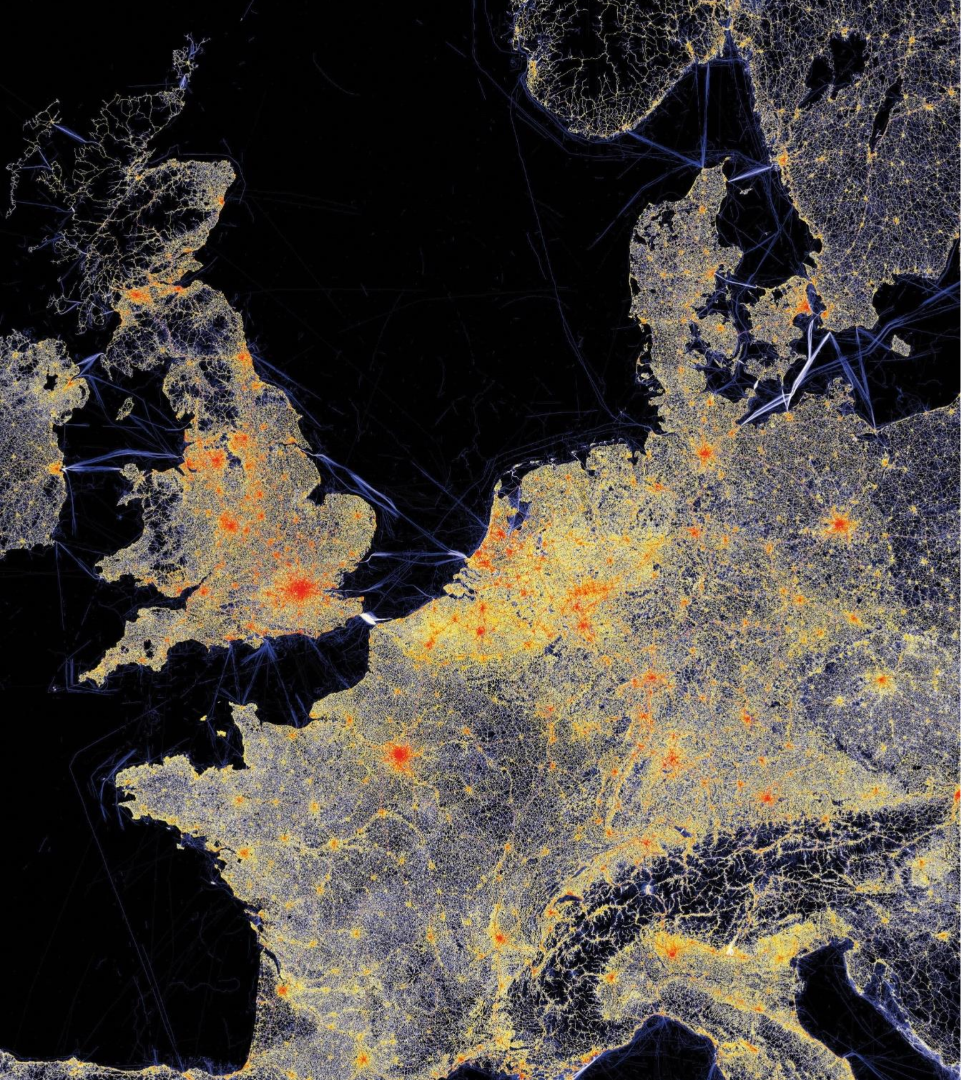
Using Prometheus and Grafana for PostgreSQL monitoring in TomTom

KEEPING THE WORLD MOVING

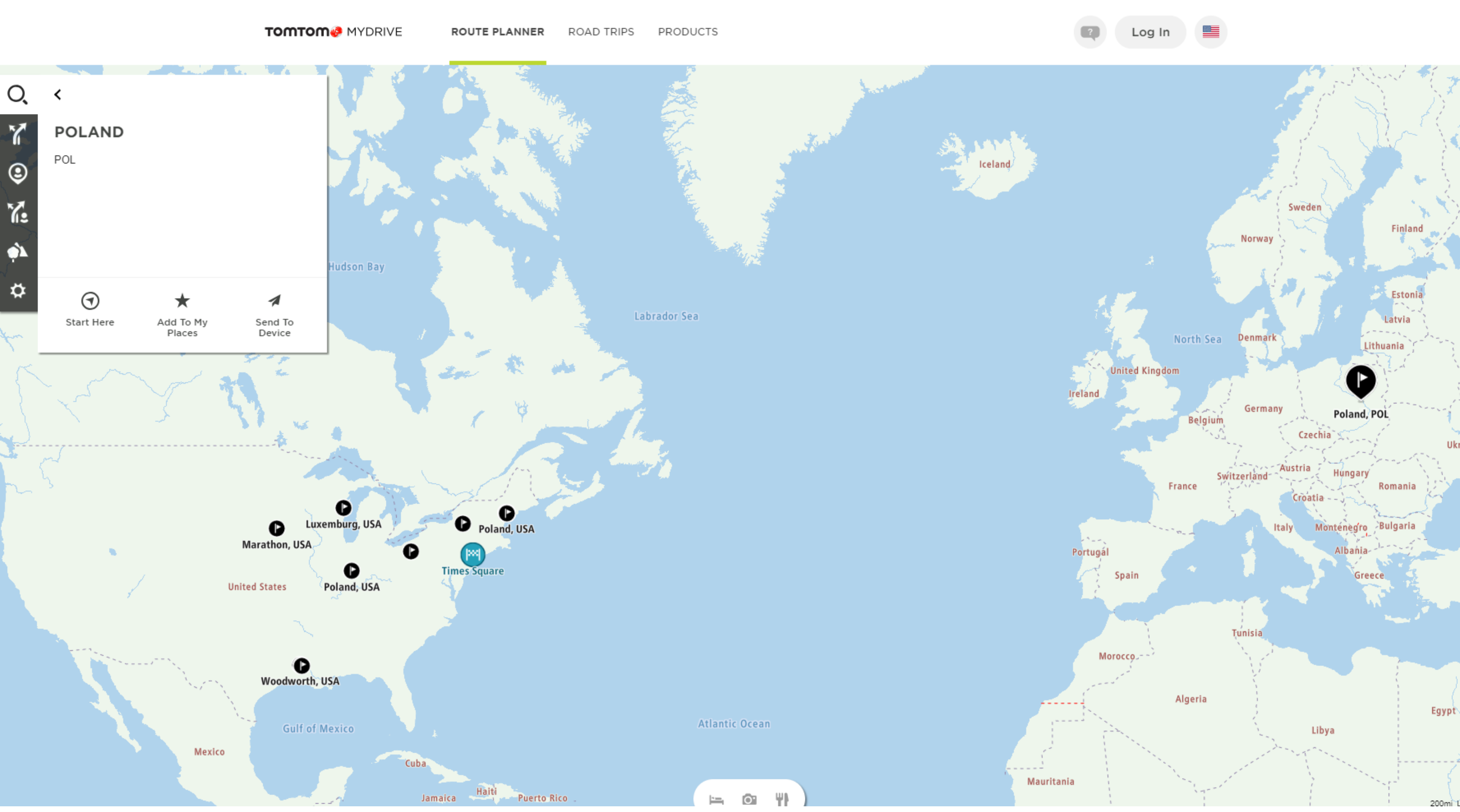
TOMTOM



Agenda

- Who are we?
- Prometheus
  - Architecture
  - Collecting and Quering data
  - Alerting
  - High Availability
  - Installation and configuration
- Grafana
  - Data sources
  - Dashboards

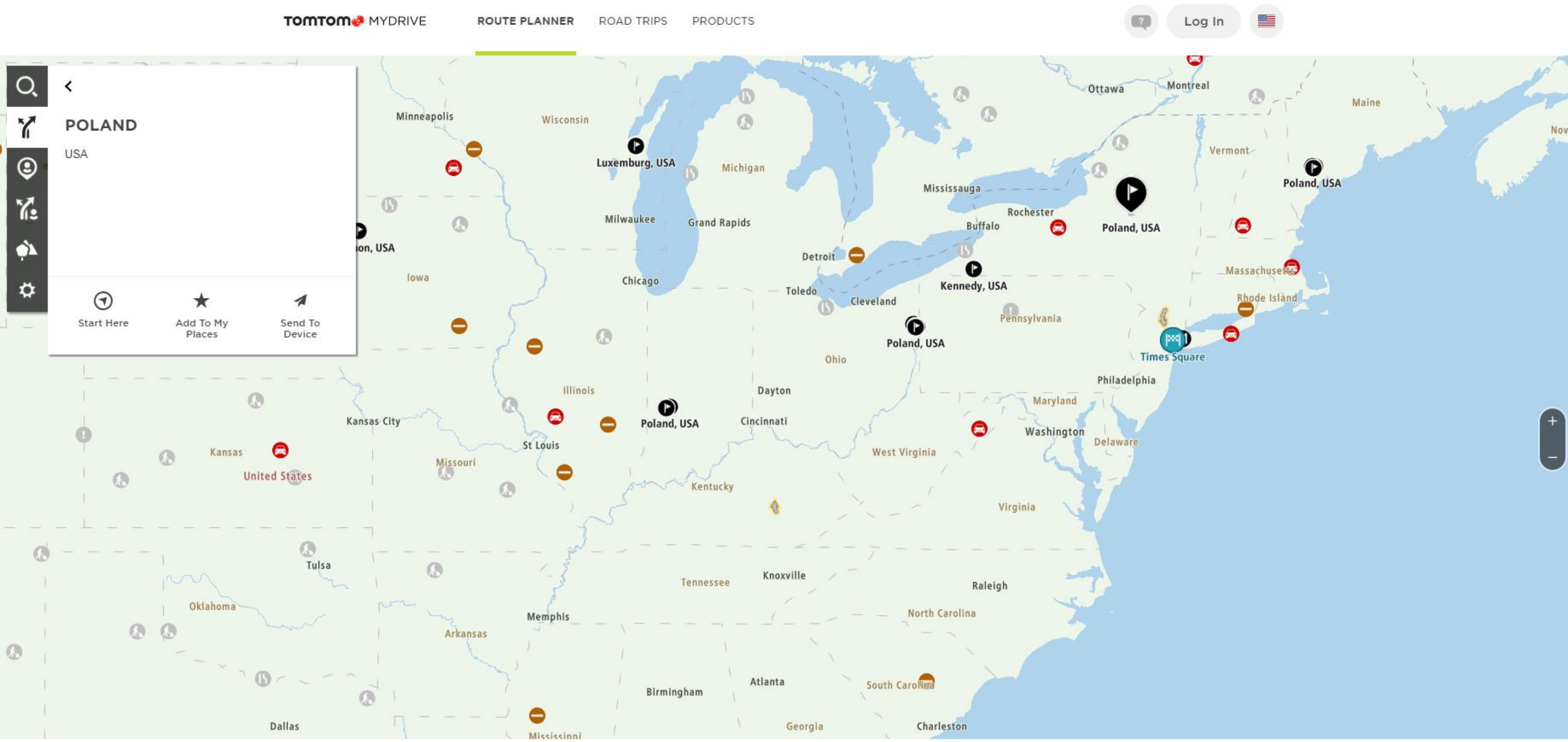
- Summary


TOMTOM MYDRIVE ROUTE PLANNER ROAD TRIPS PRODUCTS Log In

POLAND

POL

Start Here | Add To My Places | Send To Device


TOMTOM MYDRIVE
ROUTE PLANNER
ROAD TRIPS
PRODUCTS
Log In
POLAND
USA
Start Here
Add To My Places
Send To Device
Luxemburg, USA
Poland, USA
Kennedy, USA
Poland, USA
Poland, USA
Poland, USA
Times Square
Minneapolis
Wisconsin
Michigan
Milwaukee
Grand Rapids
Chicago
Detroit
Toledo
Cleveland
Mississauga
Rochester
Buffalo
Ottawa
Montreal
Maine
Vermont
Massachusetts
Rhode Island
Pennsylvania
Philadelphia
Ohio
Dayton
Cincinnati
Iowa
Illinois
Kansas City
St Louis
Kansas
United States
Missouri
Kentucky
West Virginia
Maryland
Washington
Delaware
Virginia
Tulsa
Oklahoma
Tennessee
Knoxville
Raleigh
North Carolina
Memphis
Arkansas
Birmingham
Atlanta
South Carolina
Dallas
Georgia
Charleston
Mississippi

