WHITE PAPER

CORE ELEMENTS OF CONTINUOUS TESTING

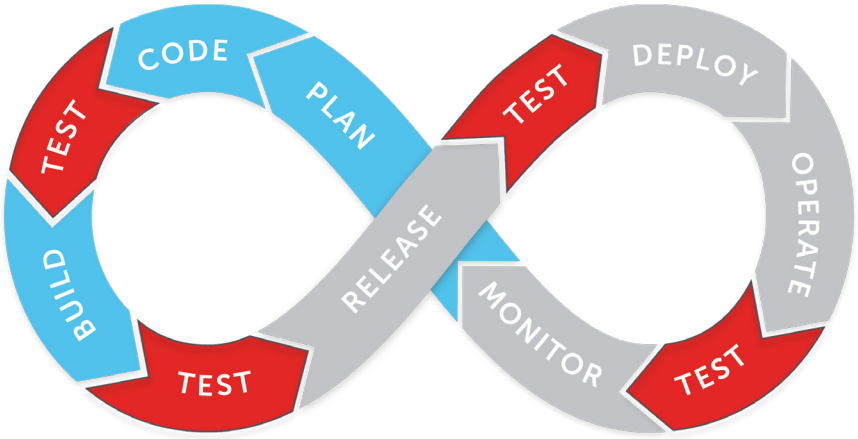
Today's modern development disciplines -- whether Agile, Continuous Integration (CI) or Continuous Delivery (CD) -- have completely transformed how teams develop and deliver applications. Companies that need to compete in today's fast-paced digital economy must also transform how they test. Successful teams know the secret sauce to delivering high quality digital experiences fast is continuous testing. This paper will define continuous testing, explain what it is, why it's important, and the core elements and tactical changes development and QA teams need to make in order to succeed at this emerging practice.

SAUCELABS

TABLE OF CONTENTS

WHAT IS CONTINUOUS TESTING?

Continuous testing is the practice of executing automated tests throughout the software development cycle. It's more than just automated testing; it's applying the right level of automation at each stage in the development process. Unlike legacy testing methods that occur at the end of the development cycle, continuous testing occurs at multiple stages, including development, integration, pre-release, and in production. Continuous testing ensures that bugs are caught and fixed far earlier in the development process, improving overall quality while saving significant time and money.



CONTINUOUS TESTING

WHY CONTINUOUS TESTING?

Continuous testing is a critical requirement for organizations that are shifting left towards CI or CD, both modern development practices that ensure faster time to market. When automated testing is coupled with a CI server, tests can instantly be kicked off with every build, and alerts with passing or failing test results can be delivered directly to the development team in real time. Continuous delivery means that once all tests have passed, updates can be pushed directly to production with confidence.

CORE ELEMENTS OF CONTINUOUS TESTING

We've found that customers who are delivering quality digital experiences at high velocity focus on six essential pillars of continuous testing:

1. **Cultural commitment to quality**

   Since a robust quality user experience is so critical to success, it must be owned by the entire team. From a cultural perspective, continuous testing succeeds when ***everyone*** owns quality. Test cases are outlined before coding even begins or tests are written as needed. Either way, developers and test automation architects work together to ensure that the code is optimized for test automation. In addition, teams collaborate on test results via tools such as Slack and HipChat to speed feedback and debugging.

2. **Testing at every stage of the development cycle**

   The key to moving from one stage of the development cycle to the next is often predicated on the passing of tests. These are the "gates" of development, as it's required you pass through that gate in order to proceed. Using testing as a gate to each step of the process ensures the code continues to operate as designed when new changes are introduced. Testing early and often enables you to catch bugs sooner in the development cycle, when it is much less expensive than catching and remediating that same bug in production. We typically see our most successful customers executing the following types of functional tests throughout the software development cycle:

   - *Unit testing*. Unit tests exercise only a single behavior. The goal of a unit test is to get fast feedback on whether the new code works as intended. Well-designed unit tests, especially when developed with Test Driven Development, lead to well-structured code that has isolated components. This will mean fewer failures, and when those failures occur, they will tend to be isolated, easy to debug, and fix.

   - *Integration Testing*. In this phase, individual software modules (components) are merged into the shared pipeline and tested. The goal of an integration test is to ensure the build remains stable as new code is introduced, and that no existing functionality is broken by new code. These "pull request tests" can be automated and kicked off via a CI server to ensure that new features that are added work as desired and don't break existing functionality or conflict with other new feature.

- *Automated End-to-End and Regression Testing.* Once the code is finalized, it must be packaged and deployed across various servers and resources to demonstrate that the application works for a user from login to logout in an environment that mirrors production. The goal here is to ensure that the entire system works as expected. Automated testing at this point ensures all the capabilities work with each other as planned, and that there are not any problems with dependencies or other environment specific issues.

- *Production Testing.* Testing doesn't stop once you've released the software, it must be tested frequently to ensure it continues to work as designed. Testing the actual code in production serves to alert you quickly to issues before they cause major problems for your users and your business.

- *Exploratory and Live Testing.* While automation is essential to Continuous Testing, manual (or live, interactive) testing is still a necessity. Computers are ideal for covering a broad swath of well defined, repetitive tasks; however, it takes creativity and human intervention to dream up ways to try and break an application or notice an unexpected result.

