

# How to monitor your micro-service with Prometheus?



Wojciech Barczyński - SMACC.io | [LI](#) | [Twitter](#) | 16 October 2018

# ABOUT ME

- Lead Software Developer- SMACC (FinTech/AI)
- Before:  
System Engineer && Developer Lyke
- Looking:  
Tools for efficient and :) teams

# POINT OF VIEW

- Software Developer
- startups && fast-moving environment
- Infra and platform should just work
- Infra and platform ~ invisible

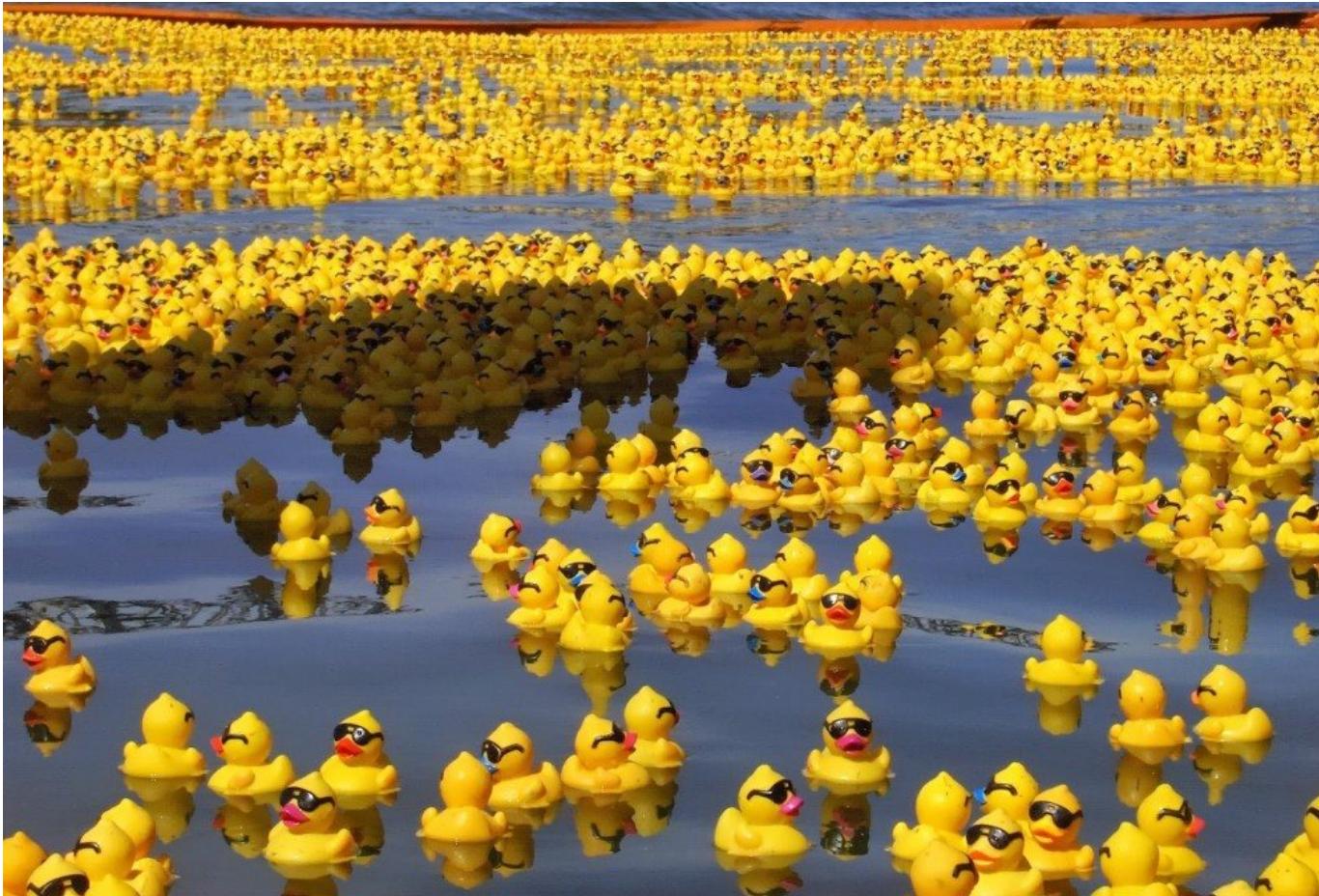
# MY GOAL

- Start with monitoring
- Metrics? Start with Up/Down -> USE || RED
- 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
- Usually much too late
- Post-mortem
- Hard to find the needle

