

# **Git & GitHub**

**3DC - Early Mat Workshop Series**

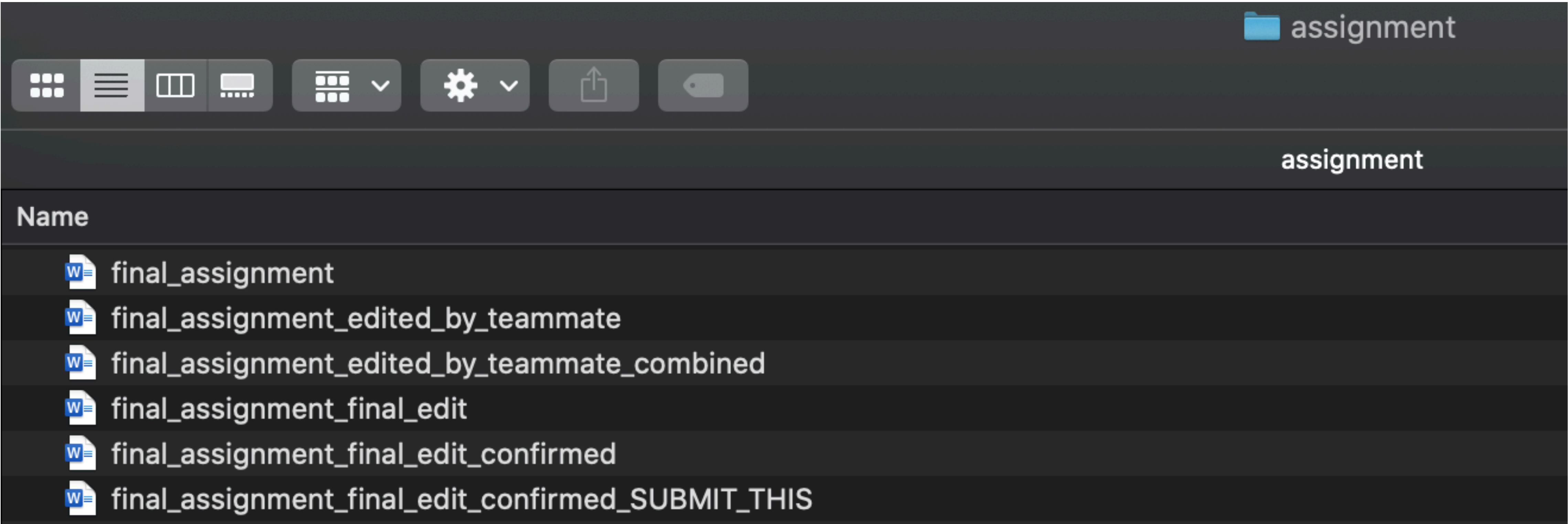
**[https://3dc.opensutd.org/Introduction\\_to\\_Programming/](https://3dc.opensutd.org/Introduction_to_Programming/)**

# What is Git and why is it important?



Standard version control system used by most developers and companies

# What is Git and why is it important?



# What is Git and why is it important?



Version control system

“Saves your game progress midway”



Host git repositories

“Google drive to store git repositories”

# How does git work?



## Legend

 Snapshot

 Pointer to parent snapshot

# Git data model

## Common terminology

 Snapshot

Files

Folders

## Git terminology

 Commit

Blob

Tree

# Git data model



Id: sha1 hash (40 char string)

Author: developerName

Parent: String (Hash String)

Commit Message: String

Snapshot: Tree

The commit that precedes the current commit

## Key takeaway

- Every commit has information of the current state of the files.
- It can be identified by a unique ID.
- It contains a commit message that describes the current state.
  - (eg. Added new feature, fixed a bug)

# Git key concepts

## Staging Area

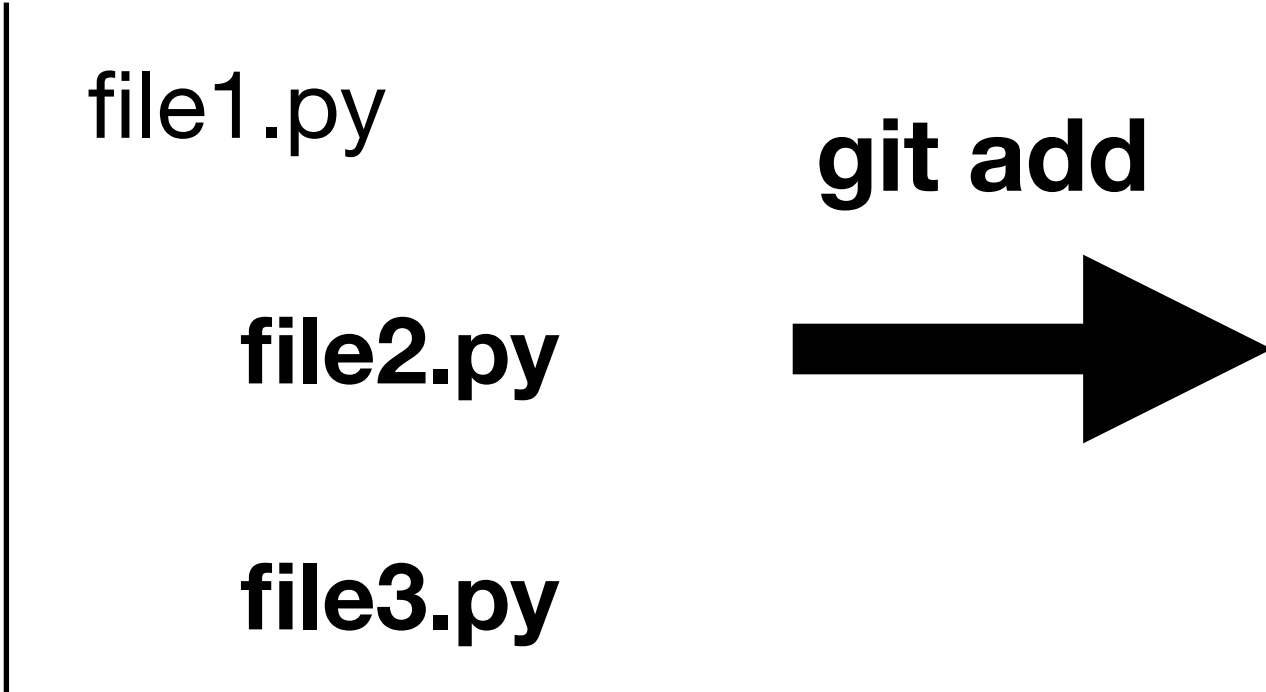
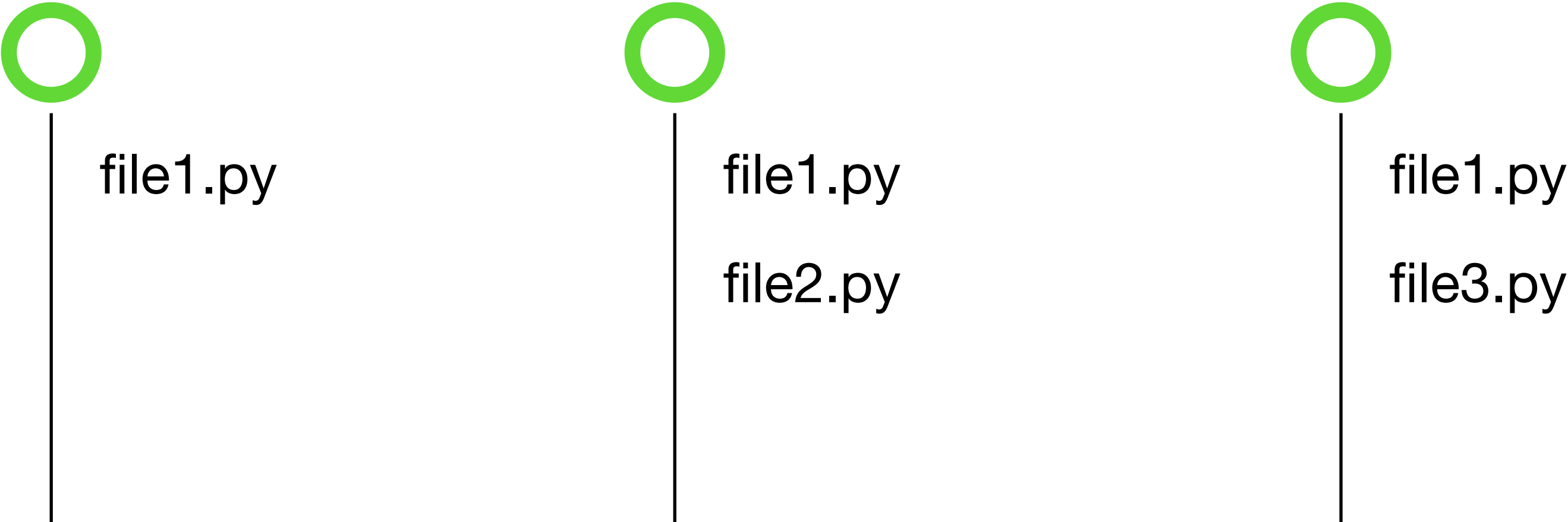
We add files into the staging area to tell git that those are the specific files that I want to create a snapshot / commit of.



