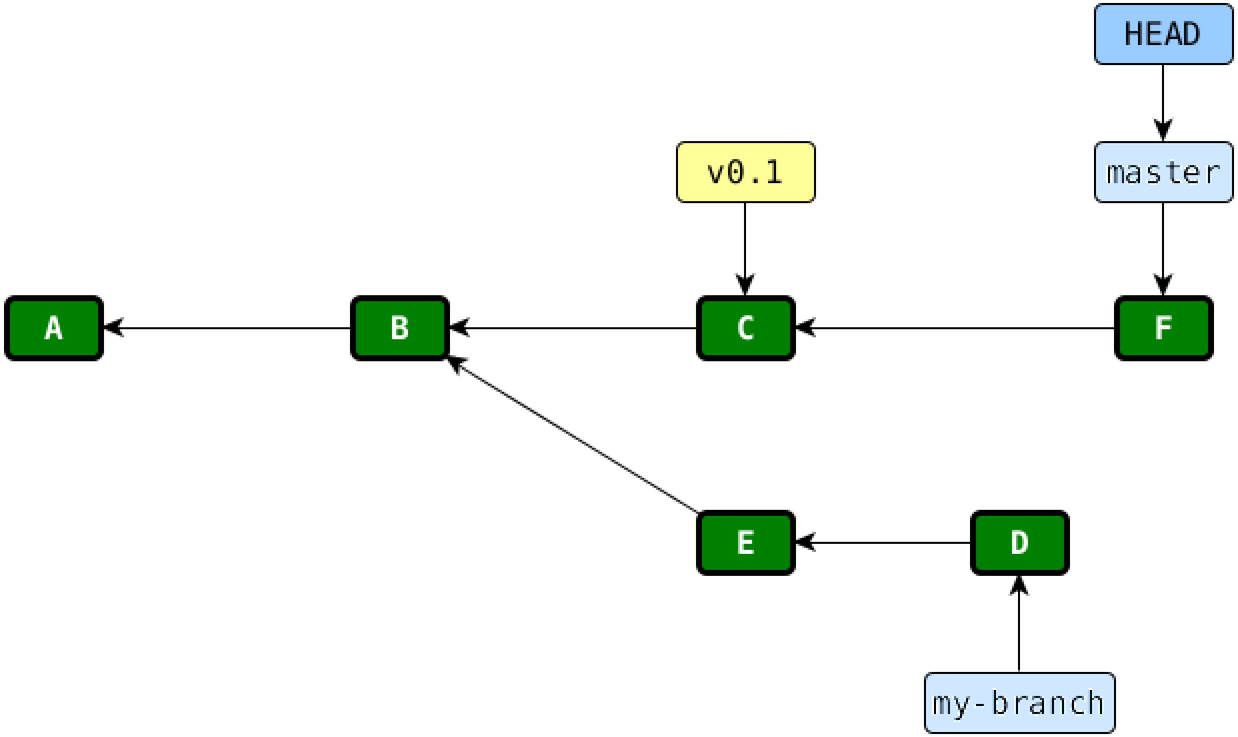




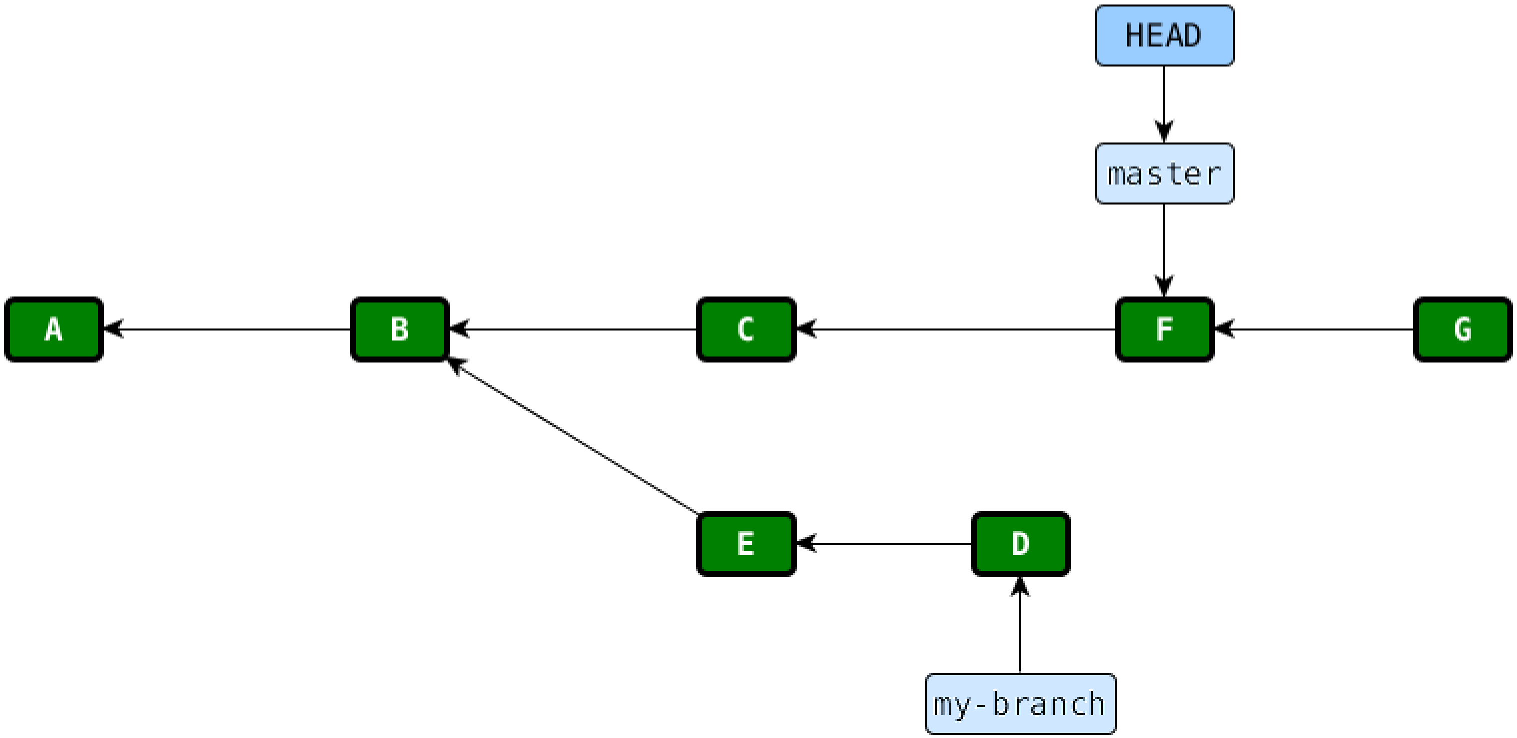
GIT vs. SVN / More on Branches

- in SVN you checkout different branches into different directories
- in GIT you switch between branches in the same directory

