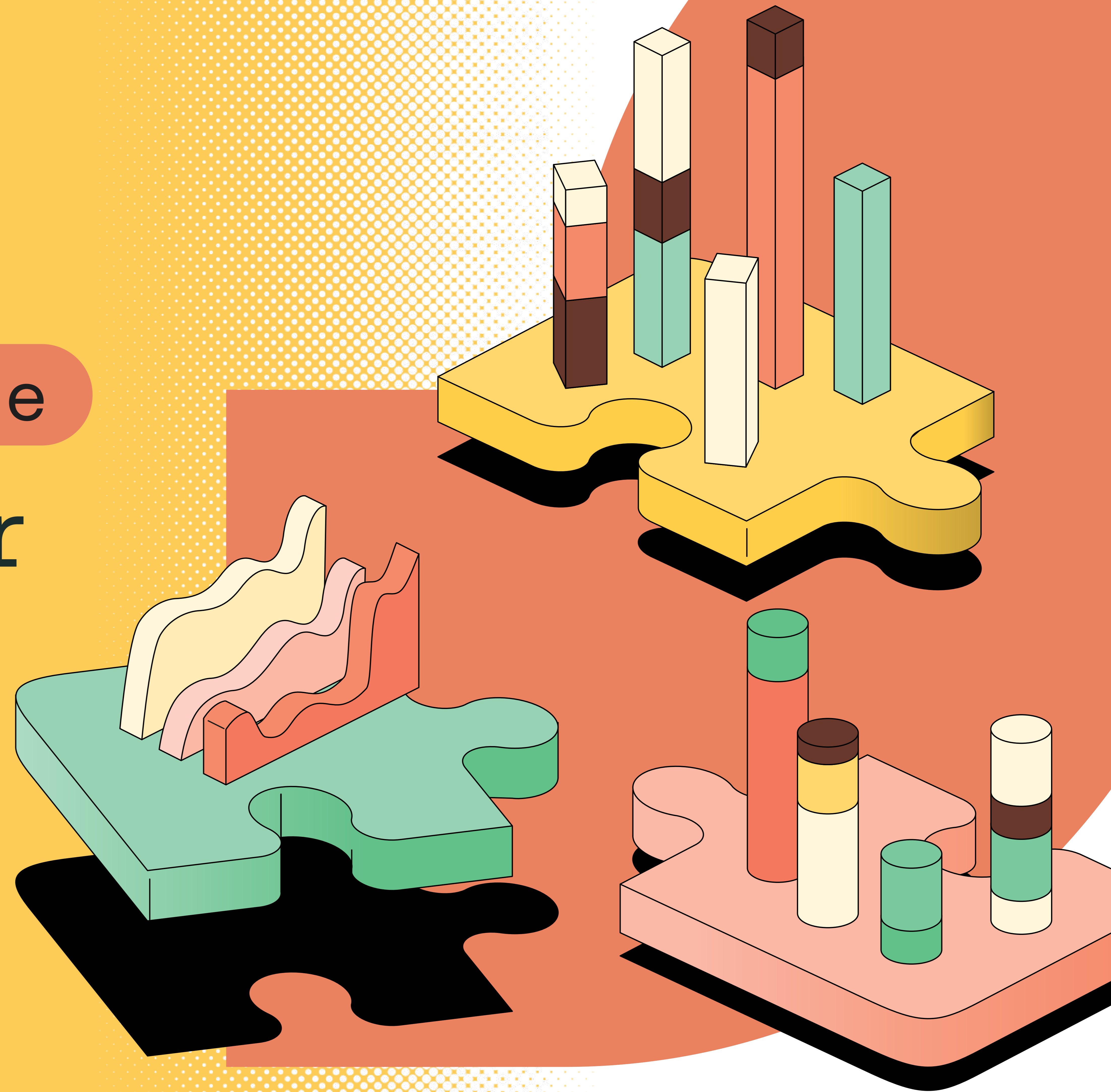


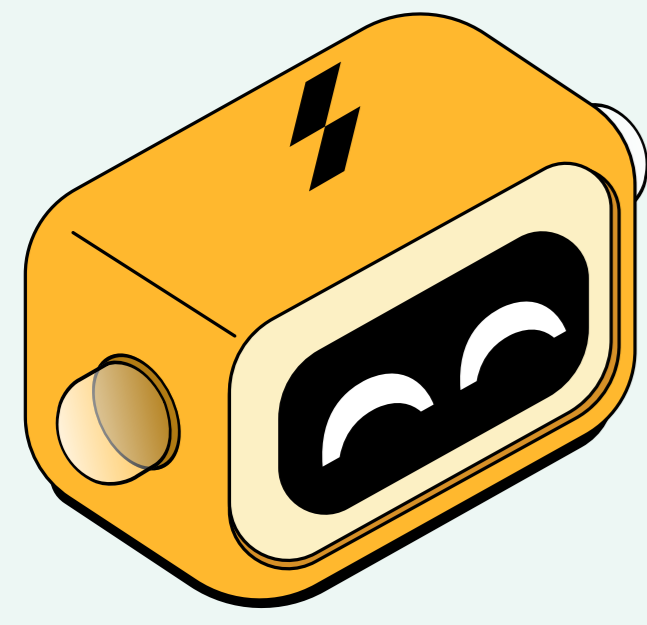


Insights Best Practices Guide

# 7 Ways to Test Smarter with Sauce Labs Insights



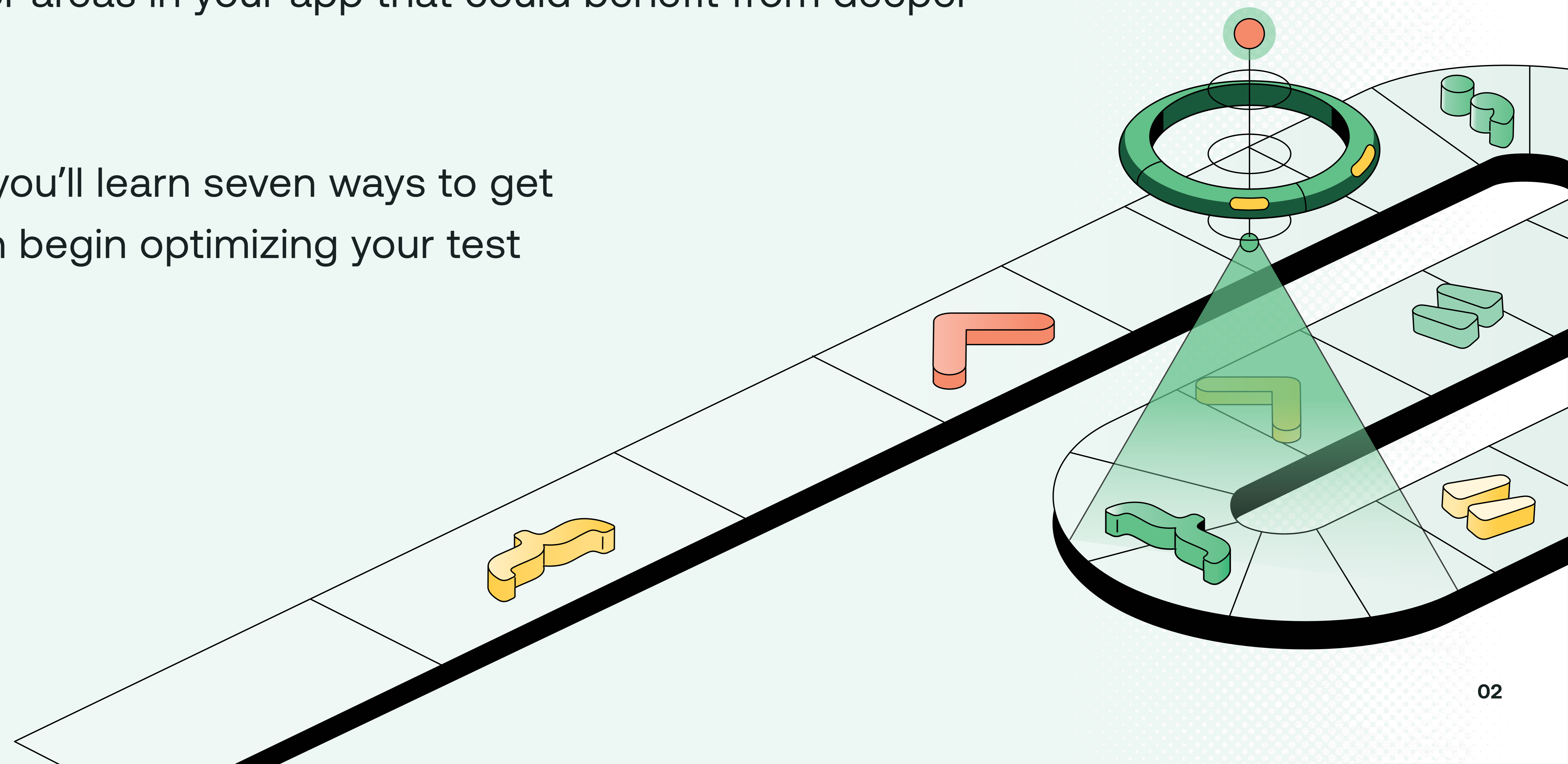




You've built out your test suite, are running tests, and are well on your way to shipping higher-quality software. But now that you are running more tests, you're generating far more data about your app quality and your test quality. What if you could use this data to improve test performance so you can test even smarter? That's where Sauce Labs' Insights comes in.

Insights is the reporting and analytics hub for your Sauce Labs test suite. Here you'll be able to interpret your test results over time, identify failure patterns across different platforms, and discover areas in your app that could benefit from deeper testing.

