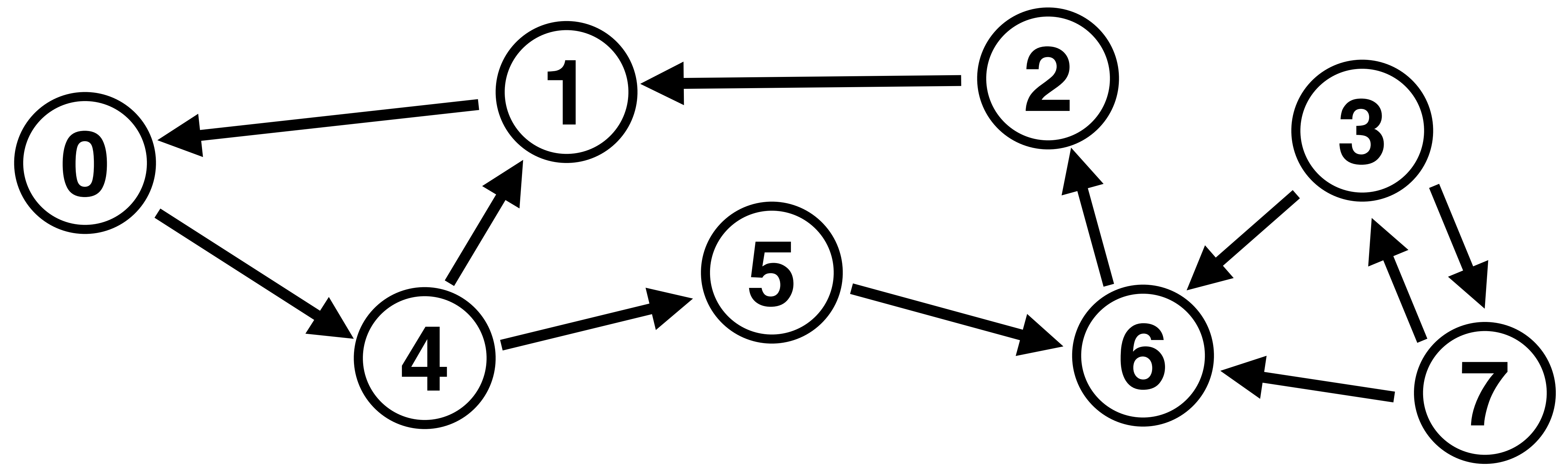
Version Control with Git

CS 121: Data Structures

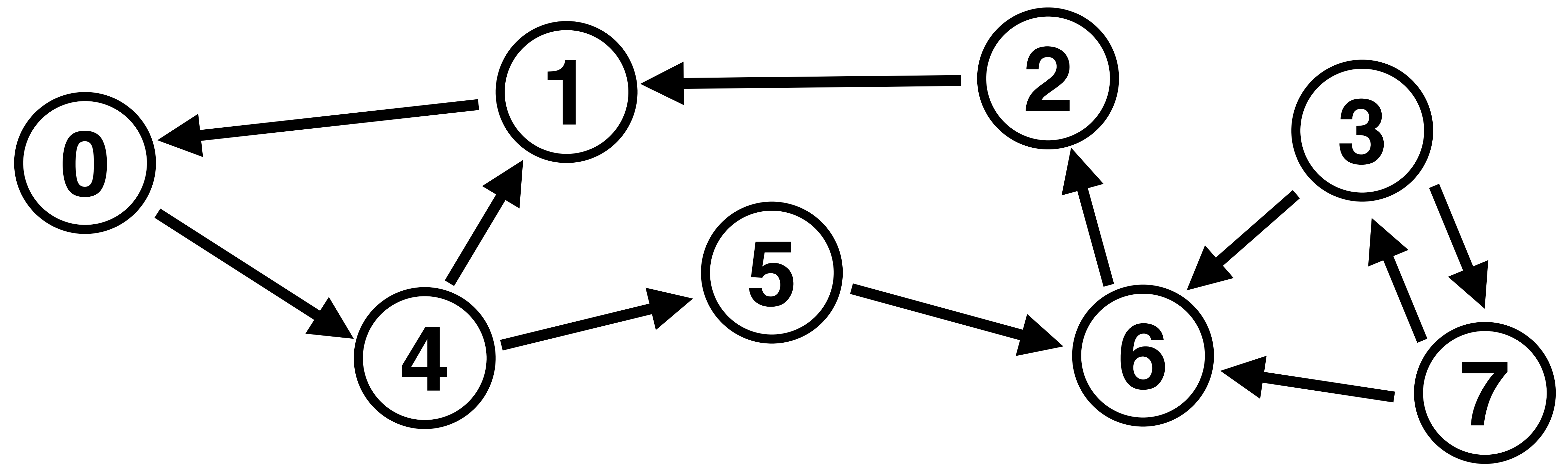
Attendance Quiz: Directed Graphs

- Scan the QR code, or find today's attendance quiz under the "Quizzes" tab on Canvas
- Password: to be announced
- Give the in-degree and out-degree of each vertex
- Which vertices can't be reached from a search starting at vertex 2?



Attendance Quiz: Directed Graphs

- Give the in-degree and out-degree of each vertex
- Which vertices can't be reached from a search starting at vertex 2?



Homework

Outline

- Motivation for version control systems
- Git:
  - Concepts
  - Data structures
  - Challenges
  - Advice

Version Control Systems

Problems: Code Versioning and Integrity

- Have you ever:
  - Accidentally deleted a file?
  - Made changes to a file that you wanted to undo?
    - Possible – **if** you haven't closed the editor
- When working on homework, these are (usually) minor inconveniences
- Working on real-world code, these become **major problems**!
  - What if you messed up a file people had been working on for years?!

Problem: Coordination

- Imagine 100 people are working on a software project together
- How can you coordinate this? All sharing the same computer? Emailing changes back and forth? Who has the latest version of the project?
  - This sounds like a nightmare!

Solution: "Version Control Systems" (VCS)

- Keep track of multiple versions of each file
  - A file may be edited over time, possibly by different people
- For any VCS to be scalable, appropriate data structures must be used
  - A good reason to cover them in this class!

VCS Landscape

Industry Standard	👑 git	👑 GitHub
Less Common	SUBVERSION	mercurial

Git Concepts

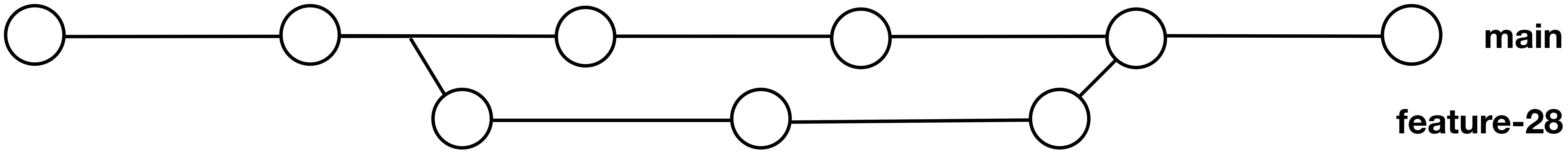
Commits

- It is helpful to record regular snapshots/checkpoints of your codebase
  - Git refers to these as "commits"
- Rule of thumb: commit code after a conceptually atomic change has been completed
  - Too granular: declaring a couple variables
  - Too coarse: adding 20 Java classes
  - Good: adding a single class, a method, or related methods, fixing a bug, etc.
- Git makes it easy to compare code to earlier commits, restore earlier commits, etc.

