Version Control with Git and GitHub

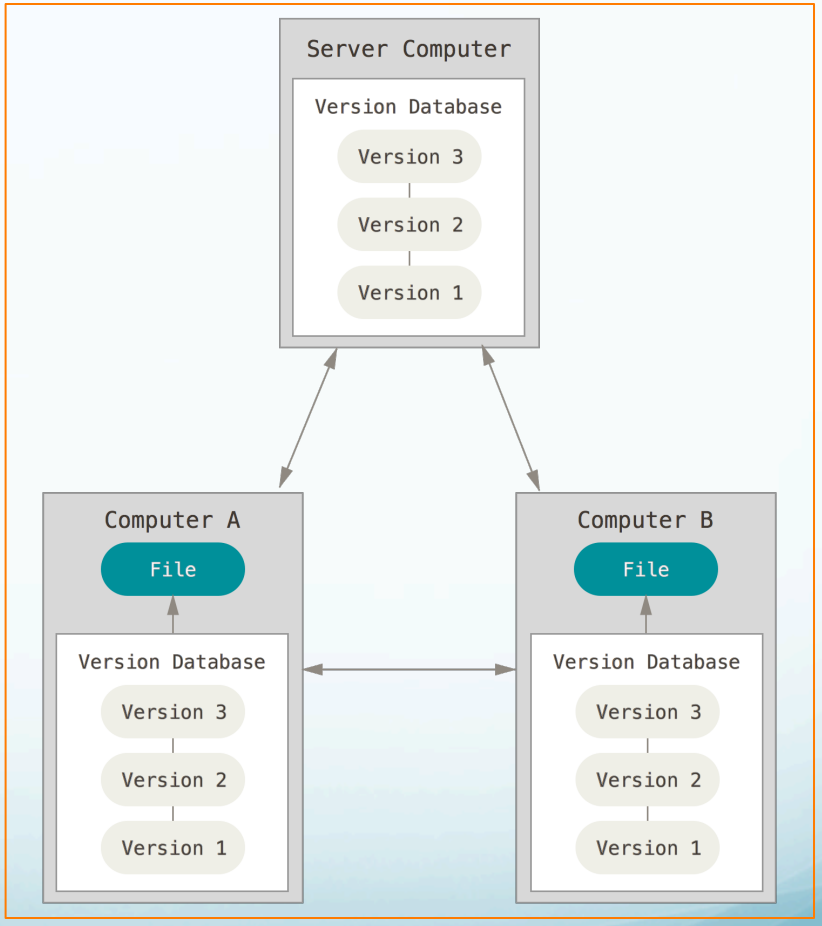
CDIPS Pre-Workshop Week
Monday, July 6, 2015
Michael Cole

Today's Outline

- Introduction to Git
- Getting Started (Installation / etc.)
- Basic Git Commands
  - Repository Basics
  - Making Changes
    - Not covered: *Rebasing*
  - Branching and Merging
  - Remote Repos
- Introduction to GitHub
- Cloning a GitHub Repo

Introduction to Git

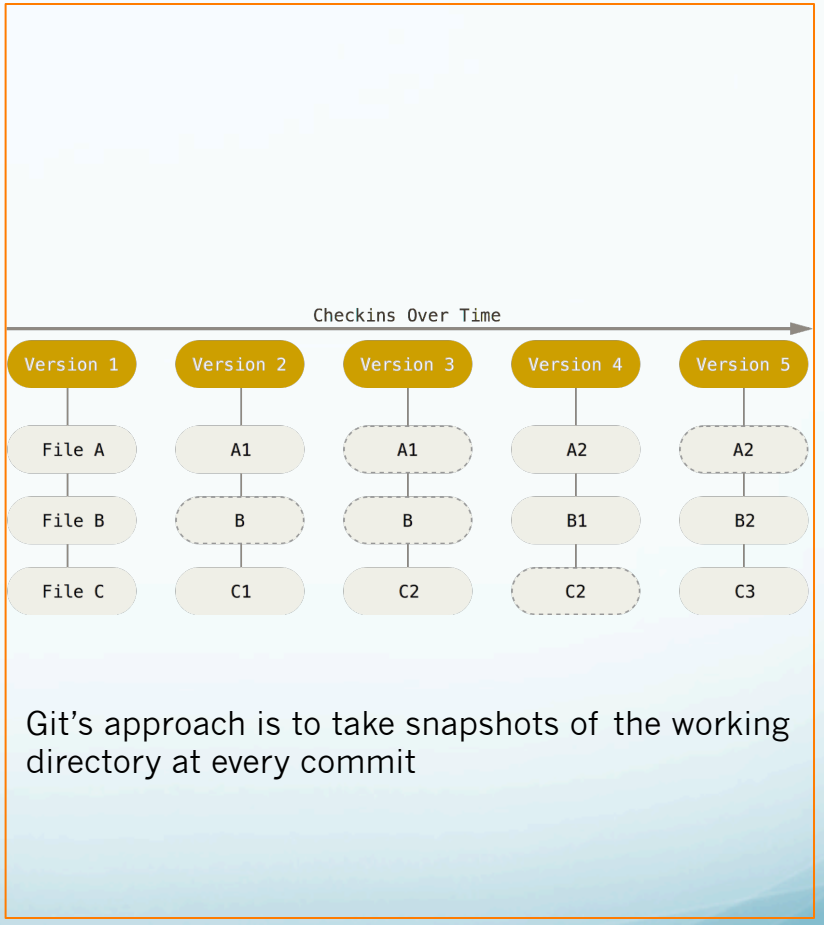
- What is **version control**?
  - "Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later."
  - "A VCS [Version Control System] allows you to: revert files back to a previous state, revert the entire project back to a previous state, review changes made over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more."
- http://git-scm.com/book/en/v2/Getting-Started-About-Version-Control



