

How to monitor your Golang App with Prometheus (and Grafana)?



Wojciech Barczyński - [Spacelift.io](#)
15 June 2023

About me

- Lead Engineering at [Spacelift.io](#)
- Background:
Developer and System Engineer
- Before:
Codility.com, Hypatos.ai, Lykehq (RocketInternet)

Point of view

- Software Engineer,
- startups && fast-moving environment,
- Infra and platform should just work,
- Infra and platform ~ invisible.

Recommendation

- Start with monitoring;
- Metrics? Start with Up/Down, than RED and USE;
- Drop-in solution Prometheus;
- How to start? With this demo app.

Why?

Monolit ;)



Why?
Microservices ;)



Observability

- Monitoring
- Logging
- Tracing

Observability

	Metrics	Logging	Tracing
CapEx	Medium	Low	High
OpEx	Low	High	Medium
Reaction	High	Medium	Low
Investigation	Low	Medium	High

Go for Industrial Programming by Peter Bourgon.

Not a silver-bullet

but:

- Easy to setup
- Immediately value

Surprisingly: the last one implemented

Centralized Logging

- Like a debugging vs testing
- Post-mortem
- Hard to find the needle
- High TCO

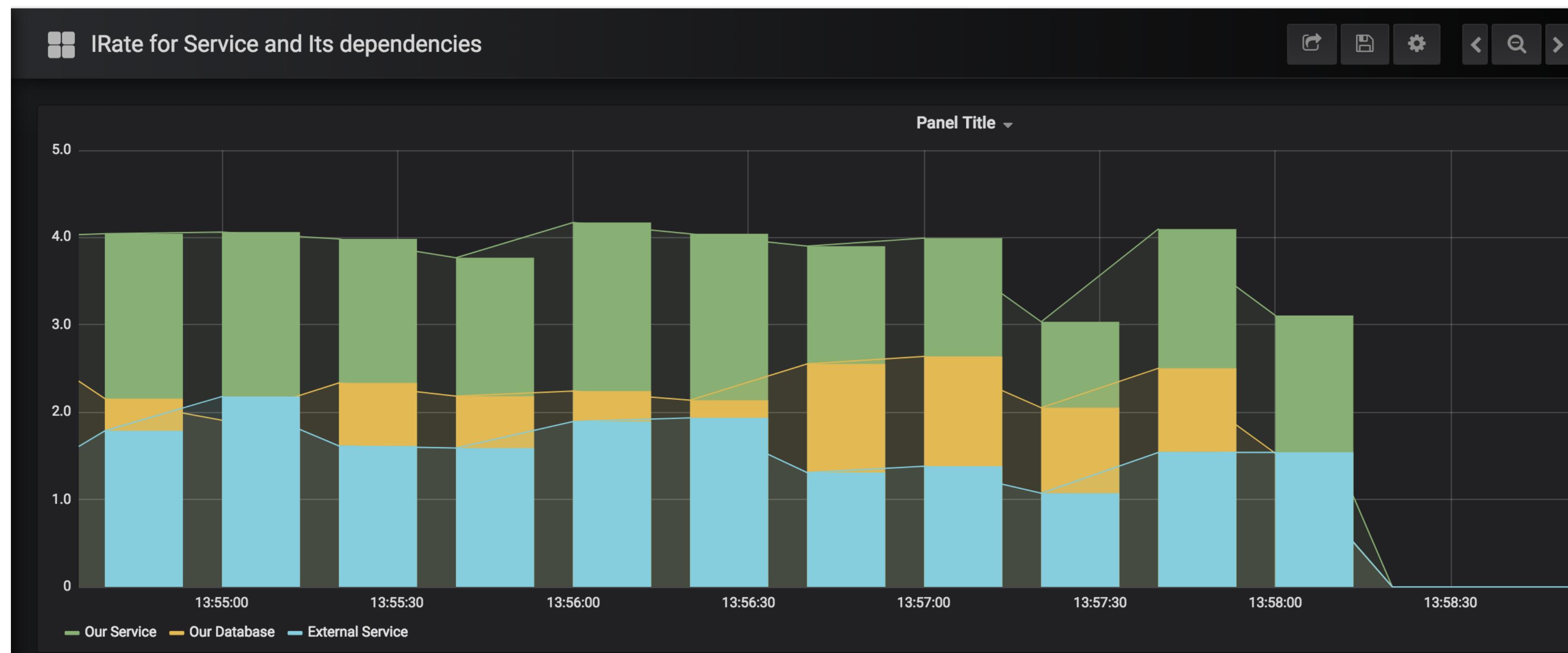
Monitoring

- Numbers
- Trends
- Dependencies
- + Actions

Metric

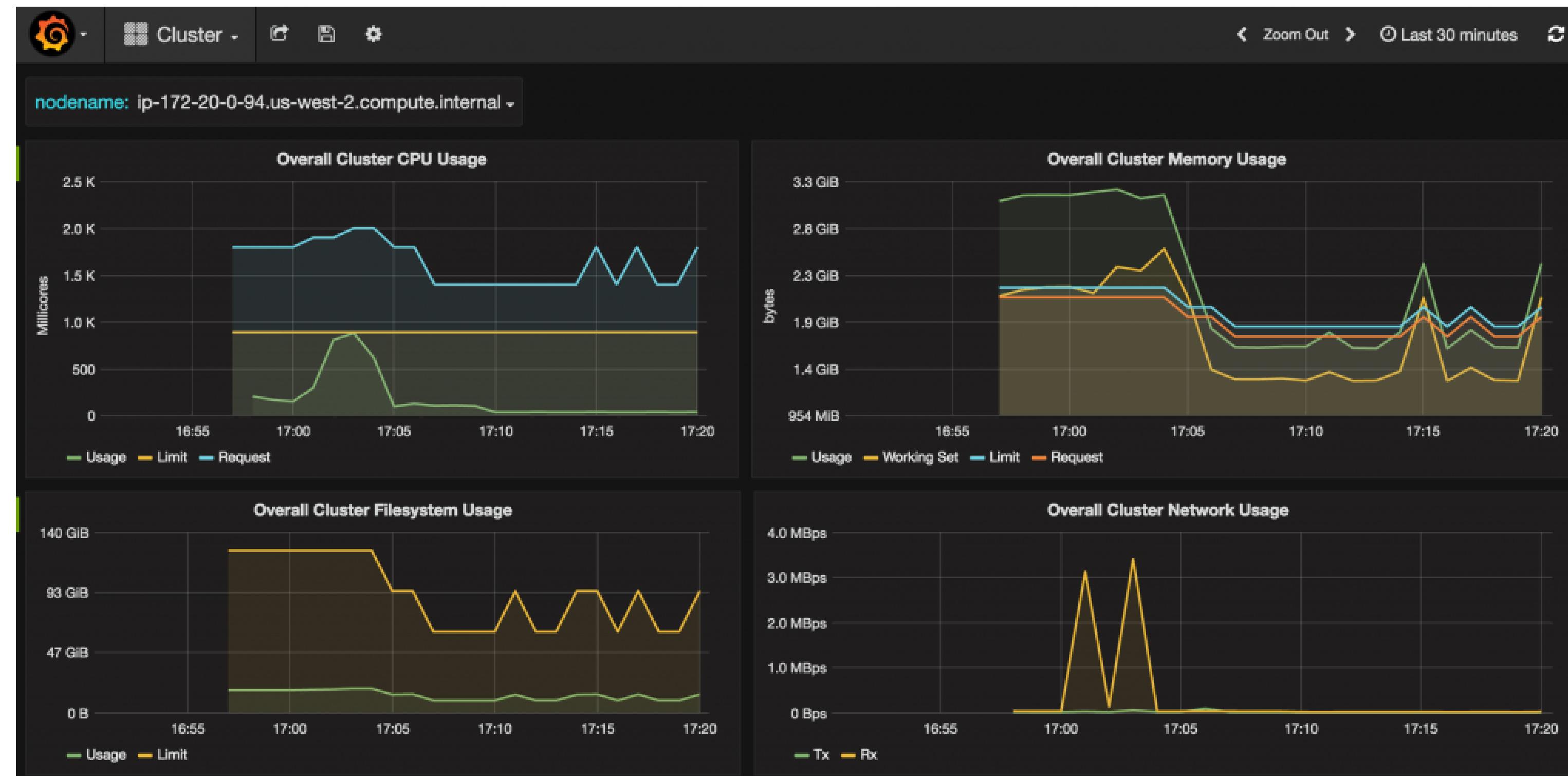
Name	Label	Value
traefik_requests_total	code="200", method="GET"	3001

Monitoring



Demo app

Monitoring



Example from [couchbase blog](#)

How to find the right metric?

How to find the right metric?

- USE
- RED

USE

Utilization the average time that the resource was busy servicing work

Saturation extra work which it can't service, often queued

Errors the count of error events

Documented and Promoted by [Berdan Gregg](#)

USE

- Utilization: as a percent over a time interval: "one disk is running at 90% utilization".
- Saturation:
- Errors:

USE

- Utilization:
- Saturation: as a queue length. eg, "the CPUs have an average run queue length of four".
- Errors:

USE

- Utilization:
- Saturation:
- Errors: scalar counts. eg, "this network interface drops packages".

USE

- traditionally more instance oriented
- still useful in the microservices world

RED

Rate

How busy is your service?

Error

Errors

Duration

What is the latency of my service?

[Tom Wilkie's guideline for instrumenting applications.](#)

RED

- **Rate** - how many requests per seconds handled
- **Error**
- **Duration (distribution)**

RED

- Rate
- Error - how many request per seconds handled we failed
- Duration

RED

- Rate
- Error
- Duration - how long the requests took

RED

- Follow Four Golden Signals by Google SREs [1]
- Focus on what matters for end-users, **their experience**
- Business metrics

[1] Latency, Traffic, Errors, Saturation ([src](#))

RED

Not recommended for:

- batch-oriented
- streaming services

Prometheus



What prometheus is?

- Aggregation of time-series data
- Not an event-based system

Prometheus stack

- Prometheus - collect
- Alertmanager - alerts
- Grafana - visualize

Prometheus

- Plain text,
- Metrics collected over HTTP [metrics/](#),
- Pull model [1],
- PromQL,
- SDKs, Exporters and support in the most of CloudNative-projects
- alertManager

[1] See *scrape time*, push-mode possible

go gc metrics

Endpoint metrics/*:

```
# HELP go_gc_duration_seconds A summary of the GC invocation d
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 9.01e-05
go_gc_duration_seconds{quantile="0.25"} 0.000141101
go_gc_duration_seconds{quantile="0.5"} 0.000178902
go_gc_duration_seconds{quantile="0.75"} 0.000226903
go_gc_duration_seconds{quantile="1"} 0.006099658
go_gc_duration_seconds_sum 18.749046756
go_gc_duration_seconds_count 89273
```

