SIG Instrumentation: Intro

@brancz, @metalmatze & @piosz

Mission

“Covers best practices for cluster observability through **metrics**, **logging**, and **events** across all Kubernetes components and development of relevant components. **Coordinates metric requirements** of different SIGs for other components through finding common APIs.”

Charter

- Core metrics pipeline
- Core logs pipeline
- Instrumenting system components
- Monitoring extensions
- Integration with 3rd party monitoring/logging solutions

Core metrics pipeline

- kubectl top
- Master Metrics API
- Metrics Server
- Kubelet Summary API (with SIG Node)
- cadvisor (with SIG Node)

Core logging pipeline

- kubectl logs
- exposing logs from node (with SIG Node)
- log retention/rotation (with SIG Node)
- handling logs in CRI (with SIG Node)

Monitoring extensions

- kube-state-metrics
- Custom Metrics API
- Custom Metrics API adapters (with monitoring vendors)
- Heapster

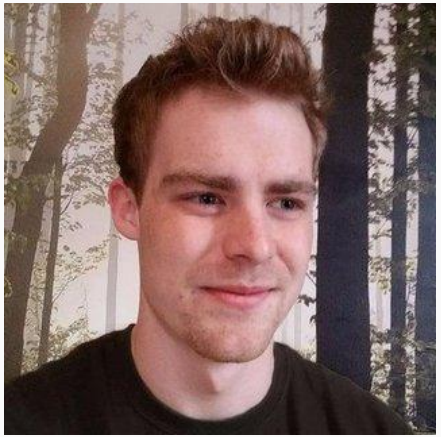
Integration with 3rd party solutions

- best practises and guidelines
- reference integrations
- consuming metadata

Leads

Frederic Branczyk
@brancz

Piotr Szcześniak
piosz@



Fabian Reinartz
@fabxc
Emeritus

Meetings

every second Thursdays at 17:30 UTC

Core pipeline in 2017

- Core Metrics API in beta
- Metrics Server in beta

Monitoring Extensions in 2017

- Custom Metrics API in beta
- Custom Metrics API adapter for Prometheus
- Custom Metrics API adapter for Stackdriver
- kube-state-metrics in GA

2018: stabilization

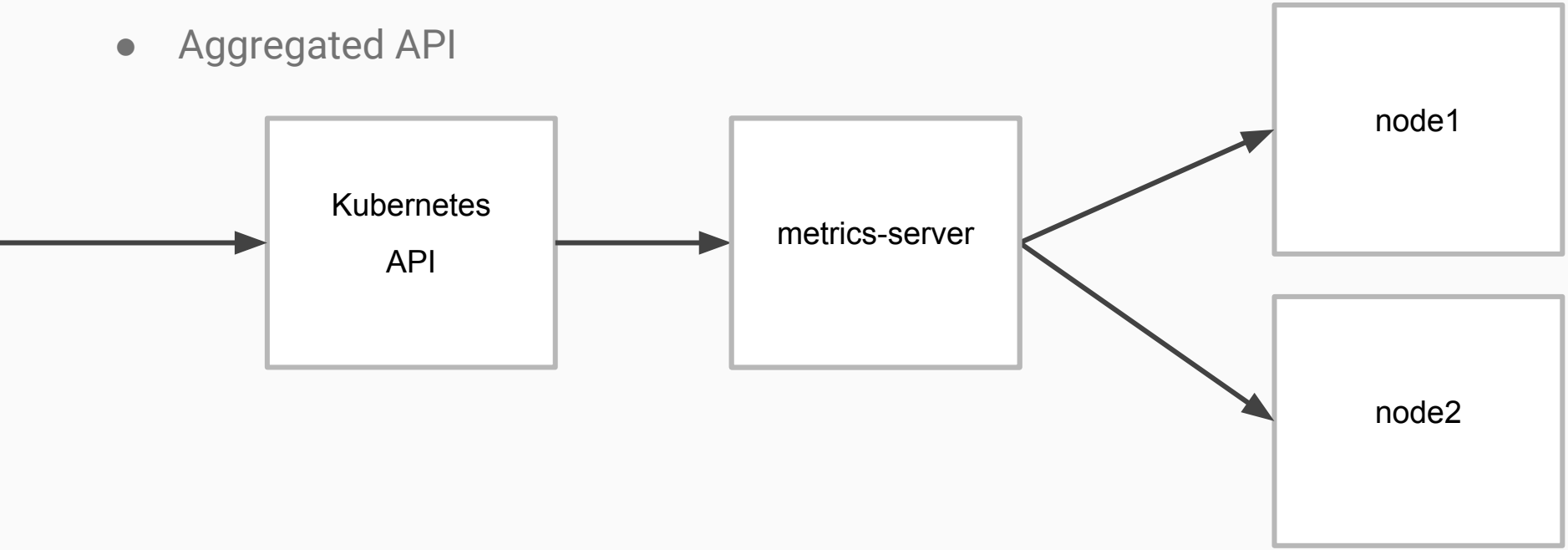
- Core Metrics API to GA
- Custom Metrics API to GA
- graduate Metrics Server from incubator
- deprecate Heapster

2018+: new

- Historical Metrics API
- Standardized way of consuming metadata for 3rd party
- Logs from files?