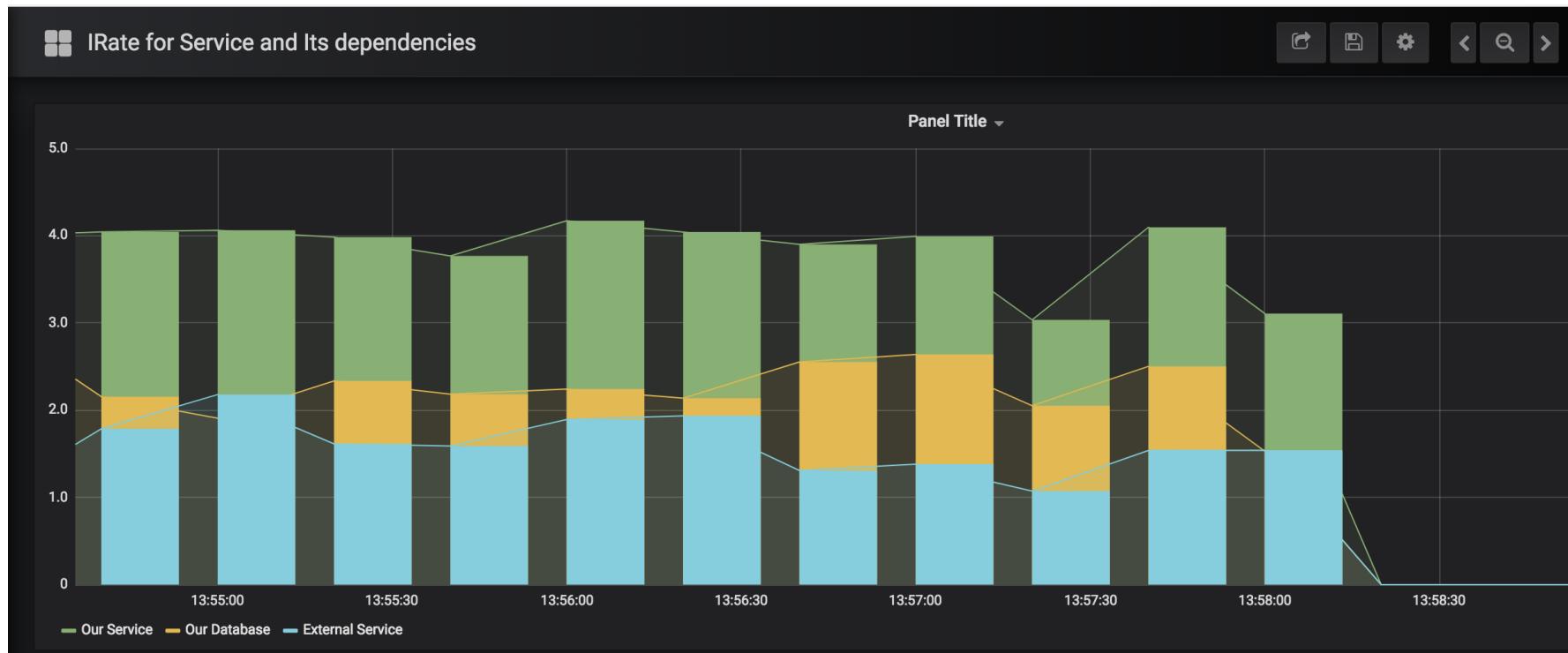
# MONITORING

- Numbers
- Trends
- Dependencies
- + Actions

## METRIC

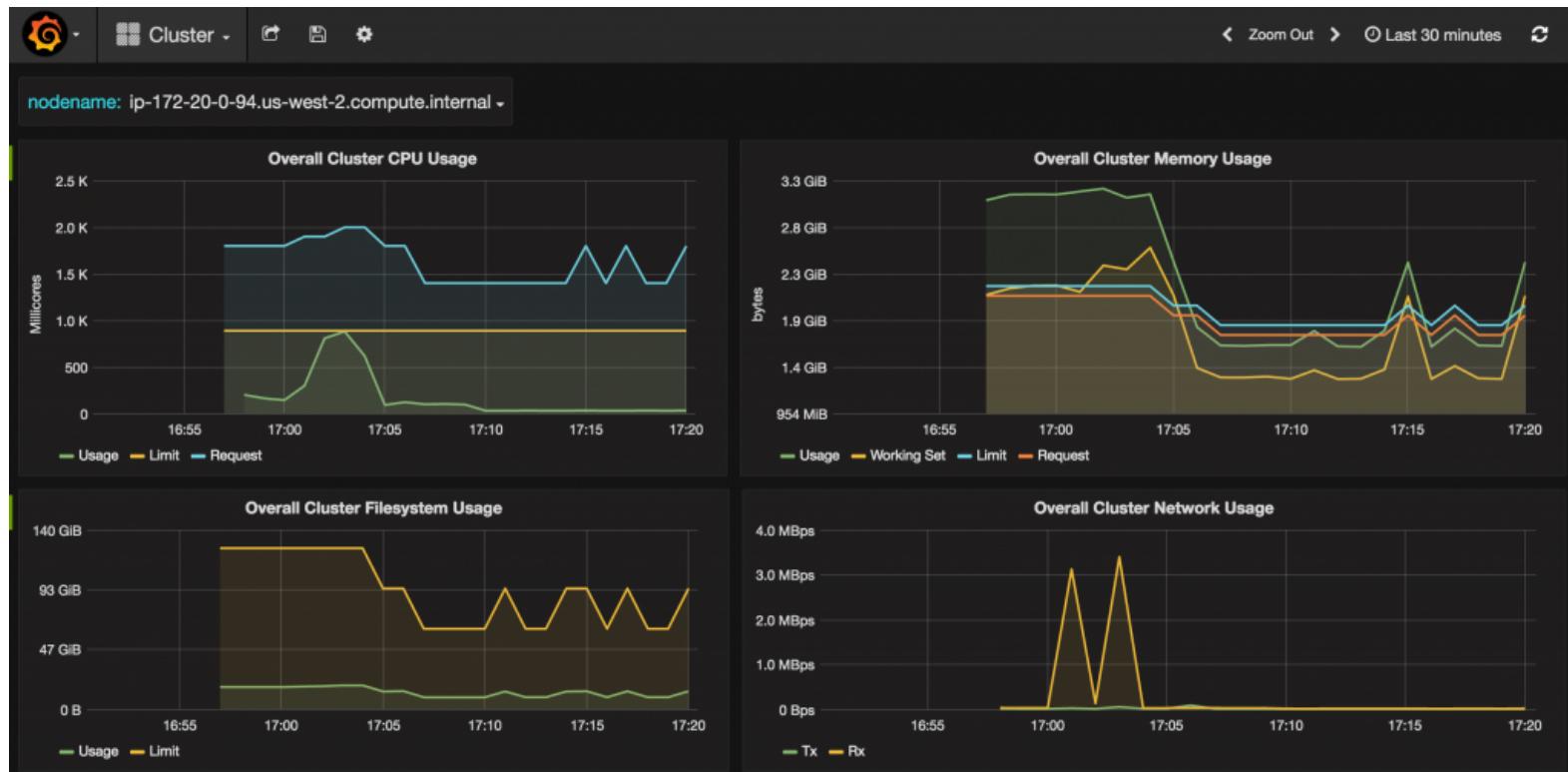
Name	Label	Value
traefik_requests_total	code="200", method="GET"	3001

# 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
  - [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*/[1]
- Pull model [2]
- Wide support for languages
- PromQL
- Integration with Kubernetes, Traefik, ...

[1] most common, *prometheus* in Spring 1.5

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

# 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 duration
# 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](#)

