DEVOPS WROCŁAW MEETUP #7

Prometheus 101: get on the hype train

Michał Bolek - OPS@Cogniance

TODO:

- ŻŻP
- What is prometheus?
- Architecture
- Short comparison
- Lifelike topics
- Demo

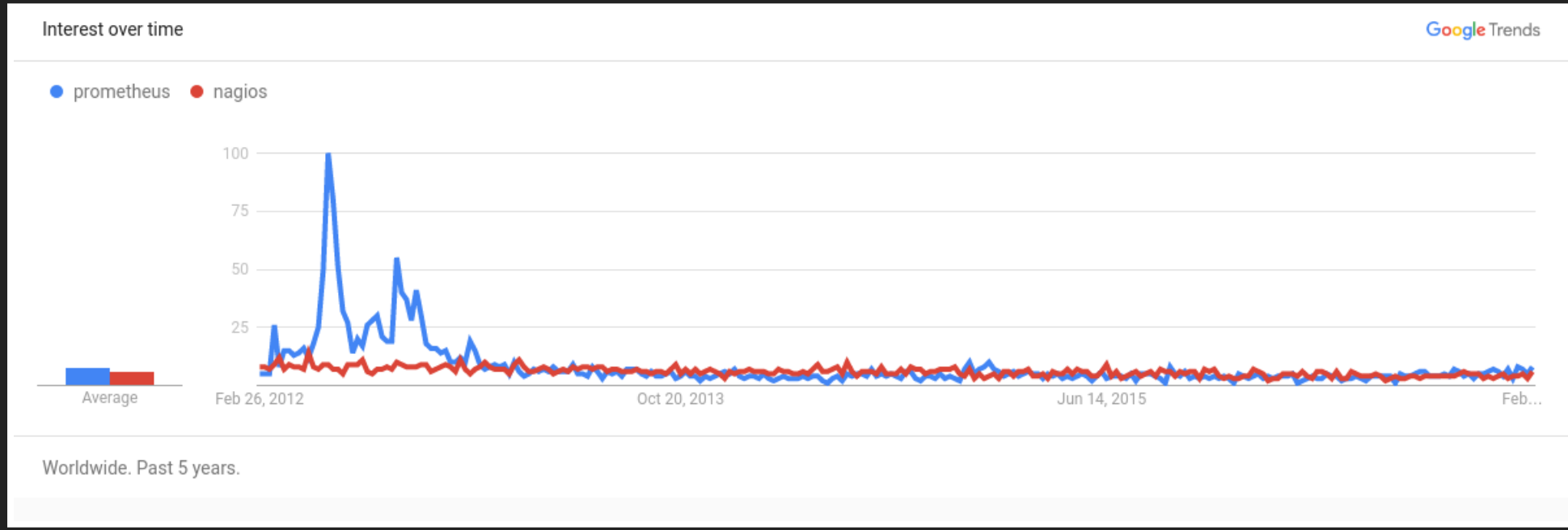
PROMETHEUS: YESTERDAY'S TECHNOLOGY TODAY

Even though Borgmon remains internal to Google, the idea of treating time-series data as a data source for generating alerts is now accessible to everyone through those open source tools like Prometheus [...]

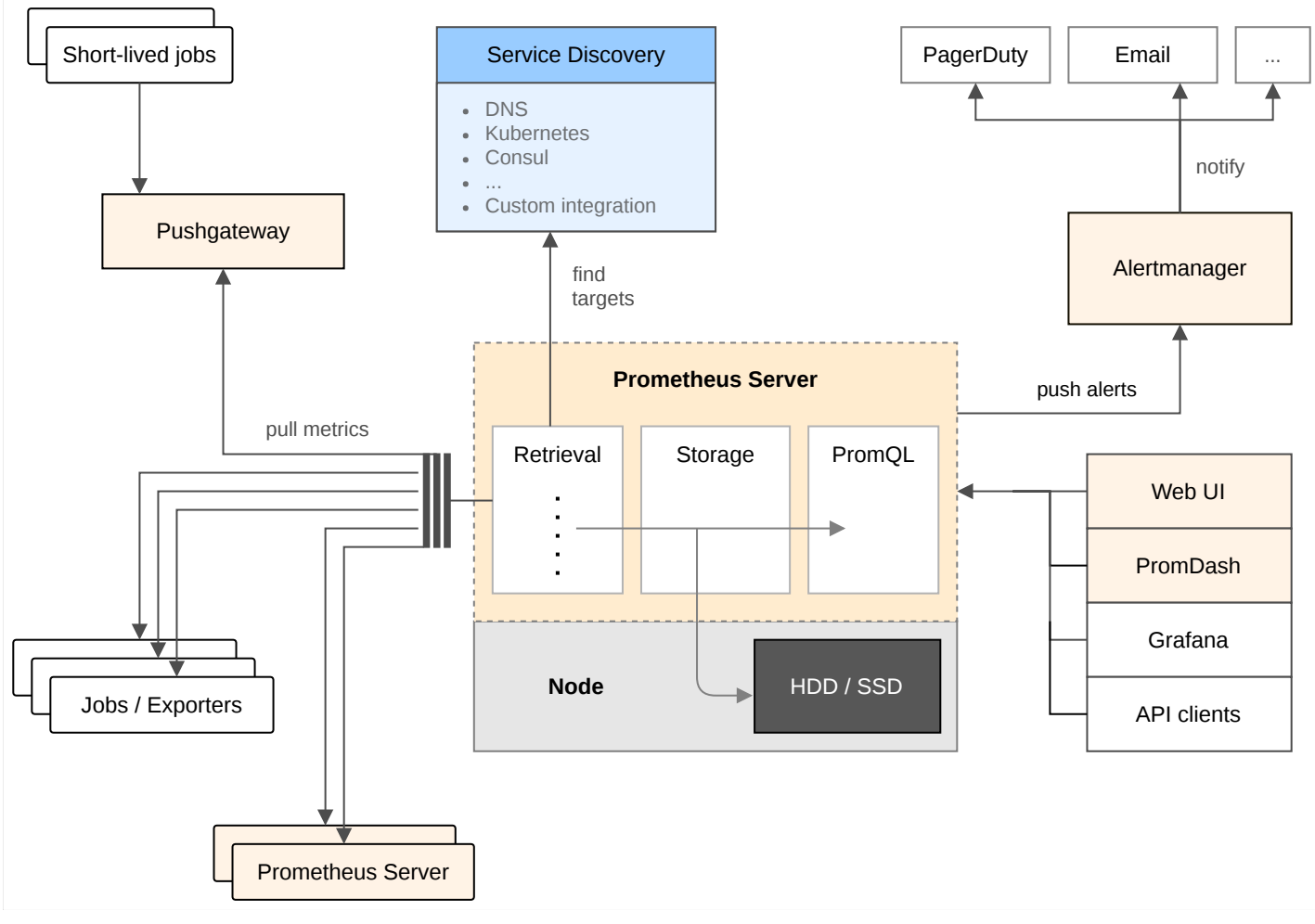
WHAT IS PROMETHEUS ?