Sometimes on a dedicated port, e.g., 9090

Service metrics

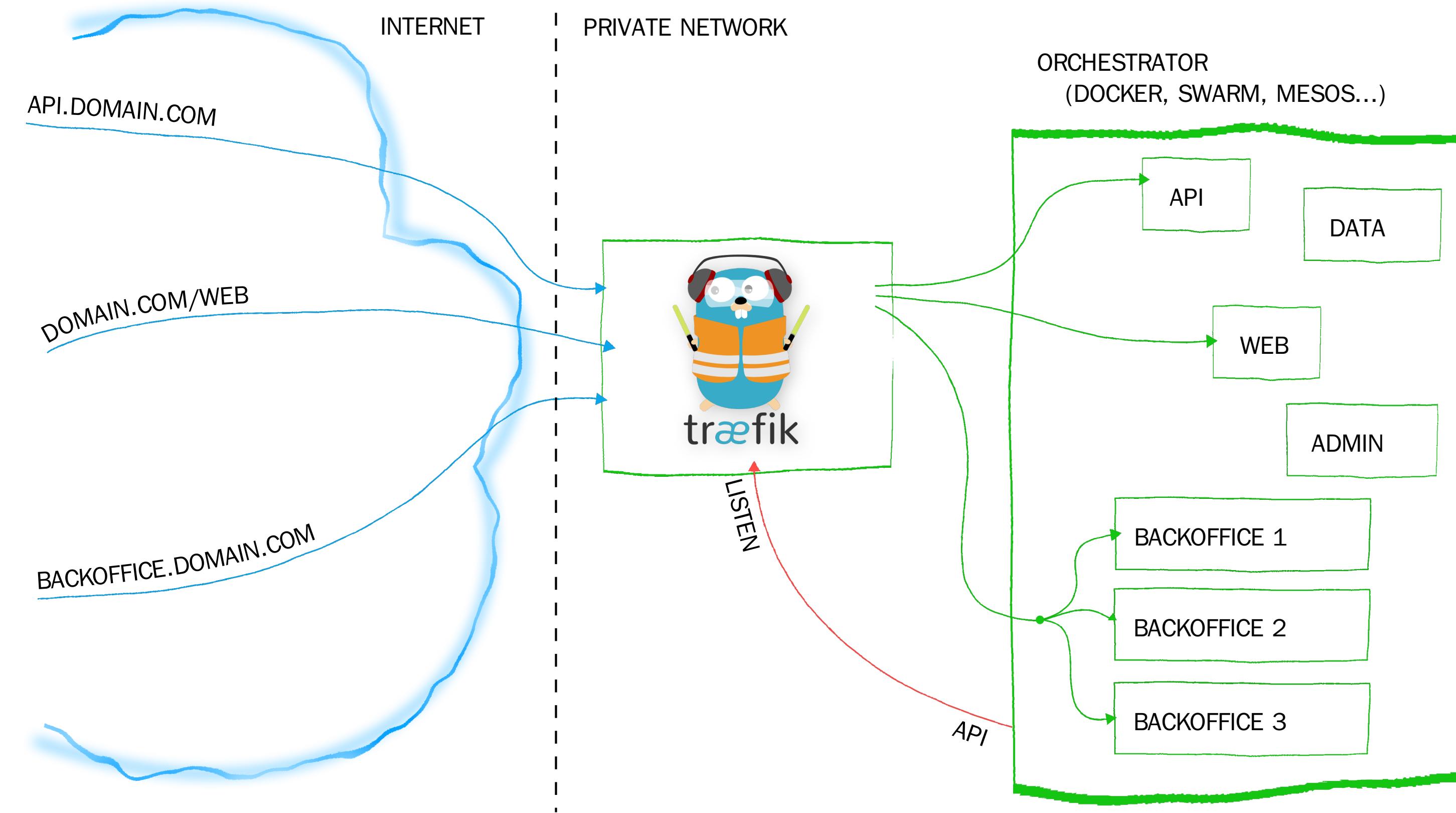
```
# HELP order_mgmt_audit_duration_seconds Multiprocess metric
# TYPE order_mgmt_audit_duration_seconds summary
order_mgmt_audit_duration_seconds_count{status_code="200"} 41.
order_mgmt_audit_duration_seconds_sum{status_code="200"} 27.44
order_mgmt_audit_duration_seconds_count{status_code="500"} 1.0
order_mgmt_audit_duration_seconds_sum{status_code="500"} 0.716
# HELP order_mgmt_duration_seconds Multiprocess metric
# TYPE order_mgmt_duration_seconds summary
order_mgmt_duration_seconds_count{method="GET",path="/complex"}
order_mgmt_duration_seconds_sum{method="GET",path="/complex",s}
order_mgmt_duration_seconds_count{method="GET",path="/",status_c}
order_mgmt_duration_seconds_sum{method="GET",path="/",status_c}
order_mgmt_duration_seconds_count{method="GET",path="/complex"}
order_mgmt_duration_seconds_sum{method="GET",path="/complex",s}
```

Exporters and SDKs

- Postgresql
- Mysql
- Rabbitmq
- ...
- Blackbox exporter
- SDKs: go, ..., Java

[Exporters and integrations](#)

Cloud-native projects integration



traefik: --web.metrics.prometheus

Prometheus

Scraping configuration (pull model):

```
global:  
  scrape_interval: 2s  
  scrape_timeout: 2s  
  evaluation_interval: 2s  
  external_labels:  
    environment: production
```

Prometheus

Scraping configuration:

scrape interval to 2x the amount of time it takes for
your application to generate new metric.

PromQL

```
irate(order_mgmt_duration_seconds_bucket[2m])  
  
sum(irate(order_mgmt_duration_seconds_count{job=~".*"}[1m]))  
    by (status_code)  
  
redict_linear()  
holt_winters()
```

more during the demo

Recording Rules

For frequently needed and expensive expressions:

```
groups:  
  - name: example  
    rules:  
      - record: code:prometheus_http_requests_total:sum  
        expr: sum by (code) (prometheus_http_requests_total)
```

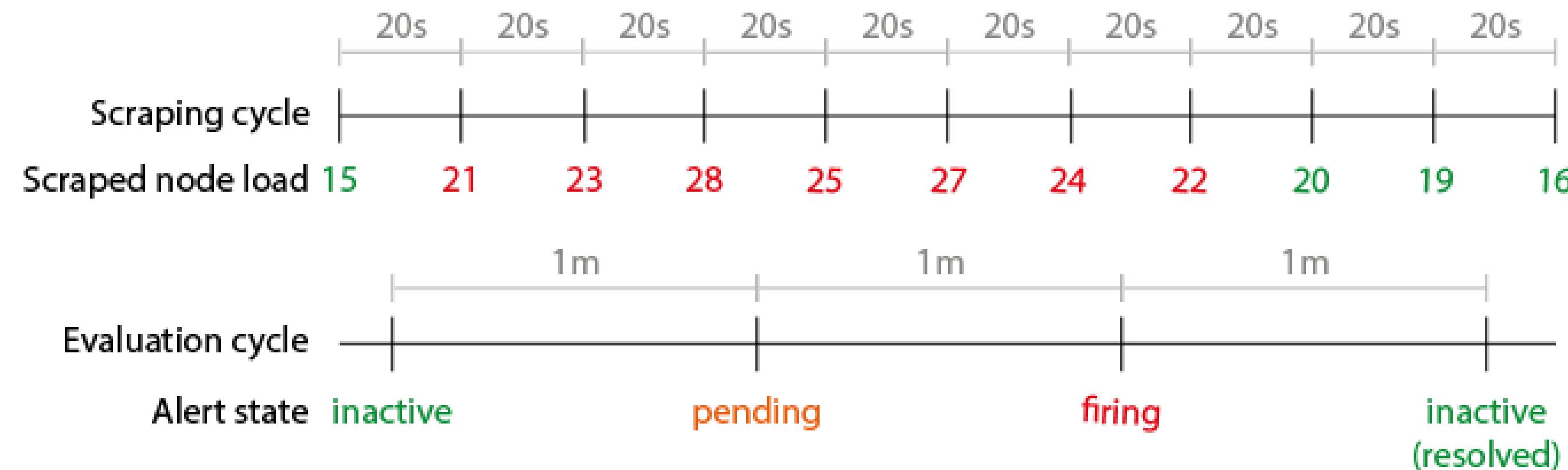
[docs](#)

