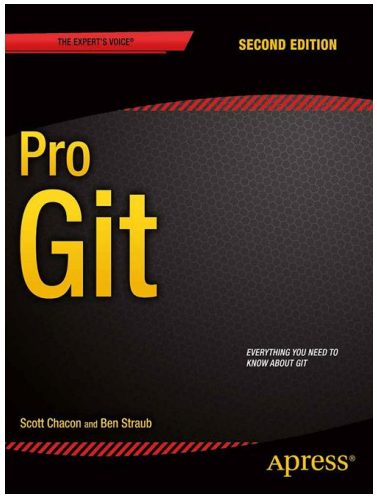
Git Branching

What is a branch?

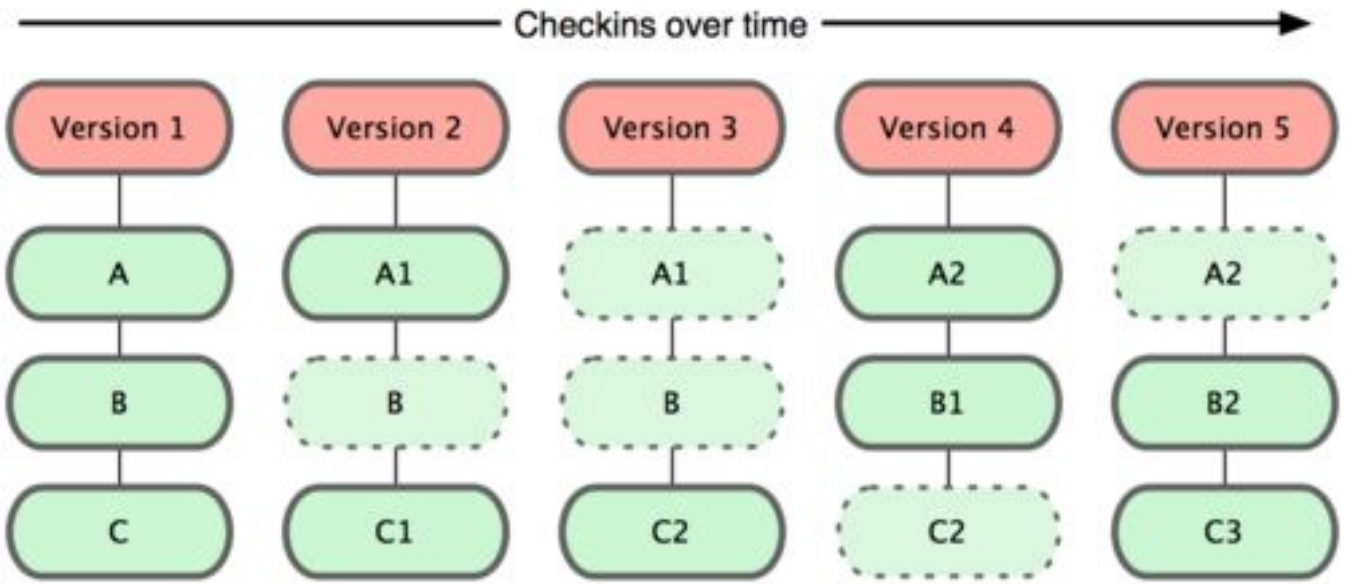
Attribution

- Slides attributed to Dr. Cyndi Rader. The original version can be found at http://eecs.mines.edu/Courses/csci306/ChapterNotes.html
- Figures and info for this lecture come primarily from Pro Git, available: http://www.git-scm.com/book/en/v2
- Material is used in accordance with the Creative Commons Attribution Non Commercial Share Alike 3.0 license.



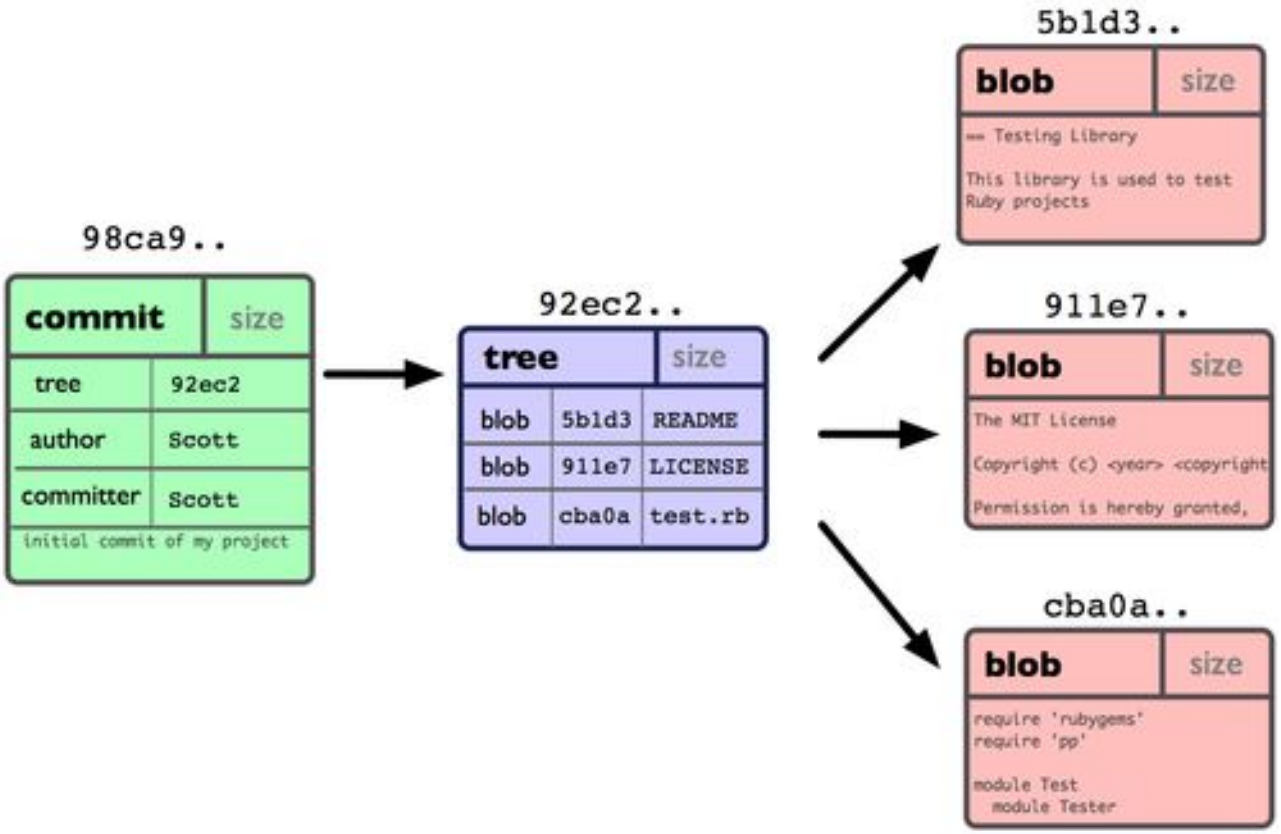
Book available online for free at: http://git-scm.com/book/en/v2