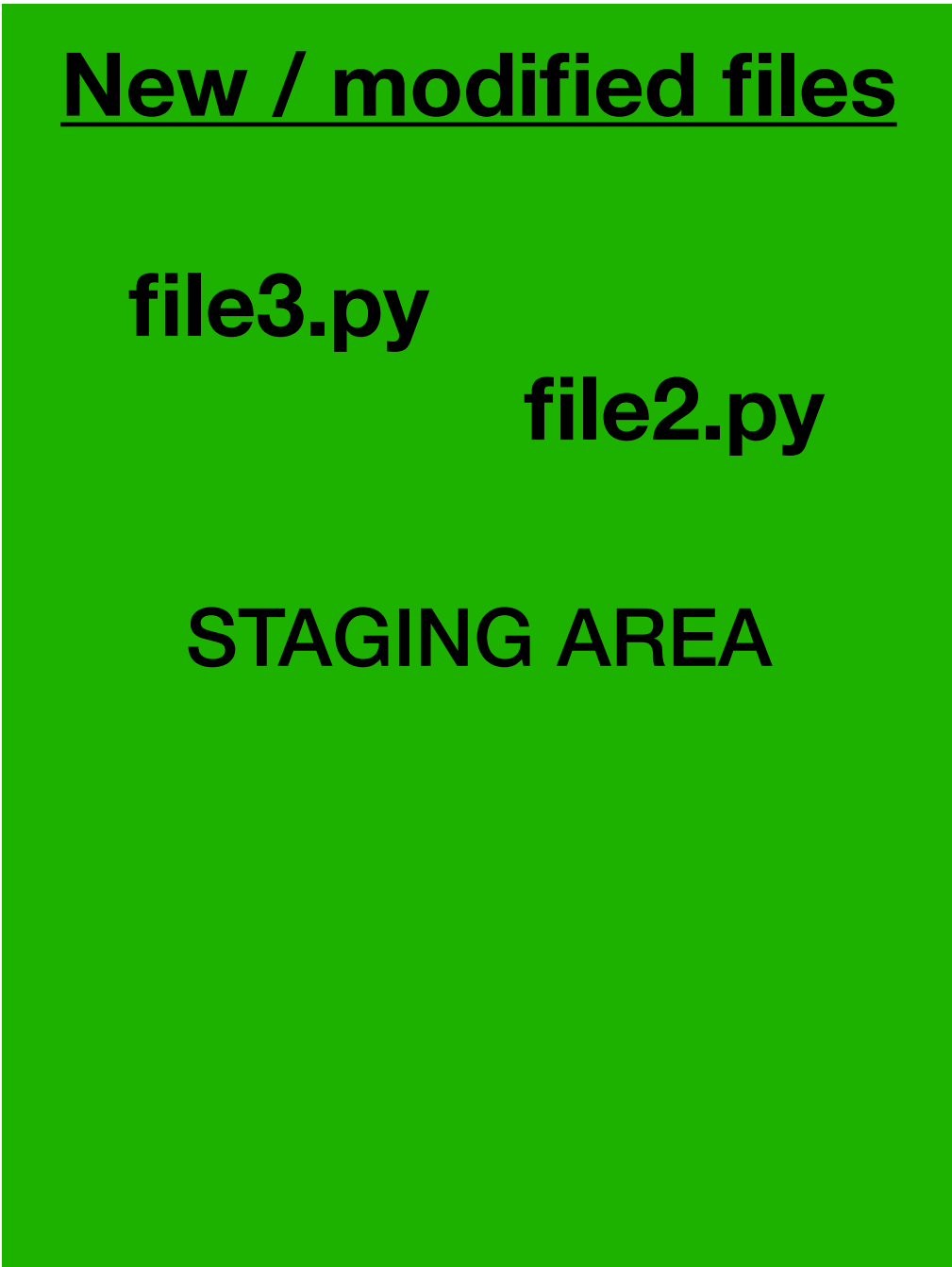
# Git key concepts

## Staging Area

We want to create commits like this



**git add**



By being able to choose what files goes into the commit, we can segment our work.

# Git commands

## **Staging Area**

git add .

git add <filename>

# Git key concepts

# Commit



# Git commands

## **Staging Area**

git add .

git add <filename>

## **Commit**

git commit -m “message”

# Git key concepts

# Branches



Snapshot

branch: someMeaningfulBranchName



Map meaningful human readable Branch name to a commit id

Id: sha1 hash (40 char string)

Author: developerName

Parent: String (Hash String)

Commit Message: String

Snapshot: Tree

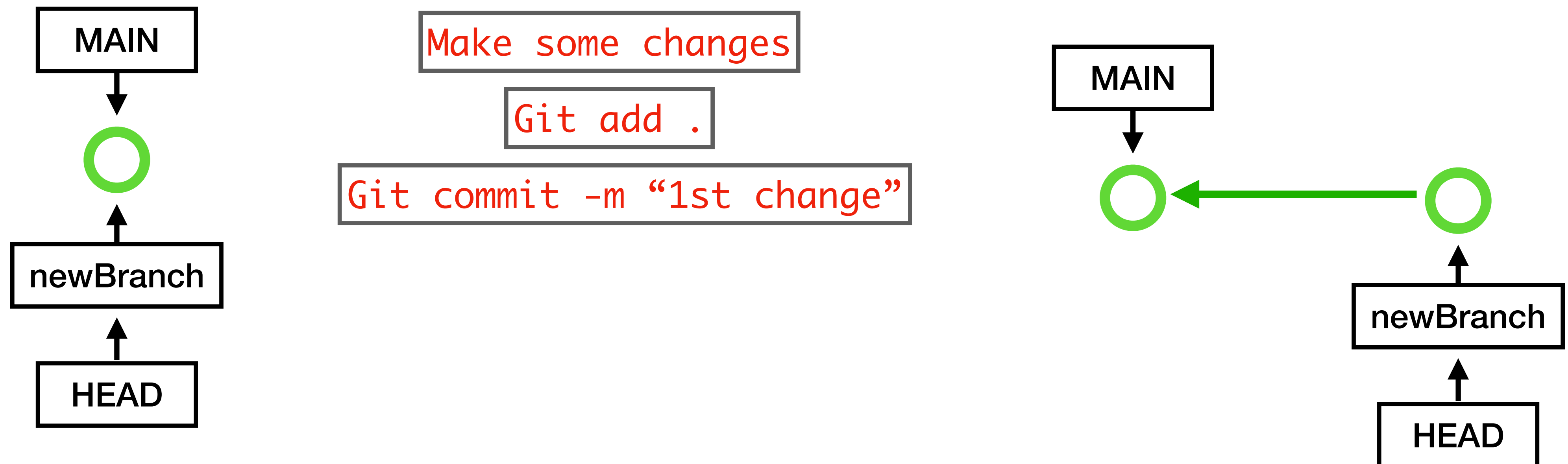
The commit that precedes the current commit

# Git key concepts

## Branches

HEAD branch points to the current branch you are on.

Main / master branch is the default initial branch name. Can be seen as the main line of development where production code lives.



# Git commands

## Staging Area

git add .

git add <filename>

## Commit

git commit -m “message”

