Introducing Reliability Toolkit: easy-to-use monitoring and alerting

Robin van Zijll & Janna Brummel

PromCon 2018 • 10 August 2018

ING

Hi! Robin Janna

What do we work on with whom, how and why?

Who? Team of 7 SREs with the goal to reduce mean time to repair and increase mean time between failures for IT services within a bank

Why? We do not reach availability levels expected by customers or regulators

How? We enable ~300 BizDevOps squads through engineering, delivery of tooling, consulting and education

What? We deliver a monitoring solution: the **Reliability Toolkit**, a ChatOps platform, we facilitate postmortems and we educate engineers about SRE- related topics

Why did we develop the Reliability Toolkit?

Alerting not directly to teams

- Time before engineer starts resolving (major) incident is 69 minutes on average

Lack of white-box

- Currently only real monitoring is black-box, does not fit with ‘you build it, you run it’

High level of technology diversity

- Prometheus exporters make monitoring highly adoptable

A bank can be a documentation factory

- It is a pain for teams to create something new

Simplicity

- One toolkit to cover reliability building blocks, easy to get started, easy to use

What’s in the Reliability Toolkit?

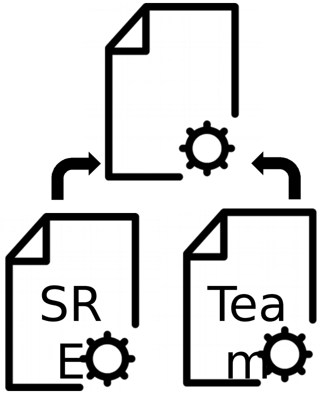
Prometheus

Alert Manager

Grafana

Model Builder*

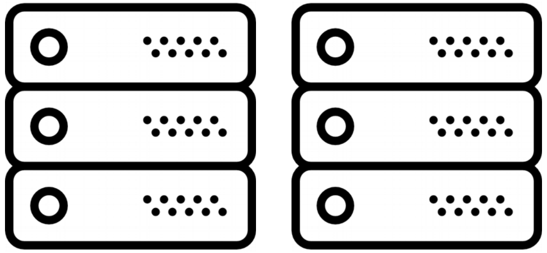
How do we provision the Reliability Toolkit?



Together with a team we create a joint config

We maintain and update the bin files

We deliver the Reliability Toolkit on 5 machines over 3 environments, we remain responsible



We deliver client libraries so metrics can be scraped from servers

Increasing and improving usage of Reliability Toolkit

Include client libraries in engineering frameworks

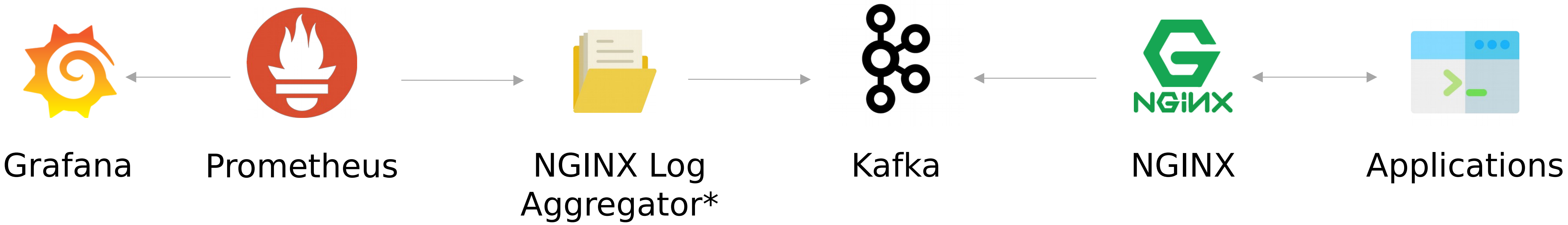
Ensure a good feedback loop with your customers

