Zap the Flakes!

Leveraging AI to Combat Flaky Tests with CANNIER

SFSCON ‘25

Daniel Hiller

Red Hat

SFSCON

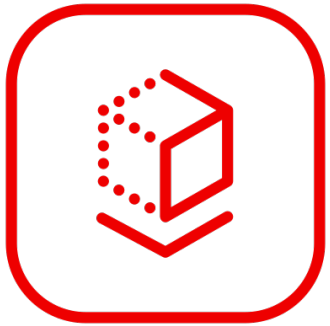
agenda

- about me
- about flakes
- pre-merge-detection v1
- CANNIER
- pre-merge-detection v2
- Q&A

about me

- Software Engineer | OpenShift Virtualization
- KubeVirt | CI & automation in general

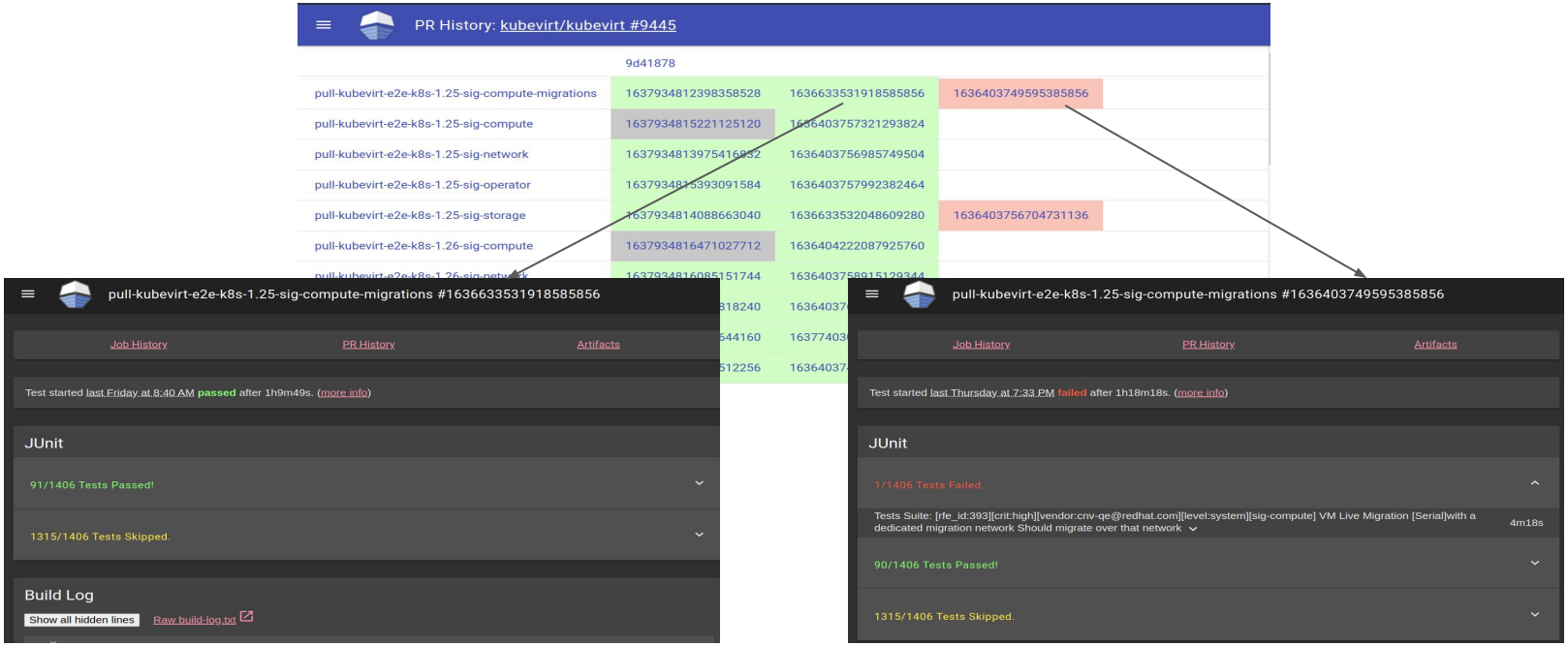
kubevirt.io - Virtualization for Kubernetes



