Gympass

Golden Metrics

With Prometheus and Grafana

Johnathan Fercher

Gympass

Golden Metrics

With Prometheus and Grafana + Datadog

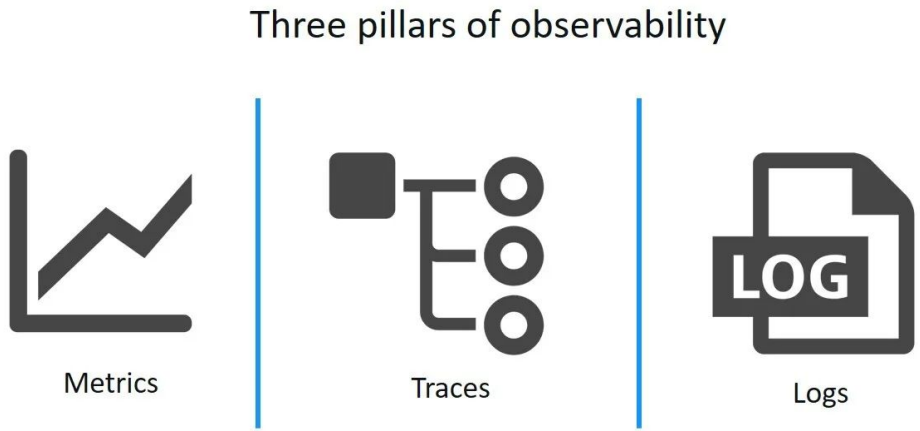
Johnathan Fercher

INDEX

Gympass

Introduction

"Observability is the ability to measure the internal states of a system by examining its outputs."

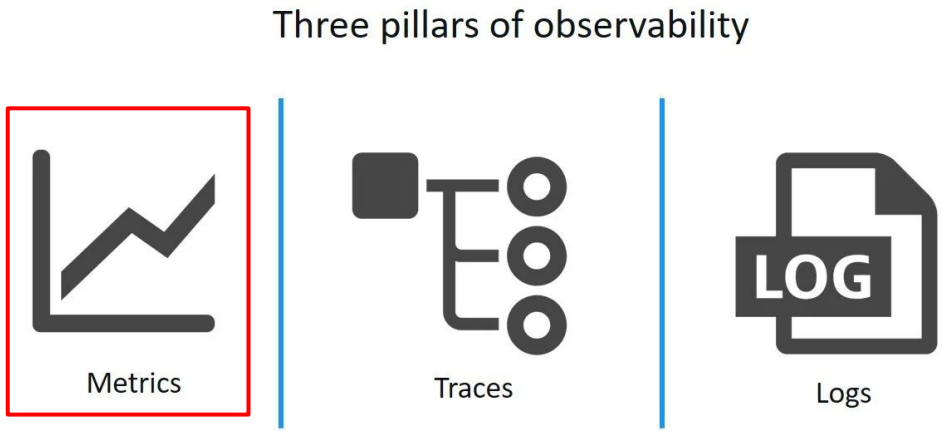


Reference:
https://www.confluent.io/learn/observability/#:~:text=What%20is%20Observability%3F,in%20the%20context%20of%20APM.

Gympass

Introduction

"Observability is the ability to measure the internal states of a system by examining its outputs."



Reference:
https://www.confluent.io/learn/observability/#:~:text=What%20is%20Observability%3F,in%20the%20context%20of%20APM.

Gympass

Golden Signals

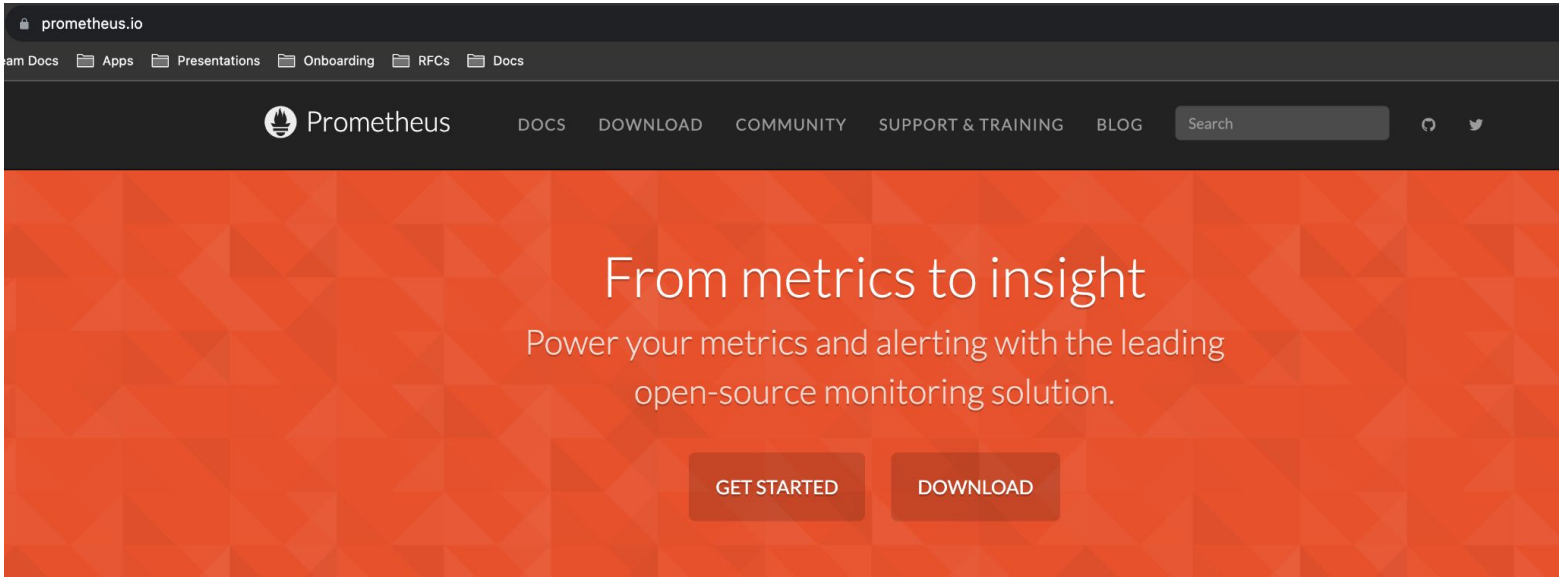
The four golden signals of monitoring are latency, traffic, errors, and saturation. If you can only measure four metrics of your user-facing system, focus on these four.

