git Crash Course

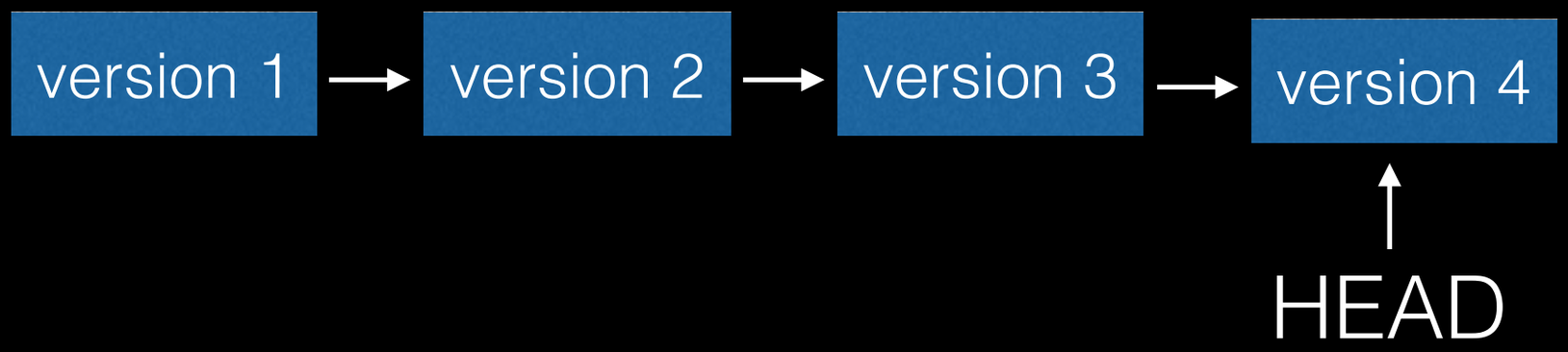
Andrew Whitaker
Figures from Scott Chacon

Motivation

- Imagine using the file system for a software project. What can go wrong?
  - Not easy to recover from mistakes.
  - Versioning must be done manually — e.g., foo_v1.py, foo_v2.py, foo_v3.py.
  - Coordinating across developers is hard.
  - Local file system is not backed up.

