

Lab4. Running Spark Script on AWS Glue Studio

1. AWS Glue에서 spark-rds-demo.py 실행하기

1)[서비스] > [분석] > [AWS Glue] > [Data Integration and ETL] > [AWS Glue Studio] > [Jobs]

2)[Create job] 섹션에서 [Spark script editor] 선택 > [Create] 버튼 클릭

3)[Script] 탭에서 job이 초기화(job.init())한 다음 라인 즉, 16라인 다음에 다음과 같은 코드를 붙여 넣는다.

```
host_ = 'jdbc:mysql://' + " + '/newyork_taxi' #RDS DNS 엔드포인트 값 넣을 것
user_ = 'admin'
password_ = 'datalakemysql'
table_ = 'taxi_zone_lookup'

df = spark.read.format('jdbc').option('url', host_).option('driver', 'com.mysql.cj.jdbc.Driver') \
    .option('dbtable', table_).option('user', user_).option('password', password_).load()

df.show()
```

4)코드의 마지막 라인은 job.commit() 이다.

5)페이지 상단의 [Save] 버튼을 클릭한다.

6)[Job details] 탭으로 이동하여

-[Basic properties]

-[Name] : spark-rds-demo-job

-[IAM Role] : {계정}-glue-role

-[Type] : Spark

-[Glue version] : Glue 3.0 - Supports spark 3.1, Scala 2, Python 3

-[Language] : Python 3

-[Worker type] : G 1X

-[Requested number of workers] : 3

-[Job bookmark] : Disable

-[Number of retries] : 0

7)페이지 상단의 [Save] 버튼 클릭

8)[Run] 버튼 클릭

9)[Runs] 탭으로 이동

-방금 실행한 Job의 [Run status]가 "Succeeded"임을 확인한다.

-혹시 에러메시지가 다음과 같으면 보안 그룹의 소스가 0.0.0.0/0인지 확인한다.

"Caused by: com.mysql.cj.exceptions.CJCommunicationsException: Communications link failure"

-실행결과를 확인하기 위해 [Cloudwatch logs] > [Output logs]의 링크를 클릭하여 [CloudWatch] 페이지로 이동한다.

10)[로그] > [로그 그룹] 에 자동으로 생성된 로그 그룹으로 이동하게 된다.

11)[로그 스트림] 탭의 3개의 로그 링크 중에 접미사가 없는 제일 마지막 3번째 로그의 링크를 클릭한다.

12)여기서 출력결과를 확인한다.

13)다시 AWS Glue Studio의 해당 Job 페이지로 이동하여

14)[Version Control] 탭으로 이동하여

-[Git Configuration] 섹션에서

-[Git service] : GitHub

-[Personal access token] : 미리 생성한 토큰 값

-[Repository owner] : GitHub 계정

-[Repository configuration] 섹션에서

-[Repository] : 미리 생성한 repository

-[Branch] : main

15)페이지 상단의 [Actions] > [Version control] > [Push to repository] 클릭

16)[Push to repository] 창에서 [Confirm] 클릭

17)Push가 성공하면 "Last commit" 메시지와 CommitID가 보인다.

18)GitHub 페이지에서 Push된 결과를 확인한다.

2. Parameter Store에 새로운 패스워드 설정하기

1)AWS Console의 검색창에 "Parameter Store"라고 입력하여 검색한다.

2)검색결과 중 [Parameter Store]를 클릭하여 해당 서비스로 이동한다.

3)[AWS Systems Manager] 페이지로 들어왔다.

4)좌측 메뉴 중 [애플리케이션 관리] > [파라미터 스토어]를 클릭한다.

5)페이지 우측의 [파라미터 생성] 버튼을 클릭한다.

6)[파라미터 생성] 페이지에서

-[이름] : /henry/mysql/password

-[계층] : 표준

-[유형] : 보안 문자열

-[값] : datalakemysql

-[파라미터 생성] 버튼 클릭

3. AWS Glue Studio에서 spark-rds-demo1.py 실행하기

1)새로운 [Spark script editor]를 생성한다.

2)이름을 "spark-rds-demo1-job"이라고 입력한다.

