Octopus Deploy

THE DEVOPS ENGINEER'S HANDBOOK

Delivery

TABLE OF CONTENTS

Software delivery involves repeated iterations to update a software application. To have confidence in new code updates, software engineers rely on software tests to screen for any other errors.

Software testing is the process of verifying that software behaves as intended. This verification covers all stages of the delivery pipeline. A comprehensive testing environment helps each iteration of the DevOps loop strengthen the quality of the product. A weak testing approach can mean defects progress to release, and developers must fix bugs while the product is live.

Software testing has proven to be a strong indicator of organizational success. A survey by Mabl on the state of testing in DevOps indicates that automated testing (at least 4–5 different types of tests) is key to customer happiness. The 2021 State of DevOps DORA Report reveals that continuous testing is an indicator of success, with elite performers who meet their reliability targets being 3.7 times more likely to use continuous testing.

Test automation and continuous testing as part of continuous delivery drive software delivery success. You should have tests for functionality and quality attributes running as part of your deployment pipeline. Continuous delivery requires different types of software tests. If you want to learn more about the different types of software tests, you can read about them on our blog.

While the technical implementation of testing is essential, organizational culture may be just as important as establishing a robust testing environment in your company. Since testing is about uncovering and dealing with failure, how your culture treats failure can

NEXT ARTICLE
Deployments vs releases

How culture helps software testing

In Westrum's typology of organizational cultures, cultures are divided into pathological, bureaucratic, and generative categories. These cultures can be observed in any industry and have common traits.

Pathological	Bureaucratic	Generative
Power oriented	Rule oriented	Performance oriented
Low cooperation	Modest cooperation	High cooperation
Messengers shot	Messengers neglected	Messengers trained
Responsibilities shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to inquiry
Novelty crushed	Novelty leads to problems	Novelty implemented

The 2022 State of DevOps Report suggests

High-trust and low-blame cultures tend to have higher organizational performance.

In an environment where failure is common, it is important to be low blame, where a failure leads to the inquiry of the root cause rather than scapegoating. The DevOps cultures with high organizational performance are generative. Generative cultures share information with the right people, which leads to critical feedback in the system at the right time. The effective sharing of information affects software testing. Any error should be transparent and known to other teams. If there is any block of information due to the culture, teams lack transparency and cannot act on complete details.

Generative cultures encourage teams to share responsibilities. In a bureaucratic culture, teams are only concerned with their domain and work in isolation. In a generative culture, the software, QA, Ops, and UX engineers share joint responsibility for the end product. If QA finds an error introduced by development, the two teams will work together to solve the issue rather than leaving it to the team that discovered the error.

A generative culture encourages innovation and experimentation. In this environment, engineers can task risks and try new things without fear of blame. Freedom to innovate leads to more tests being written and triggered. A culture that stifles creativity may incentivize engineers to write code that never fails for fear of being blamed. If you create an environment where failure is accepted and learned from, innovation can happen.

Common testing tools

There are a variety of tools you could use to implement software testing. Here are some commonly used in industry:

- **Selenium Web Driver:** A web framework where you can execute cross-browser tests
- **Cucumber or SpecFlow:** A testing tool for Behavior Driven Development (BDD)
- **Appium:** Open source testing automation tool for web apps
- **Robot framework:** Open source automation framework used for test automation and robotic process automation (RFA)
- **Sauce Labs:** DevOps test toolchain for mobile and web applications
- **Smart Bear:** A toolchain for software development that includes a testing framework
- **Other:** There are many other testing framework tools that you can use outside of the listed tools.

Conclusion

Testing is an important part of the continuous delivery lifecycle. Testing can give you confidence in your software application as you progress releases into production. Research has shown that software teams that test more are more reliable and have happier customers.

In addition to implementing the right software tests, culture plays a vital part in a testing strategy. Using Westrum's typology of organizational culture, generative cultures with high-trust and low-blame environments can contribute to successful organizational performance.

Software testing in continuous delivery is a mixture of the right tools and culture to help give you happier teams and customers.

For more detailed information about specific testing use cases, check out our blogs on testing.

Terence Wong

Tuesday, September 27, 2022

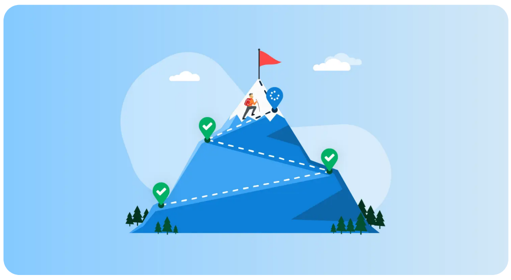
Tags:

- Testing

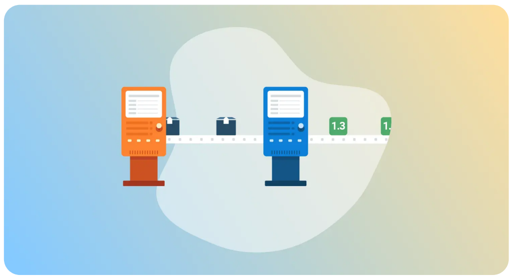
Categories:

- Continuous Delivery

More resources



Capabilities needed for Continuous Delivery



Automation's importance to Continuous Delivery

Read more →

Read more →

CATEGORIES

Continuous Delivery

DevOps

Platform Engineering

Site Reliability Engineering

TAGS

Automation

Books

CI-CD

Deployments

QUICK LINKS

Octopus Slack Community

Octopus Deploy

Back to top