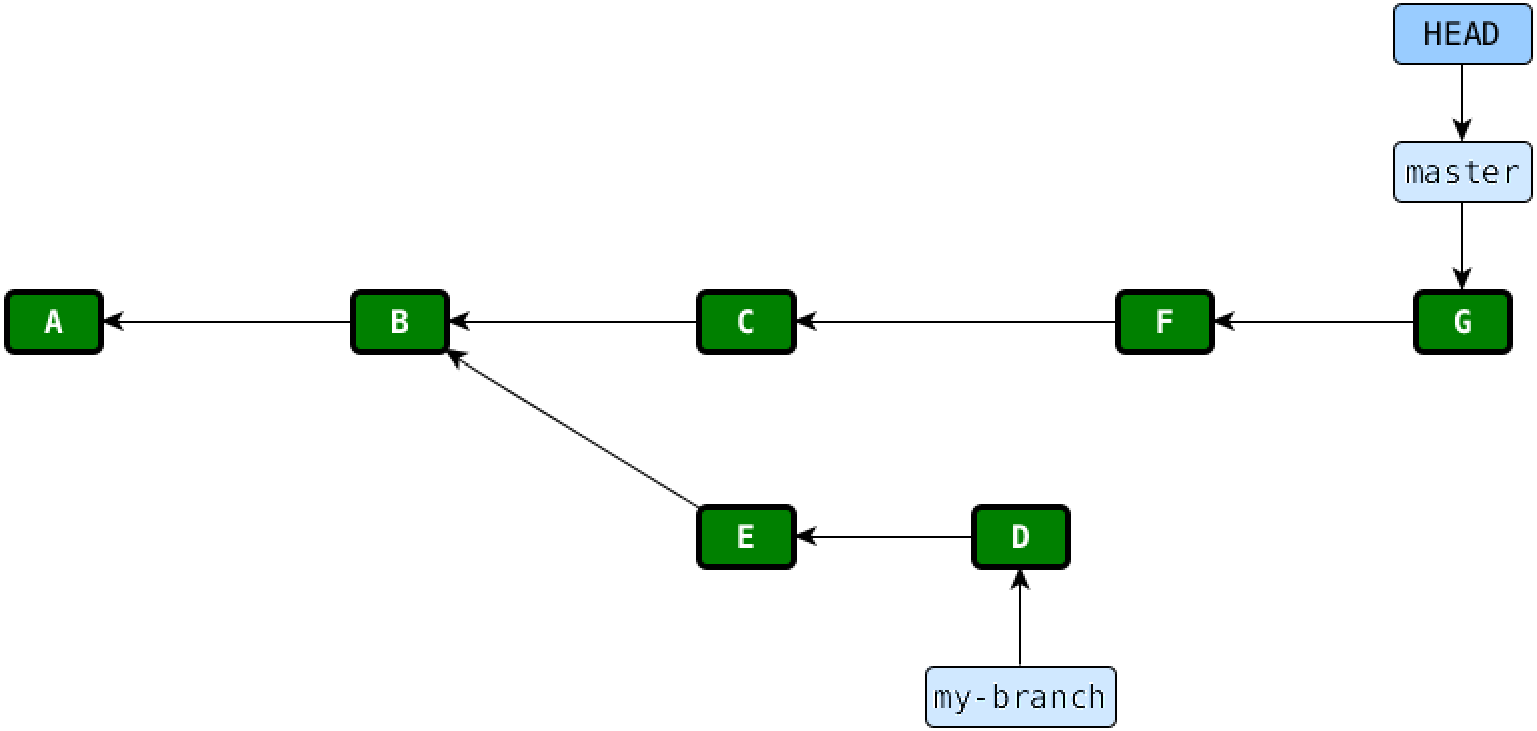
GIT concepts / DAG



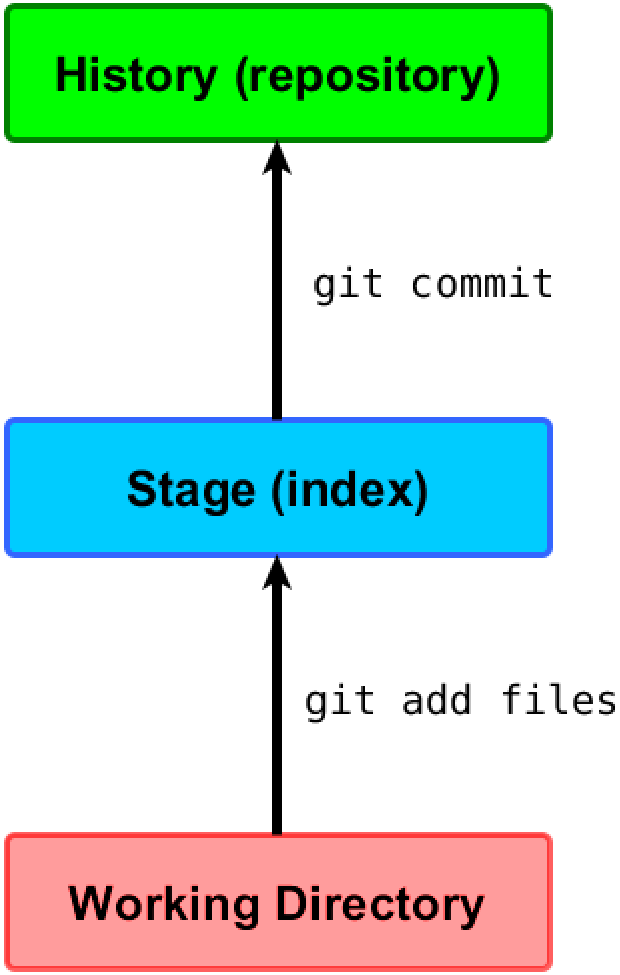
GIT commit / 1 of 2



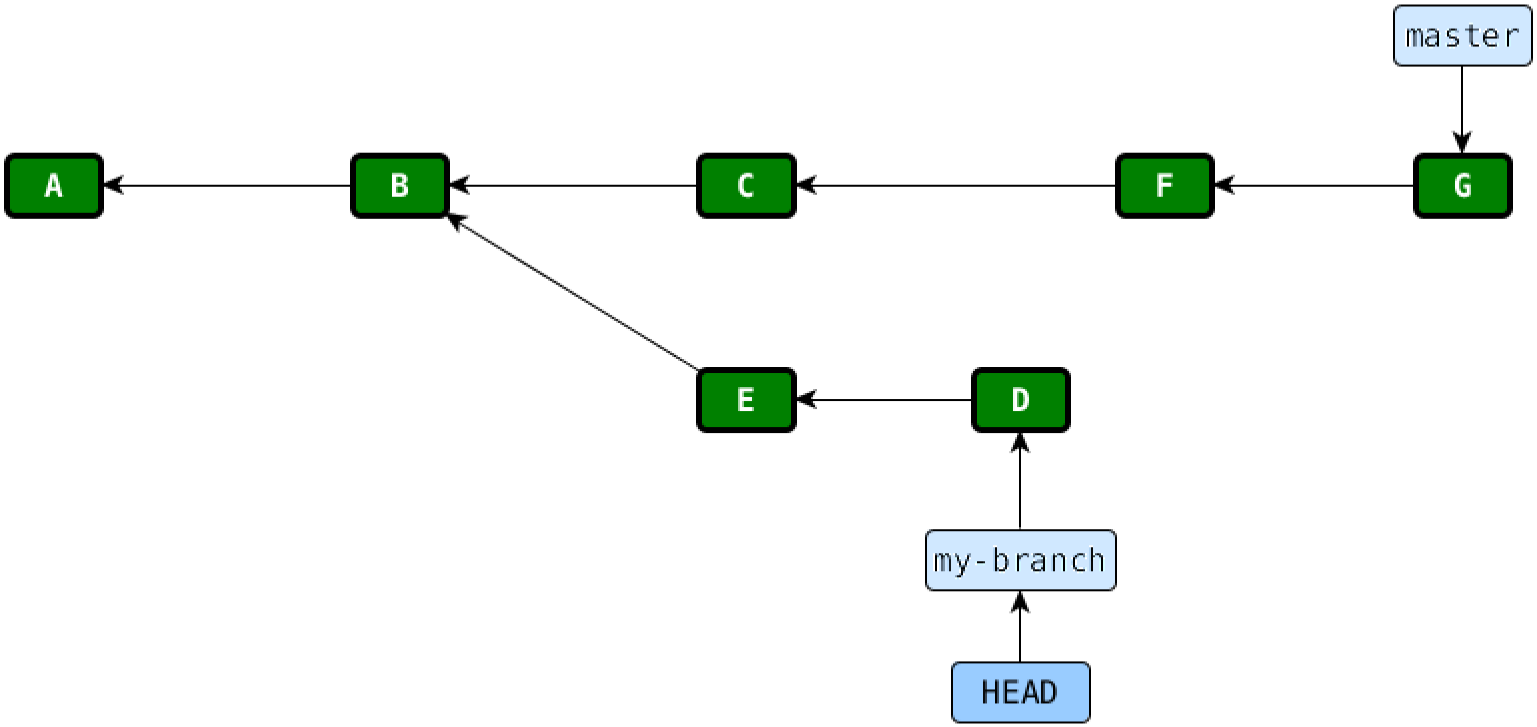
GIT commit / 2 of 2



GIT commit / staging area



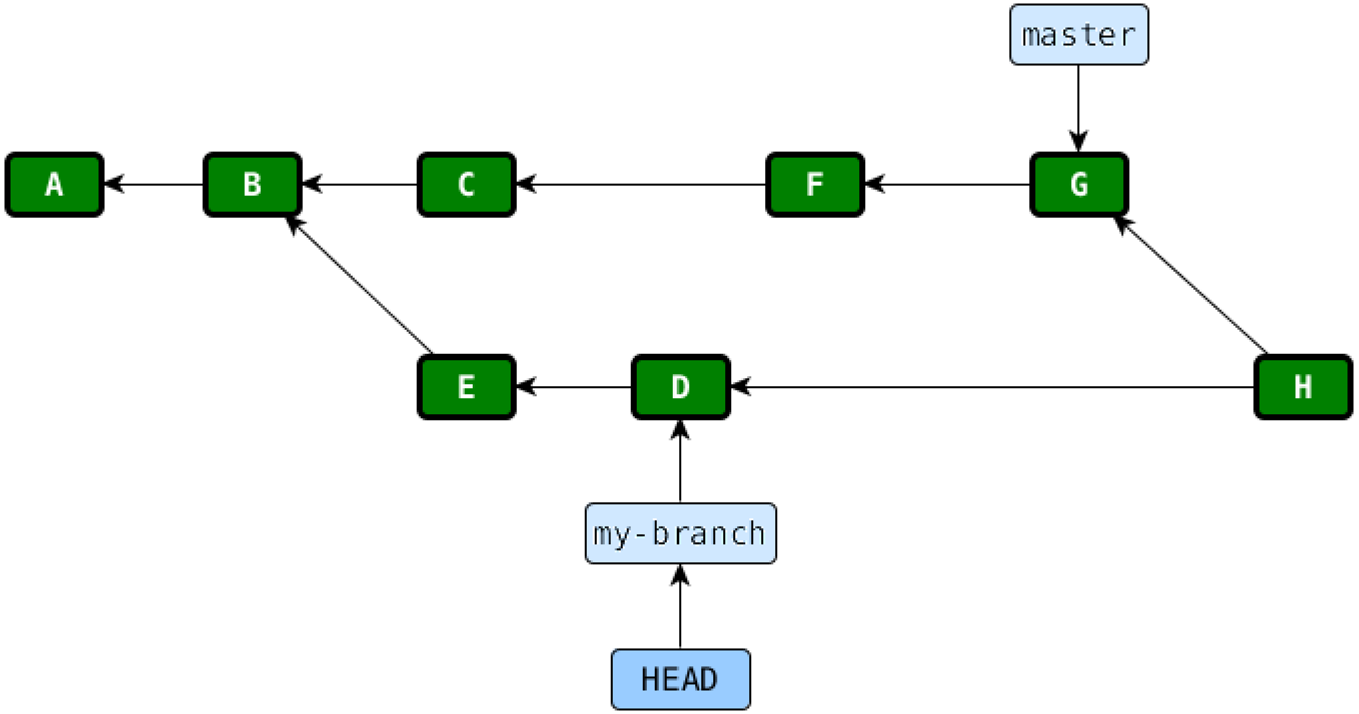
GIT checkout my-branch



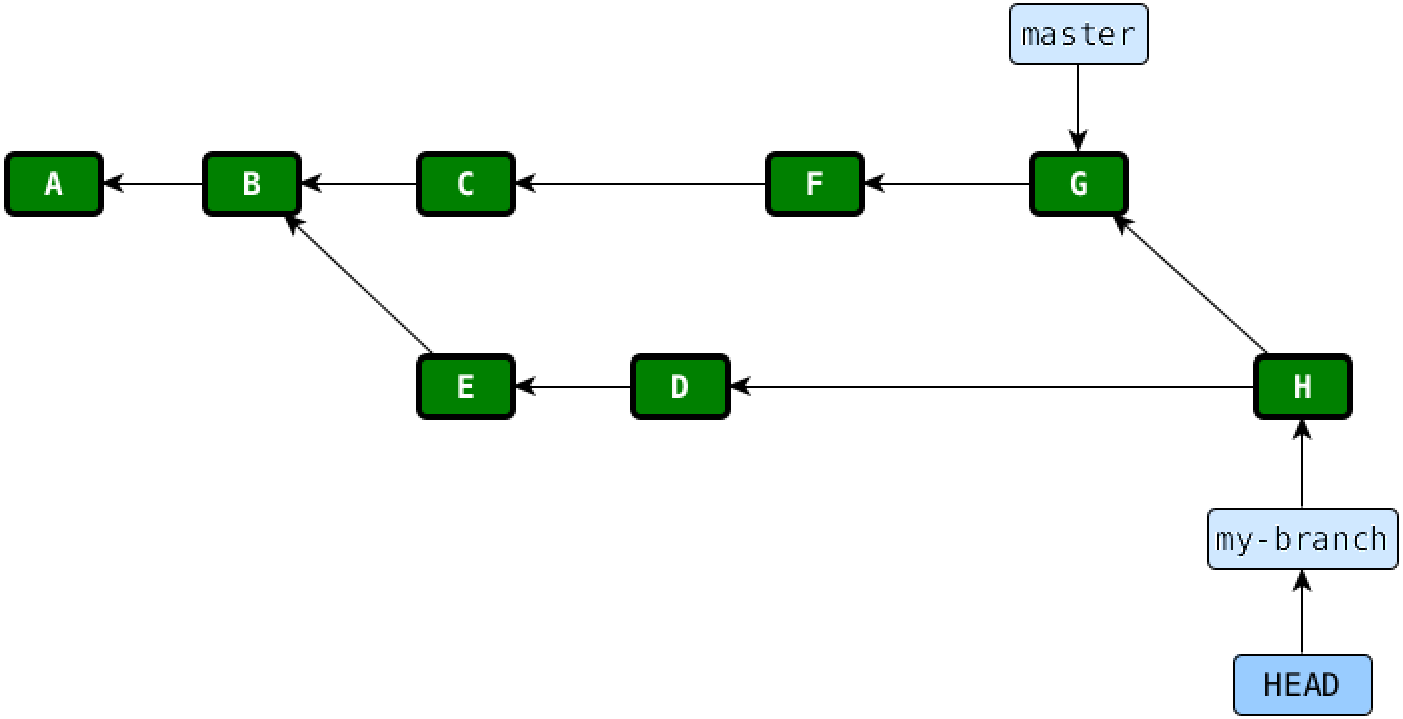
GIT unifying divergent histories

- two ways:
  - merge
  - rebase

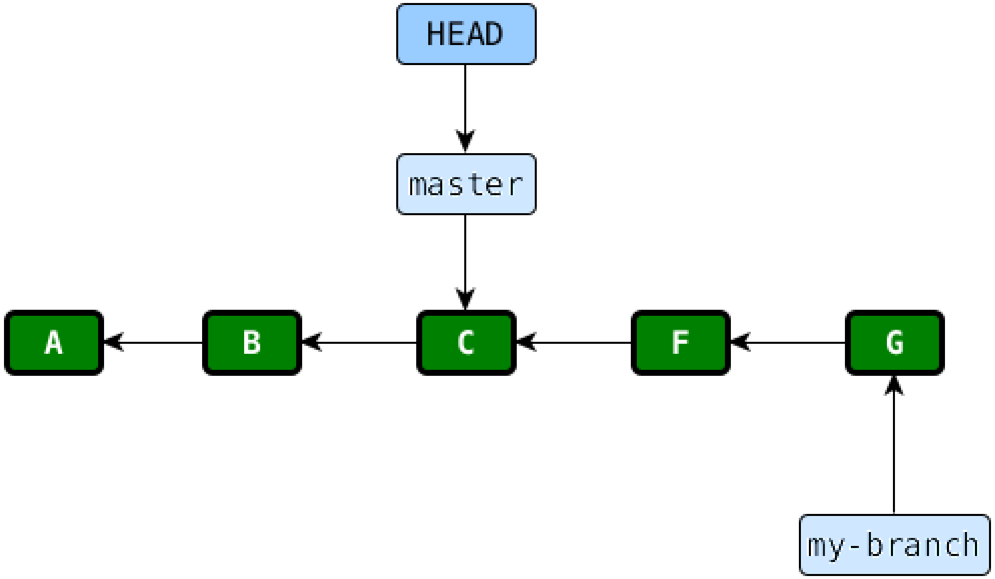
GIT merge master into my-branch /1 of 2



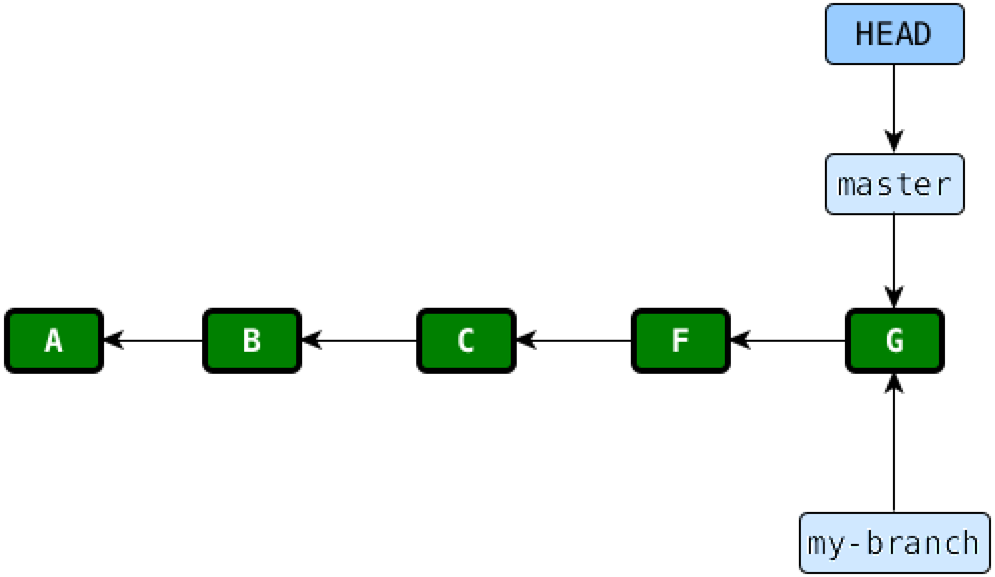
GIT merge master into my-branch /2 of 2



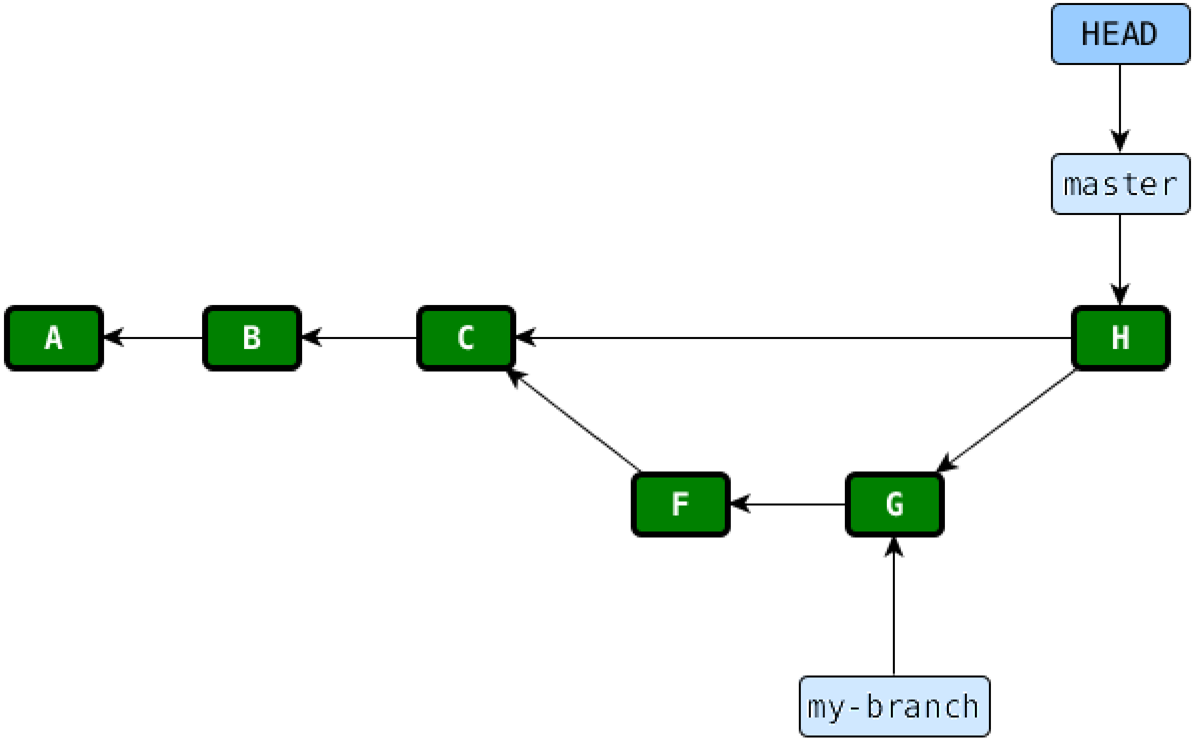
GIT fast-forward merges



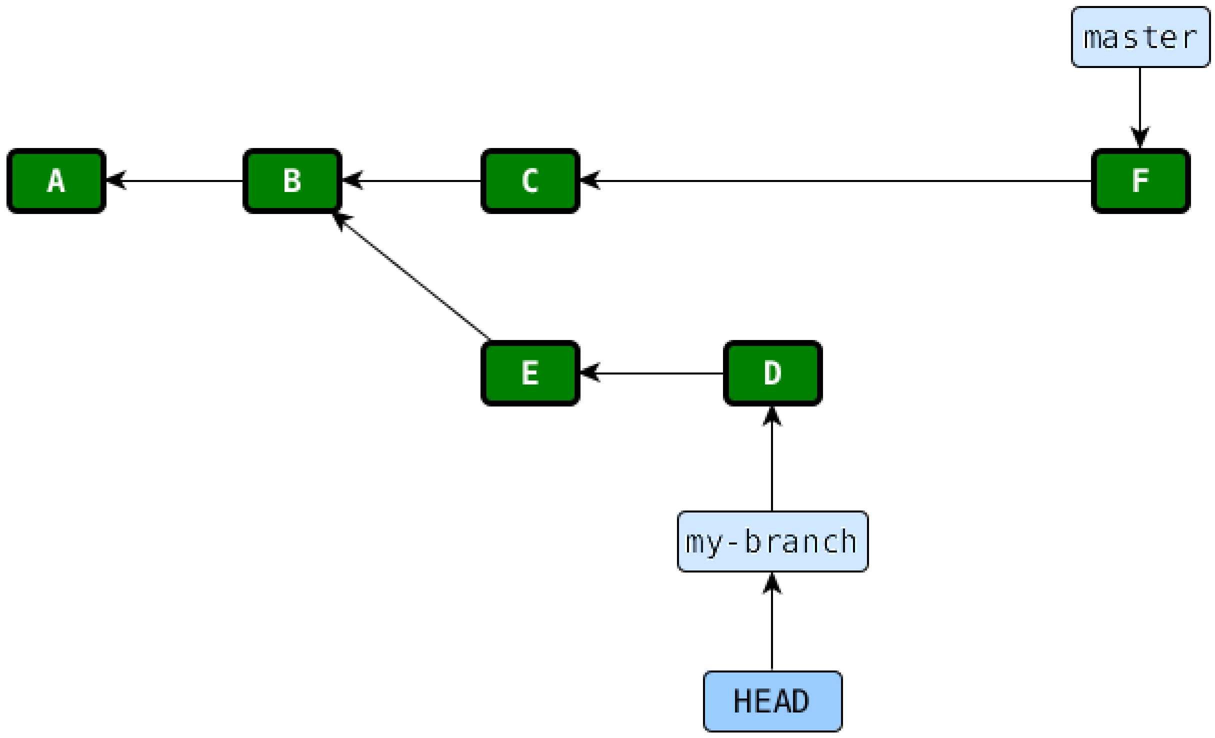
GIT fast-forward merge my-branch into master



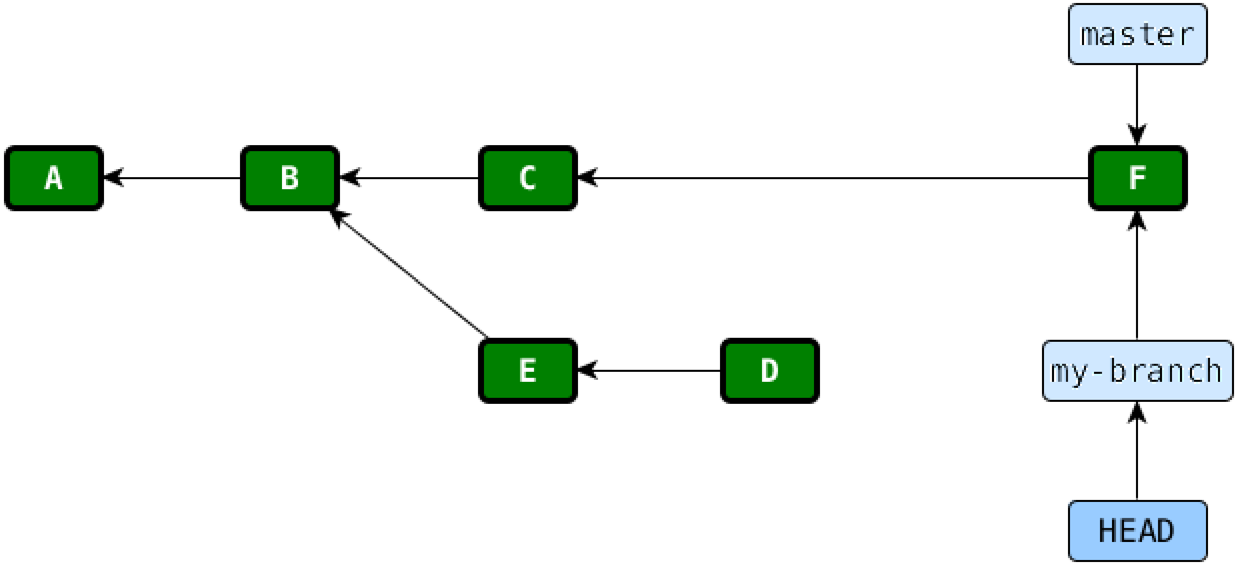
GIT merge my-branch into master forcing non-fast-forward (--no-ff)



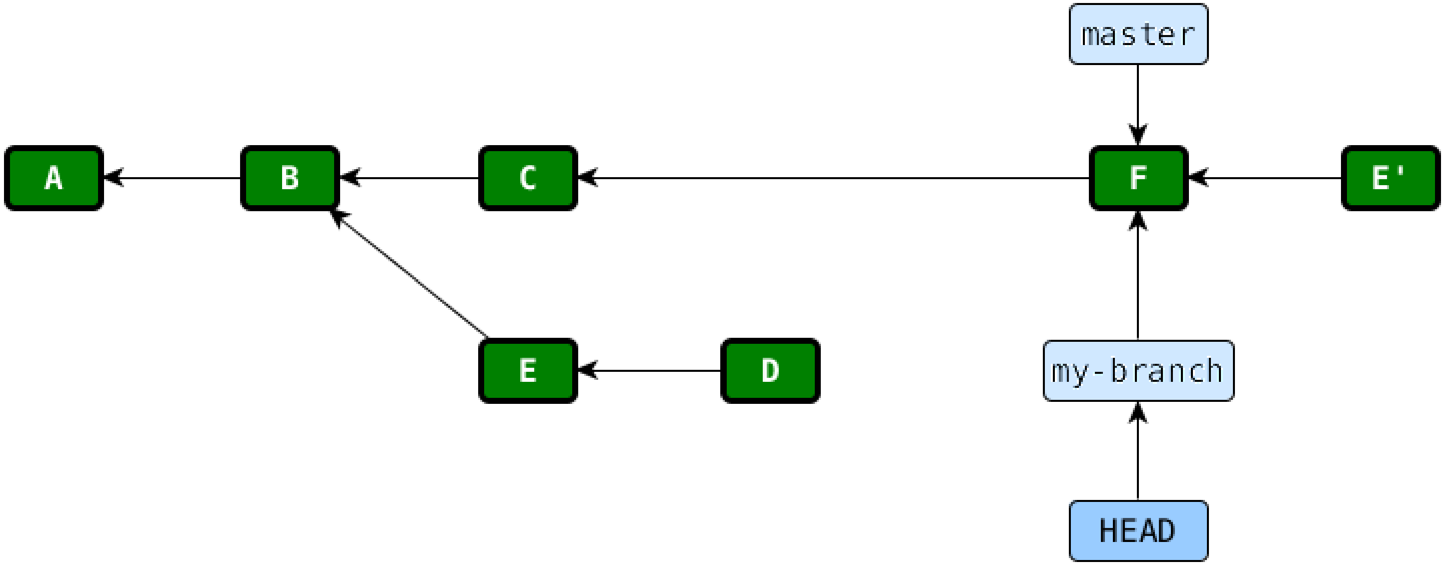
GIT rebase my-branch on top of master / 1 of 6



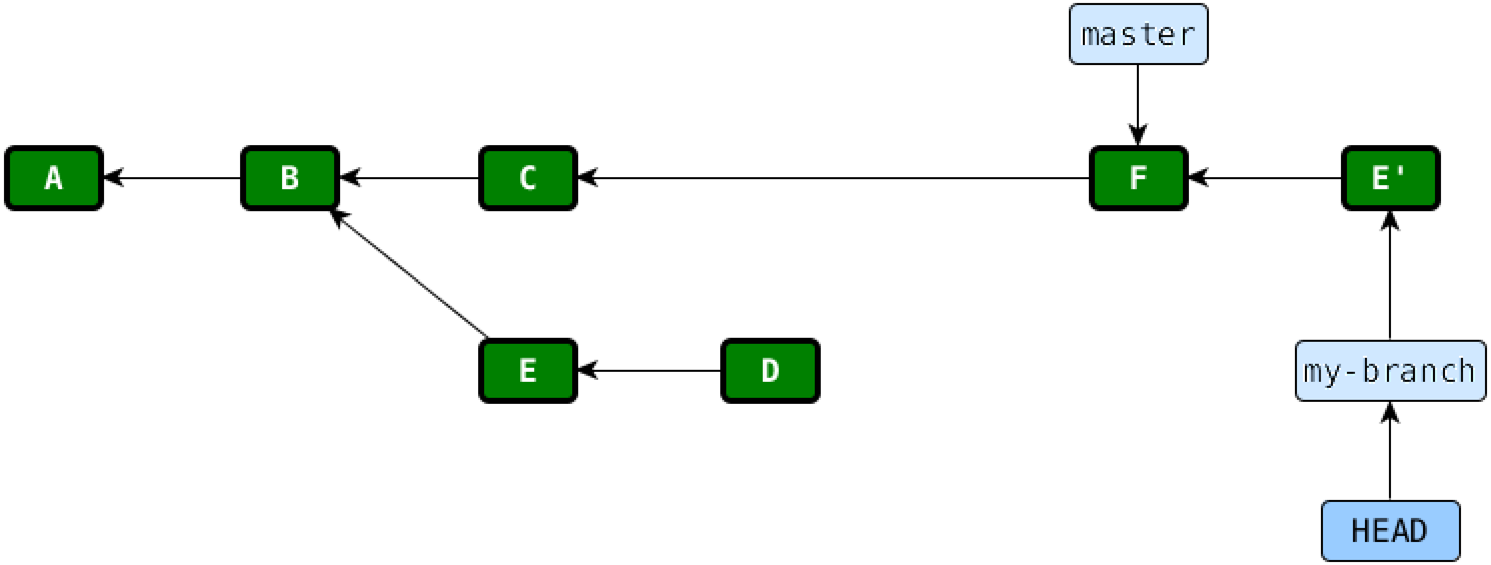
GIT rebase my-branch on top of master / 2 of 6



