Metrics as a First-Class citizen in the E2E Testing landscape

Jéssica Lins & Matej Gera

Jéssica Lins

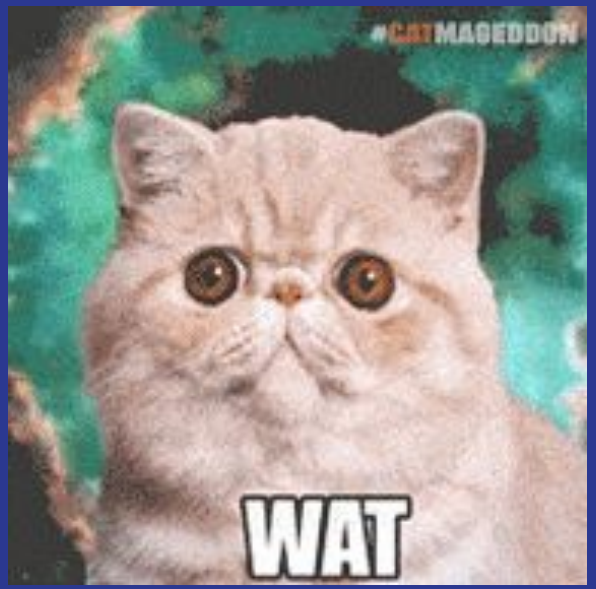
- Software Engineer @ Red Hat
- Contributor/approver for Portuguese content @ CNCF glossary
- Maintainer @ Observatorium
- Currently working with Go and interested in distributed systems & observability

Matej Gera

- Software Engineer @ Red Hat
- Contributor & triage @ Thanos
- Maintainer @ Observatorium

Content

- TL;DR Metrics
- Metrics and E2E testing as a concept
  - vs. conventional E2E testing
  - Patterns and added value
  - Extending to similar types of tests
- Practical application - efficientgo/e2e framework
  - Real world project usages
- Demo time!


#CATMAGEDDON
WAT

???

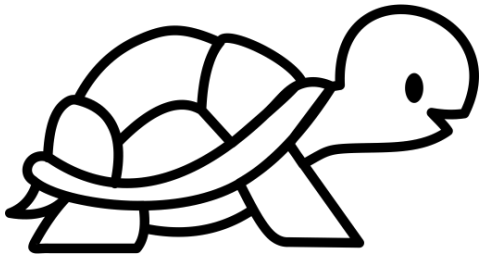
Why metrics?

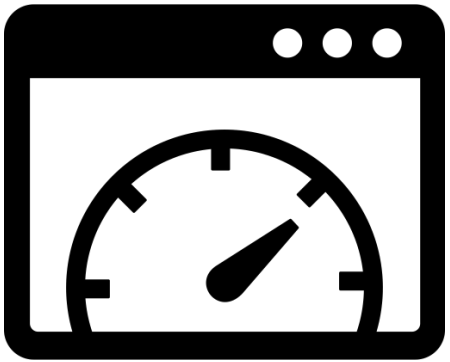
Why metrics?

Get performance insights about your application

Why metrics?

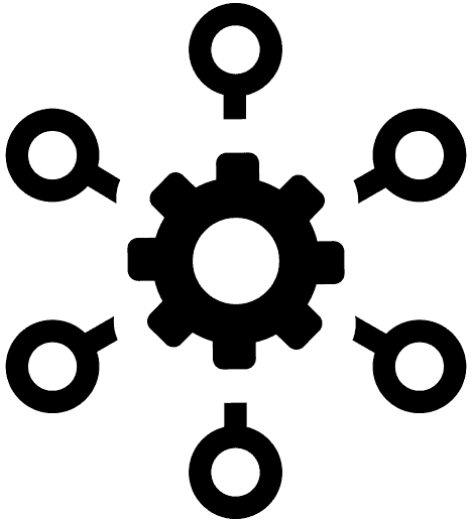
How much load does your application get?
Is it throwing errors? How many?
How slow is it?





