



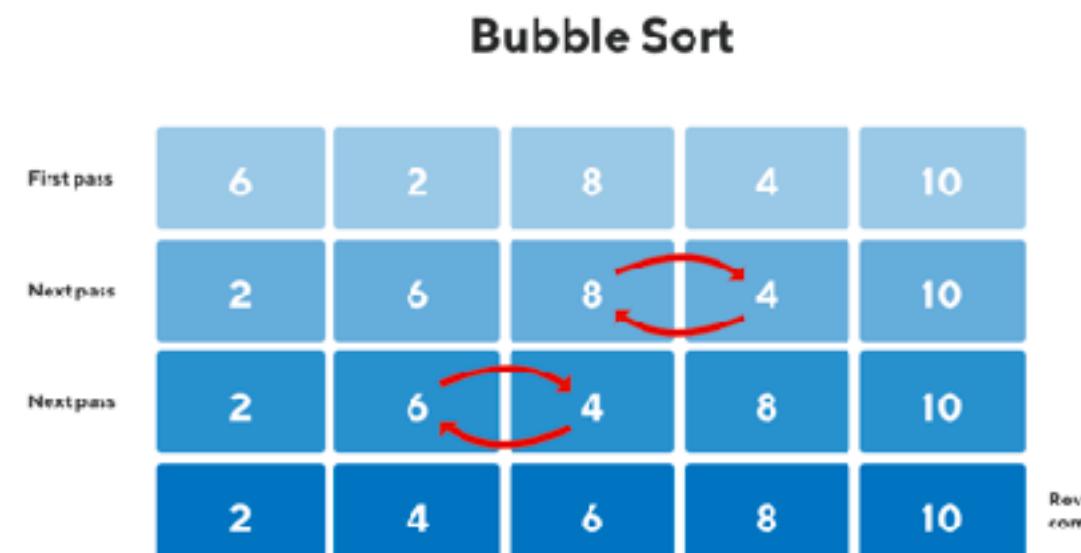
# Lecture 5: Teams and Communication

17-313: Foundations of Software Engineering

# Learning Goals

- Describe the pros and cons of working as a team
- Recognize the importance of communication in collaboration
- Recognize the need of having multiple communication channels
- Select an appropriate communication tool for a given communication goal
- Ask technical questions effectively
- Write clear and specific Github issues, pull requests, and comments

