ORACLE

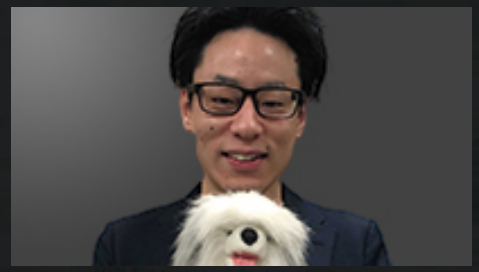
Developer Days 2022 Spring

未来を創造する最新テクノロジーを今、あなたの手に。

【Showcase】
OCI Cloud Native Observability Service

市川 豊

Oracle Groundbreakers Advocate
ソリューション・アーキテクト本部
プリンシパルクラウドソリューション エンジニア



仁井田 拓也

Oracle Groundbreakers Advocate
ソリューション・アーキテクト本部
クラウドソリューション エンジニア

Profile


Name

Yutaka Ichikawa/市川 豊

Belong

Solutions Architect

Role

Principal Cloud Solution Engineer

SNS

Twitter/GitHub/Qiita:cyberblack28

Blog

https://cyberblack28.hatenablog.com/

Materials

https://speakerdeck.com/cyberblack28/

Community

- Oracle Cloud Hangout Café #ochacafe
- CloudNative Days Tokyo #CNDT2021 #CICD2021 #o11y2022

Certified

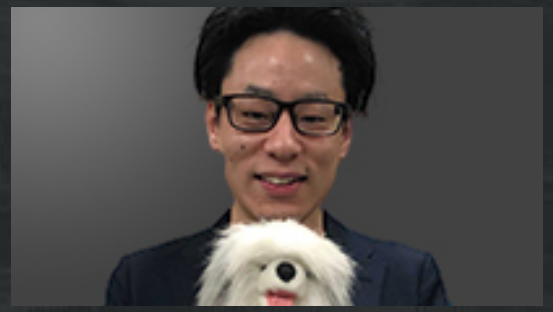
- Certified Kubernetes Administrator
- Certified Kubernetes Application Developer
- Certified Kubernetes Security Specialist
- Kubernetes and Cloud Native Associate

Publications

Dockerコンテナ開発・環境構築の基本

2021/7/27 発売

Support Engineer



Takuya Niita/仁井田 拓也

Oracle Groundbreaker Advocate

Cloud Solution Engineer

タイムスケジュール

■オブザバビリティの概要とOCI Observability Service 紹介 (40分)

■デモ / 解説

1. OKEとサンプルアプリケーションについて (10分)
2. Application Performance Monitoring (30分)
3. Logging (10分)
4. Monitoring & Notifications (15分)

＊セッションやデモの進行状況によって多少終了時間が前後する場合がございます。

オブザバビリティの概要と OCI Observability Service 紹介

Agenda

Observability

- そもそも監視って？
- 背景
- What’s Observability ?
- Observabilityの3要素

Metrics Logs Traces

- Metrics
- Logs
- Traces

デモンストレーションについて

- デモンストレーション概要
- 参考資料

Observability

そもそも監視って？

Observability

そもそも監視って？

Monitoring = 監視、観察して記録する

Observability

そもそも監視って？

何のために、監視、観察して記録する？

Observability

そもそも監視って？

- サービスやアプリケーションの健全性を確認
- 障害やトラブルの原因調査
- キャパシティ分析
- サービス利用者の行動分析

技術、運用、ビジネスなどに多岐にわたる

Observability

そもそも監視って？

サービスやシステムを利用するユーザに
影響を与えないため

▼

ユーザ体験を損なわないようにするため

Observability

そもそも監視って？

サービスやシステムの利用者が、問題なく利用できる、
「安定稼働している状態」を維持する！

Observability

そもそも監視って？

監視の意義

- より良い方法で、システムの稼働状況を把握できている状態
- システム運用において、判断に必要となる情報を取得できている状態
- 迅速に障害やトラブルに対応できる状態

これまでもこれからも、こうした本質は変わらない

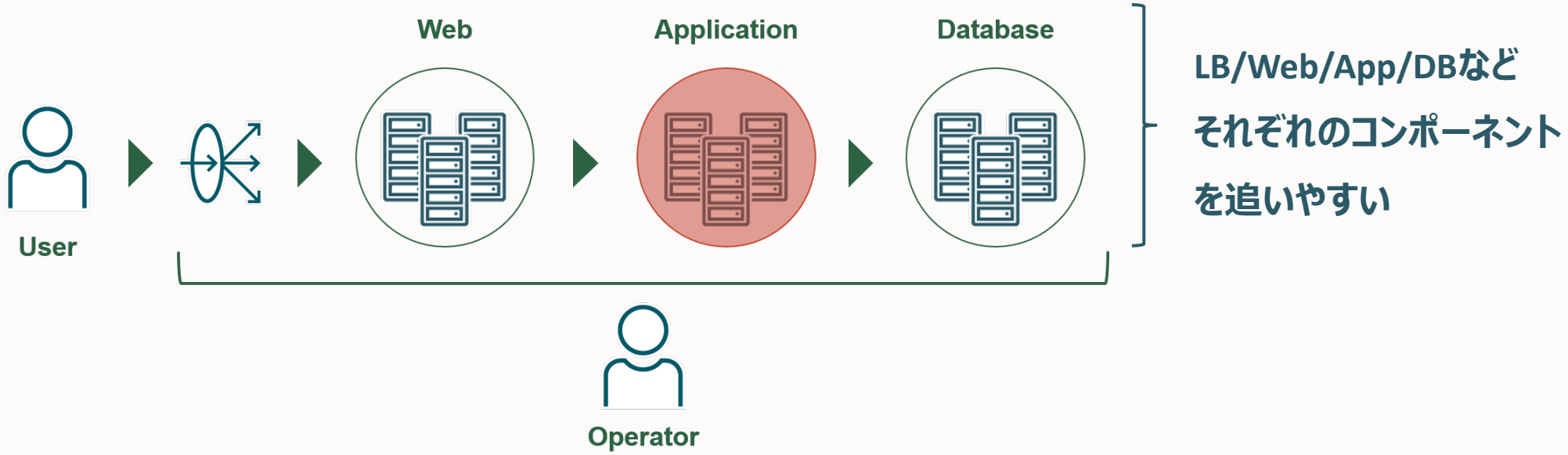
背景

Observability

背景

これまでのシステム

従来のWeb3層モデルのようなシンプルな構成のシステムであれば、比較的容易に障害を調査することが可能。

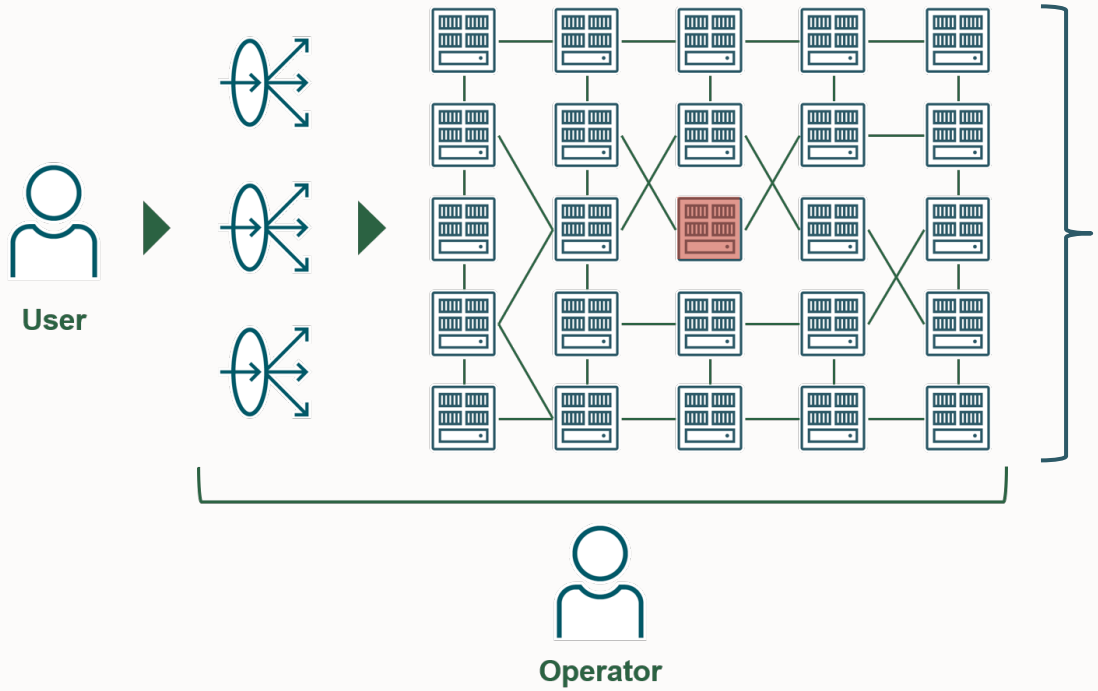


Observability

背景

分散システム

分散システムのような小さなサービスが疎結合するようなシステムでは、構成が複雑となり障害発生個所や原因追求が困難であり、まして人の手で行うことは非現実的。

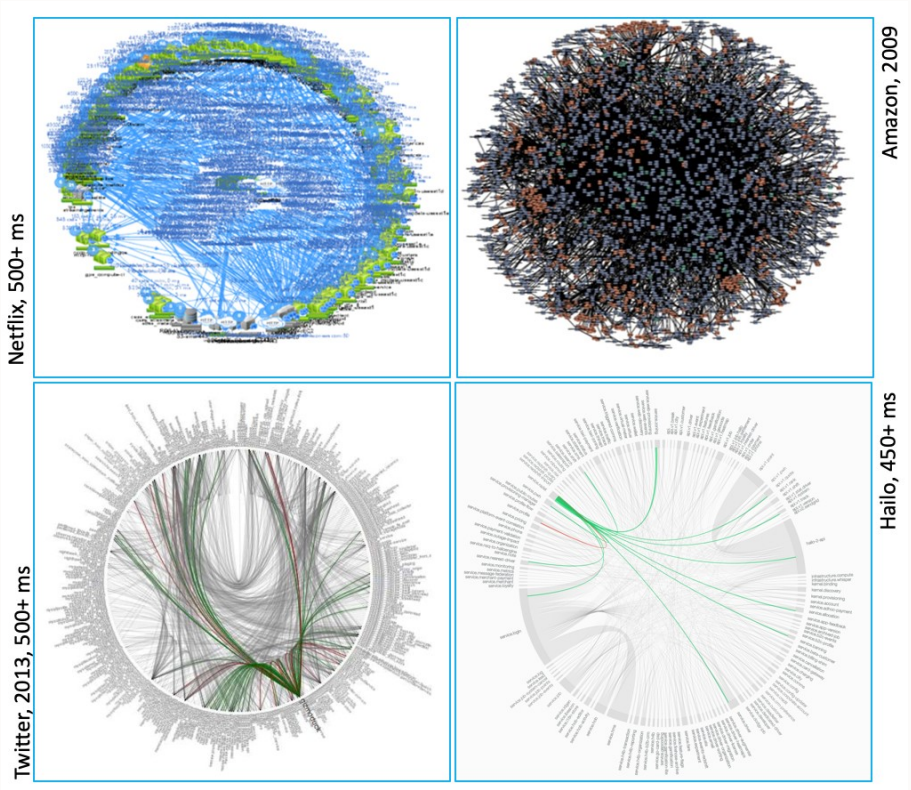


Observability

背景

分散システム

右図にあるような分散システムでは、大量のサービスが連携して、一つのシステムとして成り立っているため、障害が発生した際の検知など、これまでのように容易にはいかない…



『Adoption of Cloud Native Architecture, Part 2: Stabilization Gaps and Anti-Patterns』
https://www.infoq.com/articles/cloud-native-architecture-adoption-part2/

What’s Observability ?

Observability

What’s Observability ?

Observabilityの意味

Observability = 可観測性

> ***Observability might mean different things to different people.***
>
> **可観測性は、人によって意味が異なる場合があります。**

『Distributed Systems Observability』

https://www.oreilly.com/library/view/distributed-systems-observability/9781492033431/

Observability（可観測性）は、人によってまたはシステムによって
基準、観点、解釈の仕方が違うものなので、本セッションの内容も一例と捉えてください。

Observability

What’s Observability ?

クラウドネイティブにおけるObservability

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

クラウドネイティブ技術は、パブリッククラウド、プライベートクラウド、ハイブリッドクラウドなどの近代的でダイナミックな環境において、スケーラブルなアプリケーションを構築および実行するための能力を組織にもたらします。このアプローチの代表例に、コンテナ、サービスメッシュ、マイクロサービス、イミュータブルインフラストラクチャ、および宣言型APIがあります。

これらの手法により、回復性、管理力、および可観測性のある疎結合システムが実現します。これらを堅牢な自動化と組み合わせることで、エンジニアはインパクトのある変更を最小限の労力で頻繁かつ予測どおりに行うことができます。

『CNCF Cloud Native Definition v1.0』

https://github.com/cncf/toc/blob/main/DEFINITION.md

クラウドネイティブの文脈では、Observability（可観測性）は、
クラウドネイティブなシステムを実現、支える一要素

Observability

What’s Observability ?