- prow.ci.kubevirt.io
- CI-Health
- Flaky Tests

SFSCON

about flakes



source: https://prow.ci.kubevirt.io/pr-history/?org=kubevirt&repo=kubevirt&pr=9445

SFSCON

about flakes

a ***flake***

is a **test** that

without any code change

will either **fail** or **pass** in successive runs

about flakes

"... of the **91% of developers** who claimed to deal with flaky tests at least a few times a year, ... **23%** *[of developers]* thought that they were a **serious** problem. ..."

"... test flakiness was a **frequently encountered problem**, with ... **15%** *[of developers]* dealing with it **daily**"

source: "A survey of flaky tests"

impact of flakes

impact of flakes

Flaky tests cause

- for individual contributors
  - **prolonged feedback cycles**
  - **test trust issues**
- for the project community
  - **slowdown of merging** pull requests - “retest trap”
  - **reversal of acceleration** effects (i.e. batch testing)
  - **waste** of CI resources

Goal:
flaky tests must not
enter the code base!

The only way to find out the flakiness of a test is to run it as often as you can!

pre-merge-detection v1.5

[check-tests-for-flakes test lane]

why: catch flakes before entering main

gather the exact set of changed tests

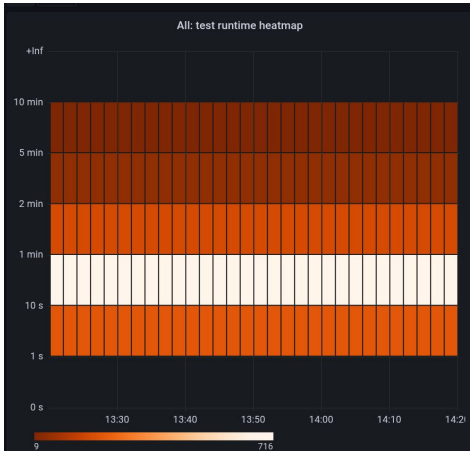
([source])

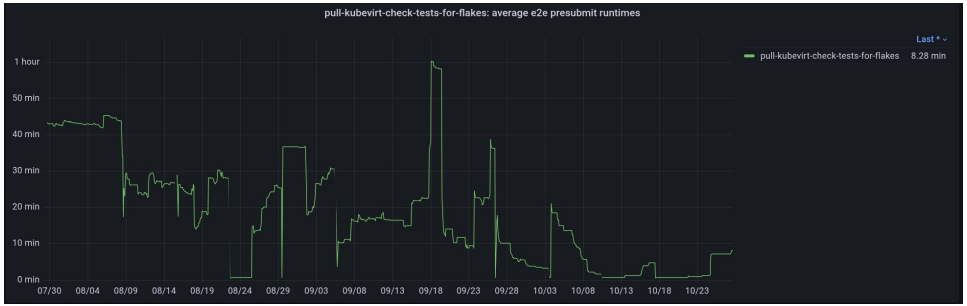
```
ginko_params="$ginko_params -no-color -succinct --label-filter=!QUARANTINE -randomize-all"
for test_file in $(echo "${NEW_TESTS}" | tr '|' '\n'); do
    ginko_params+=" -focus-file=${test_file}"
done

echo "Test lane: ${TEST_LANE}, preparing cluster up"

if [[ ! "$ginko_params" =~ -dry-run ]]; then
    make cluster-up
    make cluster-sync
else
    # Ginkgo only performs -dryRun in serial mode.
    export KUBEVIRT_E2E_PARALLEL="false"
    NUM_TESTS=1
fi

for i in $(seq 1 "$NUM_TESTS"); do
    echo "Test lane: ${TEST_LANE}, run: $i"
    if ! FUNC_TEST_ARGS="$ginko_params" make functest; then
        echo "Test lane: ${TEST_LANE}, run: $i, tests failed!"
        exit 1
    fi
done
```


pre-merge-detection v1.5

problems:

- most e2e tests (~900) take 10sec - 2mins to run
- 5 times re-run has “only” 88% chance of detection
- re-run lane takes ~30mins on average for a small set of tests
- capping amount of tests re-run required





sources: KubeVirt e2e test runtimes, check-tests-for-flakes job history, automation/repeated_test.sh

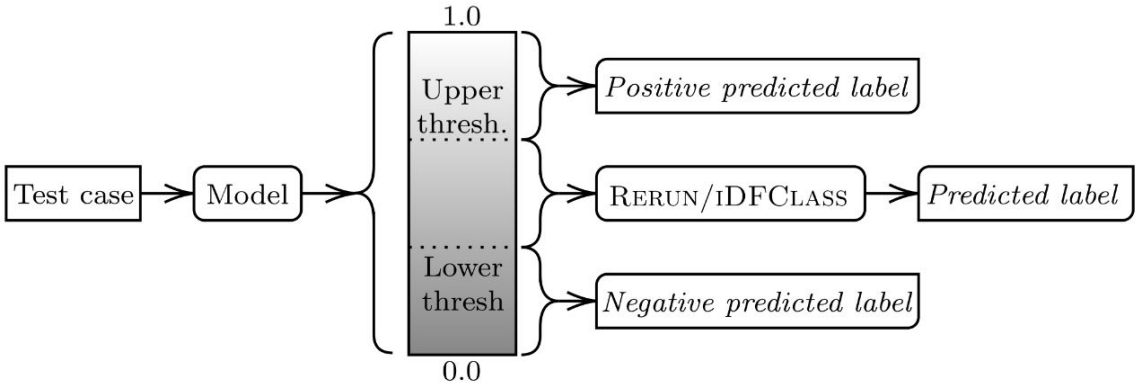
CANNIER

"... we found that CANNIER was able to reduce the time cost (and therefore monetary cost) *[of re-running tests]* by an **average of 88%** ..."

source: https://www.gregorykapfhammer.com/research/papers/parry2023/

CANNIER

single model



source: https://www.gregorykapfhammer.com/research/papers/parry2023/

CANNIER

feature set

#	Feature	Description
1	Read Count	Number of times the filesystem had to perform input [9].
2	Write Count	Number of times the filesystem had to perform output [9].
3	Run Time	Elapsed wall-clock time of the whole test case execution.
4	Wait Time	Elapsed wall-clock time spent waiting for input/output operations to complete.
5	Context Switches	Number of voluntary context switches.
6	Covered Lines	Number of lines covered.
7	Source Covered Lines	Number of lines covered that are not part of test cases.
8	Covered Changes	Total number of times each covered line has been modified in the last 75 commits.
9	Max. Threads	Peak number of concurrently running threads.
10	Max. Children	Peak number of concurrently running child processes.
11	Max. Memory	Peak memory usage.
12	AST Depth	Maximum depth of nested program statements in the test case code.
13	Assertions	Number of assertion statements in the test case code.
14	External Modules	Number of non-standard modules (i.e., libraries) used by the test case.
15	Halstead Volume	A measure of the size of an algorithm's implementation [21, 57, 59].
16	Cyclomatic Complexity	Number of branches in the test case code [39, 57, 59].
17	Test Lines of Code	Number of lines in the test case code [57, 59].
18	Maintainability	A measure of how easy the test case code is to support and modify [19, 71].

