OSSで構築・運用する
いまどきのデータ分析・可視化システム

株式会社スクウェア・エニックス
テクノロジー推進部
伊澤 徹

自己紹介

株式会社スクウェア・エニックス テクノロジー推進部
分散システム エンジニア
伊澤 徹

- 経歴
  - 大学: 並列計算機 (ハードウェア + ネットワーク)
  - 前職: ネットワークR&Dと機器設計
  - 現職: サーバー方面全般

対象

- 対象範囲
  - システム構築～運用

- 対象外
  - 得られたデータの使い方、運営施策

モチベーション

- なぜOSS？
  - 個別にツールを実装すると…
    - 何度も同じようなものを作る
    - 実装が後回しになる
    - 継続性の問題

- なぜ今？
  - 安定して使えるソフトウェアが揃ってきた
    - 特にメッセージキューと可視化部分
  - 分析系への期待、理解が高まっている

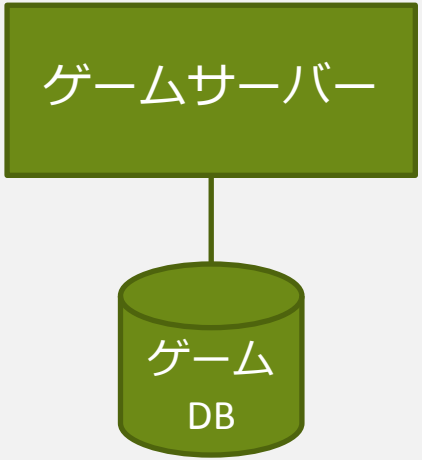
本日の流れ

- システム構成
  - 一般的な話
  - 構成例

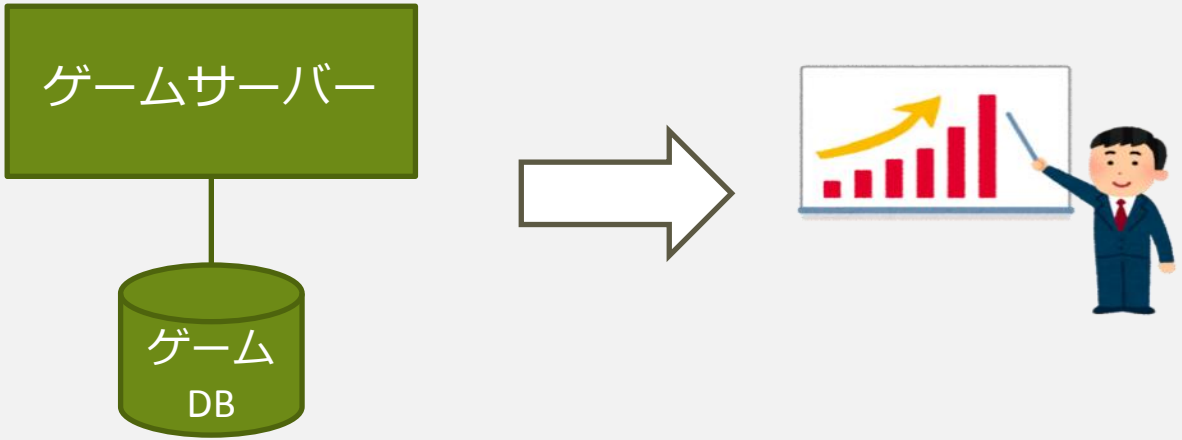
- 運用
  - 必要な知識
  - 事例

システム構成

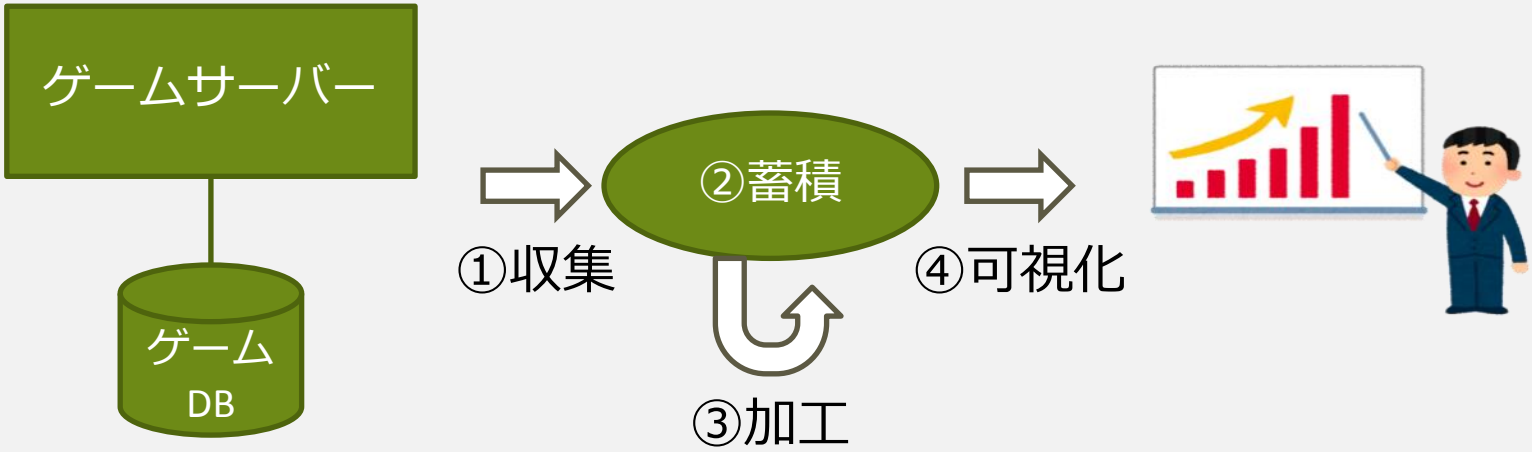
システム構成



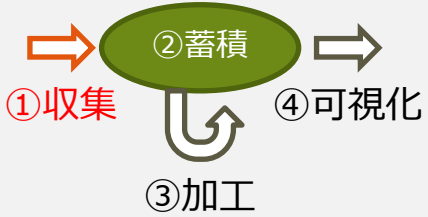
システム構成



システム構成