クラウドネイティブにおけるObservability

クラウドネイティブ技術

速く、正確に高品質なサービスを提供して、
エンドユーザ様の満足度、企業収益、ビジネス価値の向上

Observability

What’s Observability ?

クラウドネイティブにおけるObservability

提供だけではなく、常にユーザエクスペリエンスを損なうことが無いよう維持、

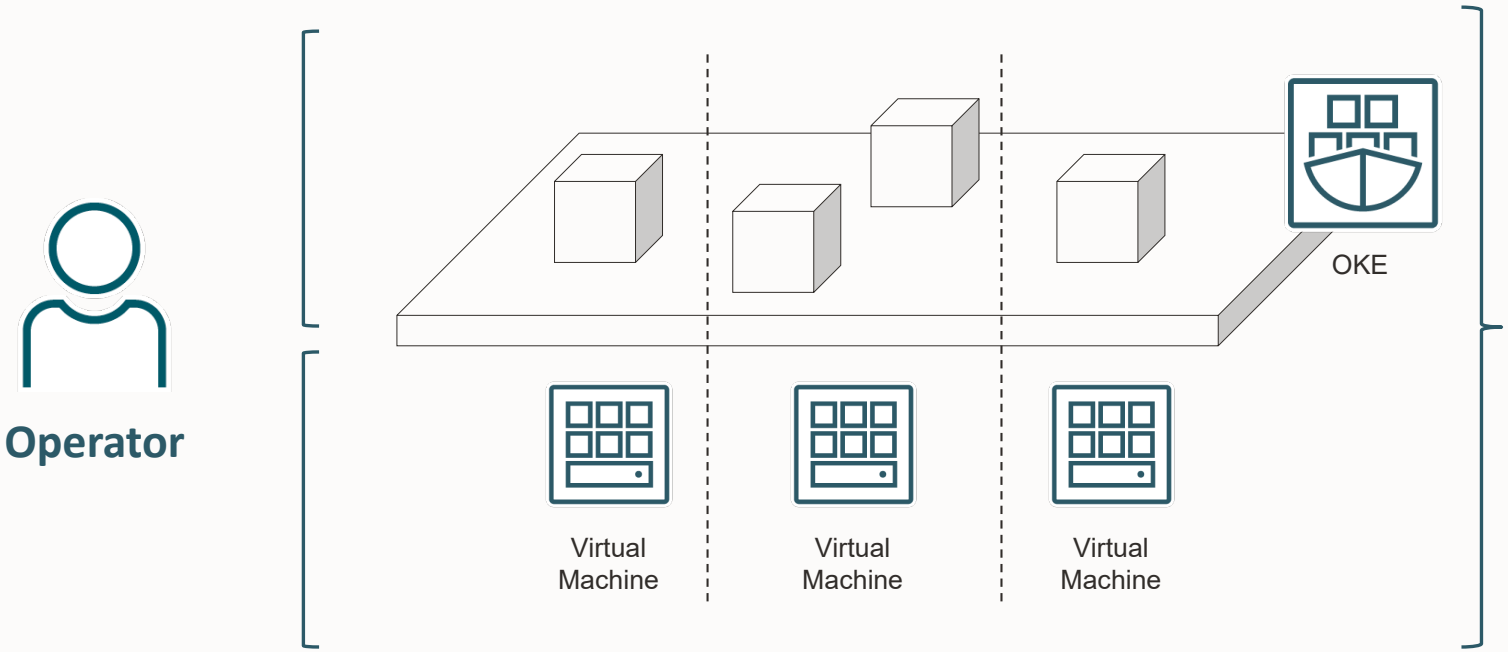
そして、障害やトラブルが発生した場合も速く、正確に対応できる体制も維持する！

Observability

What’s Observability ?

クラウドネイティブにおけるObservability

Kubernetes上のPod(コンテナ)を例に考えてみると、決められたNodeに決められたPod(コンテナ)が稼働するとは限らないため、これまでの監視方法とは違うアプローチが必要となる。



Node状況、Kubernetes Cluster状況（Podなど）、アプリケーション、データベースなど色々ある

Observability

What’s Observability ?

結局のところObservabilityとは？

提供だけではなく、常にユーザエクスペリエンスを損なうことが無いよう維持、

そして、障害やトラブルが発生した場合も速く、正確に対応できる体制も維持する！

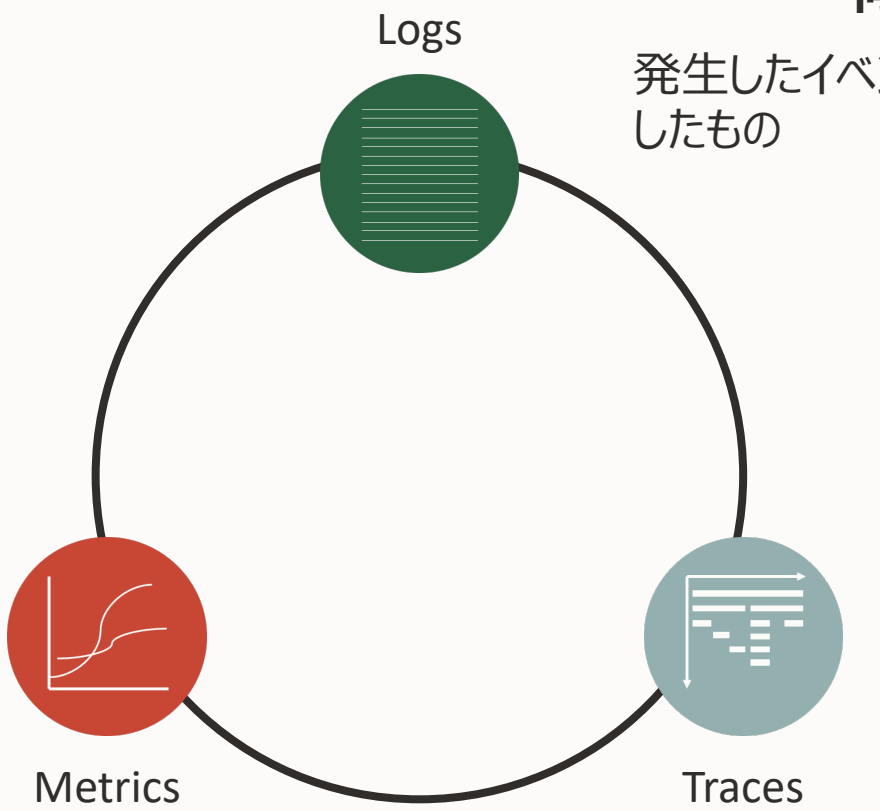
事象を捉えて、なぜ発生したのかを究明、そして解明する

Obervabilityの3要素

Observability

Observabilityの3要素

各要素が連携してObservabilityを実現させることが重要



何が起きたのか

発生したイベントをタイムスタンプで記録したもの

何が起きているのか

特定の時間間隔で測定して統計したもの

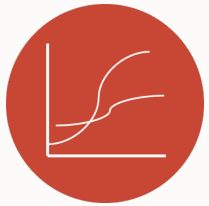
どこで起きたのか

依存関係がある一連のリクエストフローを始まりから終わりまでを変換したもの

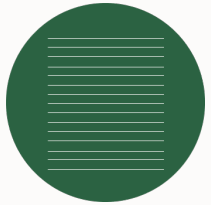
Observability

Observabilityの3要素

Observabilityを実現する主なツール



Metrics



Logs

Traces

Prometheus & Grafana

Grafana Loki

EFK

Jaeger

Zipkin

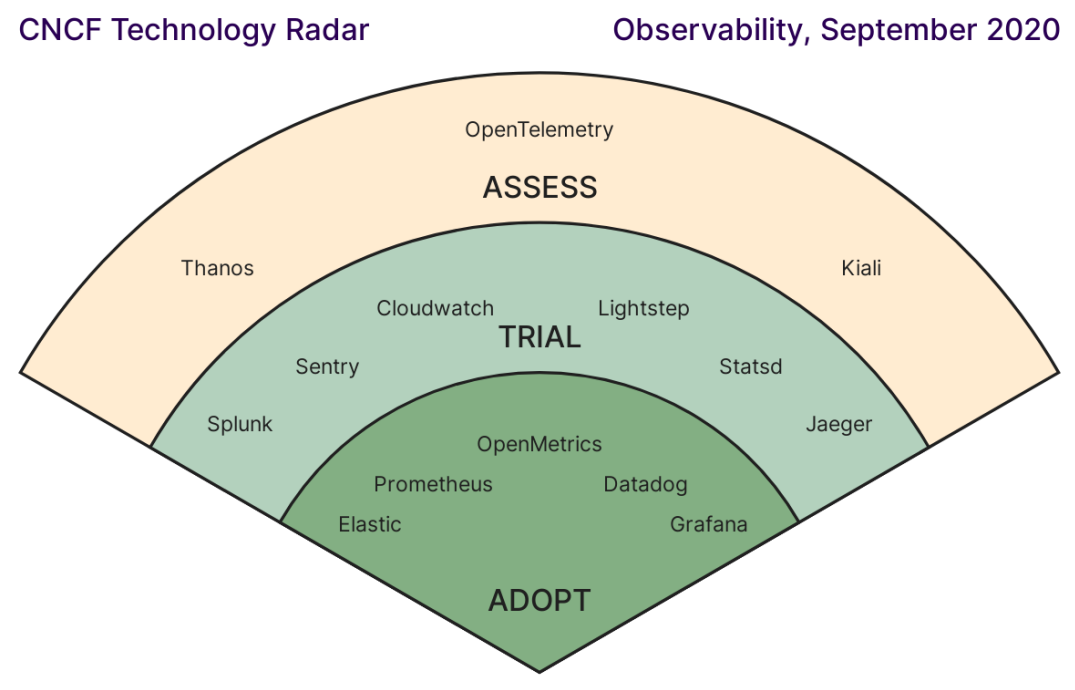
Open Telemetry

Observability

Observabilityの3要素

The CNCF End User Technology Radar

The CNCF End User Technology Radarは、CNCFエンドユーザーコミュニティに代わって、クラウドネイティブテクノロジーを評価するためのガイド



1.最も一般的に採用されているツールはオープンソース

最も「採用」票を獲得した3つのツール（Prometheus、Grafana、Elastic）と最も合計票を獲得した5つのツール（Prometheus、Grafana、Elastic、Jaeger、OpenTelemetry）はすべてオープンソースです。

2.可観測性の領域に統合はない

多くの企業が複数のツールを使用しています。企業の半数は5つ以上のツールを使用しており、3分の1は10以上のツールを使用した経験があります。

3.PrometheusとGrafanaはほぼ一緒に利用

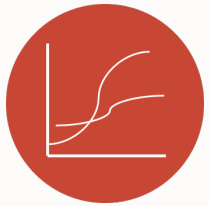
回答者の3分の2は、これら2つのツールを組み合わせて使用しています。これは当然のことですが、高い相関関係は注目に値します。

『The CNCF End User Technology Radar Observability, September 2020』
https://radar.cncf.io/2020-09-observability

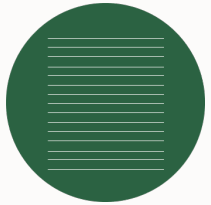
Observability

Observabilityの3要素

Observabilityを実現するOCIサービス



Metrics



Logs

Traces

Monitoring

Notifications

Logging

Logging Analytics

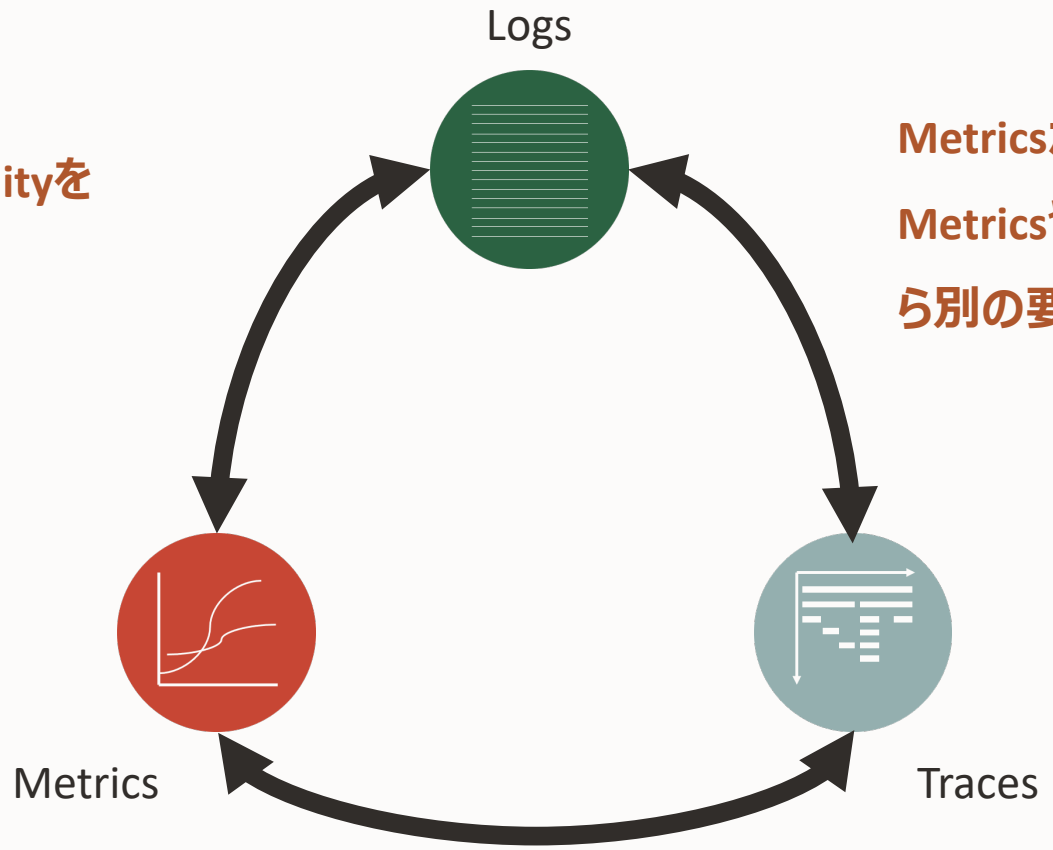
Application Performance Monitoring

Observability

Observabilityの3要素

3要素の調和が重要

各要素が連携してObservabilityを実現させることが重要！！



MetricsからLogsやTraces、TracesからMetricsやLogsのように、1つの要素から別の要素を見据える！！

Metrics Logs Traces

Metrics

Metrics Logs Traces

Metrics

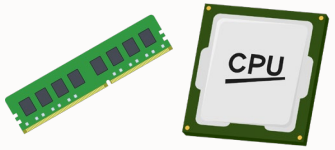
What’s metrics ?

