JAPAN | 2024

aws SUMMIT

AWS-41

実践! AWS Glue の ETL 開発運用ベストプラクティス

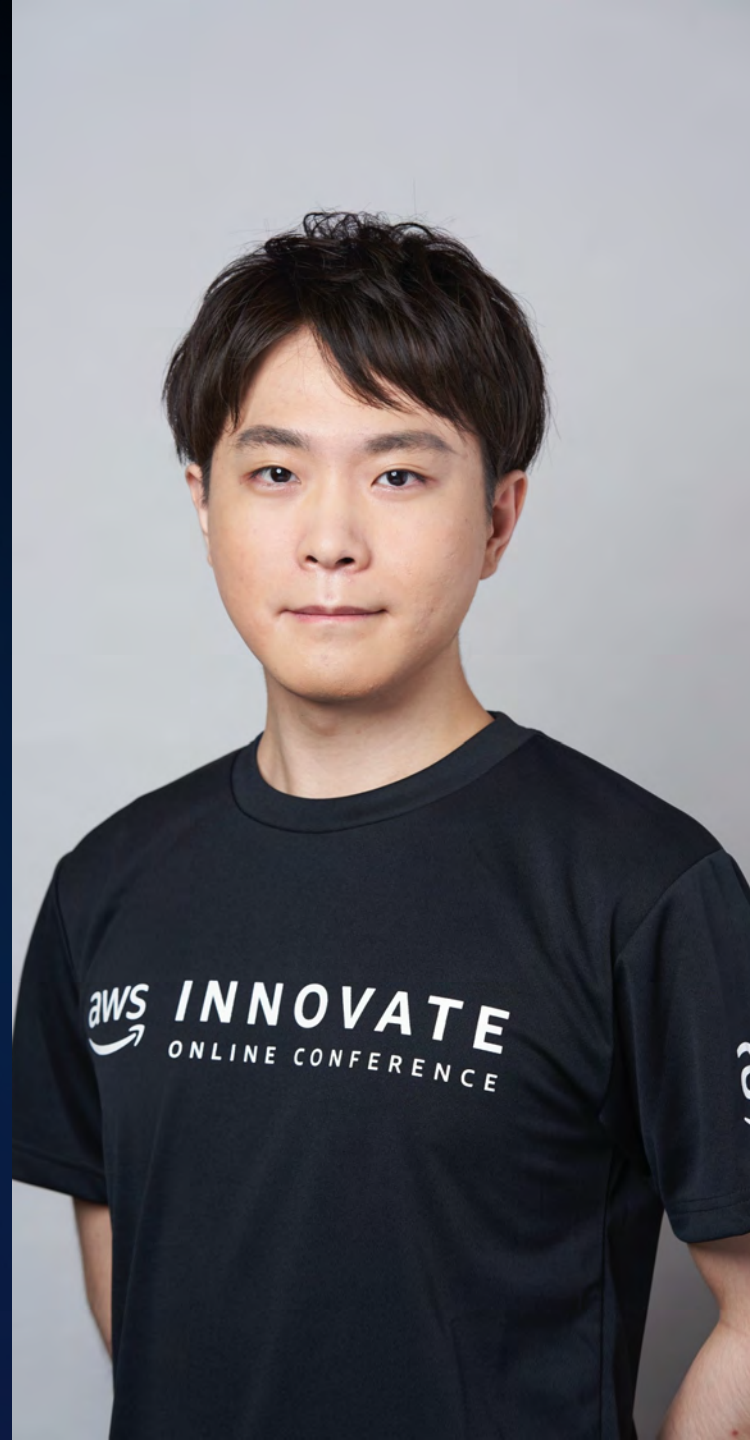
佐藤 祥多
アマゾン ウェブ サービス ジャパン合同会社
サービス&テクノロジー事業統括本部 Data & AI ソリューション本部
アナリティクススペシャリストソリューションアーキテクト

自己紹介

佐藤 祥多（さとう しょうた）
アマゾン ウェブ サービス ジャパン
アナリティクススペシャリストソリューションアーキテクト

サーバーエンジニアをからデータコンサルタントを経て現職
ゲーム業界やインターネットメディア業界のお客様に
データ分析基盤の構築をサポート

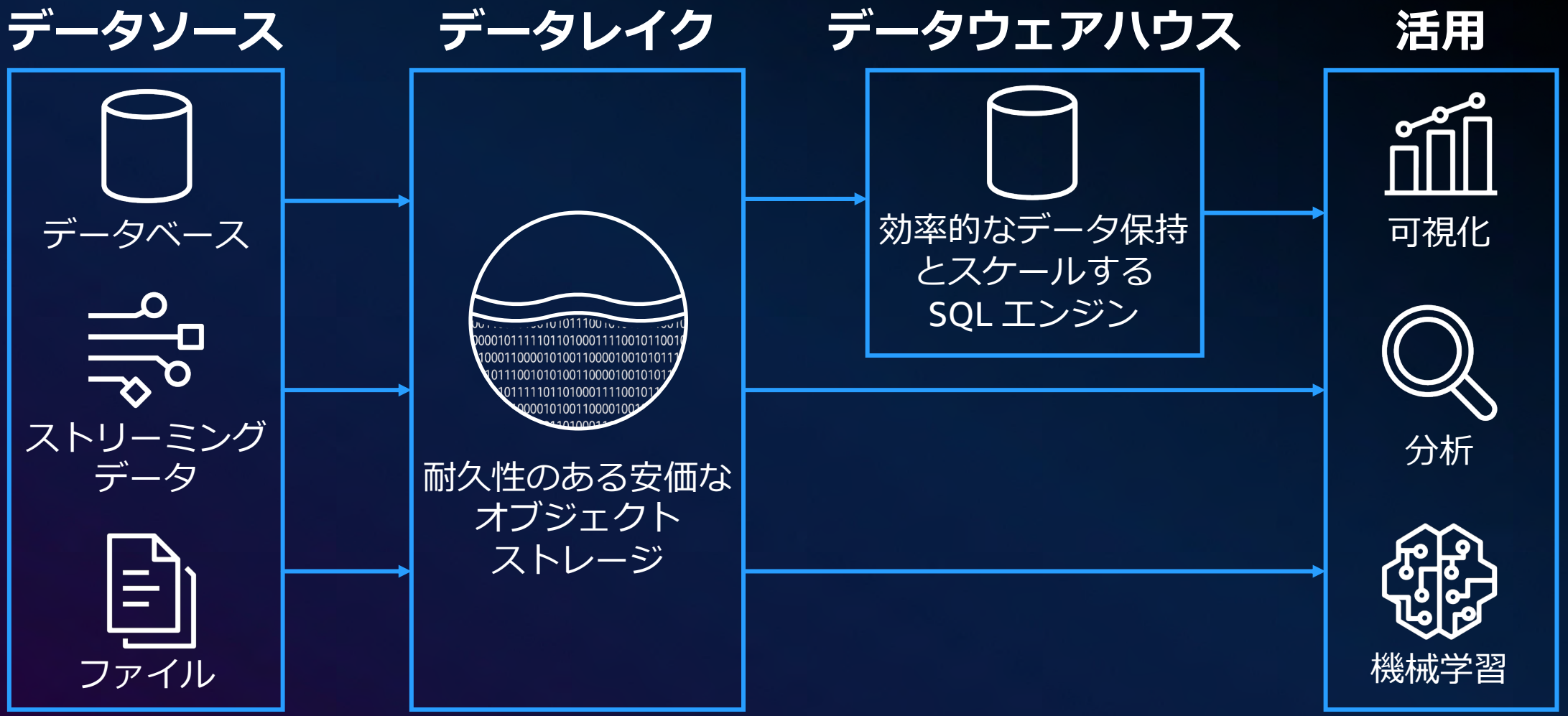
専門：データレイク周辺技術
趣味：紅茶を入れて飲むこと



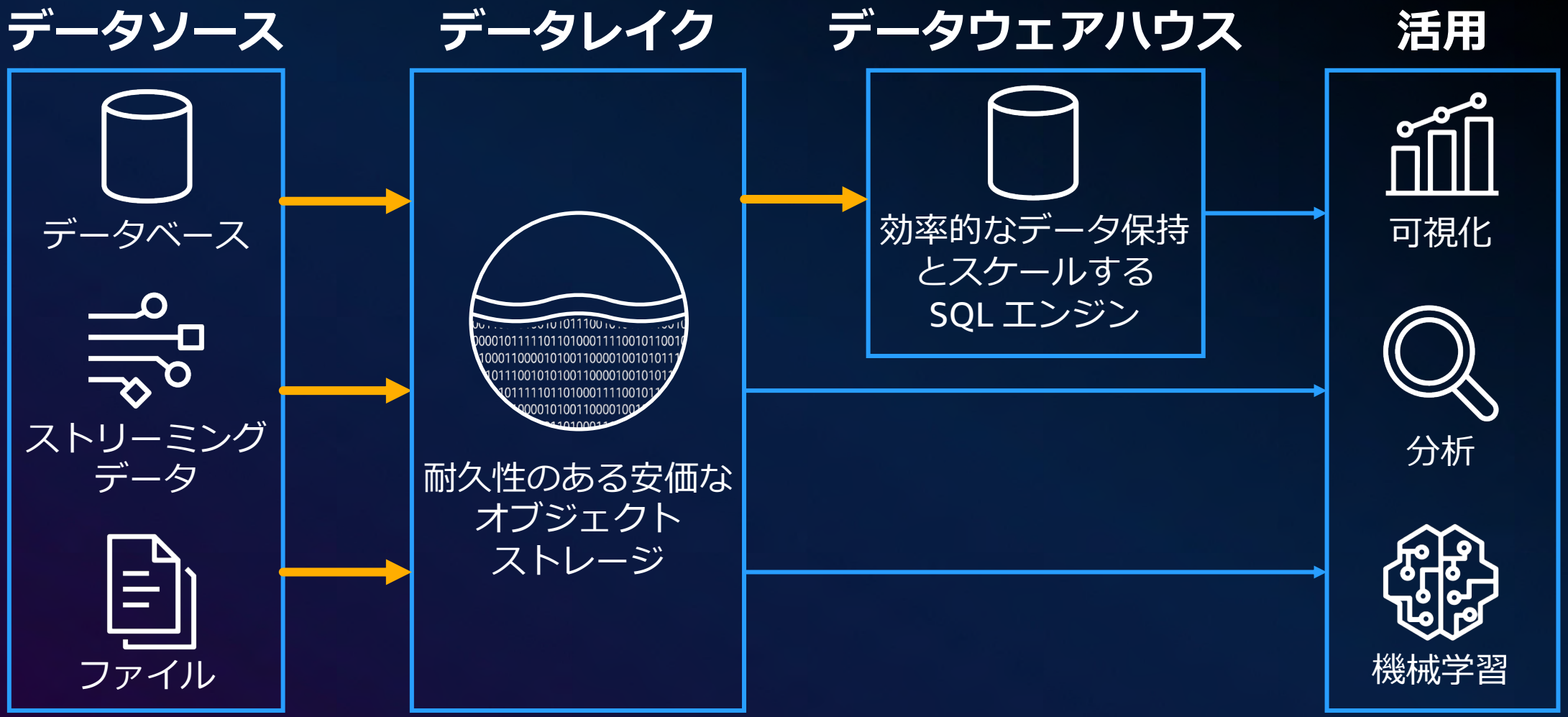
本日ここで話すこと

- サーバーレスデータ統合サービスである AWS Glue の説明とそれが使われる理由
- AWS Glue で ETL ジョブを実際に開発・運用を行っていく際のベストプラクティス

モダンなデータ分析基盤のアーキテクチャ



モダンなデータ分析基盤のアーキテクチャ



様々なデータソースに対応し、データ移動や
活用のための加工を行えるソリューションが必要

AWS Glue

規模を問わずあらゆるデータを発見、準備、統合する

オールインワンの
データ統合サービス

スケーラブル、サーバーレスで
高費用対効果

すべてのデータユーザを
サポートするカスタムツール

1つの場所ですべての
ワークロードをサポート

AWS Glue

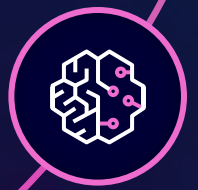
ユースケース

最新のデータ戦略 (データレイク、データメッシュなど) を適用してスケーラブルなデータ分析を実現

高額な従来のETLソリューションから移行して柔軟性を高め、コストを削減

データ資産をカタログ化して、最新のデータアーキテクチャ全体で利用

Apache Spark を使用して、ペタバイト単位のデータをバッチ、ストリーミング、およびリアルタイムで処理



分析と機械学習のためのデータの準備

AWS Glue ジョブの開発と運用時に考慮すること

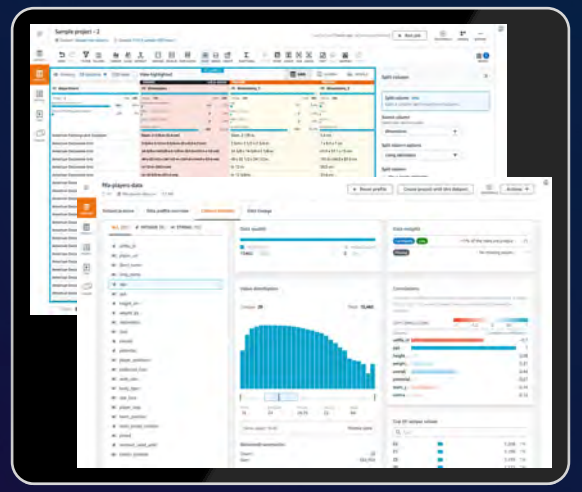
SDLC (ソフトウェア開発ライフサイクル) とは何ですか?

AWS Glue ジョブの開発と運用時に考慮すること

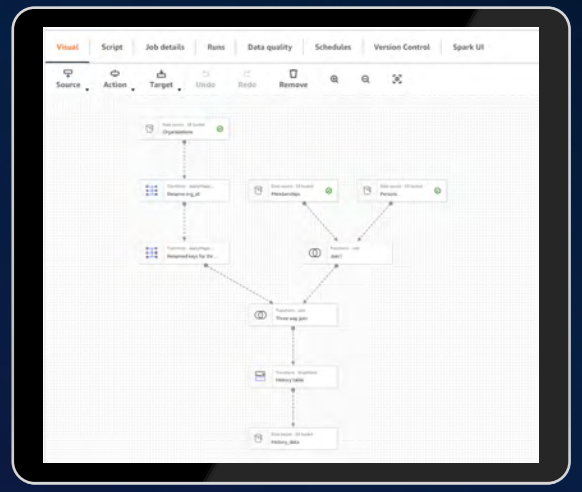
SDLC (ソフトウェア開発ライフサイクル) とは何ですか?