Review: Distributed - Snapshots

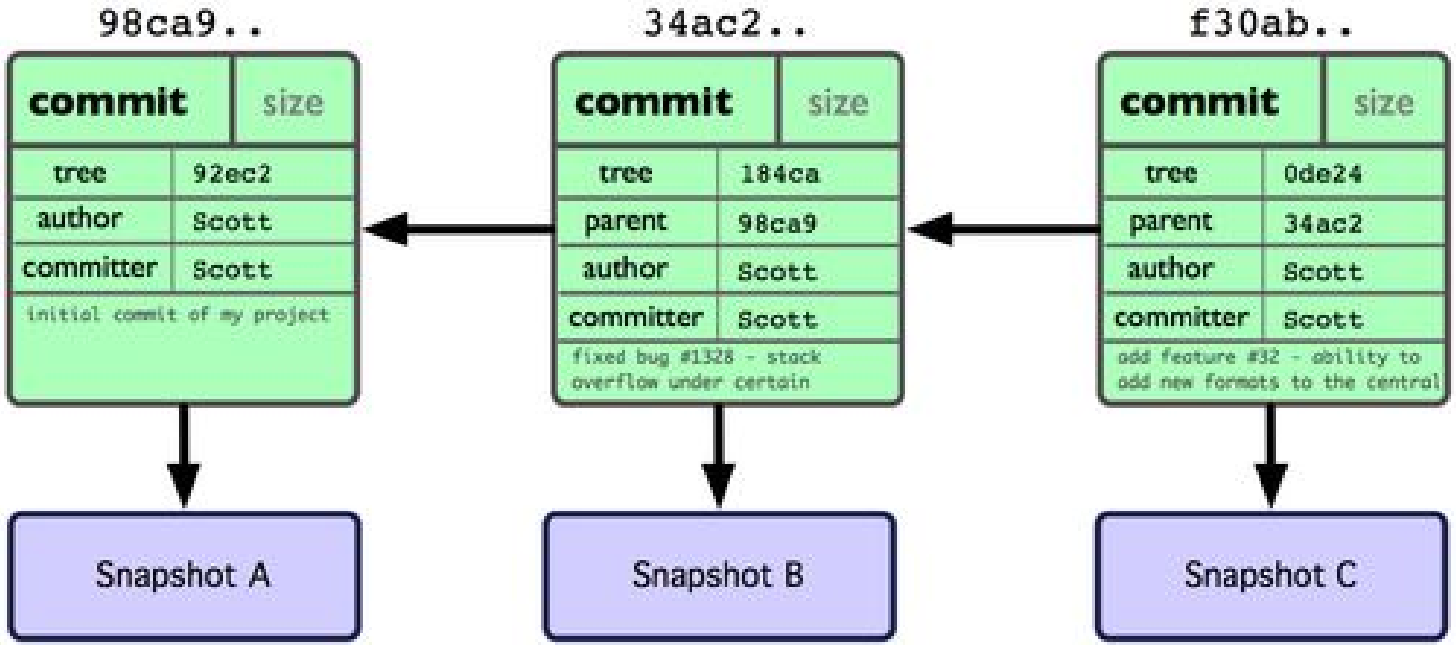


Files are stored are stored in snapshots (commits) in git database (in compressed format).
Snapshots are referenced by SHA-1 hash.
Must “checkout” from database into working directory to edit.

