



BigData

2017年5月18日

# Redshiftまでの大規模データフロー制御

---

NTT  
**docomo**

サービスイノベーション部 ビッグデータ担当  
鈴木浩之

## 所属/氏名

NTTドコモ サービスイノベーション部 ビッグデータ担当  
鈴木浩之（素人エンジニア）

## 主な業務

社内ビッグデータ基盤の開発&運用

## 好きなAWSサービス

Glue（まだ出てない）

## 趣味



- ① どんなシステムで？
- ② ETLについて（HOW?）
- ③ 苦労ポイント + 課題



良い事例を  
教えてください

## ① どんなシステムで？



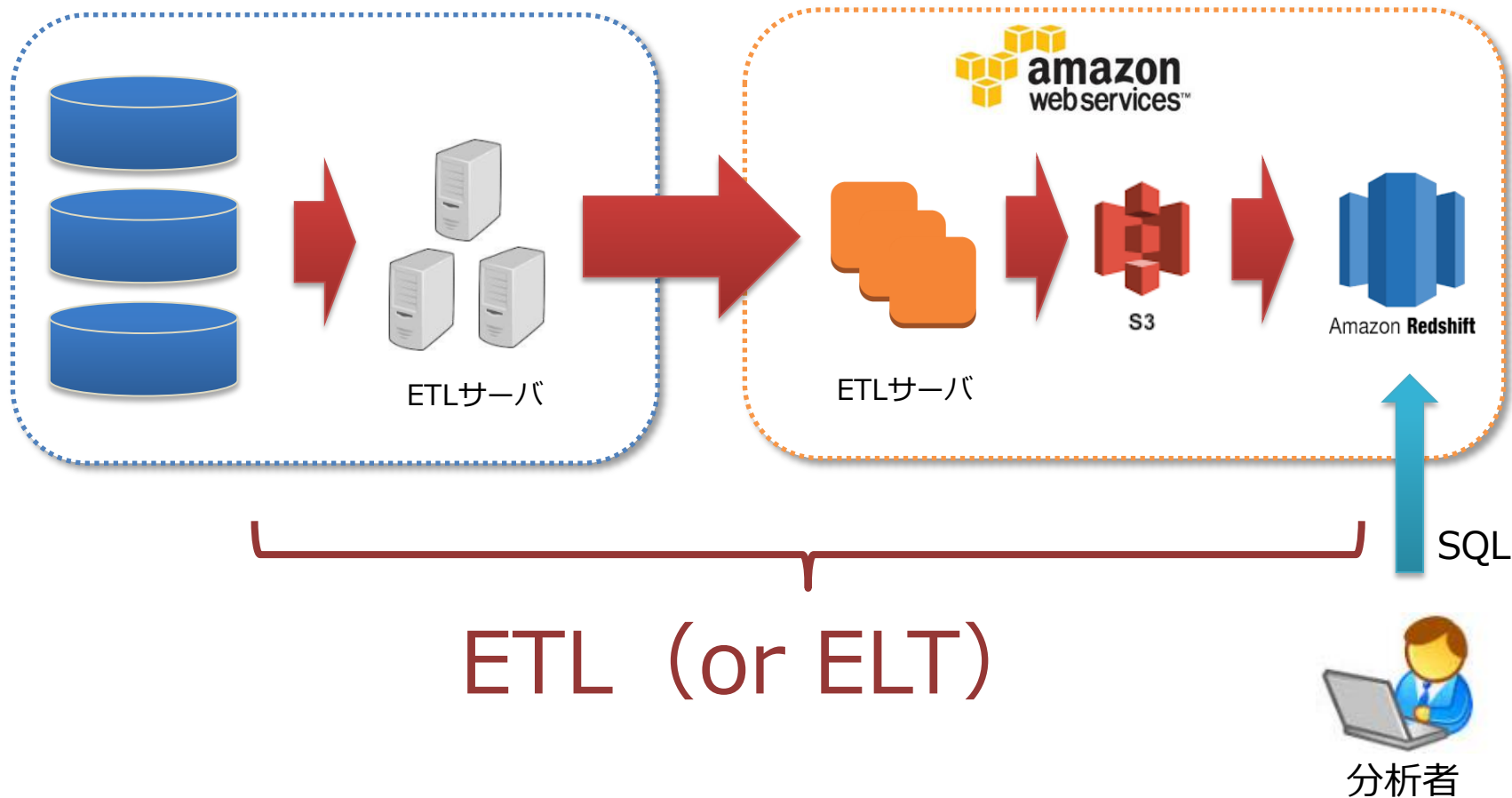
- ビッグデータ統合分析システム
- オンプレ + AWSクラウド
- 社内分析者 数百名
- データサイズ 数PB
- 毎日 数十TB
- 内製（ピザ2枚人程度？）

