

A close-up photograph of a sunflower's petals, showing their vibrant yellow color and dark veins. The petals are arranged in a fan-like pattern, with some pointing towards the top left and others towards the bottom right. The background is a clear blue sky with soft, wispy white clouds.

OpenShift Infrastructure Monitoring with Prometheus

Ulrike Klusik
Senior Consultant

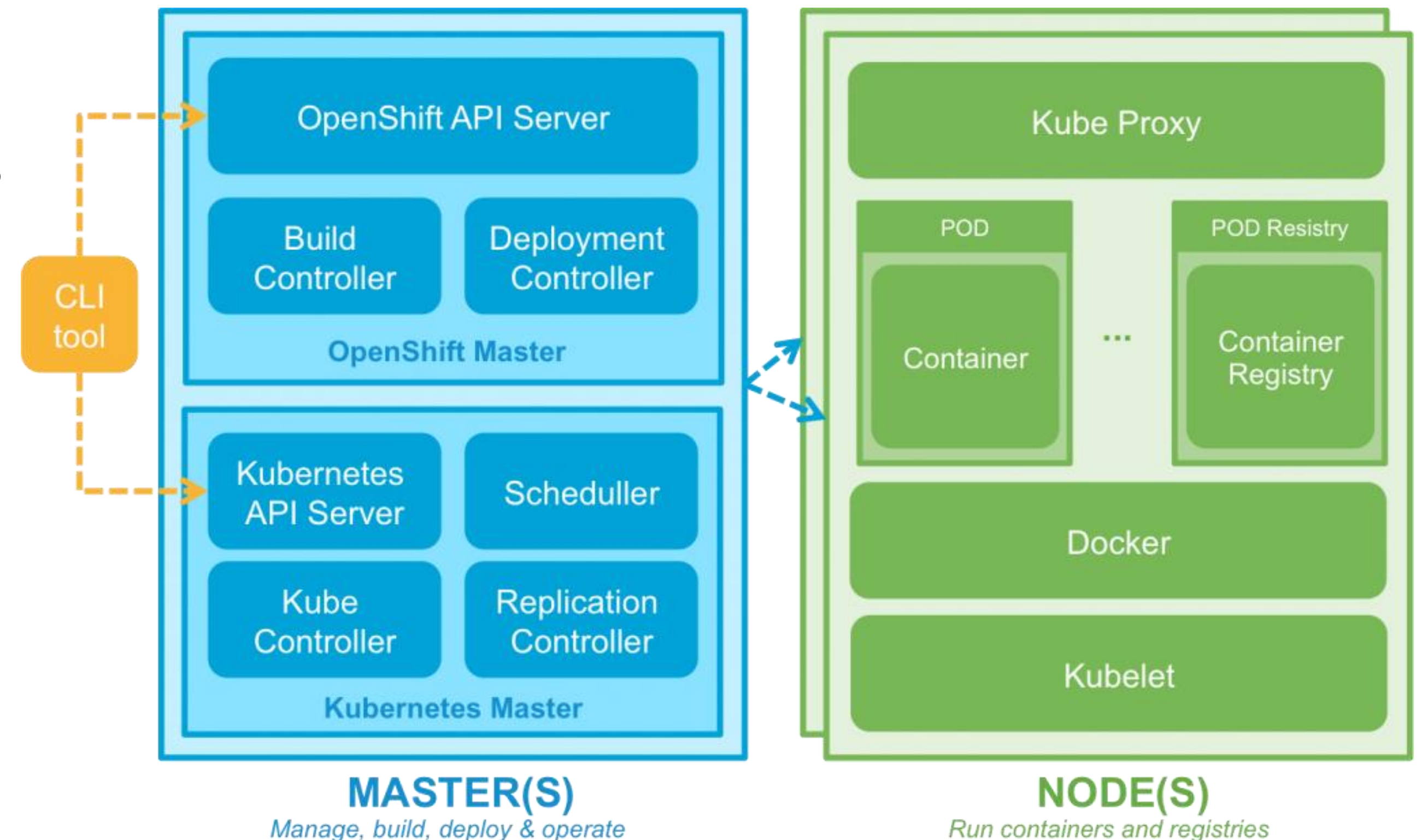
28.5.2019

Agenda

- Overview of OpenShift and Prometheus
- Architecture
- Demo Dashboards
- Configuration management
- Coping with High cardinality Metrics
- Conclusions

Overview OpenShift

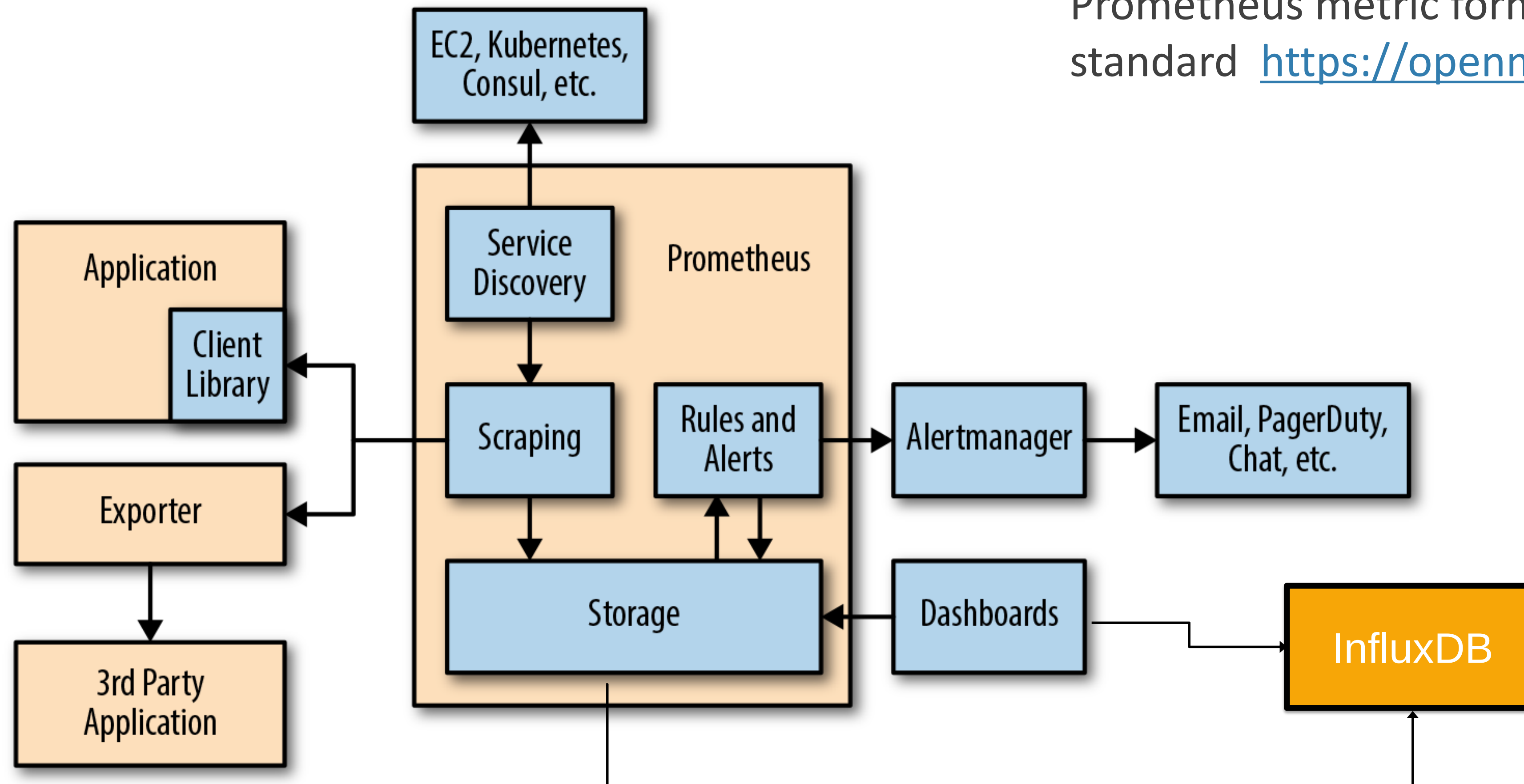
- Kubernetes Version from RedHat,
Some added features:
 - Container Registry/Image Streams
 - Router/HAProxy
- Also as OpenSource version OKD available



