CS35L – Spring 2019

Slide set:	9.2
Slide topics:	Git – Branching, Merging, Remotes
Assignment:	9

Why Branching?

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

What Is a Branch?

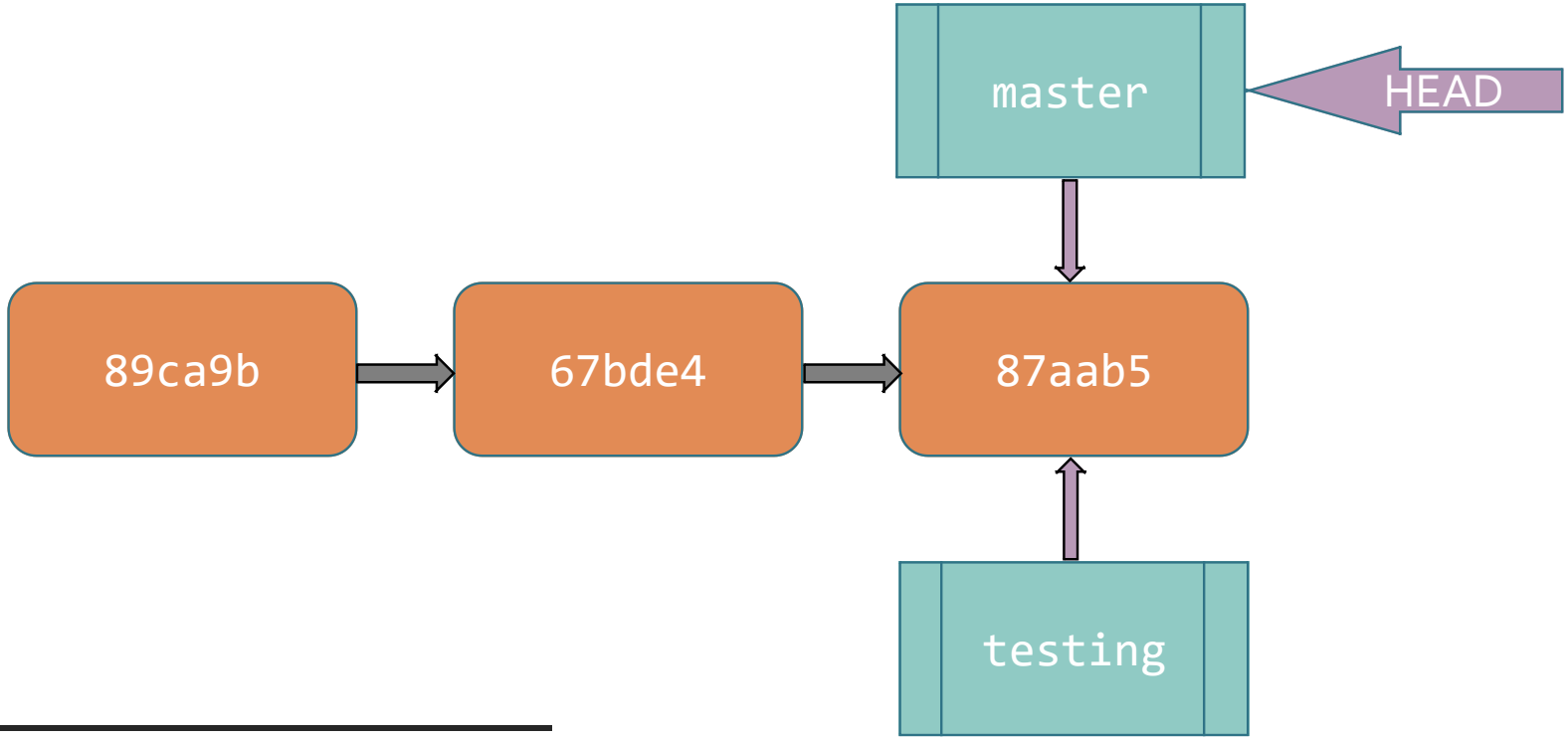
- A pointer to one of the commits in the repo (HEAD) + all ancestor commits
- When you first create a repo, are there any branches?
  - Default branch named ‘master’
- The default master branch
  - points to last commit made
  - moves forward automatically, every time you commit

