Source Code Management

Aroon Chande

Lecture 3

Tuesday, January 16, 2018

Outline

- Source code management and version control
- Introduction to `git`
- Class Github

Outline

- Source code management and version control
- Introduction to `git`
- Class Github

The problem…

Many people, working on many files in a codebase, at the same time, collaboratively

A codebase is not just code

- A “codebase” is all the files that go into a project

- In a normal project this includes:
  - Code (Perl, Python, and R scripts for example)
  - Documentation
  - Build configuration and tools
  - Examples for new users
  - Test cases to ensure your code does what it is supposed to do after each edit

- For you it will also include results

Does this sound familiar?

1. You are writing a script for a homework assignment and just need to add a few more things to finish

2. You make what seems like a small edit, but it breaks your script

3. You manually undo your changes…

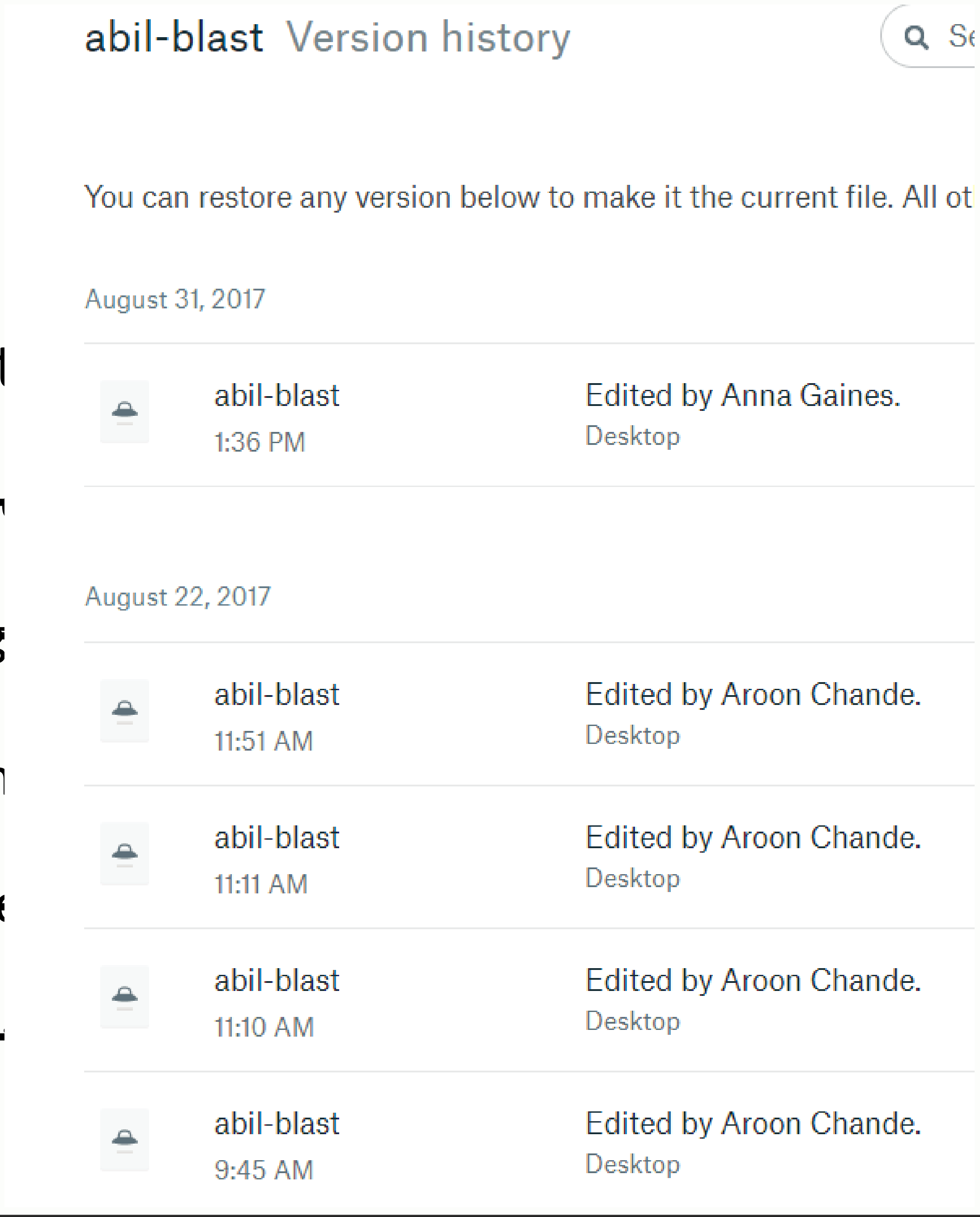
4. …But it is still broken!?

This semester’s problem

- Each group will generate a codebase but:
    - How do you keep track of it?
    - What is the latest and greatest version?
    - What if you accidentally delete all your work?
    - How do you share your codebase?

Potential solutions

- How do you keep track of it and what version?
  - ~~Just edit one big script that everyone w~~
  - ~~Copy and paste code snippets into a big~~
  - ~~Put it on Dropbox?~~ → You're on the righ
  - ~~The version that's been copy and paste~~
  - ~~The one your groupmate emailed you a~~



abil-blast Version history

Se

You can restore any version below to make it the current file. All ot

August 31, 2017

abil-blast 1:36 PM	Edited by Anna Gaines. Desktop

August 22, 2017

abil-blast 11:51 AM	Edited by Aroon Chande. Desktop
abil-blast 11:11 AM	Edited by Aroon Chande. Desktop
abil-blast 11:10 AM	Edited by Aroon Chande. Desktop
abil-blast 9:45 AM	Edited by Aroon Chande. Desktop

Potential solutions

- What if you accidentally delete all your work?
  - Go through the 5 stages of grief, eat a pizza all by yourself, and don’t tell anyone?
  - Frantically email everyone trying to find another copy?
- How do you share your work?
  - Email?
  - Dropbox?

