



TREASURE DATA

# EmbulkとDigdagとデータ分析基盤と

～第5回 ゲームサーバ勉強会～

2016.06.18



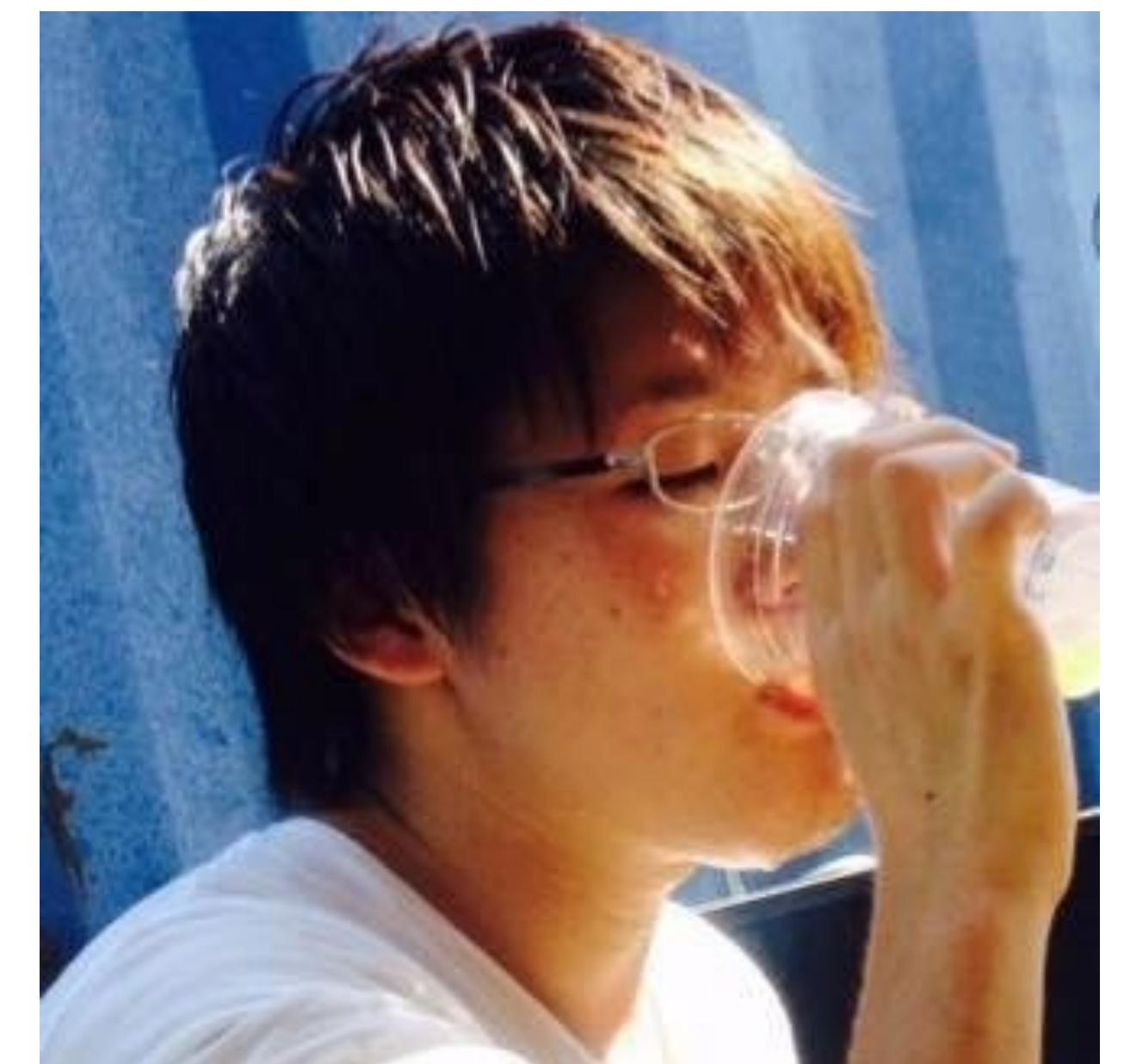
Toru Takahashi

Support Engineering Manager, Treasure Data, Inc.



# WHO AM I ?

- Toru Takahashi (@nora96o)
- Treasure Data, Inc.
- Support Engineering Manager
  - メールにチャットに、ブログ書いたり、コードを書いたり、
  - <http://qiita.com/toru-takahashi>
- 気づくと、社会人4年目に突入・・・





## 質問です！

- ・ Treasure Data を知っている人は？
- ・ Fluentd を知っている人は？
- ・ Embulkを知っている人は？
- ・ Digdagを聞いたことがある人は？
- ・ インフラ / 分析基盤 を普段から運用や開発をしている人は？
- ・ iOS / Android / Unity / フロントエンドの人は？



