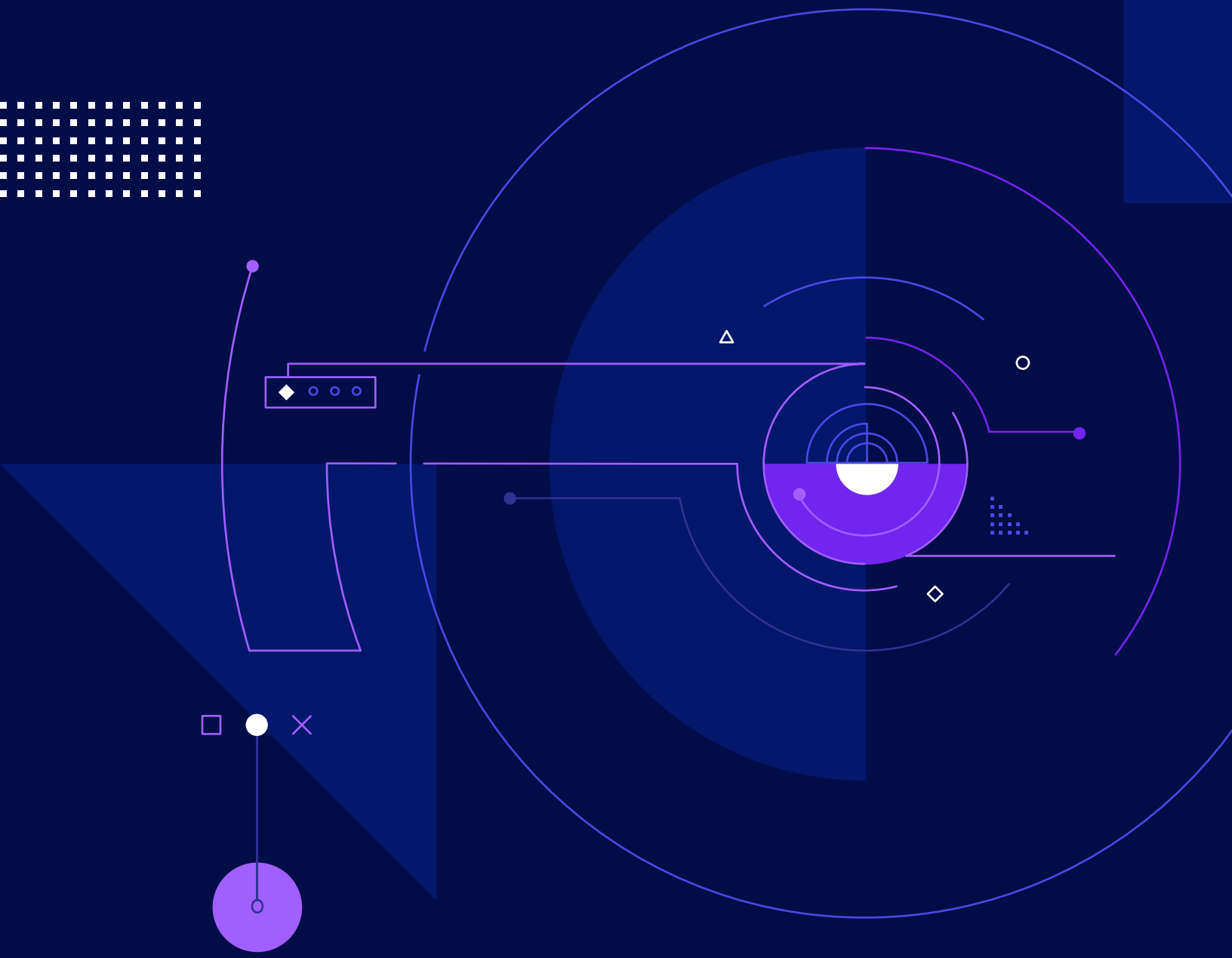


A Guide to Effective Release Management

bugsnag

You've finally received word from your product team that the new features they've been debating will be added to the upcoming release. With the design spec in hand, it's now time for your release team and the engineers to plan for this next release.