- www.prometheus.io
- 'quite' old monitoring and alerting toolkit - started around 2012
- built at SoundCloud (by some of the exgooglers)
- scrapes and stores time series data
- and then uses them for monitoring
- pull based (but can push if needed)
- standalone
- PromQL
- joined the Cloud Native Computing Foundation in 2016 just after Kubernetes...

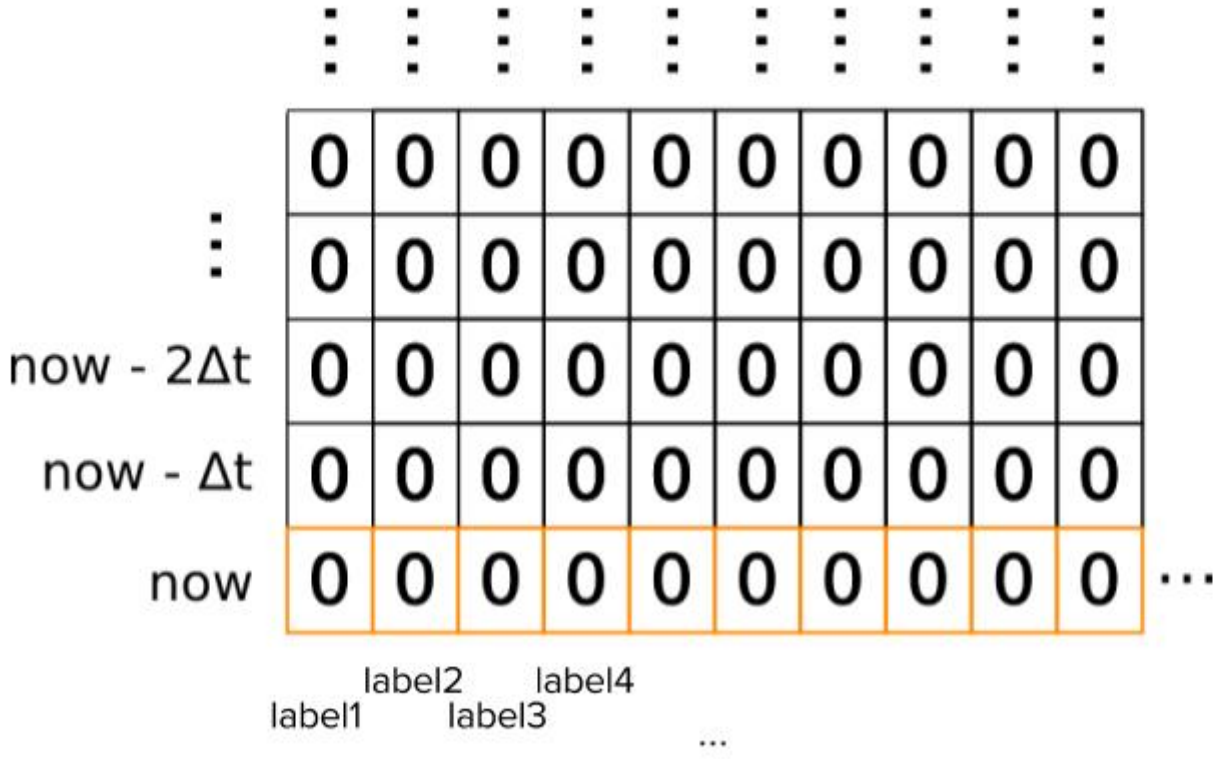
TRENDS IN SEARCH:



ARCHITECTURE



Structure of timeseries



api_http_requests_total{method="GET", endpoint="/api/", status="200"} - 123

PROMETHEUS SERVER

- scrapes endpoints' /metrics
- saves data
- run queries

SERVICE DISCOVERY

- DNS
- consul
- kubernetes
- etcd
- EC2
- marathon
- static file

BASIC PROMETHEUS CONFIG

```
global:
  scrape_interval:     15s # By default, scrape targets every 15s.

  # Attach these labels to any time series or alerts when communicating
        # with external systems (federation, remote storage, Alertmanager).
  external_labels:
    monitor: 'codelab-monitor'

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=`
  - job_name: 'prometheus'

# Override and scrape targets from this job every 5 seconds.
    scrape_interval: 5s

    static_configs:
      - targets: ['localhost:9090']
```

ALERTMANAGER

CONFIG

```
global:

route:
  group_by: ['alertname', 'cluster']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 1h
  continue: true
  receiver: 'default-slack'

receivers:
  - name: 'default-slack'
    slack_configs:
      - api_url: 'https://hooks.slack.com/services/wc4l3n13pR4wD21wy70k3n'
        send_resolved: true
        channel: '#monitoring'
        username: 'Prometheus'
        icon_emoji: ':crystal_ball:'
        text: {% raw %}"{{ .CommonAnnotations.description }}" {% endraw %}
```

Prometheus APP 11:27 AM

[FIRING:1] BackendNotificationFail (https://[redacted])

[redacted] - notifications issue

Prometheus APP 11:49 AM

[FIRING:5] InstanceDown (maven)

Prometheus APP 11:58 AM

[FIRING:3] InstanceLowMem (node maven)

Prometheus APP 12:32 PM

[FIRING:1] BackendNotificationFail (http[redacted])

[redacted] - notifications issue

Prometheus APP 12:54 PM

[FIRING:5] InstanceDown (maven)

Prometheus APP 1:03 PM

[FIRING:3] InstanceLowMem (node maven)

Prometheus APP 1:26 PM

[RESOLVED] InstanceSlowIO (10.2.1.188:9100 maven)
10.2.1.188:9100 slow IO - 1234.4

[FIRING:1] InstanceSlowIO (10.2.1.188:9100 maven)
10.2.1.188:9100 slow IO - 426.1333333333333

