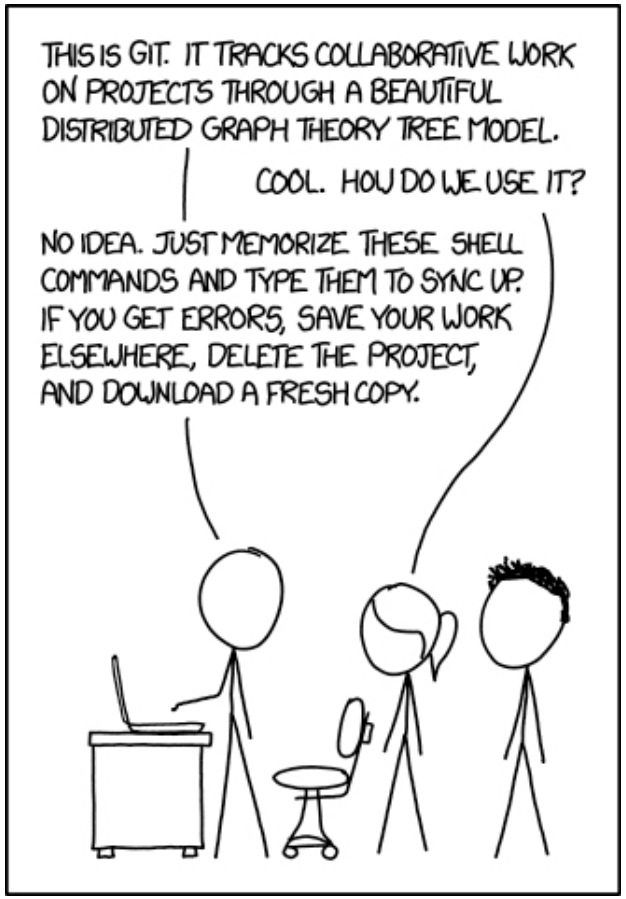
GIT

http://github.com/abpwrs/git-talk

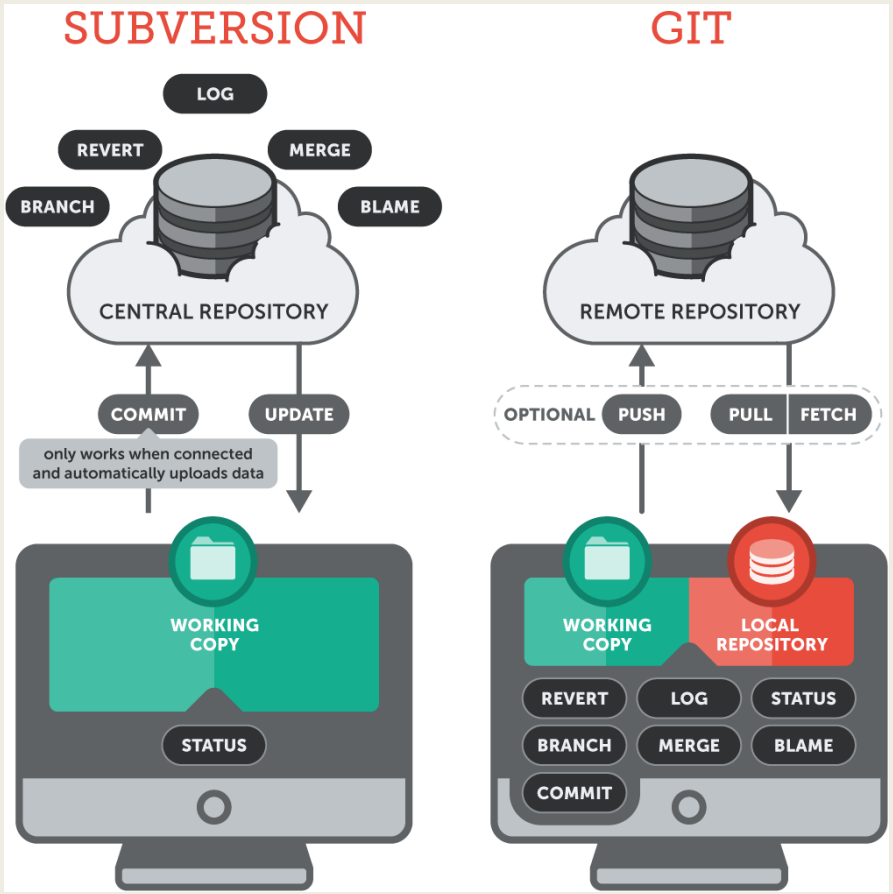
Acknowledgements

- LOL, This isn't all my intellectual property


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 MEMORIZE 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.

Overview

- Why Git?
- Git Basics
- The .gitignore
- Branches
- Conflict


SUBVERSION
LOG
REVERT
MERGE
BRANCH
BLAME
CENTRAL REPOSITORY
COMMIT
UPDATE
only works when connected
and automatically uploads data
WORKING
COPY
STATUS
GIT
REMOTE REPOSITORY
OPTIONAL
PUSH
PULL
FETCH
WORKING
COPY
LOCAL
REPOSITORY
REVERT
LOG
STATUS
BRANCH
MERGE
BLAME
COMMIT

WHY GIT?

Data Storage

- Git
  - *40 Character SHA-1 hashes*
  - *Every copy has the entire repo history*
- SVN
  - *Version # increments on every change*
  - *Need backups in case the server fails*

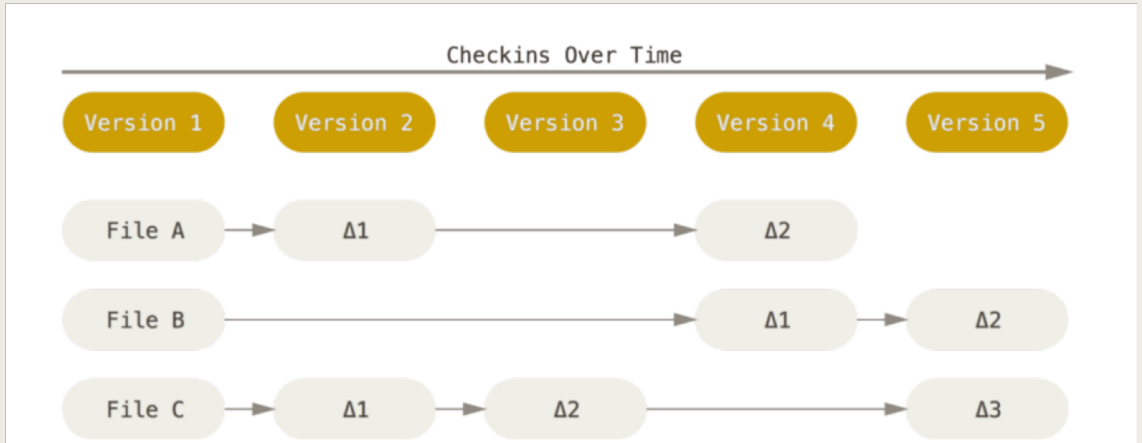


Figure 4. Storing data as changes to a base version of each file.

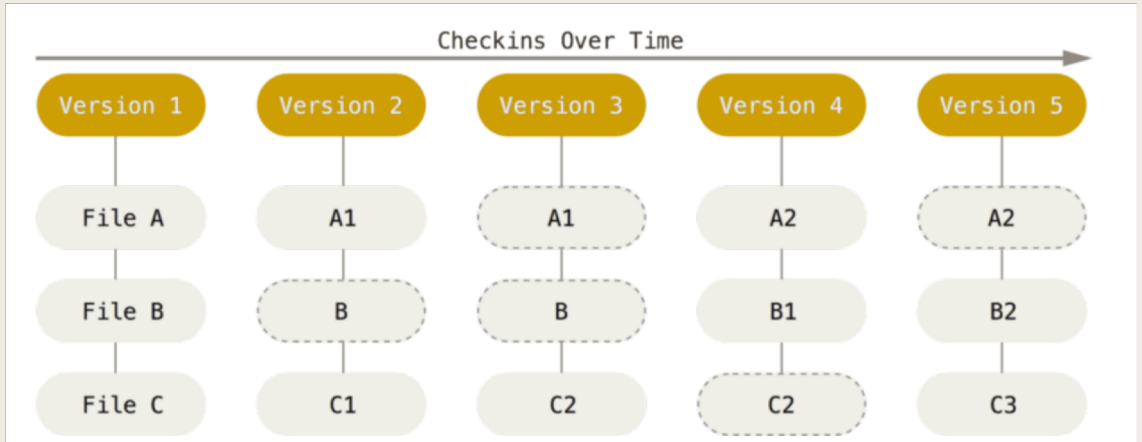


Figure 5. Storing data as snapshots of the project over time.

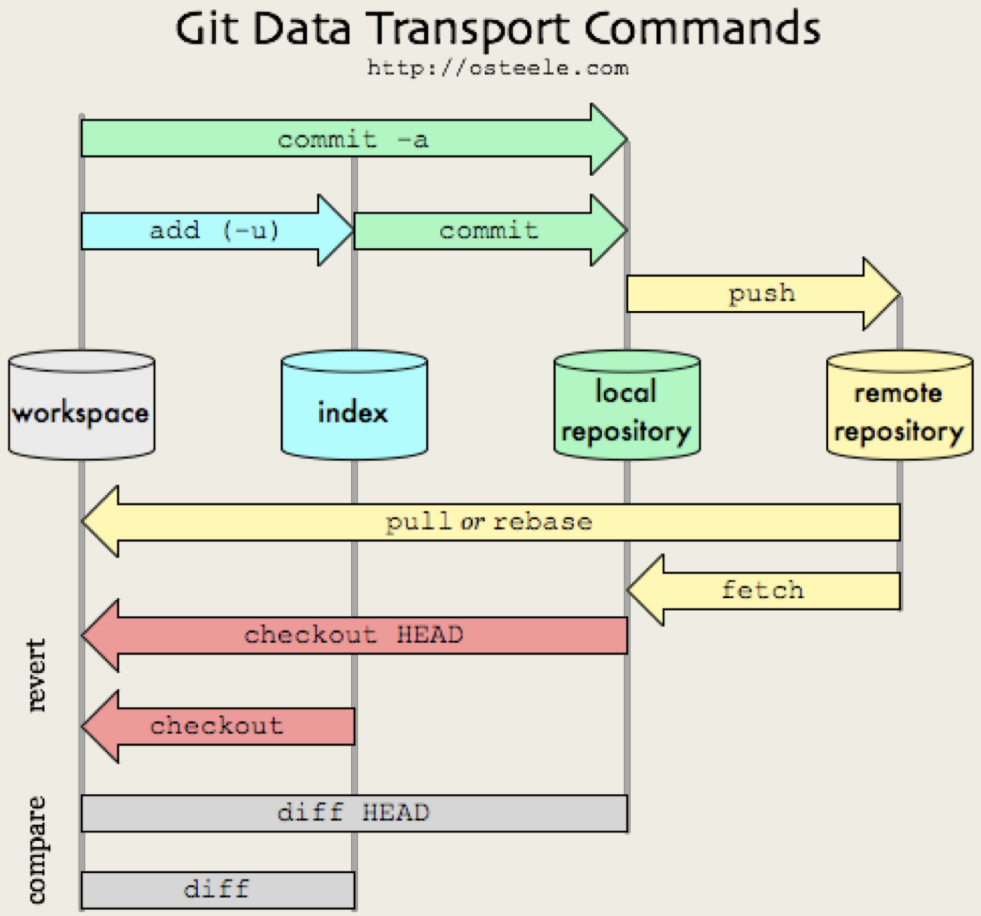
Centralized vs Distributed

Centralized

- Server w/ database
- Client w/ working copy
- Serialized Workflow
- Must be on a network to communicate w/ server

Distributed

- Central server only by convention
- Backups aren’t necessary, as everyone has a complete history
- Distributed Workflow
- VCS features online or offline


Git Data Transport Commands
http://osteele.com
commit -a
add (-u)
commit
push
workspace
index
local repository
remote repository
pull or rebase
fetch
checkout HEAD
checkout
diff HEAD
diff
revert
compare

GIT BASICS

The Bare Necessities

- git add
  - *Stages files to be a part of the next commit*
- git commit
  - *Commits files from staging area*
  - *(adds a commit “node” to the git history “linked list”)*

Remote Commands

- git fetch
  - *Fetches a branch or all branches of a remote repository*
- git pull
  - *Does git fetch and then merges that branch*
- git push
  - *Pushes a branch and its commits to a remote repository*
- git remote add
  - *Adds a remote repository*

The .gitignore

What is a .gitignore?

