aws DEV DAY
ONLINE JAPAN

20-22.10.2020

DEV DAY

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

E - 6

Glue version 2.
数十秒で起動するサーバーレス
Spark による分散処理開発の手引き

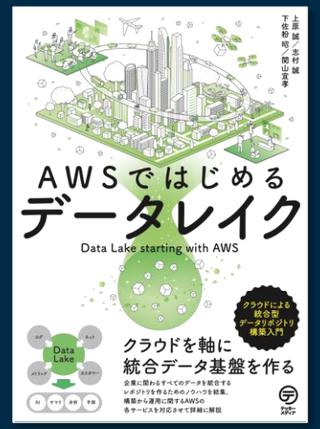
Noritaka Sekiyama (@moomindani)
Senior Big Data Architect
Amazon Web Services Japan / AWS Glue & Lake Formation

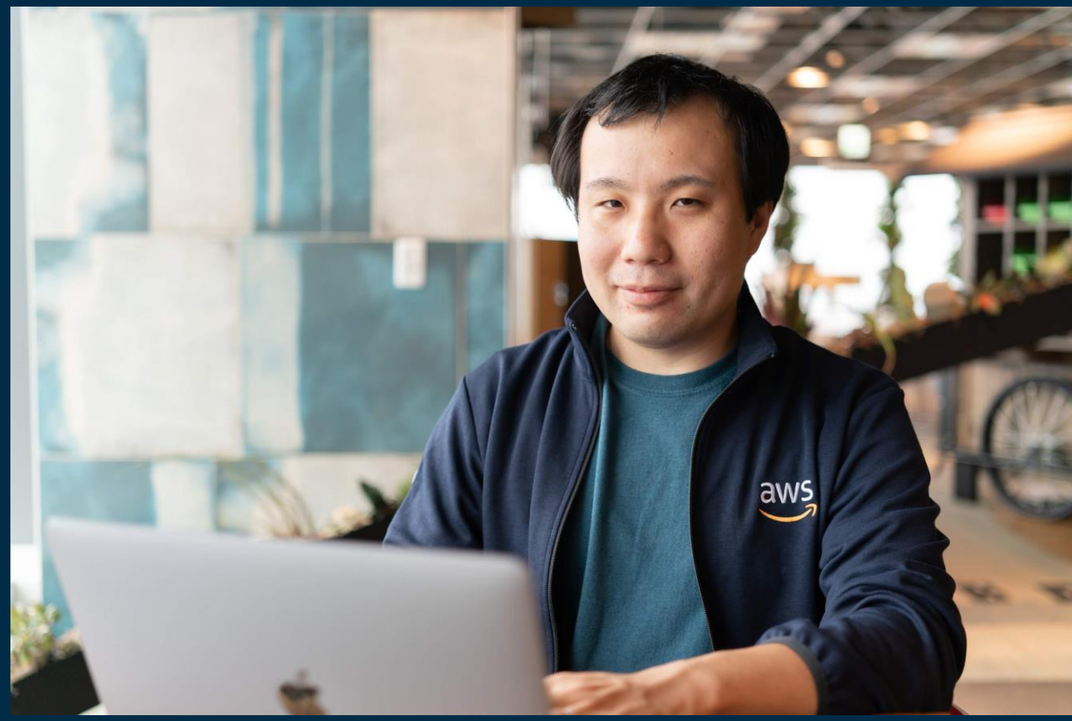
In Partnership with intel

自己紹介

関山 宜孝

Senior Big Data Architect
AWS Glue & Lake Formation





- 5年間 AWS サポートにて技術支援を担当
- 2019年からプロダクト開発チームにジョイン
- GlueとLake Formationに関する
  ユーザーに近い部分の開発を担当

aws Forum **NoritakaS-AWS**

@moomindani

moomindani

In Partnership with intel

アジェンダ

数十秒で起動する分散処理環境 AWS Glue とは

Glue でのコーディングの始め方

典型的なユースケースのスクリプティング

より高いパフォーマンスを実現する最適化手法

本番運用には欠かせないユニットテストとCI/CDプロセス

こんなことありませんか？

CSVファイルをJSONに変換したい

テキストファイルから特定の文字列を検索・集計したい

データベースからデータを抽出/変換してファイルに書き出したい

Amazon S3 上のデータを Amazon DynamoDB に移動したい

CSVファイルをJSONに変換したい

Standard

```
import json
import csv
import s3fs

json_list = []
json_data = {}
fs = s3fs.S3FileSystem(anon=True)

with fs.open('amazon-reviews-pds/tsv/sample_us.tsv', 'r') as f:
    for line in csv.DictReader(f, delimiter='¥t’):
        json_list.append(line)
        json_data["data"] = json_list

with open('sample_us.json', 'w') as f:
    json.dump(json_data, f)
```