## 目次

対象：データ分析基盤を普段触っていない人向け。

1. はじめに～データ分析基盤について～

2. **Embulk**～Bulk Data Loader～

3. **Digdag**～Workflow Automation System～

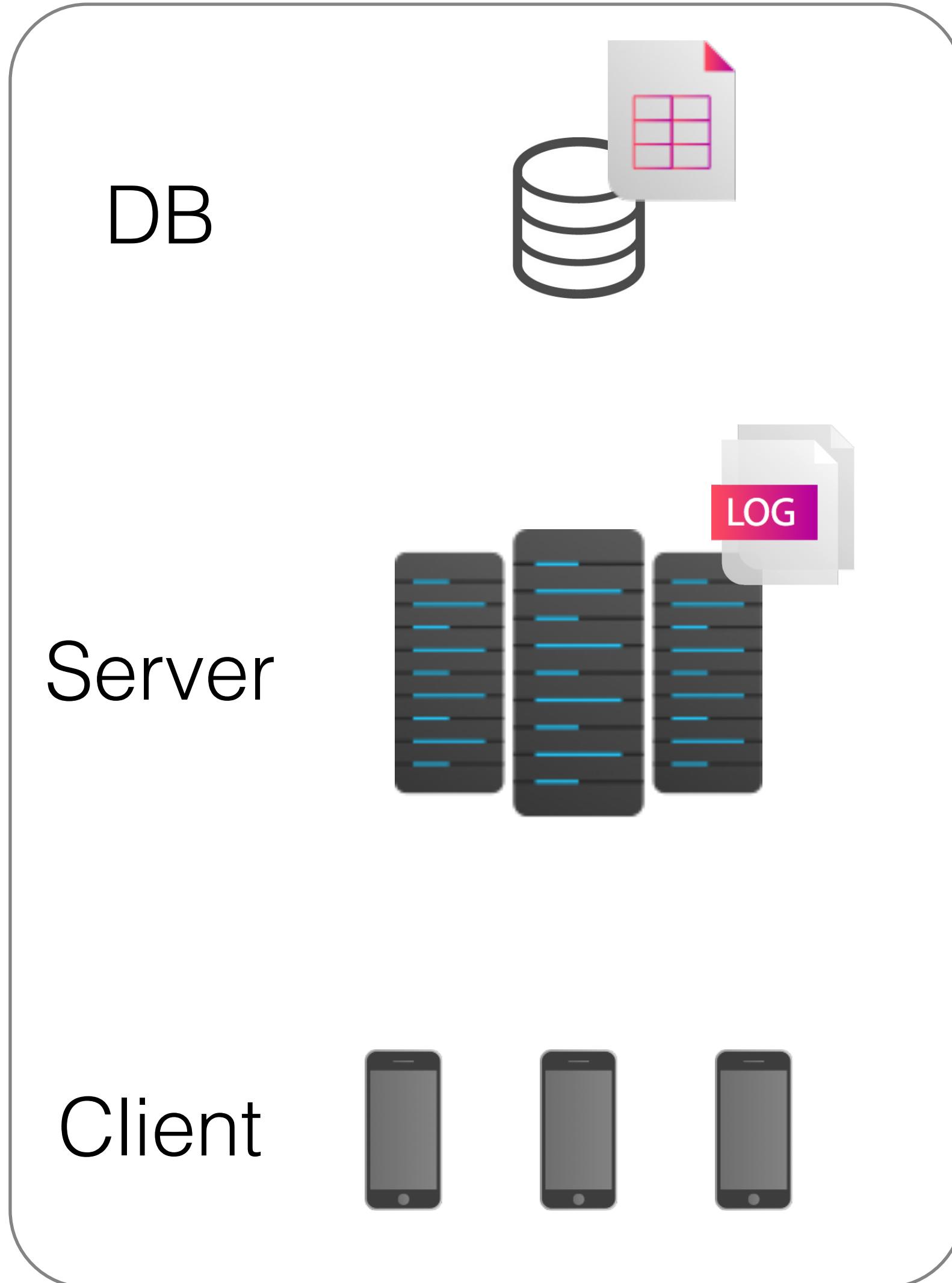


# 1. はじめに～データ分析基盤について～



# オンラインゲームシステムとデータ分析基盤の関わり

## オンラインゲームシステム

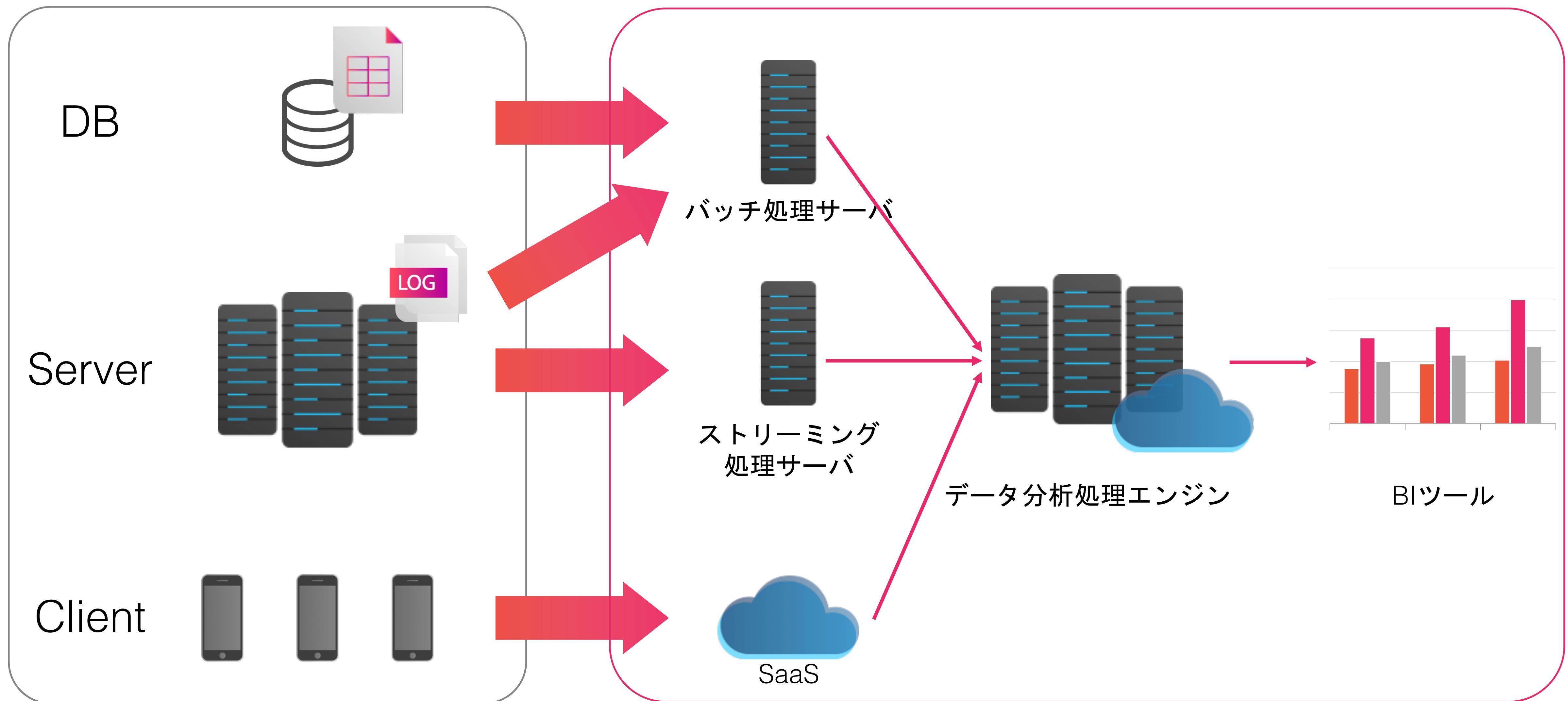




# オンラインゲームシステムとデータ分析基盤の関わり

## オンラインゲームシステム

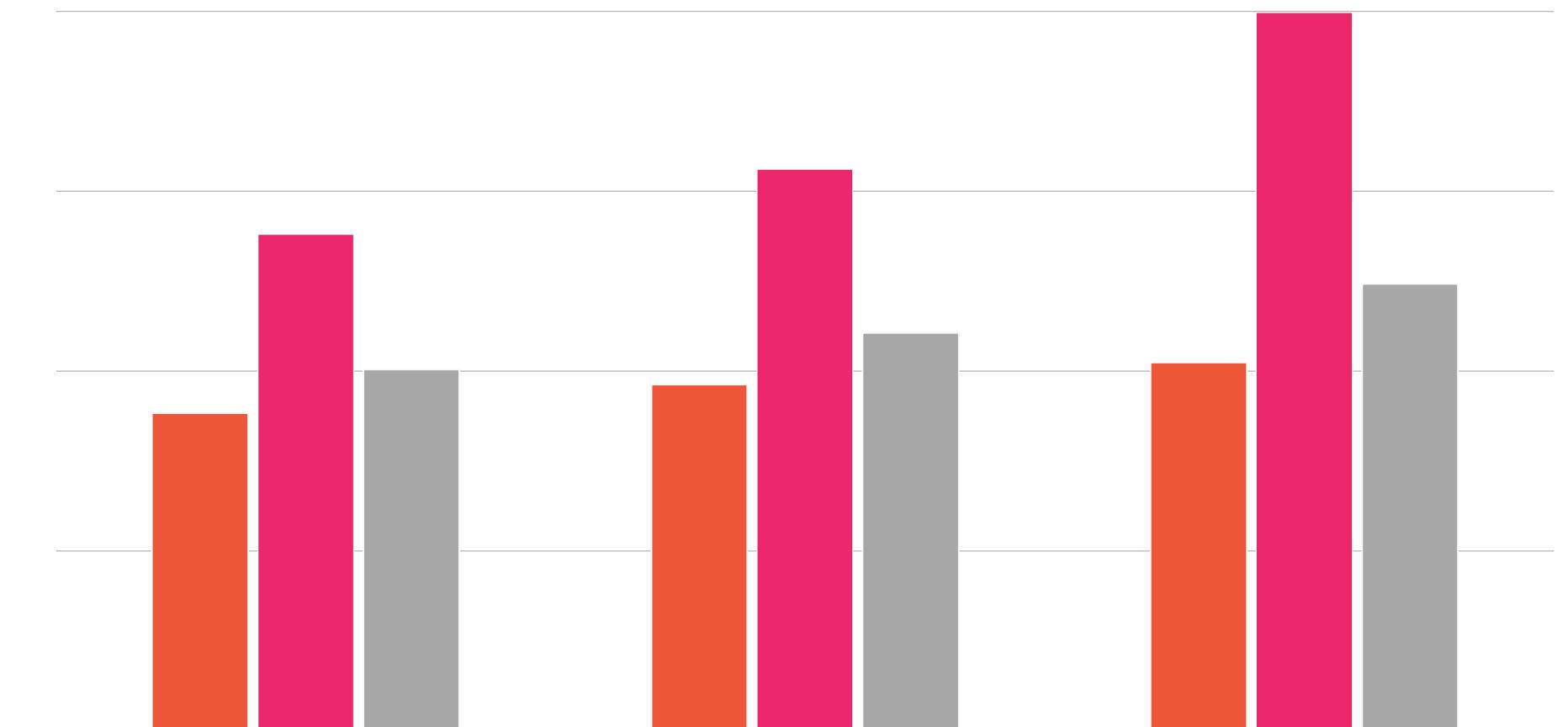
## データ分析基盤





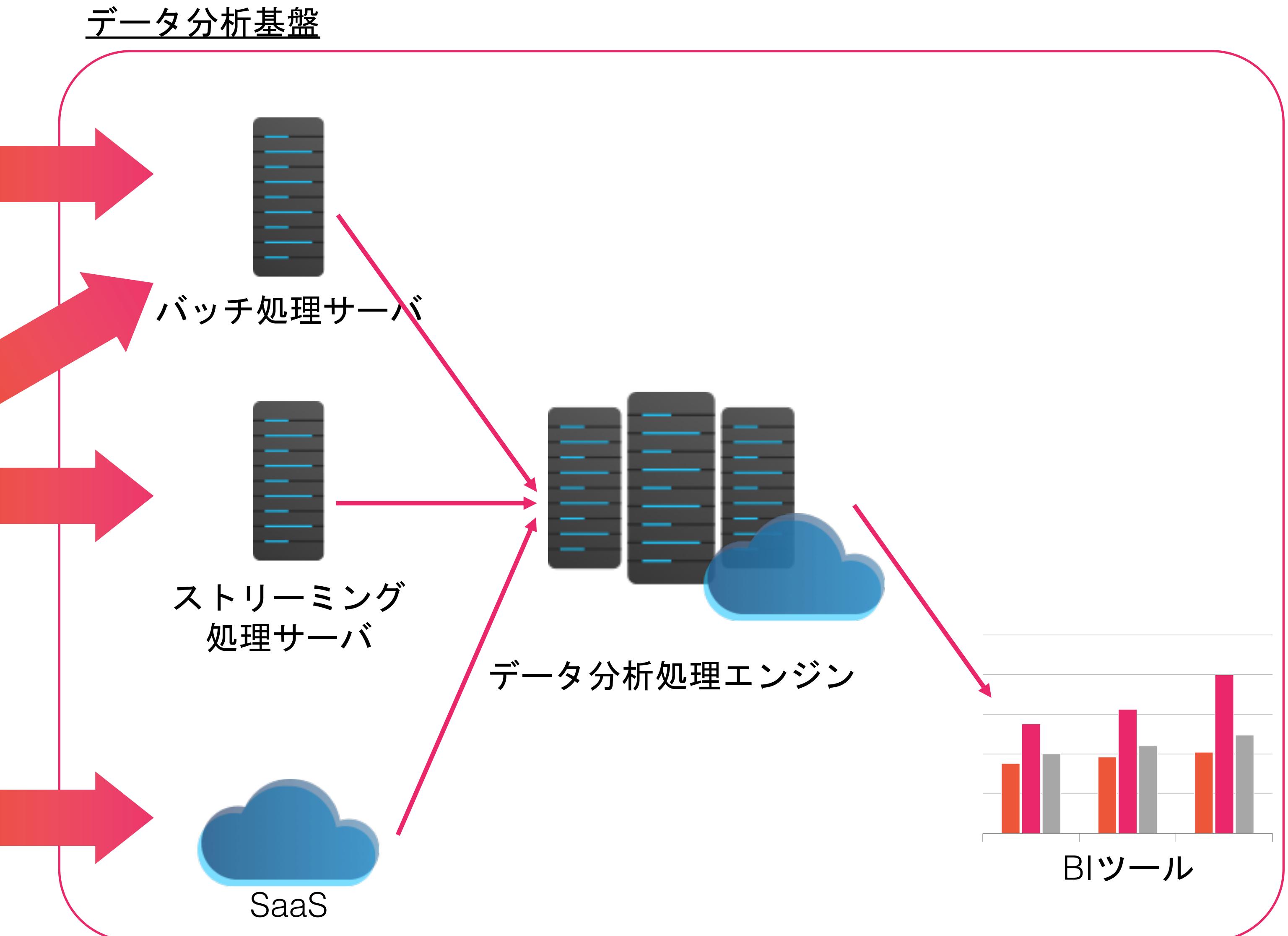
## データ分析基盤をなぜ必要なのか？

- ・ 現状を知る
  - ・ アクセスUU数や、継続率、課金額、課金率等の基本的なKPI
  - ・ イベント参加率や達成状況
  - ・ ガチャやアイテムの売れ行き
  - ・ ステータスの上位層・中位層・下位層の推移
- ・ データ分析の流れをスムーズに行う
  - ・ 顧客を特徴別に分類
  - ・ ターゲットの決定
  - ・ ターゲットの行動を見る
  - ・ 行動から要因について仮説を立てる
  - ・ 仮説を統計的に検証
- ・ データの民主化





