Achieving Continuous Testing Excellence

WRITTEN BY MARCUS MERRELL
DIRECTOR OF TECHNICAL SERVICES, SAUCE LABS

The Metrics that Matter, the Benchmarks to Target, and the Best Practices to Implement

Think about your business for a moment. Maybe you're a retailer hoping to capture a larger share of the eCommerce market. Maybe you're a bank hoping to make it more convenient for your customers to transact online. Or maybe you're a publisher looking to push new content your readers can consume on the go.

Whoever you are, whatever you sell, and whatever service you provide, you probably have a digital business, and nothing is more important to your brand or your bottom line than your ability to deliver a high-quality digital experience to customers. In the modern era, that means continuously delivering fast, visually appealing, and flawless web and mobile applications.

Most organizations have replaced the traditional waterfall approach to software delivery with modern agile development methodologies. They believe that doing so will accelerate release cycles, improve the quality of applications, and lead to a better overall experience for users.

By all accounts, we should be delivering software better and more efficiently than ever before. And yet, the opposite is happening. Despite our collective commitment to speed and agility, release velocity is actually stalling, with the percentage of organizations releasing software on an at least monthly basis declining from 36 percent in 2017 to 27 percent in 2018, according to Forrester.[1]

1 Forrester: The Path To Autonomous Testing: Augment Human Testers First, Jan. 2019

The Continuous Testing Roadblock

Why is release velocity stalling out? The answer is testing: specifically, continuous testing.

According to a recent Gitlab developer survey, testing remains the most common source of development delays, cited by more than half of respondents. Most development teams are unable to implement continuous testing effectively throughout the software development lifecycle. As a result, they wind up compromising on either release velocity or release quality (or both), undermining their most ardent plans to become more agile.

Sauce Labs provides the world's largest hosted Selenium grid, just for you.

Sign up for a free trial today; saucelabs.com/signup/trial

SAUCE LABS

A brief history of web and mobile app testing.

Find out how Sauce Labs can accelerate your testing to the speed of awesome.

For a demo, please visit saucelabs.com/demo
Email sales@saucelabs.com or call (855) 677-0011 to learn more.

SAUCELABS
Testing at the speed of awesome.

The good news, though, is that if you can overcome the continuous testing roadblock, you can deliver quality software with the stability, predictability, and confidence that Agile promises.

Even organizations that have shifted to agile development still tend to take a legacy approach to testing, throwing it over the wall to a dedicated QA team as an isolated (and usually final) step in the delivery cycle. This isolates the testing team from the Product Owners and Developers, and often leads to disparate tech stacks and impenetrable barriers between QA and the rest of the organization.

Within this isolated world, even the most sophisticated teams are still too reliant on manual testing. No matter how large your pool of human test resources, you cannot scale testing to keep up with the growth of your business and the expectations of your customers without automation.

Then, too, those organizations implementing test automation are often doing so poorly. Test automation is a complex endeavor and requires engineers who understand your domain, the test automation craft, and software development—a hard-to-find mixture of skills. Without these skills, and a well-defined strategy, either your release velocity or product quality will eventually suffer. Usually both.

Given the struggles in the development community, the testing world is in need of some well-defined guideposts organizations can use to chart a manageable path to success. Fortunately, a new report based on millions of actual end-user tests has shed light on the four key benchmark metrics organizations should focus on to achieve continuous testing excellence.

By understanding these benchmarks, and the best practices necessary to achieve them, we can create a clear roadmap to continuous delivery—with stability, predictability, and confidence.

What is Continuous Testing, Exactly?

Before we do that, let's examine exactly what we mean when we say continuous testing. As the testing and DevOps industries continue to grow and new entrants join the space, the concept of continuous testing has started to mean different things to different people.

For this analysis, we define continuous testing as a best practice approach to automated testing, which allows you to deliver both quality and speed by continuously testing your web and mobile apps throughout the software development lifecycle.

As with any significant technological undertaking, culture plays a critical role in formulating the right strategy: Too many organizations make the mistake of diving into testing without first having the necessary cultural commitment to quality in place.

That commitment starts at the top with the highest-level executive leaders and filters down. As it does, the antiquated notion that developers own development and QA owns testing must be set aside, replaced by an understanding that quality is everyone's responsibility. Developers, test engineers, and product managers must have a mutual understanding of the baseline levels of quality and efficiency they intend to maintain throughout the development process.