- Latency
- Traffic
- Errors
- Saturation

Reference: https://sre.google/sre-book/monitoring-distributed-systems/

Gympass

Prometheus



Gympass

Agent

```
func Start() {
        fmt.Println("starting prometheus")
        http.Handle("/metrics", promhttp.Handler())

        go func() {
                http.ListenAndServe(":2112", nil)
        }()
        fmt.Println("started prometheus")
}
```

Reference: https://johnfercher.medium.com/go-observabilidade-739b6d6b649c

Gympass

Agent

```
type Metrics struct {
        // Metric
        Latency float64

        // Labels
        Endpoint            string
        Verb                string
        Pattern             string
        ResponseCode        int
        Failed              bool
        Error               string
        HasAvailabilityError bool
        HasReliabilityError  bool
}
```

Reference: https://johnfercher.medium.com/go-observabilidade-739b6d6b649c

Gympass

Agent

```
func Send(metrics Metrics) {
        labels := map[string]string{
                endpoint:           metrics.Endpoint,
                verb:               metrics.Verb,
                pattern:            metrics.Pattern,
                responseCode:       fmt.Sprintf("%d", metrics.ResponseCode),
                failed:             fmt.Sprintf("%v", metrics.Failed),
                error:              metrics.Error,
                isAvailabilityError: fmt.Sprintf("%v", metrics.HasAvailabilityError),
                isReliabilityError:  fmt.Sprintf("%v", metrics.HasReliabilityError),
        }

        countermetrics.Increment(countermetrics.Metric{
                Name:   endpointRequestCounter,
                Labels: labels,
        })

        histogrammetrics.Observe(histogrammetrics.Metric{
                Name:  endpointRequestLatency,
                Value: float64(metrics.Latency),
                Labels: map[string]string{
                        endpoint: metrics.Endpoint,
                },
        })
}
```