Version control to the rescue

- Use source code management tools, like `git`, for version control and file management

Version control

- A source code management (SCM) tool:
  - Keep multiple version of everything in your codebase, all in one place
  - Requires a comment or description be given for every change made before you update the master copy
  - Can show you the differences between versions of a file
  - Allows everyone to edit anything, at the same time

History of version control systems

- 1972 – Source Code Control System (SCCS)
  - Human readable history files
  - File integrity checking
  - Delta files, or only the changed lines, for updates – saved lots of space!
  - Could only track text files, no binaries
  - Operations get slower for every new file or change being added
- 1982 – Revision Control System (RCS)
  - Based on the ideas from SCCS, scaled a little better

- Both systems could only edit one file at a time and were local only

History of version control systems

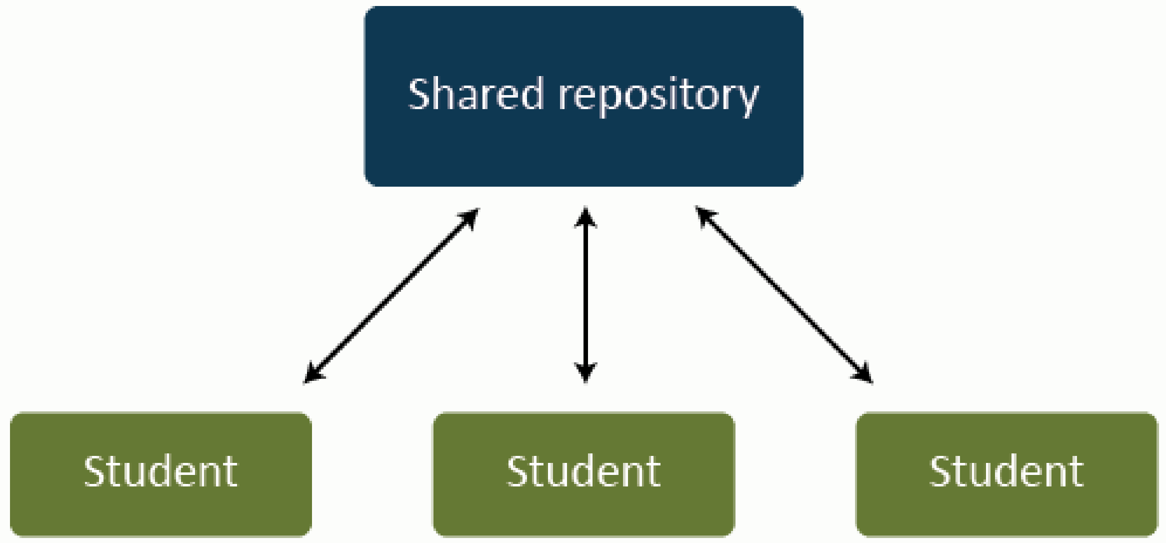
- 1986 – Concurrent Versions System (CVS)
  - Client-server model of data distribution, first popular (and F/OSS) SCM that supported networking
  - Multi-file editing per version with the concept of branching
  - Support for binary files
  - Multi-user, but not concurrently
- 2000 – Subversion (SVN)
  - Client-server model with diffs instead of files being transferred
  - Atomic operations(!)
  - Per file / folder versioning and binary support

History of version control systems

- 2005 – Mercurial (hg)
    - One of the first, widely used distributed version control systems
    - Designed to be highly scalable and performant
    - Multi-file, multi-user concurrent editing with branching
- 2005 – Git
    - Created for use by the Linux kernel developers
    - The other widely used distributed version control system
    - Built for flexibility and scalability

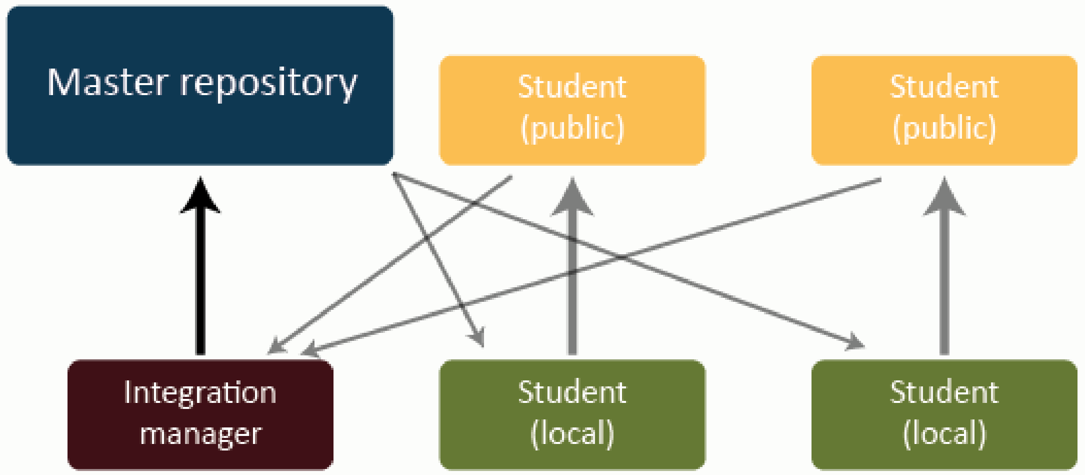
Management styles

Everybody edits, where each **collaborator** can make changes directly to the **shared master**



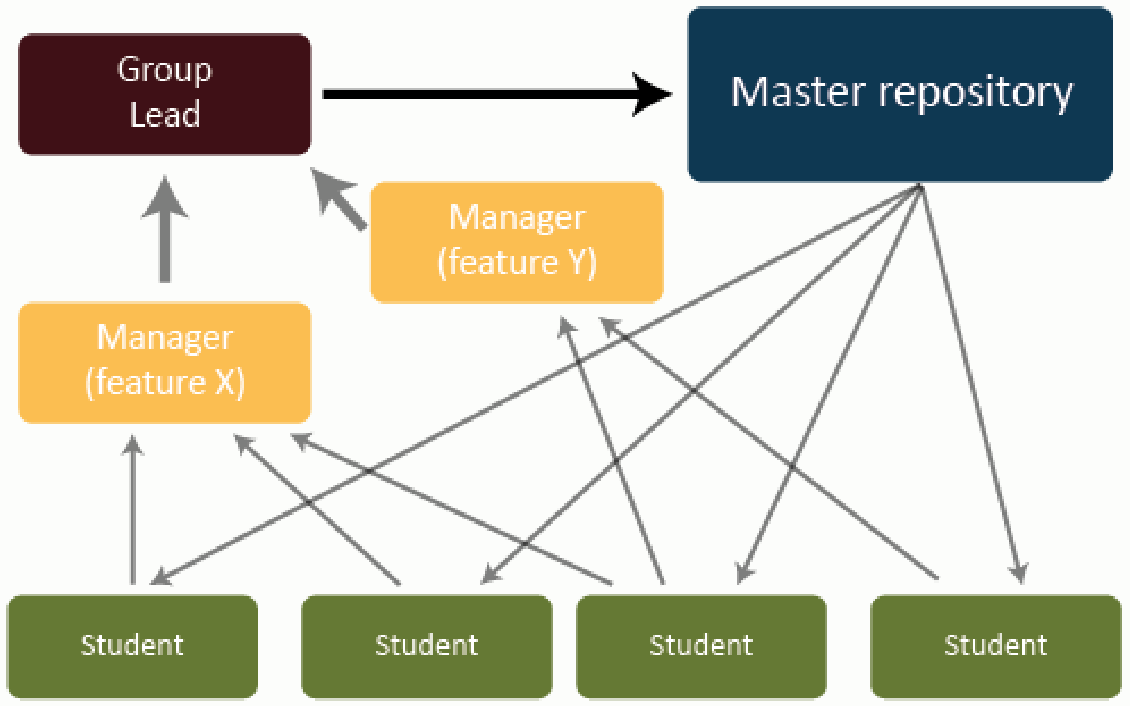
Management styles

Managed integration, where the **integration manager** reviews and merges changes into the **master** repository



