



# git

- Free, open source **distributed version control system**
- **Record changes** to a set of files over time
- Promotes **rapid experimentation**
- **Collaborate** with other developers

# Who uses git?

Google

facebook.

Microsoft

twitter

LinkedIn

NETFLIX



PostgreSQL



# The Working Directory

Our project is a set of files in a **directory** that contains:

- Text files (code)
- Sub-directories

We'd like to be able to store **versions** (snapshots) of our code

```
my_project
├── Makefile
├── cstart.c
└── libmypi
    ├── Makefile
    ├── backtrace.c
    ├── console.c
    ├── fb.c
    ├── gl.c
    ├── gpio.c
    ├── keyboard.c
    ├── malloc.c
    ├── printf.c
    ├── shell.c
    ├── strings.c
    └── timer.c
└── main.c
└── memmap
└── start.s

1 directory, 17 files
```

# Recording Changes

Save a snapshot of your working directory at any time

```
› git add <file>  
› git commit -m "Passed Test 1"
```

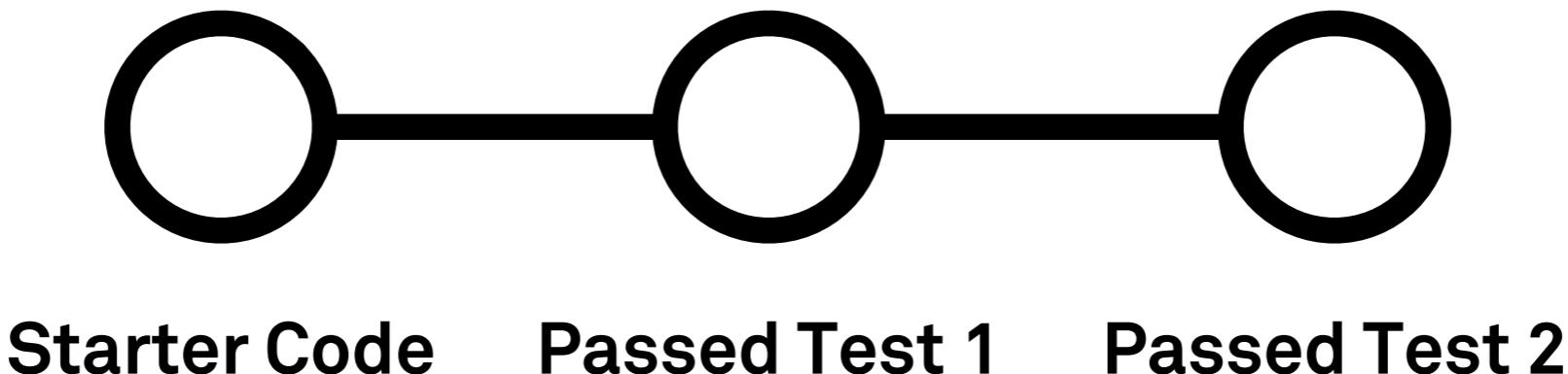


**Starter Code    Passed Test 1**

# Recording Changes

**Strategy:** Save a snapshot whenever you pass a new test

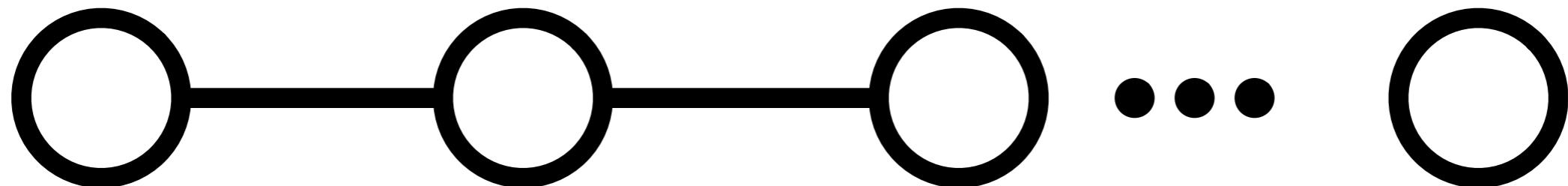
```
› git add <file>  
› git commit -m "Passed Test 2"
```



# Recording Changes

**Strategy:** Save a snapshot whenever you pass a new test

```
› git add <file>  
› git commit -m "Finished Basic!"
```

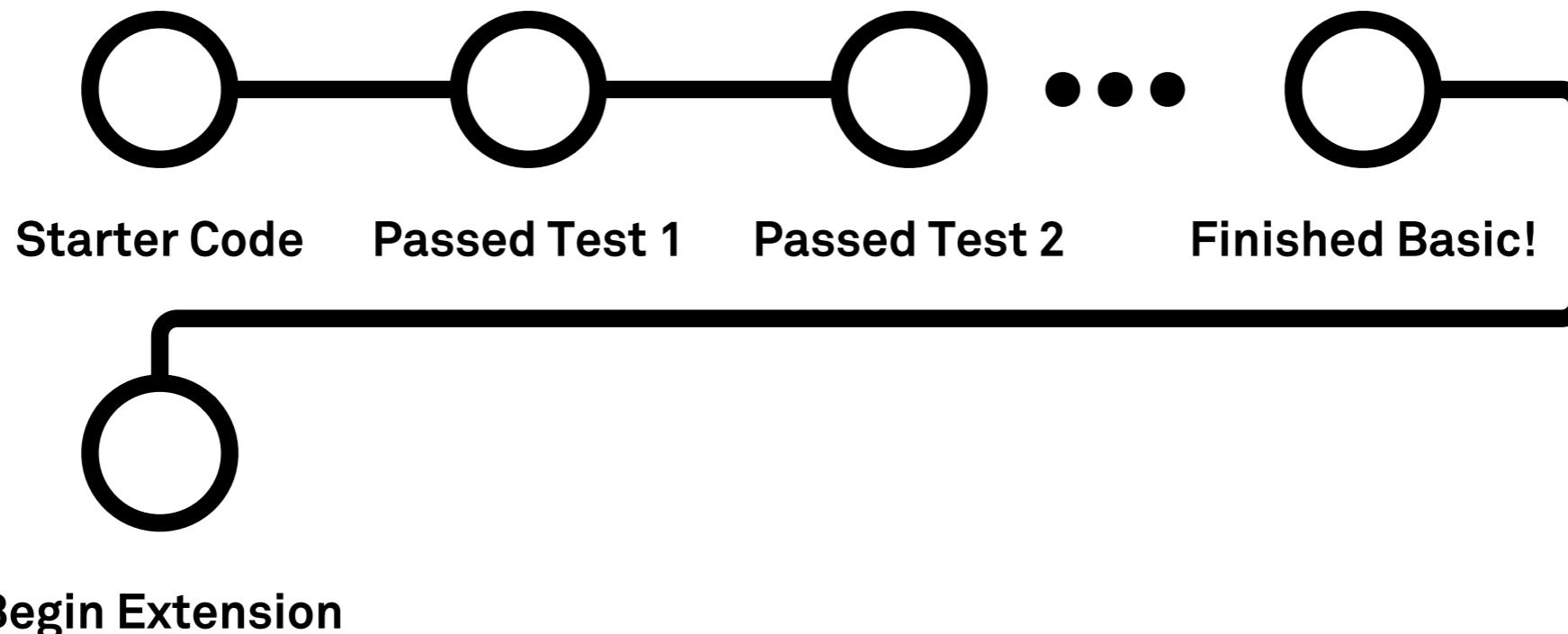


**Starter Code**    **Passed Test 1**    **Passed Test 2**    **Finished Basic!**

# Recording Changes

Begin working on your extension

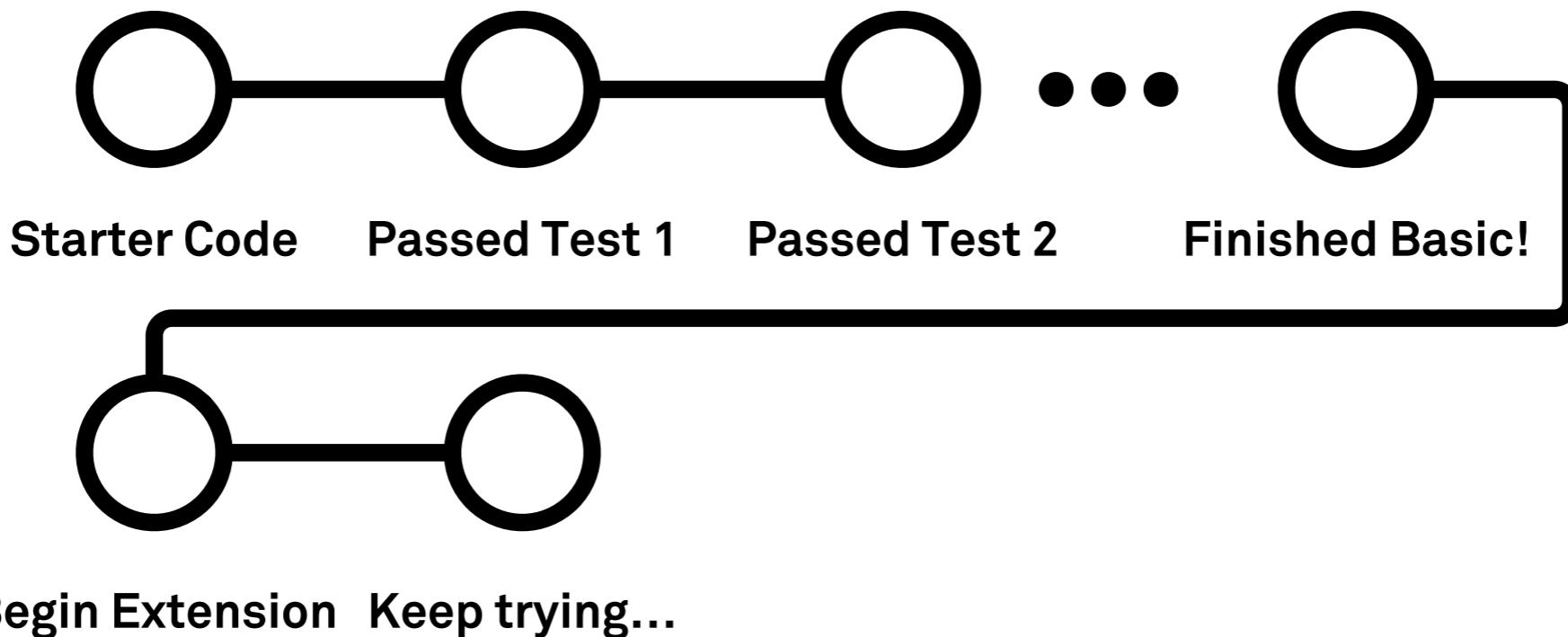
```
› git add <file>  
› git commit -m "Begin Extension"
```