Pandas

```
import pandas as pd
df = pd.read_csv("s3://amazon-reviews-pds/tsv/sample_us.tsv", sep='¥t')
df.to_json("sample_us.json")
```

データが小さければ簡単

CSVファイルをJSONに変換したい

データサイズ	圧縮形式	処理時間
15 KB	非圧縮	2 秒
442 MB	gzip	719 秒
2.7 GB	gzip	5336 秒

Macbook Pro 2019, Python 3.7.2
Pandas による CSV->JSON 変換
us-east-1 上の S3 バケット (Public Dataset) を使用

In Partnership with intel

データが大きいと・・・？

10GBのCSVファイルをJSONに変換したい

100GBのテキストファイルから特定の文字列を検索・集計したい

データベースから**1TBの**データを抽出/変換してファイルに書き出したい

Amazon S3 上の**10TBの**データを Amazon DynamoDB に移動したい

クラウド x 分散処理
の出番!

In Partnership with intel

aws DEV DAY
ONLINE JAPAN

数十秒で起動する分散処理環境 AWS Glue とは

In Partnership with intel

AWS Glue

サーバーレスな実行環境

高速なスタートアップ

豊富な ETL ライブラリ

クラウドに分散処理を
簡単にオフロード可能

In Partnership with intel

サーバーレス Spark 実行環境としての Glue

x

AWS Glue は (必ずしも ETL と関係なくても)

サーバーレス Apache Spark 実行環境

としても利用可能

In Partnership with intel

Glue version 2.0

高速なスタートアップ

In Partnership with intel

データを処理するサービスの使い分け

小規模処理

AWS Lambda

- 実行時間に最大15分間の制限あり
- 豊富なトリガーを持ち、S3に配置されたタイミングで逐次処理することも可能
- Python以外の言語も選択可能

Data < 500 MB

中規模処理

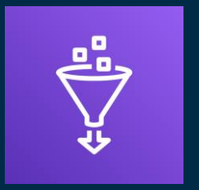
AWS Glue
Python Shell

データを加工するコードのみを実装
Python であれば NumPy, Pandas などのライブラリも利用可能

- 実行時間の制限なし
- Lambda に比べて利用できるメモリ量が多い（1GBまたは16GB）
- Athena、Redshift、EMR に対するSQLベースの処理も可能

