オブザーバビリティと監視

松浦 隼人

Jul 8, 2022

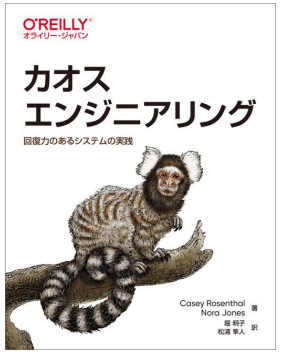
ゴール

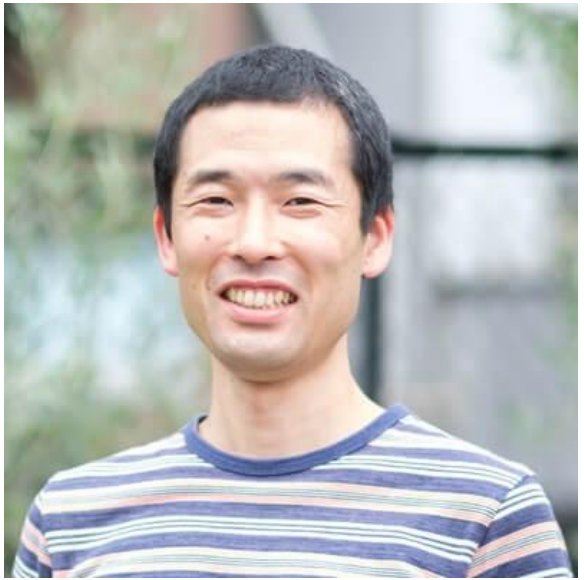
- オブザーバビリティという概念がなんなのか分かる
- 監視とオブザーバビリティの違いが分かる
- なぜ今オブザーバビリティが重要なのかが分かる
- どうやったらオブザーバビリティの高いシステムを作れるのかが分かる

自己紹介

松浦 隼人

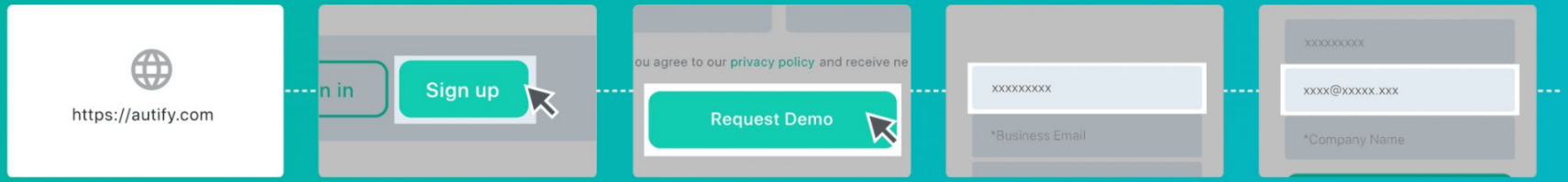
- オーティファイ株式会社
- Twitter : dblmkt
- 経歴
  - インフラ何でも屋
  - 某ブログサービスのインフラ担当
  - テクニカルサポート
  - インフラ + Rails
  - CTO
- 趣味
  - 翻訳

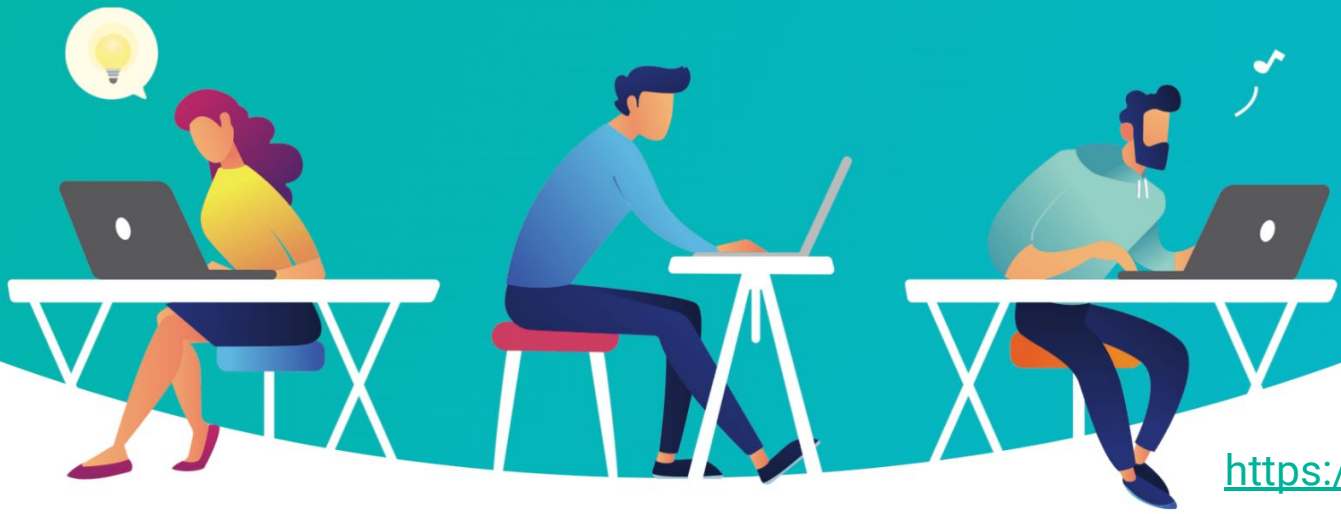




ブラウザ操作を記録するだけで

テストが ノーコード で 誰にでも簡単 に作れます👍





https://autify.com/ja

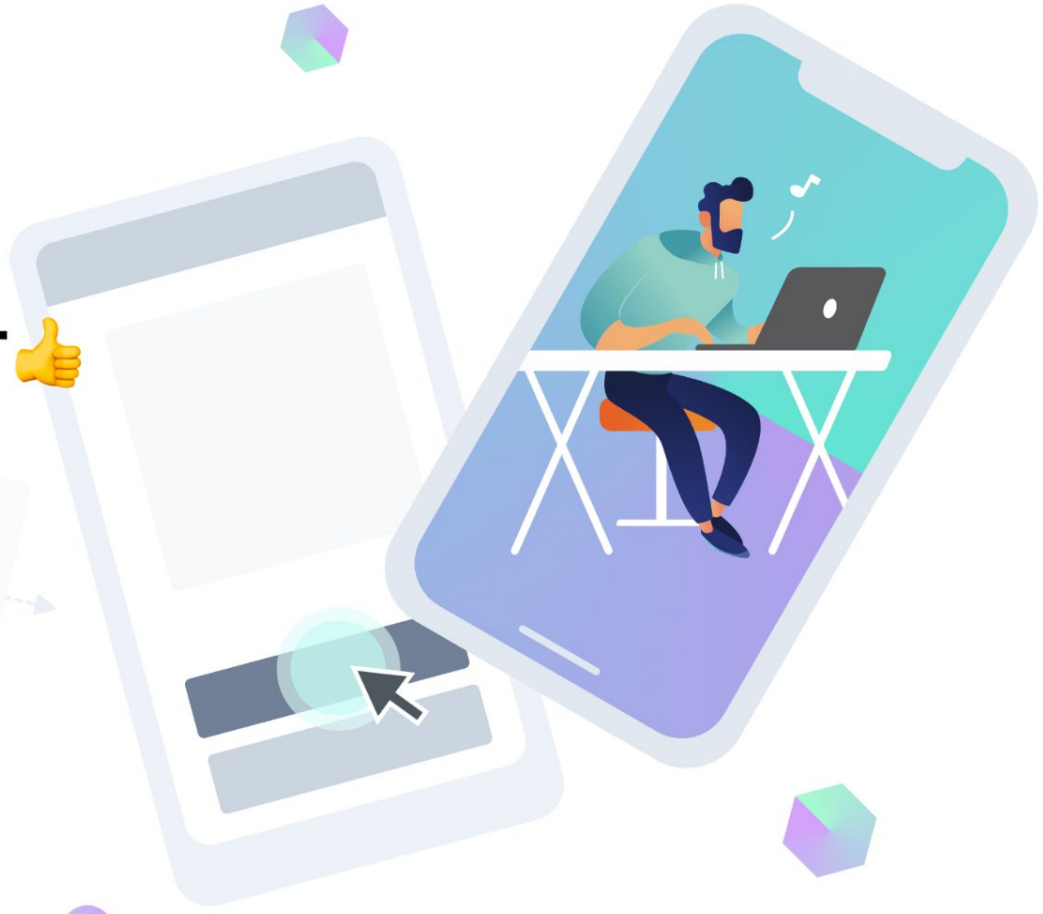
for Mobile

アプリ操作を記録するだけで

テストがノーコードで
誰にでも簡単に作れます👍

Autify for Mobileからアプリを操作するだけで、誰にでも簡単にテストが作成・実行できます。テストのためだけにいくつもの端末実機を用意する必要もありません。