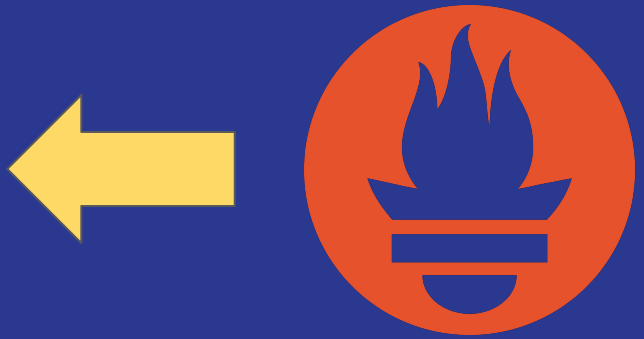
Why metrics?

What about resources?



/metrics

/metrics



Metrics and… testing?

Metrics and… testing?

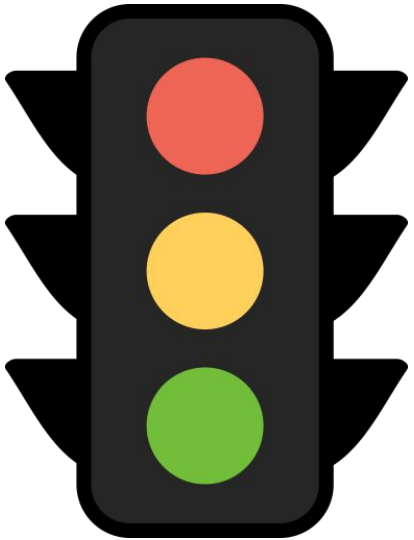
Already instrumented?
Get better E2E testing experience!

Why?

- More granular condition assertions
  - In contrast to binary check (e.g. is exit code 0?)

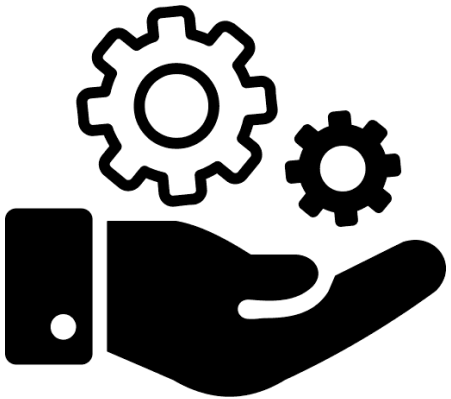
Why?

- Better control over test scenarios
  - Check preconditions more easily
  - Fail fast
  - Wait until metric X equals Y
  - Retry



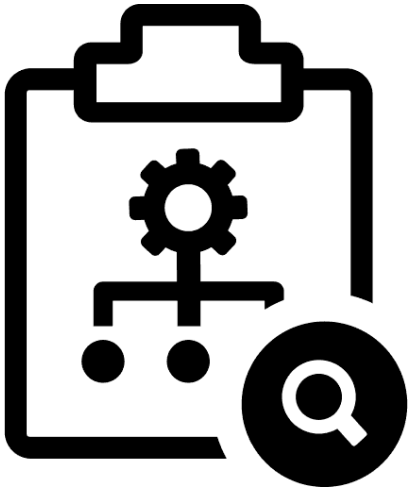
Why?

- Complex test scenarios with ease



Why?

- Gain extra visibility into tested applications
  - Collect more (meta)data about applications



