rainforest

Applying Continuous Testing to Continuous Delivery

In This Guide:

As development teams move faster and project deadlines become more aggressive, continuous delivery (CD) has emerged as a means of achieving a higher bar for quality and a faster time to market. When implemented successfully, continuous delivery can improve efficacy and reduce the number of bugs that reach customers.

But for many teams, implementing a QA strategy that maps to continuous delivery is challenging. Traditional testing tactics are designed for longer development cycles, and can drag down the speed and efficiency of continuous delivery. The continuous testing methodology provides a faster, scalable approach to QA.

What is Continuous Testing?

In the Continuous Testing Manifesto, Russ Smith outlines the following guidelines for continuous testing. A continuous testing process is one which:

- Aims for a modular system, built around small, manageable chunks;
- Takes a balanced, rightsized approach to testing;
- Measures the results of testing to show quality improvement;
- Begins testing as early as possible;
- Uses unit testing to enable fast, specific feedback;
- Applies version control best practices to QA processes;
- Uses continuous integration as an early safety valve for quality;
- Integrates QA with development tooling.

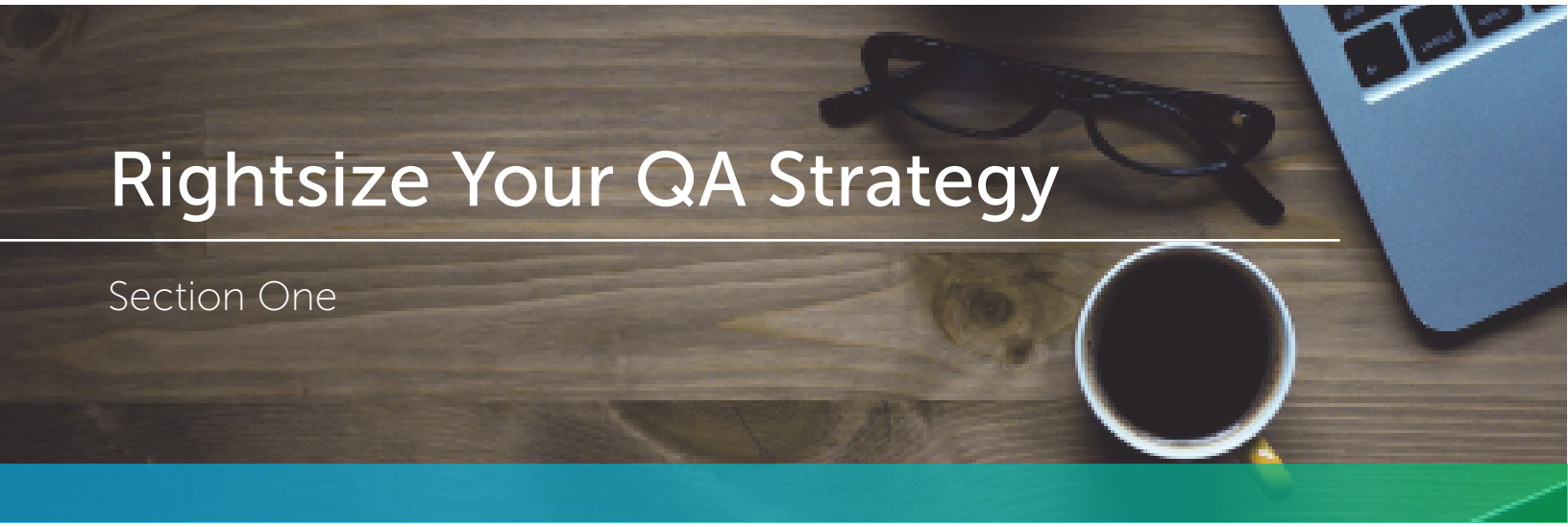
Teams that use continuous delivery as part of their DevOps workflow spend 21% less time on unplanned work and rework, and 44% more time spent on new work.

