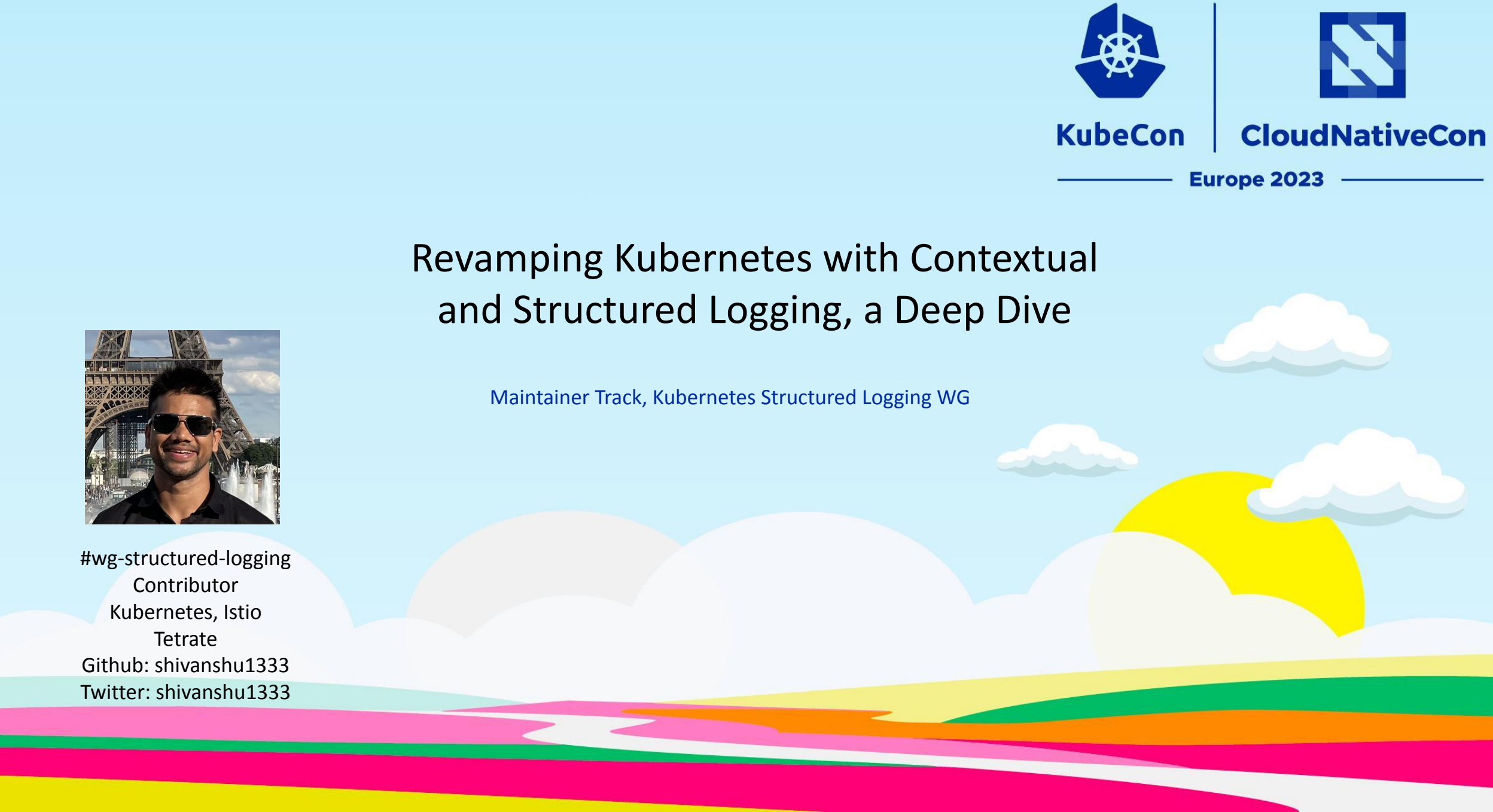

KubeCon
CloudNativeCon
Europe 2023
TiKV

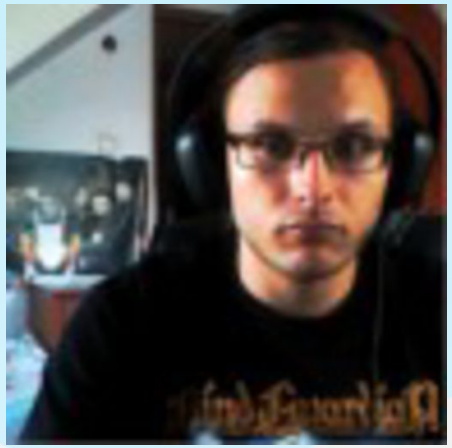


Revamping Kubernetes with Contextual and Structured Logging, a Deep Dive

Maintainer Track, Kubernetes Structured Logging WG

#wg-structured-logging
Contributor
Kubernetes, Istio
Tetrate
Github: shivanshu1333
Twitter: shivanshu1333

Shout out to 🥳



Marek Siarkowicz
#wg-structured-logging
lead
Github: serathius
Twitter: serathius

Patrick Ohly
#wg-structured-logging
lead
Github: pohly

Agenda

- Introduction (~10 mins)
- Deep Dive (~15 mins)
- Migration instructions (~5 mins)
- Questions (~5 mins)

