Integrating Prometheus: Towards a Unified Monitoring

CHECKMK CONFERENCE #6 – MUNICH, APRIL 29, 2020

Wontek Hong
Developer
tribe29

tribe29

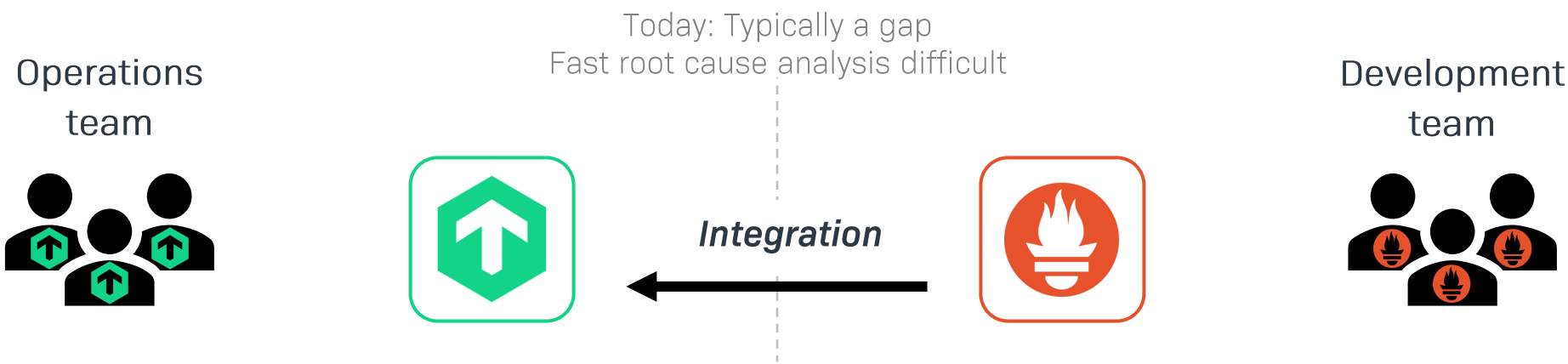
Agenda

1. PROMETHEUS: AN OVERVIEW
2. THE INTEGRATION: WHY AND HOW
3. DEMO
4. Q&A

Prometheus: Metric-focused monitoring tool

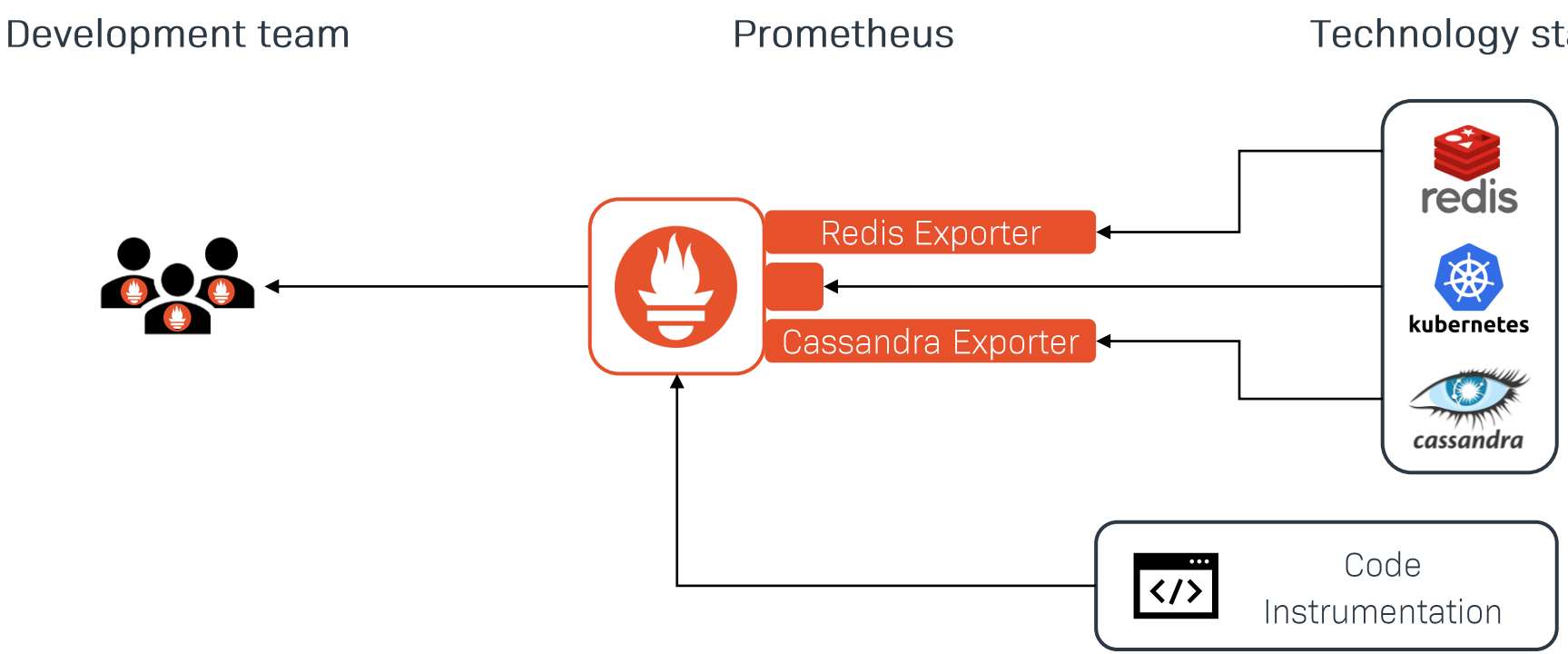
- Open-source ecosystem for metric monitoring and alerting
- Public since 2015, purely community driven and built
- Records real-time metrics in a time series database
- Many applications already expose Prometheus metrics
- Own query language: PromQL