- システムの状態を収集後、付加情報を付与して数値に変換したもの
- 監視対象が経時的にどのような変化をするか統計的に予測するもの

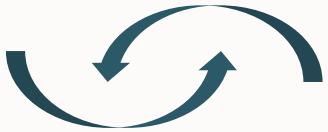
Use of metrics

- Metrics自体では、単なる事実でしかないので、経時的に見て、予測につなげる
- Metricsの数値を閾値として、アラート通知につなげる

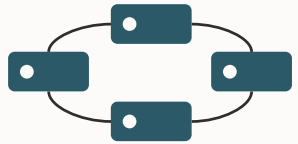
Example of metrics



CPU・メモリの使用率



リクエスト数



ネットワーク通信料

Metrics Logs Traces

Metrics

収集するメトリクス例

メトリクス監視を始める際に、収集するメトリクスを何にすべきか悩む場合の抽象化された例。

1.USE

リソースをベースとしたメトリクス収集

Utilization

リソースの単位時間あたりの使用率(例:CPU、メモリの使用率等)

Saturation

リソースの飽和状況(例:実行キューの長さ等)

Error

エラーイベントのカウント(例:ネットワーク、I/Oのエラーなどをカウント)

2.RED

サービスをベースとしたメトリクス収集

Rate(Request)

秒間のリクエスト数

Error

失敗しているリクエスト数

Duration

リクエストの処理に要した時間

Metrics Logs Traces

Metrics

3.The Four Golden Signals

『SREサイトリライアビリティエンジニアリング』で述べられている4大シグナル

https://www.oreilly.co.jp/books/9784873117911/

Latency

リクエストを処理するのに要した時間。正常なレスポンスと異常なレスポンスは分けるようにする。

Traffic

システムに対するリクエスト量。リクエスト数やネットワークI/O、セッション数など。

Error

処理の失敗。

Saturation

サービスが手一杯になっている状態。メモリ、ディスク、CPUやI/Oのなど。

Metrics Logs Traces

Monitoring

Metrics

Monitoring (モニタリング)

OCI上の様々なリソースのメトリックを監視、ダッシュボードで閲覧、および通知

■ ユースケース

OCI上のサービスやリソースの状態監視、アプリケーションの性能監視、リアルタイムでの異常検出

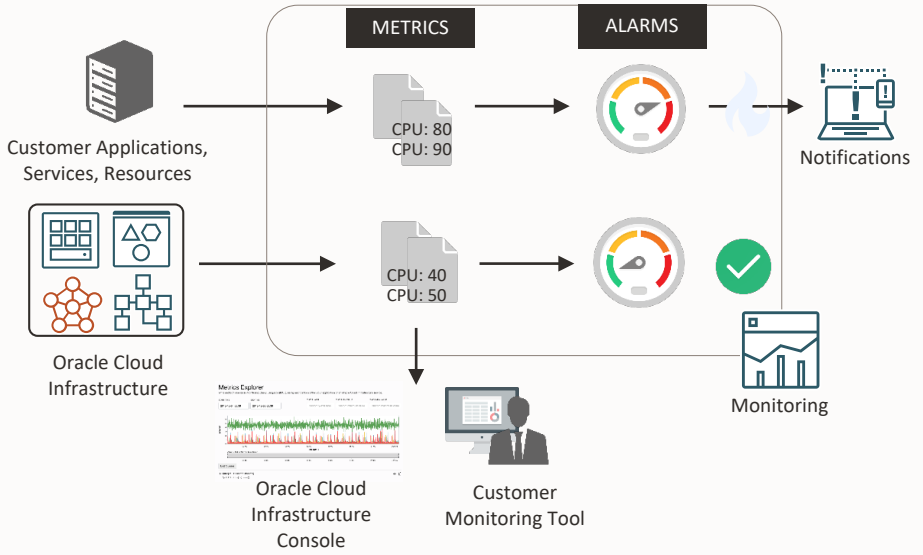
■ 特徴

- 特別な設定は不要でOCI上の各サービスやリソースのメトリックを自動的に取得（コンピュート・インスタンス/VNIC/ブロック・ボリューム/ロードバランサーなど）
- 事前定義済のビジュアライゼーション・ダッシュボードの提供
  カスタム・メトリックの定義も可能
- メトリックに対し、あらかじめ指定した条件にメトリックが合致した場合にアラームを作成することも可能

■ 価格

カスタム・メトリックの取り込み：
最初の5億データポイントは無料、以降100万データポイントごとに\0.3

分析メトリック：
最初の10億データポイントは無料、以降100万データポイントごとに\0.18



■ 関連するOracle Cloud Service

- Notifications(通知)、コンピュート、ネットワーク
  ストレージ、その他OCIサービス全般

Metrics Logs Traces

Monitoring

サービス・メトリクス

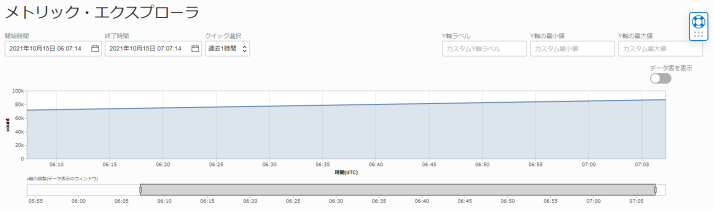
- **ノード・ステータス**

OCI Computeサービスによって示されたコンピュート・ノードのステータス。

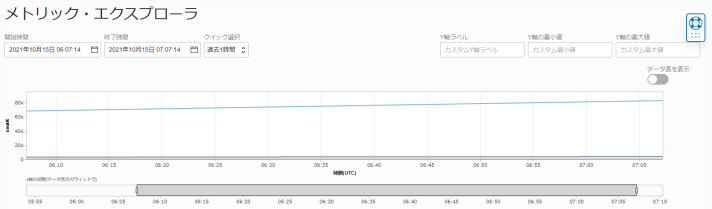
- **Kubernetesのノード条件**

Kubernetes API Serverによって示されたワーカー・ノードの条件。

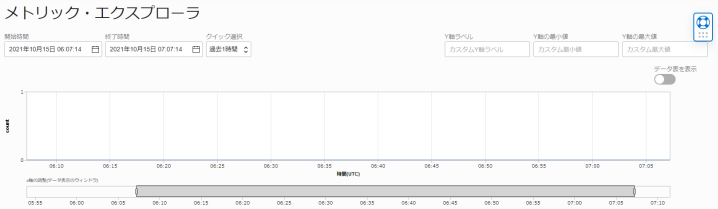
メトリック・エクスプローラー



APIServerRequestCount



APIServerResponseCount



UnschedulablePods

Metrics Logs Traces


Notifications

Notifications (通知)

OCIおよび外部でホスティングされているアプリケーションに対してメッセージをブロードキャスト

■ ユースケース

アプリケーションの統合（アプリケーションと連携して通知処理を実施）

クラウド・ネイティブ・メッセージング（特定のイベントが発生した時に通知処理を実施）

メトリックと監視（Monitoringサービスと連携して通知処理を実施）

■ 特徴

エコシステムからのエンドポイントの選択（SMS、Slack、PagerDuty、HTTPSエンドポイントにメッセージを送信）

Events Serviceによるトリガー（様々なイベントに基づいて通知を実行）

サーバーレス・アプリケーションを簡単に実行

■ 価格

- HTTPS配信：1か月あたり100万件までの配信操作 無料、以降100万件ごと \72.00
- メール配信：1か月あたり1,000通までのメール送信 無料、以降1000通ごと \2.40円
- SMS配信：1か月あたり100件までのSMSメッセージ送信 無料、以降1件ごと（国ゾーン1）\1.80

SMSはゾーン数で価格は変更となるので詳細は、こちらで確認。https://www.oracle.com/jp/devops/notifications/

Logs

Metrics Logs Traces

Logs

What’s logs ?

正常・異常動作など、システムにより生成されるテキストデータ、ファイル、標準出力、標準エラー出力として出力されるもの

Use of logs

- システム安定運用 「いつ、どこで、何が起きた」
- セキュリティ監査 「いつ、誰が、何をした」

Example of logs

システム安定運用

- システムログ
- イベントログ
- アプリケーションログ
- 通信ログ

セキュリティ監査

- 監査ログ
- 認証ログ

Metrics Logs Traces

Logs

取得するログの例 Kubernetes

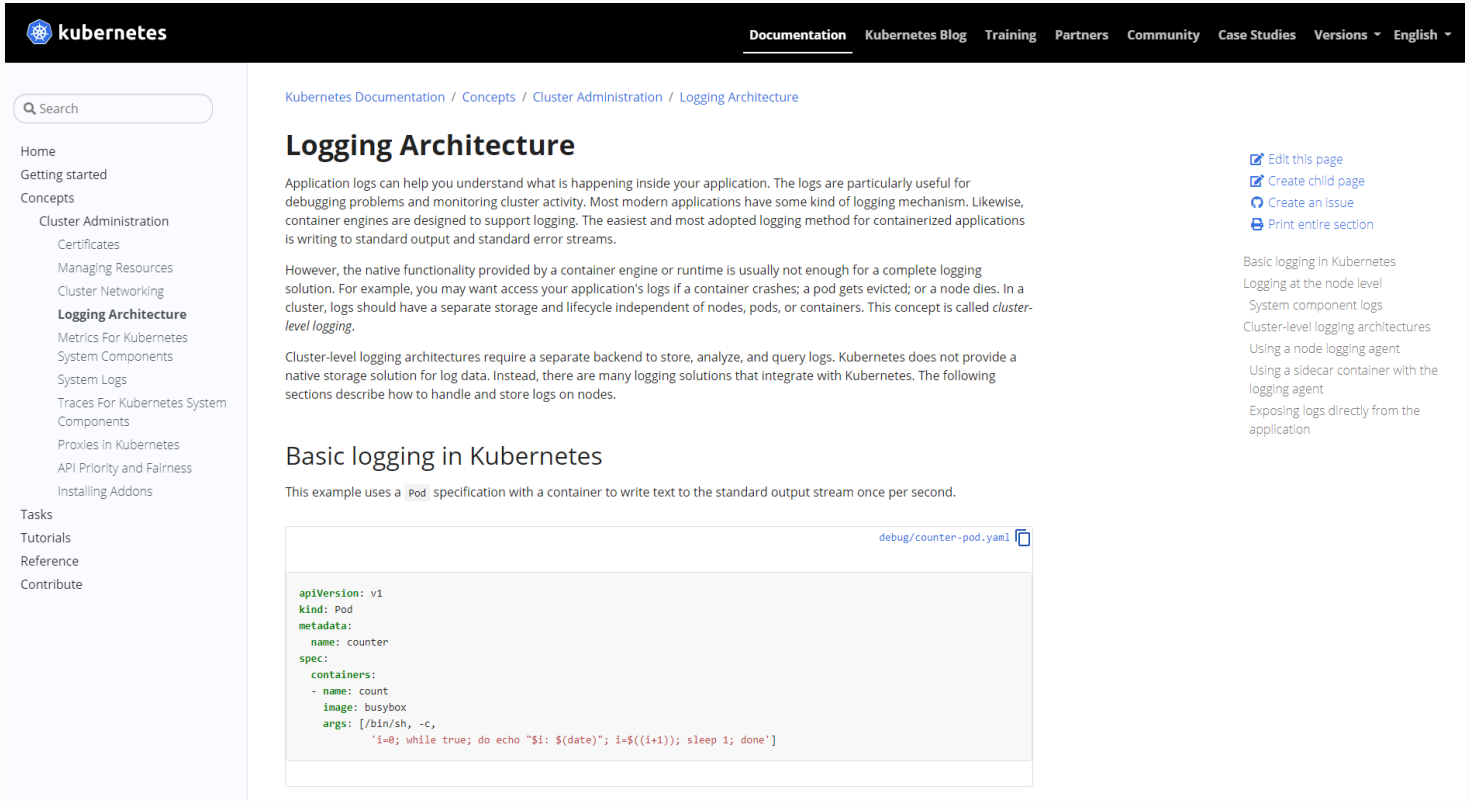
- アプリケーションログ
  - ✓ Kubernetesクラスタ上で稼働しているコンテナアプリケーション
- システムログ
  - ✓ システムコンポーネント（kube-controller-manager、kubeletなど）
  - ✓ システム準コンポーネント（CoreDNSやCNIなど）
  - ✓ Kubernetes Node
- 監査ログ
  - ✓ Kubernetes APIへの接続情報

