Kubernetes meetup Tokyo#39

FUJITSU
shaping tomorrow with you

Kubernetesコミュニティへ提案中のKEP「ロギング強化機能」についての紹介

2021/02/25

富士通株式会社
小林 大介

自己紹介

FUJITSU

- **名前・所属**
  - 小林大介
  - 富士通株式会社

- **やってきたこと・やっていること**
  - OSSコミュニティで開発者として活動
  - 2019年からK8sの開発、CKA取得やコミュニティへのコントリビュート
  - ログ強化機能の開発

アジェンダ

FUJITSU

- **提案中の機能の概要**
  - モチベーション・既存の問題点
  - 提案している解決策
- **機能の実装について**
  - 機能実現に必要な要素
  - 機能の実装方針
  - 実装に関する現在の課題
- **コミュニティ活動で苦労した点**

過去の発表とのつながり

FUJITSU

- 前回（Kubefest）の発表資料はこちら
  （https://www.slideshare.net/JinHase/kube-fest-tokyolog）
- 今回は、前回からどのように変化したか、差分を主に話します
- 前回話した内容は、概要の部分でサマリーを話します

アジェンダ

- **提案中の機能の概要**
  - モチベーション・既存の問題点
  - 提案している解決策
- **機能の実装について**
  - 機能実現に必要な要素
  - 機能の実装方針
  - 実装に関する現在の課題
- **コミュニティ活動で苦労した点**

Kubernetesにおけるロギング機能

FUJITSU

- K8sでは主に以下の種類のログを記録

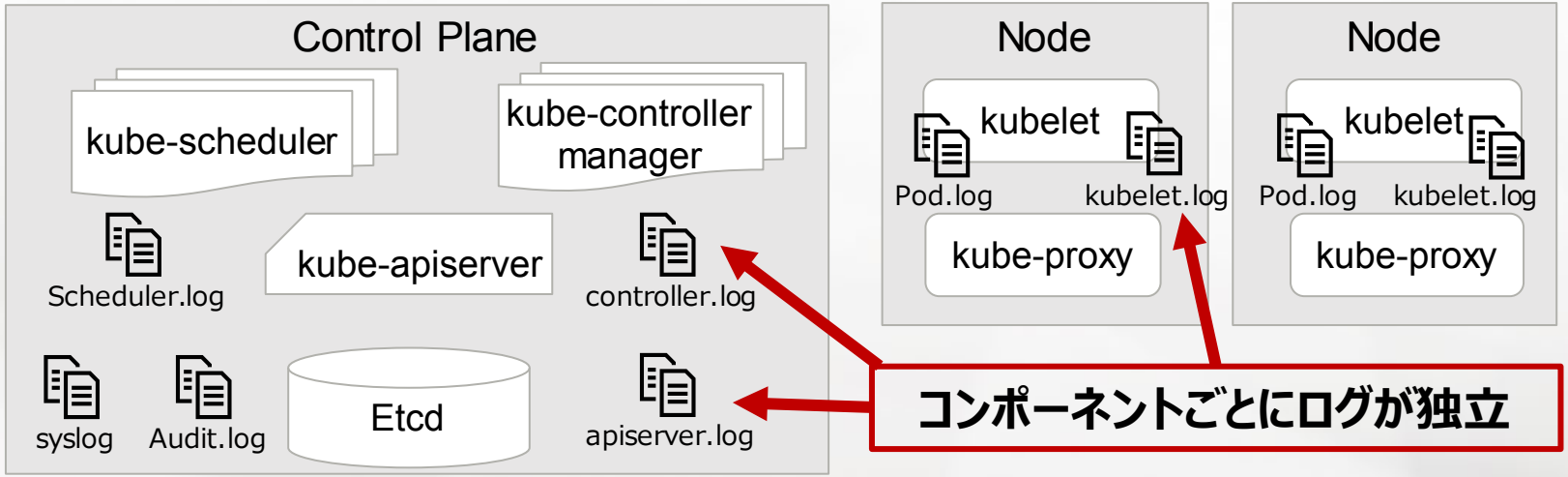
ログの種類	概要	主な用途
Podログ	Pod上で動作するアプリの標準出力/エラー出力	アプリ側のトラブルシューティング
イベントログ	K8sクラスタで発生したイベント	K8sクラスタ環境側のトラブルシューティング
コンポーネントログ	K8sコンポーネントごとの個別ログ(kubelet、kube-apiserverなど)	イベントログより詳細な内部動作を追う場合に利用
Auditログ	kube-apiserverとやりとりするHTTPリクエスト/レスポンス	セキュリティ/監査など