# PROMETHEUS PromQL

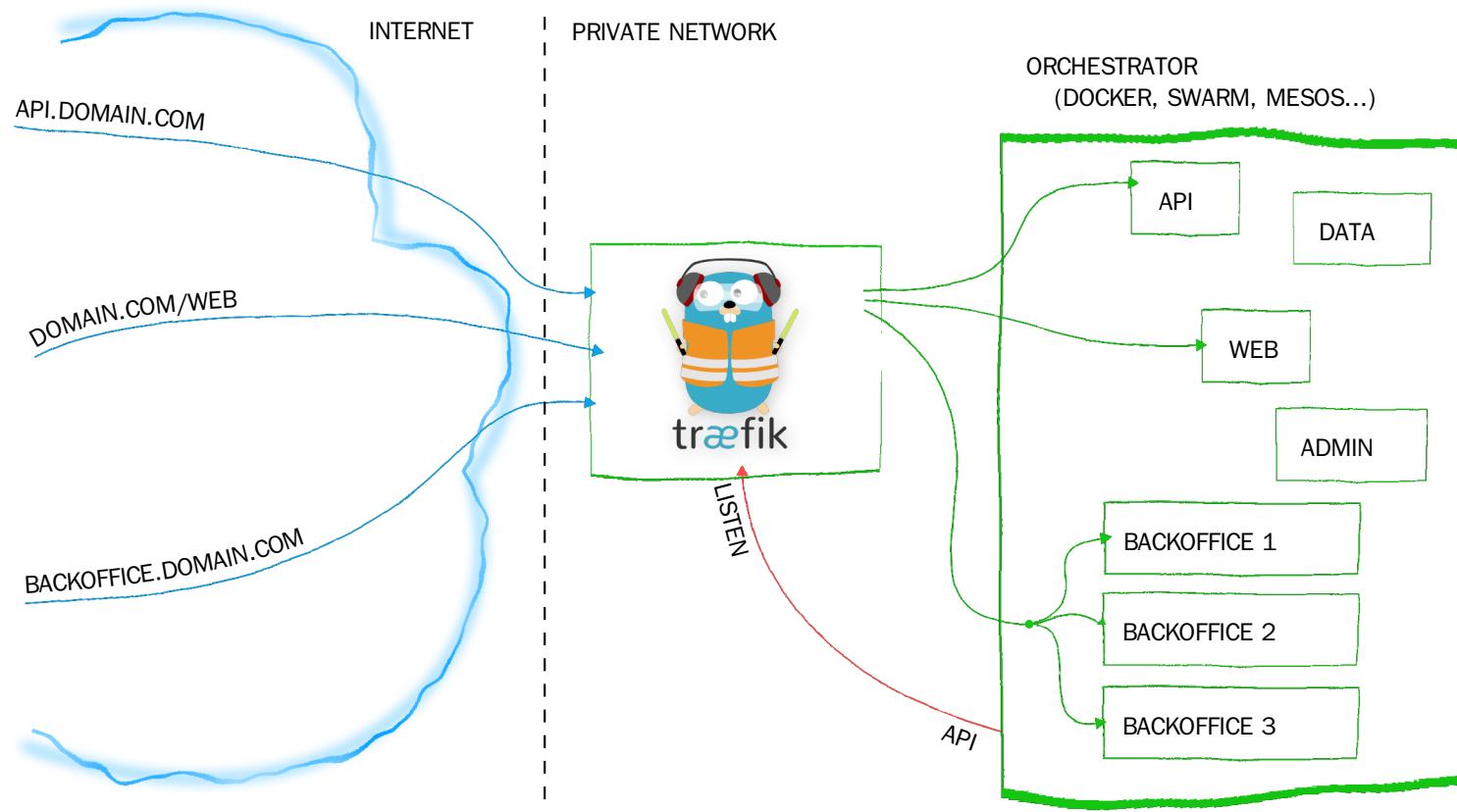
```
# working with histograms:  
  
histogram_quantile(0.9,  
    rate(http_req_duration_seconds_bucket[10m]) )  
  
# rates:  
  
rate(http_requests_total{job="api-server"} [5m])  
irate(http_requests_total{job="api-server"} [5m])  
  
# more complex:  
  
redict_linear()  
holt_winters()
```

