CloudBees® Case Study

doist

Industry:
Computer Software

Geography:
Global

Product:
CloudBees Feature Management

"Having a tool like CloudBees Feature Management increases the stability of our products and the flexibility of our team, relieving pressure not just on the DevOps team but on our marketing, product, and support teams too. It gives all of us peace of mind knowing that our production infrastructure and our applications are under control at every stage of their lifecycle."

Roman Imankulov
Head of the Web Development Team
Doist

How Doist Built Stability for Our DevOps Team in an Entirely Remote Environment

Contributed by Roman Imankulov, Head of the Web Development Team, Doist

When you're a remote DevOps team maintaining products and deploying new features across over 15 time zones, you need tools that allow everyone to take control, act decisively, and resolve issues quickly. Regardless of where they're located.

For our team, CloudBees Feature Management has become an indispensable tool for releasing features simultaneously across our supported platforms, freeing the marketing team to run A/B tests without the need for developer resources, ensuring the highest standard of quality for our releases, and minimizing disruption when things do go wrong.

Walking the Remote Walk

At Doist, we believe that remote work is the future. Our 80 employees collaborate from over 20 different countries. Our company is well known for Todoist, our top-ranked productivity app for individuals and businesses, and Twist, our communication app for remote teams. As the leader of backend development at Doist, I manage the team responsible for developing, deploying, and maintaining the backbone for all features of both products.

We're very proud of our cross-platform support for both Todoist and Twist – if you can do something on our Android app, you can do the same thing on our web app or our iOS app. That feature parity is one of the main reasons our users love our apps, so if the release on one platform is delayed, it can cause a lot of frustration.

Cross-platform feature parity means that coordinated releases were a must, but pushing new sections of code across Android, iOS, web, Mac, and Windows all at the same time was a messy process. Running a big release was always like opening the floodgates for developers. Once you made a deployment and let users install your app, there was no way back. No matter how well we tested our apps internally, there was always a surge of bug reports and crashes during the first hours of use. The only remedy was issuing bug fix updates that users would install with some delay. To ensure we gave a consistently exceptional experience to our millions of users, we turned to feature flags.

Feature flags would allow us to deploy features across all platforms and flip the switch when we were ready to show users that feature. We hoped it would give us the platform unity that our users have come to expect.

Building Versus Buying Feature Flags

Initially, we started developing our own ad hoc feature flag solution that was custom built for each platform. But it lacked consistency as we had essentially just hard-coded if/else clauses that dictated what we showed various users. Making sure the right users saw the right features was messy without a simple feature flag interface that you might find with a third-party solution.

In reality, feature flags are more complicated than if/else clauses. We anticipated there being a server-side API endpoint, so users would periodically pull this endpoint. Initially, we planned to build a minimal feature flag service to manage our coordinated releases.

We estimated that we could make it work in one month, and would need one developer per platform for the implementation. Quite quickly, we understood that these estimations were far from accurate. There would be much more hidden work that we did not anticipate initially.

What data can we return? How can we make sure it's scalable? If millions of clients start pulling this data, do we have the right architecture to cope with the load? Even something as simple as building a dashboard—which I hadn't initially considered—would be time-consuming. Effective cross-platform feature flag synchronization and resolution is another non-trivial issue that can take much longer than planned. Our final estimation was closer to three to four months end-to-end for the initial implementation, with the active involvement of developers from all the platforms on almost all stages.

A member of the team made me realize that we were wasting our time. The effort and risk of a home-grown solution are quite high, as well as the burden to maintain it. Why try to build a hacked-together feature flag solution when it's not our core competency? Especially when there are entire companies out there dedicated to getting this right. We considered two feature-flag solutions: LaunchDarkly and CloudBees Feature Management. Feature-wise, the two platforms were similar but there were a few ways that CloudBees Feature Management stood out.

First, I appreciated the attention CloudBees gave us. Their solution architect took the time to walk us through the solution and explain the architecture behind it. Second, when I examined the CloudBees architecture, I understood it. It's all quite straightforward with no moving parts. That gave me another reason to trust CloudBees: I was convinced they knew what they were doing.

As a bootstrapped company, we're always conscious of where we invest our resources. Our freemium model means we have tons of users but only a portion of those users are paying customers. We weren't sure if the investment in a third-party feature flag service was worth it, and we were still toying with the idea of building our own solution.

