
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

ANT304

Run serverless Spark workloads with AWS analytics

Noritaka Sekiyama

Principal Big Data Architect, AWS Glue
AWS

Vincent Gromakowski

Principal SA, Analytics
AWS

Agenda

Why Apache Spark on AWS

Use cases across AWS services for running Spark

Focus on AWS Glue and Amazon EMR

Apache Spark is popular

Support multiple workloads in one application

Accelerate the speed of analytics and processing

Highly fault tolerant

Developer friendly, supporting multiple programming languages

Why Spark on AWS

on

- Key enabler across multiple AWS services
- Provide Spark offering optimized for AWS
- Eliminate Spark deployment and update challenges
- Deployed on the most comprehensive cloud platform
- Supported by the largest partner community
- Have serverless deployment options
- Unique AWS features to improve resiliency, scale, and reduce costs

Spark on AWS comparison

	Amazon EMR Serverless	AWS Glue for Apache Spark
Use case	General purpose big data	Data integration
When to use	Run big data apps with open source frameworks like Spark and Hive	Build data lakes, DW, and data pipelines for different workloads (ETL/ELT/Streaming)
Personas	Data Analysts Data Engineers Data Scientists	Data Engineers ETL Developers Data Architects Data Stewards

Cost model

	 Amazon EMR Serverless	 AWS Glue for Apache Spark
vCPU	Per vCPU per hour	Per DPU per hour (4 vCPU, 16 GB RAM)
RAM	Per GB per hour	
Storage	Per storage GB per hour	-

Amazon EMR Serverless

Amazon EMR Serverless

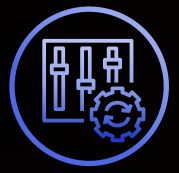
SERVERLESS GENERAL-PURPOSE BIG DATA PLATFORM

Run big data applications on Spark and Hive

60-day Spark release cadence

“Code first” approach

Petabyte scale



Bring your own customizations

Transactional support

aws

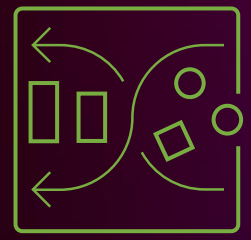
Use cases

Big data analytics

- Large-scale data processing and what-if analysis using statistical algorithms and predictive models
- Specific or latest Spark versions
- Custom environment setup

Interactive workloads

- Start immediately on the workers without any delays (pre-init capacity)



Lift and shift

- Migrate existing Hadoop applications from on-premise
- Migrate from EMR on EC2 or EMR on EKS to serverless

Benefits

Advanced configuration control and flexibility

Latest Spark versions

70–90 second start time

Dashboard

Studio: transactional-data-lake

Create workspace

Overview

Launch EMR Serverless applications and submit jobs

Launch EMR Serverless applications using the latest open-source frameworks and submit jobs. Learn more

Create application

Use fully-managed Jupyter notebooks

Create fully-managed Jupyter notebooks in seconds. Organize similar notebooks with workspaces, and use code repositories directly from notebooks. You can also install custom Python libraries or custom kernels. Learn more

Create workspace

Attach notebooks to clusters to run jobs

With EMR Studio you can create new clusters from pre-defined templates. You can also attach or detach Workspaces from clusters to run fully-managed Jupyter Notebooks. Learn more

Quickly diagnose and debug jobs

Debug applications by searching for a cluster and choosing an application UI. Learn more

Search clusters

Advanced Spark tuning

Override

- Spark properties defined in the EMR Serverless application (driver and executor conf)
- Any advanced Spark configuration including JVM tuning, offheap, etc. for performance

```
aws emr-serverless start-job-run \
    --application-id $APPLICATION_ID \
    --execution-role-arn $JOB_ROLE_ARN \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://'${S3_BUCKET}'/code/pyspark/extreme_weather.py",
            "sparkSubmitParameters": "--conf spark.driver.cores=1 --conf spark.driver.memory=3g
                                      --conf spark.executor.cores=4 --conf spark.executor.memory=3g
                                      --conf spark.executor.instances=10
                                      --conf spark.memory.fraction=0.5 --spark.memory.overhead=0.2"
        }
    }'
```