# 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
  }
```

# CLOUD-NATIVE PROJECTS INTEGRATION



--web.metrics.prometheus

# 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 base unit `_seconds` and `_bytes`

# PROMETHEUS + JAVA



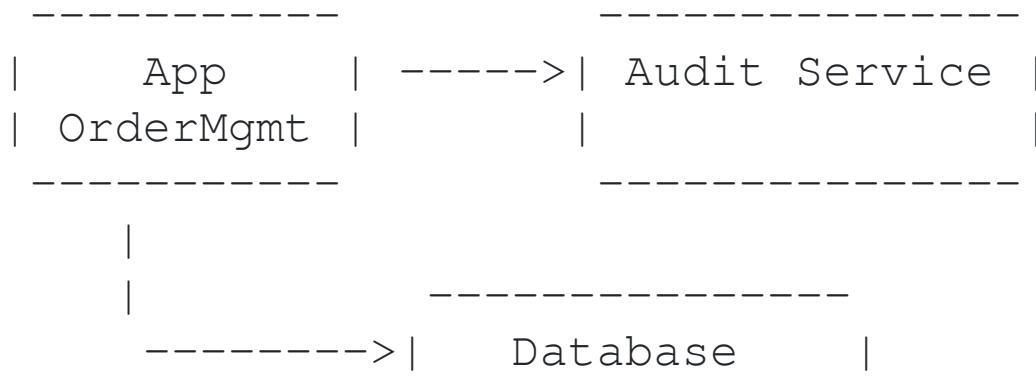
# JAVA CLIENT

- 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](https://github.com/prometheus/jmx_exporter)

# DEMO: SIMPLE REST SERVICE



## **DEMO:**

- <http://127.0.0.1:8080/hello> - service
- <http://127.0.0.1:8080/prometheus/>
- <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



# PROMETHEUS

The screenshot shows the Prometheus web interface running locally at `127.0.0.1:9090/graph`. The interface has a dark-themed header with navigation links for Prometheus, Alerts, Graph, Status, and Help. Below the header is a search bar with the placeholder "Expression (press Shift+Enter for newlines)". A blue "Execute" button is positioned next to a dropdown menu containing the text "- insert metric at cursor -". Below the search bar is a table with two columns: "Element" and "Value". The table contains a single row with the element "no data". At the bottom right of the table is a blue "Remove Graph" button. A blue "Add Graph" button is located at the bottom left of the main content area. The browser's address bar shows the URL `127.0.0.1:9090/graph`. The top of the browser window displays several open tabs, including Prometheus, Alertmanager, Grafana, and New Tab.