Data: 500MB - 3 GB

大規模処理



AWS Glue
Spark

PySpark や Scalaで実装
必要なものはコードのみ

- 実行時間の制限なし
- 複数のワーカーで並列分散処理
- 数100GB以上の大量データ処理も可能
- DPU 設定によりスケーラビリティを調整可能

Data > 3GB

aws DEV DAY
ONLINE JAPAN

Glue でのコーディングの始め方

In Partnership with intel

選べる開発方法

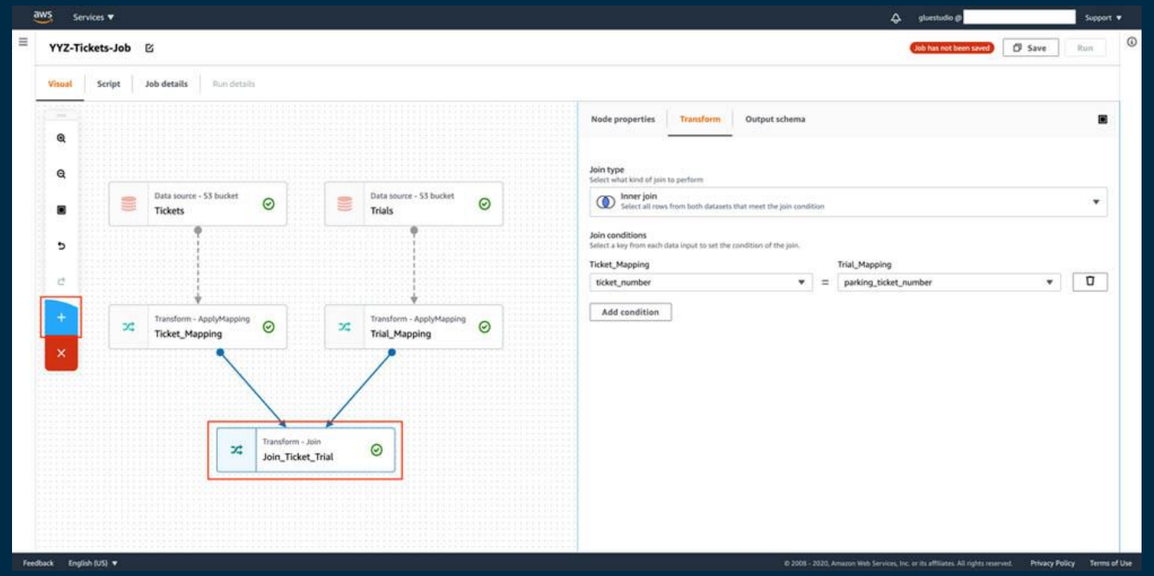
ローカルでの開発

IDE (IntelliJ IDEA, PyCharm, etc.)
Jupyter Notebook

おすすめ→ローカル開発用全部入り Docker イメージ
https://aws.amazon.com/blogs/big-data/developing-aws-glue-etl-jobs-locally-using-a-container/

クラウドでの開発

Glue ジョブコンソール
Glue Studio コンソール
SageMaker (Jupyter) Notebook
Glue Spark REPL



In Partnership with intel

aws DEV DAY
ONLINE JAPAN

Demo

In Partnership with intel

デモ：CSVファイルをJSONに変換したい

Amazon Customer Review Dataset

Amazon.com の商品レビューのデータセット
https://s3.amazonaws.com/amazon-reviews-pds/readme.html

TSV ファイルの総容量: GZIP 圧縮済み 32.2GB

Amazon Customer Reviews Dataset

Amazon Customer Reviews (a.k.a. Product Reviews) is one of Amazon's iconic products. In a period of over two decades since the first review in 1995, millions of Amazon customers have contributed over a hundred million reviews to express opinions and describe their experiences regarding products on the Amazon.com website. This makes Amazon Customer Reviews a rich source of information for academic researchers in the fields of Natural Language Processing (NLP), Information Retrieval (IR), and Machine Learning (ML), amongst others. Accordingly, we are releasing this data to further research in multiple disciplines related to understanding customer product experiences. Specifically, this dataset was constructed to represent a sample of customer evaluations and opinions, variation in the perception of a product across geographical regions, and promotional intent or bias in reviews.

Accessing the Amazon Customer Reviews Dataset

Over 130+ million customer reviews are available to researchers as part of this release. The data is available in TSV files in the `amazon-reviews-pds` S3 bucket in AWS US East Region. Each line in the data files corresponds to an individual review (tab delimited, with no quote and escape characters). Samples of the data are available in English and French; more details on the information in each column can be found here.

If you use the AWS Command Line Interface, you can list data in the bucket with the "ls" command:

```
aws s3 ls s3://amazon-reviews-pds/tsv/
```

To download data using the AWS Command Line Interface, you can use the "cp" command. For instance, the following command will copy the file named amazon_reviews_us_Camera_v1_00.tsv.gz to your local directory:

```
aws s3 cp s3://amazon-reviews-pds/tsv/amazon_reviews_us_Camera_v1_00.tsv.gz .
```

devday_demo.mov

aws DEV DAY
ONLINE JAPAN

典型的なユースケースのスクリプティング

In Partnership with intel

典型的なユースケースのスクリプティング

大量のCSVファイルをJSONに変換したい

大量のテキストファイルから特定の文字列を検索・集計したい

データベースから大量のデータを抽出/変換してファイルに書き出したい

Amazon S3 上の大規模データを Amazon DynamoDB に移動したい

大量のCSVファイルをJSONに変換したい

```
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.context import SparkContext

glue_context = GlueContext(SparkContext.getOrCreate())
spark = glue_context.spark_session
job = Job(glue_context)
job.init('devday’)

df = spark.read.format("csv").option("header", True).option("delimiter", "¥t").load("s3://amazon-reviews-pds/tsv/")
df.write.format("json").save("s3://sekiyama-bucket/devday/data/amazon-reviews-pds/json/")

job.commit()
```