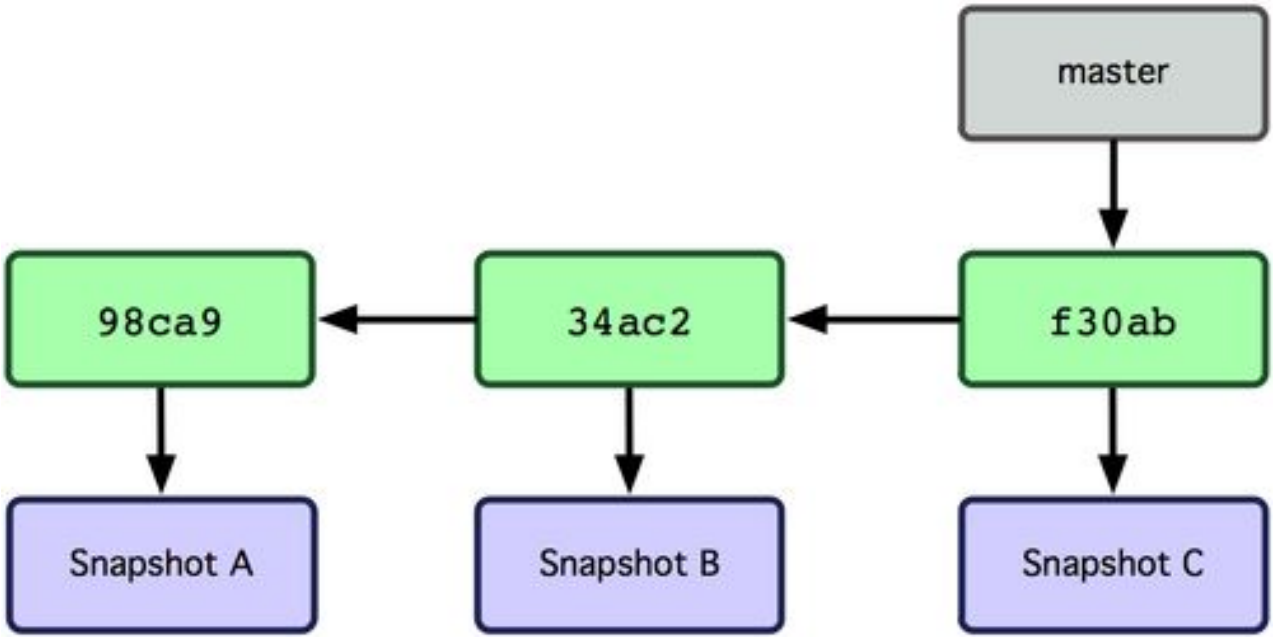
Example commit, 3 files



After 3 commits



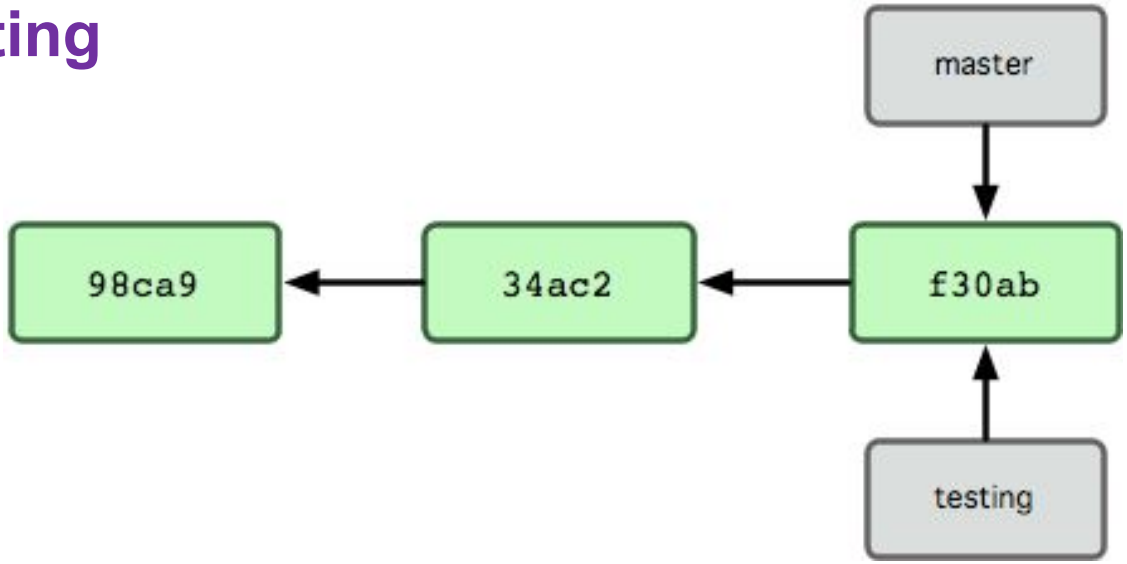
Default branch is master



Branch pointer (master) points to the tip of the commit tree
master moves forward automatically when you commit

Add a branch

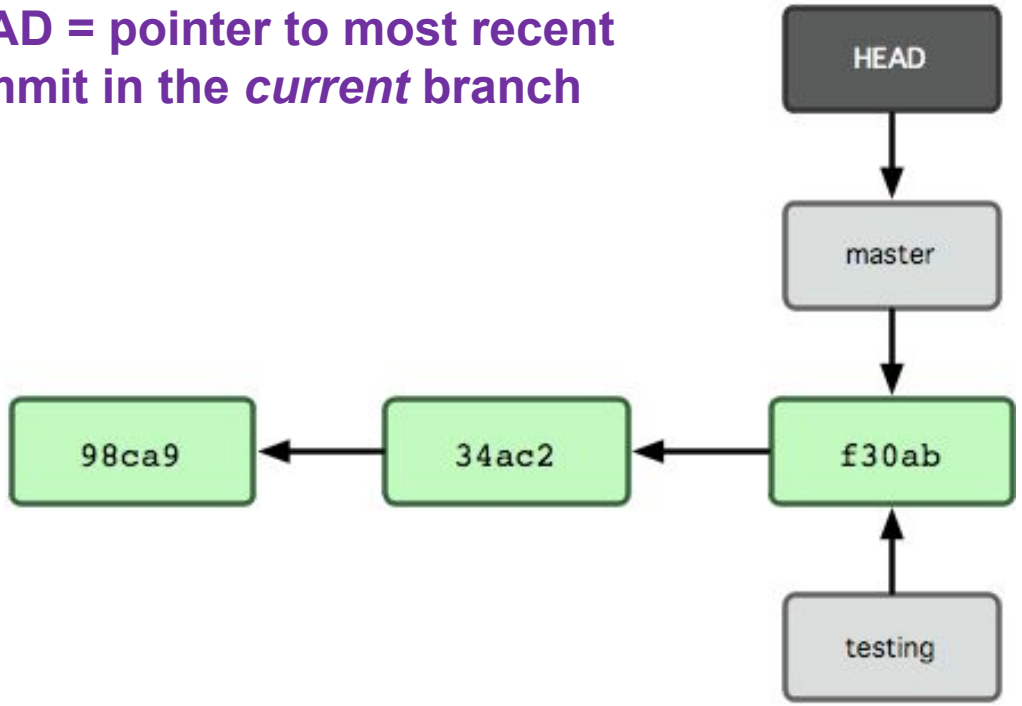
$ git checkout master
$ git branch testing



Why is creating a branch in git cheap and quick?
It's just creating a 40-character SHA-1 checksum of the commit it points to