- A .gitignore file is a list of files, file extensions, and paths that should not be tracked by git

```
1 temp +
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST

# PyInstaller
#  Usually these files are written by a python script from a template
#  before PyInstaller builds the exe, so as to inject date/other infos into it.
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.nox/
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
.hypothesis/
.pytest_cache/

# Translations
NORMAL  temp  +
```

What do I put in a gitignore?

- Most communities and languages have guidelines -- https://www.gitignore.io/
- All of your personal environment configuration should be ignored:
  - *.idea*
  - *.vscode*
- Certain confidential components of applications you will build
  - *Client secrets*
  - *Databases*
  - *API keys*

BRANCHES

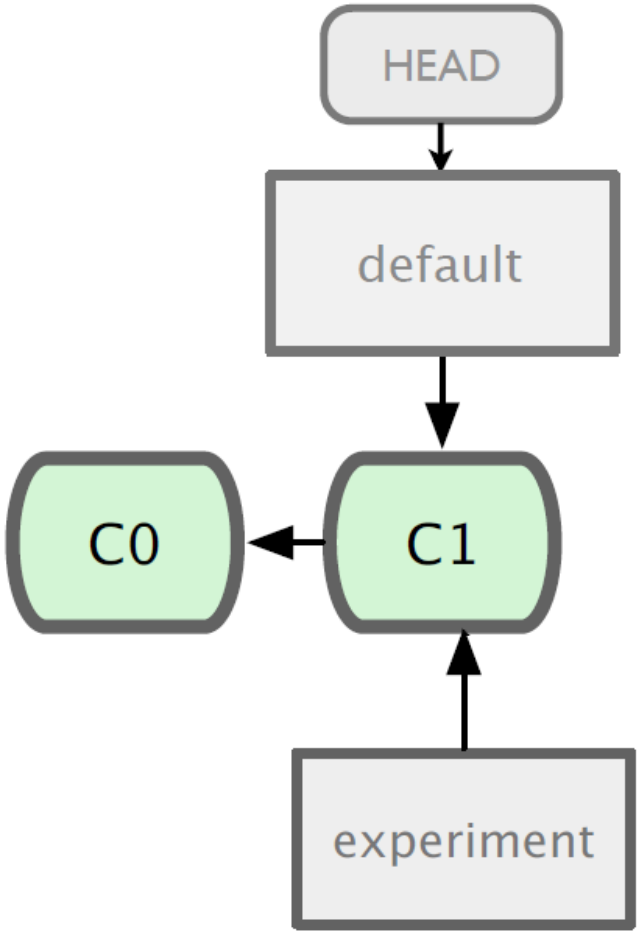
What is a branch?

- A branch is a pointer to a commit
- Your entire git history is actually a LinkedList of commits
- Changing branches is simply changing your HEAD pointer
- HEAD is a special pointer that points to the latest commit of the currently checked out branch

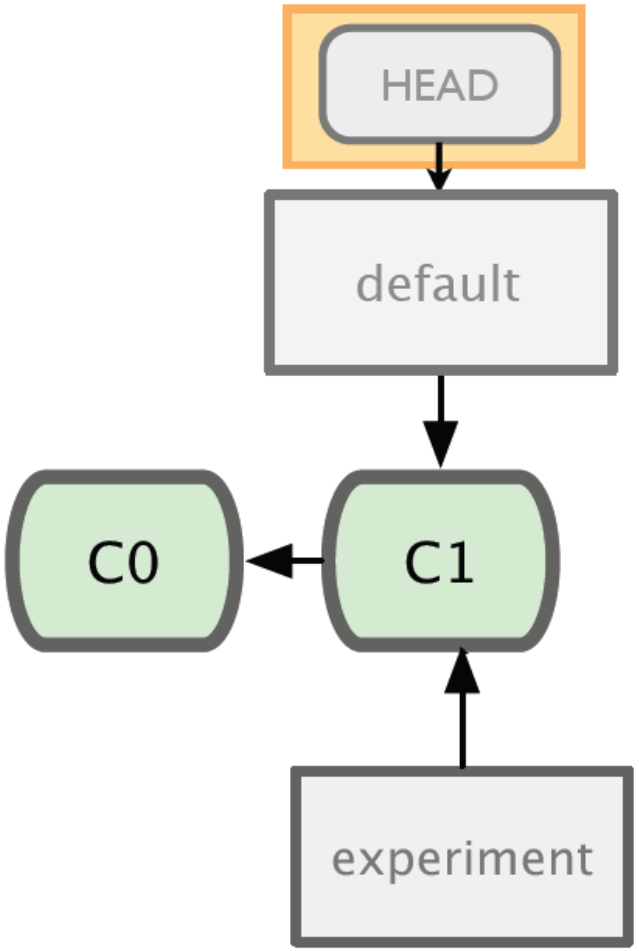
Git Branch Commands

- git branch
  - *No args – lists branches*
  - *<name> -- creates a new branch named <name>*
- git checkout <branch>
  - *Switches which branch you are currently working on*
- git merge <branch>
  - *Merges the contents of <branch> into the branch you are currently on*

