SAUCELABS

WHITE PAPER

Optimizing CI/CD for Continuous Testing - Environment Management

HOW FAST AND SCALABLE ENVIRONMENTS HELP ACCELERATE RELEASE VELOCITY

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 first 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 your CI/CD initiatives.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Moving to an effective and efficient Continuous Integration (CI) and Continuous Delivery (CD) pipeline requires significant effort from organizations. It involves process and policy changes across every team. The payoff can be extraordinary: continual improvement as the organization moves to consistently deliver high quality digital experiences to their customers at speed. However, as your release velocity begins to increase, how can you still ensure your apps are well-tested?

Adding continuous testing best practices from the outset can enable the transformation to CI/CD by allowing code to move through an accelerated build 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 first in a series discussing the approaches, requirements and processes to consider when implementing continuous testing in a CI/CD workflow, and will focus on environment management.

MANAGING FAST ENVIRONMENTS

Fast-moving teams can't wait weeks or months to get environments for builds and testing in place, nor can they wait for already overloaded DBAs to manually inject data or schema changes. This is simply untenable for projects where builds and testing are ongoing many times each day. CD requires environments be created, provisioned, and ready for test in just a few minutes--more than ten or 15 minutes makes it nearly impossible to have a smooth-functioning automated pipeline. This need for speed requires significant changes to how infrastructure and environments are managed. Cloud-based hosting offerings like Microsoft's Azure or Amazon's Elastic Computing Cloud (EC2) allow teams to offload their build and server management. Testing on demand in parallel can be easily added to a pipeline through offerings like the Sauce Labs Continuous Testing Cloud. With various infrastructure options (full cross-browser, headless browsers, mobile devices, and more), services like Sauce Labs enable teams to avoid management of potentially huge matrices of devices, operating system versions, and features.

SELF-SERVICE FOR TEAMS

Luckily, technology has advanced to the point where there are affordable, industry standard solutions. In addition to the cloud-based solutions mentioned above, on-premise solutions range from commercial products like VMWare to open source containers like Docker.

All of the tools, both on- and off-premise, allow developers to wrap code, test suites, and configurations in a CI/CD tool set. Scripts, plugins, adapters, etc. to manage infrastructure are available for every popular solution listed above - it's simple to integrate provisioning and deployment into a job on Jenkins or other toolsets. Moreover, many cloud vendors provide easy secure tunneling to backend (on-premise) infrastructure to support a mix of on- and off-premise systems. The Sauce Labs Sauce Connect Proxy is just one example of this.

It's important to remember, however, that with "do it yourself" infrastructure comes the extra added requirements to maintain, update, and scale as the pipeline grows in size. To learn more about some of the benefits and drawbacks of building your own continuous testing infrastructure or outsourcing to a provider like Sauce Labs, check out this Build vs. Buy whitepaper.

No matter which option you choose (in-house or outsourced), infrastructure management isn't limited to provisioning and deploying. Fast moving teams need self-service. They need to create an environment on-demand for any given build, and, obviously, their tools need that capability as well. Organizations need to move system management and access rights down from centralized administration to individual teams. This doesn't mean ignoring or bypassing regulatory compliance and auditing issues emplaced by concerns such as HIPAA, GDPR or Sarbanes-Oxley, it means understanding the power of automation via a delivery pipeline to handle security, artifact storage, configuration, and auditing.

Teams also need to quickly access log, performance, or environmental data from production systems. Traditional, slow-moving organizations often use ticketing systems with long SLA times to handle these situations. These times frames are untenable in a CI/CD environment where teams need to have constant monitoring of their software in production.

Logging, monitoring, and advanced log search tools can also have benefits for testability and debugging. Building this capability into dev and test environments means it will be "free" in production and enable programmers to perform production support, removing a barrier between development and operations.

SCRIPTABLE DATABASES

Continuous Deployment "but manual database changes" puts Continuous Delivery quality in jeopardy. Scriptable database changes means the schema and data can be stored in source control right alongside the appropriate version of the system code. If the database versioning and the production code are separate, then changes need to be coordinated so they don't break production - and they certainly will break self-service dev and test environments. Even if the changes are carefully coordinated in dev and test, anyone trying to patch an old version of production with new database code will find problems and create new problems, unwittingly.

Many toolsets offer this scripting for major databases, regardless of whether they're traditional RDBMSs or "NoSQL." Tools range from commercial products to open source tools such as LiquiBase and Ruby Migrations.

SCRIPTABLE DATA

Some applications have an API to create a user, an account, or a product; wrapping that in a command-line application is trivial. Adding the capability to batch-export, clear, or delete data in the database and import data in the same way, by logical groups, adds a feature to the product that is incredibly valuable for testability. With this capability in hand, programmers can create test data sets that are imported automatically for any test. The test can then login as those users, perform transactions, check the results - and detect when the test ends, clear system of the all test, data and results, and re-set, re-import it for the next test. Advanced features to save time include allowing a test to be 'read only' (no need for teardown) or to perform its own, unique setup. Once the team has achieved the trifecta of fast self-service scriptable environments, databases, and data, the continuous deployment system can truly run end-to-end tests atomically, without introducing errors and challenges to the deployment pipeline.

CONCLUSION

When building your CI/CD pipeline, it is crucial that your environments be fast, agile, reliable and scalable. Enabling teams with various environment options not only eases the barriers of getting organizational buy in, it also allows them to deliver on the promise of CI/CD. Namely, this means producing high quality features and applications to users, and ultimately delivering on an excellent customer experience. There are a number of options available - from DIY to a host of services and cloud-based platforms. But the key is creating an infrastructure that allows for developers to easily create, manage and execute their tests, without slowing down release velocity. Providing this new and seamless workflow allows the organization to evolve their CI/CD pipeline and drive true adoption of this advanced processes.

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