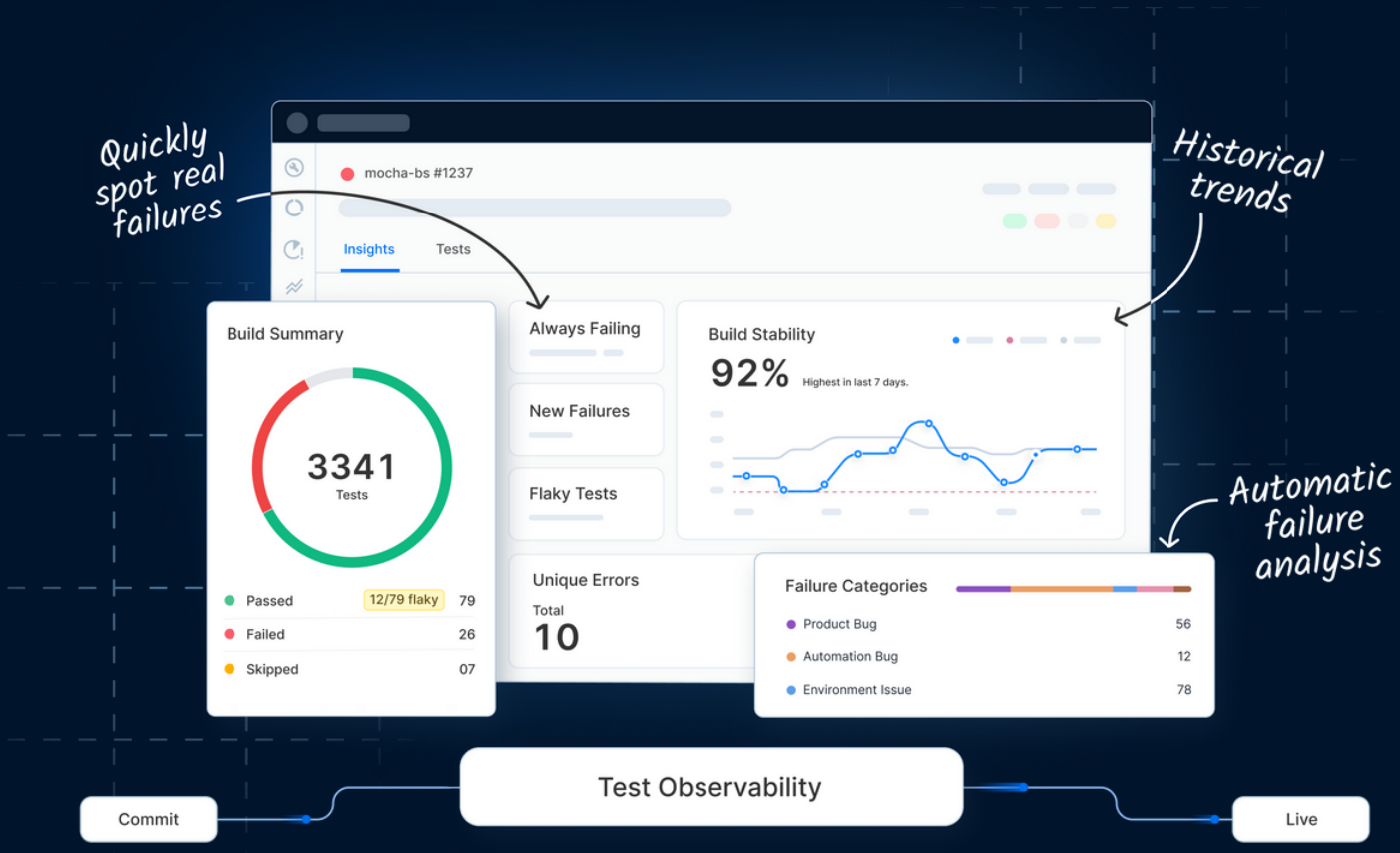
BrowserStack

Test Observability

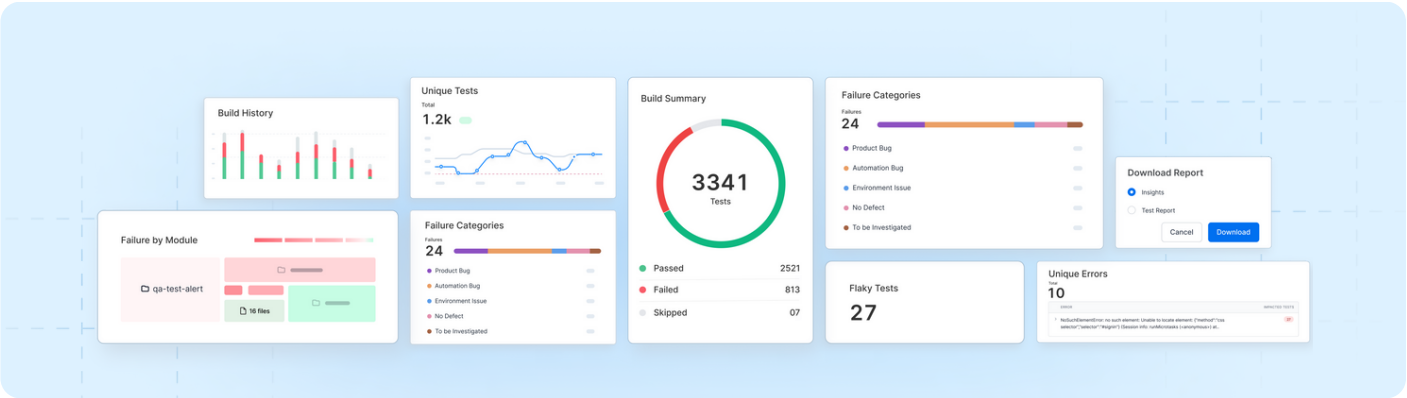
Improve test stability and reliability



TRUSTED BY GLOBAL ENTERPRISES

mastercard Microsoft The Washington Post

RBS cisco HSBC Discovery



Executive Summary

In today's digital-first world, organizations are constantly adapting to ever-evolving customer demands. To win customer loyalty, releasing bug-free code more frequently is paramount. Teams often turn to continuous testing and test automation to achieve this but face significant challenges in maintaining the health, reliability, and stability of their test suites.

Take flaky tests, for instance. In a recent survey, 59% of software developers reported regularly encountering flaky tests. Among the 91% of developers who dealt with flaky tests at least a few times a year, more than 75% viewed them as a moderate-serious category issue. Moreover, increasing test automation leads to an unorganized mess of test result data scattered across platforms, making manual analysis a cumbersome task.

Teams struggle with differentiating false failures from genuine ones, rummaging through multiple log sources to determine the reasons behind failures and anomalies. This process often involves switching between multiple tools to file bugs, create analysis-ready reports, and manually track data for calculating QA metrics, leading to errors and delays that cost valuable time and resources.

Teams grappling with test suite quality issues like this need a comprehensive solution to improve the reliability of their test suites for fast and accurate run verification.

