

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

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低い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

○ 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

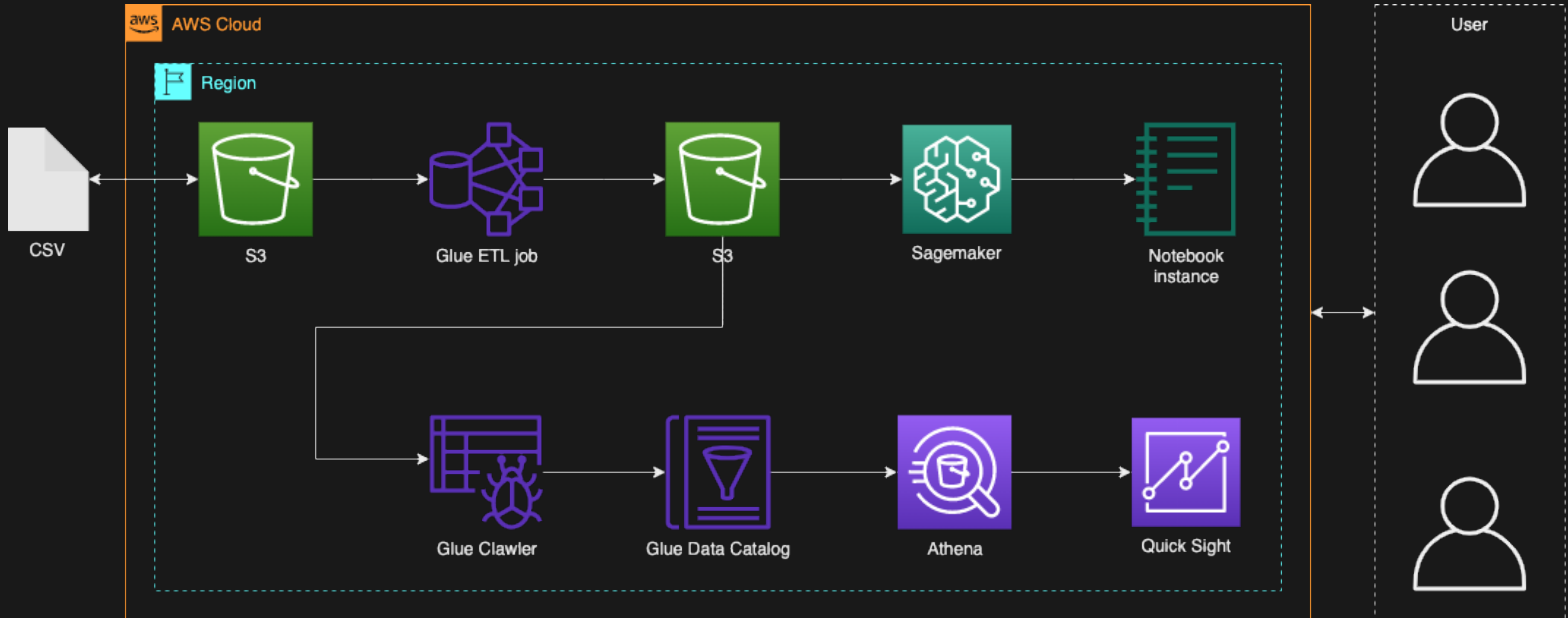
2. 加工