Reference: https://johnfercher.medium.com/go-observabilidade-739b6d6b649c

Agent
localhost/metrics

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 9
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.20.2"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.08112e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.08112e+06
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.447357e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 360
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system m
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 5.707792e+06
```

Reference: https://johnfercher.medium.com/go-observabilidade-739b6d6b649c

Gympass

Types

- Counter
  - **Accumulative values**
  - **Only increase or reset**
  - Ex: 10 request on endpoint X in the last 5 minutes
  - Ex: 5 errors on endpoint Y in the last 10 minutes
- Gauge
  - **Current values**
  - **Can increase or decrease**
  - Ex: PostgreSQL have 5 connections enabled
  - Ex: Errors threshold is in 10% to circuit-breaker
- Histogram
  - **Sample values in buckets**
  - How many milliseconds does an endpoint spent to return on average?
- Summary
  - **Similar to histogram**

Reference: https://prometheus.io/docs/concepts/metric_types/

Grafana

Grafana Labs | Products | Open source | Solutions | Learn | Company | Downloads | Contact us | Sign in

Your LGTM Observability stack

Get there much faster. From dashboards to centralized observability.

✨🧰☁️✨

The (actually useful) free forever plan

Grafana, of course +
10K series Prometheus metrics,
50GB logs, 50GB traces, 50GB profiles,
500VUh k6 testing

Create free account

(No credit card required)

Gympass

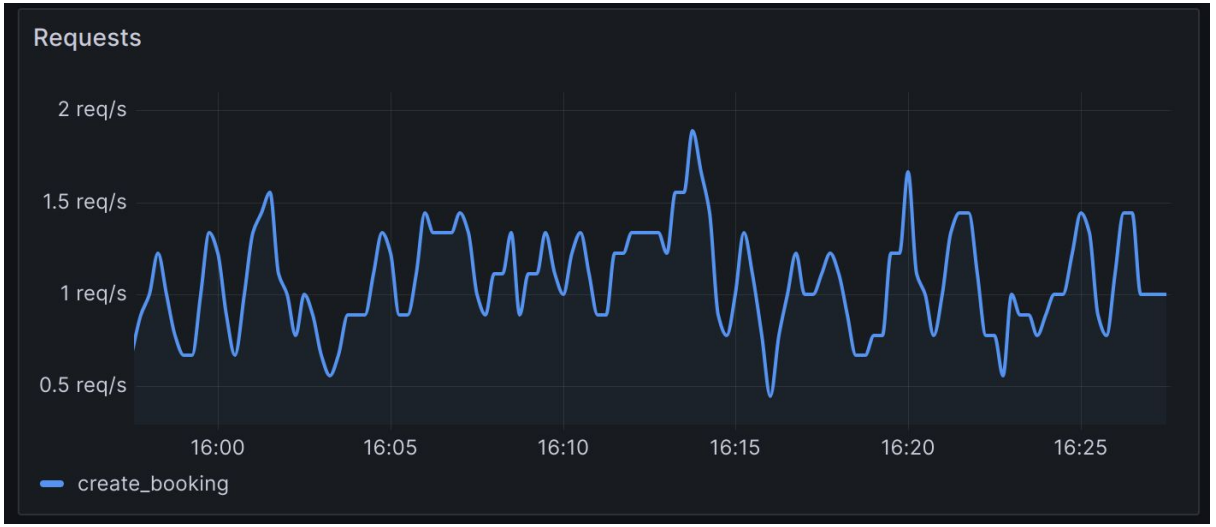
Dashboard

Gympass

Dashboard

Gympass

Requests



Gympass

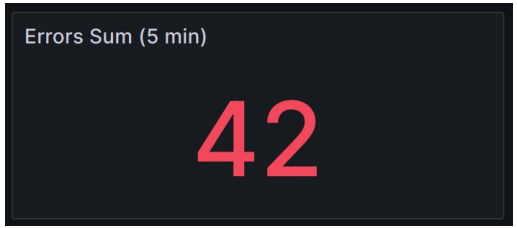
Latency


Latency
700 ms
600 ms
500 ms
400 ms
300 ms
16:00
16:05
16:10
16:15
16:20
16:25
create_booking
2023-11-07 16:23
create_produ


Latency (5 min)
470 ms

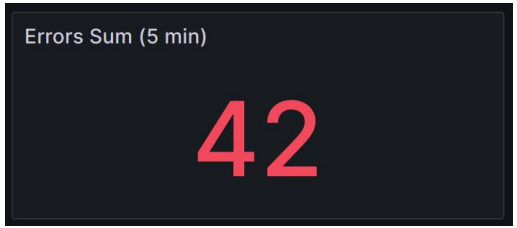
Errors


Errors
15
12.5
10
7.5
5
2.5
0
16:00
16:05
16:10
16:15
16:20
16:25
create_booking


Errors Sum (5 min)
42

Errors

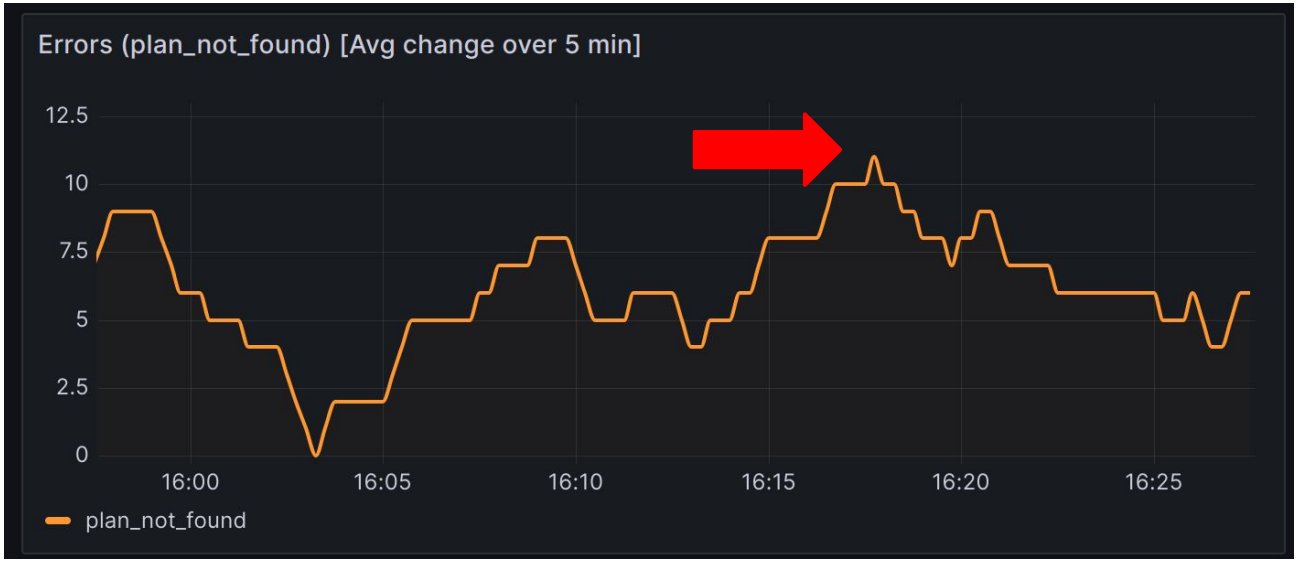

Errors
15
12.5
10
7.5
5
2.5
0
16:00
16:05
16:10
16:15
16:20
16:25
create_booking


Errors Sum (5 min)
42

Errors


Errors (5 min)
booking_already_validated
booking_is_not_valid
booking_not_found
plan_not_found
plan_doesnt_allow_booking

Errors

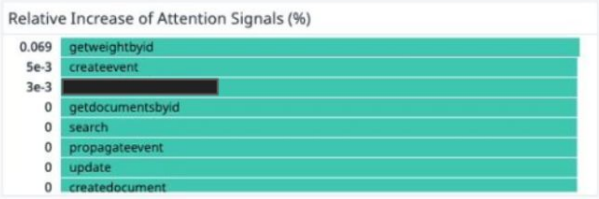

Errors (plan_not_found) [Avg change over 5 min]
12.5
10
7.5
5
2.5
0
16:00
16:05
16:10
16:15
16:20
16:25
plan_not_found

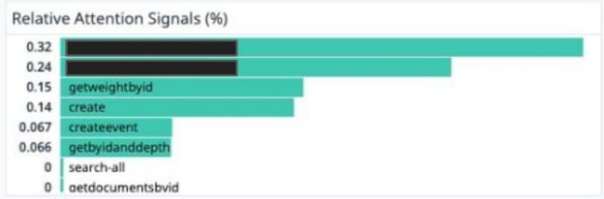
Datadog

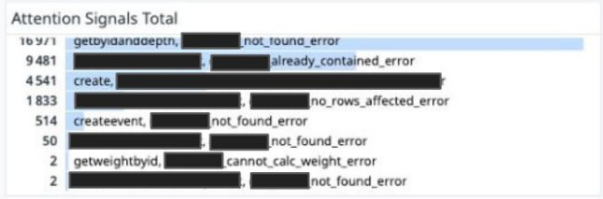

Relative Increase of Errors (%)
0.55 getweightbyid
2e-3 getcountbyid
0 update
0 createdocument
0 delete
0 propagateevent
0 getdocumentsbyid
0 active


Relative Errors (%)
10.49 getweightbyid
5.94 [redacted]
1.36 getdetailsbyid
0.13 search-all
0.074 search
0.048 [redacted]
0.043 createevent
0.023 create


Errors Total
30 create, internal_server_error
