

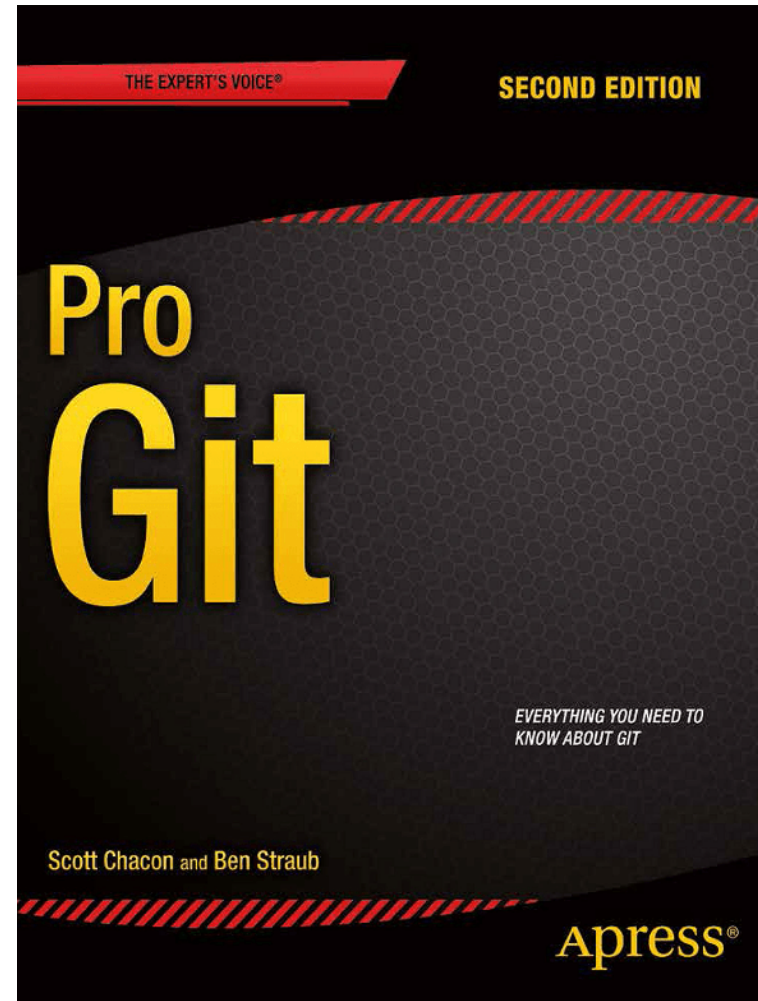
TDDE51 Lecture 2

git internals

August Ernstsson

Based on slides by Martin Sjölund & Mikael Asplund

Why Git?



Why do you need version control?

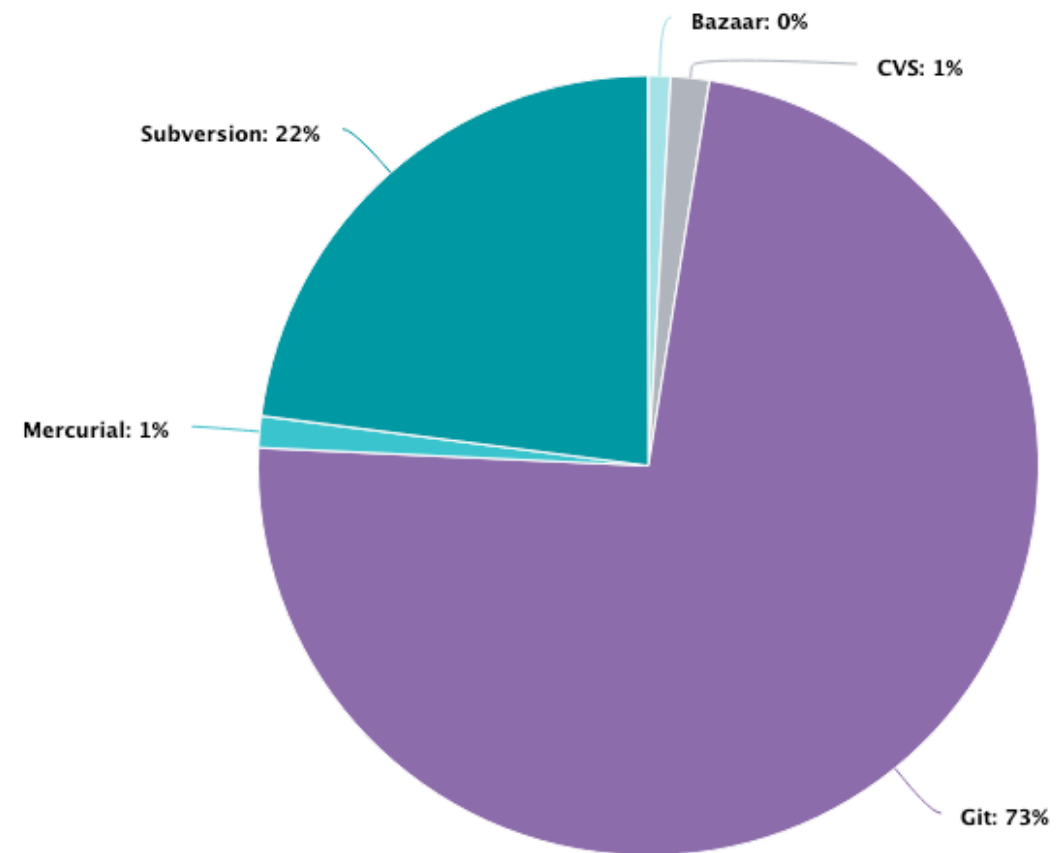
- Backup (go back in time)
 - Not only useful for source code(!)
- Maintain different versions (stable, development)
- Collaborate with others
 - Share your code
 - Let others suggest contributions
 - Work together
 - Manage project workflow

A bit of history

- 1973: Source Code Control System (SCCS)
 - Centralized, interleaved deltas
- 1982: Revision Control System (RCS)
 - Centralized, revision-based
- 1990: Concurrent Versions System (CVS)
 - Based on RCS, manages *projects* rather than individual files
- 2000: Subversion (SVN)
 - Designed to be mostly compatible with CVS. Transaction model, database backend.
- 2000-2005: Bitkeeper, Darcs, DVCs, Mercurial, Git,
 - Distributed version control
 - Flexible workflows



Compare Repositories



Source code management

- GitHub
- GitLab
 - LiU gitlab
- Bitbucket
- Azure DevOps
- ...