With this best practices guide, you'll learn seven ways to get started with Insights so you can begin optimizing your test processes.





# Populate your data

To get the most out of Insights, you must ensure the data your tests generate is accurate, relevant, and properly structured. Because Insights leverages data that is provided by each test, your test code should be set up to make it easily readable by Insights. Here's how.

### Set unique test names

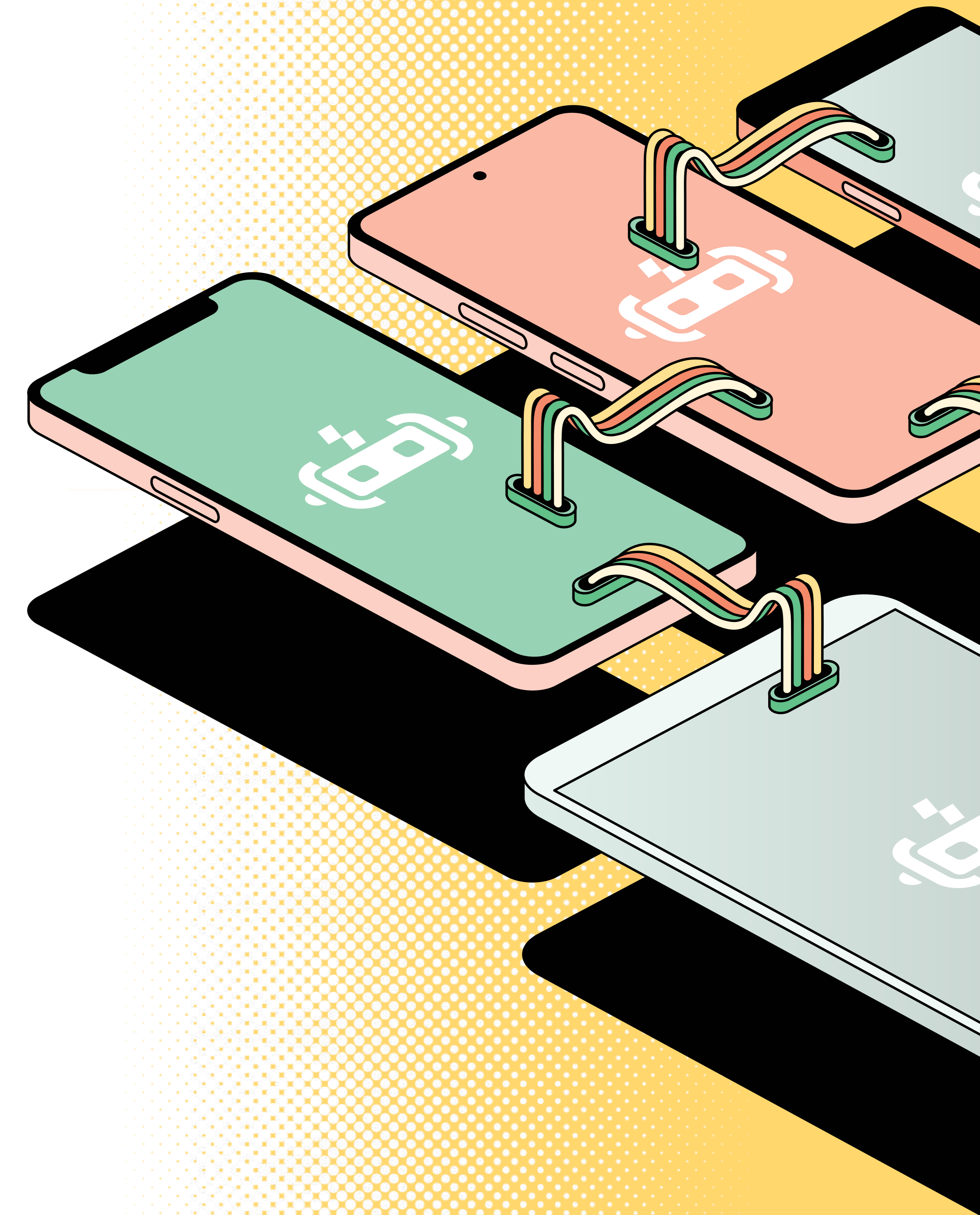
Unique test names make it easier to track the performance of specific tests so you can optimize them over time. Because Insights groups tests by name, make sure the name you provide for each test is unique. Use the test method name, and verify that all method names are unique across all test builds. When setting up the test, add the following code with your unique test name:

```
sauceOptions.put("name", testinfo.getDisplayName());
```

