Three ingredients for great software releases

Combine one part architecture with two parts teamwork. Add automation, and stir.

BY DAN RADIGAN

At some point in your career–if you haven't already–you'll be involved with a monolithic software release. That is, a release with recalcitrant bugs and interdependencies, and one that requires the entire team on deck around the clock. Not to mention that once it goes out to production, it'll likely require several patches.

Shipping code–the release–is a strong barometer of agility for software developers. Every effort to make planning, coding, and testing faster are for naught if the release isn't a smooth process. To make the release an agile affair, deploy automation is key, as is bringing coders and operators together early in the development phase, practicing continuous integration, and addressing defects immediately.

Keeping the code in a releaseable state is the hallmark of agile development. All the lean planning and iterative development in the world won't mean a thing if you can't ship code the moment you decide it's ready.

Great software releases start with modular architecture

In any software program, it's best to release easily and often. A team can make release a natural part of their agile culture by building (or refactor towards) a modular architecture. Rather than have one large application (like the monolith mentioned above), early on in the program *modularize it into several pieces*. Group similar features into smaller applications or components, and have clear API contracts between each of the applications and components. Those APIs can be tested automatically with every build to ensure compatibility and reduce risk in the software release.

A modular architecture means that you don't have to release the entire software stack in a "big bang style" release, and the API contracts make it easy to update components and ensure compatibility between versions. In a nutshell, modular releases require fewer moving parts. And that translates into simpler releases.

Great software releases are powered by great relationships

Software development is seldom done in a vacuum. Indeed, *great software development* involves the entire team from product management to operations. For example, the operations team is a key partner in delivering software to production since they help the software reach end users.

Development teams can help inform and empower operations teams with these techniques:

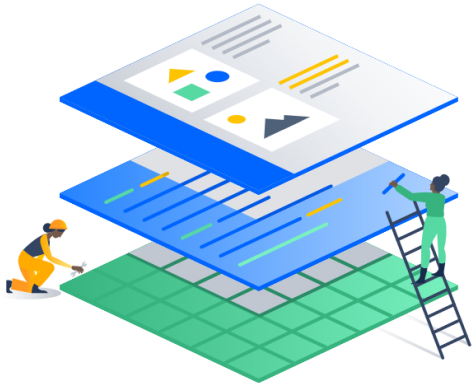
- Make the bill of materials for each release clear. Operations teams don't always have the same level of context around the release as the development team does.
- For each issue that's resolved in the release, provide a link back to your issue tracker and source control system so the ops team has the same level of context if problems arise during the deployment.
- Sometimes issues appear when pushing code from the development environment to the staging environment. Call these issues out, as they may pop up again during the production push.
- Deployment glitches happen, so always give the operations team a clear escalation path to resolve problems smoothly.

Operations teams can assist their counterparts in

development with these suggestions:

- When problems arise in production, take the time to understand root causes and solutions. They'll be avoided (or handled more gracefully) in the future.
- Migrate configuration data from production back into staging and development environments to prevent configuration drift.

As code migrates from development to staging and out to production, key configuration and user data migrates just the opposite way: production to staging to development. Having that bidirectional relationship helps the development environment closely model the production environment. This means fewer bugs and surprises on release day.



Great software releases are simple to push

Automate! Automate! Automate!

Automating a release is the best way to improve a release culture. If releasing is not automated today, start by automating the release to a staging environment. Once everyone sees how simple it is, the natural step will be to automate production deploys as well.

If releases are difficult, make it a practice to release frequently–even if it's just to staging. Having the development team feel the pain points of release will inspire innovation to make it easier (and automated).

Automated testing and continuous integration are key disciplines that power great releases. Ensure that build times and testing times are as short as possible, and remember that builds that are easy to validate are easier to release. That's because the validation cycle more closely follows the team.

Great software releases are great!

Keeping the code in a releasable state is the hallmark of agile development.

> All your lean planning and iterative development won't mean a thing if you can't ship quickly.

> HOW WE DO IT
>
> **We find small, frequent releases easiest to manage for our SaaS properties. For downloadable products, close collaboration between development, operations, and build engineering teams goes a long way. These groups should work together to automate releases, and proactively adapt the automation to upcoming changes to the products. Many of Atlassian's teams automatically deploy each successful build of main to a test environment. When it's time to promote a release to staging, or release it out to customers, these teams can trigger the deploy automation with the push of a button.**

As software developers, release should be the highlight of our innovation cycle. We get to see customers interact with the code we've written and provide feedback. Yay! Making releases a natural part of your workday makes it easier to get code out to production and enjoy the satisfaction of saying: "That's my code!"

SHARE THIS ARTICLE

DAN RADIGAN

Agile has had a huge impact on me both professionally and personally as I've learned the best experiences are agile, both in code and in life. You'll often find me at the intersection of technology, photography, and motorcycling.

TUTORIAL

Learn versions with Jira Software

Learn how to use versions to organize your work around milestones you can aim for. Learn to assign issues in your project to a specific version.

Try this tutorial →

ARTICLE

Journey to a stress-free software release

Great software releases start with modular architecture and are powered by great relationships. Learn how to have great agile software releases here.

Read this article →

Agile Topics

Agile project management
Scrum
Kanban
Design

Software development
Product management
Teams
Agile at scale
DevOps

Sign up for more agile articles and tutorials.

Email

email@example.com

Subscribe

Up Next
Stress free release →