KubeCon CloudNativeCon
OPEN SOURCE SUMMIT
China 2023

SIG Instrumentation:
Intro, Deep Dive and Recent Developments

Mengjiao Liu
Shivanshu Raj Shrivastava

About us

Mengjiao Liu@DaoCloud

SIG Instrumentation Reviewer
SIG Docs Reviewer & Chinese localization Owner
Github: mengjiao-liu
Twitter: Mengjiao99

Shivanshu Raj Shrivastava

Contributor: K8S, OpenTelemetry, Istio
Twitter: shivanshu1333
Github: shivanshu1333

Agenda

- Introduction
- SIG Subprojects
- Recent Developments
  - Metrics
  - Logs
  - Traces
  - Demo
- Get involved

What is SIG Instrumentation?

Defined in SIG Instrumentation Charter: Owns best practices for cluster observability through metrics, logging, events, and traces across all k8s components & development of components required for all k8s clusters.

- Scope:
  - Logs
  - Events
  - Metrics
  - Traces

- SubProjects:
  - kube-state-metrics
  - klog
  - metrics-server
  - and more!

How do we do it?

- Triage and fix relevant instrumentation issues
  - All open SIG Instrumentation-labelled issues and pull requests
- Review all code changes for metrics, logs,events and traces
- Develop new features and enhancements
  - Kubernetes Enhancement Proposals (KEPs) for SIG Instrumentation
- Maintain and support subprojects
- Mentor new contributors

Subprojects

Subprojects

- klog
- kube-state-metrics
- metrics-server
- prometheus-adapter
- usage-metrics-collector

klog

Package klog implements logging analogous to the Google-internal C++ INFO/ERROR/V setup. It provides functions Info, Warning, Error, Fatal, plus formatting variants such as Infof. It also provides V-style logging controlled by the -v and -vmodule=file=2 flags.

The kubernetes/kubernetes repo uses klog as the logging library

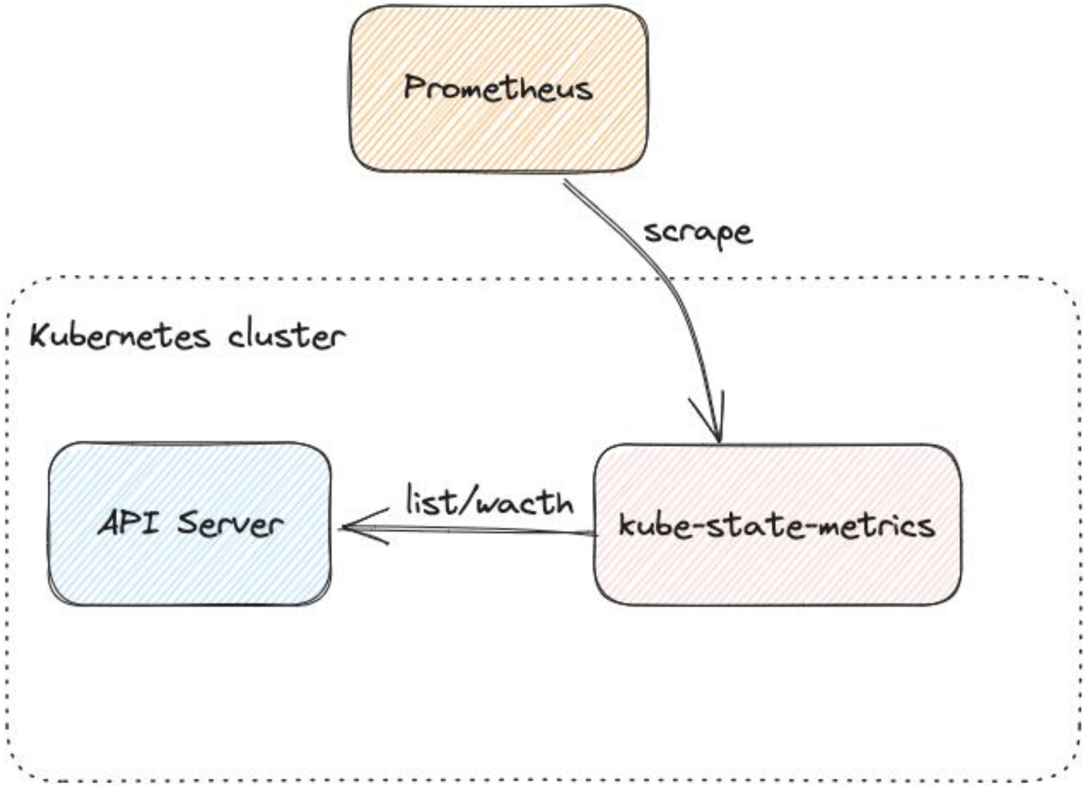
- ➔ Supports structured logging（include text and json format）
- ➔ Supports contextual logging

🎉 **Release**: v2.100.1

kube-state-metrics

kube-state-metrics generates Prometheus format metrics based on the current state of the Kubernetes native resources

- Deployments, Pods, Services, StatefulSets etc. A full list of resources is available in the documentation of kube-state-metrics.



Examples:

kube_deployment_status_condition{namespace="kube-system",deployment="coredns",condition="Available",status="true"} 1

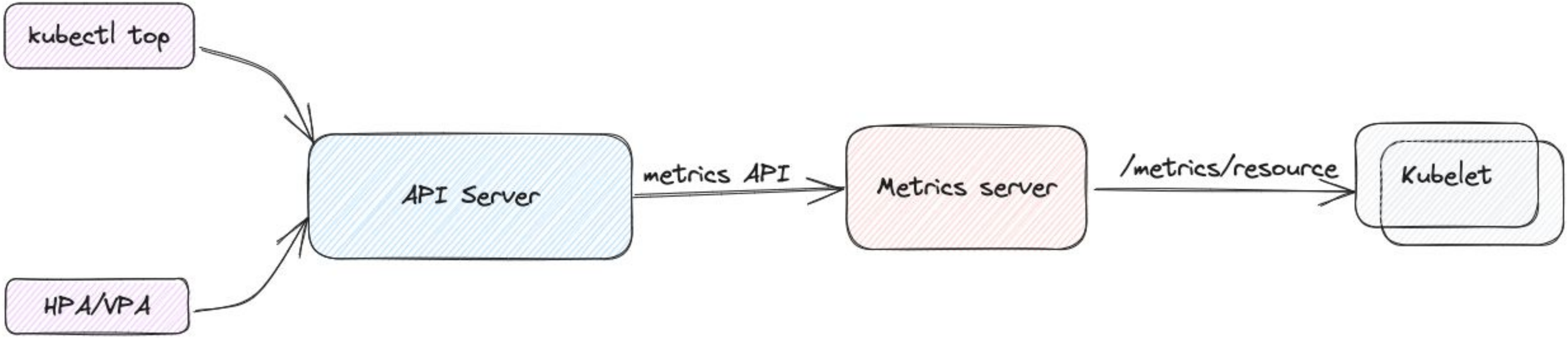
kube_pod_status_ready{namespace="default",pod="nginx",uid="2d3795ca-e53d-4bd3-860f-ced3e74a3db9",condition="true"} 1