More on Git

- Created by Linus Torvalds in 2005
 - Inspired by BitKeeper and Monotone
 - Development started when BitKeeper stopped being available for free
- Non-linear development
 - Thousands of branches
- Fully distributed
 - But mostly not used as such
- Open source
- Designed to be fast and scalable
 - Mercurial did not have the performance or features needed for the Linux kernel
- Most operations are local
- Focus on data integrity (checksums)

Distributed state management

- Generic problem in distributed systems
 - Inherent conflict – **C**onsistency, **A**vailability, **P**artition tolerance (**CAP**)
- State-based propagation
 - Efficient
 - Conflict-prone (requires merging)
- Operation-based propagation
 - Better concurrency (but conflicts can remain)
 - Potentially slow



git

Git objects



- Blobs
 - Think of them as files

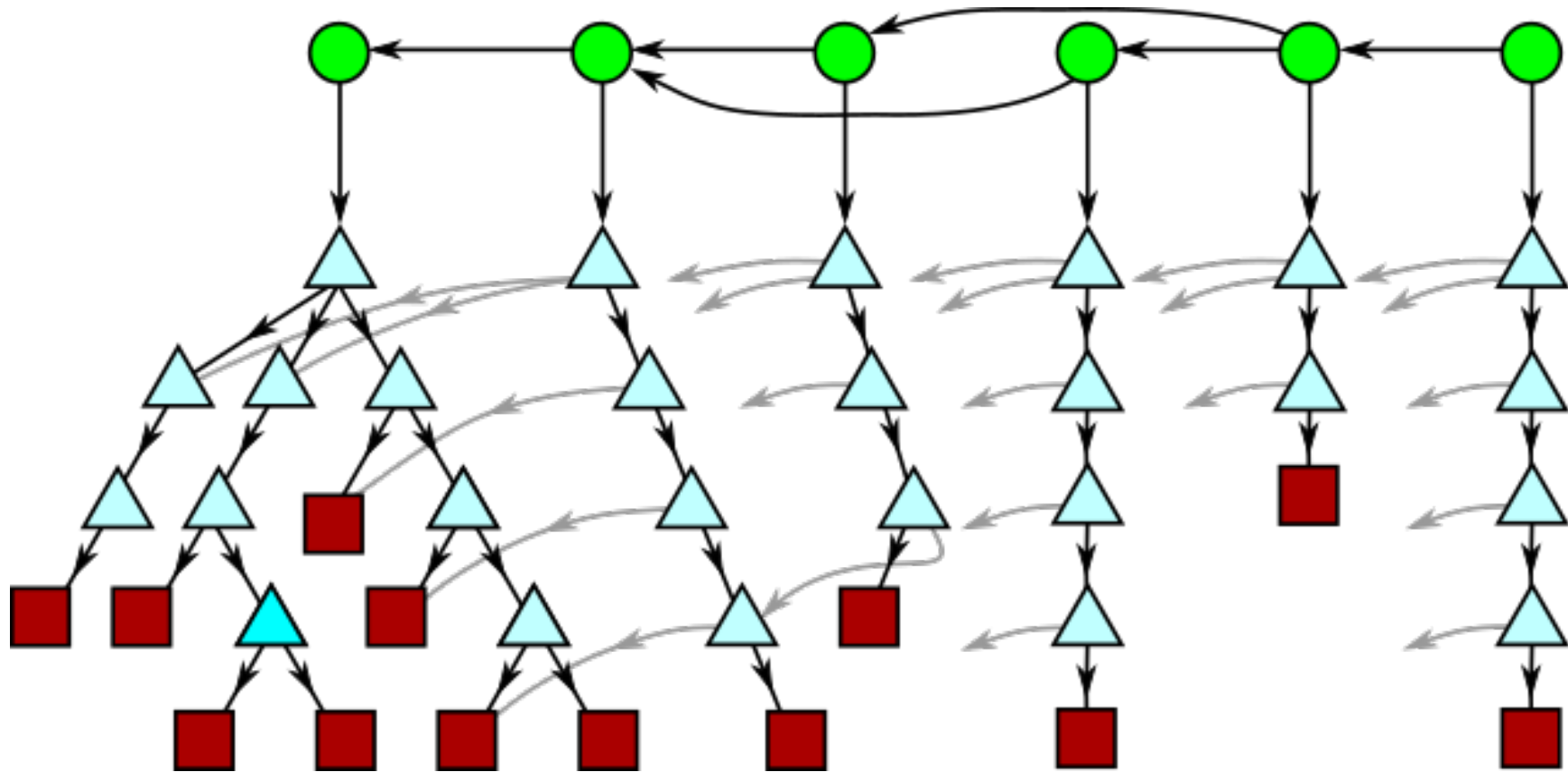


- Trees
 - Think of them as directories



- Commits
 - Think of them as versions

Object model



<https://stolee.dev/>

Blobs

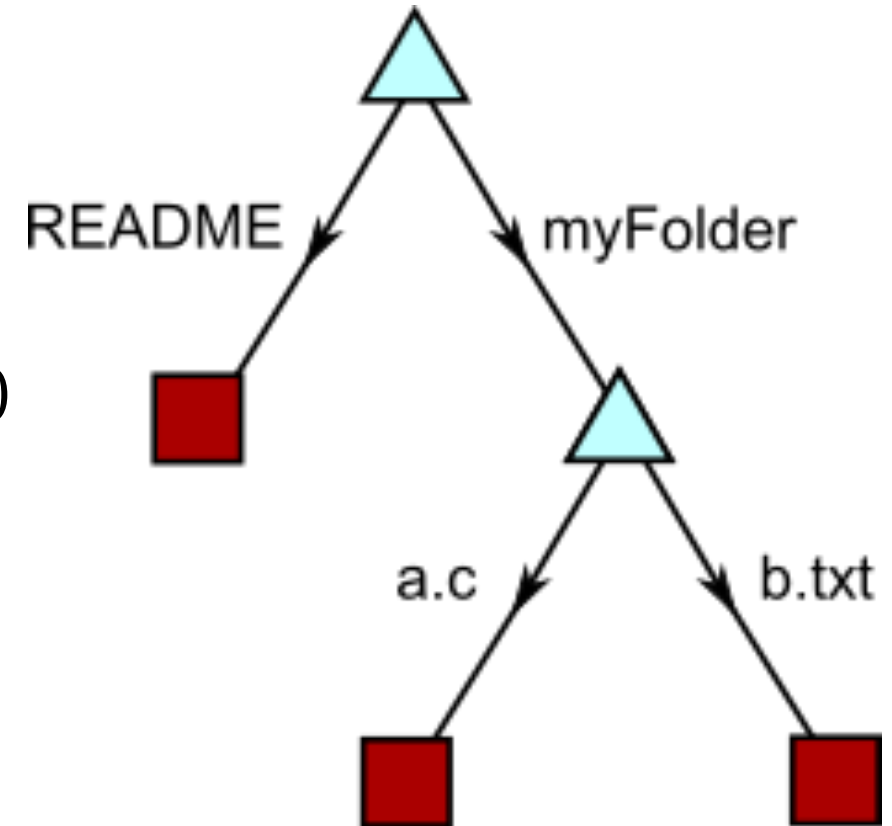


- Content of file (not name or metadata)
- Identified by hash
- Nice commands
 - `git hash-object (-w)`
 - `git cat-file -p`

Trees



- Names of files (blobs) and directories (trees)
- Permissions (mode)
- Changing contents create a new tree (new hash)
- Commands
 - `git cat-file -p` (again)
 - `git ls-tree`
 - `git mktree`



Example: Accessing git tree/blob

```
marsj@WIN01811:~/OpenModelica$ git ls-tree HEAD | head -n 4
040000 tree 14c745190454bb576b25421af2110179fdfa5706      .CI
040000 tree 1d04cfa0917f1fa56fa2c040e65dfcaa90e6f738      .externalToolBuilders