Tuesday @ 1:00pm
Add skeleton for Tour class
Tuesday @ 4:00pm
Implemented constructors and toString()
Thursday @ 5:00pm
Completed implementation of Tour class
Friday @ 5:00pm
Fixed bug in addNearest()

Branches

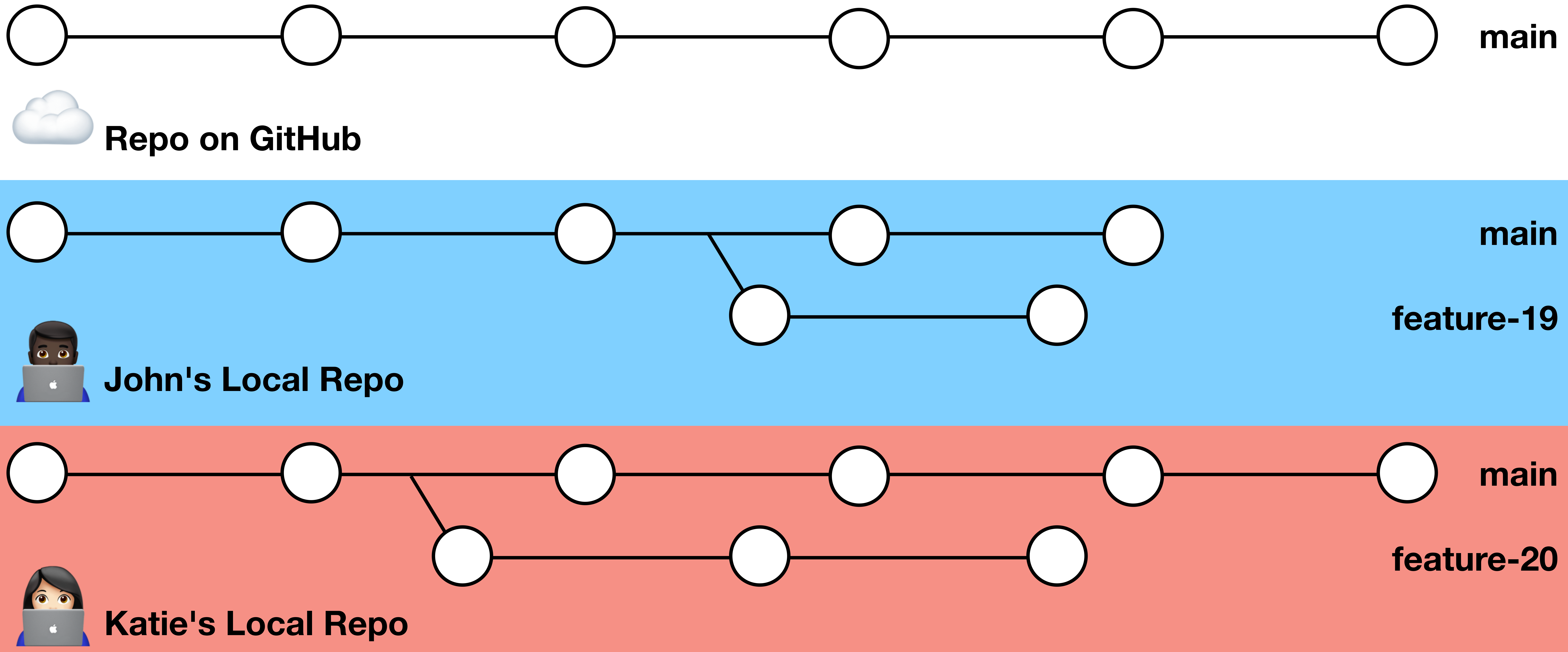
- What if you want to work on multiple changes at the same time?
    - For example, developing a new feature, while also fixing bugs?
- Git uses "branches" to allow code to develop along different paths
    - When changes are completed, they should be "merged"



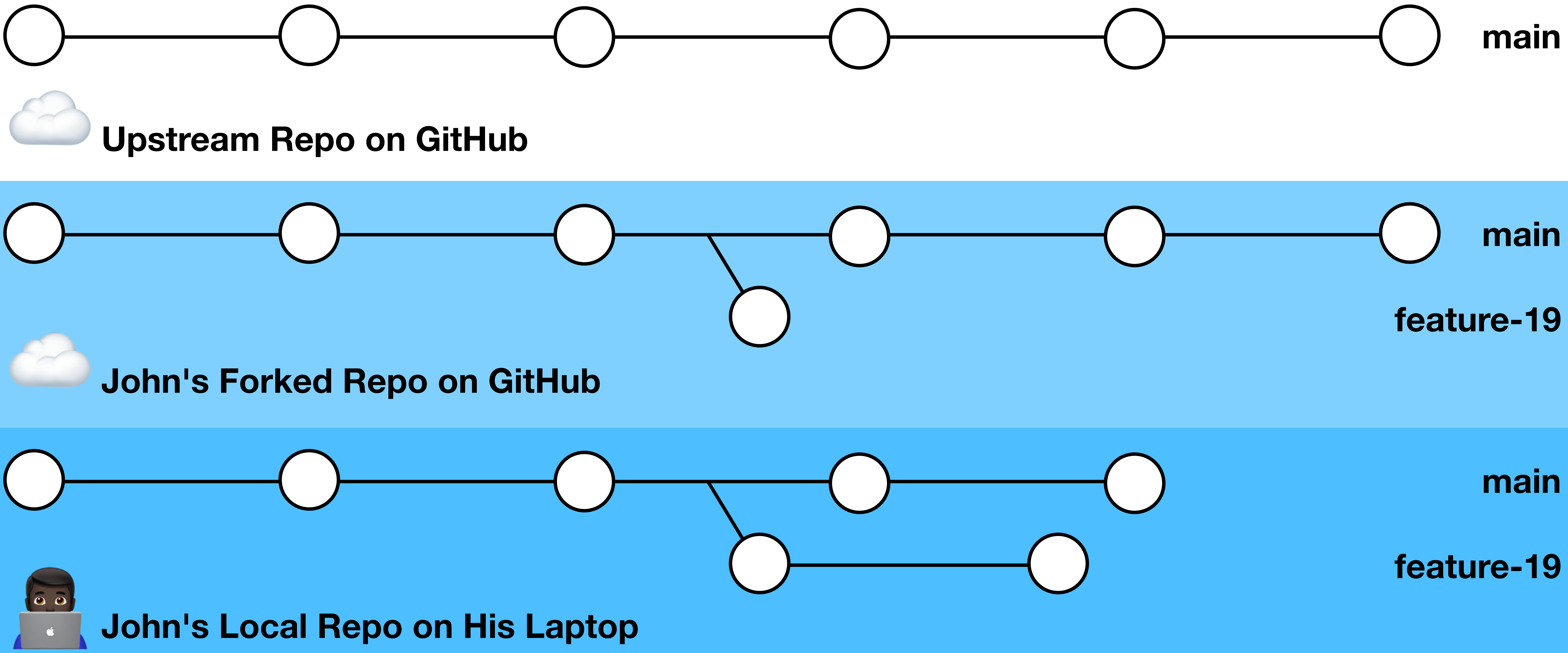
Multiple Copies of a Repository

- How can multiple people collaborate on a project, without getting in each other's way?
- Multiple versions of the git repository: one or more on servers (e.g., hosted by GitHub), other versions on developer's own computers
- Developers should periodically push their changes to the remote server, and eventually merge their changes into the main remote branch
  - They should also update their local version of the main branch regularly, so they can build on others' contributions