[RESOLVED] InstanceSlowIO (10.2.1.188:9100 maven)
10.2.1.188:9100 slow IO - 426.1333333333333

[FIRING:1] BackendNotificationFail (https://[redacted])

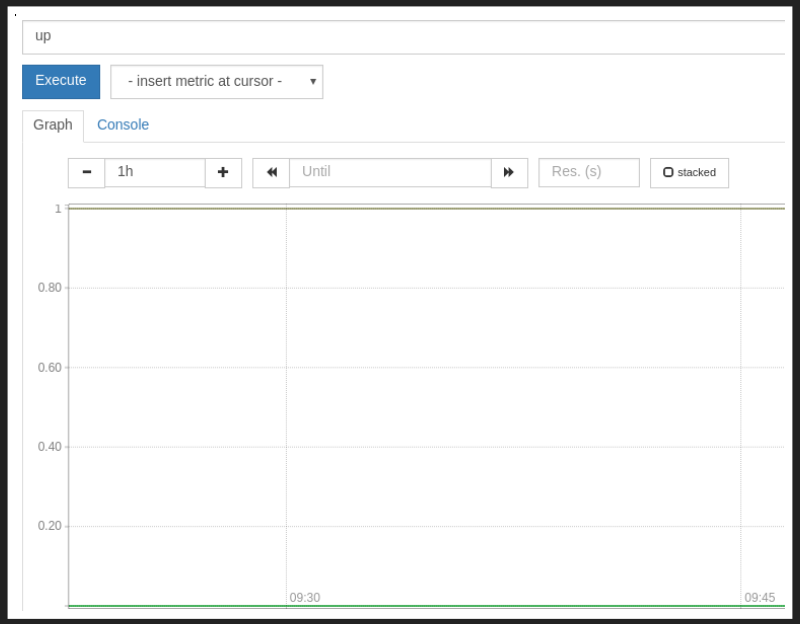
[redacted] - notifications issue

EXPORTERS

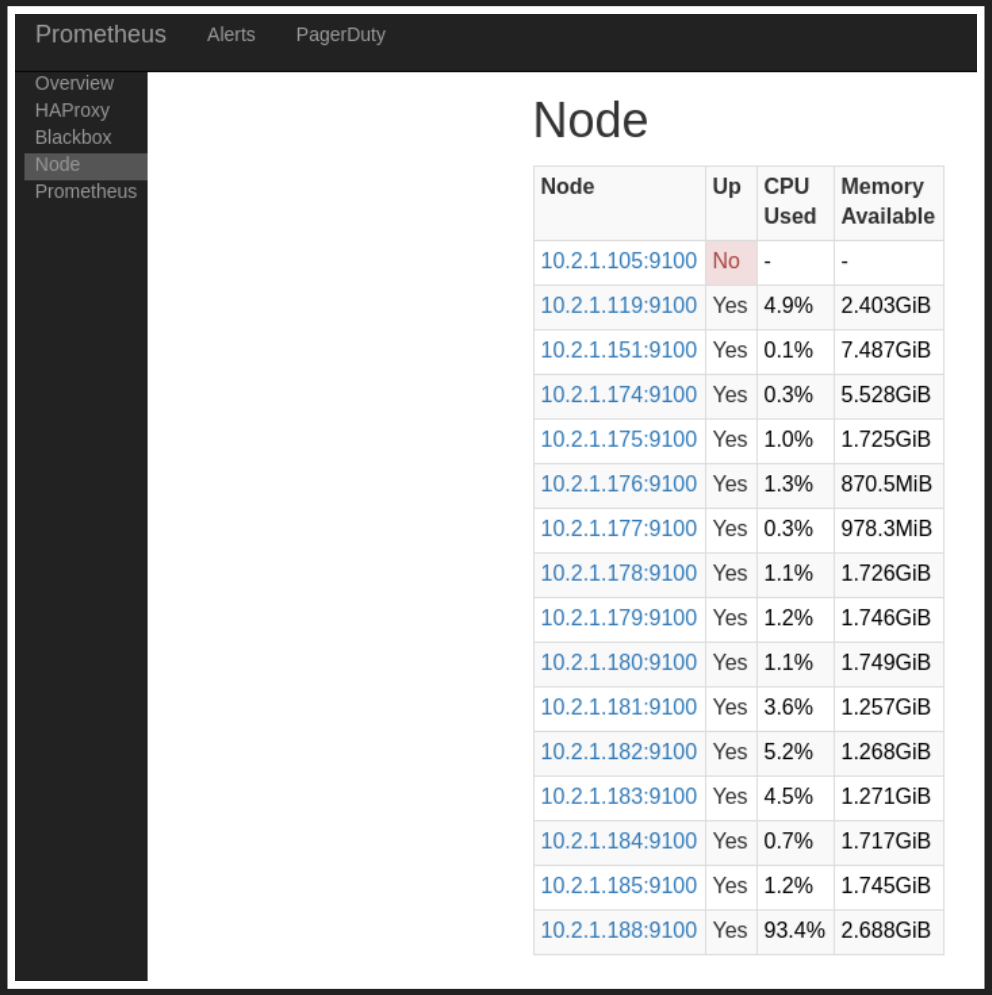
- node_exporter
- Apache/Nginx/Varnish/haproxy_exporter
- MySQL/PostgreSQL/MongoDB/Redis_exporter
- AWS/Cloudflare/DockerHub/GitHub/Openweatherma
- collectd/graphite/munin/statsd_exporter
- jenkins/rtorrent/minecraft_exporter/blackbox
- JSON_exporter
- +native apps

```
- job_name: 'blackbox'
  metrics_path: /probe
  params:
    module: [module_name]  # Look for a HTTP 200 response.
  static_configs:
    - targets:
        - ['127.0.0.1:9999']

  relabel_configs:
    - source_labels: [__address__]
      regex: (.*?)(:80)?
      target_label: __param_target
      replacement: https://${1}
    - source_labels: [__param_target]
      target_label: instance
    - target_label: __address__
      replacement: 127.0.0.1:9115  # Blackbox exporter.
```

