Code Reviews at Medium

Small PRs, quick reviews, and working like we're in this together

Christine Donovan
Jul 12 · 5 min read ★

Photo by Harley-Davidson on Unsplash

Engineering teams have different norms and policies when it comes to code reviews. It can be frustrating to join a team and not know what is expected, or to operate with old assumptions and discover that they are at odds with a new team. In an effort to make it easier for engineers who come from different companies, we wrote down some of the features of our code review culture.

Note: We use Git and Github for version control, so when we're talking about code reviews, we're mainly talking about Github pull requests. Also, we have a tool that integrates Github and Slack, so many of our Github notifications come through as Slack notifications.

Changes require at least one approval

At Medium, engineers have to get their code approved by at least one other engineer before committing it to the codebase. We enforce this requirement with settings in our Github repository.

We have a handful of in-house scripts to standardize our Git workflow, and one of these scripts is responsible for creating a pull request. Most of the time, engineers use this script to create their pull requests. (If you are curious, this repo is the open-source version of these scripts.)

Reviewers

Our pull request script includes a list of the Github usernames for all the engineers at Medium. Faced with many options, who should an engineer choose as a reviewer?

In general, we try to to make sure that engineers are working in pairs or groups on projects, and one of the benefits is that engineers have built-in reviewing buddies. If an engineer is working on a project like this, it's natural for them to tag these people on their PRs.

If an engineer has been working on something in isolation, we encourage them to tag engineers from their team, who are more likely to be aware of the work they're doing. If an engineer has been working on something that falls outside of their team's domain, we suggest that they do a git blame to see who's recently worked on the same files, and get their perspective.

In some cases, other people are automatically assigned as reviewers, in addition to the reviewers that an engineer chooses. We have a few configurations that will auto-tag certain people in certain files. The key thing to know is that the auto-tagging is designed for increasing visibility versus getting approval. The rule stated at the beginning still applies: code only needs an approval from one other engineer (any engineer).

How many reviewers should be assigned? I've seen this vary. I would say the average is two or three. But I've also sent PRs to a single person, and sent PRs to my entire team (~6 people).

Help me help you

We don’t have hard requirements for adding descriptions to pull requests. We only require that: if you are making a change that affects the UI, include a screenshot of the UI.

That said, anyone who is reviewing code will need to figure out what is changing and why it’s changing. To facilitate this, engineers do often add descriptions, either in the PR itself or in a comment. In general, we think it’s good to consider what people may or may not know about your PR, and aim to give them context so that they can help you.

We also think it’s useful to look over your pull request, and see if there are places where someone might ask, “Why did this need to change?” An engineer can consider adding preemptive comments in Github (or updating the code if that feels called for).

Quick reviews

Our engineering team values and encourages the combination of small PRs and quick reviews.

In this section, I’ll talk about what I mean by quick reviews. In later section, I’ll talk about what I mean by small PRs.

Most of my PRs get reviewed within a couple of hours. We recommend that a PR should be reviewed within 4 hours. This guideline doesn’t mean that every reviewer has responded within a few hours; it means that at least one reviewer has responded within a few hours.

Note: this only applies to PRs that are submitted in working hours. We don’t expect (and don’t want) engineers to be reviewing code outside of working hours.

PTAL

But what if code doesn’t get reviewed within a few hours? Or, maybe, it did get reviewed, but there was feedback to address, and now it needs a second pass from the reviewer.

In both situations, an engineer will add a comment to their PR that says “ptal” (the lightning bolt emoji will work as well). PTAL stands for Please Take A Look or Please Take Another Look. When this comment is added, our Slack integration tool will send a notification to all reviewers. The notification will tell them that the author is asking them to “please take another look.”

We use PTAL frequently, and it’s understood way to let other engineers know that a review is needed (again).

Beyond PTAL

Maybe an engineer wrote “ptal,” but there’s still no responses on their PR. Or, maybe this code is especially urgent, and it needs attention sooner that the average PR.

In general, we think of the author as being responsible for getting the review that they need. We ask engineers to remember that other engineers want to help and be responsive, but some days and some workloads don’t allow for it. That’s why it’s good to escalate when someone is sensing that their PR has slipped through the cracks.

One example of escalation is dropping the PR in a Slack channel and asking if anyone has time to review. Another example is pinging reviewers directly. We encourage engineers to figure out the best way to escalate a PR that needs attention.

Paradoxically, small PRs > big PRs

At Medium, we strive to build things in small units. We want to break our code down as much as possible and submit small, digestible PRs. By doing this, we can review quicker and move faster.

Okay, but what is a small PR? One way to think about it is in terms of conceptual changes, not lines of code. For example, changing the name of a variable might cause changes in many files, but it is still small because it is a single concept — updating the name of a variable. We try to minimize PRs in terms of conceptual changes.

We try to see things from other people’s perspectives

We have an unofficial company motto of “safe to try.” This is motto is about feeling free to make a call without needing consensus or certainty (as long as it won’t be really bad). I think this motto influences our code review culture as well. When we’re reviewing code, we don’t act like gatekeepers; we aim to help our co-workers do what it is they’re trying to do.

We also value the Reasonable Person Principle. It’s pretty self-explanatory; it’s about starting with the assumption that another person is acting from reasonable intentions. This principle reminds us to see each other as humans first, and be generous in our interpretations of one another.

Software Development Code Review