### Leverage tags to manage tests

Tags can help you organize your tests in a logical manner so you can analyze larger subsets of tests like specific builds or features. Here's how to add a tag to your test:

```
sauceOptions.put("tags", context.getTags());
```

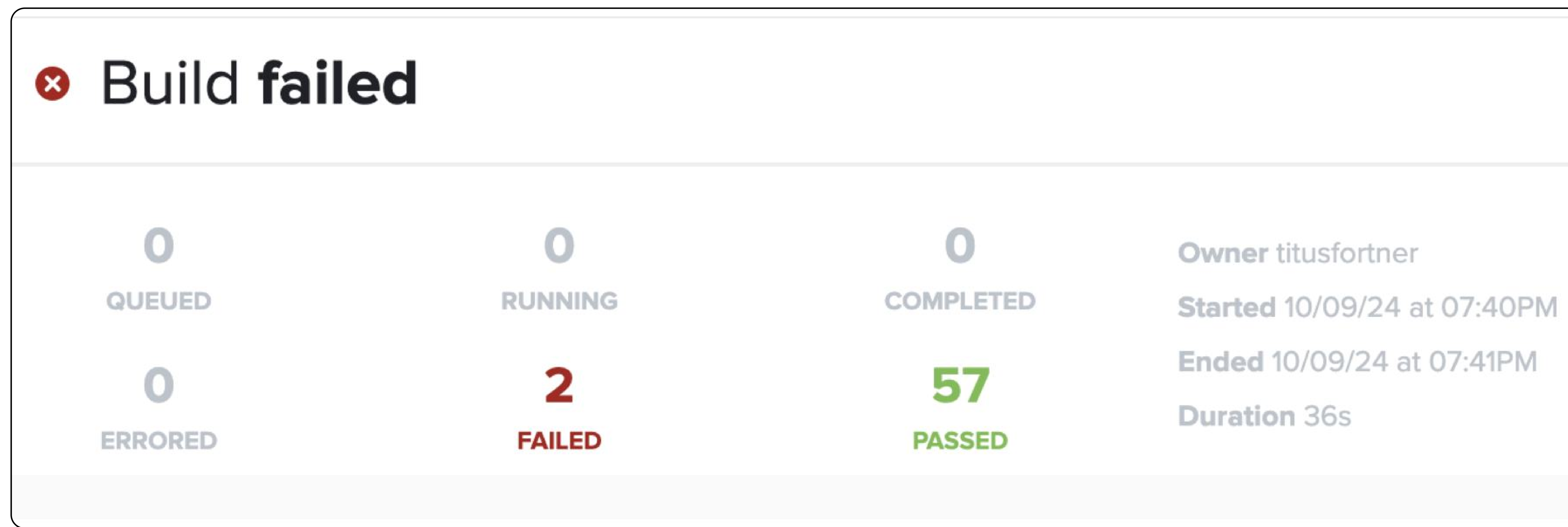




## 7 Ways to Test Smarter with Sauce Labs Insights

### Group tests with a build name

Test jobs, or builds, have a name and number that gets incremented after each execution. Grouping your tests by build name helps you track the performance of your testing for a specific build over multiple test runs so you can easily compare test performance results. Here's what this looks like in the Results View.



The best way to assign a build name is to use the environment variables in your CI tool. For example, here's what it looks like in Jenkins:

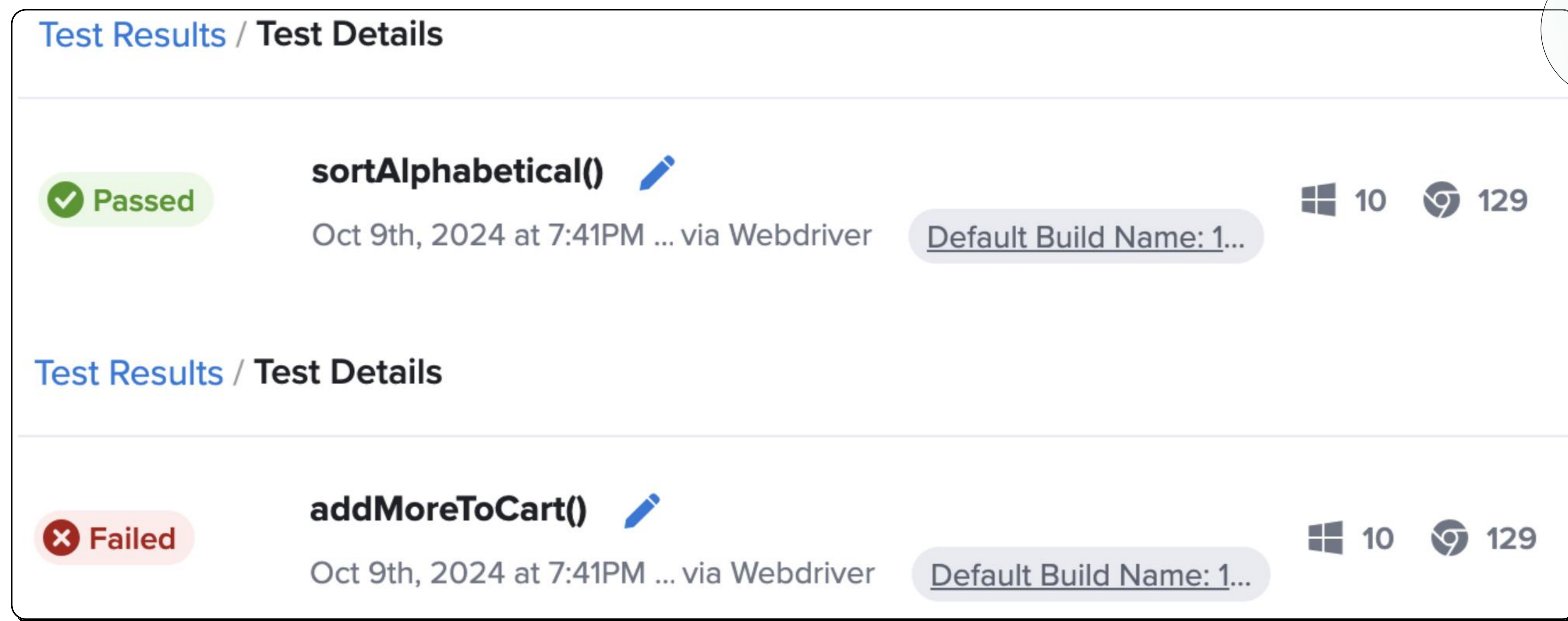
```
sauceOptions.put("build", System.getenv("BUILD_NAME") +  
": " + System.getenv("BUILD_NUMBER"));
```



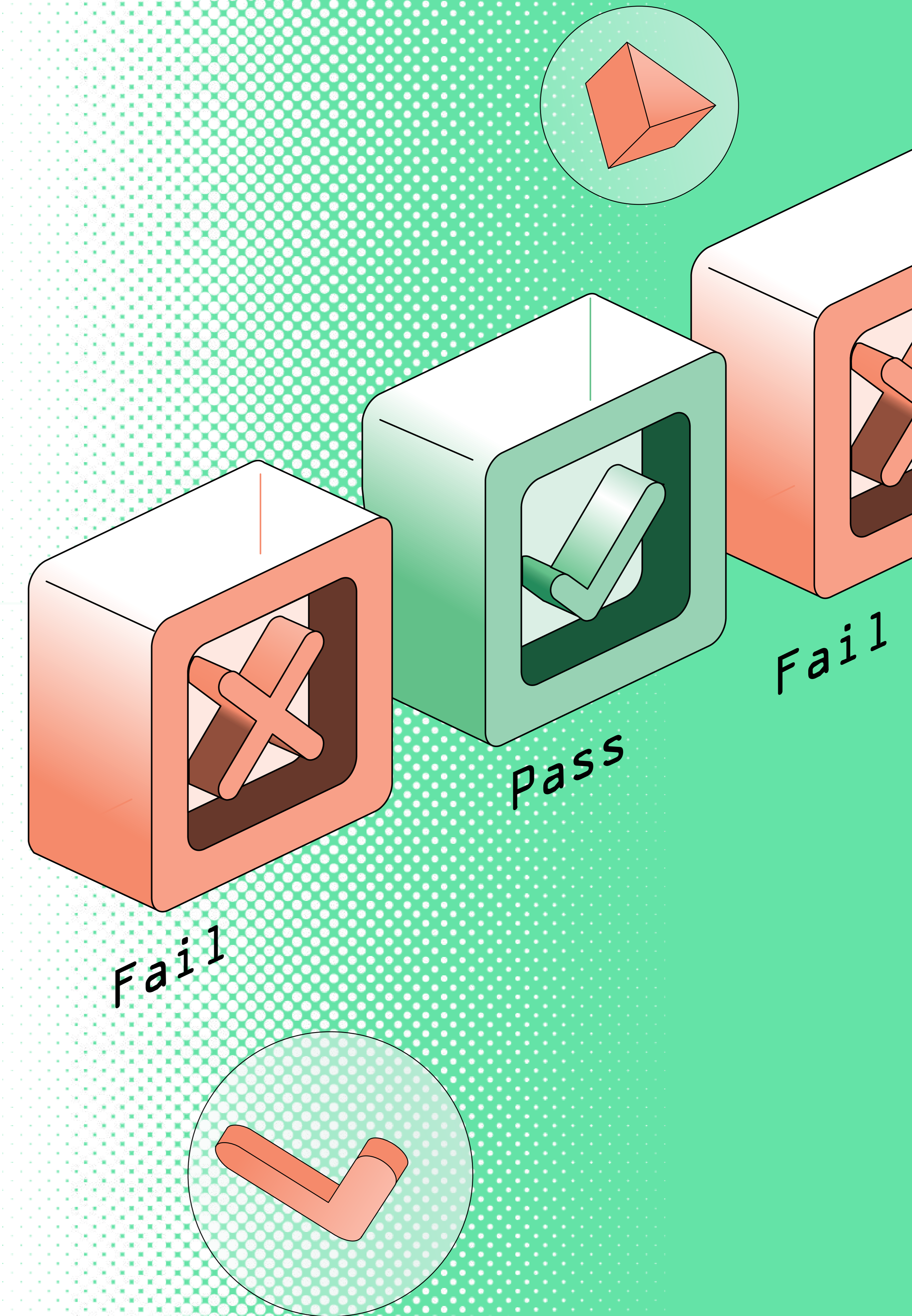


# Set results to improve reporting accuracy

While Sauce Labs Platform for Test can help you test at scale, it doesn't necessarily know the intention of the test. That's where you come in. Setting the Pass/Fail status of your tests is the key to giving Insights the context it needs to fuel its results. By telling the system whether the test was a pass or a fail at the end of the session, you can then use Insights to analyze why tests are failing so you can improve your process.