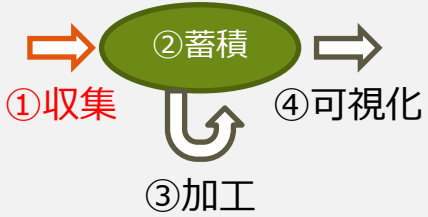


①収集: ゲームサーバーから収集



- 対象
  - イベントログ: 何がいつ発生したか
- 送信方法
  - 一度ローカルに出してFluentdなどで拾う (簡易)
  - ゲームサーバーから直接送る
    - 要： 失敗時の挙動設計
- データフォーマット
  - JSONが楽
  - 性能面では勿論バイナリがよいが…
    - 後々のフィールド変更(追加削除)も考慮
    - 変更作業の容易さは開発分担にもよる

①収集: ゲームDBから収集



- DBにあるデータの例
  - ユーザーデータのスナップショット
  - マスターデータ
  - 失われてはいけないログ

  DBにあるものはそれを使えばよい

- 方法
  - スレーブから定期的に取ってくる

②蓄積

- 保存先候補
  - クラウド業者のオブジェクトストレージ
  - HDFSを自前運用
  - 分散DB層を動かす: Apache Cassandra, Apache HBase

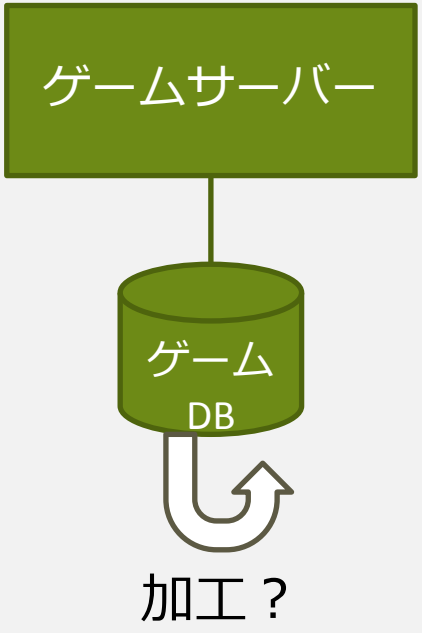
  手間と速度のトレードオフ

- 受信用バッファ
  - 細かい書き込みは性能面で不利なので、バッファを設け時々まとめて書く
  - Apache Kafkaがお勧め
    - データ保持期間内なら何度でも読み出せる
      - 複数システムから読める
      - インデックスを記録しておけば、後段の処理再走や復旧も容易