About us

- Rafał Hawrylak

  rafal.hawrylak@tomtom.com

  Software developer and database expert

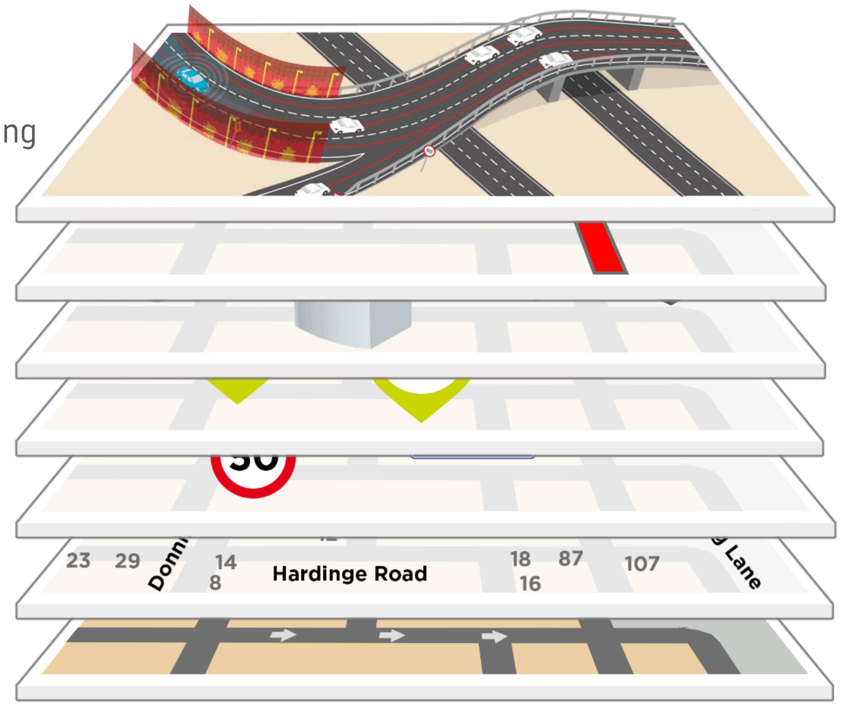
- Michał Gutkowski

  michal.gutkowski@tomtom.com

  Software engineer solving problems with Java, Python, Bash… and SQL

Leading independent location technology specialist, shaping mobility with highly accurate maps, navigation software, real-time traffic information and services.

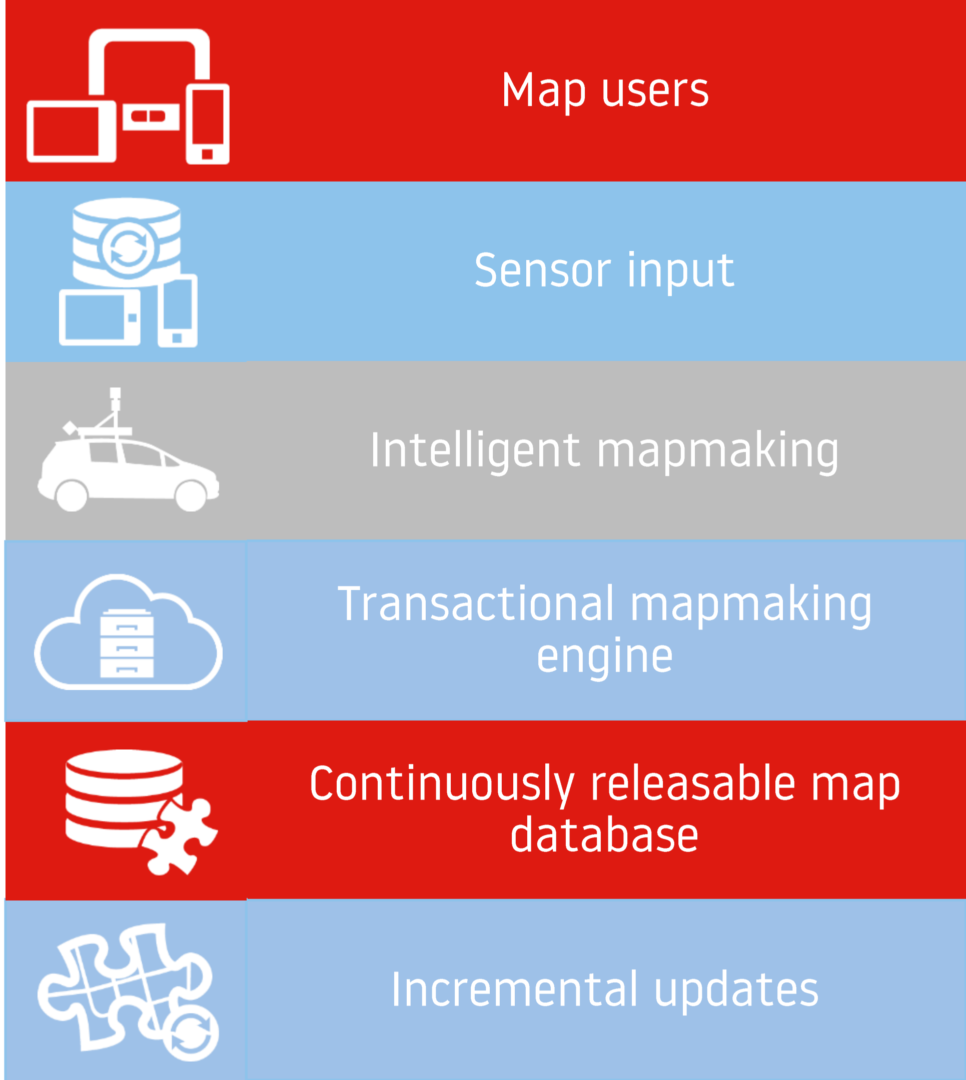
Sophisticated & detailed maps



Automated driving	HD & RoadDNA
Driver assistance	Advanced Driver Assistance System
Visualization	3D
Search	Poi
Navigation	Traffic signs
Geocoding	Addressing
Routing	Road network

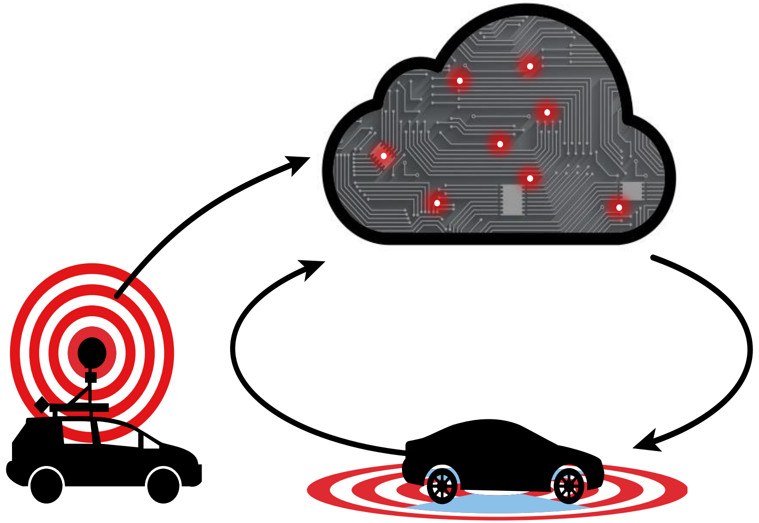
Map-making platform in 2019

- PostgreSQL + Postgis
- Data sharding and scalable reads
- 150TB of live data
- Daily db size increase: 400GB – up to 15k rows / sec
- Daily db transfers: 200TB – up to 500k queries / sec


Map users
Sensor input
Intelligent mapmaking
Transactional mapmaking engine
Continuously releasable map database
Incremental updates

Why monitoring is important?

- System health-check and maintenance
- Alerting and reliable notification system
- Detect performance regression
- Measure optimizations – software or business process
- Best value for money – maximum utilization
- Adjust business processes – self healing system

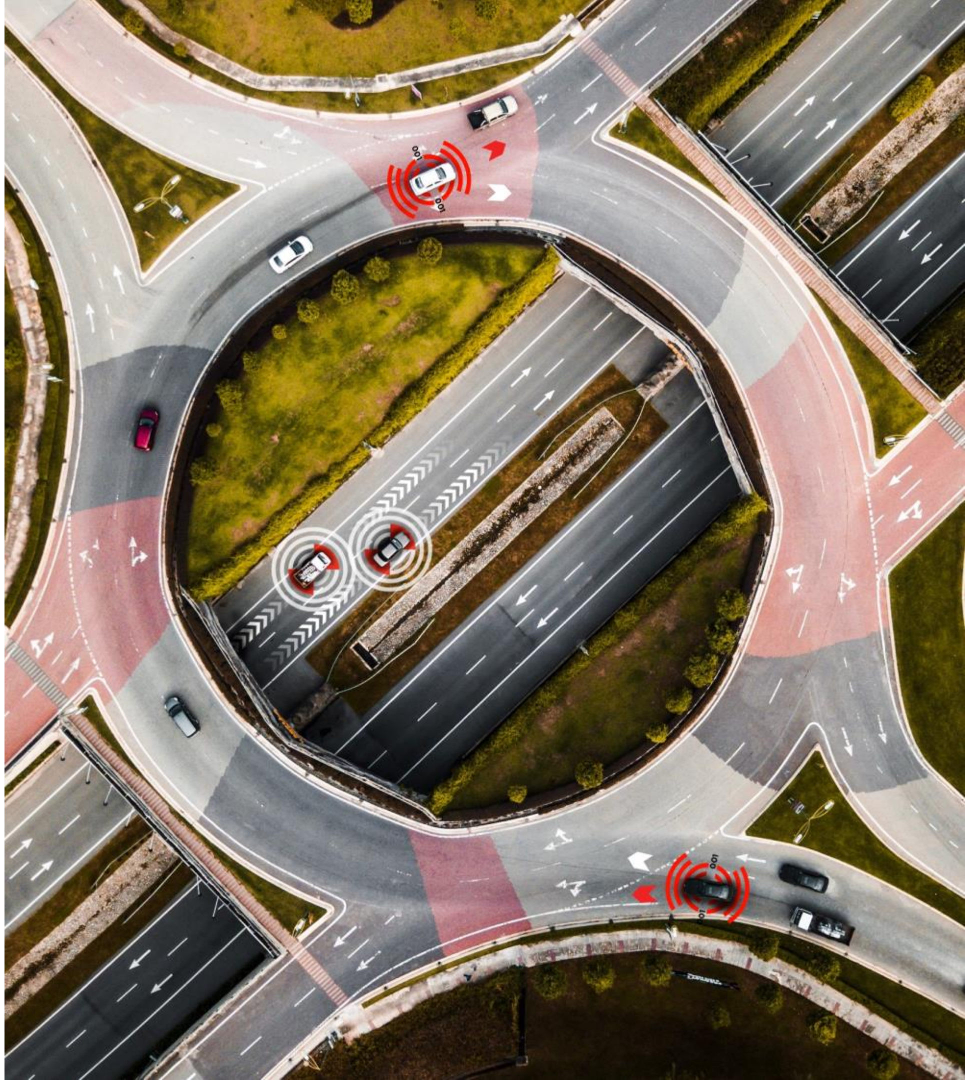


TOMTOM

Prometheus

Prometheus

What is it?

