A Journey Through Integration Testing with Go

What Could Go Wrong?

With Samantha Coyle

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure. Your costs and results may vary.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex

Intel is committed to respecting human rights and avoiding complicity in human rights abuses. See Intel's Global Human Rights Principles. Intel's products and software are intended only to be used in applications that do not cause or contribute to a violation of an internationally recognized human right.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy

No product or component can be absolutely secure. Your costs and results may vary. Results have been estimated or simulated.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others

Overview

Introduction

Test Dependencies

Test Cases

Test Report

+ Challenges -> Learnings

Conclusion

Speaker & Team

Samantha Coyle

Software Engineer

http://samcoyle.me/

Github – sicoyle

Twitter - @thesamcoyle

The Guardians Team

intel

Introduction

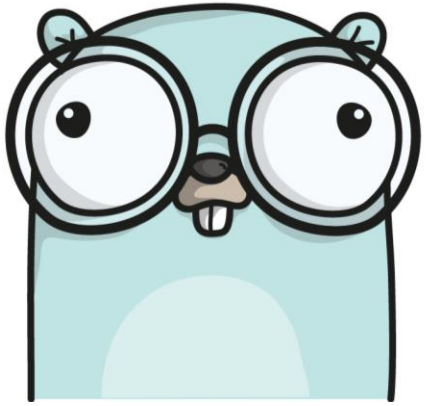
This Photo by Ashley Willis is licensed under CC BY-SA-NC.

Software development life cycle (SDLC)

Planning & Analysis

Design Product Architecture

Develop & Code



Test

Maintenance

This Photo by Ashley Willis is licensed under CC BY-SA-NC.

Integration testing

Ensure services work when integrating with other services

Considerations:

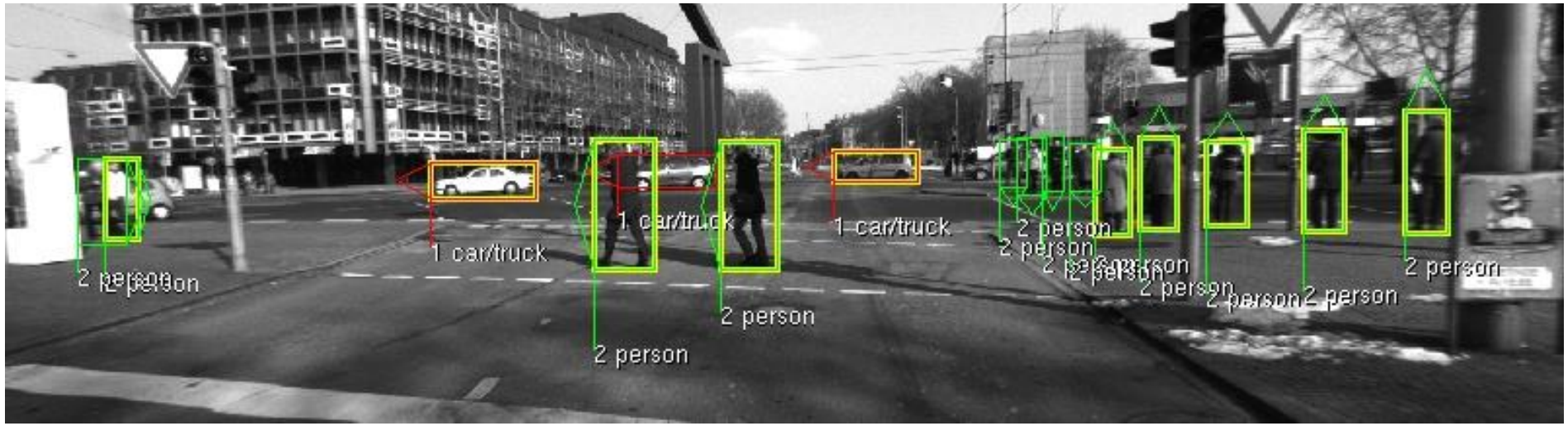
- Maintainable
- Speed
- Go Solution
- Release requirements

This Photo by Unknown author is licensed under CC BY-SA-NC.

Smart city solution use case

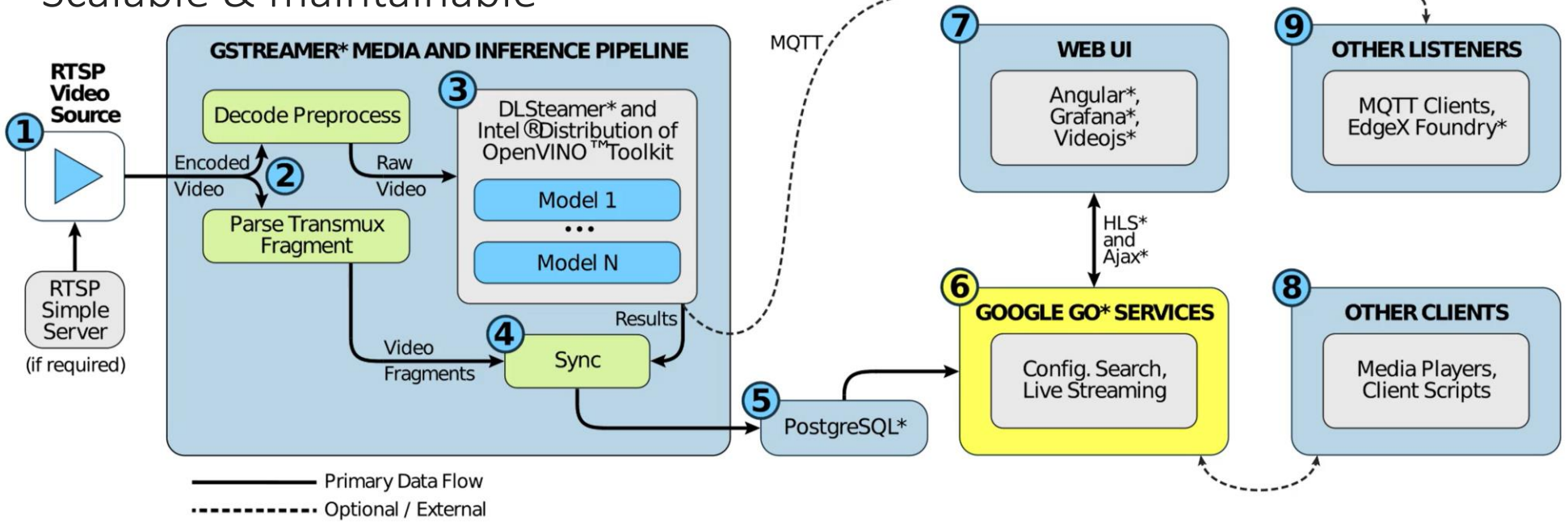
Situational awareness, property security, and management

Improve city safety



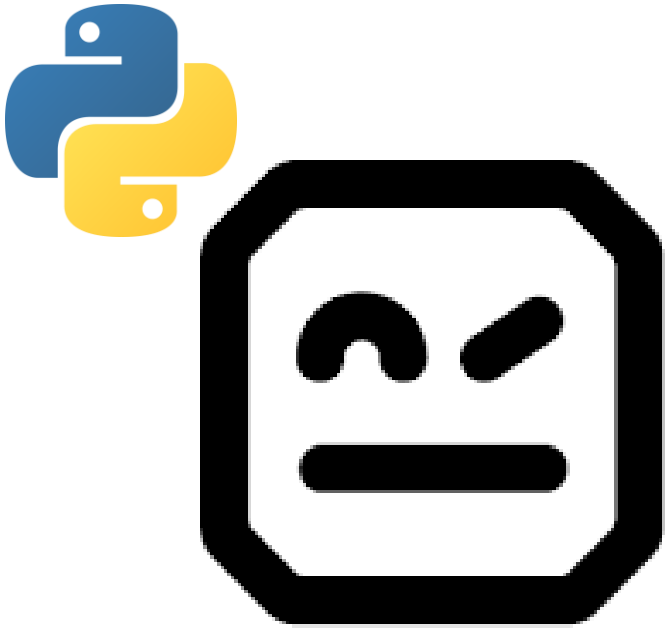
Smart city integration tests needs

1. Seamless test flow given current architecture
2. Ensure video data to work with before tests run
3. Scalable & maintainable



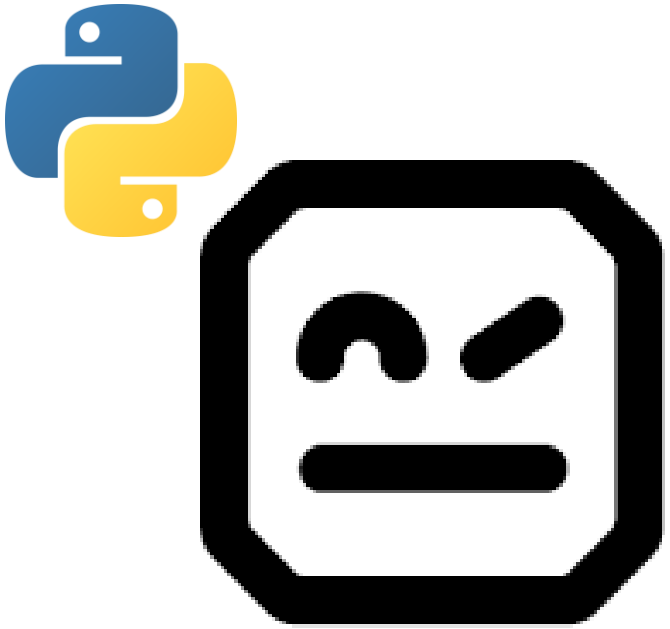
This Photo and this photo by Unknown author is licensed under CC BY-SA and CC By-NC.

