

# Version Control & GIT

Presented By:

# \$man git

## **NAME**

git - the stupid content tracker

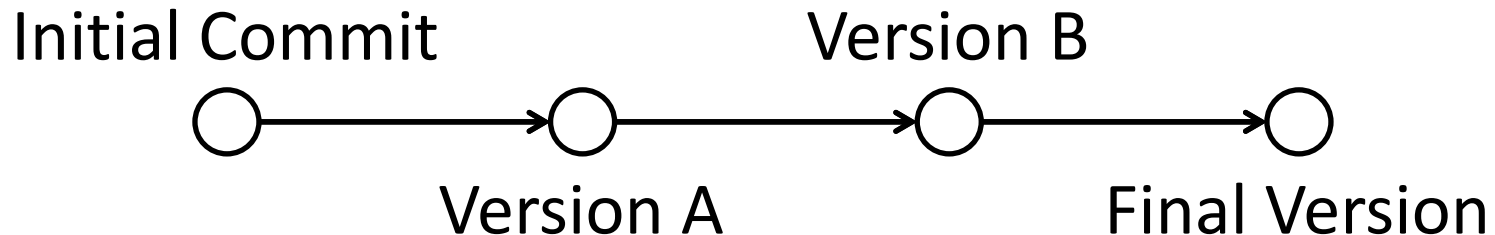
## **DESCRIPTION**

GIT is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

# Theory (Simplified)

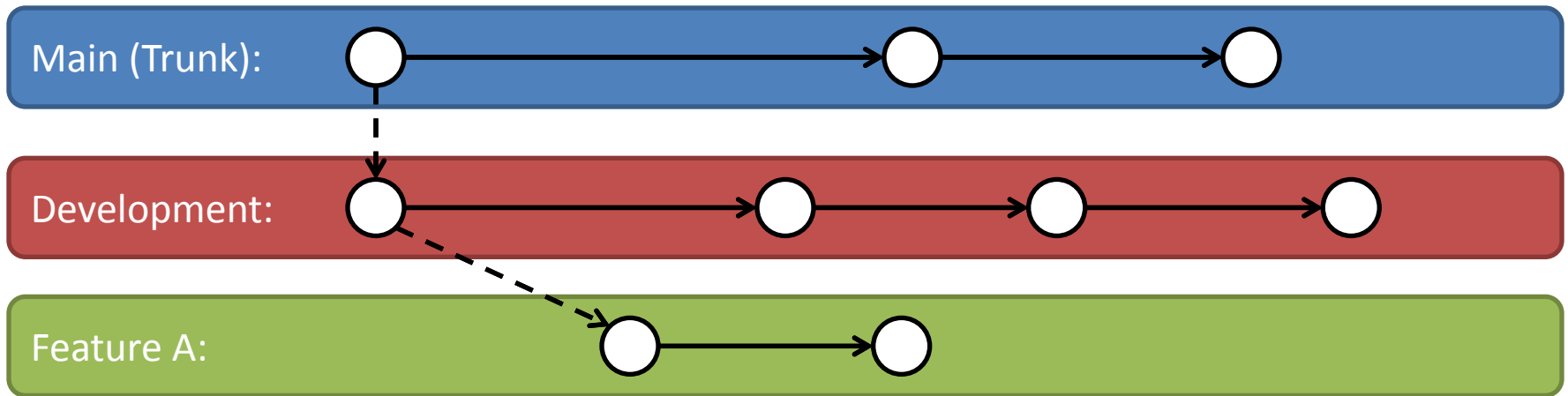
A Simplified Summary of Commit,  
Branches, and Merging

# Commits



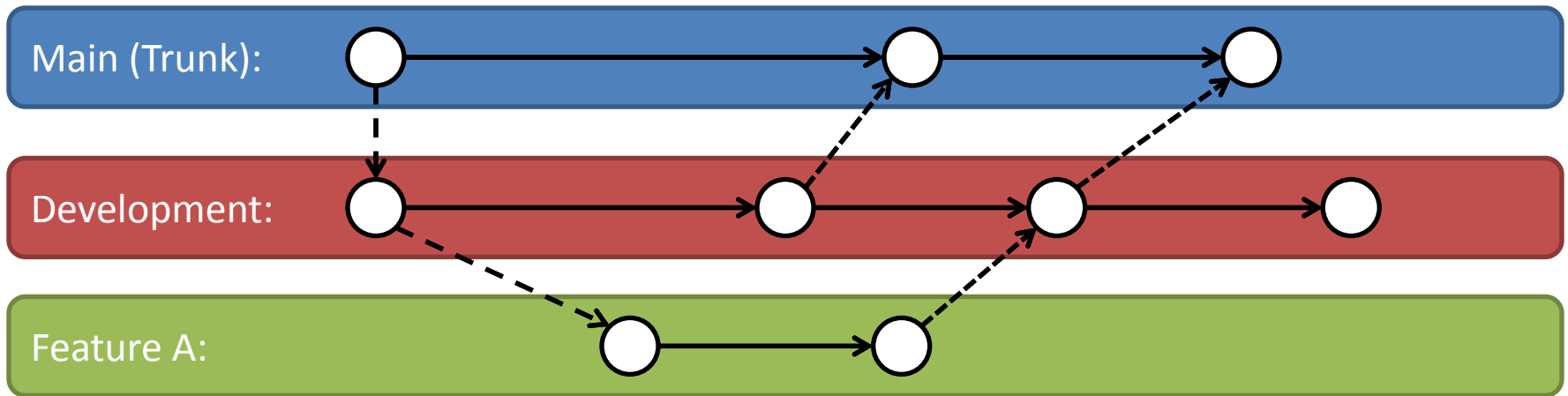
- Commits
  - Represented as nodes on a graph.
  - Symbolize an instance (version) of a project's files.
  - Can add, remove, or modify a file (or multiple files).
- (Implementation) Features:
  - Copy on Write- Only save the changes

# Branches



- Branch
  - A branch is a distinct variant of the project base.
  - Branches allow isolated environments for working on new features and fixes without disrupting the project.
  - Branches are created by “**forking**” from either the main branch (Trunk) or another branch.

# Merging



- Merge
  - A branch may be merged into another.
  - The branch creates a “**pull request**” requesting that its present version be merged into another. The requestor details how to make the changes.
  - An authority from the branch being merged into then is expected to review the request.