- トラブルシューティングで最もよく扱うログはPodログとイベントログ
- この2つのログで原因を絞り込めない場合、コンポーネントログを見る場合あり

Kubernetesのロギング機能の課題

FUJITSU

- コンポーネント間を跨るログ調査が困難
  - K8sでは任意の操作(kubectl applyなど)の延長で、複数のコンポーネントが動作する
  - このため、クラスタ内部の一連の動作を追う場合は、コンポーネントを跨ったログ解析が必要
  - しかし、K8sのコンポーネントログは、**コンポーネントごとに独立して出力**される
  - このため、個々のコンポーネントのログから関連するログを抽出し、紐づける作業が必要



コンポーネント間のログの紐づけ方法

FUJITSU

- 既存の手法
  - 各コンポーネントのログをリソース名でフィルタリング
    (例: Pod作成に関するログを紐づける場合は、Pod作成時の名前で各ログをクエリする)
  - 各コンポーネントのログを時系列で並べ、近い時間帯のログを精査して手作業で紐づける
- 既存手法の課題
  - リソース名のないログは抽出不可
  - 同じ時間帯に複数のAPIリクエストが投入された場合、仕分けに時間がかかる
  - 階層的なリソースに対する複数Controller開発をする場合、ログの抽出や紐づけが大変

