Open Source Software

Git

Distributed Version Control System

http://git-scm.com/

http://git-scm.com/doc

https://help.github.com/articles/what-are-other-good-resources-for-learning-git-and-github

http://githowto.com/

Goal of Version Control

- Other options besides Git:
  - CVS, Subversion, Bazaar, BitKeeper, Team Foundation Server, ClearCase, Mercurial (hg)

History

- Allow multiple people to work on the same software easily
- Allow a single user to track all his/her changes
- Developed for use with the Linux Kernel
  - move away from proprietary BitKeeper
- Modeled after Linux Kernel work f bw
  - branches
  - distributed
  - data assurance
- Mix of local and remote repositories

Let's f rst look at using the command line then we'll look at GitHub.

Documentation

- http://git-scm.com/docs/
  - link to GitHub cheat sheet (PDF - 2 pages)
  - videos
  - free book (Pro Git)
  - http://git-scm.com/book/en/Git-Basics-Undoing-Things

Setup

- Open a terminal
  - terminator
- Go to Documents

```
script GitIntro.txt

git config --global core.editor "nano"
```

script will terminate when you type exit!

Typical Workflow - Single User

- **`mkdir MyCoolProject; cd MyCoolProject`**
- **`git init`**
  - builds the repository (.git directory) **`ls -al`**
  - the repository is in your local working directory
- create .gitignore
  - list types of files to not put into version control
    - any file that is generated: *.obj, *.o, *.class, *.pyc
- Create files! Do work!
- **`git add [filenames]`**
  - add the files you just created to the index for **staging**
- **`git commit -m "commit message"`**
  - actually commit changes to the repository
- **`git log`**

test.c

```
#include <stdio.h>

int main()
{
  printf(“HELLO”);
  return 0;
}
```

I need to revert!

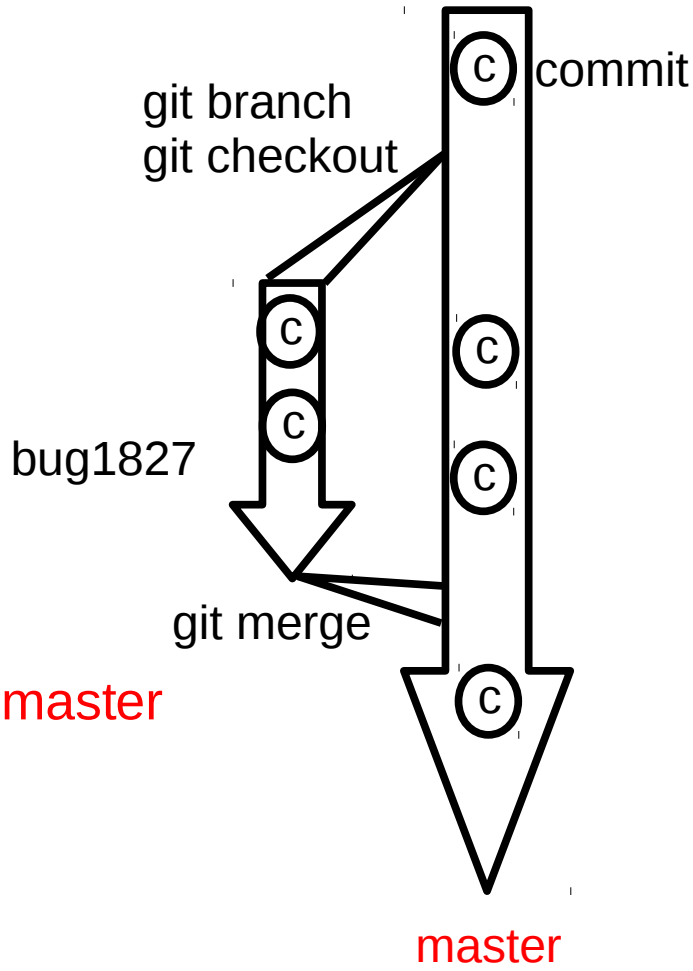
- http://git-scm.com/book/en/Git-Basics-Undoing-Things
- `git log --name-status`
- `git diff <commit hash> <filename>`
- `git checkout <commit hash> <filename>`
- `edit file`
- `git add`
- `git commit`

http://stackoverflow.com/questions/215718/reset-or-revert-a-specific-file-to-a-specific-revision-using-git/373848#373848