<https://blog.octo.com/wp-content/uploads/2015/05/Architecture-OpenShift-v3-OCTO-Technology-1024x619.png>

Prometheus Architecture

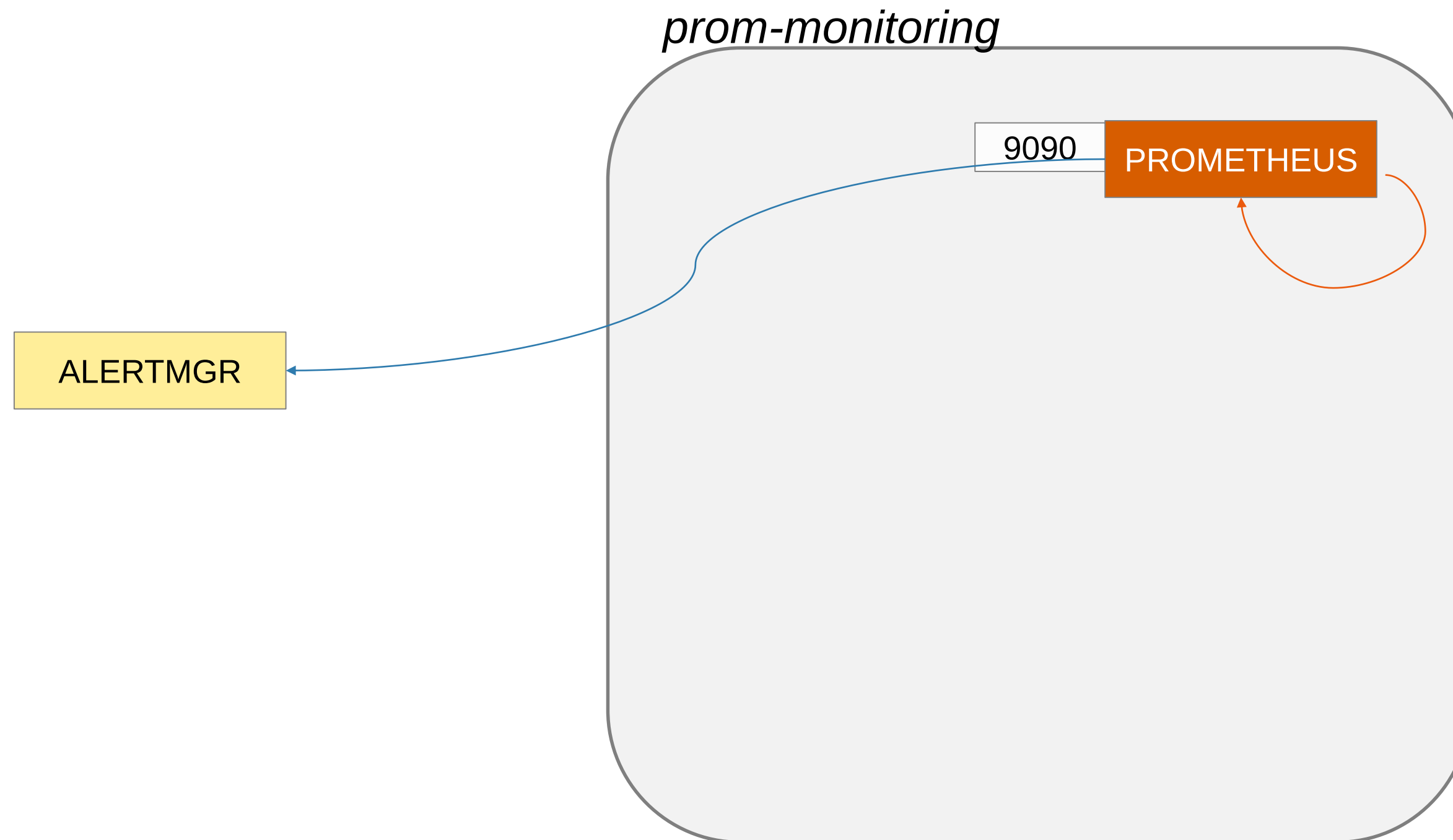
Prometheus metric format base for standard <https://openmetrics.io/>



Source: [Prometheus: Up & Running by Brian Brazil](#)

Prometheus OpenShift Monitoring

Monitoring the Monitor



- Prometheus exposes metrics about itself, which is used for „self-monitoring“:

- all targets available
- notification working
- remote write working

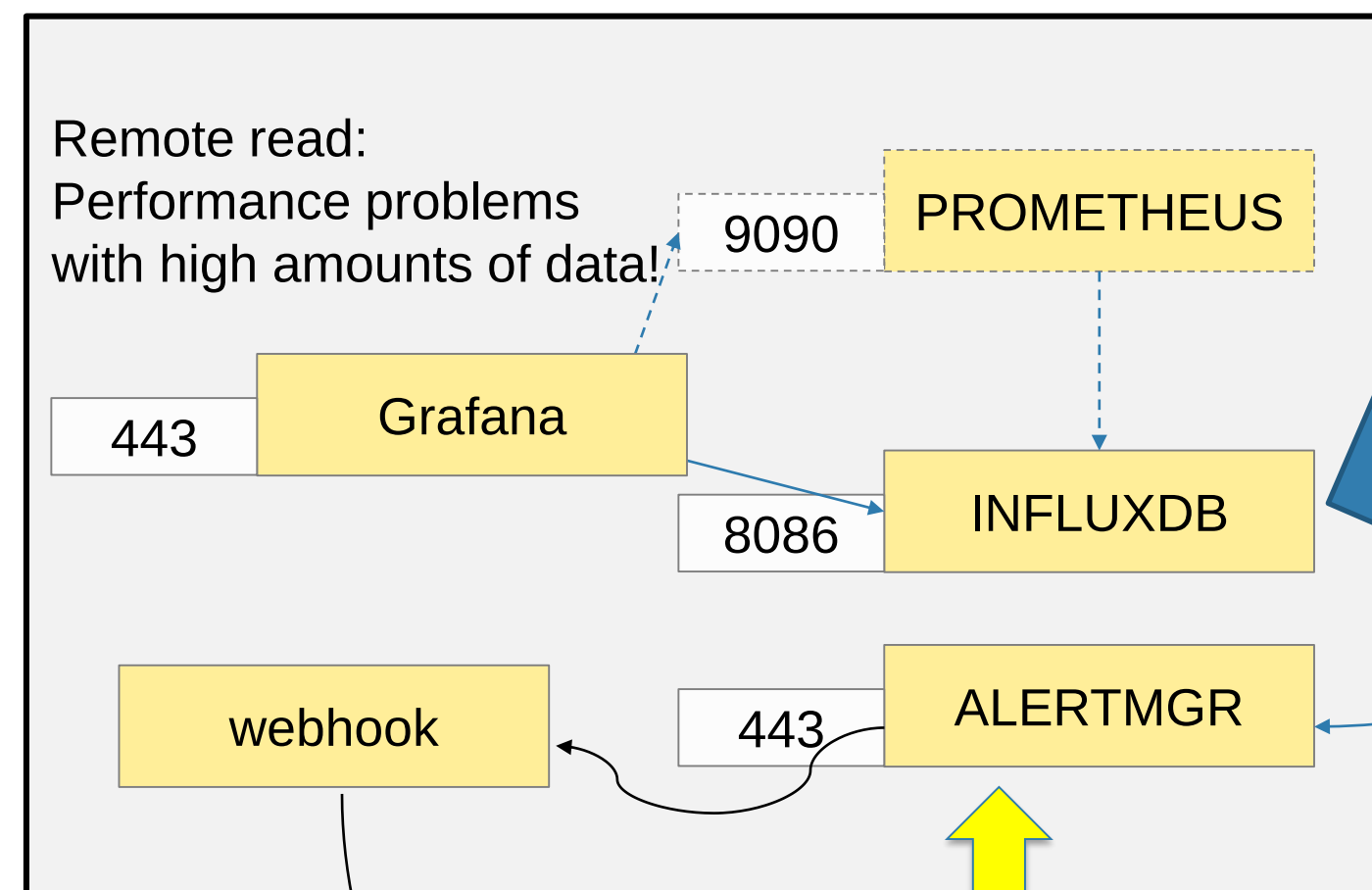
External availability check:

- Alert chain via DeadMansSwitch alert,
e.g via `check_http` from `naemon`.