3)[Script] 탭에서

4)먼저 7라인에 다음과 같이 코드를 넣는다.

```
import boto3
```

5)job이 초기화(job.init())한 다음 라인 코드 즉 18번째 라인에 다음과 같은 코드를 붙여 넣는다.

```

85 ssm = boto3.client('ssm')
86 ssm_parameter = ssm.get_parameter(Name='/henry/mysql/password', WithDecryption=True)
87 print(ssm_parameter)
88
89 spark = SparkSession.builder \
90     .config(conf=conf) \
91     .appName('Amazon Athena Program') \
92     .getOrCreate()
93
94 host_ = 'jdbc:mysql://' + " + '/newyork_taxi' #RDS DNS 엔드포인트 값 넣을 것
95 user_ = 'admin'
96 password_ = ssm_parameter['Parameter']['Value']
97 table_ = 'taxi_zone_lookup'
98
99 df = spark.read.format('jdbc').option('url', host_).option('driver', 'com.mysql.cj.jdbc.Driver') \
100     .option('dbtable', table_).option('user', user_).option('password', password_).load()
101
102 df.show()
103

```

6)그리고 이 코드의 마지막은 동일하게 `job.commit()`으로 한다.

7)페이지 상단의 [Save] 버튼을 클릭하여 저장한다.

8)[Job details]에서 위의 1번과 동일하게 설정한다.

9)[Save] 버튼 클릭

10)페이지 상단의 [Run] 버튼 클릭

11)[Runs] 탭으로 이동

12)1번 예제와 다르게 오류가 발생한다.

13)방금 실행한 Job의 [Run status]가 "Failed"임을 확인한다.

14)오류메시지는 다음과 같다. 오류메시지는 따로 복사해서 임시로 메모장같은 에디터에 붙여 넣는다.

```

ClientError: An error occurred (AccessDeniedException) when calling the GetParameter operation: User:
arn:aws:sts::789534828835:assumed-role/henry-glue-role/GlueJobRunnerSession is not authorized to perform:
ssm:GetParameter on resource: arn:aws:ssm:ap-northeast-2:789534828835:parameter/henry/mysql/password because no
identity-based policy allows the ssm:GetParameter action

```

15)오류의 이유는 현재의 Role로는 Parameter Store의 `sim:GetParameter` 권한이 없다는 것이다.

16)그래서 [IAM]의 [역할]에서 {계정}-glue-role을 찾고 해당 역할로 이동한다.

17)현재 {계정}-glue-role은 [AWSGlueServiceRole]과 [{계정}-glue-s3-access-policy]의 정책만 가지고 있기 때문에 필요한 정책을 추가해야 한다.

18)[권한 추가] > [인라인 정책 생성]을 클릭한다.

19)[정책 생성] 페이지에서, [JSON] 탭을 클릭하고 다음과 같이 편집한다.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowSSMGetParameter",
      "Effect": "Allow",
      "Action": ["ssm:GetParameter"],
      "Resource": ["arn:aws:ssm:ap-northeast-2:789534828835:parameter/*"]
    },
    {
      "Sid": "AllowDecryptSSM",
      "Effect": "Allow",
      "Action": ["kms:Decrypt"],
      "Resource": ["arn:aws:kms:ap-northeast-2:789534828835:key/420d08e6-5f64-4523-bfc9-da17a8deeba2"]
    }
  ]
}

```

20)[서비스] > [보안,자격 증명 및 규정 준수] > [Key Management Service] > [AWS 관리형 키] 으로 이동한다.

21)[AWS 관리형 키] 목록에서 [별칭]이 "aws/ssm"을 찾고 그 링크를 클릭한다.

22)해당 키 페이지의 [일반 구성] > [ARN] 값을 복사한다. <----위 코드의 "AllowDecryptSSM" > "Resource" 값에 복사

23)[정책 검토]를 클릭한다.

24)[정책 검토] 페이지에서,

-[이름] : {계정}-glue-ssm-access-role

-[정책 생성] 버튼 클릭

25)새로 [권한 정책] 목록에 방금 생성한 [정책]의 이름이 있음을 확인한다.

26){계정}-glue-role 에 정책을 추가했으면, 다시 [AWS Glue Studio]의 "spark-rds-demo1" Job으로 이동한다.

27)[Your jobs] 목록에서 "spark-rds-demo1" 의 링크를 클릭한다.

28)"spark-rds-demo1" Job 페이지에서 [Run] 버튼을 클릭하여 실행한다.

29)만일 에러(NameError: name 'SparkSession' is not defined가 발생하면 [Cloudwatch logs] > [Error logs] 링크를 클릭하여 보다 자세한 에러 메시지를 확인한다.

30)[로그 스트림] 중에 접미사가 없는 로그 스트림 링크를 클릭한다.

31)[로그 이벤트] 페이지에서 [메시지] 중 "INFO" 사이에 "ERROR" 이 있고 해당 이벤트를 확장해서 보면 자세한 에러메시지를 확인할 수 있다.