2 create, [redacted]_open_transaction_error
1 [redacted], [redacted]commit_transaction_error
89 search, [redacted]execute_query_error
1 347 search, [redacted]search_count_[redacted]error
8 createevent, [redacted]execute_query_error
2 033 getbyidanddepth, [redacted]execute_query_error
1 create, [redacted]could not read bytes error


Relative Increase of Attention Signals (%)
0.069 getweightbyid
5e-3 createevent
3e-3 [redacted]
0 getdocumentsbyid
0 search
0 propagateevent
0 update
0 createdocument


Relative Attention Signals (%)
0.32 [redacted]
0.24 [redacted]
0.15 getweightbyid
0.14 create
0.067 createevent
0.066 getbyidanddepth
0 search-all
0 getdocumentsbyid

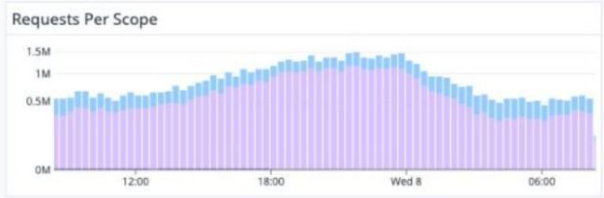

Attention Signals Total
16 971 getbyidanddepth, [redacted]not_found_error
9 481 [redacted], [redacted]already_contained_error
4 541 create, [redacted]
1 833 [redacted], [redacted]no_rows_affected_error
514 createevent, [redacted]not_found_error
50 [redacted], [redacted]not_found_error
2 getweightbyid, [redacted]cannot_calc_weight_error
2 [redacted], [redacted]not_found_error


Relative Increase of Latency
0 %
25 %
407.5 [redacted] +30.5 %
36.1 search +15.8 %
127.6 createevent +8.3 %
13.5 search-all +7.4 %
4.3 uniquesearch +6.2 %
15.9 getbyidanddepth +5.7 %


Absolute AVG Latency
Milliseconds
407.5 [redacted]
311.3 getweightbyid
235.4 getdetailsbyid
222.7 active
127.6 createevent
94.6 update
72.8 [redacted]
67.5 create


Requests Total
25.67M getbyidanddepth
17.53M propagateevent