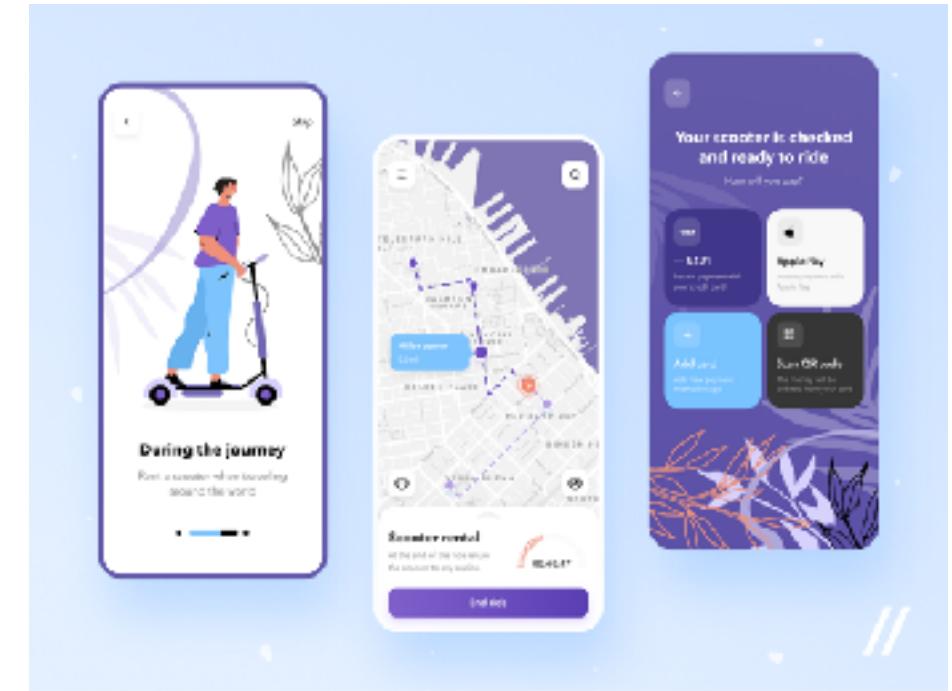
# We all work in a team



Bubble Sort



Monopoly Game



Scooter App

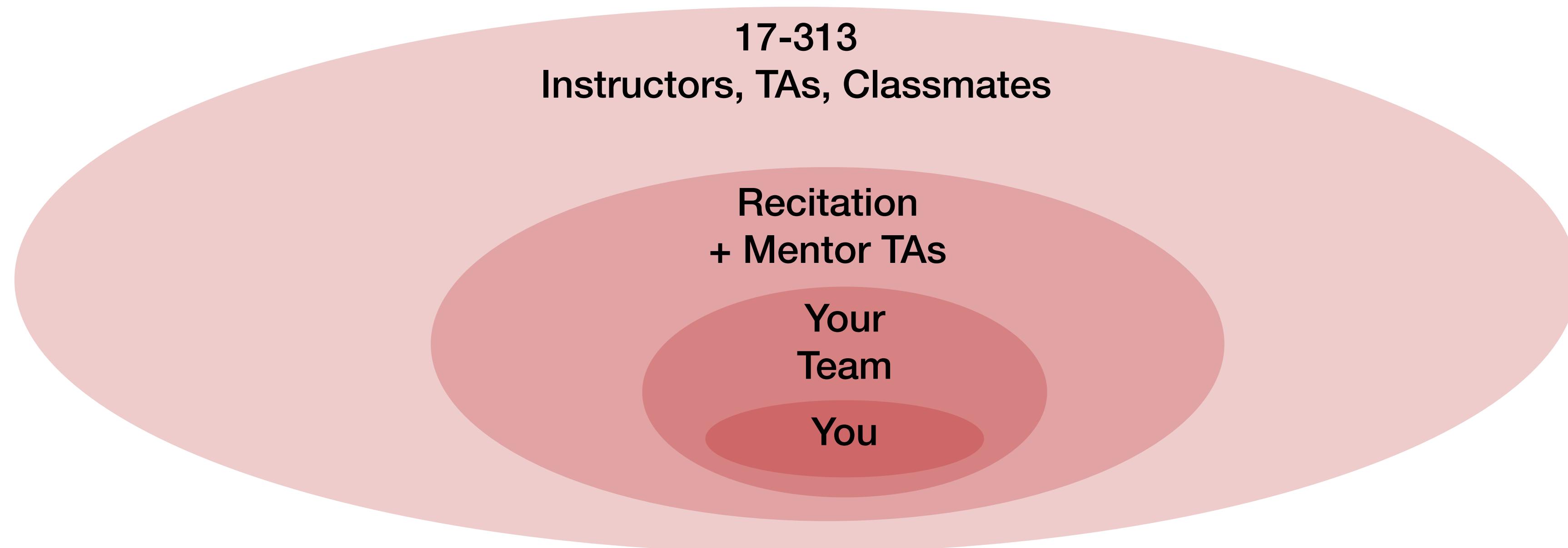
teedy

Teedy

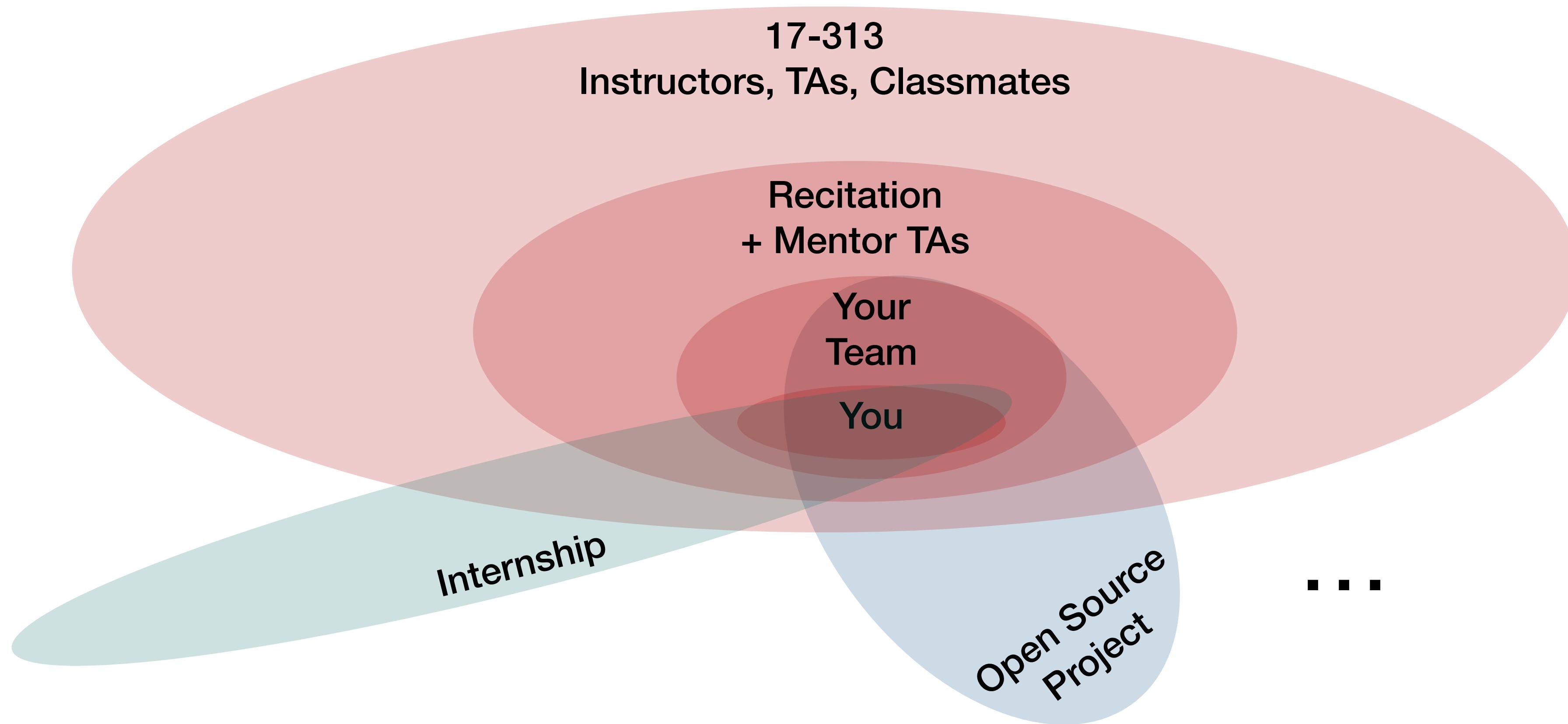


Self-driving car

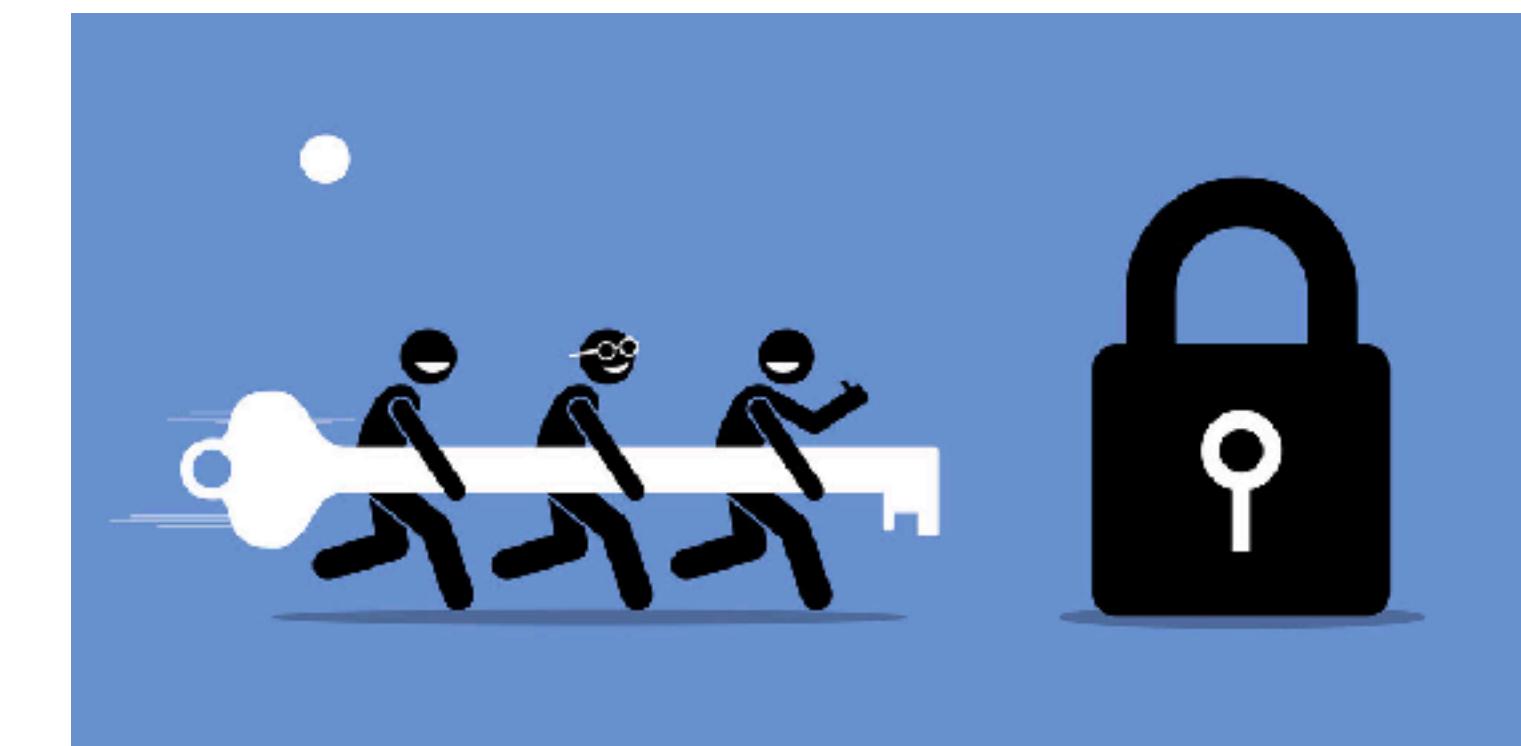
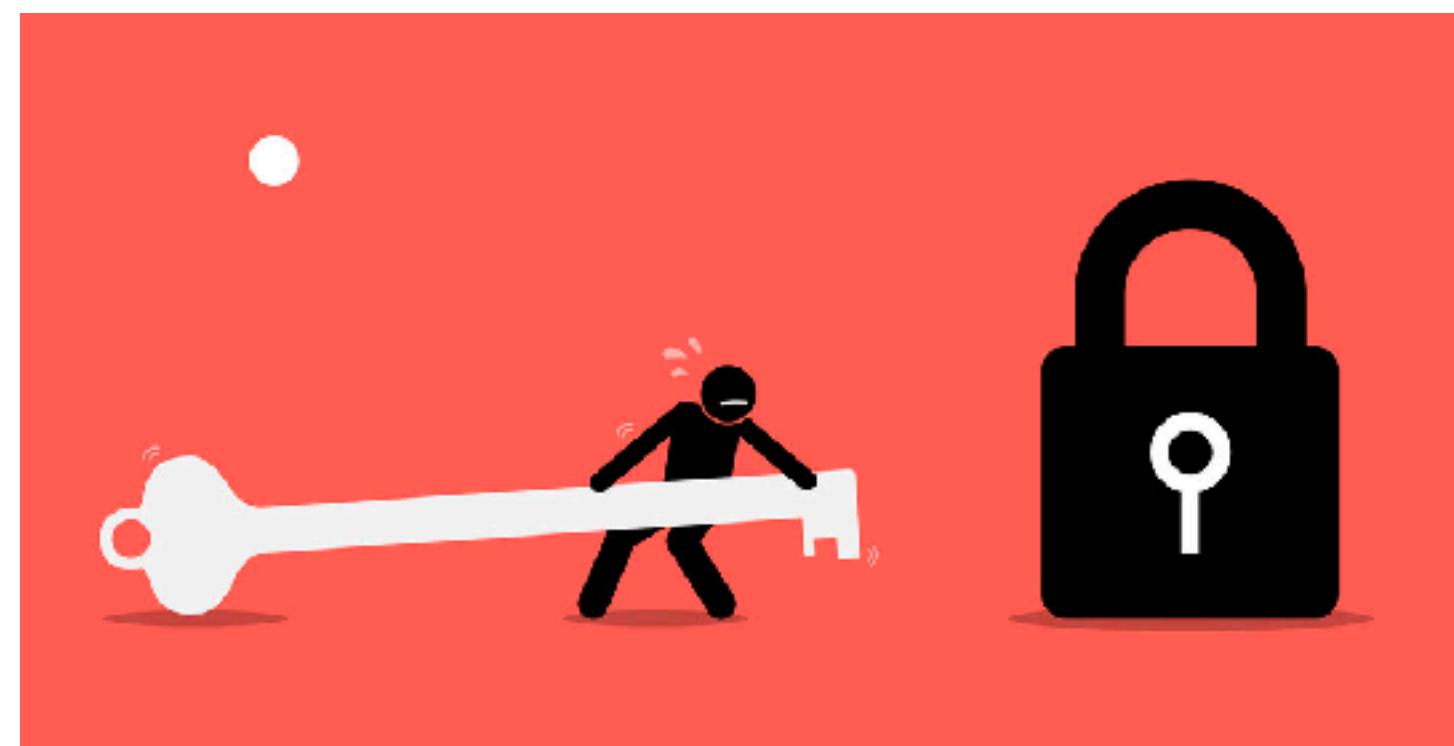
# We all work in a team



# We all work in a team



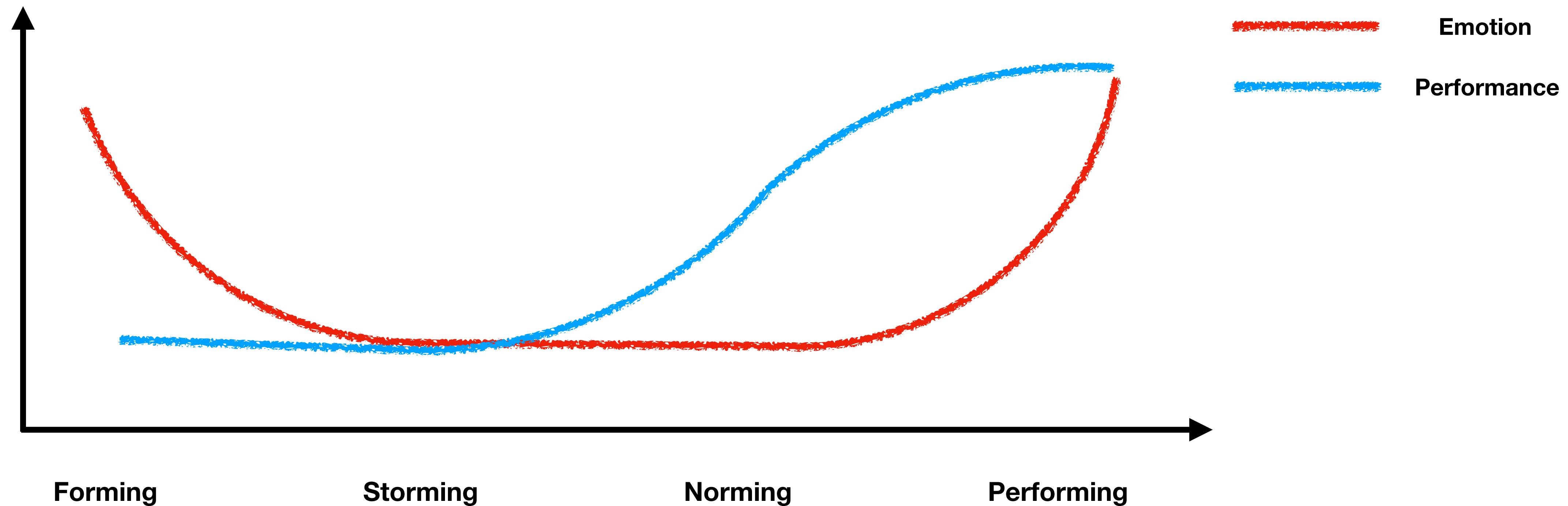
# [Activity] Working solo vs as a team



# Working as a team

- Design & implement software
  - + Establish a collaboration process
  - + Meet with the team
  - + Divide work and integrate
  - + Share knowledge
  - + Resolve conflicts

# Stages of Team Formation



Tuckman, B. W. (1965). Developmental sequence in small groups. *Psychological Bulletin*, 63, 384-399.

**Projects**

Project 1: Hello, NodeBB! &gt;

Project 2: Collaborative Development

2A Team Process &amp; Planning

2B First Sprint

2C Second Sprint

## Main Deliverables

[↑ Back to top](#)

Or

De

Te

:

I

M

I

Gr

### Teamwork Contract (40 pts)

When working with a team, it is important to discuss each team member's background, and establish common expectations of the team. Miscommunication or the general lack of communication are often the most common causes of team conflict.

#### Team Conflict Example

A common conflict in working style is when there are team members who always want to get a headstart on their work, while there are team members who are fine with doing work a few days before the deadline. It causes panic in the former team members, while the latter team members feel frustrated as to why they are being rushed.

As such, your first process task of the semester will be creating a teamwork contract with your teammates. It is a **1 - 2 page** document containing information that all teammates agree to follow. You should work on the contract **with all members present**. We recommend that you keep it to around 1 page, 2 page is a **hard limit**.