Educate others during onboarding and workshops

Create dashboards accessible to all engineers

Create awesome dashboards accessible to all engineers

NLA



Grafana

Prometheus

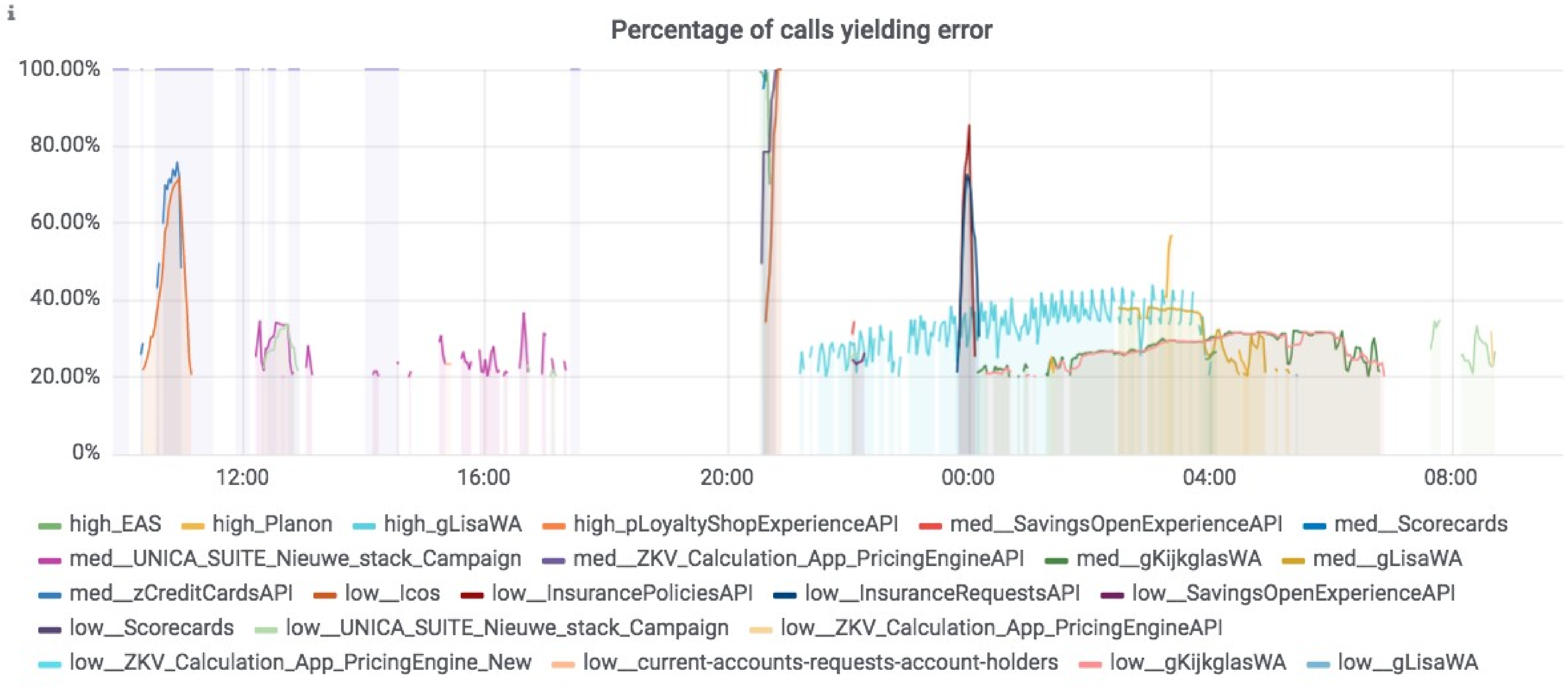
NGINX Log Aggregator*

Kafka

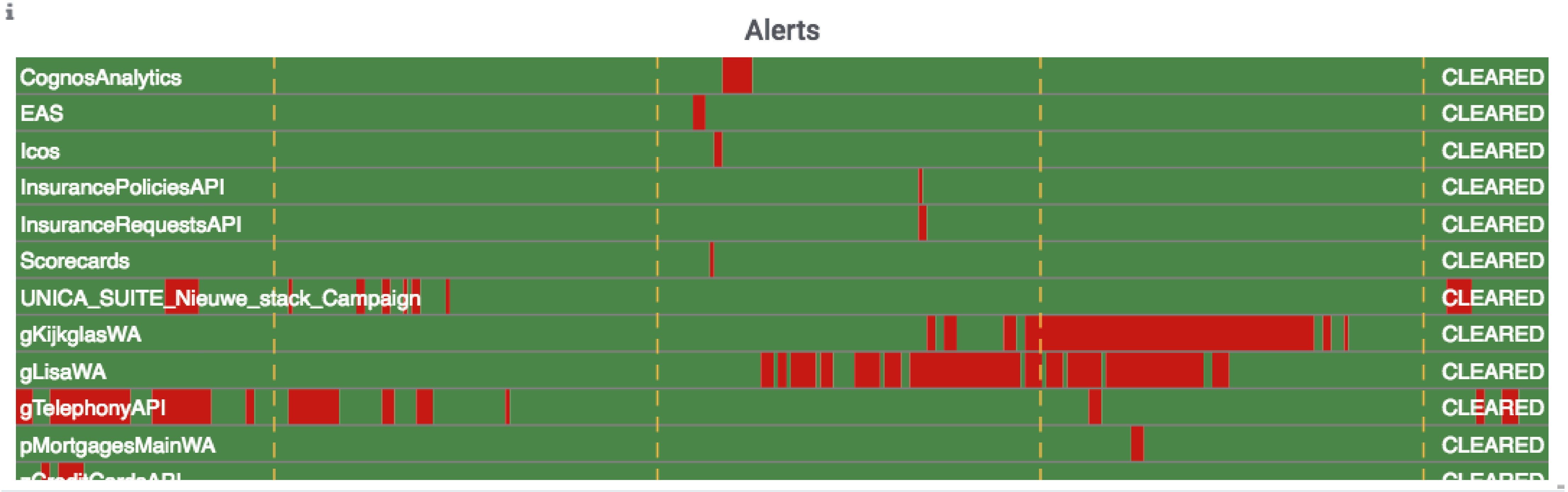
NGINX

Applications

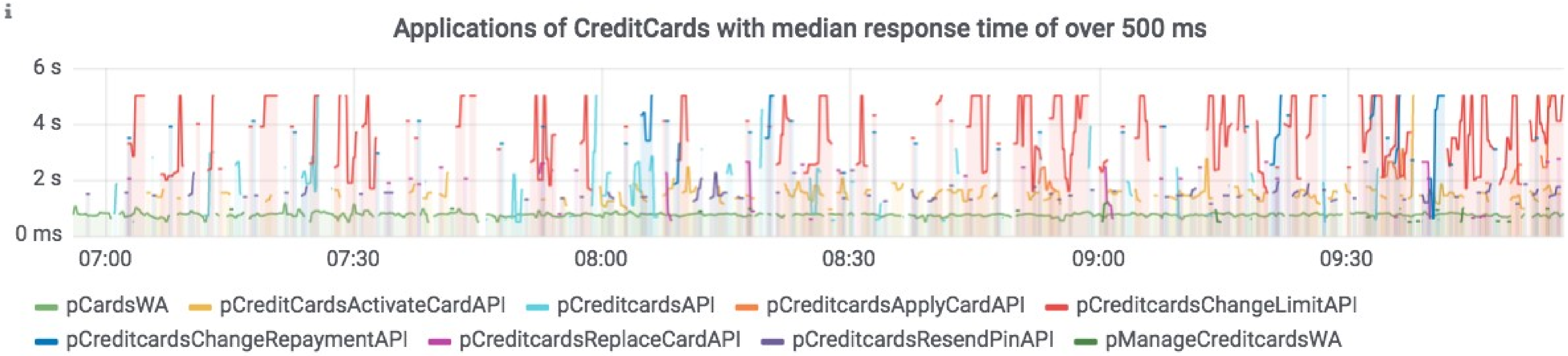
Error Overview (1)