- › 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

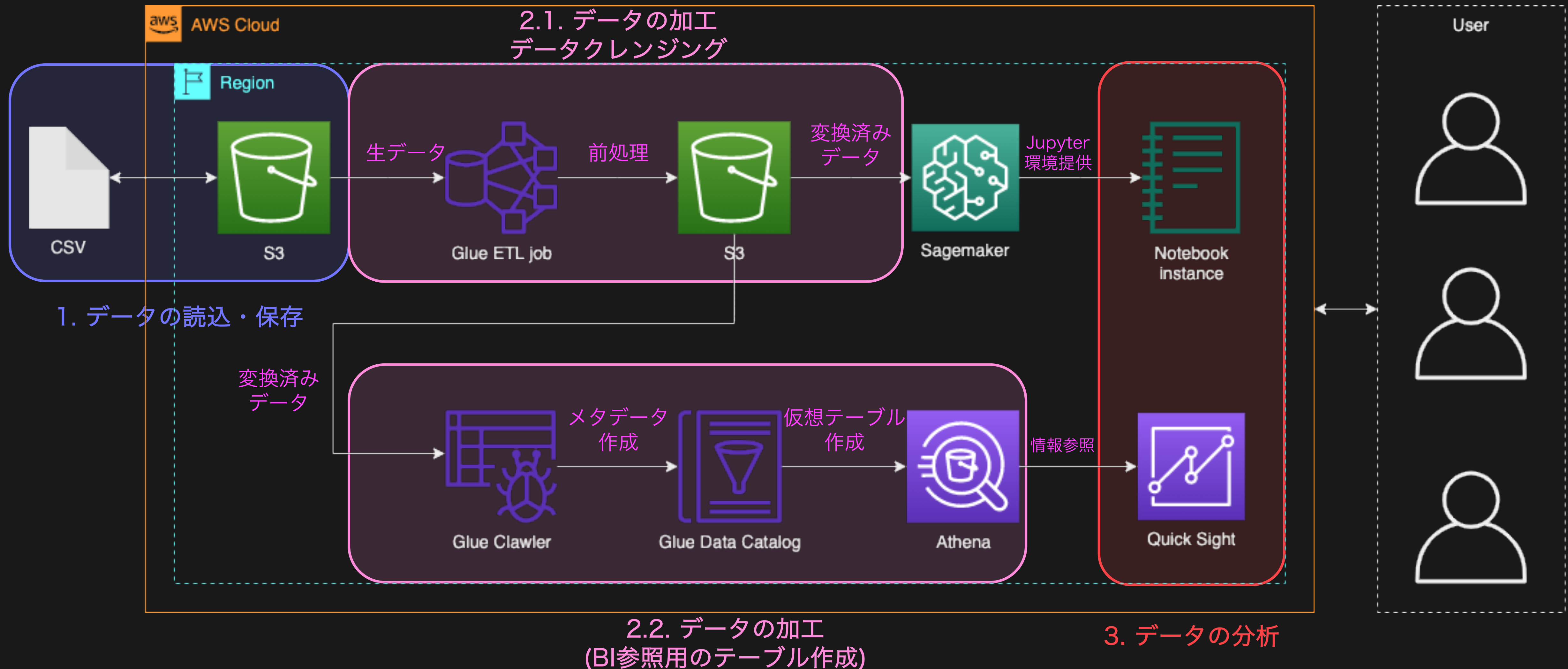
3. 分析

- › 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

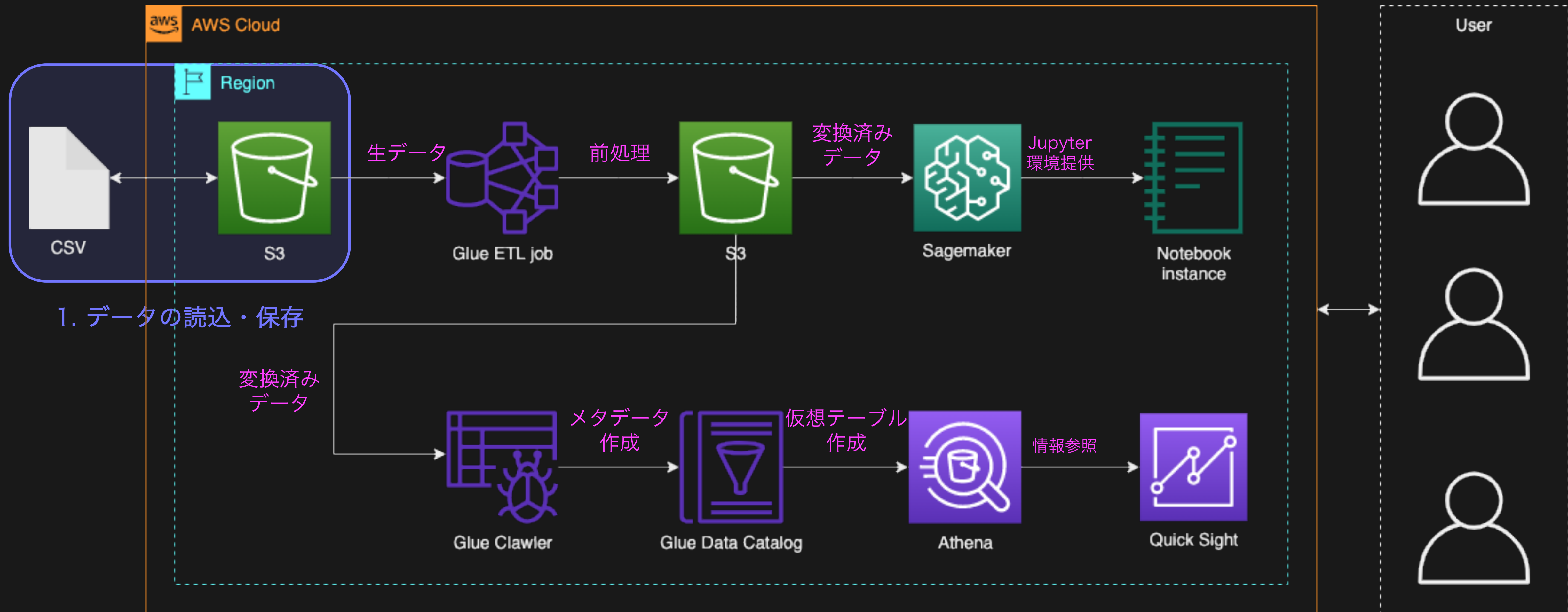
データ分析基盤の全体図



データ分析基盤の全体図



データ分析基盤の全体図

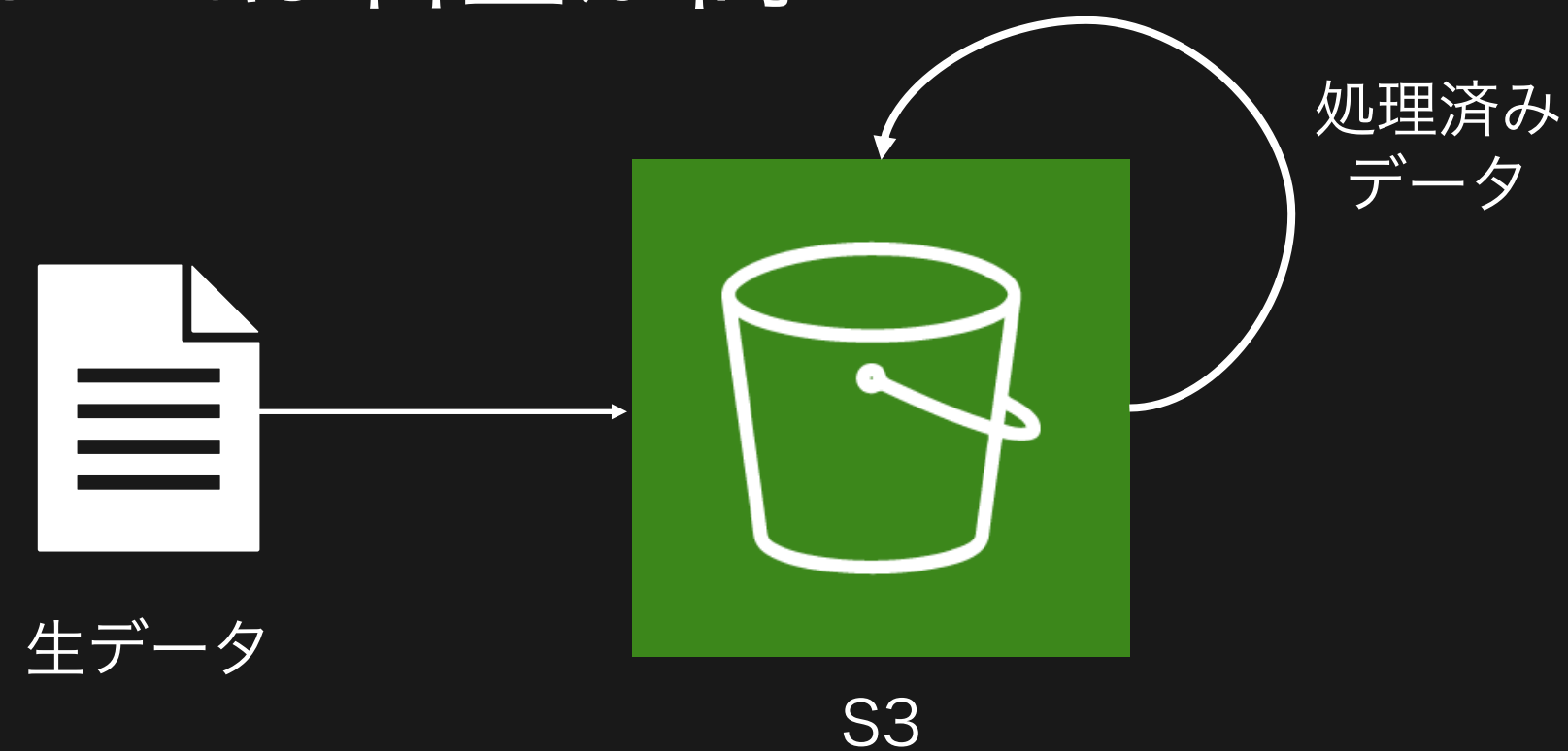


