

Git Internals

OR: HOW I LEARNED TO STOP WORRYING
AND LOVE THE REBASE



Andrew Wildman

May 10, 2019

Goals



Goals



Better mental model

- Where and how does git store data files?
- What makes up a commit? How do commits form a history?
- Why does `git rebase` create new commits?

Goals



Better mental model

- Where and how does git store data files?
- What makes up a commit? How do commits form a history?
- Why does `git rebase` create new commits?

Git Book

A Plumber's Guide to Git

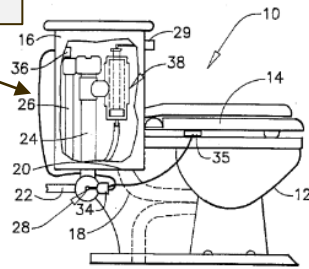
A Hacker's Guide to Git

- <https://git-scm.com/book/en/v2>
- <https://alexwlchan.net/a-plumbers-guide-to-git/>
- <https://wildlyinaccurate.com/a-hackers-guide-to-git/>

Plumbing vs. Porcelain

U.S. Patent Jul. 12, 1988 Sheet 1 of 2 4,756,031

Our focus today

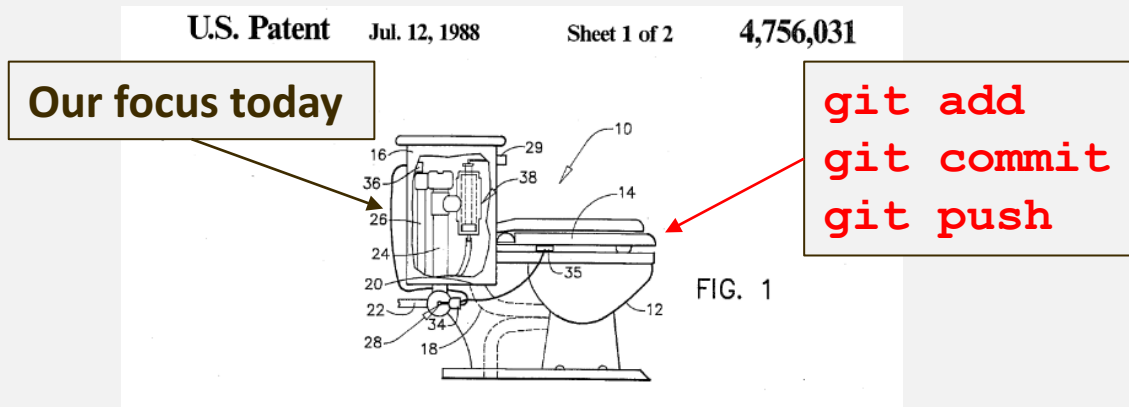


git add
git commit
git push

Plumbing vs. Porcelain

Not covering

- Basics of git, such as:
 - `git add/commit/push`
- Git hooks
- Internet git repositories (GitHub, GitLab, Bitbucket)
- Diffs and packfiles
- Communication protocols (behind `git pull`)
- Git tools, such as:
 - `git bisect/rerere/stash`




Agenda

- 1. Git Object Store**
- 2. Blobs and Trees**
- 3. Commits**
- 4. References**
- 5. Attendees' Choice(?)**

Git Object Store

Exercises

1. Create a new directory, and initialize git
2. Look inside the .git folder. Make sure you understand its contents.
3. Create a file, and put some contents in it.
- 2 x  4. Save that file to the Git object store. Find it in the Git object database.
5. Use a plumbing command to see the contents of the object you just saved.
6. Make a change to a file. Save the changed file to the object database.
 1. What do you expect to see in .git/objects?
 2. Were you right?
7. *Bonus:* Delete a file, and recreate it from the Git object database.
8. *Bonus:* Change the name of a file, and save it to the object database.
 1. What do you expect to see in .git/objects?
 2. Were you right?

