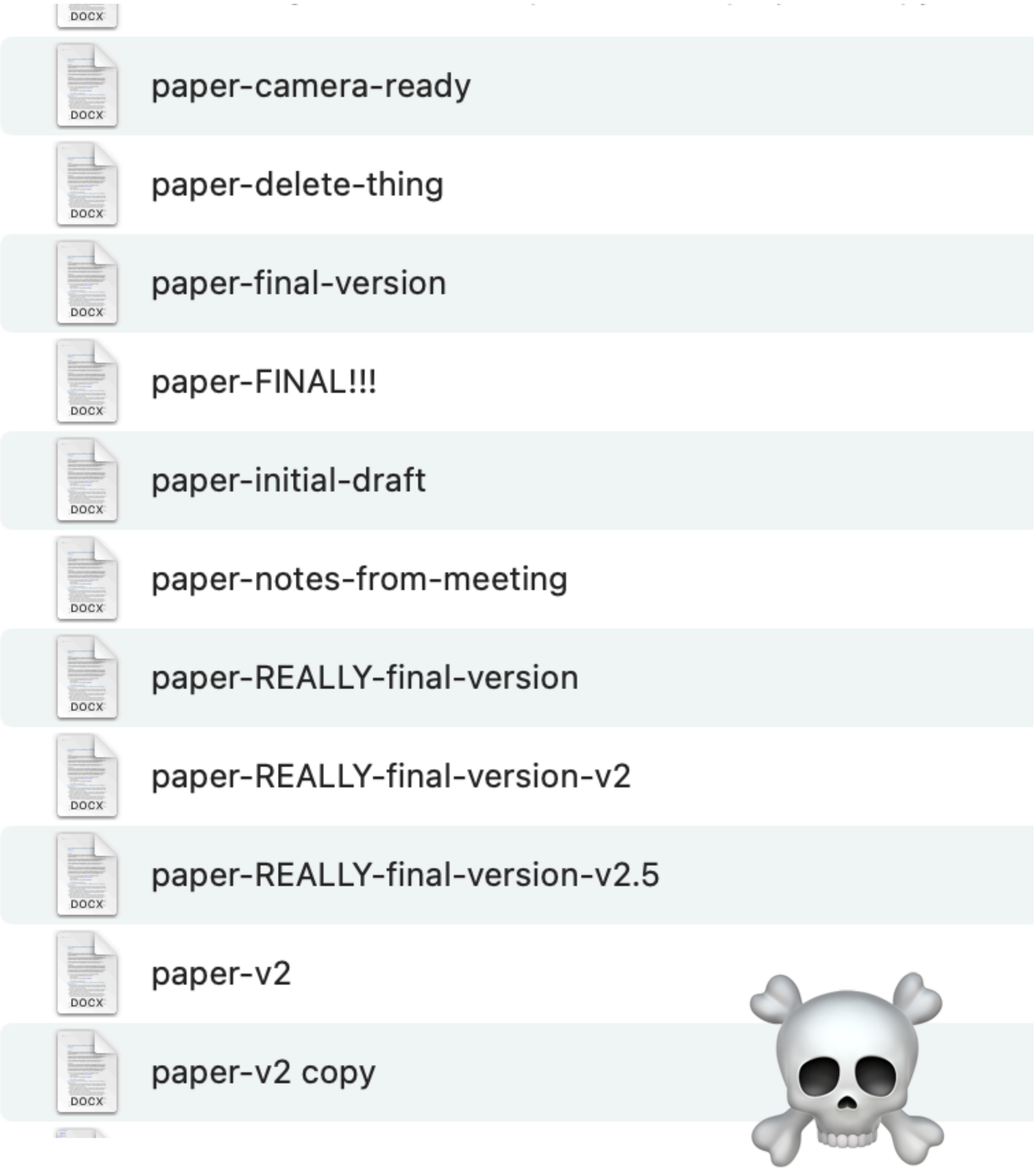
Workshop

Using Git and GitHub effectively

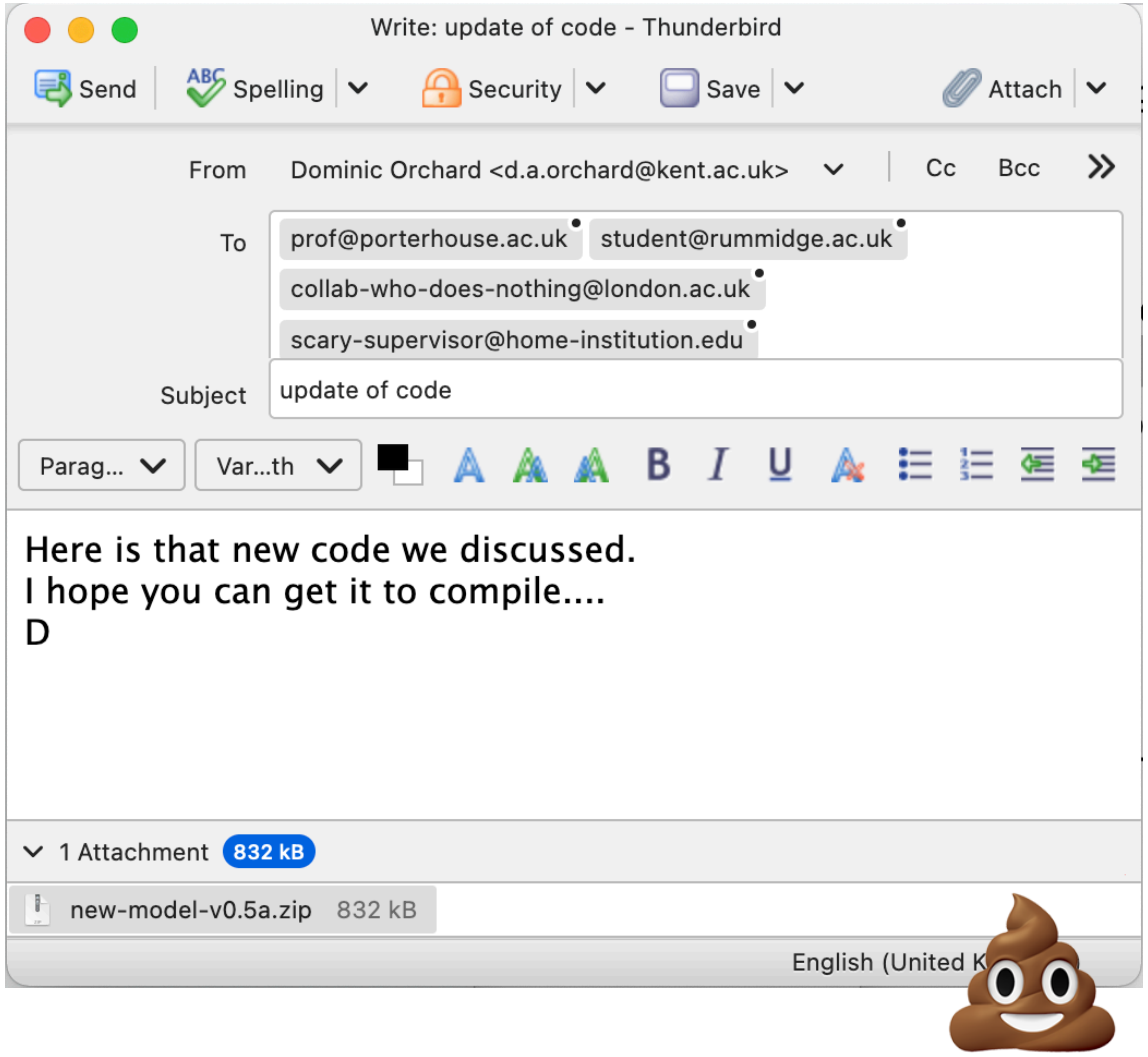
Dominic Orchard

UNIVERSITY OF CAMBRIDGE

Institute of Computing
for Climate Science



paper-camera-ready
paper-delete-thing
paper-final-version
paper-FINAL!!!
paper-initial-draft
paper-notes-from-meeting
paper-REALLY-final-version
paper-REALLY-final-version-v2
paper-REALLY-final-version-v2.5
paper-v2
paper-v2 copy



Write: update of code - Thunderbird

From: Dominic Orchard <d.a.orchard@kent.ac.uk>

To: prof@porterhouse.ac.uk student@rummidge.ac.uk collab-who-does-nothing@london.ac.uk scary-supervisor@home-institution.edu

