How automated testing enables DevOps

Test automation helps development teams build, test, and ship faster and more reliably.

ANTON HRISTOV

Product Manager at mabl

In the early 2000s, companies began adopting agile practices, embracing an accelerated development lifecycle marked by frequent customer feedback. This later drove the adoption of tools that enable continuous integration and continuous delivery that automated the build, test, configure, and deploy processes.

However, key functions such as development, testing, and delivery to production were performed by separate teams that operated within their own silos. This caused inefficiencies and bogged down the software development lifecycle. It also led to DevOps, the organizational philosophies, practices, and tools that enable small, cross-functional teams, also known as squads, that are responsible for the continuous delivery and quality of product updates, end-to-end.

At first, DevOps unified just development and IT operations, while testing continued to be performed by a separate team in a largely manual fashion. This helped address the challenges of cloud application delivery and monitoring. It also led to the creation of fully automated CI/CD pipelines. However, it didn't result in much faster release cycles, since testing was siloed and often a time-consuming, manual process.

To address the testing bottleneck, organizations are now moving away from centralized QA teams to embedding QA across the entire development team.

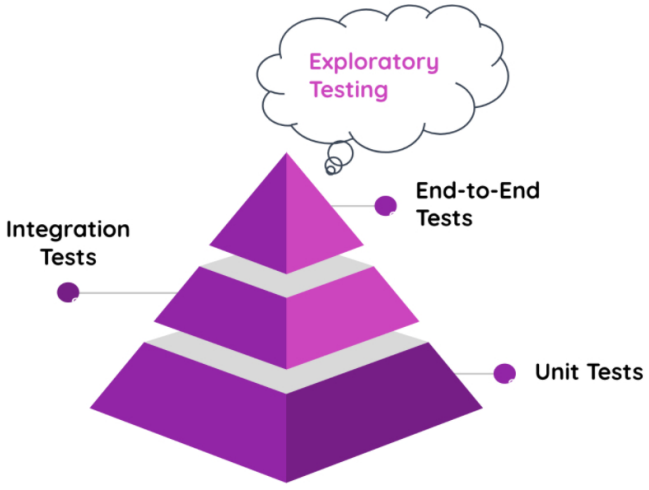
What is test automation?

Test automation is the practice of automatically reviewing and validating a software product, such as a web application, to make sure it meets predefined quality standards for code style, functionality (business logic), and user experience.

Testing practices typically involve the following stages:

- **Unit testing**: validates individual units of code, such as a function, so it works as expected
- **Integration testing**: ensures several pieces of code can work together without unintended consequences
- **End-to-end testing**: validates that the application meets the user's expectations
- **Exploratory testing**: takes an unstructured approach to reviewing numerous areas of an application from the user perspective, to uncover functional or visual issues

The different types of testing are often visualized as a pyramid. As you climb up the pyramid, the number of tests in each type decreases, and the cost of creating and running tests increases.



Historically, all testing within the pyramid was performed manually. This was a slow, expensive, and error-prone process until automated testing tools were created.

Today, nearly all unit tests are fully automated and unit testing automation is considered a best practice. Integration tests are also largely automated and, if not, are typically skipped in favor of more manual end-to-end testing. The current wave of test automation efforts is largely focused on automating the end-to-end layer of the testing pyramid, which reduces the need for integration tests.

Even though automation tools have existed for over a decade, many require coding skills and often result in flaky, brittle tests that are extremely costly to troubleshoot and maintain at scale. Many teams end up creating their own custom test automation frameworks, which makes it difficult and time-consuming to onboard new team members due to the steep learning curve. Custom frameworks also end up requiring

their own maintenance and improvements to keep up with the changing technology stack. As a result, most end-to-end testing was a manual process -- until now.

As organizations mature their DevOps practices, the need for test automation across the lifecycle is important to unlock the key benefits of DevOps – the ability to build, test, and ship faster and more reliably, streamline incident responses, and improve collaboration and communication across teams. It is no longer an option to have a release build sit with the QA team for several days before developers receive feedback and fix identified issues. QA teams need to align their efforts in the DevOps cycle by ensuring test cases are automated and achieve near 100% code coverage. Environments need to be standardized and the deployment on their QA boxes should be automated. Pre-testing tasks, cleanups, post-testing tasks, etc. should be automated and aligned with the continuous integration cycle.

There are now low-code tools like mabl that make it possible to incorporate reliable and automated end-to-end tests at every stage of the CI/CD pipeline, which helps catch issues much earlier in the development lifecycle. It's well known that the earlier you detect problems with a release, the faster and cheaper it is to fix them.

Automated testing in DevOps

> DevOps makes testing a shared responsibility of the entire team, while test automation enables developers to ship code changes quickly with high confidence in quality.

In practice, this means that developers gravitate towards writing unit tests to validate the code works as expected, while quality practitioners and product owners create automated UI tests that validate the end-to-end user experience. Quality practitioners also organize exploratory testing sessions where the team manually examines various application areas for issues.

A DevOps best practice is to run automated tests as early and as often as possible within the CI/CD pipeline. This includes running automated UI tests in production to proactively monitor for user experience issues. Because today's applications rely on numerous services with multiple moving parts, performing synthetic transaction monitoring by running tests in production can detect issues with third-party services before your users do.

RELATED MATERIAL

Get started for free

Learn more →

RELATED MATERIAL

Learn more about DevOps tools

Read more →

Getting started with automated testing

There is no one-size-fits-all solution but here are some important things to consider when defining a test automation strategy for your team:

Frequency of release

The more frequent the releases, the more you need to invest in test automation, especially in end-to-end tests that should run on every deployment. If you don't have a frequent release cycle and would like to accelerate it, you can start by adding more unit test coverage and creating simple automated UI smoke tests to perform a quick sanity check on every build. You can then gradually invest in creating more automated end-to-end tests that help you reduce the time it takes to check a release for regressions.