An Integration: Bridging the gap between teams



One integrated view of Dev and Infra Ops team to jointly prevent and fix problems faster

Development teams employ Prometheus to instrument their code and monitor applications



Scrape targets: Data sources for Prometheus

Exporters

- Intermediate layer between application and Prometheus database
- Converts application metrics to be Prometheus compatible
- All Exporters are open source
  - today 161 available Exporters
  - 13 official Exporters (maintained by Prometheus GitHub organization)

Code instrumentation

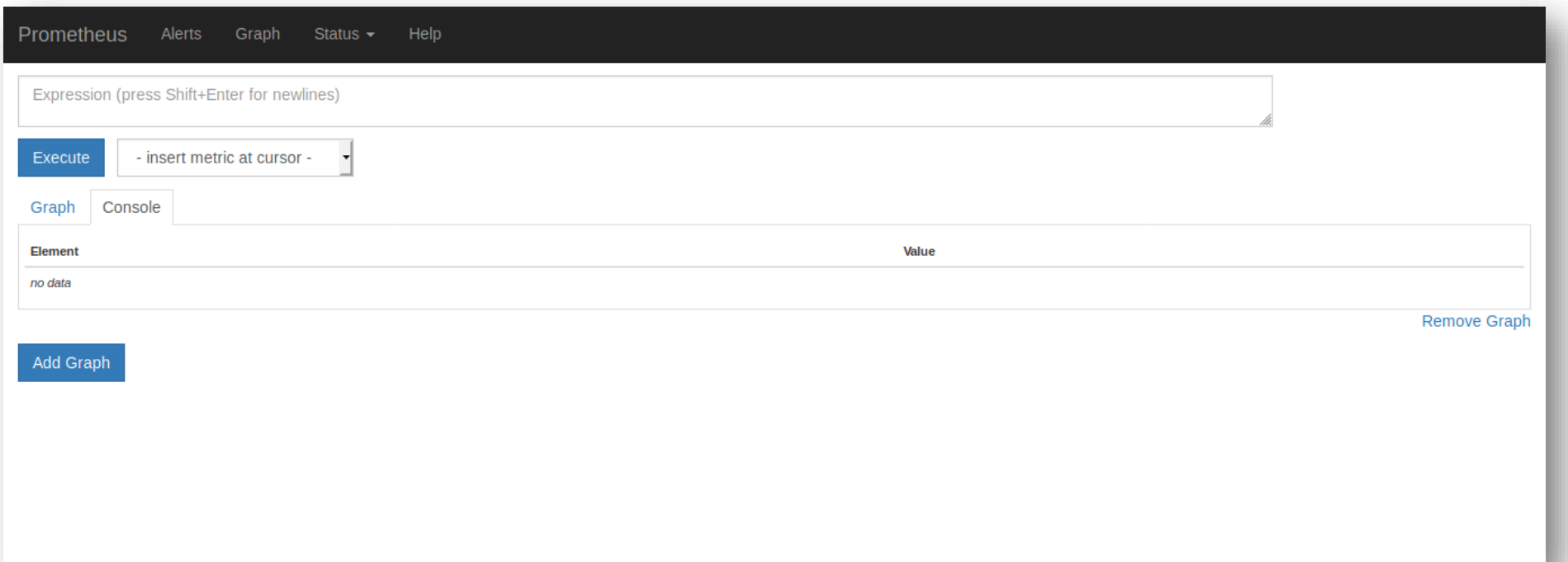
- Officially available for Python, Ruby, Go, Java/Scala; many more unofficially

