社内の分析コンペ用のデータ分析基盤を作ってみた話

分析コンテスト2021

Sekikawa

データ分析基盤を作成するモチベーション

1. Pythonが書けなくとも, データを可視化できる何かが欲しい
   - チームメンバーは基本的にPython初心者

2. GPU環境が欲しい
   - 自然言語処理をするにはGPUが使えないと話にならない

3. データ分析の基盤部分の勉強
   - せっかくAWSの勉強をしたので実践の機会を作りたい

何で実現するか考える

1. データを可視化できる何かが欲しい

- 何でも良いからデータが可視化できれば良い
  - データから示唆を得ることが目的で、プログラミングは手段
    - Excel：データ量的に厳しい
    - BIツール：データ量は問題ない, エンジニア以外の人も使用するので簡単に可視化できそう

- 簡単に作れて料金が高くないBIツールが良い
  - AWS Quick Sight

何で実現するか考える

GPU環境が欲しい

- 基本的に自分しか使わないはず
  - AWS Sagemaker
    - Notebookインスタンス

Deep Learningのインスタンスは
P系統

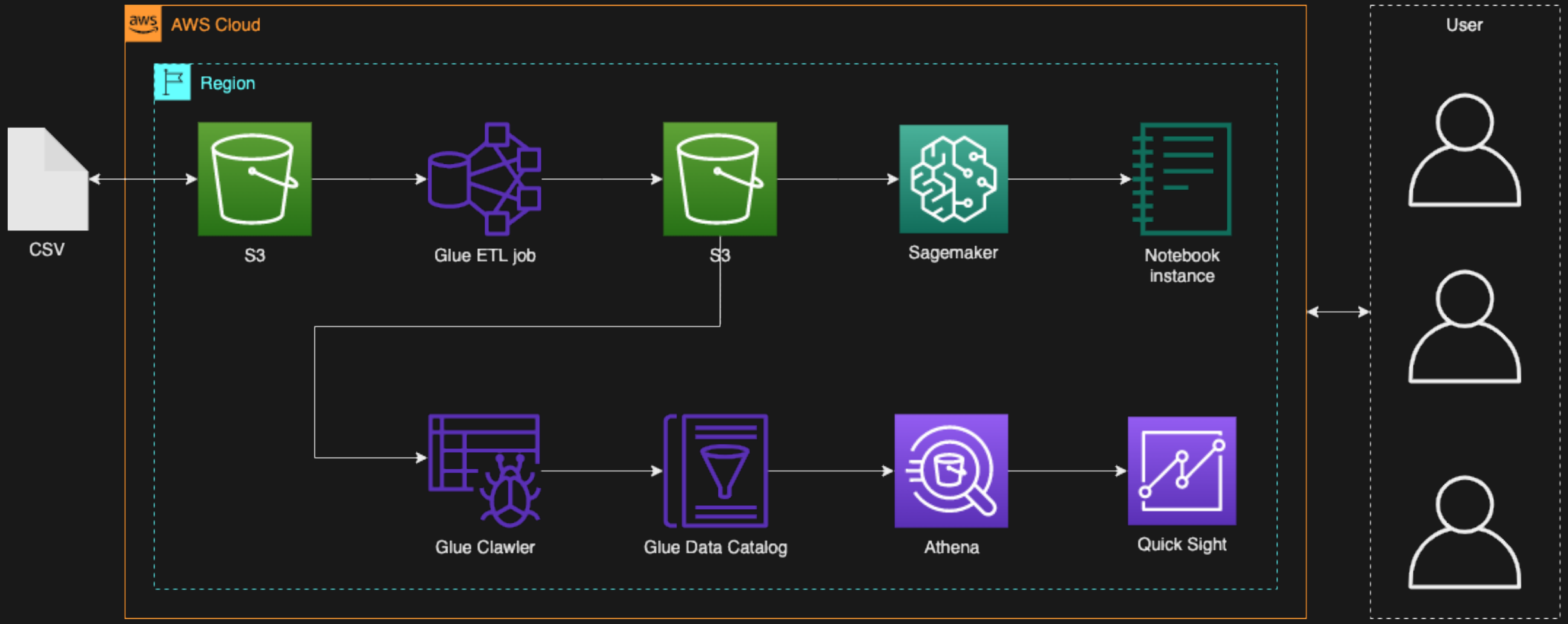
〇P系のスペック・価格比較表

インスタンス	GPUの種類	GPU数	クロック(MHz)	CUDAコア数	GPUメモリ(GiB)	vCPU	RAM	価格/時間(USD)
p2.xlarge	Tesla K80	1	560	2,496	12	4	61	$0.9000
p2.8xlarge	Tesla K80	8	560	19,968	96	32	488	$7.2000
p2.16xlarge	Tesla K80	16	560	39,936	192	64	732	$14.4000
p3.2xlarge	Tesla V100	1	1,245	5,120	16	8	61	$3.0600
p3.8xlarge	Tesla V100	4	1,245	20,480	64	32	244	$12.2400
p3.16xlarge	Tesla V100	8	1,245	40,960	128	64	488	$24.4800
p3dn.24xlarge	Tesla V100	8	1,230	40,960	256	96	768	$31.2120