The screenshot displays two test result entries. The first entry, 'sortAlphabetical()', is marked as 'Passed' with a green checkmark icon. The second entry, 'addMoreToCart()', is marked as 'Failed' with a red 'X' icon. Both entries include the test name, a timestamp ('Oct 9th, 2024 at 7:41PM ... via Webdriver'), and a build name ('Default Build Name: 1...'). The browser and version information 'Windows 10' and '129' are also visible for both tests.





## 7 Ways to Test Smarter with Sauce Labs Insights

There are two ways to do this. One way is to simply add a line of code in the JavaScript Executor:

```
((RemoteWebDriver)driver).executeScript("sauce:job-result=passed");
```

While this method works most of the time, a problem with the test that causes the driver to become unstable could force the test to quit unexpectedly before it is finished, which means Insights won't be able to leverage the results. To ensure the result gets sent every time regardless of driver performance, we recommend using a REST client. Here's what that looks like using Java's SauceREST library:

```
SauceREST rest = new SauceREST(DataCenter.US_WEST);  
String sessionId = ((RemoteWebDriver)  
driver).getSessionId().toString();  
rest.getJobsEndpoint().changeResults(String.valueOf(sessionId), true);
```





# Explore!

Now that you have Insights set up, it's time to start exploring what you can do. Go to the Insights tab on your Sauce Labs platform to see all your options for analyzing test results.

### Resolve pervasive issues with Failure Analysis

[Learn more](#) →

**What is it:** Failure Analysis helps you optimize test efficiency and efficacy by identifying common failure patterns within your test suite so you can solve problems at their root.

**Who is it for:** SDET individual contributors who want to uncover patterns impacting the overall test suite.

**What you can do:** Use Failure Analysis to improve developer efficiency by streamlining the detection and triage of your most pervasive errors. It can also help validate an investment in test automation by showing larger patterns as a source of failure, allowing for global mitigation and faster time-to-market with better quality.

### Examine your test suite health with Job Overview

[Learn more](#) →

**What is it:** Job Overview gives you a comprehensive overview of your test suite, along with the ability to drill down into builds, operating systems, browsers, and frameworks.

**Who is it for:** QA leaders seeking insights into the current health of their tests.

**What you can do:** Leverage enhanced reporting to uncover consistently passing, failing, or erroring jobs so you can better communicate quality metrics across cross-functional teams, improving their efficiency and collaboration.





## Adapt to issues quickly with Job History

[Learn more](#) →

**What is it:** Job History provides a visual overview of test performance over time, helping you identify patterns and issues with test stability across various platforms and browsers.

**Who is it for:** QA leaders who want to dive deeper into test results and performance metrics like errors, failures, and runtime.

**What you can do:** View time-specific test results, identify test anomalies or patterns, and make more targeted improvements by identifying long-term issues related to test performance and flakiness.

## Improve test performance with Trends

[Learn more](#) →

**What is it:** Trends provides a quick overview of what's going on with your tests as a whole. Applying filters to the visualizations enables you to dig into the data to generate answers to specific questions about test performance.

**Who is it for:** QA leaders who want to pinpoint exactly where to focus efforts, identify improvement opportunities, and run a more efficient test suite.

**What you can do:** Use interactive visualizations to uncover persistent errors so you can optimize test suite performance, efficiency, and use of resources.

## Test what matters to your users with Coverage

[Learn more](#) →

**What is it:** The Coverage report provides a breakdown of the browsers and mobile devices your organization tests against so you can make more informed decisions about where to focus testing resources.

**Who is it for:** QA leaders who want to ensure their testing strategy aligns with the browsers and devices most used by current or prospective customers.

**What you can do:** Gather user data through Google Analytics to identify the browsers, devices, and platforms your customers use. In Insights, you can then specify the type of coverage you wish to view by selecting the Devices, Browsers, or OS tab to ensure your testing is tailored to your audience.

## Plan and execute your test strategy with Usage

[Learn more](#) →

**What is it:** Usage provides an accurate view of your concurrency data and lets you compare it to your subscription limit so you can maximize your investment.

**Who is it for:** QA leaders who want to understand testing patterns at an organization level or team level.

**What you can do:** Compare concurrency usage between teams across time to see who is contributing most. You can also visualize your maximum concurrency usage as it approaches the subscription limit so you can make informed decisions about your usage.



# 04 Make the most out of each tool

Insights is more than just a reporting tool. It's a powerful resource you can use to react to optimization opportunities while proactively planning for future strategic needs.

### Adaptive

Use Job Overview, Job History, and Trends to triage test failures right now and over time to better understand how to improve results. Each tab allows you to filter your tests by platform, browser version, or device so you can quickly identify the exact tests to troubleshoot. Use tags to monitor custom test sets so you can analyze specific aspects of your test suite, such as a particular feature.

### Proactive

Use Coverage and Usage to strategically plan how you test and what you should test for. In addition to maximizing resources by testing the device/OS/browsers most important to your users and business, it can help you maximize your Sauce Labs investment by seeing if your team is utilizing its available sessions efficiently.

