

Flaky Test Handler Plugin

Plugin Information

View Flaky Test Handler [on the plugin site](#) for more information.

This plugin is used to provide various support for handling flaky tests. It currently supports for Git and Maven. It includes support for the latest version of the Maven surefire plug-in which produces additional data about test flakiness using the new "rerunFailingTestsCount" option. It also supports re-running only failed tests for a failed build at the exact failed Git revision. Finally it aggregates statistics of tests (passes, fails and flakes) over Git revisions.

Introduction and Prerequisite

A flaky test is a test which could fail or pass for the same configuration. Flaky tests could be harmful for developers because their failures do not always indicate bugs in the code.

This plugin is designed to handle flaky tests, including re-running failed tests, aggregate and report flaky tests statistics and so on.

Note: Currently most features of this plugin are designed for Git, Maven and freestyle build projects.

Publish re-run information for flaky tests

We recently made some contributions to Maven Surefire to add the new option "rerunFailingTestsCount". It lets users to choose to re-run failed tests up to N times, and if it passes within any of those N times, the re-run will stop and the test will be marked as a "Flake". The build will be marked as successful if there is no failed tests but only "flaky" tests.

The new option may be release with Maven Surefire 2.18 in the near future: <https://github.com/apache/maven-surefire/commit/fefaae7f0534a59f52c046a64c96987e8561dd48>

So the first part of our plugin is to integrate with this new feature from Maven Surefire. Once the "rerunFailingTestsCount" is used either by command line or in pom.xml, the plugin will parse the report and display flaky tests, output information of all the re-runs(stacktrace, output, etc) on test result page.

Configuration:

After the plugin is installed and JUnit test report is published, check on "Publish JUnit flaky test reports" under "Additional test report features".

Publish JUnit test result report

Test report XMLs

[Fileset "includes" setting that specifies the generated raw XML report files, such as "myproject/t"](#)

Retain long standard output/error

Additional test report features Publish JUnit flaky tests reports

Usage:

Display all the re-runs in the test result page:

Passed

com.google.HelloWorldFlakyCotTest.testHelloWorldText2Flaky20



Flaked with 1 failures

Run 1:

Error Message

expected:<[Hello world]> but was:<[Goodbye, cruel world.]>

Stacktrace

```
org.junit.ComparisonFailure: expected:<[Hello world]> but was:<[Goodbye, cruel world.]>
    at org.junit.Assert.assertEquals(Assert.java:115)
    at org.junit.Assert.assertEquals(Assert.java:144)
    at com.google.HelloWorldFlakyCotTest.testHelloWorldText2Flaky20(HelloWorldFlakyCotTest.java:27)
```

Run 2: PASS

Display flaky tests information in the test result table as a badge:

Test Result : com.google

0 failures (±0)

All Tests

Class
HelloWorldFlakyCotTest 🌞 1 tests flaked among all the passing tests
HelloWorldFlakyErrorTest 🌞 0 tests flaked among all the passing tests
HelloWorldTest 🌞 0 tests flaked among all the passing tests
SimpleTest 🌞 0 tests flaked among all the passing tests

Aggregating Tests Statistics over Revisions

Configuration:

Add "Publish JUnit flaky stats" as a post build step

Note: This feature requires JUnit test result report being published, and this step has to be placed **after** "Publish JUnit test result report".

Usage:

For each Git revision and each test, we count how many times it passed, how many times it failed. So if a test passed and failed for the both revision, we count it as a flake; if it always failed, then it's a fail, otherwise it's a pass.

If Git is not used as scm, then we use build number as revision number.