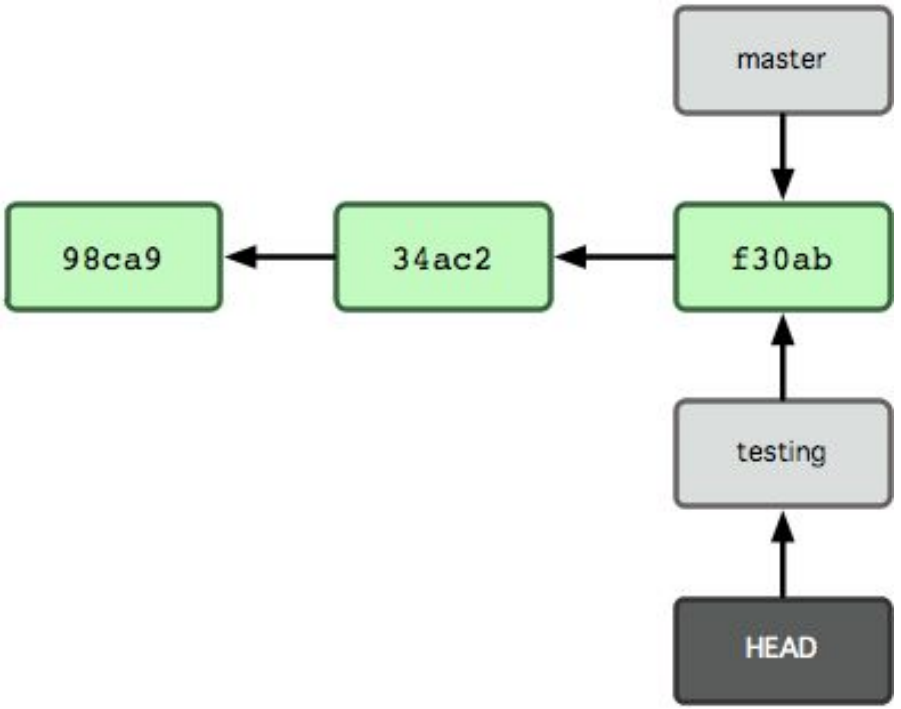
But you're still on master

HEAD = pointer to most recent commit in the *current* branch



Use checkout to change branch

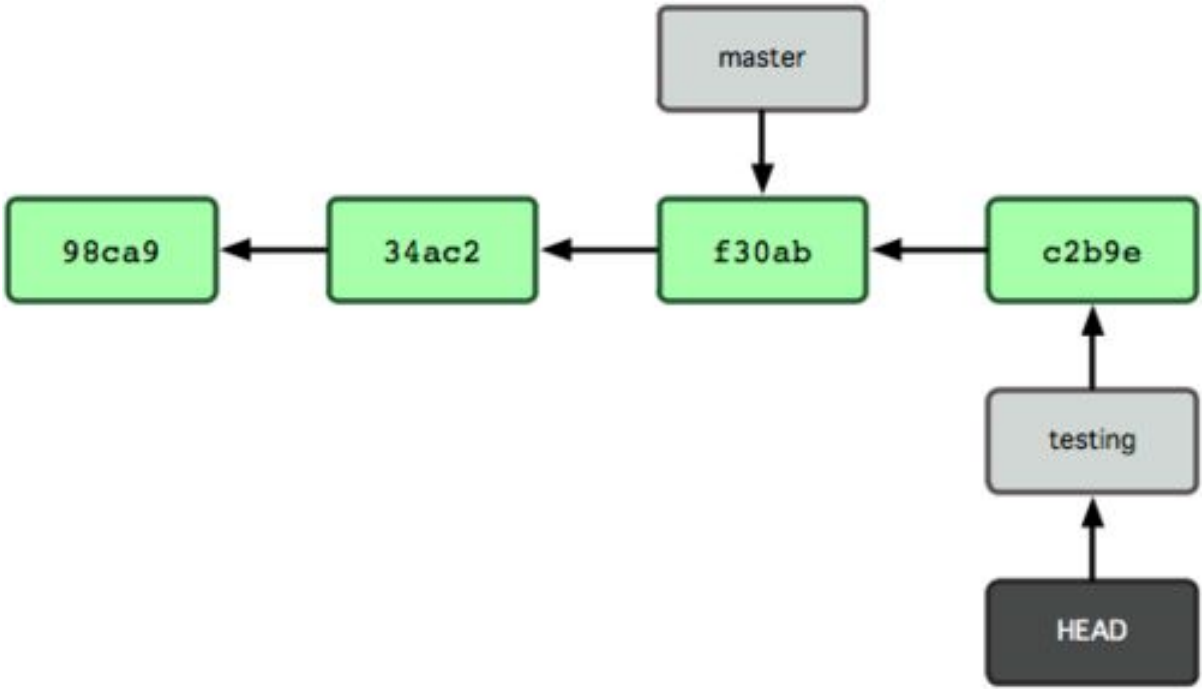
$ git checkout testing



$ git checkout moves the HEAD and updates the stage and the working directory to match the commit

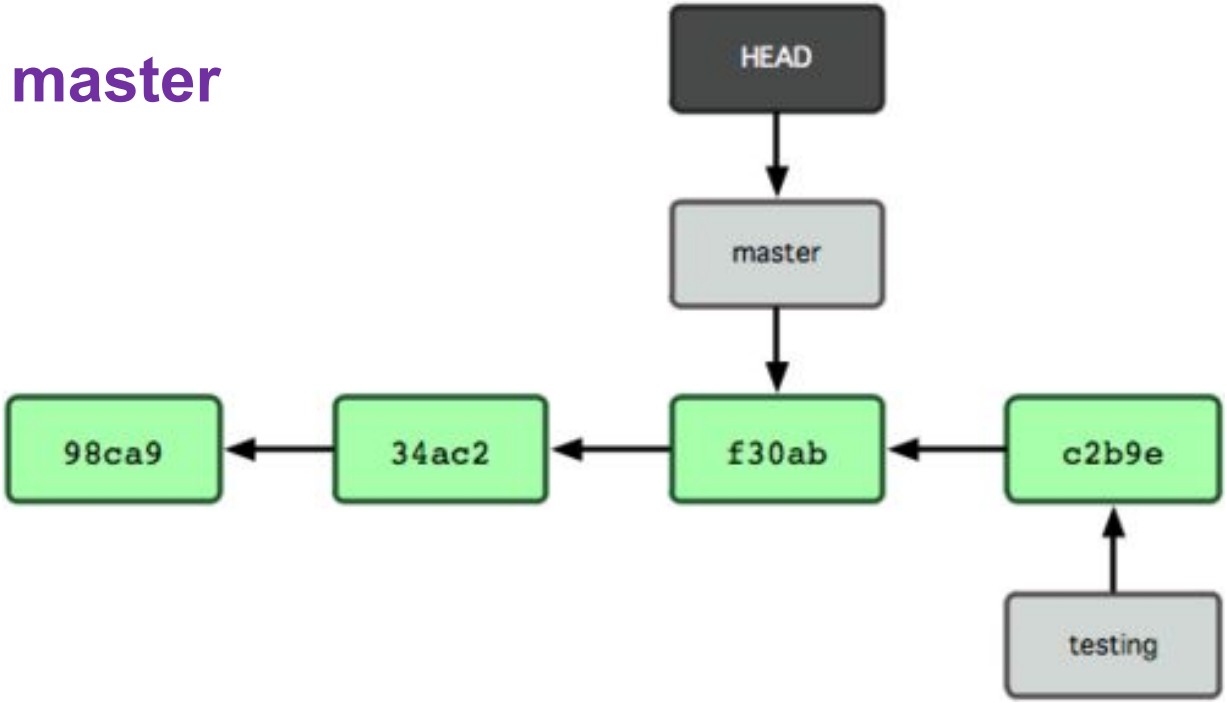
Make changes on that branch

edit test.rb (e.g., vim index.html)
$ **git commit –a –m ‘Fix test problem’**



Switch back to master

$ git checkout master



This also reverts files in your working directory back to master.

So the edit to index.html is no longer in your working directory (but you haven't lost it, as long as you committed before switching –it will be in your local database).

