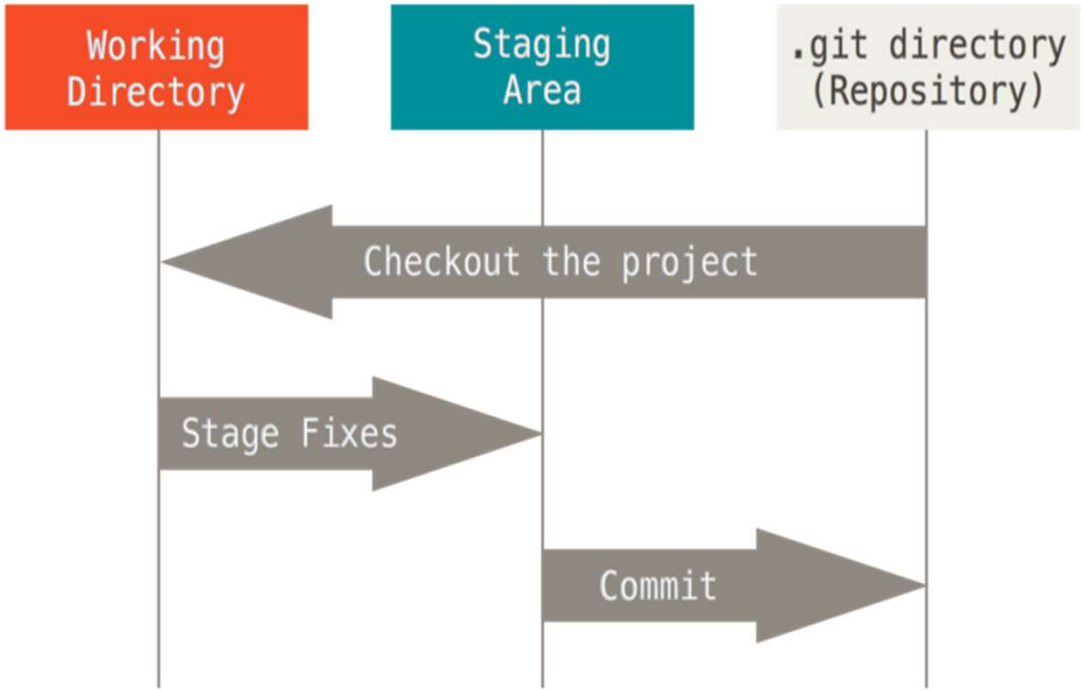
Git Basics

- Check out some code
  - Get a copy of the files
    - Could be from your machine
    - Could be from someone else's
- Make your changes
- Stage the changes you want to commit
  - Which changes you want to track
- Commit your staged changes
  - Like hitting "save"
- Share your changes

Git States

- Files can exist in three main states
  - Modified
    - File changed but not committed
  - Staged
    - Modified and marked to be committed
  - Committed
    - Safely stored in database



git-scm.org

Git Commands

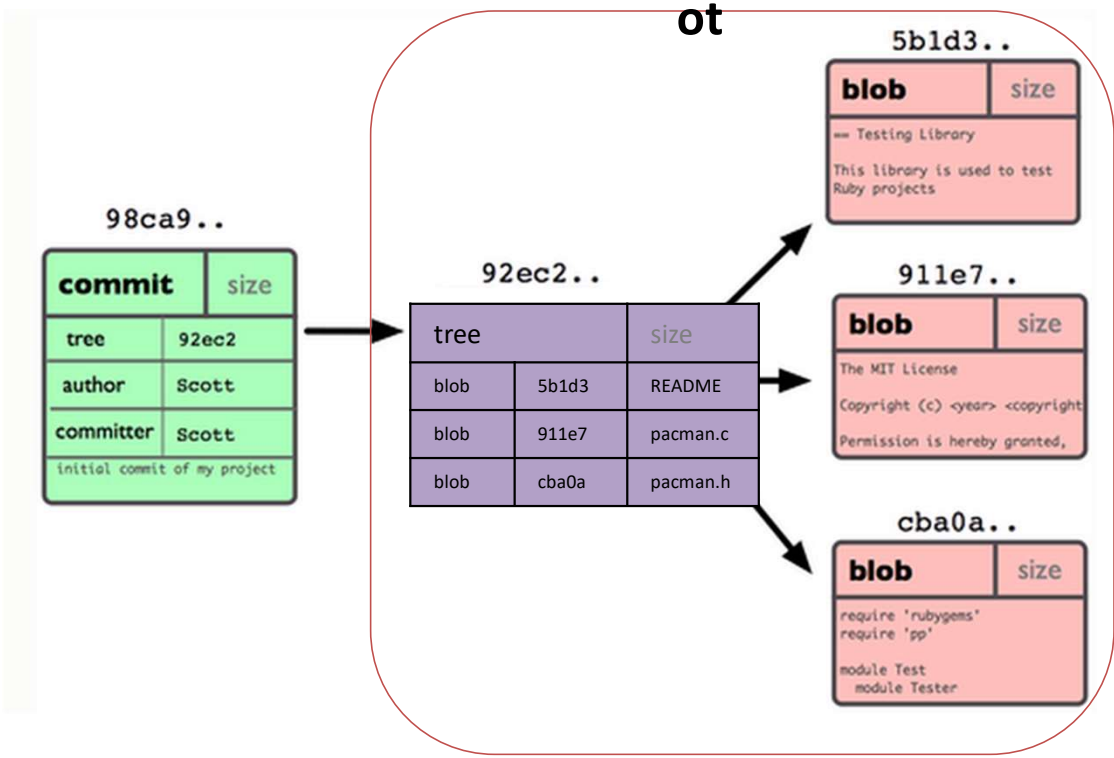
- Repository creation
  - $ git init (Create a new repository)
  - $ git clone (Create a copy of an existing repo)
- Branching
  - $ git branch <new_branch_name>
  - $ git checkout <tag/commit> -b <new_branch_name>
- Commits
  - $ git add (Stage modified/new/deleted files)
  - $ git commit (Save changes to repository)

