

Ship Features 10X Faster with Shift-Left Testing

Enhanced Quality and Speed

TABLE OF CONTENTS

Executive Summary	3
Problems With Traditional Testing	4
Shift-Left Testing: The Sooner The Better	5
Shift-left Testing Vs E2E Testing: Which is Better?	7
How Shift-left Testing Transforms Your SDLC?	9
Ways To Implement Shift-left Testing	11
Redefine Your Release Strategy With HyperTest	12
Conclusion	15

Executive Summary

The best engineering teams have smooth DevOps, the use of microservices, and a team of good, responsive developers to stay competitive, but they are still facing a great deal of risk. Teams are making use of DevOps to adapt to the demanding and volatile environment of the market, but **what still comes up as a challenge is testing.**

To mitigate the risks of failure and have a robust testing phase in place, **shift-left testing is encouraged.**

“ A defect identified during the production phase will cost around 100x more than the one that is identified and fixed during the development phase. ”

Shift-left testing is not a new concept, but it has gained popularity in recent years due to the increasing reliance on agile development methodologies and the need to deliver high-quality products quickly. In this white paper, we have covered everything about shift-left testing that **will boost your release strategy.**

Problems with Traditional Testing

The traditional way of testing places application testing at the end of the development cycle. However, this approach can be suboptimal because if applications do not meet quality standards or fail to function correctly, they are sent back to the development teams for fixing.

- **Slower development:** Traditional models place testing closer to the deployment phase. This creates a bottleneck in the release process because it accumulates too many changes to be tested together, which slows down testing and in turn releases.
- **Difficult to catch and fix issues later:** Bugs found later in the development process tend to be harder and more tedious to fix than bugs found earlier.
- **More expensive:** Identifying and resolving issues during the production phase definitely hits hard on budget. Also, the need to retest after fixing issues can also add to the overall cost of testing.
- **Delayed Release Cycle:** Since testing happens at the later stages, and if any bugs or issues are found at that time, the whole development schedule gets delayed.

Shift Left Testing: The Sooner The Better

Have you ever worked on or lead a software project that was over budget or under time constraints? Most likely, you did. Contrary to popular belief, poor planning isn't always to blame when a project runs over its deadline. The project's code validation process is where the true issue is.

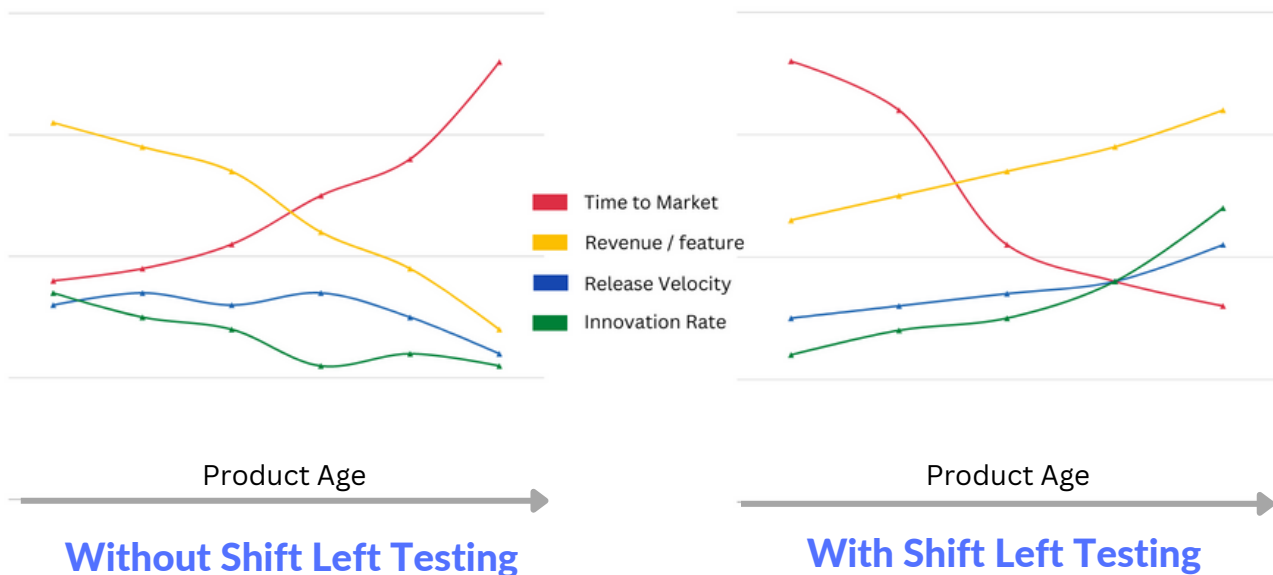
In other words, software testing is the key to it all. Or, more particularly, to software testing that is carried out insufficiently often and at a late stage in the project. Shift left testing is a suggested remedy for this issue.