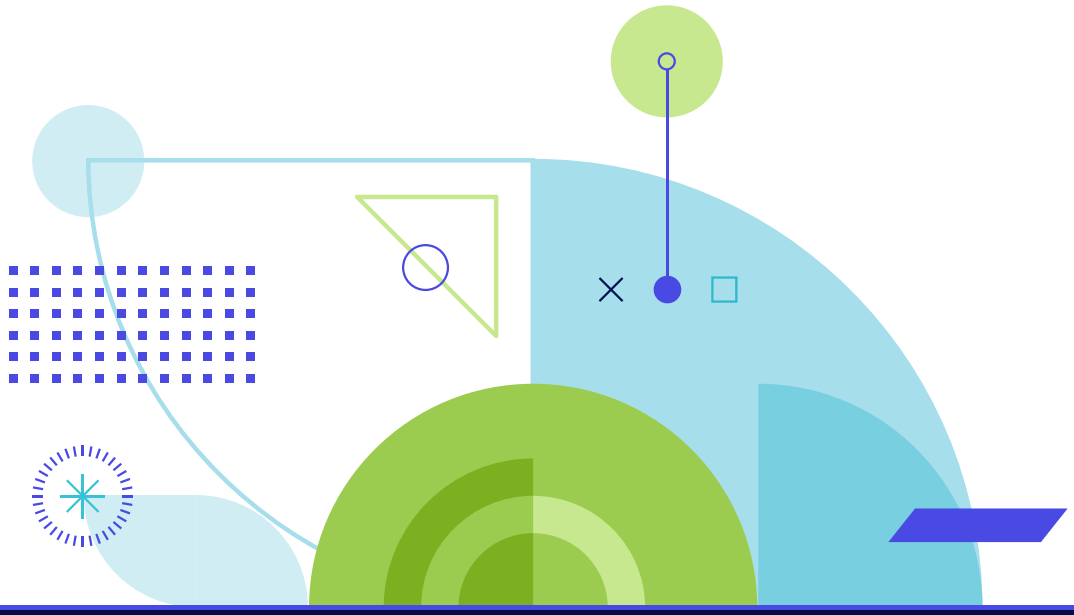
Before you get started, you think back to previous releases and notice a couple patterns:

- The reliability of the release and the process itself doesn't always go as smoothly as you'd like, even if previous releases have been "okay."
- Releases often get rushed and pushed out quickly when customers are in dire need of a feature.
- When pressing product changes are needed, timelines get moved up and builds are forced out faster.

You may find yourself asking, "Am I doing releases right?!"

Relax. It's common. Design changes happen, new priorities are raised, and engineering teams often experiment with new engineering processes.

Release management is a forever evolving process that needs continuous improvement and tweaking.



Creating repeatable release processes

The key is to adopt a **repeatable release process** in which you can:

Quickly identify major issues in testing stages and know exactly where to fix them

Remain agile and focus solely on the major bugs

Constantly monitor stability health and adherence to stability targets

That's where **Bugsnag** comes into play, allowing you to fully manage your release health and give you control over the quality and the reliability of your software launch.

The end result? You and your team can continue to build product iterations and put forth the best experiences possible for your customers.

The challenges of managing releases

Managing releases can be difficult because there are so many teams and moving parts involved.

Release managers must be able to:

- Adjust when non-scoped features are added to a release at the last minute
- Ensure additional functionality does not impact the stability of each release iteration
- Recognize potential obstacles immediately and make sure to address them before things get out of hand
- Provide instantaneous feedback to each team on release health
- Know with certainty when it's safe to move to the next phase of your release process

---

Pro tip: Product and executive teams should use Bugsnag as well. (It's not only for software and release teams!) By sharing knowledge about the health of a release, the entire company can be aligned around realistic expectations for the release launch.