Distributed Version Control



Distributed Version Control



Demos

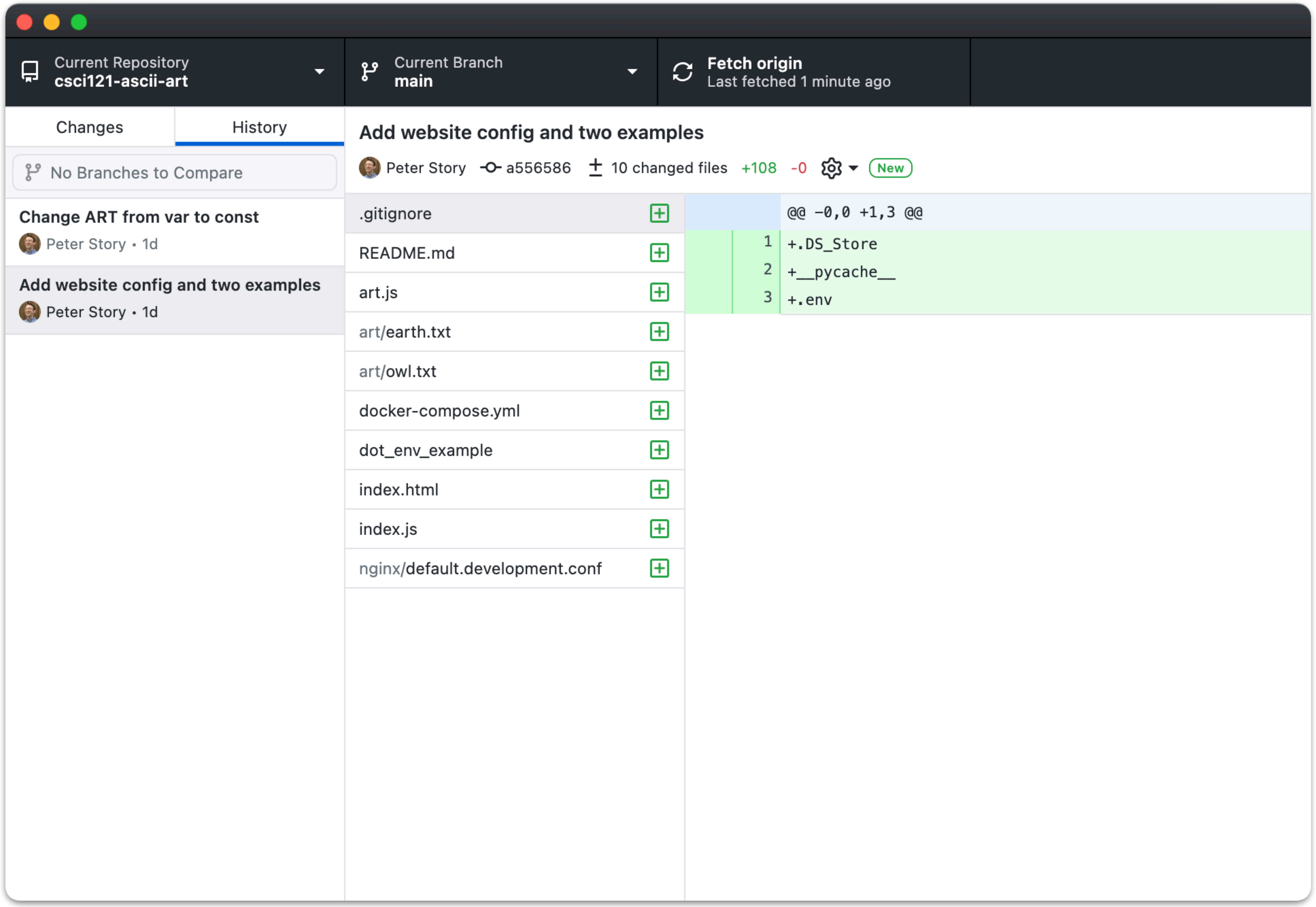
Demo: Using Git on GitHub

- Browse projects
- Make your own projects
- Contribute changes to others' projects
  - Fork
  - Create a branch
  - Make changes
  - Open a pull request

GitHub

Demo: Using Git with GitHub Desktop

- Update my fork's main branch
- Clone my fork
- Create a branch
- Make changes
- Commit my changes
- Publish (push) my changes
- Open a pull request



Change ART from var to const
Peter Story • 1d

Add website config and two examples
Peter Story • 1d

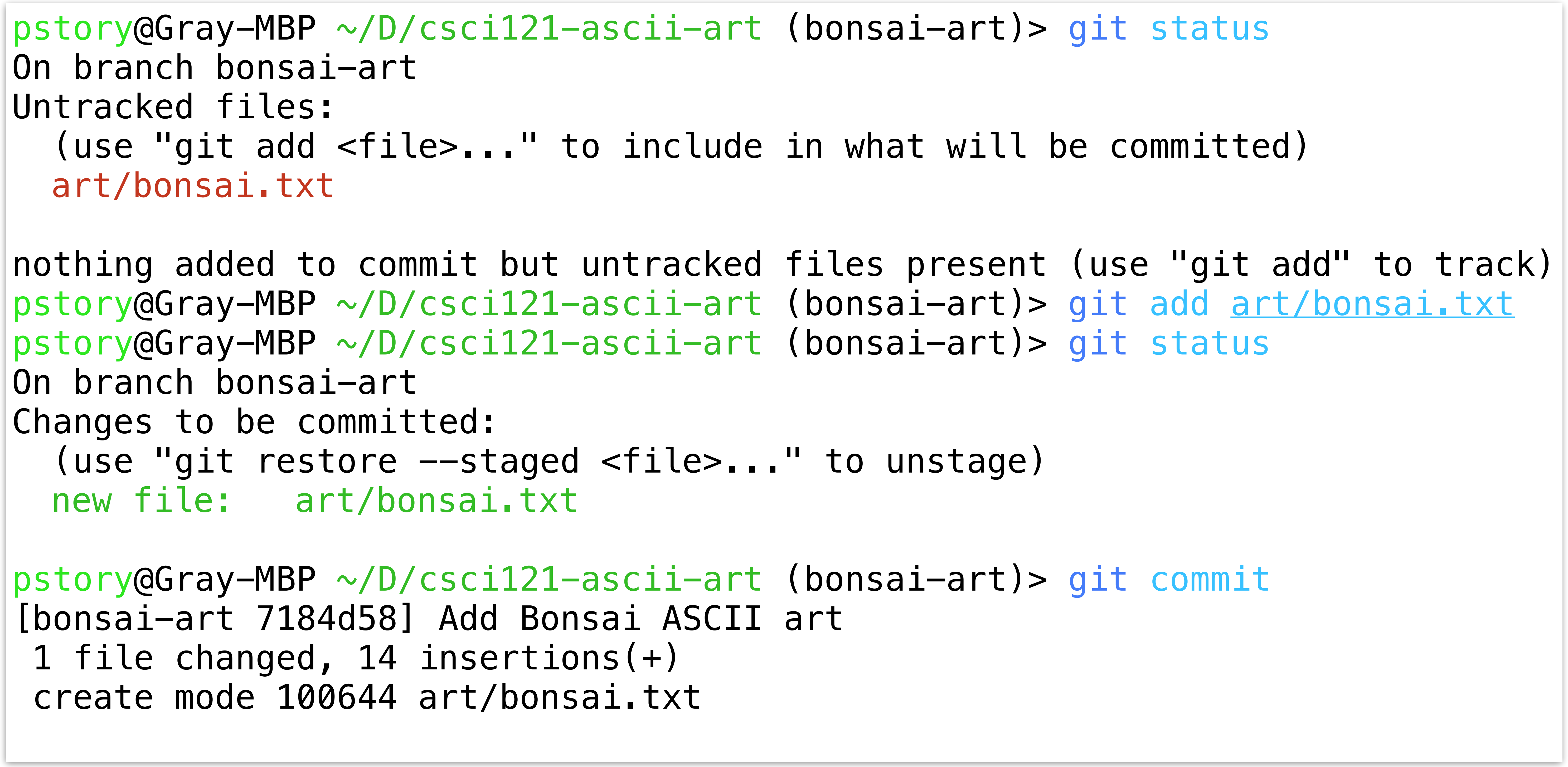
Add website config and two examples
Peter Story a556586 10 changed files +108 -0