# The Case for Version Control

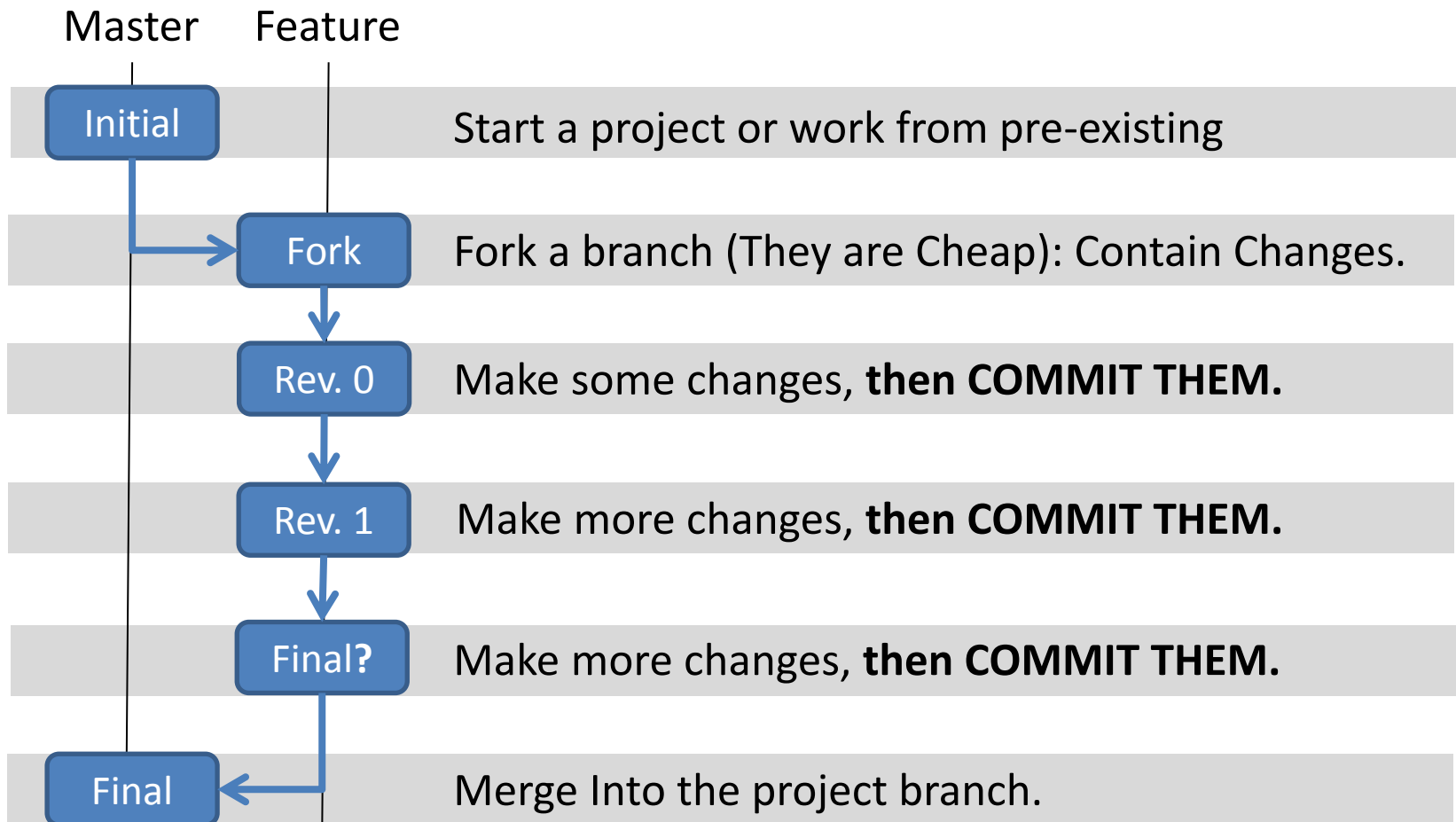
(Or How You Have Already Made It)

# CASE A: Giant Undo Button

- Implement a solution, doubt, and delete.
  - Is there a backup of an older version?
    - Is the older version the most recent?
    - What work, if any, must be redone (lost productivity)?
- This is why **backups are important**
  - This is a fundamental tenant of version control.
  - **Lesson 1: VERSION CONTROL AS BACKUP**