- State of DevOps Report, 2017

Applying Continuous Testing to CI/CD Processes

This guide provides an overview of how continuous testing practices can be applied to continuous delivery. The following chapters explore how continuous testing provides a more stable, scalable balance between the speed that teams need to stay competitive with the quality they want to deliver to customers.

rainforest



Rightsize Your QA Strategy

Section One

55% of teams claim that they have challenges managing the size of test data sets.

- HPE World Quality Report, 2017-18

How much testing is appropriate for continuous delivery?

Slower development methodologies can afford to err on the side of too much testing; if you have a 2-week QA cycle, then the time required to execute a few dozen additional tests is trivial overall. But more test cases require more time not only to execute, but also to manage. Because continuous delivery cycles are so fast, every task added to the process can greatly impact the total time required to ship. Additionally, in cases where developers are heavily involved in QA processes, the additional requirements for test writing and management eat into time that could otherwise be used for developing new or improved features.

The ultimate goal of QA is to provide confidence in the quality of every release. Finding the right amount of testing resources and time required to provide you with enough confidence to ship — without slowing down development — transforms QA from a bottleneck into an accelerator.

When Less is Better: Staying Focused on Quality Essentials

When it comes to continuous testing, less is often better. Less time spent testing means less lag between developing a feature and delivering it to customers. Fewer test cases means fewer opportunities for broken tests, and less noise in the quality measurement. Eliminating unnecessary testing cycles and test cases helps keep the testing process overall streamlined and easier to manage at scale.

To make your continuous testing as efficient as possible, focus on building a core set of critical tests that can be executed quickly. These tests will form the foundation of a test suite that focuses on gaining insight into quality that maps more clearly to customer and business value.

Learn more about developing a streamlined test suite in the webinar, "Creating a Custom QA Strategy."

In a webinar on Creating a Custom QA Strategy, Russ commented on how teams should think about the relative value of their QA tests: "The most valuable tests tend to be the highest level ones. The most difficult to maintain and get right —

just because of the sheer volume of them — are the edge cases. Optimizing your test suite creation to make sure you have all your bases covered with smoke tests is critical."

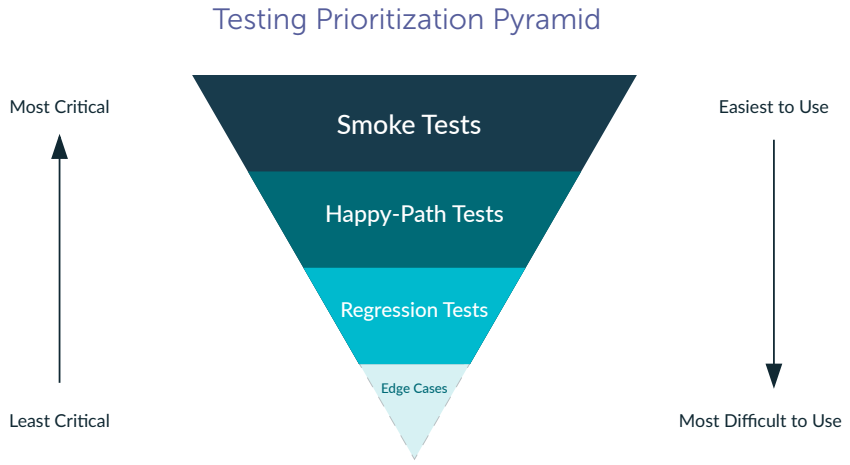


Fig. 1
Functional testing via Rainforest QA executed in parallel with unit tests.

Best Practices for Building a Rightsized QA Strategy

Follow Your Users

Don't test on all available platforms. Test what is used, and used most often: from key features and user flows to the most popular browsers and devices. Start small, focusing on what's essential to the majority of users, and build out from there. Your product team and your customer success team are great resources to help you understand what belongs in your QA strategy and what does not.