Distributed Version Control System

Introduction to Git

- What is **git**, in a nutshell?
  - Version control
  - Fast
  - Simple
  - Strongly supports non-linear development (thousands of branches)
  - Fully distributed
  - Able to handle large projects efficiently
- http://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git



Git's approach is to take snapshots of the working directory at every commit

Set up Git + Github

- Set up git on your personal computer:
  - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
  - Windows users may want to install the GitHub application:
    - https://windows.github.com/
- Configure git:
  - https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup
- Set up GitHub account:
  - https://github.com/join

Command Line Git

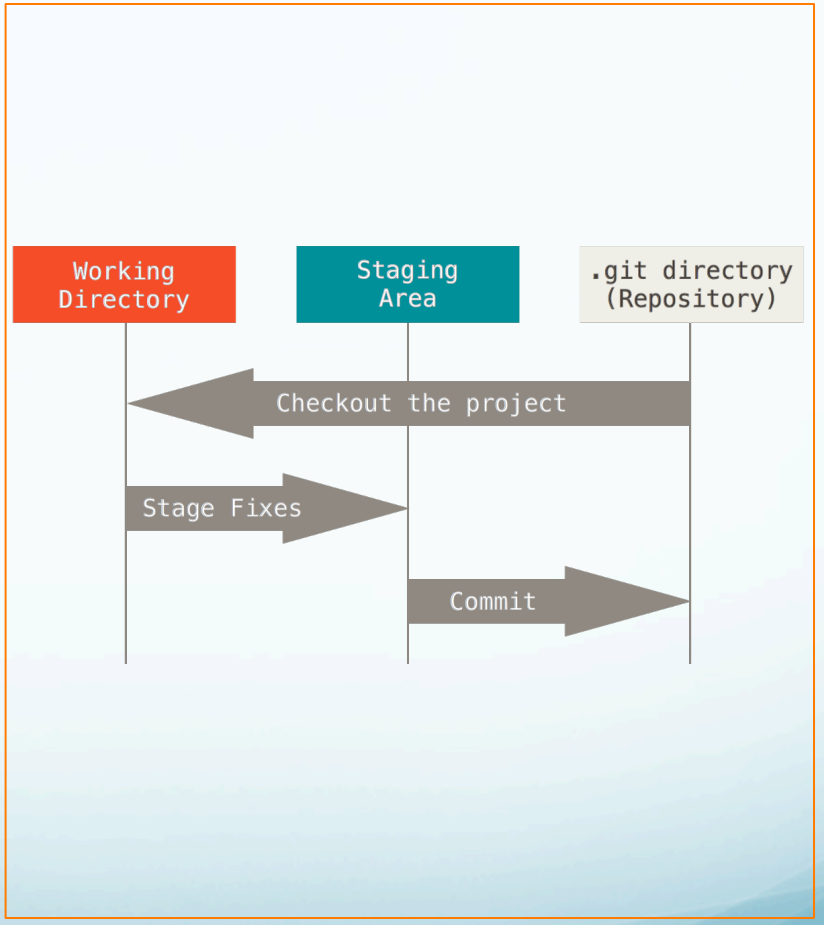
- Navigate inside the git working directory
  - Contains .git directory
  - Use git commands:
    - git <verb>
- Or use the –C flag to run git commands outside of the working directory:
  - git –C <path> <verb>

Repository Basics

- git init:
  - Initialize git repository in current (or named) directory
- git config -l:
  - List configured git/repo parameters
- git status:
  - Check status of the repository
- git log:
  - See trace of commits
  - --graph
    - Visualize commit tree