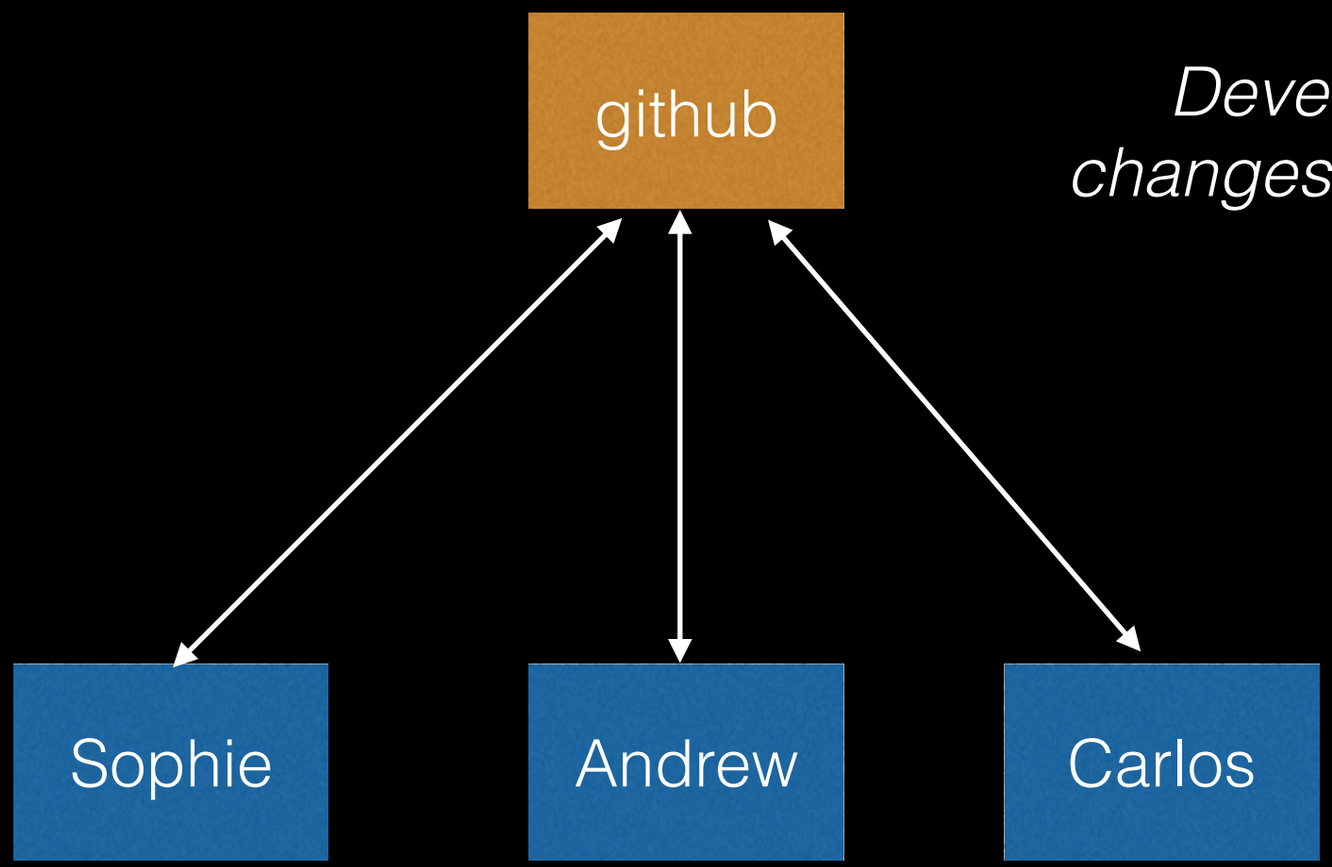
git Overview

- git maintains **all** versions of a software project.



- It is easy to view or rollback to a previous version.
    - In git parlance, we move the HEAD pointer.