Types of tests

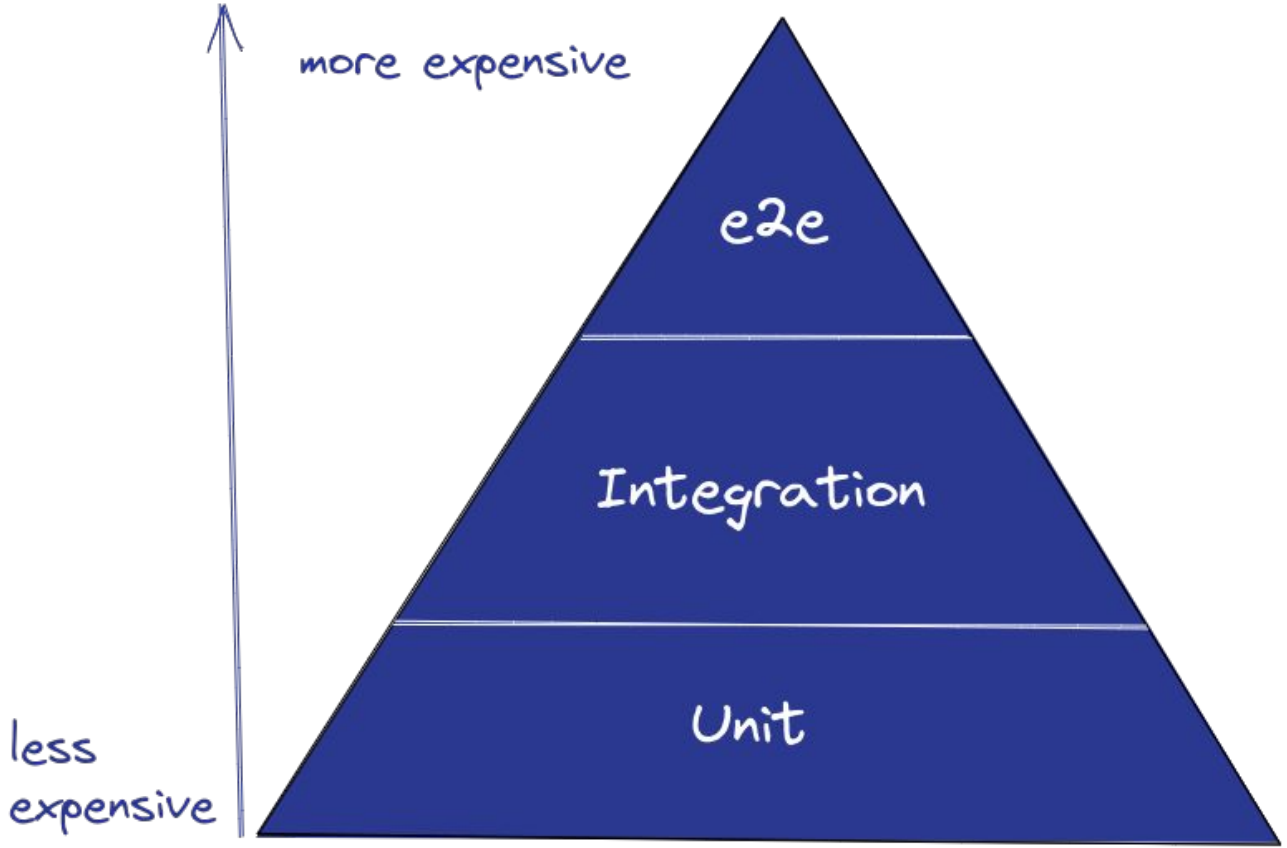
Benchmark test

- Focus on testing performance
- A function is executed multiple times and for each time the output is compared to a standard
- Useful when new external services are added to your application
- golang's testing package natively supports benchmarking

Interactive test

- Makes possible to pause the test and manually play with the scenario in progress
- Helps with getting an idea of how multiple services interact with each other

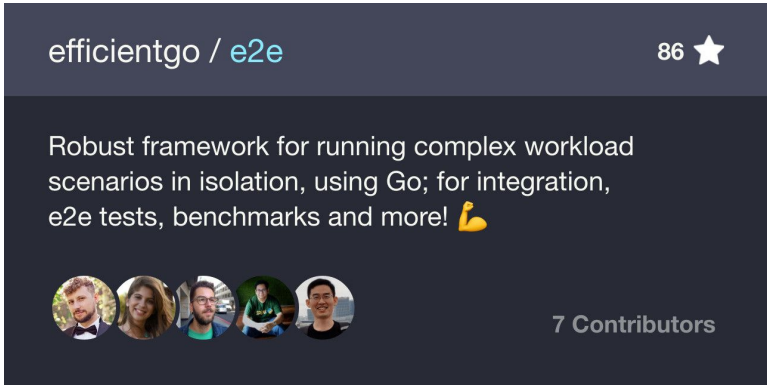
Types of tests



Practical application

Introducing efficientgo E2E framework

- https://github.com/efficientgo/e2e
- Originated with the Cortex project



- Written in Go, for use in Go-based projects
- Services as containers (Docker)
- Monitoring included out of the box
- Supports different usage models (E2E testing, benchmarks, interactive setups)

