Source Code Mgmt. Basics II

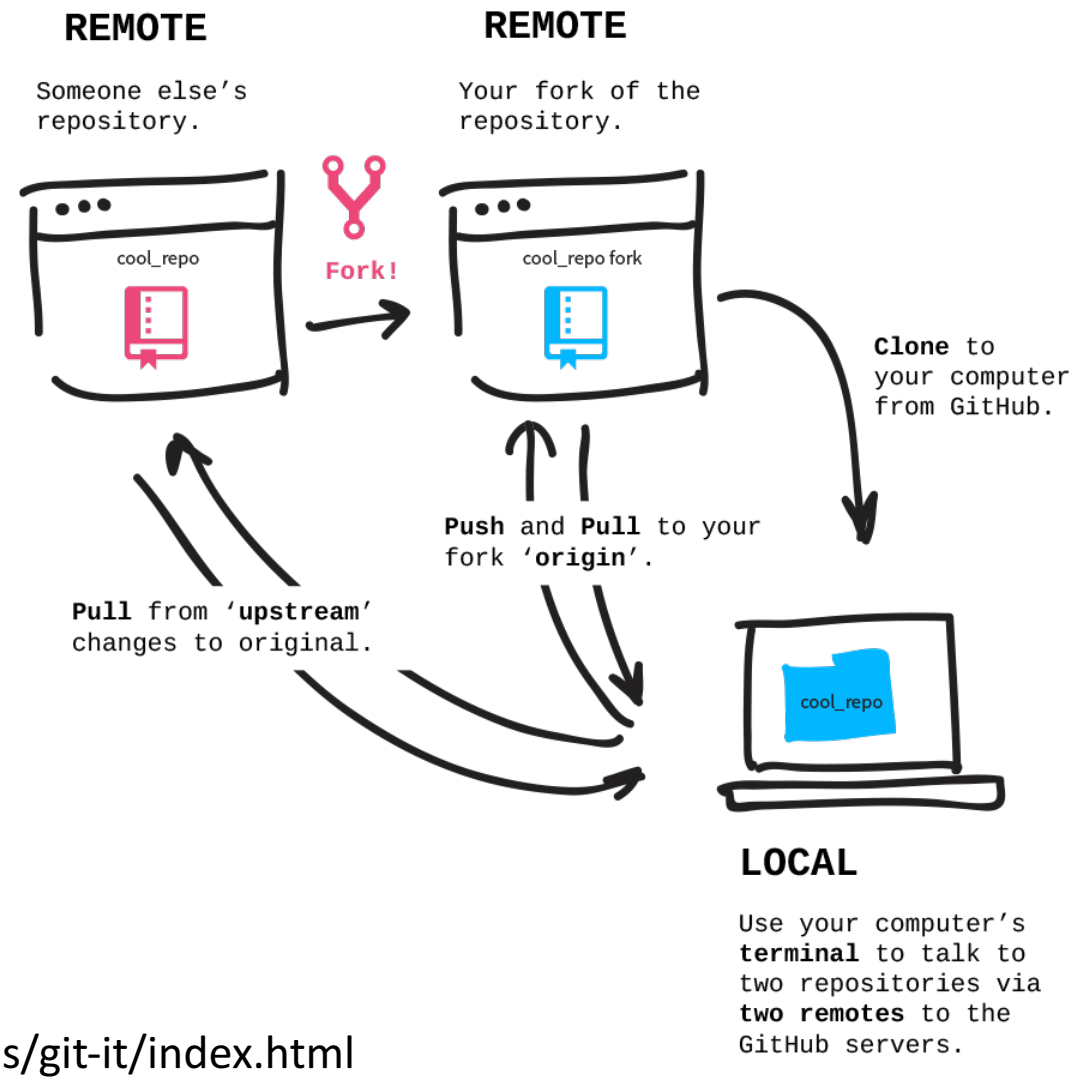
Harry J. Wang, Ph.D.