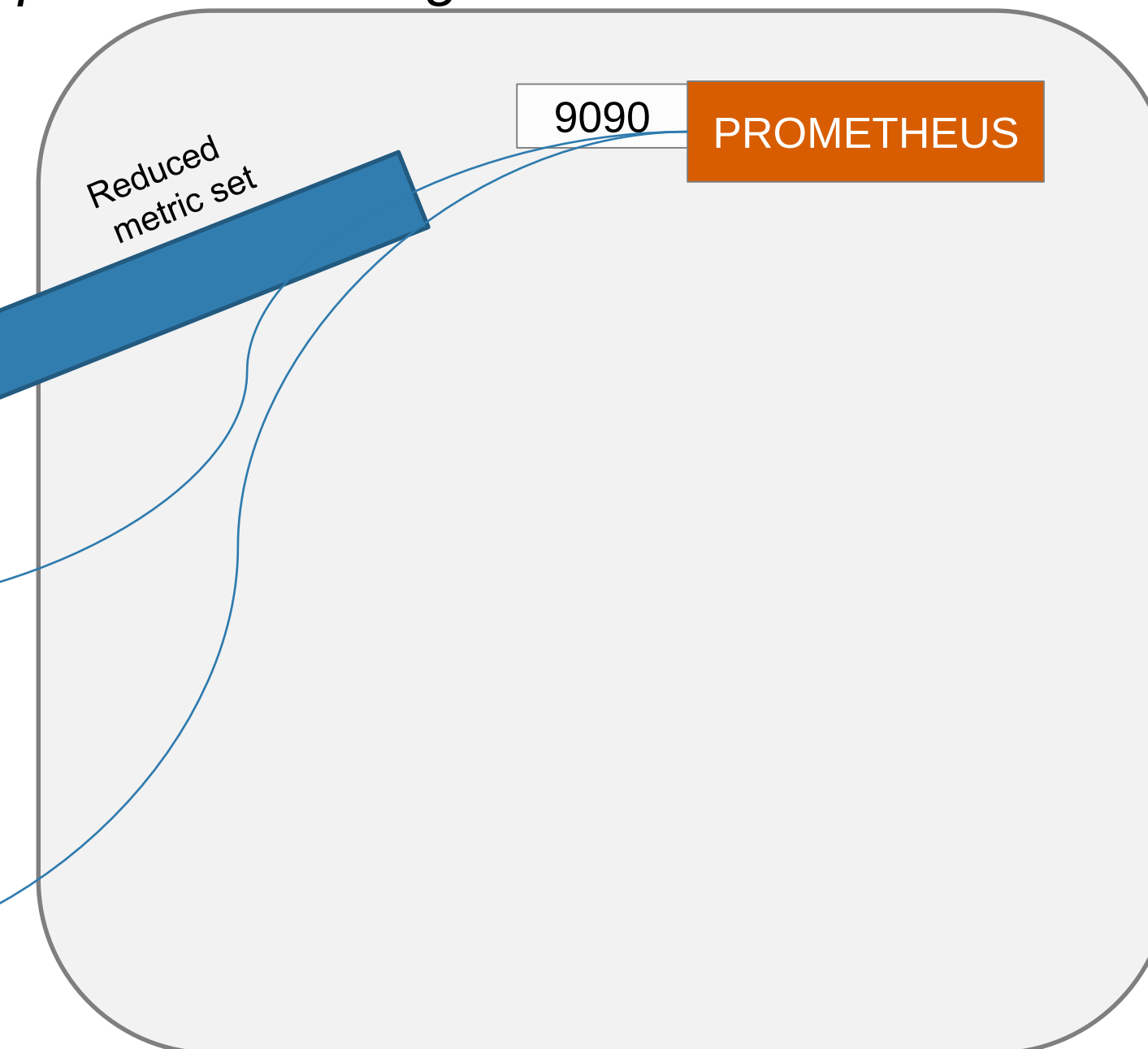
Prometheus OpenShift Monitoring

Long Term Storage and Alert Notifications

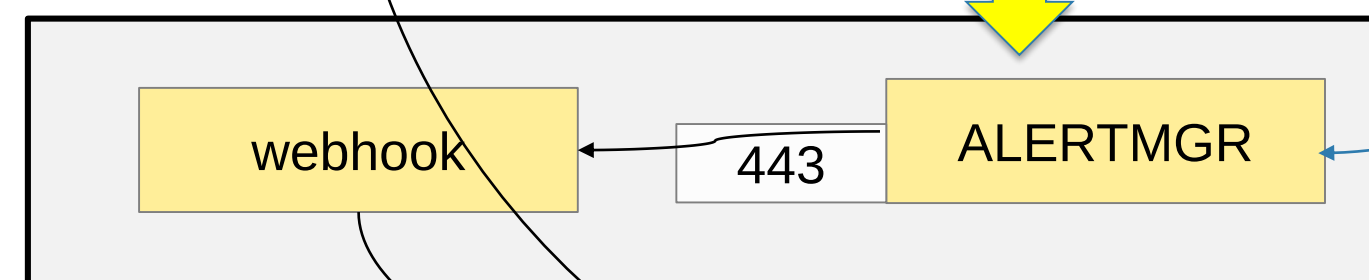
OMD server1



prom-monitoring

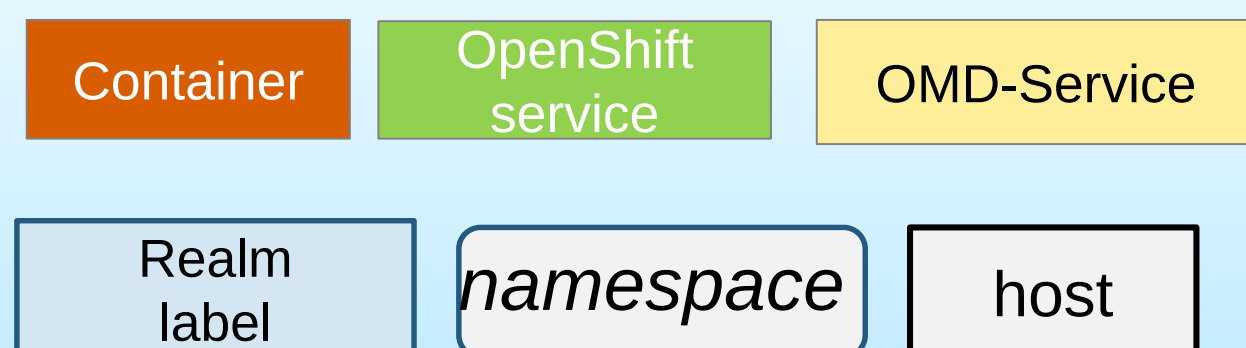


OMD server2



clustered

ITSM-Suite



OMD sites provide

- InfluxDB: to stored selected metrics via remote write
- Grafana: to visual the data
- Alertmanager: to receive the alerts, deduplication and notification
- Webhook (custom): to create / close incident ticket in ITSM Solutions