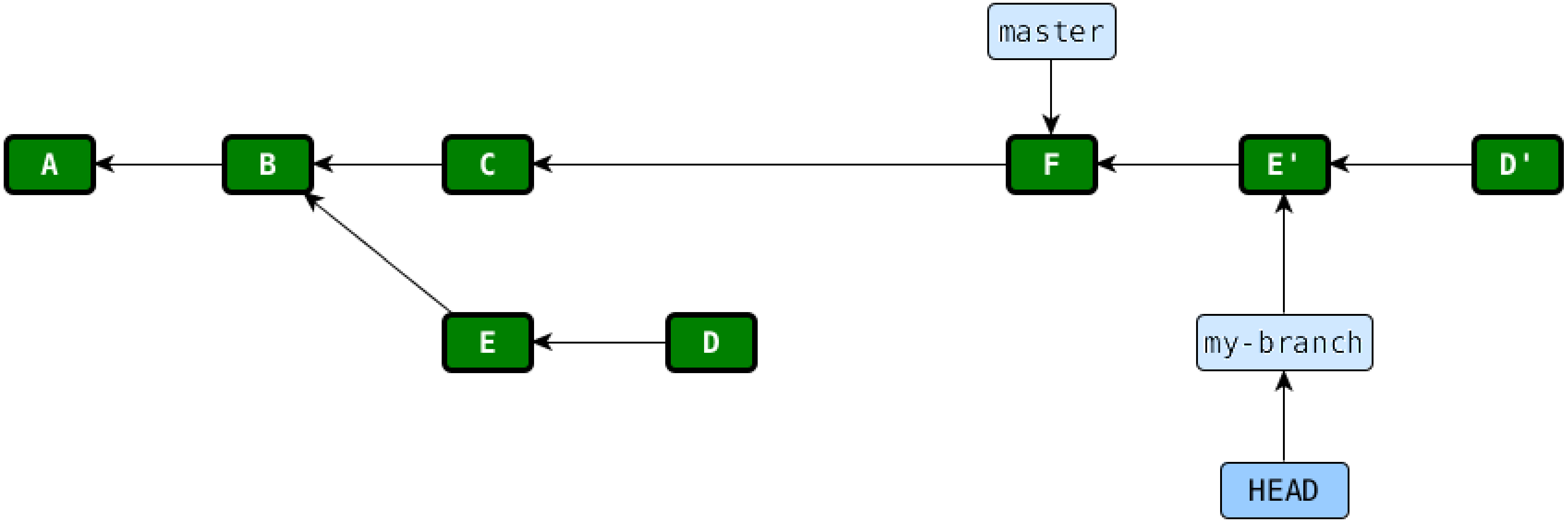
GIT rebase my-branch on top of master / 3 of 6



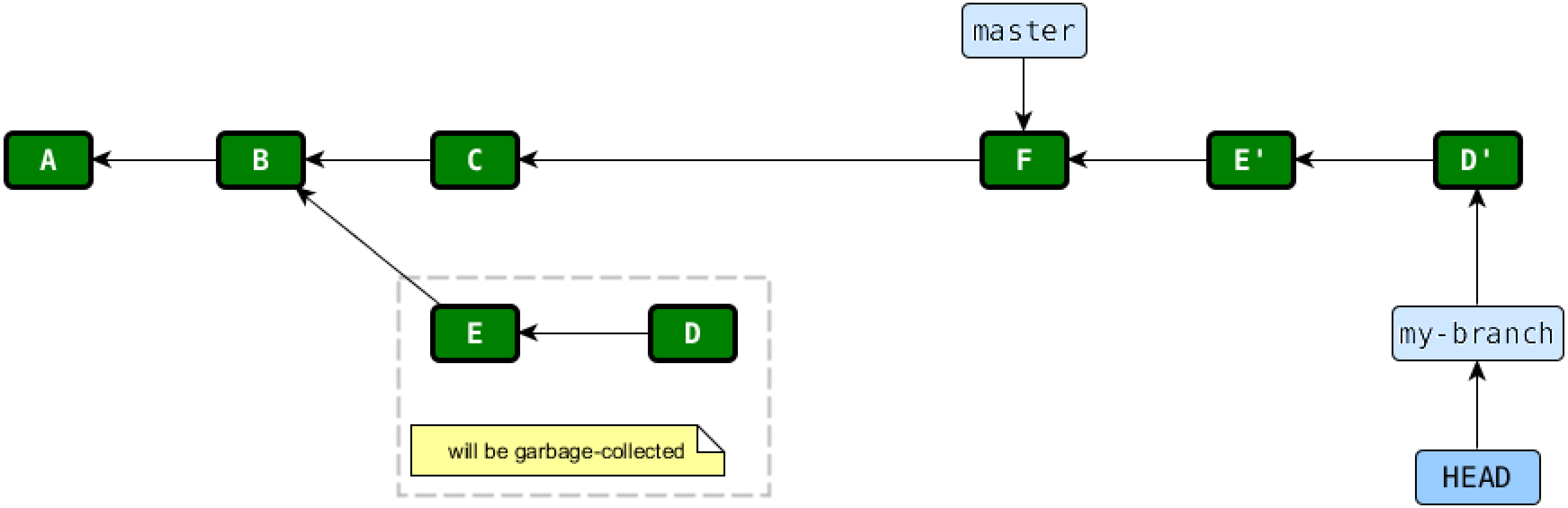
GIT rebase my-branch on top of master / 4 of 6



GIT rebase my-branch on top of master / 5 of 6



GIT rebase my-branch on top of master / 6 of 6



GIT commit --amend

- "`git commit --amend`" creates a new commit that contains changes from both the **last commit** and changes in the **working directory**

GIT rebase --interactive

- "`git rebase --interactive`" creates an alternate history and allows you to:
  - reorder commits
  - drop some commits (like they were never done)
  - insert commits in between other commits
  - squash multiple commits into one
  - split one commit into multiple commits
- some of these operations (like reordering) may result in conflicts that need to be manually resolved

GIT filter-branch

- "`git filter-branch`" allows you to create an alternate history using scripts (e.g. run "`sed`" for all commit messages)

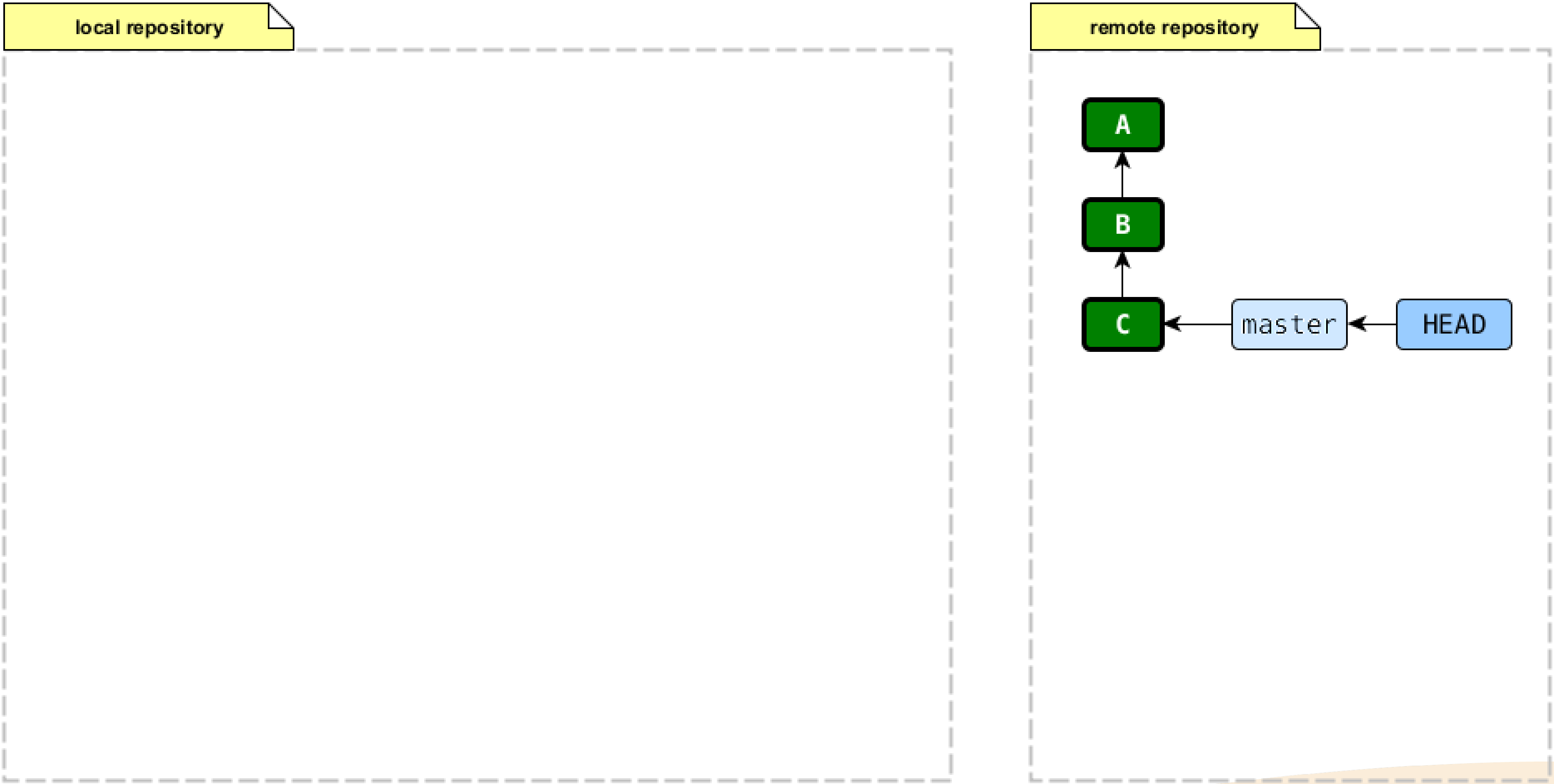
GIT / re-writing history

- these commands allows to re-write history
  - `rebase`
  - `rebase --interactive`
  - `commit --amend`
  - `filter-branch`

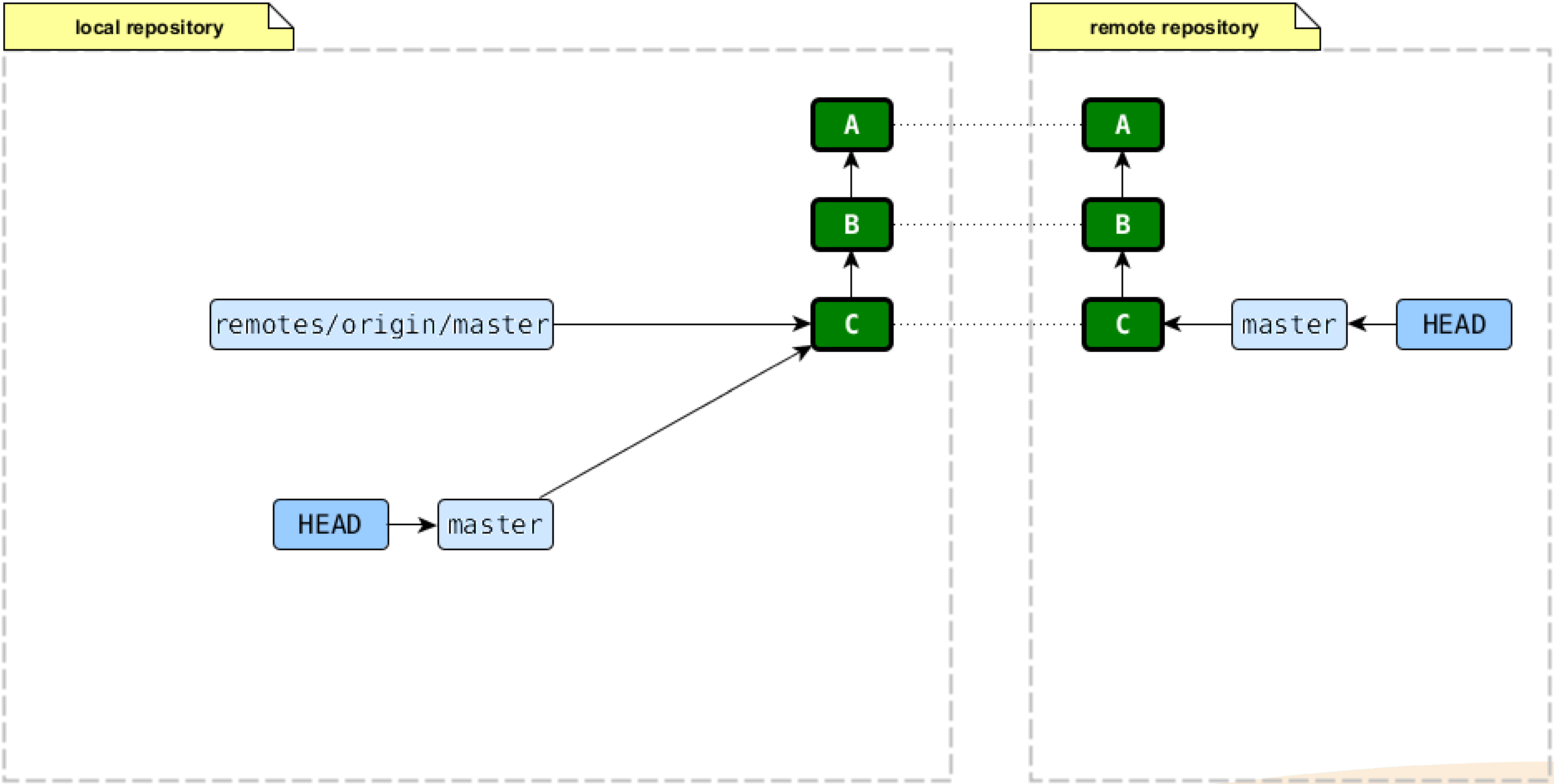
- **never!** (ever, ever, ok sometimes , ⏪) re-write public history (commits that have been pushed to a public repository

GIT clone / 1 of 4

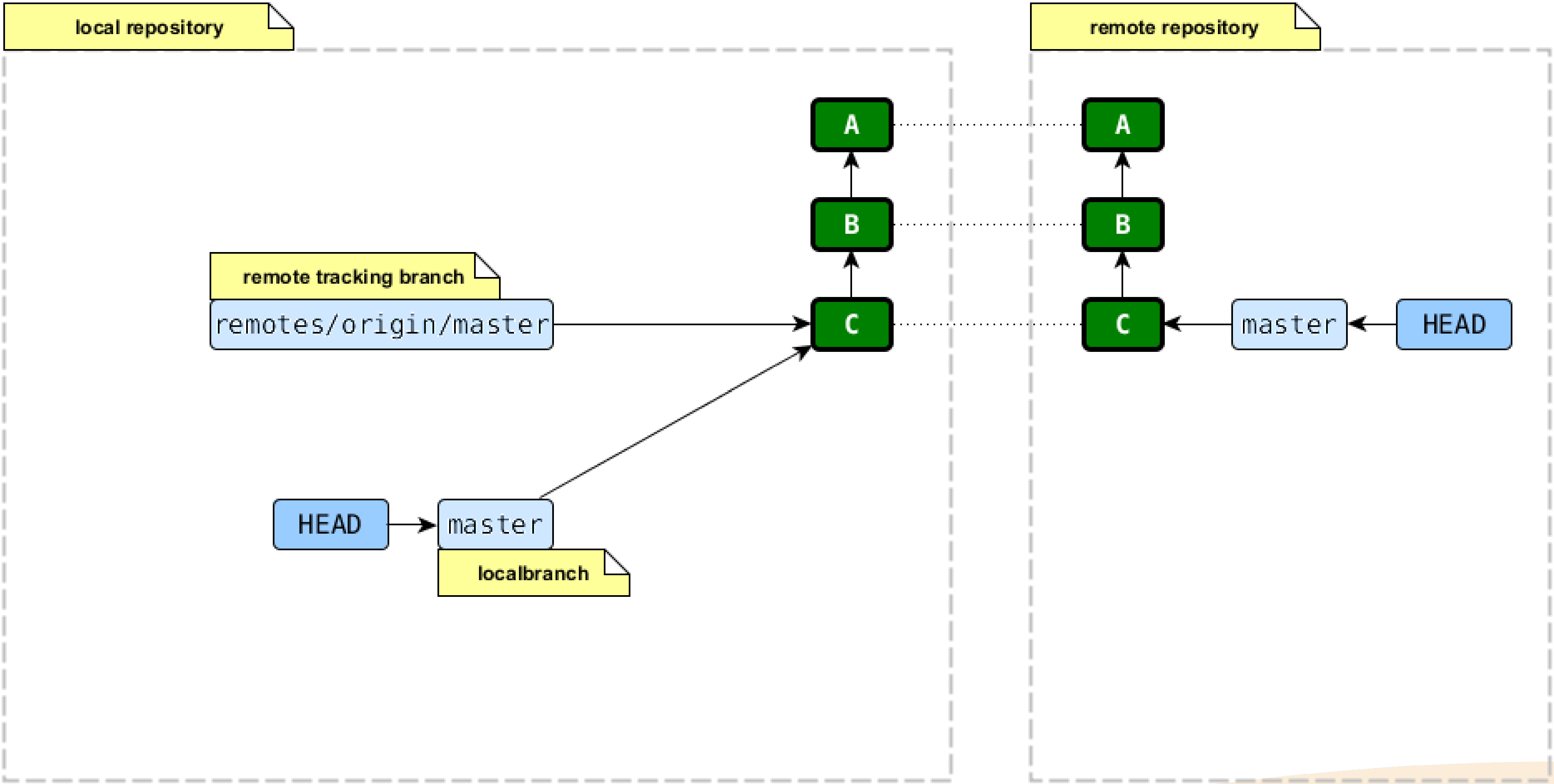
- use "git clone" to "copy" a remote repository, to start work