100644 blob 55061b584455324ff7432098a20e1e78cfd0dfc1      .gitattributes
040000 tree 428e8384158e8f90c8c9f2a08fc09f8dd3caeea4      .github

marsj@WIN01811:~/OpenModelica$ git ls-tree 428e8384158e8f90c8c9f2a08fc09f8dd3caeea4
040000 tree 7932e141f46f8ce07c9c1f1cd436dc9a6a77bdc6      ISSUE_TEMPLATE
100644 blob 0124c04e8a553dbf5c6b452cdd1c26d301ea1aa0      pull_request_template.md

marsj@WIN01811:~/OpenModelica$ git cat-file -p 0124c04e8a553dbf5c6b | head -1

### Related Issues
```

Pack files

- Conceptually, git objects are trees and blobs
- To send files over the network, files are packed (storing deltas of files)
- git repack can change your packs (on disk), and git push/fetch will send appropriate packs

```
~/OpenModelica$ for file in $(find .git/objects/pack -name '*.idx'); do  
git verify-pack -v "$file" | grep $(git hash-object Jenkinsfile) && echo  
"$file"; done
```

```
59f528edfcfef53bd81e679d527d3aad8265eba0 blob    32743 5112 195796  
6ad3ef9f1ad69252d31d29af4fb08278f13e337c blob    22 35 200908 1 \  
59f528edfcfef53bd81e679d527d3aad8265eba0  
.git/objects/pack/pack-2861aaeca53c0c2127f8f591665ddab77f774d43.idx  
SHA-1 type size size-in-packfile offset-in-packfile depth base-SHA-1
```

Pack file format

- Contains objects (commit, tree, tag, blob)
- Objects can be stored as is or as a delta of another object (in a chain if they are all in the same pack-file)
- Deltas are stored in the following way:
 - Instruction to copy from base object

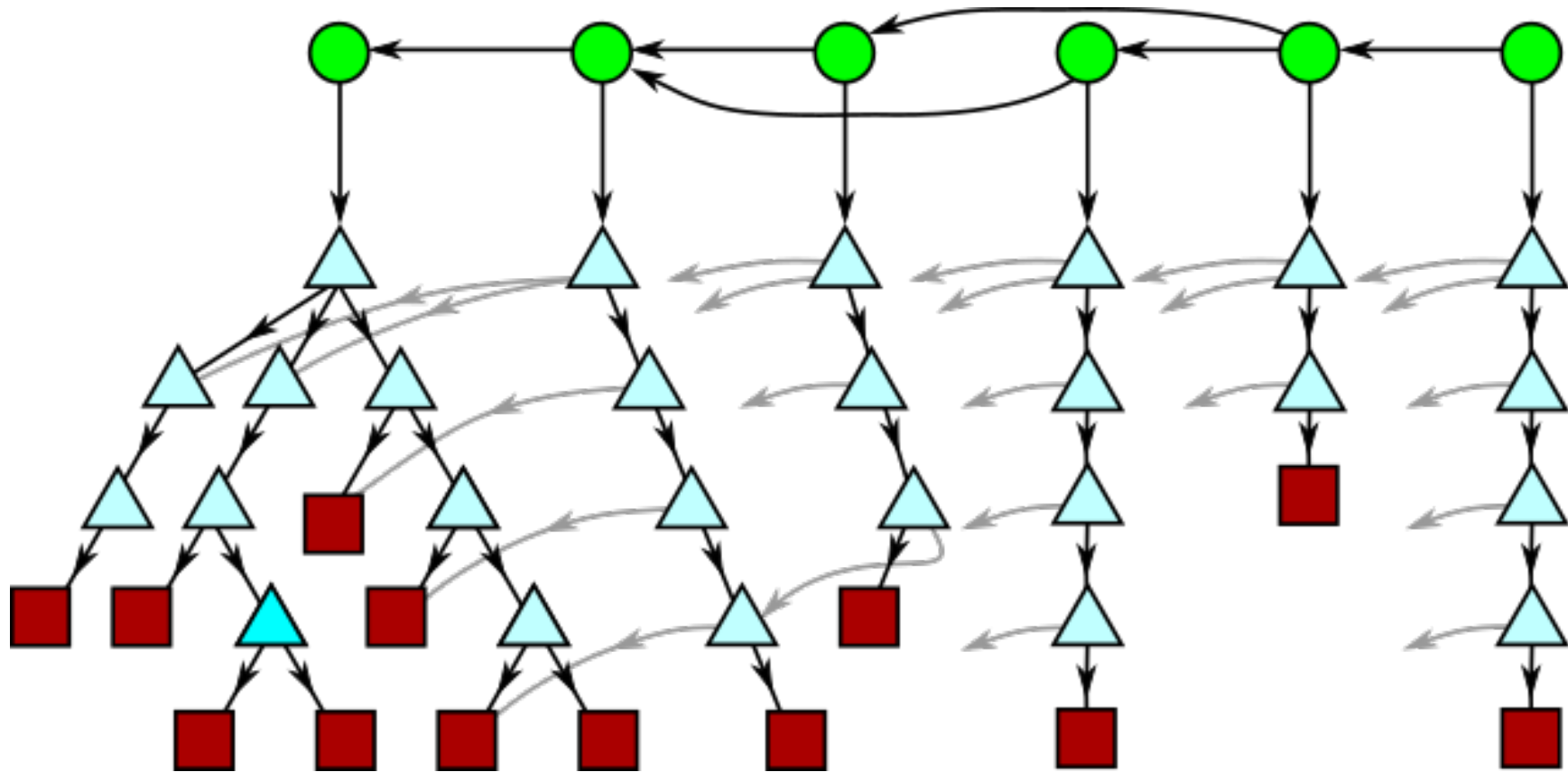
Byte 1 (1xxxxxxx): flags which offset (max 4)/size (max 3) bytes are used (to save size).
These bytes are copied from the base object to the new one.
 - Instruction to add new data