Element	Value
no data	

Add Graph

# PROMETHEUS

The screenshot shows the Prometheus web interface running at `127.0.0.1:9090/graph`. The interface includes a navigation bar with links for Prometheus, Alerts, Graph, Status, and Help. A sidebar on the left provides navigation options like Execute, Graph, Element, and Add Graph. The main area features a search bar labeled "Expression (press Shift+Enter for newlines)" and a dropdown menu listing various Prometheus metrics. One metric, `order_mgmt_audit_duration_seconds_count`, is highlighted with a red background. To the right of the dropdown, there is a "Value" input field and a "Remove Graph" button.

Execute

Graph

Element

no data

Add Graph

✓ - 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`

Value

Remove Graph

# PROMETHEUS

The screenshot shows a web browser window with the URL `127.0.0.1:9090/alerts`. The browser has several tabs open, including Prometheus, Alertmanager, Grafana, and New Tab. The main content area displays the Prometheus Alerts interface.

The top navigation bar includes links for Prometheus, Alerts, Graph, Status, and 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

The screenshot shows a web browser window for the Prometheus interface at `127.0.0.1:9090/alerts`. The browser has several tabs open, including Prometheus, Alertmanager, Grafana, and other Prometheus instances.

The main content area is titled "Alerts". It 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
alername="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

The screenshot shows a web browser window with the URL `127.0.0.1:9090/alerts`. The browser has several tabs open, including Prometheus, Alertmanager, Grafana, and New Tab. The main content area displays the Prometheus 'Alerts' section.

**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
<code>alertyname="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 a web browser window with multiple tabs open. The active tab is titled "Alertmanager".

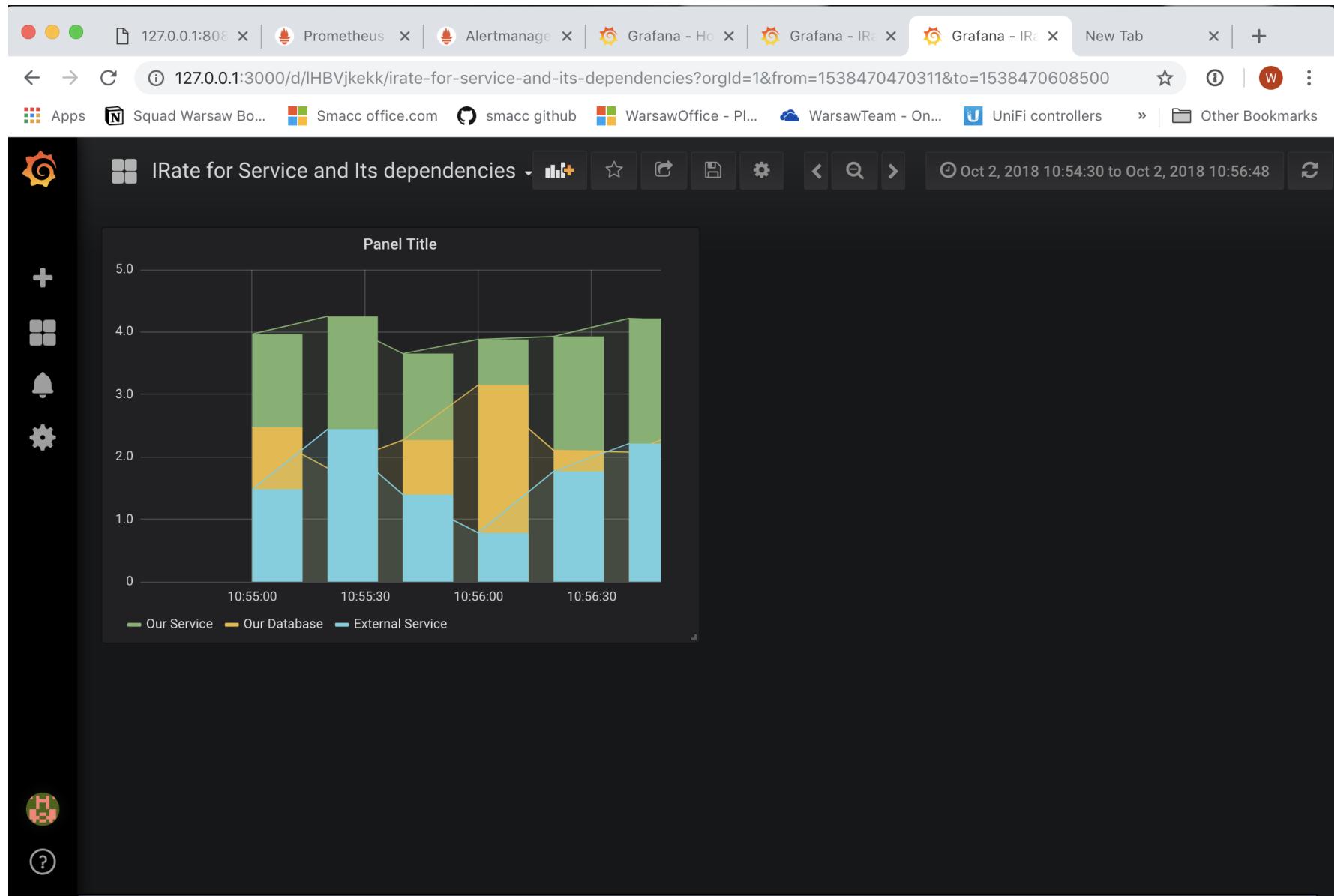
The main interface has a navigation bar with tabs: "Alertmanager", "Alerts", "Silences", and "Status". A "New Silence" button is located in the top right corner.

Below the navigation bar is a search/filter section with two tabs: "Filter" (selected) and "Group". It includes a "Custom matcher, e.g. `env="production"`" input field and a "+" button. To the right, it shows "Receiver: All" and three status filters: "Silenced" and "Inhibited".

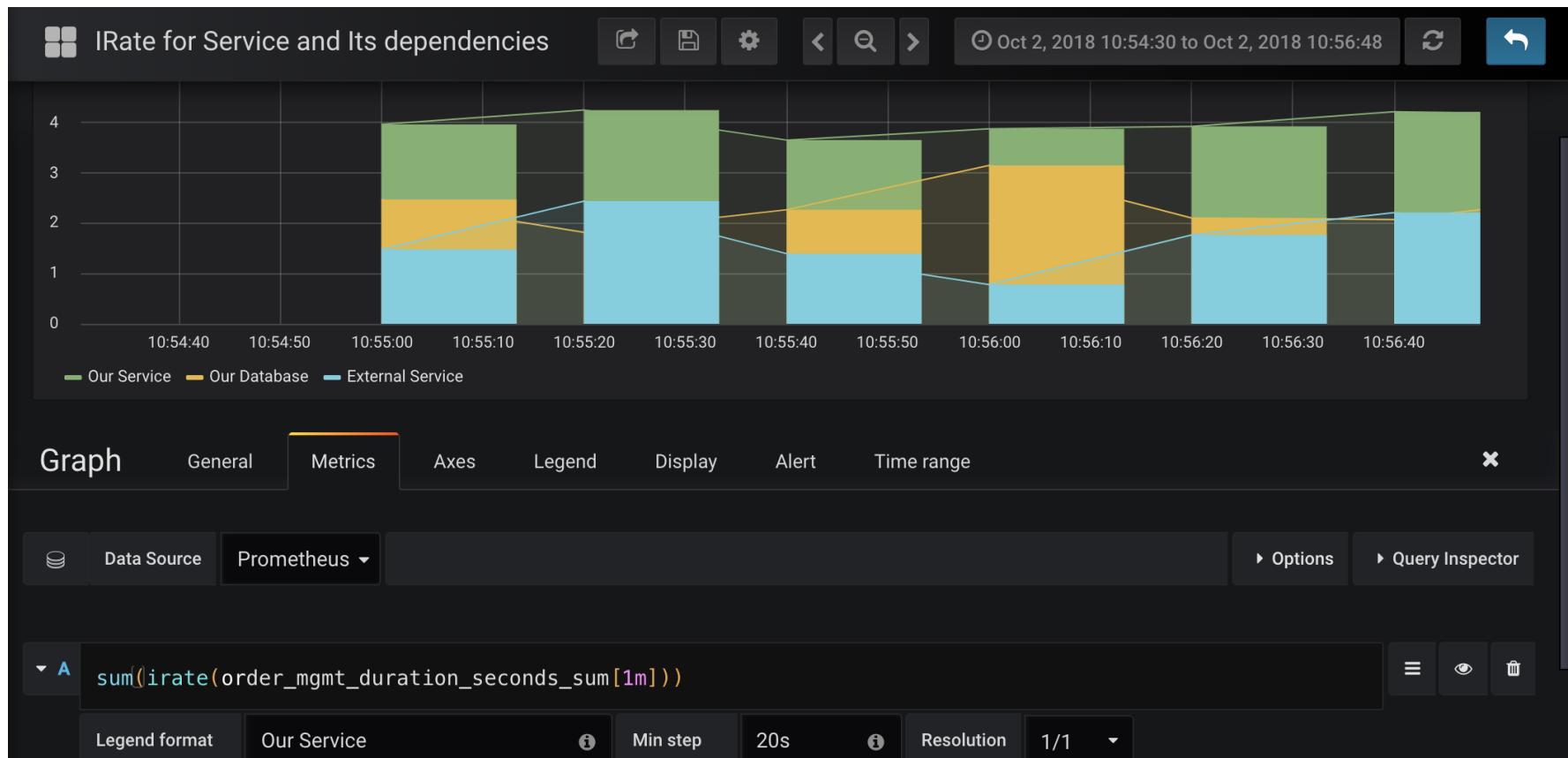
A single alert card is displayed, showing the following details:

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