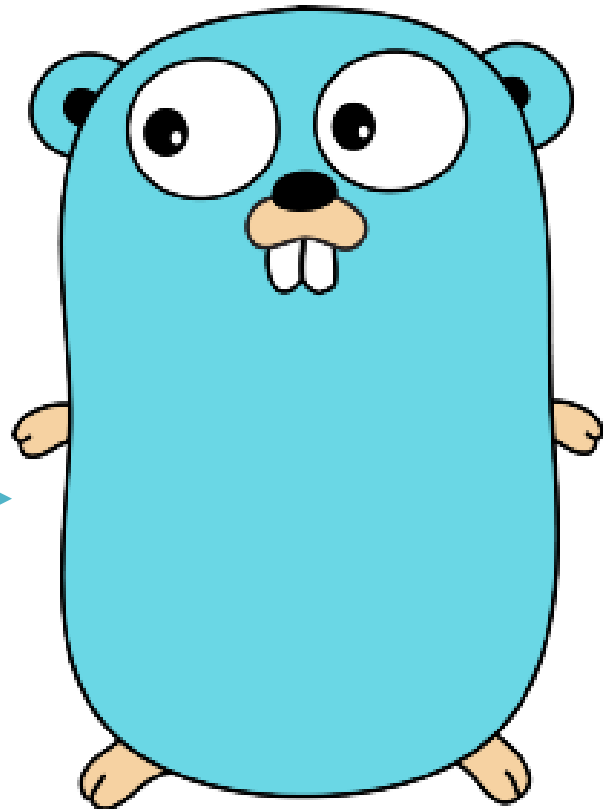
Go take a risk



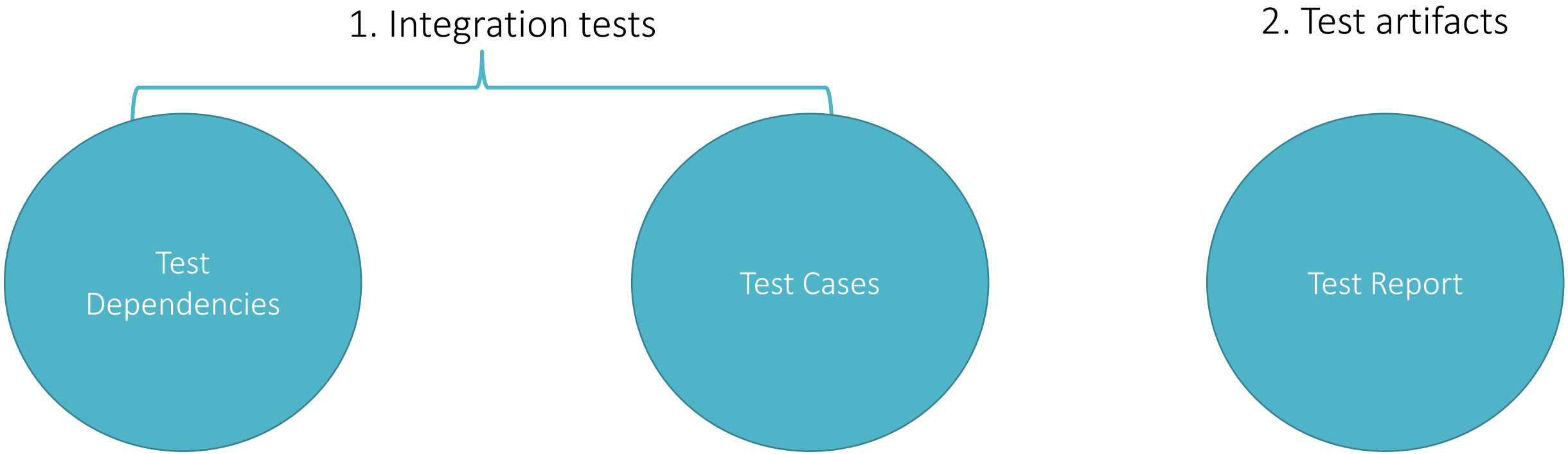
This Photo and this photo by Unknown author is licensed under CC BY-SA and CC By-NC.

Go take a risk





Integration Testing Solution Components



Test Dependencies

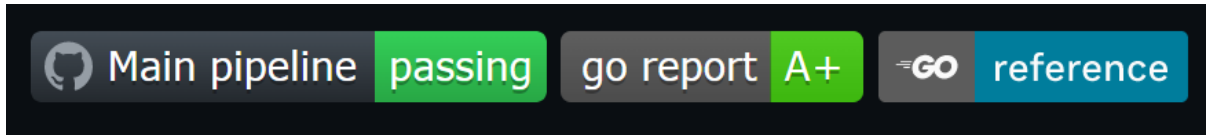
Testcontainers-go package

Stars: 1.5k

License: MIT

Contributors: ~80

Used by: Elastic, Telegraf, Intel, …



https://www.testcontainers.org/

Testcontainers-go package

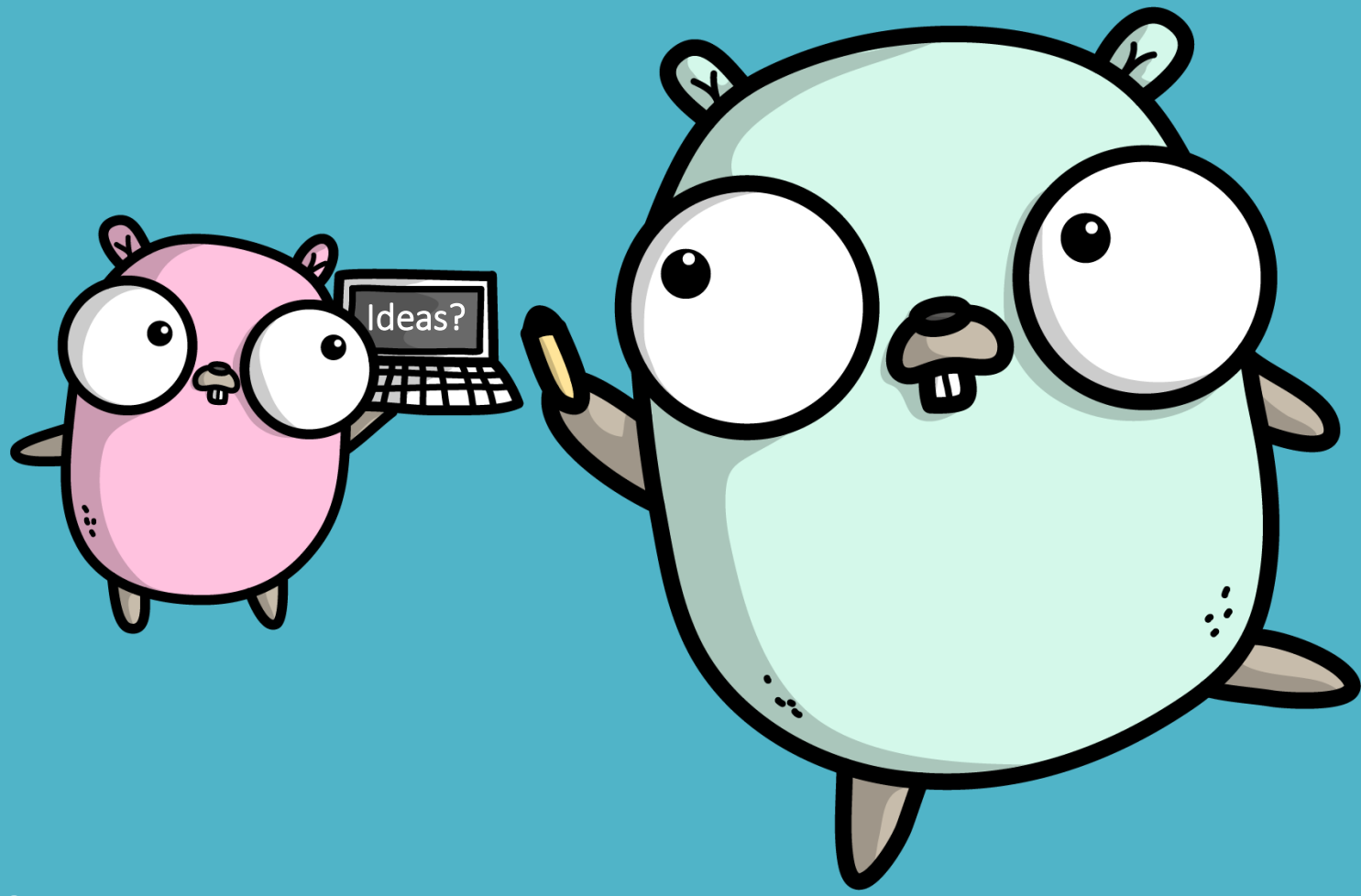
Github:

https://github.com/testcontainers/testcontainers-go
https://golang.testcontainers.org/

Features

- Garbage collection
- Build to test what you deploy
- Logs
- Container interactions
- Docker alternatives

This Photo by Ashely Willis is licensed under CC BY-SA-NC.



How to create a simple test service?

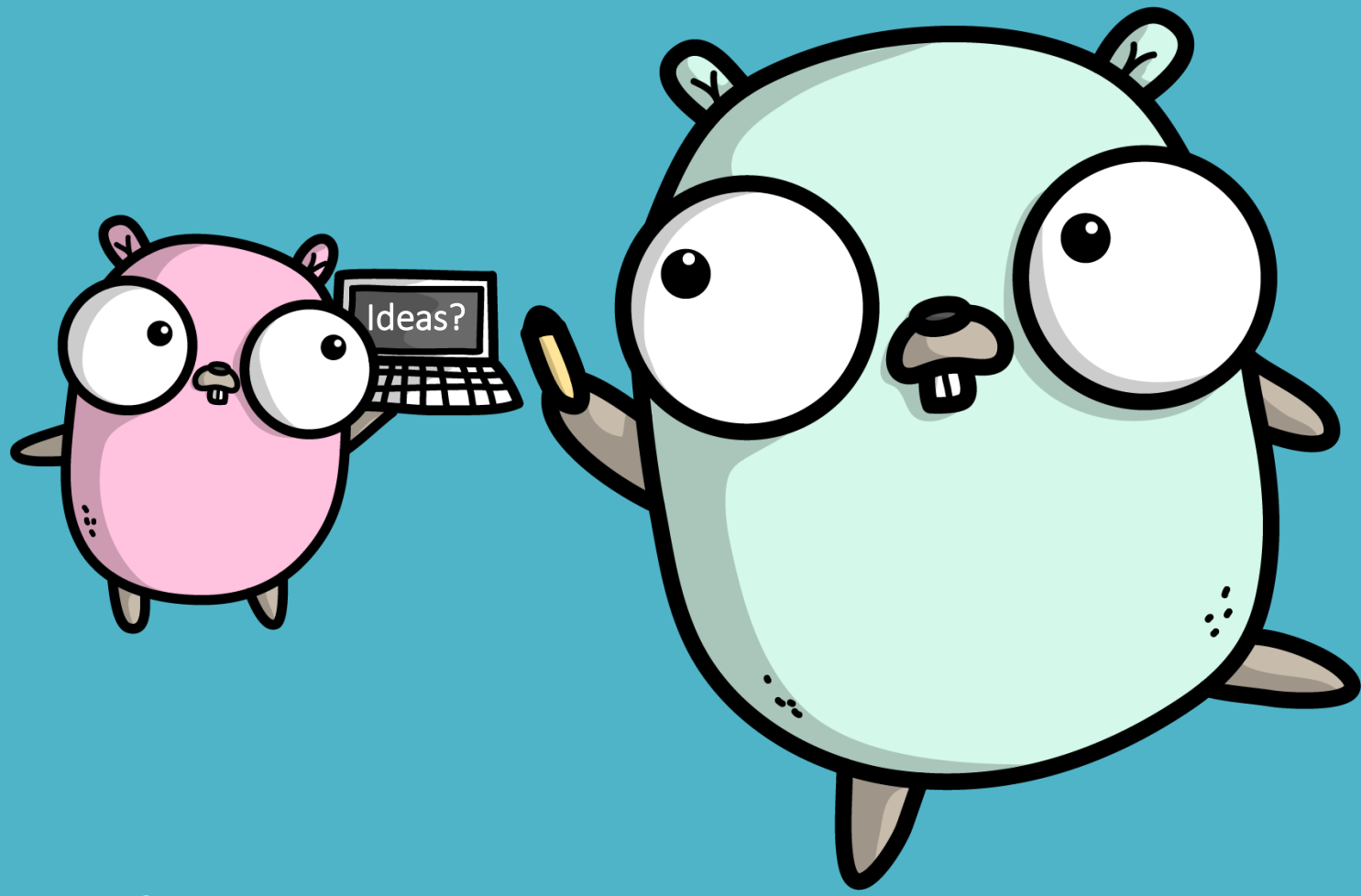
Define container

```
req := testcontainers.ContainerRequest{
        FromDockerfile: testcontainers.FromDockerfile{
                Context: "../.././rtspserver/",
        },
        Env:          env,
        ExposedPorts: []string{"8554/tcp"},
        WaitingFor:   wait.ForLog("[RTSP/TCP listener] opened on :8554"),
        Networks:     []string{td.NetworkName},
        BindMounts: map[string]string{
                mountFromVideos: mountToVideos,
                mountFromScript: mountToScript,
                mountFromConfig: mountToConfig,
        },
        Hostname: "test-rtsp-server",
}
```

