WHITE PAPER

THE DEVOPS JOURNEY: FROM WATERFALL TO CONTINUOUS DELIVERY

Implementing an agile, DevOps-centered workflow involves several distinct steps. In other words, the process required to evolve from traditional, "waterfall"-style development to continuous delivery of software is a journey, not something that organizations can implement all at once. While many organizations today have begun the DevOps journey by adopting some tools and methodologies that promote agility, a lack of fully automated testing often prevents them from completing it. The integration of efficient automated testing into workflows is therefore a crucial step for organizations seeking to achieve full agility and complete the DevOps journey.

SAUCELABS

TABLE OF CONTENTS

EXECUTIVE SUMMARY

DevOps. Agility. Continuous delivery. These are the buzzwords of the new software development and testing landscape. They describe some of the core values and processes that characterize an efficient, modern workflow.

But they are also terms that can be misleading. That's because they're easy to conceptualize as qualities or values that an organization either has or does not have. In many common depictions, a company either "does" DevOps or it doesn't. It's agile or it's not. It delivers its software continuously, or it adheres to archaic, staccato delivery schedules.

In reality, however, the DevOps landscape is not black and white. Adopting DevOps is a journey, not a choice that an organization opts to make or not, and implements all at once. That journey entails various steps and stages.

This whitepaper explains the DevOps journey and identifies the phases that the typical organization crosses as it adopts DevOps practices. It highlights the various degrees of agility associated with each stage, then discusses the barriers, including manual and sequential testing, that organizations have to overcome in order to progress further toward the endpoint of the DevOps journey, which means reaching full continuous delivery.

STAGES OF AGILITY

Because modern software delivery chains involve so many tools and processes, it is best to think of the DevOps journey as a continuum. Organizations progress slowly from one phase to the next. They do not make the jump between phases overnight.

That said, it is possible to identify four main stages of agility within the DevOps continuum. They include:

- **Waterfall.** Organizations with a waterfall delivery process are living in the yesteryear of software development. They write, test and deliver code according to a sequential staccato rhythm. Their programming, Ops and quality assurance teams operate in silos, without collaborating with one another. They rely on manual processes, including manual testing.
- **Fast waterfall.** Organizations that have implemented a certain degree of agility are in the fast waterfall stage. They still deliver software sequentially, but they take advantage of tools, such as GitHub, that introduce a certain degree of automation and scalability to their workflow. They also do some amount of automated testing, but constraints remain; for example, they

may not take advantage of parallel testing, which adds a magnitude of efficiency to automated testing.

- **Continuous integration.** This stage of the DevOps journey is hallmarked by the use of continuous integration platforms, such as Jenkins, CircleCI and TeamCity. These tools do much to automate the process of building software. They also provide a framework that helps developers, Ops and quality assurance teams to operate in sync. However, running a continuous integration server and eliminating collaboration silos within the organization does not guarantee full agility. That requires a fully automated and optimized testing process. It also involves next-generation approaches to deployment, such as the use of scripted infrastructure.

- **Continuous delivery.** This is the final phase of the DevOps journey. Organizations that have achieved continuous delivery have adopted a fully automated development process, including automated testing at high frequencies. In addition, their development, Ops and quality assurance teams function as a single group, constantly in sync with one another.

	Waterfall	Fast Waterfall	Continuous Integration	Continuous Delivery
Process	Traditional sequential design model	Initial adoption of Agile	Full adoption of Agile	Fully automated Development process
Tools	**Manual testing dominates**	**Automated testing begins**	**Automated testing dominates;** manual only for debugging	**Automated testing core to Dev. & Delivery**
People	Dev. & QA completely separate	Dev. & QA start communicate	Dev. and QA collaborate closely	Dev. and QA functions merge

Description of the four major stages software development groups go through on their journey to Continuous Delivery.

Again, in practice, these stages stretch across a continuum. The line separating each phase from another is blurry. In addition, the specific characteristics associated with each phase will vary from organization to organization. There is no single tool or practice that necessarily demarcates one phase of the DevOps journey from another.

In general, however, these stages characterize the major landmarks by which a company passes as it makes the long journey from traditional software delivery to continuous delivery.

MOVING DOWN THE TRAIL WITH AUTOMATING TESTING

Progressing from one phase to another within the DevOps spectrum requires simply identifying the parts of the delivery chain that are not yet as agile as they could be, then making them more agile.

In many cases, software testing is high on the list of barriers to agility that organizations could easily address. Yet it is also likely not the first place that DevOps teams think to look when they seek ways to increase agility. In fact, many organizations may believe they are already fully agile because they have adopted other DevOps tools and overlook the pitfalls of inefficient testing.

This tendency is not surprising. After all, the most popular DevOps tools that have appeared to date -- continuous integration servers, code repositories, containers and the like -- focus primarily on making the production, management and delivery of source code more efficient. They do little to address the testing piece of the puzzle.

As a result, organizations that have embarked on the DevOps journey by adopting tools related to development, delivery and production are likely to find their agility constrained by a lack of automated testing -- even if they are not aware of it. For these organizations, improving testing efficiency is an obvious and easy way to grow more agile.

INCREASING TESTING AGILITY

Fortunately for such organizations, a number of DevOps-inspired tools and methods exist to improve the efficiency of the testing part of the delivery pipeline. They include:

- **Automated testing.** Automated tests are a sine qua non of an efficient, DevOps-based delivery pipeline. Manual tests impose enormous bottlenecks on an otherwise automated workflow, since they require the software development and delivery process to pause whenever tests need to be performed. Manual testing is also not scalable because the number of manual tests that an organization can perform is limited by the size of the staff on hand.

