

JAPAN | 2024

aws SUMMIT



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

データソース



データレイク



データウェアハウス



活用



モダンなデータ分析基盤のアーキテクチャ

データソース



データレイク



データウェアハウス



活用



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?

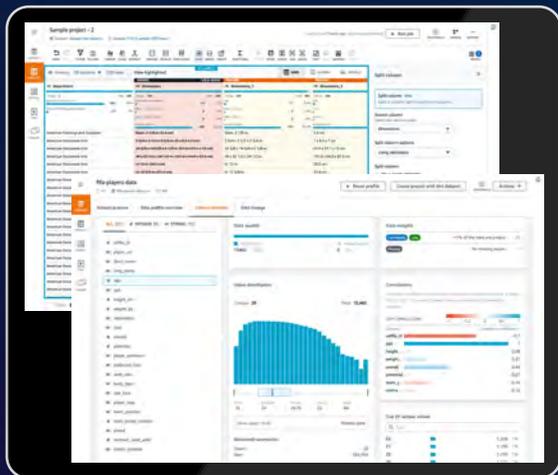
AWS Glue ジョブの開発と運用時に考慮すること



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～

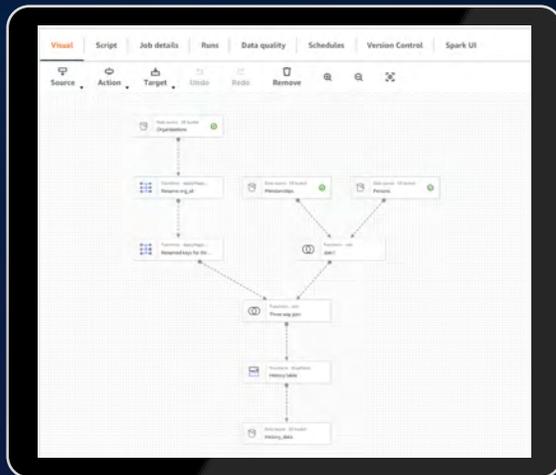
AWS Glue のジョブ実装を支えるツール



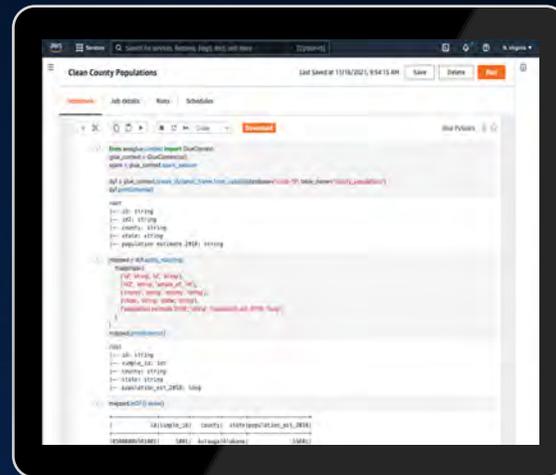
ビジュアル



ビジネスアナリスト
データサイエンティスト



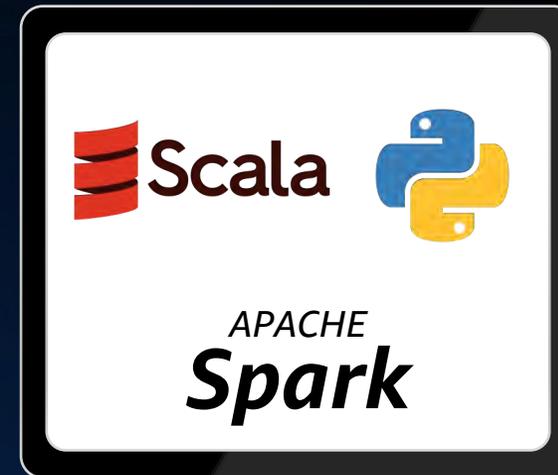
ETL エンジニア



Notebook



データエンジニア
データサイエンティスト

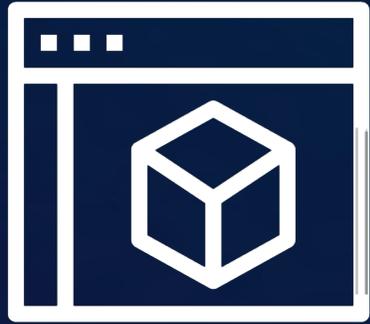


IDE



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

The screenshot displays the AWS Glue Studio Visual Editor interface. The main workspace shows a workflow with two data source nodes: "Data source - S3 bucket Plan assignment" and "Data source - S3 bucket Subscribers". Both are connected to a "Transform - Join Join" node. The interface includes tabs for "Visual", "Script", "Job details", "Runs", "Data quality - updated", "Schedules", and "Version Control".

At the bottom, the "Data preview" section shows a table with 200 records. The previewed data is as follows:

| msisdn | plan_id |
|--------------------------------------|--------------|
| 0EE9CF575F6DFC439912 83EBEFDE9DBF | 422426142648 |
| E2F7FFF946A31AB8C67B 8D0C7B74D0D1 | 968763331038 |

The right-hand panel shows the "Data source properties - S3" configuration for the selected "Plan assignment" data source. The properties are:

- Name: Plan assignment
- S3 source type: S3 location (selected)
- S3 URL: s3://aws-bigdata-blog/artifacts/tel
- Recursive: (selected)
- Data format: CSV
- Delimiter: Comma (,)
- Quote character: Double quote (")
- First line of source file contains column headers: (selected)

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
- DB**
 - MySQL
 - PostgreSQL
 - SAP HANA
- DWH**
 - Snowflake
 - Google BigQuery
- SaaS**
 - Salesforce

- Amazon Redshift
- Amazon DynamoDB
- Amazon OpenSearch Service
- Oracle SQL
- Microsoft SQL Server
- Teradata Vantage

など

100種類以上

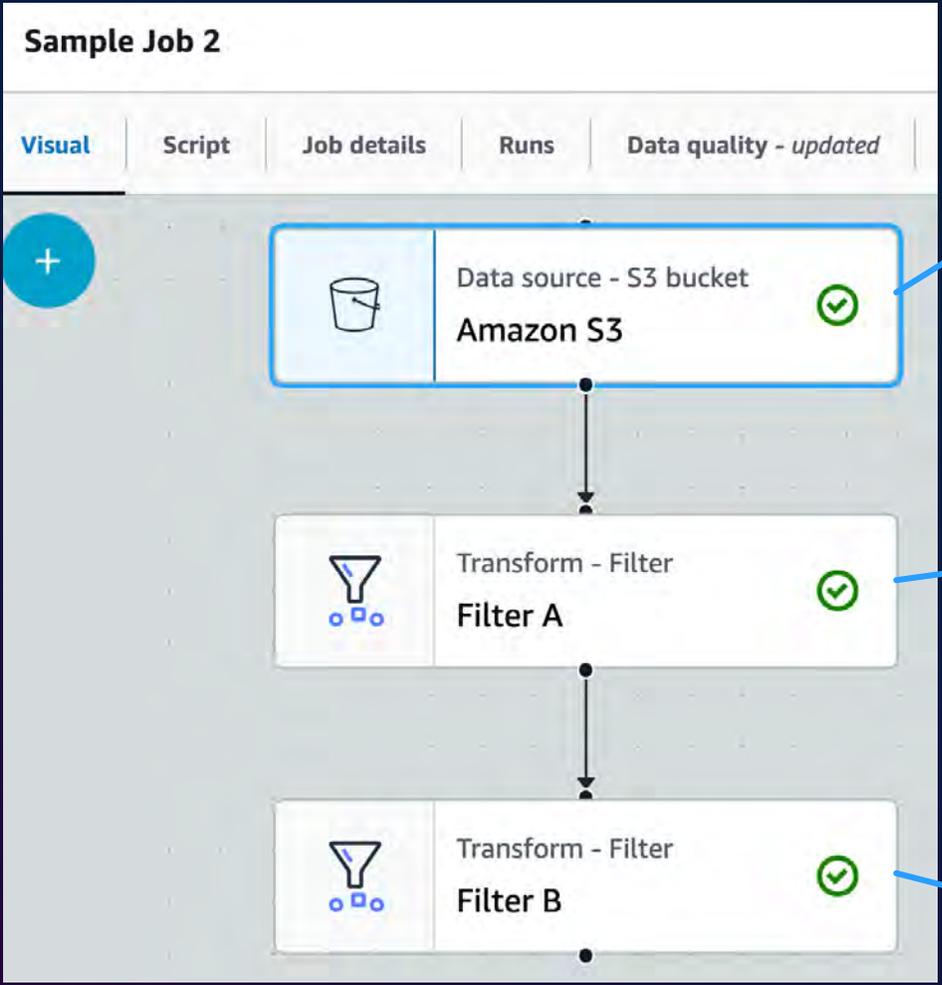
変換

- 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

Sample Job Last modified on 2024/5/20 19:13:49 Actions Save Run

Visual | Script | **Job details** | Runs | Data quality - updated | Schedules

Version Control

Basic properties [Info](#)

Name
Sample Job

Description - optional

Descriptions can be up to 2048 characters long.

IAM Role
Role assumed by the job with permission to access your data stores. Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job.

↕ ↻

Type
The type of ETL job. This is set automatically based on the types of data sources you have selected.

Spark

Glue version [Info](#)

Glue version [Info](#)
Glue 4.0 - Supports spark 3.3, Scala 2, Python 3

Language
Python 3

Worker type
Set the type of predefined worker that is allowed when a job runs.
G 1X
(4vCPU and 16GB RAM)

Automatically scale the number of workers
 AWS Glue will optimize costs and resource usage by dynamically scaling the number of workers up and down throughout the job run. Requires Glue 3.0 or later.

Requested number of workers
The number of workers you want AWS Glue to allocate to this job.
10

Generate job insights
 AWS Glue will analyze your job runs and provide insights on how to optimize your jobs and the reasons for job failures.

