Using branch-based deploys in your continuous delivery pipeline

One of the questions we have to puzzle through is, continuous delivery to where? And from where, for that matter.

BY SARAH GOFF-DUPONT

We often use terms like "pipeline" to describe the continuous flow of code from your repo to your users. But when you map it out on paper, that pipeline will look more like a network – especially if you're using a branch-and-merge workflow (which we highly recommend).

If you're developing on a branch, it makes sense for that branch to have its own deployment path so changes can be thoroughly tested before merging up to main, or even released straight to customers. Atlassian dev tools support branch deploys quite nicely, as you'll see.

I'll start by going over the three primary use cases, then walk through the integrations between Jira Software, Bitbucket, and Bamboo that help you put it into practice.

The case for branch-based deploys

At its core, the case for integrating your network of branches into your continuous delivery pipeline is convenience. If you share a testing environment with other teammates, it's easier to coordinate a cadence of deploys representing discrete work streams than to merge all streams before deploying to test and have to untangle them when further changes are (inevitably) needed.

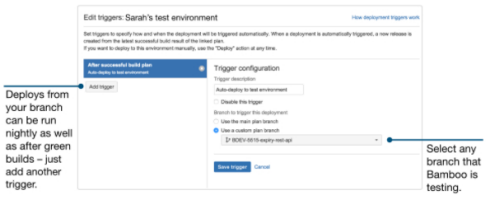
If you're lucky enough to have multiple test environments available, it's even easier. Likewise with releasing to customers: some teams like to release straight from branches, then merge updates from the branch down to main afterwards.

Automatically triggering deploys from a feature branch

Using feature branches for user stories and bug fixes is a great way to push changes to your repo in Bitbucket (thus enabling CI to run against them) without subjecting the rest of your team to broken builds and test failures while the work is in progress.

If your team prefers to ship from main, you'll likely want to deploy that code to internal environments for exploratory and acceptance testing. This works best when there's a handful of test environments your team can deploy to so nobody has to wait for their turn to use a single, shared environment.

With branch deploys in Bamboo, you can configure the deployment trigger for any environment to automatically deploy the artifacts built on a given branch – either at scheduled intervals, or with each successful CI build of that branch. You can even use the two trigger types in combination.

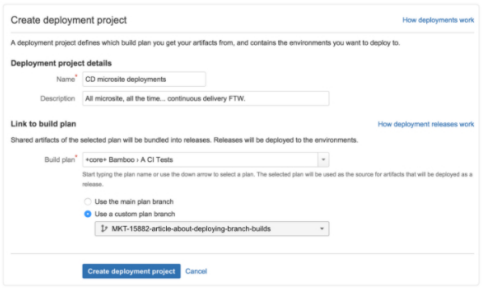


Let's say you have a single performance testing environment that everyone shares. Throughout the day, deploy main there whenever it builds successfully – for most branch-loving teams, builds of main are usually the result of new work being merged in, so you'd want a gut-check on performance right away.

Then overnight when the repo is idle, schedule deploys from each developer's branch at appropriate intervals so everyone gets performance feedback on their work at least once a day. (Deployment jobs in Bamboo can conclude with

a script, Ant, or Maven task that kicks off performance tests.) All the while, developers can use automatic deployment triggers to send builds out to their individual test environments whenever their feature branch builds successfully.

To set this all up, you can either a create discrete deployment project in Bamboo for your dev branch, or associate your branch to an environment within a deployment project by customizing how deploys to the environment are triggered.



Check out the Bamboo docs for step-by-step instructions on associating a deployment project with a branch and customizing deploy triggers.

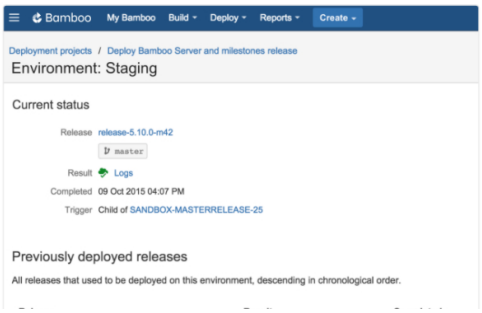
Releasing from a release branch

For teams that prefer to put work in progress directly on the main code line, releasing from a branch is more appealing. While main or trunk gets deployed to the test environment, release branches are the ones you deploy to staging, and ultimately production. Release branches can be used as part of the Gitflow approach as well.

