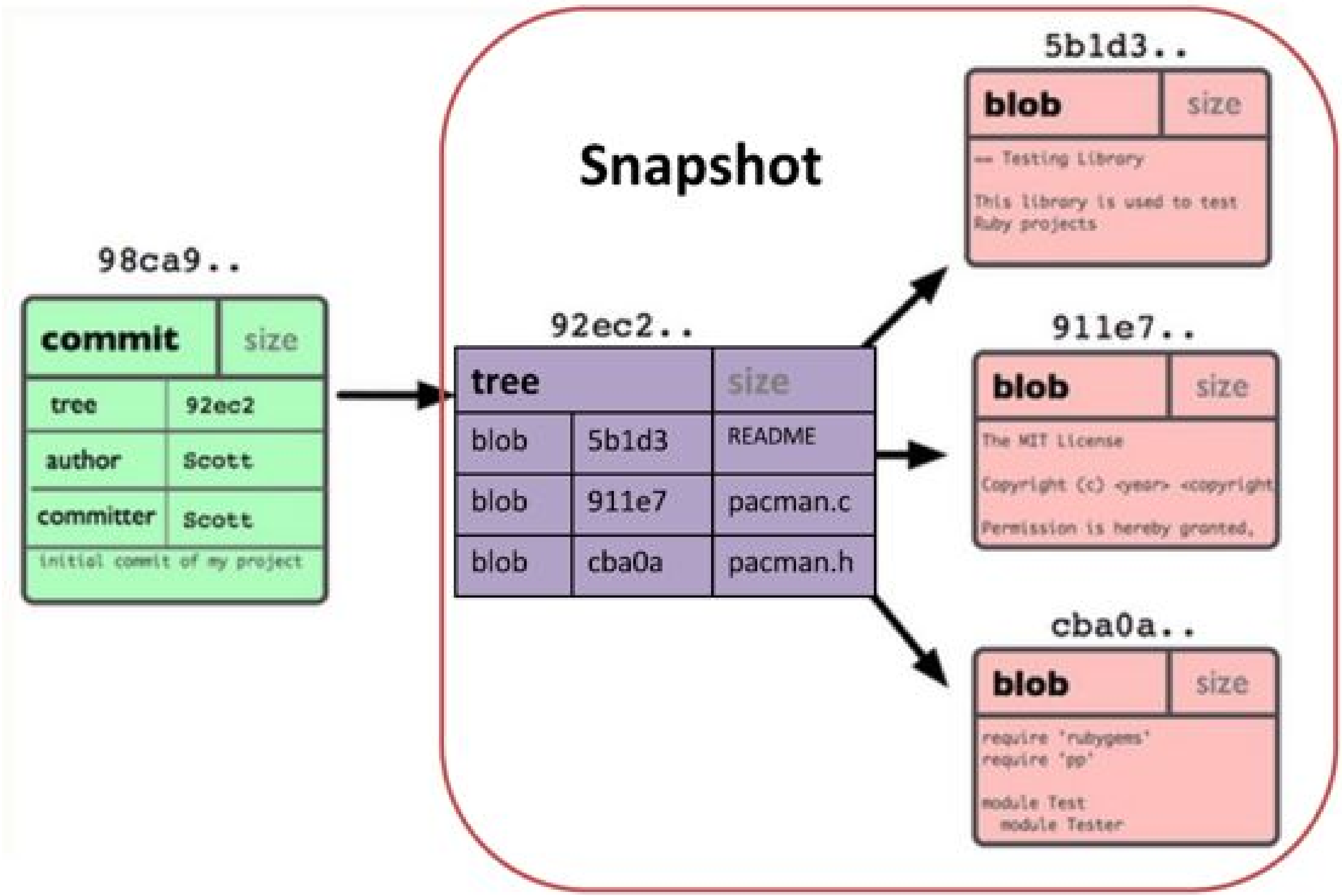
Change Management

Week 5

Git Repository Objects

- Objects used by Git to implement source control
  - **Blobs**
    - Sequence of bytes
  - **Trees**
    - Groups blobs/trees together
  - **Commit**
    - Refers to a particular “git commit”
    - Contains all information about the commit
  - **Tags**
    - A named commit object for convenience (e.g. versions of software)
- Objects uniquely identified with **hashes**