.gitignore
README.md
art.js
art/earth.txt
art/owl.txt
docker-compose.yml
dot_env_example
index.html
index.js
nginx/default.development.conf

```
@@ -0,0 +1,3 @@
+.DS_Store
+__pycache__
+.env
```

Demo: Using Git on the Command-Line

- Clone my fork
- View history
- Create a branch
- Make changes
- Stage my changes
- Review my changes
- Commit my changes
- Push my changes
- Open a pull request



```
pstory@Gray-MBP ~/D/csci121-ascii-art (bonsai-art)> git status
On branch bonsai-art
Untracked files:
  (use "git add <file>..." to include in what will be committed)
	art/bonsai.txt

nothing added to commit but untracked files present (use "git add" to track)
pstory@Gray-MBP ~/D/csci121-ascii-art (bonsai-art)> git add art/bonsai.txt
pstory@Gray-MBP ~/D/csci121-ascii-art (bonsai-art)> git status
On branch bonsai-art
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
	new file:   art/bonsai.txt

pstory@Gray-MBP ~/D/csci121-ascii-art (bonsai-art)> git commit
[bonsai-art 7184d58] Add Bonsai ASCII art
 1 file changed, 14 insertions(+)
 create mode 100644 art/bonsai.txt
```

Git Command-Line Reference

Clone my fork	`git clone git@github.com:peterstory/csci121-ascii-art.git`

View history	`git log`
	`git log --all --decorate --oneline --graph --name-status`

Create a branch	`git checkout -b bonsai-art`
Stage my changes	`git add art.js art/bonsai.txt`

Review my changes	`git status`
	`git diff --cached`

Commit my changes	`git commit`

Push my changes	`git push`
	`git push --set-upstream origin bonsai-art`

Git Data Structures

Review: Hash Tables

- Basic operations:
  - `put(key, value)`
  - `get(key)`
  - `delete(key)`
- Compute and use a key's *hash*
  - `hash("This string is a key")` →
    `687498451` (32-bit, represented as an integer),
    `feacb7c9c8b87bbec18a6fd58dc4119829676c7a` (160-bit, represented as hex),
    etc.
- Operations are very fast: *constant order* average complexity

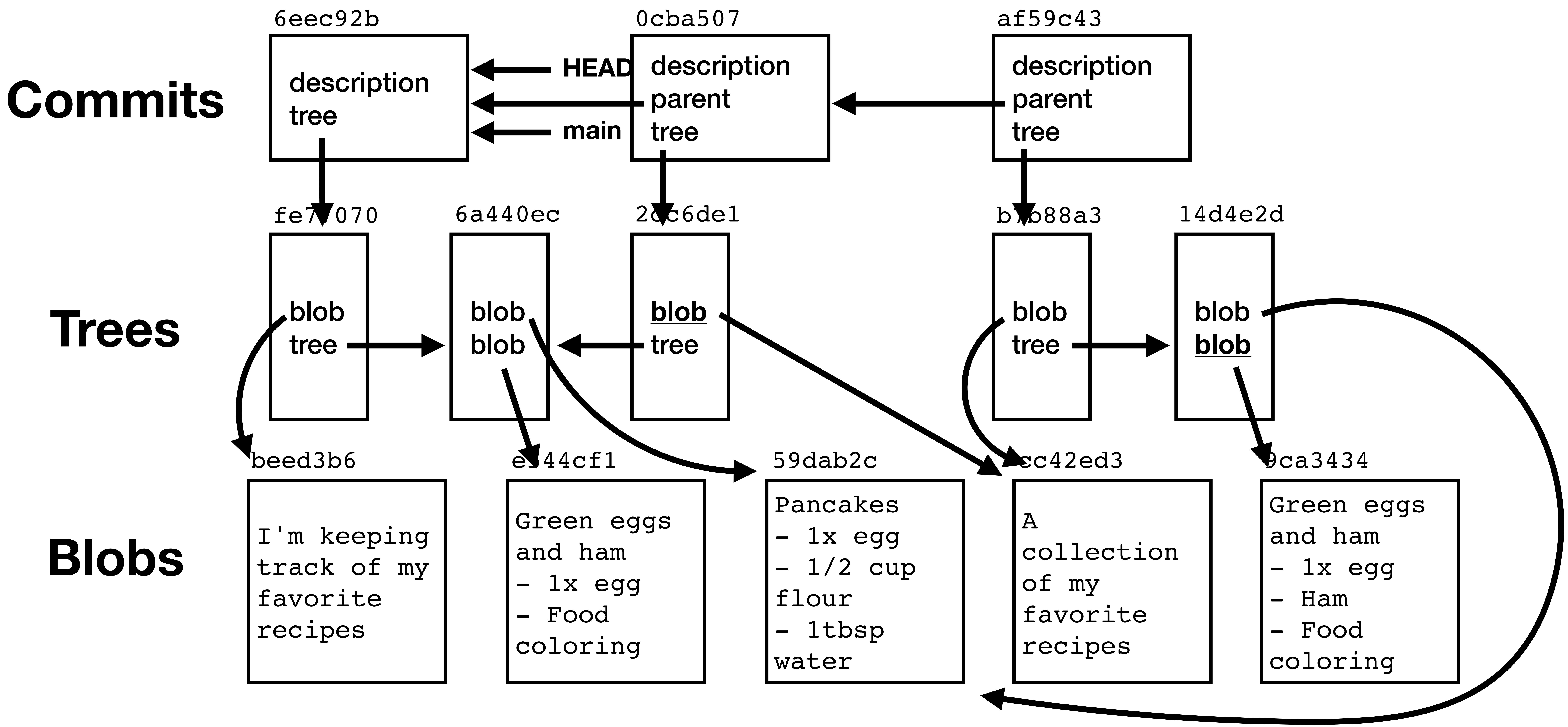
Git Data Structures

- Git stores data using SHA-1 checksums (i.e., a hash)
  - Files (i.e., blobs), directories (i.e., trees), commit data, etc.
- Data objects are stored on disk, with filenames based on the hash
  - If you know the hash, locating the file containing the object is a constant order operation!