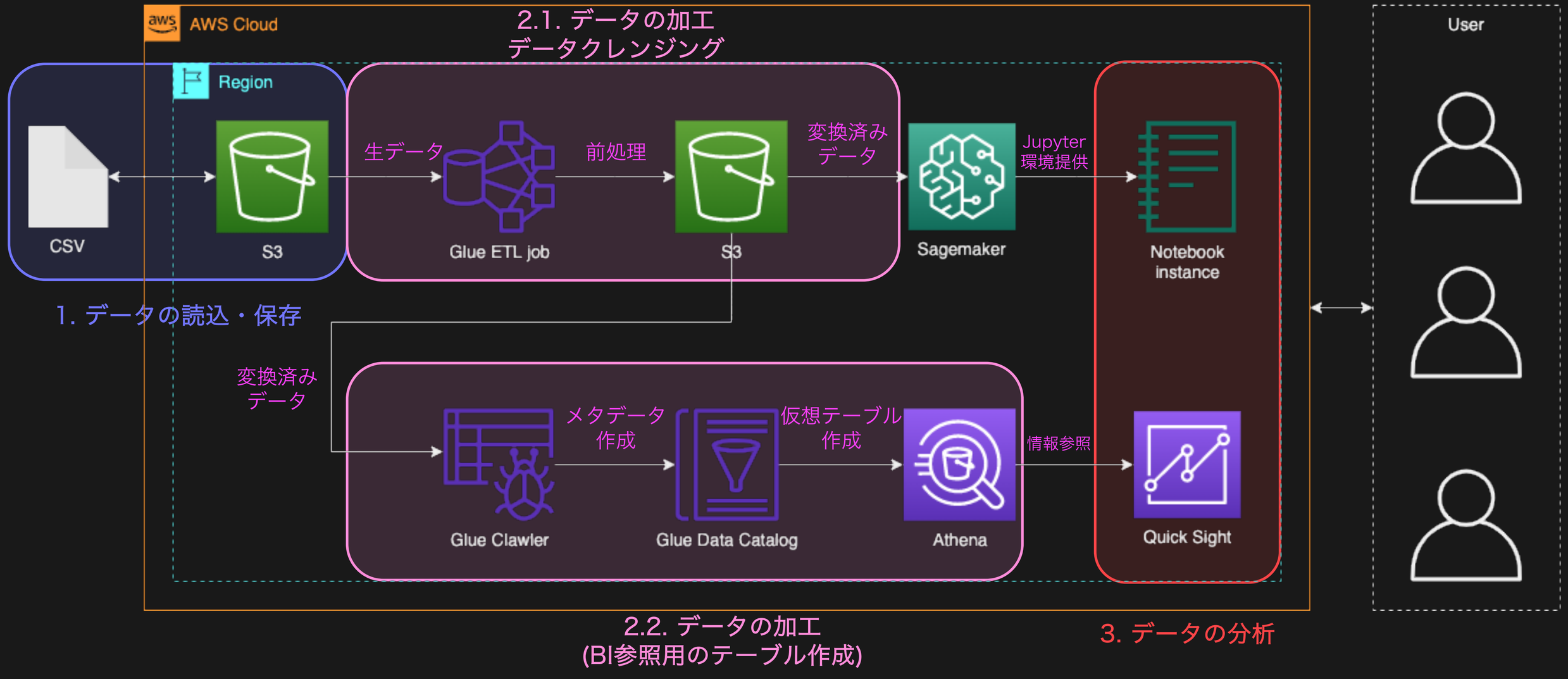
データ分析基盤の流れ

1. **収集・保管**
   - CSVファイル(生データ)を読み込み・保存をする
2. **加工**
   - いらないデータを削除や紐付けなどをして, 分析しやすいデータにする
3. **分析**
   - 可視化・統計・機械学習などやりたいことをする
   - 今回は、BIツールでの可視化とGPUインスタンスへのアクセスまで