Management styles

Corporate model, feature-specific groups with **managers**, and a **leader** in charge of final integration into the **master** repository



Outline

- Source code management and version control
- Introduction to `git`
- Class Github

Introduction to `git`

- Git is a free and open source SCM tool (https://git-scm.com/)

- It was originally developed by Linus Torvalds, the creator of the Linux kernel, to manage the kernel codebase
  - Git is *opinionated*

- The Linux kernel is a ***massive*** codebase

  >20 million lines of code

  ~37,000 files

  1000's of contributors

  The word "crap" appears surprisingly few times, only 191 times

https://www.linuxcounter.net/statistics/kernel

Introduction to `git`

- A project in git is called **repository**, or a “”repo”
- Files are added, or “checked in”, to a repo by **commit**ting your local changes to a shared **master** version
- A repo is copied by **cloning** or **forking**
- There can be multiple versions of a repo, all with different changes, called **branches**

Introduction to `git`

- The integrity of the files in your repo is guaranteed
  - You will get exactly the same file out of git as you put into git

- Git is a distributed version control system
  - You make changes to a local copy and share your changes
  - An internet connection is not required to work

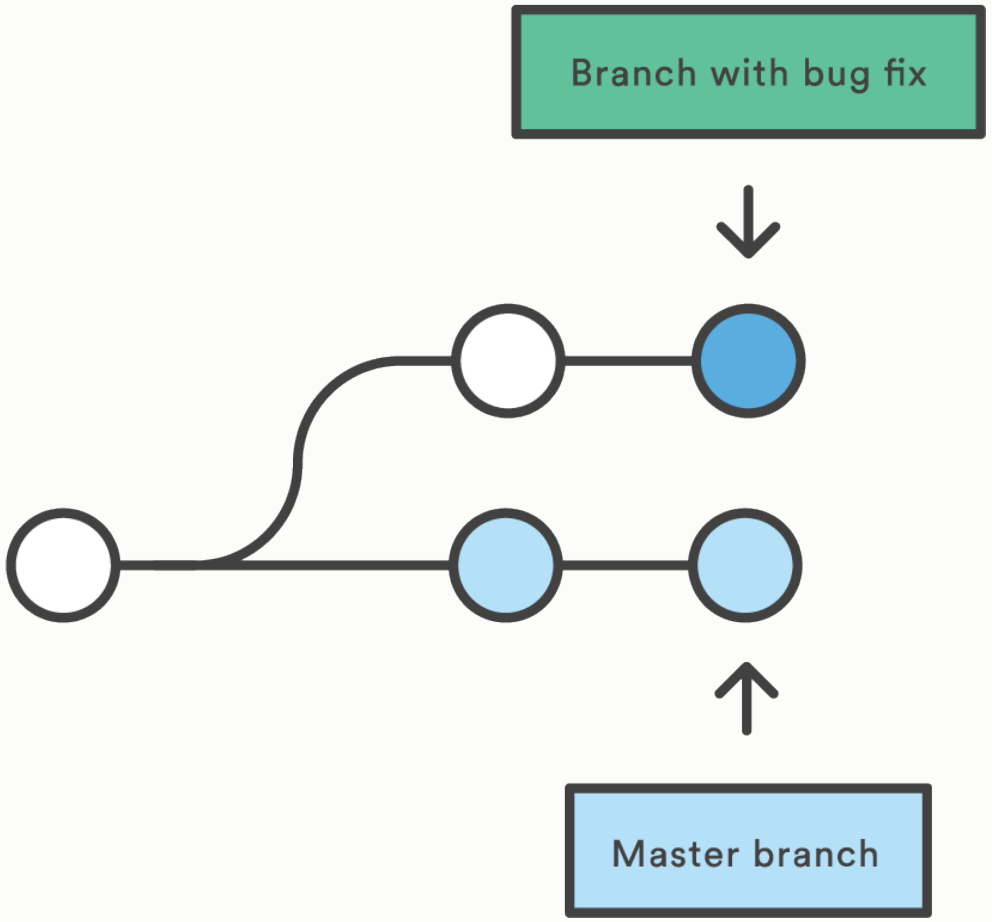
Introduction to `git`

- When you **clone** a repo, you download the current version and history of every file from the server
  - Changes you make get sent back to the same server/repo
- When you **fork** a repo, you download the current version and history of every file from the server **and** claim ownership of the repo
  - Changes you make get sent to a different master copy on the server
  - Your copy no longer has the same changes as the original repo you forked

Introduction to `git`

- Git repos are sometimes viewed as trees
- There’s a **master** branch that is like the trunk of the tree, the source of each subsequent branch
- Branches can be split from the master branch or other branches
- We use branches for working on and staging fixes and new features