Git Repo Structure



Basic branching

- Git commit will create a commit object

```
$ git add README test.rb LICENSE
$ git commit -m 'The initial commit of my project'
```

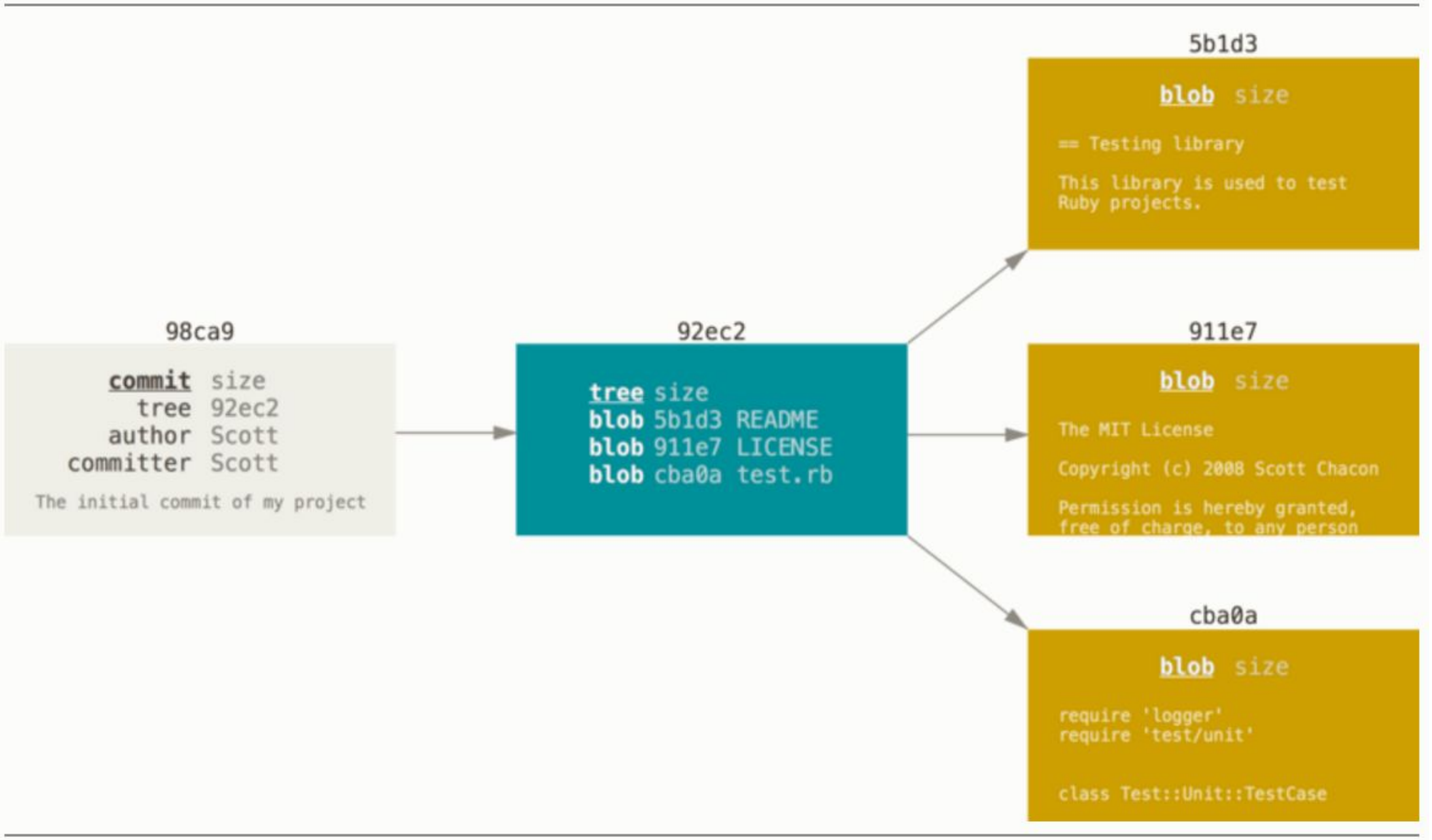


Figure 3-1. A commit and its tree

- While you have more commits

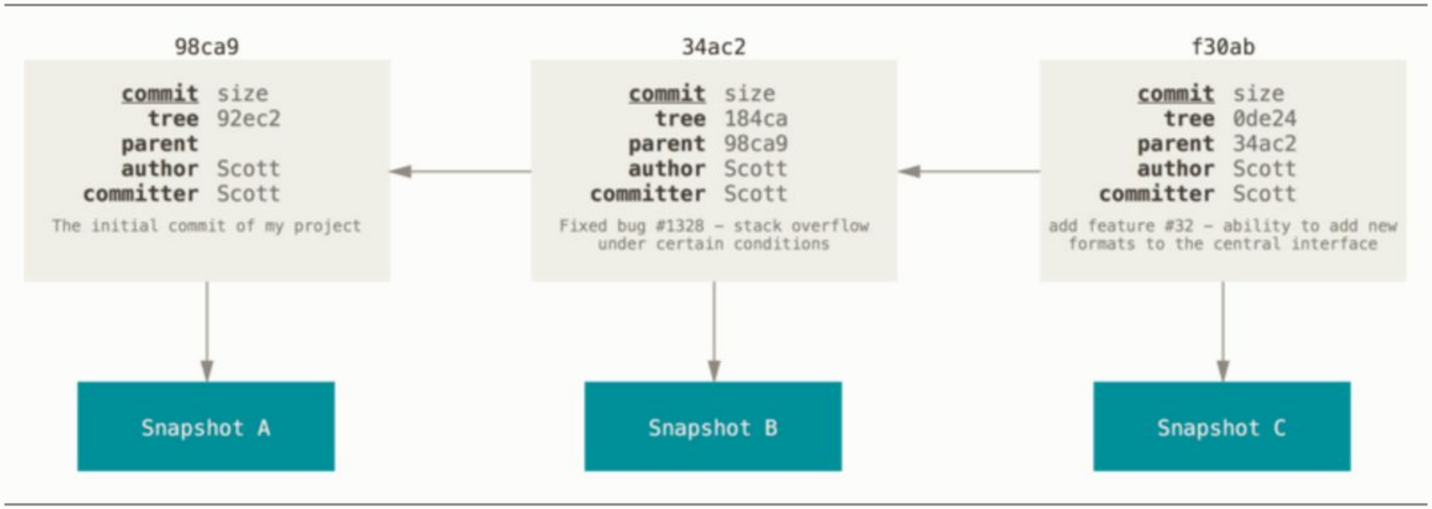
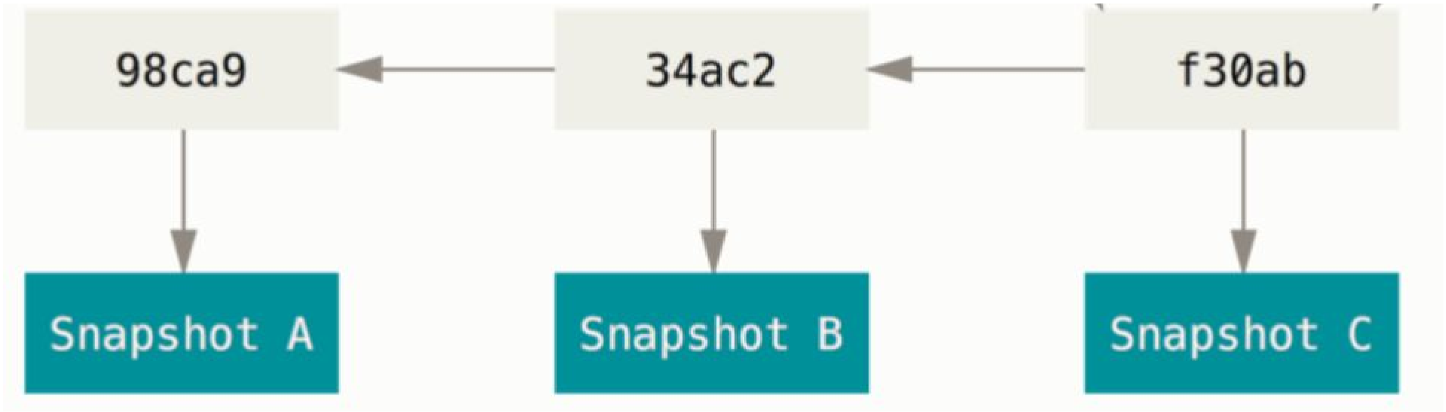