Kubernetesへの変更はシステムコンポーネントやエコシステムを含めて、すべてAPIサーバを介して行われるので、
どのようなユーザがどのようなリクエストを送っているかを記録した監査ログはセキュリティ上重要

Metrics Logs Traces

Logs

ログの取得方法 Kubernetes



Logging Architecture | https://kubernetes.io/docs/concepts/cluster-administration/logging/

Metrics Logs Traces

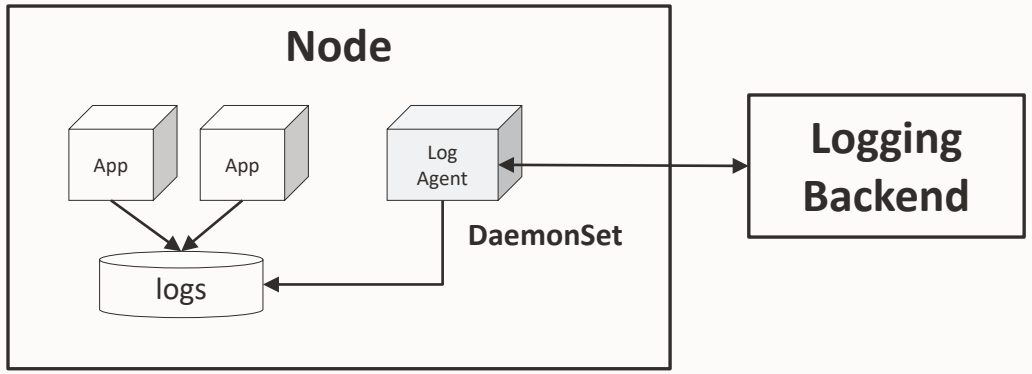
Logs

1.アプリケーションログ

「重要度」、「ログ量」、「整形、フィルタリング有無」などによって、いくつかのパターンでログを取得・送信することが可能

a.DaemonSet

Node単位でログを集約して、DaemonSetが一括にログを転送。コンテナランタイムが出力するログファイルのディレクトリを監視して、ラベルなどの情報を付与して転送。



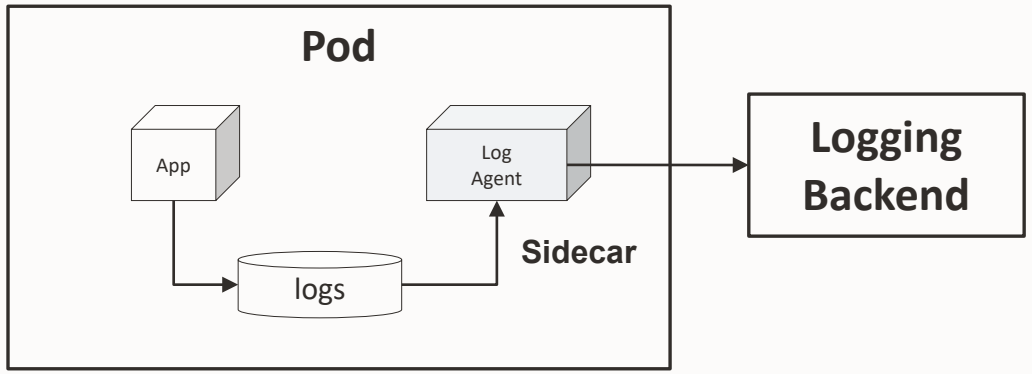
- 個別フォーマットに整形不要、多少の遅延が許容できる場合に有効
- DaemonSet Podが他のPodよりも先に削除されると一部のログが欠損する

Metrics Logs Traces

Logs

b.Sidecar

アプリケーションコンテナのSidecarとして起動しているログエージェントが転送する。



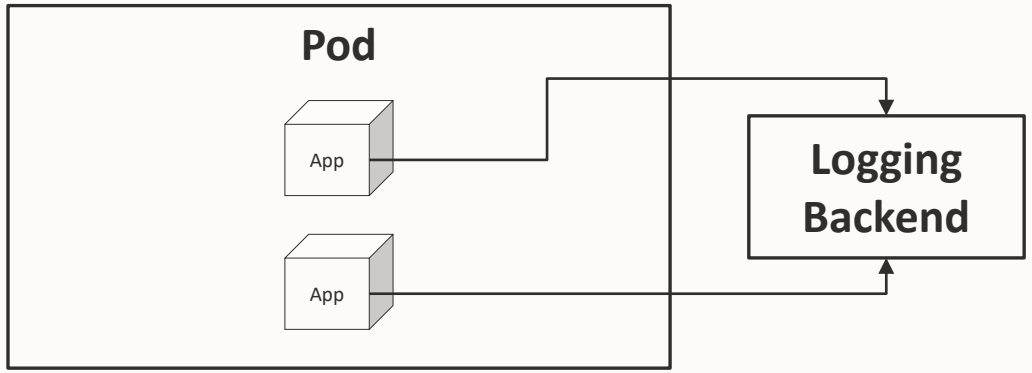
- 汎用的なログエージェントコンテナを作成すれば、他のシステムでもSidecarとして利用できる
- ログエージェントコンテナがアプリケーションコンテナよりも先に停止した場合はログが欠損するため、コンテナの停止タイミングに注意する必要がある

Metrics Logs Traces

Logs

c.Library

アプリケーションからライブラリを利用して直接ログを転送する。



- アプリケーション側にログ転送先の設定が入ることで結合度が高くなる
- ログサービスに対する認証情報をアプリケーション側で持つ必要がある

Metrics Logs Traces

Logs

2.システムログ

Kubernetesのシステムコンポーネント、準システムコンポーネントは主に以下の方法で稼働しているケースが多いので、それに合わせた方法で取得

- **Namespace「kube-system」上にPodとして稼働**
  - Podとして起動している場合は、基本的にDaemonSetによるログエージェントで取得
- **Node上にsystemdとして稼働**
  - Node側でログを取得
- **ユーザが確認できないブラックボックスで稼働**
  - マネージドサービスの場合、Control Planeのコンポーネントはユーザからアクセスできないので、サービス提供側が提示する方法で取得

Metrics Logs Traces

Logs

3.監査ログ

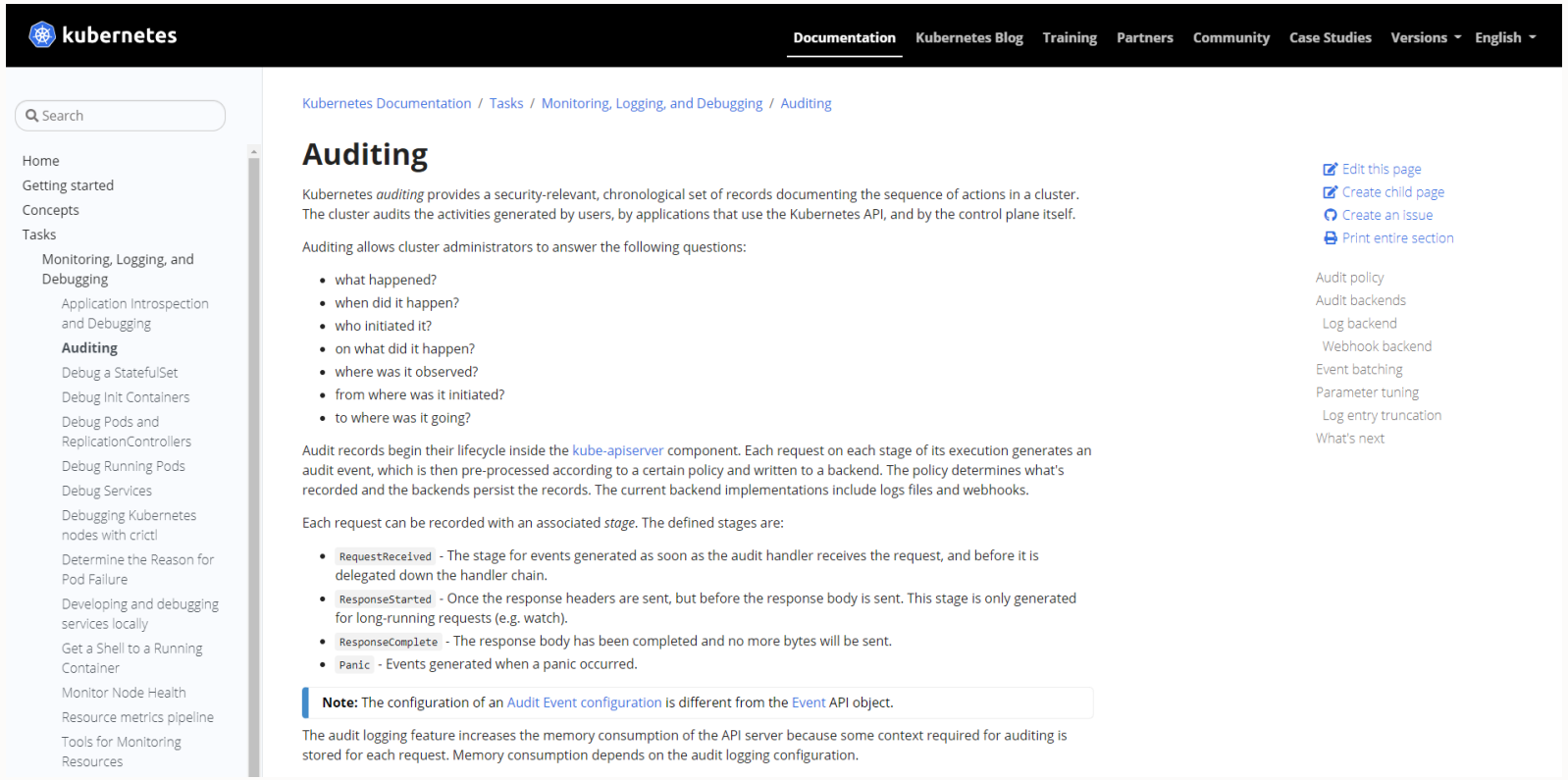
- 監査ログは、Kubernetes のシステムコンポーネント（kube-apiserver）に対してどこに出力するかを設定して取得可能
- 監査ログの主な設定
  - どのリソースに対する操作の監査
  - 「リクエスト受信時」「レスポンス開始時」「レスポンス完了後」「エラー発生後」などのフェーズによる監査
  - 「メタデータのみ」「リクエスト」「リクエストとレスポンス」などどの程度情報を記録するか

全ての監査ログを取得を試みるとログのデータ量が膨大に増えるので、注意が必要！

Metrics Logs Traces

Logs

3.監査ログ



Auditing | https://kubernetes.io/docs/tasks/debug-application-cluster/audit/

Metrics Logs Traces

Logging（ロギング）

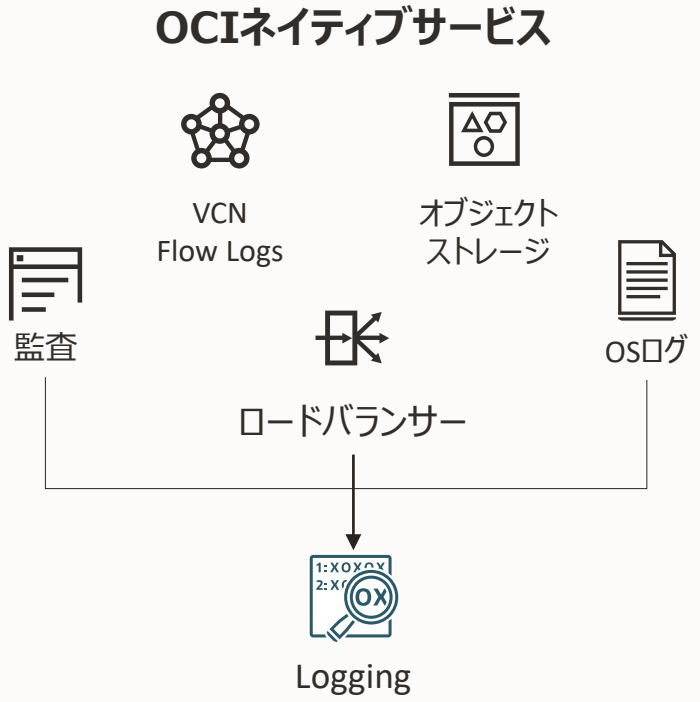
OCIのリソースからログにアクセス

■ ユースケース

- Oracle Cloud Infrastructureリソースへのアクセスを有効にして、問題のデバッグとトラブル・シューティング。
- OCI、オンプレミス、または他社クラウド環境で実行されているアプリケーションからログの取り込み。
- Notifiationサービスと連携して通知したり、サーバーレスのfunctionを呼び出して、アプリケーションの問題を復旧。