source: https://www.gregorykapfhammer.com/research/papers/parry2023/

pre-merge-detection v2.0

questions

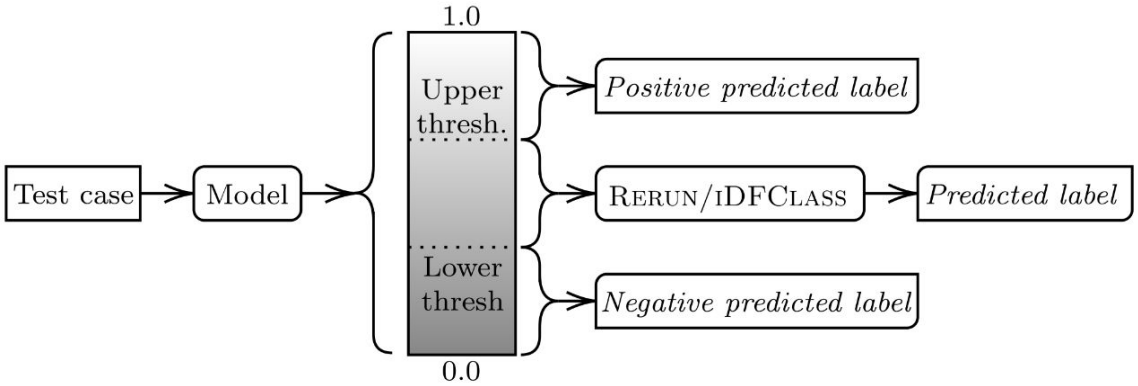
- CANNIER
  - implemented in Python
  - KubeVirt uses Go
- Runtime Data
  - where to store
  - when to capture
- data science
  - Python has well known frameworks
  - Go state unsure

pre-merge-detection v2.0

implementation

parts:

- code (Go)
  - test set extraction
  - feature extraction
  - model prediction
  - model generation
- test lane (Bash)
- model deployment (YAML)



source: https://www.gregorykapfhammer.com/research/papers/parry2023/

pre-merge-detection

possible optimizations

- components of feature vector provide insightful advice to the contributor
  - i.e. high cyclomatic complexity advises to reduction etc.
  - therefore it’s valuable to attach the analysis data to the PR
- possibly increase number of reruns (>5)
  - reduced runtime overall leaves time for more reruns
  - feature vector contains runtimes, thus we can estimate the total re-run time better and optimize for it, i.e. group tests by runtime classes

source: https://www.gregorykapfhammer.com/research/papers/parry2023/

pre-merge-detection v2.1

improvements

parts:

- model
  - generation
  - deployment
- test lane
- code
  - test set extraction
  - feature extraction
  - model prediction

parts:

- model
  - ...
  - **automatic updates**
- ~~test lane~~ -> **prow external-plugin**
  - runs on presubmits
  - runs on postcommits (probably with a larger test set)
  - adds helpful feedback
- code
  - test set extraction
  - feature extraction
  - model prediction

CANNIER

some final remarks:

- CANNIER approach is basically language agnostic
- our implementation is highly dependent on Ginkgo testing framework for Go
- adaptation to default go tests or to other frameworks should be an implementation detail

source: https://www.gregorykapfhammer.com/research/papers/parry2023/

Links

- CANNIER
  - paper: https://www.gregorykapfhammer.com/research/papers/parry2023/
  - implementation: https://github.com/flake-it/cannier-framework/
- KubeVirt
  - Initial pull request (draft): https://github.com/kubevirt/project-infra/pull/3930
  - Presentation Squash The Flakes @ FOSDEM ‘24: https://archive.fosdem.org/2024/schedule/event/fosdem-2024-1805-squash-the-flakes-how-to-minimize-the-impact-of-flaky-tests/

Q&A

Any questions?

Any suggestions for improvement?

Who else is trying to tackle this problem?

What have you done to solve this?

Thank you for attending!

Further questions?

Feel free to send questions and comments to:

dhiller@redhat.com

kubernetes.slack.com/
@dhiller

@dhiller@fosstodon.org

dhiller.dev

contact me:

kubevirt.io

Virtualization for Kubernetes

KubeVirt welcomes all kinds of contributions!

- **Weekly community meeting every Wed 3PM CET**
- Links:
  - KubeVirt website
  - KubeVirt user guide
  - KubeVirt Contribution Guide
  - GitHub
  - Kubernetes Slack channels
    - #virtualization
    - #kubevirt-dev