デモを申し込む



https://autify.com/ja/mobile

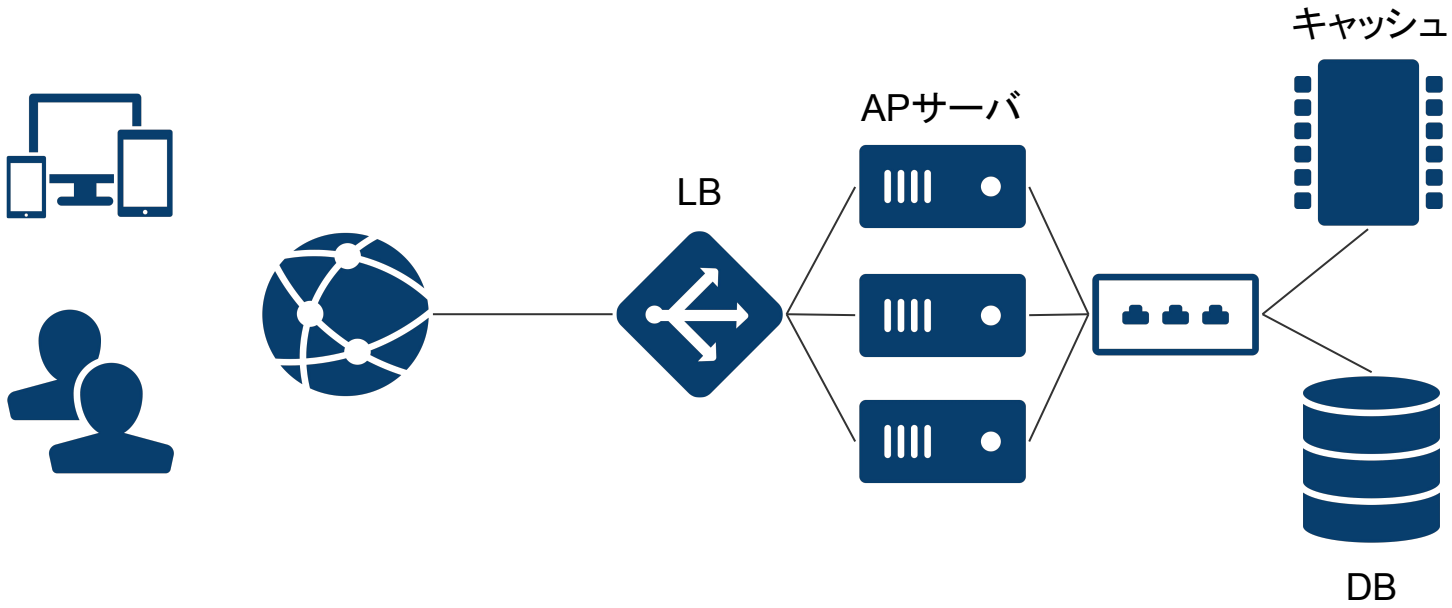
オブザーバビリティとは

(その前に) 監視とは

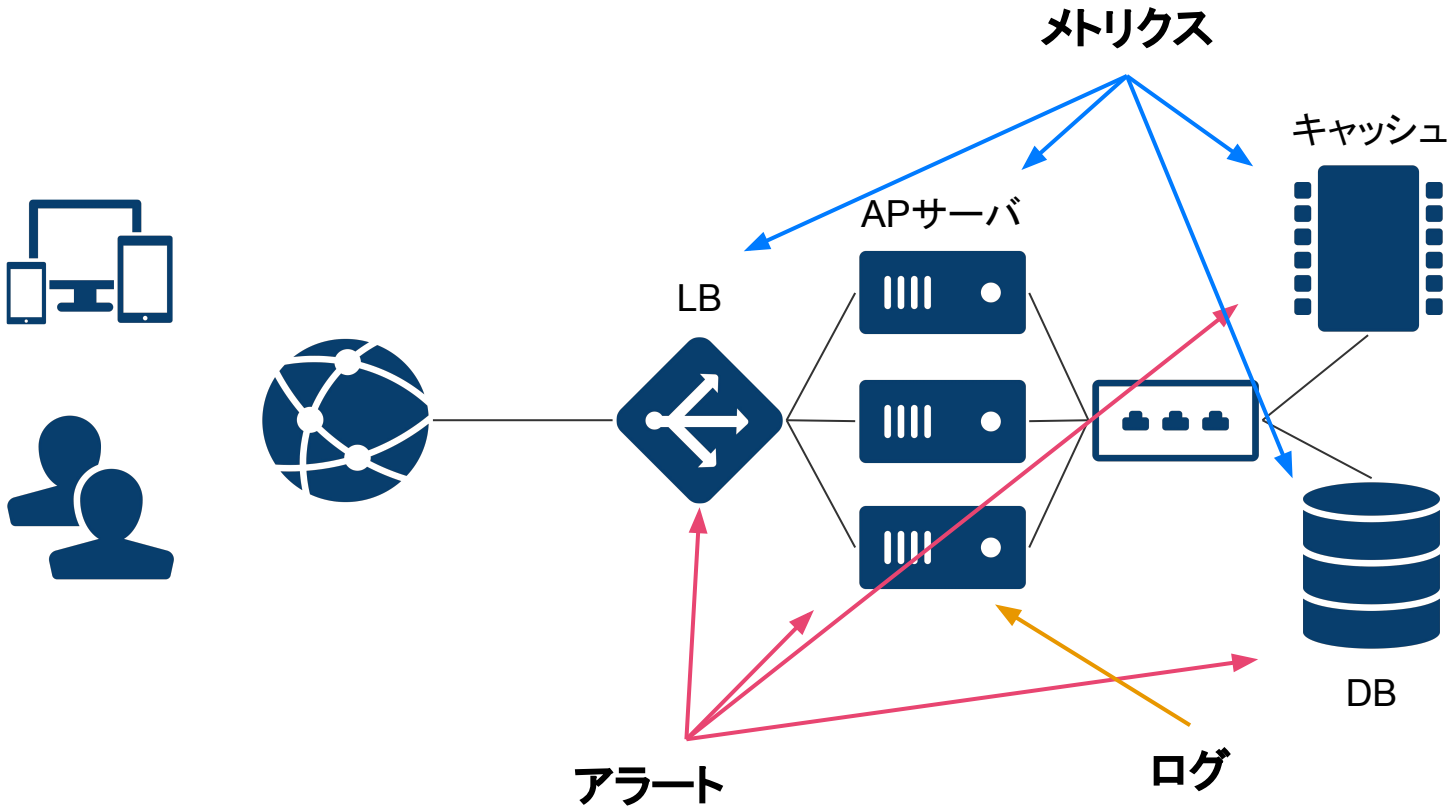
監視とは

- あるシステムやコンポーネントの振る舞いや出力を観察し続けること
  - 障害の検知
  - 問題特定のための情報収集
  - 傾向の把握
    - 改善の必要性の判断

よくあるシステム



よくあるシステム



監視とは

- あるシステムやコンポーネントの振る舞いや出力を観察し続けること
- 任意の点に注目して観察し続ける
- 過去の経験、システムやアクセスの傾向をもとに監視の仕組みを改善していくのが重要

オブザーバビリティとは

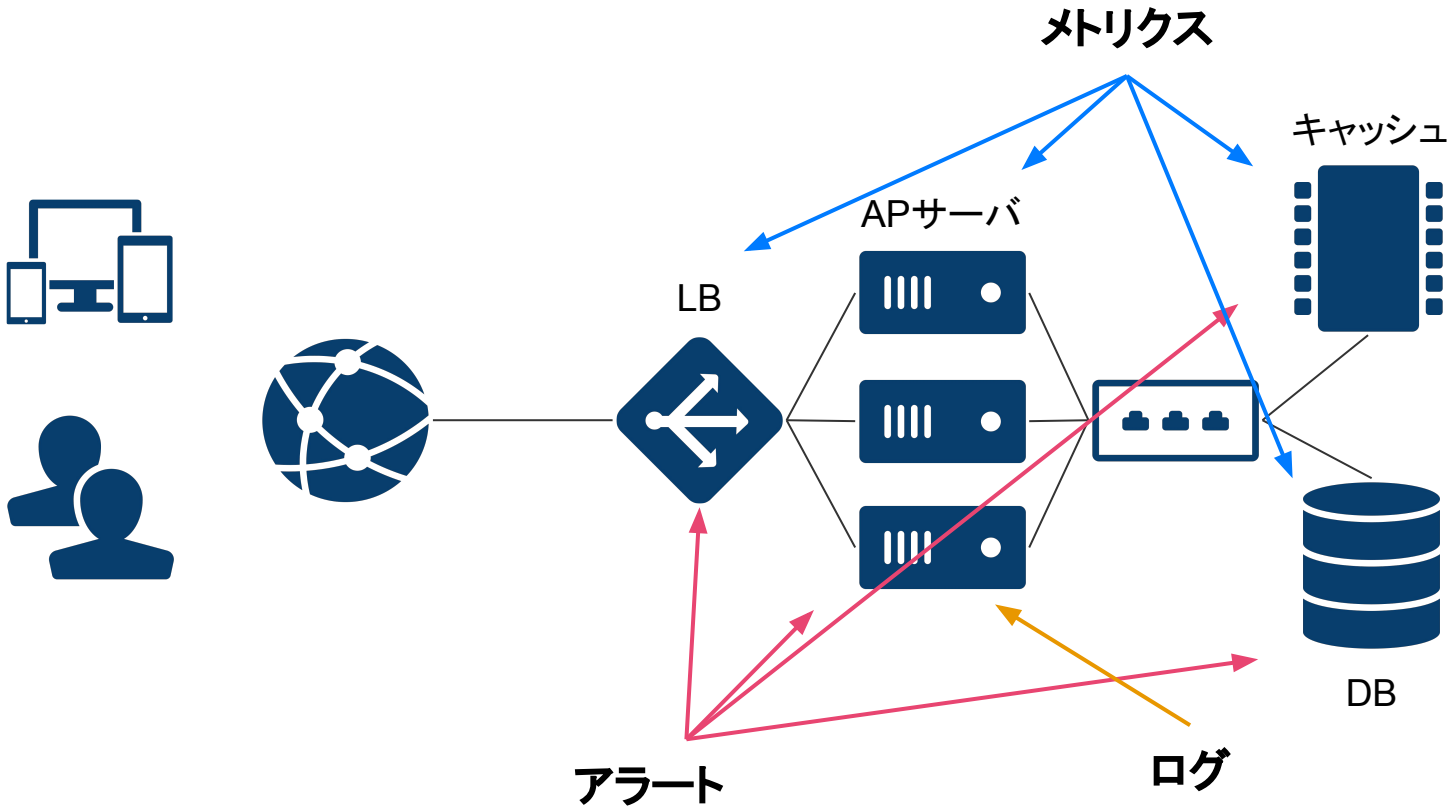
オブザーバビリティとは

- Observability = 可観測性
  - Observe : 観察する、観測する
  - Ability : ～できること
- 元々は制御理論における考え方
- 制御理論におけるオブザーバビリティ
  - ある時間システムの出力を観察すると、観察開始時点のパラメータがわかること