4.27M uniquesearch
4.01M [redacted]
3.78M getcountbyid
3.62M create
1.97M search
0.89M createevent


Relative Increase of Requests
0 %
25 %
0.8 getweightbyid +45 %
12.4 search +20.1 %
18.8 [redacted] +18.8 %
41.8 getcountbyid +17.8 %
0.9 update +17.5 %
1.2k propagateevent +14.2 %


Requests Per Scope
1.5M
1M
0.5M
0M
12:00
18:00
Wed 8
06:00

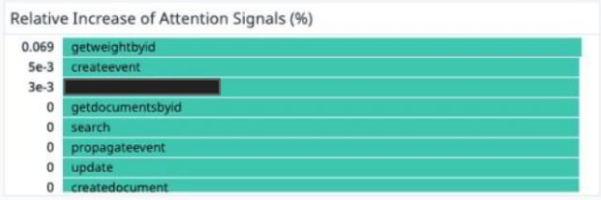

Total Requests
63.17M

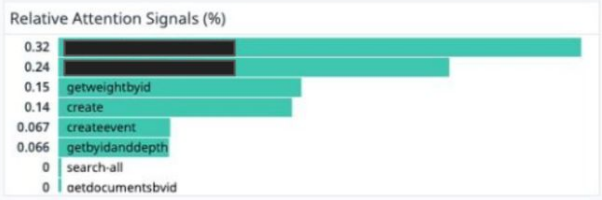
Datadog

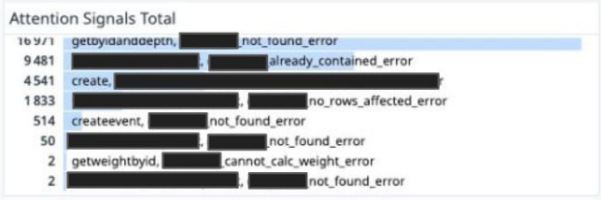

Relative Increase of Errors (%)
0.55 getweightbyid
2e-3 getcountbyid
0 update
0 createdocument
0 delete
0 propagateevent
0 getdocumentsbyid
0 active


Relative Errors (%)
10.49 getweightbyid
5.94 [redacted]
1.36 getdetailsbyid
0.13 search-all
0.074 search
0.048 [redacted]
0.043 createevent
0.023 create

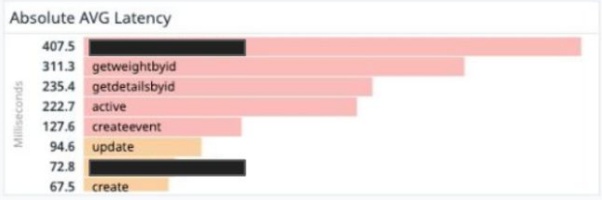

Errors Total
30 create, internal_server_error
2 create, [redacted] open_transaction_error
1 [redacted], [redacted] commit_transaction_error
89 search, [redacted] execute_query_error
1 347 search, [redacted] search_count [redacted] error
8 createevent, [redacted] execute_query_error
2 033 getbyidanddepth, [redacted] execute_query_error
1 create, [redacted] could not read bytes error


Relative Increase of Attention Signals (%)
0.069 getweightbyid
5e-3 createevent
3e-3 [redacted]
0 getdocumentsbyid
0 search
0 propagateevent
0 update
0 createdocument

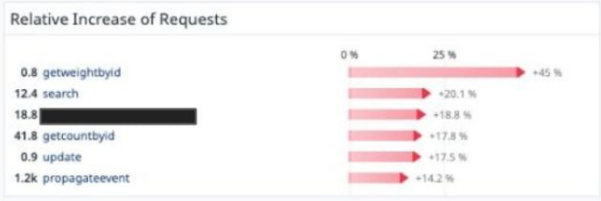

Relative Attention Signals (%)
0.32 [redacted]
0.24 [redacted]