Hands on branching

- Create Java Project
- Create a class TheBrancher, including main
- Create .gitignore (copy)
- $ git init
- $ git add .
- $ git commit -m "Initial load"

- Update source as follows:

```
public class TheBrancher {
  private String theLeaf;
   public void updateLeaf(String newLeaf) {
       theLeaf = newLeaf;
   }
   public void showLeaf() {
       System.out.println(theLeaf);
   }
   public static void main(String[] args) {
       TheBrancher brancher = new TheBrancher();
        brancher.updateLeaf("Turning yellow for fall");
        brancher.showLeaf();
   }
}
```

Hands on branching continued

- Test/run the code in NetBeans
- $ git status
- $ git commit -am "Add a leaf"
- $ git log --oneline --graph --decorate

- Create a branch to work on a new feature
- $ git branch needTree # needTree points to HEAD (=master)
- See the branch
- $ git branch [-v] # Current branch marked with *

- Switch to new branch
- $ git checkout needTree
- $ git status

Tip: The create and checkout of a branch can be combined into one command:
$ git checkout –b needTree

Hands on branching continued

- Edit TheBrancher class

```
private String theTree;
public void makeATree() {
   theTree = ''Aspen'';
 }
 @Override
 public String toString() {
   return ''TheBrancher [theLeaf='' + theLeaf + '',  theTree='' + theTree + '']'';
   }

# and in main
    brancher.makeATree();
   System.out.println(brancher);
```

Hands on branching continued

- $ git status
- $ git commit -am "Add a tree"
- $ git status

- Feature finished, go back to master
- $ git checkout master

- What happened in NetBeans?

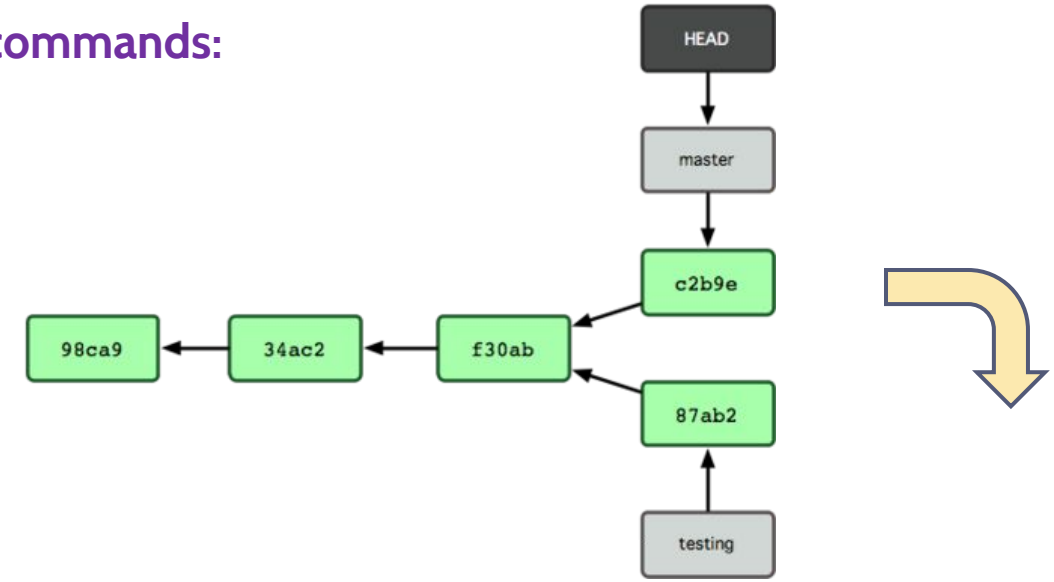
- The results of the feature branch need to be integrated into master branch, meet the **git merge**

Basic Merging

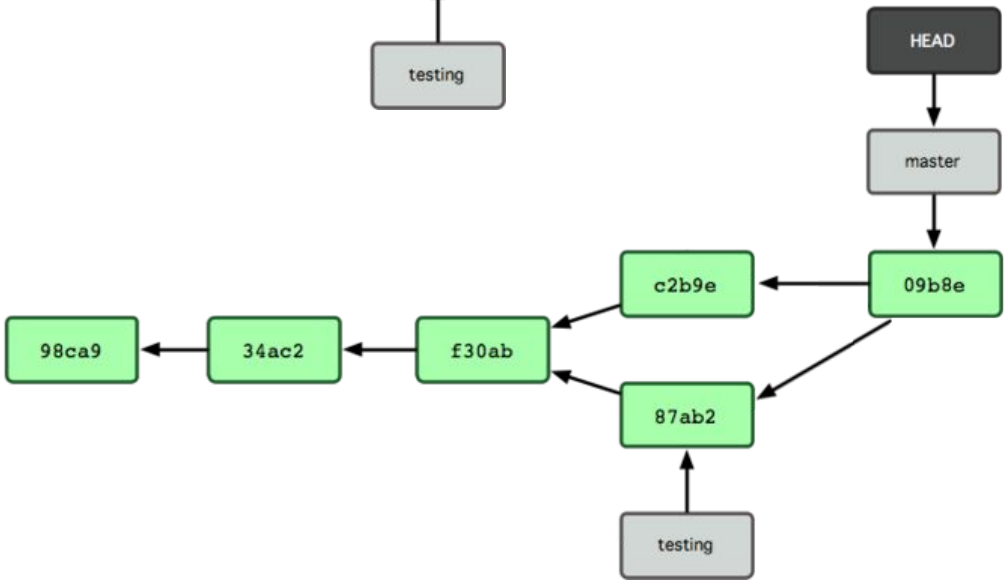
No conflicts

Merging changes into master

State after following git commands:
$ git checkout –b testing
Some editing …
$ git commit –am "Fix …"
$ git checkout master



Next you want to merge into master the changes in the branch "testing":
$ git merge testing



Hands on: Getting it back together

- We left with feature branch finished but not merged into the master. Good to double check the feature branch:
  - $ git checkout needTree
  - $ git status
- Everything OK, lets do the merge
  - $ git checkout master
  - $ git branch --no-merged
  - $ git merge needTree
  - $ git log --oneline # New feature in the master finally
  - This merge was done with the FAST FORWARD option
- We're done with the needTree branch, let's delete it
  - $ git branch –d needTree

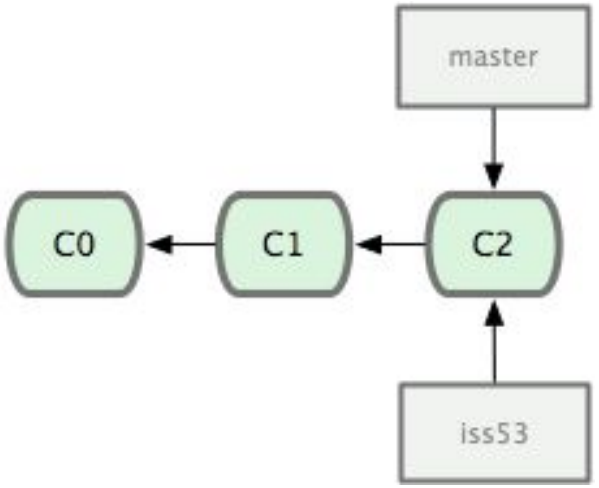
Merging urgent fixes

Putting current work aside for a moment..

