Software Systems

Design – 3B

Version Control

Marcus Gerhold

Based on slides by Klaas Sikkel

Contents

- Purpose of this lecture:

  A conceptual model
  of distributed version control

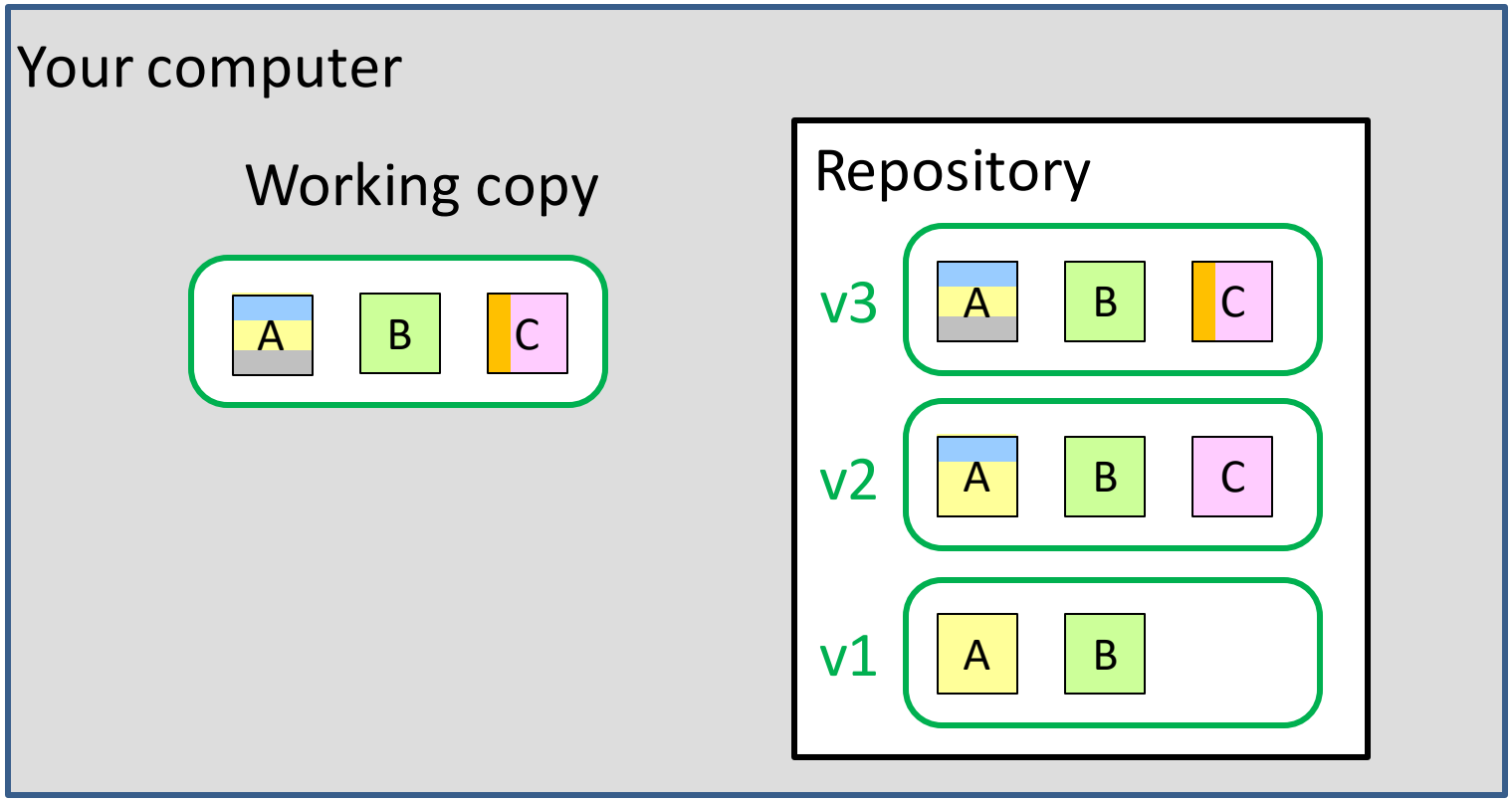
- Only a few words about systems and implementations

- 2nd hour: you can start with lab session 3b

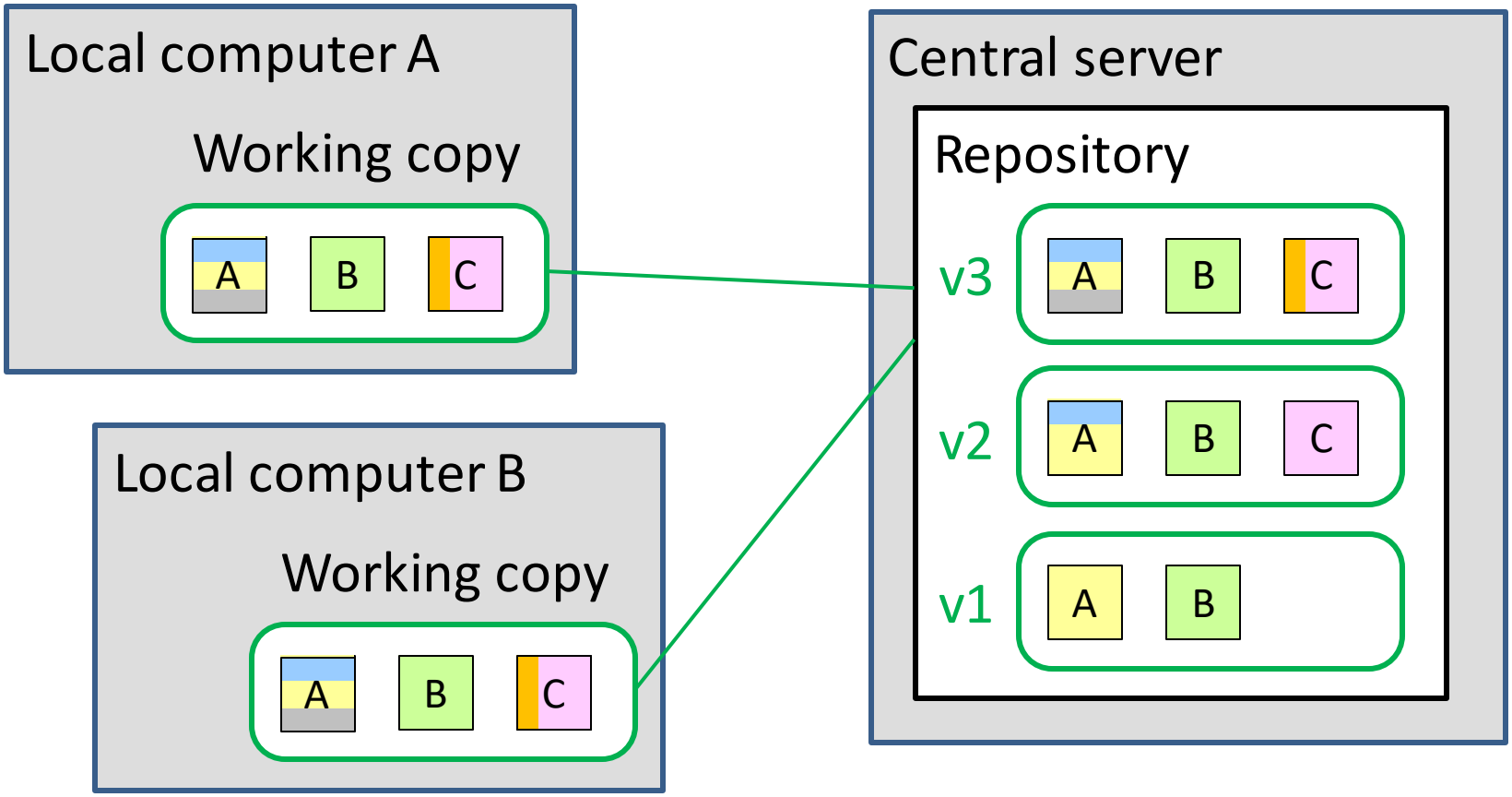
Why version control?

