データエンジニアリング研修
Day 2

技術部データ基盤チーム 小松ももか

2022.08.01

GMOペパボ

（おさらい）1日目のゴール

目標

- 自サービスで ◯◯ を分析するために Bigfoot を使う方法が分かる

演習

- データを目的に沿って変換する
- データをData Studioで可視化する

💡 Bigfoot にデータを入れてしまえば、あとは柔軟に変換できそう!
では、データを入れる方法は?

2日目のゴール

目標

- データのELT(Extract, Load, Transform)する方法を学ぶ

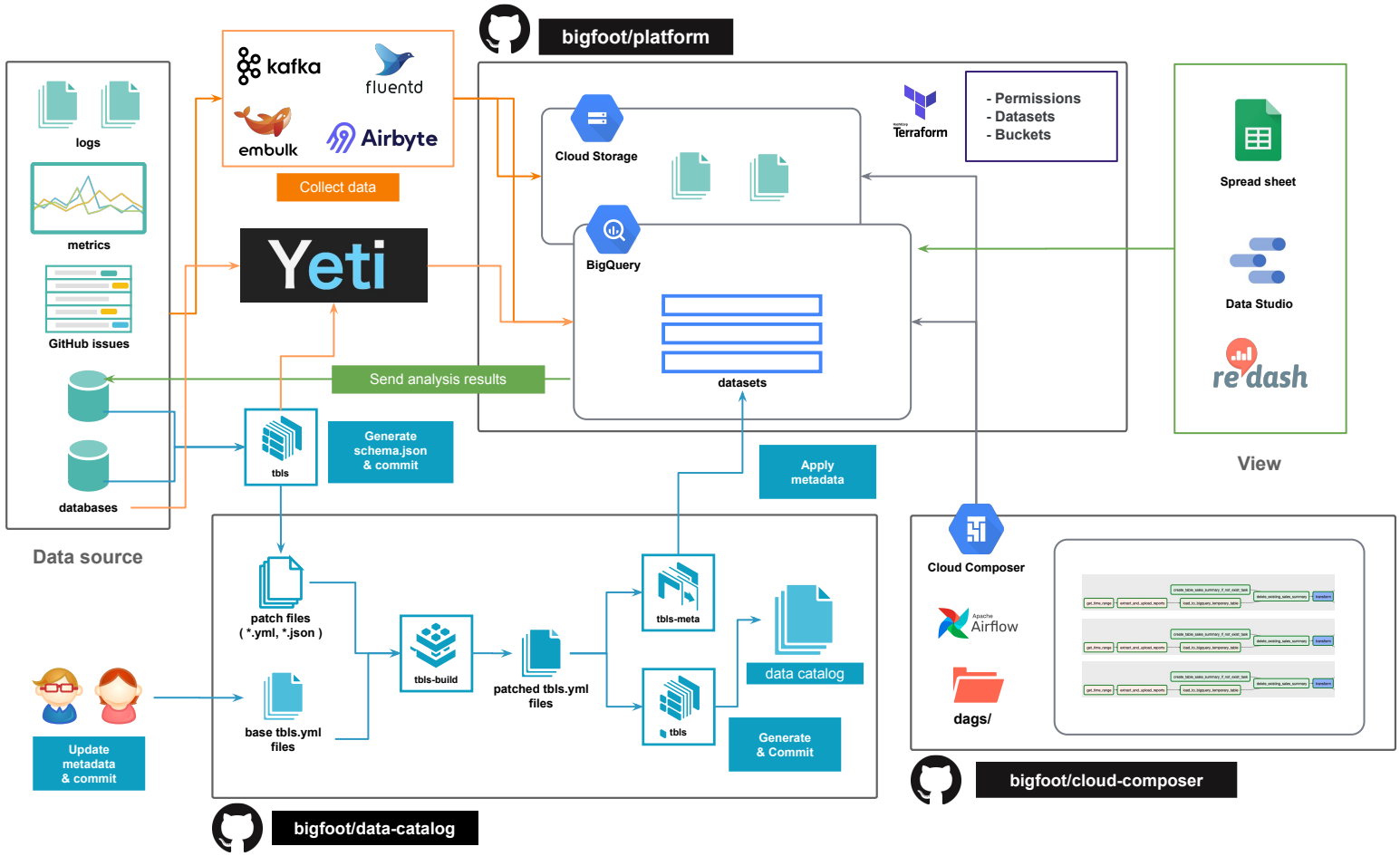
演習

- BigQueryにデータをロードする

