
A TELUS Adventure in Diagnosing and Defeating
**Non-Deterministic** Pipeline Failures

TELUS

NotebookLM

# Our Mission Briefing

## 1. The CI/CD Battlefield

A developer's daily struggle with mysterious pipeline failures.

## 2. Unmasking the Enemy

Introducing a systematic approach to identify and categorize every type of flaky failure.


## 3. Calculating the Damage

Quantifying the real cost of these failures in time and resources.

## 4. Tracking Their Movements

Analyzing failure patterns to see which are old news and which are new threats.

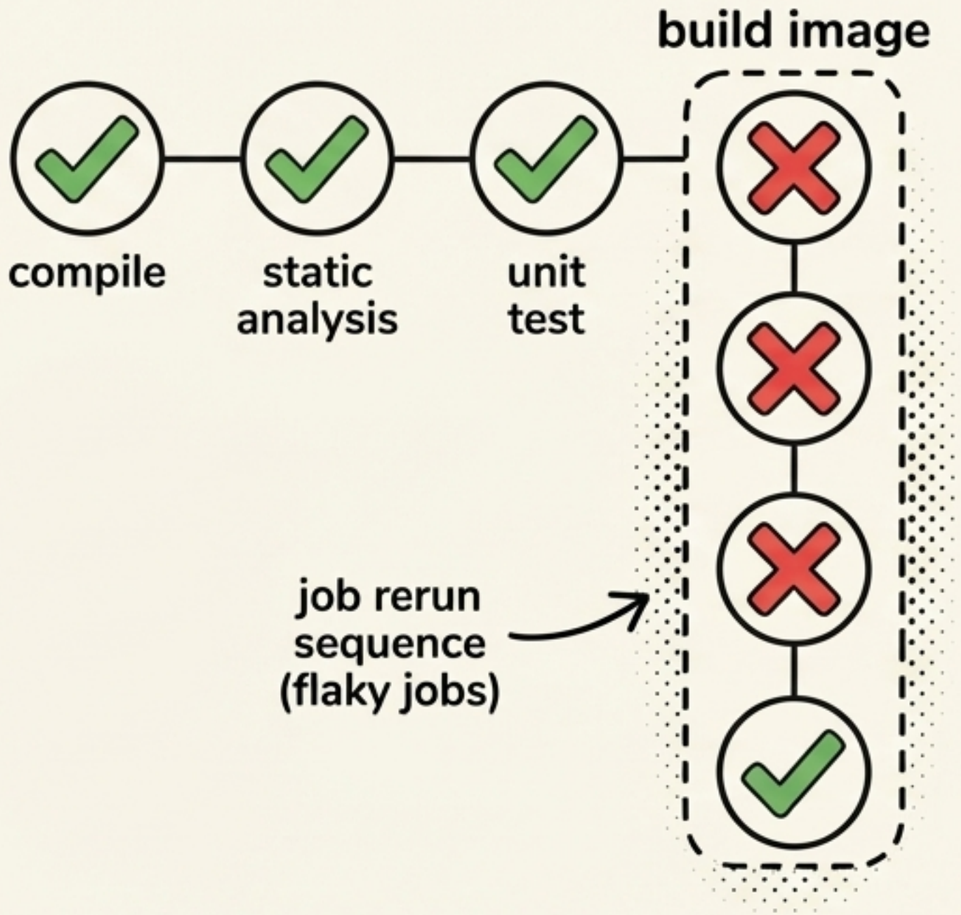
## 5. The Ultimate Gadget

Unveiling a powerful model to prioritize the most critical failures.


## 6. The Battle Plan

Our final, data-driven strategy to conquer flaky failures.

# A Quarter of Our CI Failures are Ghosts in the Machine




At TELUS, Continuous Integration (CI) pipelines are crucial for rapid software delivery. But they are haunted by "flaky" failures—jobs that fail unpredictably for reasons unrelated to code changes.