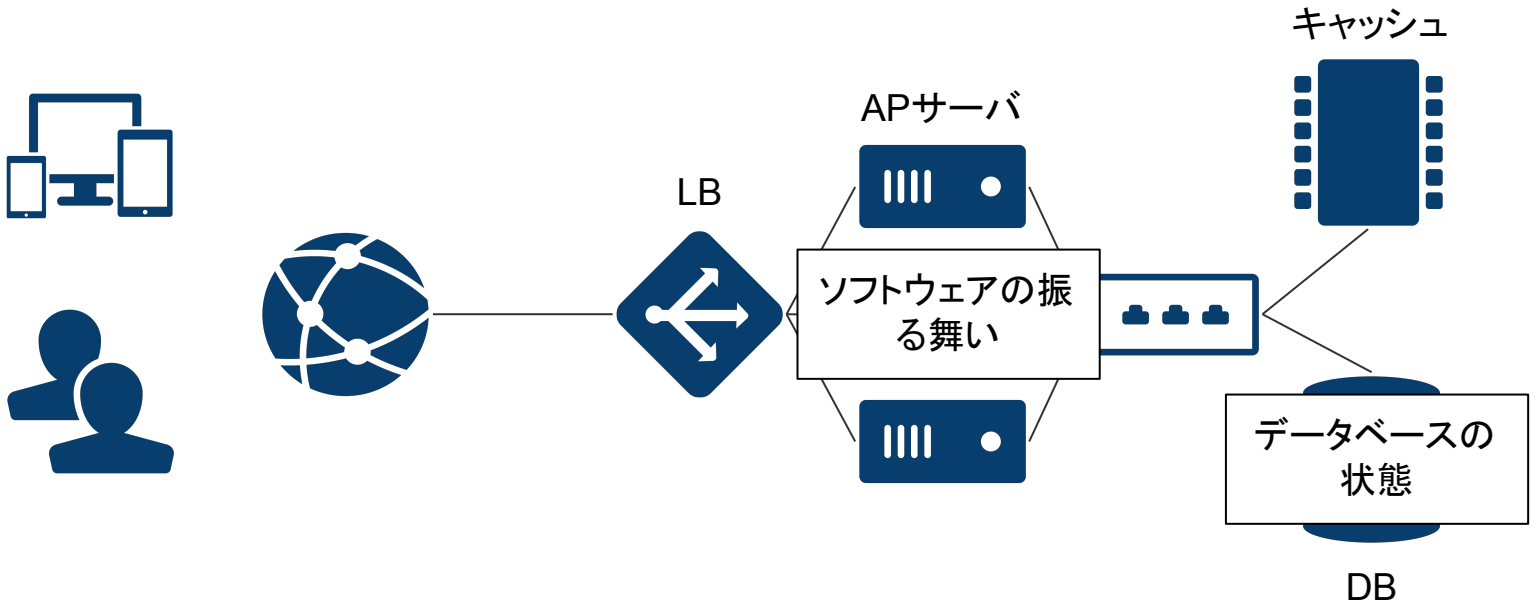
オブザーバビリティとは

- ソフトウェアシステムにおけるオブザーバビリティ
  - システムの振る舞いや内部をどれだけ把握できるかの程度
  - あるいはそれを高めるための取り組み

よくあるシステム



よくあるシステム



オブザーバビリティと監視

オブザーバビリティ	監視
ソフトウェアに焦点	コンポーネントあるいはインフラに焦点
システムの内部を知る	システムを外部から見る

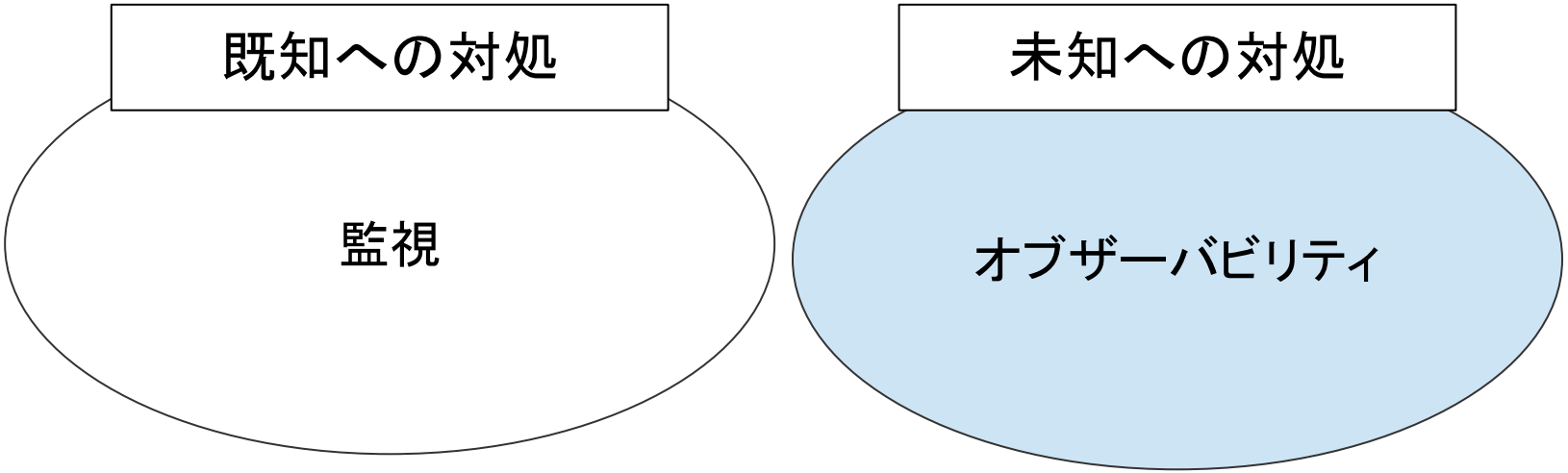
オブザーバビリティがあるシステムの条件

- アプリケーションの内部の動きが把握できる
  - 見たこともないし予測したこともないも含め、アプリケーションがとりうるあらゆる状態(state)がわかる
  - 外部ツールを使って観測したり問い合わせしたりするだけで、システムの内部の動きとその状態がわかる
  - 新しいカスタムコードを追加しなくてもアプリケーションの内部の動きがわかる