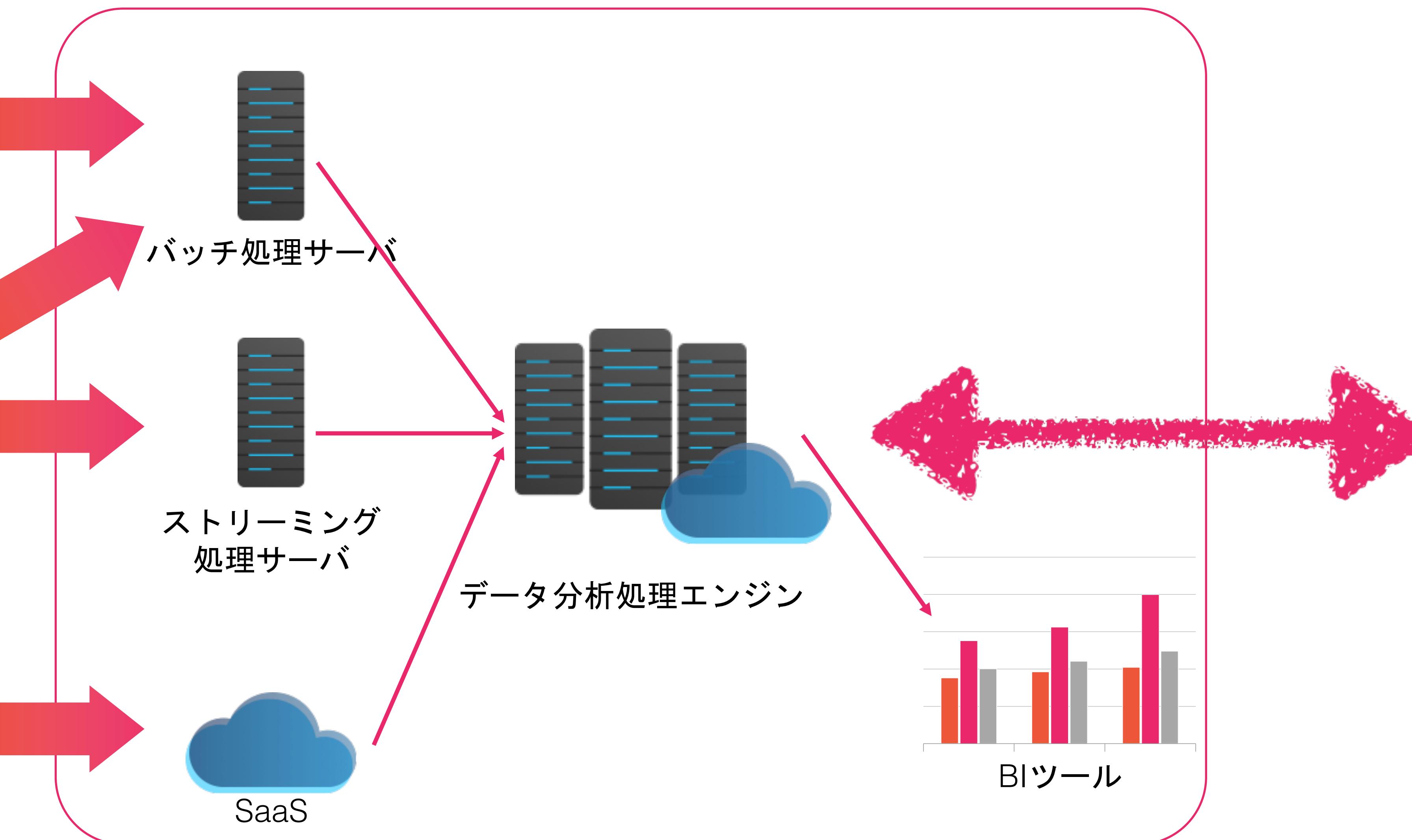
# データ分析基盤への更なる期待：3rd Partyとの連携





# データ分析基盤への更なる期待：3rd Partyとの連携

## データ分析基盤





# データ分析基盤を支えるOSS



**Fluentd**

ストリーミングログコレクタ



**Embulk**

バルクデータローダ



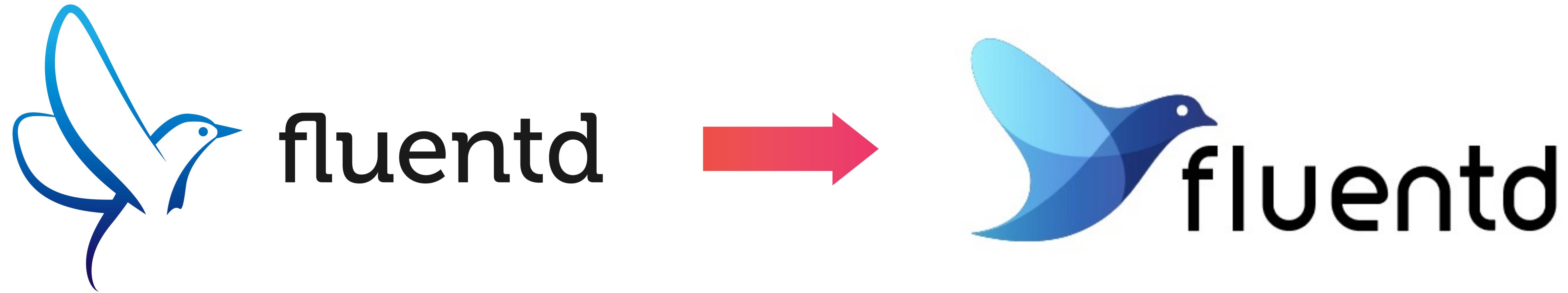
**Digdag**

ワークフロー  
オートメーションシステム

注: 画像はイメージです



お知らせ) Fluentd、ロゴかわるってよ



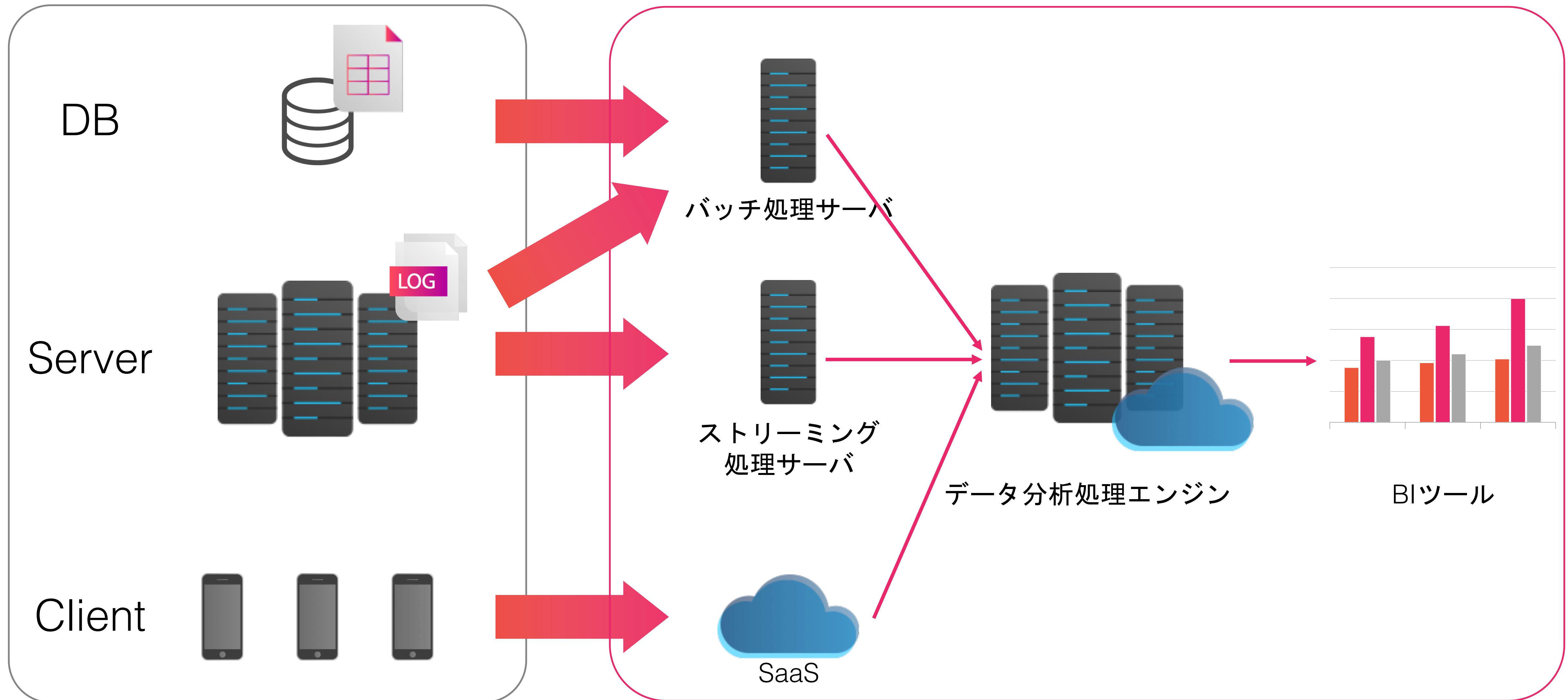
[#924 Fluentd new logo proposal](#)