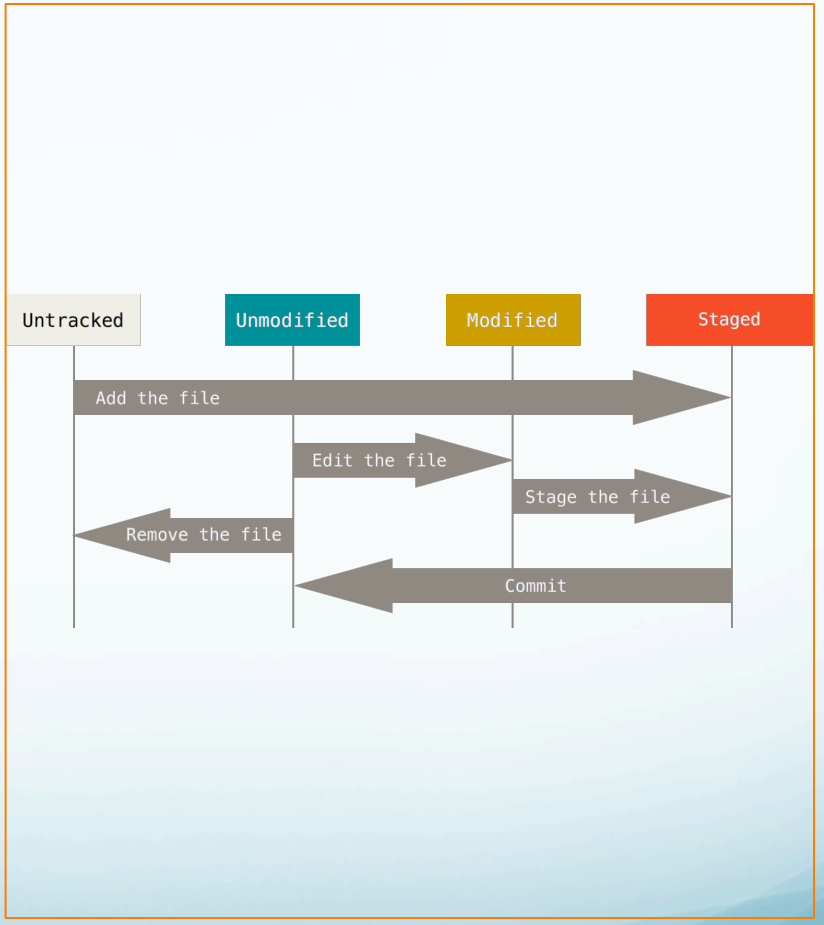
Git Commit Structure

- We use git in three stages:
  - Working Directory
    - Modified Data
  - Staging Area (Index)
    - Staged Data
  - Git Directory
    - Committed Data
- https://git-scm.com/book/en/v2/Getting-Started-Git-Basics



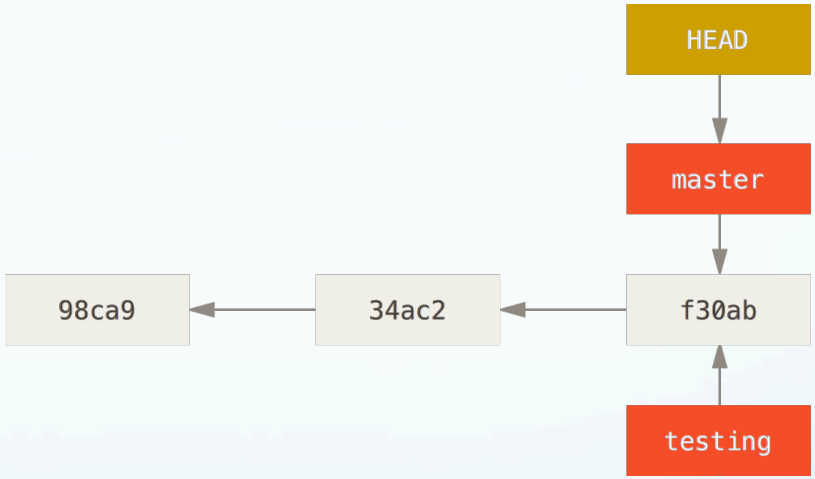
Making Changes

- git add <file>:
  - Stage modified (or new) files for commit
- git rm <file>:
  - Stage file removal for commit
- git mv <file>:
  - Stage file for rename
- git reset HEAD:
  - Unstage all changes
- git commit:
  - Commit staged changes
  - -m for quick comment
  - -a to skip staging!
- git reset HEAD~1
  - Undo last commit (consider --soft or --hard)



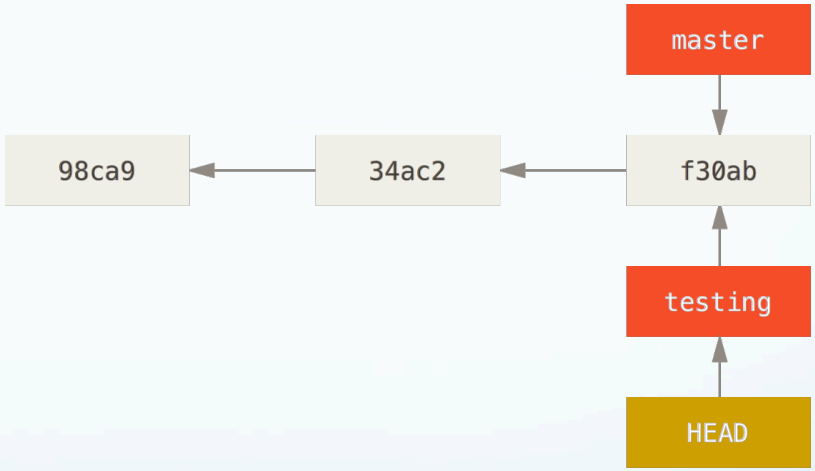
Basic Branching

- What is a **branch** in git?
  - A branch is a pointer to a particular commit (snapshot of the repository)
  - The **master** branch is the *canonical* distribution branch (shared and up-to-date)
- https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell



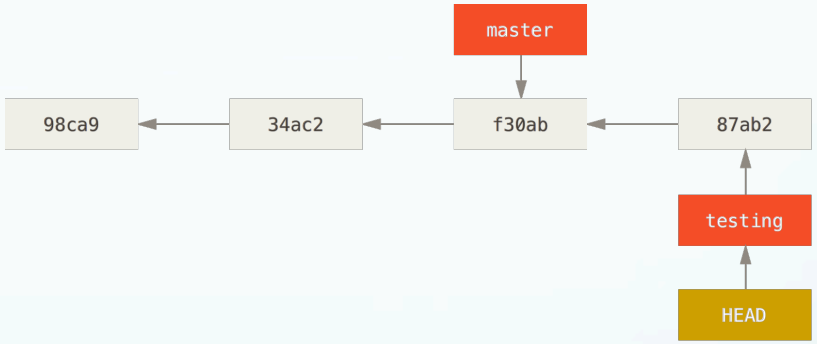
Basic Branching

- What is a **branch** in git?
  - A branch is a pointer to a particular commit (snapshot of the repository)
  - The **master** branch is the *canonical* distribution branch (shared and up-to-date)
- https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell



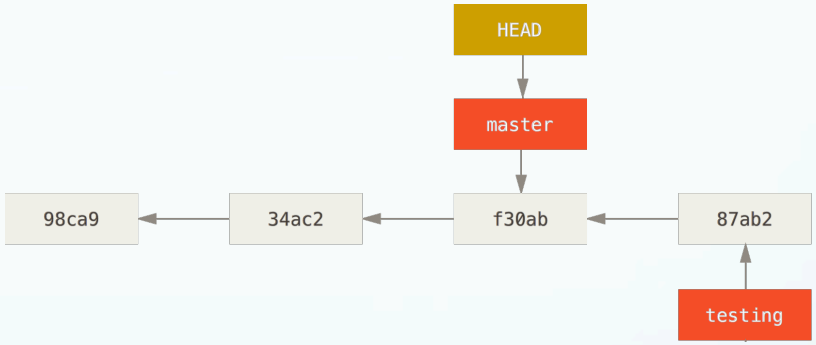
Basic Branching

- What is a **branch** in git?
  - A branch is a pointer to a particular commit (snapshot of the repository)
  - The **master** branch is the *canonical* distribution branch (shared and up-to-date)
- https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell



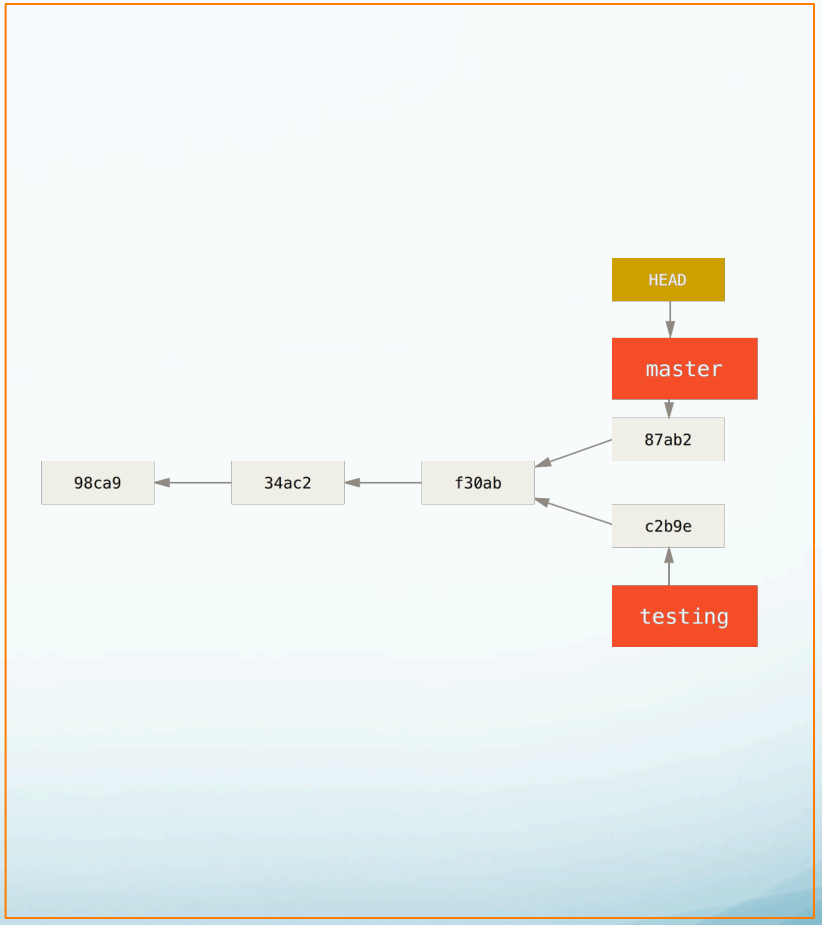
Basic Branching

- What is a **branch** in git?
  - A branch is a pointer to a particular commit (snapshot of the repository)
  - The **master** branch is the *canonical* distribution branch (shared and up-to-date)
- https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell



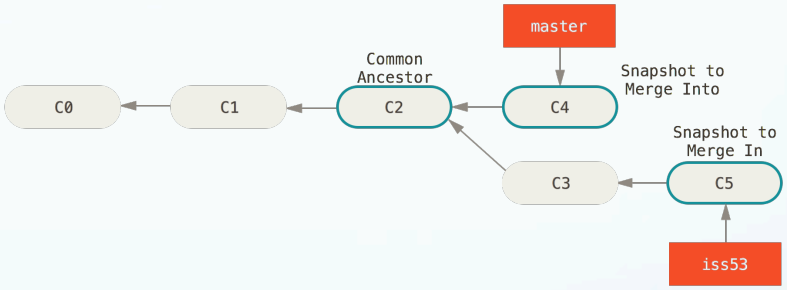
Basic Branching

- What is a **branch** in git?
  - A branch is a pointer to a particular commit (snapshot of the repository)
  - The **master** branch is the *canonical* distribution branch (shared and up-to-date)
- https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell



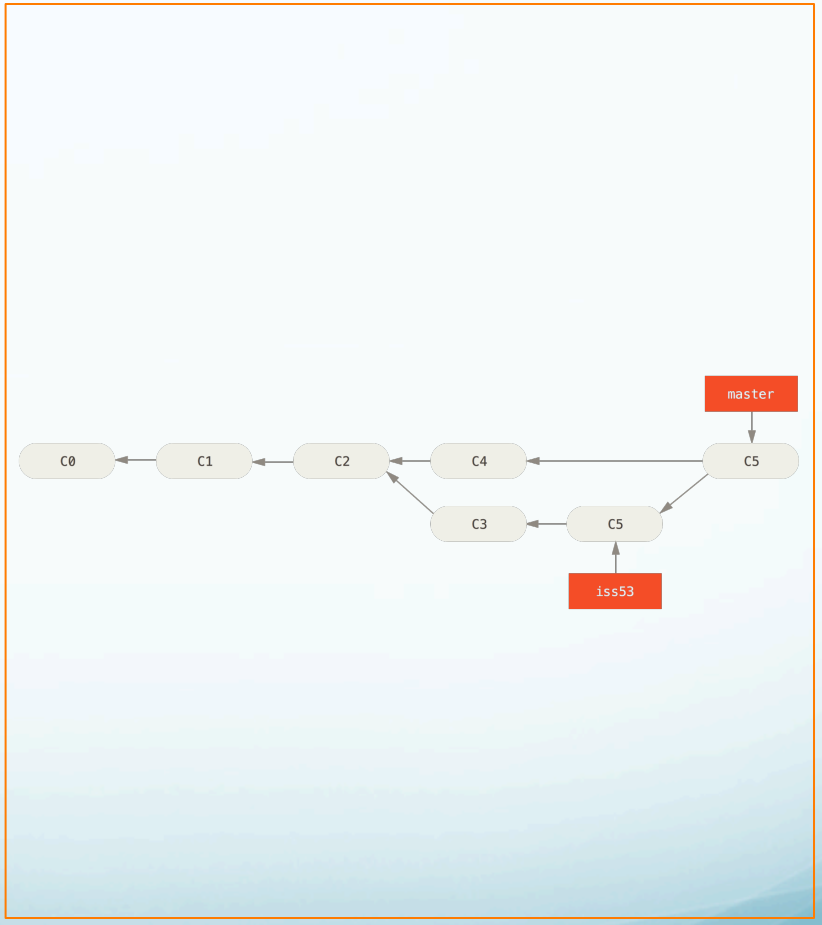
Basic Merging

- What is a **merge** in git?
  - A merge is a commit that includes changes made in two branches.
  - If the same data was edited in both branches, the merge must be resolved by a user.
- https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging



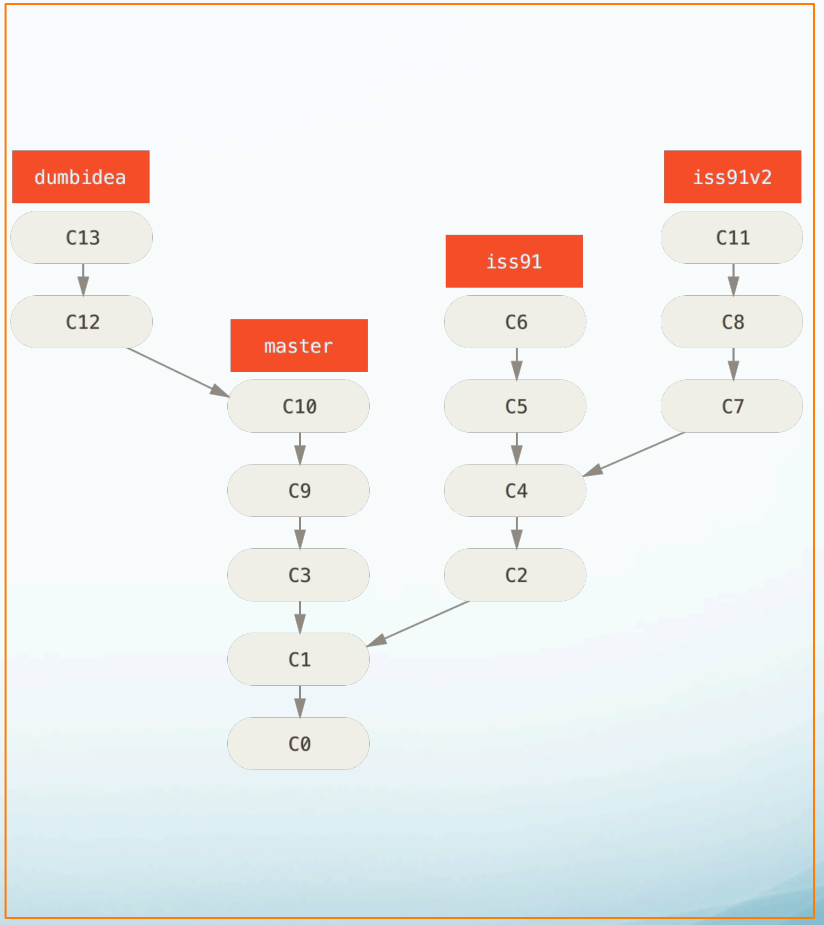
Basic Merging

- What is a **merge** in git?
  - A merge is a commit that includes changes made in two branches.
  - If the same data was edited in both branches, the merge must be resolved by a user.
- https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging



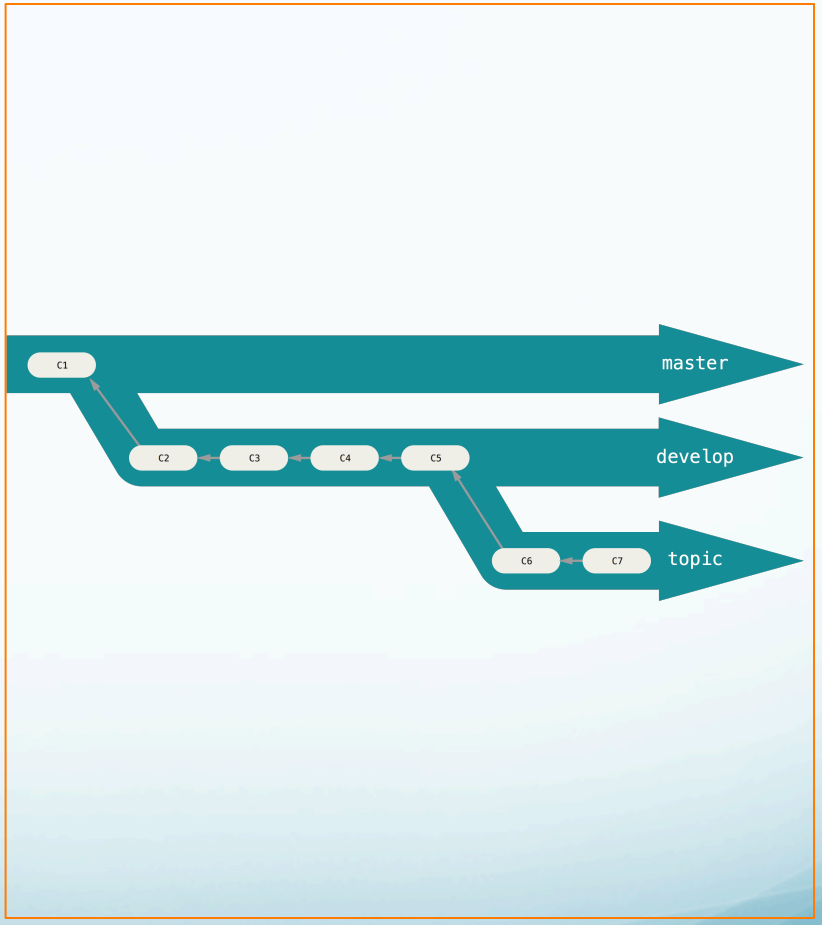
Branching and Merging

- git branch <branch>:
  - Create new branch at current (or earlier) commit
- git checkout <branch>:
  - Decompress branch to working directory – move to it
  - Can be used to decompress select files from other branches
- git branch -d <branch>:
  - Delete branch
- git merge <branch>:
  - Merge with other branch