1. データの取り込み・保存

＞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

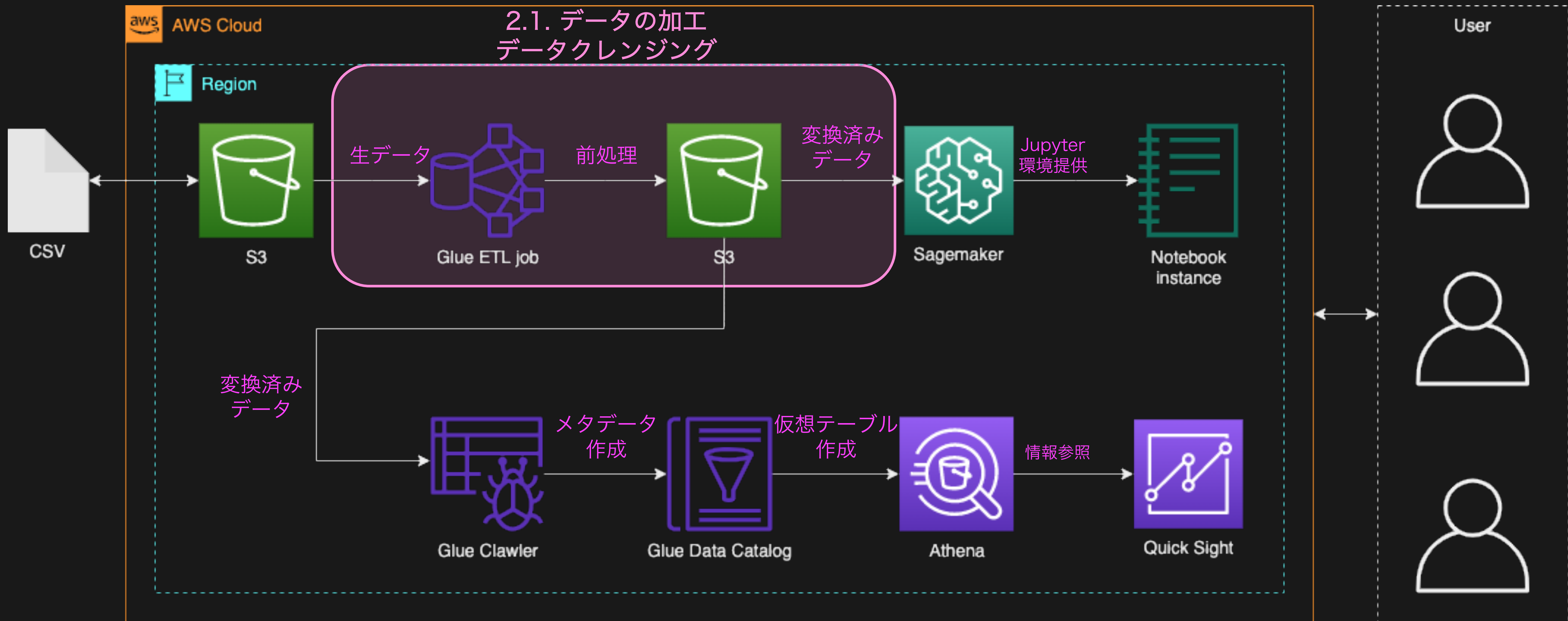


今回は、
データレイク：S3
データウェアハウス：S3 (+Athena)



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)

＞ $f(\text{input}) = \text{output}$

＞ input : 生のCSVファイル (S3)