Blobs and Trees

Directory structure

Blob
echo "Hello world"

Blob
#!/usr/bin/env python
print("Hello world")

File name?
Path?
Permissions?

Directory structure



Exercises

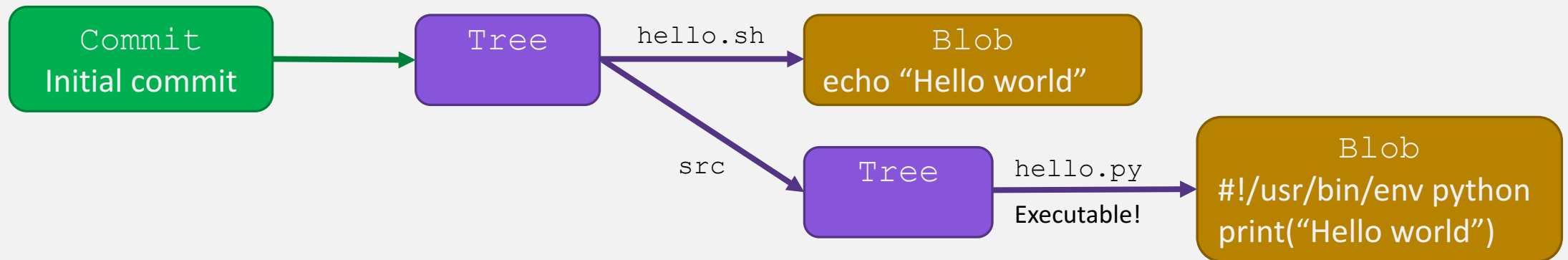
1. Add a file from part 1 to the index.
2. Look inside the .git folder. Find the index file.
3. Check the contents of the index with a plumbing command.
 1. Use `git status` to confirm the contents of the index.
4. Save the index as a tree in the object database. Find the file in the database.
5. Look at the contents of the tree object. Make sure you understand each line.

Repeat 1-5 until you are comfortable saving trees to the database

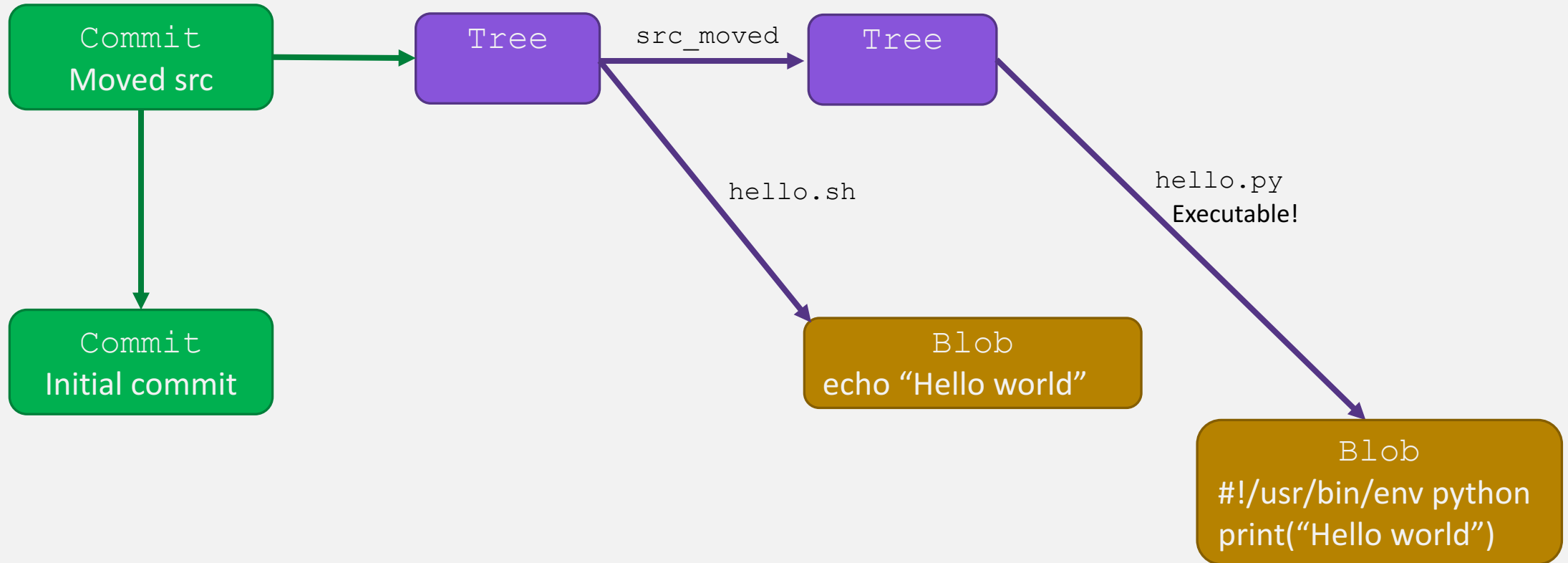
6. Change a file and add it to the index. Save the tree to the object database.
 1. What do you expect to be the contents of the tree?
 2. Inspect the object file. Were you right?
7. Save a subdirectory with files to your database. Inspect both the root tree and subtree. Make sure you understand each of the trees' contents.

