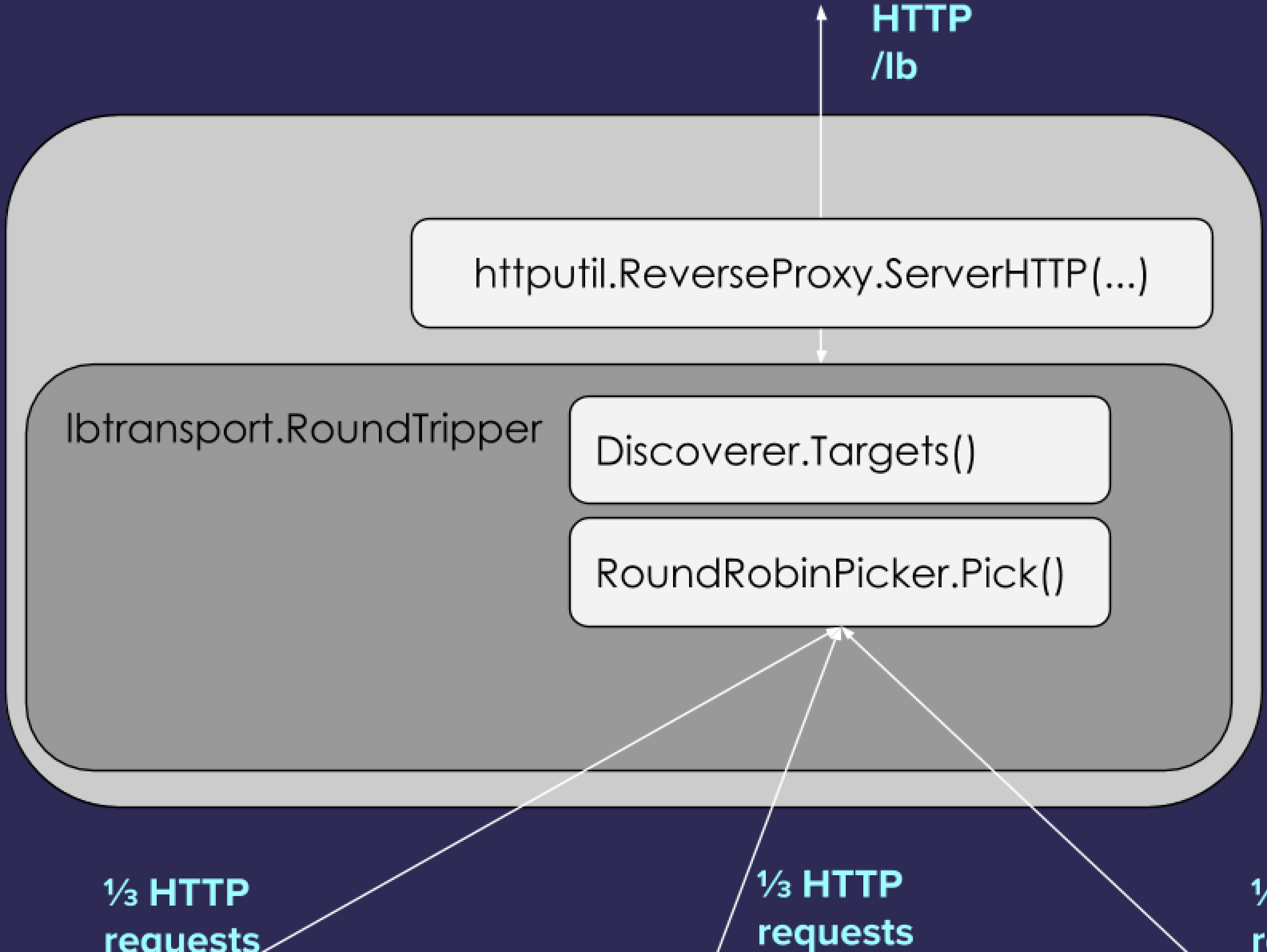
```go
package main

import "fmt"

func main() {
    fmt.Println("Are you testing your Observability?")
    fmt.Println("      --- Metrics Edition ---      ")



    fmt.Println("    GoDays 22.01.2020, Berlin      ")
}
```