Release branches are cut periodically and changes are shipped out from there. And most teams want to make a human decision around when to and where to deploy a release branch, rather than an automated trigger. No problem.

Bamboo has this concept of "releases" which are entities within Bamboo that encapsulate the most recent artifacts built from a given branch, plus all the commits, test results, and Jira issues associated with all the builds on that branch since the last time a release was created. So, under the hood, a release is basically your packaged artifacts plus a whole lot of meta-data.

And we want all that rich, juicy data to flow through your continuous delivery pipeline along with the artifacts so you can trace it all back to the first commit and the user story that started the whole thing. (I believe it was Geoffrey Chaucer who first said "All roads lead to Jira Software.") So conceptually, it's the release you're promoting through your environments – not just a build. And you'll see that reflected in Bamboo's UI.



Anyway, back to triggers. When deploys are triggered automatically, as in the case above, Bamboo creates a release automatically.

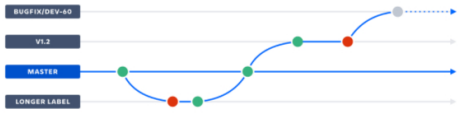
But push-button deploys are different. In this case, you create the release (which can be done from a build results screen or from within a deployment project), give it a name, and select the build to pull artifacts from. With the release in hand, you can now deploy it to any environment – even straight to production, if that's what you really need to do. Teams that use the approach typically create several releases (release candidates, really) before ultimately giving one the stamp of approval and sending it further through the pipeline.

When it's time to promote the release to the next environment, head to the *all deployment projects* screen, where you'll see all your deployment projects (duh) and all the environments associated with each project. Click the deploy icon next to your target environment, and walk through the deployment preview wizard, selecting the option to deploy an existing release (as described in much greater detail in the link).

Shipping updates to supported versions

If you don't live in the SaaS world, you probably have to support multiple versions of your software simultaneously, and occasionally ship critical fixes, back-port security updates, etc. This means maintaining stable version branches for a very long time – years, most likely. Builds from these branches are usually deployed to a common location, like a website or external repository, where customers can pull down updates to the versions they're using.

Multiple Version Workflow



For these branches, the easiest way to turn builds into releases is with the *create release* button found on the build result screen for the build you're shipping. Sure, you can start at the *all deployment projects* screen as described above. But because you're zeroing in on a very specific build you want to ship – which isn't necessarily the most recent build on that branch – I think starting from the build result screen provides a little extra assurance that yes: this is the build I want to ship.

Putting it all together (one continuous delivery pipeline at a time)

Branch-based deployments are the natural extension of Bamboo's plan branches, which, in turn, are the natural extension of whatever branching scheme you set up for your team in Bitbucket. As I've shown here, and discuss at length

in other articles, there are several CD-friendly models to choose from.

Your branching scheme should reflect how your team tackles and delivers work.

So let's review the various pieces in play here:

- **Jira issues** – Remember that issue keys are magic. Include them in the name of your branches and your commit messages so all commits, builds, and deployments (even pull requests!) link back to the issue.
- **Branches** – Did you know that when you've got Bitbucket connected to JIRA Software, each issue has a convenient *Create branch* link? And that when you use that link, the

issue key will automatically be added to the branch's name? This is how strongly we believe in creating a branch for each issue you work on. Try it in your next sprint, if only just for fun.

- **Plan branches** – Bamboo automatically detects new branches in your Git, Mercurial, and Subversion repos if you enable automatic branch management. By the time you've made your first commit, Bamboo has already put that branch under test.
- **Deployment projects** – The "delivery" phase of your continuous delivery pipeline. You'll associate a repository and one or more environments with each deployment project.
- **Environments** – Bamboo's representation of the environments on your network. Each environment can be associated with different branches, use different deploy triggers, and use different steps (called *tasks*) to execute a successful deploy.
- **Release** – Packaged artifacts + meta-data. When looking at a given release in Bamboo, you'll be able to see all the JIRA issues (because issue keys are magic!), commits, test results, and artifacts associated with it. You can even see which environments it's been deployed to and flag it as *approved* for promotion through the pipeline, or *rejected*.

Every use case I've covered here is used by at least one