Start container

```
rtspServerC, err := testcontainers.GenericContainer(ctx, testcontainers.GenericContainerRequest{
        ContainerRequest: req,
        Started:          true,
})
```

This Photo by Ashely Willis is licensed under CC BY-SA-NC.



What about projects with many services?

Docker compose

Contained many services...

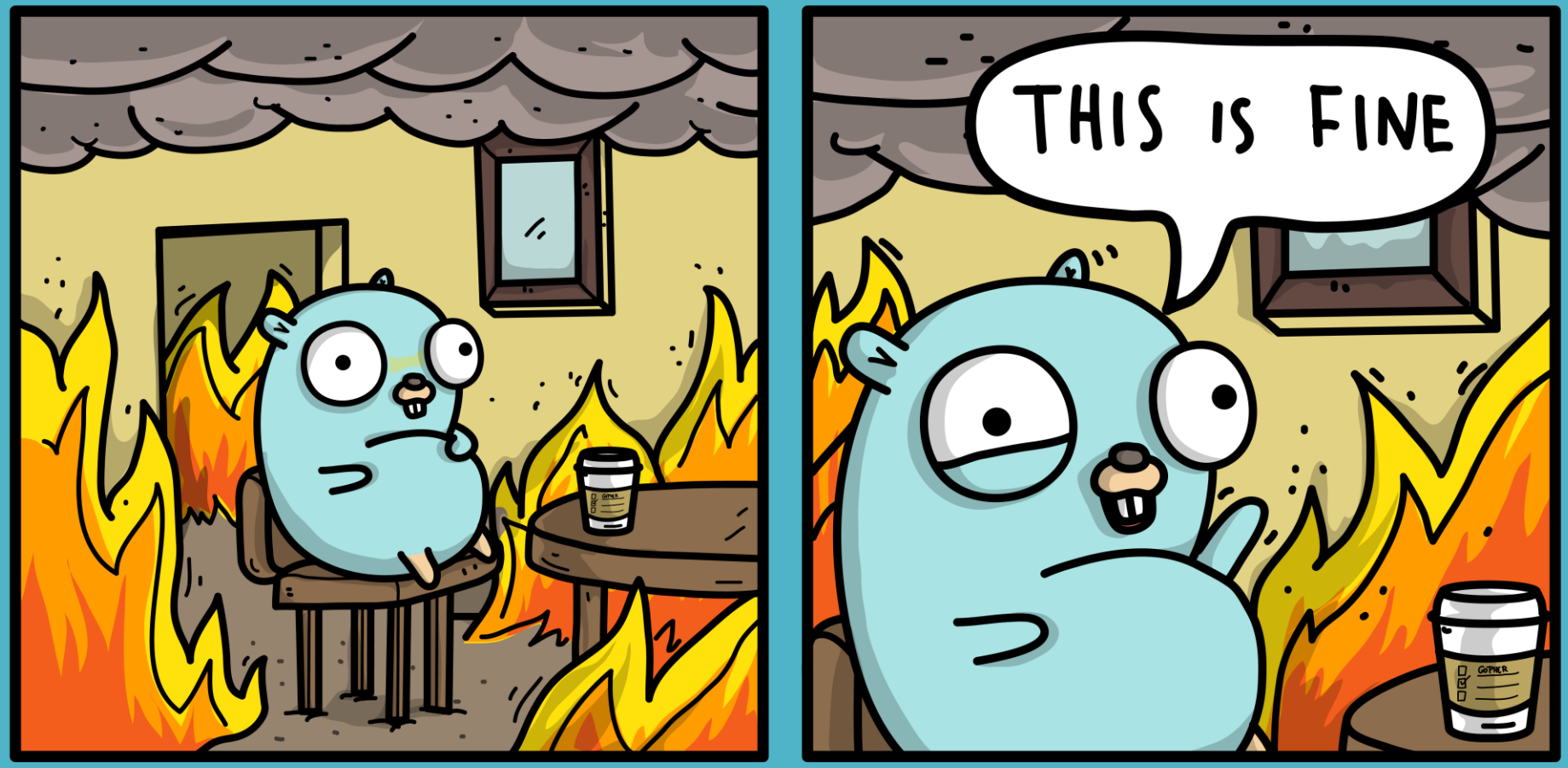
```
version: '3.7'

networks:
  rtsp-server-net:

services:
  rtsp-server:
    image: ${REPOSITORY}/rtsp-server
    networks:
      - rtsp-server-net
    volumes:
      - './rtsp-server/videos:/videos'
      - './rtsp-server/rtsp-config.yml:/rtsp-simple-server.yml'
```

Start service using docker compose

```
rtspCompose := testcontainers.NewLocalDockerCompose(rtspComposeFilePath, id)
rtspCompose.WithCommand([]string{"up", "-d"}).
        WithExposedService("rtsp-server_1", 8554, wait.ForLog("TCP listener opened on :8554").
                WithStartupTimeout(5*time.Second))
```

This Photo by Ashely Willis is licensed under CC BY-SA-NC.

Testcontainers-go Learnings



Wait Strategies

Difficult to know which to choose from the *many* options

Can customize

Pick the one that makes the most sense:

- Using WaitForLog at first, but not forever
- Smart City Solution camera data to propagate before continuing

```
func ForSQL(port nat.Port, driver string, url func(nat.Port) string) *waitForSql
type ExecStrategy
    func ForExec(cmd []string) *ExecStrategy
    func NewExecStrategy(cmd []string) *ExecStrategy
    func (ws ExecStrategy) WaitUntilReady(ctx context.Context, target StrategyTarget) error
    func (ws *ExecStrategy) WithExitCodeMatcher(exitCodeMatcher func(exitCode int) bool) *ExecStrategy
    func (ws *ExecStrategy) WithPollInterval(pollInterval time.Duration) *ExecStrategy
    func (ws *ExecStrategy) WithStartupTimeout(startupTimeout time.Duration) *ExecStrategy
type ExitStrategy
    func ForExit() *ExitStrategy
    func NewExitStrategy() *ExitStrategy
    func (ws *ExitStrategy) WaitUntilReady(ctx context.Context, target StrategyTarget) (err error)
    func (ws *ExitStrategy) WithExitTimeout(exitTimeout time.Duration) *ExitStrategy
    func (ws *ExitStrategy) WithPollInterval(pollInterval time.Duration) *ExitStrategy
type HTTPStrategy
    func ForHTTP(path string) *HTTPStrategy
    func NewHTTPStrategy(path string) *HTTPStrategy
    func (ws *HTTPStrategy) WaitUntilReady(ctx context.Context, target StrategyTarget) (err error)
    func (ws *HTTPStrategy) WithAllowInsecure(allowInsecure bool) *HTTPStrategy
    func (ws *HTTPStrategy) WithBody(reqdata io.Reader) *HTTPStrategy
    func (ws *HTTPStrategy) WithMethod(method string) *HTTPStrategy
    func (ws *HTTPStrategy) WithPollInterval(pollInterval time.Duration) *HTTPStrategy
    func (ws *HTTPStrategy) WithPort(port nat.Port) *HTTPStrategy
    func (ws *HTTPStrategy) WithResponseMatcher(matcher func(body io.Reader) bool) *HTTPStrategy
    func (ws *HTTPStrategy) WithStartupTimeout(startupTimeout time.Duration) *HTTPStrategy
    func (ws *HTTPStrategy) WithStatusCodeMatcher(statusCodeMatcher func(status int) bool) *HTTPStrategy
    func (ws *HTTPStrategy) WithTLS(useTLS bool, tlsconf ...*tls.Config) *HTTPStrategy
type HealthStrategy
    func ForHealthCheck() *HealthStrategy
    func NewHealthStrategy() *HealthStrategy
    func (ws *HealthStrategy) WaitUntilReady(ctx context.Context, target StrategyTarget) (err error)
    func (ws *HealthStrategy) WithPollInterval(pollInterval time.Duration) *HealthStrategy
    func (ws *HealthStrategy) WithStartupTimeout(startupTimeout time.Duration) *HealthStrategy
type HostPortStrategy
    func ForExposedPort() *HostPortStrategy
    func ForListeningPort(port nat.Port) *HostPortStrategy
```

wait.ForSQL Limitations v0.11.1

```
if _, err := db.ExecContext(ctx, "SELECT 1"); err != nil {
        continue
}
```

testcontainers-go Limitations *at the time*

Feature request: Enable specifying query for `wait.ForSQL`. #350

Closed **sicoyle** opened this issue on Aug 23, 2021 · 2 comments · Fixed by #451

sicoyle commented on Aug 23, 2021 — Contributor

It would be a nice feature if we could specify a query instead of being forced to use the default query of `SELECT 1` here. It is stated that this repo helps with `integration/smoke tests` in the `README`, and I think this would help enable that better. With allowing users to specify what query to run, we could then wait for specific test data to propagate properly in the database before running test cases using only one wait strategy. This could potentially save people from having to specify a bunch of wait strategies for each specific service too (pending set up, number of services, project, use case, etc)!