@bwplotka
@kakkoyun

@bwplotka
@kakkoyun

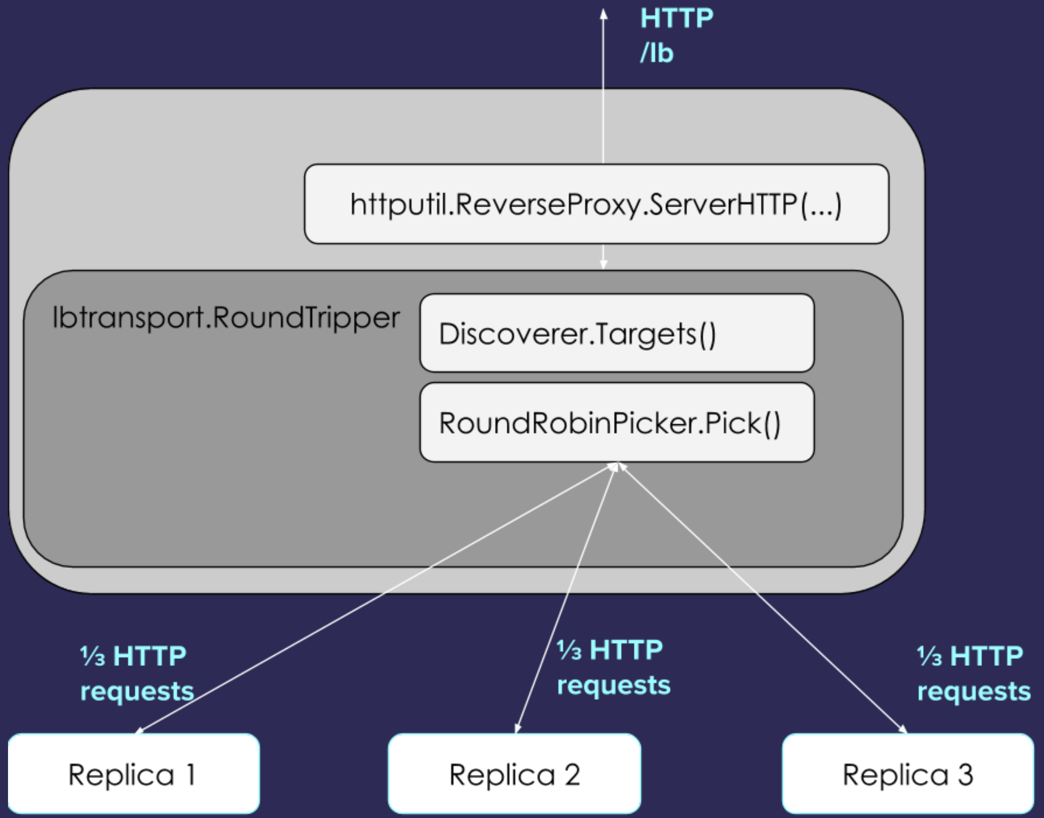
Thanos

@bwplotka
@kakkoyun



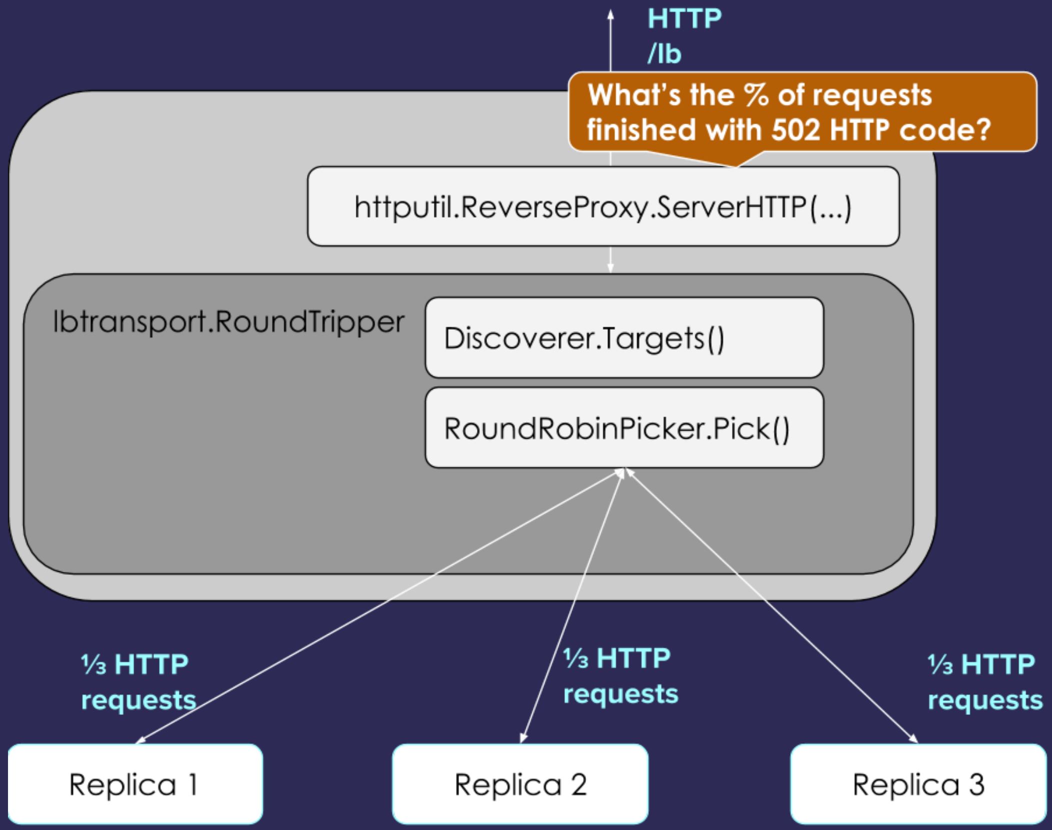
@bwplotka
@kakkoyun

## Wait… Anything missing?



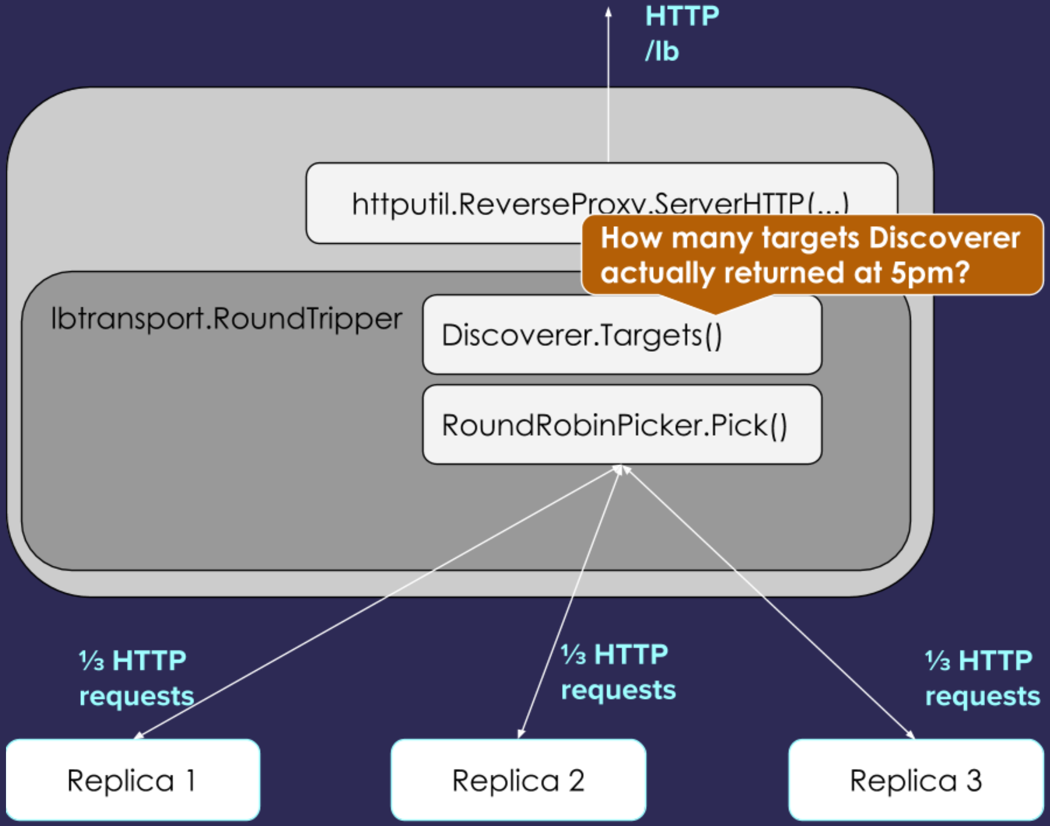
@bwplotka
@kakkoyun

## Wait... Anything missing?



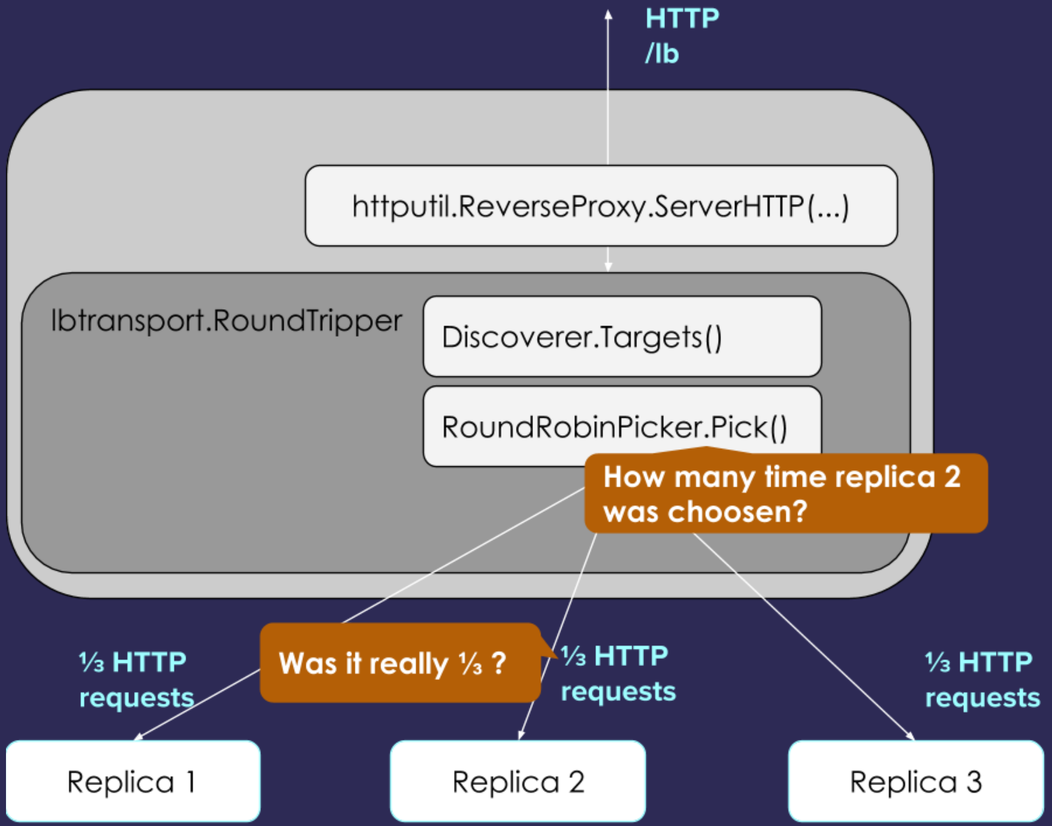
@bwplotka
@kakkoyun

## Wait... Anything missing?

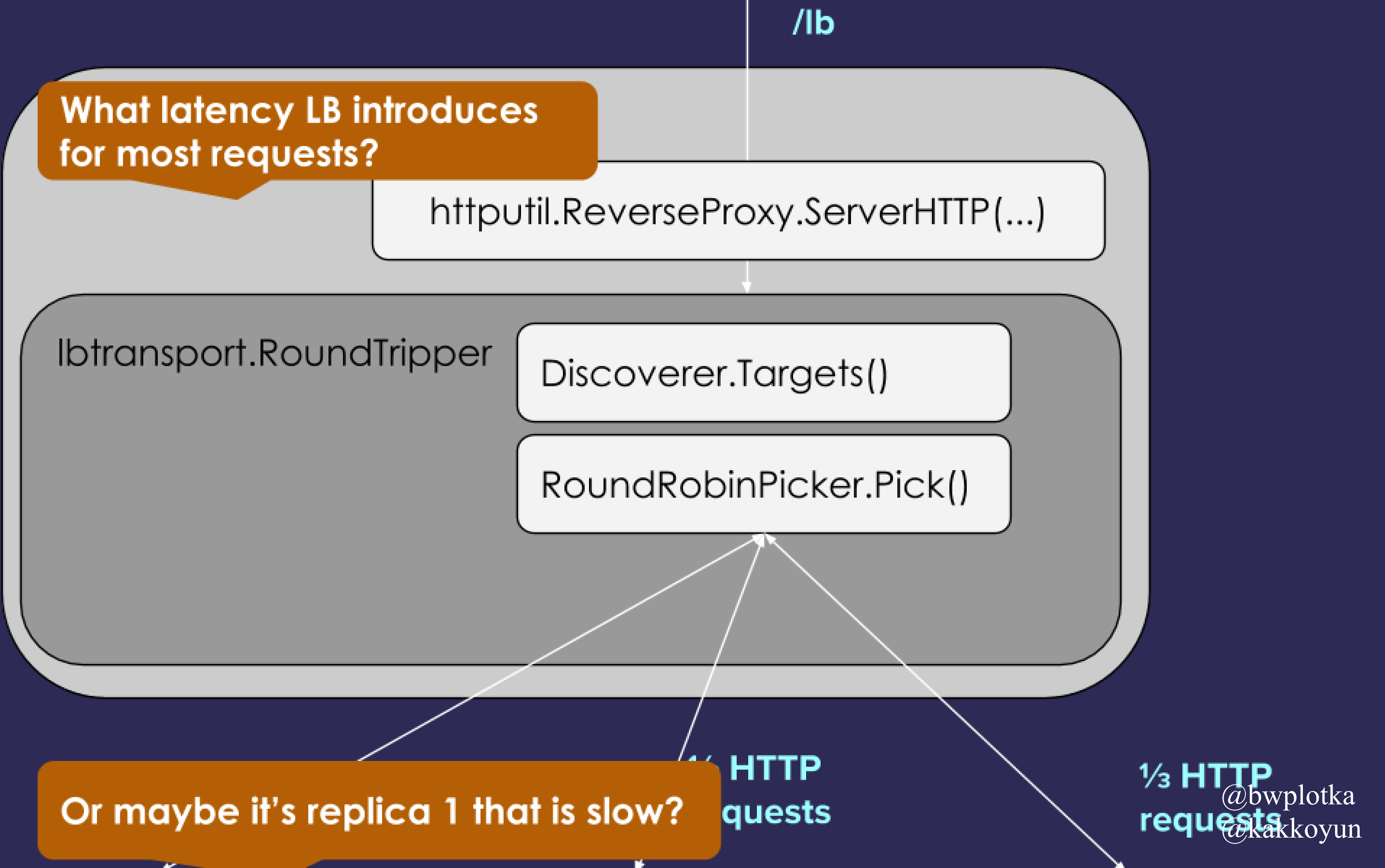


@bwplotka
@kakkoyun

## Wait... Anything missing?

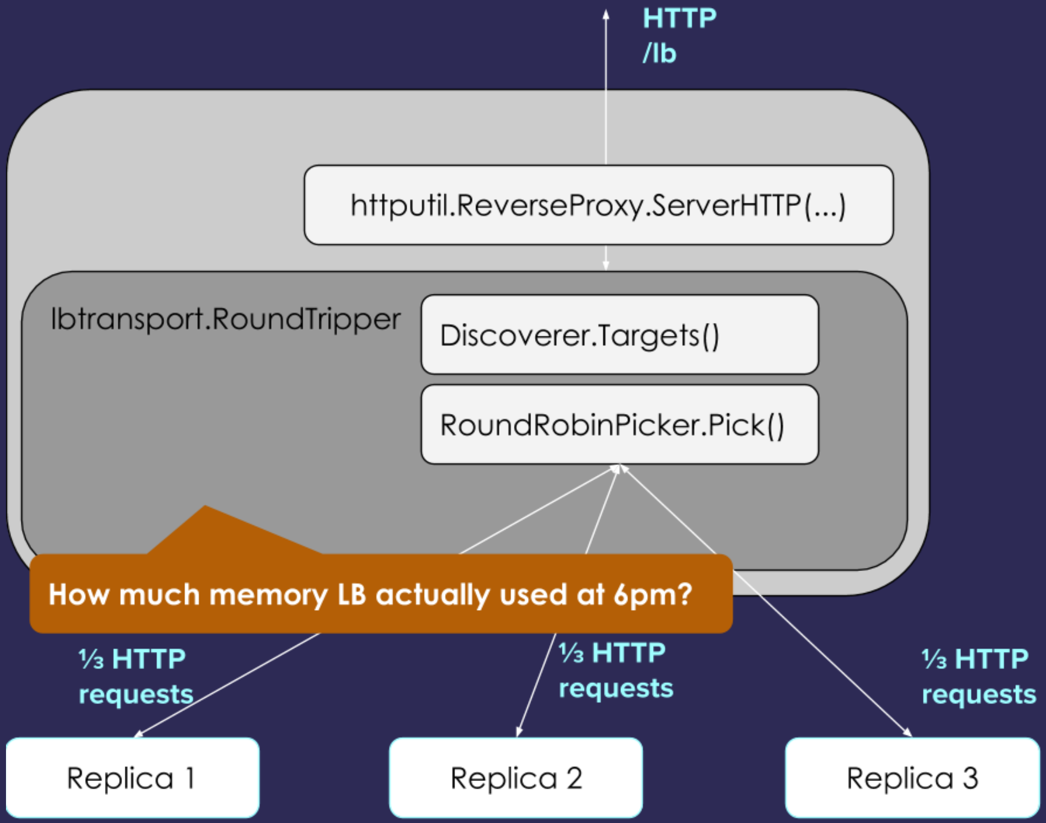


@bwplotka
@kakkoyun


/lb
What latency LB introduces for most requests?
httputil.ReverseProxy.ServerHTTP(...)
lbtransport.RoundTripper
Discoverer.Targets()
RoundRobinPicker.Pick()
HTTP
quests
⅓ HTTP
requests
Or maybe it's replica 1 that is slow?

