Journey to a stress-free release

This video guarantees to reduce stress levels before your next blockbuster release

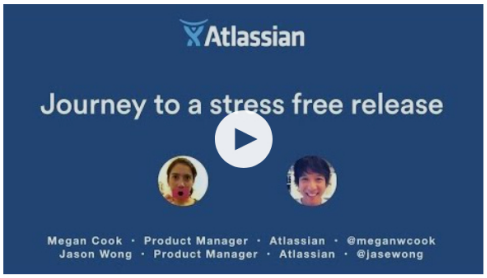
BY LAURA DALY

An agile team's best measure of success is when working software is released to customers. But even experienced software teams feel pain when it's time to validate completed issues against artifacts; code reviews have been skipped, code hasn't been merged, builds for merged code fail... Sound familiar?

In this presentation you will learn:

- **Coding best practices that will improve your ability to deliver quality product.** Hear why code reviews are essential to delivering quality, and how monitoring and fixing failing builds will guarantee faster time to release.
- **How to set up and maximize Jira Software's release hub.** Learn how to save hours of work by allowing the release hub to provide a clear picture into the progress and status of a release.
- **Complete automation from build, code, to release.** Streamline your workflow by releasing your version straight from release hub.

Watch & learn



Q & A

Our hosts Jason Wong and Megan Cook picked their favorite Q&As from this presentation. Read on to learn more about how to have successful and stress-free release.

Q1: What are the 3 top non-technical factors that make using release hub successful?

A1: (1) Ship with confidence: Stakeholders inside and outside of the team will be able to know what exactly is ready to be released at anytime throughout the release cycle.

(2) Save time, make decisions faster: Automatically and instantly know what features are done and ready to be shipped, and which have problems. release hub checks through all you issues in your version for you, so that you don't have to manually reconcile issues and code.

(3) Record of releases: Know what went out (when & which release) by looking at released versions, and what is currently planned for each of the upcoming releases by looking at unreleased versions.

Q2: Who generally manages release versions at Atlassian?

A2: Each team within Atlassian has it's own specific approach, but in general, we try to automate as much of the process as possible, so that putting out production bug fixes or releasing a new bug fix version of a product can be done by any developer in the team with minimal overhead.

Teams typically have a list of things that need to be done, but we try to minimize this as much as possible. Tools like the release hub help in this process to make sure that what we are planning to release is of the highest quality and

there is no mismatch between the states of the Jira Software issues and their actual development.

Some of the larger teams (e.g. Jira Software and Confluence) will actually have a dedicated rotation for a release manager who manages all releases.

For larger Major and Minor releases, there is generally a dedicated person who looks after the release, making sure the scope is being managed and all work leading up to the release is managed for risk and time. This may be overseen by a Team Lead or Development Manager, but we try to make sure this rotates amongst team members so that everyone knows the process and understands the requirements of releasing quality software.

For timeline planning, the Team Leads will coordinate with the product managers and marketing about setting expectations for when a release will be ready and if scope or time needs to be managed. It is amongst these people that decisions are made on which features will be released in what versions.

Q3: How do you get a branch/commit/pull request/build/deployment to associate with an issue?

A3:

1. Branch - include the issue key in the branch name
2. Commit - include the issue key in the commit message
3. Pull request - Include the issue key in the source branch name, included commit message, or the pull request title
4. Build - include the issue key in an included commit message
5. Deployment - include the issue key in an included commit message

Q4: What is your experience with dealing with the conflicts that rise when working on the same code on isolated issue branches?

A4: Our experience is that this is rarely a problem. Most issues have little code overlap, but occasionally conflicts do occur.

Typically there are 2 problems that happen:

- Long running feature branches isolated from other ongoing development
- Massive refactoring tasks that need to be separated until they are completed, tested and ready for release

For the former, a serious option is to do the development and integrate frequently, but keep the features themselves behind feature flags, so that only our own dog fooding or a select few customers see the ongoing changes. This ensures conflicting code is continually integrated and minimizes the chances of conflict as well as minimizing the risk and impact when a large feature is added to the main development branch.

