GMOペパボ

機械学習研修
Day 2

author = ‘Toshifumi Tsutsumi’ AND
presented_at = ‘2021-07-16’

機械学習研修: 前半2日間のゴール

- 大規模データを ETL (Extract, Transform, Load)する方法を知る

- データウェアハウスの使い方やETLの基本的な考え方に慣れて、自分が(将来)利用するデータを抽出・加工できるようになる

NOTE:
演習では、一般社団法人 データサイエンティスト協会が提供する
“データサイエンス100本ノック（構造化データ加工編）” のデータを BigQuery にロードしています
https://github.com/The-Japan-DataScientist-Society/100knocks-preprocess

機械学習研修: 前半2日間の予定

2021/7/15(木) 15:00 - 18:00

1. データエンジニアリングってなんですか？
2. Bigfoot 最速入門
3. Google BigQuery 入門
4. Google DataStudio 入門

機械学習研修: 前半2日間の予定

2021/7/16(金) 15:00 - 17:00

5. ETLってなんですか？
6. ETL Ultraquick Tutorial
7. データエンジニアリングってなんですか？

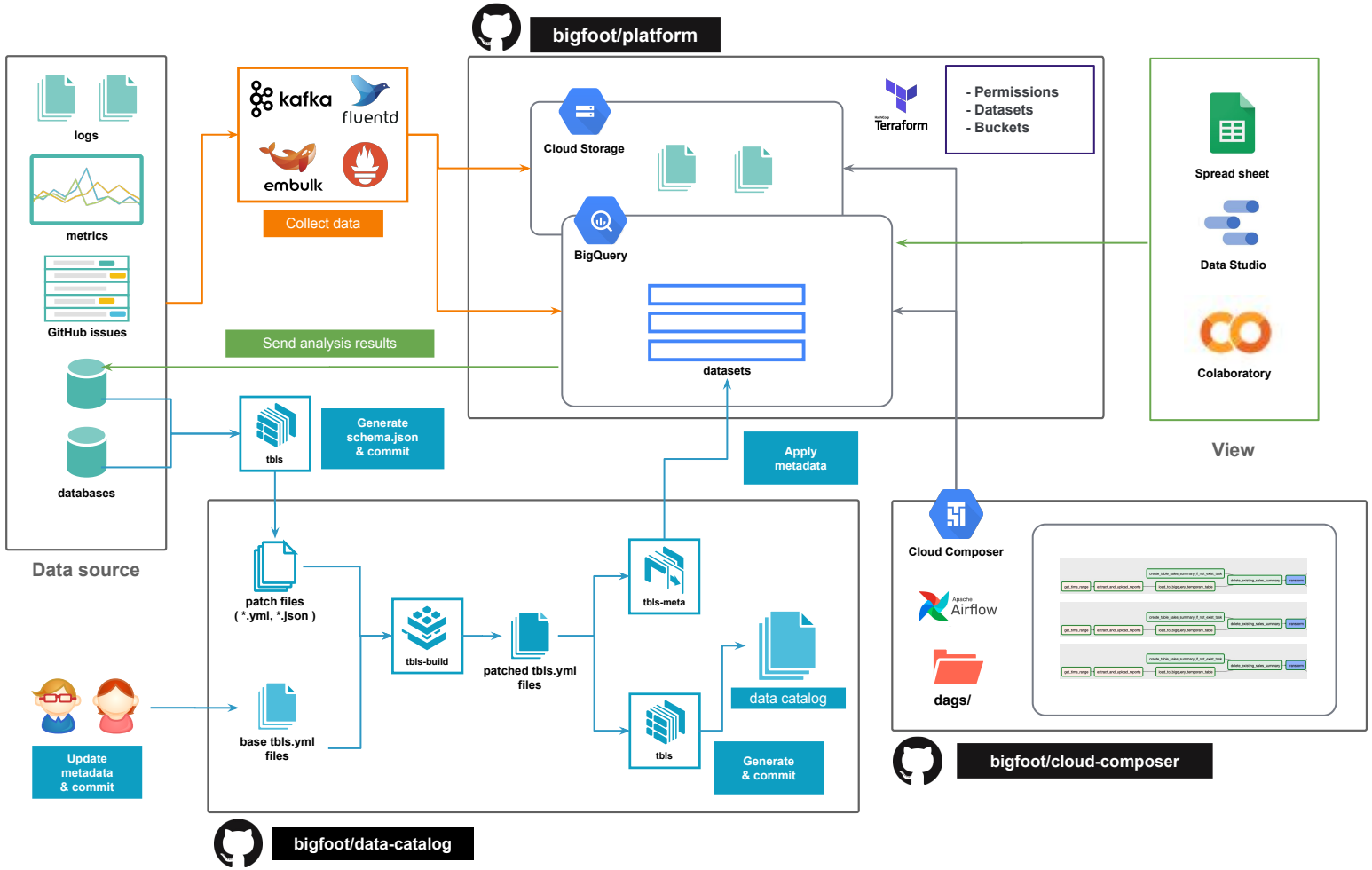
Section 5

ETLってなんですか？

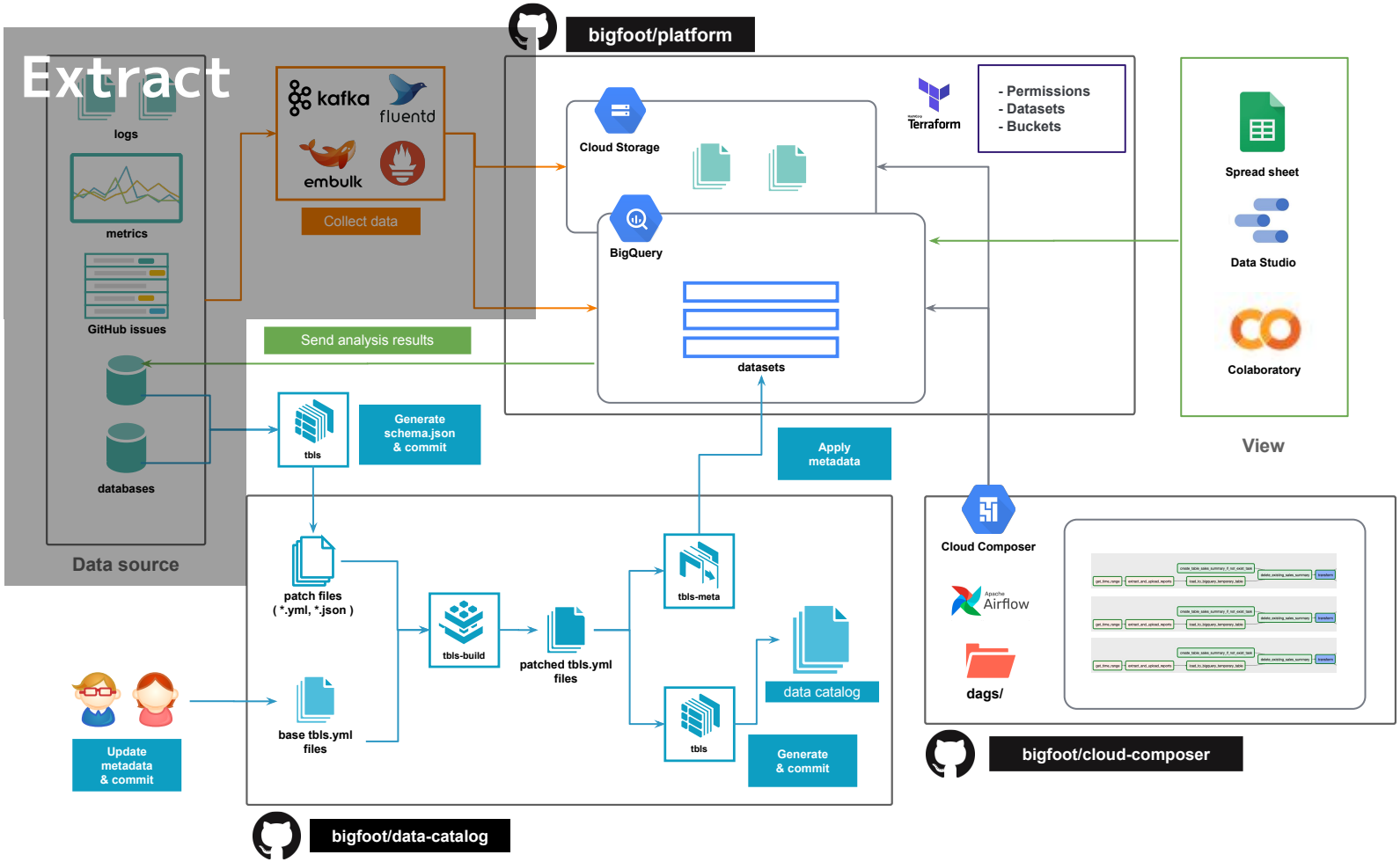
Extract