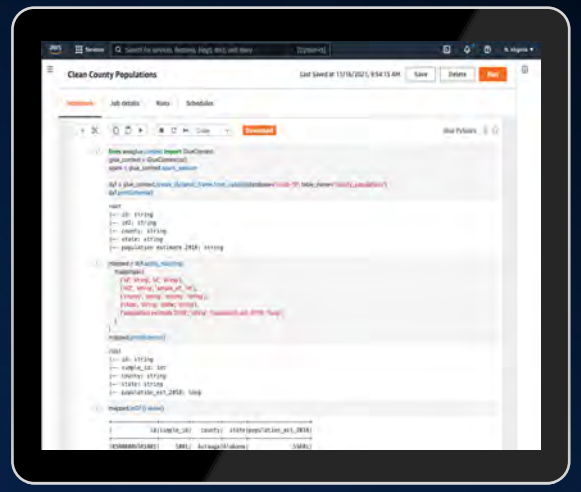
開発運用ベストプラクティス
～開発編～

AWS Glue のジョブ実装を支えるツール

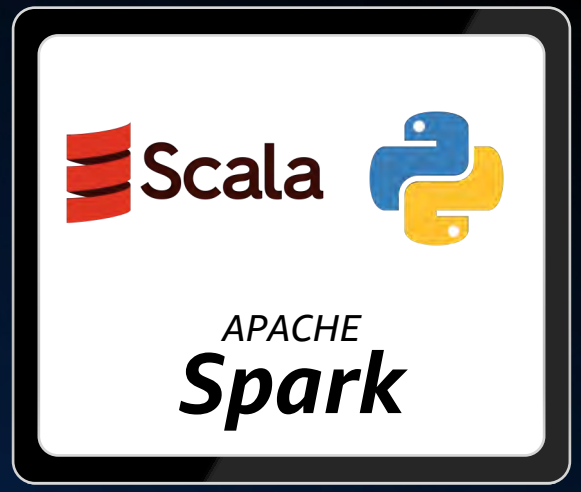


ビジュアル





Notebook



IDE

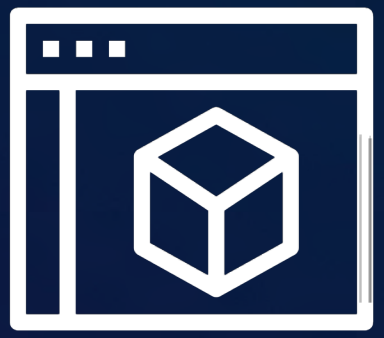
ビジネスアナリスト
データサイエンティスト

ETL エンジニア

データエンジニア
データサイエンティスト

ETL エンジニア
データエンジニア

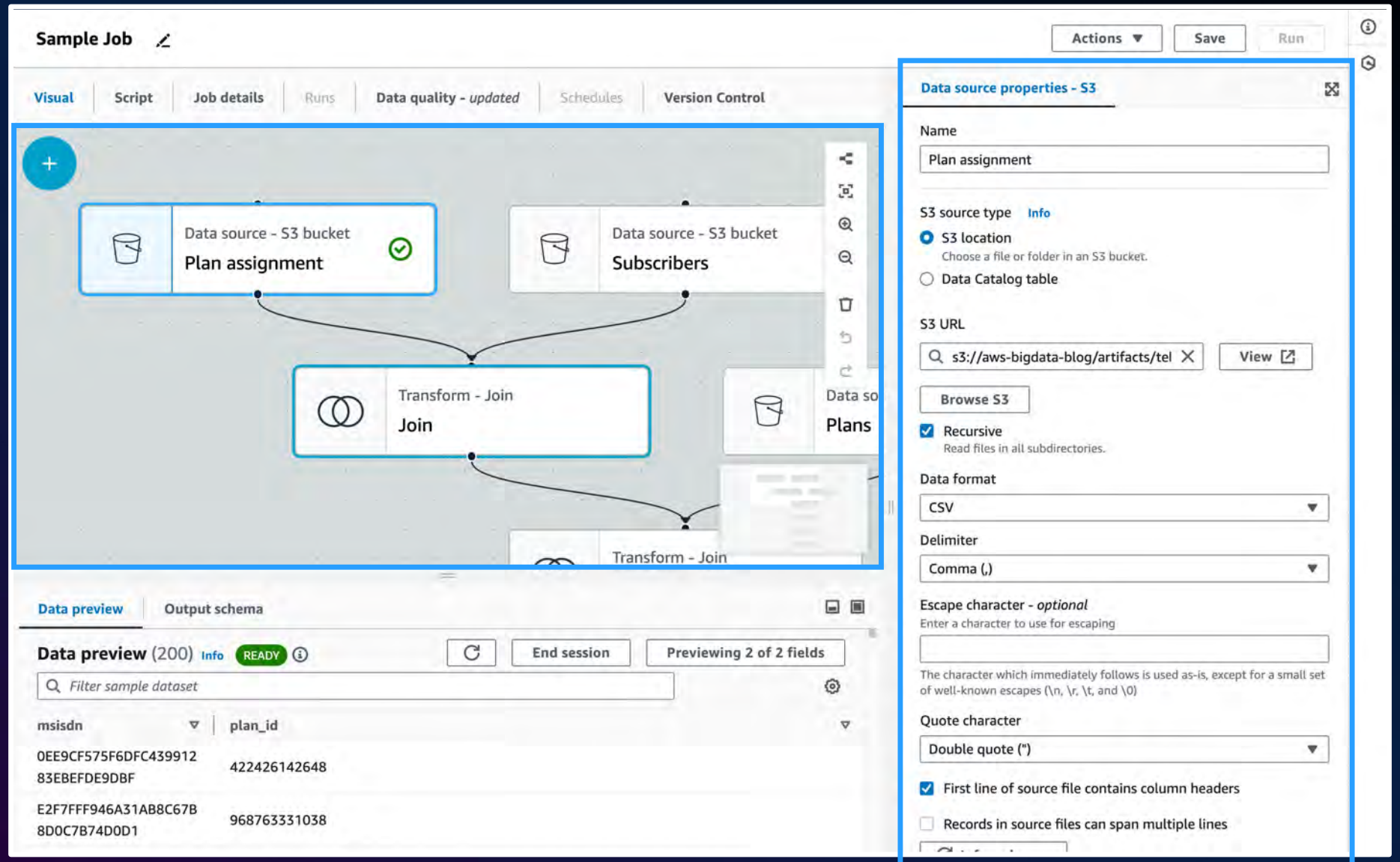
AWS Glue のジョブ実装パターン



ノーコード
ローコード実装

コーディング実装

AWS Glue Studio ビジュアルエディタ



対応している接続と変換

読み込み/書き込み

AWS
- AWS Glue Data Catalog
- Amazon Simple Storage Service (Amazon S3)
- Amazon Kinesis
- Amazon MSK
- Amazon Redshift
- Amazon DynamoDB
- Amazon OpenSearch Service

DB
- MySQL
- PosgtreSQL
- SAP HANA
- Oracle SQL
- Microsoft SQL Server

DWH
- Snowflake
- Google BigQuery
- Teradata Vantage

SaaS
- Salesforce

など

100種類以上

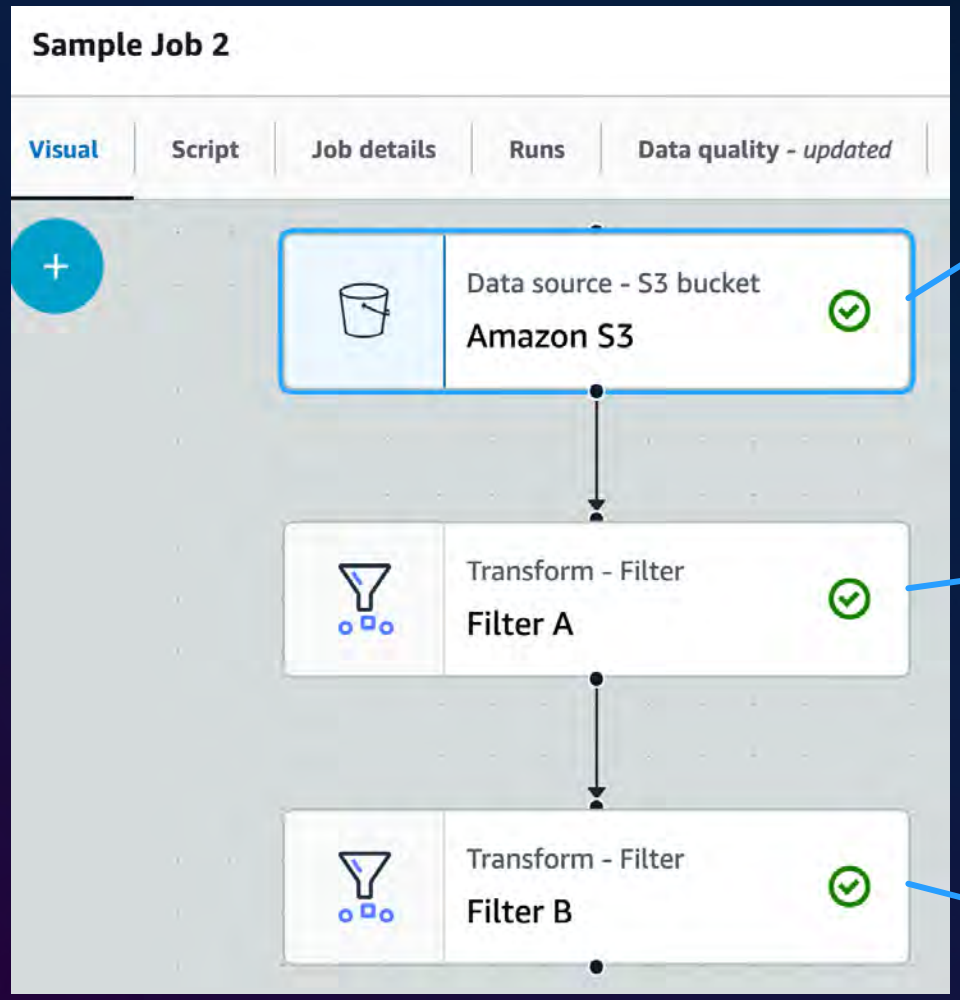
変換

- 結合 (Join)
- 集計 (Aggregation)
- 除外 (Filter)
- 分割 (Split)
- 結合 (Union)
- 機密データ検知 (Detect Sensitive Data)
- 正規表現 (Regex Extractor)
- **Spark SQL**
- **カスタム変換**

など

40 種類

AWS Glue Studio Interactive Data Preview



Data preview (3) Info READY

Filter sample dataset

name	column2	column4
ggg	111	1111
hhh	222	2222
aaa	999	9999

Data preview (2) Info READY

Filter sample dataset

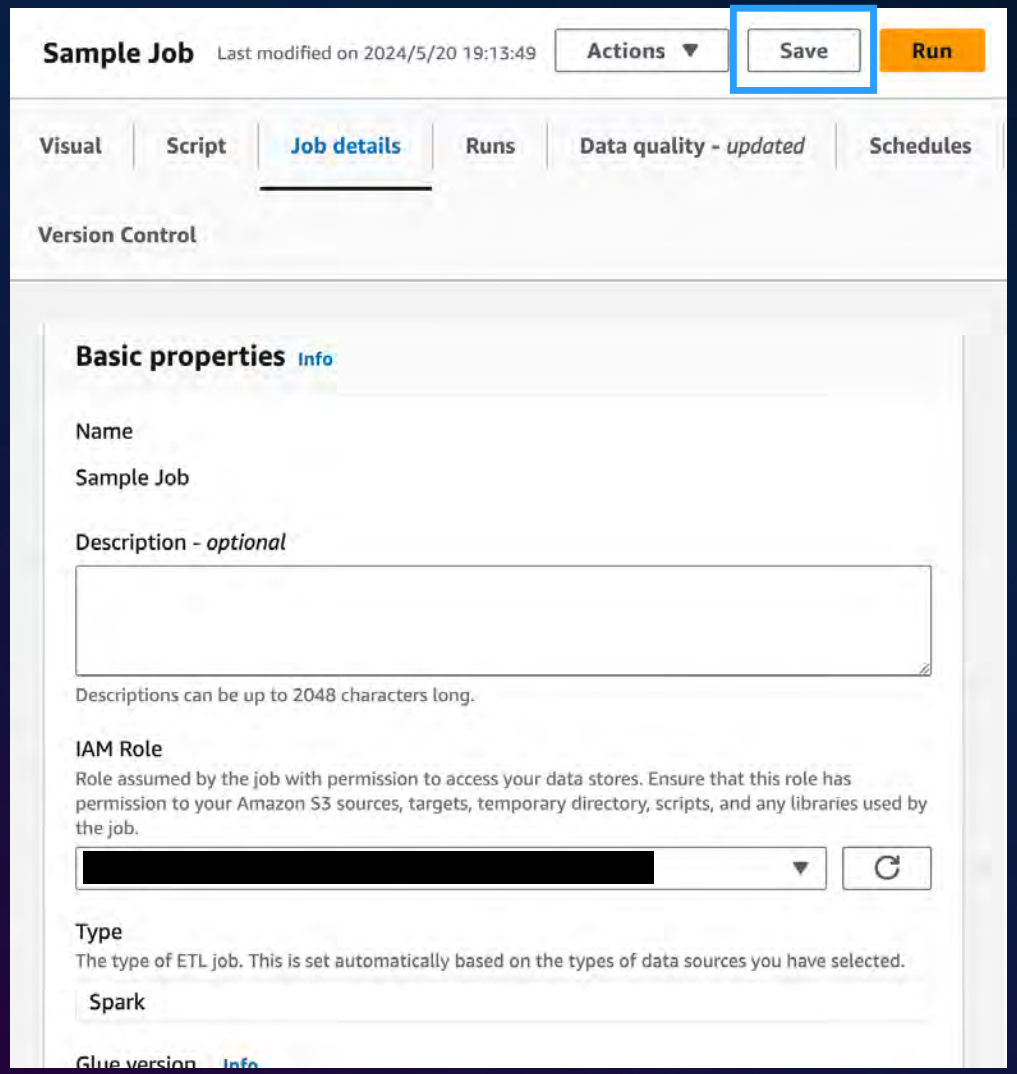
column2	column4	name
111	1111	ggg
999	9999	aaa