Subject: update of code

Here is that new code we discussed.
I hope you can get it to compile....
D

1 Attachment 832 kB

new-model-v0.5a.zip 832 kB

git

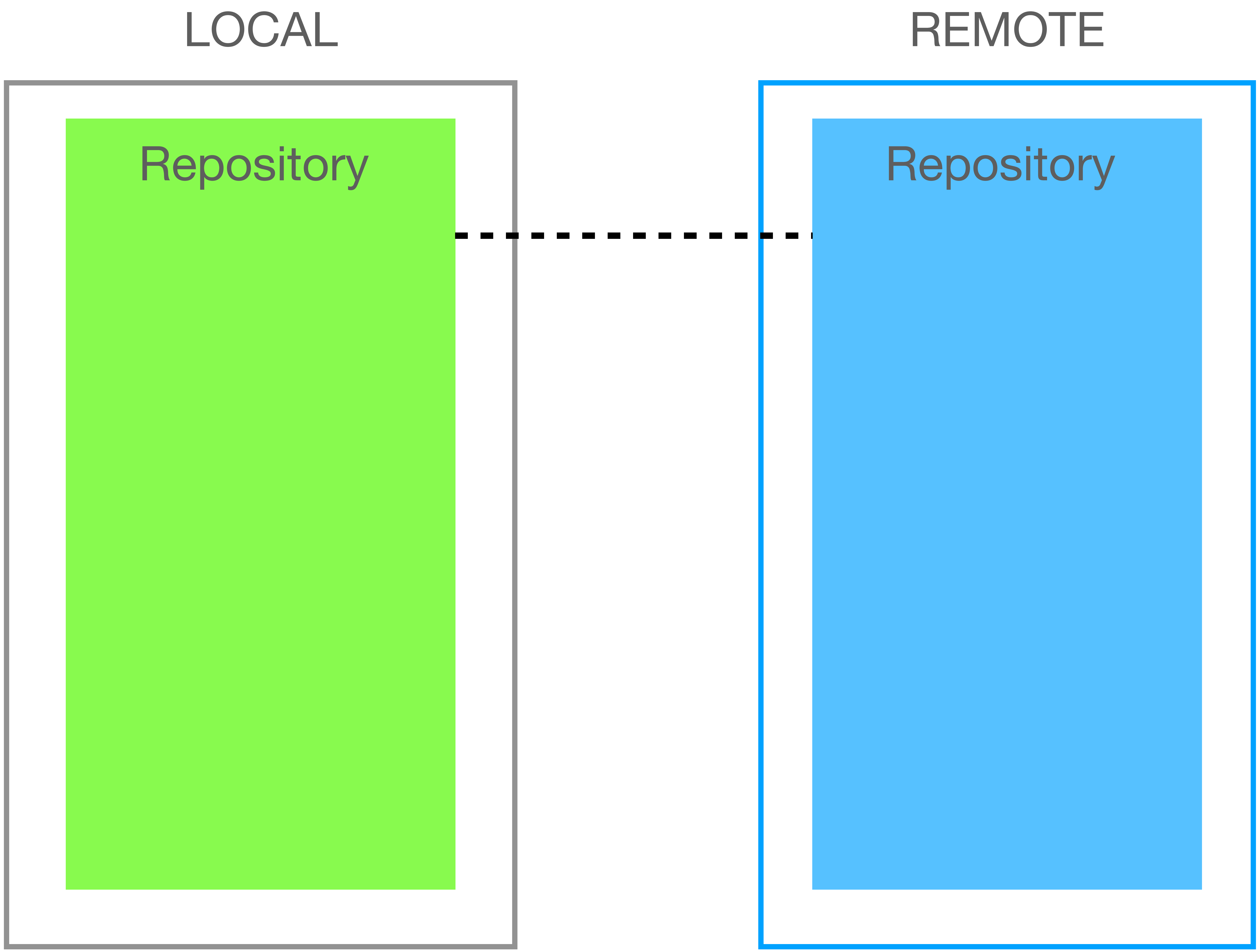
GitHub

"It is easy to shoot your foot off with `git`, but also easy to revert to a previous foot and merge it with your current leg."

Jack William Bell

Plan and learning outcomes

- Basis of git paradigm and improving accountability
- Lesser known corners: *stash*, *rebase*
- GitHub's view and its organisational aspects
- **+ Do some practice**

