Writing Commits for You, Your Friends, and Your Future Self

Victoria Dye

OPEN SOURCE 101

MARCH • 29 • 2022

Who am I?

Name - Victoria Dye (@vdye)

Occupation - Software Developer

Company - GitHub

Where I contribute - Git

I. Context
II. Writing Good Commits
III. Performing Commit-by-Commit Reviews
IV. Utilizing the Commit History

What is a commit (and why should I care)?

[Commits] are snapshots of your entire repository at specific times…based around logical units of change.

Over time, commits should tell a story of the history of your repository and how it came to be the way that it currently is.[1]

[1] https://github.com/git-guides/git-commit (2022 Mar 10)

Number of commits in `git`[1] - 66,016

Lines of text (code, documentation) in the repository - 1,412,339

Word count of non-merge commit messages - 3,292,050

Word count of *War and Peace* (English translation)[2] - 562,493

[1] As of 715d08a9e5 (The eighth batch, 2022-02-25)
[2] https://www.gutenberg.org/files/2600/2600-h/2600-h.htm (last accessed: Mar 18, 2022)

What do you mean by “you, your friends, and your future self”?

OPEN SOURCE 101

You

…you (in the present)

Your Friends

reviewers,
co-contributors

Your Future Self

someone reading your code in the future

Writing good commits for…

You

➢ “This is a huge project, where do I start?”

Your friends

➢ “How do I review this?”

Your future self

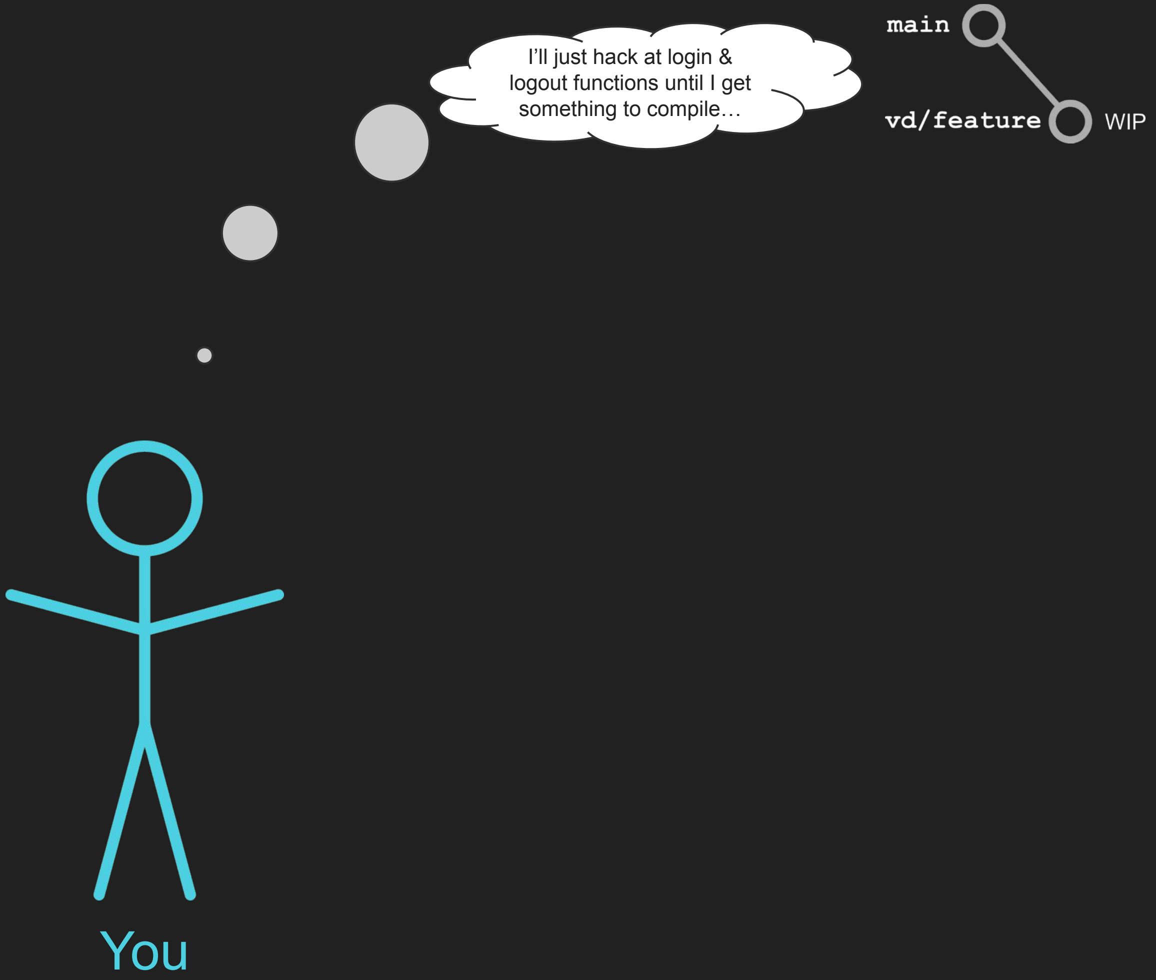
➢ “What was this code supposed to do?”

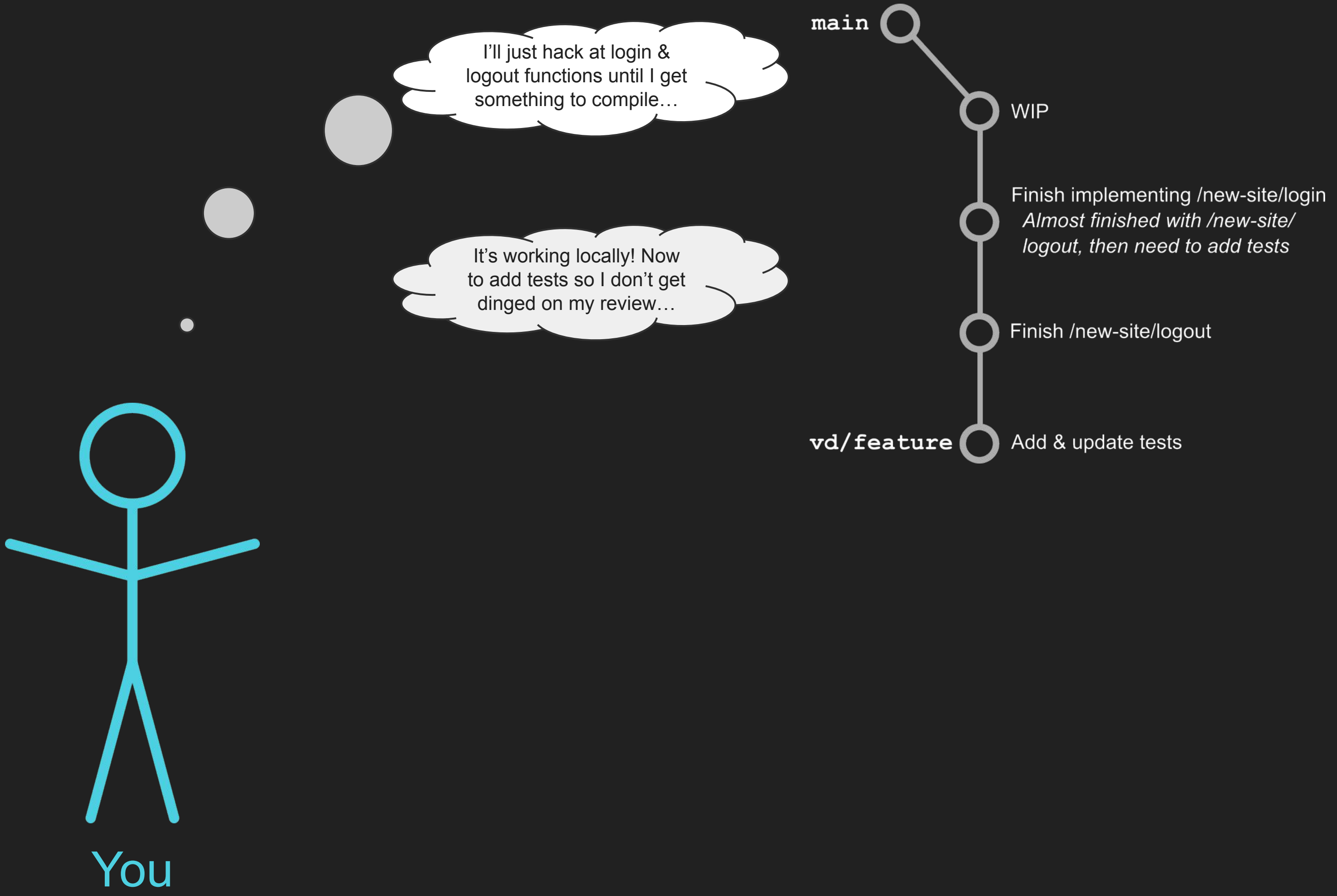
Guidelines for writing good commits

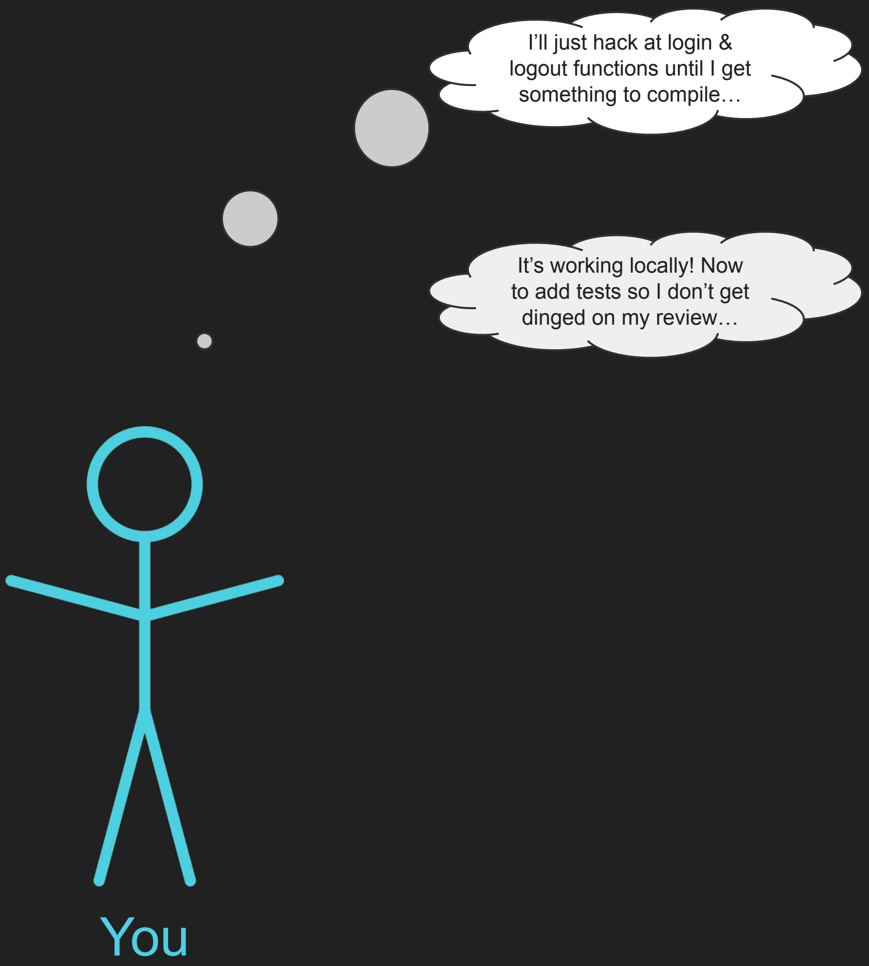
1. Outline your changes as a narrative structure
2. Break your changes into small, atomic increments
3. Use the commit message to explain “what” and “why”

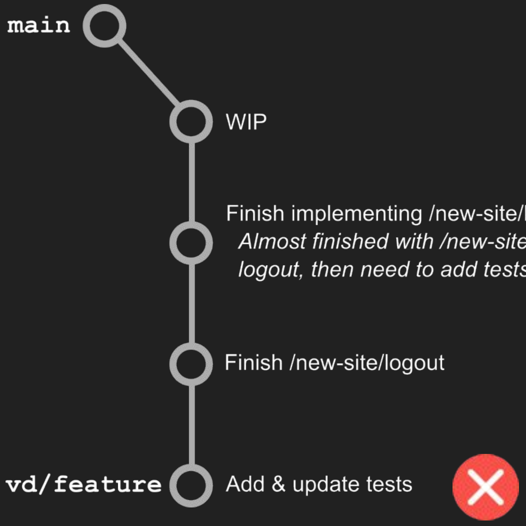
Guidelines for writing good commits

1. **Outline your changes as a narrative structure**
2. Break your changes into small, atomic increments
3. Use the commit message to explain “what” and “why”

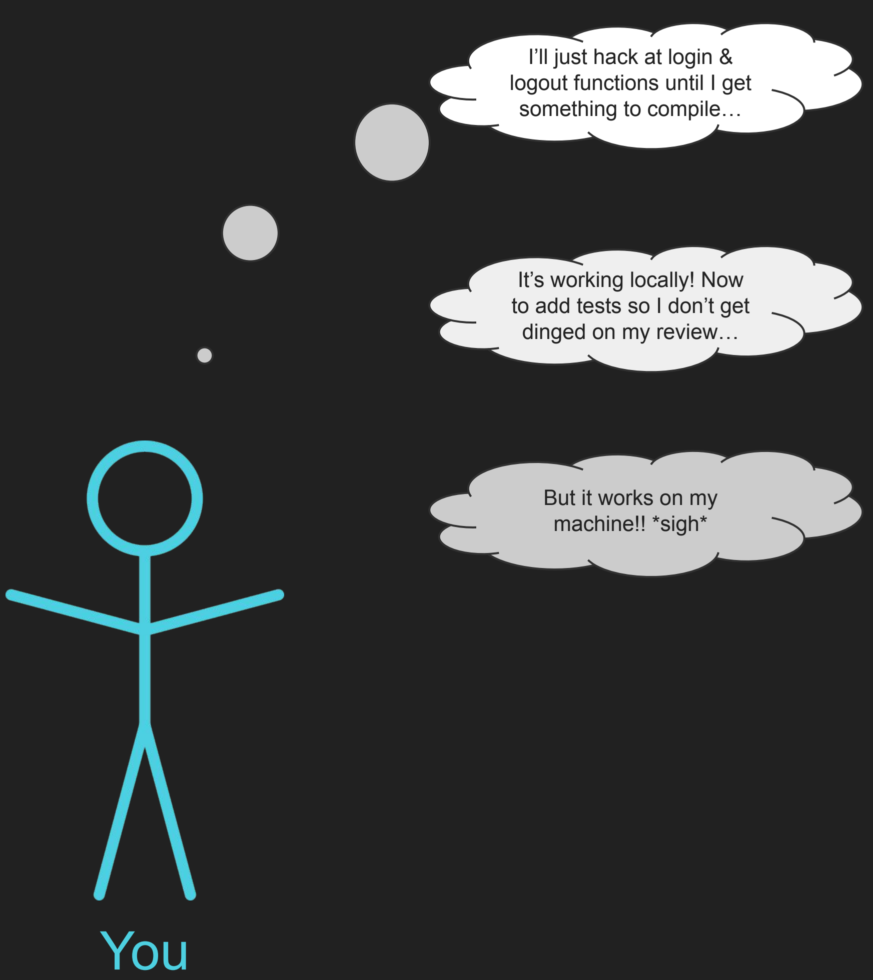

I'll just hack at login & logout functions until I get something to compile…
main
vd/feature
WIP
You