New Branch

- Creating a new branch = creating new pointer

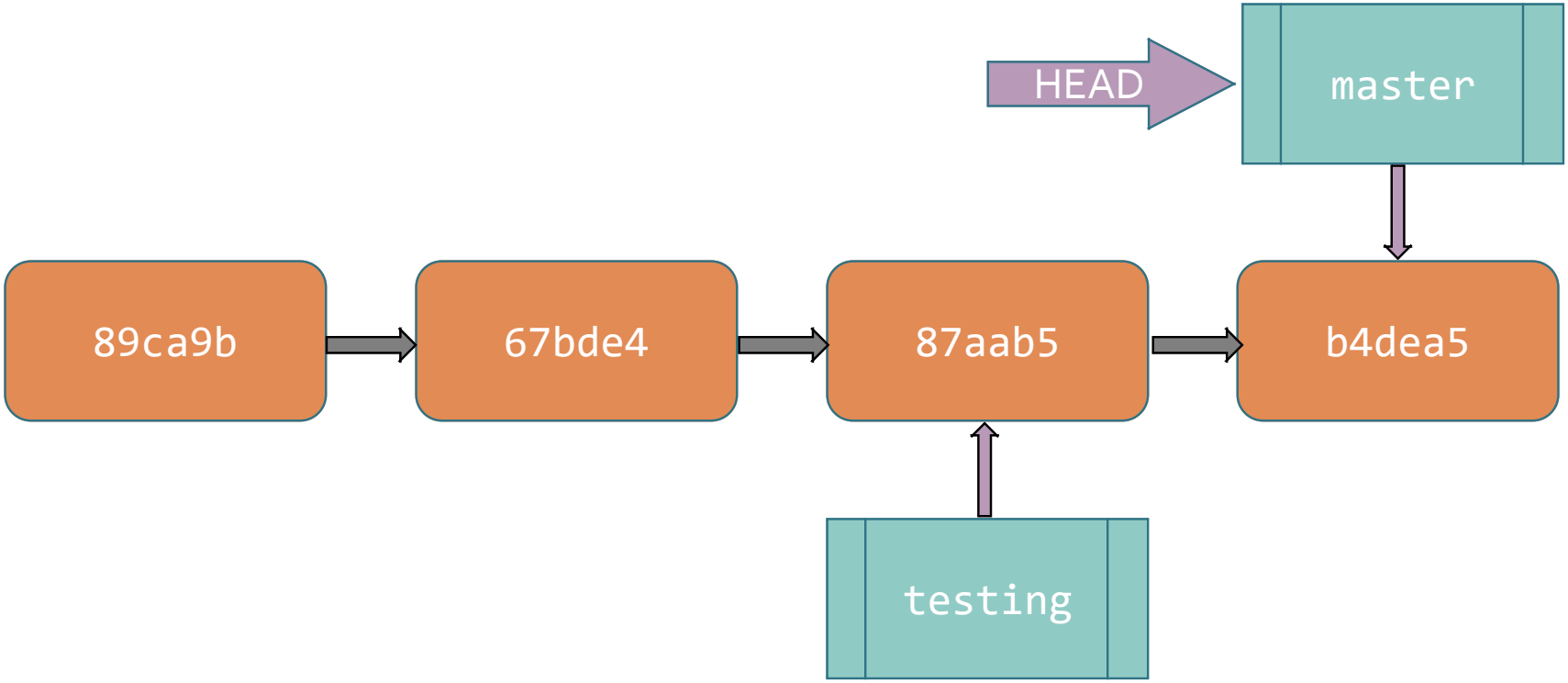