PROMETHEUS EXPRESSION BROWSER

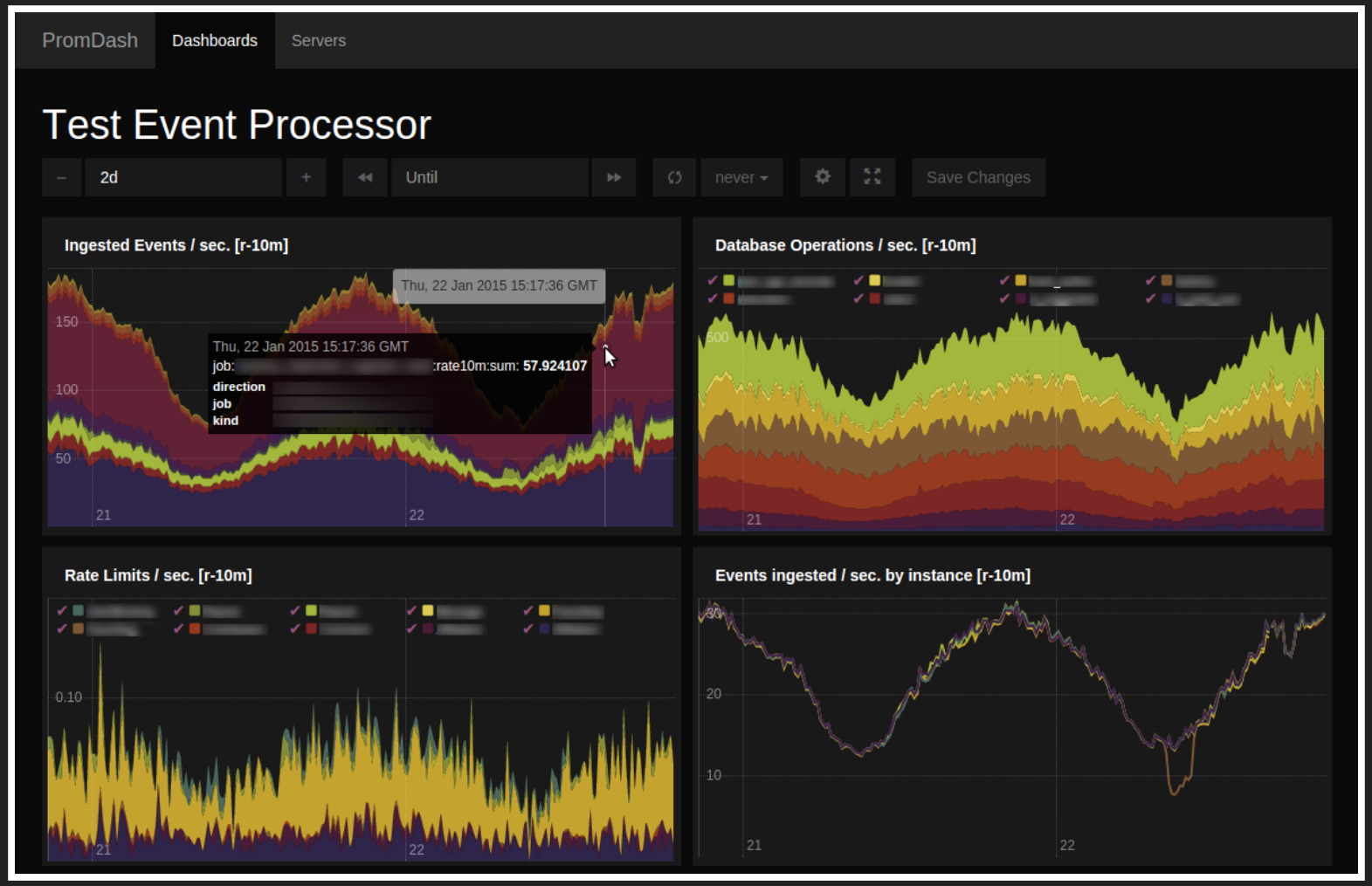


CONSOLES

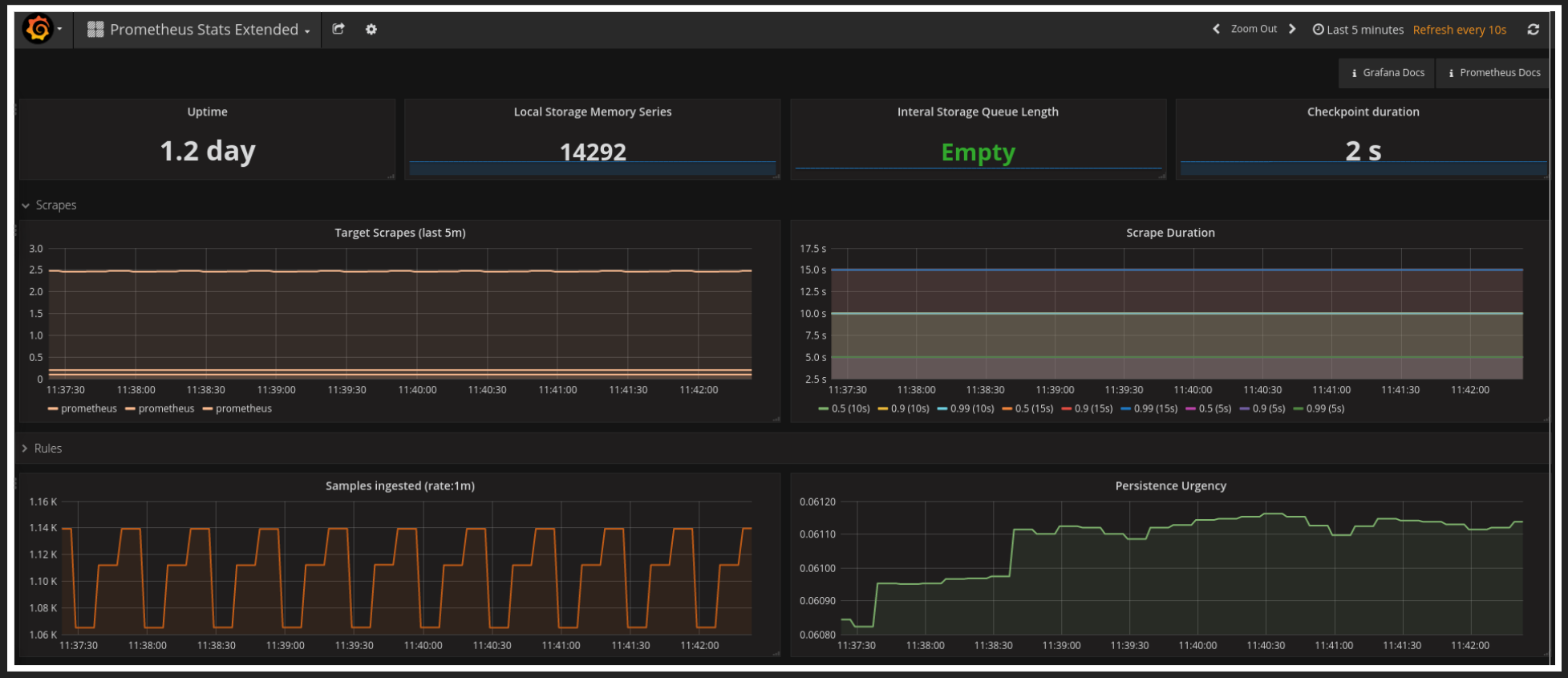


Node	Up	CPU Used	Memory Available
10.2.1.105:9100	No	-	-
10.2.1.119:9100	Yes	4.9%	2.403GiB
10.2.1.151:9100	Yes	0.1%	7.487GiB
10.2.1.174:9100	Yes	0.3%	5.528GiB
10.2.1.175:9100	Yes	1.0%	1.725GiB
10.2.1.176:9100	Yes	1.3%	870.5MiB
10.2.1.177:9100	Yes	0.3%	978.3MiB
10.2.1.178:9100	Yes	1.1%	1.726GiB
10.2.1.179:9100	Yes	1.2%	1.746GiB
10.2.1.180:9100	Yes	1.1%	1.749GiB
10.2.1.181:9100	Yes	3.6%	1.257GiB
10.2.1.182:9100	Yes	5.2%	1.268GiB
10.2.1.183:9100	Yes	4.5%	1.271GiB
10.2.1.184:9100	Yes	0.7%	1.717GiB
10.2.1.185:9100	Yes	1.2%	1.745GiB
10.2.1.188:9100	Yes	93.4%	2.688GiB

PROMDASH


PromDash
Dashboards
Servers
Test Event Processor
2d
Until
never
Save Changes
Ingested Events / sec. [r-10m]
Thu, 22 Jan 2015 15:17:36 GMT
150
100
50
21
22
Thu, 22 Jan 2015 15:17:36 GMT
job:
:rate10m:sum: 57.924107
direction
job
kind
Database Operations / sec. [r-10m]
500
21
22
Rate Limits / sec. [r-10m]
0.10
21
22
Events ingested / sec. by instance [r-10m]
30
20
10
21
22

GRAFANA


Prometheus Stats Extended
Zoom Out
Last 5 minutes
Refresh every 10s
Grafana Docs
Prometheus Docs
Uptime
1.2 day
Local Storage Memory Series
14292
Interal Storage Queue Length
Empty
Checkpoint duration
2 s
Scrapes
Target Scrapes (last 5m)
3.0