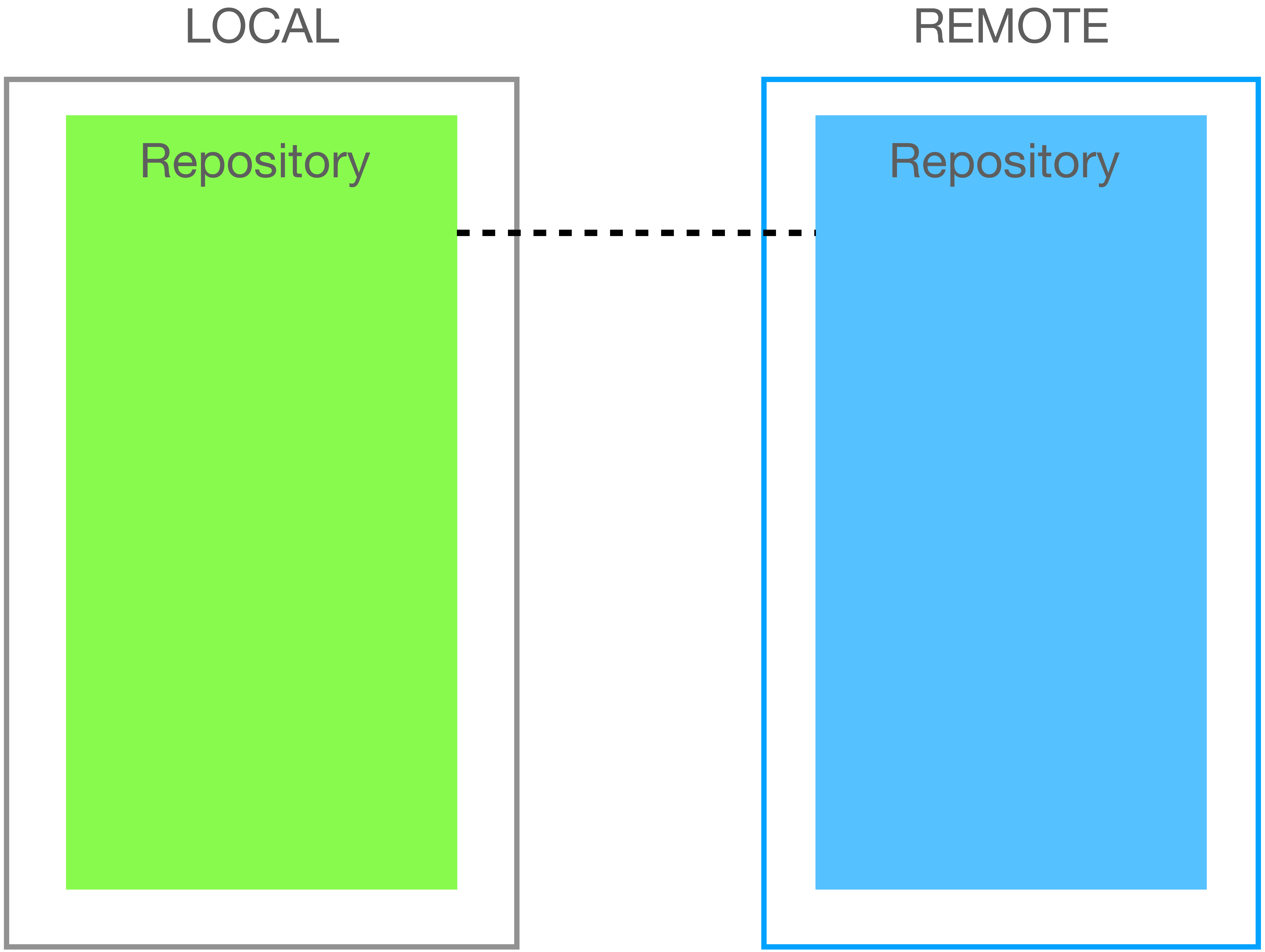


LOCAL

Repository

REMOTE

Repository

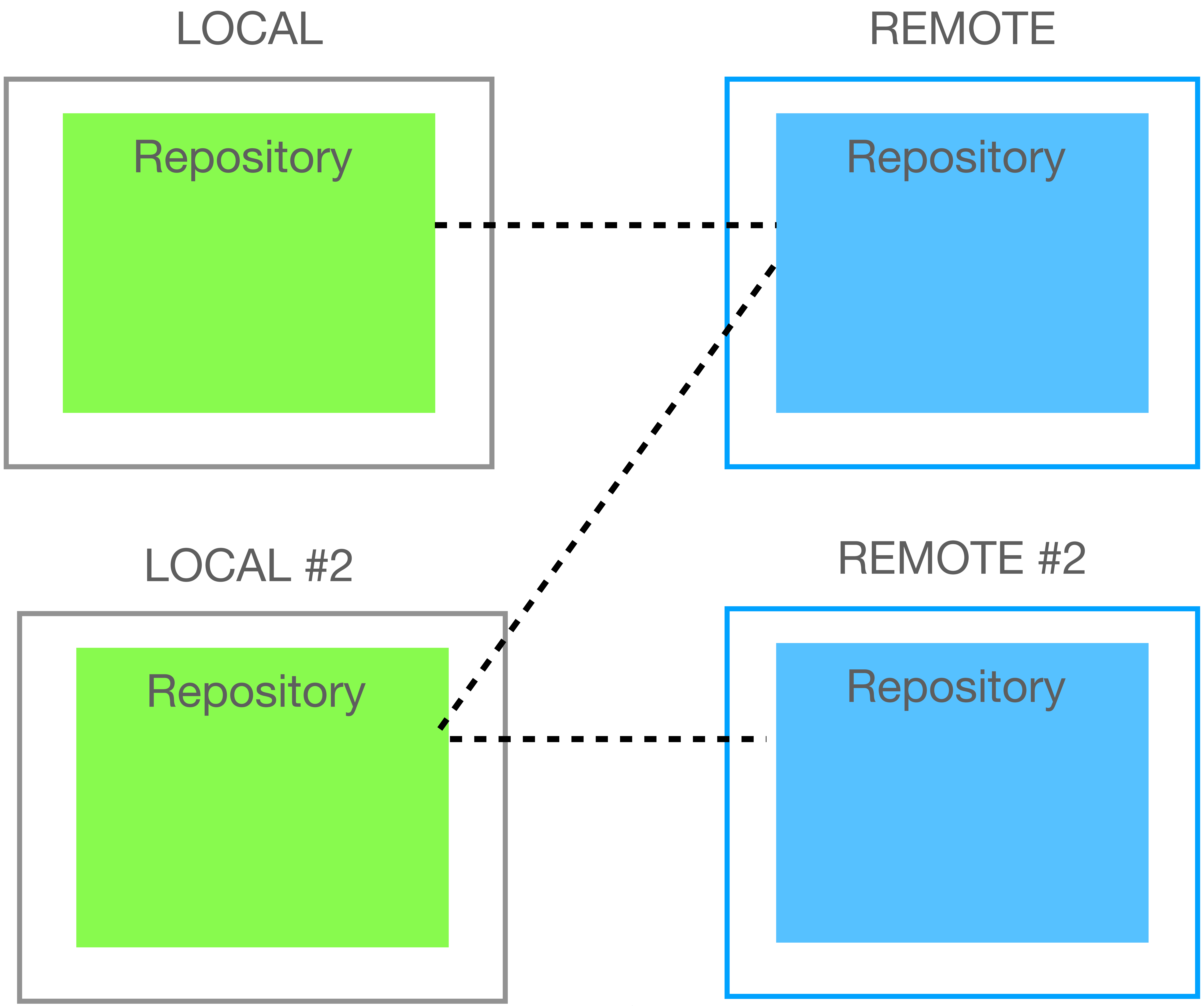


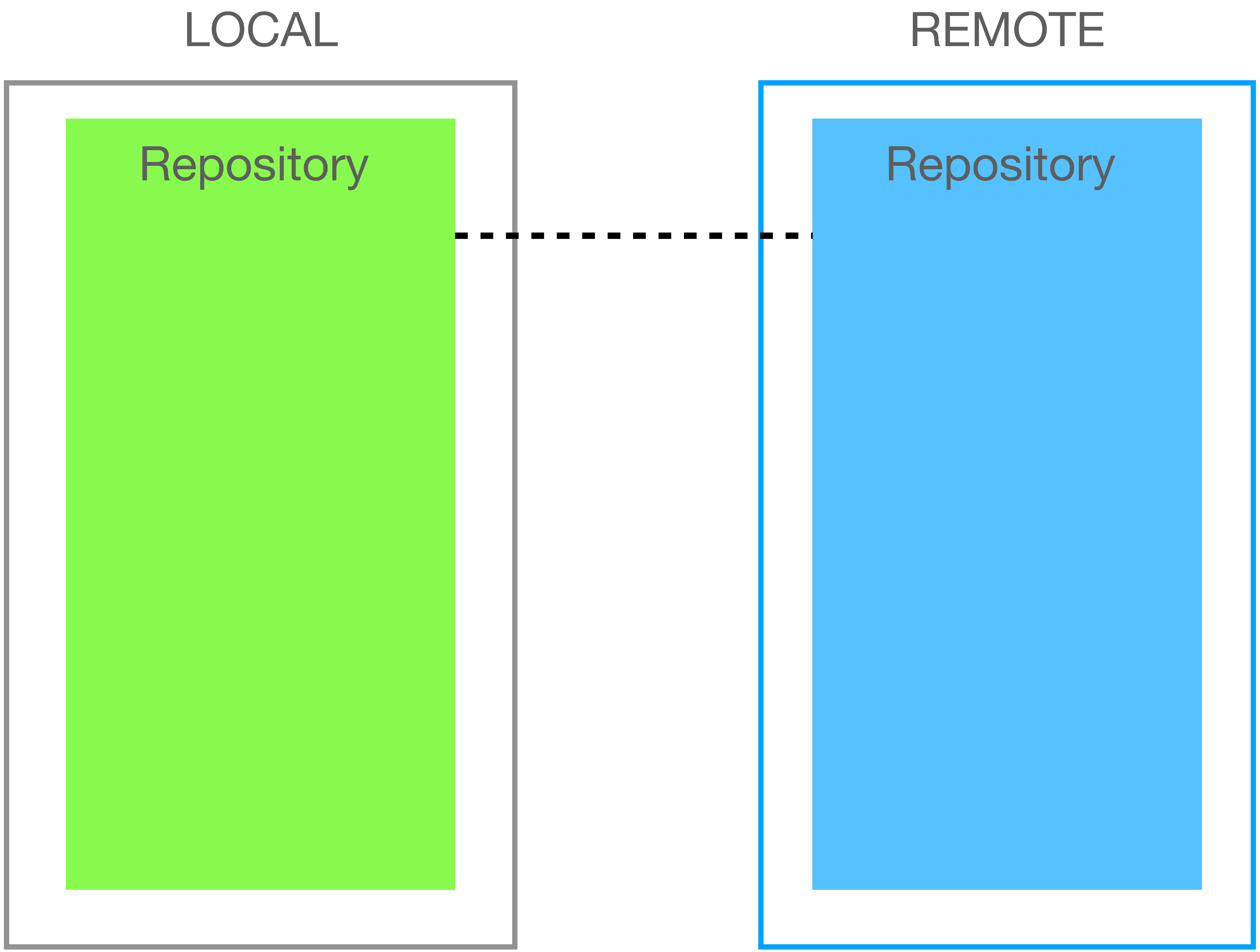
LOCAL

Repository

REMOTE

Repository


LOCAL
Repository
REMOTE
Repository
LOCAL #2
Repository
REMOTE #2
Repository

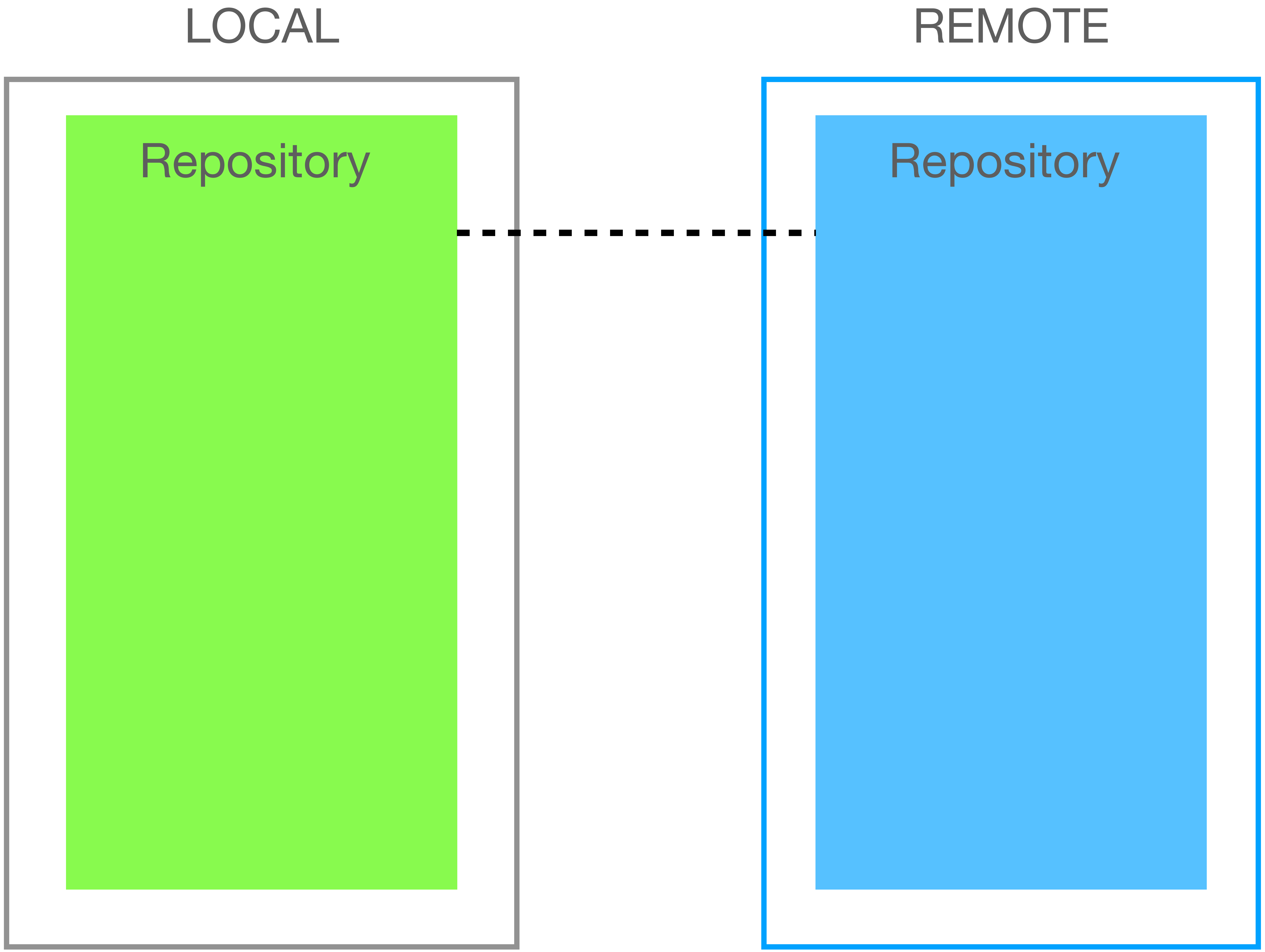


LOCAL

Repository

REMOTE

Repository

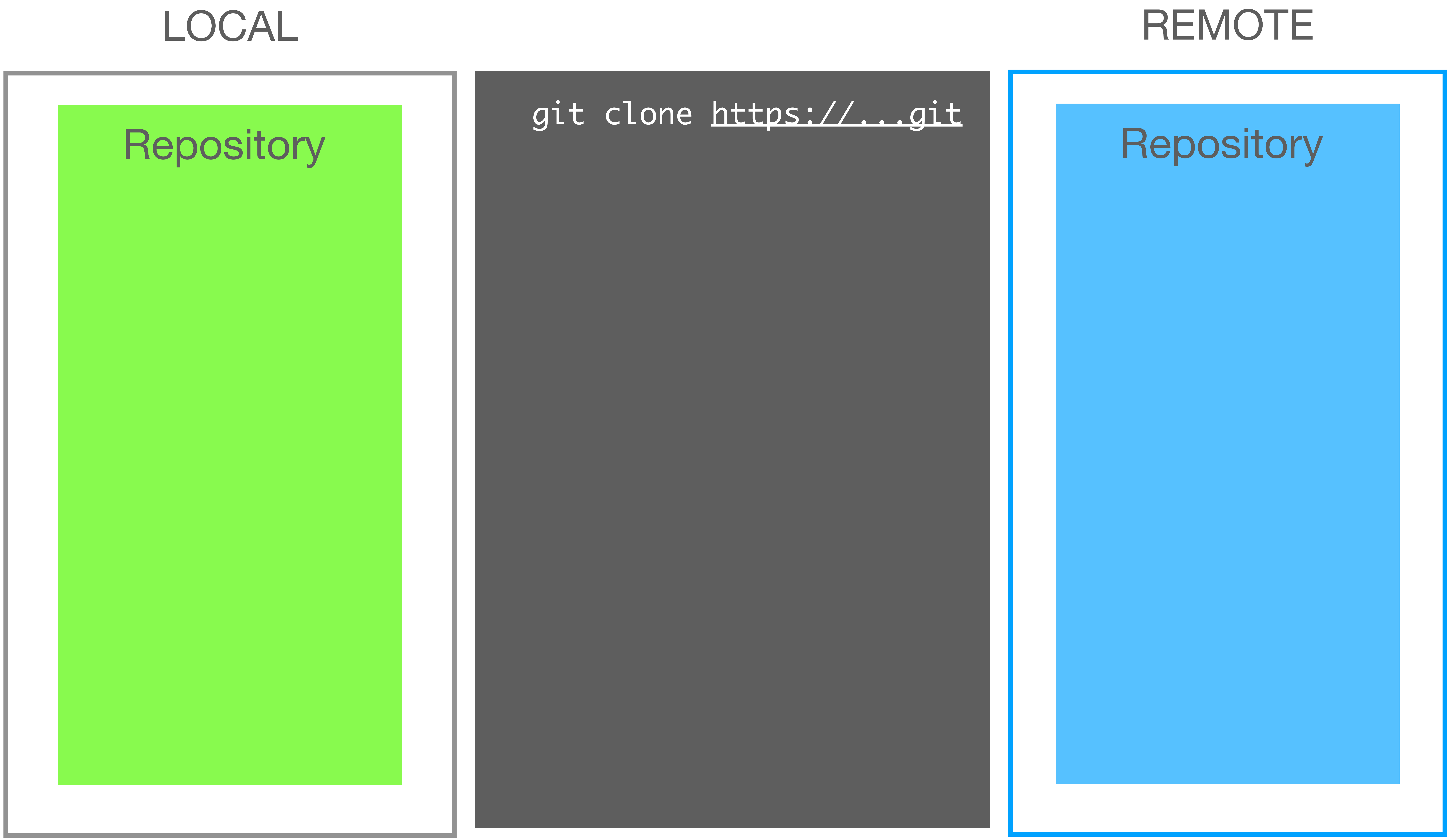


LOCAL

Repository

REMOTE

Repository



LOCAL

Repository

```
git clone https://...git
```

REMOTE

Repository

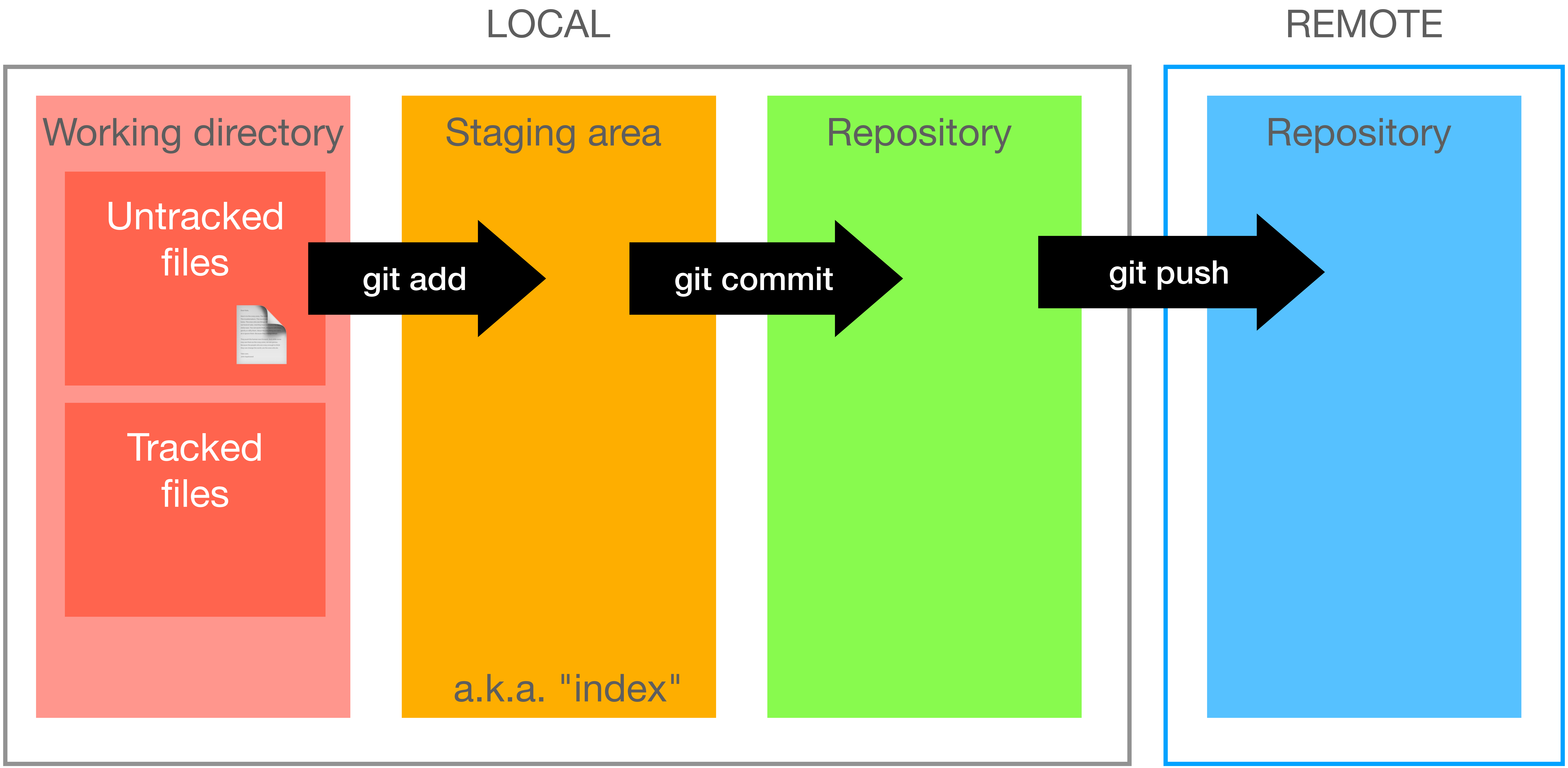
GitHub

LOCAL
Repository
REMOTE
Repository
GitHub

```
mkdir tuesday
cd tuesday
git init
git add remote origin https://...git
```

LOCAL
Repository
REMOTE