# ① どんなシステムで？

## ビッグデータ統合分析システム

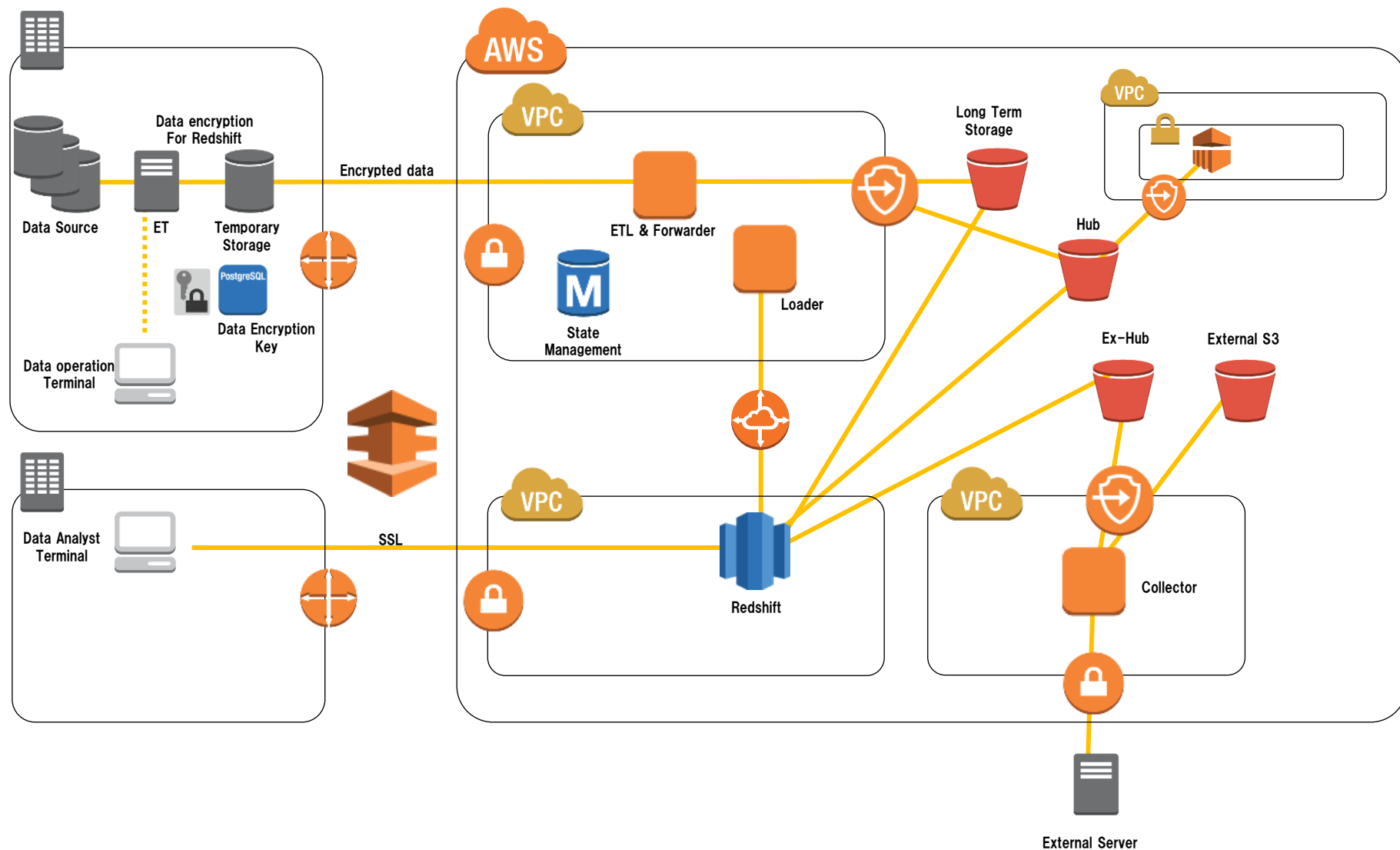
オンプレミス


パブリッククラウド



# ① どんなシステムで？

## 構成概略図





## ② ETLについて

# ETLツール

(ワークフローマネジメントツール)