# Recording Changes

Still isn't working

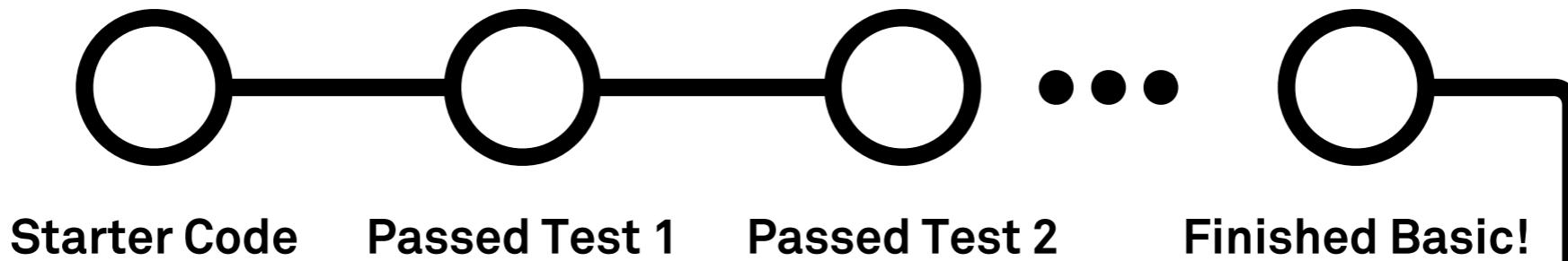
```
› git add <file>  
› git commit -m "Keep trying..."
```



# Recording Changes

Tuesday at 4:30pm

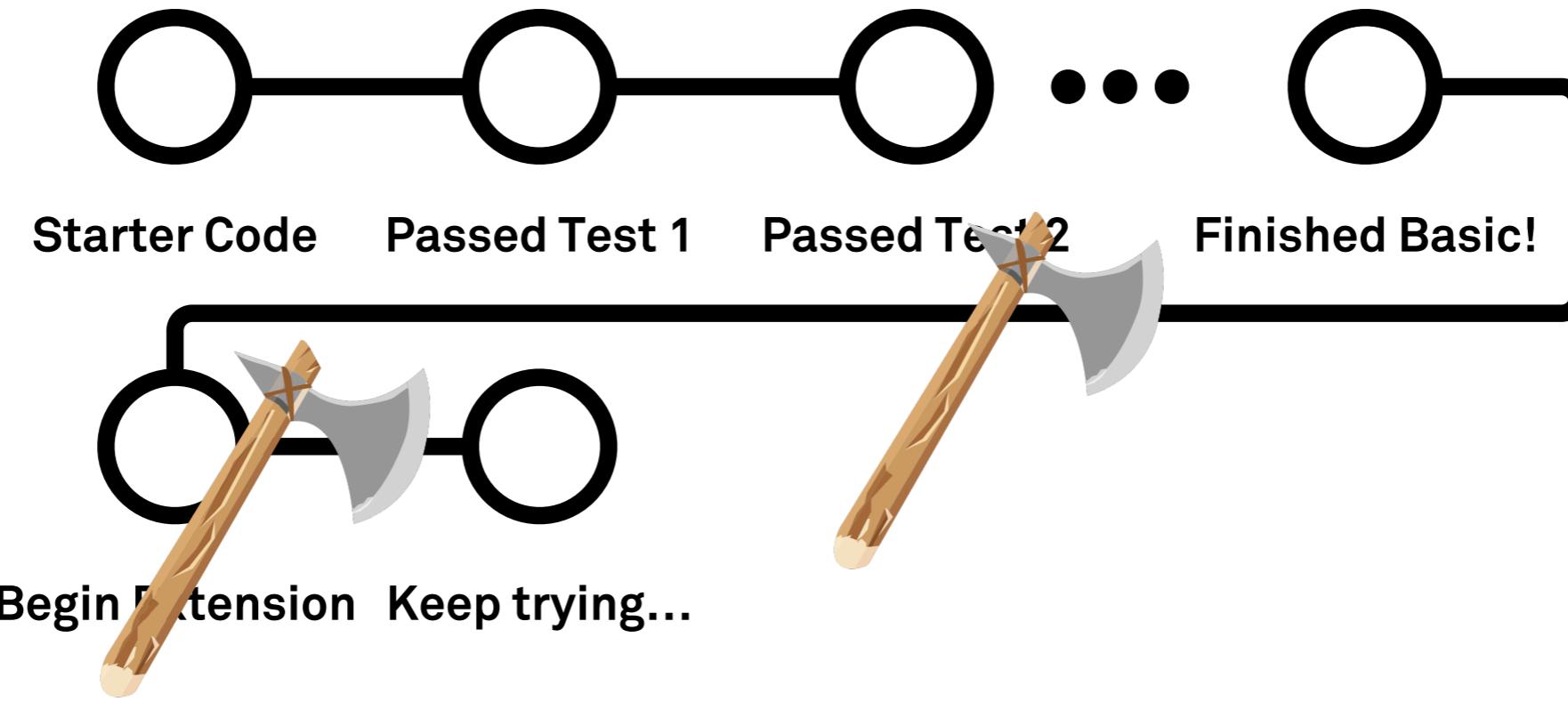
```
› git revert <commit1>
```



# Recording Changes

Tuesday at 4:30pm

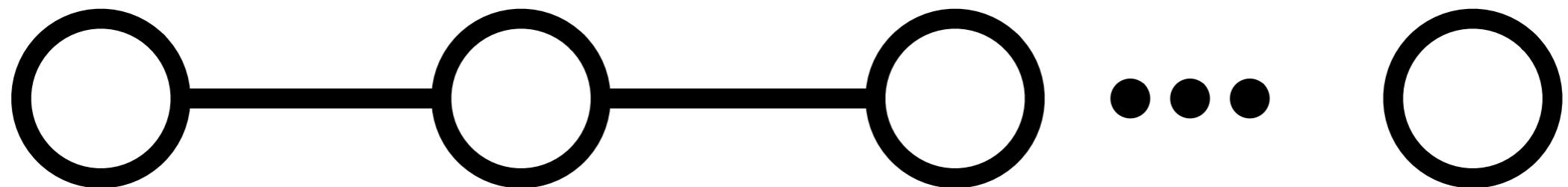
```
› git revert <commit1>  
› git revert <commit2>
```



# Recording Changes

Tuesday at 4:31pm

```
› git push
```



**Starter Code**

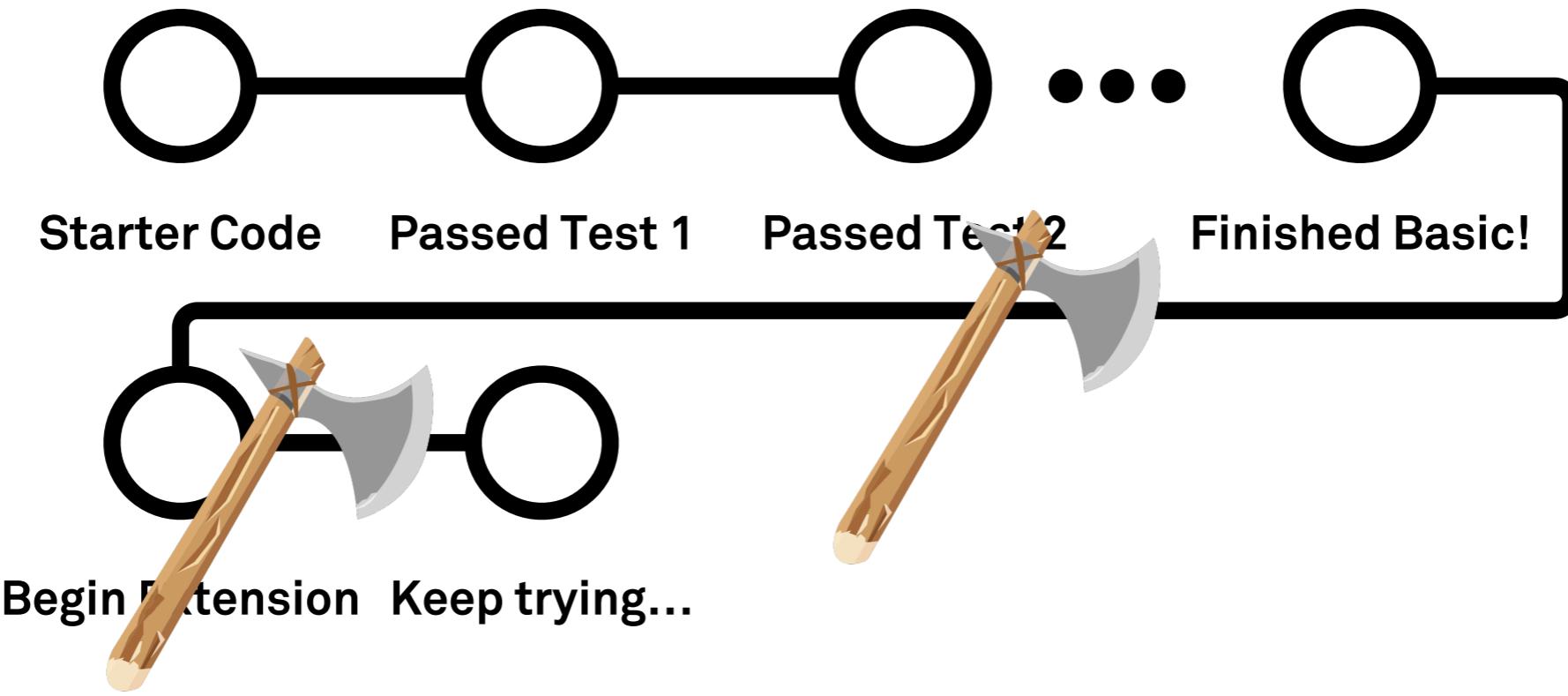
**Passed Test 1**

**Passed Test 2**

**Finished Basic!**

# Recording Changes

- Keeping a change history enables experimentation
- You can always revert to a good state!
- What are some problems with this approach?