- **Cloud-based testing.** Moving testing to the cloud increases agility in two main ways. First, it makes tests more scalable, since it is much easier and faster to acquire additional cloud-based infrastructure than it is to expand on-premise infrastructure. Second, cloud-based testing can improve test speed by providing access to more resources than those available locally.

- **Parallel testing.** Rather than running tests sequentially, organizations can become more agile by running them in parallel when possible. Parallel tests mean that the full test suite is completed faster. They also prevent hold-ups in the delivery process, which would occur if a problem with one sequential test delays other tests.

- **Shift-left testing.** Even if organizations have already adopted automated tests, they may be able to derive additional agility by shifting those tests to the left. In other words, they should test earlier in the development and delivery process, so that problems can be discovered and fixed sooner. That leads to faster overall delivery while also increasing the DevOps team's ability to create and test new features without delaying the delivery pipeline.

These strategies for improving agility by making testing more efficient have so far not been a central part of the DevOps conversation. But they constitute simple, low-cost ways of growing more agile in order to work toward the goal of full continuous delivery.

AUTOMATED TESTING IS NOT A CHOICE -- IT'S A NECESSITY

On that point, it's worth noting that, in order to be fully agile, automated testing is not simply a choice, but an absolute necessity. Even if an organization has in place all of the other tools and practices that constitute a DevOps delivery chain, inefficient testing will keep complete agility far out of reach.

That means that automated tests, combined with other methods of optimizing testing as outlined above, are not just one of many optional tools that an organization could potentially choose to include in its workflow if it wishes to maximize agility. Automated tests are instead an essential ingredient.

Web apps: From waterfall to continuous delivery

To understand why automated testing is so essential to achieving full agility, consider the example of an organization that has been developing a Web app for the past decade. The organization has done much over that time to take advantage of DevOps-inspired tools and techniques. It has moved its code to a GitHub repository. It has adopted Jenkins, the continuous integration tool, to automate builds. It has assured that the programmers who write the app's code can easily communicate with the QA team that tests it and the IT Ops staff who deploy it.

This organization, in other words, has completed a large portion of the DevOps journey. But it lacks one crucial DevOps resource, which is automated testing. Instead of fully automating its tests, the organization relies mostly on manual tests. It scripts some tests, but other tests are performed on an ad-hoc basis, and the team has not integrated automated testing into the build process.

The lack of full test automation means that delivering quality-assured versions of the new app to users on a rapid basis remains impossible. This undercuts the value of the other changes that the organization has implemented to its development and delivery pipeline for the Web app. Those changes have optimized development and deployment, but without an automated testing procedure, delivering the app to users on a continuous basis is not possible. The result is a workflow that is much more agile than it was before the organization began its DevOps journey, but has not yet reached the continuous delivery stage.

Fortunately, taking the next DevOps step in the scenario described above is easy enough. The organization needs simply to build fully automated testing into its workflow. One obvious way to do that, based on the organization's needs and the tools it already has in place, is to take advantage of tools like the generic Selenium plugin for Jenkins, which would allow the organization to make automated tests a part of the continuous integration workflow by having Jenkins run them as part of builds. That would add a significant amount of agility.

However, the organization could gain even more agility by leveraging the optimized Jenkins plugin from Sauce Labs, which provides broader coverage and delivers faster test results by taking advantage of cloud-based testing. This tool enables developers to manage, execute and review test results all from within Jenkins, making them more efficient.

WHY AUTOMATED TESTING IS UNIQUELY IMPORTANT TO DEVOPS

Automated testing's essential role in achieving continuous delivery makes automated tests different from many other DevOps tools. For example, containers are a common part of the automated delivery pipeline, but they're not a strict necessity for a fully agile workflow. Depending on the type of app a company builds, containerizing it in order to make delivery more agile may not make sense; storage apps, for instance, are generally difficult to run inside containers.

As another example, GitHub is a massively popular way to host code and track changes for DevOps teams. But GitHub is not the only solution of its kind. And in some cases, especially where development teams are very small, it may be a poor fit for an agile workflow.

However, any organization that builds software needs to test that software, and automated testing is the only means of performing such tests in a way that is efficient and agile. Automated tests are therefore an unusually and absolutely crucial part of a DevOps delivery chain. They represent a clear resource to adopt for organizations seeking to take the next step on the DevOps journey.

CONCLUSION

DevOps is a central part of efficient software delivery. Most companies seeking to keep their users happy, and their developers and admins working at their best, have already begun the DevOps journey. In order to reach the continuous delivery stage and achieve full agility, however, companies need to do more than adopt a handful of DevOps tools or begin speaking in DevOps-inspired language.

Reaching the final stage of the DevOps journey takes commitment. Organizations need to overhaul the practices they established in the age of waterfall development. They also need to ensure that all stages of their workflow are fully agile, which requires a complete set of DevOps tools. And while many of the tools that organizations adopt along the DevOps journey will vary from company to company, automated testing is an essential resource for virtually any organization that wishes to reach continuous delivery.

ABOUT SAUCE LABS

Sauce Labs provides the world's largest cloud-based platform for the automated testing of web and mobile applications. Its award-winning service eliminates the time and expense of maintaining an in-house testing infrastructure, freeing development teams of any size to innovate and release better software, faster.

Sauce Labs is a privately held company funded by Toba Capital, Salesforce Ventures, Triage Ventures and the Contrarian Group. For more information, please visit saucelabs.com.

SAUCELABS

SAUCE LABS INC. 539 BRYANT STREET #303 SAN FRANCISCO, CA 94107 USA