If this is not feasible, and in the case for large refactorings, we make sure that the development branch is integrated into the feature branch as often as possible (by merging the changes on the base/development branch into the feature branch). This ensures that all ongoing development is completed and tested against the long-running feature branch as often as possible. If there are any merge conflicts, it is much easier to get the people who made the changes to help resolve merge conflicts or at least to keep their scope minimal so that resolution is easier.

The best solution is to break any work down into chunks that can be merged into the stable or development branch as frequently as possible. This minimizes the chance that changes to the same files are being made in feature branches at the same time.

Q5: What strategy do you recommend for managing parallel branches for ongoing development (features), hotfixes and their deployment to different environments (QA/Staging/Production)?

A5: We have a number of branch strategies documented on our git web site. In particular see the comparing workflows section.

More advanced details can be found in some previous talks, Getting Git Right and continuous deployment workflows are covered in more depth in Git workflows for SaaS teams.

In brief, there are 2 predominant workflows: a product release workflow for downloadable products and a SAAS workflow for online services (Git workflows for SaaS teams).

For downloadable products, most teams use a variant of Gitflow Workflow where main is used for ongoing development, and each previous minor release has it's own branch, from which bugfix branches are made and merged back to, and when required, a bugfix release is built. Each previous release branch has all changes merged downstream to subsequent releases and main to make sure all bugfixes are included in all subsequent versions.

Q6: Does the release hub work well with Kanban and frequent releases?

A6: Yes, it works very well. The Release button on the Kanban board can be used to create a new version containing all of the issues for that release. Once this version has been created, the release hub can be used to check for any warnings or to get an overview of the version.

Even without the Kanban board, you can create a version at any time and add issues to it even if those issues have been completed. The release hub can then be used to check for any warnings to ensure the release is ready.

Q7: Can the release hub show information about issues from multiple Jira Software projects or is the release hub project and fix Version specific?

A7: release hub is a detailed view of a version. As versions are currently specific to a project, the release hub is also specific to a project.

Q8: Can you have release hub warnings automatically populate in a Hipchat room?

A8: As of today there are no integrations between release hub warnings and Hipchat and we don't currently have plans to build any. We have been thinking about the different ways release hub could enhance team collaboration with Hipchat and would love to hear more from our customers on how they might use this integration or any other integrations with our other products.

Q9: What is the 'Release' button wired to? A script to deploy to your production server as a Job in Bamboo?

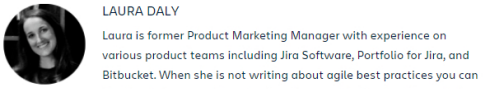
A9: The 'Release' button has a few functions associated with it:

- It can set the release date and change the status of a version to 'Released'. If there are any open issues in the version it will also give the option to move those issues to another version. This is all available with or without Bamboo integration.
- When Bamboo is integrated with Jira Software, the Release button can be used to kick off a new build that can be configured in Bamboo to take the steps necessary to release your version (e.g. a script to deploy to production, or produce a new artefact).
- The Release button can also be used to kick off a manual stage for a Bamboo build that has been run. This allows you to have a build that runs automatically, but have an optional stage that is only triggered manually that actually performs the deployment/release. (The stage would be equivalent to the whole build in Option #2, but can use the artefacts the build produces in it's execution.)

Q10: Any plans to integrate release hub with Github/GitHub Enterprise?

A10: The release hub already works with GitHub and GitHub Enterprise. All you have to do is connect your Jira Software instance to your GitHub account using DVCS Accounts found under the add-ons administrator menu in Jira Software. Then you can start tracking the progress of any of your versions with any related development information included in the release hub.

SHARE THIS ARTICLE



LAURA DALY

Laura is former Product Marketing Manager with experience on various product teams including Jira Software, Portfolio for Jira, and Bitbucket. When she is not writing about agile best practices you can find her in the mountains chasing storms or looking for the perfect berm.

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

How to deliver quality assurance at speed

Learn how a small team of Quality Assistance engineers evangelizes and coaches sustainable testing methods across the development team.

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

Qa at speed →