🎊 Release: v2.10.0

metrics-server

Metrics Server is a scalable, efficient source of container resource metrics for Kubernetes built-in autoscaling pipelines.

- offers a basic set of metrics to support automatic scaling and similar use cases
- view these metrics using the `kubectl top` command

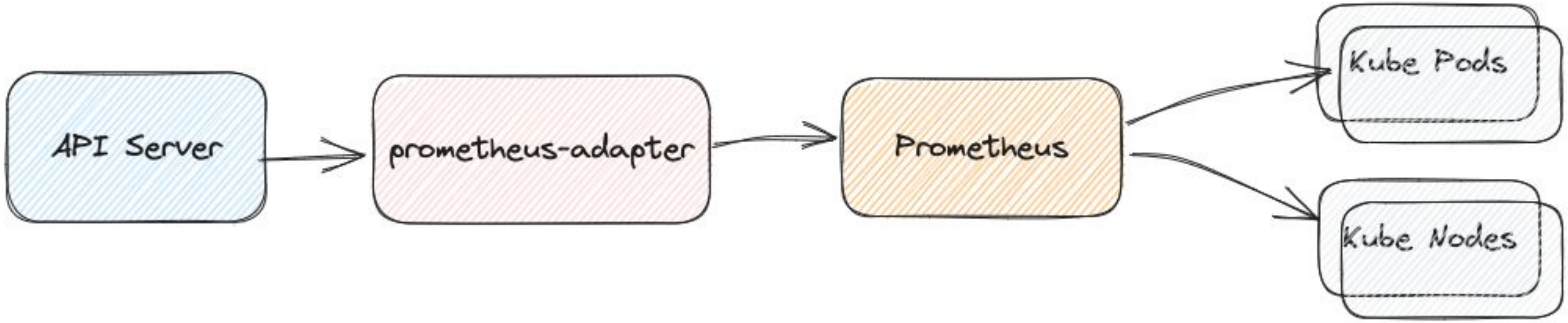


🎊 Release: v0.6.4

prometheus-adapter

Prometheus Adapter Implementation of the Kubernetes Custom, Resource and External Metric APIs

- suitable for use with the autoscaling/v2 Horizontal Pod Autoscaler
- replace the metrics server on clusters that already run Prometheus



🎉 **Release:** v0.11.1

usage-metrics-collector

- Repo:https://github.com/kubernetes-sigs/usage-metrics-collector
- Prometheus metrics collector optimized for collecting kube usage and capacity metrics.
    - Simple
        - Don’t require writing complex promql statements to combine metrics – native support for joining metadata across resources
    - Scalable
        - Support large clusters with lots of Pods
        - Optimize cardinality and storage requirements by exporting pre-aggregated results
    - Performant
        - Quickly compute values without requiring tiering prometheus instances

Example Collector Configurations

Get p95 utilization (cpu and memory) using 1 second sampling intervals for all containers in each workload.

```
resources:
  "cpu": "cpu_cores" # get cpu metrics
  "memory": "memory_bytes" # get memory metrics
aggregations:
- sources:
    type: "container"
    container: [ "utilization" ] # export container utilization
  levels:
  - mask:
      name: "container"
      builtIn: # aggregate on these labels
        exported_container: true
        exported_namespace: true
        workload_name: true
        workload_kind: true
        workload_api_group: true
        workload_api_version: true
      operation: "p95" # take the 95th percentile sample
```

Result Metrics:

```
workload_p95_utilization_cpu_cores{exported_container="",exported_namespace="",workload_name="",
workload_kind="",workload_api_group="",workload_api_version=""}

workload_p95_utilization_memory_bytes{exported_container="",exported_namespace="",workload_name="",
workload_kind="",workload_api_group="",workload_api_version=""}
```