Alerting rules

```
- alert: ProductionInstanceDown
  expr: up{env = 'production'} == 0
  for: 1m
  labels:
    severity: opsgenie
  annotations:
    summary: "Instance {{ $labels.instance }} of {{ $labels.
    description: "{{ $labels.instance }} of job {{ $labels.j
  }
```

Alerting rules

Understanding delays on alerting (src)



Alertmanager - dispatcher

```
route:  
  group_by: ['environment']  
  receiver: email-general  
  routes:  
    - match:  
        environment: production  
        severity: opsgenie  
        receiver: opsgenie-general  
    - match_re:  
        severity: email  
        receiver: email-general
```

Metric types

Metric types

- Counter - just up
- Gauge - up/down
- Histogram
- Summary

Histogram

http_request_duration_seconds{status_code="200"}

{le="0.1"}	2
{le="0.3"}	4
{le="1.2"}	100
{le="+Inf"}	2229
<u>_sum</u>	
<u>_count</u>	2229

le = lower or equal ([tutorial](#))

Histogram

To quantile:

```
histogram_quantile(  
    0.9, order_mgmt_audit_duration_seconds_bucket)
```

Summary

- Cheaper but lose more data,
 - Computed at the client,
 - example, `go_gc_duration_seconds`.

Summary

http_request_duration_seconds	
{quantile="0.5"}	4 (seconds)
{quantile="0.9"}	5
_sum	9
_count	3

Histogram / Summary:

- Latency of services
- Request or Request size

Histograms recommended ([docs](#))

RED

Histogram + PromQL:

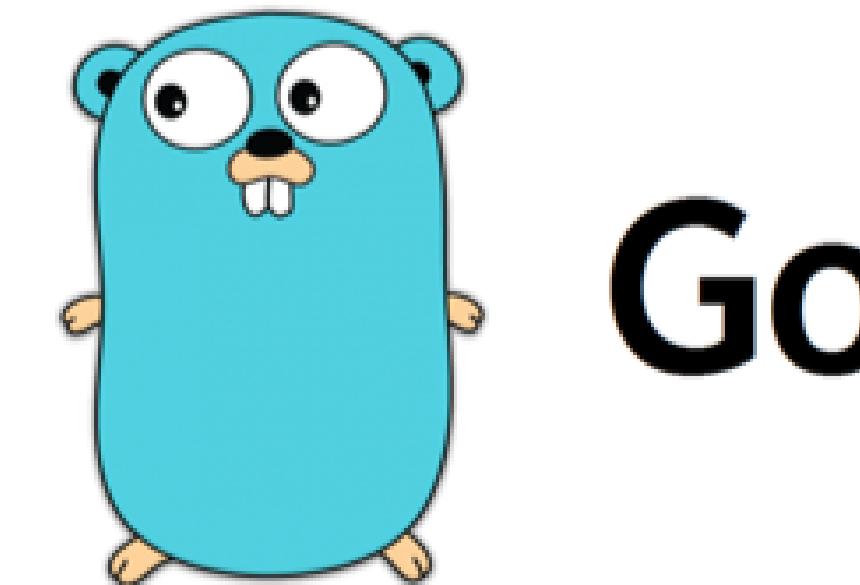
```
sum(irate(order_mgmt_duration_seconds_count  
{job=~".*"}[1m])) by (status_code)
```

Metric and label naming

Best practices on metric names:

- service name is your prefix `user_`,
- state the base unit `_seconds` and `_bytes`,
- notice - `http_request_duration_seconds`.