## Branches

git branch

git checkout -b <new branch name>

Git key concepts

Merging

fast-forward merge

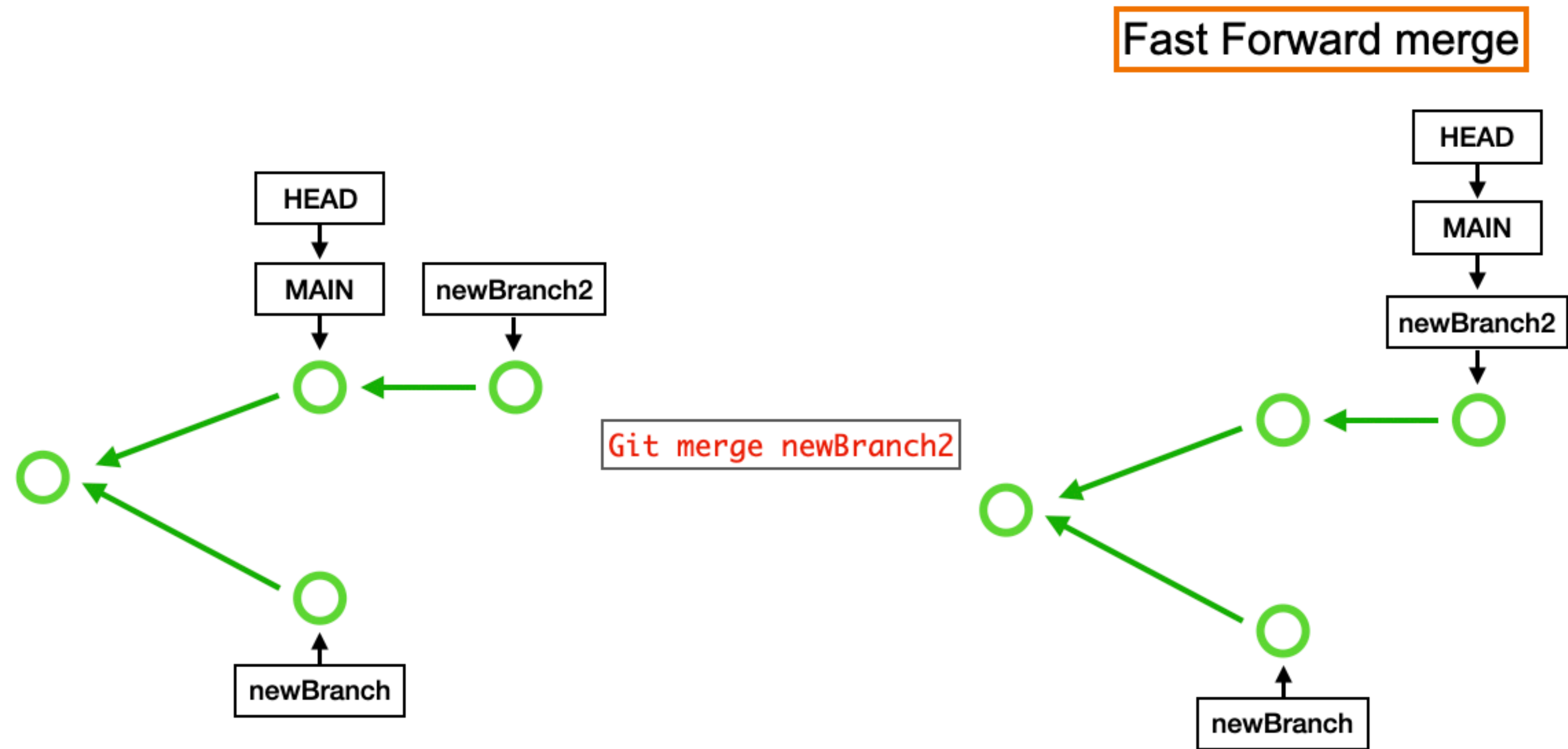
3 way merge



# Git key concepts

# Merging

fast-forward merge

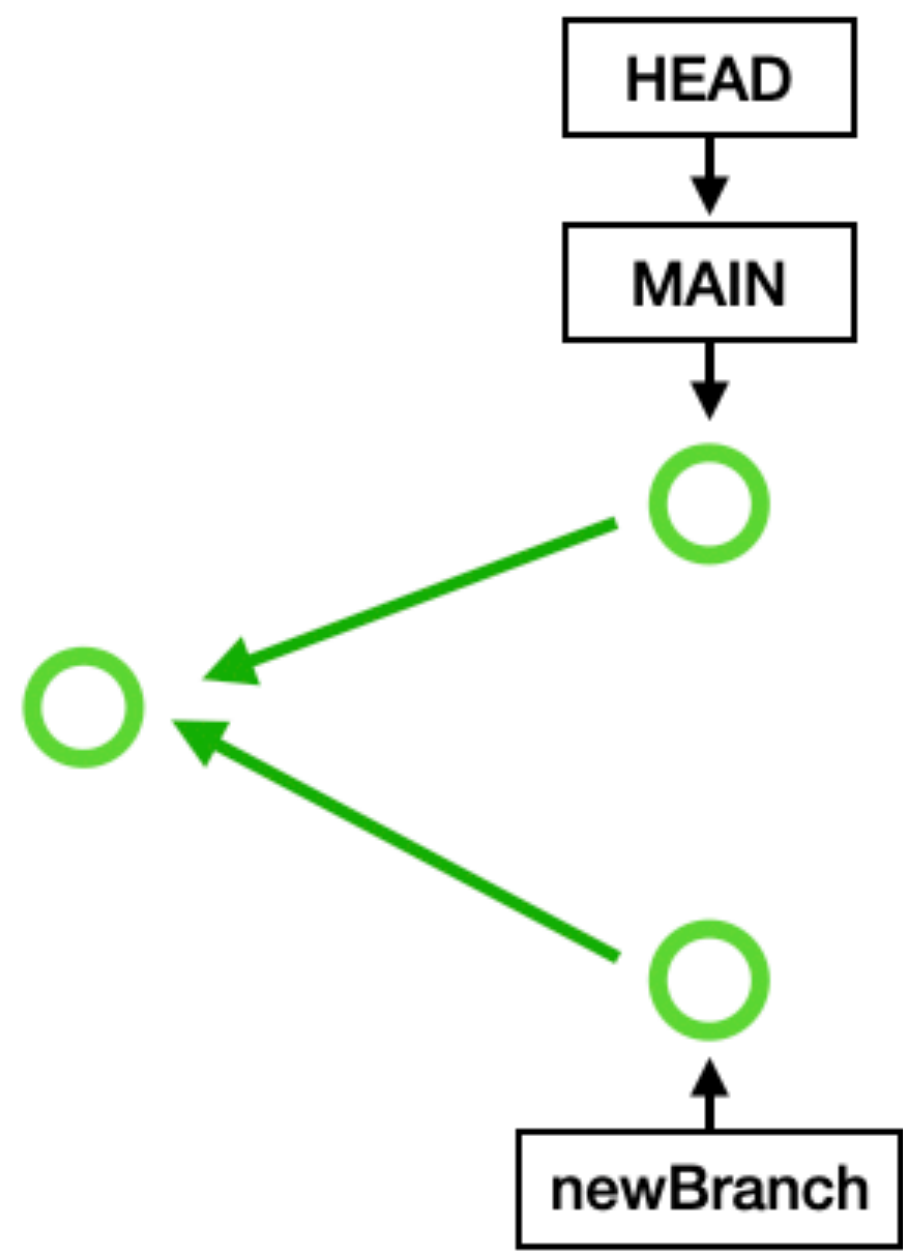


# Git key concepts

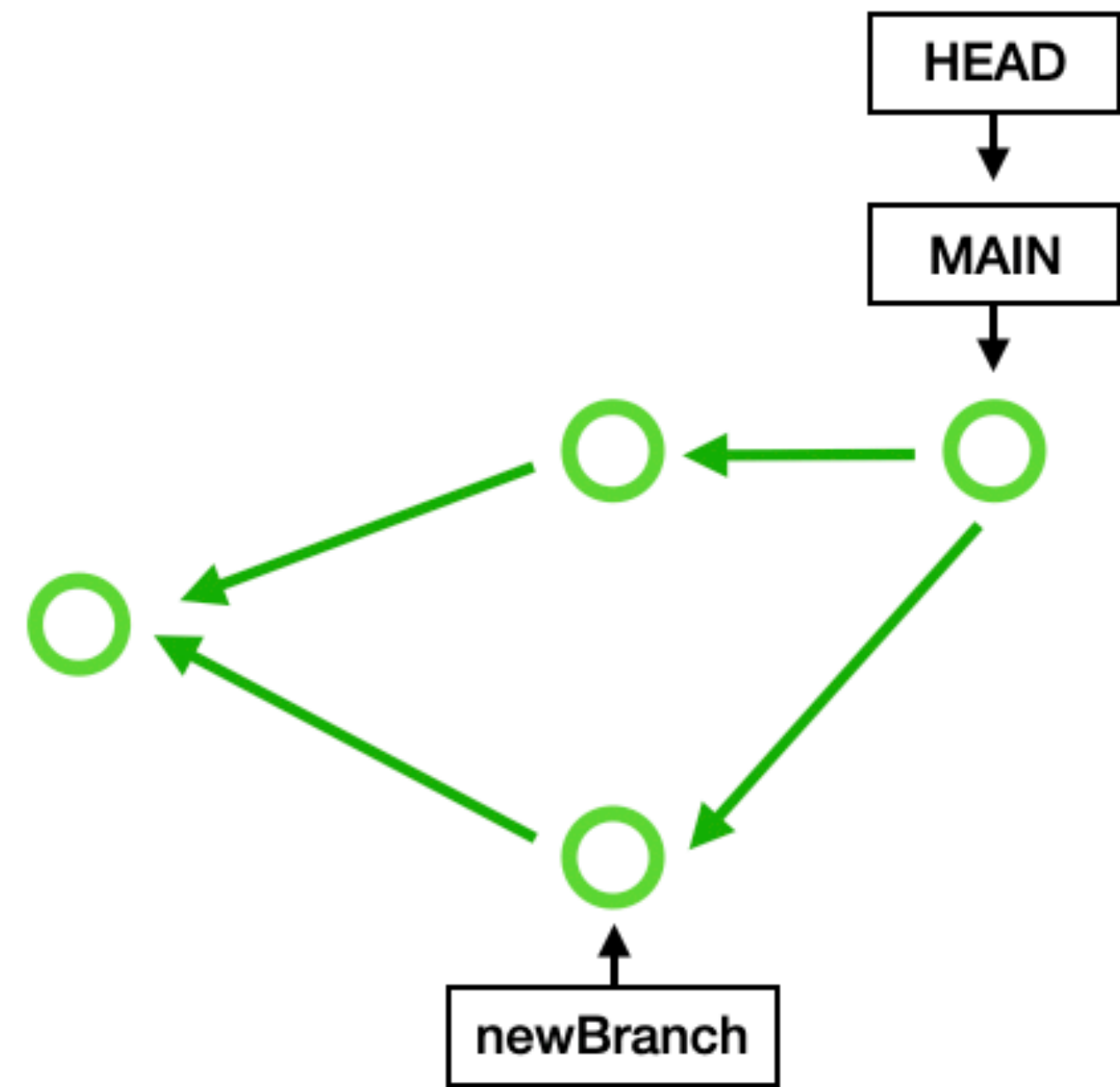
# Merging

## 3 way merge

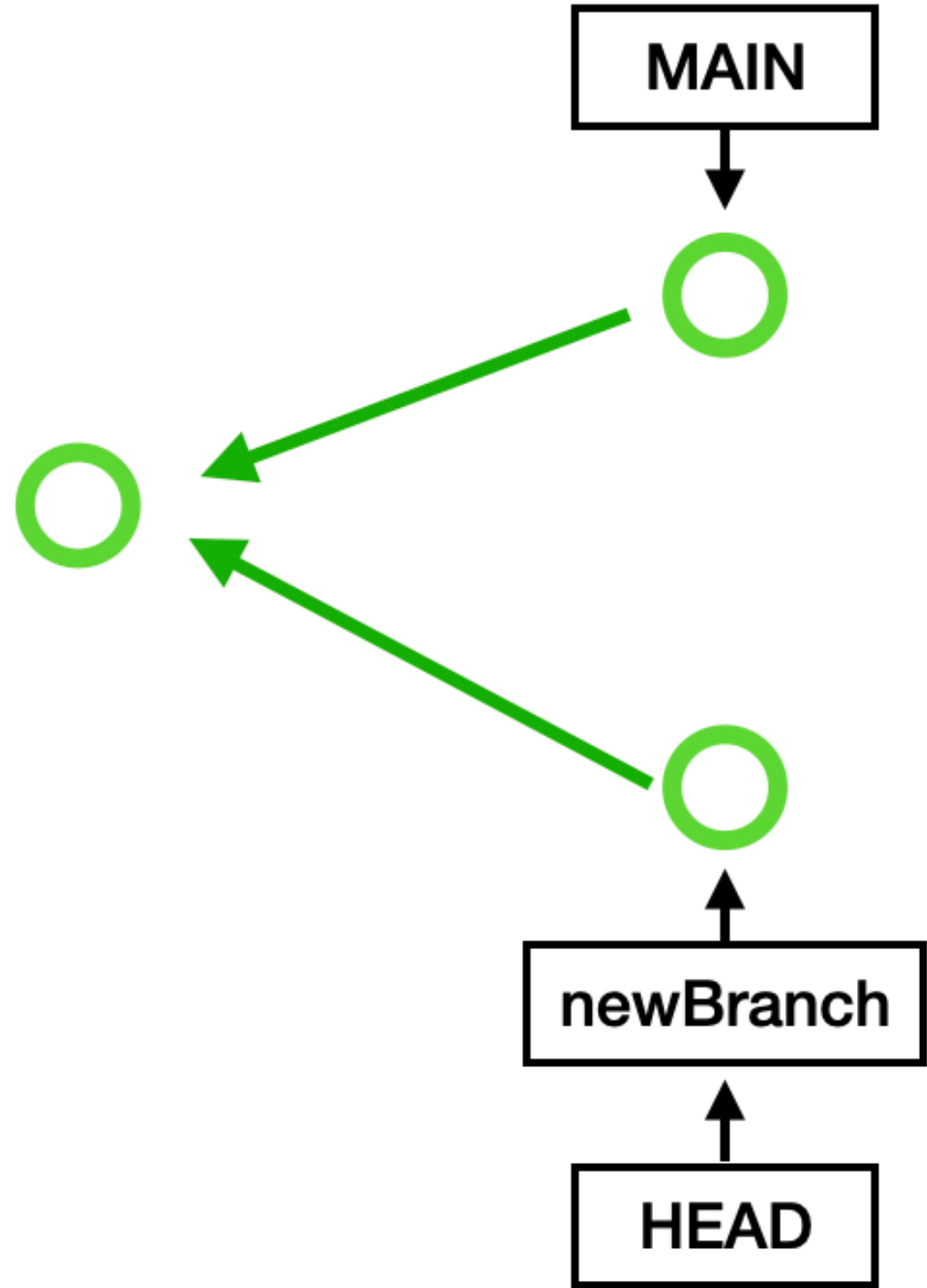
Three way merge



Git merge newBranch