Additionally, **it is more important that you only include statements that the team will adhere to** than it is to fulfill the length requirement (quality over quantity!) You do not need to write full sentences (bullet points are okay), but your decisions must be clearly conveyed in the document.

You are free to include anything that your team deems necessary, but you should minimally address the following sections:

#### 1. Expectations

How much time is each team member expected to be putting into working on projects? Punctuality? How

[https://cmu-313.github.io/projects/P2/1\\_teamprocess/](https://cmu-313.github.io/projects/P2/1_teamprocess/)

(11) General Interference with Organizations and Production

(a) Organizations and Conferences

(1) Insist on doing everything through "channels." Never permit short-cuts to be taken in order to expedite decisions.

(2) Make "speeches." Talk as frequently as possible and at great length. Illustrate your "points" by long anecdotes and accounts of personal experiences. Never hesitate to make a few appropriate "patriotic" comments.

(3) When possible, refer all matters to committees, for "further study and consideration." Attempt to make the committees as large as possible — never less than five.

(4) Bring up irrelevant issues as frequent as possible.

(5) Haggle over precise wordings of communications, minutes, resolutions.

(6) Refer back to matters decided upon at the last meeting and attempt to re-open the question of the advisability of that decision.

(7) Advocate "caution." Be "reasonable" and urge your fellow-conferees to be "reasonable" and avoid haste which might result in embarrassments or difficulties later on.

(8) Be worried about the propriety of a decision — raise the question of whether such action as is contemplated lies within the jurisdiction of the group or whether it might conflict with the policy of some higher echelon.

(b) Managers and Supervisors

(1) Demand written orders.

(2) "Misunderstand" orders. Ask endless questions or engage in long correspondence about such orders. Quibble over them when you can.

(3) Do everything possible to delay the delivery of orders. Even though parts of an order may be ready beforehand, don't deliver it until it is completely ready.

(4) Don't order new working material until your current stocks have been virtually exhausted, so that the slightest delay in filling your order will mean a shutdown.

(5) Order high-quality materials which are hard to get. If you don't get them argue about it. Warn that inferior materials will mean inferior work.

(6) In making work assignments, always sign out the unimportant jobs first. See that the important jobs are assigned to inefficient workers or poor machines.

(7) Insist on perfect work in relatively unimportant products; send back for refinishing those which have the least flaw. Approve other defective parts whose flaws are not visible to the naked eye.

(8) Make mistakes in routing so that parts and materials will be sent to the wrong place in the plant.

(9) When training new workers, give incomplete or misleading instructions.

(10) To lower morale and with it, production, be pleasant to inefficient workers; give them undeserved promotions. Discriminate against efficient workers; complain unjustly about their work.

(11) Hold conferences when there is more critical work to be done.

(12) Multiply paper work in plausible ways. Start duplicate files.

(13) Multiply the procedures and clearances involved in issuing instructions, pay checks, and so on. See that three people have to approve everything where one would do.

(14) Apply all regulations to the last letter.

(c) Office Workers

(1) Make mistakes in quantities of material when you are copying orders. Confuse similar names. Use wrong addresses.

(2) Prolong correspondence with government bureaus.

(3) Misfile essential documents.

(4) In making carbon copies, make one too few, so that an extra copying job will have to be done.

(5) Tell important callers the boss is busy or talking on another telephone.

(6) Hold up mail until the next collection.

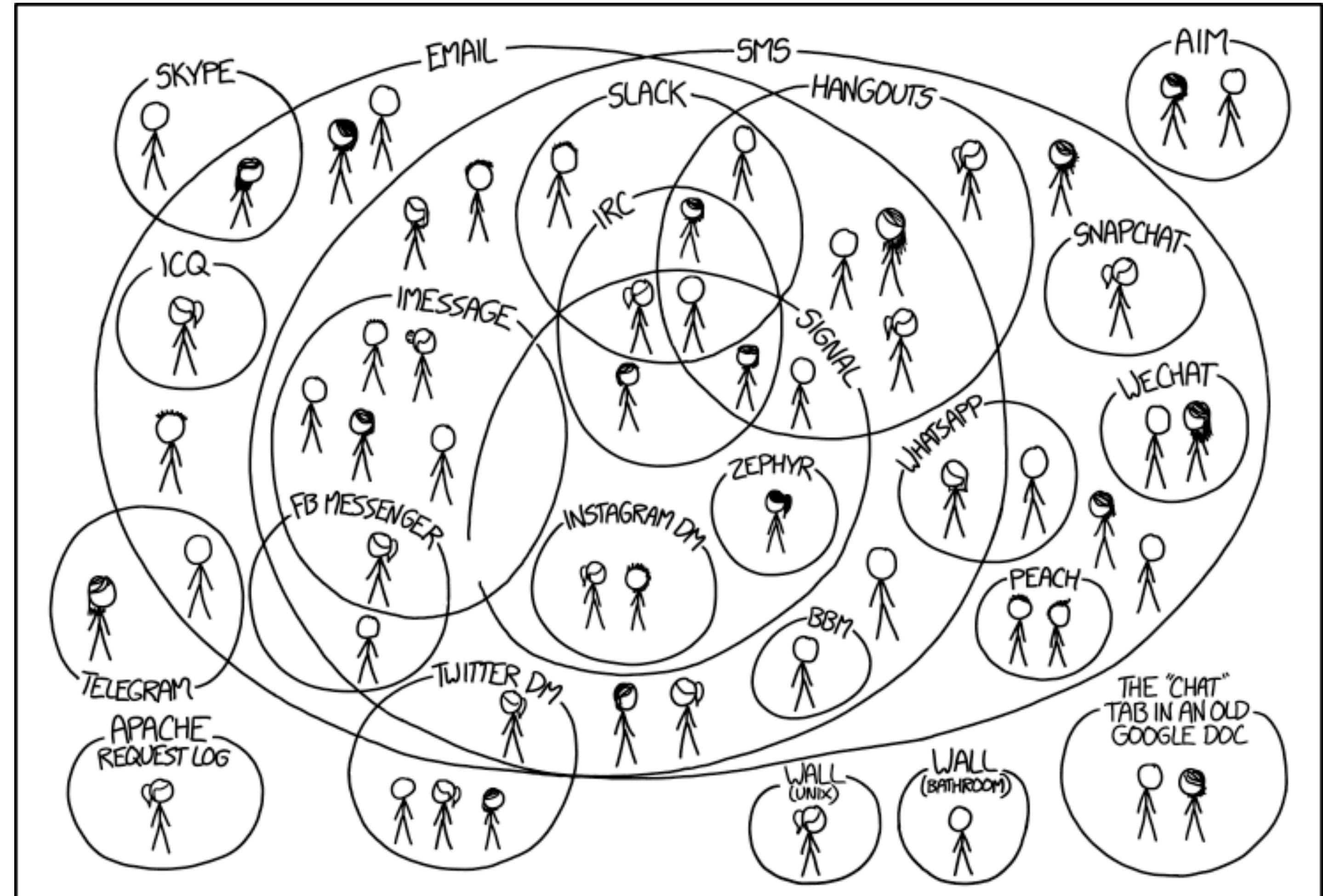
(7) Spread disturbing rumors that sound like inside dope.

(d) Employees

(1) Work slowly. Think out ways to increase the number of movements necessary on your job: use a light hammer instead of a heavy one, try to make a small wrench do when a big one is necessary, use little force where considerable force is needed, and so on.

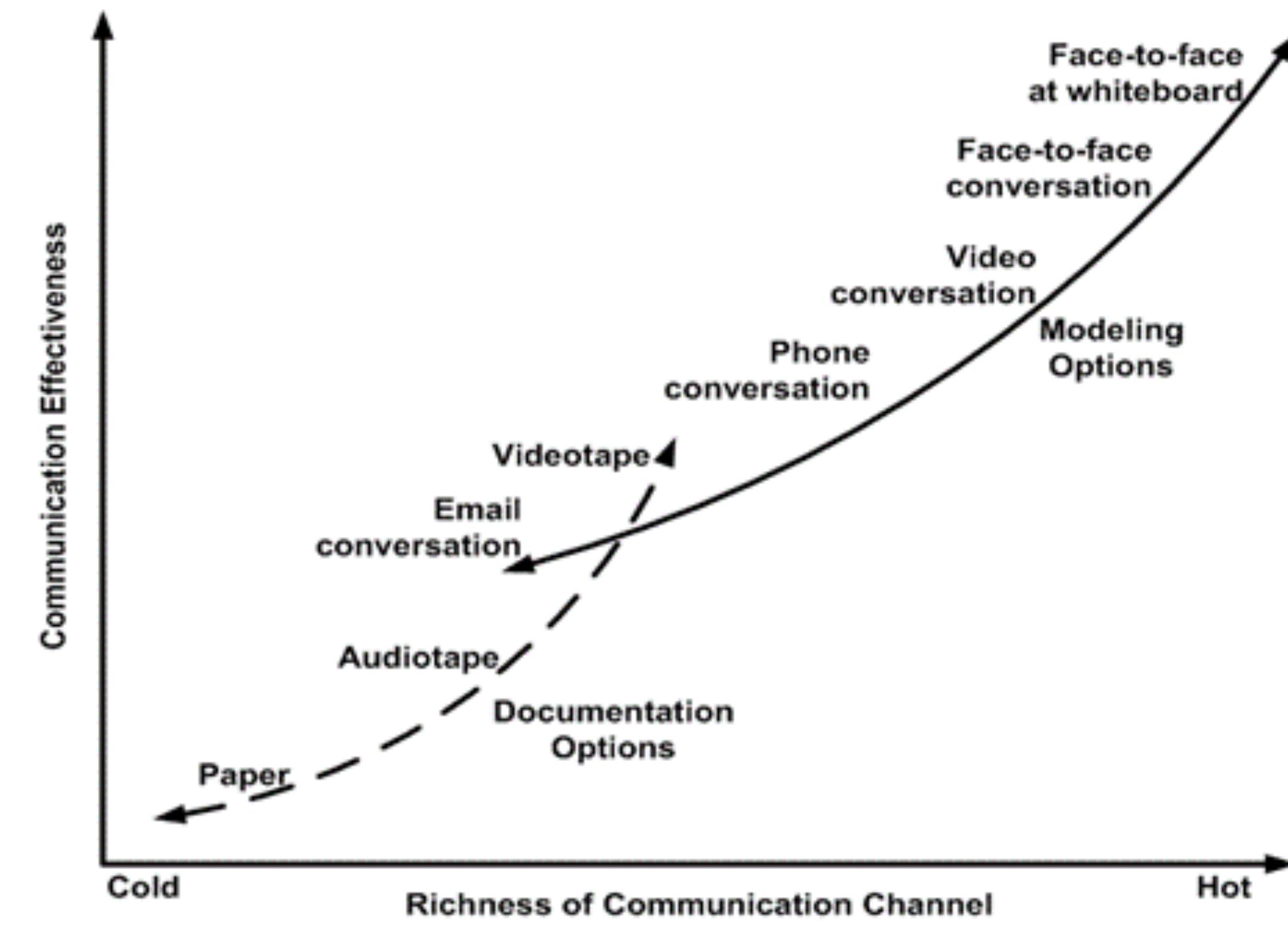
(2) Contrive as many interruptions to your work as you can: when changing the material on which you are working, as you would on a lathe or punch, take needless time to do it. If you are cutting, shaping or doing other measured work, measure dimensions twice as often as you need to. When you go to the lavatory, spend a longer time there than is necessary. Forget tools so that you will have to go back after them.

