# Fighting Flaky Tests at Scale

Inez Korczyński
DPE Summit 2024

DPE SUMMIT

# Block

Inez Korczyński
Android Developer Experience

1

**Flaky tests**

The problem and the context

2

**Initial response**

Disabling flaky tests...

3

**Evolving solutions**

Detecting flaky tess early on

4

**Impact & results**

# The Problem: Flaky Tests

# Context at Block

# Context at Block

- Multiple mono/mega repos: Android, iOS, Java, Go, …

# Context at Block

- Multiple mono/mega repos: Android, iOS, Java, Go, …
- Multiple CI systems: Kochiku, Jenkins

# Context at Block

- Multiple mono/mega repos: Android, iOS, Java, Go, …
- Multiple CI systems: Kochiku, Jenkins
- Multiple data collection pipelines

# Initial response

# Initial response

Semi-automatically disabling flaky tests and notifying team

# Evolving Solutions

# Ignore flaky tests #105494

Edit | Code

Merged | code-health-for... merged 1 commit into `master` from `disable_flaky_tests` | 1 hour ago

Conversation 2 | Commits 1 | Checks 1 | Files changed 3 | +4 −0

Changes from all commits | File filter | Conversations | 0 / 3 files viewed | Ask Copilot | Review in codespace | Review changes

Filter changed files

- apps
  - t2-restaurants/app/src/androidT...
    - T2RstOpenCheckCartTest.kt
    - T2RstTrueHoldsTest.kt
  - x2-retail/app/src/androidTest/jav...
    - X2RetailCartV2LoyaltyTest.kt

1

apps/t2-restaurants/app/src/androidTest/java/com/squareup/instrumentation/tests/t2/rst/T2RstOpenCheckCartTe…

Viewed

```
@@ -378,6 +378,7 @@ class T2RstOpenCheckCartTest {
378         .selectCheck("Test Order")                    | 378         .selectCheck("Test Order")
379     }                                                 | 379     }
380                                                       | 380
                                                          | 381 +   @Ignore("RST-35711")
381     @Test                                             | 382     @Test
382     fun printBillWithUnsavedItemsExistingTicket() {   | 383     fun printBillWithUnsavedItemsExistingTicket() {
383       setupOrdersAndLogin()                           | 384       setupOrdersAndLogin()
```

1

apps/t2-restaurants/app/src/androidTest/java/com/squareup/instrumentation/tests/t2/rst/T2RstTrueHoldsTest.kt

Viewed

```
@@ -213,6 +213,7 @@ class T2RstTrueHoldsTest {
213     * This tests a specific case with splitting where a printed item wouldn't have its | 213     * This tests a specific case with splitting where a printed item wouldn't have its
214     *  PRINT event saved and would be re-printed when the ticket was next opened.       | 214     *  PRINT event saved and would be re-printed when the ticket was next opened.
215     */                                                                                 | 215     */
                                                                                          | 216 +   @Ignore("RST-35712")
216     @Test                                                                             | 217     @Test
217     fun splitTicketIntoTwoWithOnFireItems() {                                         | 218     fun splitTicketIntoTwoWithOnFireItems() {
218       setupCheckWithStartersOnHold(                                                   | 219       setupCheckWithStartersOnHold(
```

> **Tip**
>
> Below is the Code Health bot's best guess at what could be causing flakiness in these tests.

Based on the provided Go test file and the error messages associated with the flaky test, there are a couple of potential issues in the test code that may be contributing to the flakiness:

1. Deadlock caused by improper use of locks:
   In the test case "it returns an error", a mutex lock is used to control the access to `callCount` variable. However, there's an issue within the handler function that may lead to a deadlock. Specifically:

```
handler: func(input int) (byte, error) {
    callMutex.Lock()
    if callCount == 3 {
        return 0, errors.New("failed job")
    }
    callCount += 1
    callMutex.Unlock() // The unlock should be deferred right after .Lock()
    return '0' + uint8(input), nil
},
```