③加工

- データの変換・結合
  - 結果をそのまま表示したり
  - 結果をストレージに書き戻したり
- 可視化(④)とセットで考える
  - 大抵可視化用のインターフェイスから処理を実行するので
- サーバー1台では足りない場合は管理が必要
  - 計算リソースとタスクの管理
  - YARN (Apache Hadoop) とか Apache Mesos とか

なぜDB上で加工を済ませないのか

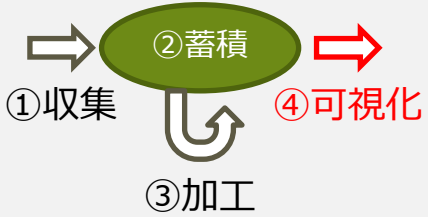


- スキーマ変更が大変

- 書き込みの量や経路を増やしたくない
- 開発チームの工数を消費したくない
- DB設計スキルに依存したくない

- 特に開発が別組織の場合…

④可視化

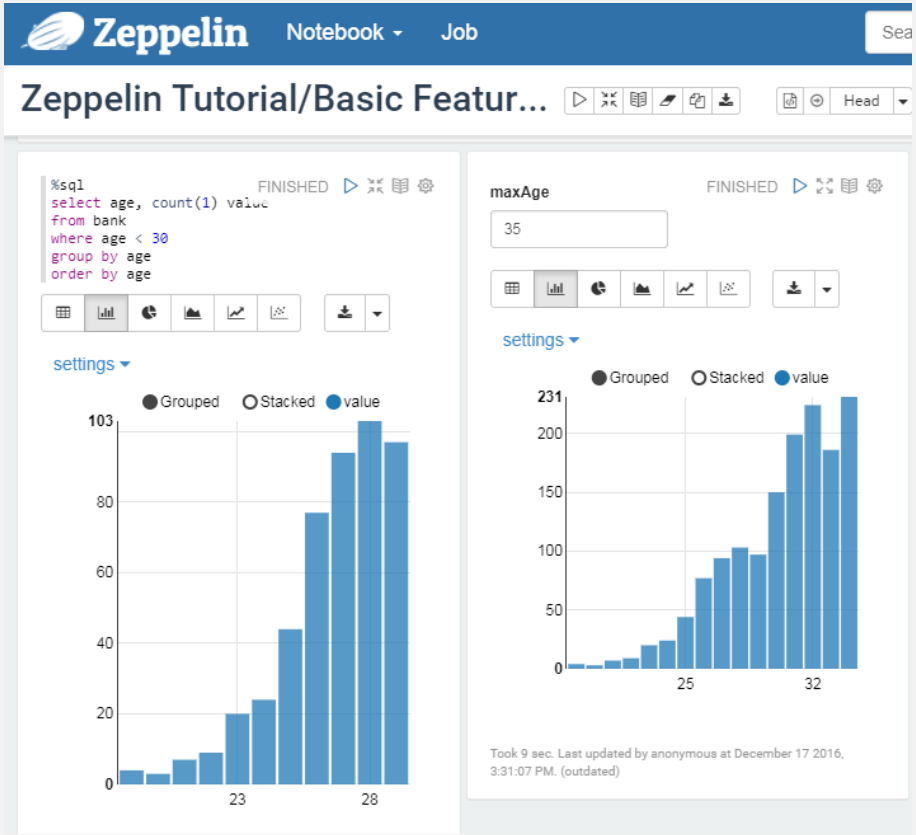


- クエリーを投げ、グラフを出す

- ここ1～2年で実用的な選択肢がだいぶ増えた
  - Jupyter (+ Apache Toree: Spark用Kernel)
  - Redash
  - Apache (Incubator) Superset
  - Apache Zeppelin