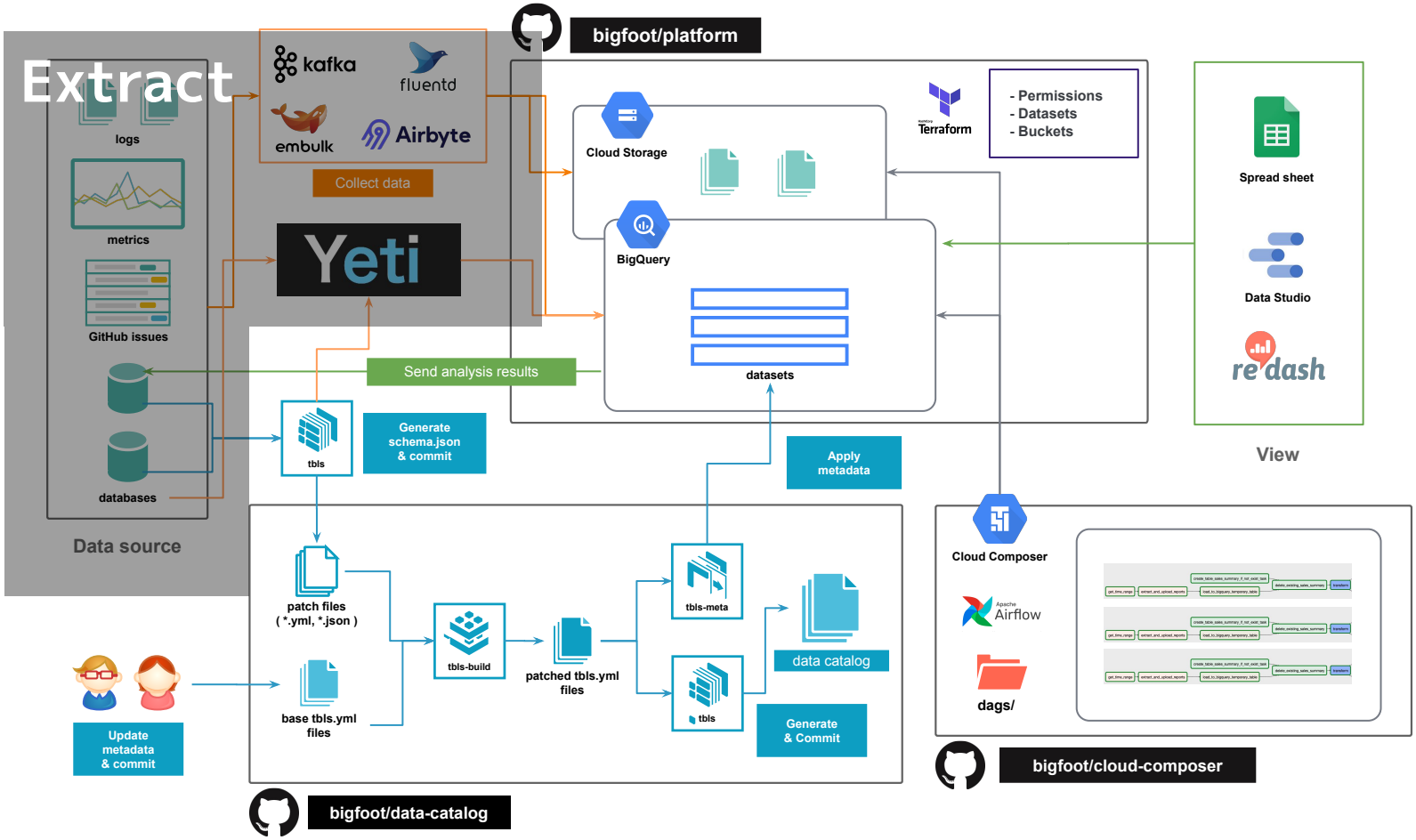
ELTってなんですか？

Extract 抽出
Load 取り込み
Transform 変換

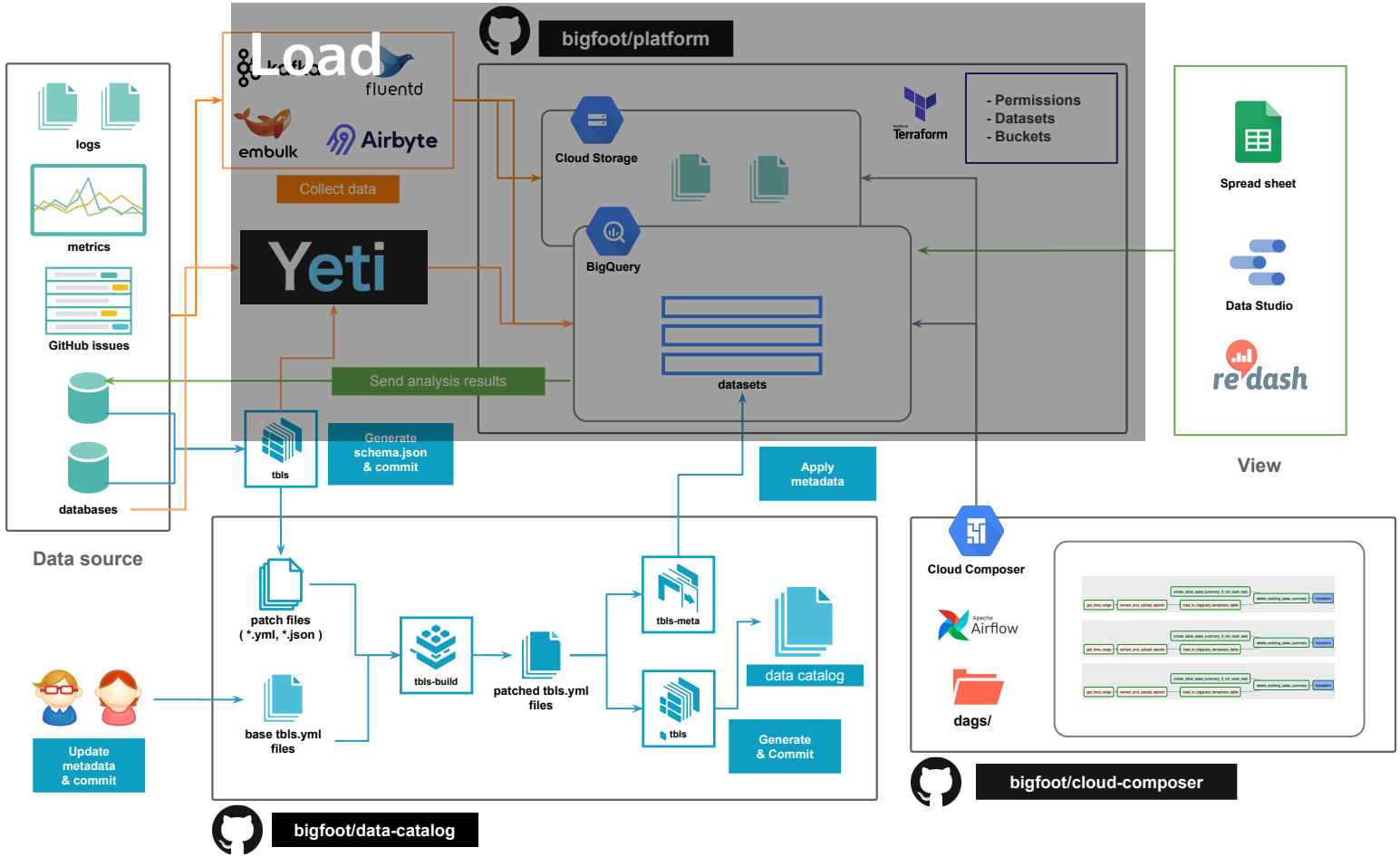
Bigfoot 概略図


bigfoot/platform
kafka
fluentd
embulk
Airbyte
Collect data
logs
metrics
GitHub issues
databases
Data source
Yeti
Cloud Storage
BigQuery
datasets
Terraform
- Permissions
- Datasets
- Buckets
Send analysis results
tbls
Generate schema.json & commit
Apply metadata
Spread sheet
Data Studio
re'dash
View
patch files (*.yml, *.json)
base tbls.yml files
Update metadata & commit
tbls-build
patched tbls.yml files
tbls-meta
tbls
data catalog
Generate & Commit
bigfoot/data-catalog
Cloud Composer
Apache Airflow
dags/
bigfoot/cloud-composer