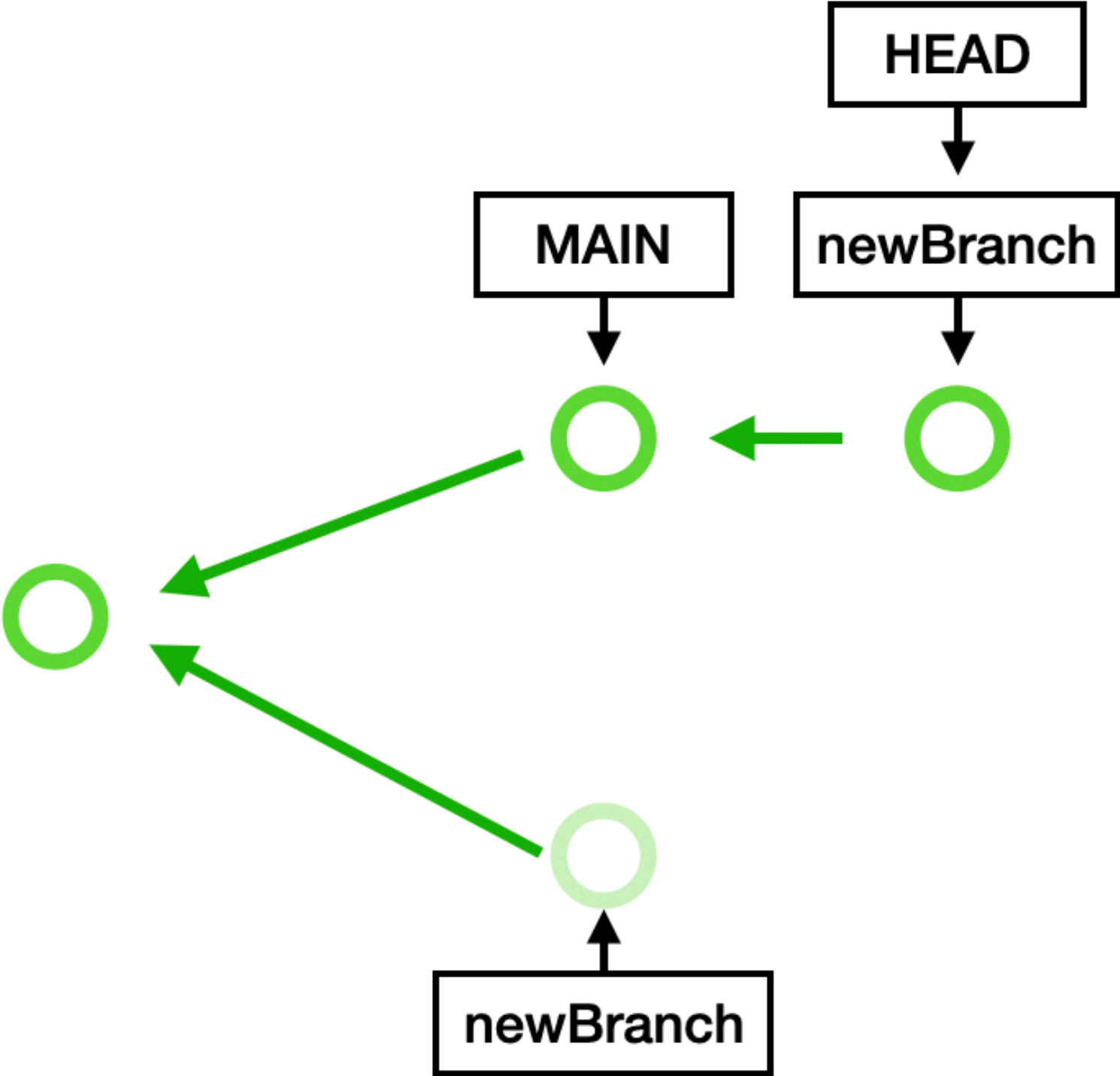


# Git key concepts



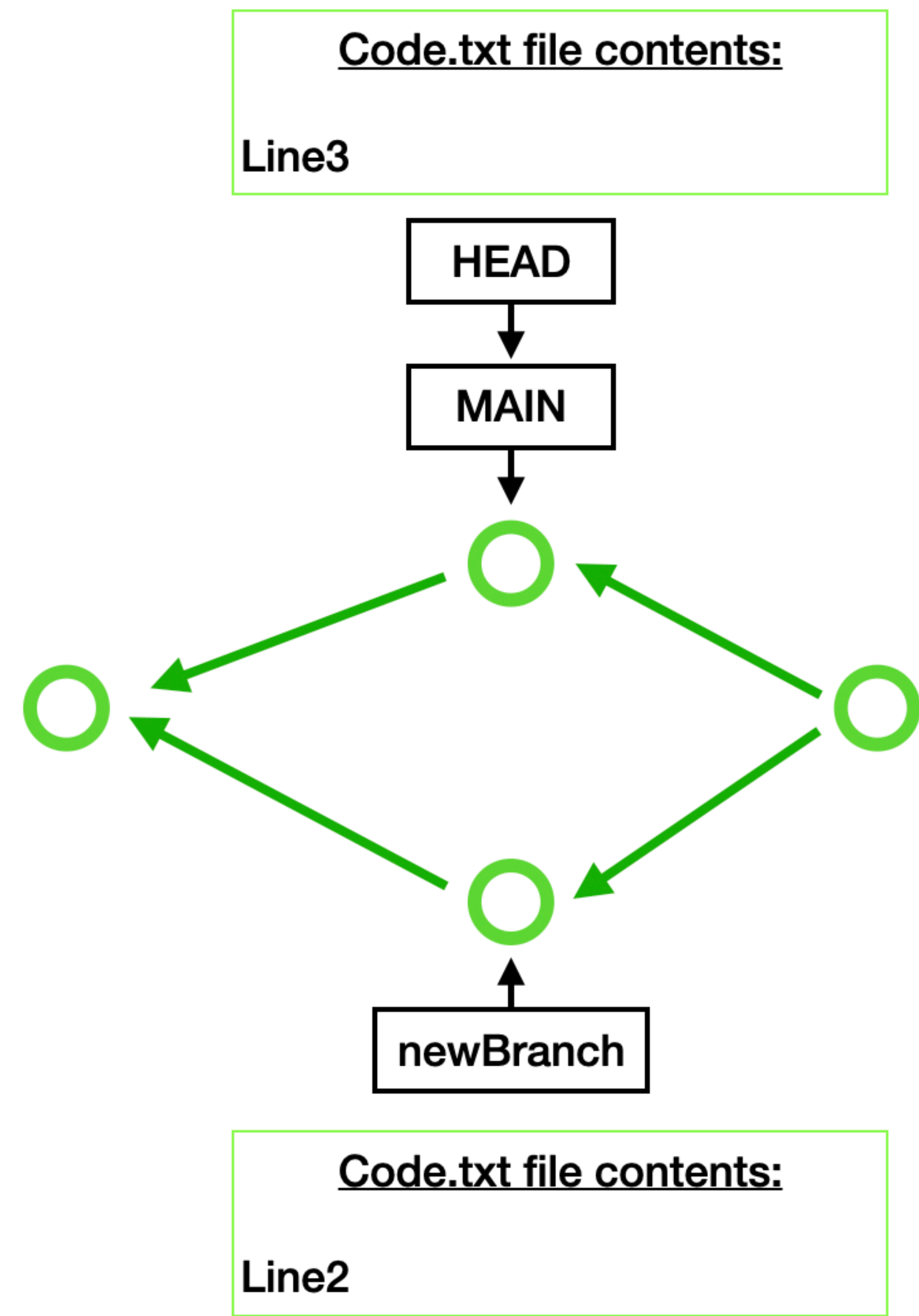
# Merging

Git rebase main



Git rebase

# Git key concepts



# Merge conflicts

```
code.txt x
code.txt
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
1 <<<<<<< HEAD (Current Change)
2 line-3
3 =====
4 line-2
5 >>>>>> newBranch (Incoming Change)
6
```

# Git commands

## Staging Area

git add .

git add <filename>

## Commit

git commit -m “message”