## Revert .gitignore changes as they didn't work

[Browse files](#)

by master (#30)

 leonardt committed on Jul 27, 2017

1 parent e979765

commit 6234bffa7f54fcfa796f7ab23ce9bb353f31fdfe

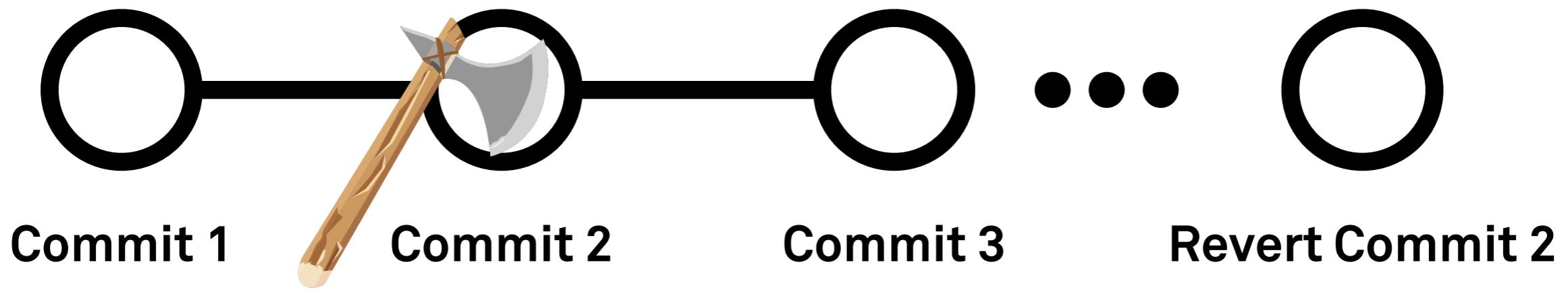
Showing 1 changed file with 1 addition and 3 deletions.

Unified Split

4 4 .gitignore

[View](#)

```
diff --git a/.gitignore b/.gitignore
@@ -9,9 +9,7 @@ __pycache__/
 9   9  # Distribution / packaging
10  10  .Python
11  11  env/
12  12  -**/build/*
13  13  -!**/build/.gitignore
14  14  -!**/build/Makefile
15  12  +build/
16  13
17  14  develop-eggs/
18  15  dist/
```



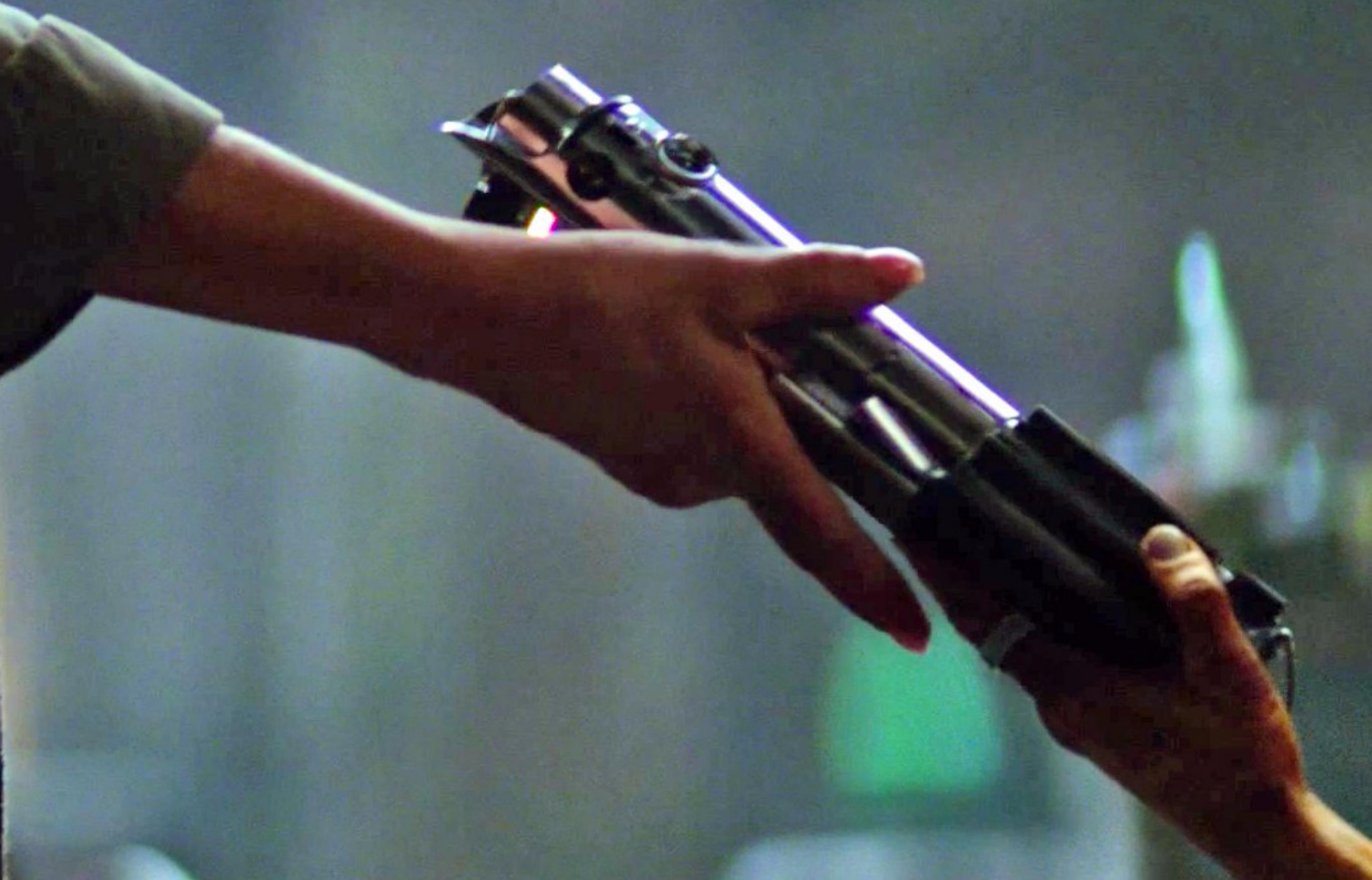
THIS IS GIT. IT TRACKS COLLABORATIVE WORK  
ON PROJECTS THROUGH A BEAUTIFUL  
DISTRIBUTED GRAPH THEORY TREE MODEL.

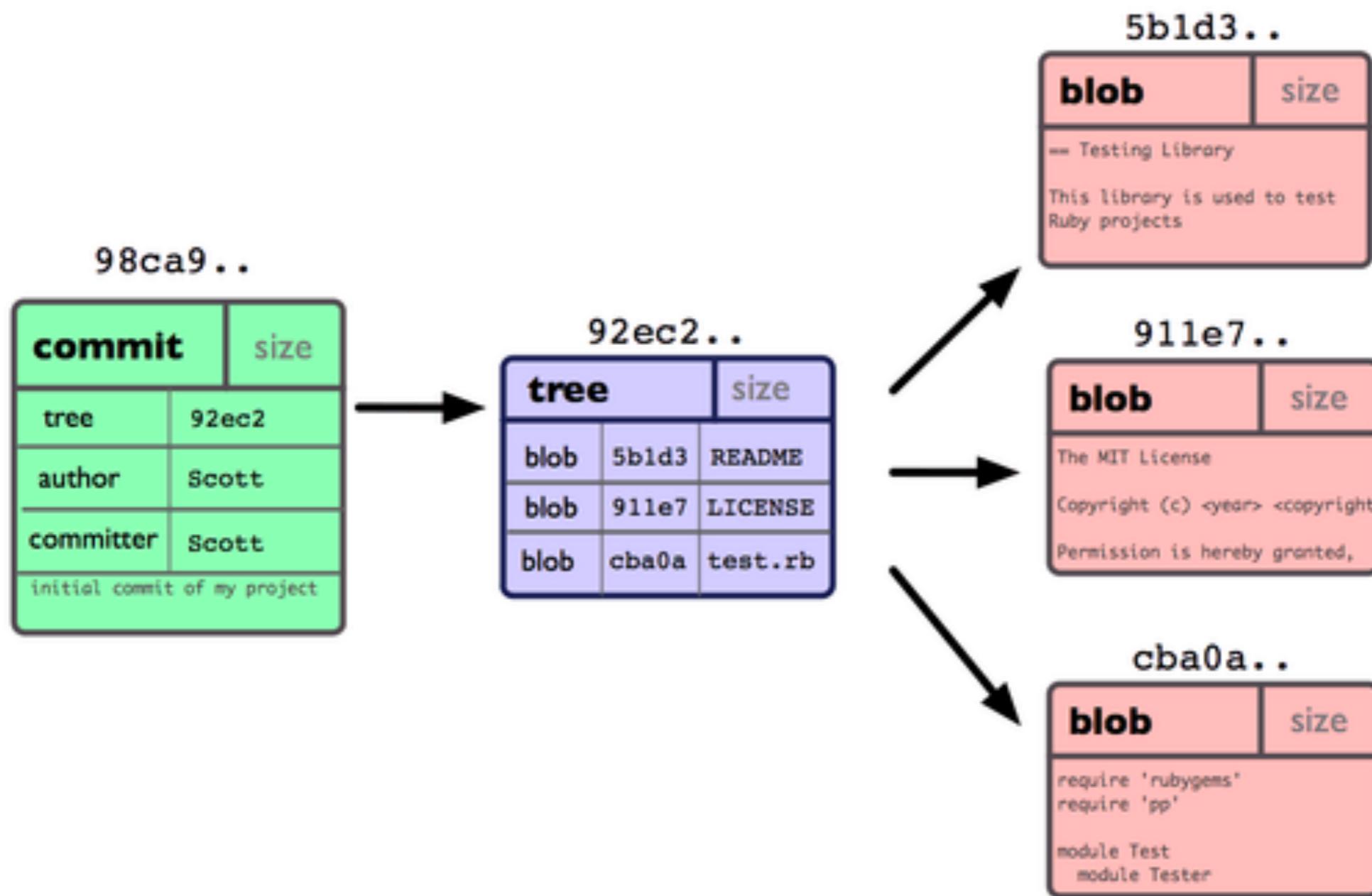
COOL. HOW DO WE USE IT?