3

Our work around V1

```
// Note: WaitForSQLQuery() implements the wait.Strategy type.
//       This allows for customizations on the SQL query to be executed against the database
//       to ensure that segment data from the database is available to work with for the test requests/queries.
WithExposedService("postgres_1", NoPort, WaitForSQLQuery("5432", "pgx",
        func(port nat.Port) string {
                return fmt.Sprintf("postgres://[redacted]@localhost:%d/[redacted]?sslmode=disable", port.Int())
        }).
        // Note: This query was chosen with a LIMIT of 4 as that will sufficient data to work with for the tests.
        WithQuery(`SELECT camera_id, session_id FROM sessions WHERE camera_id = 1 LIMIT 4`).
        WithRowMatcher(func(rows *sqlx.Rows) bool {
                var i int
                for rows.Next() {
                        i++
                }
                logrus.WithField("row count", i).
                        Info("Got rows")
                return i >= 4
        }).WithTimeout(90*time.Second)).
```

Our work around V2

```
// Note: A NewHTTPStrategy() was added below
//             to ensure that segment data from the database is available to work with for the test requests/queries.
WithExposedService("reverse-proxy_1", 80, wait.ForAll(
        // Note: This wait strategy guarantees camera 1 data is in the database
        wait.NewHTTPStrategy("/video/camera/1/playlist-0-1.m3u8").
                WithPort("80/tcp").
                WithStatusCodeMatcher(func(status int) bool {
                        logrus.Infof("response code: %d", status)
                        return 200 >= status && status < 300
                }).
                WithResponseMatcher(func(body io.Reader) bool {
                        data, err := ioutil.ReadAll(body)
                        logrus.Infof("body: %s", string(data))
                        return err == nil && bytes.Contains(data, []byte("segment/1.ts"))
                }).
                WithPollInterval(time.Second).
                WithMethod(http.MethodGet),
```

wait.ForSQL ~~limitations~~ improvements

```
if _, err := db.ExecContext(ctx, "SELECT 1"); err != nil {
        continue
}

if _, err := db.ExecContext(ctx, w.query); err != nil {
        continue
}
```

Test Cases

Define what to test

Camera management

Video service playlists

Inference service searchability

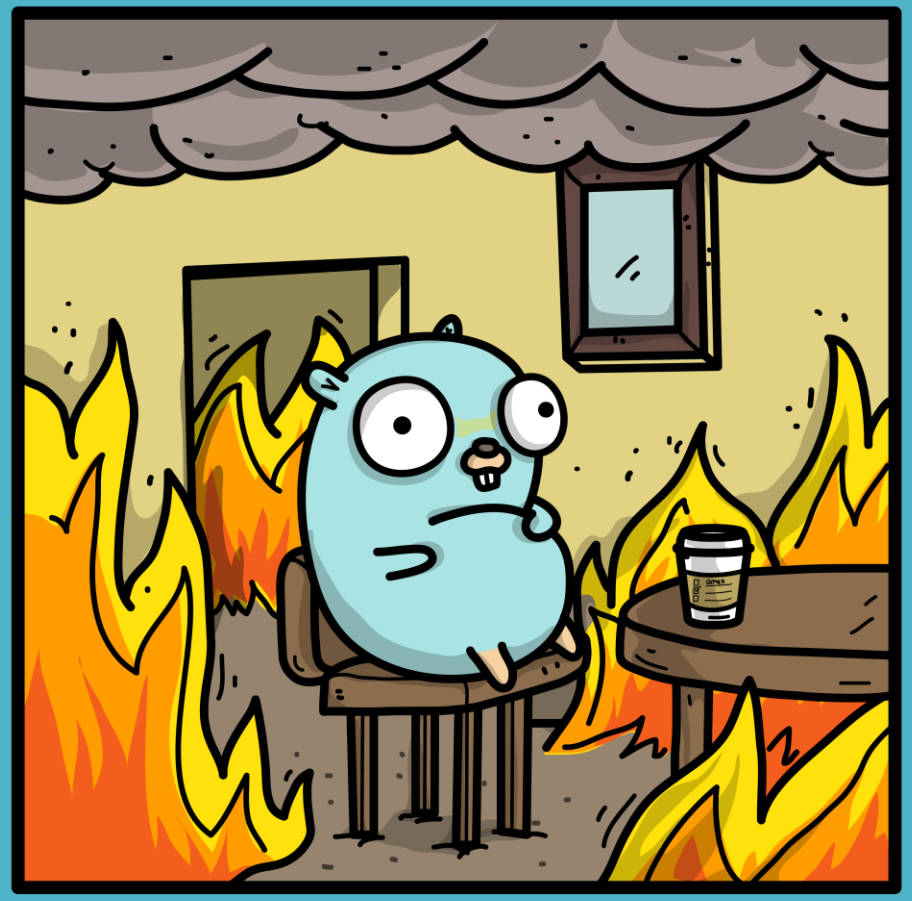
Media pipeline

Stopped camera feed

...

This Photo by Ashely Willis is licensed under CC BY-SA-NC.

Test Case Learnings



Start adding scenarios

Got repetitive quickly checking all the services

```
{
        scope:          "Camera Config",
        name:           "get cameras",
        method:         http.MethodGet,
        url:            "/api/v1/cameras",
        expectedCamera: []Camera{c},
        statusCode:     http.StatusOK,
},
{
        scope:      "Camera Config",
        name:       "update default camera 1",
        method:     http.MethodPut,
        payload:    updateCameraPayload,
        url:        "/api/v1/cameras/1",
        statusCode: http.StatusOK,
},
{
        scope:      "Camera Config",
        name:       "update invalid camera ID",
        method:     http.MethodPut,
        payload:    updateCameraPayload,
        url:        "/api/v1/cameras/1111",
        statusCode: http.StatusInternalServerError,
},
```

Increasing scope

```
testCases := []struct {
        scope          string
        name           string
        method         string
        payload        []byte
        url            string
        expectedBody   []Camera
        statusCode     int
}{
```

```
testCases := []struct {
        scope          string
        name           string
        method         string
        payload        []byte
        url            string
        expectedCamera []Camera
        query          string
        queryWant      []responseData
        queryNotWant   []responseData
        expectedBody   string
        statusCode     int
}{
```

Template things out

```
type SQLQueryTestCase struct {
        TestCase
        query     string
        queryArgs []interface{}
        queryWant interface{}
}
```

```
type HTTPTestCase struct {
        TestCase
        method              string
        payload             []byte
        url                 string
        expectedBody        string
        expectedBodyRegexp  *regexp.Regexp
        statusCode          int
        expectData          interface{}
        bodyValidator       BodyValidator
        headers             http.Header
}
```

Using our custom package

```
client.RunAll([]ExecutableTestCase{
        NewTestCase("update added camera").
                PUT(fmt.Sprintf("/api/v1/cameras/%d", testCamera.ID), mustMarshalJSON(t, updatedCamera)).
                ExpectStatusCode(http.StatusNoContent),

        NewTestCase("get updated cameras").
                GET("/api/v1/cameras").
                ExpectData(camResponse{Data: append(defaultCameras, updatedCamera)}),

        NewTestCase("delete added test camera").
                DELETE(fmt.Sprintf("/api/v1/cameras/%d", testCamera.ID)).
                ExpectStatusCode(http.StatusNoContent),

        NewTestCase("verify deleted test camera").
                GET("/api/v1/cameras").
                ExpectData(camResponse{Data: defaultCameras}),
})
```

Alternatives moving forward

HTTPS://GITHUB.COM/STEINFLETCHER/APITEST

```
func TestApi(t *testing.T) {
        apitest.New().
                Handler(handler).
                Get("/user/1234").
                Expect(t).
                Body(`{"id": "1234", "name": "Tate"}`).
                Status(http.StatusOK).
                End()
}
```

HTTPS://GITHUB.COM/GAVV/HTTPEXPECT

```
func TestFruits(t *testing.T) {
        // create http.Handler
        handler := FruitsHandler()

        // run server using httptest
        server := httptest.NewServer(handler)
        defer server.Close()

        // create httpexpect instance
        e := httpexpect.New(t, server.URL)

        // is it working?
        e.GET("/fruits").
                Expect().
                Status(http.StatusOK).JSON().Array().Empty()
}
```

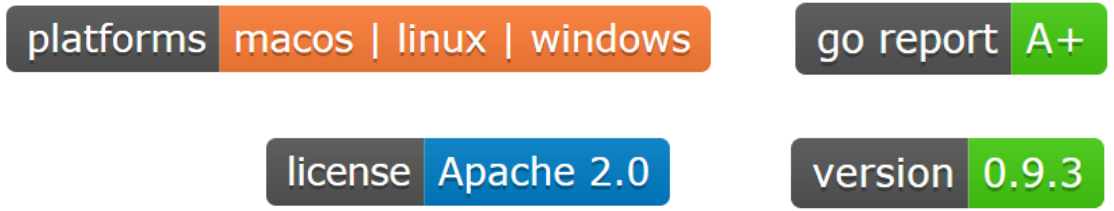
Test Report

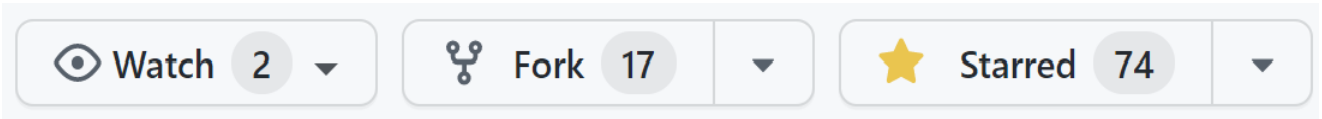
Go-test-report package

Github:

https://github.com/vakenbolt/go-test-report

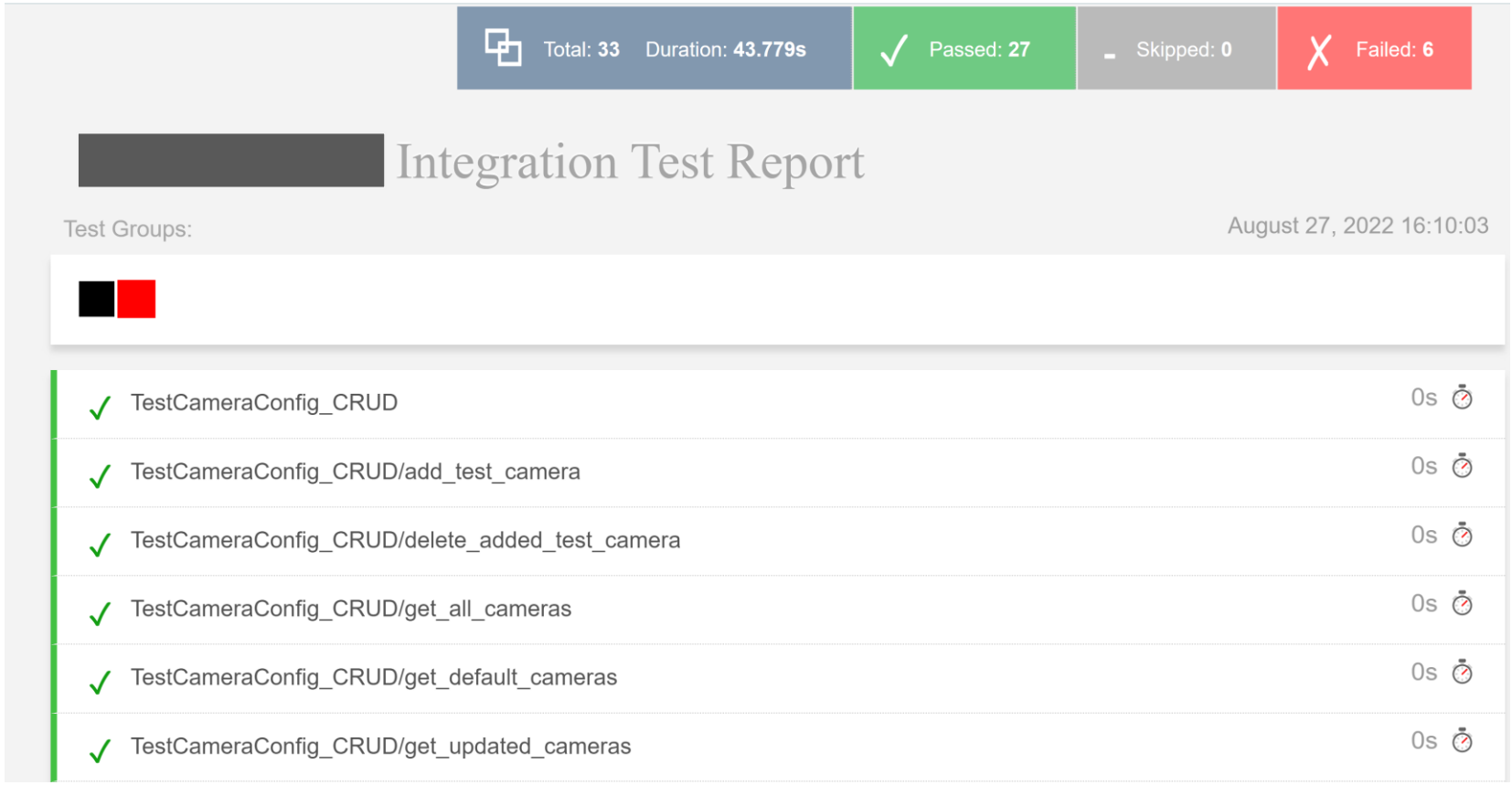
Project requirement for us





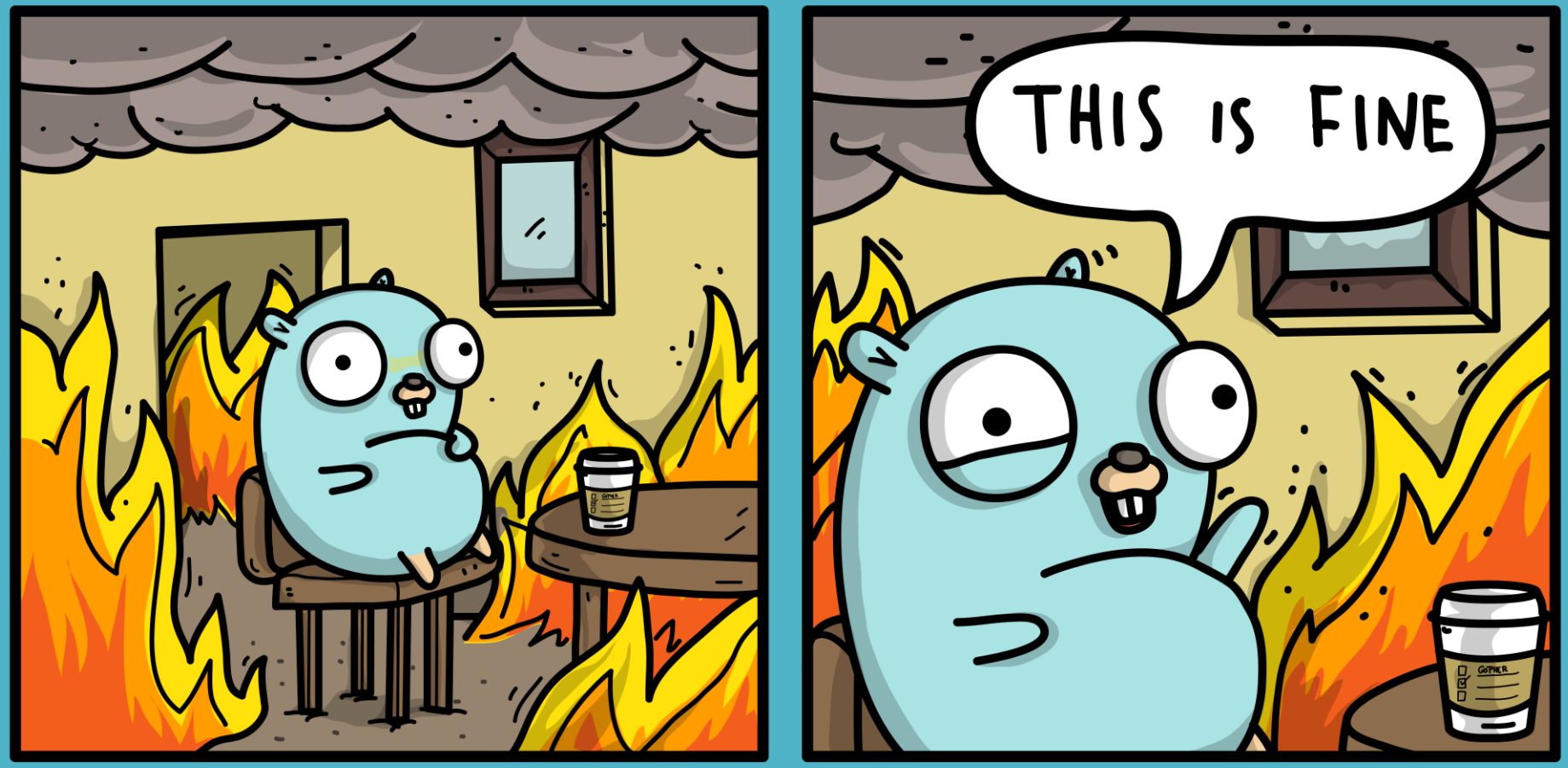
Go-test-report package

Requires json as input.



This Photo by Ashely Willis is licensed under CC BY-SA-NC.

Test Report Learnings



Validation team norms

Go test framework "ok“ wrapper

Passing tests on test report

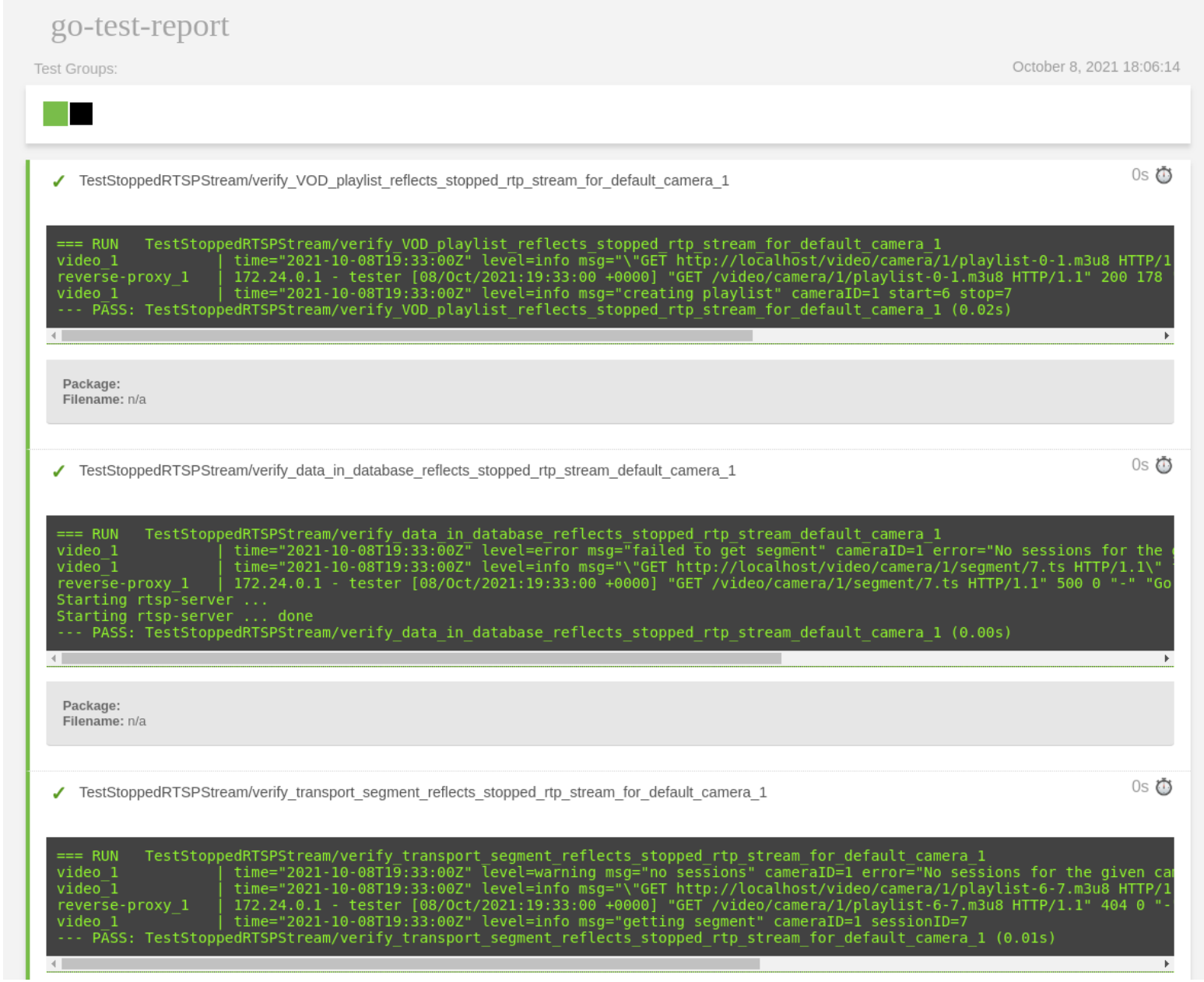
Validation engineer double checks

1. Check tests succeed


```
Starting rtsp-server ... done
--- PASS: TestStoppedRTSPStream (2.99s)
    --- PASS: TestStoppedRTSPStream/check_valid_rtsp_stream (0.01s)
    --- PASS: TestStoppedRTSPStream/verify_VOD_playlist_for_default_camera_1 (0.01s)
    --- PASS: TestStoppedRTSPStream/verify_VOD_playlist_reflects_stopped_rtp_stream_for_default_camera_1 (0.01s)
    --- PASS: TestStoppedRTSPStream/verify_transport_segment_reflects_stopped_rtp_stream_for_default_camera_1 (0.00s)
    --- PASS: TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1 (0.00s)
```