Transform
Load

Extract
Load
Transform

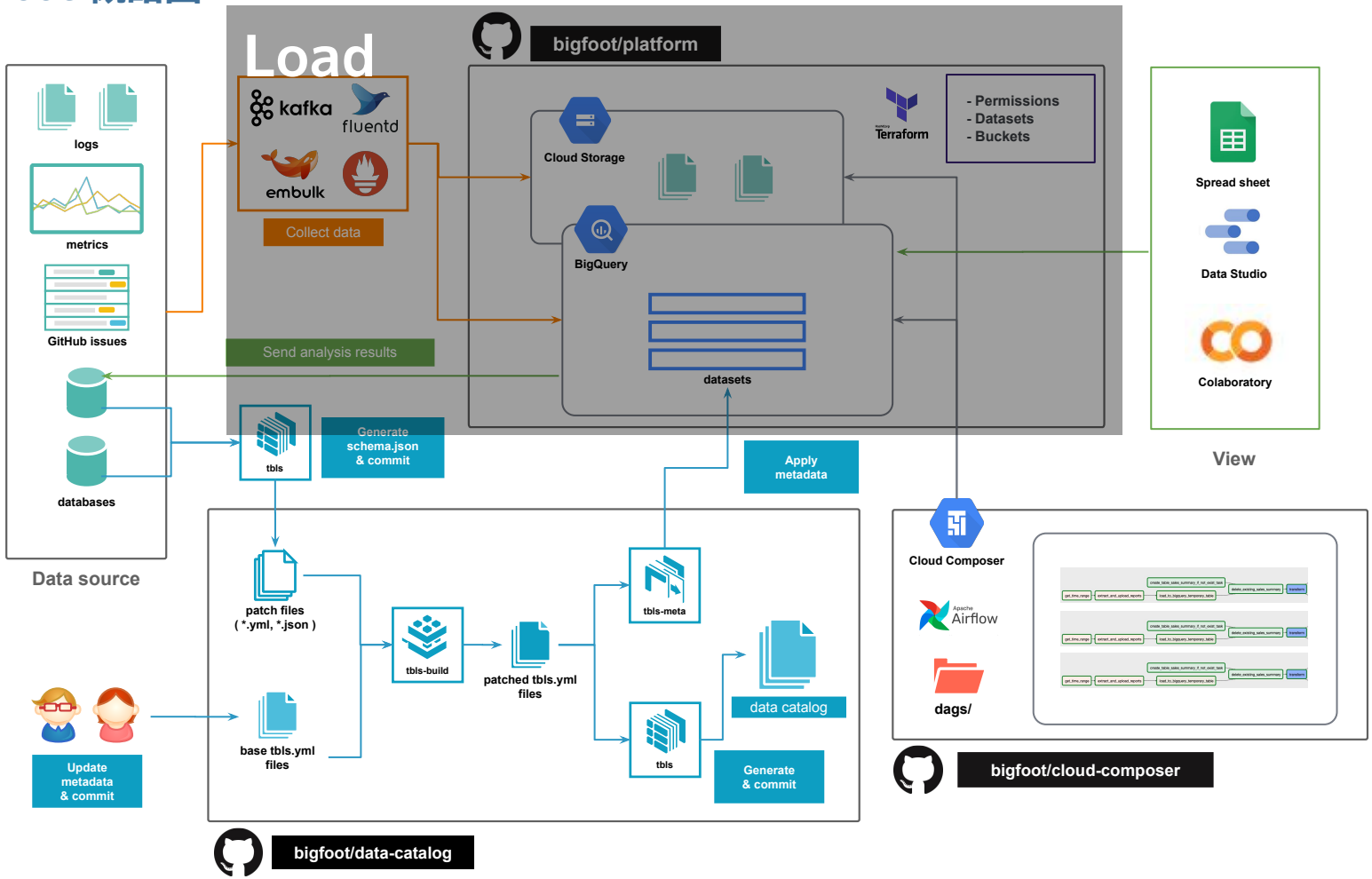
Bigfoot 概略図


bigfoot/platform
logs
metrics
GitHub issues
databases
Data source
kafka
fluentd
embulk
Collect data
Cloud Storage
BigQuery
datasets
Terraform
- Permissions
- Datasets
- Buckets
Spread sheet
Data Studio
Colaboratory
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