1. You made a wrong change to your program (report, diagram, …)
   - You realise this the next day, after also making some *correct* changes
   - Now you want to roll back history
2. You and your lab partner both locally changed the same program
   - Neither of you remember all your changes precisely
   - You *certainly* do not want to do them again
   - Now you want to combine the changes
3. Your machine crashed
   - You kept all your stuff there
   - Rather unfortunate for you
4. The remote file server is down
   - You kept all your stuff there
   - Now you can only twiddle thumbs and wait for maintenance

Local version control

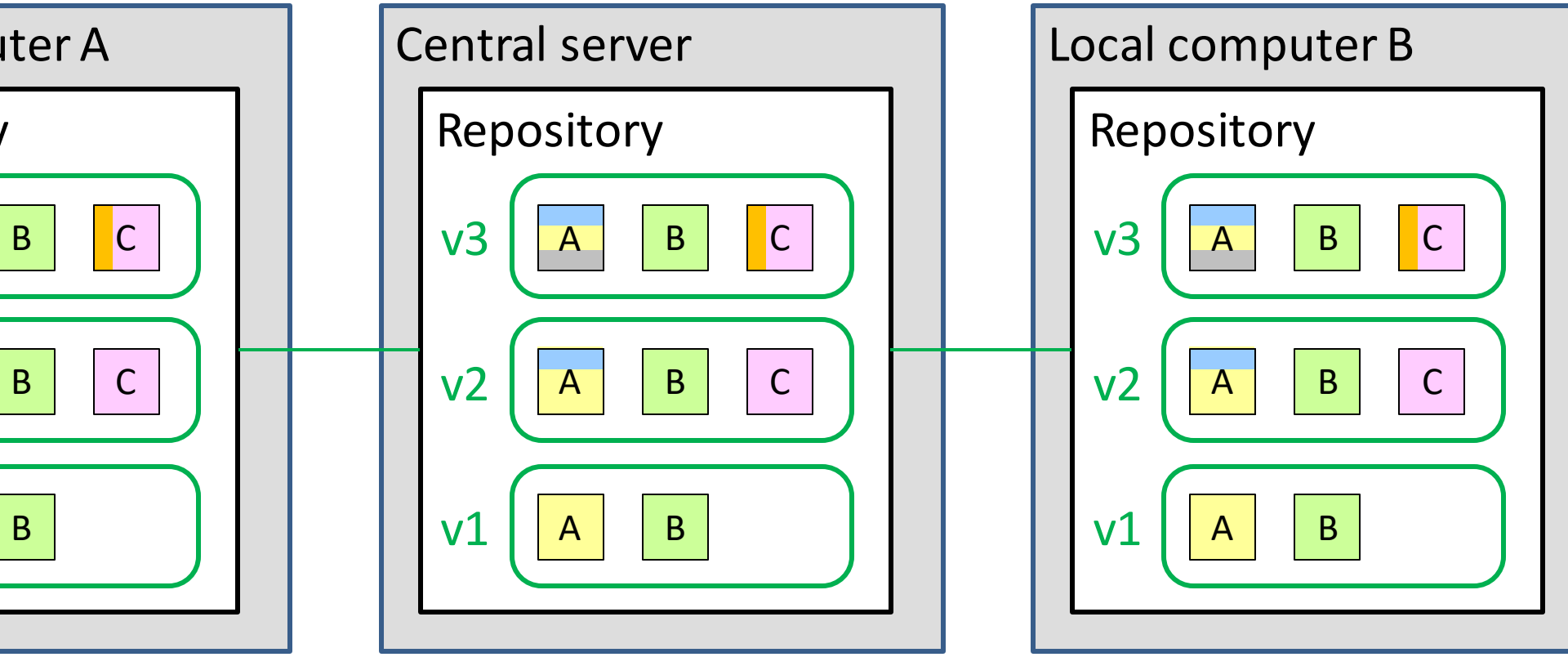


Central version control



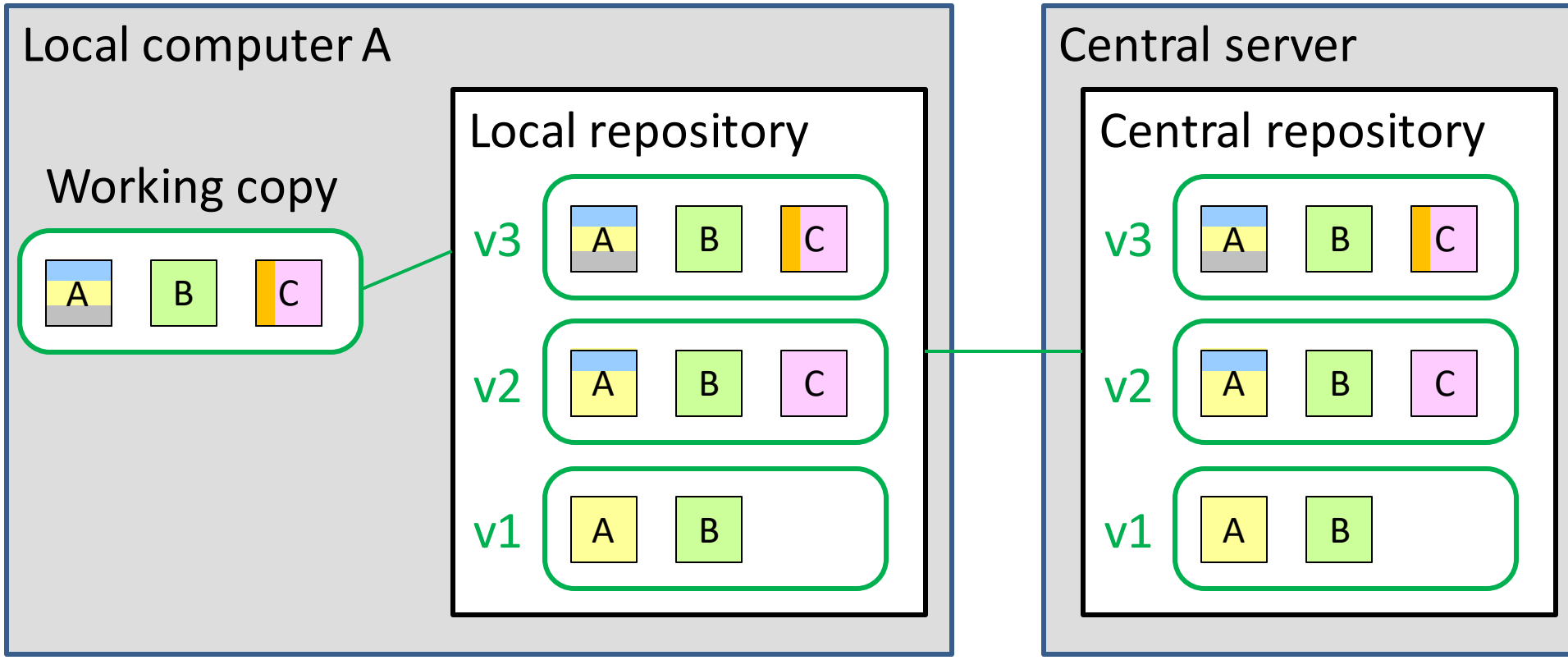
Examples: *CVS* (*Concurrent Versioning System*), *SVN* (*SubVersioN*)