オブザーバビリティと監視

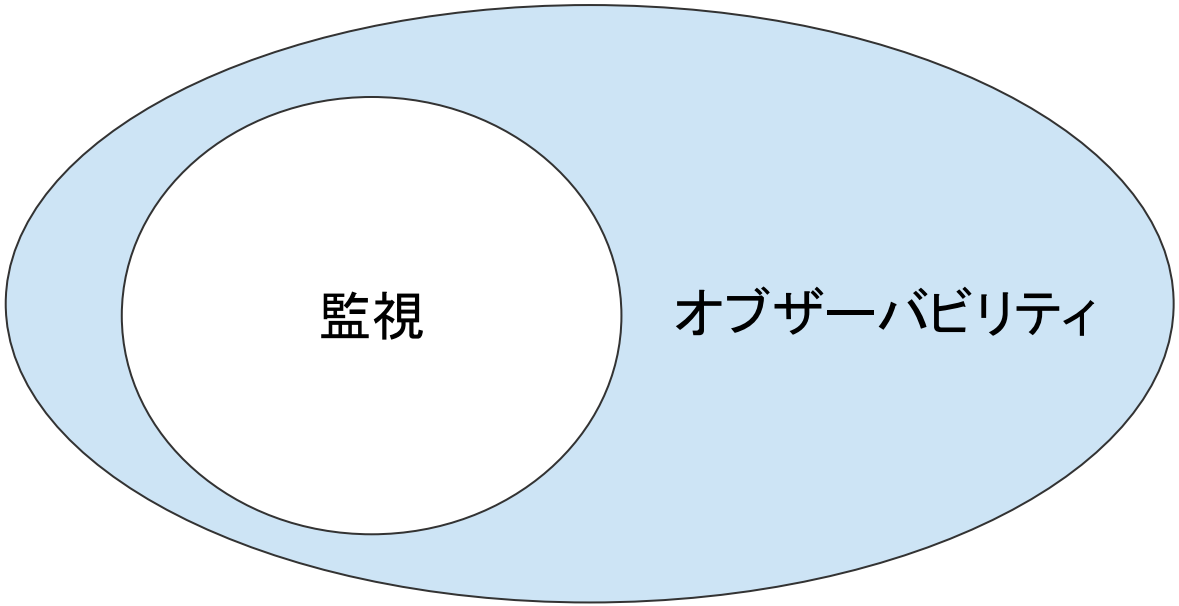
オブザーバビリティ	監視
未知の状況に対応できる	過去の経験(既知)を活かす

監視とオブザーバビリティの関係



監視とオブザーバビリティの関係

- 監視はオブザーバビリティを高める1つの手段でもある

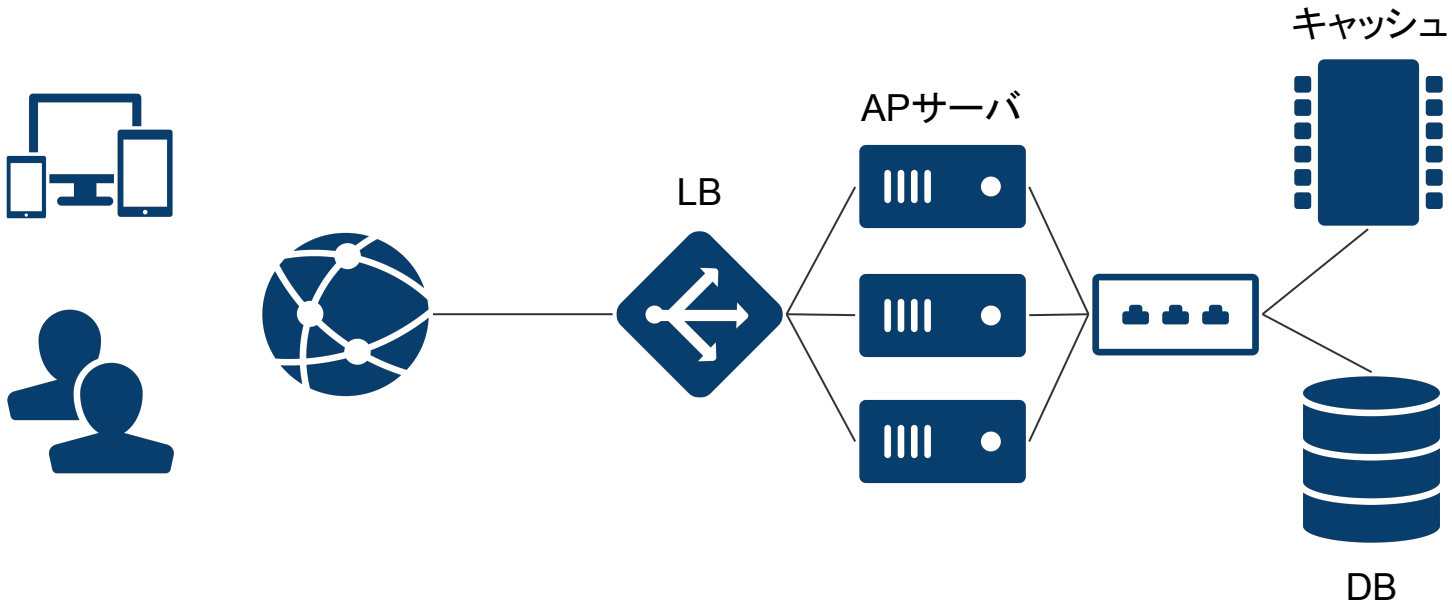


なぜオブザーバビリティなのか

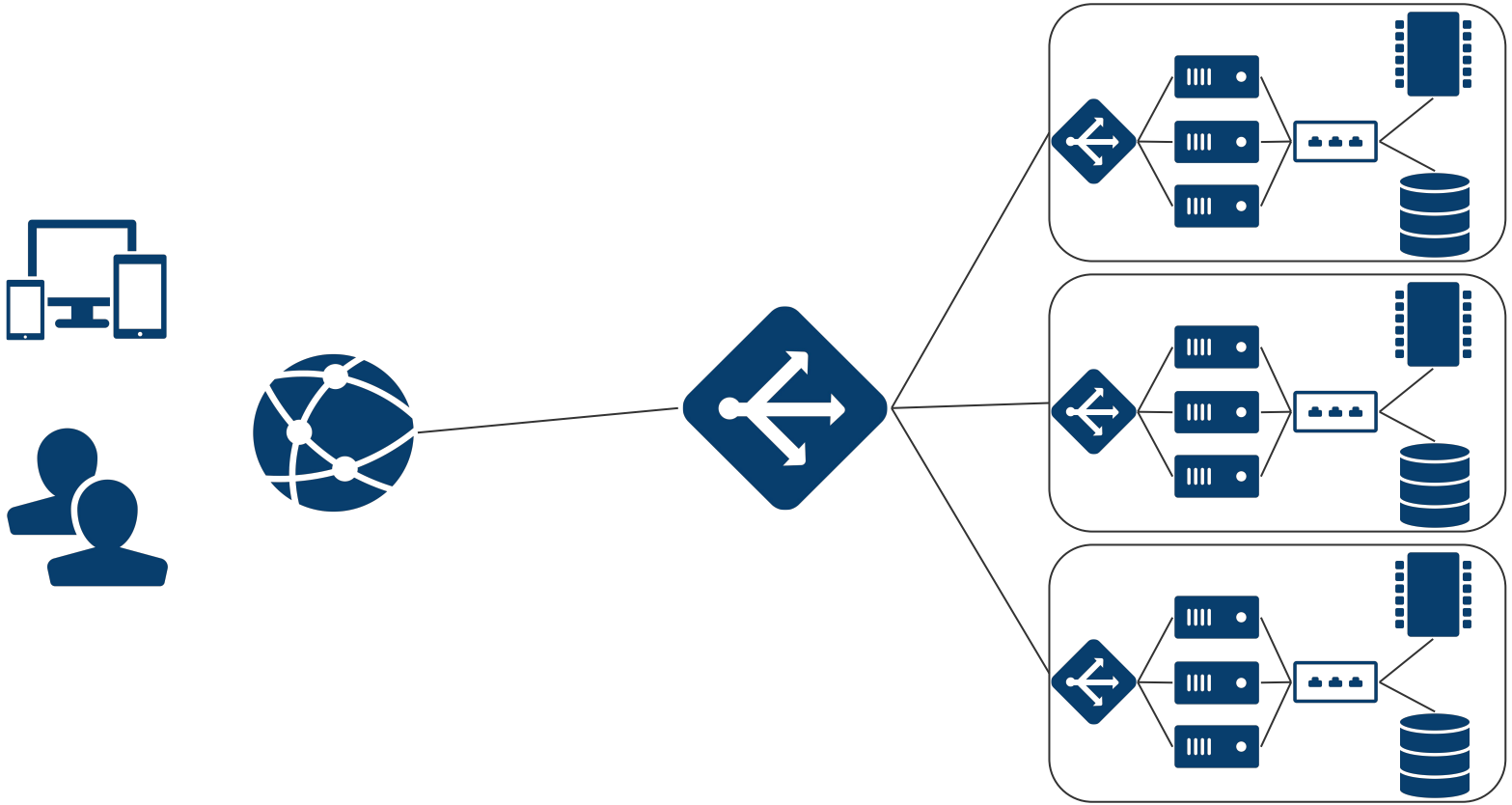
なぜオブザーバビリティなのか

- システムの複雑化、抽象化

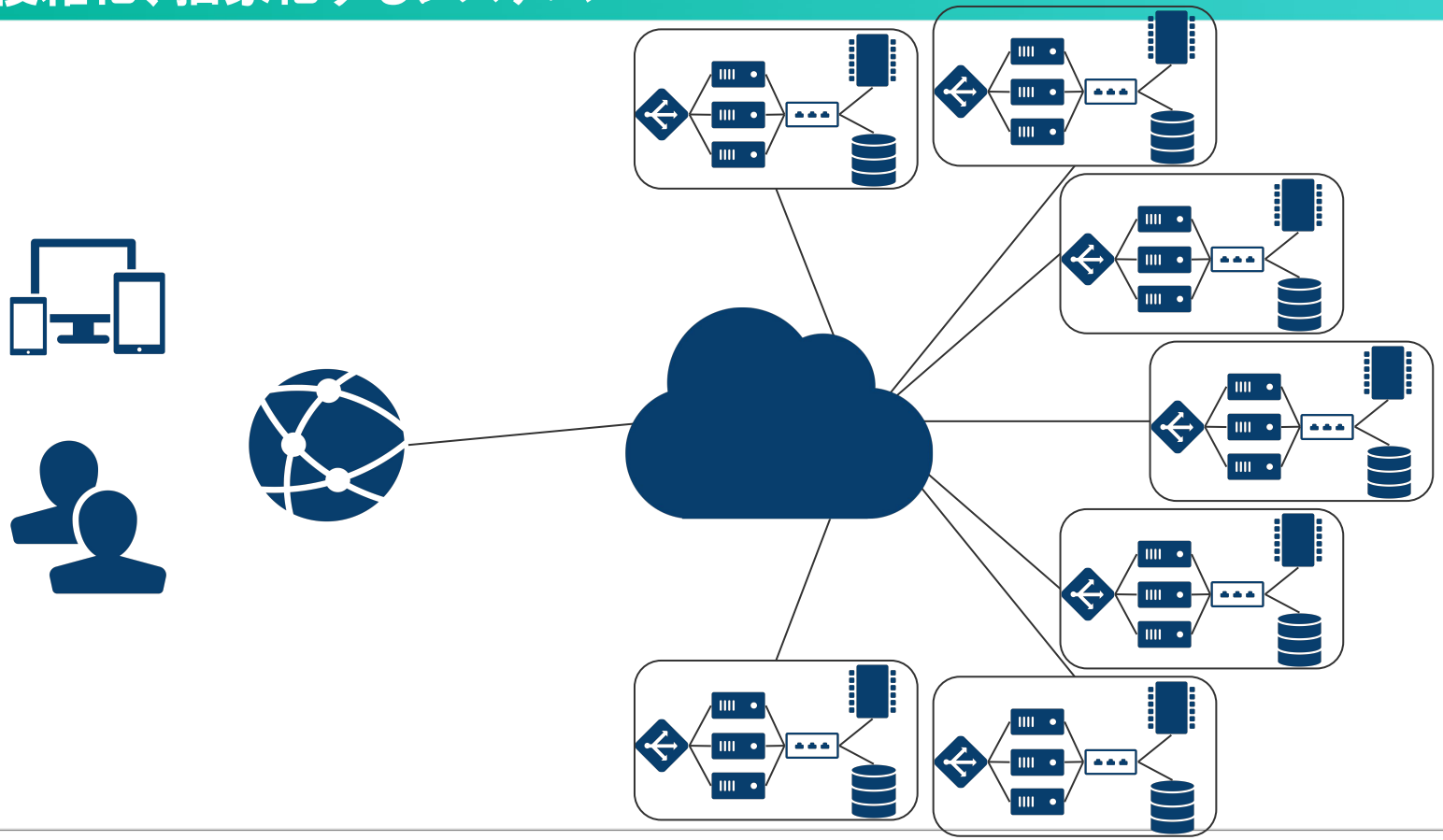
よくあるシステム



複雑化、抽象化するシステム