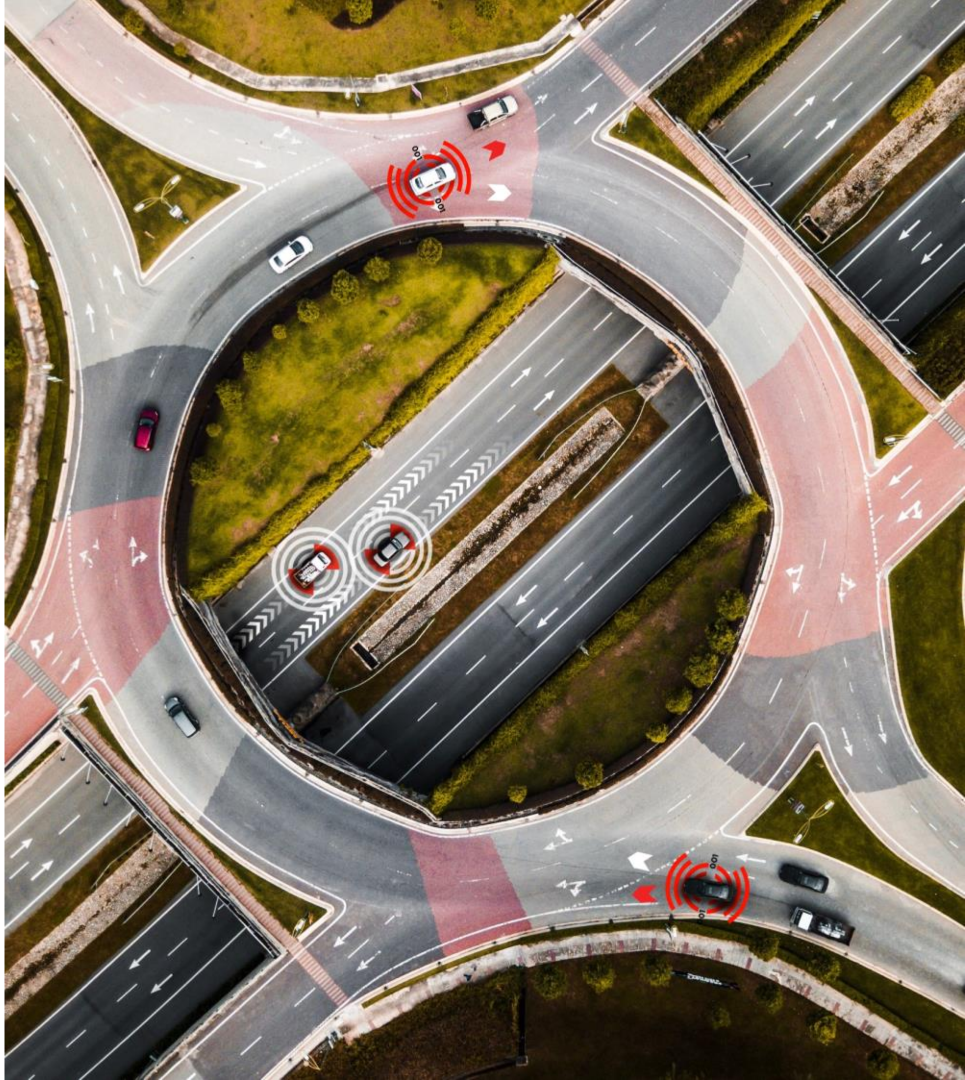


Prometheus

Prometheus

What is it?

- Open-source (Apache 2.0 License) monitoring toolkit https://prometheus.io
- Metrics collection
- Metrics storage in built-in time series database
- Flexible query language (promQL)
- Alerting with alert-manager



Prometheus

What is it?

Metric examples:

```
#TYPE metric_name gauge
426689100 metric_name{label1="value1",label2="value2"} 89
426689100 metric_name{label1="value3",label2="value4"} 110
426689160 metric_name{label1="value3",label2="value4"} 32
```

Prometheus

What is it?

Metric examples:

```
#TYPE metric_name gauge
426689100 metric_name{label1="value1",label2="value2"} 89
426689100 metric_name{label1="value3",label2="value4"} 110
426689160 metric_name{label1="value3",label2="value4"} 32
```

Timestamp
Metric name
Labels
Metric value

Prometheus

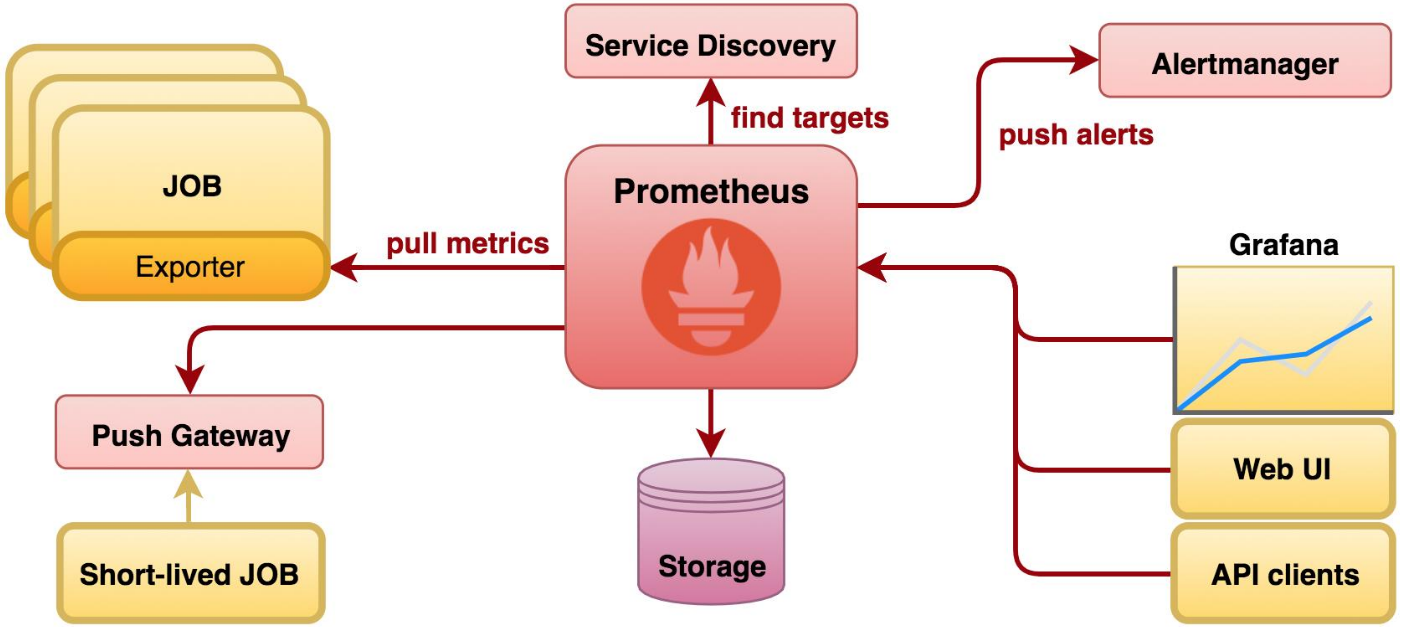
Prometheus

What is it not?

- Logging
- Tracing
- Anomaly detection
- Durable storage (infinitely)

Prometheus

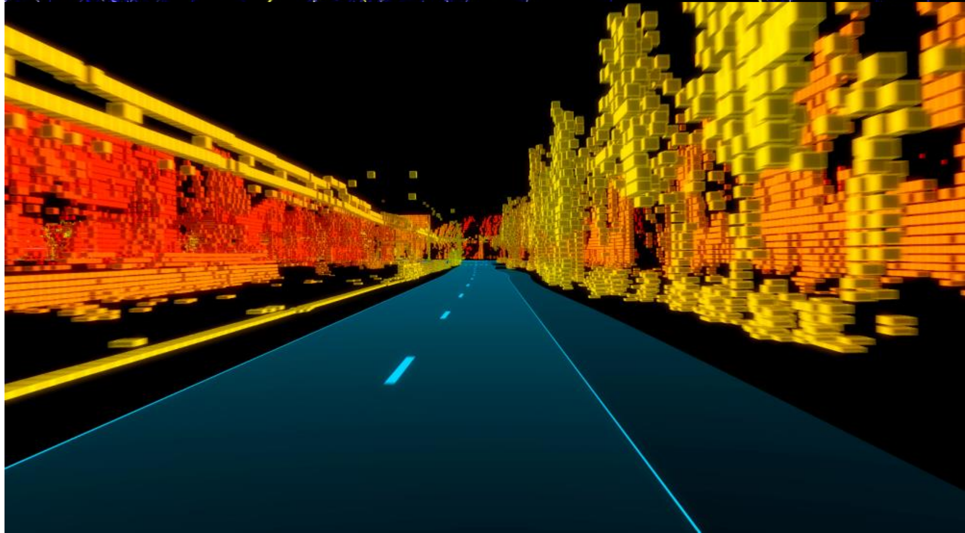
Architecture



Prometheus

Collecting data – pulling

- Static configuration (list of IPs)
- Service discovery (Consul, AWS EC2)
- Fetching via HTTP
- Instrumentation

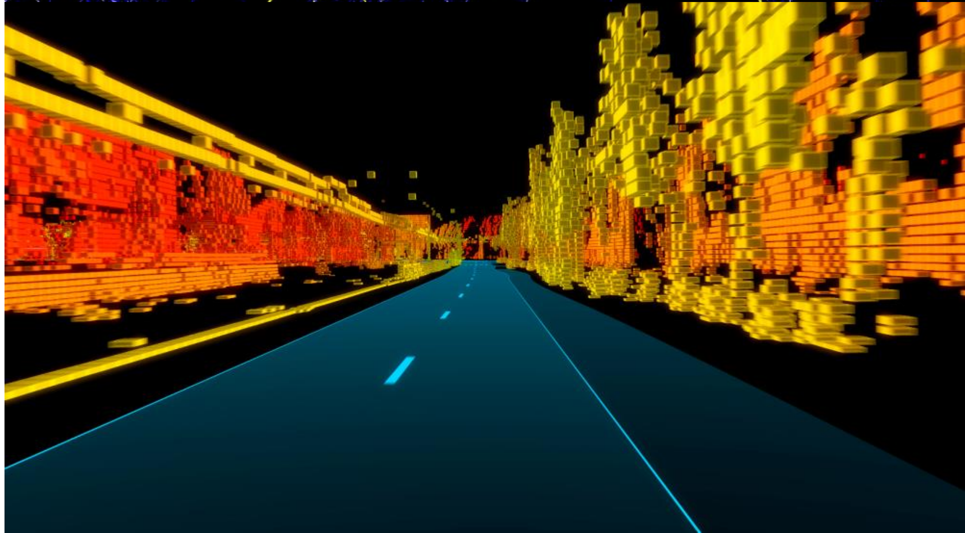


Prometheus

Collecting data - exporters

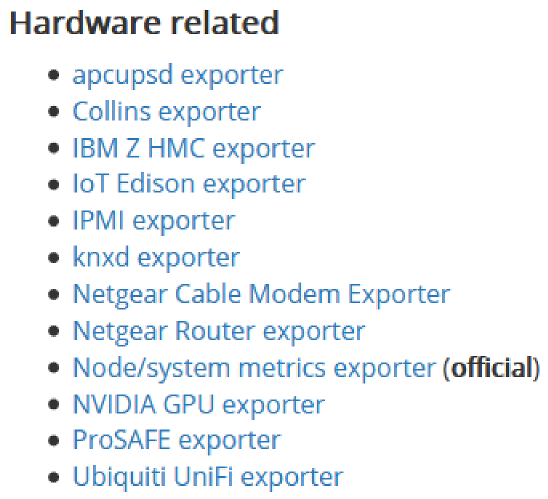
- Lots of third party exporters:
    - Hardware
    - Database
    - Messaging
    - Storage
    - HTTP
    - API
    - Other monitoring systems

https://prometheus.io/docs/instrumenting/exporters/



Prometheus

Collecting data - exporters



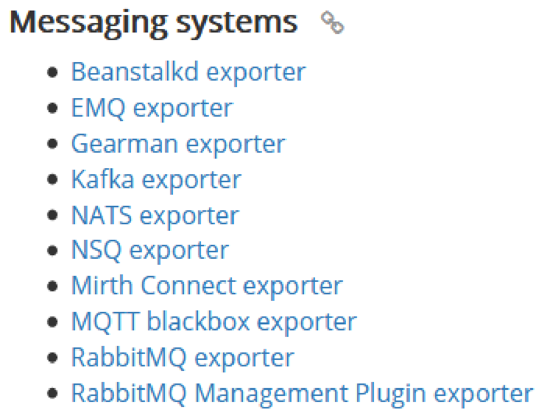
Hardware related

- apcupsd exporter
- Collins exporter
- IBM Z HMC exporter
- IoT Edison exporter
- IPMI exporter
- knxd exporter
- Netgear Cable Modem Exporter
- Netgear Router exporter
- Node/system metrics exporter (**official**)
- NVIDIA GPU exporter
- ProSAFE exporter
- Ubiquiti UniFi exporter

https://prometheus.io/docs/instrumenting/exporters/

Prometheus

Collecting data - exporters



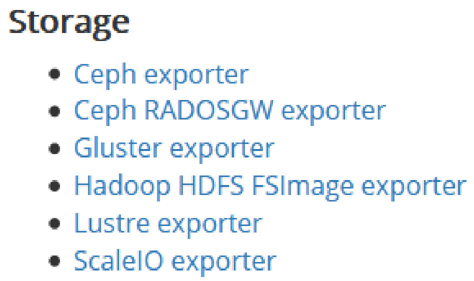
Messaging systems

- Beanstalkd exporter
- EMQ exporter
- Gearman exporter
- Kafka exporter
- NATS exporter
- NSQ exporter
- Mirth Connect exporter
- MQTT blackbox exporter
- RabbitMQ exporter
- RabbitMQ Management Plugin exporter