In this whitepaper, we delve into the importance of BrowserStack Test Observability for Dev and QA teams. We'll show you how it can help ensure quality test suites, enhance CI stability, and boost productivity to accelerate your time to market.

Challenges of evolving testing needs

Evolving testing needs could be jeopardizing speed, productivity, and visibility

As test automation matures, test suites grow exponentially. With increasing test automation, teams often end up drowning in a jumble of unorganized test result data scattered across different platforms. Manually making sense of chaotic test data is a nightmare. It's hard to find critical failures, debug them, and identify areas for improvement in test suites.

Let’s look at the challenges in detail:

Manual effort

A lot of manual effort is required to improve the signal-to-noise ratio, leading to:

Wasted time and effort

Teams invest time and effort in repeatedly identifying and isolating flaky, always failing, tests failing due to environment issues, etc., which delays the identification of genuine test failures. Imagine the frustration of having to manually debug and rerun tests, relying solely on intuition and guesswork to figure out if a test is flaky or failed due to a genuine error. The process is tiring, prone to human error, and non-scalable.

Bottlenecked deployment and increased lead time

Manual methods almost always slow things down, making releases less frequent and increasing the lead time for any changes. And that's a big roadblock when you're aiming for full-fledged Continuous Delivery. Besides, automation engineers have to spend time on run verification instead of writing newer automation scenarios. This results in the QA team getting almost always stretched.