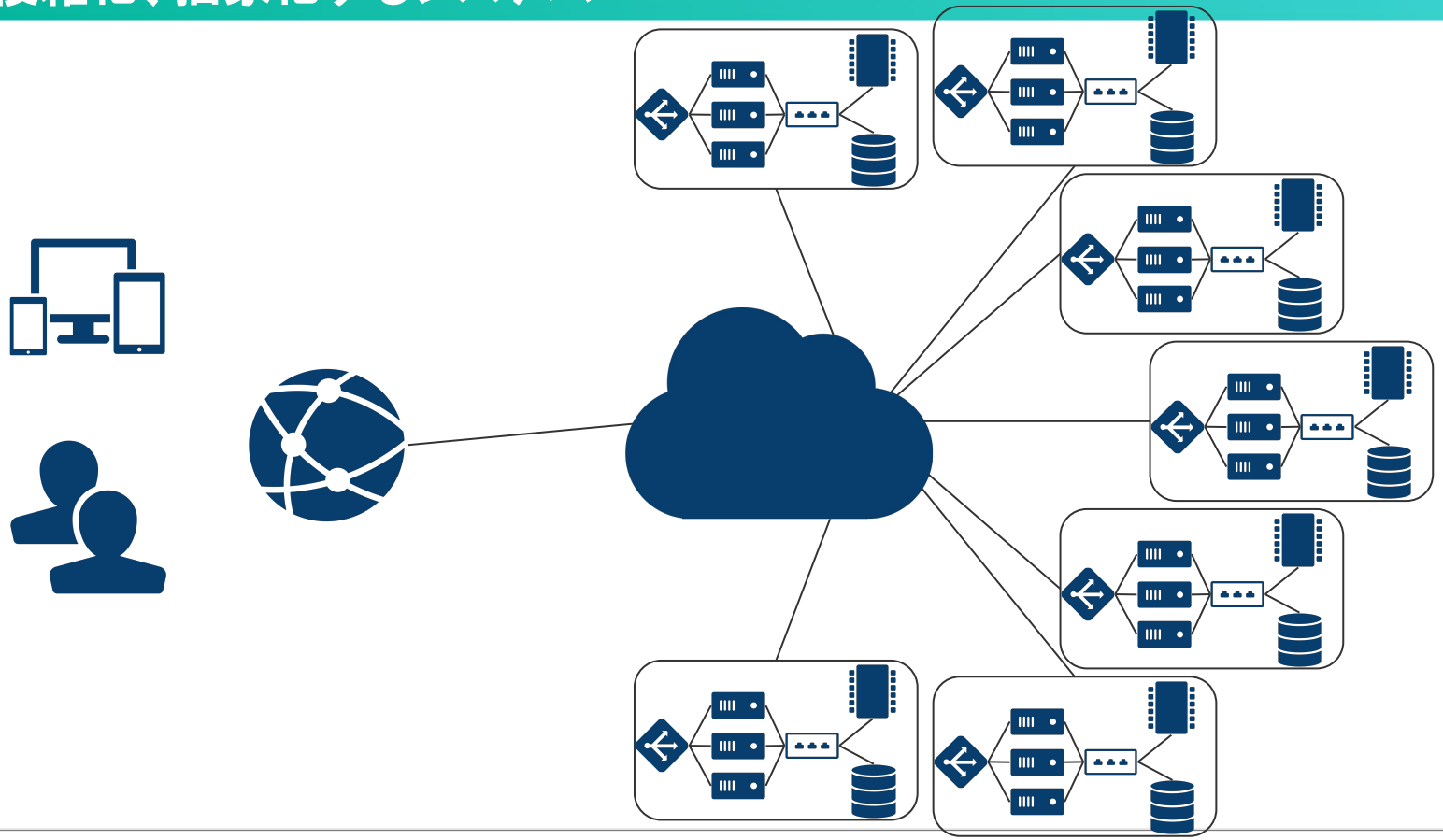
複雑化、抽象化するシステム



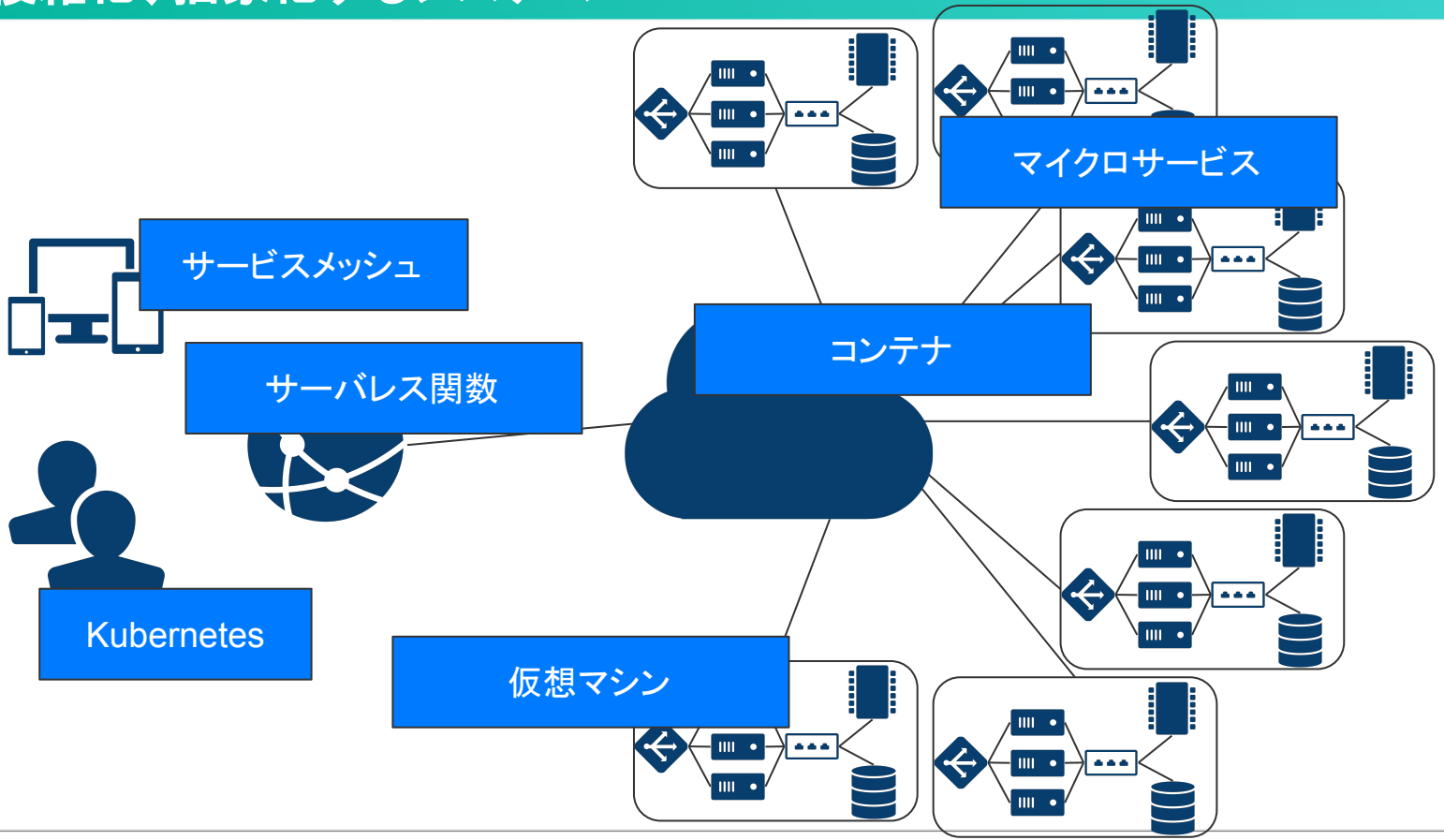
従来の監視のアプローチだと不十分

- 複雑化・抽象化するシステム
  - コンポーネントの数が増加
    - 問題が発生するパターンの増加

複雑化、抽象化するシステム



複雑化、抽象化するシステム



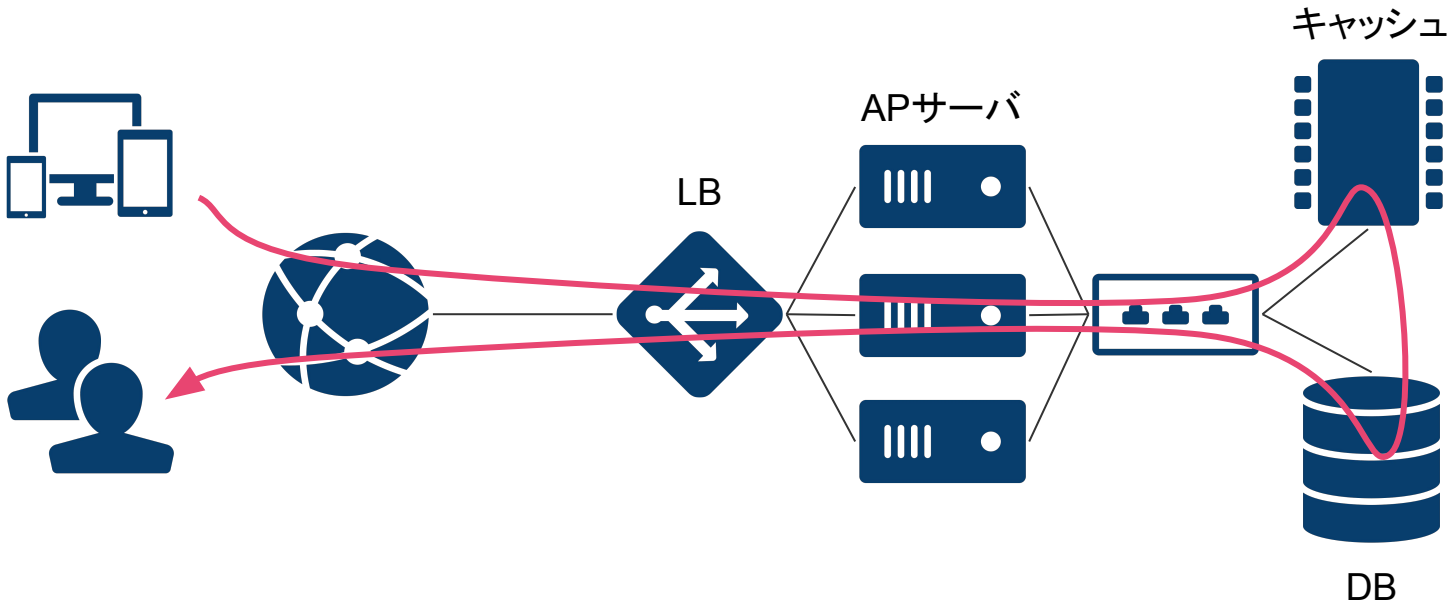
従来の監視のアプローチだと不十分

- 複雑化・抽象化するシステム
  - コンポーネントの数が増加
    - 問題が発生するパターンの増加
  - コンポーネントの種類が増加
  - 抽象化された仕組み
    - そもそも外からの監視が難しい