## Branches

git branch

git checkout -b <new branch name>

## Merging

git merge <branch to merge>

git rebase <branch to rebase to>

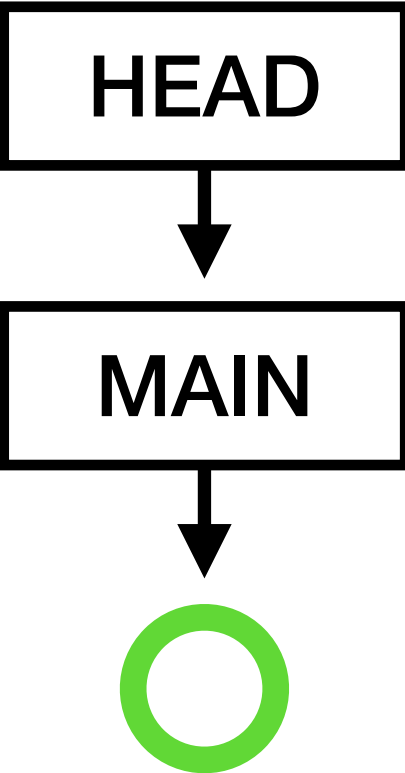
## Utility

git status

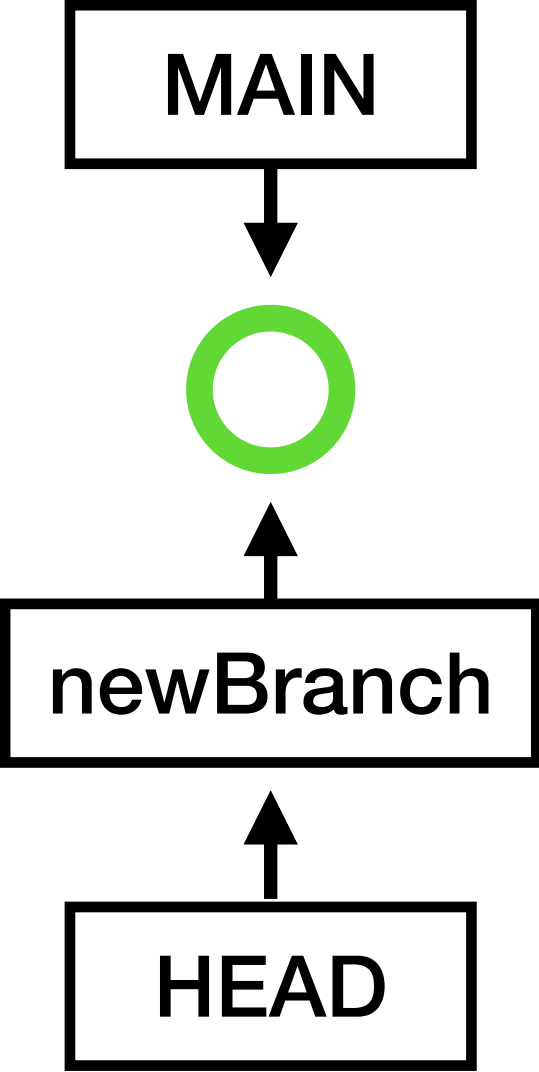
git log --all --graph --oneline

git diff <filename>

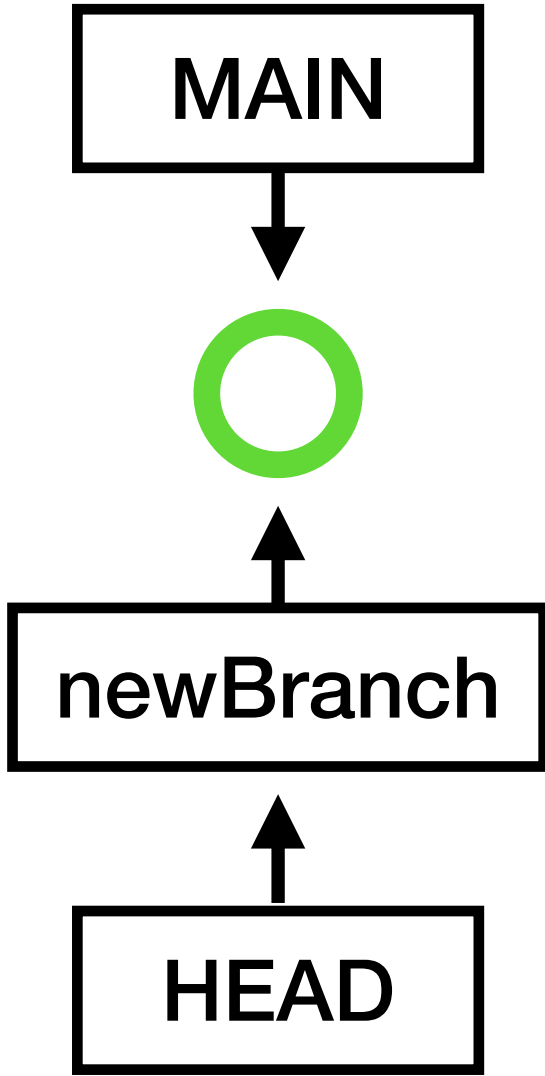
# Visualise git



Git checkout -b "newBranch"



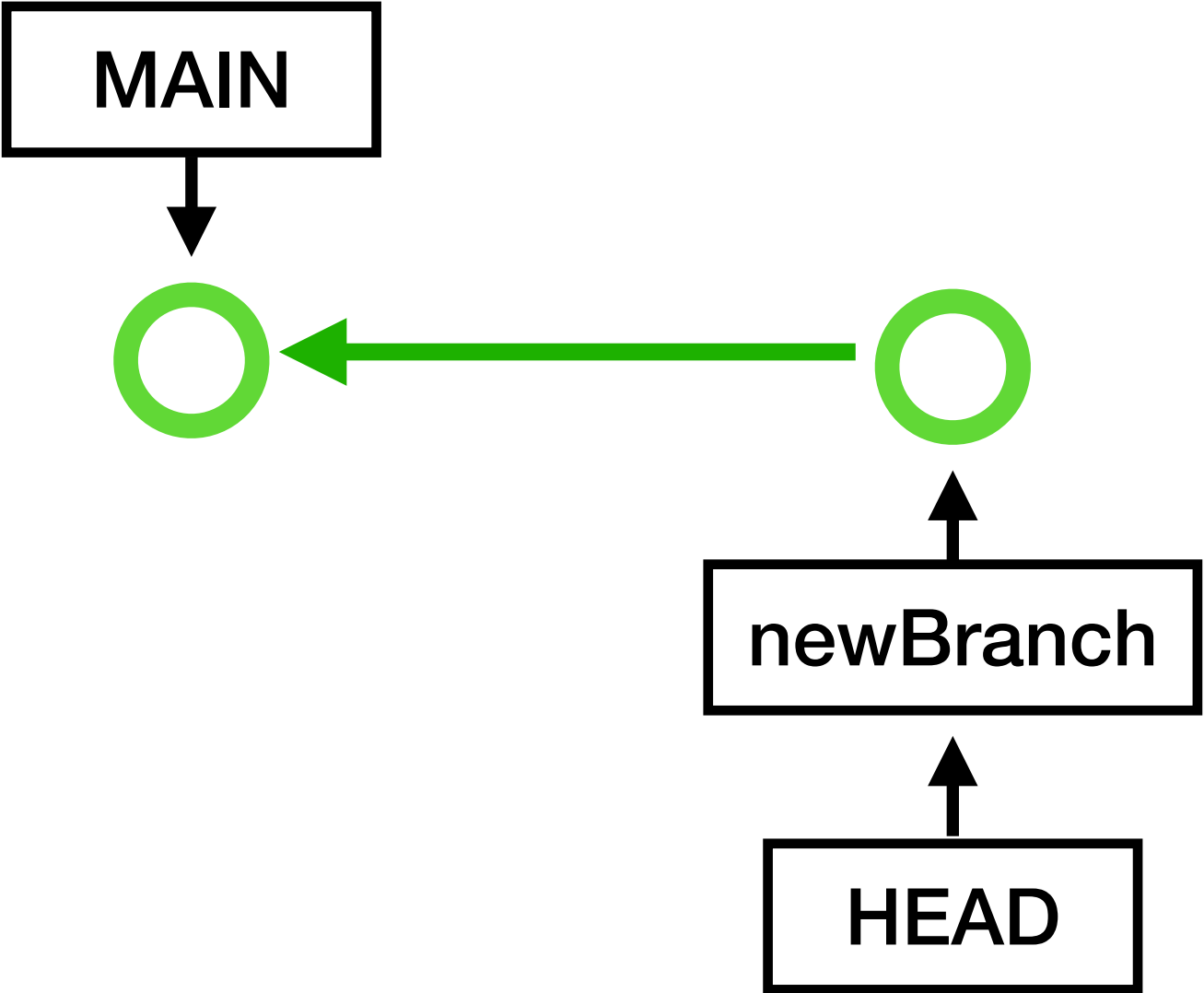
# Visualise git



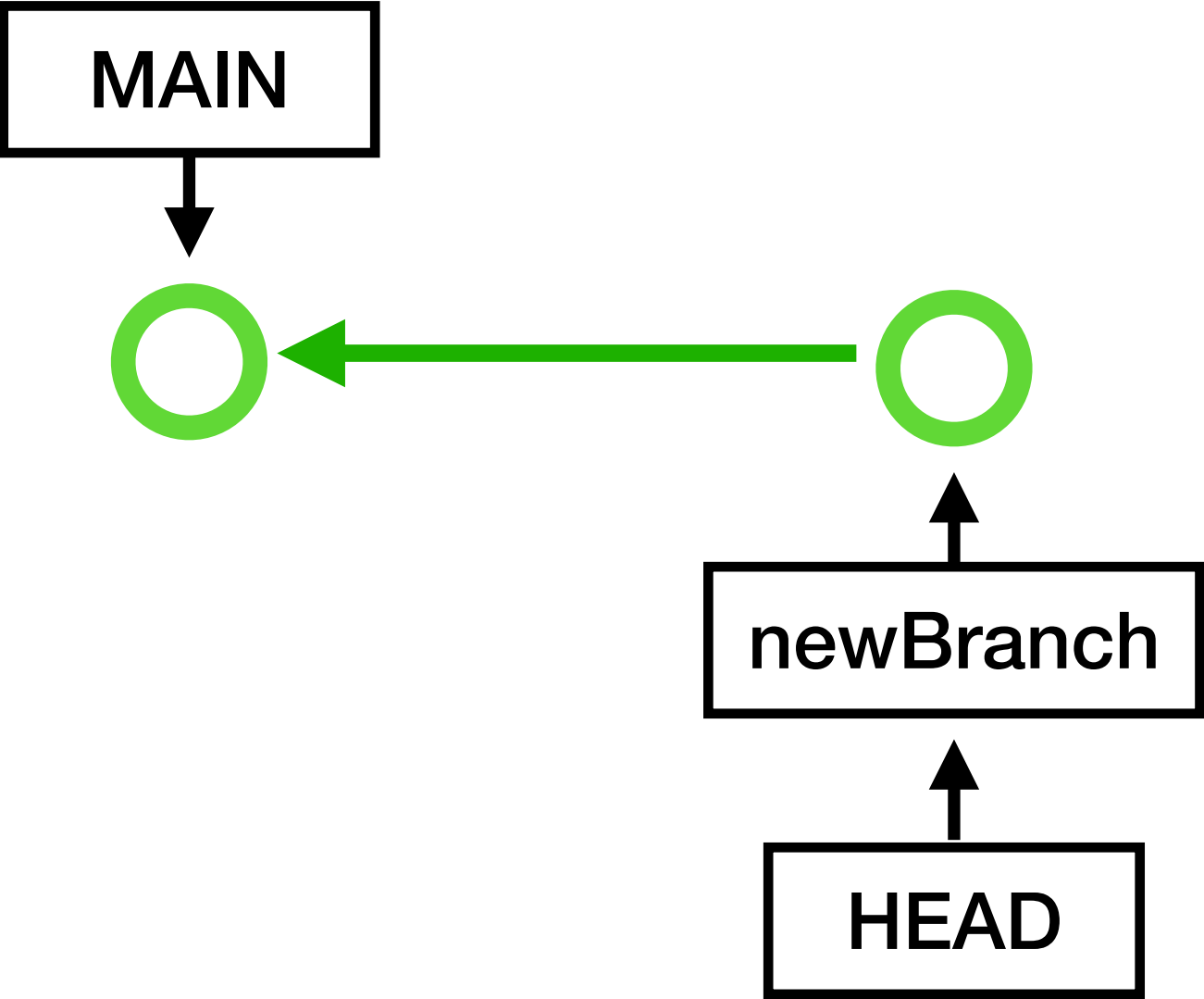
Make some changes

Git add .