Prune Your Test Suite Often

As your product evolves your users' preferences and habits will, too. Regularly assessing where your QA resources are being used keeps those resources aligned with your business goals. Again, regularly checking in with the product team is essential for understanding which test cases to keep (and what to cut) over the course of time.

Focus On Development Goals

Focus on the execution methods and tools that will minimize overhead appropriate to your team's needs. If you're aiming to release on a weekly basis or longer, manual testing may meet your needs. If you want to move faster than that — daily, hourly or more — solutions like automation and on-demand platforms like Rainforest are more effective for returning fast results.

Measure to Improve

How do you know your testing strategy is in the right range? Measuring how successful your strategy is is crucial. Be sure that you don't get trapped in distracting metrics, such as the number of tests in your database, or number of test cases automated. We'll cover this more in the section of this guide on measuring QA.

rainforest

Shift Quality Left

Section Two

How early can you start thinking about quality?

The earlier you find bugs, the faster you can resolve them. The later bugs are found, the more they cost to fix and take more development time away from your team. Shifting left to start the QA process as early as possible is the most effective way to minimize QA cycles for continuous delivery.

Continuous delivery aims to ship code in small increments frequently, as soon as possible; your testing process should mirror this "early and often" cadence. Continuous testing should start as soon as a feature is in development. Some teams start writing preliminary test cases during the feature spec process, then amend the tests as needed after the feature is developed.

The Changing Role of QA Teams

To learn more about the skills that QA teams need to be successful, check out our guide, "Level Up Your QA Career."

Continuous delivery is often implemented by teams that are lean and fast-moving. In some cases, that means that there's a very small in-house QA team (or none at all); 43% of organizations have a testing team of between 1-5 people (Practitest State of Testing Report, 2017). Whether you have a QA team of one or one hundred, the nature of continuous delivery will change the way that quality must be approached.

Continuous testing is marked by close collaboration between development, product and QA teams. There's no months-long testing phase, because testing must happen on an ongoing basis. As a result, the QA team must be well integrated with development activities. Members of the QA team will need to become more technical.

Functional Testing and Continuous Delivery

Unit testing is a critical component of any continuous integration process because of the fast, specific feedback it provides early in the development process. But teams overlook the value of starting functional testing early.

Early functional testing is as critical as unit testing to providing fast quality feedback early in development. However, where unit testing is now universally automated as part of the continuous integration process, functional testing often still requires a more hands-on approach, whether through writing automated test scripts or running tests manually.

Ideally, functional testing execution in a continuous delivery workflow can be as hands-off as possible. Many teams use testing automation tools like Selenium to execute tests automatically, but the management of automated scripts can slow down deployment just like manual test execution can. An alternative to automation is an on-demand crowdsourced platform like Rainforest, which offloads execution without the technical requirements of test scripting and maintenance.

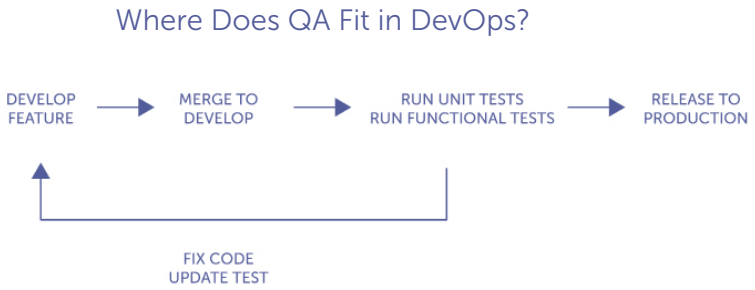


Fig. 2
Functional testing via Rainforest QA executed in parallel with unit tests.

Kicking off functional test execution in tandem with unit tests reduces the time required for a full QA cycle and provides more context when something breaks. For teams where developers own quality, this tandem approach provides more holistic quality insights in a short time frame.

Who Owns Quality in Continuous Testing?