Commits

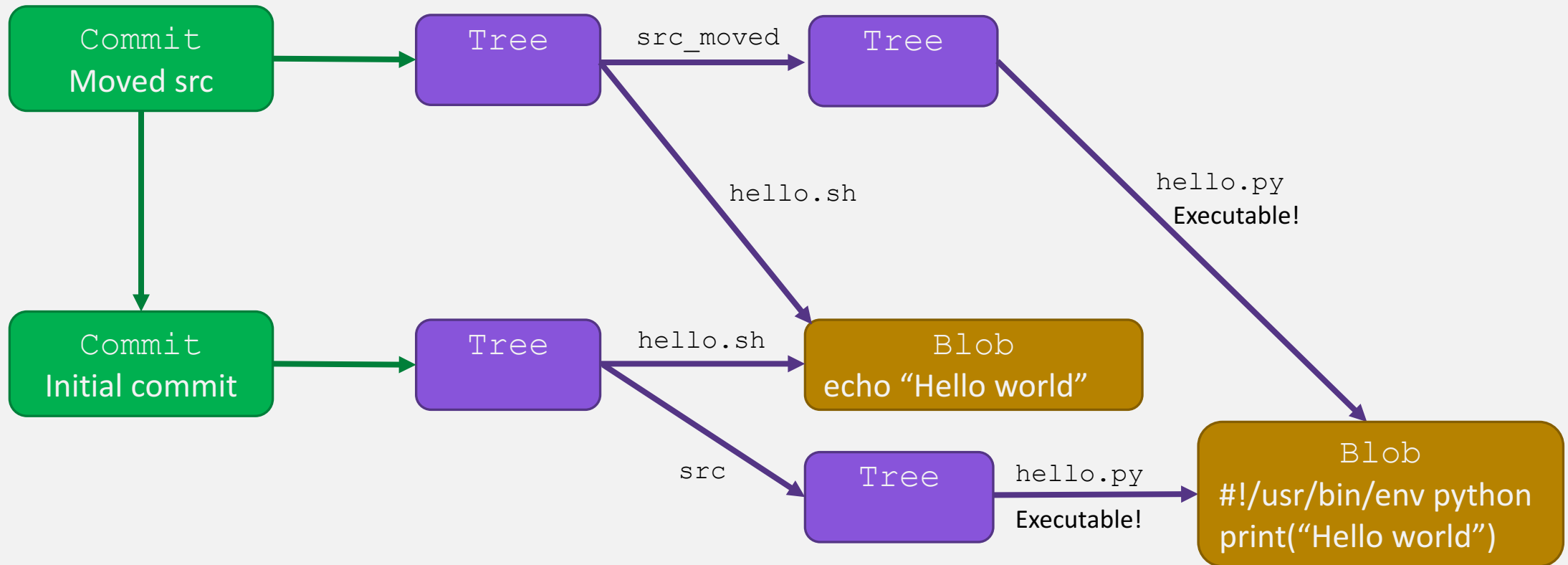
Linear history



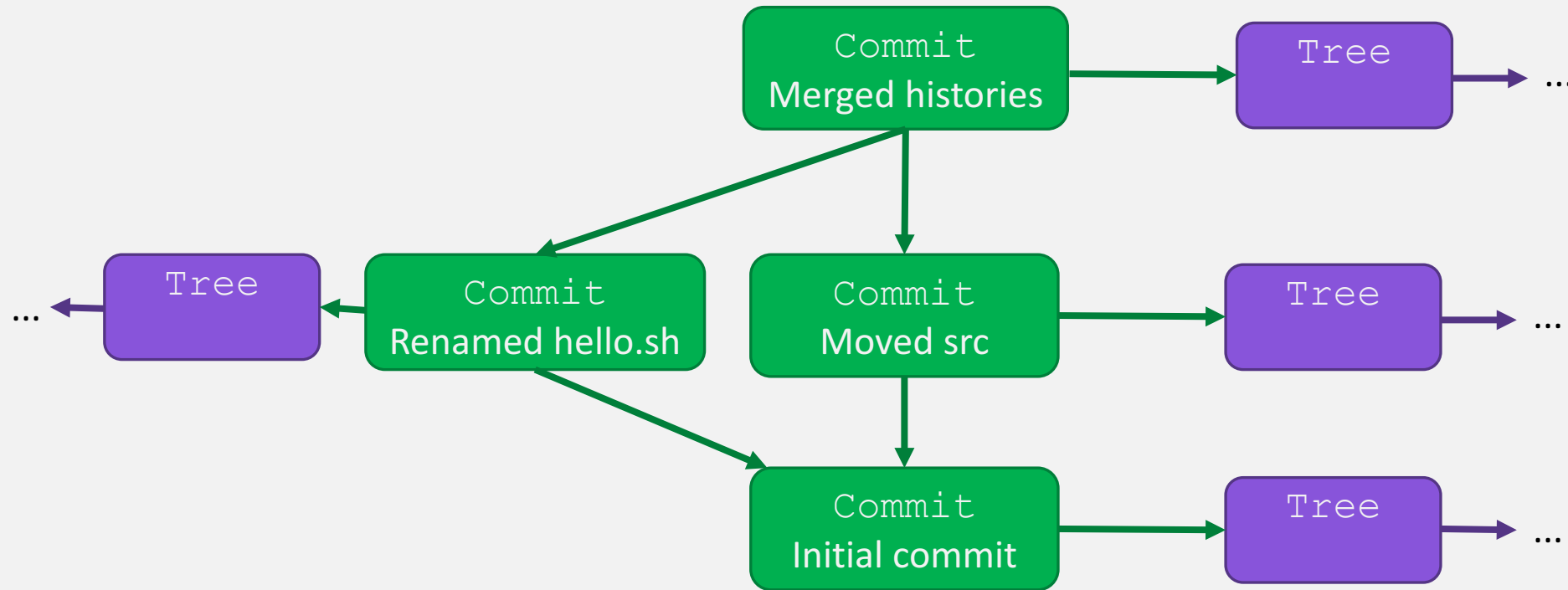
Linear history