In the end, CloudBees made this decision easy. They came back to us with pricing that accommodated our business model, making the cost manageable. We finally decided to scrap the idea of building our own solution and move forward with CloudBees Feature Management.

"When I examined the CloudBees architecture, I understood it. It's all quite straightforward with no moving parts. That gave me another reason to trust CloudBees: I was convinced they knew what they were doing."

Roman Imankulov
Head of the Web Development Team
Doist

The Perfect Solution for A/B Testing and Incremental Rollouts, Too

We dove right in by starting with just one platform to make sure CloudBees Feature Management really worked for us. After that, we went on to coordinated releases and incremental rollouts. It wasn't until we got deeper into it that I realized how many applications feature flags could have.

A/B testing was something we had tried before CloudBees Feature Management, but it never really stuck. There were so many moving parts involved that were hard to coordinate. Something as simple as testing how users responded to a change in the subject line of our daily digest emails, for example, required a lot of extra work from developers that took away from time directly improving our apps.

Developers are busy with their own tasks, so when a request like this comes from the marketing department, that team had to wait because they couldn't create the cohorts themselves. When the data came in from the test, we'd have to study and then report the results. It was a very involved process, and everything required developers' resources.

Using CloudBees Feature Management changed the game. Instead of me hard-coding cohorts, it's easy for the marketing and growth teams to select users based on different criteria, add them to a cohort, and maintain it.

For example, a colleague from the marketing team recently asked if we could create new user cohorts in order to compare user engagement before and after a new feature release. Before CloudBees Feature Management, this would have been a heavy lift for the backend team. Now, I simply pointed her to the new CloudBees Feature Management dashboard where she was able to understand and set up the cohorts herself: "It's all so obvious, I don't need your help."

The marketing team has used the CloudBees Feature Management dashboard to run several A/B tests to figure out how to better convert users into paying, premium customers.

"Using CloudBees Feature Management changed the game. Instead of me hard-coding cohorts, it's easy for the marketing and growth teams to select users based on different criteria, add them to a cohort, and maintain it."

Roman Imankulov
Head of the Web Development Team
Doist

Using CloudBees Feature Management offers a hidden benefit that I hadn't expected: the simplification that comes from a decoupling of concerns. Developers must develop just like marketers must market. Now, the marketing and product teams can find what they need in an easy-to-understand dashboard. They don't have to bog down developers with additional tasks, and the developers don't have to be the bottleneck for a simple A/B test like our digest emails.

Another big benefit is that with incremental rollouts, we can actually release features to our internal teams before releasing it to our real users. It makes experimentation easy, because we can always easily fix issues that arise. We can squash any bugs before they negatively affect our users, and we gain more certainty in our releases. We can deploy to production without being afraid that we're going to break something in the process. If we break something, we just break it for ourselves. If it doesn't work, we just toggle the switch back and get to work on fixing it.

Perhaps most importantly, CloudBees Feature Management makes it easy for people who might be working on opposite sides of the world to coordinate their efforts in real time. That's a big win when your company is completely remote.

"CloudBees Feature Management makes it easy for people who might be working on opposite sides of the world to coordinate their efforts in real time. That's a big win when your company is completely remote."

Roman Imankulov
Head of the Web Development Team
Doist

The Perfect Solution for the Future of Work

When you have a shiny new tool, all of a sudden you start seeing applications for it everywhere. I see discussions in Twist and I'm quick to jump in to suggest using a feature flag. It's not just me, either; other coworkers have seen the value in this tool, and have begun suggesting new ideas for feature flags, too.

In the future, I see us further exploring the use of kill switches. If something goes wrong, anyone on the DevOps team will be able to quickly and easily disable the problematic part of the application without affecting the upstream services. If one service goes down, we can isolate the problem and fix it without it affecting the overall performance of the application. CloudBees Feature Management will give our remote development team the ability to fix issues in production immediately.

Having a tool like CloudBees Feature Management increases the stability of our products and the flexibility of our team, relieving pressure not just on the DevOps team but on our marketing, product, and support teams too. It gives all of us peace of mind knowing that our production infrastructure and our applications are under control at every stage of their lifecycle.

"CloudBees Feature Management will give our remote development team the ability to fix issues in production immediately."

Roman Imankulov
Head of the Web Development Team
Doist

Learn more about Doist

www.doist.com

CloudBees, Inc.
4 North Second Street | Suite 1270
San Jose, CA 95113
United States
www.cloudbees.com
info@cloudbees.com