Whether you have an in-house QA team or not, continuous testing spreads the quality assurance process across the entire development lifecycle. As a result, implementing continuous testing for continuous delivery requires consistent collaboration across teams. A product team might own the initial test drafting when they write feature specs, while developers write unit tests and refine functional tests after the feature is built. In some organizations, features may be handed back to the product team or QA team for test writing.

Regardless of the breakdown, speed is essential. Many teams find it useful to have a quality facilitator -- someone who is responsible for keeping track of the various moving pieces of a continuous testing process and holds the team accountable for their respective parts

"You will need to convince everybody on the team to genuinely embrace whole-team responsibility for quality. Testing time must be included in estimates for developer story work, and must actually be done. Do away with the expectation that others will catch mistakes: you are not there to be a safety net or gatekeeper."

-Mark Hrynczak, Quality Assistance Team Lead, Atlassian

rainforest

Continuous Test Case Management

Section Three

How can you scale test case management for the speed of CD?

The best way to manage test suites is to think small and think dynamic. Because code changes frequently for continuous delivery models, test cases and suites must be easy and fast to manage, execute and update. Test case management to fit the continuous delivery workflow and minimize the amount of time and effort required to keep test cases updated.

Modular Test Cases

Even lightweight test case management can add up. Modular test cases can be reused and updated easily, reducing the effort required to maintain a healthy test suite. You'll need a test case management system (TCM) that supports composable tests. Find a modular test writing and management system that lets you nest key flows within each other and update core tests across the entire database quickly.

Regardless of what TCM or execution method you use, break down test cases into small, manageable chunks. Additionally, focusing on building test cases around functional areas (such as features) helps ensure that your team builds impact-based coverage

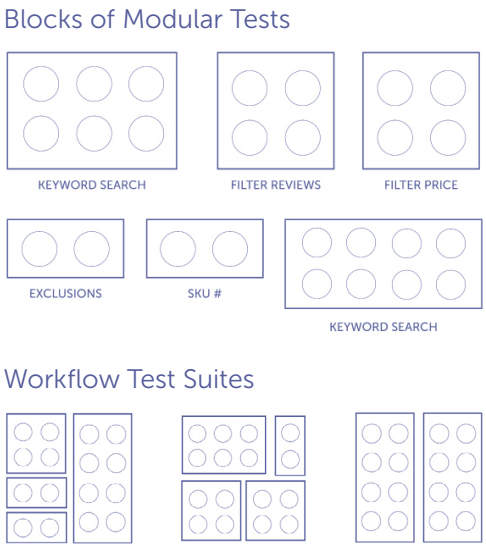


Fig. 3
An example of modular testing blocks and how they can be applied to a testing workflow

so your team spends less time on test management, and can execute and maintain tests more effectively.

Version Control

Learn more about this topic in our webinar on implementing version control for QA testing.

Because continuous delivery necessitates frequent, small changes to the codebase, corresponding test cases can easily become outdated. Whether you have a database of test cases run manually or automation scripts, frequent changes to the code will likely impact the accuracy of your QA results overall if left unmonitored.

Version control is standard practice for development, but continuous delivery requires it to become standard practice for testing as well. The advantages of version control are similar for software testing and software development; every version of the code will have associated test case versions.

For teams where developers own or are heavily involved in the QA process, using a version control system for test management helps ensure that test cases and code are always aligned. Version control reduces the flakiness of active test cases and helps teams better understand how their test coverage has changed over time.

rainforest

Integrations for CI/CD Workflows

Section Four

How can you make QA more consistent and less siloed?