Git Commands

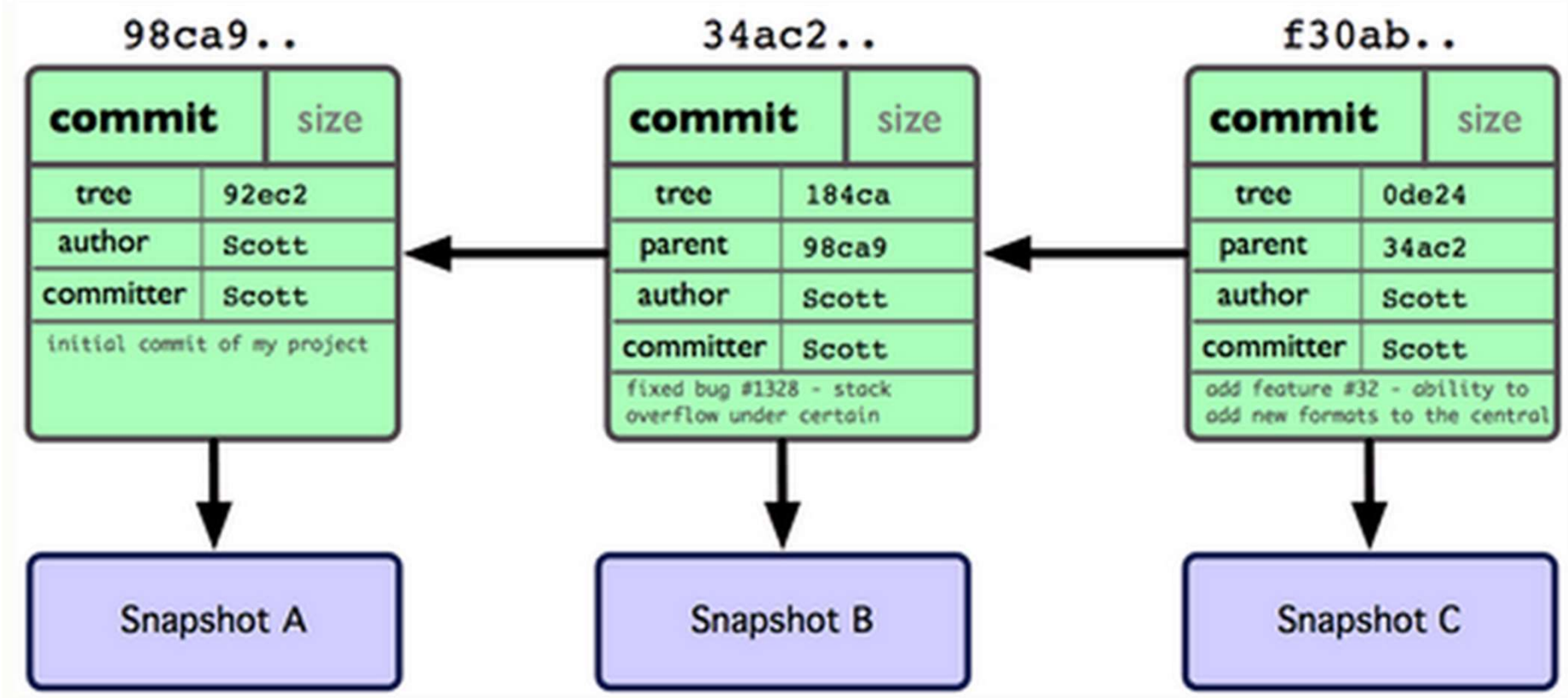
- Getting information
  - $ git status (Shows state of modified files, new files, etc.)
  - $ git diff (Compare different versions of files)
  - $ git log (Shows history of commits)
  - $ git show (Shows object in the repository)
- Help
  - $ git help