＞ output : どんなoutputが欲しいか

＞ f : input \rightarrow 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

```
20 def f(df):
21     print("#### Start Preprocessing ####\n")
22     # ----- 全て同じ値・欠損値の列, 重複する列を削除する
23     # ----- 全く同じ情報を持つ行を削除する
24     def dropColumns(df):
25         print("#### Start dropColumns ####")
26         AllColumns = set(df.columns)
27         dropColumns = set()
28         prevDropColumns = set()
29
30         # ----- 全てnanの列の抽出
31         isAllNanDict = dict(df.isna().all())
32         for col, isAllNan in isAllNanDict.items():
33             if isAllNan and col not in dropColumns:
34                 dropColumns.add(col)
35         print("### All Nan Columns ###")
36         print(dropColumns - prevDropColumns)
37         prevDropColumns = dropColumns.copy()
38         # -----
39
40         # ----- 全て同じ値の列の抽出
41         isUniqueDict = dict(df.nunique(axis=0) == 1)
42         for col, isUnique in isUniqueDict.items():
43             if isUnique and col not in dropColumns:
44                 dropColumns.add(col)
45         print("### All Same Value Columns ###")
46         print(dropColumns - prevDropColumns)
47         prevDropColumns = dropColumns.copy()
48         # -----
49
50         # ----- 同じ情報を持つ列の抽出
51         # temp = pd.DataFrame()
52         # notDroppedColumns = AllColumns - dropColumns
53         # for col in notDroppedColumns:
54             # labels, uniques = pd.factorize(df[col])
55             # temp[col] = labels
56         # dropColumns = (dropColumns) | (notDroppedColumns - set((temp.T.drop_duplicates().T.columns)))