# データ分析基盤を支えるOSS

オンラインゲームシステム

データ分析基盤

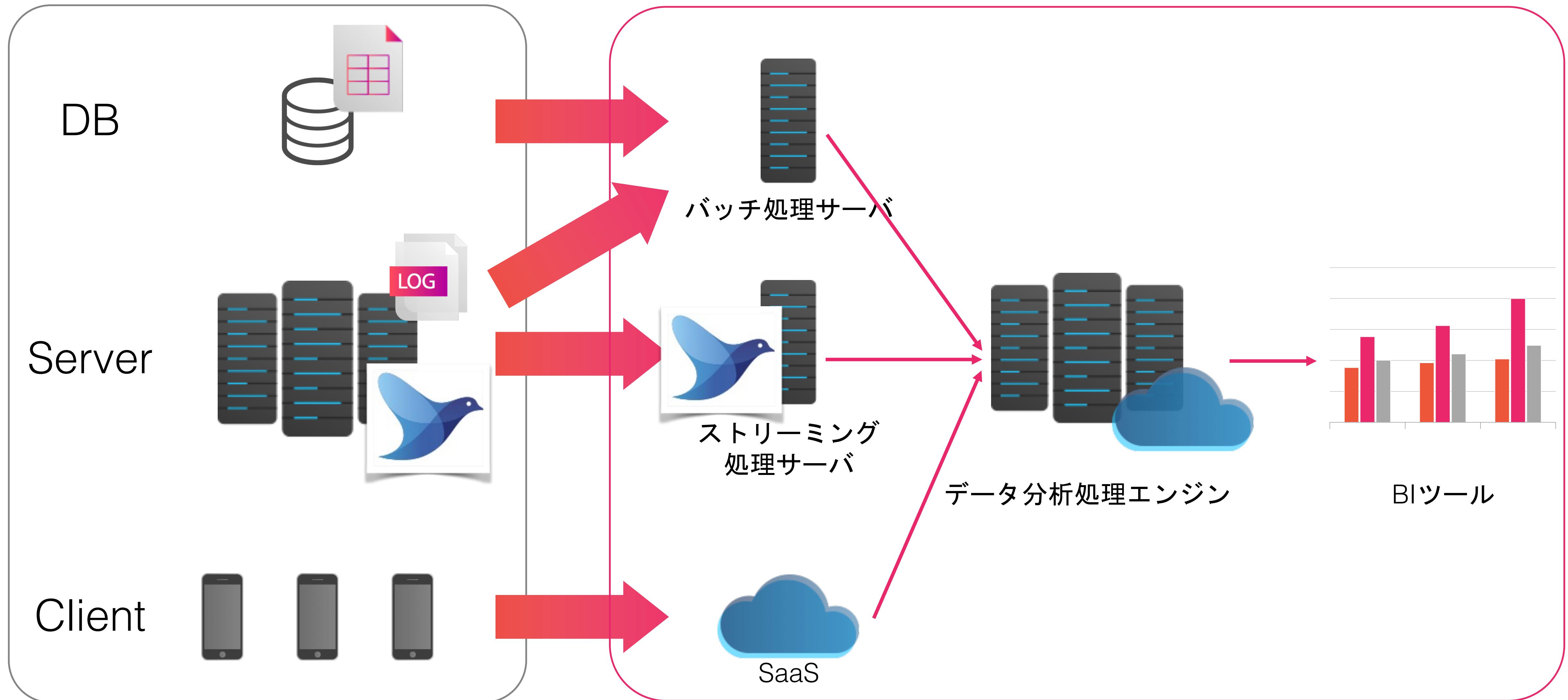




# データ分析基盤を支えるOSS

オンラインゲームシステム

データ分析基盤

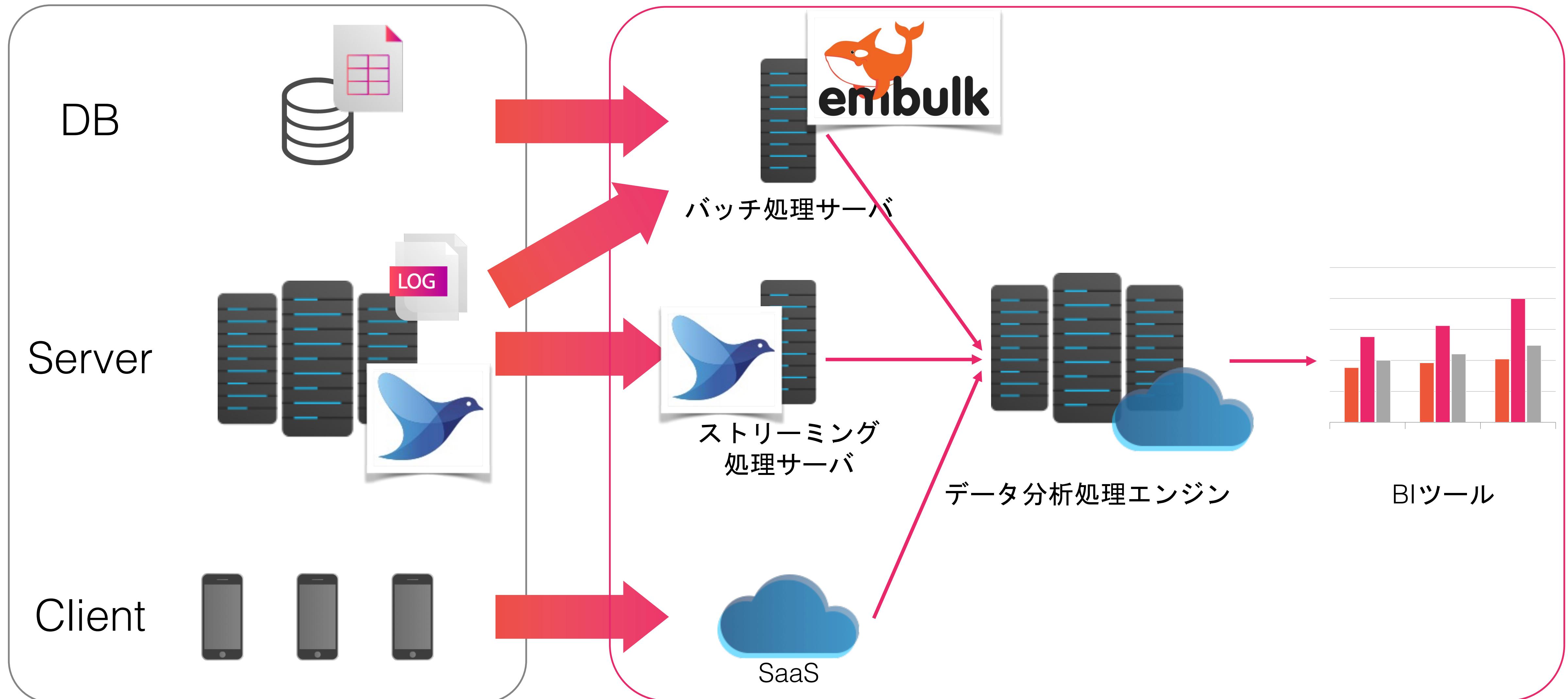




# データ分析基盤を支えるOSS

オンラインゲームシステム

データ分析基盤

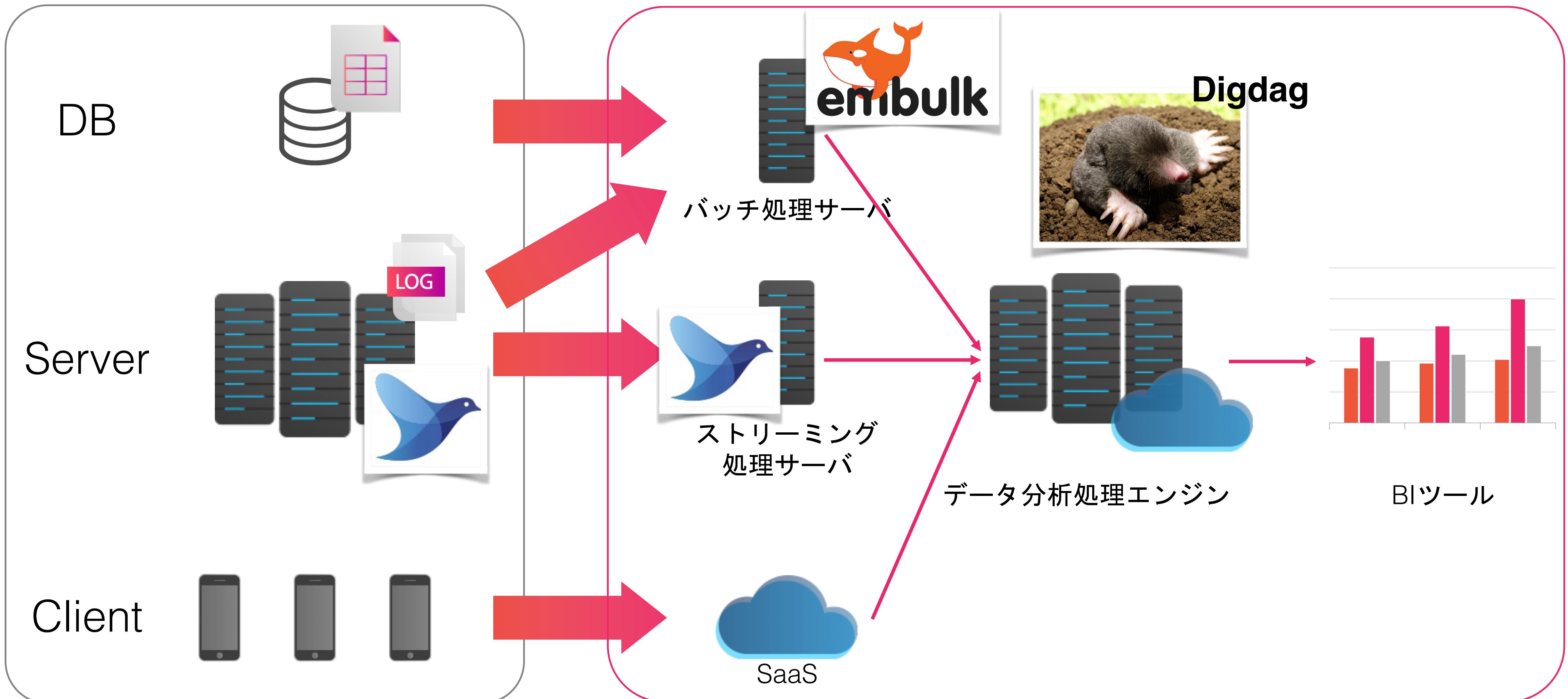




# データ分析基盤を支えるOSS

オンラインゲームシステム

データ分析基盤

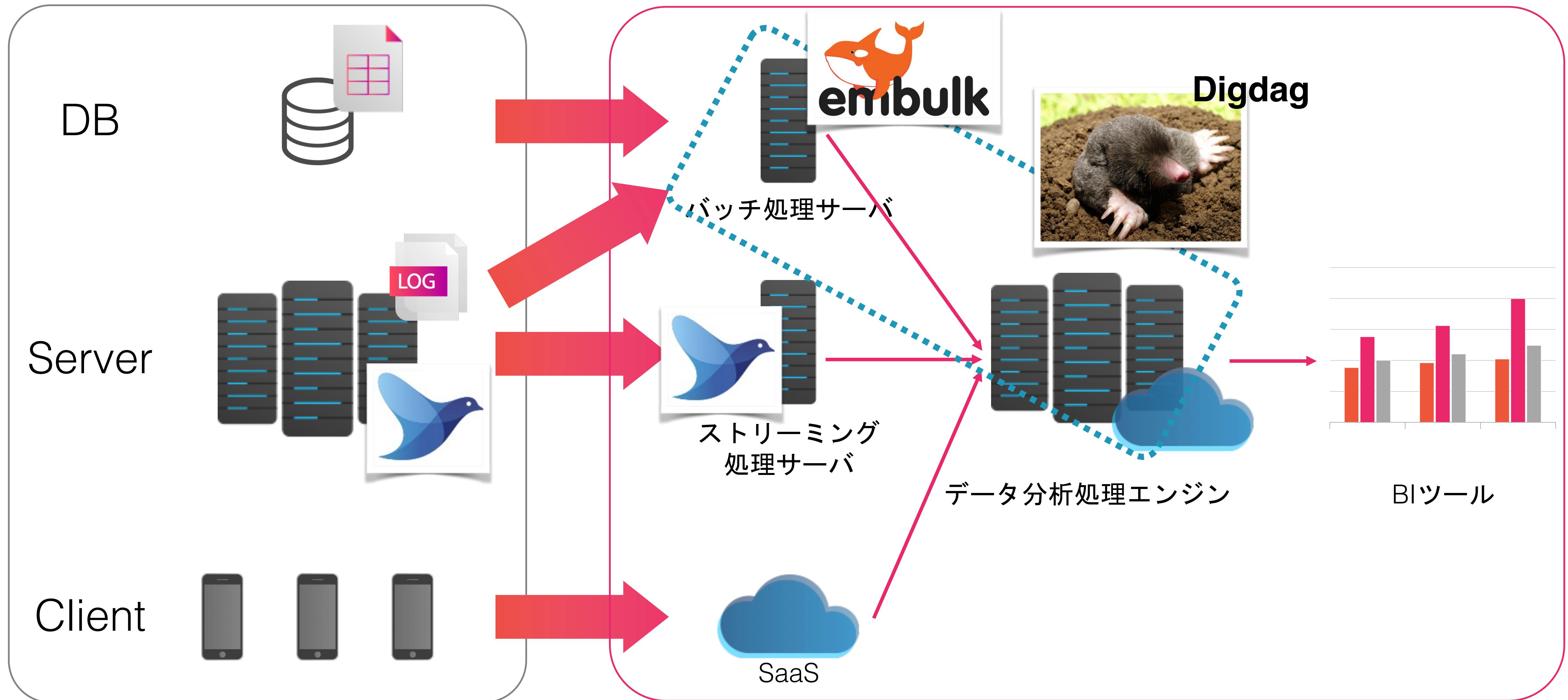




# データ分析基盤を支えるOSS

オンラインゲームシステム

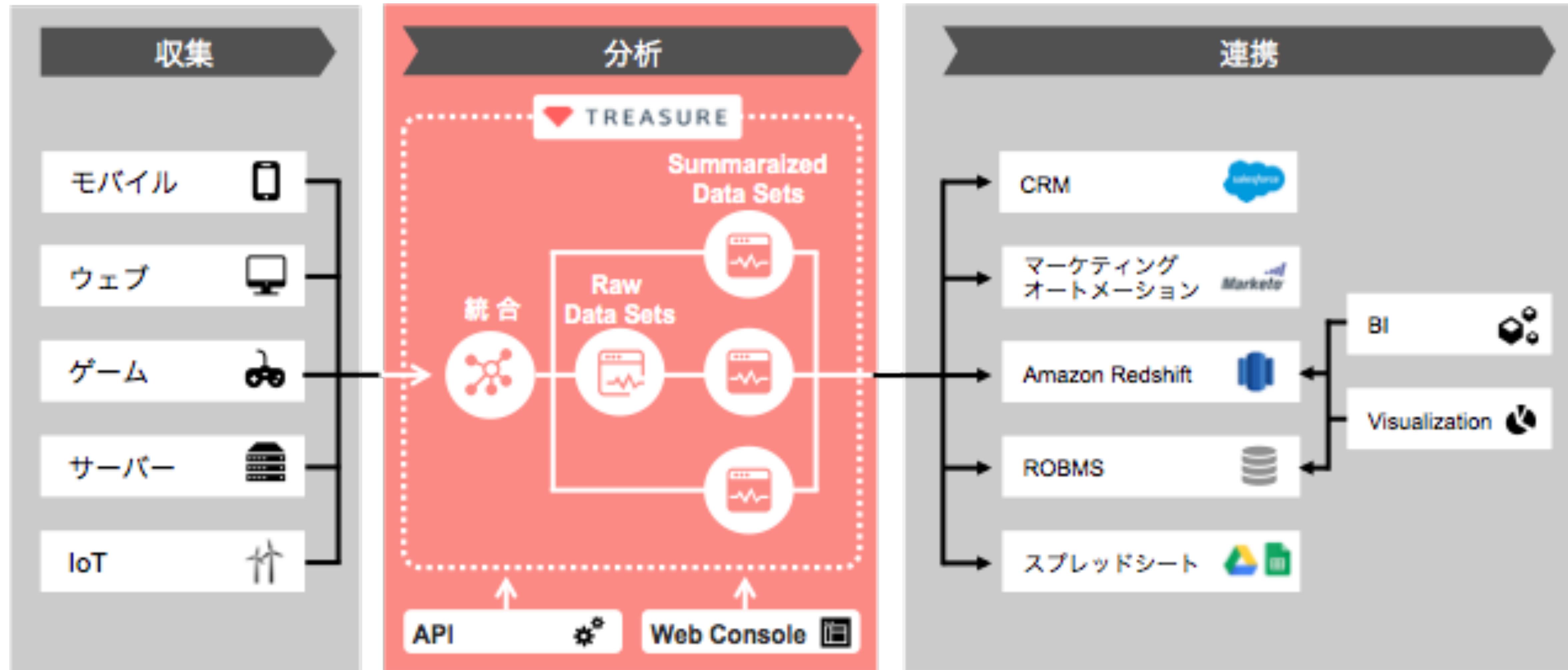
データ分析基盤





# TreasureData: データ分析基盤を一気通貫で提供

[https://www.treasuredata.com/jp/contact\\_us](https://www.treasuredata.com/jp/contact_us)





## 2. Embulk ~ Bulk Data Loader ~



## Embulkとは？ - <http://embulk.org/>

- ・オープンソースのバルク転送ツール
  - ・”A”から”B”へレコード転送
- ・プラグイン機構
  - ・多様な”A”と”B”の組み合わせ
- ・データ連携を容易に
  - ・システム構築の頭痛の種の一つ



Storage, RDBMS,  
NoSQL, Cloud Service,

...

broken records,  
error recovery,  
maintenance,  
performance, ...



## バルクデータ転送の難しさ

1. 入力データの正規化
2. エラー処理
3. メンテナンス
4. パフォーマンス



## 1. 入力データ正規化の難しさ

データエンコーディングのバリエーション

- **null**, 時刻, 浮動小数点
  - 改行, エスケープ, レコード/カラム区切り