テキストファイルから文字列を検索・集計したい

```
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.context import SparkContext
from pyspark.sql.functions import *

glue_context = GlueContext(SparkContext.getOrCreate())
spark = glue_context.spark_session
job = Job(glue_context)
job.init('devday')

df = spark.read.format("text").load("s3://amazon-reviews-pds/tsv/").withColumn("filepath", input_file_name())
df.filter(col("value").contains(”star")).groupBy("filepath").count().show()

job.commit()
```

データベースから大量のデータを抽出/変換してファイルに書き出したい

```
glue_context = GlueContext(SparkContext.getOrCreate())
spark = glue_context.spark_session
job = Job(glue_context)
job.init('devday’)

df = spark.read ¥
.format("jdbc") ¥
.option("url", "jdbc:postgresql:dbserver") ¥
.option("dbtable", "schema.tablename") ¥
.option("user", "username") ¥
.option("password", "password") ¥
.option("fetchsize", 1000) ¥
.option("numPartitions", 64) ¥
.option("partitionColumn", "primaryKey") ¥
.load()

df.write.format("csv").save("s3://sekiyama-bucket/devday/data/jdbc/csv/")
job.commit()
```

Amazon S3 上の大規模データを Amazon DynamoDB に移動したい

```
glue_context = GlueContext(SparkContext.getOrCreate())
spark = glue_context.spark_session
job = Job(glue_context)
job.init('devday')

df = spark.read.format("csv").load("s3://amazon-reviews-pds/tsv/")
dyf = DynamicFrame.fromDF(df, glue_context, "df2dyf")

glue_context.write_dynamic_frame_from_options(
    frame=dyf,
    connection_type="dynamodb",
    connection_options={
        "dynamodb.output.tableName": "test_sink",
        "dynamodb.throughput.write.percent": "1.0"
    }
)
job.commit()
```

aws DEV DAY
ONLINE JAPAN

より高いパフォーマンスを実現する最適化手法

In Partnership with intel

より高いパフォーマンスを実現する最適化手法

データを最適化する

クラスタのサイズをチューニングする

アプリケーション処理の並列度をチューニングする

In Partnership with

データを最適化する

Spark はデータ処理のためのフレームワーク

繰り返し処理する場合（Write once, Read many）
データを一度最適化しておくと、その後の処理効率が劇的に向上する

データフォーマット・圧縮形式

コンパクション

データの配置・パーティショニング

クラスタのサイズをチューニングする

Spark はクラスタ環境で動作するフレームワーク

クラスタのサイズをデータやアプリケーションの規模・特性に応じてチューニングすることで、パフォーマンスの向上を期待できる

特に、Ephemeral な処理の場合、データとアプリケーションの規模・特性に合ったサイズのクラスタを使うと効率が良い

ヒューリスティックアプローチ

データドリブン・アプローチ

処理の並列度をチューニングする (基本編)

Spark は分散処理を効率的に実行するフレームワーク

処理の並列度が不足すると、本来のパフォーマンスを期待できない

Spark は RDD や DataFrame などの形式でデータを取り扱う

RDD パーティション数 = Spark タスク数

repartition/coalesce でRDD パーティション数を調整し、
処理の並列度をチューニングできる

In Partnership with

処理の並列度をチューニングする (JDBC編)

