SAUCELABS

WHITE PAPER

Automated Testing: The Glue That Holds DevOps Together

UPDATED MARCH 2022

Testing should never happen in a silo. In order to reap the full benefits of DevOps, organizations must integrate software testing into their continuous delivery pipelines. The software testing solution they choose should be highly automated, scalable, and secure, and it must enable fast, on-demand testing.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

DevOps, which originated in the late aughts, is now a well-established discipline. As of 2021, nearly three-quarters of organizations reported that they had adopted DevOps, according to a survey by RedGate Software.

But merely implementing DevOps is one thing. Doing DevOps efficiently, and leveraging the most possible value from it, is another.

A mistake many organizations make as they embrace DevOps is failing to integrate testing successfully into their DevOps processes. Without automating software quality tests as much as other stages of the CI/CD operations, businesses are likely to face software delivery delays, poor quality in production applications, or both.

Automated testing is therefore the key to successful integration of quality assurance into DevOps workflows. It is the only way to ensure that quality assurance is as continuous, agile and reliable as the rest of the DevOps operation.

Ultimately, automated testing is the glue that binds together all of the other processes that comprise the continuous delivery pipeline. Without automated testing, DevOps just doesn't work.

WHAT DEVOPS SHOULD ENABLE

To understand why automated testing is crucial for effective DevOps, it is necessary first to identify the benefits that organizations seek to achieve by following DevOps principles.

The primary benefits of a well-designed and maintained DevOps environment include:

- **Seamless communication across all parts of the organization.** DevOps does this by eliminating the silos that have traditionally separated different teams from one another. In many cases, those teams include not just developers and IT operations engineers – the "dev" and "ops" groups on whom DevOps originally focused – but a variety of additional stakeholders, like designers, security experts and – last but not least – QA professionals.
- **Software changes that are delivered on a rapid, continuous and reliable basis.** This requires software updates to be broken into small parts that can be designed, written, tested and put into production continuously, in contrast to the "waterfall" rhythm of traditional software delivery.

- **Maximum agility.** When software delivery is agile, applications in your toolchain can scale easily in response to fluctuations in demand. In addition, software delivery teams have the ability to switch easily between development frameworks and tools according to changing needs or preferences.
- **The elimination of unforeseen delays in software production.** These delays typically result from having to fix problems with code after it is in production, at which point rollbacks are costly and time-consuming. DevOps can help to avoid this risk by ensuring that code is tested automatically, as part of the continuous delivery pipeline, before it goes into production. That's especially true when you embrace "shift-left" strategies, which focus on identifying issues early in the DevOps pipeline, when they are easier to remediate.

These, at least, are the intended benefits of DevOps. Whether DevOps actually yields all of these benefits fully depends on how well your DevOps software delivery chain is organized, and how tightly your DevOps processes integrate with one another.

SOFTWARE TESTING'S ROLE IN DEVOPS

A common source of weak links within the DevOps software delivery chain is a lack of well-integrated software tests.

Indeed, although DevOps originally focused on collaboration just between development teams and IT operations teams, modern DevOps – as we've noted – requires a more holistic approach. Software testers also need to be seamlessly integrated into the continuous delivery chain and work alongside development and IT Ops teams. If software testers and test operations remain in a silo, separate from the continuous delivery chain, a number of problems are apt to arise, including:

- **Slow test results.** If automated software testing is not part of the continuous delivery pipeline, code changes cannot be tested continuously. They will instead have to be tested irregularly, whenever the testing team is able to address them. Under these conditions, testing becomes a bottleneck and the risk of releasing problematic code into production greatly increases, as does the likelihood of delays that prevent updates from reaching production continuously.
- **Lower agility.** Even if the rest of the software delivery chain is agile, failure to integrate automated testing into the continuous delivery

pipeline will undercut the organization's ability to derive value from that agility. Programmers will lack the ability to switch between development frameworks easily because they will not be able to assure that the testing team is ready to support the change. Software delivery will not scale because software tests cannot scale when they are not integrated with the continuous delivery chain.

- **Lower quality.** Part of the value of DevOps is its ability to standardize and streamline software delivery processes. When automated testing is not part of the continuous delivery pipeline, tests remain irregular and ad hoc. That undercuts the overall quality of the application being developed.
- **Frequent rollbacks.** To make the most of DevOps, the software delivery chain should be fully automated and continuous. The introduction of bugs that force developers to roll back code once it has been written is a serious hamper to continuity. Without automated testing, the likelihood of rollbacks is high, because code is pushed down the pipeline before it is tested properly.
- **Costly rollbacks.** DevOps means going from infrequent, large releases to frequent, tiny releases. In this environment, rollbacks are much less frightening, because they typically involve fewer features, less code, and much less risk. You still want to avoid rollbacks, but when you have to make one, a true DevOps practice will ensure that you know as soon as possible when you need to do one, and that you limit the "blast radius" when it becomes necessary. Automated testing at each step of the pipeline will ensure that when rollbacks are necessary, you can have confidence that regressions will be kept to a minimum.