Top 10 exporters

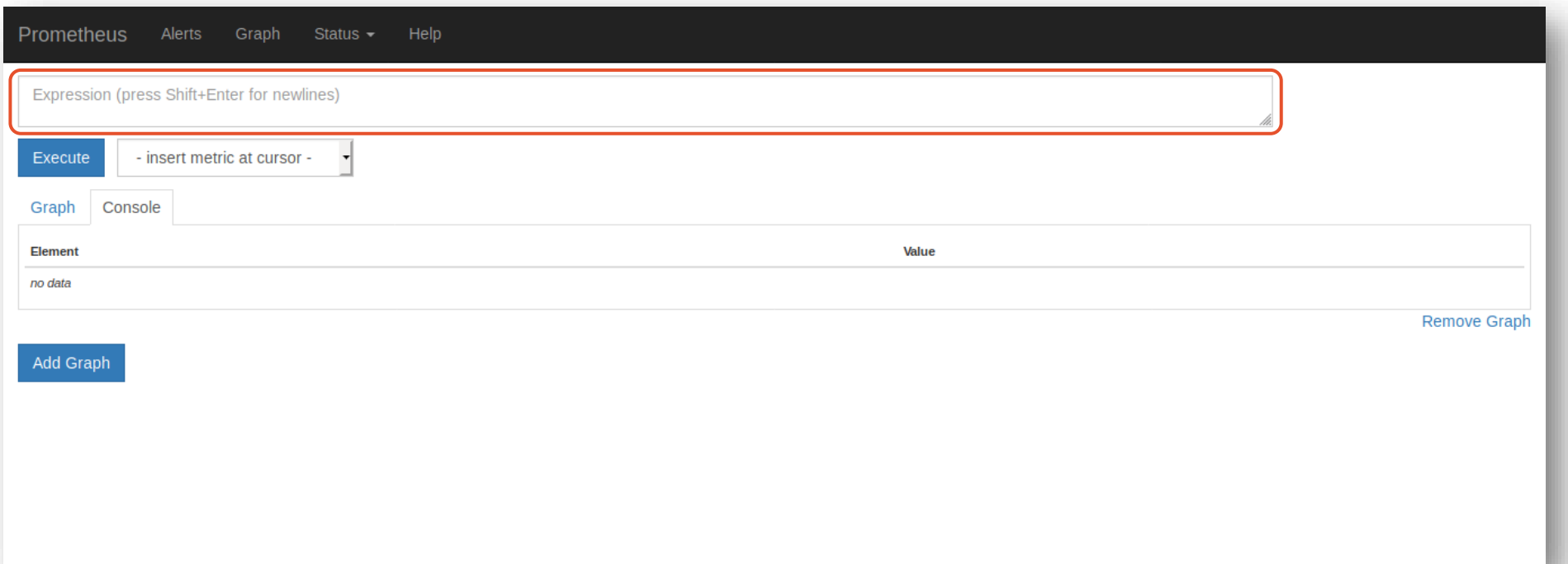
161 exporters listed on official Prometheus Homepage

Rank	Name	Application
1	cAdvisor	Miscellaneous
2	Node exporter	Hardware
3	kube-state-metrics	Miscellaneous
4	JMX exporter	Other monitoring systems
5	Blackbox exporter	Miscellaneous
6	Redis exporter	Databases
7	ElasticSearch exporter	Databases
8	PostgreSQL exporter	Databases
9	MySQL server exporter	Databases
10	Google‘s mtail log data extractor	Logging

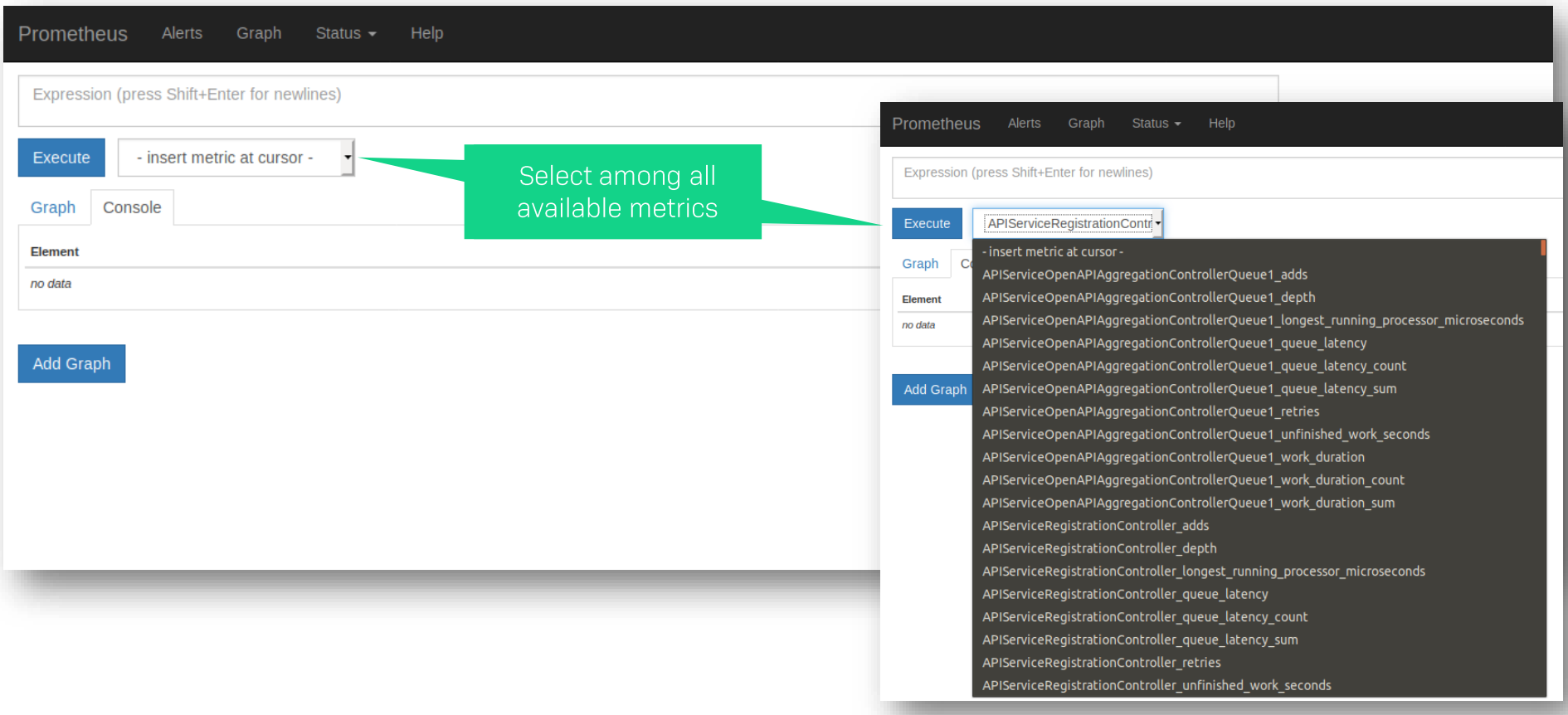
Prometheus interface: ‚Expression Browser‘