Job bookmark [Info](#)
Specifies how AWS Glue processes job bookmark when the job runs. It can remember previously processed data (Enable), update state information (Pause), or ignore state information (Disable).
Disable

Flex execution [Info](#)
 Reduce costs by running this job on spare capacity. Ideal for non-urgent workloads.

AWS Glue Studio の Git 連携

The screenshot shows the 'Version Control' tab in the AWS Glue Studio interface for a job named 'gluecopytest'. At the top, it indicates the job was last modified on 2024/5/5 17:49:36 and provides 'Actions', 'Save', and 'Run' buttons. Below this are tabs for 'Script', 'Job details', 'Runs', 'Data quality - updated', 'Schedules', and 'Version Control'. A notification box states: 'Latest commit Latest available commit [476a8f](#) is from AWS Code Commit on glue-git/main.' The main configuration area is titled 'Git configuration' and includes a 'Reset' button. It is divided into two sections: 'Git configuration' and 'Repository configuration'. Under 'Git configuration', the 'Git service' is set to 'AWS Code Commit'. Under 'Repository configuration', the 'Repository' is 'glue-git', the 'Branch' is 'main', and the 'Folder' is 'gluecopytest'. A note below the folder field says: 'Override the default folder name otherwise the job name will be used if blank.'

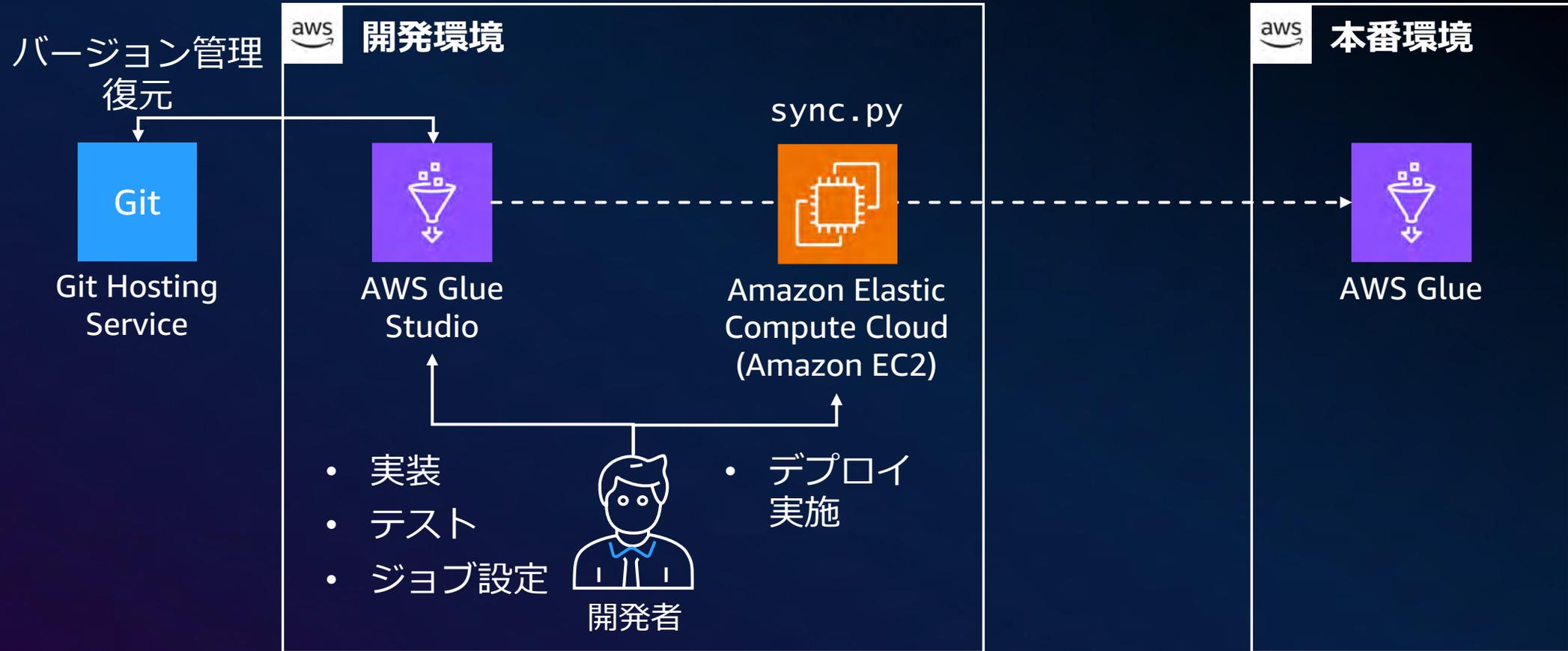
対応サービス

- AWS CodeCommit
- GitHub
- Gitlab
- Bitbucket

ノーコードローコード開発例1

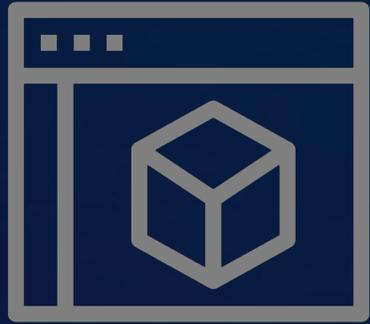


ノーコードローコード開発例2



[AWS Blog: Synchronize your AWS Glue Studio Visual Jobs to different environments](#)