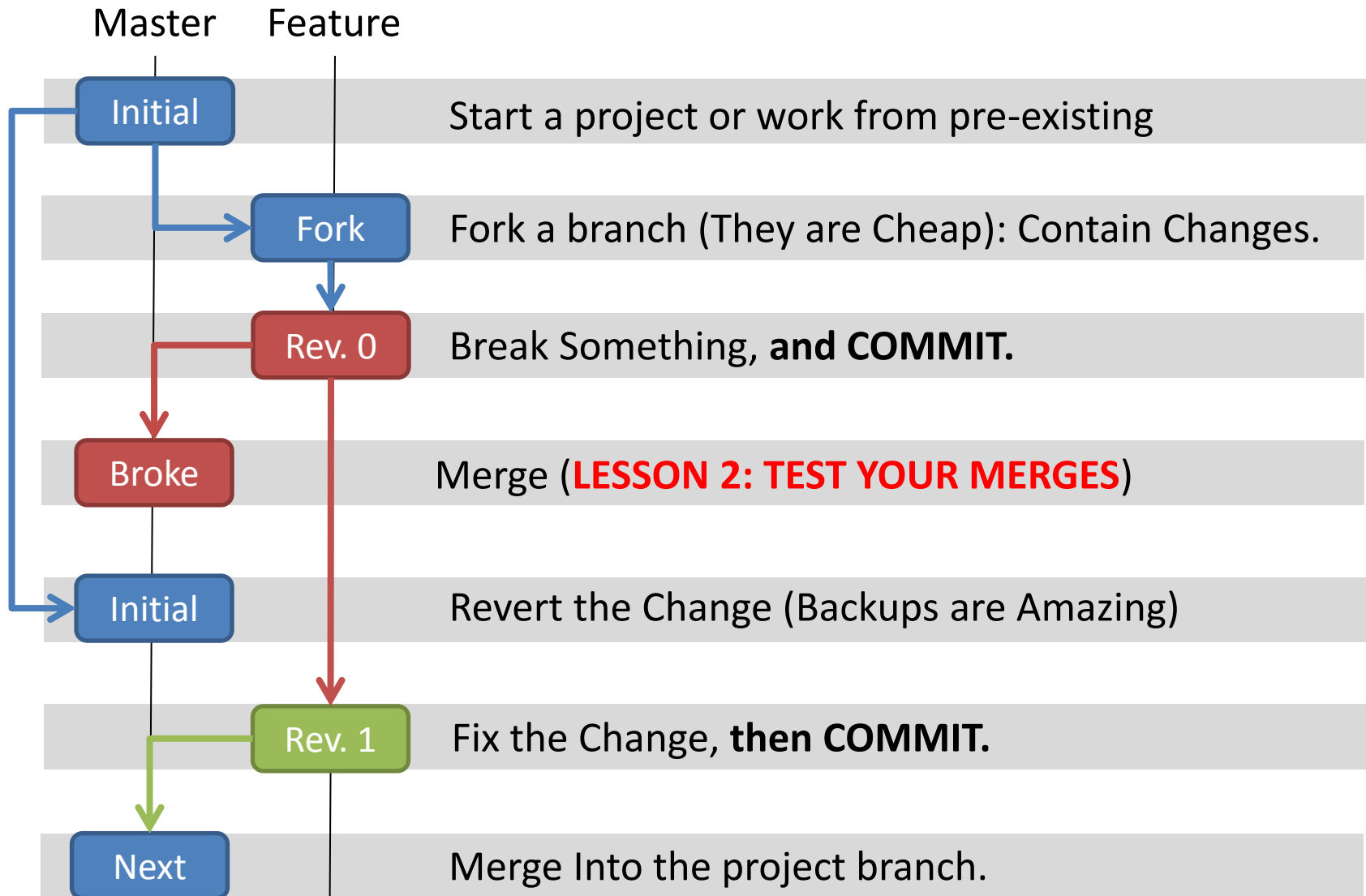
# Solution A: Save Incremental Changes



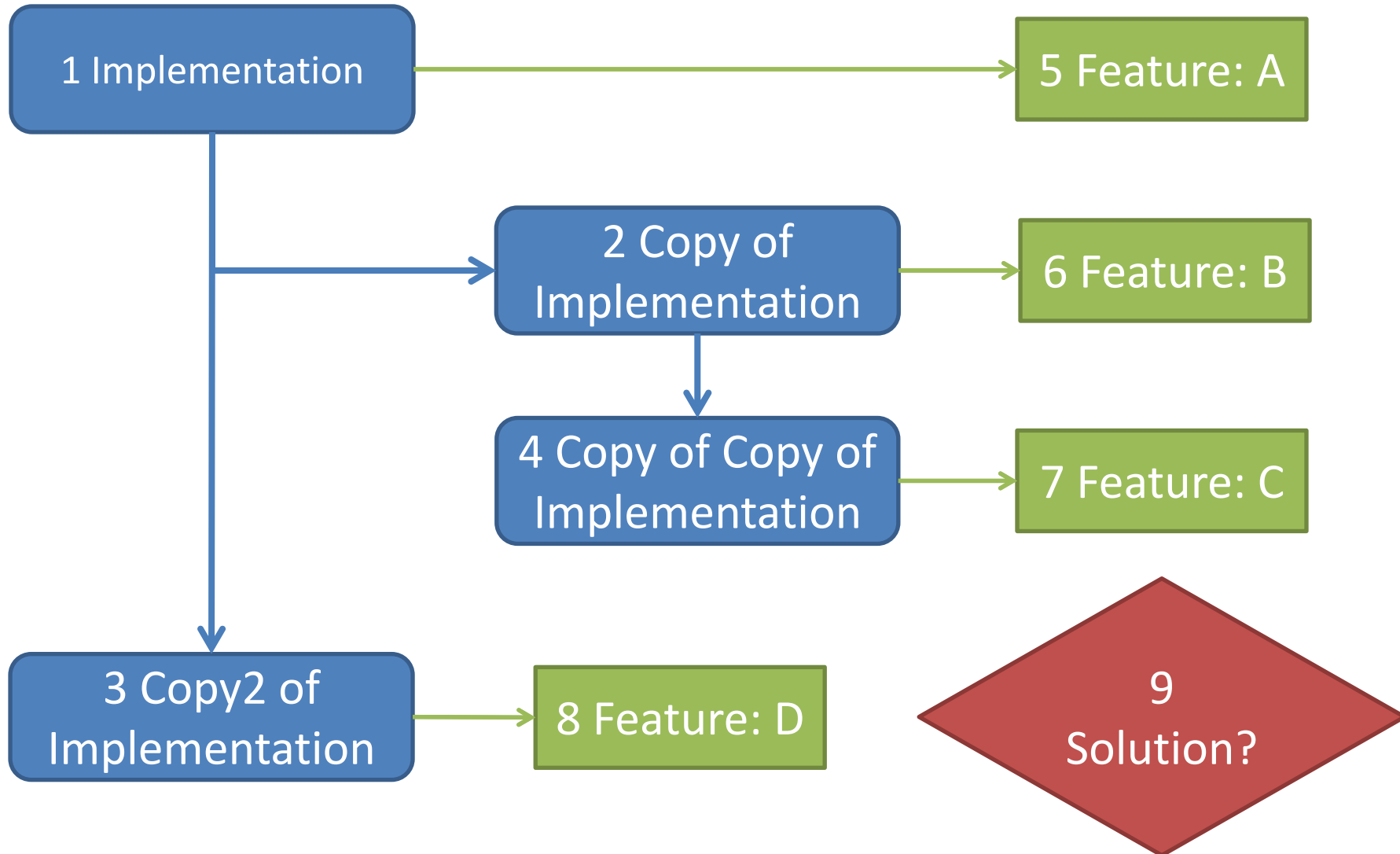
# CASE B: The New Breaks The Old

- Fixing a smaller problem breaks a significant feature.
  - What changed?
    - First step in fixing the problem.
    - Humans **SUCK** at answering this question.
  - **Can I revert the changes?**
    - This is why version control is important.