  `$ git branch` testing

  – Created at current commit



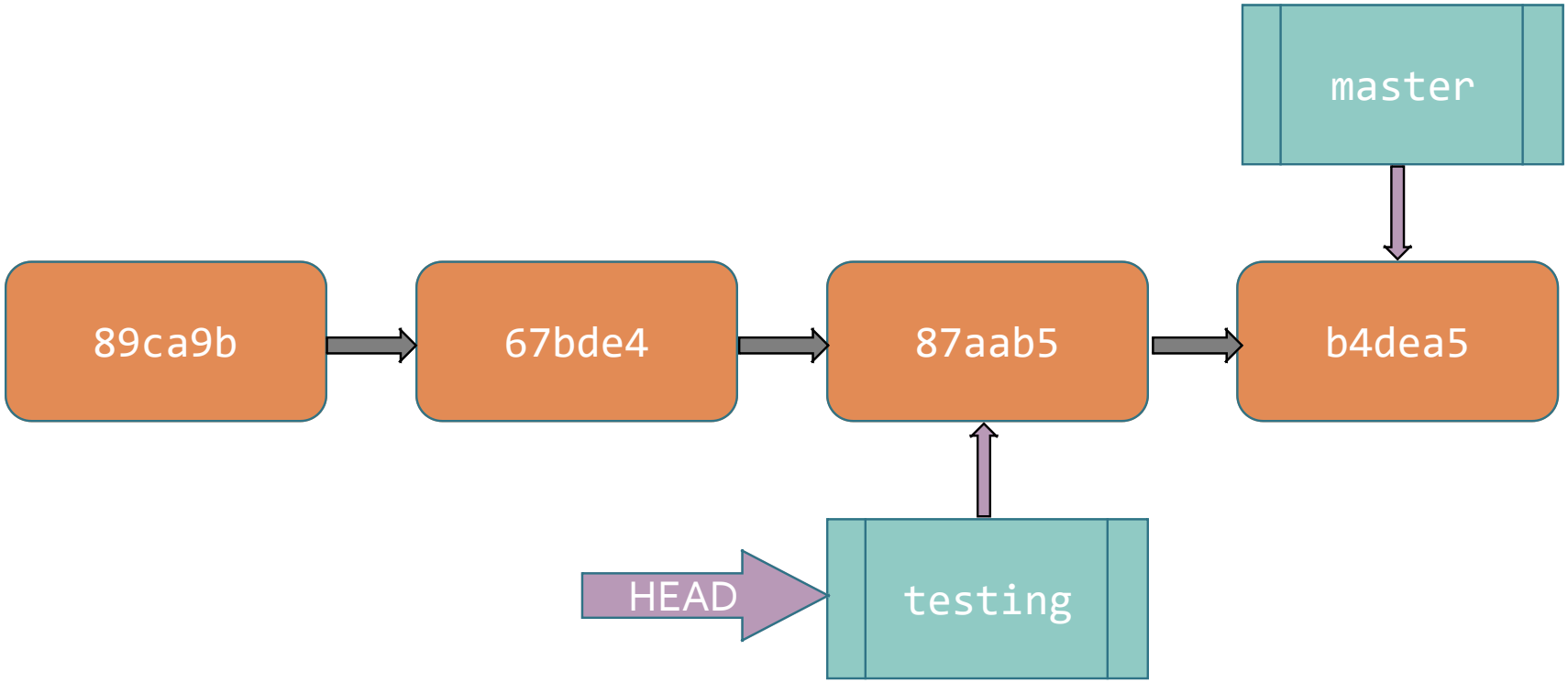