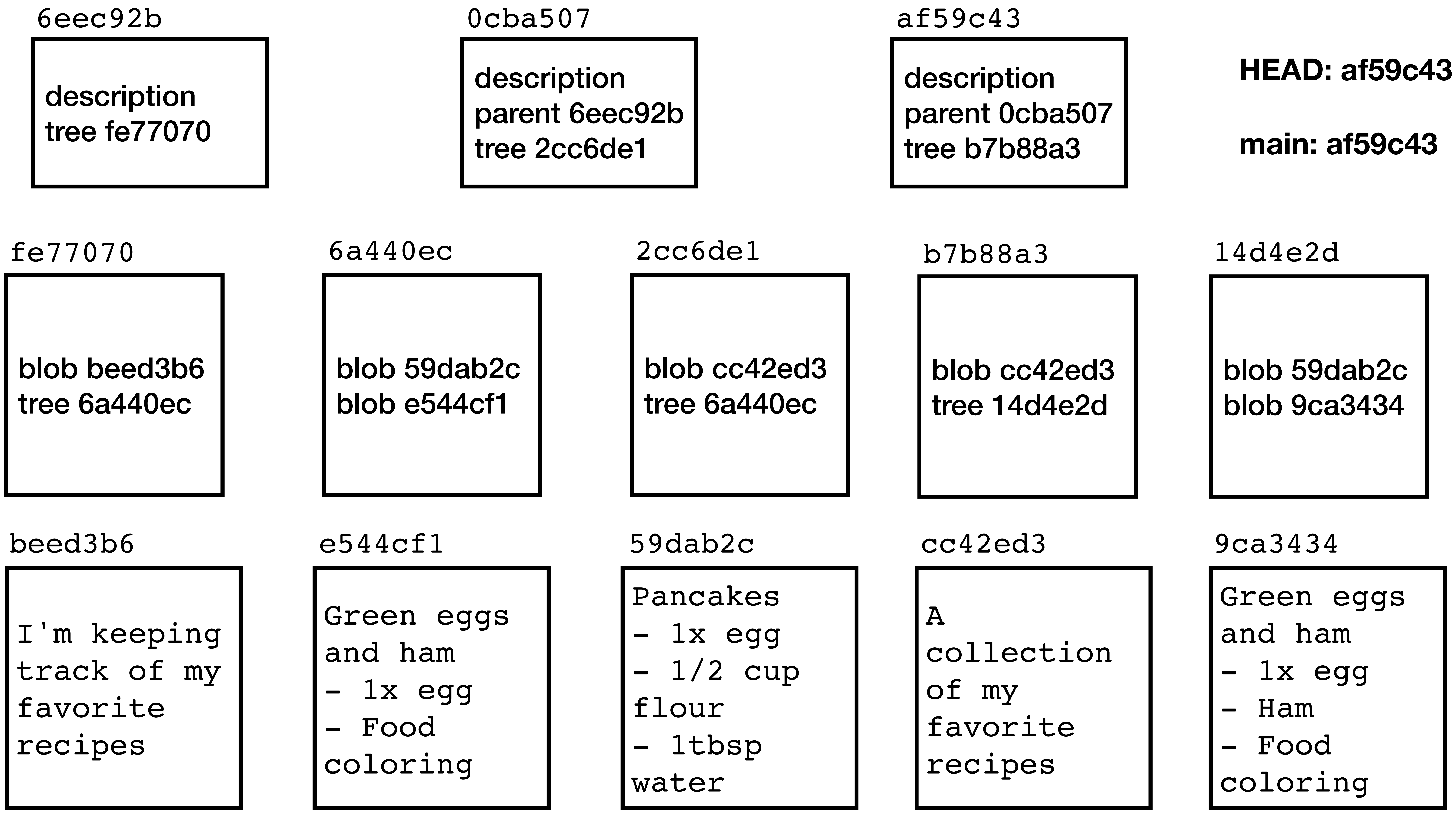
Example Git Repository

- readme.txt
- recipes/
  - green_eggs_and_ham.txt
  - pancakes.txt

How Git Objects Are Stored



How Git Objects Are Stored



```
6eec92b
description
tree fe77070
```

```
0cba507
description
parent 6eec92b
tree 2cc6de1
```

```
af59c43
description
parent 0cba507
tree b7b88a3
```

HEAD: af59c43

main: af59c43

```
fe77070
blob beed3b6
tree 6a440ec
```

```
6a440ec
blob 59dab2c
blob e544cf1
```

```
2cc6de1
blob cc42ed3
tree 6a440ec
```

```
b7b88a3
blob cc42ed3
tree 14d4e2d
```

```
14d4e2d
blob 59dab2c
blob 9ca3434
```

```
beed3b6
I'm keeping
track of my
favorite
recipes
```

```
e544cf1
Green eggs
and ham
- 1x egg
- Food
coloring
```

```
59dab2c
Pancakes
- 1x egg
- 1/2 cup
flour
- 1tbsp
water
```

```
cc42ed3
A
collection
of my
favorite
recipes
```

```
9ca3434
Green eggs
and ham
- 1x egg
- Ham
- Food
coloring
```

Demo

Demo: Inspecting Git Objects

- `ls .git/objects/*`
- `git log # To identify commit hash`
- `git cat-file -t COMMIT_HASH`
- `git cat-file -p COMMIT_HASH`
- `...`

Based on: https://www.youtube.com/watch?v=P6jD966jzlk

Git Challenges

Moving Data Between Locations

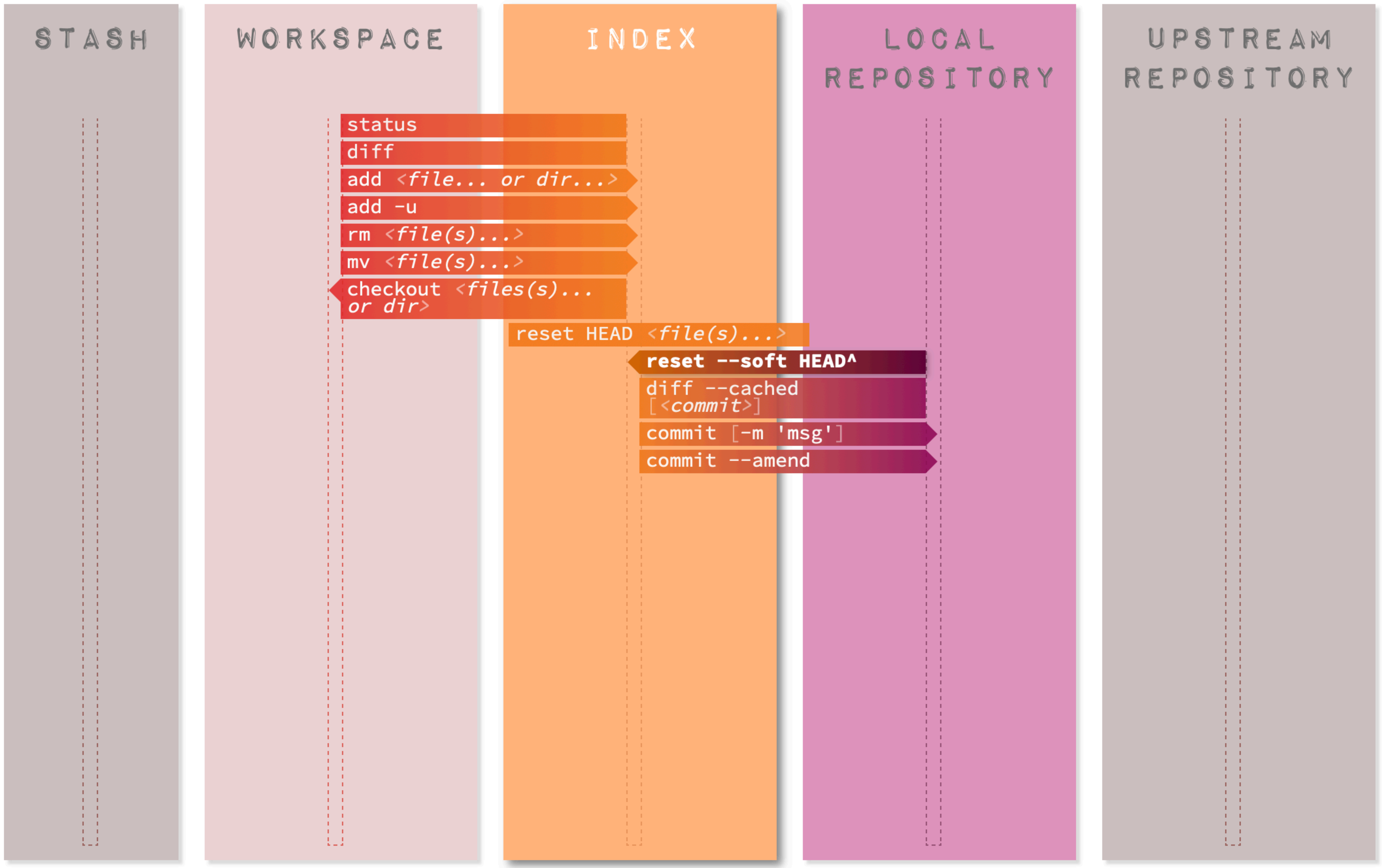
- Major challenge: moving data between different locations:
  - Upstream repository (e.g., ClarkuCSCI/csci121-ascii-art on GitHub)
  - Fork of repository (e.g., peterstory/csci121-ascii-art on GitHub)
  - Local clone of fork (i.e., on your laptop)
    - Repository
    - Index
    - Workspace
- Necessary complexity, to support distributed version control, but still a challenge!