Usage in real world projects

- Thanos
  - https://github.com/thanos-io/thanos

- Observatorium
  - https://github.com/observatorium/api

Real world usage - Thanos

- https://github.com/thanos-io/thanos/blob/main/examples/interactive/interactive_test.go

```
// createData generates some blocks for us to play with and makes them
// available to store and Prometheus instances.
//
// You can choose different profiles by setting the BLOCK_PROFILE environment variable
// Available profiles can be found at https://github.com/thanos-io/thanosbench/blob/mas
func createData() (perr error) {
        profile := os.Getenv("BLOCK_PROFILE")
        if profile == "" {
                profile = defaultProfile //     Use "continuous-1w-1series-10000apps" i
        }

        fmt.Println("Re-creating data (can take minutes)...")
        defer func() {
                if perr != nil {
                        _ = os.RemoveAll(data)
                }
        }()

        if err := exec(
                "sh", "-c",
                fmt.Sprintf("mkdir -p %s && "+
                        "docker run -i quay.io/thanos/thanosbench:v0.3.0-rc.0 block pla
                        "docker run -v %s/:/shared -i quay.io/thanos/thanosbench:v0.3.0
        ); err != nil {
                return err
        }
```

```
// TestReadOnlyThanosSetup runs read only Thanos setup that has data from `maxTimeFresh - 2w` to `maxTim
// Run with test args `-timeout 9999m`.
func TestReadOnlyThanosSetup(t *testing.T) {
        t.Skip("This is interactive test - it will run until you will kill it or curl 'finish' endpoint.

        // Create series of TSDB blocks. Cache them to 'data' dir so we don't need to re-create on every
        _, err := os.Stat(data)
        if os.IsNotExist(err) {
                testutil.Ok(t, createData())
        } else {
                testutil.Ok(t, err)
                fmt.Println("Skipping blocks generation, found data directory.")
        }

        e, err := e2e.NewDockerEnvironment("interactive")
        testutil.Ok(t, err)
        t.Cleanup(e.Close)

        m, err := e2emonitoring.Start(e)
        testutil.Ok(t, err)
```