Once the right cultural foundation is firmly established, the following benchmarks can serve as guideposts for organizations to follow on their quest for continuous testing excellence.

Performance Metric #1: Test Quality

Target Benchmark: Pass at least 90 percent of all tests run

Nothing is more central to effective continuous testing than test quality. All other components of an effective continuous testing strategy are only relevant to the extent that they're built on a foundation of dependable, high-quality tests.

And the most direct and reliable determinant of test quality is your pass rate. You should be writing, managing, and executing your test suites such that the overwhelming majority of your tests pass.

Now, to be clear, some tests should fail. The reason you test applications in the first place is to discover bugs and remedy them before they're pushed into production and create a sub-optimal user experience. A failed test that exposes a hidden flaw in your application is a test that has served its purpose well, and if none of your tests ever fail, you can safely assume something is wrong with your test scripts.

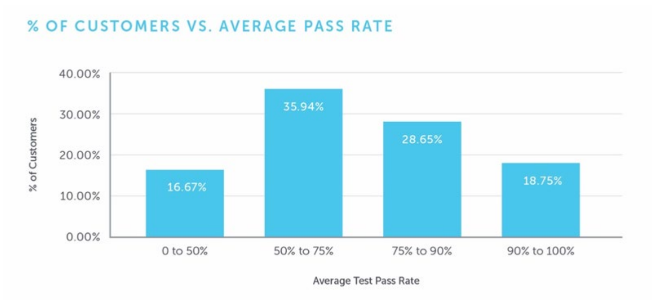
But there's a difference between tests failing occasionally and tests failing unpredictably, but frequently. (We call these "flaky" tests.)

Counterintuitive as it might seem, tests that fail only occasionally are actually quite reliable: Developers can safely assume new code they've introduced caused the application to break, which caused the test to fail. They can then quickly turn their attention to remedying that code and continuing on with the development process.

When tests fail with regularity, however, development teams begin to question the tests' validity, uncertain as to whether the failure is attributable to newly introduced code or to a problem within the underlying test script itself. When that uncertainty creeps in, manual follow-up and exploration are needed, and velocity drops.

As a general rule, organizations should aim for a pass rate of at least 90 percent on all tests run—with emphasis on "at least." What's more important than your numeric pass rate is ensuring the number of tests that fail doesn't exceed your bandwidth to follow up on those tests manually. If that breaking point occurs when you fail just 5 percent of your tests, for example, then you'll need to aim for a pass rate of at least 95 percent.

Whatever the case, most organizations still have a long way to go, as data from the aforementioned benchmark report shows that just 18.75 percent of organizations currently pass at least 90 percent of their tests.



Metric #2: Test Run Time

Target Benchmark: Complete tests in an average of 2 minutes or less

Agile development is ultimately about accelerating the release cycle and delivering more frequent releases. The faster your tests execute, the faster you can get feedback to developers, and the more quickly they can update their code and pass it along to the next stage of development.

The longer it takes your test suites to execute, the longer your developers have to sit around waiting for feedback, and the more likely you are to wind up with delayed releases.

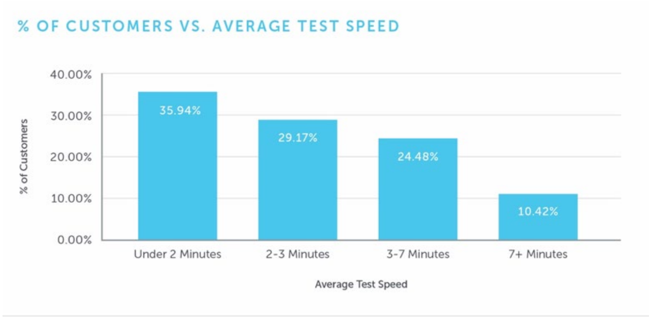
Test run time is one of the most misunderstood aspects of continuous testing. Development teams tend to focus on the number of tests in a suite rather than the length of those tests, mistakenly believing a test suite with just a handful of long tests will execute faster than one with a long list of short tests.

In reality, the opposite is true. If you're leveraging parallelization (and as we'll discuss momentarily, you should be), executing a test suite featuring many short tests is a much faster approach than executing one with just a few long tests. After all, when executing tests in parallel, you can only move as quickly as the slowest test in your suite.

Not only is test run time critical to speed and release velocity, it's also one of the most reliable predictors of test quality. The shorter a test is, the less opportunity there is for something to go wrong, and the more likely it is to pass.