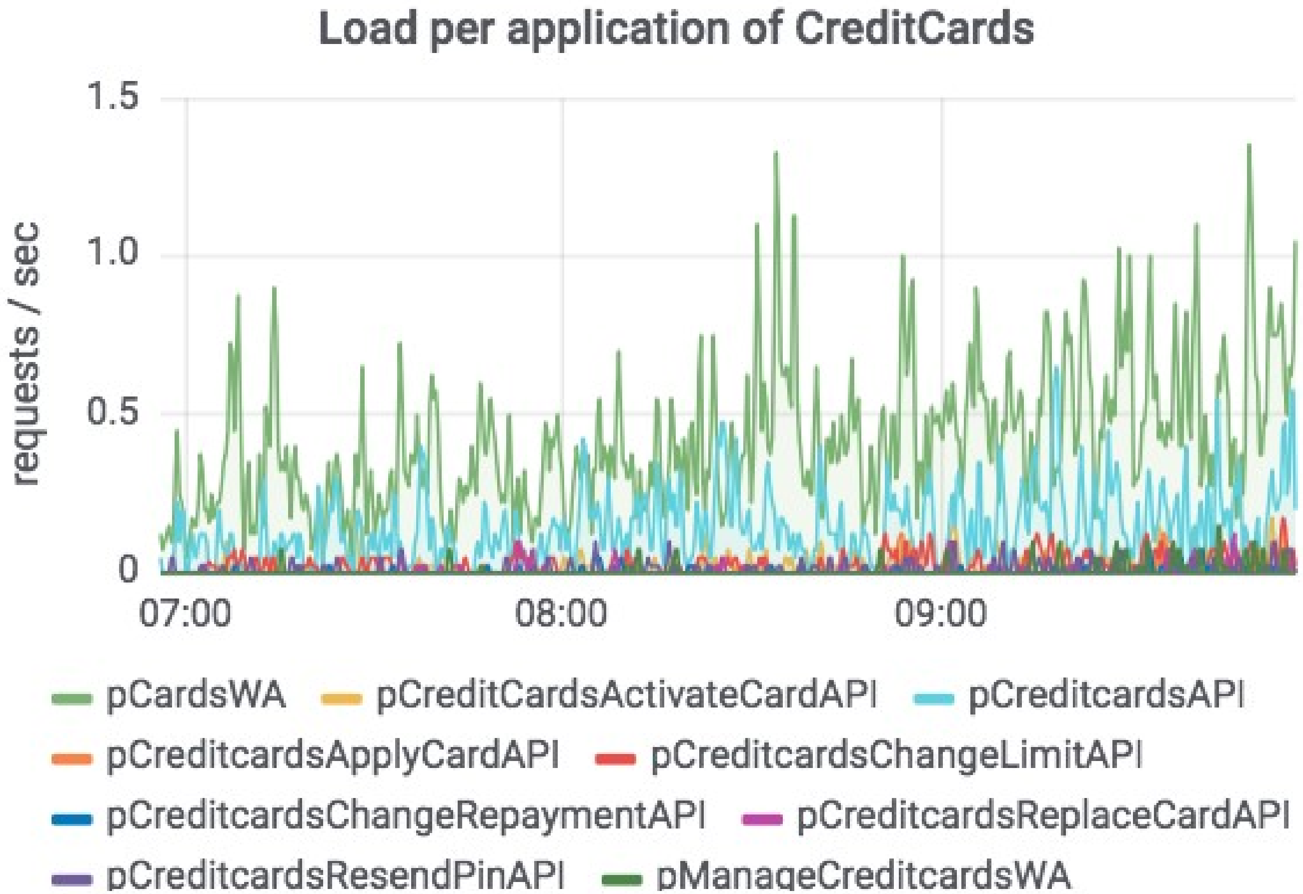
Error Overview (2)