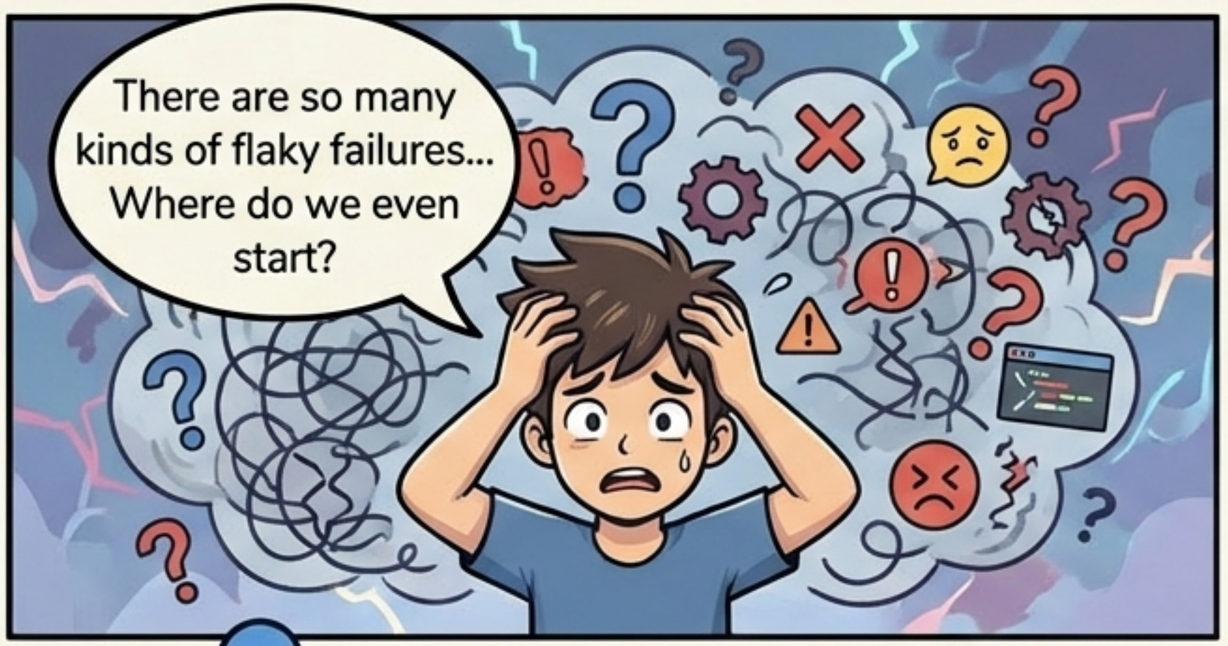
# 1 in 4

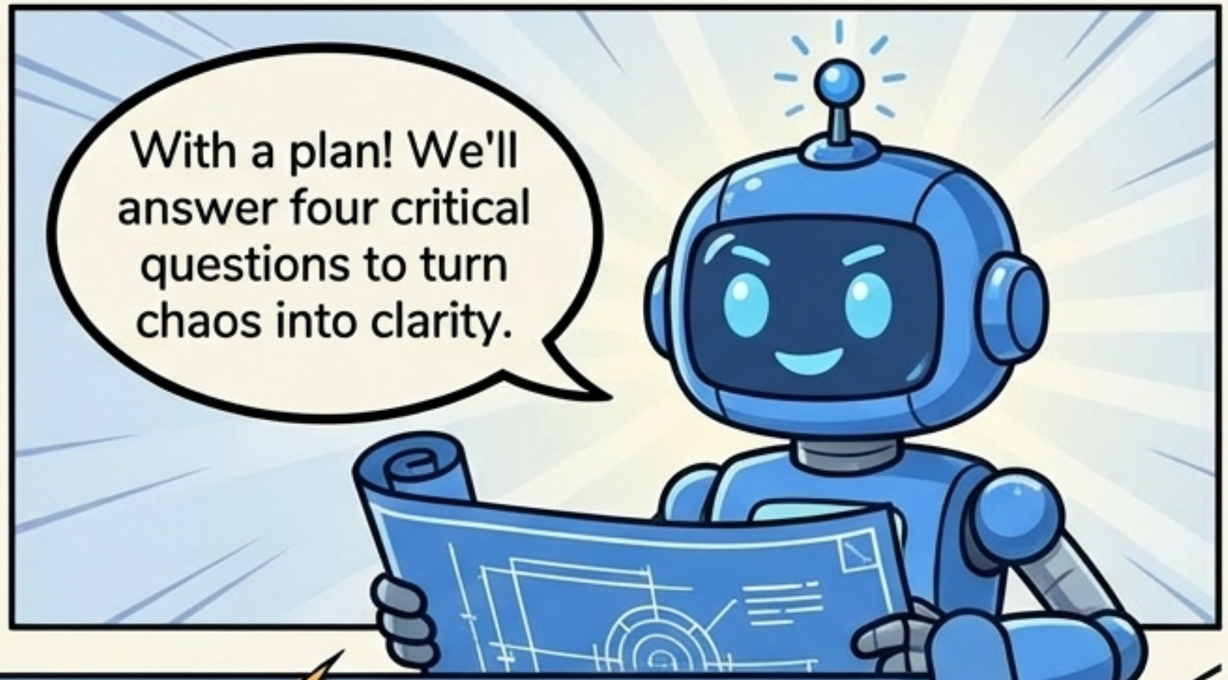
## (25%)