Repository
GitHub

```
mkdir tuesday
cd tuesday
git init
git remote add origin https://...git
```



LOCAL

REMOTE

Working directory

Untracked files

Tracked files

git add

Staging area

a.k.a. "index"

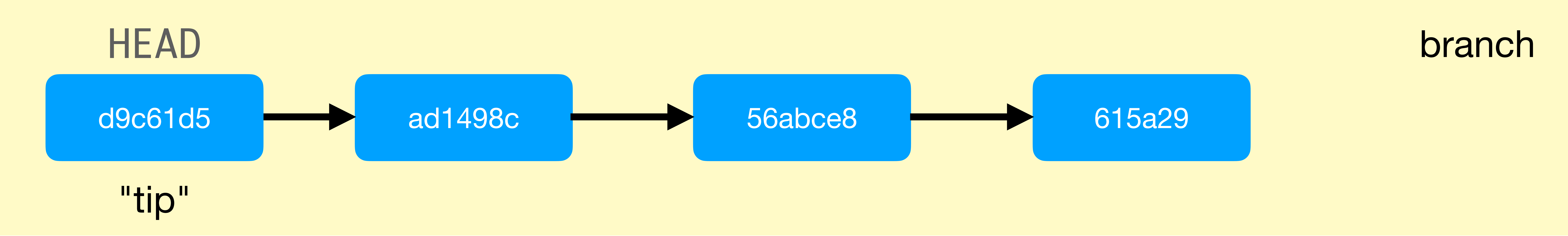
git commit

Repository

git push

Repository

Version history: chain of commits



HEAD

branch

d9c61d5 → ad1498c → 56abce8 → 615a29

"tip"

7-digit prefix of "hash"

d9c61d5b916b34ad06085b336688a230c3485a41

Aside on git commit messages

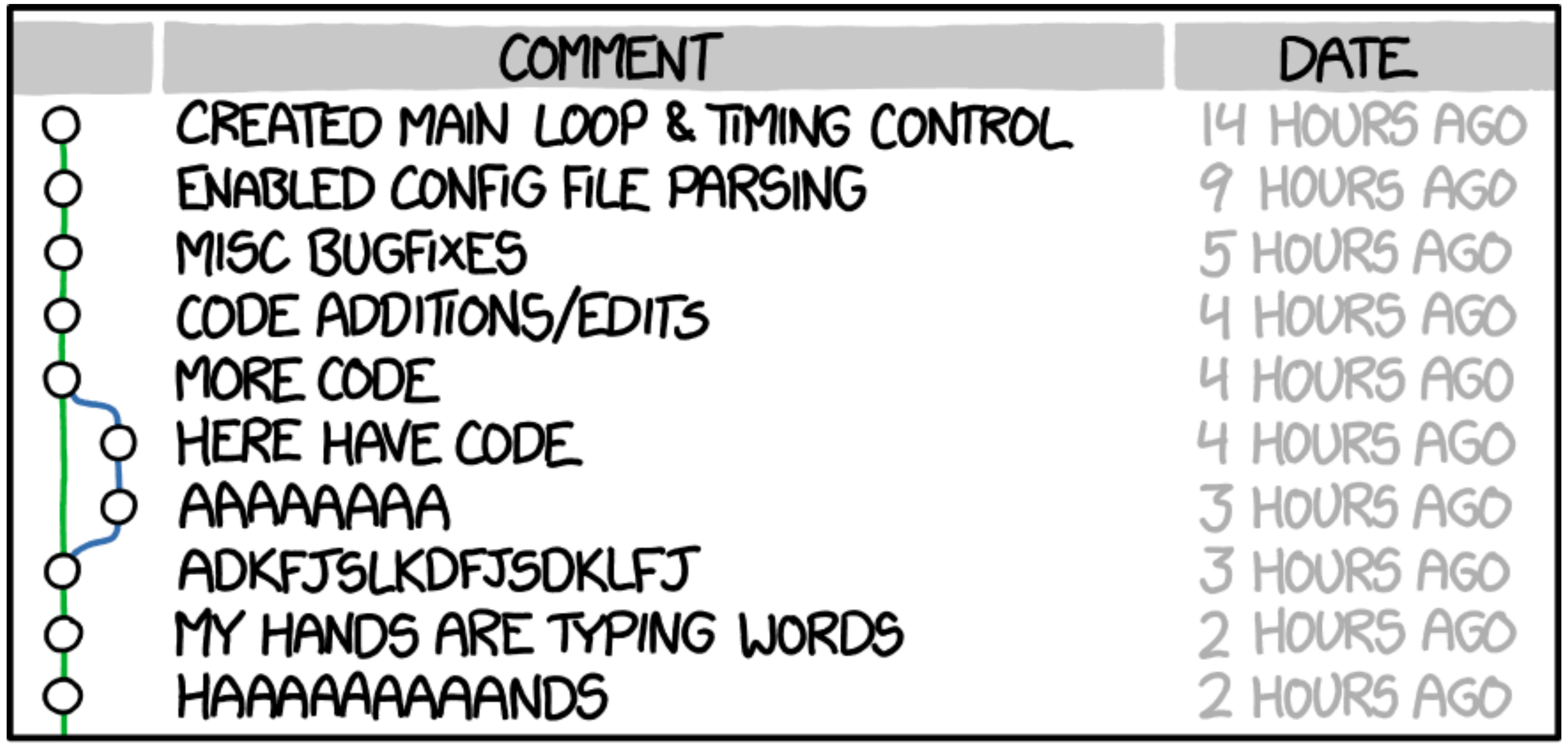
```
git commit -m "..."
```

- Be descriptive but concise
- Trace ideas back (and filter unrelated things)

```
git commit -m "adds feature X"
```

```
git commit -m "fix bug #152"
```