Distributed version control



Examples: *Git, Mercurial*

Distributed version control



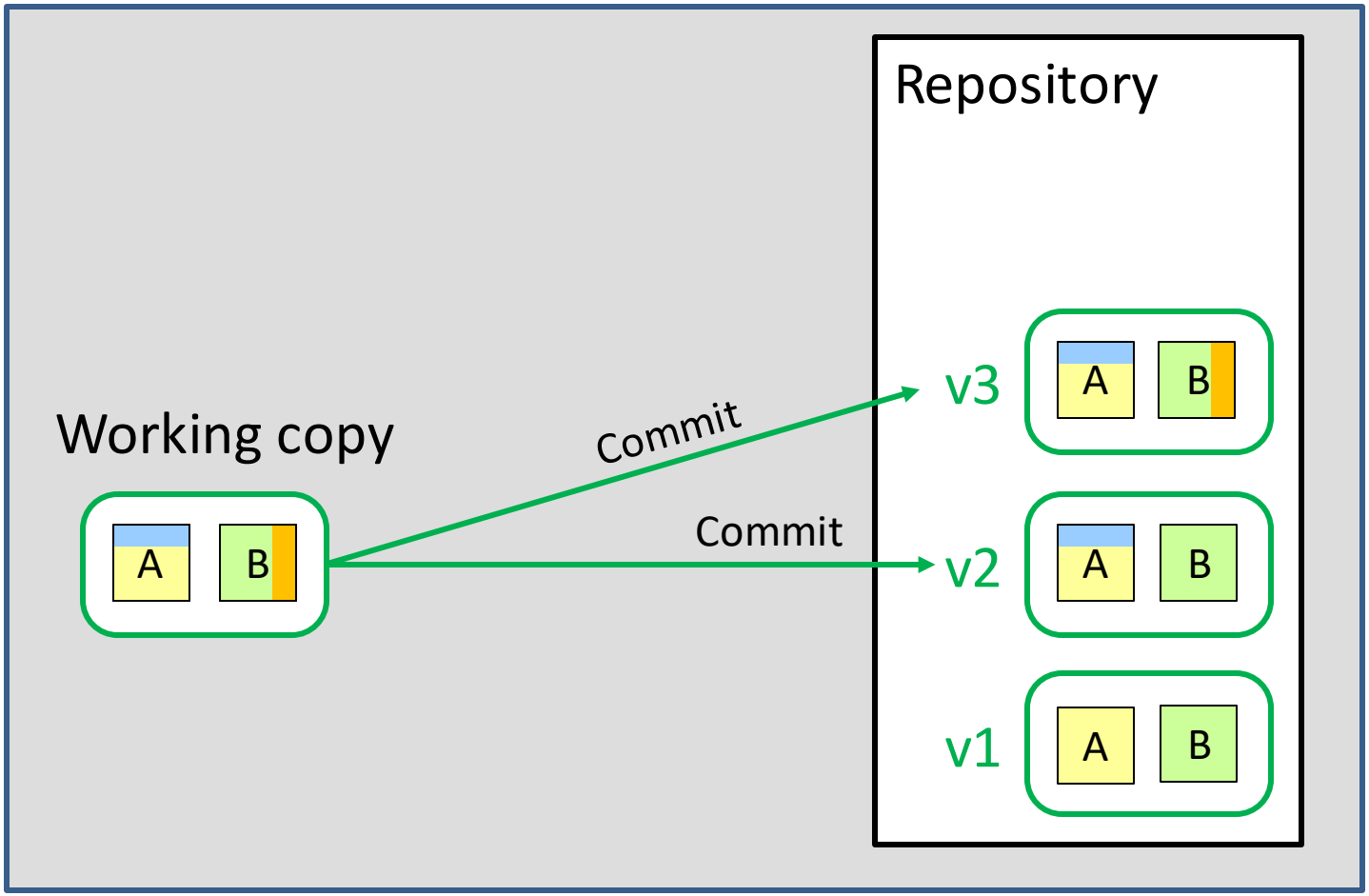
Examples: *Git, Mercurial*

Why version control?

1. You made a wrong change to your program (report, diagram, …)
   - You can *roll back* a previous change without affecting later changes
2. You and your lab partner both locally changed the same program
   - You can *merge* the changes
3. Your machine crashed
   - Everything (except last edits) is backed-up
4. The remote file server is down / you are offline
   - You have a complete repository locally