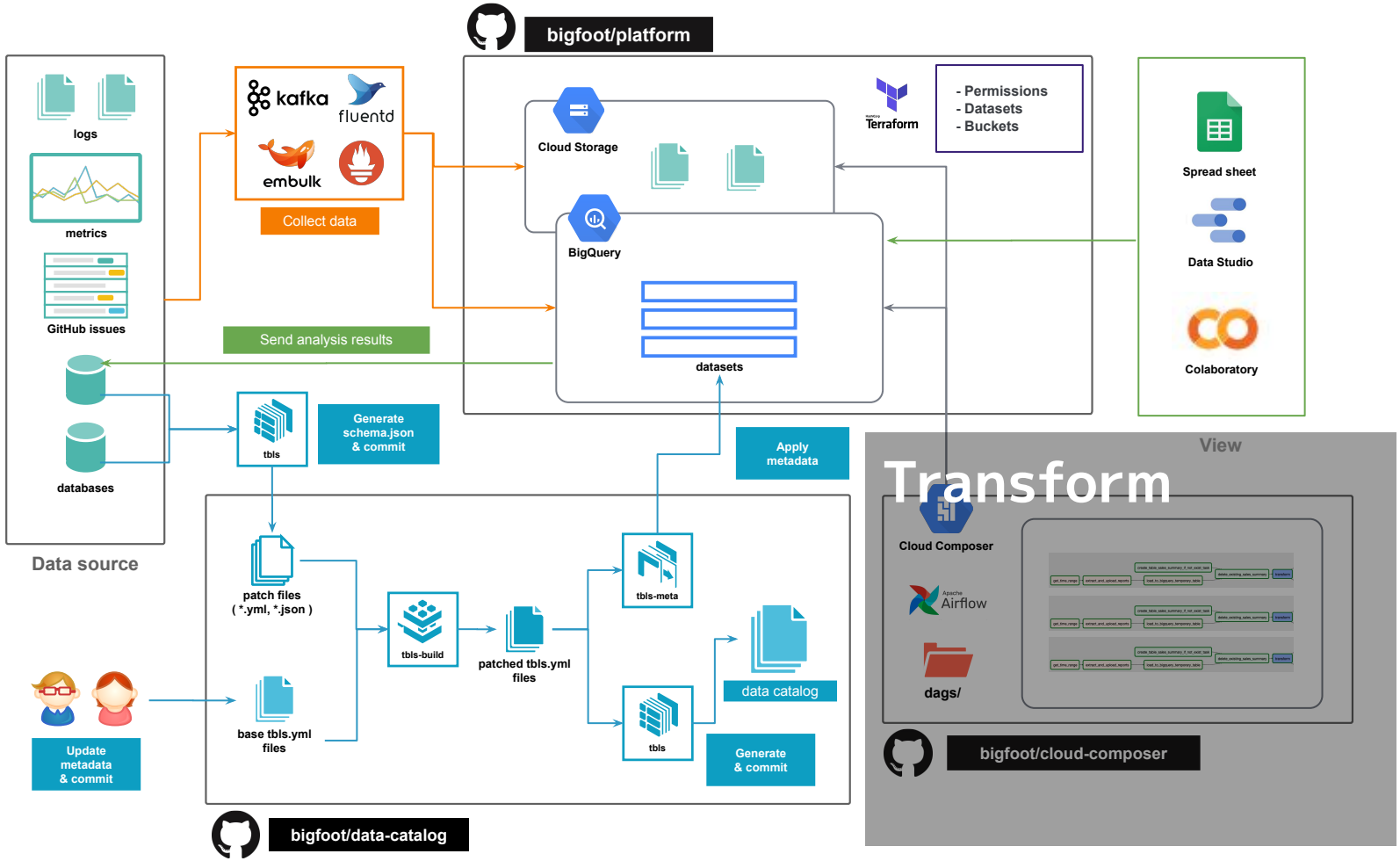
Bigfoot 概略図


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

Bigfoot 概略図


Load
bigfoot/platform
kafka
fluentd
embulk
Collect data
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
Colaboratory
View
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


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

収集: 何を集めるか

GMOペパボ

対象例

- 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
- などなど

目的(=売上や利益を生み出す施策)にあわせて
何を集めるかを決める
↓
その場でできることなら移動しない

収集: 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

抽出処理をどこで動かすか？
どういったルートで転送するか？
ネットワーク帯域は十分か？

- SaaS
  - Google Analytics
  - Zendesk
  - Salesforce

どういうAPIがあるか？
更新頻度は？
データ構造は？

- オープンデータ
  - 公的機関
  - SNS

収集: どうやって集めるか

GMOペパボ

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 

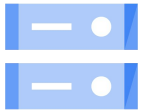
- ストリーミング処理
  - ログファイルへの書き込みイベントをもとにデータ抽出 

- 外部サービスの API にリクエスト

GMOペパボ

データの保存場所

- 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



Google
Cloud Storage

Amazon S3

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

Google
BigQuery

GMOペパボ

データの保存期間

- 溜めるのは結構かんたん...でも
  - お金がかかる！
  - 管理しないと、データを特定するのが難しくなっていく

- 必要なものを必要な分だけ
  - 目的に照らしてデータのライフサイクルを定める
  - Spark Joy

GMOペパボ

データ加工が必要な例

- 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
  - CSVファイルに混入していると、BigQuery へのインポートでエラーになる
  - JSONファイルに変更したり、trコマンドで地道に消したり...

ETL での心構え

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
  - Input/Output システムの種類数
  - 利用人数
- そしてコストが少なく済むか...?

ETL での心構え

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?

更新方法と周期変更
に対する柔軟性

- 利用者/運用者が使いやすいか?

運用負担を減らす
冪等性のある処理

- 上記の要素が経年変化しないか?
  - データの種類・量
  - Input/Output システムの種類数
  - 利用人数

スケーラビリティ

- そしてコストが少なく済むか...?