57         # print("### All Same Info Columns ###")
58         # print(dropColumns - prevDropColumns)
59         # prevDropColumns = dropColumns.copy()
60         # -----
61
62         # ----- 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)

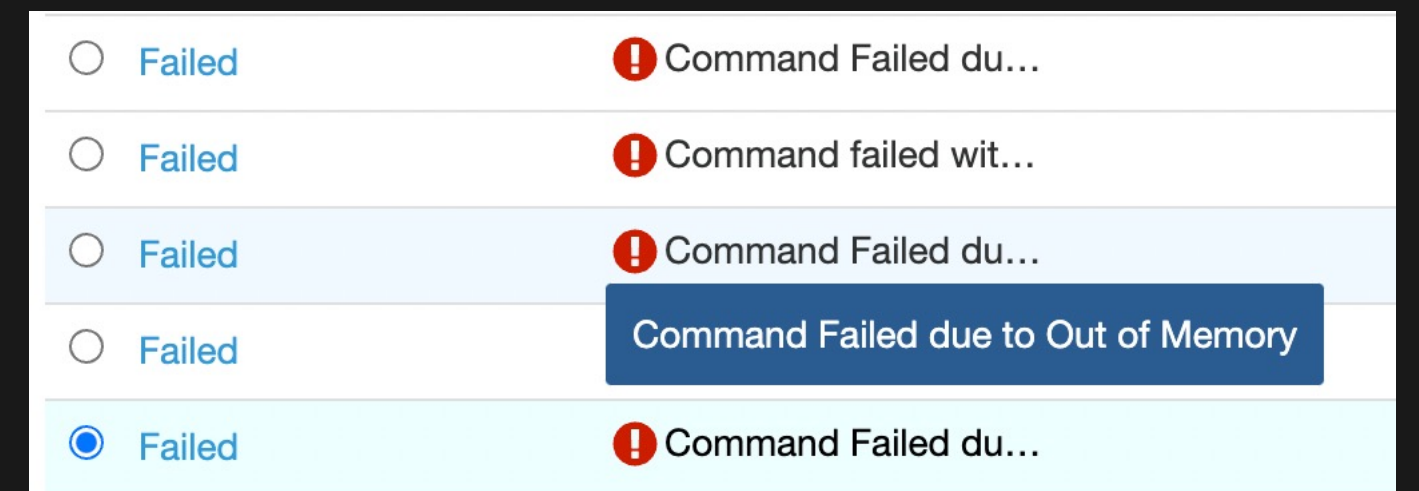
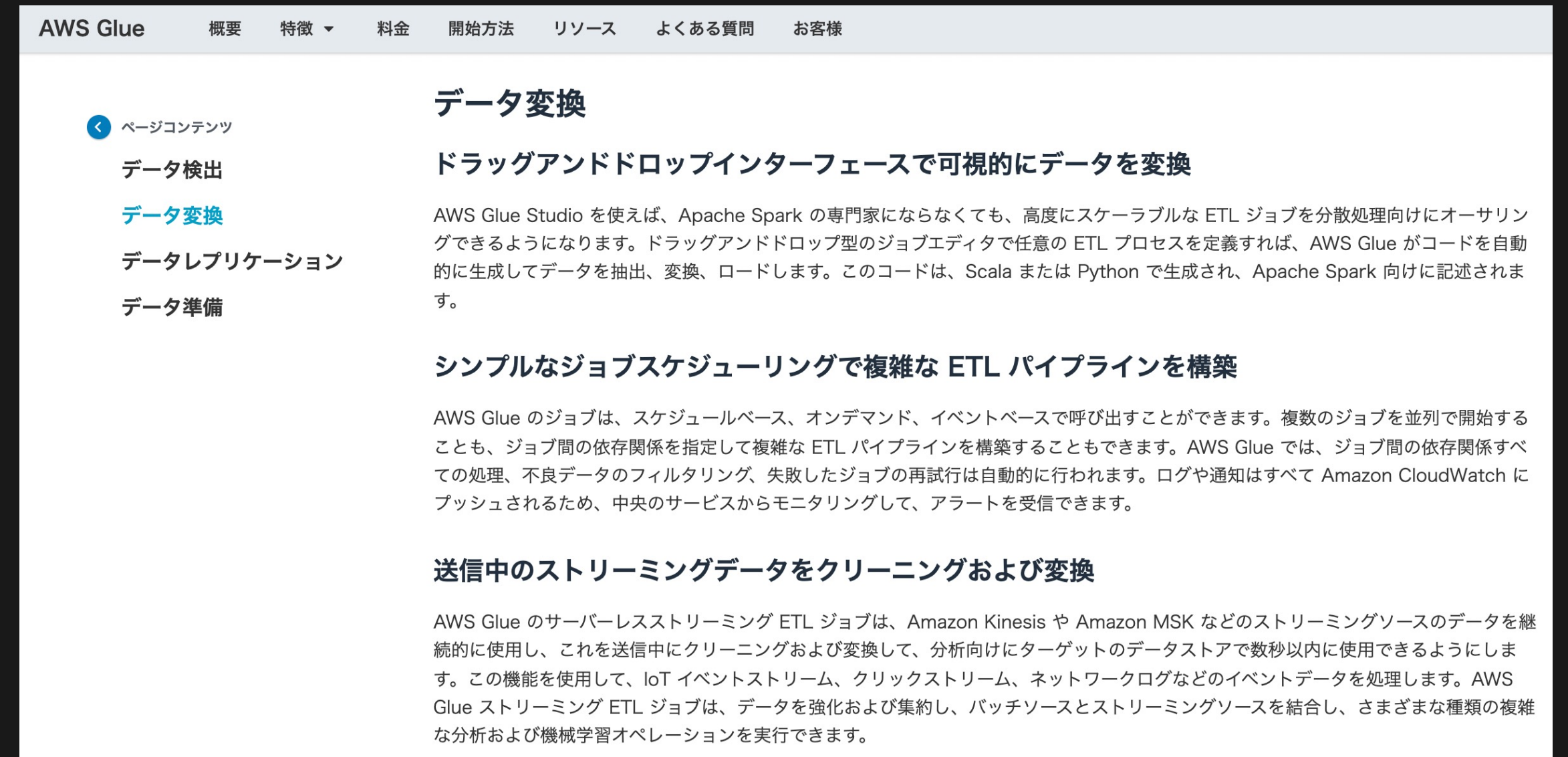
› 今回詰まったのは大体Glue関連

› Out of Memory

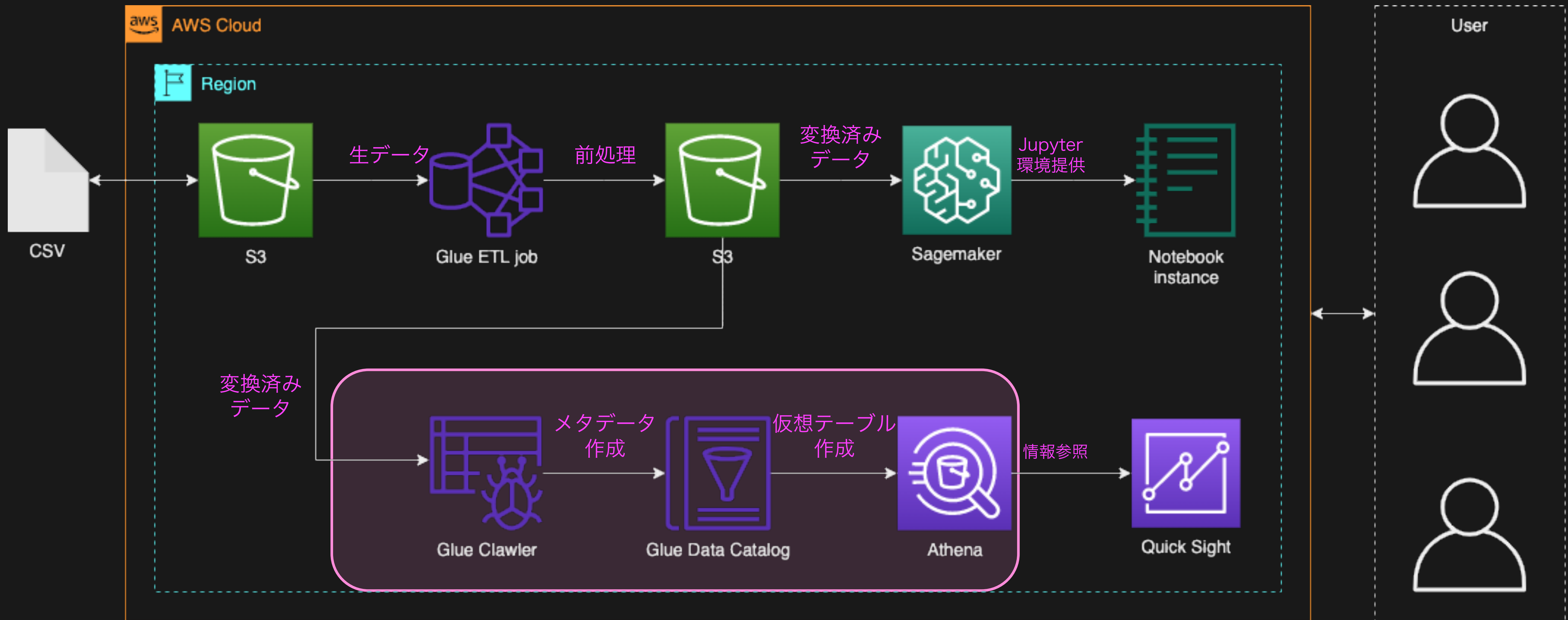
› メモリを上限まで上げて解決 → 本来はPythonではなくPySparkとかの分散処理を使った方が良い

› 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

Amazon Athena

概要

特徴

料金

開始方法

リソース

よくある質問

◀ ページの内容

サーバーレス

簡単に利用開始

クエリが容易

クエリごとの料金

高速なパフォーマンス

高い可用性と耐久性

セキュア

統合

横串検索

機械学習

標準 SQL を使用するだけで簡単にクエリを実行

Amazon Athena では [Presto](#) が使用されます。Presto は、低レイテンシーでアドホックなデータ分析用に最適化された、オープンソースの分散 SQL クエリエンジンです。