Python versions and dependencies

- Package a Python virtual env from a Docker image
- Optionally, use a different Python version
- Upload the archive to Amazon S3
- Load the package in the EMR Serverless job

```
ARG PYTHON_VERSION=3.10.6
FROM --platform=linux/amd64 amazonlinux:2 AS base
ARG PYTHON_VERSION

# Install Python 3.10.6 - Note that python 3.10 requires OpenSSL >= 1.1.1
RUN yum install -y gcc openssl11-devel bzip2-devel libffi-devel tar gzip wget make && \
    wget https://www.python.org/ftp/python/${PYTHON_VERSION}/Python-${PYTHON_VERSION}.tgz && \
    tar xzf Python-${PYTHON_VERSION}.tgz && \
    cd Python-${PYTHON_VERSION} && \
    ./configure --enable-optimizations && \
    make install

# Create our virtual environment
ENV VIRTUAL_ENV=/opt/venv
RUN python3 -m venv $VIRTUAL_ENV --copies
RUN cp -r /usr/local/lib/python3.10/* $VIRTUAL_ENV/lib/python3.10/

# Ensure our python3 executable references the virtual environment
ENV PATH="$VIRTUAL_ENV/bin:$PATH"

# Upgrade pip (good practice), install dependencies and venv-pack
RUN python3 -m pip install --upgrade pip && \
    python3 -m pip install venv-pack==0.2.0 great_expectations==0.15.6

# Package the env
RUN mkdir /output && \
    venv-pack -o /output/pyspark_${PYTHON_VERSION}.tar.gz --python-prefix /home/hadoop/environment

FROM scratch AS export
ARG PYTHON_VERSION
COPY --from=base /output/pyspark_${PYTHON_VERSION}.tar.gz /
```

```
aws emr-serverless start-job-run \
    --application-id $APPLICATION_ID \
    --execution-role-arn $JOB_ROLE_ARN \
    --job-driver '{
        "sparkSubmit": {
            "entryPoint": "s3://'${S3_BUCKET}'/code/pyspark/ge_profile.py",
            "sparkSubmitParameters": "--conf spark.archives=s3://'${S3_BUCKET}'/artifacts/pyspark/pyspark_ge.tar.gz#environment
                                      --conf spark.emr-serverless.driverEnv.PYSPARK_DRIVER_PYTHON=./environment/bin/python
                                      --conf spark.emr-serverless.driverEnv.PYSPARK_PYTHON=./environment/bin/python
                                      --conf spark.emr-serverless.executorEnv.PYSPARK_PYTHON=./environment/bin/python"
```

Applications (1) Info

Start application | Stop application | Action | Create application

Find applications | All Status | Last modified in 30 days

1

Name	Status	Application ID	Type	Release version	Last modified (UTC +2:00)
spark-serverless-emr	Stopped	00f4oslhkdkvcv09	Spark	emr-6.8.0	13 Oct 2022, 21:27

AWS Glue

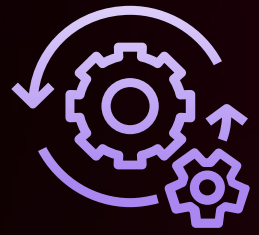
AWS Glue

Focus on data

Low maintenance serverless solution

Scale on demand

Avoid licensing cost and infrastructure idle time



Powerful open source engine

No lock in, support from wide innovative eco system

All in one

Support all your users personas and workloads

AWS Glue overview

SERVERLESS DATA INTEGRATION SERVICE

Connectors

Data warehouse

Data lakes

Marketplace

NoSQL

Streams

Author

Visual

Notebook

Built-in transformations

IDE

Operationalize

Workflow

Monitoring

Schedule

Data Management

Data Catalog

Data quality

Sensitive data detection

Engines

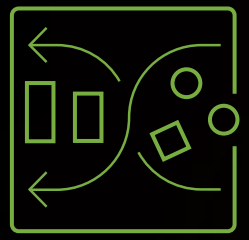
Serverless infrastructure

Choice of data integration engines

Compliance and security

Stream data processing

Use cases



Various data sources

- Built-in connectors (S3, Amazon Redshift, Amazon RDS/JDBC, Amazon DynamoDB, Amazon Kinesis/Kafka, MongoDB, etc.)
- Connector marketplace

Data management

- Incremental processing
- 250+ transformations
- Sensitive data detection
- Data lake format support (Hudi, Delta Lake, Iceberg)

Low-code job authoring

- AWS Glue Studio visual editor with visual transformations and data preview
- Automatic code generation

Benefits

Moderate level of configurations and flexibility

Many options to write and run code; IDE, notebook, GUI

5–10 seconds of start time

AWS Glue Studio

Jobs
Monitoring
Connectors
Sensitive data detection
What's new

Glue console
Glue catalog
Crawlers
Security configurations
Interactive sessions

Marketplace
Documentation

Jobs Info

Create job Info

Create

- (●) Visual with a source and target
  Start with a source, ApplyMapping transform, and target.
- ( ) Visual with a blank canvas
  Author using an interactive visual interface.
- ( ) Spark script editor
  Write or upload your own Spark code.
- ( ) Python Shell script editor
  Write or upload your own Python shell script.
- ( ) Jupyter Notebook
  Write your own code in a Jupyter Notebook for interactive development.