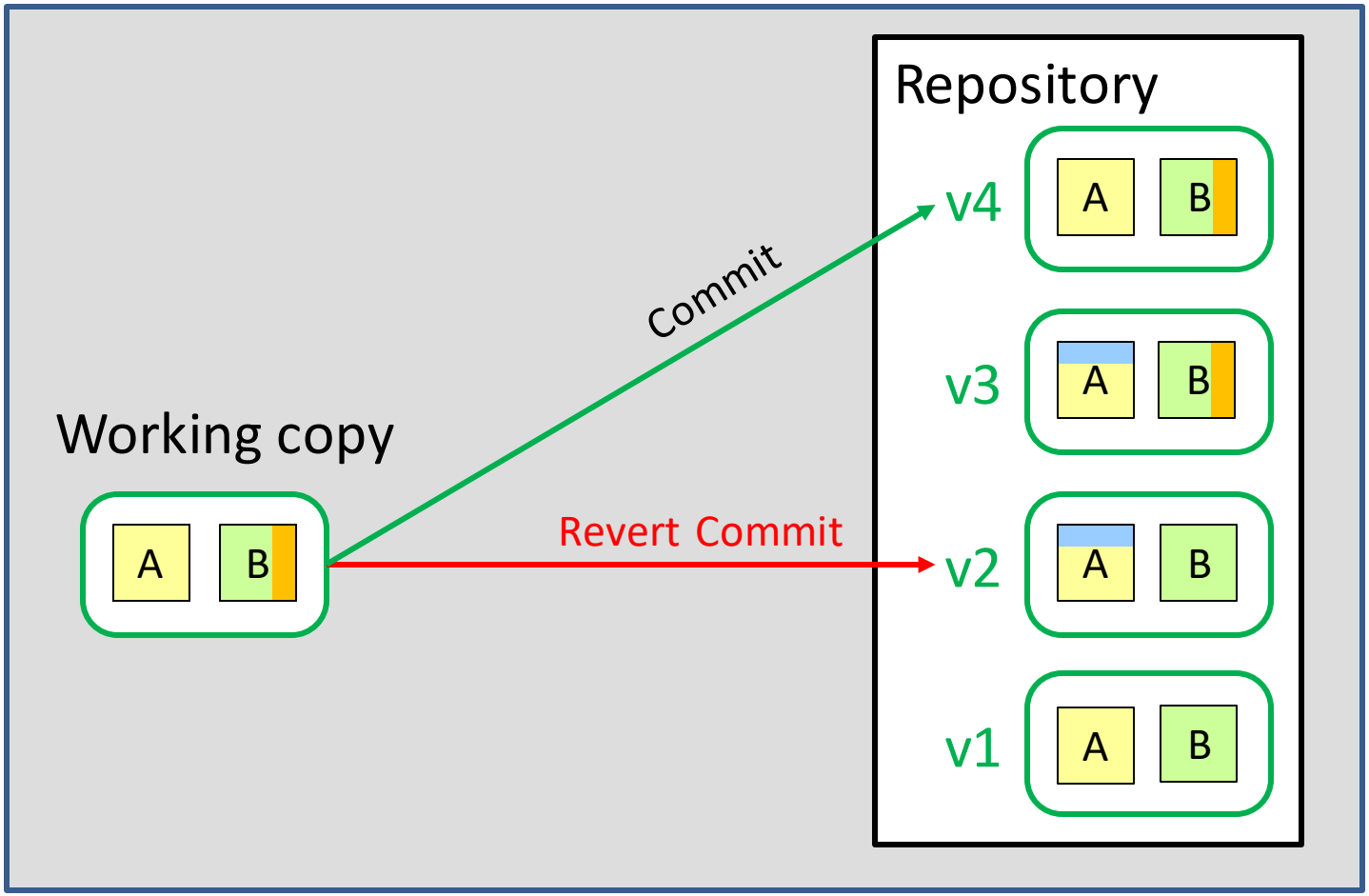
Roll back

- Undo an earlier change

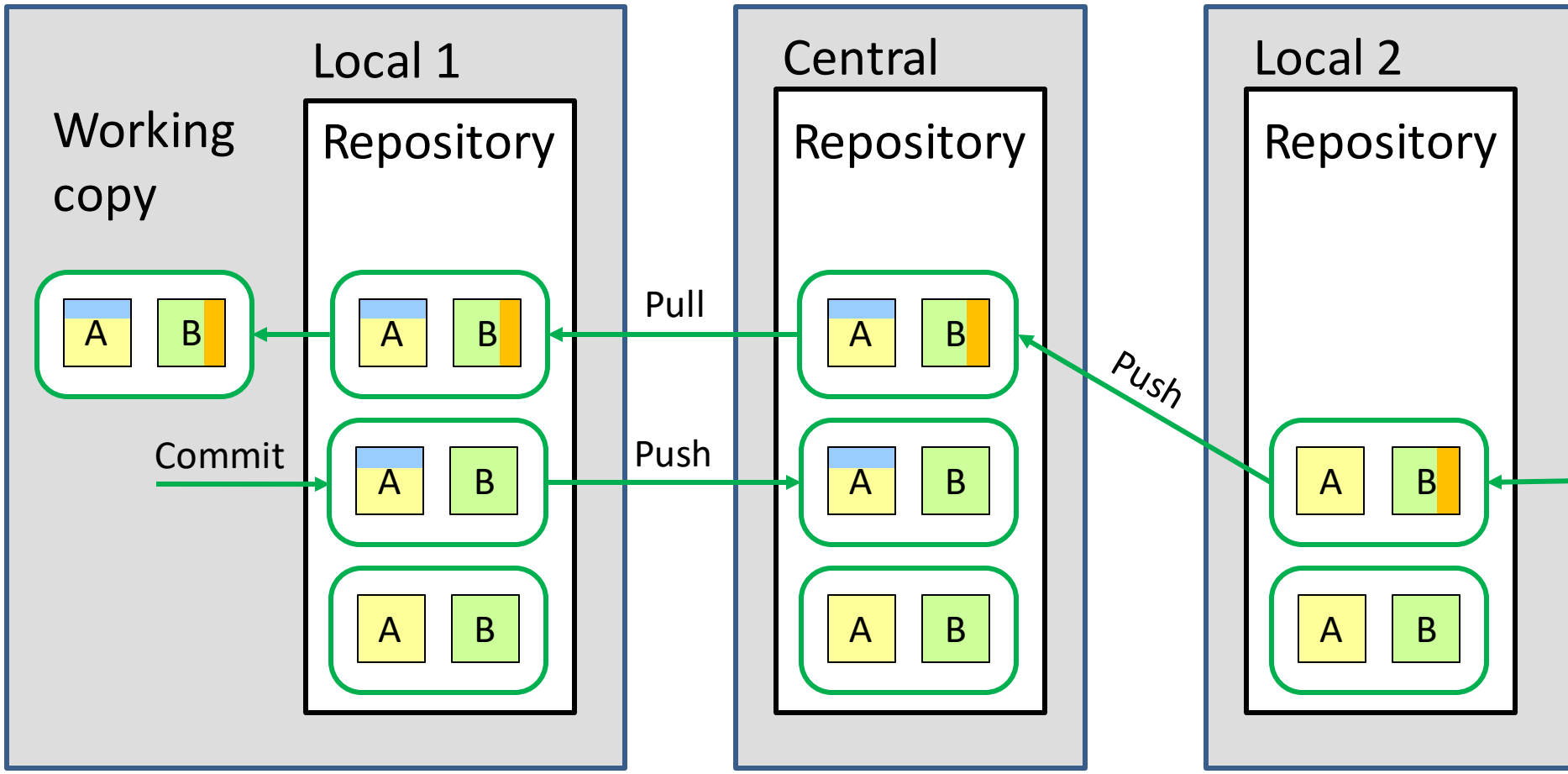


Roll back

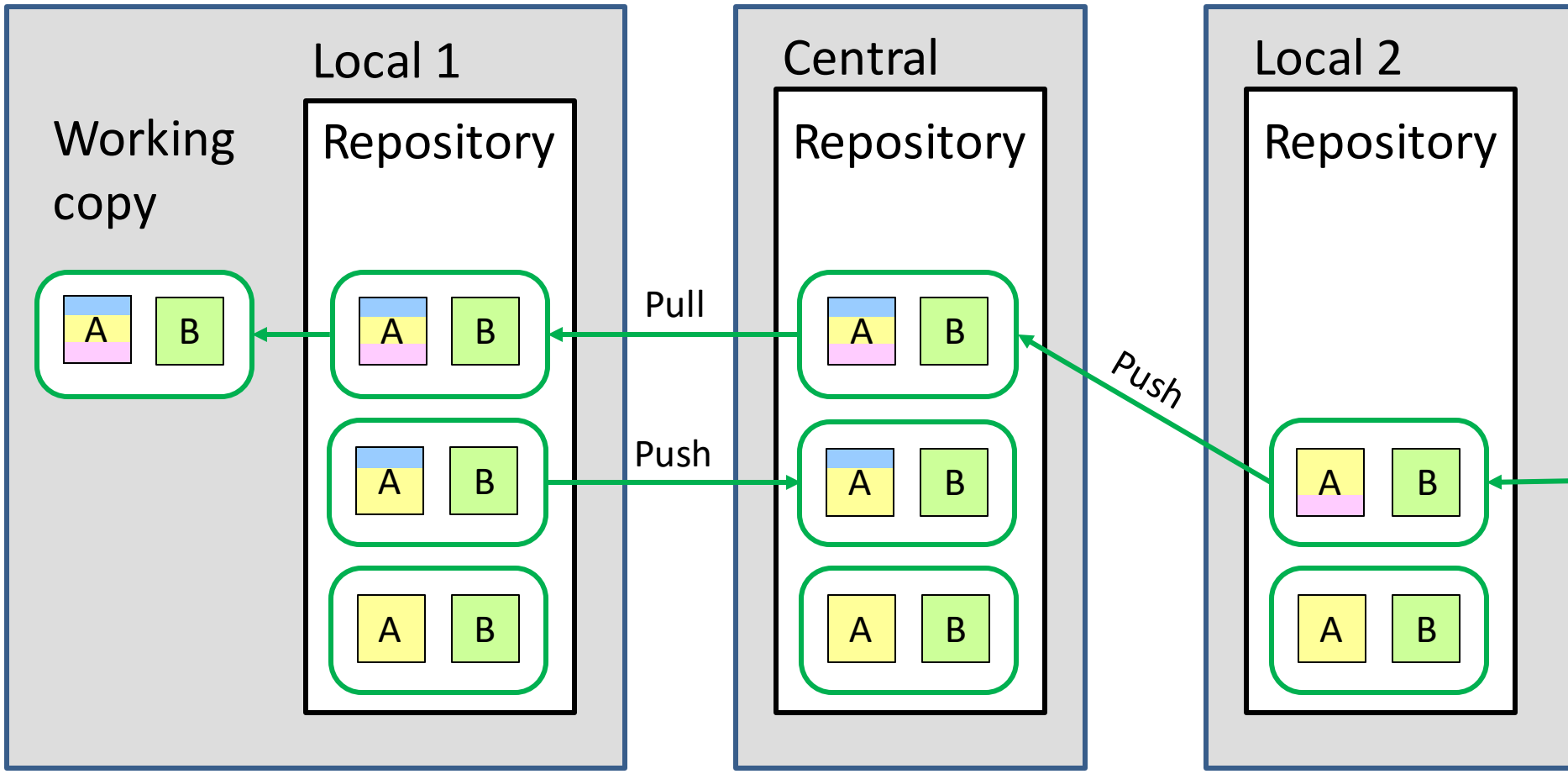