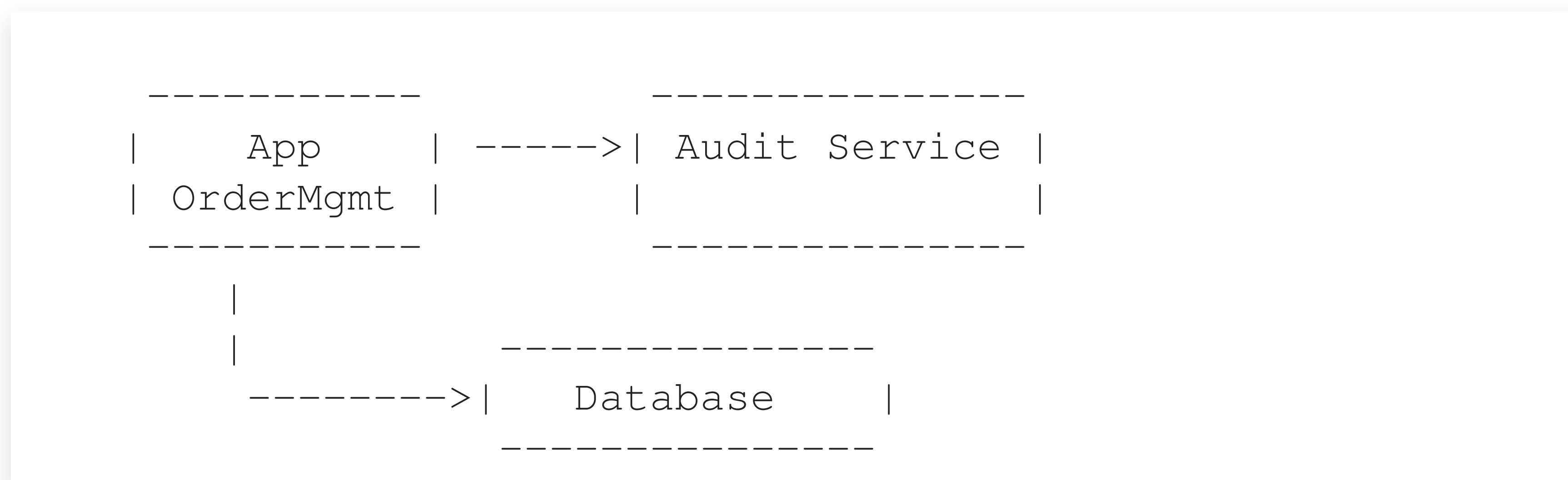
Prometheus + Golang



Golang client

- github.com/prometheus/client_golang

Demo: Simple REST service



Demo

- <http://127.0.0.1:8080/hello> - service
- <http://127.0.0.1:8080/metrics>
- <http://127.0.0.1:9090> - prometheus
- <http://127.0.0.1:3000> - grafana (admin:secret)
- <http://127.0.0.1:9093> - alertmanager

Demo

```
▶ demo ⚡ make start
```

```
▶ demo ⚡ docker ps
```

CONTAINER ID	IMAGE	PORTS
5f824d1bc789	grafana/grafana:10.0.0	0.0.0.0:3000->3
d681a414a8b6	prom/prometheus:v2.44.0	0.0.0.0:9090->
ea0d9233e159	prom/alertmanager:v0.25.0	0.0.0.0:9093->9
732c59fb3753	order-manager	0.0.0.0:8080->8080/tcp