# **Establish a collaboration process**



I HAVE A HARD TIME KEEPING TRACK OF WHICH CONTACTS USE WHICH CHAT SYSTEMS.

# Select the right comm. tools



# Establish communication patterns

- Asana, Trello, Microsoft Projects, ...
- Github Wiki, Google Docs, Notion, ...
- Github Issues, Jira, ...
- Email, Slack, Facebook groups, ...
- Zoom, Microsoft Teams, Skype, Phone call, ...
- Face-to-face meetings

# 17-313 Communication channels

- Slack
- Regular meeting (Lectures, Recitations)
- Office Hour
- Canvas, Gradescope
- Webpage

# Check out other projects'

## Communication

- Forums: Discuss implementations, research, etc. <https://discuss.pytorch.org>
- GitHub Issues: Bug reports, feature requests, install issues, RFCs, thoughts, etc.
- Slack: The PyTorch Slack hosts a primary audience of moderate to experienced PyTorch users and developers for general chat, online discussions, collaboration, etc. If you are a beginner looking for help, the primary medium is PyTorch Forums. If you need a slack invite, please fill this form:  
<https://goo.gl/forms/PP1AGvNHpSaJP8to1>
- Newsletter: No-noise, a one-way email newsletter with important announcements about PyTorch. You can sign-up here: <https://eepurl.com/cbG0rv>
- Facebook Page: Important announcements about PyTorch. <https://www.facebook.com/pytorch>
- For brand guidelines, please visit our website at [pytorch.org](http://pytorch.org)

# Communication expectation

- Quality of service guarantee
  - How soon will you get back to your teammates?
  - Weekend? Evening?
- Emergency
  - Tag w/ 911
  - Notify everyone with @channel

# Running a meeting

# How to run a meeting

- The Three Rules of Running a Meeting
  - Set the Agenda
  - Start on Time. End on Time.
  - End with Action Items (and share them - Github Issues, Meeting Notes, ...)

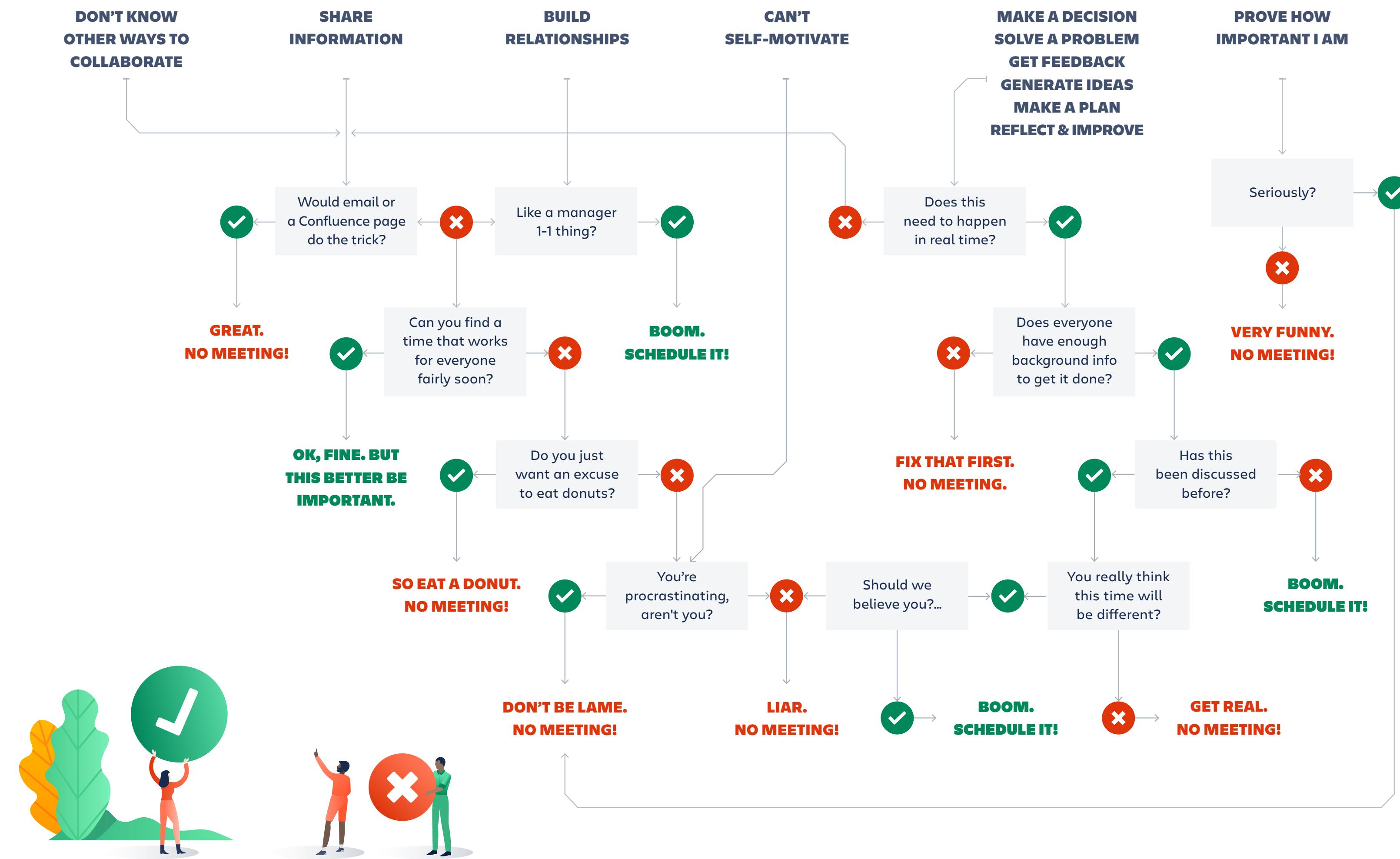
# How to run a meeting

- Set and document clear responsibilities and expectations
- Make everyone contribute
  - Possible Roles: Coordinator, Scribe, Checker
  - Manage Personalities
  - Be Vulnerable

# Random Advice

- Note takers have a lot of power to steer the meeting
  - Collaborative notes are even better!
- Different meeting types have different best practices
  - Decision-making meeting
  - Brainstorming meeting
  - One-on-one meeting
  - Working sessions

# WHY DO YOU WANT TO CALL A MEETING?



ATLASSIAN

**Divide work and integrate**

# Is this issue useful?

◀ Image Slider #2

① Open calebsylvest opened this issue just now · 0 comments

calebsylvest commented just now

The image slider is broken

Write Preview Comments are parsed with GitHub Flavored Markdown

Leave a comment

Attach images by dragging & dropping, [selecting them](#), or pasting from the clipboard.

Close Comment

Unsubscribe You are receiving notifications because you were assigned.

Edit New issue

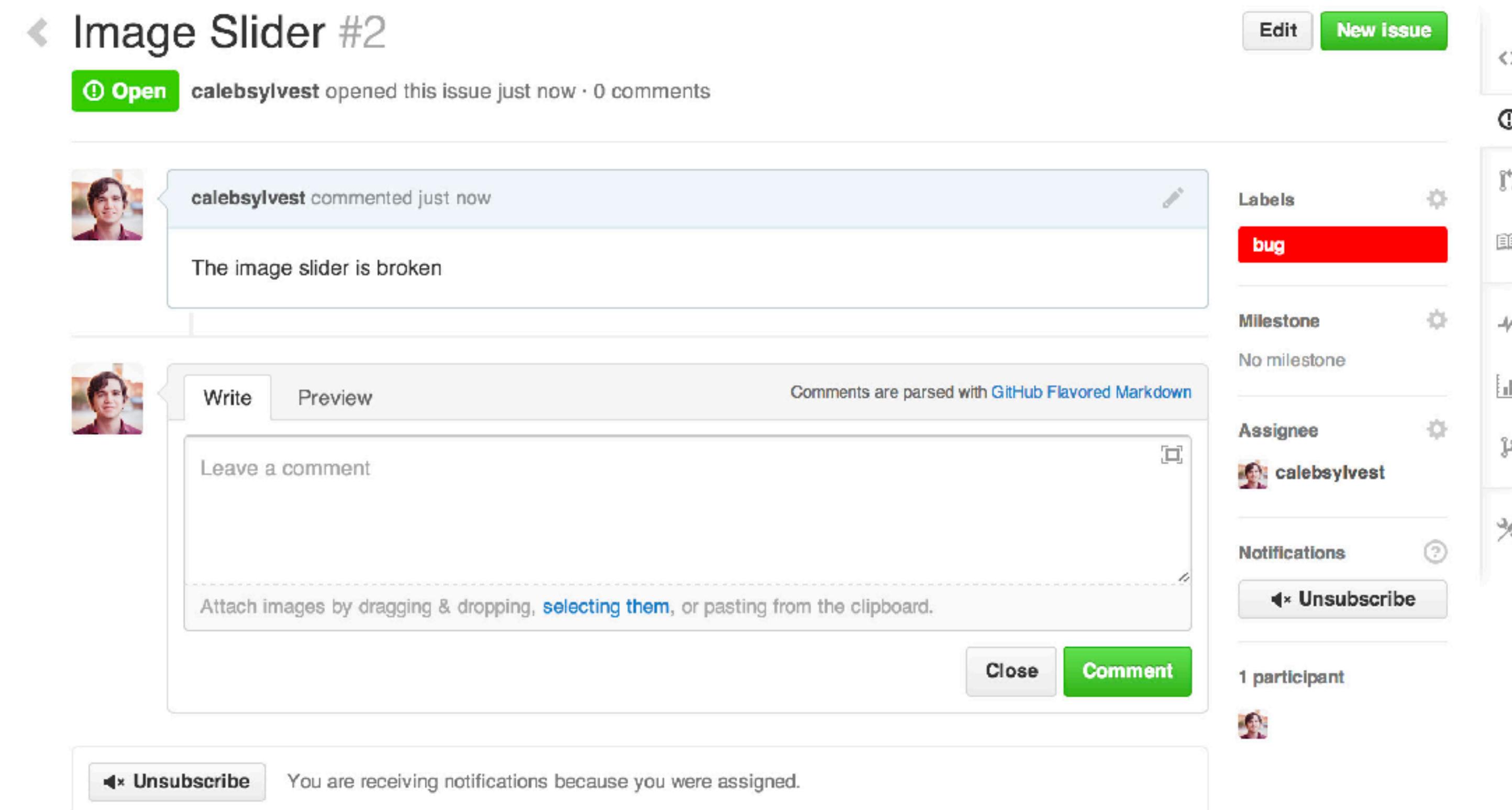
Labels bug

Milestone No milestone

Assignee calebsylvest

Notifications Unsubscribe

1 participant



# Writing useful Github issues

◀ Cropping of Image Slider Pics #3

**Open** calebsylvest opened this issue just now · 0 comments

calebsylvest commented just now

<http://calebsylvest.com/>

The cropping of the images in the slideshow seem to be off. The text is not visible and partially hidden by content below. The Developer Tools show the full-size un-cropped image is being loaded, but obviously not displaying.

