CS 35L

Software Construction Laboratory

Lecture 10.1

3rd June, 2019

Logistics

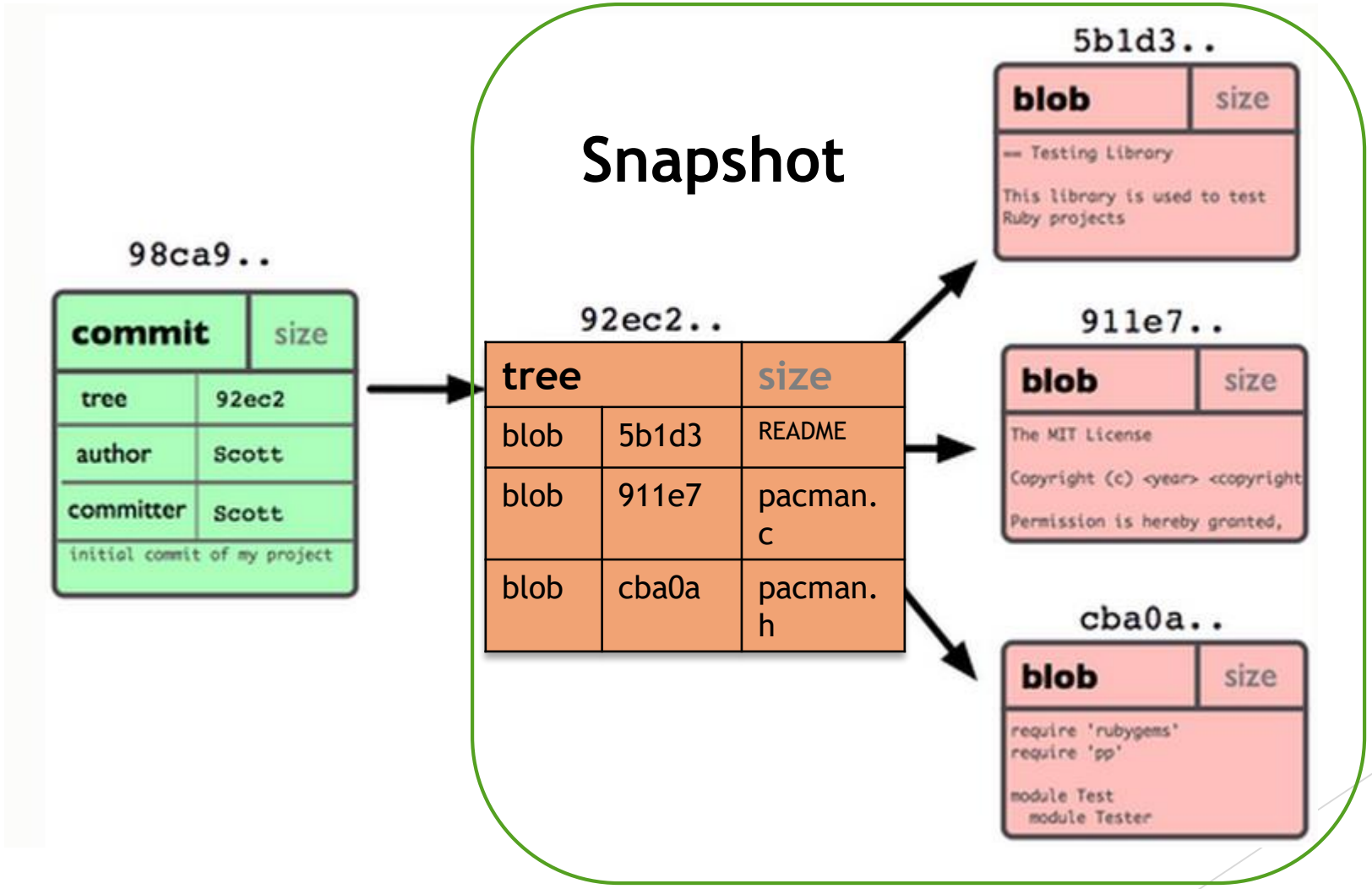
- Final Exam
  - Date: 9th June, 2019 (Sunday)
  - Time: 3pm to 6pm
  - Location: Broad 2160E
- Assignment 8 is due on 29*th May, 2019* at 11:55pm
  - Limited Late Policy – Up till 3rd June, 2019 only
- Assignment 9 is due on 5th June, 2019 at 11:55pm
  - Limited Late Policy – Up till 7th June, 2019 only
  - Extended!
- Instructor Evaluation
  - Chocolates!
  - Tips for Final Exam!