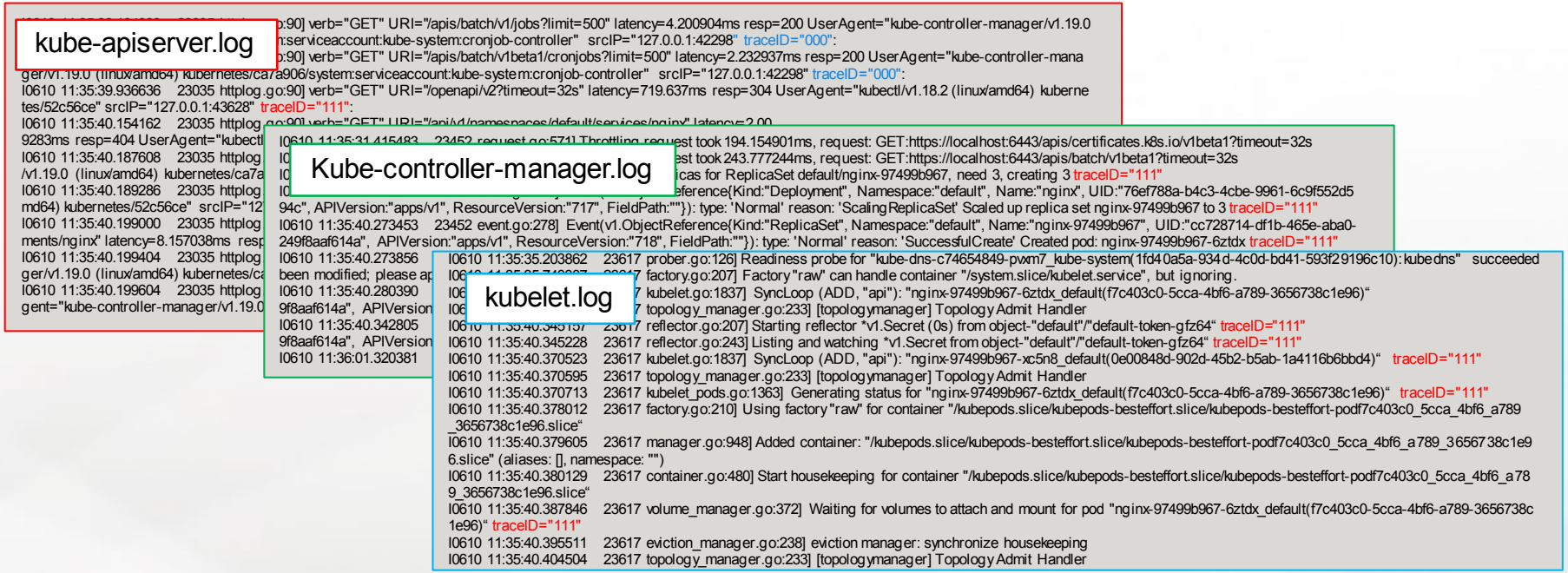
提案している解決策

FUJITSU

- 任意のユーザー操作(kubectl applyなど)の結果として各コンポーネントが出力するログに一意なID(以降Trace IDと呼ぶ)を付与する
- Trace IDのメリット
  - 一連のK8s内部の動作を追う場合に、各コンポーネントのログをTrace IDでフィルタリングするだけで関連するログを簡単に抽出可能
  - リソース名のないログも対応可能
  - タイムスタンプやログの内容精査などの、関連するログを仕分ける作業も不要
  - 自作のオブジェクトのコントローラーにも導入可能

実際のログ例でのイメージ(1/2)

- 例としてDeploymentの作成操作に関するログの抽出(一部を抜粋)
- ログの末尾に付与されている**TraceIDの数値をキー**として、関連する処理を抽出可能



実際のログ例でのイメージ(2/2)