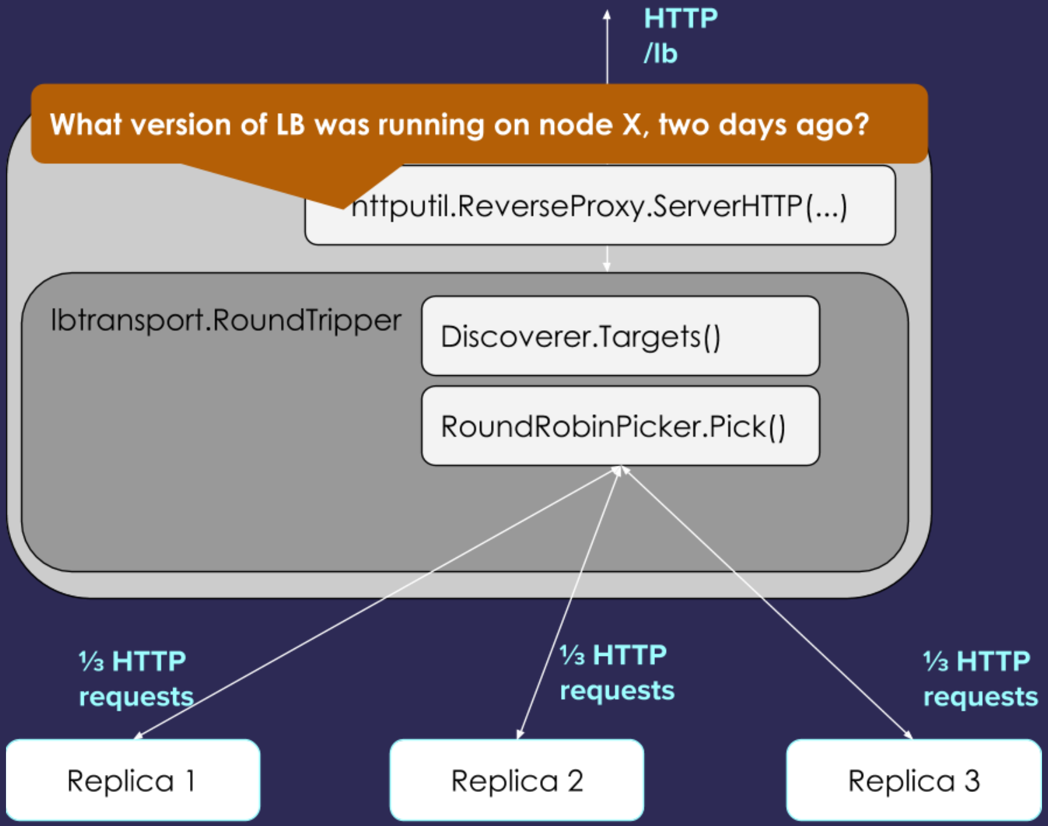
@bwplotka
@kakkoyun

## Wait... Anything missing?



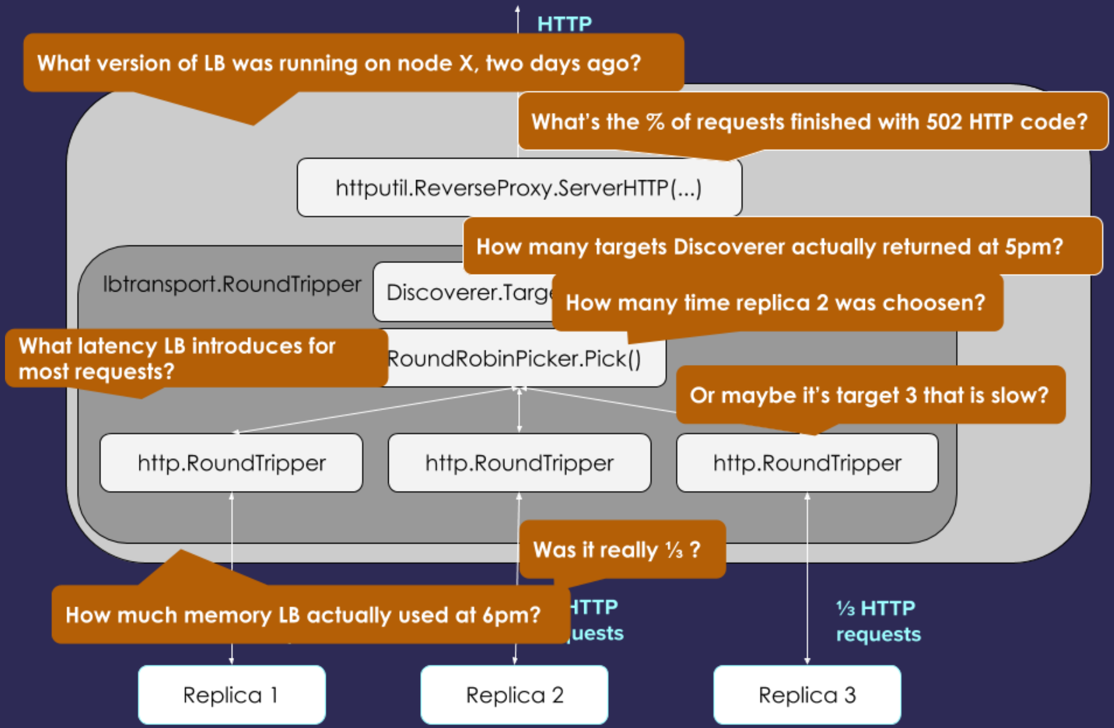
@bwplotka
@kakkoyun

## Wait... Anything missing?



@bwplotka
@kakkoyun

# We need Monitoring!


HTTP
What version of LB was running on node X, two days ago?
What's the % of requests finished with 502 HTTP code?
httputil.ReverseProxy.ServerHTTP(...)
How many targets Discoverer actually returned at 5pm?
lbtransport.RoundTripper
Discoverer.Targ
How many time replica 2 was choosen?
What latency LB introduces for most requests?
RoundRobinPicker.Pick()
Or maybe it's target 3 that is slow?
http.RoundTripper
http.RoundTripper
http.RoundTripper
Was it really ⅓ ?
HTTP
uests
⅓ HTTP requests
How much memory LB actually used at 6pm?
Replica 1
Replica 2
Replica 3

