# How to monitor your micro-service with Prometheus?
# How to design the metrics?

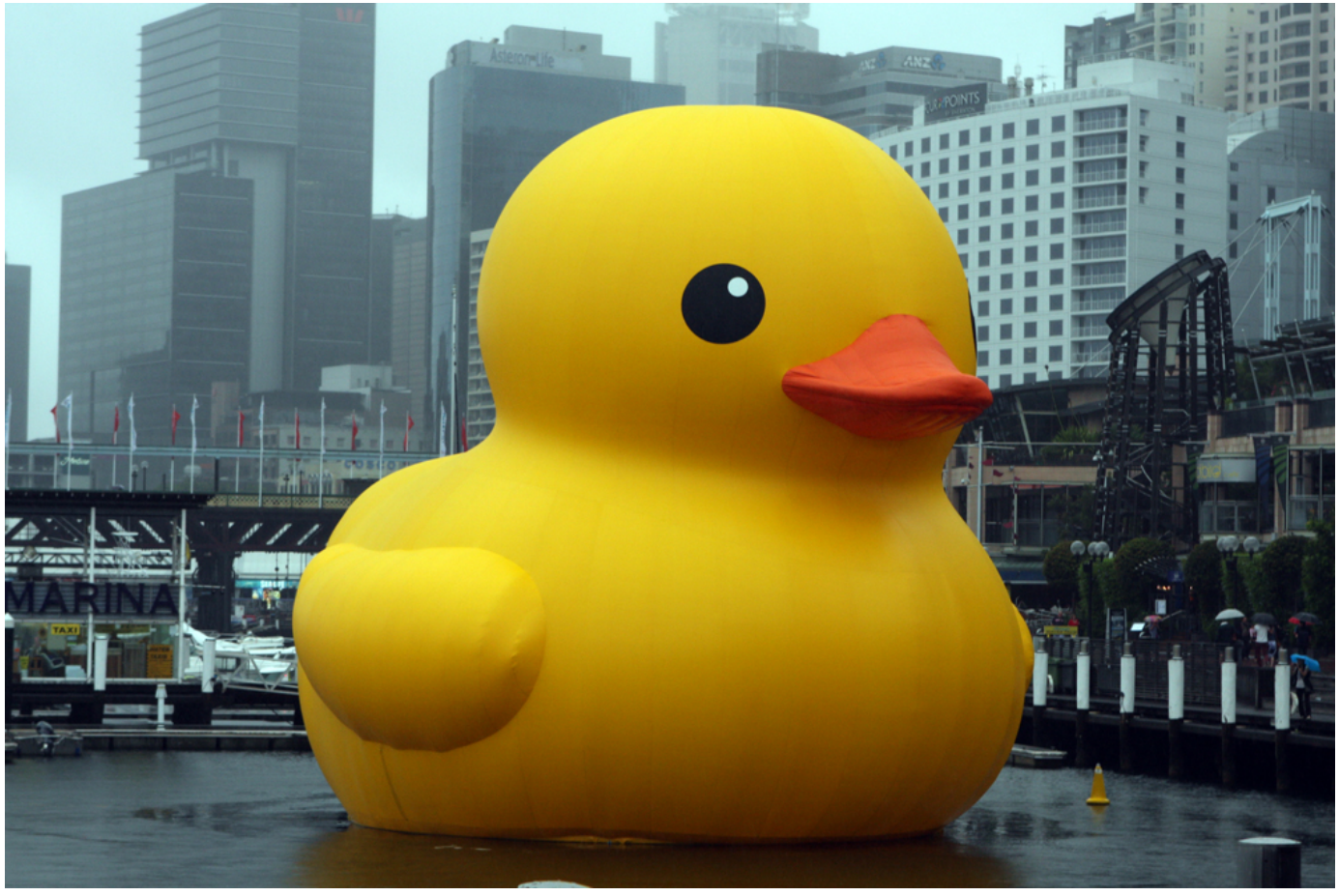
WOJCIECH BARCZYŃSKI - SMACC.IO | 17 OCTOBER 2018

# ABOUT ME

- Lead Software Developer - SMACC (FinTech/AI)
- Before:
  System Engineer i Developer Lyke
- Before:
  1000+ nodes, 20 data centers with Openstack
- Point of view:
  Startups, fast-moving environment

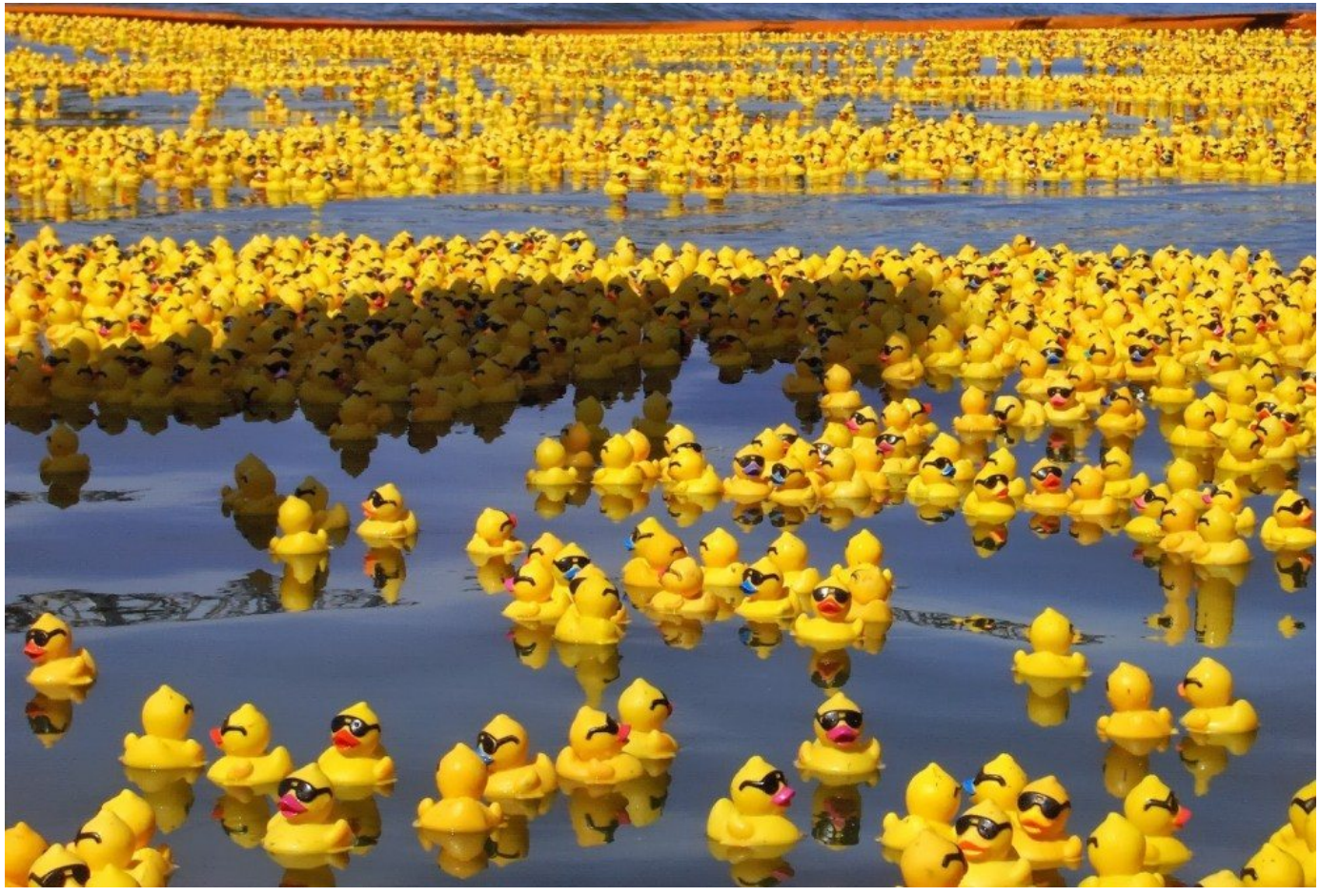
# WHY?