https://prometheus.io/docs/instrumenting/exporters/

Prometheus

Collecting data - exporters



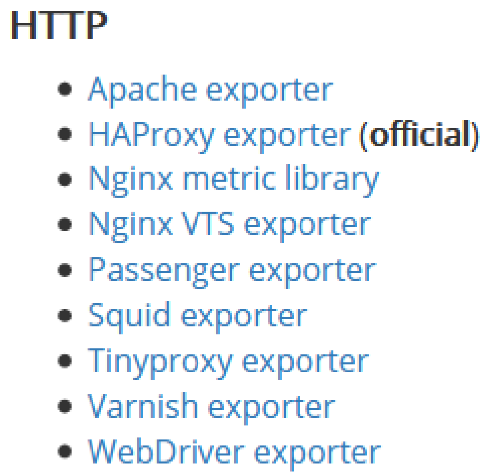
Storage

- Ceph exporter
- Ceph RADOSGW exporter
- Gluster exporter
- Hadoop HDFS FSImage exporter
- Lustre exporter
- ScaleIO exporter

https://prometheus.io/docs/instrumenting/exporters/

Prometheus

Collecting data - exporters



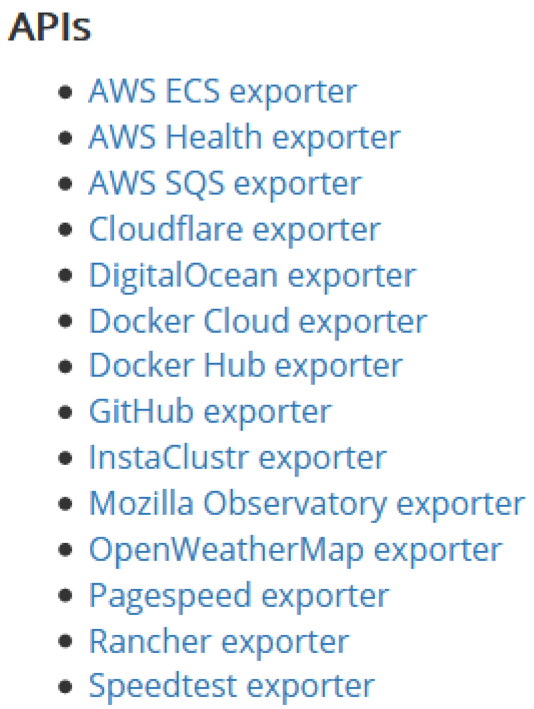
HTTP

- Apache exporter
- HAProxy exporter (**official**)
- Nginx metric library
- Nginx VTS exporter
- Passenger exporter
- Squid exporter
- Tinyproxy exporter
- Varnish exporter
- WebDriver exporter

https://prometheus.io/docs/instrumenting/exporters/

Prometheus

Collecting data - exporters



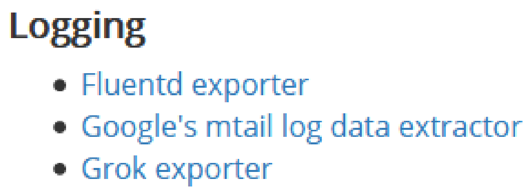
APIs

- AWS ECS exporter
- AWS Health exporter
- AWS SQS exporter
- Cloudflare exporter
- DigitalOcean exporter
- Docker Cloud exporter
- Docker Hub exporter
- GitHub exporter
- InstaClustr exporter
- Mozilla Observatory exporter
- OpenWeatherMap exporter
- Pagespeed exporter
- Rancher exporter
- Speedtest exporter

https://prometheus.io/docs/instrumenting/exporters/

Prometheus

Collecting data - exporters



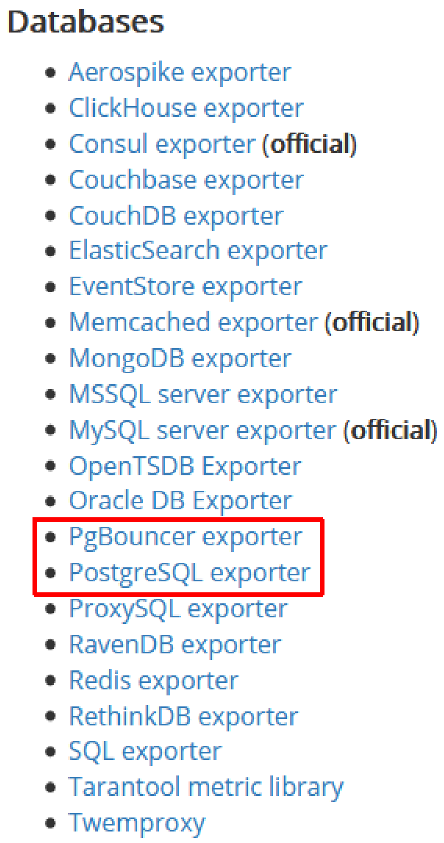
Logging

- Fluentd exporter
- Google's mtail log data extractor
- Grok exporter

https://prometheus.io/docs/instrumenting/exporters/

Prometheus

Collecting data - exporters



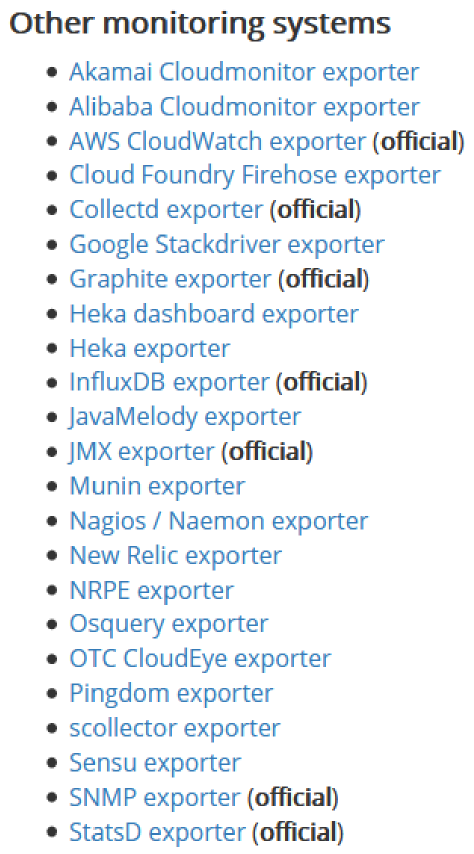
Databases

- Aerospike exporter
- ClickHouse exporter
- Consul exporter (**official**)
- Couchbase exporter
- CouchDB exporter
- ElasticSearch exporter
- EventStore exporter
- Memcached exporter (**official**)
- MongoDB exporter
- MSSQL server exporter
- MySQL server exporter (**official**)
- OpenTSDB Exporter
- Oracle DB Exporter
- PgBouncer exporter
- PostgreSQL exporter
- ProxySQL exporter
- RavenDB exporter
- Redis exporter
- RethinkDB exporter
- SQL exporter
- Tarantool metric library
- Twemproxy

https://prometheus.io/docs/instrumenting/exporters/

Prometheus

Collecting data - exporters



Other monitoring systems

- Akamai Cloudmonitor exporter
- Alibaba Cloudmonitor exporter
- AWS CloudWatch exporter (**official**)
- Cloud Foundry Firehose exporter
- Collectd exporter (**official**)
- Google Stackdriver exporter
- Graphite exporter (**official**)
- Heka dashboard exporter
- Heka exporter
- InfluxDB exporter (**official**)
- JavaMelody exporter
- JMX exporter (**official**)
- Munin exporter
- Nagios / Naemon exporter
- New Relic exporter
- NRPE exporter
- Osquery exporter
- OTC CloudEye exporter
- Pingdom exporter
- scollector exporter
- Sensu exporter
- SNMP exporter (**official**)
- StatsD exporter (**official**)

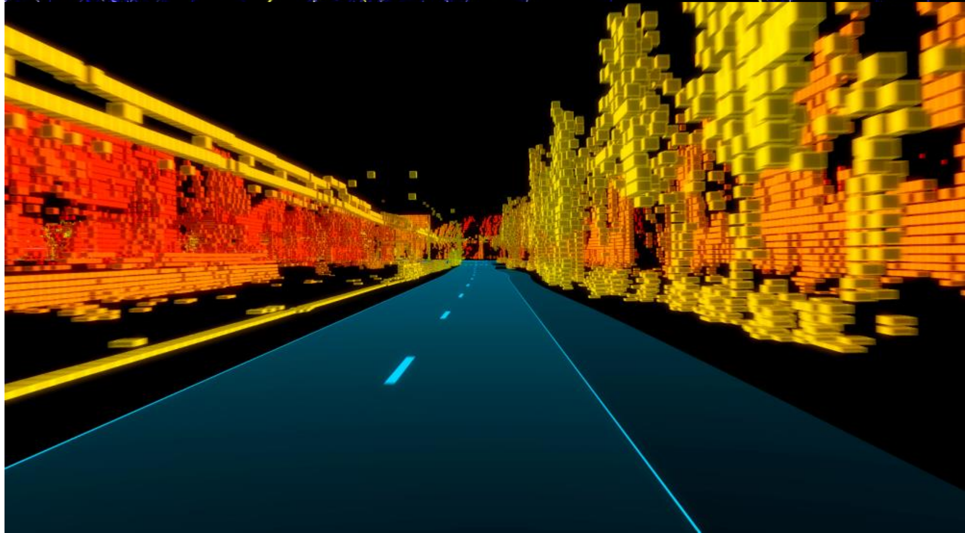
https://prometheus.io/docs/instrumenting/exporters/

Prometheus

Collecting data – postgres exporter

- Out of the box
  - Replication lag
  - Uptime
  - pg_stat_user_tables
  - pg_statio_user_tables
  - pg_database
- You may add anything you need to monitor
  - Connections
  - Schema/tables/indexes sizes
  - Vacuums
  - Query Times
  - Bloat

https://github.com/wrouesnel/postgres_exporter



Prometheus

Custom metrics in postgres exporter

Definitions in queries.yml. Example:

```
pg_user_active_per_core:
  query: "select usename, datname, count(*) as active_userfrom pg_stat_activity where state =
'active' group by datname, usename"
  metrics:
  - usename:
    usage: "LABEL"
    description: "User"
  - datname:
    usage: "LABEL"
    description: "DB name"
  - active_user:
    usage: "GAUGE"
    description: "Active users"
```

Prometheus

Custom metrics in postgres exporter

What we get from HTTP request from Prometheus:

```
#HELP pg_user_active_per_core_active_user Active users
#TYPE pg_user_active_per_core_active_user gauge
pg_user_active_per_core_active_user{datname="core103",usename="cpp_ro"} 89
pg_user_active_per_core_active_user{datname="core104",usename="cpp_wr"} 34
pg_user_active_per_core_active_user{datname="postgres",usename="postgres"} 1
```

Prometheus

Quering data

It uses its own query language – PromQL

- Designed for time series data
- Not SQL style
- Functional

Prometheus

Quering data

Example PromQL query:

```
sum by (datname, usename)
(pg_user_active_per_core_active_user{datname=
~"^core.*?"}[1m])
```

Prometheus

Prometheus

Quering data

Example PromQL query:

```
sum by (datname, usename)
(pg_user_active_per_core_active_user{datname=
~"^core.*?"}[1m])
```