AWS Glue のジョブ実装パターン



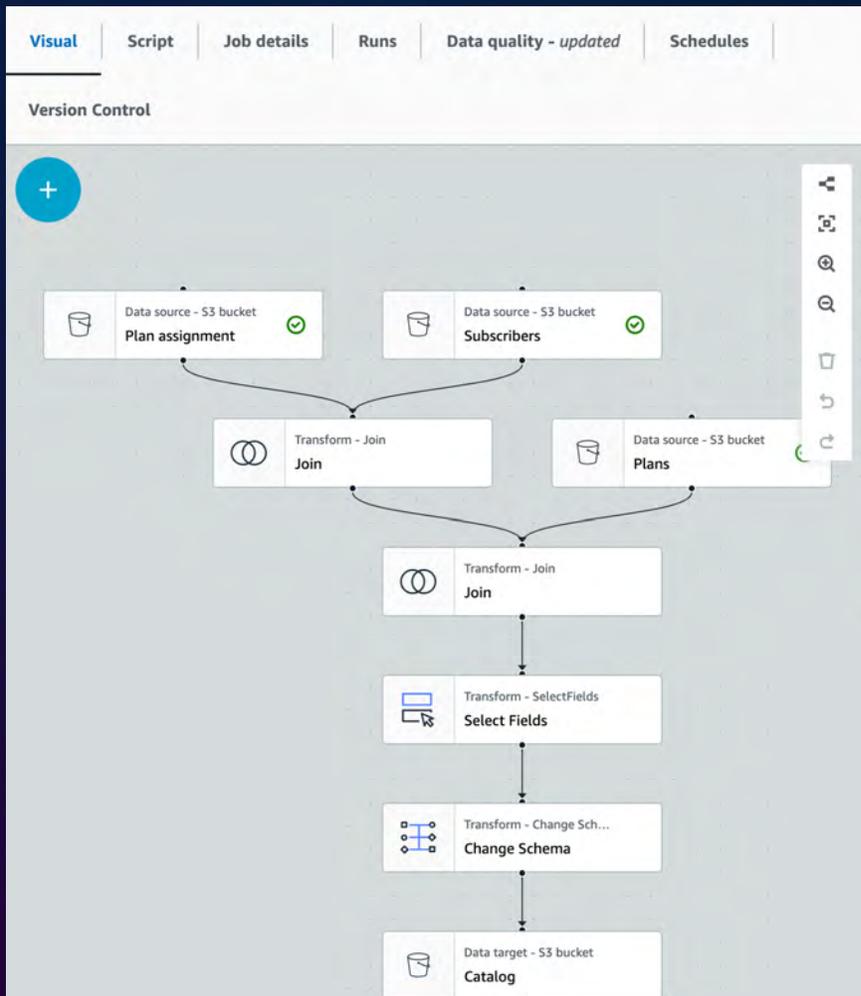
ノーコード
ローコード実装



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

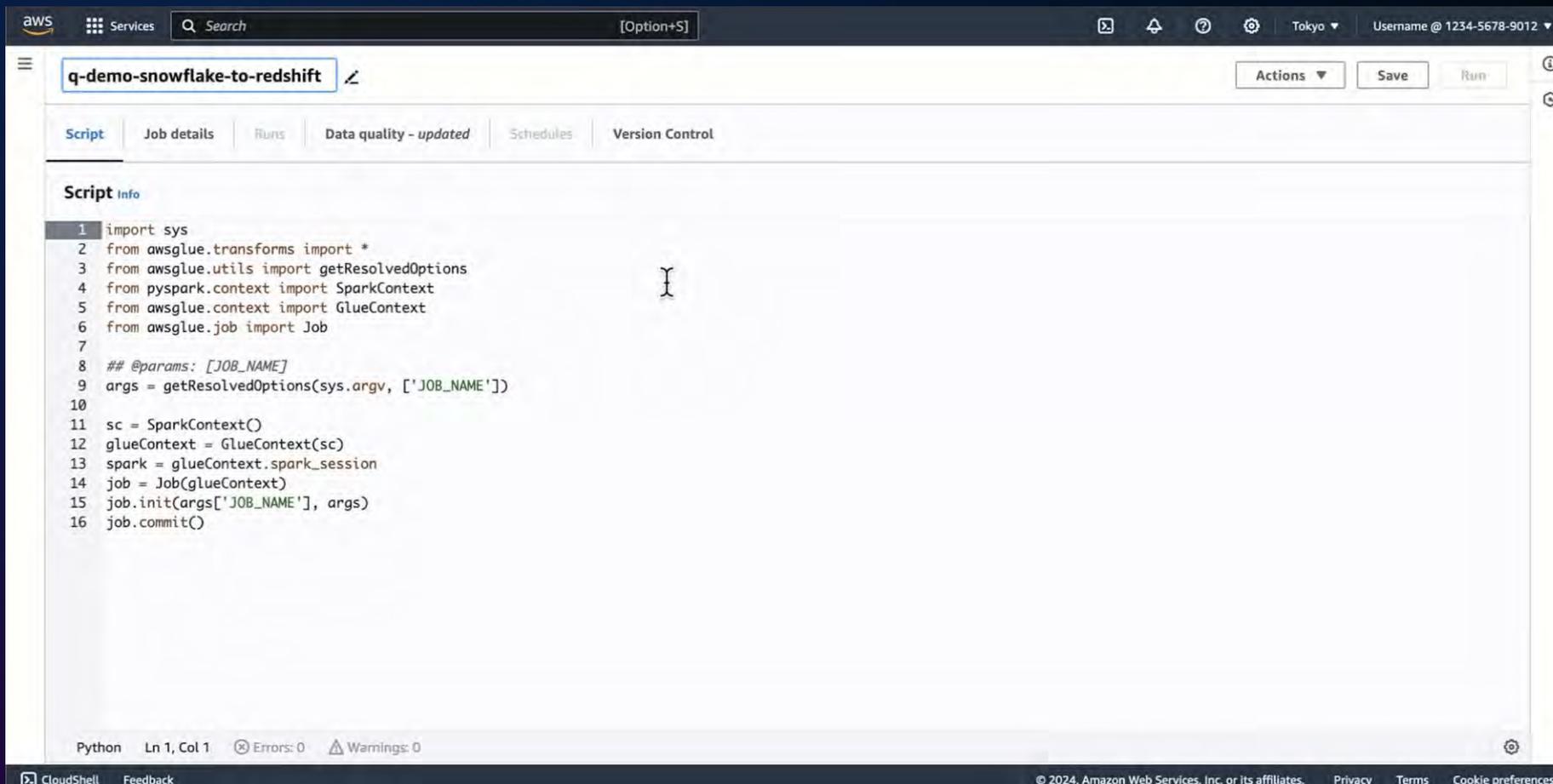


The screenshot shows the Script tab of AWS Glue Studio, displaying the code generated from the workflow diagram. The code is as follows:

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
9 sc = SparkContext()
10 glueContext = GlueContext(sc)
11 spark = glueContext.spark_session
12 job = Job(glueContext)
13 job.init(args['JOB_NAME'], args)
14
15 # Script generated for node Plans
16 Plans_node1685352206564 = glueContext.create_dynamic_frame.from_options(format_opt
17
18 # Script generated for node Plan assignment
19 Planassignment_node1685352066932 = glueContext.create_dynamic_frame.from_options(f
20
21 # Script generated for node Subscribers
22 Subscribers_node1 = glueContext.create_dynamic_frame.from_options(format_options={
23
24 # Script generated for node Join
25 Join_node1685352536679 = Join.apply(frame1=Planassignment_node1685352066932, frame
26
27 # Script generated for node Join
28 Join_node1685352743084 = Join.apply(frame1=Join_node1685352536679, frame2=Plans_no
29
30 # Script generated for node Select Fields
31 SelectFields_node1685353779136 = SelectFields.apply(frame=Join_node1685352743084,
32
33 # Script generated for node Change Schema
34 ChangeSchema_node1685353487597 = ApplyMapping.apply(frame=SelectFields_node1685353
35
36 # Script generated for node Catalog
37 Catalog_node1685352920969 = glueContext.getSink(path="s3://aws-glue-temporary-3805
38 Catalog_node1685352920969.setCatalogInfo(catalogDatabase="default", catalogTableNam
39 Catalog_node1685352920969.setFormat("glueparquet", compression="snappy")
40 Catalog_node1685352920969.writeFrame(ChangeSchema_node1685353487597)
41 job.commit()
```

AWS Glue をコードで実装する補助ツール

Amazon Q data integration in AWS Glue



The screenshot shows the AWS Glue console interface. At the top, there's a search bar and navigation tabs for 'Script', 'Job details', 'Runs', 'Data quality - updated', 'Schedules', and 'Version Control'. The 'Script' tab is active, displaying a Python script for a job named 'q-demo-snowflake-to-redshift'. The script includes imports for sys, awsglue, pyspark, and GlueContext, and defines a job that reads data from Snowflake and writes it to Redshift. The status bar at the bottom indicates 'Python Ln 1, Col 1' with 0 errors and 0 warnings.

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 ## @params: [JOB_NAME]
9 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 spark = glueContext.spark_session
14 job = Job(glueContext)
15 job.init(args['JOB_NAME'], args)
16 job.commit()
```

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

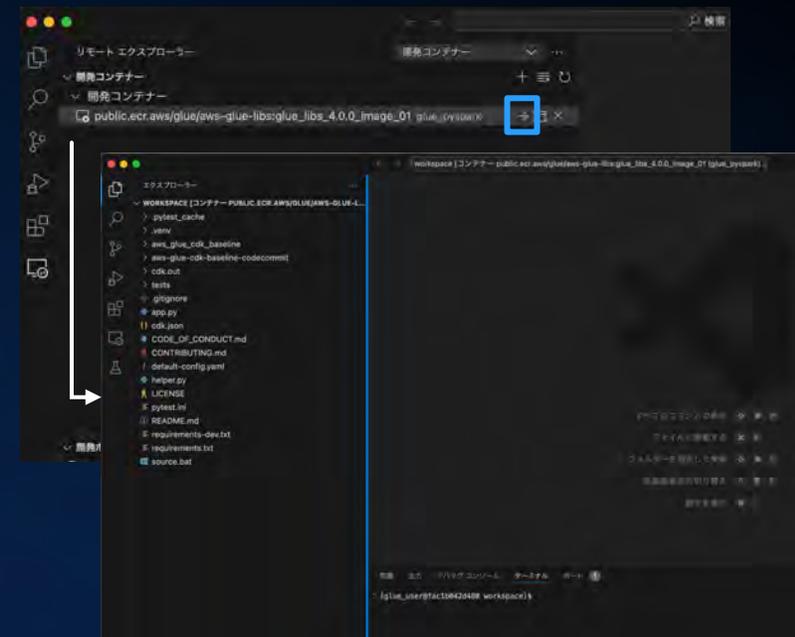
1. Docker image を Pull

```
$ 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

20 @pytest.fixture(scope="module", autouse=True)
21 def glue_context():
22     sys.argv.append('--JOB_NAME')
23     sys.argv.append('JobName')
24
25     args = getResolvedOptions(sys.argv, ['JOB_NAME'])
26     context = GlueContext(SparkContext.getOrCreate())
27     job = Job(context)
28     job.init(args['JOB_NAME'], args)
29
30     yield(context)
31
32     job.commit()
33
34 def test_counts(glue_context):
35     dyf = create_dyf(glue_context)
36     dyf = glue_job.apply_transform(dyf)
37     assert dyf.toDF().count() == 11
38
39 def create_dyf(glue_context):
40     spark_session = glue_context.spark_session
41     df = spark_session.createDataFrame(
42         [
43             ("TestMovie1", 1.0, 3),
44             ("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

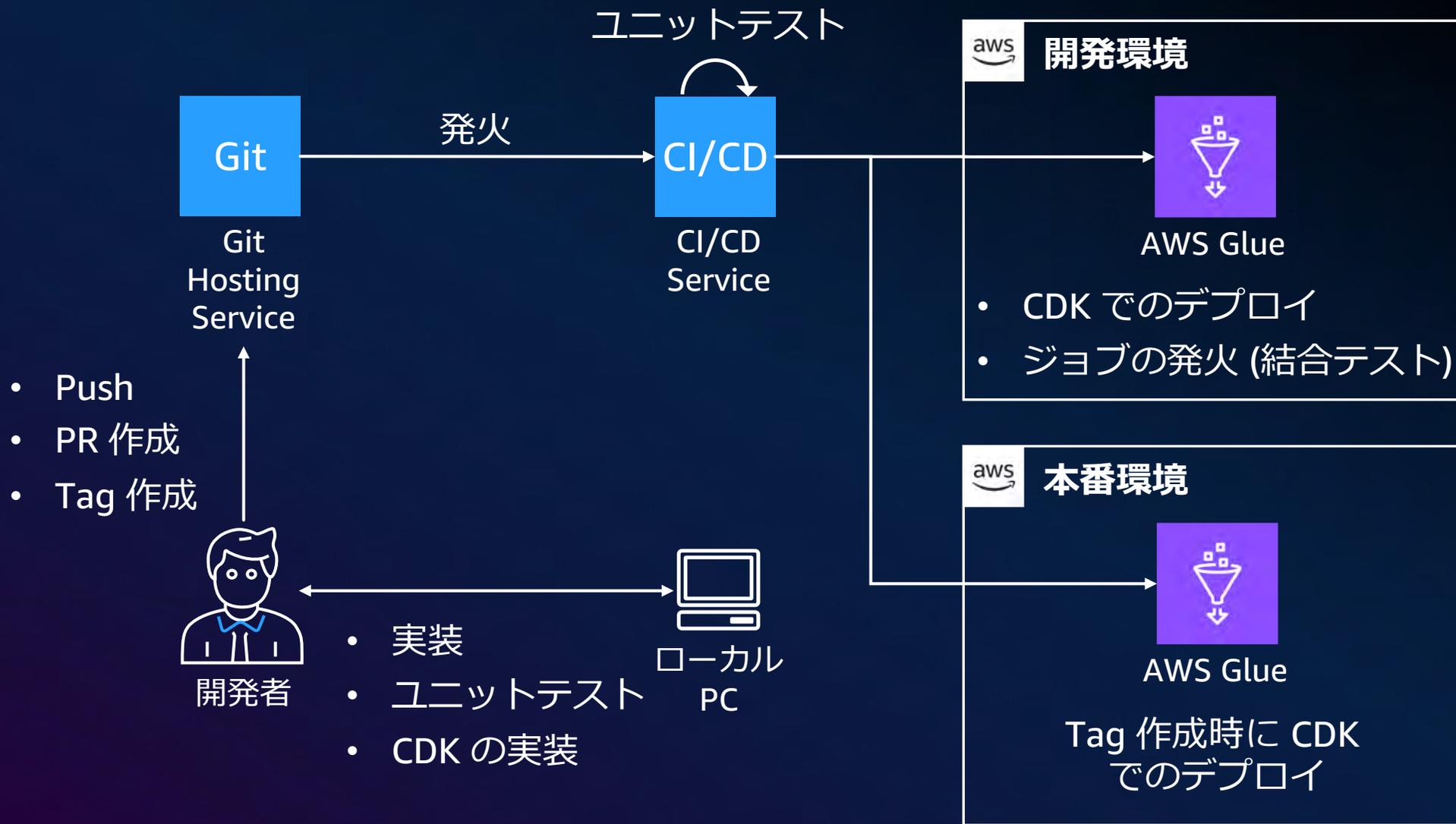
など

- IaC 系

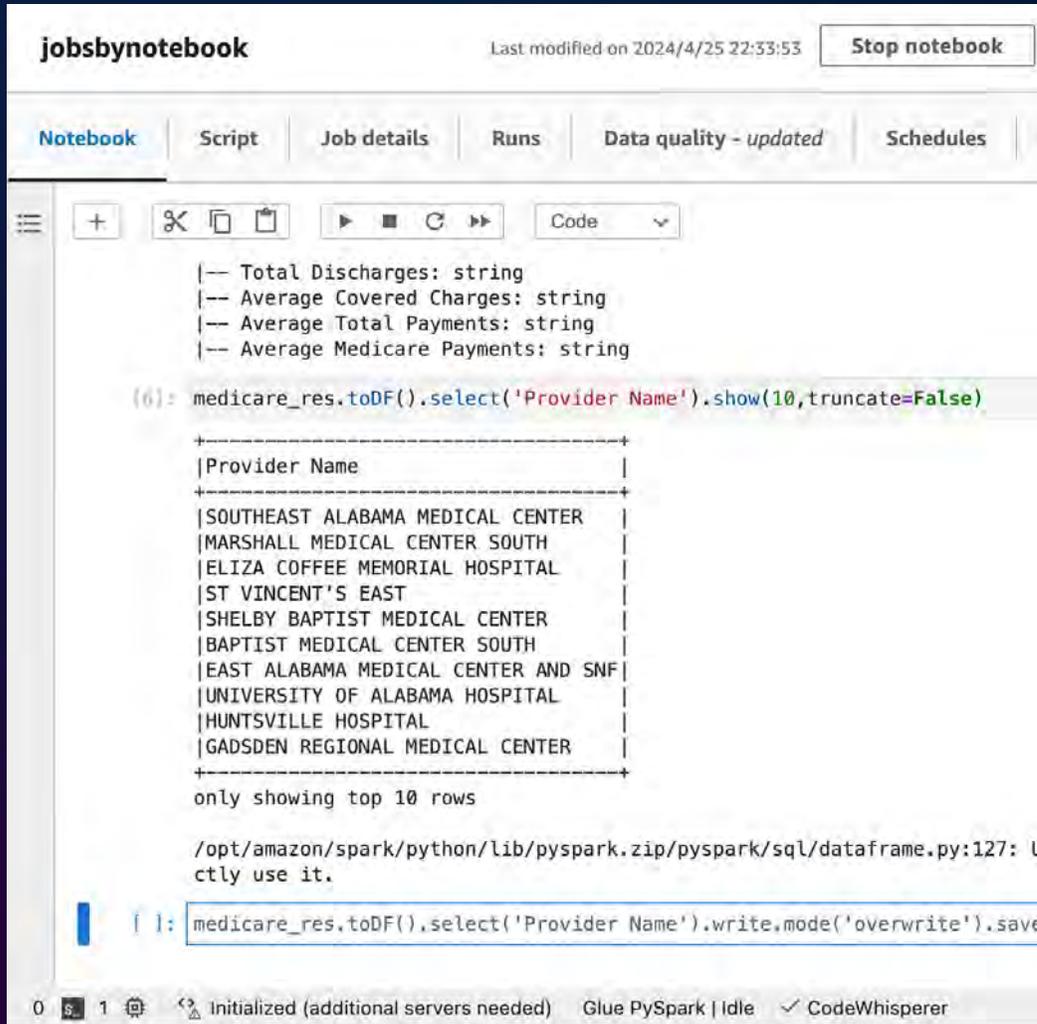
- AWS CloudFormation
- AWS CDK (Cloud Developer Kit)
- Terraform

など

ローカル環境でのコーディング開発例



補足 : Jupyter Notebook



jobsbynotebook Last modified on 2024/4/25 22:33:53 Stop notebook

Notebook Script Job details Runs Data quality - updated Schedules