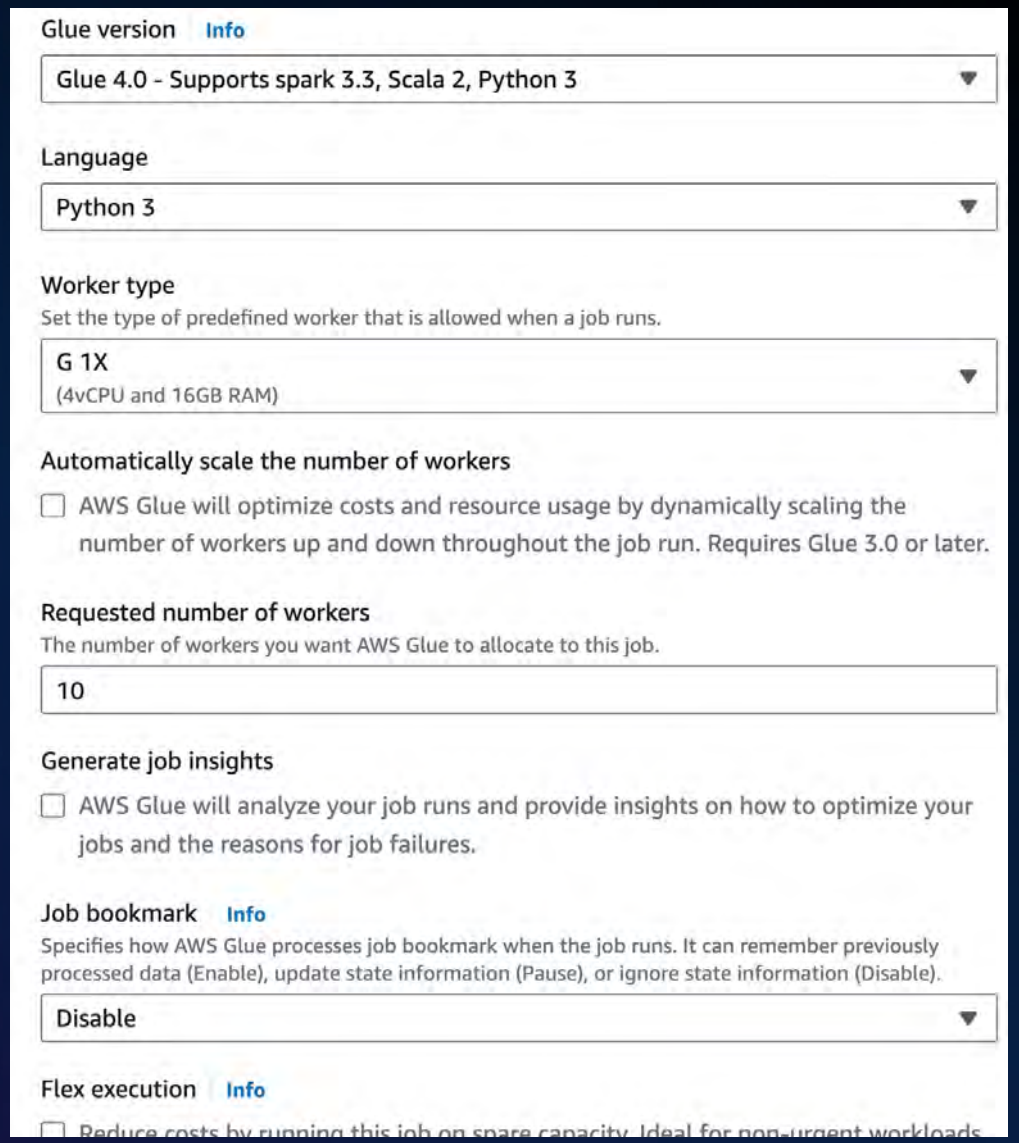
Data preview (1) Info READY

Filter sample dataset

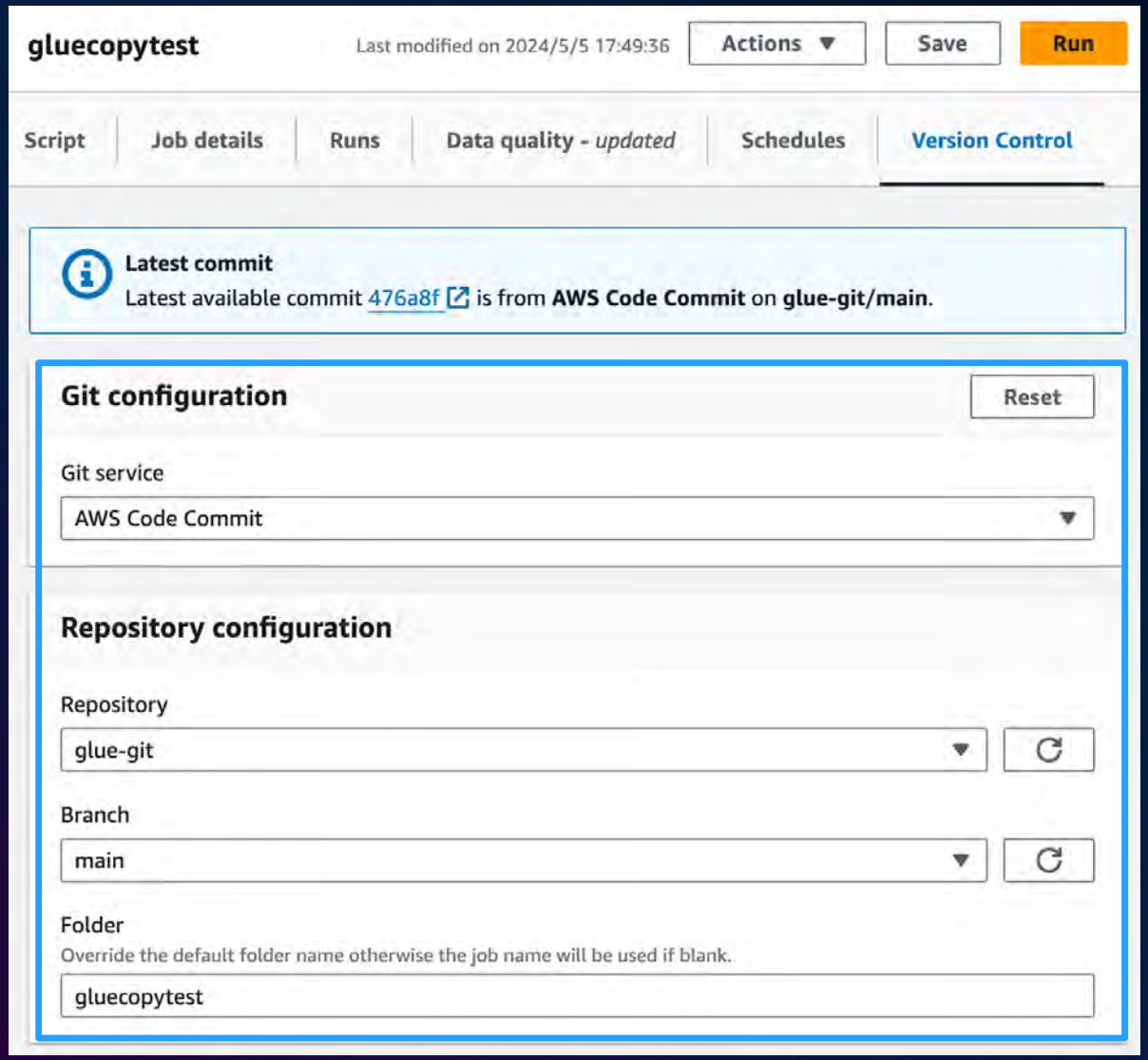
column2	column4	name
999	9999	aaa

AWS Glue Studio ジョブの作成





AWS Glue Studio の Git 連携



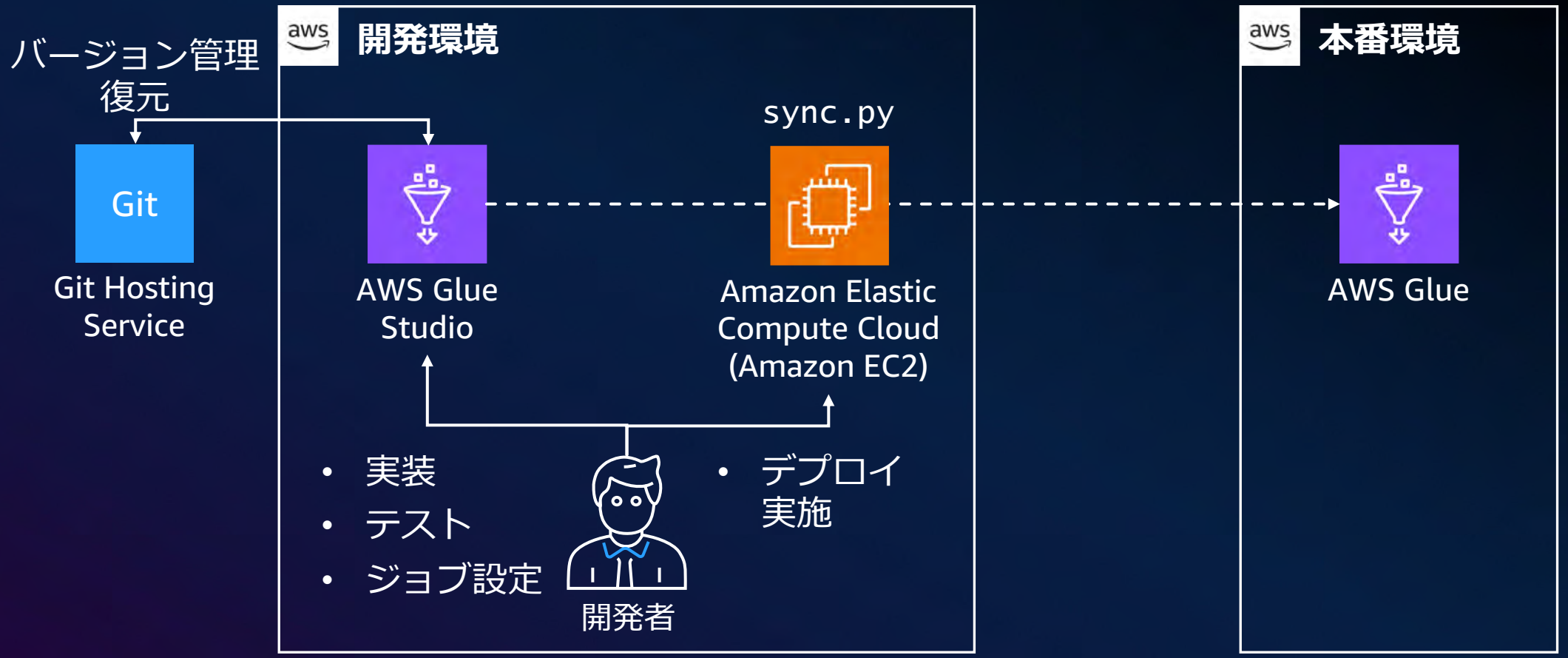
対応サービス

- AWS CodeCommit
- GitHub
- Gitlab
- Bitbucket

ノーコードローコード開発例1

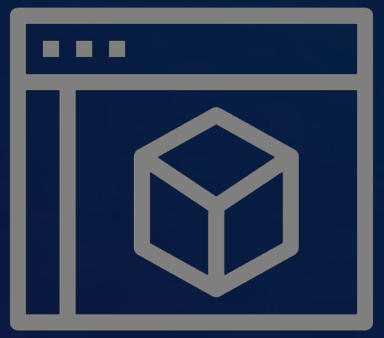
- 実装
- テスト
- ジョブ設定

ノーコードローコード開発例2



AWS Blog: Synchronize your AWS Glue Studio Visual Jobs to different environments

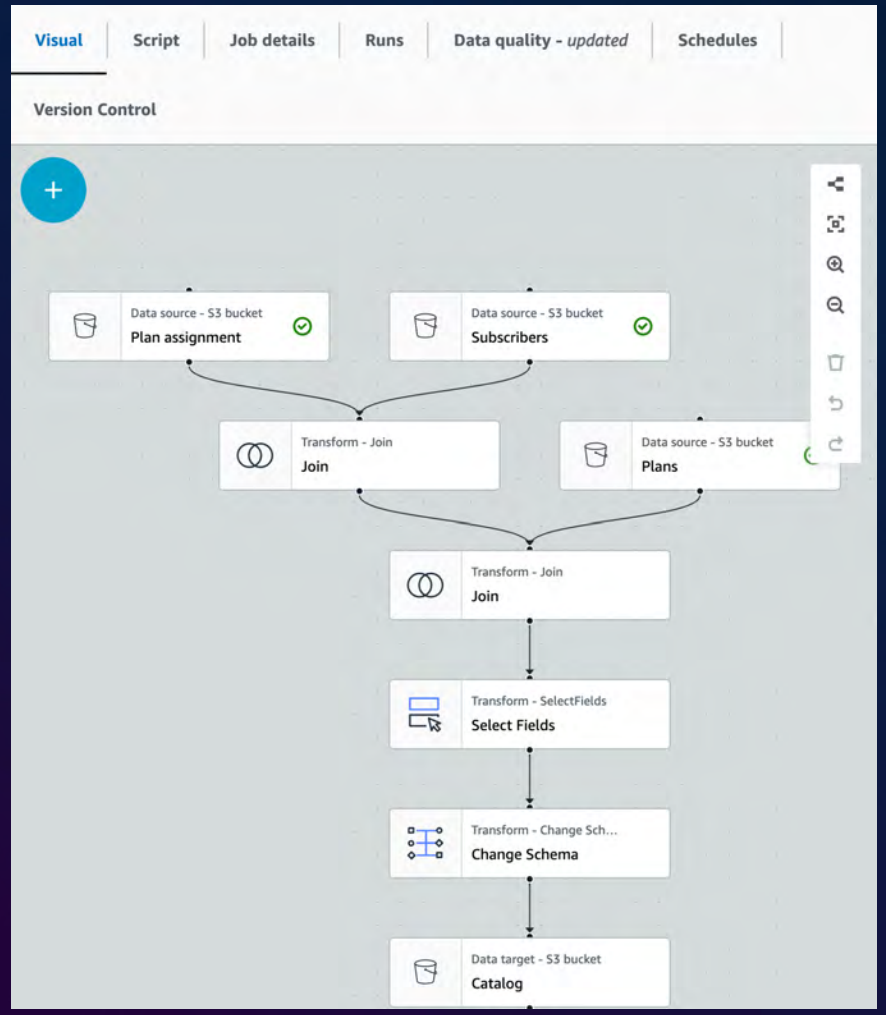
AWS Glue のジョブ実装パターン



ノーコード
ローコード実装

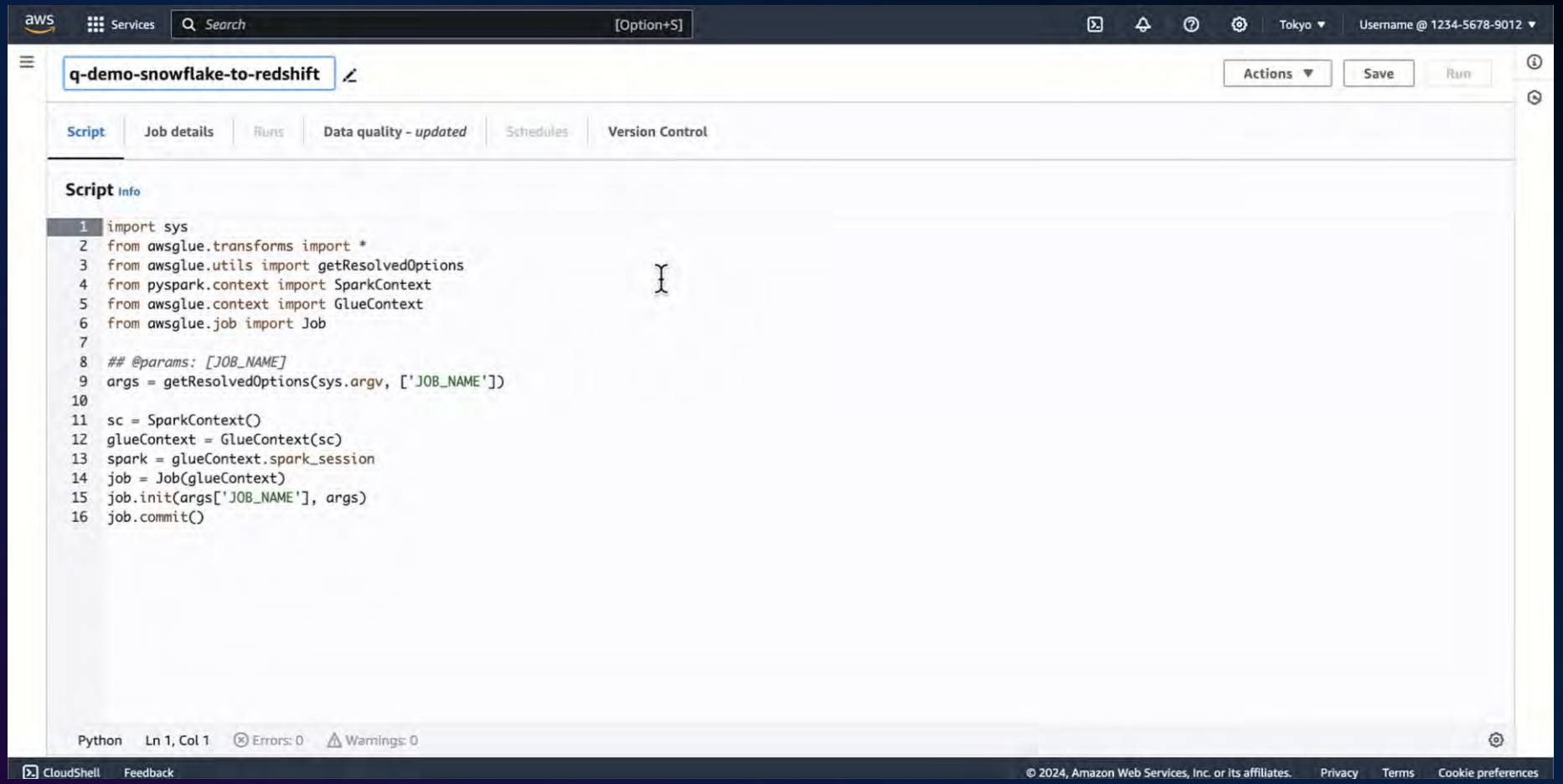
コーディング実装

AWS Glue をコードで実装する補助ツール
AWS Glue Studio でのコード自動生成



自動生成

AWS Glue をコードで実装する補助ツール
Amazon Q data integration in AWS Glue



Please provide a Glue script that reads from Snowflake, rename the fields to Redshift
Snowflake からデータを読んでフィールド名を変えながら Redshift に書き込む