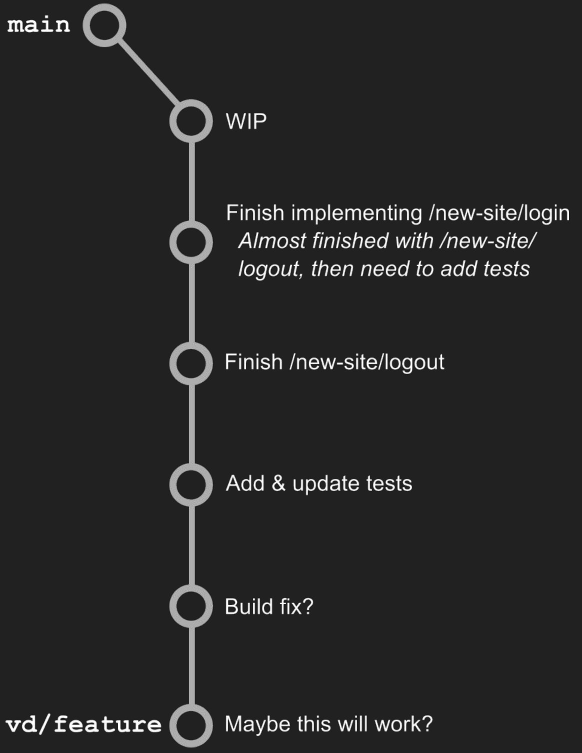

I'll just hack at login & logout functions until I get something to compile…
It's working locally! Now to add tests so I don't get dinged on my review…
You
main
WIP
Finish implementing /new-site/login
Almost finished with /new-site/ logout, then need to add tests
Finish /new-site/logout
vd/feature
Add & update tests


I'll just hack at login & logout functions until I get something to compile…
It's working locally! Now to add tests so I don't get dinged on my review…
You


main
WIP
Finish implementing /new-site/login
Almost finished with /new-site/ logout, then need to add tests
Finish /new-site/logout
vd/feature
Add & update tests

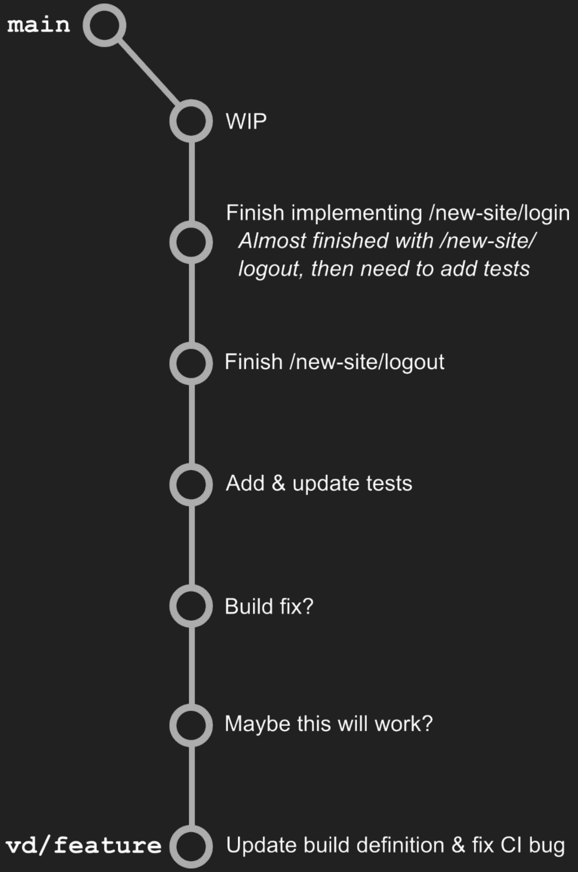
OPEN SOURCE 101


I'll just hack at login & logout functions until I get something to compile…
It's working locally! Now to add tests so I don't get dinged on my review…
But it works on my machine!! *sigh*
You


main
WIP
Finish implementing /new-site/login
Almost finished with /new-site/ logout, then need to add tests
Finish /new-site/logout
Add & update tests
Build fix?
vd/feature
Maybe this will work?

OPEN SOURCE 101

I'll just hack at login & logout functions until I get something to compile…
It's working locally! Now to add tests so I don't get dinged on my review…
But it works on my machine!! *sigh*
Why didn't I see that before! That should fix it.
You



WIP

Finish implementing /new-site/login
Almost finished with /new-site/ logout, then need to add tests

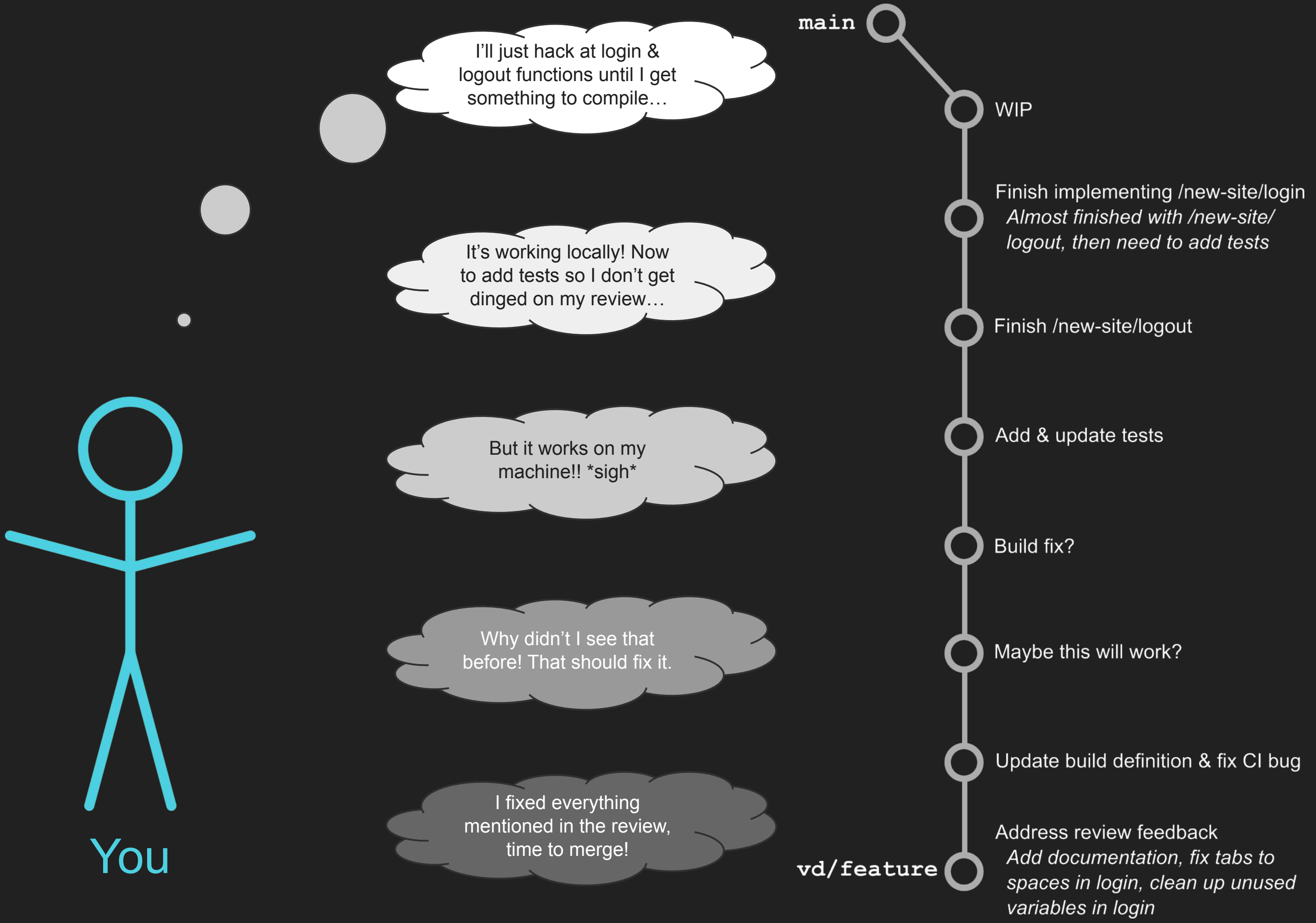
Finish /new-site/logout

Add & update tests

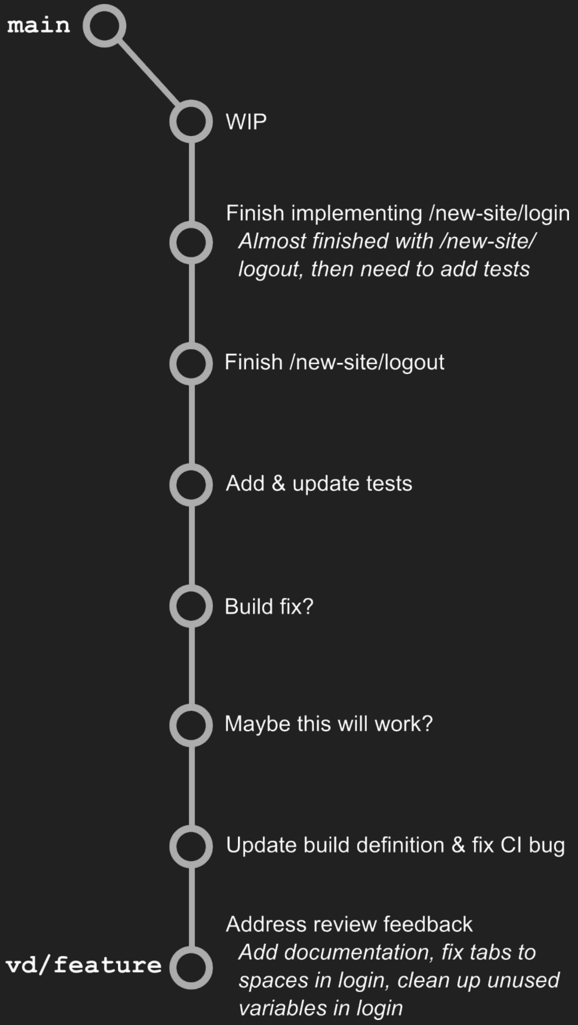
Build fix?

Maybe this will work?

Update build definition & fix CI bug


I'll just hack at login & logout functions until I get something to compile…
It's working locally! Now to add tests so I don't get dinged on my review…
But it works on my machine!! *sigh*
Why didn't I see that before! That should fix it.
I fixed everything mentioned in the review, time to merge!
You
main
WIP
Finish implementing /new-site/login *Almost finished with /new-site/ logout, then need to add tests*
Finish /new-site/logout
Add & update tests
Build fix?
Maybe this will work?
Update build definition & fix CI bug
Address review feedback *Add documentation, fix tabs to spaces in login, clean up unused variables in login*
vd/feature

Half of the functions called here aren't implemented. Maybe they show up later?
Your Reviewer



WIP

Finish implementing /new-site/login
Almost finished with /new-site/ logout, then need to add tests

Finish /new-site/logout

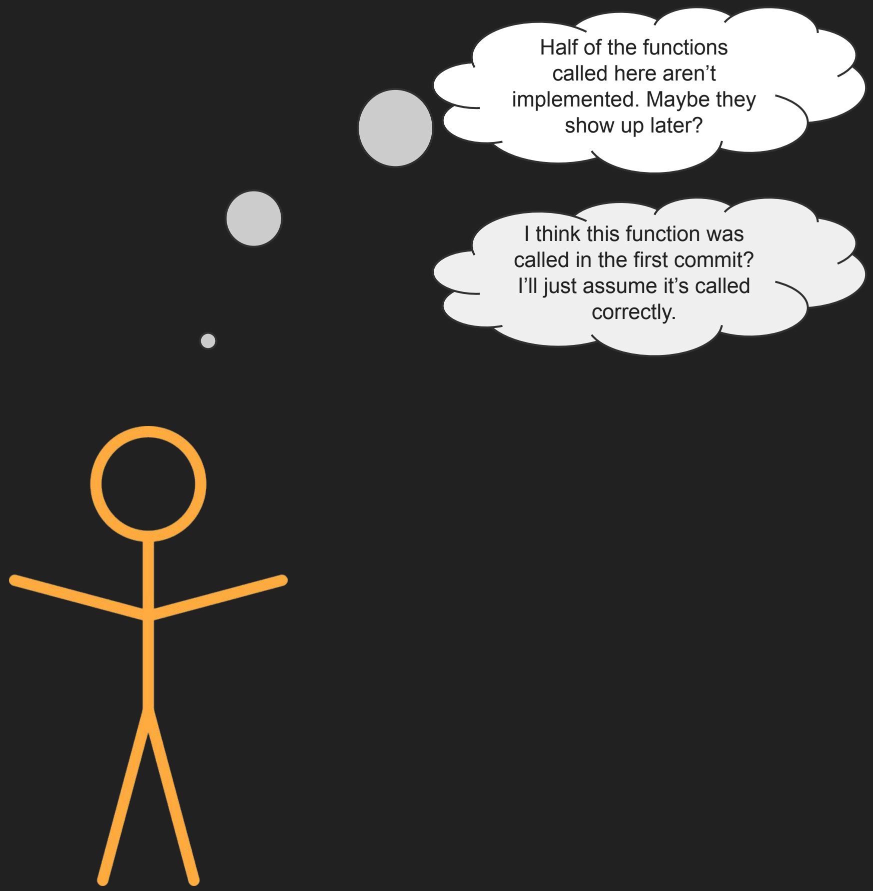
Add & update tests

Build fix?

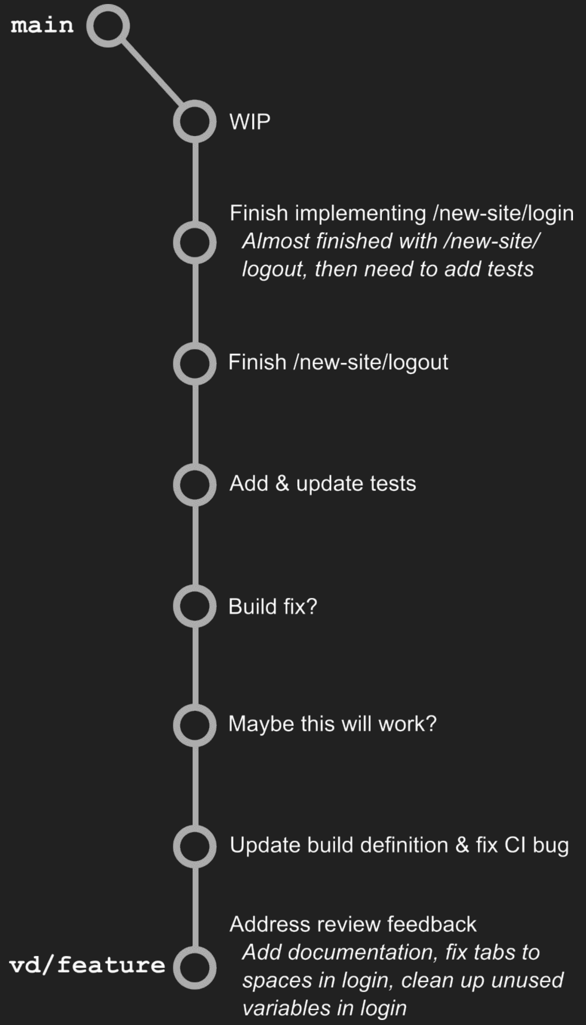
Maybe this will work?