## 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

Suprisengly: the last one implemented

# CENTRALIZED LOGGING

- Usually much too late
- Post-mortem
- Hard to find the needle
- Like a debugging vs testing

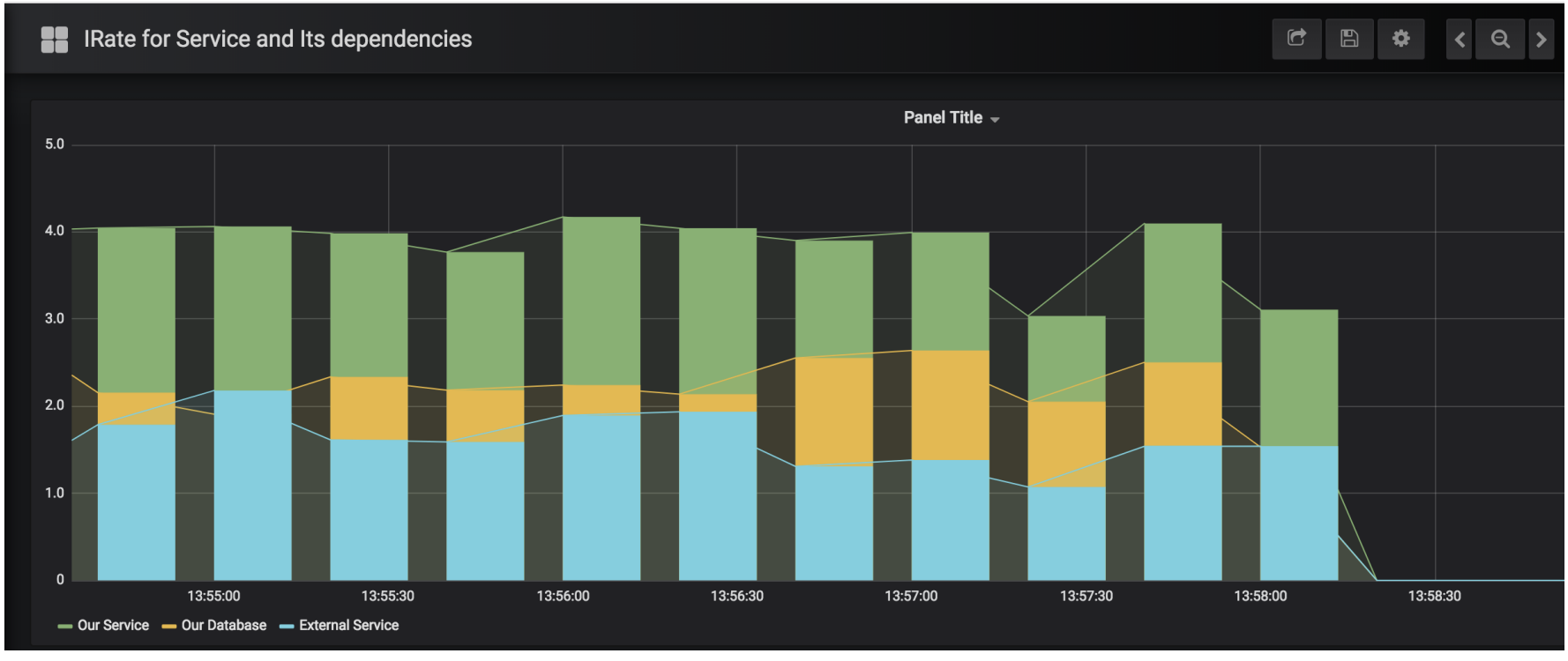
# MONITORING

- 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

| | |
|---|---|
| **U**tilization | the average time that the resource was busy servicing work |
| **S**aturation | extra work which it can't service, often queued |
| **E**rrors | the count of error events |

Documented and Promoted by Berdan Gregg

# USE

- **U**tilization: as a percent over a time interval: "one disk is running at 90% utilization".
- **S**aturation:
- **E**rrors:

# 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

- **traditionaly** more instance oriented
- still useful in the microservices world

# RED

| | |
|---|---|
| **R**ate | How busy is your service? |
| **E**rror | Errors |
| **D**uration | What is the latency of my service? |

Tom Wilkie's guideline for instrumenting applications.

# RED

- **Rate** - how many request 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

[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

- Wide support for languages
- Metrics collected over HTTP *metrics/*
- Pull model (see *scrape time*), push-mode possible
- integration with k8s
- PromQL

# METRICS IN PLAIN TEXT

```
# 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
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
```

# METRICS IN PLAIN TEXT

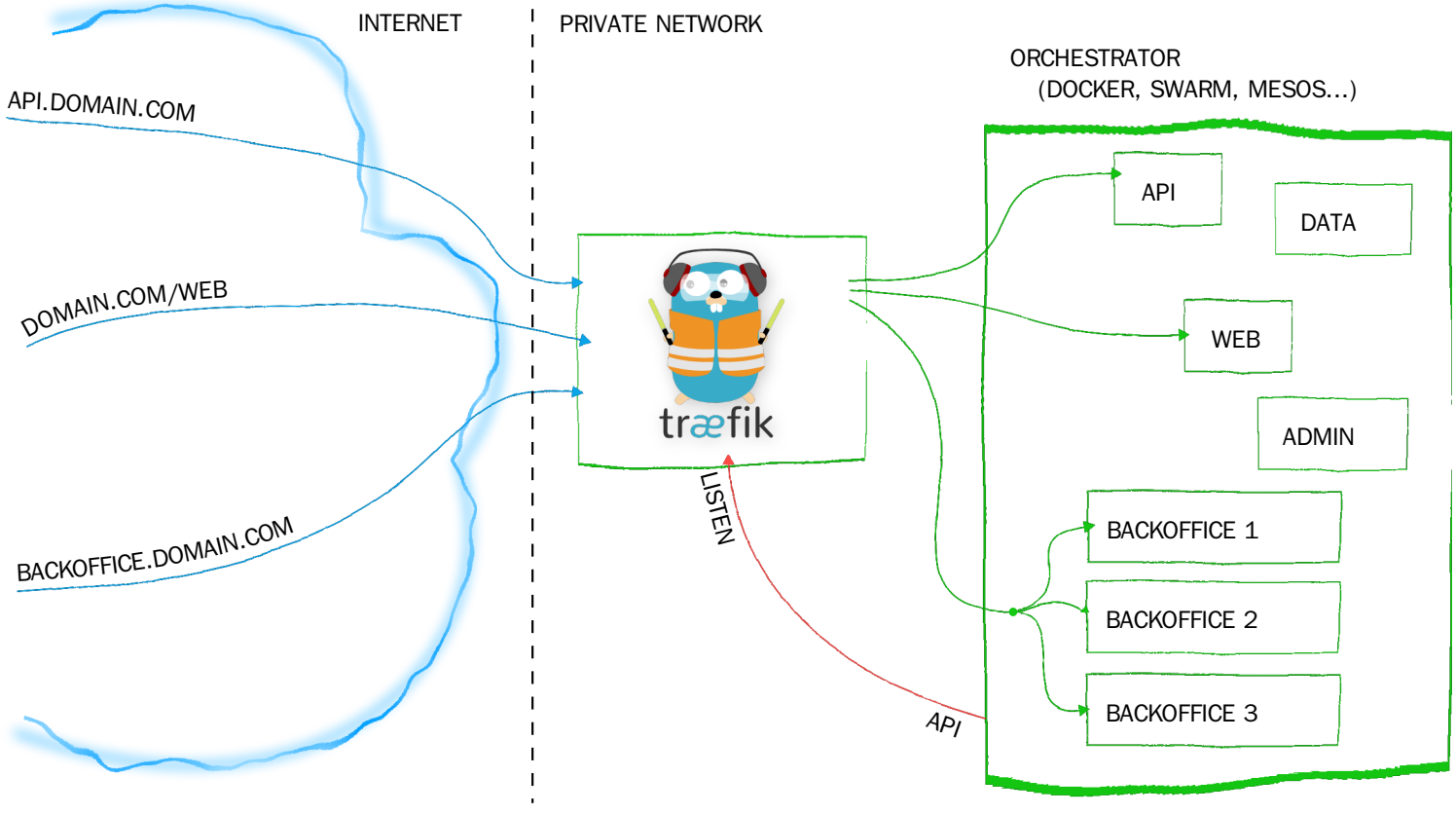
```
# HELP go_gc_duration_seconds A summary of the GC invocation d
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 9.01e-05
go_gc_duration_seconds{quantile="0.25"} 0.000141101
go_gc_duration_seconds{quantile="0.5"} 0.000178902
go_gc_duration_seconds{quantile="0.75"} 0.000226903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

# EXPORTERS

- Mongodb
- Mysql
- Postgresql
- rabbitmq
- ...
- also Blackbox exporter

examples: memcached, psql

# CLOUD-NATIVE PROJECTS INTEGRATION




- --web.metrics.prometheus

# PROMETHEUS PromQL

```
# working with historams:

histogram_quantile(0.9,
  rate(http_req_duration_seconds_bucket[10m]))

# rates:

    rate(http_requests_total{job="api-server"}[5m])
    irate(http_requests_total{job="api-server"}[5m])

# more complex:

    redict_linear()
    holt_winters()
```

# PROMETHEUS PromQL

## Alarming:

```
ALERT ProductionAppServiceInstanceDown
   IF up { environment = "production", app =~ ".+"} == 0
   FOR 4m
   ANNOTATIONS {
        summary = "Instance of {{$labels.app}} is down",
        description = " Instance  {{$labels.instance}} of app
   }
```

# METRICS

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

## HISTOGRAM

`traefik_duration_seconds_bucket{method="GET,code="200"}`

| | |
|---|---|
| {le="0.1"} | 2229 |
| {le="0.3"} | 107 |
| {le="1.2"} | 100 |
| {le="5"} | 4 |
| {le="+Inf"} | 2 |
| _sum | |
| _count | 2342 |

## SUMMARY

`http_request_duration_seconds`

| | |
|---|---|
| {quantile="0.5"} | 4 |
| {quantile="0.9"} | 5 |
| http_request_duration_seconds_sum | 9 |
| http_request_duration_seconds_count | 3 |

# HISTOGRAM / SUMMARY:

- Latency of services
- Request or Request size

Histograms recommended

# RED

## Metric + PromQL:

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

# METRIC AND LABEL NAMING

Best practises on metric names:

- service name is your prefix `user_`
- state the bae unit `_seconds` and `_bytes`

# PROMETHEUS + JAVA

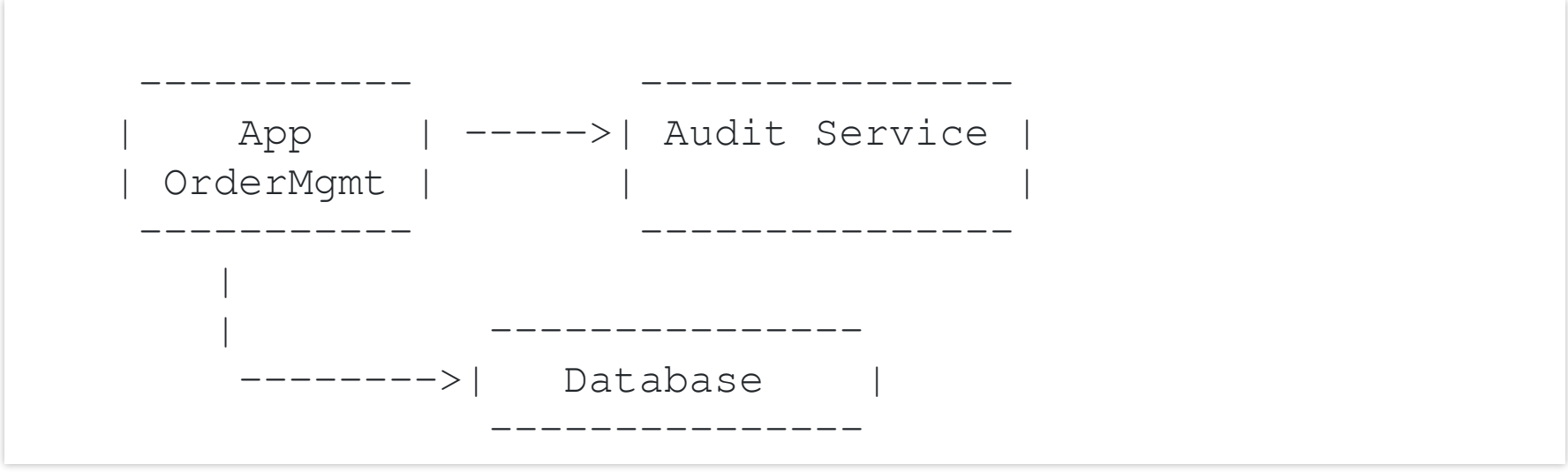

# JAVA CLIENT

- client_python
- Counter
- Gauge
- Summary
- Histogram

# JAVA

- Sprint 1.5 - demo
- Sprint 2.x - wrapper with `io.micrometer.core`
- JMX exporter - https://github.com/prometheus/jmx_exporter

# DEMO: SIMPLE REST SERVICE



## DEMO:

- http://127.0.0.1:8080 - service
- http://127.0.0.1:8080/metrics/
- http://127.0.0.1:9090 - prometheus
- http://127.0.0.1:3000 - grafana
- http://127.0.0.1:9093 - alertmanager

# DEMO

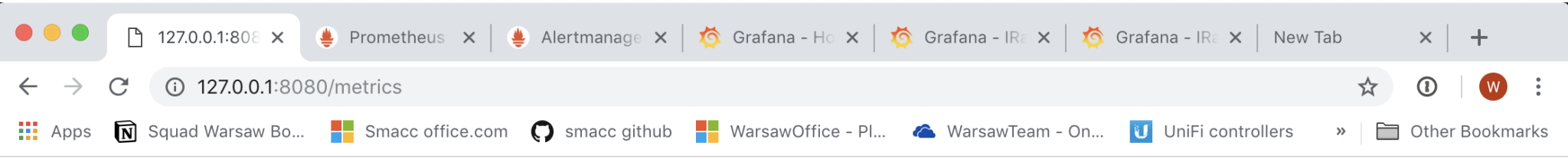
```
☁  demo ⚡ make start

☁  demo ⚡ docker ps
CONTAINER ID        IMAGE                          PORTS
5f824d1bc789        grafana/grafana:5.2.2          0.0.0.0:3000->3
d681a414a8b6        prom/prometheus:v2.1.0         0.0.0.0:9090->9
ea0d9233e159        prom/alertmanager:v0.15.1      0.0.0.0:9093->9
732c59fb3753        java-prom_order-manager        0.0.0.0:8080->8
```

# DEMO: GENERATE CALLS

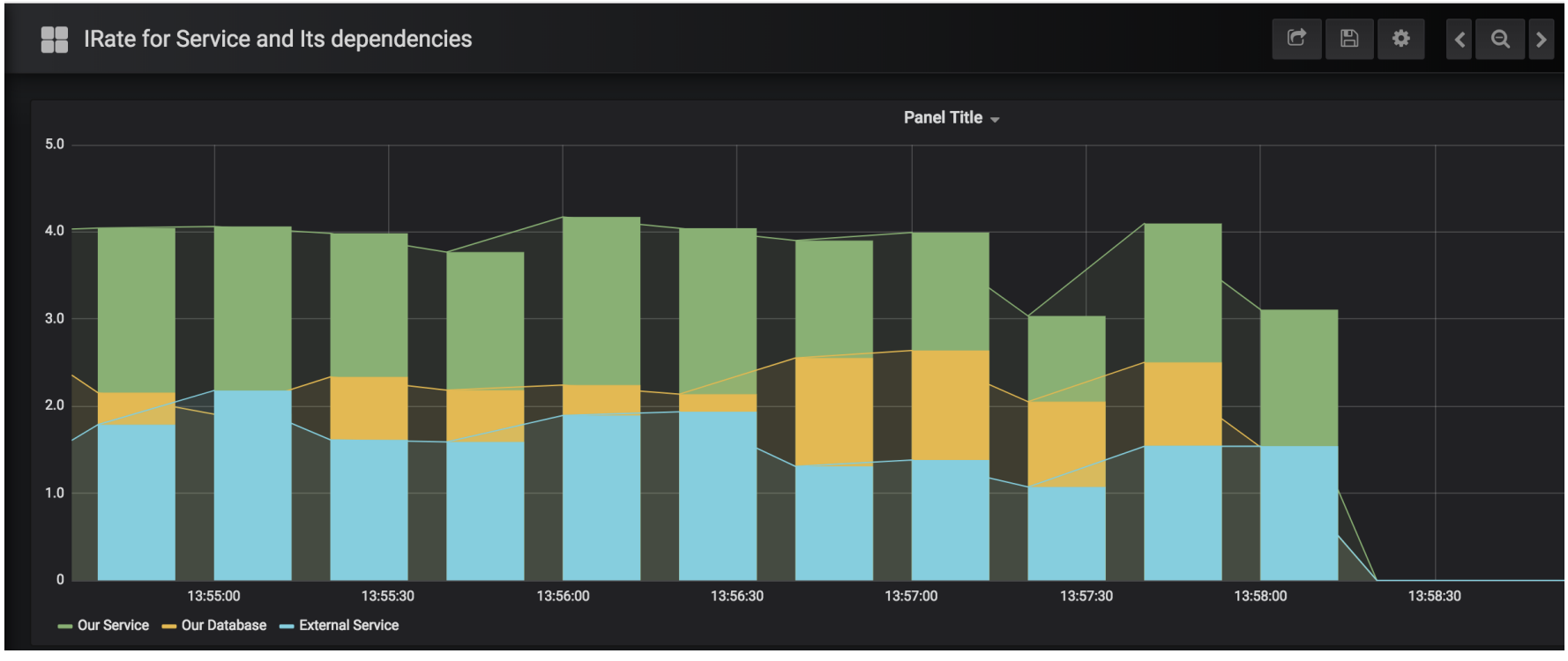
```
demo      ⚡ make srv_wrk_random
```

With error injection

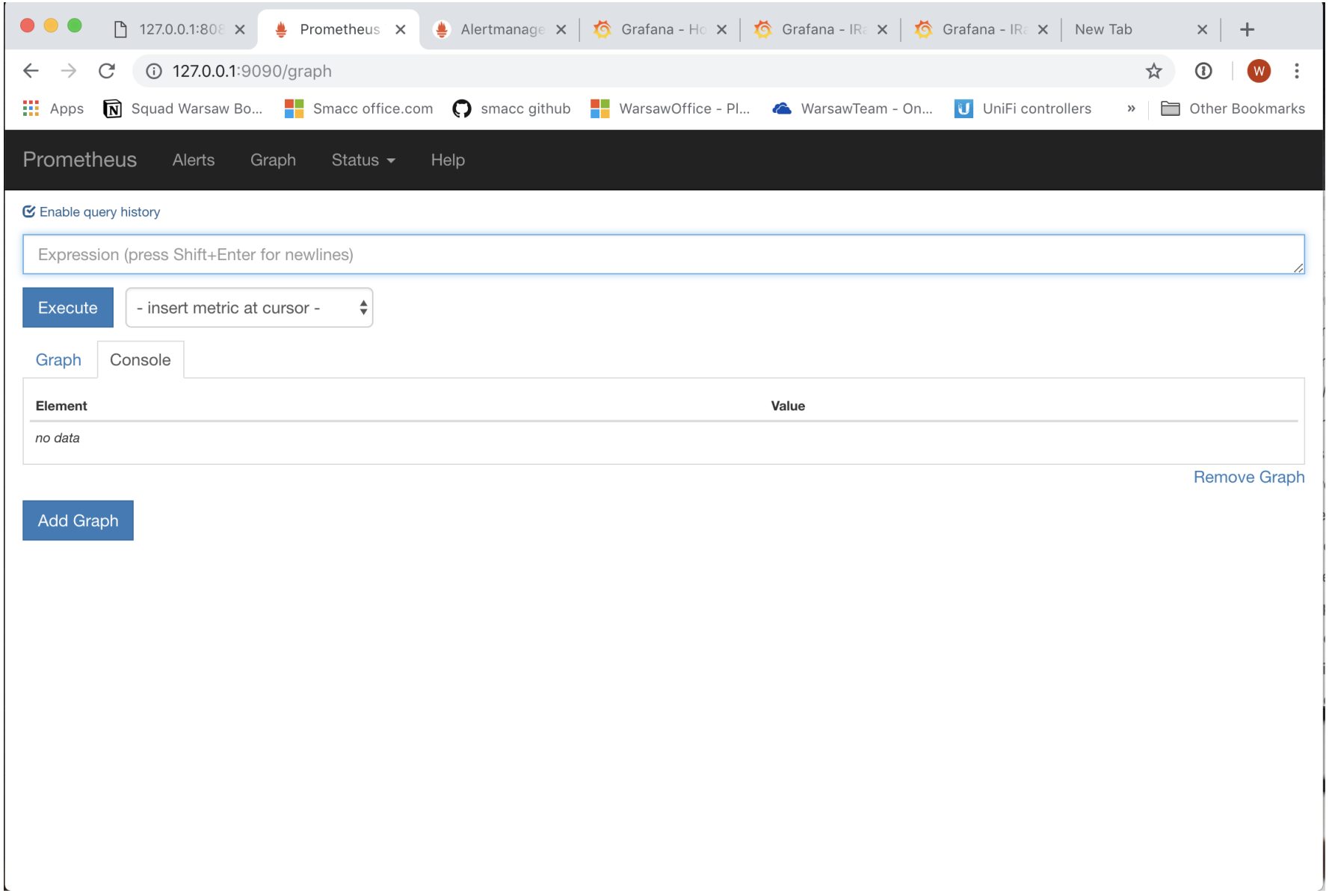