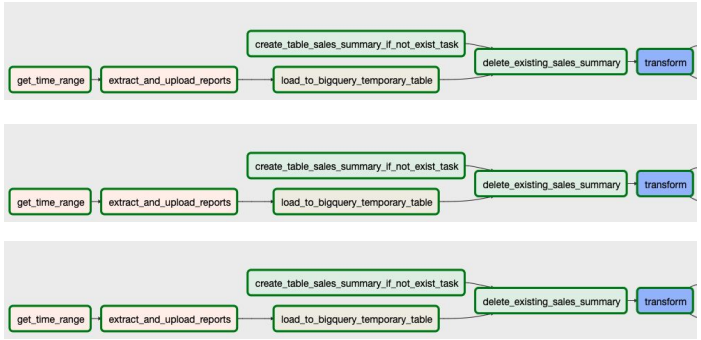
乗り換えのしやすさ
E・L・Tを疎結合に

Google Cloud Composer

Cloud Composer

Apache Airflow

dags/

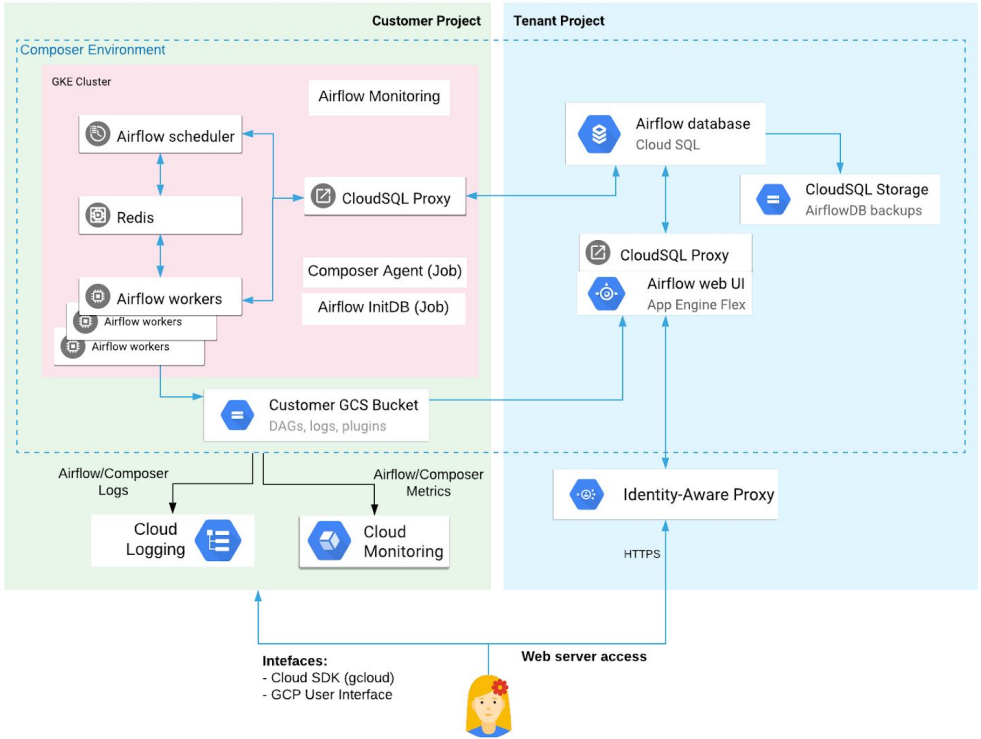


bigfoot/cloud-composer

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
- ETL のオーケストレーター。”ものすごく高機能な Cron”
  Bigfootでは、場合によって Extract/Transform/Load も兼任
- 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