ローカルでのコーディング実装のための環境準備

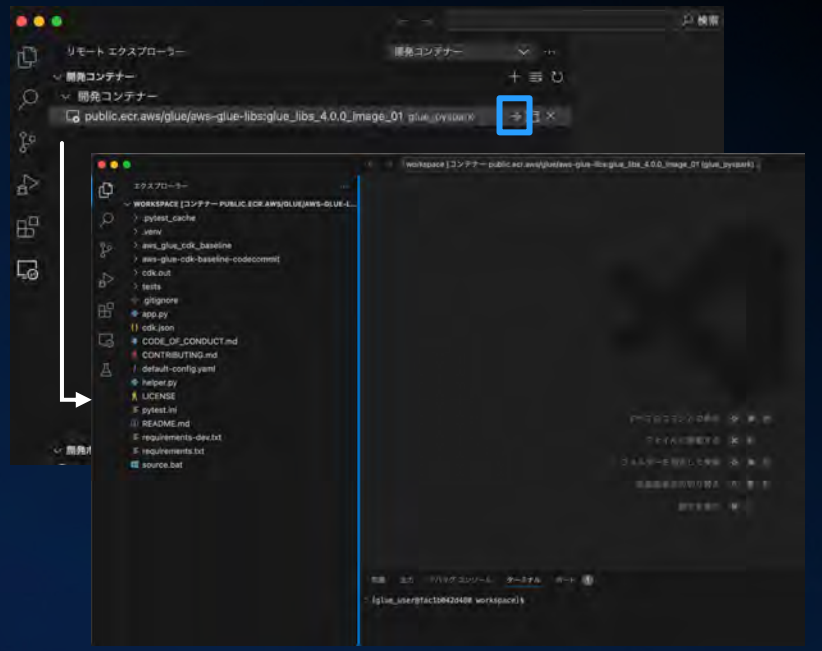
1. Docker image を Pull

```
$ docker pull
public.ecr.aws/glue/aws-
glue-
libs:glue_libs_4.0.0_ima
ge_01
```

2. コンテナを起動

```
$ docker run -it -v
~/.aws:/home/glue_user/.
aws -v <作業領域
>:/home/glue_user/worksp
ace/ -e
AWS_PROFILE=$PROFILE_NAM
E -e DISABLE_SSL=true --
rm -p 4040:4040 -p
18080:18080 --name
glue_pyspark amazon/aws-
glue-
libs:glue_libs_4.0.0_ima
ge_01 pyspark
```

3. IDE から接続



[AWS Blog: Develop and test AWS Glue version 3.0 and 4.0 jobs locally using a Docker container]

ユニットテストの実行

glue_job.py U　test_glue_job.py U ×

aws_glue_cdk_baseline > job_scripts > tests > test_glue_job.py > ir

```python
@pytest.fixture(scope="module", autouse=True)
def glue_context():
    sys.argv.append('--JOB_NAME')
    sys.argv.append('JobName')

    args = getResolvedOptions(sys.argv, ['JOB_NAME'])
    context = GlueContext(SparkContext.getOrCreate())
    job = Job(context)
    job.init(args['JOB_NAME'], args)

    yield(context)

    job.commit()

def test_counts(glue_context):
    dyf = create_dyf(glue_context)
    dyf = glue_job.apply_transform(dyf)
    assert dyf.toDF().count() == 11

def create_dyf(glue_context):
    spark_session = glue_context.spark_session
    df = spark_session.createDataFrame(
        [
            ("TestMovie1", 1.0, 3),
            ("TestMovie2", 2.0, 1),
```

問題　出力　デバッグ コンソール　ターミナル　ポート 1

```
[glue_user@fac1b042d480 job_scripts]$ python3 -m pytest
============================================================================
============================
platform linux -- Python 3.10.2, pytest-7.2.1, pluggy-1.3.0
rootdir: /home/glue_user/workspace, configfile: pytest.ini
plugins: anyio-4.0.0
collected 1 item

tests/test_glue_job.py .                                          [100%]
```

[AWS Blog: End-to-end development lifecycle for data engineers to build a data integration pipeline using AWS Glue]