Project Flaky

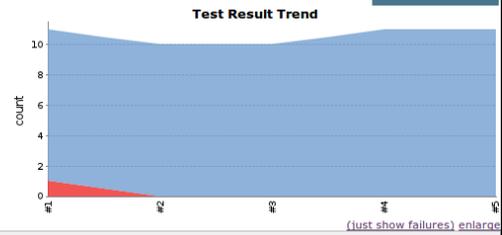
[add description](#)

[Disable Project](#)

 [Workspace](#)

 [Recent Changes](#)

 [Latest Test Result](#) (no failures)



Flaky History

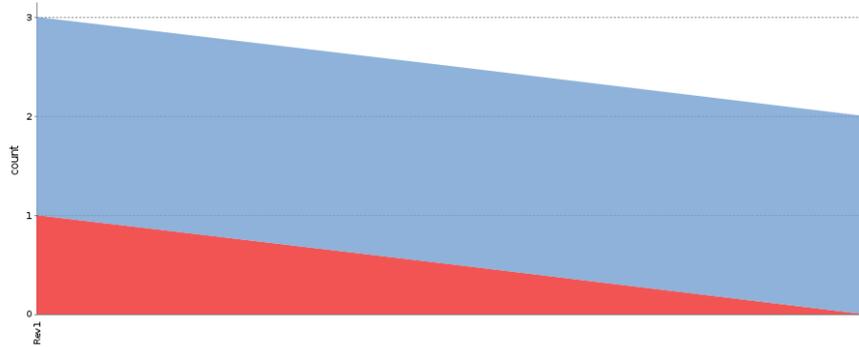
Test Name	Passes	Fails	Flakes
com.google.HelloWorldFlakyCoTest.testHelloWorldText2Flaky20	2	0	0
com.google.HelloWorldFlakyCoTest.testHelloWorldTextFlaky20	2	0	0
com.google.HelloWorldFlakyErrorTest.testHelloWorldText2Flaky20	1	0	1
com.google.HelloWorldFlakyErrorTest.testHelloWorldTextFlaky20	1	0	1
com.google.HelloWorldTest.testHelloWorldText	2	0	0
com.google.HelloWorldTest.testHelloWorldText2Flaky20	2	0	0
com.google.HelloWorldTest.testHelloWorldTextFlaky20	1	0	1
com.google.SimpleTest.testAddition	2	0	0
com.google.SimpleTest.testAdditionThree	1	0	0
com.google.SimpleTest.testAdditionTwo	2	0	0
com.google.SimpleTest.testDivision	2	0	0

Permalinks

- [Last build \(#5\), 6 hr 18 min ago](#)
- [Last stable build \(#5\), 6 hr 18 min ago](#)
- [Last successful build \(#5\), 6 hr 18 min ago](#)
- [Last failed build \(#1\), 22 hr ago](#)
- [Last unstable build \(#4\), 22 hr ago](#)
- [Last unsuccessful build \(#4\), 22 hr ago](#)

com_google_HelloWorldFlakyErrorTest_testHelloWorldText2Flaky20

Rev Number	Revision	Passes	Fails
Rev 1	2718a4984bb4717b78ba8d89922ae10e31048d4b	2	1
Rev 2	0b00b1b06d5c63f1e9d7e90b5e65d74d4e490619	2	0



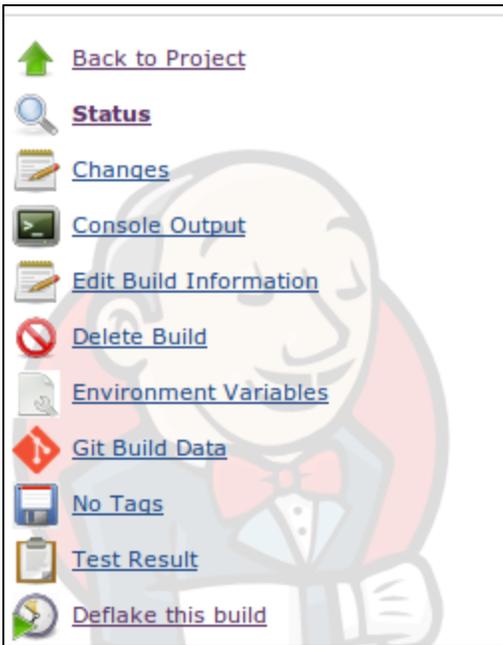
"Deflake" Action for Builds with Failed Tests

Usage:

For each failing build we provide a "Deflake" action. It is different from rebuilding the project, as it will:

- 1) Checkout the exact revision of that failing build (GIT)
- 2) Only run all those failed tests.

