OverOps

OverOps for QA

Stop Sev1s Before They Hit Production

Prevent poor quality code from reaching your customers. Integrate with Jenkins to ensure faulty releases never get shipped, and route issues back to the right developer with complete context every time.

	reased Severit 20% Critica
ase Candidate Volume / Rate Inc	
Volume / Rate Inc	
Broker	20% Critica
DIOREI	20% Chuca
vice_Manager	5% Mediur
	10% Low
	5% Low
1	

Have Confidence in the Reliability of Every Release

Today's QA teams are faced with fast release cycles, limited time to test and verify releases and pressure to limit the errors in production.

The OverOps Platform helps solve these challenges by delivering actionable code-level insights and context into the quality of software across the automated deployment pipeline. With OverOps, QA teams see all new anomalies introduced by a release, and automatically assign them a severity based on potential impact to the code. Through our Jenkins plugin, users can automatically block poor quality releases from production.



Identify all critical errors and slowdowns even those not detected by APMs and logs.

- Certify every release as safe to promote based on individual reliability scores.
- Integrate with your CI/CD workflow to \bigcirc automatically block unstable releases.



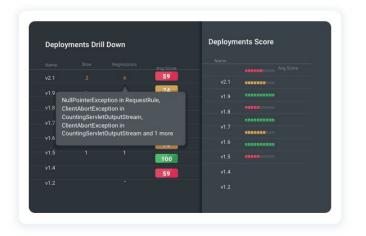
Route issues to the right developer using your tool of choice.

Never Promote Bad Code Again



Assign Issues to the Right Developer

- Use OverOps True Root Cause to recreate issues and create new test cases.
- Pinpoint each issue and get the complete variable state from the moment it occurred.
- Identify who is accountable for errors and route them back to the right developer.



Catch Critical Errors Before Production

- Automatically detect new and increasing issues.
- Classify anomalies according to when they were introduced, their rate and severity.
- Apply Machine Learning to detect anomalies in Staging and Production environments.

True Root Cause	Recorded Variables	
classPreprocessingRequest	Thread-local state	4 items
(Object ID) 1 bj	Log message	Class che
checksum "Gcqbe6f14RIHBhlgZPoCDg"	classChecksum	null
className "UJligxc01zJ100/UpRXS31	class Name	"UJliqxc0
ALE DUC (CODJEGE	classPreprocessing	Backends
<pre>lal String jarName = classPreprecessingRec lal String className = classPreprecessing</pre>	(Object ID)	
al ServiceId = PreprecessedSourceCodeKey	checksum	"Gcqbe6f
al Set <string> relatedClassNames =</string>	className	"UJliqxc0
Map unprocessedItems =	jarName	″J185388
failedBatch.getUnprocessedItems();	codeAnalveisInterface	codeAnal

Measure the Reliability of Your App

- Score every release to tackle issues before they're introduced into production.
- Increase code coverage automatically without depending on unit tests to find errors.
- Configure CI/CD quality gate thresholds to block poor quality builds.