Shift Left testing is an approach that involves moving the tasks related to testing earlier in the development process. This means that testing tasks that are traditionally done at a later stage should instead be performed earlier – particularly during development i.e. right around when new code changes are about to be merged

Approach of Shift-left Testing

- Non-UI smaller, more atomic tests that **tests output of the system under test (SUT), with defined inputs**
- Shift-left testing approach provides very quick and extremely precise feedback for developers on breaking changes, that devs can debug quickly, fix and release patches
- The kind of tests that usually cover shift-left approach are unit or integration tests but never E2E tests
- These tests should cover functional as well as logical issues

Shift-left testing vs E2E testing: Which is better?



There has been a constant debate on nailing down on the best way of testing. Here are a few quick comparisons:

- **'Shift-left testing'** by virtue of testing smaller commits tests user contracts, testing the output of the application (or service) with available inputs, hence **runs faster and gives quick feedback to developers**
- In contrast, **E2E testing** follows the 'shift-right' testing approach which aims to test the entire user story closer to the deployment process, that might involve testing several components of the application together, **making it slower and possibly catching issues very late in the SDLC cycle**

- Waiting to test during production means the **team is always playing catch up and fighting, inherently greater risk**
- So with the **'shift-left' approach teams can catch issues faster, and much early in the development** and design cycle making it much easier to patch and release vs E2E approach that catches issues, if any, much late, taking it longer for devs to fix, slowing down releases and the entire SDLC cycle

How Shift-left Testing can transform SDLC?

For development teams, moving testing earlier in the process has a wealth of advantages. Two unique outcomes—**faster innovation and reduced time to market** for new features can be used to describe these advantages. Here are a few more:

- **Automation** - Testing can be automated more effectively by shifting to the left. Some important advantages of test automation are:
 - Considerably fewer human errors
 - More thorough test coverage (conducting multiple tests concurrently)
 - Capacity for testers to concentrate on more important activities
 - Less problems in production

- **Faster Innovation** - Early API testing also allows you to increase code sanity without slowing down development. Continuous testing can lower the expenses associated with duplicate testing while increasing your organisation's confidence in APIs.

- **Delivery Velocity** - In this case, faster is also earlier. Defects are much easier to rectify when discovered early in the production cycle. The result - The interval between releases may shorten dramatically and the software gets better in quality.
- **Lower Costs & Higher revenue** - Early and frequent API testing greatly lowers remediation costs since flaws can be fixed before they pose a risk to the company in production. By assuring that new releases are bug free and unlikely to need rework in the future, automated testing enables developers to move fast to fulfil the needs of customers.
- **Increased Satisfaction** - One of the main advantages of the shift-left strategy is faster delivery of software with fewer flaws. Products can keep their competitive edge or increase a competitive lead in the market because they can meet client expectations and hence deliver outstanding customer experiences.

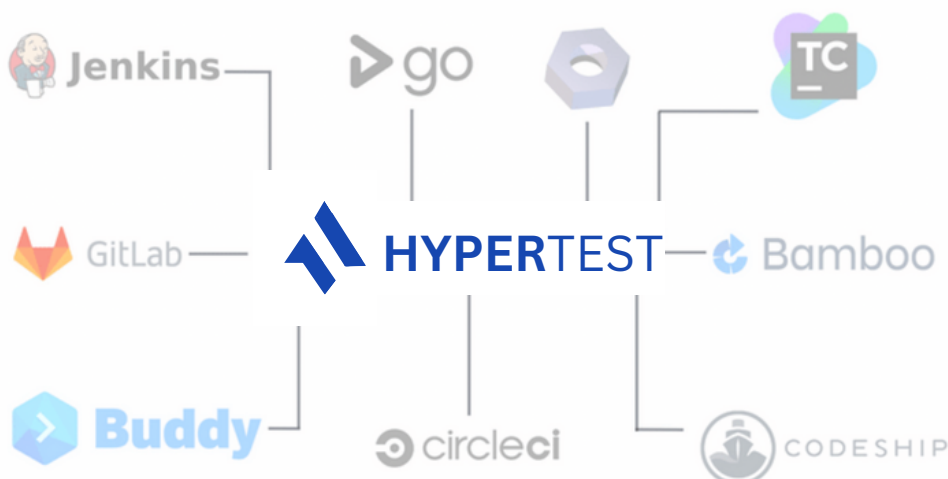
Ways To Implement Shift-left Testing

If you have a **micro-services architecture**, a shift-left testing approach becomes by default the best testing approach and something you can easily put in place.