2. Check report

```
go test |
  go tool test2json |
  go-test-report
```



3. Validation engineer checks


```
Starting rtsp-server ... done
--- PASS: TestStoppedRTSPStream (2.99s)
    --- PASS: TestStoppedRTSPStream/check_valid_rtsp_stream (0.01s)
    --- PASS: TestStoppedRTSPStream/verify_VOD_playlist_for_default_camera_1 (0.01s)
    --- PASS: TestStoppedRTSPStream/verify_VOD_playlist_reflects_stopped_rtp_stream_for_default_camera_1 (0.01s)
    --- PASS: TestStoppedRTSPStream/verify_transport_segment_reflects_stopped_rtp_stream_for_default_camera_1 (0.00s)
    --- PASS: TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1 (0.00s)
```

This Photo by Ashley Willis is licensed under CC BY-SA-NC.

3. Validation engineer checks

✗ TestStoppedRTSPStream

```
=== RUN   TestStoppedRTSPStream
```

Package:
Filename: integration_test.go **Line:** 269 **Col:** 1

✗	TestStoppedRTSPStream/check_valid_rtsp_stream	0s
✗	TestStoppedRTSPStream/verify_VOD_playlist_for_default_camera_1	0s
✗	TestStoppedRTSPStream/verify_VOD_playlist_reflects_stopped_rtp_stream_for_default_camera_1	0s
✗	TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1	0s
✗	TestStoppedRTSPStream/verify_transport_segment_reflects_stopped_rtp_stream_for_default_camera_1	0s

This Photo by Ashley Willis is licensed under CC BY-SA-NC.

3. Validation engineer checks

✗ TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1 0s

```
verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1
9-06T16:33:58Z" level=error msg="failed to get segment" cameraID=1 error="No sessions for the given camera ID" se
9-06T16:33:58Z" level=info msg="\"GET http://localhost/video/camera/1/segment/7.ts HTTP/1.1\" from 172.27.0.2:609
 tester [06/Oct/2022:16:33:58 +0000] "GET /video/camera/1/segment/7.ts HTTP/1.1" 500 0 "-" "Go-http-client/1.1"

 PASS: TestStoppedRTSPStream (3.48s)--- PASS: TestStoppedRTSPStream/check_valid_rtsp_stream (0.03s)--- PASS: Test
```

Package:
Filename: n/a

This Photo by Ashley Willis is licensed under CC BY-SA-NC.

What to do?

Don’t want this merged into main

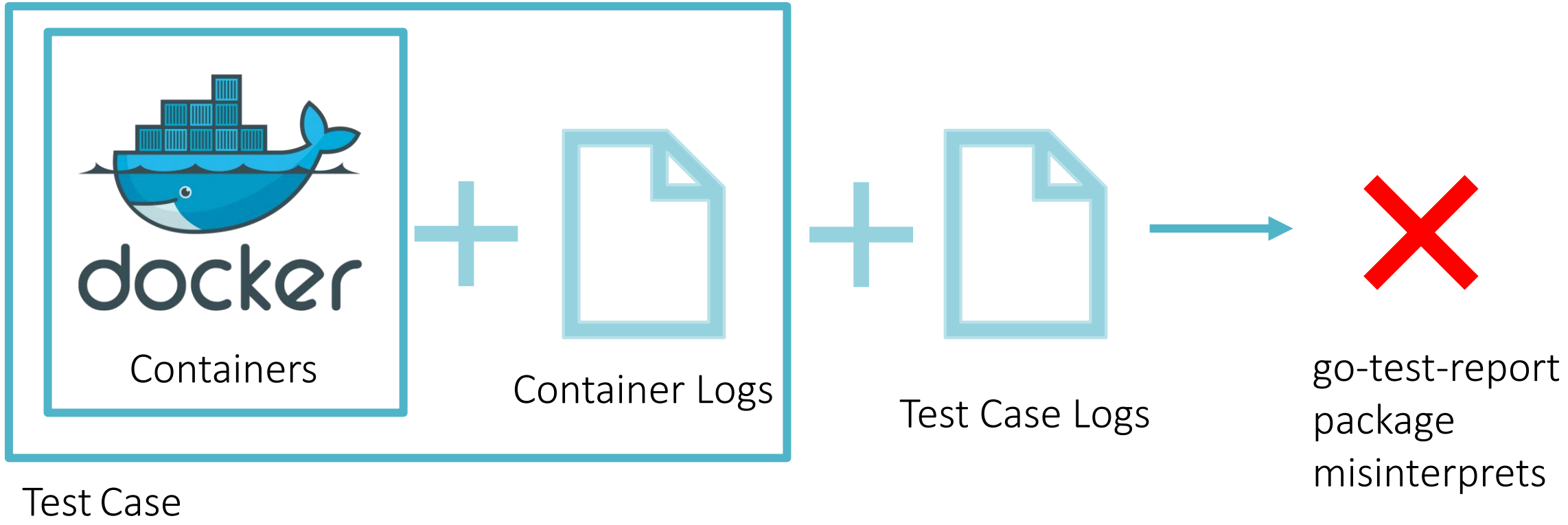
Failing inconsistently

Couldn’t prove tests pass

Investigate further!

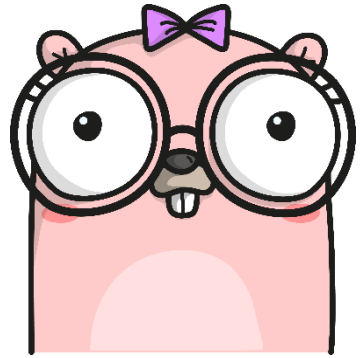
This Photo by Unknown author is licensed under CC.

Initial approach



This Photo by Ashley Willis is licensed under CC BY-SA-NC.

Spurious logs

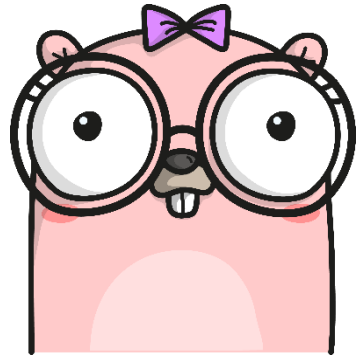


Developer machines:

{"Action":"output","Test":"TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1","Output":"Starting rtsp-server ... done\n"}
{"Action":"output","Test":"TestStoppedRTSPStream","Output":"--- PASS: TestStoppedRTSPStream (3.12s)\n"}

This Photo by Ashley Willis is licensed under CC BY-SA-NC.

Spurious logs



Developer machines:

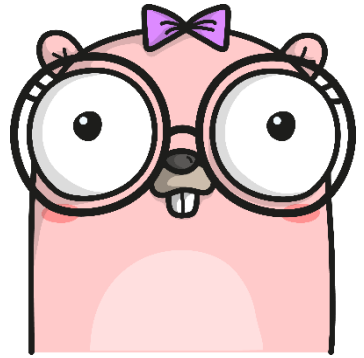
{"Action":"output","Test":"TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1","Output":"Starting rtsp-server ... done\n"}
{"Action":"output","Test":"TestStoppedRTSPStream","Output":"--- PASS: TestStoppedRTSPStream (3.12s)\n"}

Validation engineer machine:

{"Action":"output","Test":"TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1","Output":"\rStarting rtsp-server ... done\r--- PASS: TestStoppedRTSPStream (3.95s)\n"}

This Photo by Ashley Willis is licensed under CC BY-SA-NC.

Spurious logs



Developer machines:

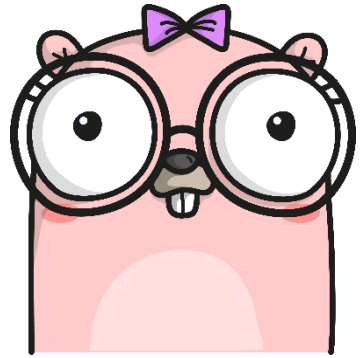
{"Action":"output","Test":"TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1","Output":"Starting rtsp-server ... done\n"}
{"Action":"output","Test":"TestStoppedRTSPStream","Output":"--- PASS: TestStoppedRTSPStream (3.12s)\n"}

Validation engineer machine:

{"Action":"output","Test":"TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1","Output":"\rStarting rtsp-server ... done\r--- PASS: TestStoppedRTSPStream (3.95s)\n"}

This Photo by Ashley Willis is licensed under CC BY-SA-NC.

Spurious logs



Developer machines:

{"Action":"output","Test":"TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1","Output":"Starting rtsp-server ... done\n"}
{"Action":"output","Test":"TestStoppedRTSPStream","Output":"--- PASS: TestStoppedRTSPStream (3.12s)\n"}

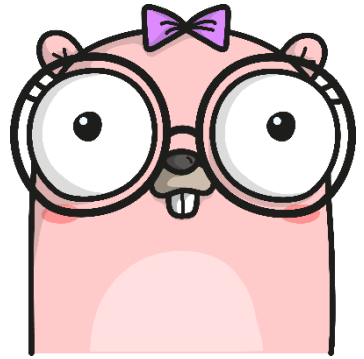
{"Action":"pass","Test":"TestStoppedRTSPStream"}

Validation engineer machine:

{"Action":"output","Test":"TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1","Output":"\rStarting rtsp-server ... done\r--- PASS: TestStoppedRTSPStream (3.95s)\n"}

This Photo by Ashley Willis is licensed under CC BY-SA-NC.