GIT CHEATSHEET

http://www.ndpsoftware.com/git-cheatsheet.html

AN INTERACTION FROM NDP SOFTWARE

en fr it es de 简中 繁中 한국어

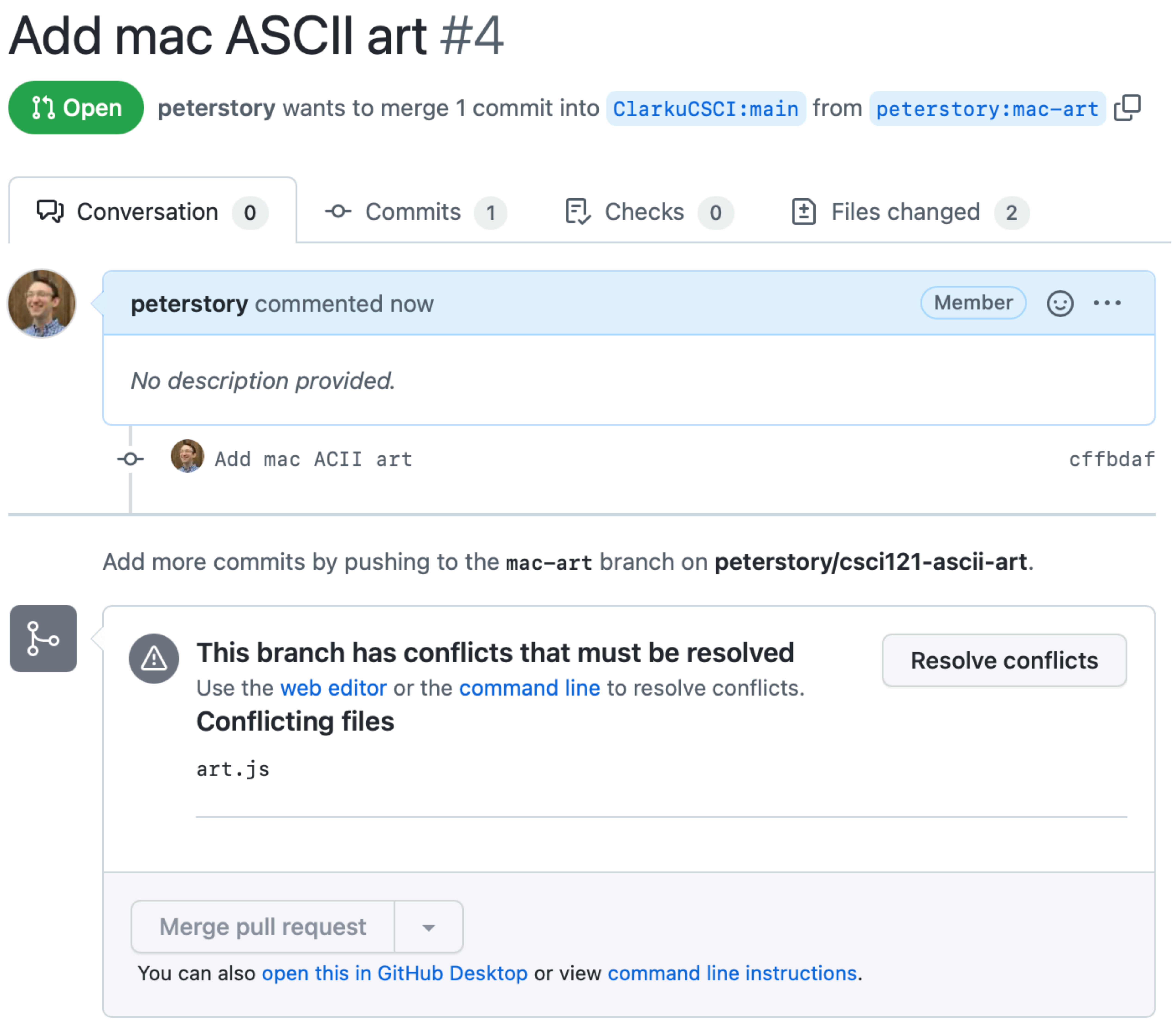


```
status
diff
add <file... or dir...>
add -u
rm <file(s)...>
mv <file(s)...>
checkout <files(s)...
or dir>
reset HEAD <file(s)...>
reset --soft HEAD^
diff --cached
[<commit>]
commit [-m 'msg']
commit --amend
```

Merge Conflicts

- What happens if two people edit the same file (e.g., art.js)?
    - Keep one person's edits? Combine their changes in some way?
- Merges *can* be easy to resolve
- Messy merges are not git's fault – it's a problem of team communication
    - People shouldn't be simultaneously making conceptually interfering changes
    - Use code beautifiers to avoid "fighting" over formatting decisions

Merge Conflict



Add mac ASCII art #4

Open peterstory wants to merge 1 commit into `ClarkuCSCI:main` from `peterstory:mac-art`

peterstory commented now

No description provided.

`Add mac ACII art` `cffbdaf`

Add more commits by pushing to the `mac-art` branch on **peterstory/csci121-ascii-art**.