New Commit

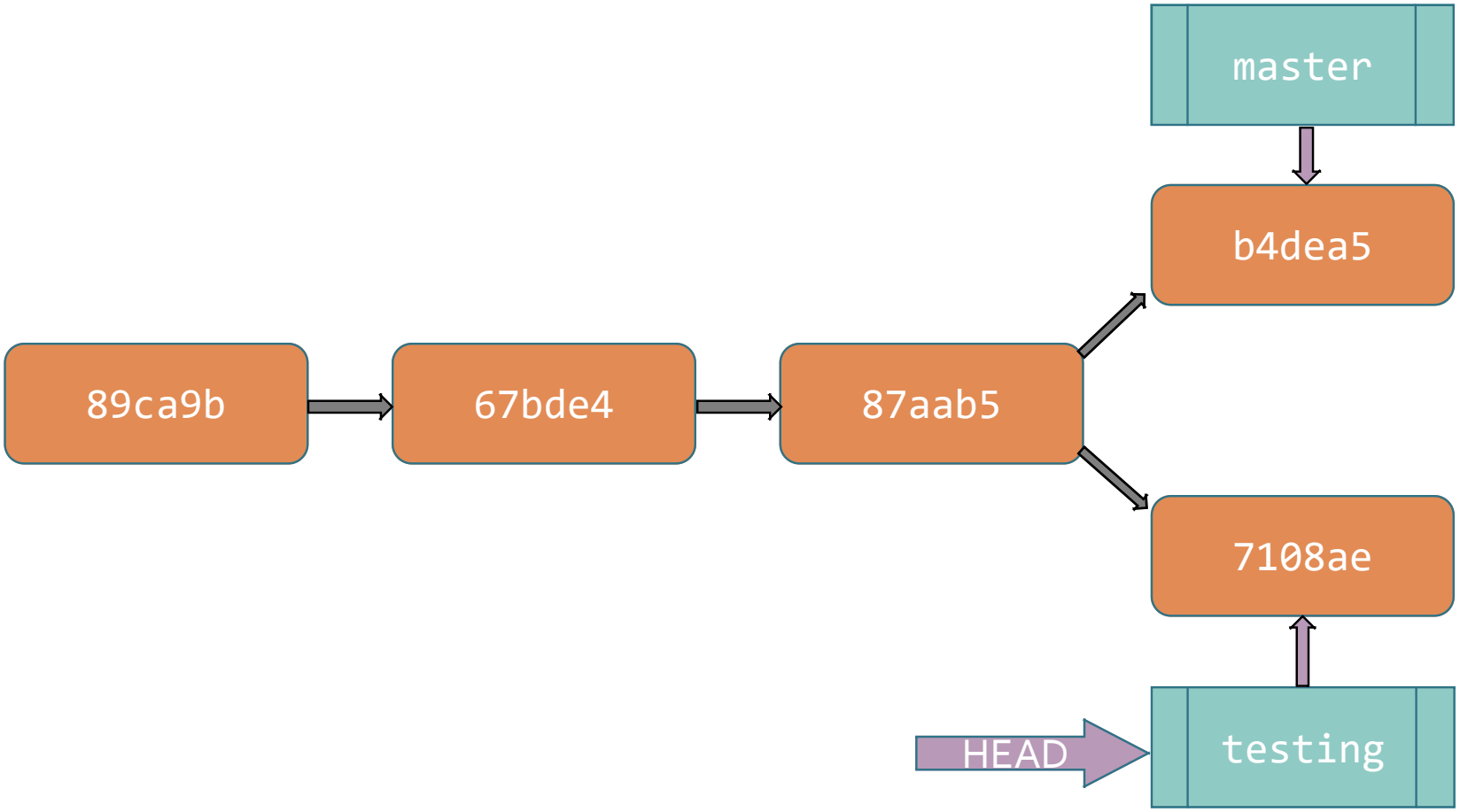
- What happens if we make another commit?