アセットの自動デプロイ

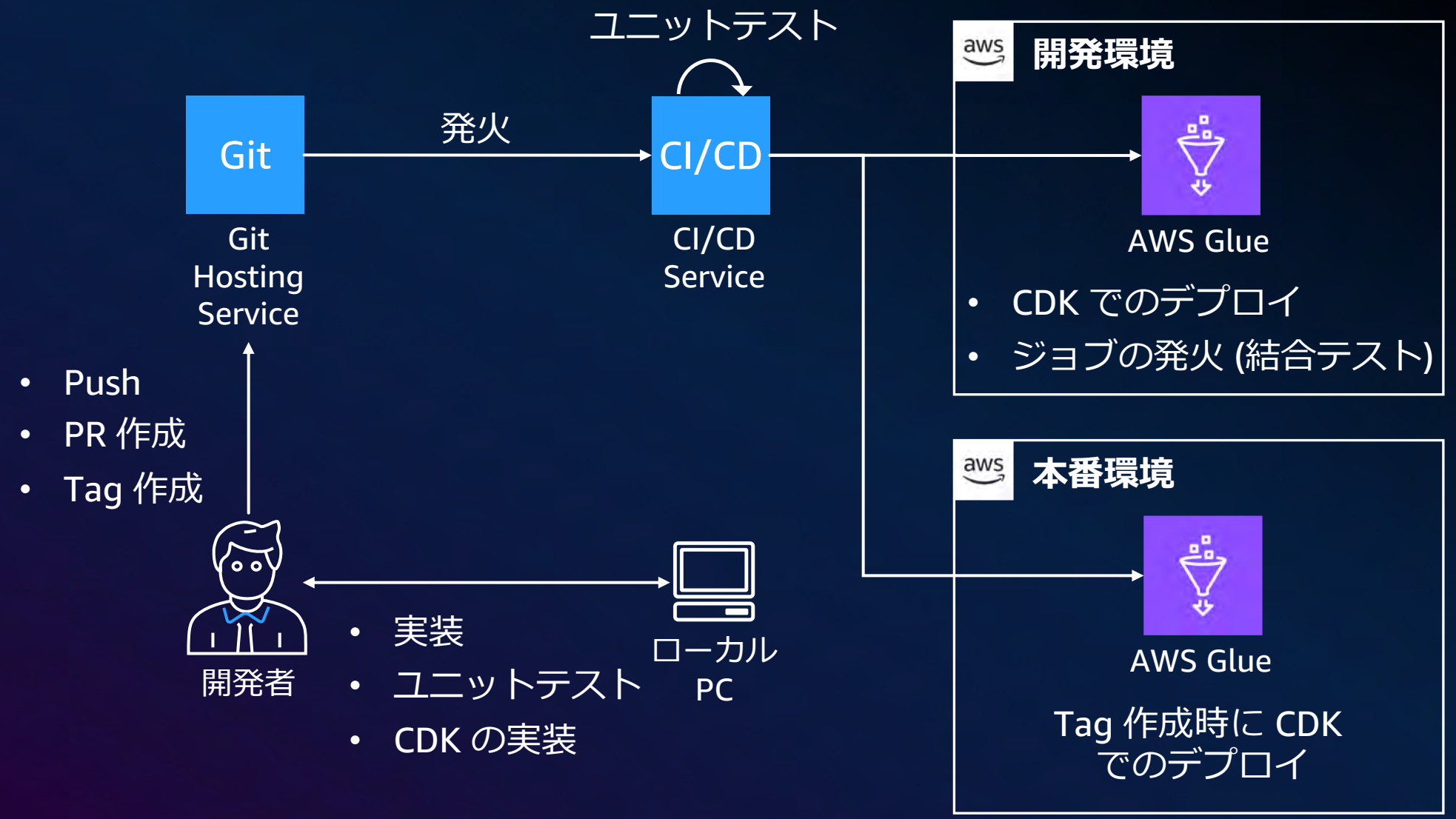
- API 系
  - AWS SDK
  - AWS CLI

など

- IaC 系
  - AWS CloudFormation
  - AWS CDK (Cloud Developer Kit)
  - Terraform

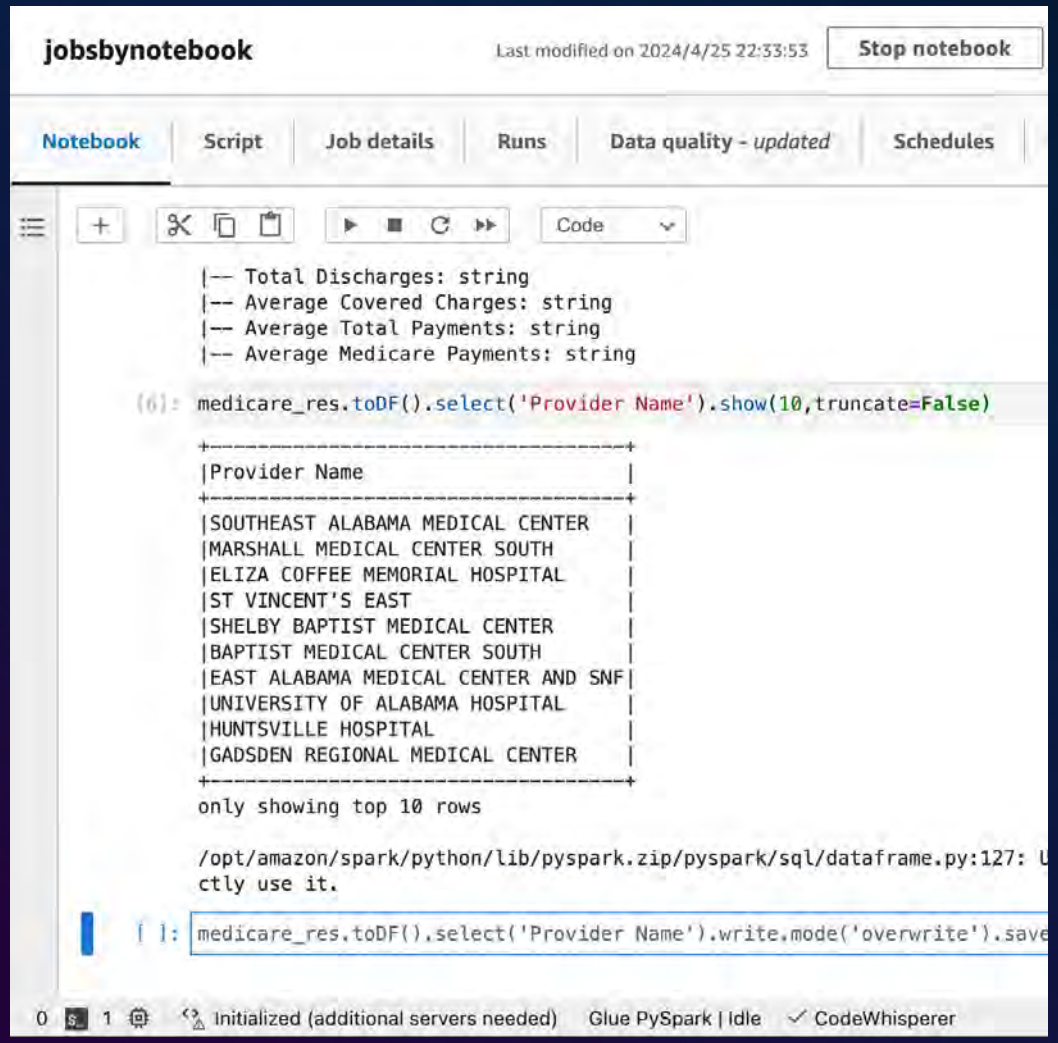
など

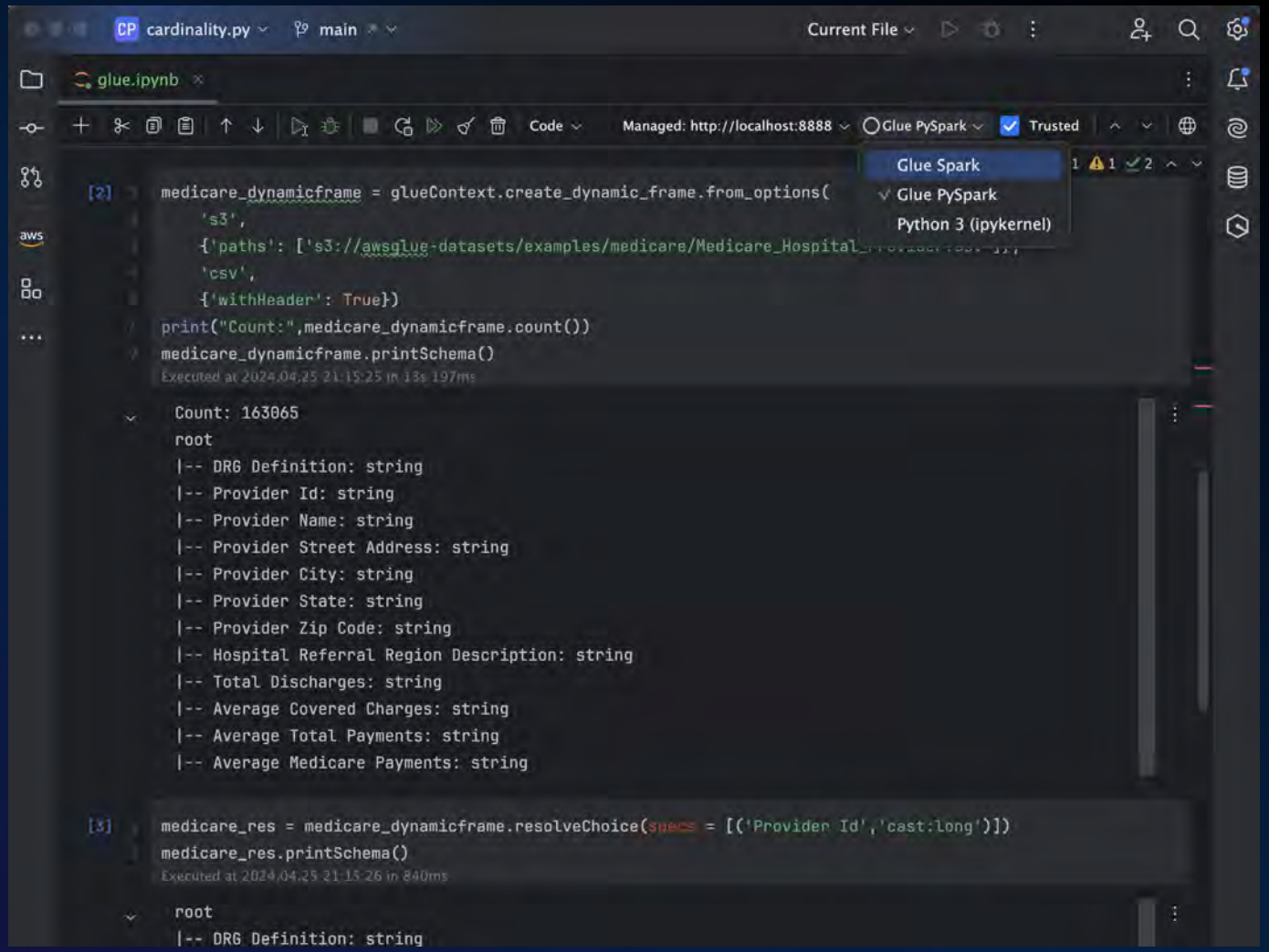
ローカル環境でのコーディング開発例



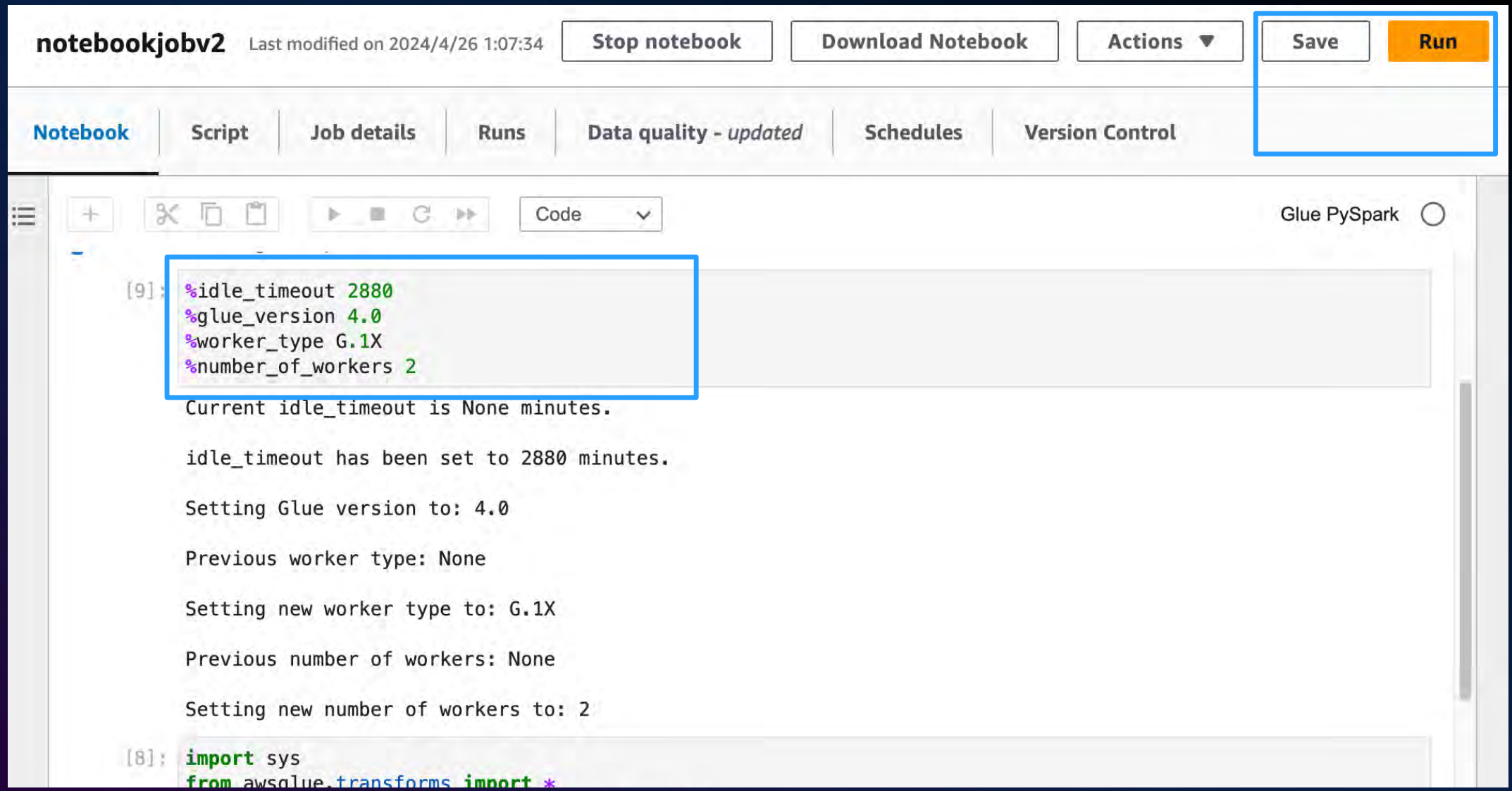
AWS Blog: End-to-end development lifecycle for data engineers to build a data integration pipeline using AWS Glue

補足：Jupyter Notebook





補足：Jupyter Notebook のジョブ実行

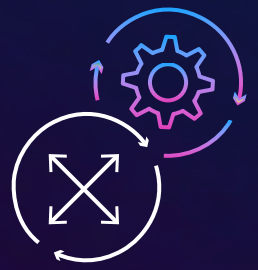
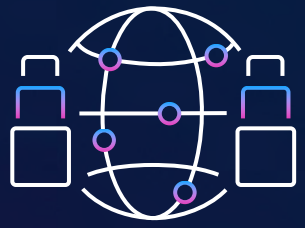


補足：AWS Glue のジョブ実行管理の実装

	AWS Glue workflows	AWS Step Functions	Amazon Managed Workflows for Airflow (MWAA)
			
概要	Glue のジョブや クローラーの ジョブフローを管理	AWS サービスと連携可能な フルマネージド ワークフローエンジン	マネージドな Apache Airflow
強み	シンプル	AWS の各サービス 実行が簡単にできる	高機能で Apache Airflow の 知見を生かせる/貯められる
いつ使うか	**シンプルがよい コスト掛けずに実装**	**AWS サービスと連携 サーバーレスが良い**	**Apache Airflow を 使いたいとき**

開発運用ベストプラクティス
～運用編～

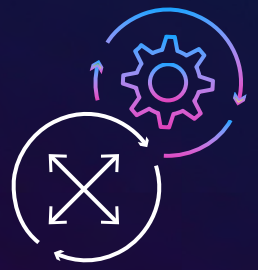
AWS Glue ジョブの運用時に考慮すること

運用上の優秀性	セキュリティ	信頼性	パフォーマンス効率	コスト最適化	持続可能性
					

AWS Documentation: AWS Well-Architected Framework Data Analytics Lens

AWS Glue ジョブの運用時に考慮すること

運用上の優秀性	セキュリティ	信頼性	パフォーマンス効率	コスト最適化	持続可能性
1. ワークロードの健全性を監視	4. データアクセスコントロールを実装				

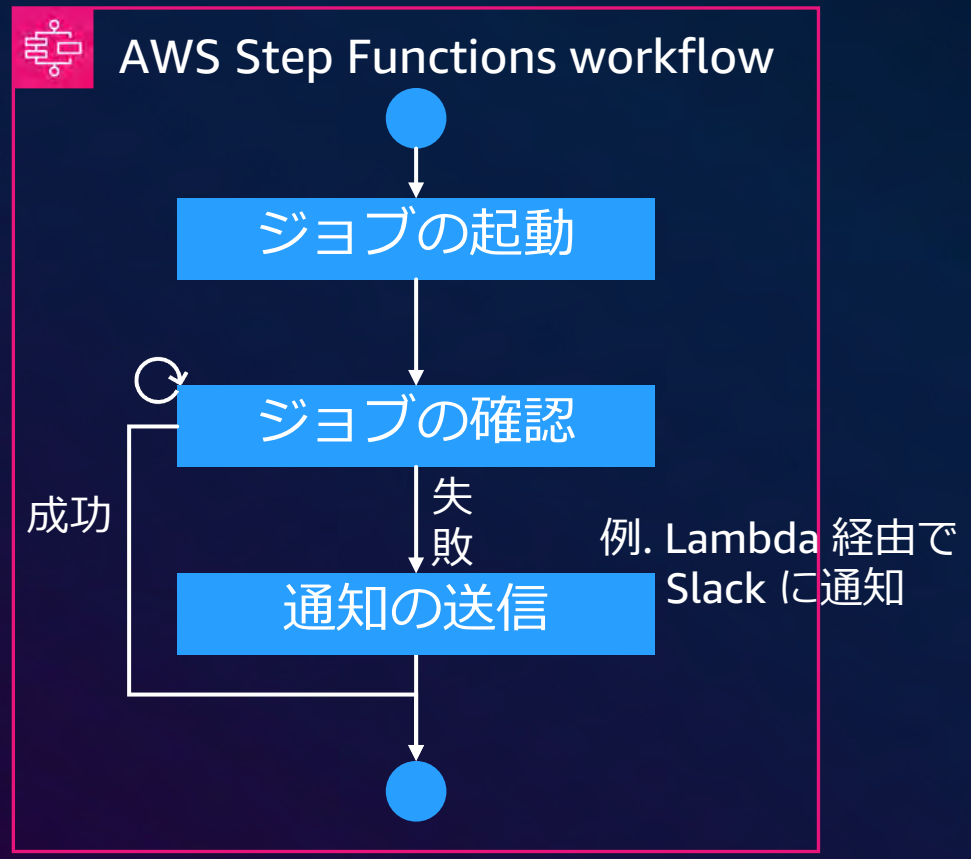


[AWS Documentation: AWS Well-Architected Framework Data Analytics Lens]