Source: Amazon S3 — JSON, CSV, or Parquet files stored in S3.

Target: Amazon S3 — S3 bucket by specifying a bucket path as the data target.

Your jobs (42) Info

Actions Run job

Find jobs

1 2 3 4 5

Job name	Type	Last modified	AWS Glue version
s3-to-redshift	Glue ETL	8/30/2022, 4:53:28 PM	3.0
j_emit_ended_event	Python shell	8/17/2022, 3:10:54 PM	

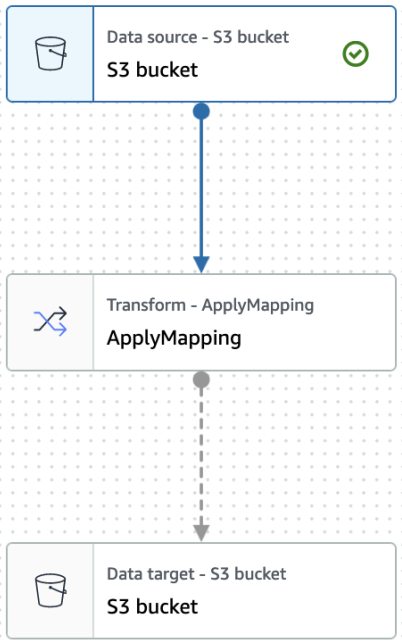
tickets

Last modified on 10/17/2022, 2:10:06 PM

Actions | Save | Run

Visual | Script | Job details | Runs | Schedules | Version Control

Source | Action | Target | Undo | Redo | Remove



Node properties | Data source properties - S3 | Output schema | Data preview

S3 source type Info

- Data Catalog table
- S3 location
  Choose a file or folder in an S3 bucket.

S3 URL

s3://aws-bigdata-blog/artifacts/gluestudio/ticket: | View | Browse S3

Recursive
Read files in all subdirectories.