## of all failed jobs at TELUS is flaky


- **Wasted Developer Time:** Endless hours spent diagnosing non-existent code bugs.
- **Delayed Releases:** Critical updates are stalled, waiting for pipelines to pass.
- **Resource Drain:** Multiple job reruns consume valuable machine resources and overload CI servers.

# TO DEFEAT THE ENEMY, WE MUST FIRST UNDERSTAND IT




## THE BLUEPRINT

- **RQ1:** What are the different **types** of flaky failures?
- **RQ2:** Which types are the most **costly**?
- **RQ3:** How have these failures **evolved** over time?
- **RQ4:** Which ones should we prioritize **first**?

# Our Gadget: The “Flaky-Failure Labeling Gun”

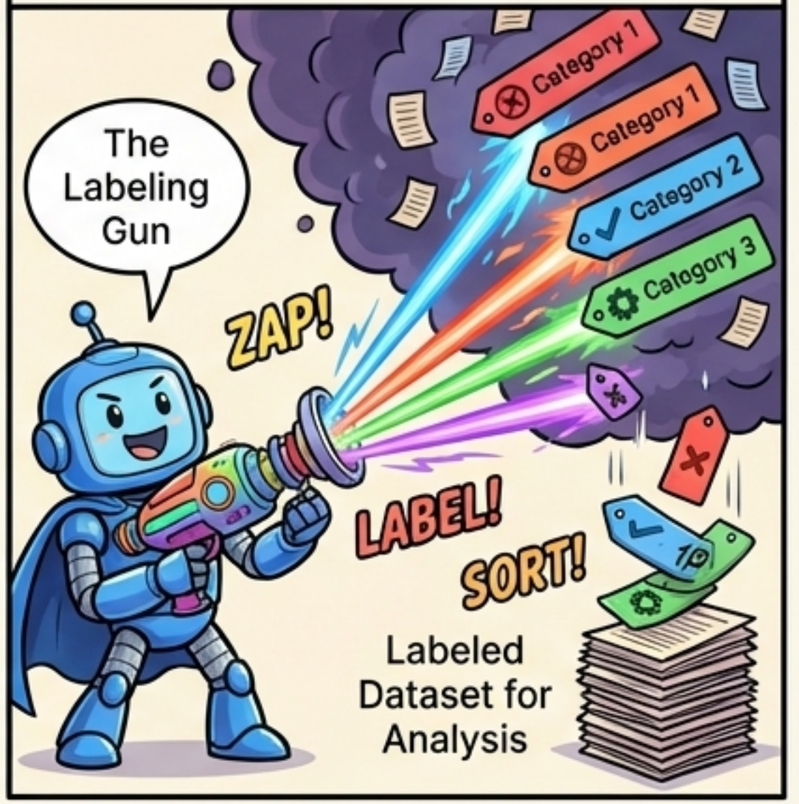
## Collect Samples


We started with thousands of unlabeled failure logs.

## Manual Analysis & Rule Creation


We manually analyzed a statistically significant sample to create 46 distinct categories and regex rules to find them.