@bwplotka
@kakkoyun

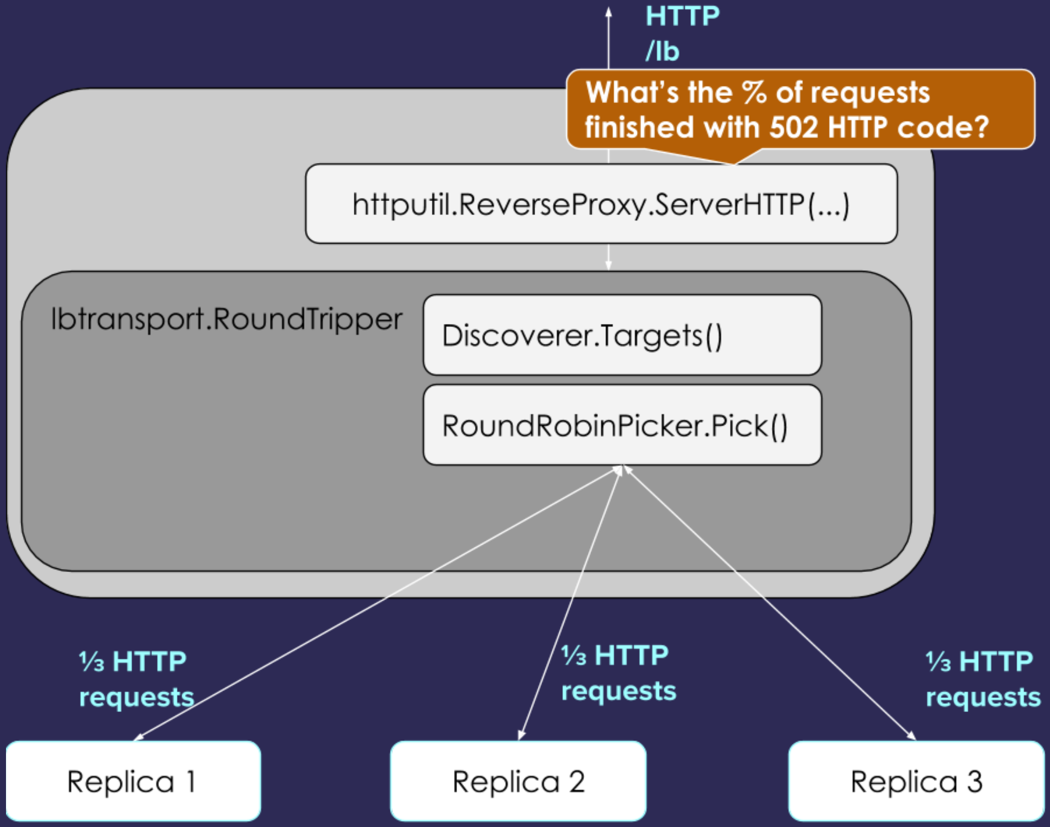
## Let's instrument our LB with Prometheus metrics!

- Cheap
- Near Real Time
- Actionable (Alert-able)

http://prometheus.io

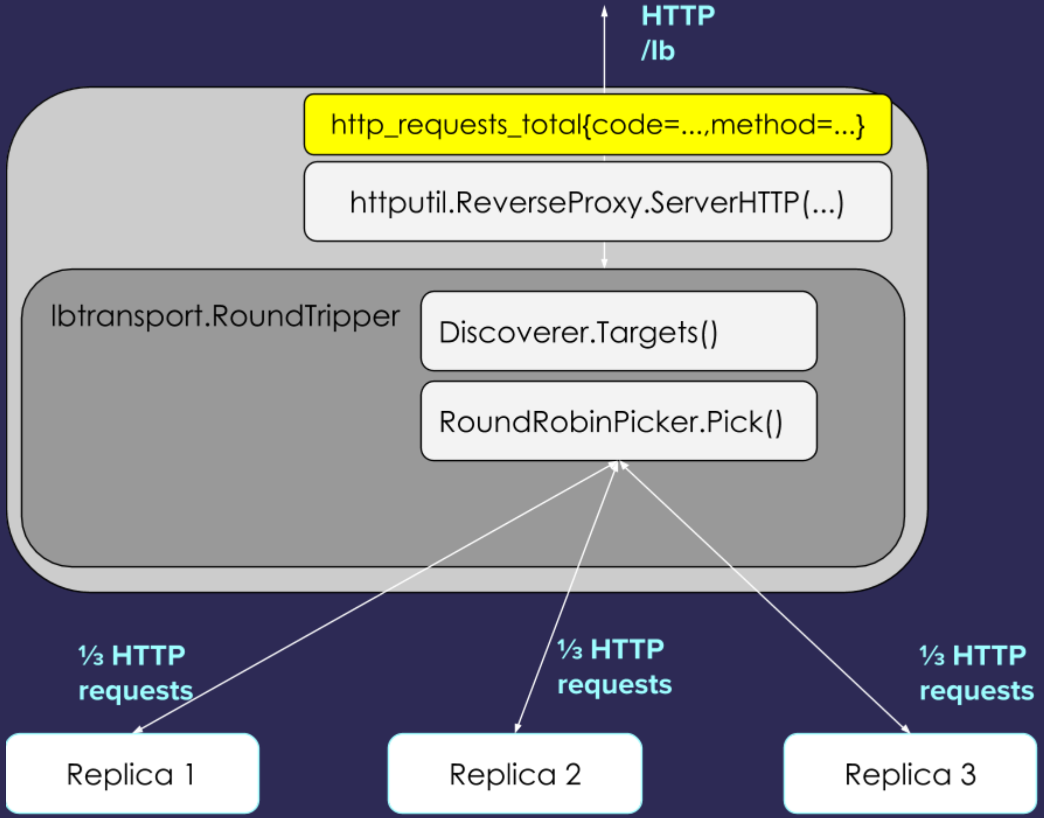
@bwplotka
@kakkoyun

## Let's instrument our LB with Prometheus metrics!



@bwplotka
@kakkoyun

## Let's instrument our LB with Prometheus metrics!



@bwplotka
@kakkoyun

## Server HTTP request counter

```go
    // Increment our counter with written status code and request method.
    serverRequestsTotal.WithLabelValues(statusRec.Status(), r.Method)).Add(1)
  }

  func init() {
      prometheus.MustRegister(serverRequestsTotal)
  }

  mux := http.NewServeMux()
  mux.Handle("/metrics", promhttp.Handler())
  mux.Handle("/lb", ...)

  // Other handlers...

  srv := &http.Server{Handler: mux}

  // Run server...
```

Add /metrics handler.

@bwplotka
@kakkoyun

## From code to graph

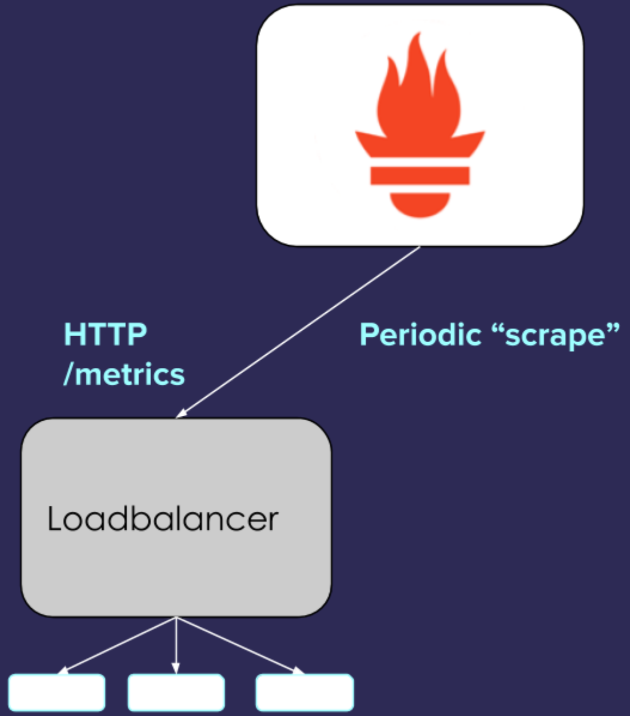
```
var (
    serverRequestsTotal = prometheus.NewCounterVec(
        prometheus.CounterOpts{
            Name: "http_requests_total",
            Help: "Tracks the number of HTTP requests.",
        }, []string{"code", "method"},
    )
)

// Part of response from /metrics HTTP endpoint.
# HELP http_requests_total Tracks the number of HTTP requests.
# TYPE http_requests_total counter
http_requests_total{code="200", method="get"} 1089
http_requests_total{code="500", method="get"} 46
```

@bwplotka
@kakkoyun

It is now exposed under /metrics.