Review - Previous Lab

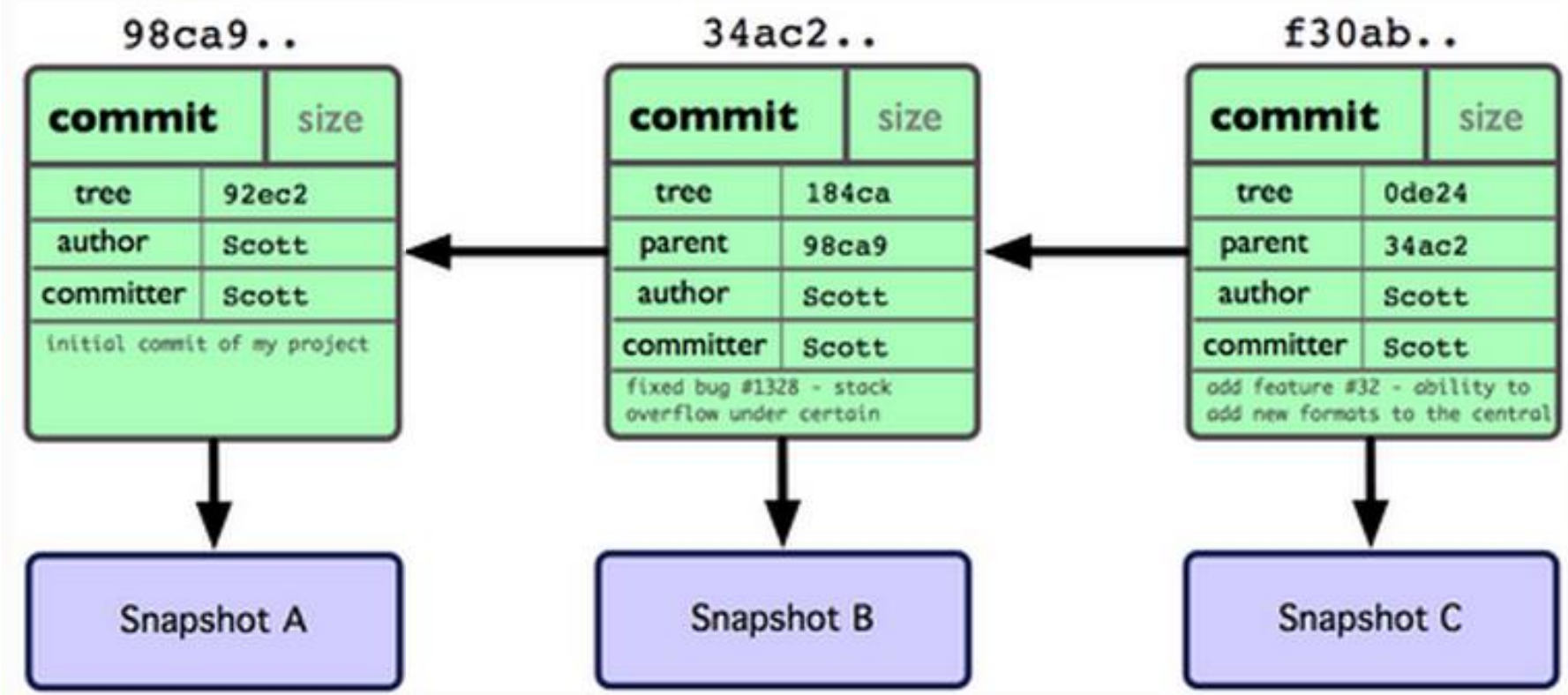
- Change Management
  - Why change management?
- Version Control System
  - Local, Centralized and Distributed
- Git
  - Git Repository Objects
    - Blobs, Trees, Commits, Tags
  - Git States
    - Working directory, staging area, git directory
- Initial git commands