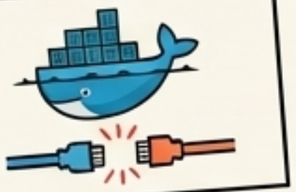
## Automated Labeling




The tool then automatically labeled 4,511 failures with 91% precision, giving us our dataset for analysis.

# The Rogues' Gallery: Meet the 46 Types of Flaky Failures

## WANTED



docker daemon connection failure

7.2%

## WANTED

job execution timeout

6.78%

## WANTED


## Misconfigured Environment Variable

**Accounts for 15% of all flaky failures!**

Caused by missing, misspelled, or incorrect environment variables during job execution.

## WANTED

container platform auth failure

6.96%

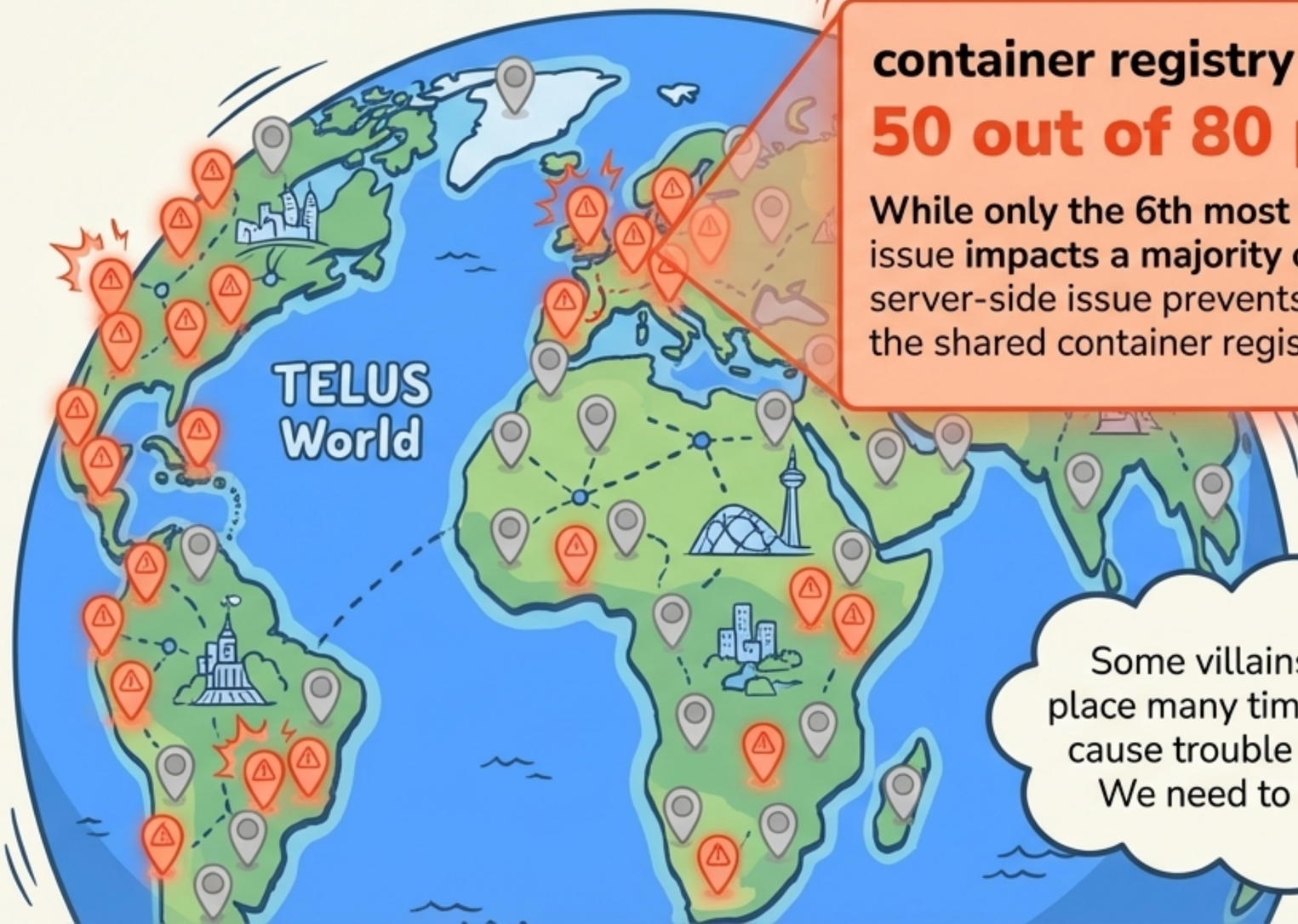
## WANTED

image not found

4.9%

**Together, these top 5 culprits are responsible for over 40% of all flaky failures.**

# Frequency Isn’t Everything: The Most Widespread Gremlin Affects 60% of Projects

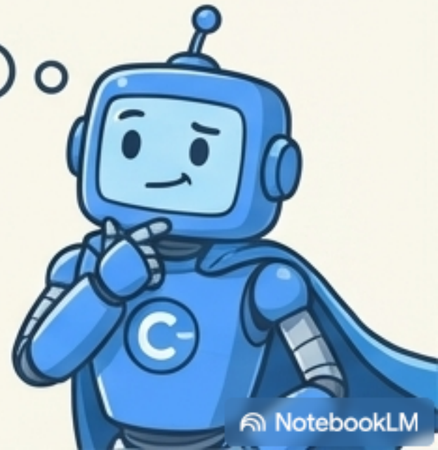



**container registry server error**

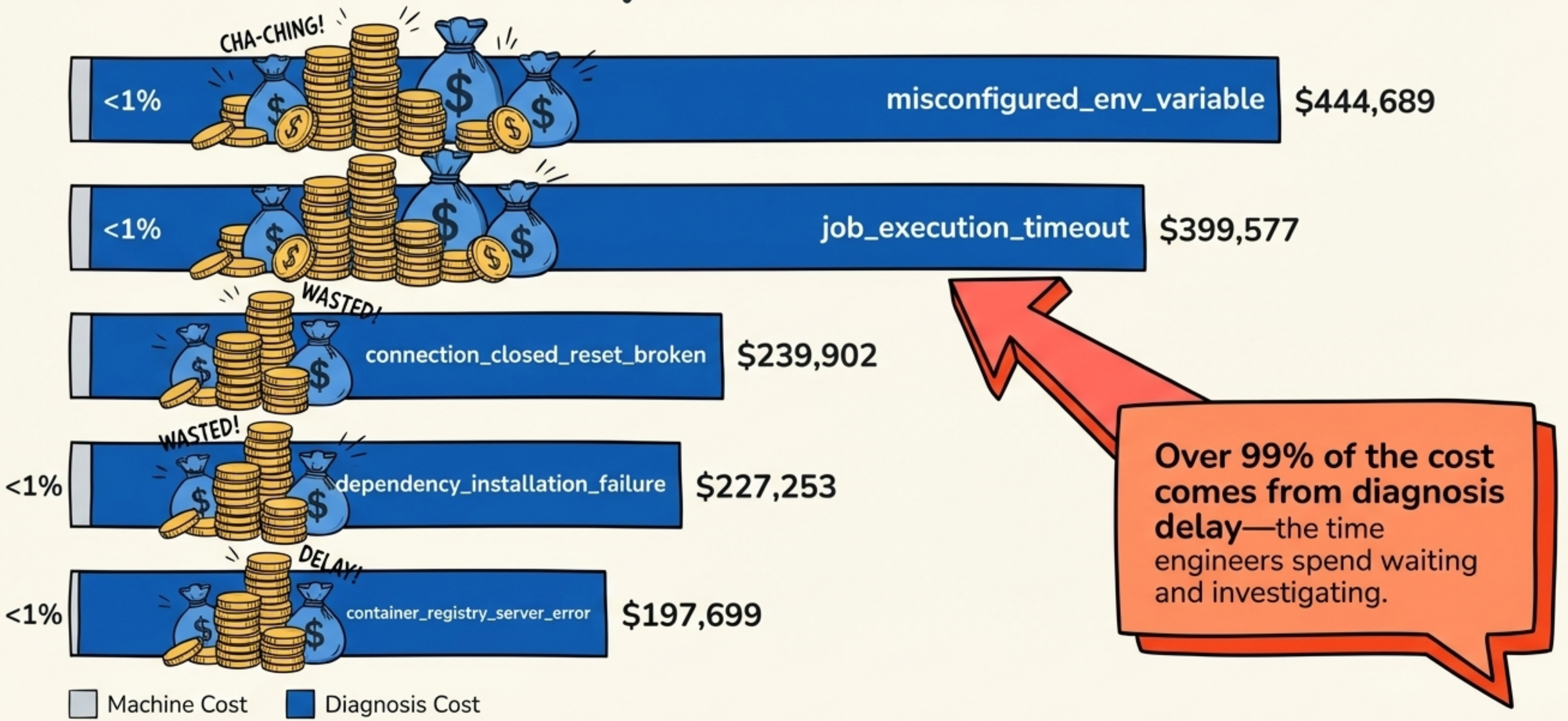
## 50 out of 80 projects (62.5%)