Byte 1 (0xxxxxxx): 7-bit data size=n. Next n bytes are copied to the object
- See <https://git-scm.com/docs/pack-format> for details

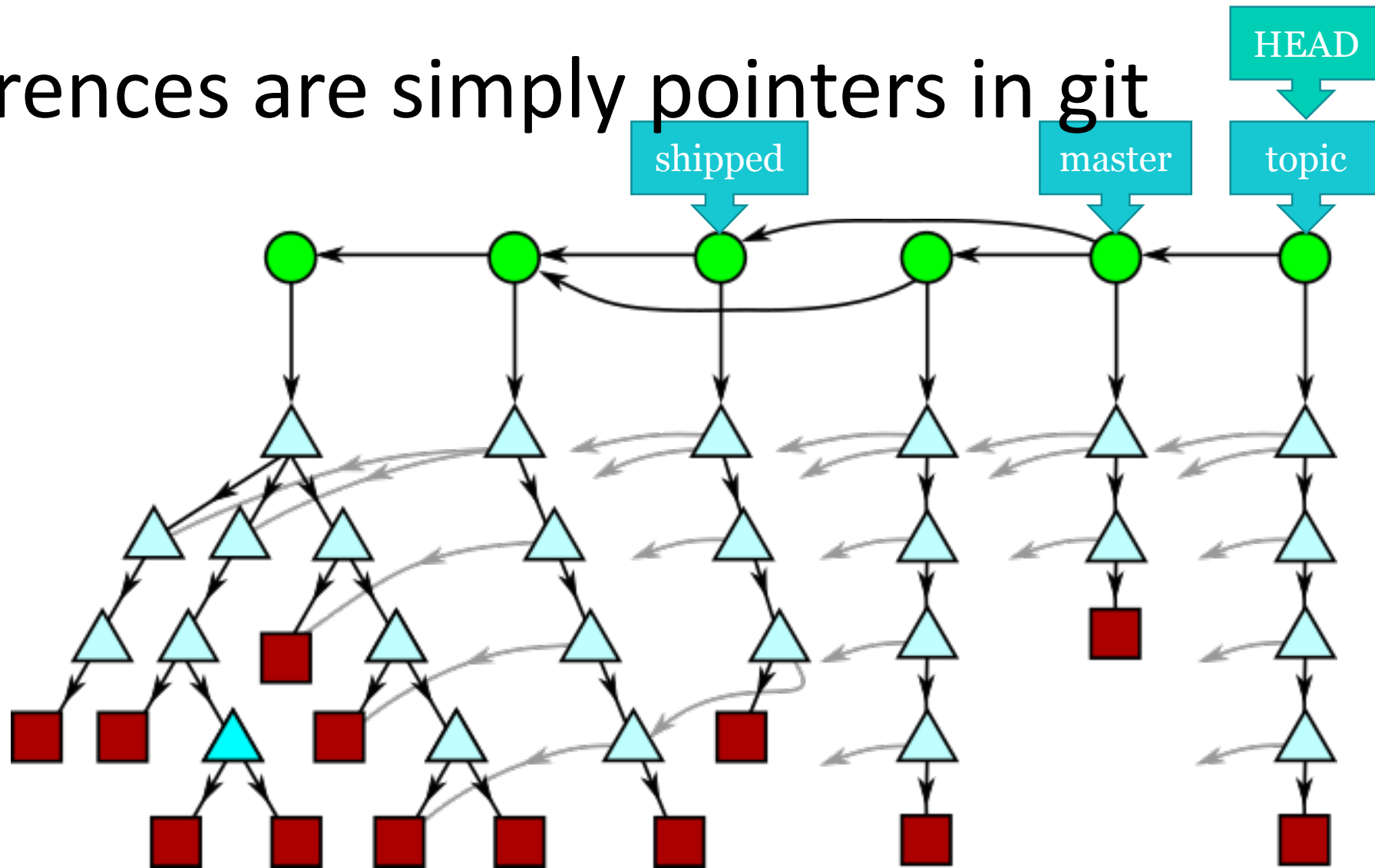
Commits

- A given version of the contents of the repository
- Contents
 - A tree
 - Commit message
 - Author name, email and time
 - Committer name, email and time
 - List of parents
- Commands
 - `git log --graph --oneline`
 - `git show` (also useful for other objects)