Demo: generate calls

• demo

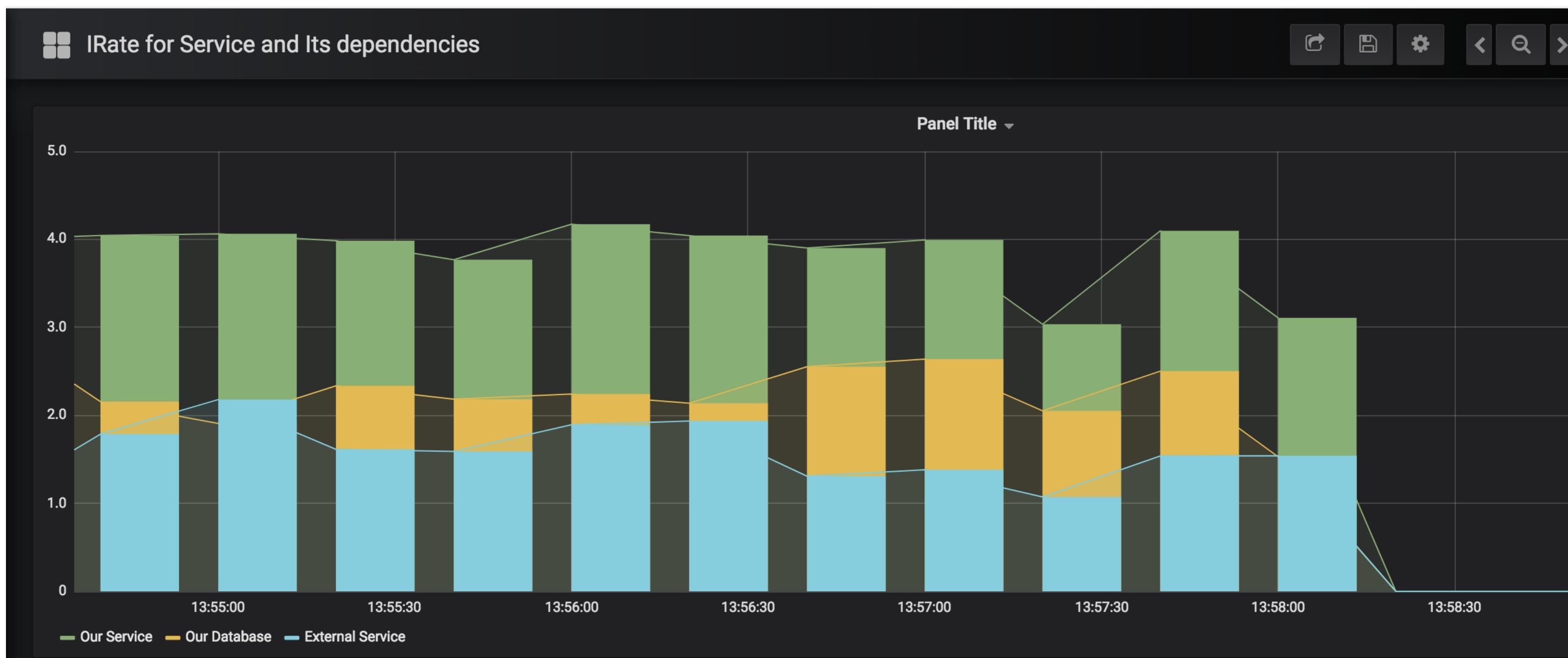


make srv_wrk_random

With error injection

HELP order_mgmt_database_duration_seconds Multiprocess metric
TYPE order_mgmt_database_duration_seconds histogram
order_mgmt_database_duration_seconds_sum{sql_state="0",status_code="0"} 338.9096608161926
order_mgmt_database_duration_seconds_sum{sql_state="HY000",status_code="1001"} 92.89293241500854
order_mgmt_database_duration_seconds_bucket{le="0.1",sql_state="0",status_code="0"} 72.0
order_mgmt_database_duration_seconds_bucket{le="0.25",sql_state="0",status_code="0"} 168.0
order_mgmt_database_duration_seconds_bucket{le="0.5",sql_state="0",status_code="0"} 287.0
order_mgmt_database_duration_seconds_bucket{le="0.75",sql_state="0",status_code="0"} 459.0
order_mgmt_database_duration_seconds_bucket{le="0.9",sql_state="0",status_code="0"} 573.0
order_mgmt_database_duration_seconds_bucket{le="1.0",sql_state="0",status_code="0"} 646.0
order_mgmt_database_duration_seconds_bucket{le="2.5",sql_state="0",status_code="0"} 648.0
order_mgmt_database_duration_seconds_bucket{le="+Inf",sql_state="0",status_code="0"} 648.0
order_mgmt_database_duration_seconds_count{sql_state="0",status_code="0"} 648.0
order_mgmt_database_duration_seconds_bucket{le="0.1",sql_state="HY000",status_code="1001"} 24.0
order_mgmt_database_duration_seconds_bucket{le="0.25",sql_state="HY000",status_code="1001"} 86.0
order_mgmt_database_duration_seconds_bucket{le="0.5",sql_state="HY000",status_code="1001"} 134.0
order_mgmt_database_duration_seconds_bucket{le="0.75",sql_state="HY000",status_code="1001"} 190.0
order_mgmt_database_duration_seconds_bucket{le="0.9",sql_state="HY000",status_code="1001"} 217.0
order_mgmt_database_duration_seconds_bucket{le="1.0",sql_state="HY000",status_code="1001"} 225.0
order_mgmt_database_duration_seconds_bucket{le="2.5",sql_state="HY000",status_code="1001"} 225.0
order_mgmt_database_duration_seconds_bucket{le="+Inf",sql_state="HY000",status_code="1001"} 225.0
order_mgmt_database_duration_seconds_count{sql_state="HY000",status_code="1001"} 225.0
HELP order_mgmt_audit_duration_seconds Multiprocess metric
TYPE order_mgmt_audit_duration_seconds summary
order_mgmt_audit_duration_seconds_count{status_code="200"} 490.0
order_mgmt_audit_duration_seconds_sum{status_code="200"} 231.30700039863586
order_mgmt_audit_duration_seconds_count{status_code="500"} 158.0
order_mgmt_audit_duration_seconds_sum{status_code="500"} 99.42281174659729
HELP order_mgmt_duration_seconds Multiprocess metric
TYPE order_mgmt_duration_seconds summary
order_mgmt_duration_seconds_count{method="GET",path="/complex",status_code="200"} 490.0
order_mgmt_duration_seconds_sum{method="GET",path="/complex",status_code="200"} 471.54110622406006
order_mgmt_duration_seconds_count{method="GET",path="/complex",status_code="503"} 383.0
order_mgmt_duration_seconds_sum{method="GET",path="/complex",status_code="503"} 291.444673538208

Grafana



Prometheus

The screenshot shows the Prometheus web interface running at `127.0.0.1:9090/graph`. The browser tab bar includes tabs for Prometheus, Alertmanager, Grafana, and other local services like UniFi controllers.

The main interface has a dark header with navigation links: Prometheus, Alerts, Graph, Status, and Help. A sidebar on the left contains a checkbox for "Enable query history" and a text input field labeled "Expression (press Shift+Enter for newlines)". Below this is a button labeled "Execute" followed by a dropdown menu with the placeholder "- insert metric at cursor -".