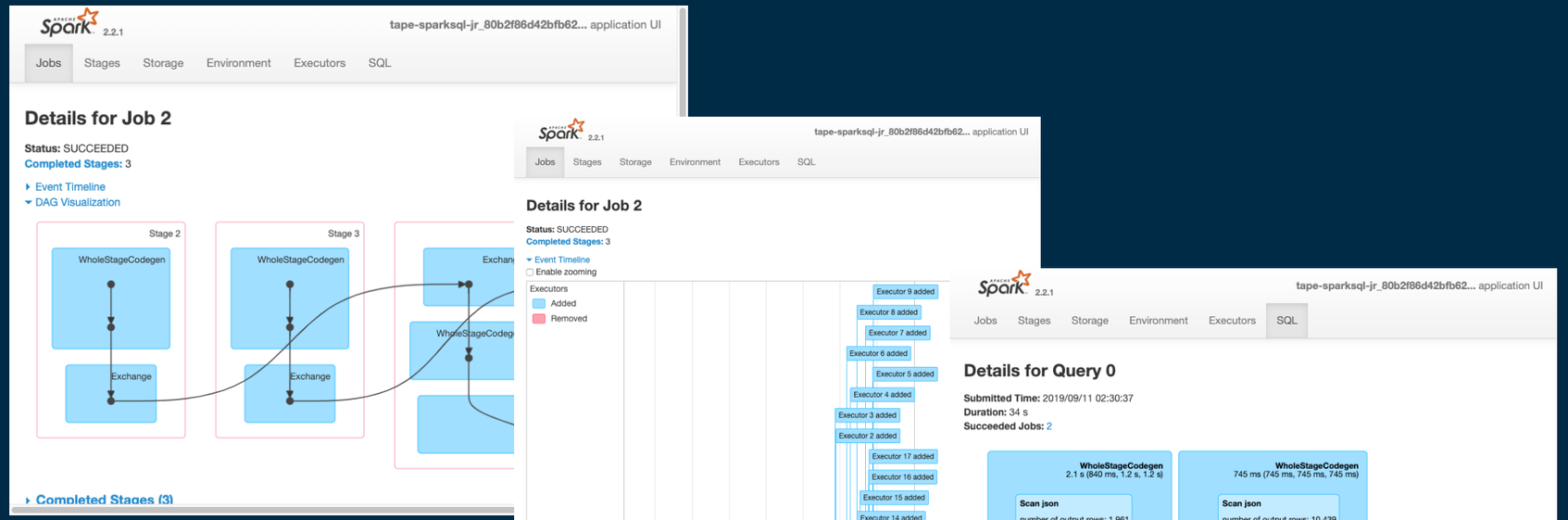
JDBC からの読み取りの場合、デフォルト設定では
Spark は単一のタスクでテーブル全体を読み取る

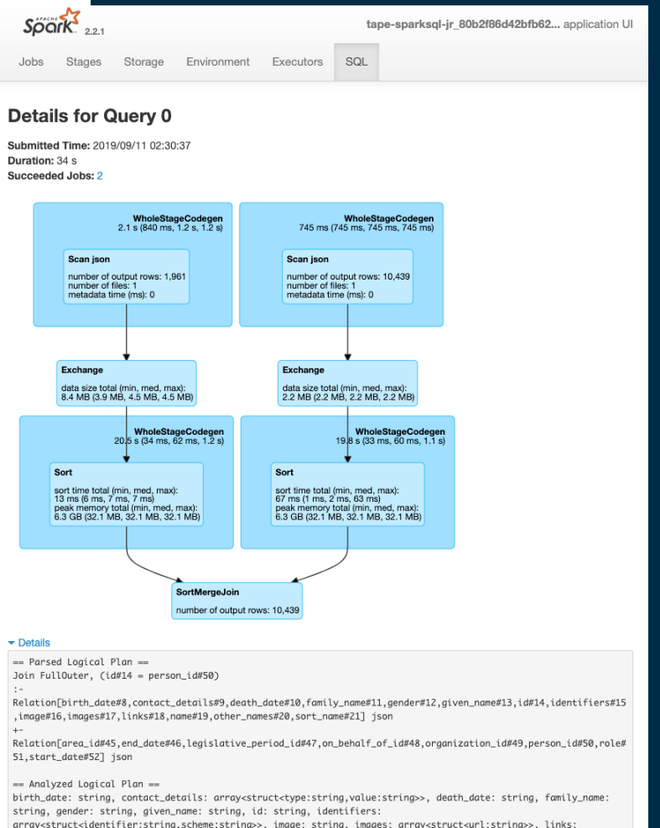
JDBC 読み取りの並列化にはパラメータの指定が必要

- numPartitions
- partitionColumn
- lowerBound
- upperBound

https://spark.apache.org/docs/latest/sql-data-sources-jdbc.html

Spark UI を用いたチューニング





In Partnership with intel

aws DEV DAY
ONLINE JAPAN

本番運用には欠かせない ユニットテストとCI/CDプロセス

In Partnership with intel

Glue ジョブの Unit Test (Python)