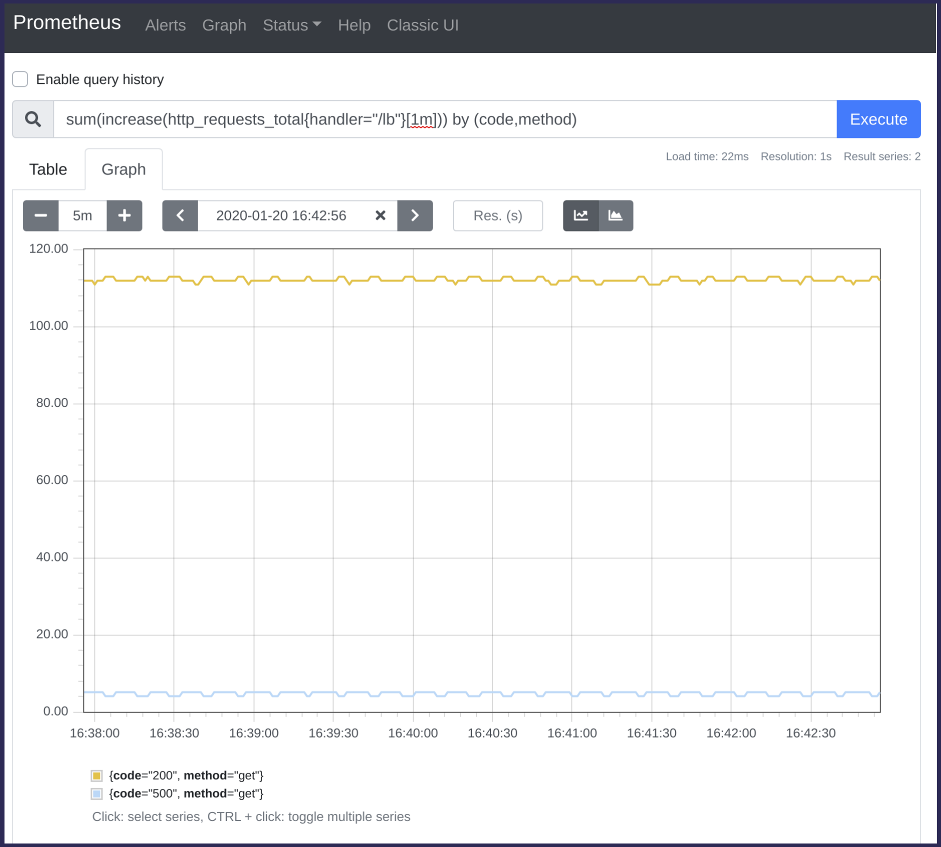
## Prometheus can now collect metrics from our Loadbalancer

*Pulling e.g every 15 seconds*



@bwplotka
@kakkoyun

# Graph: Prometheus UI



```
sum(increase(http_requests_total{handler="/lb"}[1m])) by (code,method)
```

Load time: 22ms Resolution: 1s Result series: 2

2020-01-20 16:42:56

🟨 {code="200", **method**="get"}
🟦 {code="500", **method**="get"}

@bwplotka
@kakkoyun

## + Pitfalls

@bwplotka
@kakkoyun

## Pitfall #1: Global Registry

"magic is bad; global state is magic" by Peter Bourgon

@bwplotka
@kakkoyun

## Pitfall #1: Getting rid of globals.

```
)
metrics3 := NewServerMetrics(
  prometheus.WrapWithLabels(prometheus.Labels{"handler":"/three"}, reg),
)

// For endpoint /one:
metrics1.requestsTotal.WithLabelValues(statusRec.Status(), r.Method)).Add(1)

// For endpoint /two:
metrics2.requestsTotal.WithLabelValues(statusRec.Status(), r.Method)).Add(1)

// For endpoint /three:
metrics3.requestsTotal.WithLabelValues(statusRec.Status(), r.Method)).Add(1)
```

```
# HELP http_requests_total Tracks the number of HTTP requests.
# TYPE http_requests_total counter
http_requests_total{handler="/one", code="200", method="get"} 1445
http_requests_total{handler="/one", code="500", method="get"} 23
http_requests_total{handler="/two", code="200", method="get"} 445
http_requests_total{handler="/two", code="500", method="get"} 0
http_requests_total{handler="/three", code="200", method="get"} 645
http_requests_total{handler="/three", code="500", method="get"} 40
```

We can have request counter per handler (:

@bwplotka
@kakkoyun

## Pitfall #2: No Tests For Metrics

@bwplotka
@kakkoyun

## Pitfall #2: No testing: Let's add tests!

```
    // other cases...
} {
    if ok := t.Run("", func(t *testing.T) {
        // Prepare mocks...

        rec := httptest.NewRecorder()
        lb.ServeHTTP(
            rec,
            httptest.NewRequest("GET", "http://mocked", nil),
        )
        testutil.Equals(t, tcase.expectedCode, rec.Code)
        testutil.Equals(t, tcase.expectedHost, rec.Header().Get("X-lb-host"))

        // === RUN   TestLoadbalancer/#00
        //       --- FAIL: TestLoadbalancer/#00 (0.00s)
        //           transport_test.go:190:
        //
        //               exp: 1
        //
        //               got: 0
        //   FAIL
        //
        testutil.Equals(
            t,
            tcase.requestsCode502,
            promtestutil.ToFloat64(metrics.requestsTotal.WithLabelValues("502", "get")),
         )
        testutil.Equals(t, 1, promtestutil.CollectAndCount(lb.metrics.requestsTotal))

    }); !ok {
        return
    }
}
```

Now, test will catch our bug!

@bwplotka
@kakkoyun

## Pitfall #3: Lack of Consistency

**The Four Golden Signals, USE method, RED method etc...**

- R: Requests per second (saturation).
- E: Errors per second.
- D: Duration (tail latency).

@bwplotka
@kakkoyun

## Pitfall #3: Consistency

```go
type ServerMetrics struct {
    requestsTotal    *prometheus.CounterVec
    requestDuration  *prometheus.HistogramVec
}

// NewServerMetrics provides ServerMetrics.
func NewServerMetrics(reg prometheus.Registerer) *ServerMetrics {
    m := &ServerMetrics{
        requestsTotal: prometheus.NewCounterVec(
            prometheus.CounterOpts{
                Name: "http_requests_total",
                Help: "Tracks the number of HTTP requests.",
            }, []string{"code", "method"},
        ),
        requestDuration: prometheus.NewHistogramVec(
            prometheus.HistogramOpts{
                Name:    "http_request_duration_seconds",
                Help:    "Tracks the latencies for HTTP requests.",
                Buckets: []float64{0.001, 0.01, 0.1, 0.3, 0.6, 1, 3, 6, 9, 20, 30, 60, 90, 120},
            },
            []string{"code", "method"},
        ),
    }
    reg.MustRegister(m.requestsTotal, m.requestDuration)
    return ins
}

// Top level ServeHTTP handler.
func ServeHTTP(w http.ResponseWriter, r *http.Request) {
  start := time.Now()
  defer metrics.requestDuration.WithLabelValues(statusRec.Status(), r.Method)).Observe(time.Since(start))

  statusRec := newStatusRecorder(w)
  next.ServeHTTP(statusRec, r)

  // Increment our counter with written status code and request method.
  metrics.requestsTotal.WithLabelValues(statusRec.Status(), r.Method)).Add(1)
}
```

Red method satisfied

@bwplotka
@kakkoyun

## Pitfall #4: Naming: Not conforming naming convention

@bwplotka
@kakkoyun

## There is an official documentation on naming conventions

https://prometheus.io/docs/practices/naming/#metric-and-label-naming

@bwplotka
@kakkoyun

## Pitfall #4: Naming: Not conforming naming convention

```
http_request_duration_seconds
node_memory_usage_bytes

http_requests_total
process_cpu_seconds_total

build_info
```