■ 特徴

- ログエージェントにFluentd、データ形式にcloudevents、CNCFのオープンソース標準仕様を採用
- 監査ログ、サービスログ、カスタムログの3種類のログを生成し、管理することが可能
- 監査ログは、最大365日間、サービスログとカスタムログは最大6か月間、Loggingサービスに保持、それ以上保持したい場合はObject Storageに転送保持が可能
- Logging Analyticsと連携して、高度なログ分析やアラート、ビジュアライゼーションが可能



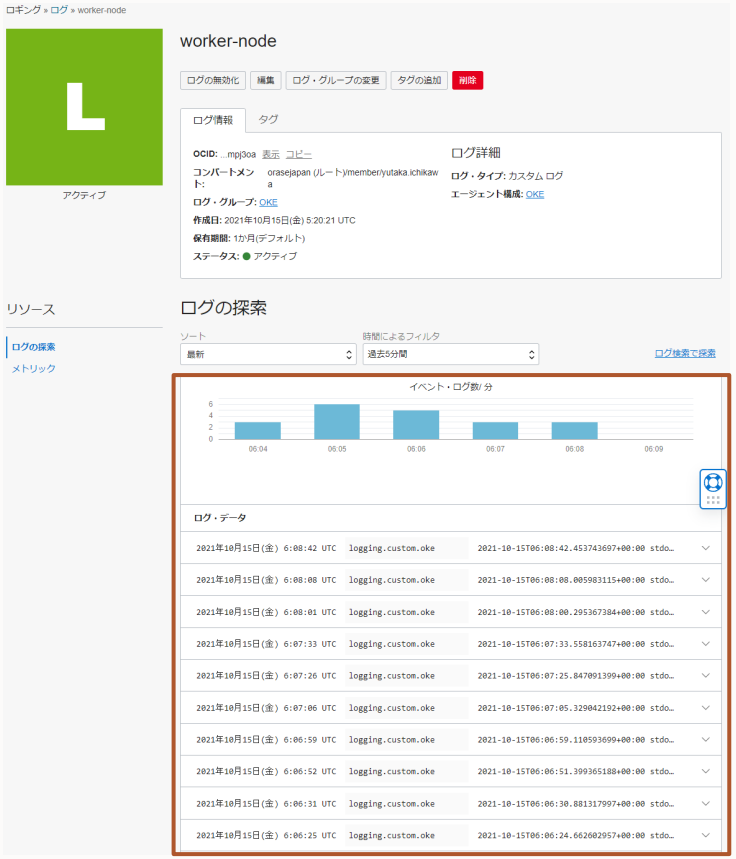
■ 価格

- 1GBあたり\6(時間単位)
- 月額10GBの無料利用枠

Metrics Logs Traces

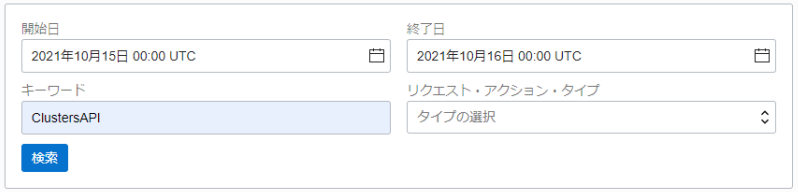
Logging

カスタム・ログ

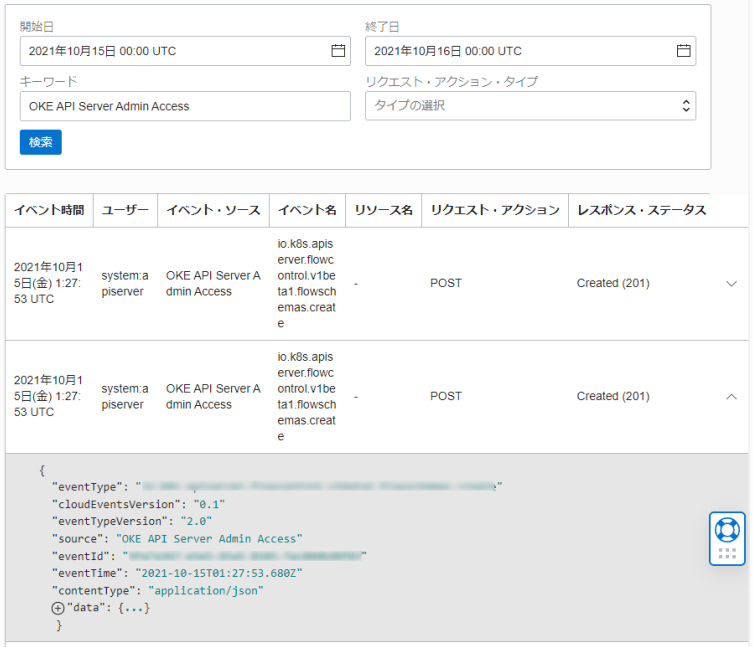


ワーカーノード上で実行されているアプリケーション(Pod)の実行ログ

監査ログ



OKEによって実行された操作のログを表示



Kubernetes APIサーバーによって実行された操作のログを表示

Traces

Metrics Logs Traces

Traces

What’s traces ?

コンポーネント間を跨ぐイベントまたはトランザクションの因果連鎖の指標

トレースが必要となる背景

分散システム、マイクロサービスにおけるメトリクスとログの限界

- 1リクエストが複数のシステムやサービスを跨ぐ状況がある
- リクエストがどのシステムまたはサービスのどのアプリケーションで稼働しているのか不明瞭

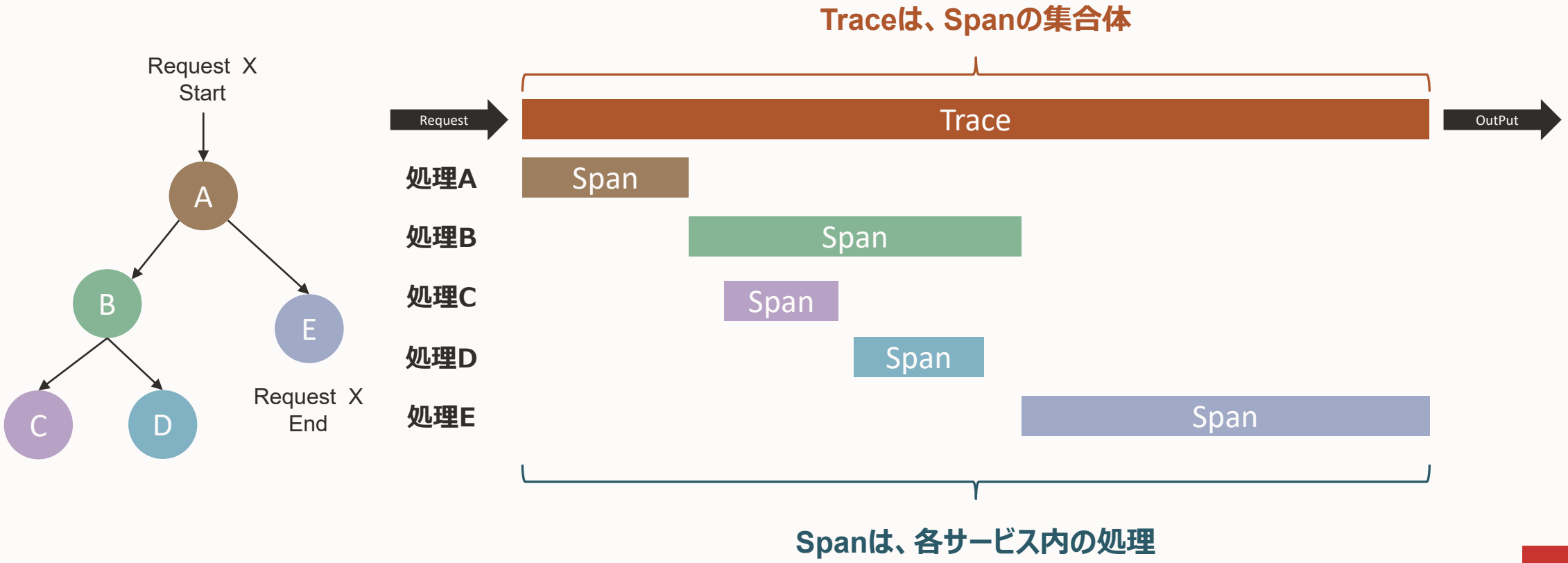
リクエスト、トランザクションのようなある範囲内のイベントを記録して、追跡を可能にする必要がある！

Metrics Logs Traces

Traces

Trace & Span

トレーシング（分散トレーシング）は、TraceとSpanを可視化して、問題個所を特定する



Metrics Logs Traces

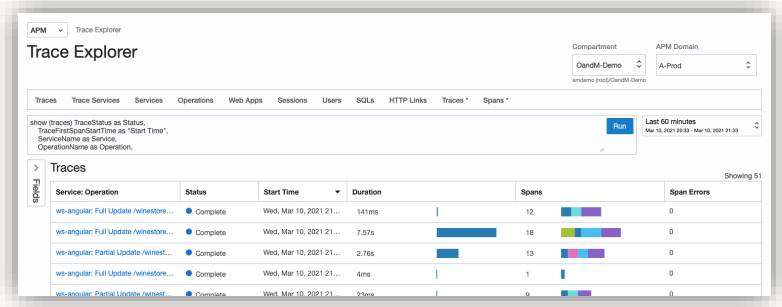
Application Performance Monitoring

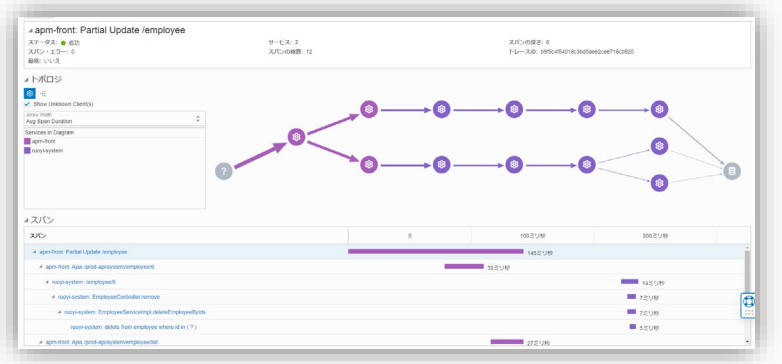
Application Performance Monitoring

アプリケーションのパフォーマンスを可視化し、サーバーおよびユーザー・サイドからの問題の根本的原因を分析

■ 特徴

- トレース・エクスプローラー
  - ✓マイクロサービスのアプリケーションのトレース情報の分析に特化した分散トレーシング
  - ✓OpenTracing、Open Telemetry互換
- リアル・ユーザー・モニタリング
  - ✓エンドユーザーのパフォーマンスを可視化するエンドユーザー監視
- 合成モニタリング
  - ✓アプリケーションの死活監視を行う合成モニタリング
- サーバー監視
  - ✓サーバの可用性、負荷、パフォーマンスを監視するサーバー監視
- ダッシュボード
  - ✓O&Mの他のサービスのデータとAPMのデータを組み合わせて、ダッシュボードを作成、カスタマイズすることが可能