How much more likely? According to the industry benchmark report, a test that completes in two minutes or less is twice as likely to pass as one that takes longer than two minutes. That's as clear a target benchmark as you can get. Organizations should design tests that execute in two minutes or less, and create test suites that favor many short tests over a few long ones.

Unfortunately—though perhaps not surprisingly given organizations' struggle with test quality and the clear correlation between test quality and test length—most development teams today fail to meet this critical performance benchmark, with just 36 percent of organizations completing their tests in an average of two minutes or less.



Metric #3: Test Platform Coverage

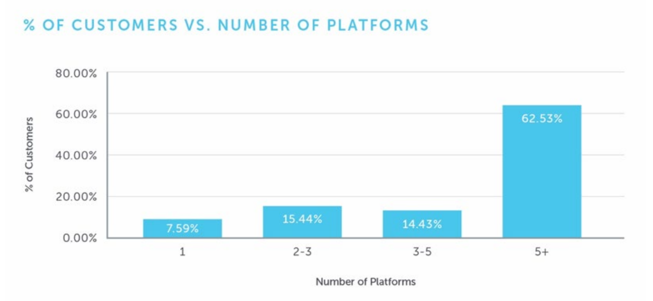
Target Benchmark: Test across at least 5 platforms on average

The universe of platforms and devices on which customers consume information and services is ever-expanding. Creating a flawless user experience means delivering apps that work as intended whenever, wherever, and however customers want to access them.

At any given time, you may have one customer accessing your website from her PC using an older version of Chrome, a second from his iPad using a just-updated version of Firefox, and a third customer using your native mobile app from her Android phone. All three customers need to be treated to the same delightful experience, both visually and functionally.

As a target benchmark, you should aim to test across at least five platforms (defined as any combination of a desktop/mobile operating system and browser) on average. The majority of organizations are doing that, with 62.53 percent meeting or exceeding the five-platform benchmark.

We live in an increasingly mobile-first world, so be sure to invest a proportional amount of resources into the testing and development of your mobile web and mobile native applications, and work to incorporate real-device testing into your continuous testing strategy as well. Your user analytics can also guide you in this.



Metric #4: Test Concurrency

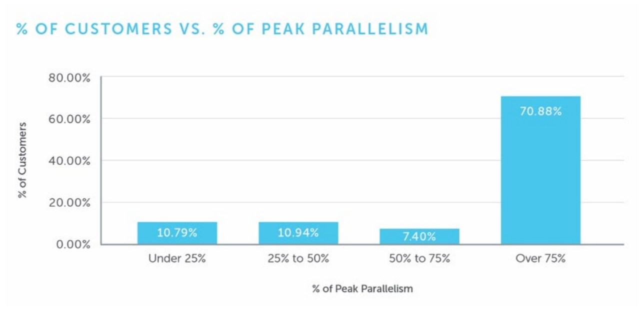
Target Benchmark: Leverage at least 75 percent of available test capacity during peak testing periods

Parallelization is the engine that makes continuous testing run. Without parallelization, your test suites will inevitably take too long.

Consider a hypothetical suite of 100 tests, each of which takes two minutes to run. If you can run those tests in parallel, your entire test suite will execute in just two minutes, and your developers will know the state of their application before they can get to the watercooler and back.

If, however, you don't have the capacity for parallel test execution, or don't leverage the capacity you do have, you'd have to run those 100 tests sequentially, and that suite of tests would now take more than 3 hours to execute. That's a lot of trips to the watercooler!

It's essential then to design your test environment in such a manner that you have enough available capacity to run tests concurrently. Just as important, you need to leverage that capacity to the fullest extent when executing your suites. As a general rule, you should leverage at least 75 percent of your available test capacity during peak testing periods, something 70 percent of organizations are doing successfully.



Four Best Practices to Help You Get There

Now that we've established a clear set of benchmarks to target in your pursuit of continuous testing excellence, let's look at some of the best practices that can help you get there.

#1: Run Atomic Tests

If you take just one thing away from this, let it be this: Run atomic tests.

Running atomic tests is the most powerful and effective way to decrease test run time and improve test quality. Atomic tests validate one single application feature. So, rather than writing one test to validate that your homepage loads and visitors can log in, you'd break it out into two separate tests, each responsible for just one piece of functionality.