- データ取得から加工、DB投入までのジョブとそのフロー制御
- 昨今色々出てきたが、当時内製で構築



digdag



nifi



などなど



## ② ETLについて (参考)

おすすめあれば  
教えてください

You Tube #ALLTHE MOMS

検索

### Products

#### OSS

- Makefile
- Jenkins
- Luigi
- Airflow
- Rundeck
- Azkaban
- Grid Engine
- OpenLava
- Obsidian Scheduler
- Hinemos
- Platform LSM

#### Proprietary

- Tivoli Workload Scheduler (IBM)
- CA Workload Automation (CA Technologies)
- JP1/AJS3 (Hitachi)
- Systemwalker Job Workload Server (Fujitsu)
- Workload Automation (Automatic)
- BatchMan (Honico)
- Control-M (BMC)
- Schedulix
- ServiceNow Workflow



【基調講演】分散ワークフローエンジン

『Digdag』の実装 古橋 貞之氏

参考 : [https://youtu.be/OAaFr\\_CnXvk](https://youtu.be/OAaFr_CnXvk)

## ② ETLについて（参考）

独自実装は案外世の中も同じ（？）



The screenshot shows a YouTube video player interface. The video is titled "AWS re:Invent 2016: NEW LAUNCH! Introduction to AWS Glue: A Fully Managed ETL Service (BDA209)". The video content shows a speaker on stage with a large screen behind them displaying the text: "The problem is 70% of ETL jobs are hand-coded With no use of ETL tools." The video player includes standard controls like play, pause, volume, and a progress bar showing 1:46 / 24:26. Below the video, the channel name "Amazon Web Services" is visible, along with a "チャンネル登録" (Subscribe) button and a subscriber count of "11万".

AWS re:Invent

amazon  
aws

The problem is

70% of ETL jobs are hand-coded

With no use of ETL tools.

1:46 / 24:26

AWS re:Invent 2016: NEW LAUNCH! Introduction to AWS Glue: A Fully Managed ETL Service (BDA209)

Amazon Web Services

チャンネル登録 11万

参考 : <https://www.youtube.com/watch?v=4N ktE4NFIk&feature=youtu.be>

# ETLツール

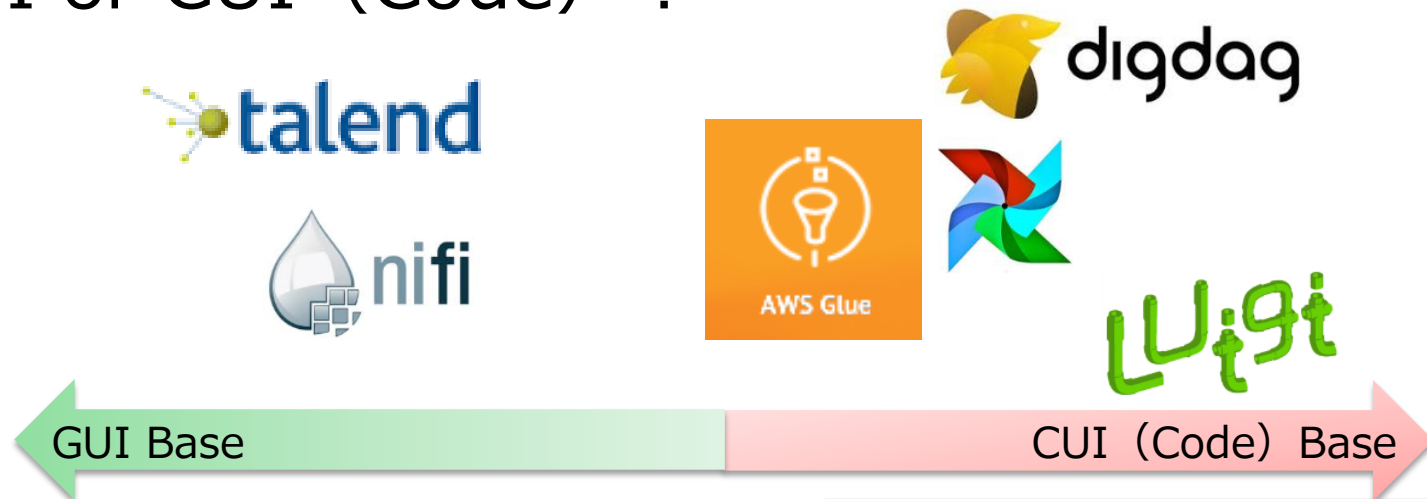
(ワークフローマネジメントツール)