Recent Developments

Metrics

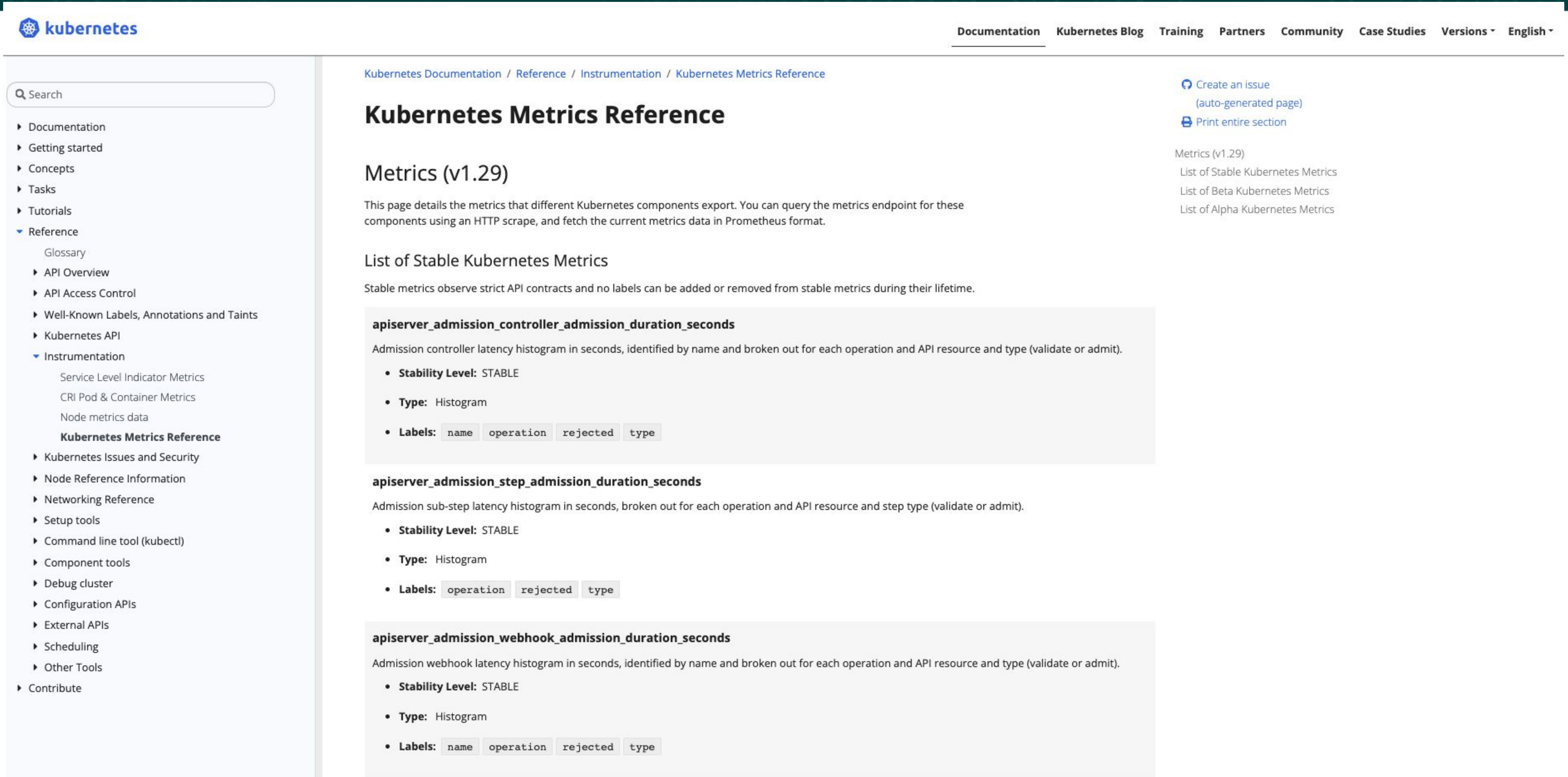
KEP-3498: Extend metrics stability

- Enhancement status
  - Alpha: 1.26
  - Beta: 1.27
  - Stable: 1.28
- Extend on the existing metrics stability framework to achieve stability guarantees
- Document all metrics exported by the core distribution of Kubernetes

KEP-3498: Extend metrics stability

Stability Levels:

- **Internal**: does not have any stability guarantees
- **Alpha**: does not have any stability guarantees
  - included in metric auto-documentation
- **Beta**: has *some* stability guarantees
  - deprecate metrics need a release or more
  - forward compatible,labels can *only be added*, *and not removed*
  - included in metric auto-documentation
- **Stable**: has stability guarantees
  - follow a deprecation policy(12 months or 3 releases)
  - **not change** in respect to labels
  - included in metric auto-documentation



Kubernetes Metrics Reference

Metrics (v1.29)

This page details the metrics that different Kubernetes components export. You can query the metrics endpoint for these components using an HTTP scrape, and fetch the current metrics data in Prometheus format.

List of Stable Kubernetes Metrics

Stable metrics observe strict API contracts and no labels can be added or removed from stable metrics during their lifetime.

apiserver_admission_controller_admission_duration_seconds

Admission controller latency histogram in seconds, identified by name and broken out for each operation and API resource and type (validate or admit).

- **Stability Level:** STABLE
- **Type:** Histogram
- **Labels:** `name` `operation` `rejected` `type`

apiserver_admission_step_admission_duration_seconds

Admission sub-step latency histogram in seconds, broken out for each operation and API resource and step type (validate or admit).

- **Stability Level:** STABLE
- **Type:** Histogram
- **Labels:** `operation` `rejected` `type`

apiserver_admission_webhook_admission_duration_seconds

Admission webhook latency histogram in seconds, identified by name and broken out for each operation and API resource and type (validate or admit).

- **Stability Level:** STABLE
- **Type:** Histogram
- **Labels:** `name` `operation` `rejected` `type`

Logs

Structured Logging

- Structured Logging
  - Future of observability in Kubernetes
  - Goals/Non Goals
  - Implementation details
  - Migration details

Structured Logging

- GA in v1.27; introduced as Alpha in v1.19
- Migration
  - Most components have been migrated.
  - The remaining part is to convert code directly to contextual logging.

Structured Logging

Goals

- Make most common logs more queryable by standardizing log message and references to Kubernetes objects (Pods, Nodes etc.)
- Enforce log structure by introduction of new klog methods that could be used to generate structured logs.
- Simplify ingestion of logs into third party logging solutions by adding an option to output logs in the JSON format

Non Goals

- We are not replacing currently used logging library (klog) or the way in which it is used

Structured Logging

- Implementation details
  - Log message structure
  - References to Kubernetes objects
  - Introduce JSON output format in klog
  - Logging configuration
  - Performance
- Migration details

Structured Logging

- Log message structure
  - There could be multiple ways of standardising the logging structure in Kubernetes, the one we agreed to implement in the KEP is to have following logging structure

    ```
    <message> <key1>=<value1> <key2>=<value2> …
    ```

    e.g.

    ```
    pod := corev1.Pod{Name: "kubedns", Namespace: "kube-system", ...}
    klog.InfoS("Pod status updated", "pod", klog.KObj(pod), "status",
    "ready")
    ```

Structured Logging

- References to Kubernetes objects:
  - The idea is to use k8s api first approach to get k8s objects and embed the object related information into the logs
  - Correlate between different kubernetes objects

```
func KObj(obj ObjectMeta) ObjectRef
func KRef(namespace, name string) ObjectRef


type ObjectRef struct {
  Name      string `json:"name"`
  Namespace string `json:"namespace,omitempty"`
}
```

Structured Logging

- References to Kubernetes objects:

  Namespaced objects: <namespace>/<name>
  e.g. kube-system/kubedns

  Non-namespaced objects: <name>
  e.g. node cluster1-vm-72x33b8p-34jz

  e.g.
  klog.InfoS("Pod status updated", "pod", klog.KObj(pod), "status", "ready")

  .