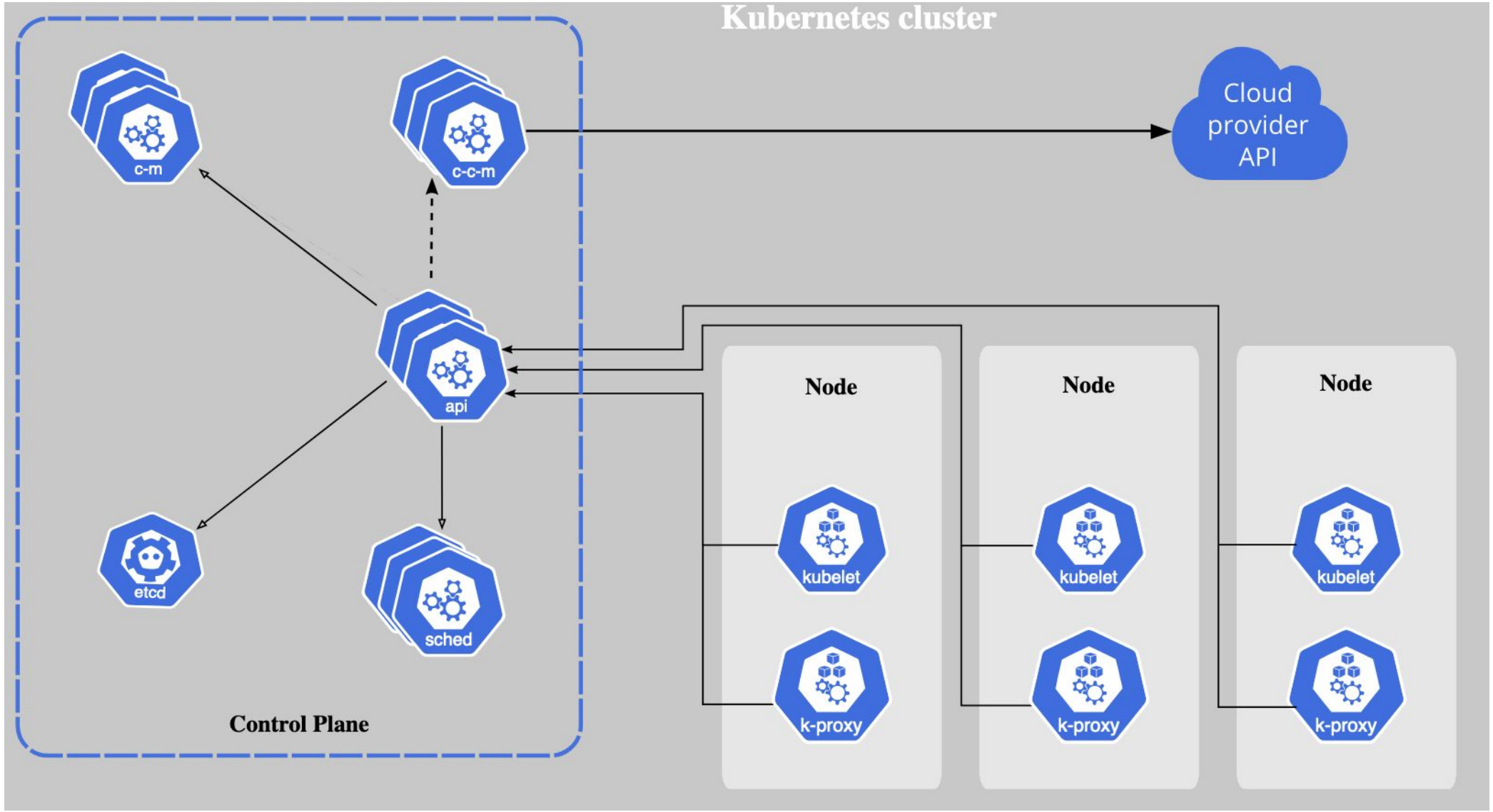
Audience

- Kubernetes core contributors
- Contributors to logging agents (fluentd, opentelemtry etc)
- End users of Kubernetes
- New contributors :)