```
# HELP order_mgmt_database_duration_seconds Multiprocess metric
# TYPE order_mgmt_database_duration_seconds histogram
order_mgmt_database_duration_seconds_sum{sql_state="0",status_code="0"} 338.9096608161926
order_mgmt_database_duration_seconds_sum{sql_state="HY000",status_code="1001"} 92.89293241500854
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
```

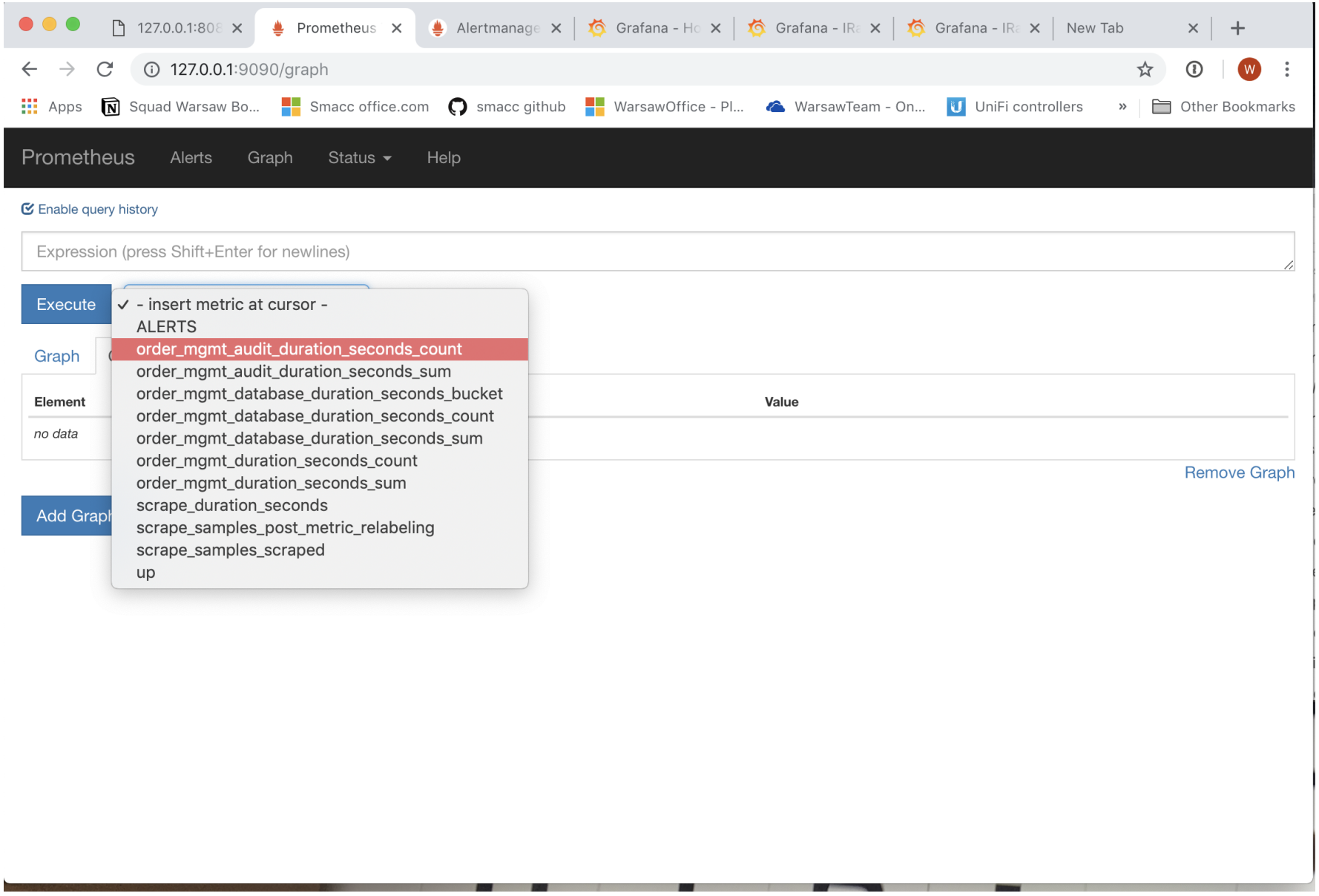
# GRAFANA


IRate for Service and Its dependencies
Panel Title
5.0
4.0
3.0
2.0
1.0
0
13:55:00
13:55:30
13:56:00
13:56:30
13:57:00
13:57:30
13:58:00
13:58:30
Our Service
Our Database
External Service

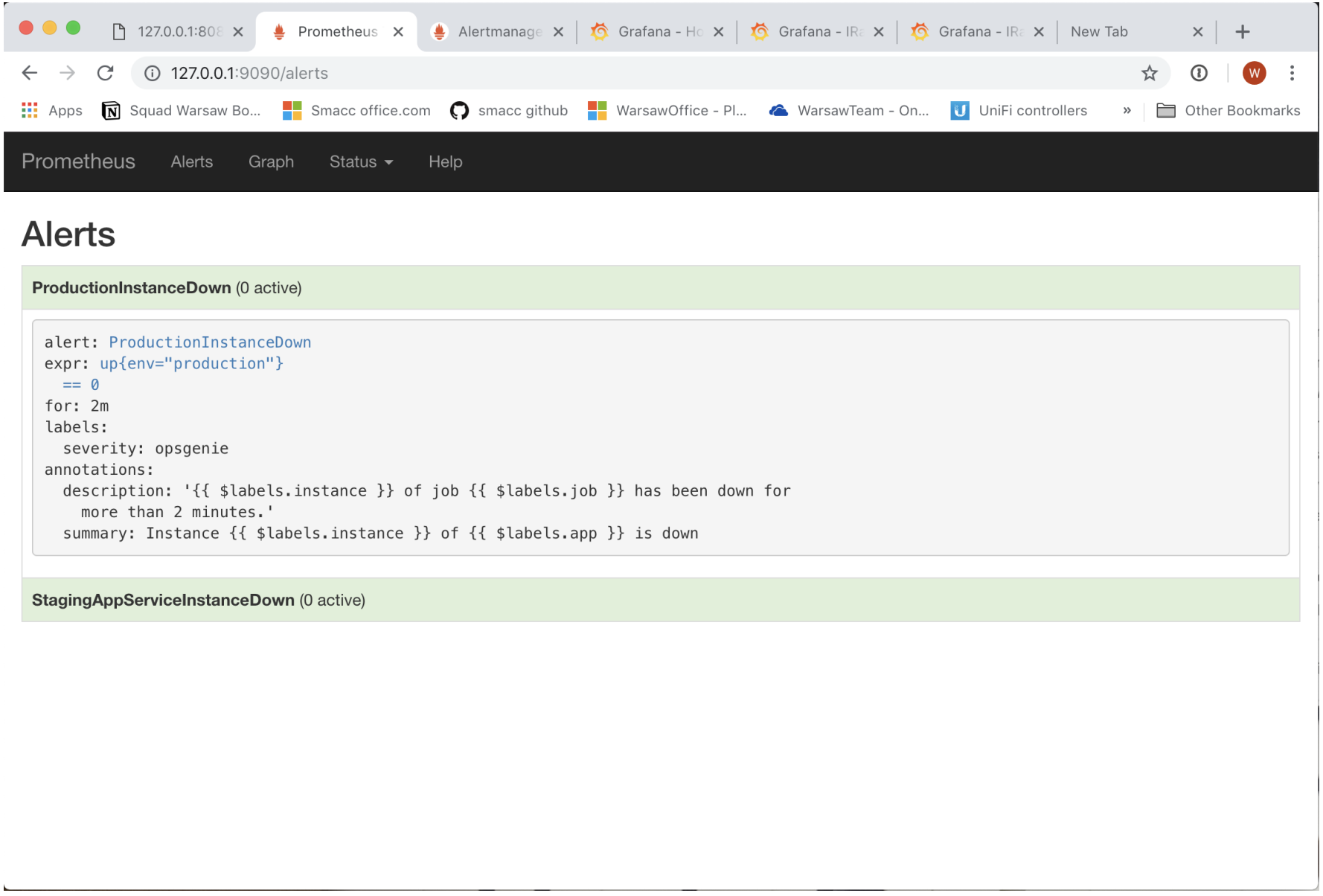
# PROMETHEUS


127.0.0.1:808
Prometheus
Alertmanage
Grafana - Ho
Grafana - IRa
Grafana - IRa
New Tab
127.0.0.1:9090/graph
Apps
Squad Warsaw Bo...
Smacc office.com
smacc github
WarsawOffice - Pl...
WarsawTeam - On...
UniFi controllers
Other Bookmarks
Prometheus
Alerts
Graph
Status
Help
Enable query history
Expression (press Shift+Enter for newlines)
Execute