0.15 getweightbyid
0.14 create
0.067 createevent
0.066 getbyidanddepth
0 search-all
0 getdocumentsbyid

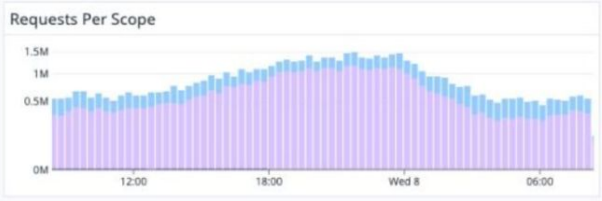

Attention Signals Total
16 971 getbyidanddepth, [redacted] not_found_error
9 481 [redacted], [redacted] already_contained_error
4 541 create, [redacted]
1 833 [redacted], [redacted] no_rows_affected_error
514 createevent, [redacted] not_found_error
50 [redacted], [redacted] not_found_error
2 getweightbyid, [redacted] cannot_calc_weight_error
2 [redacted], [redacted] not_found_error

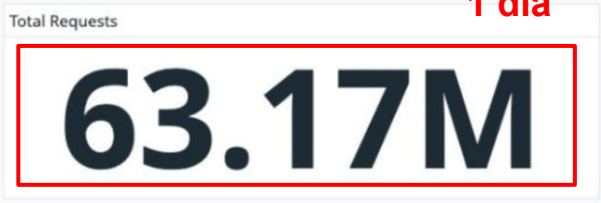

Relative Increase of Latency
0 %
25 %
407.5 [redacted] +30.5 %
36.1 search +15.8 %
127.6 createevent +8.3 %
13.5 search-all +7.4 %
4.3 uniquesearch +6.2 %
15.9 getbyidanddepth +5.7 %


Absolute AVG Latency
Milliseconds
407.5 [redacted]
311.3 getweightbyid
235.4 getdetailsbyid
222.7 active
127.6 createevent
94.6 update
72.8 [redacted]
67.5 create

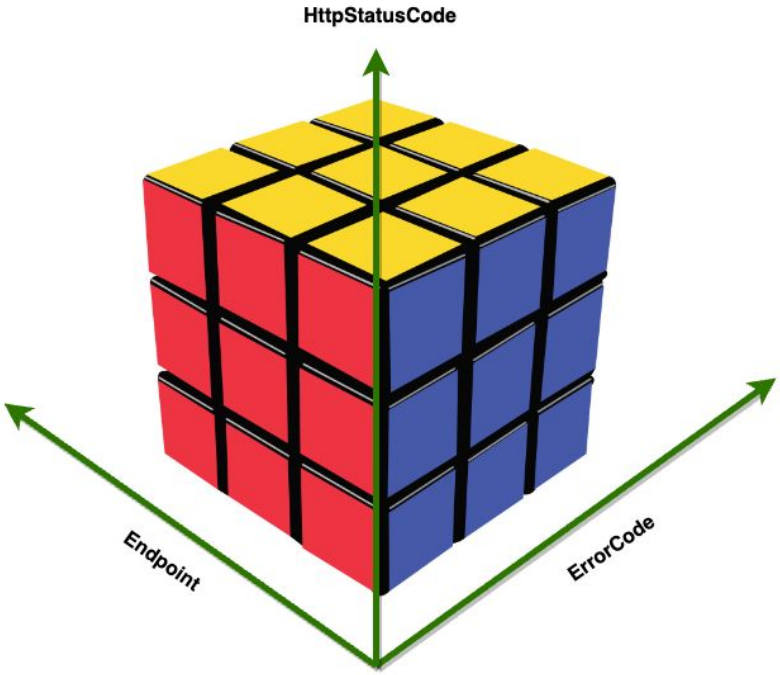

Requests Total
25.67M getbyidanddepth
17.53M propagateevent
4.27M uniquesearch
4.01M [redacted]
3.78M getcountbyid
3.62M create
1.97M search
0.89M createevent

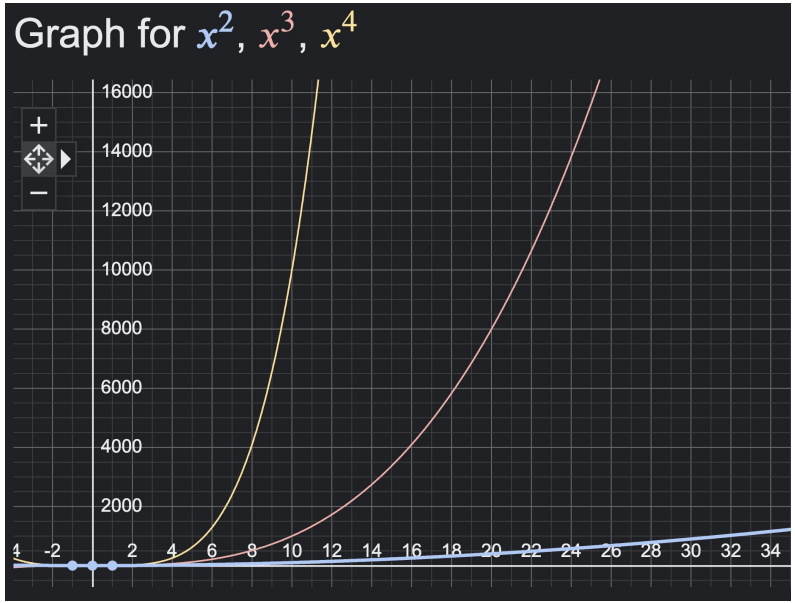

Relative Increase of Requests
0 %
25 %
0.8 getweightbyid +45 %
12.4 search +20.1 %
18.8 [redacted] +18.8 %
41.8 getcountbyid +17.8 %
0.9 update +17.5 %
1.2k propagateevent +14.2 %


Requests Per Scope
1.5M
1M
0.5M
0M
12:00
18:00
Wed 8
06:00

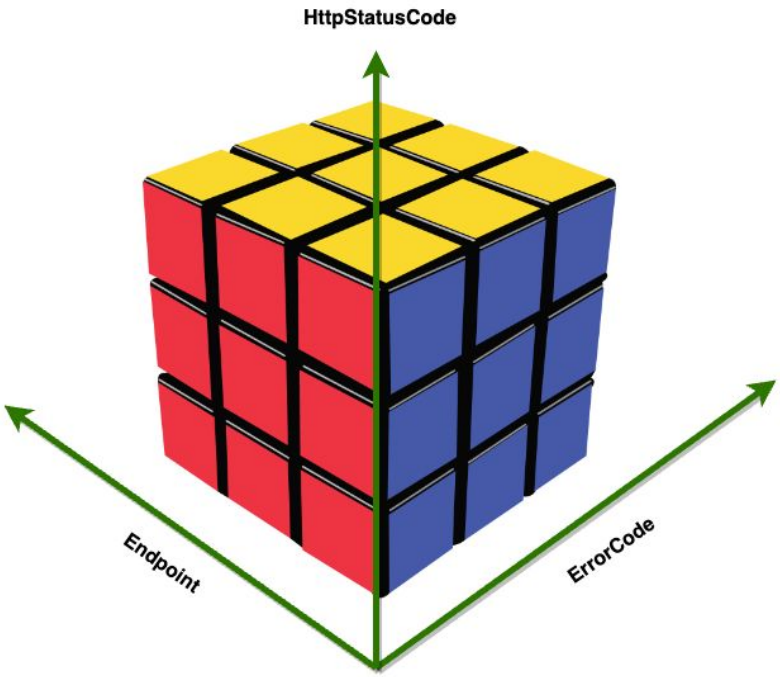

Total Requests
1 dia
63.17M

Is This Expensive?