Tools availability

Modern test automation tools will significantly improve your team's ability to continuously deliver high-quality software. When evaluating testing tools, consider easy test creation, reliability, need for maintenance, and integration with your CI/CD stack.

It's equally important to understand the learning curve and required skills for a given tool. The easier your solution is to use, the faster your team can ramp up. And it will be more accessible to more people on your team, which can lead to increased test coverage and help cultivate a culture of quality. One effective way to evaluate testing solutions is to have the entire team spend time automating a few test case scenarios with leading contenders on your shortlist.

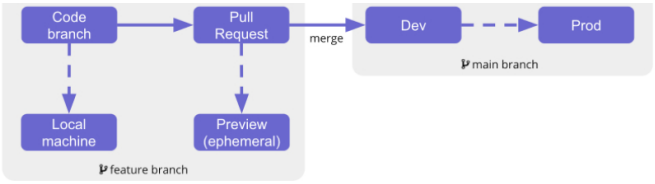
Product maturity

If your team works on a product with numerous existing customers and a mature codebase, chances are you already have an established release cadence and testing practices. As your team moves to continuous integration or full CI/CD, it's important to include test automation as a key part of pipeline automation. Fast delivery and fast feedback are unsustainable without automating testing earlier and throughout development.

On the other hand, if your team is building a new product, it's an ideal opportunity to instrument automated testing from the beginning. Right out of the gate, set a goal for unit test coverage and focus on defining the end-to-end test cases for each feature. It's best to wait until a feature is close to a release before adding automated end-to-end tests so you avoid test failures due to breaking UI changes.

CI/CD environments and testing data

Creating automated tests is a challenge in itself, but often it's the lack of pristine environments with test data that prevents teams from adopting test automation earlier in the CI/CD pipeline. Therefore, it is important to have a team discussion early about the testing strategy and commit to creating the necessary testing infrastructure. For example, developers need to implement support for test user accounts and have the ability to load an environment with test data via an API. Building infrastructure for provisioning ephemeral test environments early will significantly speed up the release review and feedback cycle.



How does automated testing change the role of QA?

DevOps elevates the role of quality professionals to a strategic level and presents amazing opportunities for career growth.

In the past, the role of QA focused primarily on executing testing activities – writing test cases, executing manual tests, and reporting issues to the developers. There were typically only a few automation engineers in the product organization, while the majority of quality professionals were manual testers. The reason was test automation engineers needed a strong technical background, some development skills, strong communication skills, and a solid understanding of business requirements. Historically there was a high demand and low supply of people with this unique skill-set. Therefore product teams relied heavily on manual testers for quality assurance.

However, DevOps changes everything. With multiple releases to production per day, testing a build needs to take minutes, not days. Software teams need qualified professionals to lead and coach the rest of the team - especially developers - by advocating for the user, teaching best practices, and helping achieve the benefits of end-to-end testing.

Fortunately, low-code test automation tools like mabl can help manual testers or quality analysts to become automation engineers. As the time spent on manual testing is reduced, QAs can dedicate more time to playing the strategic role of quality coach on the team and help everyone participate in the quality assurance process.

How automated testing powers DevOps

Automated testing is now considered a DevOps best practice. Implementing automated tests across a large portion of your development pipeline may seem intimidating at first, but you can start by automating a single end-to-end scenario and running that test on a schedule. New tools also make automated testing easier than ever and the results are more than worth it. After all, who doesn't want happy users?

Embracing automated testing helps unlock the following DevOps benefits:

- **Speed without sacrificing quality**: gain high product velocity that makes developers happy and enables them to deliver more value to users, faster
- **Improved team collaboration**: shared responsibility for quality empowers better collaboration among team members
- **Reliability**: improve the reliability of releases by increasing coverage with test automation. Issues in production should be a rare occurrence rather than the norm
- **Scale**: produce consistent quality outcomes with reduced risk by distributing development across multiple small teams that operate in a self-sufficient manner
- **Security**: move quickly without compromising security and compliance by leveraging automated compliance policies, fine-grained controls, and configuration management techniques
- **Increased customer happiness**: improved reliability and fast responses to user feedback increases user satisfaction and leads to more product referrals

In conclusion...

Embracing test automation to unlock the full potential of DevOps will ultimately reduce bottlenecks and increase efficiency, both of which will have a direct impact on employee and customer happiness, and ultimately the bottom line.

Begin automating your testing with Bitbucket Pipelines or one of the many test automation tools and resources available on the Atlassian Marketplace.

ANTON HRISTOV

Anton Hristov is a product manager at mabl and has more than 10 years of experience in software development, testing, and delivery. He enjoys working at the intersection of people, technology, and design. Anton particularly loves deriving insight from data and helping people realize their full potential, which is reflected in the products he helps bring to life.

SHARE THIS ARTICLE

NEXT TOPIC

DevOps Tools →

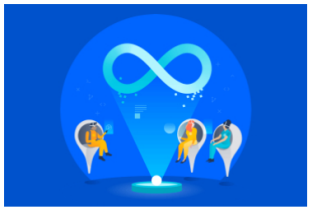
Recommended reading

Recommended reading

Bookmark these resources to learn about types of DevOps teams, or for ongoing updates about DevOps at Atlassian.

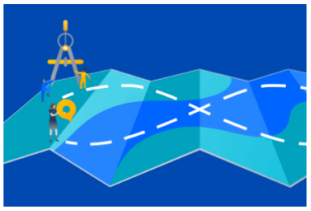
DevOps community

Learn more →



Simulation workshop

Learn more →



Get started for free

Learn more →

Sign up for our DevOps newsletter

Email address

Sign up