Introduction to `git`

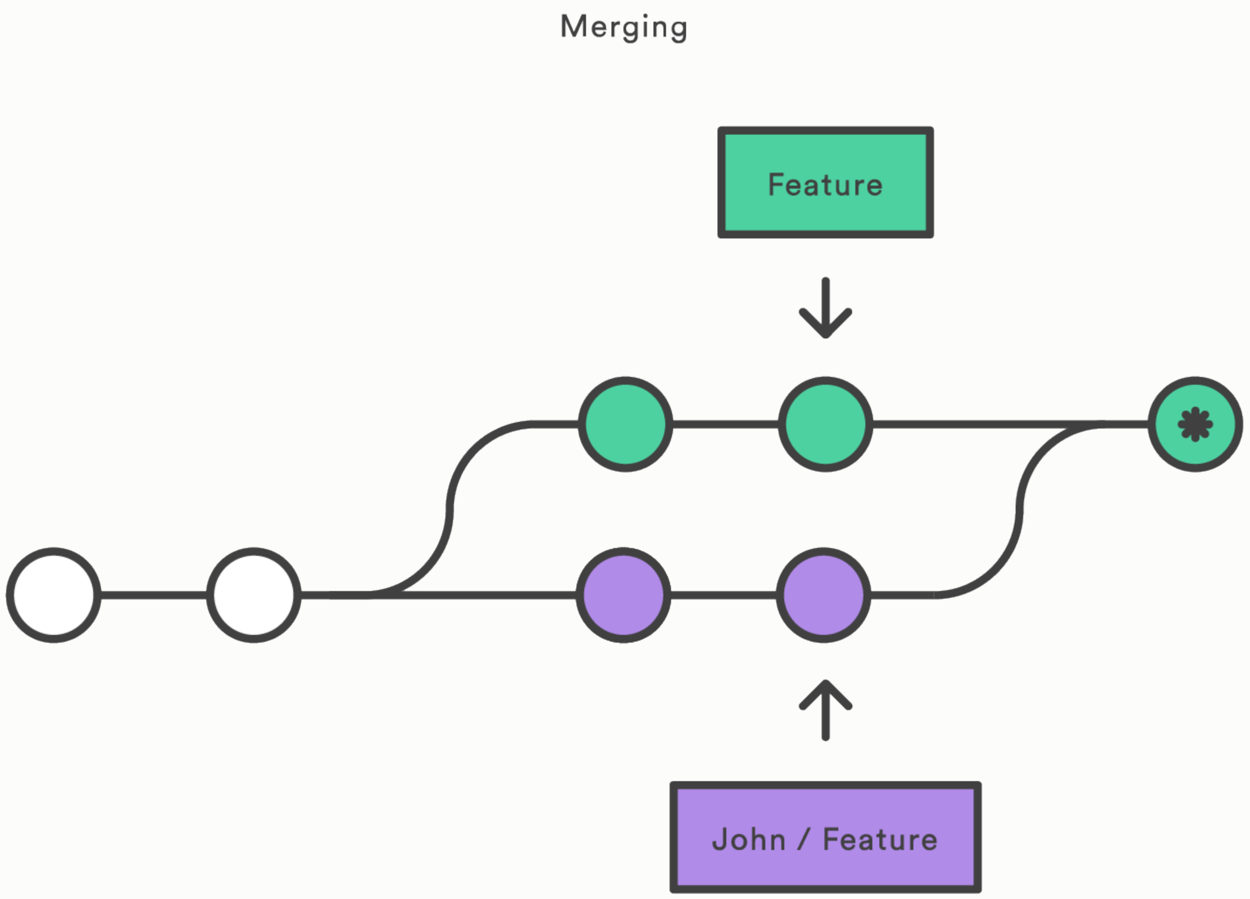


https://www.atlassian.com/git/tutorials

Introduction to `git`

- The work done on two branches can be **merged** together
- This process saves the history of each branch, then tries to combine changes
- Changes to the same file in both branches may result in conflicts

Introduction to `git`



https://www.atlassian.com/git/tutorials/merging-vs-rebasing

Introduction to `git`

- Git follows the Work-Stage-Commit lifestyle
  - *Work* on files in a repo you've cloned or forked
  - *Stage* your work and describe work to be added (`git add`)
  - *Commit* your changes to the repo, saving a snapshot in time (`git commit`)

Introduction to `git`

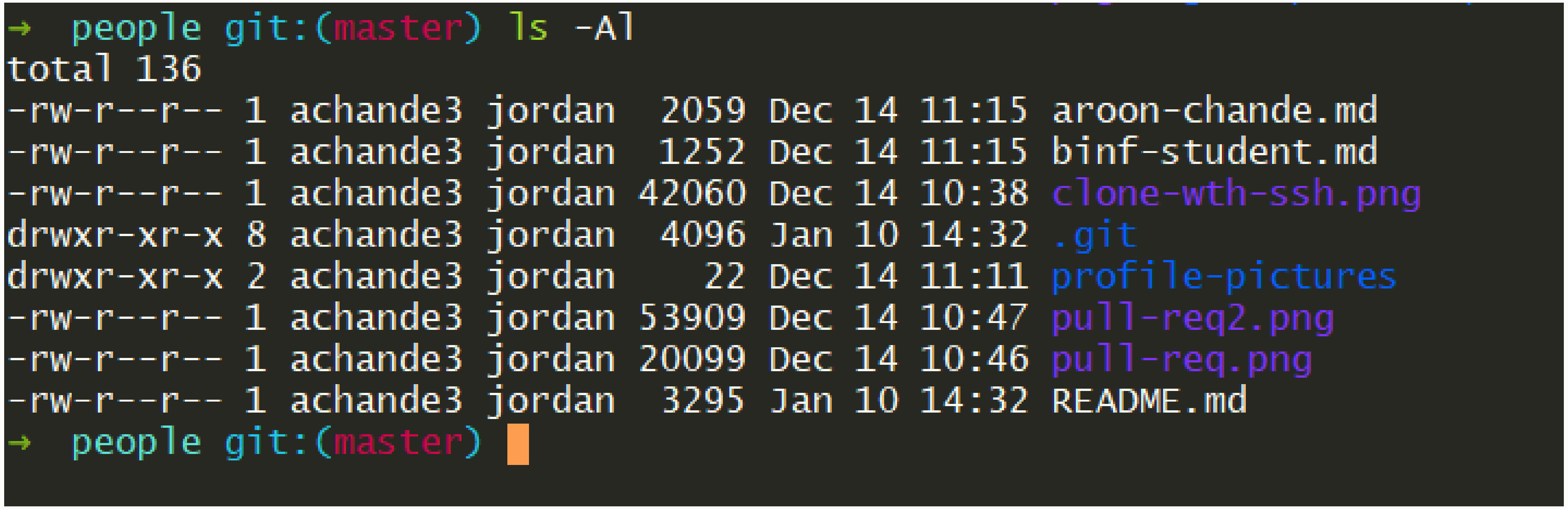
- You **push** your local changes (commits) to the server
  - This allows other contributors to view your work
- You can **pull** other users’ changes from the server copy into your local copy
  - This is like merging two branches but instead merges forward and backward in history

Introduction to `git`

- A special case of pulling, `git fetch`
- `git fetch` downloads and informs you of any changes but does not apply any changes, if present
- Fetch lets you answer the question, “Has anything changed on the server?”, without fear