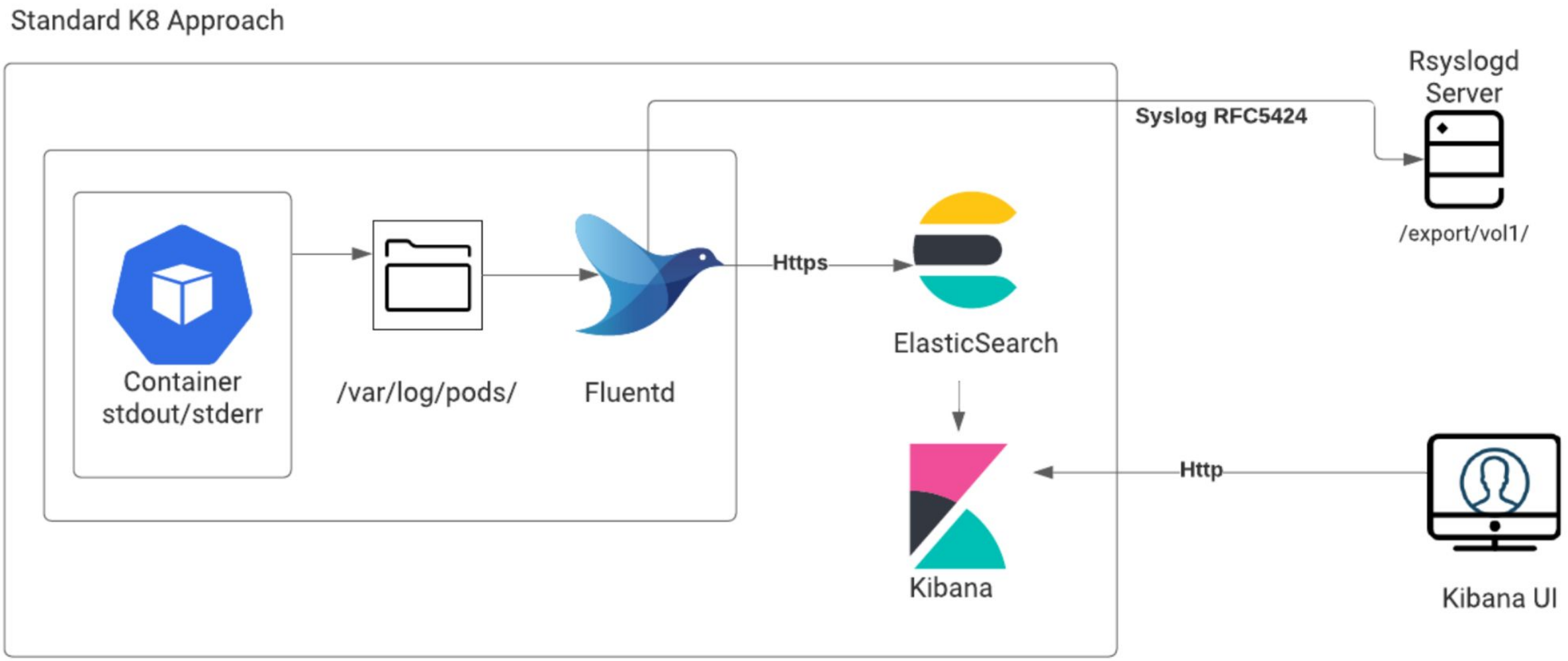
Introduction

- Motivation
  - Kubernetes logs are messy :|
  - Kubernetes uses klog as its default logger (fork of glog)
  - Easy and standardized log collection
  - Fine grained information of Kubernetes components
  - Easy and automated monitoring

Introduction



Introduction



Introduction

- Structured Logging
  - Introduction
    - Proposal
    - Goals/Non Goals
  - Deep dive
    - Implementation details
    - Migration details

Introduction

- Proposal
  - Define standard structure for Kubernetes log messages
  - Add methods to klog to enforce this structure
  - Add ability to configure Kubernetes components to produce logs in JSON format
  - Initiate migration to structured logging

eg:

I1025 00:15:15.525108 1 controller_utils.go:116] "Pod status updated" pod="kube-system/kubedns" status="ready"