Switching to New Branch

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





Commit After Switch

Working With Branches - Commands

```
$ git branch test
```

Create new branch

```
$ git branch
```

Lists all branches

```
$ git checkout test
```

Switch to test branch

```
$ echo “hello world!” > hw
$ git commit –a –m “Hello”
```

Commit the change in new branch

```
$ git checkout master
```

Back to master branch

```
$ git log
$ git merge test
```

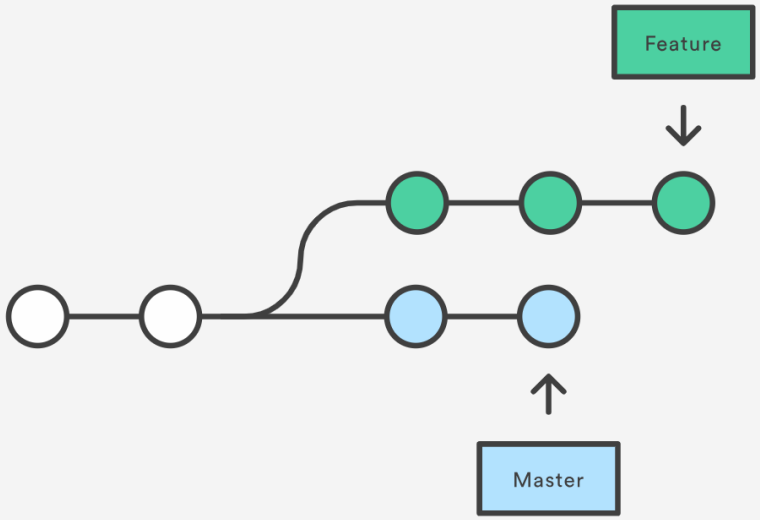
Merges commits from test branch to current branch (here master)