Unreliable results

Teams have to re-run the entire test suite multiple times for deterministic results, leading to:

Delays

Regression time goes up 2-3x due to re-runs. It also slows down development because teams need to wait for deterministic results.

Resource wastage

Each new re-run contributes to additional resource time, effort, and cost wastage.

Inefficiencies

Multiple people in the same team often analyze the test failure root cause over and over again. It's a recipe for inefficiency. Plus, the lead time to make changes increases significantly.

Disparate data streams

It is cumbersome for teams to analyze and correlate different logs, leading to:

Frustrating RCA

Logs are often spread across multiple tools in different formats. Teams must correlate these with internal playbooks, if documented, or rely on memory to determine the cause of test failure.

Endless debugging

Teams get stuck in a never-ending loop of manually categorizing failures. The analysis takes up a lot of time, slowing down the debugging process.

Longer time to market

All the extra time and effort spent in RCA and debugging prolongs Dev-QA and deployment cycles and increases time to market.

Blindspots and gaps

Teams lack visibility and can’t measure the health of their test suites, leading to:

Unrealible progress tracking

Without means to measure the current health of test suites, teams cannot track the progress, efficiency, and effectiveness of QA over time.

Gaps in testing

The lack of metrics and data makes it difficult for teams to pinpoint gaps in test coverage, stability, and performance over time.

Ineffective roadmap planning

When visibility is limited, it's hard to determine which areas of the test suite need improvement and in what priority.

Flaky delivery

Teams struggle to identify flaky tests and the root cause, leading to:

Ignored flaky tests

Teams often unintentionally ignore flaky tests, always-failing tests, slow tests, etc., to keep up with release cycles. The flaky tests, as a result, remain in the test suite for a long time.

Loss of trust

Flaky tests return nondeterministic results. When teams are unable to trust test results, developers can't trust their code. They turn to manual testing to gain confidence before releasing, which slows down the entire cycle and increases the chances of bugs in production.

Lowered confidence in QA

When teams can't deterministically tell why and which tests are flaky, they start to doubt the effectiveness of testing and skip tests. Bugs creep into production as a result, lowering the organization's confidence in QA and impeding efforts for CI/CD implementation.

Introducing BrowserStack Test Observability

BrowserStack Test Observability was created especially for engineering teams looking to optimize and ramp up their testing operations with data. It helps improve the quality, stability, and performance of your test suite over time with rich insights. By highlighting persistent issues such as flakiness and always failing tests in your test suite, it helps to increase the quality of testing and, therefore, the quality of the end product. It also simplifies inefficient workflows that users perform many times a day, massively increasing efficiency and reducing frustration.

If your team is facing the following challenges, BrowserStack Test Observability can help!

Lack of visibility	You can't see what's going on with your tests, so you don't know which ones are failing, why they're failing, or how often they're failing.
Flaky tests	You have tests that fail sometimes but pass other times, and you can't figure out why.
Slow root cause analysis	It takes a long time to figure out why a test is failing, so you can't fix it quickly.

Flaky tests led to ~$312K wasted time per month

Source: GitLab Engineering Blog

57% of failing builds failed due to test job failures consisting of flaky and failing automated tests

Source: Slack Engineering Blog

Transform your testing operations with BrowserStack Test Observability, so your team can focus on writing more tests instead of debugging the ones that failed.

Debug & optimize your tests with surgical precision	We bring every kind of log into a single dashboard, helping you catch every bug. Our solution filters irrelevant tests, so you can focus on genuine test failures that need attention.
Optimize slow, monotonous workflows	We've simplified root cause analysis workflows and integrated with bug reporting tools and CI/CD platforms to help save time. Root cause analysis is now 10x faster than manual methods.
Do more with rich test suite insights	View rich, historical data on testing trends in your team. Leverage these insights to ship higher-quality products to market faster. Understand test suite health, identify issues, and optimize tests for continuous improvement.

"The most helpful feature of BrowserStack is the Test Observality tool. We rely on the data and results shared through this tool every morning during our test analysis."