```

ERROR [main] glue.ProcessLauncher (Logging.scala:logError(73)): Error from Python:Traceback (most recent call last):
  File "/tmp/spark-rds-demo1.py", line 22, in <module>
    spark = SparkSession.builder \
NameError: name 'SparkSession' is not defined.

```

165 32) 즉, 22라인에서 SparkSession이 정의되어 있지 않다는 오류이다.
166 33) Job의 [Script] 탭으로 이동하여 해당 라인은 필요 없기 때문에 삭제하고 [Save] 버튼 클릭하고 다시 Run을 수행한다.
167 34) [Run status]가 "Succeeded"가 돼서 성공하면 [Cloudwatch logs] > "Output logs"의 링크를 클릭하여 [CloudWatch] > [로그 스트림] 목록
중에서 접미사가 없는 "로그 스트림" 링크를 클릭하고, [로그 이벤트]에서 출력 결과를 확인한다.

168
169
170
171
172
173
174
175

4. AWS Glue Studio에서 spark-rds-demo2.py 실행하기

- 1) 새로운 [Spark script editor]를 생성한다.
- 2) 이름을 "spark-rds-demo2-job"이라고 입력한다.

3) [Script] 탭에 다음과 같이 코드를 수정하고 [Save] 한다.

4) 먼저, 다음 코드를 7라인에 넣는다.

```
import pyspark.sql.functions as f
import boto3
```

5) 그 다음, job을 init 한 다음 라인인 18 라인 이후에 다음의 코드를 넣는다.

```
ssm = boto3.client('ssm')
ssm_parameter = ssm.get_parameter(Name='/henry/mysql/password', WithDecryption=True)

host_ = 'jdbc:mysql://' + " + '/newyork_taxi' #RDS DNS 엔드포인트 값 넣을 것
user_ = 'admin'
password_ = ssm_parameter['Parameter']['Value']
table_ = 'taxi_zone_lookup'

df = spark.read.format('jdbc').option('url', host_).option('driver', 'com.mysql.cj.jdbc.Driver') \
    .option('dbtable', table_).option('user', user_).option('password', password_) \
    .load().selectExpr('LocationID as location_id', 'borough', 'Zone as zone', 'service_zone')

# df.show()
taxi = spark.read.option('header', 'true').csv('s3a://henry-datalake-bucket/output/ym=2022-11')
taxi.show(truncate=False)

join_df = taxi.join(df, taxi.pu_location_id == df.location_id, 'left')
# join_df.show(truncate=False)

# join_df.limit(20).show(truncate=False)
join_df.limit(20).withColumn('pickup_datetime', f.to_timestamp('pickup_datetime', "yyyy-MM-dd'T'HH:mm:ss.SSSX")) \
    .withColumn('dropoff_datetime', f.to_timestamp('dropoff_datetime', "yyyy-MM-dd'T'HH:mm:ss.SSSX")) \
    .withColumn('time_diff', f.unix_timestamp('dropoff_datetime') - f.unix_timestamp('pickup_datetime')) \
    .show(truncate=False)
```

6) 마지막 라인은 job.commit() 이다.

7) [Save] 버튼을 클릭하고 [Job details] 탭으로 이동한다.

8) [Basic properties] 섹션에서 [IAM Role]을 {계정}-glue-role로 지정한다.

9) 나머지 설정 값은 위의 랩과 같다.

10) [Save] 버튼 클릭하고 [Run] 버튼을 클릭하여 실행한다.

11) 실행 후 [Runs] 탭으로 이동한다.

12) [Run status] 값이 "Failed" 이면 다음과 같은 오류 메시지가 나올 것이다.