```
|-- Total Discharges: string
|-- Average Covered Charges: string
|-- Average Total Payments: string
|-- Average Medicare Payments: string

(6): medicare_res.toDF().select('Provider Name').show(10,truncate=False)
```

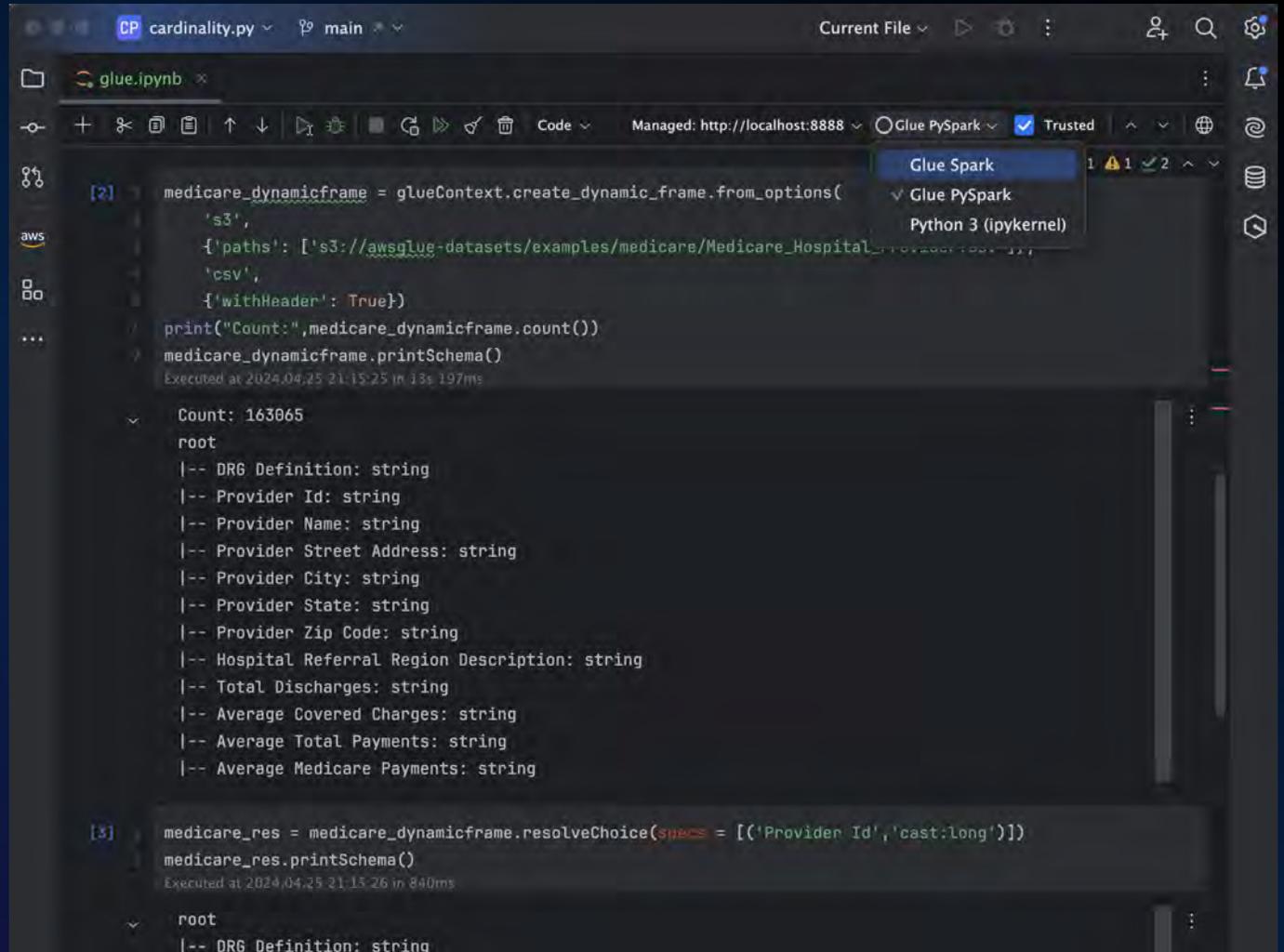
| Provider Name |
|-------------------------------------|
| SOUTHEAST ALABAMA MEDICAL CENTER |
| MARSHALL MEDICAL CENTER SOUTH |
| ELIZA COFFEE MEMORIAL HOSPITAL |
| ST VINCENT'S EAST |
| SHELBY BAPTIST MEDICAL CENTER |
| BAPTIST MEDICAL CENTER SOUTH |
| EAST ALABAMA MEDICAL CENTER AND SNF |
| UNIVERSITY OF ALABAMA HOSPITAL |
| HUNTSVILLE HOSPITAL |
| GADSDEN REGIONAL MEDICAL CENTER |

only showing top 10 rows

/opt/amazon/spark/python/lib/pyspark.zip/pyspark/sql/dataframe.py:127: U
ctly use it.