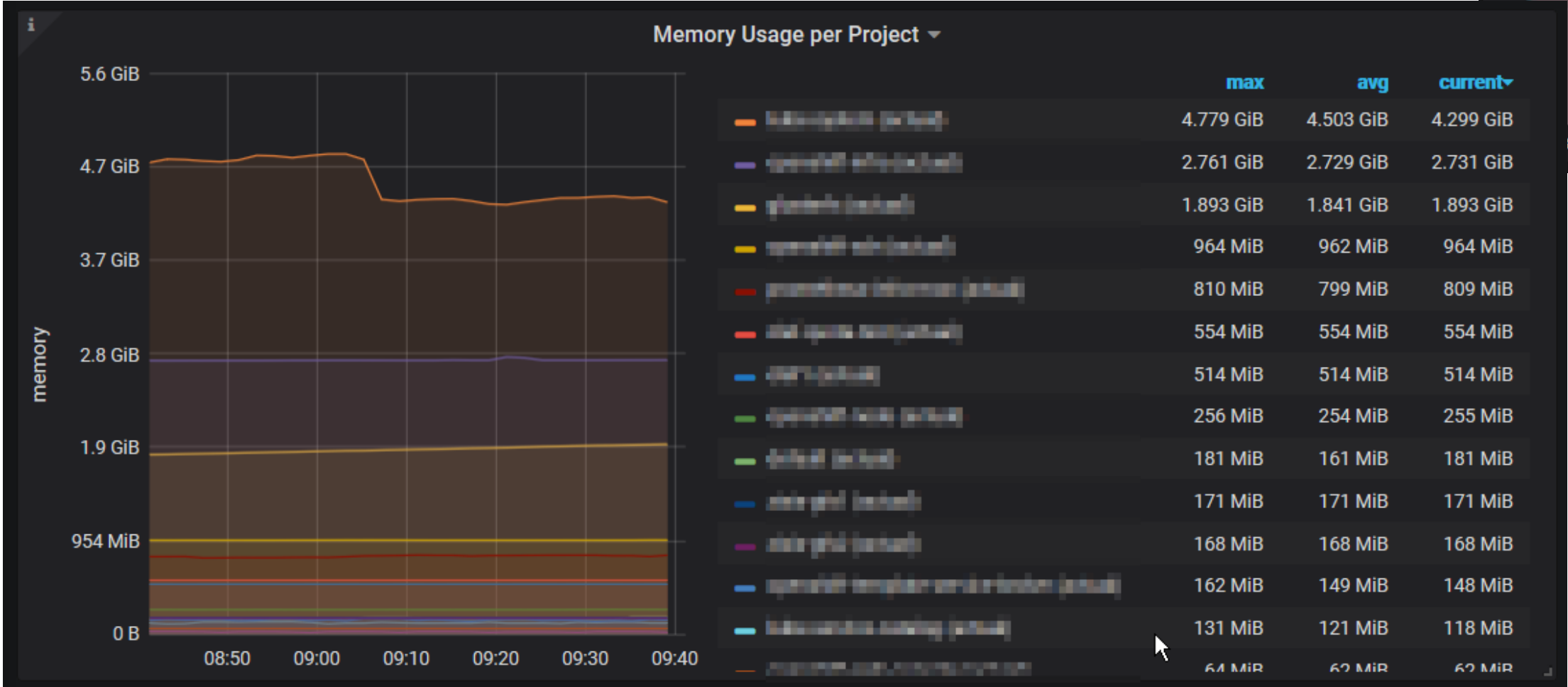
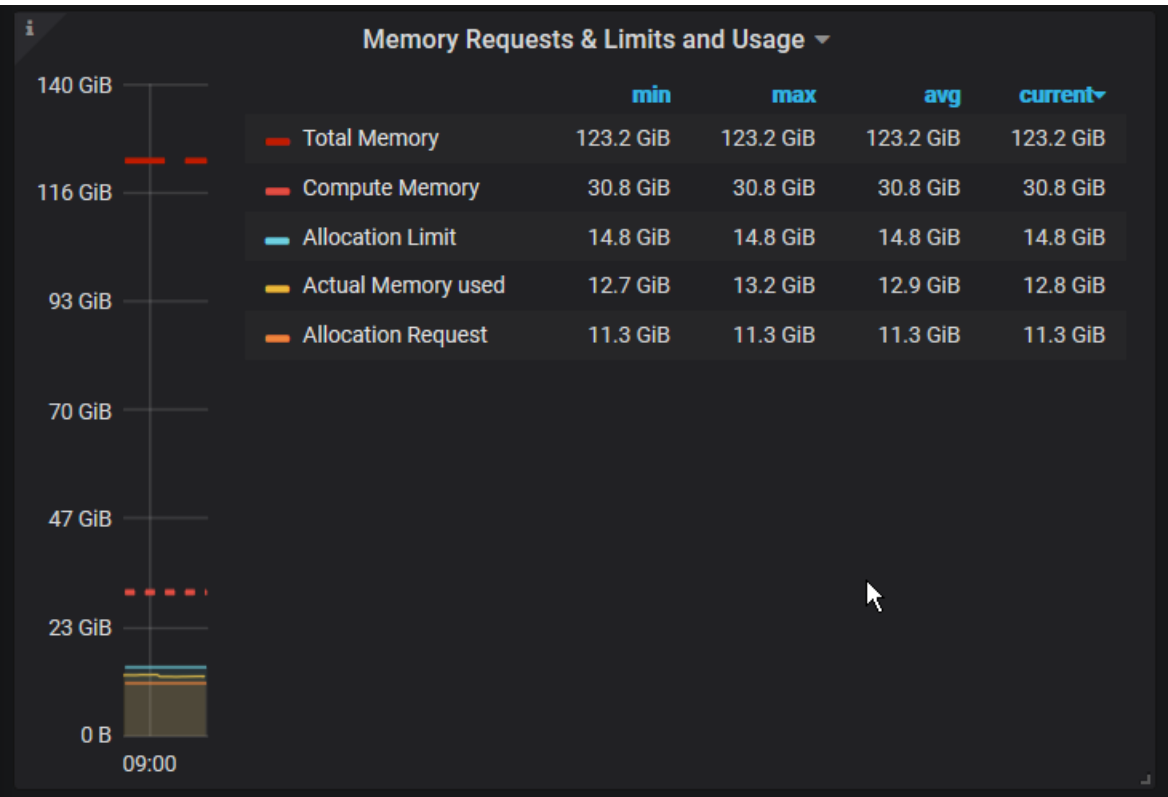
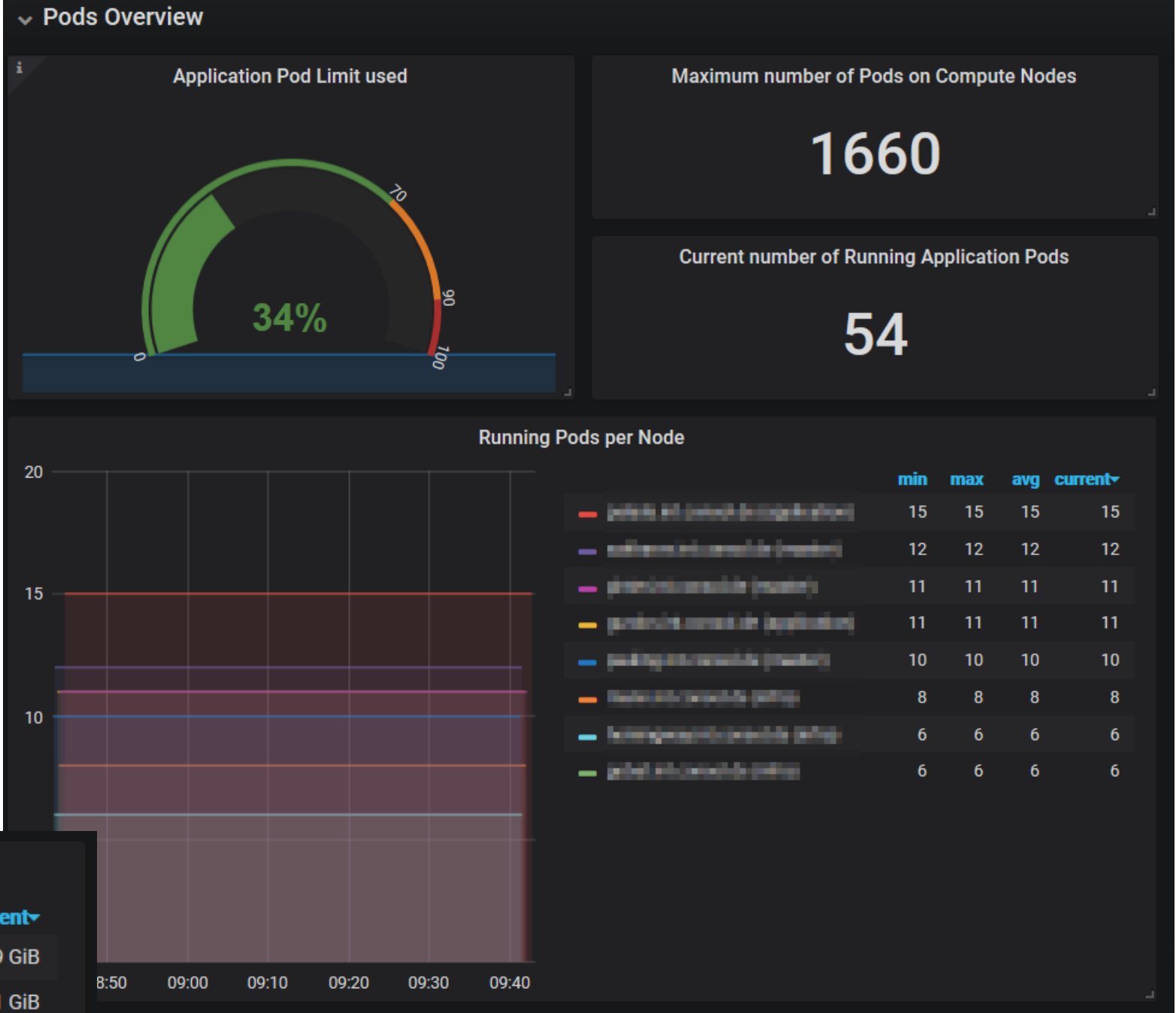
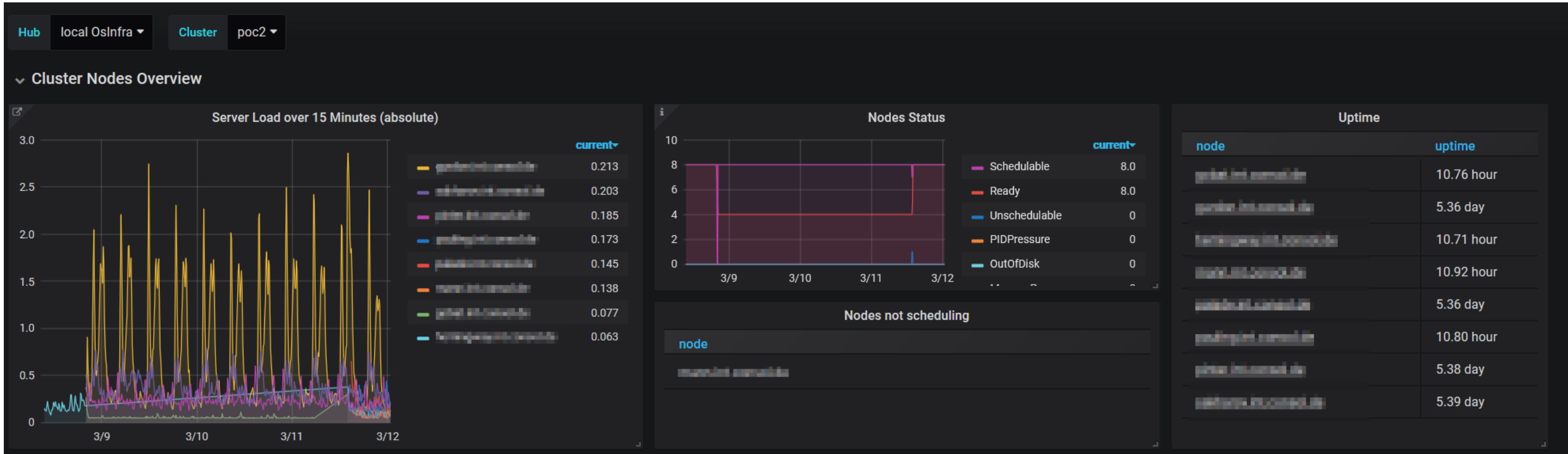
Central solution:

- One installation can be used for several clusters.
- Alertmanager and InfluxDB should be local to the cluster. E.g. per datacenter.

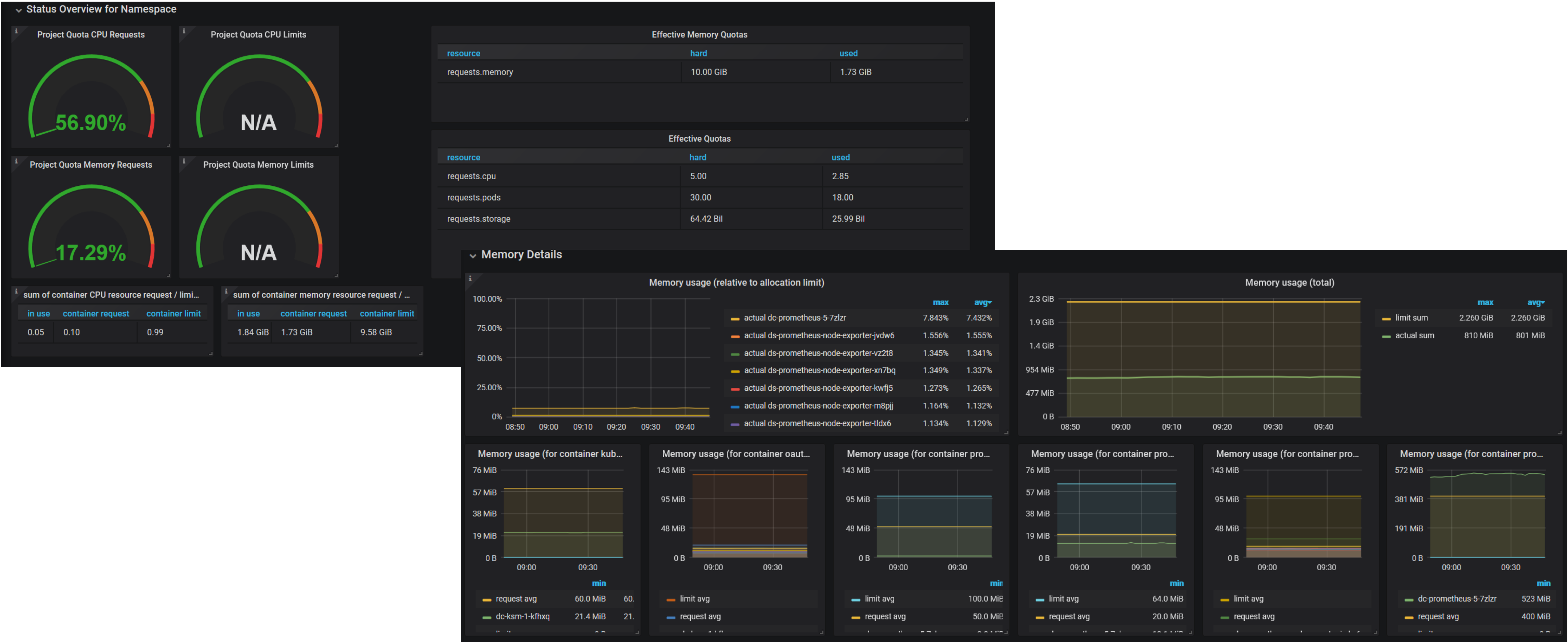
DEMO

- Grafana Dashboards:
 - Cluster Overview
 - Project Resources
- Prometheus:
 - Alert Details
 - Target overview