Distributed git



Developers push and pull changes to/from an “origin” repo

Developers commit changes to their local repo

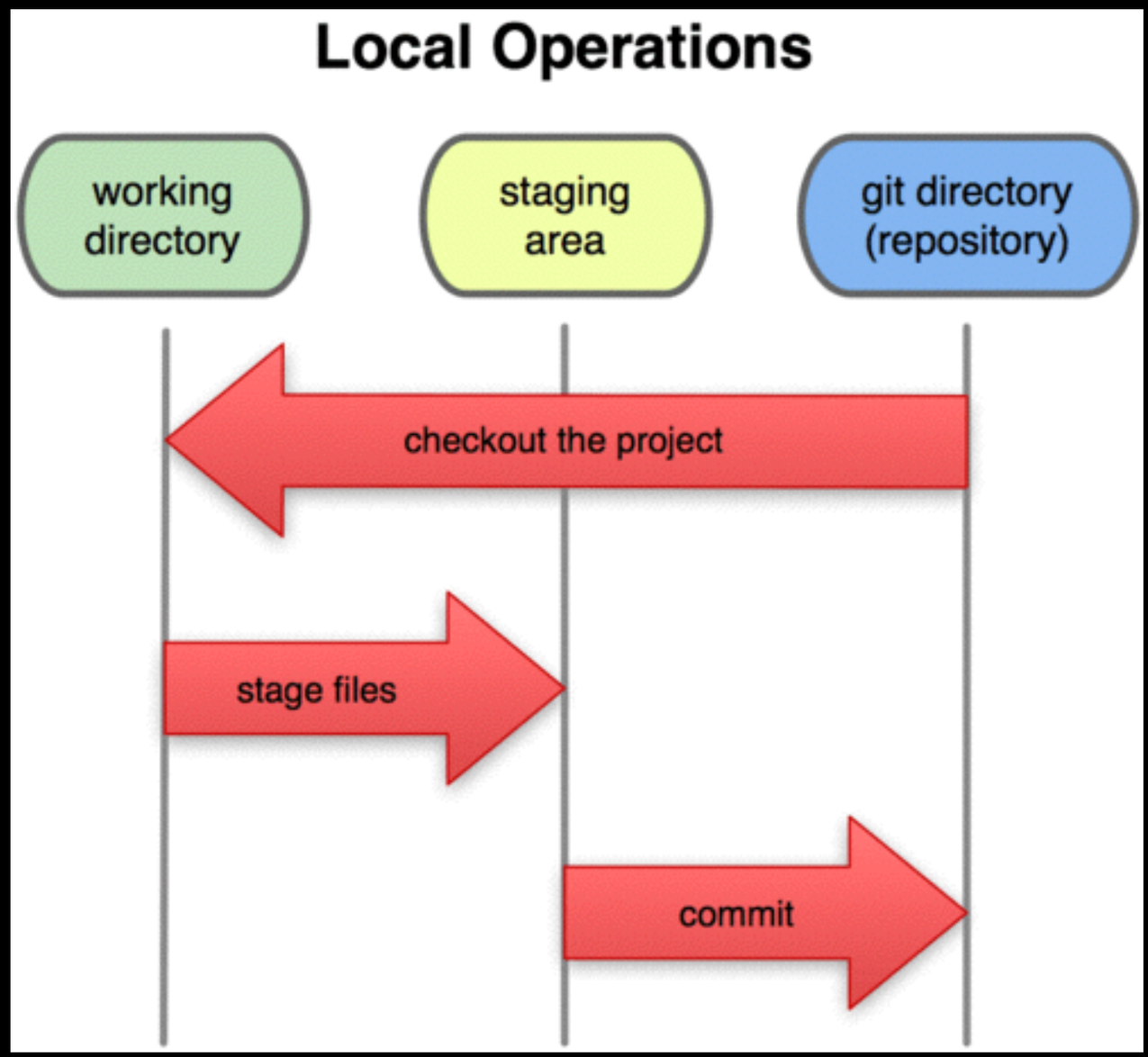
Why git{hub}?