※機能や接続性の○×表を見て選ぶことはせず、試してみる
(時々怪しい)

Apache Zeppelinの紹介



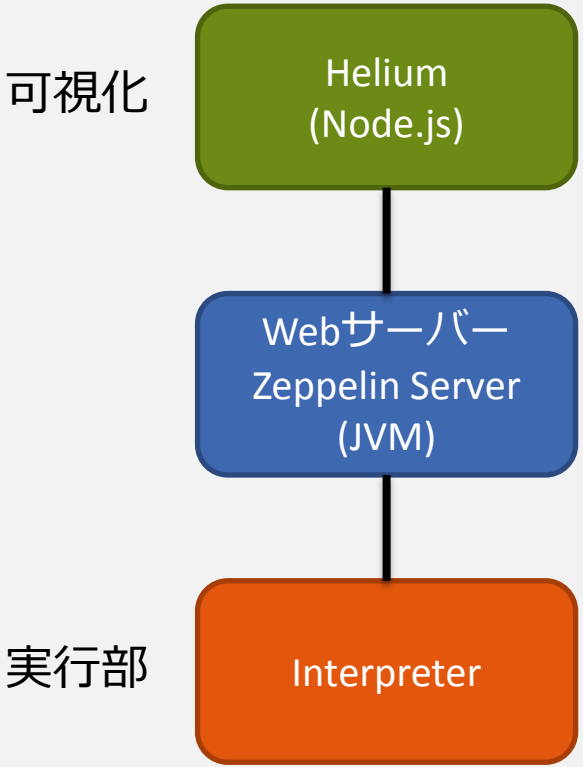
- コード・クエリーを実行、結果をテーブルやグラフで表示
- コード＋描画内容を保持
  - export/import可能
- cronのような定期実行機構を持つ
  - 定時・定期更新を回せる
- CSV出力対応
  - あの大人気表計算ソフトウェアとの連携が可能
    - 導入に際して説得力は高い

Apache Zeppelinの紹介

```
var dateTimeStartHour = z.input("開始日") + " " + z.input("開始時間","00:00");
```