Update build definition & fix CI bug

Address review feedback
Add documentation, fix tabs to spaces in login, clean up unused variables in login


Half of the functions called here aren't implemented. Maybe they show up later?
I think this function was called in the first commit? I'll just assume it's called correctly.

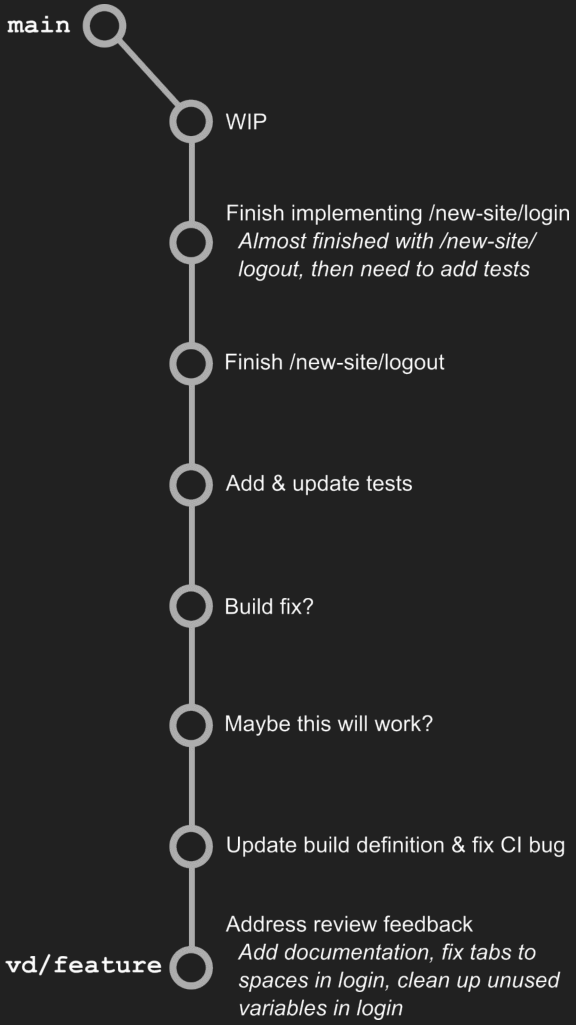
Your Reviewer


main
WIP
Finish implementing /new-site/login
Almost finished with /new-site/ logout, then need to add tests
Finish /new-site/logout
Add & update tests
Build fix?
Maybe this will work?
Update build definition & fix CI bug
Address review feedback
Add documentation, fix tabs to spaces in login, clean up unused variables in login
vd/feature

OPEN SOURCE 101

Half of the functions called here aren’t implemented. Maybe they show up later?
I think this function was called in the first commit? I’ll just assume it’s called correctly.
This is a *ton* of tests, but I guess that makes sense given login/logout is such a core functionality.

Your Reviewer



WIP

Finish implementing /new-site/login
Almost finished with /new-site/logout, then need to add tests

Finish /new-site/logout

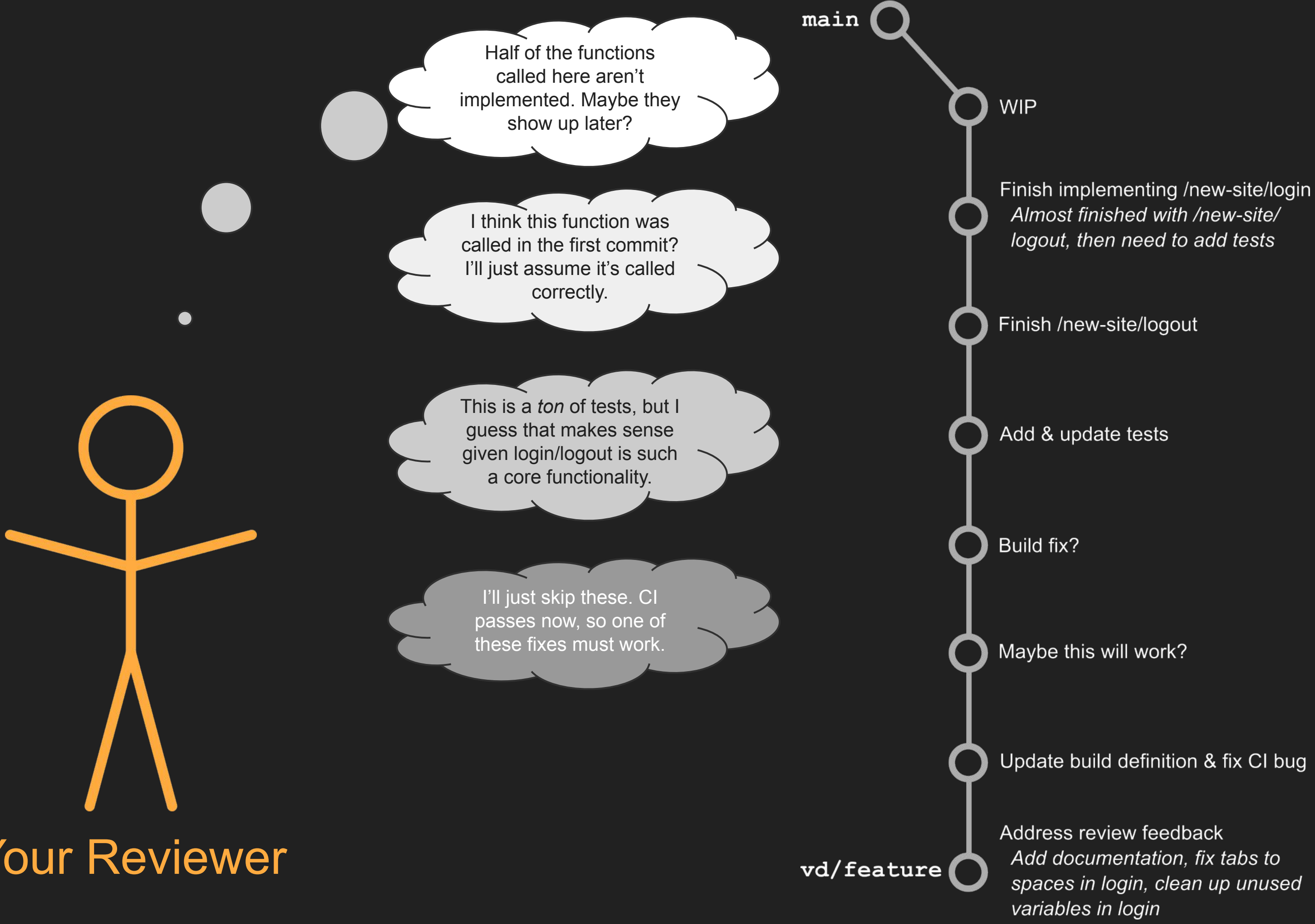
Add & update tests

Build fix?

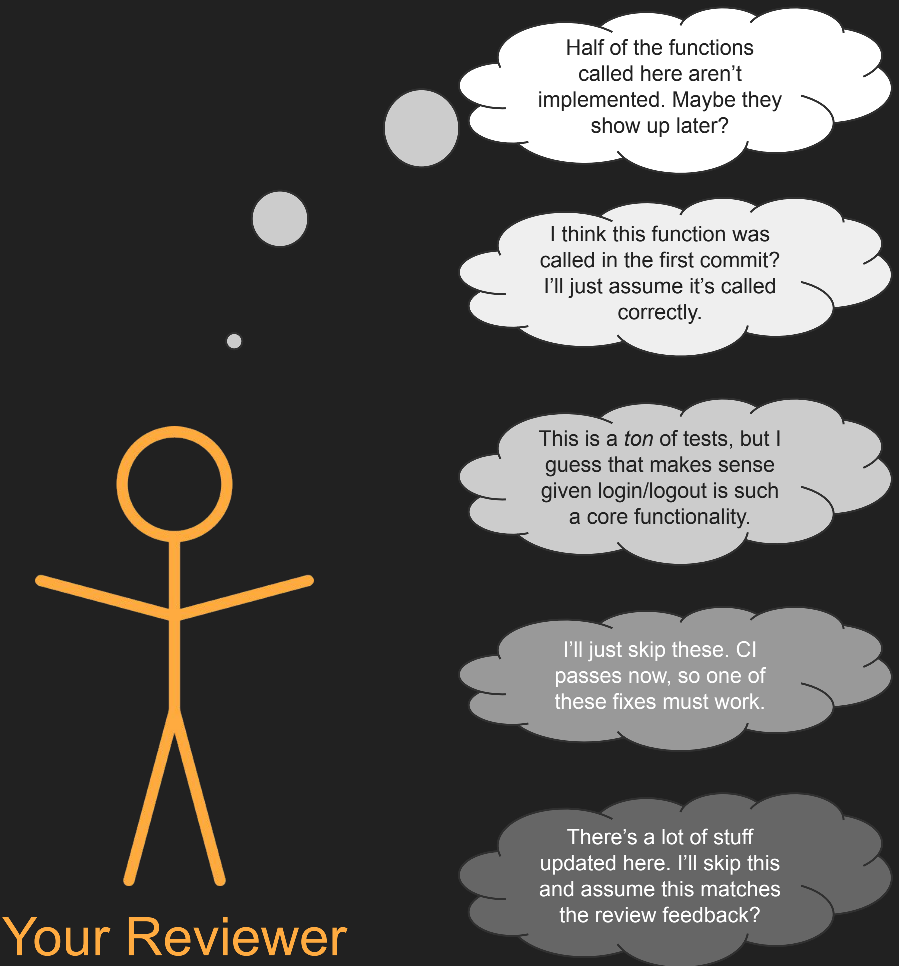
Maybe this will work?

Update build definition & fix CI bug

Address review feedback
Add documentation, fix tabs to spaces in login, clean up unused variables in login


Half of the functions called here aren't implemented. Maybe they show up later?
I think this function was called in the first commit? I'll just assume it's called correctly.
This is a *ton* of tests, but I guess that makes sense given login/logout is such a core functionality.
I'll just skip these. CI passes now, so one of these fixes must work.
Your Reviewer
main
WIP
Finish implementing /new-site/login *Almost finished with /new-site/ logout, then need to add tests*
Finish /new-site/logout
Add & update tests
Build fix?
Maybe this will work?
Update build definition & fix CI bug
Address review feedback *Add documentation, fix tabs to spaces in login, clean up unused variables in login*
vd/feature

OPEN SOURCE 101



Half of the functions called here aren't implemented. Maybe they show up later?

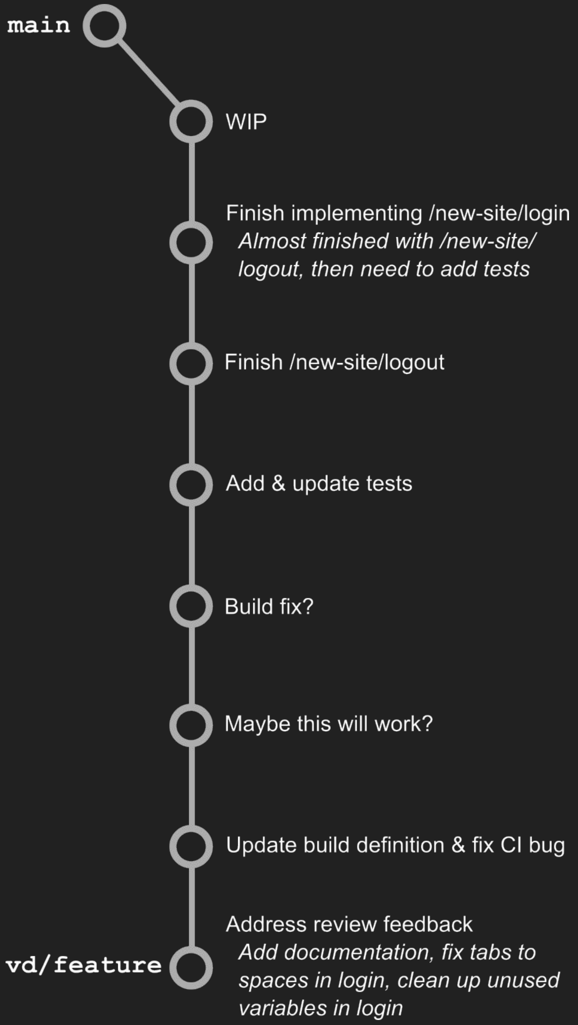
I think this function was called in the first commit? I'll just assume it's called correctly.

This is a *ton* of tests, but I guess that makes sense given login/logout is such a core functionality.

I'll just skip these. CI passes now, so one of these fixes must work.

There's a lot of stuff updated here. I'll skip this and assume this matches the review feedback?

Your Reviewer



`main`

WIP

Finish implementing /new-site/login
Almost finished with /new-site/logout, then need to add tests

Finish /new-site/logout

Add & update tests

Build fix?

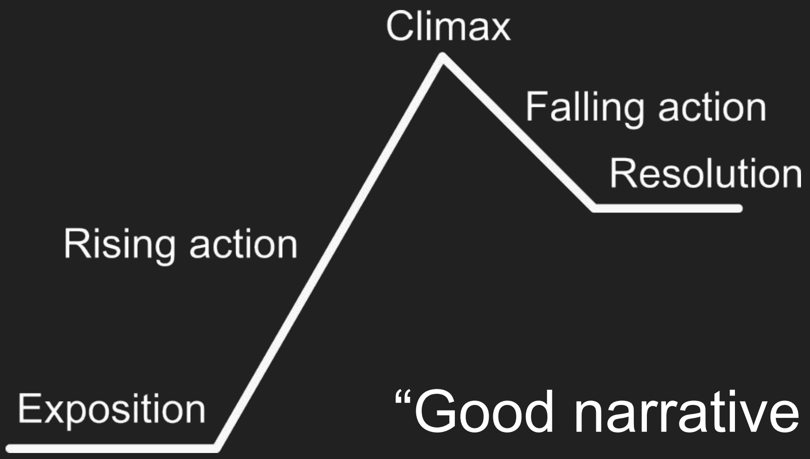
Maybe this will work?