Linear history



Merging histories



Exercises

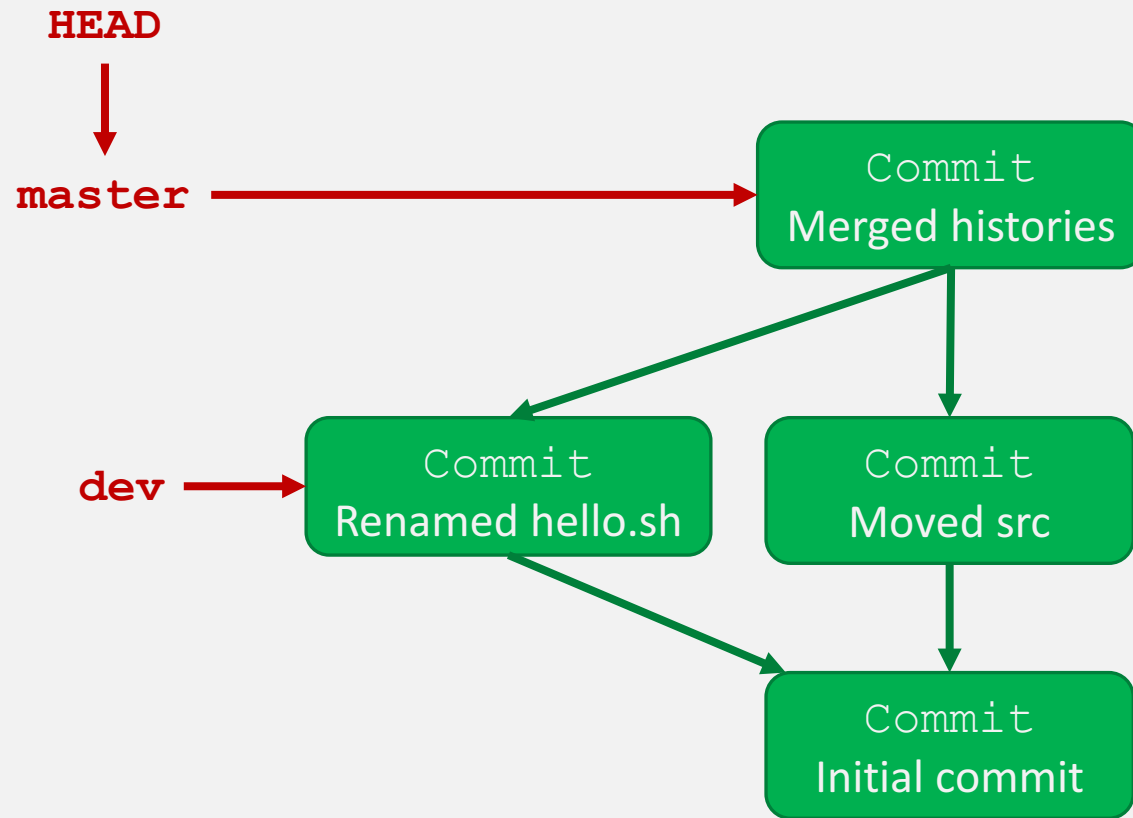
1. Create a commit from a tree from part 2.
2. Find your commit object in the object database.
3. Check the contents of the commit you created