Aggregating function

Metric name

Filters

Time window for aggregation

Prometheus

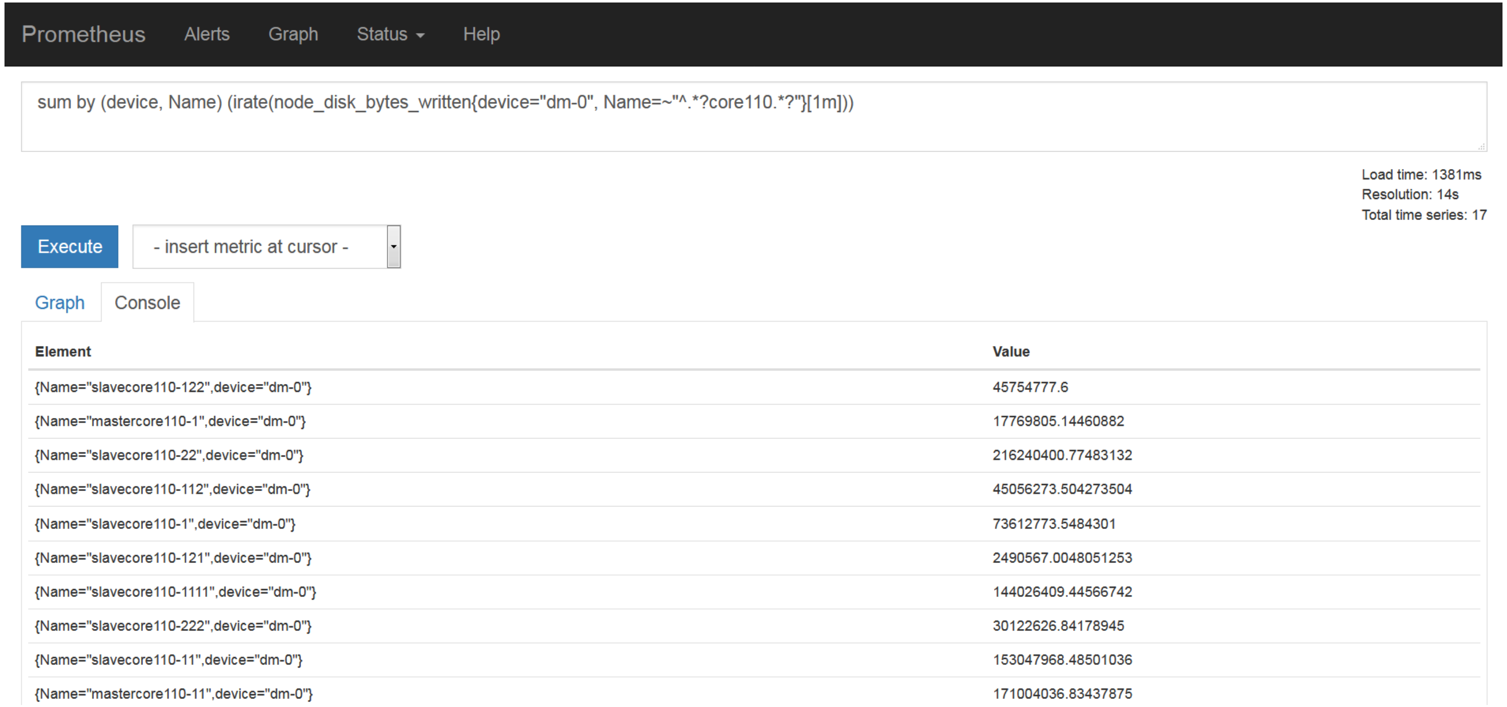
Quering data

- Data types:
  - Instant vector
  - Range vector
  - Scalar
- Operators
  - Arithmetic
  - Comparision
  - Logical
  - Aggregation
- Functions

Prometheus

Quering data

Built in UI query browser



```
sum by (device, Name) (irate(node_disk_bytes_written{device="dm-0", Name=~"^.*?core110.*?"}[1m]))
```

Load time: 1381ms
Resolution: 14s
Total time series: 17

Element	Value
{Name="slavecore110-122",device="dm-0"}	45754777.6
{Name="mastercore110-1",device="dm-0"}	17769805.14460882
{Name="slavecore110-22",device="dm-0"}	216240400.77483132
{Name="slavecore110-112",device="dm-0"}	45056273.504273504
{Name="slavecore110-1",device="dm-0"}	73612773.5484301
{Name="slavecore110-121",device="dm-0"}	2490567.0048051253
{Name="slavecore110-1111",device="dm-0"}	144026409.44566742
{Name="slavecore110-222",device="dm-0"}	30122626.84178945
{Name="slavecore110-11",device="dm-0"}	153047968.48501036
{Name="mastercore110-11",device="dm-0"}	171004036.83437875

Prometheus

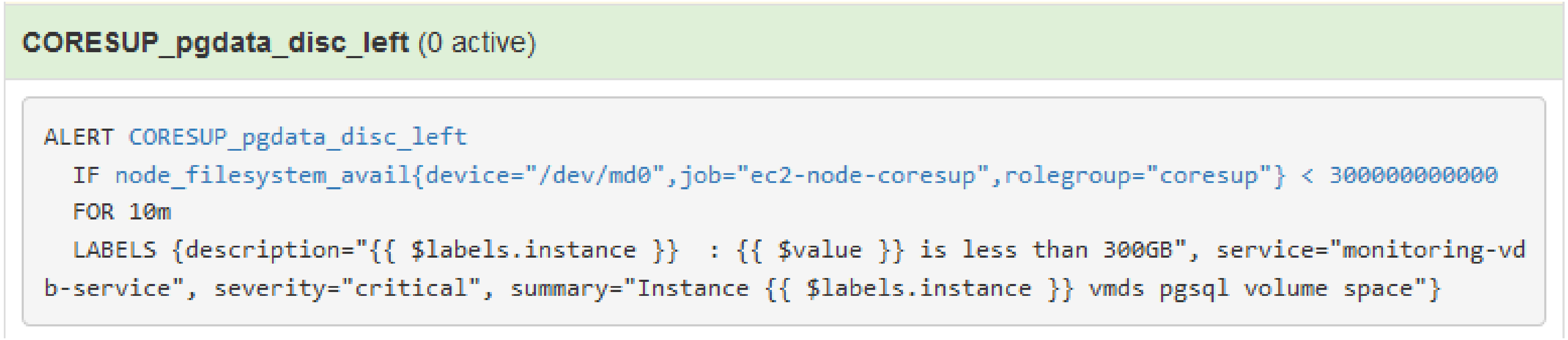
Prometheus

Alerting

- Alerting rules
    - Use PromQL
    - Templating
- Alerting manager
    - Inhibition
    - Aggregation
    - Routing

Prometheus

Alerting


CORESUP_pgdata_disc_left (0 active)

```
ALERT CORESUP_pgdata_disc_left
  IF node_filesystem_avail{device="/dev/md0",job="ec2-node-coresup",rolegroup="coresup"} < 300000000000
  FOR 10m
  LABELS {description="{{ $labels.instance }}  : {{ $value }} is less than 300GB", service="monitoring-vd
b-service", severity="critical", summary="Instance {{ $labels.instance }} vmds pgsql volume space"}
```

Prometheus

Prometheus

High Availability

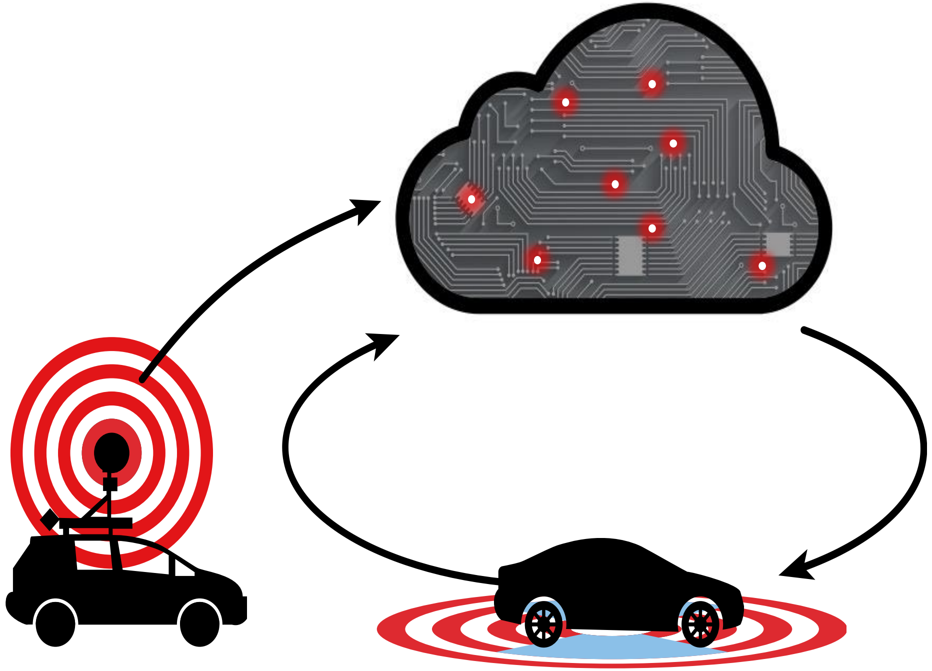
- Autonomous single server nodes
- Local storage, no clustering
- Minimal network dependency
- Preferred metrics collection by pulling
- To avoid single point of failure run two identical but independent Prometheus servers that collect the same metrics
- To backup your metrics use decoupled remote storage (starting from 1.6)

Prometheus

Prometheus

Installation and configuration

- Precompiled binaries
- Makefile
- Docker
- Ansible
- Chef
- Puppet
- Salt



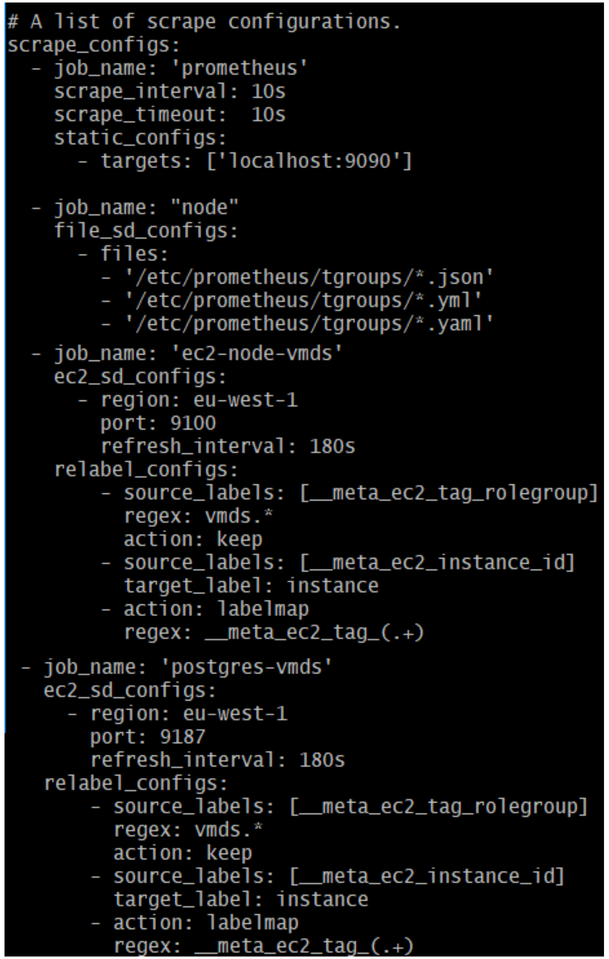
TOMTOM

Prometheus

Installation and configuration

- YAML format
- Reload at runtime

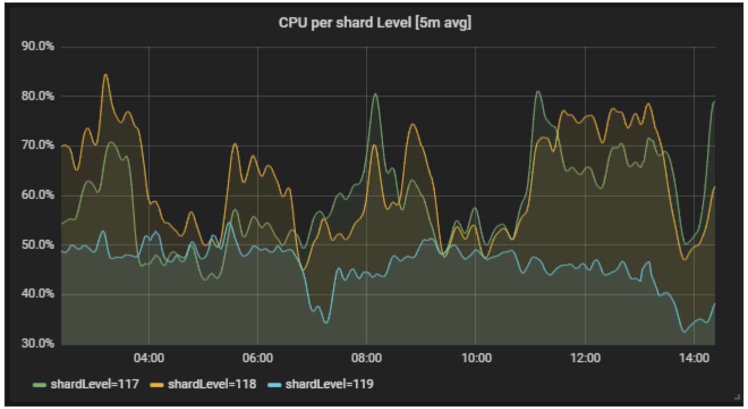
- Sections:
    - global
    - rule_files
    - scrape_configs
    - alerting
    - remote_write
    - remote_read