  .

  klog.ErrorS(err, "Failed to update pod status", "pod", klog.KObj(pod))

Structured Logging

- Introduce JSON output format in klog:
  - Introduction of new methods to klog library to support JSON.
  - With klog v2 we can take further advantage of this fact and add an option to produce structured logs in JSON format.

- Some pros of using JSON:
  - Broadly adopted by logging libraries with very efficient implementations (zap, zerolog).
  - Out of the box support by many logging backends (Elasticsearch, Stackdriver, BigQuery, Splunk, Open Telemetry)
  - Easily parsable and transformable
  - Existing tools for ad-hoc analysis (jq)

Structured Logging

- Introduce JSON output format in klog:

```
klog.InfoS("Pod status updated", "pod", klog.KObj(pod), "status", "ready")
```

```
{
  "ts": 1580306777.04728,
  "v": 4,
  "msg": "Pod status updated",
  "pod":{
      "name": "nginx-1",
      "namespace": "default"
  },
  "status": "ready"
}
```

Structured Logging

- Introduce JSON output format in klog:

```
type Request struct {
  Method  string
  Timeout int
  secret  string
  Con  *Connection
}
req := Request{Method: "GET", Timeout: 30, secret: "pony"}
klog.InfoS("Request finished", "request", Request)
```

Structured Logging

- Introduce JSON output format in klog:

```
{
  "ts": 1580306777.04728,
  "v": 4,
  "msg": "Request finished",
  "request":{
    "Method": "GET",
    "Timeout": 30
  }
}
```

Structured Logging

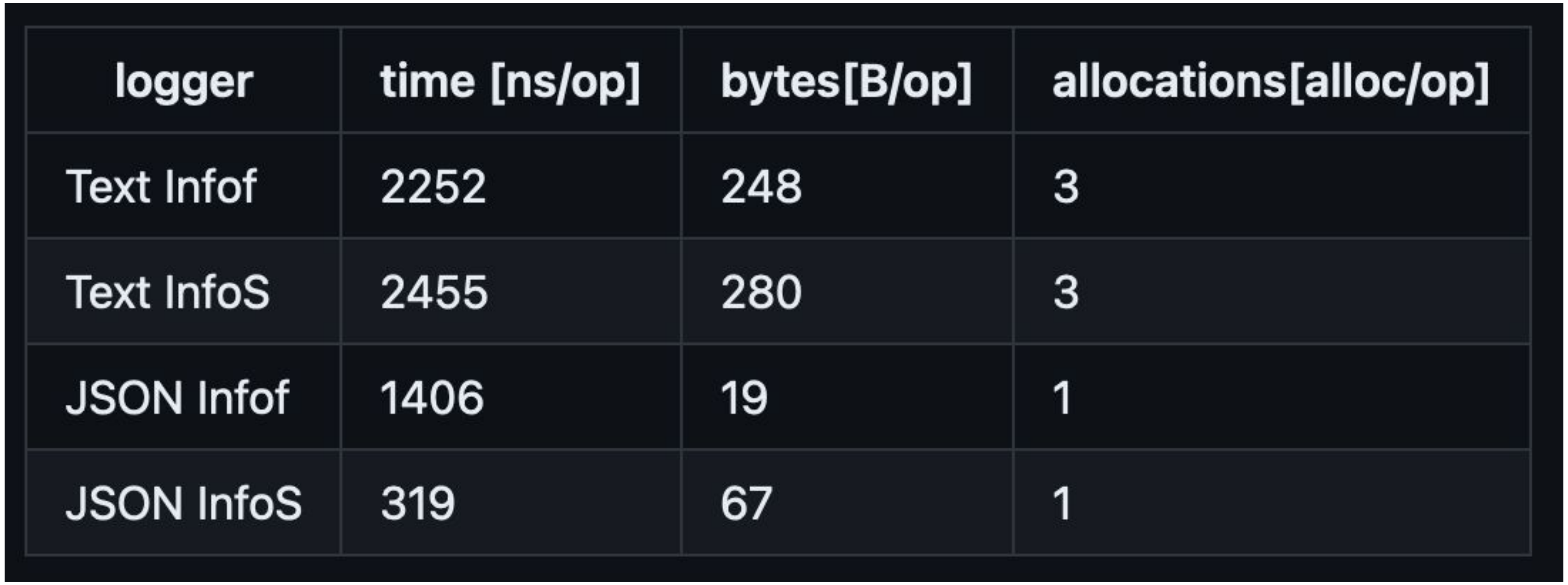
- Logging configuration
  - Implementation of LoggingConfig structure as part of k8s.io/component-base

    introduced flag **--logging-format** values:

    a) text: for text-based logging format (default)
    b) json: for new JSON format

Structured Logging

- Performance
    - Logging performance with the new implementation. Performance is wrt to log volume and the performance impact.



logger	time [ns/op]	bytes[B/op]	allocations[alloc/op]
Text Infof	2252	248	3
Text InfoS	2455	280	3
JSON Infof	1406	19	1
JSON InfoS	319	67	1

    - InfoS implementation for text is 9% slower than Infof.
    - Kubernetes performance as logging takes less than 2% of overall CPU usage.