GIT Source Control

Git Repo Structure



After 2 More Commits…

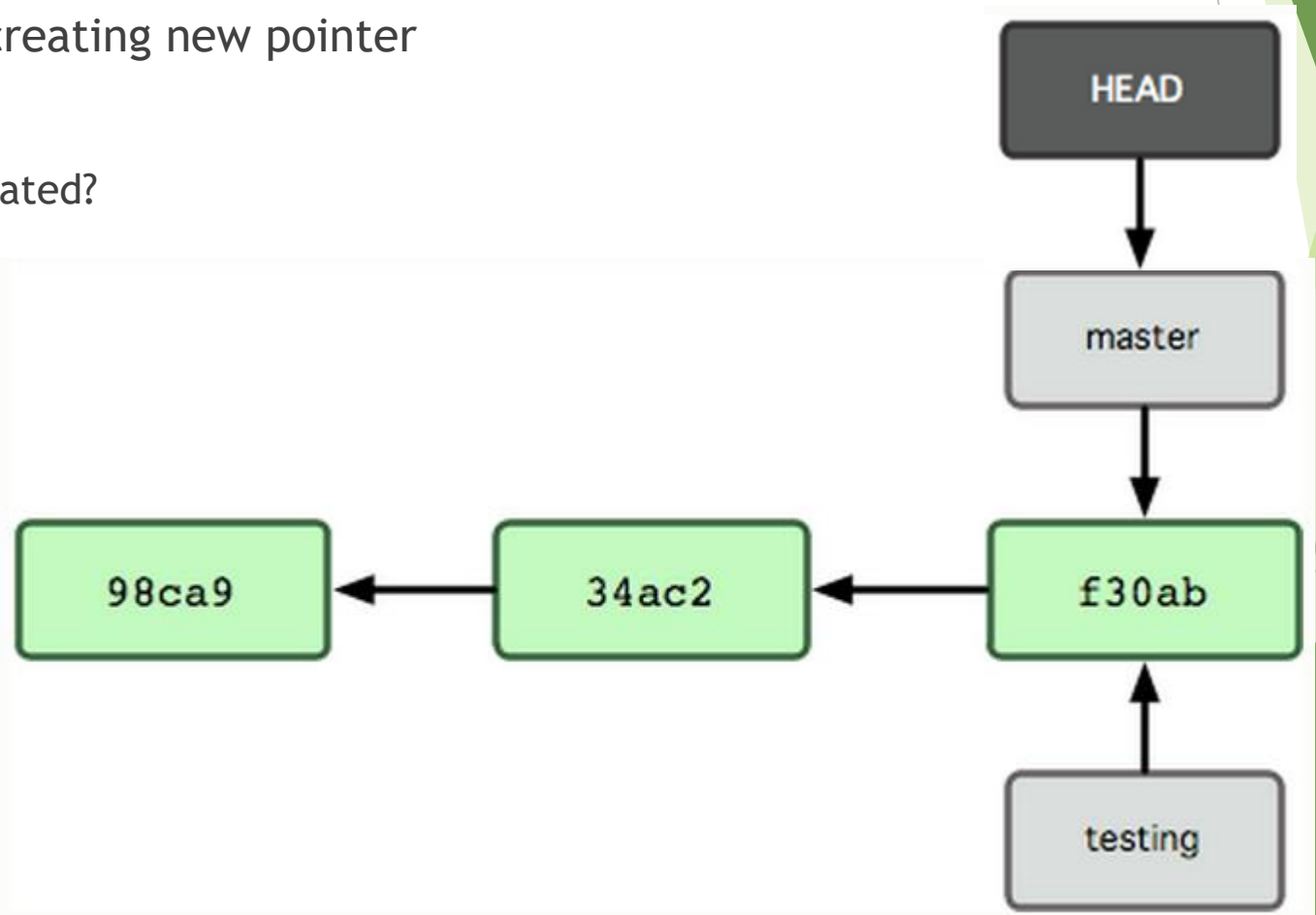


What is a Branch?

- A pointer to one of the commits in the repo (head) + all ancestor commits
- When you first create a repo, are there any branches?
  - Default branch named ‘master’
- The default master branch
  - points to last commit made
  - moves forward automatically, every time you commit