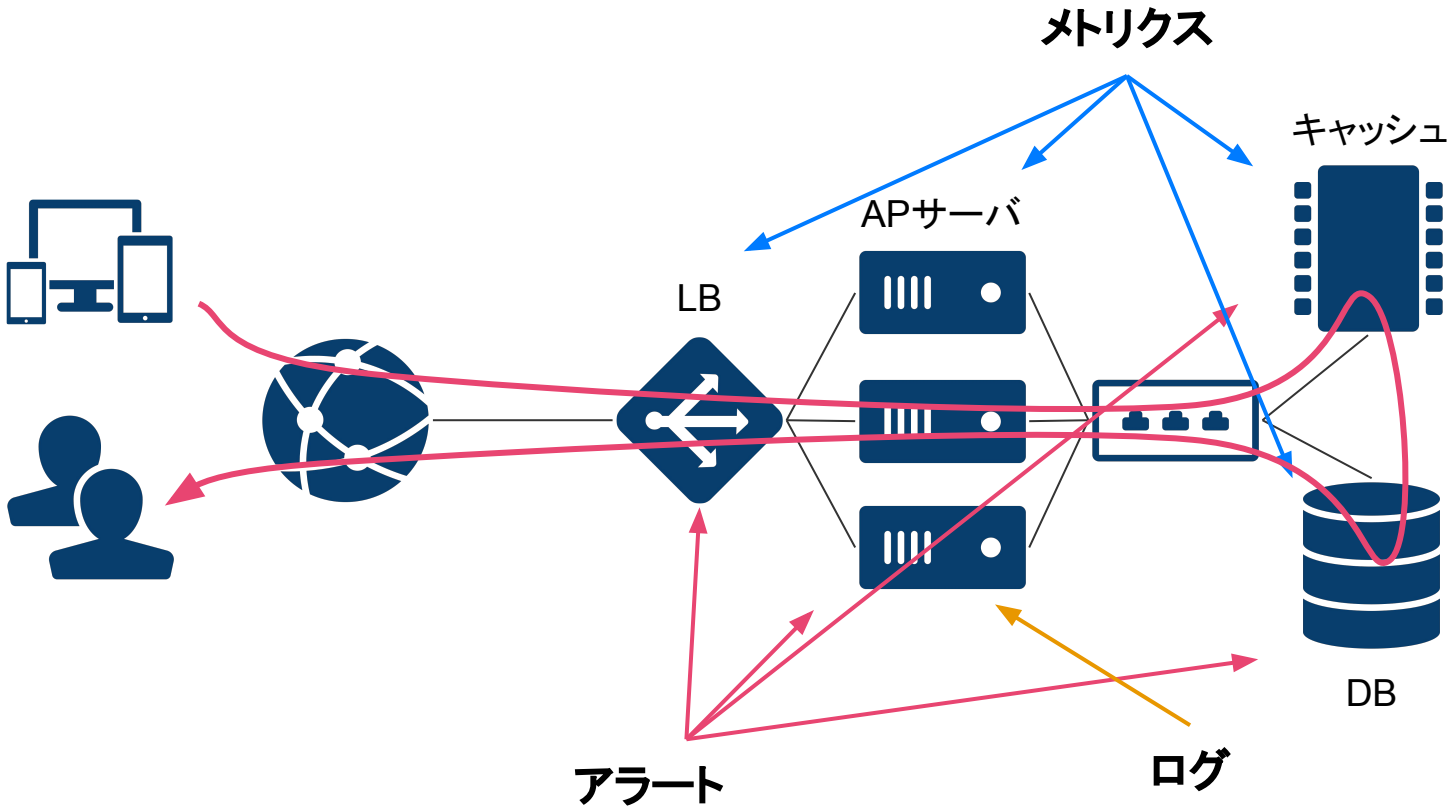
従来の監視のアプローチだと不十分

- 複雑化・抽象化するシステム
  - システムの使われ方の複雑化→複雑なアクセスパターンの発生
    - マルチテナント化
    - ユーザができることの多様化、複雑化

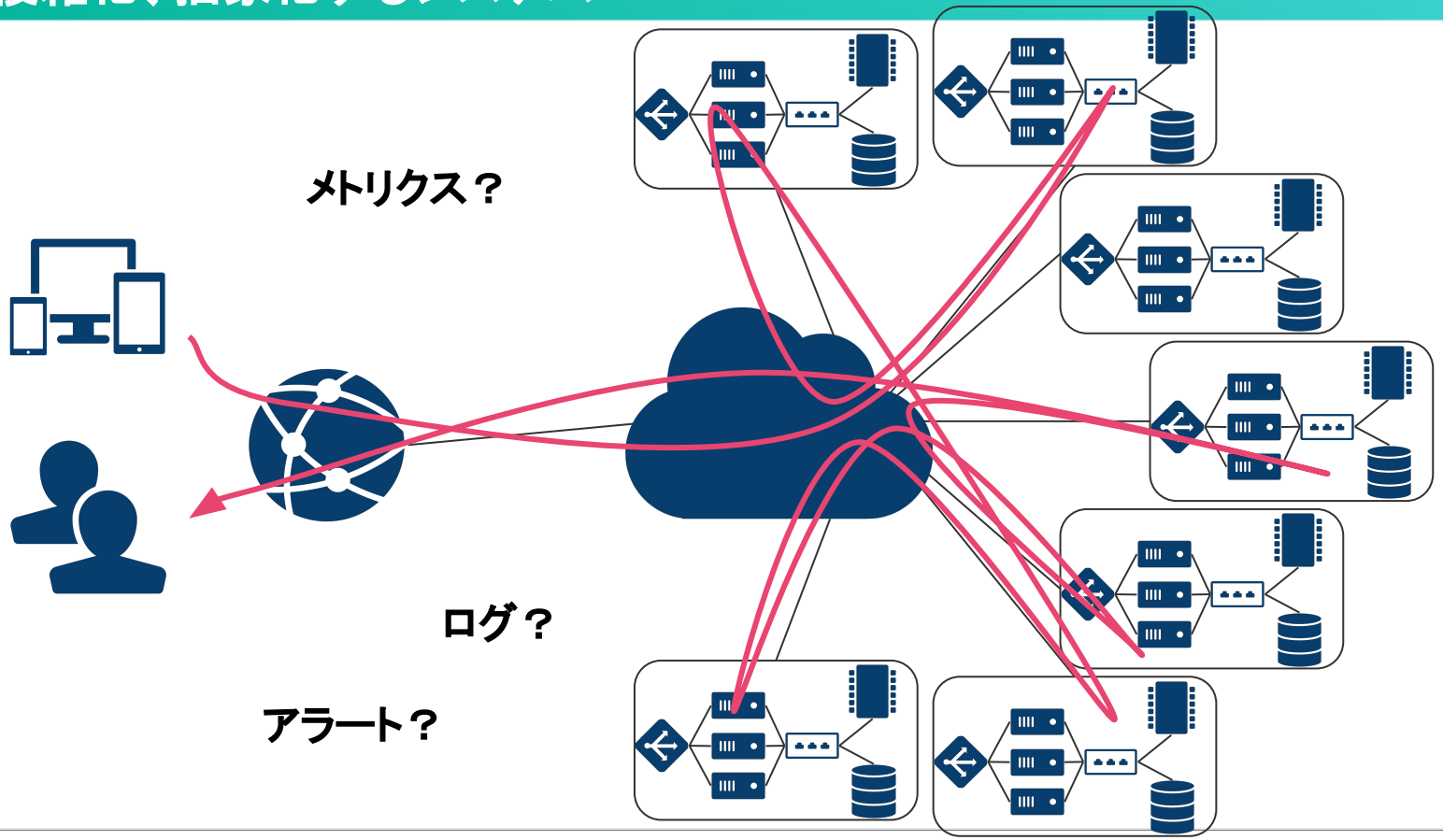
よくあるシステム



よくあるシステム



複雑化、抽象化するシステム

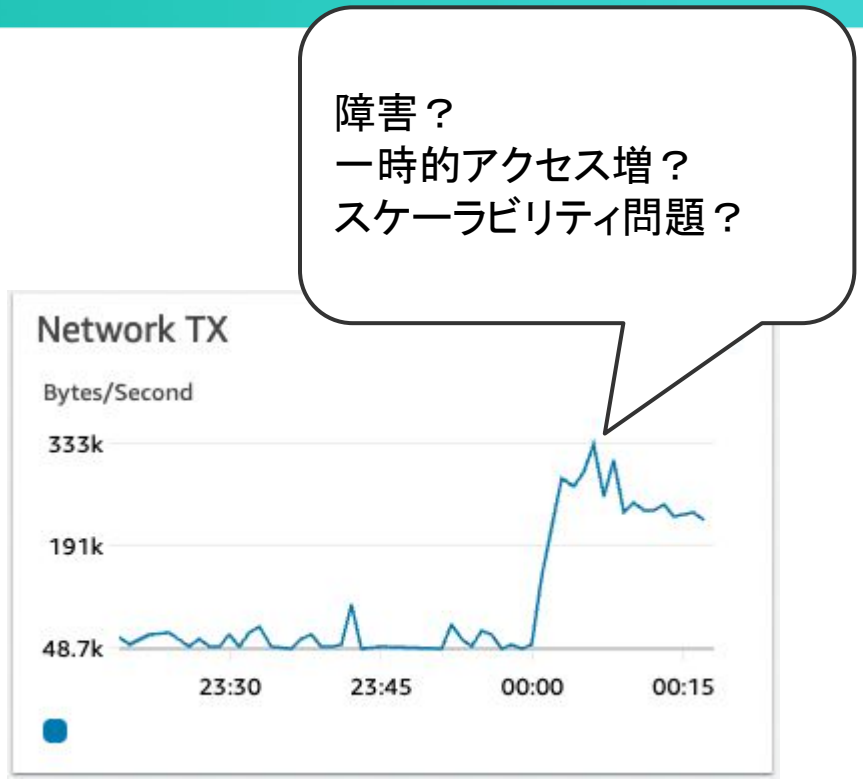


従来の監視のアプローチだと不十分

- 複雑化・抽象化するシステム
  - 点に注目した監視の限界
    - 1アクセスへの応答に多数のコンポーネントが関連する
    - 点→線あるいは面に注目する必要性