Contextual Logging

- Beta in v1.28; introduced as Alpha in v1.24
- Migration
    - kube-controller-manager has been converted
    - kube-scheduler has converted most of it
    - Other components are being converted

Contextual Logging

- Contextual Logging
    - Goals/Non Goals
    - Implementation details
    - Migration details

Contextual Logging

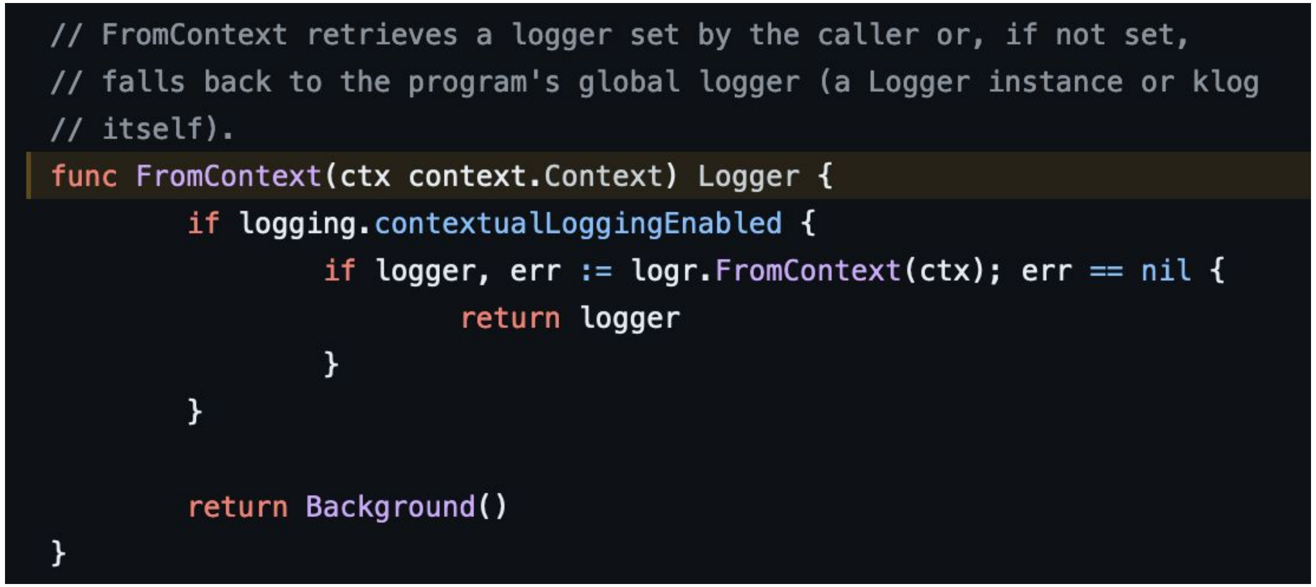
- Contextual Logging
  - Goals
    - Grant the caller of a function control over logging inside that function, either by passing a logger into the function or by configuring the object that a method belongs to.
    - Provide documentation and helper code for setting up logging in unit tests.
    - Change as few exported APIs as possible
  - Non-Goals
    - Remove the klog text output format
    - Deprecate klog

Contextual Logging

- Removing the dependency on the global klog logger
  - klog.ErrorS -> logger.Error
  - logger is a logr.Logger instance. klog.Logger is an alias for that type

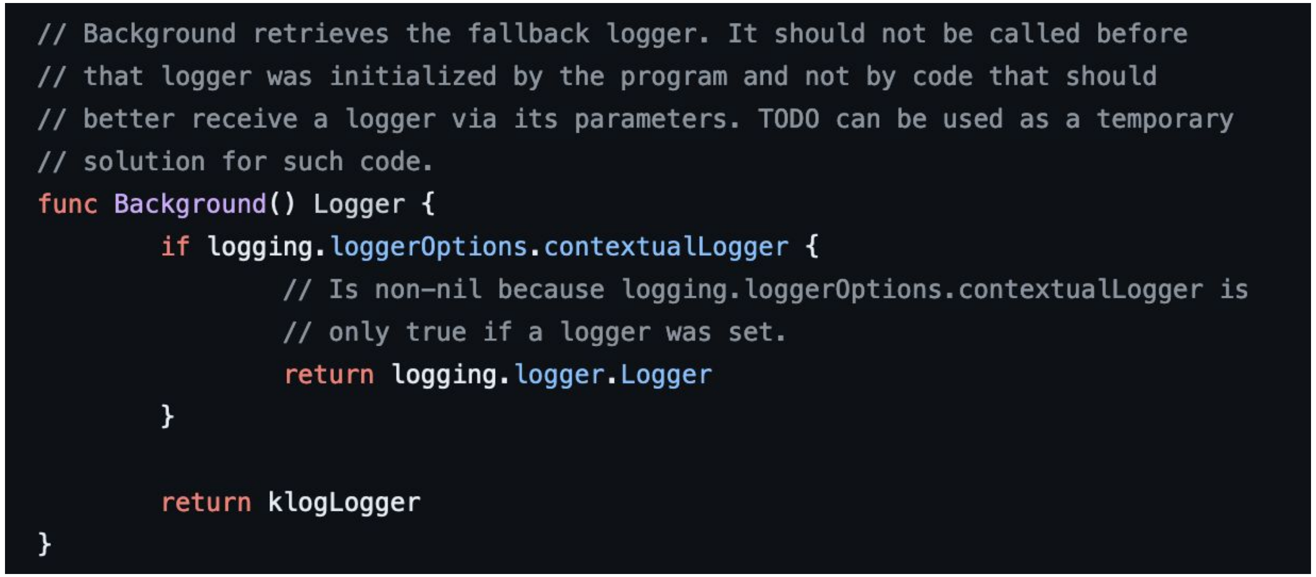
- Extend klog for contextual logging
  - Several new klog functions help with that:
    - klog.FromContext
    - klog.Background
    - klog.TODO

Contextual Logging



```go
// FromContext retrieves a logger set by the caller or, if not set,
// falls back to the program's global logger (a Logger instance or klog
// itself).
func FromContext(ctx context.Context) Logger {
        if logging.contextualLoggingEnabled {
                if logger, err := logr.FromContext(ctx); err == nil {
                        return logger
                }
        }

        return Background()
}
```



```go
// Background retrieves the fallback logger. It should not be called before
// that logger was initialized by the program and not by code that should
// better receive a logger via its parameters. TODO can be used as a temporary
// solution for such code.
func Background() Logger {
        if logging.loggerOptions.contextualLogger {
                // Is non-nil because logging.loggerOptions.contextualLogger is
                // only true if a logger was set.
                return logging.logger.Logger
        }

        return klogLogger
}
```