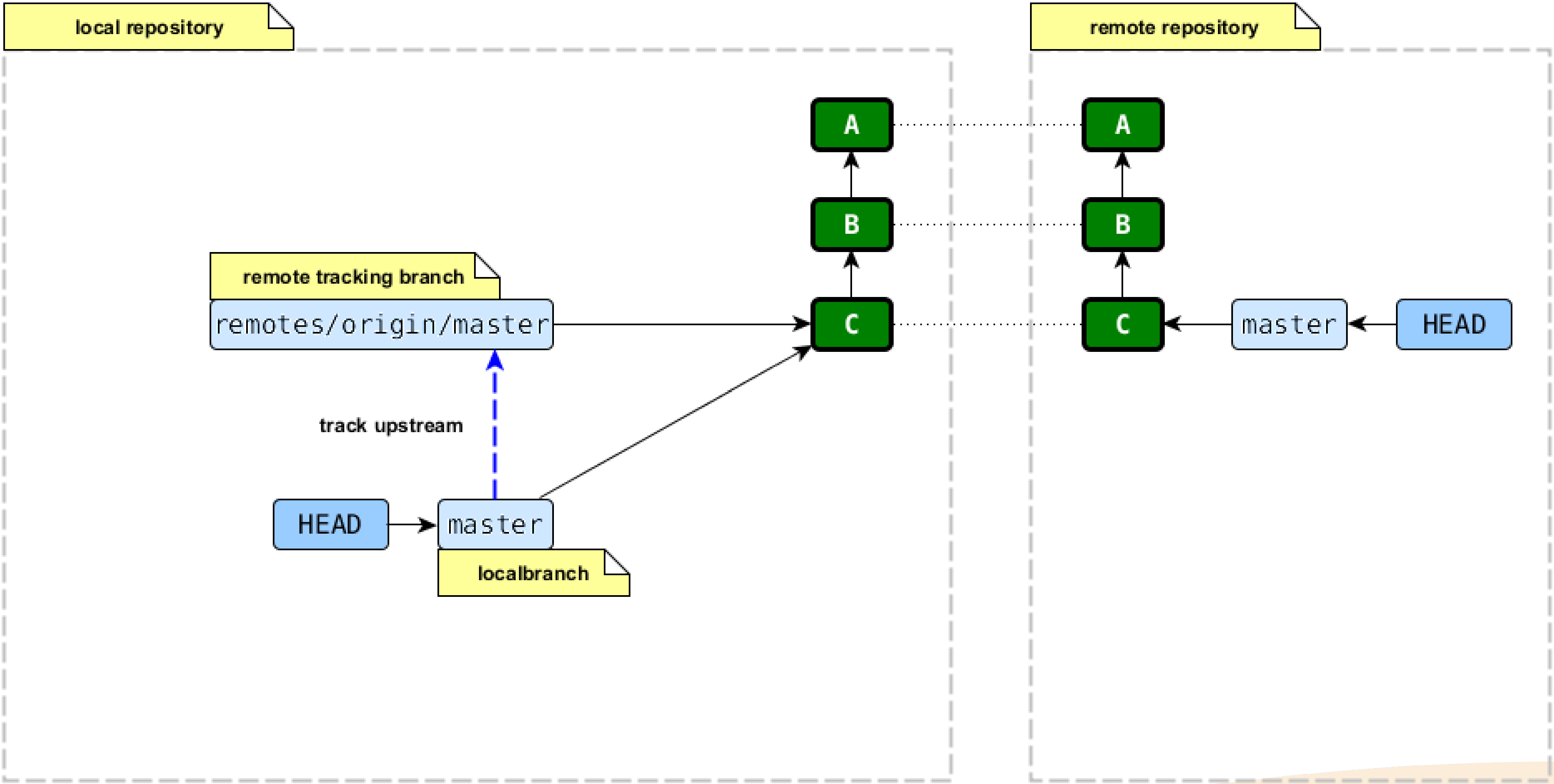
GIT clone / 2 of 4



GIT clone / 3 of 4

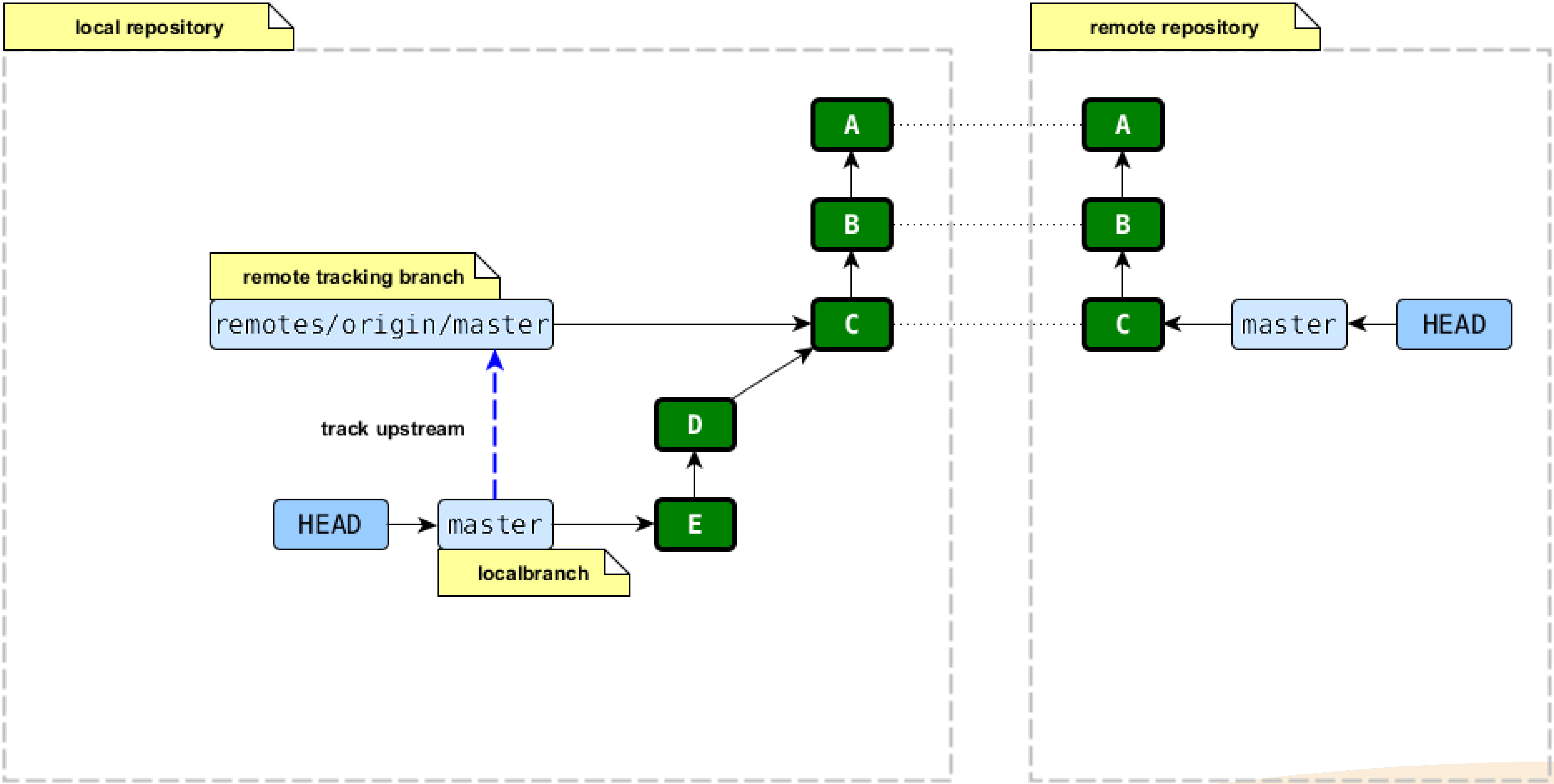


GIT clone / 4 of 4

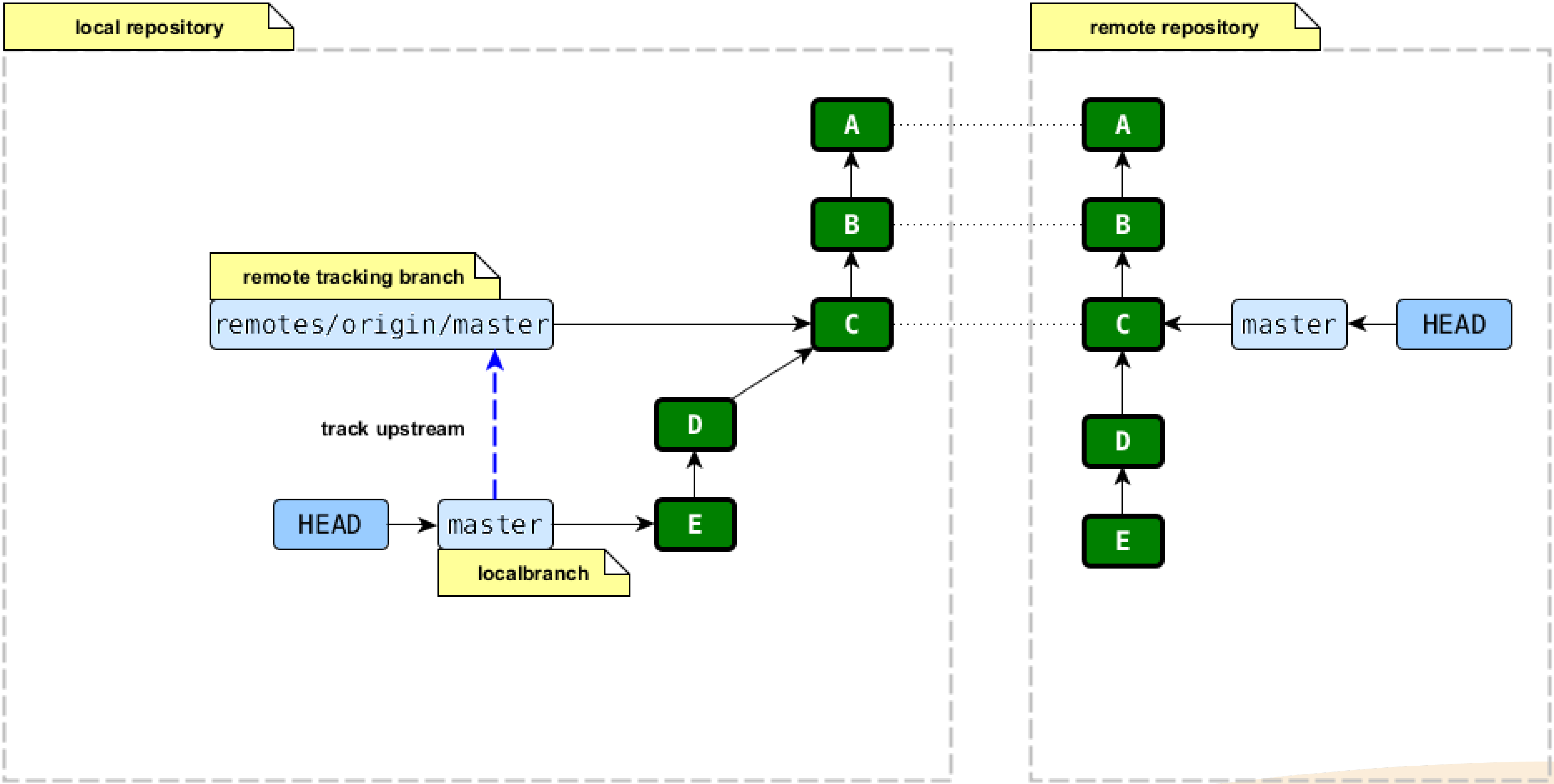


GIT push / success case / 1 of 5

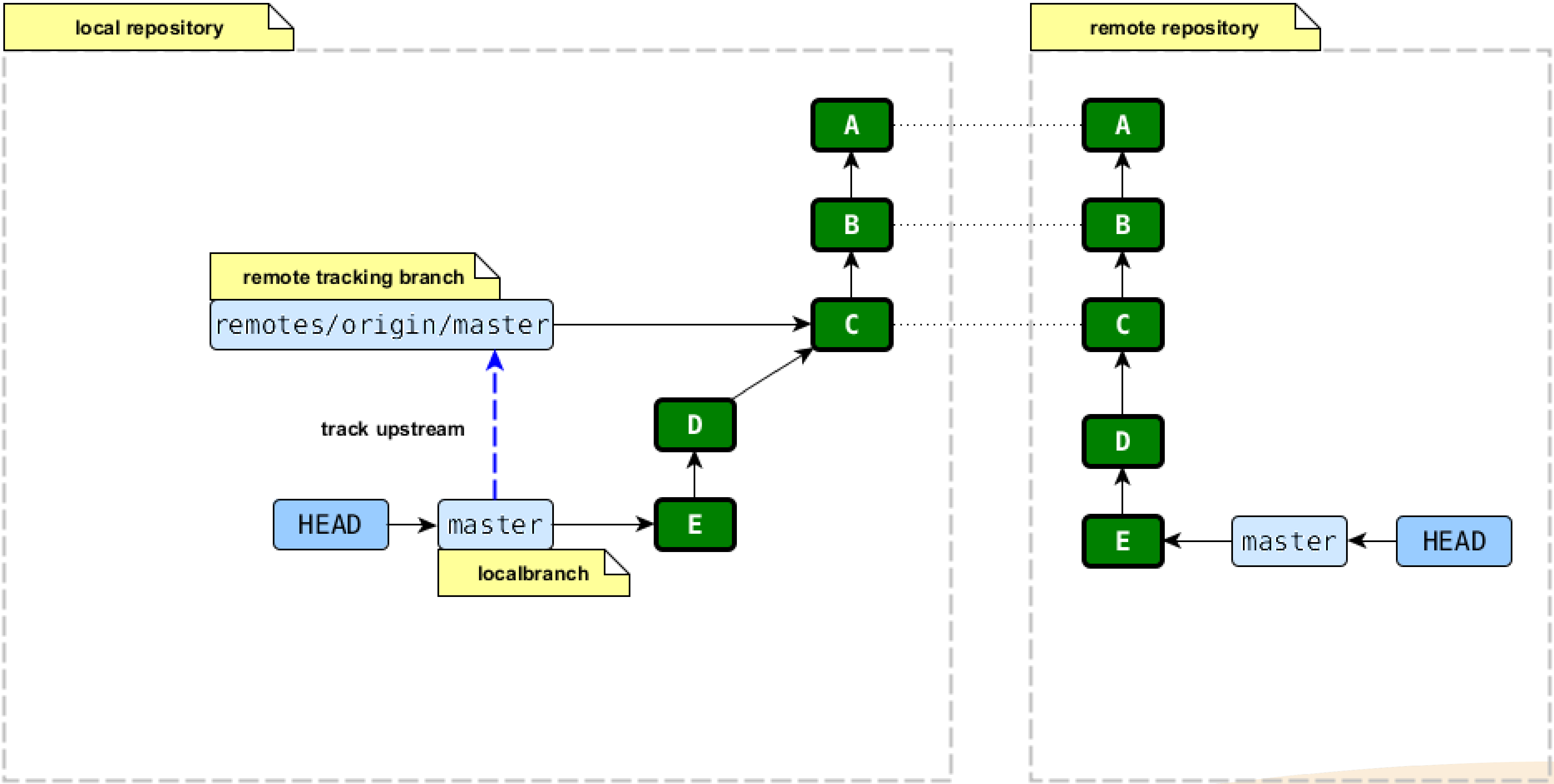
- all commit/merge/rebase operations are local
- "git push" sends the commits to another GIT repo