Data format

Parquet

Infer schema

Advanced options

tickets

Last modified on 10/17/2022, 2:10:06 PM

Actions Save Run

Visual | Script | Job details | Runs | Schedules | Version Control

Script (Locked) Info

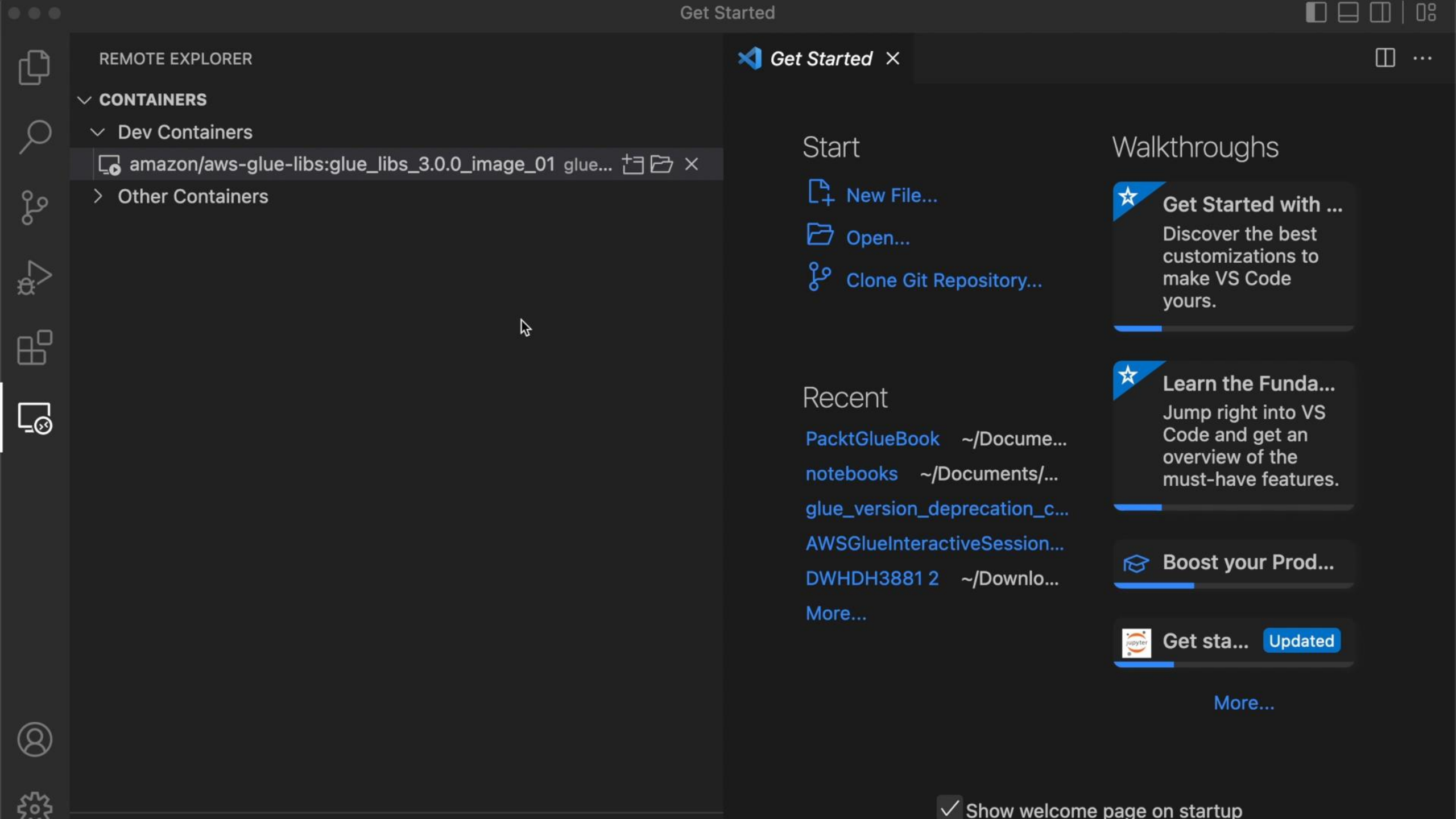
Generate classic script. Download script Edit script

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

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 bucket
S3bucket_node1 = glueContext.create_dynamic_frame.from_options(
    format_options={},
    connection_type="s3",
    format="parquet",
    connection_options={
        "paths": ["s3://aws-bigdata-blog/artifacts/gluestudio/tickets/"],
        "recurse": True,
    },
    transformation_ctx="S3bucket_node1",
)