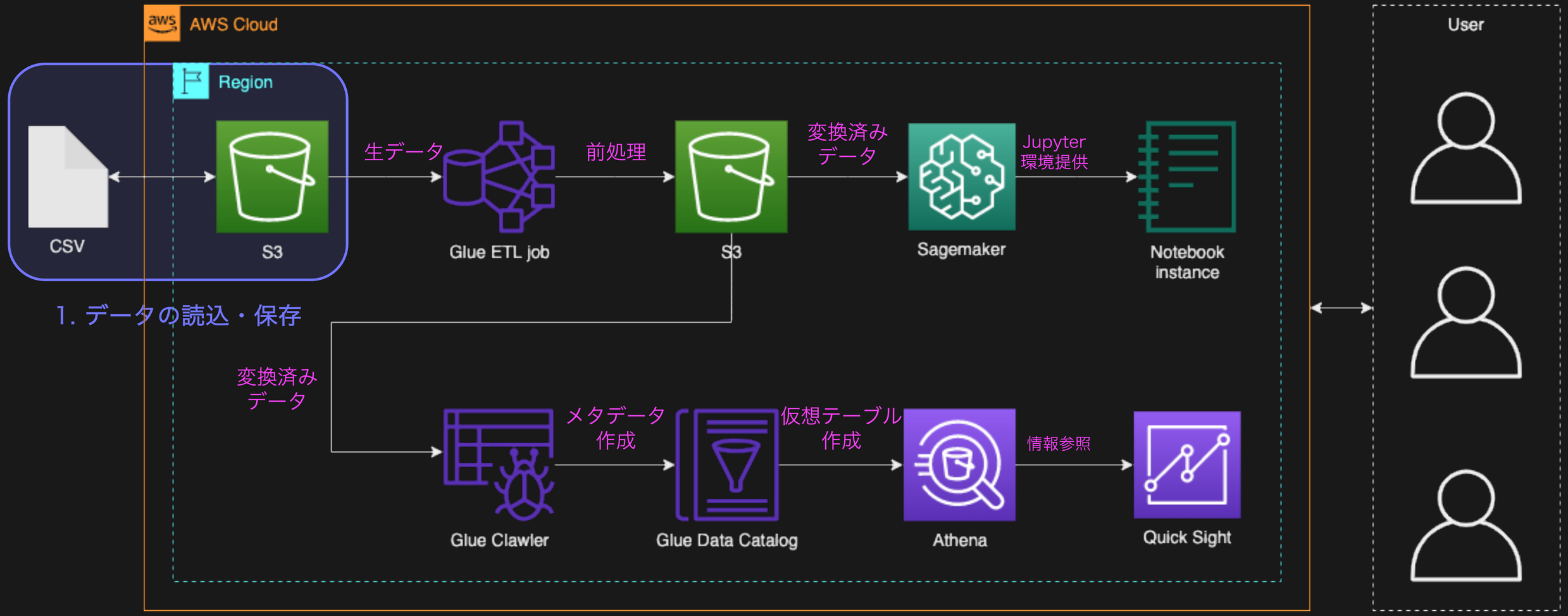
データ分析基盤の全体図



データ分析基盤の全体図

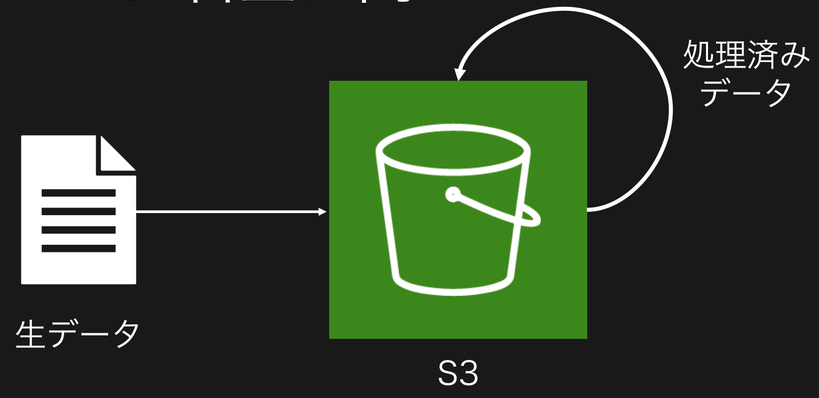


データ分析基盤の全体図



1. データの取り込み・保存

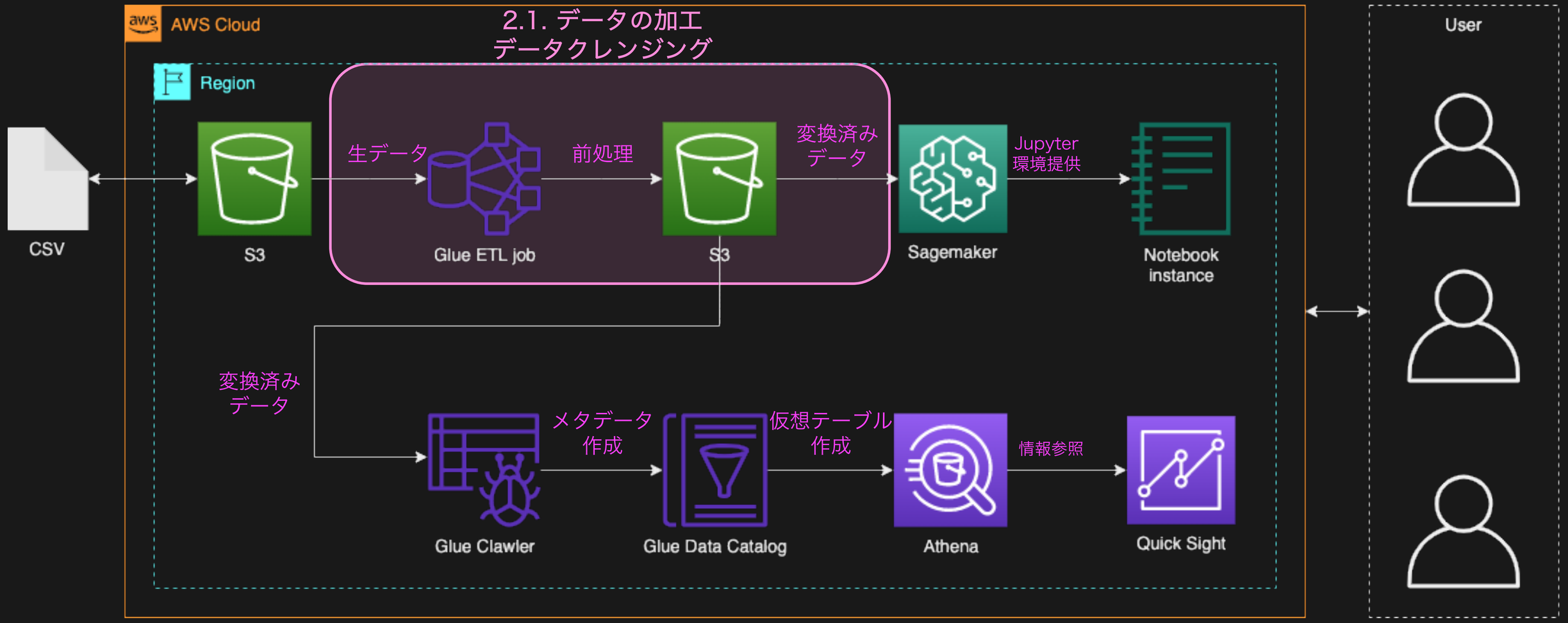
- 今回はS3でデータを取り込み
  - その後のデータ(処理済みデータ)も全てS3でデータを保持する
    - Redshiftは料金が高い



今回は、
データレイク：S3
データウェアハウス：S3 (+Athena)

一般的なのは以下の構成？
データレイク：S3
データウェアハウス：Redshift

データ分析基盤の全体図



2. データの加工

データクレンジング(ETL)

- f(input) = output
  - input：生のCSVファイル (S3)
  - output： どんなoutputが欲しいか
  - f：input → outputを実現するような処理

2. データの加工

データクレンジング(ETL)

- データの可視化の部分とJupyterでの分析で使いたい形式にしたい
  - Jupyter(GPU)での分析
    - その都度自分でコードを書くので最低限の処理のみで良い
  - データの可視化
    - 前もってデータの結合が必要？
      - Athena使えばSQLでJoin出来そうな気がする
    - Quick Sightの制約が分からない → とりあえず最低限の処理

2. データの加工

データクレンジング(ETL)

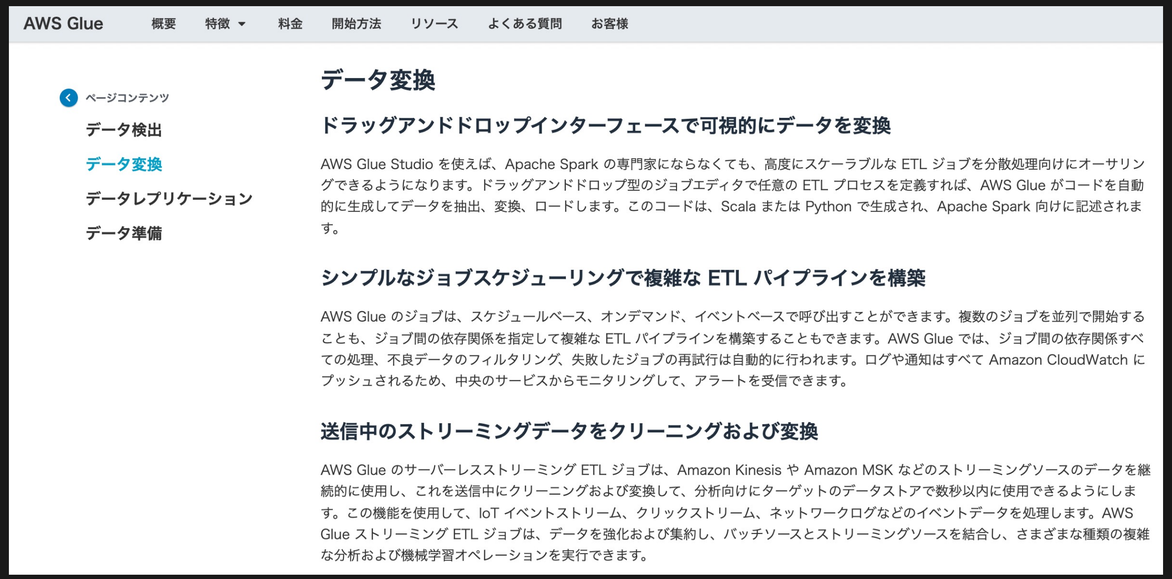
- 最低限の処理
  - 私がいつも行う前処理のみ
    - 下記のデータを削除
      - 全て同じ値の列
      - 全てNanの列
      - 重複レコード
      - 全く同じ情報を持つ列
      - 全く同じ情報を持つ行

```python
def f(df):
    print("##### Start Preprocessing #####\n")
    # ----- 全て同じ値・欠損値の列，重複する列を削除する
    # ----- 全く同じ情報を持つ行を削除する
    def dropColumns(df):
        print("##### Start dropColumns #####")
        AllColumns = set(df.columns)
        dropColumns = set()
        prevDropColumns = set()

        # ----- 全てnanの列の抽出
        isAllNanDict = dict(df.isna().all())
        for col, isAllNan in isAllNanDict.items():
            if isAllNan and col not in dropColumns:
                dropColumns.add(col)
        print("### All Nan Columns ###")
        print(dropColumns - prevDropColumns)
        prevDropColumns = dropColumns.copy()
        # -----

        # ----- 全て同じ値の列の抽出
        isUniqueDict = dict(df.nunique(axis=0) == 1)
        for col, isUnique in isUniqueDict.items():
            if isUnique and col not in dropColumns:
                dropColumns.add(col)
        print("### All Same Value Columns ###")
        print(dropColumns - prevDropColumns)
        prevDropColumns = dropColumns.copy()
        # -----

        # ----- 同じ情報を持つ列の抽出
        # temp = pd.DataFrame()
        # notDropedColumns = AllColumns - dropColumns
        # for col in notDropedColumns:
        #     labels, uniques = pd.factorize(df[col])
        #     temp[col] = labels
        # dropColumns = (dropColumns) | (notDropedColumns - set((temp.T.drop_duplicates().T.columns)))
        # print("### All Same Info Columns ###")
        # print(dropColumns - prevDropColumns)
        # prevDropColumns = dropColumns.copy()
        # -----

        # ----- 列の削除
```