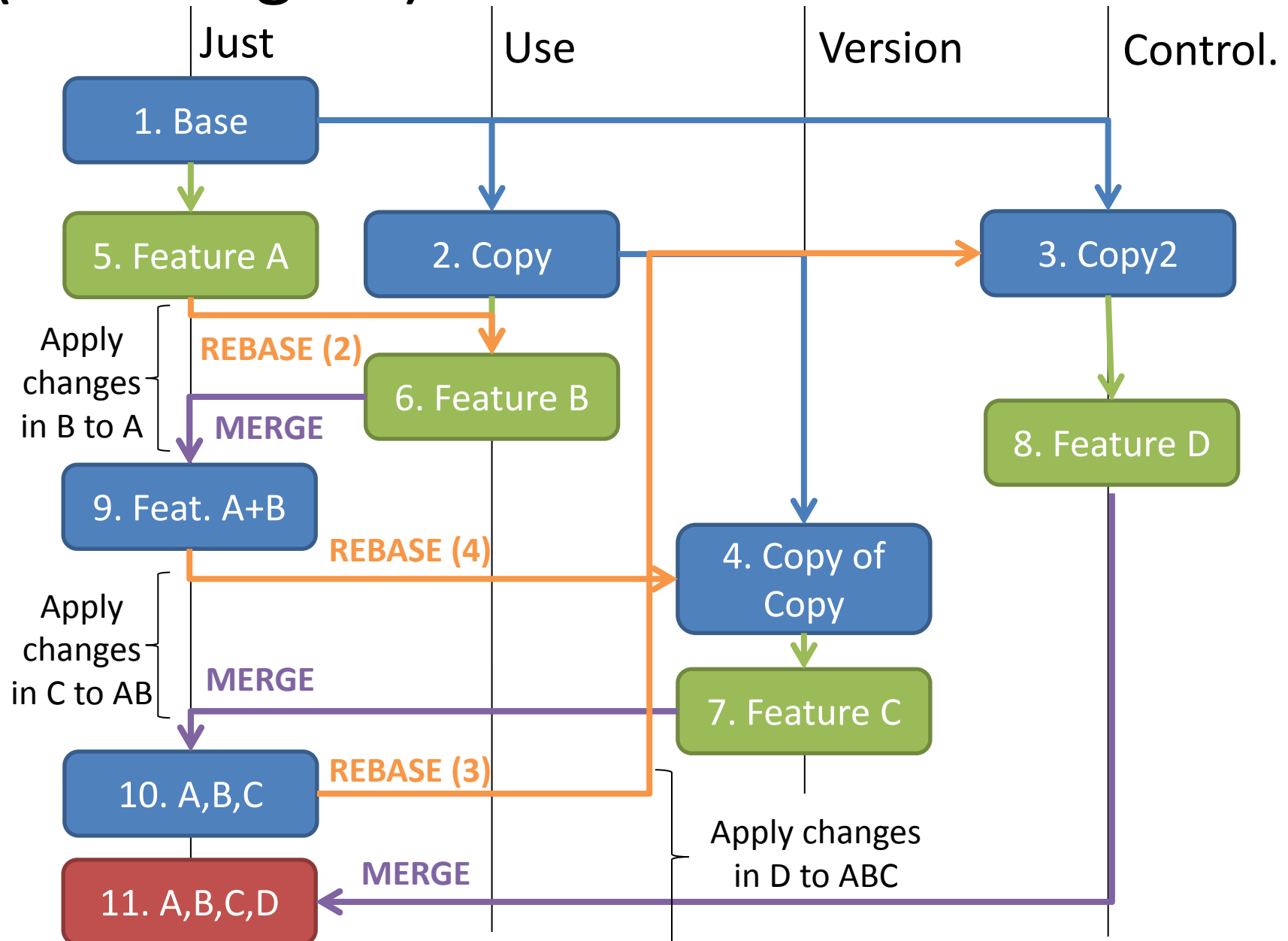
# Scenario B: Something Broke



# CASE C: Multiple Copies



# (Tautological) Solution C: Version Control

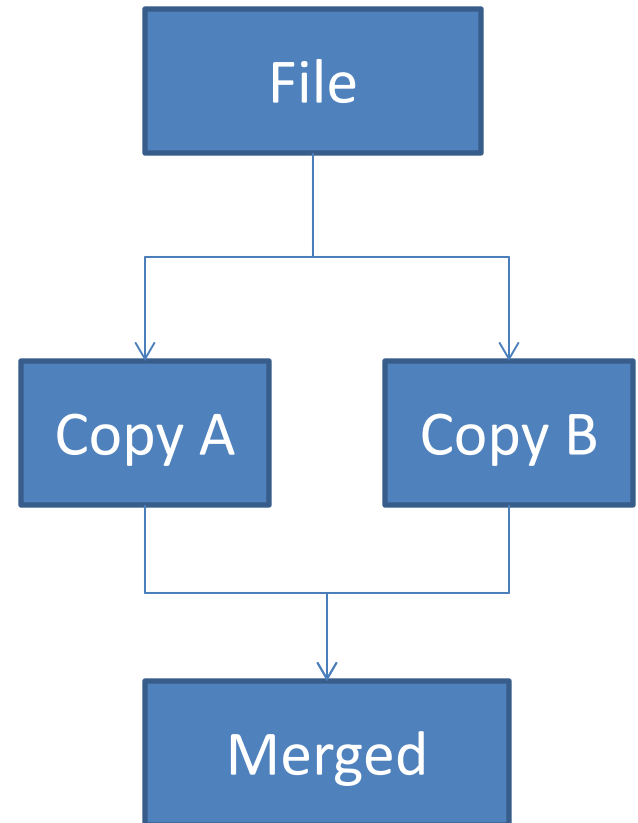


# Engineering: Applications

Why should I care?