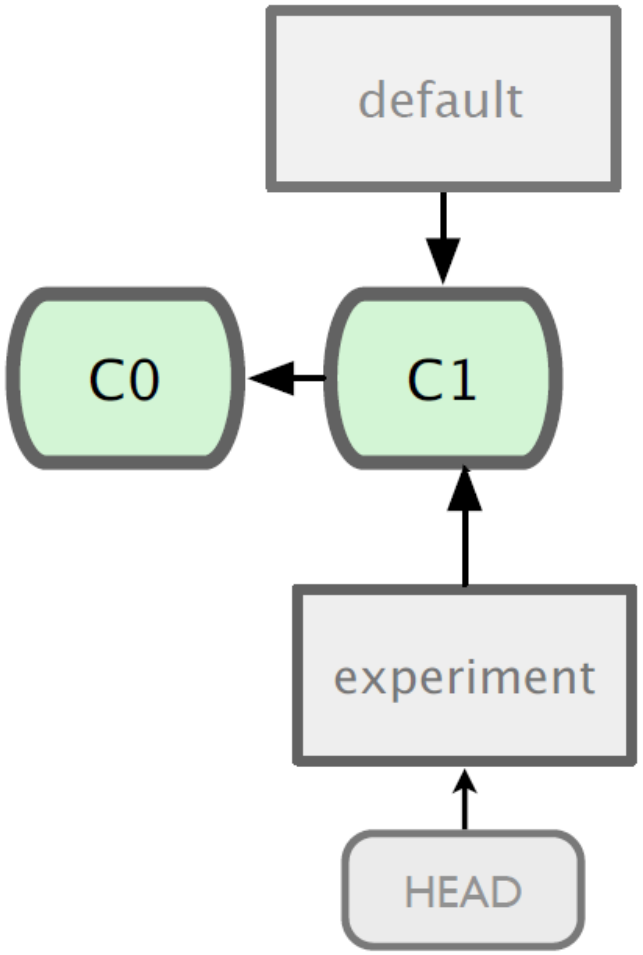
BRANCH EXAMPLE



git branch experiment

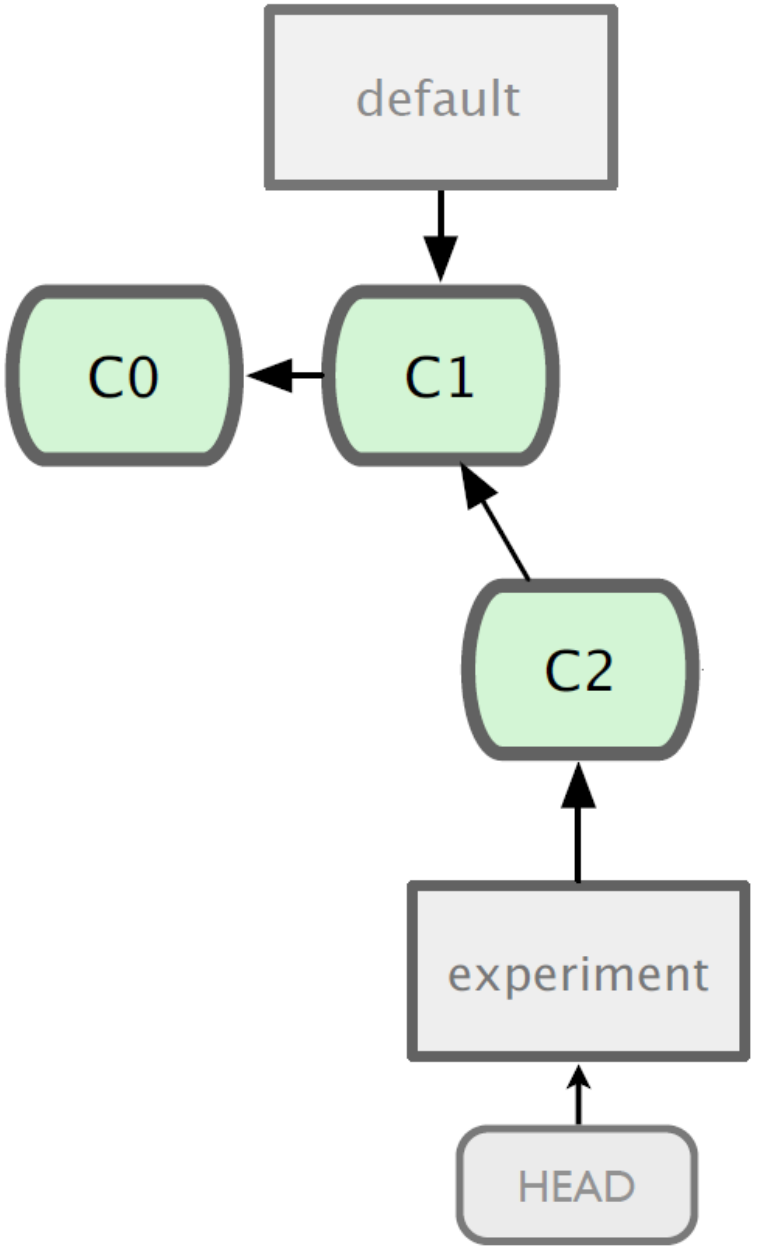


$$WorkingDirectory = \sum_{i=0}^{HEAD} C_i$$



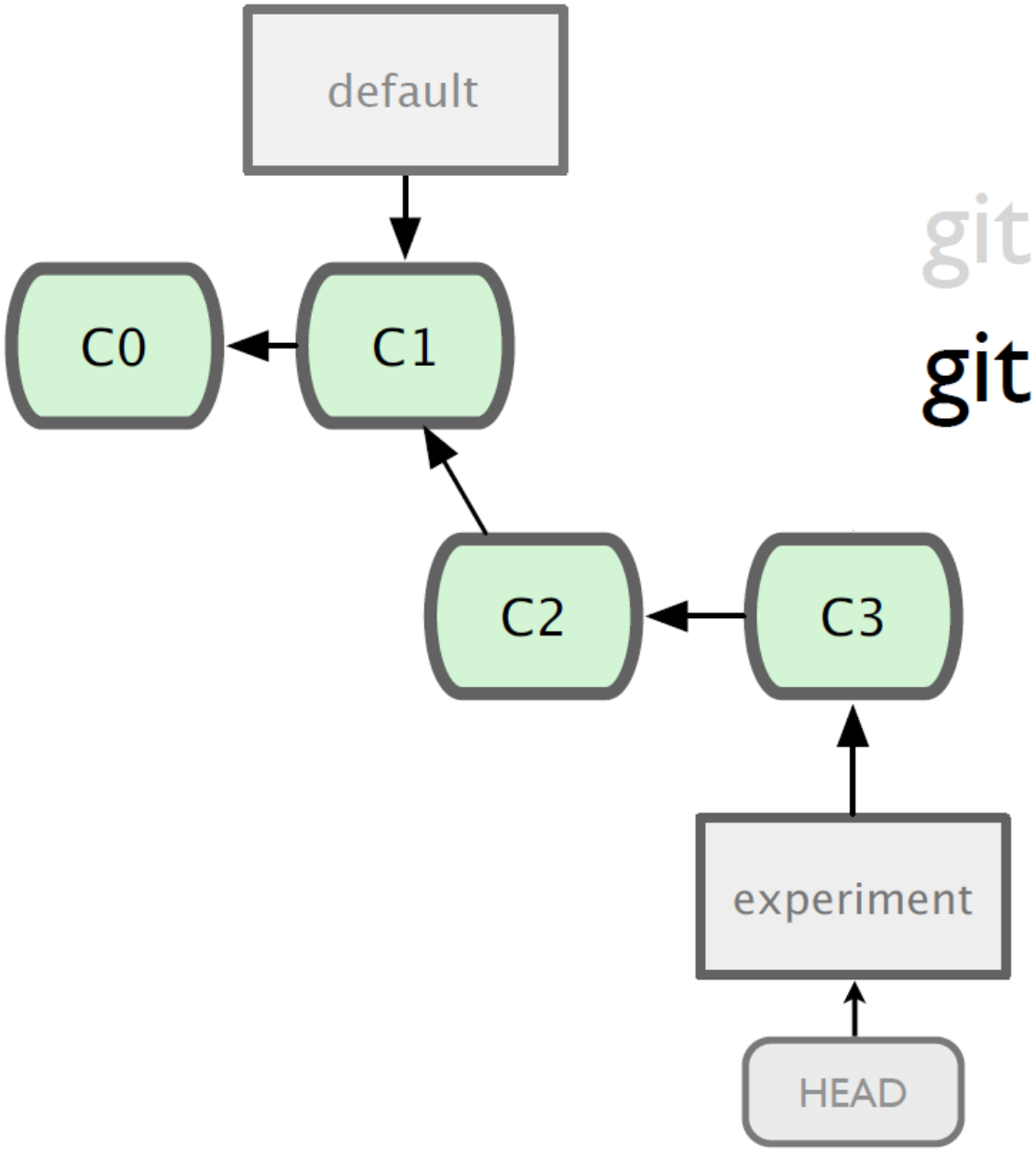
$$WorkingDirectory = \sum_{i=0}^{Experiment} C_i$$

git checkout experiment



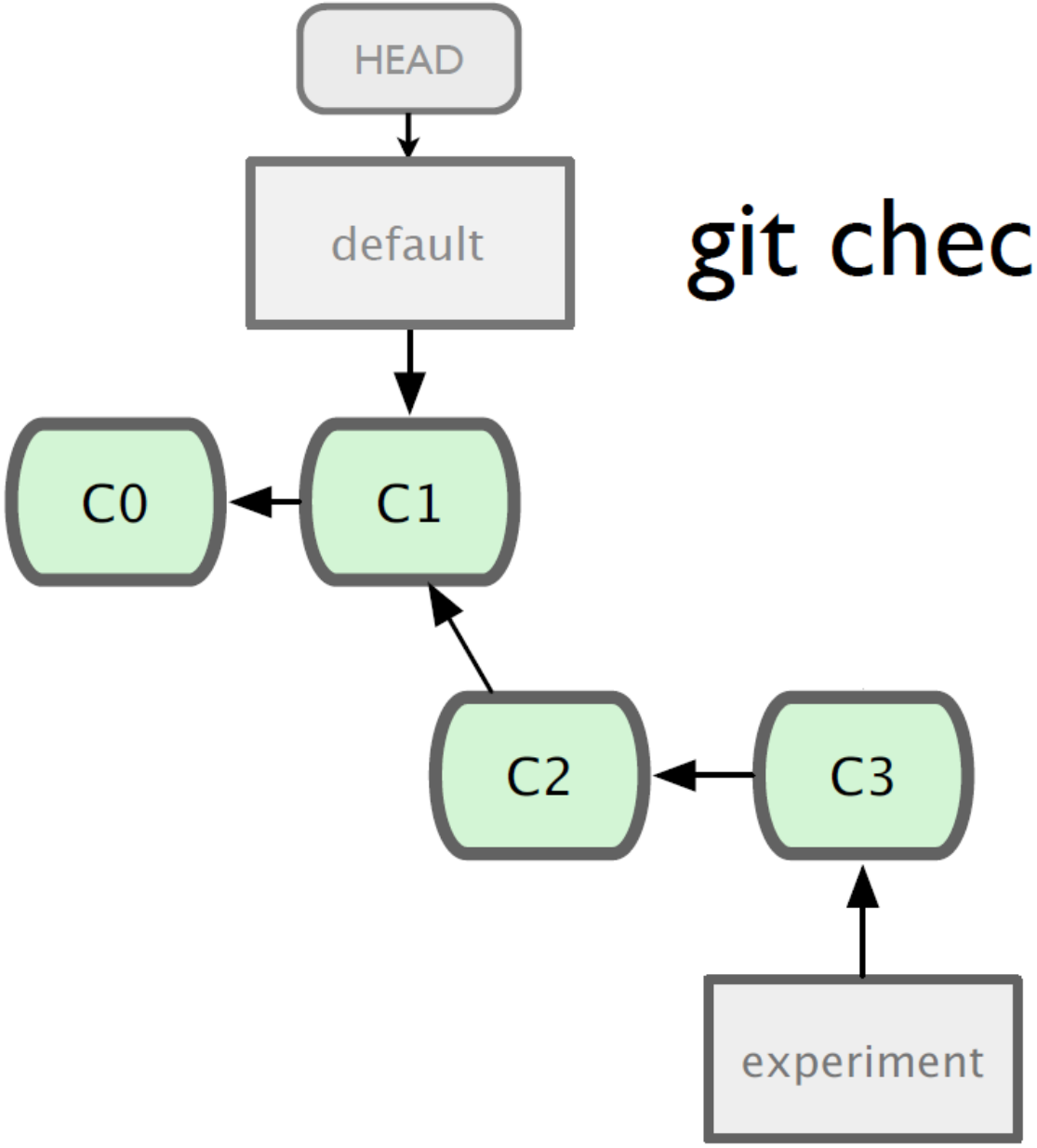
$$WorkingDirectory = \sum_{i=0}^{Experiment} C_i$$