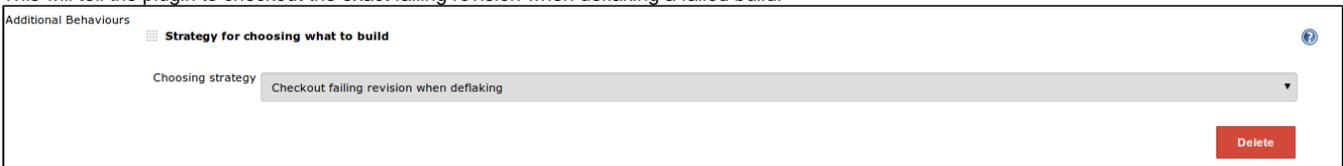
So this will give developers a good idea about how many tests are flaky for this build.



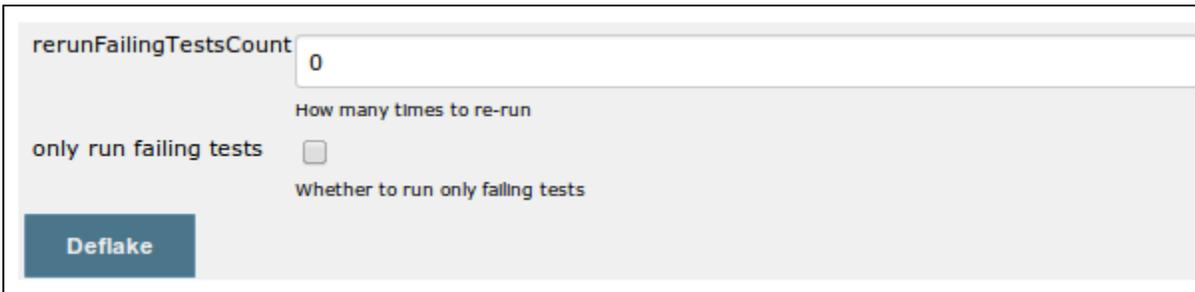
Configuration:

When configuring project's scm with Git plugin, use Add button to add "Strategy for choosing what to build", and then select "Checkout failing revision when deflaking".

This will tell the plugin to checkout the exact failing revision when deflaking a failed build.



After clicking "Deflake", there is another configuration window.



"rerunFailingTestsCount" is the new Surefire option, it can be set and passed to the deflake build. However if this parameter is already set in the project pom.xml, then it cannot be overwritten.

"only run failing tests": when checked, the plugin will generate -Dtest=... to pass to Maven, so only previous failed tests will be re-run.

Deflake build will show up as "Deflake Build # ...".

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build with Parameters](#)

[Delete Project](#)

[Configure](#)

[Rebuild Last](#)

Build History (trend)

#301: Deflake Build# 294	Jul 10, 2014 7:07:10 PM
#300: Deflake Build# 294	Jul 10, 2014 6:59:20 PM
#299: Deflake Build# 294	Jul 10, 2014 6:23:09 PM
#298: Deflake Build# 294	Jul 10, 2014 6:22:44 PM
#297: Deflake Build# 294	Jul 10, 2014 3:04:01 PM
#296: Deflake Build# 294	Jul 10, 2014 3:02:38 PM
#295: Deflake Build# 294	Jul 10, 2014 2:56:13 PM
#294	Jul 10, 2014 2:55:35 PM
#293	Jul 10, 2014 2:55:19 PM
#292	Jul 10, 2014 2:54:39 PM
#291	Jul 10, 2014 11:41:25 AM

Project Flaky



[Workspace](#)



[Recent Changes](#)



[Latest Test Result](#) (no failures)

Flaky History

Test Name

com.google.HelloWorldTest.testHelloWorldTextFlaky20
com.google.HelloWorldFlakyCotTest.testHelloWorldText2Flaky20
com.google.HelloWorldFlakyCotTest.testHelloWorldTextFlaky20
com.google.SimpleTest.testAddition
com.google.HelloWorldTest.testHelloWorldText
com.google.SimpleTest.testDivision
com.google.HelloWorldTest.testHelloWorldText2Flaky20
com.google.HelloWorldFlakyErrorTest.testHelloWorldTextFlaky20