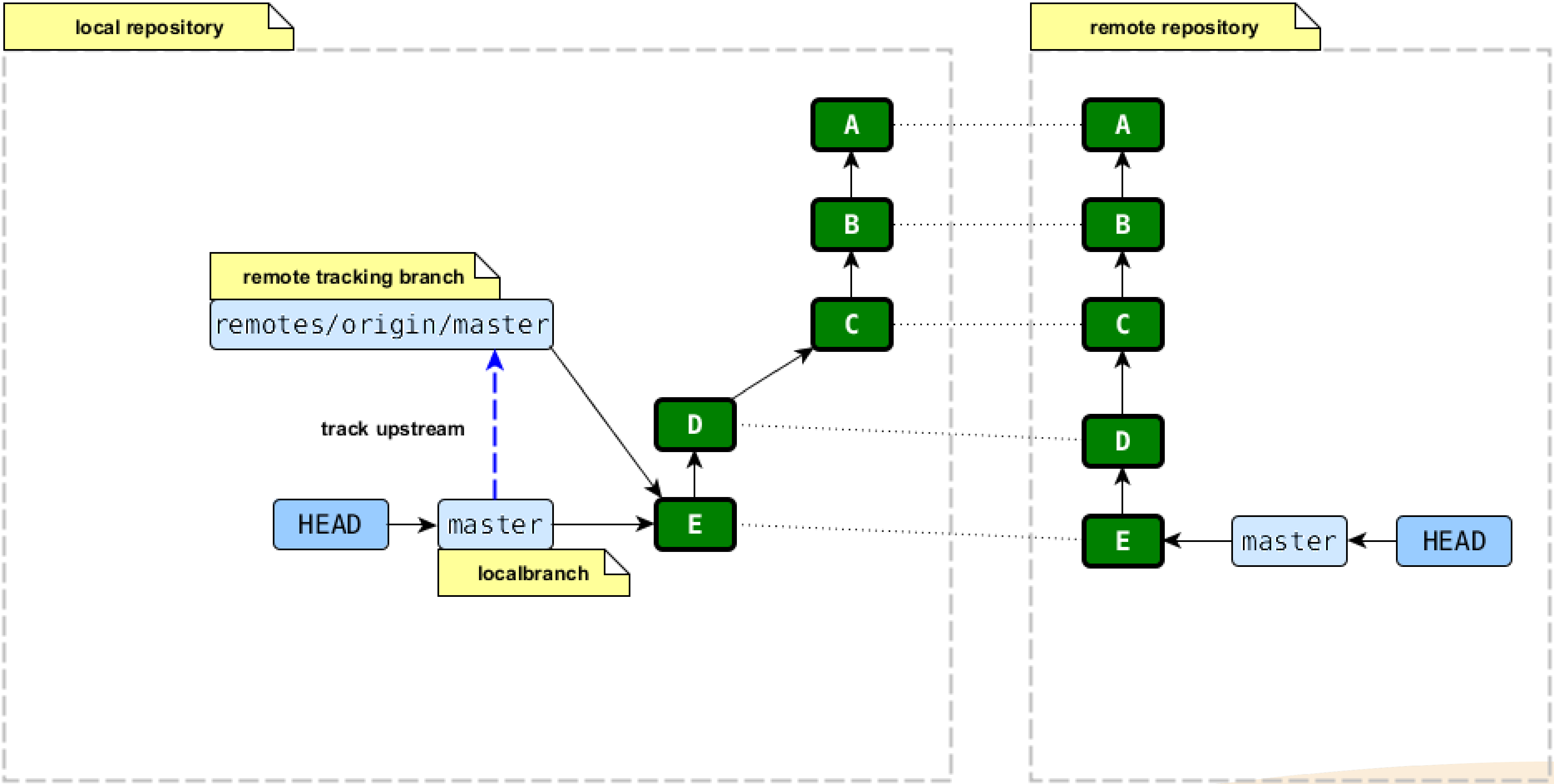
GIT push / success case / 2 of 5



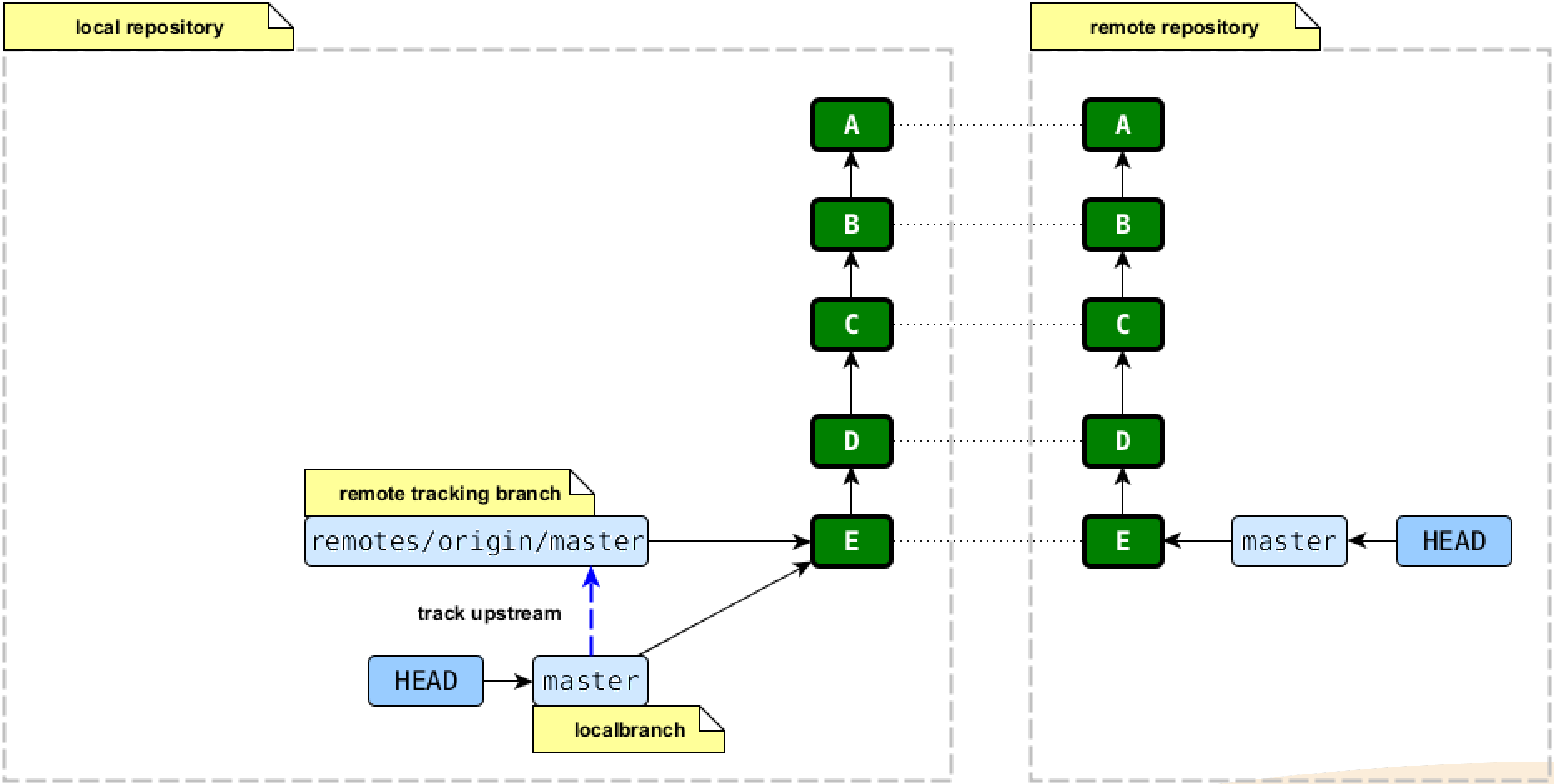
GIT push / success case / 3 of 5



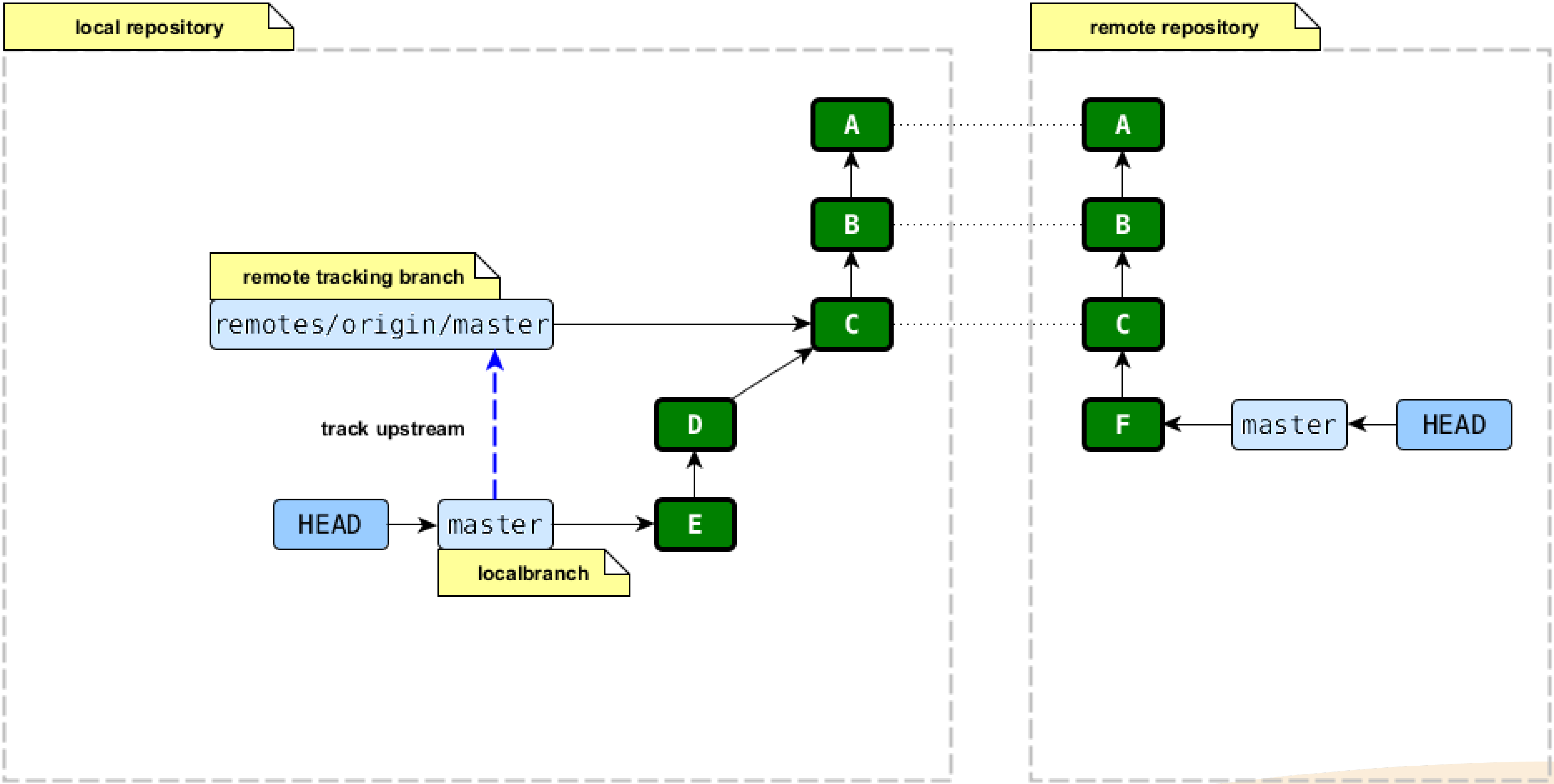
GIT push / success case / 4 of 5



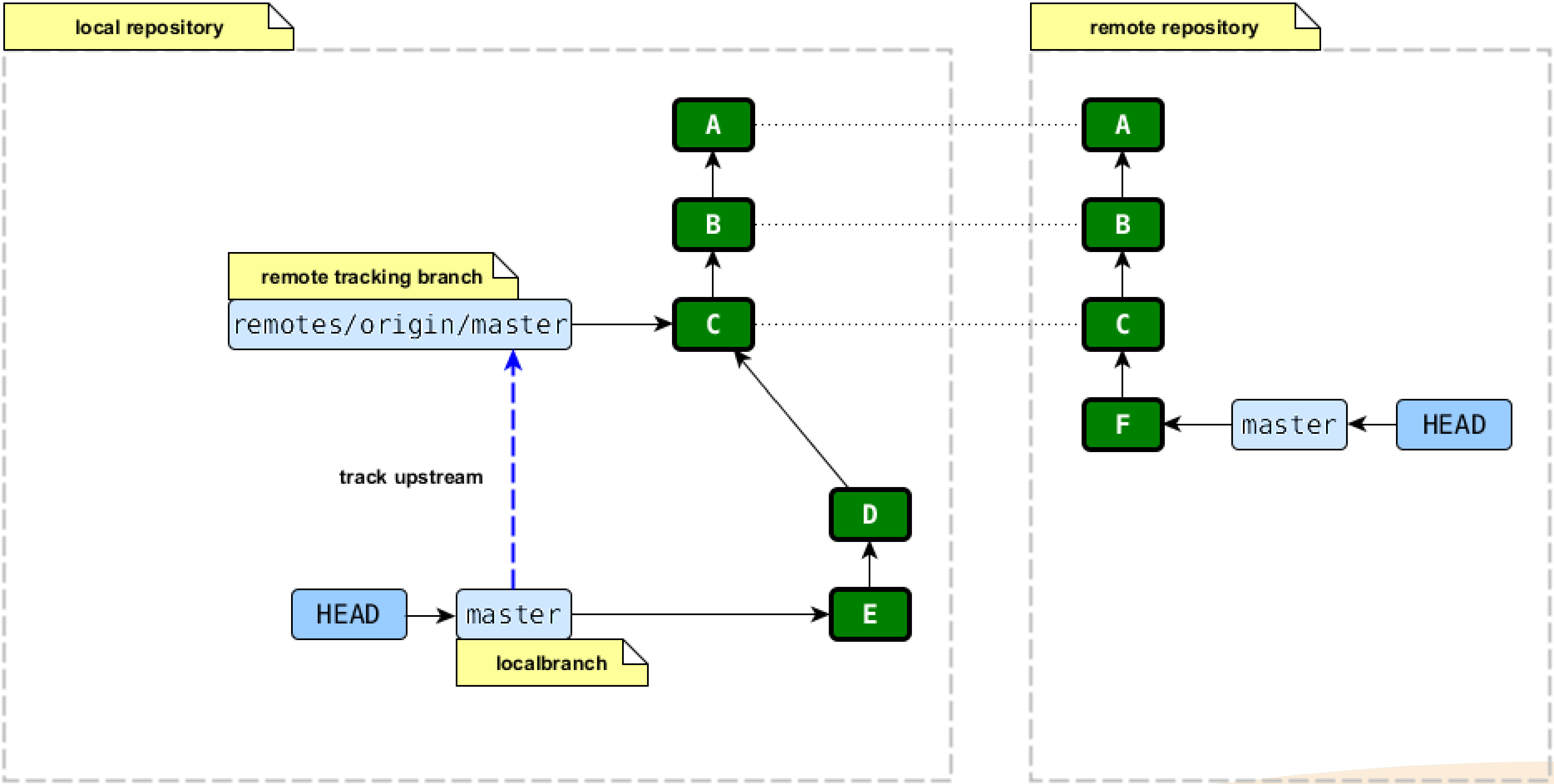
GIT push / success case / 5 of 5



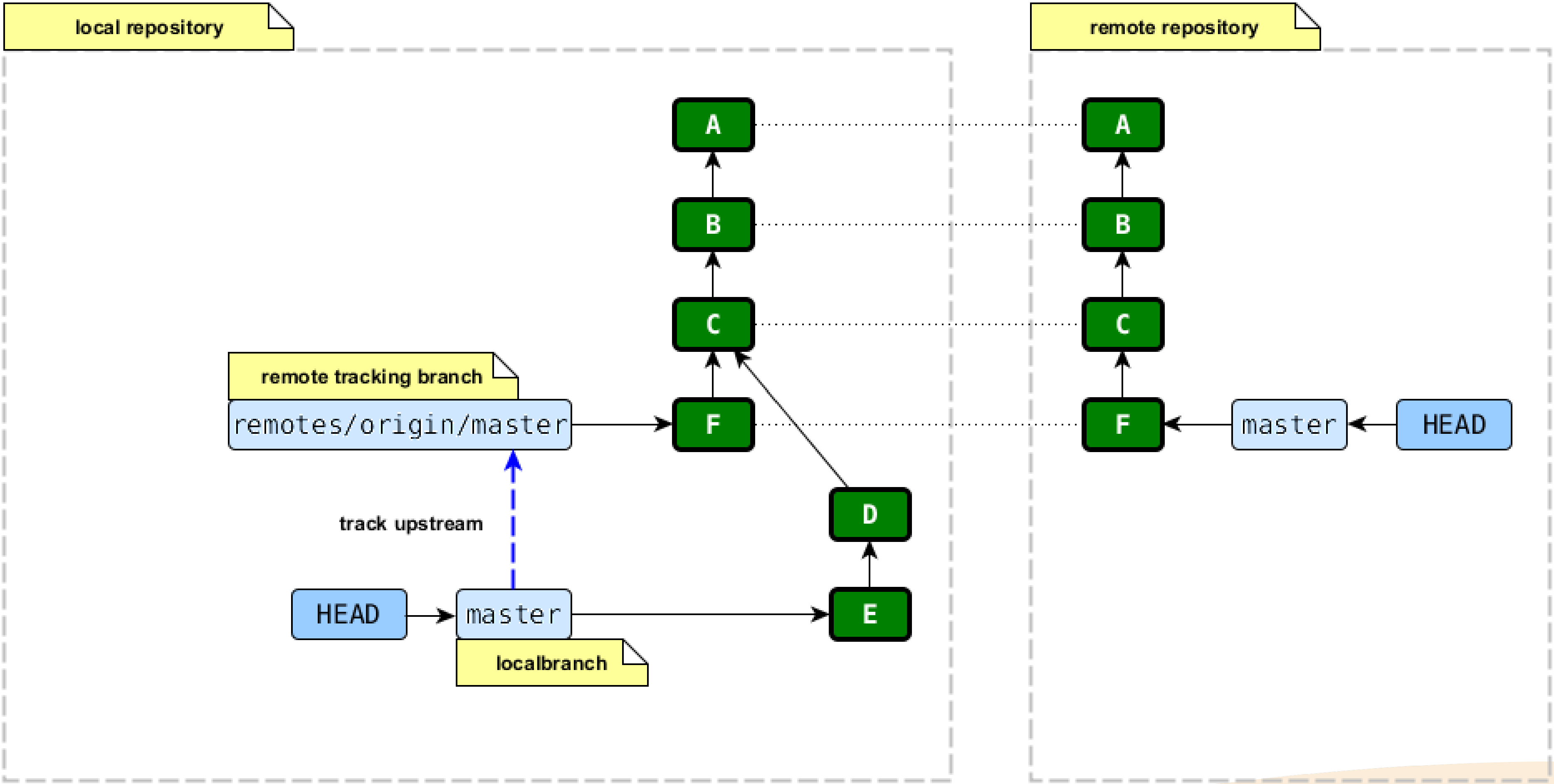
GIT push / error case / 1 of 4



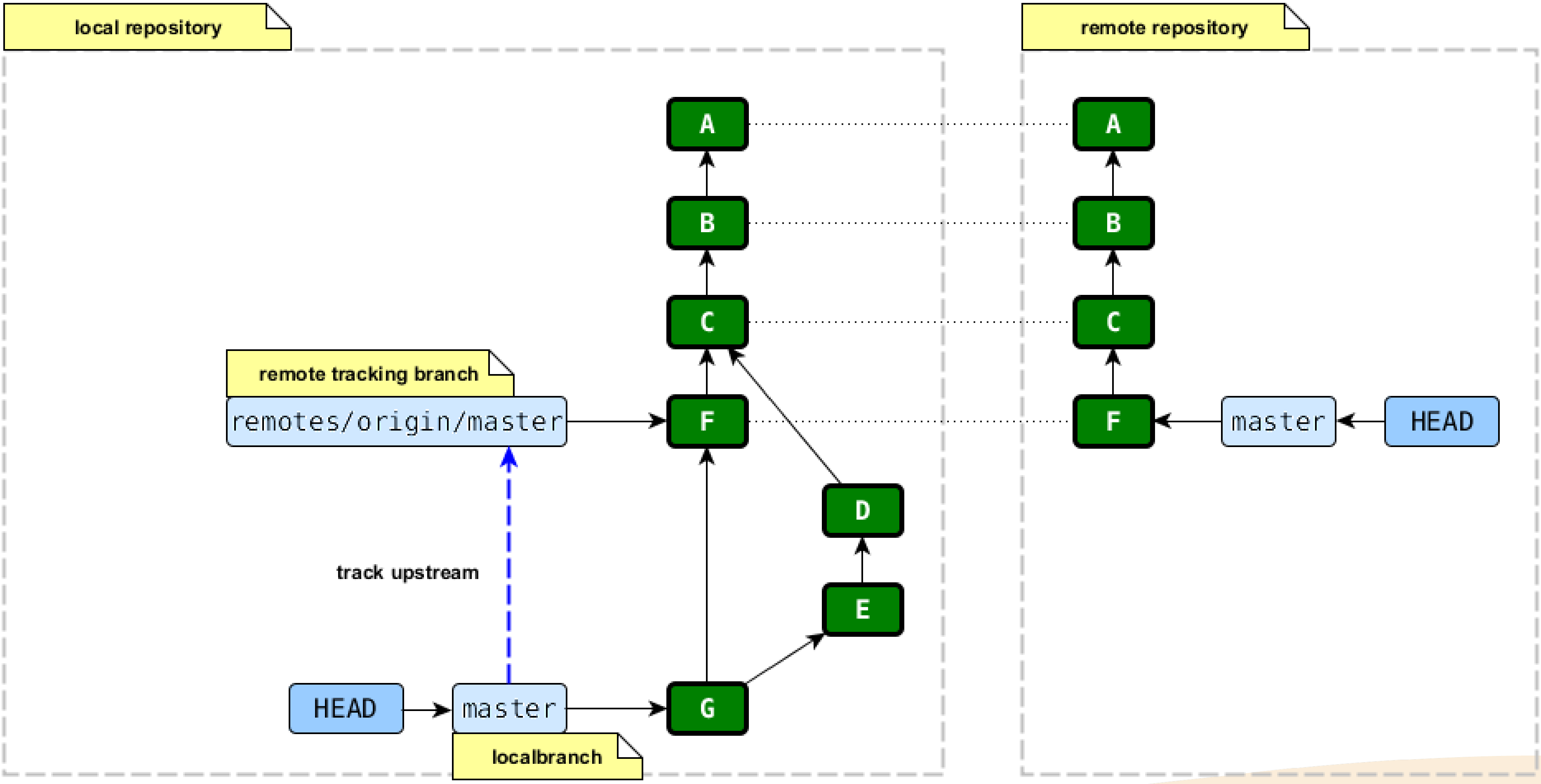
GIT push / error case / 2 of 4



