Git branching for agile teams

Moving to Git opens up a whole new level of agility for software teams – here's why

BY SARAH GOFF-DUPONT

Freed from the clunky code freezes and monolithic mega-merges that plague centralized version control, developers can isolate work in progress and build in narrow vertical slices with ease. Branching and merging are so painless with Git that many teams are making new branches for each user story or bug fix they implement. This model is quickly becoming the new gold standard for agile teams – and for good reason!

Grab some popcorn and sit back with one of our most popular webinars ever. You'll learn:

- How a branch-per-issue model helps teams deliver working code in a continuous stream
- What the workflow looks like for developers
- How it integrates with your existing continuous integration and code review practices
- Trade-offs to consider when evaluating this model

RELATED TUTORIAL
Learn scrum with Jira Software
Try this tutorial →

SUBSCRIBE
Sign up for more articles
Email
email@example.com
Subscribe

Watch & learn

Q & A

Good stuff, right?

Now, if you're like me, you rarely sit through the Q&A portion of a webinar. It's ok, you can admit it. So I transcribed a handful of questions for you to scan through and read at your leisure.

Q: How you deal with version numbers in all these branches? How do you differentiate those code lines?

A: Teams typically name the branch after the corresponding release version. For example, when the team that makes Stash was getting their 2.9 release ready, they created a stable branch named 'stash-2.9', so when they see it in their repo, it's very clear what that branch is all about. You can use whatever naming convention works for your team – just include the version number in it somewhere.

Q: In your examples, you've had one person working on each branch. What's the right branch structure and naming convention for when two people are working on a story?

A: You can have two people working on the same issue branch. Just make sure to follow basic shared branch etiquette such as not rebasing, and generally avoid doing things on your local copy of the branch that will create headaches for the other person. Another option is to have each collaborator create their own branch for that issue, then merge them together frequently. The naming convention for that could be -- (e.g., sarah-DEV-1234-company-name-misspelled-on-homepage), which is more or less the same as we use for single-developer branches.

Q: How do you handle dependencies if you're not merging every change into a single branch as soon as it's made on

a development branch?

A: If the dependent pieces are both works in progress, then have the developers working on each piece of code merge their changes together frequently. And you can do this in Git without having to merge either branch to a centralized branch first – you and the other developer can just merge your branches directly. The other option is to use the shared integration branch we talked about earlier.

Q: A while ago, Martin Fowler argued against feature branching, saying that it hinders refactoring and that while the feature branch is alive you're doing continuous building but not continuous integration.

A: Yep, I'm familiar his post on that subject. It was quite a while ago – maybe 2008 or 2009. I think that our opportunities for running CI on feature branches have advanced quite a bit since then. And as I said in the "considerations" part of this webinar, a branch-per-issue approach does mean that you're not doing pure CI. You are doing continuous building, and continuous testing, but not continuous integration. You can get close to pure CI, however, by including a shared integration branch in your branching scheme.

SARAH GOFF-DUPONT

I've been involved in software for over 10 years: testing it, automating it, and now writing about it. When not at work, I can be found reading contemporary fiction, smashing it out & keeping it real at CrossFit, or rolling around on the floor with my kids. Find me on Twitter!@DevToolSuperfan

Up Next
Code reviews →