Contextual Logging

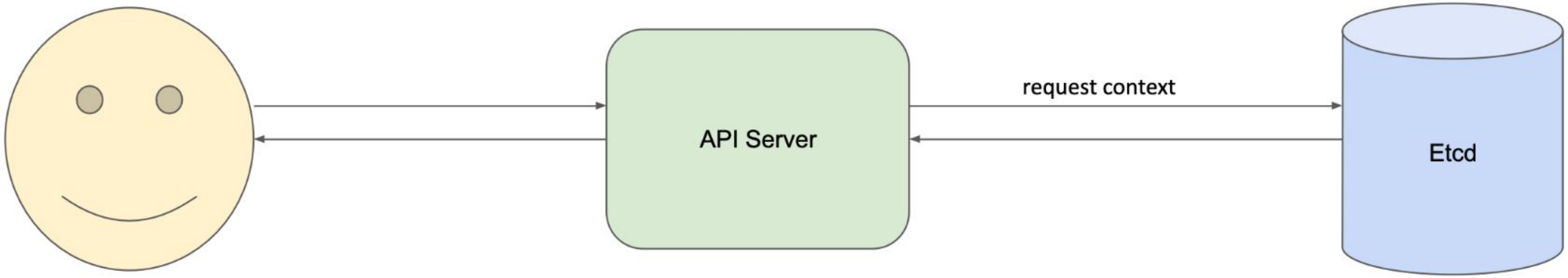
- Extend klog for contextual logging
    - Use migrated structured logs and attach context with it
        - With the helper functions that we added in klog, structured logs can also be transformed to more fine grained contextual logs

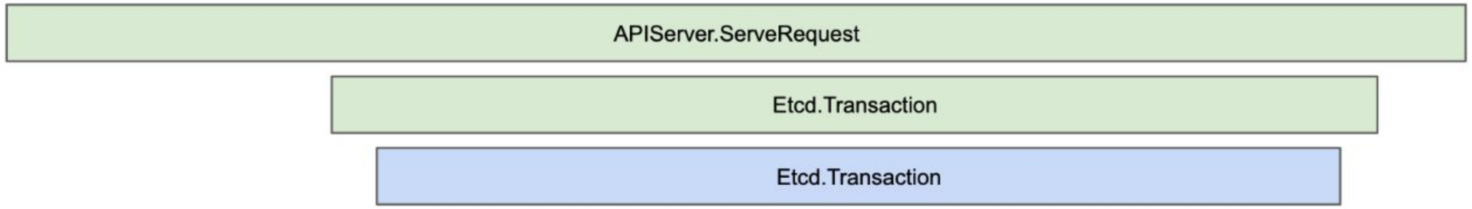
Traces

Tracing

Tracing in Kubernetes

OpenTelemetry





Tracing

API Server Tracing:

- Beta in 1.27
- Trace requests from the API Server to Etcd.

Kubelet Tracing

- Beta in 1.27
- Trace requests from the Kubelet to the Container Runtime

OpenTelemetry dependency updated to 1.0+ in K8s 1.26

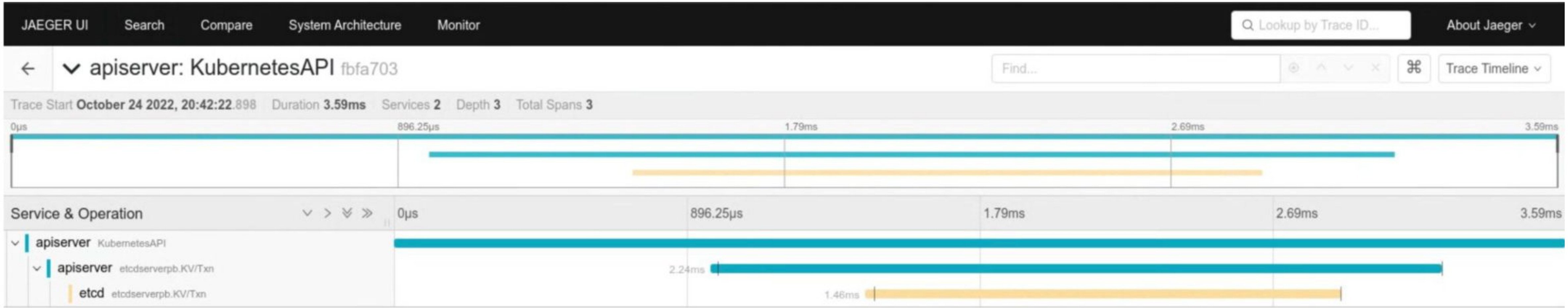
Tracing

Log-Based “Tracing”

```
Trace[1395114870]: "Create" url:/api/v1/nodes,user-agent:tracing.test/v0.0.0 (linux/amd64)
kubernetes/$Format,audit-id:c0282104-8068-44b1-a088-756b1253326d,client:127.0.0.1,accept:application/vnd.kubernetes.protobuf, */*,protocol:HTTP/2.0
(28-Oct-2022 13:42:35.876) (total time: 2ms):
Trace[1395114870]: ---"limitedReadBody succeeded" len:86 0ms (13:42:35.876)
Trace[1395114870]: ---"About to convert to expected version" 0ms (13:42:35.876)
Trace[1395114870]: ---"Conversion done" 0ms (13:42:35.876)
Trace[1395114870]: ---"About to store object in database" 0ms (13:42:35.876)
Trace[1395114870]: ["Create etcd3"
audit-id:c0282104-8068-44b1-a088-756b1253326d,key:/minions/fake,type:*core.Node,resource:nodes 2ms (13:42:35.876)
Trace[1395114870]: ---"About to Encode" 0ms (13:42:35.876)
Trace[1395114870]: ---"Encode succeeded" len:177 0ms (13:42:35.876)
Trace[1395114870]: ---"TransformToStorage succeeded" 0ms (13:42:35.876)
Trace[1395114870]: ---"Txn call succeeded" 1ms (13:42:35.878)
Trace[1395114870]: ---"decode succeeded" len:177 0ms (13:42:35.878)]
Trace[1395114870]: ---"Write to database call succeeded" len:86 0ms (13:42:35.878)
Trace[1395114870]: ---"About to write a response" 0ms (13:42:35.878)
Trace[1395114870]: ---"Writing http response done" 0ms (13:42:35.878)
Trace[1395114870]: [2.771143ms] [2.771143ms] END
```