This branch has conflicts that must be resolved

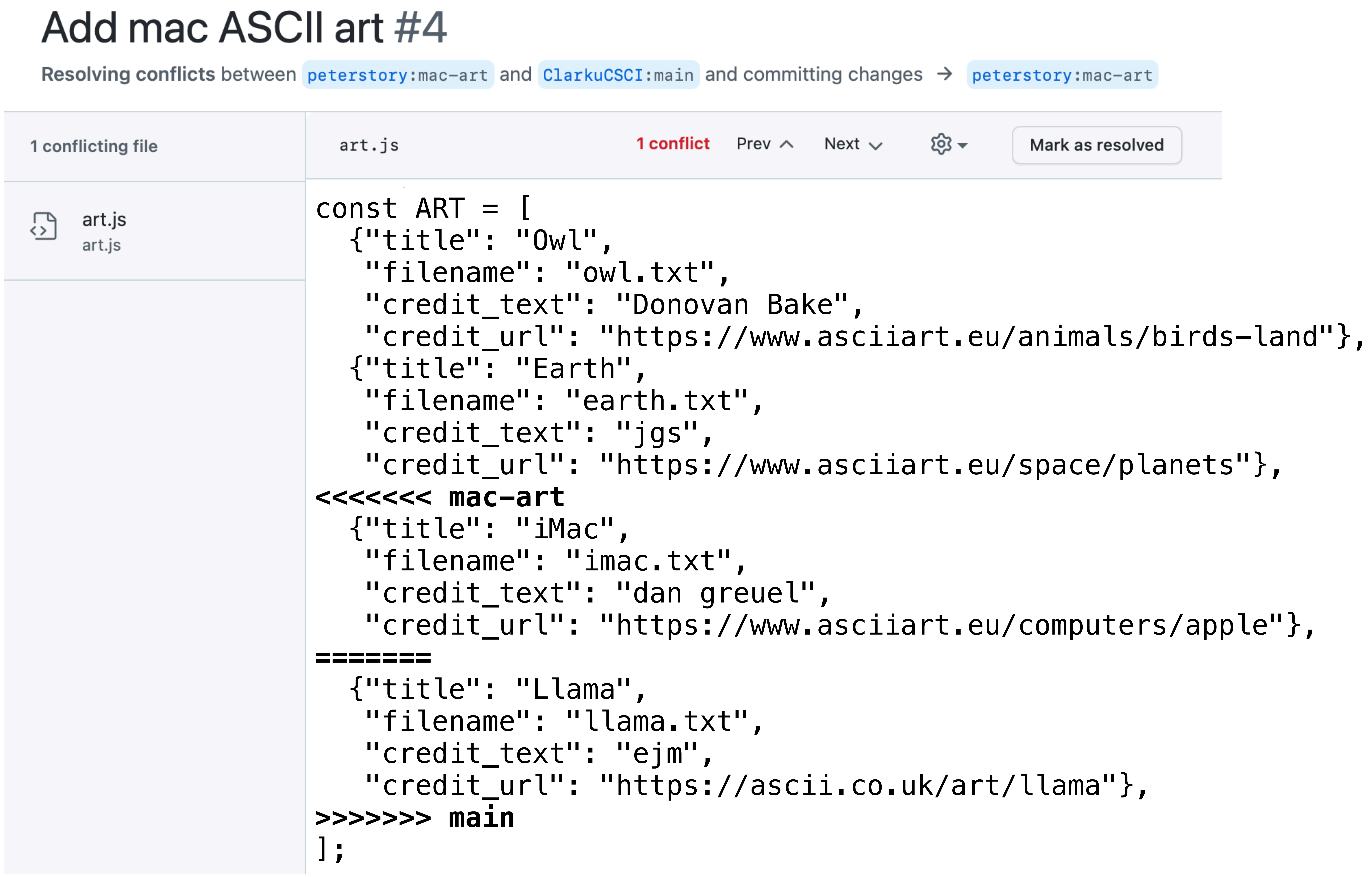
Use the web editor or the command line to resolve conflicts.

Conflicting files

`art.js`

You can also open this in GitHub Desktop or view command line instructions.

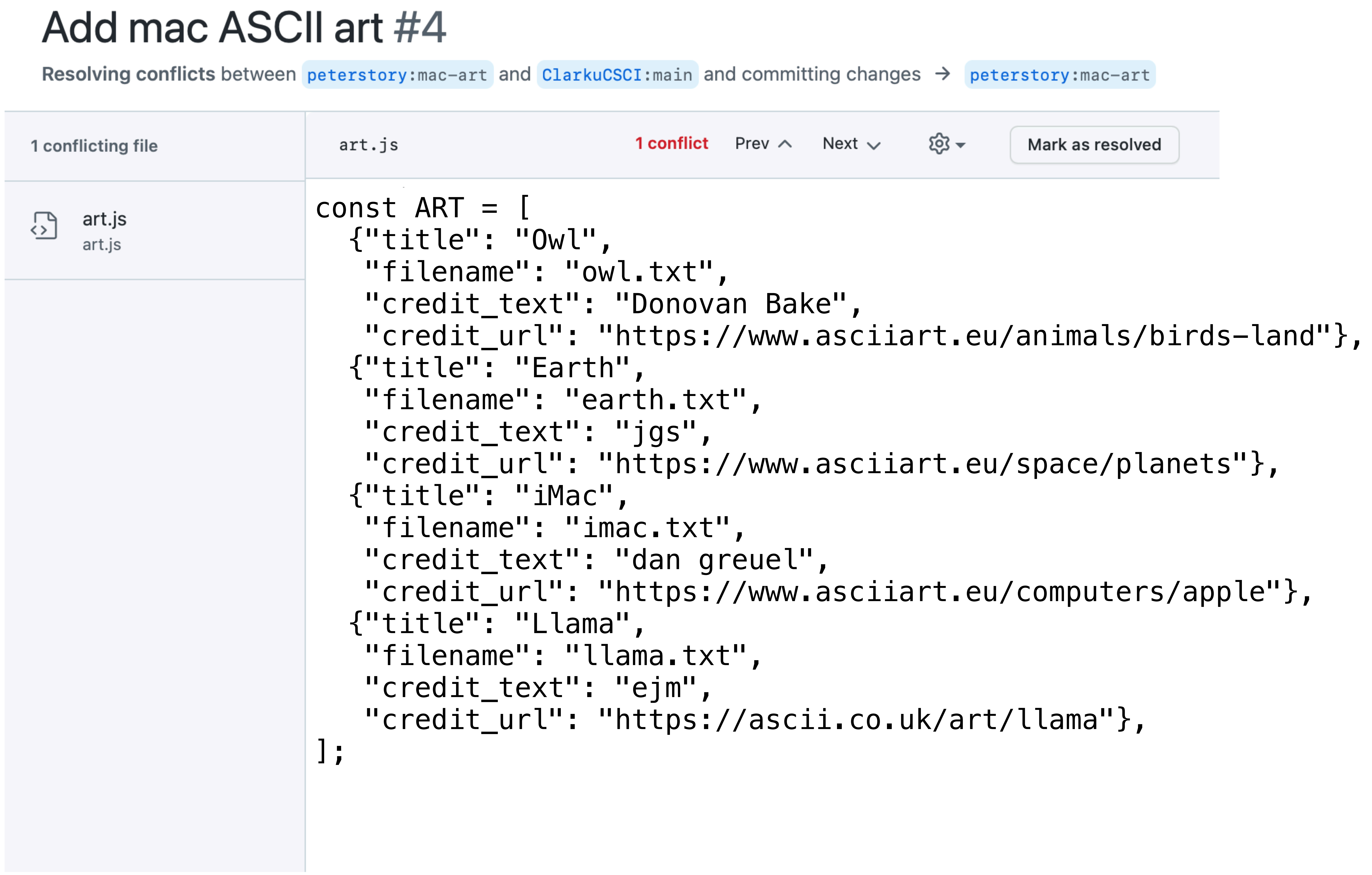
Merge Conflict



Add mac ASCII art #4
Resolving conflicts between peterstory:mac-art and ClarkuCSCI:main and committing changes → peterstory:mac-art
1 conflicting file
art.js
art.js
art.js
1 conflict
Prev
Next
Mark as resolved

```
const ART = [
  {"title": "Owl",
   "filename": "owl.txt",
   "credit_text": "Donovan Bake",
   "credit_url": "https://www.asciiart.eu/animals/birds-land"},
  {"title": "Earth",
   "filename": "earth.txt",
   "credit_text": "jgs",
   "credit_url": "https://www.asciiart.eu/space/planets"},
<<<<<<< mac-art
  {"title": "iMac",
   "filename": "imac.txt",
   "credit_text": "dan greuel",
   "credit_url": "https://www.asciiart.eu/computers/apple"},
=======
  {"title": "Llama",
   "filename": "llama.txt",
   "credit_text": "ejm",
   "credit_url": "https://ascii.co.uk/art/llama"},
>>>>>>> main
];
```

Merge Conflict Resolution



Add mac ASCII art #4

Resolving conflicts between `peterstory:mac-art` and `ClarkuCSCI:main` and committing changes → `peterstory:mac-art`

```
const ART = [
  {"title": "Owl",
   "filename": "owl.txt",
   "credit_text": "Donovan Bake",
   "credit_url": "https://www.asciiart.eu/animals/birds-land"},
  {"title": "Earth",
   "filename": "earth.txt",
   "credit_text": "jgs",
   "credit_url": "https://www.asciiart.eu/space/planets"},
  {"title": "iMac",
   "filename": "imac.txt",
   "credit_text": "dan greuel",
   "credit_url": "https://www.asciiart.eu/computers/apple"},
  {"title": "Llama",
   "filename": "llama.txt",
   "credit_text": "ejm",
   "credit_url": "https://ascii.co.uk/art/llama"},
];
```