Figure 3-2. Commits and their parents



- branch: a lightweight movable pointer to one of these commits and all ancestor commit
- Master branch: the default branch name in Git. Every time you commit, it moves forward automatically.
- To create/delete a new branch

  git branch <new_branchname>

  git branch -d <branch_name>

```
$ git branch testing
```

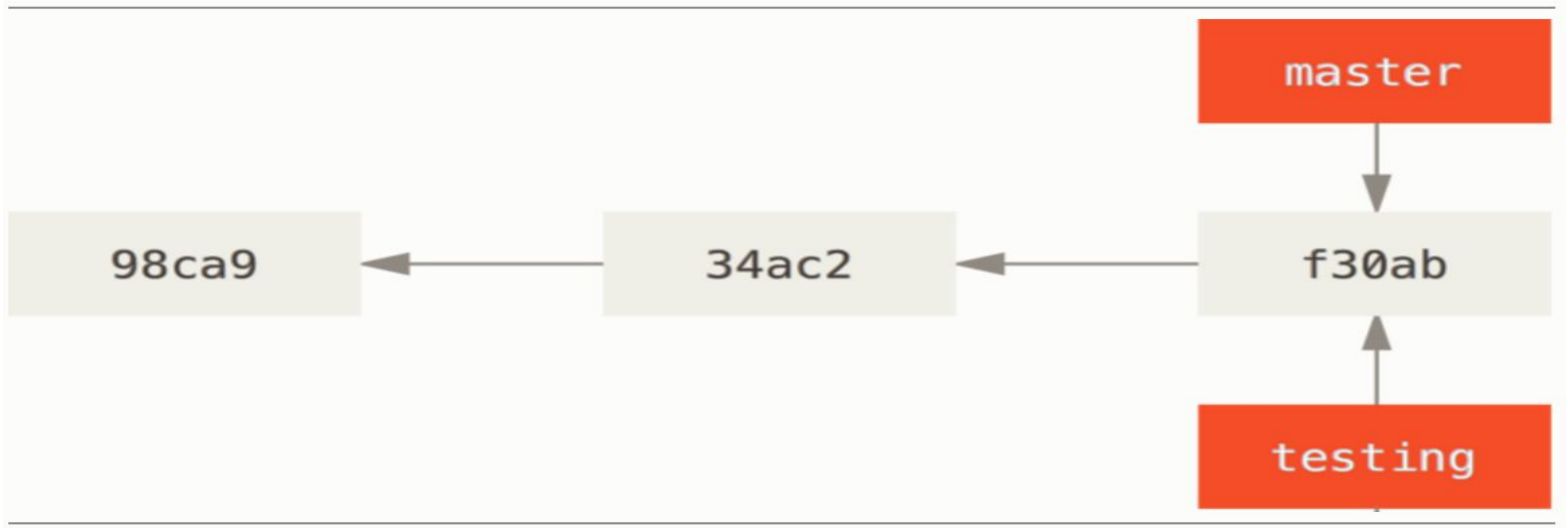
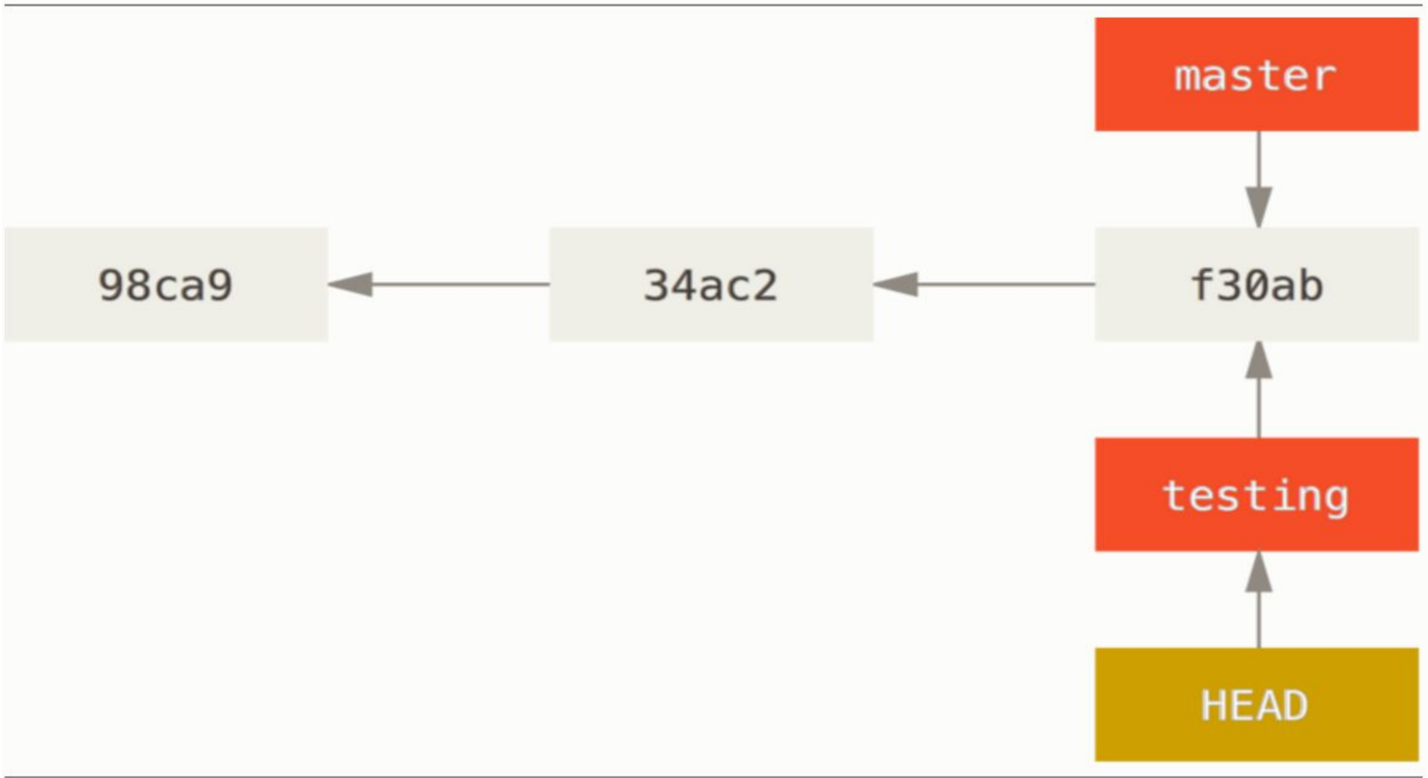