従来の監視のアプローチだと不十分

- メトリクスの限界
  - 全体傾向を把握するのに向いている
  - 「何か」起きていることしかわからない



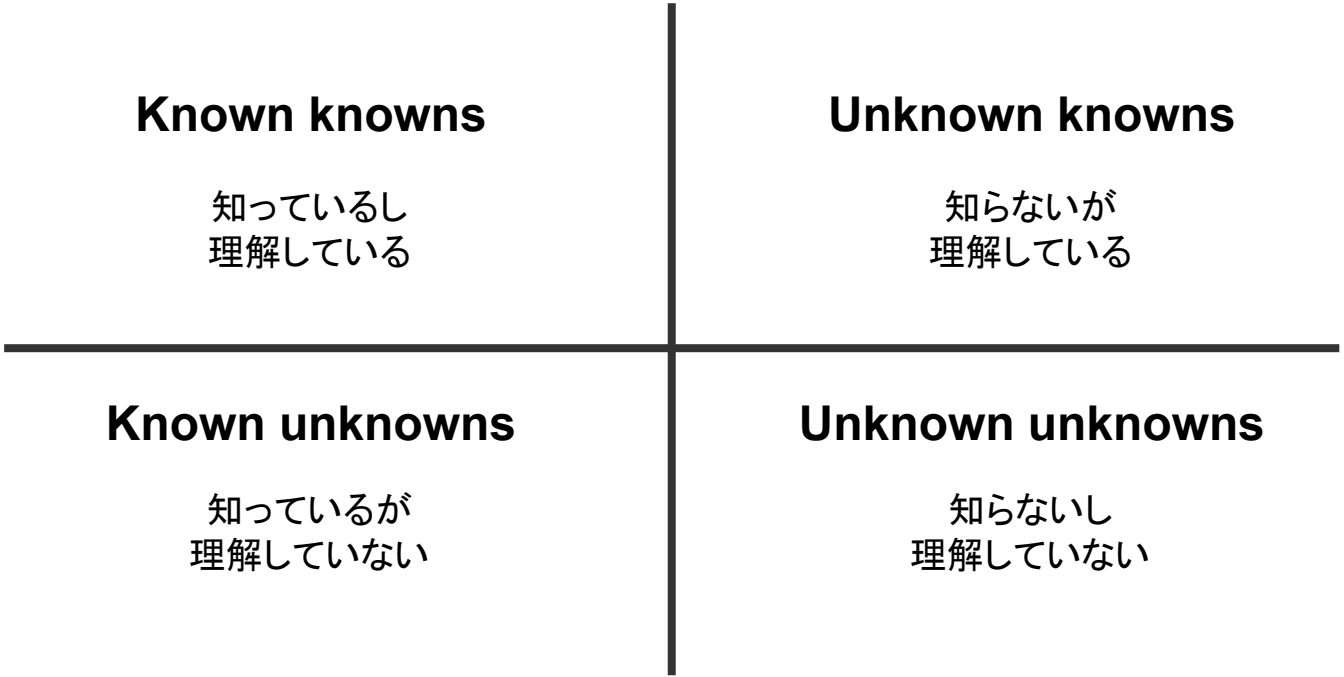
従来の監視のアプローチだと不十分

- 経験をもとにした問題特定の弊害
  - 傾向や「何か起こってる」ことしかわからない
  - 経験の長い人の知識・直感に依存する傾向(ヒーローカルチャー)

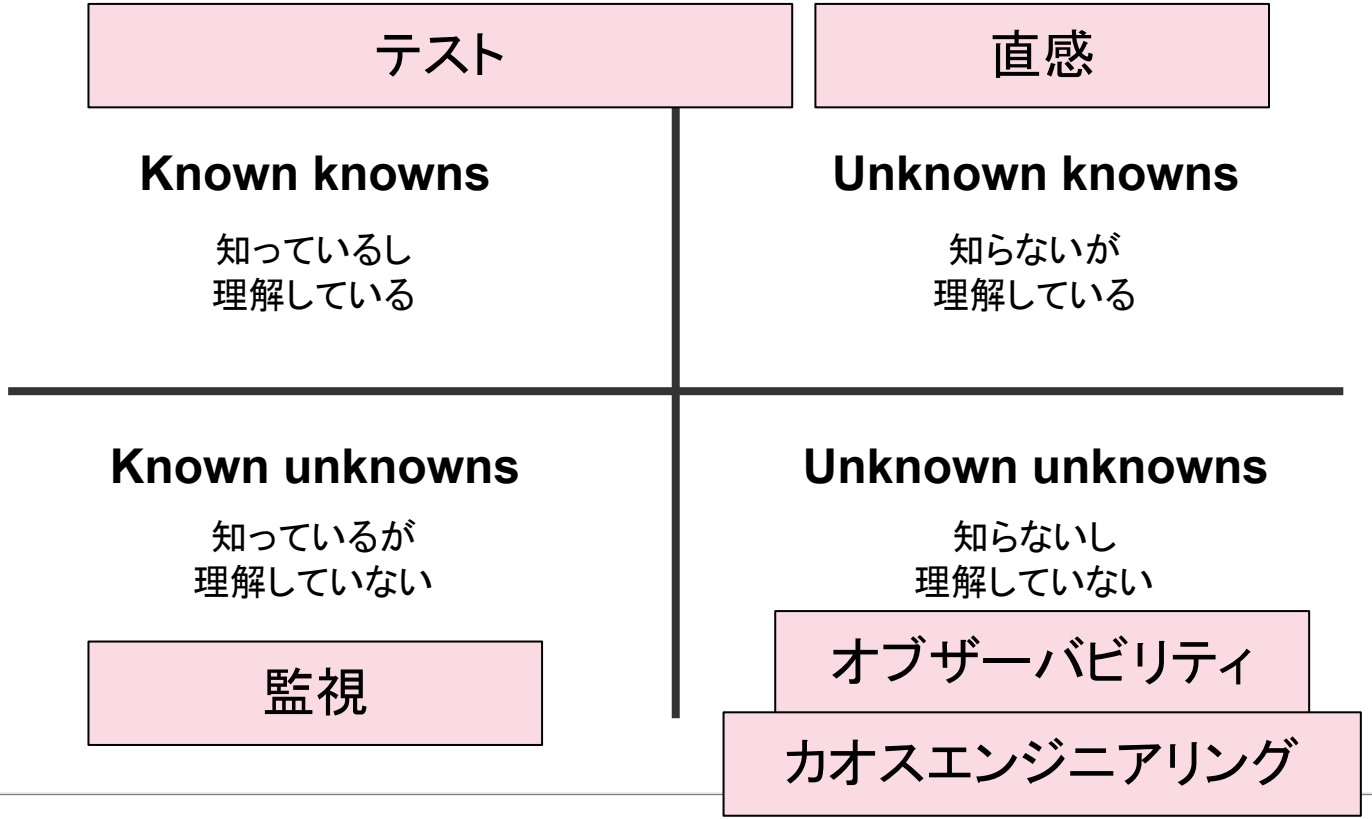
従来の監視のアプローチだと不十分

- 過去の経験を元にするアプローチで対処できなくなってきた
- 未知の状況に対処できるようにする必要

Unknown unknownsに対処する



Unknown unknownsに対処する



オブザーバビリティと監視 まとめ

オブザーバビリティ	監視
ソフトウェアに焦点	システムあるいはインフラに焦点
システムの内部を知る	システムを外部から見る
未知の状況に対応できる	過去の経験(既知)を活かす
シフトレフトが求められる	本番にデプロイされた後
個別のアクセスを見る	アクセス全体の傾向を見る
線または面	点

監視すべきところ、オブザーバビリティを考えるべきところ

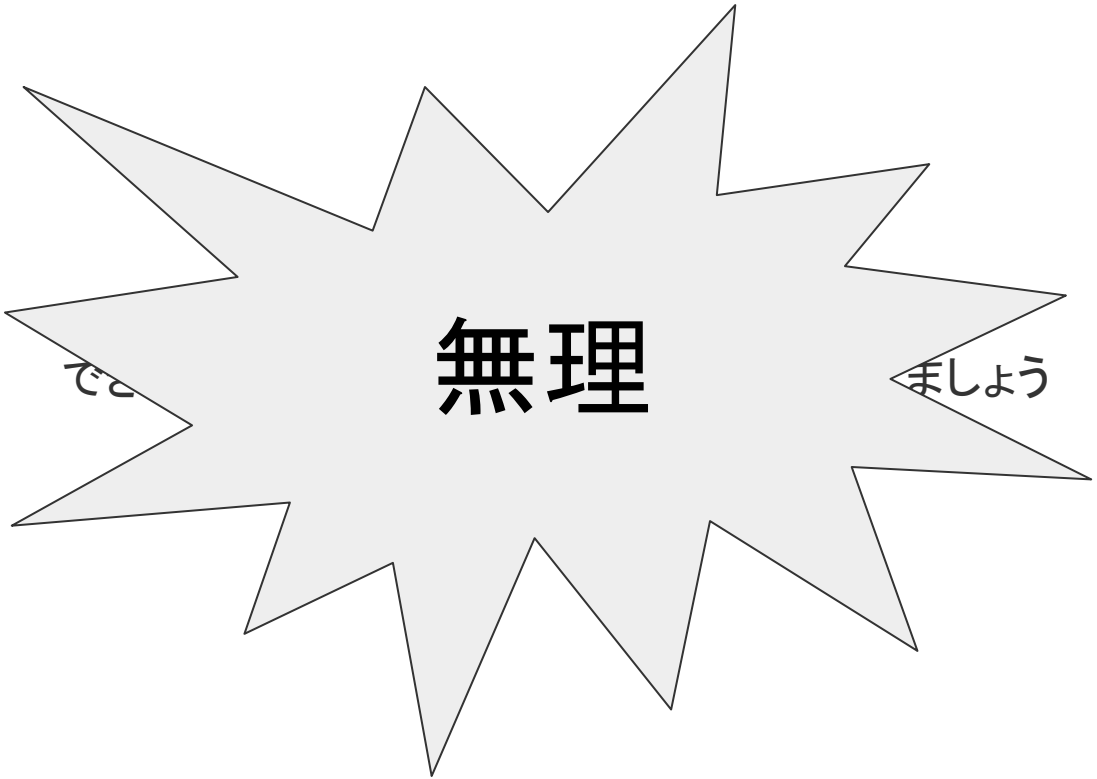
- 既知の問題・未知の問題どちらにも対応する必要
  - 監視・オブザーバビリティのどちらかではない
- ビジネス上の価値が高いのがどこか
  - IaaSなどインフラに価値がある
    - 監視 ＞ オブザーバビリティ
  - ソフトウェアに価値がある
    - オブザーバビリティ ＞ 監視