  - 文字コード, 圧縮有無
- 試行錯誤をしてデータ正規化



## 2. エラー処理の難しさ

- 例外値の扱い
  - ネットワークエラーからの復旧
  - ディスクフルからの復旧
  - 重複データ転送の回避
- データバリデーション, リトライ, リジューム



### 3. メンテナンスの難しさ

- ・ 繼続的な動作の確保
  - ・ データ転送要件変更への対応
- ドキュメント, 汎用化, OSS化



## 4. 性能の問題

- 転送データ量は通常増えていく
  - 対象レコードも増えたりする
- 並列・分散処理



## バルクデータ転送の例 (1/3)

指定された **10GB CSV file** を **PostgreSQL** にロード

1. コマンド叩いてみる → 失敗
2. データを正規化するスクリプトを作成  
“20150127T190500Z” → “2015-01-27 19:05:00 UTC”に  
“null” → “NULL”に変換  
元データを見ながら気付く限り...
3. 再度チャレンジ → 取り込まれたが元データと合わない  
“Inf” → “Infinity”に変換
4. ひたすら繰り返す
5. うっかりレコードが重複して取り込まれた...



## バルクデータ転送の例（2/3）

指定された **10GB CSV file** を **PostgreSQL** にロード

6. スクリプトが完成
7. **cron**に登録して毎日バルクデータロードするよう登録
8. ある日、別の原因でエラーに...

不正な**UTF-8 byte sequence**をU+FFFFに変換



## バルクデータ転送の例（3/3）

過去の日次 **10GB CSV file** を **730個** を取り込む（2年分）

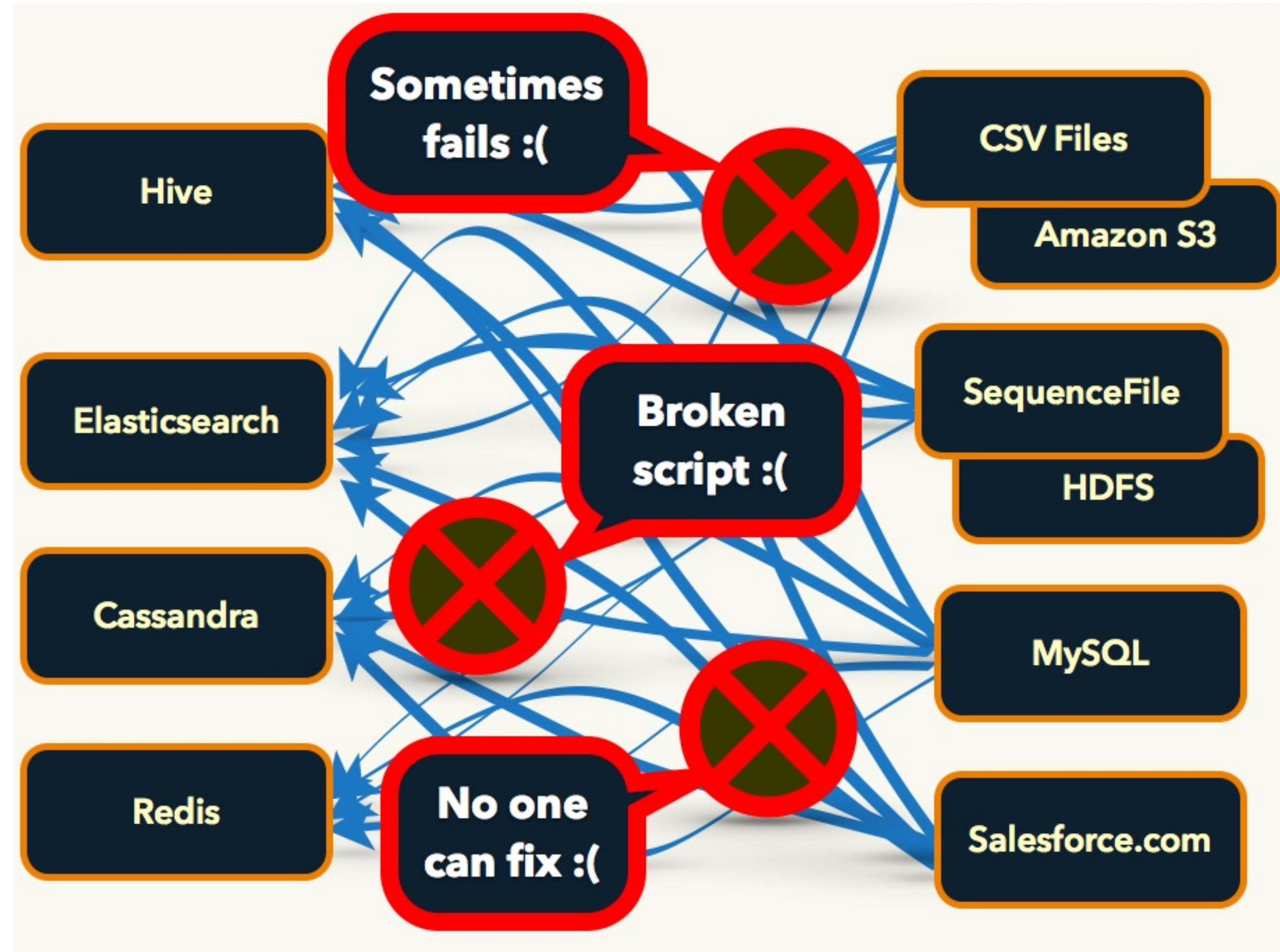
1. たいていのスクリプトは遅い  
最適化している暇がない  
1ファイル1時間、エラーが発生しなくても1ヶ月必要
2. 並列データロードするようにスクリプト変更
3. ある日、ディスクフル/ネットワークエラーで失敗  
どこまで読み込まれた？
4. 障害後に再開し易いよう、データロード単位を調整
5. 安全な再開機能をスクリプトに追加