Working on an issue in a branch

$ git checkout –b iss53

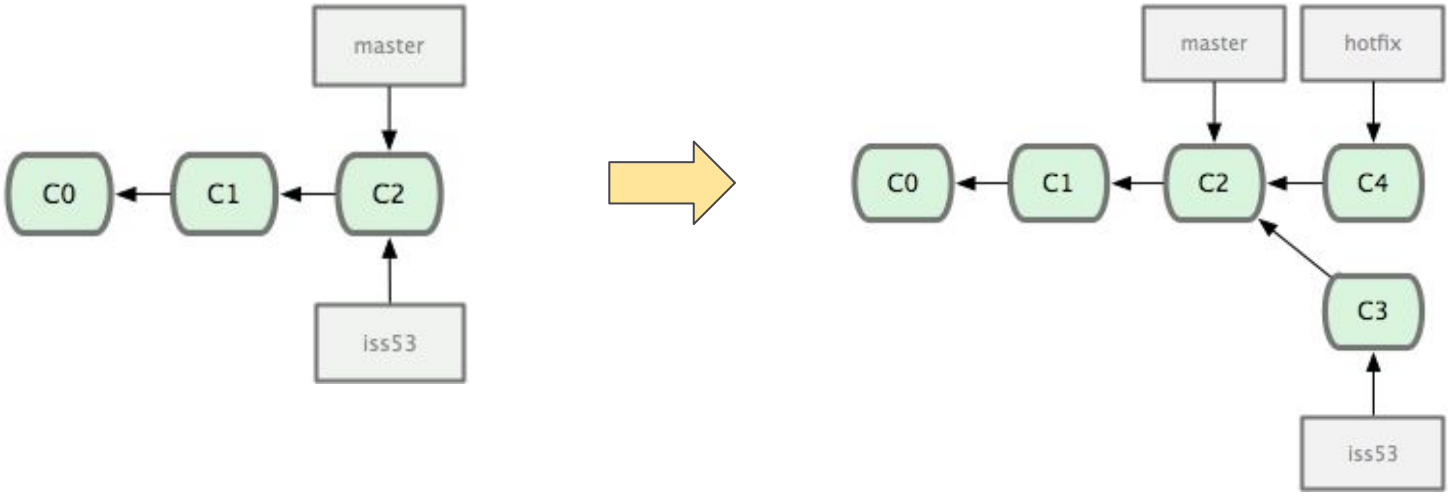
-> Edit some java class in NetBeans



git checkout -b # Creates branch AND checks out

Need to switch to an urgent fix

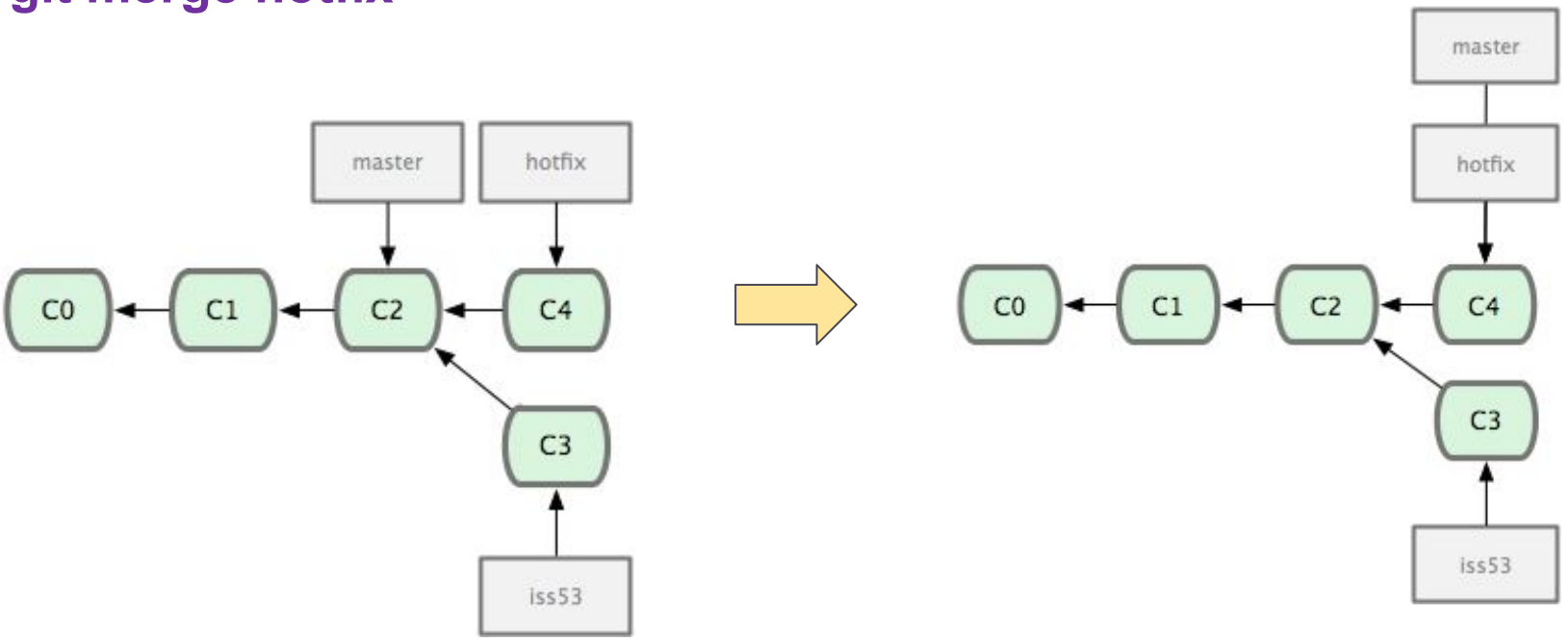
$ git commit –a –m “Fix Client constructor”
$ git checkout master
$ git checkout –b hotfix
$ vim index.html
$ git commit –a –m “Fix broken index.html”