For all these reasons, organizations that leave software testing in a silo rather than integrating it into the continuous delivery pipeline fail to achieve the value of DevOps—even if the rest of their DevOps operation is well designed.

AUTOMATED SOFTWARE TESTING: THE GLUE BETWEEN DEV AND OPS

So far, we've explained why testing and QA are part and parcel of a DevOps workflow. But that's not the only reason why integrating automated testing into DevOps is so critical.

Equally important is the role that automated testing plays in binding together the various teams and processes that comprise a DevOps software delivery pipeline. Tests are the medium that ties development to IT Ops, and testing ensures that updates flow smoothly from the beginning to the end of the continuous delivery pipeline.

To understand how automated testing links development and IT Ops, you must first recognize what a typical DevOps delivery chain looks like. It starts with developers, who design application changes and then write the code to implement them. And it ends with IT Ops, which is responsible for pushing the updates into production and maintaining them.

What happens in the middle of the delivery chain, between the development and operations phases? The answer is software tests – many types of them. Integration tests ensure that changes or new features written by developers can be added to the application without breaking it. Usability testing identifies flaws that developers might not have foreseen when designing code, and prevents those problems from reaching end users. Device compatibility tests ensure that code written and built in development environments will work as intended in real-world settings, which are much more complex and involve many more hardware and software variables than development environments.

This is why software tests are the glue that binds the code written by developers to the production-level application deployed by IT Ops. A continuous delivery pipeline that lacks automated, continuous testing will not enable developers and IT Ops teams to interact effectively with one another.

HOW TO ENABLE CONTINUOUS TESTING

To integrate software testing effectively into a continuous delivery pipeline, DevOps teams should implement testing solutions that enable and reinforce DevOps goals. When choosing a testing platform, look for the following essential features:

- **Support for a variety of frameworks.** The environments and code repositories that your DevOps team uses today are likely to change in the future. So are other aspects of your development process. For example, the web app that you develop today may be transformed into a hybrid app that takes advantage of HTML-based and native features at the same time. Since your development needs will change in ways that you cannot fully predict, it is important to seek a testing solution that can support a broad array of frameworks and adapt to your changing needs. Otherwise, you will undercut the agility of your continuous delivery pipeline because you will be wed to particular tools for developing and managing your code.
- **The ability to scale.** Your testing platform should perform tests as quickly as needed, and it should support as many parallel tests at one time as you

require. On-premises testing solutions are unlikely to offer the necessary scalability because they will be constrained by limited hardware resources. In contrast, a cloud-based testing platform can scale as seamlessly as the rest of your continuous delivery pipeline.

- **The ability to test quickly.** To avoid delays to your continuous delivery chain, you need to perform tests quickly. Performing parallel tests on a large scale is one way to achieve this. Another is running compatibility tests on simulated devices first (since these tests scale better), and performing more time-consuming tests on real devices later in the pipeline, just before code enters production.
- **High automation.** DevOps teams achieve their speed and agility by automating as much of the software delivery process as possible. Your testing solution should be as automated as the rest of your DevOps toolset. You should be able to trigger tests, analyze results and share testing information across the organization (using features like ChatOps integration) in a completely automated fashion. While manual tests will always be part of the process for User Experience testing, the rest of your testing should be as automated as possible.
- **On-demand testing.** To avoid kinks in the continuous delivery pipeline, you need to be able to perform tests whenever they are necessary. There are two ways to do this. One is to maintain a massive on-premises testing environment with enough resources to perform tests whenever you need them. The other solution is cloud-based testing, which can complete tests quickly upon request. The latter is much more cost-efficient because it obviates the implementation and management of an expensive on-premises test grid that will sit idle most of the time. Cloud-based testing also allows you to avoid false failures generated by on-premises test grids, and the inefficiencies that arise from finding and fixing bugs in on-premises test infrastructure.
- **Security.** In DevOps, all members of the team—including software testers—have an important role to play in keeping applications secure. Testing platforms, therefore, need to include security features, such as the encryption of test data over the network, and robust access control policies.

These qualities allow your developers and IT Ops teams to work together as efficiently as possible. They ensure that the code you put into production is reliable and stable across diverse environments. And they empower your organization to derive full value in the migration to a DevOps-based workflow by maximizing the agility, visibility, scalability and continuity of your software delivery pipeline.

SAUCELABS

ABOUT SAUCE LABS

Sauce Labs is the leading provider of continuous test and error reporting solutions that gives companies confidence to develop, deliver and update high quality software at speed. The Sauce Labs Continuous Testing Cloud identifies quality signals in development and production, accelerating the ability to release and update web and mobile applications that look, function and perform exactly as they should on every browser, operating system and device, every single time. Sauce Labs is a privately held company funded by TPG, Salesforce Ventures, IVP, Adams Street Partners, and Riverwood Capital. For more information, please visit saucelabs.com.

saucelabs.com/sign-up

FREE TRIAL

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