Figure 3-4. Two branches pointing into the same series of commits

- HEAD: Git records the branch you are currently working on by a pointer HEAD
- git branch will only create a new branch, but will not switch branch
- To switch to another branch

  git checkout <branch_name>

```
$ git checkout testing
```



- To create a branch and switch to it

  git checkout -b <branch_name>

Q1: What will the commit graph be like after running the following command?

```
$ vim test.rb
$ git commit -a -m 'made a change'
```

```
$ git checkout master
```

```
$ vim test.rb
$ git commit -a -m 'made other changes'
```

You can see the whole history by

```
git log --oneline --decorate --graph --all
```

Q2: Why do we need branching?

Why Branching?

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

Basic Merging

- To merge another branch into current branch

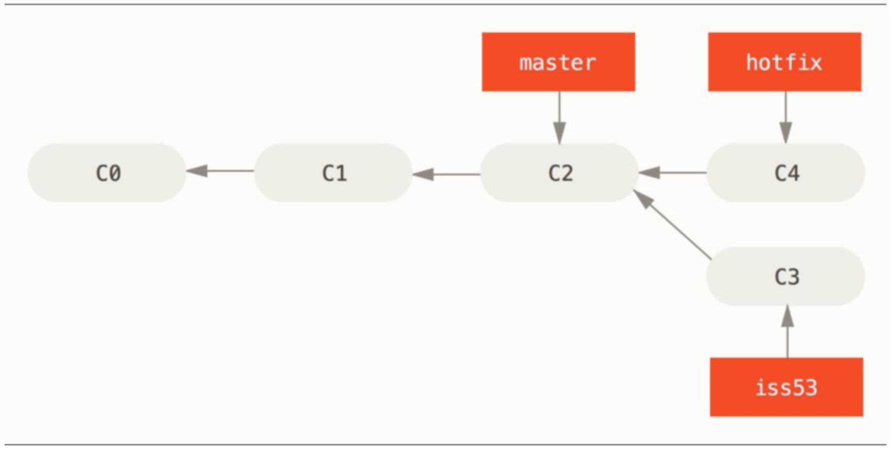
  ```
  git merge <branch_name>
  ```

- Two types

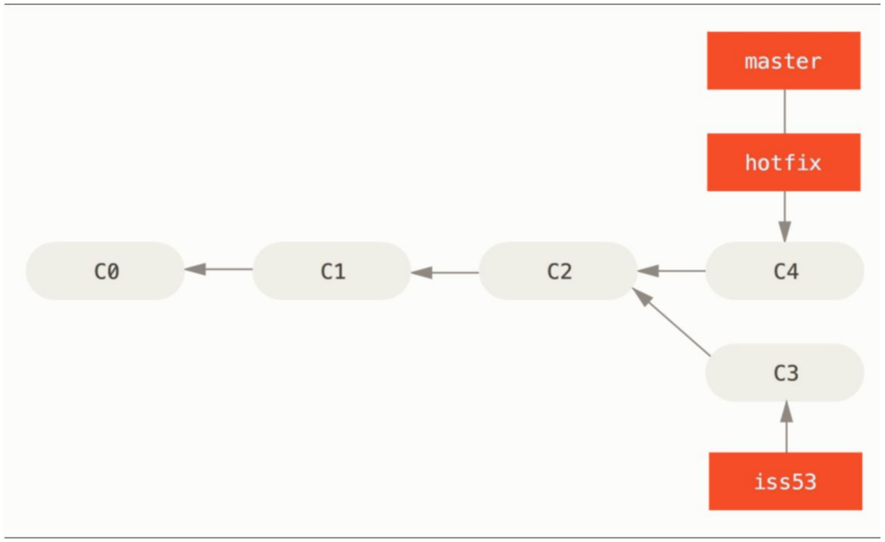
- fast-forward
- Three-way merge

Fast-forward

- merge one commit with a commit that can be reached by following the first commit’s history
- Git merge will simply move the pointer forward



```
$ git checkout master
$ git merge hotfix
```



Three-way merge

- Three parties: two snapshots pointed to by the branch tips and the common ancestor of the two
- Git create a new snapshot from the merge and automatically create a new commit pointing to it
- Git will find the appropriate common ancestor automatically