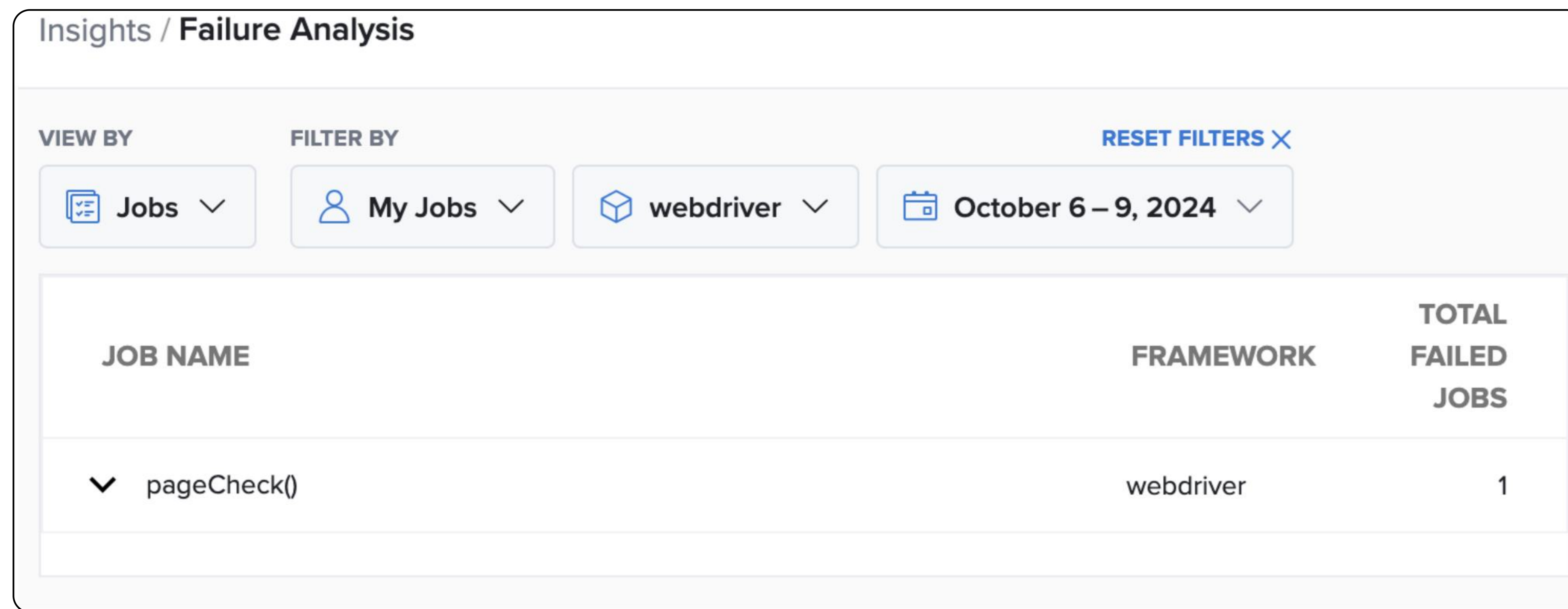




# Troubleshooting failures 101

If you discover that your recent tests have an unusually high number of failures, it can often indicate a common cause. Failure Analysis can help you speed up debugging by fixing the root issue so you can get code into production more efficiently.

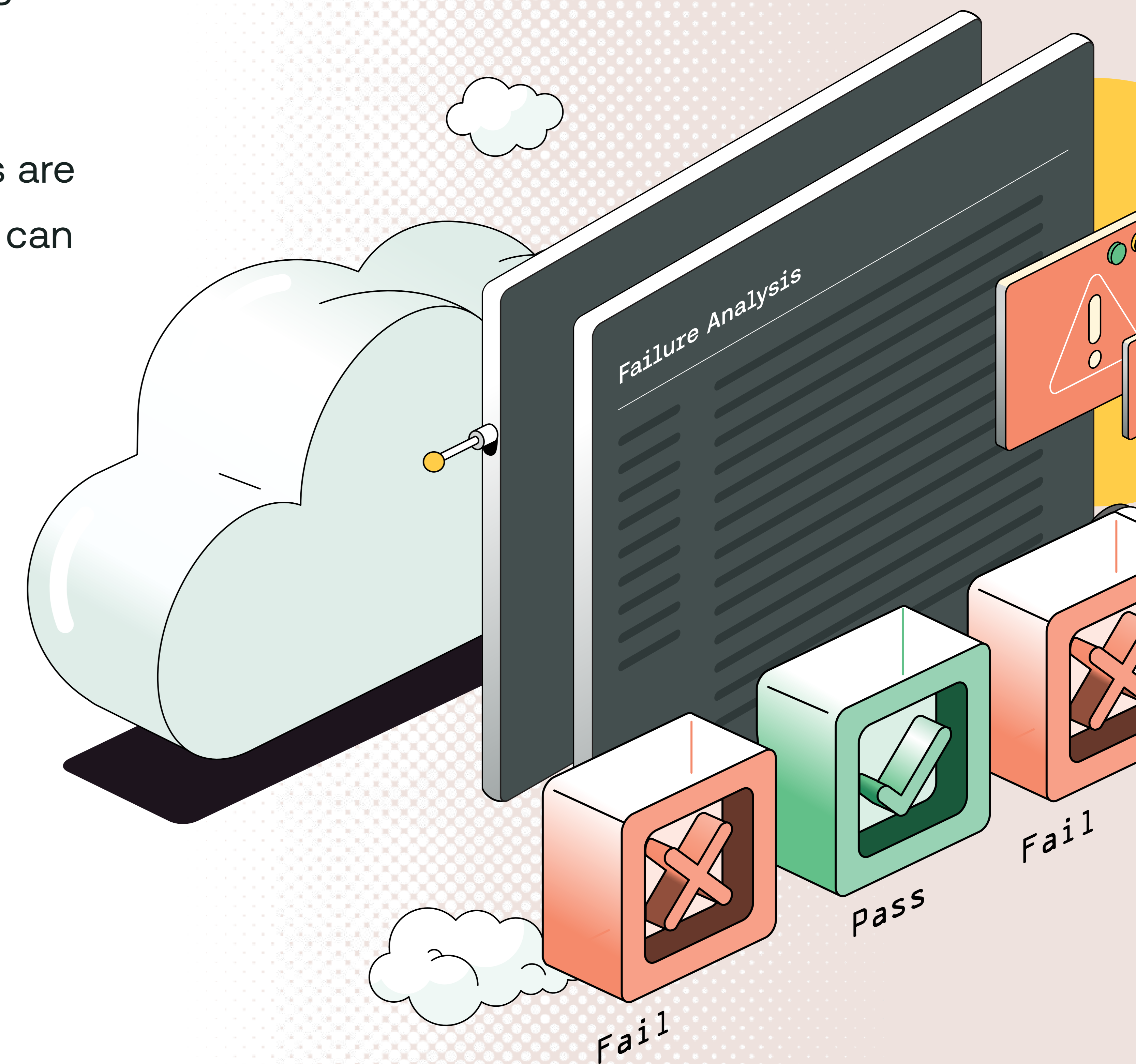
Failure Analysis works by grouping similar test failures together, helping you quickly see if multiple tests are failing for the same reason, such as a broken login page. Rather than go through tests one by one, you can identify and fix all those failures at once.