The central area features a "Graph" tab selected, showing a table with two columns: "Element" and "Value". The table displays the message "no data". To the right of the table is a "Remove Graph" button. At the bottom left is an "Add Graph" button.

Prometheus

The screenshot shows the Prometheus web interface at `127.0.0.1:9090/graph`. The top navigation bar includes tabs for Prometheus, Alertmanager, Grafana, and New Tab. Below the navigation is a toolbar with icons for Apps, Squad Warsaw Bo..., Smacc office.com, smacc github, WarsawOffice - Pl..., WarsawTeam - On..., UniFi controllers, and Other Bookmarks.

The main area has a dark header with links for Prometheus, Alerts, Graph, Status, and Help. A checkbox labeled "Enable query history" is checked. Below it is a text input field with placeholder text "Expression (press Shift+Enter for newlines)".

A dropdown menu is open over the expression input, listing various Prometheus metrics. The "Graph" tab is selected in the dropdown menu. The listed metrics include:

- ✓ - insert metric at cursor -
- ALERTS
- order_mgmt_audit_duration_seconds_count
- order_mgmt_audit_duration_seconds_sum
- order_mgmt_database_duration_seconds_bucket
- order_mgmt_database_duration_seconds_count
- order_mgmt_database_duration_seconds_sum
- order_mgmt_duration_seconds_count
- order_mgmt_duration_seconds_sum
- scrape_duration_seconds
- scrape_samples_post_metric_relabeling
- scrape_samples_scraped
- up

To the right of the dropdown, there is a "Value" input field and a "Remove Graph" button.

Prometheus

The screenshot shows a web browser window with the URL `127.0.0.1:9090/alerts`. The browser tab bar includes tabs for Prometheus, Alertmanager, Grafana, and other monitoring tools. The bookmarks bar contains links to Apps, Squad Warsaw Bo..., Smacc office.com, smacc github, WarsawOffice - Pl..., WarsawTeam - On..., UniFi controllers, and Other Bookmarks.

The main content area is titled "Alerts". It displays two alert definitions:

ProductionInstanceDown (0 active)

```
alert: ProductionInstanceDown
expr: up{env="production"} == 0
for: 2m
labels:
  severity: opsgenie
annotations:
  description: '{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 2 minutes.'
  summary: Instance {{ $labels.instance }} of {{ $labels.app }} is down
```

StagingAppServiceInstanceDown (0 active)

Kill the service

```
☁ demo ⚡ docker stop talk-golang-prom-red_alertmanager_1
```

Prometheus

The screenshot shows the Prometheus web interface at `127.0.0.1:9090/alerts`. The top navigation bar includes tabs for Prometheus, Alertmanager, Grafana, and other monitoring tools. The main content area is titled "Alerts" and displays one active alert named "ProductionInstanceDown".

ProductionInstanceDown (1 active)

```
alert: ProductionInstanceDown
expr: up{env="production"} == 0
for: 2m
labels:
  severity: opsgenie
annotations:
  description: '{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 2 minutes.'
  summary: Instance {{ $labels.instance }} of {{ $labels.app }} is down
```

Labels	State	Active Since	Value
<code>alertname="ProductionInstanceDown"</code> <code>app="order-manager"</code> <code>env="production"</code> <code>instance="order-manager:8080"</code> <code>job="my-service"</code> <code>owner="wb@example.com"</code> <code>severity="opsgenie"</code> <code>tier="front-end"</code>	PENDING	2018-10-02 09:35:20.5392318 +0000 UTC	0

StagingAppServiceInstanceDown (0 active)

Prometheus

The screenshot shows a browser window with the URL `127.0.0.1:9090/alerts`. The page title is "Prometheus". The main content area is titled "Alerts" and displays a single active alert named "ProductionInstanceDown".