## 欲しい機能例

- データ種別ごとにタスク（処理）を定義
- 決められたタスクを順番に実行
- ジョブの実行トリガー制御（スケジューラ）
- 並列分散実行制御
- 並列分散時のサーバ環境共通管理
- 失敗したら自動リトライ
- 実行状況を確認できる（モニタリング）
- データ項目定義の管理（データカタログ）
- ジョブとワークフローの履歴管理
- ハイブリッド環境(オンプレ + クラウド)対応
- :
- OSSで

# ETLツール

GUI or CUI (Code) ?



## 個人的な理想形

- モニタリングはGUI
- 処理部やフロー設定管理はCUI (Codeで管理)  
※全体フローのデザイン・確認はGUI

究極は完全自動リカバリで  
モニタリング不要にしたい

- ✓ 処理の自由度はCUI (Code) > GUI
- ✓ 元GUI推奨派だったが、GUIも結構学習コストが高い
- ✓ 履歴管理 (git) やフローの複製にはCodeがBetter

### ETL処理例

#### ◆ データ取得

- 対向FTPサーバor対向S3バケットからの定期GET (cron)  
※基本的には構造化データ (CSV/TSV) 中心

#### ◆ データ加工

- データ暗号/復号 (独自暗号、CSE)
- バイナリ⇔テキスト変換
- 特定カラム(不要 or 機密項目)の削除、ハッシュ化
- ID変換
- テーブル結合
- テーブル構成変更・分割 ※列数が多い場合など
- etc.

#### ◆ データロード

- RedshiftへのCOPY  
※オンプレ環境からは、EC2が仲介しS3アップロード後対応

## ② ETL関連の構成要素



独自実装



独自実装

独自実装



- ジョブ定期実行のスケジューラ
- ETL処理プログラムの自動テスト〈CI〉
- ETL処理プログラムの自動デプロイ（実行サーバ配布）〈CI〉
- ETL処理プログラムの履歴管理
- ETL処理プログラムの実行環境（仮想化）
- docker仮想環境のオーケストレーション
- ワークフロー型ETLジョブの指示（集中制御）
- ETLプロセス（コンテナ）停止時の自動リカバリ
- ETLプロセス（コンテナ）の負荷分散制御
- ETLジョブのログの集中管理
- ETL処理プログラム（タスク内容）の記述
- ETLワークフロー制御
  - タスク実行の順序制御
  - エラー検知とリトライ（冪等性ベース）
- ETLステータスのモニタリング
- チケット管理