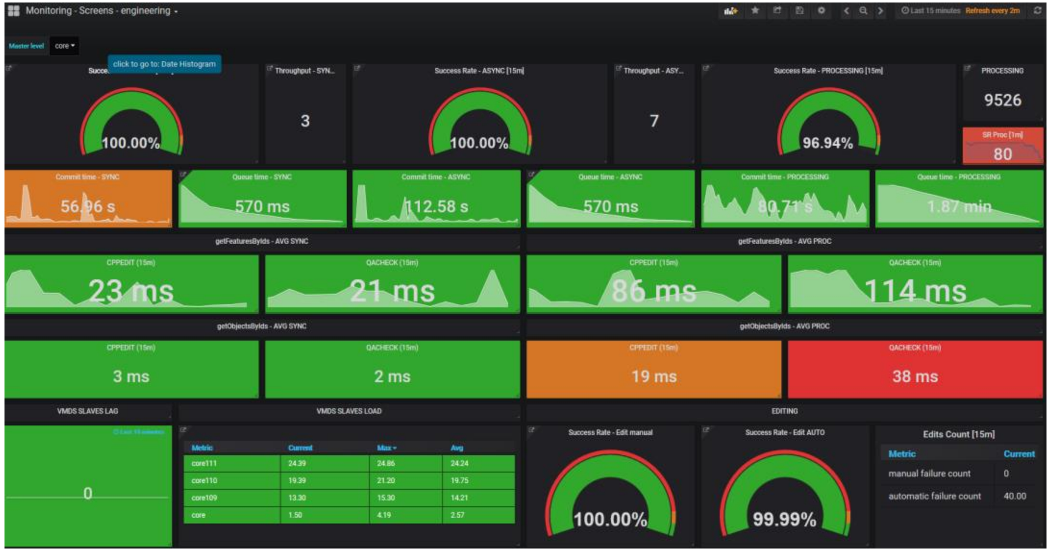


```
# A list of scrape configurations.
scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 10s
    scrape_timeout:  10s
    static_configs:
      - targets: ['localhost:9090']

  - job_name: "node"
    file_sd_configs:
      - files:
        - '/etc/prometheus/tgroups/*.json'
        - '/etc/prometheus/tgroups/*.yml'
        - '/etc/prometheus/tgroups/*.yaml'
  - job_name: 'ec2-node-vmds'
    ec2_sd_configs:
      - region: eu-west-1
        port: 9100
        refresh_interval: 180s
    relabel_configs:
        - source_labels: [__meta_ec2_tag_rolegroup]
          regex: vmds.*
          action: keep
        - source_labels: [__meta_ec2_instance_id]
          target_label: instance
        - action: labelmap
          regex: __meta_ec2_tag_(.+)
 - job_name: 'postgres-vmds'
   ec2_sd_configs:
     - region: eu-west-1
       port: 9187
       refresh_interval: 180s
   relabel_configs:
       - source_labels: [__meta_ec2_tag_rolegroup]
         regex: vmds.*
         action: keep
       - source_labels: [__meta_ec2_instance_id]
         target_label: instance
       - action: labelmap
         regex: __meta_ec2_tag_(.+)
```

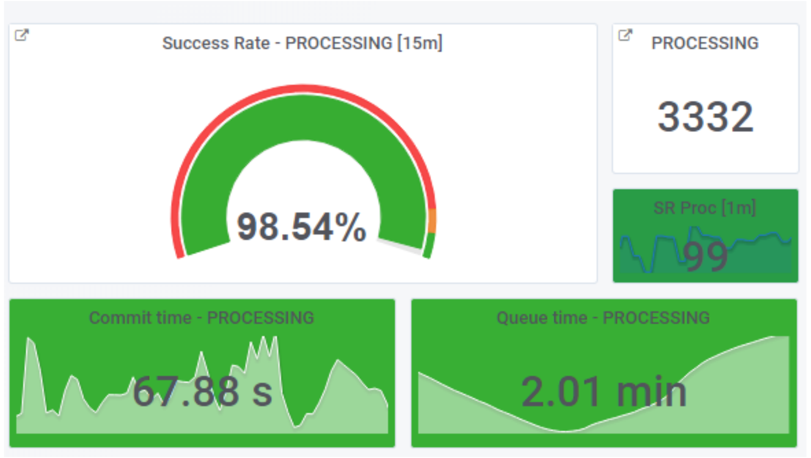
Grafana





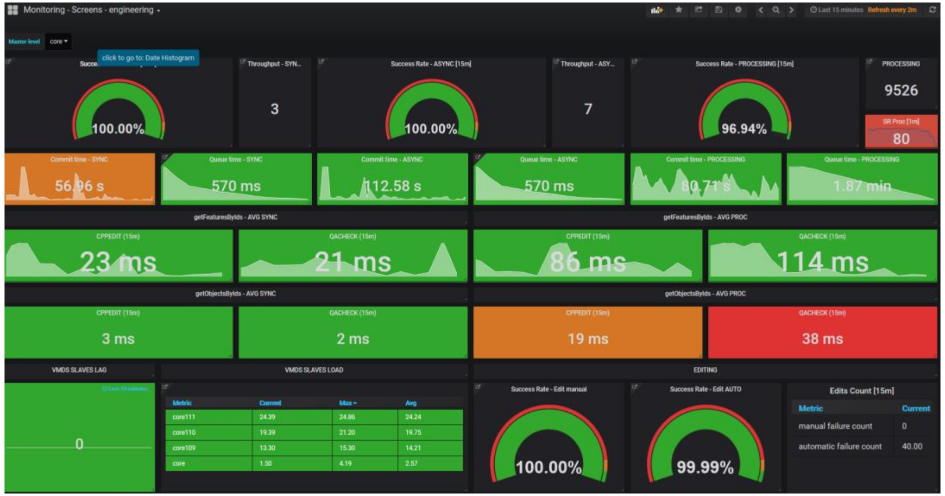
Metric	Current	Max ▾	Avg
core111	24.39	24.86	24.24
core110	19.39	21.20	19.75
core109	13.30	15.30	14.21
core	1.50	4.19	2.57

Metric	Current
manual failure count	0
automatic failure count	40.00



Grafana

- Rich visualization toolkit https://grafana.com/
- Open-source (Apache 2.0 License) with enterprise support
- Flexible dashboards helps understand data
- Time series data-sources:
  - Elasticsearch,
  - Promethues,
  - AWS CloudWatch,
  - Graphite



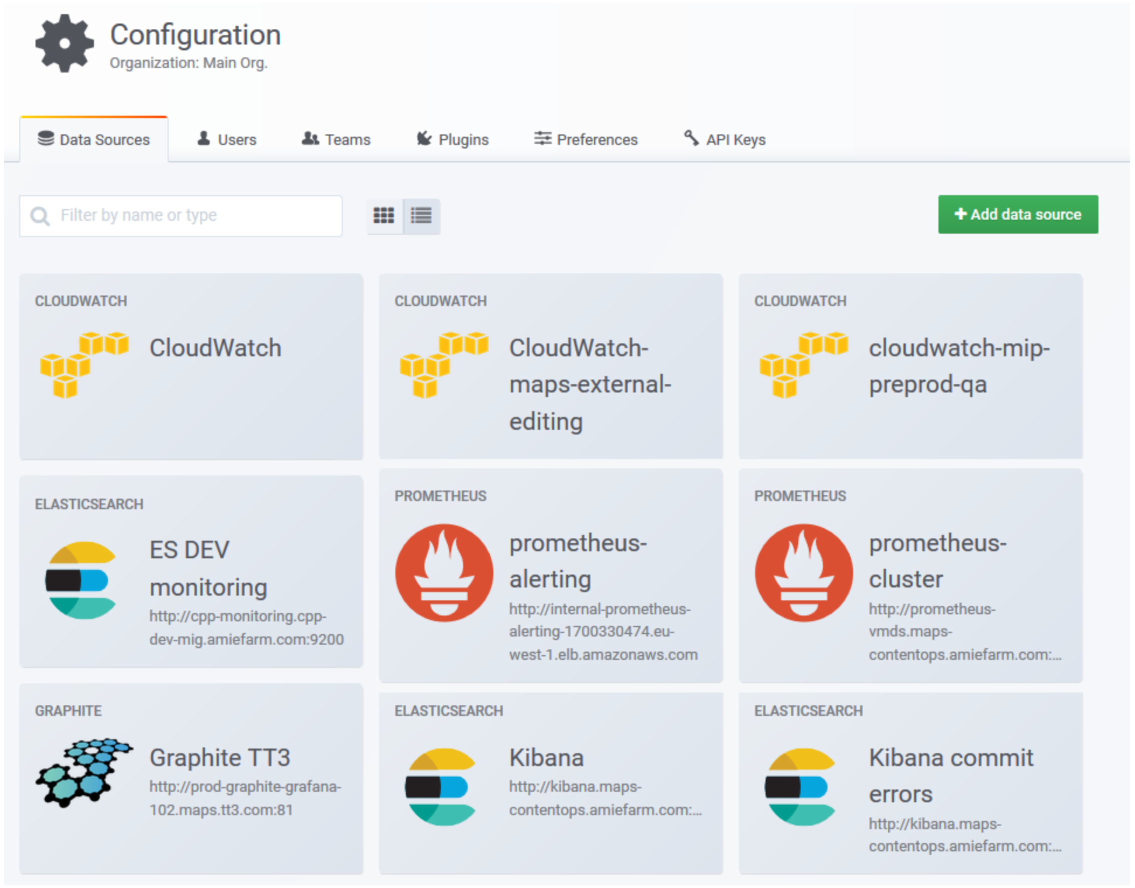
Metric	Current	Max	Avg
core111	24.39	24.86	24.24
core110	19.39	21.20	19.75
core109	13.30	15.30	14.21
core	1.50	4.19	2.57

Metric	Current
manual failure count	0
automatic failure count	40.00

Grafana

Data-sources

- Installed as plugins
- Query editors to get metrics



Configuration
Organization: Main Org.

CLOUDWATCH
CloudWatch

CLOUDWATCH
CloudWatch-maps-external-editing

CLOUDWATCH
cloudwatch-mip-preprod-qa

ELASTICSEARCH
ES DEV monitoring
http://cpp-monitoring.cpp-dev-mig.amiefarm.com:9200

PROMETHEUS
prometheus-alerting
http://internal-prometheus-alerting-1700330474.eu-west-1.elb.amazonaws.com

PROMETHEUS
prometheus-cluster
http://prometheus-vmds.maps-contentops.amiefarm.com:...

GRAPHITE
Graphite TT3
http://prod-graphite-grafana-102.maps.tt3.com:81

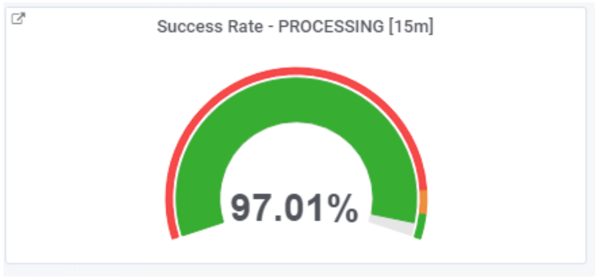
ELASTICSEARCH
Kibana
http://kibana.maps-contentops.amiefarm.com:...

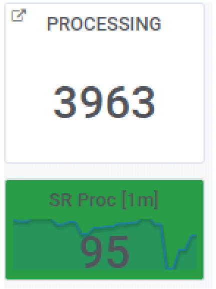
ELASTICSEARCH
Kibana commit errors
http://kibana.maps-contentops.amiefarm.com:...