git commit



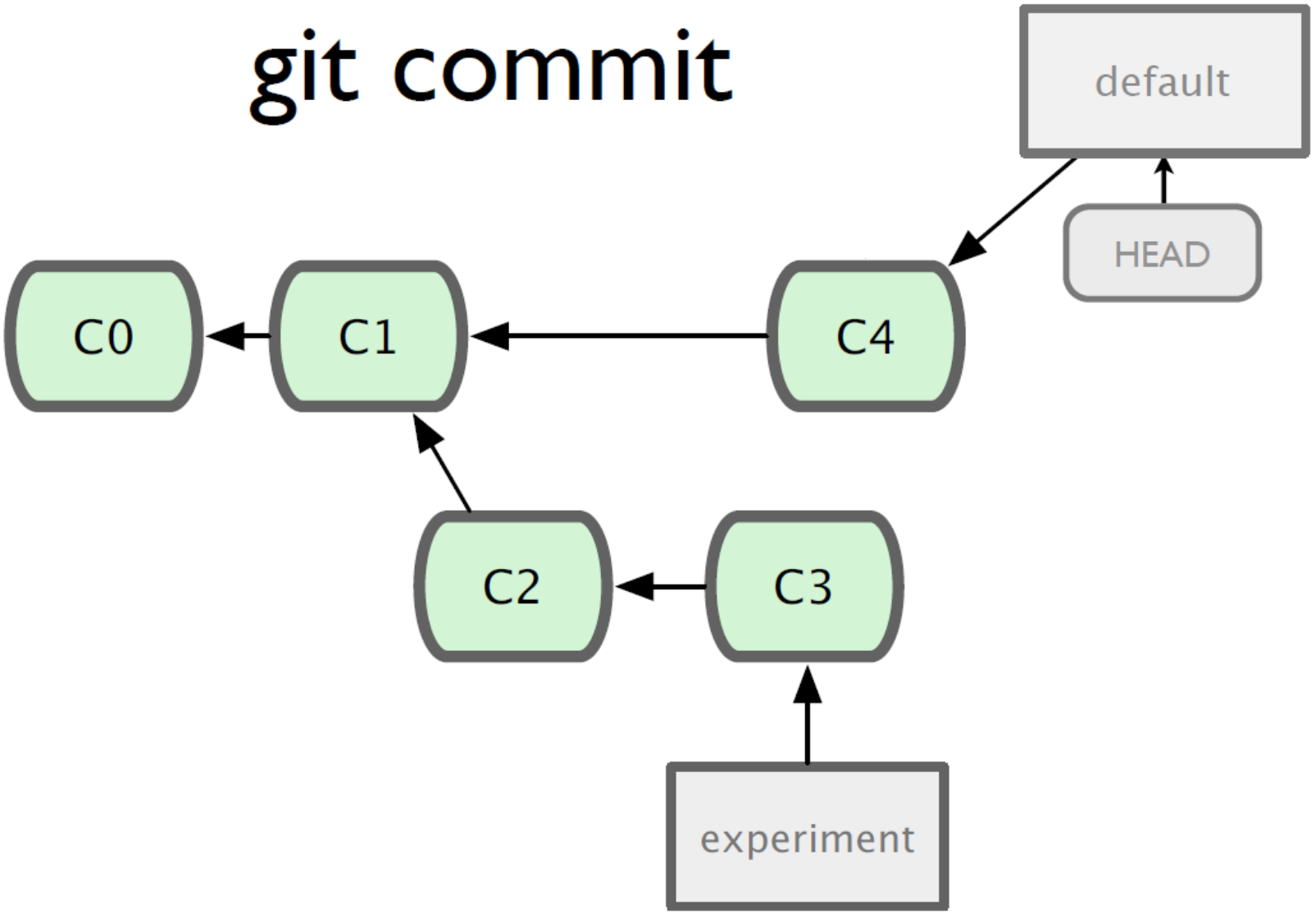
git commit

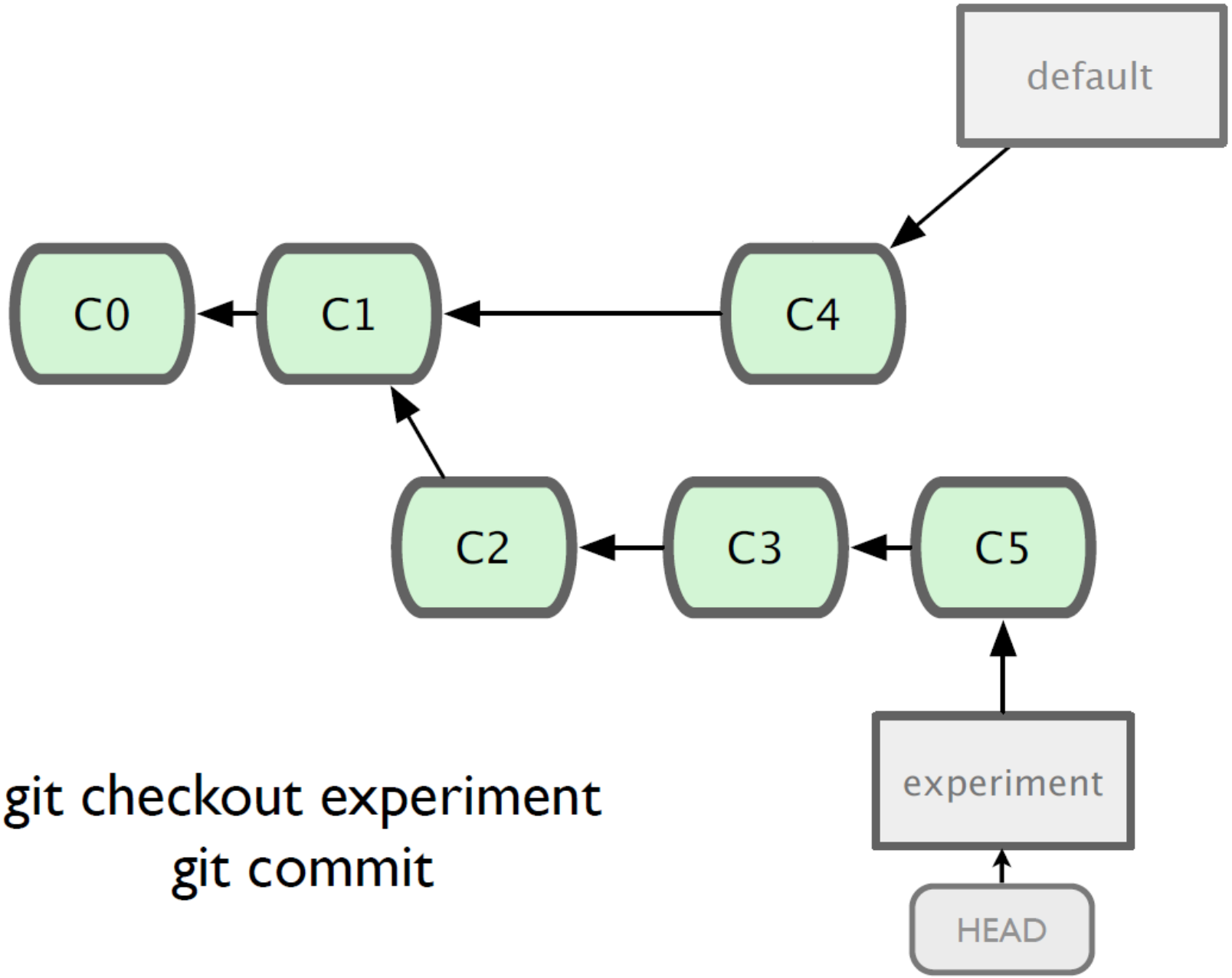
git commit



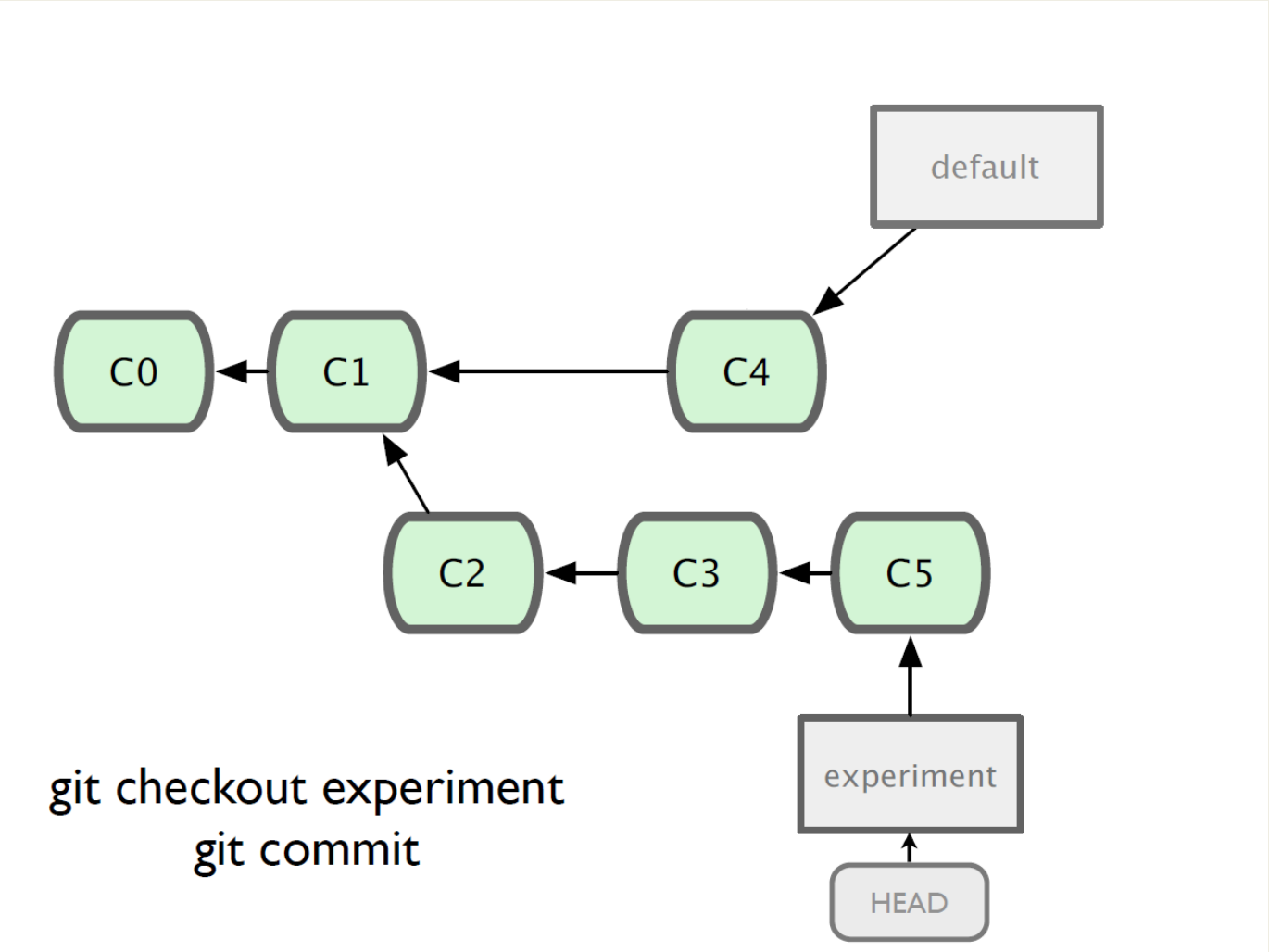
git checkout default

git commit





git checkout experiment
git commit



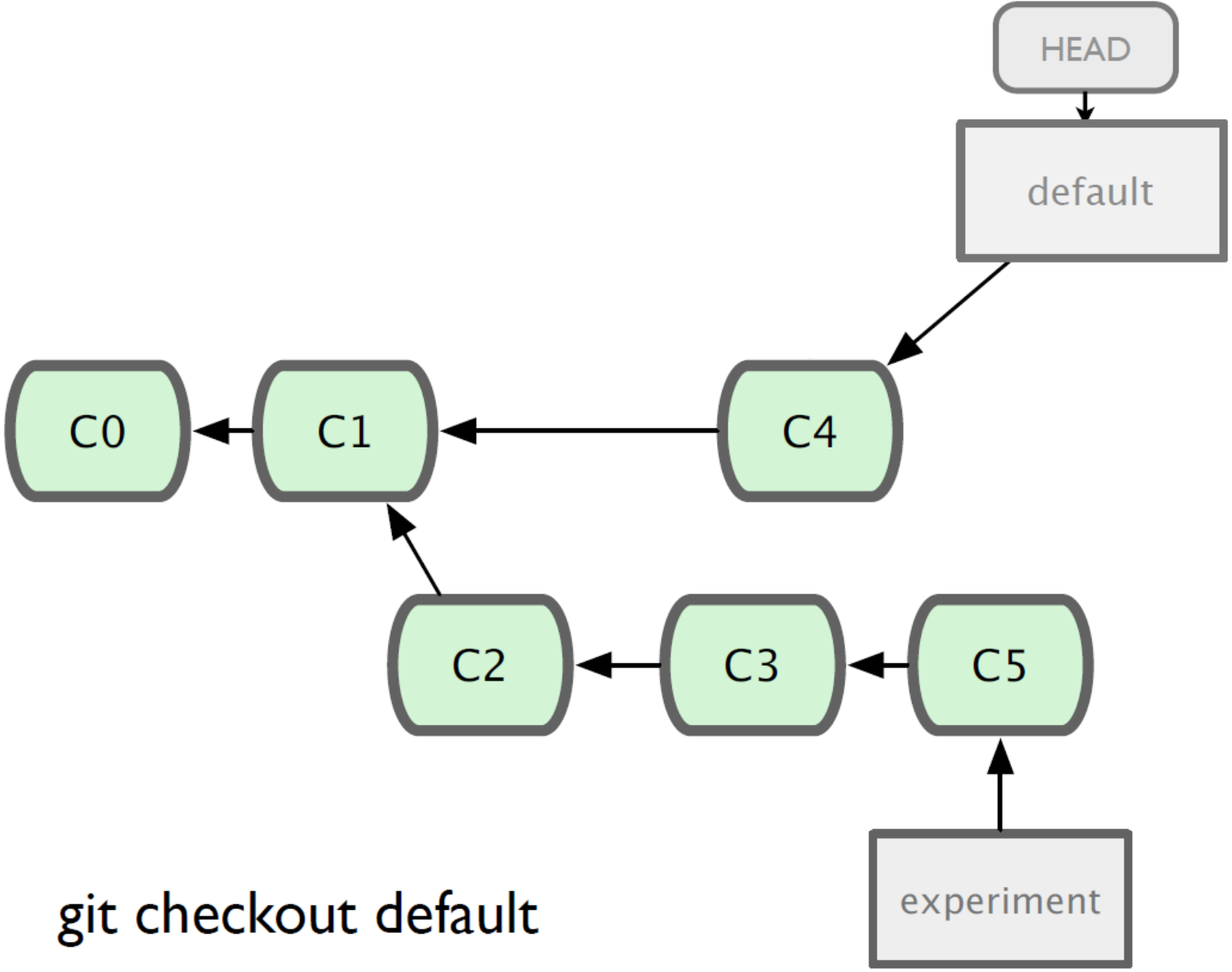
Merging