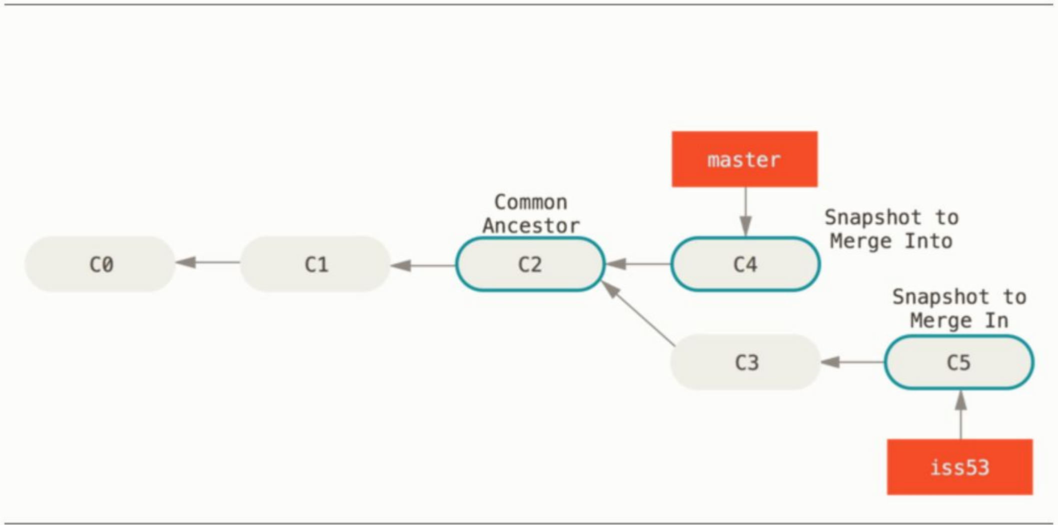


Figure 3-16. Three snapshots used in a typical merge

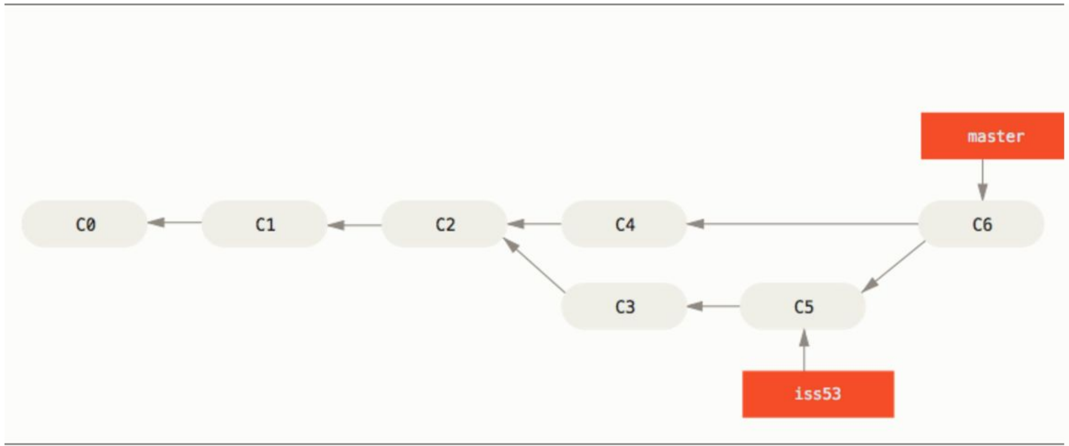


Figure 3-17. A merge commit

Basic merge conflict

- 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

Try to merge iss53 to current branch

```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

Check conflicts using git status

```
$ git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)

    both modified:      index.html

no changes added to commit (use "git add" and/or "git commit -a")
```

- Git adds standard conflict-resolution markers to the files that have conflicts
- you can open them manually and resolve those conflicts

```
<<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=======
<div id="footer">
 please contact us at support@github.com
</div>
>>>>>>> iss53:index.html
```

Manually process the conflicting lines

```
<div id="footer">
please contact us at email.support@github.com
</div>
```

- Check whether conflicts have been resolved

```
$ git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

    modified:   index.html
```

- Run git commit to submit the changes

```
Merge branch 'iss53'

Conflicts:
    index.html
#
# It looks like you may be committing a merge.
# If this is not correct, please remove the file
#       .git/MERGE_HEAD
# and try again.


# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch master
# All conflicts fixed but you are still merging.
#
# Changes to be committed:
#       modified:   index.html
#
```

Hints for Assignment 4

Cite Jin Wang

Homework 4

- Publish the patch made in lab 4
- Create a new branch “quote” of version 3.0
  - $ git checkout v3.0 –b quote
- Use patch from lab 4 to modify this branch
  - $ patch –pnum < quote-3.0-patch.txt
- Modify the change log in *diffutils* directory
  - Add entry for your changes into those in the change log

Homework 4

- Commit changes to the new branch

  $ git add . *$ git commit –F [change log file]*

- Generate a patch that other people can use to get your changes

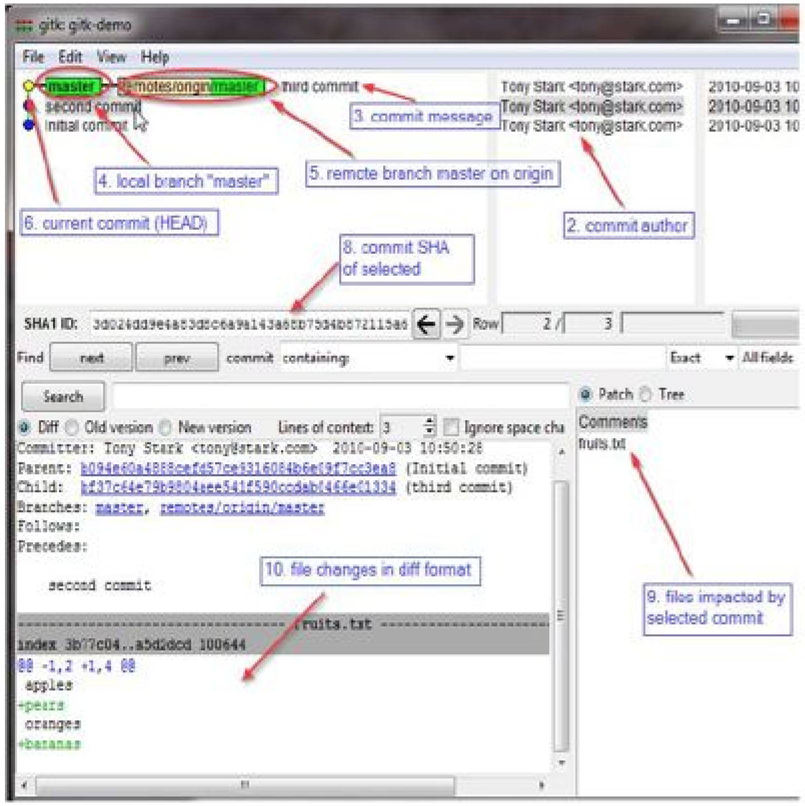
  $ git format-patch -[num] –stdout > [patch file]

- Test your partners patch
  - Choose a partner from this class
  - Apply patch with command *git am*
  - Build and test with command *make check*

Homework 4 -- Gitk

- A repository browser
  - Visualize commit graphs
  - Understand the structure of repo
  - Tutorial: [Use gitk to understand git]

  See supplement materials


gitk: gitk-demo
File Edit View Help
master
remotes/origin/master
third commit
second commit
initial commit
Tony Stark <tony@stark.com> 2010-09-03 10
3. commit message
4. local branch "master"
5. remote branch master on origin
6. current commit (HEAD)
2. commit author
8. commit SHA of selected
SHA1 ID: 3d026dd9e4a53d5c6a9a143a60b75d4b57211586
Row 2 / 3
Find next prev commit containing: Exact All fields
Search
Patch Tree
Diff Old version New version Lines of context: 3 Ignore space cha
Comments
fruits.txt
10. file changes in diff format
9. files impacted by selected commit

```
Committer: Tony Stark <tony@stark.com>  2010-09-03 10:50:28
Parent: b094e60a4888cefd57ce6316084b6e09f7cc3ea8 (Initial commit)
Child:  bf37c44e79b9804aee541f590ccdab1466e01334 (third commit)
Branches: master, remotes/origin/master
Follows:
Precedes:

    second commit

---------------------------------- fruits.txt ----------------------
index 3b77c04..a5d2dcd 100644
@@ -1,2 +1,4 @@
 apples
+pears
 oranges
+bananas
```

Homework 4 -- Gitk

- Usage
  - ssh –X for linux and MacOS
  - Select “X11” option if using putty (Windows)
  - See supplement materials [Putty X11 forwarding]
- Run gitk in the ~/eggert/src/gnu/emacs directory
  - Need first update your path

    ```
    export PATH=/usr/local/cs/bin:$PATH
    ```
  - Run X locally before running gitk

    Xming on Windows