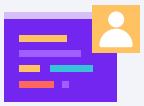
Begin with the development stage

As a sample use case, let's take a look at the launch of a website refresh.

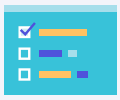
The workflow for adding new features to a front-end application has many steps and hand-off moments. While it starts with the product team specing product requirements and the design team explaining the new features, the engineering team must scope the work and develop a plan for how to best architect the features to meet the product requirements. During and after the immediate build by the engineering team, an internal QA and code review will also need to be completed.

All of these handoffs introduce risk at each stage. It's important to have the right tools in place to monitor stability throughout the development process. That way, you can pinpoint roadblocks as soon as they come up and avoid missing problems too far down the line.

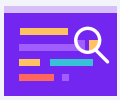
01 New feature design spec from product team



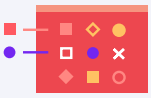
02 Hand to release manager and engineering for planning and development



03 Define a 'development' stage in Bugsnag for early-issue detection



04 New feature goes through testing



05 New feature deployed to staging (alpha, beta, etc.) for review

06 Monitor the release by filtering to see your staging deployment

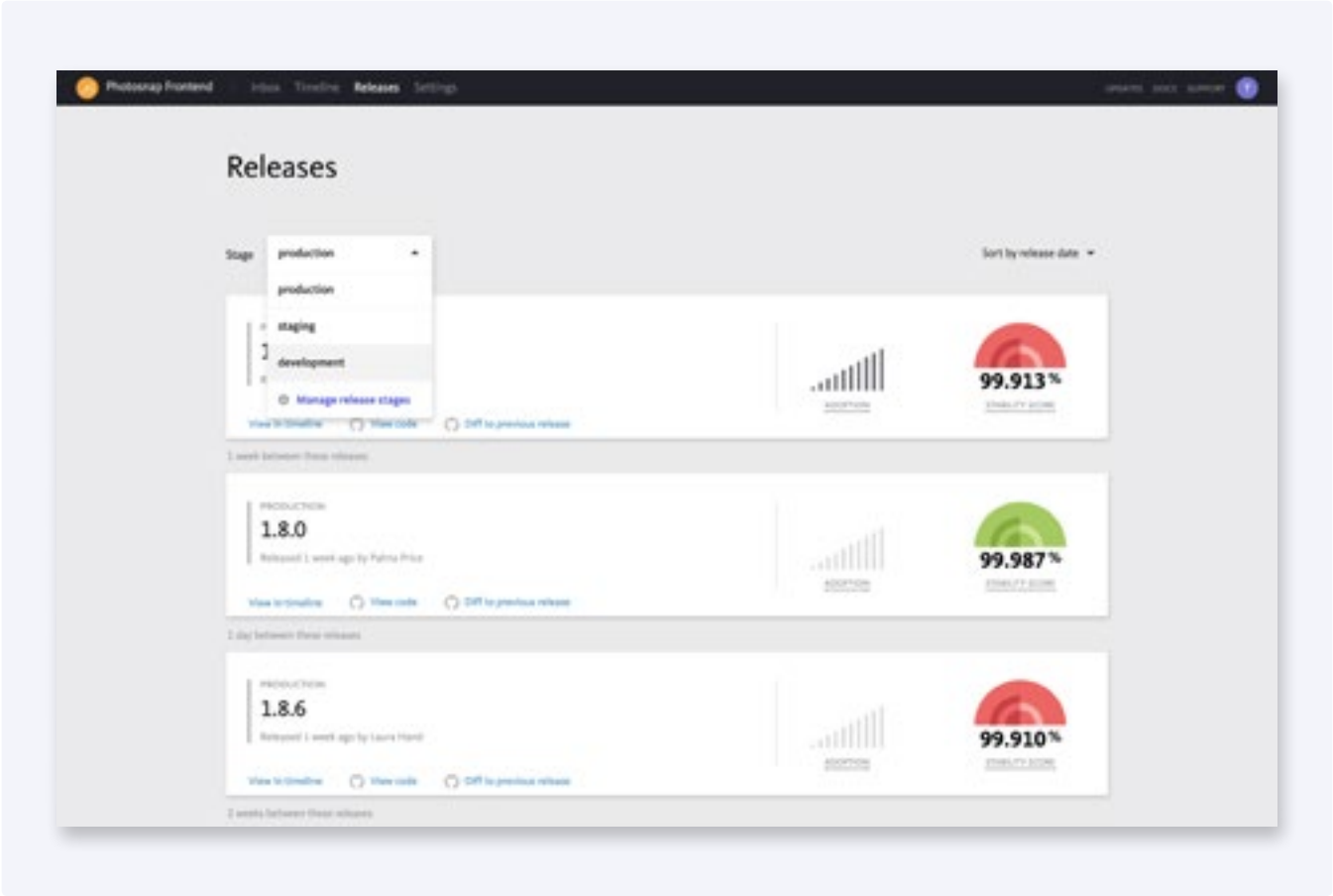
07 Filter by a specific build to see how it compares to others. If it introduces more errors, send feedback to engineering team