ProductionInstanceDown (1 active)

```
alert: ProductionInstanceDown
expr: up{env="production"} == 0
for: 2m
labels:
  severity: opsgenie
annotations:
  description: '{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 2 minutes.'
  summary: Instance {{ $labels.instance }} of {{ $labels.app }} is down
```

Labels	State	Active Since	Value
<code>alertname="ProductionInstanceDown"</code> <code>app="order-manager"</code> <code>env="production"</code> <code>instance="order-manager:8080"</code> <code>job="my-service"</code> <code>owner="wb@example.com"</code> <code>severity="opsgenie"</code> <code>tier="front-end"</code>	FIRING	2018-10-02 09:35:20.5392318 +0000 UTC	0

StagingAppServiceInstanceDown (0 active)

Alertmanager

The screenshot shows the Alertmanager interface running in a web browser at `127.0.0.1:9093/#/alerts`. The browser tab bar includes links to Prometheus, Grafana, and other local services.

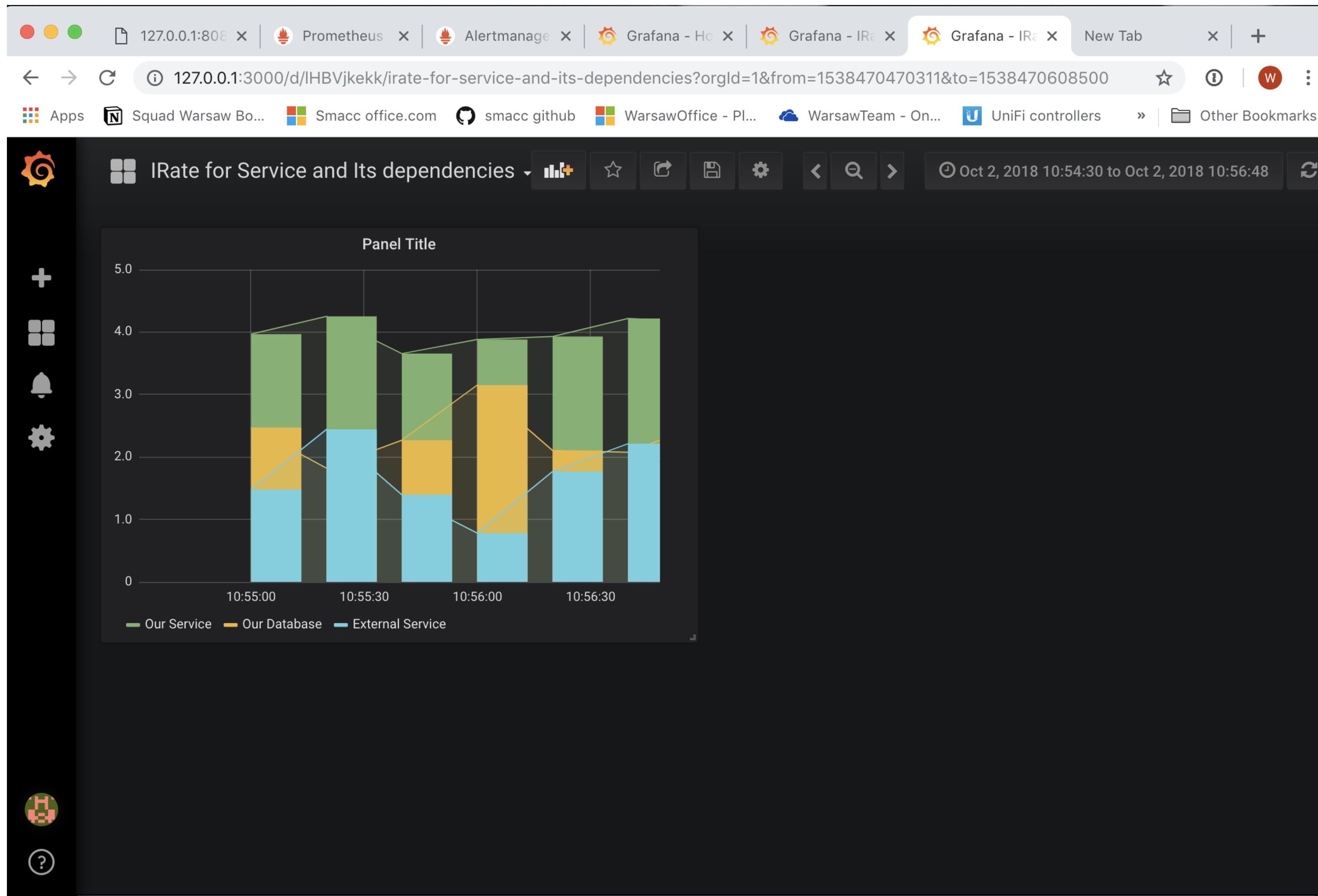
The main interface has a navigation bar with tabs: Alertmanager (selected), Alerts, Silences, and Status. A "New Silence" button is located in the top right corner of the main content area.

The central area contains a search/filter component with tabs for "Filter" and "Group". It includes a "Custom matcher, e.g. `env="production"`" input field and a "+" button to add more filters. To the right, there are buttons for "Receiver: All", "Silenced", and "Inhibited".