Object model



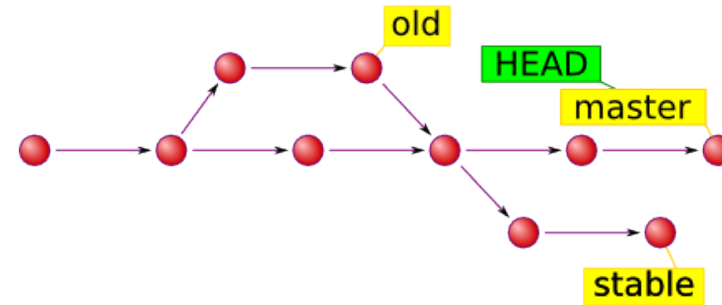
References are simply pointers in git



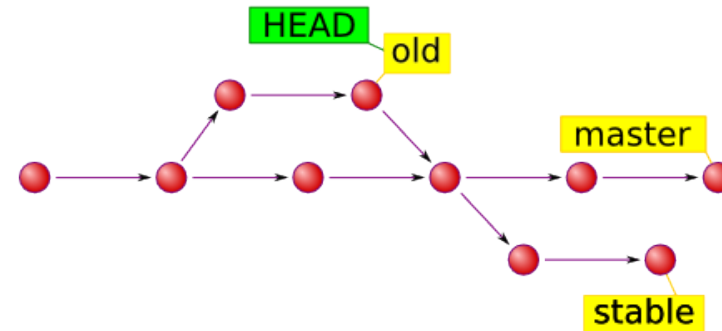
Git references

- Find them under `.git/refs/`
- Three types
 - Branches
 - Tags
 - HEAD (symbolic link, `.git/HEAD`)
- Commands
 - `git tag`
 - `git branch`
 - `git checkout` (`git switch`)

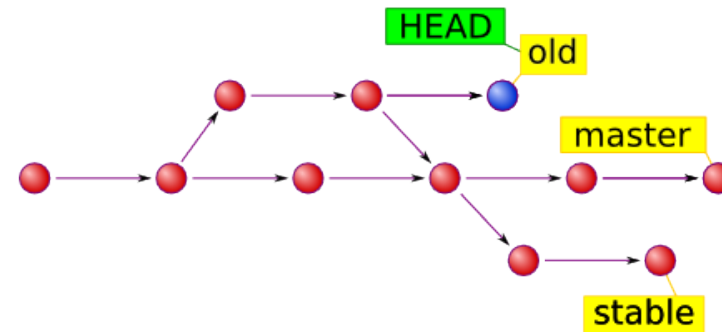
Branch example



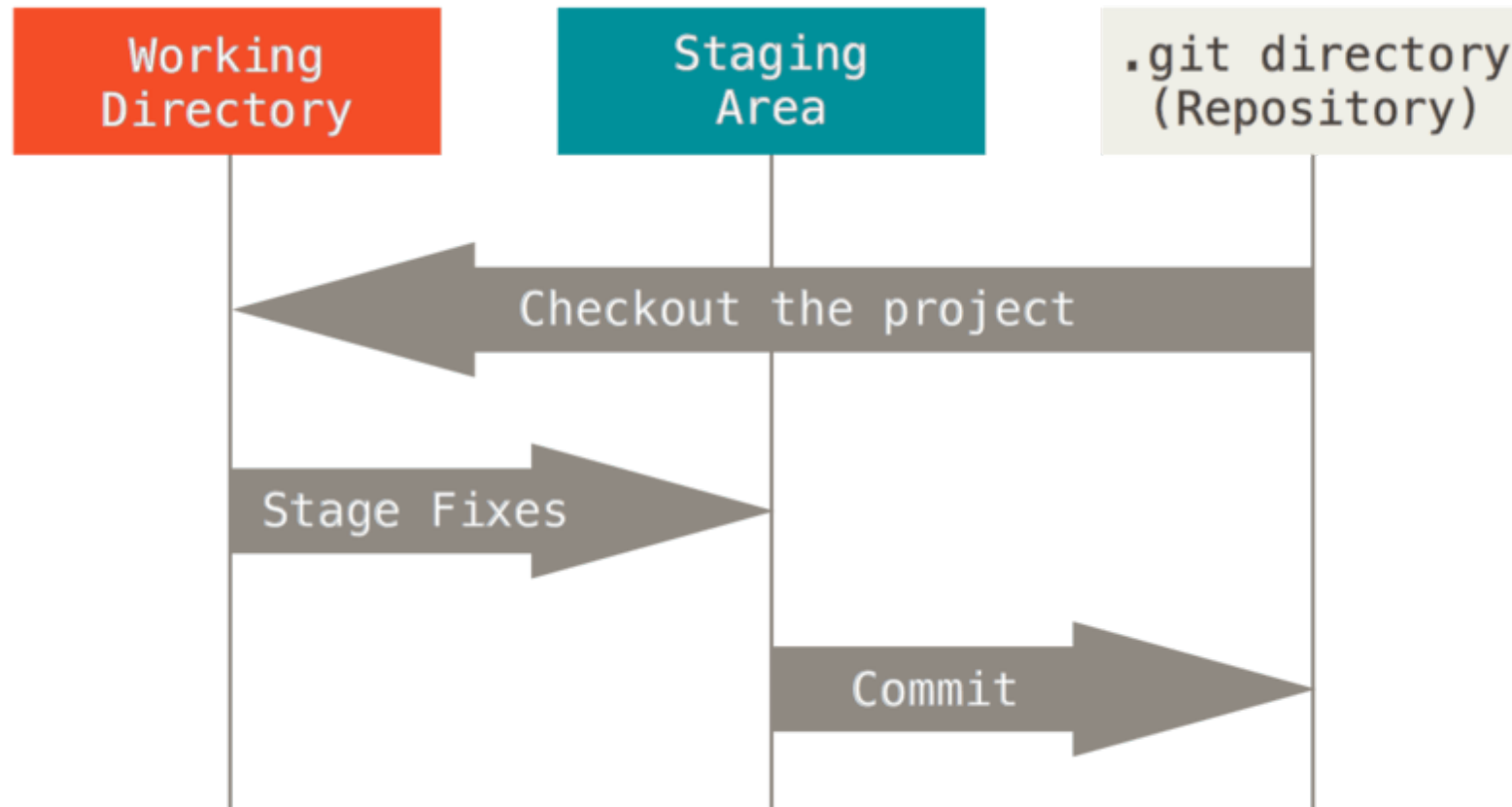
After "git checkout old":



After "git commit":