## システム構築の頭痛の種

様々な転送データ、データストレージ

- CSV, TSV, JSON, XML, MessagePack, SequenceFile, RCFile
- AWS S3, Salesforce.com, Google Cloud Storage, Elasticsearch
- MySQL, PostgreSQL, Oracle, MS SQL Server, Amazon Redshift, Redis, MongoDB, BigQuery





# N x M scripts!

- > Poor error handling
- > No retrying / resuming
- > Low performance
- > Often no maintainers

fails :(

CSV Files

Amazon S3

SequenceFile

HDF

MySQL

Salesforce.com

Elasticsearch

Cassandra

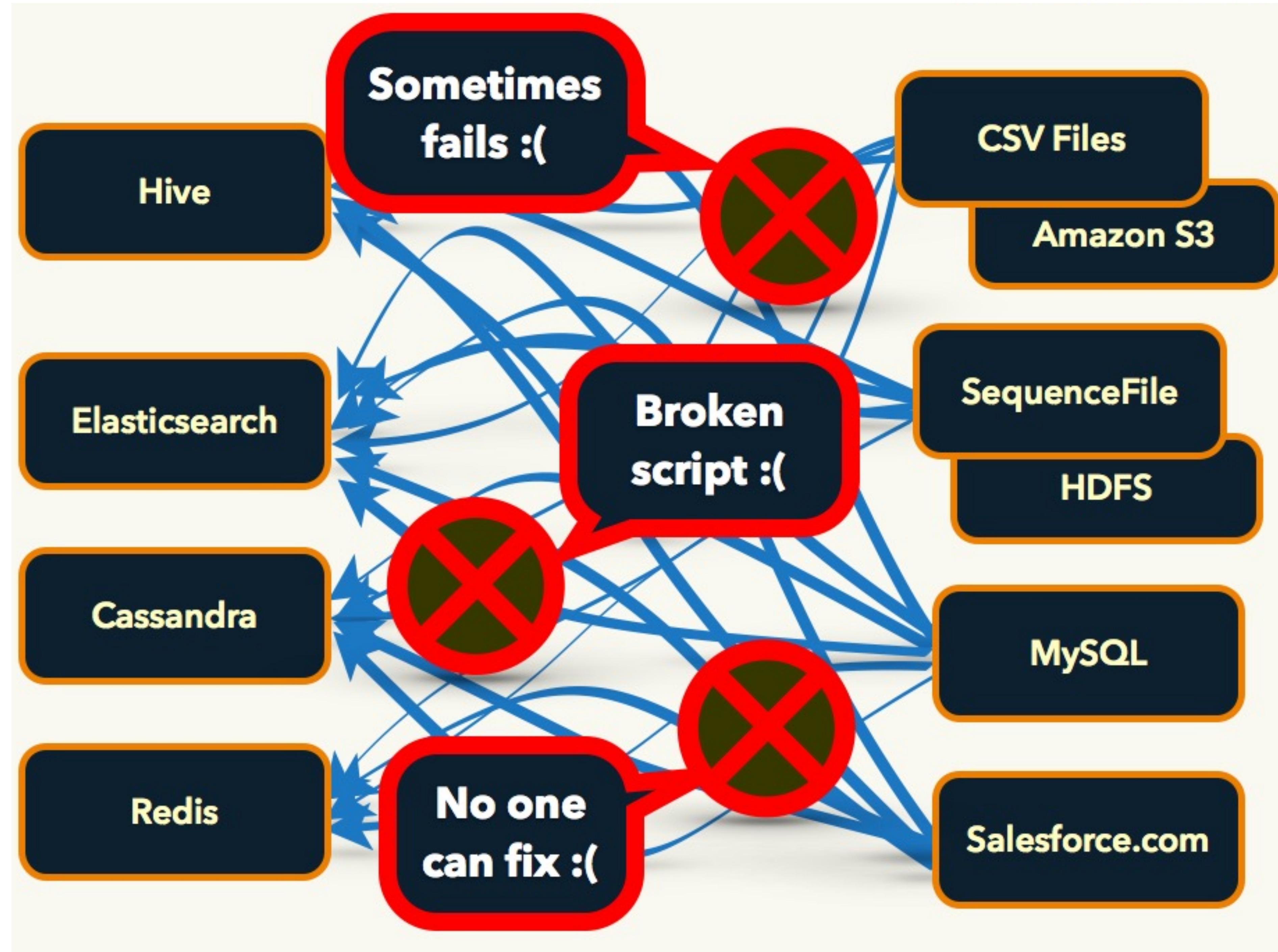
Redis

No one  
can fix :(



## Embulkのアプローチ

- ・ プラグインアーキテクチャ
- ・ 入力データ正規化支援: **guess, preview**
- ・ 並列・分散実行
- ・ 繰り返し実行
- ・ トランザクション制御





**Reliable  
framework :-)**

Hive

Elasticsearch

Cassandra

Redis

CSV Files

Amazon S3

SequenceFile

HDFS

MySQL

Salesforce.com



## プラグインアーキテクチャ

- フレームワークによる恩恵
  - 並列処理、繰り返し実行、エラー処理、リカバリ
- RubyGemとして配布 (<http://www.embulk.org/plugins/>)
  - DB
    - Oracle, MySQL, PostgreSQL, Amazon Redshift, ...
  - SaaS
    - Salesforce.com, Amazon S3, Google Cloud Storage, Google BigQuery, ...
  - File Format
    - CSV, TSV, JSON, XML, ...
    - gzip, bzip2, zip, tar, ...



# DEMO - Embulk

**Local の CSV file を PostgreSQL にロード**

(ネットワークに問題があると悲しい気持ちになるので、ローカルで全て済ませてます。)

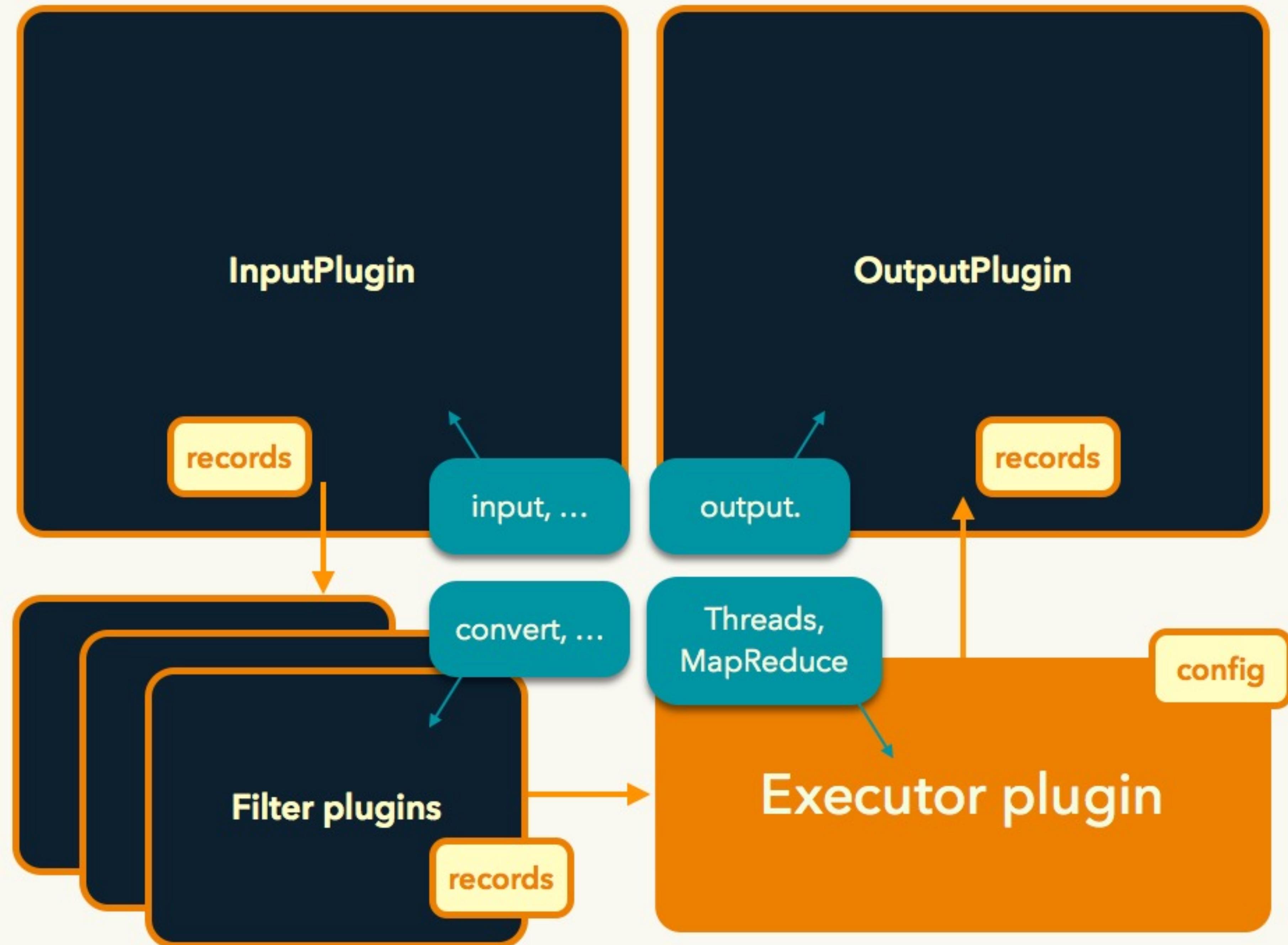
1. **guess** と **preview**
2. 実行
3. 繰り返し実行