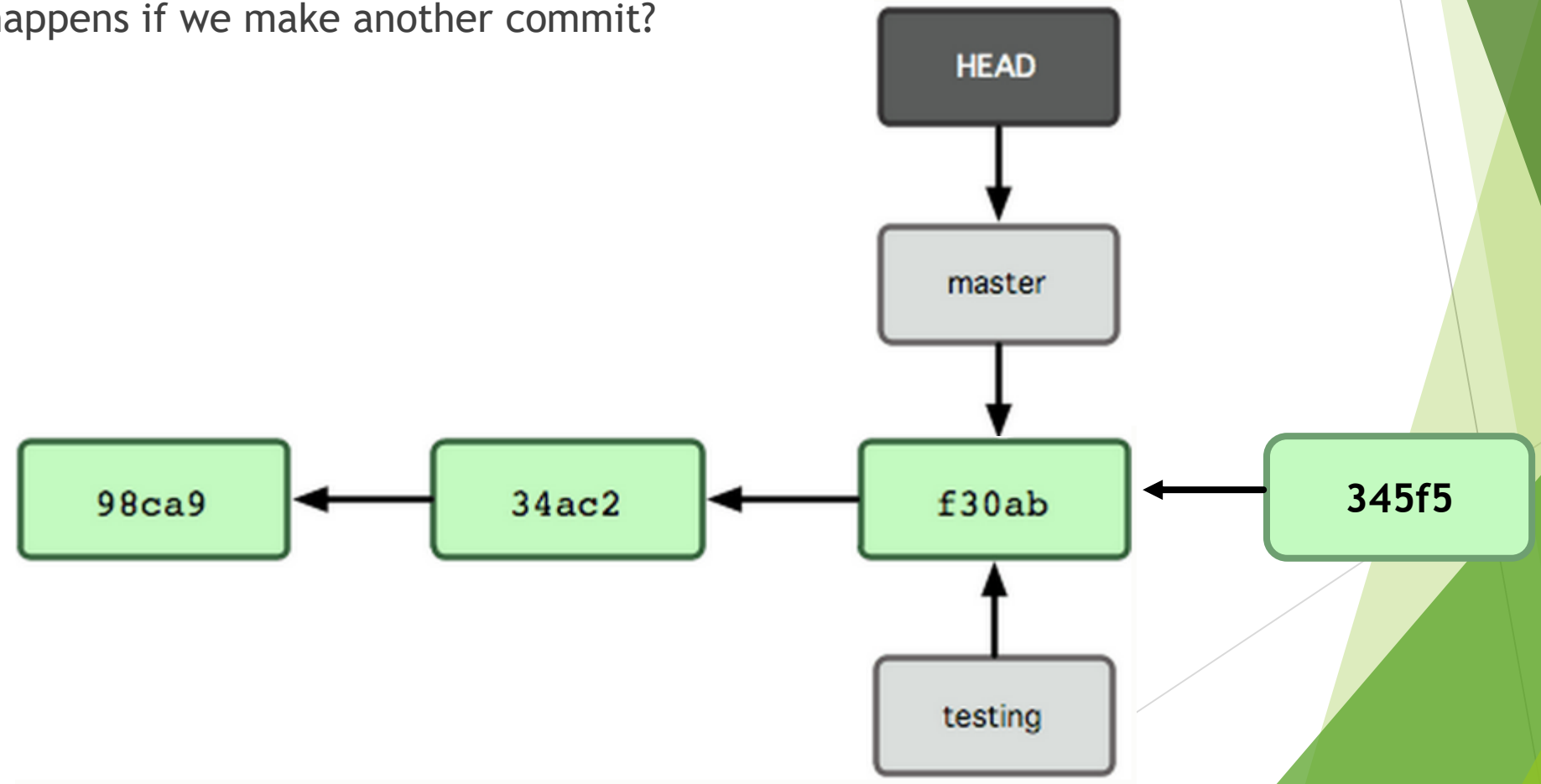
New Branch

- Creating a new branch = creating new pointer
  - `$ git branch` testing
  - Where is new branch created?
    - Current commit
- Where is current commit?
  - HEAD



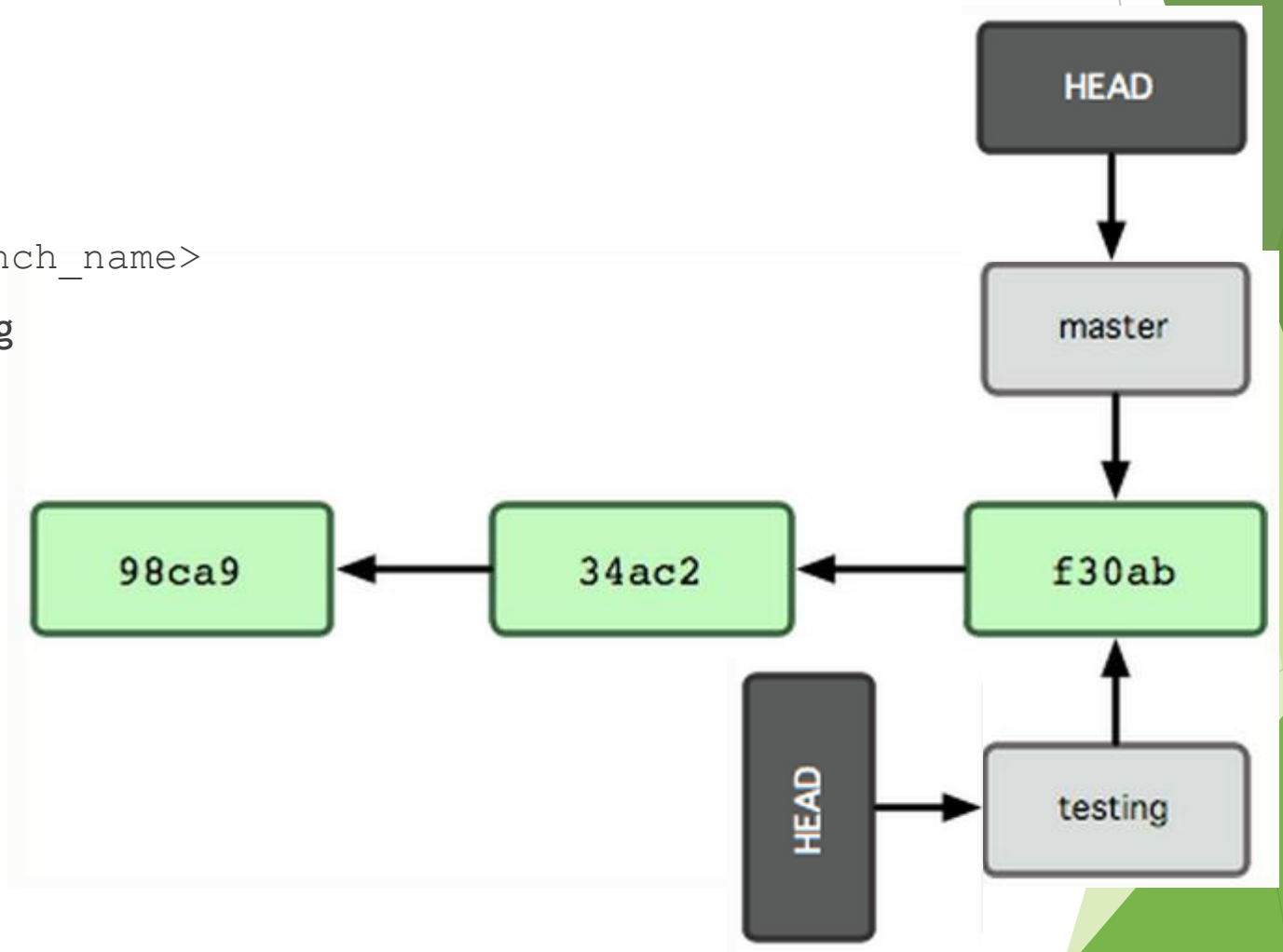
New Commit

- What happens if we make another commit?