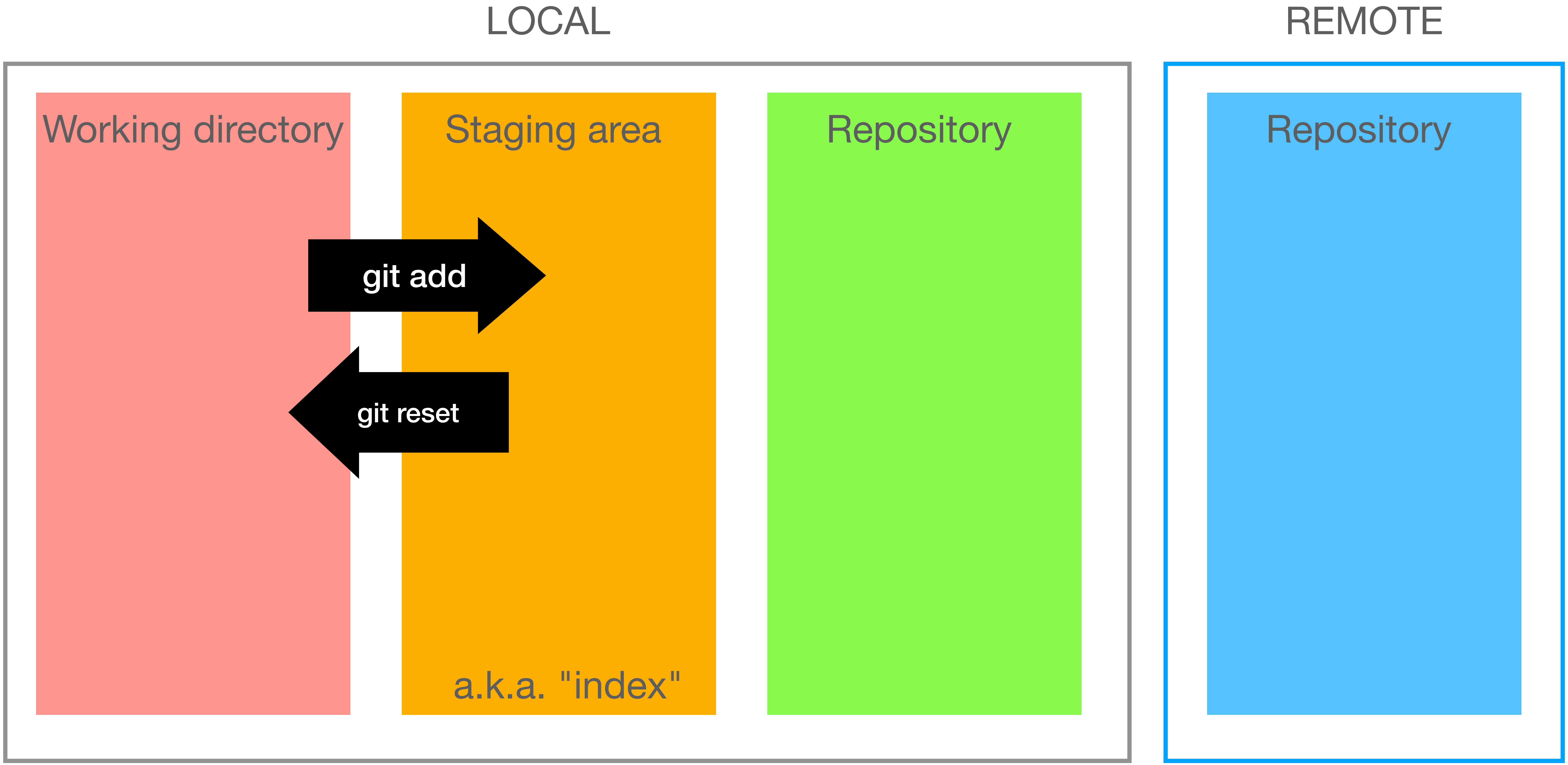
```
git commit --amend
```

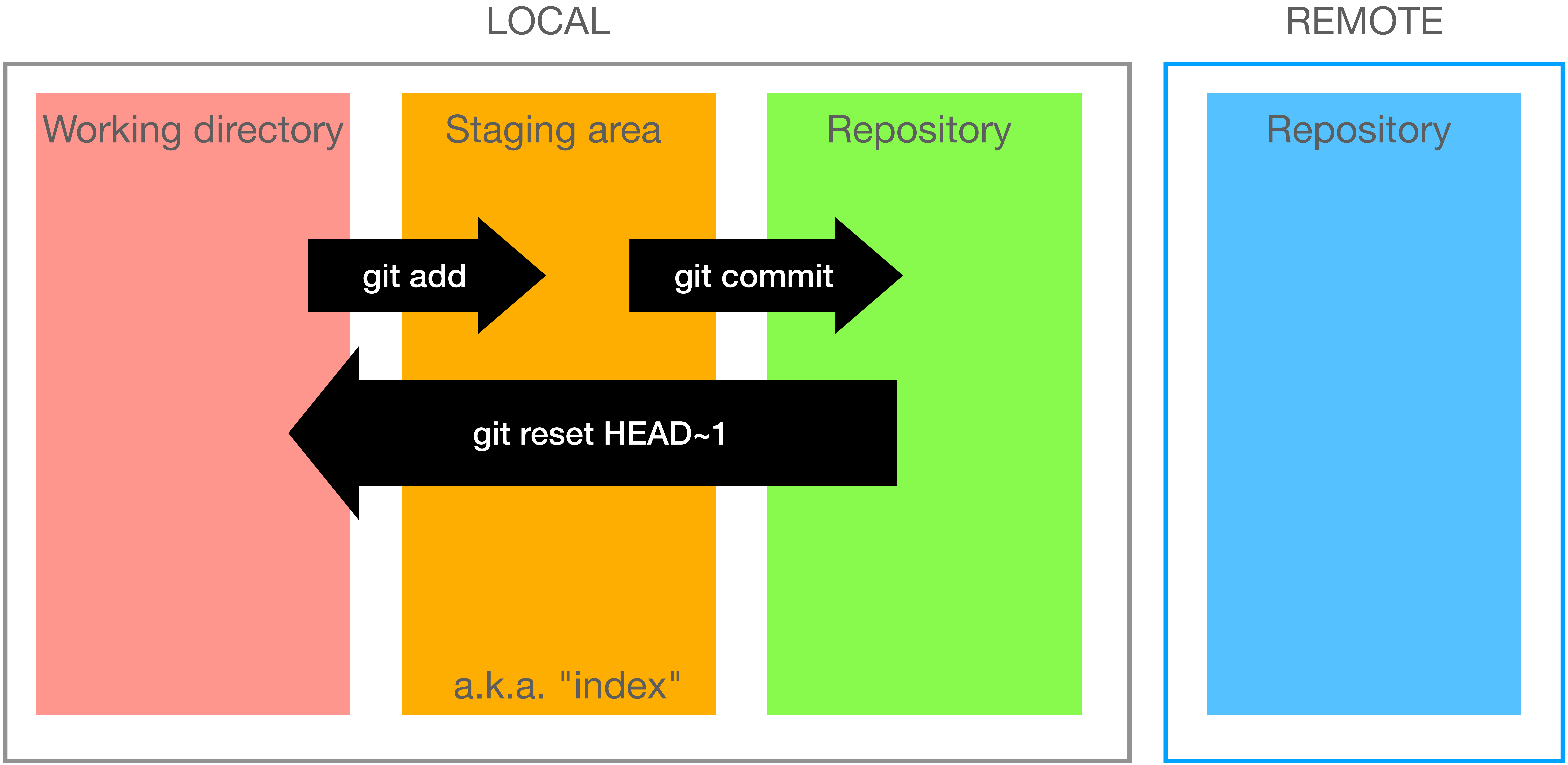


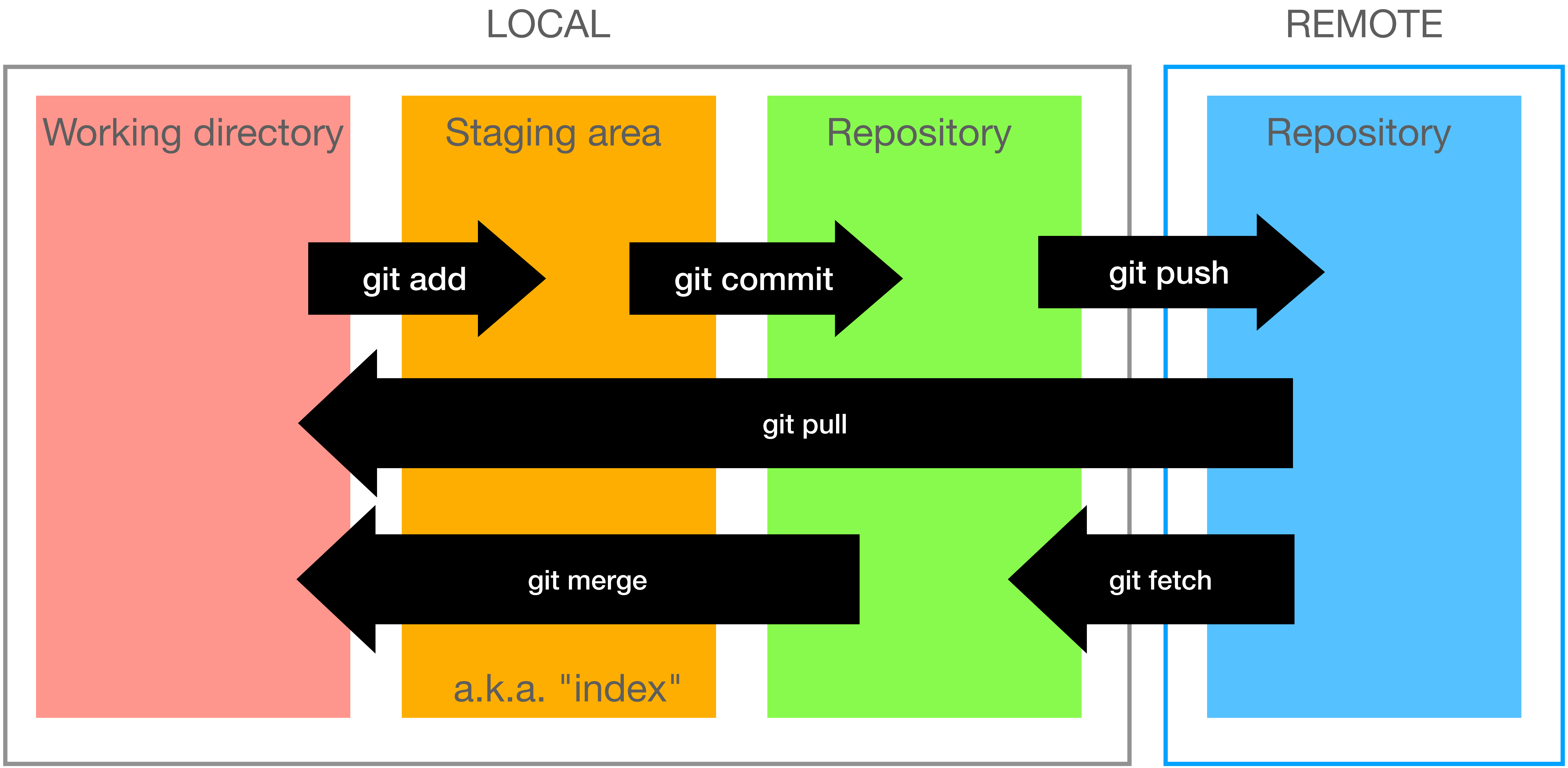
COMMENT	DATE
CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
ENABLED CONFIG FILE PARSING	9 HOURS AGO
MISC BUGFIXES	5 HOURS AGO
CODE ADDITIONS/EDITS	4 HOURS AGO
MORE CODE	4 HOURS AGO
HERE HAVE CODE	4 HOURS AGO
AAAAAAAA	3 HOURS AGO
ADKFJSLKDFJSDKLFJ	3 HOURS AGO
MY HANDS ARE TYPING WORDS	2 HOURS AGO
HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

https://xkcd.com/1296/


LOCAL
REMOTE
Working directory
Staging area
Repository
Repository
git add
git reset
a.k.a. "index"