- TraceIDをキーにログを抽出して時系列順に整理することで、処理の流れを把握可能
- エラーログからの原因調査は、エラーログに付与されているTraceIDをキーとして用いる
- 実際のログの収集や解析はElasticsearchやKibanaなど他OSSの使用を想定

```
I0610 11:35:39.936636  23035 httplog.go:90] verb="GET" URI="/openapi/v2?timeout=32s" latency=719.637ms resp=304 UserAgent="kubectl/v1.18.2 (linux/amd64) kubernetes/52c56ce" srcIP=
“127.0.0.1:43628" traceID="111":
I0610 11:35:40.154162  23035 httplog.go:90] verb="GET" URI="/api/v1/namespaces/default/services/nginx" latency=2.009283ms resp=404 UserAgent="kubectl/v1.18.2 (linux/amd64) Kubernete
s/52c56ce" srcIP="127.0.0.1:43628" traceID="111":
I0610 11:35:40.189286  23035 httplog.go:90] verb="POST" URI="/api/v1/namespaces/default/services" latency=33.947136ms resp=201 UserAgent="kubectl/v1.18.2 (linux/amd64) kubernetes/5
2c56ce" srcIP="127.0.0.1:43628" traceID="111":
I0610 11:35:40.199000  23035 httplog.go:90] verb="GET" URI="/apis/apps/v1/namespaces/default/deployments/nginx" latency=8.157038ms resp=404 UserAgent="kubectl/v1.18.2 (linux/amd64
) kubernetes/52c56ce" srcIP="127.0.0.1:43628" traceID="111":
I0610 11:35:40.244334  23452 replica_set.go:562] Too few replicas for ReplicaSet default/nginx-97499b967, need 3, creating 3 traceID="111"
I0610 11:35:40.245614  23452 event.go:278] Event(v1.ObjectReference{Kind:"Deployment", Namespace:"default", Name:"nginx", UID:"76ef788a-b4c3-4cbe-9961-6c9f552d594c", APIVersion:"a
pps/v1", ResourceVersion:"717", FieldPath:""}): type: 'Normal' reason: 'ScalingReplicaSet' Scaled up replica set nginx-97499b967 to 3 traceID="111"
I0610 11:35:40.273453  23452 event.go:278] Event(v1.ObjectReference{Kind:"ReplicaSet", Namespace:"default", Name:"nginx-97499b967", UID:"cc728714-df1b-465e-aba0-249f8aaf614a", API
Version:“apps/v1", ResourceVersion:"718", FieldPath:""}): type: 'Normal' reason: 'SuccessfulCreate' Created pod: nginx-97499b967-6ztdx traceID="111"
I0610 11:35:40.273856  23452 deployment_controller.go:485] Error syncing deployment default/nginx: Operation cannot be fulfilled on deployments.apps “nginx”: the object has been modified; p
lease apply your changes to the latest version and try again traceID="111"
I0610 11:35:40.280390  23452 event.go:278] Event(v1.ObjectReference{Kind:"ReplicaSet", Namespace:"default", Name:"nginx-97499b967", UID:"cc728714-df1b-465e-aba0-249f8aaf614a", API
Version:"apps/v1", ResourceVersion:"718", FieldPath:""}): type: 'Normal' reason: 'SuccessfulCreate' Created pod: nginx-97499b967-xc5n8 traceID="111"
I0610 11:35:40.342805  23452 event.go:278] Event(v1.ObjectReference{Kind:"ReplicaSet", Namespace:"default", Name:"nginx-97499b967", UID:"cc728714-df1b-465e-aba0-249f8aaf614a", API
Version:"apps/v1", ResourceVersion:"718", FieldPath:""}): type: 'Normal' reason: 'SuccessfulCreate' Created pod: nginx-97499b967-sgfg2 traceID="111"
I0610 11:35:40.345157  23617 reflector.go:207] Starting reflector *v1.Secret (0s) from object-"default"/"default-token-gfz64“ traceID="111"
I0610 11:35:40.345228  23617 reflector.go:243] Listing and watching *v1.Secret from object-"default"/"default-token-gfz64“ traceID="111"
I0610 11:35:40.370523  23617 kubelet.go:1837] SyncLoop (ADD, "api"): "nginx-97499b967-xc5n8_default(0e00848d-902d-45b2-b5ab-1a4116b6bbd4)“ traceID="111"
I0610 11:35:40.370595  23617 topology_manager.go:233] [topologymanager] Topology Admit Handler
I0610 11:35:40.370713  23617 kubelet_pods.go:1363] Generating status for "nginx-97499b967-6ztdx_default(f7c403c0-5cca-4bf6-a789-3656738c1e96)“ traceID="111"
I0610 11:35:40.387846  23617 volume_manager.go:372] Waiting for volumes to attach and mount for pod "nginx-97499b967-6ztdx_default(f7c403c0-5cca-4bf6-a789-3656738c1e96)“ traceID="111"
```