つまり、大規模な結合、ウィンドウ関数、配列を完全にサポートしている ANSI SQL を使用して、Amazon S3 内の大規模なデータセットに対してクエリを実行できます。Athena では、CSV、JSON、ORC、Avro、Parquet といったさまざまなデータ形式がサポートされています。Athena の JDBC ドライバーを使用して、さまざまな BI ツールから Athena に接続することもできます。

クエリごとの料金

Amazon Athena では、実行したクエリに対してのみ料金が発生します。各クエリでスキャンされるデータ量に基づいて課金されます。データの圧縮、分割、列形式への変換を行うと、大幅なコスト削減とパフォーマンス向上を実現できます。このようなオペレーションにより、Athena でクエリを実行するためにスキャンする必要のあるデータ量が減少するためです。

高速なパフォーマンス

Amazon Athena では、高速なパフォーマンスを得るために、クラスターの管理やチューニングについて心配する必要はありません。Athena は、Amazon S3 で高速なパフォーマンスが得られるように最適化されています。Athena ではクエリが自動的に並列で実行されるため、大規模なデータセットであってもクエリ結果が数秒で表示されます。

BI参照用のテーブル作成

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

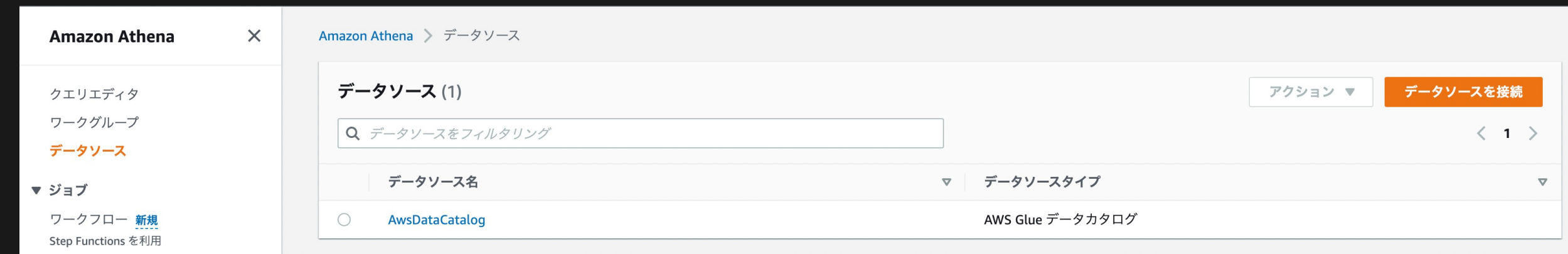
- Glue Data Catalog → テーブル定義のようなものをクローラが自動作成する

3. 作成したメタデータを元に、仮想テーブルを作成する

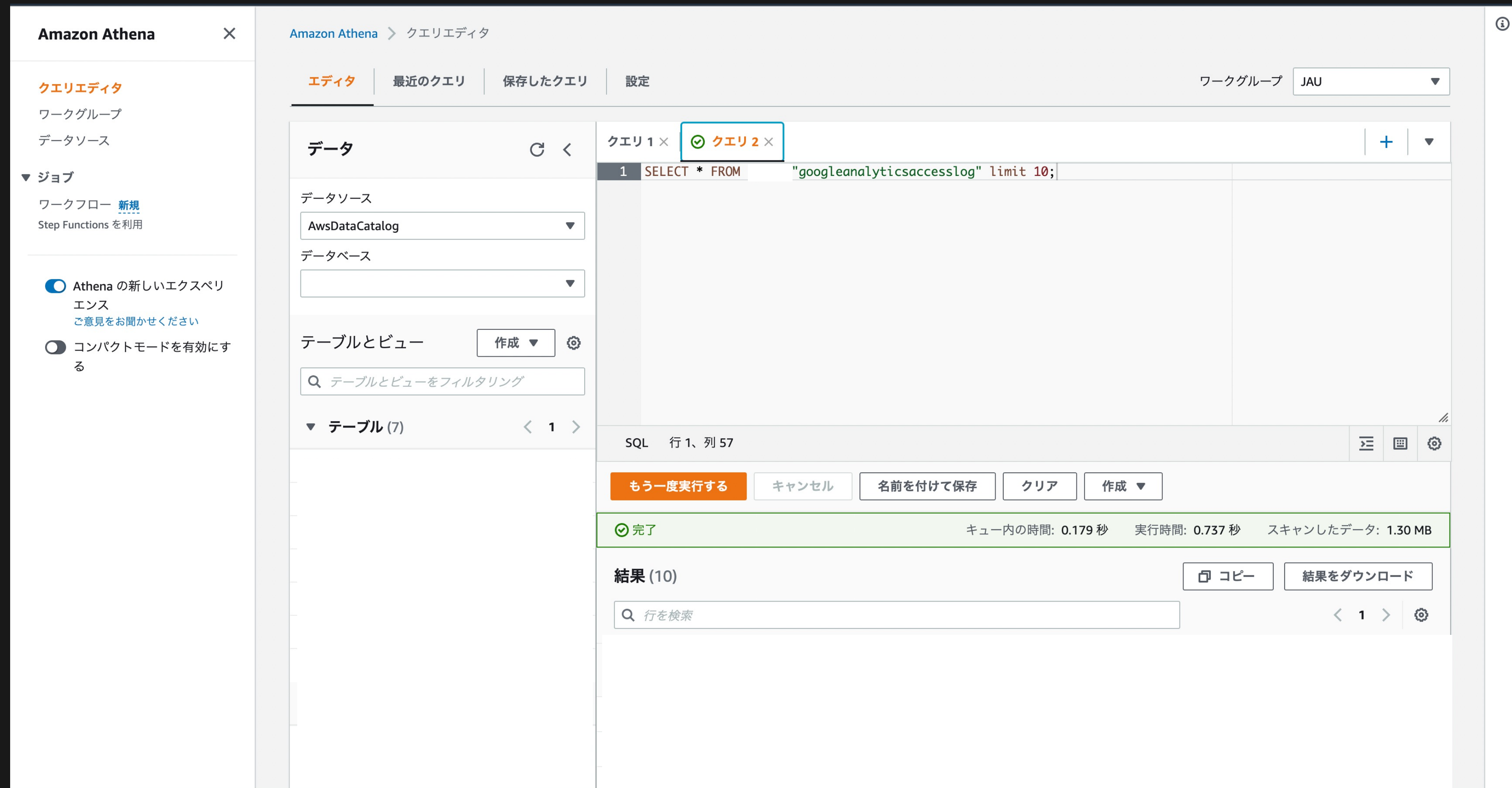