**While only the 6th most frequent failure (4.72%)**, this issue **impacts a majority of our teams.** It occurs when a server-side issue prevents jobs from pulling images from the shared container registry, acting as a single bottleneck.

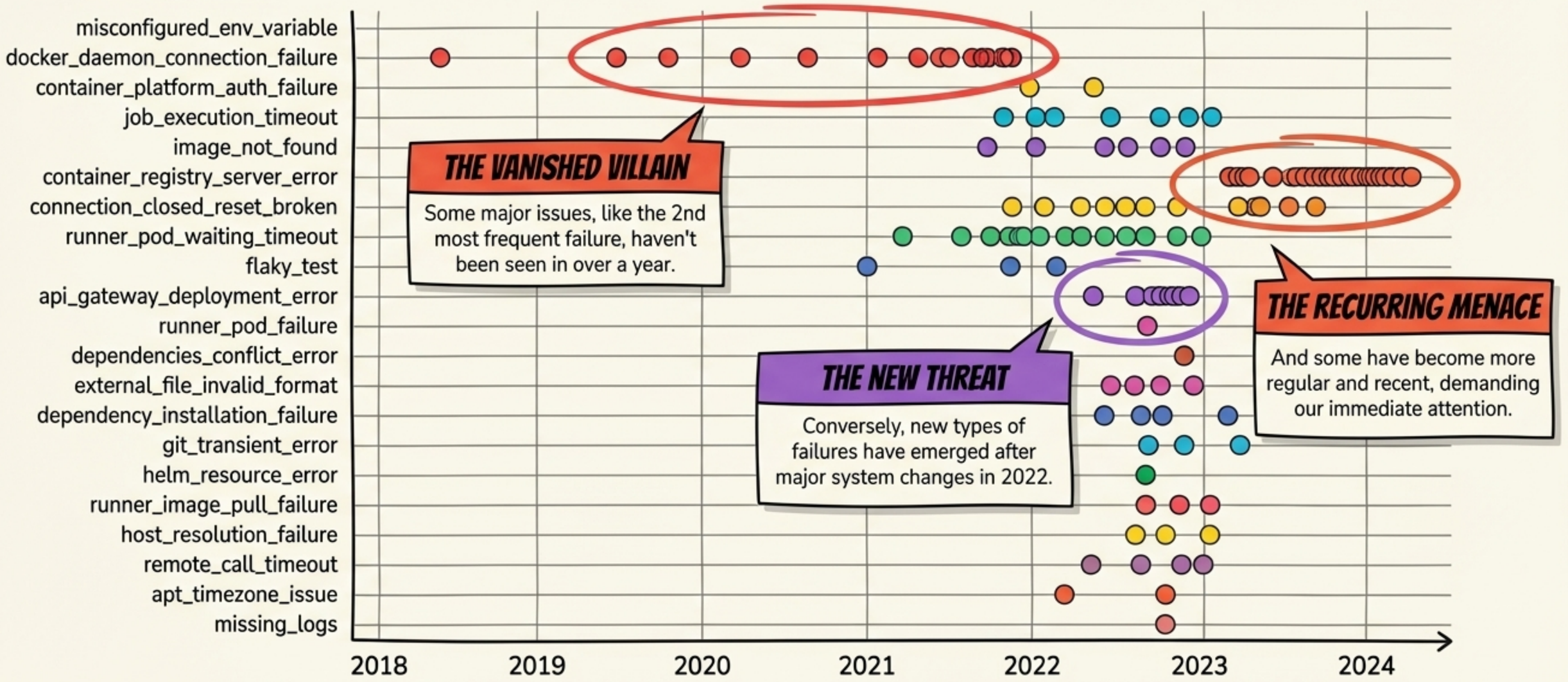
Some villains strike one place many times, but others cause trouble everywhere! We need to track both.