App

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

class GluePythonSampleTest:
    def __init__(self):
        args = getResolvedOptions(sys.argv, ['JOB_NAME'])
        self.context = GlueContext(SparkContext.getOrCreate())
        self.job = Job(self.context)
        self.job.init(args['JOB_NAME'], args)

    def run(self):
        dyf = read_csv(self.context, "s3://awsglue-datasets/examples/medicare/")
        write_parquet(self.context, dyf,
            "s3://sekiyama-bucket/glue/python-test-sample/output01/")
        self.job.commit()


def read_csv(glue_context, path):
    dynamicframe = glue_context.create_dynamic_frame.from_options(
        connection_type='s3',
        connection_options={
            'paths': [path],
            'recurse': True
        },
        format='csv'
    )
    return dynamicframe


def write_parquet(glue_context, dynamicframe, path):
    glue_context.write_dynamic_frame_from_options(
        frame=dynamicframe,
        connection_type='s3',
        connection_options={
            'path': path
        },
        format='parquet'
    )


if __name__ == '__main__':
    GluePythonSampleTest().run()
```

Test

```
import pytest
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from src import glue_python_test_sample
from awsglue.utils import getResolvedOptions

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

    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 = glue_python_test_sample.read_csv(glue_context,
        "s3://awsglue-datasets/examples/medicare/")
    assert dyf.toDF().count() == 163066
```

```
$ gluepytest
```


In Partnership with intel

Glue ジョブの Unit Test (Scala)

App

```scala
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object SampleGlueScalaTest {
  def readCsv(glueContext: GlueContext, path: String): DynamicFrame = {
    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "s3",
      options = JsonOptions(Map(
        "paths" -> Array(path),
        "recurse" -> true
      )),
      format = "csv",
      formatOptions = JsonOptions(Map(
        "withHeader" -> true
      ))
    ).getDynamicFrame()
    return dynamicFrame
  }

  def writeParquet(glueContext: GlueContext, dynamicFrame: DynamicFrame, path: String): Unit = {
    glueContext.getSinkWithFormat(
      connectionType = "s3",
      options = JsonOptions(Map(
        "path" -> path
      )),
      format = "parquet"
    ).writeDynamicFrame(dynamicFrame)
  }

  def main(sysArgs: Array[String]): Unit = {
    val sc: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(sc)
    val dyf = readCsv(glueContext, "s3://awsglue-datasets/examples/medicare/")
    writeParquet(glueContext, dyf, "s3://sekiyama-bucket/glue/scala-test-sample/output05/")
    sc.stop()
  }
}
```

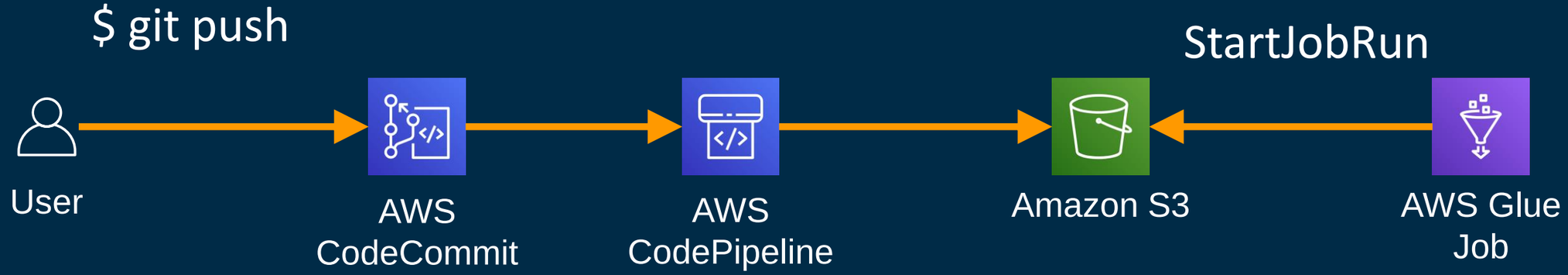
Test