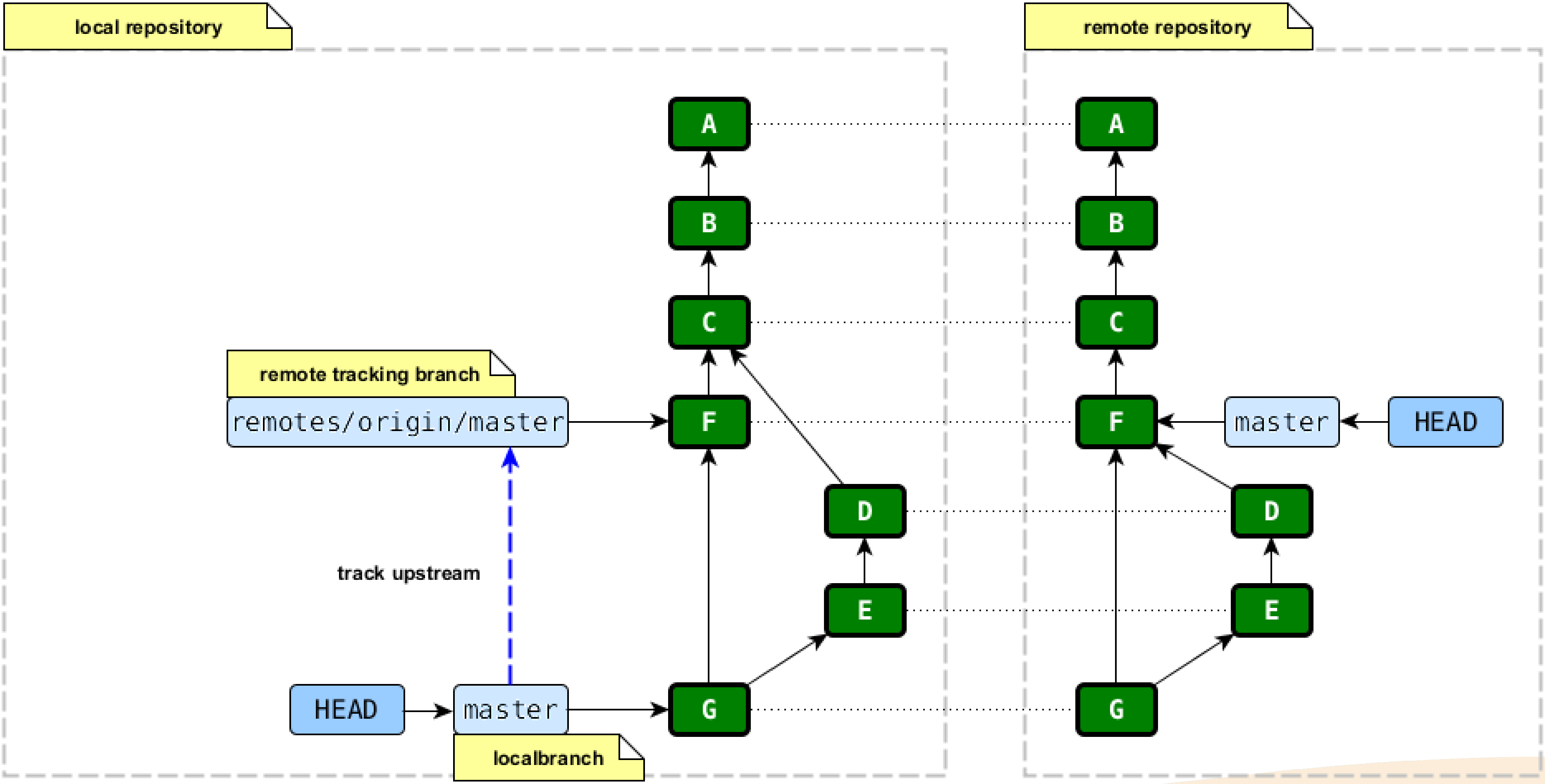
GIT push / error case / 3 of 4 / fetch



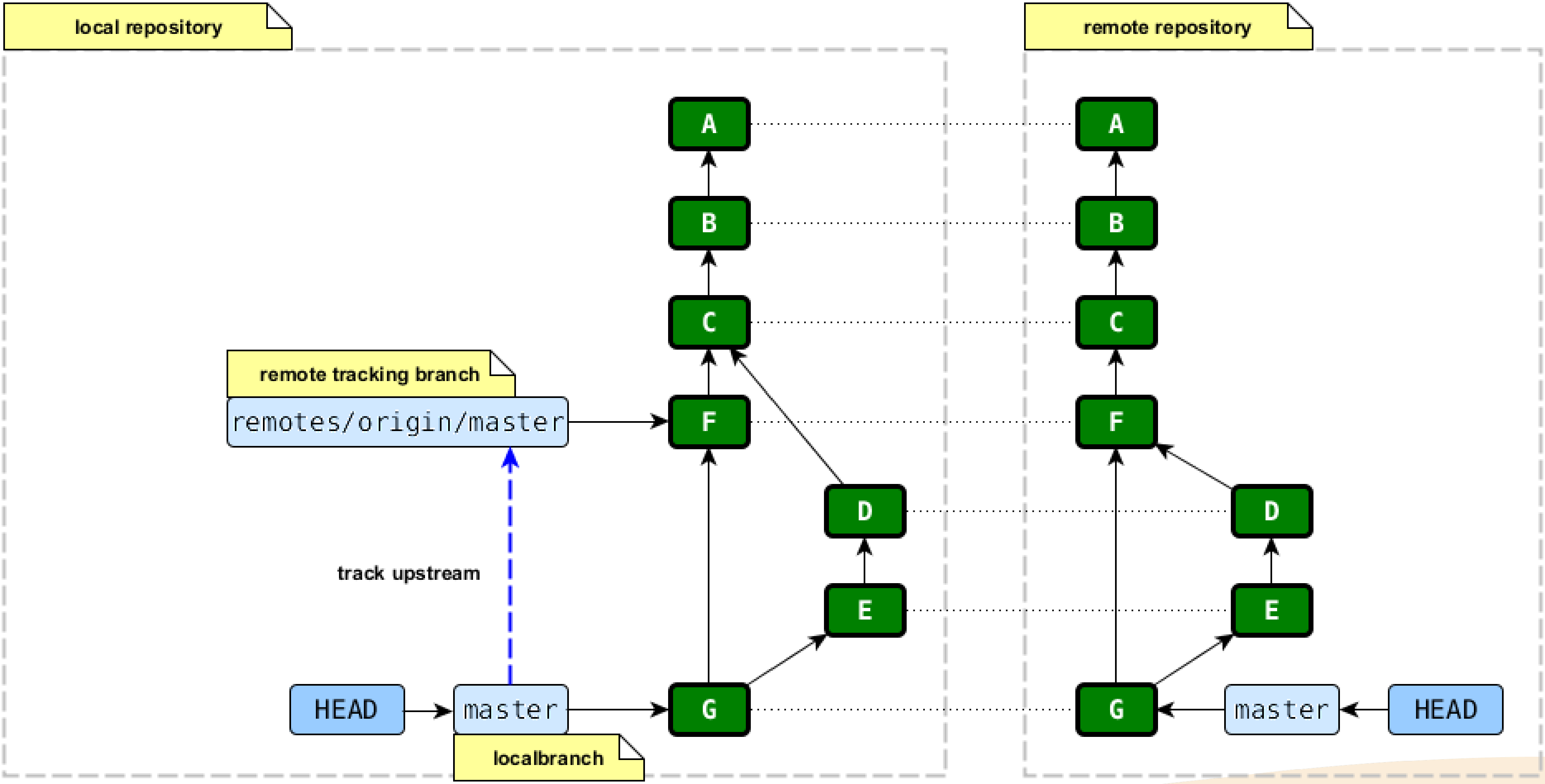
GIT push / error case / 4 of 4 / merge 1 of 4



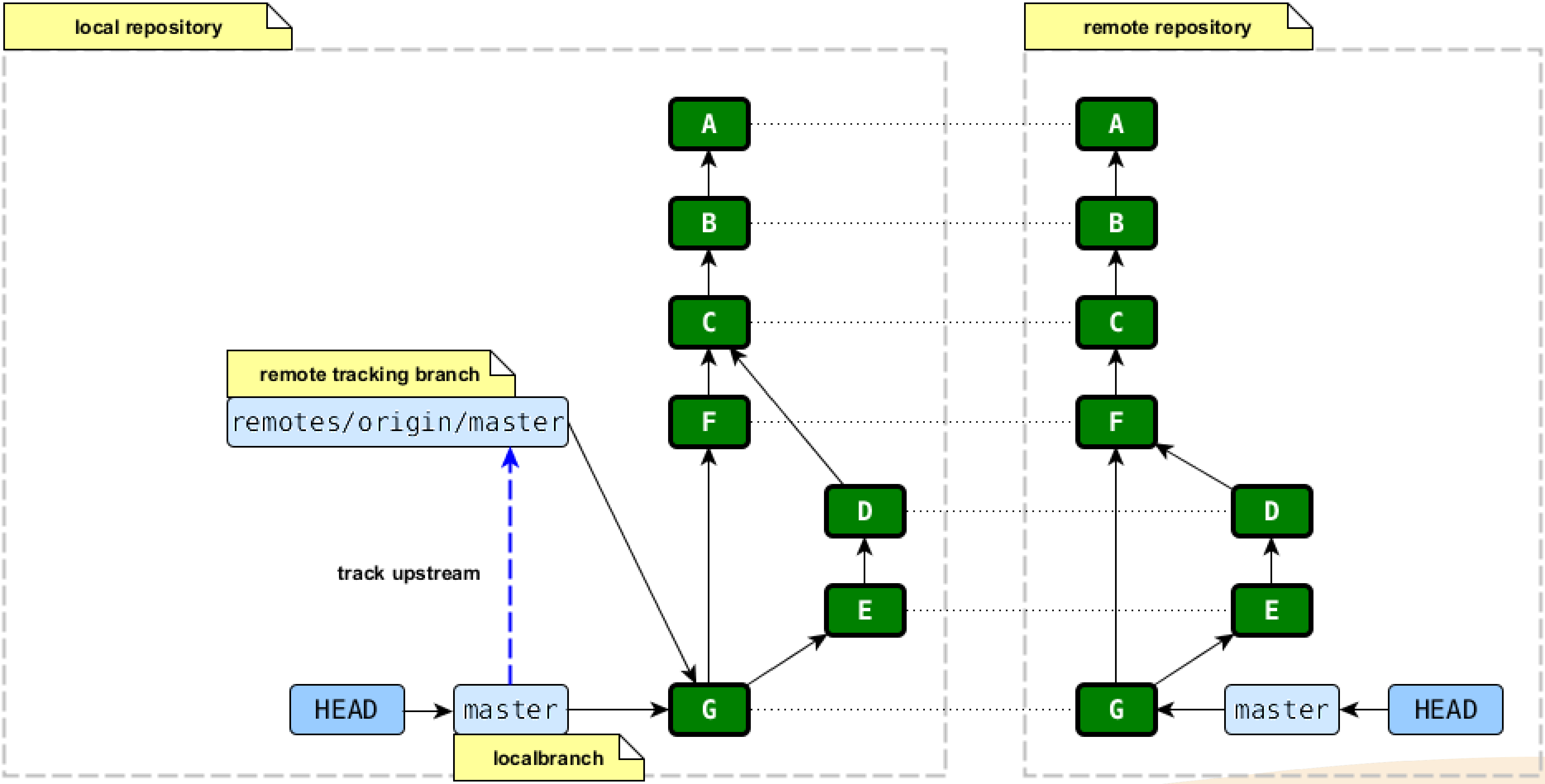
GIT push / error case / 4 of 4 / merge 2 of 4



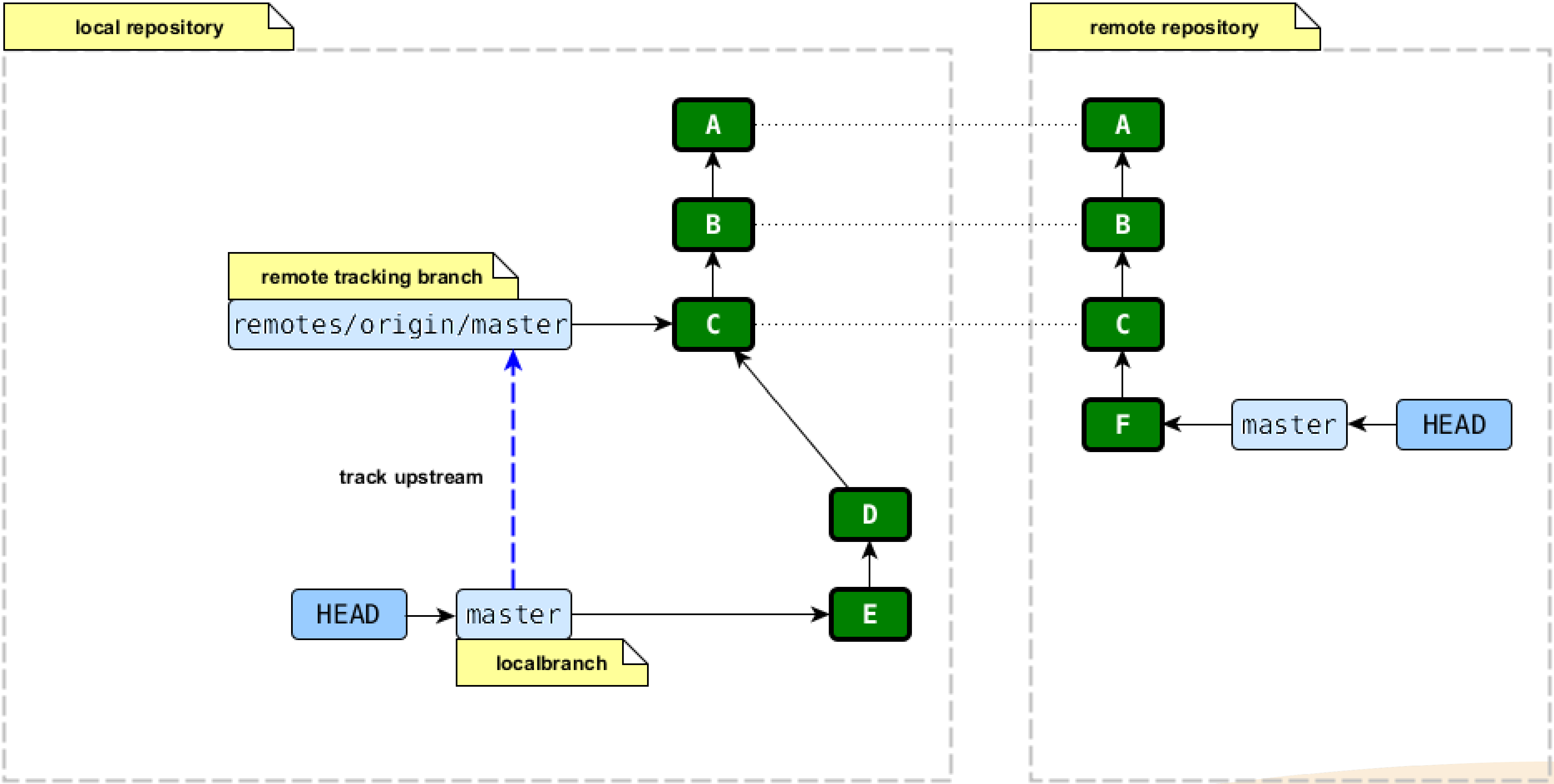
GIT push / error case / 4 of 4 / merge 3 of 4



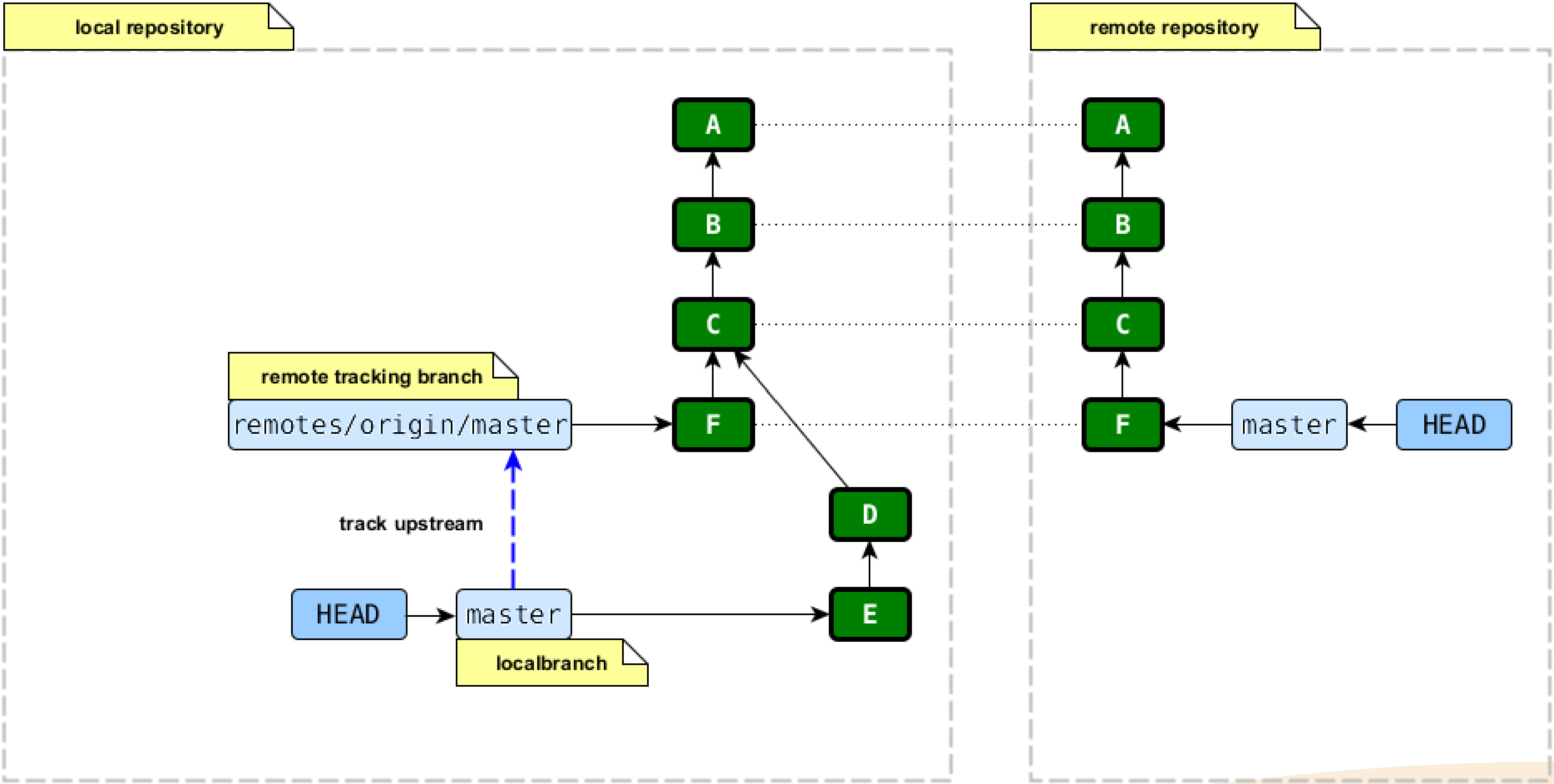
GIT push / error case / 4 of 4 / merge 4 of 4