Resource Metrics API

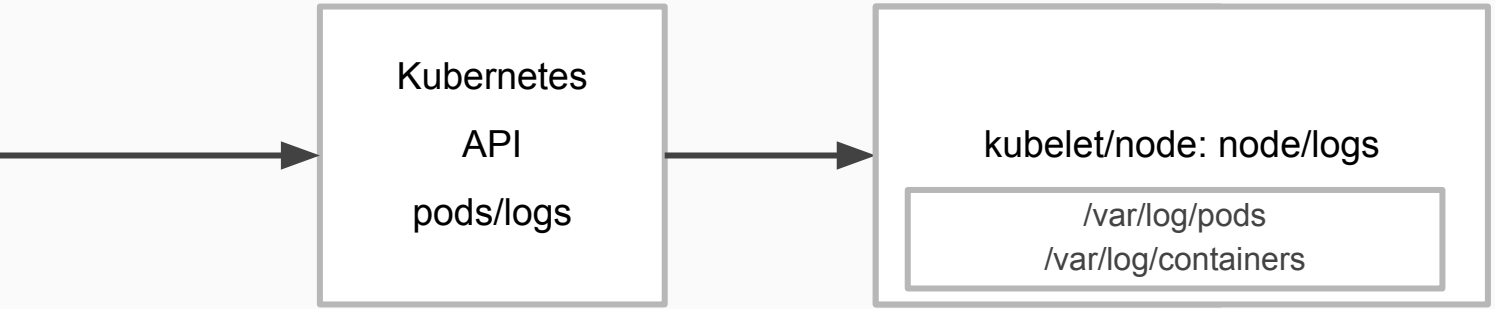
- Aggregated API



kubectl top

- Resource metrics API (v1.10)
  - Pod/Container/Node metrics
  - CPU/Memory
- Pod/Container -> namespaced
- Node -> non-namespaced

kubectl logs



* logs rotated at 10mb by default

Events

- Information about decisions by the scheduler, information about pod, …
- Updated over time: first seen, last seen, description
- Stored in etcd (--etcd-servers-overrides=/events#http://127.0.0.1:4002)

Prometheus data model

- Identified by unique label combination

__name__ = http_requests_total	code = 200	method = GET	1

Prometheus data model

- Counter
- Gauge
- Histogram
- Summary

Prometheus format

```
http_requests_total{code=”200”,methed=”GET”} 12
```

Prometheus clients

- Official:
  - Go, Java or Scala, Python, Ruby
- Community:
  - Bash, C++, Common Lisp, Elixir, Erlang, Haskell, Lua for Nginx, Lua for Tarantool, .NET / C#, Node.js, PHP, Rust

```
registry.MustRegister(requestCounter)
requestCounter.withLabels(“200”, “GET”).Inc()
```

Prometheus endpoint

- /metrics
  - Usually registered by user with a registry
- Text format

Prometheus endpoint

```
/ # wget -O- localhost:10054/metrics
Connecting to localhost:10054 (127.0.0.1:10054)
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 1.265e-05
go_gc_duration_seconds{quantile="0.25"} 1.5234e-05
go_gc_duration_seconds{quantile="0.5"} 1.8863e-05
go_gc_duration_seconds{quantile="0.75"} 2.8004e-05
go_gc_duration_seconds{quantile="1"} 0.002089893
go_gc_duration_seconds_sum 3.588437028
go_gc_duration_seconds_count 158582
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 11
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 4.003968e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 4.8699947548e+11
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.552039e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 6.327640647e+09
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 530432
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 4.003968e+06
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 3.334144e+06
```

Monitoring system components

https://github.com/kubernetes/autoscaler/blob/master/cluster-autoscaler/metrics/metrics.go

```
var (
        /**** Metrics related to cluster state ****/
        clusterSafeToAutoscale = prometheus.NewGauge(
                prometheus.GaugeOpts{
                        Namespace: caNamespace,
                        Name:      "cluster_safe_to_autoscale",
                        Help:      "Whether or not cluster is healthy enough for autoscaling. 1 if it is, 0 otherwise.",
                },
        )

        nodesCount = prometheus.NewGaugeVec(
                prometheus.GaugeOpts{
                        Namespace: caNamespace,
                        Name:      "nodes_count",
                        Help:      "Number of nodes in cluster.",
                }, []string{"state"},
        )

        nodeGroupsCount = prometheus.NewGaugeVec(
                prometheus.GaugeOpts{
                        Namespace: caNamespace,
                        Name:      "node_groups_count",
                        Help:      "Number of node groups managed by CA.",
                }, []string{"node_group_type"},
        )

        unschedulablePodsCount = prometheus.NewGauge(
                prometheus.GaugeOpts{
                        Namespace: caNamespace,
                        Name:      "unschedulable_pods_count",
                        Help:      "Number of unschedulable pods in the cluster.",
                },
        )

        /**** Metrics related to autoscaler execution ****/
        lastActivity = prometheus.NewGaugeVec(
                prometheus.GaugeOpts{
                        Namespace: caNamespace,
                        Name:      "last_activity",
                        Help:      "Last time certain part of CA logic executed."
```

kube-state-metrics

- Creates additional metrics for various objects
- Generates metrics about the Kubernetes' state
- Exposes raw data unmodified from the Kubernetes API
- For almost all objects there are metrics
  - Nodes
  - Pods
  - Deployments
  - DaemonSets
  - ...

Thank you!

Questions?

Demo