- What do we do with this mess?
  - *Merge them*

Merging

Checkout the branch you want to merge onto

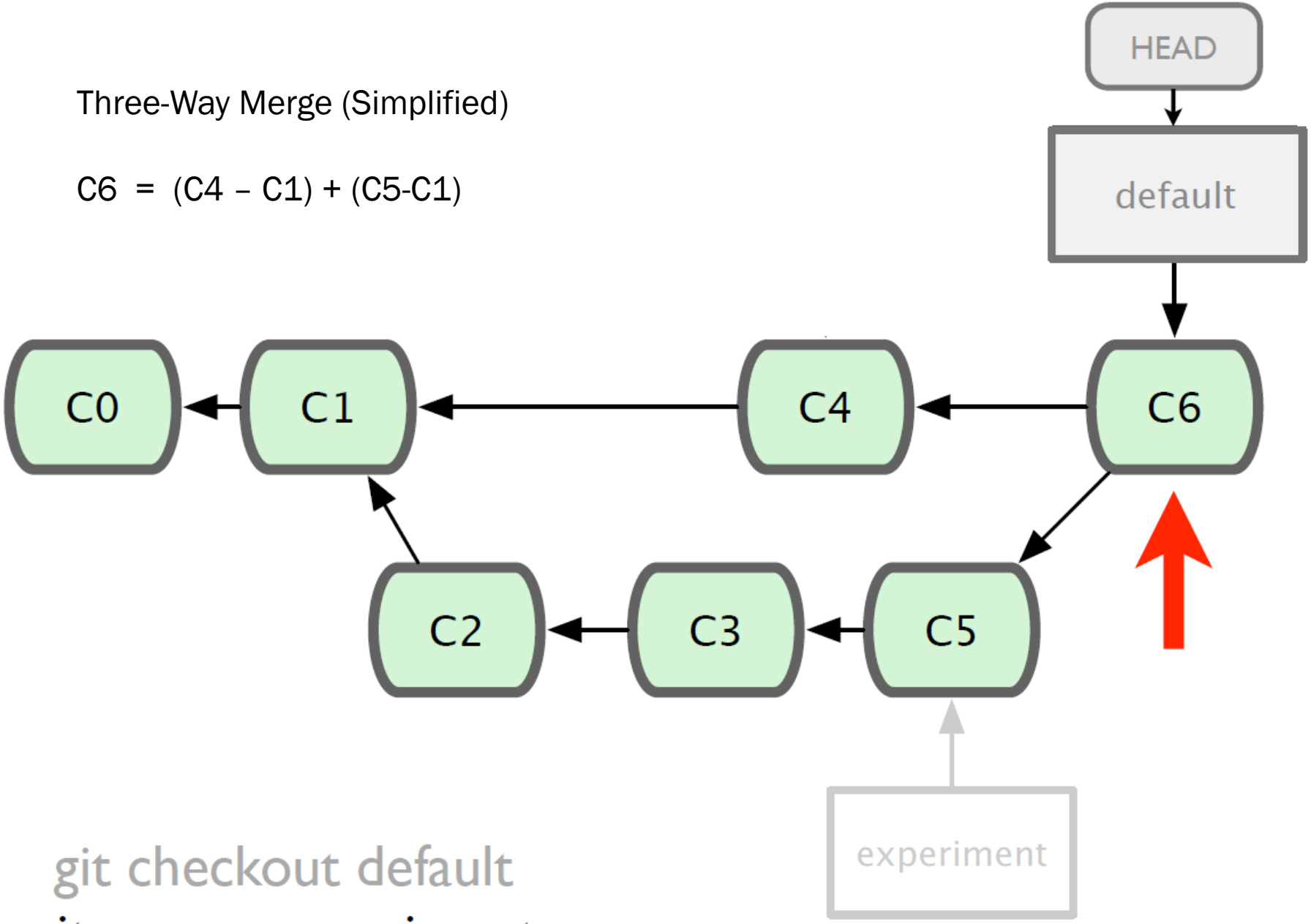
Merge the branch you want to merge



git checkout default

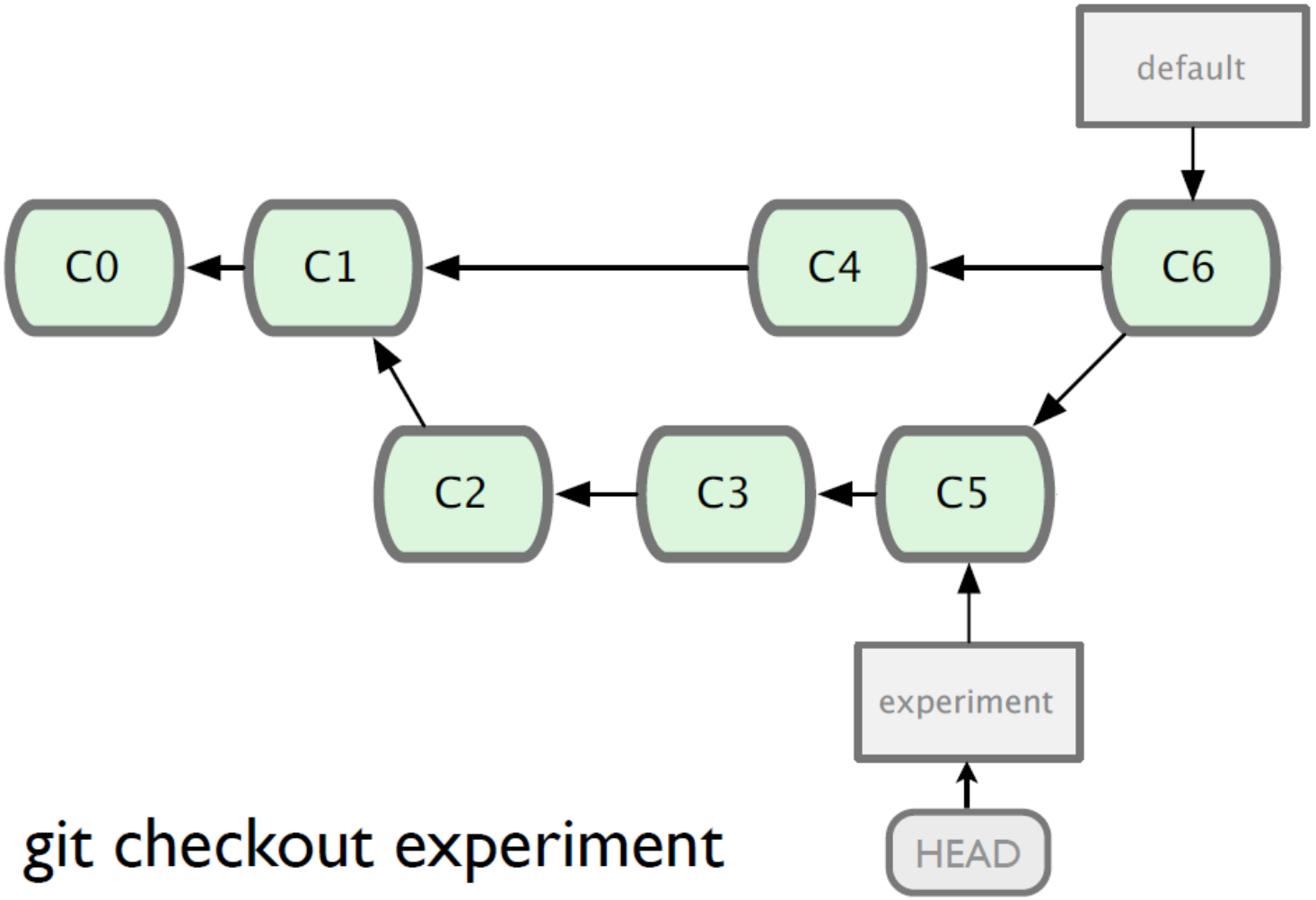
Three-Way Merge (Simplified)

C6 = (C4 – C1) + (C5-C1)