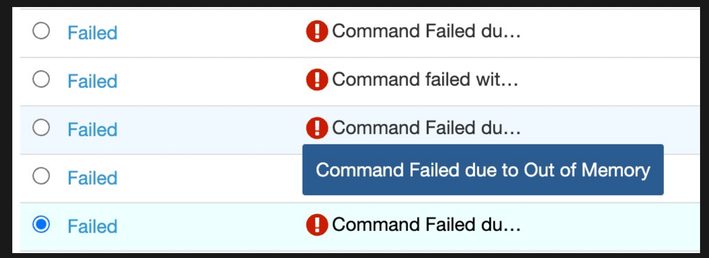
この二つは計算量がかなり大きいので、
今回は処理しない

2. データの加工

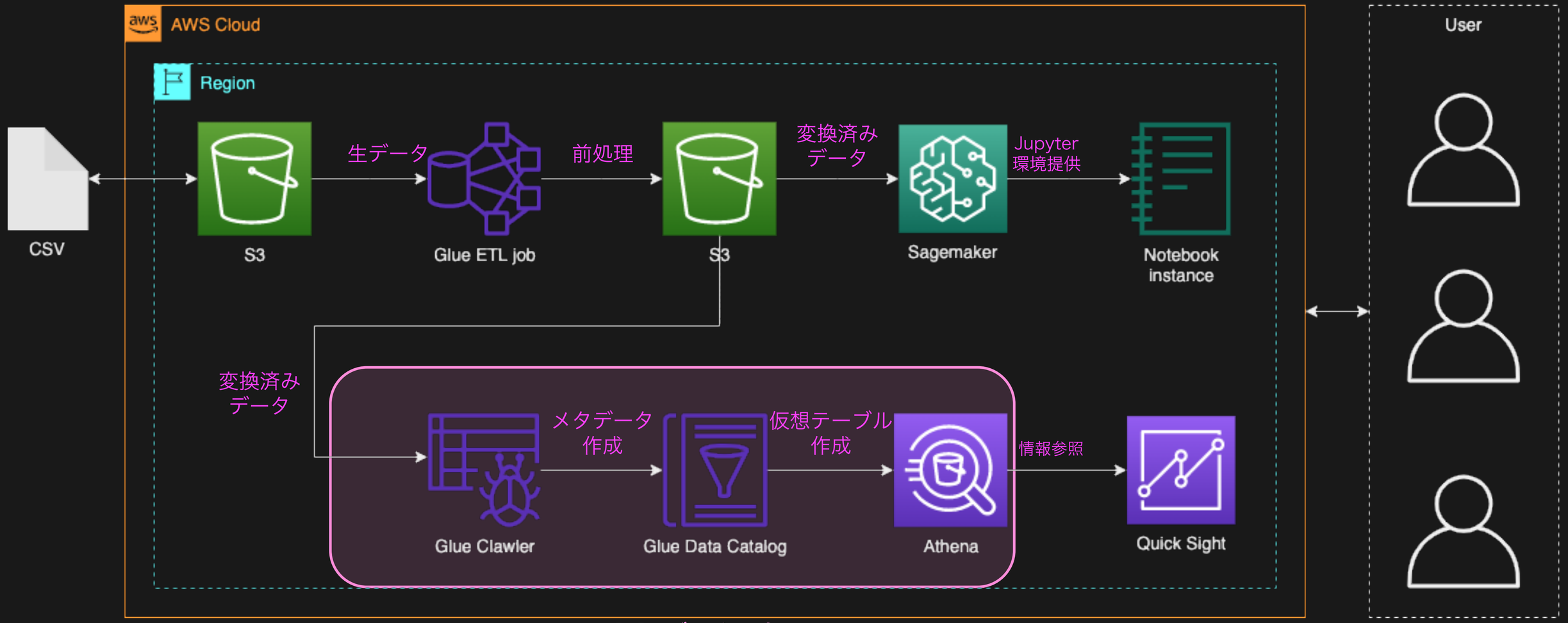
データクレンジング(ETL)