Repeat the above to create a linear history

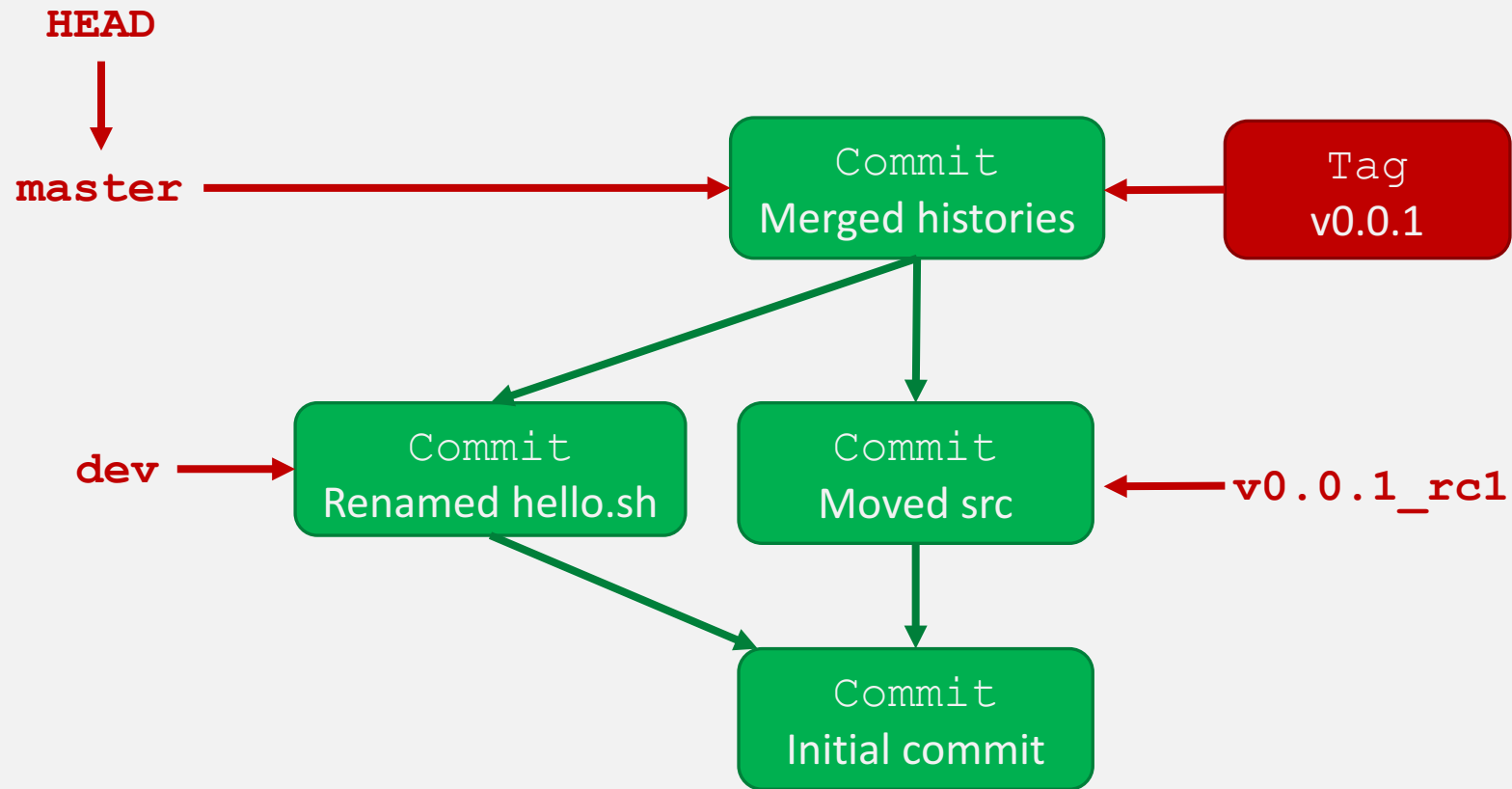
4. Check the commit history with `git log`
5. *Bonus:* Create a second commit off of your initial commit. Create a merge commit from your two “branches.”

References

Human accessible history



Permanent human accessible history



Exercises

1. Look inside your `.git/refs` directory. Check that there are two subdirectories.
2. Look inside the `.git` folder. Find the index file.
3. Check the contents of the index with a plumbing command.
 1. Use `git status` to confirm the contents of the index.
4. Save the index as a tree in the object database. Find the file in the database.
5. Look at the contents of the tree object. Make sure you understand each line.

Repeat 1-5 until you are comfortable saving trees to the database

6. Change a file and add it to the index. Save the tree to the object database.
 1. What do you expect to be the contents of the tree?
 2. Inspect the object file. Were you right?
7. Save a subdirectory with files to your database. Inspect both the root tree and subtree. Make sure you understand each of the trees' contents.

Example: Rebase

Recap