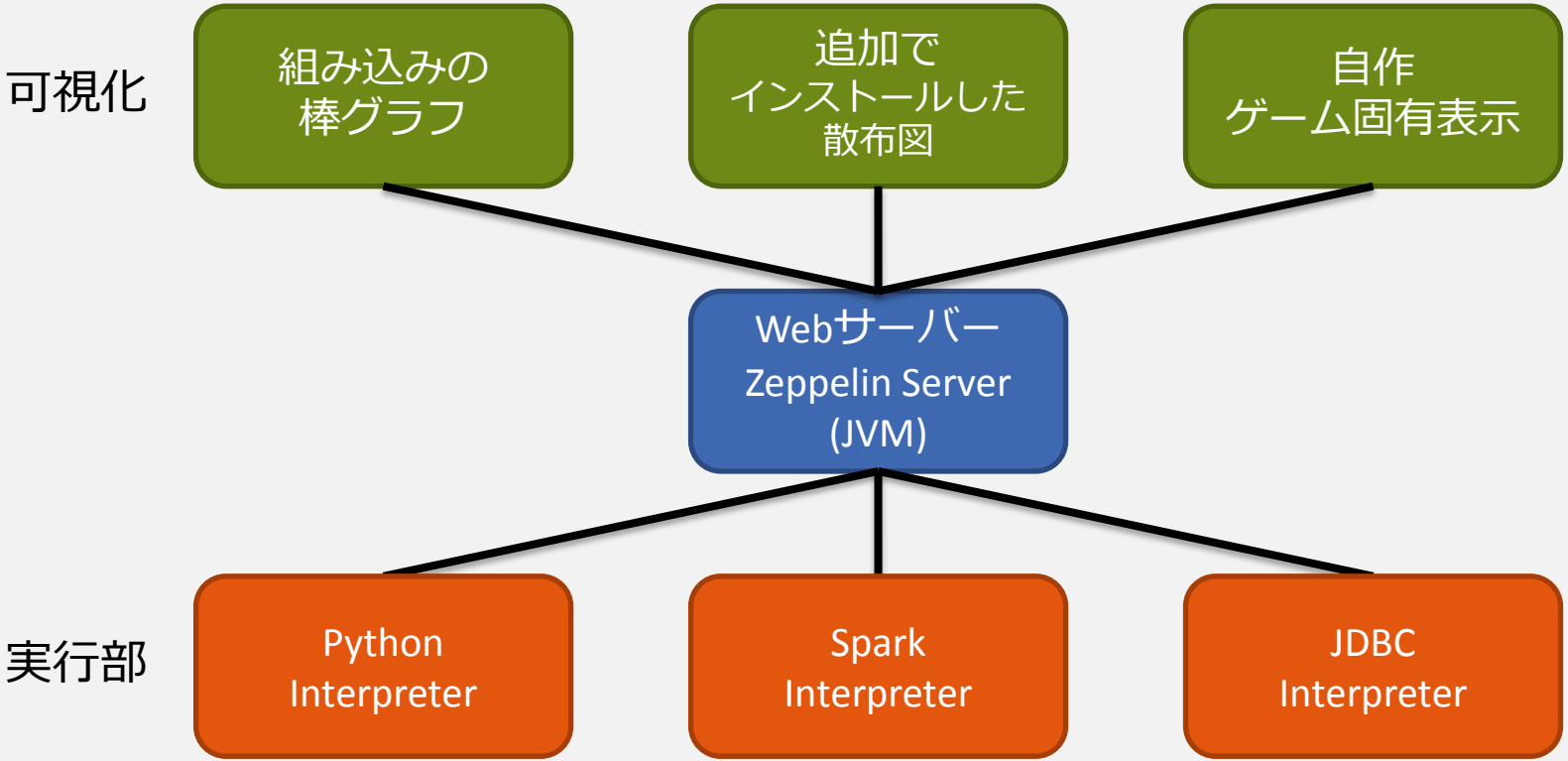
- Dynamic Formが便利
  - Input, Checkbox, Select
  - 利用者の編集する箇所を限定した運用が可能