The screenshot shows the 'Insights / Failure Analysis' page. It features a 'VIEW BY' section with 'Jobs' selected, and a 'FILTER BY' section with 'My Jobs', 'webdriver', and 'October 6 - 9, 2024' selected. A 'RESET FILTERS' link is also present. Below the filters is a table with the following data:

JOB NAME	FRAMEWORK	TOTAL FAILED JOBS
pageCheck()	webdriver	1

Another troubleshooting tip is to use the Job Overview page to see how tests have performed over the last week or month. This gives you a high-level view of the tests you're running and their results so you can identify spikes tied to recent work, which are easier to fix now rather than later in the future.






## 7 Ways to Test Smarter with Sauce Labs Insights

To aid in triage, the Job Overview tab includes a Snapshot section that differentiates jobs that always fail from jobs that fail intermittently. Click on any link in the Snapshot section to go to the Job History tab, where you can easily dig into both categories so you can divide and conquer test failures based on root cause and the difficulty of implementing a solution. Tip: Filter results by what you want or don't want to include to see specific issues, such as only bugs impacting Safari users.

**SNAPSHOT - latest 55 jobs**



- 1 Consistently Failing vs. 0 (--%)
- 52 Consistently Passing vs. 0 (--%)
- 0 Consistently Error vs. 0 (0%)
- 0 Missing Status vs. 0 (0%)
- 2 Inconsistent Results vs. 0 (--%)

View by: **Inconsistent Result** ^ Items per page: 25 v [Export CSV](#)

Job N	All	Total Runs	Average Duration	Total Duration	Pass Count	Pass Rate	Fail Count	Fail Rate	Error Count	Error Rate	Complete Count
viewC	Consistently Failing	26	22.3s	9m 39s	20	76.9%	3	11.5%	3	11.5%	0
viewI	Consistently Passing	26	21.8s	9m 27s	21	80.8%	2	7.7%	3	11.5%	0
useC	Consistently Error	91	11.4s	17m 19s	90	98.9%	0	0.0%	1	1.1%	0
unsuc	Missing Status	28	31.4s	14m 39s	23	82.1%	0	0.0%	5	17.9%	0
succe	<b>Inconsistent Result</b>	28	23.6s	11m 0s	24	85.7%	0	0.0%	4	14.3%	0



# Go deeper with Extended Debugging

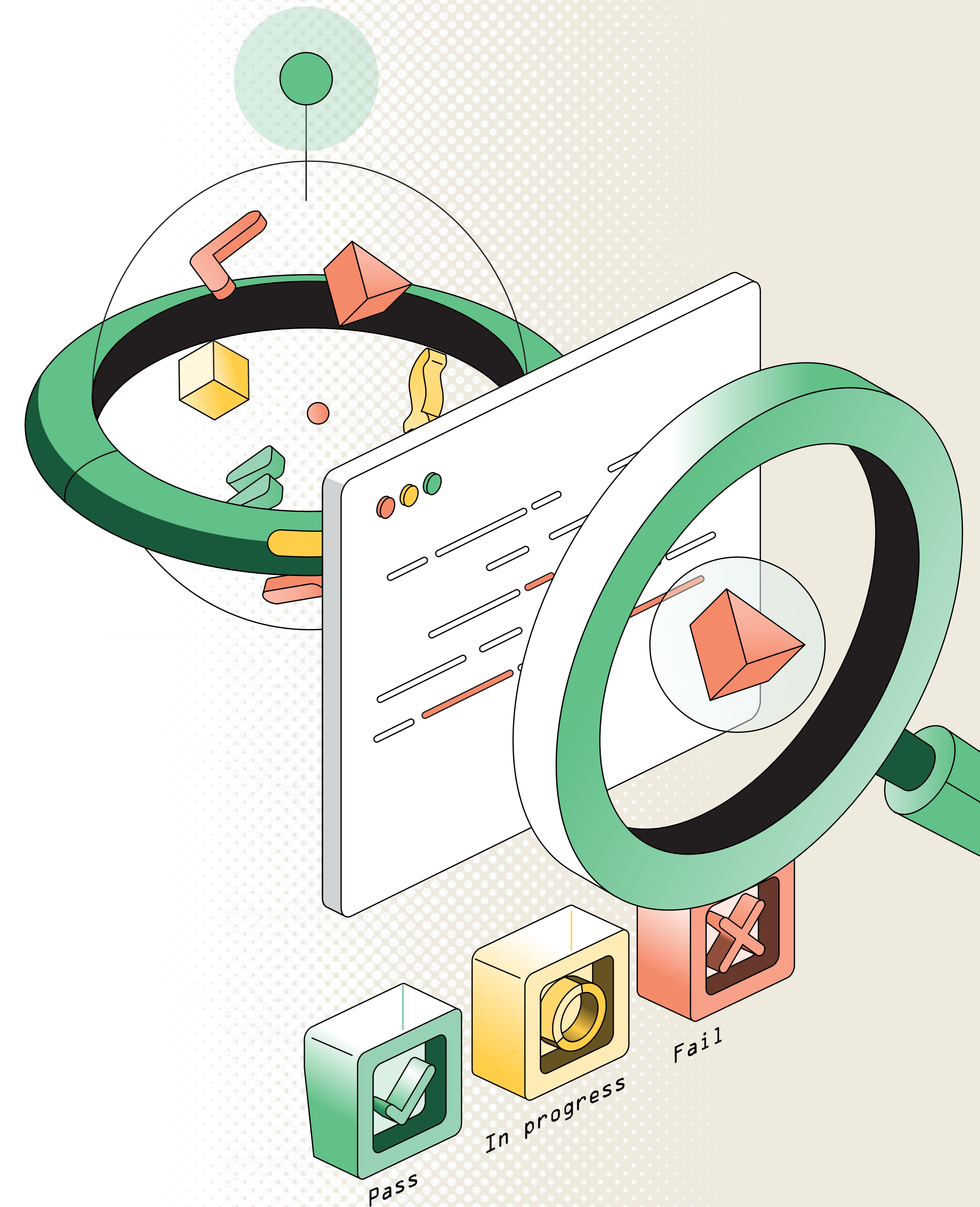
Extended Debugging does what it says: this feature offers additional capabilities that you can use to gain deeper insight when diagnosing flaky tests or performance degradation. Use Extended Debugging to collect the following additional data:

- **JavaScript (JS) Console Logs:** The JS console collects security errors, warnings, and messages that are explicitly logged by the browser. Use these logs to find out which components of your web app failed to load or ran into an error, what warnings were logged by the browser, and to get more information about app performance.
- **HTTP Archive Format Files:** HAR files collect all network requests and responses made and received by the browser during testing. Use HAR files to identify browser requests that time out, pinpoint requests slowing down the loading process, and locate faulty API calls.