Tracing

Kubernetes 1.22

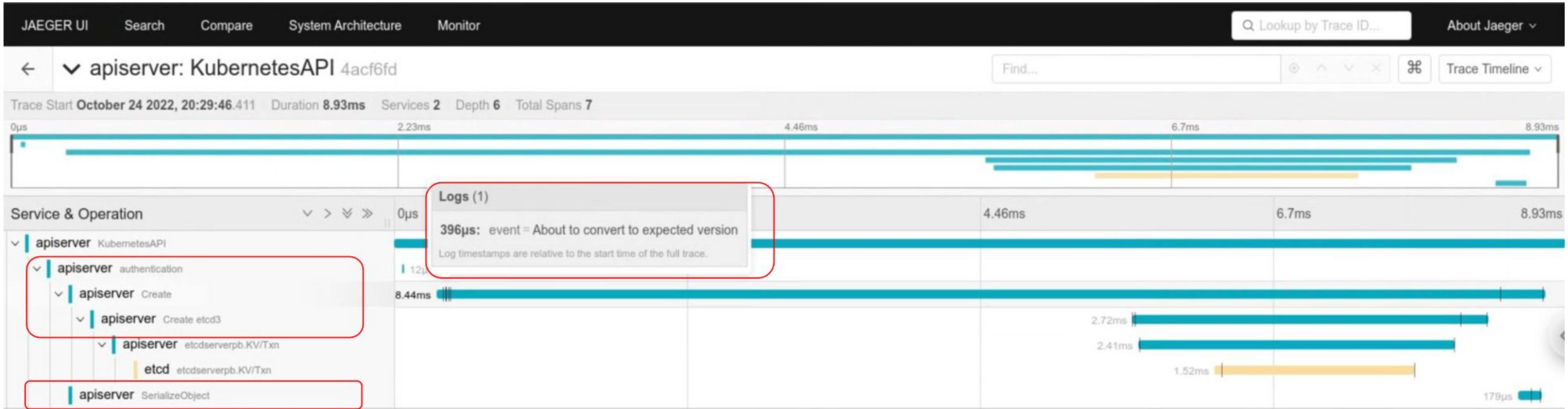


apiserver: KubernetesAPI fbfa703

Trace Start October 24 2022, 20:42:22.898 Duration 3.59ms Services 2 Depth 3 Total Spans 3

Service & Operation	
apiserver KubernetesAPI	
apiserver etcdserverpb.KV/Txn	2.24ms
etcd etcdserverpb.KV/Txn	1.46ms

Kubernetes 1.26



apiserver: KubernetesAPI 4acf6fd

Trace Start October 24 2022, 20:29:46.411 Duration 8.93ms Services 2 Depth 6 Total Spans 7

Logs (1)

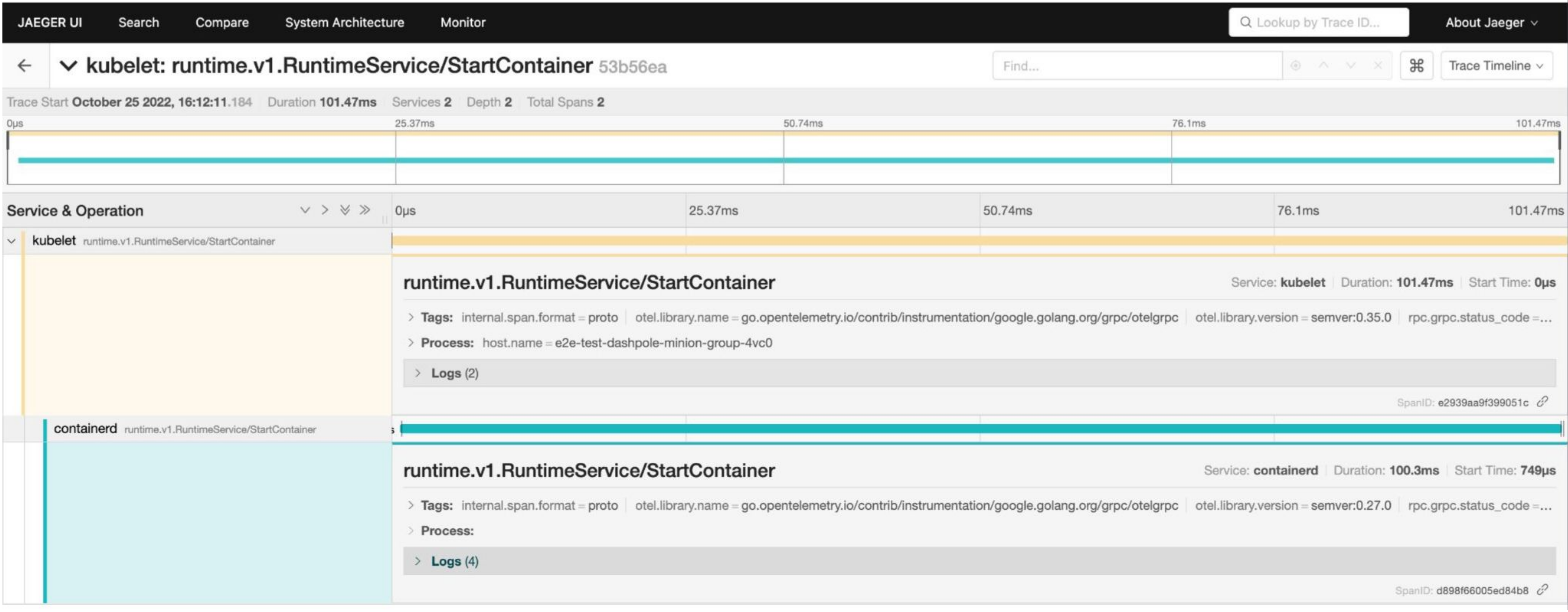
396μs: event = About to convert to expected version

Log timestamps are relative to the start time of the full trace.

Service & Operation	
apiserver KubernetesAPI	
apiserver authentication	12μ
apiserver Create	8.44ms
apiserver Create etcd3	2.72ms
apiserver etcdserverpb.KV/Txn	2.41ms
etcd etcdserverpb.KV/Txn	1.52ms
apiserver SerializeObject	179μs

Tracing

Alpha: CRI Traces



kubelet: runtime.v1.RuntimeService/StartContainer 53b56ea

Trace Start **October 25 2022, 16:12:11**.184 Duration **101.47ms** Services **2** Depth **2** Total Spans **2**

kubelet runtime.v1.RuntimeService/StartContainer

runtime.v1.RuntimeService/StartContainer

Service: **kubelet** | Duration: **101.47ms** | Start Time: **0μs**

> **Tags:** internal.span.format = proto | otel.library.name = go.opentelemetry.io/contrib/instrumentation/google.golang.org/grpc/otelgrpc | otel.library.version = semver:0.35.0 | rpc.grpc.status_code = ...