Git - integrating changes

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

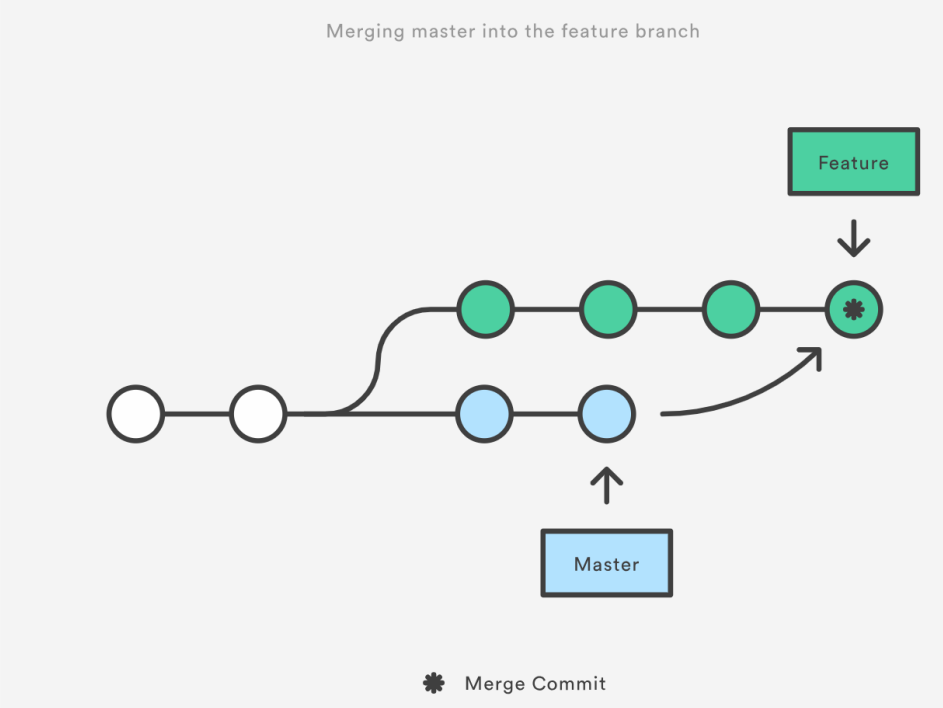
Merge & Rebase

A forked commit history

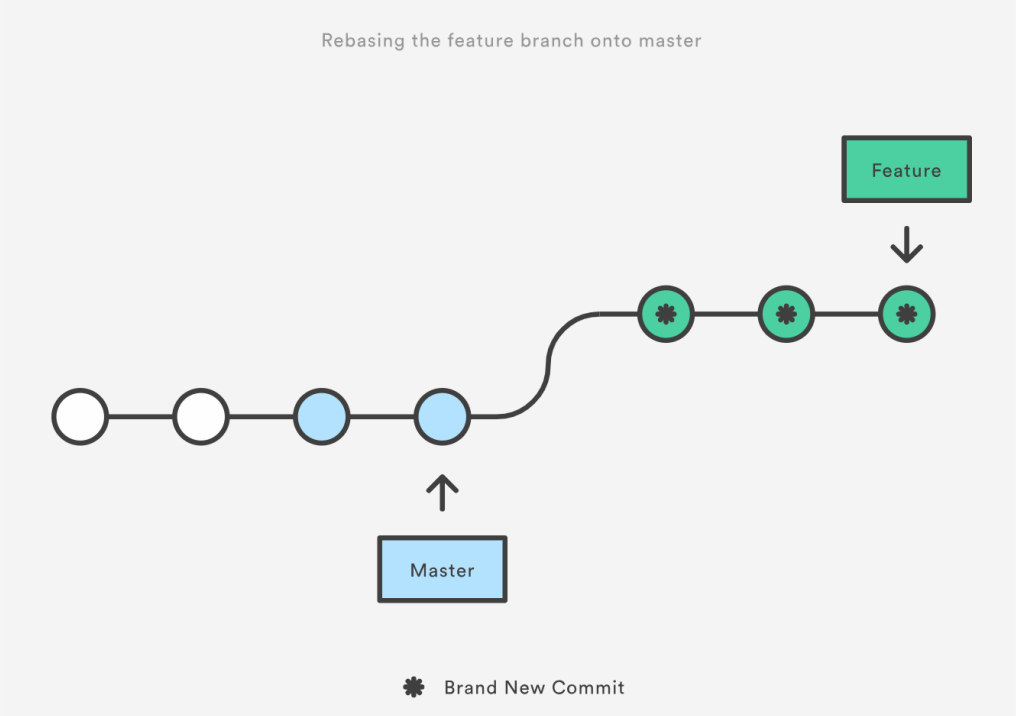


- Feature and Master need to be integrated now
- Two options – merge or rebase

GIT MERGE

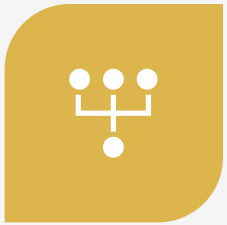

Merging master into the feature branch
Feature
Master
Merge Commit

GIT REBASE


Rebasing the feature branch onto master
Feature
Master
Brand New Commit

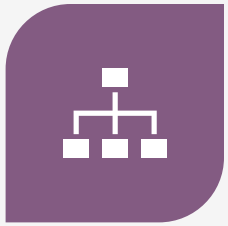
Merge Conflicts

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

Remote Repositories

Remote repositories are hosted on the network somewhere.

Collaborating with others involves managing these remote repositories and pushing and pulling data to and from them when you need to share work.

Managing remote repositories
`git remote show origin`

- add remote repositories*

  ***git clone** command implicitly adds the **origin** remote for you
- remove remotes that are no longer valid
- manage various remote branches

Remote branches

- References to the state of remote branches
- Git makes sure they accurately represent the state of the remote repository
- Take the form `<remote>/<branch>`
  - `origin/master branch`
  - `origin/iss53`
- When you clone a repository, master branch that tracks origin/master automatically created
- Can set up other branches to track
  - `git checkout -b <branch> <remote>/<branch>`
    OR
    `git checkout --track <remote>/<branch>`

Fetching and pulling from remotes

- git fetch only downloads the data to your local repository — it doesn't automatically merge it with any of your work
  - `git fetch <remote>`
  - `<remote>` is `origin` by default
- git pull automatically fetches and then merges *tracked* remote branch into your *current* branch
  - `git pull`
  - Essentially `git fetch` followed by `git merge`

Git push

Sends commit to remote repository

Might be restricted

Syncs all changes with remote branch


In case of fire
1. git commit
2. git push
3. leave building

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`
- Tagging
  - Human readable pointers to specific commits
  - `$ git tag  -a v1.0 –m 'Version 1.0'`
    - This will name the HEAD commit "v1.0", with the description "Version 1.0"

Assignment 9

DEADLINE - MONDAY
JUNE 3RD 11:55 PM

Late submission until Friday
(but you have to study for
the exam!)

