SAUCELABS

WHITE PAPER

Optimizing CI/CD for Continuous Testing - Feature Management

BREAK UP THE MONOLITH AND DELIVER NEW FEATURES FASTER

As more organizations embrace Continuous Integration (CI) and Continuous Delivery (CD) as a mechanism to release apps faster, many find that there are a number of options to consider when making this transformational shift. However, while there is significant thought put into how development practices will change, very few teams consider how CI/CD will change the way they test the code that they create.

This technical paper is the second in a series outlining various topics development organizations of all sizes should consider when optimizing their processes for CI/CD, and how they relate specifically to testing. This critical piece of your engineering strategy can influence not only the quality of your applications, but also how quickly you can deliver them to your users. For many teams, these considerations can effectively make or break their CI/CD initiatives.

TABLE OF CONTENTS

INTRODUCTION

Moving to an effective and efficient CI/CD pipeline requires significant effort from organizations. It involves process and policy changes across every team. The payoff can be extraordinary:

- Continual product improvement
- Consistently deliver high quality digital experiences to their customers
- Speed without adding risk

However, as your release velocity begins to increase, how can you ensure your apps are still thoroughly tested?

Adding continuous testing best practices from the outset can enable the transformation to CI/CD by allowing code to move through an accelerated pipeline without testing becoming a bottleneck. To achieve this requires an entirely new set of features -- testability features -- built into the architecture itself, along with other changes to the way software is built. Without these changes, organizations typically struggle to see the benefits that CI/CD promise. This technical paper is the second in a series discussing the approaches, requirements and processes to consider when implementing continuous testing in a CI/CD workflow, and will focus on feature management - including how you should consider selecting, conceiving and coding them up.

SMALLER FEATURES

Similar to application development's shift to microservices, learning to decompose large monolithic "features" into smaller ones takes hard work. These changes don't just hit technical issues; more importantly, they also impact business concerns. Stakeholders, product owners, and engineers all need to understand the advantages of a microservices architecture.

Smaller features are generally less complex, thereby making them easier to incorporate into a CI/CD pipeline. It's far easier to build, integrate, and deploy a smaller component with a small set of APIs, database changes, and UI, versus a huge footprint of massive schema updates, interwoven service dependencies, and numerous components to be installed for the end user.

The other major benefit to this approach is that it makes each feature easier to test. By limiting feature branches to the component level, it is much easier to write automated tests that are both atomic (short, succinct, focused only

on one feature) and autonomous (not dependent on the results of another test to run successfully). These tests, when written correctly, can be integrated throughout the CI/CD pipeline. Whether it's part of the end-to-end regression cycle, or shifted left earlier in the pipeline to provide fast feedback to developers pre-commit, testing smaller components in isolation allows for bugs to be caught earlier, and stable code to move through the pipeline more quickly.

Small features will be easier to turn off (make switchable). To be successful with larger features under CI/CD, organizations will need to understand and use feature flags to make features switchable.

SWITCHABLE FEATURES

Here's a common scenario - a skilled team asks "How do I test CAPTCHA in my user registration form?" because they are struggling with automating a tool (CAPTCHA) that is specifically designed to prevent automation.

The answer? "Don't test it. Cheat instead."

What's meant by this is to design your test architecture so that you can turn off features wholesale to make the system more testable. In this case, you don't need to test CAPTCHA itself; it's a well-tested, high-quality third-party component. You obviously need to test its integration and functionality, but that's likely a one-time thing.

Instead of trying to incorporate it into your checks, automating tests for your user registration process should first disable CAPTCHA, run the test, then turn CAPTCHA back on. This requires coding and engineering effort to make the feature switchable, but it's an extraordinary help when trying to wrap testing into your CI/CD pipeline.

Moreover, feature switching can be used for much larger chunks of functionality- -a technique sometimes called "canary releases." You could deploy features to your entire production farm, but enable them only on a carefully monitored small number of servers. This allows you to gradually roll out your features to a small population, ensure they're working properly, and gradually turn on those features to more and more users.

CONCLUSION

Implementing CI/CD means not only rethinking your pipeline, but also the way you create your apps. By decomposing your monolithic applications into smaller, more manageable features, allows for continuous iteration, deploys and testing. It keeps autonomous teams more productive, as no one is left waiting when something inevitably breaks. And most importantly, it allows for mechanisms that allow for gradual deploys to ensure that teams are delivering the best quality experiences to users.

Sauce Labs provides the world's most comprehensive Continuous Testing Cloud. Optimized for CI/CD with integrations to the industry's most popular tools, Sauce Labs is the perfect platform for all of your continuous testing requirements throughout your CI/CD pipeline. To learn more, take a look at this tech talk on integrating continuous testing into your CI/CD pipeline.

SAUCELABS

ABOUT SAUCE LABS

Sauce Labs is the leading provider of continuous testing solutions that deliver digital confidence. The Sauce Labs Continuous Testing Cloud delivers a 360-degree view of a customer's application experience, ensuring that web and mobile applications look, function, and perform exactly as they should on every browser, OS, and device, every single time. Sauce Labs is a privately held company funded by Toba Capital, Salesforce Ventures, Centerview Capital Technology, IVP, Adams Street Partners and Riverwood Capital. For more information, please visit saucelabs.com.

saucelabs.com/signup/trial

FREE TRIAL

SAUCE LABS INC. - HQ 116 New Montgomery Street, 3rd Fl San Francisco, CA 94105 USA