Consider the benefits:

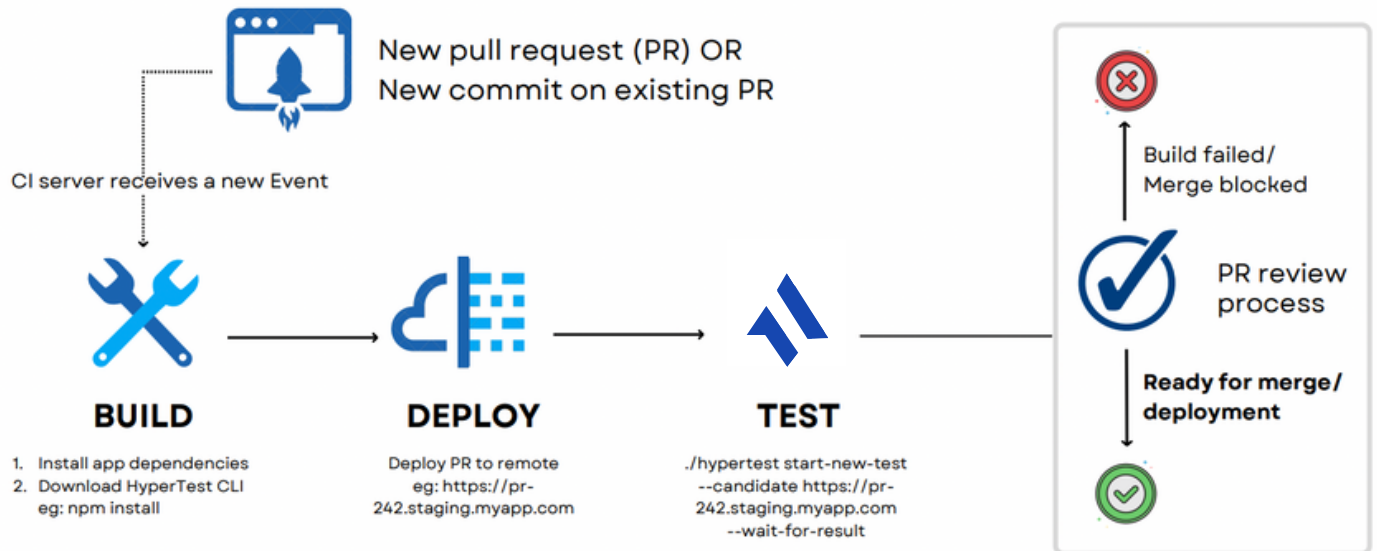
- In a micro-services architecture , services are loosely coupled that give devs the freedom to make and deploy changes to each of these services independently.
- A shift-left approach tests these commits one at a time, independent of the dependent services or alongside them, but providing quick bit-sized feedback on what can be fixed immediately.

HyperTest using its CLI can integrate natively with any CI tool used for automated releases, and tests every new change or update in the application automatically with a new PR or commit as the trigger



HyperTest CLI utility integrates with any CI tool

Redefine Your GitHub



CI pipeline with HyperTest

When a PR is raised by the dev using GitHub, GitLab, Bitbucket or any other version control system,

2 things happen -

1. Their CI server would receive a **new event notification** which would then let it build and deploy the app.
2. The **native CLI utility of HyperTest** is in parallel notified of these events, making it automatically run all the tests.

The best part is that the final report that HyperTest generates can be viewed by devs inside their VCS, without ever moving out.

hypertestco / sample-ci-checksPublic

Code

Issues

Pull requests2

Actions

Projects

Wiki

Security

Insights

Settings

Test branch #2

Openkaranssj4 wants to merge 20 commits into main from test_branch

Conversation0

Commits20

Checks4

Files changed2

tc 139fc6c01

hypertest-app

XHypertest checkRe-run

>CI

on: pull_request

hypertest-app / Hypertest check

failed 9 days ago in 17s

Test #113 is completed.
Results accepted by karan@example.com at Tue Oct 18 2022 05:14:50 GMT+0000 (Coordinated Universal Time)
Comments: qwertyuio

DETAILS

Commit 9fc6c01 - Test Report

Manual Interventions:
[Force Skip this check](#) [Force Fail this check](#)