Browser: Google Chrome  
OS: Mavericks  
Hardware: MacBook Pro Retina

Labels

bug

Milestone

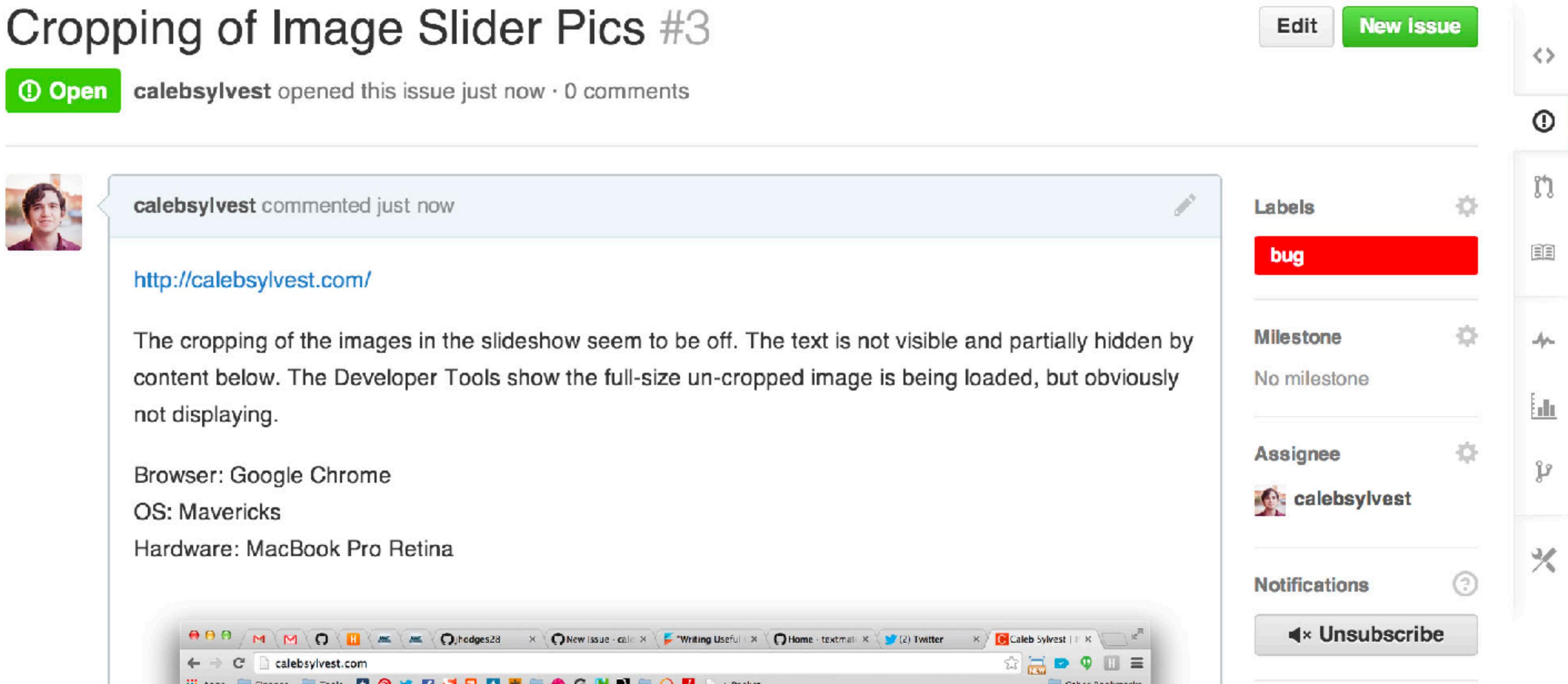
No milestone

Assignee

calebsylvest

Notifications

Unsubscribe



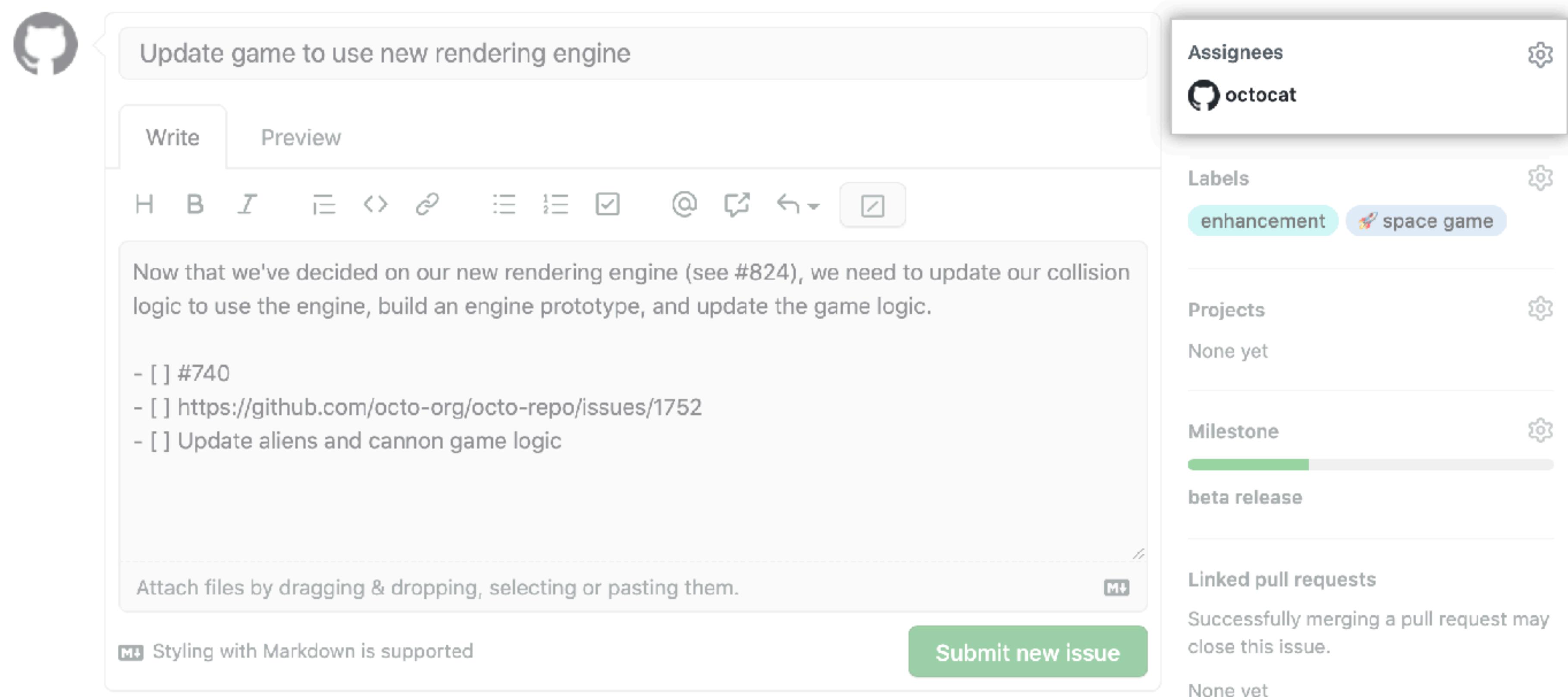
# Writing useful Github issues

- Issue should include:
  - Context: explain the conditions which led you to write the issue
  - Problem or idea: the context should lead to something
  - Previous attempts to solve
  - Solution or next step (if possible)
- Don't be vague!
  - Include environment settings, versions, error messages, code examples when necessary

# Writing useful Github issues

- Check out guidelines
  - Google: <https://developers.google.com/issue-tracker/concepts/issues>
  - Rust: <https://rustc-dev-guide.rust-lang.org/contributing.html#bug-reports>
- Don't assume the solution
- One issue per issue
- Keep titles short and descriptive
- Format your messages

# Mention or assign appropriate people



The screenshot shows a GitHub issue creation interface. The title of the issue is "Update game to use new rendering engine". The body of the issue contains the following text:

Now that we've decided on our new rendering engine (see #824), we need to update our collision logic to use the engine, build an engine prototype, and update the game logic.

A bulleted list of tasks follows:

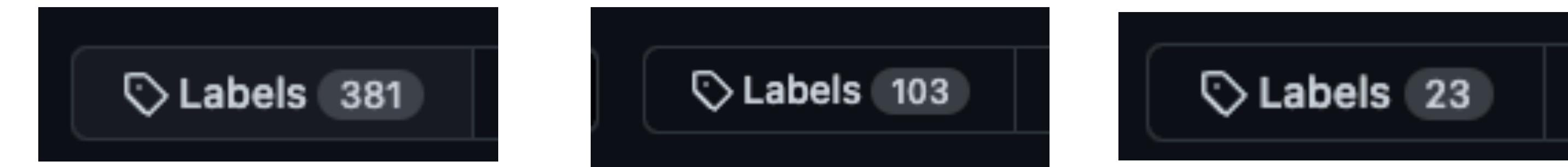
- [ ] #740
- [ ] <https://github.com/octo-org/octo-repo/issues/1752>
- [ ] Update aliens and cannon game logic

On the right side of the interface, there are several settings sections:

- Assignees:** octocat
- Labels:** enhancement, space game
- Projects:** None yet
- Milestone:** beta release
- Linked pull requests:** Successfully merging a pull request may close this issue. None yet