08 Keep in mind your stability target. At what point must you stop feature work and fix bugs?

09 Launch time

10 Monitor stability closely in the first 24 hours after a release and at all times as stability is something you continually work on.

The Bugsnag release dashboard is especially good for addressing this process. While the project is with the engineering team, you can define a builds stage called “development” and begin monitoring errors and issues in real-time.



Move on to staging environments

After several iterations of testing, the product team reviews the latest release in your dedicated staging environment (or alpha / beta environment). They can troubleshoot, look at specific use cases, complete a review of high-risk or sensitive areas, and submit suggested changes back to the engineering team.

Throughout this process, you have the power to continually monitor the deploy and compare the number of errors seen in previous builds to those in the latest release. If your team is fortunate enough to have the ability to run automated testing and regression testing, additional custom stages can be created and monitored in the Bugsnag release dashboard as well.

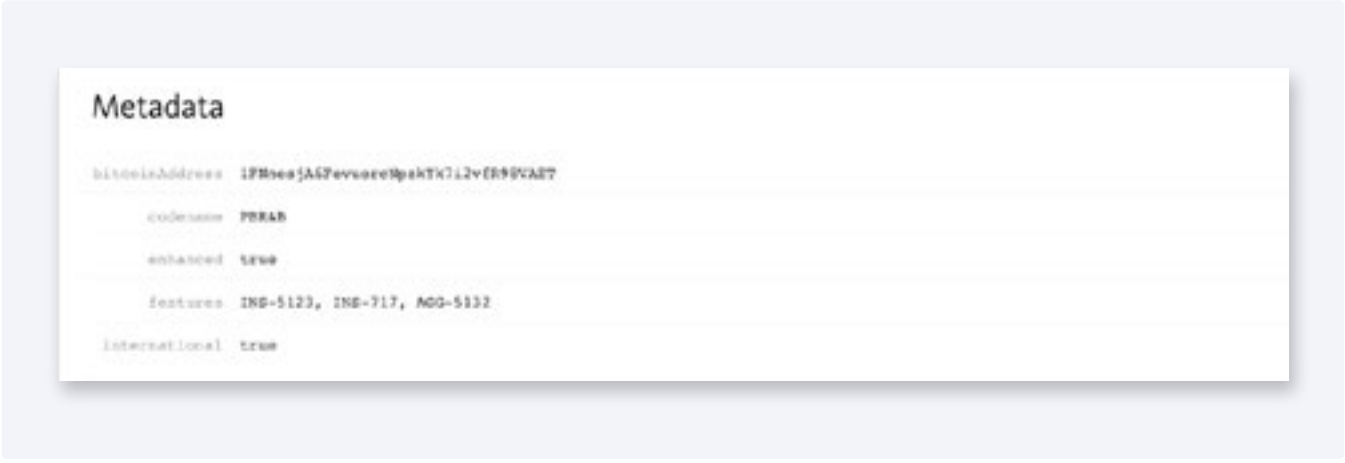
It's also becoming common practice for partial and phased rollouts to be used that deliver new software to only a small percentage of users first. Wouldn't it be great to see and catch the errors that hit the first 3% of your users and resolve these errors before they reach the other 97%? With Bugsnag, this process is easy.