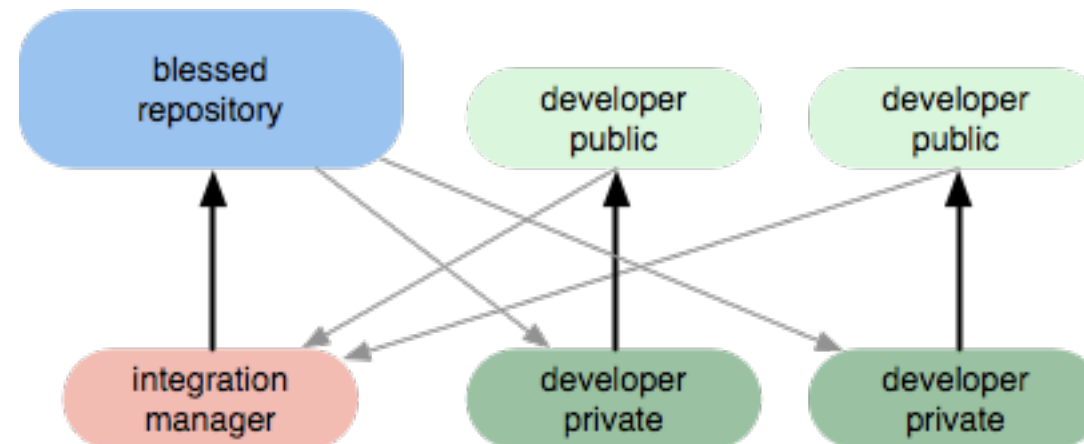


Staging



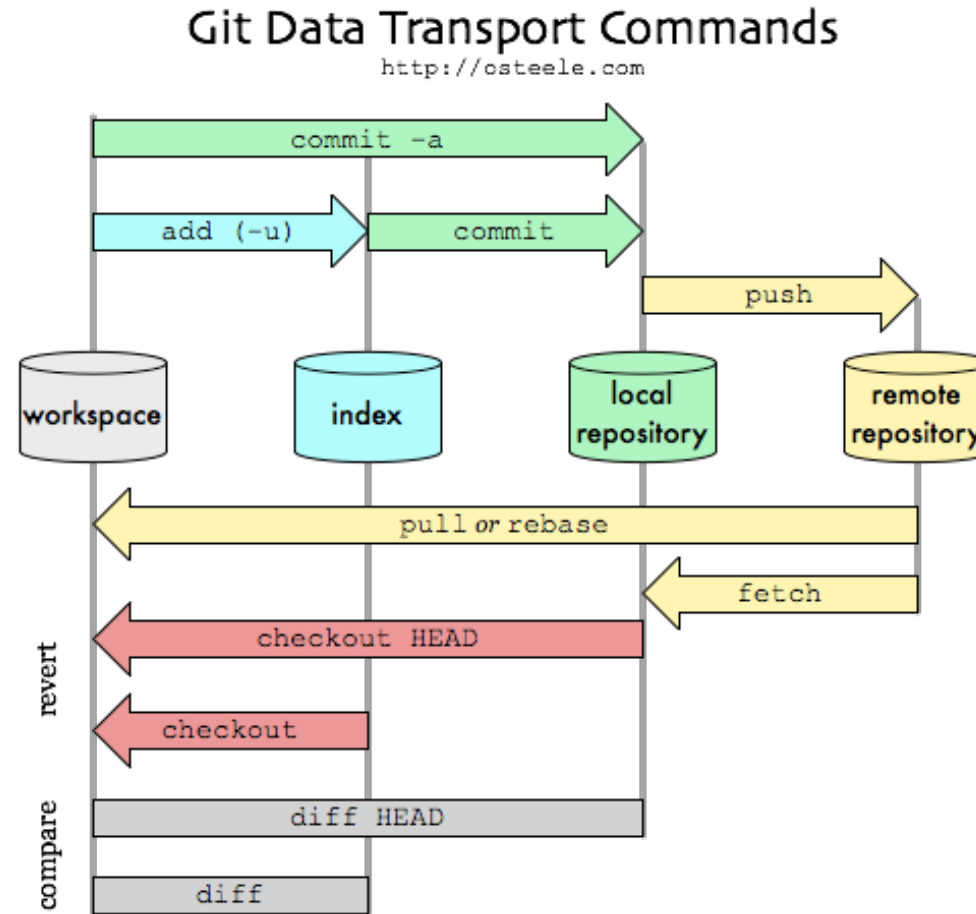
Working with others

”The GitHub model”



http://web.archive.org/web/20090210020404id_/http://whygitisbetterthanx.com/#the-staging-area

Working with remotes



Differences to Subversion (SCM, not distributed)

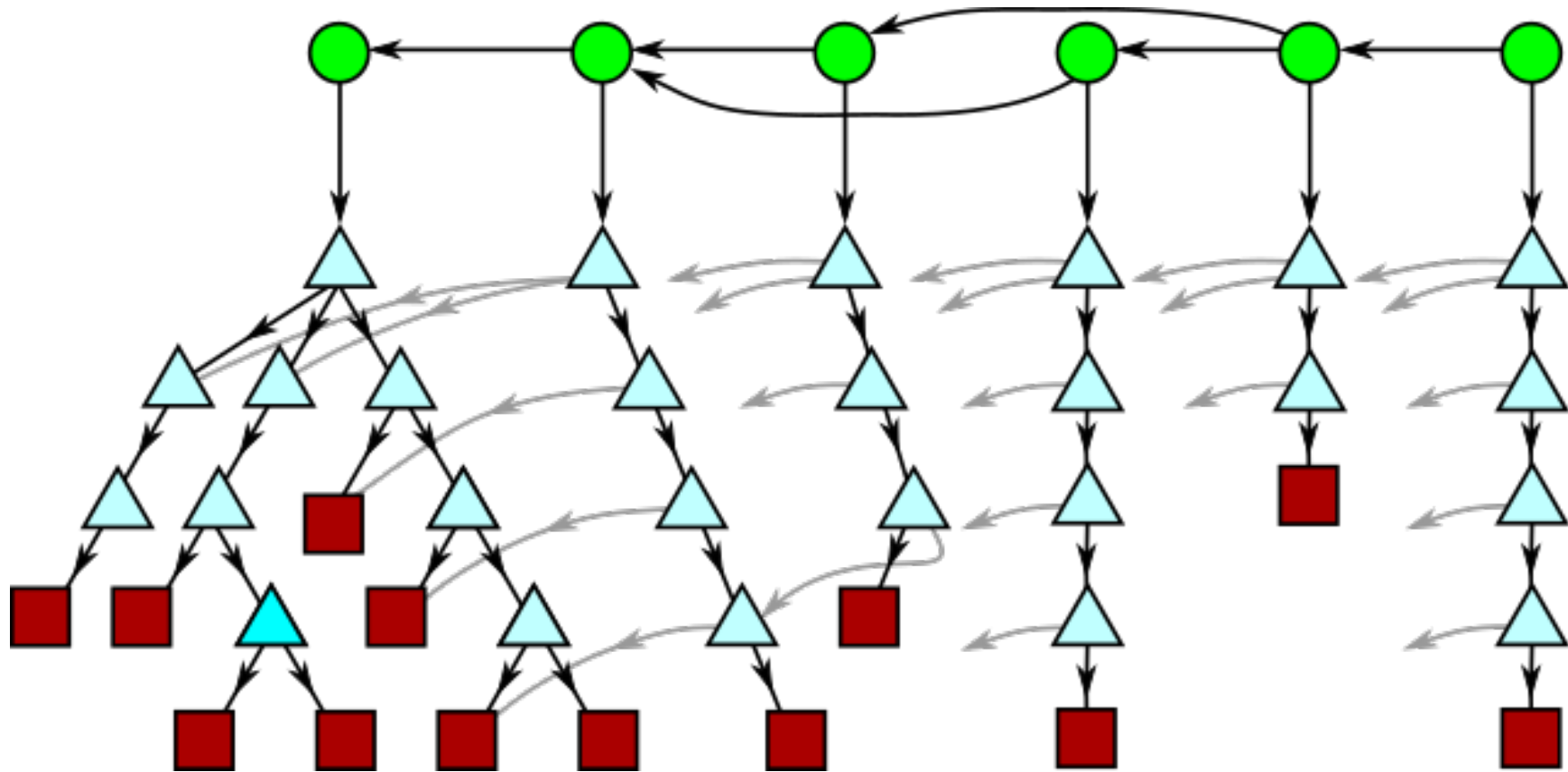
- svn commit – communicates with the server, is equivalent to git commit && git push
- Unable to create local commits, to save your work
- Unable to share your work with others before pushing to the server
- Branches do exist, but merging branches does not really work
- Need to query the server to see history (diffs) or logs (offline work is much harder)
- In Subversion you only have the HEAD commit available (can save space)