> **Process:** host.name = e2e-test-dashpole-minion-group-4vc0

> **Logs** (2)

SpanID: e2939aa9f399051c

containerd runtime.v1.RuntimeService/StartContainer

runtime.v1.RuntimeService/StartContainer

Service: **containerd** | Duration: **100.3ms** | Start Time: **749μs**

> **Tags:** internal.span.format = proto | otel.library.name = go.opentelemetry.io/contrib/instrumentation/google.golang.org/grpc/otelgrpc | otel.library.version = semver:0.27.0 | rpc.grpc.status_code = ...

> **Process:**

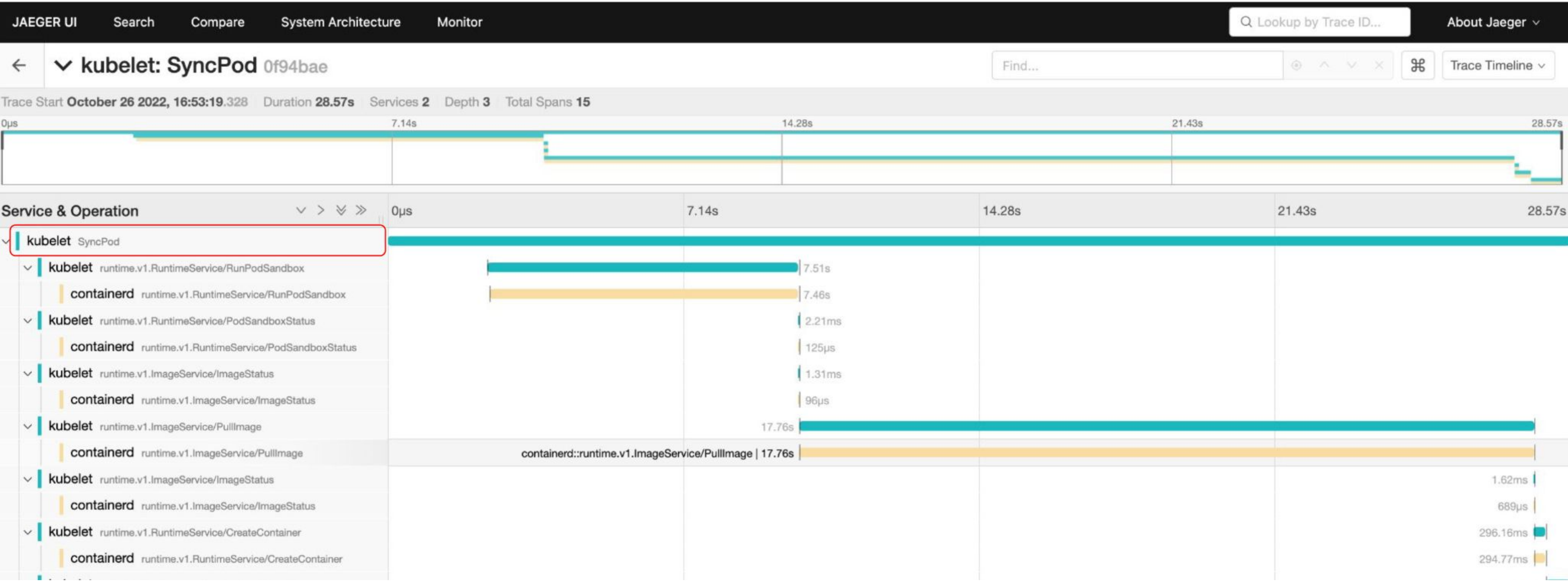
> **Logs** (4)

SpanID: d898f66005ed84b8

Tracing

KubeCon CloudNativeCon OPEN SOURCE SUMMIT China 2023

Proof of Concept: Complete Pod traces



kubelet: SyncPod 0f94bae

Trace Start October 26 2022, 16:53:19.328 Duration 28.57s Services 2 Depth 3 Total Spans 15

Service	Operation	Duration
kubelet	SyncPod	
kubelet	runtime.v1.RuntimeService/RunPodSandbox	7.51s
containerd	runtime.v1.RuntimeService/RunPodSandbox	7.46s
kubelet	runtime.v1.RuntimeService/PodSandboxStatus	2.21ms
containerd	runtime.v1.RuntimeService/PodSandboxStatus	125μs
kubelet	runtime.v1.ImageService/ImageStatus	1.31ms
containerd	runtime.v1.ImageService/ImageStatus	96μs
kubelet	runtime.v1.ImageService/PullImage	17.76s
containerd	runtime.v1.ImageService/PullImage	containerd::runtime.v1.ImageService/PullImage | 17.76s
kubelet	runtime.v1.ImageService/ImageStatus	1.62ms
containerd	runtime.v1.ImageService/ImageStatus	689μs
kubelet	runtime.v1.RuntimeService/CreateContainer	296.16ms
containerd	runtime.v1.RuntimeService/CreateContainer	294.77ms

Tracing

Future Plans:

- Add kubelet spans to track “create pod” instead of just “create container”
- Link from Metrics to Traces with Prometheus Exemplars
- Link from Logs to Traces with Trace + Span IDs in Logs

Demo

Get involved!

Get involved

- Join #sig-instrumentation on Slack
- Attend our SIG meetings
  - Regular SIG Meeting: Thursdays at 9:30 PT (Pacific Time) (biweekly).
  - Regular Triage Meeting: Thursdays at 9:30 PT (Pacific Time) (biweekly - alternating with regular meeting)
- Subscribe mailing list: kubernetes-sig-instrumentation
- Participate in reviews, issues, and docs!
- Kubernetes Enhancement Proposals (KEPs)

🎉 **We are seeking more contributors! You are welcome to join us !**

KubeCon | CloudNativeCon

OPEN SOURCE SUMMIT

China 2023