Introduction to `git`

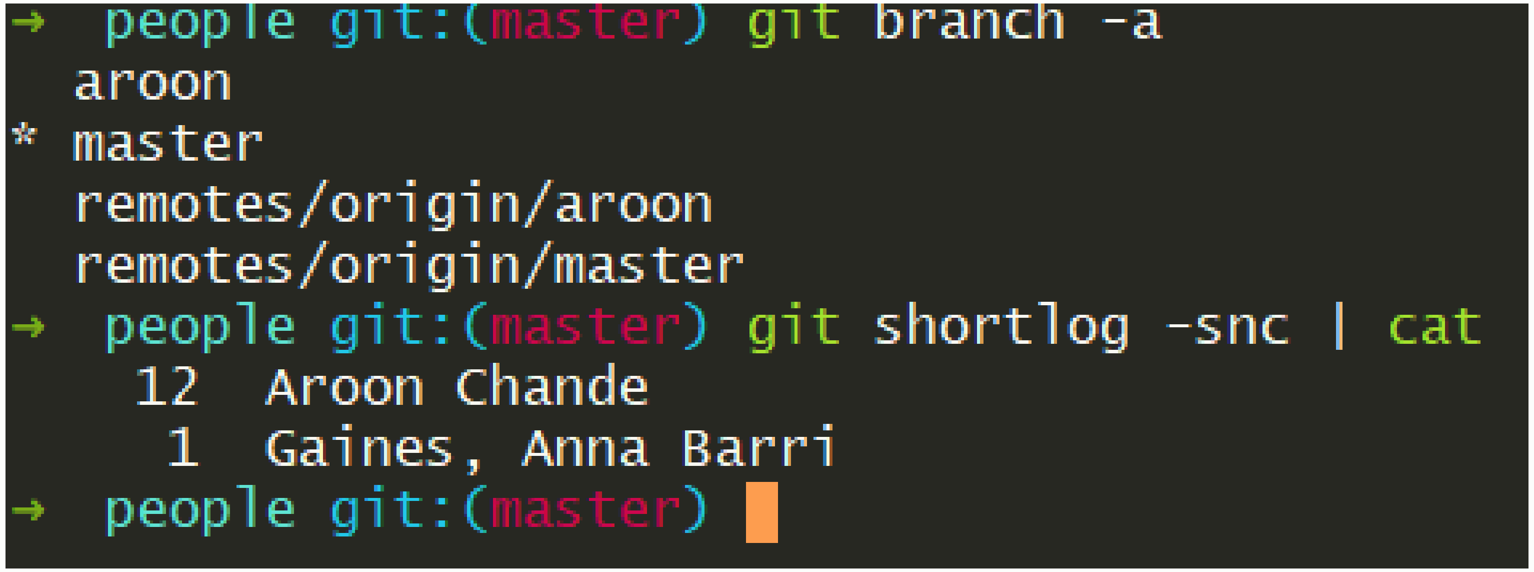
A simple git repository



```
➜  people git:(master) ls -Al
total 136
-rw-r--r-- 1 achande3 jordan  2059 Dec 14 11:15 aroon-chande.md
-rw-r--r-- 1 achande3 jordan  1252 Dec 14 11:15 binf-student.md
-rw-r--r-- 1 achande3 jordan 42060 Dec 14 10:38 clone-wth-ssh.png
drwxr-xr-x 8 achande3 jordan  4096 Jan 10 14:32 .git
drwxr-xr-x 2 achande3 jordan    22 Dec 14 11:11 profile-pictures
-rw-r--r-- 1 achande3 jordan 53909 Dec 14 10:47 pull-req2.png
-rw-r--r-- 1 achande3 jordan 20099 Dec 14 10:46 pull-req.png
-rw-r--r-- 1 achande3 jordan  3295 Jan 10 14:32 README.md
➜  people git:(master)
```

Introduction to `git`

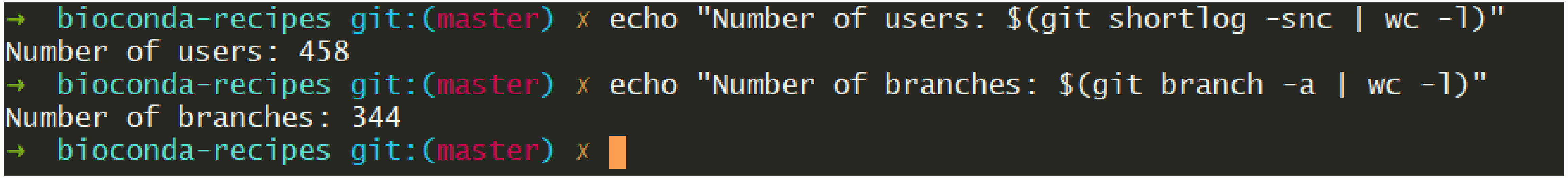
A simple git repository



```
→  people git:(master) git branch -a
  aroon
* master
  remotes/origin/aroon
  remotes/origin/master
→  people git:(master) git shortlog -snc | cat
    12  Aroon Chande
     1  Gaines, Anna Barri
→  people git:(master)
```

Introduction to `git`

- A >400 user science git repo



```
➜  bioconda-recipes git:(master) ✗ echo "Number of users: $(git shortlog -snc | wc -l)"
Number of users: 458
➜  bioconda-recipes git:(master) ✗ echo "Number of branches: $(git branch -a | wc -l)"
Number of branches: 344
➜  bioconda-recipes git:(master) ✗ 
```

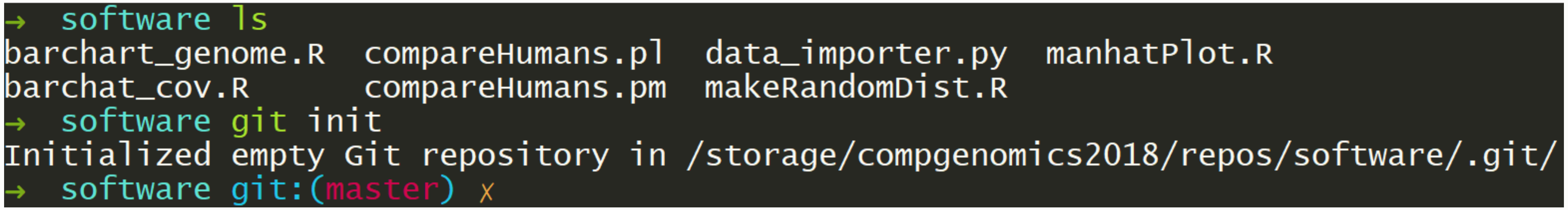
Introduction to `git`

- You will be using git from the command line and the Github enterprise web UI (github.gatech.edu) for this class

- You can edit files from the website
  - But save this mostly for text files (READMEs)
  - Editing scripts might break things

Introduction to `git`

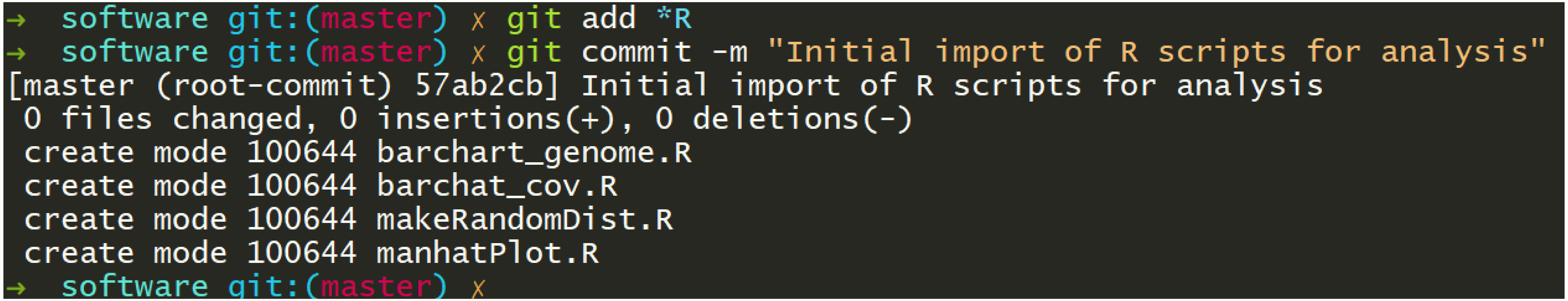
- Making a new git repo using `git init`



```
→  software ls
barchart_genome.R  compareHumans.pl  data_importer.py  manhatPlot.R
barchat_cov.R      compareHumans.pm  makeRandomDist.R
→  software git init
Initialized empty Git repository in /storage/compgenomics2018/repos/software/.git/
→  software git:(master) ✗
```