- insert metric at cursor -
Graph
Console
Element
Value
no data
Remove Graph
Add Graph

# PROMETHEUS


127.0.0.1:808
Prometheus
Alertmanage
Grafana - Ho
Grafana - IRa
Grafana - IRa
New Tab
127.0.0.1:9090/graph
Apps
Squad Warsaw Bo...
Smacc office.com
smacc github
WarsawOffice - Pl...
WarsawTeam - On...
UniFi controllers
Other Bookmarks
Prometheus
Alerts
Graph
Status
Help
Enable query history
Expression (press Shift+Enter for newlines)
Execute
- 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
Graph
Element
Value
no data
Remove Graph
Add Graph

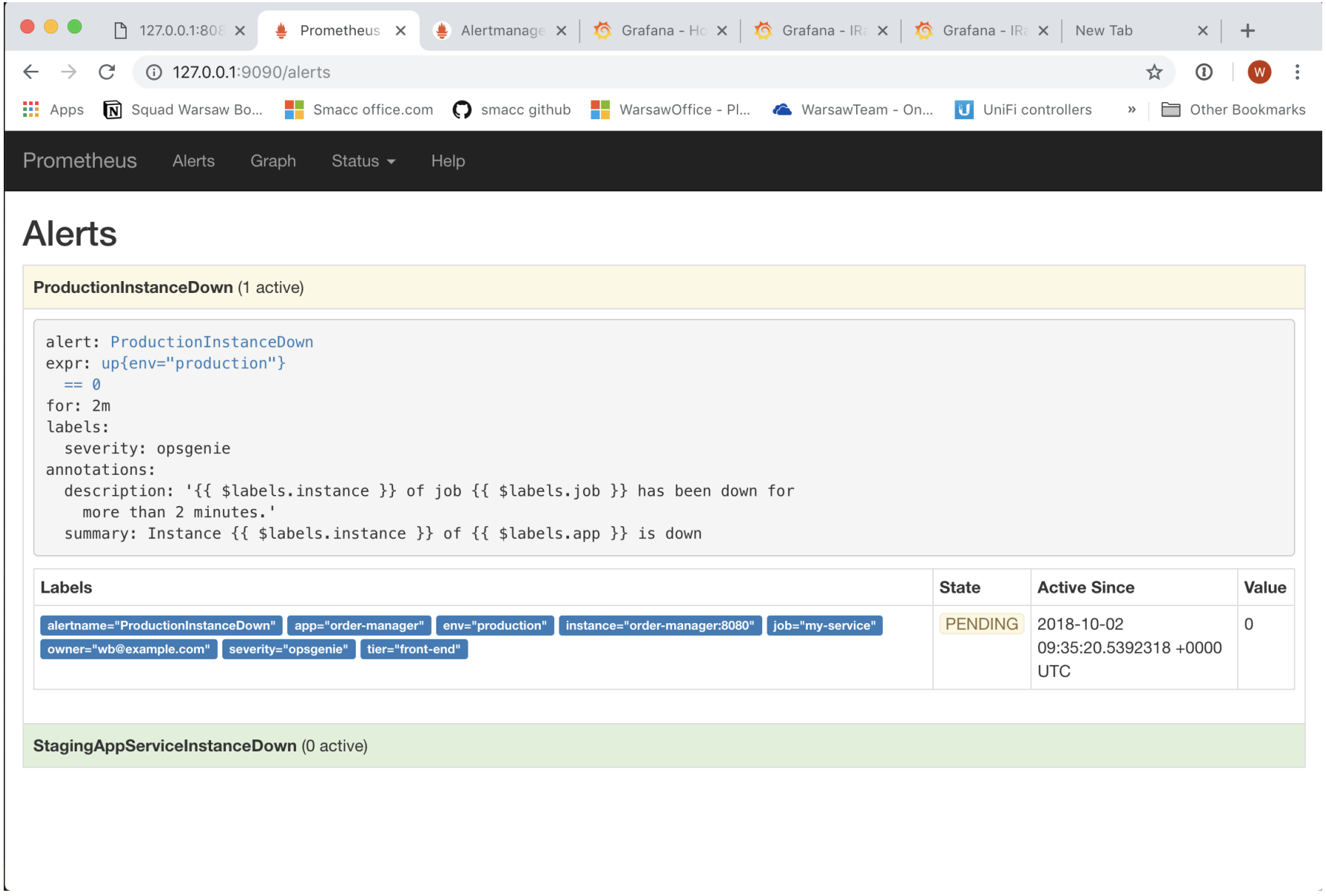
# PROMETHEUS


127.0.0.1:808
Prometheus
Alertmanage
Grafana - Ho
Grafana - IRa
Grafana - IRa
New Tab
127.0.0.1:9090/alerts
Apps
Squad Warsaw Bo...
Smacc office.com
smacc github
WarsawOffice - Pl...
WarsawTeam - On...
UniFi controllers
Other Bookmarks
Prometheus
Alerts
Graph
Status
Help
Alerts
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)

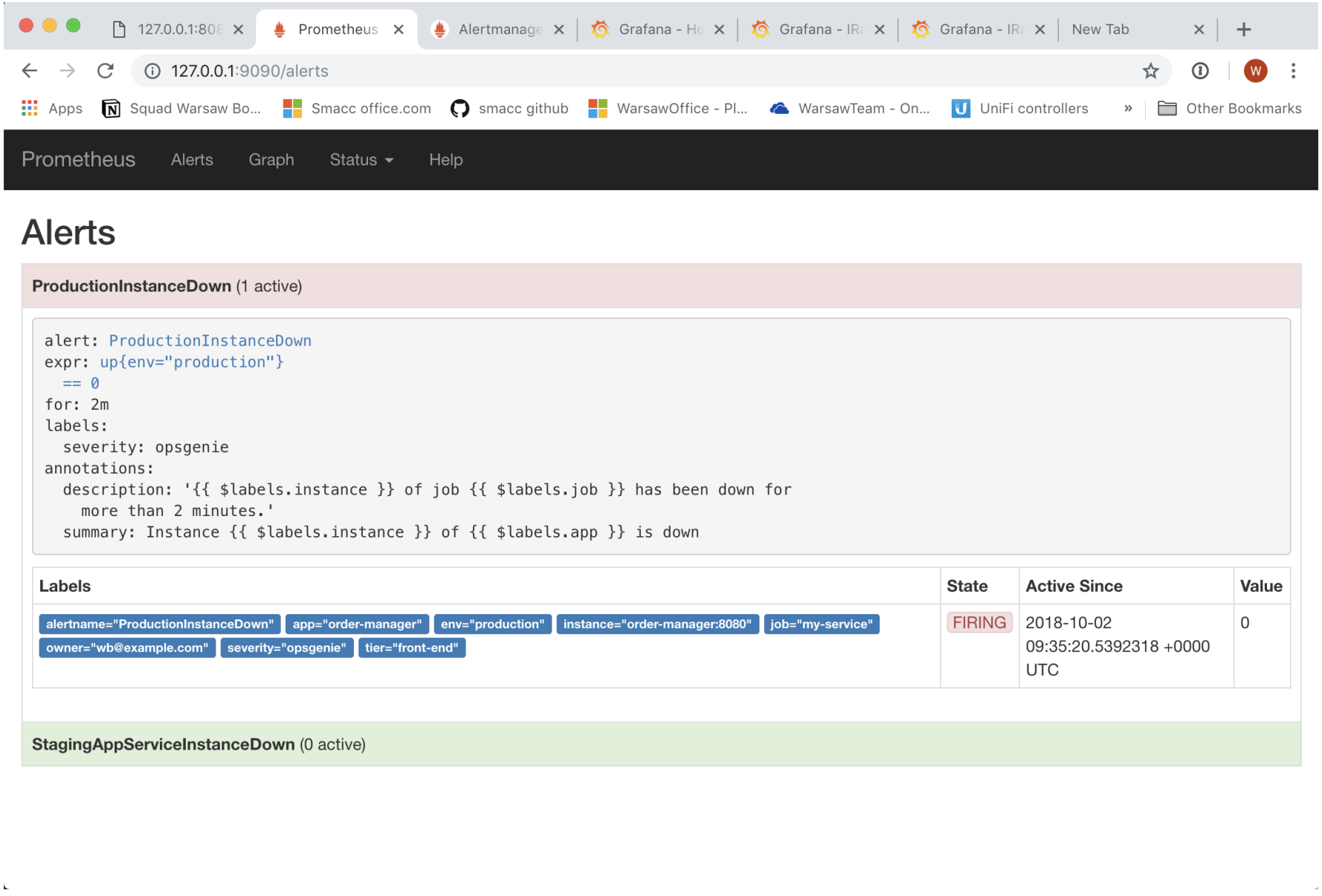
# KILL THE SERVICE