Summary

Regressions	Affected APIs	Covered APIs
8	12	20

Details

API	Incident
GET /api/v0/Merchants/{id}/transactions GET /api/v0/Merchants/{id} POST /api/v0/Users/registerUserViaOtp	STATUS CODE CHANGED to 500
POST /api/v0/Merchants/{id}/getDailyCollection POST /api/v0/Otps POST /api/v0/Users/{id}/addNewMerchant	KEY REMOVED at path /ts4_primary
POST /api/v0/Merchants/{id}/getDailyCollection POST /api/v0/Otps POST /api/v0/Users/{id}/addNewMerchant	VALUE MODIFIED at path /type
GET /getUser/{id}/value_modified_example	DATA TYPE CHANGED at path /credit VALUE MODIFIED at path /rating
GET /Top5RatedUsers/array_order_example GET /Top5RatedUsers/value_modified_example	DATA TYPE CHANGED at path /
GET /getUser/{id}/content_type_example	CONTENT TYPE CHANGED from application/json; charset=utf-8 to application/xml; charset=utf-8
GET /getUser/{id}/key_removed_example	KEY REMOVED at path /rating
GET /getUser/{id}/status_code_example	STATUS CODE CHANGED to 404

Top 10 Slowest APIs

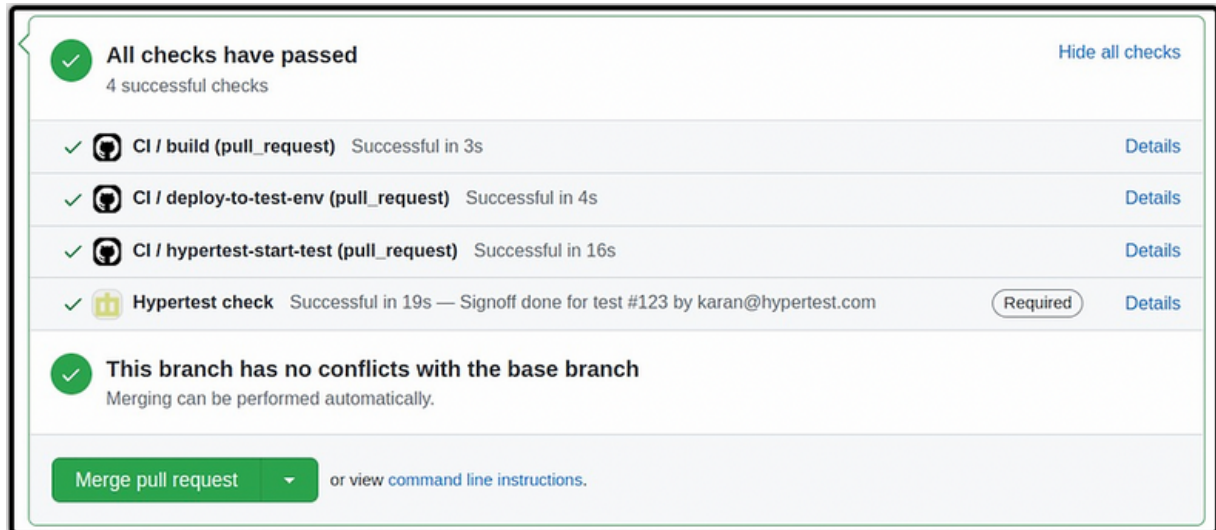
API	Response Time
POST /api/v0/Users/registerUserViaOtp	0.10s (candidate)
POST /api/v0/Transactions/recordNewTransaction	0.02s (candidate)
POST /api/v0/Users/{id}/addNewMerchant	0.01s (candidate)
GET /getUser/{id}/key_added_example	0.01s (candidate)
GET /api/v0/Merchants/{id}/transactions	0.01s (candidate)
POST /api/v0/Merchants/{id}/getDailyCollection	0.01s (candidate)
GET /dashboard/	0.01s (candidate)
GET /getUser/{id}/header_added_example	0.01s (candidate)
POST /api/v0/Otps	0.01s (candidate)
GET /api/v0/Merchants/{id}/settlements	0.01s (candidate)

Test Instance Details

Date = Tue Oct 18 2022 05:14:51 GMT+0000 (Coordinated Universal Time)

View more details on hypertest-app

Tests every commit inside GitHub



- This helps devs review any breaking changes with the current build in minutes.
- Devs would only be able to merge clean builds to prod pending a manual sign-off

Conclusion

Believe in the power of "shift-left" testing and let that experience improve your release management cycle. We use the shift-left approach at HyperTest, which lets us help teams build software faster and find and fix bugs before it goes live.

“

We have recently upgraded our code framework. **And by running one instance of HyperTest, we got the first-cut errors in less than an hour**, which could have taken us a few days.

”

Vibhor G.
VP - Engineering



Cut down on the heavy cost of fixing errors in production and adopt shift-left testing with HyperTest today to catch all bugs before production and improve the quality of your software.

Request a Demo >