_info suffix for metadata metrics

@bwplotka
@kakkoyun

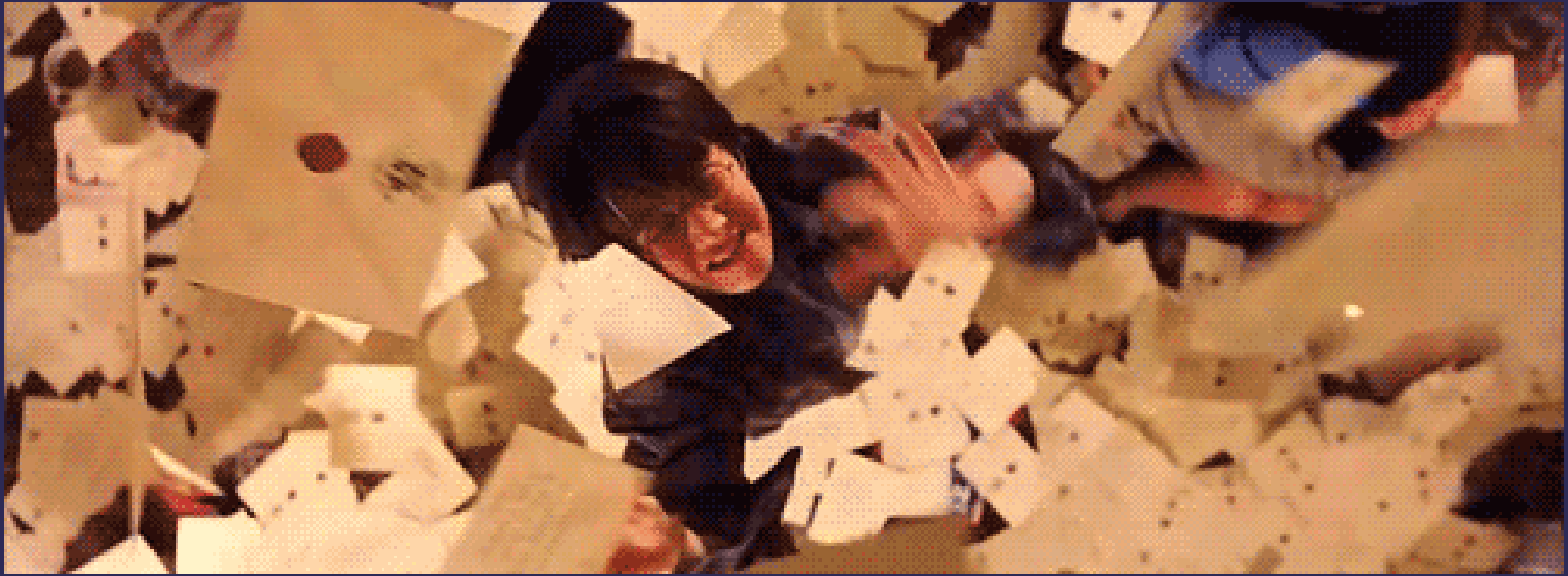
## Pitfall #4: Naming: stability

```
// NewServerMetrics provides ServerMetrics.
func NewServerMetrics(reg prometheus.Registerer) *ServerMetrics {
    m := &ServerMetrics{
        requestsTotal: prometheus.NewCounterVec(
            prometheus.CounterOpts{
                Name: "http_protocol_requests_total",
                Help: "Tracks the number of HTTP requests.",
            }, []string{"code", "method"},
        ),
    }
    reg.MustRegister(m.requestsTotal)
    return ins
}

## BOOM!💥 This alert will never fire but also will not fail!
## Rename can cause issues like this in Alerts, Recording rules, Dashboards and more..
alert: HttpToMany502Errors
  expr: |
    sum(rate(http_requests_total{status="502"}[1m])) /
      sum(rate(http_requests_total[1m])) * 100 > 5
  for: 5m
  labels:
    severity: error
  annotations:
    summary: "HTTP errors 502 (instance {{ $labels.instance }})"
    description: |
      "Too many HTTP requests with status 502 (> 5%)\n  VALUE = {{ $value }}\n
      LABELS: {{ $labels }}"
```

Ups...

@bwplotka
@kakkoyun

## Pitfall #5: Cardinality: Unbounded labels



@bwplotka
@kakkoyun

```
# HELP conntrack_dialer_conn_failed_total Total number of connections failed to dial by the dialer.
# TYPE conntrack_dialer_conn_failed_total counter
conntrack_dialer_conn_failed_total{reason="<nil>"} 1
conntrack_dialer_conn_failed_total{reason="lookup example.com on localhost: err..."} 1
conntrack_dialer_conn_failed_total{reason="lookup thanos.io on 8.8.8.8: err..."} 1
conntrack_dialer_conn_failed_total{reason="lookup redhat.com on localhost: err..."} 1
conntrack_dialer_conn_failed_total{reason="syscall: unimplemented EpollWait"} 1
conntrack_dialer_conn_failed_total{reason="syscall.SIGINT: series of unfortunate things happened"} 1
conntrack_dialer_conn_failed_total{reason="unix: test returned fd in TestEpoll"} 1
conntrack_dialer_conn_failed_total{reason="Invalid value for argument: client: nil"} 1
```

Let's check out our metrics... 😲

@bwplotka
@kakkoyun

```go
        return conn, err
    }

    return &clientConnTracker{...}, nil
}

func dialErrToReason(err error) string {
    var e *net.OpError
    if errors.As(err, &e) {
        switch nestErr := e.Err.(type) {
        case *net.DNSError:
            return failedResolution
        case *os.SyscallError:
            if nestErr.Err == syscall.ECONNREFUSED {
                return failedConnRefused
            }

            return failedUnknown
        }

        if e.Timeout() {
            return failedTimeout
        }
    } else if err == context.Canceled || err == context.DeadlineExceeded {
        return failedTimeout
    }

    return failedUnknown
}
```

Map errors to label values

@bwplotka
@kakkoyun

## Pitfall #6: Cardinality: Histogram Explosion

@bwplotka
@kakkoyun

## Pitfall #6: Histogram Cardinality Explosion

```
# HELP http_request_duration_seconds Tracks the latencies for HTTP requests.
# TYPE http_request_duration_seconds histogram
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.001"} 0
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.01"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.1"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.3"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.6"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="1"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="3"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="6"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="9"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="20"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="30"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="40"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="50"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="60"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="90"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="120"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="150"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="200"} 18
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="+Inf"} 18
http_request_duration_seconds_sum{code="200",handler="/metrics",method="get"} 0.0480237
http_request_duration_seconds_count{code="200",handler="/metrics",method="get"} 18
```

@bwplotka
@kakkoyun

@bwplotka
@kakkoyun

# Pitfall #6: Histogram Cardinality Explosion

```
# HELP http_request_duration_seconds Tracks the latencies for HTTP requests.
# TYPE http_request_duration_seconds histogram
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="0.001"} 0
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="0.01"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="0.1"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="0.3"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="0.6"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="1"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="3"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="6"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="9"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="20"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="30"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="40"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="50"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="60"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="90"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="120"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="150"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="200"} 2
http_request_duration_seconds_bucket{code="200",handler="/lb",method="post",le="+Inf"} 2
http_request_duration_seconds_sum{code="200",handler="/lb",method="post"} 0.0026940999999999996
http_request_duration_seconds_count{code="200",handler="/lb",method="post"} 2
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.001"} 0
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.01"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.1"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.3"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.6"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="1"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="3"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="6"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="9"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="20"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="30"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="40"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="50"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="60"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="90"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="120"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="150"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="200"} 146
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="+Inf"} 146
http_request_duration_seconds_sum{code="200",handler="/metrics",method="get"} 0.3082099000000001
http_request_duration_seconds_count{code="200",handler="/metrics",method="get"} 146
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="0.001"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="0.01"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="0.1"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="0.3"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="0.6"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="1"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="3"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="6"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="9"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="20"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="30"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="40"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="50"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="60"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="90"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="120"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="150"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="200"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-500-sometimes",method="post",le="+Inf"} 1
http_request_duration_seconds_sum{code="200",handler="demo-500-sometimes",method="post"} 4.01e-05
http_request_duration_seconds_count{code="200",handler="demo-500-sometimes",method="post"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="0.001"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="0.01"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="0.1"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="0.3"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="0.6"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="1"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="3"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="6"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="9"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="20"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="30"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="40"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="50"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="60"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="90"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="120"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="150"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="200"} 1
http_request_duration_seconds_bucket{code="200",handler="demo-refused-conn-sometimes",method="post",le="+Inf"} 1
http_request_duration_seconds_sum{code="200",handler="demo-refused-conn-sometimes",method="post"} 4.46e-05
http_request_duration_seconds_count{code="200",handler="demo-refused-conn-sometimes",method="post"} 1
```

@bwplotka
@kakkoyun

## Pitfall #7: Poorly Chosen Histogram Buckets

@bwplotka
@kakkoyun

```
# HELP http_request_duration_seconds Tracks the latencies for HTTP requests.
# TYPE http_request_duration_seconds histogram
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.001"} 3
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.0012000000000000001"} 3
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.0014000000000000002"} 10
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.0016000000000000003"} 21
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.0018000000000000004"} 25
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="0.0020000000000000005"} 29
http_request_duration_seconds_bucket{code="200",handler="/metrics",method="get",le="+Inf"} 49
http_request_duration_seconds_sum{code="200",handler="/metrics",method="get"} 0.10939249999999999
http_request_duration_seconds_count{code="200",handler="/metrics",method="get"} 49
```