LOCAL
REMOTE
Working directory
Staging area
Repository
Repository
git add
git commit
git reset HEAD~1
a.k.a. "index"


LOCAL
REMOTE
Working directory
Staging area
Repository
Repository
git add
git commit
git push
git pull
git merge
git fetch
a.k.a. "index"

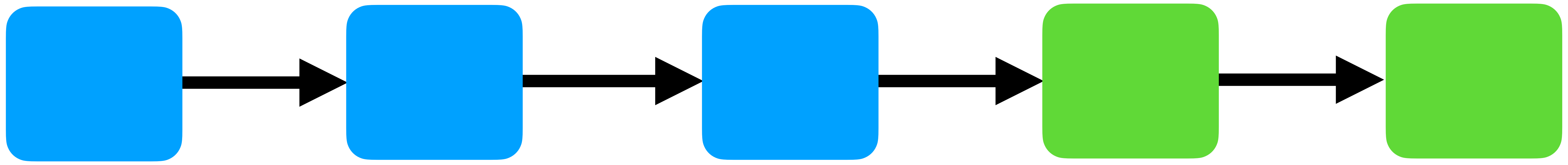
Why `git pull` (fetch-then-merge) is bad….

- **May generate a "merge" commit (if both changed)**
  - ▸ **Muddies history**
  - ▸ **Merge commit misattributes changes**

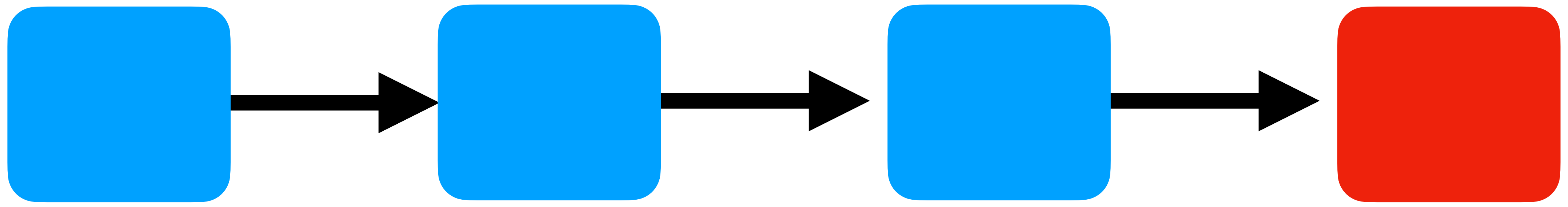
Instead: fetch-then-*rebase*

Changes in two places….

Local

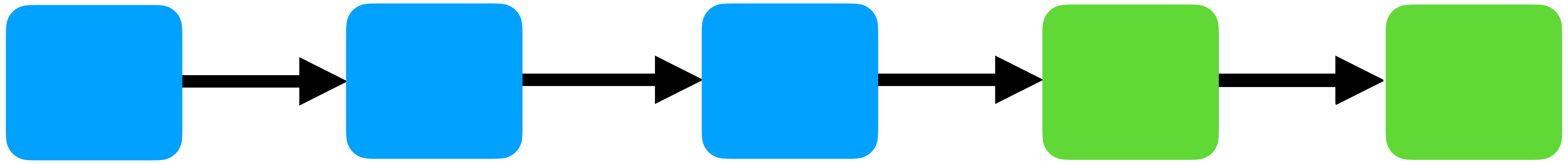


Remote



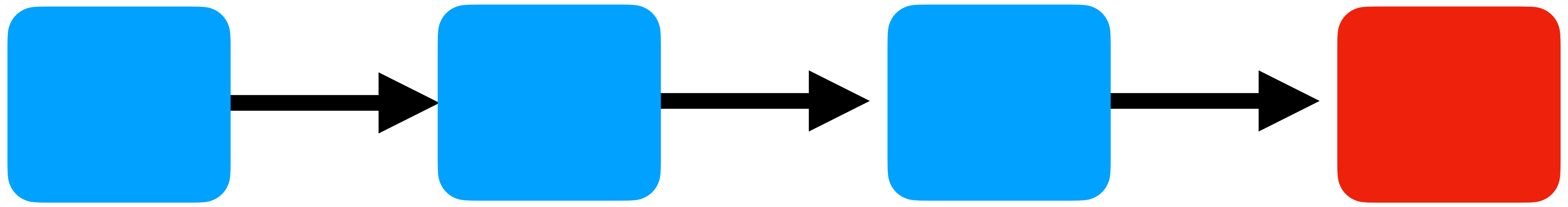
`git pull —rebase` *to the rescue!*

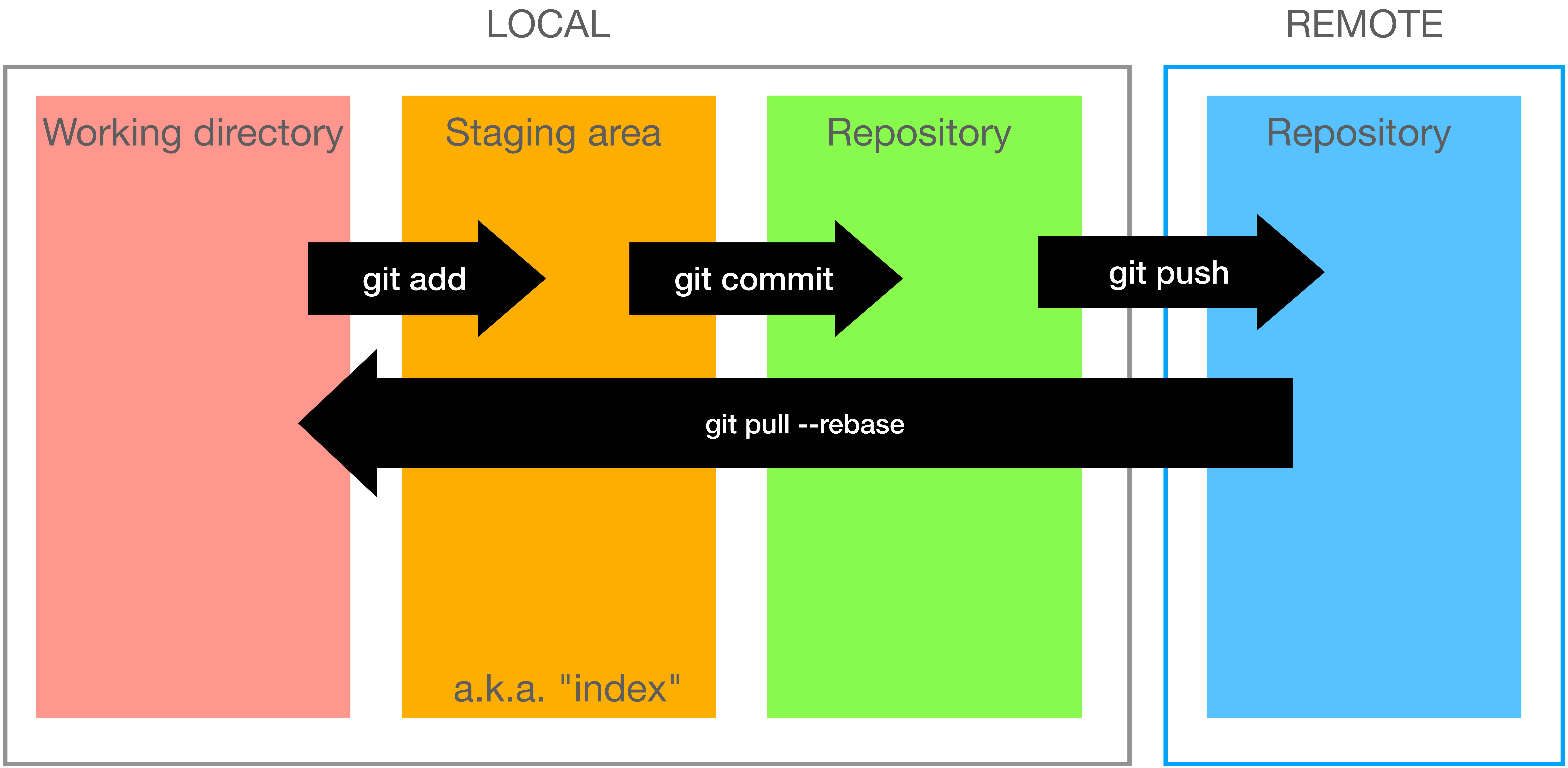
Local



Note: order of commits is not time-dependent

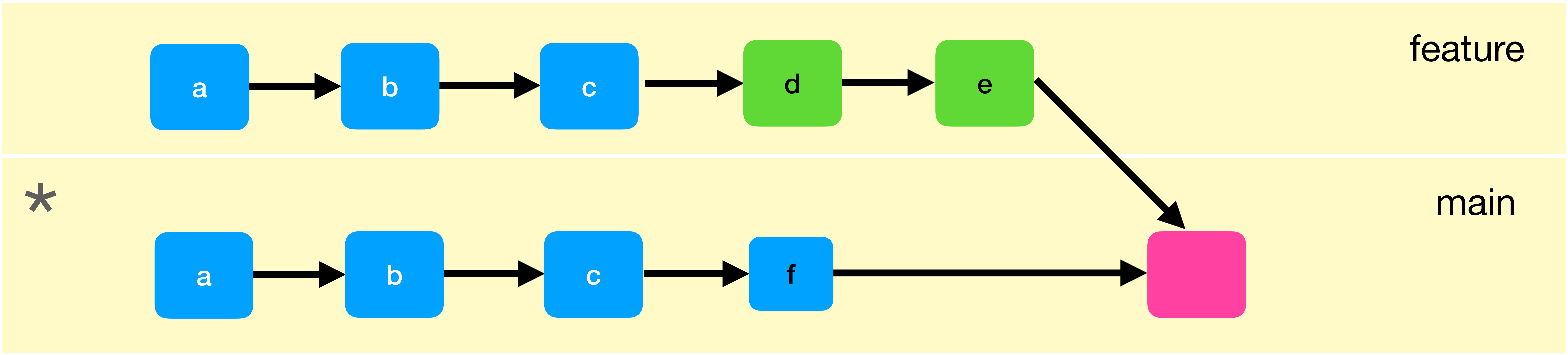
Remote




LOCAL
REMOTE
Working directory
Staging area
Repository
Repository
git add
git commit
git push
git pull --rebase
a.k.a. "index"