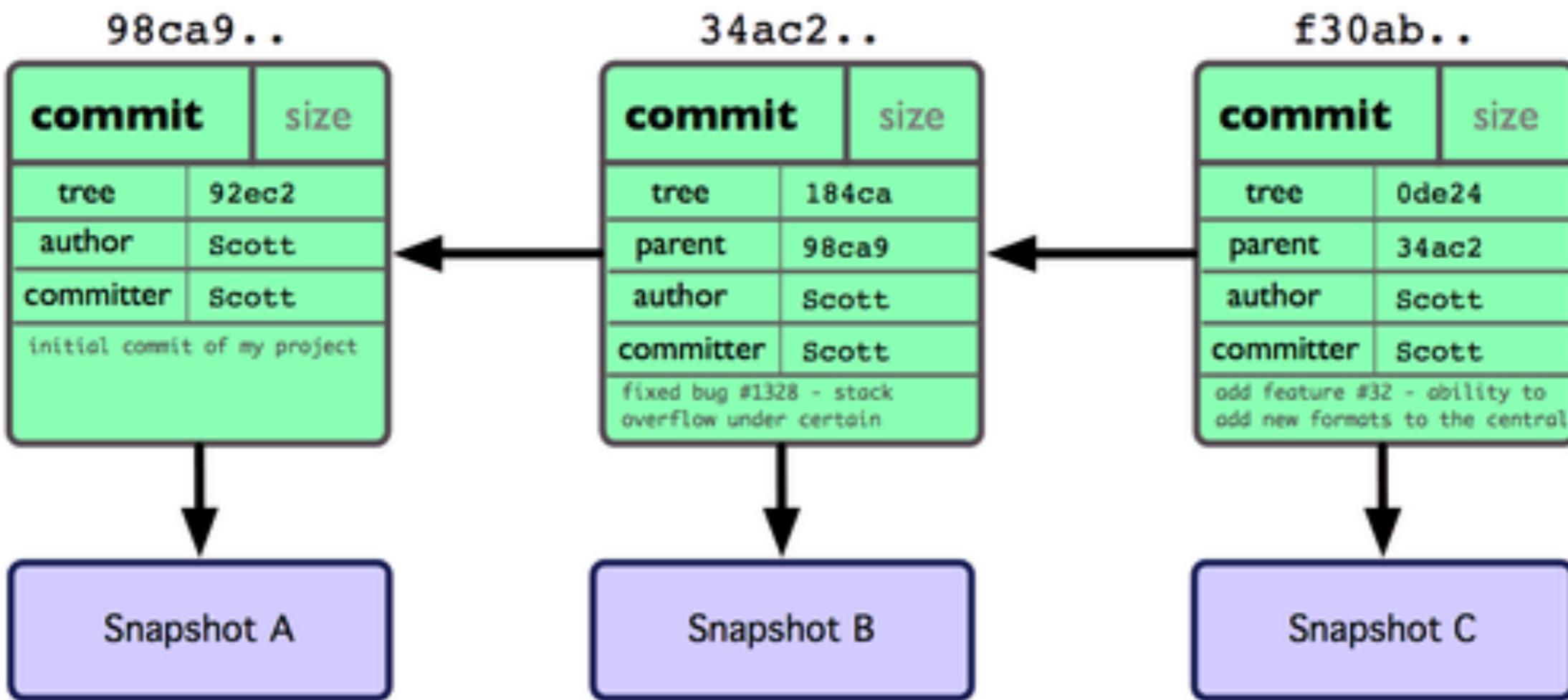
NO IDEA. JUST MEMORIZIZE THESE SHELL  
COMMANDS AND TYPE THEM TO SYNC UP.  
IF YOU GET ERRORS, SAVE YOUR WORK  
ELSEWHERE, DELETE THE PROJECT,  
AND DOWNLOAD A FRESH COPY.



The Force Awakens in You



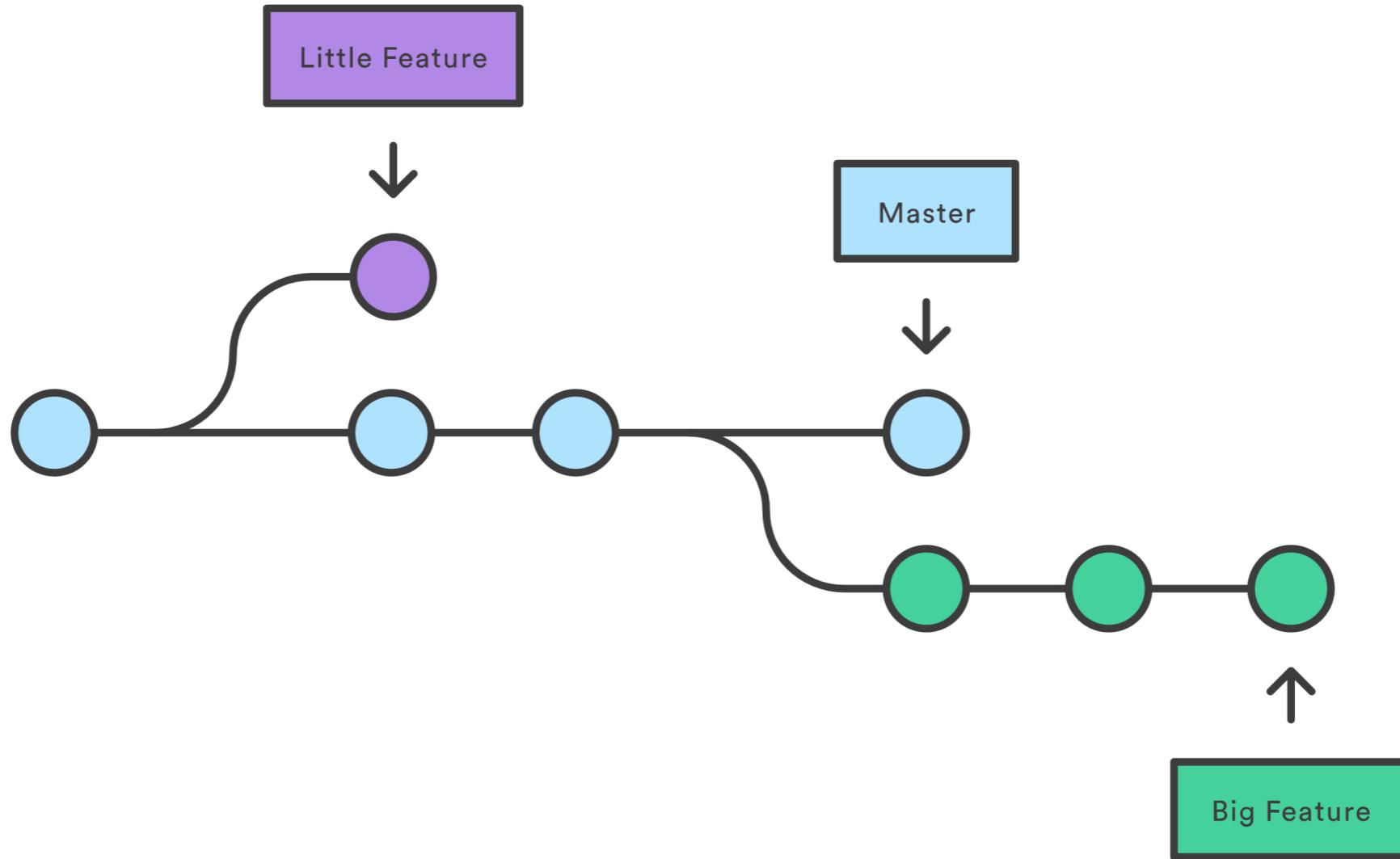




# Commit Messages

	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
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSOKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAANDS	2 HOURS AGO