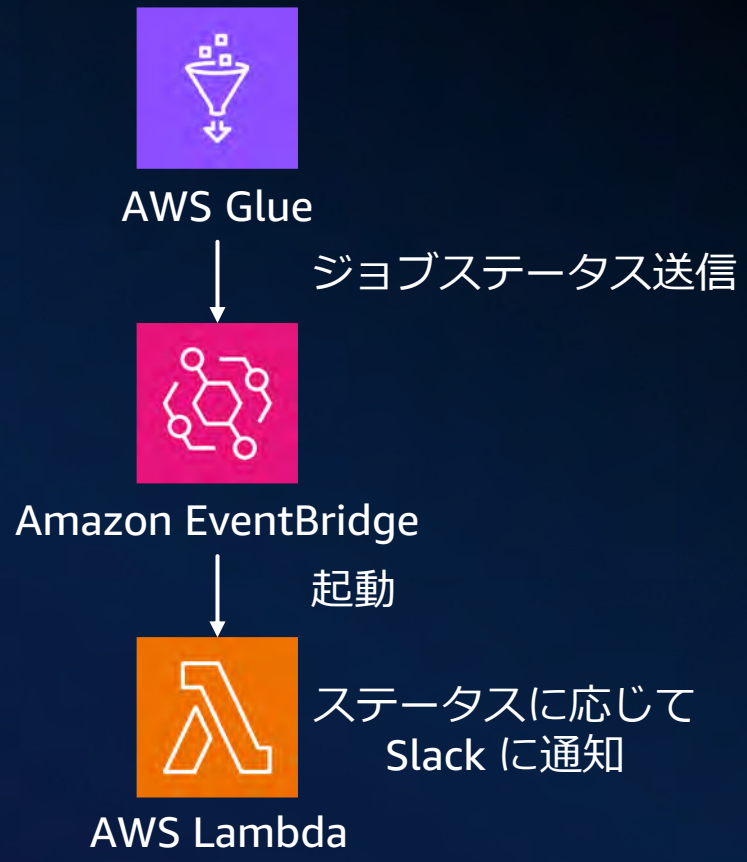
ジョブ実行エラーの検知方法

運用上の優秀性
1. ワークロードの健全性を監視

A. ワークフローエンジンを使う

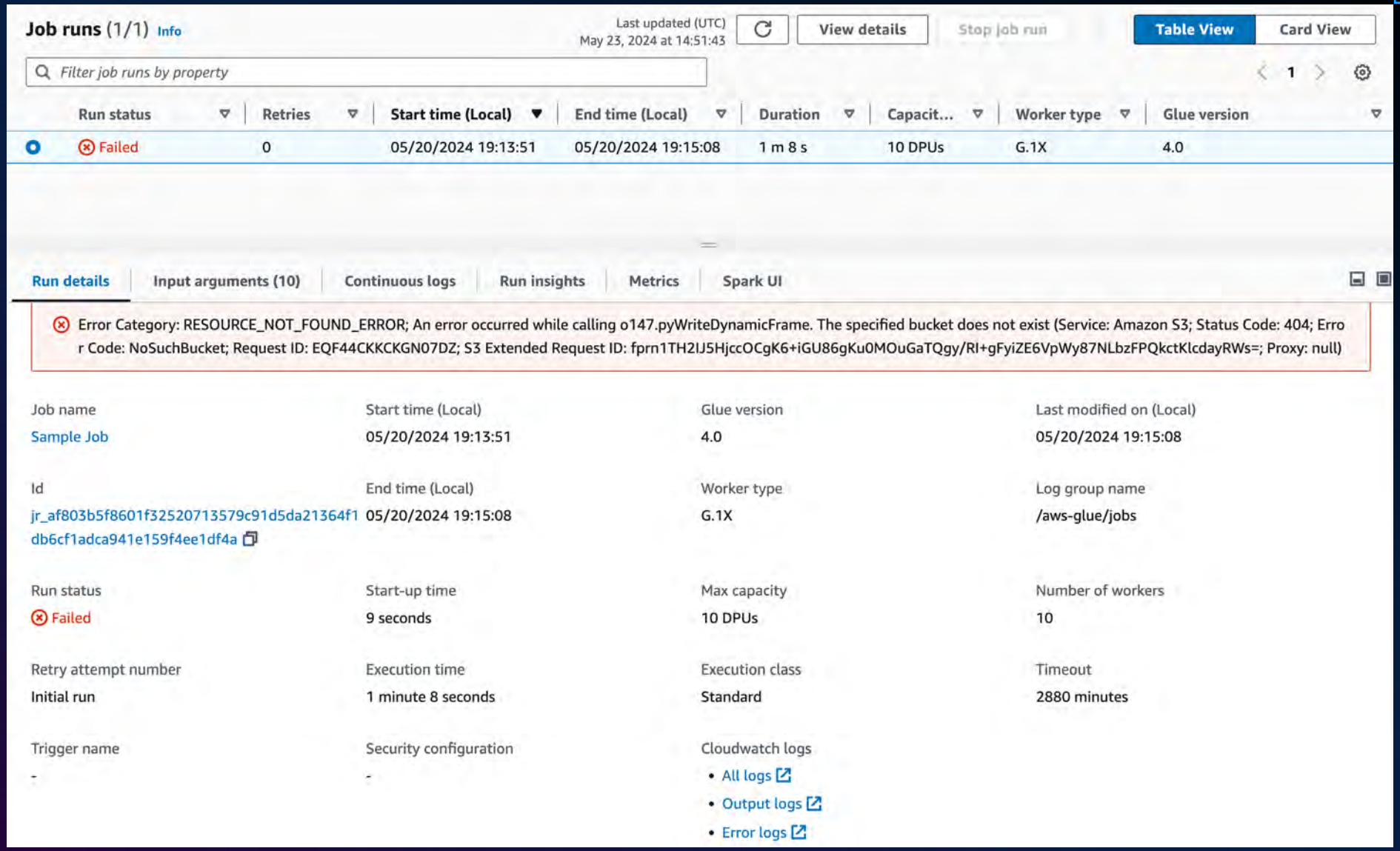


B. EventBridge を使う



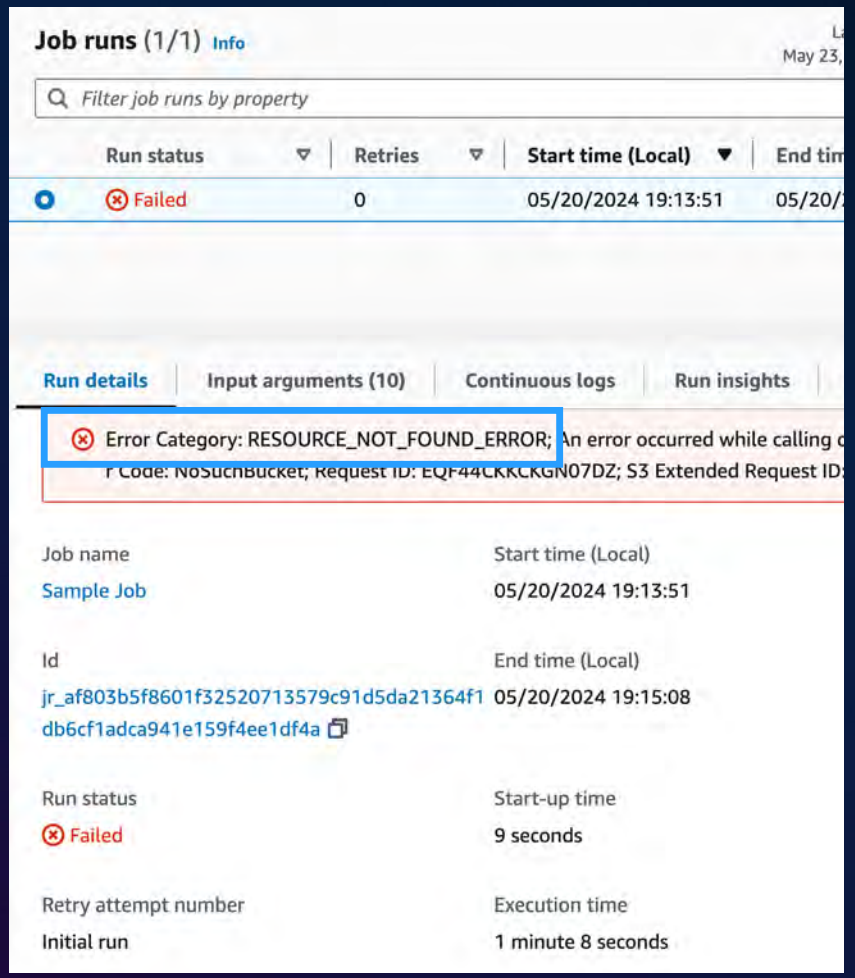
ジョブ実行エラーの調査方法

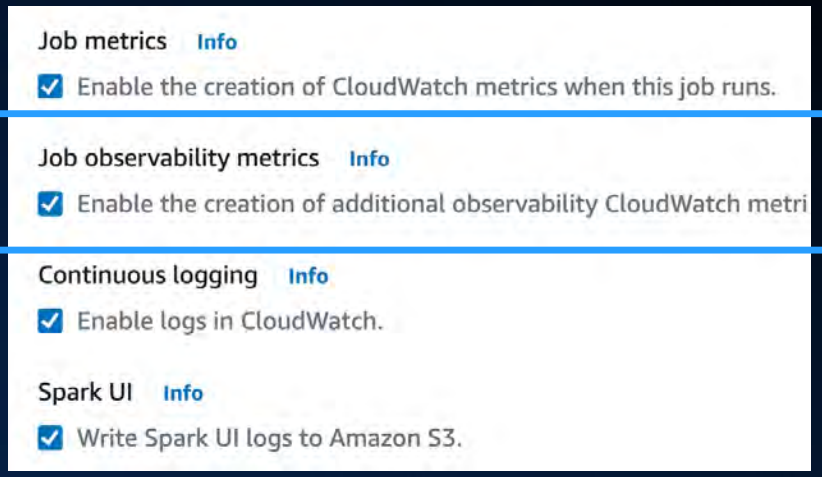
運用上の優秀性
1. ワークロードの健全性を監視



Observability metrics

運用上の優秀性
1. ワークロードの健全性を監視





Glue 4.0
デフォルトで ON

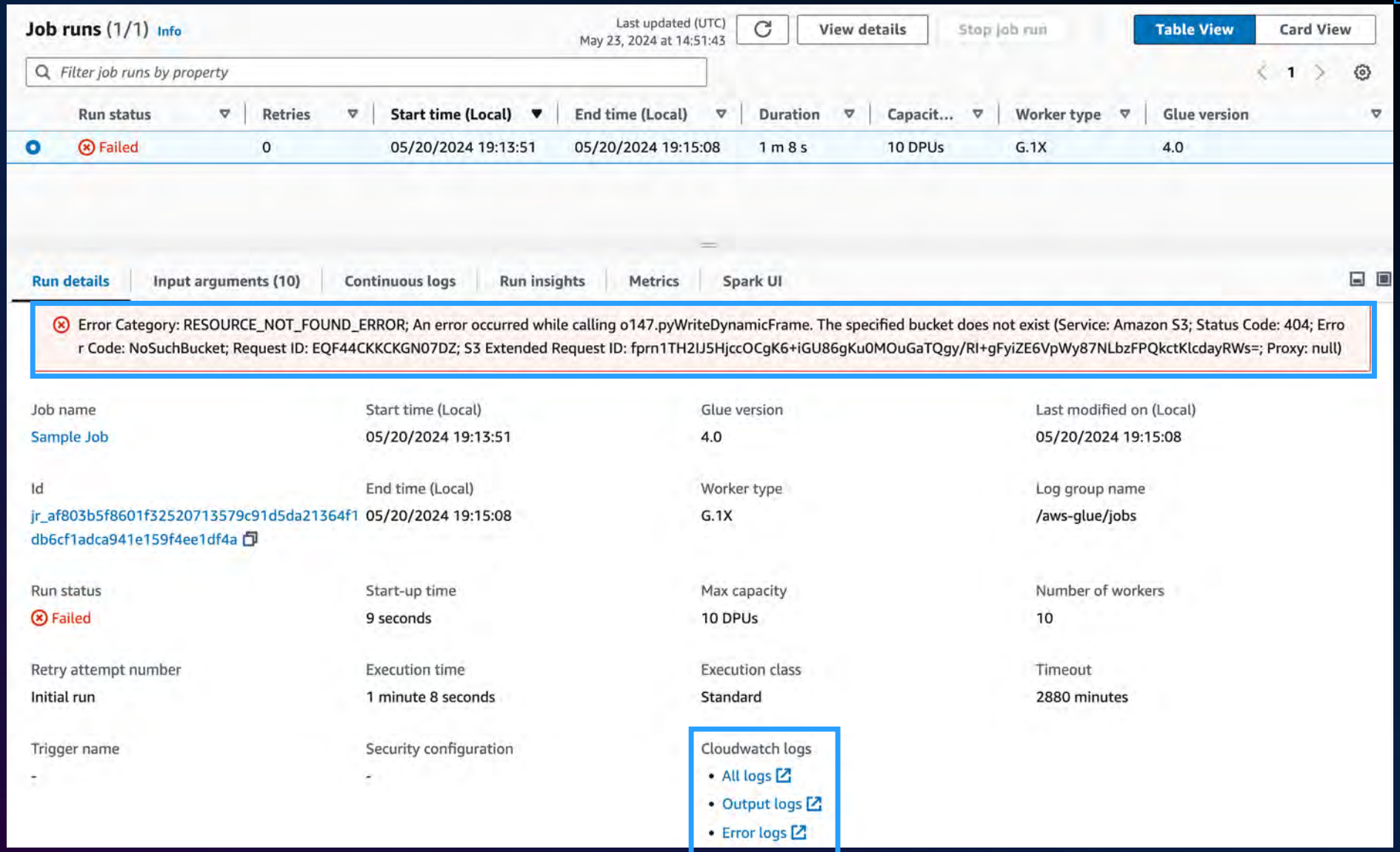
- CONNECTION_ERROR
- DISK_NO_SPACE_ERROR
- OUT_OF_MEMORY_ERROR
- INVALID_ARGUMENT_ERROR
- RESOURCE_NOT_FOUND_ERROR

など

[AWS Documentation: Monitoring with AWS Glue Observability metrics]

エラーメッセージ

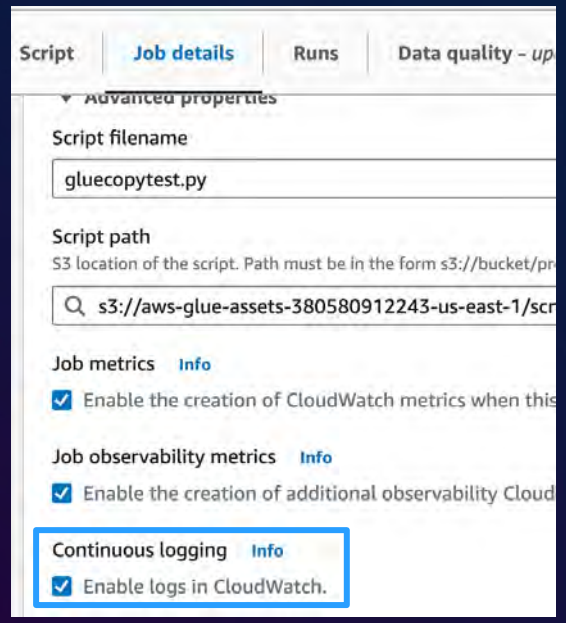
運用上の優秀性
1. ワークロードの健全性を監視



補足：アプリケーションロギングの例

運用上の優秀性
1. ワークロードの健全性を監視

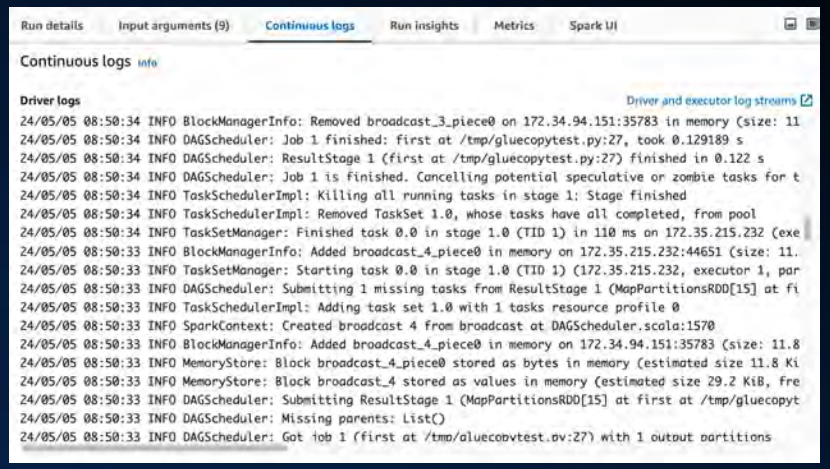
1. Continuous Logging を有効化



2. Logger を GlueContext から取得してロギング

```python
# Python
from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)
logger = glueContext.get_logger()
logger.info("info message")
logger.warn("warn message")
logger.error("error message")
```

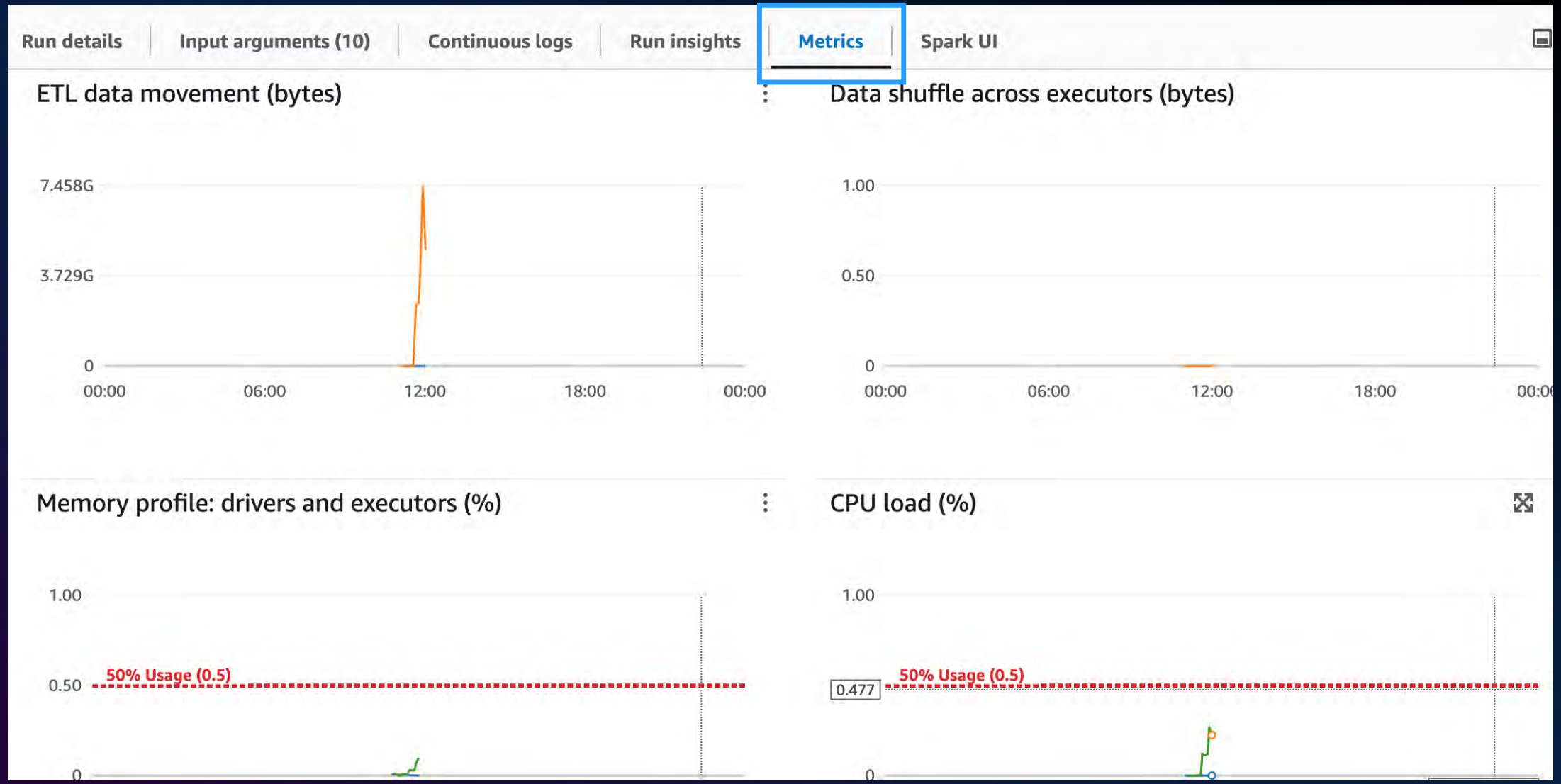


リアルタイムに Glue のログを Glue Studio のコンソールから確認が可能

[AWS Documentation: Enabling continuous logging for AWS Glue jobs]

