Advanced Git

Dr. Axel Kohlmeyer

Associate Dean for High-Performance Computing
Associate Director, ICMS
College of Science and Technology
Temple University, Philadelphia

http://sites.google.com/site/akohlmey/

a.kohlmeyer@temple.edu

A Few git Basics

- Git is a distributed source code management system: every “clone” has its own repository
- Development history is marked by a sequence of commits; a commit has one or two ancestors
- Each commit represents the complete state of the project and is identified by an SHA256 hash
- The commit hash includes the state but also the date, commit message, and hash of ancestor(s)
  → not possible to change a single “old” commit
  → have to “rewrite history” => diverging history

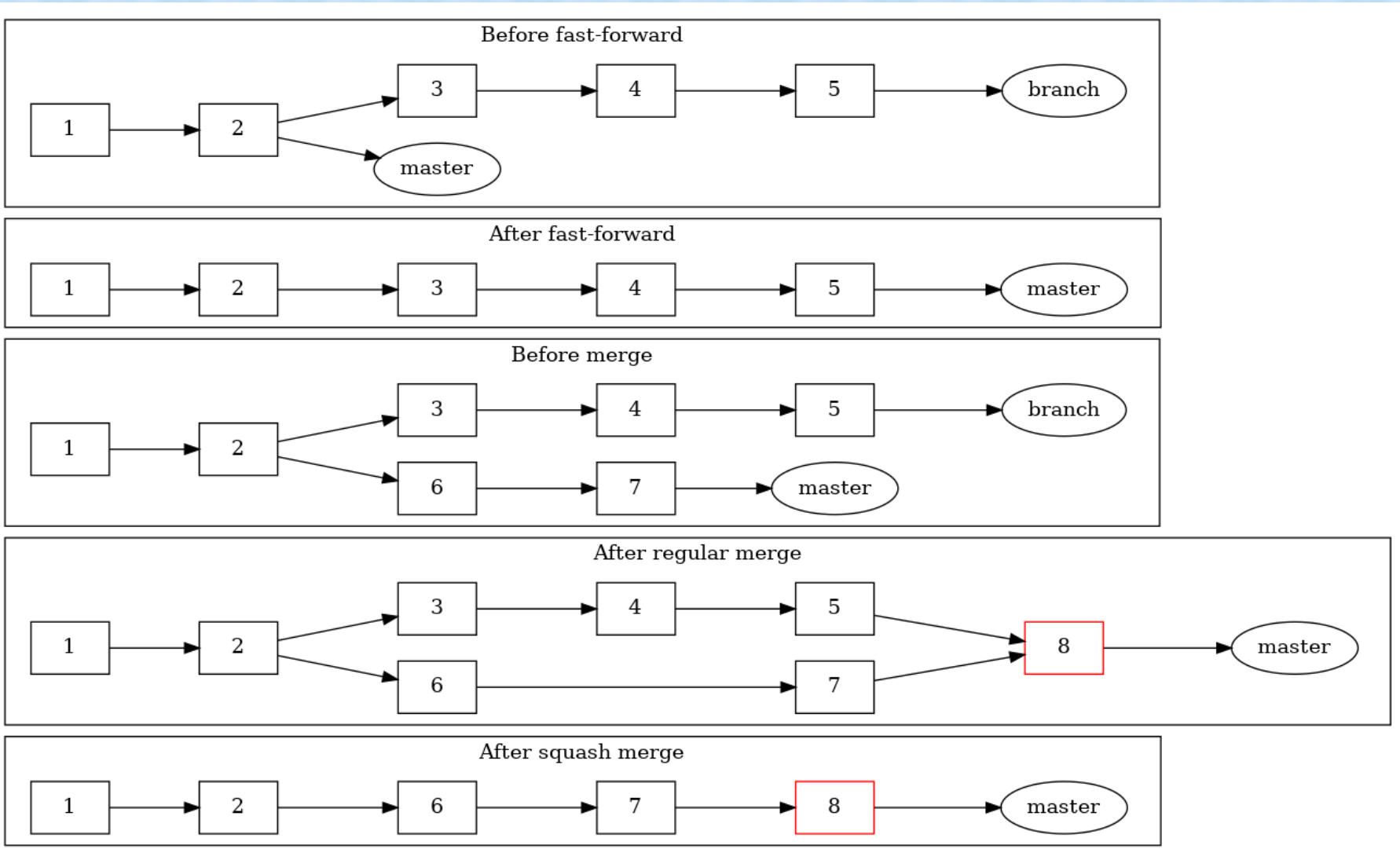
Branches vs. Commits

- Branches are identified by hash of the “head” all previous commits can be tracked down from the hashes of the respective ancestors
- Creating/deleting a branch is fast in git as it does not change anything in the repository
- When committing changes to a branch the info about the hash for the head revision is updated
- The repository is just a collection of commits similar to a filesystem with a snapshot capability

Merging Branches

- Three types of merges: 1) fast-forward merge, 2) regular merge and 3) squash merge
  1) Fast forward only possible when there were no commits to upstream → simply change head revision
  2) New commit is added with **two** ancestors allowing to track back all commits until where they were branched in case of conflicts, include resolution in merge commit
  3) Combine all changes from the branch (plus merge conflict resolution) into a single commit on upstream
     advantage: cleaner, simpler commit history
     disadvantage: information about development is lost