- Undo an earlier change



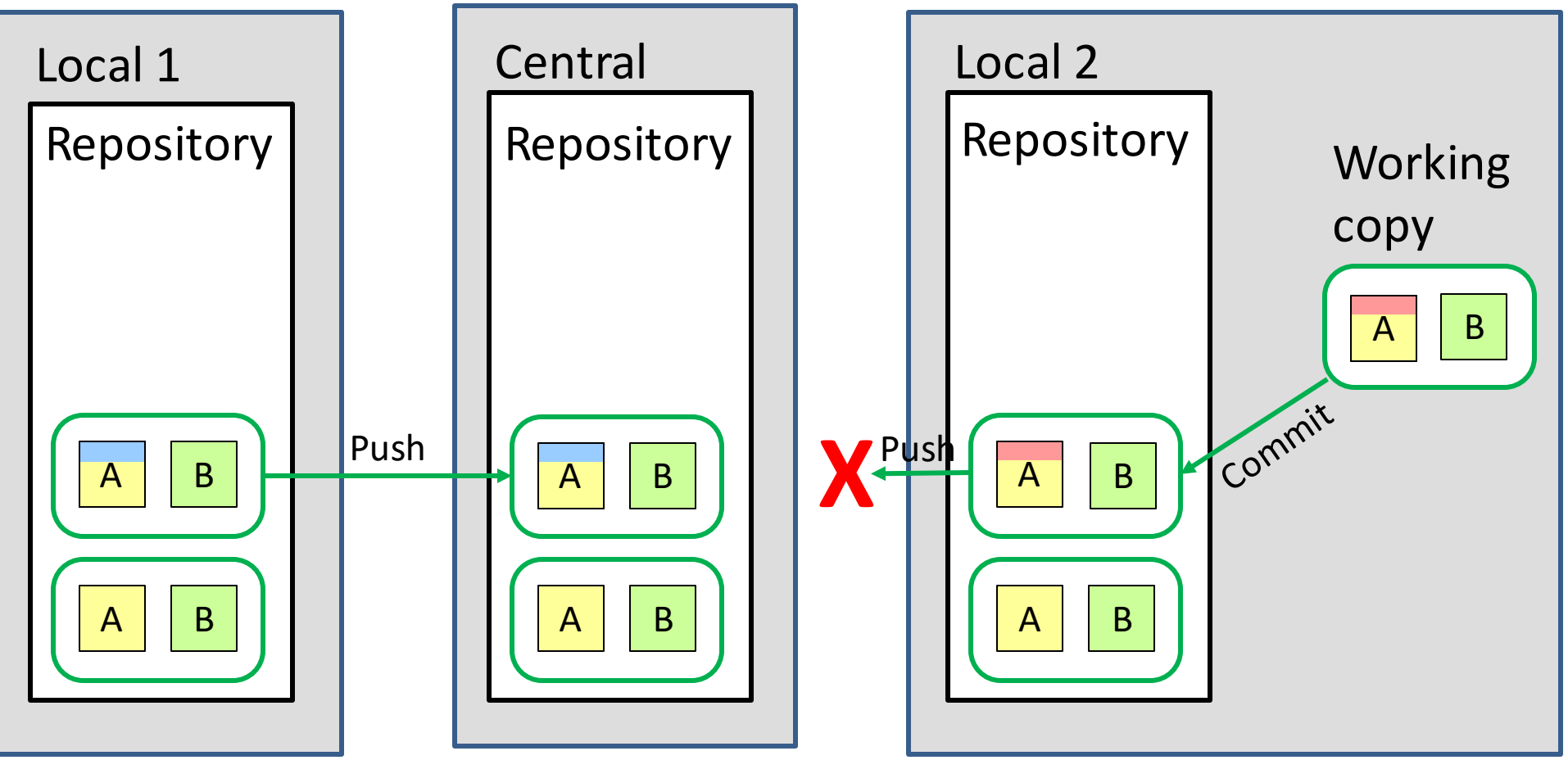
Push and pull



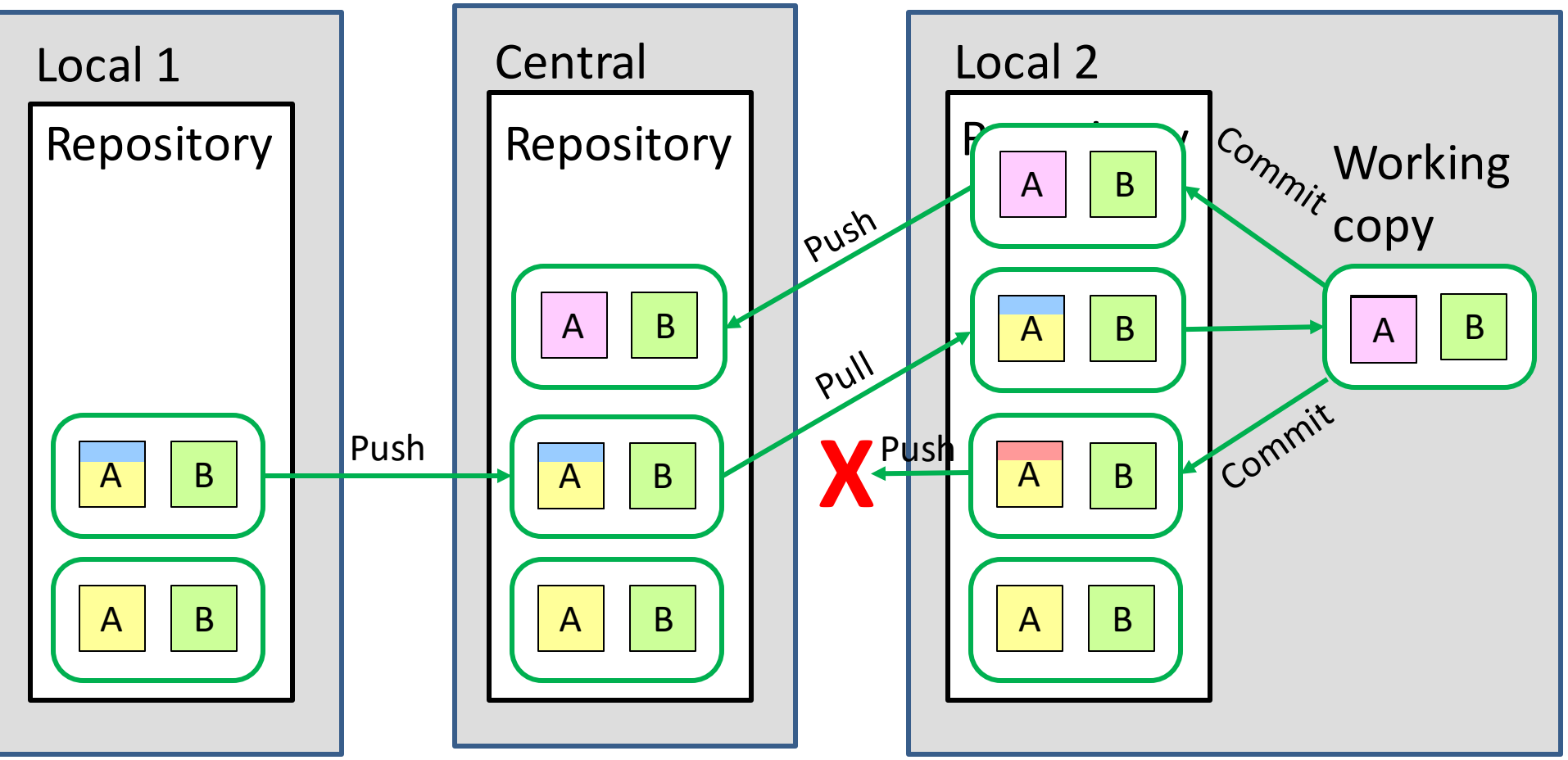
Push and pull:



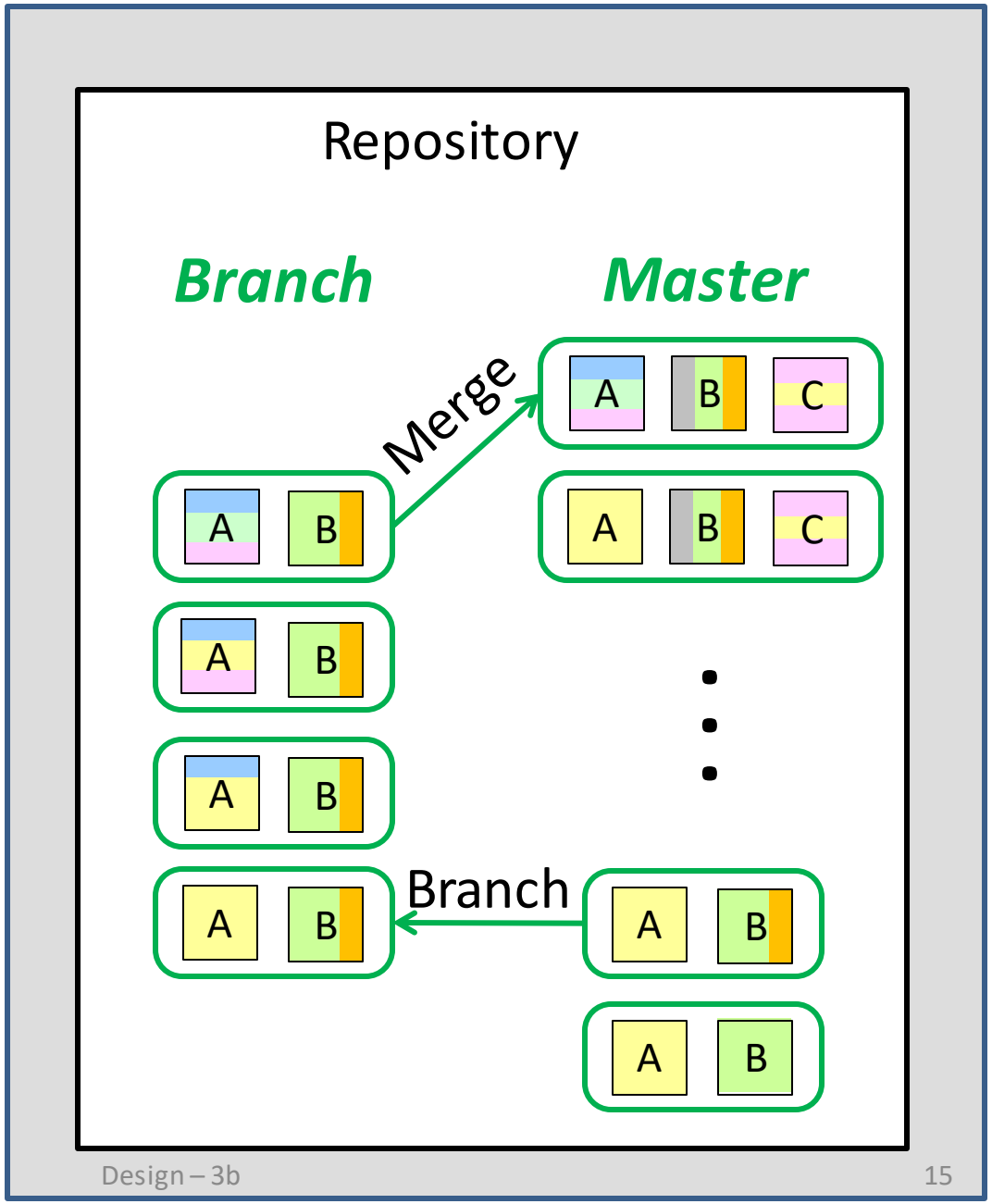
Push and pull: conflict



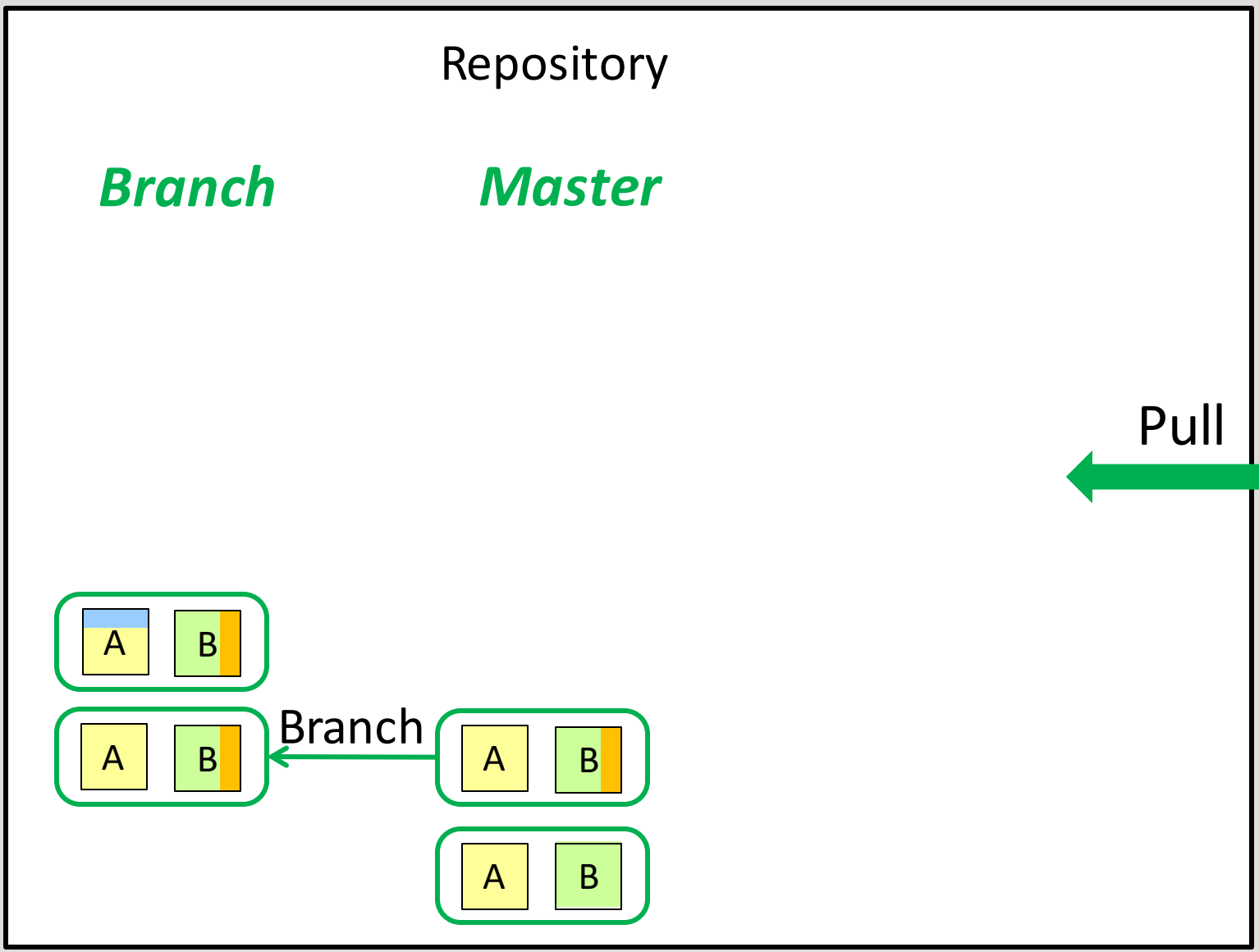
Push and pull: conflict



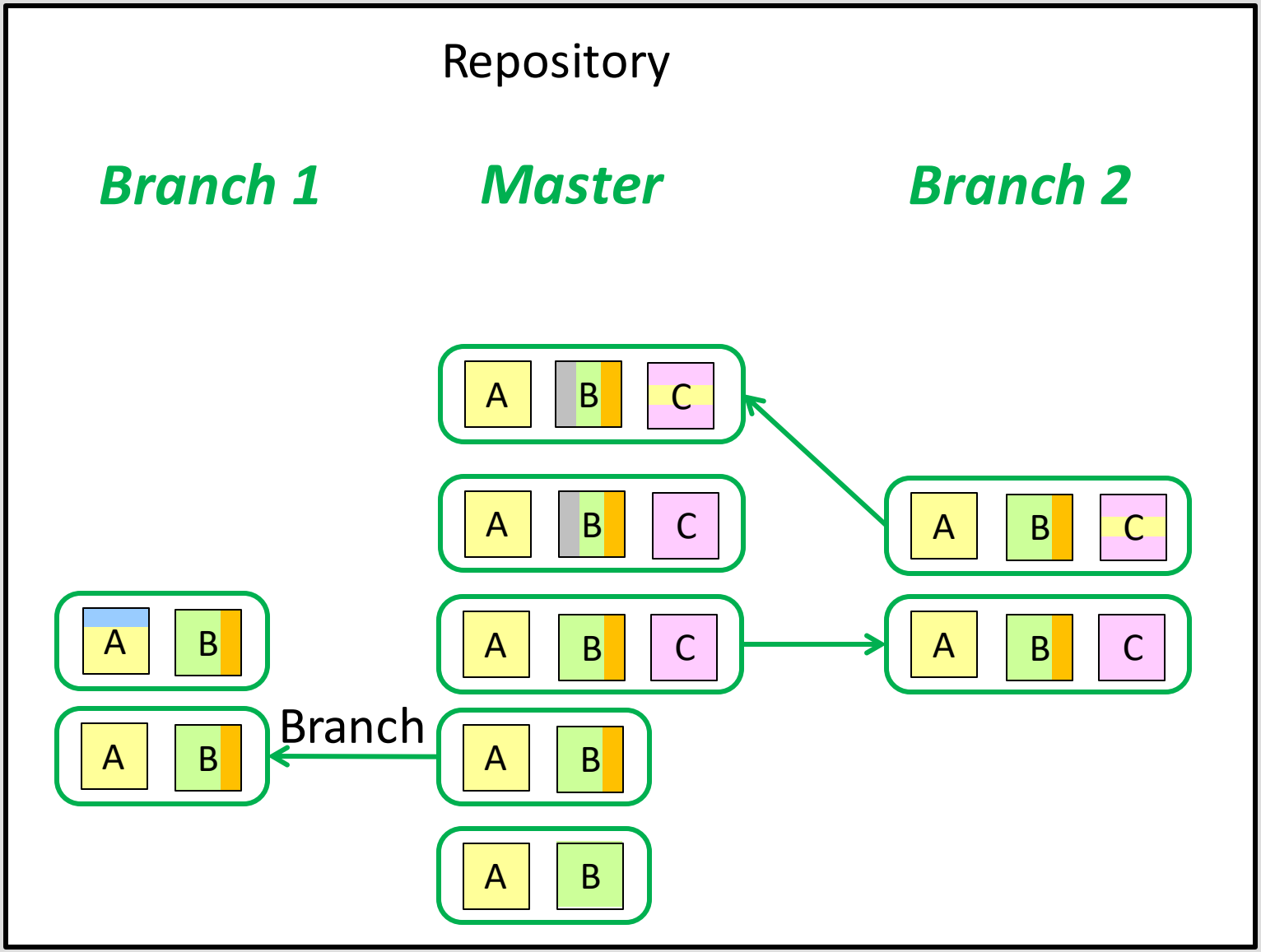
Branching



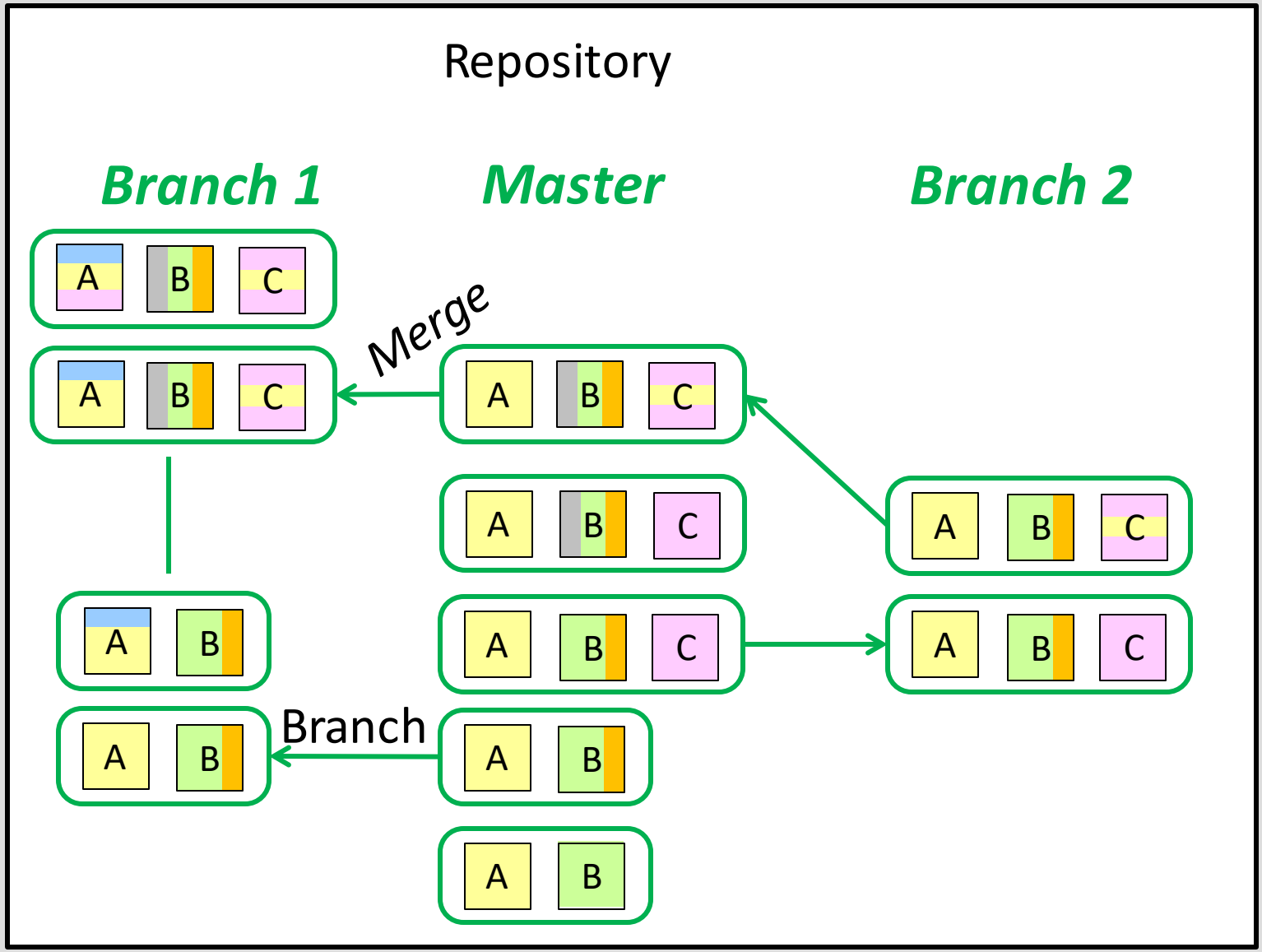
Branching



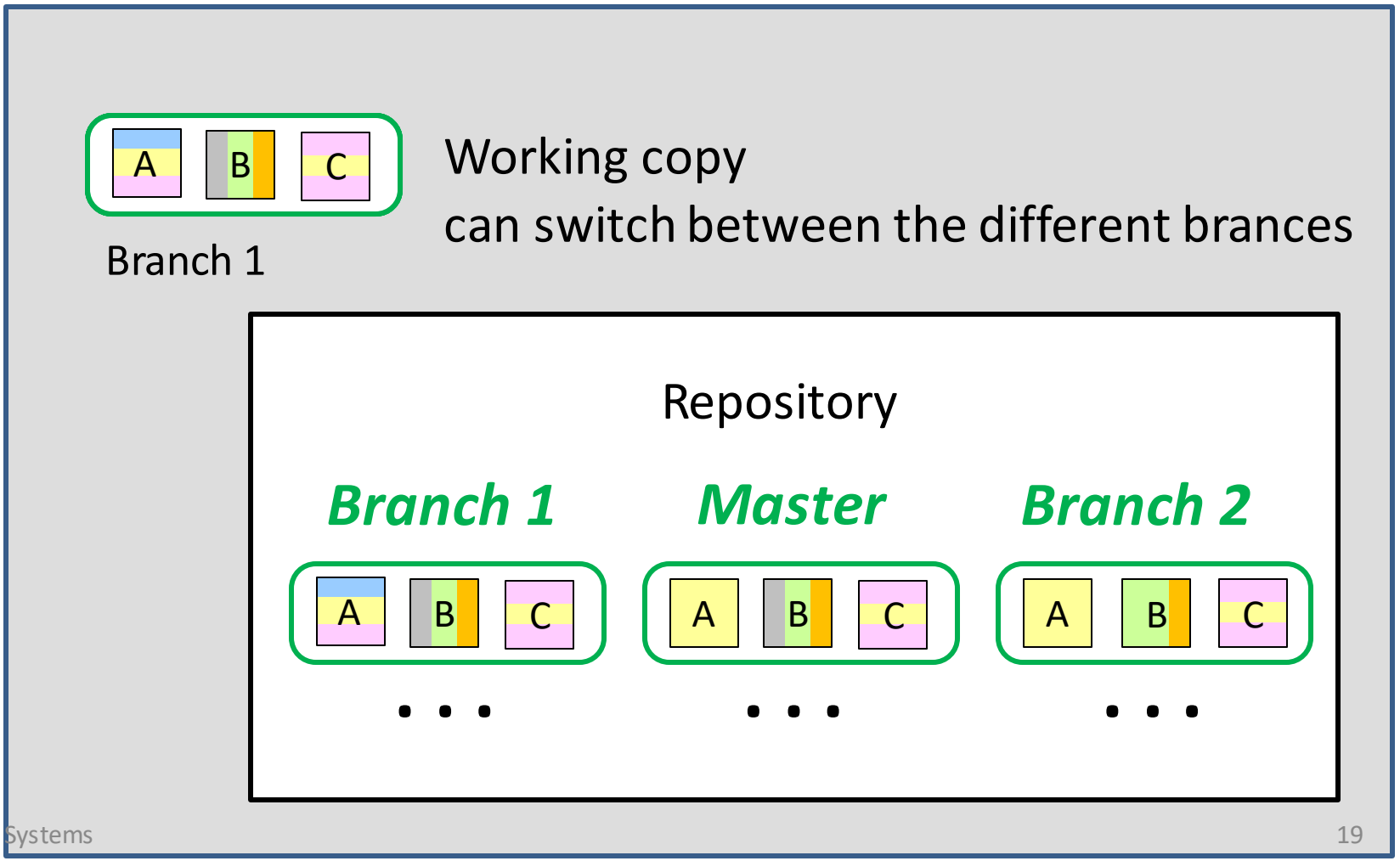
Branching



Branching



Branching


A
B
C
Branch 1
Working copy
can switch between the different brances
Repository
Branch 1
Master
Branch 2
A
B
C
A
B
C
A
B
C

Push/Pull vs. Branch/Merge

- **Push and Pull** propagates changes (new versions) **across different repositories**

- **Branch and Merge** allows working on **different branches within a repository**
  - By Push and Pull the branches with their versions will be propagated across the different repositories

Summary

- **Roll back**: undo a change in the past (while retaining later changes)
- **Push & Pull**: Synchronizing copies of the repository
  - Repositories can be merged automatically when changes are not conflicting