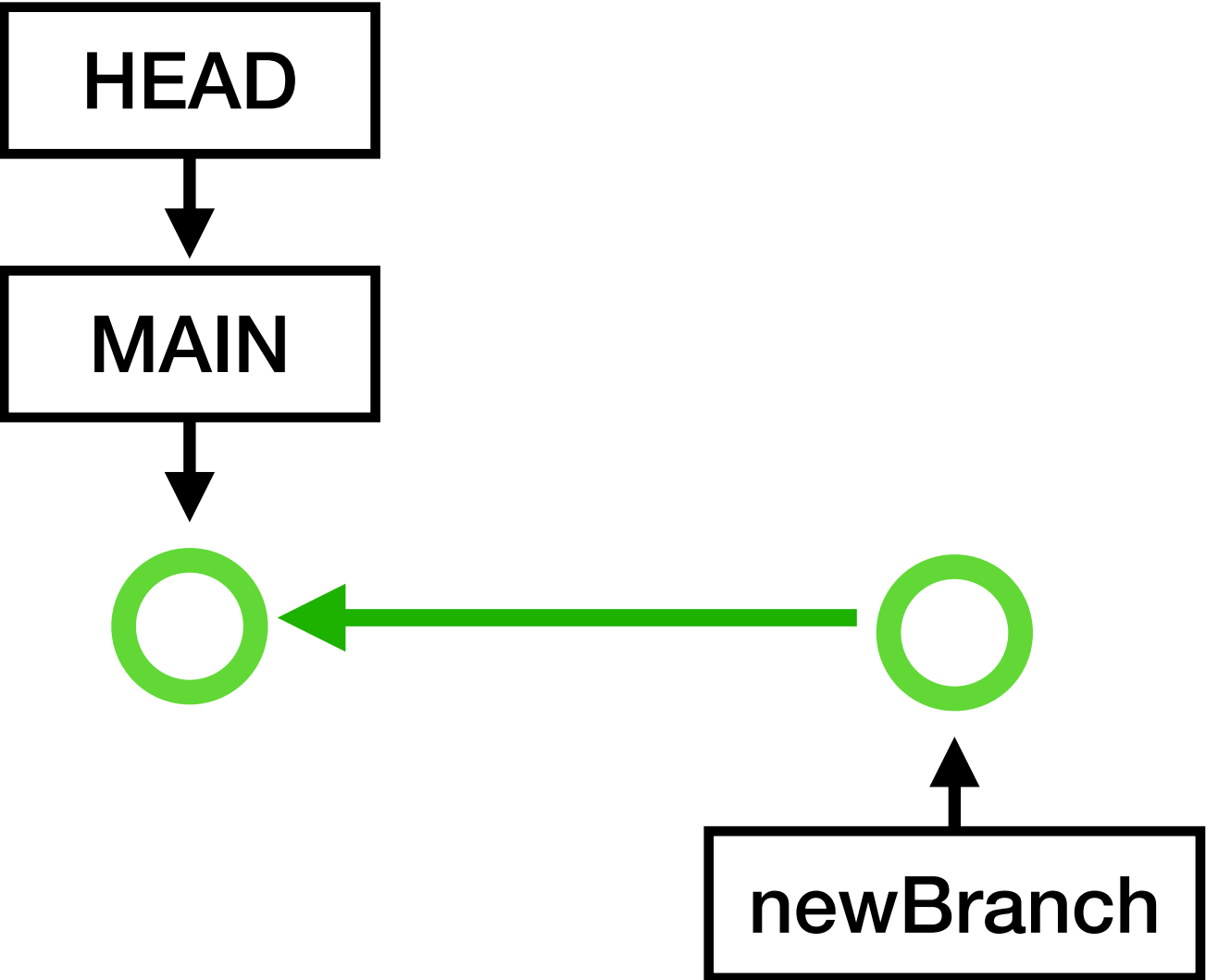
Git commit -m "1st change"