Working with a shared repo

- git fetch origin
 - Do this often
- Someone pushed changes before me!
 - git merge
 - git rebase (especially when working on smaller feature branches)
 - Do this often or the number of merge conflicts might be too many to handle
- git pull
 - git fetch & git merge in one step
 - To be avoided or change the default behaviour: git config --global pull.ff only
 - You can override this with git pull --rebase or by using git fetch && git merge

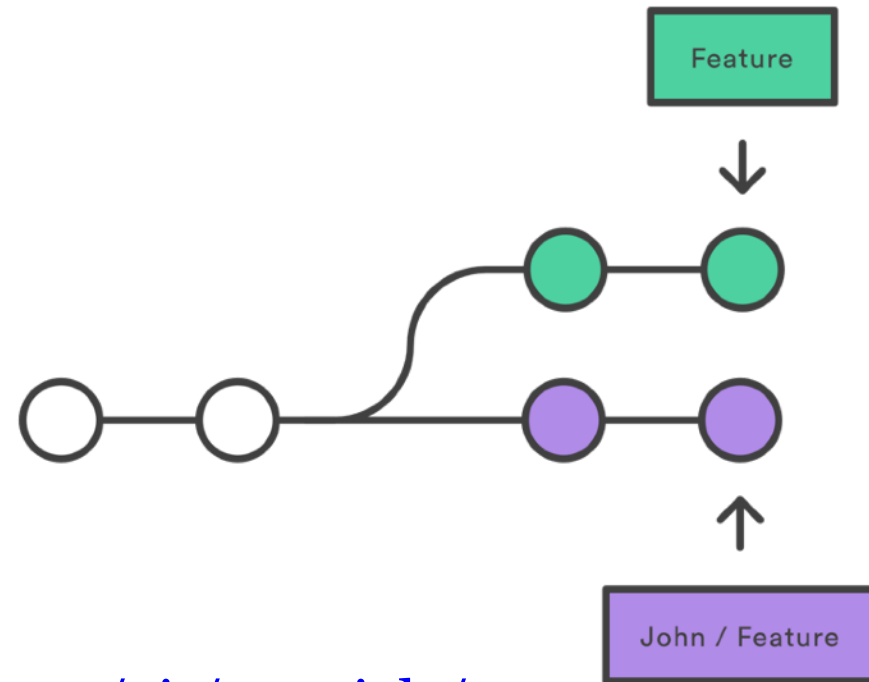
Why do we need to merge?

- Recall:



git merge or git rebase?

Collaborating on the same feature branch



<https://www.atlassian.com/git/tutorials/merging-vs-rebasing>

<https://creativecommons.org/licenses/by/2.5/au/>

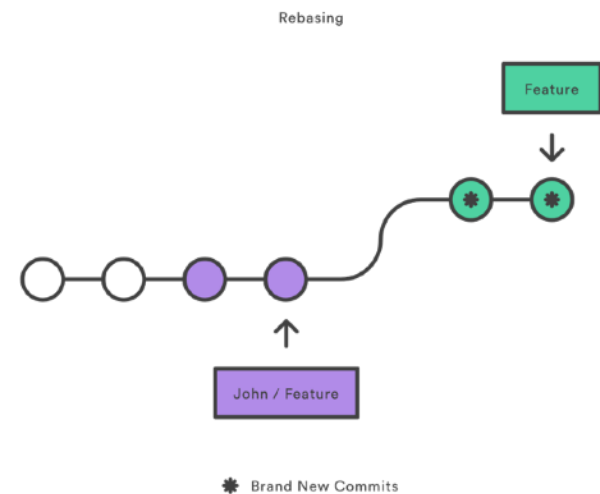
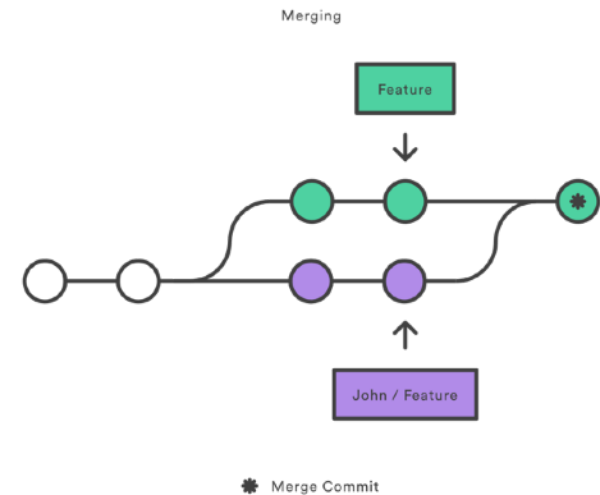
git merge or git rebase?