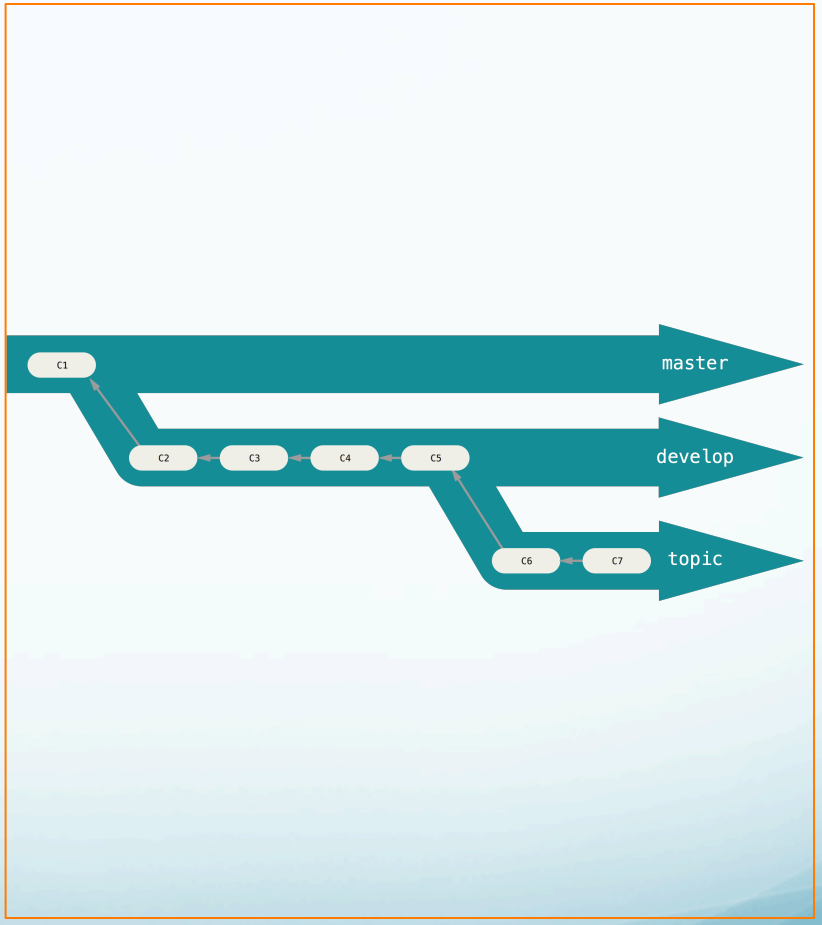
Merging: When and How?

- Consider a master branch and a dev branch
- When:
  - The master branch has been updated with useful tools that you want to use for your own development
  - Fixes have been committed to core code in the master branch
  - You want to stay up-to-date...
- Merge branch master into dev
  - git checkout dev
  - git merge master



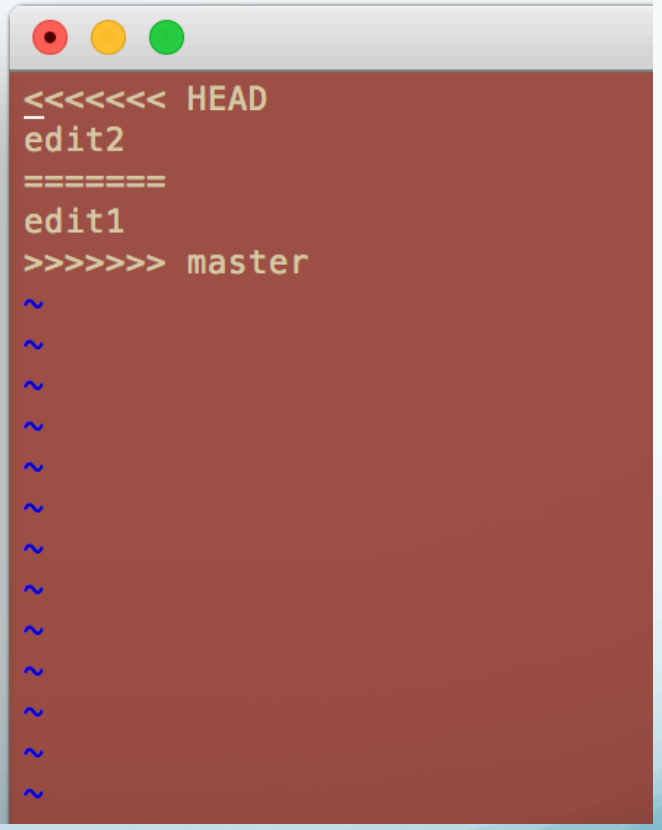
Merging: When and How?

- Now suppose:
  - You want to add code produced in the dev branch to the master branch
  - You've made edits you want to add to the master branch
- Merge branch dev into master
  - git checkout master
  - git merge dev
- Merge resolution effects everyone on master!