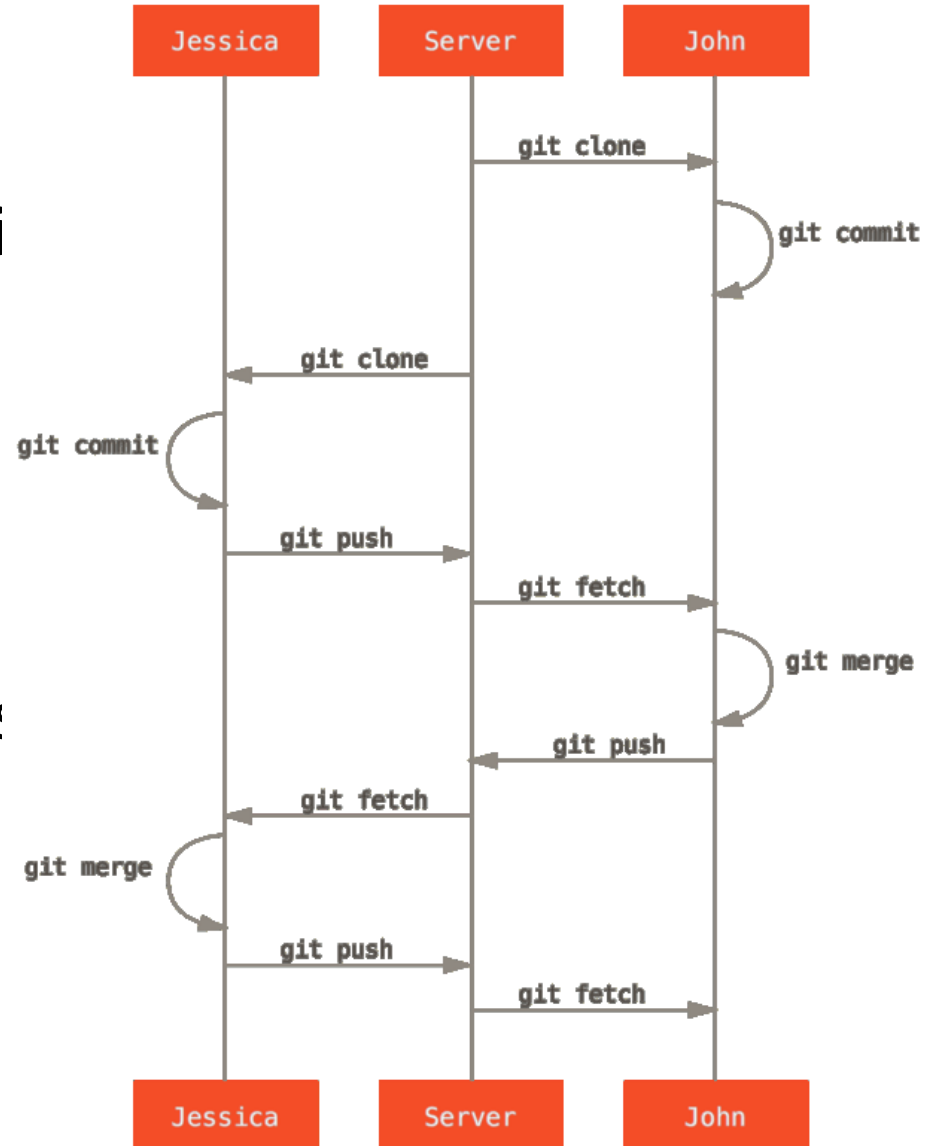
# CASE 1: Multiple Team Members

- Multiple individuals need to work on the same file.
  - Option A: Only one may work on the file at a time (Checkout Model).
  - Option B: Everyone gets a copy, but you must reconcile the differences.



# Solution 1: User Branches

- GIT follows Option B.
  - Some overhead reconciling differences is created.
  - GIT's model requires changes be made with respect to the current version.
    - Remember REBASE.





# CASE 2: Everything is Broke

- Changes are made irresponsibly.
  - Commits track whom made the change.
    - “**BLAME**” Cites whom made the previous change(s)
  - “**DIFF**” allows side-by-side comparison of versions logging additions, changes, and removals.

# Solution 2: Diff

## GitHub (3<sup>rd</sup> Party) Implementation

88 ■■■■■ Makefile View ▾

@@ -7,11 +7,55 @@ BINARY\_NAME = circle\_box

7

8 # COMPILER OPTIONS

9 COMPILE\_OPTIONS = -pthread

10 -COMPILE\_OPTIMIZATION = -O0

11

12 -# [DEBUGGING] Settings

13 -LINK\_DEBUG = -g3

14 #https://gcc.gnu.org/onlinedocs/gcc-4.2.4/gcc/Debugging-Options.html#Debugging-Options

7

8 # COMPILER OPTIONS

9 COMPILE\_OPTIONS = -pthread

10

11 +#####

12 +# Support for Additional compile-time parameters

13 +#####

14 +# override directive utilized for text assertions printed to end-user

15 +#https://www.gnu.org/software/make/manual/make.html#Override-Directive

16 +

17 +# [BUILD TYPE]

18 +# Defaults to debug build configuration if not specified

19 +# Supported options: debug, internal, release

20 #https://gcc.gnu.org/onlinedocs/gcc-4.2.4/gcc/Debugging-Options.html#Debugging-Options

21 +ifeq (\$(build),release)

22 + override build = RELEASE

23 + LD\_BUILD =

24 + CPP\_DEBUG =

25 +else

26 + ifeq (\$(build),internal)

27 + override build = INTERNAL

28 + LD\_BUILD =

29 + CPP\_DEBUG = -g3

30 + else

31 + # DEFAULT CASE

32 + override build = DEBUG

33 + LD\_BUILD = -Xlinker -M=\$(BINARY\_NAME).map -Xlinker --cref

34 + CPP\_DEBUG = -pedantic -Wextra -Wconversion -g3

35 + endif

36 +endif

37 +

38 +# [OPTIMIZATIONS]

39 +# Defaults to NO optimizations

40 +# Supported options: O0, O1, O2, O3

41 +ifeq (\$(optimize),O3)

↑

-{Removed}

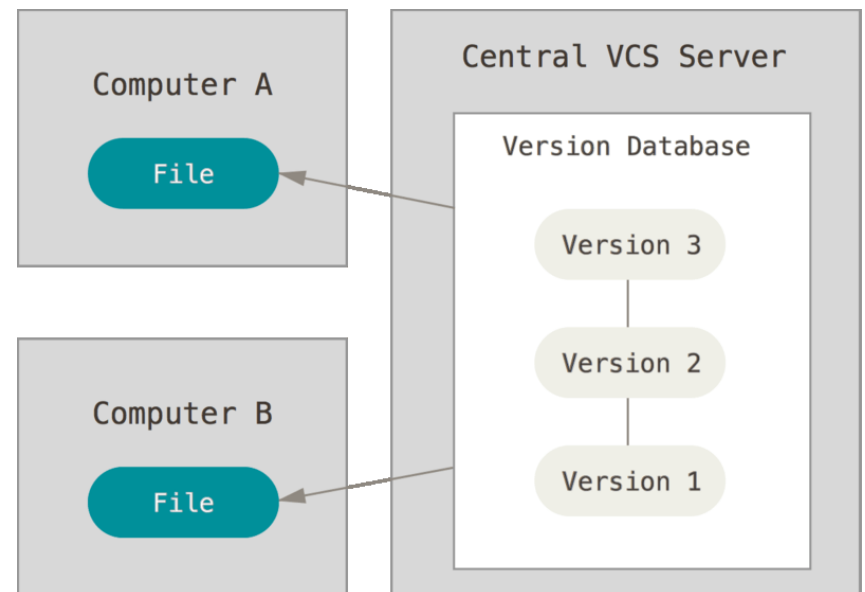
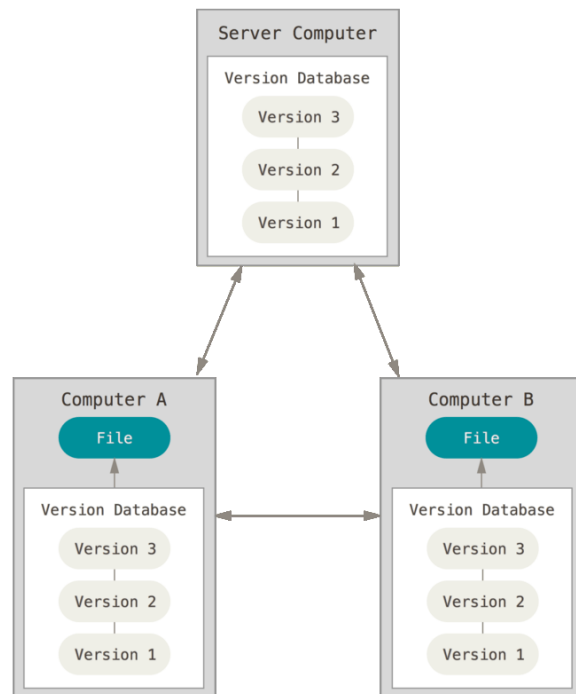
+{Added}

→

Note: Plain text files work best.  
Binary files are not supported.