```
demo ⚡ docker stop java-prom_order-manager_1
```

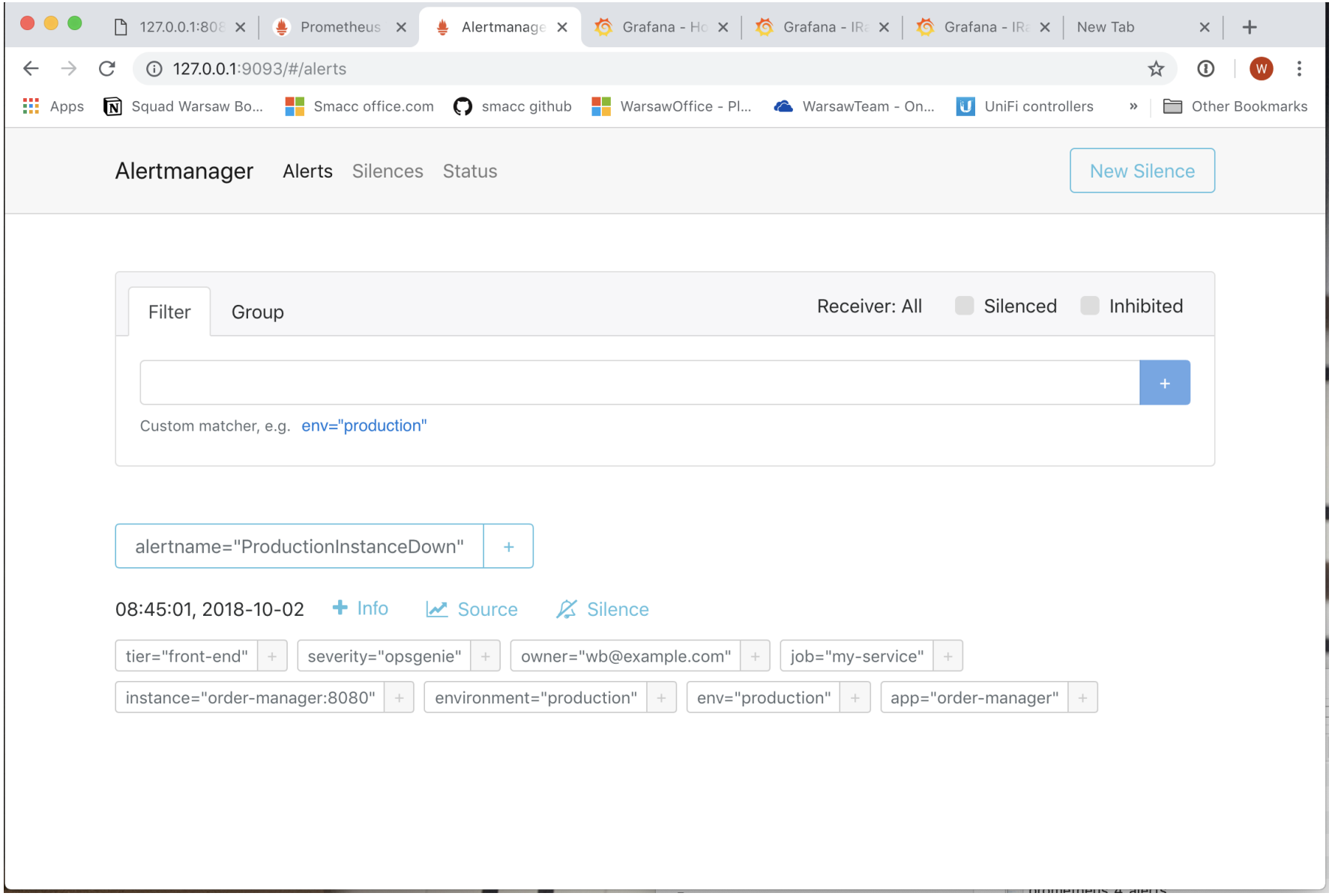
# PROMETHEUS


127.0.0.1:808
Prometheus
Alertmanage
Grafana - Ho
Grafana - IRa
Grafana - IRa
New Tab
127.0.0.1:9090/alerts
Apps
Squad Warsaw Bo...
Smacc office.com
smacc github
WarsawOffice - Pl...
WarsawTeam - On...
UniFi controllers
Other Bookmarks
Prometheus
Alerts
Graph
Status
Help
Alerts
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
alertname="ProductionInstanceDown"
app="order-manager"
env="production"
instance="order-manager:8080"
job="my-service"
owner="wb@example.com"
severity="opsgenie"
tier="front-end"
PENDING
2018-10-02 09:35:20.5392318 +0000 UTC
0
StagingAppServiceInstanceDown (0 active)

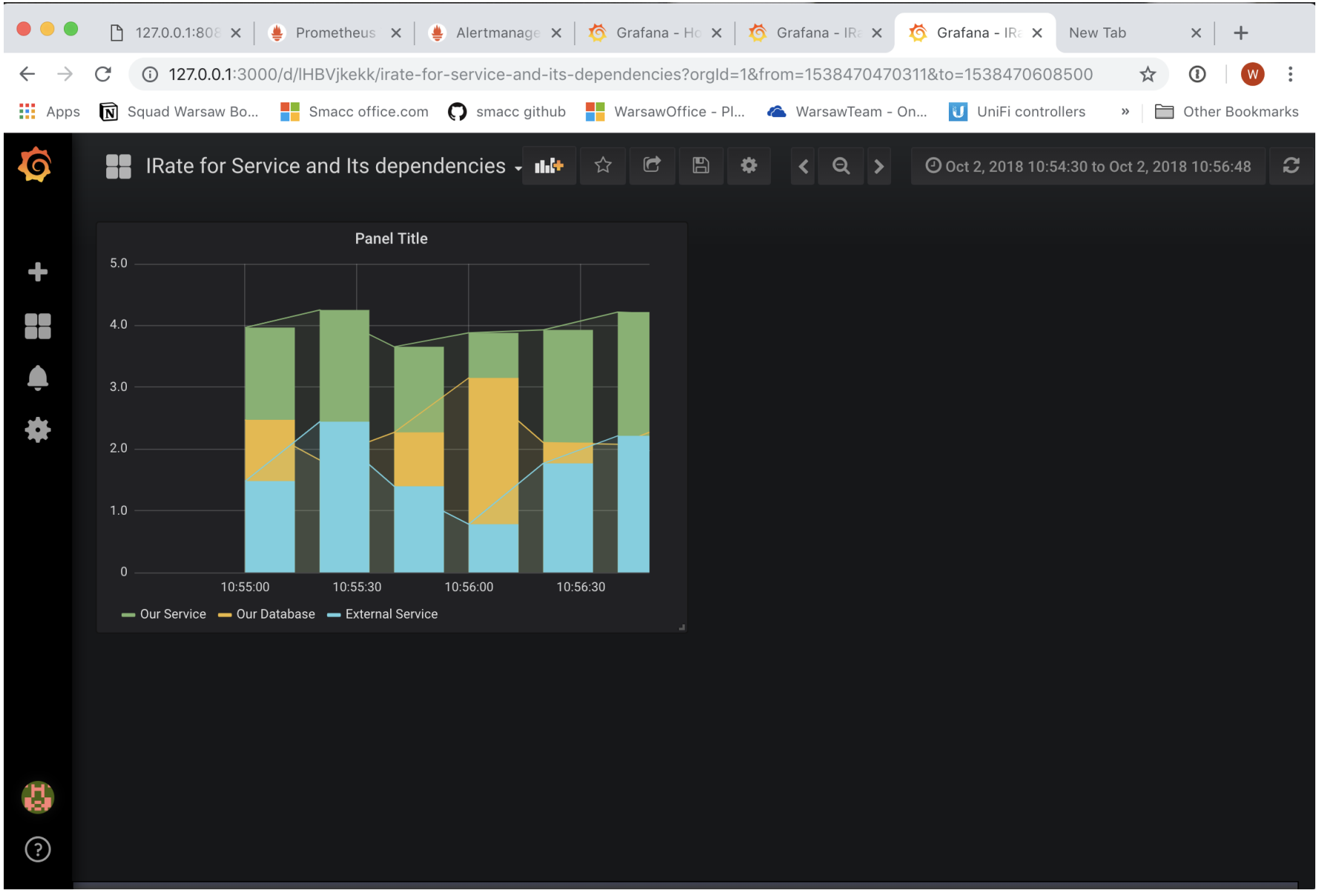
# PROMETHEUS


127.0.0.1:808
Prometheus
Alertmanage
Grafana - Ho
Grafana - IRa
Grafana - IRa
New Tab
127.0.0.1:9090/alerts
Apps
Squad Warsaw Bo...
Smacc office.com
smacc github
WarsawOffice - Pl...
WarsawTeam - On...
UniFi controllers
Other Bookmarks
Prometheus
Alerts
Graph
Status
Help
Alerts
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
alertname="ProductionInstanceDown"
app="order-manager"
env="production"
instance="order-manager:8080"
job="my-service"
owner="wb@example.com"
severity="opsgenie"
tier="front-end"
FIRING
2018-10-02 09:35:20.5392318 +0000 UTC