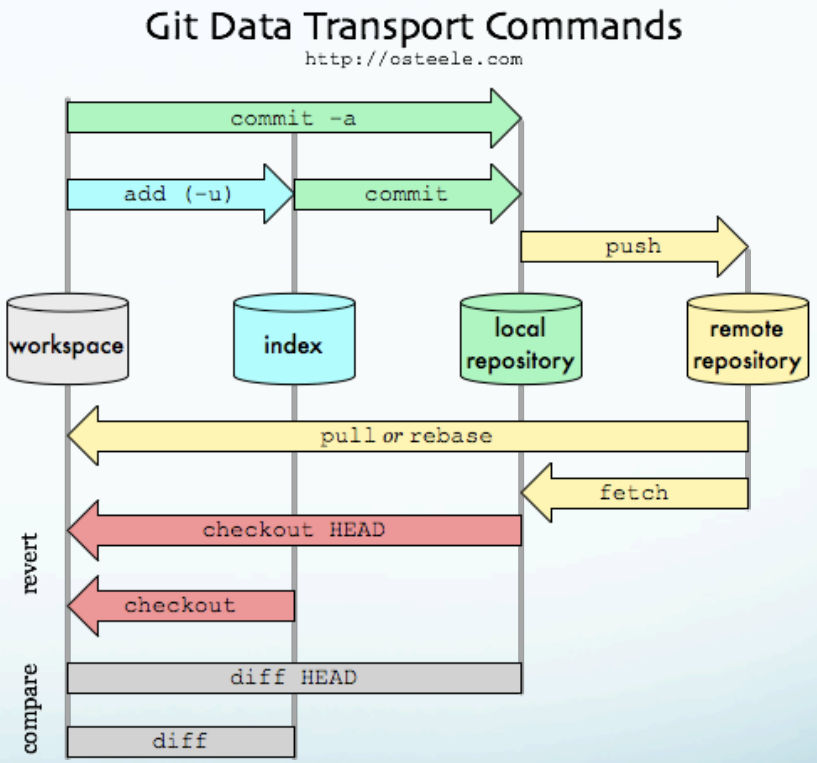
!!!CONFLICT!!!

- **Don't panic!**
  - Well.. not right away...
- Look for conflict resolution markers
  - <<<<<<<, =======, >>>>>>>
  - Remove markers and resolve conflict manually
  - stage and commit – done!
- git mergetool
- Best way to avoid conflicts:
  - Work on different files, or different modules within the same file
- Give up? Just abort!
  - git merge --abort



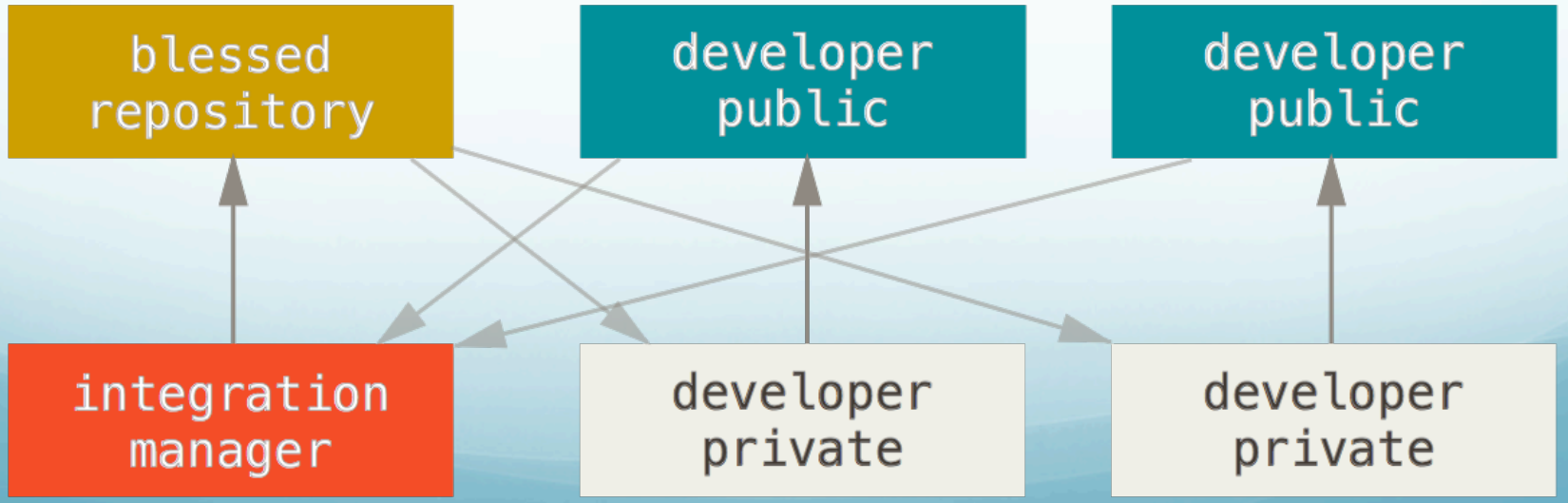
Remote Repos

- git clone <url>:
  - Clone a remote repository
- git fetch:
  - Fetch all changes from remote repo
  - Current branch must track a remote branch, often “origin master”
- git pull:
  - Fetch and merge
- git push:
  - Push commits to remote repo



GitHub: An Integration-Manager Workflow

- A **fork** is a personal copy of another user's repository that lives on your account.
- **Pull requests** are proposed changes to a repository submitted by a user and accepted or rejected by a repository's collaborators.



Clone a GitHub Repo