Twitter Thread by A Brian For All Four Seasons



A Brian For All Four Seasons
@bguthrie



So, here's my contrarian take on Why Trunk-Based Development Is Great But Probably Won't Work For You. Buckle up, it's a long one.

From time to time I talk about, or boost someone else's discussion of, "classical" continuous integration/trunk-based development/push-to-master/whatever. I do this because I've personally worked like this and I genuinely think it's a marvelous way to write software.

I've also, at this point in my career, failed—a bunch of times!—to bring this technique into practical use on the teams and organizations I've worked with. As a result, while I still do it myself, I don't focus as much on encouraging it in others.

The technoculture that's presently in vogue is what I think of as async/individual—optimize for the individual, performance-manage the individual, decouple the individual from others so they can be their best, most productive, most heroic selves.

This is in contrast to what I think of as sync/team—a culture that leans into interdependence and collaboration. If you've never worked on a team that collaborates very closely every day for months or years on end (and likes it) you've likely not experienced a team like this.

There are a bunch of reasons why this technique failed to gain broader traction, but briefly (1) difficult to achieve (high-collab work was never very popular) (2) tooling has closed the gap between good async and great sync teams (3) dual pressures of COVID + tech salaries.

Before I go further, the technique I'm describing involves devs working in pairs or mobs, pushing small atomic commits directly to the main branch, and those commits being deployed more or less immediately—in contrast to individuals working in branches.

What makes it difficult? It requires new skills, technical process changes, and organizational buy-in. These tend to be difficult to roll out all together. In no particular order:

Pairing + review discipline, which includes:
* pairing skills
* schedules built around pairing

* atomic commit practices

Cycle time discpline, which includes:
* sync work over async work
* WIP limits
* adequate feedback loops
* "just in time" planning

Build discipline, which includes:
* local pre-commit builds
* fast builds
* reliable builds
* trustworthy test coverage
* keep-it-green practice + rapid rollbacks

CD & observability discipline, which includes:
* robust feature toggles
* robust observability
* rapid deploys
* robust continuous deployment model

And finally, it almost goes without saying, but: Trust and courage, which includes:
* A mutual-support & learning culture
* Trust in others
* Rapid failure recovery
* Openness to risk-taking in approach

One could meaningfully write a book about each of of these. Indeed, many such books exist.

These disciplines require, at best, hands-on upskilling from disciplined practitioners to get right, and that's if—IF!—your team is bought in. More likely you're swimming upstream against a bunch of contemporary trends—most notably, the move to more async work.

At worst you're looking at months or years of work to get them right, all the while dangling out on a limb and trying to make a case to the organization that it'll be worth it—someday.

Conversely, many of these disciplines aren't necessary or interesting in an async/individual work culture. Once you've decided to make your builds async, you tend not to feel as much immediate pain if they're flakey or slow.

And who cares if a build runs locally? Who's hacking on a plane these days anyway, with or without onboard wifi?

Even if you decide to become an advocate for these (significant) changes, you may ultimately find that this was never the bottleneck worth optimizing for—that the problem lay in product or leadership or finance or a half-dozen other places.

So let's say you want to do it after all. How do you achieve it?

1. For an existing team, it has to come from within but it's probably not going to arise spontaneously—i.e. you're not going to retro your way into doing it. It requires either an internal champion to take a risk, or an external hire with chops, or both.

2. It's unlikely to happen without practical hands-on experience doing it. So, a hire (again) or a coach. This is where management/leadership fiat has failed for me personally.

3. It has to demonstrate real improvements over the alternative—from a sustainability, velocity, and happiness standpoint—or it'll decay. Doing it takes discipline, and discipline is hard to sustain.

4. It requires, at minimum, buy-in from leadership up the chain, particularly in environments with a lot of interdependence—monoliths or monorepos, which will also often share a common method of change management.

5. It requires settling clear expectations with new hires: that you're a pairing culture, that you're a courage culture, that they'll be expected to try things that may feel uncomfortable. If you don't set expectations up front, that discomfort may turn into attrition.

6. Tighter upfront expectations will likely narrow your hiring funnel, which your recruiters may be uncomfortable with.

Lots more to write about this, but that's me for today.

It almost goes without saying, I think I implied but omitted it above: pairing remotely is hard. Tight collaboration is much easier when you're colocated.

Also, to be clear: please understand this thread to be an attempt to understand + explain my own failures in leading through this change and its implications, and not of any other person or team I've worked with.

To be clear, I say this not out of some kind of normative judgment but because I think there simply aren't many teams that work like this. There are also plenty of people who've worked on teams like this but disliked it, and I respect that.

https://t.co/cVc4gUWYmO

> This is in contrast to what I think of as sync/team\u2014a culture that leans into interdependence and collaboration. If you've never worked on a team that collaborates very closely every day for months or years on end (and likes it) you've likely not experienced a team like this.
>
> — A Brian For All Four Seasons (@bguthrie) January 4, 2021