Spurious logs continued

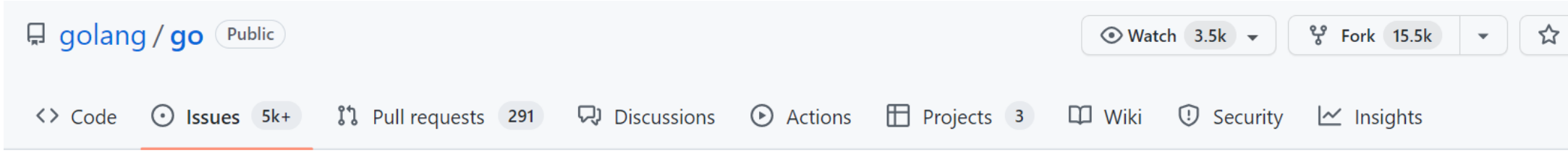


✗ TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1 0s

```
verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1
0-06T16:33:58Z" level=error msg="failed to get segment" cameraID=1 error="No sessions for the given camera ID" se
0-06T16:33:58Z" level=info msg="\"GET http://localhost/video/camera/1/segment/7.ts HTTP/1.1\" from 172.27.0.2:609
 tester [06/Oct/2022:16:33:58 +0000] "GET /video/camera/1/segment/7.ts HTTP/1.1" 500 0 "-" "Go-http-client/1.1"

PASS: TestStoppedRTSPStream (3.48s)--- PASS: TestStoppedRTSPStream/check_valid_rtsp_stream (0.03s)--- PASS: Test
```

Package:
Filename: n/a

Interesting find #1



testing: stdout output with no newline produces confusing/corrupt -v output #26325

Open **neild** opened this issue on Jul 10, 2018 · 12 comments · May be fixed by #53506

"We have some **tools that parse test output** which **get confused** in this situation..."

```
$ go test .
ok      testout 0.016s
$ go test . -v
=== RUN   TestOut
Test--- PASS: TestOut (0.00s)
PASS
ok      testout 0.017s
```

Interesting find #2

golang / go Public

Watch 3.5k | Fork 15.5k | Star 104k

Code | Issues 5k+ | Pull requests 287 | Discussions | Actions | Projects 3 | Wiki | Security | Insights

cmd/go: test2json does not recognize test status if output did not end with new line #47032

Closed **suzmue** opened this issue on Jul 2, 2021 · 2 comments

Interesting find #2

jayconrod commented on Jul 6, 2021 Contributor

Closing as a duplicate of #26325.

Unfortunately, we still don't have a solution for that. The testing package writes structured line-based output to stdout, and tests can write stuff there too that interferes.

Altered logs

dominikh commented on Aug 13, 2013

Comment 20:

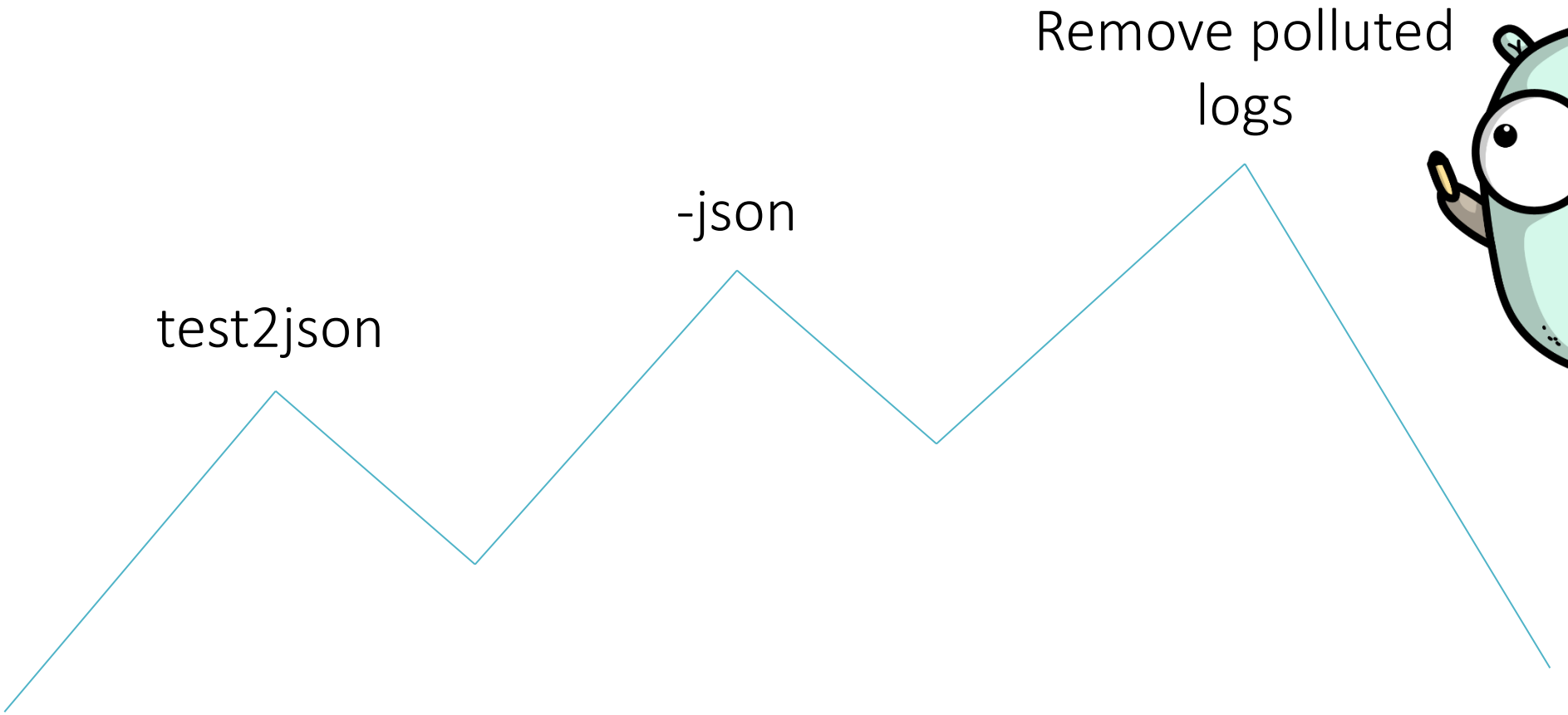
```
One problem I see with the "parsing `go test` output" solution is that there are 3rd
party `go test` enhancements, which tend to change the output format.
For example, github.com/stretchr/testify/assert uses \r to overwrite t.Errorf's
"file:line: message" output with its own format and, more importantly, caller
information.
Such uses would break the parser while they'd still work fine if the json generation
was integrated into `testing`.
```

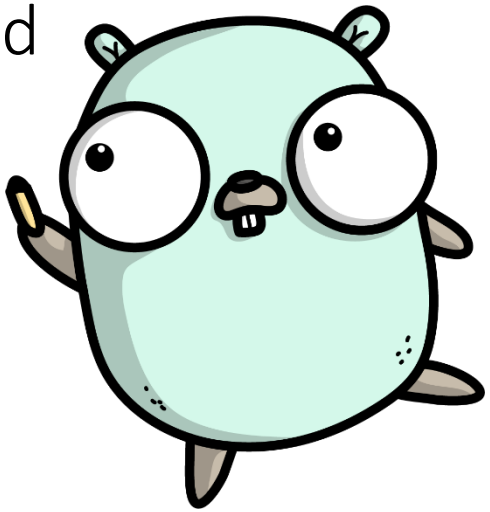
The issue


Go test json
Invalid/missing characters
Incorrect go-test-report output

This Photo by Ashley Willis is licensed under CC BY-SA-NC.

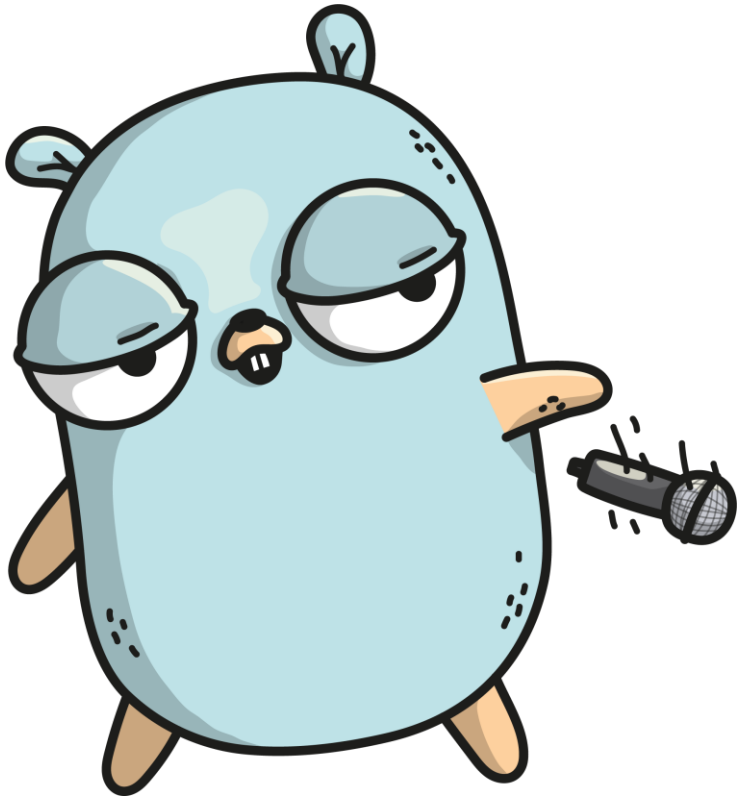
Our workaround attempts





This Photo by Ashley Willis is licensed under CC BY-SA-NC.