Introduction to `git`

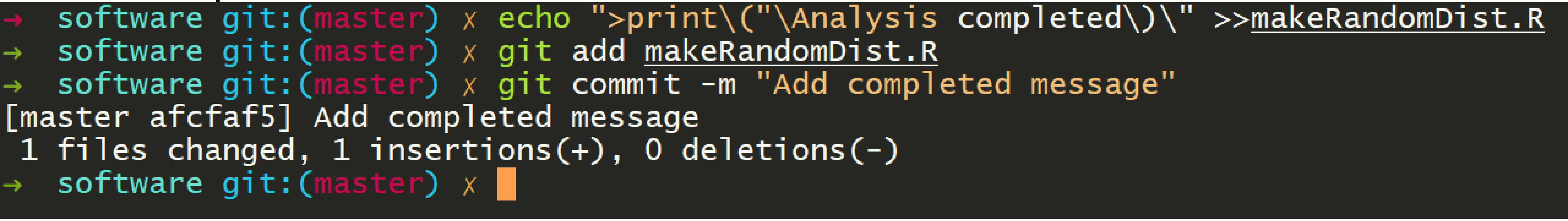
- Stage the R scripts and commit them



```
→  software git:(master) ✗ git add *R
→  software git:(master) ✗ git commit -m "Initial import of R scripts for analysis"
[master (root-commit) 57ab2cb] Initial import of R scripts for analysis
 0 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 barchart_genome.R
 create mode 100644 barchat_cov.R
 create mode 100644 makeRandomDist.R
 create mode 100644 manhatPlot.R
→  software git:(master) ✗
```

Introduction to `git`

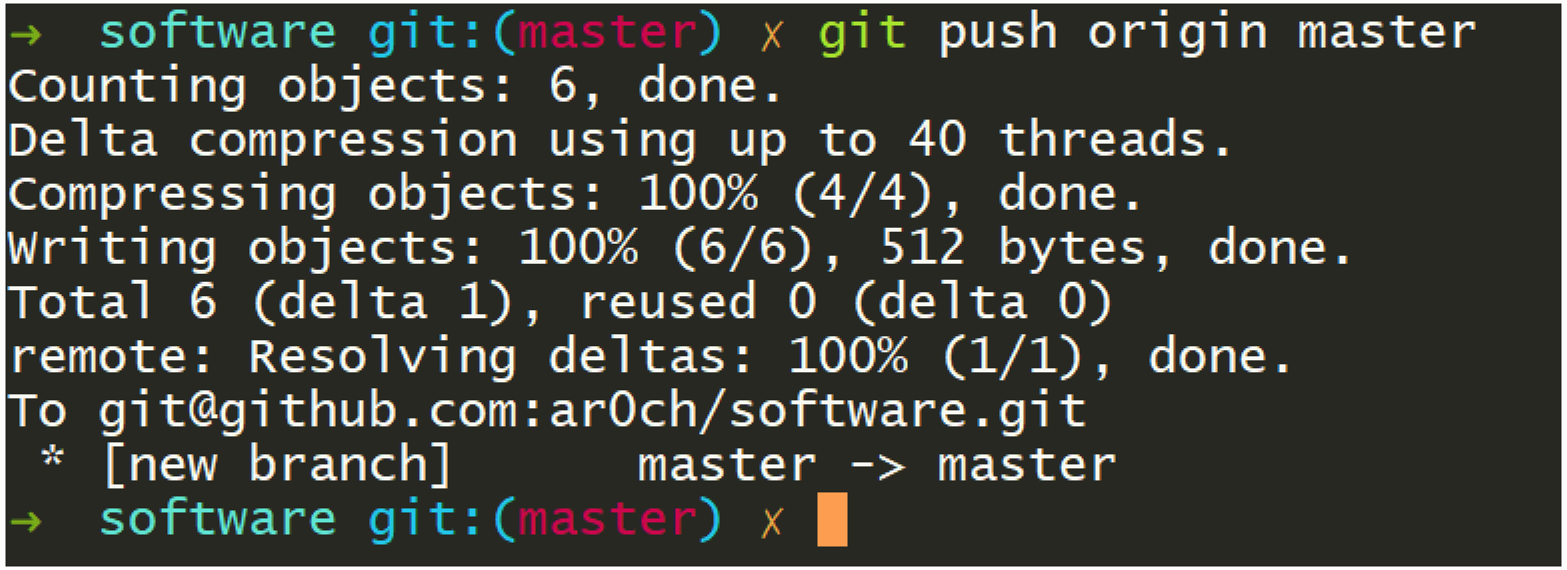
- Realize you forgot to add the completion message to one of the scripts



```
→  software git:(master) ✗ echo ">print\("\Analysis completed\)\" >>makeRandomDist.R
→  software git:(master) ✗ git add makeRandomDist.R
→  software git:(master) ✗ git commit -m "Add completed message"
[master afcfaf5] Add completed message
 1 files changed, 1 insertions(+), 0 deletions(-)
→  software git:(master) ✗ █
```