- Glue(ETL)で実施する
  - データ前処理に特化したLambda
    - Sparkなどの分散処理やML変換をサポート
    - Extract(抽出), Transpose(変換), Load(ロード)
  - 今回詰まったのは大体Glue関連
    - Out of Memory
      - メモリを上限まで上げて解決 → 本来はPythonではなくPySparkとかの分散処理を使った方が良い
    - Data Catalog作成時、指定したS3パスにファイル置いてるのにファイルないとか
      - 一つのフォルダに一つの処理対象ファイルを置かないとダメ、とか（~~初見殺しでは~~）



データ分析基盤の全体図



2.2. データの加工
(BI参照用のテーブル作成)

2. データの加工

BI参照用のテーブル作成

- Quick Sightが読み込めるデータソースは多く、S3のデータも読み取れる
  - Redshift, S3, Athena, Aurora, …

- 少しデータをいじって可視化したい時にS3(CSV)だと面倒
  - 処理済みCSVは、最低限の処理だけ行った元データとして残してもおきたい

- Athenaで仮想テーブルを作れば、SQL叩いてデータ結合して読み込みとか出来そう
  - CSVファイルのままだと結合してデータ読み込みとか出来ない？
  - 単純にAthena使ってみたいという気持ちもあった

2. データの加工

BI参照用のテーブル作成

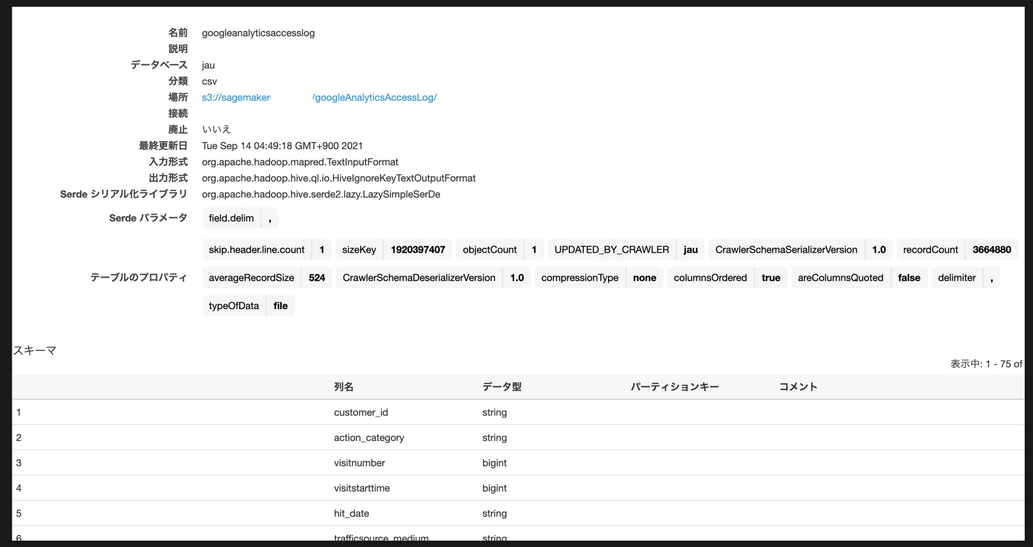
- Athena
  - AWSの言葉を借りると、
    - “S3 内のデータを標準 SQL を使用して簡単に分析” できるサービス
  - S3のデータを元に仮想テーブルを作る
    - CSVファイルのデータをあたかもDBのデータのように扱える
    - 既存のデータを元に、新規テーブルも作成できる