スキーマ					表示中: 1 - 75 of
	列名	データ型	パーティションキー	コメント	
1	customer_id	string			
2	action_category	string			
3	visitnumber	bigint			
4	visitstarttime	bigint			
5	hit_date	string			

2. データの加工

BI参照用のテーブル作成

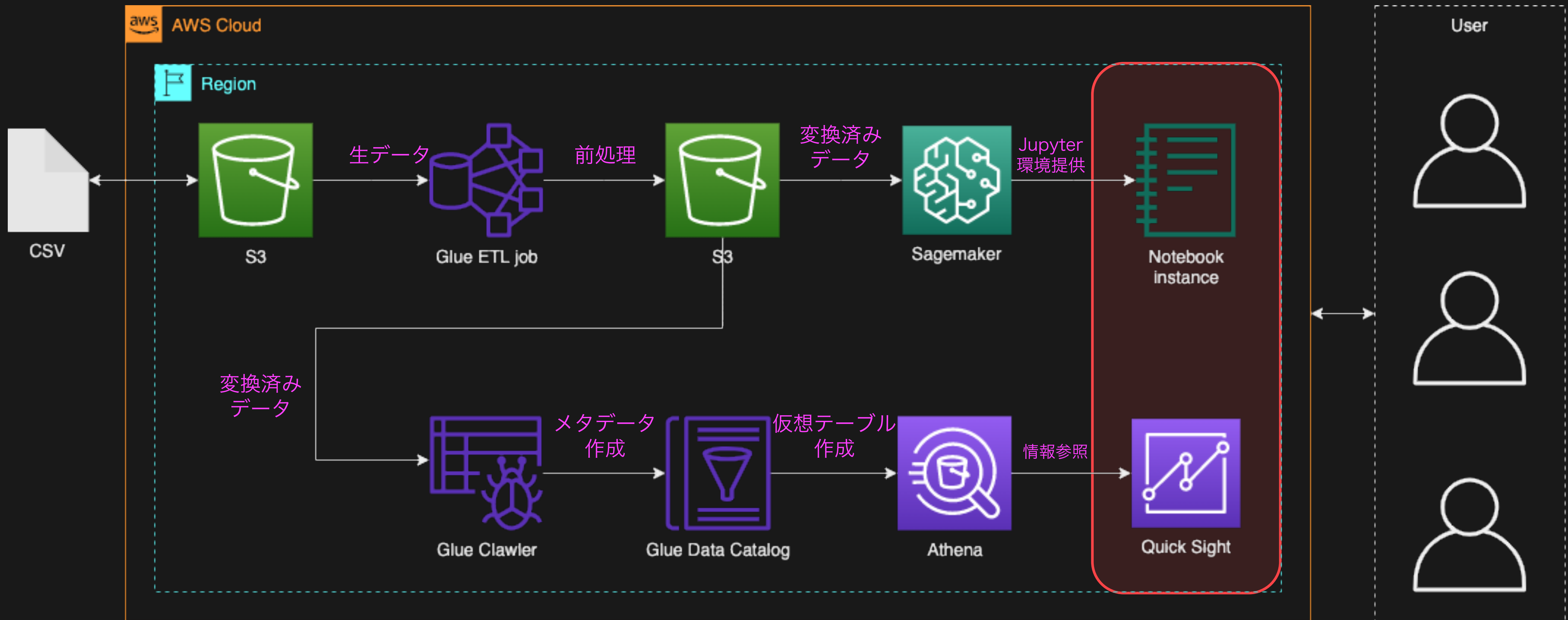


先程のData Catalog
がデータ元



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

インテントを理解する



Q は機械学習を使用して、ビジネスデータ間の意味と関係を自動的に理解し、関連する視覚化を用いて正確な回答を提供します。

ビジネスのために設計



Q は、販売、マーケティング、金融サービス、ヘルスケア、スポーツ分析などのドメインから得たデータで事前にトレーニングされているため、ビジネスの言語と用語を理解できます。

より良い答えを出す



Q は、時間の経過とともに質問に基づいて学習します。作成者は、ダッシュボードを改善するために、読者から寄せられた使用できる質問のうち最も人気のあるものを見ることができます。

情報入手までの時間を短縮



Q は、ビジネスインテリジェンスチームが新しい質問を確認するたびにデータとダッシュボードの更新を待つ必要がなくなるため、データに関するインサイトをすばやく得ることができます。

データ分析基盤は役に立ったか？

› Quick Sight (可視化)

› Dataiku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