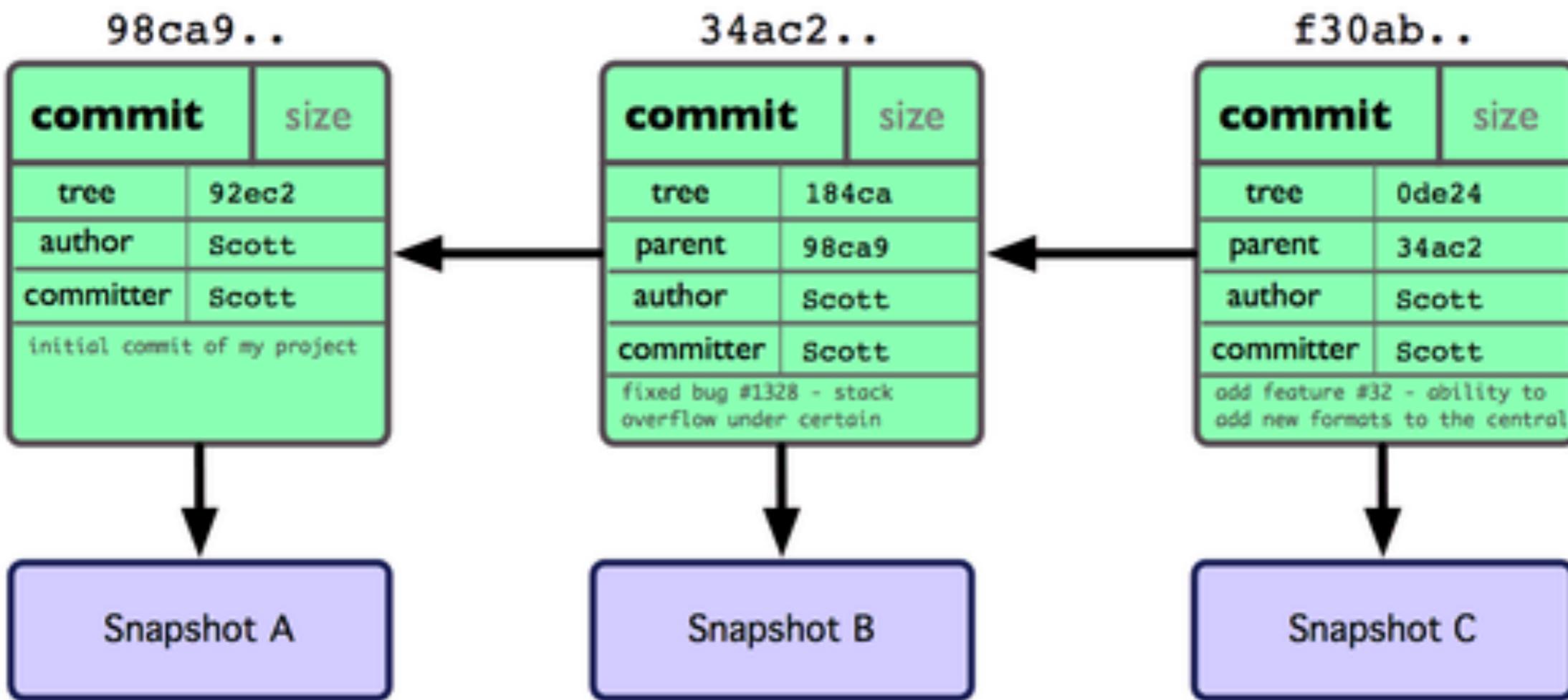
AS A PROJECT DRAGS ON, MY GIT COMMIT  
MESSAGES GET LESS AND LESS INFORMATIVE.

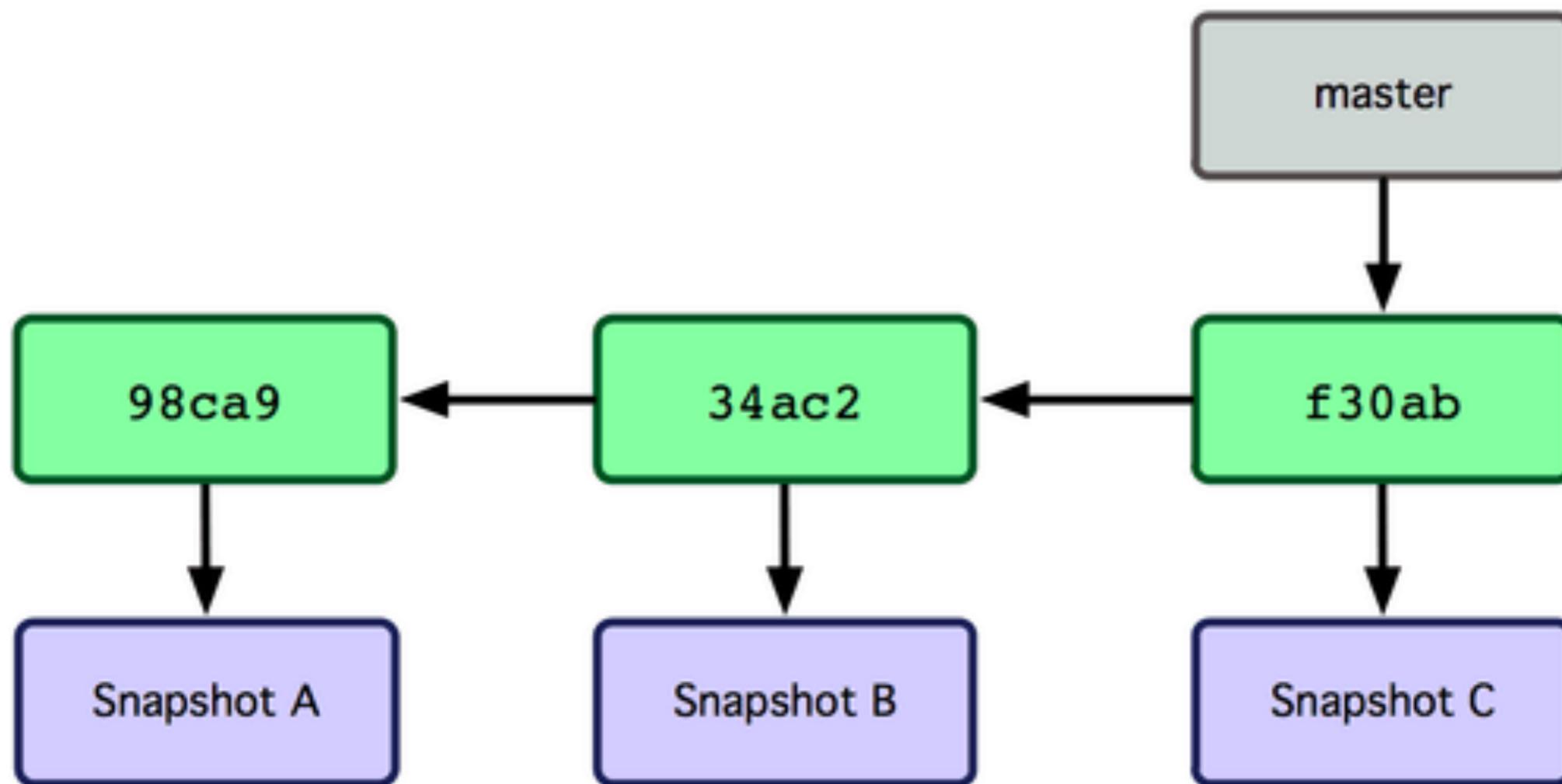


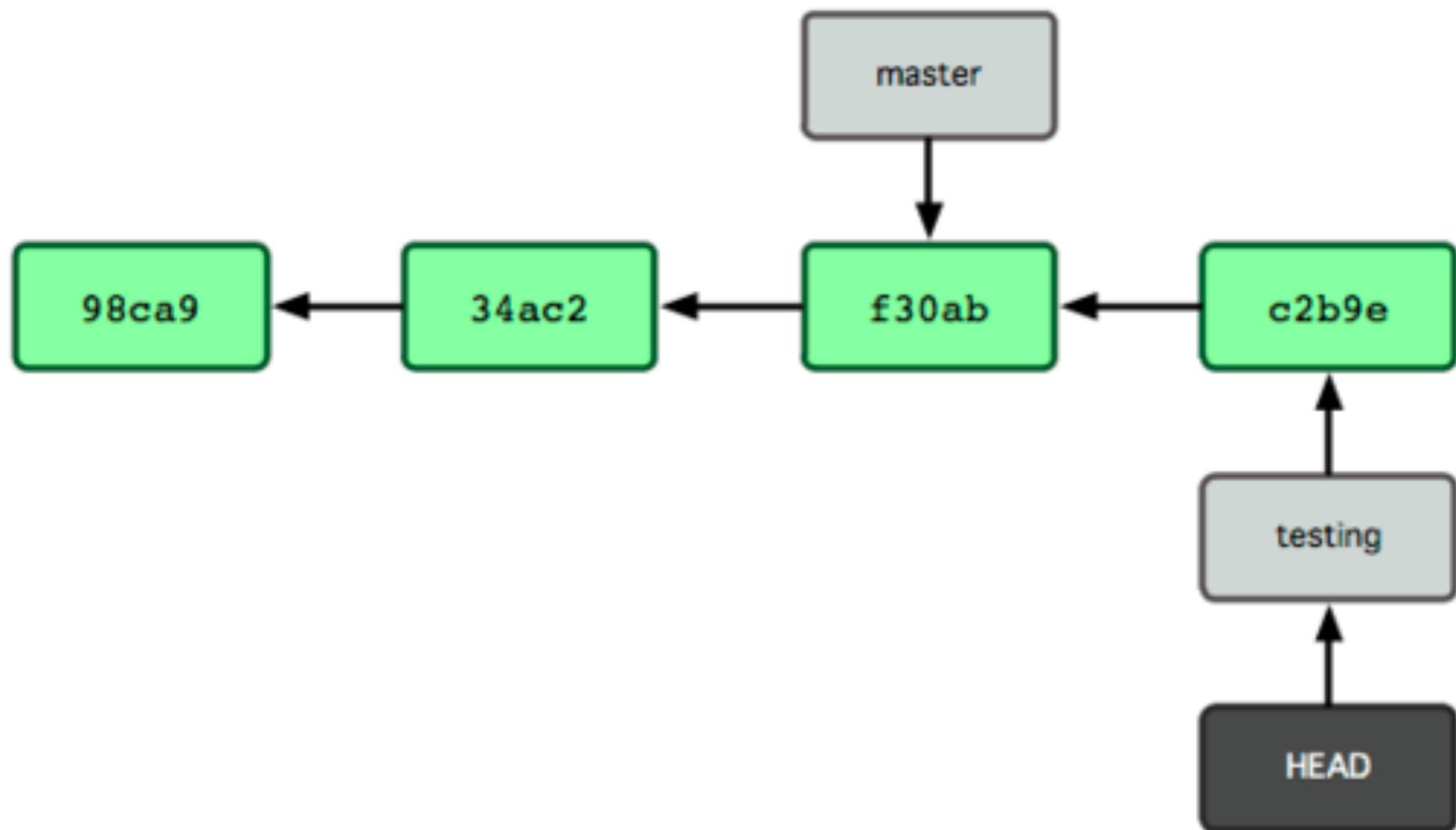
- Git branches are lightweight, encouraging branch based workflows
- Compare to other version control systems that use a heavyweight approach such as copying directories