Git Repo Structure

- A commit corresponds to a snapshot
- Snapshot is a picture of your repo at the time you commit
  - If file is unchanged since last snapshot, just point at its last version
- Tree
  - Think a “collection of files”
- Blob
  - A version of a file
- Checksum
  - Run SHA1 on object to get an identifier
  - This is how git refers to the object

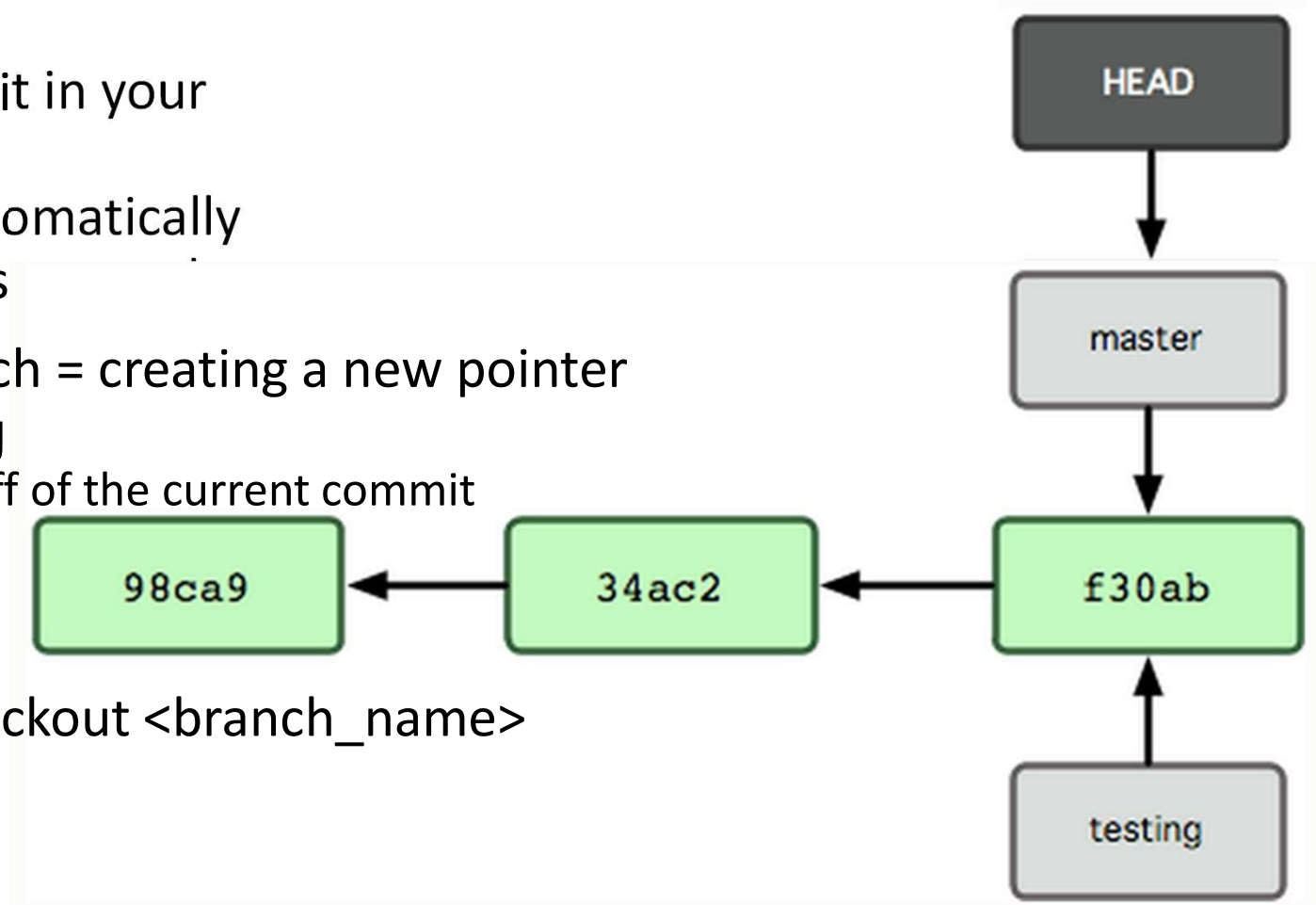