# Visualise git

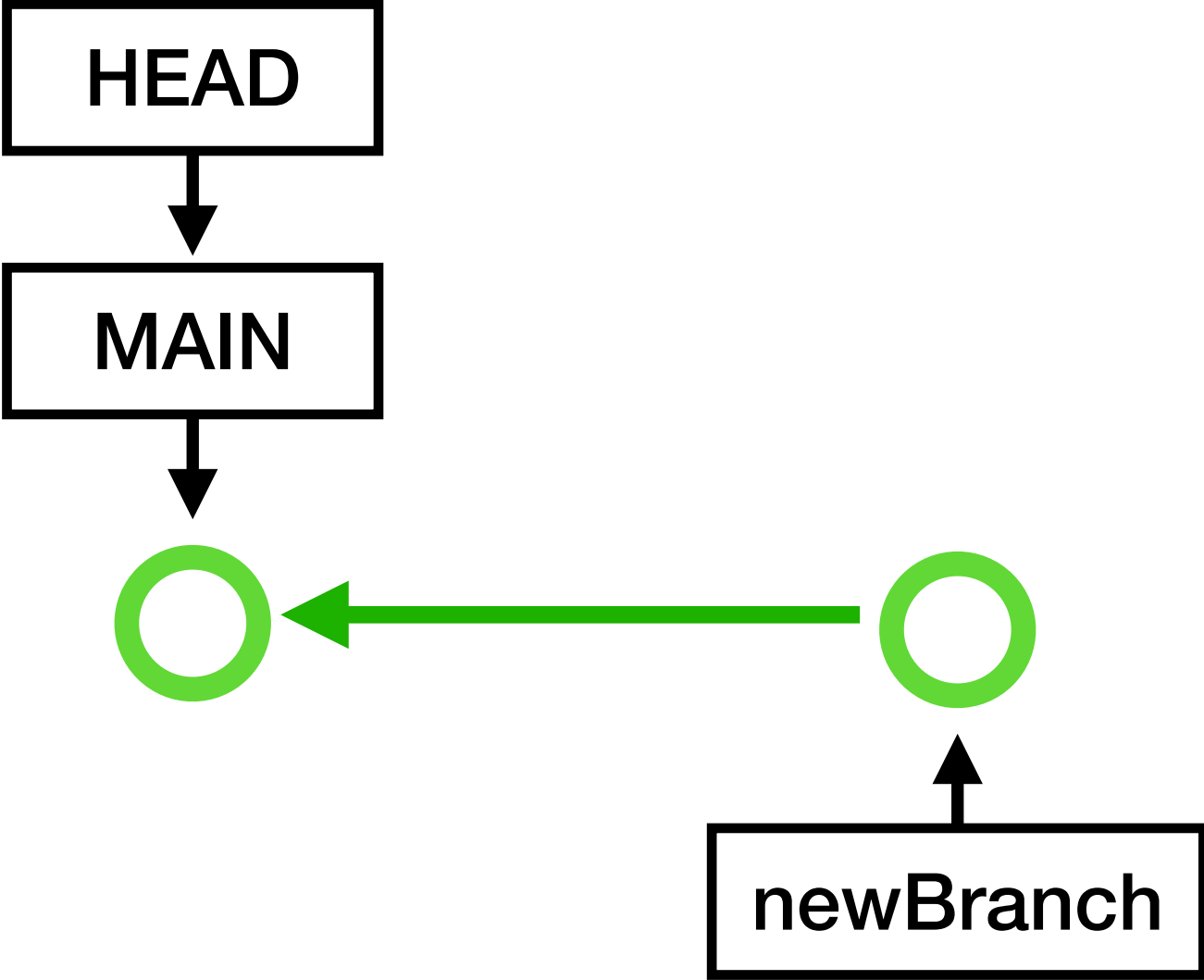


Git checkout main





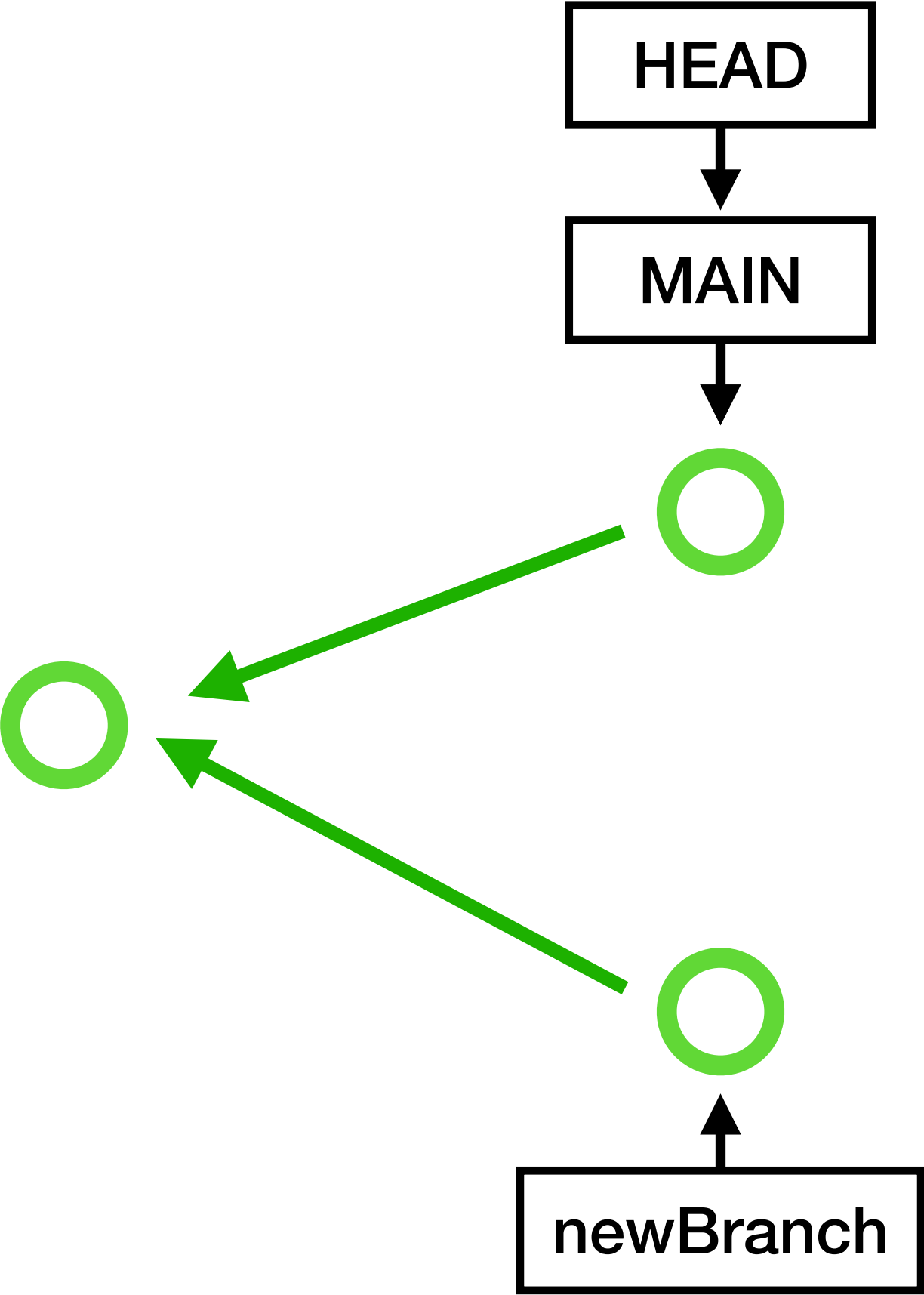
# Visualise git



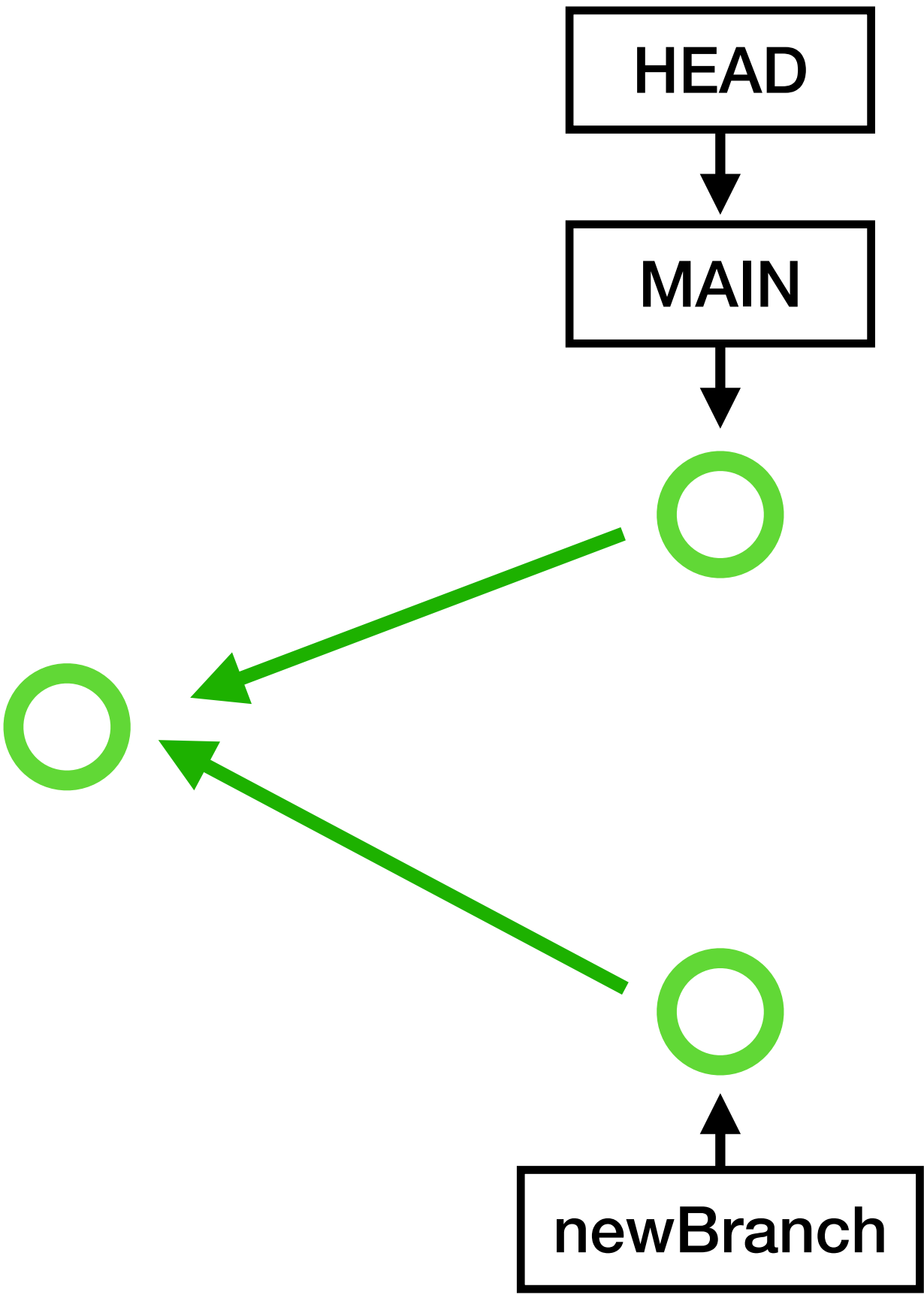
Make some edits

Git add .

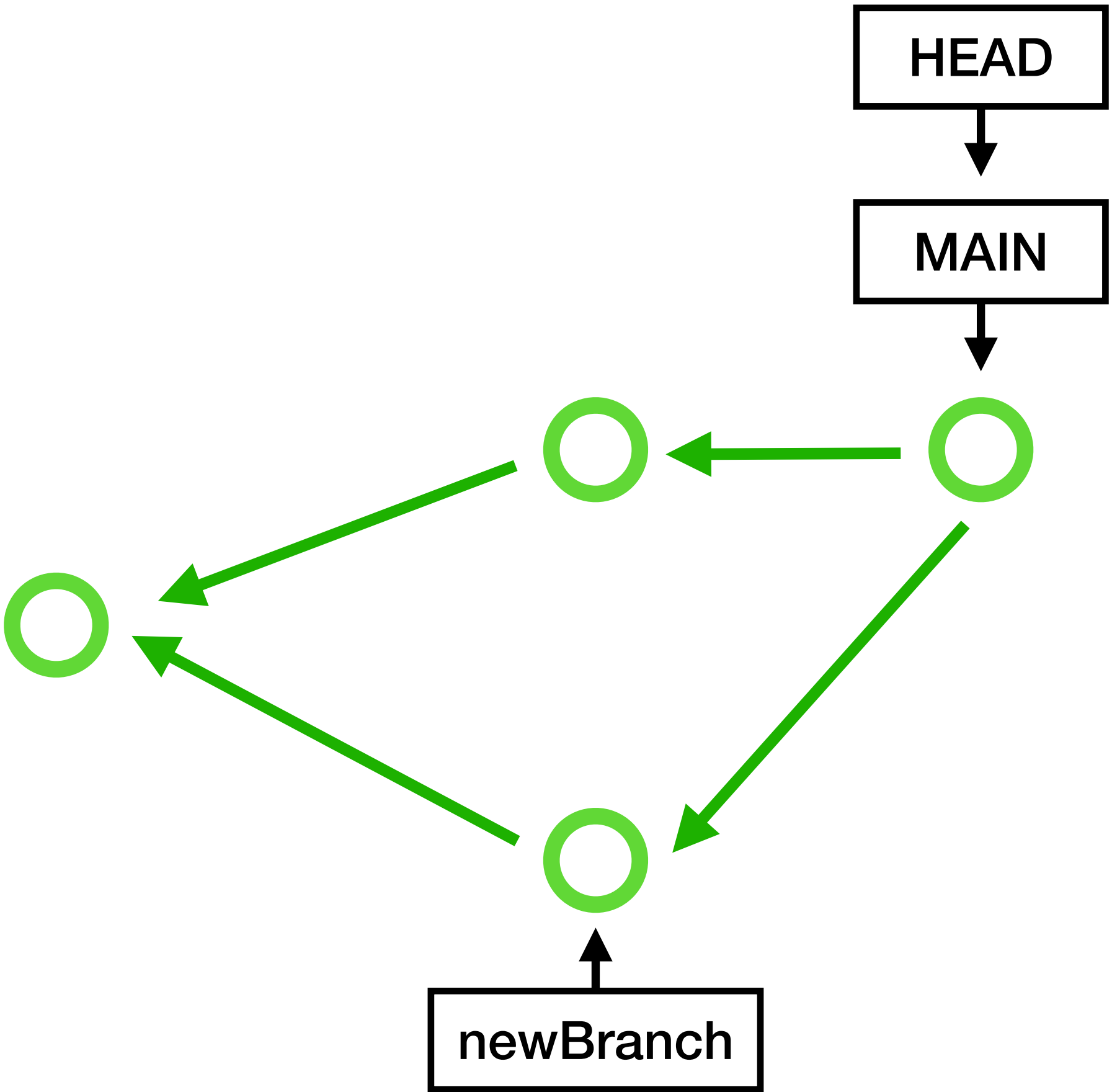
Git commit -m "second commit"