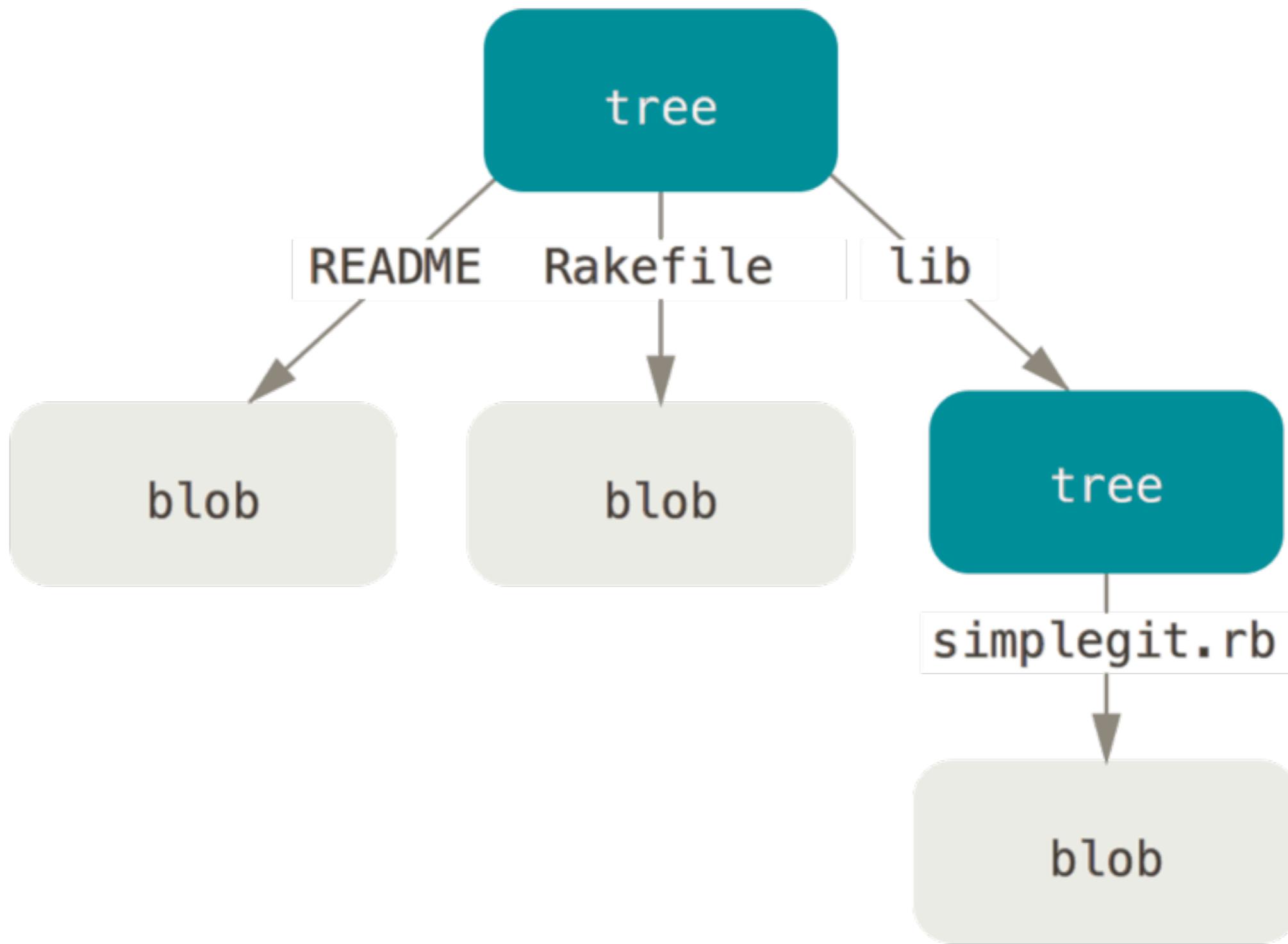
# Frictionless Context Switching

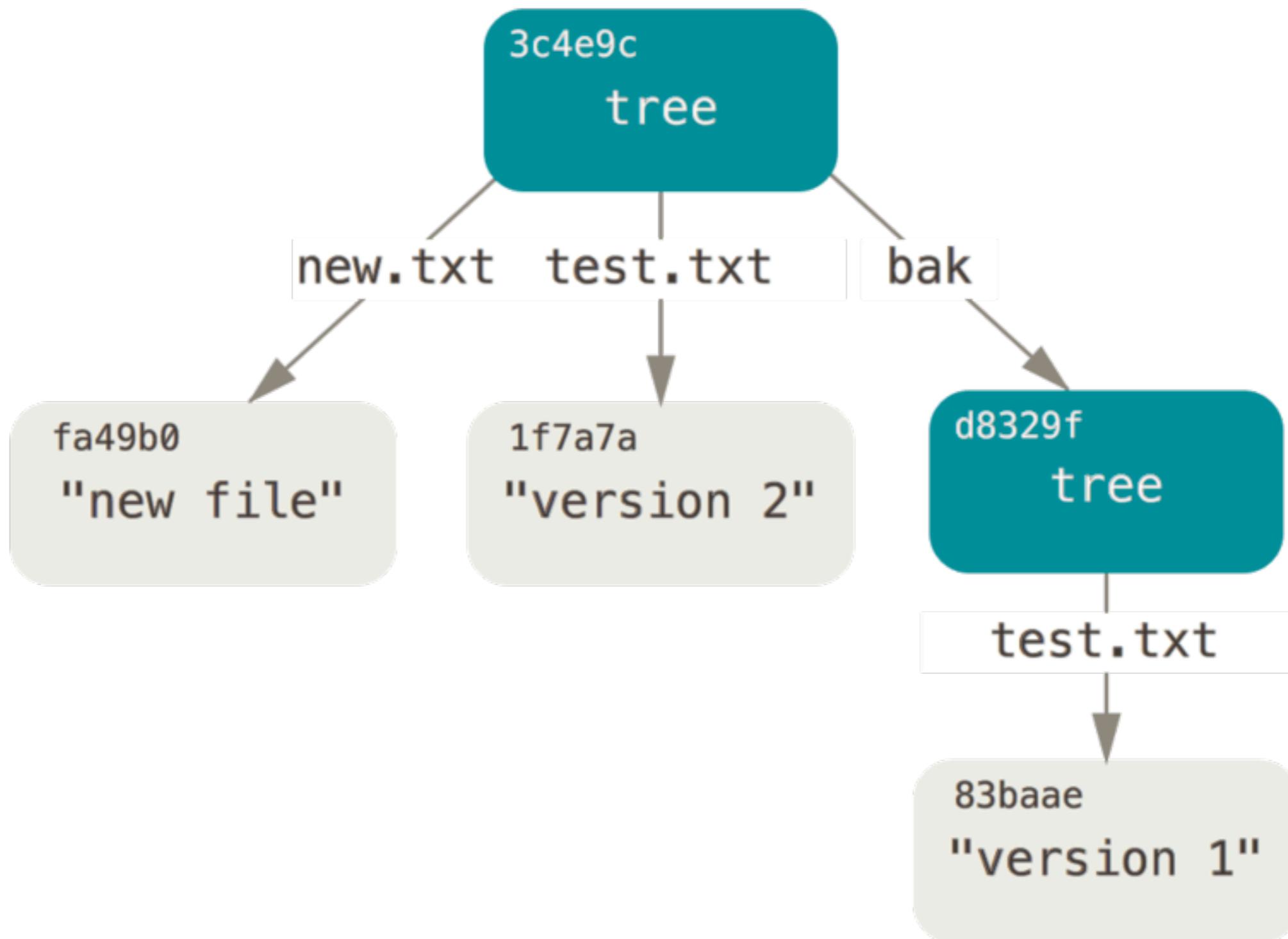
- Create a branch, try out an idea
- Switch back to another branch, fix a bug for your teammate
- Switch back to your branch, merge in the bug fix
- Disposable experimentation