After Two More Commits



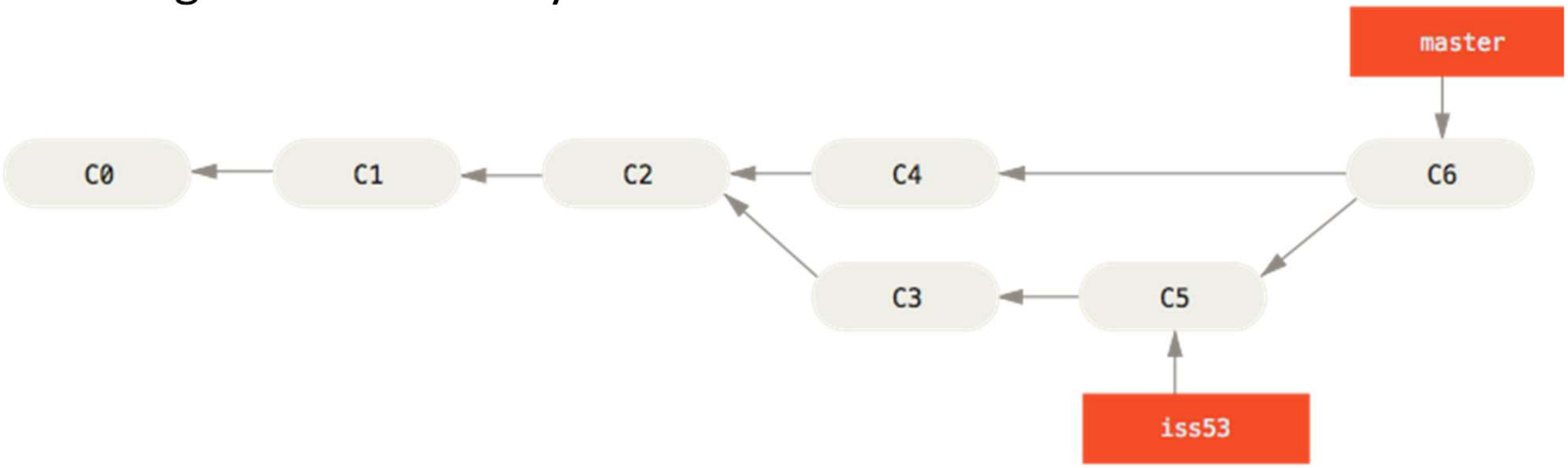
What is a branch?

- A pointer to a commit in your repo + its history
- “Master” branch automatically created when repo is
- Creating a new branch = creating a new pointer
- $ git branch testing
  - Creates a branch off of the current commit
    - Aka HEAD
- Switch with $ git checkout <branch_name>