- git coordinates actions across multiple developers.
- git maintains all previous versions
  - It's nearly impossible to lose committed changes.
- github maintains backups "in the cloud"
- Other github niceties: code reviews, wikis, issue tracking, unit testing, …

Demo

The Three (Local) File Locations

- Working directory: Your copy of the files
- Staging area: files that have been staged for commit (via git add)
- git directory: Files that have been committed (via git commit)

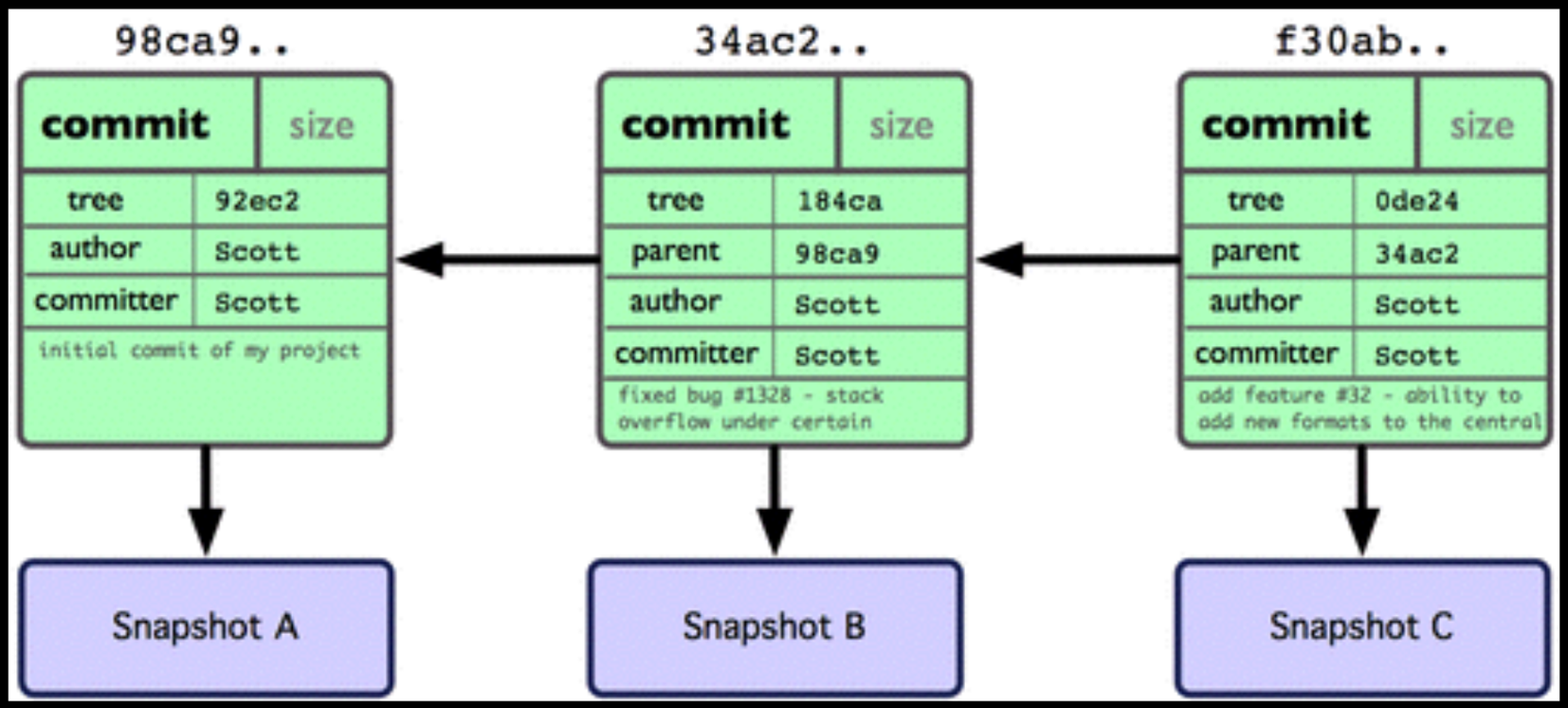


git Commands

- git checkout: Set your working directory to a particular revision
- git add: Stage a file for commit
- git commit: Commit a file to the (local) repository
- git commit -a: Stage and commit in one step
  - Only works for already added files
- git log: Show recent commits
- git status: Show status of file states