```
1: medicare_res.toDF().select('Provider Name').write.mode('overwrite').save
```

0 s 1 Initialized (additional servers needed) Glue PySpark | Idle CodeWhisperer



CP cardinality.py main Current File

glue.ipynb Managed: http://localhost:8888 Glue PySpark Trusted Python 3 (ipykernel)

```
[2] medicare_dynamicframe = glueContext.create_dynamic_frame.from_options(
    's3',
    {'paths': ['s3://aws-glue-datasets/examples/medicare/Medicare_Hospital_Provider_001.csv'],
     'withHeader': True})

print("Count:", medicare_dynamicframe.count())
medicare_dynamicframe.printSchema()

Executed at 2024/04/25 21:15:25 in 13s 197ms
```

```
Count: 163065
root
|-- DRG Definition: string
|-- Provider Id: string
|-- Provider Name: string
|-- Provider Street Address: string
|-- Provider City: string
|-- Provider State: string
|-- Provider Zip Code: string
|-- Hospital Referral Region Description: string
|-- Total Discharges: string
|-- Average Covered Charges: string
|-- Average Total Payments: string
|-- Average Medicare Payments: string
```

```
[3] medicare_res = medicare_dynamicframe.resolveChoice(specs = [('Provider Id', 'cast:long')])
medicare_res.printSchema()

Executed at 2024/04/25 21:15:26 in 840ms
```

```
root
|-- DRG Definition: string
```

補足：Jupyter Notebook のジョブ実行

The screenshot displays the AWS Glue console interface for a Jupyter Notebook. At the top, the notebook is identified as 'notebookjobv2', last modified on 2024/4/26 at 1:07:34. Action buttons include 'Stop notebook', 'Download Notebook', 'Actions', 'Save', and 'Run'. The 'Run' button is highlighted with a blue border. Below the notebook title, navigation tabs are visible: 'Notebook', 'Script', 'Job details', 'Runs', 'Data quality - updated', 'Schedules', and 'Version Control'. The main area shows a code cell with the following configuration commands:

```
[9]: %idle_timeout 2880
      %glue_version 4.0
      %worker_type G.1X
      %number_of_workers 2
```

The output of the code cell shows the following messages:

```
Current idle_timeout is None minutes.
idle_timeout has been set to 2880 minutes.
Setting Glue version to: 4.0
Previous worker type: None
Setting new worker type to: G.1X
Previous number of workers: None
Setting new number of workers to: 2
```

Below the configuration cell, a code cell is partially visible with the following code:

```
[8]: import sys
      from aws glue.transforms import *
```

補足：AWS Glue のジョブ実行管理の実装

AWS Glue workflows

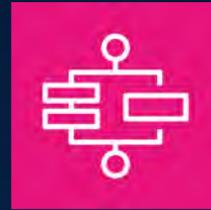


Glue のジョブや
クローラーの
ジョブフローを管理

シンプル

シンプルがよい
コスト掛けずに実装

AWS Step Functions

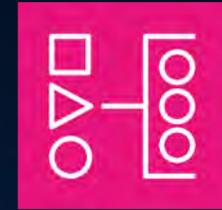


AWS サービスと連携可能な
フルマネージド
ワークフローエンジン

AWS の各サービス
実行が簡単にできる

AWS サービスと連携
サーバーレスが良い

Amazon Managed Workflows for Airflow (MWAA)



マネージドな
Apache Airflow

高機能で Apache Airflow の
知見を生かせる/貯められる

Apache Airflow を
使いたいとき

概要

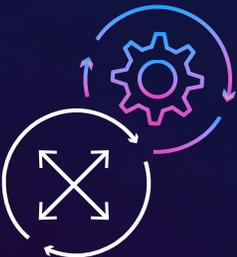
強み

いつ使うか

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

1. ワークロードの健全性を監視



セキュリティ

4. データアクセスコントロールを実装



信頼性



パフォーマンス 効率



コスト最適化



持続可能性



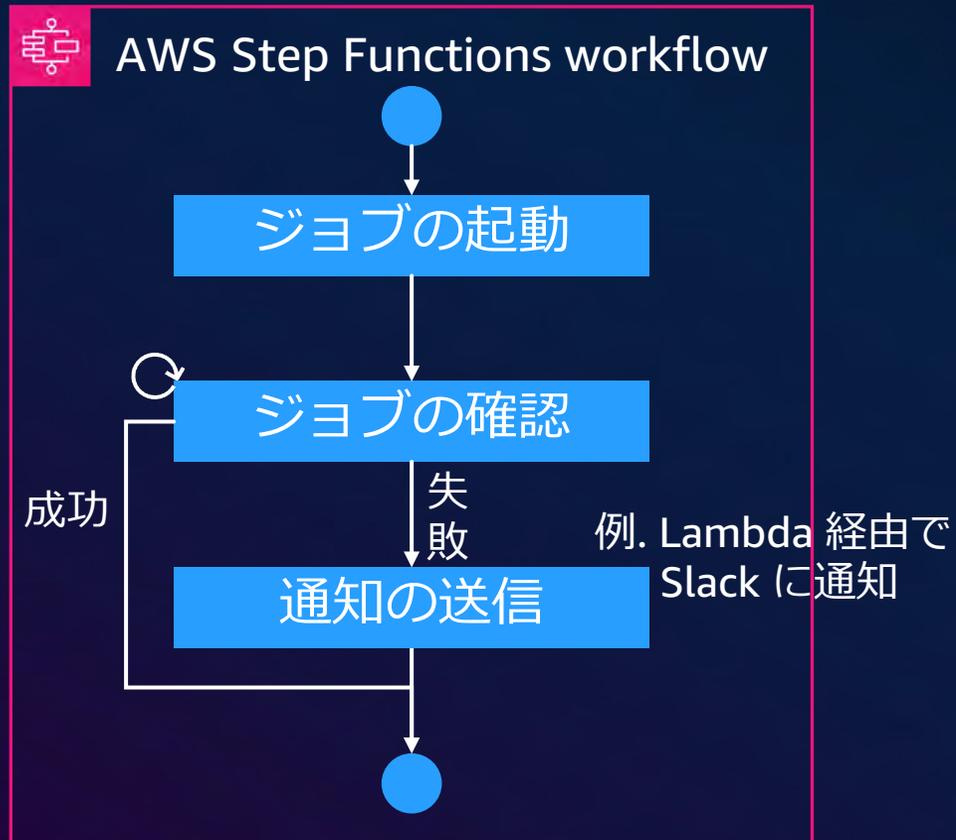
[AWS Documentation: AWS Well-Architected Framework Data Analytics Lens](#)

ジョブ実行エラーの検知方法

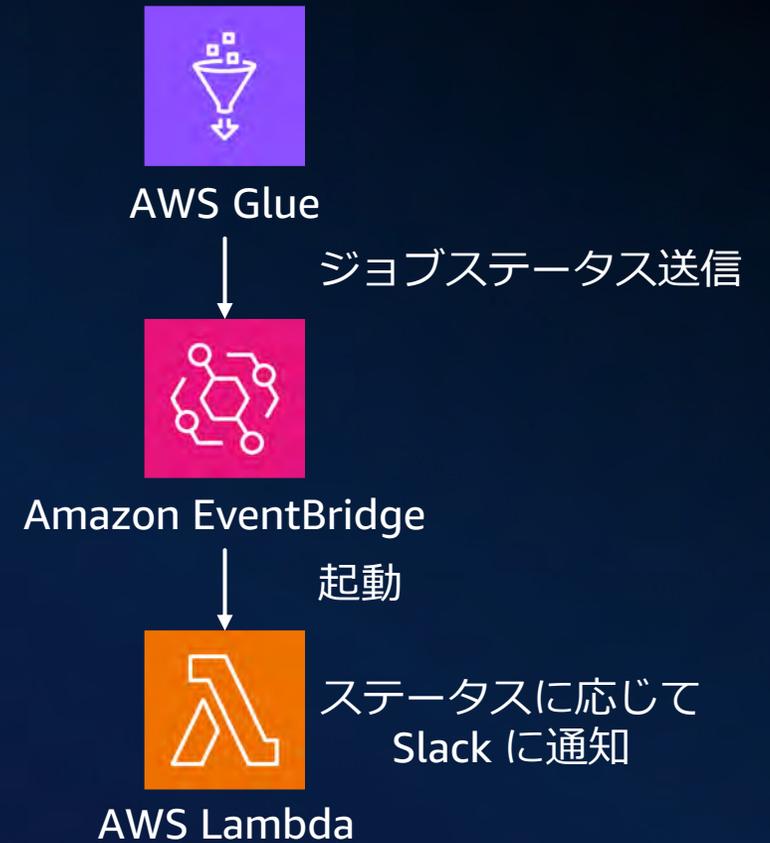
運用上の優秀性

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

The screenshot displays the AWS Glue console interface for a job run. At the top, it shows 'Job runs (1/1) Info' with a refresh button and 'View details' link. The last update is noted as 'May 23, 2024 at 14:51:43'. A search bar is present for filtering job runs by property. Below this is a table with columns: Run status, Retries, Start time (Local), End time (Local), Duration, Capacit..., Worker type, and Glue version. The first row shows a 'Failed' status with 0 retries, starting at 05/20/2024 19:13:51 and ending at 05/20/2024 19:15:08, with a duration of 1 m 8 s, 10 DPU capacity, G.1X worker type, and Glue version 4.0.

Below the table, there are tabs for 'Run details', 'Input arguments (10)', 'Continuous logs', 'Run insights', 'Metrics', and 'Spark UI'. The 'Run details' tab is active, showing an error message in a red-bordered box: 'Error Category: RESOURCE_NOT_FOUND_ERROR; An error occurred while calling o147.pyWriteDynamicFrame. The specified bucket does not exist (Service: Amazon S3; Status Code: 404; Error Code: NoSuchBucket; Request ID: EQF44CKKCKGN07DZ; S3 Extended Request ID: fprn1TH2IJ5HjccOCgK6+iGU86gKu0MOuGaTQgy/RI+gFyiZE6VpWy87NLbzFPQkctKlcldayRWs=; Proxy: null)'. Below the error message is a table of job properties:

| | | | |
|---|------------------------|---|--------------------------|
| Job name | Start time (Local) | Glue version | Last modified on (Local) |
| Sample Job | 05/20/2024 19:13:51 | 4.0 | 05/20/2024 19:15:08 |
| Id | End time (Local) | Worker type | Log group name |
| jr_af803b5f8601f32520713579c91d5da21364f1db6cf1adca941e159f4ee1df4a | 05/20/2024 19:15:08 | G.1X | /aws-glue/jobs |
| Run status | Start-up time | Max capacity | Number of workers |
| Failed | 9 seconds | 10 DPU | 10 |
| Retry attempt number | Execution time | Execution class | Timeout |
| Initial run | 1 minute 8 seconds | Standard | 2880 minutes |
| Trigger name | Security configuration | Cloudwatch logs | |
| - | - | <ul style="list-style-type: none">All logsOutput logsError logs | |

Observability metrics

運用上の優秀性

1. ワークロード
の健全性を監視

The screenshot shows the AWS Glue console interface. At the top, it displays 'Job runs (1/1) Info' with a search filter 'Filter job runs by property'. Below this is a table of job runs with columns for 'Run status', 'Retries', 'Start time (Local)', and 'End time'. One job is listed as 'Failed' with 0 retries, starting on 05/20/2024 at 19:13:51. Below the table, there are tabs for 'Run details', 'Input arguments (10)', 'Continuous logs', and 'Run insights'. The 'Run details' tab is active, showing an error message: 'Error Category: RESOURCE_NOT_FOUND_ERROR; An error occurred while calling...'. Below the error message, there is a table of job details with columns for 'Job name', 'Start time (Local)', 'Id', 'End time (Local)', 'Run status', 'Start-up time', 'Retry attempt number', and 'Execution time'. The job name is 'Sample Job', start time is 05/20/2024 19:13:51, id is 'jr_af803b5f8601f32520713579c91d5da21364f1db6cf1adca941e159f4ee1df4a', end time is 05/20/2024 19:15:08, run status is 'Failed', start-up time is 9 seconds, and execution time is 1 minute 8 seconds.

The screenshot shows the 'Job metrics' settings in the AWS Glue console. It includes several checkboxes and options: 'Enable the creation of CloudWatch metrics when this job runs.' (checked), 'Job observability metrics' (checked), 'Enable the creation of additional observability CloudWatch metrics' (checked), 'Continuous logging' (checked), 'Enable logs in CloudWatch.' (checked), 'Spark UI' (checked), and 'Write Spark UI logs to Amazon S3.' (checked).

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

Job runs (1/1) Info Last updated (UTC) May 23, 2024 at 14:51:43 View details Stop job run Table View Card View

Filter job runs by property

| Run status | Retries | Start time (Local) | End time (Local) | Duration | Capacit... | Worker type | Glue version |
|------------|---------|---------------------|---------------------|----------|------------|-------------|--------------|