■ 価格

Tracing Data: 100,000 イベント \78（時間単位）

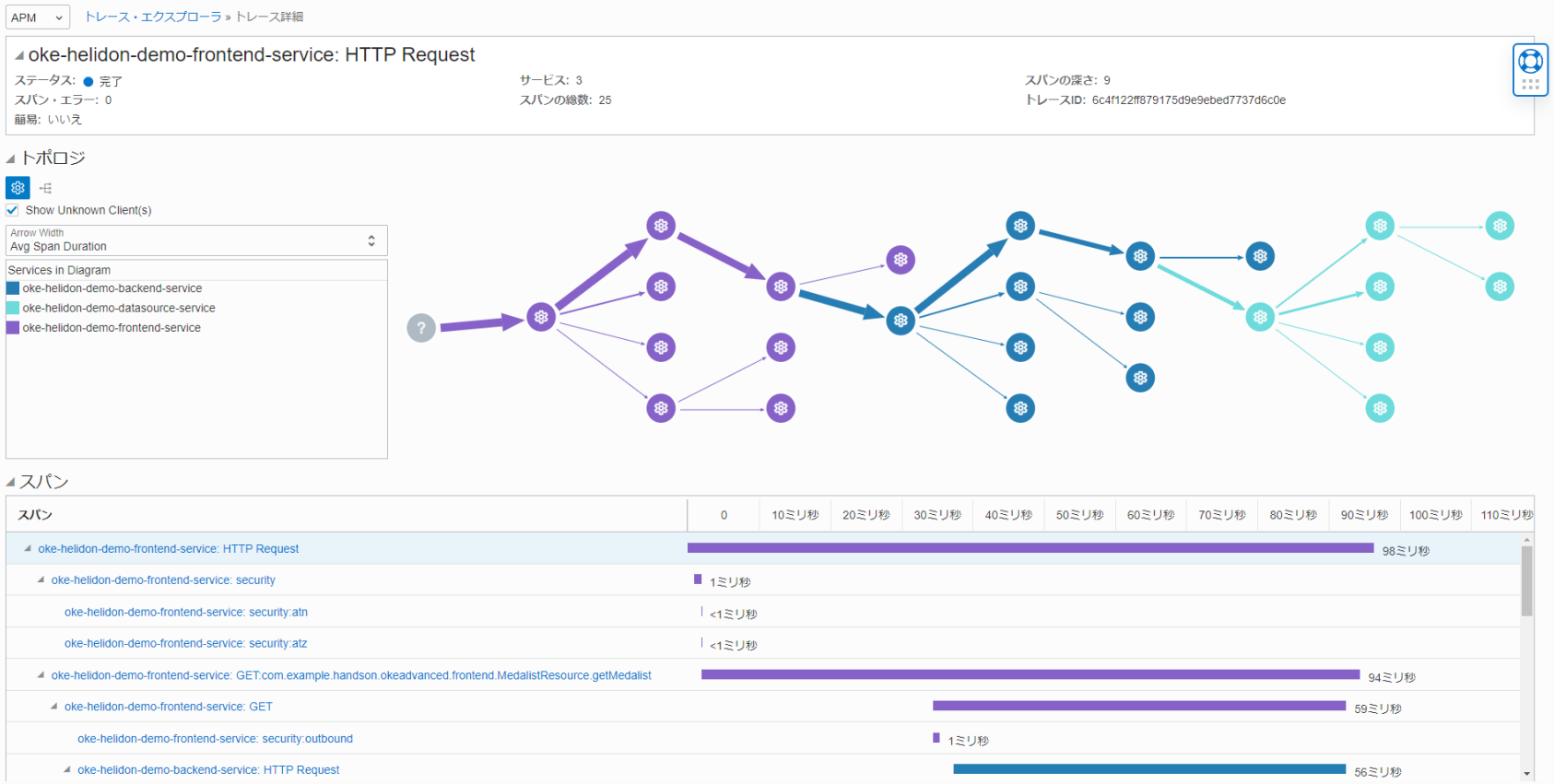
Synthetic usage: 10 モニター稼働 \2.4（時間単位）

Metrics Logs Traces

Application Performance Monitoring

トレース・エクスプローラ

サンプルマイクロサービスアプリケーションのリクエストの軌跡をEnd to Endで可視化



Metrics Logs Traces

Application Performance Monitoring

**分散したシステムで構成されているマイクロサービスで、
リクエストの軌跡をEnd to Endで可視化**

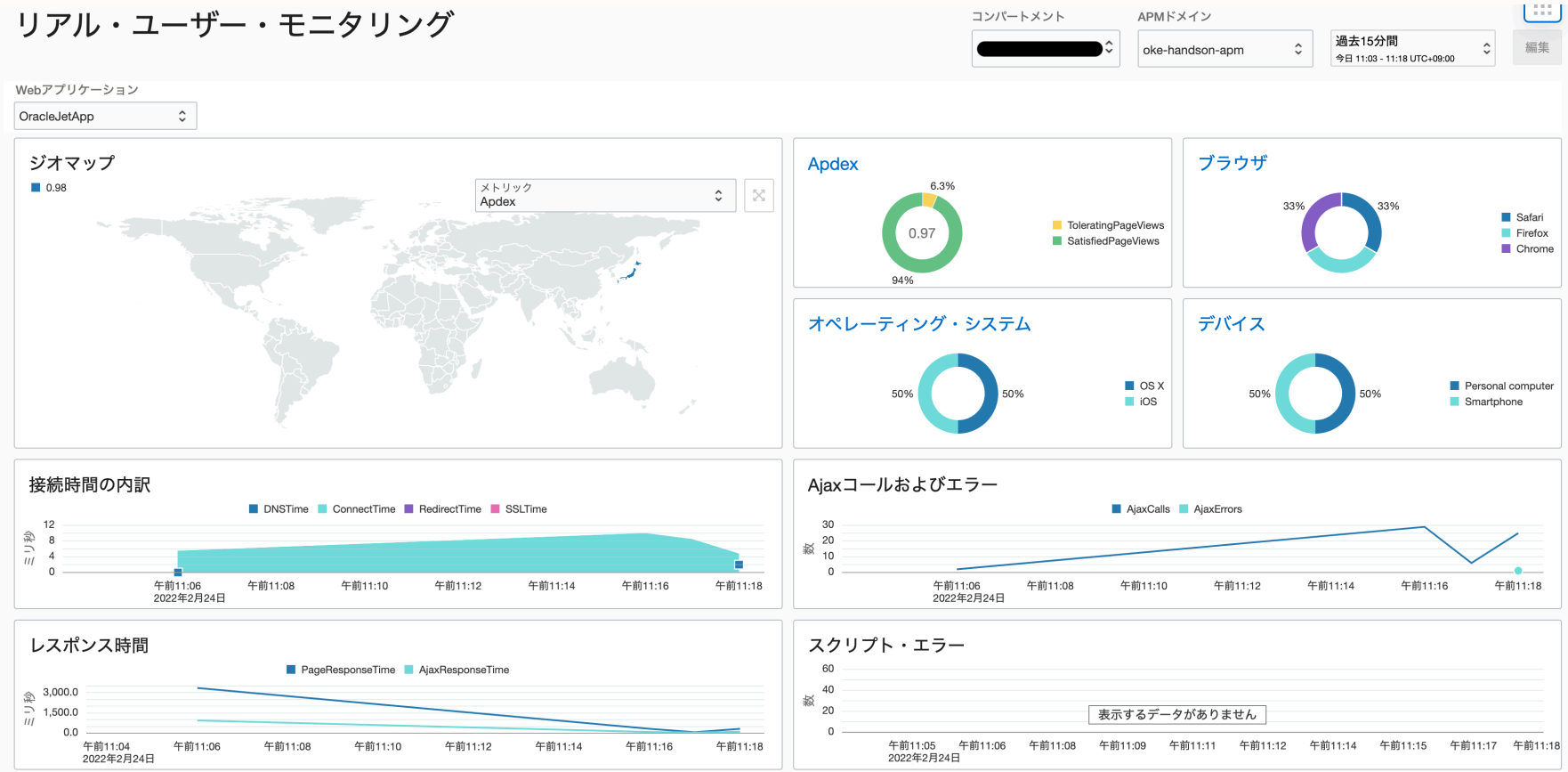
- **全てのトランザクション（トレース）の全ての処理（スパン）を確認することが可能**
- **トレースに関する以下の情報を参照することが可能**
  - 低速のトランザクション
  - エラーのあるトランザクション
  - サーバー・データセンター・バージョンなどの特定のリソースを使用するトランザクション
- **スパンに関する以下の情報を参照することが可能**
  - ページの読み込み
  - Ajax呼出し
  - サービスリクエスト
  - JDBC
  - HTTPアウトバウンド呼出し
  - 応答時間 etc

Metrics Logs Traces

Application Performance Monitoring

リアル・ユーザー・モニタリング

Webページのパフォーマンスを実ユーザのPCもしくはスマートフォンからのアクセスを元に計測・分析する



Metrics Logs Traces

Application Performance Monitoring

エンドユーザーのリアルな反応を可視化し、
優れたエンドユーザー・エクスペリエンスの提供を支援

- **Javascriptのブラウザーエージェントによる監視**
  - アプリケーションのソースコードにJavascriptを埋め込むことで監視が可能
- **各ユーザーセッションをEnd to Endで追跡**
- **Webアプリケーション、およびユーザー・エクスペリエンスのパフォーマンスを把握**
  - Apdexスコア
  - レスポンス時間
  - スクリプト・エラー
  - Ajaxコールとエラー
- **Webアプリケーションの速度とパフォーマンスを測定することで、ユーザー・エクスペリエンスを向上させるための修正処理を実行が可能**

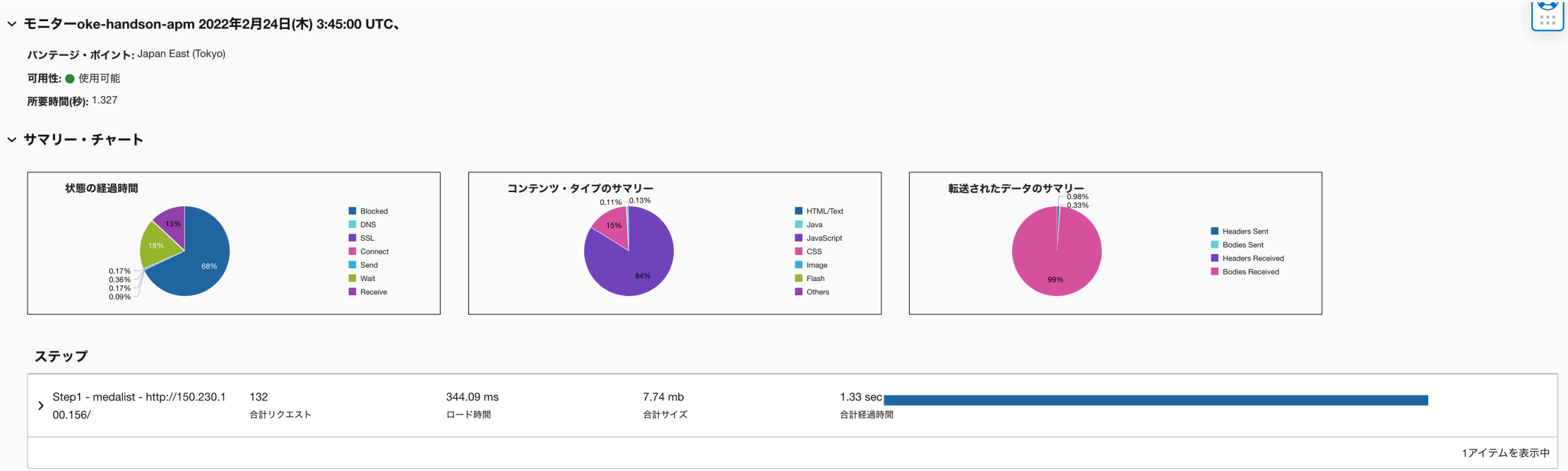
Metrics Logs Traces

Application Performance Monitoring

合成モニタリング

アプリケーションの可用性を監視し、エンドユーザーに影響を与える前に可用性に関する問題を特定

地理的に分散したエージェントを使用して、能動的に対象のWebサイトにアクセスして計測する



Metrics Logs Traces

Application Performance Monitoring

アプリケーションの可用性を監視し、
エンドユーザーに影響を与える前に可用性に関する問題を特定することが可能

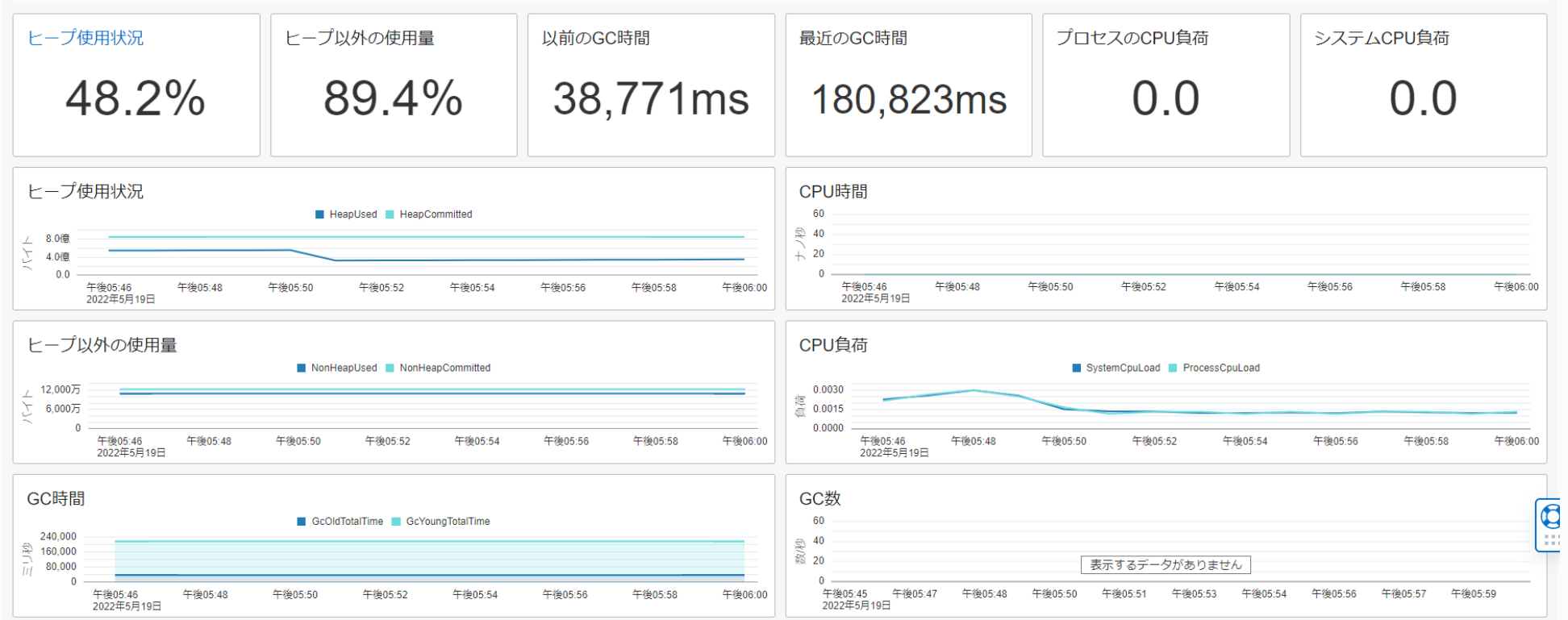
- **シンセティック監視の実行結果の概要を表示**
- **モニターを定期的に実行**
  - スクリプト・ブラウザー・モニター(Selenium）
  - ブラウザー・モニター
  - スクリプト・RESTモニター(Postman）
  - RESTモニター
- **アプリケーションの可用性を確認することが可能**
  - 成功または失敗したモニター実行
  - クリティカルなバンテージ・ポイントでのモニター実行
  - ロード時間
- **モニター実行結果はメトリックとしても収集される**
- **メトリックを表示することで、モニターの可用性、確立された接続の数、エラーの有無を即座に確認することも可能**