# Script generated for node ApplyMapping
ApplyMapping_node2 = ApplyMapping.apply(
```


Get Started
REMOTE EXPLORER
CONTAINERS
Dev Containers
amazon/aws-glue-libs:glue_libs_3.0.0_image_01 glue...
Other Containers
Get Started
Start
New File...
Open...
Clone Git Repository...
Recent
PacktGlueBook ~/Docume...
notebooks ~/Documents/...
glue_version_deprecation_c...
AWSGlueInteractiveSession...
DWHDH3881 2 ~/Downlo...
More...
Walkthroughs
Get Started with ...
Discover the best customizations to make VS Code yours.
Learn the Funda...
Jump right into VS Code and get an overview of the must-have features.
Boost your Prod...
Get sta... Updated
More...
Show welcome page on startup

AWS Glue Studio

Jobs
Monitoring
Connectors
Sensitive data detection
What's new

Glue console
Glue catalog
Crawlers
Security configurations
Interactive sessions

Marketplace
Documentation

Analytics

AWS Glue Studio

Visually create job flows and monitor their performance

AWS Glue Studio is an easy-to-use graphical interface for creating, running, and monitoring AWS Glue extract, transform, and load (ETL) jobs.

Getting started

Create and manage jobs

Monitor job runs

Connect external data

Tutorials

What is AWS Glue Studio?

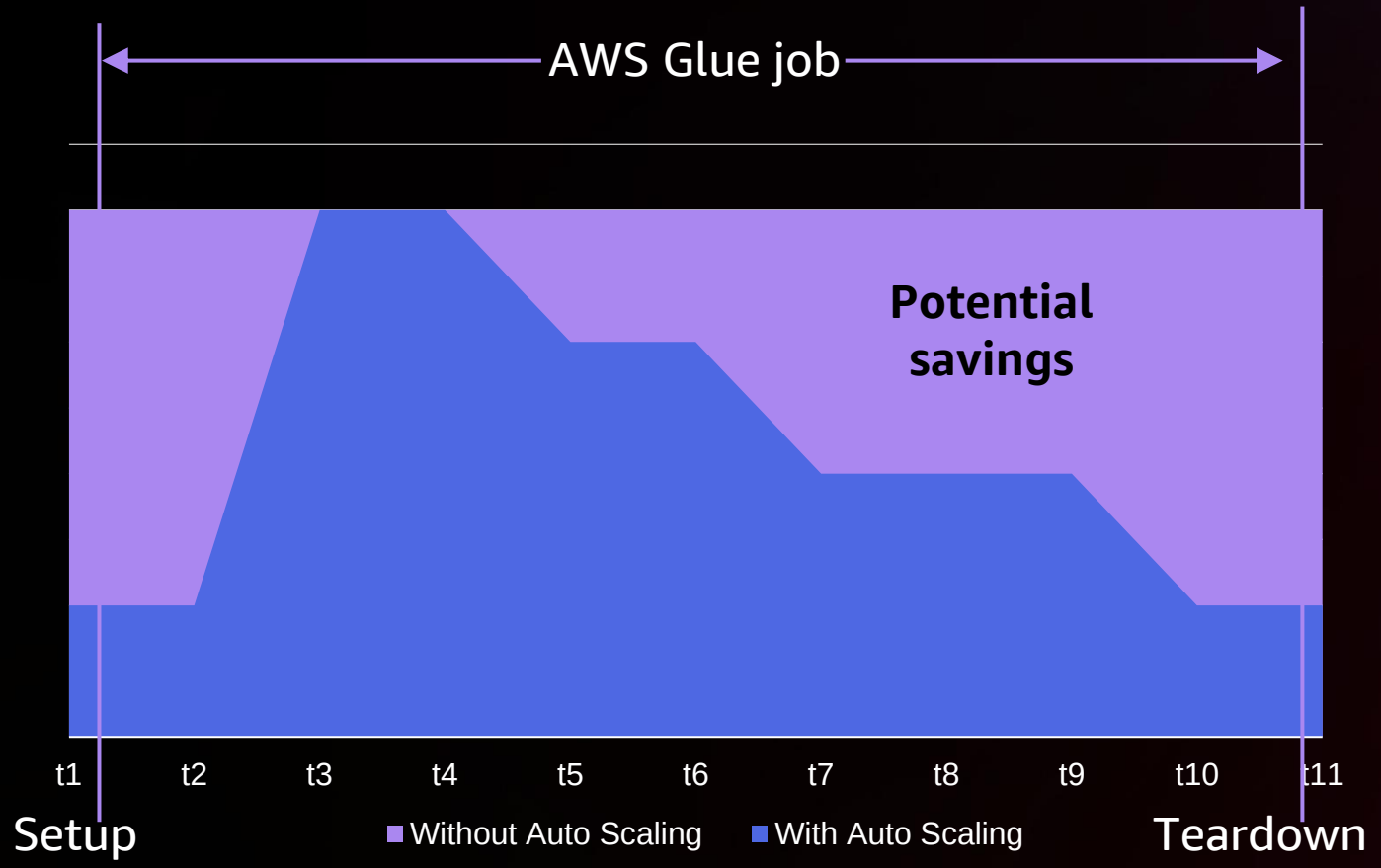
Getting started with Glue Studio

Video: Working with AWS Glue Studio, Part 1

Video: Working with AWS Glue Studio, Part 2

Video: Working with AWS Glue Studio, Part

AWS Glue Auto Scaling



Cost = f(compute)

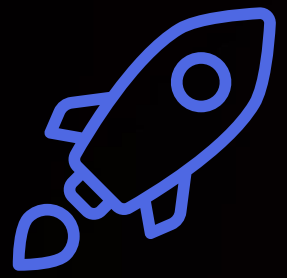
Job execution timeline

Reduce cost

Automate

Simplify capacity planning

AWS Glue Flex jobs



Standard
class

~1 min job start times
Predictable job latencies

Enables **micro-batching**
Latency-sensitive workloads

Flex
class

Up to
34% cost savings

Cost effective for **one-time**
data-load workloads

Extra library management in AWS Glue

Python

packages

pip integration
to install **additional libraries**

```
Key: --additional-python-modules
Value: boto3==1.24.89
```

or

```
Key: --additional-python-modules
Value: s3://path_to_your_whl_file
```

Java/Scala

JAR files

Extra JAR support
to install **additional libraries**

```
Key: --extra-jars
Value: s3://path_to_your_jar_file
```

Q&A

Thank you!

Noritaka Sekiyama
@moomindani
sekiyama@amazon.com

Vincent Gromakowski
gromav@amazon.com

Please complete the session survey in the **mobile app**