* two dashes precede a command line option of more than on character

Typical Workf bw

- Single user – bug f ix! (or maybe feature add)
  - **`git branch bug1827`**
    - create a branch to contain all the work for the bug f ix
  - **`git checkout bug1827`**
    - start using that branch
  - Do work (add/commit)
  - **`git checkout master`**
    to work on master again.
  - **`git merge --no-ff bug1827`**
    - replay the commits on bug1827 into master
  - **`git log`**
  - **`git branch -d bug1827`**



More commands

- http://git-scm.com/docs
  - **`git status`**
    - what files have uncommitted changes?
  - **`git log`**
    - show the commits and log messages
  - **`git diff`**
    - show the differences between local and committed files
    - build a patch you can email to someone else
  - **`git apply`**
    - apply a patch to your working directory
  - **`git blame`**
    - who last changed each line of a file?
  - **`git bisect`**

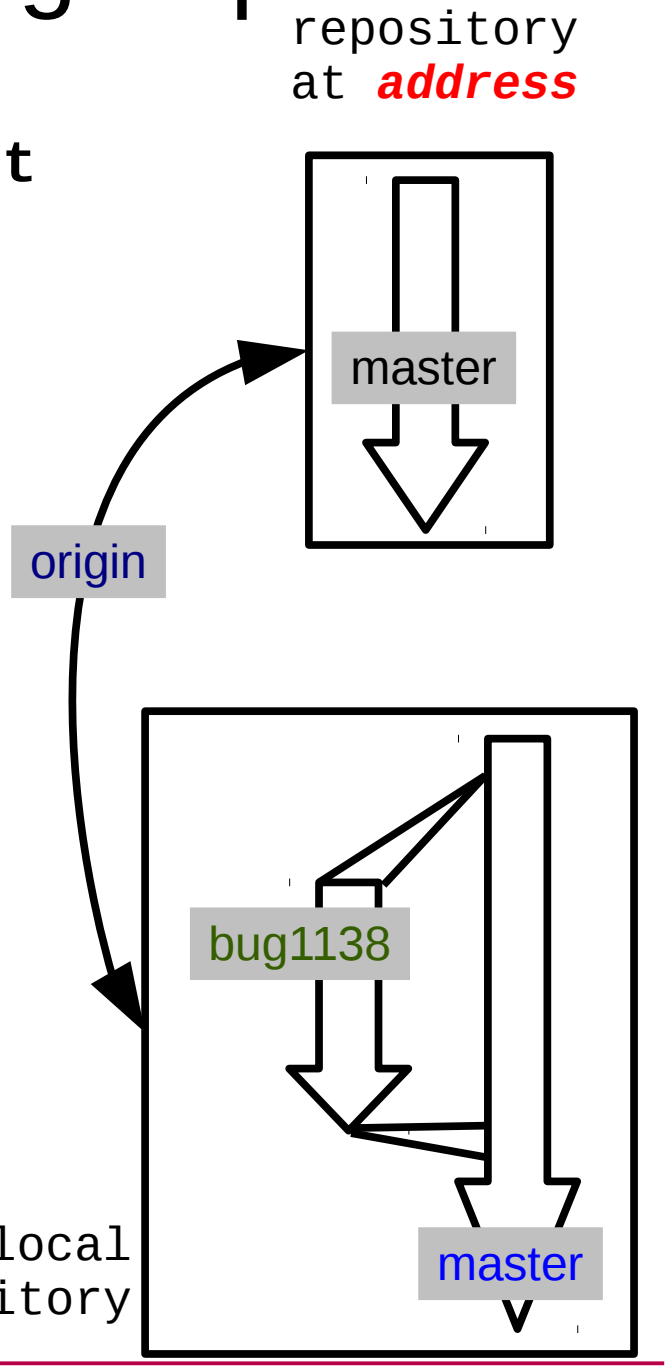
Workf bw - Group

- http://nvie.com/posts/a-successful-git-branching-model/

- http://scottchacon.com/2011/08/31/github-f bw.html

Typical Workf bw - group

- Group of developers
  - someone else: **`git --bare --shared init`**
  - **`git clone address`**
    - pull down code and setup **`origin`**
    - **`git remote -v`**
  - **`git branch bug1138`**
    - only a local branch is created
  - **`git checkout bug1138`**
  - do work
  - **`git add files / git commit`**
  - **`git checkout master`**
  - **`git merge --no-ff bug1138`**
  - **`git push origin master`**
  - **`git branch -d bug1138`**



Typical Workf bw - group

```
git fetch
git log ..origin/master
git checkout origin/master
git checkout master
git merge origin/master
```

OR

```
git pull
```

- git pull performs lots of magic
- hard to fix things when magic fails.

Conf lct

- edit/add/commit/push
  - edit/add/commit/push (ERROR)
  - fetch/merge (ERROR)
  - edit file (resolve conflict)
  - add/commit/push

```
<<<<<<< HEAD
BUY
=======
BYE
>>>>>>> origin/master
```