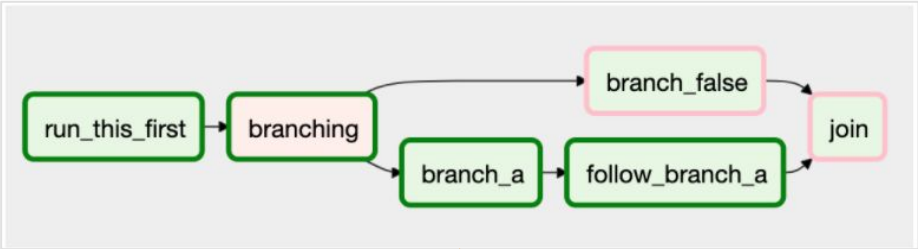


https://cloud.google.com/composer/docs/concepts/architecture

GMOペパボ

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
```

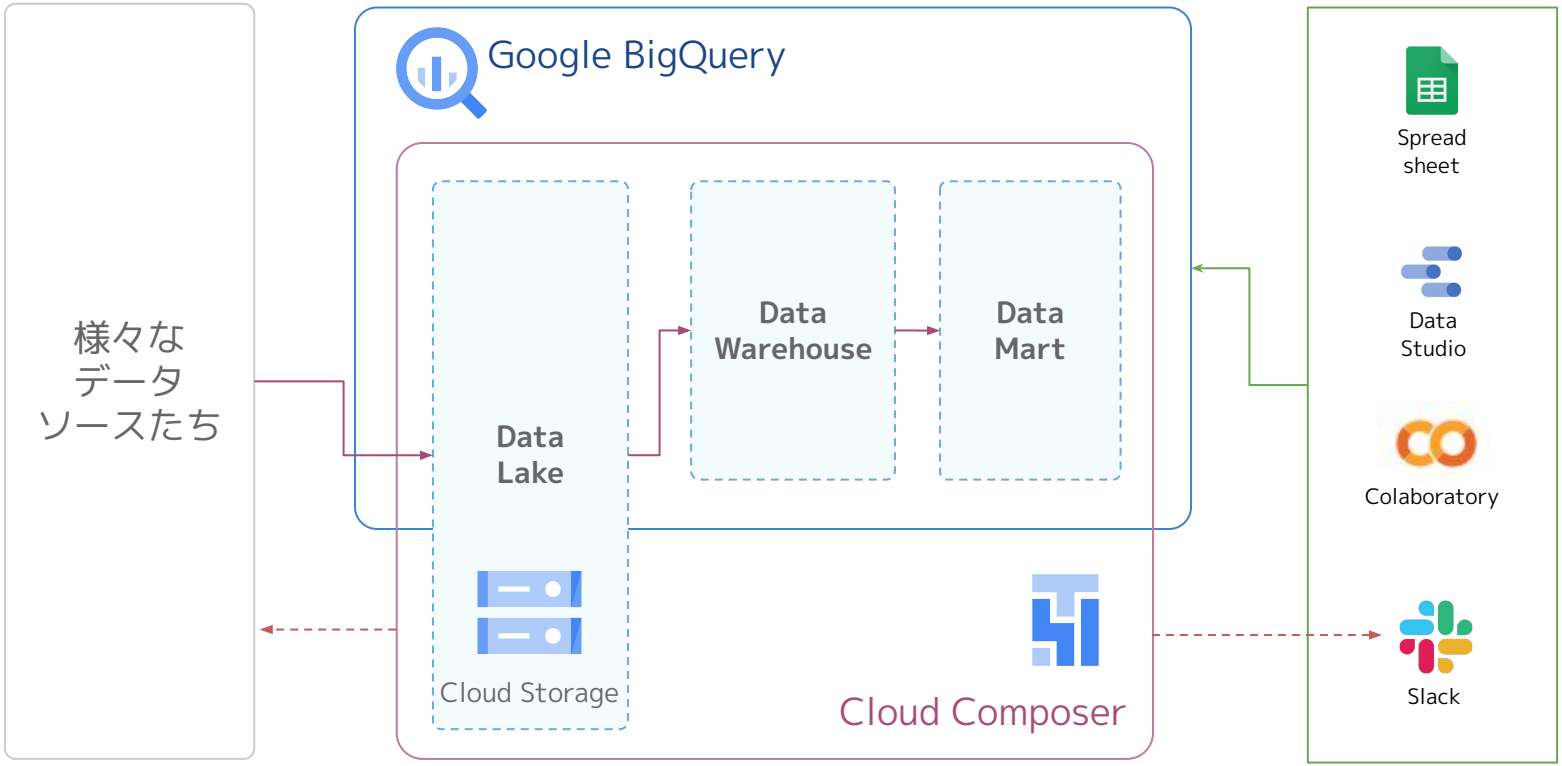
https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html#trigger-rules

Operator

ワークフロータスクのテンプレート。便利。
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

ETLのステージによってデータを分類



DataLake Layer

生データ(Raw data、未加工データ) を保存する層

- 構造化データ: CSV, TSV
- 半構造化データ: JSON
- 非構造化データ: 画像, 音声, 動画

<u>データレイク層の目的</u>

- データの真正性を担保
  - “結局、正しいデータはどれだっけ？” がなくなる
  - Single Source of Truth

- Transform のやり直し・組み換えが容易になる

DataWarehouse Layer

構造化データ、半構造化データを格納する層
※ データベースの文脈の DataWarehouse とは異なる

<u>データウェアハウス層の目的</u>

- 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

DataMart Layer

分析やレポーティングなど、特定の目的に最適化した形でデータを保持する。