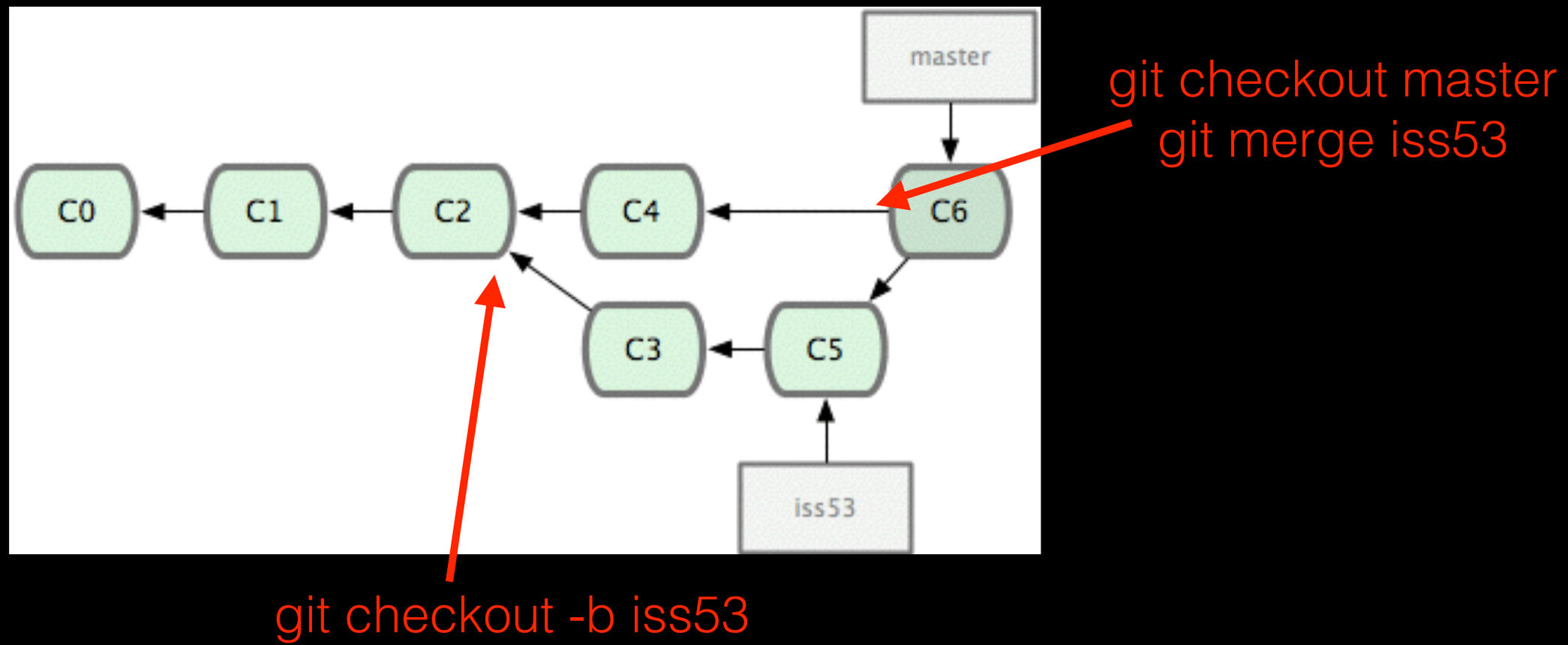
git Internals

- git maintains a directed-acyclic graph of file system snapshots

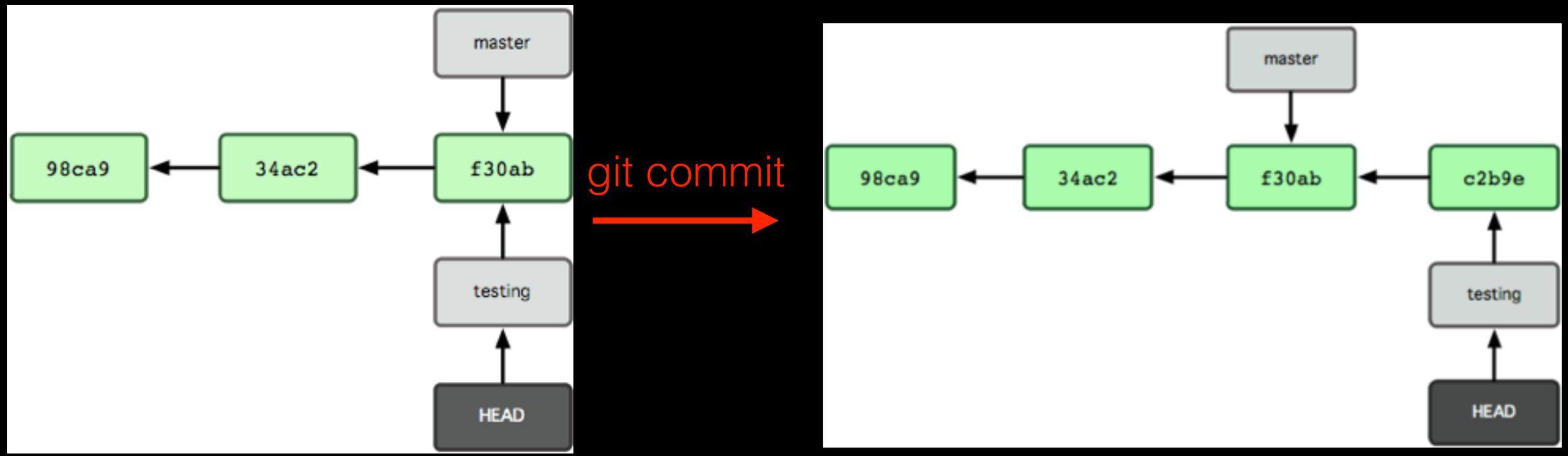


Branches

- Git history is a graph: nodes can have multiple children / parents
- A branch is simply a pointer into the DAG