Introduction

- Goals/Non Goals
  - 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

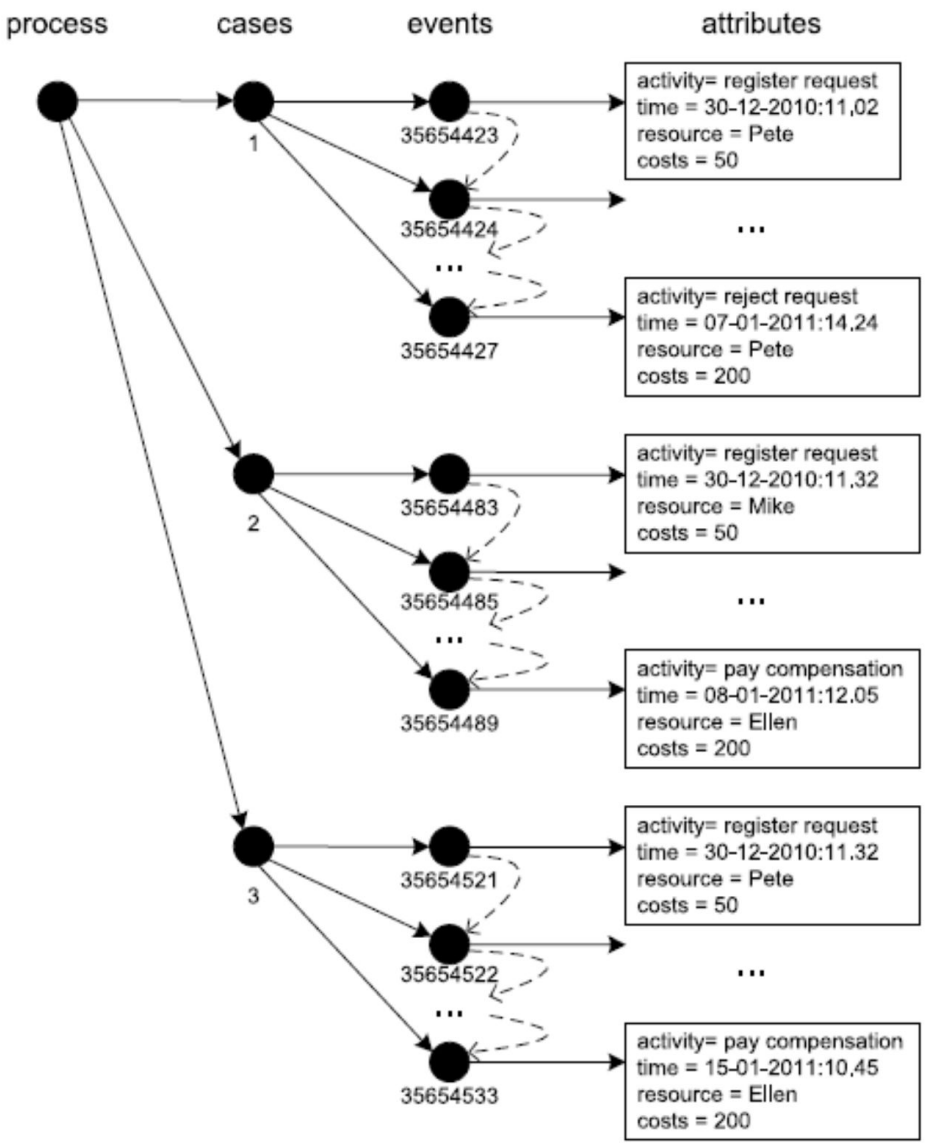
Introduction

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

Introduction

- Contextual Logging
  - Introduction
    - Proposal
    - Goals/Non Goals
  - Deep dive
    - Implementation details
    - Migration details

Introduction



Introduction

- Proposal
  - Replaces the global logger by passing a logr.Logger instance into functions via a context.
  - Adding extended support for klog to support contextual logging (i.e. adding support to use go-logr/logr apis in klog)

eg:

```
I0404 18:00:02.916429  451895 logger.go:94] "example/myname: runtime" foo="bar" duration="1m0s"
```

Introduction

- Goals/Non Goals
  - 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

Introduction

- Goals/Non Goals
  - Non-Goals
    - Remove the klog text output format
    - Deprecate klog

Deep Dive

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

Deep Dive (Implementation details)

- 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")
    ```

Deep Dive (Implementation details)

- 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"`
}
```

Deep Dive (Implementation details)

- 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))

Deep Dive (Implementation details)

- 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)

Deep Dive (Implementation details)

- 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"
}
```

Deep Dive (Implementation details)

- 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)
```

Deep Dive (Implementation details)

- Introduce JSON output format in klog:

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

Deep Dive (Implementation details)

- 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

Deep Dive (Implementation details)