# You can use label

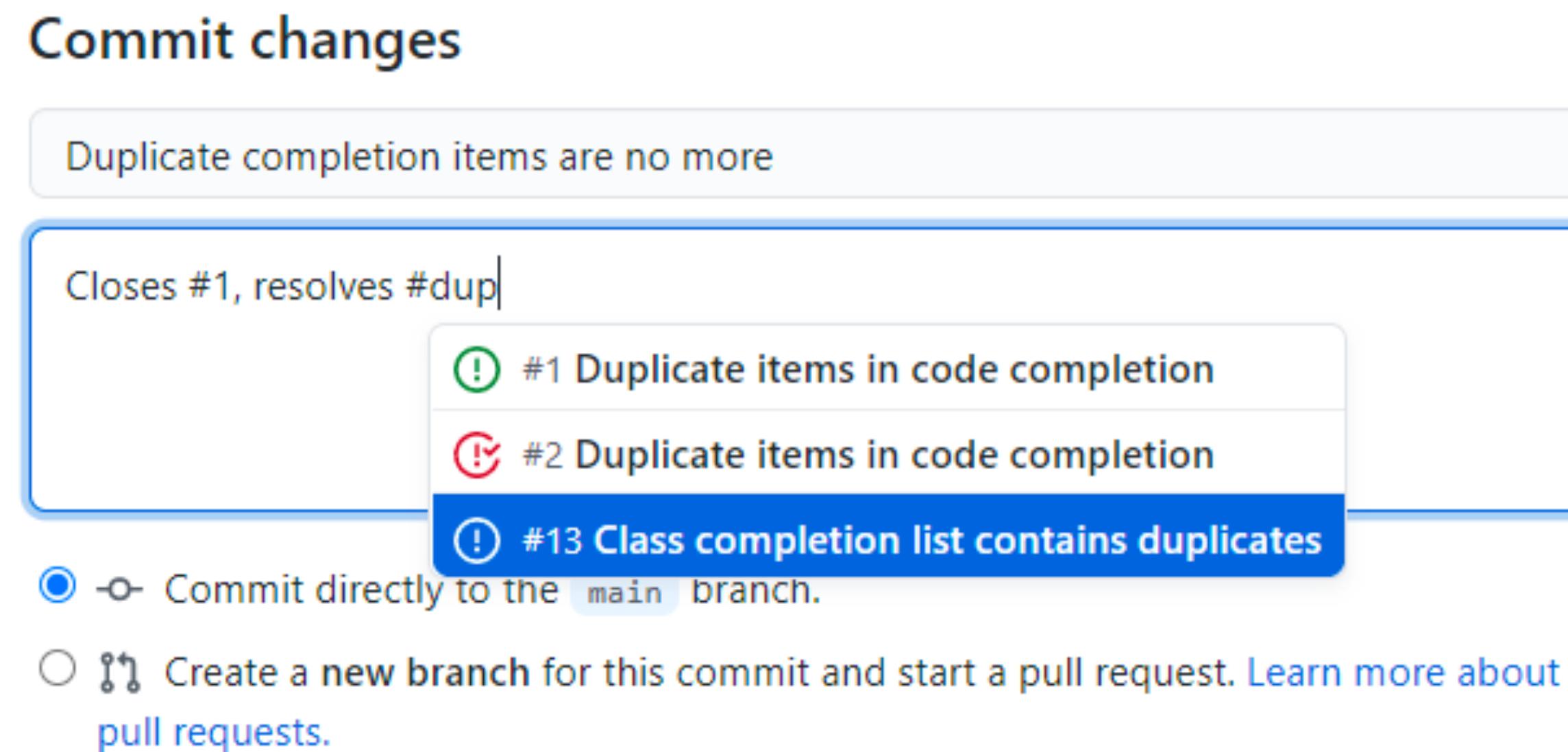
- Break the project down by areas of responsibility
- Mark non-triaged issues
- Isolate issues that await additional information from the reporter



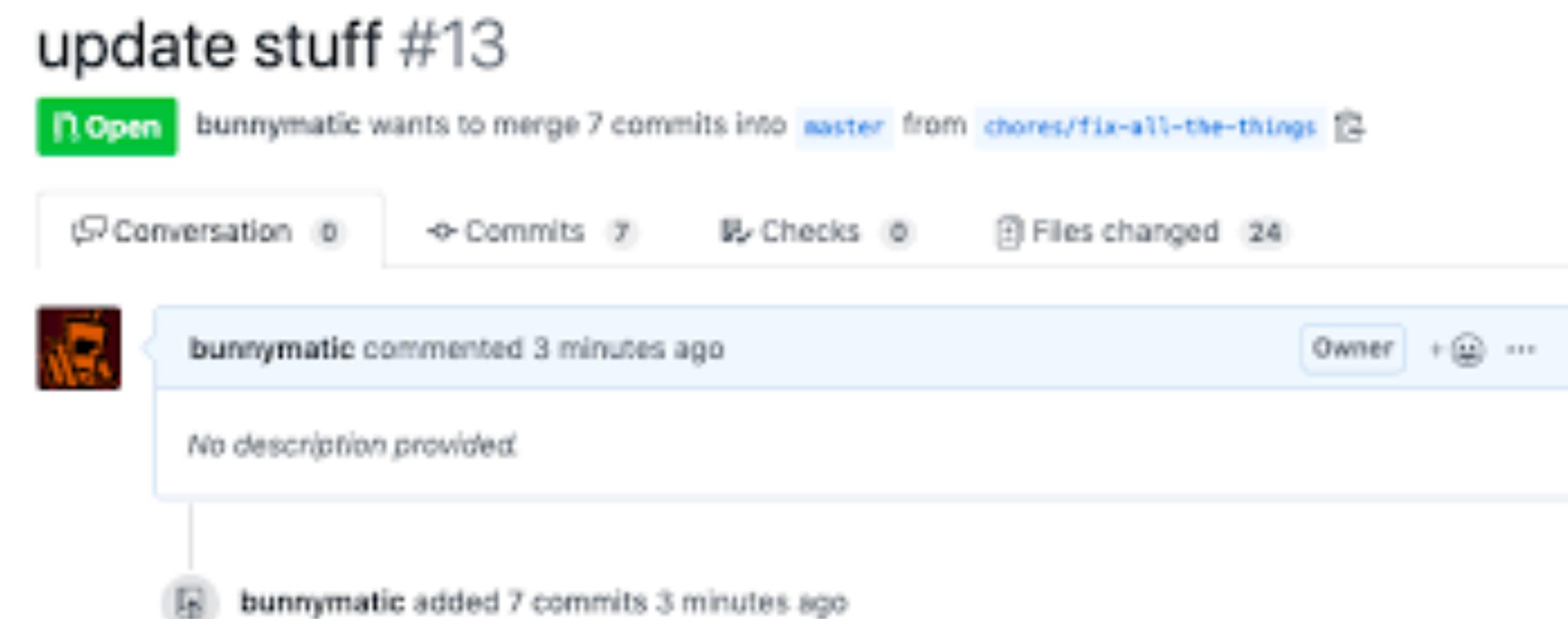
- Example:
  - Bug / Duplicate / Documentation / Help Wanted / Invalid / Enhancement
  - status: wip, status: ready to implement, status: needs discussion

# Don't forget to follow-up and close issues

- closes/resolves #issue\_number



# Pull requests



# How to write good pull requests

```
## What?  
## Why?  
## How?  
## Testing?  
## Screenshots (optional)  
## Anything Else?
```

## What?

I've added support for authentication to implement Key Result 2 of OKR1. It includes model, table, controller and test. For more background, see ticket #JIRA-123.

## Why?

These changes complete the user login and account creation experience. See #JIRA-123 for more information.

## How?

This includes a migration, model and controller for user authentication. I'm using Devise to do the heavy lifting. I ran Devise migrations and those are included here.

## Testing?

I've added coverage for testing all new methods. I used Faker for a few random user emails and names.

## Screenshots (optional)

0

## Anything Else?

Let's consider using a 3rd party authentication provider for this, to offload MFA and other considerations as they arise and as the privacy landscape evolves. AWS Cognito is a good option, so is Firebase. I'm happy to start researching this path. Let's also consider breaking this out into its own service. We can then re-use it or share the accounts with other apps in the future.

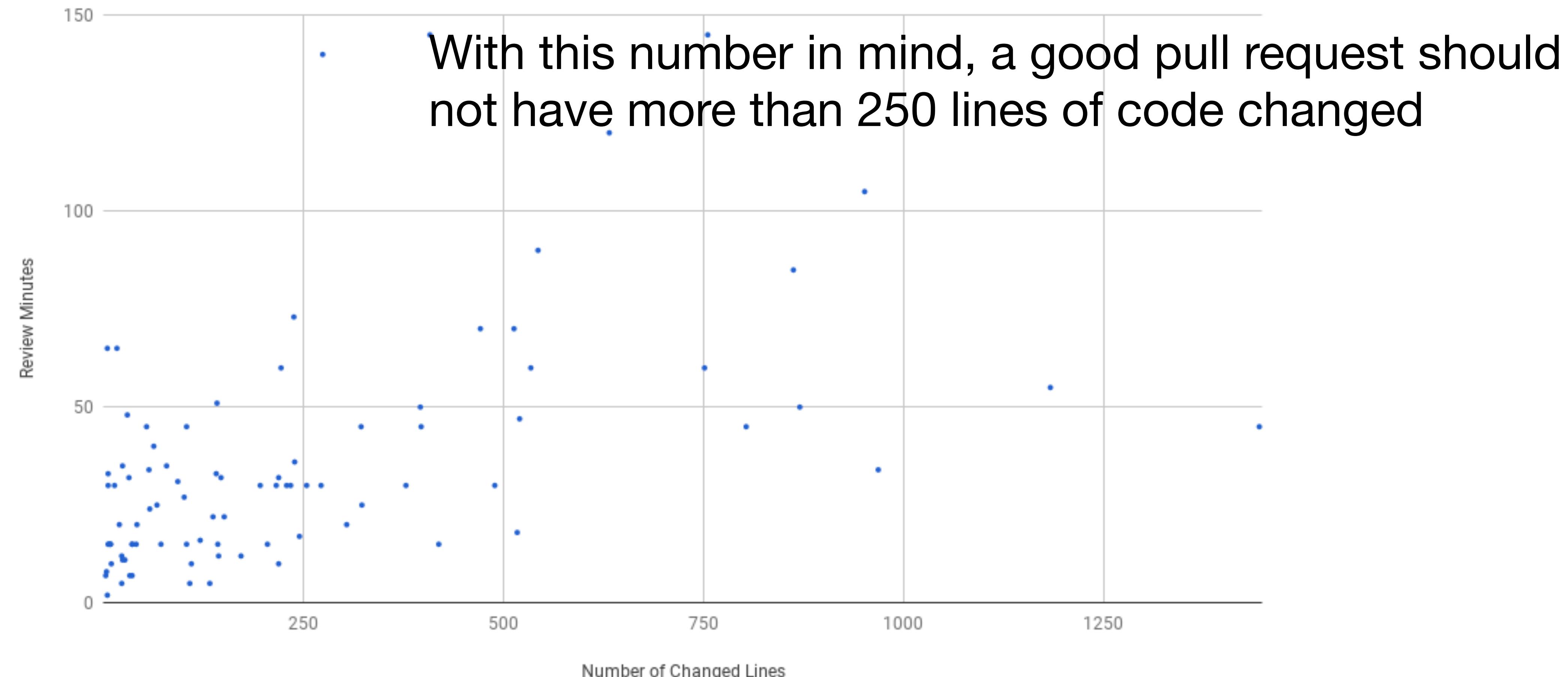
# How to write good pull requests

- Remember that anyone (in the company) could be reading your PR
- Be explicit about what/when feedback you want
- @mention individuals that you specifically want to involve in the discussion, and mention why.
- “/cc @jesseplusplus for clarification on this logic”