| Failed | 0 | 05/20/2024 19:13:51 | 05/20/2024 19:15:08 | 1 m 8 s | 10 DPU | G.1X | 4.0 |

Run details | Input arguments (10) | Continuous logs | Run insights | Metrics | Spark UI

Error Category: RESOURCE_NOT_FOUND_ERROR; An error occurred while calling o147.pyWriteDynamicFrame. The specified bucket does not exist (Service: Amazon S3; Status Code: 404; Error Code: NoSuchBucket; Request ID: EQF44CKKCKGN07DZ; S3 Extended Request ID: fprn1TH2IJ5HjccOCgK6+iGU86gKu0MOuGaTQgy/RI+gFyiZE6VpWy87NLbzFPQkctKlcldayRWs=; Proxy: null)

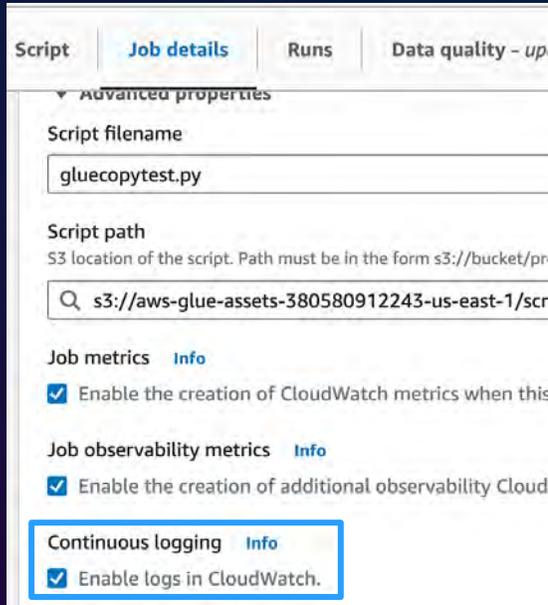
| | | | |
|---|------------------------|---|--------------------------|
| Job name | Start time (Local) | Glue version | Last modified on (Local) |
| Sample Job | 05/20/2024 19:13:51 | 4.0 | 05/20/2024 19:15:08 |
| Id | End time (Local) | Worker type | Log group name |
| jr_af803b5f8601f32520713579c91d5da21364f1db6cf1adca941e159f4ee1df4a | 05/20/2024 19:15:08 | G.1X | /aws-glue/jobs |
| Run status | Start-up time | Max capacity | Number of workers |
| Failed | 9 seconds | 10 DPU | 10 |
| Retry attempt number | Execution time | Execution class | Timeout |
| Initial run | 1 minute 8 seconds | Standard | 2880 minutes |
| Trigger name | Security configuration | Cloudwatch logs | |
| - | - | <ul style="list-style-type: none">All logsOutput logsError logs | |

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
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

The screenshot displays the AWS Serverless Spark UI interface. At the top, there are navigation tabs: Run details, Input arguments (10), Continuous logs, Run insights, Metrics, and Spark UI (which is highlighted with a blue box). Below the tabs is a breadcrumb trail: Jobs > Stages > Storage > Environment > Executors > SQL / DataFrame. The current job ID is jr_71bdcd6f180f2c23a7fa7821a... and the application is application UI.

Jobs (?)

User: spark
Total Uptime: 59.5 min
Scheduling Mode: FIFO
Completed Jobs: 2
Failed Jobs: 1

[Event Timeline](#)

Completed Jobs (2)

Page: 1 . Show 100 items in a page. Go

| Job Id | Description | Submitted | Duration | Stages: Succeeded/Total | Tasks (for all stages): Succeeded/Total |
|--------|--|-------------------------------------|-----------|-------------------------|---|
| 1 | fromRDD at DynamicFrame.scala:305 fromRDD at DynamicFrame.scala:305 | Monday, May 20, 2024 at 10:14:40 AM | 3.35 s | 1/1 | 1/1 |
| 0 | runJob at GlueParquetHadoopWriter.scala:176 runJob at GlueParquetHadoopWriter.scala:176 | Friday, April 5, 2024 at 2:03:28 AM | 59.18 min | 1/1 | 10/10 |

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

The screenshot displays the Spark UI interface with the following components:

- Navigation tabs:** Run details, Input arguments (10), Continuous logs, Run insights, Metrics, and Spark UI (selected).
- Log entries:** A list of log entries showing error messages. One entry is highlighted with a blue box: `com.amazonaws.services.s3.model.AmazonS3Exception: The specified bucket does not exist (Service: Amazon S3; Status Code: 404; Error Code: NoSuchBucket; Request ID: 0PJRXY0XR65RRRZ; S3 Extended Request ID: LvogumYXEipdMHOQYAFuL0jlou259Fzyzy7MH1jY4ShpX4vUkpGTEfsFS4i43sUImljfJtxdl+Q=; Proxy: null), S3`.
- Summary table:** A table with the following structure:

| Errors |
|---|
| TaskKilled |
| com.amazonaws.emr.hadoop.fs.shaded.com.amazonaws not exist (Service: Amazon S3; Status Code: 404; Error Code: NoSuchBucket; Request ID: 0PJRXY0XR65RRRZ; S3 Extended Request ID: LvogumYXEipdMHOQYAFuL0jlou259Fzyzy7MH1jY4ShpX4vUkpGTEfsFS4i43sUImljfJtxdl+Q=; Proxy: null), S3 |
| TaskKilled |
| com.amazonaws.emr.hadoop.fs.shaded.com.amazonaws not exist (Service: Amazon S3; Status Code: 404; Error Code: NoSuchBucket; Request ID: KpFNxvgu0gNTToY/Pz1wTxA45VSYcuNzyMI/VOTRMYOm... |
| TaskKilled |
| com.amazonaws.emr.hadoop.fs.shaded.com.amazonaws not exist (Service: Amazon S3; Status Code: 404; Error Code: NoSuchBucket; Request ID: ...) |

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

The screenshot displays the Spark UI interface for a job. At the top, there are tabs for 'Run details', 'Input arguments (10)', 'Continuous logs', 'Run insights', 'Metrics', and 'Spark UI'. Below the tabs, a progress bar indicates the job is 'Running'. A timeline shows the job's progress from 14:55 to 15:20 on Tuesday, October 24, 2023.

Below the timeline, there is a section for 'Completed Stages (2)'. The page number is set to 1. The table below shows the details of the completed stages:

| Stage Id | Description | Submitted | Duration | Tasks: Succeeded/Total | Input | Output | Shuffle Read | Shuffle Write |
|----------|--|--|-----------|------------------------|-------|--------|--------------|---------------|
| 1 | runJob at GlueParquetHadoopWriter.scala:176 <small>+details</small> | Wednesday, October 25, 2023 at 12:20:17 AM | 1.67 min | 20/20 | | | 7.01 GB | |
| 0 | rdd at DynamicFrame.scala:1970 <small>+details</small> | Tuesday, October 24, 2023 at 11:52:36 PM | 27.41 min | 1/1 | | | | 7.01 GB |