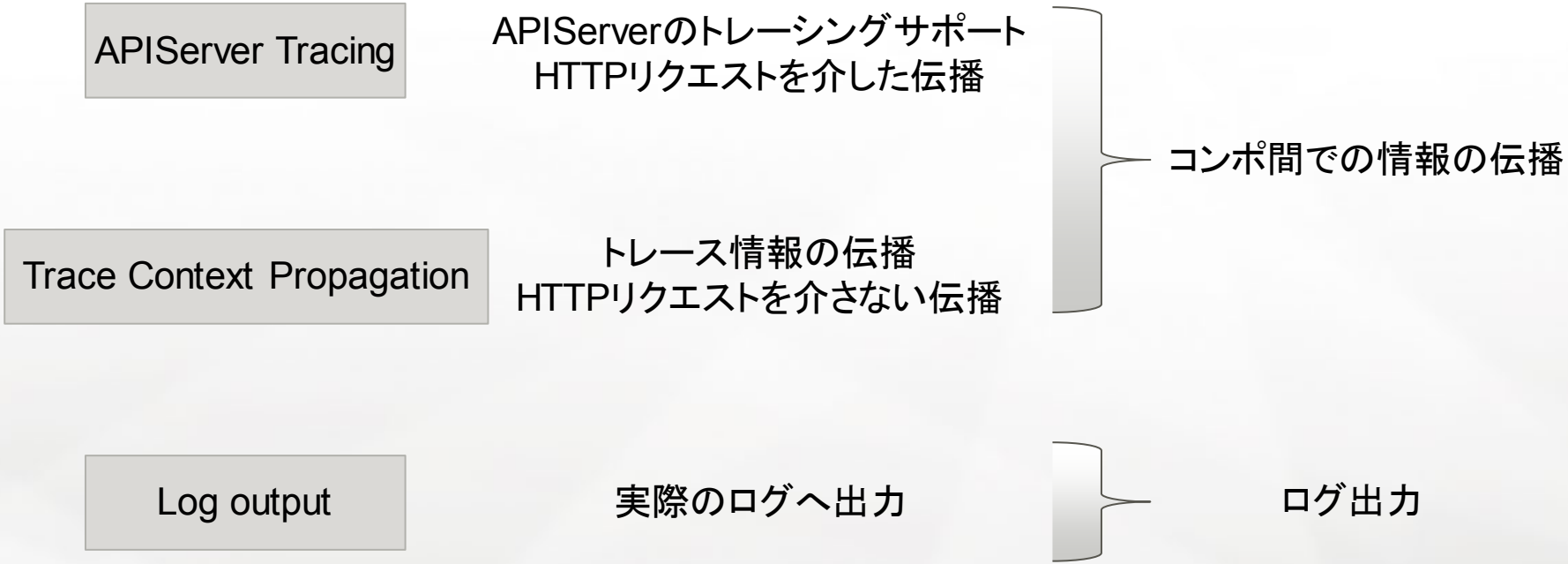
アジェンダ

- 提案中の機能の概要
  - モチベーション・既存の問題点
  - 提案している解決策
- **機能の実装について**
  - 機能実現に必要な要素
  - 機能の実装方針
  - 実装に関する現在の課題
- コミュニティ活動で苦労した点

機能実現に必要な要素

FUJITSU

■ ログ強化の機能は、主に以下の要素から実現される



APIServer Tracing

FUJITSU

- **どのような要素か？**
  - APIServerが受け取ったリクエストのトレーシングが可能になる
  - トレース情報の生成とエクスポートを行う
  - 他KEPで進行中
    （https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/647-apiserver-tracing）

- **用語の定義**
  - スパン：開始と終了の時刻を持ったトレースの最小要素
  - トレース：一つの処理に関するスパンを集めたもの
  - トレースコンテキスト：トレース情報への参照

APIServer Tracing

FUJITSU

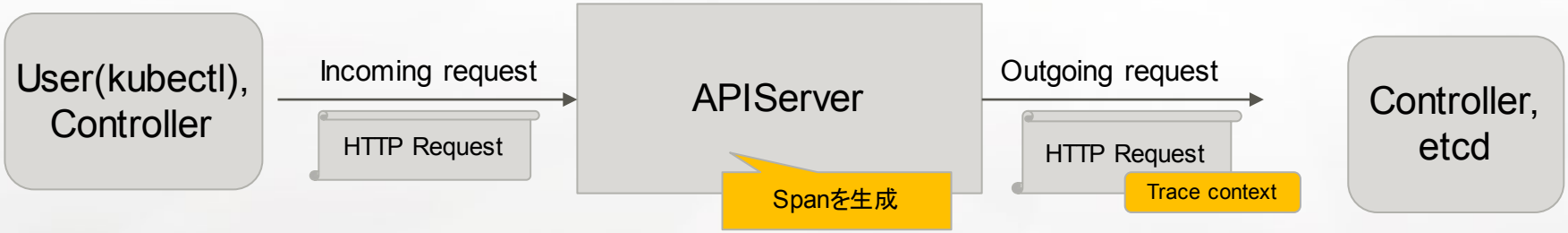
- **OpenTelemetryを使用し、APIServerのトレーシングを可能に**
  - incoming/outgoing requestのスパンを生成し、エクスポート
  - トレースコンテキストをincoming requestからoutgoing requestに伝播