Prometheus interface: ‚Expression Browser‘



Prometheus interface: ‚Expression Browser‘



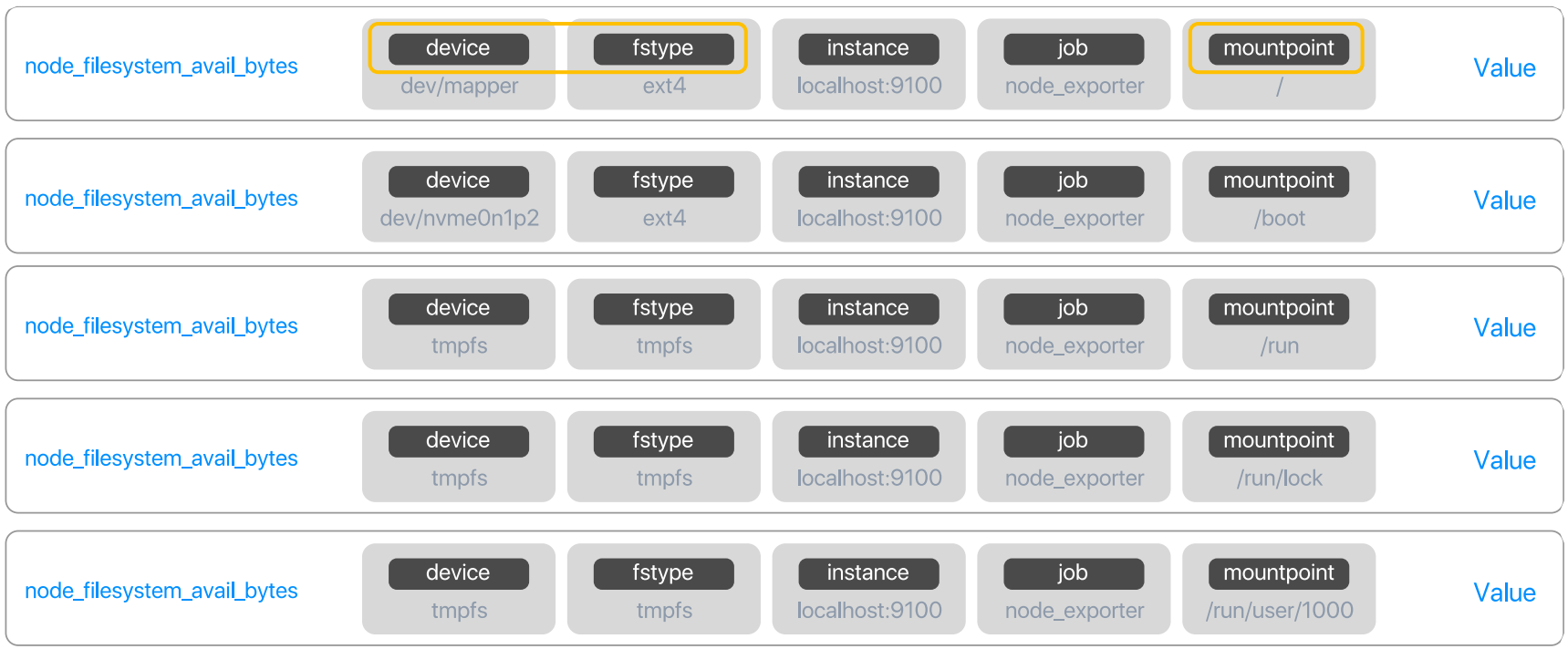
Prometheus interface: ‚Expression Browser‘