オブザーバビリティの高いシステムを作る

備えているべき性質

できるだけ多くの情報を事前に集めておきましょう

備えているべき性質

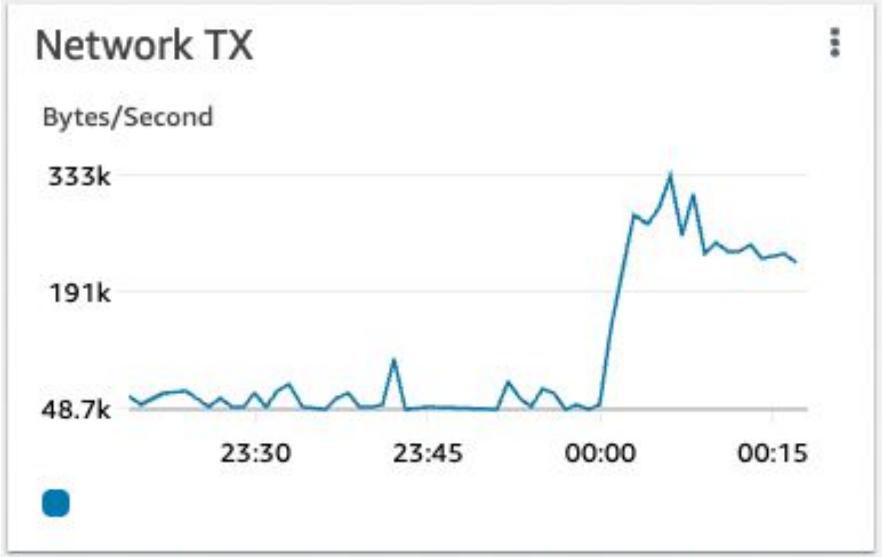


オブザーバビリティの「3つの柱」

- メトリクス
- ログ(イベント情報)
- トレース

オブザーバビリティの高いシステム

- メトリクス
  - 現在の状態を知る
  - トレンドを知る
  - アラートを送る



オブザーバビリティの高いシステム

- ログ(イベント情報)
  - 個別のアクセスについて知る
  - 3つの要件
    - 構造化
    - 高カーディナリティ
    - 高ディメンショナリティ

構造化されていないログ

```
2022-07-06T12:46:40.110Z Starting the test execution for test_id=123.
2022-07-06T12:46:40.110Z Downloading files for the execution.

...

2022-07-06T12:47:32.796Z Uploading artifacts from the worker to the storage
2022-07-06T12:47:34.538Z Marking the test result as: done.
2022-07-06T12:47:34.587Z Deleting artifacts on the worker.
2022-07-06T12:47:34.590Z Done deleting artifacts.
2022-07-06T12:47:34.590Z Waiting 20 seconds to let Monitoring Agent flush most of
metrics/traces they have...
2022-07-06T12:47:54.590Z Dying
```

構造化されたログ

```
time="2022-07-06T12:46:40.110Z" msg="Starting the test execution" test_id="123"
organization_id="3"
time="2022-07-06T12:46:40.110Z" msg="Downloading files for the execution"
test_id="123" organization_id="3"

...

time="2022-07-06T12:47:32.796Z" msg="Uploading artifacts from the worker to the
storage" test_id="123" organization_id="3"
time="2022-07-06T12:47:34.538Z" msg="Marking the test result as: done"
test_id="123" organization_id="3"
time="2022-07-06T12:47:34.587Z" msg="Deleting artifacts on the worker"
test_id="123" organization_id="3"
...
```

構造化されたログ

```
{
    “time”:”2022-07-06T12:46:40.110Z”,
    “msg”:”Starting the test execution”,
    “test_id”:”123”,
    “organization_id”:”3”,
    “user_id”:"5678”,
    “duration”:378,
    “status”:200,
    “trace_id”:”8d94158f-70ec-4a27-bcf4-54f139f61b5c”
}
```

カーディナリティとは

- カーディナリティ (Cardinality)
  - 数学
    - 基数
  - データベース
    - あるカラムに含まれるデータのバリエーションの多寡

カーディナリティとは

- カーディナリティの高いイベント情報
  - 一意にアクセスを特定できる可能性が高い情報
  - アクセスログの例
    - IPアドレス ＜ ユーザ名 ＜ セッションID ＜ アクセスUUID
  - 調査の際に役立つ可能性が高い

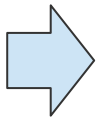
ディメンショナリティとは

- ディメンショナリティ (Dimensionality)
  - 含まれるキーの数の多寡

ディメンショナリティとは

- ディメンショナリティの高いイベント情報
  - 1イベント情報に対して多くの付加情報を持つ
    - 調査の際に多くの情報を得られる
    - 多くの視点からイベントを検索できる
    - 多くの切り口で複数のイベントを関連付けられる

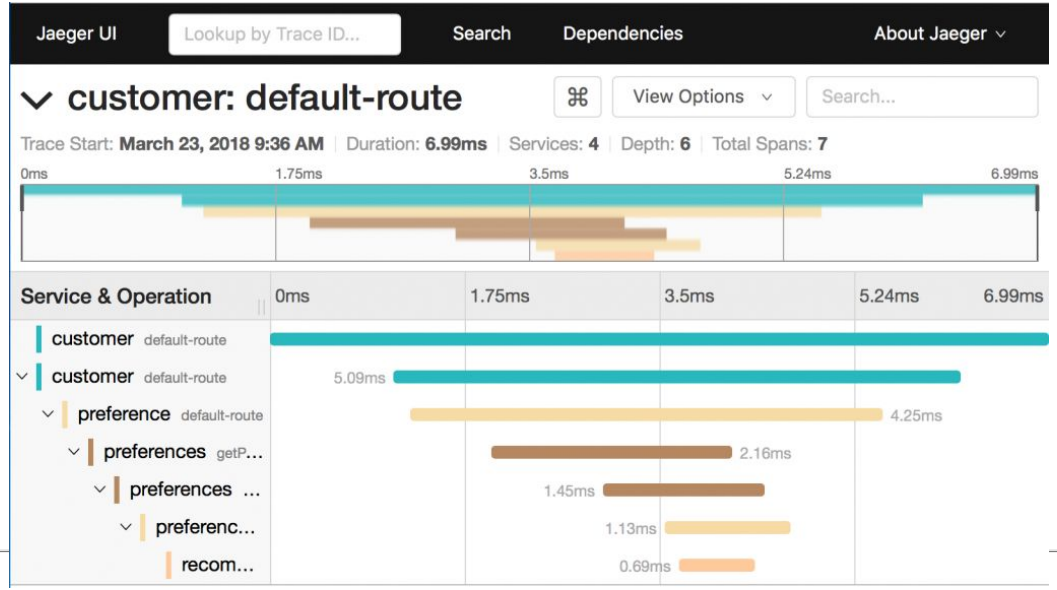
```
{
  "time":"2022-07-06T12:46:40.110Z",
  "msg":"Starting the test execution",
  "status":200
}
```



```
{
  "time":"2022-07-06T12:46:40.110Z",
  "msg":"Starting the test execution",
  "test_id":"123",
  "organization_id":"3",
  "user_id":5678,
  "duration":324,
  "status":200,
  "trace_id":"54f139f61b5c"
}
```

オブザーバビリティの高いシステム

- トレース
  - 1リクエストで行われたトランザクションの詳細を知る
  - リクエストがどのような処理をたどったのかを確認する情報



オブザーバビリティの高いシステム

- 集計された情報(例、メトリクス)よりも個別のリクエストの情報
  - 構造化されたイベント情報(ログ)
    - 高いカーディナリティ
    - 高いディメンショナリティ
  - トレース

オブザーバビリティの「3つの柱」(の功罪)

- メトリクス、ログ、トレース
- 確かにこの3つは重要な要素
- 3つの柱がある ＝ オブザーバビリティが高い、ではない
- 未知の問題の解決に必要な情報は何か？
    - 3つの柱は手段

Observability Driven Development

- 作ってからオブザーバビリティを考えても遅い
  - オブザーバビリティもシフトレフトする必要
  - オブザーバビリティを高める情報は開発者を助ける
    - どの部分をデバッグしたらいいのかをわかりやすくする情報

テスタビリティとオブザーバビリティ

- テスタビリティを実現する1要素
- システムの品質を高めるために欠かせない性質

まとめ

まとめ

- 監視とオブザーバビリティはパラダイムの違い
  - 既知に対処するか、未知に対処できるようにするか
- どちらも共存できる
- 複雑なシステムではよりオブザーバビリティが重要になる
- オブザーバビリティを考えながら開発する

参考資料

- 入門 監視 (Mike Julian, 松浦隼人訳)
- Observability Engineering (Charity Majors, Liz Fong-Jones, George Miranda)
- Observability Whitepaper (CNCF)


Autify