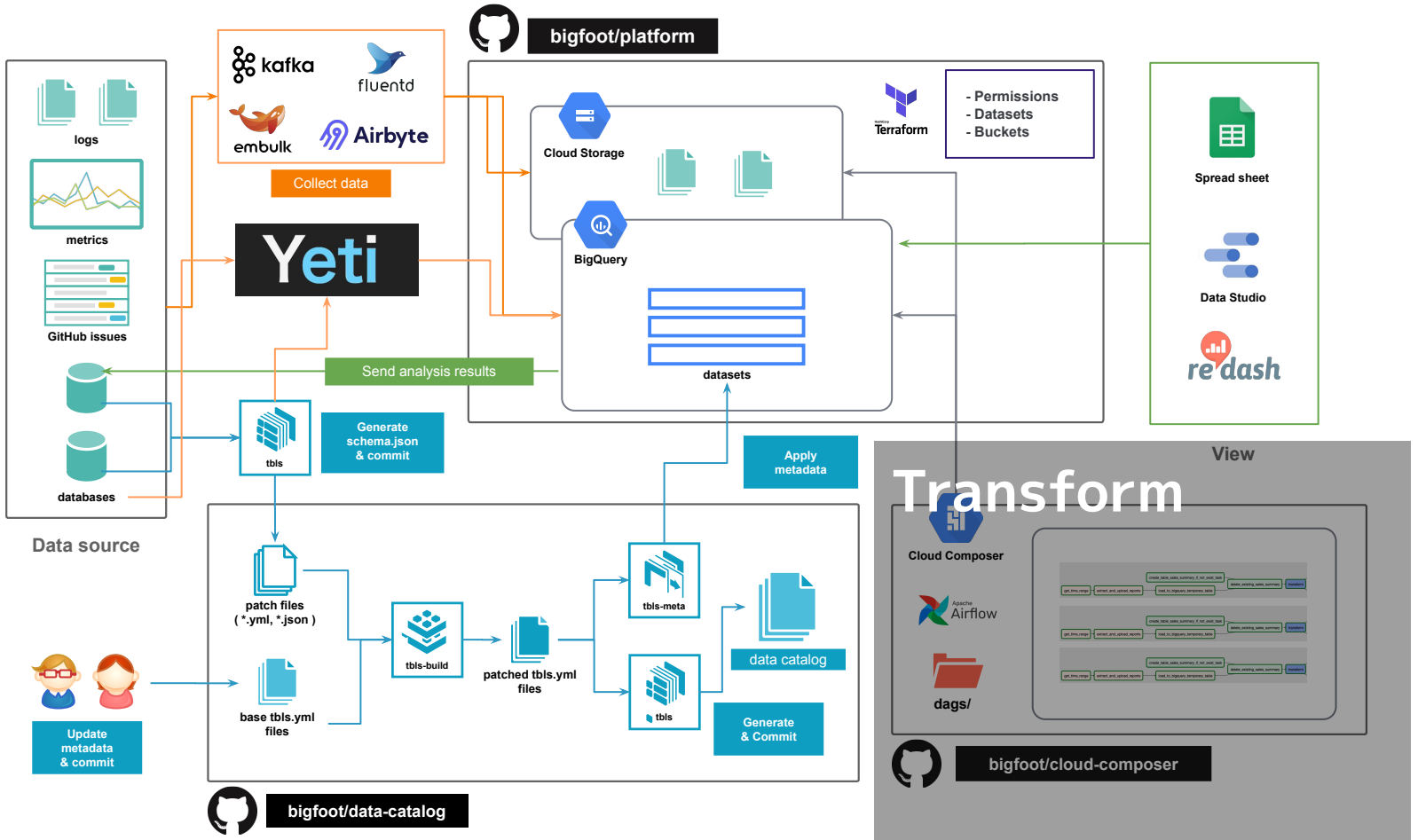
Bigfoot 概略図


bigfoot/platform
Extract
kafka
fluentd
embulk
Airbyte
Collect data
Yeti
logs
metrics
GitHub issues
databases
Data source
Cloud Storage
BigQuery
datasets
Terraform
- Permissions
- Datasets
- Buckets
Spread sheet
Data Studio
re:dash
View
Send analysis results
tbls
Generate schema.json & commit
Apply metadata
patch files (*.yml, *.json)
base tbls.yml files
tbls-build
patched tbls.yml files
tbls-meta
tbls
data catalog
Generate & Commit
Update metadata & commit
bigfoot/data-catalog
Cloud Composer
Apache Airflow
dags/
bigfoot/cloud-composer

Bigfoot 概略図


Load
fluentd
embulk
Airbyte
Collect data
bigfoot/platform
Yeti
Cloud Storage
BigQuery
datasets
Terraform
- Permissions
- Datasets
- Buckets
Send analysis results
logs
metrics
GitHub issues
databases
Data source
tbls
Generate schema.json & commit
Apply metadata
Spread sheet
Data Studio
re:dash
View
Cloud Composer
Apache Airflow
dags/
bigfoot/cloud-composer
patch files (*.yml, *.json)
base tbls.yml files
Update metadata & commit