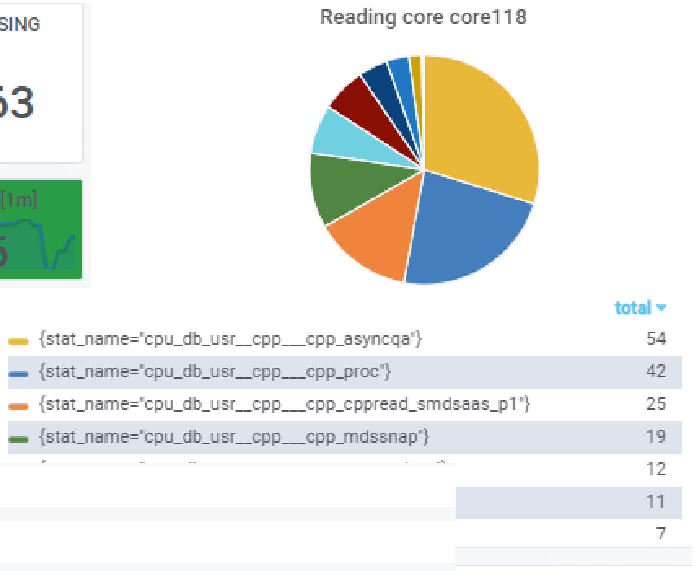
Grafana

Dashboards

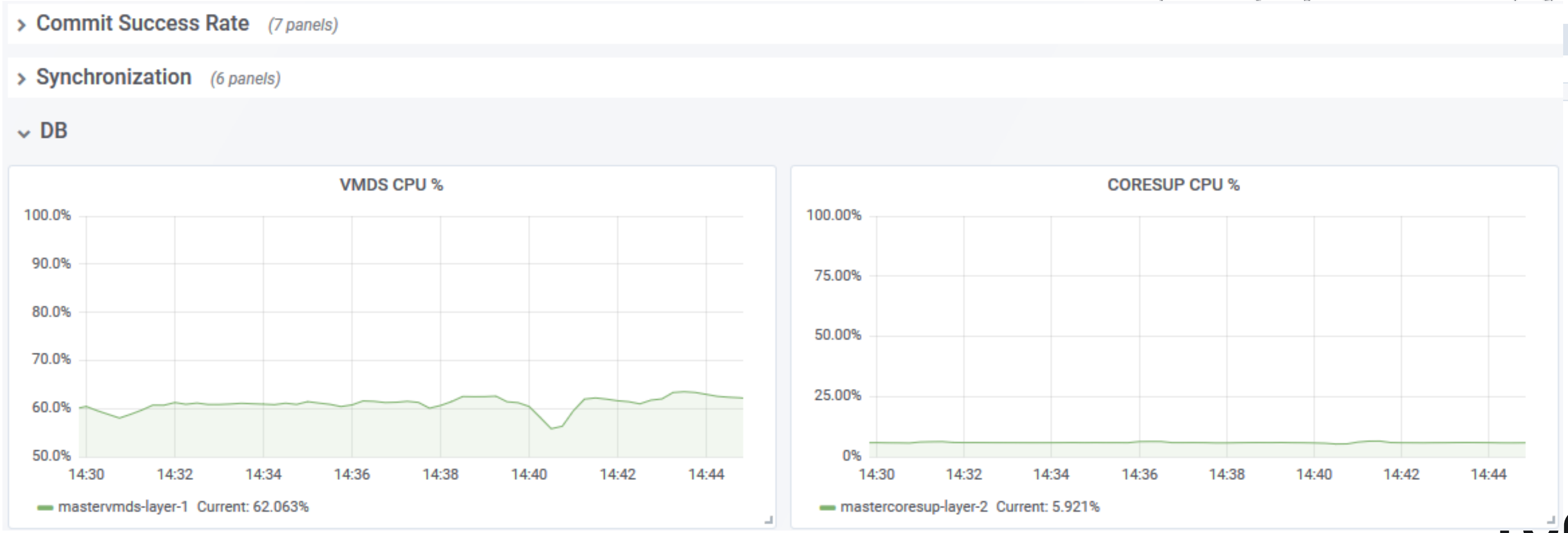
- Rows
- Panels
- Query editors







	total
🟨 {stat_name="cpu_db_usr__cpp___cpp_asyncqa"}	54
🟦 {stat_name="cpu_db_usr__cpp___cpp_proc"}	42
🟧 {stat_name="cpu_db_usr__cpp___cpp_cppread_smdsaas_p1"}	25
🟩 {stat_name="cpu_db_usr__cpp___cpp_mdssnap"}	19
	12
	11
	7



mastervmds-layer-1 Current: 62.063%

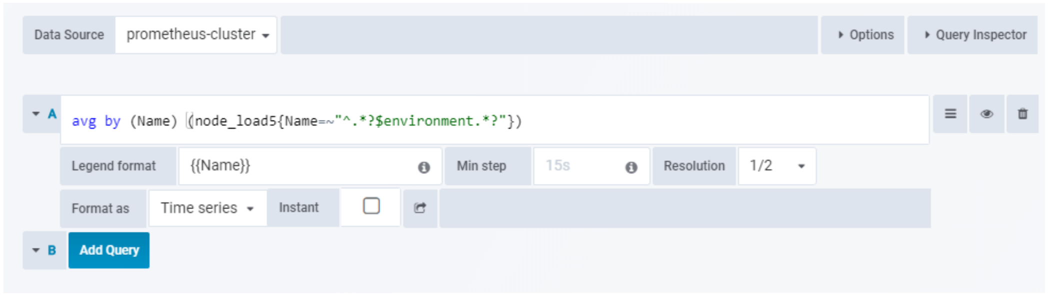
mastercoresup-layer-2 Current: 5.921%

Grafana

Query editors

- Difference in data-source settings
- Native queries

- AWS CloudWatch exporter
- Elasticsearch exporter



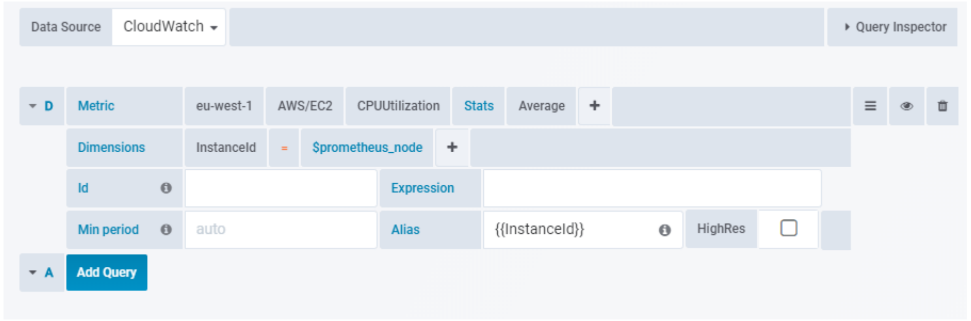
Data Source: prometheus-cluster | Options | Query Inspector

A: `avg by (Name) (node_load5{Name=~"^.*?$environment.*?"})`

Legend format: {{Name}} | Min step: 15s | Resolution: 1/2

Format as: Time series | Instant: ☐

B: Add Query



Data Source: CloudWatch | Query Inspector

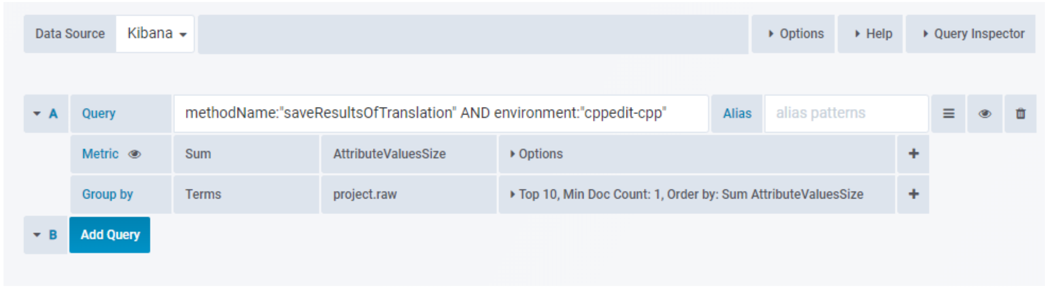
D: Metric: eu-west-1 | AWS/EC2 | CPUUtilization | Stats: Average | +

Dimensions: InstanceId = $prometheus_node | +

Id: | Expression:

Min period: auto | Alias: {{InstanceId}} | HighRes: ☐

A: Add Query



Data Source: Kibana | Options | Help | Query Inspector

A: Query: methodName:"saveResultsOfTranslation" AND environment:"cppedit-cpp" | Alias: alias patterns

Metric: Sum | AttributeValuesSize | Options | +

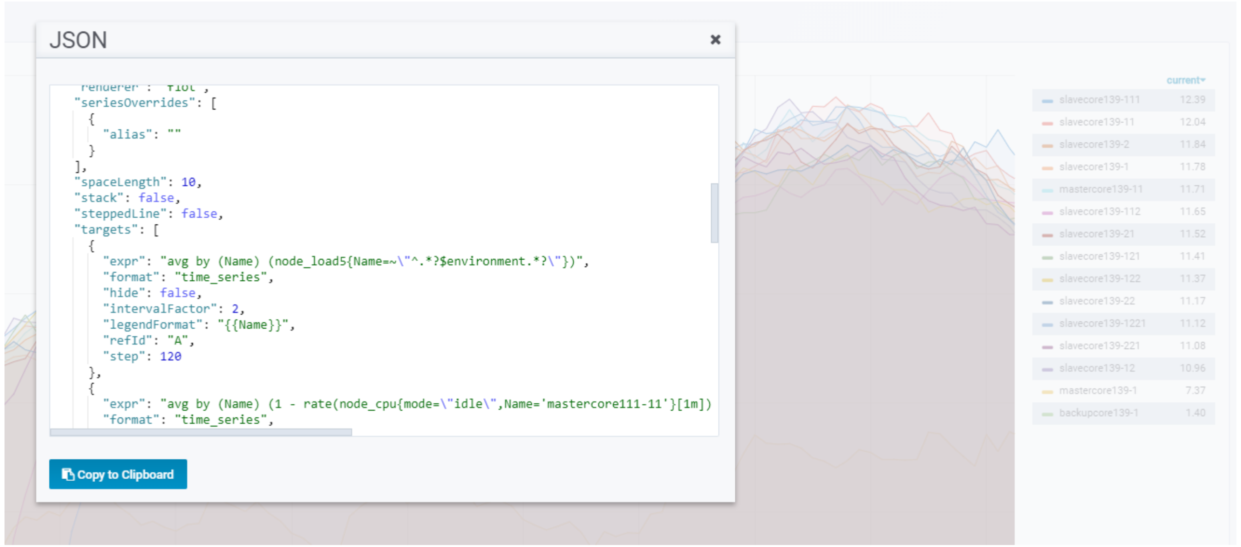
Group by: Terms | project.raw | Top 10, Min Doc Count: 1, Order by: Sum AttributeValuesSize | +

B: Add Query

Grafana

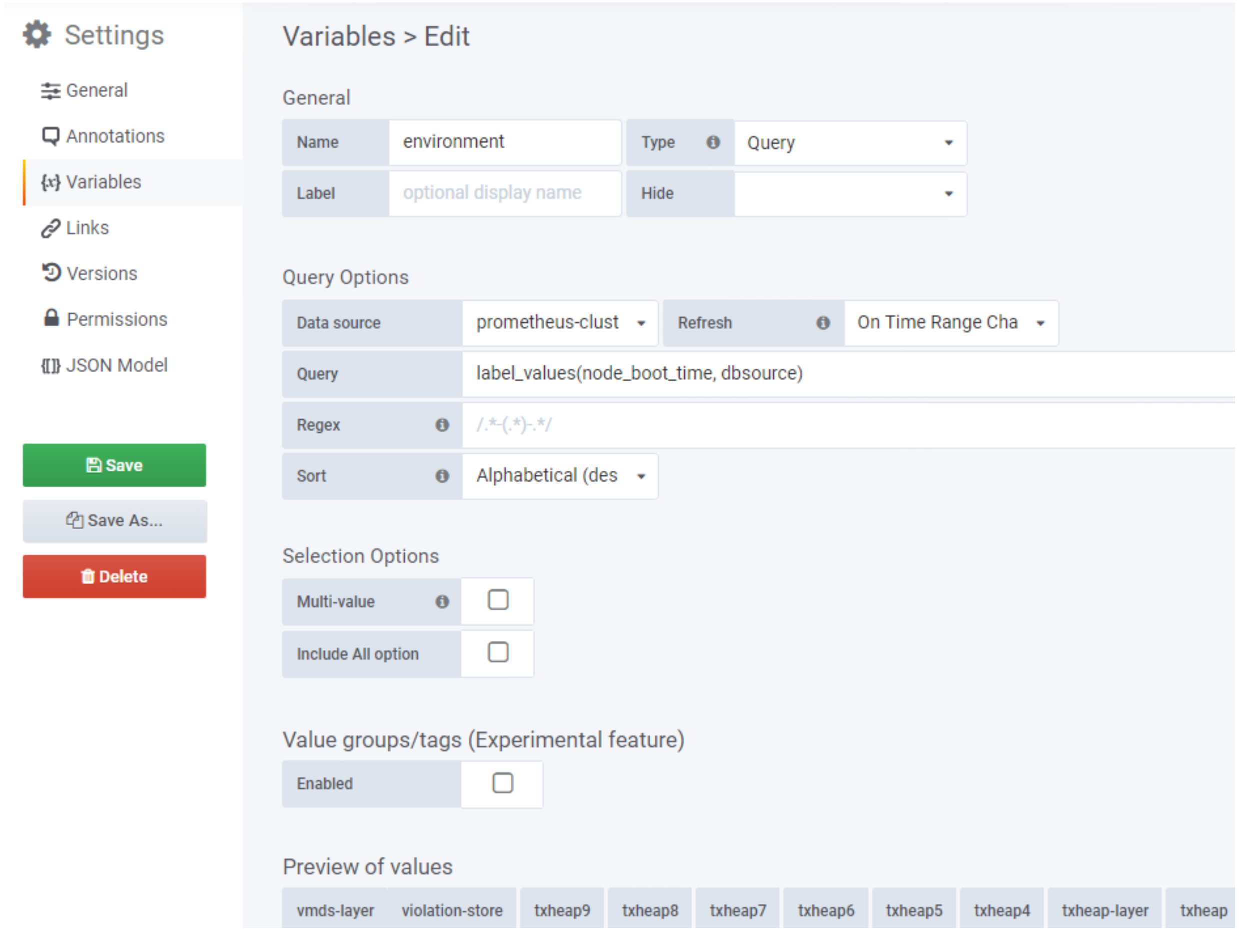
Advanced dashboards

- Templating with dynamic variables
- Annotations for marking events
- ACL per user per dashboard



```
  "renderer": "flot",
  "seriesOverrides": [
    {
      "alias": ""
    }
  ],
  "spaceLength": 10,
  "stack": false,
  "steppedLine": false,
  "targets": [
    {
      "expr": "avg by (Name) (node_load5{Name=~\"^.*?$environment.*?\"})",
      "format": "time_series",
      "hide": false,
      "intervalFactor": 2,
      "legendFormat": "{{Name}}",
      "refId": "A",
      "step": 120
    },
    {
      "expr": "avg by (Name) (1 - rate(node_cpu{mode=\"idle\",Name='mastercore111-11'}[1m])
      "format": "time_series",
```

	current▾
slavecore139-111	12.39
slavecore139-11	12.04
slavecore139-2	11.84
slavecore139-1	11.78
mastercore139-11	11.71
slavecore139-112	11.65
slavecore139-21	11.52
slavecore139-121	11.41
slavecore139-122	11.37
slavecore139-22	11.17
slavecore139-1221	11.12
slavecore139-221	11.08
slavecore139-12	10.96
mastercore139-1	7.37
backupcore139-1	1.40