Continuous delivery succeeds or fails depending on how well-automated and well-integrated your development tools and processes are. The same is true for continuous testing, which works most effectively when it is fully integrated into the development and delivery process. There are a few key points of integration into the development process that provide the best results for continuous delivery:

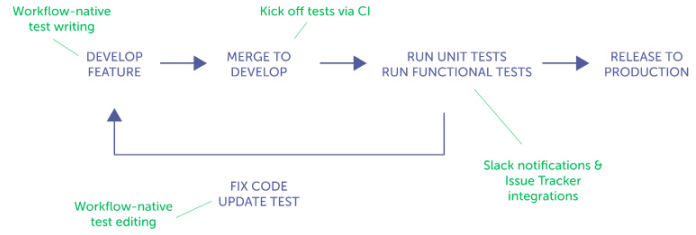


Fig. 4
Key testing integrations in the continuous delivery dev workflow

Continuous Integration

Recommended:
CircleCI
Jenkins

Integrating your functional QA workflow into your CI server is one of the most impactful things you can do to ensure the success of your continuous testing process. The primary benefit of using a CI tool isn't speed but consistency. By kicking off your functional tests via CI alongside your unit tests, you'll have clean, consistent execution for a strong quality baseline every single time.

To get the most out of using continuous integration for functional testing, you'll need your staging environment to be as close as possible to production. As recommended in the section on Shifting Quality Left, initiating automated functional test case execution at the same time as unit test execution provides fast, actionable feedback with minimal effort.

Top Choice:
Rainforest QA

Workflow-Native Test Writing & Management

Unit testing is already well integrated into the development workflow for CD, but functional testing is generally an afterthought for developers. Context switching between writing code and writing tests on different platform can take developers out of their flow and eats into work time.

Look for a test writing and management system that lets you add and make changes to your test cases using a text editor. For teams where developers are heavily involved in test writing, staying within their workflow will improve overall adoption.

Top Choices:
JIRA
Pivotal Tracker
Github Issues

Issue Trackers

Issue trackers help your team triage, track and organize issues, so integrating these platforms with the rest of your CD workflow is essential. Because all information is ported in automatically, retaining context is one of the biggest benefits of integrating test execution with your issue tracker, reducing the time required to replicate and resolve issues.

One potential roadblock for automatically generated issues is that if the team isn't properly notified of them, they can potentially be ignored. Don't let automatically generated issues slip through the cracks. Using different projects tags or other flags will help the team keep track of what needs to be triaged, assigned to a developer, or addressed in some other way. Organization also minimizes the time required to dig through duplicates.

Top Choice:
Slack

Communication Apps

Many teams now use a communication app that isn't specific to their QA process. If your dev team is responsible for QA, having bug alerts in the channels where they're already spending time improves communication and ensures that issues don't go unnoticed.

To use communication apps for QA alerts most effectively, focus on finding the right channel. For some teams, having a dedicated #bug-alerts channel might be the best way to keep signal from getting lost in the noise. For others, integrating those alerts into an existing channel might be a better fit.

Having a timely plan for what to do with these alerts is also critical, especially if these notifications will be the primary alert for new issues. Identify who is responsible for following up on issues as they come in, and make sure that a process is in place for triage and resolution to ensure that issues are addressed quickly.

rainforest

Measuring Quality

Section Five

""As we scaled up, we found that we had a lot of challenges with our existing automated CI tools, both in terms of how long it took to run those tests, and the overall stability of those test suites.

When a test takes 10 minutes to run and it occasionally fails on some flapping test that's ok, but when it takes 40 minutes to run and half of the tests are flapping, then you can't trust your suite." [4]

-Matt Sanders, SolarWinds

How can the impact of continuous testing be measured?

Measuring the success of your continuous testing process is essential to improving it. Focusing on a streamlined set of metrics helps give your team, your executive staff, and your board insight into the impact that your QA team is having on business goals. Every team's metrics will vary, but there are a few core measurements that every team should start off with:

Measuring How Long Issues Take to Fix

Understanding how effective your team is at finding and resolving issues is another essential area of QA measurement. These metrics will help you understand how efficiently your QA team (or tools) are doing their job, as well as how closely aligned your QA and development functions are. For organizations aiming for continuous delivery, measuring the speed of your QA processes is a necessary component of moving fast.