# Consider the size of PRs



## Relationship between Pull Request Size and Review Time

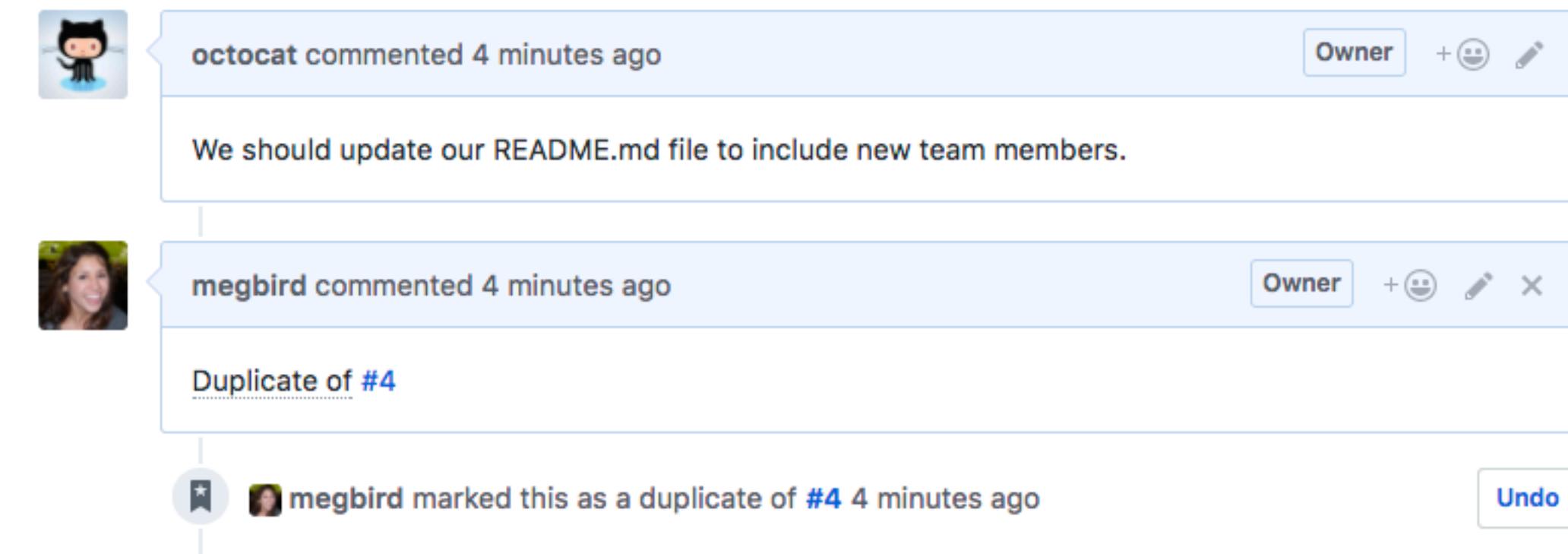


# Offer useful feedback

- If you disagree strongly, consider giving it a few minutes before responding; think before you react.
- Ask, don't tell. ("What do you think about trying...?" rather than "Don't do...")
- Explain your reasons why code should be changed. (Not in line with the style guide? A personal preference?)
- Be humble. ("I'm not sure, let's try...")
- Avoid hyperbole. ("NEVER do...")
- Be aware of negative bias with online communication.

# Avoid Duplicates

- “Duplicate of” issue/pull request number



# Be a nice person

**Date** Sat, 13 Jul 2013 15:40:24 -0700  
**Subject** Re: [GIT pull] x86 updates for 3.11  
**From** Linus Torvalds <>

 share 638

On Sat, Jul 13, 2013 at 4:21 AM, Thomas Gleixner <tglx@linutronix.de> wrote:  
>  
> \* Guarantee IDT page alignment

What the F\*CK, guys?

This piece-of-shit commit is marked for stable, but you clearly never even test-compiled it, did you?

Because on x86-64 (the which is the only place where the patch matters), I don't see how you could have avoided this honking huge warning otherwise:

```
arch/x86/kernel/traps.c:74:1: warning: braces around scalar
initializer [enabled by default]
    gate_desc idt_table[NR_VECTORS] __page_aligned_data = { { { { 0, 0 } } }, };
```

# **Knowledge Sharing**

# No matter the format, documentation is important

Building on top of others' work in a community-like way can be an accelerator, both in open source and in companies. Documentation often signals if a repository is reliable to reuse code from, or if it's an active project to contribute to. What signs do developers look for?

In both open source projects and enterprises, developers see about

50 %

productivity boost with easy-to-source documentation

**What the data shows:** At work, developers consider documentation trustworthy when it is up-to-date (e.g., looking at time-stamps) and has a high number of upvotes from others. Open source projects use READMEs, contribution guidelines, and GitHub Issues, to elevate the quality of any project, and to share information that makes them more attractive to new contributors. Enterprises can adopt the same best practices to achieve similar success.

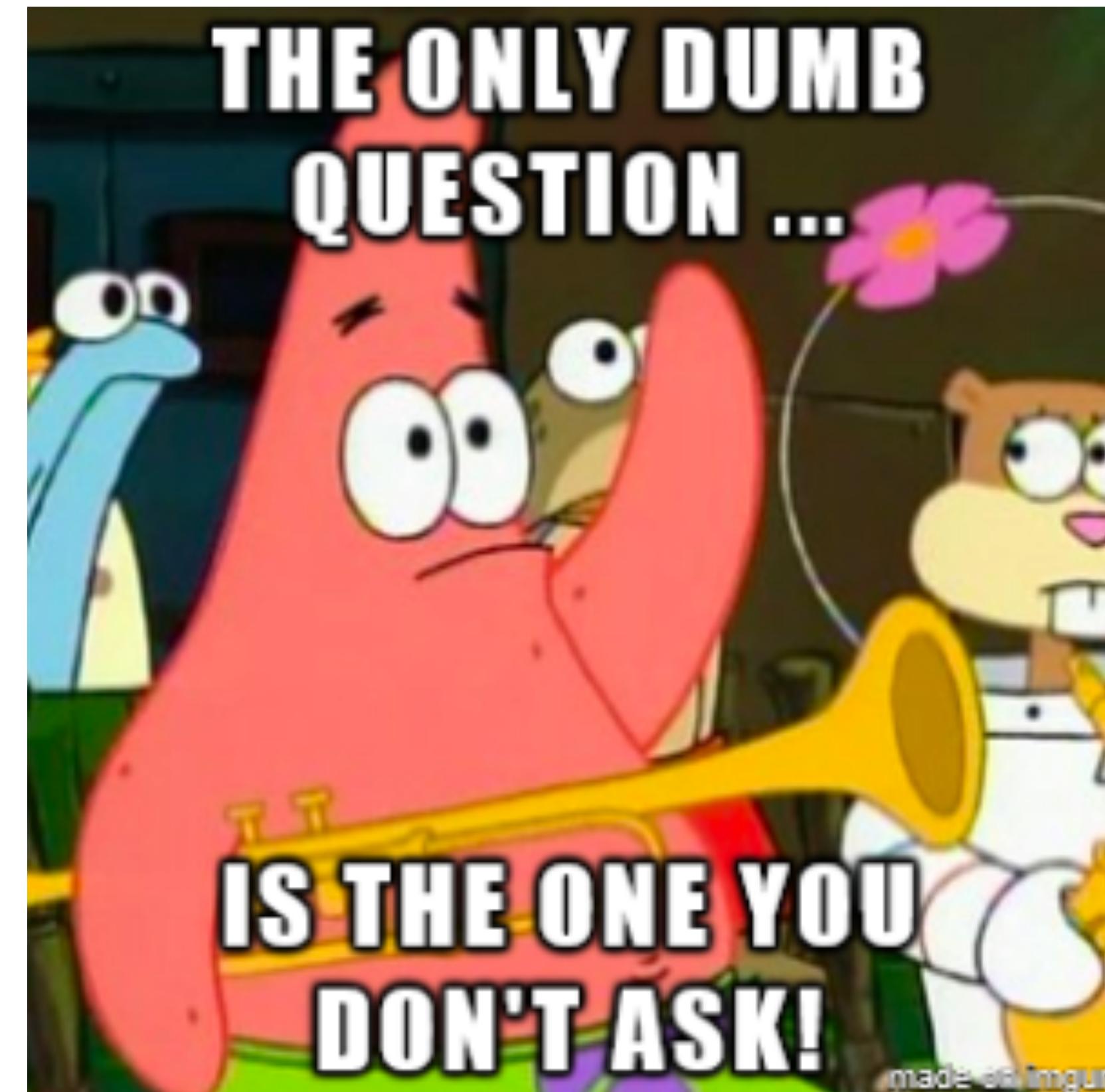
In both environments, developers see about a 50% productivity boost when documentation is up-to-date, detailed, reliable, and comes in different formats (e.g. articles, videos, forums).

**Using the data:** Review the documentation your team consumes: When was the last time it was updated? Can everyone on your team improve the documentation? Check this frequently to stay on track.

Knowledge Type	Description (Excerpt)
<b>Functionality and Behavior</b>	Describes what the API does (or does not do) in terms of functionality or features. Describes what happens when the API is used (a field value is set, or a method is called).
<b>Concepts</b>	Explains the meaning of terms used to name or describe an API element, or describes design or domain concepts used or implemented by the API.
<b>Directives</b>	Specifies what users are allowed / not allowed to do with the API element. Directives are clear contracts.
<b>Purpose and Rationale</b>	Explains the purpose of providing an element or the rationale of a certain design decision. Typically, this is information that answers a "why" question: Why is this element provided by the API? Why is this designed this way? Why would we want to use this?
<b>Quality Attributes and Internal Aspects</b>	Describes quality attributes of the API, also known as non-functional requirements, for example, the performance implications. Also applies to information about the API's internal implementation that is only indirectly related to its observable behavior.
<b>Control-Flow</b>	Describes how the API (or the framework) manages the flow of control, for example by stating what events cause a certain callback to be triggered, or by listing the order in which API methods will be automatically called by the framework itself.
<b>Structure</b>	Describes the internal organization of a compound element (e.g. important classes, fields, or methods), information about type hierarchies, or how elements are related to each other.
<b>Patterns</b>	Describes how to accomplish specific outcomes with the API, for example, how to implement a certain scenario, how the behavior of an element can be customized, etc.
<b>Code Examples</b>	Provides code examples of how to use and combine elements to implement certain functionality or design outcomes.
<b>Environment</b>	Describes aspects related to the environment in which the API is used, but not the API directly, e.g., compatibility issues, differences between versions, or licensing information.
<b>References</b>	Includes any pointer to external documents, either in the form of hyperlinks, tagged "see also" reference, or mentions of other documents (such as standards or manuals).
<b>Non-information</b>	A section of documentation containing any complete sentence or self-contained fragment of text that provides only uninformative boilerplate text.

# Know your audience

- Internal document for your team (e.g., meeting note)
- Documentation for project contributors
- Documentation for non-developer collaborators (e.g., UX researchers)
- Documentation for developer users
- Documentation for clients with no software knowldge
- User manual for end users



# [Activity] How to ask questions

## New To Coding. Can anyone assist me?

Asked 7 years, 1 month ago Modified 7 years, 1 month ago Viewed 47 times

I am trying to make a word counter and I just cant seem to get it. Can anyone help?