3. **Using expertise to build out automation**
   Organizations are departing from the legacy testing paradigm of taking manual test cases and automating them with simple methods like record and playback; while that provides an easy way to get started, it also creates brittle tests that do not scale as the amount of automation increases. Now, more experienced teams leverage intelligent frameworks to write automated test scripts as part of the development process. Automation is the future and we will increasingly see businesses deploy automation within their organizations. Manual testing still serves an important purpose but it doesn't scale to meet all the needs of today's modern development practices. Automated tests have to fill the gap and do what manual testing can't---accurate repetitive, parallel checks of expected behavior. Freeing up manual testing from repetitive tasks to focus on intelligent, creative exploratory and usability testing.

4. **Test execution platform that provides you with comprehensive coverage**
   To ensure your digital experiences will work for all your users, it's essential to have a test execution platform that includes virtually any combination of browser, OS and device. This provides development and QA teams with

the coverage and flexibility they need to deliver a flawless user experience for all users. It also provides the ability to reproduce bugs in virtually any environment as users report issues.

5. **Ability to scale up and down your test infrastructure instantly as needed**
   As your team begins to automate more tests at multiple stages in the development cycle, scalability becomes a critical requirement. Whether you are preparing for a major launch or a busy season, testing will likely surge at various times throughout the year, particularly as multiple teams check in and test new code throughout the day. One way to instantly scale up the amount of tests you can run is to implement parallel testing, that is running multiple tests across various environments in parallel rather than in serial order, so you achieve the coverage without adding significant test run time. To ensure scalability and flexibility, you need a test execution platform that can scale seamlessly as well as stay up to date as new releases and devices come out.

6. **Visibility and analytics**
   To know where you're going, you must be able to see where you've been. Analytics are critical to understanding how tests are performing and quickly identifying bottlenecks and quality issues. Teams need real-time visibility into coverage, test run times, failures and efficiency, so they can identify trends and make changes to code or practices to increase quality and speed releases.

TACTICAL ENGINEERING CONSIDERATIONS

We've reviewed the essential elements to Continuous Testing, as well as the cultural and operational requirements that can help it succeed. There are also a number of engineering considerations that teams should take into account in order to make Continuous Testing successful in their organizations.

Keep the tests green. If the team is spending a great deal of time "greening" the tests, they may be covering a user interface that is not mature enough to handle them. Alternatively, a passing test run might be required to call the story done. Keeping the test run passing should be a whole team responsibility, not something done by Software Development Engineering in Test (SDET) at the end of the cycle after a handoff.

Having the engineering practices in place to make a release decision automatically. Continuous Testing can actually slow down delivery if the team has a low first time quality or high bug-injection rate. If the team is spending

a large amount of time on bug fixes, work on engineering practices like unit tests, code reviews, user experience and better shared understanding of what to build ***before*** building it.

A full run in under 5 minutes. It's easy to start a new project with a CI system and five minute build; the challenge is maintaining it as the feature list increases. Larger applications will want to move to a component architecture, run tests in parallel, or both, to make this happen.

What tests to put in the build. Some organizations, especially those that don't run tests in parallel, maintain two or more test suites. The first, smaller test suite is a "sanity" run, designed to keep the build fast. A larger, slower set of more (or all) tests can run in a continuous loop. This creates two feedback loops: A tight loop of under an hour (actually we prefer under fifteen minutes) and a multi-hour run that provides more coverage.

Continuous testing without continuous delivery. For teams with a legal requirement for final User Acceptance Test (UAT), such as medical devices, it might make sense to use Continuous Testing without Continuous Delivery. Continuous Testing, after all, is just good engineering work. The result will be incredibly high quality code delivered to UAT with considerably less rework at the end, faster code delivery, along with the ability to de-scope and begin UAT immediately several times a day.

BENEFITS OF CONTINUOUS TESTING

Continuous testing is not simply a nice-to-have technology or a way to make engineers' work a little easier. It is an essential resource for helping the business to remain competitive and continue to innovate. By implementing continuous testing practices, you can delight your customers by automatically releasing new features at an extremely rapid pace.

Amazon is an excellent example of these practices in action. They test at all the integration points between development and production, and automate the testing of production code. Along with continuous delivery, they average an update release every 11.6 seconds. You may not have that kind of release velocity, but it's reassuring to know that by implementing the core elements of continuous testing covered in this paper, you can set yourself on a path to accelerate the release of high quality software at a pace much faster than you do today.

For help setting up a continuous testing discipline in your organization, please contact the experts in continuous testing at sales@saucelabs.com

ABOUT SAUCE LABS

Sauce Labs ensures the world's leading apps and websites work flawlessly on every browser, OS and device. Its award-winning Continuous Testing Cloud provides development and quality teams with instant access to the test coverage, scalability, and analytics they need to deliver a flawless digital experience. Sauce Labs is a privately held company funded by Toba Capital, Salesforce Ventures, Centerview Capital Technology, IVP and Adams Street Partners. For more information, please visit saucelabs.com.

SAUCELABS

SAUCE LABS INC. - HQ
116 NEW MONTGOMERY STREET, 3RD FL
SAN FRANCISCO, CA 94105 USA

SAUCE LABS EUROPE GMBH
NEUENDORFSTR. 18B
16761 HENNIGSDORF GERMANY

SAUCE LABS INC. - CANADA
134 ABBOTT ST #501
VANCOUVER, BC V6B 2K4 CANADA