NOTE: Git won’t let you switch to master if you have uncommitted changes that conflict with the code you’re checking out. So you should have a clean state before you switch (stash and amend can deal with this – advanced topic)

Now merge hotfix and master

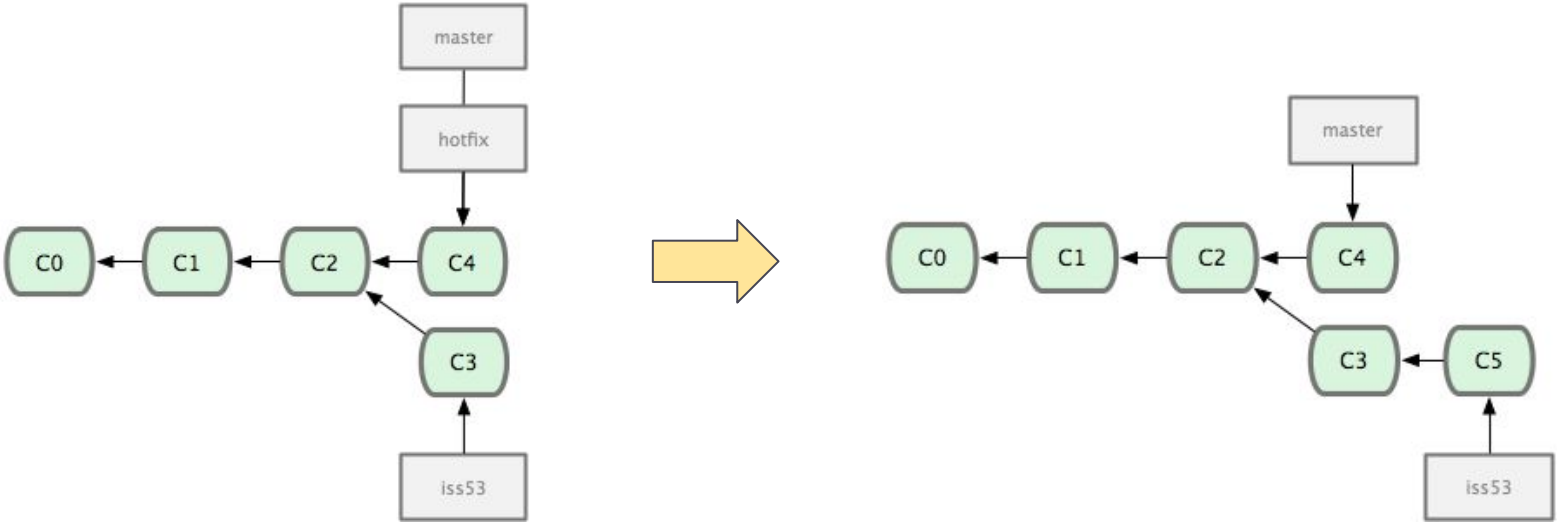
$ git checkout master
$ git merge hotfix



This will be a “fast forward” merge – because branch was directly upstream, git just moves pointer forward.

A little cleanup, then return to issue 53

$ git branch –d hotfix
$ git checkout iss53
$ vim index.html
$ git commit –a –m 'Refactor Client'



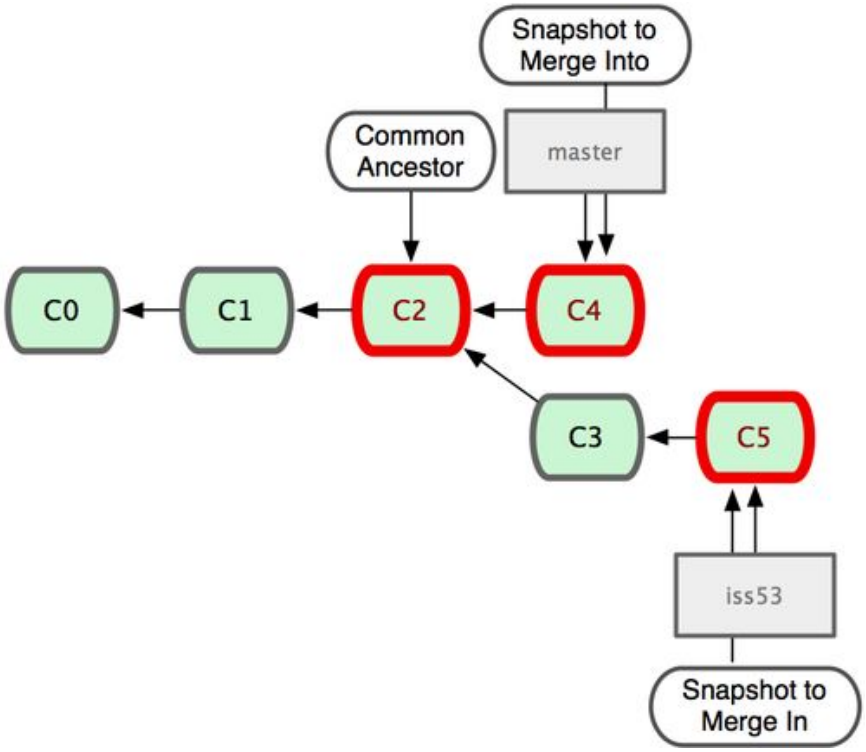
Be careful with $ git branch –d. OK in this example, just a duplicate pointer. May not always be the case.

Note that work done on hotfix is not part of iss53 branch.

Basic merge

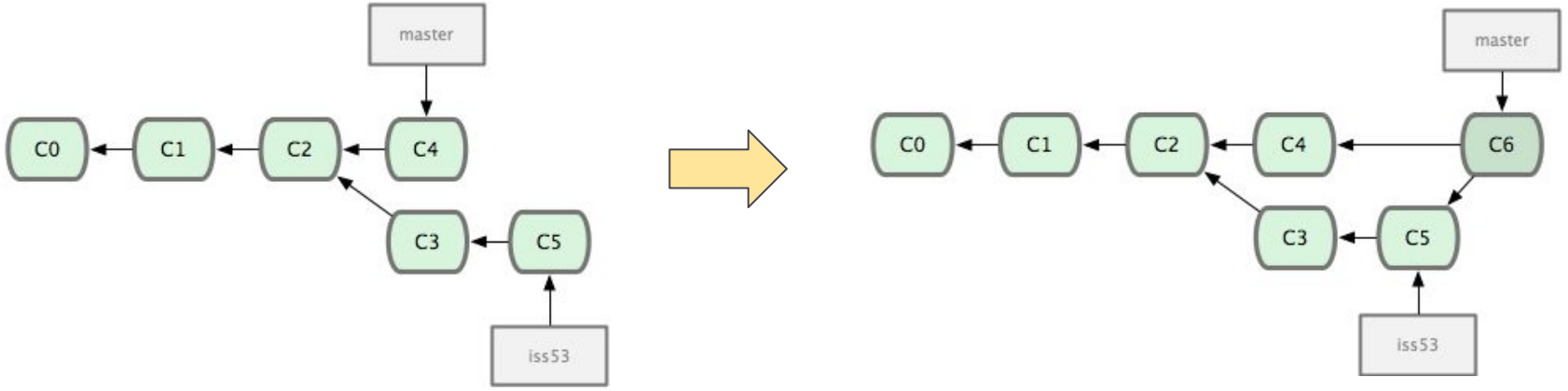
$ git checkout master
$ git merge iss53

Merge made by recursive.
README | 1 +
1 files changed, 1 insertions(+),
0 deletions(-)