```scala
import SampleGlueScalaTest.writeParquet
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.{SparkConf, SparkContext}
import org.scalatest.{BeforeAndAfterEach, FunSuite}

class SampleGlueScalaTestSpec extends FunSuite with BeforeAndAfterEach{
  var sc: SparkContext = _
  var glueContext: GlueContext = _

  override def beforeEach(): Unit = {
    val conf = new SparkConf
    conf.setMaster("local[*]")
    conf.setAppName("localrun")
    sc = new SparkContext(conf)
    glueContext = new GlueContext(sc)
  }

  test("counts should match with number of records in a csv file") {
    val dyf = SampleGlueScalaTest.readCsv(glueContext, "s3://awsglue-datasets/examples/medicare/")
    dyf.printSchema()
    assert(dyf.toDF().count() == 163065)
  }

  override def afterEach(): Unit = {
    sc.stop()
  }
}
```

```
$ mvn exec:java
-Dexec.mainClass="SampleGlueScalaTest"
-Dexec.args="--JOB-NAME glue-scalatest-sample"
```

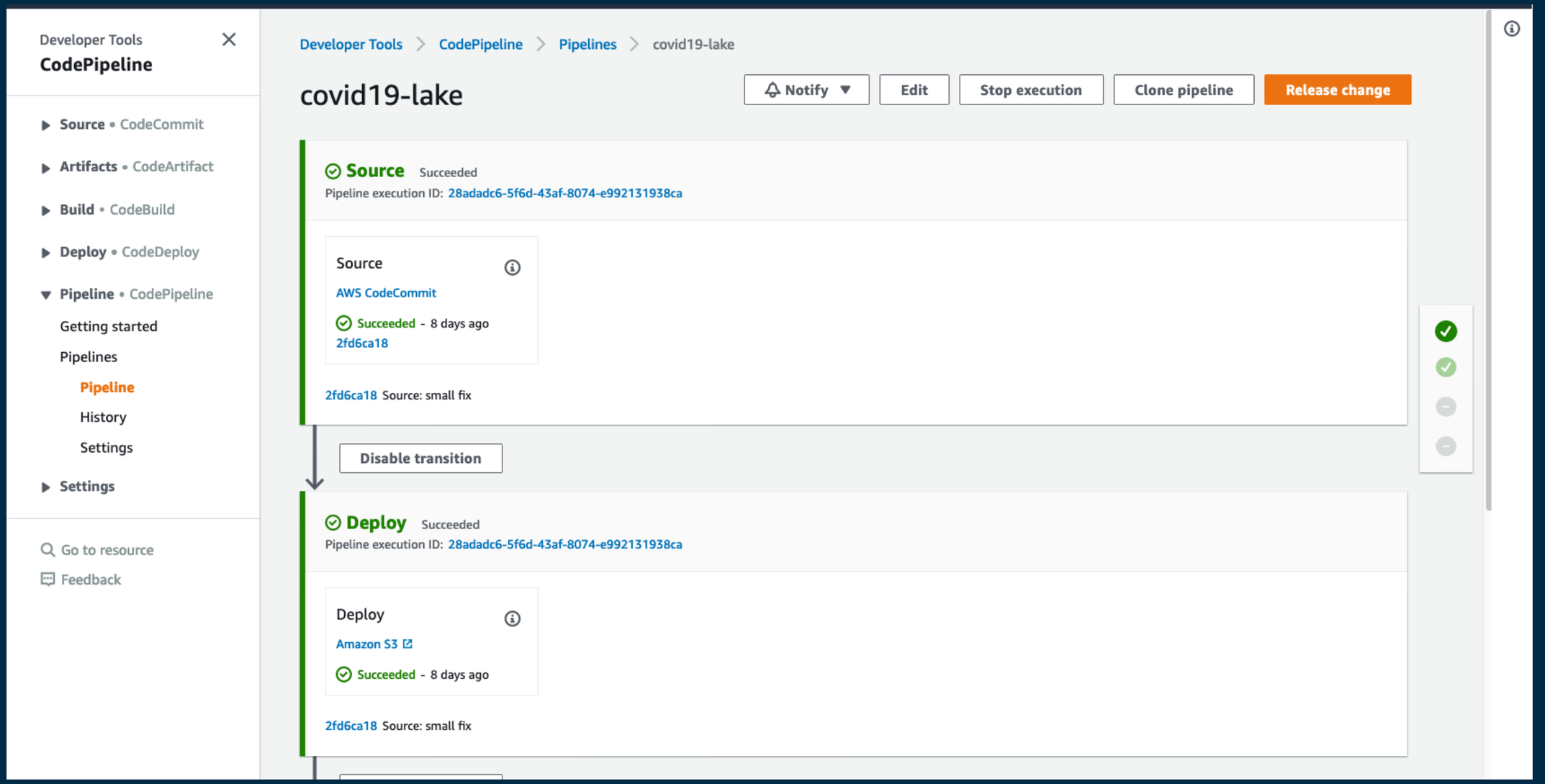

In Partnership with intel