Introduction to `git`

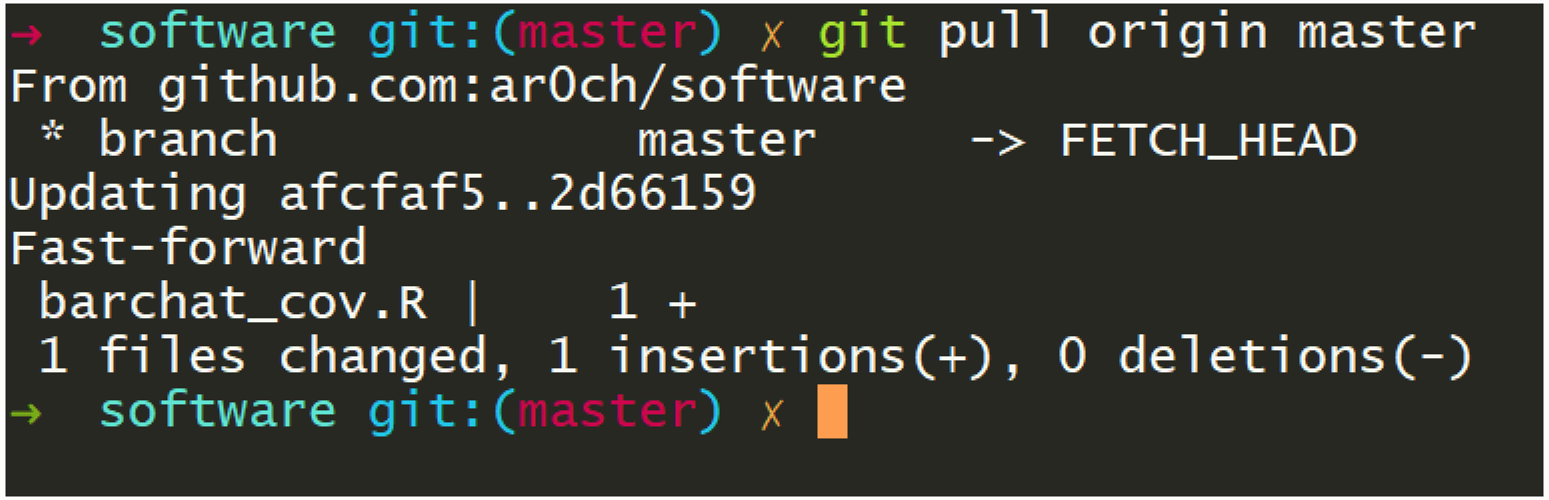
- Push the local changes to our master version on Github



```
→  software git:(master) ✗ git push origin master
Counting objects: 6, done.
Delta compression using up to 40 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 512 bytes, done.
Total 6 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To git@github.com:ar0ch/software.git
 * [new branch]      master -> master
→  software git:(master) ✗ 
```

Introduction to `git`

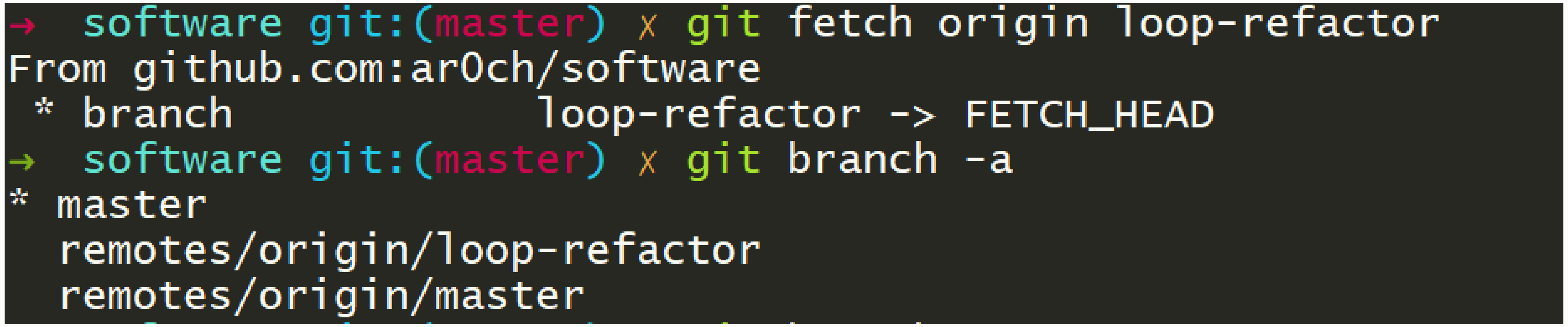
- Pull changes made by others into the master branch



```
➜  software git:(master) ✗ git pull origin master
From github.com:ar0ch/software
 * branch            master     -> FETCH_HEAD
Updating afcfaf5..2d66159
Fast-forward
 barchat_cov.R |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
➜  software git:(master) ✗ 
```

Introduction to `git`

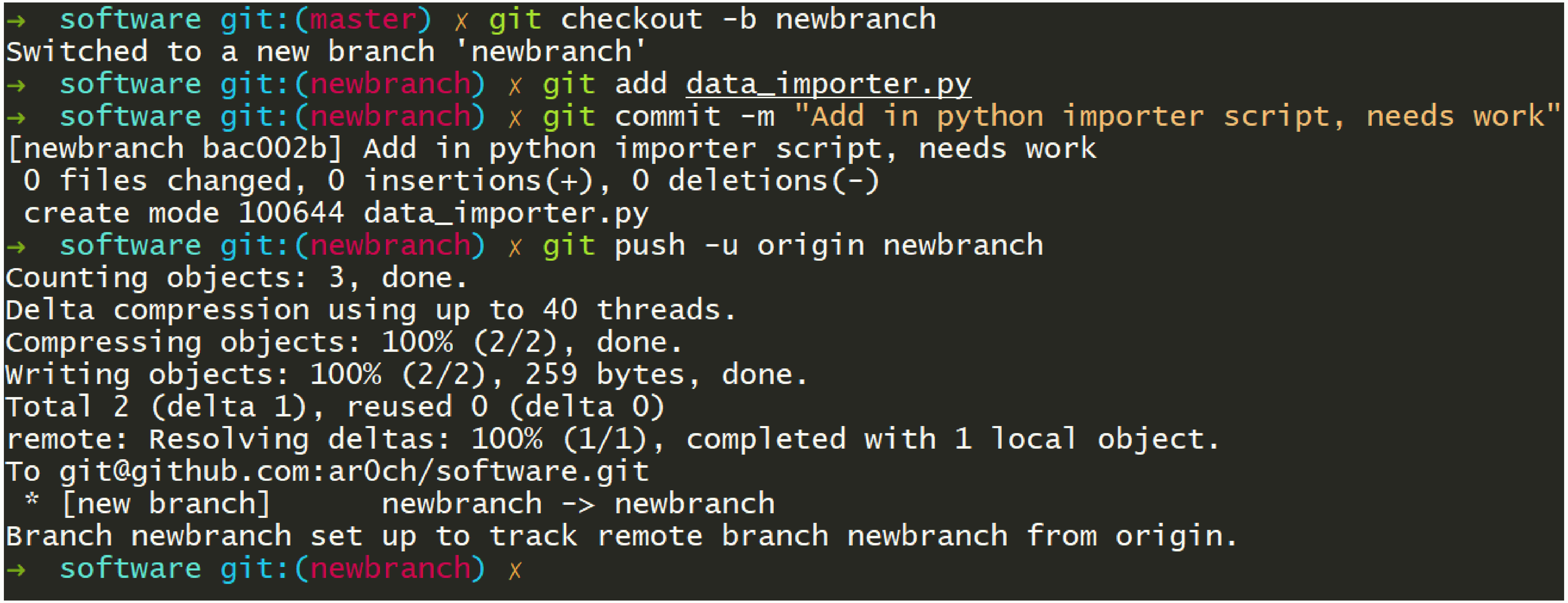
- Pull new braches off of master



```
→  software git:(master) ✗ git fetch origin loop-refactor
From github.com:ar0ch/software
 * branch            loop-refactor -> FETCH_HEAD
→  software git:(master) ✗ git branch -a
* master
  remotes/origin/loop-refactor
  remotes/origin/master
```