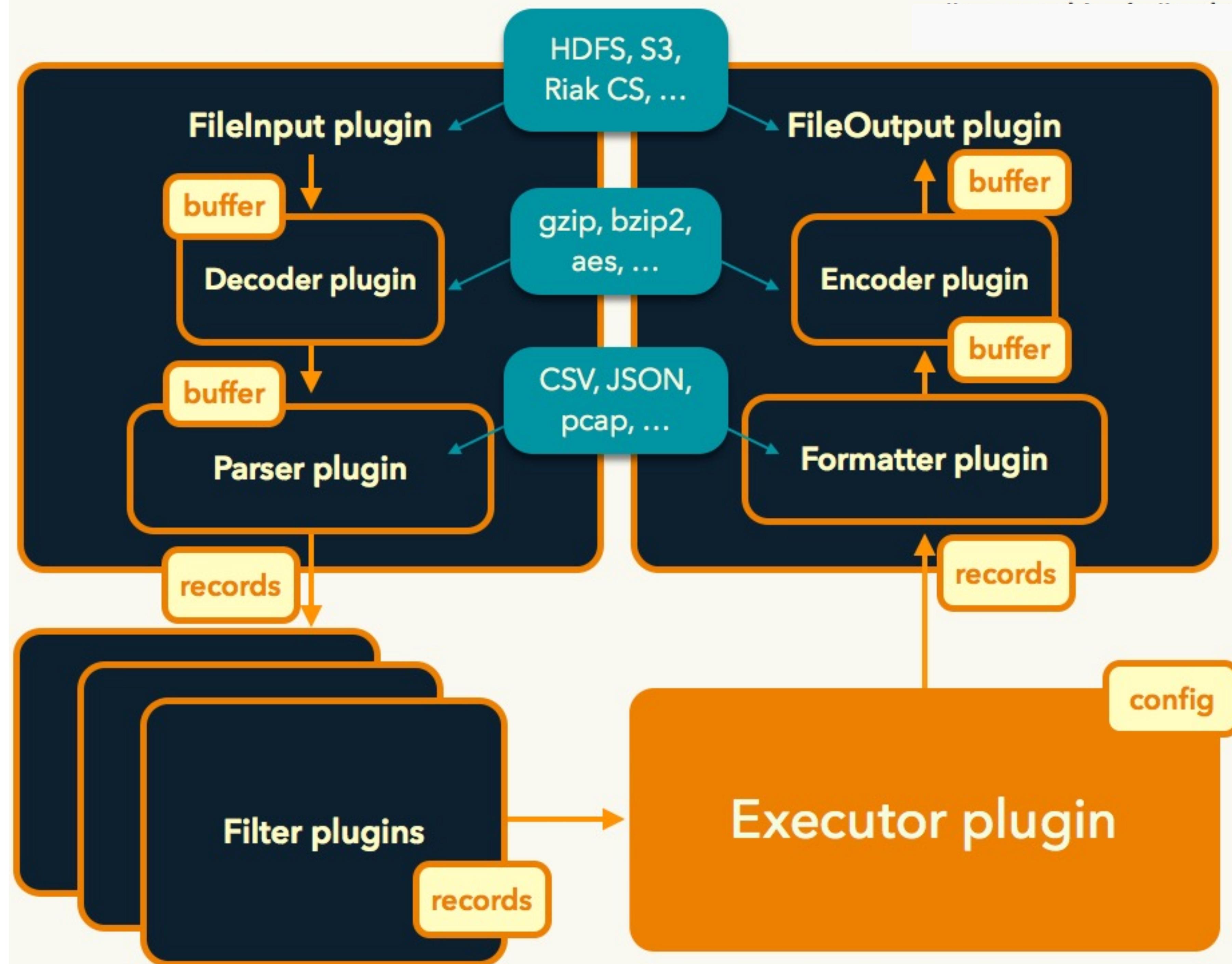




# Embulkプラグインアーキテクチャ概要

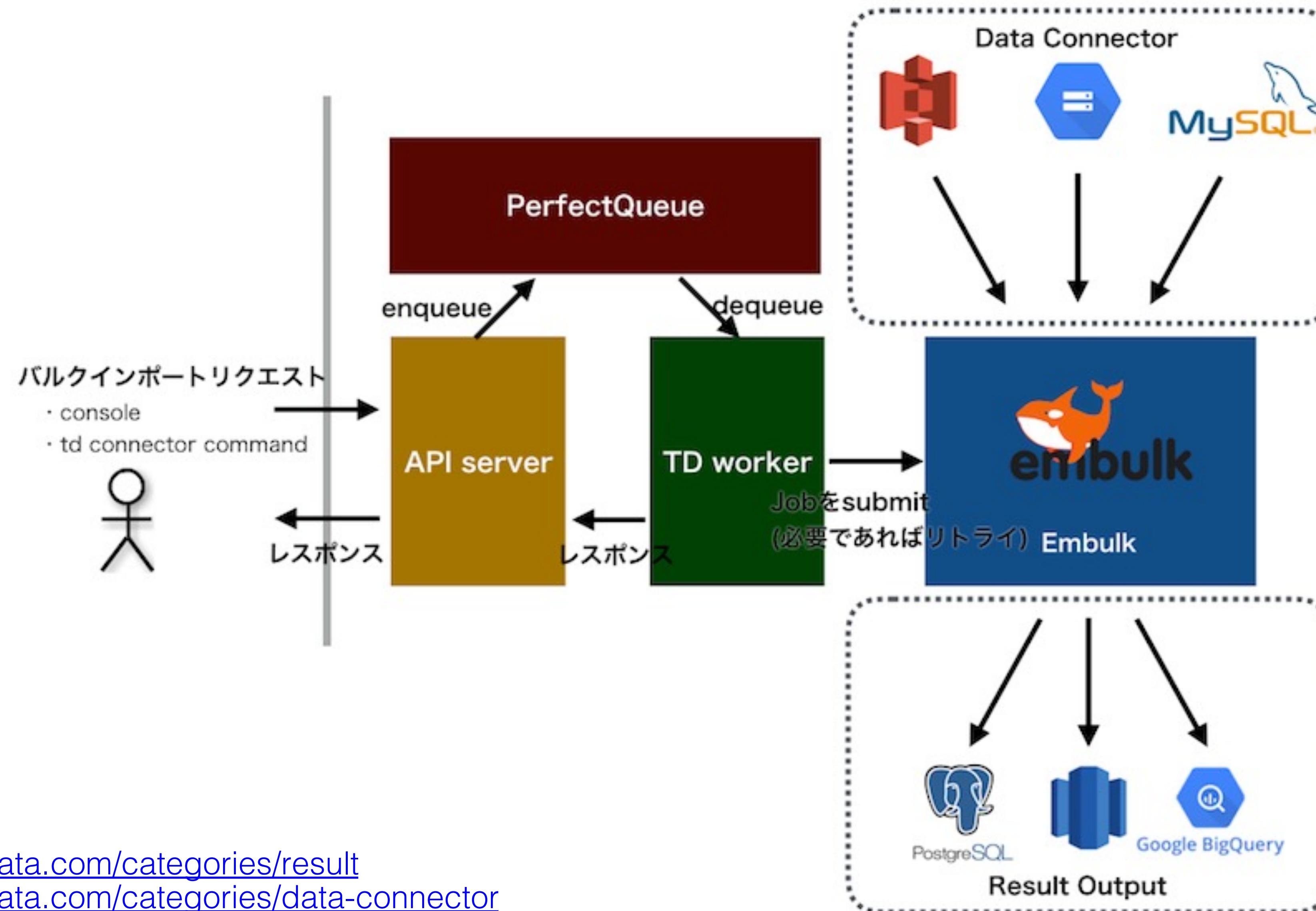
- **4種のプラグインとそれを組み上げるフレームワーク**
  1. **Executor:** 実行
  2. **Input:** バルクデータのレコード群を取り込み
    - **FileInput, Decoder, Parser:** ファイル操作
  3. **Filter:** レコードに対するデータ操作
  4. **Output:** バルクデータのレコード群を出力
    - **FileOutput, Encoder, Formatter:** ファイル操作







# TreasureDataでのEmbulkのユースケース





# Embulkで解決できないこと

1. **YAML**管理
2. 単体処理フレームワーク
  - 事前処理・事後処理
3. ジョブキュー
4. 並列数制御
5. 続きから実行
  - 処理間の依存関係

参考: Embulkに足りない5つのこと  
<https://speakerdeck.com/civitaspo/embulknizu-rinai5tufalsekoto>



# Embulkで解決できないこと

1. ~~YAML管理~~
2. 単体処理フレームワーク
  - 事前処理・事後処理
3. ジョブキュー
4. ~~並列数制御~~
5. 続きから実行
  - 処理間の依存関係

処理の一連の流れを管理する  
ところで解決すべきこと

参考: Embulkに足りない5つのこと

<https://speakerdeck.com/civitaspo/embulknizu-rinai5tufalsekoto>



### 3. Digdag ~ Workflow Automation System ~



# Workflow Automationとは？

- ・ あらゆる手作業の自動化
  - ・ 例えば...
    - ・ データ解析の自動化
      - ・ データロード → 前処理 → 複数テーブルの結合 → 集計 → レポート生成 → 通知
    - ・ テスト・デプロイの自動化 (CI: 繼続的インテグレーション)
    - ・ メール一斉通知
      - ・ メールアドレス抽出→条件ごとにフィルタ→メール送信



# Workflow Automation Systemに求められる機能

- ・ 基本機能
  - ・ タスクの順次実行
  - ・ 定期的な実行
  - ・ イベントに反応して実行
- ・ エラー処理
  - ・ 失敗したらリトライ
  - ・ 失敗したらアラート
  - ・ 実行時間が長ければアラート
  - ・ 途中からリジューム
  - ・ 幕等なりトライ
- ・ 状態監視
  - ・ タスクの依存関係の可視化
  - ・ タスクの実行ログの収集
- ・ 高速化
  - ・ タスクの並列実行
  - ・ 複数サーバの分散実行
  - ・ 同時実行数の制限 / フェアスケジューラ / プリエンプション
- ・ ワークフロー開発
  - ・ GUIベースのワークフロー定義
  - ・ ソースコードベースのワークフロー定義
  - ・ ワークフローのバージョン管理
  - ・ 多数のシステムを制御する為のプラグイン / ライブラリシステム
  - ・ どこでも同じ動作をする再現性・仮想環境内のタスク実行



# 既存のWorkflow Automation System

- OSS
  - Makefile
  - Jenkins
  - Luigi
  - Airflow
  - Rundeck
  - Azkaban
  - ...
- 商用
  - JP1 / AJS3
  - ...
- ETLツールも類似のツール
  - Talend
  - DataSpidar
  - ...



## Digdag とは？

Digdag is a simple tool that helps you to build, run, schedule, and monitor complex pipelines of tasks. It handles dependency resolution so that tasks run in order or in parallel.

Digdag fits simple replacement of cron, IT operations automation, data analytics batch jobs, machine learning pipelines, and many more by using Directed Acyclic Graphs (DAG) as the infrastructure.