Branches

Merge commit created if changes in both



```
git branch                          see branches
git branch feature                  create new branch
git checkout feature                switch branch
…..
git checkout main                   switch branch
git merge feature                   merge branches
```

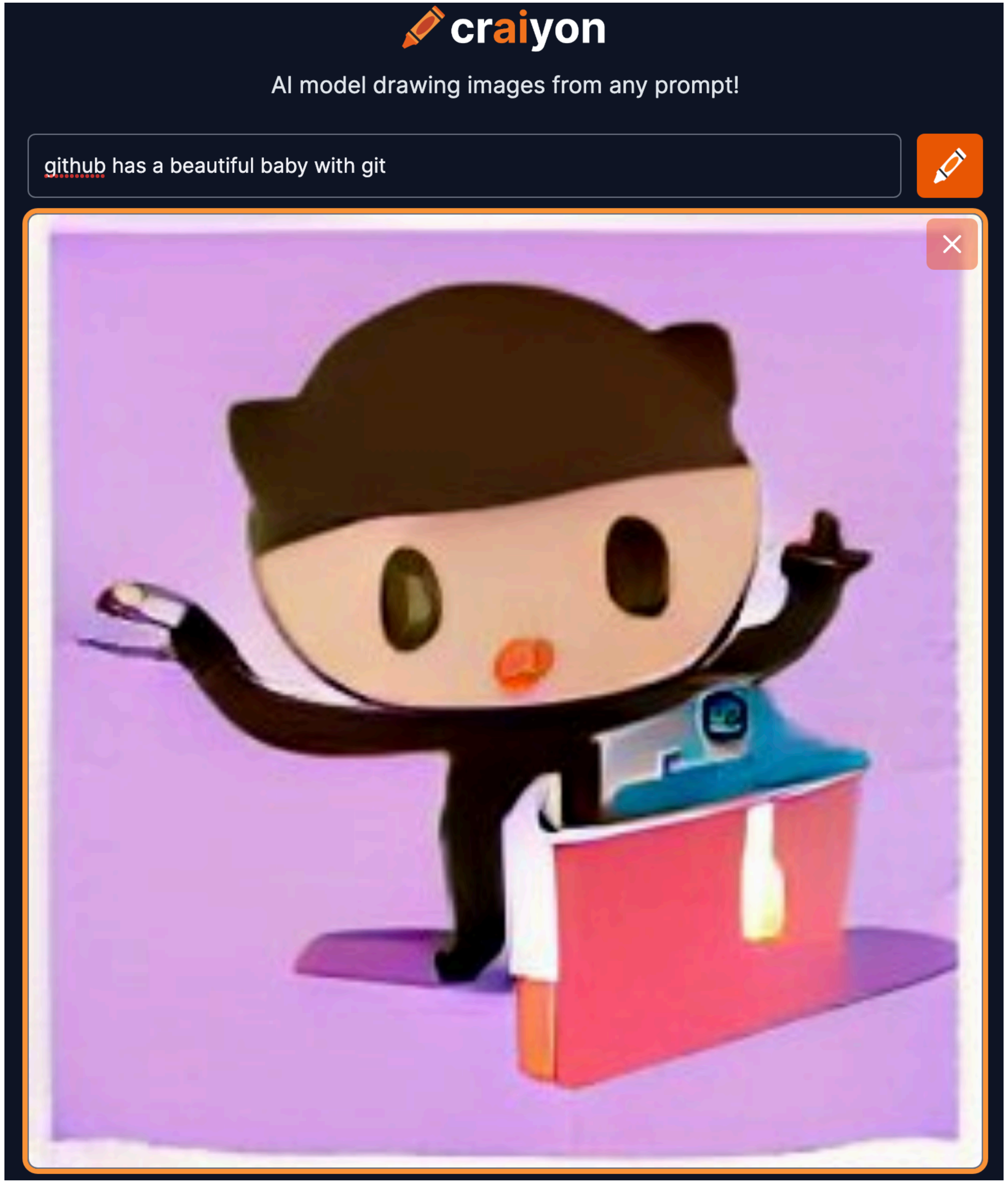
Git commands (so far)

- git status
- git log
- git show
- git add (-p)
- git reset
- git commit
- git push

- git pull —rebase
- git branch
- git checkout
- git merge

git

GitHub

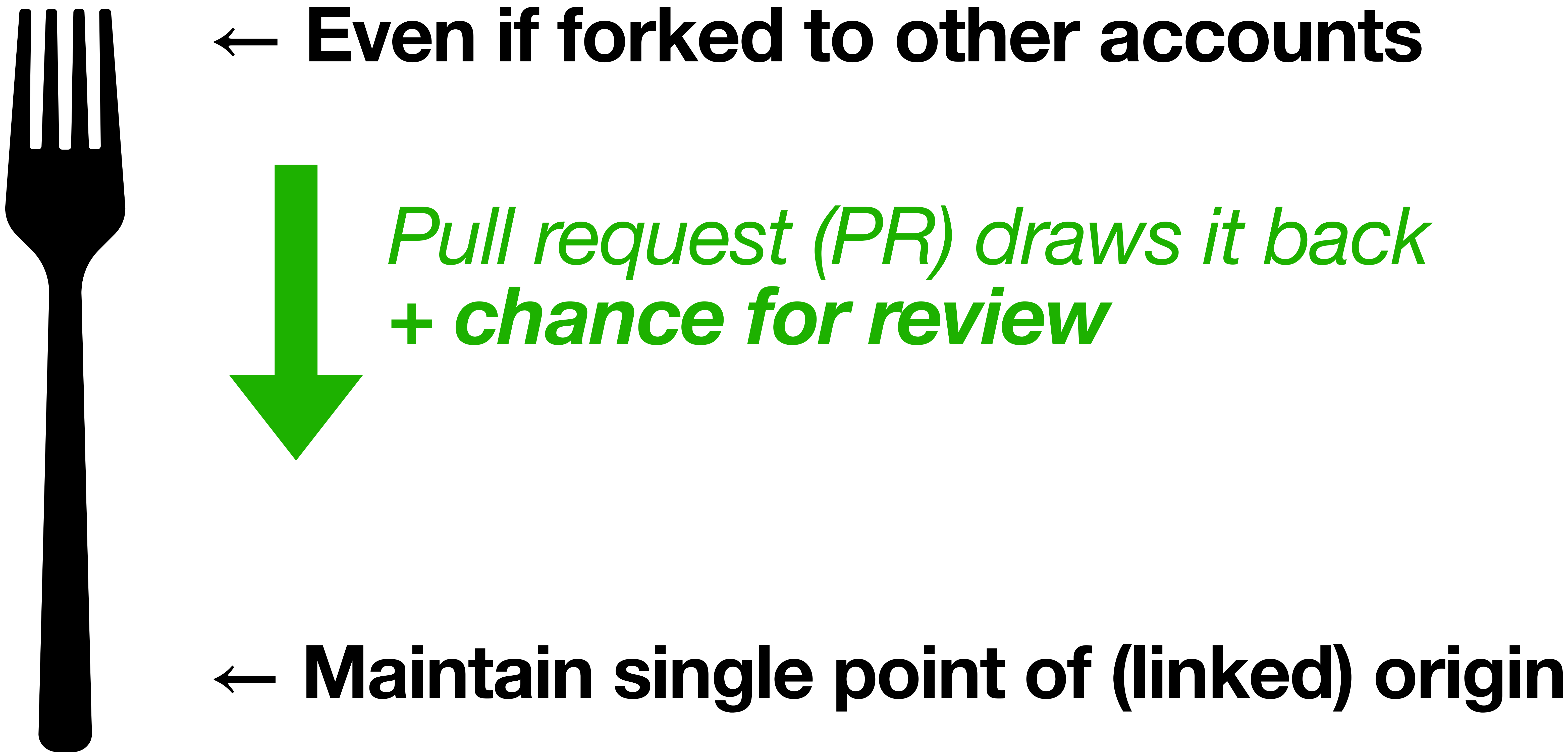


craiyon

AI model drawing images from any prompt!