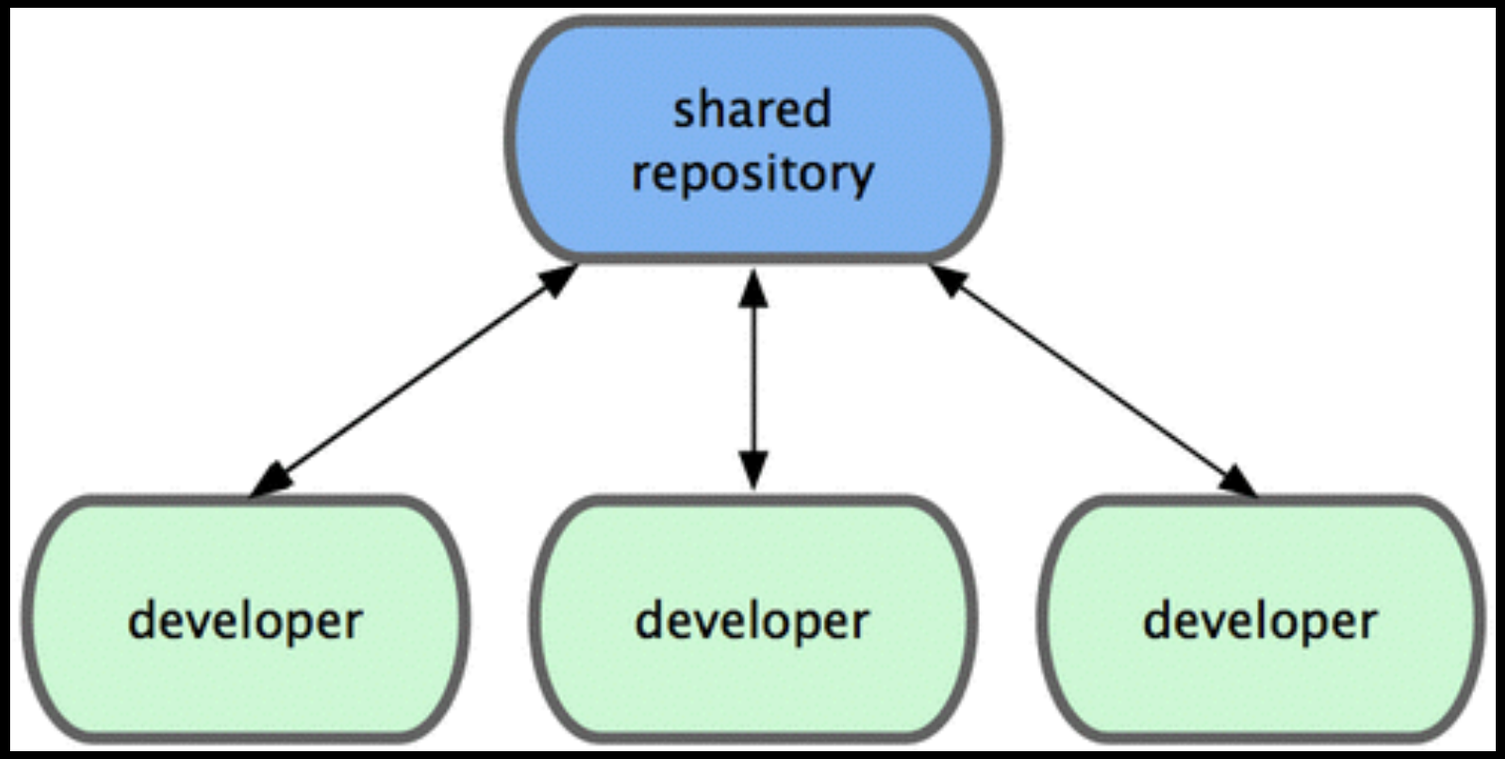


Commit Behavior



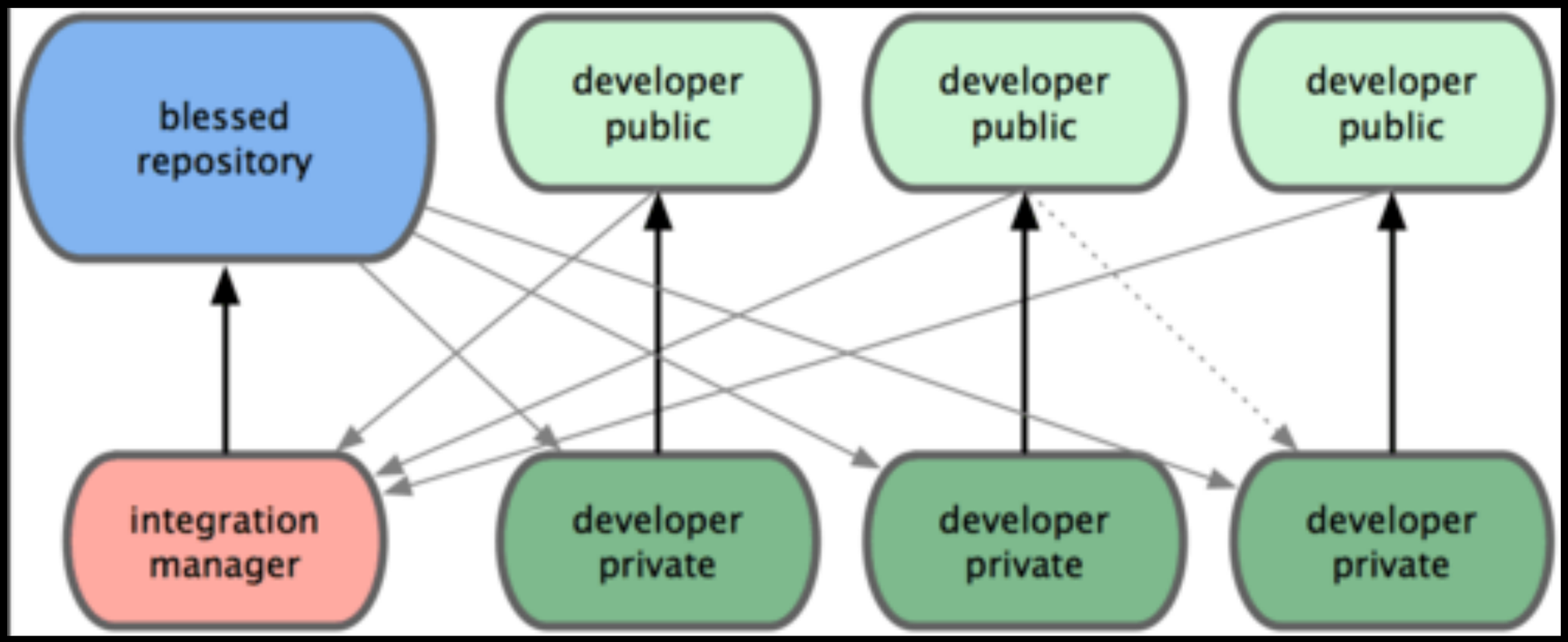
- HEAD is a special “you are here” pointer.
- On a commit, the HEAD branch advances, but no others.

git Distributed Workflow #1



Centralized Workflow

Distributed Workflow #2



Integration Manger

git Remotes

- A remote is a repository on a remote machine:
  - git@github.com:7andrew7/FantasyBaseball.git
- Tracking branch: A local branch that has a direct relationship with a remote branch.
  - git push: push local commits to the remote branch.
  - git pull: fetch and merge changes from remote branch.
- Prefer fetch + merge to pull; for details: http://longair.net/blog/2009/04/16/git-fetch-and-merge/

Command Quick Reference

- git status: information dump of local files; super-useful.
- git log: list recent changes.
- git add: stage a file for commit.
- git commit: commit a file to the local repo.
- git push: push changes to the default remote repo/branch.
- git pull: fetch and merge changes from default remote repo/branch. Equivalent to: git fetch followed by git merge.

Best Practices

- Use git for managing everything.
- Make lots of small self-contained commits.
- Don’t commit broken stuff to master branch.
  - See: git stash, private branches
- Write a good commit message.
- Use feature branches to group multiple commits.