Update build definition & fix CI bug

Address review feedback
Add documentation, fix tabs to spaces in login, clean up unused variables in login

`vd/feature`

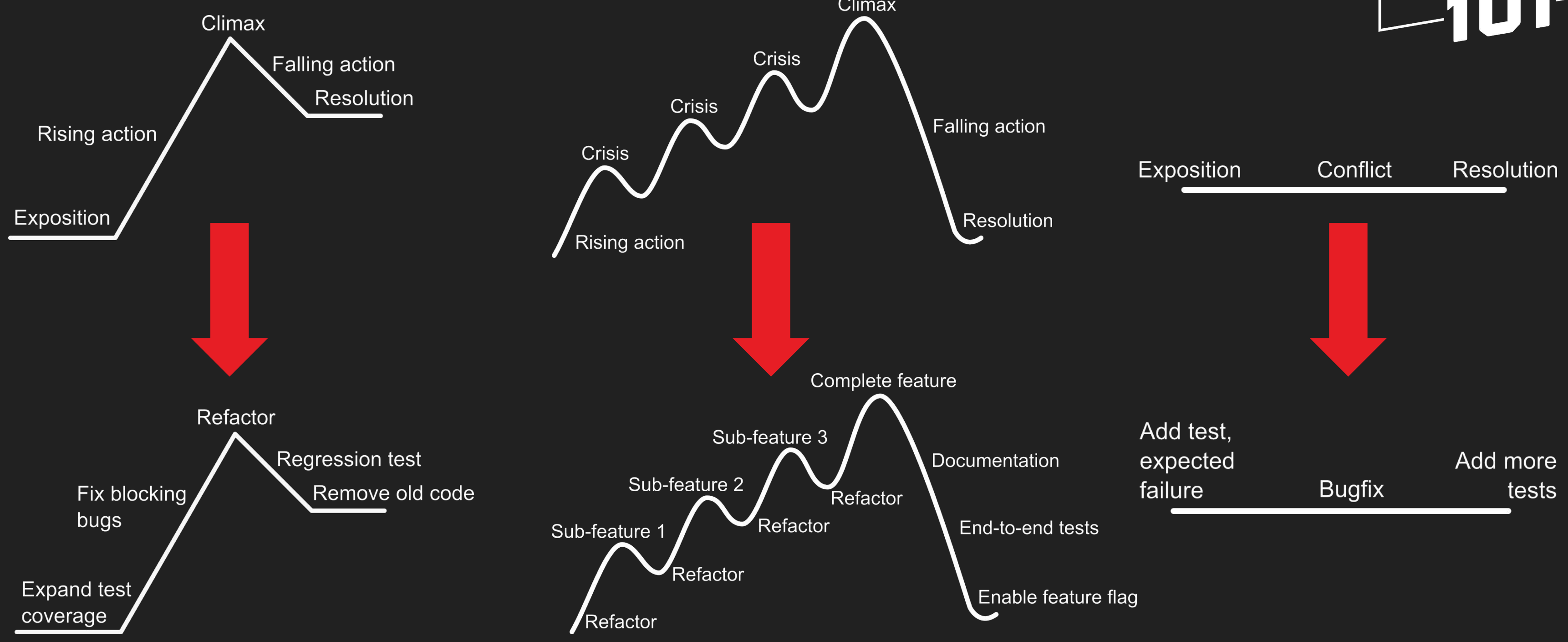
Narrative structure



"Good narrative structure is about presenting the plot and story elements to allow readers to understand what is happening *and* what it all means."[1]

[1] https://blog.reedsy.com/guide/story-structure/ (last accessed: Mar 14, 2022)

Narrative structure



No one-size-fits-all

DO

Create an outline, include it in the PR description or “cover letter”

Stay on-topic

DON’T

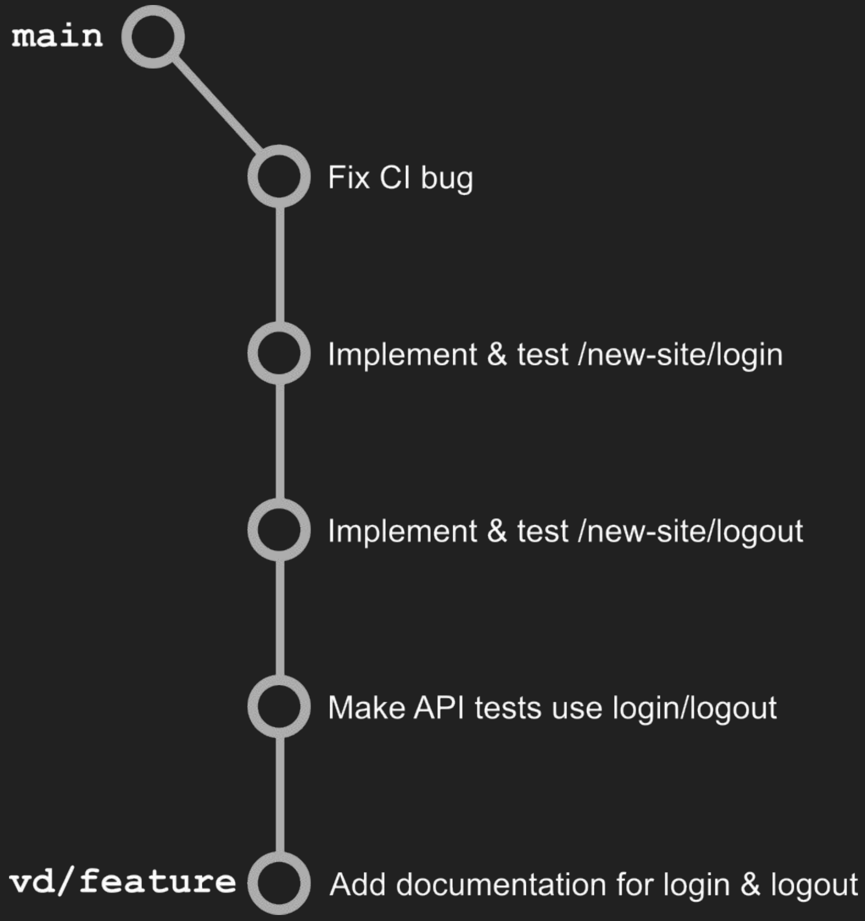
Put partial or independent changes together in a commit

“Correct” a commit in a later commit

Most importantly, tell *your* story

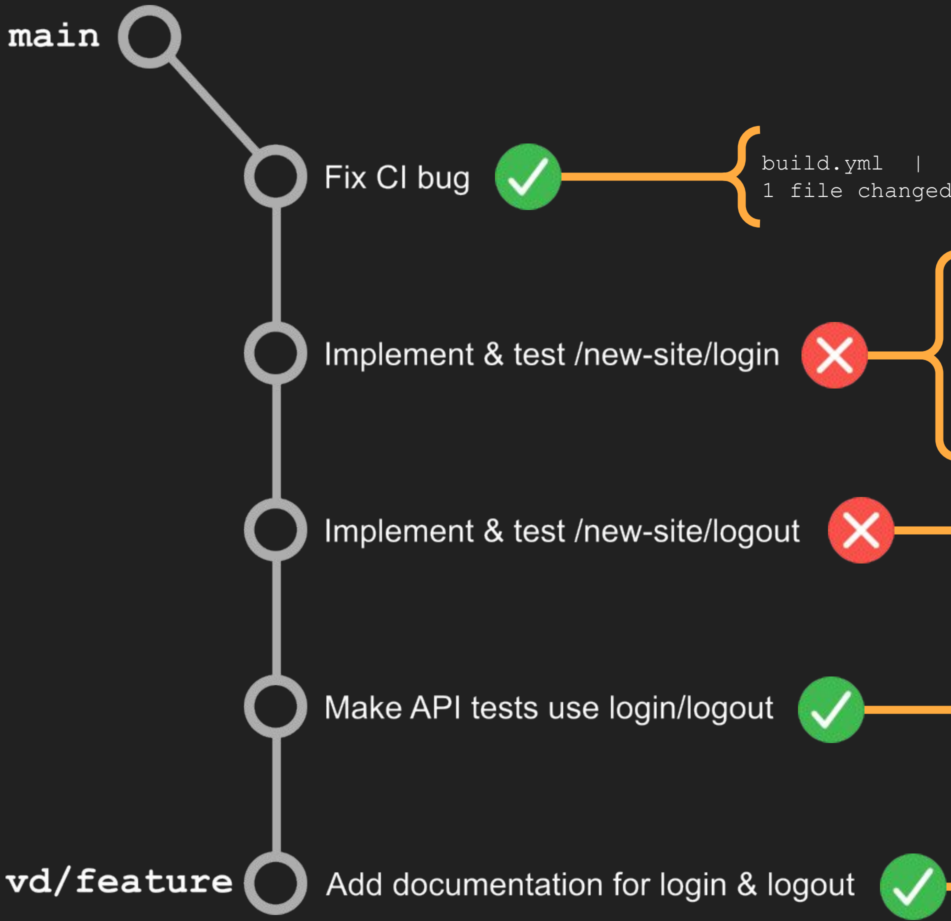
Guidelines for writing good commits

1. Outline your changes as a narrative structure
2. **Break your changes into small, atomic increments**
3. Use the commit message to explain “what” and “why”


main
Fix CI bug
Implement & test /new-site/login
Implement & test /new-site/logout
Make API tests use login/logout
vd/feature
Add documentation for login & logout

Outline

1. Setup
   a. Fix CI bug
2. Implement feature
   a. Implement login
   b. Implement logout
3. Test
4. Document


main
Fix CI bug
Implement & test /new-site/login
Implement & test /new-site/logout
Make API tests use login/logout
vd/feature
Add documentation for login & logout

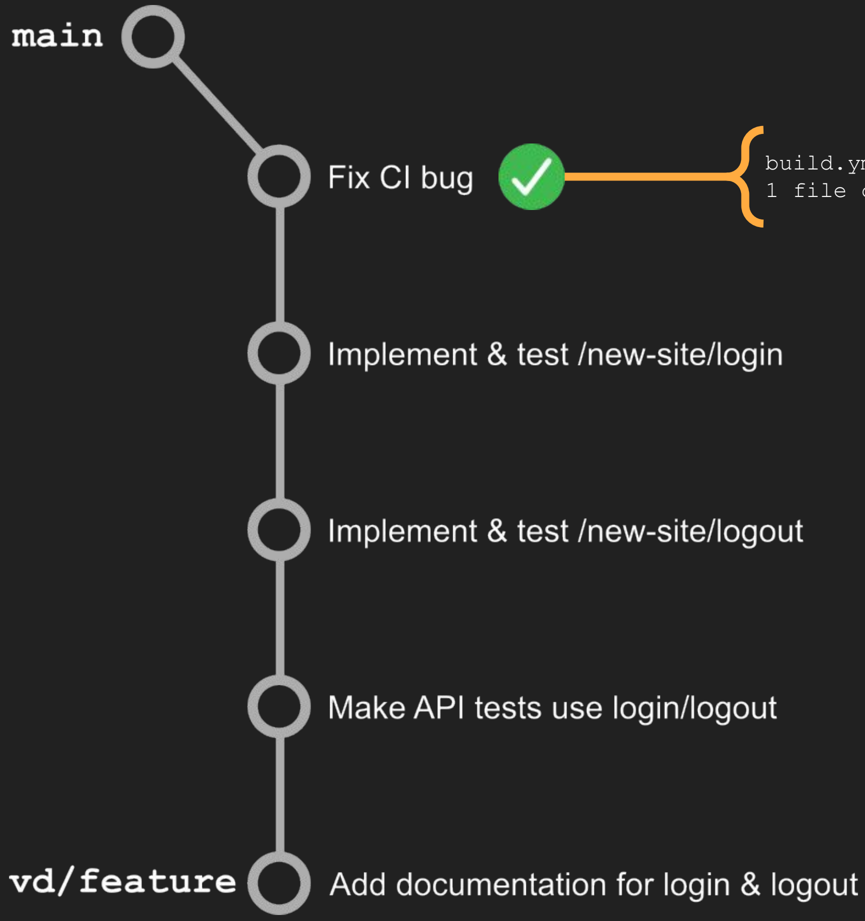
```
build.yml   |  5 +++--
1 file changed, 3 insertions(+), 2 deletions(-)
```

```
src/common.code     |  81  ++--
src/login.code      |  574++++++++++++++++++++++++++++
src/users.code      |  126++++++
test/login.test     |  362++++++++++++++++++
new-site.build      |  22  +-
5 files changed, 1146 insertions(+), 19 deletions(-)
```

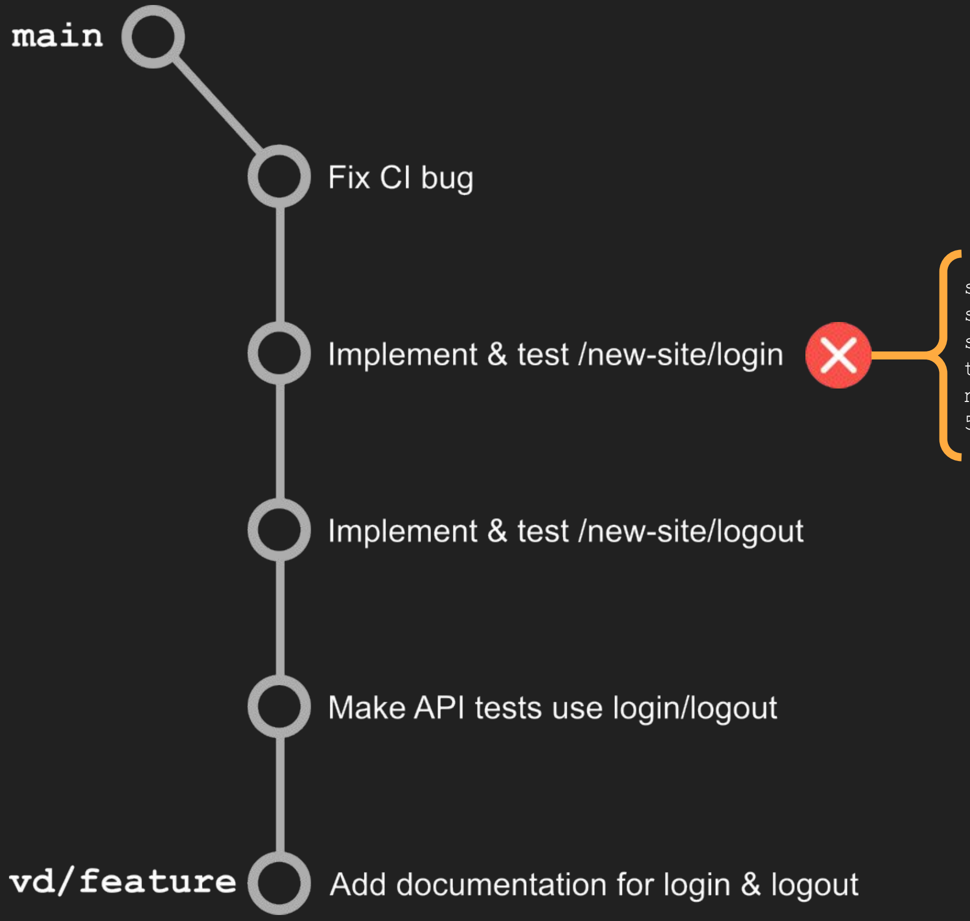
```
src/common.code    |  135++++++--
src/logout.code    |  126++++++
test/logout.test   |  362++++++++++++++++++
3 files changed, 609 insertions(+), 14 deletions(-)
```

```
test/common.code      |  57+++++++++++
test/pictures.test    |  39+++++++-
test/polls.test       |  28++++--
test/text-posts.test  |  45++++++---
test/videos.test      |  17++-
5 files changed, 164 insertions(+), 22 deletions(-)
```