2.5
2.0
1.5
1.0
0.5
0
11:37:30
11:38:00
11:38:30
11:39:00
11:39:30
11:40:00
11:40:30
11:41:00
11:41:30
11:42:00
prometheus
prometheus
prometheus
Scrape Duration
17.5 s
15.0 s
12.5 s
10.0 s
7.5 s
5.0 s
2.5 s
11:37:30
11:38:00
11:38:30
11:39:00
11:39:30
11:40:00
11:40:30
11:41:00
11:41:30
11:42:00
0.5 (10s)
0.9 (10s)
0.99 (10s)
0.5 (15s)
0.9 (15s)
0.99 (15s)
0.5 (5s)
0.9 (5s)
0.99 (5s)
Rules
Samples ingested (rate:1m)
1.16 K
1.14 K
1.12 K
1.10 K
1.08 K
1.06 K
11:37:30
11:38:00
11:38:30
11:39:00
11:39:30
11:40:00
11:40:30
11:41:00
11:41:30
11:42:00
Persistence Urgency
0.06120
0.06110
0.06100
0.06090
0.06080
11:37:30
11:38:00
11:38:30
11:39:00
11:39:30
11:40:00
11:40:30
11:41:00
11:41:30
11:42:00

COMPARISON WITH GRAPHITE

Prometheus	Graphite
full monitoring	time series DB
rich metadata model	simple metadata model
LevelDB	whisper

COMPARISON WITH INFLUXDB

Prometheus	InfluxDB
full monitoring	time series DB
rich metadata model	even richer metadata model
LevelDB	distributed cluster storage

COMPARISON WITH OPENTSDB

Prometheus	OpenTSDB
full monitoring	time series DB
rich metadata model	rich metadata model
LevelDB	Hadoop/HBase

COMPARISON WITH NAGIOS

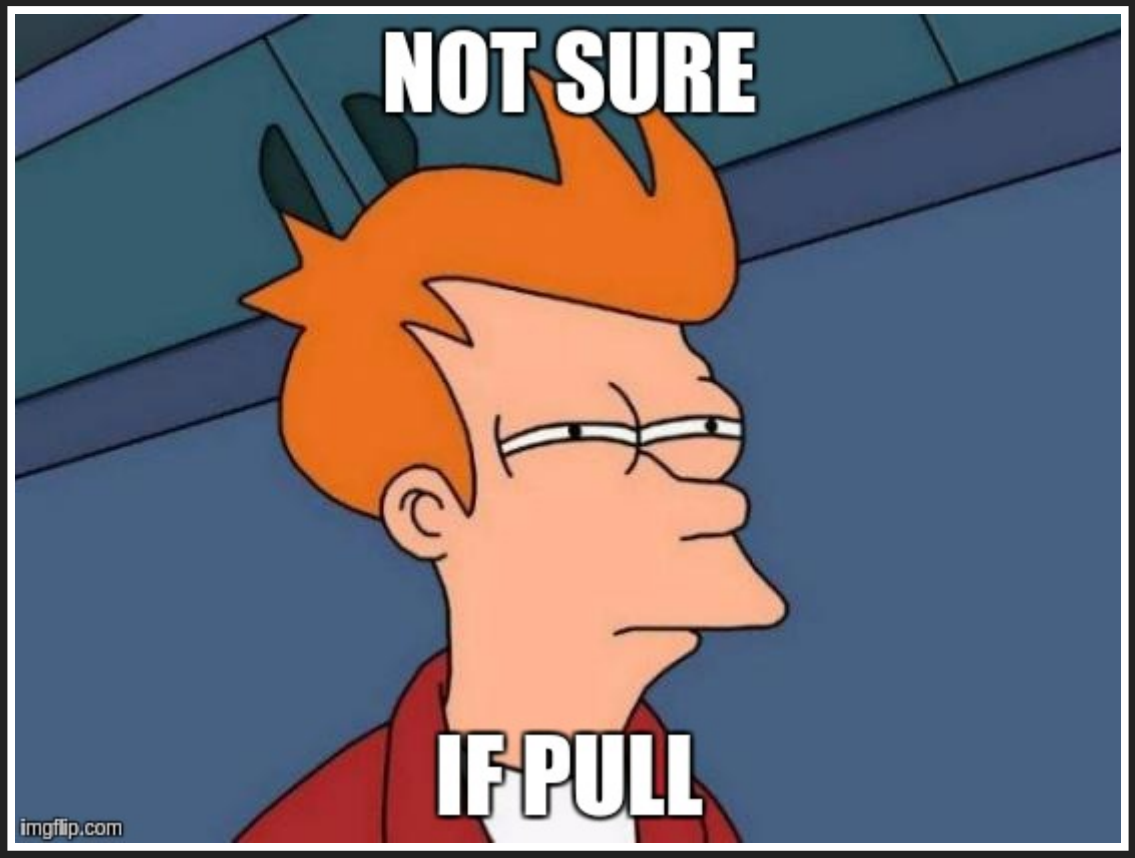
Prometheus	Nagios
full monitoring solution	full monitoring solution
easy automation	hard automation
multi-component	monolithic