Lab 9

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/local/cs/bin:$PATH`

Make a directory 'gitroot' and get a copy of the Diffutils Git repository

- `$ mkdir gitroot`
- `$ cd gitroot`
- `$ git clone <URL>`
- Follow steps in lab and use man git to find commands

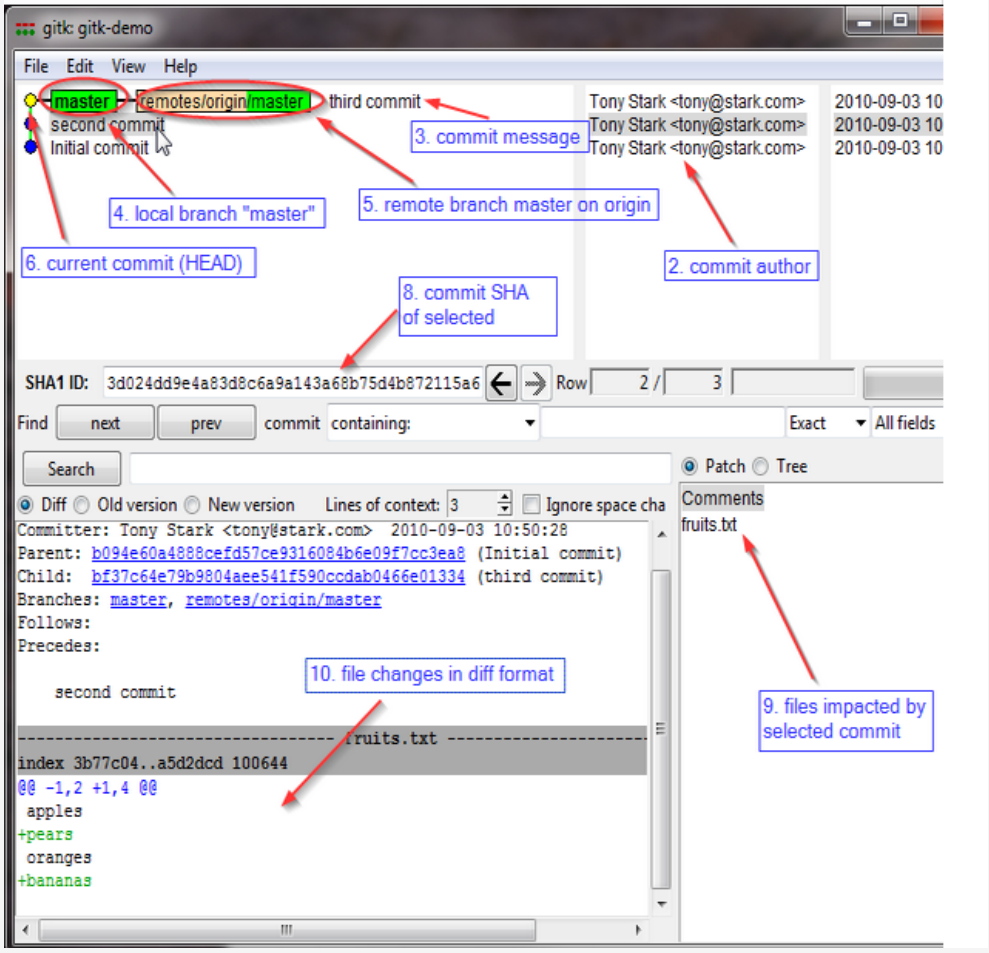
Publish patch you made in lab 9

- 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

Homework 9

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