# GRAFANA



# GRAFANA



# GITHUB

The screenshot shows a GitHub repository page. At the top, the URL is https://github.com/wojciech12/talk\_java\_2018\_prometheus. The repository name is displayed in blue. A search bar and navigation links for Pull requests, Issues, Marketplace, and Explore are visible. Below the header, there's a banner with the repository name and an 'Unwatch' button. A navigation bar includes links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. A message states 'No description, website, or topics provided.' There are buttons for managing topics and viewing statistics: 13 commits, 1 branch, and 0 releases. A dropdown for the branch is set to 'master'. Below the stats, there are buttons for creating a new file or uploading files. The main content area shows a commit from 'wojciech12' adding a LI link in README, followed by a list of files: demo (Update README.rst), slides (Slides), .gitignore (Demo Application), README.rst (Add LI link in README), and a detailed view of the README.rst file.

No description, website, or topics provided.

Manage topics

Branch: master ▾ New pull request Create new file Upload fi

wojciech12 Add LI link in README

demo Update README.rst

slides Slides

.gitignore Demo Application

README.rst Add LI link in README

README.rst

# DEMO

Everything Included:

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

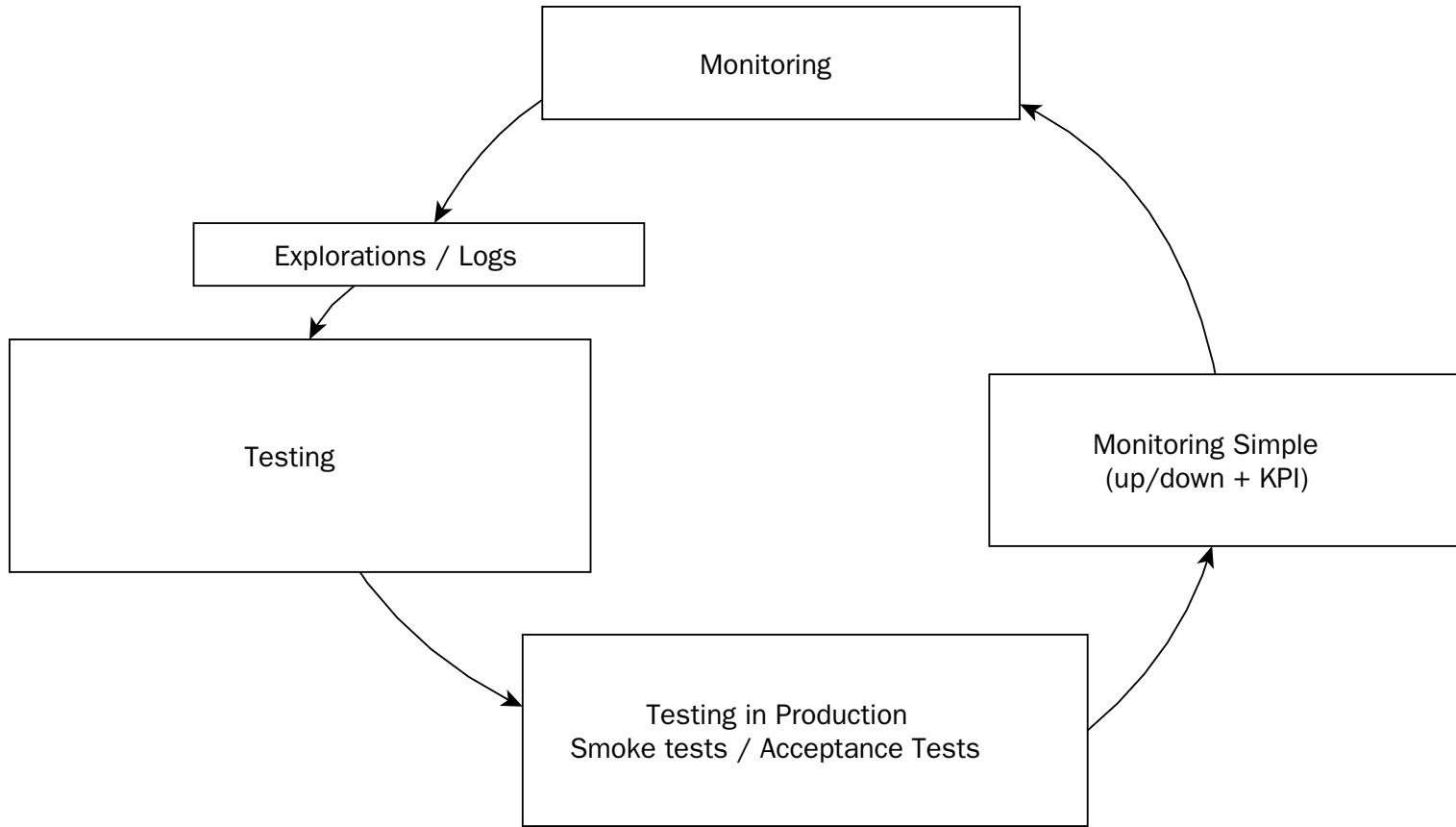
## BEST PRACTISES

- 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 == debugging vs having tests
- Logging -> high TCO

# SUMMARY



# THANK YOU