Apache Zeppelinの紹介

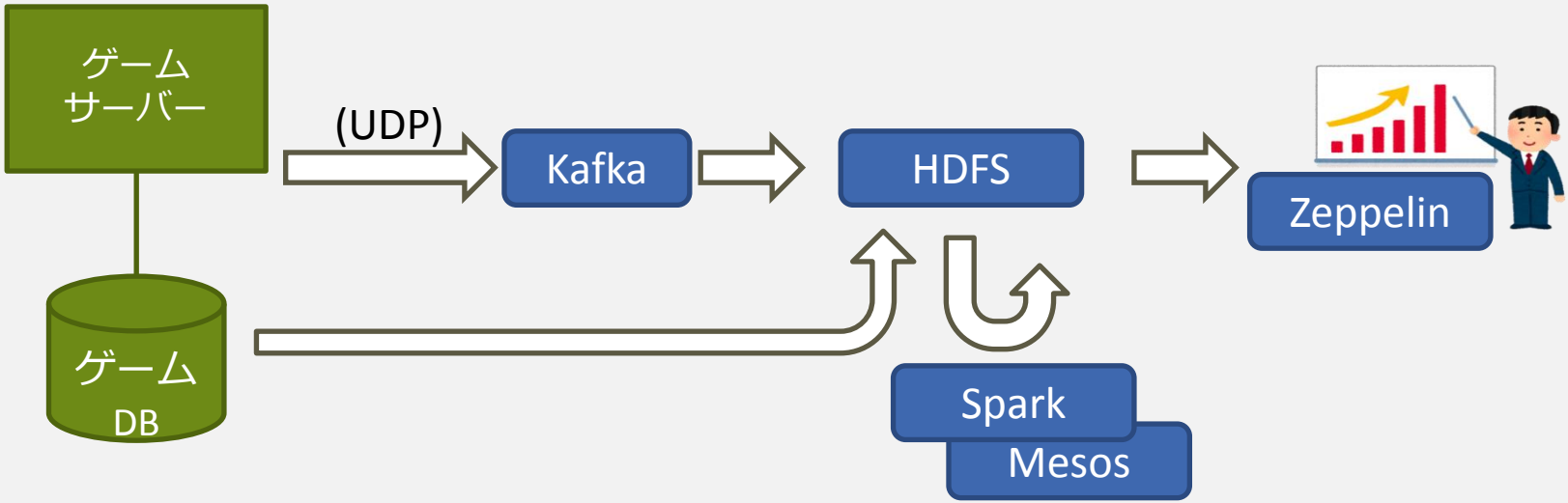


- 実行部、可視化が分離
  - どのInterpreterからでも
  - どの可視化形式にも出せる
  - Interpreterで最終出力(HTML/画像)を描画してしまってもOK
    - PythonとかSparkRとか
    - 開発者向け

Apache Zeppelinの紹介



システム構成実例



運用

大事なこと

- ソフトウェアを知る

- 何が起こっているのかを知る

ソフトウェアを知る

- 仕組みを理解
  - 公式ドキュメントや仕組みを解説した書籍
    - 性能チューニングや経験談を書いたものより先に…
- 設定可能項目を全て眺めておく
  - 項目間の相互作用、優先順位があることも