<u>データマート層の目的</u>

- 非機能要件での制約事項を回避するため
  - クエリ処理時間を短縮
  - データソースへのアクセス制限

- ビジネスロジックが局所的に隠蔽されるのを避ける
  - データマート層がなくても、BI ツール や スプレッドシートにクエリを埋め込めば、機能的には同じことができる。が、それはいずれ「○○さんだけが知ってる例のアレ」になる
  - データマート層をつくることで、コード管理の対象となり、処理過程がパブリックになる

では、ETL やってみましょう

Section 6

ETL Ultraquick Tutorial

ETL演習(1)

Python の便利ライブラリを試してみましょう

- Numpy
  - 数値計算モジュール
  - 型付き多次元配列での行列計算ができる
  - Numpy Ultraquick Tutorial (Google Colaboratory)

- Pandas
  - データ解析支援モジュール
  - 表形式のデータ構造(DataFrame)を提供する
  - Pandas Ultraquick Tutorial (Google Colaboratory)

ETL演習(2)

突然ですが、trainingデータセットの売上(training.receipt.amount)は
正確ではない、という衝撃の事実が判明しました。
Day1で見た数字は誤りだったのです…

実は販売時に割引しているケースがあったのですが、それをすっかり忘れてしまい
割引前の金額でそのまま売上額として計上していたようです。

実際に割引された金額は、なんとか明細単位でCSVファイルにすることができました。
receiptテーブルにある売上金額を、あるべき形に加工しなければなりません。

Google Colaboratory で加工してみましょう！