# No (Internet) Connection?

- Distributed Version Control
  - The user “**clone**” a repository.
  - The copy is a complete version
  - GIT uses this model.
- Centralized Version Control
  - The user has copies of files.
  - The repository only on the server
  - Subversion (SVN) uses this model.

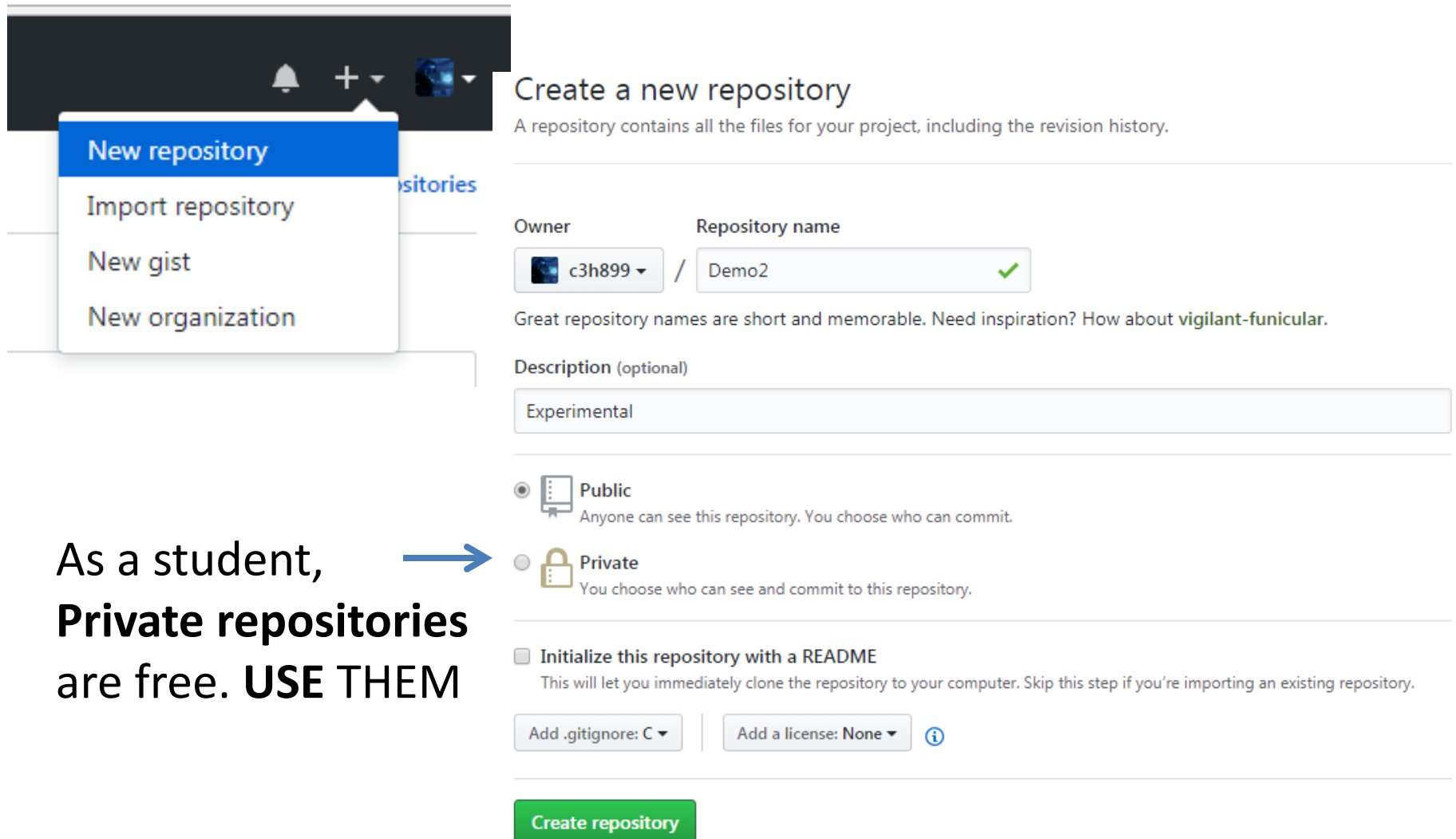


# Getting Started

- GIT
  - <https://git-scm.com/downloads>
- Choose a license
  - <https://choosealicense.com/>
- Configure .gitignore
  - <https://www.gitignore.io/>
- Consider a Web Front-End
  - <https://education.github.com/pack> (Recommended)
  - <https://about.gitlab.com/pricing/>

# Simple GitHub Usage

# Create a New Repository



**Create a new repository**

A repository contains all the files for your project, including the revision history.

Owner: c3h899 / Repository name: Demo2 ✓

Great repository names are short and memorable. Need inspiration? How about **vigilant-funicular**.

Description (optional): Experimental


☒ **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: C | Add a license: None ⓘ

**Create repository**

As a student,  **Private repositories**  
are free. **USE THEM**

# Create an Experimental Branch

Experimental

[Manage topics](#)

🔄 1 commit

1. Click Here

Branch: master ▼

New pull request

2. Give it a Name

Experimental

Branches

Tags

3. Create the Branch

🔗 Create branch: Experimental  
from 'master'

# Add Collaborators (Optional)

The screenshot shows the GitHub repository settings interface. At the top, a navigation bar contains links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. The Settings link is highlighted with a red box and labeled (1). On the left, a sidebar lists various settings categories: Options, Collaborators, Branches, Webhooks, Integrations & services, Deploy keys, Moderation, and Interaction limits. The Collaborators category is highlighted with a red box and labeled (2). The main content area is titled 'Collaborators' and includes a link to 'Push access to the repository'. It contains a message stating that the repository has no collaborators yet and provides instructions on how to search for users by username, full name, or email address. A search input field is present, and a button labeled 'Add collaborator' is highlighted with a red box and labeled (3).