- **OpenTelemetry : https://opentelemetry.io/**
  - CNCFがホストするobservabilityのフレームワーク
  - メトリクス・ログ・トレースの計測・生成・収集・エクスポートを行う

APIServer Tracing

FUJITSU

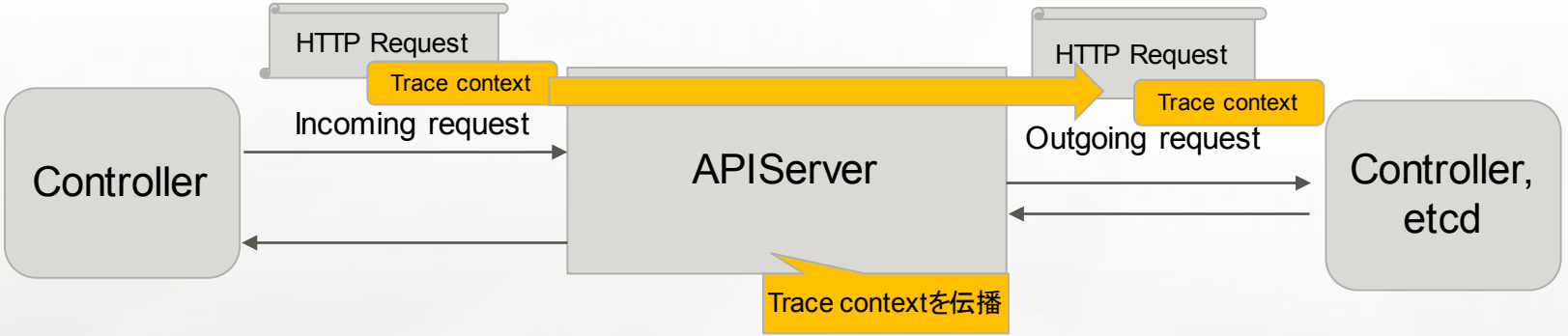
- **OpenTelemetryを使用し、APIServerのトレーシングを可能に**
  - incoming/outgoing requestのスパンを生成し、エクスポート
  - トレースコンテキストをincoming requestからoutgoing requestに伝播



APIServer Tracing

FUJITSU

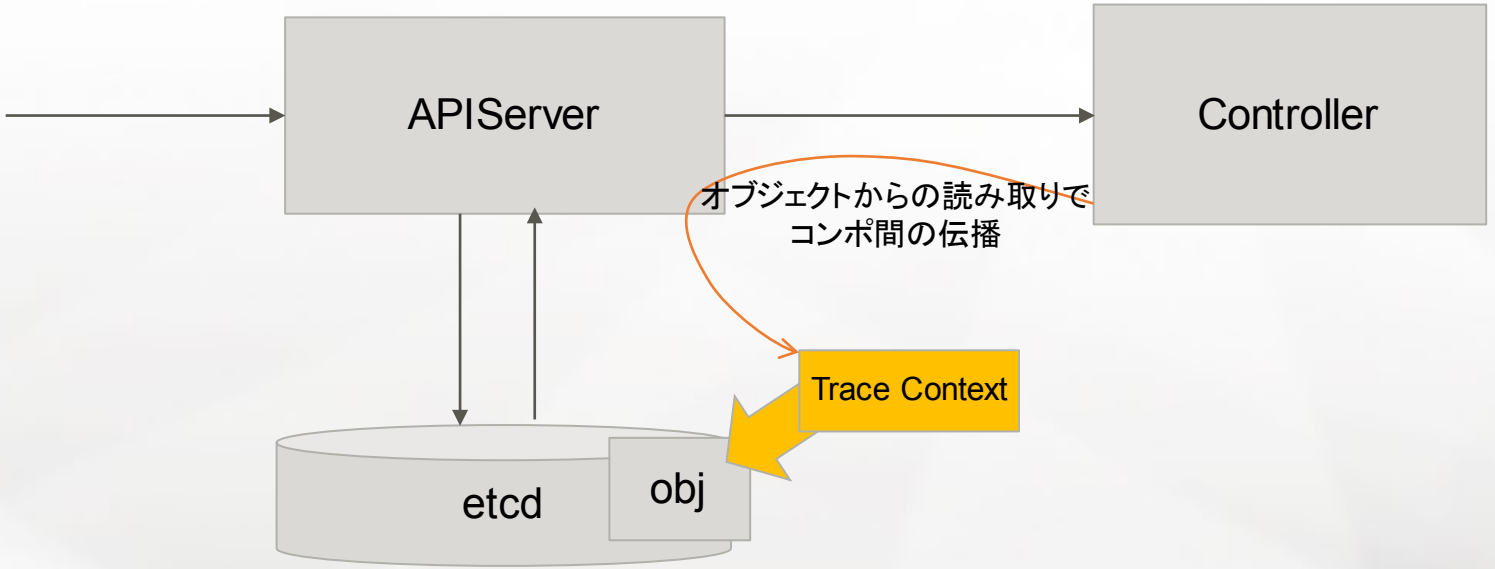
- **OpenTelemetryを使用し、APIServerのトレーシングを可能に**
  - incoming/outgoing requestのスパンを生成し、エクスポート
  - トレースコンテキストをincoming requestからoutgoing requestに伝播