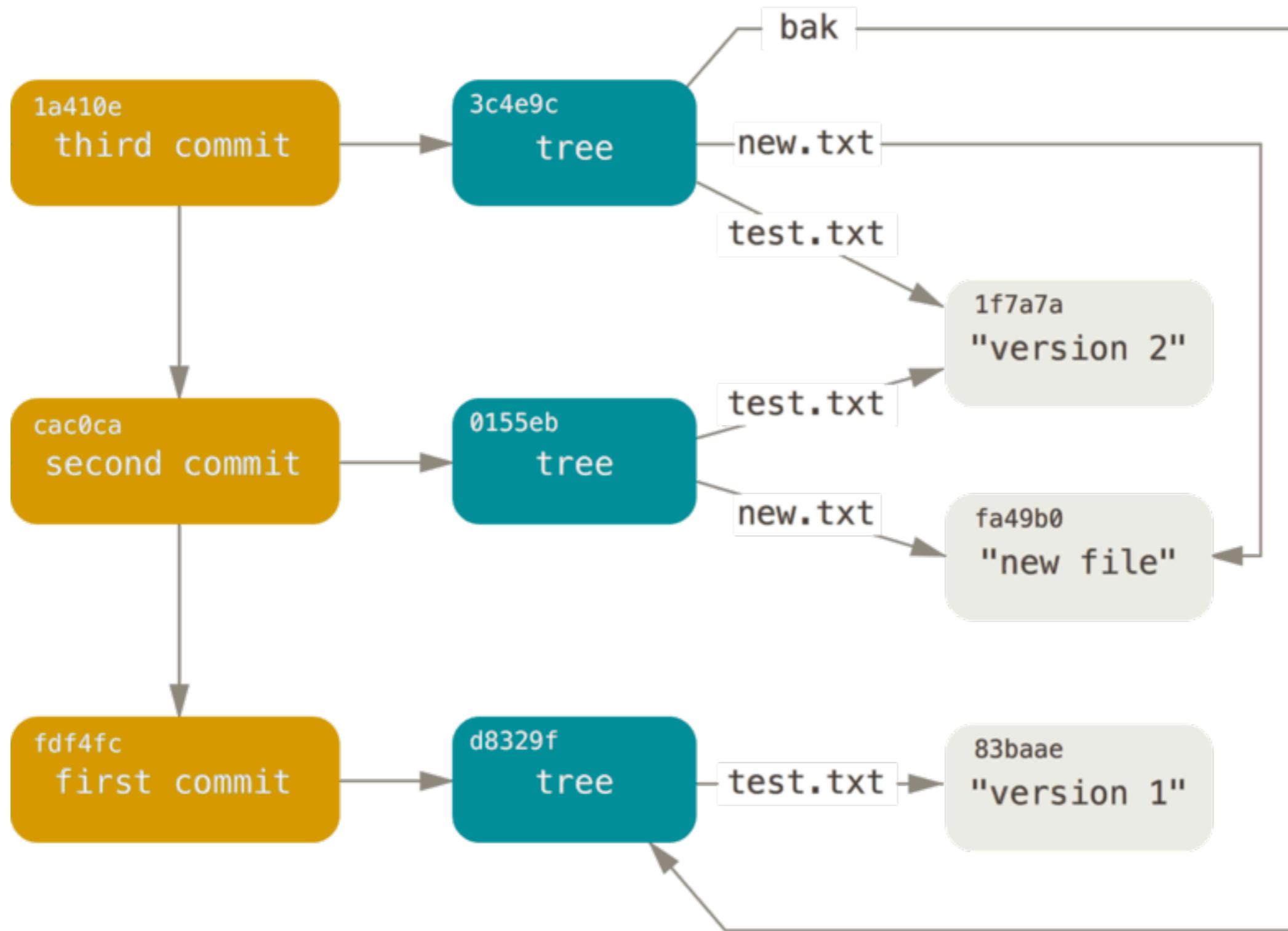


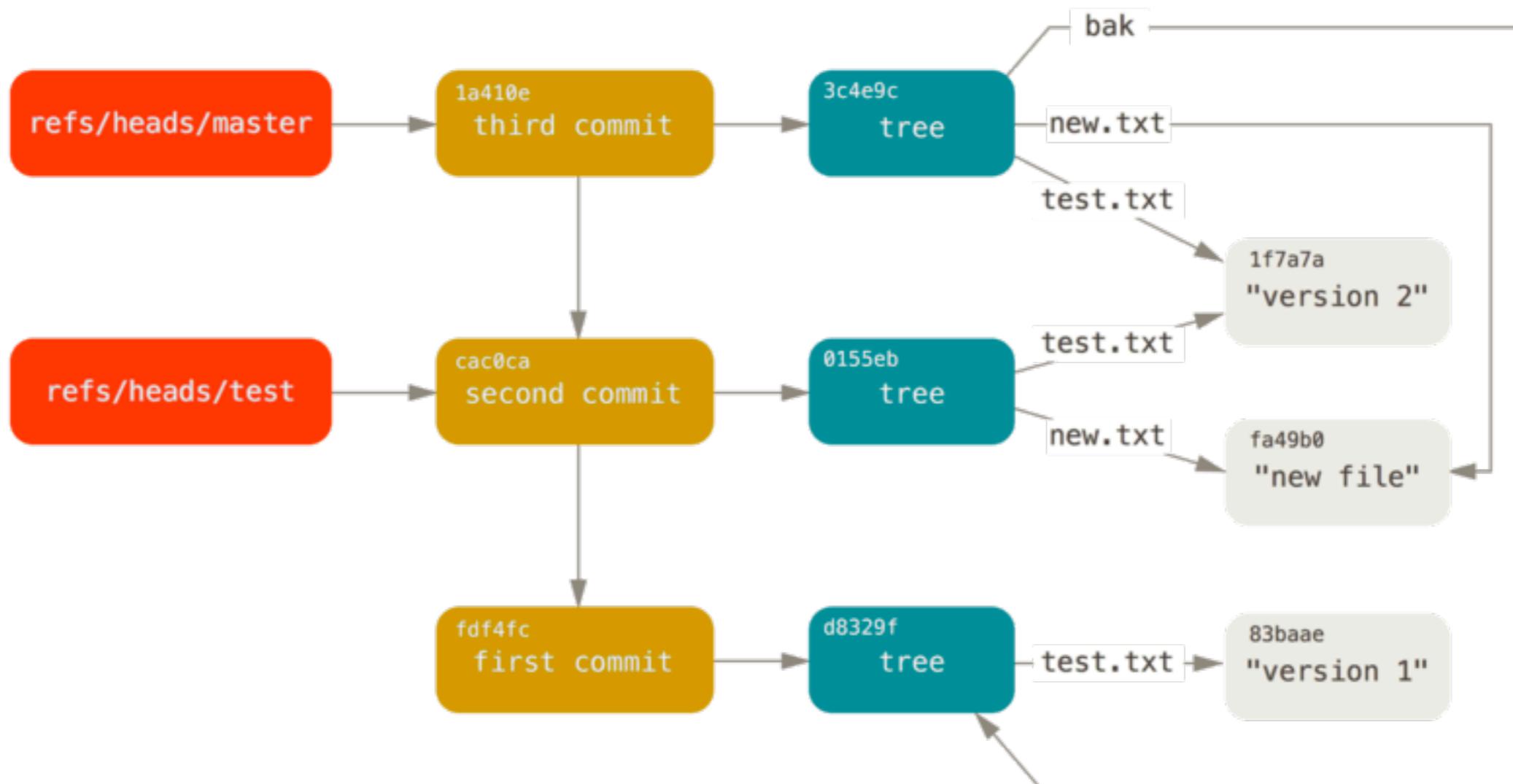












Make each program do one thing well. To do a new job, build afresh rather than complicate old programs by adding new "features".

*Doug McIlroy, 1978 Bell Labs*

# Unix Philosophy

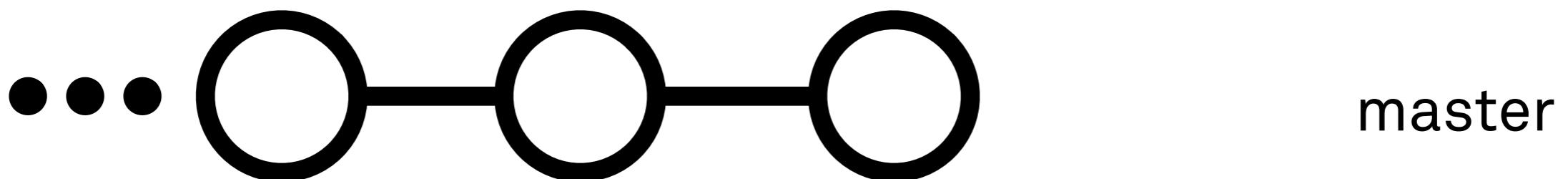
- Compose git commands to build a workflow
  - `git hash-object` (hash a file)
  - `git write-tree` (create a tree)
  - `git commit-tree` (create a commit)
  - `git update-ref` (create a branch)
  - Many, many more

# Feature Branch Workflow

- All feature development should occur in a dedicated branch (not master!)
- Master branch never contains broken code
- Encapsulating feature development makes it possible to leverage pull requests
  - Get reviews and help from your team

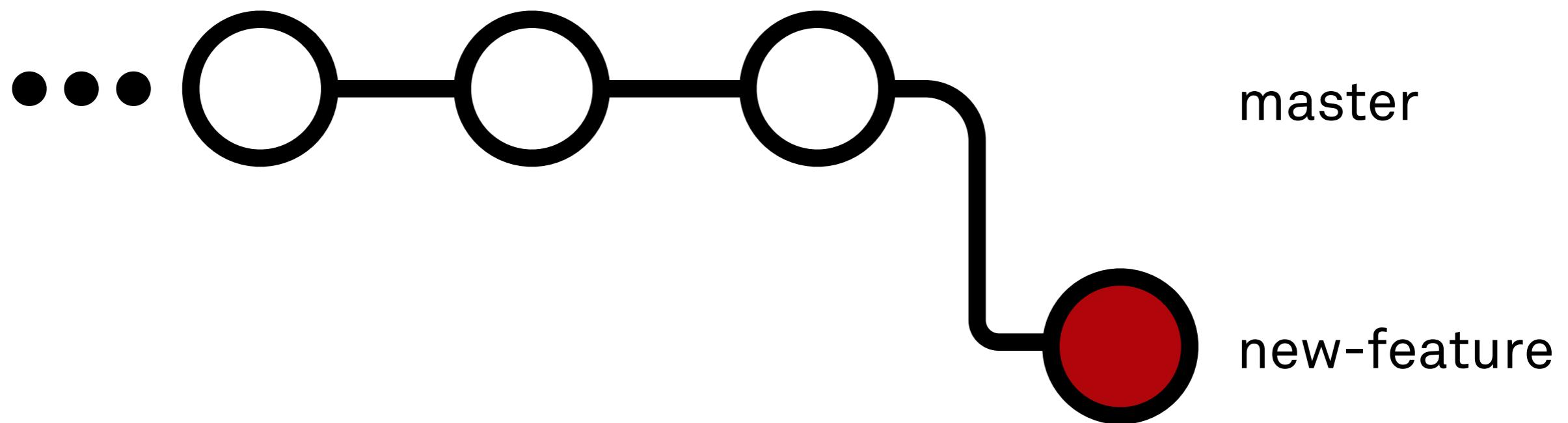
<https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow>