Element	Value
node_filesystem_avail_bytes{device="/dev/mapper",fstype="ext4",instance="localhost:9100",job="node_exporter",mountpoint="/"}	374393245696
node_filesystem_avail_bytes{device="/dev/nvme0n1p1",fstype="vfat",instance="localhost:9100",job="node_exporter",mountpoint="/boot/efi"}	519733248
node_filesystem_avail_bytes{device="/dev/nvme0n1p2",fstype="ext4",instance="localhost:9100",job="node_exporter",mountpoint="/boot"}	510828544
node_filesystem_avail_bytes{device="tmpfs",fstype="tmpfs",instance="localhost:9100",job="node_exporter",mountpoint="/run"}	1664176128
node_filesystem_avail_bytes{device="tmpfs",fstype="tmpfs",instance="localhost:9100",job="node_exporter",mountpoint="/run/lock"}	5238784
node_filesystem_avail_bytes{device="tmpfs",fstype="tmpfs",instance="localhost:9100",job="node_exporter",mountpoint="/run/user/1000"}	1666330624

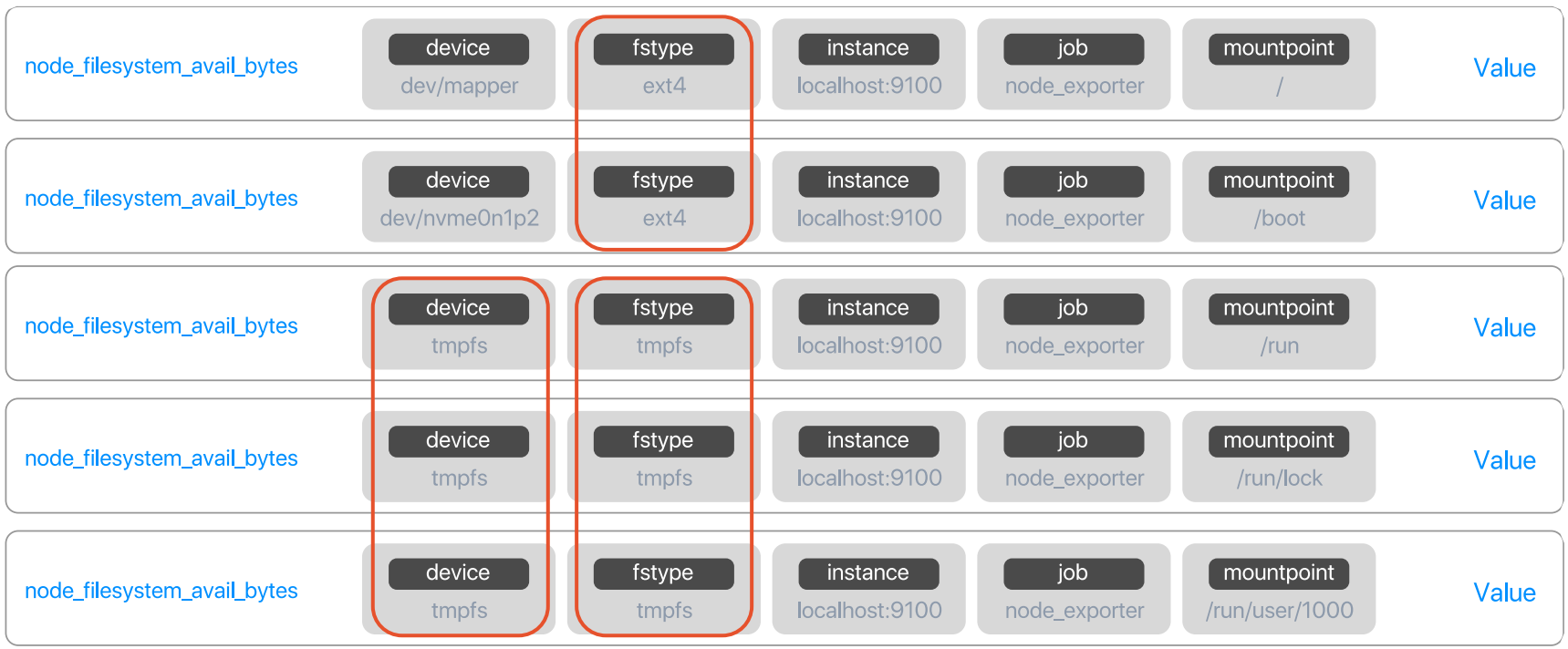
Metric = Name + Labels + Value

node_filesystem_avail_bytes	device: dev/mapper	fstype: ext4	instance: localhost:9100	job: node_exporter	mountpoint: /	Value
node_filesystem_avail_bytes	device: dev/nvme0n1p2	fstype: ext4	instance: localhost:9100	job: node_exporter	mountpoint: /boot	Value
node_filesystem_avail_bytes	device: tmpfs	fstype: tmpfs	instance: localhost:9100	job: node_exporter	mountpoint: /run	Value
node_filesystem_avail_bytes	device: tmpfs	fstype: tmpfs	instance: localhost:9100	job: node_exporter	mountpoint: /run/lock	Value
node_filesystem_avail_bytes	device: tmpfs	fstype: tmpfs	instance: localhost:9100	job: node_exporter	mountpoint: /run/user/1000	Value