How can I use Bugsnag during the staging process?

Bugsnag allows you to monitor the process through custom metadata that's set up using the build tool integration. That means key metadata is specific to a build. For example, you can identify key differences between various releases by adding the parameters used to build the application, as well as details of the changes contained in the build.

What are some Bugsnag customer strategies that have worked?

Many Bugsnag customers have used custom data to run experiments that can help developers know if a particular new feature in a release has more bugs than other similar builds without that feature. Others have created custom identifiers for customers or users who have opted into a beta build of a new product, which is then used to attribute crashes to users who have opted in versus those on existing releases.



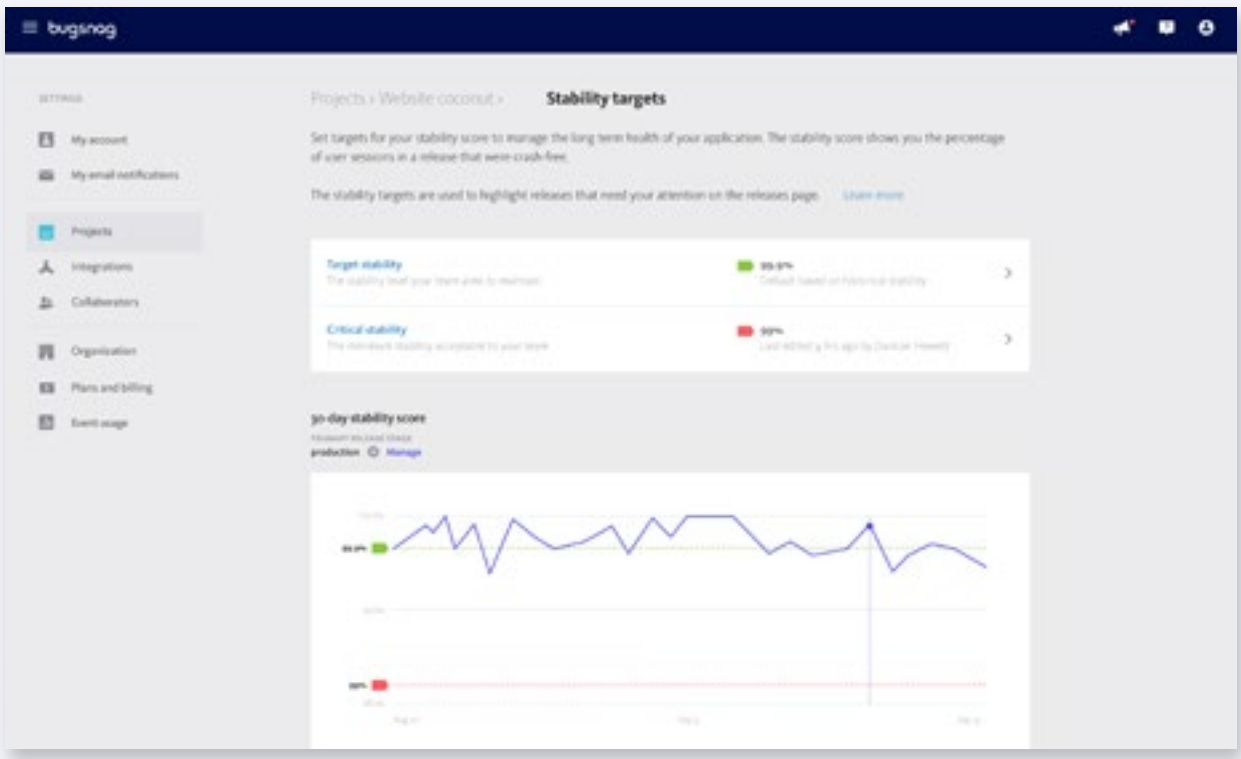
Set stability targets throughout the process