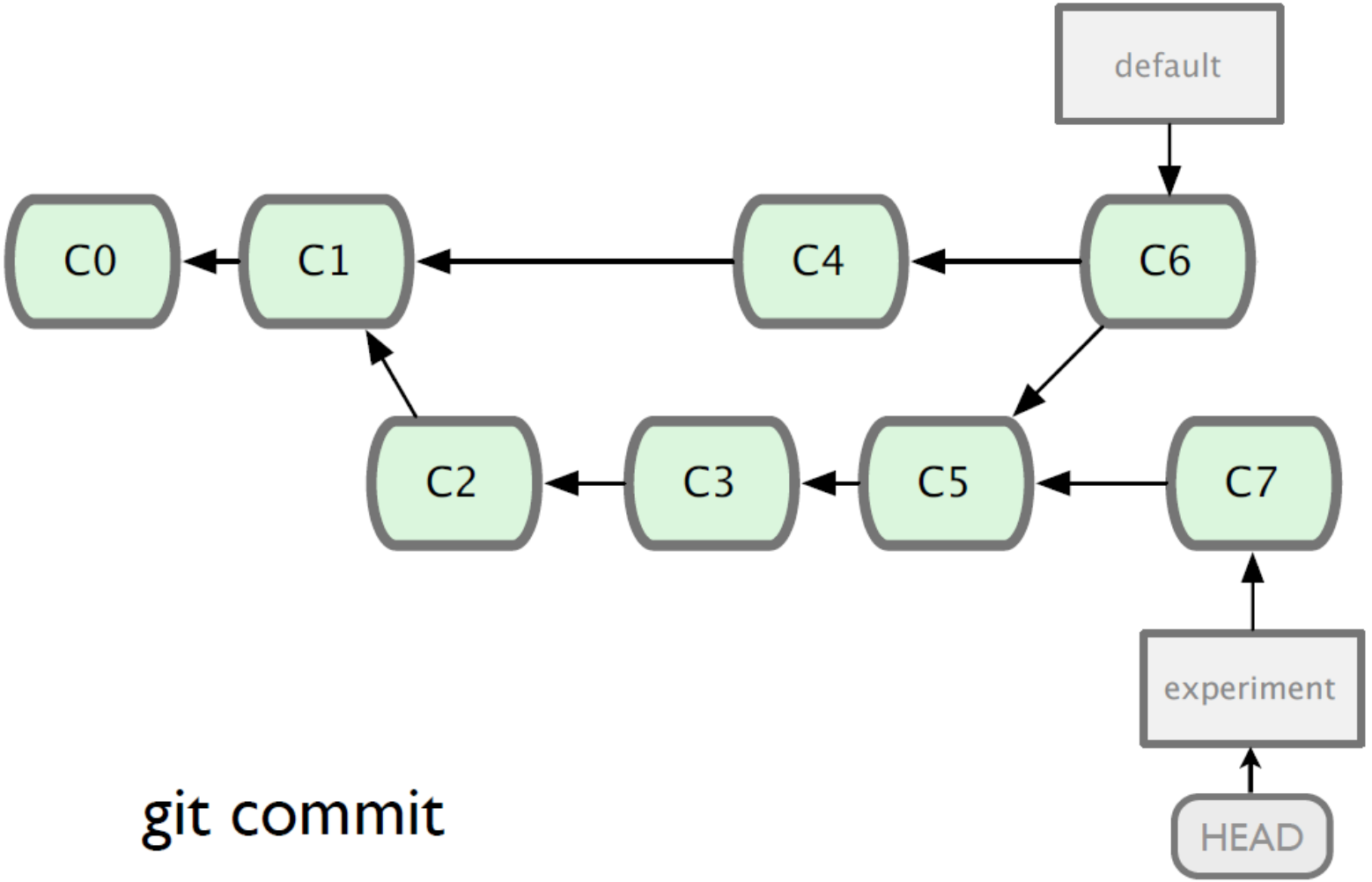


git checkout default

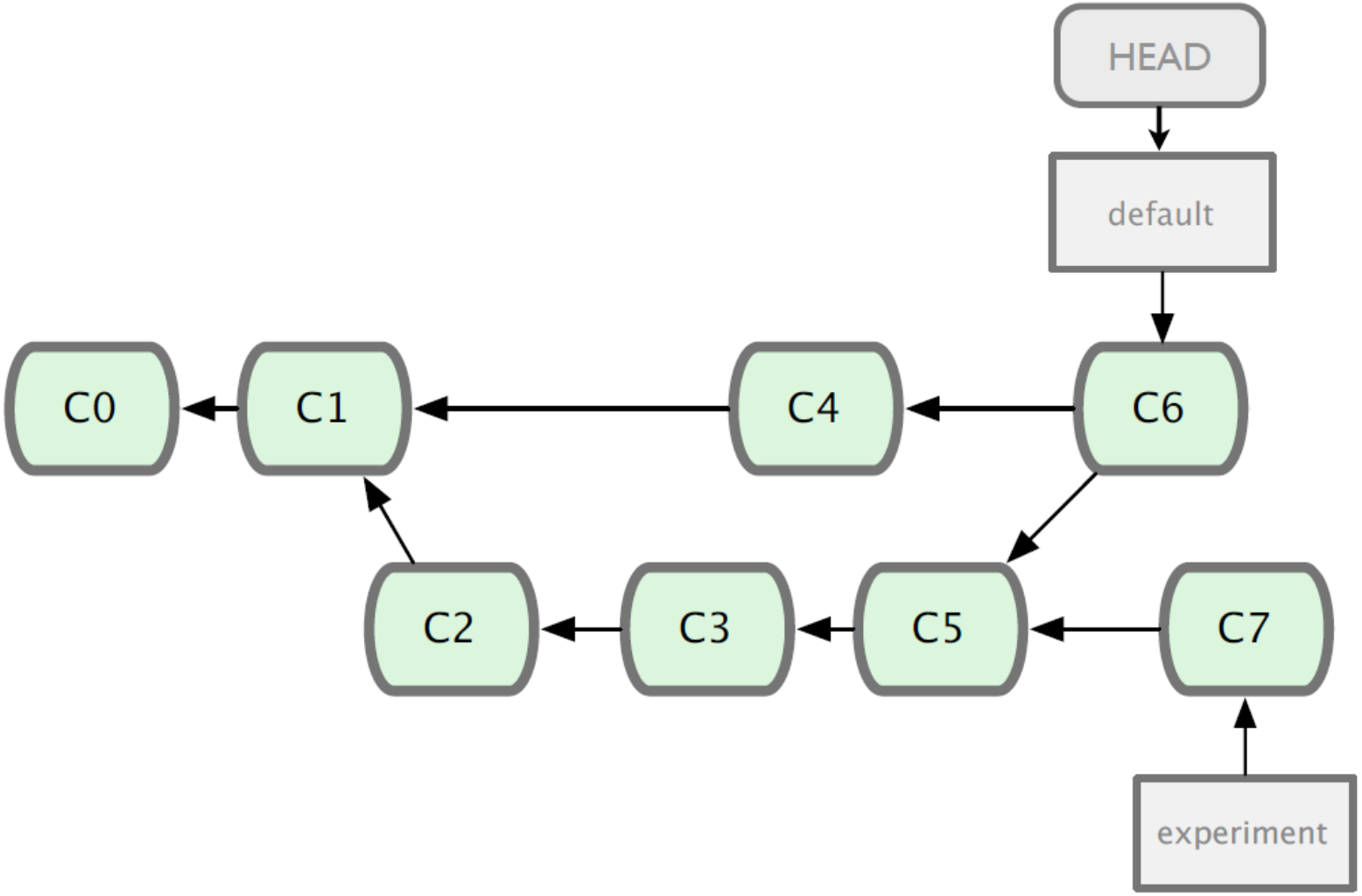
git merge experiment



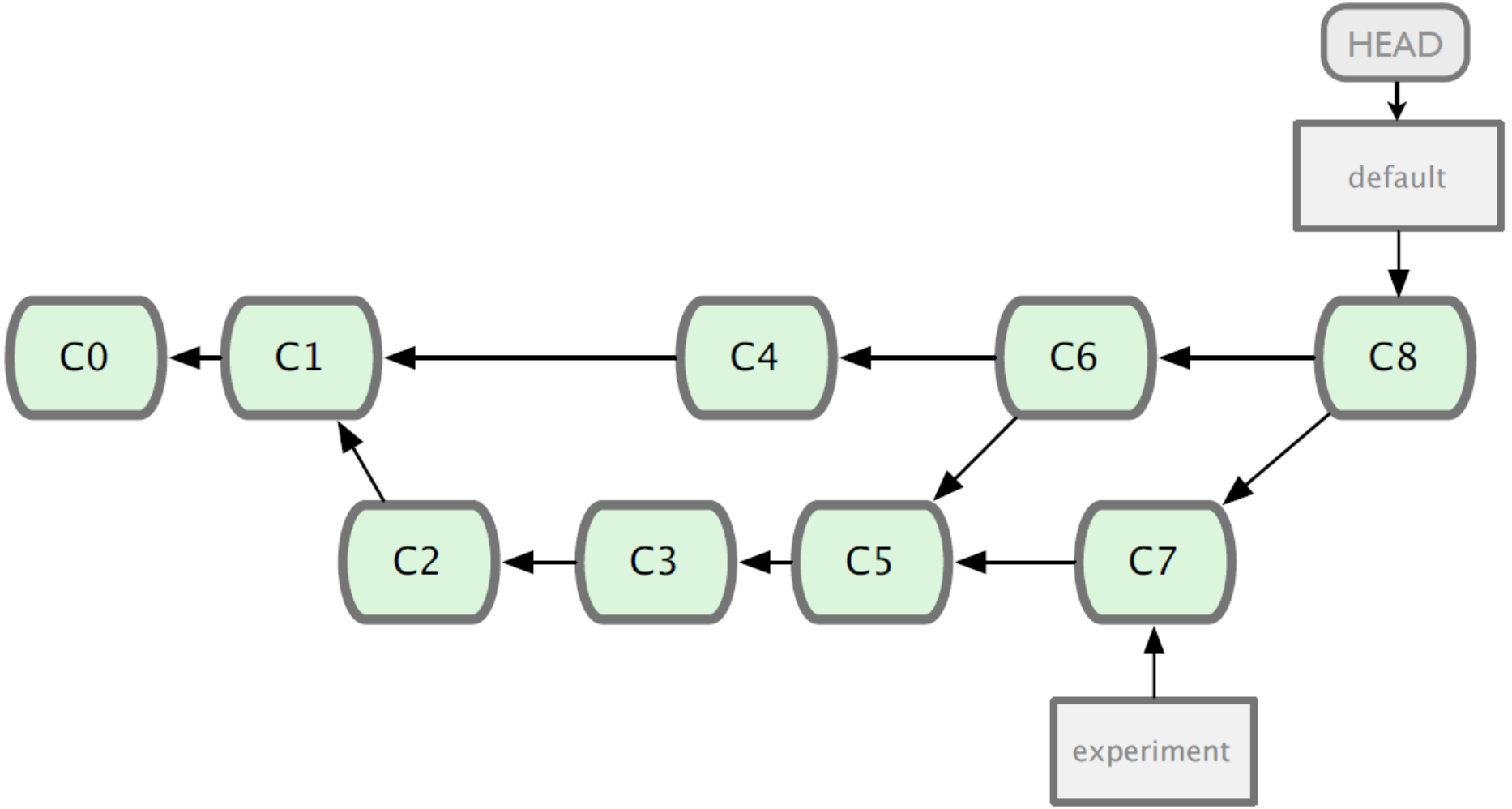
git checkout experiment



git commit



git checkout default



git merge experiment