# The True Cost of Flakiness is Measured in Wasted Hours, Not Just Machine Time



# The Battlefield is Always Changing



# Time for a New Gadget: The R-F-M Prioritizer!

## R - Recency

How *recently* have we seen this failure? We focus on the last three occurrences to measure consistent, current threats.


## F - Frequency

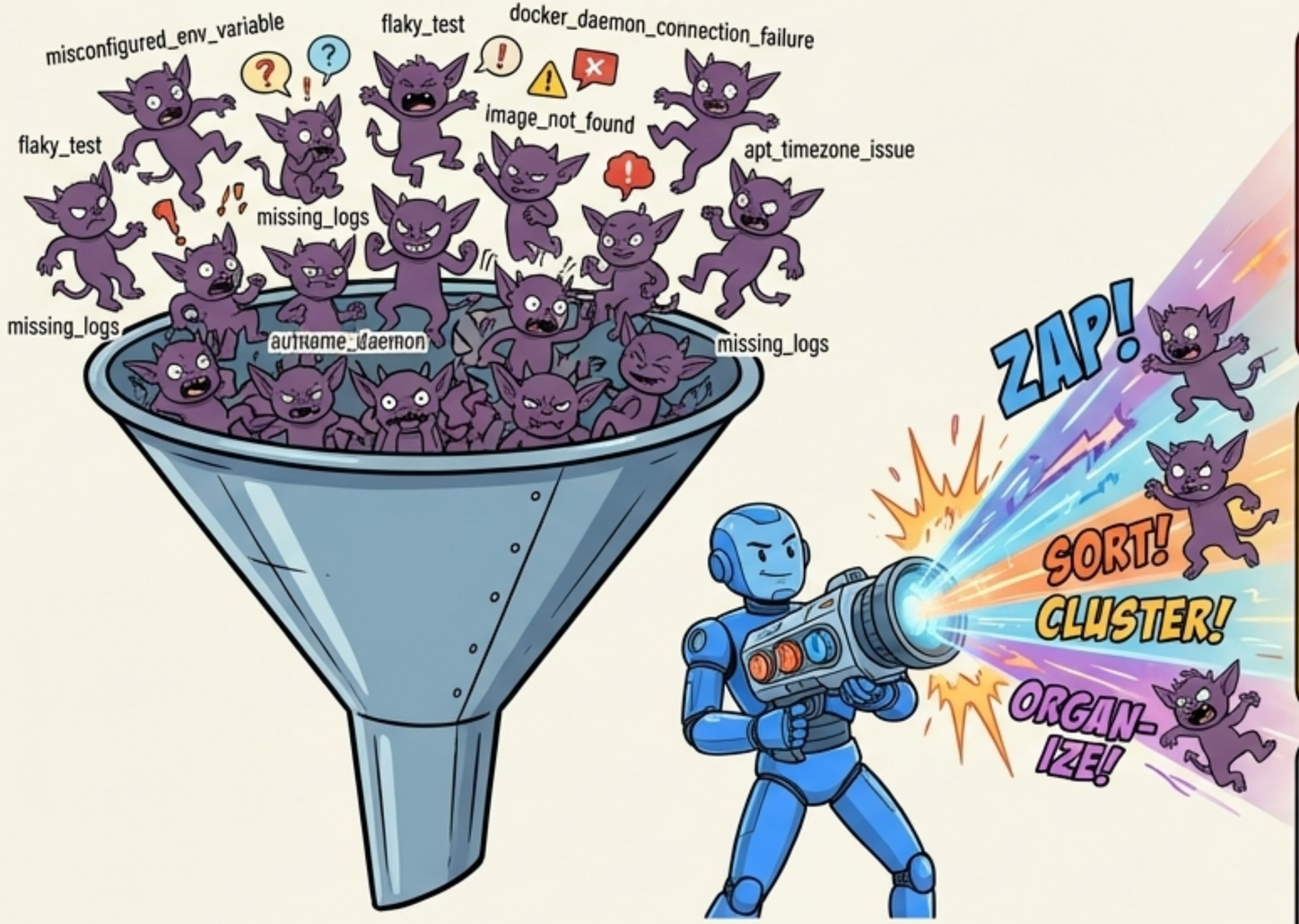
How *often* does this failure occur? We count the total number of times it has appeared.