Glue ジョブの CI/CD パイプライン



In Partnership with

Glue ジョブの CI/CD パイプラインの例



In Partnership with intel

おわりに

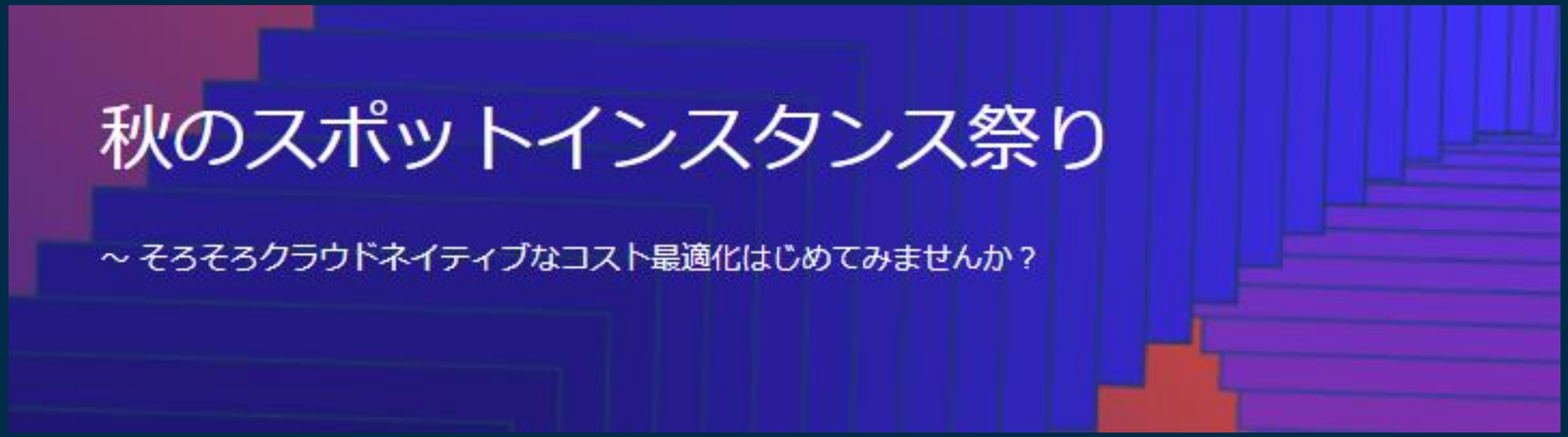
AWS Glue は数十秒で起動する分散処理環境

Glue でのコーディングはすぐ始められる

典型的なユースケースを簡単なスクリプティングで実現

簡単な最適化手法により高いパフォーマンスを実現

ユニットテストとCI/CDプロセスで本番環境にスムーズに導入



開催日時：2020 年 11 月 11 日（水曜日）　開演：13:00 ～ 16:15

申し込み URL: https://pages.awscloud.com/JAPAN-event-OE-Spot-instance-Matsuri-20201111-reg-event-LP.html

アジェンダ：
13:00 ～ 13:30　自動スケールから始める EC2 スポットインスタンス、講師：滝口　開資（アマゾン ウェブ サービス ジャパン株式会社　ソリューション アーキテクト）
13:30 ～ 14:00　実例から学ぶAWS Batch x スポットインスタンスによる大規模バッチ処理、講師：宮本 大輔（アマゾン ウェブ サービス ジャパン株式会社　ソリューション アーキテクト）
14:00 ～ 14:30　スポットインスタンス/インスタンスフリートを活用した EMR のコスト最適化、講師：川村　誠（アマゾン ウェブ サービス ジャパン株式会社　ソリューション アーキテクト）
14:45 ～ 15:15　お客様事例 1 がんばらないスポットインスタンス運用、講師：面白法人カヤック 藤原俊一郎様
15:15 ～ 15:45　お客様事例 2 機械学習ワークロードにおけるSpot & Batchの活用、講師：UNICORN 株式会社 井上碩様

aws

In Partnership with intel

aws DEV DAY ONLINE JAPAN

Thank you!

Noritaka Sekiyama
sekiyama@amazon.com

In Partnership with intel