Dashboard Cluster Overview



Dashboard Project Resources



Dashboard Alert Details

via influxDB / Cluster Alert Overview

HUB

local OsInfra

cluster

poc2

Current Alerts for Cluster

Last Active	alertname	routeurl	severity
2019-03-11 14:05:05	KubernetesInstanceDown	https://rt-prometheus-	CRITICAL
2019-03-11 14:04:17	KubePodNotReady	https://rt-prometheus-	critical
2019-03-11 14:04:17	KubeDaemonSetRolloutStuck	https://rt-prometheus-	
2019-03-11 14:04:00	KubeNodeNotReady	https://rt-prometheus-	
2019-03-11 14:04:00	KubeAPIErrorsHigh	https://rt-prometheus-	
2019-03-11 14:04:00	KubeAPIErrorsHigh	https://rt-prometheus-	

Alerts

Show annotations

DeadMansSwitch (1 active)

alert: DeadMansSwitch

expr: vector(1)

labels:
priority: OK
selfmonitoring: "true"
severity: none

annotations:
description: This is a DeadMansSwitch meant to ensure that the entire Alerting pipeline is functional.
summary: Alerting DeadMansSwitch

Labels	State	Active Since	Value
alertname="DeadMansSwitch" priority="OK" selfmonitoring="true" severity="none"	FIRING	2019-03-08 19:02:01.11767761 +0000 UTC	1

KubeAPIErrorsHigh (0 active)

KubeAPIErrorsHigh (0 active)

KubeCPUOvercommit (0 active)

alert: KubeCPUOvercommit

expr: sum(namespace_name:kube_pod_container_resource_requests_cpu_cores:sum) / sum(node:node_num_cpu:sum) > (count(node:node_num_cpu:sum) - 1) / count(node:node_num_cpu:sum)

for: 5m

labels:
event_id: "27200"
priority: medium
severity: warning
sub_source: cluster

annotations:
description: The CPU requests from container definitions of all pods in cluster is larger than (All CPU) - (cpu resources of avg node).
summary: Overcommitted CPU resource requests on Pods, cannot tolerate node failure.

OS Infrastructure Monitoring

Folie 10

Prometheus Targets

Service Discovery

- glusterfs-heketi (1/3 active targets)
- kube-state-metrics (1/27 active targets)
- kubernetes-apiservers (3/23 active targets)
- kubernetes-cadvisor (8/8 active targets)
- kubernetes-controllers (3/23 active targets)
- kubernetes-kubelet (8/8 active targets)
- node-exporter (8/8 active targets)
- openshift-routers (3/23 active targets)
- prometheus (1/22 active targets)

Prometheus Alerts Graph Status ▾ Help

Targets

All Unhealthy

glusterfs-heketi (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://heketi-storage.glusterfs.svc:8080/metrics	UP	glusterfs="heketi-storage-service" heketi="storage-service" instance="heketi-storage.glusterfs.svc:8080" job="glusterfs-heketi"	1m17.327s ago	9.363ms	

kube-state-metrics (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://10.129.0.98:8080/metrics	UP	instance="10.129.0.98:8080" job="kube-state-metrics"	22.125s ago	129.7ms	

kubernetes-apiservers (3/3 up) show less

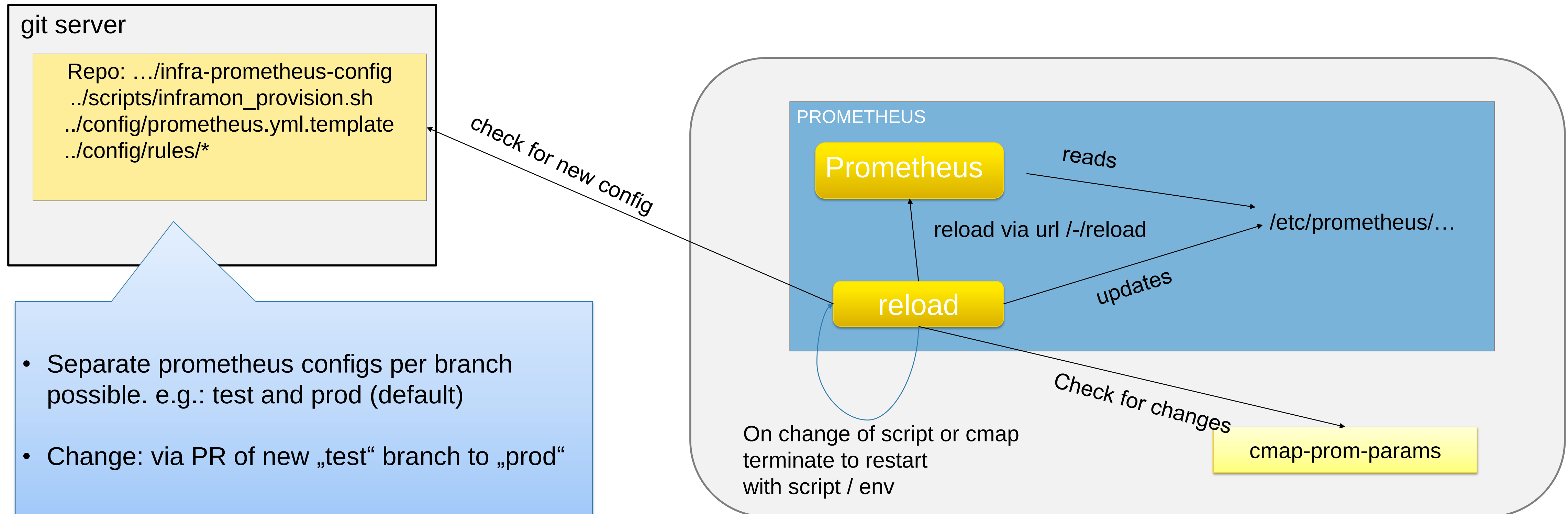
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://10.0.6.154:443/metrics	UP	component="apiserver" instance="10.0.6.154:443" job="kubernetes-apiservers" provider="kubernetes"	46.504s ago	163ms	
https://10.0.6.247:443/metrics	UP	component="apiserver" instance="10.0.6.247:443" job="kubernetes-apiservers" provider="kubernetes"	1m51.586s ago	361.4ms	
https://10.0.6.254:443/metrics	UP	component="apiserver" instance="10.0.6.254:443" job="kubernetes-apiservers" provider="kubernetes"	19.4s ago	88.09ms	

kubernetes-cadvisor (8/8 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://10.0.6.149:10250/metrics/cadvisor	UP	beta_kubernetes_io_arch="amd64" beta_kubernetes_io_os="linux" glusterfs="storage-host" instance="10.0.6.149:10250" job="kubernetes-cadvisor" kubernetes_io_hostname="hemingway.int.consol.de" logging_infra_fluentd="true" node="hemingway.int.consol.de" no	1m2.681s ago	165.6ms	