- 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.

Deep Dive (Implementation details)

- Contextual Logging
  - Deep dive
    - User stories:
      - *"kube-scheduler developer Joan wants to know which pod and which operation and scheduler plugin log messages are associated with"*
      - *"wants to increase the verbosity of the scheduler while it processes a certain pod ("per-flow additional log")"*
      - *…many more*
    - Implementation details
      - Removing the dependency on the global klog logger
      - Extend klog for contextual logging
      - Use migrated structured logs and attach context with it
    - Migration details

Deep Dive (Implementation details)

- 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

Deep Dive (Implementation details)

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

```
// 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()
}
```

Deep Dive (Implementation details)

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

```
// 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
}
```

Deep Dive (Implementation details)

- 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

Migration Details

- Structured and Contextual Logging migration instructions

We Need Your Help!!

- From SIGs and Working Groups
  - Consider a big volume of PRs, we request you to PTAL :)
  - Adopt the new practice for latest PRs
  - LGTM on us, Approval is on you (mostly)

- From New contributors
  - Join #wg-structured-logging channel on slack, Join the mailing list
  - Regular Meeting: Thursdays at 15:30 London-UK (biweekly)
  - Meeting notes and Agenda, Meeting recordings.
  - Pick up migration issues (label wg-structured-logging)
  - Please ask questions!!


KubeCon
CloudNativeCon
Europe 2023
That's it from my side please ask your
Questions
I'll try to answer them :)
Thank you!!