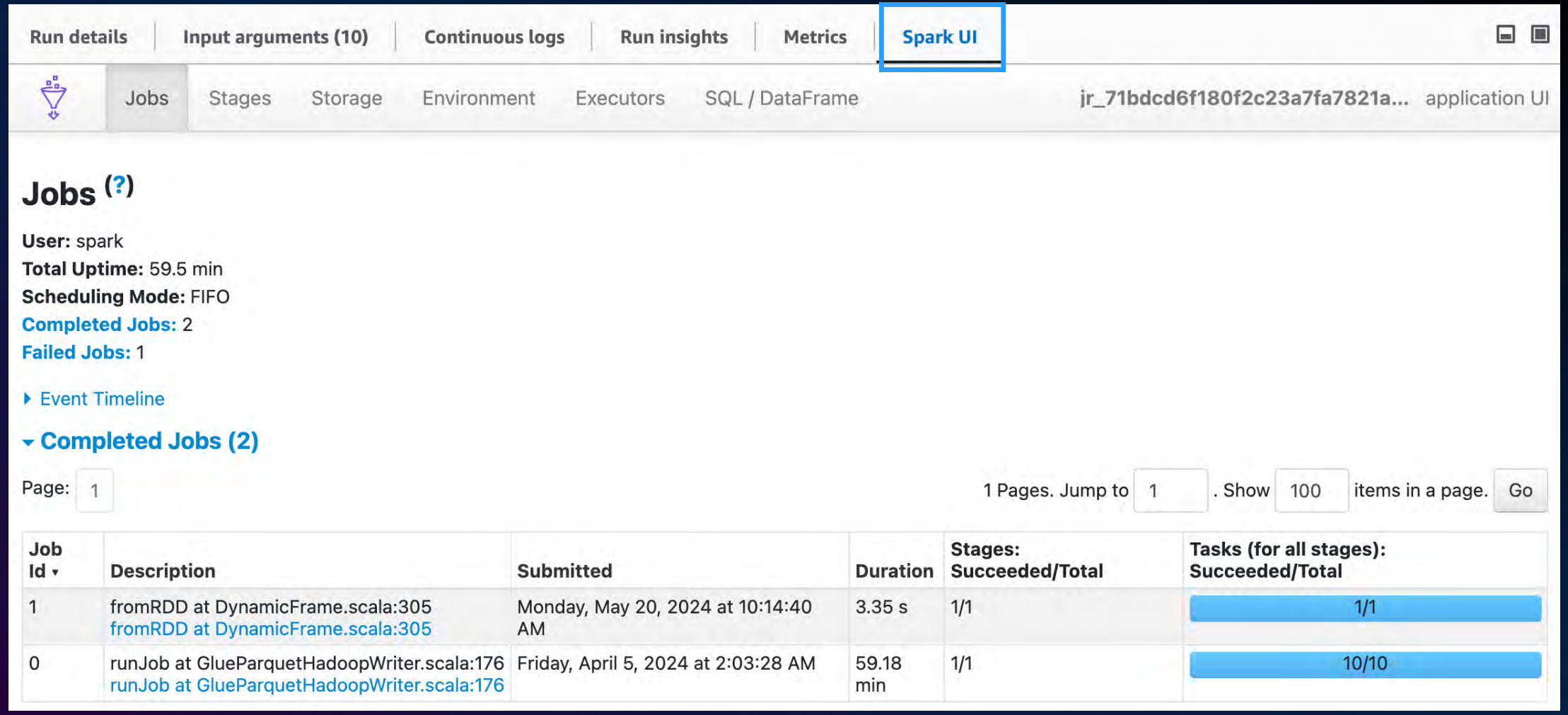
Metrics

運用上の優秀性
1. ワークロードの健全性を監視



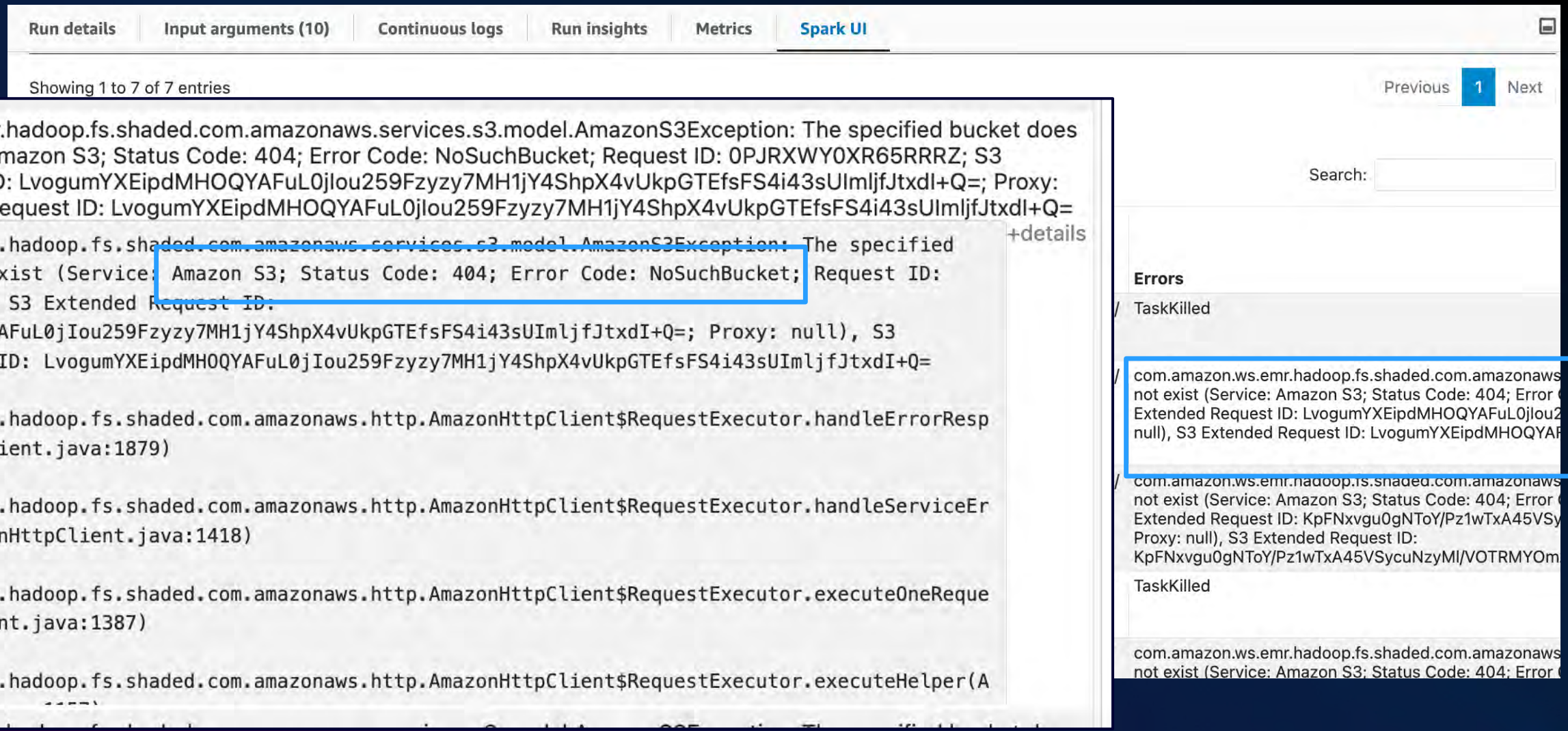
Serverless Spark UI

運用上の優秀性
1. ワークロードの健全性を監視



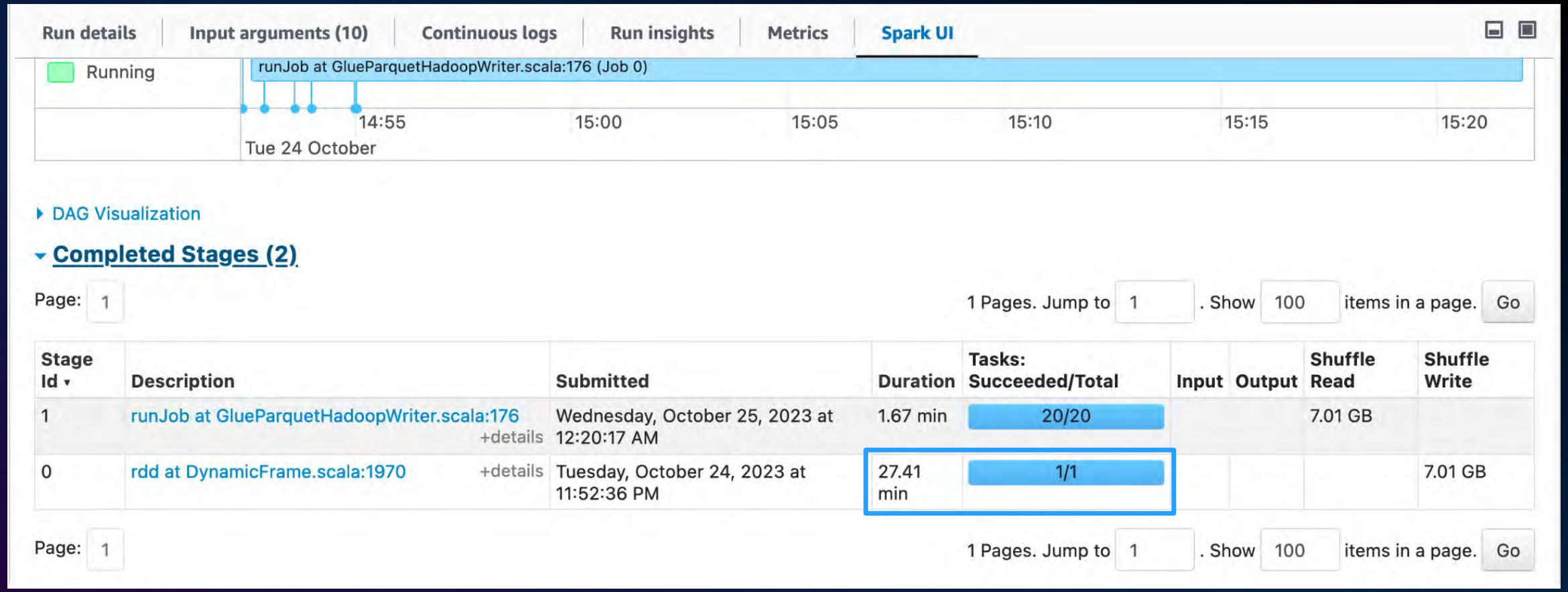
Spark UI によるエラー確認の例

運用上の優秀性
1. ワークロードの健全性を監視



補足：Spark UI によるボトルネックの特定

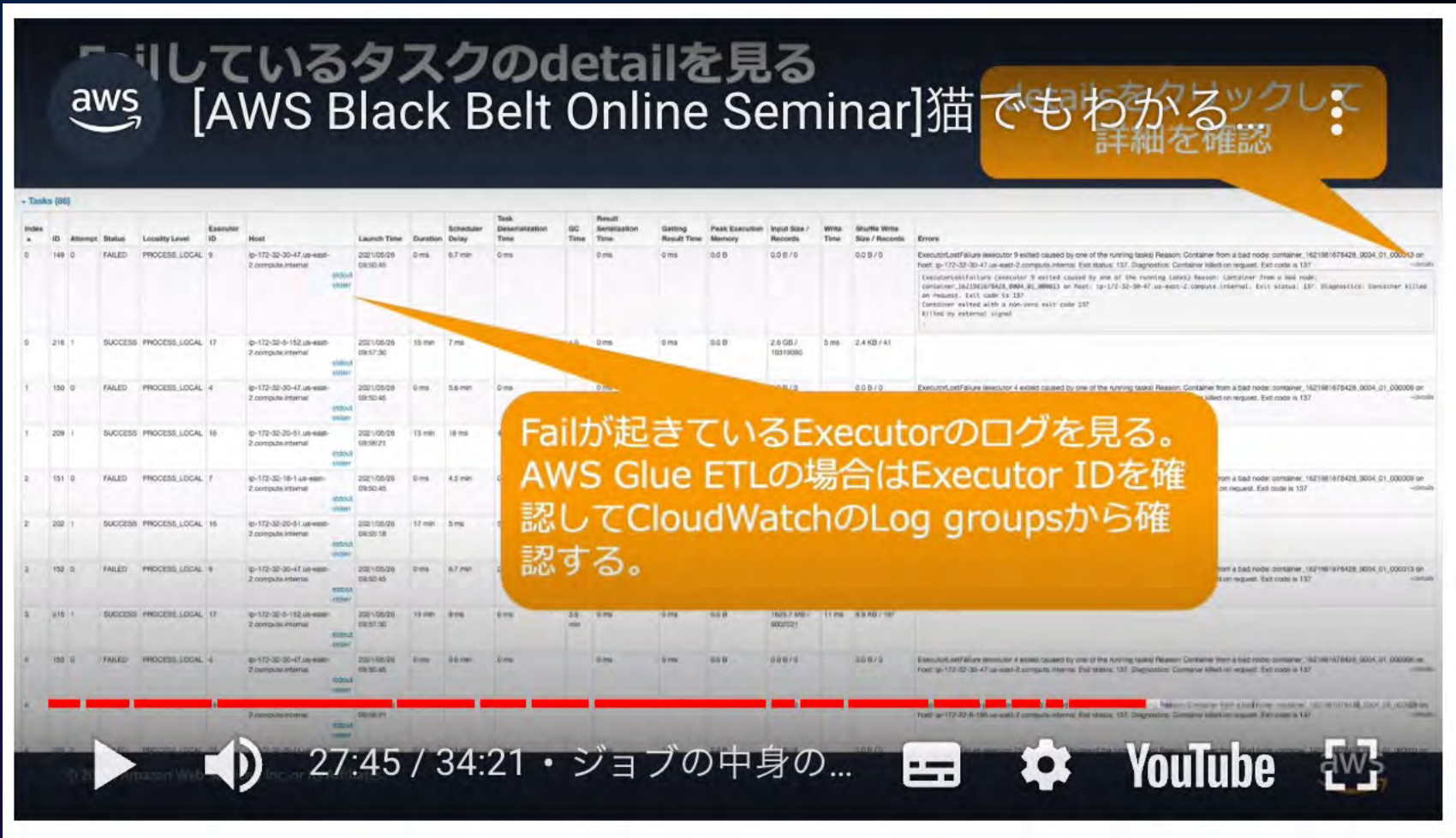
運用上の優秀性
1. ワークロードの健全性を監視



[AWS Blog: AWS Glue サーバーレス Spark UI導入によるモニタリングとトラブルシューティングの改善]