```
123 def distance_matrix(regions):  
124     """ Computes a distance matrix against a region list """  
125     tuples = [r.as_tuple() for r in regions]  
126     return cdist(tuples, tuples, region_distance)  
127  
128  
129 def clusterize(words, **kwargs):  
130     # TODO: write a cool docstring here  
131     db = DBSCAN(metric="precomputed", **kwargs)  
132     X = distance_matrix([Region.from_word(w) for w in words])  
133     labels = [int(l) for l in db.fit_predict(X)]
```



# QUESTIONS?

ps. We are hiring.

```
123 def distance_matrix(regions):  
124     """ Computes a distance matrix against a region list """  
125     tuples = [r.as_tuple() for r in regions]  
126     return cdist(tuples, tuples, region_distance)  
127  
128  
129 def clusterize(words, **kwargs):  
130     # TODO: write a cool docstring here  
131     db = DBSCAN(metric="precomputed", **kwargs)  
132     X = distance_matrix([Region.from_word(w) for w in words])  
133     labels = [int(l) for l in db.fit_predict(X)]
```



# SMACC



Go



PYTORCH

TensorFlow™



amazon  
web services™

Azure

λ



# BACKUP SLIDES

```
123 def distance_matrix(regions):  
124     """ Computes a distance matrix against a region list """  
125     tuples = [r.as_tuple() for r in regions]  
126     return cdist(tuples, tuples, region_distance)  
127  
128  
129 def clusterize(words, **kwargs):  
130     # TODO: write a cool docstring here  
131     db = DBSCAN(metric="precomputed", **kwargs)  
132     X = distance_matrix([Region.from_word(w) for w in words])  
133     labels = [int(l) for l in db.fit_predict(X)]
```



# 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"
  annotations:
```

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