BRANCHING STRUCTURE

Pro Git if you want to learn more
https://git-scm.com/book/en/v2

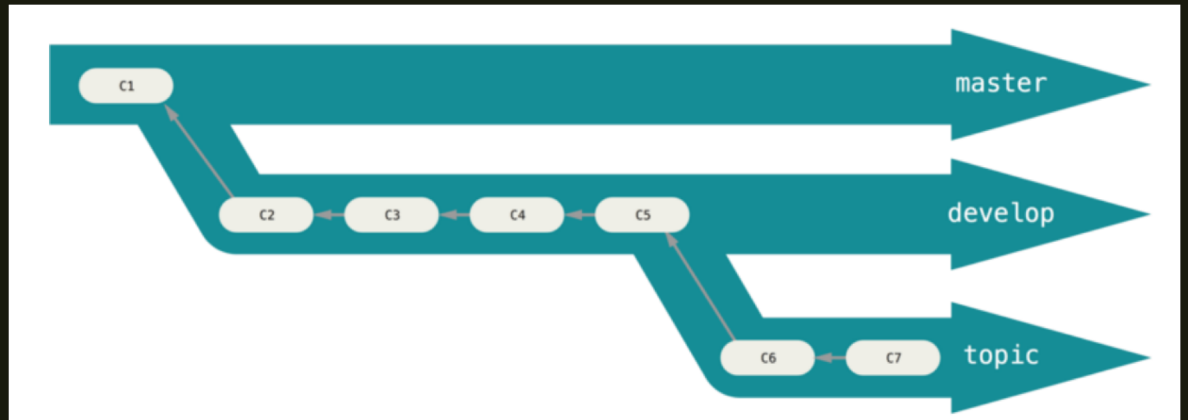
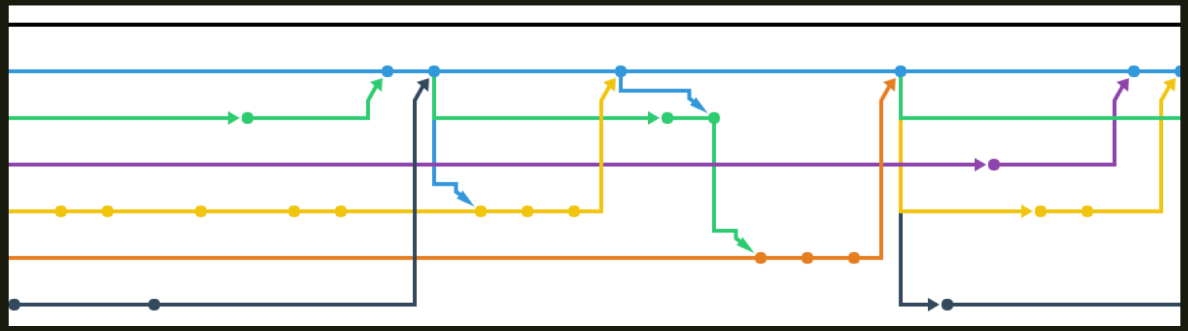


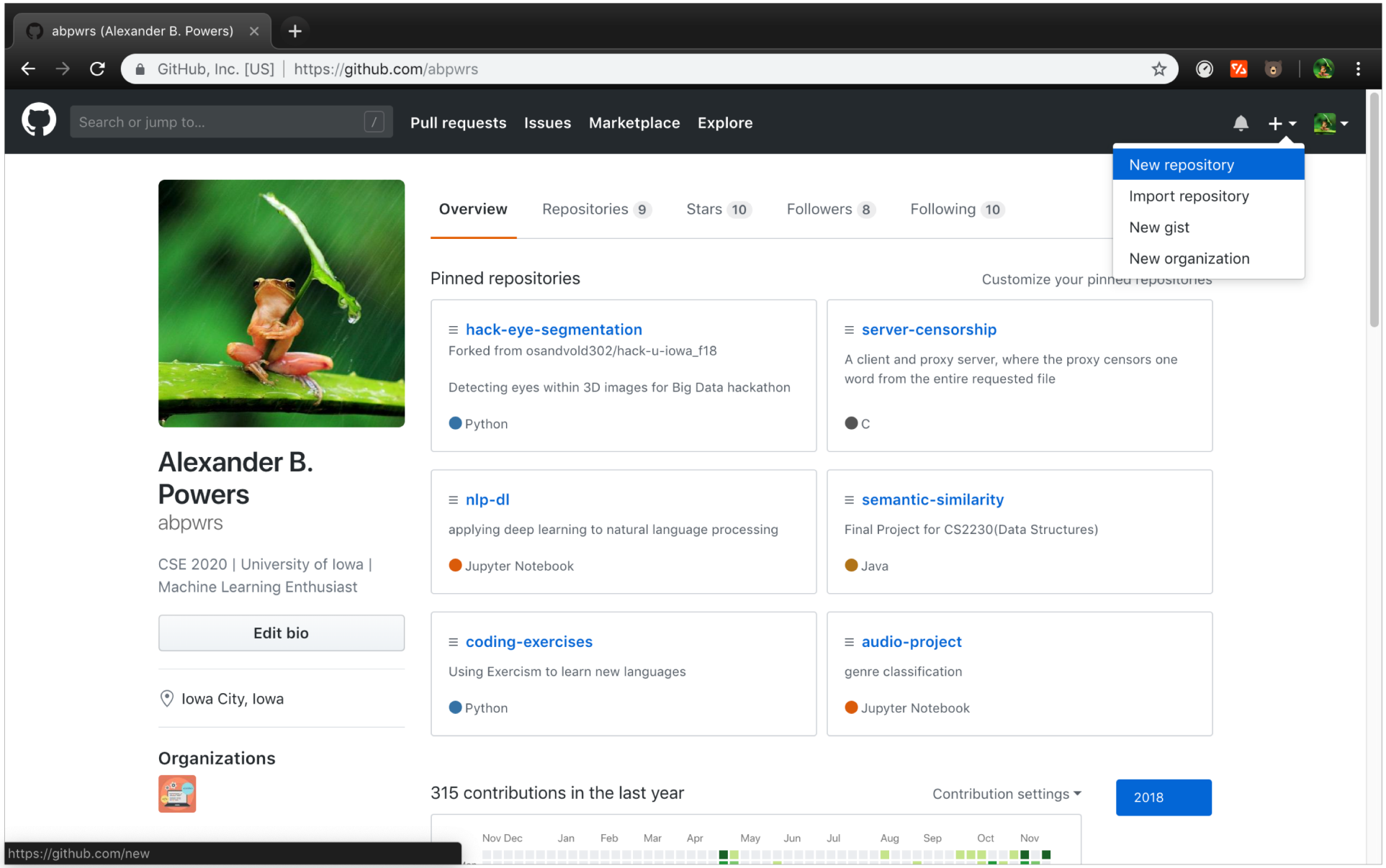
Figure 27. A "silo" view of progressive-stability branching