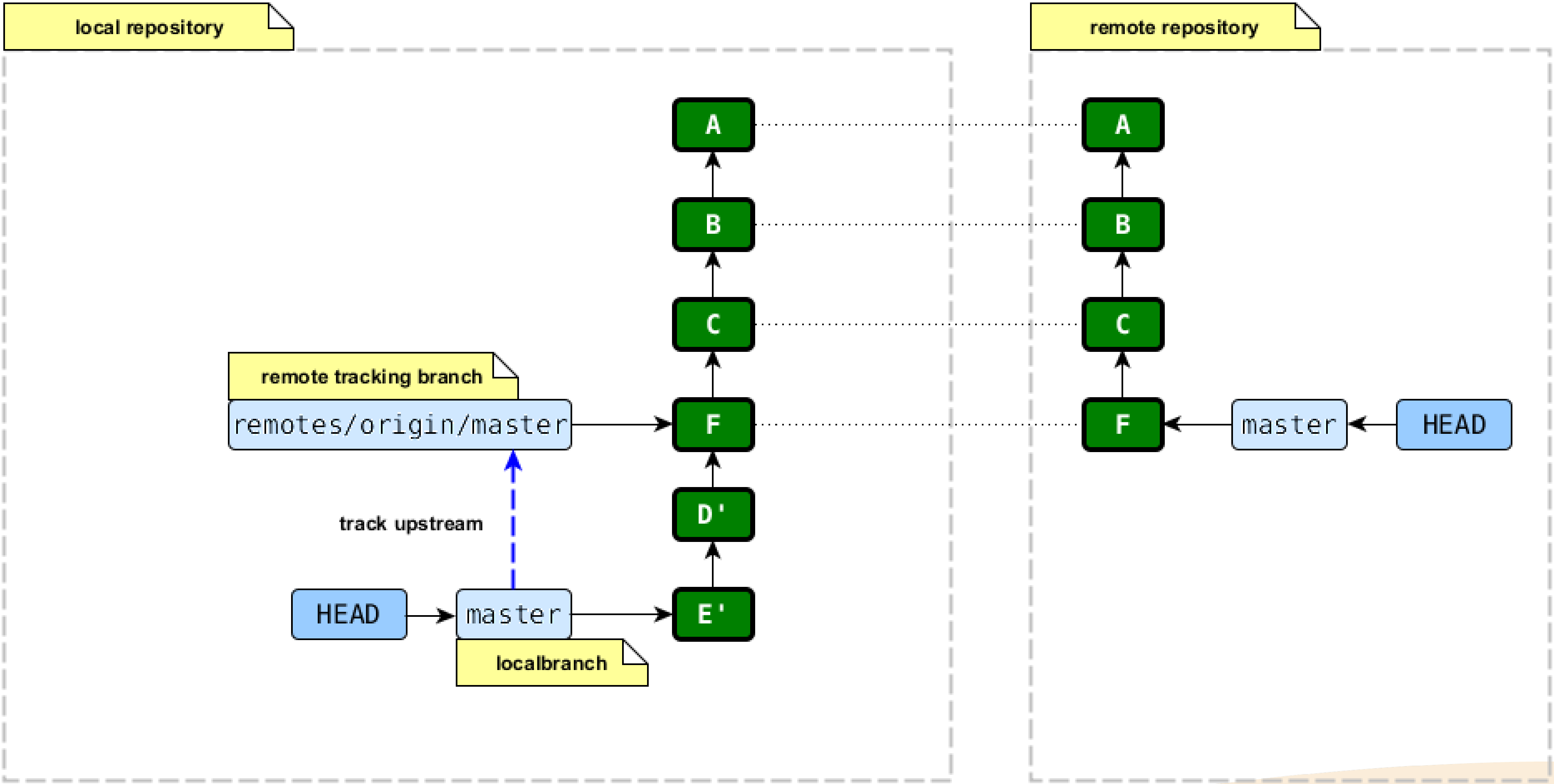
GIT push / error case / 2 of 4



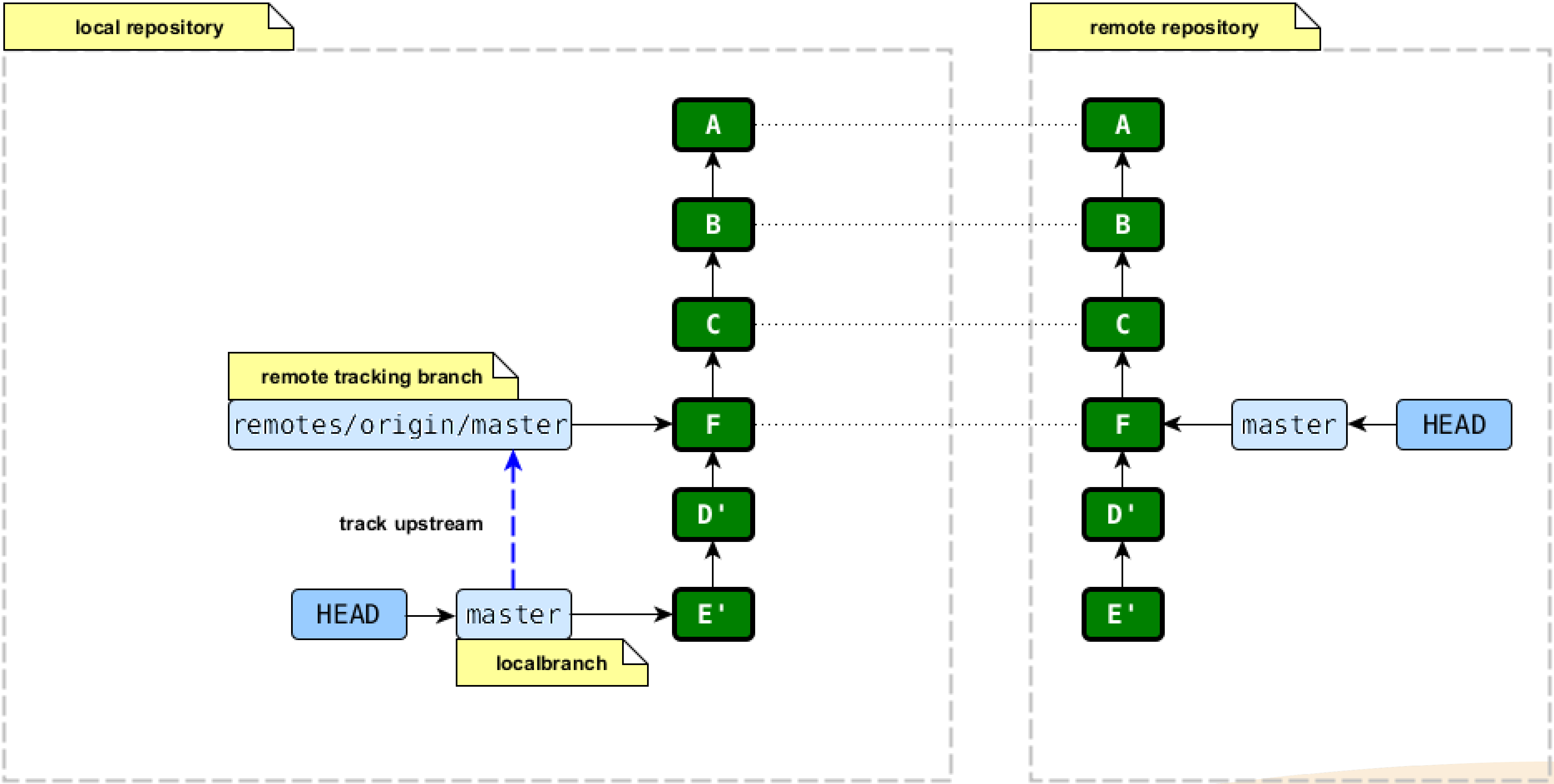
GIT push / error case / 3 of 4 / fetch



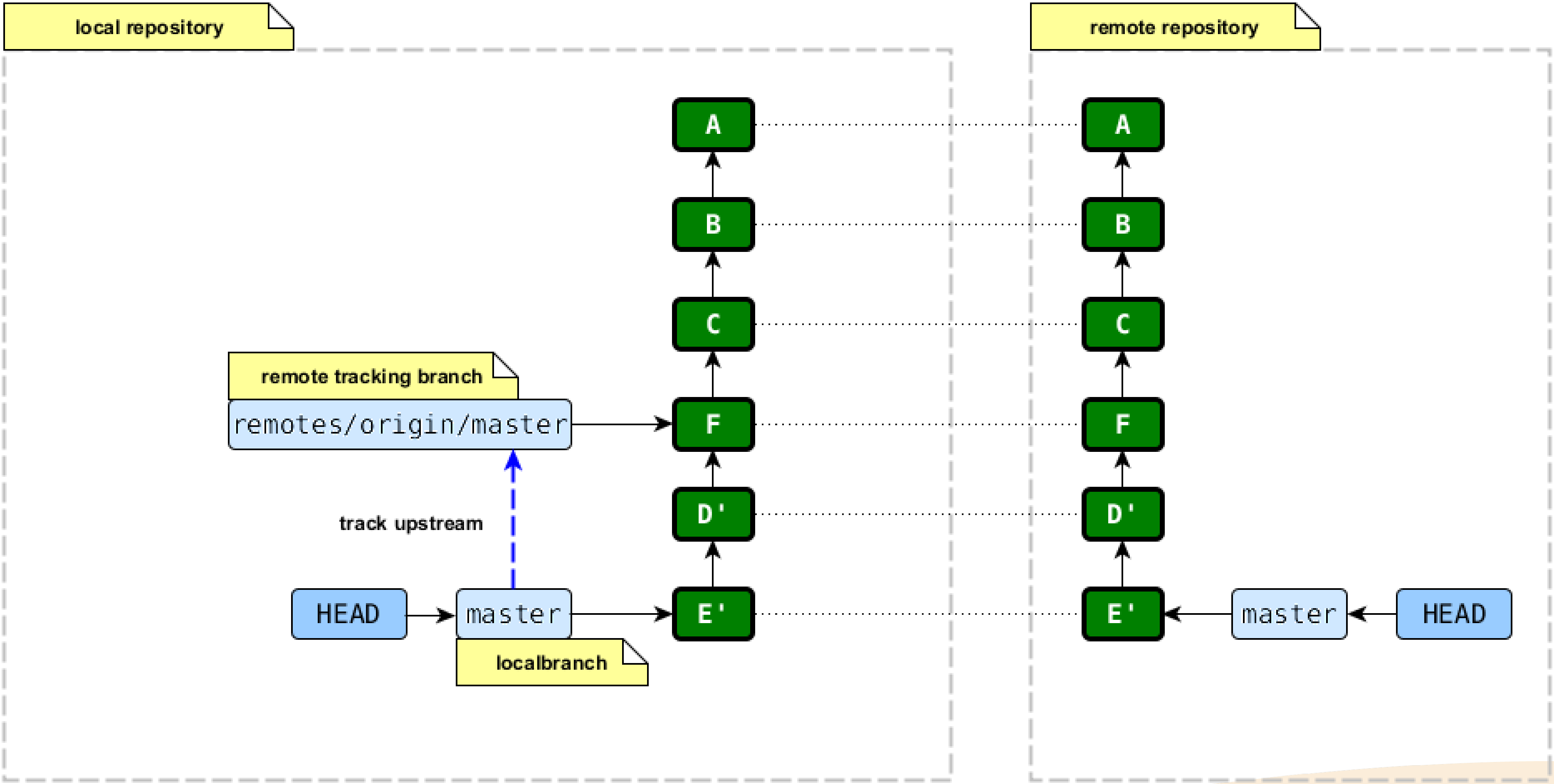
GIT push / error case / 4 of 4 / rebase 1 of 4



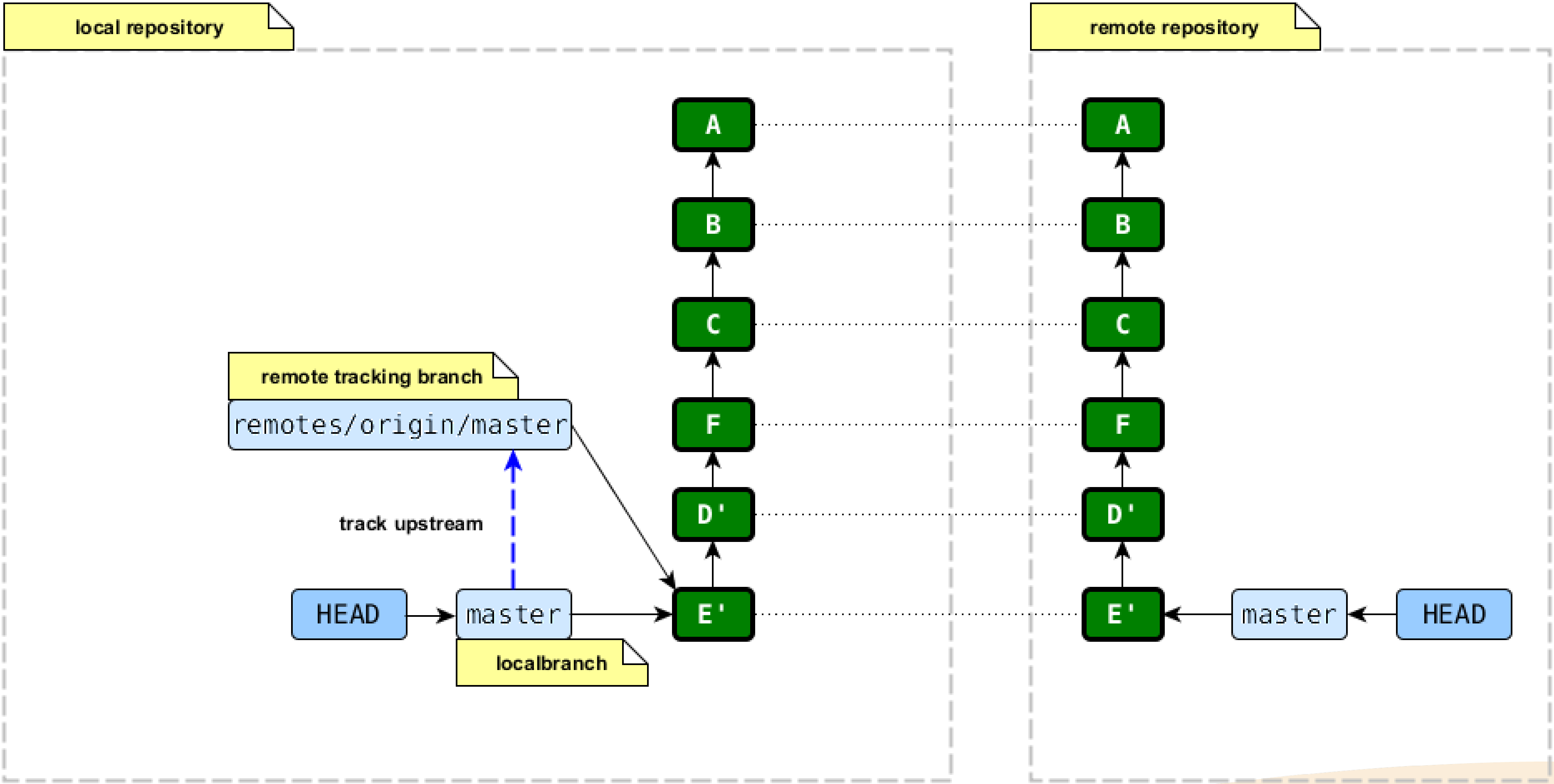
GIT push / error case / 4 of 4 / rebase 2 of 4



GIT push / error case / 4 of 4 / rebase 3 of 4



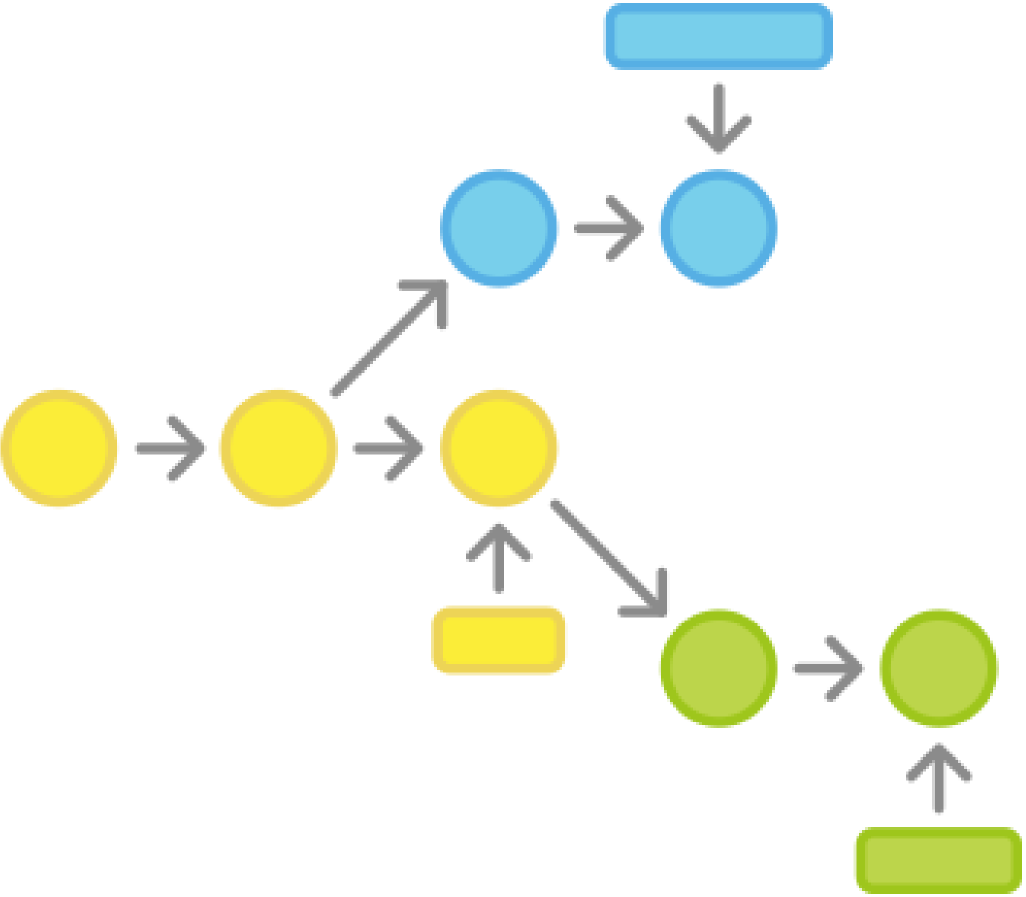
GIT push / error case / 4 of 4 / rebase 4 of 4



GIT pull vs. fetch

- "git fetch" brings commits from the remote repository into the local repository
- "git pull" == "git fetch" + "git merge"
- "git pull --rebase" == "git fetch" + "git rebase"

Workflows

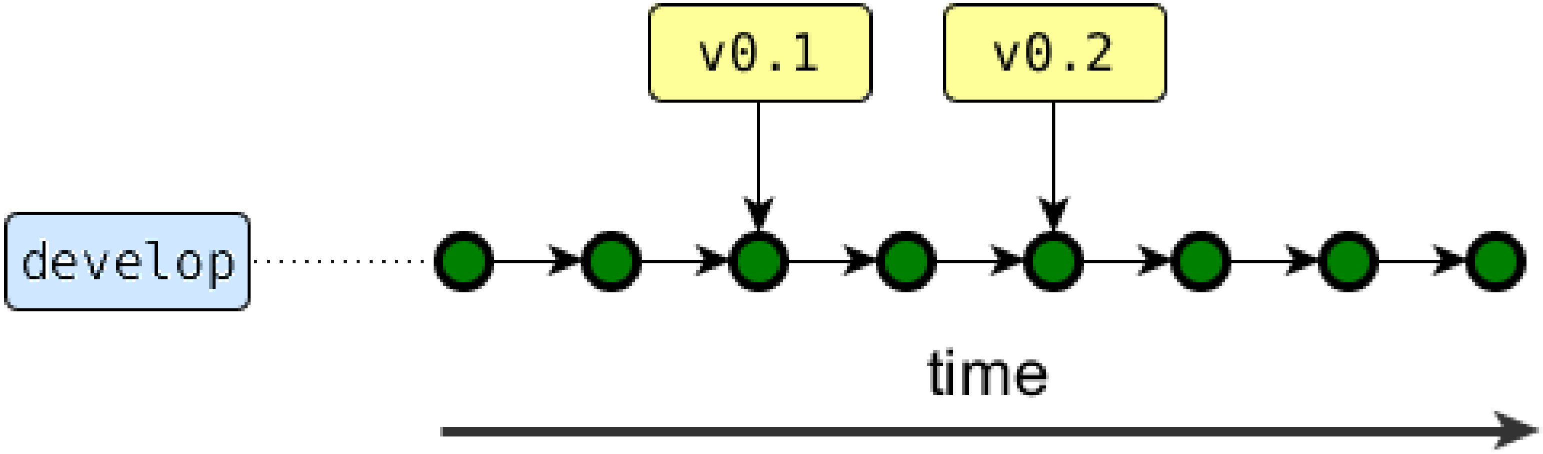


Workflows

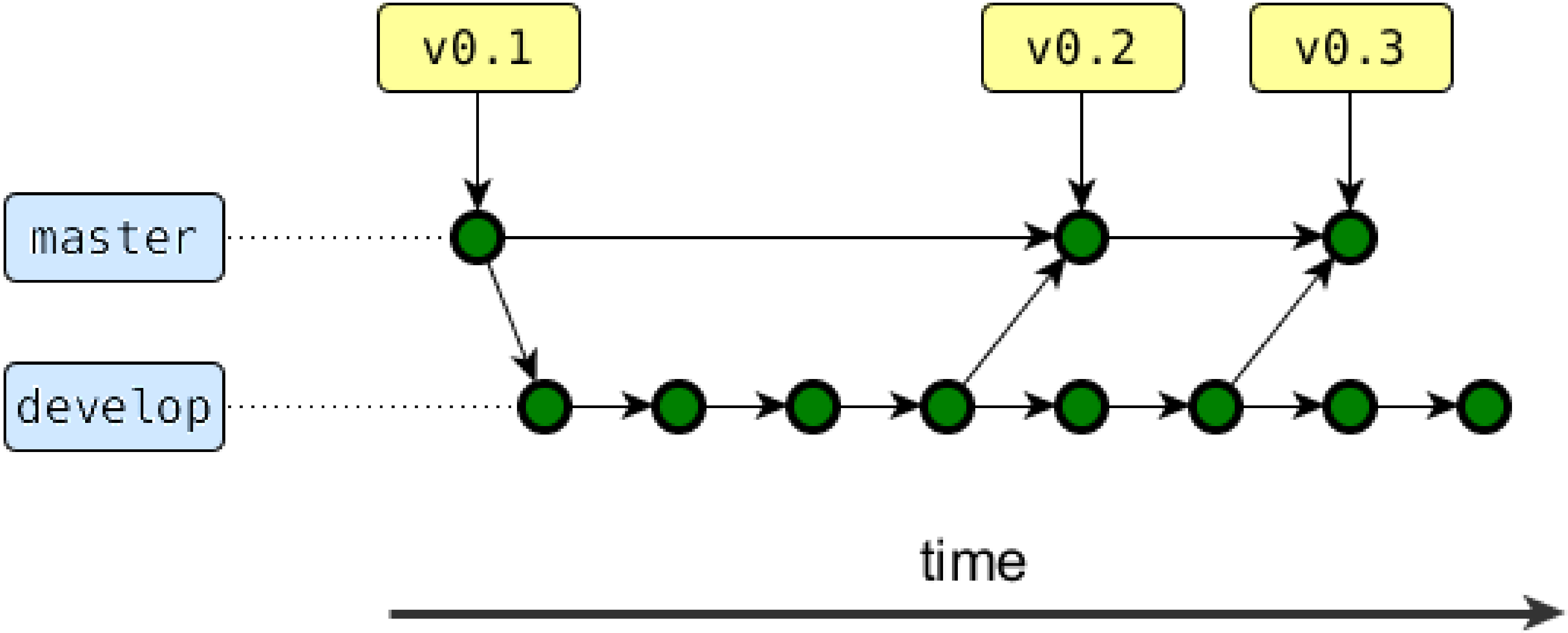
- only one integration branch
- one separate release (stable) branch
- separate release branches
- separate release branches without stable branch
- separate hotfix branches
- separate feature branches
- GIT Flow
    - diagram
    - resources