COMPARISON WITH SENSU

Prometheus	Sensu
full monitoring solution	full monitoring solution
easy automation	easy automation
multi-component	distributed

LIFELIKE TOPICS

- Pull model, is it any good?
- Nagging disks
- Expiring certs
- Need for status



Pull is ok!:)

NAGGING DISKS

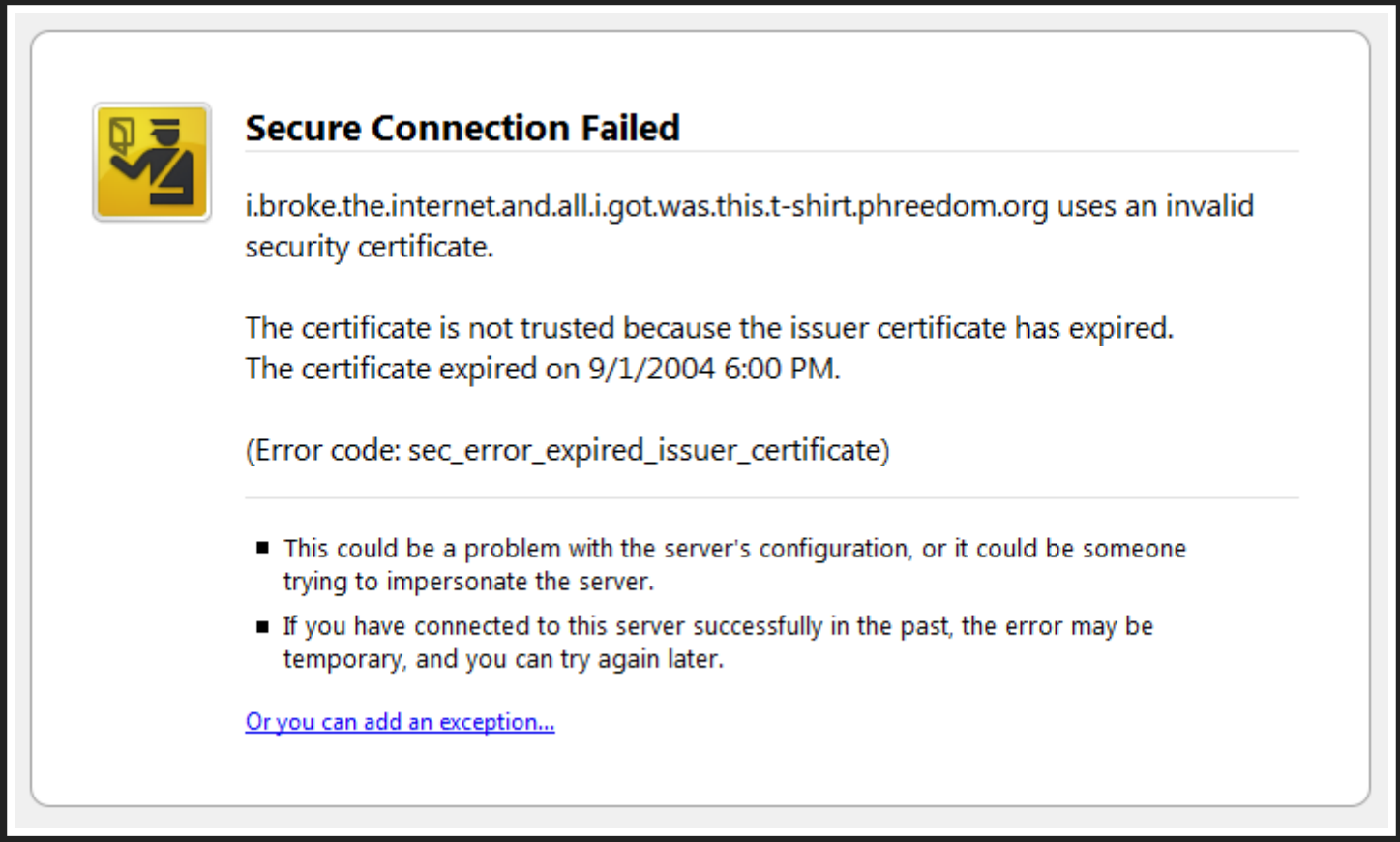
Service State Information

Current Status:	WARNING (for 1d 6h 48m 55s)
Status Information:	DISK WARNING - free space: / 21505 MB (19% inode=97%):
Performance Data:	/=86231MB;90207;101483;0;112759
Current Attempt:	3/3 (HARD state)
Last Check Time:	02-23-2017 15:01:26
Check Type:	ACTIVE
Check Latency / Duration:	0.000 / 0.137 seconds
Next Scheduled Check:	02-23-2017 15:06:26
Last State Change:	02-22-2017 08:16:34
Last Notification:	02-22-2017 08:16:34 (notification 3)
Is This Service Flapping?	NO (0.00% state change)
In Scheduled Downtime?	NO
Last Update:	02-23-2017 15:05:28 (0d 0h 0m 1s ago)

NOTIFY IN ADVANCE!

```
ALERT DiskWillFillIn24Hours
  IF predict_linear(node_filesystem_free{job='node'}[1d], 24*3600) < 0
  FOR 5m
  ANNOTATIONS {
    summary = "Instance {{$labels.instance}} disk fillup in 24h!",
    description = "{{$labels.instance}}s on of the disks fill fill up in 24h!"
  }
```