Not only are atomic tests faster and more reliable, they're also considerably easier for developers to debug. Because atomic tests are inherently short and thus execute more quickly than tests examining multiple pieces of functionality, developers are getting feedback on code they just wrote.

As any developer will tell you, fixing code you just wrote is considerably easier than fixing code you may have written hours or even days ago.

In addition, because an atomic test is focused on just one piece of application functionality, there's usually no ambiguity about what's gone wrong in the event of a failure. It can only be that one feature, and knowing that, developers don't have to burn precious cycles rooting around for the cause of the problem. They can instead immediately fix the broken code and get back to the task of building great products.

#2: Ensure Test Autonomy

Another often-overlooked best practice that will help you both decrease your test run time and improve your test quality is to keep your tests autonomous. An autonomous test is one that can run independently of any other tests in the suite.

Autonomy and atomicity go hand-in-hand, and just as many teams make the mistake of designing a single test to validate multiple pieces of application functionality, many teams mistakenly design their test suites so that one test cannot successfully execute until the preceding tests in the suite have done the same.

This means, for example, that before you can execute a test validating the efficacy of your trial download, you first have to successfully execute tests for each and every function preceding it in the application workflow (account creation, login, filling out personal information, etc.).

When you design your suite this way, once one test fails, all of the dependent tests in the suite will fail as well. To avoid this common roadblock, design your suite such that individual tests can run on their own, and in any order necessary.

#3: Take a "Just-in-Time" Approach to Test Data

Another effective best practice to help ensure the stability and reliability of your tests is to leverage what's known as "just-in-time" test data. As opposed to traditional hard-coded test data, the static nature of which doesn't fit with the dynamic nature of automated testing, leveraging a just-in-time approach means you create test data, utilize it for a given automated test suite, then destroy it once your suite has been executed.

Not only does this cut down on the complexity of your tests, but it also ensures that data from a previously executed test doesn't falsify or interfere with the results of your current test.

#4: Leverage Cloud for Simplicity and Scale

Though the previous three best practices will help you execute tests with a greater degree of speed, stability, and reliability, you'll still need to find a manageable way to test at scale across multiple platforms—not an easy task.

In DZone's own 2019 Automated Testing Survey, 60 percent of respondents cited maintaining an up-to-date test lab as the most challenging aspect of testing, and with good reason! Setting up and maintaining the infrastructure (physical or virtual) needed to test across the variety of browsers, operating systems, and mobile platforms is both costly and complex—and that's before you even get into scaling your test environment to run tests in parallel.

This is where you leverage the cloud. The promise of cloud, after all, is increased agility, scalability, elasticity, and cost control. That makes it a perfect fit for continuous testing.

Leveraging a cloud-based, cross-browser testing platform enables you to offload all the infrastructure-related headaches you'd otherwise be grappling with on a daily basis, test across virtually any combination of browsers, operating systems, and devices, and ensure that you have the scale necessary to execute tests in parallel.

Keep in mind that every minute you spend maintaining your own test lab is one minute spent neglecting your business. No part of this effort will help you move features out the door, and eventually it will become a costly and frustrating distraction for your engineers.

In Closing...

We started by talking about the crossroads at which development teams now find themselves. Release velocity is stalling at the precise moment when it's become vital to the success or failure of the business, and there's a reckoning coming for organizations that can't figure out how to get the needle to point back in the right direction.

Once an afterthought, testing is now the competitive advantage that will define the coming decade. Organizations that focus on the right metrics, target the right benchmarks, and implement the right best practices will be the ones who reap the rewards.

Written by **Marcus Merrell,** *Director of Technical Services, Sauce Labs*
Marcus Merrell is well-versed in helping software developers with release management, CI/CD, cloud software, and test architecture. A frequent speaker at industry conferences worldwide, Marcus's work in quality engineering spans nearly 20 years, and he is also a contributor to the Selenium project and the co-chair of the Selenium Conference Organizing Committee. Marcus holds a BA in Linguistics from the University of Texas in Austin.

DZone communities deliver over 6 million pages each month to more than 3.3 million software developers, architects, and decision makers. DZone offers something for everyone, including news, tutorials, cheat sheets, research guides, feature articles, source code, and more. "DZone is a developer's dream," says PC Magazine.

Devada, Inc.
600 Park Offices Drive
Suite 150
Research Triangle Park, NC

888.678.0399 919.678.0300

Copyright © 2019 DZone, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.