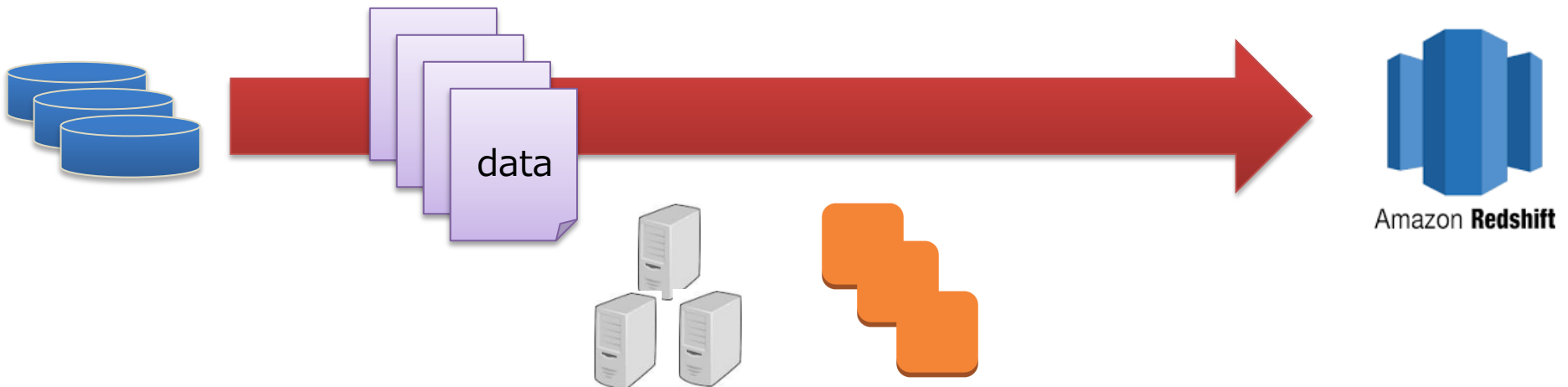
## ② ETL関連の基本フロー

定期実行トリガー

ジョブフロー制御

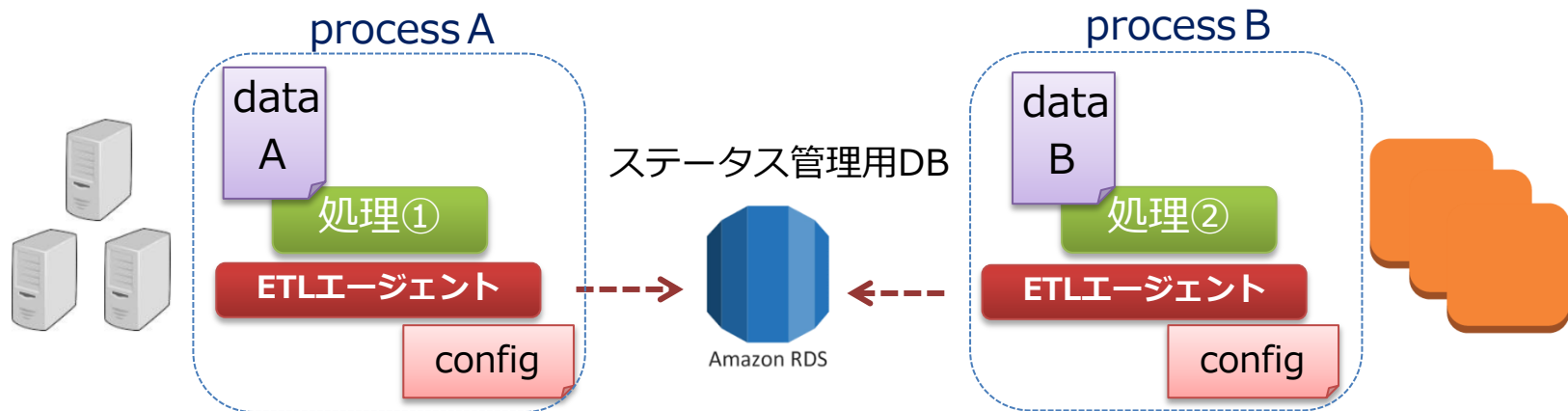


例



## ② 内製ETLワークフローツールについて

- 各サーバでETLエージェントサービスを起動
- エージェントconfigにタスク・処理フローを規定
- 各ETLエージェントはconfigを基に自身のタスクを順次実行
- 具体的なタスク内容（処理）は別プログラムでプラグイン化
- 各タスク実行状況はステータス管理用DB（RDS）で集中管理



ステータス管理テーブルイメージ

| Data   | Status | Error Flag | Task |
|--------|--------|------------|------|
| data A | 処理中    | 0          | 処理①  |
| data B | 処理待ち   | 1          | 処理②  |
| :      |        |            |      |

ETLエージェントconfigイメージ

**Data :**  
data A

**Task :**  
処理① [スクリプトパス]  
処理② [スクリプトパス]





# digdag

最新ver. : 0.9.10

- Treasure Data社が開発し、Apache License 2.0でOSS化された分散ワークフローエンジン
- Java環境で稼働
- クライアントサーバモデルで動作
- bash、python、rubyなど各種言語対応
- タスクは設定ファイル（digファイル）で規定し、digdag serverへpushすることでデプロイ（バージョン管理機能）
- 定期実行のスケジュール機能（cron置き換え）
- リトライ、エラー通知（メール連携）
- Docker連携
- Etc.