   The `Unlock()` function is only called when the `callCount` is not `3`. If the `callCount` is `3`, the function will return without unlocking the mutex, potentially causing a deadlock when another goroutine attempts to lock it. This should be rectified by deferring the unlock operation right after the lock is acquired. Like so:

```
handler: func(input int) (byte, error) {
    callMutex.Lock()
    defer callMutex.Unlock() // Use defer to ensure that Unlock is called.
    if callCount == 3 {
        return 0, errors.New("failed job")
    }
    callCount += 1
    return '0' + uint8(input), nil
},
```

# Stress Testing

# Common Reasons of Flakiness

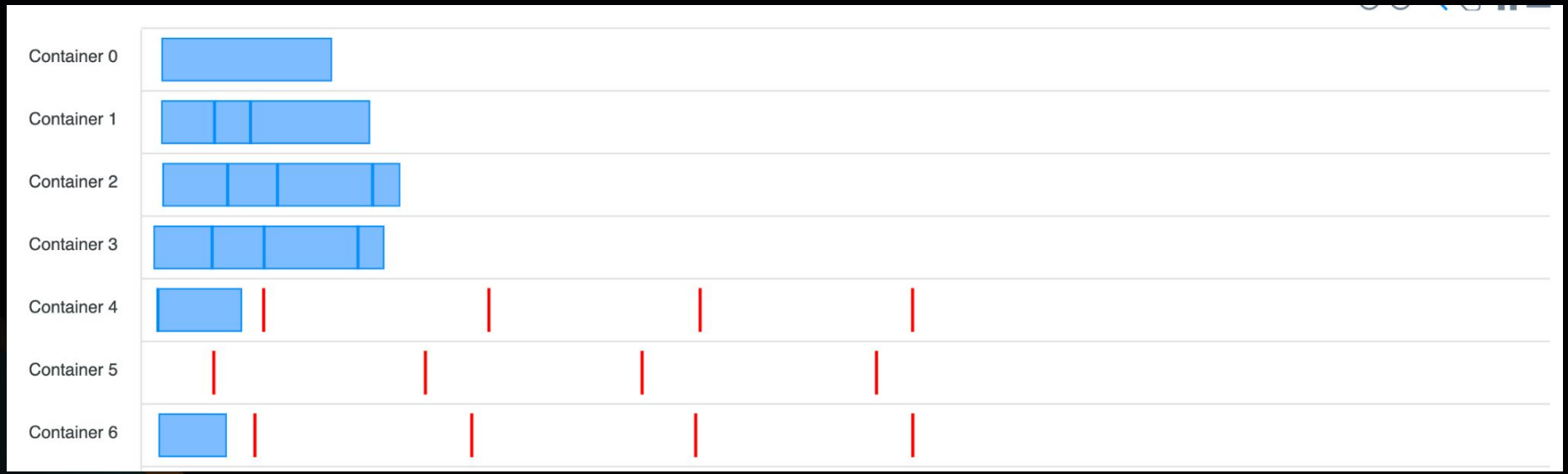
```
dimension: stacktrace_snippet_normalized {
  type: string
  sql: REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_REPLACE(REGEXP_
    '<[0-9]{10}([/>])', '<REDACTED_ID\\1'),
    'DefaultDispatcher-worker-[0-9]{1,2}', 'DefaultDispatcher-worker-REDACTED_ID'),
    '[0-9a-z]{8}-[0-9a-z]{4}-[0-9a-z]{4}-[0-9a-z]{4}-[0-9a-z]{12}', 'REDACTED_GUID'),
    '@[0-9a-z]{5,10}', '@REDACTED_ID'),
    'for [0-9]{1,5} iterations', 'for REDACTED_NUMBER iterations'),
    'id=[0-9]{10}', 'id=REDACTED_ID'),
    'Sq-io-server-[0-9]{3}', 'Sq-io-server-REDACTED_ID'),
    'when=-[0-9]+ms', 'when=Nms'),
    'Timed out waiting for condition (\(.*\))', 'Timed out waiting for condition (<duration>)'),
    'IllegalStateException: Could not delete: (.*) because it isn\'t empty, contains: (.*)\n', 'IllegalStateException:
    '(^|\\n)\\s+', '\\1'),
    '\\.safeWaitUntil\\(ComposeScreenRobot\\.kt:\\d{1,4}\\)','.safeWaitUntil(ComposeScreenRobot.kt:LINE_NUMBER)');;
}
```