```
An error occurred while calling o122.showString. Text '2022-11-15T05:28:58.000+09:00' could not be parsed, unparsed
text found at index 26
```

217
218
219
220
221
222

13) [Script] 탭으로 이동하여 해당 라인을 찾아 다시 수정한다.

```
"yyyy-MM-dd'T'HH:mm:ss.SSS'+09:00"
```

14) [Save] 버튼 클릭하고 다시 Run을 수행한다.

15) [Run status]가 "Succeeded"가 돼서 성공하면 [Cloudwatch logs] > "Output logs"의 링크를 클릭하여 [CloudWatch] > [로그 스트림] 목록
중에서 접미사가 없는 "로그 스트림" 링크를 클릭하고, [로그 이벤트]에서 출력 결과를 확인한다.

223
224
225
226
227
228
229
230

5. AWS Glue Studio에서 spark-rds-demo3.py 실행하기

1) 새로운 [Spark script editor]를 생성한다.

2) 이름을 "spark-rds-demo3-job"이라고 입력한다.

3) [Script] 탭에 다음과 같이 코드를 수정하고 [Save] 한다.

4) 다음 코드를 7라인에 넣는다.

```
import pyspark.sql.functions as f
import boto3
```

5) 그 다음, job을 init 한 다음 라인인 18 라인 이후에 다음의 코드를 넣는다.

```
ssm = boto3.client('ssm')
ssm_parameter = ssm.get_parameter(Name='/henry/mysql/password', WithDecryption=True)

host_ = 'jdbc:mysql://' + " + '/newyork_taxi' #RDS DNS 엔드포인트 값 넣을 것
user_ = 'admin'
password_ = ssm_parameter['Parameter']['Value']
table_ = 'taxi_zone_lookup'
write_table = 'taxi_output'

df = spark.read.format('jdbc').option('url', host_).option('driver', 'com.mysql.cj.jdbc.Driver') \
```

245

```

246         .option('dbtable', table_).option('user', user_).option('password', password_) \
247         .load().selectExpr('LocationID as location_id', 'borough', 'Zone as zone', 'service_zone')
248
249     # df.show()
250     taxi = spark.read.option('header', 'true').csv('s3a://henry-datalake-bucket/output/ym=2022-11')
251     taxi.show(truncate=False)
252
253     join_df = taxi.join(df, taxi.pu_location_id == df.location_id, 'left')
254     # join_df.show(truncate=False)
255
256     # join_df.limit(20).show(truncate=False)
257     join_df.limit(20).withColumn('pickup_datetime', f.to_timestamp('pickup_datetime',
258         "yyyy-MM-dd'T'HH:mm:ss.SSS'+09:00'")) \
259         .withColumn('dropoff_datetime', f.to_timestamp('dropoff_datetime', "yyyy-MM-dd'T'HH:mm:ss.SSS'+09:00'")) \
260         .withColumn('time_diff', f.unix_timestamp('dropoff_datetime') - f.unix_timestamp('pickup_datetime')) \
261         .repartition(1).write.format('jdbc').mode('overwrite') \
262         .option('url', host_).option('driver', 'com.mysql.cj.jdbc.Driver') \
263         .option('dbtable', write_table).option('user', user_).option('password', password_).save()
264
265 6)마지막 라인은 job.commit() 이다.
266 7)[Save] 버튼을 클릭하고 [Job details] 탭으로 이동한다.
267
268 8)[Basic properties] 섹션에서 [IAM Role]을 {계정}-glue-role로 지정한다.
269 9)나머지 설정 값은 위의 랩과 같다.
270 10)[Save] 버튼 클릭하고 [Run] 버튼을 클릭하여 실행한다.
271 11)실행 후 [Runs] 탭으로 이동한다.
272 12)[Run status]가 "Succeeded"가 돼서 성공하면 HeidiSQL(또는 MySQL Workbench)에서 "network_taxi" 데이터베이스의 "taxi_output"
    테이블이 생성된 것을 확인하고, 결과 데이터가 삽입된 것을 확인한다.

```