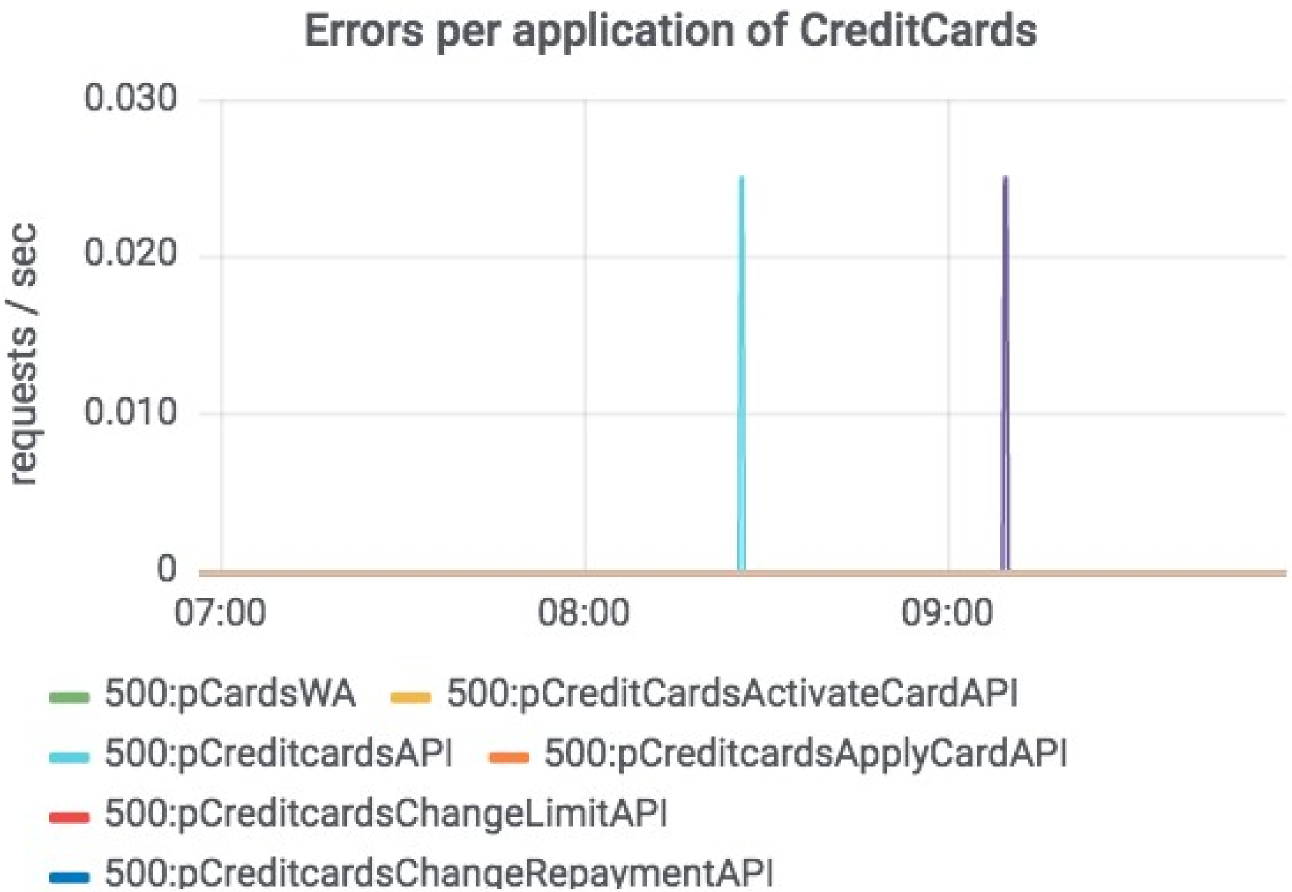
Team Overview (1)