Each release should have stability targets. At Bugsnag, we call this metric your "**Error Budget**," which is defined as your threshold for taking immediate action on fixing and prioritizing errors seen in your releases.

This error budget might be different for each team within your organization. Your engineering team might decide that a 96.55% crash-free rate is an acceptable standard for your staging builds. However, your Executive team might not let builds go out until you have reached a minimum 98.95% score due to SLAs with high-paying customers or a desire not to drive away users who won't come back and use your service again.

Therefore, it's important to have an open, healthy dialogue amongst teams and keep in mind what the most important driving metric for each release should be.

Pro tip: Using tools integrated with Bugsnag such as Splunk, SQS Data Forwarding, or webhooks allows information to be passed seamlessly so each team can process it and compare with other datasets that impact decisions.



Launch your release

Congratulations! You have launched your latest release to your production environment. However, your journey doesn't stop there.

- **Ongoing stability management** requires constant review and discipline. It's not something you fix once but rather a continuous process.

- **Monitoring release health** is especially important during the 24 hours after launch. Errors during this time provide an indication of the end user experience.

In Bugsnag, we do all the heavy lifting to allow you to quickly identify and fix errors. Some of the tools you can use in Bugsnag include:

- View Errors introduced, new errors
- Integrating Bugsnag stack traces with your source control (GitHub, BitBucket, GitLab)
- Creating issues from Bugsnag errors in your issue tracker

You can also easily integrate Bugsnag into your everyday workflow by integrating with tools like Slack, Jira, email, and many more. Bugsnag currently offers 45+ integrations and a dedicated API to make sure that you have all the necessary alerts, information, and data to make the best release decisions.

Pro tip: Dedicated Slack channels and custom email notifications can alert you to major spikes or increases in errors. These integrations make it easy for you to immediately know when you should be reviewing Bugsnag.

Conclusion

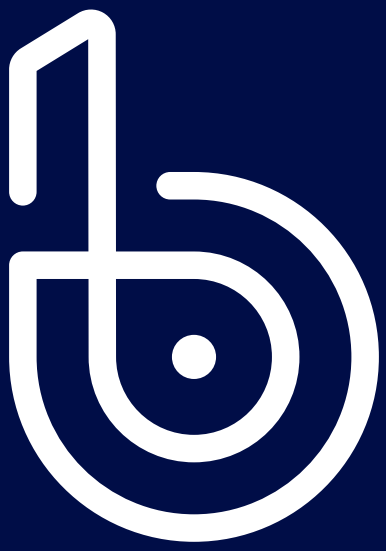
Inject confidence into your entire company by adopting a release process that teams can follow. By utilizing Bugsnag, you make it easy for everyone to track release stability and continually monitor your releases. No surprises here.

While you'll always need to remain flexible and ready for curve balls, a well-calibrated and predictable release cycle goes a long way to providing company-wide peace of mind. When Bugsnag's stability scores and targets are used, everyone becomes accountable for the stability of the release and the customer experience, which equates to stronger production releases from the word "go."

Bugsnag is the number one solution for full stack stability monitoring with best in class support for mobile apps. Trusted by engineering teams at Airbnb, Lyft, HotelTonight, Pandora, and more.

When monitoring a release, Lyft uses filters to find errors by version, a powerful functionality that gives them visibility into each release without much effort.

LYFT | **VIEW CASE STUDY**



bugsnag.com sales@bugsnag.com San Francisco, CA & Bath, UK