補足：Spark UI の確認の仕方

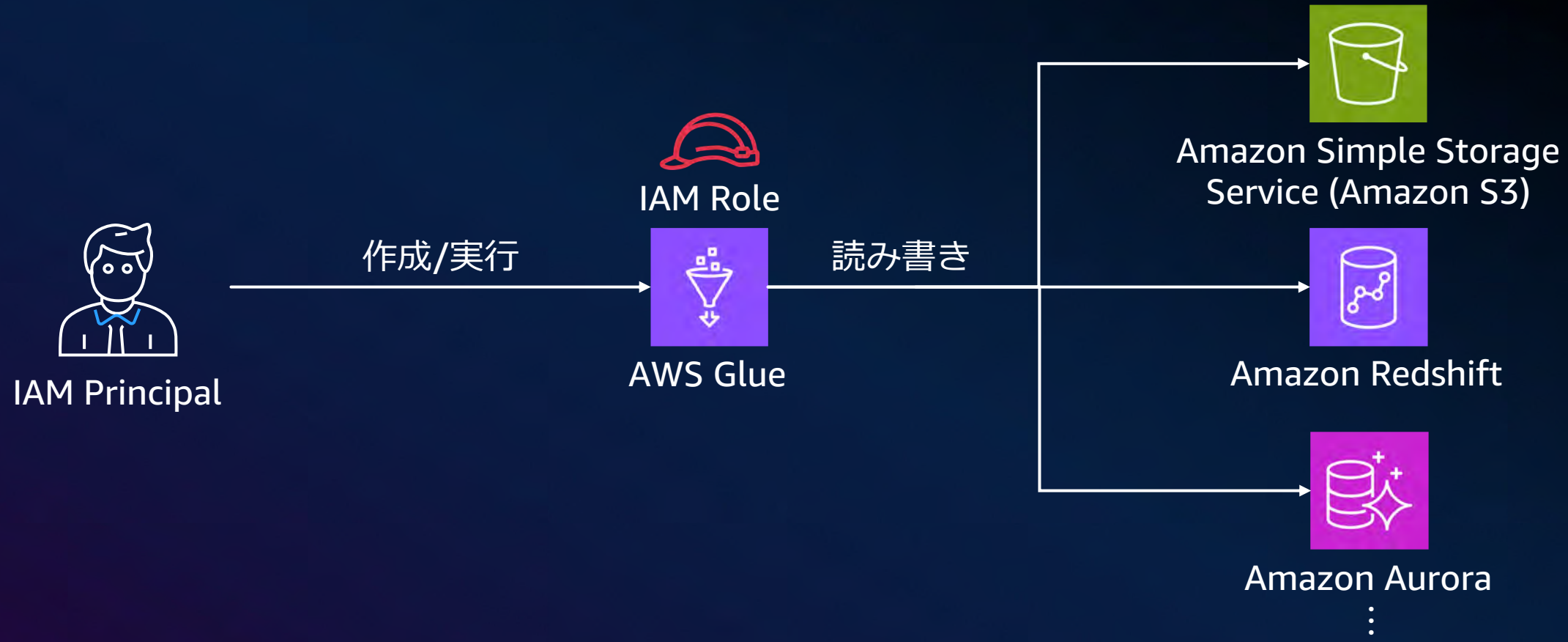
運用上の優秀性
1. ワークロードの健全性を監視



[AWS Black Belt Online Seminar] 猫でもわかる、AWS Glue ETLパフォーマンス・チューニング

AWS Glue のアクセスコントロールの全体像

セキュリティ
4.データアクセス
コントロールを実装

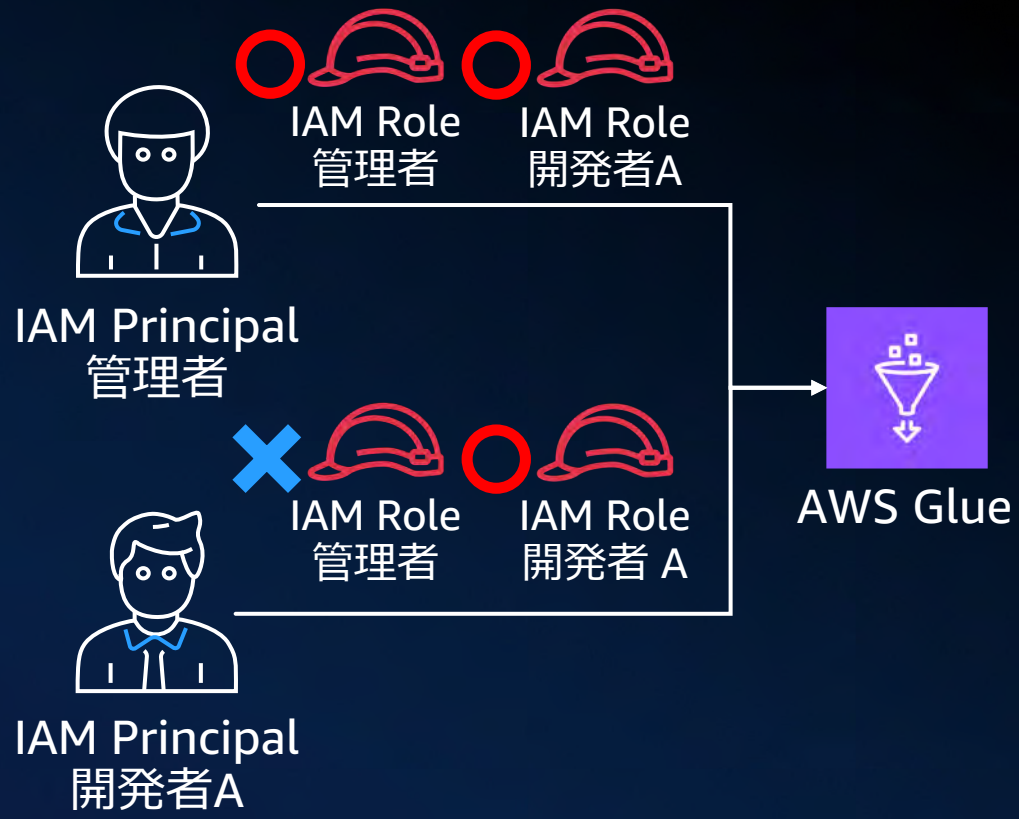


1. ジョブを作成/実行できる権限

2. 各種データソースに
アクセスできる権限

AWS Glue ジョブ作成/実行時の権限

セキュリティ
4.データアクセス
コントロールを実装



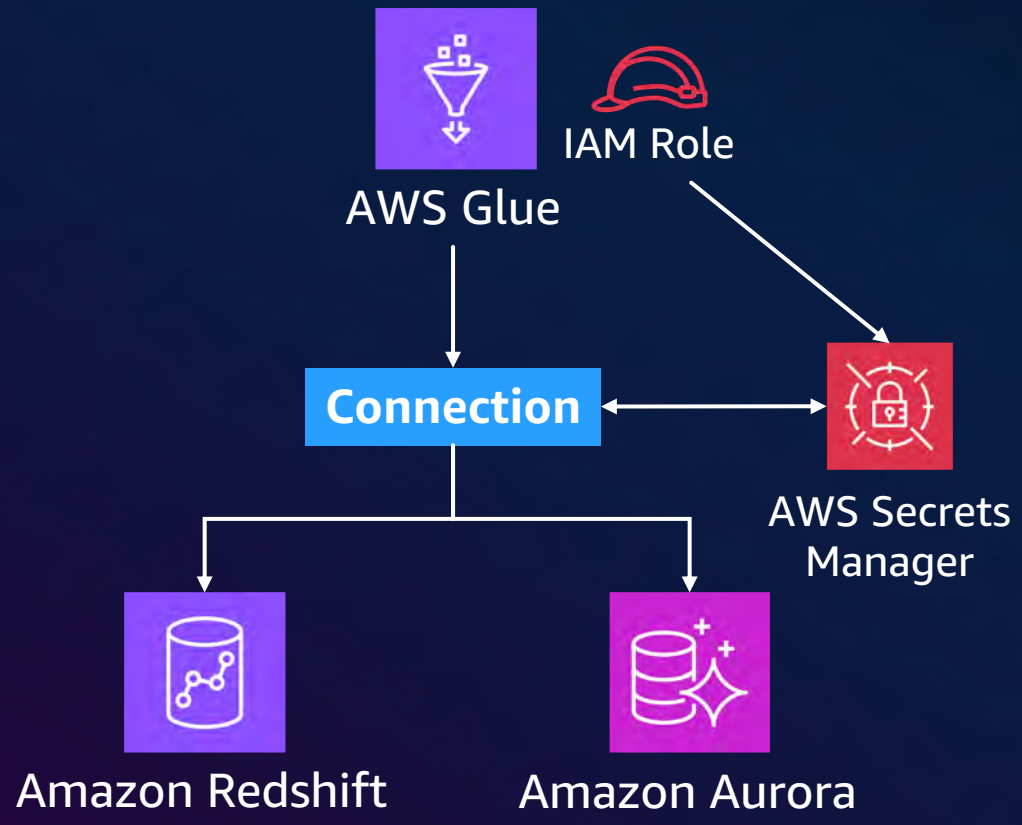
- AWS Glue Studio にアクセスできる
- AWS Glue の Script を S3 における
- AWS Glue ジョブを作成できる
- etc …

- IAM Policy (**IAM:PassRole**) で制御可能

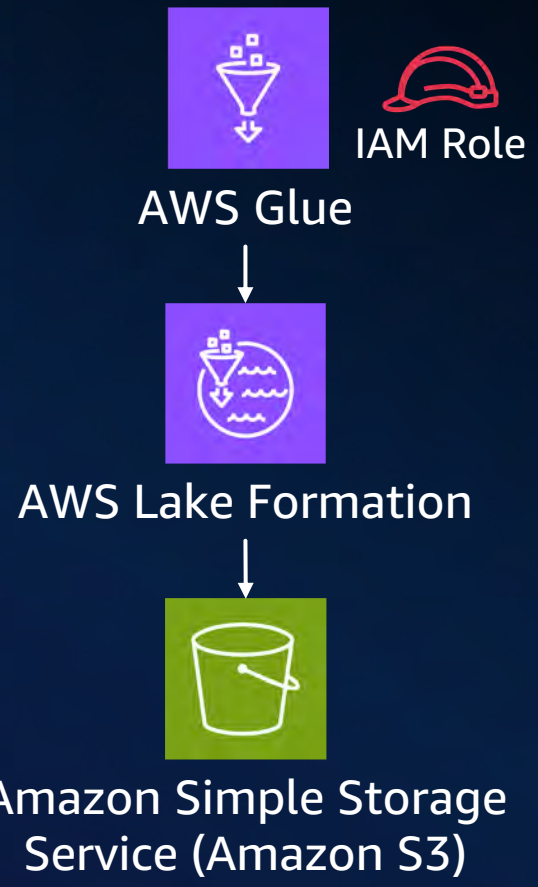
データソースへのアクセス制御

セキュリティ
4.データアクセス
コントロールを実装

Glue Connection

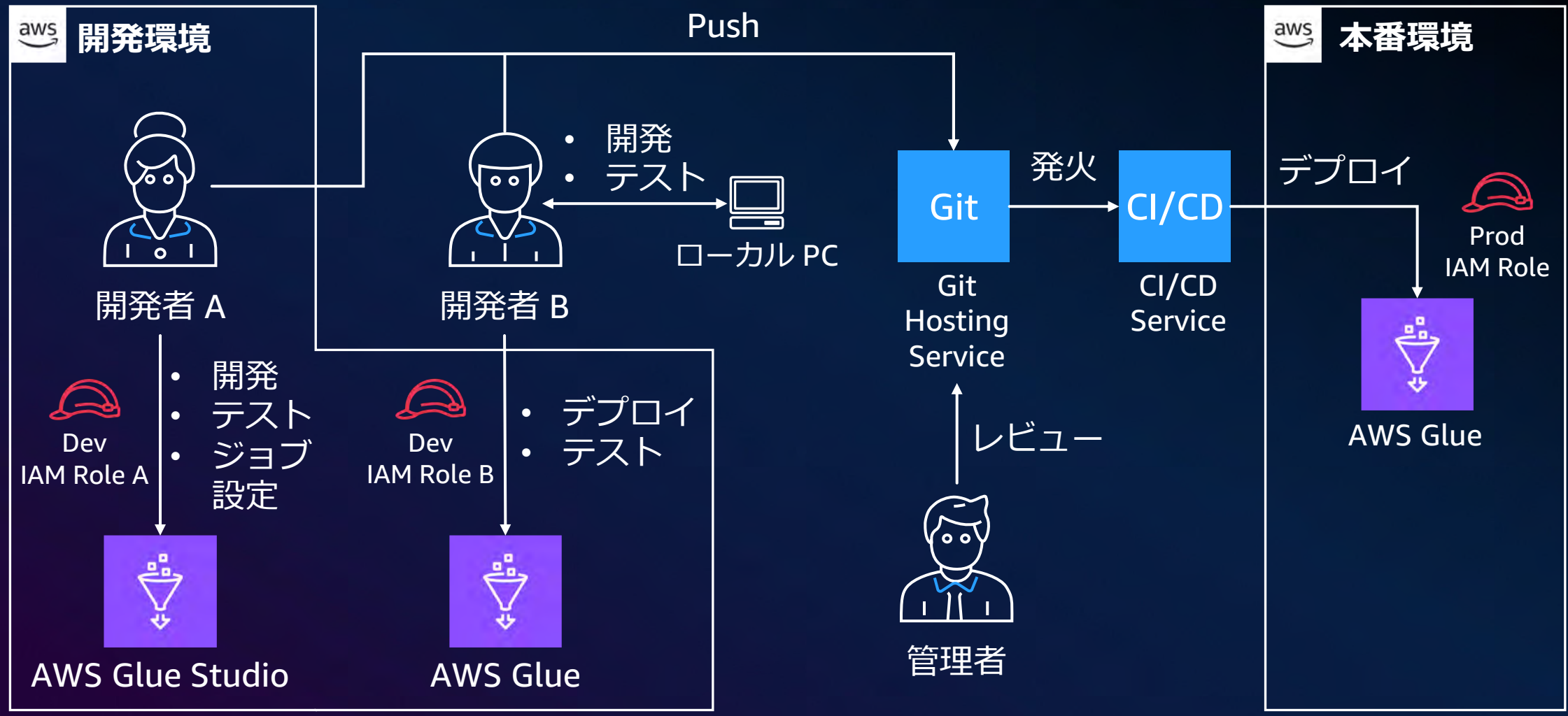


AWS LakeFormation



アクセスコントロールによる AWS Glue ジョブ実装の民主化

セキュリティ
4.データアクセス
コントロールを実装



まとめ

AWS Glue で ETL ジョブを実際に開発・運用を行っていく際のベストプラクティスについて説明した

- 開発時のベストプラクティス
  - ノーコードローコード開発では AWS Glue Studio の各機能を有効活用する
  - コーディング開発では開発をサポートする様々な資源があるので有効活用する。特にテストやデプロイなどを CI/CD することを推奨する
  - ジョブの実行はワークフローエンジンを利用する

- 運用時のベストプラクティス
  - エラー発生時は Error Category/エラーメッセージ/Metrics/Spark UI を確認する
  - アクセスコントロールの仕組みを理解しながら Glue ジョブのデータアクセスをコントロールする

Thank you!

佐藤 祥多

shotast@amazon.co.jp