Trace Context Propagation

FUJITSU

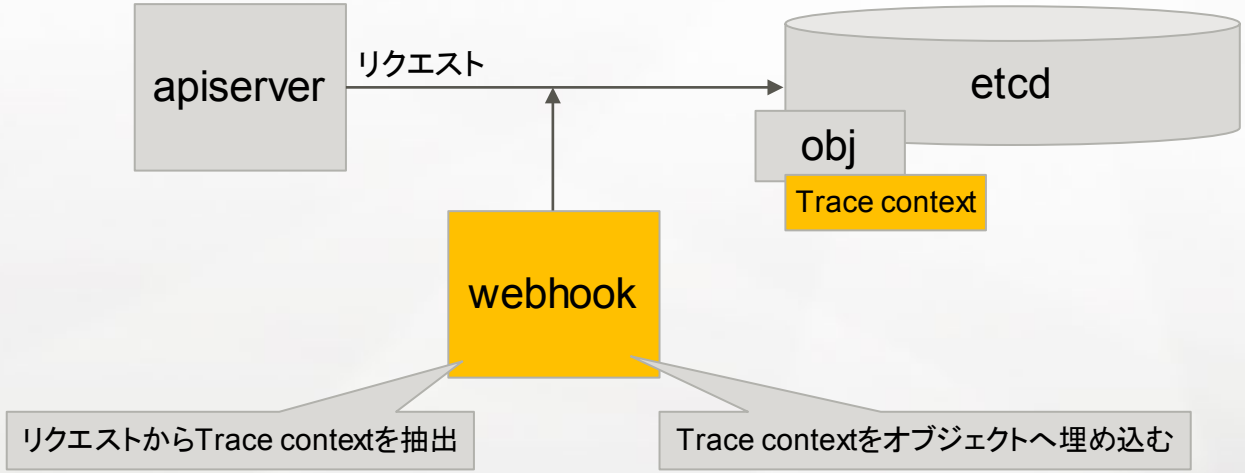
- **どのような要素か？**
  - オブジェクトへのトレースコンテキストの保存
  - コントローラーへのトレースコンテキストの伝播



オブジェクトへのトレースコンテキストの保存

FUJITSU

- **Mutating Admission Webhookを導入**
  - Admission Webhookは、リソースの作成や更新をトリガーとして、validationやmutationを行うことができる
  - リクエストからTrace Contextを抽出し、オブジェクトへ埋め込む



コントローラへのトレースコンテキストの伝播

FUJITSU

- **トレースコンテキストをオブジェクトアノテーションとして伝播**
- **ユーティリティ関数を導入して伝播の仕組みを整える**
  - オブジェクトからTrace Contextを取り出すなどの処理を行う関数群
  - 実行された作業を、同じアクション（user request）に属するものとして、コンポーネントを跨ってリンクするには、プロセスの境界を越えてトレースコンテキストを渡す必要がある
  - 従来の分散システムでは、このコンテキストはRPCメタデータまたはHTTPヘッダーを介して受け渡される
  - ただし、Kubernetesはウォッチベースのため、トレースコンテキストをターゲットオブジェクトに直接アタッチする必要がある。