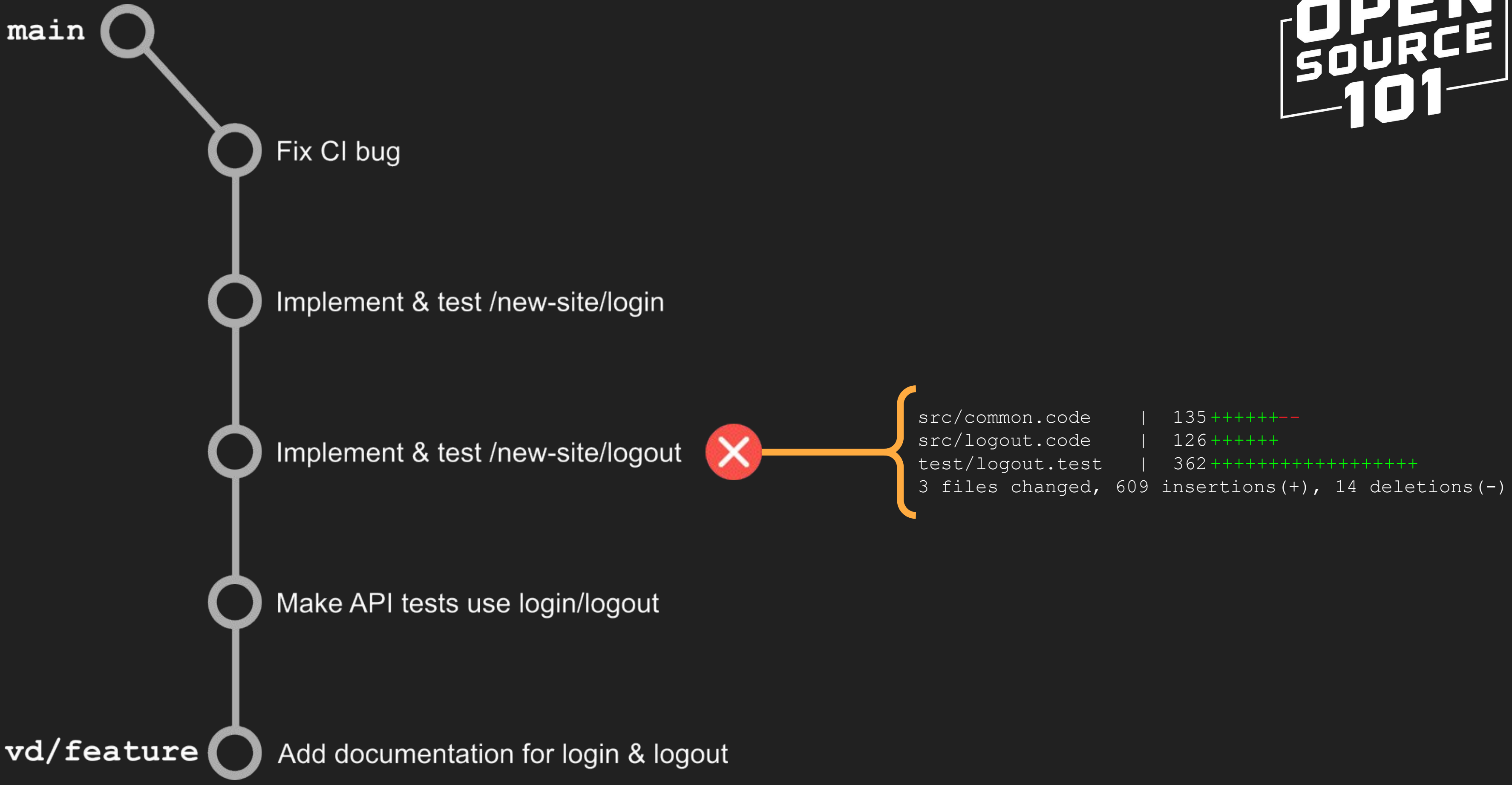
```
docs/login-logout.md  |  229++++++++++++
1 file changed, 229 insertions(+)
```


main
Fix CI bug
Implement & test /new-site/login
Implement & test /new-site/logout
Make API tests use login/logout
vd/feature
Add documentation for login & logout

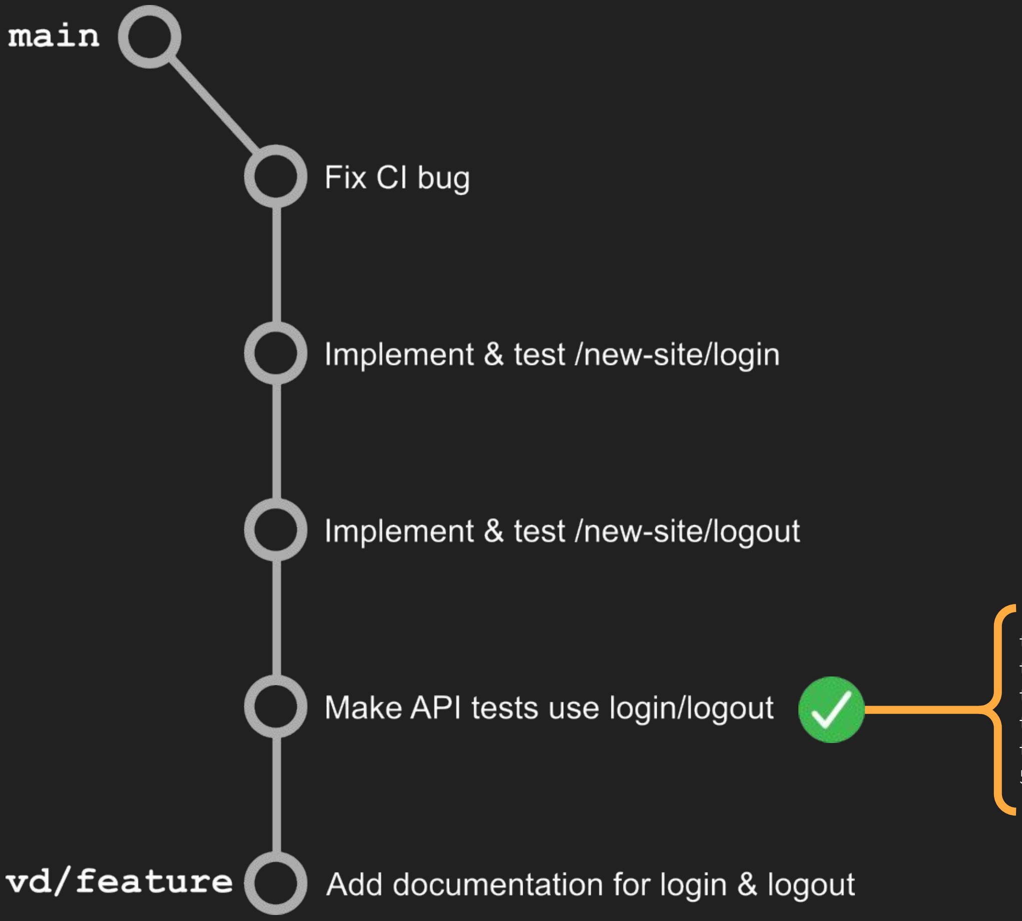
```
build.yml   |   5 +++--
1 file changed, 3 insertions(+), 2 deletions(-)
```


main
Fix CI bug
Implement & test /new-site/login
Implement & test /new-site/logout
Make API tests use login/logout
vd/feature
Add documentation for login & logout

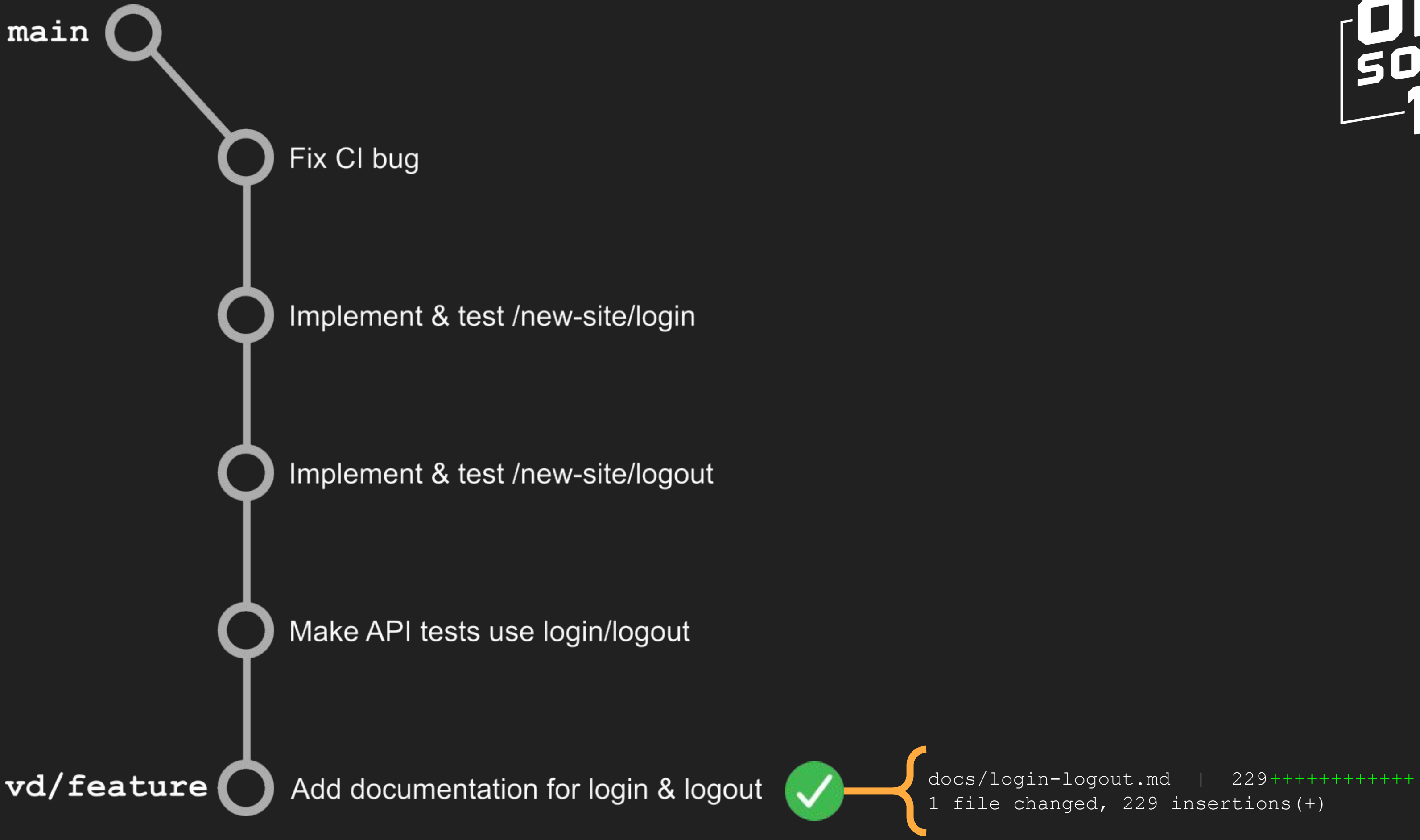
```
src/common.code      |  81 ++--
src/login.code       |  574++++++++++++++++++++++++++++
src/users.code       |  126++++++
test/login.test      |  362++++++++++++++++++
new-site.build       |  22 +-
5 files changed, 1146 insertions(+), 19 deletions(-)
```


main
Fix CI bug
Implement & test /new-site/login
Implement & test /new-site/logout
Make API tests use login/logout
vd/feature
Add documentation for login & logout

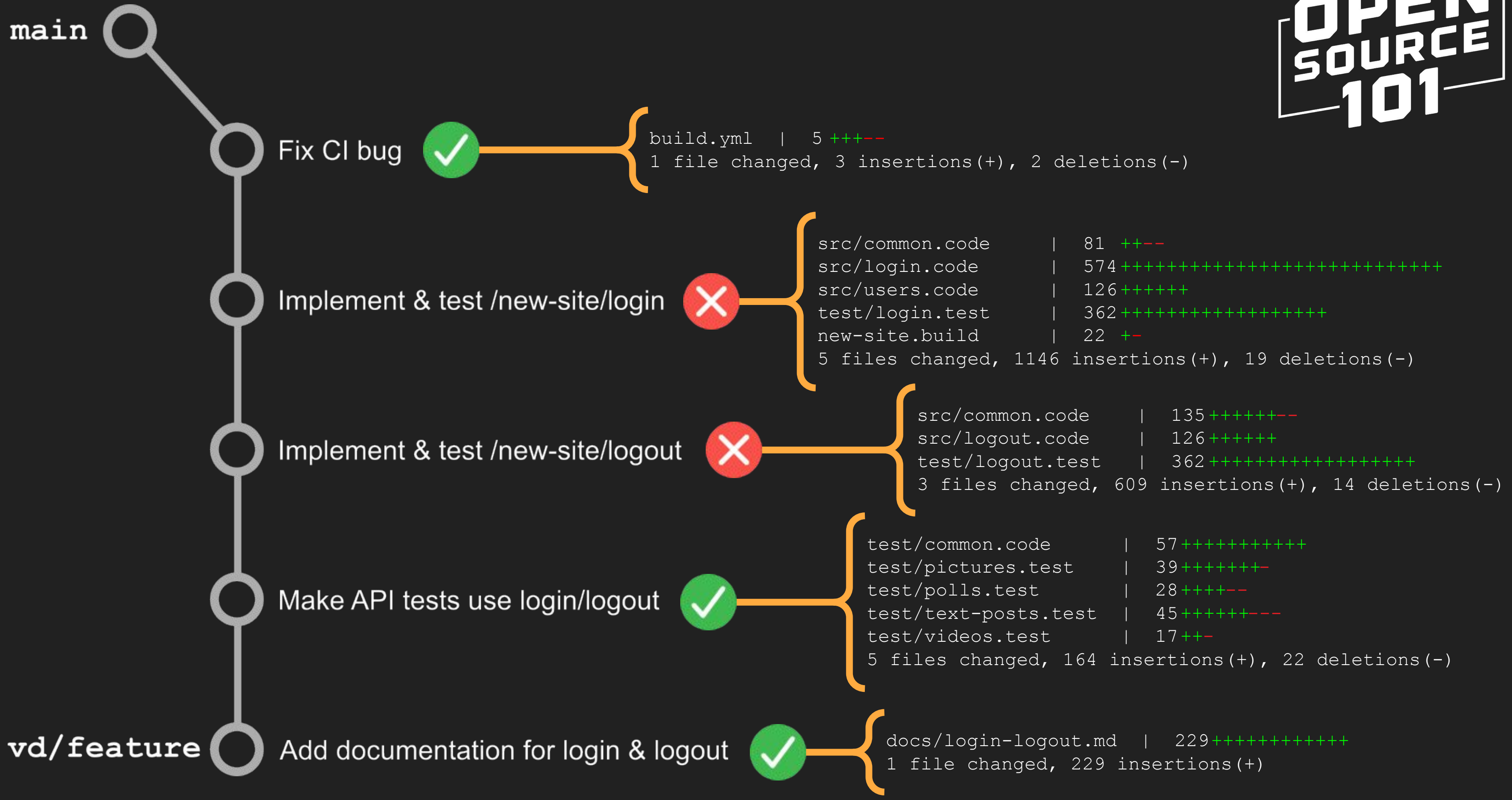
```
src/common.code     |  135 ++++++--
src/logout.code     |  126 ++++++
test/logout.test    |  362 ++++++++++++++++++
3 files changed, 609 insertions(+), 14 deletions(-)
```


main
Fix CI bug
Implement & test /new-site/login
Implement & test /new-site/logout
Make API tests use login/logout
vd/feature
Add documentation for login & logout

```
test/common.code        |  57++++++++++++
test/pictures.test      |  39++++++++-
test/polls.test         |  28++++--
test/text-posts.test    |  45++++++---
test/videos.test        |  17++-
5 files changed, 164 insertions(+), 22 deletions(-)
```


main
Fix CI bug
Implement & test /new-site/login
Implement & test /new-site/logout
Make API tests use login/logout
vd/feature
Add documentation for login & logout

```
docs/login-logout.md   |  229++++++++++++
1 file changed, 229 insertions(+)
```


main
Fix CI bug
Implement & test /new-site/login
Implement & test /new-site/logout
Make API tests use login/logout
vd/feature
Add documentation for login & logout

```
build.yml  |  5 +++--
1 file changed, 3 insertions(+), 2 deletions(-)
```