Curtis M. Source: G2

Enhance every stage of your test automation journey with BrowserStack Test Observability

In our previous whitepaper on Turn Testing into a Business Advantage, we defined five stages of test maturity:

- **Manual**
- **Adopting Automation**
- **Ramping Automation**
- **Almost Automated**
- **Fully Automated**

BrowserStack Test Observability is a great solution for teams that are **New to Automation** - doing either manual testing or adopting automation - as they can ‘do it right the first time' and write flakiness-free test suites to avoid the problem altogether. They spend less time debugging failed test cases and can use the time saved to write new automation scenarios, thereby making further progress in quality improvement.

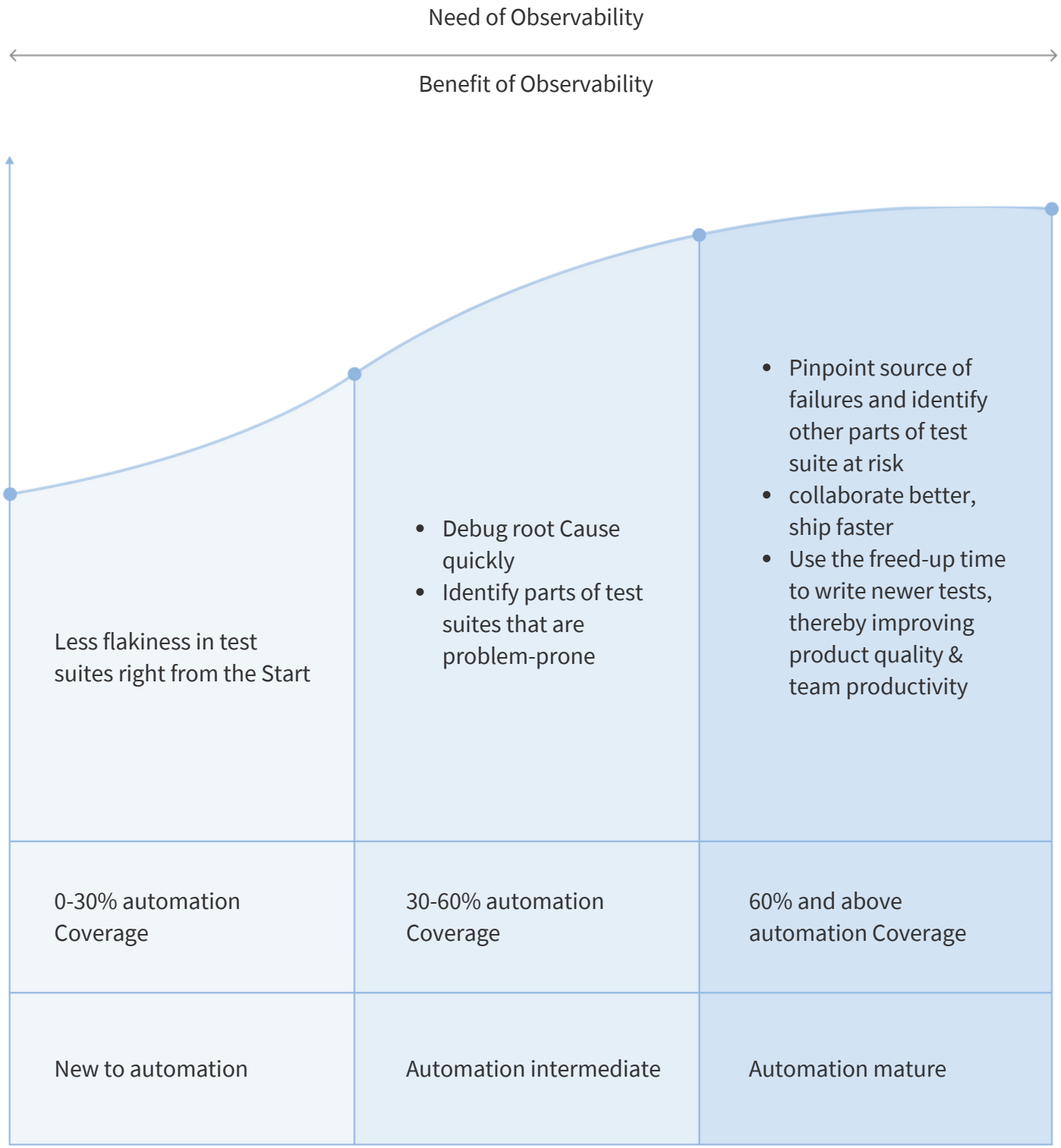
But for teams with mature test automation, it is an absolute game-changer and a must-have for improving the quality of QA like never before.