University of Delaware

Fall 2017

Work with Remotes

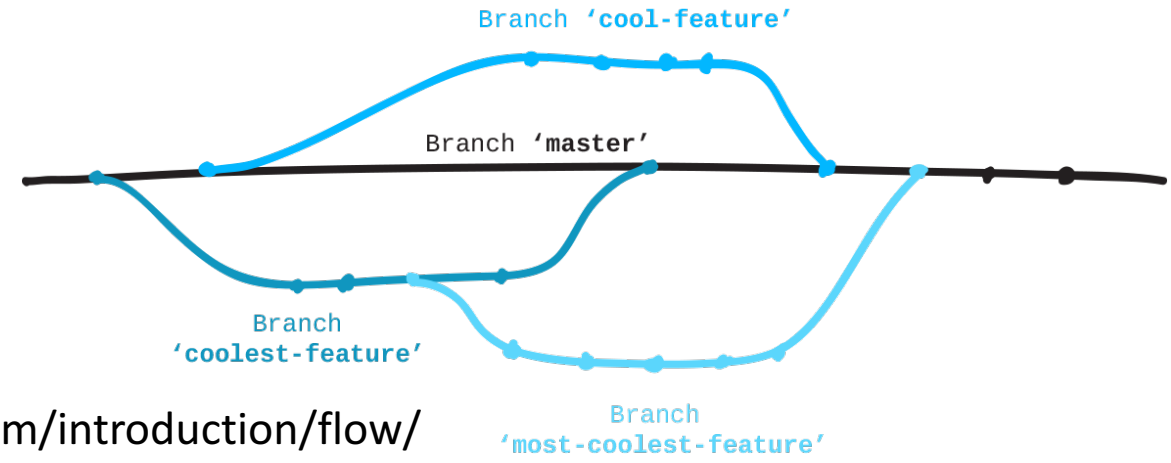
- Remote repositories are versions of your project that are hosted on the Internet
- Collaborating with others via managing these remote repositories and pushing and pulling data to and from them
- **upstream vs. origin**



source: http://jlord.us/git-it/index.html

Git Branching

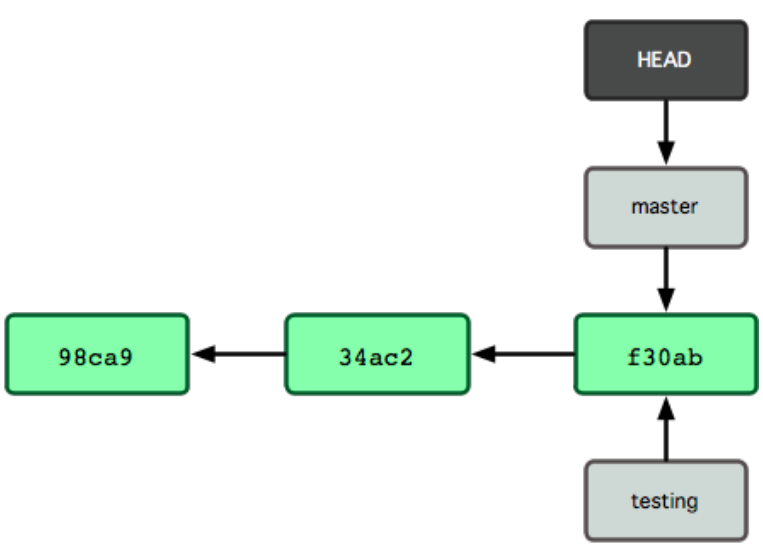
- By default, there is “master” branch: **anything in the master branch is always deployable!**
- During a project, there are a bunch of different features or ideas in progress at any given time.
- Branching allows developers to work on a new feature without messing with the deployable code in the main “master” branch