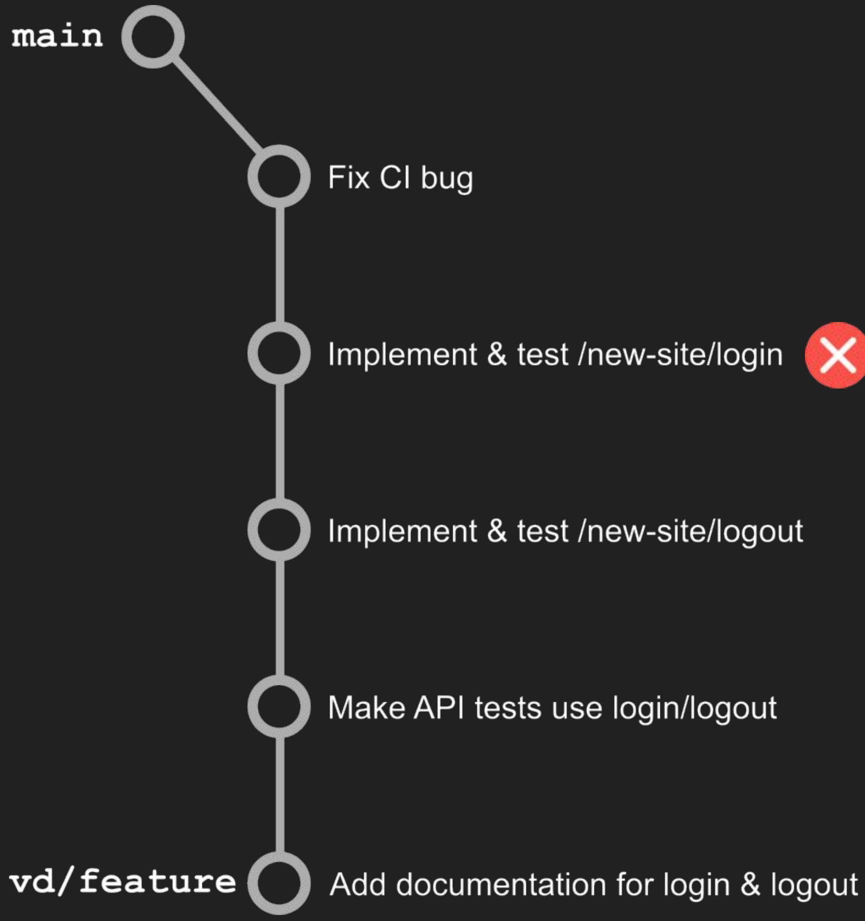
```
src/common.code     |  81  ++--
src/login.code      |  574++++++++++++++++++++++++++++
src/users.code      |  126++++++
test/login.test     |  362++++++++++++++++++
new-site.build      |  22  +-
5 files changed, 1146 insertions(+), 19 deletions(-)
```

```
src/common.code    |  135++++++--
src/logout.code    |  126++++++
test/logout.test   |  362++++++++++++++++++
3 files changed, 609 insertions(+), 14 deletions(-)
```

```
test/common.code       |  57+++++++++++
test/pictures.test     |  39+++++++-
test/polls.test        |  28++++--
test/text-posts.test   |  45++++++---
test/videos.test       |  17++-
5 files changed, 164 insertions(+), 22 deletions(-)
```

```
docs/login-logout.md  |  229++++++++++++
1 file changed, 229 insertions(+)
```

OPEN SOURCE 101

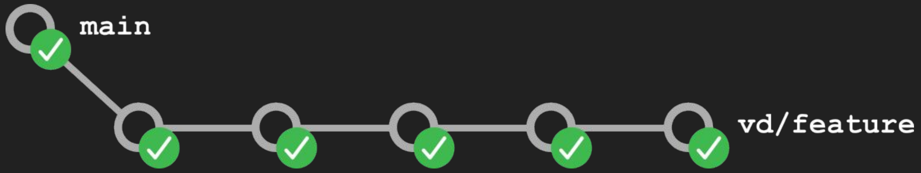

main
Fix CI bug
Implement & test /new-site/login
Implement & test /new-site/logout
Make API tests use login/logout
vd/feature
Add documentation for login & logout

```
src/common.code      |  81  ++--
src/login.code       |  574 ++++++++++++++++++++++++++++
src/users.code       |  126 ++++++
test/login.test      |  362 ++++++++++++++++++
new-site.build       |  22  +-
5 files changed, 1146 insertions(+), 19 deletions(-)
```

Atomic

Every commit is an *independent unit*

The repo is *stable* after every commit



Small

Each commit *does one thing*

Change stack has *minimal depth*

Commit A

```
+ int get_user(UUID user_id)
+ {
+       ...
+ }
```

Commit B

```
+ int login(uuid)
+ {
+       ...
+       exists = get_user(uuid);
+       ...
+ }
```

Guidelines for writing good commits

1. Outline your changes as a narrative structure
2. Break your changes into small, atomic increments
3. **Use the commit message to explain “what” and “why”**

```
commit <SHA>
Author: Jeff Hostetler <jeffhost@microsoft.com>
Date:   Mon Oct 4 22:29:03 2021 +0000

    t/perf/perf-lib.sh: remove test_times.* at the end test_perf_()

    Teach test perf () to remove the temporary test_times.* files
    at the end of each test.

    test_perf_() runs a particular GIT_PERF_REPEAT_COUNT times and
creates
    ./test_times.[123...].  It then uses a perl script to find the
minimum
    over "./test times.*" (note the wildcard) and writes that time to
    "test-results/<testname>.<testnumber>.result".

    If the repeat count is changed during the pXXXX test script, stale
    test times.* files (from previous steps) may be included in the min()
    computation.  For example:

    ...
    GIT PERF REPEAT COUNT=3 \
    test_perf "status" "
               git status
    "

    GIT PERF REPEAT COUNT=1 \
    test_perf "checkout other" "
               git checkout other
    "
    ...

    The time reported in the summary for "XXXX.2 checkout other" would
    be "min( checkout[1], status[2], status[3] )".

    We prevent that error by removing the test_times.* files at the end
of
    each test.
```

```
commit <SHA>
Author: Victoria Dye <vdye@github.com>
Date:   Fri Dec 17 10:26:59 2021 -0500

    Make error text more helpful
```

What

High-level **intent** of the commit (*what* does this accomplish?)

Explanation of the **implementation** (*what* did you do to accomplish your goal?)

Why

Context for your implementation (*why* does the code do what it does now?)

Justification for the change (*why* is this change being made?)

```
commit <SHA>
Author: Jeff Hostetler <jeffhost@microsoft.com>
Date:   Mon Oct 4 22:29:03 2021 +0000

    t/perf/perf-lib.sh: remove test_times.* at the end test_perf_()

    Teach test perf () to remove the temporary test_times.* files
    at the end of each test.

    test_perf_() runs a particular GIT_PERF_REPEAT_COUNT times and
creates
    ./test_times.[123...].  It then uses a perl script to find the
minimum
    over "./test times.*" (note the wildcard) and writes that time to
    "test-results/<testname>.<testnumber>.result".

    If the repeat count is changed during the pXXXX test script, stale
    test times.* files (from previous steps) may be included in the min()
    computation.  For example:

    ...
    GIT PERF REPEAT COUNT=3 \
    test_perf "status" "
               git status
    "

    GIT PERF REPEAT COUNT=1 \
    test_perf "checkout other" "
               git checkout other
    "
    ...

    The time reported in the summary for "XXXX.2 checkout other" would
    be "min( checkout[1], status[2], status[3] )".

    We prevent that error by removing the test_times.* files at the end
of
    each test.
```

Implementation

Context

Justification

Intent

What

Most commits only (lightly) cover these

High-level **intent** of the commit (*what* does this accomplish?)

Explanation of the **implementation** (*what* did you do to accomplish your goal?)

Why

Justification for the change (*why* is this needed?)

Context for your implementation (*why* is it implemented this way?)

```
commit <SHA>
Author: Victoria Dye <vdye@github.com>
Date:   Fri Dec 17 10:26:59 2021 -0500

    Make error text more helpful
```

Intent

```
commit <SHA>
Author: Victoria Dye <vdye@github.com>
Date:   Fri Dec 17 10:26:59 2021 -0500

    Make error text more helpful
```

Intent

What it's actually doing

```
$ ./git-portable.sh invalid-command
Not a valid command: invalid-command

$ ./gitportable.sh
Not a valid command:
```

```
$ ./git-portable.sh invalid-command
Not a valid command: invalid-command

$ ./gitportable.sh
Please specify a command
```

```
commit <SHA>
Author: Victoria Dye <vdye@github.com>
Date:   Fri Dec 17 10:26:59 2021 -0500

    git-portable.sh: make error text more helpful

    When provided with incorrect argument, return a message more indicative of
    the cause of the error.

    If a user did not provide an argument to 'git-portable.sh', the error
    message returned would be:

    $ ./git-portable.sh
    Not a valid command:

    This does not clearly indicate that the problem is that 'git-portable.sh'
    must be called with a subcommand (e.g., ./git-portable.sh install).

    To guide the user towards the correct usage, instead print "Please specify a
    command" when no subcommand is specified.
```

Intent

Context

Justification

Implementation

```
commit <SHA>
Author: Victoria Dye <vdye@github.com>
Date:   Fri Dec 17 10:26:59 2021 -0500

    git-portable.sh: make error text more helpful

    The message “Not a valid command: <invalid command>” is
    intended to notify the user that their subcommand is invalid.
    However, when no subcommand is given, the "empty" subcommand
    results in the same message: "Not a valid command:". This does
    not clearly guide the user to the correct behavior, so print
    "Please specify a command" when no subcommand is specified.
```

Intent

Context

Justification

Implementation

Recap: guidelines for writing good commits

1. Outline your changes as a narrative structure

   → Takeway: guides you & your reviewer through changes

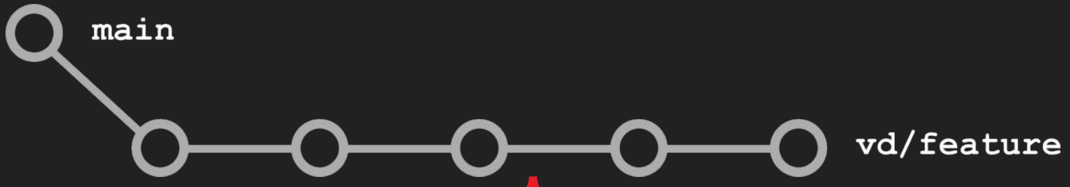
2. Break your changes into small, atomic increments

   → Takeaway: makes review as efficient as possible

3. Use the commit message to explain “what” and “why”

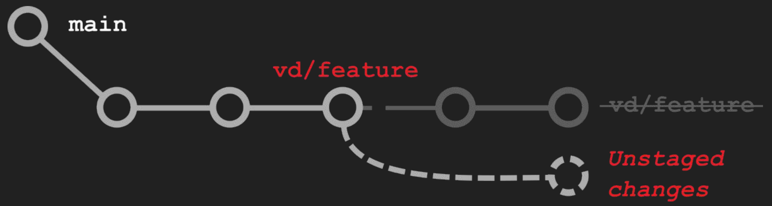
   → Takeaway: lets readers understand the code how you do

But how do I actually do this?



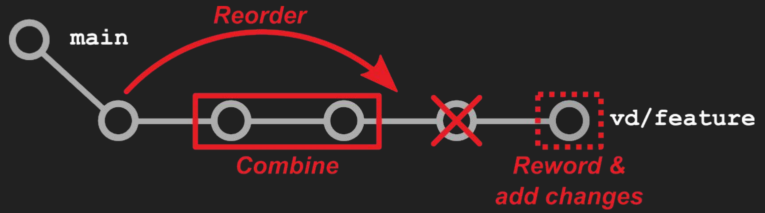
Re-committing from scratch?

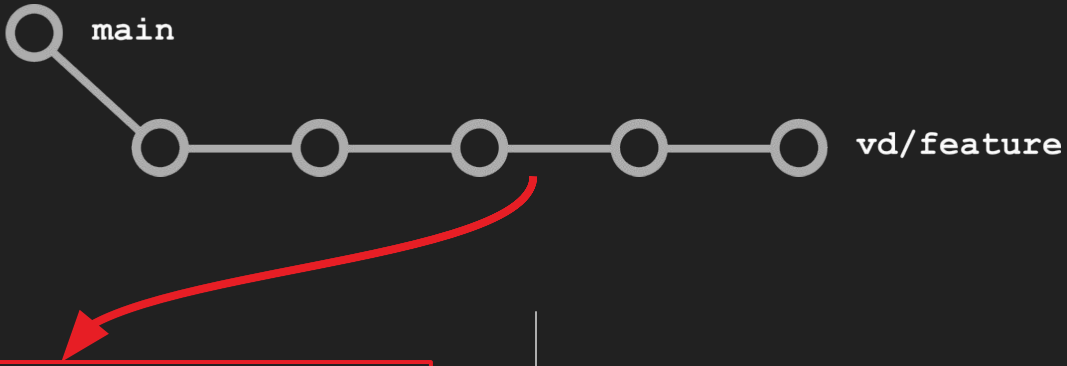
```
git reset <target>
```



Adjusting what you have?

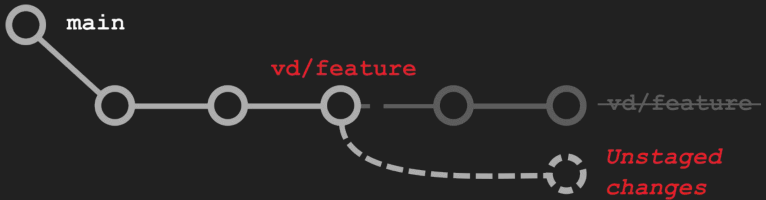
```
git commit --amend
git commit --fixup <target>
git rebase -i --keep-base <main>
```



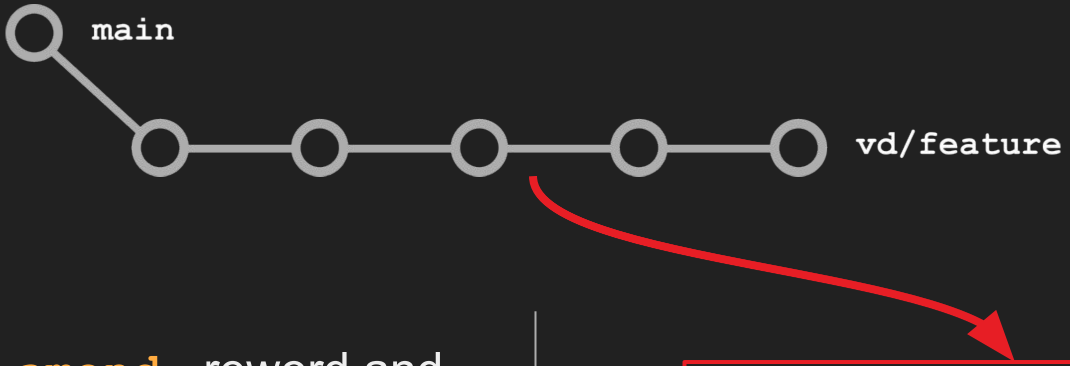


Re-committing from scratch?

```
git reset <target>
```



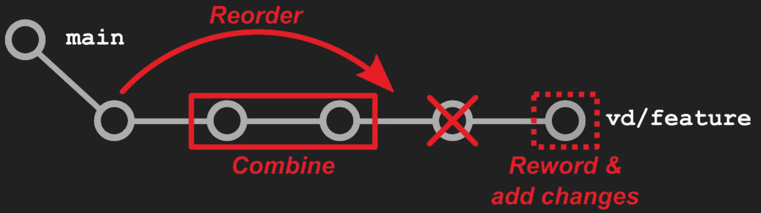
- The commits are “undone”.
- Your files don’t change!