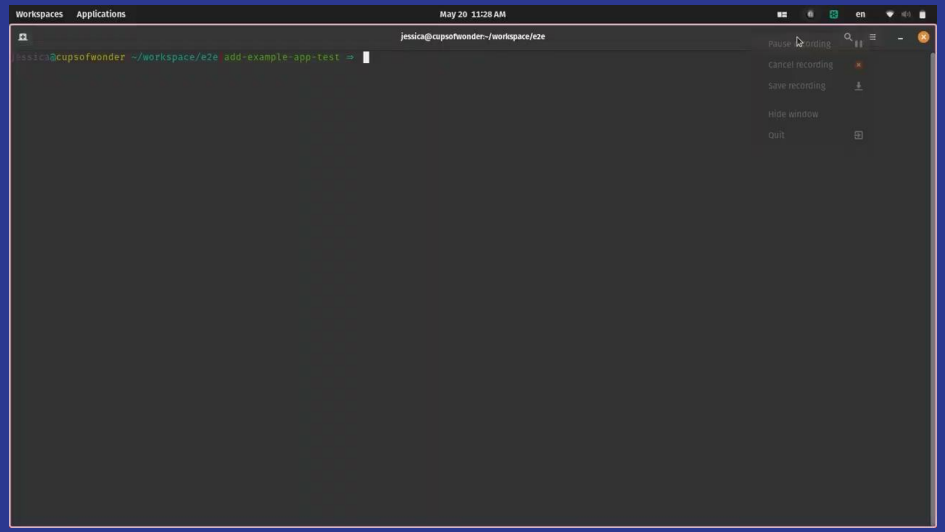
Real world usage - Observatorium

- https://github.com/observatorium/api/blob/main/test/e2e/interactive_test.go

```go
//go:build interactive

package e2e

import (
	"fmt"
	"testing"

	"github.com/efficientgo/e2e"
	e2einteractive "github.com/efficientgo/e2e/interactive"
	"github.com/efficientgo/tools/core/pkg/testutil"
)

func TestInteractiveSetup(t *testing.T) {
	fmt.Printf("Starting services...\n")

	e, err := e2e.NewDockerEnvironment(envInteractive)
	testutil.Ok(t, err)
	t.Cleanup(e.Close)

	prepareConfigsAndCerts(t, interactive, e)
	_, token, rateLimiterAddr := startBaseServices(t, e, interactive)
	readEndpoint, writeEndpoint, readExtEndpoint := startServicesForMetrics(t, e)
	logsEndpoint, logsExtEndpoint := startServicesForLogs(t, e)
	rulesEndpoint := startServicesForRules(t, e)
	internalOtlpEndpoint, httpExternalQueryEndpoint, httpInternalQueryEndpoint := startServicesForTraces(t, e)

	api, err := newObservatoriumAPIService(
		e,
		withMetricsEndpoints("http://"+readEndpoint, "http://"+writeEndpoint),
		withLogsEndpoints("http://"+logsEndpoint),
		withRulesEndpoint("http://"+rulesEndpoint),
		withRateLimiter(rateLimiterAddr),
		withGRPCListenEndpoint(":8317"),
		withOtelTraceEndpoint(internalOtlpEndpoint),
		withJaegerEndpoint("http://"+httpInternalQueryEndpoint),
	)
	testutil.Ok(t, err)
	testutil.Ok(t, e2e.StartAndWaitReady(api))
```

```go
	up, err := newUpRun(
		e, "up-metrics-read-write", metrics,
		"https://"+api.InternalEndpoint("https")+"/api/metrics/v1/"+defaultTenantName+"/api/v1/query",
		"https://"+api.InternalEndpoint("https")+"/api/metrics/v1/"+defaultTenantName+"/api/v1/receive",
		withToken(token),
		withRunParameters(&runParams{period: "5000ms", threshold: "1", latency: "10s", duration: "0"}),
	)
	testutil.Ok(t, err)
	testutil.Ok(t, e2e.StartAndWaitReady(up))

	testutil.Ok(t, e2einteractive.OpenInBrowser("http://"+readExtEndpoint))

	fmt.Printf("\n")
	fmt.Printf("You're all set up!\n")
	fmt.Printf("==========================================\n")
	fmt.Printf("Observatorium API on host machine:          %s \n", api.Endpoint("https"))
	fmt.Printf("Observatorium internal server on host machine:      %s \n", api.Endpoint("http-internal"))
	fmt.Printf("Thanos Query on host machine:                       %s \n", readExtEndpoint)
	fmt.Printf("Loki on host machine:                               %s \n", logsExtEndpoint)
	fmt.Printf("Observatorium gRPC API on host machine:           %s\n", api.Endpoint("grpc"))
	fmt.Printf("Jaeger Query on host machine (HTTP):              %s\n", httpExternalQueryEndpoint)

	fmt.Printf("API Token:                                      %s \n\n", token)

	testutil.Ok(t, e2einteractive.RunUntilEndpointHit())
}
```

Demo time!


Workspaces
Applications
May 20 11:28 AM
en
jessica@cupsofwonder:~/workspace/e2e
jessica@cupsofwonder ~/workspace/e2e add-example-app-test =
Pause recording
Cancel recording
Save recording
Hide window
Quit

Thank you!

Resources:

- *https://github.com/efficientgo/e2e*
- *https://github.com/brancz/prometheus-example-app*
- *https://github.com/thanos-io/thanos*
- *https://github.com/observatorium/api*

Questions?

@cupsofwonder
@gmat90