github has a beautiful baby with git

Forking and pull requests



← **Even if forked to other accounts**

Pull request (PR) draws it back
+ chance for review

← **Maintain single point of (linked) origin**

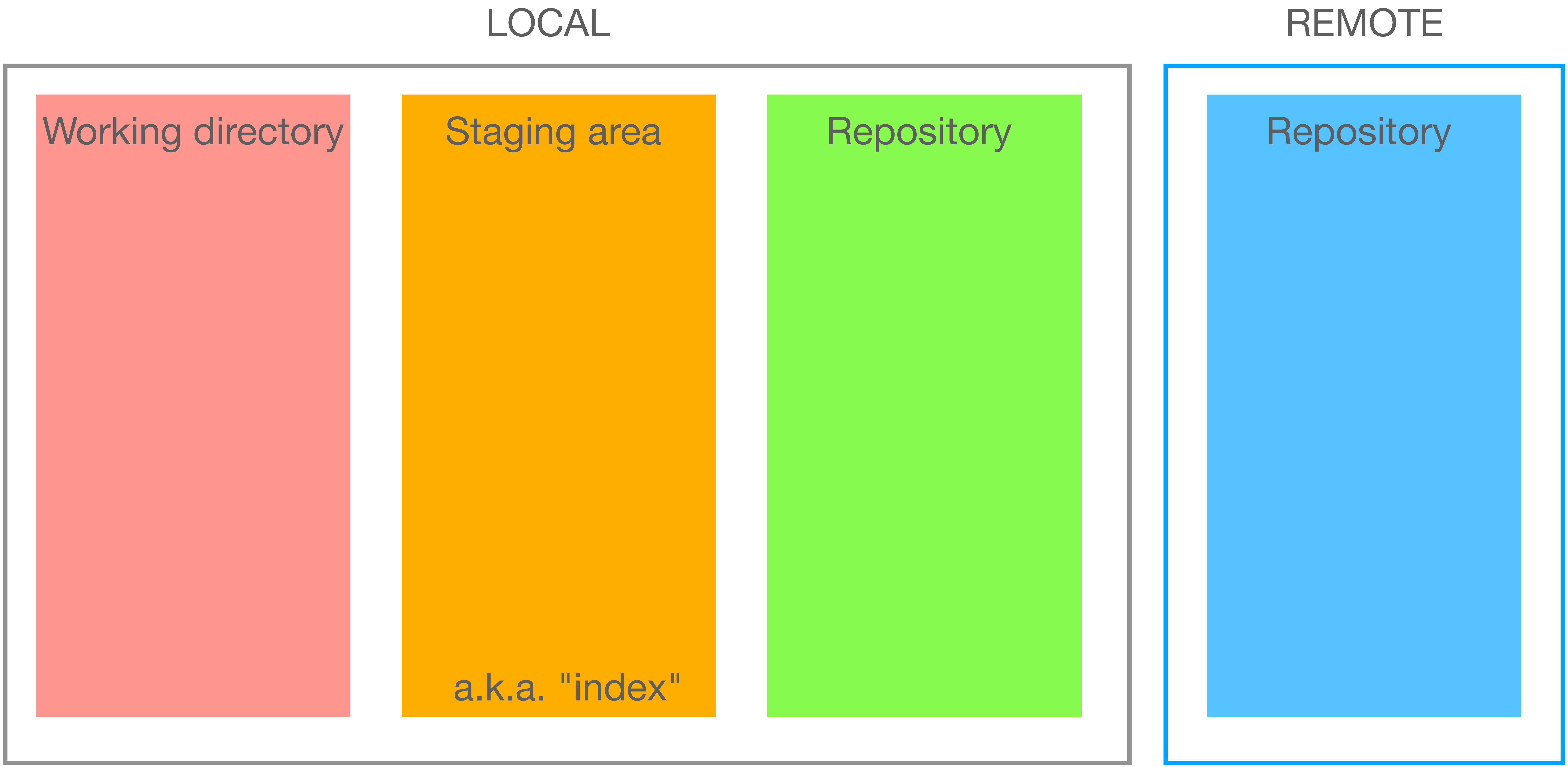
- *Don’t copy, fork!*
- *Fork if you actually anticipate contributing*
- *Follow-up on intention with PR changes back*

Issues - live demo

- Good organising principle
- Recall `git commit -m “working on bug #152”`
- Auto close using words `fix`, `close`, or `resolve` (and most conjugations)

Tags (git) and releases (github) - live demo

- `git tag`
- `git tag name`
- `git push –tag`



LOCAL

Working directory

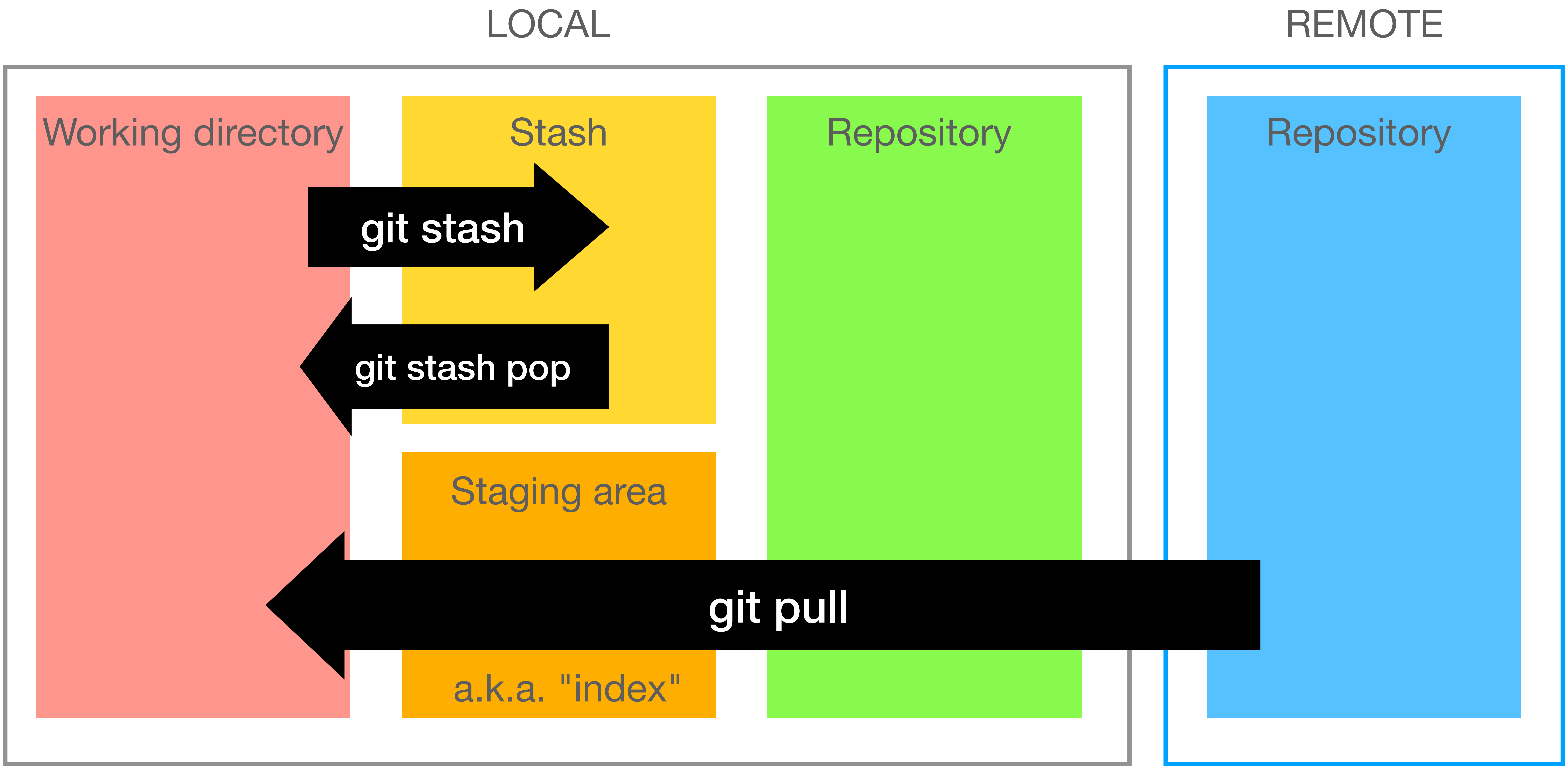
Staging area

a.k.a. "index"

Repository

REMOTE

Repository

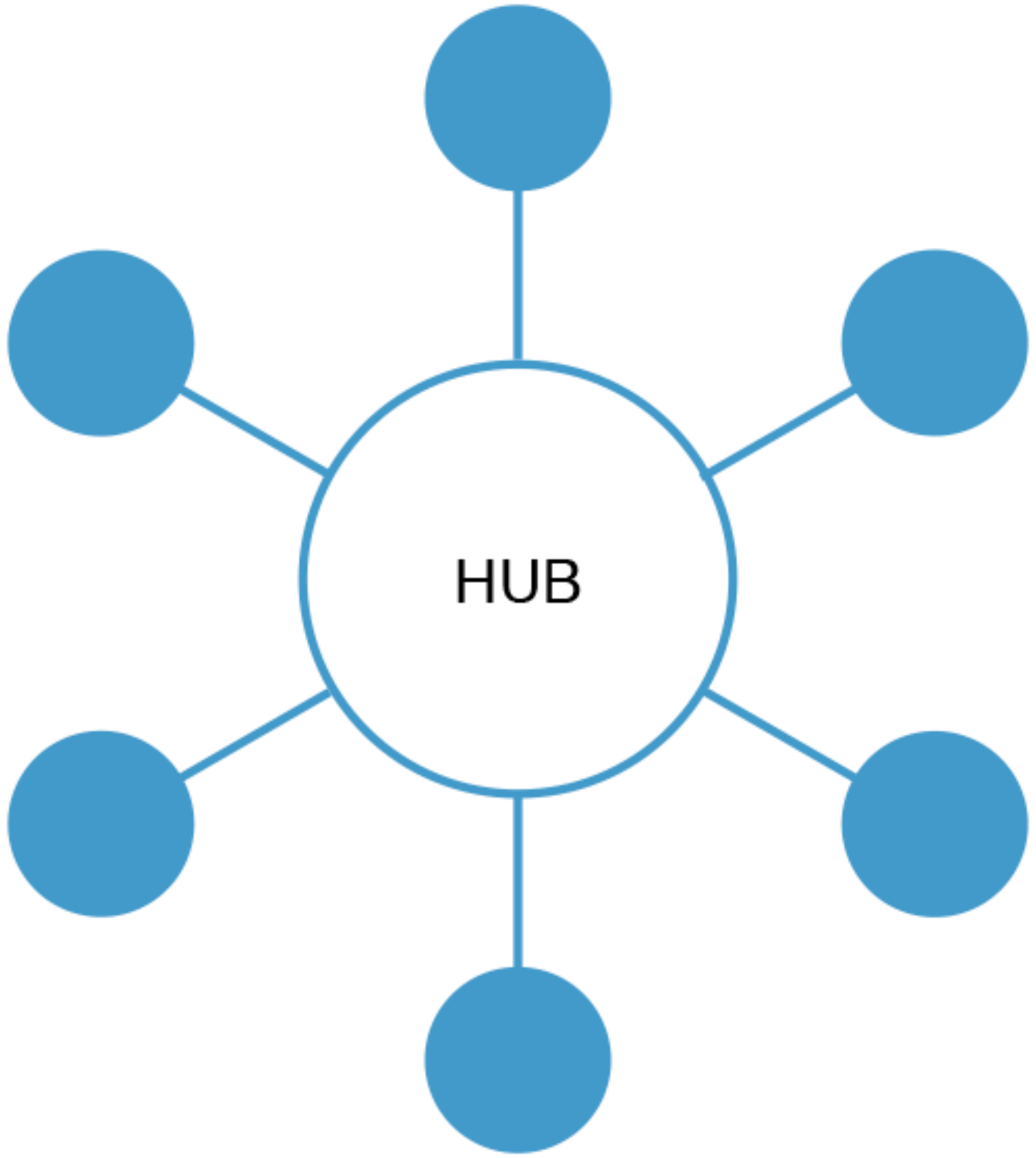

LOCAL
REMOTE
Working directory
Stash
Repository
Repository
git stash
git stash pop
Staging area
git pull
a.k.a. "index"

Exercise and break

Project management

Summary

- Forks and PRs for hub-and-spoke model
- Organisation with
  - Issues
  - PRs
  - Project board
- *Accountability* and *traceability*



Later in the week…much more

- Code review via PRs
- GitHub "Actions" for CI/CD (continuous integration/deployment)