Variables > Edit

General

Name	environment	Type	Query
Label	optional display name	Hide	

Query Options

Data source	prometheus-clust	Refresh	On Time Range Cha
Query	label_values(node_boot_time, dbsource)		
Regex	/.*-(.*)-.*/		
Sort	Alphabetical (des		

Selection Options

Multi-value	☐
Include All option	☐

Value groups/tags (Experimental feature)

Enabled	☐

Preview of values

vmds-layer violation-store txheap9 txheap8 txheap7 txheap6 txheap5 txheap4 txheap-layer txheap

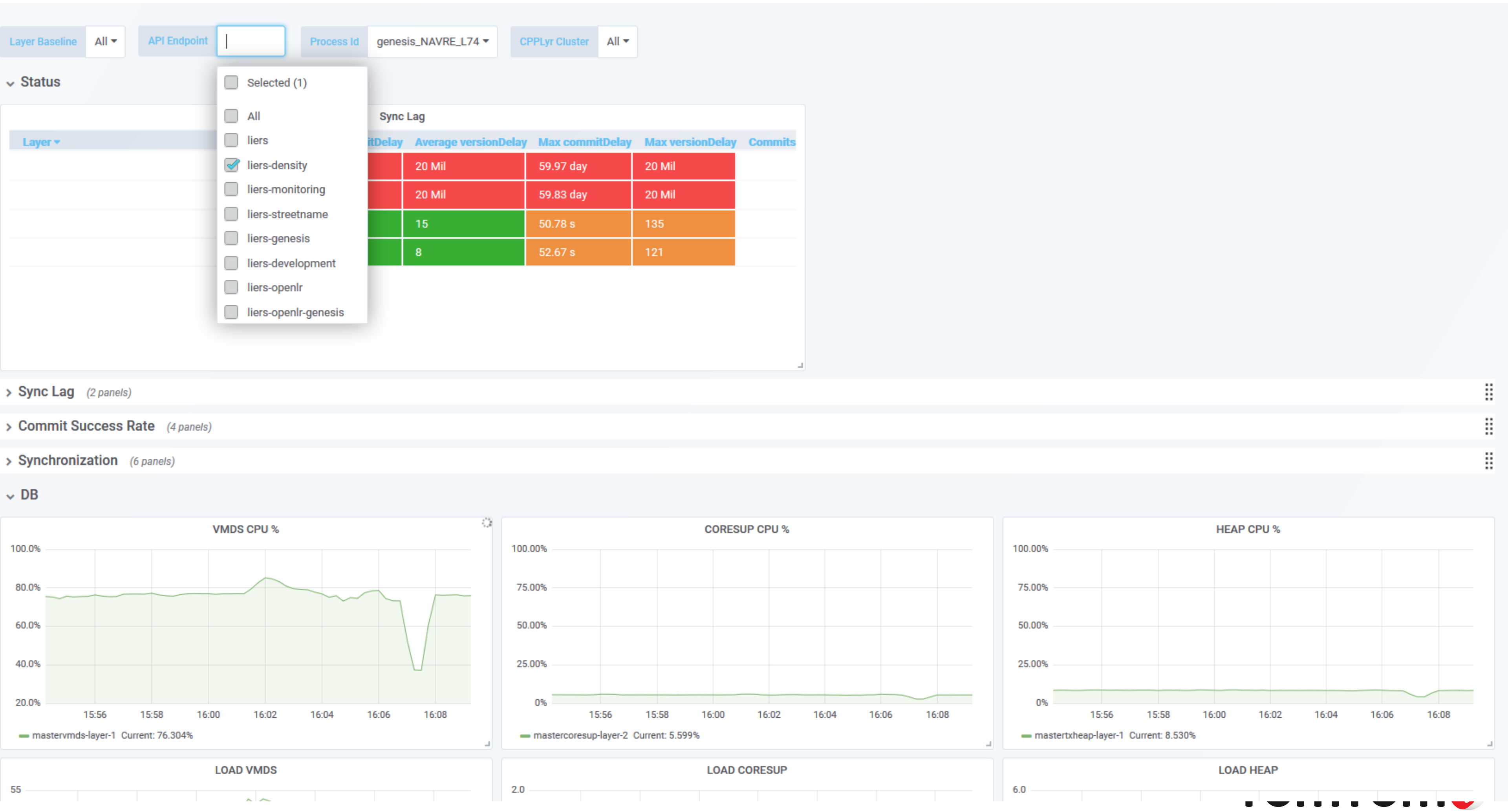
Grafana

Advanced dashboards

- HTTP API – manage Grafana and dashboards
- JSON model: properties, variables, panels and queries
- Versioning and changes tracking
- Benefit from automation - limit manual work

```
POST /api/dashboards/db HTTP/1.1
Accept: application/json
Content-Type: application/json
Authorization: Bearer eyJrIjoiT0tTcG1pUl

{
  "dashboard": {
    "id": null,
    "uid": null,
    "title": "Production Overview",
    "tags": [ "templated" ],
    "timezone": "browser",
    "schemaVersion": 16,
    "version": 0
  },
  "folderId": 0,
  "overwrite": false
}
```



Layer Baseline All ▾ | API Endpoint | Process Id genesis_NAVRE_L74 ▾ | CPPLyr Cluster All ▾

- Selected (1)
- All
- liers
- ✓ liers-density
- liers-monitoring
- liers-streetname
- liers-genesis
- liers-development
- liers-openlr
- liers-openlr-genesis

Status

Sync Lag

Layer ▾	itDelay	Average versionDelay	Max commitDelay	Max versionDelay	Commits
	🟥	20 Mil	59.97 day	20 Mil	
	🟥	20 Mil	59.83 day	20 Mil	
	🟩	15	50.78 s	135	
	🟩	8	52.67 s	121	

› Sync Lag (2 panels)

› Commit Success Rate (4 panels)

› Synchronization (6 panels)

DB

VMDS CPU %

mastervmds-layer-1 Current: 76.304%

CORESUP CPU %

mastercoresup-layer-2 Current: 5.599%

HEAP CPU %

mastertxheap-layer-1 Current: 8.530%

LOAD VMDS

LOAD CORESUP

LOAD HEAP

Summary

- Prometheus can collect and store time-series data with fast lookups.
- External datastores for long-term storage and HA (InfluxDB or PostgreSQL).
- Grafana enables visualization of data from different sources.
- Both tools allow automation in building monitoring infrastructure.
- Dashboards are important, but an alerting system should notify about crossing treshholds.

What’s next?

- Start playing with it!

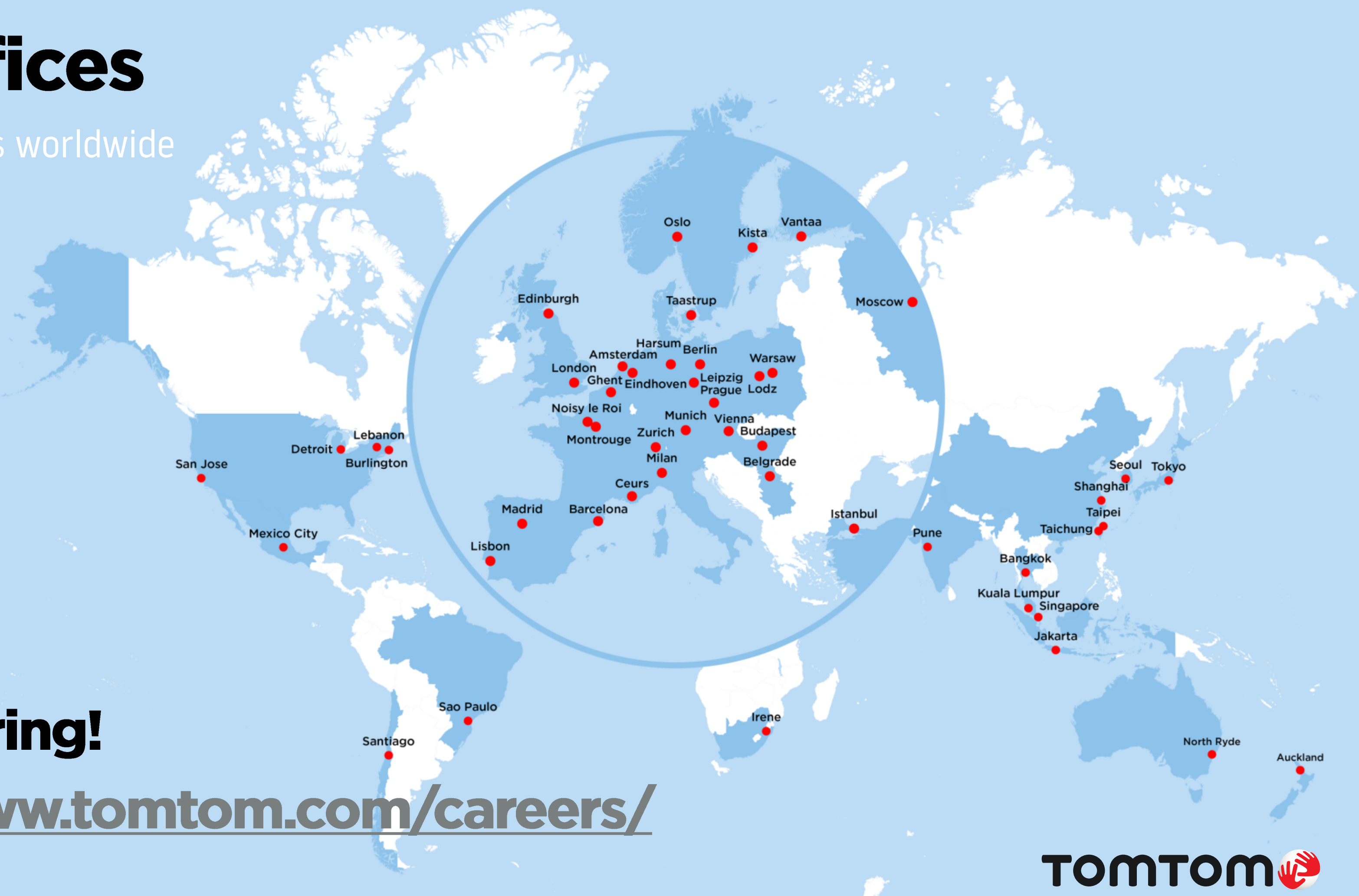
- wget https://github.com/prometheus/prometheus/releases/download/v2.8.0/prometheus-2.8.0.linux-amd64.tar.gz

  tar –zxvf prometheus-...

- wget https://dl.grafana.com/oss/release/grafana-6.0.2.linux-amd64.tar.gz
  tar -zxvf grafana-..

Our offices

5000 employees worldwide



We are hiring!

https://www.tomtom.com/careers/

TOMTOM

THANK YOU

TOMTOM