## M - Monetary

How *costly* is this failure? We use our cost model based on diagnosis time and machine resources.

This gadget doesn't just count the gremlins; it tells us which ones are the biggest, most recent, and most expensive threats *right now*!

# Sorting the Threats: From 46 Categories to 8 Actionable Clusters




## High Priority ($R↑F↑M↑)
Recent, Frequent, Costly

## Medium Priority ($R↑F↑M↑)
Also recent, frequent, costly, but less extreme

## Emerging Issues ($R↑F↓M↓$)
Very recent, but not yet frequent or costly

## Idle Threats ($R↓F↑M↑$)
Frequent & costly, but haven't appeared recently

## Grey Units ($R↓F↓M↓)

### Relics
old, low frequency, low cost

### Negligible
low R, low F, low M

### Sporadic

### Low Impact

Using a K-means clustering algorithm, the RFM model groups failures with similar recency, frequency, and cost profiles.

CLASSIFIED

# The Battle Plan: Our 14 Highest Priority Targets


TARGET #1: ARCH-VILLAIN

## misconfigured env variable

R: 27 days | F: 673 | M: $444k


TARGET #2: ARCH-VILLAIN

## job execution timeout

R: 8 days | F: 306 | M: $399k

### HIGH & MEDIUM PRIORITY LIEUTENANTS

- dependency installation failure
- runner pod waiting timeout
- api gateway deployment error
- container registry server error
- flaky test
- git transient error
- external file invalid format
- host resolution failure
- runner image pull failure
- cloud token limit exceeded
- remote call timeout
- helm resource error

Future efforts on **automated diagnosis and repair** should focus squarely on these **14 categories** to maximize impact.

# On the Horizon: 14 More Threats to Keep on Our Radar

## Emerging Issues ($R↑F↓M↓$)

These 4 categories (like `runner pod not found`) are very recent. While their frequency and cost are currently low, they represent new problems that could grow over time. We must monitor them closely.

## Idle Threats ($R↓F↑M↑$)


These 5 categories (like `runner pod failure`) were highly frequent and costly but haven't surfaced for a while.
Monitoring is essential to ensure they are truly resolved and don't re-emerge.

**A smart strategy isn't just about fighting today's battles, but also anticipating tomorrow's.**

# From Chaos to Clarity: Our Quest is Complete



**We Faced the Chaos:** Acknowledged the 25% flaky failure rate.

**We Profiled the Enemy:** Built a taxonomy of 46 distinct failure types.

**We Assessed the Threat:** Quantified their cost and tracked their evolution.

**We Forged a Weapon:** Used the RFM model to pinpoint the 14 most critical targets.

**We now have a data-driven strategy to systematically reduce waste and improve the reliability of our CI/CD pipelines.**

# The Blueprint for Conquering Flaky Failures

## A Taxonomy for Diagnosis

We provide a comprehensive taxonomy of 46 flaky failure categories, offering a starting point for any organization to begin diagnosing their own CI/CD issues.

## The RFM Prioritization Framework


We introduce a novel approach using Recency, Frequency, and Monetary (RFM) analysis to prioritize technical issues, a framework that can be adapted to many other contexts beyond flaky jobs.

**This work lays the foundation for building targeted, automated solutions that fix the problems that matter most.**