tbls-build
patched tbls.yml files
tbls-meta
tbls
data catalog
Generate & Commit
bigfoot/data-catalog

Bigfoot 概略図


bigfoot/platform
kafka
fluentd
embulk
Airbyte
Collect data
logs
metrics
GitHub issues
databases
Data source
Yeti
Cloud Storage
BigQuery
datasets
Terraform
- Permissions
- Datasets
- Buckets
Spread sheet
Data Studio
re'dash
Send analysis results
tbls
Generate schema.json & commit
Apply metadata
View
Transform
Cloud Composer
Apache Airflow
dags/
bigfoot/cloud-composer
patch files (*.yml, *.json)
base tbls.yml files
Update metadata & commit
tbls-build
patched tbls.yml files
tbls-meta
tbls
data catalog
Generate & Commit
bigfoot/data-catalog

抽出: 何を集めるか

- サービス(事業)情報
- ユーザー属性情報
- ユーザー行動ログ
- カスタマーサポート情報
- (サーバー|アプリケーション)ログ
- GHESデータベース

抽出: どこから集めるか

事業・サービスのDB

- オンプレミス
- プライベートクラウド
- パブリッククラウド

SaaS

- Google Analytics
- Zendesk
- Salesforce

抽出: どうやって集めるか

対象データの種類・量によって収集方法は異なる

- バッチ処理
  - データベースから一括エクスポート

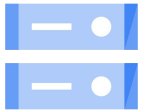
- ストリーミング処理
  - ログファイルへの書き込みイベントをもとにデータ抽出