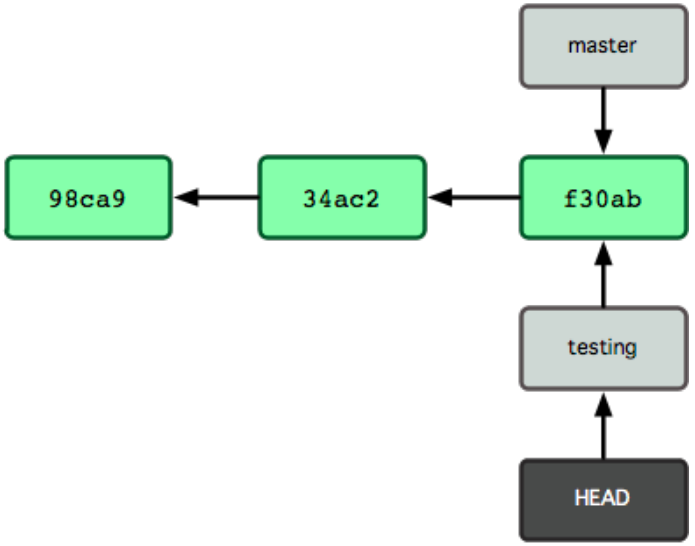


Source: https://guides.github.com/introduction/flow/

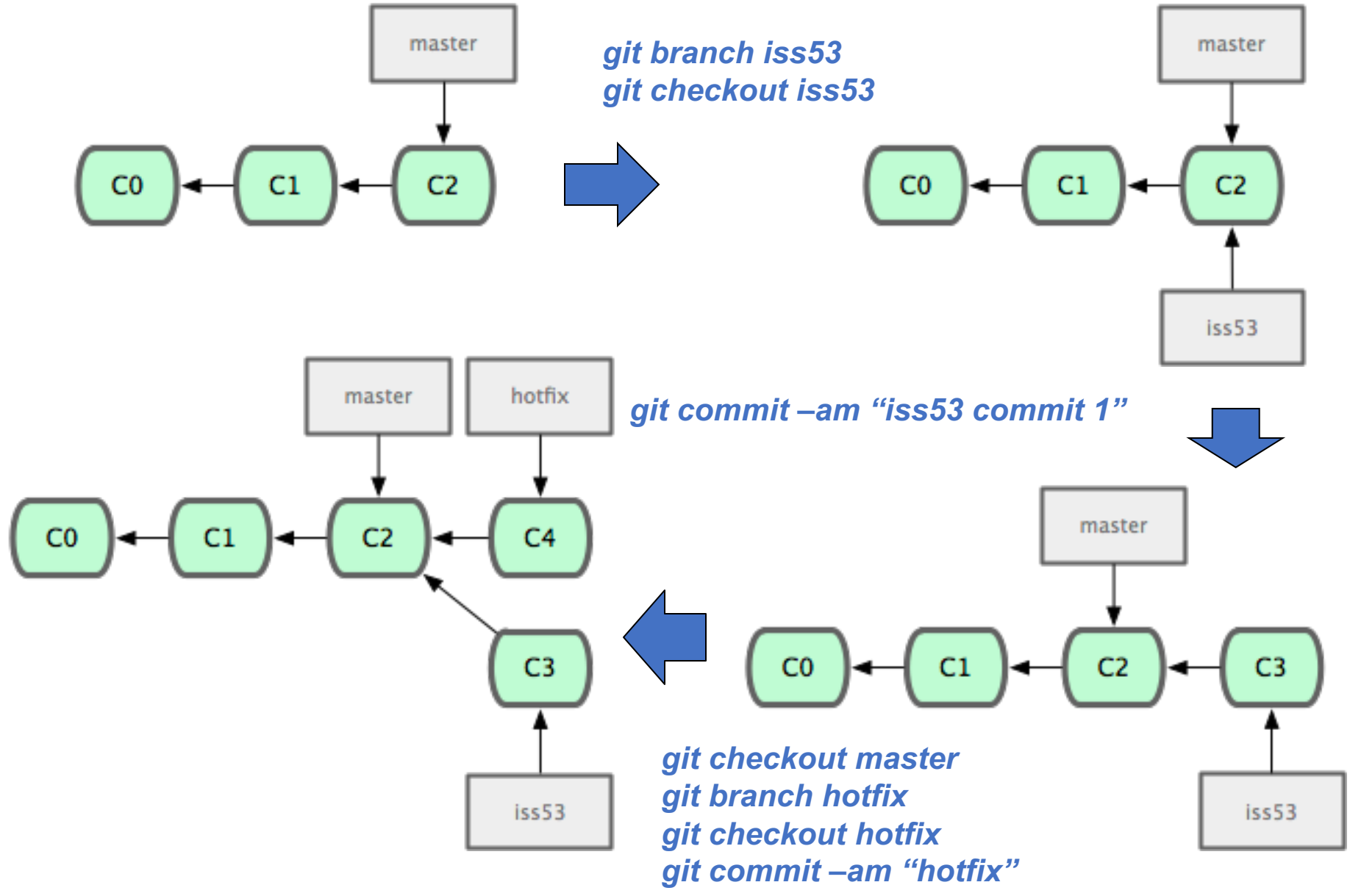
Create and Switch to Branches

- Create a new branch
  - git branch [branch-name], e.g. git branch testing
- HEAD indicates the current branch
- Switch branch
  - git checkout [branch-name]