2. データの加工

BI参照用のテーブル作成

› Athenaで仮想テーブルを作るまで

1. クローラを作成する
   - Glue Clawler → 指定したS3バケットを探索する



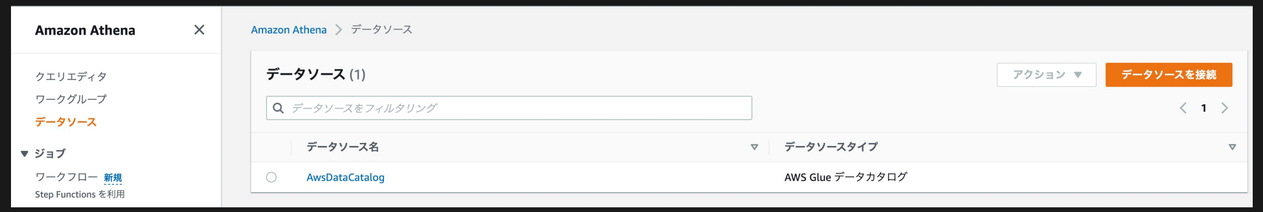
Glue Data Catalog

2. クローラを動かして、メタデータを作成する
   - Glue Data Catalog → テーブル定義のようなものをクローラが自動作成する

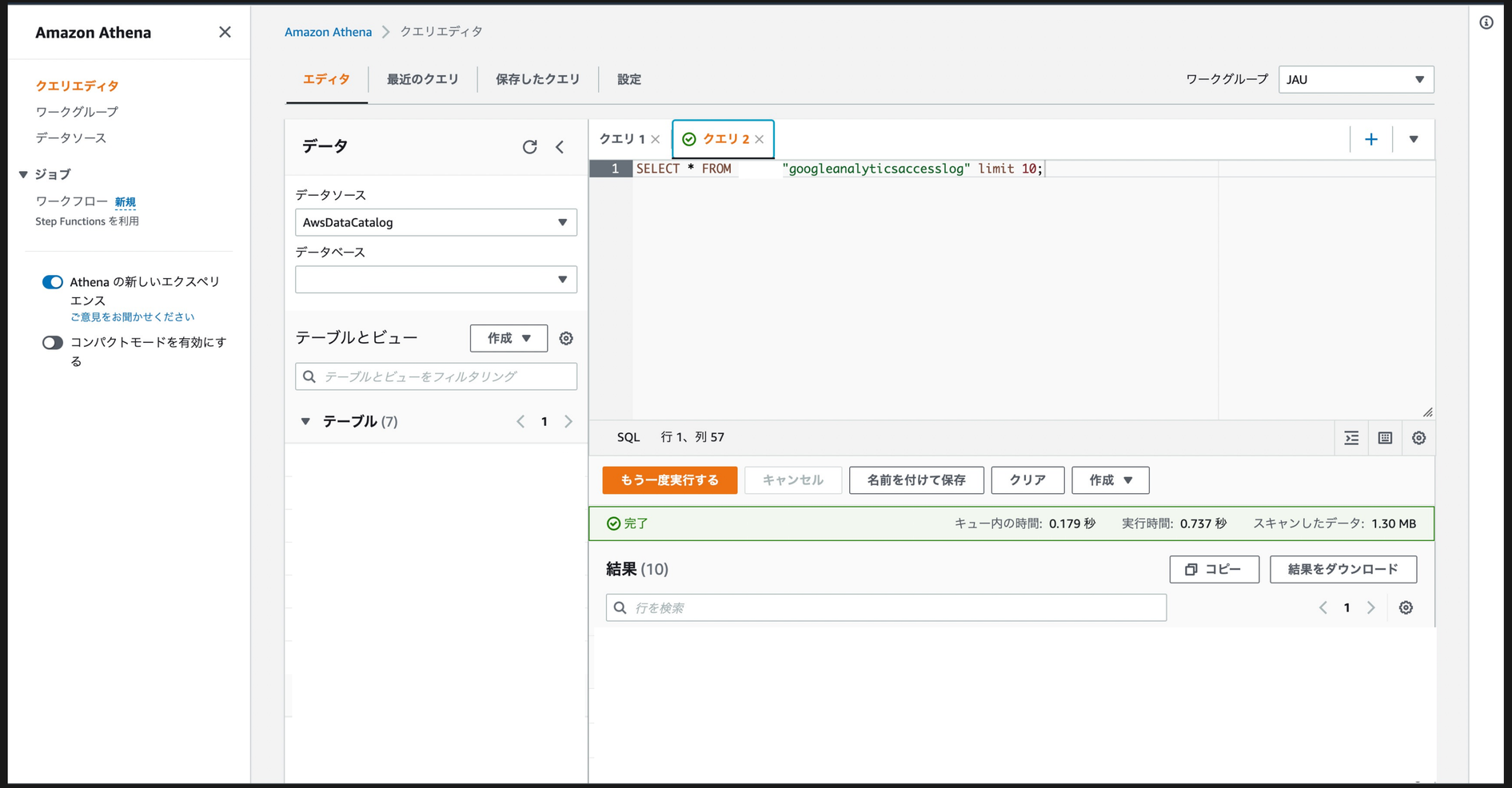
3. 作成したメタデータを元に、仮想テーブルを作成する

2. データの加工

BI参照用のテーブル作成

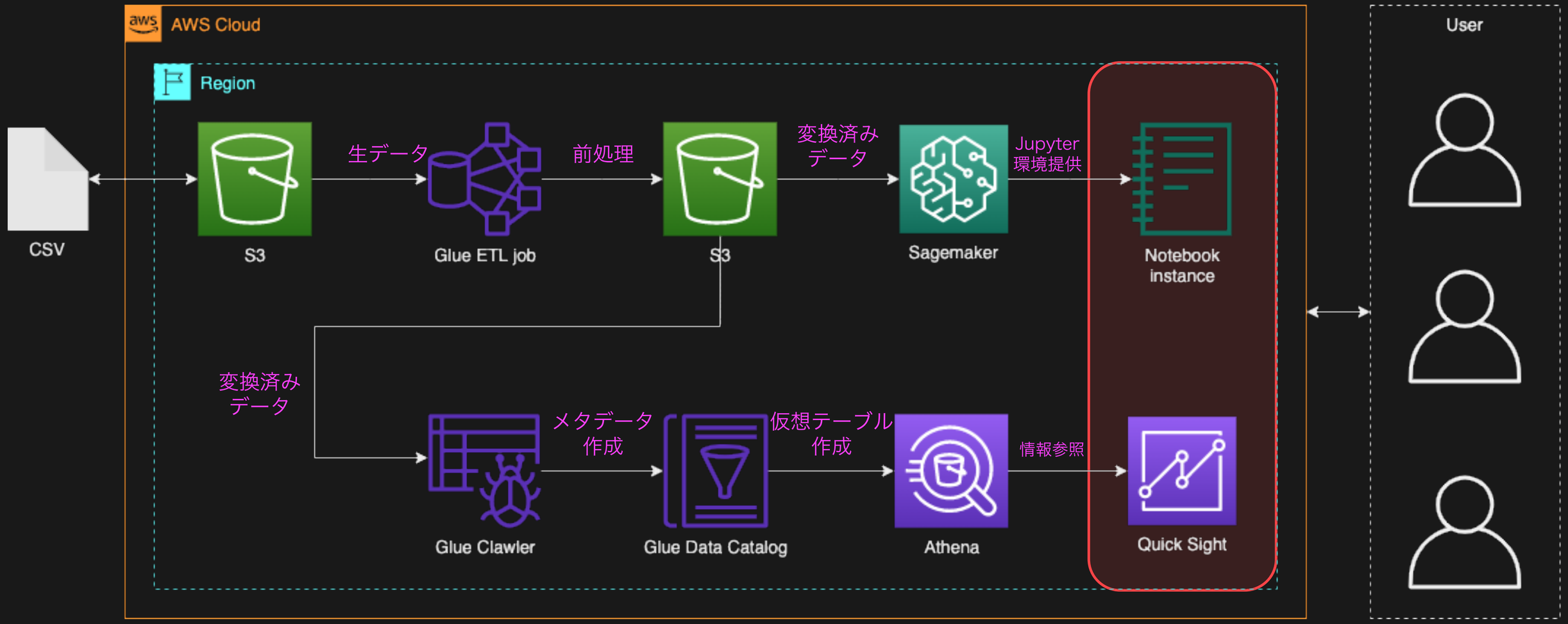


先程のData Catalog
がデータ元



Athena

データ分析基盤の全体図



3. データの分析

- Sagemaker
  - AWSにおけるML/DLサービスの根幹
    - ”すべてのデータサイエンティストとデベロッパーのための機械学習”
  - Sagemakerで提供するサービスがとても多い
    - Notebookインスタンス, モデルのデプロイ(API化), AutoML, パイプライン, モニタリング, …
    - 今回使うのはNotebookインスタンスのみ

3. データの分析

- Quick Sight
  - AWSのサービスと相性の良いBIツール
    - “最も人気のあるクラウドネイティブのサーバーレスビジネスインテリジェンスサービス”