- fetch/merge

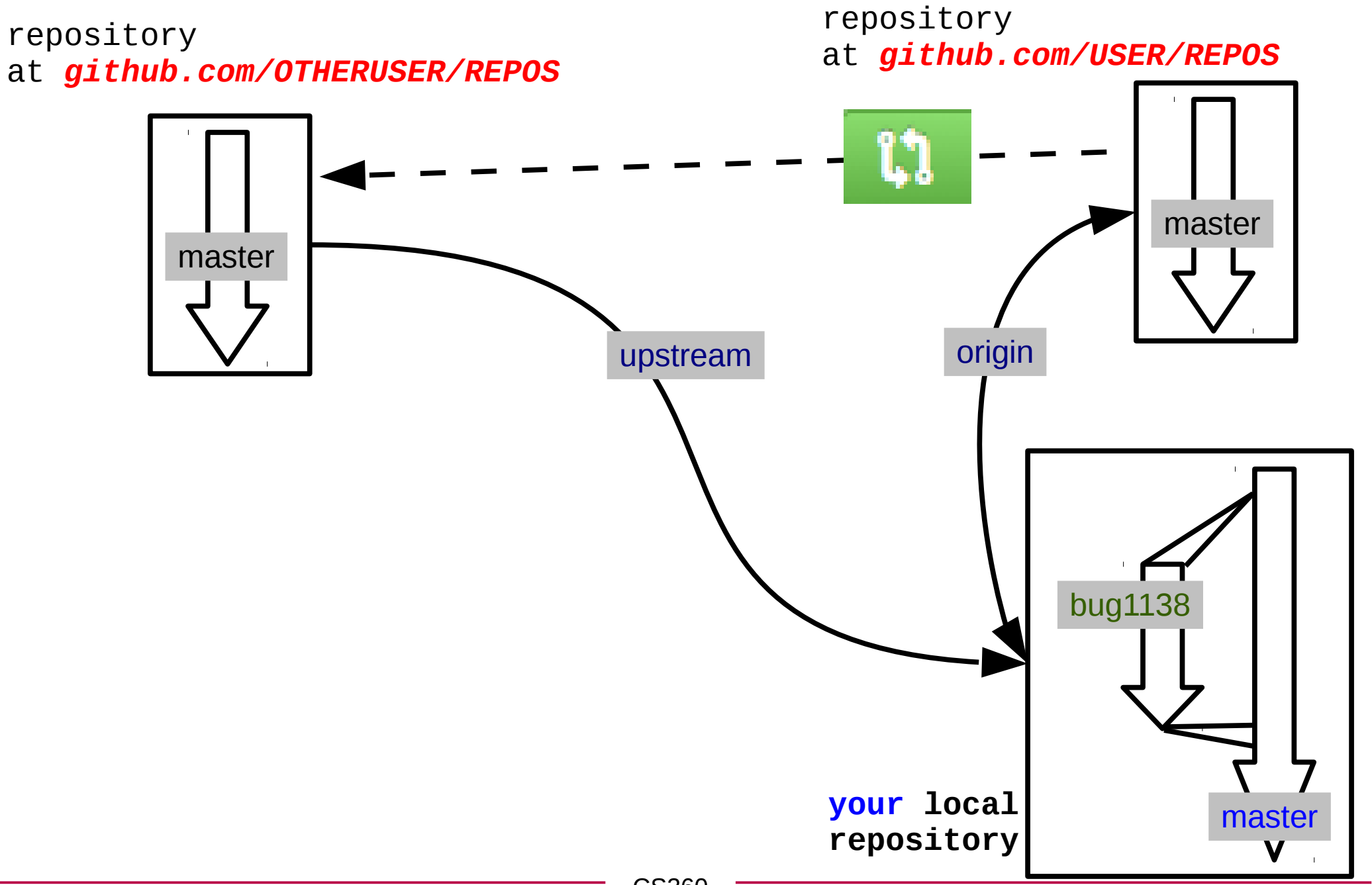
Typical Workf bw - GitHub

- Individual
  - Create repository at GitHub
    - setup .gitignore and license.
  - **`git clone git@github.com:USER/REPOS.git`**
    - pull down code and setup **`origin`**
  - **`git checkout -b bug1138`**
  - do work
  - **`git add files`**
  - **`git commit …`**
  - **`git checkout master`**
  - **`git merge --no-ff bug1138`**
  - **`git push origin master`**

Typical Workf bw - GitHub

- Group of developers
  - someone else creates a repository on GitHub (**OTHERUSER**)
  - on your GitHub account, fork the repository
  - **git clone *git@github.com:USER/repos.git***
  - **git remote add upstream git@github.com:OTHERUSER/FirstGitPractice.git**
  - **git checkout -b bug1138**
  - **do work/add/commit**
  - **git push origin bug1138**
    - **push to YOUR GitHub repository**
  - **On GitHub, issue a Pull request!**

  - **git fetch upstream**
  - **git merge upstream/master**


repository
at github.com/OTHERUSER/REPOS
repository
at github.com/USER/REPOS
master
master
upstream
origin
bug1138
master
your local
repository

Create newFileForPullRequest.txt #2

Edit

Open **chaddcw** wants to merge 1 commit into `cs360f14:master` from `chaddcw:master`

Conversation 0 | Commits 1 | Files changed 1 | +1 −0

chaddcw commented just now — Owner

This file was added to demonstrate what a pull request looks like.

Create newFileForPullRequest.txt ··· e3eae4e

Add more commits by pushing to the `master` branch on **chaddcw/FirstGitPractice**.

This pull request can be automatically merged.
You can also merge branches on the command line.

Merge pull request

Write | Preview — Parsed as Markdown — Edit in Fullscreen

Leave a comment

Attach images by dragging & dropping or selecting them.

ProTip Add comments to specific lines under Files changed.

Close | Comment

Milestone
No milestone

Assignee
No one assigned

Notifications
Unsubscribe
You're receiving notifications because you authored the thread.

1 participant

htt

Providers

- GitHub
  - free ($$$), proprietary (not open)
  - Go get a GitHub account and email me your username
- GitLab
  - open source
  - https://about.gitlab.com/
  - https://gitlab.com/gitlab-org/gitlab-ce/

GitLab

CI

- http://docs.travis-ci.com/user/getting-started/
- http://computer-vision-talks.com/articles/2014-02-23-using-travis-ci/