1. Time-to-Test
Turnaround time from kicking off a test run to getting results can mean the difference between delivering continuously and being bottlenecked. Measuring time-to-test will help you pinpoint exactly where your test suite is dragging down overall QA turnaround time. Understanding which tests require the most time to test will also provide insight into where automation or crowdsourcing might be applied to the greatest impact.

2. Time-to-Fix
What happens after a bug is identified is also an important indicator of the overall health of your QA process. For teams doing continuous delivery, it's not uncommon for developers to own their own issue triage and resolution. When time-to-fix creeps higher for developer-owned QA processes, it can be an indicator that devs don't have time for proper bug fixing, or they aren't having

issues surfaced to them effectively. For organizations with dedicated QA teams, a long time-to-fix can be an indicator of communication breakdowns or more significant process problems.

Measuring How Issues Effect Customers

Mapping your quality initiatives to business goals is essential to the long-term success of any QA strategy. These measurements are often the ones that will matter most to your exec staff and board members, and can create the most compelling case for the future of continuous testing at your organization.

The average NPS for a software company is 41.

- Delighted.com

3. NPS

Net promoter score (NPS) is quickly becoming an industry standard for measuring customer happiness, which can be mapped to product quality. There are a lot of factors that go into a high (or low) NPS score, so it's hard to draw a straight line from QA to NPS. But used alongside other metrics, NPS can provide a high-level, big-picture view of how successful your QA process is.

4. Escaped Defects

Even if you have thousands of test cases that return results in the blink of an eye, if bugs are reaching customers, then there's something missing from your QA execution. Measuring how many defects slip through the cracks and make it into product is one of the ultimate indicators of product quality. Be sure to track where these bugs are being reported: internally, by customers, or by smoke tests in production.

Measuring QA Efficiency

Some of your metrics should be focused on understanding the quality of your test database. They will help you understand where your QA or dev team should be spending their time, and help identify potential weak points before they become issues for customers.

5. Test Coverage

Test coverage can be a bit of a red herring metric; it's tempting to interpret coverage as a "more is better" measurement and focus on the number of test cases. Where test coverage is most useful is as more of a heat map, giving your team insight into areas of the product that might be overlooked.

6. Flakiness

A major pitfall of many automated testing methods is that test cases are not entirely reliable. If your test cases are lightning-fast but only provide reliable results a portion of the time, they can become a major resource sink. Measuring how frequently your test cases provide results you can have confidence in is an essential metric for understanding whether your automation strategy is providing ROI or not.

rainforest

Designing Your Continuous Testing Strategy

Learn More about Continuous Testing

To learn more about how teams can use continuous testing, including a case study of how Zenefits uses continuous testing as part of their continuous delivery workflow, download our guide "**Continuous Testing for CTOs.**"

When implemented effectively, continuous testing has the ability to unlock the full potential of fast-moving development teams and improve product quality at the speed of continuous delivery.

Focus on Aligning Workflows

Continuous testing requires thoughtful, strategic integration of testing throughout the development lifecycle. If there's a QA team, they must be closely aligned with both development and product teams to be effective.

Automate & Integrate as Much as Possible

Aim for ways to automate and offload work, rather than just changing the nature of the task. Focus on automating processes and linking tools together as efficiently as possible, without creating an excess of new work.

Get Cross-Functional Buy-In

Continuous testing requires collaboration across multiple disciplines. It's important for every team to understand the goals of continuous testing and how it will impact top-line business goals.

About Rainforest QA

Rainforest is changing the way QA is done in an era of continuous delivery. Our on-demand QA solution improves the customer experience by enabling development teams to discover significantly more problems before code hits production.

Hundreds of companies including Adobe, Oracle and Solarwinds use Rainforest to automate their QA testing process and easily integrate it with their development workflow via a simple API. Headquartered in San Francisco, Rainforest is a 2012 Y Combinator graduate funded by Bessemer Venture Partners and SVB Capital, among others.