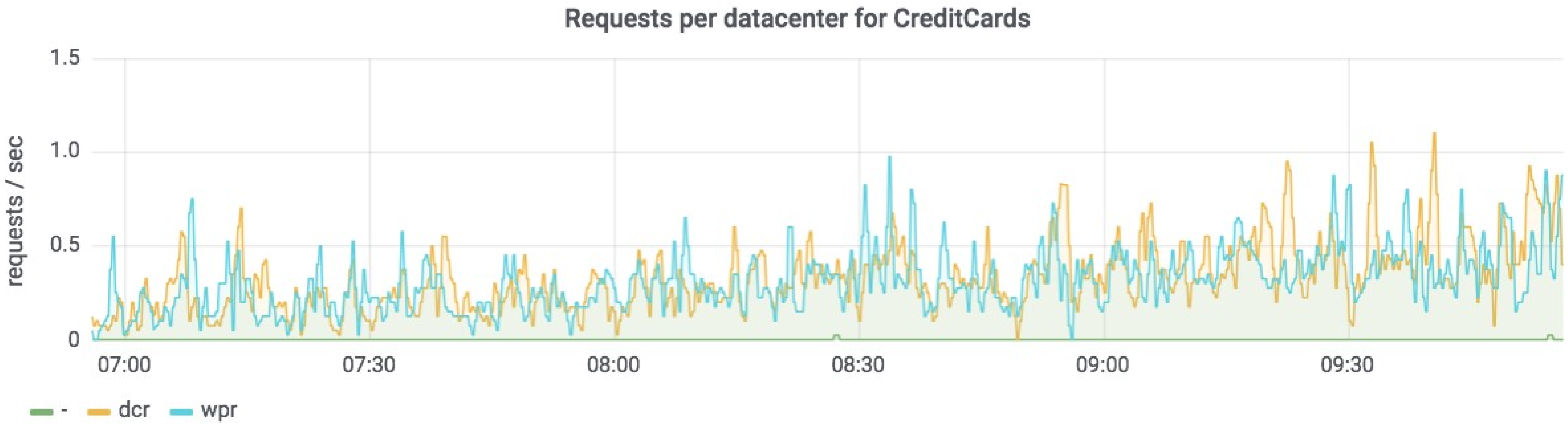
Team Overview (2)



Team Overview (3)



Team Overview (4)



🏫 Educate others during onboarding and workshops

PromQL Workshop: Example Assignment

Selecting a range vector in Prometheus is done by appending a time window specification between square brackets to your metric (for example: my_metric[1m] selects 1 minute). These ranges allow the use of all sorts of functions in Prometheus that manipulate the data.

You can also have Prometheus calculate the change in the number of logged in customers using functions like delta() or deriv()
.

delta : change in value between the first and last value of a time series in a range vector (time range)

idelta: change in value between the 2 last values of a time series in a range vector (time range)

derive: per-second derivative of a time series in range vector

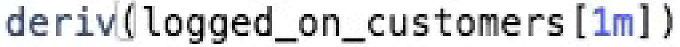
These functions should only be used with GAUGES. Note that the idelta function is somewhat less useful as it depends on the scrape interval in order to give it meaning.

Objective: Understand the delta(), deriv() and idelta() functions

1. Use the 'logged_on_customers' metric
2. Add a panel showing the per second change in the number of logged on customers for each site

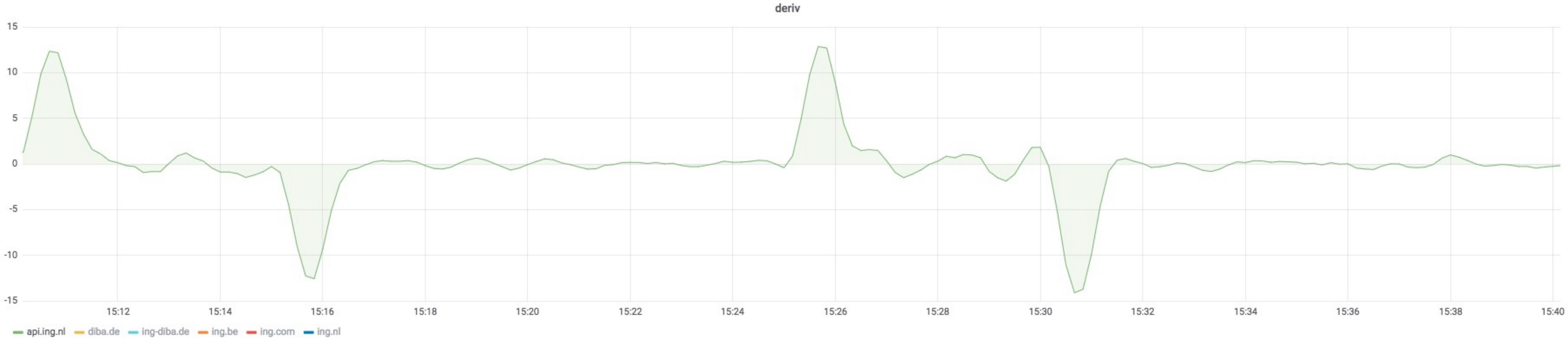
PromQL Workshop: Example Solution

You should have filled in: "deriv(logged_on_customers[1m])”



```
deriv(logged_on_customers[1m])
```