- 外部サービスの API にリクエスト

取り込み: どこに貯めるか

- ストレージ
  - ファイルストレージ
  - オブジェクトストレージ

- データベース
  - RDB
  - NoSQL
  - データウェアハウス



Google
Cloud Storage

Amazon S3

Google
BigQuery

変換: データの変換が必要な例

- 生成元固有のデータ表現がある
- データの粒度/構造が利用者のほしい形になっていない
- データ間を結合するためのキー情報が足りない
- 結合元データの管理組織が異なる
  - データ A を管理している事業部 X
  - データ B を管理している事業部 Y
  - 事業部 X, Y が共通して参照できるデータ C をつくる
- ASCII 0（NUL; ヌル文字; \0）が混入している

ELTの心構え

目的を実現するために、どのような手段をつかうと...

- 求められた更新頻度に近いか？
- 利用者/運用者が使いやすいか？
- 上記の要素が経年変化しないか？
  - データの種類・量の増加
  - Input/Output システムの種類数の増加
  - 利用人数の増加
- コストが少なく済むか？

更新方法と周期変更
に対する柔軟性

運用負担を減らす
冪等性のある処理

スケーラビリティ

乗り換えのしやすさ
E・L・Tを疎結合に

Google Cloud Composer

- Managed Apache Airflow on Google Kubernetes Engine(GKE)
  - Apache Airflow: オープンソースのワークフローエンジン
  - ELT のオーケストレーター。”ものすごく高機能な Cron”
  - Python製。ワークフローの定義も Pythonで実装
  - スケジューラーやワーカーが GKE で動くので、
    スケールアップ/ダウン/イン/アウトを柔軟に行うことができる
  - 周期的で冪等な処理が得意

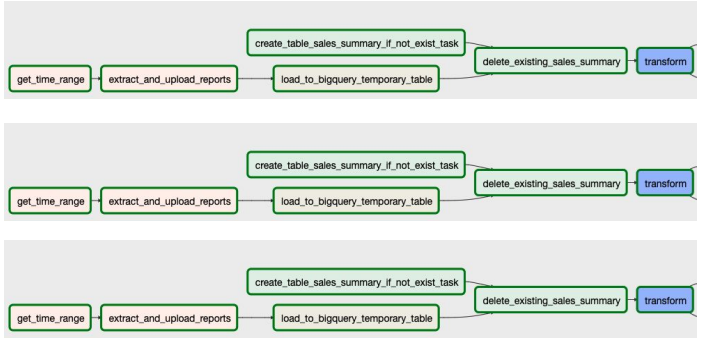
Google Cloud Composer

GMOペパボ

Cloud Composer

Apache Airflow

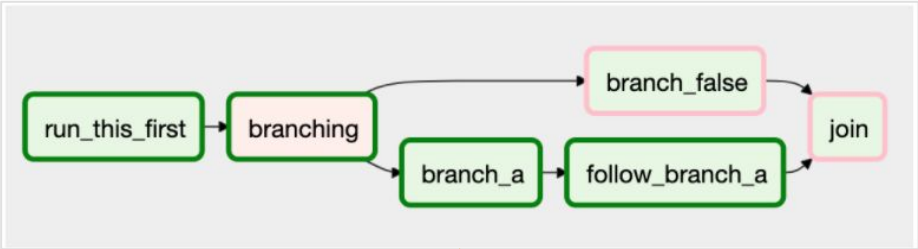
dags/



bigfoot/cloud-composer

DAG

Directed acyclic graph(有向非循環グラフ)
Pythonコードで、ワークフローを
グラフとして定義する




```python
#dags/branch_without_trigger.py
import datetime as dt

from airflow.models import DAG
from airflow.operators.dummy_operator import DummyOperator
from airflow.operators.python_operator import BranchPythonOperator

dag = DAG(
    dag_id='branch_without_trigger',
    schedule_interval='@once',
    start_date=dt.datetime(2019, 2, 28)
)

run_this_first = DummyOperator(task_id='run_this_first', dag=dag)
branching = BranchPythonOperator(
    task_id='branching', dag=dag,
    python_callable=lambda: 'branch_a'
)

branch_a = DummyOperator(task_id='branch_a', dag=dag)
follow_branch_a = DummyOperator(task_id='follow_branch_a', dag=dag)

branch_false = DummyOperator(task_id='branch_false', dag=dag)

join = DummyOperator(task_id='join', dag=dag)

run_this_first >> branching
branching >> branch_a >> follow_branch_a >> join
branching >> branch_false >> join
```