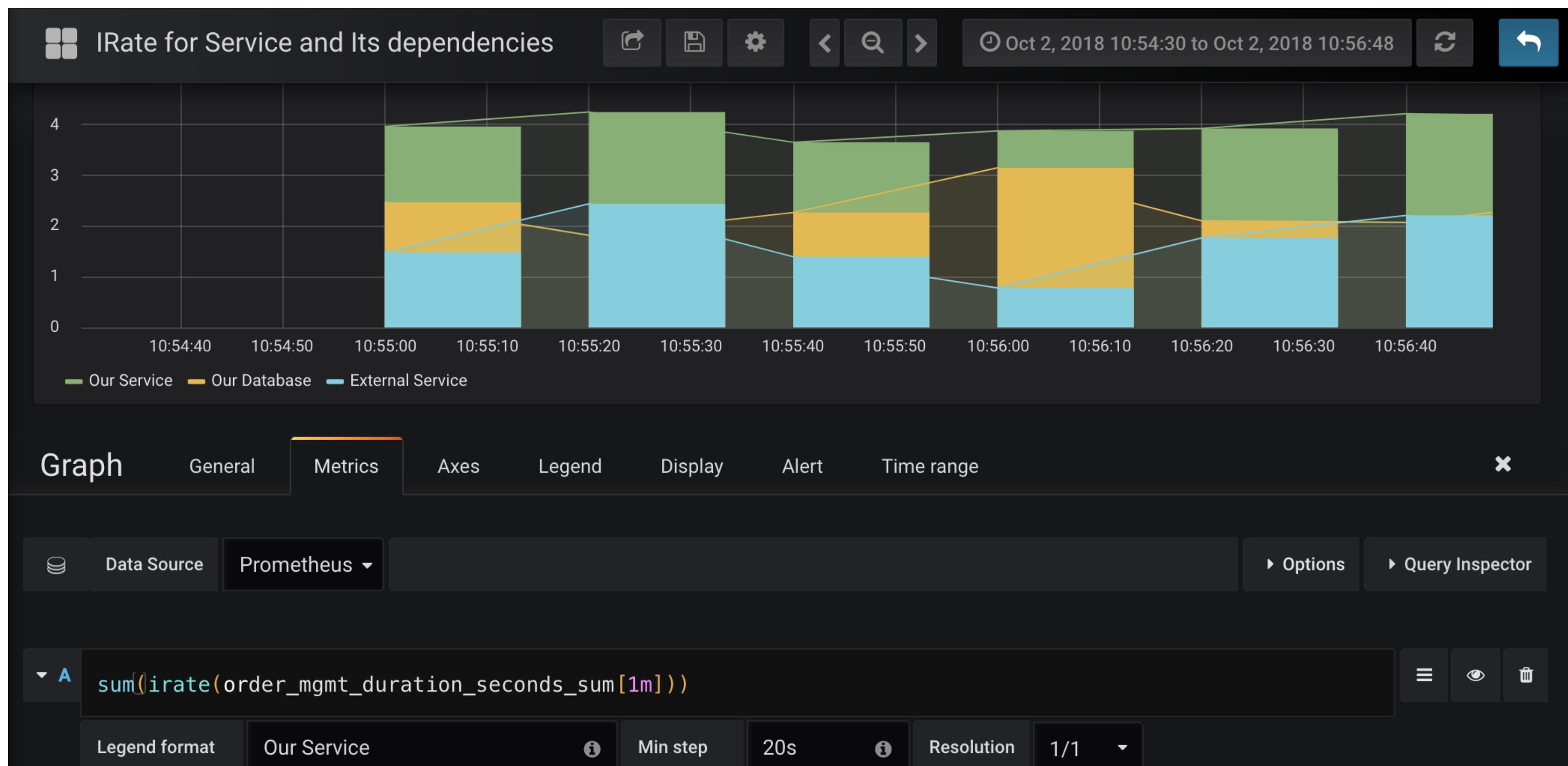
An alert detail card is displayed, showing the following information:

- Alertname:** ProductionInstanceDown
- Timestamp:** 08:45:01, 2018-10-02
- Info:** +
- Source:** +
- Silence:** +
- Labels:** tier="front-end" +, severity="opsgenie" +, owner="wb@example.com" +, job="my-service" +, instance="order-manager:8080" +, environment="production" +, env="production" +, app="order-manager" +

Grafana



Grafana



PromQL

```
order_mgmt_duration_seconds_sum{status_code='200'}
```

PromQL

```
order_mgmt_duration_seconds_sum{status_code='200'}
```

PromQL

```
sum(irate(order_mgmt_duration_seconds_count{job=~".*"}[1m])  
    by (status_code)
```

Explanation: `irate` ([vs rate](#)), [1m], sum.

PromQL

```
sum(irate(order_mgmt_duration_seconds_sum[1m]) )
```

```
sum(irate(order_mgmt_database_duration_seconds_sum[1m]) )
```

```
sum(irate(order_mgmt_audit_duration_seconds_sum[1m]) )
```

Best practices

- Start simple (up/down), later more complex rules
- If in doubt, use histogram ([docs](#))

Summary

- Monitoring saves your time
- Checking logs == debugging vs having tests
- One of the lines of defence

Summary

One of the lines of defence:

- Retrospectives,
- Regular performance reviews by product/dev teams,
- Synthetic tests, end2end, ...
- Alerts from the user perspective (see [docs](#))

Github

[wojciech12 / talk_monitoring_with_prometheus](#)

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. Edit

Manage topics

6 commits 1 branch 0 releases 1 contributor

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

wojciech12 Rename	Latest commit ef2d4eb 24 minutes ago
demo_go	Init 18 hours ago
demo_java	Add demo for java 18 hours ago
slides_go	Rename 24 minutes ago

github.com/wojciech12/talk_monitoring_with_prometheus

Demo

Everything Included:

- Prometheus dashboard and config
- AlertManager dashboard and config
- Simulate the successful and failed calls
- Examples of queries for rate

Questions?

Backup slides

Prometheus - Push model

- See:

<https://prometheus.io/docs/instrumenting/pushing/>

Good for short living jobs in your cluster.

Designing metric names

Which one is better?

- request_duration{app=my_app}
- my_app_request_duration

see documentation on best practises for [metric naming](#) and [instrumentation](#)

Designing metric names

Which one is better?

- `order_mgmt_db_duration_seconds_sum`
- `order_mgmt_duration_seconds_sum{dep_name='db'}`

Prometheus + K8S = <3

Labels are propagated from k8s to prometheus

Integration with Prometheus

```
cat memcached-0-service.yaml
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: memcached-0
  labels:
    app: memcached
    kubernetes.io/name: "memcached"
    role: shard-0
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/scheme: "http"
    prometheus.io/path: "metrics"
    prometheus.io/port: "9150"
```

<https://github.com/skarab7/kubernetes-memcached>