Log output

FUJITSU

- **どのような要素か？**
  - トレースコンテキストをログに出力することで、ログの紐づけが可能になる
  - Trace IDにより、任意のユーザーアクションに関連したログを一括で抽出できる

- Structured logging（https://github.com/kubernetes/enhancements/tree/master/keps/sig-instrumentation/1602-structured-logging）
  - 構造化されたログ出力を実現するKEP
  - この機能を利用して、構造化形式でのログ出力を予定している

実装に関する現在の課題

FUJITSU

- これまで提案していた内容は、深い議論が必要な部分が二つ存在する
  - Contextをオブジェクト間で伝播する
  - Trace id をログに出力する
- このままでは承認が難しいという指摘
  - 議論が二つ立ち上がる
  - 承認が必要なSIGが増える　などの理由
- 二つの部分を分割して提案することに
  - KEPではコンテクストのコントローラーへの伝播をメインスコープに
  - ログの部分は、issueを起こしてそこで実装する予定

開発の現状

FUJITSU

- **コミュニティ状況**
  - KEP状況（https://github.com/kubernetes/enhancements/pull/2312）
  - Trace context propagationの機能のαを1.22に実現が目標
- **提案中の主要機能：**
  - Trace context をオブジェクトアノテーションに付与して伝播
  - Trace context から必要な情報をログに出力
- **分割後の状況**
  - Trace context propagation と Log output の二つに分割
  - 現在はまずTrace context propagationの実現に向けて活動中
  - Log output はKEPで提案かissueで提案して実現を目指す

アジェンダ

- 提案中の機能の概要
  - モチベーション・既存の問題点
  - 提案している解決策
- 機能の実装について
  - 機能実現に必要な要素
  - 機能の実装方針
  - 実装に関する現在の課題
- **コミュニティ活動で苦労した点**

コミュニティ活動で苦労した点

FUJITSU

- **なぜ分割という話に？**
  - 最初は一人のTech leadの人にレビューをもらいながら進めていた
  - 締め切りが近くなると、ほかのレビュアーからもコメントをもらえるように
  - KEPが複雑になりすぎている、承認が必要なSIGが増えるといった指摘
  - Trace Context伝播部とログ出力部を分割することにした
  - とはいえ、承認が簡単になったわけではなく、各部分へ集中するための分割

コミュニティ活動で苦労した点

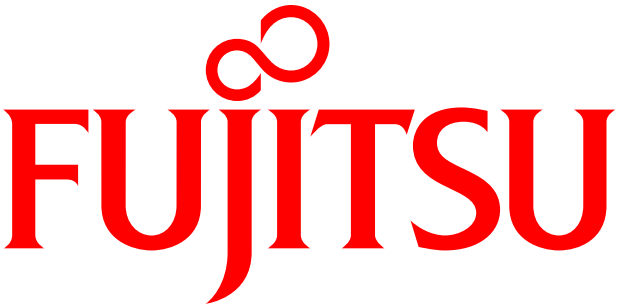
FUJITSU

- **実際に起こったこと**
  - 他のKEPやOSSの影響でスケジュールがどんどん後ろ倒しに
  - Production Readiness Review Questionnaireの書き方が大きく変わり、KEPを書き直した
  - SIG-Instrumentationだけの範囲で済むように調整していたが、ほかのSIG（SIG-Architecture）の承認も必要になった
  - ”ほかのSIG”がSIG-API-machineryに変わった

まとめ

FUJITSU

- **ログ機能の強化のためのKEPを提案しています**
  - Trace Context Propagation
  - Log output
- **1.22でαが導入できるように活動しています**
- **コミュニティとのやりとりはいろいろと大変ですが、勉強になることも面白いことも多いです**



shaping tomorrow with you