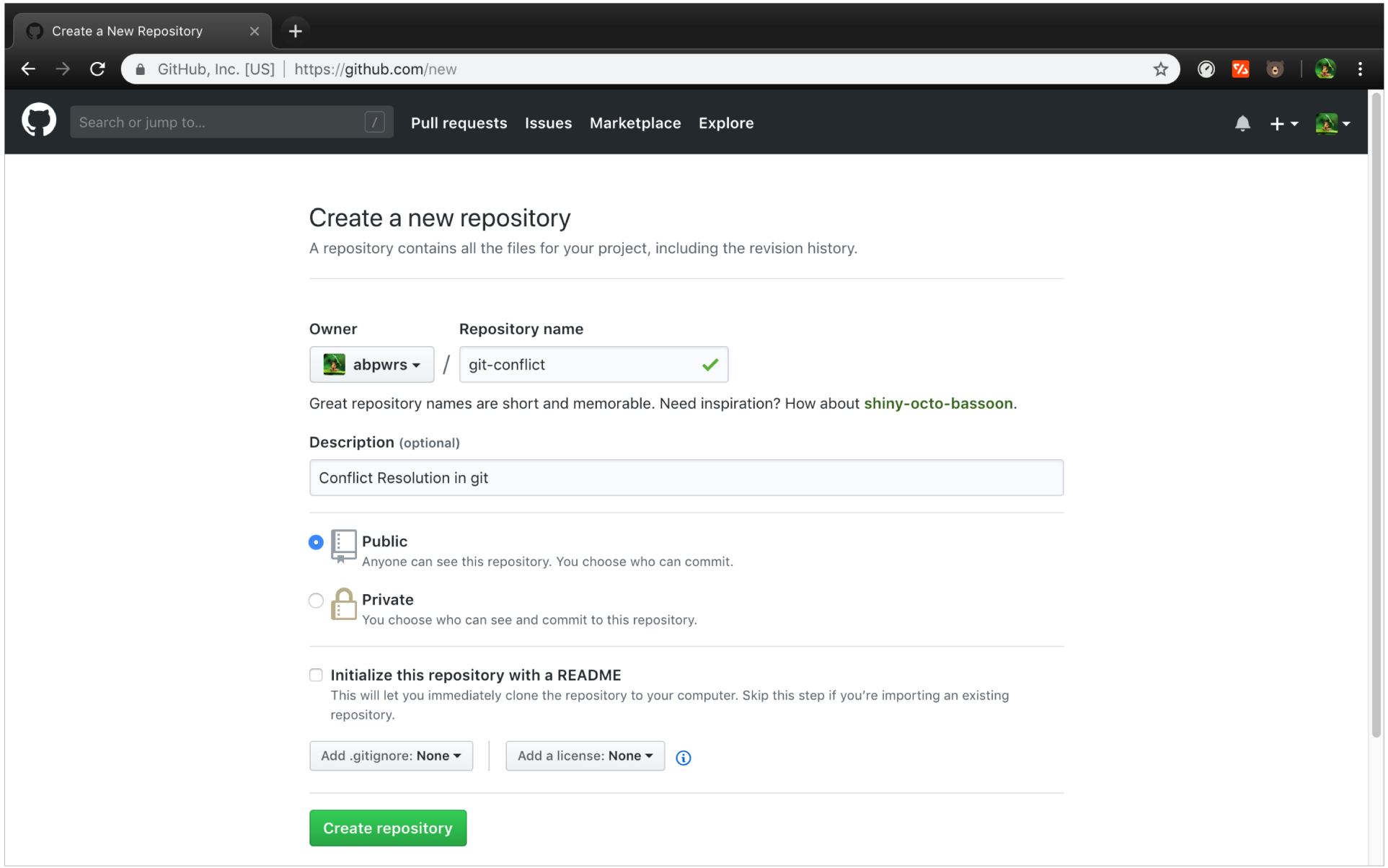


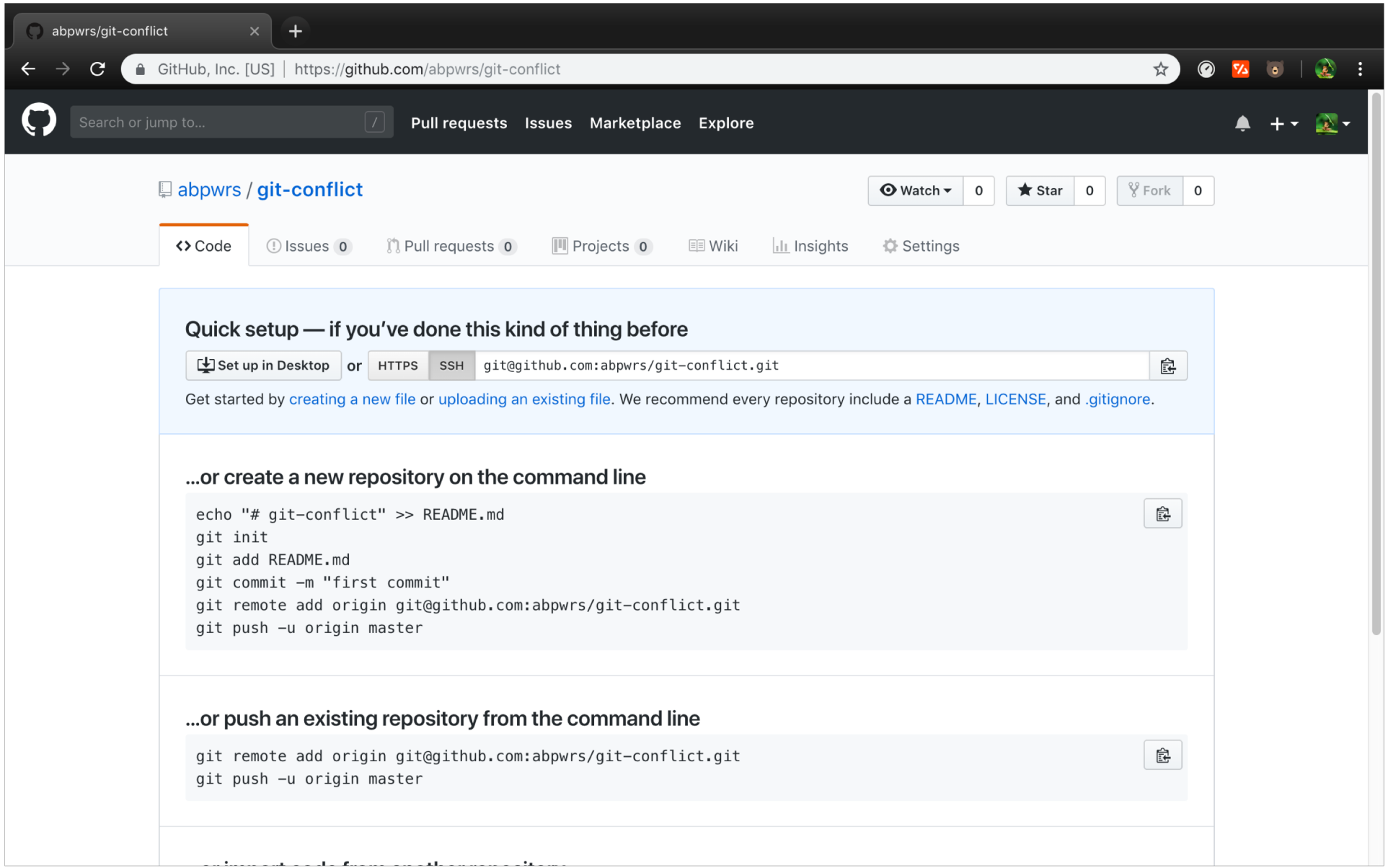
CONFLICT

- What do you do when you run into a merge conflict?!?!?
  - *Panic*
  - *Cry*
  - *Re-clone your repo*
  - *All of the above*
- Lets work through a merge conflict together

MAKE A REPO


abpwrs (Alexander B. Powers)
GitHub, Inc. [US] | https://github.com/abpwrs
Search or jump to...
Pull requests
Issues
Marketplace
Explore
New repository
Import repository
New gist
New organization
Overview
Repositories 9
Stars 10
Followers 8
Following 10
Pinned repositories
Customize your pinned repositories
hack-eye-segmentation
Forked from osandvold302/hack-u-iowa_f18
Detecting eyes within 3D images for Big Data hackathon
Python
server-censorship
A client and proxy server, where the proxy censors one word from the entire requested file
C
nlp-dl
applying deep learning to natural language processing
Jupyter Notebook
semantic-similarity
Final Project for CS2230(Data Structures)
Java
coding-exercises
Using Exercism to learn new languages
Python
audio-project
genre classification
Jupyter Notebook
Alexander B. Powers
abpwrs
CSE 2020 | University of Iowa | Machine Learning Enthusiast
Edit bio
Iowa City, Iowa
Organizations
315 contributions in the last year
Contribution settings
2018
Nov Dec Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
https://github.com/new


Create a New Repository
GitHub, Inc. [US] | https://github.com/new
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Create a new repository
A repository contains all the files for your project, including the revision history.
Owner
Repository name
abpwrs
/
git-conflict
Great repository names are short and memorable. Need inspiration? How about shiny-octo-bassoon.
Description (optional)
Conflict Resolution in git
Public
Anyone can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.
Add .gitignore: None
Add a license: None
Create repository


abpwrs/git-conflict
GitHub, Inc. [US] | https://github.com/abpwrs/git-conflict
Search or jump to...
Pull requests
Issues
Marketplace
Explore
abpwrs / git-conflict
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Quick setup — if you've done this kind of thing before
Set up in Desktop
or
HTTPS
SSH
git@github.com:abpwrs/git-conflict.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.
...or create a new repository on the command line
echo "# git-conflict" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:abpwrs/git-conflict.git
git push -u origin master
...or push an existing repository from the command line
git remote add origin git@github.com:abpwrs/git-conflict.git
git push -u origin master

Initialize local repo to use remote we just made

$ mkdir git-conflict

$ cd git-conflict

$ git init

$ echo “git-conflict resolution” >> README.md

$ git add README.md

$ git commit –m “initial commit”

$ git remote add origin git@github.com:USERNAME/git-conflict.git

$ git push –u origin master

MAKE A CONFLICT

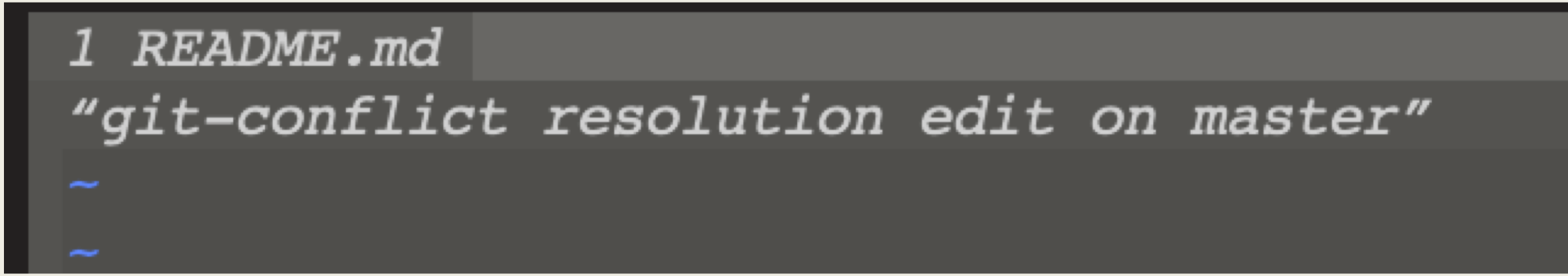
```
[> git branch
 * master
[> git branch conflict-branch
[> git branch
   conflict-branch
 * master
[> git checkout conflict-branch
 Switched to branch 'conflict-branch'
[> git branch
 * conflict-branch
   master
 >
```

```
> l
total 8
-rw-r--r--  1 AlexPowers  staff    30B Nov 19 21:18 README.md
> vim README.md
```

```
1 README.md +
"git-conflict resolution edit on conflict-branch"
~
~
~
~
```

```
[> l
 total 8
 -rw-r--r--  1 AlexPowers  staff    30B Nov 19 21:18 README.md
[> vim README.md
[> git st
 On branch conflict-branch
 Changes not staged for commit:
   (use "git add <file>..." to update what will be committed)
   (use "git checkout -- <file>..." to discard changes in working directory)

         modified:   README.md

 no changes added to commit (use "git add" and/or "git commit -a")
[> git add .
[> git ci -m "modified README.md on conflict branch"
 [conflict-branch 92b91be] modified README.md on conflict branch
  1 file changed, 1 insertion(+), 1 deletion(-)
 >
```

```
> git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
> vim README.md
```

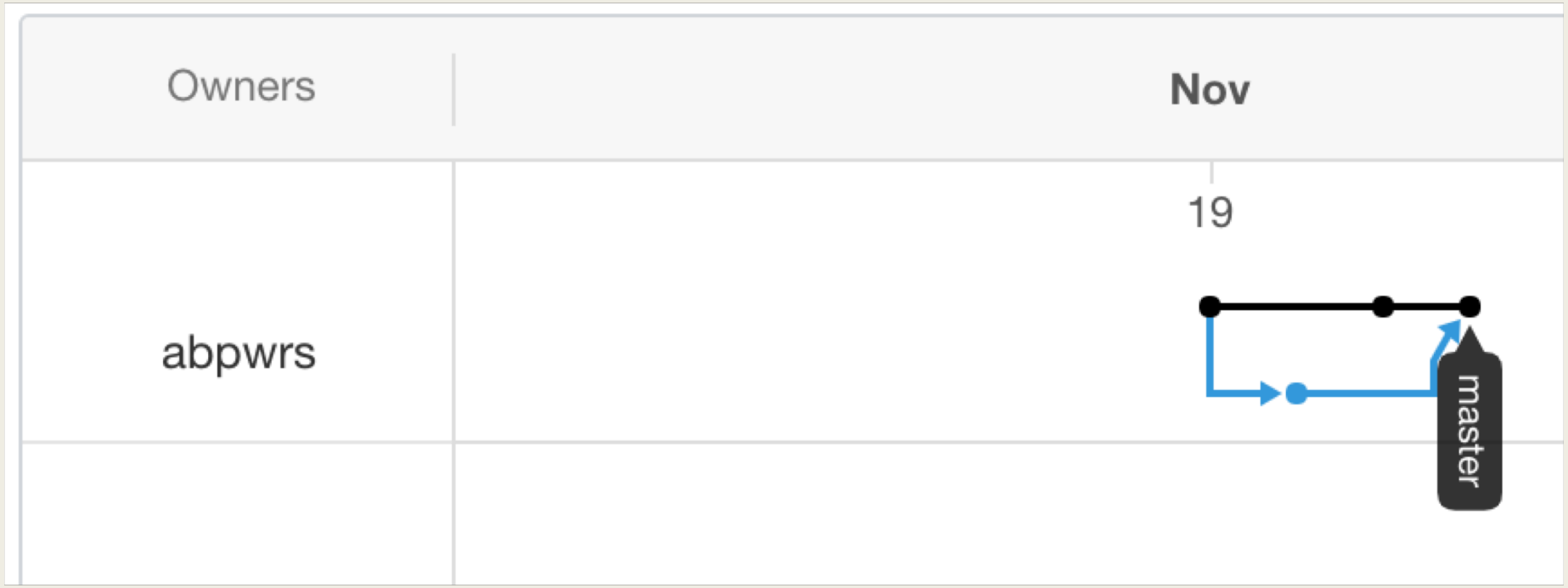

1 README.md
"git-conflict resolution edit on master"
~
~

```
[> git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
[> vim README.md
[> git st
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
[> git add .
[> git ci -m "edit on master to README.md"
[master ba9cbbe] edit on master to README.md
 1 file changed, 1 insertion(+), 1 deletion(-)
[> git merge conflict-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
> _
```

```
1 README.md
<<<<<<< HEAD
"git-conflict resolution edit on master"
=======
"git-conflict resolution edit on conflict-branch"
>>>>>>> conflict-branch
~
~
```

```
> git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
> vim README.md
> git st
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
> git add .
> git ci -m "edit on master to README.md"
[master ba9cbbe] edit on master to README.md
 1 file changed, 1 insertion(+), 1 deletion(-)
> git merge conflict-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
> vim README.md
> git st
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
> git add .
> git ci -m "fixing conflict"
[master 63ed43c] fixing conflict
> _
```

This results in the following



QUESTIONS?

THANK YOU!