EXPIRING CERTS



NEVA' AGAIN!

```
ALERT SSLCertExpiringSoon
IF probe_ssl_earliest_cert_expiry{job="blackbox"} - time() < 86400 * 30
FOR 10m
ANNOTATIONS {
  summary = "SSL Cert on {{ $labels.instance}} will expire in less than 30 days",
  description = "SSL Cert on {{ $labels.instance}} will expire in less than 30 days"
}
```

NEED FOR STATUS

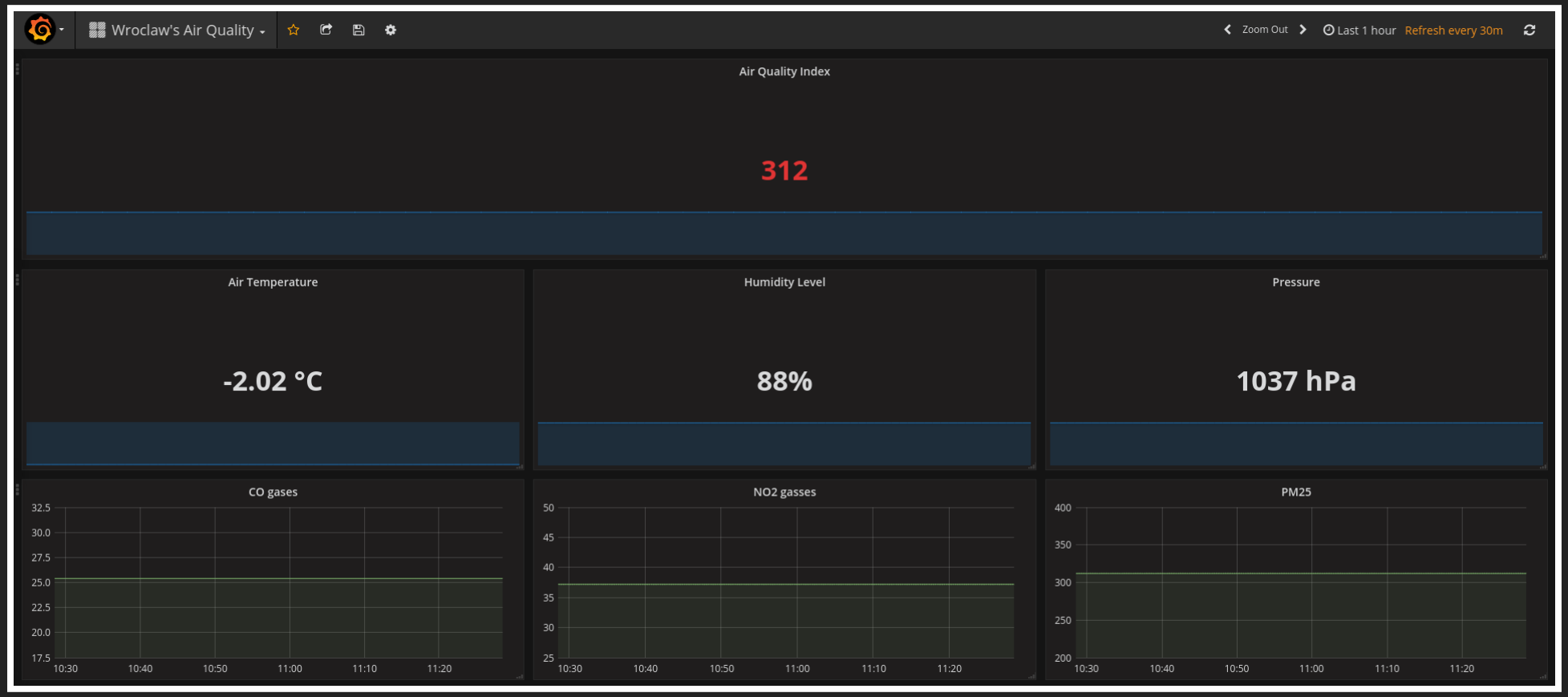
{"smsSuccesses":[],"smsFailures":[{"version":1,"phoneNumber":"+1555123XXXX","dateTime":"2017-02-23T13:10:00Z"},{"version":1,"phoneNumber":"+1555123XXXX","dateTime":"2017-02-23T13:13:00Z"}],"successCount":0,"failureCount":2}

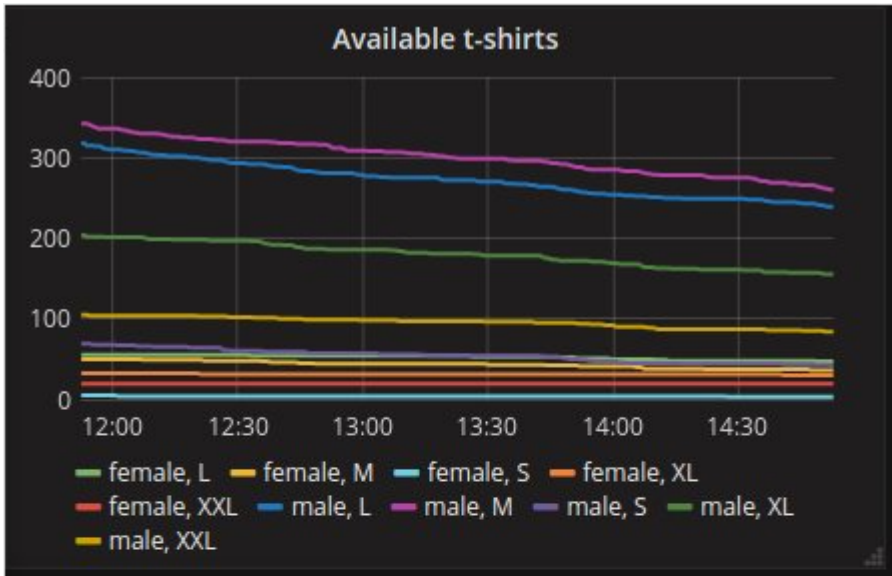
NEED FOR STATUS

```
mvn_backend_notifications:
        prober: http
        timeout: 5s
        http:
                method: GET
                valid_status_codes: [200]
                no_follow_redirects: false
                fail_if_ssl: false
                fail_if_not_ssl: true
                fail_if_not_matches_regexp:
                - '\"failureCount\"\:0
```