- git merge will create a commit with 2 parents
 - git supports octopus merges with even more parents
 - Multiple parents can be hard to visualize
 - Try git log in the terminal
 - Preserves the full history for feature and john/feature
 - Any other branches based on feature would still work
 - One merge commit each time you merge from the main branch (!)
- git rebase will destroy parts of the history
 - You must never use this on code in the blessed repository
 - The history is cleaner and easier to understand

Image is taken from:

<https://www.atlassian.com/git/tutorials/merging-vs-rebasing>

<https://creativecommons.org/licenses/by/2.5/au/>



Cleaning up your branch

```
commit be435f4b751eb010382c31140720f290b4fae78f (HEAD -> my-feature)
```

```
Author: Martin Sjölund <martin.sjolund@liu.se>
```

```
Date:   Wed Jan 25 12:32:35 2023 +0100
```

```
    Fixup feature X
```

```
commit 13d66eccdc083257b7696e735d5ac5769eb124d6
```

```
Author: Martin Sjölund <martin.sjolund@liu.se>
```

```
Date:   Wed Jan 25 12:32:17 2023 +0100
```

```
    Added another feature Y
```

```
commit 78d9b831ccfee5622639fd4ce0a066bcacf6905f3 (sjoelund/my-feature)
```

```
Author: Martin Sjölund <martin.sjolund@liu.se>
```

```
Date:   Wed Jan 25 11:58:57 2023 +0100
```

```
    Added new feature
```

be435 should be part of the first commit (78d)

The history will then look like we added a working feature X, and then feature Y

Interactive rebase (what git shows)

```
~/OpenModelica$ git rebase -i origin/master  
pick 78d9b831cc Added new feature  
pick 13d66eccdc Added another feature Y  
pick be435f4b75 Fixup feature X  
  
# Rebase e689e7ba19..be435f4b75 onto e689e7ba19 (3  
commands)
```

Interactive rebase (what you edit it into)

```
~/OpenModelica$ git rebase -i origin/master
```

```
pick 78d9b831cc Added new feature  
fixup be435f4b75 Fixup feature X  
pick 13d66eccdc Added another feature Y
```

We changed the order and set it to fixup
(merges into previous commit and does not
prompt for a new commit message)

```
# Rebase e689e7ba19..be435f4b75 onto e689e7ba19 (3  
commands)
```

Log after the merge

```
commit ece78ef5dd86ed067dc245e402b040647ea18d29 (HEAD -> my-feature)
```

```
Author: Martin Sjölund <martin.sjolund@liu.se>
```

```
Date:   Wed Jan 25 12:32:17 2023 +0100
```

```
    Added another feature Y
```

```
commit 28fb302c13b5c44d0af55da70da64995d1952aeb
```

```
Author: Martin Sjölund <martin.sjolund@liu.se>
```

```
Date:   Wed Jan 25 11:58:57 2023 +0100
```

```
    Added new feature
```

Both commit hashes are new

You can use this to

- Change commit messages
- Order of commits
- Remove commits (useful if you have a commit adding printf debug statements; removing the commit removes all statements)
- Squash 2 commits together

Use it on origin/master as target so you don't change public commits by accident

This rewrites the history!

Working with a tool like GitHub

- Others decide if your changes can be included
- Main mechanism: pull request
- You keep your own branch in your own fork of the repo or a branch in the public repo (if you are on the development team and that method is used)
- When done you send in a pull request which is then
 - Put through continuous integration (testing the code, CLA, etc)
 - Reviewed
 - Probably fixed by you
 - Accepted
 - Merged

My Github workflow

```
~/OpenModelica$ git checkout master
~/OpenModelica$ git pull --ff-only
~/OpenModelica$ git checkout -b my-feature
~/OpenModelica$ nano README.md
~/OpenModelica$ git commit -m "Added new feature" README.md
~/OpenModelica$ git remote add sjoelund git@github.com:sjoelund/OpenModelica.git
~/OpenModelica$ git push -u sjoelund
...
remote: Create a pull request for 'my-feature' on GitHub by visiting:
remote:      https://github.com/sjoelund/OpenModelica/pull/new/my-feature
```

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

 base repository: OpenModelica/OpenModelica base: master ... head repository: sjoelund/OpenModelica compare: my-feature

✓ **Able to merge.** These branches can be automatically merged.



Added new feature

Write Preview

H B I         

Related Issues

<!-- Link to the issues that are solved with this PR. -->

Purpose

<!-- Describe the problem or feature. -->

Approach

<!-- How does this address the problem? -->

Attach files by dragging & dropping, selecting or pasting them. 

☒ Allow edits by maintainers 

Create pull request

Reviewers

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Development 

Use [Closing keywords](#) in the description to automatically close issues

Helpful resources

[Contributing](#)

[GitHub Community Guidelines](#)

 1 commit

 1 file changed

 1 contributor

Commits on Jan 25, 2023

Added new feature

 **sjoelund** committed 1 minute ago

 78d9b83 

li.u LINKÖPING
UNIVERSITY

GitHub pull requests go beyond Git

 **Merged**

Use `libraries.openmodelica.org` mirror for package manager (#9704) #9712
casella merged 1 commit into `OpenModelica:maintenance/v1.20` from `sjoelund:1.20-gh-mirror` on Nov 17, 2022

old code.



sjoelund commented on Nov 17, 2022

Member Author 😊 ⋮

And this fixes the problem where you have a slow or failed github download and on next start, you have a 0-size file that cannot be unzipped and you have to manually remove.



casella commented on Nov 17, 2022

Member 😊 ⋮

@sjoelund you convinced me :)

 casella merged commit `b5c2f52` into `OpenModelica:maintenance/v1.20` on Nov 17, 2022

Hide details Revert

2 checks passed

✓  continuous-integration/jenkins/pr-merge This commit looks good

Details

✓  license/ccla Contributor License Agreement is signed.

Details

 sjoelund deleted the `1.20-gh-mirror` branch 2 months ago

Restore branch

<https://ida.liu.se/~TDDE51>