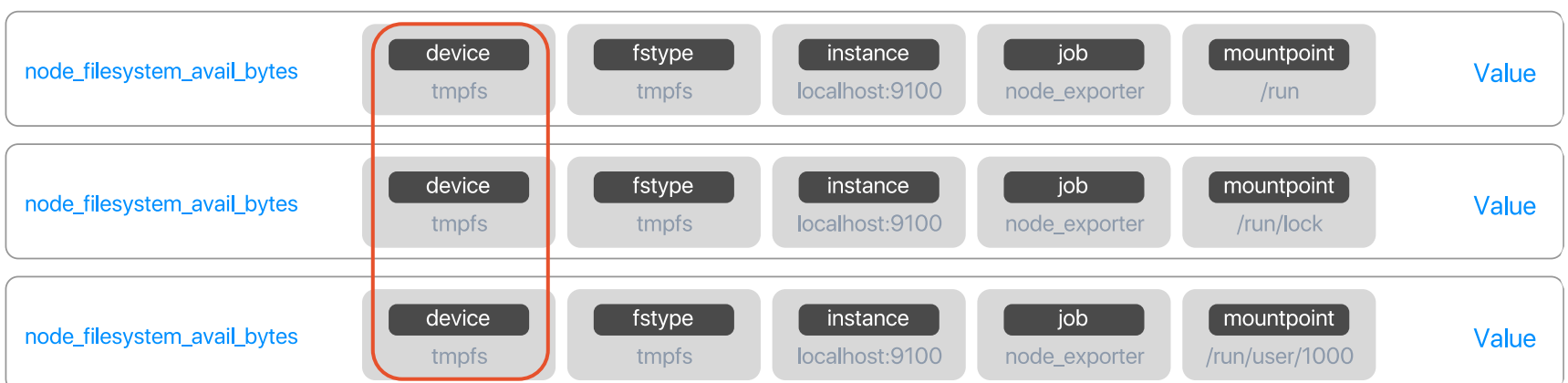
Metric = Name + Labels + Value



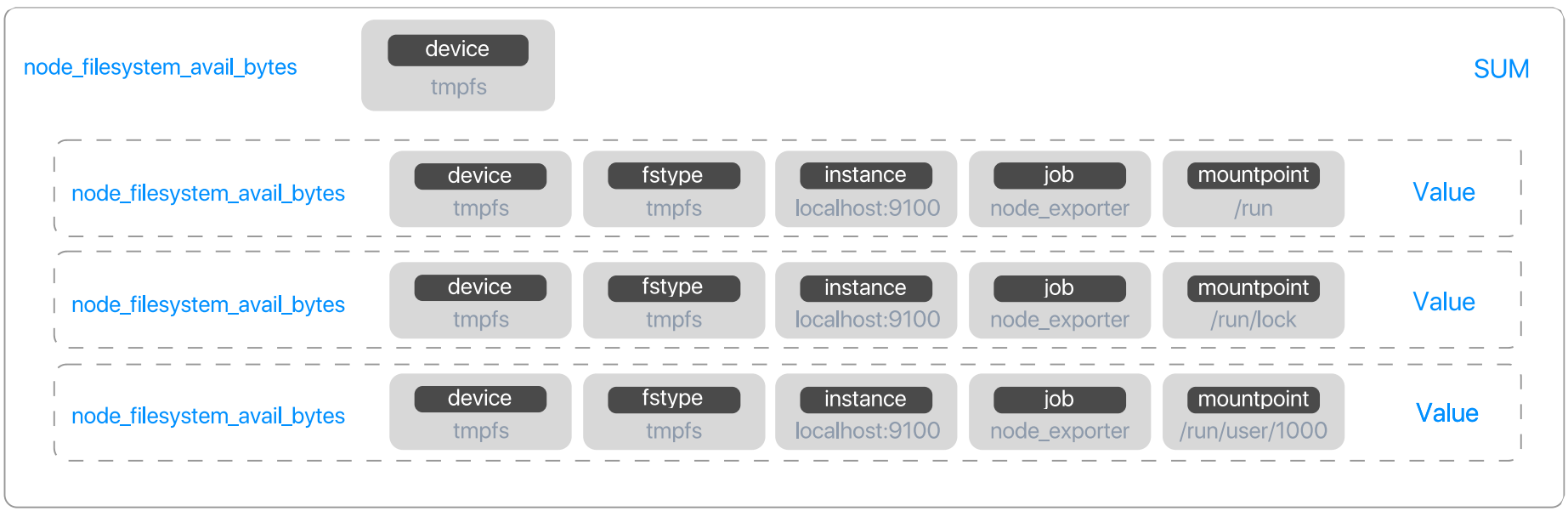
Metric = Name + Labels + Value



PromQL language



PromQL language



Key differences between Checkmk and Prometheus

Prometheus	Checkmk
• Metrics focus (‘host-less’)	• Hosts, services (state), metrics
• Nothing „out of the box“	• Pre-defined services and default thresholds
• Flexible aggregations through PromQL queries, slicing and dicing in all dimensions	• Very limited aggregation of metrics
• > 160 Exporters	• > 1800 plugins
Flexibility	**Complexity Reduction**

Best of both worlds: Combine Prometheus flexibility with Checkmk's contextual monitoring

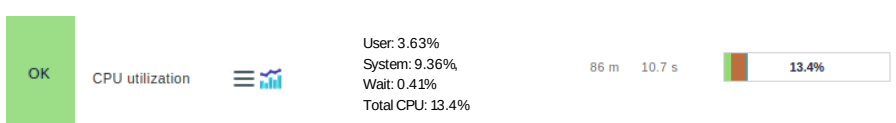
Prometheus

- Prometheus flexibility comes at the price of complexity and lack of context

node_memory_SwapFree_bytes{instance="localhost:9100",job="node_exporter"}	217284608
node_memory_SwapCached_bytes{instance="localhost:9100",job="node_exporter"}	353615872
node_memory_VmallocTotal_bytes{instance="localhost:9100",job="node_exporter"}	35184372087808
node_memory_VmallocUsed_bytes{instance="localhost:9100",job="node_exporter"}	62521344
node_memory_Inactive_bytes{instance="localhost:9100",job="node_exporter"}	1885241344

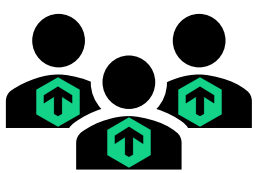
Checkmk

- Use Checkmk‘s powerful contextual monitoring to complement Prometheus’ flexibility