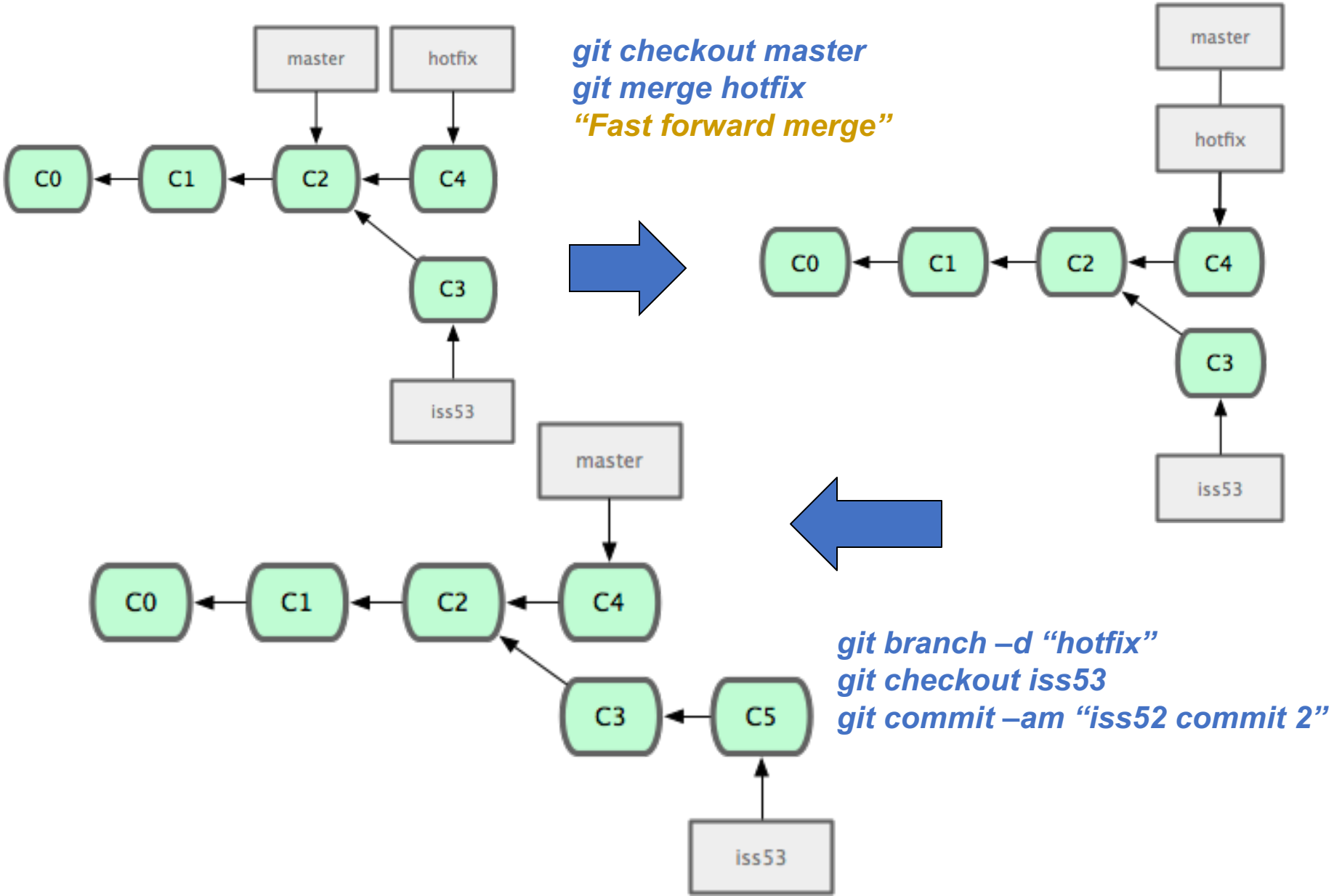




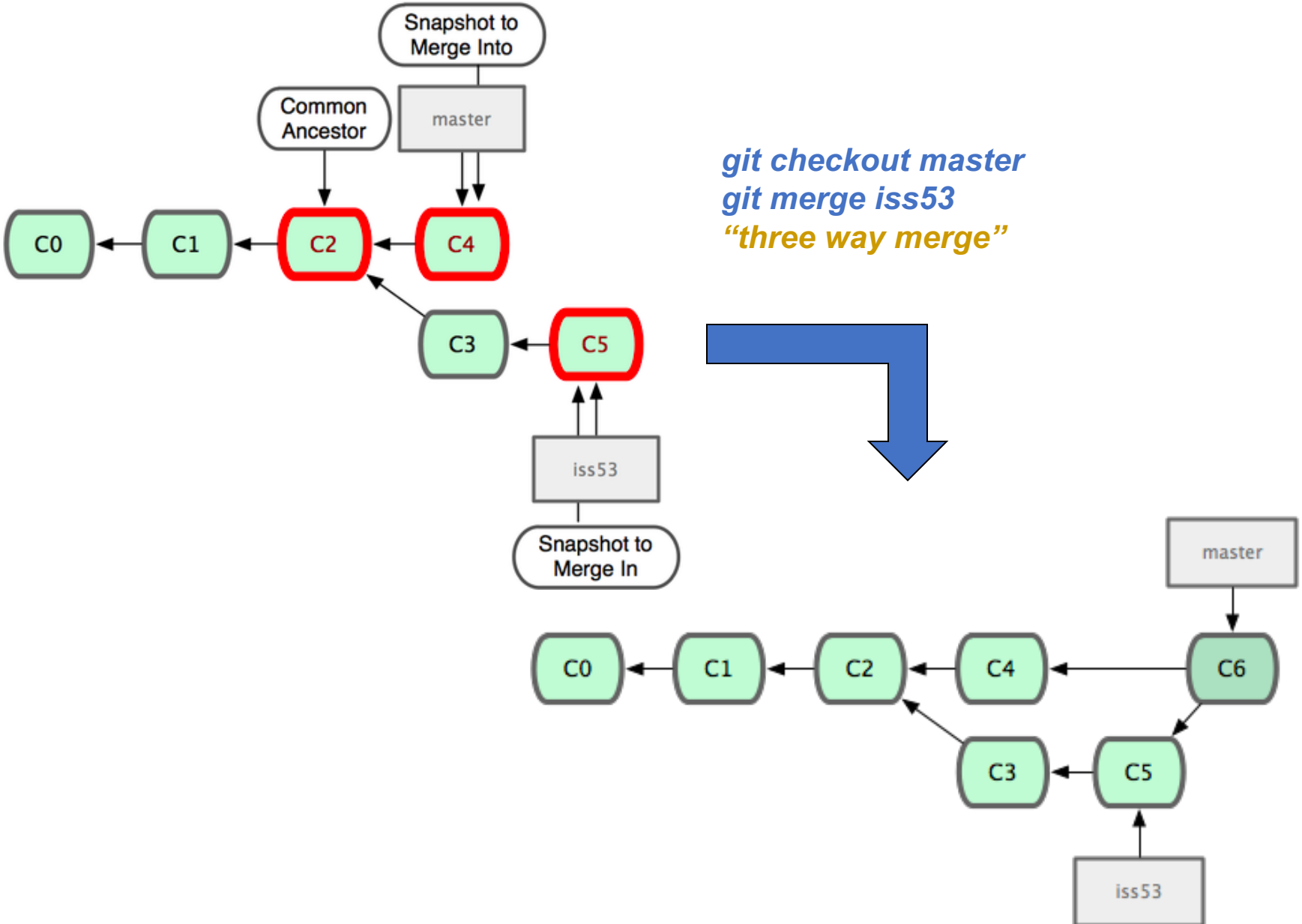
Branching and Merging Sample Workflow



Branching and Merging Sample Workflow



Branching and Merging Sample Workflow



Merge Conflicts and Resolution

- Merge conflicts occur when the same part of the file was edited in both merging branches
- Files with merge conflicts are marked as unmerged: git status
- Git adds standard conflict-resolution markers to the files so that you can open them manually and resolve those conflicts
- You can use a graphical tool to resolve these issues by running git mergetool

```
<<<<<<< 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
```

Code in head/master

Code in merging branch

Roll Back to a Previous Commit

- Use git log to see commit history

```
commit 08a6de19b71e314697147cc02800828d372bb90c
Author: Harry Wang <harryjwang@gmail.com>
Date:   Wed Nov 12 22:17:40 2014 -0500

    added two files
```

- Rename the current master branch:
  - git branch –m experiment
- Check out the good commit:
  - git checkout 08a6de
- Make the new master branch from the good commit
  - git checkout -b master
- Force push to remote
  - git push -f origin master

Pulling from a Remote

- If multiple people are working on the same project, additional changes may have been merged to the remote repo, while you work on your local files
- You can keep you local files updated with the changes by pulling from the remote:

  $ git pull <REMOTENAME> <BRANCHNAME>

Push to Remote

- Once you finish a feature locally, such as on a branch called feature-123", you need to push that branch to your forked repo:
  git push origin feature-123
- Then, you need to issue a pull request (PR) to notify the upstream repo admin to check on your code and merge your code if approved.
- Once your branch is merged into the upstream repo, you need to switch to the local master branch and pull the new changes from upstream:
  git checkout master
  git pull upstream master
- Update your forked repo master with the changes:
  git push origin master

Questions?

Undo Things

- Be careful: you can't always undo some of the "undos", which may end up losing some work if doing it wrong
- revise the previous commit: git commit –amend

  Example:

  $ git commit -m 'initial commit'

  $ git add forgotten_file

  $ git commit –amend
- Discard changes in working directory:
  git checkout -- <file>

Tagging

- Git has the ability to tag specific points in history as being important.
- People use this functionality to mark release points, v1.0, v2.0, etc.
- Lightweight tag: a simple pointer to a specific commit.
- Annotated tag: checksummed, more info on tagger name, e-mail, date, etc.
  - List all tags: git tag
  - Add a lightweight tag: git tag [tag-name]
  - Add an annotated tag: git tag -a [tag-name] -m ’tag-msg’
  - Show tag: git show [tag-name]
  - Share tag: git push [remote-name] [tag-name]