  - 少なくともAthenaをデータソースとするとモデリング言語を書く必要がない
    - ぽちぽちするだけでデータインポートできる
    - もちろんSQLでテーブルJOINしてインポートとかもできる

データ分析基盤は役に立ったか？

- Quick Sight (可視化)
  - Datailkuに敗北
    - BIと言っても使いこなすのはそれなりに難しく、私も教えられない
    - 早めに見切りをつければ良かった(月30ドル強)
- Sagemaker (GPU)
  - Google Colab Proに敗北
    - 月々1200円で使い放題の安さ

勉強になったし、楽しかったのでOK

感想

- 去年と比べると分析基盤周りの知識はかなり増えた
  - 構成の全体図に関しては簡単にイメージできた
  - けど、実際に手を動かすとつまる
    - 実践は足りない (知ってる → 出来るまで押し上げる)
- ベストプラクティスが分からない
  - 実用レベルの基盤にはなっていない (今回は簡易的なものがあれば良かったのでそれで良いけど)
  - 実用レベルの基盤との違いを挙げられない (実用レベルの分析基盤を知らない)

Next Step

- GCPにおける分析基盤の勉強 (知ってるレベル)
  - AWSと結構違うはずなので知識を増やすところから
- AWSにおける分析基盤の実践経験を増やす (出来るレベル)
  - 今回は簡易的な基盤 → より実用レベルに近い基盤作りの実践
- Sagemakerを深く触ってみる (出来るレベル)
  - ML/DLモデルをデプロイして公開するとかしてみたい