Workflows / only one integration branch

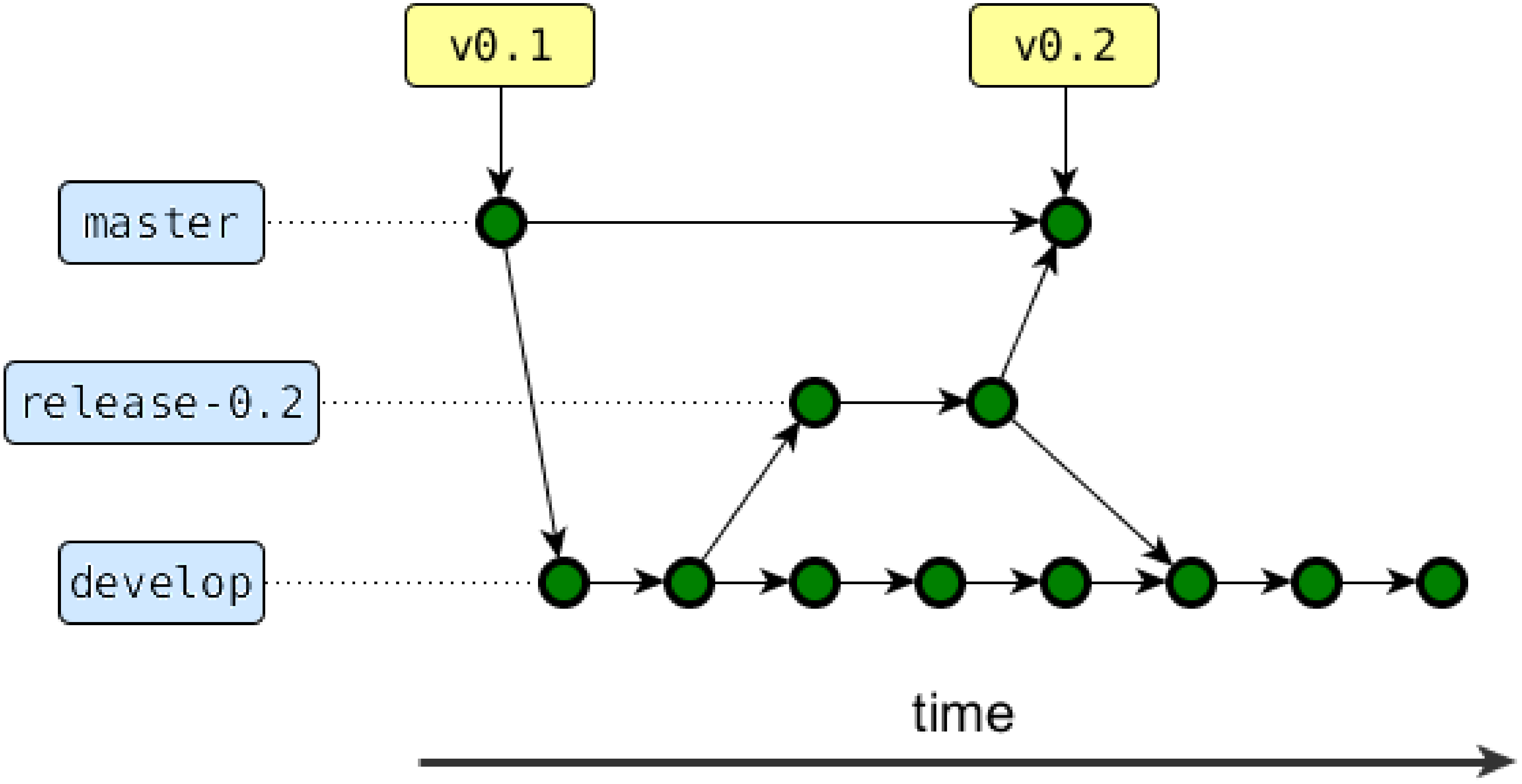
- only one branch where everybody works
  - simple
  - does not work well when not all team is releasing



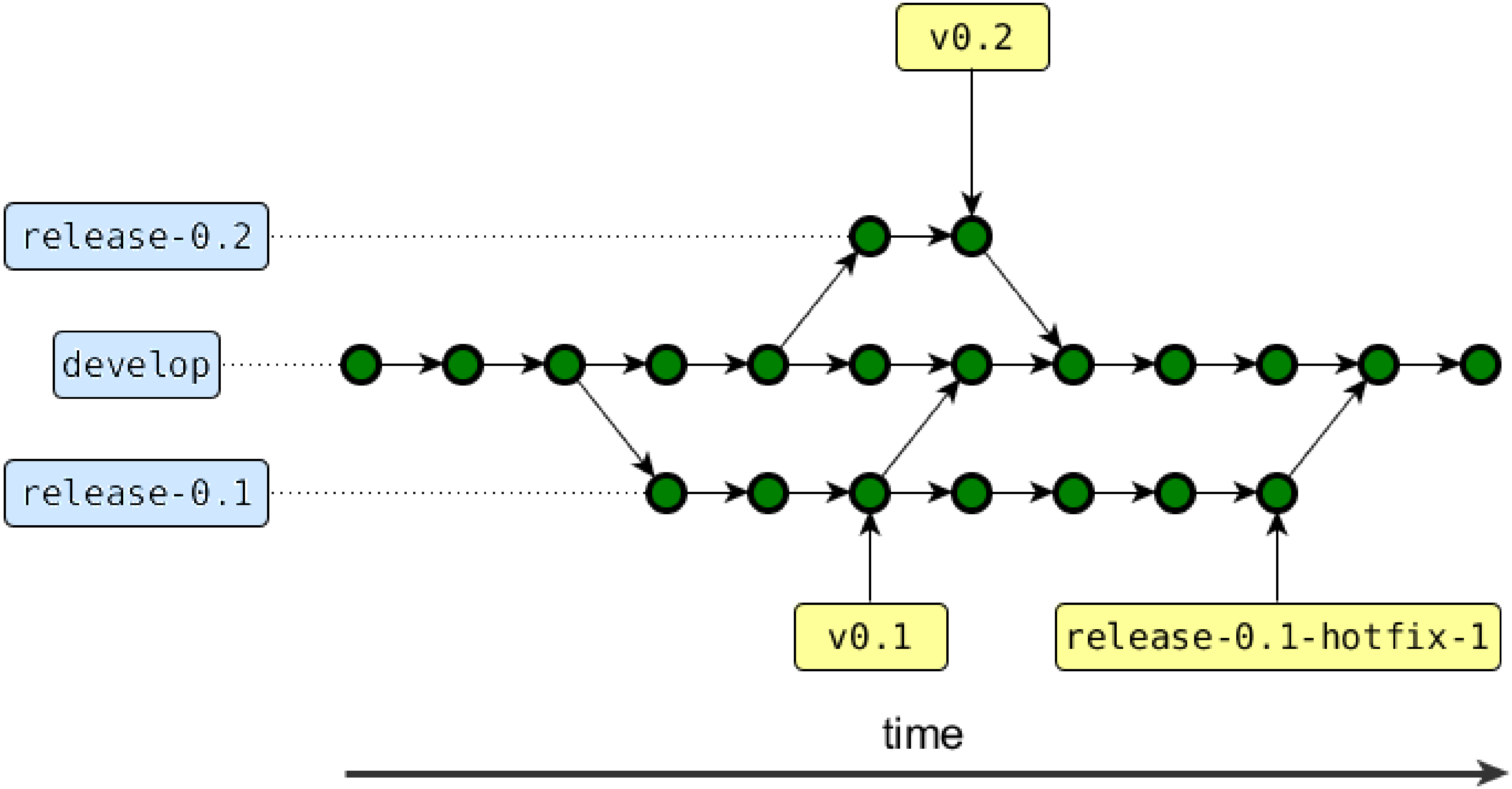
Workflows / one separate release (stable) branch



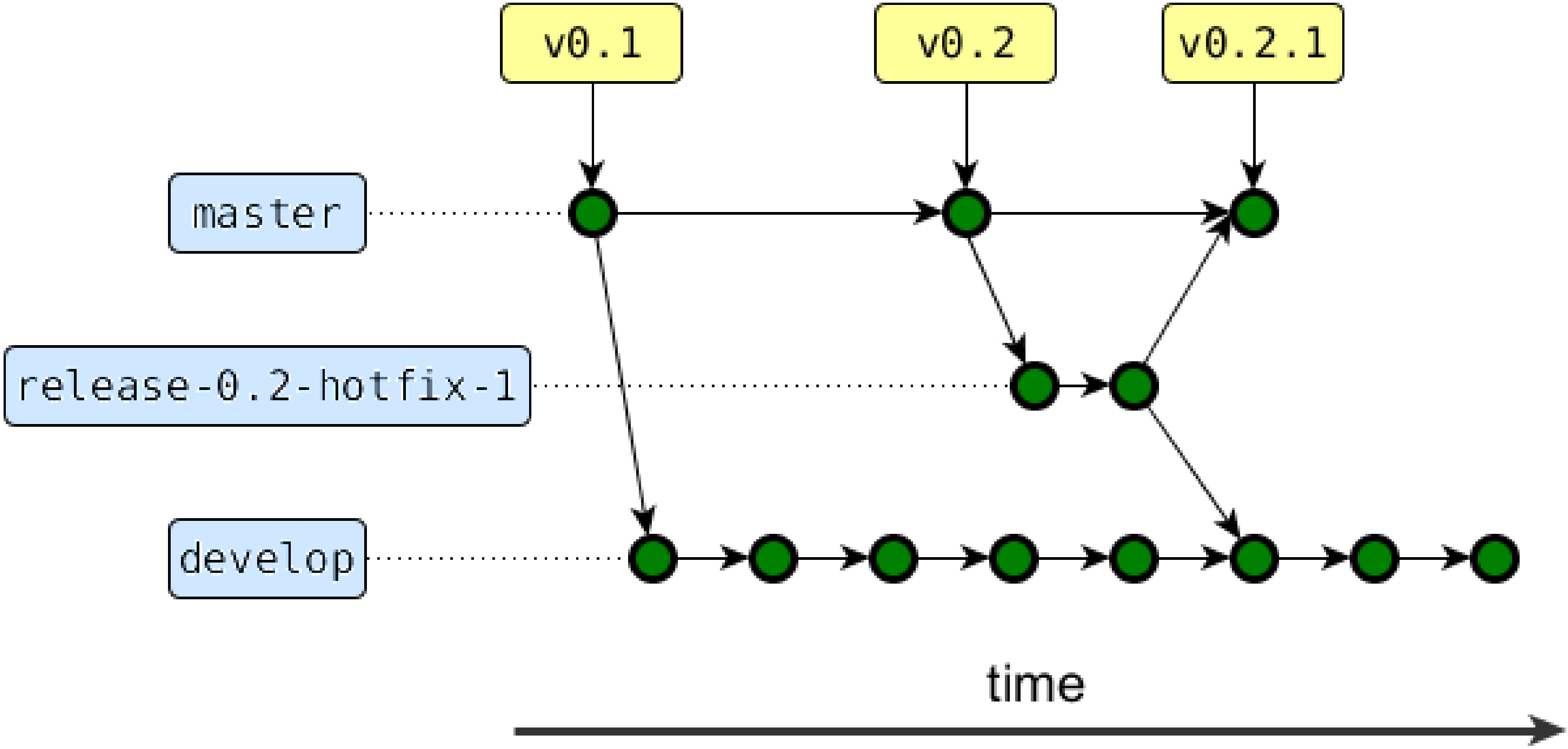
Workflows / multiple release branches



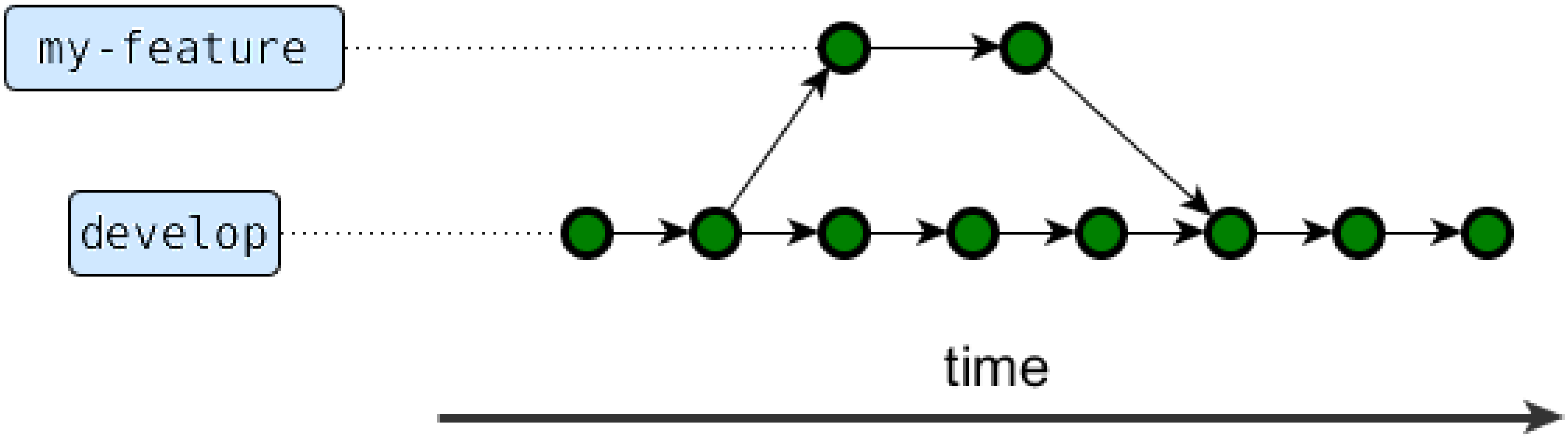
Workflows / separate release branches without stable branch



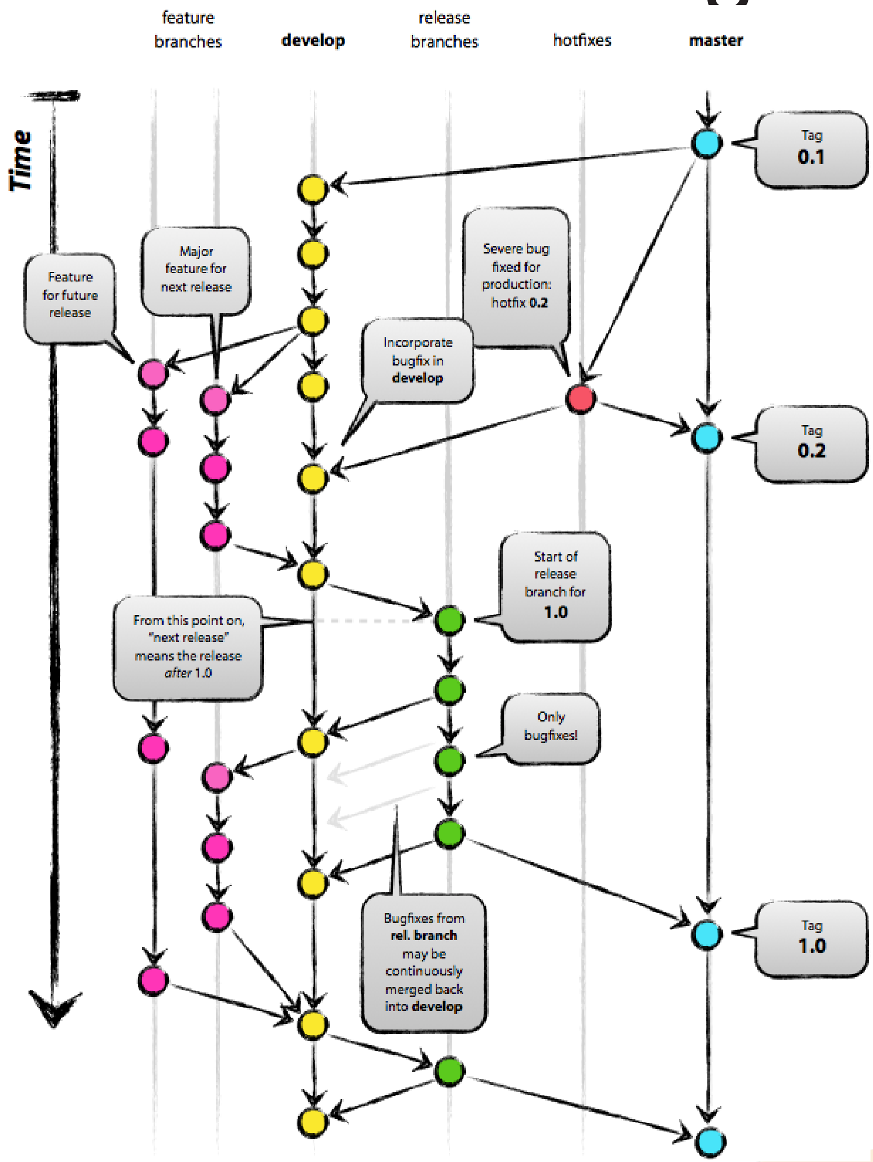
Workflows / separate hotfix branches



Workflows / separate feature branches



Workflows / GIT Flow / diagram


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Workflows / GIT Flow / resources

- Original GIT Flow article:
  http://nvie.com/posts/a-successful-git-branching-model/
- GIT Flow supporting extension (adds "git flow" command):
  https://github.com/nvie/gitflow
- GIT Flow completion (adds command line bash completion for the GIT flow supporting extension):
  https://github.com/bobthecow/git-flow-completion