0
StagingAppServiceInstanceDown (0 active)

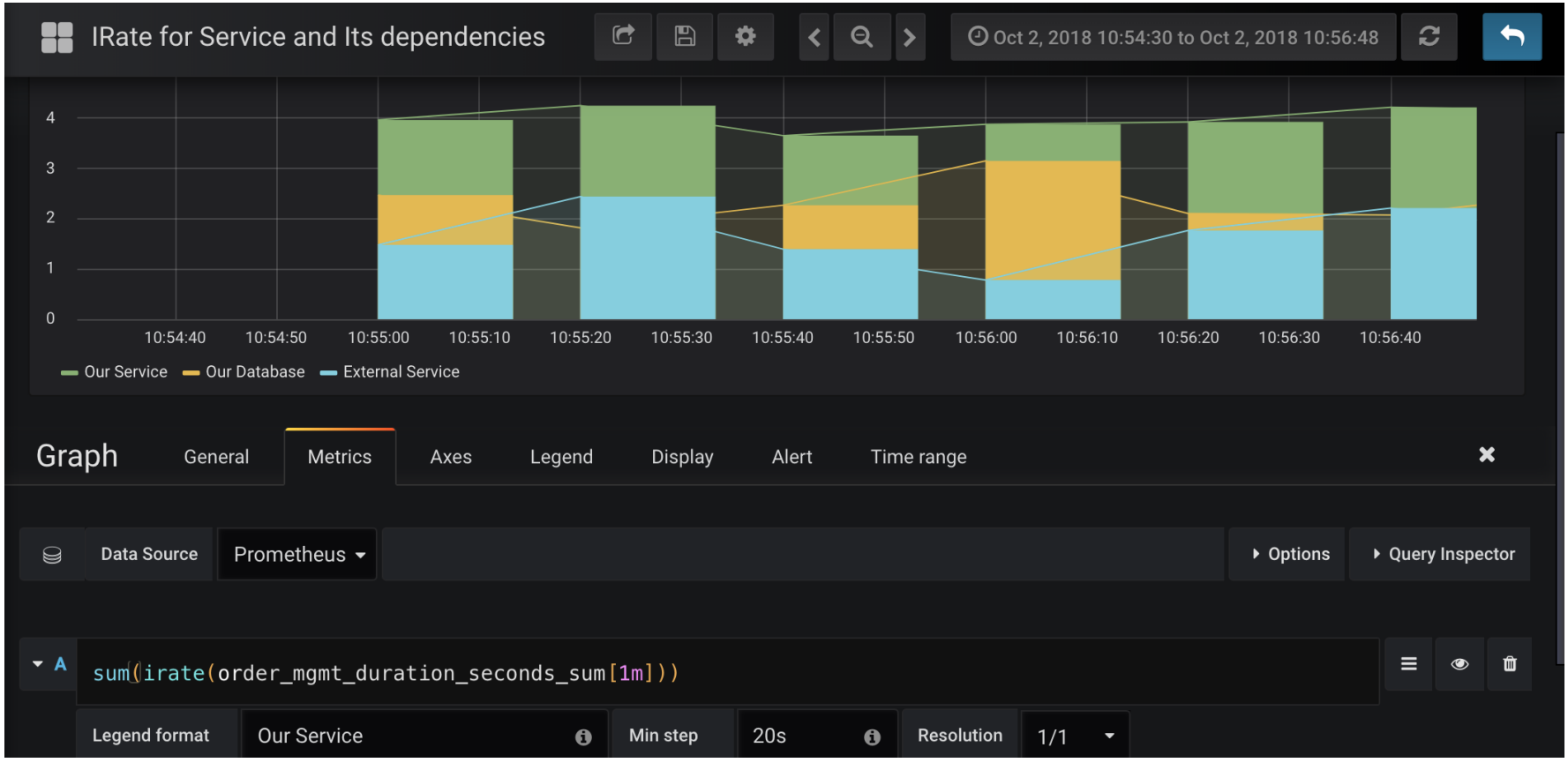
# ALERTMANAGER



# GRAFANA


Prometheus
Alertmanage
Grafana - Ho
Grafana - IRa
Grafana - IRa
New Tab
127.0.0.1:3000/d/lHBVjkekk/irate-for-service-and-its-dependencies?orgId=1&from=1538470470311&to=1538470608500
Apps
Squad Warsaw Bo...
Smacc office.com
smacc github
WarsawOffice - Pl...
WarsawTeam - On...
UniFi controllers
Other Bookmarks
IRate for Service and Its dependencies
Oct 2, 2018 10:54:30 to Oct 2, 2018 10:56:48
Panel Title
5.0
4.0
3.0
2.0
1.0
0
10:55:00
10:55:30
10:56:00
10:56:30
Our Service
Our Database
External Service

# GRAFANA


IRate for Service and Its dependencies
Oct 2, 2018 10:54:30 to Oct 2, 2018 10:56:48
4
3
2
1
0
10:54:40
10:54:50
10:55:00
10:55:10
10:55:20
10:55:30
10:55:40
10:55:50
10:56:00
10:56:10
10:56:20
10:56:30
10:56:40
Our Service
Our Database
External Service
Graph
General
Metrics
Axes
Legend
Display
Alert
Time range
Data Source
Prometheus
Options
Query Inspector
A
sum(irate(order_mgmt_duration_seconds_sum[1m]))
Legend format
Our Service
Min step
20s
Resolution
1/1

# GITHUB

# DEMO: JAVA CODE

- Metric Definition

# DEMO: PROM STACK

- Prometheus dashboard and config
- AlertManager dashboard and config
- Simulate the successful and failed calls
- Simple Queries for rate

# PromQL