-4

```
import re
print("Welcome To This Software Made By Aaron!")
word = raw_input("Enter Your Words: ")
Check = 0
Right = 0
Length = len(word)
while True:
    if Right == 1:
        if Length < Check:
            Check = Check + 1
            print(Check)
    if Length == Check:
        Right = 1
print("Your Word Count Is " +Check)
```



# Make it easy for people to help you

I am trying to \_\_\_, so that I can \_\_\_\_.

I am running into \_\_\_\_.

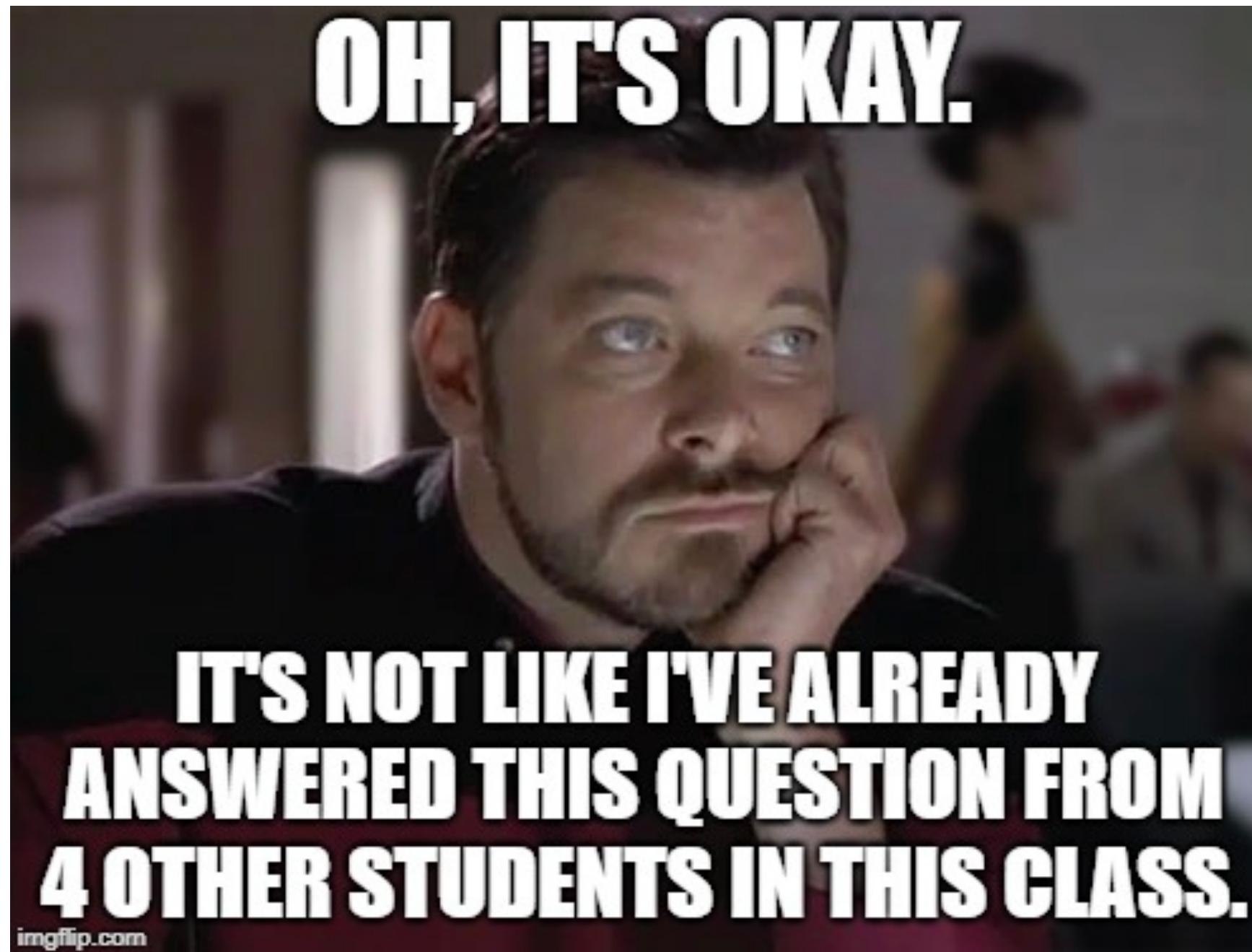
I have looked at \_\_\_\_ and tried \_\_\_\_.

+ I'm using this tech stack: \_\_\_\_.

+ I'm getting this error/result: \_\_\_\_.

+ I think the problem could be \_\_\_\_.

# Avoid Duplication



RESEARCH-ARTICLE



## Mining duplicate questions in stack overflow

Authors: Muhammad Ahsanuzzaman, Muhammad Asaduzzaman, Chanchal K. Roy, Kevin A. Schneider

[Authors Info & Claims](#)

[Published: 04 November 2015](#)

Studying the needed effort for identifying duplicate issues

[Mohamed Sami Rakha](#) , [Weiyi Shang](#) & [Ahmed E. Hassan](#)

[Empirical Software Engineering](#) 21, 1960–1989 (2016) | [Cite this article](#)

748 Accesses | 19 Citations | 1 Altmetric | [Metrics](#)

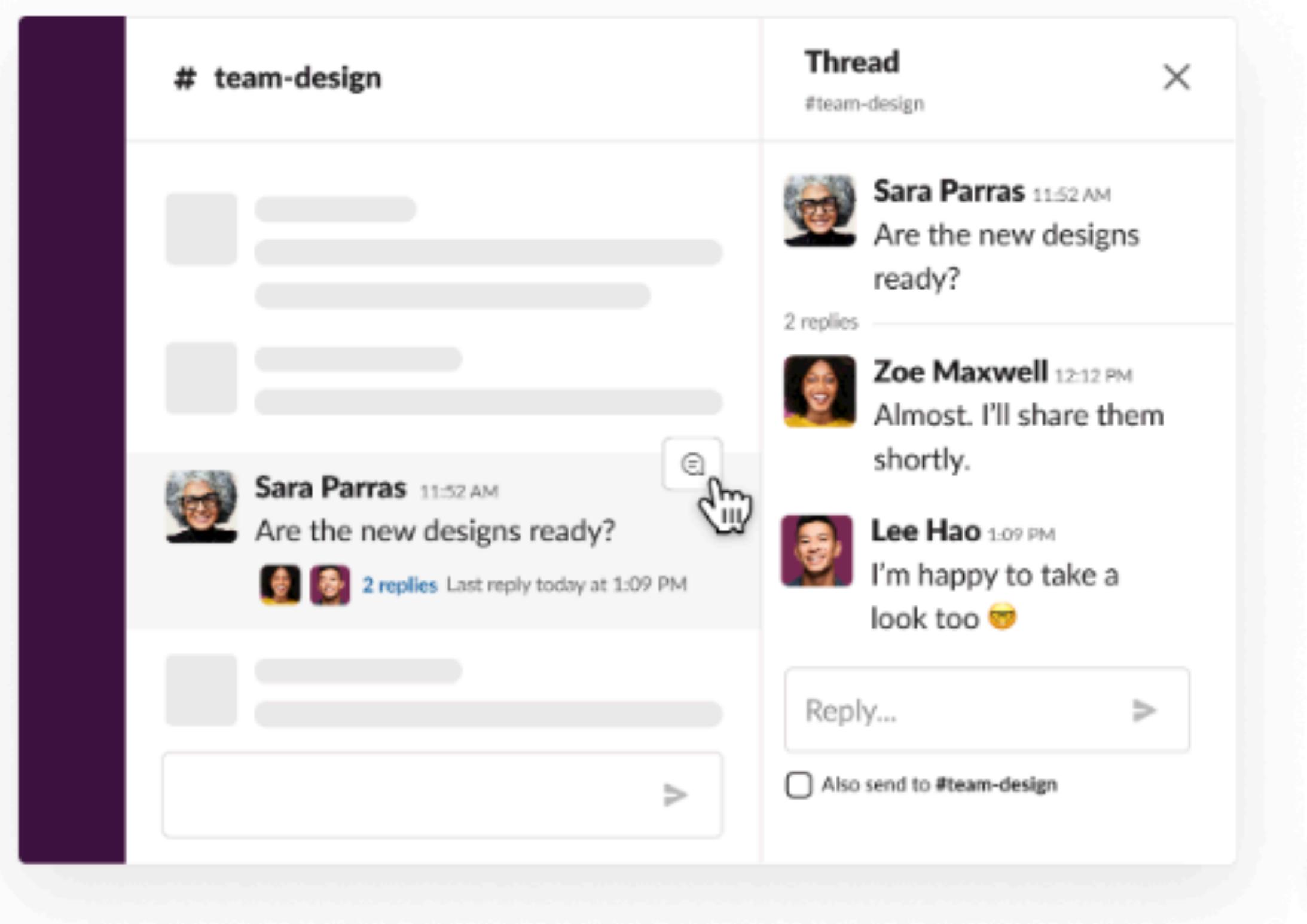
### Abstract

Many recent software engineering papers have examined duplicate issue reports. Thus far, duplicate reports have been considered a hindrance to developers and a drain on their resources. As a result, prior research in this area focuses on proposing automated approaches to accurately identify duplicate reports. However, there exists no studies that attempt to

# Avoid Duplication - Slack

- Add **quotation marks** to search a specific phrase
  - “Connection refused errors” will find results containing the entire phrase
- Add **from:** in front of a display name to search for information shared by someone specific
  - HW1 from:@Michael Hilton
- Add **is:thread** to search within threads
  - WSL is:thread

# Use threads



- Threads help us create organized discussions around specific messages, without adding clutter to a channel.
- You can manage thread notifications.

# Use channels properly

- ▼ Channels
  - # announcements
  - # general
  - # random
  - # technicalsupport
- + Add channels

- : Class / homework announcements
- : Administrative / logistics questions
- : Anything! Useful links, memes, ...
- : Technical issues (e.g., env setup, errors)

# Archive and share the answers

- Avoid duplication!
- You're probably not the only one who's wondering.
- For 313, post your questions in public channels if possible.
  - Feel free to answer too!
- For your team, create a team wiki (e.g., Github project wiki) or shared google document.

# **Resolve Conflicts**

# Escalation

- Make sure that everybody is working from the same set of facts.
- If an impasse occurs, find the person who can resolve.
- Ensure that every problem has an ultimate decider
- Goal: Find a resolution to the problem and move forward.
  - (Not “score points”)

# Team survey

RESEARCH-ARTICLE



## Identifying Struggling Teams in Software Engineering Courses Through Weekly Surveys

Authors:  Kai Presler-Marshall,  Sarah Heckman,  Kathryn T. Stolee [Authors Info & Claims](#)

SIGCSE 2022: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 • February 2022

- Pages 126–132 • <https://doi.org/10.1145/3478431.3499367>