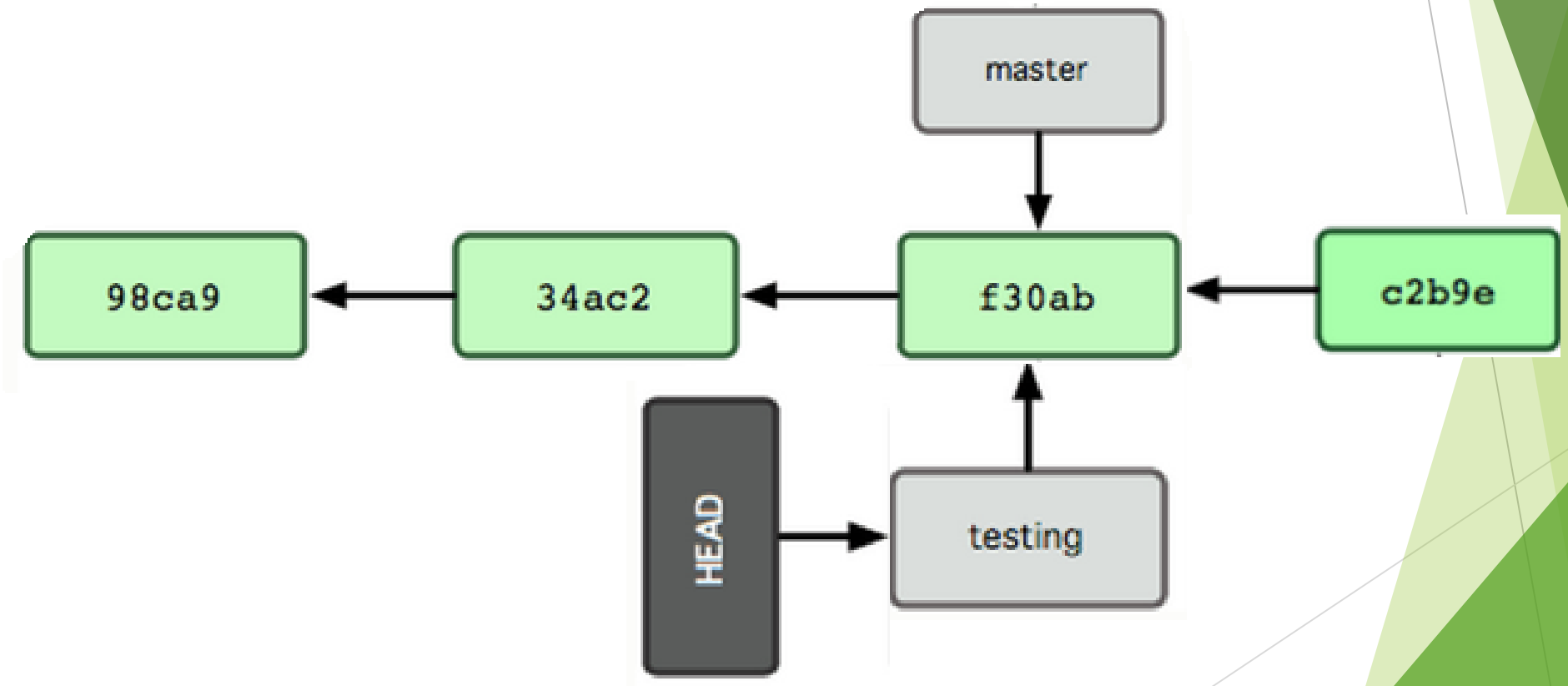


Switching to New Branch

- Check out new branch
  - `$ git checkout <branch_name>`
  - `$ git checkout` testing



Commit After Switch



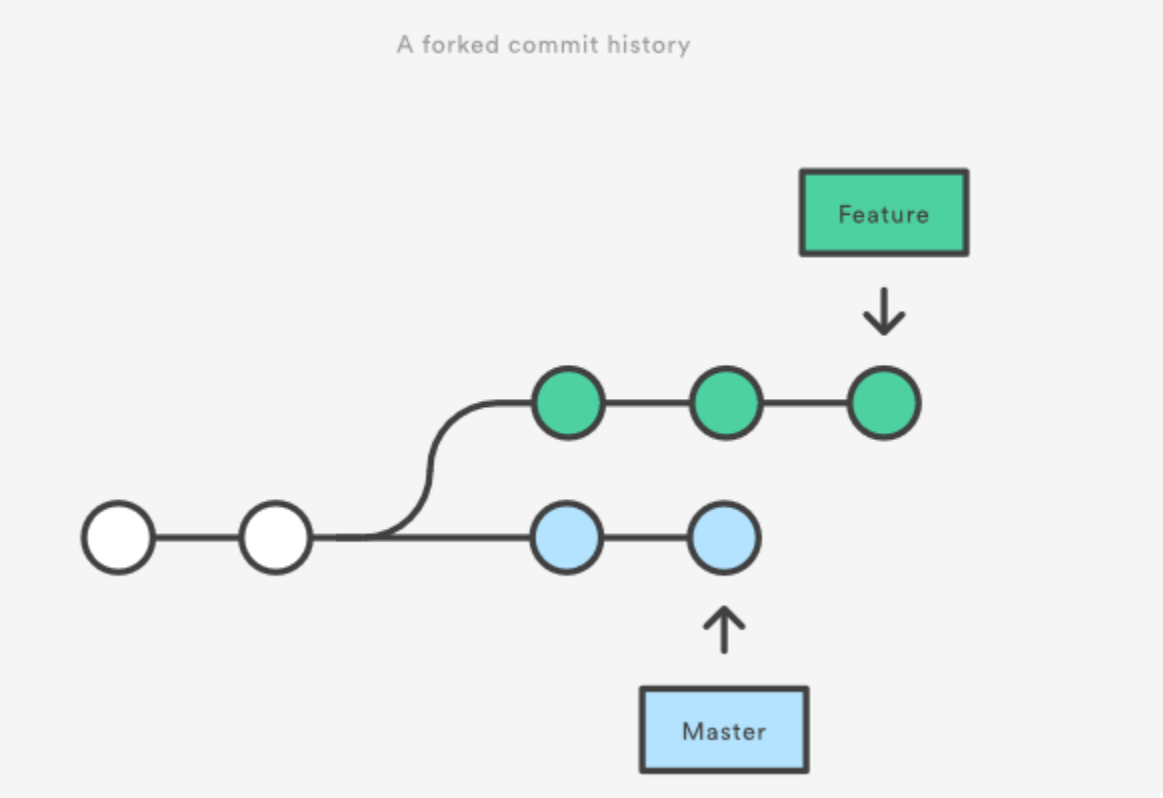
Why Branching?