git identifies the best common ancestor to use for merge

Basic Merge result



master now includes hotfix *and* iss53

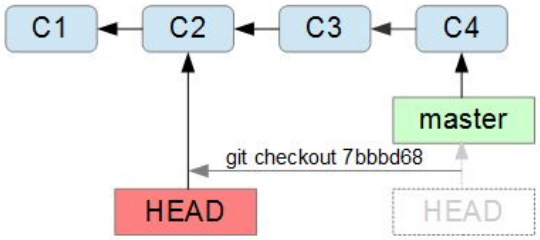
Remove iss53 if you want: $ git branch –d iss53

Advanced branching

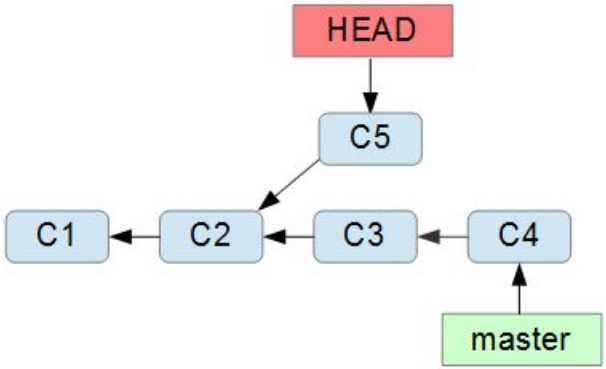
Based on earlier snapshot

Creating a branch from an earlier commit

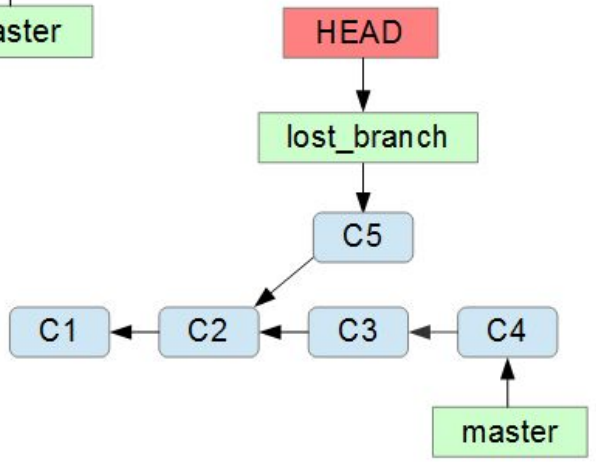
- $ git checkout [commit]
- If [commit] is not the latest, this will put you in the "detached HEAD state"
- git will warn you that if you will retain any changes you make you must create a new branch with command



- Edit something…
- $ git commit –am "Add …"

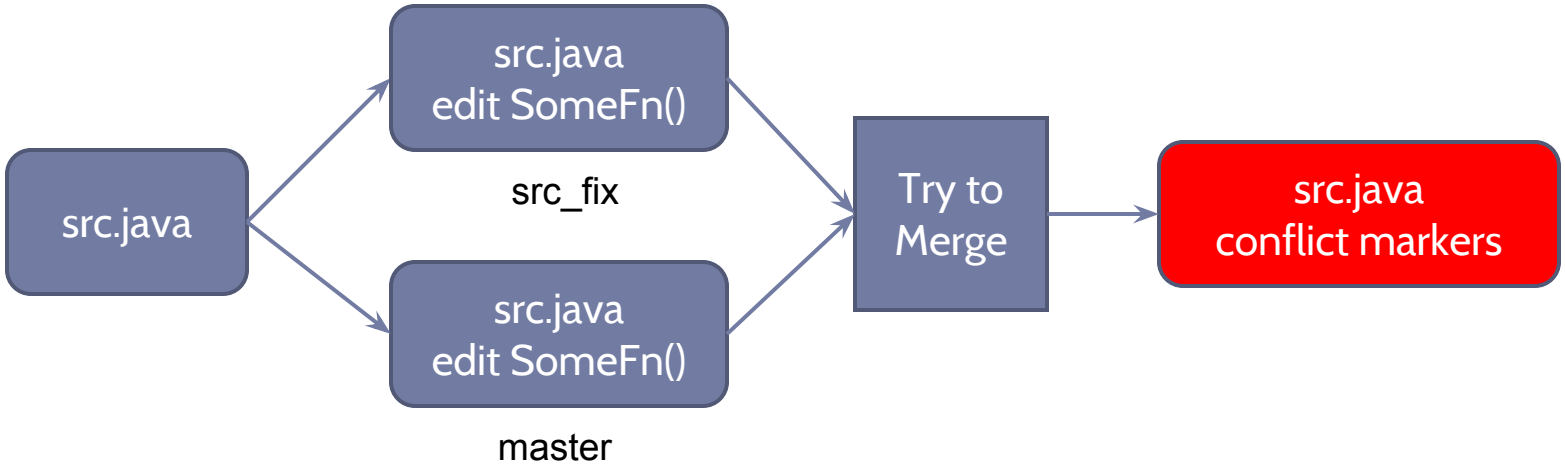


- $ git checkout -b <lost_branch>
- After this you can checkout master and merge



Images: http://pr0git.blogspot.fi/2015/03/git.html

Merging with conflicts




```
$ git checkout master
$ git merge src_fix
```

→

```
Auto-merging src/src.java
CONFLICT (content): Merge conflict in src/src.java
Automatic merge failed; fix conflicts and then commit the result.
```

What to do?

Edit files, removing all <<<<< and >>>>>> markers

```
$ git commit -am “Merge branch src_fix”
```

Visualizing Git Concepts

- Check out this website if you want to study visually the effects of git commands for committing and checking out snapshots, undoing changes and using remote server: http://onlywei.github.io/explain-git-with-d3/