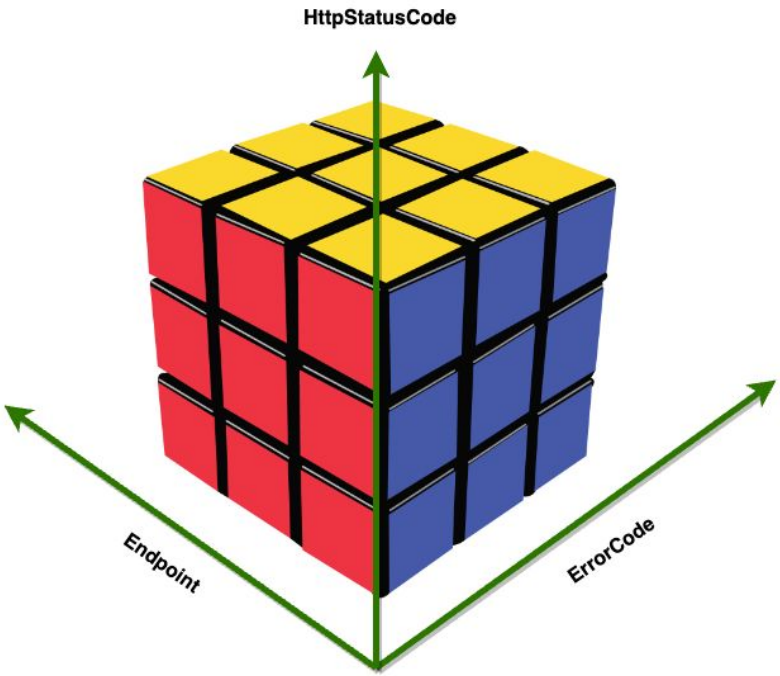


Is This Expensive?



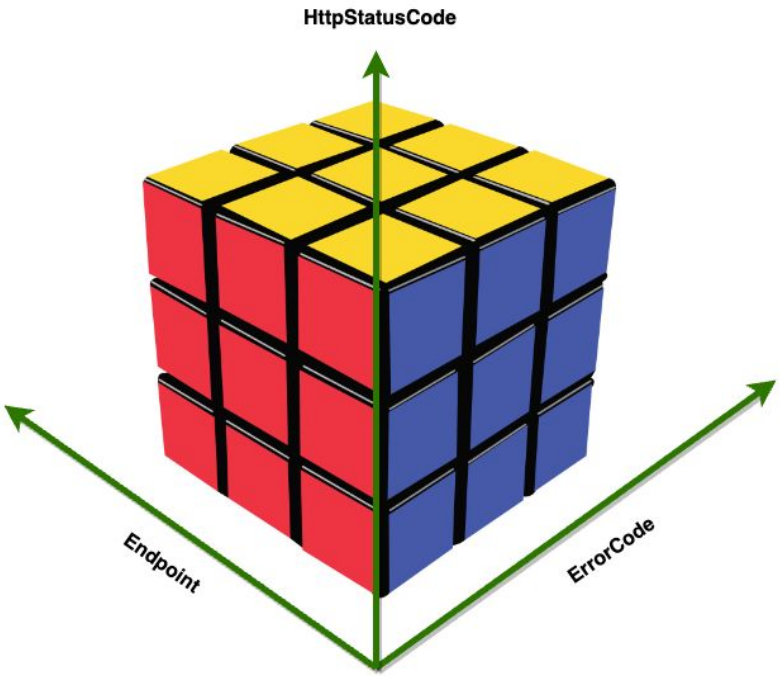
Cost = Space + Write + Read

Is This Expensive?



Cost = Space + Write + Read

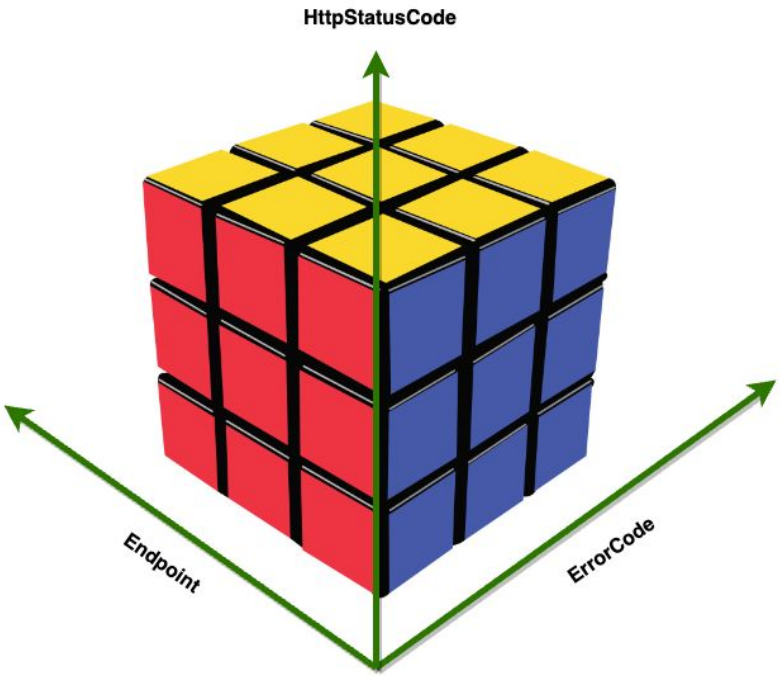
Is This Expensive?



Cost = Space + Write + Read

Space = Len(Endpoint) * Len(ErrorCode) * Len(HttpStatusCode) * …

Is This Expensive?

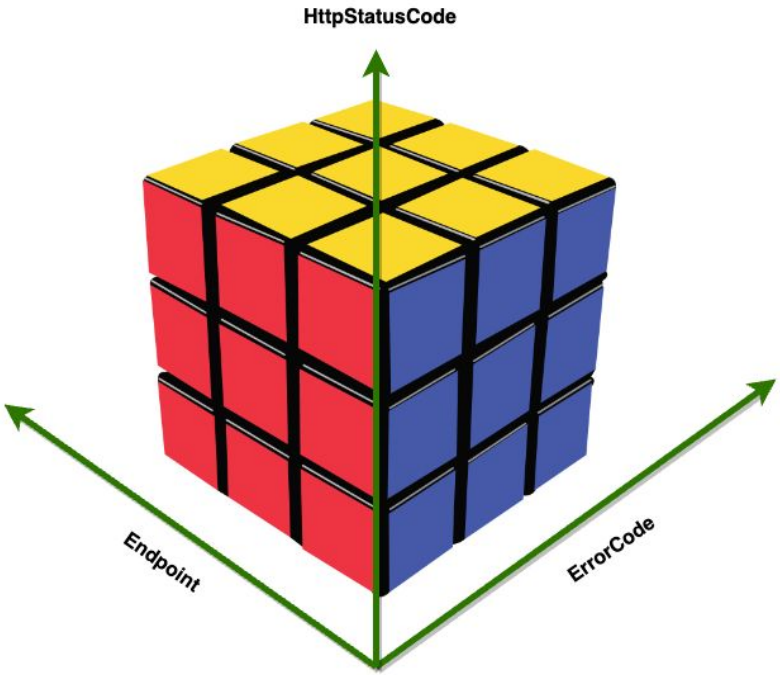


Cost = Space + Write + Read

Space = Len(Endpoint) * Len(ErrorCode) * Len(HttpStatusCode) * …

Small API
Space = 12 endpoints * 25 error codes * 10 http status
Space = 3.000
(This would not increase by time)

Is This Expensive?



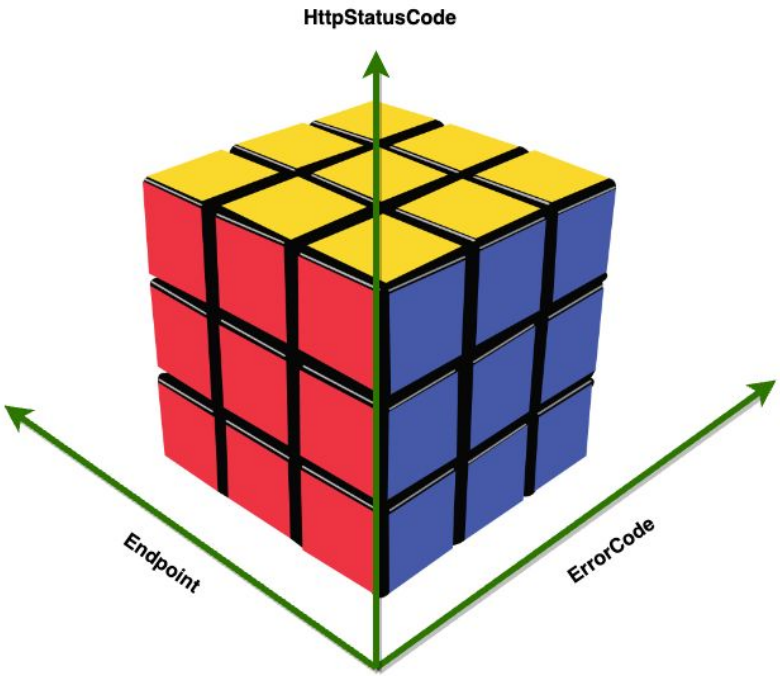
Cost = Space + Write + Read

Space = Len(Endpoint) * Len(ErrorCode) * Len(HttpStatusCode) * …

Small API
Space = 12 endpoints * 25 error codes * 10 http status
Space = 3.000
(This would not increase by time)

Big API
Space = 50 endpoints * 100 error codes * 10 http status
Space = 50.000
(This would not increase by time)

Is This Expensive?



Cost = Space + Write + Read

Space = Len(Endpoint) * Len(ErrorCode) * Len(HttpStatusCode) * …

Small API
Space = 12 endpoints * 25 error codes * 10 http status
Space = 3.000
(This would not increase by time)

Big API
Space = 50 endpoints * 100 error codes * 10 http status
Space = 50.000
(This would not increase by time)

Wrong Usage (Small API)
Space = 12 endpoints * **25000 booking_id** * 10 http status
Space = 3.000.000
(This would increase **daily** by time)

Gympass

Thank You