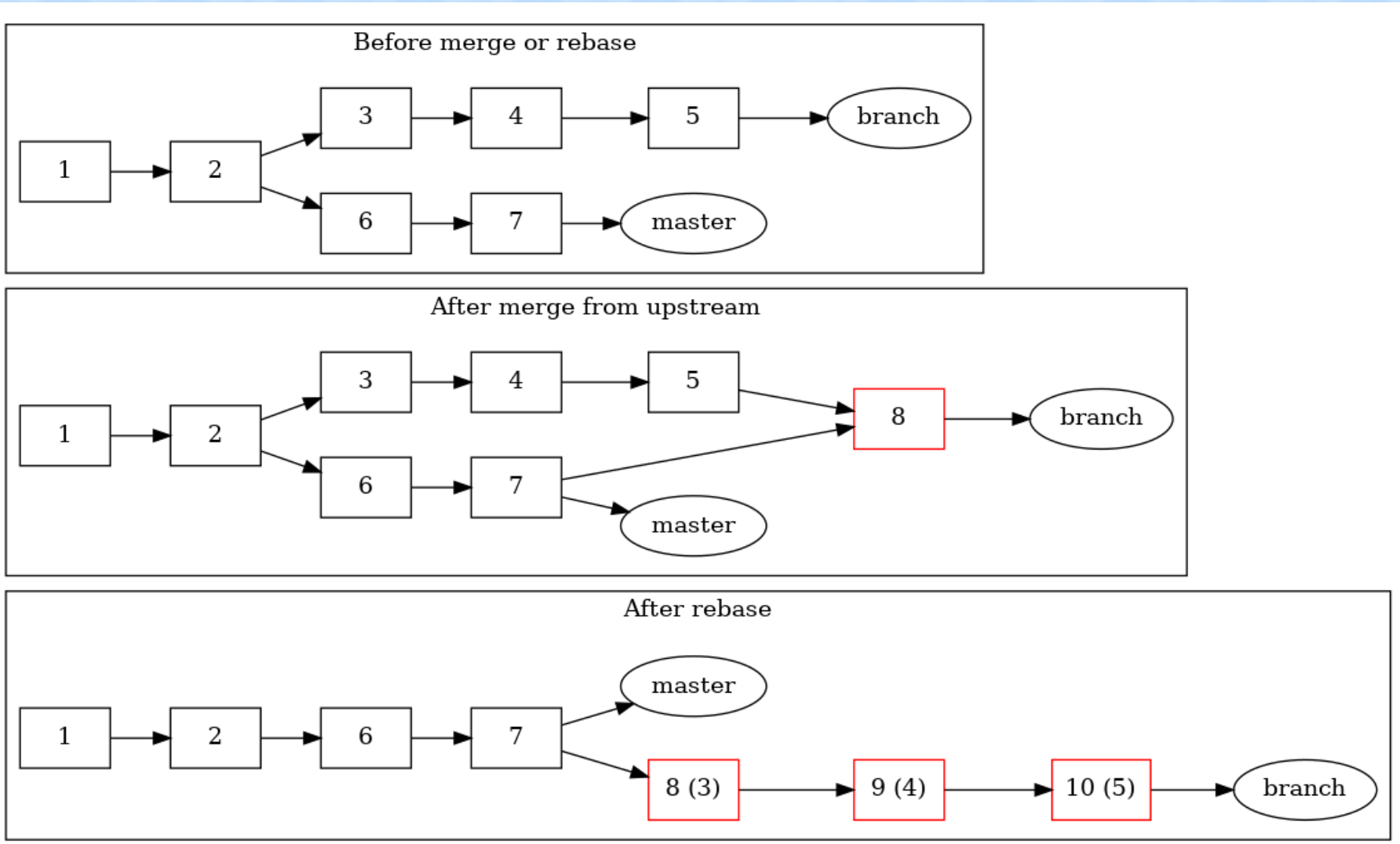
Merging Branches Visualization



Integrating Changes from Upstream: Merging versus Rebasing

- Merging – as shown before – can be used to integrate branches into upstream, but also to move changes from upstream into a branch
- Rebasing is a different way to integrate changes from upstream into a branch:
  1) All branch commits are stored as patches in reverse
  2) The head revision is fast-forwarded to the new base
  3) All patches are applied and committed again in order
  4) In case of conflicts the automatic procedure stops
  5) Can be done interactively to select patches to apply

Merging vs Rebasing Visualization



Pro/Contra Rebasing

- Produces a simpler and cleaner commit history
- Alternative to squash merging that preserves the history of changes but with the time stamps at which point they were added to upstream
- Rewrites history so the same changes will appear as two different commits
- Leads to diverging branches; local branch should be deleted if merged into upstream with rebase
- Best do rebasing for small local changes only. Use merging once pushed to a remote repository

To Branch or Not To Branch?

- Since branches in git have very little overhead, they can (and should!) be used **a lot**
- This is particularly true for short lived branches
- **Each** new feature or logical chunk of work should be done in a separate “feature branch”
- Even for complex projects there should be many short-lived feature branches than can then be subsumed into a long-lived branch before being merged into the main branch
- This helps organizing the project work and identifying the origin of problems efficiently through bisectioning

Working in a Team with git

- There are multiple strategies for how to work as a team on a project while using git for managing the project's source code
- Git provides capabilities, the team the policy
  - - One common upstream repository
    - All team members have write access to everything
    - Before pushing to a branch one needs to pull
  - - Use service like GitHub for management
    - Users "fork" from "canonical" repository
    - Additions only via pull requests and review
    - Main branch ("master") only updated by merges

Advanced Git

Dr. Axel Kohlmeyer

Associate Dean for High-Performance Computing
Associate Director, ICMS
College of Science and Technology
Temple University, Philadelphia

http://sites.google.com/site/akohlmey/

a.kohlmeyer@temple.edu