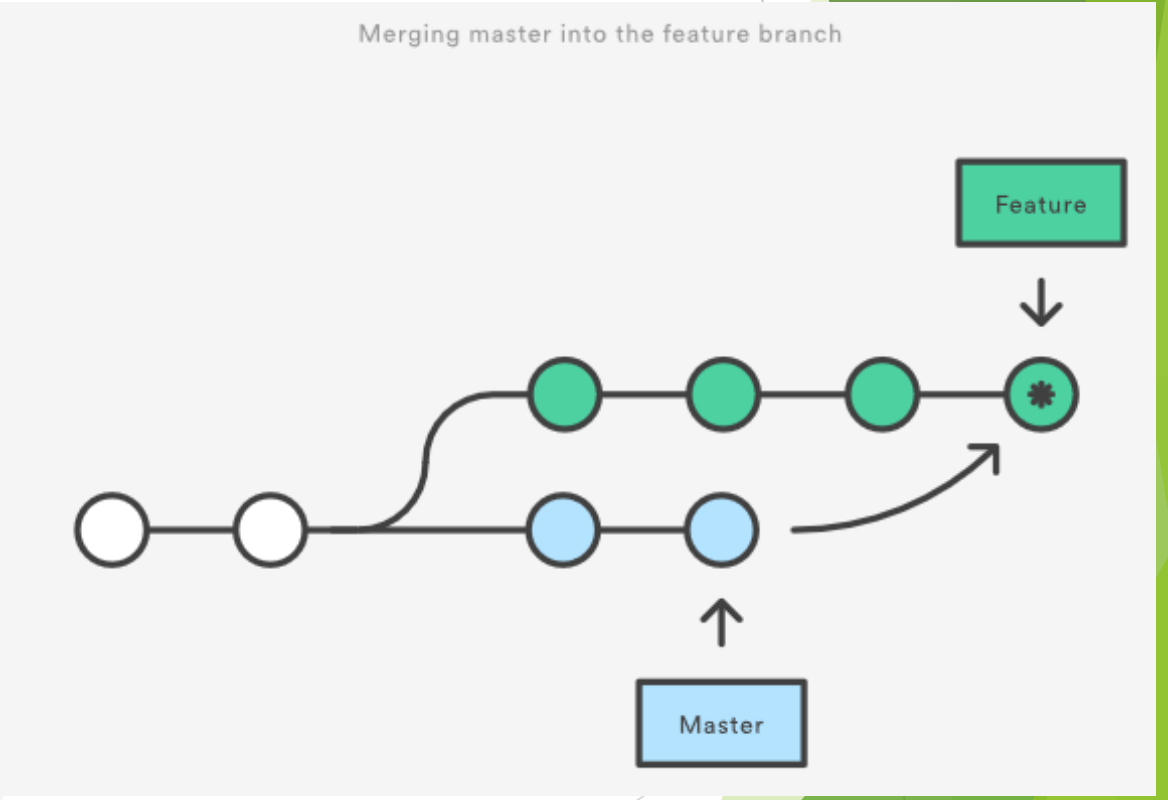
- Experiment with code without affecting main branch
- Separate projects that once had a common code base
- 2 versions of the project