```
order_mgmt_duration_seconds_sum{job=~".*"} or
order_mgmt_database_duration_seconds_sum{job=~".*"} or
order_mgmt_audit_duration_seconds_sum{job=~".*"}
```

# BEST PRACTISES

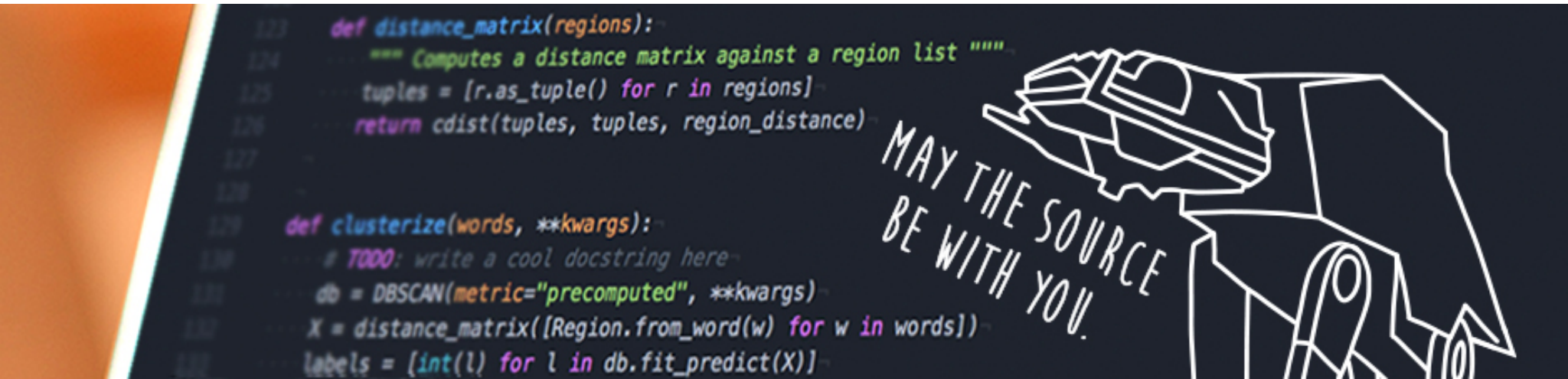
- Start simple (up/down), later add more complex rules
- Sum over Summaries with Q leads to incorrect results, see prom docs

# SUMMARY

- Monitoring saves your time
- Checking logs **Kibana** vs **Grafana** is like debuging vs having tests
- Logging -> high TCO

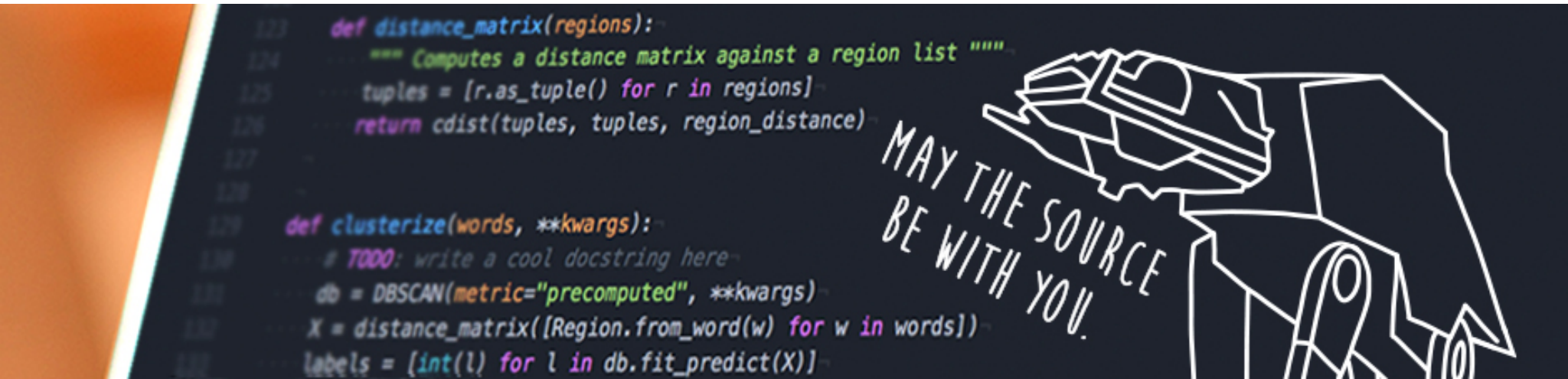
# SUMMARY

# THANK YOU


def distance_matrix(regions):
""" Computes a distance matrix against a region list """
tuples = [r.as_tuple() for r in regions]
return cdist(tuples, tuples, region_distance)
def clusterize(words, **kwargs):
# TODO: write a cool docstring here
db = DBSCAN(metric="precomputed", **kwargs)
X = distance_matrix([Region.from_word(w) for w in words])
labels = [int(l) for l in db.fit_predict(X)]
MAY THE SOURCE BE WITH YOU.

**QUESTIONS?**

ps. We are hiring.



SMACC

Go

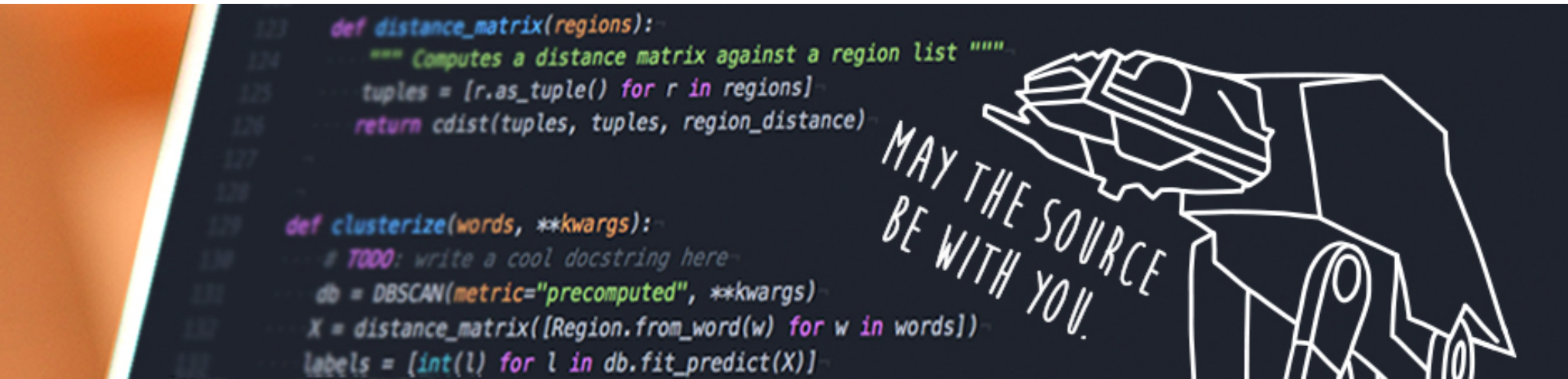

PYTORCH

TensorFlow™

amazon web services™

Azure

# BACKUP SLIDES



# PROMETHUS - LABELS IN ALERT RULES

The labels are propageted to alert rules:

```
ALERT ProductionAppServiceInstanceDown
   IF up { environment = "production", app =~ ".+"} == 0
   FOR 4m
   ANNOTATIONS {
        summary = "Instance of {{$labels.app}} is down",
        description = " Instance  {{$labels.instance}} of app
   }
```

see ../src/prometheus/etc/alert.rules

# ALERTMANGER - LABELS IN ALERTMANGER

Call somebody if the label is `severity=page`:

```
---
 group_by: [cluster]
 # If an alert isn't caught by a route, send it to the pager.
 receiver: team-pager
 routes:
  - match:
      severity: page
    receiver: team-pager

receivers:
- name: team-pager
  opsgenie_configs:
  - api_key: $API_KEY
    teams: example_team
```

see ../src/alertmanager/*.conf

# 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"
  spec:
```

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