How it works: Special Agent for Prometheus

Operations team

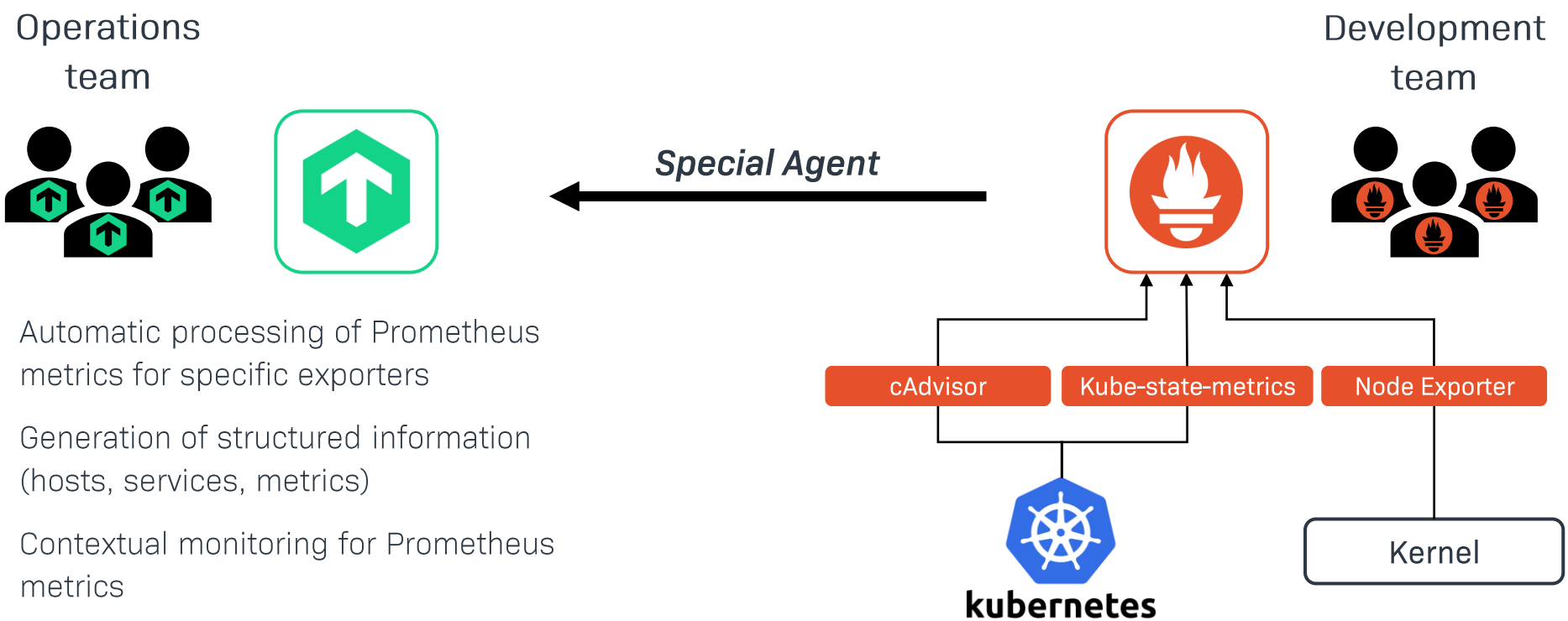


Special Agent

Development team

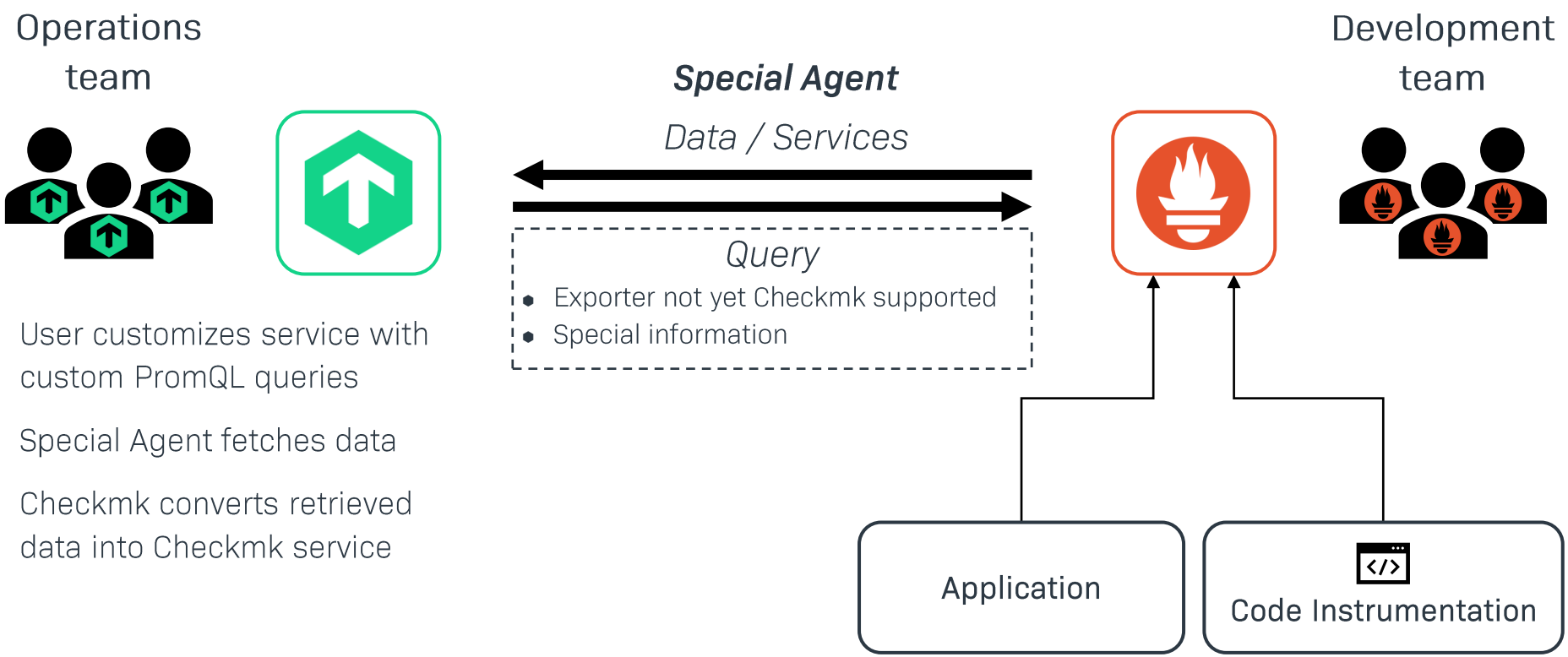
- Host configuration within Checkmk for Prometheus instance
- Datasource programs for configuration of Prometheus Special Agent

Use Case 1: Use most common Prometheus exporters directly integrated in Checkmk



- Automatic processing of Prometheus metrics for specific exporters
- Generation of structured information (hosts, services, metrics)
- Contextual monitoring for Prometheus metrics

Use Case 2: Use custom PromQL queries, e.g. to get data from code instrumentation



- User customizes service with custom PromQL queries
- Special Agent fetches data
- Checkmk converts retrieved data into Checkmk service

Demo

Thank you

tribe29

tribe29 GmbH
Kellerstraße 29
81667 München
Deutschland

Web — tribe29.com
E-Mail — mail@tribe29.com