Metrics Logs Traces

Application Performance Monitoring

サーバ監視

アプリケーション・サーバーのヒープおよびCPUメトリックに関する情報を収集



Metrics Logs Traces

Application Performance Monitoring

アプリケーション・サーバーのヒープおよびCPUメトリックに関する情報を収集

- **メトリックはAPMダッシュボードおよびMonitoringサービスから表示することが可能**
  - 平均ページ読み込み時間とApdex
  - リクエストサービス毎にグループ化された平均レスポンス時間
  - ロケーションごとのページグループ単位のリクエストレート
  - サードパーティサービスの呼出しのカウントと最大応答時間
- **Monitoringサービス同様、カスタムメトリックの作成も可能**

デモンストレーション概要

デモンストレーションについて

デモンストレーション概要

以下サイトにある手順をベースに進めていきます。

『Oracle Container Engine for Kubernetes(OKE)でサンプルマイクロサービスアプリケーションをデプロイしてOCIのオブザバビリティサービスを利用してみよう』

https://bit.ly/oracle-o11y

利用するサンプルアプリケーションは、以下GitHubにあります。

https://github.com/oracle-japan/code-at-customer-handson

デモンストレーションについて

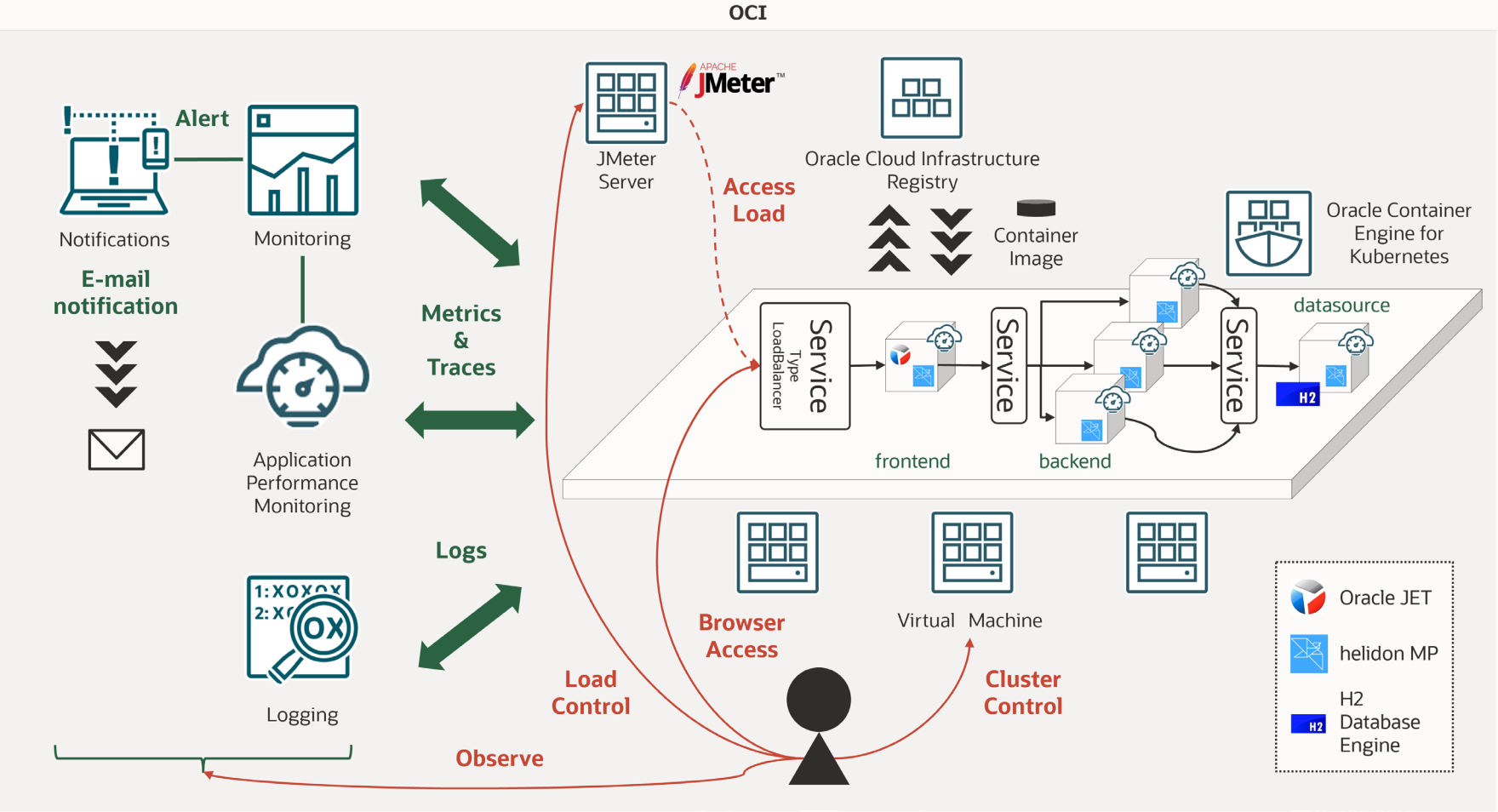
デモンストレーション概要

『Oracle Container Engine for Kubernetes(OKE)でサンプルマイクロサービスアプリケーションをデプロイしてOCIのオブザバビリティサービスを利用してみよう』

1. **OKEクラスタ構築とOCIRセットアップ**
    1. OCIダッシュボードからOKEクラスタの構築
    2. Cloud Shellを利用してクラスタを操作
    3. OCIRのセットアップ
2. **Application Performance Monitoring**
    1. サンプルアプリケーションの概要説明
    2. サンプルアプリケーションとAPM連携設定
    3. APMドメインの作成
    4. サンプルアプリケーションへのAPM設定(ブラウザ側)とコンテナイメージ作成
    5. サンプルアプリケーションへのAPM設定(サーバサイド側)
    6. OCI APMでのトレーシング
    7. OCI APMでのアプリケーションサーバのメトリクス監視
    8. OCI APMでのリアルユーザモニタリング(RUM)
    9. OCI APMでの合成モニタリング(Synthetic Monitoring)
3. **Logging**
    1. カスタム・ログの設定
    2. ワーカーノード上のアプリケーションログの確認
    3. Kubernetes APIサーバーの監査ログの確認
4. **Monitoring & Notifications**
    1. Notificationsの設定
    2. Monitoringの設定
    3. MonitoringとNotificationsの実践
5. **今回利用したサンプルアプリケーションの補足説明**

デモンストレーションについて

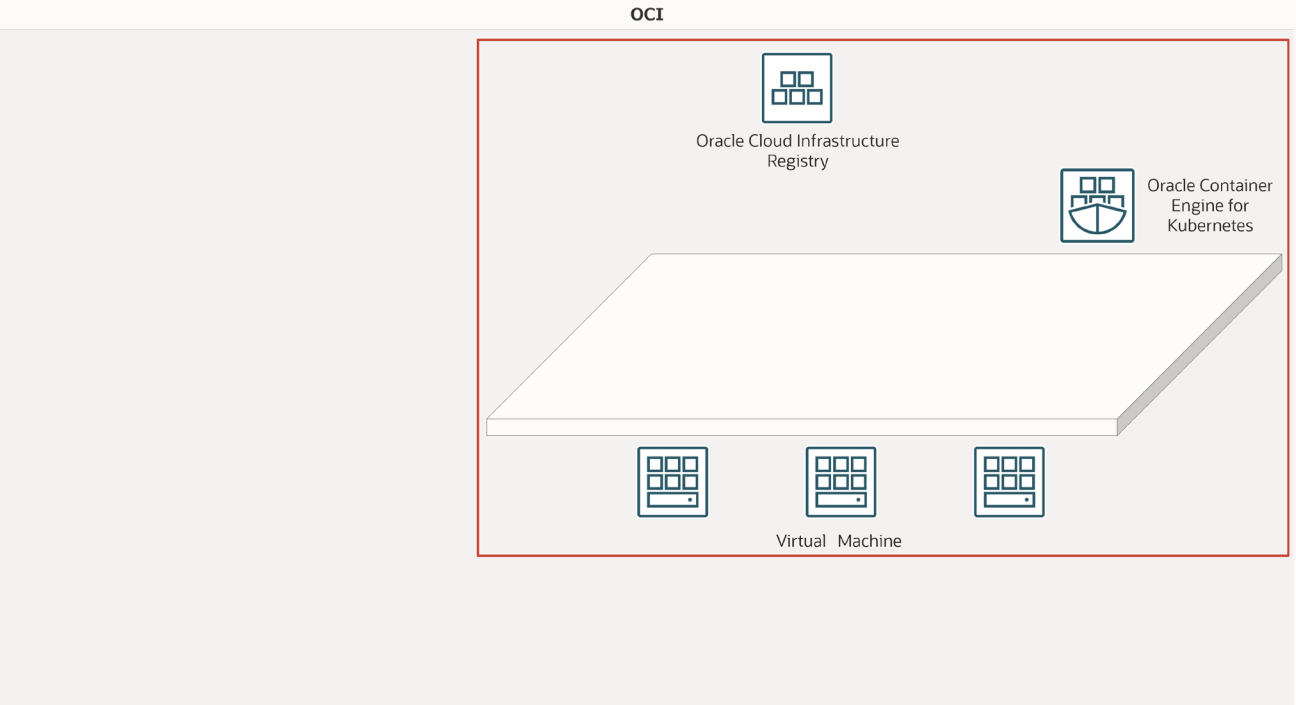
デモンストレーション概要



デモンストレーションについて

デモンストレーション概要

1. **OKEクラスタ構築とOCIRセットアップ**
    1. OCIダッシュボードからOKEクラスタの構築
    2. Cloud Shellを利用してクラスタを操作
    3. OCIRのセットアップ



デモンストレーションについて

Container Engine
For Kubernetes

Oracle Container Engine for Kubernetes

高可用性と開発生産性を両立するKubernetesプラットフォーム

■ユースケース

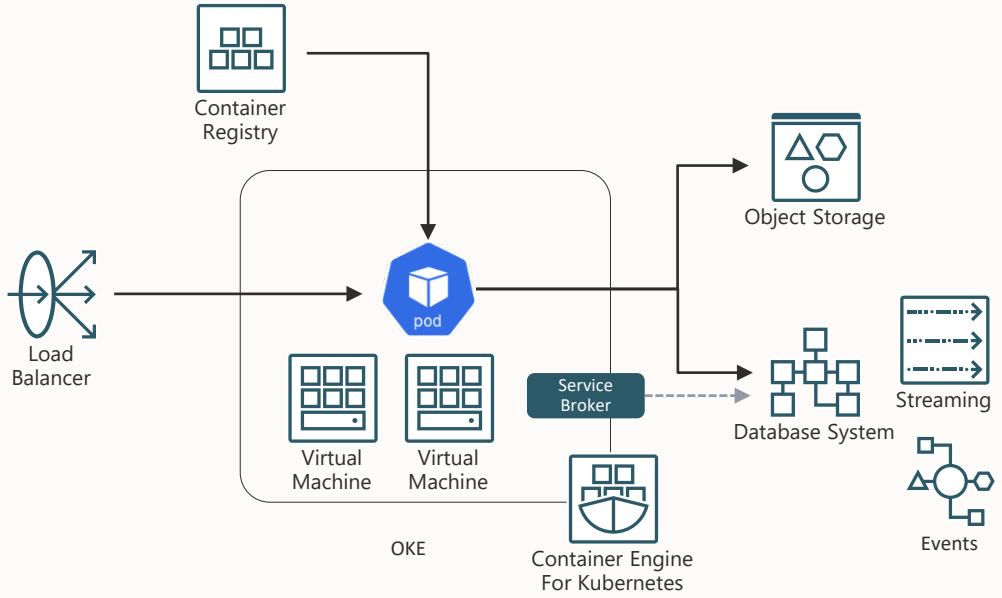
- コンテナ化されたアプリケーションを迅速かつ簡単にデプロイおよび管理が可能

■特徴

- コンソール上で迅速なクラスタ作成および管理が可能
- 仮想サーバー/ベアメタルサーバー/HPC/GPUを選択可能
- 従来のx86に加えて、Armを加えたマルチアーキテクチャをサポート
- OCI DevOps（CI/CDサービス）と連携することによるシームレスなビルド、テスト及びデプロイが可能

■価格

- Kubernetesのコントロールプレーン(Master Node等)は課金対象外
- Oracle Cloud Infrastructure(Compute/Block Volume/Network/Load Balancer)利用分のみ課金



■ 関連するOracle Cloud Service

- OCI Registry (OCIR)