Success!



go-test-report

Test Groups:

October 8, 2021 18:06:14

✓ TestStoppedRTSPStream/verify_VOD_playlist_reflects_stopped_rtp_stream_for_default_camera_1 0s

```
=== RUN   TestStoppedRTSPStream/verify_VOD_playlist_reflects_stopped_rtp_stream_for_default_camera_1
video_1           | time="2021-10-08T19:33:00Z" level=info msg="\"GET http://localhost/video/camera/1/playlist-0-1.m3u8 HTTP/1
reverse-proxy_1   | 172.24.0.1 - tester [08/Oct/2021:19:33:00 +0000] "GET /video/camera/1/playlist-0-1.m3u8 HTTP/1.1" 200 178
video_1           | time="2021-10-08T19:33:00Z" level=info msg="creating playlist" cameraID=1 start=6 stop=7
--- PASS: TestStoppedRTSPStream/verify_VOD_playlist_reflects_stopped_rtp_stream_for_default_camera_1 (0.02s)
```

Package:
Filename: n/a

✓ TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1 0s

```
=== RUN   TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1
video_1           | time="2021-10-08T19:33:00Z" level=error msg="failed to get segment" cameraID=1 error="No sessions for the
video_1           | time="2021-10-08T19:33:00Z" level=info msg="\"GET http://localhost/video/camera/1/segment/7.ts HTTP/1.1\"
reverse-proxy_1   | 172.24.0.1 - tester [08/Oct/2021:19:33:00 +0000] "GET /video/camera/1/segment/7.ts HTTP/1.1" 500 0 "-" "Go
Starting rtsp-server ...
Starting rtsp-server ... done
--- PASS: TestStoppedRTSPStream/verify_data_in_database_reflects_stopped_rtp_stream_default_camera_1 (0.00s)
```

Package:
Filename: n/a

✓ TestStoppedRTSPStream/verify_transport_segment_reflects_stopped_rtp_stream_for_default_camera_1 0s

```
=== RUN   TestStoppedRTSPStream/verify_transport_segment_reflects_stopped_rtp_stream_for_default_camera_1
video_1           | time="2021-10-08T19:33:00Z" level=warning msg="no sessions" cameraID=1 error="No sessions for the given ca
video_1           | time="2021-10-08T19:33:00Z" level=info msg="\"GET http://localhost/video/camera/1/playlist-6-7.m3u8 HTTP/1
reverse-proxy_1   | 172.24.0.1 - tester [08/Oct/2021:19:33:00 +0000] "GET /video/camera/1/playlist-6-7.m3u8 HTTP/1.1" 404 0 "-
video_1           | time="2021-10-08T19:33:00Z" level=info msg="getting segment" cameraID=1 sessionID=7
--- PASS: TestStoppedRTSPStream/verify_transport_segment_reflects_stopped_rtp_stream_for_default_camera_1 (0.01s)
```

Conclusion

Integration testing solution benefits

Maintainable
Scalable
Test what you deploy

Got the job done

User friendly
Scalable
Project requirement

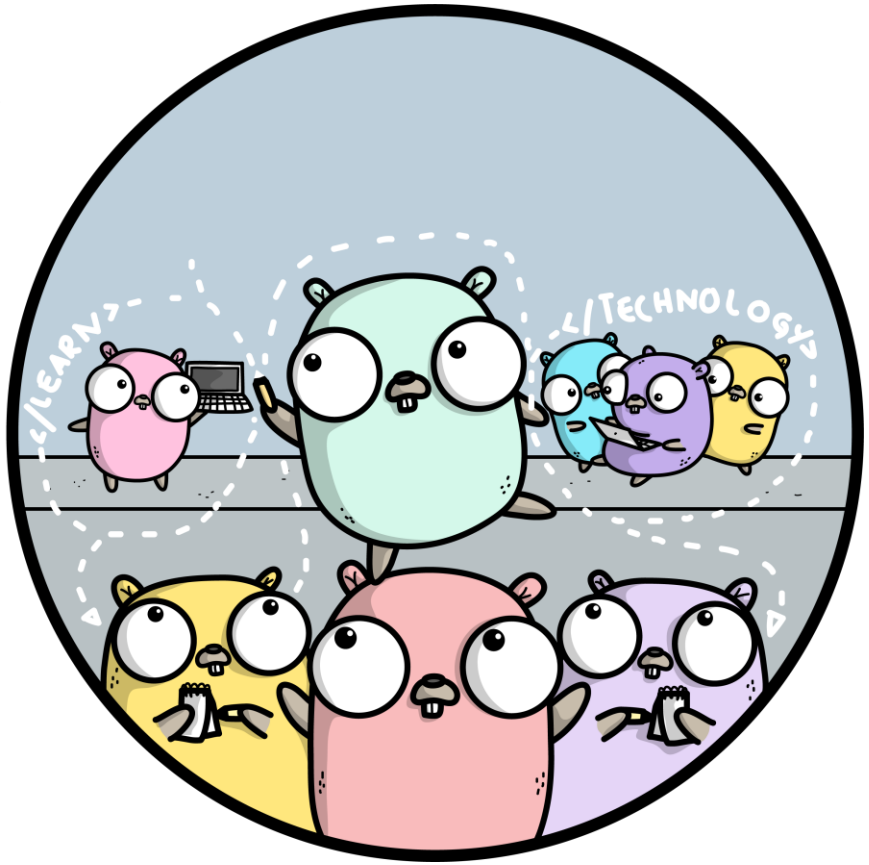
This Photo by Ashely Willis is licensed under CC BY-SA-NC.

Risks do pay off sometimes :)

More Go integration testing solutions within our org

Internal release tooling to support Go in future

Always improving



This Photo by Ashely Willis is licensed under CC BY-SA-NC.

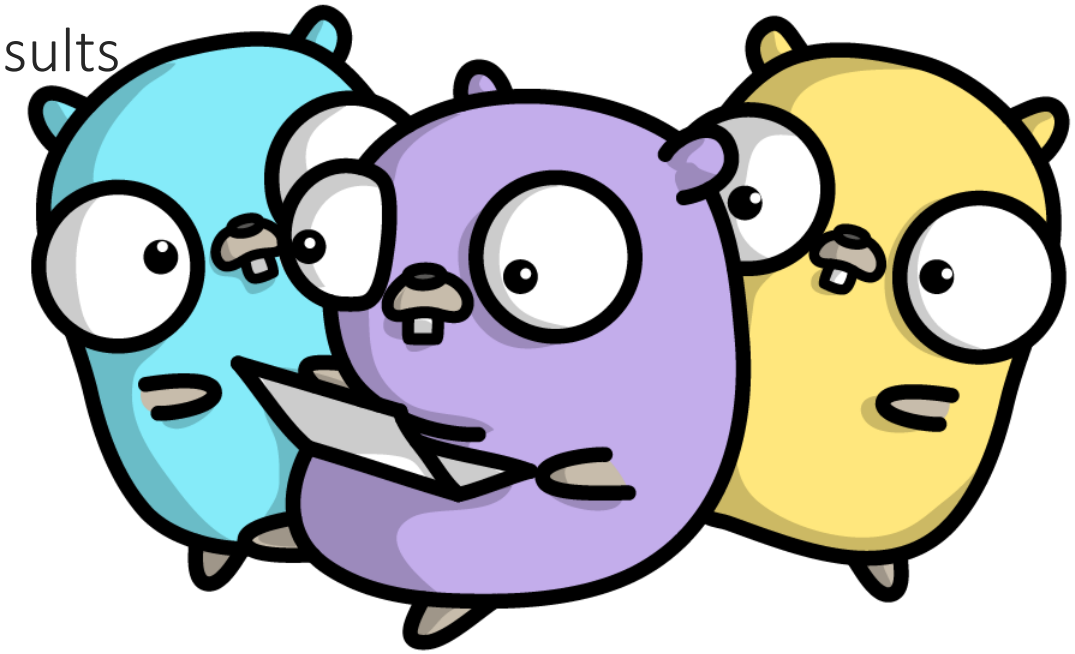
Takeaways

Useful Go integration testing solution insights

Learn from our mistakes (point of these talks!)

Be mindful when using parsers for Go test results

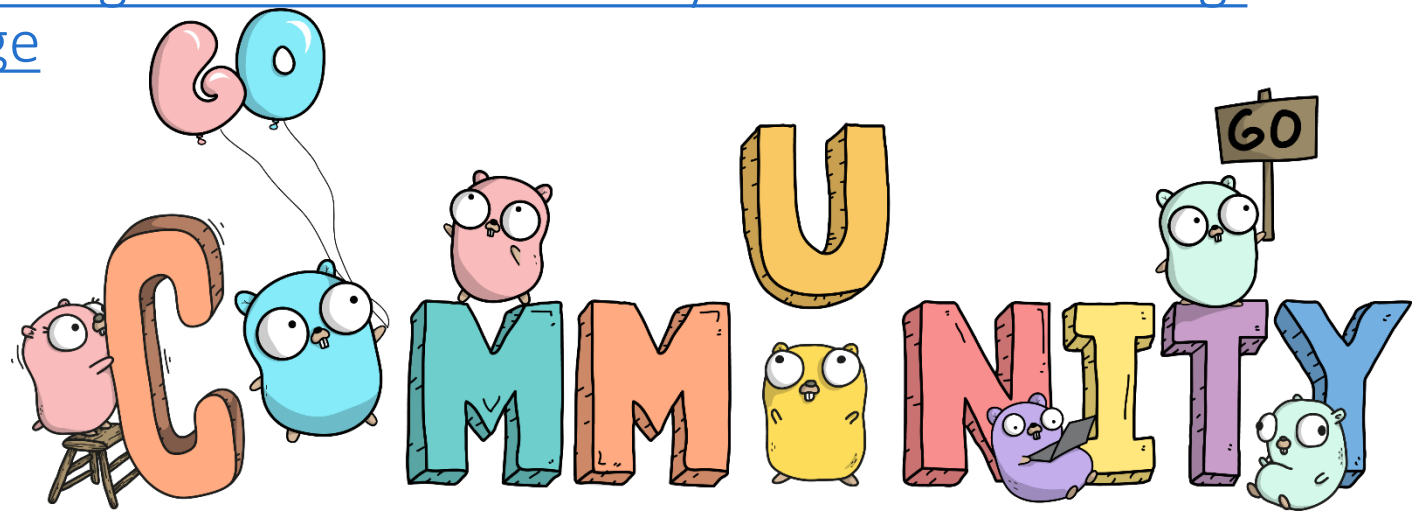
Be fearless with your journey always



This Photo by Ashely Willis is licensed under CC BY-SA-NC.

Want to know more?

1. Open Source Summit presentation - https://youtu.be/l2ttxMVFM3s
2. Slides - https://github.com/sicoyle/cfp or https://samcoyle.me/
3. IEEE paper - https://smartcities.ieee.org/newsletter/september-2022/computer-vision-to-secure-your-surroundings-with-ai-ml-smart-city-solution-built-using-open-source-tools-at-the-edge



Thank you!

Samantha Coyle

Software Engineer

http://samcoyle.me/

Github – sicoyle

Twitter - @thesamcoyle

A Journey Through Integration Testing with Go

What Could Go Wrong?

With Samantha Coyle