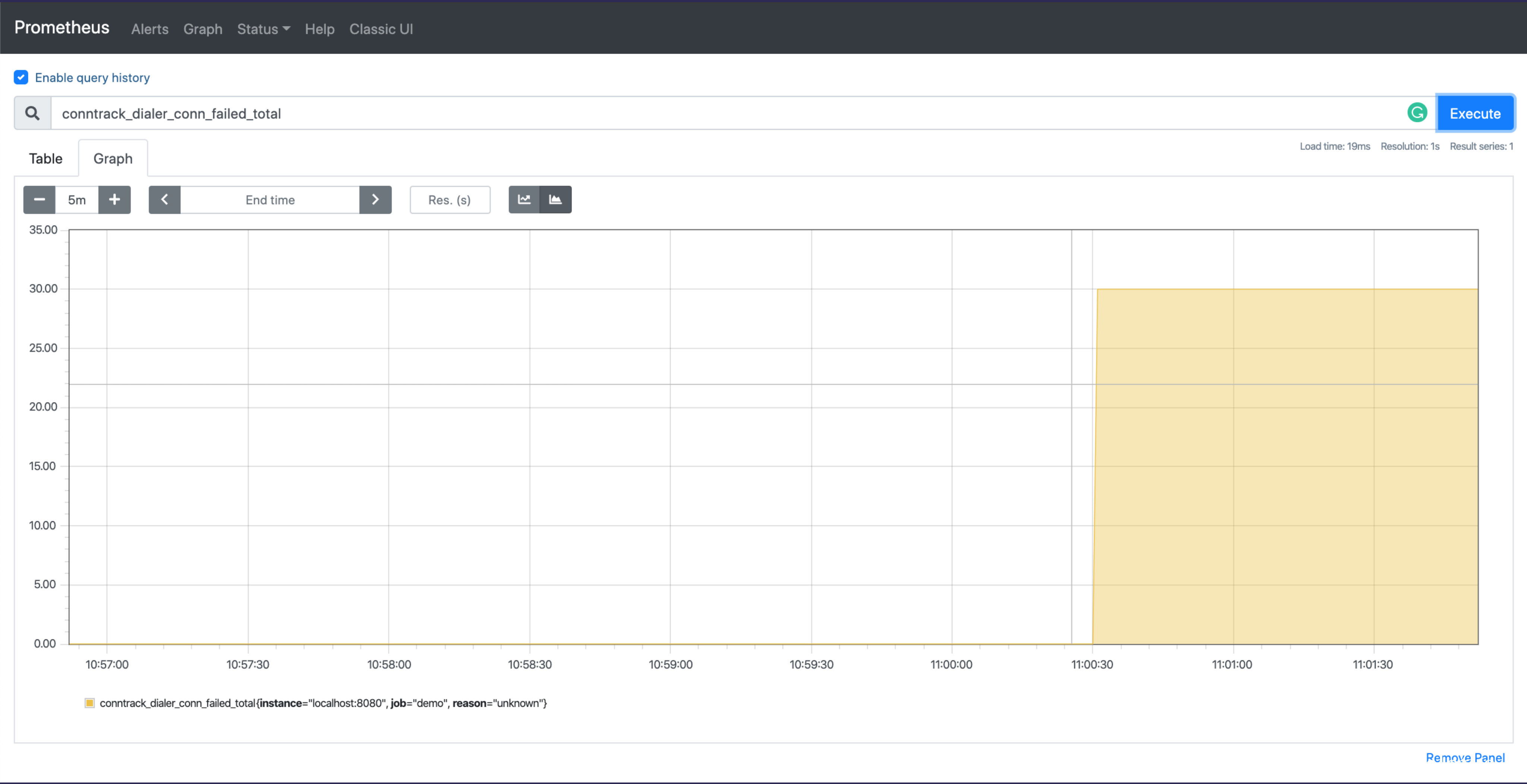
@bwplotka
@kakkoyun

## Pitfall #8: Not Initialized Metrics

@bwplotka
@kakkoyun

```
m.connFailedTotal.WithLabelValues("unknown").Add(30)
```

@bwplotka
@kakkoyun

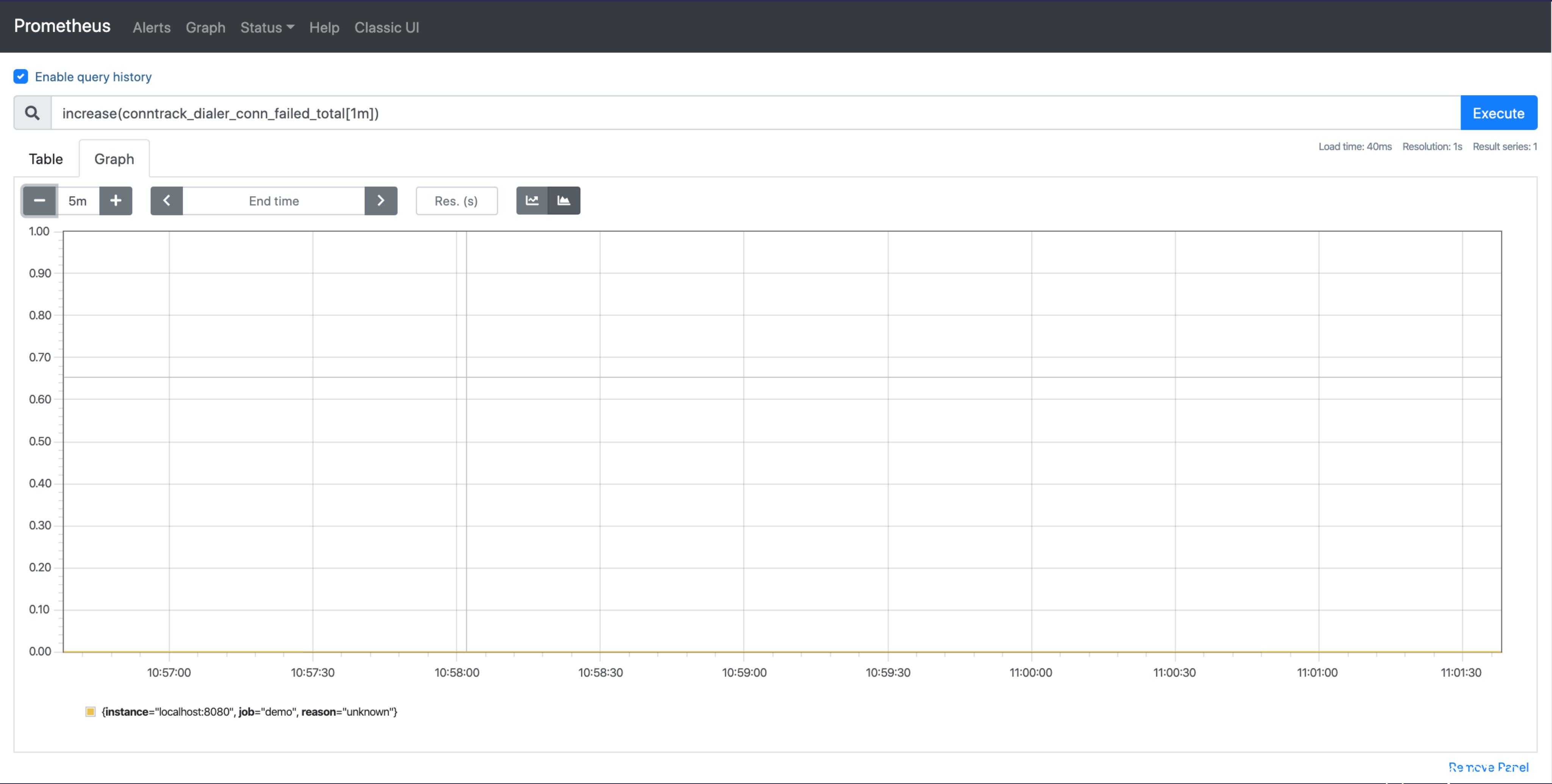


```
conntrack_dialer_conn_failed_total
```

Load time: 19ms Resolution: 1s Result series: 1

🟨 conntrack_dialer_conn_failed_total{**instance**="localhost:8080", **job**="demo", **reason**="unknown"}