何が起こっているのかを知る

- モニタリング
- アラート

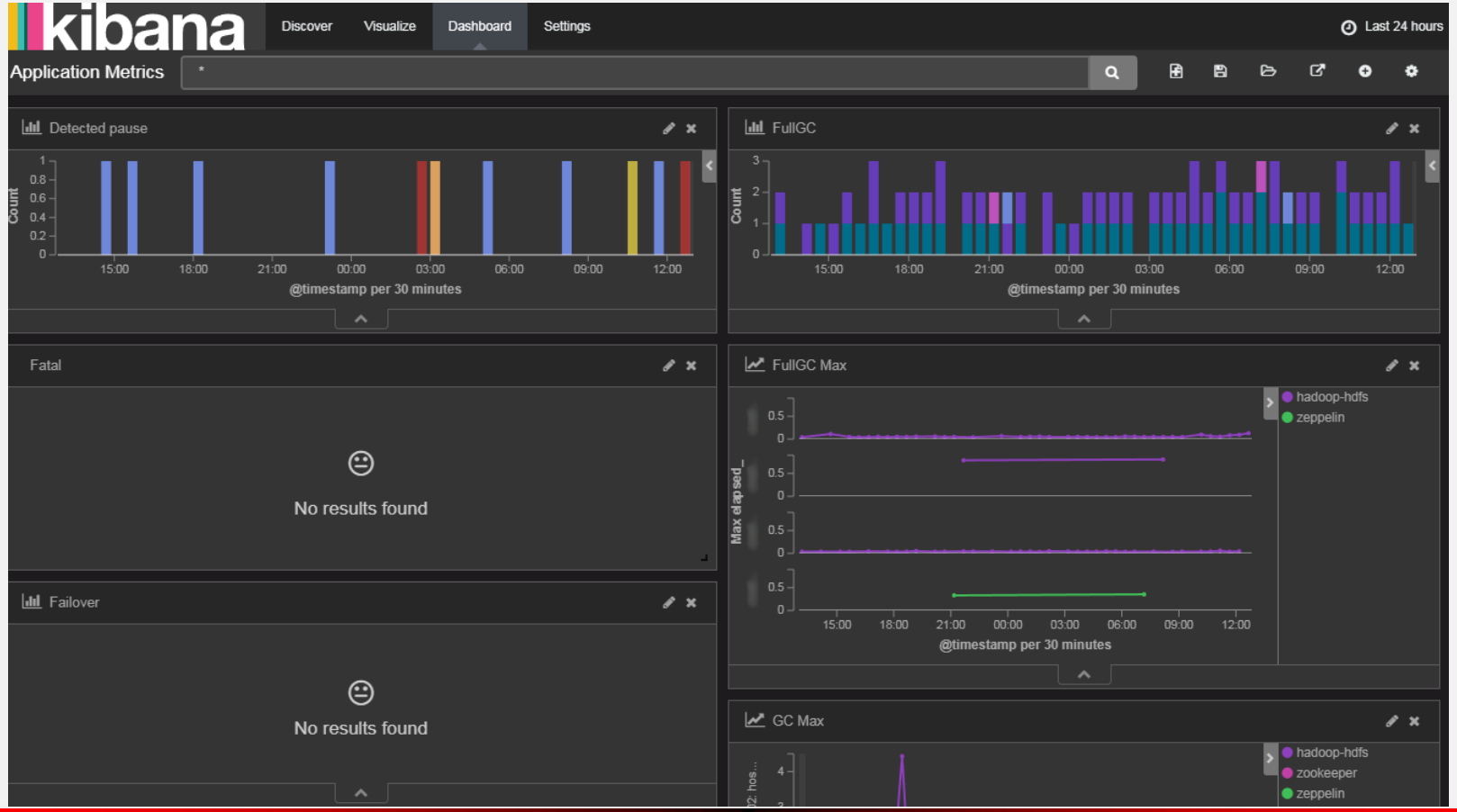
…は当然として

ログ集約

- sshでログインし見て回るのは不可能
  - 各ノード/各サービスごとにログが出る
- 複数の指標を比較したい
  - 性能指標と近い時刻のイベントを比較
- 実例: Fluentd → Elasticsearch → Kibana
  - INFOログ仕分けが必要なため、テキスト検索に強いシステムに

※ゲームのイベントログではなくて、構築した分析・可視化システムのログです

ログ集約の実例



悲しい事例集

ディスクが溢れる

- Kafkaが溢れる、Elasticsearchが、InfluxDBが…
  - Retention (保持期間)が設定できるシステムはそれで
  - 生logはローテート、定期削除

ディスクが空いてるのに溢れる

- inode (管理領域)が細かいファイルで埋まる
  - 例: Sparkの一時ファイル
  - 容量とinode両方監視
  - パーティションを別に切る手も

- ファイル削除コマンドを実行したが掴んだまま
  - lsofで確認

監視のために定期実行するコマンドが重い

- 監視タイミングで高負荷
  - 監視プラグインをザクザク挿した結果
    - それぞれがhadoopコマンドでJVM起動していた
    - しかも同じタイミングで…

Spark上でデータ型が合わない

- 途中から同じフィールド名で違うデータ型が！
  - そういう送り方をしないように頼む、最初から約束する

- 同じJSON形式で送り続けているのに、途中から型が合わないエラーが出る
  - Spark等でJSONの型を推測させた場合の挙動
    - JSONの空配列[]やnullの型はわからないので、Stringとして保存
      - 対策：空を送るくらいならその要素は送らないほうがまし
    - 正しいデータ型で格納されたファイルとマージできない

Sparkが想定外に遅い

- spark.locality.wait を待っていることがある
  - 処理対象データのあるノードで処理が出来なければ○秒待つ
  - デフォルト3秒

- 並列性のパラメーターが高すぎて無駄に細切れの処理
  - ノード数よりはるかに多いpartitionを生成

※ 大規模処理が想定されたシステムに対し、小規模な問題を与えていることが原因
試しに少ないデータでやってみた場合など

まとめ

まとめる前に…

- 必要なスキルを振り返ってみる
  - Linuxの性能・リソース指標がわかる
  - JVMのメモリ管理(ヒープ設定・GC)の仕組みがわかる
  - 英語ちょっと読める
    - ドキュメント、ログ…できればIssueやPull Requestの状況理解
  - ネットワークちょっとわかる
    - 特定のhost:port間の疎通状況確認くらいは
  - SQL、Python、Scala(Spark) いずれかでデータ操作できる
- Linuxサーバーエンジニアの必要スキルと大きくは変わらないはず

まとめ

- 運用を考慮した準備をして
- 持続可能な分析・可視化システムを実現して
- みんなで幸せになりましょう
  - OSSのバグ報告も布教も小さな貢献

権利表記

- 掲載されている会社名、商品名は、各社の商標または登録商標です。