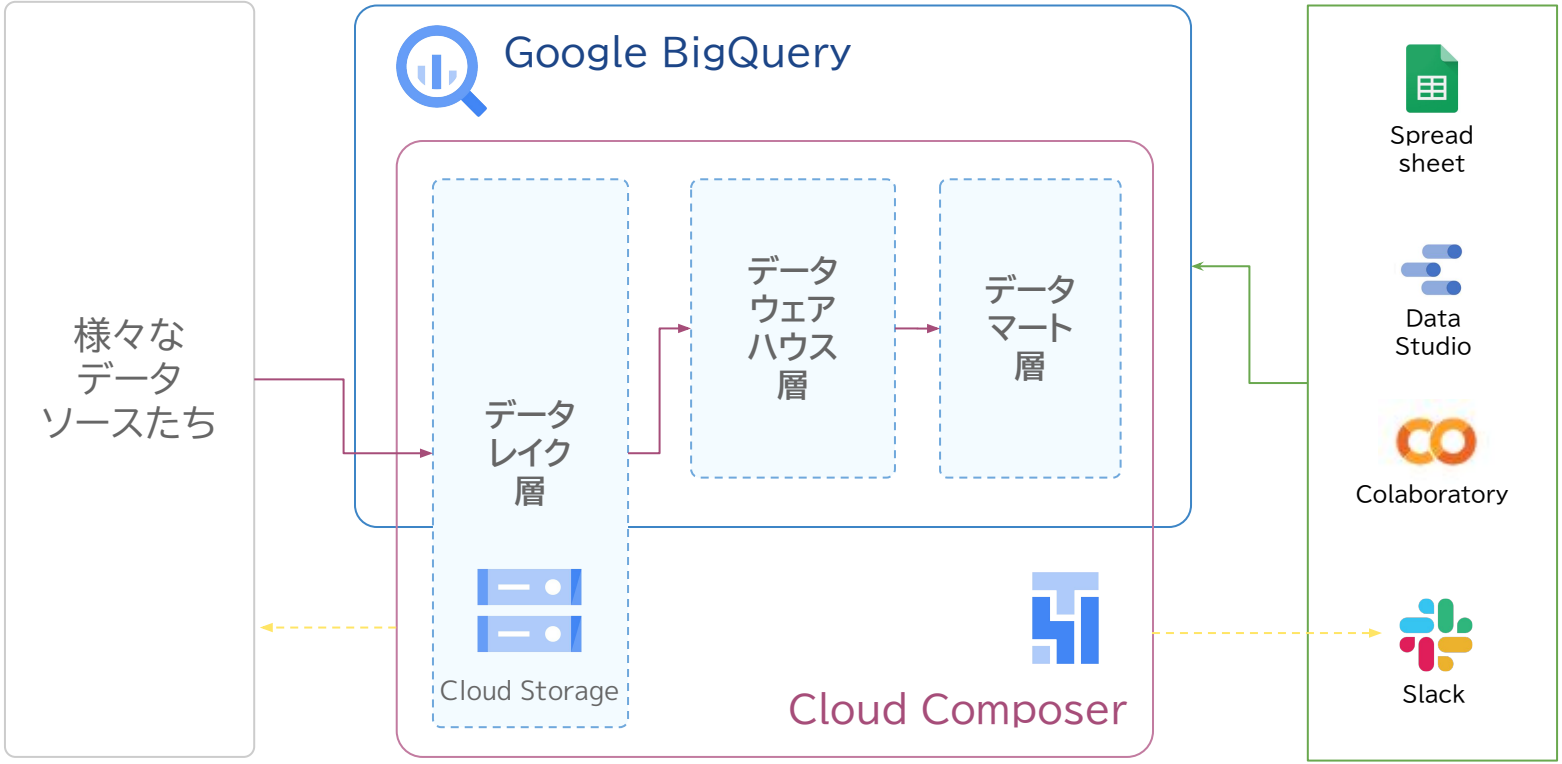
Operator

ワークフロータスクのテンプレート

たくさん種類があるので一例

- BigQueryExecuteQueryOperator
  - BigQueryへSELECTを実行して、その結果を別テーブルに保存する
- GoogleCloudStorageToBigQueryOperator
  - GCSに保存されているファイルのデータをBigQueryへ登録する
- AwsBatchOperator
  - AWS Batch のジョブを実行する

ELTのステージによってデータを分類



GMOペパボ

データレイク層

- 元データをコピーして、一つのシステムに集約したもの
- データソース（=水源）から流れてきたデータを蓄えるレイク（湖）

目的

- データを一箇所に集約する
  - Single Source of Truth（信頼できる唯一の情報源）

GMOペパボ

データウェアハウス層

- 共通指標となるデータ置き場
- 大量のデータを意味のある形に管理することからウェアハウス（倉庫）と呼ぶ

目的

- ドメイン依存な表現をニュートラルにする
- 加工しやすい汎用的な状態にする
    - 欠損埋め、重複削除、名寄せ

GMOペパボ

データマート層

- 「特定の利用者」「特定の用途」向けに加工・整理したデータ置き場
- すぐに使える完成品を取り揃えていることからマート（市場）と呼ぶ
- データマート層は用途（ユースケース）と一対一の関係にある

目的

- 集計の効率化
  - 「データを利用したい」から「実際にデータを利用する」までの時間の削減
- 集計ロジックの管理
  - コード管理すれば処理が明文化され、再利用も可能になる

ETLの事例: Yeti

https://speakerdeck.com/tosh2230/yeti-yet-another-extract-transfer-infrastructure

GMOペパボ

Google BigQuery: インターフェース

- Google Data Studio
- Google Sheets
- BigQuery Web Console
- Google Apps Script