SIMPLE EXPORTER

```
# -*- coding: UTF-8 -*-
from prometheus_client import start_http_server, Metric, REGISTRY
import json
import requests
import sys
import time

class JsonCollector(object):
def __init__(self):
  pass

def collect(self):
  token = 'asdadsadadasdadsasdasd' #apply for a token here: http://aqicn.org/data-platform/token/
  url = 'http://api.waqi.info/feed/'
  city = 'Wrocław' # Cities: http://aqicn.org/city/all/
  response = json.loads(requests.get(url+city+"/?token="+token).content.decode('Utf-8'))

  metric = Metric('aqi_airquality', 'Air Quality Index', 'gauge')
  metric.add_sample('aqi_airquality', value=response['data']['aqi'], labels={})
  yield metric

if __name__ == '__main__':
start_http_server(int(sys.argv[1]))
REGISTRY.register(JsonCollector())

while True: time.sleep(1)
```


Wroclaw's Air Quality
Zoom Out
Last 1 hour
Refresh every 30m
Air Quality Index
312
Air Temperature
-2.02 °C
Humidity Level
88%
Pressure
1037 hPa
CO gases
32.5
30.0
27.5
25.0
22.5
20.0
17.5
10:30
10:40
10:50
11:00
11:10
11:20
NO2 gasses
50
45
40
35
30
25
10:30
10:40
10:50
11:00
11:10
11:20
PM25
400
350
300
250
200
10:30
10:40
10:50
11:00
11:10
11:20



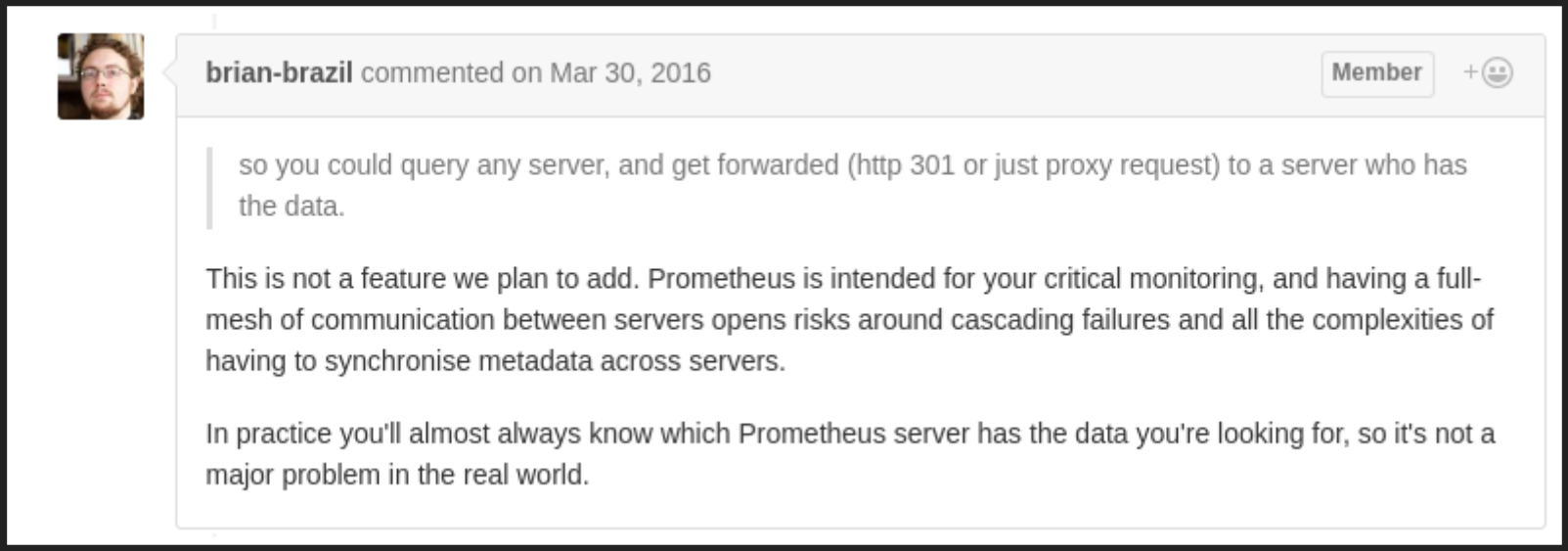
Robust Perception @RobustPerceiver · Feb 4

See t-shirt sales graphs at @fosdem live, with @PrometheusIO and @grafana: dashboard.fosdem.org/dashboard/db/s…
pic.twitter.com/42pThF1tWO

1 39 48

HA

ha ha ha...



brian-brazil commented on Mar 30, 2016 — Member

> so you could query any server, and get forwarded (http 301 or just proxy request) to a server who has the data.

This is not a feature we plan to add. Prometheus is intended for your critical monitoring, and having a full-mesh of communication between servers opens risks around cascading failures and all the complexities of having to synchronise metadata across servers.

In practice you'll almost always know which Prometheus server has the data you're looking for, so it's not a major problem in the real world.

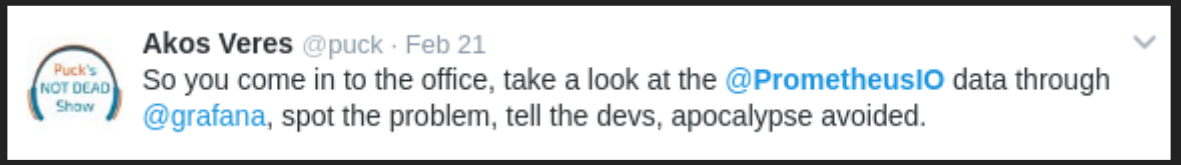
https://www.robustperception.io/scaling-and-federating-prometheus/

https://www.robustperception.io/federation-what-is-it-good-for/

DEMO

Q'S

and possibly A's



LINKS:

- www.prometheus.io
- www.robustperception.io
- 1st post on prometheus ujeb.se/1stpost
- SRE - https://landing.google.com/sre/book/index.html
- Cool presentation on Prom - http://ujeb.se/prometheus
- Scaling to a million machines https://www.youtube.com/watch?v=likpVWB5Lvo
- https://www.robustperception.io/scaling-and-federating-prometheus/
- Prometheus-as-a-service https://github.com/weaveworks/cortex
- Design and Philosophy https://www.youtube.com/watch?v=QgJbxCWRZ1s
- Push vs Pull http://www.boxever.com/push-vs-pull-for-monitoring/

THANK YOU!