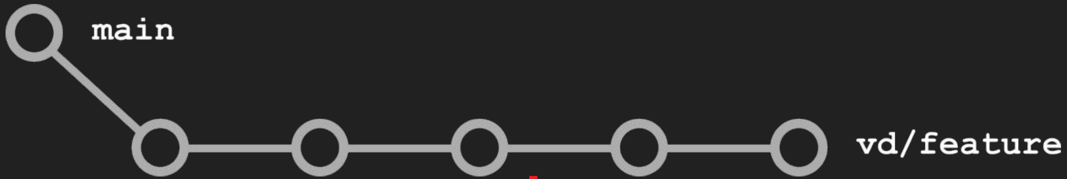


- `commit --amend` - reword and add changes to your *latest* commit
- `commit --fixup` - create a commit with a *special message* that, when rebased, combines with the target ("squashes")
- `rebase -i` - reorder, reword, drop, squash, etc. a list of commits

Adjusting what you have?

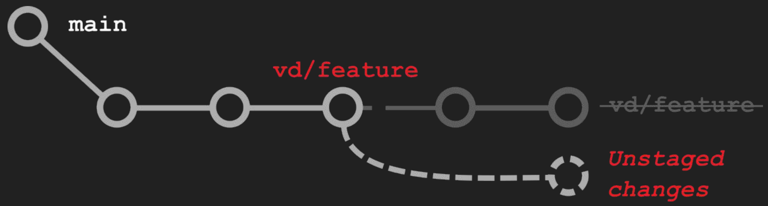
```
git commit --amend
git commit --fixup <target>
git rebase -i --keep-base <main>
```





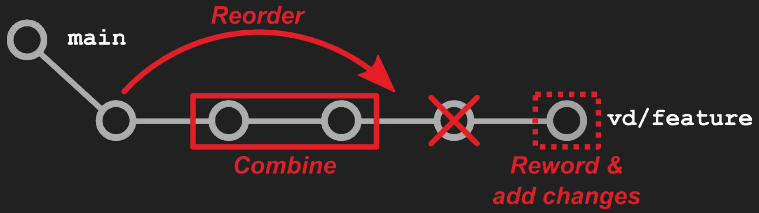
Re-committing from scratch?

```
git reset <target>
```



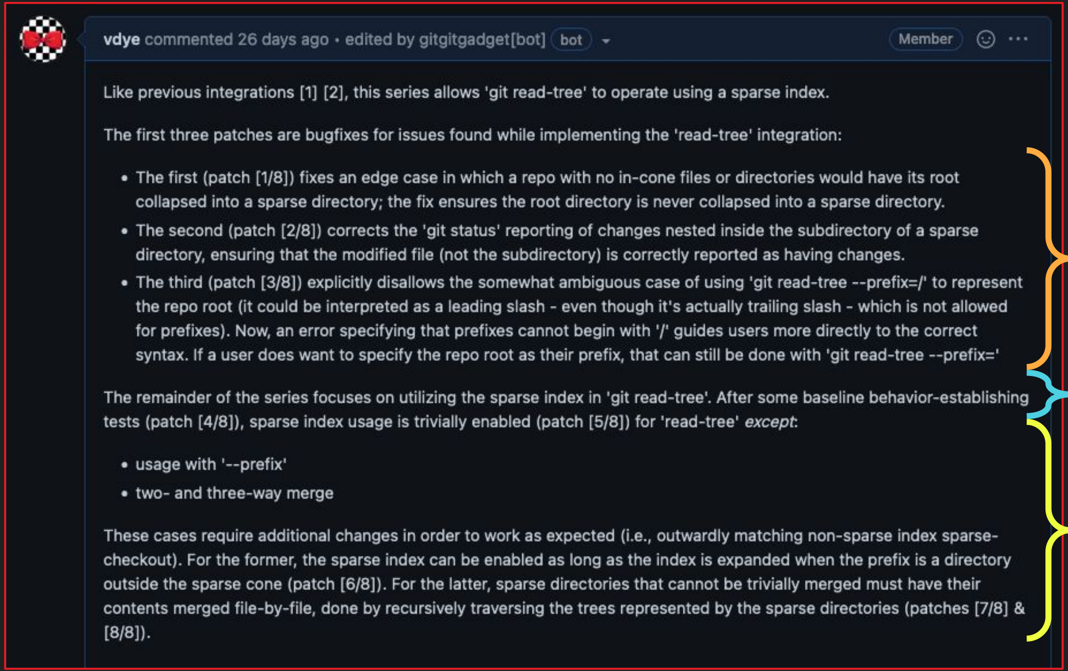
Adjusting what you have?

```
git commit --amend
git commit --fixup <target>
git rebase -i --keep-base <main>
```



I. Context
II. Writing Good Commits
III. Performing Commit-by-Commit Reviews
IV. Utilizing the Commit History

Reviewing commit-by-commit


vdye commented 26 days ago • edited by gitgitgadget[bot] bot Member

Like previous integrations [1] [2], this series allows 'git read-tree' to operate using a sparse index.

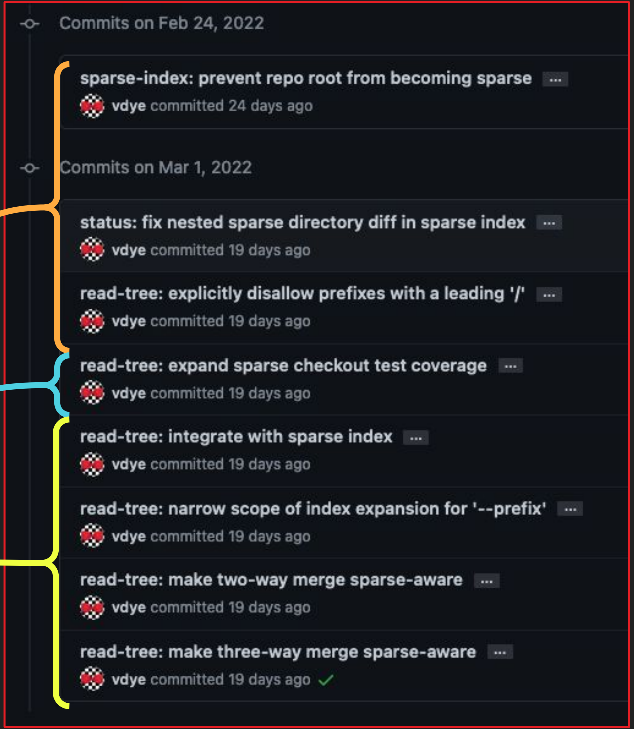
The first three patches are bugfixes for issues found while implementing the 'read-tree' integration:

- The first (patch [1/8]) fixes an edge case in which a repo with no in-cone files or directories would have its root collapsed into a sparse directory; the fix ensures the root directory is never collapsed into a sparse directory.
- The second (patch [2/8]) corrects the 'git status' reporting of changes nested inside the subdirectory of a sparse directory, ensuring that the modified file (not the subdirectory) is correctly reported as having changes.
- The third (patch [3/8]) explicitly disallows the somewhat ambiguous case of using 'git read-tree --prefix=/' to represent the repo root (it could be interpreted as a leading slash - even though it's actually trailing slash - which is not allowed for prefixes). Now, an error specifying that prefixes cannot begin with '/' guides users more directly to the correct syntax. If a user does want to specify the repo root as their prefix, that can still be done with 'git read-tree --prefix='

The remainder of the series focuses on utilizing the sparse index in 'git read-tree'. After some baseline behavior-establishing tests (patch [4/8]), sparse index usage is trivially enabled (patch [5/8]) for 'read-tree' *except*:

- usage with '--prefix'
- two- and three-way merge

These cases require additional changes in order to work as expected (i.e., outwardly matching non-sparse index sparse-checkout). For the former, the sparse index can be enabled as long as the index is expanded when the prefix is a directory outside the sparse cone (patch [6/8]). For the latter, sparse directories that cannot be trivially merged must have their contents merged file-by-file, done by recursively traversing the trees represented by the sparse directories (patches [7/8] & [8/8]).


Commits on Feb 24, 2022

sparse-index: prevent repo root from becoming sparse
vdye committed 24 days ago

Commits on Mar 1, 2022

status: fix nested sparse directory diff in sparse index
vdye committed 19 days ago

read-tree: explicitly disallow prefixes with a leading '/'
vdye committed 19 days ago

read-tree: expand sparse checkout test coverage
vdye committed 19 days ago

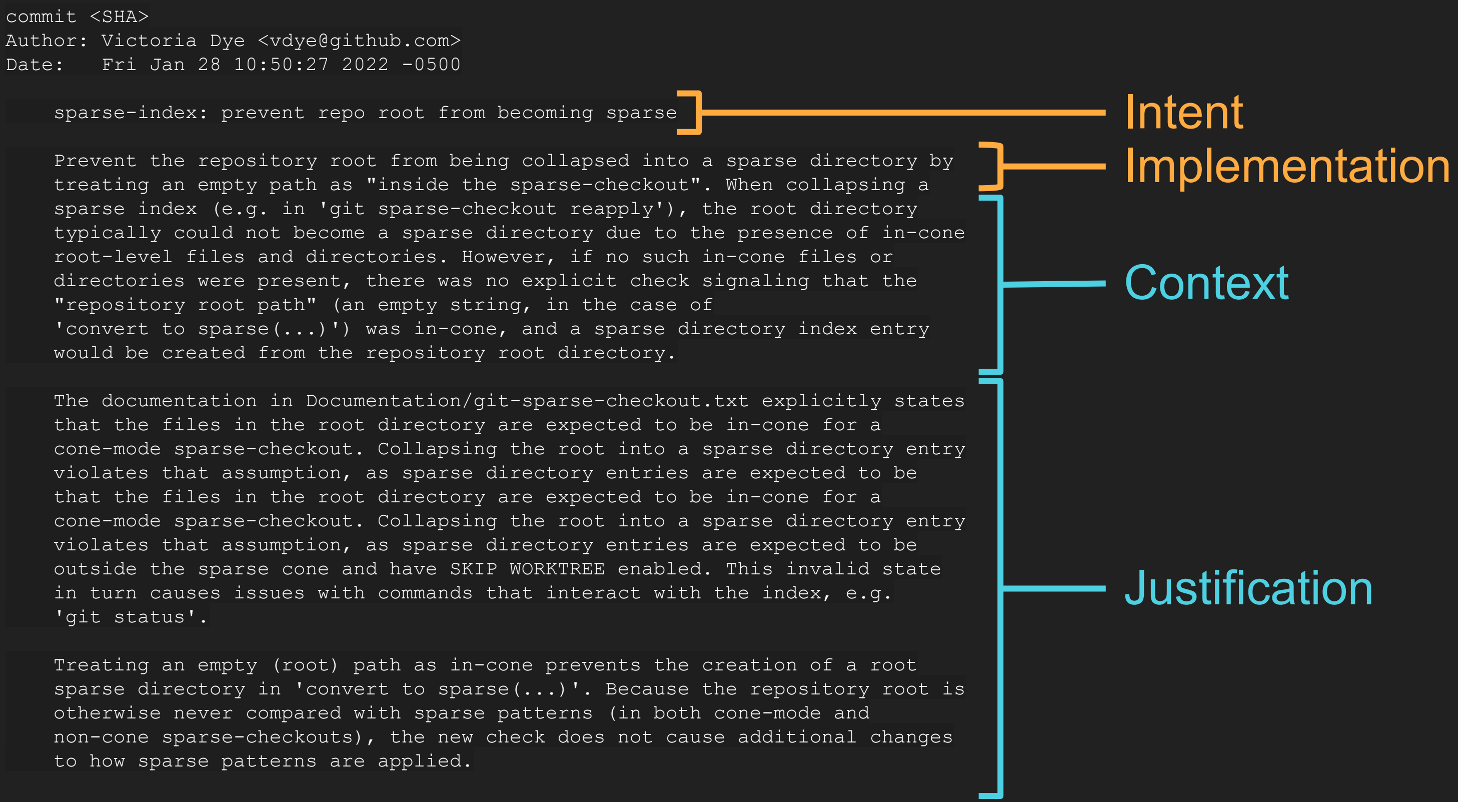
read-tree: integrate with sparse index
vdye committed 19 days ago

read-tree: narrow scope of index expansion for '--prefix'
vdye committed 19 days ago

read-tree: make two-way merge sparse-aware
vdye committed 19 days ago

read-tree: make three-way merge sparse-aware
vdye committed 19 days ago ✓

Reviewing commit-by-commit



```
commit <SHA>
Author: Victoria Dye <vdye@github.com>
Date:   Fri Jan 28 10:50:27 2022 -0500

    sparse-index: prevent repo root from becoming sparse

    Prevent the repository root from being collapsed into a sparse directory by
    treating an empty path as "inside the sparse-checkout". When collapsing a
    sparse index (e.g. in 'git sparse-checkout reapply'), the root directory
    typically could not become a sparse directory due to the presence of in-cone
    root-level files and directories. However, if no such in-cone files or
    directories were present, there was no explicit check signaling that the
    "repository root path" (an empty string, in the case of
    'convert to sparse(...)') was in-cone, and a sparse directory index entry
    would be created from the repository root directory.

    The documentation in Documentation/git-sparse-checkout.txt explicitly states
    that the files in the root directory are expected to be in-cone for a
    cone-mode sparse-checkout. Collapsing the root into a sparse directory entry
    violates that assumption, as sparse directory entries are expected to be
    that the files in the root directory are expected to be in-cone for a
    cone-mode sparse-checkout. Collapsing the root into a sparse directory entry
    violates that assumption, as sparse directory entries are expected to be
    outside the sparse cone and have SKIP WORKTREE enabled. This invalid state
    in turn causes issues with commands that interact with the index, e.g.
    'git status'.

    Treating an empty (root) path as in-cone prevents the creation of a root
    sparse directory in 'convert to sparse(...)'. Because the repository root is
    otherwise never compared with sparse patterns (in both cone-mode and
    non-cone sparse-checkouts), the new check does not cause additional changes
    to how sparse patterns are applied.
```

Reviewing commit-by-commit

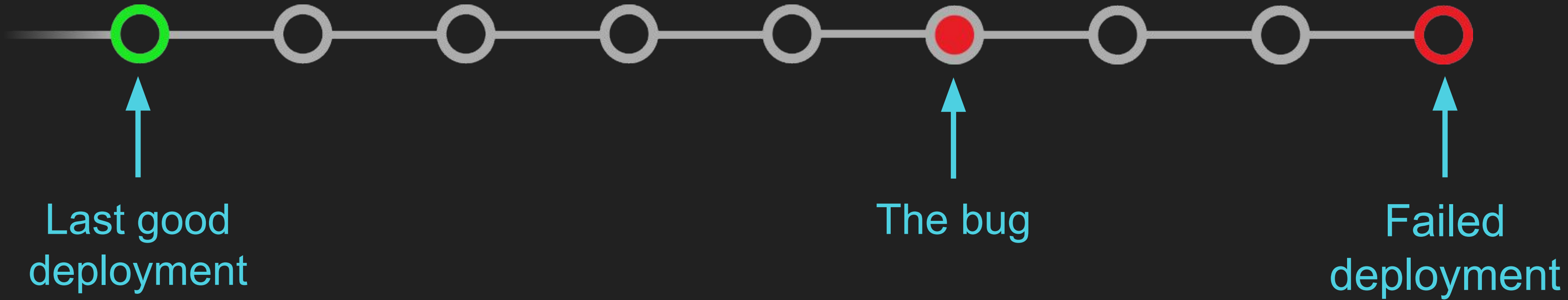
Implementation

Prevent the repository root from being collapsed into a sparse directory by **treating an empty path as "inside the sparse-checkout"**.