The graph should look similar to this:

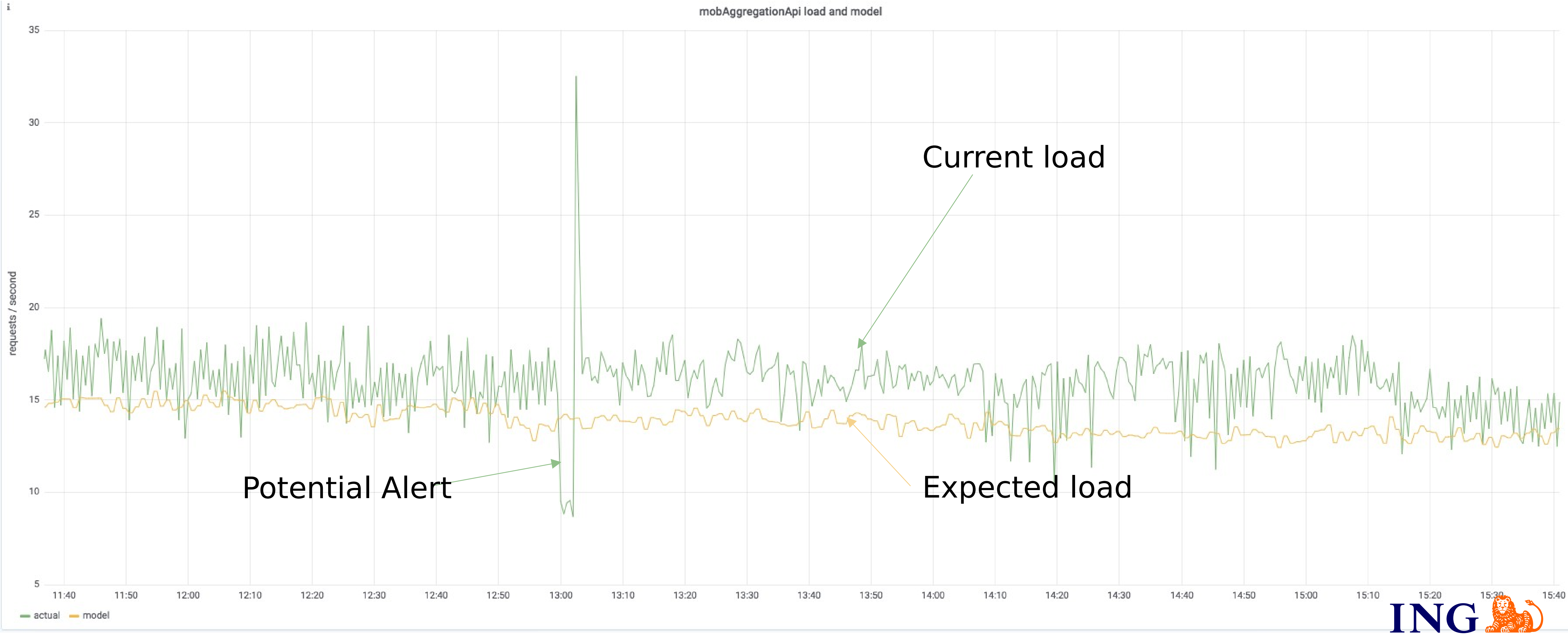


Difference between Delta and Deriv:

Delta shows you the **difference** between two points of time where the two valuables are subtracted from each other. These two valuables are selected based on the given time frame (in this case 1 min). On the other hand, Deriv (v range-vector) calculates the per-second **derivative** of the time series in a range vector v, using simple linear regression. Deriv calculates the slope of the graph.

Notify when things are different than expected

Model Builder



Current load

Potential Alert

Expected load

Model Builder 📈

Input	Currently we support GAUGES and COUNTERS as input
modelType	AveragingModel. Prediction based on values in buckets
Output	**model**_http_request_rate. Model as sample in Prometheus

Questions?