シンプルdagファイル例

```
timezone: UTC

+task1:
  sh>: echo "execute task1"

+task2:
  sh>: echo "execute task2"
```

## Dockerとは？

- Docker社が開発するLinuxコンテナ(LXC)技術をベースとしたOSSのコンテナ型仮想化ソフトウェア



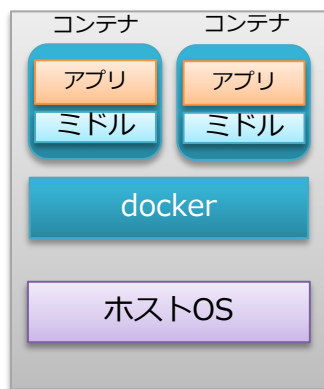
## 特徴

- 導入が容易（基本的にDockerソフトウェアのインストールのみ）
- ポータビリティ
- 軽量・高速
- 豊富なコンテナイメージ
- Dockerfileによるコード管理（Infrastructure As A Code）
- Etc.

通常構成  
(非仮想化)



コンテナ コンテナ



ハイパーバイザ型仮想化



ホスト型仮想化



Dockerは単一ホスト運用を想定したものであり、  
**大量サーバ運用**を想定した次のような**機能が不足** . . .

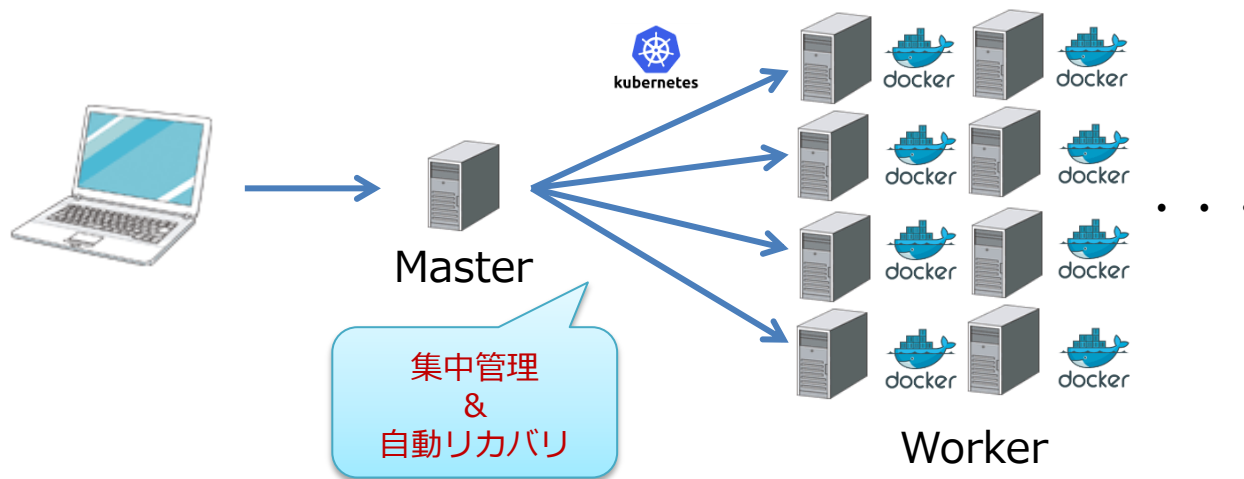
- コンテナを大量サーバへ**一括分散配備**する機能
- コンテナ実行を大量サーバへ**自動負荷分散**する機能
- コンテナが異常終了した場合の**自動リカバリ**機能  
などなど

## kubernetesとは？

- Google社が開発したOSSのコンテナ管理ソフトウェア（オーケストレーションツール）

## 特徴

- Linux Foundation傘下の組織である「Cloud Native Computing Foundation」（CNCF）に管理が委譲され、Apacheライセンスで配布。
- 1台のマスターサーバで大量サーバのコンテナ実行を集中管理
- コンテナを複数のサーバに自動負荷分散配備
- コンテナ稼働状況を死活監視し、停止したら自動でリカバリ
- Go言語
- Google GCPやRed Hat OpenShift等で採用。（コンテナ管理のデファクト化）