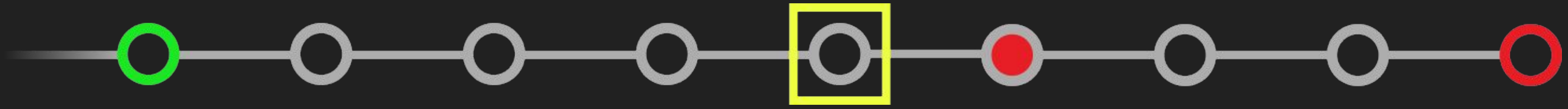
```
diff --git a/dir.c b/dir.c
index d91295f2bc..a136377eb4 100644
--- a/dir.c
+++ b/dir.c
@@ -1463,10 +1463,11 @@ static int path_in_sparse_checkout_1(const char
*path,
        const char *end, *slash;

        /*
-        * We default to accepting a path if there are no patterns or
-        * they are of the wrong type.
+        * We default to accepting a path if the path is empty, there are no
+        * patterns, or the patterns are of the wrong type.
         */
-       if (init_sparse_checkout_patterns(istate) ||
+       if (!*path ||
+           init_sparse_checkout_patterns(istate) ||
            (require_cone_mode &&
            !istate->sparse_checkout_patterns->use_cone_patterns))
                return 1;
```

git bisect

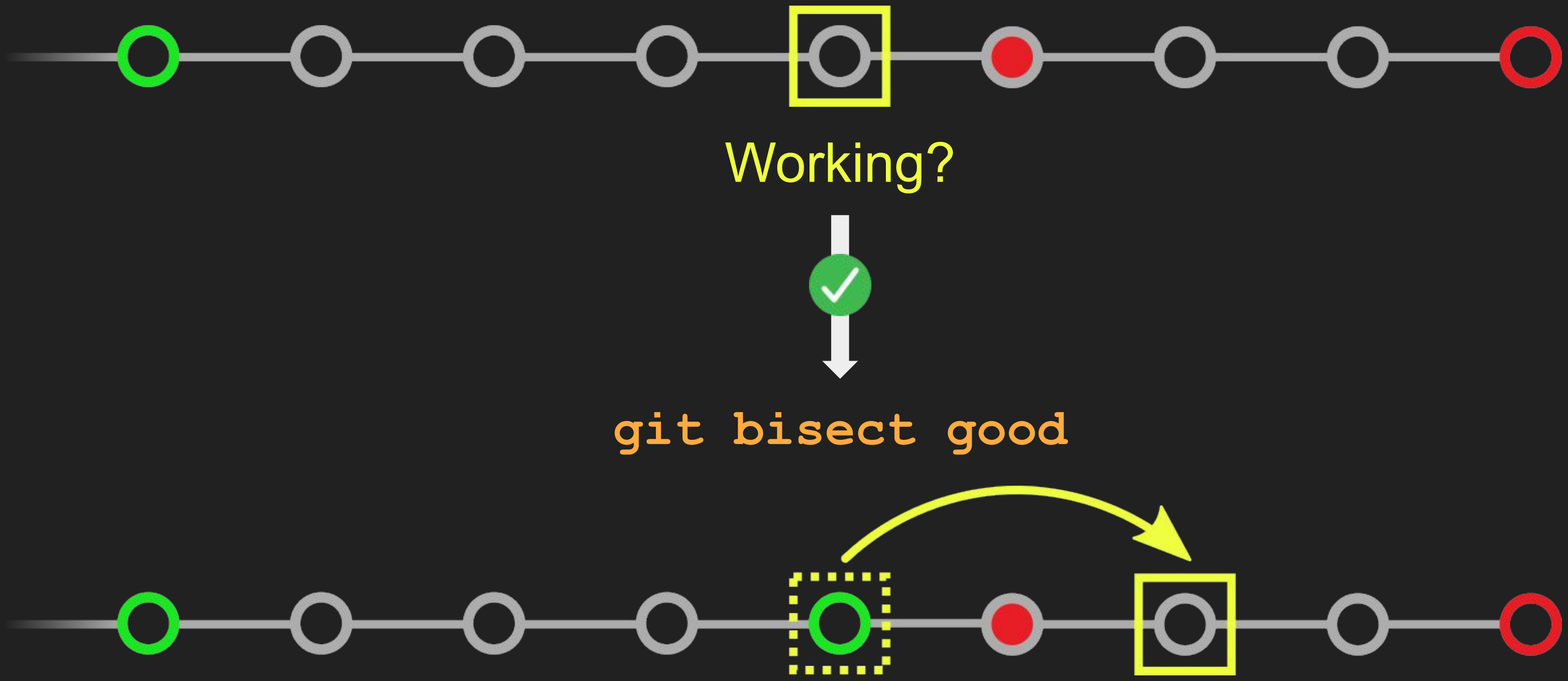
Narrow down the source of a bug to a specific commit



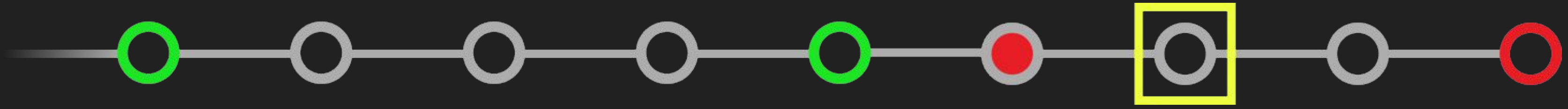
```
git bisect start <bad> <good>
```



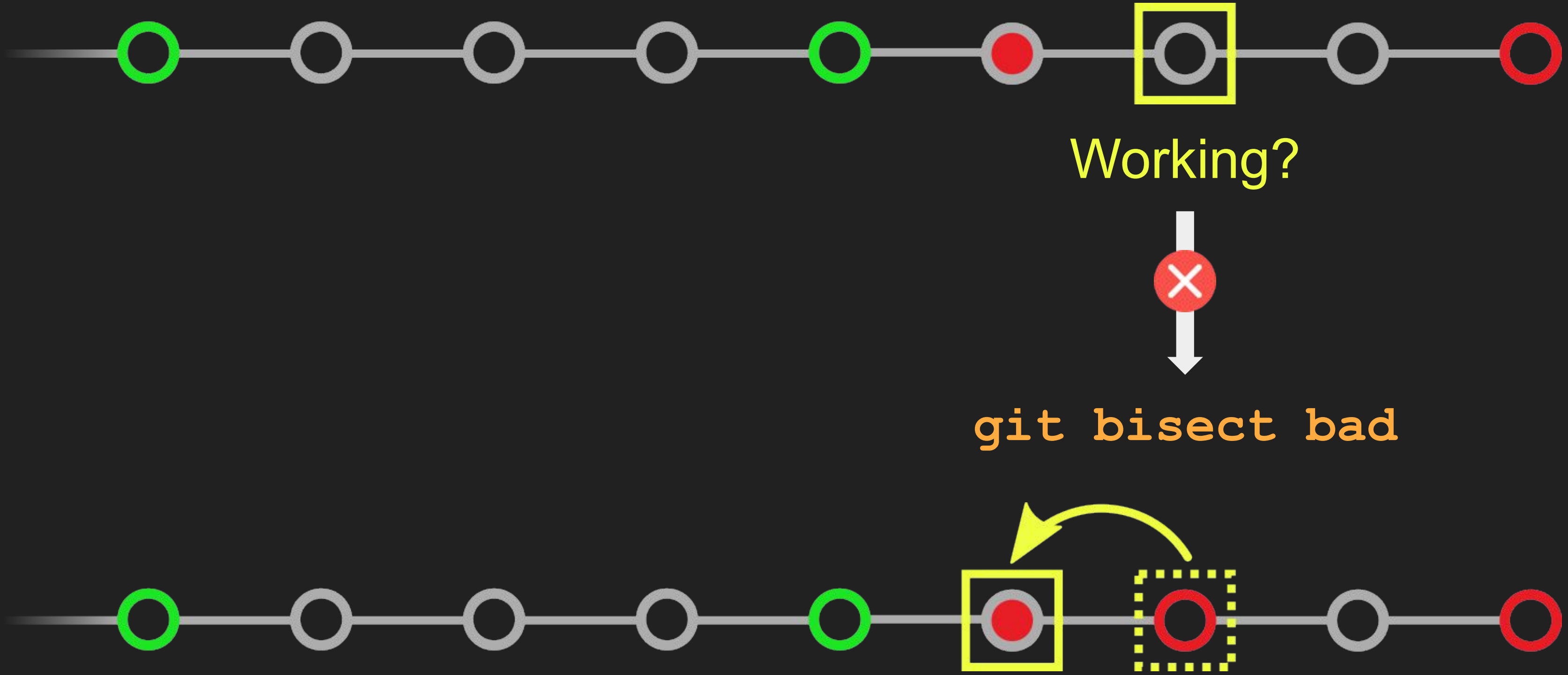
Working?



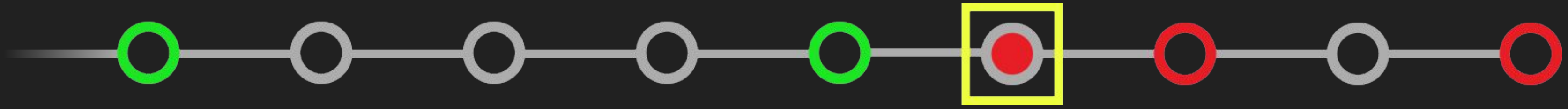
```
git bisect good
```



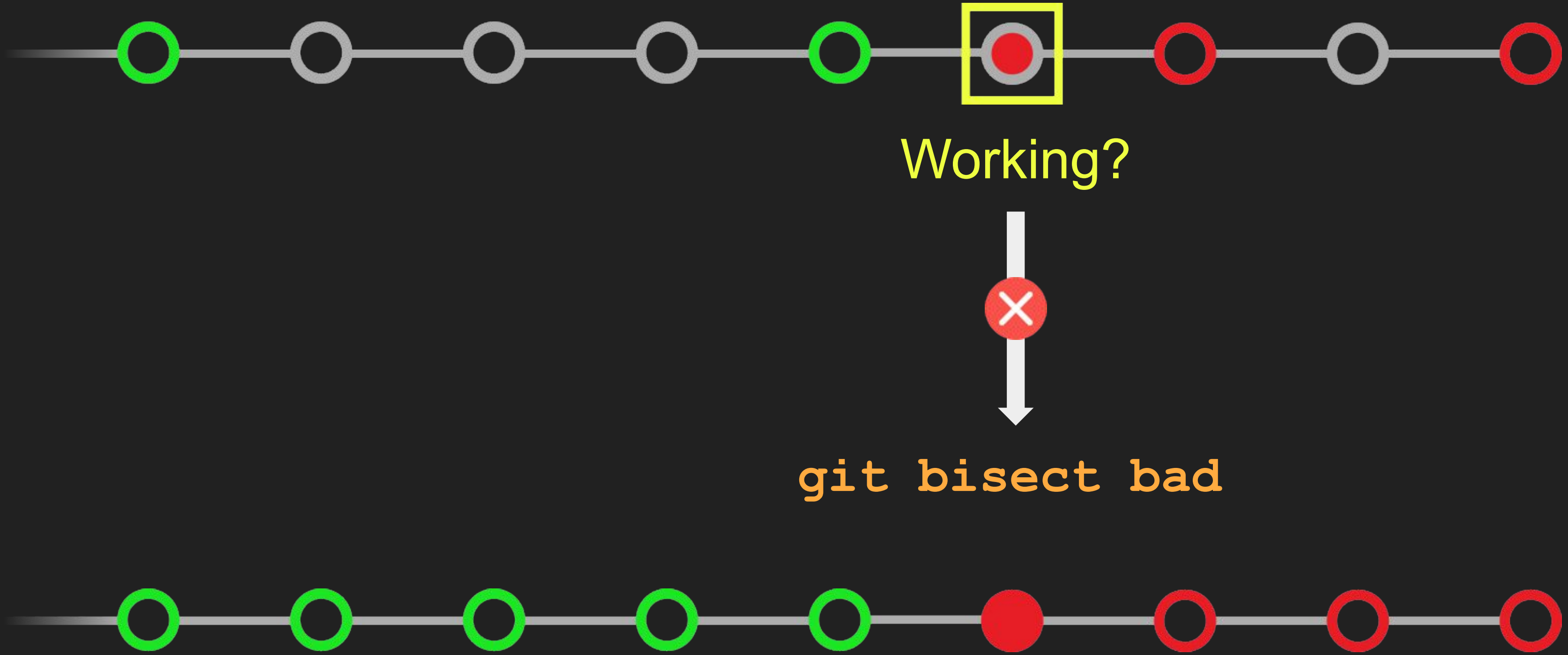
Working?



```
git bisect bad
```

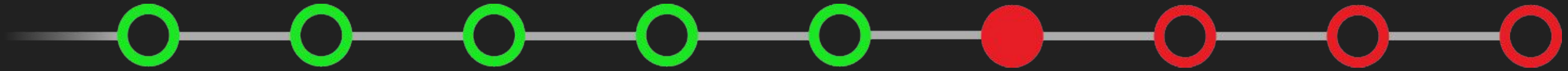


Working?



```
git bisect bad
```

🎉 Found the bug! 🎉



…but why did it happen
in the first place?

git blame

Find out which commit last changed a line of code

```
$ git blame -s my-file.py
abd52642da46 my-file.py  1) import os
603ab927a0dd oldname.py  3) import re
603ab927a0dd oldname.py  4)
603ab927a0dd oldname.py  5) print("Hello world")
abd52642da46 my-file.py  5) print(os.stat("README"))
...
```

git log

Search commits by file, change location, and/or message

```
$ git log --oneline
09823ba09de1 README.md: update maintainer contact
abd52642da46 my-file.py: add README stat printout
7392d7dbb9ae my-file.py: rename from oldname.py
5ad823d1bc48 test.py: commonize test setup
603ab927a0dd oldname.py: create printout script
...
```

```
$ git log --oneline -- my-file.py
abd52642da46 my-file.py: add README stat printout
7392d7dbb9ae my-file.py: rename from oldname.py
603ab927a0dd oldname.py: create printout script
...
```

I. 🎉 Context 🎉
II. ✨ Writing Good Commits ✨
III. 🎊 Performing Commit-by-Commit Reviews 🎊
IV. 🥳 Utilizing the Commit History 🥳

Remember these things!

Git commits **contextualize** your code for a broader audience.

You can improve the quality of your commits today by organizing a **narrative**, making changes **small & atomic**, and explaining **“what” & “why”**.

Spending time on writing high-quality commits is helpful for **anyone** and **everyone** involved in your open- or closed-source project.

Questions?

Download these slides: https://vdye.github.io/2022/OS101-Writing-Commits.pdf