Undoing Changes

- Recommended: make a new commit that puts things in the desired state
- Potentially dangerous: editing the commit history
  - If the commits are only local, they can be reworked without affecting others
  - If the commits are visible to the rest of the team, reworking the history will interfere with others' branches, and so is **not** recommended
  - Sometimes necessary (e.g., if you commit sensitive data to a repo)

On undoing, fixing, or removing commits in git: http://sethrobertson.github.io/GitFixUm/fixup.html

Editing Commit History

Oh no – real customer data was used, including their credit card numbers!
7184d58
Tuesday @ 1:00pm
Add Unit Test for Checkout Bug
cffbdaf
Tuesday @ 2:00pm
Add Sample Customer Data for Checkout Bug
9487242
Tuesday @ 3:00pm
Fix for Checkout Bug
1c10d24
Tuesday @ 4:00pm
Fix for Homepage Styling

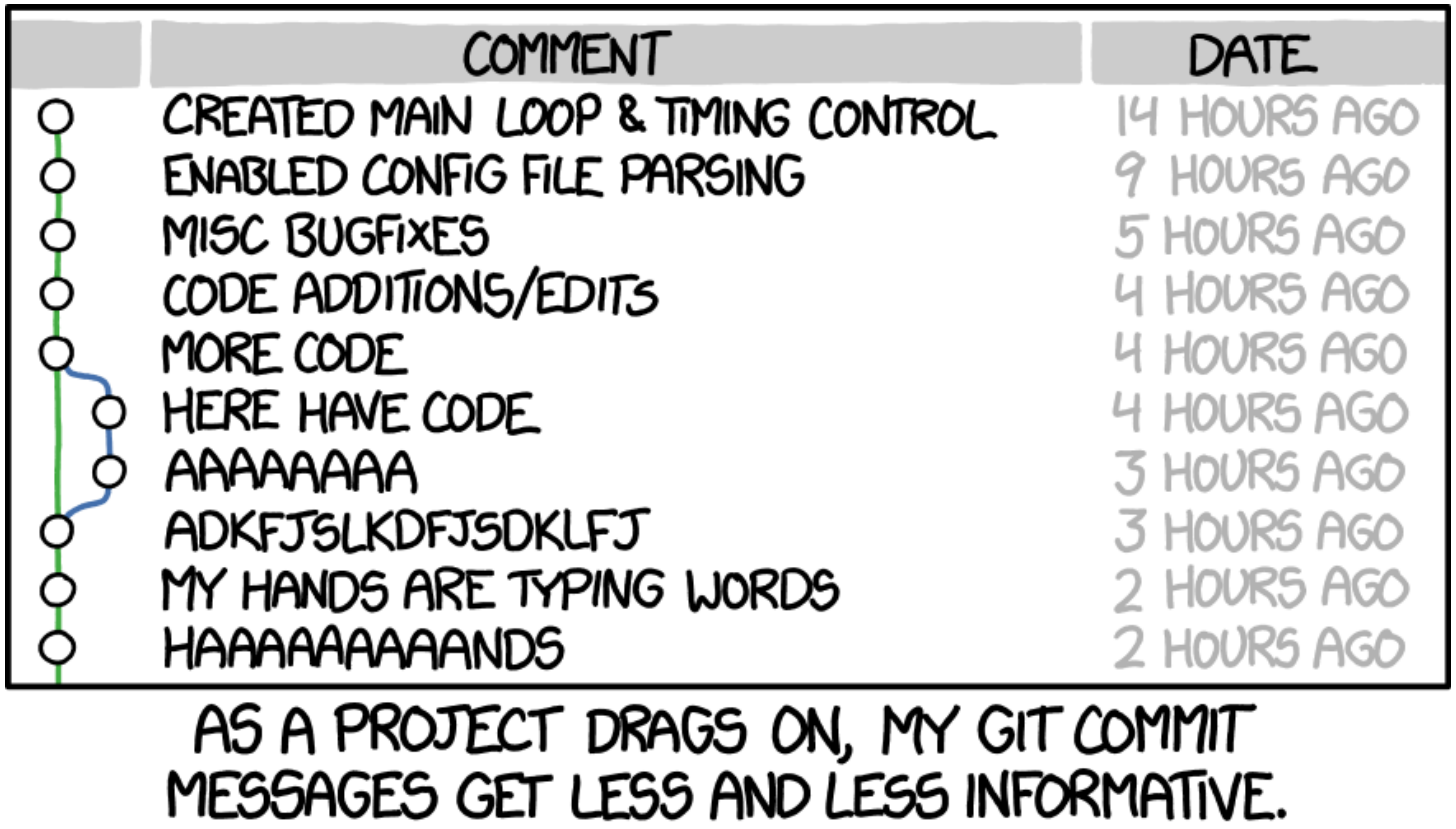
If this commit is changed, the commit hashes of all subsequent commits will change, too
7184d58
Tuesday @ 1:00pm
Add Unit Test for Checkout Bug
feacb7c
Tuesday @ 2:00pm
Add Sample Customer Data for Checkout Bug
a556586
Tuesday @ 3:00pm
Fix for Checkout Bug
3471f41
Tuesday @ 4:00pm
Fix for Homepage Styling

Git Advice

Be Careful What You Commit

- Generally, don't commit build artifacts (e.g., .class files)
- Definitely don't commit:
  - Temporary files (e.g., .DS_Store, __pycache__, etc.)
  - Passwords, SSH keys, or any other sensitive information
- Staging files one-at-a-time will help you avoid mistakes
- Use a .gitignore file to list files which shouldn't be committed

Write Good Commit Messages



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

https://xkcd.com/1296/

Write Good Commit Messages

1. Separate subject from body with a blank line
2. Limit the subject line to 50 characters
3. Capitalize the subject line
4. Do not end the subject line with a period
5. Use the imperative mood in the subject line
    - If applied, this commit will: <u>your subject line here</u>
6. Wrap the body at 72 characters
7. Use the body to explain what and why vs. how

https://cbea.ms/git-commit/

Short Commit Message

```
Add Initialize Git Submodules to setup setups
```

More Detailed Commit Message

```
Set DJANGO_DEBUG to False by default

Good to have secure defaults. When DJANGO_DEBUG is True,
DB passwords, etc. can be seen on error pages.
```

Merge Commit, Summarizing Component Commits

```
Fix APK downloading (#332)

* Update the submodule version of privacy-practice-analysis,
  to fix APK downloading
* Update the base image to Ubuntu 18.04 (to fix APK
  downloading) and update nodejs (as the older version failed
  to install)
* Update the submodule version of privacy-policy-retrieval,
  to remove pytest-catchlog, which was causing test errors.
* Update pytest (as the latest version includes the functionality
  of pytest-catchlog)
* Update psycopg2 (as an update was required, presumably
  because the postgresql-client version changed when we
  updated Ubuntu)

Fixes #331
```

Use GitHub to Organize Your Work

- Create issues for changes you plan to make
- Document research, your thought process, etc. in the issue
- Work on code in a branch named after your issue (e.g., issue-12)
- Reference the issue number in your commits
    - The commits will appear on the issue page!

Course Evaluations

- Available from the “My eUWTE’s” tab on Canvas
- Current response rate is only 25%
- See “Giving constructive feedback on course evaluations” link on the course website
- Please complete by 12:05pm, after which I’ll announce the lab