(1) Settings

(2) Collaborators

Options  
Collaborators  
Branches  
Webhooks  
Integrations & services  
Deploy keys

Moderation  
Interaction limits

Collaborators [Push access to the repository](#)

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

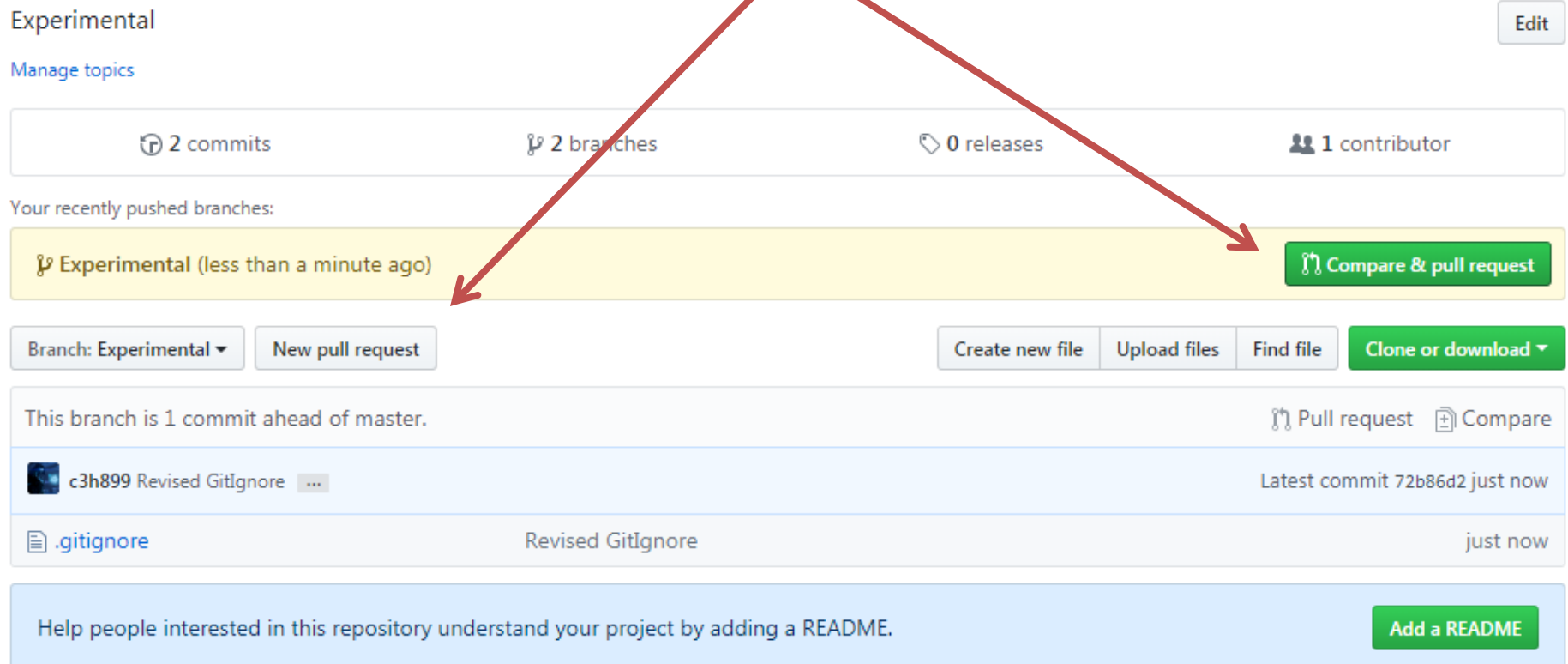
**Search by username, full name or email address**  
You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

(3) Add collaborator



# Merging Branches (Pull Request) 1/3

1. Click Here



The screenshot shows a GitHub repository interface. At the top, there's a header with 'Experimental' and an 'Edit' button. Below this, a summary bar shows '2 commits', '2 branches', '0 releases', and '1 contributor'. A section titled 'Your recently pushed branches:' contains a yellow bar for the 'Experimental' branch, noted as 'less than a minute ago'. A red arrow points from the text '1. Click Here' to the 'Compare & pull request' button on the right of this bar. Below the branch list, there's a bar with 'Branch: Experimental', a 'New pull request' button, and buttons for 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The main content area shows 'This branch is 1 commit ahead of master.' and a list of files, including '.gitignore' and 'Revised GitIgnore'. At the bottom, a blue bar encourages adding a README.

Experimental [Manage topics](#) [Edit](#)


🔖 2 commits 🌿 2 branches 📦 0 releases 👤 1 contributor

Your recently pushed branches:

🌿 Experimental (less than a minute ago) [Compare & pull request](#)

Branch: Experimental [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

This branch is 1 commit ahead of master. [Pull request](#) [Compare](#)

 c3h899 Revised GitIgnore ... Latest commit 72b86d2 just now

[.gitignore](#) Revised GitIgnore just now

Help people interested in this repository understand your project by adding a README. [Add a README](#)

# Merging Branches (Pull Request) 2/3

1. Annotate the Request

2. Open the Request


The screenshot shows a GitHub Pull Request page for a repository named 'Revised GitIgnore'. The title of the pull request is 'Revised GitIgnore #1'. The pull request is created by user 'c3h899' and is intended to merge 1 commit into the 'master' branch from the 'Experimental' branch. The interface includes a navigation bar at the top with links for Code, Issues (0), Pull requests (1), Projects (0), Wiki, Insights, and Settings. Below the navigation bar, there is a green 'Open' button. A comment by 'c3h899' is visible, stating 'Ignore Stuff'. The pull request is marked as 'Verified' and has a commit hash of '72b86d2'. Two red arrows are overlaid on the image: one points from the 'Open' button to the '2. Open the Request' text, and the other points from the comment box to the '1. Annotate the Request' text.

<> Code    ⓘ Issues 0    **🔗 Pull requests 1**    📁 Projects 0    📖 Wiki    📊 Insights    ⚙️ Settings


## Revised GitIgnore #1

**🔗 Open** c3h899 wants to merge 1 commit into master from Experimental

💬 Conversation 0    🔗 Commits 1    📋 Checks 0    📄 Files changed 1

 c3h899 commented just now    Owner    +😊    ...

Ignore Stuff

🔗  Revised GitIgnore    ...    Verified    72b86d2

# Merging Branches (Pull Request) 3/3

- Merge

The screenshot shows a GitHub Pull Request interface. At the top, there are tabs for Code, Issues (0), Pull requests (1), Projects (0), Wiki, Insights, and Settings. The main title is 'Revised GitIgnore #1'. Below the title, it says 'c3h899 wants to merge 1 commit into master from Experimental'. There are buttons for 'Open', 'Conversation' (0), 'Commits' (1), 'Checks' (0), and 'Files changed' (1). A comment from 'c3h899' is visible, stating 'Ignore Stuff'. Below the comment, there is a commit 'Revised GitIgnore' with a 'Verified' badge and commit hash '72b86d2'. At the bottom, there is a green box containing two messages: 'Continuous integration has not been set up' and 'This branch has no conflicts with the base branch'. A red arrow points from the 'Merge' bullet point to the 'Merge pull request' button in the green box.