Because Extended Debugging collects JS console logs and HAR files generated during testing, it can impact your overall test performance, so keep in mind that this isn't a tool you'll want to use for everyday testing.

To generate the JS console logs and HAR files, add the `extendedDebugging` capability to your Selenium test script and set it to true. Once a test with Extended Debugging enabled is complete, you can access the logs and files through Sauce Labs or with the REST API.

[Learn more about Extended Debugging](#) →





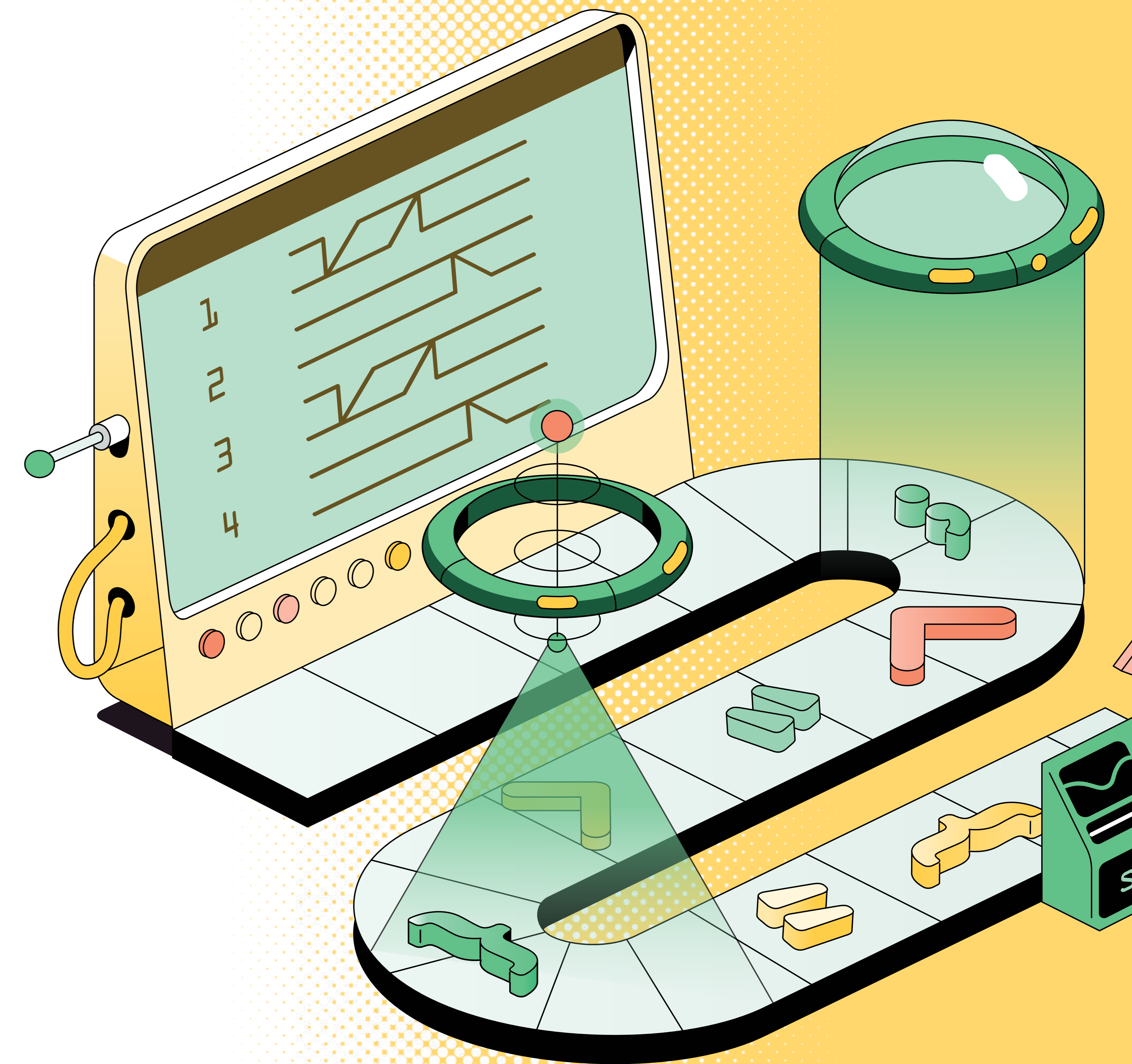
# Automate Insights with Sauce Bindings

Sauce Bindings are a growing collection of several Java libraries that automatically populate everything required to get the most out of Insights, reducing your need for maintenance, session cleanup, and status reporting. Use one of the test runner plugins (JUnit 4, JUnit 5, and TestNG) to automatically:

- Structure your Sauce tests
- Store the method name as the test name
- Populate the build name based on CI tool environment variables
- Set existing tags from your tests
- Set pass and fail information at the end of the test run

Ready to start making the most out of your test data?

[Get started with Insights today](#)





# About Sauce Labs

Sauce Labs is the leading cloud-hosted platform for automated testing of web and mobile applications, enabling fast delivery of high-quality software across the development lifecycle. Founded by the creators of Selenium, Sauce Labs has been the testing leader for over 15 years and now runs over 1 billion tests annually. Trusted by Fortune 500 companies like Toyota, Walmart, Verizon, Gannett, and Fidelity Investments, its scalable, secure platform supports testing across thousands of operating systems, browsers, and devices while meeting the highest compliance standards.

[hi@sauce labs.com](mailto:hi@sauce labs.com)

[www.sauce labs.com](http://www.sauce labs.com)