Visualise git



Git merge newBranch



Three way merge

# Working with GitHub

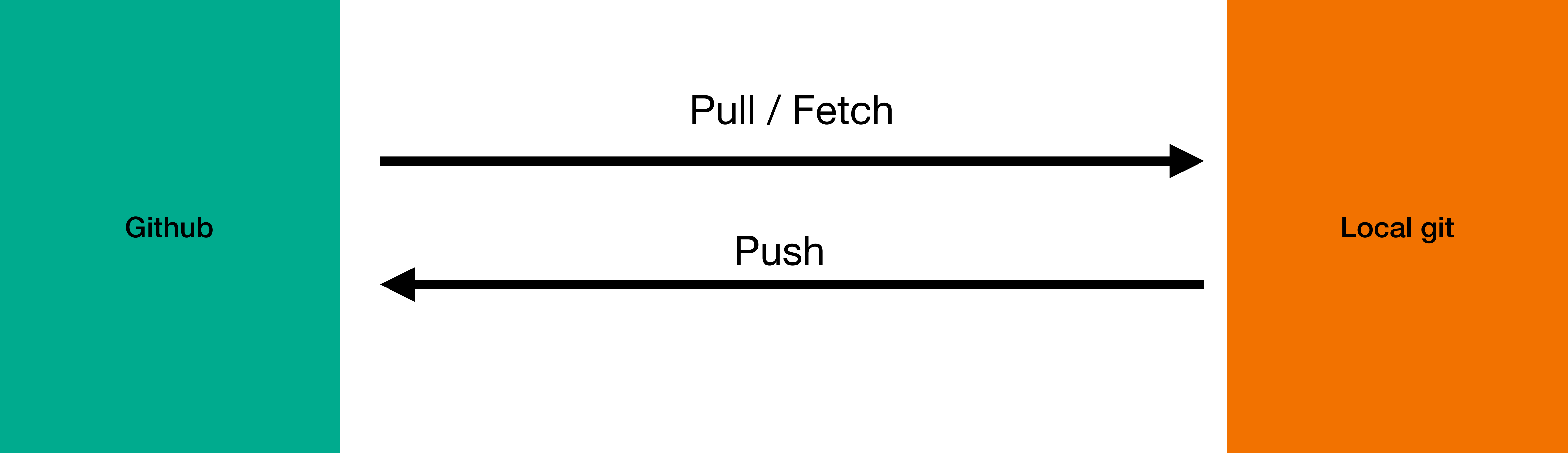
`git remote add <name of remote> <url>`



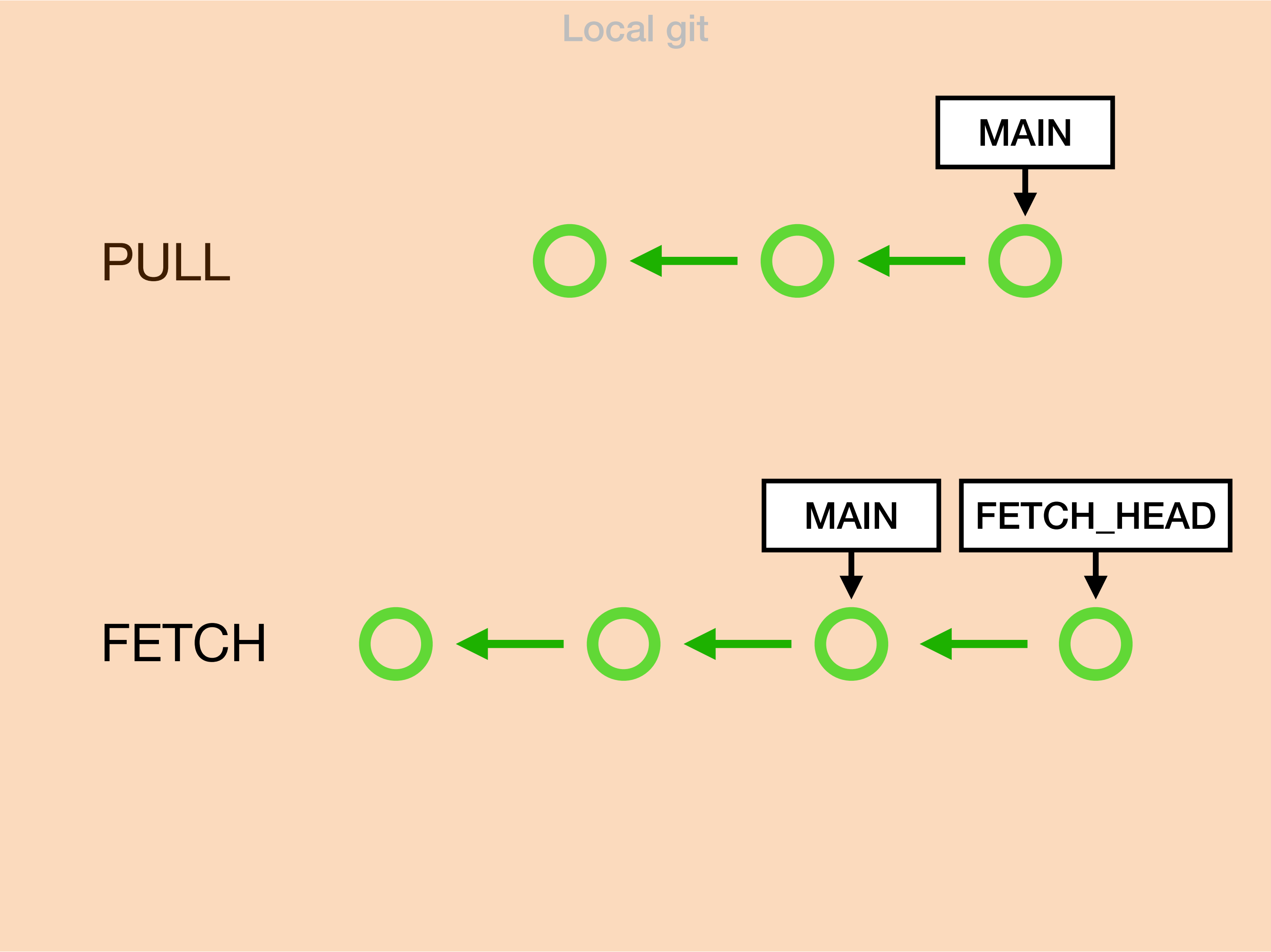
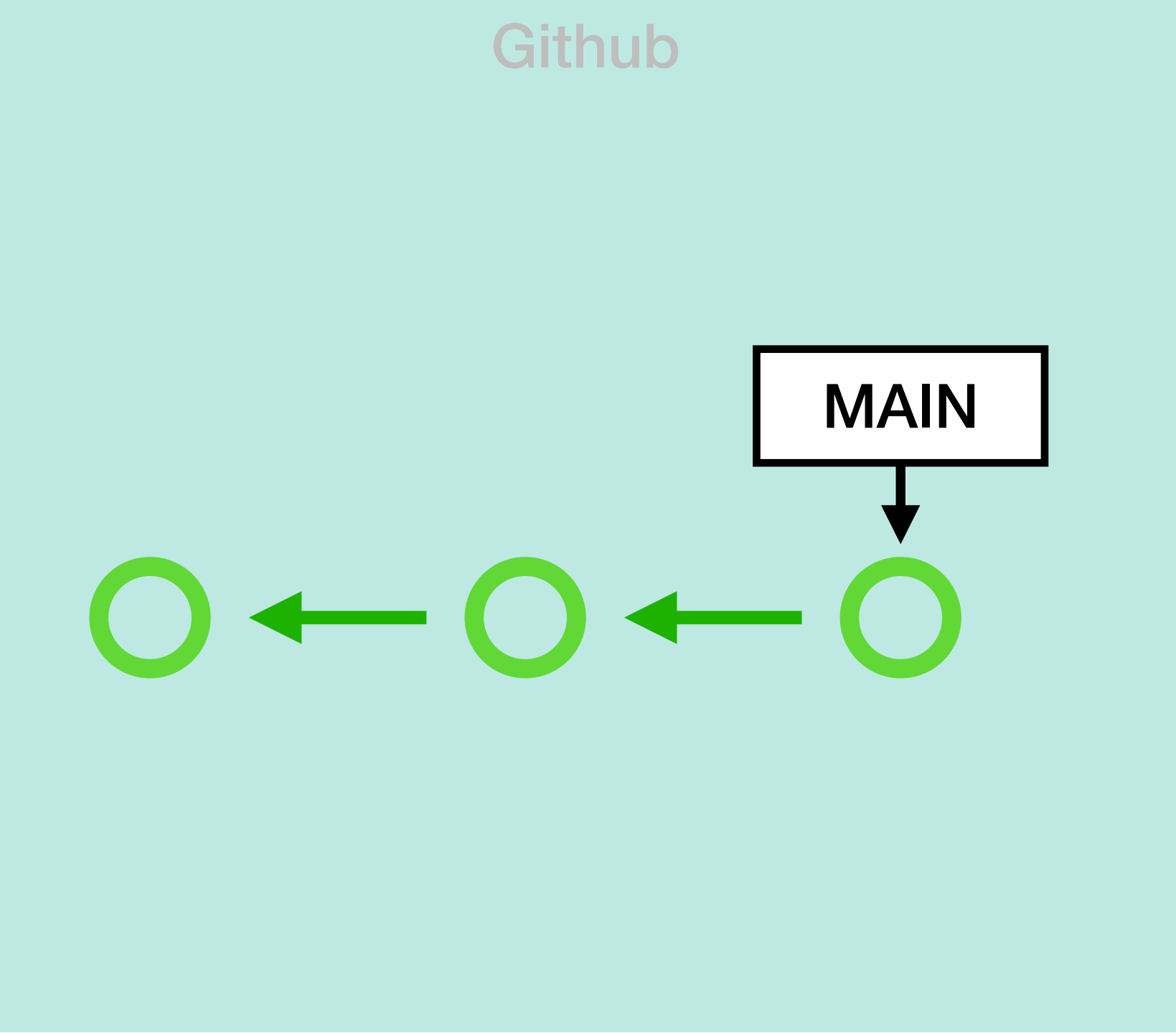
Github

Local git

# Working with GitHub



# Working with GitHub



# Working with GitHub

## .gitignore

Tells git to ignore specific files when pushing to GitHub.

Secret files such as API keys, config files

# Git commands

## Staging Area

git add .

git add <filename>

## Commit

git commit -m “message”

## Branches

git branch

git checkout -b <new branch name>

## Merging

git merge <branch to merge>

git rebase <branch to rebase to>

## Utility

git status

git log --all --graph --oneline

git diff <filename>

## .gitignore

\*\*/foldername

— every folder that has the specified folder name

\*.png

— every file with .png extension

Tells other developers on GitHub what they can do with your work



## **I need to work in a community.**

Use the **license preferred by the community** you're contributing to or depending on. Your project will fit right in.

If you have a dependency that doesn't have a license, ask its maintainers to **add a license**.



## **I want it simple and permissive.**

The **MIT License** is short and to the point. It lets people do almost anything they want with your project, like making and distributing closed source versions.

**Babel**, **.NET Core**, and **Rails** use the MIT License.



## **I care about sharing improvements.**

The **GNU GPLv3** also lets people do almost anything they want with your project, *except* distributing closed source versions.

**Ansible**, **Bash**, and **GIMP** use the GNU GPLv3.

Image from: <https://choosealicense.com>