Prometheus Configuration Management

- Use Case: central configuration for several clusters, need e.g. cluster specific labels, Alertmanager and InfluxDb connection



External storage of Prometheus metric data, especially for Long Term Storage

- Federation:
 - Scrape metrics from Prometheus as source
 - Pro: limiting metrics scraped, can be queries in PromQL.
 - Cons: timestamp from scraped Prometheus, original timestamp is lost
- Thanos Store:
 - Store all metrics from Prometheus into block storage (e.g. S3)
 - Pro: can be queries via Thanos Query in PromQL
 - Cons: ALL metrics must be stored
- Remote Write/Read:
 - Write selected metrics to another time series database (e.g. InfluxDB, Elastic, PostgreSQL Timescale, Thanos Receiver(alpha))
Read Metrics via remote read mechanism
 - Pro: limiting metrics exported, metrics keep original timestamp
 - Cons: remote read metrics access needs to transfer too much data to Read Prometheus alternative

=> Our current choice: Remote Write To InfluxDB, Central Grafana Dashboards via InfluxDB data source

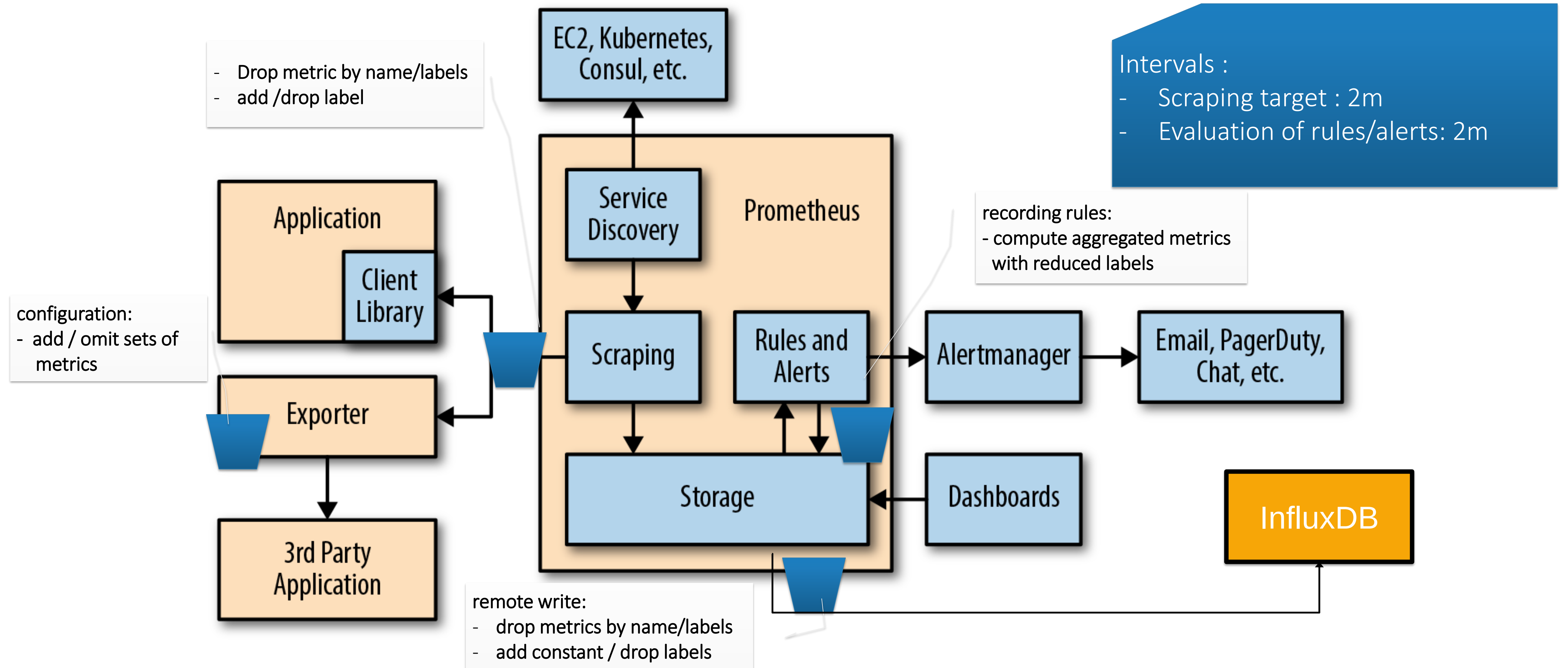
How to cope with large amounts of metrics

Use case: Metrics provided only very detailed, but aggregated metrics wanted.

Metrics With Very High Cardinality are e.g.

- Api-Server metrics:
 - per API-URL and access method!
- CPU metrics: `container_cpu_usage_seconds_total`
 - cAdvisor before v0.29/ before OpenShift 3.10: container cpu metrics only per single CPU Core!
- HAProxy metrics:
 - Detailed metrics per route / service and implementing pod
- How to find the high cardinality metrics
PROMQL: `topk(30, count by (__name__, job)({__name__=~".+"}))`

Influencing Metrics Stored



Reducing the Metric Volume for Long Term Storage

- Note:
Prometheus provides no mechanism to delete metrics in its time series DB, apart from expired by retention time.
- Our approach:
 - Drop not needed metrics with high cardinalities during scraping
 - Set the Prometheus storage retention to a few days.
Tradeoff between persistent storage volume and detailed analysis
 - Use aggregate metrics for long term storage
 - Only export specific metrics especially aggregated write remote write
- This is successfully running on OpenShift Clusters with upto ca. 55 nodes.

Links

- Standard Metrics:
 - **“A Deep Dive into Kubernetes Metrics” from Bob Cotton**
<https://blog.freshtracks.io/a-deep-dive-into-kubernetes-metrics-b190cc97f0f6>

Conclusions and Future Topics

- Prometheus can already be used to monitor OpenShift 3.6 clusters and higher
- Some limitations due to older Kubernetes service versions
- High cardinality metrics:
 - Many can already be dropped during scraping.
 - Longer retention: keep mostly only aggregates in external influx DB
- The presented solution can be used to consolidate metrics / alerts over several clusters in central database and Dashboards.
Limitation only by geographical distribution and network availability
- Open:
 - High Availability and deduplication of metrics in central storage



Noch Fragen?



Vielen Dank!



ConSol

Consulting & Solutions Software GmbH

Franziskanerstr. 38

D-81669 München

Tel.: +49-89-45841-100

info@consol.de

www.consol.de

Twitter: @consol_de