Why you need a "WTF Notebook"

There's a very specific reputation I want to have on a team: "Nat helps me solve my problems. Nat get things I care about done."

Photo by Glenn Carstens-Peters / Unsplash

Nat Bennett

Sep 24, 2021

Software

I keep a bullet journal. I'm not one of those people you see on Pinterest with the fancy spreads – I mostly just use black ink, the standard setup, and the occasional custom collection.

Every time I join a new team, I go to the next fresh page, and on top of that page I write: "WTF - [Team Name]." Then I make a note every time I run into something that makes me go "wtf," and a task every time I come up with something I want to change.

For two weeks, that's all I do. *I just write it down.* I don't tell the team everything that I think they're doing wrong. I don't show up at retro with all the stuff I think they need to change. I just watch, and listen, and I write down everything that seems deeply weird.

This is a trick I picked up from a team lead a few years ago, who learned it from a previous lead of his in turn. It's one of my most powerful techniques for making changes on a team, and managing myself while I do it. So I'm going to walk you through how I use that list, and how it helps me to build a reputation as someone who's really effective at getting stuff done, and avoid being someone who's complaining all the time.

There's always stuff that makes me go "wtf" on a new team. The team talks for an hour in retro about a serious problem, and then leaves without making any action items. The tests don't run locally and no one seems to notice. Big chunks of the build board are always red. Only one person can do some critical, time-sensitive thing. The team is spending a bunch of time on some feature, but when I ask around no one can seems to know why it's important or how it'll help a customer.

Once I've got a nice big list, I start crossing things off. There are four reasons at this point that I might cross off something I've put on that list:

1. There's actually a good reason for it
2. The team is already working on a fix
3. The team doesn't care about it
4. It's really easy to fix

If the tests don't run locally, for instance, that might be a known issue that there's an ongoing effort to address. The team might do all of their work on virtual machines, and have a simple chat command that provisions those machines for them. Or they might have a pretty good continuous integration system and good habits around making small changes, so not being able to run the tests locally isn't stopping them from deploying multiple times a day.

Sometimes, it'll turn out that there's a really simple fix for some of the things I've identified. Maybe there's some documentation I can write, once I know where it is, or maybe there's an easy change once I find the right scripts. That's not always immediately obvious when I first see a problem. When I do see an easy fix, though, I'll just go ahead and make it.

After a few weeks, though, I'll still have a bunch of weird, unresolved issues on that list. At this point I'll start talking about it with other people on the team, the team lead, and my manager.

I'll ask why things on the list are that way, and how they got to be that way. I'm trying to establish credibility as someone who's genuinely curious and empathetic, who's patient, and who respects the expertise of my coworkers. That's the reputation that's going to let me make changes later.

Generally, I'll find out that the things that problems I've noticed are around for one of a few reasons.

1. The team hasn't noticed it
2. The team has gotten used to it
3. The problem is relatively new, and the old problem it replaced was much worse
4. They don't know how to fix the problem
5. They've tried to fix the problem before and failed

On a lot of teams, when I ask some questions about things that turn out to be in the first few questions, the person I ask will just fix them immediately. Or they'll help me figure out how to fix them. If it's a technical problem, that means writing a story or a ticket together, and then we'll work on it. If it's more process or social, it means bringing the problem up at retro and talking about it with the whole team.

At this point I'm looking for one or two problems that have been bugging one of my new teammates for a while, and that have relatively simple solutions. I'm looking for something I can put on the retro board and know I won't be the only person who's bothered by that problem. Then, during the team conversation about the problem, I'll identify something that teammate suggests as an action item that we could try immediately. That way the team starts to see me as someone who helps them solve their problems.

The feeling that I want to create, the association I want people to have with me, is, "Oh, Nat joined the team and little things started to get better, almost immediately. It feels like we're starting to make some progress. And it's not like they showed up and started telling me what to do, either. They're really listening to me, they're helping me explain myself to the rest of the team."

Pretty soon, I'll start to get in to the really sticky issues. The problems the team knows about but is afraid of dealing with. The things that aren't "that bad," but that no one wants to talk about. Maybe they're missing the technical skills to deal with the problem. Maybe there's a knotty people problem at the center of it.

At this point I'm going to be talking to my manager. I'm going to bring them that list I've been working on, and I'm going to say something like, "Now that I've been on the team for a few weeks, this is what I'm seeing. We're making progress on some of it, but some of these seem like they're going to take longer. I wanted to get your thoughts before I try to do anything about them. Is there something I'm missing? Is there a particular area I'd like you to focus?"

The reaction I'm looking for from my manager, at this point, is something like, "Wow. This is really validating. I've been concerned about these things but the team doesn't seem really bothered by them, so I didn't want to push too hard. I'm glad you're bringing this up."

Then we can have a conversation about what their concerns and problems are. I can do some reflective listening to help them organize their thoughts, and I can talk about what I've seen work well, or not, in the past. They'll start to see me as someone with good judgement, and someone they can come to for help solving their harder problems.

There's a very specific reputation I want to have on a team: "Nat helps *me* solve *my* problems. Nat get things *I care about* done." That's the reputation that's going to get me the results I want in next year's performance review. That's the reputation that's going to get me a referral a few years from now.

Before I started keeping this kind of list, I brought up problem I saw immediately, as soon as I noticed it. The reputation I got was, "Nat's always complaining about things. Nat thinks we're never doing things right." People stopped listening to me. I was personally frustrated, and professionally ineffective.

There's no faster way to totally sink my credibility, as a new team member, by making a huge fuss over something that's not a problem, or that the team doesn't see as a problem, or that there's already an effort to fix, or that there's a really simple way to fix that I just didn't see at first. There are always so many problems on a team, so many things that could be better, that I'm only ever going to solve a handful of them. Working on problems in the order I noticed them is rarely the most effective order. So the WTF Notebook gives me a place to park the impulse to *fix it now, damn it!* until I have more context for deciding what to work on first.

Instead, for two weeks, I just write things down.

Jobs

Periodic reminder that Code for America is usually hiring, and they pair and write tests. Until the end of this month they have a Software Engineer role up for a team that works San Francisco hours. If you're looking for a "show up, write code, go home" experience, and want to help Americans access food stamps and other safety net services, this is a team that can deliver it – especially if you have some experience with Rails or Spring.

If, on the other hand, you're interested in gnarly cloud infrastructure and software problems for the Department of Defense, check out Rise8. If you've heard about Kessel Run, or Pivotal's work with the Air Force generally, Rise8 is where many of those folks ended up. They also practice design thinking, test-driven development, and continuous deployment, but they're teaching them to folks who have never used these practices before, and pairing with military service people. Their job listings mention experience at Pivotal Labs by name.

If you've got an active job search running and you're struggling to keep track of it all, check out Davis Frank's guide to Job Search Journaling with Obsidian.

Job listings from previous issues

Reading

I've mentioned *Seeing Like a State* before but I reread it while we were on the road, and, and, man, seriously, if there's one book I wish everyone I talk to had read, it's this one. Nothing explains systems thinking in action better. Nothing has more useful anecdotes for illustrating how large organizations work, and why they work the way they do.

The other book I've read by James C. Scott is *Against the Grain*, and if you're at all interested in the history of the earliest states and the initial development of human civilization, that book will absolutely blow your mind.

Ed Zitron's piece recently about How Our Need For Attention Online Drives Us Crazy articulated a bunch of half-formed thoughts I've been chewing on and trying to figure out how to write about. It doesn't mention Slack explicitly, but I've seen Slack drive a lot of these same processes at work.

← Previous

Next →