- Google Colaboratory
- bq command-line tool
- API Client Libraries

  C#, Go, Java, Node.js, PHP,

  Python, Ruby
- REST API

GMOペパボ

Google BigQuery: インターフェース

- Google Data Studio
- BigQuery Web Console
- Google Sheets
- bq command-line tool

1日目の演習課題で扱います

2日目の演習課題で扱います

演習

- （2-1）スプレッドシートのデータをBigQueryに取り込んでみてください
  - GCSのバケットにCSVを用意しています
  - CSVをスプレッドシートにインポートし、スプレッドシートからBigQueryに取り込んでみてください
    - 一行目は除外してください

演習

- （2-2）**2GBのCSV**をBigQueryに取り込んでみてください
  - データサイズが大きいので、ロードには少し工夫が必要です
    - Cloud Shell を使ってみましょう
    - データをロードするには？ bqコマンドラインツール
    - Cloud Storage をCUIから操作するには？ gsutil ツール

時間が余ったら

- （おかわり課題）Embulkを使ったデータロードを体験してみましょう
  - Embulkをインストールする方法を調べましょう
    - Quick Start で使い方を把握しましょう
  - 認証のための Service Account のキーを用意する方法を調べましょう
  - CSVをEmbulkを使ってロードしてみましょう

参考文献

DAMA International (2018)『データマネジメント知識体系ガイド 第二版』日経BP社

ゆずたそ, はせりょ(2020)『データマネジメントが30分でわかる本』

ゆずたそ, 渡部徹太郎, 伊藤徹郎『実践的データ基盤への処方箋』技術評論社

下田 倫大, 寶野 雄太, 饗庭 秀一郎, 吉田 啓二 (2021)『Google Cloud ではじめる実践データエンジニアリング入門』技術評論社