# Get the Latest Master



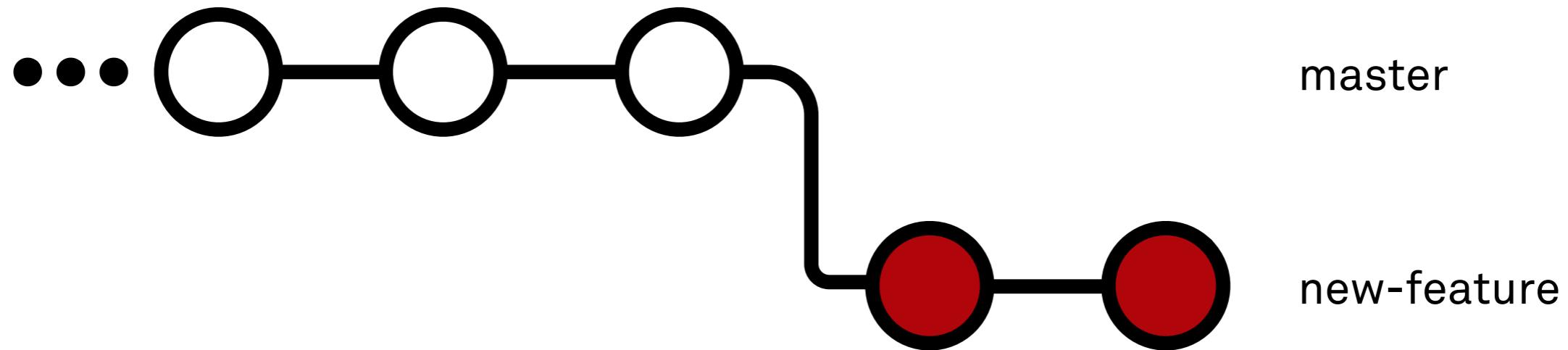
```
> cd repo  
> git checkout master  
> git fetch origin  
> git reset --hard origin/master
```

# New Feature = New Branch



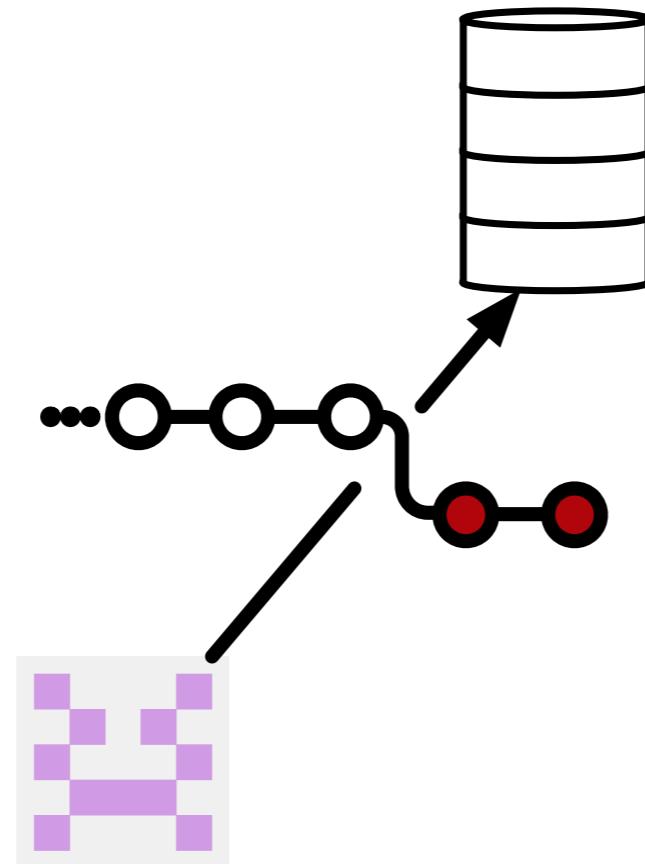
```
› git checkout -b new-feature master
```

# Update, add, commit, and push changes



```
› git status  
› git add <file>  
› git commit  
› git push -u origin new-feature
```

# Push feature branch to remote



```
› git push -u origin new-feature
```

# Pull Request

The screenshot shows a GitHub pull request page for the repository `numba / llvmlite`. The pull request is titled `Add support for loading dynamic libraries #47`. It is marked as `Merged`, with the merge commit message being `pitrou merged 5 commits into numba:master from leonardt:master` on Feb 19, 2015. The pull request has 11 comments, 5 commits, and 3 files changed, resulting in a code change of `+41 -2`.

The conversation history is as follows:

- leonardt** commented on Feb 8, 2015:  
*No description provided.*
- pitrou** commented on Feb 9, 2015:  
*Thanks for the patch. Can you explain why this is necessary? (rather than e.g. using ctypes)*
- leonardt** commented on Feb 9, 2015:  
*Some of our JITted code will make calls into run times such as OpenCL and OpenMP so we need the libraries to be linked into the engine*
- pitrou** commented on Feb 9, 2015:  
*Ok. The patch should have proper error management, though. See other exported functions to see how this is done. Also, adding a minimal test would be good.*

On the right side of the page, there are sections for **Reviewers**, **Assignees**, **Labels**, **Projects**, **Milestone**, and **Notifications**. The notifications section indicates that the user is receiving notifications because they authored the thread.

<https://github.com/numba/llvmlite/pull/47>

# Discuss the changes



pitrou commented on Feb 9, 2015

Member



Thanks for the patch. Can you explain why this is necessary? (rather than e.g. using ctypes)



leonardt commented on Feb 9, 2015

Contributor



Some of our JITted code will make calls into run times such as OpenCL and OpenMP so we need the libraries to be linked into the engine



pitrou commented on Feb 9, 2015

Member



Ok. The patch should have proper error management, though. See other exported functions to see how this is done. Also, adding a minimal test would be good.

# Resolve Feedback

pitrou reviewed on Feb 12, 2015 [View changes](#)

llvmlite/tests/test\_dylib.py [Hide outdated](#)

```
6  +from ctypes.util import find_library
7  +import unittest
8  +
9  +@unittest.skipUnless(platform.system() in ["Linux", "Darwin"], "Unsupport
```

pitrou on Feb 12, 2015 Member  
Actually, test\_bad\_library would probably work under Windows.

leonardt on Feb 12, 2015 Contributor  
not sure what the error message should look like, okay to just assert that an exception is raised?

pitrou on Feb 12, 2015 Member  
Yes, that's ok.

Reply...

pitrou reviewed on Feb 12, 2015 [View changes](#)

llvmlite/tests/test\_dylib.py [Hide outdated](#)

```
31  +        elif self.system == "Darwin":
32  +            libm = find_library("libm")
33  +            dylib.load_library_permanently(libm)
34  +            except Exception:
```

pitrou on Feb 12, 2015 Member  
No need to catch the exception.

Reply...

<https://github.com/numba/llvmlite/pull/47>

# Merge!

  pitrou merged commit **18d2060** into `numba:master` on Feb 19, 2015 [View details](#) [Revert](#)  
1 check passed

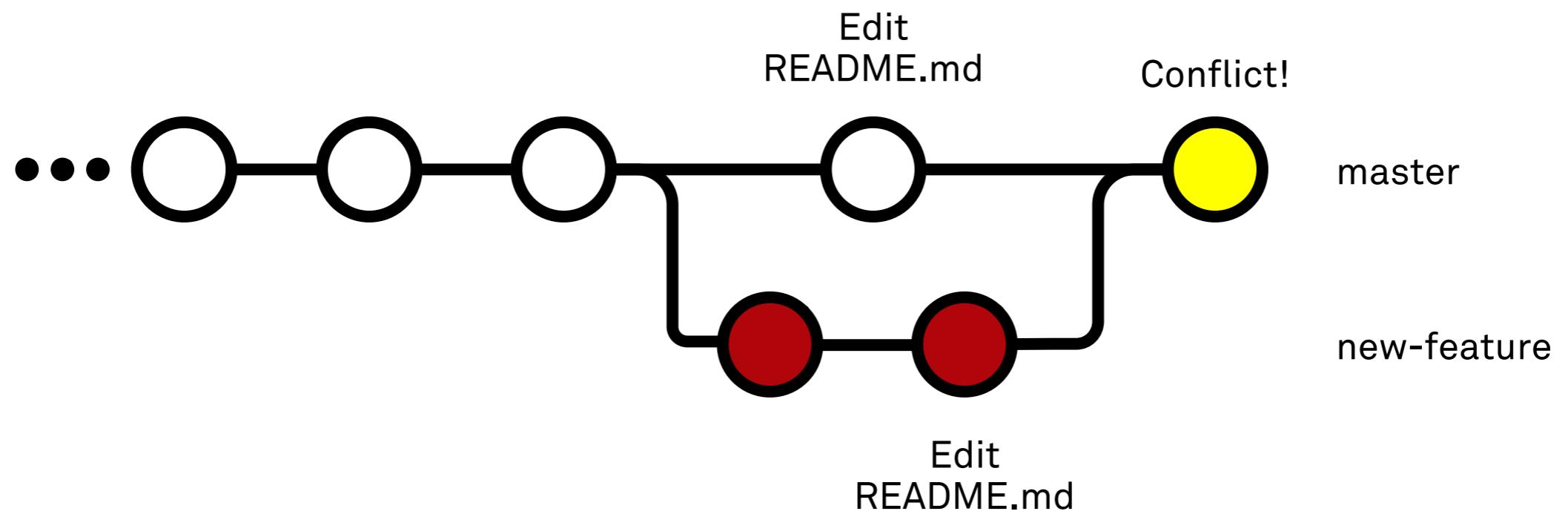
---

 pitrou commented on Feb 19, 2015 Member + 😊

Thank you!

<https://github.com/numba/llvmlite/pull/47>

# Merge Conflicts



# Merge Conflicts

```
> git status  
# On branch master  
# You have unmerged paths.  
#   (fix conflicts and run "git commit")  
#  
# Unmerged paths:  
#   (use "git add ..." to mark resolution)  
#  
# both modified: README.md
```

# Merge Conflicts

```
› cat README.md
```

```
This is a README
```

```
<<<<< HEAD
```

```
Added another line in master.
```

```
=====
```

```
Added another line in new-feature.
```

```
>>>>> new-feature
```

# Merge Conflicts

```
› cat README.md
```

```
This is a README
```

```
Added another line in master.
```

```
Added another line in new-feature.
```

```
› git add README.md
```

```
› git commit
```

# Continuous Integration

- Strive to “continuously” (multiple times a day) merge new code into the master branch
- The smaller the changes, the easier to integrate (less merge conflicts)
- Leverage testing and automation to integrate quickly and with confidence

# Decentralized

- Everyone has a **full copy** of the repository
- **Massive** impact in the open source world
  - Erase barriers to contribution
  - No need to manage **access control** between users
  - Anyone can participate!

# Sites to Know

- Git Repository Hosting



- Continuous Integration



Travis CI



# In case of fire



- 1. git commit
-  2. git push
-  3. leave building