Introduction to `git`

- Create and push new branches



```
→  software git:(master) ✗ git checkout -b newbranch
Switched to a new branch 'newbranch'
→  software git:(newbranch) ✗ git add data_importer.py
→  software git:(newbranch) ✗ git commit -m "Add in python importer script, needs work"
[newbranch bac002b] Add in python importer script, needs work
 0 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 data_importer.py
→  software git:(newbranch) ✗ git push -u origin newbranch
Counting objects: 3, done.
Delta compression using up to 40 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 259 bytes, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To git@github.com:ar0ch/software.git
 * [new branch]      newbranch -> newbranch
Branch newbranch set up to track remote branch newbranch from origin.
→  software git:(newbranch) ✗
```

Introduction to `git`

A few tips

1. Commit often, push infrequently
    1. For each feature / fix / change / file save your edit history, push at end of session
2. Write good commit messages
    1. “Fix”, “Update”, “edits” are all bad commit messages
    2. Be short but descriptive. “Fixes issue #35; >1 period in file name break script”
3. Test before you push
    1. Try not to push broken code, especially to master
4. Learn how to reset your branches and don’t be afraid to restart a feature

Outline

- Source code management and version control
- Introduction to `git`
- Class Github

Class Github

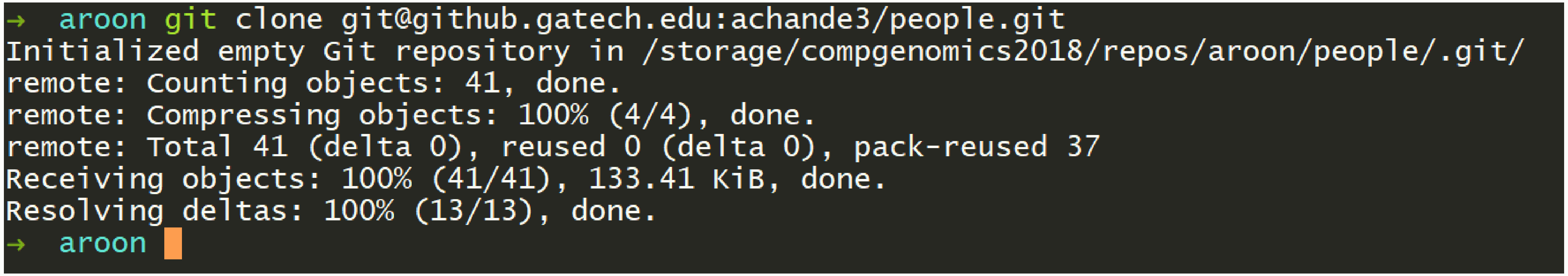
https://github.gatech.edu/compgenomics2018/

- This site will contain ***all*** your code, raw analysis and other work product
- During the semester, these data will be private, afterwards they will be open-sourced (on Github.com)

Github assignment

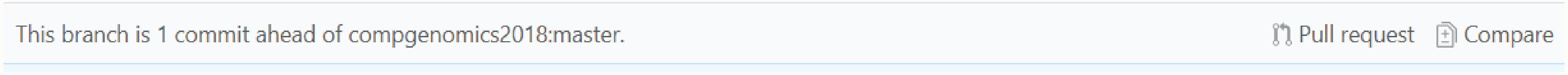
- Visit the People repo: https://github.gatech.edu/compgenomics2018/people
- Create a new fork
- Add a profile about yourself (you can copy the text from your wiki page). Be sure to include the Team and groups you're on
- Stage, commit and push your work. Then create a pull request

Cloning the repository

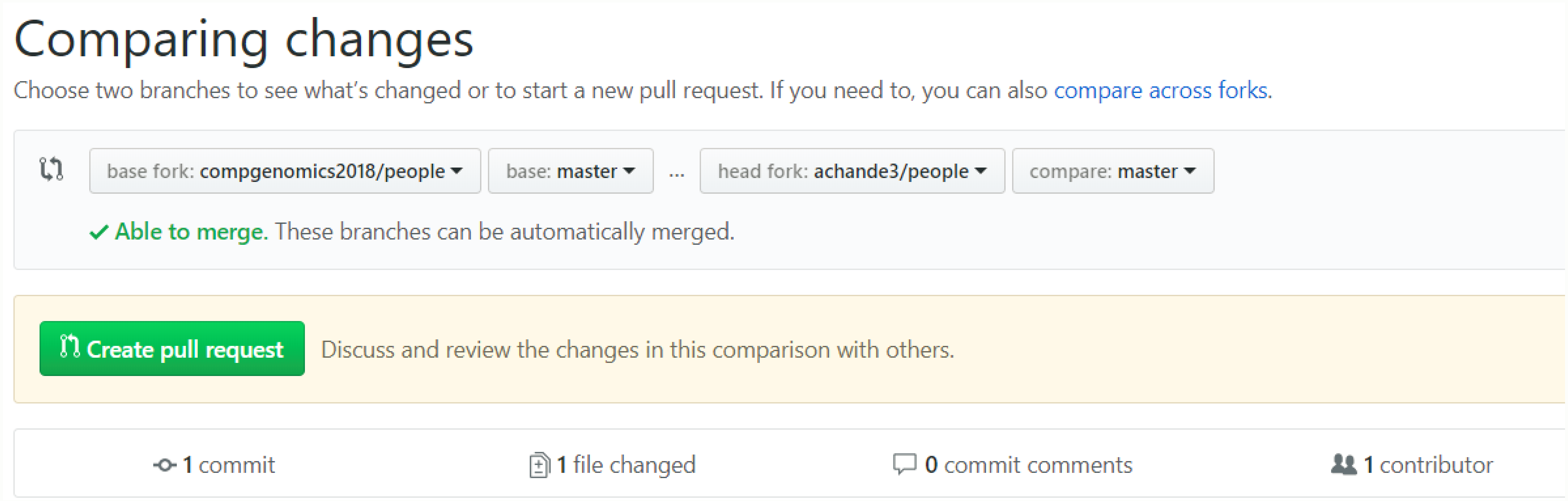


```
→  aroon git clone git@github.gatech.edu:achande3/people.git
Initialized empty Git repository in /storage/compgenomics2018/repos/aroon/people/.git/
remote: Counting objects: 41, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 41 (delta 0), reused 0 (delta 0), pack-reused 37
Receiving objects: 100% (41/41), 133.41 KiB, done.
Resolving deltas: 100% (13/13), done.
→  aroon
```

Creating a pull request



This branch is 1 commit ahead of compgenomics2018:master.



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base fork: compgenomics2018/people · base: master ... head fork: achande3/people · compare: master

✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others.

1 commit · 1 file changed · 0 commit comments · 1 contributor

Finishing up

- Once I’ve reviewed your pull request, I’ll either:
  - Accept it, merge your changes and add you to our organization
  - Or ask you to make some changes
- Feel free to explore the features available to you