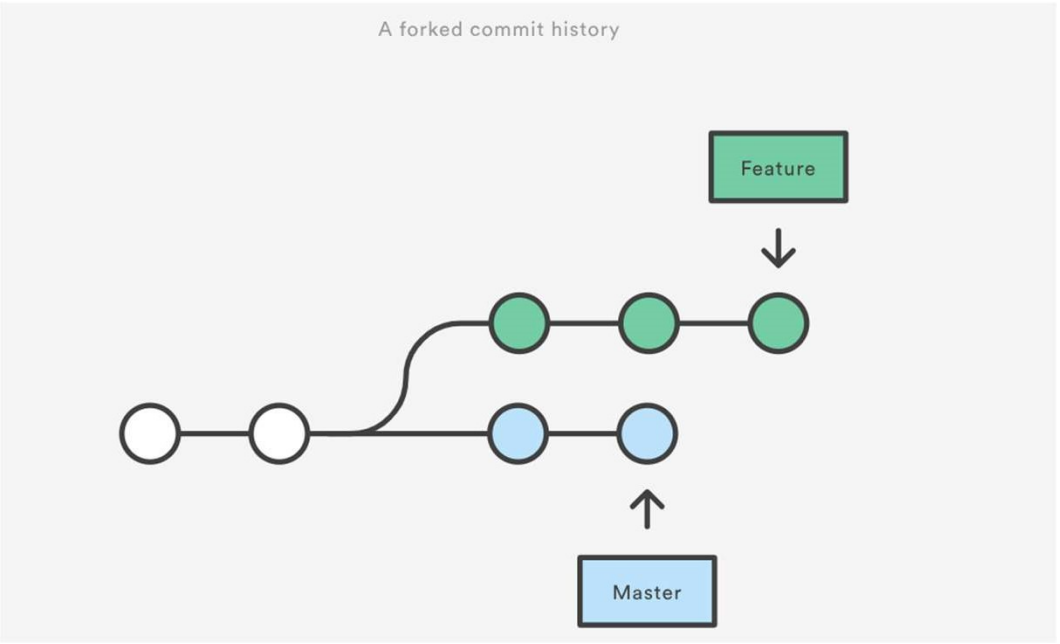
Git Merge

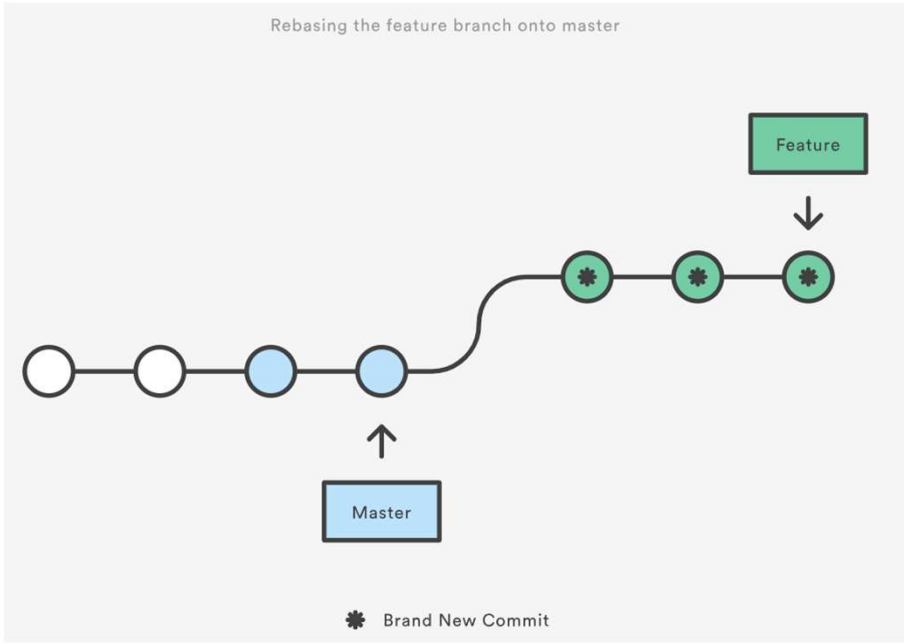
- Puts a merge commit in your history
- Created via 3 way merge between common ancestor and snapshots C4 and C5
- Con: Merge commits everywhere



Git Rebase

- Moves the “base” the branch you’re integrating with from the common ancestor to HEAD
- Cleaner, but you rewrite history





Remote Repositories

- You might want to interact with a repository elsewhere
  - On the local network
  - On GitHub
- Good for collaboration
- A bit of an offsite backup
- Git clone automatically adds an “origin” repository
- $ git remote
  - List info about remotes
  - Lets you manage them
    - Add
    - Remove
  - $ git remote show origin
    - Shows info about origin

Remote Branches

- Branches that correspond to remote branches
- Git helps you keep them in sync
- Take the form <remote>/<branch>
  - E.g. origin/master
- Can make new branches that track remote as well
  - $git checkout -b <branch> <remote/branch>
  - $git checkout --track <remote>/<branch?

Working With Remotes

- $git fetch <remote>
  - Pull all the info about <remote> to your local
    - Doesn't do any merging
  - <remote> is origin if not specified
- $git pull <remote> <branch>
  - $git pull origin master
  - Grabs changes from the remote, and merges them into **current** branch
    - Like git fetch followed by git merge
- You've made changes locally, how do you send them to the remote?
  - $ git push
- Sends all your changes to the remote repository
  - If there is a conflict, will alert you
  - Need to resolve conflicts locally, then reattempt push
- Push might be restricted
  - Only certain users can push to a repo/branch

More Git Commands

- Reverting
  - $ git checkout HEAD main.cpp
    - Gets the HEAD revision for the working copy
  - $ git checkout -- main.cpp
    - Reverts the changes in the working copy
  - $ git revert
    - Reverts a commit with a new commit
- Cleaning up untracked files
  - $ git clean
- Tags
  - Human readable pointers to specific commits
  - $git tag -a v1.0 -m ‘Version 1.0’
    - Names the current HEAD commit as v1.0

GitK

- Git is great, but the command line can be a real pain
- GitK helps
  - Visualize commit graphs
  - Understand repo structure
- Here's a tutorial
- And some missing documentation
- Other GUIs exist!
  - Github Desktop, Git Kraken, and Git Tower all popular
- Github does some of this too