split

WePay Case Study

RAMPING UP RELEASES & RELOCATING TO MICROSERVICES

When WePay needs to make a change to its infrastructure or introduce a new feature, it has the potential to have a cascading effect that could impact user experience and the business' bottom line. Their release cadence was initially targeted for every week, then that slipped to every two, but they still found that schedule restricted major feature development. With Split, WePay can now ship new discrete features as often as they'd like, testing them first internally, then with select customer groups, before slowly ramping them to all of their customers.

BUSINESS BACKGROUND

WePay powers payment experiences across apps and services, making it the heart of a complex ecosystem of payment processing. Their reach extends from the credit card processor all the way down to the individual user trying to make a transaction, and touches every application and merchant in between. When WePay needs to make a change to its infrastructure or introduce a new feature, it has the potential to have a cascading effect that could degrade or improve not only user experience, but several businesses' bottom lines. By adopting Split, their engineering team avoids these customer experience pitfalls while releasing new features faster.

HOW SPLIT BENEFITS WEPAY

- Improved feature velocity with the ability to take on big features without the fear of slowing down the release cadence.
- Enhanced end customer experience with targeted rollouts.
- Increased availability of engineering resources to focus on other mission-critical projects.

SUMMARY

- When WePay needs to make a change to its infrastructure or introduce a new feature, it has the potential to have a cascading effect that could impact user experience and the business' bottom line.
- Their release cadence was initially targeted for every week, then that slipped to every two, but they still found that schedule restricted major feature development.
- With Split, WePay can now ship new discrete features as often as they'd like, testing them first internally, then with select customer groups, before slowly ramping them to all of their customers.

DECOUPLING FEATURE RELEASES & CODE PUSHES

WePay used to release like most software companies, pushing everything all at once. Their release cadence was initially targeted for every week, then that slipped to every two, but they found even that schedule restricted large feature development. Part of the problem simply came from having a small team, but the greater part of that slowdown came from needing to focus on feature branches that could be wrapped up inside of a week or two.

In order to increase its development pace, WePay needed to decouple the feature release process from pushing code—with Split, they're now able to do just that. Split's feature flag libraries and targeting capabilities allow the team to deploy features in the "off" state, shielding users from changes or problems until the new code has been thoroughly tested. Split now enables WePay to take on big features without the fear of slowing down the release cadence; indeed, their pace has dramatically quickened under this mindset. They use Split's on/off capability in concert with targeted rollouts to ship new discrete features as often as they'd like, testing them first internally, then with select customer groups, before slowly ramping them to all of their customers.

It also means they can release new features to the public whenever, without needing to pull an all-nighter to release on a marketing timeline.

WePay helps platforms increase revenue through integrated payment processing. Constant Contact, FreshBooks, Zoho and more than 1,000 others have turned to WePay for payments within their own UX and without fraud exposure. Founded in 2008, WePay is based in Redwood City, California. For more, visit WePay.com.

Languages: Java, PHP, Python Integrations: New Relic, Slack Uses: Controlled rollout, feature flags, microservices migration

This process has helped with QA too. When a recent code release resulted in an increase in declined credit cards, they were able to quickly pinpoint the problem because it was obviously anomalous when compared to the control group, isolated by Split, not receiving the feature. As Chris Conrad, VP of Engineering recalls:

"Tying feature accessibility to releasing code makes a lot of bad things happen. Like, for example, needing to release code at midnight to correspond with a press release. At midnight engineers are tired, and you're creating this weird interdependency between things that should not be related. Adopting Split means we can keep releasing code but make features accessible only when we need them to be, and with the click of a button."

HOW AWS SUPPORTS SPLIT'S SOLUTION FOR WEPAY

Split relies on AWS to scale dynamically to the needs of WePay. Split runs its code in containers leveraging Kubernetes on top of EC2. It manages the incoming traffic through a combination of Network and Application Load Balancers from AWS to manage network request flows. AWS infrastructure allows Split to scale rapidly to adjust to traffic demands. It also allows Split to build a highly available system for WePay, leveraging the multiple availability zones and regions that AWS supports.

MIGRATING FROM MONOLITHS

Another challenge recently faced by the WePay team was moving away from a monolithic architecture and towards microservices. As the company modernized its infrastructure, it knew it wanted to move away from a "one-size-fits-all" approach, the core services of which were originally written in PHP. But, because of the breadth of services and other businesses that WePay impacts, migrating with any downtime was basically a non-starter. Enter Split.

The company already had a successful feature flag deployment within the monolith, giving them basic on/off control when they rolled out new features. But it was limited to one language, PHP, and wouldn't work to bridge the gap created by ripping features out to replace them with microservices.

"We ran into big problems with how to take big pieces of code out of the monolith, while having the new service and the monolith continue to do effectively the same thing for different portions of our customers. You can't use something that's just specific to the monolith, you can't use something that's just specific to the service, and you can't use anything that only works in one language."

– Chris Conrad, VP of Engineering, WePay

CORE COMPETENCY

WePay has also been quick to note one of the secondary benefits of adopting Split—increasingly available engineering resources.

"We don't like to spend time doing things that aren't our core competency. Why maintain a feature flagging system, build and maintain its UI, or monitor its performance? Split does what we need it to do, and any engineer in the company can use it."

– Chris Conrad, VP of Engineering, WePay

Another compelling benefit of Split: integrations. The company was able to easily connect Split to their New Relic and Slack implementations, giving them one more QA refinement while also keeping the entire team up to speed with the status of feature rollouts.

ABOUT SPLIT

The Split Feature Data Platform™ gives you the confidence to move fast without breaking things. Set up feature flags and safely deploy to production, controlling who sees which features and when. Connect every flag to contextual data, so you can know if your features are making things better or worse and act without hesitation. Effortlessly conduct feature experiments like A/B tests without slowing down. Whether you're looking to increase your releases, to decrease your MTTR, or to ignite your dev team without burning them out—Split is both a feature management platform and partnership to revolutionize the way the work gets done. Switch on a free account today or schedule a demo to learn more.

split

Switch It On at split.io