# Advanced Detection of New Flakes

# Eager Test Suppression

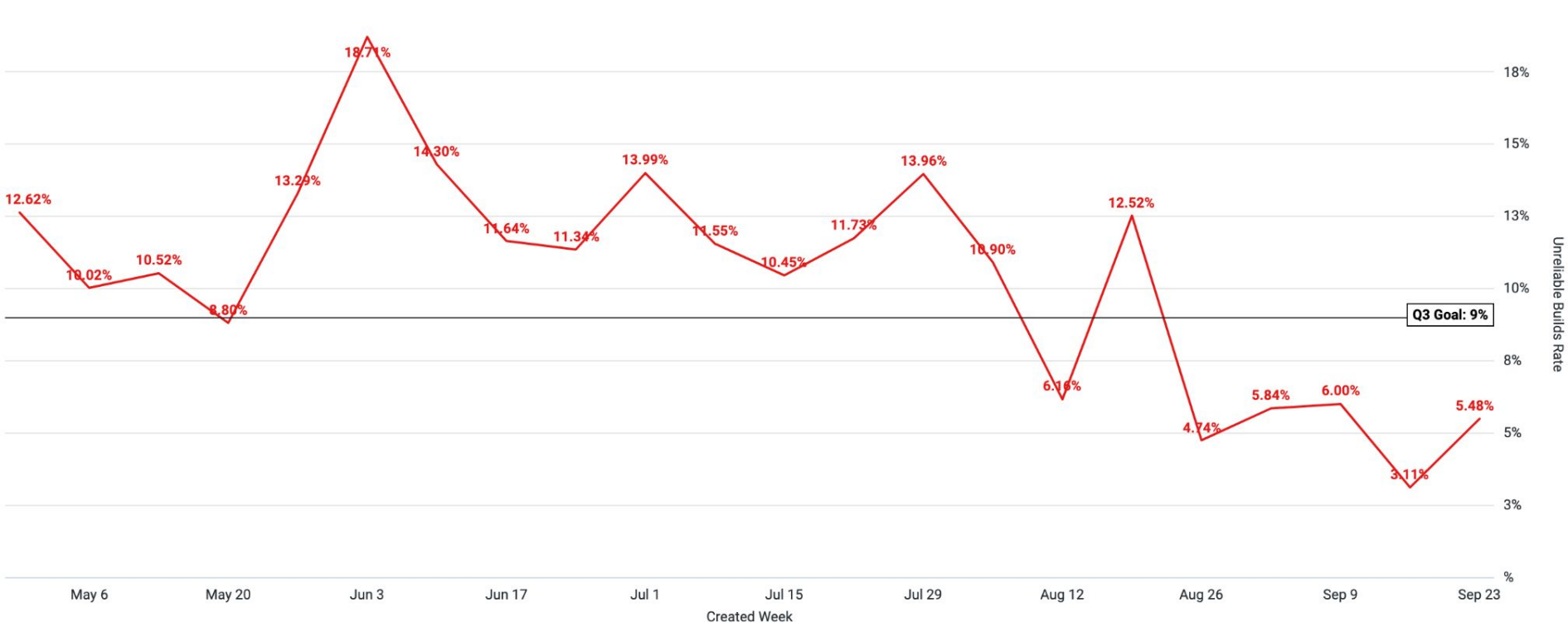
# Dynamic retries

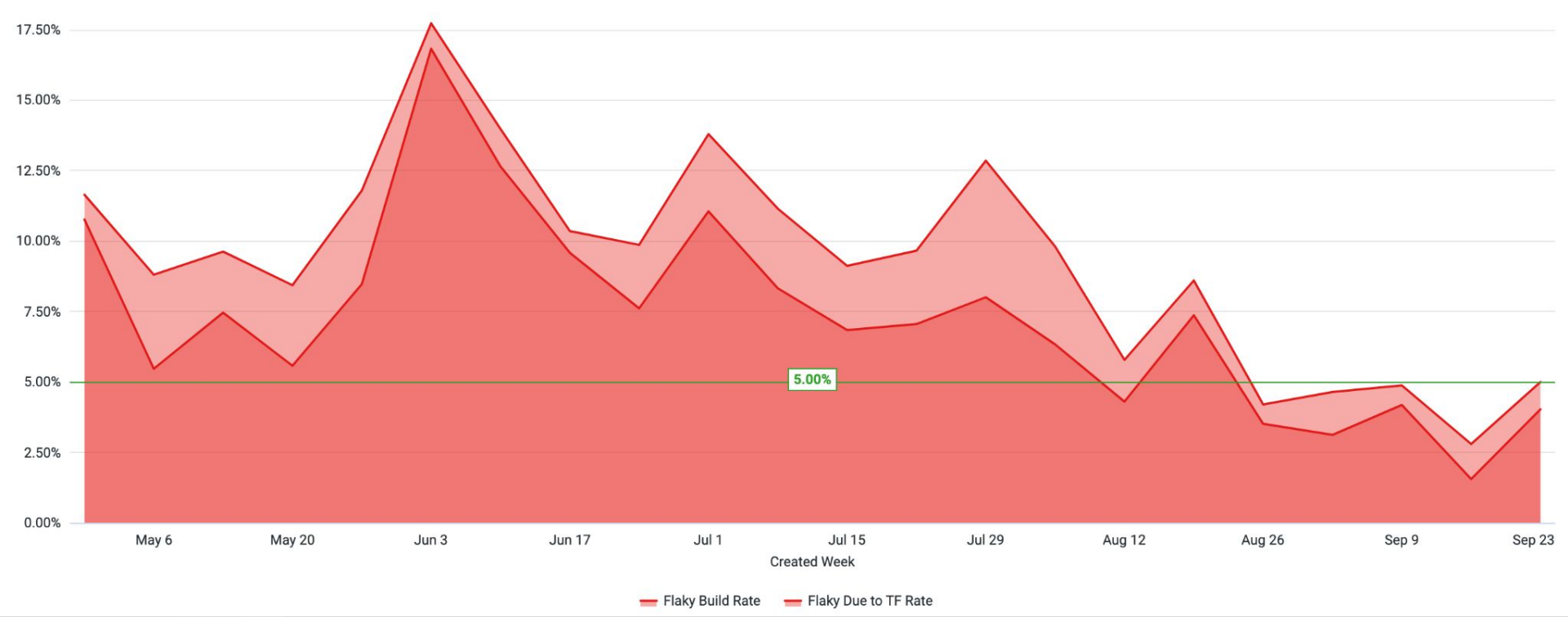
- Number of retries = 3


Container 0
Container 1
Container 2
Container 3
Container 4
Container 5
Container 6

# Dynamic retries

- ~~Number of retries = 3~~
- Number of retries = <depend on the historical data>


12.62%
10.02%
10.52%
8.80%
13.29%
18.71%
14.30%
11.64%
11.34%
13.99%
11.55%
10.45%
11.73%
13.96%
10.90%
6.16%
12.52%
4.74%
5.84%
6.00%
3.11%
5.48%
Q3 Goal: 9%
18%
15%
13%
10%
8%
5%
3%
%
Unreliable Builds Rate
May 6
May 20
Jun 3
Jun 17
Jul 1
Jul 15
Jul 29
Aug 12
Aug 26
Sep 9
Sep 23
Created Week


17.50%
15.00%
12.50%
10.00%
7.50%
5.00%
2.50%
0.00%
5.00%
May 6
May 20
Jun 3
Jun 17
Jul 1
Jul 15
Jul 29
Aug 12
Aug 26
Sep 9
Sep 23
Created Week
Flaky Build Rate
Flaky Due to TF Rate

# Strategy & next steps

# THANKS

inez@block.xyz