@kakkoyun



```
increase(conntrack_dialer_conn_failed_total[1m])
```

Load time: 40ms Resolution: 1s Result series: 1

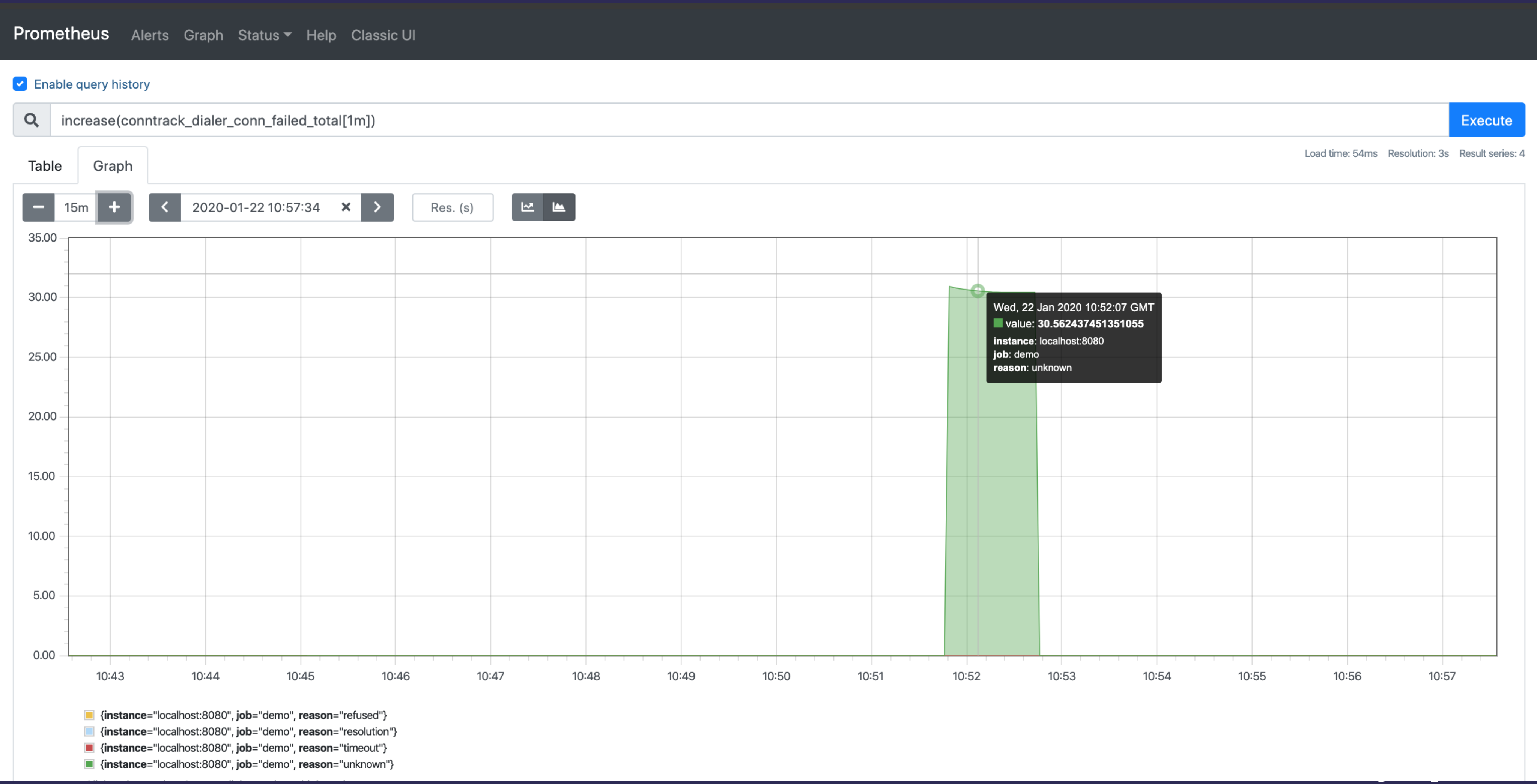
{**instance**="localhost:8080", **job**="demo", **reason**="unknown"}

@kakkoyun

## Pitfall #8: Not initializing Metrics

```
m.connFailedTotal.WithLabelValues("unknown").Add(30)
```

@bwplotka
@kakkoyun



```
increase(conntrack_dialer_conn_failed_total[1m])
```

Load time: 54ms Resolution: 3s Result series: 4

2020-01-22 10:57:34

Wed, 22 Jan 2020 10:52:07 GMT
🟩 value: **30.562437451351055**
**instance**: localhost:8080
**job**: demo
**reason**: unknown

🟨 {**instance**="localhost:8080", **job**="demo", **reason**="refused"}
🟦 {**instance**="localhost:8080", **job**="demo", **reason**="resolution"}
🟥 {**instance**="localhost:8080", **job**="demo", **reason**="timeout"}
🟩 {**instance**="localhost:8080", **job**="demo", **reason**="unknown"}

@kakkoyun

## Summary

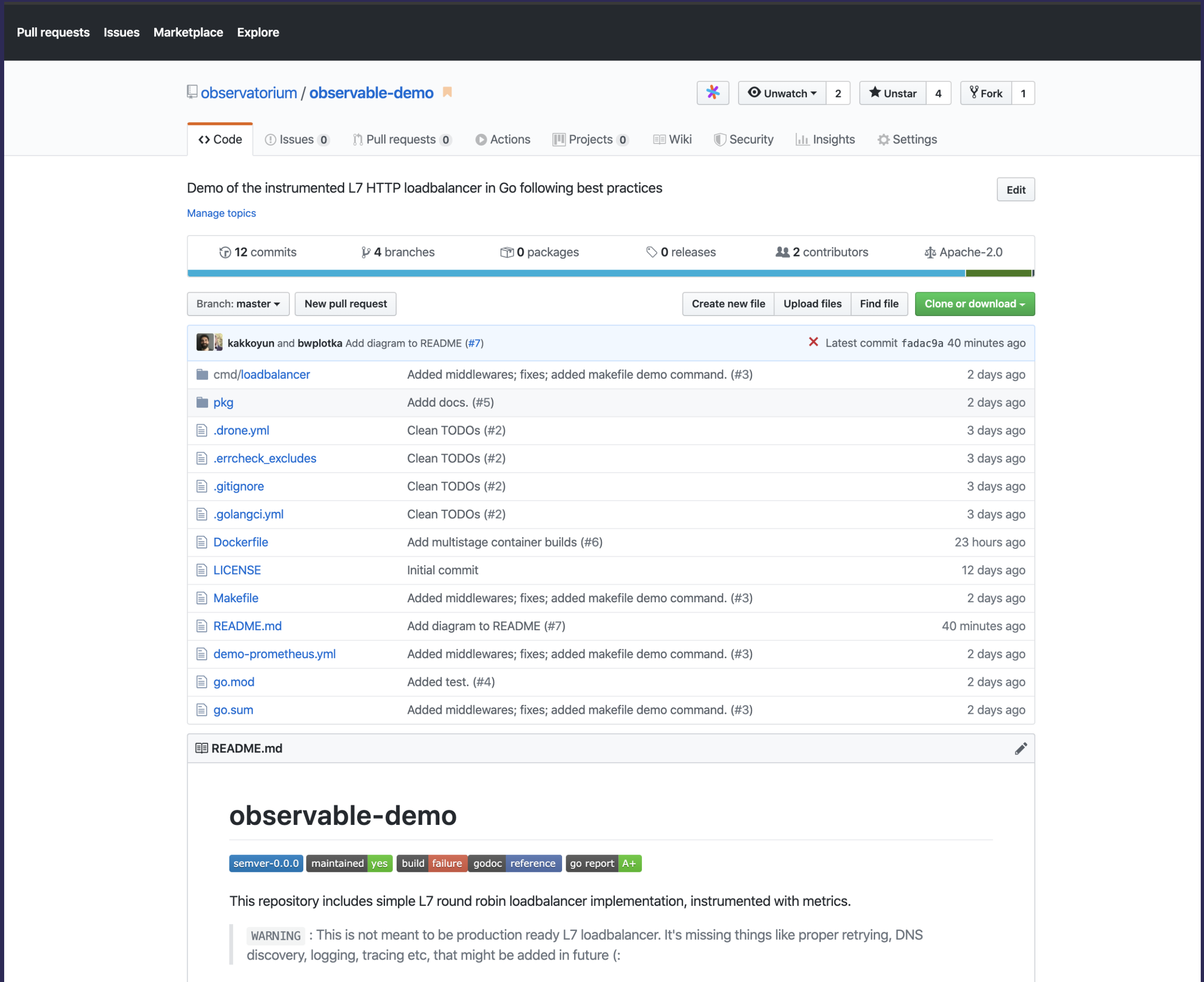
**Monitoring is essential.**

**Unit Test Your Instrumentation.**

**Avoid Global Registry.**

@bwplotka
@kakkoyun

https://github.com/observatorium/observable-demo



Pull requests Issues Marketplace Explore

observatorium / observable-demo

Unwatch 2 | Unstar 4 | Fork 1

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Wiki | Security | Insights | Settings

Demo of the instrumented L7 HTTP loadbalancer in Go following best practices

Manage topics

12 commits | 4 branches | 0 packages | 0 releases | 2 contributors | Apache-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

kakkoyun and bwplotka Add diagram to README (#7) — Latest commit `fadac9a` 40 minutes ago

| | | |
|---|---|---|
| cmd/loadbalancer | Added middlewares; fixes; added makefile demo command. (#3) | 2 days ago |
| pkg | Addd docs. (#5) | 2 days ago |
| .drone.yml | Clean TODOs (#2) | 3 days ago |
| .errcheck_excludes | Clean TODOs (#2) | 3 days ago |
| .gitignore | Clean TODOs (#2) | 3 days ago |
| .golangci.yml | Clean TODOs (#2) | 3 days ago |
| Dockerfile | Add multistage container builds (#6) | 23 hours ago |
| LICENSE | Initial commit | 12 days ago |
| Makefile | Added middlewares; fixes; added makefile demo command. (#3) | 2 days ago |
| README.md | Add diagram to README (#7) | 40 minutes ago |
| demo-prometheus.yml | Added middlewares; fixes; added makefile demo command. (#3) | 2 days ago |
| go.mod | Added test. (#4) | 2 days ago |
| go.sum | Added middlewares; fixes; added makefile demo command. (#3) | 2 days ago |

README.md

## observable-demo

semver-0.0.0 | maintained yes | build failure | godoc reference | go report A+

This repository includes simple L7 round robin loadbalancer implementation, instrumented with metrics.

> `WARNING` : This is not meant to be production ready L7 loadbalancer. It's missing things like proper retrying, DNS discovery, logging, tracing etc, that might be added in future (:

@bwplotka
@kakkoyun

## Thank you!

Red Hat

https://github.com/kakkoyun/are-you-testing-your-observability

@bwplotka
@kakkoyun

## Reference:

- Thanos
- Prometheus
- Prometheus - client_go
- Prometheus - Histogram
- Prometheus Histograms – Past, Present, and Future
- Roboust Perception Blog
- Why globals are magic
- Red Method
- Promcon 2019 - OpenMatrics
- Gopherize me

@bwplotka
@kakkoyun