Git integrating changes

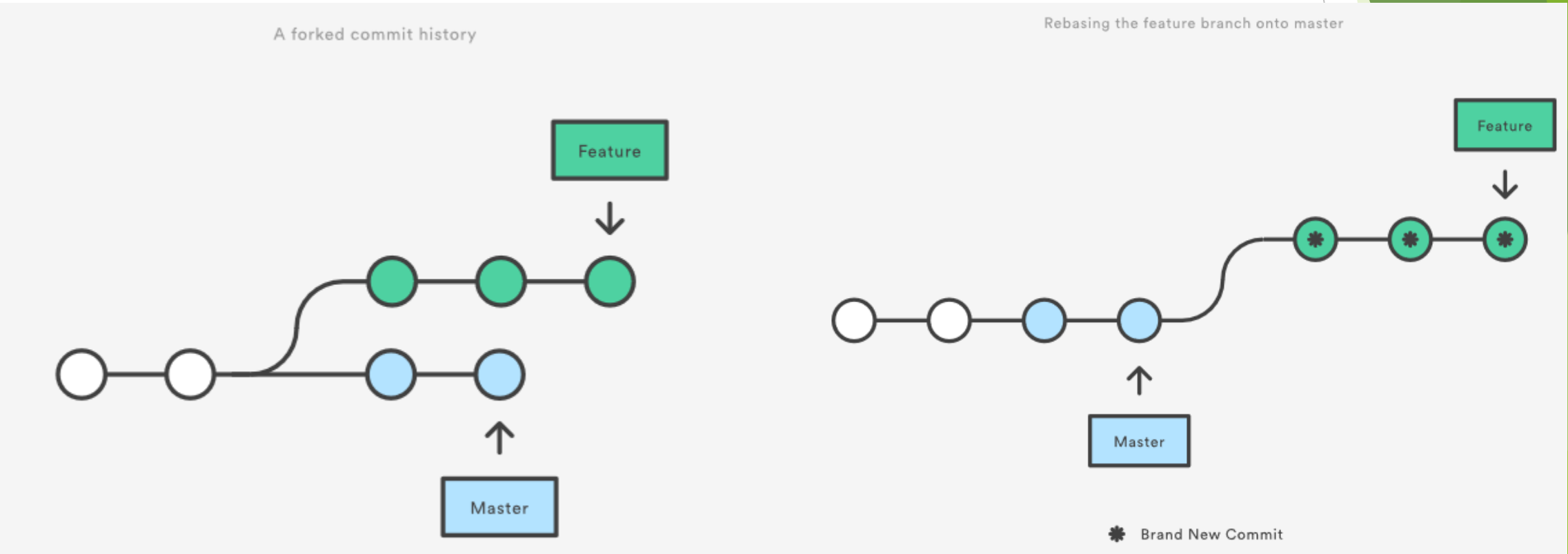
- Required when there are changes in multiple branches
- Two main ways to integrate changes from one branch to another
  - merge
  - rebase
- Merge is simple and straightforward
- Rebase is much cleaner

Git merge


A forked commit history
Feature
Master


Merging master into the feature branch
Feature
Master

Git rebase


A forked commit history
Feature
Master
Rebasing the feature branch onto master
Feature
Master
Brand New Commit

Merge Conflicts

- Usually git will do merge automatically
- Conflict arises when you changed the same part of the same file differently in the two branches you're merging together
- The new commit object will not be created
- You need to resolve conflicts manually by selecting which parts of the file you want to keep

Setting your identity on git

- Command: Git config
  - Git config –global user.name <Your name>
  - Git config –global user.email <Your email>

Setting up a repo

- Start from scratch
  - Git init
- Clone Existing Repo
  - Git clone <Link to Repo>

Working Copy of the Repo

- Adding and modifying files, tracking/staging changes - git add
- Viewing state - git status
- Ignoring files - .gitignore
- Viewing differences - git diff
- Removing staged files - git rm
- Committing changes
  - git commit
  - skipping the staging area (git commit -a)
- Viewing commits and history
  - git log
  - Patch option (-p)

Collaboration on git

- Fetching data
  - Git fetch
    - downloads the data to your local repository
    - it doesn’t automatically merge it with any of your work
  - Git pull
    - automatically fetches and then merge tracked remote branch into your current branch
- Pushing Data
  - Git push <remote> <branch>

Tagging

- Human readable pointers to specific commits
- Why do we need tags?
  - Additional information for a commit
- Types of tags
  - Annotated
  - Lightweight
- Example
  - git tag –a v1.0 –m ‘Version 1.0’
    - This will name the HEAD commit as v1.0
    - -a makes it an annotated tag
  - git tag v1.4-lw
    - Lightweight tag (Do not supply any flags)

Working with branches

- Git branch test
  - Create new branch
- Git branch
  - List all branches
- Git checkout test
  - Switch to test branch
- Echo “Hello World!” > hw
- Commit the change in new branch
  - changes on your branch need to be committed before switching
- Git checkout master
  - Back to master branch
- Git log
- Git merge test
  - Merge commits from test branch to current branch
- Deleting Branches
  - -d option