<> Code Issues 0 Pull requests 1 Projects 0 Wiki Insights Settings

## Revised GitIgnore #1

Open c3h899 wants to merge 1 commit into master from Experimental

Conversation 0 Commits 1 Checks 0 Files changed 1

c3h899 commented just now Owner + 😊 ...

Ignore Stuff

Revised GitIgnore ... Verified 72b86d2

Add more commits by pushing to the **Experimental** branch on c3h899/Demo2.

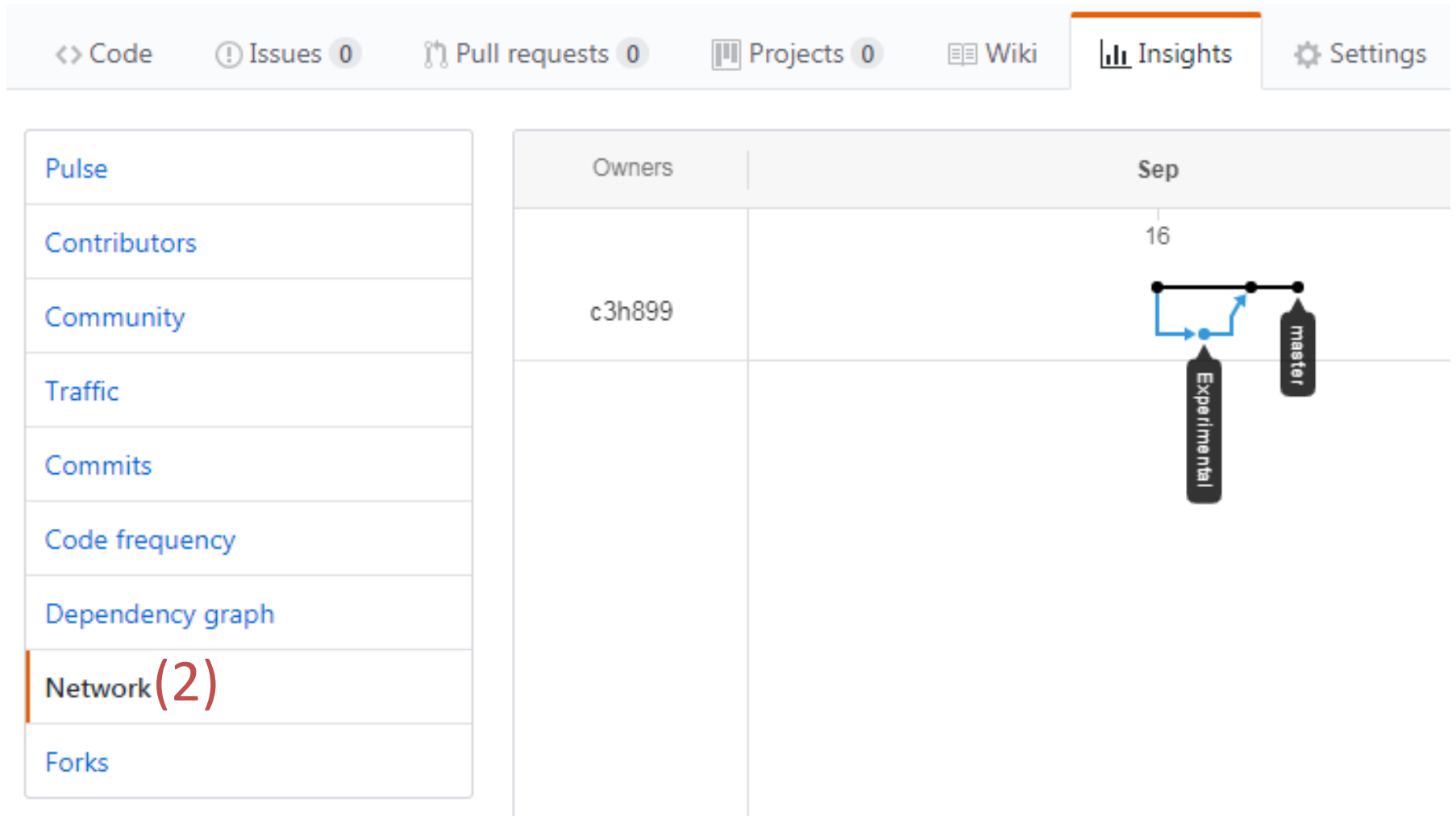
Continuous integration has not been set up  
Several apps are available to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch  
Merging can be performed automatically.

Merge pull request ▼ You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# Neat Graphs

(1)



# Try GitHub's Desktop Client

- <https://desktop.github.com/>

The screenshot displays the GitHub Desktop application window. The top bar includes a menu (File, Edit, View, Repository, Branch, Help) and status information: 'Current repository: desktop', 'Current branch: esc-pr' (with commit #3972), and 'Fetch origin' (last fetched 2 minutes ago). The main area is split into two panes. The left pane, titled 'History', shows a list of commits with their authors and commit messages. The right pane shows a diff for the commit 'Add event handler to dropdown component' by iAmWillShepherd and Markus Olsson. The diff highlights changes in the file 'app\src\ui\toolbar\dropdown.tsx', showing additions and deletions in the 'ToolbarDropdown' class.

Current repository: desktop | Current branch: esc-pr #3972 | Fetch origin (Last fetched 2 minutes ago)

Changes | History

**Add event handler to dropdown component**  
iAmWillShepherd and Markus Olsson committed c79e71c 1 changed file

Co-Authored-By: Markus Olsson <niiik@users.noreply.github.com>

app\src\ui\toolbar\dropdown.tsx

```
@@ -145,6 +145,10 @@ export class ToolbarDropdown extends React.Component<
  this.state = { clientRect: null }
}
+ private get isOpen() {
+   return this.props.dropdownState === 'open'
+ }
+
  private dropdownIcon(state: DropdownState): OcticonSymbol {
    // @TODO: Remake triangle octicon in a 12px version,
    // right now it's scaled badly on normal dpi monitors.
  }
+ private onFoldoutKeyDown = (event: React.KeyboardEvent<HTMLElement>) => {
+   if (!event.defaultPrevented && this.isOpen && event.key === 'Escape') {
+     event.preventDefault()
  }
```

# Additional References

- GIT SCM (Official Site)
  - <https://git-scm.com/>
- git - the simple guide
  - <https://rogerdudler.github.io/git-guide/>
- GitHub GIT CHEAT SHEET
  - <https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf>

# Citations

- Figures from the CASE section: ProGit 2<sup>nd</sup> Edition (2014) by Scott Chacon and Ben Straub released under a CC NC SA license.
- GitHub Diff from GitHub.com