The 'Duration' column for stage 0 is highlighted with a red box, indicating a potential bottleneck. The page number is set to 1, and the table shows 1 page with 100 items per page.

[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

実行しているタスクのdetailを見る
[AWS Black Belt Online Seminar]猫でもわかる
detailsをクリックして
詳細を確認

| Index # | ID | Attempt | Status | Locality Level | Executor ID | Host | Launch Time | Duration | Scheduler Delay | Task Deserialization Time | GC Serialization Time | Result Serialization Time | Getting Result Time | Peak Execution Memory | Input Size / Records | Write Time | Shuffle Write Size / Records | Errors |
|---------|-----|---------|---------|----------------|-------------|--|---------------------|----------|-----------------|---------------------------|-----------------------|---------------------------|---------------------|-----------------------|----------------------|------------|------------------------------|---|
| 0 | 149 | 0 | FAILED | PROCESS_LOCAL | 9 | ip-172-32-30-47.us-east-2.compute.internal | 2021/05/26 09:50:46 | 0 ms | 0.7 min | 0 ms | 0 ms | 0 ms | 0 ms | 0.0 B | 0.0 B / 0 | 0.0 B / 0 | 0.0 B / 0 | ExecutorLostFailure (executor 9 exited caused by one of the running tasks) Reason: Container from a bad node: container_1621981679428_0004_01_000000 on host ip-172-32-30-47.us-east-2.compute.internal. Exit status: 137. (diagnostics: Container killed on request. Exit code is 137) |
| 0 | 216 | 1 | SUCCESS | PROCESS_LOCAL | 17 | ip-172-32-5-152.us-east-2.compute.internal | 2021/05/26 09:57:30 | 15 min | 7 ms | 0 ms | 0 ms | 0 ms | 0 ms | 0.0 B | 2.6 GB / 10319060 | 5 ms | 2.4 KB / 41 | |
| 1 | 150 | 0 | FAILED | PROCESS_LOCAL | 4 | ip-172-32-30-47.us-east-2.compute.internal | 2021/05/26 09:50:45 | 0 ms | 5.6 min | 0 ms | 0 ms | 0 ms | 0 ms | 0.0 B | 0.0 B / 0 | 0.0 B / 0 | 0.0 B / 0 | ExecutorLostFailure (executor 4 exited caused by one of the running tasks) Reason: Container from a bad node: container_1621981679428_0004_01_000000 on host ip-172-32-30-47.us-east-2.compute.internal. Exit status: 137. (diagnostics: Container killed on request. Exit code is 137) |
| 1 | 209 | 1 | SUCCESS | PROCESS_LOCAL | 18 | ip-172-32-20-51.us-east-2.compute.internal | 2021/05/26 09:58:21 | 13 min | 18 ms | 0 ms | 0 ms | 0 ms | 0 ms | 0.0 B | 0.0 B / 0 | 0.0 B / 0 | 0.0 B / 0 | |
| 2 | 151 | 0 | FAILED | PROCESS_LOCAL | 7 | ip-172-32-18-1.us-east-2.compute.internal | 2021/05/26 09:50:45 | 0 ms | 4.5 min | 0 ms | 0 ms | 0 ms | 0 ms | 0.0 B | 0.0 B / 0 | 0.0 B / 0 | 0.0 B / 0 | ExecutorLostFailure (executor 7 exited caused by one of the running tasks) Reason: Container from a bad node: container_1621981679428_0004_01_000000 on host ip-172-32-30-47.us-east-2.compute.internal. Exit status: 137. (diagnostics: Container killed on request. Exit code is 137) |
| 2 | 202 | 1 | SUCCESS | PROCESS_LOCAL | 16 | ip-172-32-30-51.us-east-2.compute.internal | 2021/05/26 09:50:18 | 17 min | 5 ms | 0 ms | 0 ms | 0 ms | 0 ms | 0.0 B | 0.0 B / 0 | 0.0 B / 0 | 0.0 B / 0 | |
| 2 | 152 | 0 | FAILED | PROCESS_LOCAL | 8 | ip-172-32-30-47.us-east-2.compute.internal | 2021/05/26 09:50:45 | 0 ms | 6.7 min | 0 ms | 0 ms | 0 ms | 0 ms | 0.0 B | 0.0 B / 0 | 0.0 B / 0 | 0.0 B / 0 | ExecutorLostFailure (executor 8 exited caused by one of the running tasks) Reason: Container from a bad node: container_1621981679428_0004_01_000013 on host ip-172-32-30-47.us-east-2.compute.internal. Exit status: 137. (diagnostics: Container killed on request. Exit code is 137) |
| 5 | 915 | 1 | SUCCESS | PROCESS_LOCAL | 17 | ip-172-32-5-152.us-east-2.compute.internal | 2021/05/26 09:57:30 | 19 min | 9 ms | 0 ms | 0 ms | 0 ms | 0 ms | 0.0 B | 14.7 MB / 100201 | 11 ms | 8.9 KB / 107 | |
| 6 | 153 | 0 | FAILED | PROCESS_LOCAL | 4 | ip-172-32-30-47.us-east-2.compute.internal | 2021/05/26 09:50:45 | 0 ms | 8.8 min | 0 ms | 0 ms | 0 ms | 0 ms | 0.0 B | 0.0 B / 0 | 0.0 B / 0 | 0.0 B / 0 | ExecutorLostFailure (executor 4 exited caused by one of the running tasks) Reason: Container from a bad node: container_1621981679428_0004_01_000000 on host ip-172-32-30-47.us-east-2.compute.internal. Exit status: 137. (diagnostics: Container killed on request. Exit code is 137) |

Failが起きているExecutorのログを見る。
AWS Glue ETLの場合はExecutor IDを確認してCloudWatchのLog groupsから確認する。

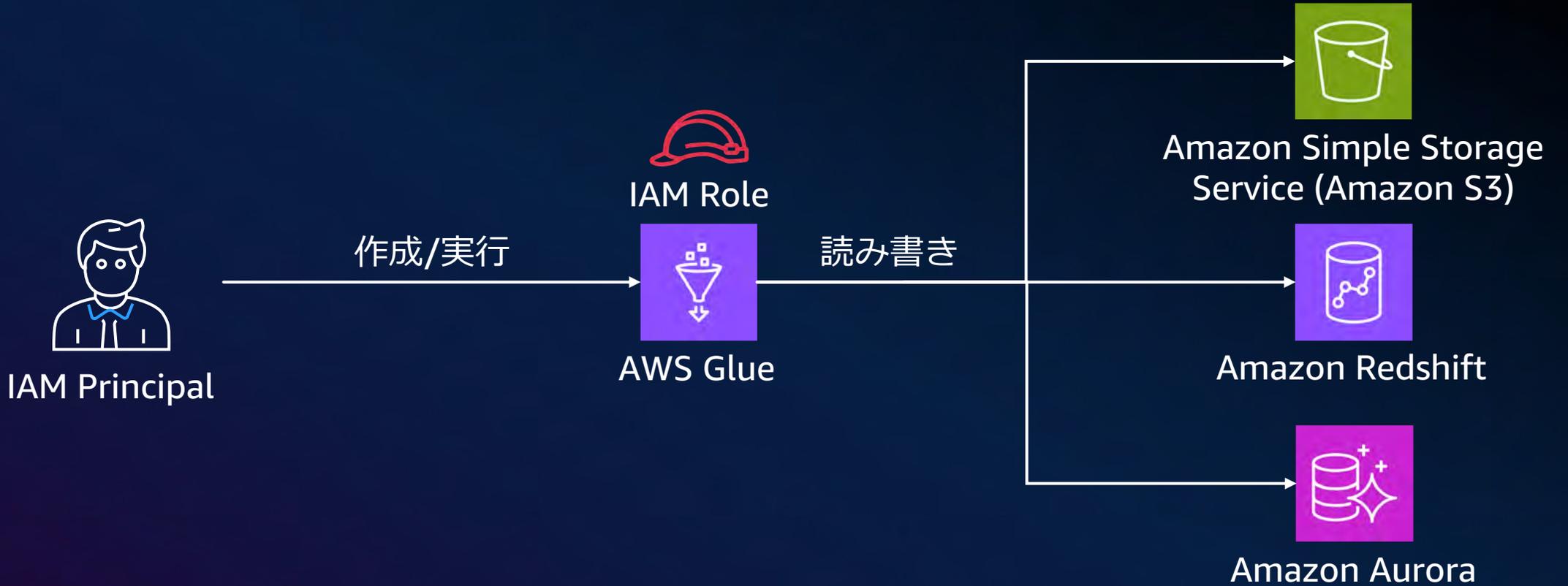
▶ 🔊 27:45 / 34:21 ・ ジョブの中身の... ⌨ ⚙️ YouTube 📺

[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

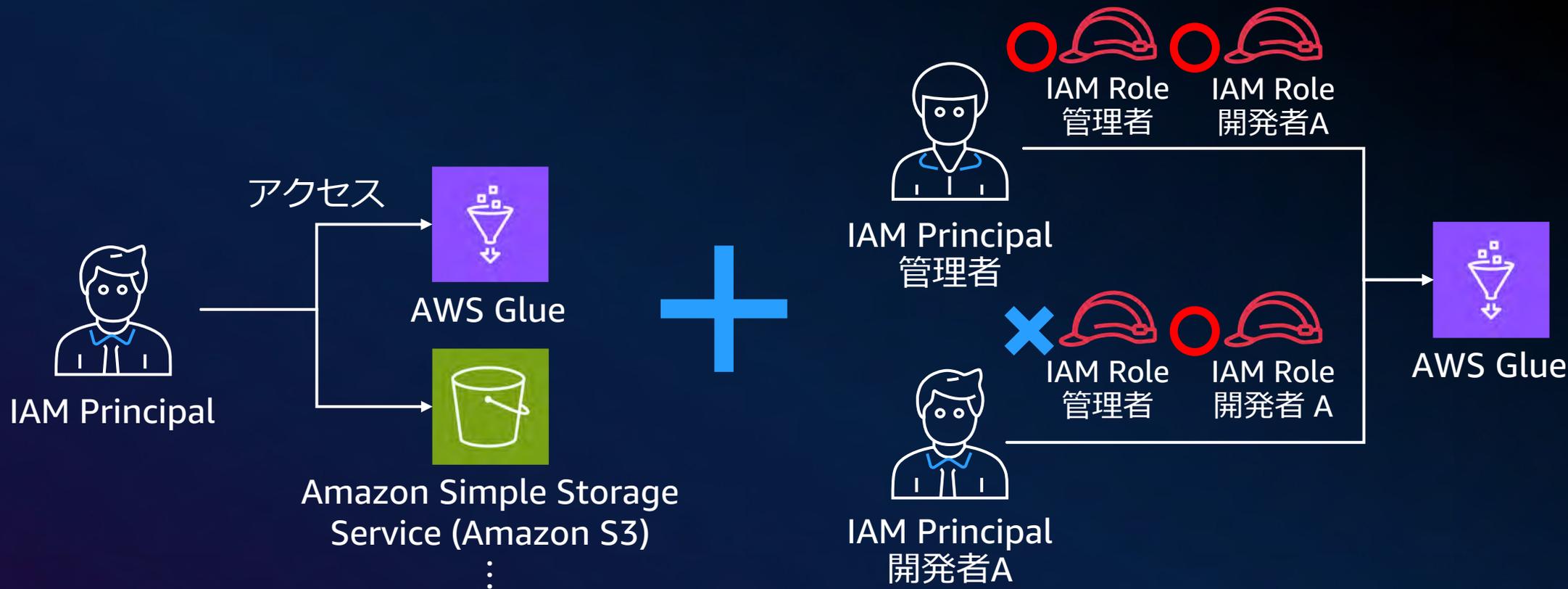


1. ジョブを作成/実行できる権限

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



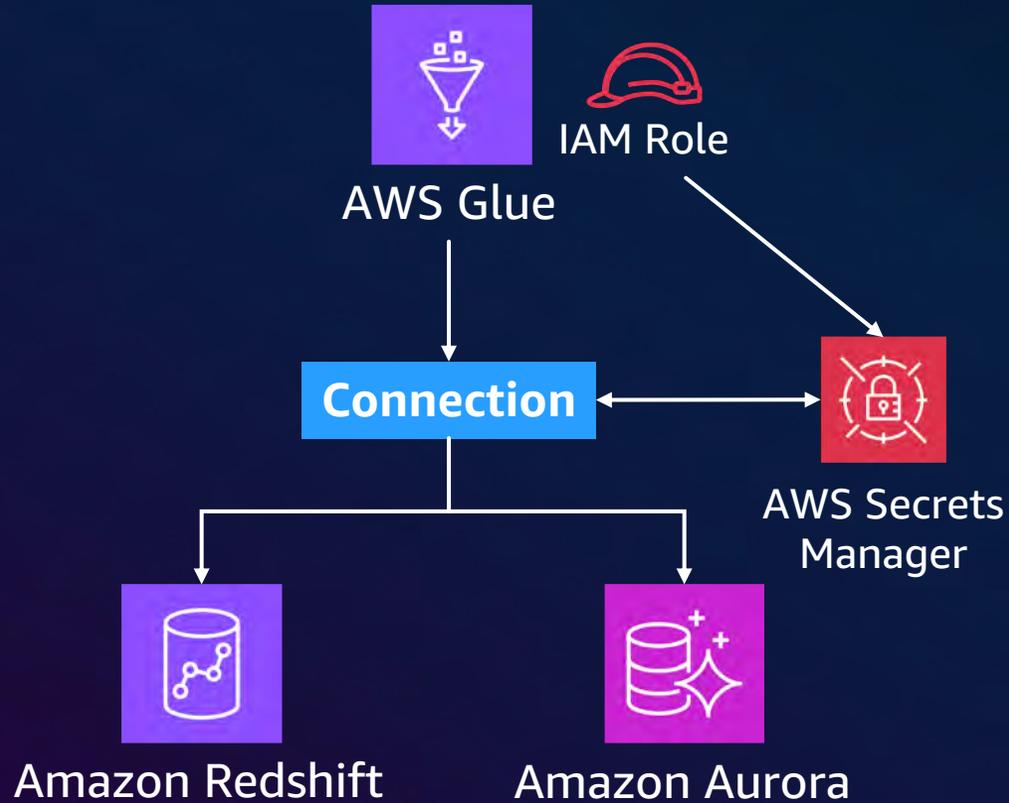
- AWS Glue Studio にアクセスできる
- AWS Glue の Script を S3 における
- AWS Glue ジョブを作成できる
- etc ...

- IAM Policy (**IAM:PassRole**) で制御可能

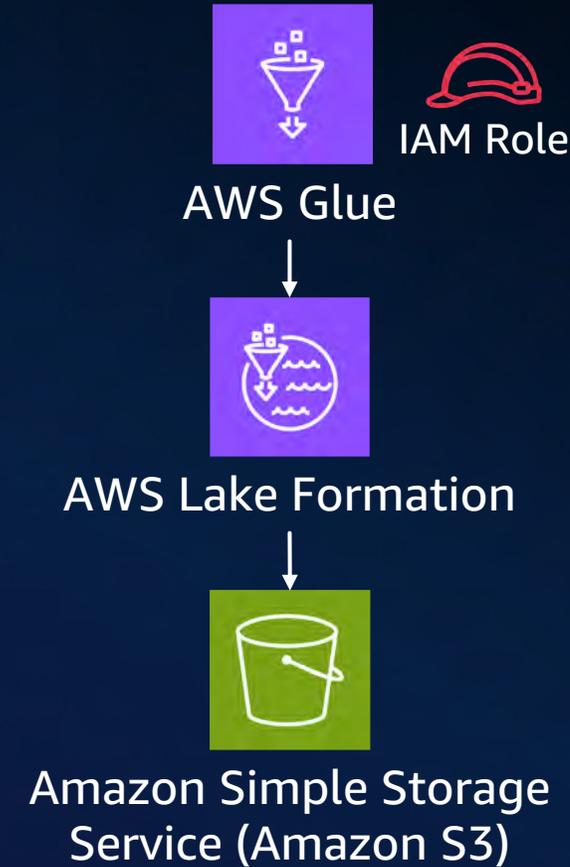
データソースへのアクセス制御

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

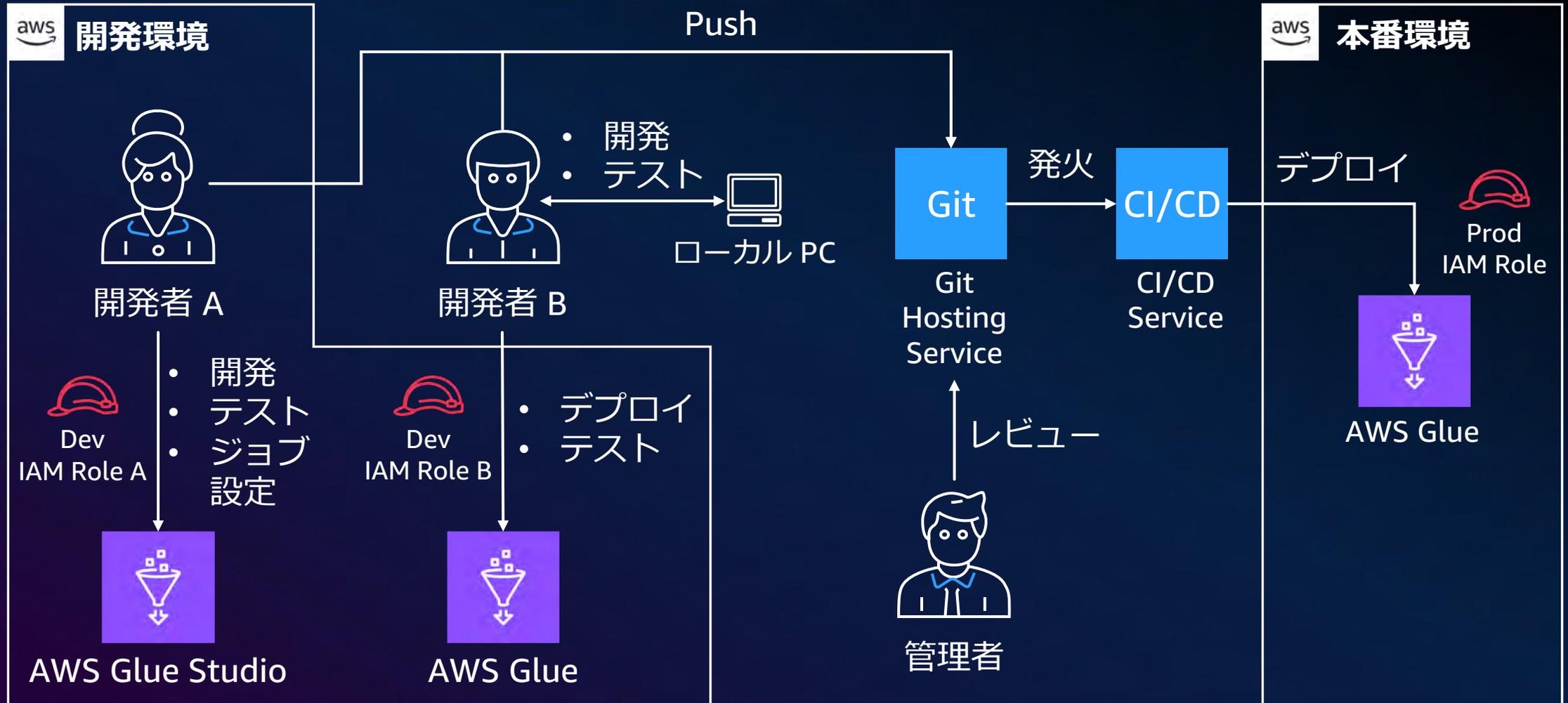


AWS LakeFormation



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