More git commands

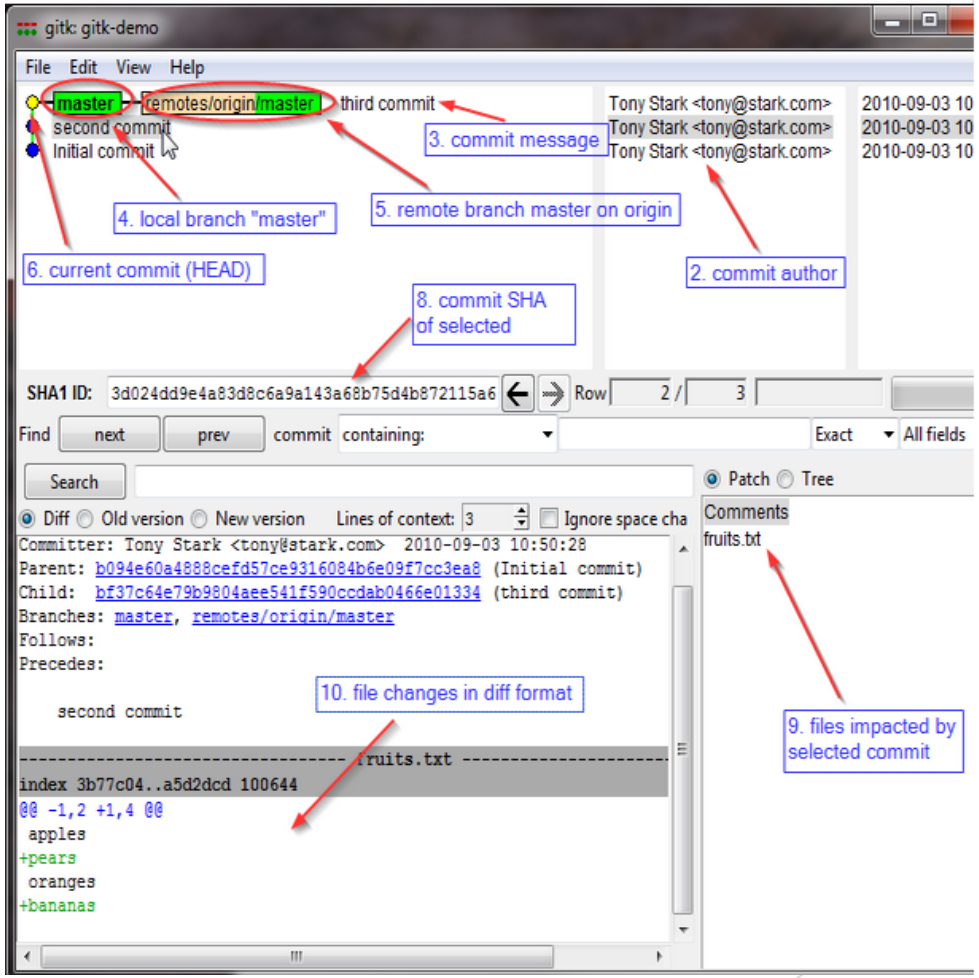
- Reverting
  - git checkout HEAD main.cpp
    - Gets the HEAD revision for the working copy
  - git checkout – main.cpp
    - Reverts changes in the working directory
  - git revert
    - Reverting commits (this creates new commits)
- Cleaning up untracked files
  - git clean

Assignment 9 - Laboratory

- Fix an issue with diff diagnostic
  - Apply a patch to a previous version
- Installing Git
  - Ubuntu: sudo apt-get install git
  - SEASNet: git is installed in /usr/local/cs/bin
  - Add it to PATH variable or use whole path
    - Export PATH=/usr/localcs/bin:$PATH
- Make a directry ‘gitroot’ and get a copy of the diffutils git repository
  - Mkdir gitroot
  - Cd gitroot
  - Git clone <url>
  - Follow steps given in the specs and use man git to find commands

Gitk

- A repository browser
  - Visualizes commit graphs
  - Used to understand the structure of the repo
  - Tutorial: http://lostechies.com/joshuaflanagan/2010/09/03/use-gitk-to-understand-git/



Gitk

- SSH into the server with X11 enabled
  - ssh -X for OS with terminal (OS X, Linux)
  - Select “X11” option if using putty (Windows)
- Run gitk in the ~eggert/src/gnu/emacs directory
  - Need to first update your PATH
    - $ export PATH=/usr/local/cs/bin:$PATH
  - Run X locally before running gitk
    - Xming on Windows, Xquartz on Mac

Assignment 9 - Laboratory

- Hints
  - Git clone
  - Git log
  - Git tag
  - Git show <hash>
  - Git checkout v3.0 –b <branchname>

Assignment 9 - Homework

- Publish patch you made in lab 9
  - Create a new branch “quote” of version 3.0
    - Branch command + checkout command **(git branch quote v3.0; git checkout quote)**
    - `$ git checkout v3.0 -b quote`
  - Use patch from lab 9 to modify this branch
    - Patch command
    - `$ patch –pnum < quote-3.0-patch.txt`
  - Modify ChangeLog file in diffutils directory
    - Add entry for your changes similar to entries in ChangeLog
  - Commit changes to the new branch
    - `$ git add .` `$ git commit –F <Changelog file>`
  - Generate a patch that other people can use to get your changes
    - `$ git format-patch -[num] --stdout > formatted-patch.txt`
  - Test your partner’s patch
    - Check out version 3.0 into a temporary branch `partner`
    - Apply patch with `git am` command: `$ git am < formatted-patch.txt`
    - Build and test with $ `make check`
    - Make sure partner’s name is in HW9.txt for #8

Presentations

- Today's Presentation:
  - Nishanth Yeddula
- Next up:
  - Bryan Pan
  - Yuchen Yao

Questions?