デモンストレーションについて

Container Registry

Oracle Cloud Infrastructure Registry (OCIR)

コンテナ・イメージを管理するプライベート・レジストリ

■ユースケース

DockerやKubernetesで使用するコンテナ・イメージをセキュアに保管、管理するための
プライベート・レジストリ

■特徴

Docker v2対応のコンテナレジストリサービス

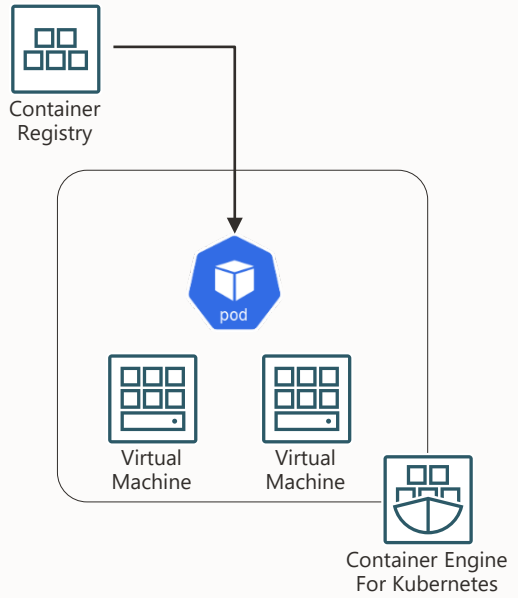
OKEと同一リージョンに展開することで低レイテンシ

■価格

Oracle Cloud Infrastructure(Storage/Network)利用分のみ課金

■ 関連するOracle Cloud Service

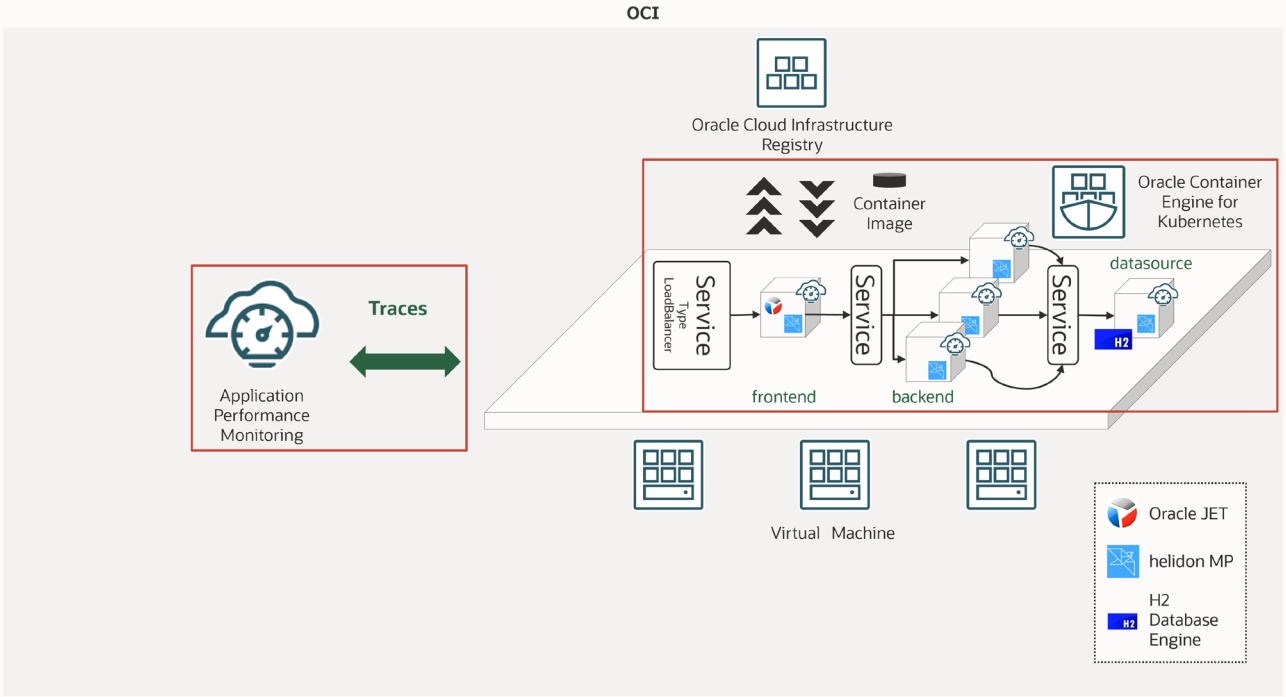
Oracle Container Engine for Kubernetes (OKE)



デモンストレーションについて

デモンストレーション概要

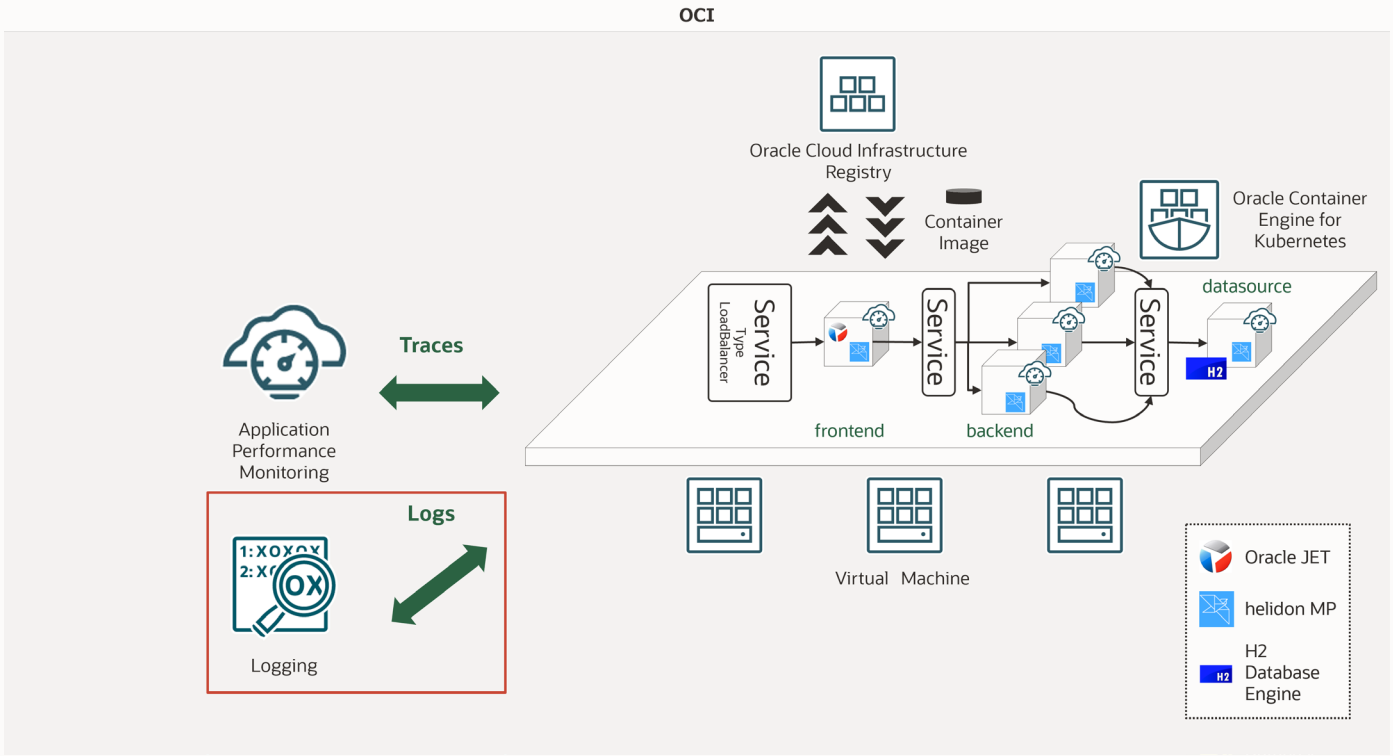
2. **Application Performance Monitoring**
    1. サンプルアプリケーションの概要説明
    2. サンプルアプリケーションとAPM連携設定
    3. APMドメインの作成
    4. サンプルアプリケーションへのAPM設定(ブラウザ側)とコンテナイメージ作成
    5. サンプルアプリケーションへのAPM設定(サーバサイド側)
    6. OCI APMでのトレーシング
    7. OCI APMでのアプリケーションサーバのメトリクス監視
    8. OCI APMでのリアルユーザモニタリング(RUM)
    9. OCI APMでの合成モニタリング(Synthetic Monitoring)



デモンストレーションについて

デモンストレーション概要

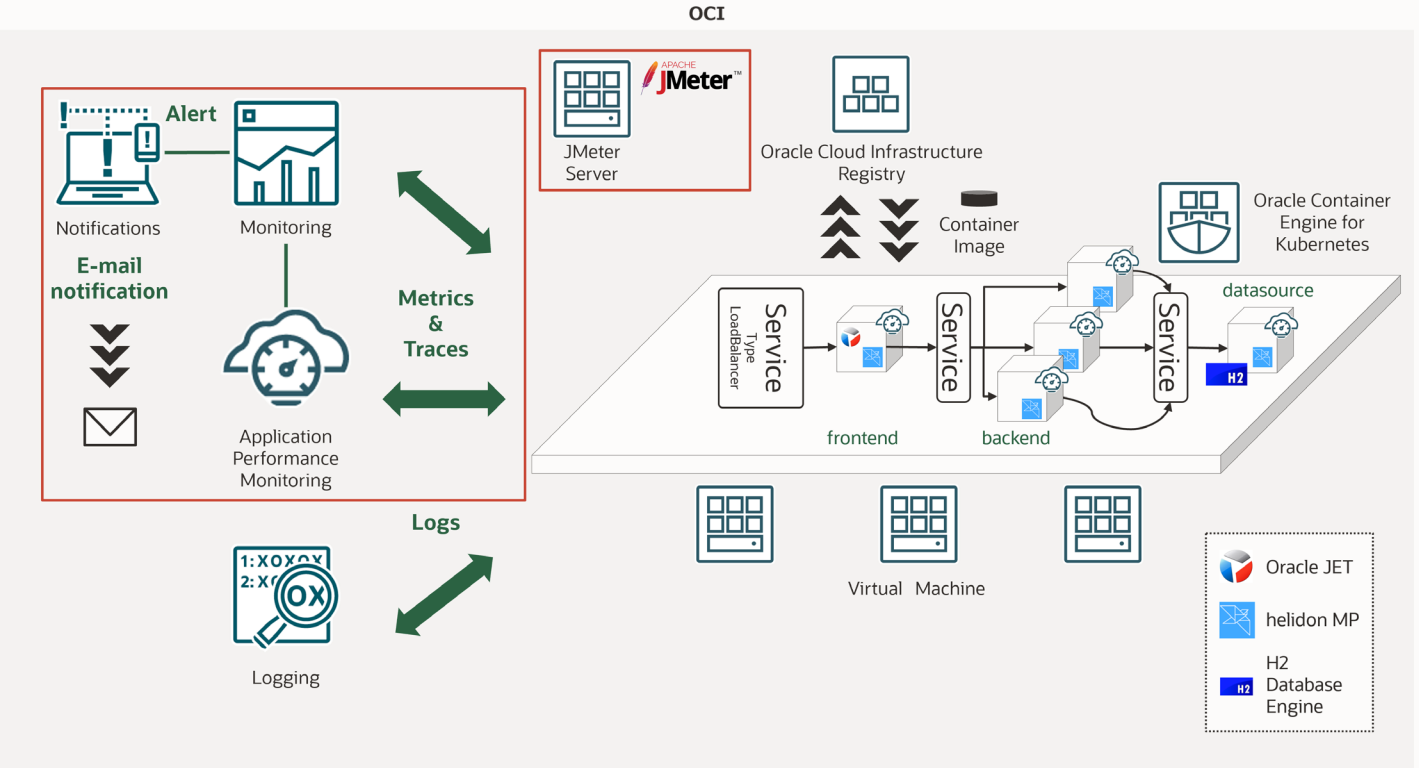
3. **Logging**
    1. カスタム・ログの設定
    2. ワーカーノード上のアプリケーションログの確認
    3. Kubernetes APIサーバーの監査ログの確認



デモンストレーションについて

デモンストレーション概要

4. **Monitoring & Notifications**
    1. Notificationsの設定
    2. Monitoringの設定
    3. MonitoringとNotificationsの実践



参考資料

参考資料

Observabilityをはじめよう！(前編) ～Observabilityの背景と構成要素～
https://knowledge.sakura.ad.jp/26395/

Observabilityをはじめよう！(後編) ～Metrics/Logs/Tracesチュートリアル～
https://knowledge.sakura.ad.jp/26409/

分散トレーシングの歴史、計装、そしてその活用プラクティス
https://event.cloudnativedays.jp/cndt2021/talks/1289

Ochacafe資料
https://speakerdeck.com/oracle4engineer/oracle-cloud-hangout-cafe-observabilityzai-ru-men

Oracle Hangout Cafe Season4 #6 Observability再入門
https://speakerdeck.com/oracle4engineer/oracle-cloud-hangout-cafe-observabilityzai-ru-men
https://www.youtube.com/watch?v=1IGyUR_OIxM&list=PL8x2FJpi0g-sDPoupp2pg6RP9P9JU2fHx&index=2
https://github.com/oracle-japan/ochacafe-s4-6