## 参考) Kubernetes と ECS



その他関連ツール



- 両者コンテナ管理に必要な以下の機能を有している  
コンテナ自動復旧、オートスケーリング、負荷分散(LB)、設定ファイルでの構成管理、プライベートリポジトリサポート、ロギング、etc.
- KubernetesはOSSであり、オンプレミス含め複数の環境での共通化が可能
- AWS ECSはAWS利用のみベースとしており、（当然ながら）AWSサービスとの親和性が高い

参考)

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

: [Amazon EC2 Container Service vs. Kubernetes](#)

[platform9.com](https://platform9.com)

: [Compare Kubernetes vs ECS \(Amazon EC2 Container Service\)](#)

JAWS-UGコンテナ支部#1

: [ECSと他のDocker管理サービスの比較](#) by 大瀧隆太氏（クラスメソッド株式会社）

## 参考) Kubernetesによるジョブフロー設定イメージ



ジョブフロー設定コマンド（タスク単位で実行）

**kubectrl create -f sampledata.yaml**

※削除 : delete / 更新 : apply

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: ***
spec:
  replicas: 4
  selector:
    app: ***
  template:
    metadata:
      labels:
        app: ***
    spec:
      volumes:
      - name: volumetmp
        hostPath:
          path: /volumetmp
      containers:
      - name: ***
        image: master:5001/***:***
        command: ["/tini"]
        args: ["-s", "python", "/*/**.py", "****"]
        volumeMounts:
        - mountPath: /***
          name: volumetmp
          : (以下省略)
```

sampledata.yaml

並列実行数

Private docker registryから  
ETL実行用ベースコンテナイメージを指定

内製ETL制御ツールパスと  
データ毎のコンフィグファイルを引数に設定

## ② ELT @ Redshift

- JOIN系処理はRedshift LOAD後にELT  
※SQLで容易に規定 & 大規模クラスターで高速処理可
- ユーザ影響を考慮し基本的には夜間実行



|             | vCPU | ECU | メモリ (GiB) | ストレージ    | I/O      |
|-------------|------|-----|-----------|----------|----------|
| ds2.8xlarge | 36   | 116 | 244       | 16TB HDD | 3.30GB/s |

× 125

# 参考) ETLモニタリングイメージ

The screenshot shows the Jenkins web interface. On the left, there is a sidebar with navigation links: '新規ジョブ作成', '開発者', 'ビルド履歴', 'Jenkinsの管理', '認証情報', 'ビルドキュー', 'ビルド待ち', and 'ビルド実行状態'. The main area displays a table of build jobs. The table has columns for 'S', 'W', '名前', '最新の成功ビルド', '最新の失敗ビルド', and 'ビルド所要時間'. Below the table, there are links for 'アイコン', 'ビルド', 'ビルド失敗', and 'ビルド成功'.

| S | W | 名前     | 最新の成功ビルド     | 最新の失敗ビルド      | ビルド所要時間 |
|---|---|--------|--------------|---------------|---------|
| ● | ● | ETLジョブ | 19時間前 - #10  | 1ヶ月11日前 - #22 | 3時間14分  |
| ● | ● | ETLジョブ | 1日0時間前 - #16 | 14日前 - #1     | 14秒     |
| ● | ● | ETLジョブ | 1ヶ月20日前 - #2 | —             | 0.14秒   |
| ● | ● | ETLジョブ | 36秒前 - #12   | —             | 1.2秒    |

データフローステータス管理 設定ファイル 参照...

Update: 2015/9/1 15:32:48