Automation Intermediate teams that are ramping automation have a significant number of tests that run on a predefined schedule. By having the capability to quickly debug the root cause of failures, teams can save a lot of time and use it instead to write new test scenarios and increase automation coverage. Teams also get visibility into parts of the test suite that are prone to problems and fix them quickly.

Automation Mature teams that are almost automated or fully automated, on the other hand, have a huge number of automation scenarios that are complex and intertwined. BrowserStack Test Observability can help teams quickly pinpoint the source of failure and identify other parts of the test suite that might be impacted. It also helps large teams collaborate effectively and resolve issues faster, improving time to release and product quality.

Regardless of their automation maturity stage, BrowserStack Test Observability helps teams significantly reduce engineering hours spent on run verification and expedite the resolution of critical issues and flaky tests. As a result, fewer bugs creep into production, ensuring a more robust and reliable software development process.

Made for every stage of your test automation journey



Type of QA team	Test Maturity Stage	Why use BrowserStack Test Observability
Teams New to Automation	• Manual testing coverage: ~90% • Test automation coverage: ~10%	• Automatically detect flaky tests while writing test automation suites • Create a robust foundation with reliable, stable tests and deterministic results • Spend less time debugging failed tests and more time writing new test scenarios to expedite test automation maturity
Automation Intermediate Teams	• Manual testing coverage: ~60% • Test automation coverage: ~40%	• Get all the relevant logs in a single view for better decision-making • Precisely pinpoint root causes to resolve issues quickly • Scale detection of flaky tests to eliminate flakiness from test suites • Ensure stable, reliable test suites to stop bugs from creeping into production • Boost productivity and reduce time to change • Free up teams from tedious manual work to focus on writing new scripts to cover more scenarios, thereby improving release quality
Automation Mature Teams	• Manual testing coverage: ~30% • Test automaton coverage: ~70%	• Save time and effort by eliminating manual troubleshooting - up to 10x faster • Cut regression time and get deterministic results for better planning • Debug tests faster to shorten Dev-QA and deployment cycles • Track relevant QA metrics such as growth of test cases, always failing & flaky tests, suite stability, and device/browser coverage to meausre the health of test suites over time • Take data-driven prioritization decisions for fixing tests to maintain trust in test results • Identify gaps in test coverage to improve QA effectiveness • Eliminate flaky tests to increase confidence in QA • Save time, effort and cost to drive team-wide efficiency • Improve time to market and product quality

Empower your teams with BrowserStack Test Observability

Improve Test Reliability

Write better tests and improve your debugging. Reduce defect leakage into production, and improve your UX.

Decrease Deployment Cycle Time and Save Engineering Costs

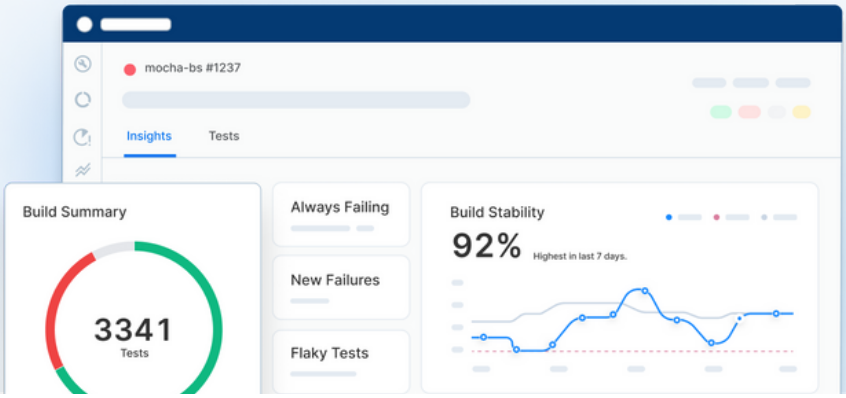
Save thousands of man-hours and ship to production faster with faster & more stable test suites.

Improve Developer Experience and Productivity

Build happier teams that avoid slow, repetitive, and frustrating debugging tasks.

See BrowserStack Test Observability in Action

Watch video



Try BrowserStack Test Observability Today!

Get Started

Contact Sales