  - A file cannot be pushed in case of conflicting changes
- **Branching**: independent development in separate branches
  - For merging branches the same principles apply
  - It is a convention to call one branch ‘master’

Staging

- Eclipse integrates version control (Git)
- One function not mentioned in the slides above: **staging**, i.e., informing the repository about the existence of a file that is to be committed
- Standard sequence:

  stage → commit → push

  or

  stage → commit & push

Git

Distributed Version Control system
created by Linus Torvalds

- Various interfaces for interacting with Git
  - Command-line interface (Linux)
  - Various graphical interfaces
  - Directly from Eclipse

GitLab

- GitLab is a Git implementation made available by SNT (*Student Network Twente*) (git.snt.utwente.nl)
- You can login with your s-number credentials
- It is strongly advised to use GitLab for the Software Systems programming project
- It is mandatory to use GitLab for the project in module 4 (Data & Information)
- You will use it for many other projects as well…

Good practices

- **Commit often**
  - Commit coherent sets of changes
- **Use meaningful commit messages**
  - Something you can understand later on
- **Branch often**
  - Whenever you start a new extension or feature
- **Merge when you need to**
  - Merge often from master if you continue working on your branch
  - Merge to master when your branch is in a consistent state

Further information & tutorials

- Basic online tutorial
  - https://try.github.io/
- More about branching
  - http://learngitbranching.js.org/
- Extensive tutorial
  - https://www.codecademy.com/learn/learn-git
- Still didn't have enough? Watch this (Git for Ages 4 and Up):
  - https://www.youtube.com/embed/1ffBJ4sVUb4

Lab Session

- D-3.4: tutorial (command-line interface)
- D-3.6–D-3.9: using GIT from Eclipse

- If you get everything done now,
  then you can clear your backlog on Friday's session (or skip it if you have none)