| カテゴリ   | データ    | 年月      | OK | ERR | OK | ERR | OK | ERR | OK | ERR | OK | ERR | OK | ERR |
|--------|--------|---------|----|-----|----|-----|----|-----|----|-----|----|-----|----|-----|
| ETLジョブ | ETLジョブ | +201508 |    |     |    |     |    |     |    |     |    |     |    |     |
| ETLジョブ | ETLジョブ | +201505 |    |     |    |     |    |     |    |     |    |     |    |     |
| ETLジョブ | ETLジョブ | +201503 |    |     |    |     |    |     |    |     |    |     |    |     |
| ETLジョブ | ETLジョブ | +201504 |    |     |    |     |    |     |    |     |    |     |    |     |
| ETLジョブ | ETLジョブ | +201505 |    |     |    |     |    |     |    |     |    |     |    |     |
| ETLジョブ | ETLジョブ | +201506 |    |     |    |     |    |     |    |     |    |     |    |     |
| ETLジョブ | ETLジョブ | +201507 |    |     |    |     |    |     |    |     |    |     |    |     |
| ETLジョブ | ETLジョブ | +201508 |    |     |    |     |    |     |    |     |    |     |    |     |
| ETLジョブ | ETLジョブ | +201509 |    |     |    |     |    |     |    |     |    |     |    |     |
| ETLジョブ | ETLジョブ | 201507  |    |     |    |     |    |     |    |     |    |     |    |     |
| ETLジョブ | ETLジョブ | 201508  |    |     |    |     |    |     |    |     |    |     |    |     |

※設定はサンプルです





### ③ 苦労ポイント + 課題



良い事例を  
教えてください

### ③ 苦労ポイント + 課題 リスト

#### 【CI/CD系】

- 終わりのなき新規データ追加
- 度重なる仕様変更
- 面倒なオンプレ環境の考慮
  - ※便利なManaged機能 < 共通化OSS
- Gitの検閲精度
- 他人のcodeイミフ問題
- 効率化ツールも増えると非効率？



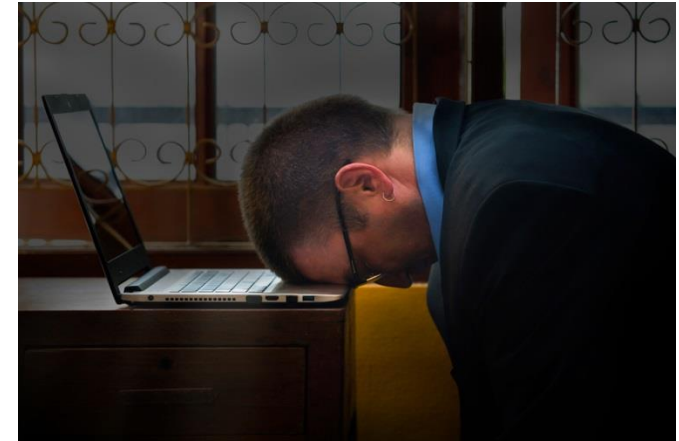
#### 対策実施中)

- 共通処理はプラグイン化
  - 定型作業はスクリプト化/自動化
    - 例) 項目定義書 → DDL変換、テスト/デプロイ
  - OSSでオンプレ/AWSを共通管理 (docker/k8s)
  - マニュアル化 (wiki/document)
  - 言語の統一化
  - コーディング規約
  - ツールの集約/クラウド移行
    - 例) Open PaaSの検討、Managedサービス移行
- etc.

### ③ 苦労ポイント + 課題 リスト

#### 【Ops系】

- DISK容量の圧迫問題
- 悲鳴ドリブン検知とエラーリカバリ
- ETLプログラム実行環境の共通化
- サーバ負荷の制御・管理
- ELT@Redshiftのユーザ影響



#### 対策実施中)

- アラート通知機能 ※ただしオオカミ少年化・・・
- データ保持期間の最適化（一定期間で自動削除）
- エラーハンドリング&自動リトライ ※提供元障害も多い…
- コンテナ（docker/k8s）の導入
- 運用系クエリの深夜帯実行

etc.

### 【データマネジメント系】

- メタデータ管理  
(データオーナー、DDL、SLA、用途、データ提供仕様、利用規約、etc.)
- データ仕様標準化 (特にID)
- データの利用価値可視化
- データカタログの賢いユーザ周知




#### 今後の課題・・・

- データ価値に応じたコストマネジメントの実現へ  
→ 作業優先度(稼働)、データ保持期間、ストレージ種別
- 投入データの利用率UP (=データドリブン加速) のための工夫 (使えるデータをユーザに見つけやすく)
- 使いやすいデータ形式により、無駄な問合せや、無駄なJOINの低減

- オンプレミス+AWSベースのビッグデータ統合分析システムについて紹介
- ETL部に関して、利用ツール・アーキテクチャについて紹介
- ETL開発/運用に関する課題について共有



良い事例を  
教えてください



ご清聴ありがとうございました