<http://www.digdag.io/index.html>



ワークフローエンジンに求められる機能

= Digdagでユーザがシンプルに実現できるようにしたい機能

- 基本機能
  - タスクの順次実行
  - 定期的な実行
  - イベントに反応して実行
- エラー処理
  - 失敗したらリトライ
  - 失敗したらアラート
  - 実行時間が長ければアラート
  - 途中からリジューム
  - 幢等なりトライ
- 状態監視
  - タスクの依存関係の可視化
  - タスクの実行ログの収集
- 高速化
  - タスクの並列実行
  - 複数サーバの分散実行
  - 同時実行数の制限 / フェアスケジューラ / プリエンプション
- ワークフロー開発
  - GUIベースのワークフロー定義
  - ソースコードベースのワークフロー定義
  - ワークフローのバージョン管理
  - 多数のシステムを制御する為のプラグイン / ライブラリシステム
  - どこでも同じ動作をする再現性・仮想環境内のタスク実行



## DEMO: DIGDAG (LOCAL MODE)

Local の CSV file を PostgreSQL にロードし、  
SQLで集計した結果をslackで通知。  
(+ ローカルでスケジューリング)



Digdag \*画像はイメージです



# mydag.dig

```
{}
timezone: UTC

_export:
rb:
require: 'tasks/myworkflow'

+dataload:
embulk>: demo/config.yml タスク

+counting:
rb>: MyWorkflow.pg_calc

+notify:
py>: tasks.MyWorkflow.slack
```

# mydag.dig - Parallel

```
{}
timezone: UTC
_export:
rb:
require: 'tasks/myworkflow'

+dataload:
embulk>: demo/config.yml

+calculation:
_parallel: true
_export: ....
+counting:
rb>: MyWorkflow.count
+summarizing:
rb>: MyWorkflow.summary

+slack:
py>: tasks.MyWorkflow.slack
```

タスクの中にタスクを作つてグループ化  
並列実行

# mydag.dig - Parallel

```
{}
  timezone: UTC
  _export:
    rb:
      require: 'tasks/myworkflow'

  +dataload:
    embulk>: demo/config.yml

  +calculation:
    _parallel: true
    _export: ....
  +counting:
    rb>: MyWorkflow.count
  +summarizing:
    rb>: MyWorkflow.summary

  +slack:
    py>: tasks.MyWorkflow.slack
```

タスク内だけで適用される設定  
並列実行



# Operators

- call>: Calls another workflow
- require>: Depends on another workflow
- py>: Python scripts
- rb>: Ruby scripts
- sh>: Shell scripts
- loop>: Repeat tasks
- for\_each>: Repeat tasks
- if>: Conditional execution
- fail>: make the workflow failed
- mail>: Sending email
- embulk>: Embulk data transfer
- td>: Treasure Data queries
- td\_run>: Treasure Data saved queries
- td\_load>: Treasure Data bulk loading
- td\_ddl>: Treasure Data operations
- td\_table\_export>: Treasure Data table export to S3

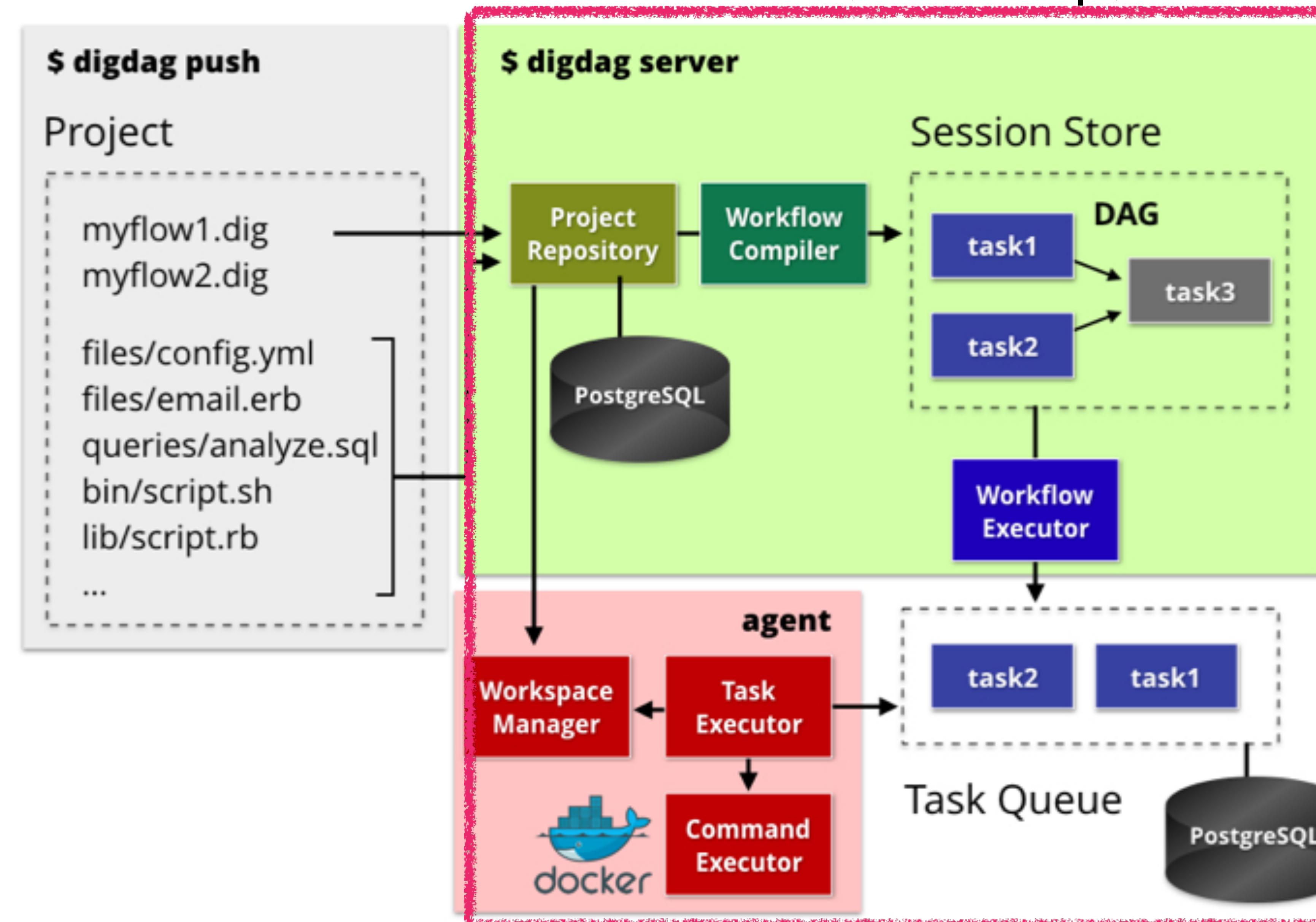
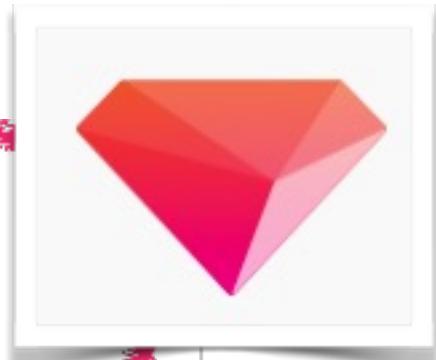


# Commands and Mode

- Local-mode commands
  - new
  - run
  - check
  - scheduler
  - selfupdate
- Server-mode commands
  - server
- Client-mode commands
  - start
  - retry
  - log
  - kill
  - workflows
  - schedules
  - backfill
  - reschedule
  - sessions
  - attempts
  - tasks
  - push



# TreasureDataでのユースケース (Privateβ)





# TreasureDataでのユースケース（Privateβ）

The screenshot shows the TD Workflows interface with the 'Projects' tab selected. The top navigation bar includes the TD logo, 'TD Workflows', 'Projects', '0.8.1', and a 'Logout' link. The main content area displays a table of projects with columns for Name, Updated, and Revision.

Name	Updated	Revision
test	a day ago	64d5f51c-5d98-45b6-95a0-6e3947ea33b7
kzk_nasdaq_analysis	2 months ago	1
poc_se_email	2 months ago	2
churn_workflow	2 months ago	2
salesforce_workflow.yml	2 months ago	1
integrations.yml	2 months ago	1
integrations	a month ago	2
nasdaq_analysis	a month ago	1
mail-test	a month ago	4

## Projects

Name	Updated	Revision
test	a day ago	64d5f51c-5d98-45b6-95a0-6e3947ea33b7
kzk_nasdaq_analysis	2 months ago	1
poc_se_email	2 months ago	2
churn_workflow	2 months ago	2
salesforce_workflow.yml	2 months ago	1
integrations.yml	2 months ago	1
integrations	a month ago	2
nasdaq_analysis	a month ago	1
mail-test	a month ago	4

## Sessions

ID	Project	Workflow	Revision	Session Time	Last Attempt	Status
8964	churn_workflow	churn_workflow	2	2016-06-16T00:00:00-07:00	7 minutes ago	<span>Failure</span>
8928	poc_se_email	poc_se_email	2	2016-06-16T00:00:00-07:00	2 hours ago	<span>Failure</span>
8893	integrations	integrations	2	2016-06-17T00:00:00+00:00	4 hours ago	<span>Success</span>



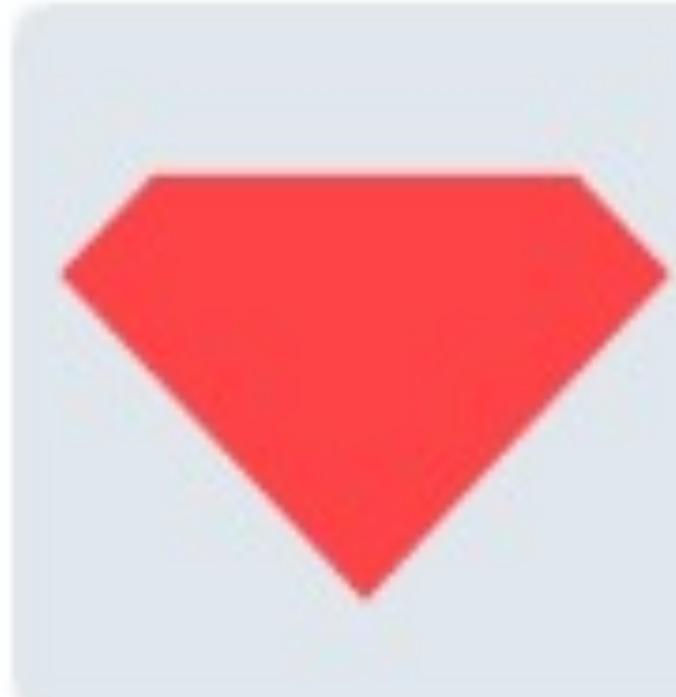
Sadayuki Furuhashi

@frsyuki



フォロー中

Digdag が Apache License 2.0 の元でオープンソース化されましたよ！ さあ試すんだ...！ 今すぐにでも！ [github.com/treasure-data/...](https://github.com/treasure-data/digdag) ドキュメント：[digdag.io](http://digdag.io)



treasure-data/digdag

Workflow Engine. Contribute to digdag development by creating an account on GitHub.

[github.com](https://github.com)

46

リツイート

39

いいね





## まとめ

- ・ オンラインゲーム・ソーシャルゲームに、データ分析基盤の存在が当たり前に！
  - ・ 分析だけじゃない役割も求められるようになってきた。
- ・ データ分析基盤にはデータの流れを作るOSSが大事！
  - ・ Fluentd
  - ・ Embulk
  - ・ Digdag
- ・ Embulkはバッチでデータ転送の役割をもち、プラグインで色んなデータソースに対応できる！
- ・ Digdagはバッチ処理のワークフローを簡単に扱うためのオートメーションシステム！
- ・ リンク集はこちら。
  - ・ [Fluentdのバッチ版Embulk\(エンバルク\)のまとめ](#)
  - ・ [ワークフローエンジンDigdagのまとめ](#)



TREASURE DATA

# Q&A

(あとで質問を思いついたら、お気軽に[@nora96o](#)までmentionしてください。)