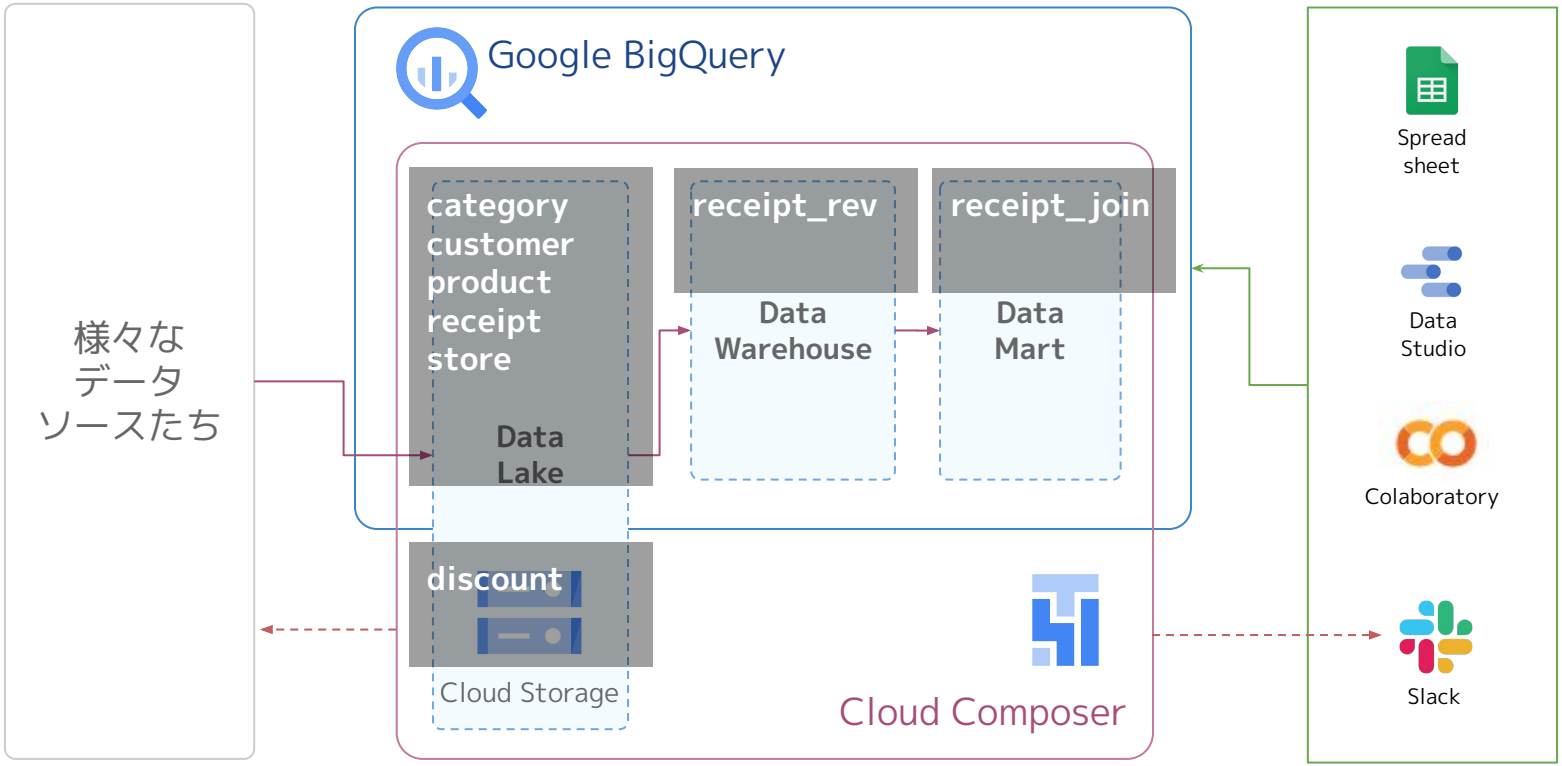
NOTE:
演習用のノートブックと追加データを使った演習です

Section 7

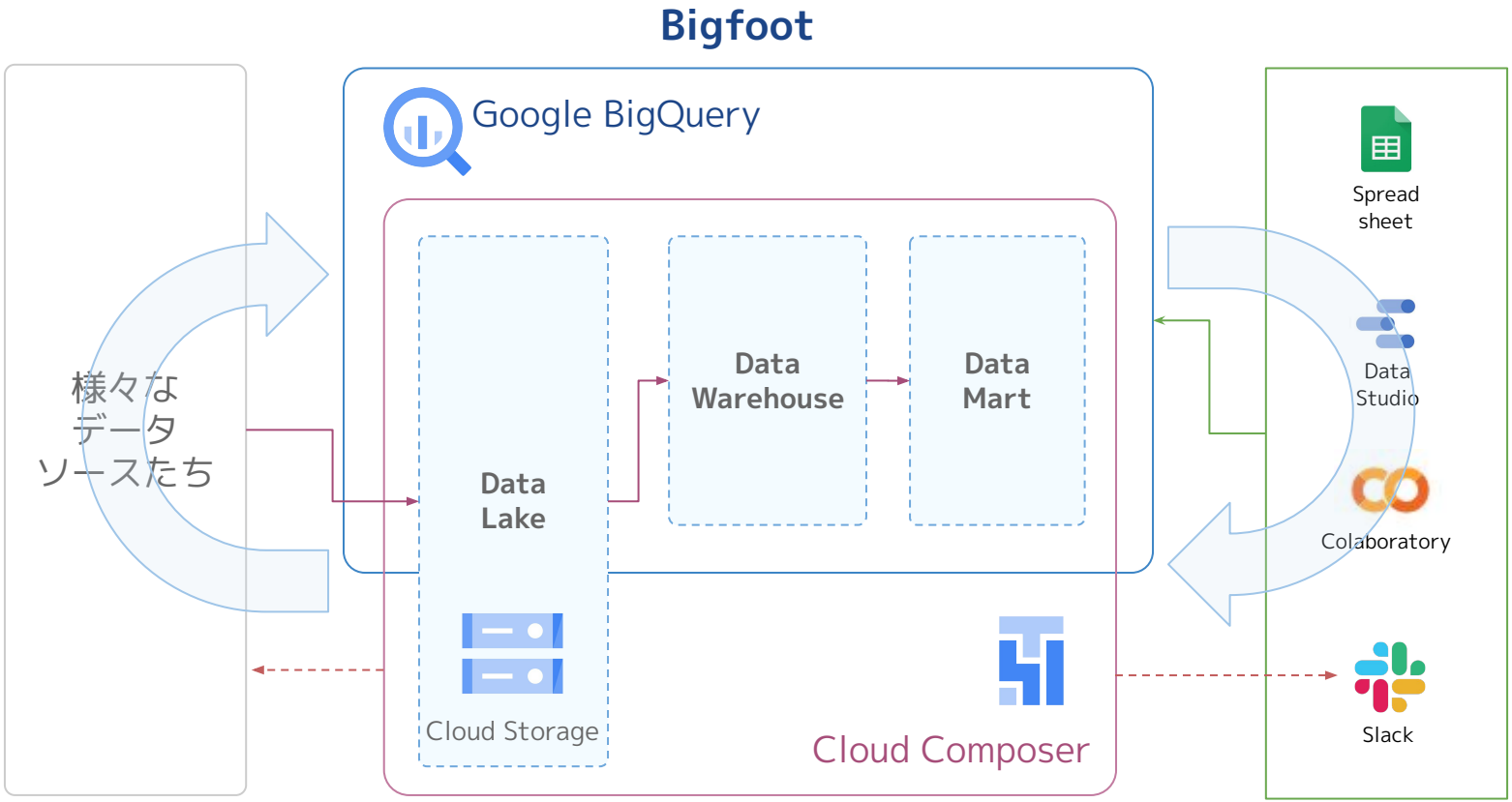
データエンジニアリングってなんですか？

おさらい

この研修で出てきたデータをマッピングしてみた



統一的な体験のために データを循環させる



参考資料(順不同)

- DAMA International, データマネジメント知識体系ガイド 第2版, 日経BP, 2018
- ゆずたそ ほか, データマネジメントが30分でわかる本 第2版, NextPublishing Authors Press, 2020
- 西田 圭介, ［増補改訂］ビッグデータを支える技術 ――ラップトップ1台で学ぶデータ基盤のしくみ WEB+DB PRESS plus, 技術評論社, 2021
- ミック, 達人に学ぶDB設計 徹底指南書, 翔泳社, 2012
- Valliappa Lakshmanan, スケーラブルデータサイエンス データエンジニアのための実践Google Cloud Platform, 翔泳社, 2019
- 下田 倫大 ほか, 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, 2021
- 上原 誠 ほか, AWSではじめるデータレイク: クラウドによる統合型データリポジトリ構築入門, テッキーメディア, 2020
- 塚本邦尊 ほか, 東京大学のデータサイエンティスト育成講座, マイナビ出版, 2019
- Wes McKinney, Pythonによるデータ分析入門 第2版 ―NumPy、pandasを使ったデータ処理, オライリージャパン, 2018
- Machine Learning Crash Course